А.Е. Сланбекова,
Ш.К. Каменова, А.Б. Тогисова

ДЕРЕКТЕР ҚОРЫНЫҢ НЕГІЗДЕРІ

Қарағанды
2023

АКАДЕМИК Е.А. БӨКЕТОВ АТЫНДАҒЫ ҚАРАҒАНДЫ
УНИВЕРСИТЕТІ

А.Е. Сланбекова, Ш.К. Каменова, А.Б. Тогисова

ДЕРЕКТЕР ҚОРЫНЫҢ НЕГІЗДЕРІ

Оқу құралы

Қарағанды
2023

ӘОЖ 004 (075.8)
КБЖ 32.973 я73
С48

Академик Е.А. Бөкетов атындағы Қарағанды университеті
Ғылыми кеңесінің шешімімен баспаға ұсынылды
(30.05.2023 ж., Хаттама № 16,.)

Пікір жазғандар:

Р. Муратхан, Академик Е.А. Бөкетов атындағы Қарағанды университеті қолданбалы математика және информатика кафедрасының қауымдастырылған профессоры, PhD д-ры;
С.Ш. Кажикенова, Ә. Сағынов атындағы Қарағанды техникалық университеті жоғары математика кафедрасының меңгерушісі, профессор, техн. ғыл. д-ры;
А.Б. Касекеева, Л.Н. Гумилев атындағы Еуразия ұлттық университеті ақпараттық жүйелер кафедрасының доценті м.а., PhD д-ры

Сланбекова А.Е.

С48 Деректер қорының негіздері : оқу құралы / А.Е. Сланбекова, Ш.К. Каменова, А.Б. Тогисова. — Қарағанды : Академик Е.А. Бөкетов атындағы Қарағанды университетінің баспасы, 2023. — 262 б.

ISBN 978-601-362-139-5

Ұсынылып отырған оқу құралы жоғары оқу орындарының IT саласы бойынша білім алушыларға, оқытушыларға және жалпы деректер қорын басқару жүйесінің негіздерін үйренемін деушілерге арналған. Мұнда MySQL ДҚБЖ орнату және деректер қорларын құру, басқару, SQL командалары арқылы деректерді өңдеу қарастырылған. Бұл оқу құралын дайындау барысында MySQL Workbench бағдарламалық өнімінің 8.0 нұсқасы қолданылды. MySQL ортасын меңгеруді жеңілдету үшін иллюстрациялық материалдармен қамтылған.

ӘОЖ 004 (075.8)
КБЖ 32.973 я73

ISBN 978-601-362-139-5

Мазмұны

Кіріспе

Деректер қоры әдетте компьютерлік жүйеде электронды түрде сақталатын құрылымдық ақпараттың немесе деректердің реттелген жиынтығы болып табылады. Деректер қорын деректер қорын басқару жүйесі (ДҚБЖ) басқарады.

Деректер қорын басқару жүйесі – бұл деректерге қол жеткізетін, оларды құруға, өзгертуге және жоюға мүмкіндік беретін, деректердің қауіпсіздігін қамтамасыз ететін және т.б. мүмкіндік беретін тіл және бағдарламалық құралдар жиынтығы. Жалпы алғанда, ДҚБЖ – деректер қорын құруға және олардан ақпаратты өңдеуге мүмкіндік беретін жүйе. ДҚБЖ деректеріне бұл қатынасты арнайы тіл – SQL арқылы жүзеге асырады.

SQL – құрылымдық сұраныстар тілі, оның негізгі мақсаты деректер қорына ақпаратты оқу және жазудың қарапайым тәсілін қамтамасыз ету болып табылады. Бүгінгі күні қолданылған ДҚБЖ саны ондаған болып табылады. Ең көп қолданушы ДҚБЖ – Ms SQL Server, Oracle және MySQL. Оқу құралында біз MySQL ДҚБЖ қолданамыз. Оның көмегімен MySQL кестелерінде сақталған ақпаратқа қол жеткізе алады.

MySQL – реляциялық деректер қорын басқару жүйесі. Бүгінгі күні ол ең танымал деректер қорын басқару жүйелерінің бірі болып табылады. Деректер қоры дегеніміз белгілі бір мәліметтердің жиынтығы. Бізге осындай мәліметтерді жазуға, оқуға және оны өңдеуге мүмкіндік беретін бағдарлама керек, міне осындай бағдарламаларды деректер қорын басқару жүйесі деп атайды.

MySQL клиент-серверлік архитектураға ие MySQL AB швед компаниясы әзірлеген жүйе болып табылады. MySQL кроссплатформалық жүйе, яғни қазіргі қолданысқа ие операциялық жүйелерде орнатуға болады. MySQL деректер қорын басқару жүйесінің ақылы(коммерциялық лицензияланған)және тегін нұсқалары таратылады.

MySQL-де көптеген бағдарламалық интерфейстер (API) бар, соның арқасында C/C++, Java, Perl, PHP, Python, .NET және Visual Studio көмегімен жасалған қосымшалар MySQL деректер қорына қосыла алады.

MySQL AB компаниясы 1995 жылы MySQL-дің бірінші нұсқасын (1.0) шығарды. 2008 жылы Sun Microsystems компаниясы MySQL AB жүйесін сатып алды. Ал Oracle 2010 жылы Sun Microsystems компаниясын сатып алып, MySQL-ді өзінің ДҚБЖ тізіміне қосты. MySQL қазіргі уақытта Oracle демеушілігімен дамып келеді. Соңғы MySQL нұсқасы 8.0 болып табылады. MySQL тамаша техникалық сипаттамаларға ие: көп ағынды, өнімділігі жоғары, масштабталуы бар, көп пайдаланушыға арналған

MySQL клиент-сервер жүйесі болып табылады, деректер қорынан әртүрлі есептеуіш машиналарды, сонымен қатар бірнеше клиенттік бағдарлама және кітапхана, әкімшілік және бағдарламалық интерфейстардың кең спектрін ұсынатын SQL-серверін қамтиды. MySQL-мен жұмыс істеу үшін сервермен қалай байланысу керектігін білуіңіз керек, яғни сұраныстарды құру үшін SQL тілін пайдалану қажет. Сондықтан бұл оқу құралында сұраныстарды жазу үшін SQL сұраныстар тілін қолдануға ерекше назар аударылады.

SQL (құрылымдық сұраныс тілі) – реляциялық деректер қорында ақпаратты сақтауға және өңдеуге арналған сұраныс тілі. 1970 жылдары британдық ғалым Эдгар Коддтың зерттеу жұмысы реляциялық деректер қоры теориясының негізін қалады. Реляциялық деректер қоры ақпаратты кестелік түрде сақтайды, жолдар мен бағандар әртүрлі деректер атрибуттарын және деректер мәндері арасындағы әртүрлі байланыстарды білдіреді. SQL командаларын деректер қорында ақпаратты сақтау, жаңарту, жою, іздеу және алу үшін пайдалануға болады. Сондай-ақ, SQL-ді деректер қорларының өнімділігін сақтау және оңтайландыру үшін пайдалануға болады. SQL тілін 1974 жылы Дональд Чемберлин мен Рэймонд Бойс негізін қалады.

Қазіргі уақытта деректер қорларында мыңға жуық кестелер және миллиардтаған жазбалар сақталады. Осындай үлкен көлемдегі деректерді өңдеу үшін арнайы құралдар пайдаланылады. MySQL ДҚБЖ басқару үшін арналған графикалық интерфейсі бар MySQL Workbench құралы қолданылады.

MySQL Workbench – деректер қорын жобалаудың және басқарудың ең қуатты визуалды құралы.

Оның көмегімен кестелер мен басқа объектілерді құруға және өңдеуге, пайдаланушылардың қол жетімділігін басқаруға және деректер қорын толығымен басқаруға болады. Workbench-тің басты ерекшеліктерінің бірі – визуалды модельдеу. Бұл барлық кестелер мен олардың арасындағы байланыстарды көрнекі түрде көруге мүмкіндік береді. Бұл деректер қорларын жобалауды әлдеқайда жеңілдетеді.

MySQL Workbench келесідей мүмкіндіктерге ие: деректер қорларының графикалық моделін құру, деректер қорларын құру және басқару, серверге бірден жіберуге және кесте түрінде жауап алуға мүмкіндік беретін ыңғайлы SQL сұраныстар редакторы, кестедегі деректерді визуалды режимде өңдеу мүмкіндігі және деректер қорының сақтық көшірмесін жасау және қалпына келтіру.

Oracle компаниясы MySQL Workbench құралының үш нұсқасын ұсынады: Community Edition, Standard Edition және Enterprise Edition. Бұл оқу құралын дайындау барысында GNU GPL лицензиясымен тегін таратылатын Community Edition нұсқасы қолданылды.

Оқу құралы алты тараудан тұрады. Бірінші тарауда MySQL Workbench құралын орнату нұсқаулығы және MySQL Command Line Client консольді клиенті арқылы деректер қорымен жұмыс істеу мысалдары берілген.

Екінші тарауда деректер қорлары мен кестелерді құру және жою командалары, MySQL деректер қоры кестелерінің бағандары үшін пайдалануға болатын деректер түрлері, баған және кесте атрибуттары, ішкі және сыртқы кілттер қарастырылады.

Үшінші тарауда деректермен орындалатын негізгі операциялар, деректерді қосу, деректерді таңдау, деректерді сүзгілеу, деректерді жаңарту, деректерді жою командалары жайлы айтылған.

Төртінші тарауда сұраныстар, бірегей мәндерді таңдау, сұрыптау операторлары, агрегатты функциялар, топтау, SQL негізгі командаларындағы ішкі сұраныстар жайлы баяндалады.

Бесінші тарауда бірнеше кестелерді біріктіру, JOIN операторларының түрлерін қолдану тәсілдері көрсетілген.

Алтыншы тарауда MySQL-дегі built-in функциялары, сақталатын процедуралар мен функциялар және триггер туралы ақпарат беріледі.

Оқу құралы деректер қорын құру және басқаруға арналған тапсырмалар және білімді бекітуге арналған тест тапсырмаларымен қамтылған.

1 тарау. MySQL Workbench ортасын орнату

1.1 MySQL Workbench ортасын орнату

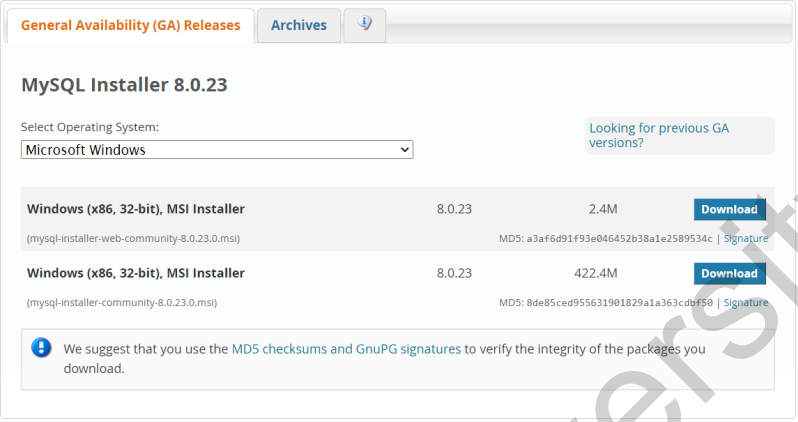
Кез келген басқа деректер қорын басқару жүйелері сияқты MySQL күрделі бағдарламалық жасақтама болып табылады. Оның өнімділігі, орнықтылығы және қауіпсіздігі дұрыс орнату мен конфигурацияға байланысты. MySQL ДҚБЖ басқару үшін арналған визуалды интерфейсі бар MySQL Workbench құралын пайданыламыз. MySQL Workbench серверді орнату, пайдаланушыны басқару, деректер қорының сақтық көшірмесін жасау, деректерді модельдеу, SQL сұраныстарын орындау және т.б. үшін кешенді басқару құралдарын ұсынады. MySQL Workbench кроссплатформалық болып табылғандықтан Windows, Linux және MacOS жүйелеріне орнатуға болады. Бұл тарау MySQL Workbench-ті орнатуға және бастапқы конфигурациялауға арналған.

1-кесте

MySQL Workbench-ті орнатудың ең төменгі (минималды) аппараттық талаптары

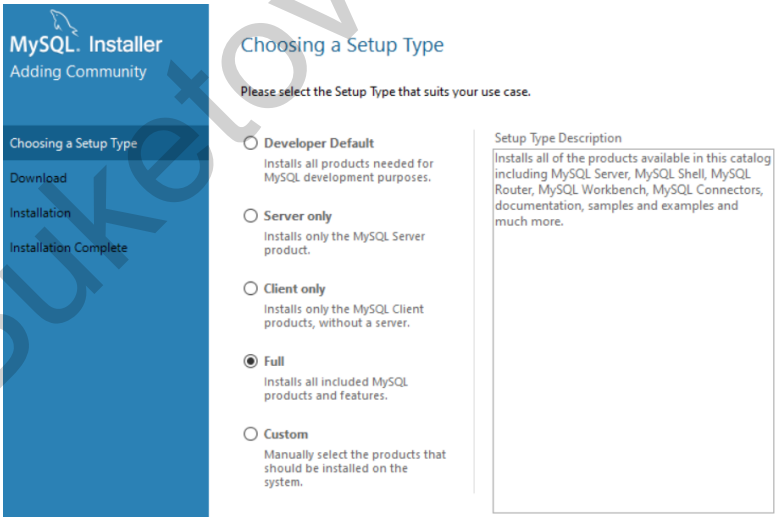
Аппаратық жабдықтама	Ең кемінде	Ұсынылады
CPU	64bit x86	Multi Core 64bit x86
Оперативті жады	4 GB	8 GB немесе жоғары
монитор	1024×768	1920×1200 немесе жоғары

MySQL деректер қорын басқару жүйесінің ақылы (коммерциялық лицензияланған) және тегін нұсқалары таратылады. MySQL орнату файлын (422.4M) ресми сайттан (https://dev.mysql.com/downloads/windows/installer/8.0.html) жүктеп алуға болады.



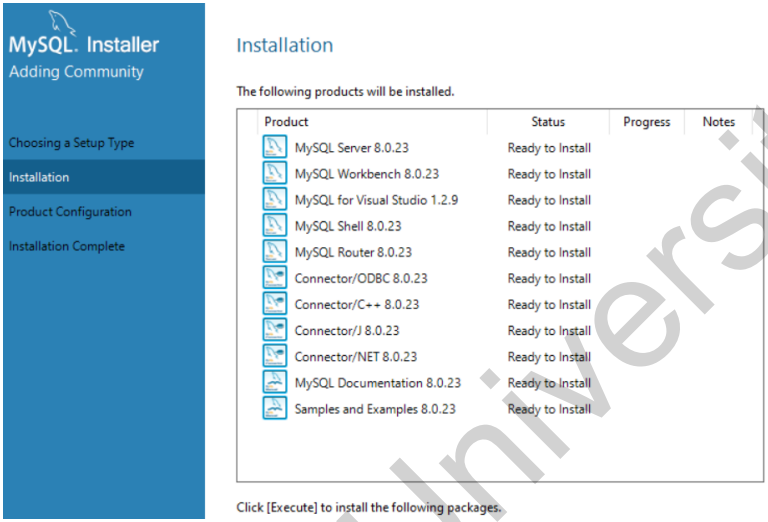
1 - сурет. MySQL орнату файлын жүктеу

Орнатушы қосымша MySQL-ді орнатудың бірнеше нұсқаларын ұсынады. MySQL-дің барлық құралдарын орнату үшін Full нұсқасын (2 сурет) таңдап, келесі (Next) батырмасын басыңыз. Full нұсқасында MySQL-дің барлық мүмкін деген қосымшалары бар: MySQL Server, MySQL Workbench, MySQL Shell, MySQL Router, MySQL Connectors, құжаттама, мысалдар және т.б.



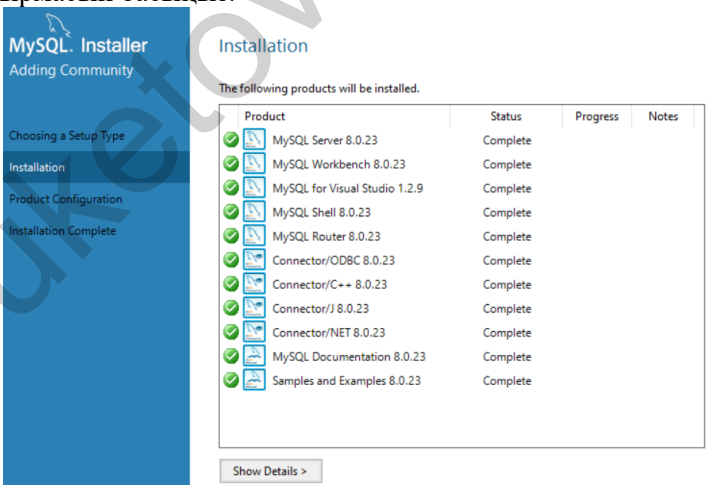
2 - сурет. Орнату нұсқасын таңдау

Келесі терезеде (3 сурет) орнатылатын құралдар құралдар тізімі беріледі. Осы құралдарды орнатамыз (Execute).



3 - сурет. MySQL құралдары

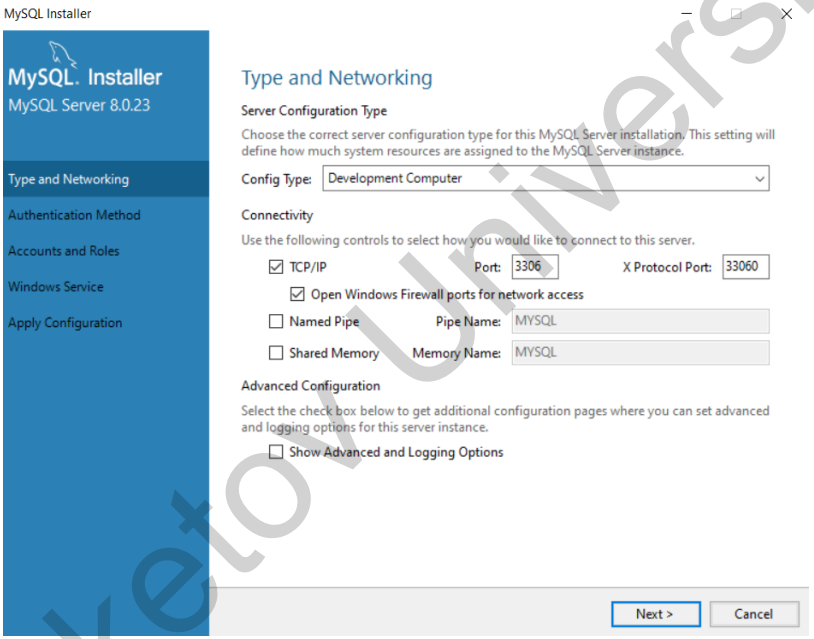
Құралдар орнатылып болғаннан кейін, Келесі (Next) батырмасын басыңыз.



4 - сурет. MySQL құралдары сәтті орнатылды

Type and Networking

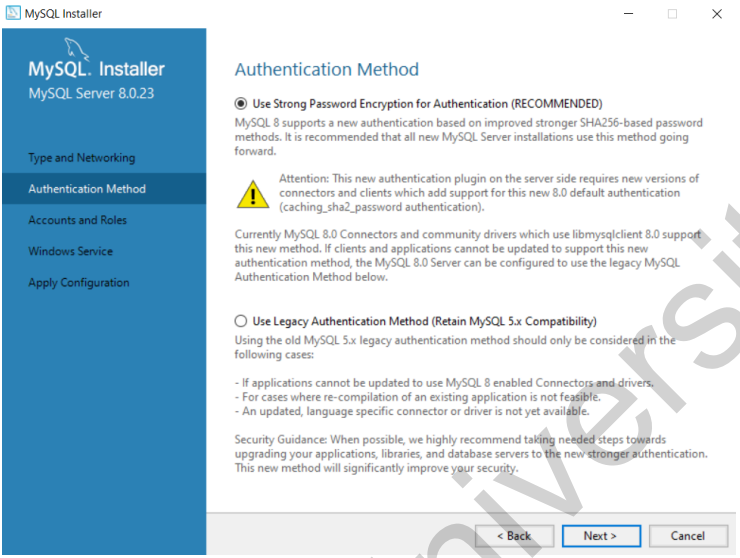
«Келесі» батырмасын басқаннан кейін сізден MySQL сервері үшін бірқатар конфигурация параметрлерін орнатуды сұрайды. Атап айтқанда, біз қосылымның TCP/IP протоколын және 3306 портын пайдаланатынын көруге болады. Осы үнсіздік бойынша қосылым мен порт параметрлерінің барлығын қалдырайық:



5 - сурет. Type and Networking

Authentication Method

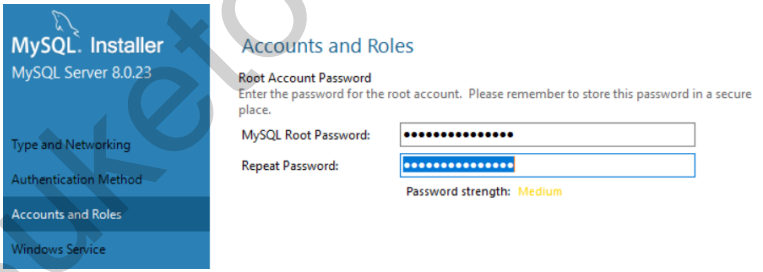
Келесі қадамда аутентификация әдісін орнатуды ұсынады. Үнсіздік келісім бойынша параметрлерді қалдырайық:



6 - сурет. Authentication Method

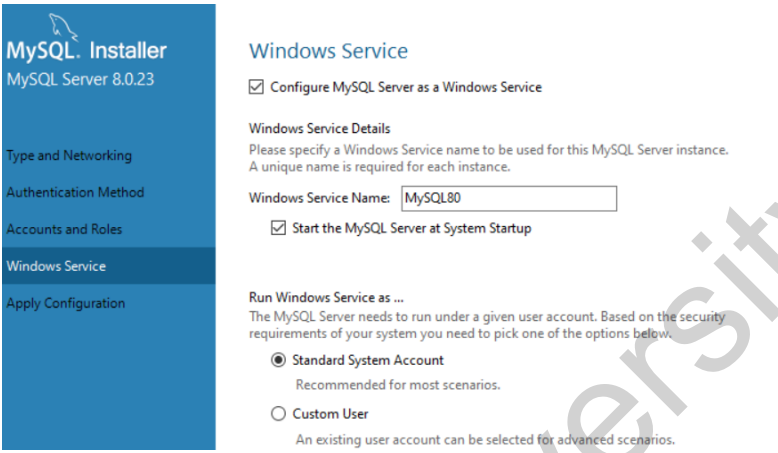
root қолданушы үшін құпия кілтті енгіземіз.

Содан кейін орнату бағдарламасының келесі терезесінде құпия сөзді енгіземіз және осы құпия сөз MySQL серверін іске қосу үшін қолданылады. Құпия сөзді ұмытып қалмаңыз.



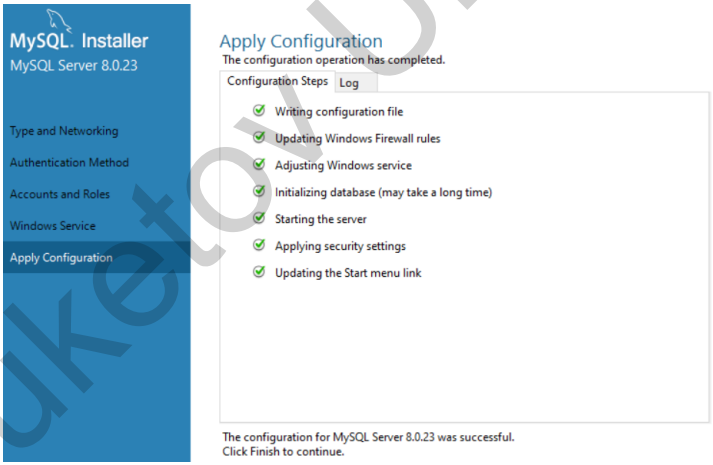
7 - сурет. *root* қолданушы үшін құпия кілтті енгізу

Windows Server терезесінде конфигурацияларды өзгертпей қалдырамыз. Бұл конфигурациялар MySQL серверінің операциялық жүйемен бірге іске қосылатынын көрсетеді.



8 - сурет. Windows қызметі

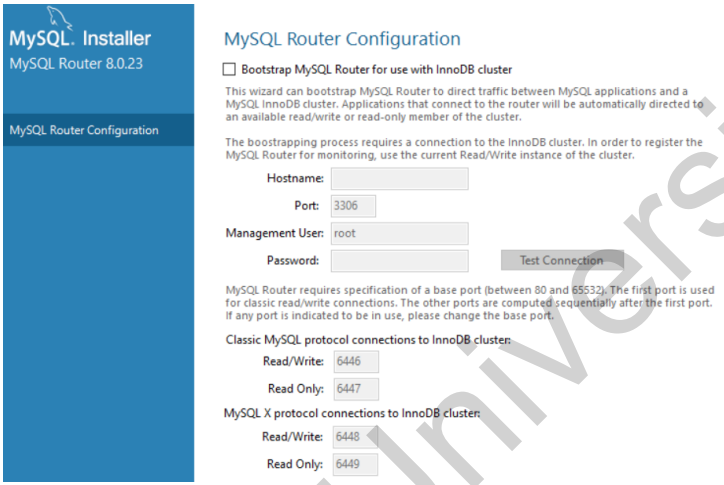
Келесі терезеде «Execute» батырмасын басу арқылы барлық көрсетілген конфигурация параметрлерін сақтаймыз:



9 - сурет. Конфигурация параметрлері

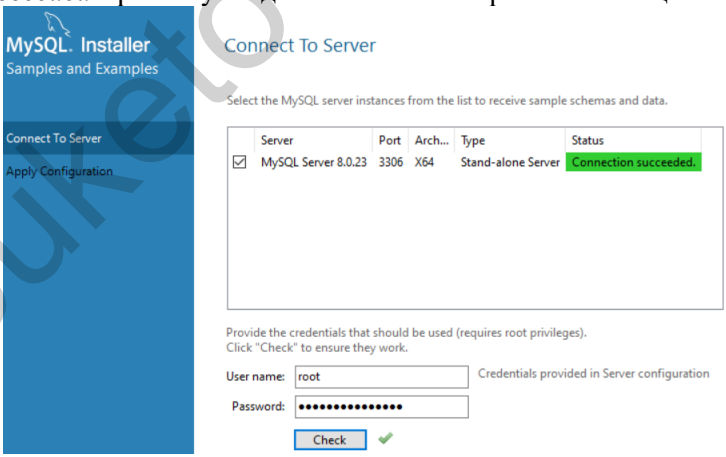
MySQL сервері толығымен орнатылады және конфигурацияланады. Орнатуды аяқтау үшін «Finish» батырмасын басамыз.

Содан кейін конфигурациялауға дайын өнімдер тізімі бар терезе қайтадан көрсетіледі. «Келесі» батырмасын басыңыз. MySQL Router үшін конфигурацияны орнату ұсынылады.



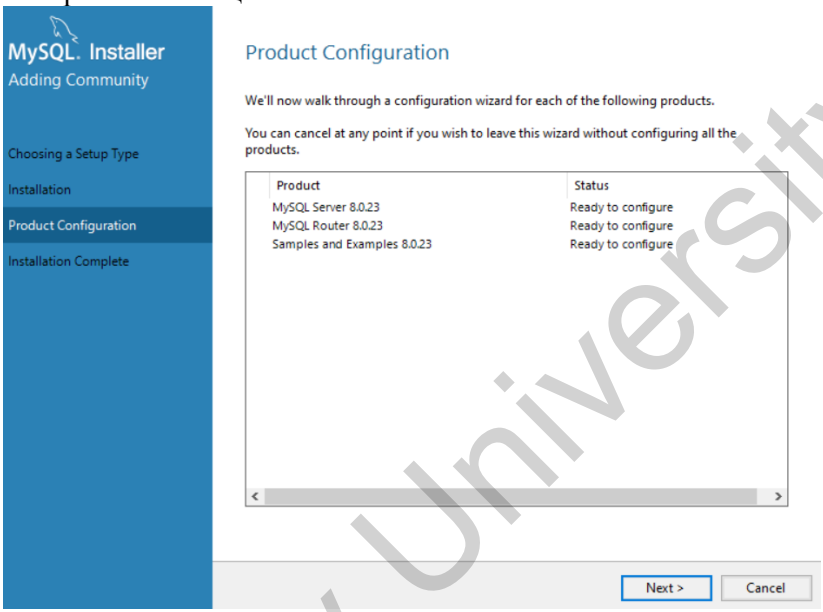
10 - сурет. MySQL Router

Құпия сөзді енгізіп, байланысты тексеріңіз. MySQL серверіне қосылу сәтті болса, сервердің күйі *Connection succeeded* күйіне ауысады. «Келесі» батырмасын басыңыз.



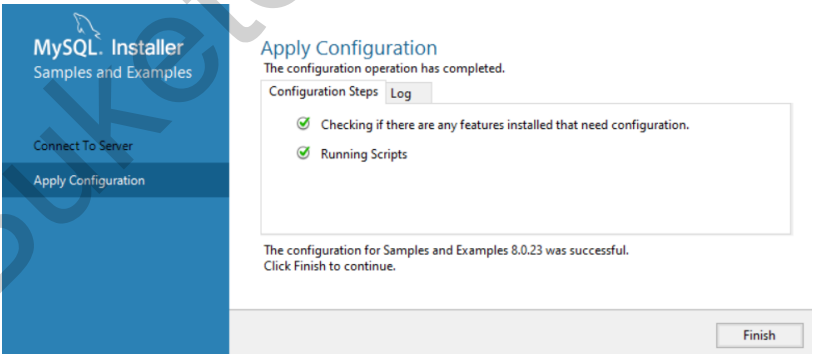
11 - сурет. Байланысты тексеру

Содан кейін орнатылған және конфигурацияланған өнімдердің тізімі бар терезені қайтадан көреміз және «Next» батырмасын басыңыз.



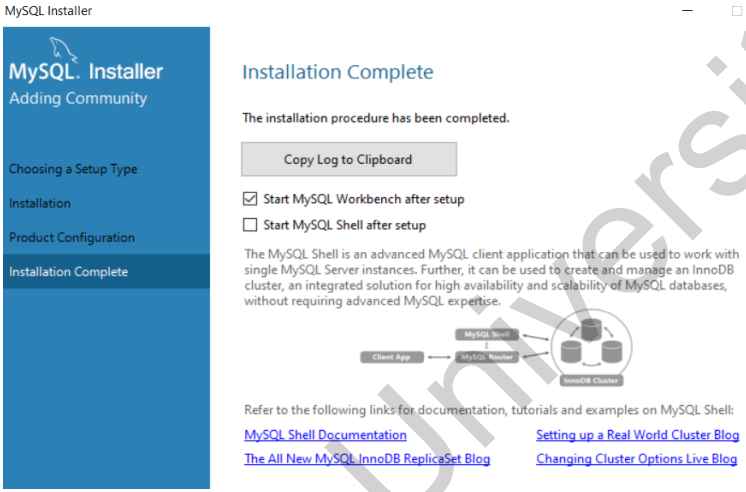
12 - сурет. Конфигурацияланған өнімдердің тізімі

Ал соңғы терезеде конфигурацияны қабылдау керек.



13 - сурет. Конфигурацияны аяқтау

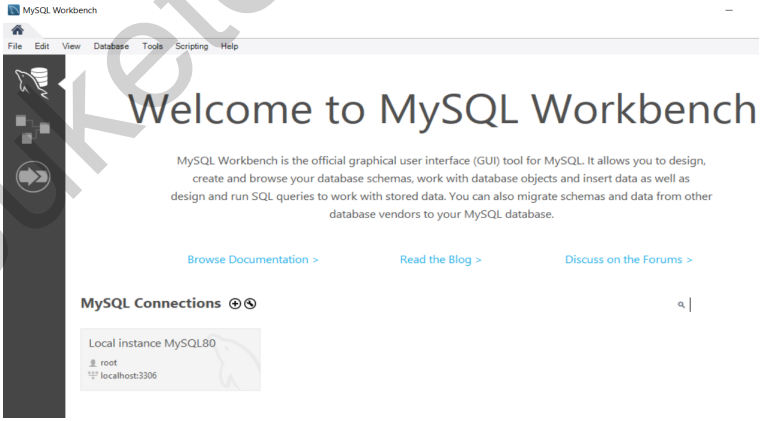
Соңғы терезеде екі белгіше бар: орнатудан кейін MySQL Workbench қосымшасын іске қосу және орнатудан кейін MySQL Shell іске қосу. Бұл қосымшалар MySQL серверін басқару үшін графикалық және консолдық клиенттерді іске қосуға мүмкіндік береді.



14 - сурет. Орнатуды аяқтау

MySQL бағдарламасы толық орнатылды.

Finish батырмасын басыңыз. MySQL толығымен орнатылды, конфигурацияланды және іске қосылды (15 сурет).

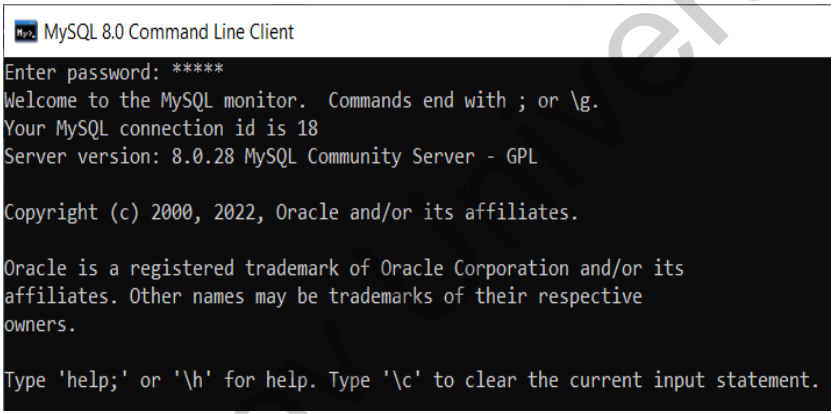


15 - сурет. MySQL Workbench ортасы

1.2 MySQL Command Line Client – консольді клиент

MySQL серверін орнату кезінде деректер қорымен жұмыс істеуге арналған консоль клиенті де орнатылады. Windows жүйесінде MySQL 8.0 Command Line Client бағдарламасын Бастау (Start) мәзірінен таба аласыз.

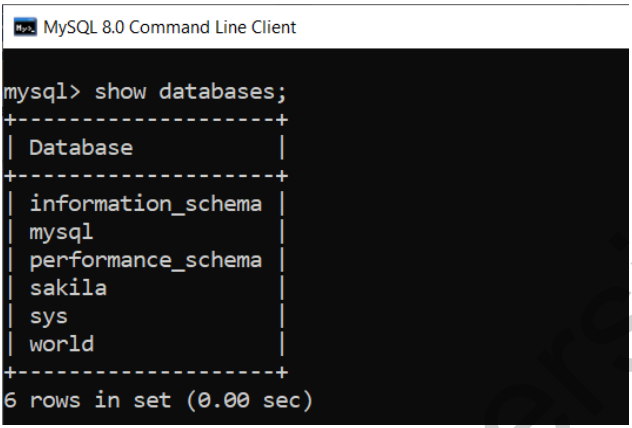
MySQL 8.0 Command Line Client іске қосамыз. Алдымен бізден құпия сөзді енгізу талап етіледі. Құпия сөздің әрбір символы жұлдызша белгісімен алмастырылады.



16 - сурет. MySQL Command Line Client

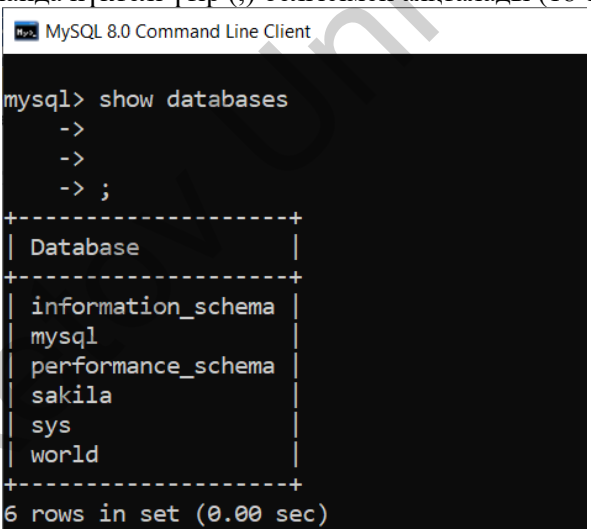
Мұнда *root* пайдаланушысы үшін MySQL орнату кезінде орнатылған құпия сөзді енгізу керек (7 сурет). Серверге сәтті қосылғаннан кейін осы консоль клиенті арқылы командаларды жіберуге болады.

Ең алдымен серверде қандай деректер қорлары бар екенін көрейік. Ол үшін SHOW DATABASES командасын енгіземіз.



17 – сурет. Локальді желідегі деректер қорларының тізімі

Әр команда нүктелі үтір (;) белгісімен аяқталады (18 сурет).



18 - сурет. Әр команда (;)-нүктелі үтір белгісімен аяқталады

Белгілі бір деректер қорын қолдану үшін алдымен қажетті деректер қорын ағымдық (current) ретінде орнату керек. Ол үшін USE командасын орындау керек, одан кейін деректер қорының атауы көрсетіледі. Мысалы, *world* деректер қорымен жұмыс істеу үшін USE *world* командаңы енгізіңіз (19 сурет).

19 - сурет. ДҚ қолдану

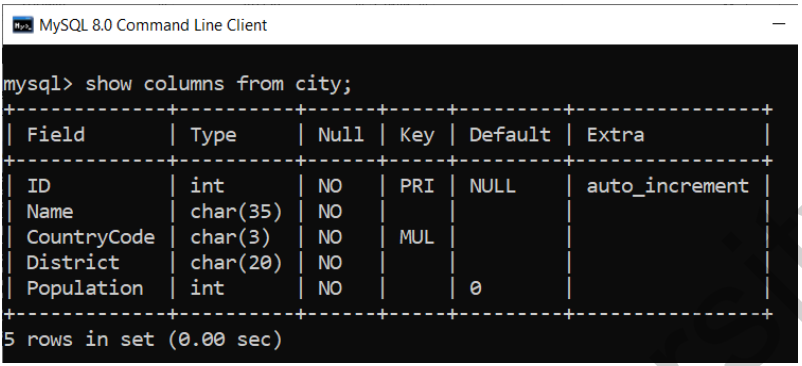
Деректер қорындағы кестелер тізімін көру үшін SHOW TABLES командасы қолданылады.

world деректер қорында 3 кесте бар: *city, country, countrylanguage* (20-сурет).

20 - сурет. ДҚ кестелер тізімі

Кесте құрылымын және өрістер тізімін көру үшін SHOW COLUMNS FROM *кесте_атауы*; командасы қолданылады. *кесте_атауы* орнына кестенің атауы жазылады.

city кестесіндегі өрістер тізімін алайық: SHOW COLUMNS FROM *city*;



21 - сурет. Кесте өрістерін алу

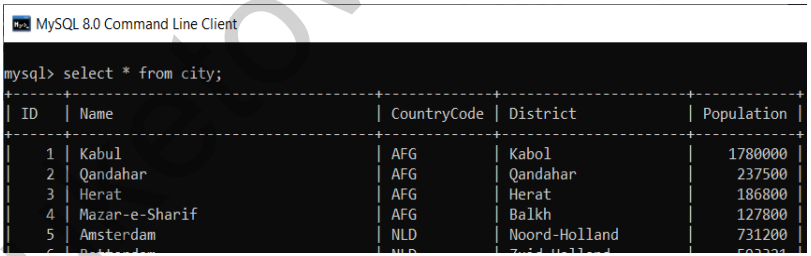
city кестесінде жазбаларды көру үшін SELECT командасы қолданылады. SELECT командасы арқылы сұраныстар құрылады. Сұраныстар туралы оқулықтың 3 және 4 тарауларында беріледі. Ең қарапайым сұранысты жазайық:

```
SELECT * FROM city;
```

Бұл сұраныс *city* кестесінен барлық (*) жазбаларды терезеге шығарады.

Жалпы сұраныс синтаксисі:

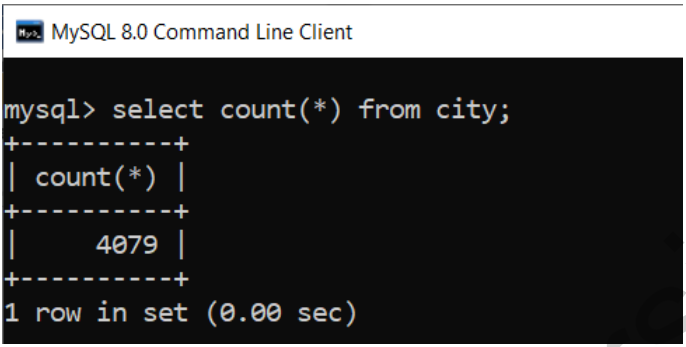
```
SELECT * FROM кесте_атауы;
```



22 - сурет. Кестедегі барлық жазбалар алу

Кестедегі жазбалар санын анықтау үшін COUNT() функциясы қолданылады. *city* кестесіндегі жазбалар санын анықтайық.

```
SELECT COUNT(*) FROM city;
```



23 - сурет. Кестедегі жазбалар саны

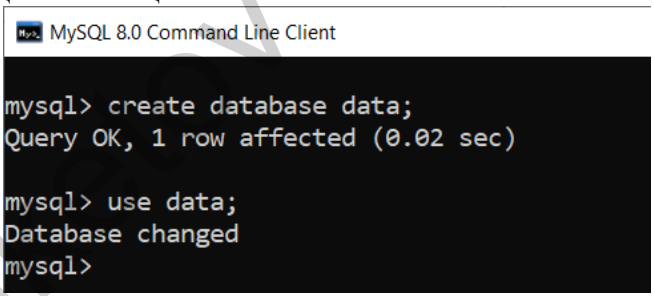
Жаңа деректер қорын құру.

Жаңа деректер қорын құру үшін CREATE DATABASE *дерекқор_атауы* командасы қолданылады. *дерекқор_атауы* – деректер қорының атауы.

Жаңа *data* деректер қорын құрайық:

```
CREATE DATABASE data;
```

Құрылған *data* деректер қорын пайдалану үшін USE *data* командасын жазыңыз.



24 - сурет. Жаңа деректер қорын құру

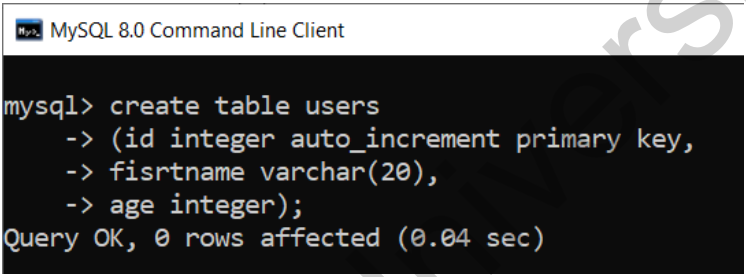
Жаңа кесте құрудың жалпы синтаксисі:

```
CREATE TABLE кесте_атауы
(өріс1_атауы      өріс1_типі      өріс1_атрибуты,
 өріс2_атауы      өріс1_типі      өріс1_атрибуты,
 ...
 өрісN_атауы      өрісN_типі      өрісN_атрибуты,
```

кесте_деңгейіндегі_атрибуттар)

Келесі құрылымдағы *users* кестесін құрайық:

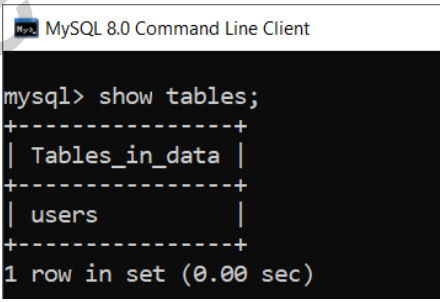
Өріс атауы	Өріс типі	Өріс атрибуты
id	integer	auto_increment, primary key
firstname	varchar	
age	integer	



25 - сурет. Жаңа кесте құру

```
CREATE TABLE users
(id INTEGER AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(20),
age INTEGER);
```

data деректер қорындағы кестелер тізімін көрейік.



26 - сурет. Деректер қорындағы кестелер тізімі

users кестесінің өрістерін көрейік:

```
mysql> show columns from users;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| id        | int         | NO   | PRI | NULL    | auto_increment |
| firstname | varchar(20) | YES  |     | NULL    |                |
| age       | int         | YES  |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

27 - сурет. Кесте өрістері

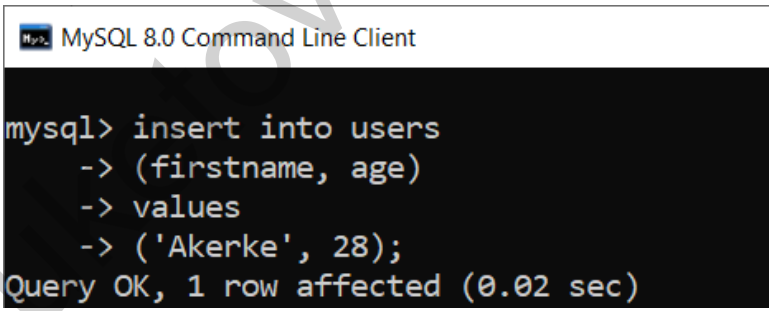
Жаңа жазба қосу

```
INSERT INTO кесте_атауы
([өріс1, өріс2, ... , өрісN)
VALUES ([мән1, мән2, …. мәнN]);
```

users кестесіне жаңа жазба қосайық:

```
INSERT INTO users (firstname, age) VALUES (‘Akerke’, 28);
```

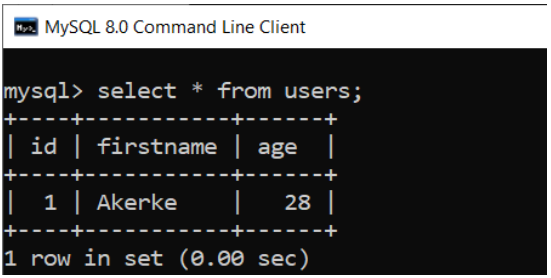
id өрісіне мән беру міндетті емес, себебі бұл өріс *auto_increment* атрибутына ие, автоматты түрде 1 санға артып, толтырылып отырады.



28 - сурет. Кестеге жаңа жазба қосу

users кестесіне енгізілген жазбаны алайық.

```
SELECT * FROM users;
```



29 - сурет. Сұраныс. Кестедегі барлық бағандарды алу

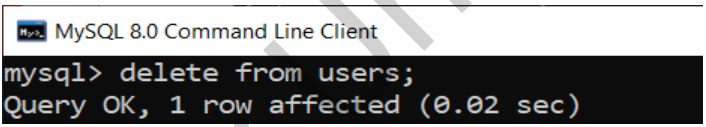
Кестеден барлық жазбаларды өшіру

Жалпы синтаксисі:

```
DELETE FROM кесте_атауы;
```

users кестесіндегі жазбаларды өшіріп тастайық:

```
DELETE FROM users;
```



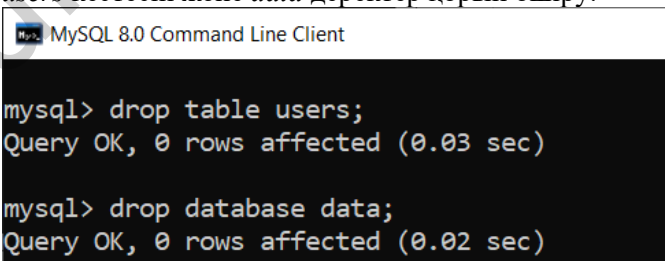
30 - сурет. Жазбаларды өшіру

Кестені немесе деректер қорын мүлдем өшіріп тастау үшін *drop* командасы қолданылады.

Жалпы синтаксисі:

```
DROP TABLE кесте_атауы;
DROP DATABASE дерекқор_атауы;
```

users кестесін және *data* деректер қорын өшіру.



31 - сурет. Кесте мен ДҚ өшіру

1.3 MySQL Workbench – графикалық клиент

MySQL серверімен жұмысты жеңілдету үшін MySQL Workbench графикалық клиенті қолданылады. Бұл құрал негізгі орнату бумасына кіреді. MySQL Workbench – деректер қорын визуалды жобалауға арналған қосымша, оның құрамында деректер қорын жобалау, модельдеу, құру және пайдалану құралдары бар.

MySQL Workbench мүмкіндіктері:

– деректер қорының моделін графикалық түрде ұсынуға мүмкіндік береді;

– кестелер арасындағы байланыстарды орнатудың көрнекі және функционалды механизмі;

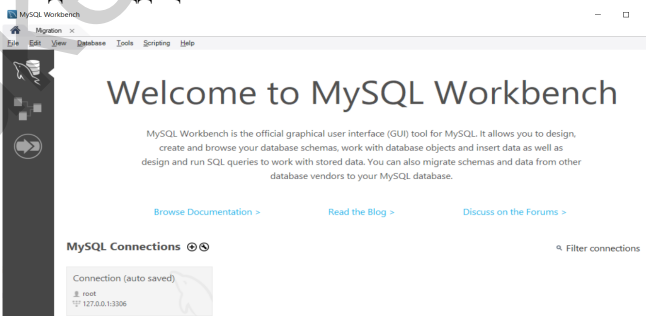
– Reverse Engineering – серверде бұрыннан бар деректер қорынан кесте құрылымын қалпына келтіру;

– серверге сұраныстарды бірден жіберуге және кесте түрінде жауап алуға мүмкіндік беретін ыңғайлы SQL сұраныстар редакторы;

– кестедегі деректерді визуалды режимде өңдеу мүмкіндігі.

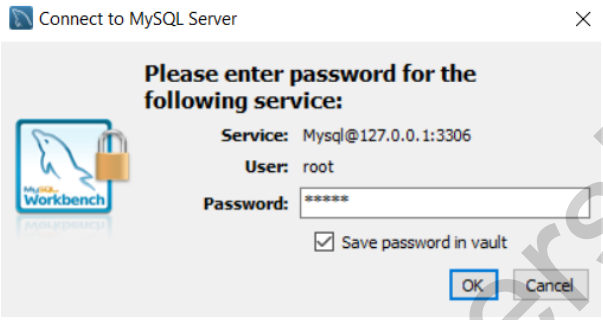
Windows жүйесіндегі Бастау мәзірінде бағдарламалар тізімінен MySQL Workbench  құралып таңдап, оны іске қосамыз.

MySQL Workbench бағдарламасын іске қосқаннан кейін серверге қосылу үшін сеансты таңдау керек. Біздің жағдайда бұл тек локальді - таңдаңыз.



32 - сурет. MySQL Workbench терезесі

root пайдаланушысы үшін құпия сөзді енгіземіз. Бұл құпия сөз MySQL орнату кезінде орнатылған (7 сурет).

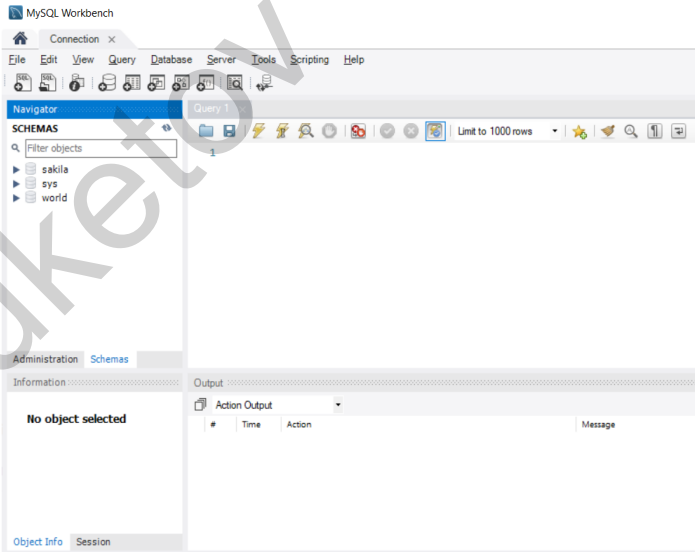


33 - сурет. Серверге кіру

MySQL Workbench жұмыс терезесі

MySQL Workbench терезесі келесі бөлімдерден тұрады:

1) SQL редакторы
2) анықтама тақтасы (Information)
3) навигация тақтасы (Navigation)
4) нәтижелер терезесі (Output)



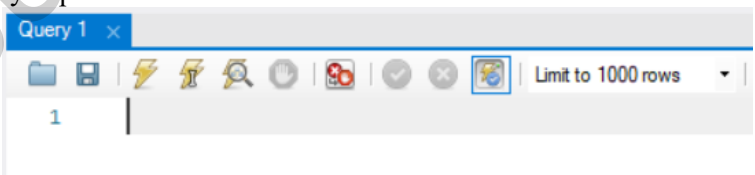
34 - сурет. MySQL Workbench жұмыс терезесі

Навигация тақтасы – белсенді MySQL серверінде деректер қорларының тізімі беріледі. Schemas қосылған *root* байланыстағы барлық деректер қорларының құрылымын (кестелер, ұсыныстар (view), сақталатын процедуралар мен функциялар, триггерлер, кілттер) көруге және басқаруға мүмкіндік береді. Деректер қорына өзгерістер енгізіп отырғаннан кейін «Жаңалау» батырмасы арқылы жаңартып отыру керек.

35 - сурет. Навигация тақтасы. Schemas бөлімі

SQL редакторы немесе скриптер жазу аймағы – сұраныстарды (скриптерді) жазуға және орындауға арналған. Жазылған скриптерді файлға сақтауға болады. SQL скрипт – мәтіндік файлда сақталған SQL операторлары.

Скрипті орындау үшін Execute (орындау) батырмасын басу керек.

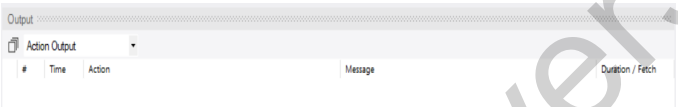


36 - сурет. Скрипт жазу терезесі

Нәтижелер терезесі (Output) - мұнда орындалған сұраныстардың қысқаша мазмұны беріледі:

– сұраныстың орындалған уақыты (Time);
– сұраныс (Action);
– сұраныстың орындалғаны туралы ақпарат немесе қателік (егер қателік анықталса);
– сұранысты орындауға кеткен уақыт (Duration).

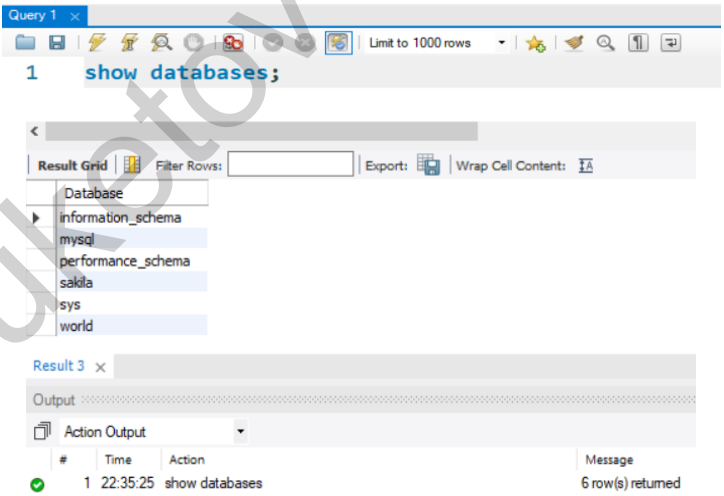
Скрипттің сәтті орындалғаны немесе қателіктерді осы терезеде көрсетілді.



37 - сурет. Output терезесі

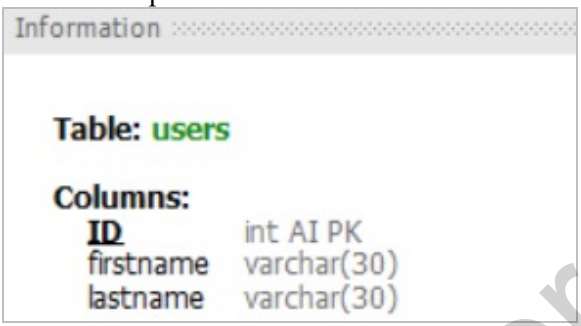
SHOW DATABASES; командасы жазылып, орындалды.

Нәтижесі кесте түрінде беріледі. Output терезесінде скриптің дұрыс орындалғаны, орындауға кеткен уақыты көрсетілген.



38 - сурет. Скрипті орындау және нәтижені алу

Анықтама тақтасынан белгілі бір кілттік сөздің синтаксисі мен сипаттамасын көре аласыз.



39 - сурет. Анықтама тақтасы

1.4 SQL сұраныстар тілі

SQL (Structured Query Language) – бұл көптеген жылдар бойы ақпараттық жүйелерді жасаушылар арасында танылған реляциялық деректер қорының тілі. SQL тілінің авторлары Дональд Чемберлин мен Рэймонд Бойс.

SQL – бұл барлық коммерциялық реляциялық ДҚБЖ өндірушілері кеңінен қабылдаған және қолдайтын бірінші және жалғыз деректер қорының тілі.

SQL пайдаланушының деректер қорымен әрекеттесуінің негізгі тәсілі олды және келесі операциялар жинағын орындауға мүмкіндік береді:

- мәліметтер қорында жаңа кесте құру;
- кестеге жаңа жазбаларды қосу;
- жазбаларды өзгерту;
- жазбаларды жою;
- бір немесе бірнеше кестелерден жазбаларды таңдау (көрсетілген шартқа сәйкес);
- кесте құрылымын өзгерту.

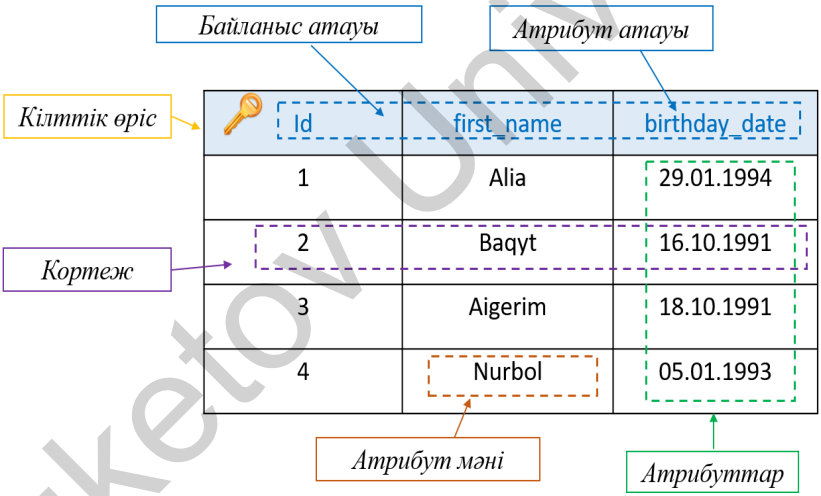
SQL тілінің ерекшеліктері:

– бұл бағдарламалау тілі емес, сұраныс тілі. Оған толыққанды веб-сайт немесе қосымша жазу мүмкін емес;

– анық және түсінікті құрылым. SQL тілі деректер қорымен жұмыс істеуді салыстырмалы түрде жеңілдетеді;

– әмбебап. Тіпті өте үлкен көлемдегі ақпаратты өңдеуге мүмкіндік беретін кез келген деректер қорлар мен браузерлер үшін сұраныстарды құрудың жалпы стандарттары бар;

– қол жетімділікті басқару. SQL көмегімен пайдаланушылардың әртүрлі топтары үшін деректерге қол жетімділікті қамтамасыз етуге, жетімділікті алып тастауға немесе шектеуге, сондай-ақ оларға белгілі бір мүмкіндіктер жиынтығын беруге болады: оқу, өзгерту, жасау, жою, көшіру. Бұл деректер қорларын бұзудан немесе келісілмеген өзгерістерден қорғайды.



1 - сызба. SQL дерекқорының құрылымы

2 - кесте

Реляциялық деректер қорында қолданылатын негізгі терминдер

Реляциялық термин	*Анықтамасы*
Деректер қоры	Кестелер жиынтығы
Деректер қорының схемасы	Кесте тақырыптарының жиынтығы
Байланыстар	Кестелер
Байланыс атауы	Кесте атауы
Байланыс денесі	Кесте
Байланыс атрибуты	Кесте өрістер (бағандар) атауы
Байланыс кортежі	Кесте жолы (жазба)
Байланыс дәрежесі	Кестедегі өрістер (бағандар) саны
Байланыс қуаты	Кестедегі жолдар (жазба) саны
Домен және деректер түрлері	Кесте ұяшығындағы деректер типтері
Бастапқы кілт	Деректер кестелерін байланыстыратын өріс

SQL деректер қорының құрылымы алты элементтен тұрады:

– Кілт – деректер кестелерін байланыстыратын ең маңызды өріс (баған).

– Байланыс – жолдар мен бағандардан тұратын кестеде берілген деректер.

– Атрибут – атау, тип, бағасы немесе басқа параметрлерді сақтайтын баған

– Домен – берілген атрибут үшін рұқсат етілген мәндер.

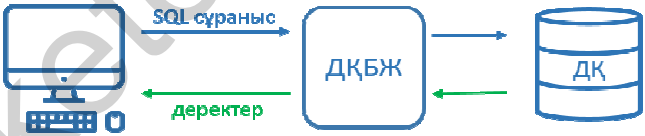
– Кортеж – қандай да бір объект туралы барлық деректерсақталған нөмірленген жол.

– Мәні – кестедегі ұяшық мазмұны.

SQL көмегімен деректер қорымен жұмыс істеу үшін пайдаланушы мен кестелер сақталған сервер арасындағы байланыс орнататын деректер қорын басқару жүйесін (ДҚБЖ) қолданылады. ДҚБЖ тегін және коммерциялық болып табылады, мысалы MySQL, Microsoft SQL Server, SQLite, Oracle, Ingres және т.б.

SQL операторлары бірнеше топқа бөлінеді:

- Деректерді анықтау мәлімдемелері (Data Definition Language, DDL):
  - CREATE деректер қорын, кесте құру,
  - ALTER кестені өзгерту,
  - DROP объектілерді өшіру;
- Деректерді басқару операторлары (Data Manipulation Language, DML):
  - SELECT белгілі шартқа байланысты деректерді таңдау,
  - INSERT жаңа дерек қосу,
  - UPDATE деректерді өзгерту,
  - DELETE деректерді өшіру;
- деректерге қол жеткізу анықтамалары (Data Control Language, DCL):
  - GRANT пайдаланушыға (топқа) объектімен белгілі бір операцияларға рұқсаттар береді
  - REVOKE бұрын берілген рұқсаттардың күшін жояды,
  - DENY рұқсаттан басым болатын тыйымды белгілейді;
- транзакцияны бақылау мәлімдемелері (Transaction Control Language, TCL):
  - COMMIT транзакцияны орындайды,
  - ROLLBACK ағымдағы транзакция жасалған барлық өзгерістерді кері қайтарады
  - SAVEPOINT транзакцияны кішірек бөлімдерге бөледі.



40 - сурет. Деректер қорын басқару жүйесімен жұмыс істеу

Деректер қорының негізгі үш элементі

1. Клиент – сұраныс жасау үшін қолданылатын интерфейс. Мысалы, MySQLWorkbench.
2. ДҚБЖ – пайдаланушы сұраныстарын деректер қорына жібереді және жауапты түсінікті түрде қайтарады.
3. Деректер қоры – бұл құрылымдалған деректер сақталған және бір-бірімен байланысқан кестелер жиынтығы.

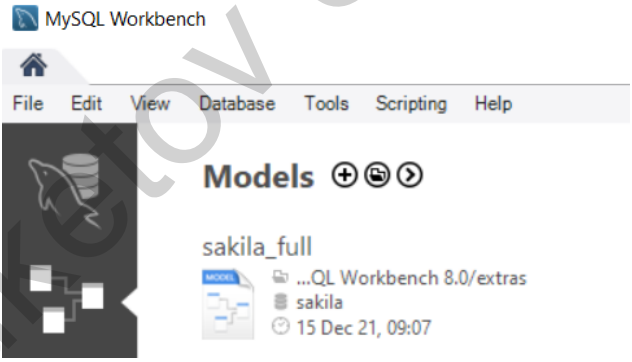
1.5 MySQL Workbench ортасында ER диаграммаларын құру

Жаңа деректер қорын іске асыруды бастағанда міндетті түрде ең алдымен жобалау керек. Деректер қорын жобалау процесінде үш негізгі кезеңді ажыратуға болады.

1) Талаптарды талдау. Алдымен деректер қорында, онда сақталған деректерге және деректер элементтерін бір-бірімен байланыстыруға қойылатын барлық талаптарды анықтау қажет. Іс жүзінде бұл кезең қосымшаның талаптарын егжей-тегжейлі анықтаудан, сондай-ақ деректер қоры және қосымшамен өзара әрекеттесетін әртүрлі адамдармен қарым-қатынастан тұрады.

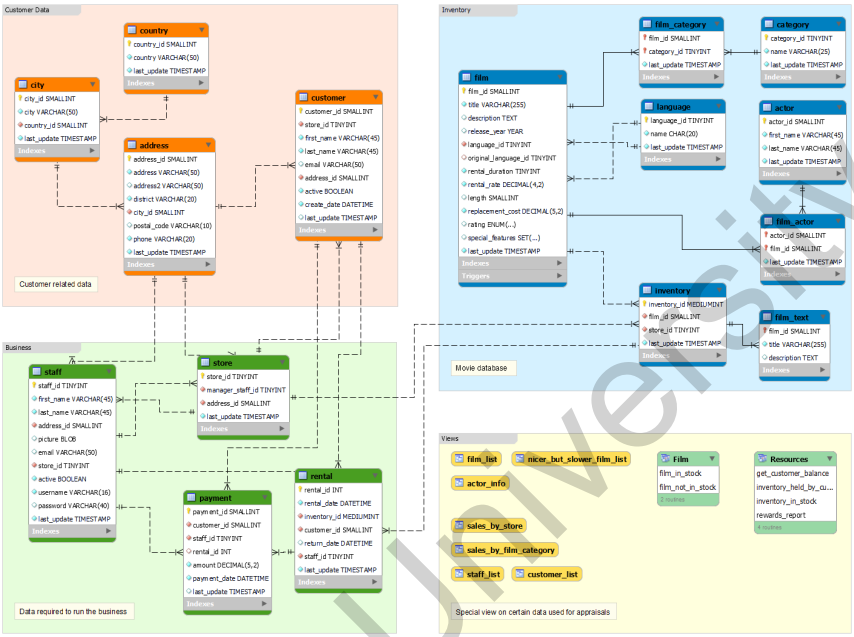
2) Тұжырымдамалық жобалау. Деректер қорына қойылатын талаптарды анықтағаннан кейін оларды ДҚ жобасының ресми сипаттамасы ретінде көрсету керек. Дәстүрлі тұжырымдамалық жобада Entity Relationship (ER) моделі қолданылады. Бұл модельді 1976 жылы Питер Чен ұсынған.

3) Логикалық жобалау –деректер қорының жобасын нақты басқару жүйесіне және деректер қорына түрлендіру.



41 - сурет. MySQL Workbench ортасында ДҚ моделін құру

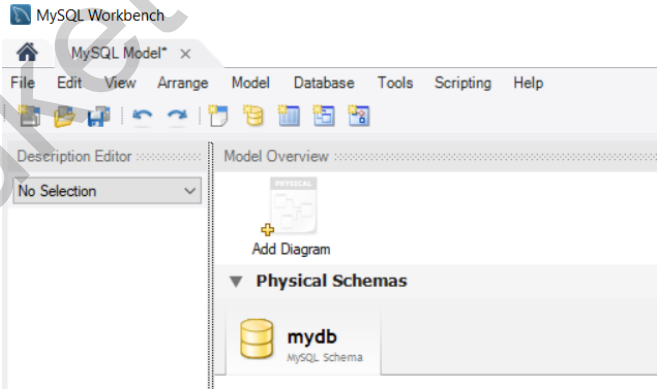
Үнсіздік күй бойынша MySQL Workbench ортасында *sakila_full* моделі бар. 42 суретте осы моделдің ER диаграммасы көрсетілген.



42 - сурет. *sakila_full* деректер қорынының ER диаграммасы

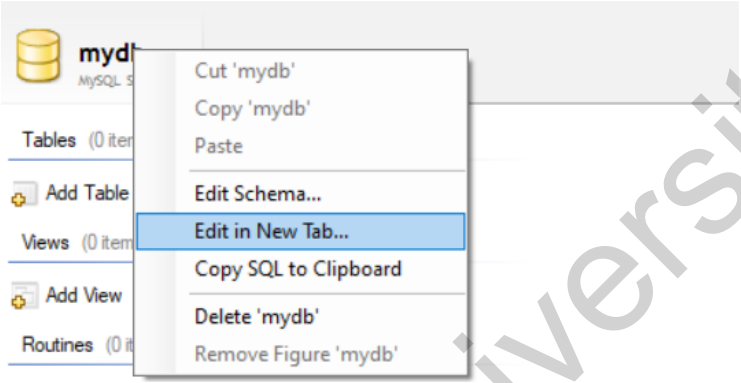
Жаңа деректер қорының моделін құру.

⊕ батырмасын басып жаңа модель құрамыз.



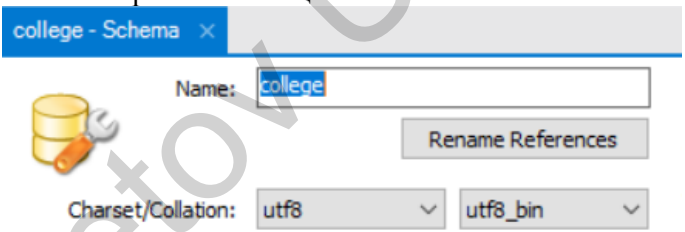
43 - сурет. Жаңа құрылған модель

Деректер қорының атауы – mydb (43 сурет). Деректер қорының атауын өзгертейік. Деректер қорын белгілеп, (тінтуірдің оң жақ батырмасы) контекстік мәзірдегі Edit in New Tab командасын таңдаңыз.



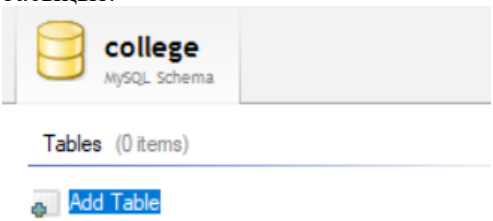
44 - сурет. Деректер қорының атауын өзгерту

Деректер қорының жаңа атауын енгізіп, Rename References батырмасын басыңыз.



45 - сурет. Деректер қорының атауын өзгерту

Деректер қорына кестелер қосайық. Add Table батырмасын басыңыз.

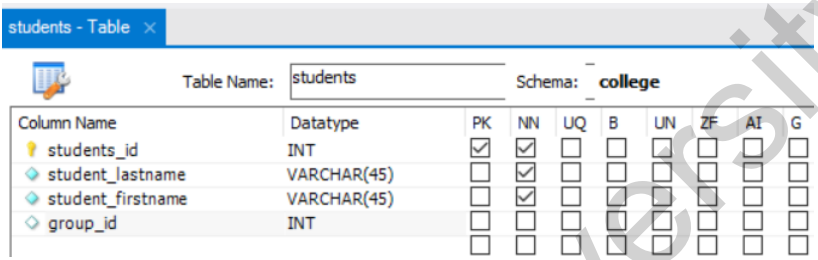


46 - сурет. Деректер қорының атауын өзгерту

Кесте атауын және өрістерді енгіземіз. Өріс типін және атрибуттарын таңдаймыз.

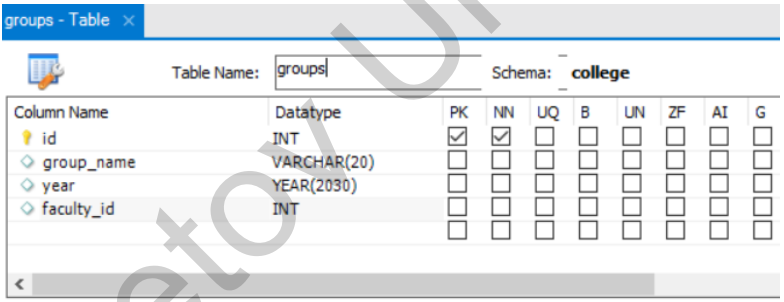
Кесте атауы – *students*

Кесте өрістері – *students_id, student_lastname, students_firstname, group_id*

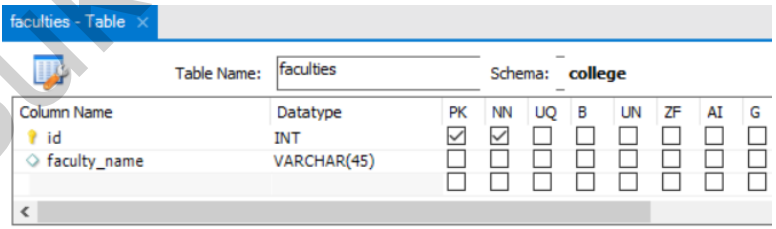


47 - сурет. *students* кестесін құру

group пен *faculites* кестелерін құрамыз.



48 - сурет. *group* кестесін құру



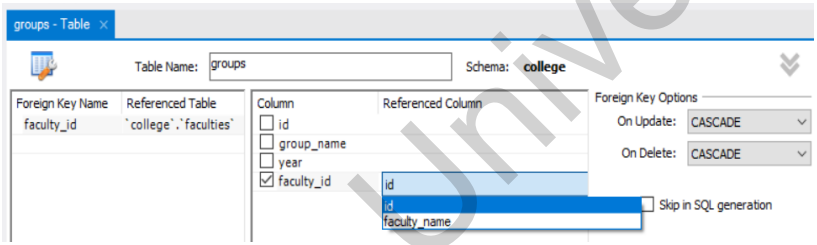
49 - сурет. *faculites* кестесін құру

groups кестесіне *faculty_id* сыртқы кілтін қосамыз. Бұл сыртқы кілт *groups* кестесін *faculites* кестесімен байланыстырамыз.

Бұл кестелердегі бірдей өріс – *groups.faculty_id* = *faculties.id*.

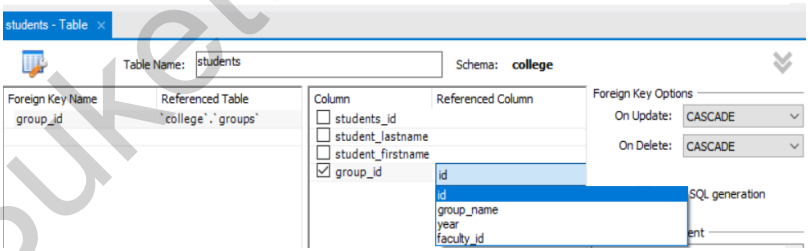
groups кестесін белгілеп, Foreign Keys тақтасына өтіңіз. Foreign Keys Name – сыртқы кілтке атау береміз және Referenced table – негізгі кестені (*faculites*) көрсетеміз. Екі кестеден байланысқан өрістерді (column және referenced column) таңдаймыз. On Update және On Delete опцияларын таңдаңыз.

Сыртқы кілттер және On Update және On Delete опциялары туралы оқулықтың 2.5 бөлімінде жазылған.



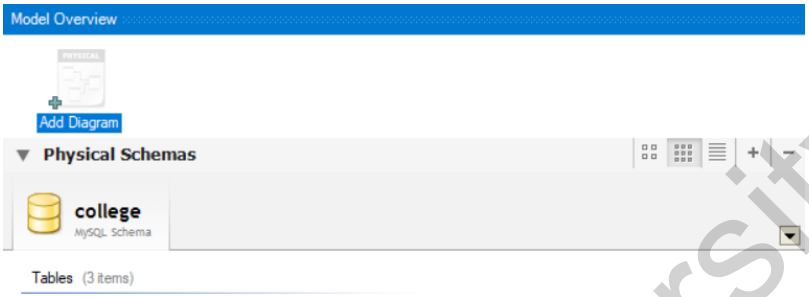
50 - сурет. Сыртқы кілт қосу

Дәл осылай *students* кестесіне сыртқы кілт қосамыз. Бұл сыртқы кілт *students* пен *groups* кестелерін байланыстырады.



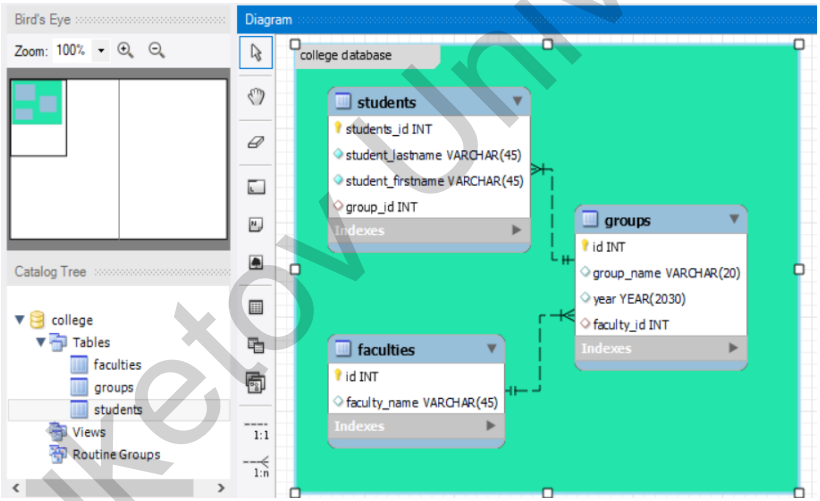
51 - сурет. Сыртқы кілт қосу

ER диаграммасын құру үшін Add Diagram таңдаймыз.



52 - сурет. Диаграмма құру

Диаграммаға кестелерді орналастырамыз.



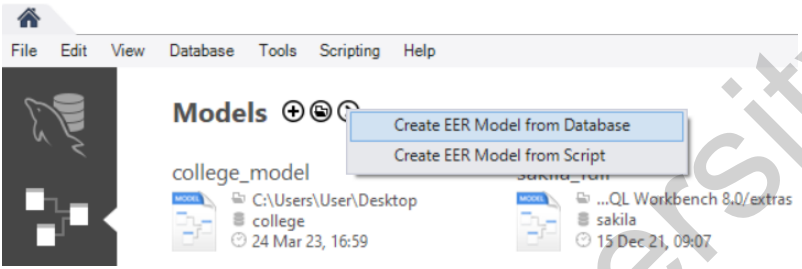
53 - сурет. Құрылған ER диаграмма

Құрылған ER диаграмманы графикалық (.PNG, .SVG) және .PDF форматта экспорттауға болады. Ол үшін File мәзіріндегі Export командасын таңдаймыз.

Құрылған модельді MySQL Workbench File (.mwb) типте сақталады.

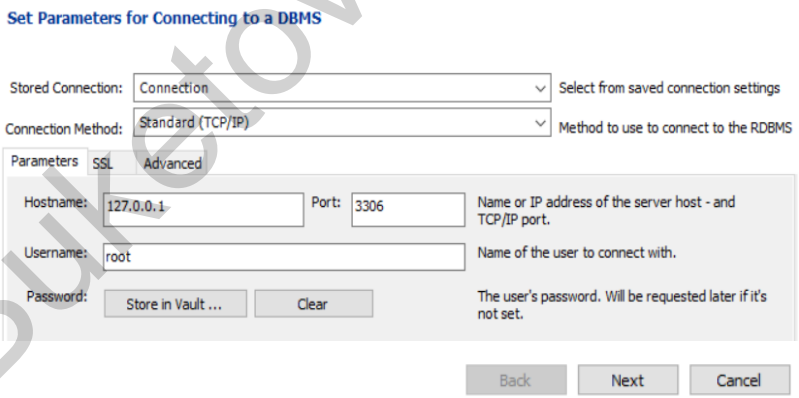
Серверде бар деректер қорының моделін құру

Серверде бар деректер қорының моделін құрайық. Create EER Model from Database командасын таңдаймыз.

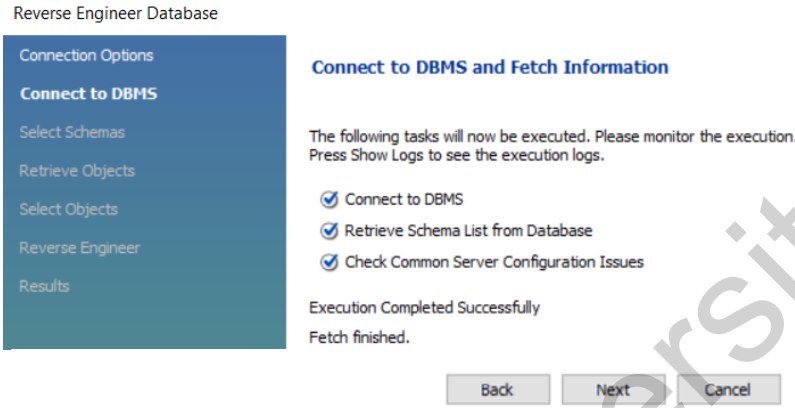


54- сурет. ДҚ үшін диаграмма құру

Reverse Engineer Database серверде бар деректер қоры үшін арқылы модель құра аламыз. Байланыс түрін таңдаймыз. Next батырмасын басамыз.
Stored Connection: *Connection*
Username: *root*

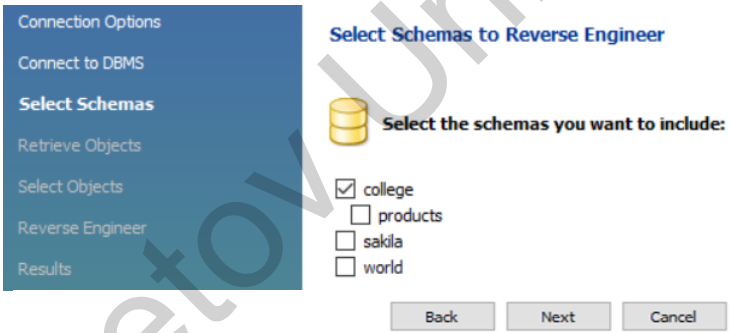


55 - сурет. Байланыс пен пайдаланушы атауын енгізу

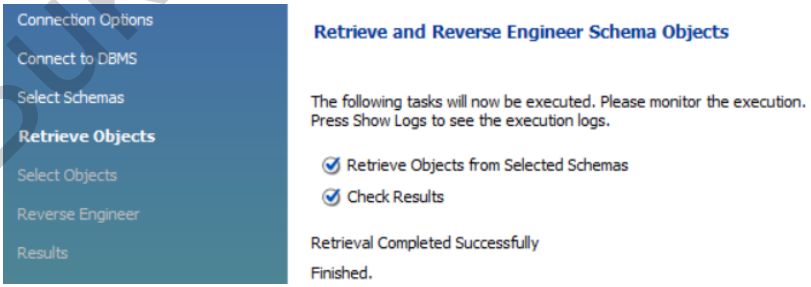


56 - сурет. Деректер қорына қосылу

Сервердегі деректер қорын таңдаймыз.

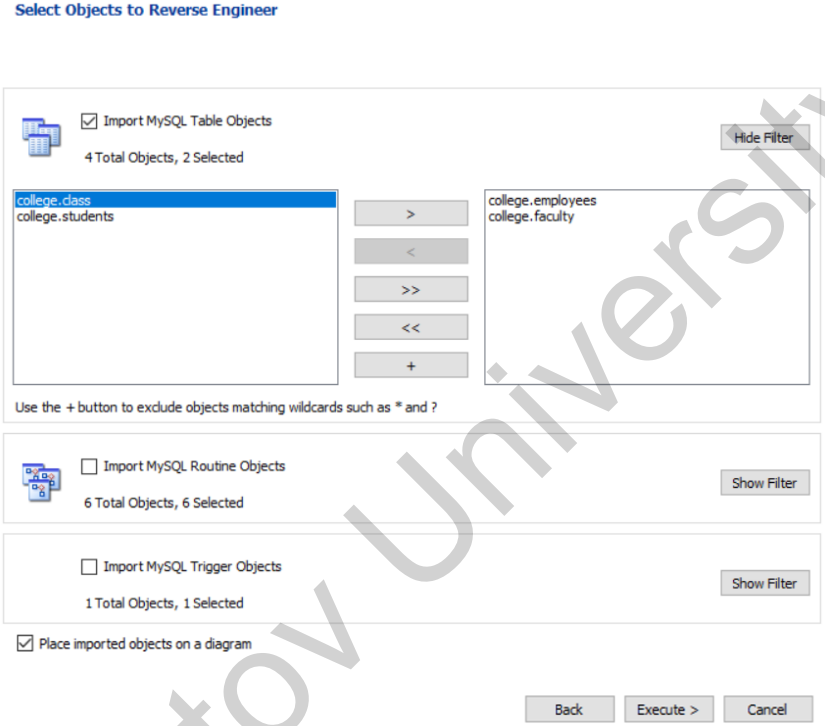


57 - сурет. Деректер қорын таңдау

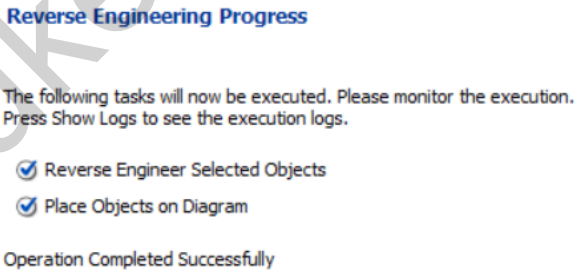


58 - сурет. Деректер қорынан объектілерді алу

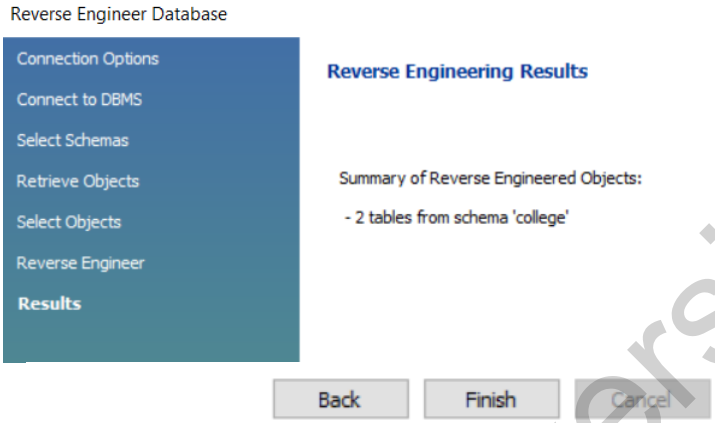
Қажетті деген кестелерді және басқа объектілерді таңдап аламыз (59 сурет). Сол жақ тақтада таңдалынған кестелер тізімі беріледі (*college.class, college.class*).



59 - сурет. Деректер қорындағы объектілерді таңдау

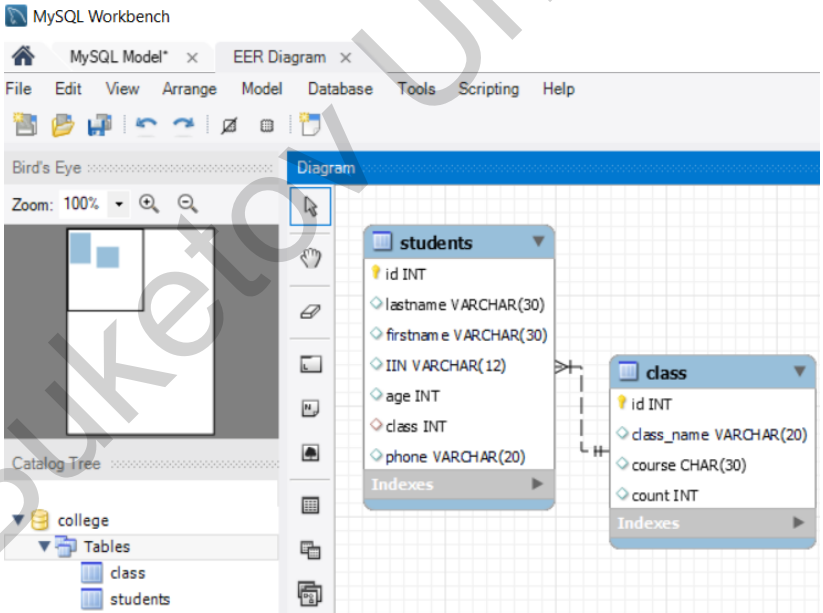


60 - сурет. Таңдалынған объектілерді алу және диаграммаға орналастыру



61 - сурет. *college* деректер қорынан 2 кесте жүктейді

Нәтижесінде келесідей диаграмманы аламыз.



62 - сурет. Құрылған диаграмма

1.6 Пайдаланушылар

MySQL серверін алғашқы рет орнатқаннан кейін, оның деректер қорларына қол жеткізу және басқару *root* пайдаланушыға ғана рұқсат етілген. *root* – барлық құқықтарға ие супер пайдаланушы болып табылады. *root*-пайдаланушы басқа пайдаланушыларды қосады, жояды және олардың құқықтарын шектей алады. Қолданушылары көп жүйелерде құқықтары шектелген пайдаланушыларды құрған тиімді.

Пайдаланушы құру

```
CREATE USER [IF NOT EXISTS] пайдаланушы_аты IDENTIFIED BY 'құпиясөз';
```

пайдаланушы_аты – пайдаланушы аты екі бөліктен тұрады пайдаланушы_аты@хост_атауы.

Мысалы:

```
CREATE USER admin@localhost IDENTIFIED BY 'password123';
```

Пайдаланушы аты: admin@localhost

Пайдаланушы құпиясөз: password123

Құпиясөзді ауыстыру

Пайдаланушы құпиясөзді ауыстырудың екі әдісі бар.

```
ALTER USER 'пайдаланушы_аты'@'хост_атауы' IDENTIFIED BY 'жаңа_құпия_сөз'
```

және

```
SET PASSWORD FOR 'пайдаланушы_аты'@'хост_атауы'= PASSWORD('жаңа_құпия_сөз');
```

Мысалы

```
ALTER USER 'admin@'localhost' IDENTIFIED BY 'new_password'
```

немесе

```
SET PASSWORD FOR 'admin'@'localhost' = PASSWORD('new_password');
```

Пайдаланушы өшірудің жалпы синтаксисі:

```
DROP USER 'пайдаланушы_аты'@'хост_атауы';
```

Мысалы:

```
DROP USER 'admin'@'localhost';
```

Пайдаланушыға құқықтар (рұқсат, privileges) беру

GRANT операторы арқылы пайдаланушыларға құқықтар беріледі.

Жалпы синтаксисі:

```
GRANT құқықтар_түрі
ON объектілер
TO пайдаланушы_аты;
```

Пайдаланушыларға ең жиі берілетін құқықтар:

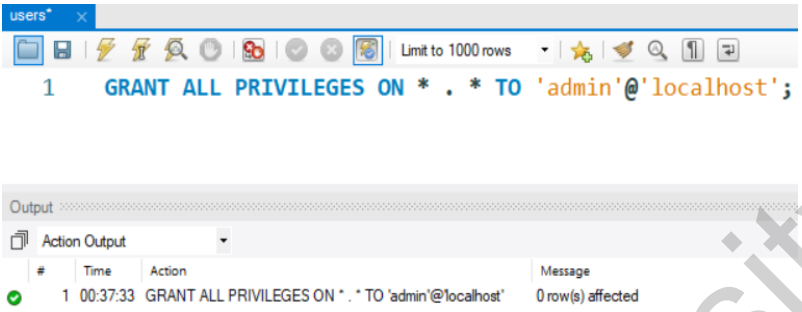
1) ALL PRIVILEGES – барлық құқықтарды беру
2) CREATE – деректер қоры мен кестелер құру
3) DROP – деректер қоры мен кестелерді жою
4) DELETE – кестелерден жазбаларды өшіру
5) INSERT – кестелерге жазбалар қосу
6) SELECT – деректерді оқу
7) UPDATE – кестедегі жазбаларды өзгерту

Пайдаланушы құқықтарды берудің 3 деңгейі бар:

1) Глобальді деңгей – сервердегі барлық объектілерге қол жеткізе алады;
2) Деректер қоры деңгейі – тек белгілі бір деректер қорын басқаруға құқығы бар;
3) Кесте деңгейі – тек белгілі бір кестелерді ғана басқара алады;
4) Баған деңгейі – тек белігілі бір өрістерді ғана басқаруға құқығы бар.

Глобальді деңгейде құқықтар беру

Құрылған пайдаланушыға барлық құқықтарды беру. Құқық беру үшін ең алдымен пайдаланушыны CREATE операторы арқылы құрып алу керек

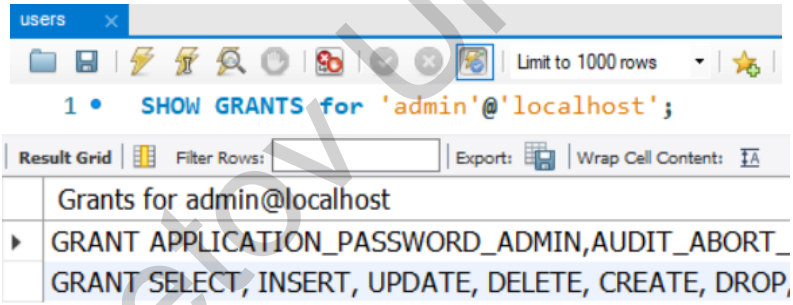


63 - сурет. *admin* пайдаланушыға барлық құқықтар беру

```
GRANT ALL PRIVILEGES ON * . * TO 'admin'@'localhost';
```

Пайдаланушының құқықтарды көру үшін

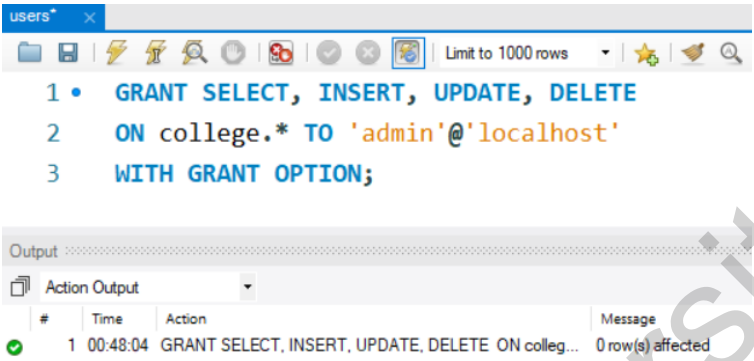
```
SHOW GRANTS for 'admin'@'localhost';
```



64 - сурет. Пайдаланушының құқықтарын көру

Деректер қоры деңгейінде құқықтар беру

admin пайдаланушыға *college* деректер қорына ғана рұқсат берейік.

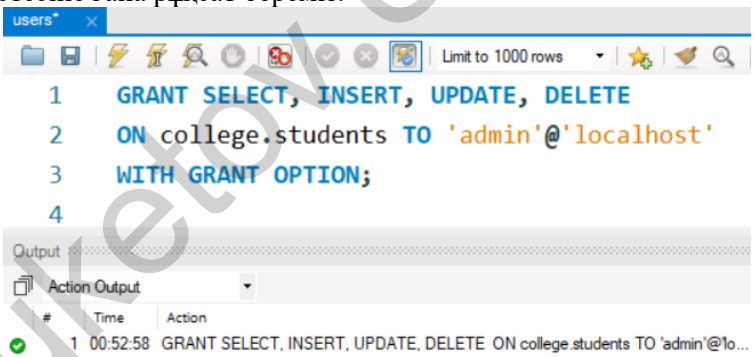


65 - сурет. Деректер қоры деңгейінде құқықтар беру

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON college.* TO 'admin'@'localhost'
WITH GRANT OPTION;
```

Кесте деңгейінде ғана рұқсат беру

admin пайдаланушыға *college* деректер қорындағы students кестесіне ғана рұқсат береміз.

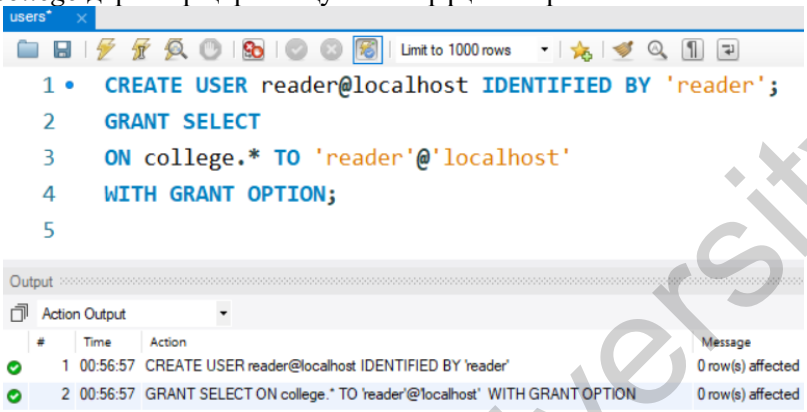


66 - сурет. Кесте деңгейінде құқықтар беру

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON college.students TO 'admin'@'localhost'
WITH GRANT OPTION;
```

Деректер қорын оқуға ғана рұқсат беру

reader пайдаланушысын құрамыз. Бұл пайдаланушыға *college* деректер қорын оқуға ғана рұқсат береміз.

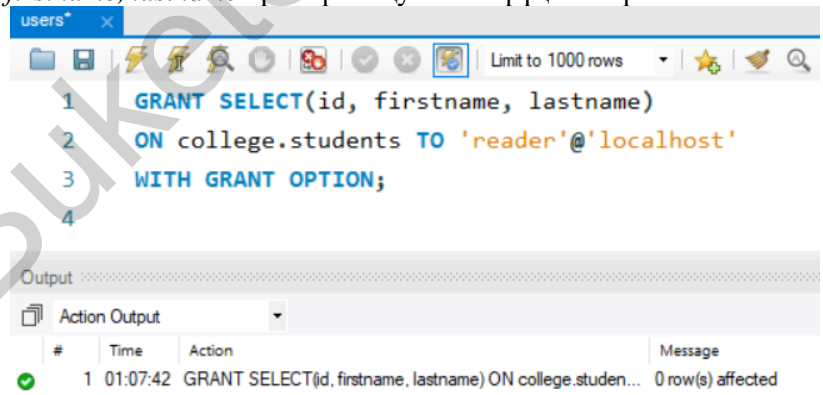


67 - сурет. Деректер қорын оқуға рұқсат беру

```
CREATE USER reader@localhost IDENTIFIED BY 'reader';
GRANT SELECT
ON college.* TO 'reader'@'localhost'
WITH GRANT OPTION;
```

Баған деңгейінде рұқсат беру

reader пайдаланушыға *college.students* кестесіндегі *id, firstname, lastname* өрістерін оқуға ғана рұқсат береміз.



68 - сурет. Баған деңгейінде рұқсат беру

```
GRANT SELECT(id, fisrtname, lastname)
ON college.students TO 'reader'@'localhost'
WITH GRANT OPTION;
```

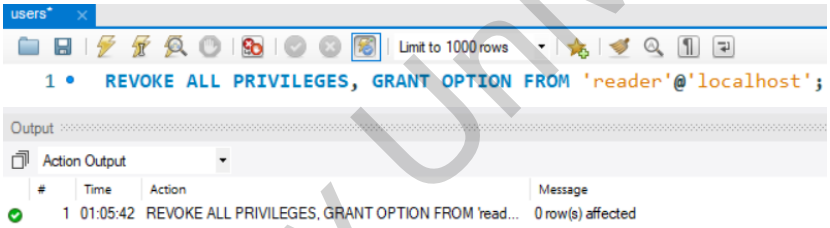
Құқықтарды алып тастау

Берілген құқықтарды алып тастау үшін REVOKE операторы қолданылады.

Жалпы синтаксисі:

```
REVOKE құқықтар_тізімі
ON объект
FROM пайдаланушы_аты;
```

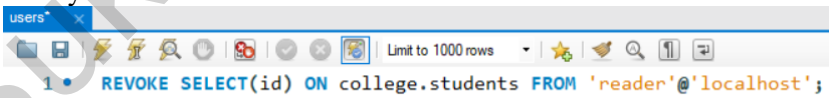
Барлық құқықтарды алып тастау



69 - сурет. Барлық құқықтарды алып тастау

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM
'reader'@'localhost';
```

college.students кестесіндегі *id* өрісін оқу құқығын алып тастау



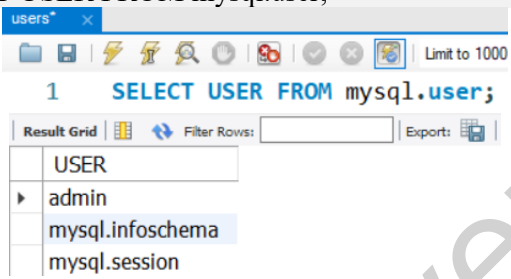
70 - сурет. Өрісті оқу құқығын алып тастау

```
REVOKE SELECT(id) ON college.students FROM
'reader'@'localhost';
```

```
REVOKE SELECT, INSERT, UPDATE, DELETE
ON college.students FROM 'admin'@'localhost'
```

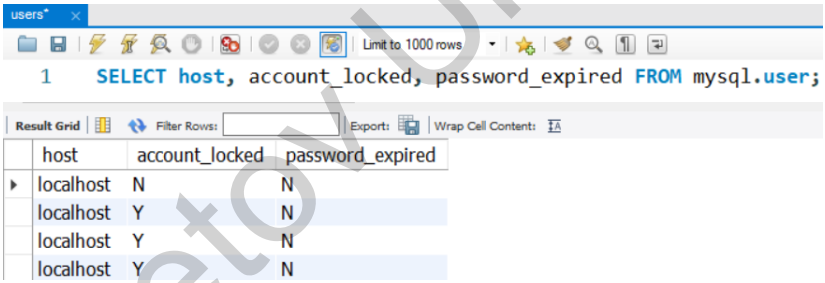
Пайдаланушылар тізімі

```
SELECT USER FROM mysql.user;
```



71 - сурет. Пайдаланушылар тізімі

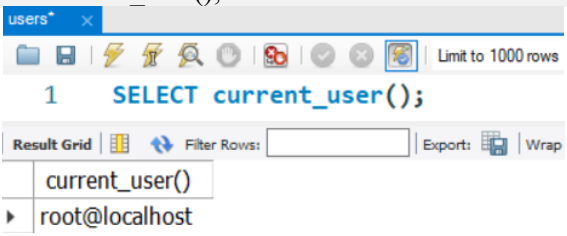
```
SELECT host, account_locked, password_expired FROM mysql.user;
```



72 - сурет. Пайдаланушылар тізімі

Ағымдағы пайдаланушы

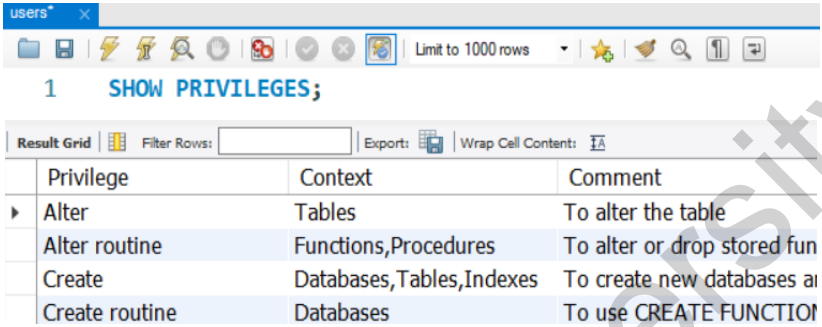
```
SELECT current_user();
```



73 - сурет. Ағымдағы пайдаланушы

Жүйедегі барлық құқықтар тізімін алу

```
SHOW PRIVILEGES;
```

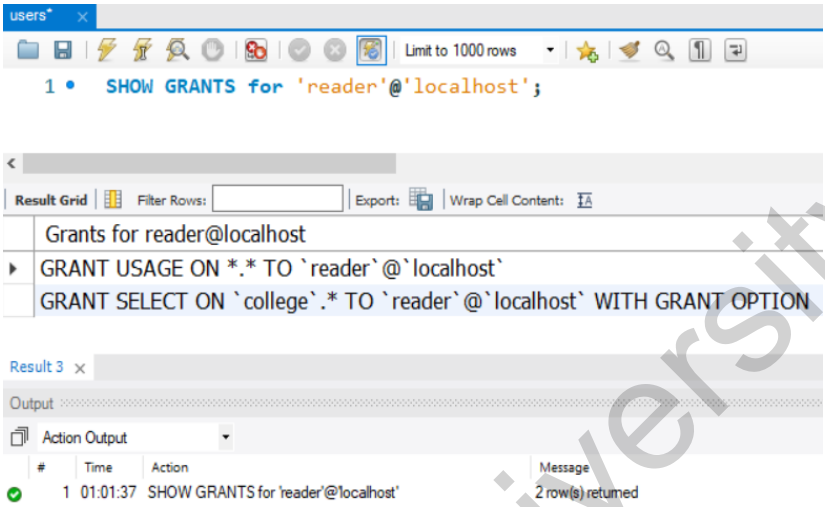


74 - сурет. Деректер қорын оқуға рұқсат беру

Ең жиі қолданылатын құқықтар мен құқық беру деңгейі. Кейбір құқықтар деңгейге байланысты қолжетімді және қолжетімсіз болады (3-кесте).

3-кесте

Құқық түрі	Global Глобадьді деңгей	Database Деректер қоры деңгейі	Table кесте деңгейі	Column баған деңгейі
ALL	+	+	+	-
ALTER	+	+	+	-
CREATE	+	+	+	-
CREATE TEMPORARY TABLES	+	+		
DELETE	+	+	+	-
DROP	+	+	+	-
INSERT	+	+	+	+
SELECT	+	+	+	+
SHOW DATABASE	+	-	-	-
UPDATE	+	+	+	+
EXECUTE	+	-	-	-



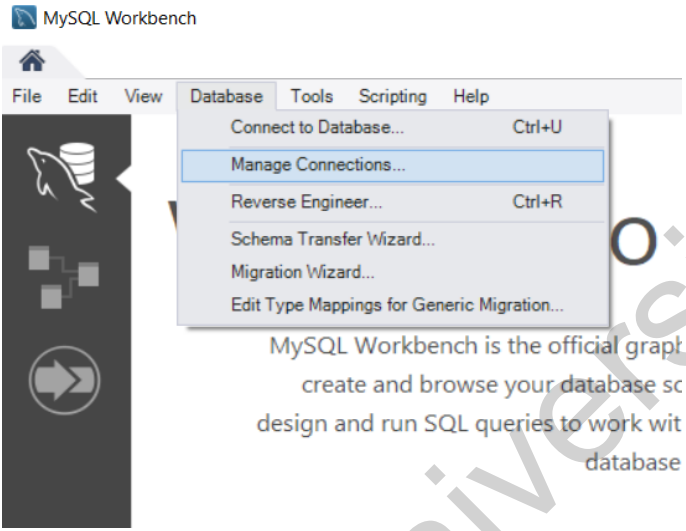
75 - сурет. Пайдаланушылар құқықтарын көру

GRANT, REVOKE немесе SET PASSWORD көмегімен орындалатын өзгертулер дереу күшіне енеді. Ал деректер қоры мен кестелерді өзгерту операторларына рұқсат берілсе (INSERT, UPDATE, DELETE және т.б.) FLUSH PRIVILEGES нұсқаулығын орындауыңыз керек. Бұл нұсқаулық пайдаланушыға құқықтар берілгеннен кейін орындалуы керек.

Ескерту! Жоғарыда көрсетілген басқа пайдаланушыларға құқық беру және алып тастау root пайдаланушы атыннан іске асырылды.

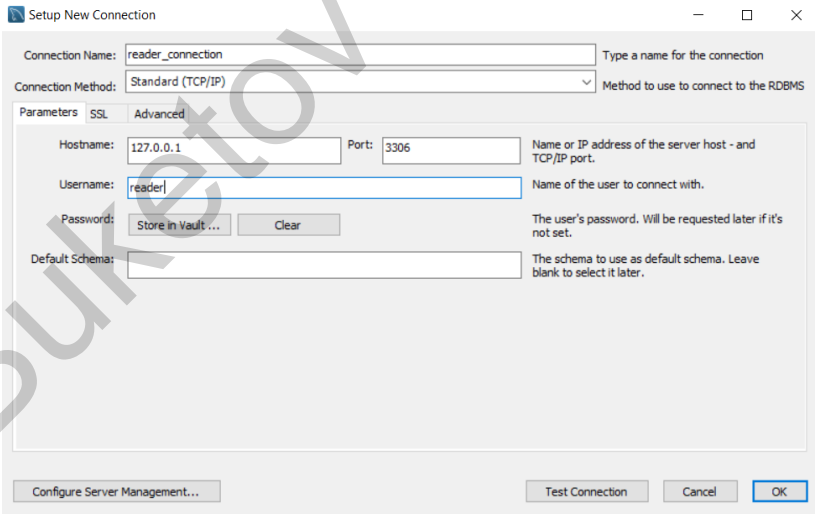
Пайдаланушыларға жеке байланыс (сеанс) ашу

Пайдаланушылар серверге қосылу үшін жеке сеанс құру қажет. Ол үшін Database мәзірндегі Manage Connections командасын таңдаңыз немесе MySQL Connection ⊕ батырмасын басыңыз.



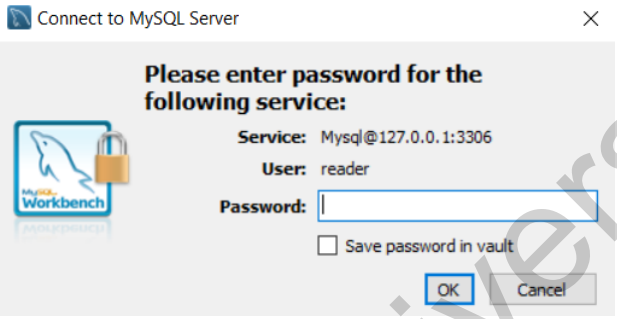
76 - сурет. Жаңа сеанс ашу

Байланыс атауы (reader_connection) мен пайдаланушы атын (reader) енгіземіз.

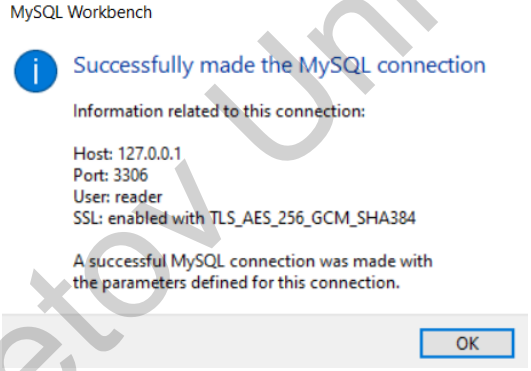


77 - сурет. Байланыс аты мен пайдаланушы атын енгізу

Байланысты тексерк үшін Test Connection батырмасын басып, пайдаланушының құпия кілтін енгіземіз. Байланыс дұрыс орнатылса, сәтті байланыс орнатылғаны туралы хабарламаны аласыз.

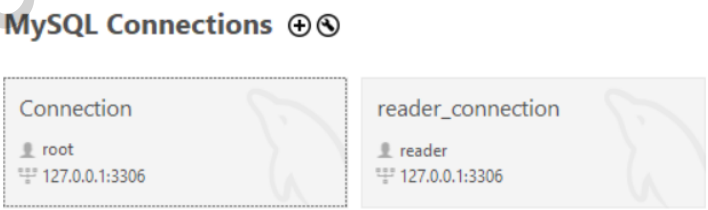


78.1 - сурет. Байланысты тексеру



78.2 - сурет. Байланысты тексеру

MySQL Connection тізімінде құрылған сеанс пайда болады.



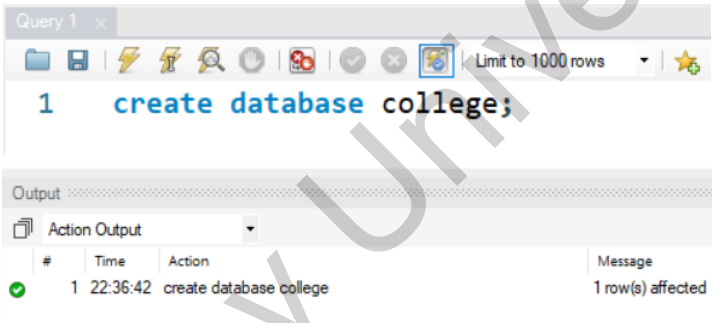
79 - сурет. Сеанстар

2 тарау. Деректер құрылымын анықтау

2.1 Деректер қорын құру және өшіру

CREATE DATABASE командасы арқылы берілген атаумен *деректер_қорының_атауы* деректер қорын құруға болады. Деректер қорының атауын қолданушы өзі анықтайды. Жаңа деректер қорын құрудың жалпы синтаксисі:

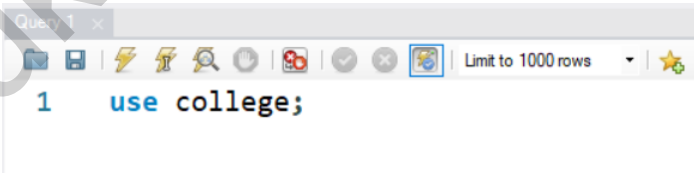
```
CREATE DATABASE деректер_қоры_атауы;
```



80 - сурет. Жаңа деректер қорын құру

Құрылған деректер қорын басқару үшін USE командасы қолданылады.

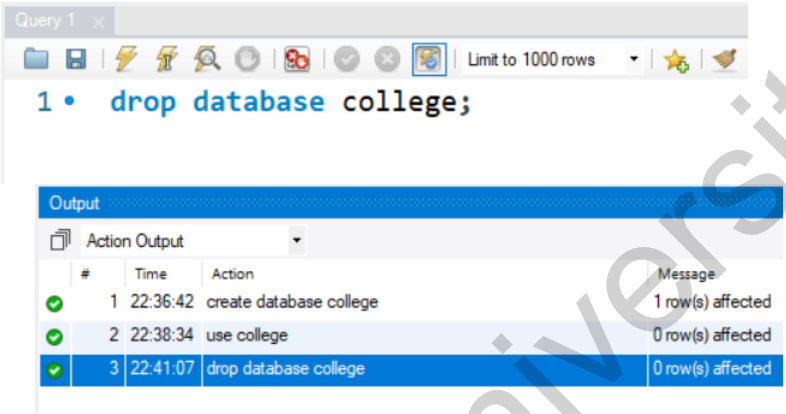
```
USE деректер_қорының_атауы;
```



81 - сурет. Деректер қорын қолдану

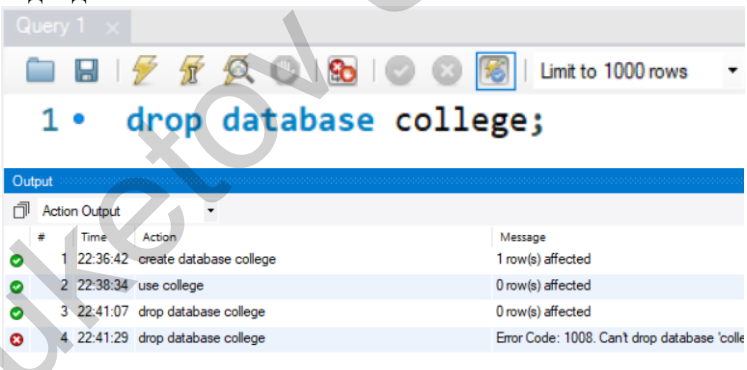
Мәліметтер қорын өшіру үшін DROP командасы қолданылады.

DROP DATABASE деректер_қорының_атауы;



82 - сурет. Деректер қорын өшіру

Серверде жоқ деректер қорын өшіргенде қателік туындайды.



83 - сурет. Скрипт орындауда туындаған қателік

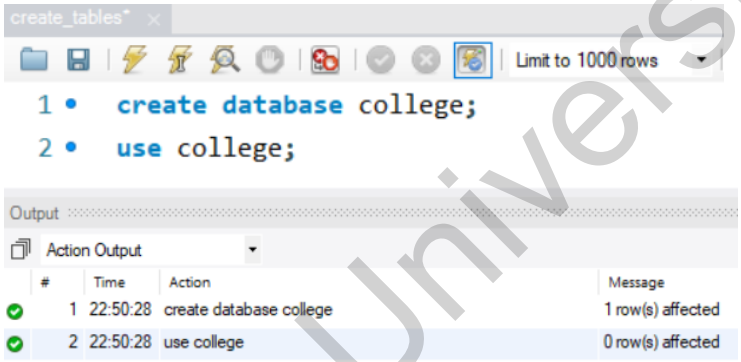
Туындаған қателік: Can't drop database 'college'; database doesn't exist – '*college*' – деректер қоры жоқ, өшіру мүмкін емес.

2.2 Кестелерді құру және өшіру

Кесте құру үшін кесте атауы,жақша ішінде кесте өрістерінің атауы, типі және атрибуттары көрсетіледі. Жаңа құрып көрейік.

Ең алдымен жаңа деректер қорын (*college*) құраңыз.

```
CREATE DATABASE college;
USE college;
```



84 - сурет. Жаңа деректер қорын құру және қолдану

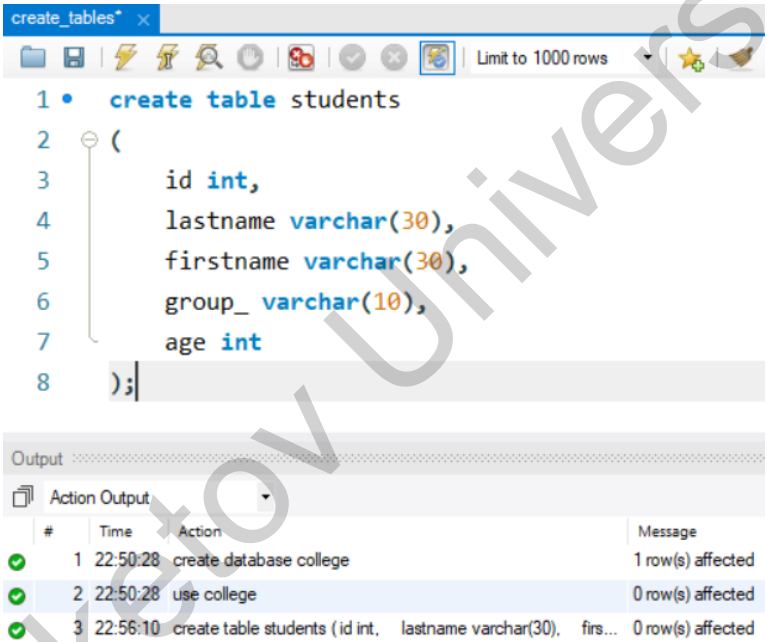
Жаңа кесте құрудың жалпы синтаксисі

```
CREATE TABLE кесте_атауы
(өріс1_атауы өріс1_типі өріс1_атрибуты,
 өріс2_атауы өріс1_типі өріс1_атрибуты,
...
өрісN_атауы өрісN_типі өрісN_атрибуты,
кесте_деңгейіндегі_атрибуттар)
```

Келесі құрылымдағы *students* кестесін құрыңыз:

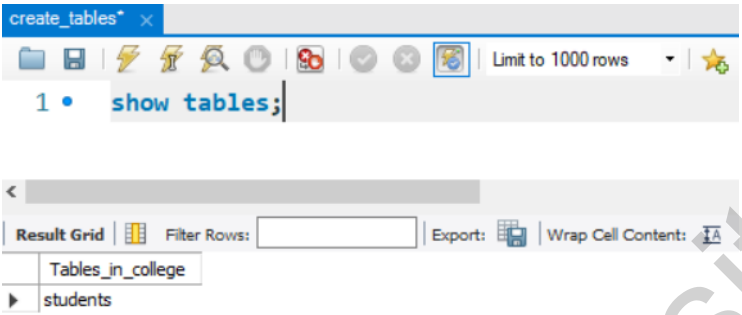
Өріс атауы	Өріс типі	Өріс атрибуты
id	INTEGER	
firstname	VARCHAR	
group_	VARCHAR	
age	INTEGER	

```
CREATE TABLE students
(
    id INT,
   lastname VARCHAR(30),
   firstname VARCHAR(30),
   group_ VARCHAR(10),
   age int
);
```



85 - сурет. Жаңа кесте құру

Ағымдағы *college* деректер қорындағы кестелер тізімін шығарайық. Деректер қорында құрылған *students* кестесі ғана бар.



86 - сурет. Кестелер тізімін алу

Кестедегі барлық жазбаларды өшіру үшін TRUNCATE командасы қолданылады.

TRUNCATE TABLE кесте_атауы;

Мысалы:

students кестесіндегі жазбаларды өшіру

TRUNCATE TABLE students;

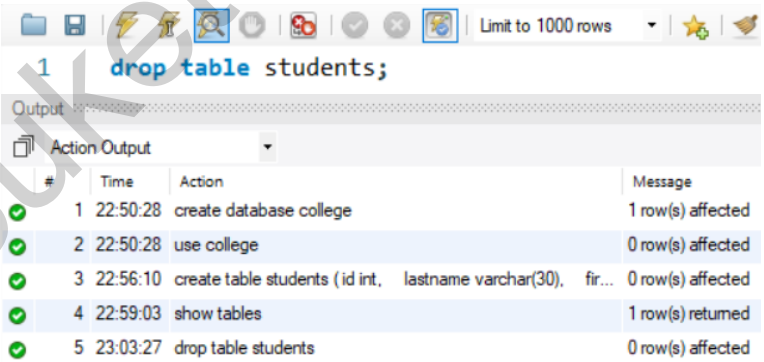
Кестені деректер қорынан мүлдем өшіріп тастау үшін DROP командасы қолданылады.

DROP TABLE кесте_атауы;

Мысалы:

students кестесін деректер қорынан өшіру

DROP TABLE students;



87 - сурет. Кестені өшіру

Ескерту! Егер кестеде жазбалар болса, бұл кестені бірден жоя алмайсыз, ең алдымен кестедегі барлық жазбаларды өшіріп тастау керек.

2.3 MySQL-дегі деректердің типтері

Кесте өрістерін (бағандарын) анықтаған кезде олар үшін деректер типін көрсету керек. Деректер типі бағанда қандай мәндерді сақтауға болатынын, олардың жадта қанша орын алатынын анықтайды.

MySQL – де деректер типтерін бірнеше топтарға жіктеледі.

4 кесте

Деректер типтері

Символдық	Сандық	Уақыт	Құрылымдық	Бинарлық
char() varchar() tinytext text blob mediumtext longtext	tinyint () smallint () mediumint() int () bigint () float double (,) decimal (,)	date datetime timestamp time year	enum set	tinyblob blob mediumblob largeblob

Символдық типтер

CHAR – ұзындығы нақты символдық тип. Жолдың ұзындығы символдардың санымен анықталады. CHAR типі 255 байтқа дейін сақтай алады.

Жолдың ұзындығы жақшада көрсетіледі, мысалы, CHAR(10) - он символдан тұратын жол. Ал егер осы бағандағы кестеде 6 таңбадан тұратын жол сақталса (яғни белгіленген ұзындығы 10 таңбадан аз), онда жол 4 бос орынмен толтырылады.

VARCHAR – өзгермелі ұзындығы бар символдық тип. Сақталған жолдың ұзындығы жақшада да көрсетіледі, мысалы, VARCHAR(10). VARCHAR типі 65535 байтқа дейін сақтай алады.

MySQL 5.6 нұсқасынан бастап CHAR және VARCHAR түрлері үнсіздік келісім бойынша UTF-8 кодтауын пайдаланады, бұл тілге байланысты таңбаны 3 байтқа дейін сақтауға мүмкіндік береді (көптеген еуропалық тілдер үшін әр таңбаға 1 байт, шығыс тілдері үшін - 2 байт, ал қытай, жапон, корей тілдері үшін - әр таңбаға 3 байт).

Бірнеше қосымша деректер типтері ұзындығы анықталмаған мәтінді білдіреді:

Ұзындығы белгісіз жолдық типтер:

- TINYTEXT – 255 байтқа дейінгі мәтінді ұсынады;
- TEXT – 65 КБ дейінгі мәтінді ұсынады;
- MEDIUMTEXT – ұзындығы 16 МБ дейінгі мәтінді ұсынады;
- LARGETEXT – 4 ГБ дейінгі мәтінді ұсынады.

Сандық типтер

TINYINT – (-128)- ден 127-ге дейінгі бүтін сандарды қамтиды, 1 байт орын алады.

BOOL – бұл тип тек 0 және 1 мәндерін сақтай алады. Алайда, бұл тип кірістірілген тұрақты мәндерді мән ретінде қабылдай алады TRUE (1 санын білдіреді) және FALSE (0 санын береді).

TINYINT UNSIGNED – 0-ден 255-ке дейінгі бүтін сандарды қабылдайды, 1 байт орын алады.

SMALLINT – (-32768)-ден 32767-ге дейінгі бүтін сандарды қабылдайды, 2 байт орын алады

SMALLINT UNSIGNED – 0-ден 65535-ке дейінгі бүтін сандарды қабылдайды, 2 байт орын алады.

MEDIUMINT – (-8388608)-ден 8388607-ге дейінгі бүтін сандарды қабылдайды, 3 байт орын алады.

MEDIUMINT UNSIGNED – 0-ден 16777215-ке дейінгі бүтін сандарды қабылдайды, 3 байт орын алады.

INT – (-2147483648)-ден 2147483647-ге дейінгі бүтін сандарды қабылдайды, 4 байт орын алады.

INT UNSIGNED – 0-ден 4294967295-ке дейінгі бүтін сандарды қабылдайды, 4 байт орын алады.

BIGINT – (-9 223 372 036 854 775 808)-ден 9 223 372 036 854 775 807-ге дейінгі бүтін сандарды қабылдайды, 8 байт орын алады.

BIGINT UNSIGNED – 0-ден 18 446 744 073 709 551 615-ке дейінгі бүтін сандарды қамтиды, 8 байт орын алады.

DECIMAL – белгіленген дәлдікпен сандарды сақтайды. Бұл тип precision және scale екі параметрін қабылдауы мүмкін: DECIMAL(precision, scale). Precision параметрі санды сақтай алатын цифрлардың максималды санын білдіреді. Бұл мән 1-ден 65-ке дейін болуы керек. Scale параметрі үтірден кейінгі мүмкін цифрлардың максималды санын білдіреді. Бұл мән 0-ден precision параметрінің мәніне дейін болуы керек. Әдепкі бойынша, ол 0-ге тең.

Мысалы, келесі өрісті анықтағанда: salary DECIMAL(5,2). Мұндағы: 5 саны-precision, ал 2 саны-scale, сондықтан бұл баған -999.99-дан 999.99-ға дейінгі диапазондағы мәндерді сақтай алады.

DECIMAL типінің жадыдан алатын орны, яғни өлшемі сақталатын мәнге байланысты. Бұл типтің NUMERIC, DEC, FIXED бүркеншік аттары бар.

FLOAT – $(-3.4028*10^{38})$-ден $3.4028*10^{38}$-ге дейінгі, бір дәлдіктегі қалқымалы нүктелі бөлшек сандарды қамтиды. Жадыдан 4 байт орын алады. FLOAT(M,D) түрінде болуы мүмкін, мұндағы M-цифрлардың жалпы саны, ал D - үтірден кейінгі цифрлар саны;

DOUBLE – $(-1.7976 * 10^{308})$-ден $1.7976 * 10^{308}$-ге дейінгі диапазондағы қос дәлдіктегі қалқымалы нүктесі бар бөлшек сандарды сақтайды, 8 байт орыналады. Ол сонымен қатар DOUBLE (M,D) түрінде болуы мүмкін, мұндағы m – цифрлар жалпы саны, ал D - үтірден кейінгі цифрлар саны. DOUBLE орнына қолдануға болатын REAL және DOUBLE PRECISION бүркеншік аттары бар.

Күн мен уақытпен жұмыс істеу үшін арналған типтер

• DATE – 1000 жылдың 1 қаңтарынан 9999 жылдың 31 желтоқсанына дейінгі даталарды сақтайды («1000-01-01» - ден «9999-12-31» - ге дейін). Әдетте, датаны сақтау үшін yyyy-mm-dd пішімі қолданылады. 3 байт орын алады;

• TIME – уақытты сақтайды -838:59:59 ден 838:59:59 дейін. Үнсіздік келісім бойынша «hh:mm:ss» форматында көрсетеді. 3 байт орын алады;

• DATETIME – уақыт пен күнді, күндер мен уақыт диапазонын біріктіреді - 1000 жылдың 1 қаңтарынан 9999 жылдың 31 желтоқсанына дейін («1000-01-01 00:00:00» дейін «9999-12-31 23:59:59»). Үнсіздік келісім бойынша «yyyy-mm-dd hh:mm:ss» пішімі қолданылады. 8 байт орын алады;

• TIMESTAMP – «1970-01-01 00:00:01 UTC» мен «2038-01-19 03:14:07 UTC» аралығындағы күн мен уақытты сақтайды, жадыдан 4 байт орын алады;

• YEAR – жыл 4 цифр түрінде сақталады. Қол жетімді мәндер ауқымы 1901-ден 2155-ке дейін. 1 байт орын алады.

DATE типі күндерді әртүрлі форматтарды қабылдай алады, бірақ деректер қорында сақтау үшін тікелей күндер «yyyy-mm-dd» форматына келтіріледі. Кейбір қабылданған пішімдер:

– yyyy-mm-dd - 2018-05-25
– yyyy-m-dd - 2018-5-25
– yy-m-dd - 18-05-25
– yyyymmdd - 20180525
– yyyy.mm.dd - 2018.05.25

Уақыт үшін TIME типі 24 сағаттық форматты қолданады. Ол уақытты әртүрлі пішінде қабылдай алады:

– hh:mi - 3:21 (сақталатын мән 03:21:00)
– hh:mi:ss - 19:21:34
– hhmiss - 192134
– DATETIME және TIMESTAMP түрлеріне арналған мысалдар:
– 2018-05-25 19:21:34
– 2018-05-25(сақталатын мән) 2018-05-25 00:00:00)

Құрама типтер

• ENUM – бір мәнді мәндер тізімінен сақтайды. 1-2 байт орын алады;

• SET – рұқсат етілген мәндер тізімінен бірнеше мәндерді (64 мәнге дейін) сақтай алады. 1-8 байт алады.

Бинарлы типтер

• TINYBLOB – екілік деректерді ұзындығы 255 байтқа дейінгі жол ретінде сақтайды;

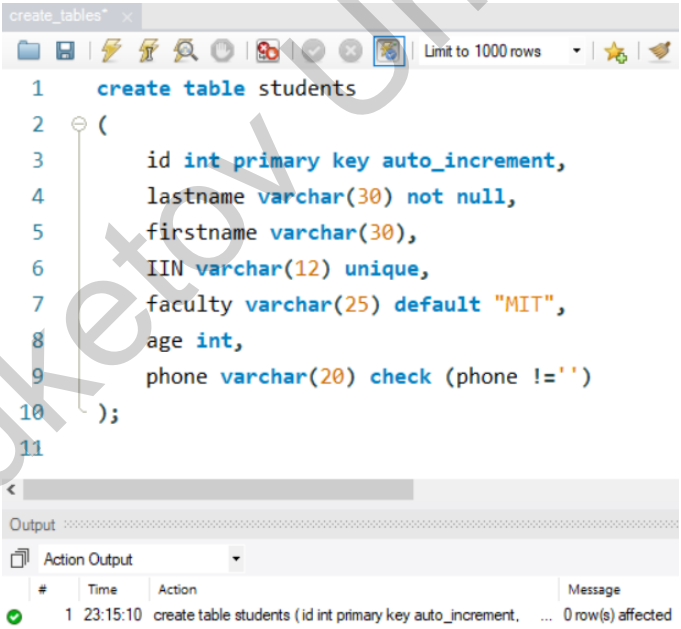
• BLOB – екілік деректерді ұзындығы 65 КБ дейінгі жол түрінде сақтайды;

• MEDIUMBLOB – екілік деректерді ұзындығы 16 МБ дейінгі жол ретінде сақтайды;

• LARGEBLOB – екілік деректерді ұзындығы 4 ГБ дейінгі жол ретінде сақтайды.

2.4 Өріс және кесте атрибуттары

students кестесін құрайық. Кесте құруда өрістер үшін арнайы атрибуттарды көрсетуге болады.



88 - сурет. Кесте құру

```
CREATE TABLE students
(
   id INT PRIMARY KEY AUTO_INCREMENT,
   lastname VARCHAR(30) NOT NULL,
   firstname VARCHAR(30),
   IIN VARCHAR(12) UNIQUE,
   faculty VARCHAR(25) DEFAULT "MIT",
   age INT,
   phone VARCHAR(20) CHECK (phone !='')
);
```

Кесте құруда қолданылған атрибуттардың әр қайсысына тоқталып өтейік:

PRIMARY KEY

PRIMARY KEY атрибуты кестенің бастапқы кілтін орнатады.

```
CREATE TABLE students
(
   id INT PRIMARY KEY,
   lastname VARCHAR(30)
);
```

Бастапқы кілт кестедегі өрісті бірегей (уникалды, қайталанбайтын) түрде анықтайды. Бастапқы кілт ретінде анықталған өріс *int* типті болуы міндетті емес, кез келген басқа типті бола алады.

Бастапқы кілтті кесте деңгейінде орнату:

```
CREATE TABLE students
(
   id INT,
   lastname VARCHAR(30)
   PRIMARY KEY(id)
);
```

Бастапқы кілт құрама болуы мүмкін. Мұндай кілт арқылы бірнеше өрісті кілттік өріс ретінде тағайындауға болады. Мысалы:

```
CREATE TABLE students
(
   id INT,
   lastname VARCHAR(30)
   PRIMARY KEY(id, lastname)
);
```

Мұнда *id* және *lastname* өрістері құрама бастапқы кілт ретінде көрсетілген. Яғни, *students* кестесінде *id* және *lastname* өрістерінде жолдар (мәндер) қайталанбайды.

AUTO_INCREMENT

AUTO_INCREMENT атрибутын бүтін немесе нақты сандық типті өрістер үшін қолданылады. Жаңа жол қосылған кезде жол мәні автоматты түрде бір санға артып отырады.

```
CREATE TABLE students
(
   id INT PRIMARY KEY AUTO_INCREMENT,
   lastname VARCHAR(30),
   firstname VARCHAR(30)
);
```

Бұл жағдайда әрбір жаңа қосылған жазбаның *id* өрісінің мәні бірлікке артады.

UNIQUE

UNIQUE атрибуты өрістің текбірегей (уникалды) мәндерді сақтай алатындығын көрсетеді.

```
CREATE TABLE students
(
   id INT,
   lastname VARCHAR(30),
   firstname VARCHAR(30),
   IIN VARCHAR(12) unique
);
```

Берілген мысалда *IIN* өрісі тек ерекше (қайталанбайтын) мәндерді сақтай алады. Бұл өріске мәндері бірдей мәндерді қоса алмаймыз.

Бұл атрибуты кесте деңгейінде де анықтай аламыз:

```
CREATE TABLE students
```

```
(
   id INT,
   lastname VARCHAR(30),
   firstname VARCHAR(30),
   IIN VARCHAR(12),
   UNIQUE(IIN)
);
```

NULL және NOT NULL

Өрістің NULL мәнін қабылдай алатындығын көрсету үшін NULL немесе NOT NULL атрибутын орнатуға болады. Егер бұл атрибут нақты қолданылмаса, онда үнсіздік келісім бойынша жол NULL мәнімен толтырылады. Бірақ бастапқы кілт ретінде анықталған өріс міндетті түрде NOT NULL атрибутына ие болады.

```
CREATE TABLE students
(
   id INT PRIMARY KEY,
   lastname VARCHAR(30) NOT NULL,
   firstname VARCHAR(30),
);
```

id бағаны бастапқы кілт болғандықтан NOT NULL мәніне, ал *firstname* өрісі үнсіздік келісім бойынша NULL атрибутына ие. Жаңа жазба қосу кезінде міндетті түрде *id, lastname* өрістеріне мән беріледі, әйтпесе қателік туындайды.

DEFAULT

DEFAULT атрибуты өрістің үнсіздік келісім бойынша (әдепкі) мәнін анықтайды. Егер өріске деректерді қосу кезінде мән берілмесе, онда ол үшін DEFAULT мән қолданылады.

```
CREATE TABLE students
(
   id INT PRIMARY KEY AUTO_INCREMENT,
   lastname VARCHAR(30) NOT NULL,
   firstname VARCHAR(30),
   faculty VARCHAR(25) DEFAULT "MIT",
);
```

Мұнда *faculty* өрісіне үнсіздік келісім бойынша (деректерді қосу кезінде мән берілмесе) «MIT» мәніне ие.

CHECK

CHECK атрибуты өрісте сақталатын мәндер ауқымына шарттар арқылы шектеу қояды. Бұл атрибутты орнату үшін CHECK кілттік сөзінен жақшада бағанға немесе бірнеше бағанға шарт көрсетіледі.

```
CREATE TABLE students
(
   id INT PRIMARY KEY,
   lastname VARCHAR(30),
   age INT CHECK(age >17 AND age < 100),
   phone VARCHAR(20) check (phone !='')
);
```

age өрісіне *age* >17 AND *age* < 100 шарты қойылды. Бұл өріс тек 17-ден үлкен және 100-ден кіші мәндерді ғана қабылдайды.

phone өрісіне *phone !=' '* шарты қойылды. Бұл өріске енгізілген мән бос орынға тең болмауы керек.

Шарттарды салыстыру операторлары мен логикалық амалдар арқылы беріледі.

5-кесте

Салыстыру операторлары

Салыстыру операторлары	Атауы	Мысал
<	кіші	age<100
>	үлкен	age>17
<=	кіші немесе тең	age<=100
>=	үлкен немесе тең	age>=17
==	тең	age==18
!=	тең емес	age!=18

6-кесте

Логикалық амалдар

Логикалық амалдар	Атауы	Мысал
AND	логикалық «және»	age>17 AND age< 100
OR	логикалық «немесе»	age>17 OR age > 18
NOT	жоққа шығару	NOT(age ==17)

CHECK атрибутын кесте деңгейінде де анықтауға болады:

```
CREATE TABLE students
(
   id INT PRIMARY KEY,
   lastname VARCHAR(30),
   age INT,
   phone VARCHAR(20),
   CHECK((age >17 AND age < 100) and (phone !=''))
);
```

CONSTRAINT операторы. Шектеулер (атрибут) атауын орнату

CONSTRAINT кілттік сөзі арқылы шектеу (атрибут) үшін атау беруге болады. Олар кесте деңгейіндегі атрибуттардың алдында CONSTRAINT кілттік сөзі арқылы көрсетіледі:

```
CREATE TABLE students
(
   id INT AUTO_INCREMENT,
   lastname VARCHAR(30),
   firstname VARCHAR(30),
   IIN VARCHAR(12) UNIQUE,
   age INT,
   phone VARCHAR(20) CHECK (phone !=''),
   CONSTRAINT student_id PRIMARY KEY(id),
   CONSTRAINT student_IIN UNIQUE(IIN)
);
```

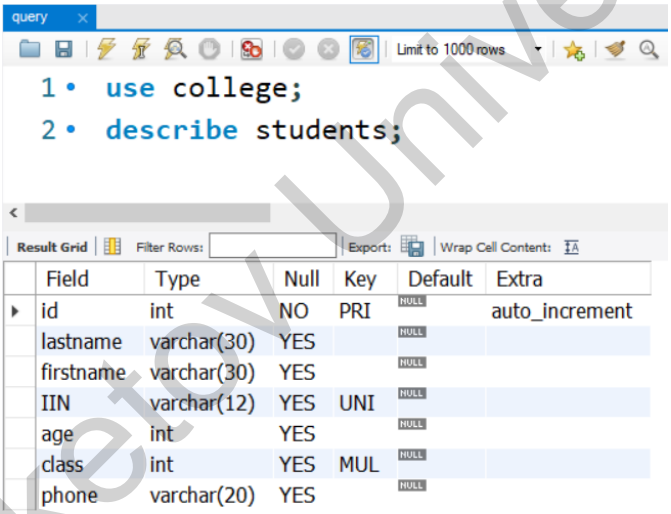
Бұл жағдайда PRIMARY KEY(*id*) атрибутына *student_id*, UNIQUE(*IIN*) үшін - *student_IIN* атауы берілді. Шектеу атауларын қолданып өрістер атрибуттарын алыптастауға және өзгертуге болады.

PRIMARY KEY, CHECK, UNIQUE, FOREIGN KEY атрибуттарына атау беру (CONSTRAINT) қарастырылған.

Кестенің сипаттамасын алу

Құрылған кестенің өрістерін және олардың типі мен атрибуттарын көру үшін DESCRIBE немесе DESC операторы қолданылады.

```
DESCRIBE кесте_атауы;
```



89 - сурет. DESCRIBE операторы

2.5 FOREIGN KEY сыртқы кілттер

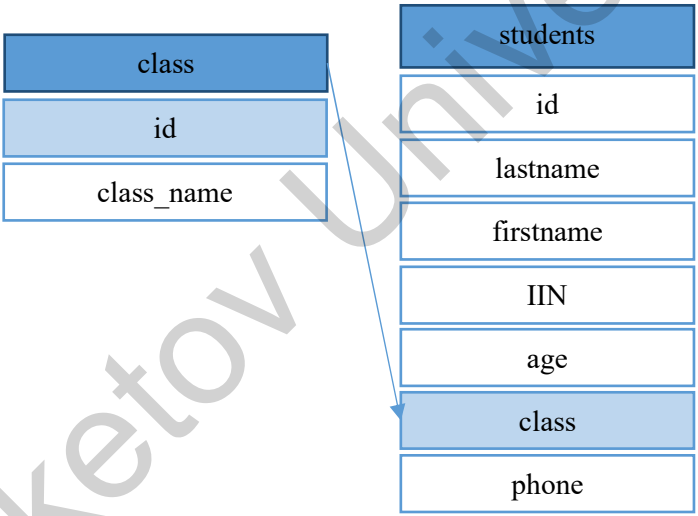
Сыртқы кілттер кестелер арасында байланыс орнатуға мүмкіндік береді. Сыртқы кілт тәуелді, бағынышты кестедегі өрістер үшін орнатылады және негізгі кестедегі өрістердің бірін

көрсетеді. Әдетте, сыртқы кілт байланыстырылған негізгі кестедегі бастапқы кілтті көрсетеді.

Кесте деңгейінде сыртқы кілтті орнатудың жалпы синтаксисі:

```
[CONSTRAINT атрибут_атауы]
FOREIGN KEY(өріс1, өріс2, ... өрісN)
REFERENCES негізгі_кесте (өріс_1, өріс_2, ... өріс_N)
[ON DELETE оператор]
[ON UPDATE оператор]
```

college деректер қорында екі кесте құрылсын. *class* кестесінде топтар туралы деректер, ал *students* кестесінде студенттер туралы деректер сақталады.

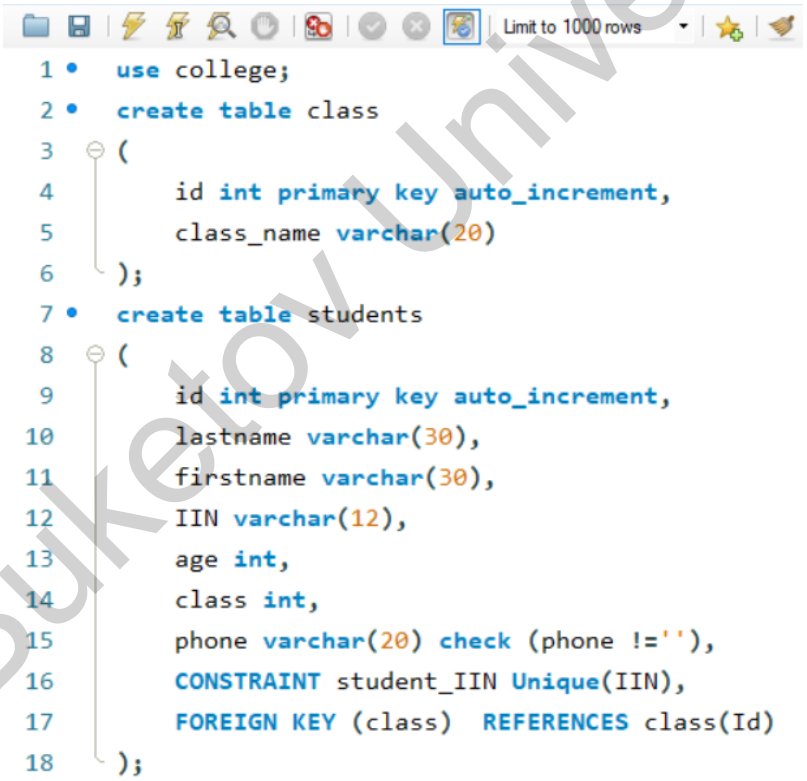


2 - сызба. Кестелер сызбасы

class және *students* кестелерін құру

```
USE college;
CREATE TABLE class
(
    id INT PRIMARY KEY AUTO_INCREMENT,
   class_name VARCHAR(20)
);
```

```
CREATE TABLE students
(
   id INT PRIMARY KEY AUTO_INCREMENT,
   lastname VARCHAR(30),
   firstname VARCHAR(30),
   IIN VARCHAR(12),
   age INT,
   class INT,
   phone VARCHAR(20) CHECK (phone !=''),
   CONSTRAINT student_IIN UNIQUE(IIN),
   FOREIGN KEY (class) REFERENCES class(Id)
);
```



90 - сурет. *class* және *students* кестелерін құру

college деректер қорында *class* және *students* кестелері құрылды.

class – негізгі кесте болып табылады, топтар тізімін сақтайды. *students* – бағынышты кесте, студенттер туралы ақпараттарды сақтайды.

students кестесіндегі *class* өрісі арқылы *class* кестесіндегі *id* өрісі арқылы байланысқан. *students* кестесіндегі *class* өрісі – сыртқы кілт болып табылады, ол *class* кестесіндегі *id* өрісін көрсетеді.

CONSTRAINT операторының көмегімен сыртқы кілт үшін атау орнатуға болады.

Сыртқы кілтке CONSTRAINT арқылы атау беру

```
CREATE TABLE students
(
   id INT PRIMARY KEY AUTO_INCREMENT,
   lastname VARCHAR(30),
   firstname VARCHAR(30),
   IIN VARCHAR(12),
   age INT,
   class INT,
   phone VARCHAR(20) CHECK (phone !=''),
   CONSTRAINT student_IIN UNIQUE(IIN),
   CONSTRAINT class_students FOREIGN KEY (class) REFERENCES class(Id)
);
```

ON DELETE және ON UPDATE

ON DELETE және ON UPDATE қолдана отырып, негізгі кестеден байланысты жолды жою және өзгерту кезінде орындалатын функцияларды (іс-әрекетті) анықтайды.

CASCADE – негізгі кестедегі байланысты жолдарды жою немесе өзгерту кезінде тәуелді кестеден жолдарды автоматты түрде жояды немесе өзгертеді.

SET NULL – негізгі кестеден байланысты жолды жою немесе жаңарту кезінде сыртқы кілт бағанына NULL мәнін орнатады. (Бұл жағдайда сыртқы кілт бағаны NULL орнатуды қолдауы керек).

RESTRICT – тәуелді кестеде байланысқан жолдар болған кезде негізгі кестедегі жолдарды жоюды немесе өзгертуді қабылдамайды.

NO ACTION – RESTRICT-пен бірдей.

SET DEFAULT – байланысты жолды негізгі кестеден алып тастағанда, сыртқы кілт бағанына әдепкі мәнді орнатады, ол DEFAULT атрибуты арқылы орнатылады.

Каскадты жою

Каскадты жою негізгі кестеден (*class*) жазбаны жойған кезде, осы жазбаға қатысты тәуелді кестеден барлық жазбаларды автоматты түрде жоюға мүмкіндік береді. Ол үшін CASCADE (*students*) опциясы қолданылады.

```
CREATE TABLE students
(
   id INT PRIMARY KEY AUTO_INCREMENT,
   lastname VARCHAR(30),
   firstname VARCHAR(30),
   IIN VARCHAR(12),
   age INT,
   class INT,
   phone VARCHAR(20) CHECK (phone !=''),
   CONSTRAINT student_IIN UNIQUE(IIN),
   CONSTRAINT class_students FOREIGN KEY (class) REFERENCES class(Id) ON DELETE CASCADE
);
```

ON UPDATE CASCADE опциясы да дәл осылай жұмыс істейді. Бастапқы кілттің мәні өзгерген кезде, байланысты сыртқы кілттің мәні автоматты түрде өзгереді. Алайда, бастапқы кілттер өте сирек өзгеретіндіктен және негізінен өзгеретін мәндері бар бағандарды бастапқы кілттер ретінде пайдалану ұсынылмайды, іс жүзінде ON UPDATE опциясы сирек қолданылады.

NULL орнату

Сыртқы кілт үшін SET NULL параметрін орнату керек, сонда сыртқы кілт өрісі NULL мәнін қабылдау алады:

```
CREATE TABLE students
(
   id INT PRIMARY KEY AUTO_INCREMENT,
   lastname VARCHAR(30),
   firstname VARCHAR(30),
   IIN VARCHAR(12),
   age INT,
   class INT,
   CONSTRAINT class_students FOREIGN KEY (class)
REFERENCES class(Id) ON DELETE SET NULL
);
```

2.6 Кестелер мен өрістерді өзгерту

Құрылған кестені өзгерту үшін ALTER TABLE командасы қолданылады. Оның қысқартылған синтаксисі:

ALTER TABLE кесте_атауы
{ ADD баған_атауы дерек_типі [атрибуттар] |
 DROP COLUMN баған_атауы |
 MODIFY COLUMN баған_атауы дерек_типі [атрибуттар] |
 ALTER COLUMN баған_атауы SET DEFAULT үнсіздік_бойынша_мән |
 ADD [CONSTRAINT] шектеу_аты |
 DROP [CONSTRAINT] шектеу_аты}

7-кесте

Кестені өзгерту үшін қолданылатын операторлар

№	Команда	Атқаратын функция
1	ADD баған_атауы	баған қосу
2	DROP COLUMN баған_атауы	бағанды өшіру
3	MODIFY COLUMN баған_атауы	бағанның типін, атрибуттарын өшіру

7-кесте жалғасы

№	Команда	Атқаратын функция
4	ALTER COLUMN баған_атауы SET DEFAULT	Бағанға үнсіздік кеісім бойынша мән орнату
5	ADD [CONSTRAINT] шектеу_аты	кесте деңгейінде атрибутар қосу. Шектеуге атау беру
6	DROP [CONSTRAINT] шектеу_аты	Шектеулерді өшіру

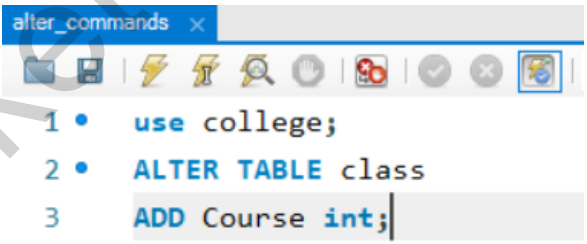
Жалпы, бұл команда көптеген опциялар мен мүмкіндіктерді қолдайды. Олардың барлығын *mysql.com* ресми сайтындағы құжаттамадан (Documentation) көруге болады. Біз жиі кездестіретін негізгі сценарийлерді ғана қарастырамыз.

Құрылған кесте

```
CREATE TABLE class
(
    id INT PRIMARY KEY AUTO_INCREMENT,
   class_name VARCHAR(20)
);
```

Жаңа өрісті қосу

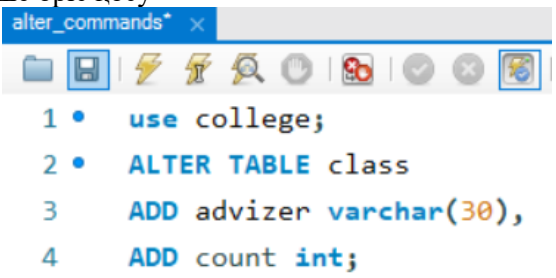
class кестесіне жаңа *int* типті *Course* өрісін қосу:



91 - сурет. Кестеге өріс қосу

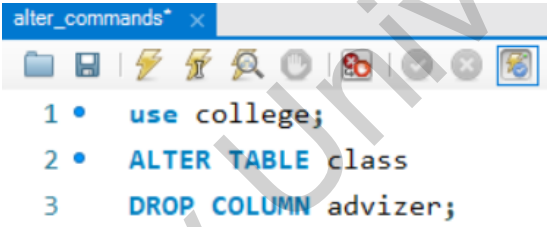
```
USE college;
ALTER TABLE class
ADD Course INT;
```

Бірнеше өріс қосу



92 - сурет. Кестеге бірнеше өріс қосу

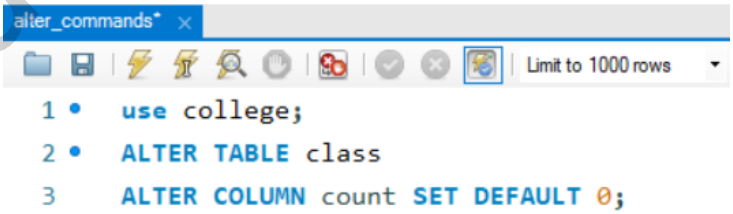
Өрісті жою
class кестесінен *advizer* өрісін жою:



93 - сурет. Кестеден өрісті өшіру

```
USE college;
ALTER TABLE class
DROP COLUMN advizer;
```

Үнсіздік келісім бойынша өріс мәнін өзгерту
count өрісіне үнсіздік келісім бойынша 0 мәнін орнату:

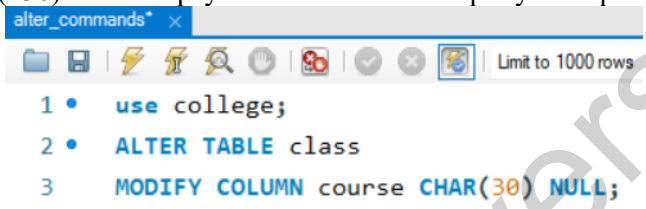


94 - сурет. Өріске DEFAULT атрибутын қосу

```
USE college;
ALTER TABLE class
ALTER COLUMN count SET DEFAULT 0;
```

Өріс типін өзгерту

class кестесіндегі *course* өрісіндегі деректер типін CHAR(130) етіп өзгерту және оған NULL атрибутын орнату:

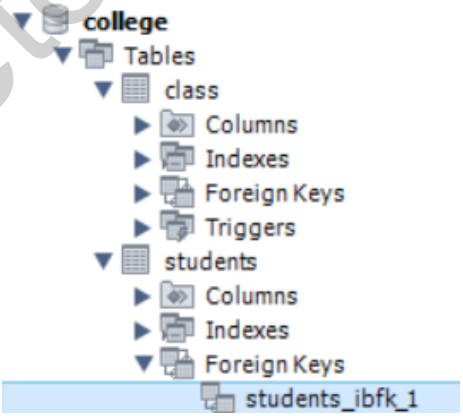


95 - сурет. Өріс типін өзгерту

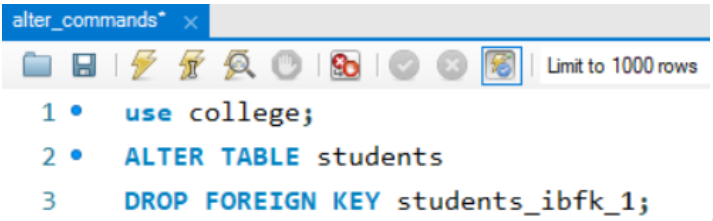
```
USE college;
ALTER TABLE class
MODIFY COLUMN course CHAR(30) NULL;
```

Сыртқы кілтті қосу және жою

students кестесінде *students.class* өрісі *class* кестесімен байланыс орнату үшін *class.id* өрісімен сыртқы кілт орнатылған. Бағдарлама бұл сыртқы кілтке *students_ibfk_1* атау берілді. Осы сыртқы кілтті өшіруге болады.



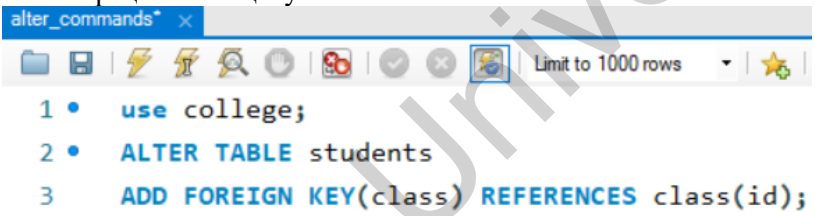
96 - сурет. Кесте құрылымы. Сыртқы кілттер



97 - сурет. Сыртқы кілтті өшіру.

```
USE college;
ALTER TABLE students
DROP FOREIGN KEY students_ibfk_1;
```

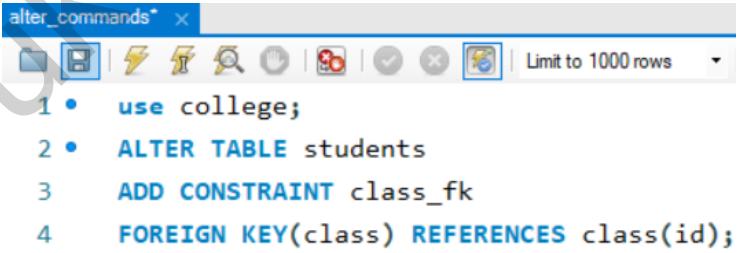
Сыртқы кілтті қосу



98 - сурет. Сыртқы кілтті қосу

```
USE college;
ALTER TABLE students
ADD FOREIGN KEY(class) REFERENCES class(id);
```

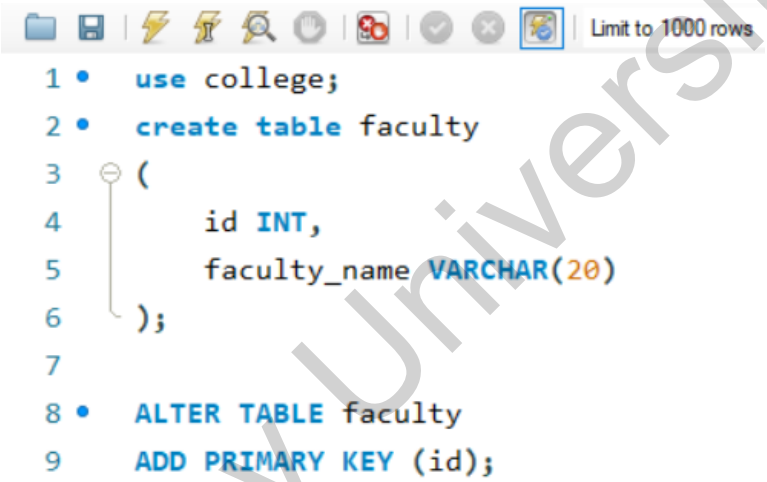
Сыртқы кілтті CONSTRAINT арқылы қосу



99 - сурет. Сыртқы кілтті қосу және оған атау беру

```
USE college;
ALTER TABLE students
ADD CONSTRAINT class_fk
FOREIGN KEY(class) REFERENCES class(id);
```

Бастапқы кілтті қосу және жою
Құрылған *faculty* кестесіне бастапқы кілтті қосу.

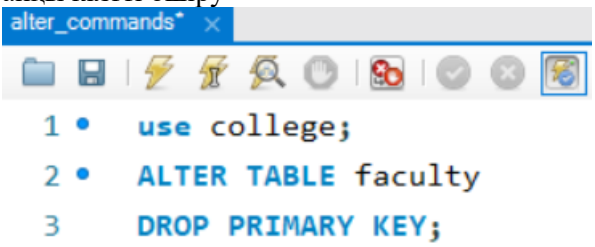


100 - сурет. Кестеге бастапқы кілтті қосу

```
USE college;
CREATE TABLE faculty
(
  id INT,
  faculty_name VARCHAR(20)
);

ALTER TABLE faculty
ADD PRIMARY KEY (id);
```

Бастапқы кілтті өшіру



101 - сурет. Бастапқы кілтті өшіру

```
USE college;
ALTER TABLE faculty
DROP PRIMARY KEY;
```

2.7 Комментарийлер

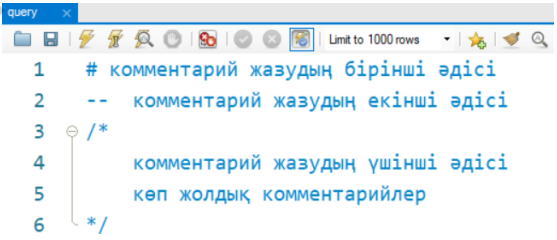
Комментарий – бұл бағдарламашыларға арналған және компилятор өңдемейтін мәтін.

Скрипте (сұраныста) комментарий жазудың әдістер

1) # символы арқылы
2) - - символдары (2 сызықша) арқылы
3) /* */ символдары арқылы(көп жолдық комментарийлер)

Мысалы:

```
#   комментарий жазудың бірінші әдісі
--  комментарий жазудың екінші әдісі
/*
    комментарий жазудың үшінші әдісі
   көп жолдық комментарийлер
*/
```



102 - сурет. Комментарий жазу

3 тарау. Деректермен орындалатын негізгі операциялар

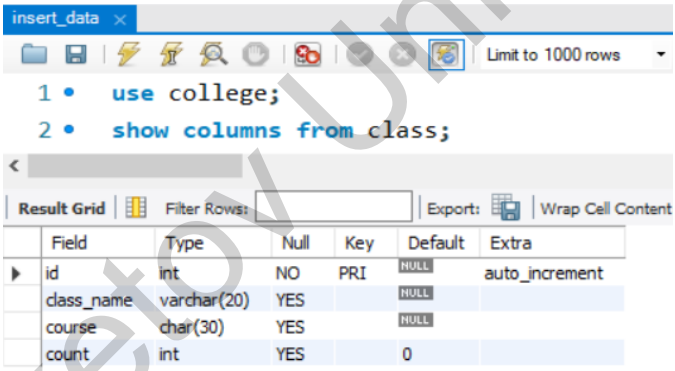
3.1 Деректерді қосу. INSERT командасы

MySQL-де деректер қорына деректерді қосу үшін INSERT командасы қолданылады:

Жалпы синтаксисі:

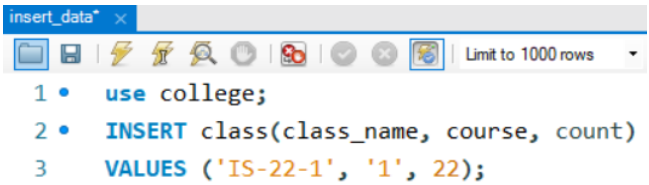
```
INSERT [INTO] кесте_атауы
[(өрістер_тізімі)]
VALUES (мән1, мән2, ... мәнN)
```

Жаңа жолды қосу үшін INSERT INTO кілттік сөзінен кейін кесте атауы, жақша ішінде өрістердің тізімі жазылады және VALUES кілттік сөзінен кейін әр өрістің мәндерді көрсетіледі.



103 - сурет. *class* кестесіндегі өрістер

class кестесіне жаңа жазба қосу.



104 - сурет. *class* кестесіне жазба қосу

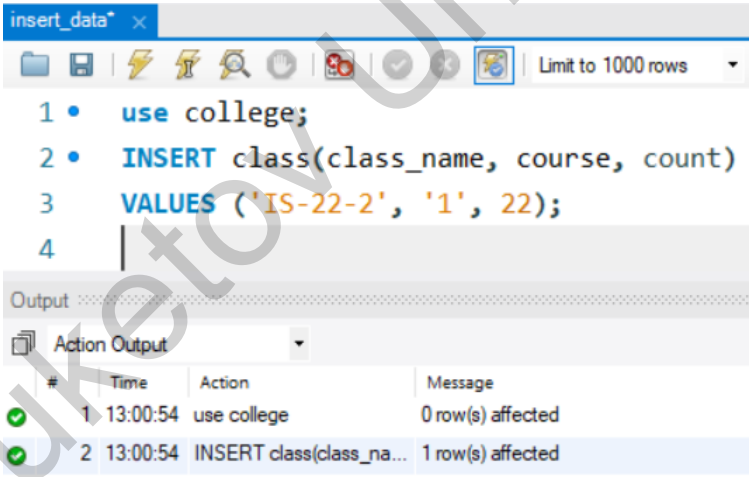
```
USE college;
INSERT class(class_name, course, count)
VALUES ('IS-22-1', '1', 22);
```

VALUES сөзінен кейінгі мәндер саны өрістер санына сәйкес келу керек. Енгізілетін дерек типі өріс типіне міндетті түрде сәйкес келуі керек.

class кестесіне қосылған жаңа жазба

№	Өріс атауы	Мәні
1	class_name	'IS-22-1'
2	course	'1'
3	count	22

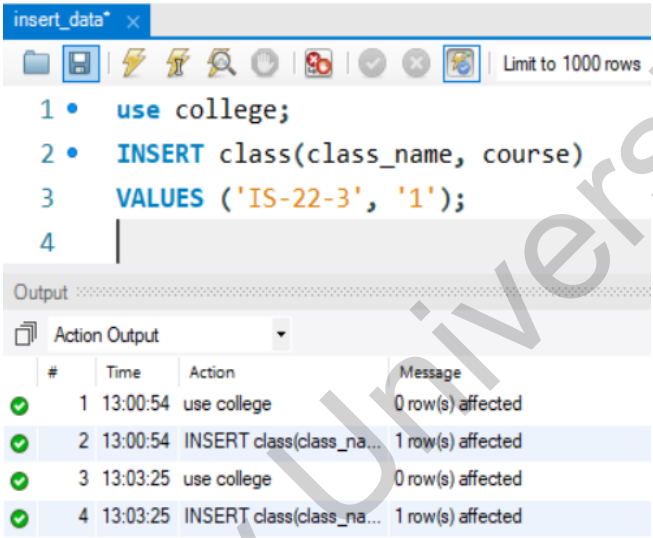
MySQL Workbench ортасында жазбаны қосу сәтті орындалса, Output терезесінде жасыл түсті маркермен «*1 row(s) affected*» хабарламасы шығады.



105 - сурет. *class* кестесіне жазба қосу

class кестесіне жазба қосуда *id* өрісі көрсетілген жоқ. Себебі *id* өрісі AUTO_INCREMENT атрибуты ие, бұл өріс автоматты түрде анықталады.

Жаңа жазба қосуда NULL немесе DEFAULT атрибуттарына ие өрістерді көрсетпеуге де болады. Мысалы: *class* кестесінде *count* өрісі DEFAULT атрибутымен анықталған. Бұл өріске мән бермеген жағдайда үнсіздік келісім бойынша мән беріледі.



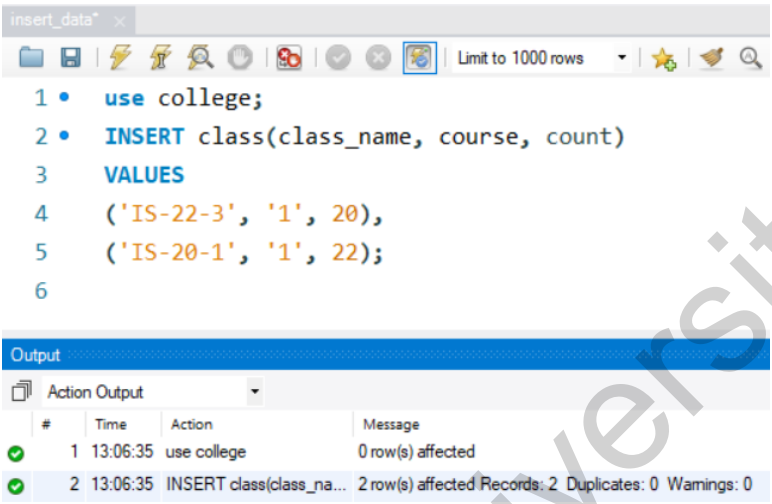
106 - сурет. *class* кестесіне жазба қосу

```
USE college;
INSERT class(class_name, course)
VALUES ('IS-22-3', '1');
```

Бірнеше жазба қосу

Кестеге бір сұраныспен бірнеше жазба қосуға мүмкіндік бар.

```
USE college;
INSERT class(class_name, course, count)
VALUES
('IS-22-3', '1', 20),
('IS-20-1', '1', 22);
```



107 - сурет. Кестеге бірнеше жазба қосу

Нәтижесінде, INSERT INTO кесте атауы *class*, жақша ішінде өрістер тізімі *class_name, course, count* жазылады және VALUES кейін жазбалардың тізімі көрсетіледі.

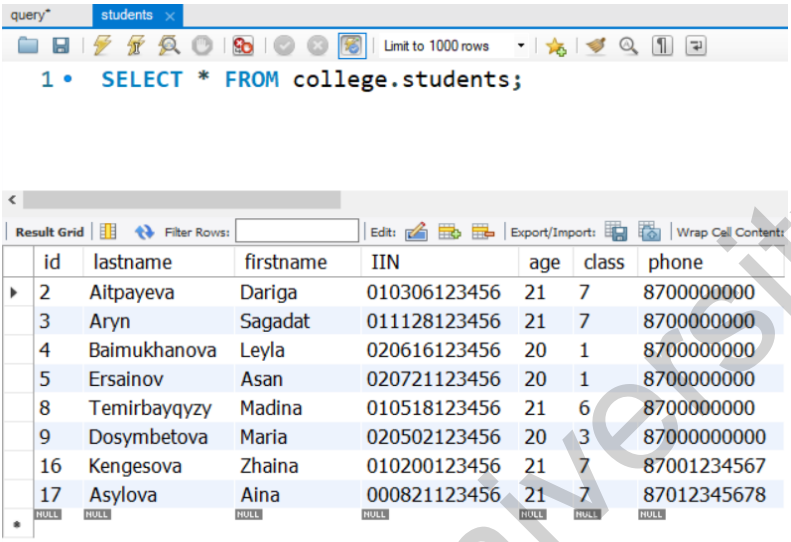
Қосылған жолдар

class_name	course	count
IS-22-3	1	20
IS-20-1	1	22

Көрсетілген мысалда *class* кестесіне 2 жол қосылды (2 rows affected).

3.2 Деректерді қосу. REPLACE операторы

REPLACE операторы кестеге жазба қосу (қайта қосу) немесе кестедегі жазбаны алмастыру үшін қолданылады.

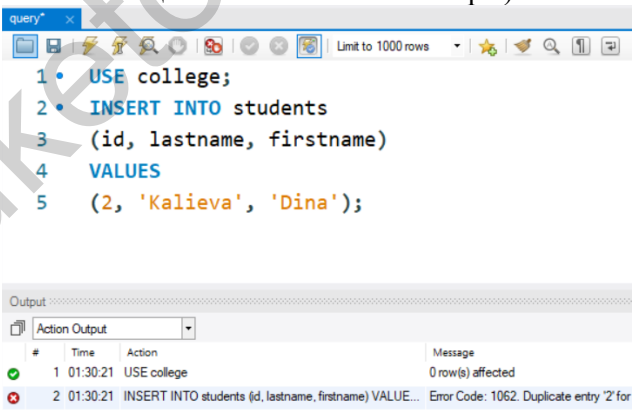


id	lastname	firstname	IIN	age	class	phone
2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
9	Dosymbetova	Maria	020502123456	20	3	87000000000
16	Kengesova	Zhaina	010200123456	21	7	87001234567
17	Asylova	Aina	000821123456	21	7	87012345678
NULL	NULL	NULL	NULL	NULL	NULL	NULL

108 - сурет. students кестесі

students кестесінде *id*=2 болатын студент жазбасы (Aitpaeva Dariga) бар.

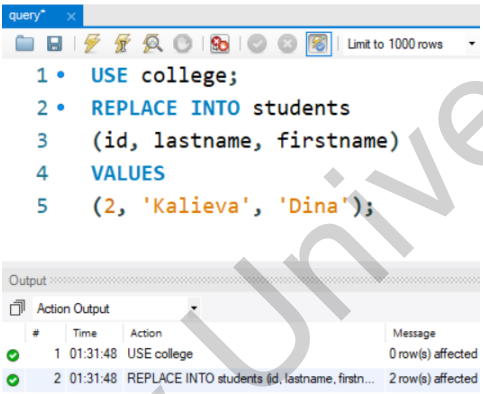
Егер *id*=2 болатын жаңа жазба қосатын болсақ, қателік анықталады. Туындаған қателік «Duplicate entry '2' for key 'students.PRIMARY'» – '2' мәнін қайта енгізу мүмкін емес (PRIMARY KEY – қайталанбайтын кілттік өріс).



109 - сурет. Жазба қосу

REPLACE операторын қолдану арқылы жаңа жазба қосылады. Егер *id*=2 жаңа жазба кестеде бар болса, оны өшіріп, жаңа жазбаны қосады.

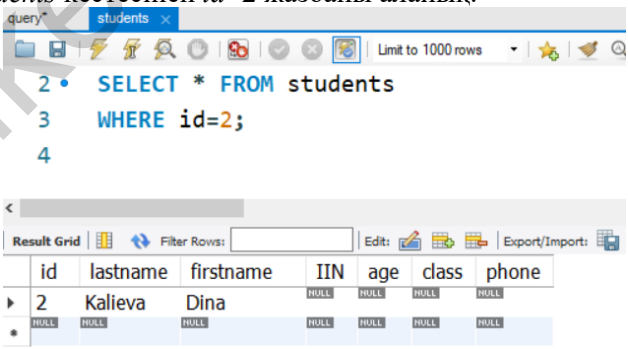
```
USE college;
REPLACE INTO students
(id, lastname, firstname)
VALUES
(2, 'Kalieva', 'Dina');
```



110 - сурет. REPLACE операторы арқылы жаңа жазба қосу

REPLACE операторы арқылы (*id, lastname, firstname*) өрістерін ғана қостық. Ал қалған өрістер NULL мәнімен толтырылады (егер өріс қасиеті NOT NULL болмаса).

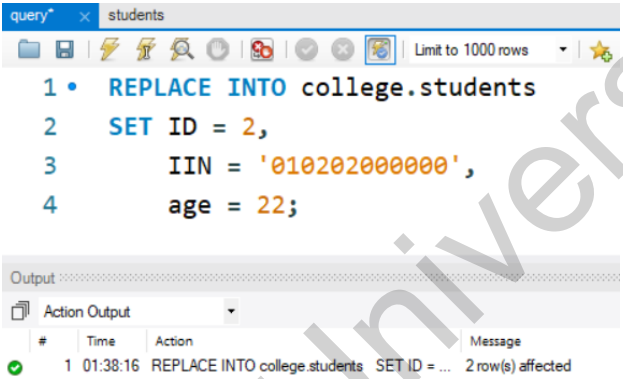
students кестесінен *id*=2 жазбаны алайық.



111 - сурет. *id*=2 жазбаны алу

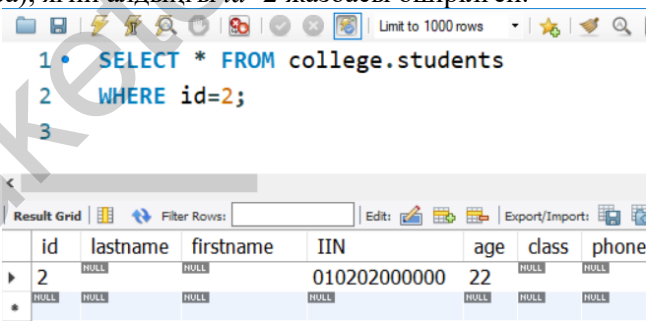
REPLACE операторы арқылы *id*=2 жазбанын енгізейік. *IIN* мен *age* өрістеріне мән береміз.

```
REPLACE INTO college.students
SET id = 2,
   IIN = '010202000000',
   age = 22;
```



112 - сурет. REPLACE операторын қолдану. Жаңа мәндерді орнату

Егер бұл жазба алсақ *IIN* мен *age* өрістерінен басқа өрістер NULL мәнімен толтырылады (егер өріс қасиеті NOT NULL болмаса), яғни алдыңғы *id*=2 жазбасы өшірілген.

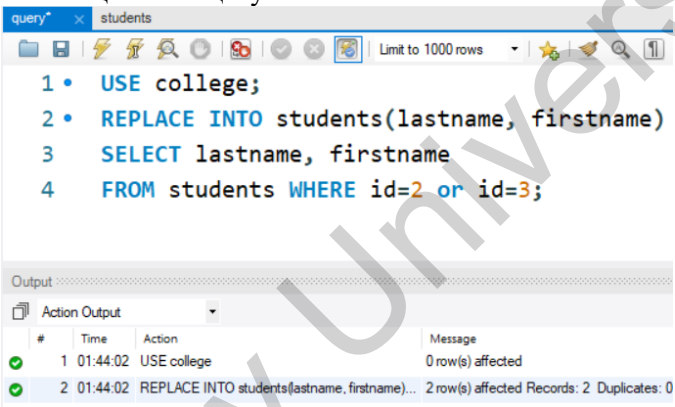


113 - сурет. *students* кестесінен *id*=2 жазбаны алу

Басқа кестеден сұраныспен алынған деректерді екінші кестеге енгізу үшін келесі REPLACE INTO нұсқаулығын пайдалана аламыз. Жалпы синтаксисі:

```
REPLACE INTO кесте1(өрістер)
SELECT кесте2.өрістер
FROM кесте2
WHERE шарт;
```

id=2 мен *id*=3 болатын студенттердің аты мен тегін қолданып жаңа жазба қосу.



114 - сурет. REPLACE INTO арқылы жазба қосу

```
USE college;
REPLACE INTO students(lastname, firstname)
SELECT lastname, firstname
FROM students WHERE id=2 or id=3;
```

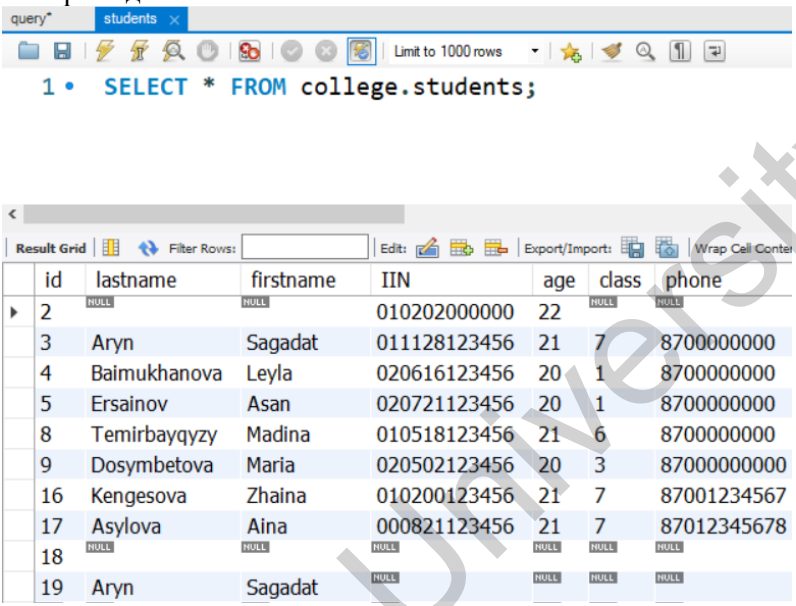
students кестесінен *id*=2 мен *id*=3 студенттердің *lastname* мен *firstname* мәндерін кестеге қайта қосамыз.

Нәтижесінде, *id*=2 жазбасының *lastname* мен *firstname* мәндерін жоқ (NULL) болғандықтан *id*=18 жазбасының барлық өрістері NULL мәнімен толтырылады.

Егер өрістер NOT NULL атрибутына ие болса, жоғарыда келтірілген мысалда қателіктер туындайды).

id=3 жазбасының *lastname=Aryn* мен *firstname=Sagadat* мәндері көшіріліп *id*=19 жазба толтырылады. *lastname* мен

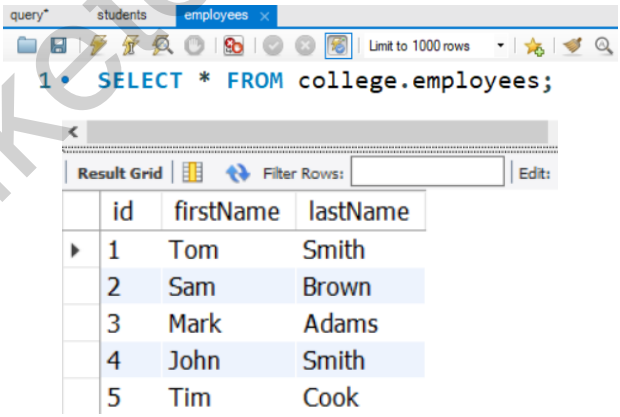
firstname өрістерінен басқа өрістер NULL мәнімен толтырылады.



id	lastname	firstname	IIN	age	class	phone
2	NULL	NULL	010202000000	22	NULL	NULL
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
9	Dosymbetova	Maria	020502123456	20	3	87000000000
16	Kengesova	Zhaina	010200123456	21	7	87001234567
17	Asylova	Aina	000821123456	21	7	87012345678
18	NULL	NULL	NULL	NULL	NULL	NULL
19	Aryn	Sagadat	NULL	NULL	NULL	NULL

115 - сурет. *id*=18 бен *id*=19 жазбалар қосылды

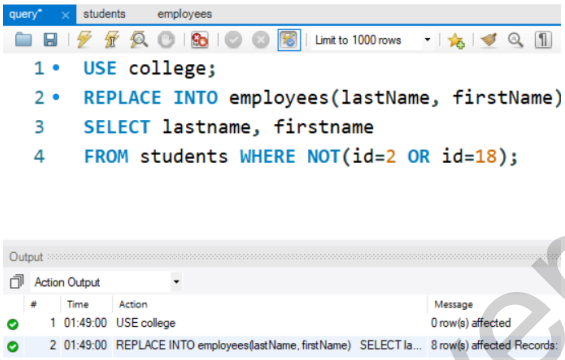
Бір кестеден басқа кестеге көшіру
employees кестесі



id	firstName	lastName
1	Tom	Smith
2	Sam	Brown
3	Mark	Adams
4	John	Smith
5	Tim	Cook

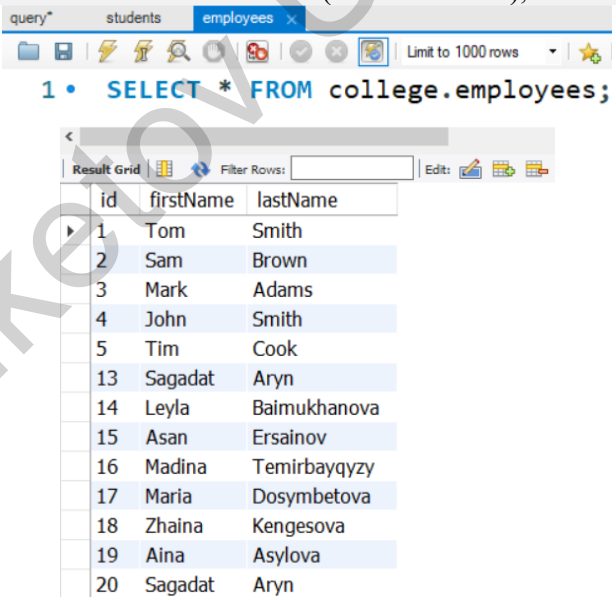
116 - сурет. *employees* кестесі

employees кестесіне *students* кестесінен *lastname, firstname* өрістерінің мәндерін қосамыз.



117 - сурет. Бір кестеден басқа кестеге көшіру

```
USE college;
REPLACE INTO employees(lastName, firstName)
SELECT lastname, firstname
FROM students WHERE NOT(id=2 OR id=18);
```



id	firstName	lastName
1	Tom	Smith
2	Sam	Brown
3	Mark	Adams
4	John	Smith
5	Tim	Cook
13	Sagadat	Aryn
14	Leyla	Baimukhanova
15	Asan	Ersainov
16	Madina	Temirbayqyzy
17	Maria	Dosymbetova
18	Zhaina	Kengesova
19	Aina	Asylova
20	Sagadat	Aryn

118 - сурет. Нәтижесі

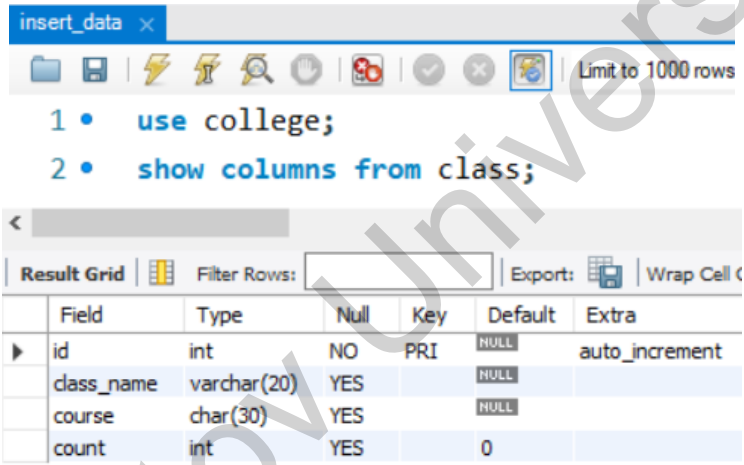
3.3 Деректерді таңдау. SELECT командасы

MySQL-де деректер қорында деректерді таңдау үшін SELECT командасы қолданылады. Ең қарапайым синтаксисі:

```
SELECT өрістер_тізімі FROM кесте_атауы;
```

class кестесіндегі өрістер тізімін шығарайық:

```
SHOW COLUMNS FROM class;
```

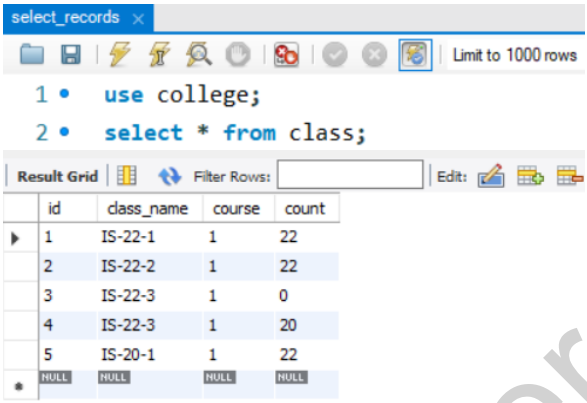


119 - сурет. Кестенің бағандарын көру

class кестесіндегі барлық өрістерді және жазбаларды (*) шығару:

```
USE college;
SELECT * FROM class;
```

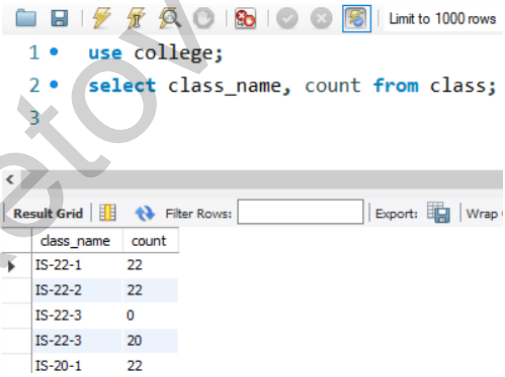
SELECT кілттік сөзінен кейін өрістер атаулары көрсетілуі керек, барлық өрістерді таңдау үшін жұлдызша (*) белгісі қойылады. Өрістер алынатын кесте атауын FROM кіліттік сөзінен соң көрсетіледі.



120 - сурет. *class* кестесіндегі барлық өрістерді (*) алу

Жұлдызша (*) символы барлық өрістерді алуымыз керек екенін көрсетеді. Тәжірибеде сұраныстар құруда * қолдану кеңес берілмейді, SELECT сөзінен кейін керек өрістер атаулары көрсетілгені дұрыс.

SELECT сөзінен кейін өрістер атаулары (,) үтір арқылы тізбектеліп жазылады:

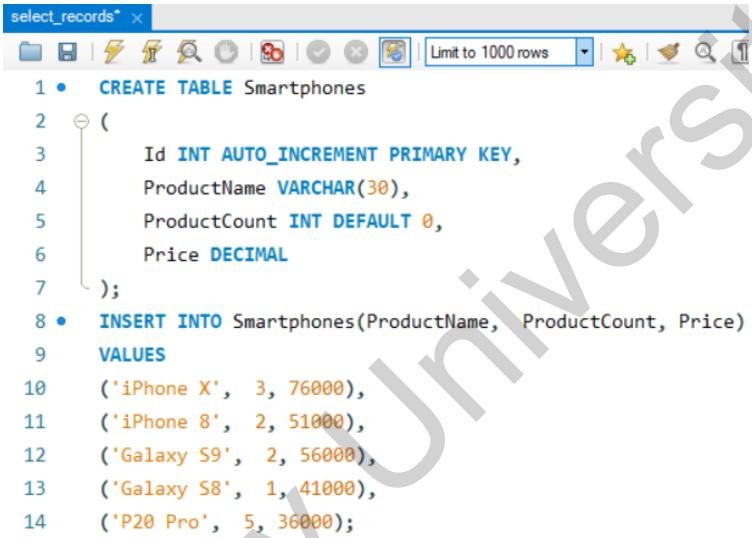


121 - сурет. *class* кестесіндегі *class_name, count* бағандарын алу

```
USE college;
SELECT class_name, count FROM class;
```

SELECT сөзінен кейін өрістер ғана емес, арнайы өрнектер көрсетуге болады. Мысалы, арифметикалық өрнектер (екі өрістердің қосындысы және т.б.).

smartphones кестесін құрып, деректерді қосайық:

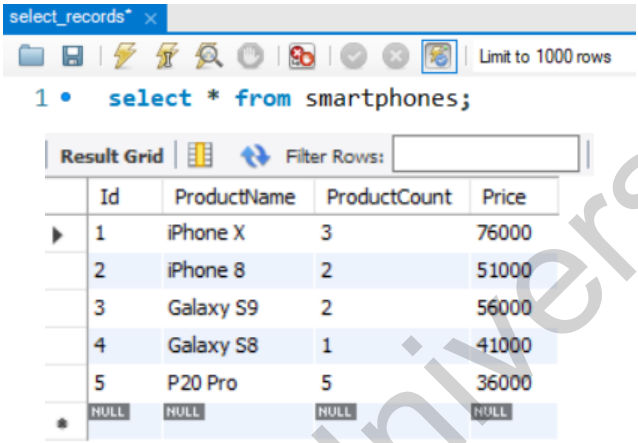


122 - сурет. *smartphones* кестесін құру және жазбалар қосу

```
CREATE TABLE smartphones
(
  Id INT AUTO_INCREMENT PRIMARY KEY,
  ProductName VARCHAR(30),
  ProductCount INT DEFAULT 0,
  Price DECIMAL
);
INSERT INTO smartphones (ProductName, ProductCount, Price)
VALUES
('iPhone X', 3, 76000),
('iPhone 8', 2, 51000),
('Galaxy S9', 2, 56000),
('Galaxy S8', 1, 41000),
```

```
('P20 Pro',  5, 36000);
```

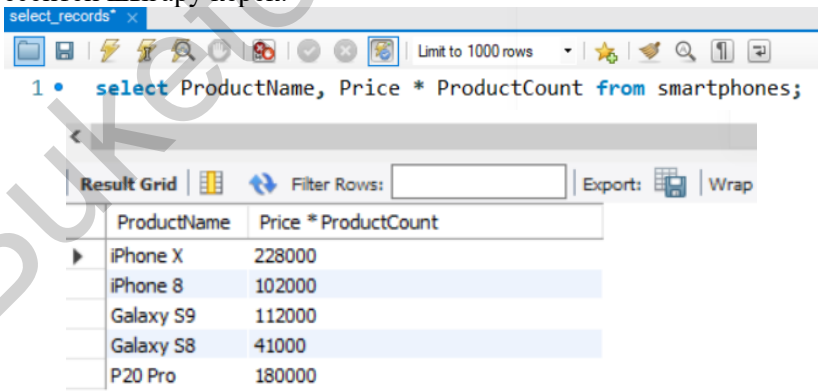
smartphones кестесіндегі барлық жазбаларды экранға шығарамыз:



123 - сурет. *smartphones* кестесіндегі барлық жазбалар

```
SELECT * FROM smartphones;
```

smartphones кестесінде смартфондар тізімі, олардың саны мен бағасы көрсетілген. Смартфондардың жалпы бағасын есептеп шығару керек:



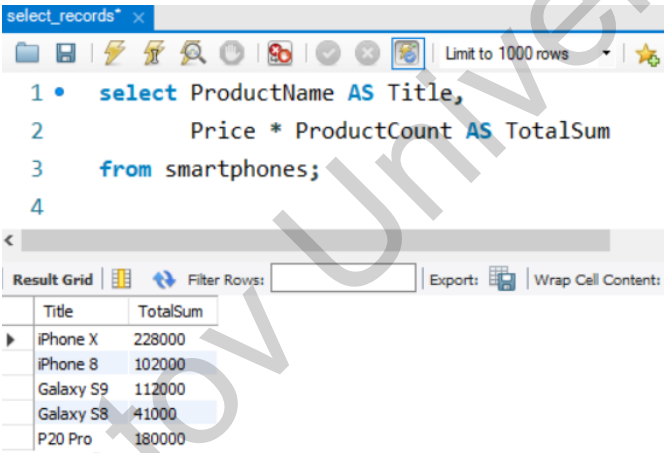
124 - сурет. Смартфондар тізімі мен олардың есептелген жалпы бағасын алу

```
SELECT ProductName, Price * ProductCount FROM smartphones;
```

Нәтижесінде екі өрісі шығарылады: *ProductName* және *Price * ProductCount*. *Price * ProductCount* бағаны *Price* және *ProductCount* өрістерінің көбейтіндісін көрсетеді.

AS операторының көмегімен экранға шығарылатын бағандарға атау (бүркеншік атау) беруге болады.

```
SELECT ProductName AS Title, Price * ProductCount AS TotalSum
FROM smartphones;
```



125 - сурет. Өрістерге AS кілттік сөзі арқылы атау беру

ProductName өрісіне *Title* атауы, *ProductCount* және *Price* өрістерінің көбейтіндісі көрсетілген бағанға *TotalSum* бүркеншік атауы беріледі.

3.5 Деректерді сүзгілеу. WHERE операторы

Көбінесе деректер базасынан барлық деректерді емес, тек белгілі бір шартқа сәйкес келетін деректерді алу қажет. Белгілі

бір шартқа сәйкес деректерді алу үшін WHERE операторы арқылы шарттар қойылады.

```
SELECT өрістер_тізімі FROM кесте_атауы
WHERE шарт
```

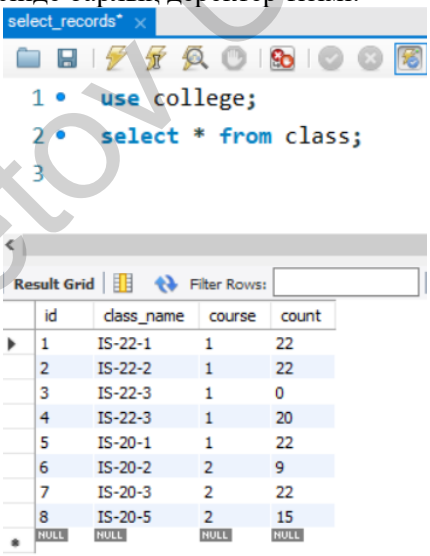
Шарттар қоюда салыстыру және логикалық операторлар қолданылады.

8-кесте

Салыстыру операторлары

=	теңдік
!=	теңсіздік
<>	теңсіздік
<	кіші
>	үлкен
<=	кіші немесе тең
>=	үлкен немесе тең

class кестесінде барлық деректер тізімі:

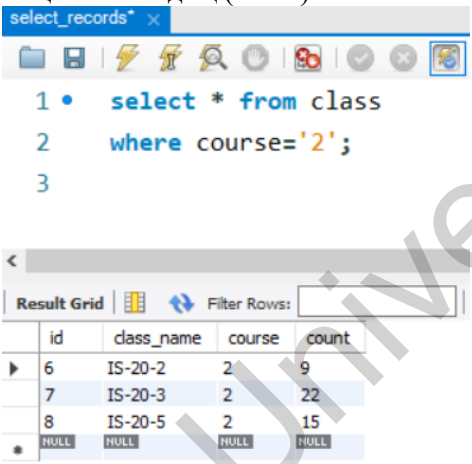


126 - сурет. *class* кестесіндегі деректер

class кестесінен 2 курс топтарын экранға шығарайық:

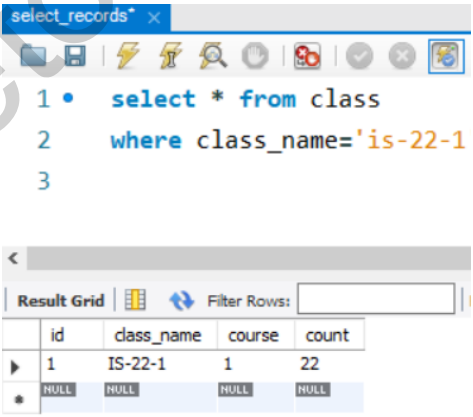
```
SELECT * FROM class
WHERE course='2';
```

course өрісінің мәні '2' апостроф белгісіне алынып жазылды, себебі бұл өрістің типі жолдық (текст) тип болып табылады.



127 - сурет. 2 курс топтарының тізімі

class кестесінен «IS-22-1» тобын ғана экранға шығарайық:



128 сурет. IS-22-1 тобы туралы ақпарат алу

Айта кету керек, MySQL үшін таңбалар регистрі маңызды емес, мысалы, «IS-22-1» жолы «is-22-1» жолына тең болады.

Логикалық операторлар

Логикалық операторлар бірнеше шарттарды біріктіруге мүмкіндік береді. MySQL ортасында келесідей логикалық операторлар бар:

AND: логикалық «ЖӘНЕ» операторы. Екі немесе одан да көп шарттарды біріктіеді:

```
шарт1 AND шарт2
```

9-кесте

AND операторының нәтижесі

шарт1	шарт2	шарт1 AND шарт2
true	*true*	*true*
true	*false*	*false*
false	*true*	*false*
false	*false*	*false*

Шарттардың барлығы орындалса, яғни шарттар мәндері *true* болған жағдайда ғана мәні *true* мәніне ие, ал қалған жағдайларда *false* мәніне ие болады. Мысалы:

(5==5) AND (6<8) = true

(9>12) AND (6<5) = false

(6!=10) AND (9<15) AND (9>11)= false

OR: логикалық НЕМЕСЕ операторы. Екі немесе одан да көп шарттарды біріктіеді:

```
шарт1 OR шарт2
```

10-кесте

OR операторының нәтижесі

шарт1	шарт2	шарт1 OR шарт2
true	*true*	*true*
true	*false*	*true*
false	*true*	*true*
false	*false*	*false*

Шарттардың барлығы орындалмаған жағдайда ғана, яғни шарттар мәндері *false* болса *OR* операторының мәні *false* мәніне ие, ал қалған жағдайларда *true* мәніне ие болады.

Мысалы:

(5==5) OR (6<8) = true

(9>12) OR (6<5) = false

(6!=10) AND (9<15) OR (9<5) = true

NOT: логикалық жоққа шығару операторы. Шарт мәніне кері мәнді қайтарады.

NOT шарт (шарттар)

11-кесте

NOT операторының нәтижесі

шарт	NOT шарт
true	*false*
false	*true*

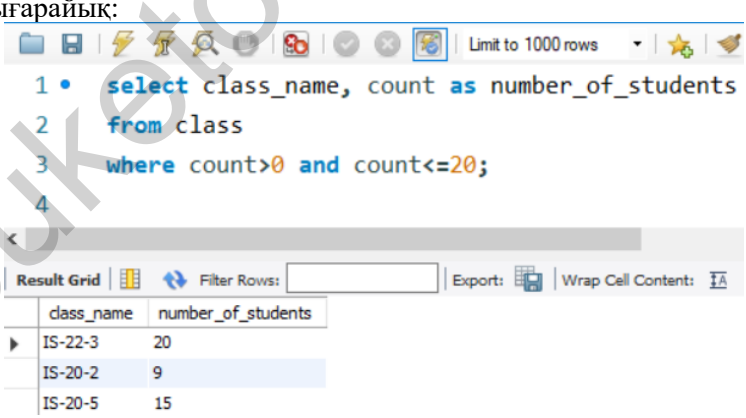
Мысалы:

NOT (6>7) = true

NOT (3<5 AND 4<6) = false

NOT (5>3 AND 9>10 OR 9<12) = false

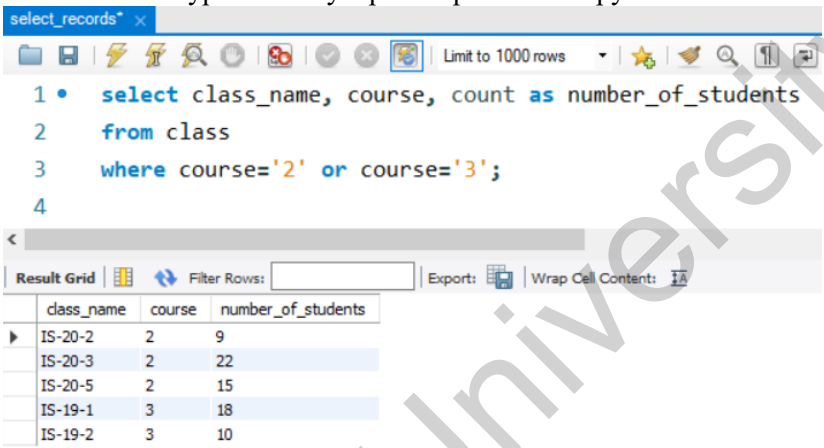
Студенттер саны 0-20 аралығындағы топтарды экранға шығарайық:



129 - сурет. Студенттер саны 0-20 аралығындағы топтарды алу

```
SELECT class_name, count as number_of_students
FROM class
WHERE count>0 AND count<=20;
```

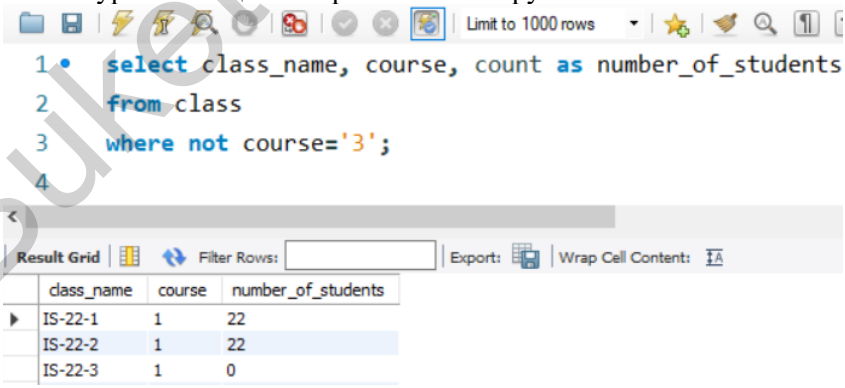
2 және 3 курс топ атауларын экранға шығару:



130 - сурет. 2 және 3 курс топтар тізімін алу

```
SELECT class_name, course, count as number_of_students
FROM class
WHERE course='2' OR course='3';
```

3 курстан басқа топтар тізімін шығару:



131 - сурет. 3 курстан басқа топтар тізімі

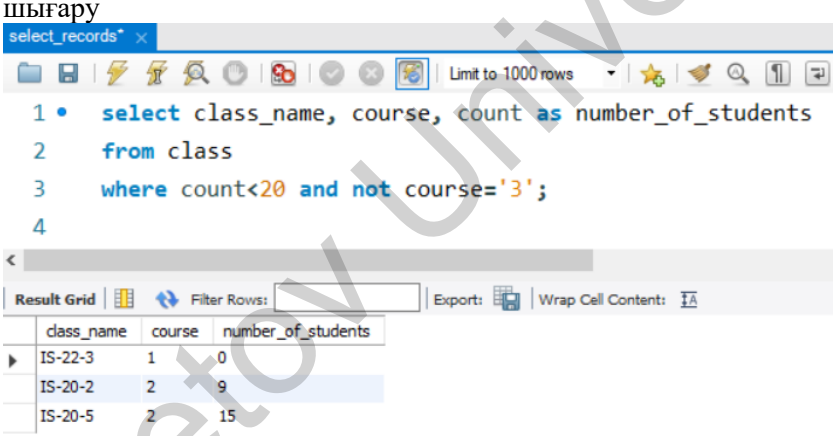
```
SELECT class_name, course, count as number_of_students
FROM class
WHERE NOT course='3';
```

Логикалық операторлардың орындалу реті (приоритеті)

Логикалық операторлар арқылы екі немесе оданда көп шарттарды біріктіруге болады. Логикалық операторлар келесі реттілікпен орындалады:

1) NOT,
2) AND,
3) OR.

Студент саны 20-дан аз және 3 курс емес топтар тізімін шығару



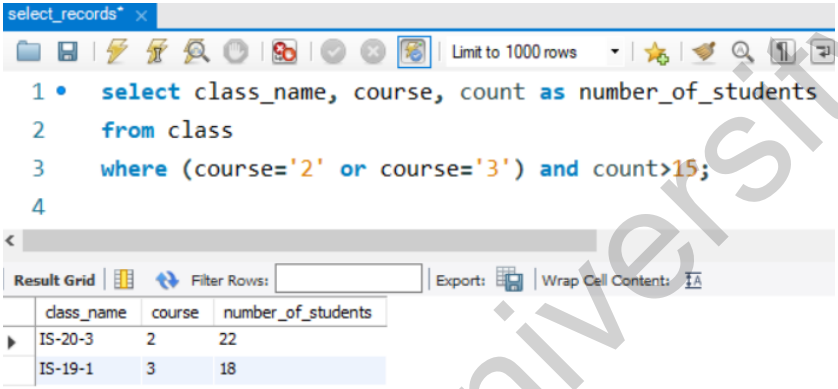
132 - сурет. Студент саны 20-дан аз және 3 курс емес топтар тізімін алу

```
SELECT class_name, course, count as number_of_students
FROM class
WHERE count<20 and not course='3';
```

Қойылған шартта NOT операторы бірінші орындалады, яғни *course*='3' тең болмауы керек. NOT операторынан кейін AND операторы орындалады.

Жақшалар көмегімен операторлардың орындалу ретін өзгертуге болады.

2 және 3 курс, студенттер саны 15-тен көп топтар тізімін шығару:



133 - сурет. 2 және 3 курс, студенттер саны 15-тен көп топтар тізімін алу

```
SELECT class_name, course, count as number_of_students
FROM class
WHERE (course='2' OR course='3') AND count>15;
```

Қойылған шартта жақшаға алынған ораператоры орындалады. Жақшадан кейін AND операторы орындалады.

3.4 Деректерді өзгерту. UPDATE командасы

UPDATE командасы кестеде бар жазбаларды өзгерту үшін қолданылады.

Жалпы синтаксисі:

```
UPDATE кесте_атауы
SET өріс1 = жаңа_мән1,
өріс2 = жаңа_мән2,
...
өрісN = жаңа_мәнN
[WHERE шарт]
```

UPDATE кілттік сөзінен кейін кесте атауы көрсетіледі, SET арқылы өрістерге жаңа мәндер беріледі. WHERE арқылы өзгертілетін жазбаларды таңдап алуға болады.

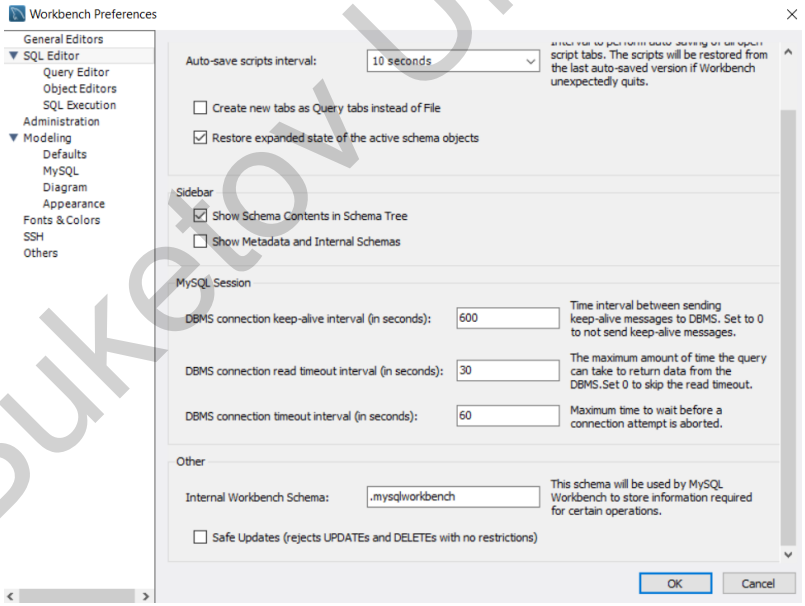
class кестесіндегі әр топтағы студенттер санын 1 санға арттырайық:

```
UPDATE class
SET count = count+1;
```

Жазылған скрипті іске қосқанда, келесідей қателік аласыз:

You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column. To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect.

Қауіпсіз режимінде болғандықтан осындай қателік алынады. Қауіпсіз режимін өшіру үшін MySQL Workbench негізгі мәзірінде Edit -> Preferences терезесіндегі SQL Editor пунктін таңдаңыз:



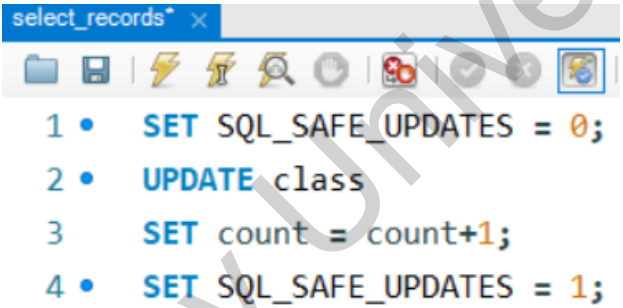
134 - сурет. Workbench Preferences терезесі

Ашылған терезеде «Safe Updates (reject UPDATEs and DELETEs with no restrictions)» белгішені алып тастап сақтаңыз (OK), серверге қайта қосылыңыз.

Бұл қателік болдырмаудың екінші жолы SET SQL_SAFE_UPDATES = 1; жолын скрипте енгізу:

```
SET SQL_SAFE_UPDATES = 0;
UPDATE class
SET count = count+1;
SET SQL_SAFE_UPDATES = 1;
```

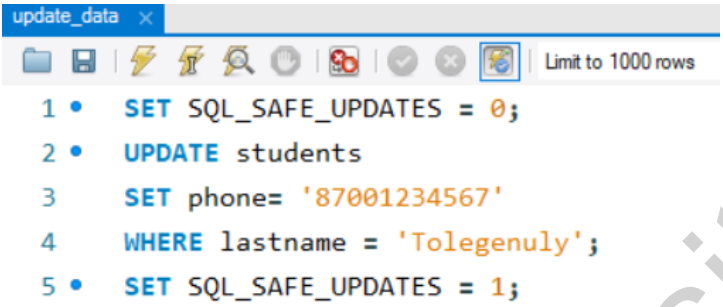
UPDATE командасы арқылы *class* кестесіндегі *count* өрісінің мәнін бір санға арттырамыз.



135 - сурет. *class* кестесіндегі әр топтағы студенттер санын 1 санға арттыру

Скрипт сәтті орындалады. Әр топтағы студенттер саны 1 санына артады.

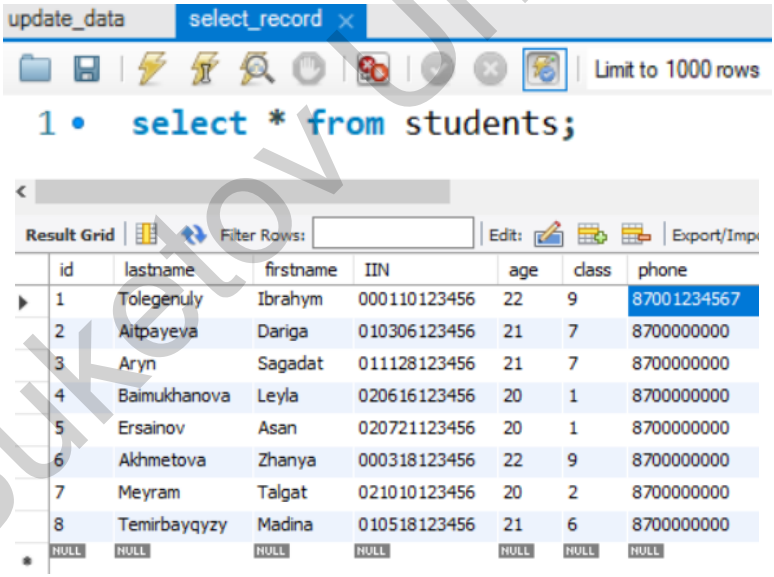
students кестесінде тегі *Tolegenuly* болатын студенттің телефон нөмірін (*phone*) өзгертейік. Мұнда тегі *Tolegenuly* болатын студенттің жазбасын таңдап алу үшін WHERE арқылы шарт қойылады.



136 - сурет. Жазбаны өшіру

```
UPDATE students
SET phone= '87001234567'
WHERE lastname = 'Tolegenuly';
```

Нәтижесі:



id	lastname	firstname	IIN	age	class	phone
1	Tolegenuly	Ibrahym	000110123456	22	9	87001234567
2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
6	Akhmetova	Zhanya	000318123456	22	9	8700000000
7	Meyram	Talgat	021010123456	20	2	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

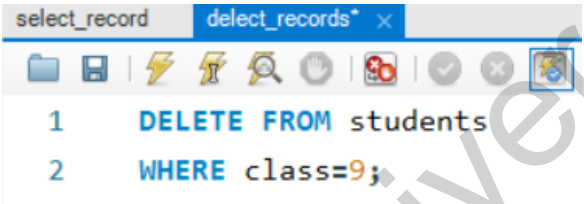
137 - сурет. Students кестесі

3.5 Деректерді жою. DELETE операторы

DELETE операторы деректер қорынан жазбаларды өшіреді.
Жалпы синтаксисі:

```
DELETE FROM кесте_атауы
[WHERE шарт]
```

students кестесінен *class* номері *id* =9 болатын студенттерді өшіру.



138 - сурет. Жазбаны өшіру

```
DELETE FROM students
WHERE class=9;
```

Кестеден барлық жазбаларды өшіру:

```
DELETE FROM students;
```

3.6 MySQL деректер қорын экспорттау және импорттау

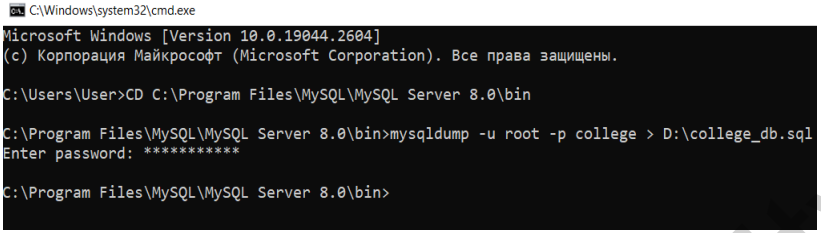
MySQL деректер қорын экспорттау және импорттау 2 әдіспен жүзеге асырылады:

1) Command Line Tool
2) MySQL Workbench

Командалық жол арқылы алдыңғы бөлімдерде құрылған college деректер қорын экспорттау.

1) Командалық жолды іске қосыңыз;
2) MySQL Server орналасқан бумаға ішіндегі bin бумасына өтіңіз.

```
CD C:\Program Files\MySQL\MySQL Server 8.0\bin
```



139 - сурет. Сервердегі деректер қорын экспорттау

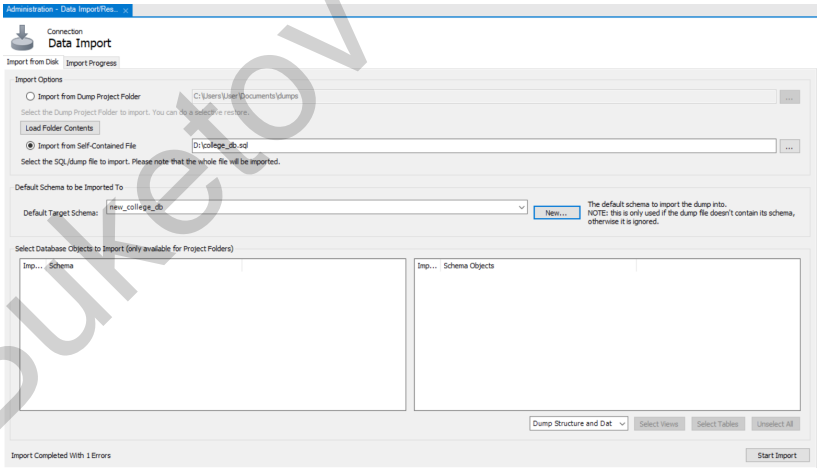
3) Деректер қорын экспорттау үшін келесі командаңы жазыңыз.

```
mysqldump -u root -p college > D:\college_db.sql
```

Экспорттау кезінде root қолданушы үшін құпия кілтті енгізуді сұрайды. Экспортталған деректер қоры D дискісінде *college_db* атаумен сақталады, файл типі – *.sql.

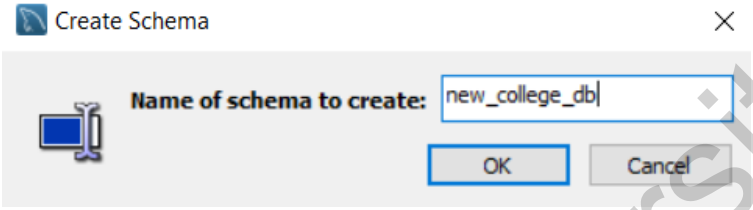
MySQL Workbench арқылы деректер қорын импорттау

Мәзірден Server→Data Import таңдап, Administration бөліміне өтіңіз.



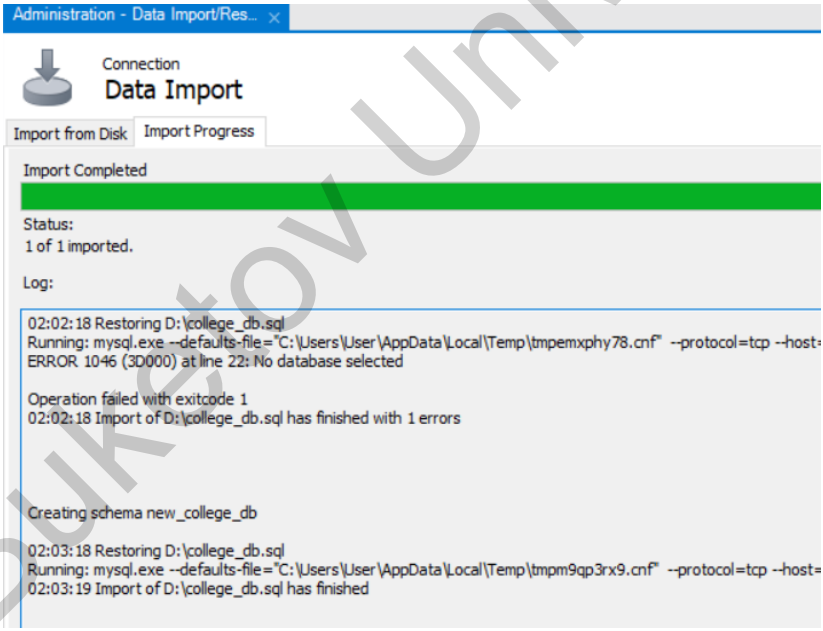
140 - сурет. Server→Data Import →Administration

Import from Disk Import from Self-Contained File таңдап, импорталатын деректер қорын таңдаймыз. New батырмасын басып импорталатын деректер қорын атау береміз.



141 - сурет. Деректер қорына атау беру

Импорттау бастау үшін Start import батырмасын басамыз. Импорттау процесі Import progress терезесінен көре аламыз.



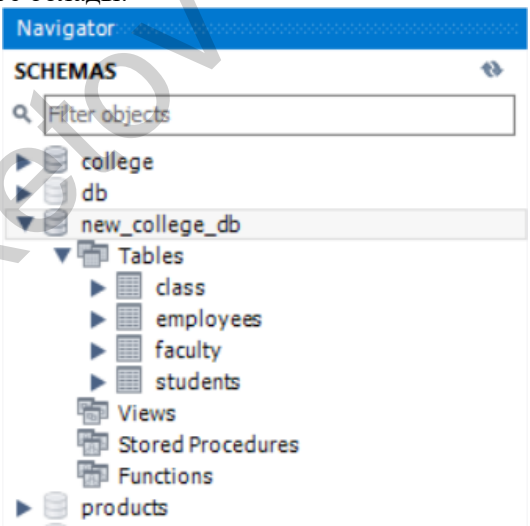
142 - сурет. Импорттау нәтижесі

```
02:02:18 Restoring D:\college_db.sql
Running: mysql.exe --defaults-
file="C:\Users\User\AppData\Local\Temp\tmpemxphy78.cnf"  --
protocol=tcp --host=127.0.0.1 --user=root --port=3306 --default-
character-set=utf8 --comments  < "D:\\college_db.sql"
ERROR 1046 (3D000) at line 22: No database selected

Operation failed with exitcode 1
02:02:18 Import of D:\college_db.sql has finished with 1 errors

Creating schema new_college_db
02:03:18 Restoring D:\college_db.sql
Running: mysql.exe --defaults-
file="C:\Users\User\AppData\Local\Temp\tmpm9qp3rx9.cnf"  --
protocol=tcp --host=127.0.0.1 --user=root --port=3306 --default-
character-set=utf8 --comments --database=new_college_db  <
"D:\\college_db.sql"
02:03:19 Import of D:\college_db.sql has finished
```

Импортталған деректер қорын Навигатор терезесінен тексеруіңізге болады.



143 - сурет. Навигатор терезесі

CSV типтегі файлдардағы жазбаларды импорттау

CSV файлының құрылымында үтірмен бөлінген деректер тізімі бар. greenhouse.csv файлын импорттаймыз.

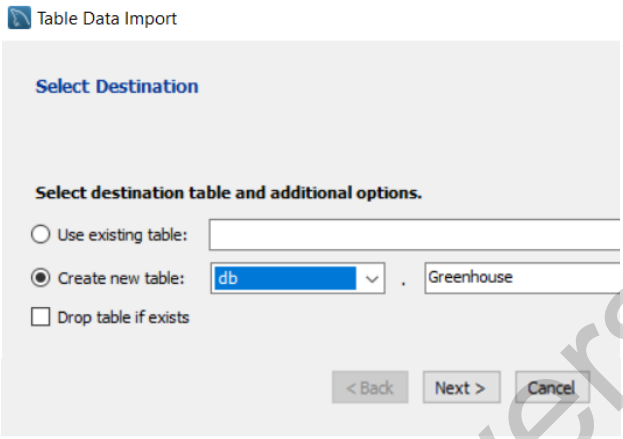
	A	B	C	D	E	F	G	H
1	Anzsic,Anzsic_description,Period,Data_value,Variable,Units,Gas,Status							
2	ZPZ,Primary industries,2014Q1,11316,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
3	ZPZ,Primary industries,2014Q2,11178,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
4	ZPZ,Primary industries,2014Q3,11126,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
5	ZPZ,Primary industries,2014Q4,11118,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
6	ZPZ,Primary industries,2015Q1,11064,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
7	ZPZ,Primary industries,2015Q2,11088,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
8	ZPZ,Primary industries,2015Q3,11061,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
9	ZPZ,Primary industries,2015Q4,11033,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
10	ZPZ,Primary industries,2016Q1,10941,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
11	ZPZ,Primary industries,2016Q2,10874,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
12	ZPZ,Primary industries,2016Q3,10887,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							
13	ZPZ,Primary industries,2016Q4,10887,Seasonally adjusted,Kilotonnes,Carbon dioxide eq							

144 - сурет. greenhouse.csv файлынан үзінді

Импорталатын *greenhouse.csv* файлын таңдаймыз.

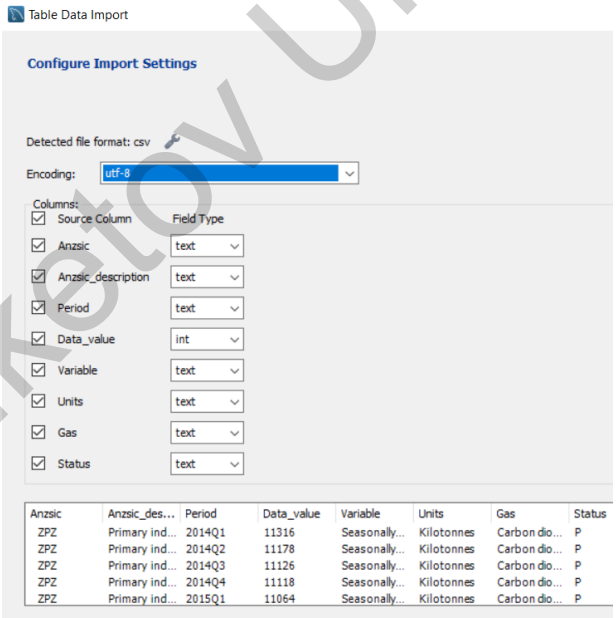
145 - сурет. Импортталатын файлды таңдау

Create new table белгілеп, деректер қорын таңдаймыз. db деректер қоры таңдалынды. «Келесі» батырмасын басыңыз.



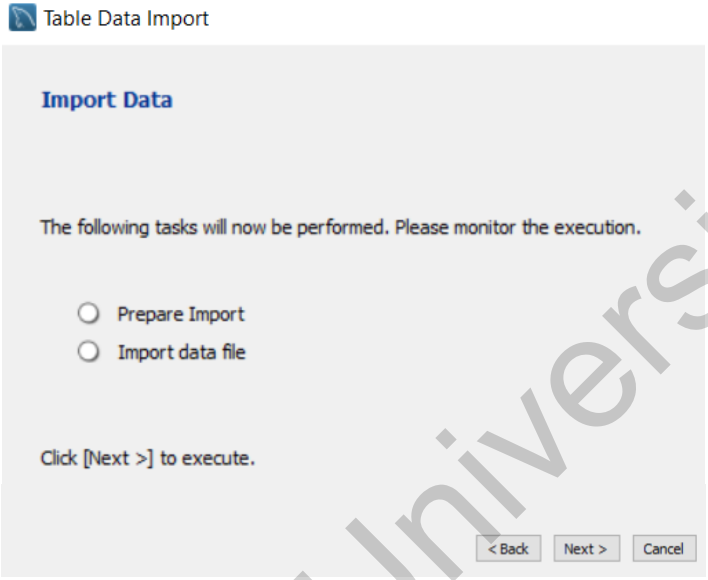
146 - сурет. Жаңа деректер қорын және кесте құру

Кодтау стандартын (UTF-8) және өрістер мен олардың дерек типін таңдаймыз. «Келесі» батырмасын басыңыз.

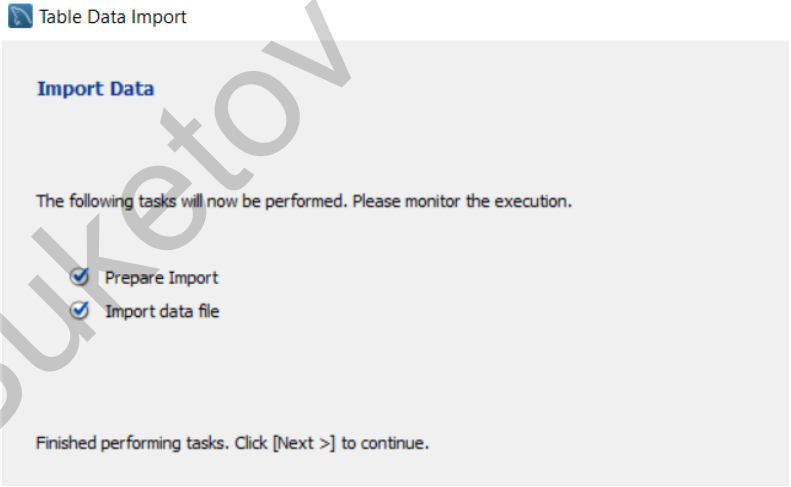


147 - сурет. Өрістер мен олардың дерек типін таңдау

Импорттауды бастау үшін «Келесі» батырмасын басыңыз.



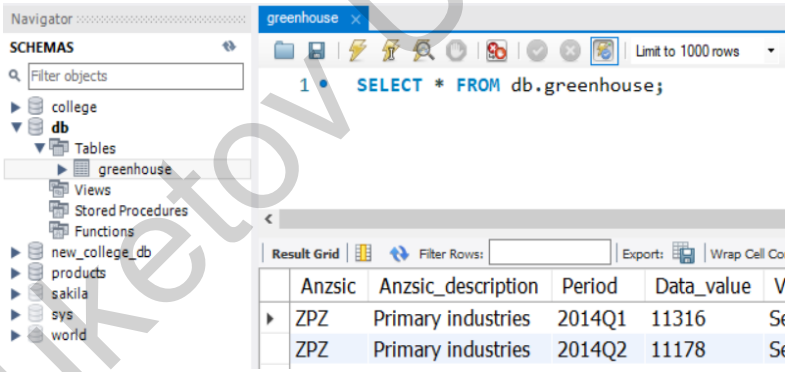
148 - сурет. Импорттауды бастау



149 - сурет. Импорттау процесі

Импорттау аяқталған кейін, импорттаудың сәтті аяқталған және импортталған жазба саны шығарылады.

150 - сурет. Импорттау аяқталды



151 - сурет. *greenhouse* кестесі

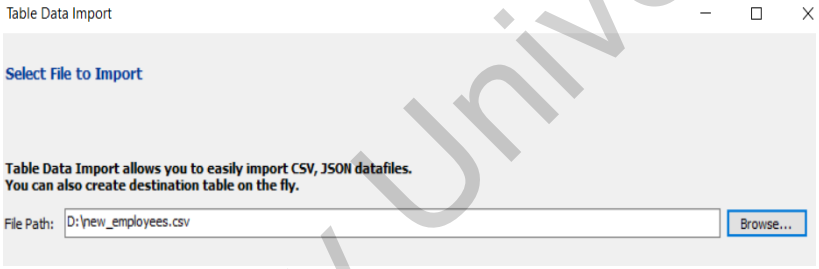
CSV файлының деректерді тізімін деректер қорындағы бар кестеге қосайық.

new_employees.csv файлындағы деректерді college деректер қорындағы employees кестесі қосамыз.

	A	B
1	firstName	lastName
2	Berton, Tom	
3	Quanysh, Sydyqov	
4	Makhmet, Ospan	

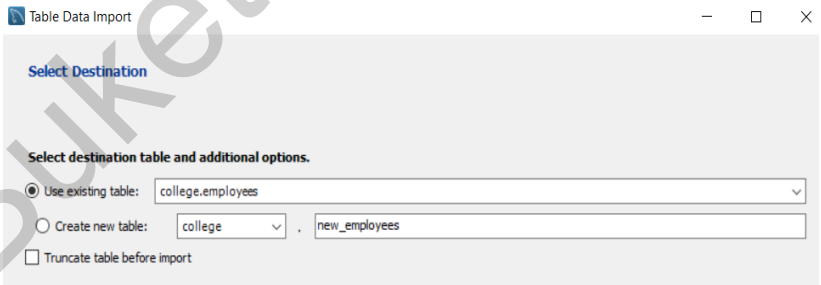
152 - сурет. *new_employees.csv* файлындағы деректер

new_employees.csv файлын таңдаймыз.



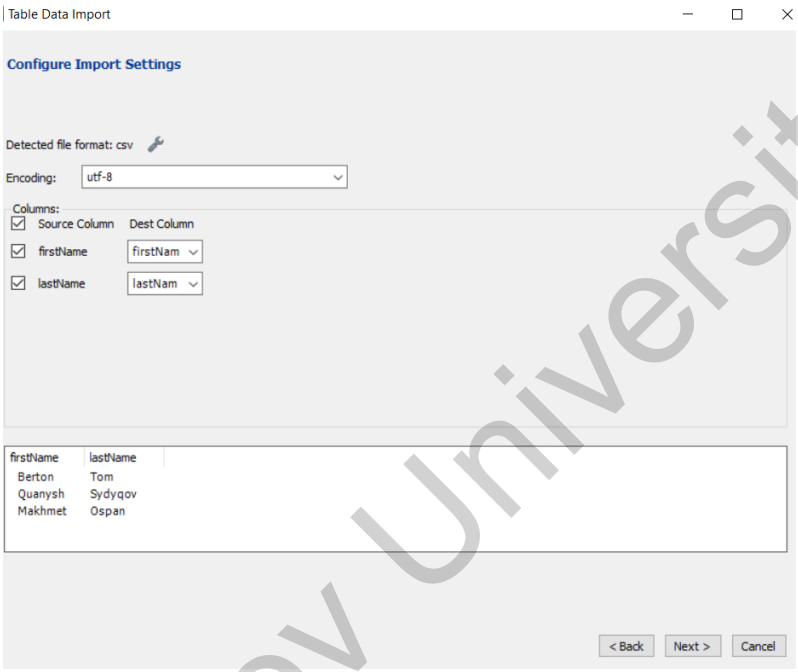
153 - сурет. Импортталатын файлды таңдау

Use existing table таңдап, деректер қорындағы кестені таңдаймыз.

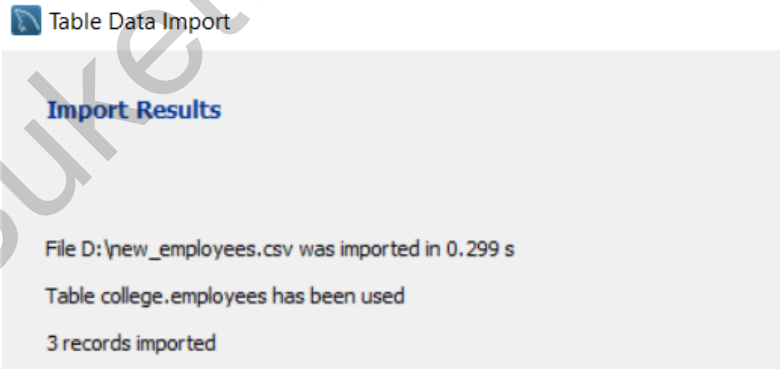


154 - сурет. Кестені таңдау

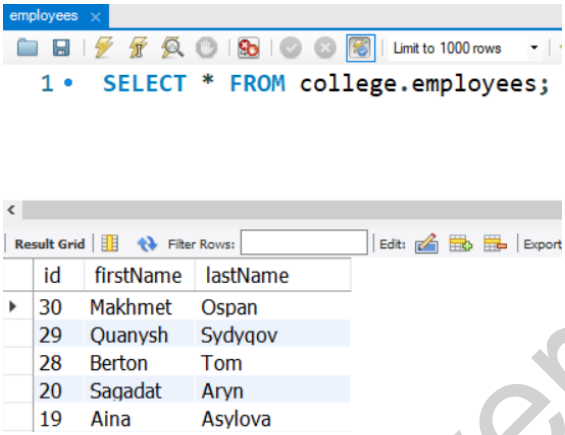
new_employees.csv файлынан импортталатын өрістерді және оларға сәйкес employees кестесінің өрістерін таңдаймыз.



155 - сурет. Импортталған кестеден өрістерді таңдау



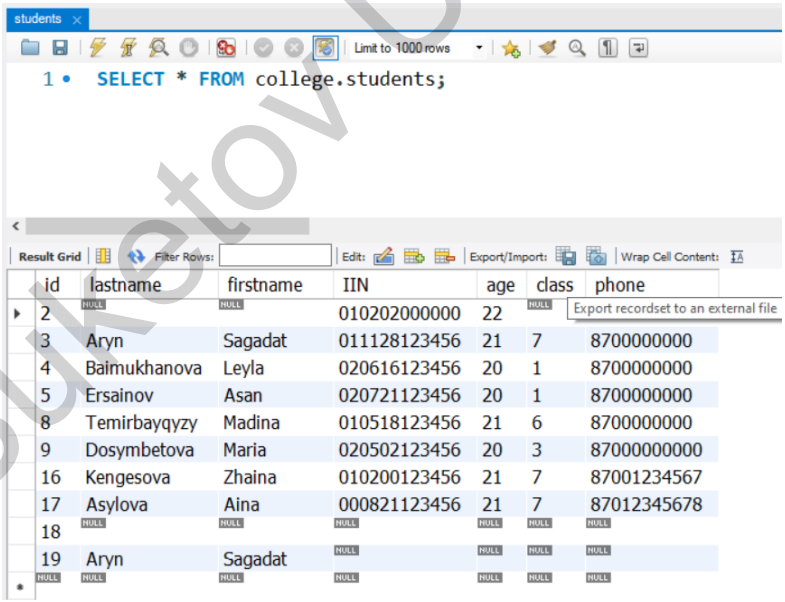
156 - сурет. Импорттау нәтижесі



157 - сурет. *employees* кестесі

Кестені CSV форматында экспорттау

Кестедегі жазбаларды шығарып, Export recordset to an external file батырмасын басып, CSV форматында сақтаймыз.



158 - сурет. Кестені CSV форматында экспорттау

4 тарау. Сұраныстар

4.1 Бірегей мәндерді таңдау. DISTINCT операторы

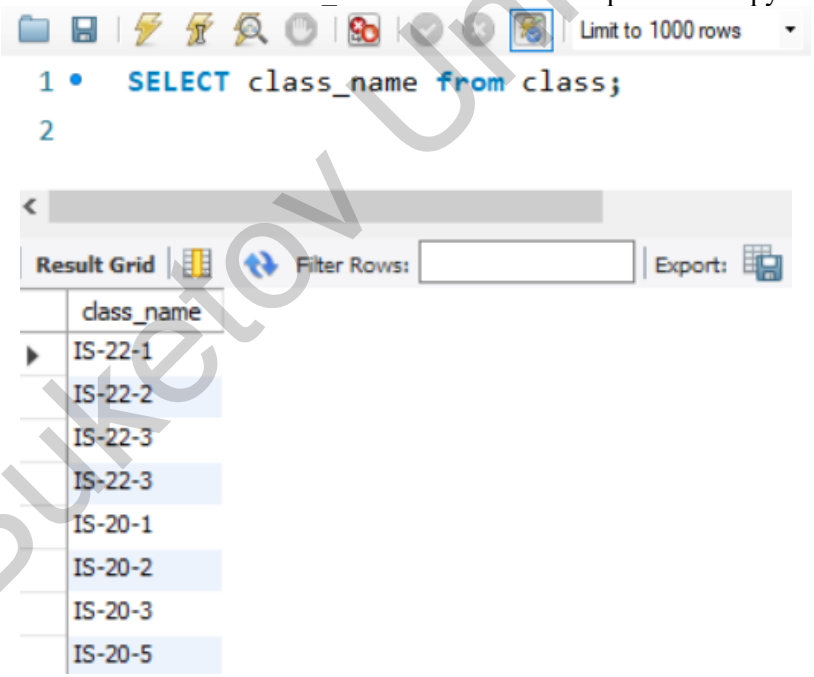
Сұраныс SELECT сөзінен басталады.
Ең қарапайым сұраныс:

```
SELECT * FROM students;
```

Бұл сұраныс студенттер кестесінен барлық бағандарды, жазбаларды экранға шығарады.

Белгілі бір бағандарды ғана таңдауға болады. Ол үшін SELECT сөзінен кейін бағандар атаулары тізбектеліп жазылады.

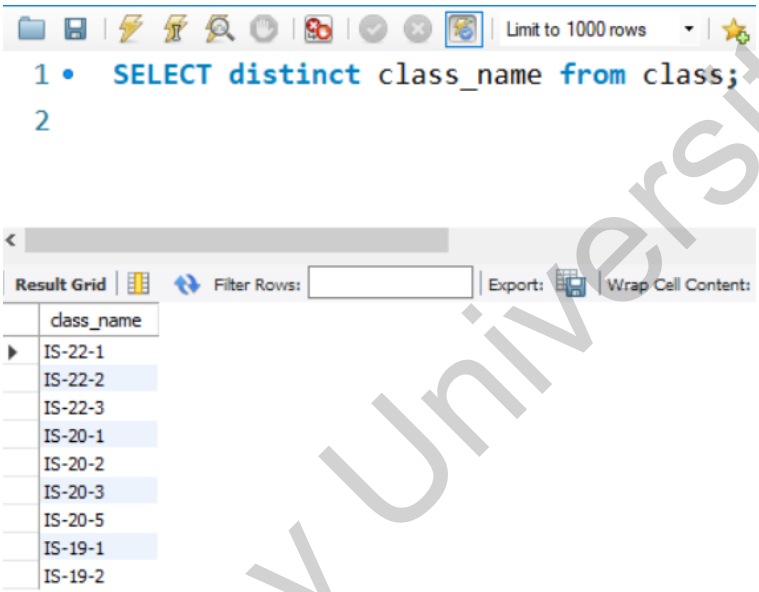
Class кестесінен class_name бағанын ғана экранға шығару:



159 - сурет. *class* кестесінен топтар атауларын алу

Мұндай сұраныстарда ұқсас жазбалар қайталануы мүмкін. Қайталанатын жазбаларды шығармау үшін DISTINCT операторы қолданылады.

```
SELECT DISTINCT class_name FROM class;
```



160 - сурет. DISTINCT операторын қолдану

4.2 Сүзгілеу операторлары

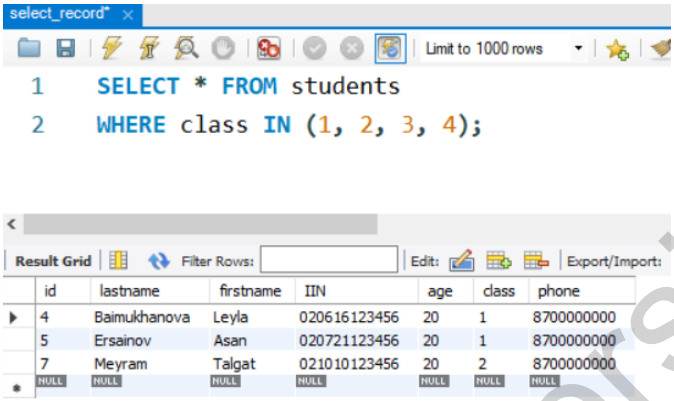
IN операторы

Өріс мәні белгілі бір жиындағы мәндерге сәйкес келетін жазбаларды алуға болады. Ол үшін IN операторы қолданылады.

Жалпы синтаксисі:

```
WHERE өріс[NOT] IN (мәндер жиыны)
```

Студенттер кестесінен *class* нөмері 1-4 болатын студенттер тізімін экранға шығару:

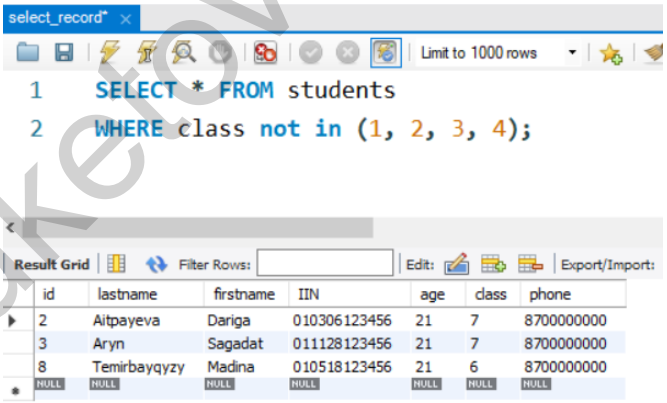


161 - сурет. IN операторын қолдану

```
SELECT * FROM students
WHERE class IN (1, 2, 3, 4);
```

NOT операторы арқылы көрсетілген тізімде жоқ жазабаларды шығаруға мүмкіндік береді:

Студенттер кестесінен *class* нөмері 1-4 болмайтын студенттер тізімін экранға шығару:



162 - сурет. NOT операторын қолдану

```
SELECT * FROM students
WHERE class NOT IN (1, 2, 3, 4);
```

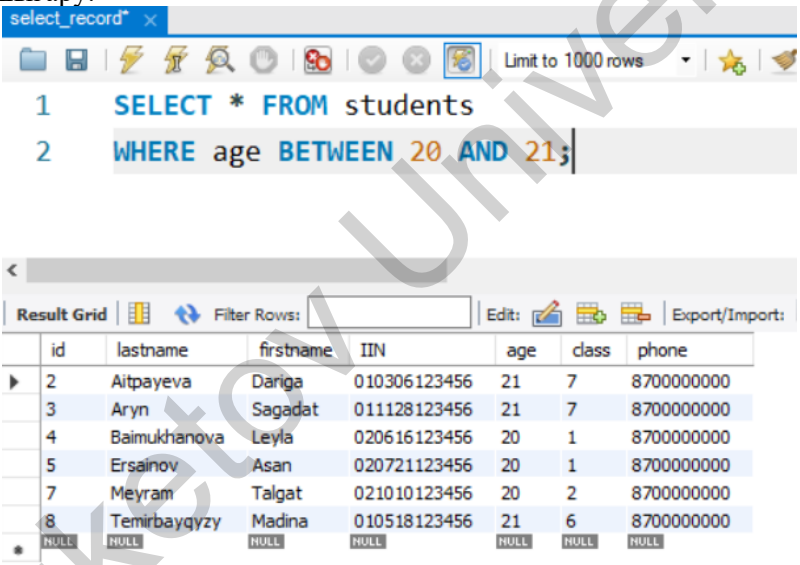
BETWEEN операторы

BETWEEN операторы арқылы белгілі бір мәндер аралығындағы (диапазонын) мәндерге сәйкес келетін жазбаларды анықтауға болады.

BETWEEN операторы бастапқы және соңғы мәнді қолдана отырып, мәндер ауқымын анықтайды:

WHERE өріс_атауы [NOT] BETWEEN бастапқы_мәні AND соңғы_мәні

Студенттер кестесінен жасы 20, 21 тең студенттер тізімін шығару.



id	lastname	firstname	IIN	age	class	phone
2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
7	Meyram	Talgat	021010123456	20	2	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

163 - сурет. BETWEEN операторы

```
SELECT * FROM students
WHERE age BETWEEN 20 AND 21;
```

Студенттер кестесінен жасы 20, 21 тең емес студенттер тізімін шығару сұранысы.

SELECT * FROM students
WHERE age NOT BETWEEN 20 AND 21

LIKE және REGEXP операторлары

LIKE операторы жол үлгісін қабылдайды және осы үлгіге сәйкес келетін өріс жазбаларын алады.

```
WHERE баған [NOT] LIKE жол_үлгісі
```

Жол үлгісін анықтау үшін арнайы символдар қолданылады:

1) %: бір немесе бірнеше символдар

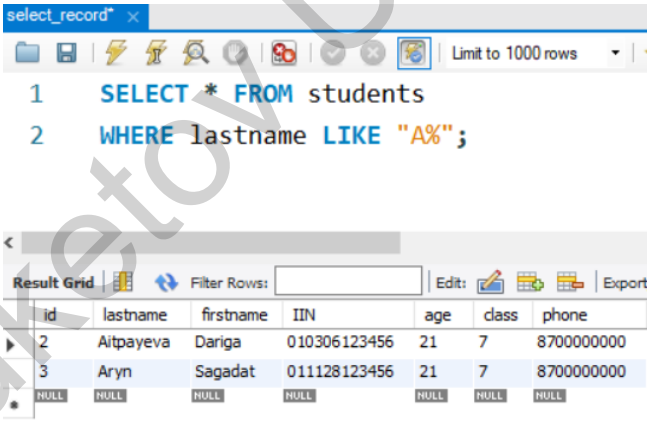
Мысалы: WHERE lastname LIKE "A%" студент тегі А әріпінен басталатын студенттер тізімін шығарады. Студент тегінде қаріптер саны маңызды емес.

2) _: бір символ

Мысалы: WHERE firstname LIKE "A_" студент аты А әріпінен кейін бір ғана символ бар екенін көрсетеді.

LIKE операторын қолданып сұраныс жазайық:

```
SELECT * FROM students
WHERE lastname LIKE "A%";
```



164 - сурет. LIKE операторы

REGEXP бағанның мәні сәйкес келетін тұрақты өрнекті орнатуға мүмкіндік береді. REGEXP операторы LIKE операторына қарағанда күрделі сүзгілеу әдісін ұсынады. REGEXP ұқсас синтаксиске ие:

WHERE өріс [NOT] REGEXP тұрақты өрнек

Тұрақты өрнектерде келесідей арнайы символдар қолданылады:

1) ^: жолдың басын көрсетеді
2) $: жолдың соңын көрсетеді
3) .: кез келген бір символға сәйкес келеді
4) [символдар]: жақшадағы символдардың біреуіне сәйкес келеді
5) [бастапқы_символ-соңғы_символ]: жақшада көрсетілген диапазондағы символдың біреуіне сәйкес келеді
6) |: екі жолды бөледі, осы екі жолдың біреуіне сәйкес келуі керек

REGEXP операторын қолдану мысалдары:

WHERE firstname REGEXP 'гүл':

firstname мәнінде "гүл" тіркесі болуы керек, мысалы, Айгүл, Гүлдария, Нұргүл

WHERE firstname REGEXP '^Гүл':

firstname мәні "Гүл" тіркесімен басталуы керек, мысалы, Гүлжан, Гүлмария

WHERE firstname REGEXP 'гүл$':

firstname мәні "гүл" тіркесімен аяқталуы керек, мысалы, Нұргүл, Айгүл, Жанаргүл

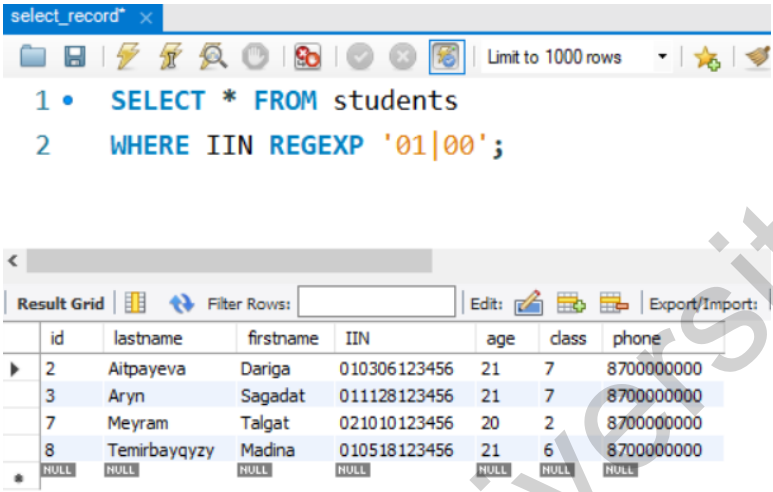
WHERE firstname REGEXP 'Гүлна [рз]':

firstname мәні 'Гүлназ' немесе 'Гүлнар' мәндеріне сәйкес келуі керек.

WHERE IIN REGEXP '01050345523 [6-8]':

IIN өрісі '010503455236', '010503455237', '010503455238' мәндеріне сәйкес келуі керек.

IIN өрісінде "00" немесе "01" бар жазбаларды шығайық:



165 - сурет. REGEXP операторын қолдану

IS NULL

IS NULL операторы арқылы өрісіндегі мәні NULL болатын жазбаларды таңдайды:

```
SELECT * FROM students
WHERE firstname IS NULL;
```

students кестесінен *firstname* өрісі бос, толтырылмаған жазбаларды таңдайды.

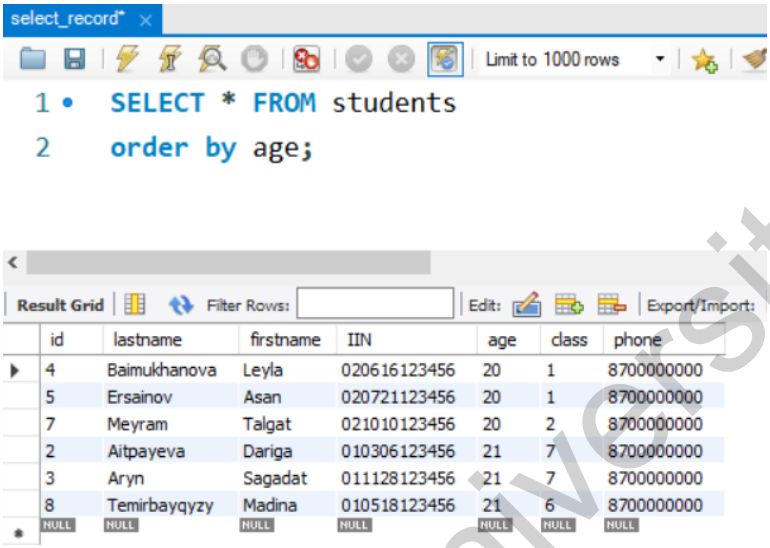
NOT операторын қосу арқылы керісінше, NULL емес жолдарды алуға болады:

```
SELECT * FROM students
WHERE firstname IS NOT NULL;
```

students кестесінен *firstname* өрісі бос, толтырылмаған жазбаларды таңдайды.

4.3 Сұрыптау. ORDER BY операторы

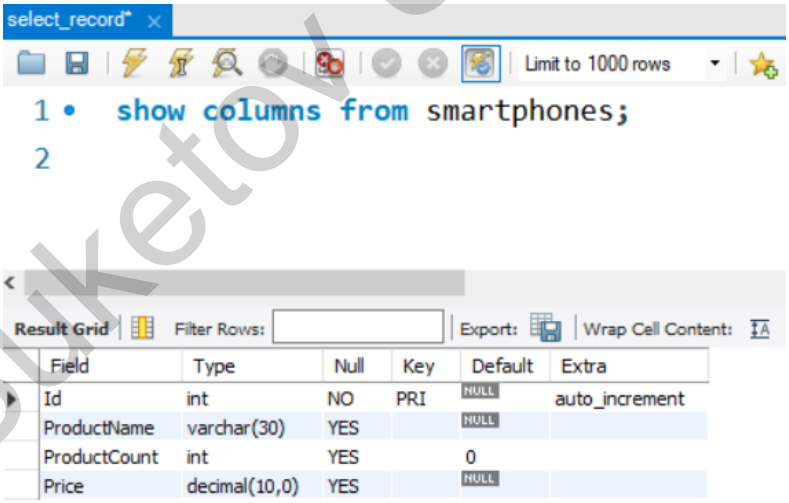
ORDER BY операторы жазбаларды бір немесе бірнеше өріс арқылы кему, өсу реті бойынша сұрыптайды. Мысалы, *students* кестесіндегі жазбаларды *age* бағаны арқылы сұрыптайық:



id	lastname	firstname	IIN	age	class	phone
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
7	Meyram	Talgat	021010123456	20	2	8700000000
2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

166 - сурет. Сұрыптау

smartphones кестесіндегі өрістерді экранға шығарамыз:



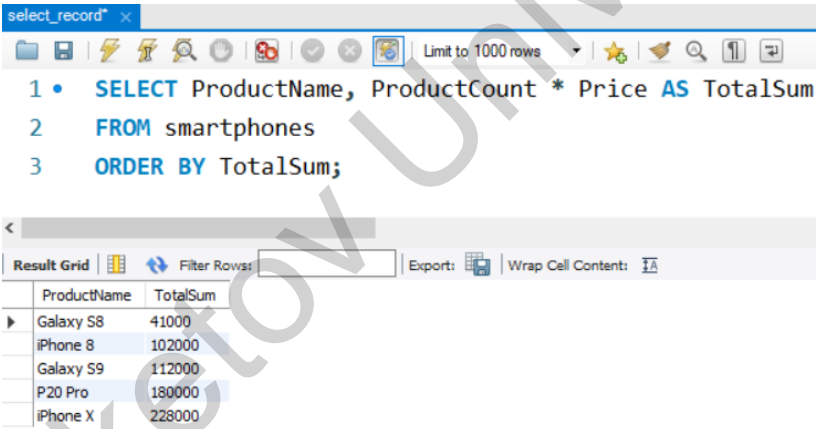
Field	Type	Null	Key	Default	Extra
Id	int	NO	PRI	NULL	auto_increment
ProductName	varchar(30)	YES		NULL	
ProductCount	int	YES		0	
Price	decimal(10,0)	YES		NULL	

167 - сурет. *smartphones* кестесінің өрістері

Нәтижесі:

Field	Type	Null	Key	Default	Extra
Id	int	NO	PRI		auto_increment
ProductName	varchar(30)	YES			
ProductCount	int	YES		0	
Price	decimal(10,0)	YES			

smartphones кестесіндегі тауар аты мен жалпы құнын есептеп (саны*бағасы) экранға шығарайық. Жалпы құны есептеу үшін *ProductCount* және *Price* өрістерінің мәнін көбейтеміз. Жалпы құны көрсетілетін бағанға AS арқылы бүркеншік атау береміз.

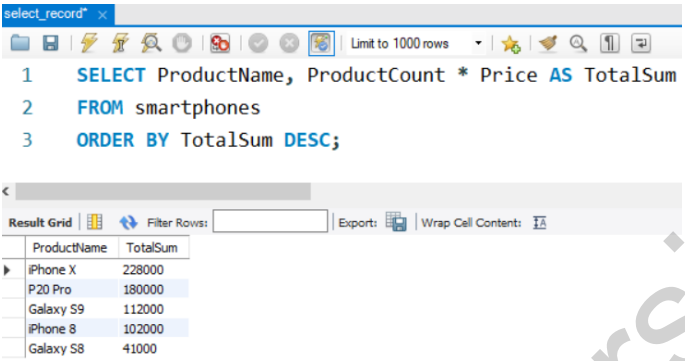


168 - сурет. *TotalSum* өрісі бойынша сұрыптау

Жазылған скрипт тауарларды жалпы құнының өсуі бойынша сұрыптап береді.

Кему реті бойынша сұрыптау

Үнсіздік келісім бойынша ORDER BY операторы деректерді өсу реті бойынша сұрыптайды. Деректерді кему реті бойынша сұрыптау үшін DESC параметрі қолданылады.



169 - сурет. Кері ретпен сұрыптау

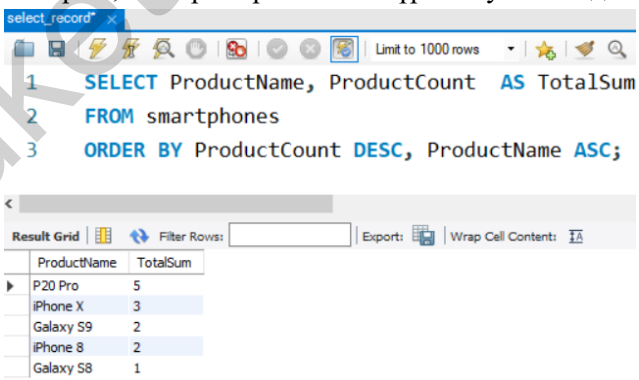
```
SELECT ProductName, ProductCount * Price AS TotalSum
FROM smartphones
ORDER BY TotalSum DESC;
```

ASC операторы көмегімен деректерді өсу ретімен сұрыптайды:

```
SELECT ProductName, ProductCount * Price AS TotalSum
FROM smartphones
ORDER BY TotalSum ASC;
```

Бірнеше өрістер бойынша сұрыптау

ORDER BY операторынан кейін бірнеше өрістер атауын көрсете отырып, сол өрістер бойыша сұрыптауға болады:



170 - сурет. Бірнеше өріс бойынша сұрыптау

```
SELECT ProductName, Price, ProductCount * Price AS TotalSum
FROM Products
ORDER BY ProductName ASC, TotalSum DESC;
```

Жазылған скрипте деректерді ең алдымен *ProductName* бағанының өсу реті бойынша сұрыптайды. Одан кейін *ProductName* өрісі бойынша бірдей жазбаларды *Price* өрісі арқылы сұрыптайды. Әр сұрыпталатын бағанға ASC немесе DESC параметрлерін қосуға болады.

4.4 Жазбалар диапазонын алу. LIMIT операторы

LIMIT сөзі шектеу дегенді білдіреді. MySQL-де LIMIT операторы арқылы алынатын жазбалар санын көрсетуге болады.

LIMIT операторының жалпы синтаксисі:

```
LIMIT [offset,] rowcount
```

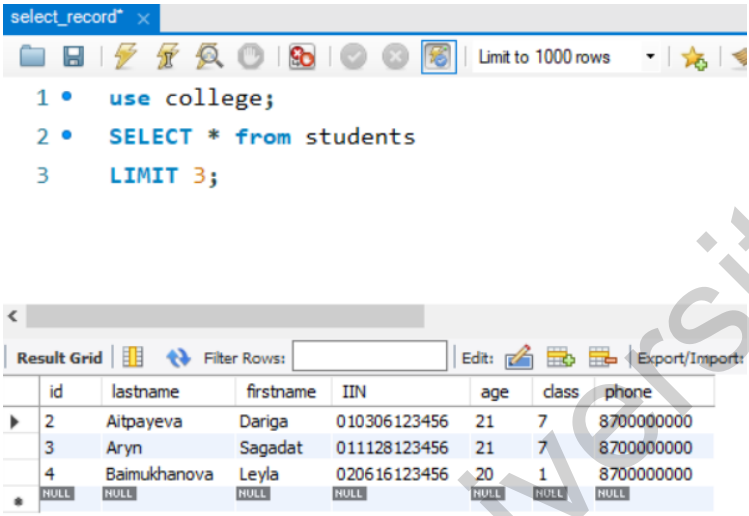
мұндағы, offset-ең алғашқы жазбаның нөмері, осы нөмерден басталады.

Rowcount-жазбалардың саны.

LIMIT операторында тек ғана бір параметрді *rowcount*, яғни жолдар санын көрсетуге болады бұл жағдайда алғашқы жазбадан басталады. Егер екі параметр көрсетілсе, бірінше параметр жолдың реттік нөмірін, ал екінші параметр жолдар санын көрсетеді.

Мысалы, *students* кестесіндегі алғашқы 3 жолды экранға шығару

```
USE college;
SELECT * FROM students
LIMIT 3;
```

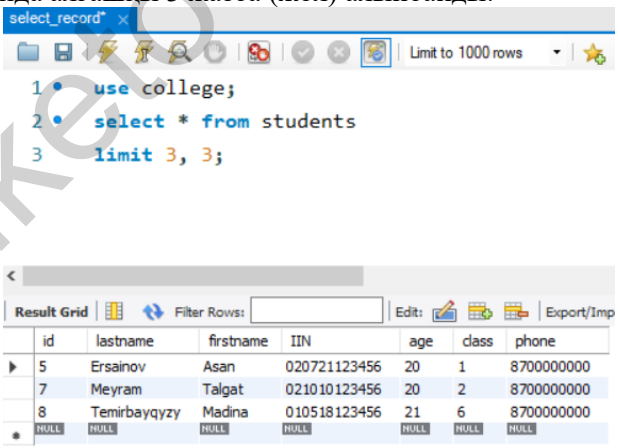


171 - сурет. LIMIT операторы

Мысалы, *students* кестесіндегі реттік нөмірі 3-ші жолдан бастап 3 жолды экранға шығару

```
SELECT * FROM students
LIMIT 3, 3;
```

Мұнда алғашқы 3 жазба (жол) алынбайды.



172 - сурет. LIMIT операторы

LIMIT операторы негізінен ORBER BY операторымен бірге қолданылады:

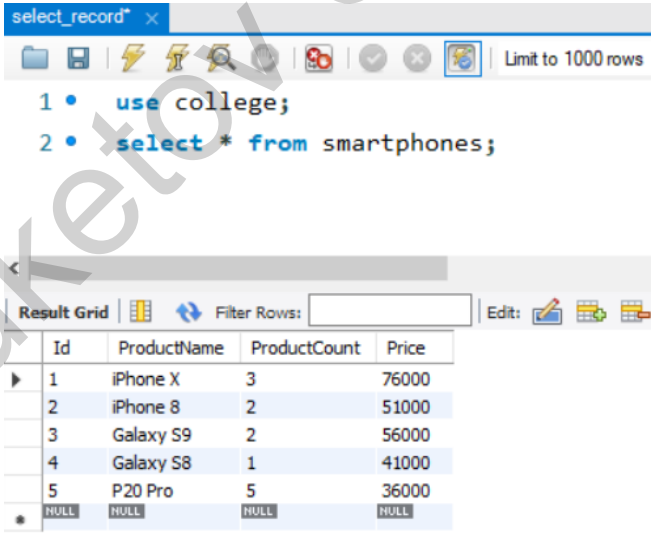
```
SELECT * FROM students
ORDER BY lastname
LIMIT 3, 3;
```

4.5 Агрегатты функциялар

Агрегатты функция скаляр мәндерді есептеу үшін қолданылады. Мысалы белгілі бір өрістегі ең үлкен мәнді не ең кіші мәнді алуға болады.

MySQL ортасында келесідей агрегатты функциялар бар:

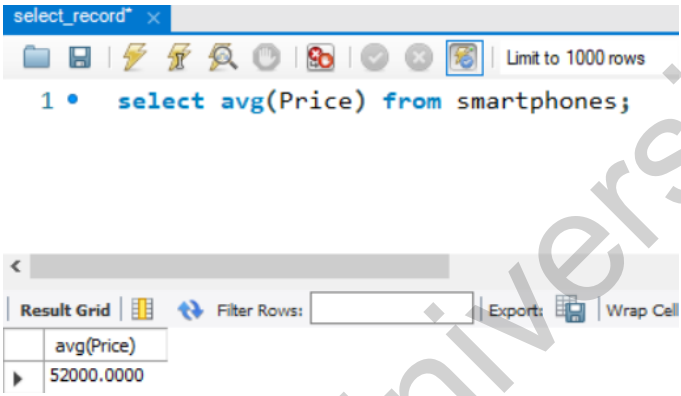
- AVG –мәндердің орташа мәнін есептейді;
- SUM– мәндердің қосындысын есептейді;
- MIN– ең кіші мәнді есептейді;
- MAX– ең үлкен мәндіесептейді;
- COUNT– сұраныстағы жазбалар санын есептейді.



173 - сурет. *smartphones* кестесі

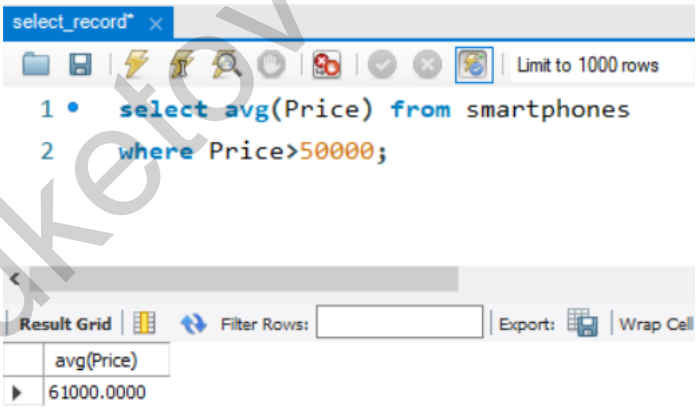
smartphones кестесіндегі тауарлардың орташа бағасын есептейік. Ол үшін AVG() агрегаттық атты функциясы қолданылады.

```
SELECT AVG(Price) FROM smartphones;
```



174 - сурет. AVG() функциясы

smartphones кестесіндегі бағасы 50000-нан жоғары тауарлардың орташа бағасын есептейік:



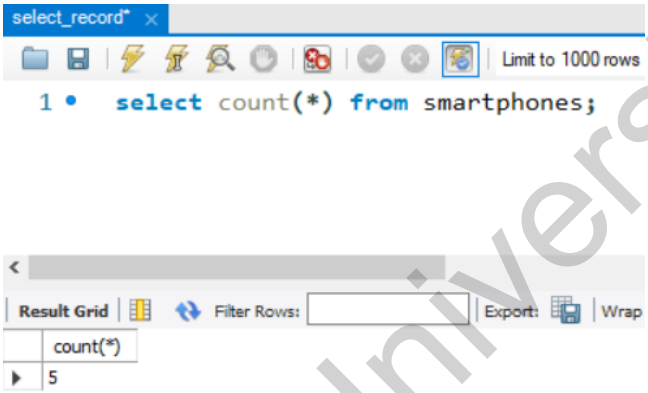
175 - сурет. Бағасы 50000-нан жоғары тауарлардың орташа бағасы

```
SELECT AVG(Price) FROM smartphones
```

```
WHERE Price>50000;
```

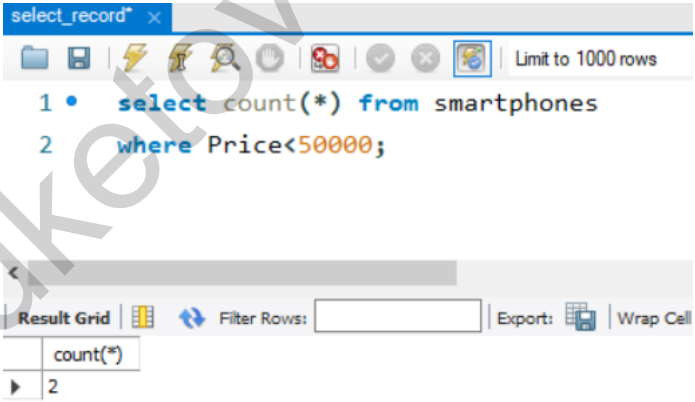
Smartphones кестесінде жазбалар саны есептейік. Ол үшін count функциясын қолданамыз.

```
SELECT COUNT(*) FROM smartphones;
```



176 - сурет. COUNT() операторы

Бағасы 50000-нан төмен тауарлар санын анықтайық. *Price* өрісіне *Price*<50000 шарты қойылады.

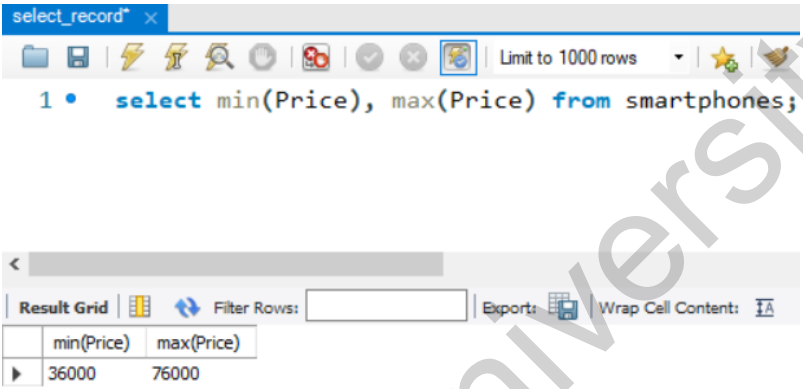


177 - сурет. Бағасы 50000 төмен тауарлар саны.

```
SELECT count(*) FROM smartphones
WHERE Price<50000;
```

smartphones кестесінде тауарлардың бағаларының ең кіші және ең үлкен мәнін анықтайық, ол үшін сәйкесінше MIN() және MAX() функциялары қолданылады.

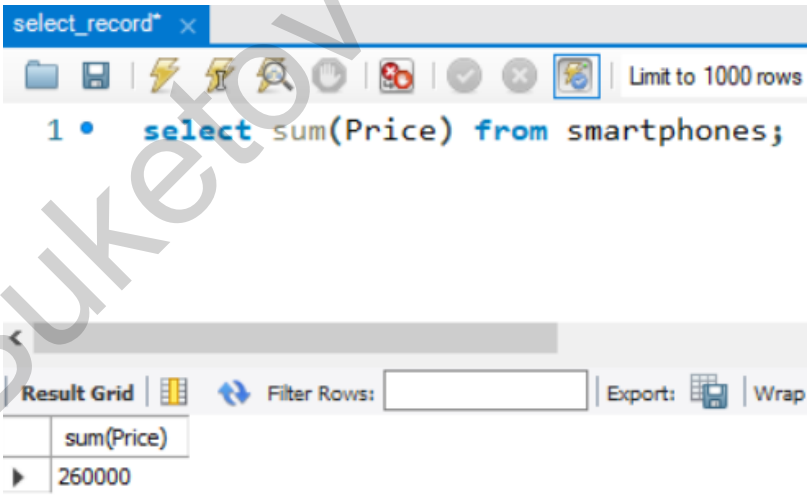
```
SELECT MIN(Price), MAX(Price) FROM smartphones;
```



178 - сурет. MIN() және MAX() операторлары

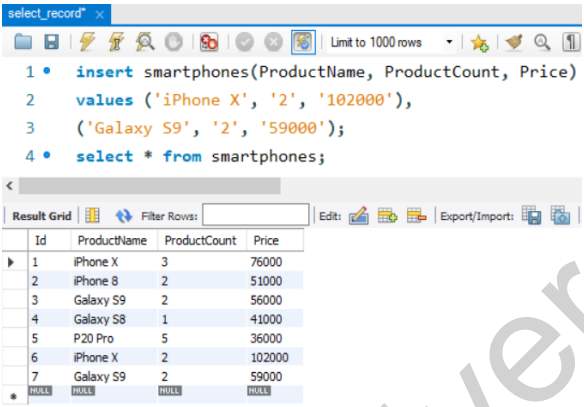
smartphones кестесінде тауарлардың бағаларының қосындысын есептейік:

```
SELECT SUM(Price) FROM smartphones;
```



179 - сурет. SUM() операторы

smartphones кестесіне деректерді қосу



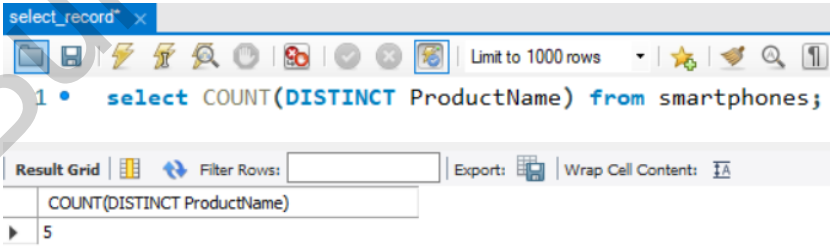
180 - сурет. *smartphones* кестесіне деректерді қосу

ALL және DISTINCT операторлары

Жоғарыда қолданылған агрегаттық функциялар орындалғанда барлық жазбаларды есептейді. Кестелерде қайталанатын жазбалар да болуы мүмкін. Қайталанбайтын жазбаларды ғана есептеу үшін DISTINCT операторы қолданылады.

smartphones кестесінен қайталанбайтын тауарларды санын есептеу.

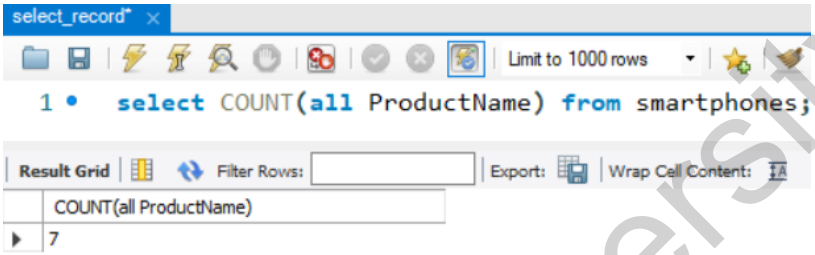
```
SELECT COUNT(DISTINCT ProductName) FROM smartphones;
```



181 - сурет. DISTINCT операторын қолдану

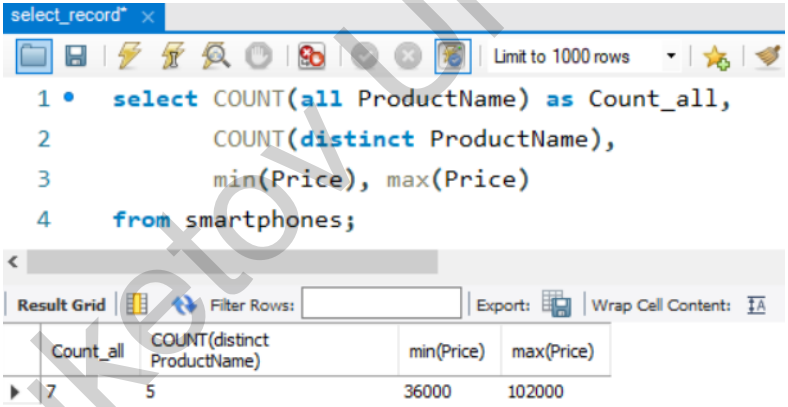
Үнсіздік күйде DISTINCT операторының орнына ALL операторы қолданылады. Бұл оператор барлық жазбаларды есептейді:

```
SELECT COUNT(ALL ProductName) FROM smartphones;
```



182 - сурет. Барлық тауарлар санын анықтау

Бір сұраныста бірнеше функцияларды бірге қолданып көрейік.



183 - сурет. Бірнеше функцияларды бір сұраныста қолдану

```
SELECT COUNT(ALL ProductName) as Count_all,
       COUNT(DISTINCT ProductName),
       min(Price), max(Price)
FROM smartphones;
```

Жазылған скриптің нәтижесінде келесідей нәтижені аламыз:

COUNT(ALL ProductName) – барлық жазбалар саны

COUNT(DISTINCT ProductName) – қайталанбайтын жазбалар саны

MIN(Price) – ең төмен баға (барлық жазбаларды ескереді)

MAX(Price) – ең жоғарғы баға (барлық жазбаларды ескереді).

4.6 Топтау. GROUP BY және HAVING операторы

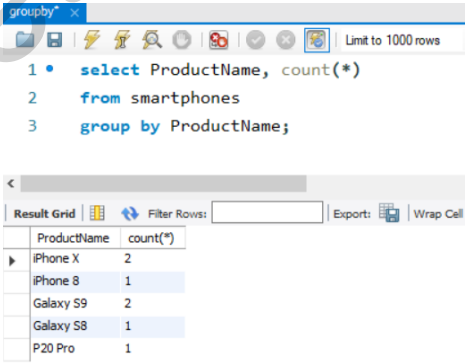
GROUP BY және HAVING операторы арқылы жазбаларды біріктіреді (топтайды). Бұл операторлар сұраныстарда, SELECT операторымен бірге қолданылады.

Жалпы синтаксисі:

```
SELECT бағандар
FROM кесте
[WHERE шарттар]
[GROUP BY баған]
[HAVING топтау_үшін_қойылатын_шарт]
[ORDER BY баған]
```

GROUP BY операторы

GROUP BY операторы жолдардың қай өріс бойынша топталатынын көрсетеді. Мысалы, тауар аты бойынша біріктірейік:

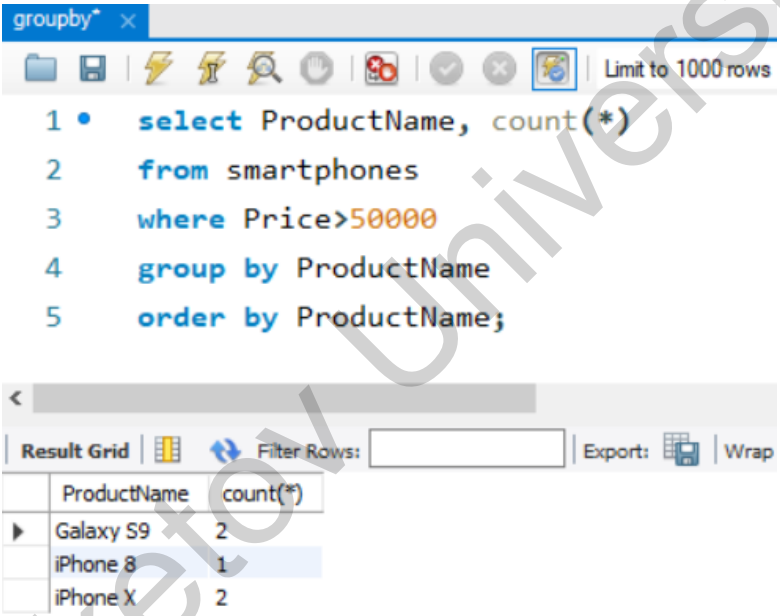


184 - сурет. GROUP BY операторы арқылы жазбаларды топтау

```
SELECT ProductName, count(*)
FROM smartphones
GROUP BY ProductName;
```

COUNT(*) – топталған тауарлардың саны

smartphones кестесіндегі бағасы 50000-нан жоғары тауарлардың аты бойынша топтайық. Тауар аты бойынша сұрыптауды қосайық:



185 - сурет. GROUP BY операторын қолдану

```
SELECT ProductName, count(*)
FROM smartphones
WHERE Price>50000
GROUP by ProductName
ORDER BY ProductName;
```

smartphones кестесі:

```
1 •  select * from smartphones;
```

Result Grid | Filter Rows: | Edit:

Id	ProductName	ProductCount	Price
1	iPhone X	3	76000
2	iPhone 8	2	51000
3	Galaxy S9	2	56000
4	Galaxy S8	1	41000
5	P20 Pro	5	36000
6	iPhone X	2	102000
7	Galaxy S9	2	59000
8	iPhone X	3	102500
9	Galaxy S8	3	45000
10	iPhone 8	3	60500

186 - сурет. *smartphones* кестесі

smartphones кестесіндегі тауарларды аты бойынша топтайық. Тауар аты бойынша сұрыптауды қосамыз:

groupby*

Limit to 1000 rows

```
1   select ProductName, count(*)
2   from smartphones
3   group by ProductName
4   order by ProductName;
```

Result Grid | Filter Rows: | Export: | Wrap

ProductName	count(*)
Galaxy S8	2
Galaxy S9	2
iPhone 8	2
iPhone X	3
P20 Pro	1

187 - сурет. Жазбаларды топтау

Топтарды сүзгілеу. HAVING операторы

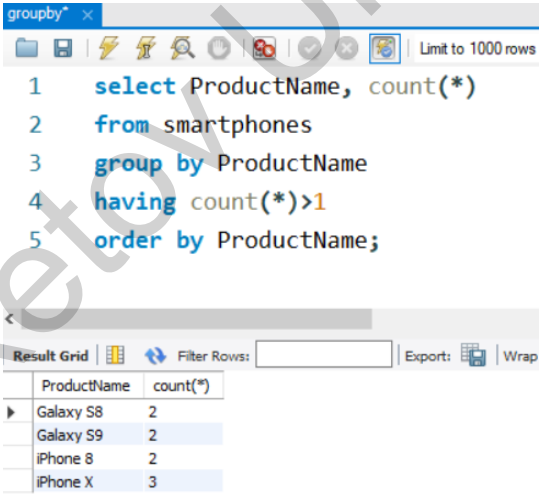
HAVING операторы GROUP BY арқылы топталған мәндерді сүзгілеуге мүмкіндік береді, яғни нәтижесінде қандай топталған мәндер алынатынын көрсетеді.

HAVING-ті қолдану WHERE-ге ұқсас. HAVING операторы арқылы шарт қойып қажетті жолдарды аламыз.

Келесідей скрипт жазайық:

```
SELECT ProductName, count(*)
FROM smartphones
GROUP BY ProductName
HAVING count(*)>1
ORDER BY ProductName;
```

Тауардың аты бойынша топтаймыз (GROUP BY ProductName), және топталған жолдар саны 1-ден көп (HAVING COUNT(*)>1) топтарды ғана аламыз.

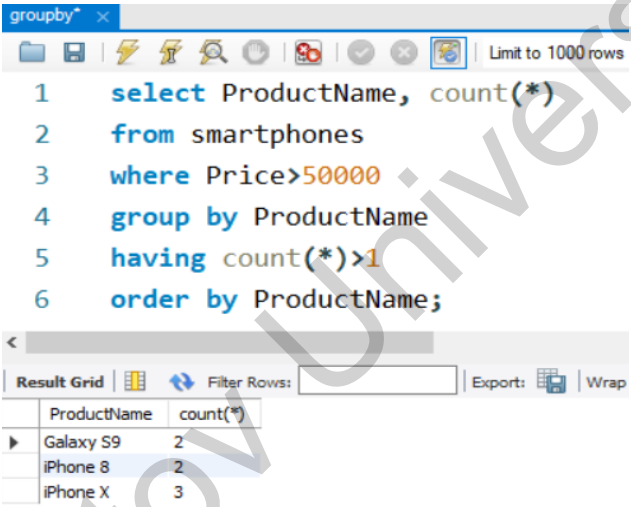


188 - сурет. Қайталатын жазбаларды алу

```
SELECT ProductName, count(*)
FROM smartphones
WHERE Price>50000
```

```
GROUP BY ProductName
HAVING COUNT(*)>1
ORDER BY ProductName;
```

Тауардың бағасы 50000-нан жоғары (WHERE Price>50000) смартфондарды аты бойынша топтаймыз (GROUP BY ProductName), және топталған жолдар саны 1-ден көп (HAVING COUNT(*)>1) топтарды ғана аламыз.



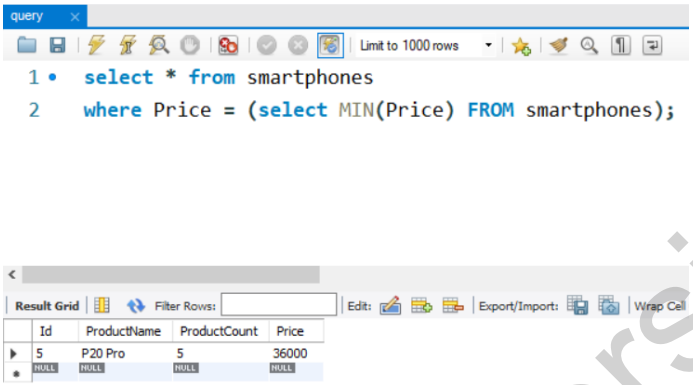
189 - сурет. Бағасы 50000-нан жоғары, саны 1-ден көп тауарлар тізімі

4.7 Ішкі сұраныстар

Ішкі сұраныстар басқа SQL сұраныстарына енгізілген SELECT өрнектерін білдіреді. Ішкі сұраныстарды қолданудың қарапайым мысалын қарастырайық.

Мысал, бағасы ең төмен смортфонды алу сұранысы:

```
SELECT * FROM smartphones
WHERE Price = (SELECT MIN(Price) FROM smartphones);
```

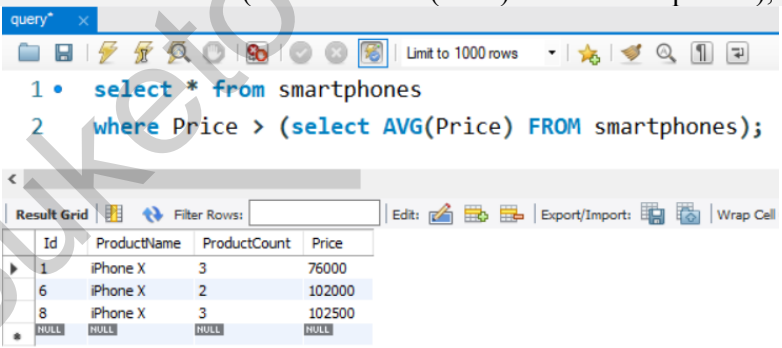


190 - сурет. Ішкі сұраныс. Ең кіші баға

Мұндағы SELECT MIN(Price) FROM smartphones; – ішкі сұраныс, *smartphones* кестесіндегі ең төмен бағаны анықтайды. WHERE Price = (SELECT MIN(Price) FROM smartphones); арқылы бағасы ең төмен бағаға тең смартфонды аламыз.

Мысал, Бағасы орташа бағадан жоғары смартфондар тізімін алайық.

```
SELECT * FROM smartphones
WHERE Price > (SELECT AVG(Price) FROM smartphones);
```



191 - сурет. Ішкі сұраныс

Мұндағы SELECT AVG(Price) FROM smartphones;– ішкі сұраныс, smartphones кестесіндегі смартфондар бағаларының

орташа мәнін анықтайды. WHERE Price > (SELECT AVG(Price) FROM smartphones); арқылы бағасы орташа бағадан жоғары смартфон тізімін аламыз.

Корреляциялық және корреляциялық емес ішкі сұраныстар

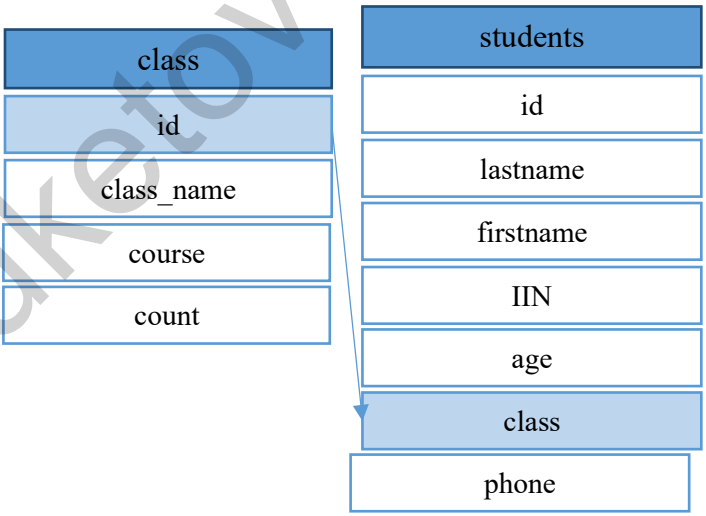
Ішкі сұраныстар корреляциялық және корреляциялық емес болып бөлінеді. Жоғарыдағы мысалдарда SELECT командалары команда шығарған барлық жолдар үшін бір ішкі сұранысты орындады. Мысалы, ішкі сұрау минималды немесе орташа бағаны қайтарады, ол негізгі сұраныста қанша жолды таңдасақ та өзгермейді. Яғни, ішкі сұраудың нәтижесі негізгі сұрауда таңдалған жолдарға байланысты болмады. Мұндай ішкі сұрау бүкіл сыртқы сұрау үшін бір рет орындалады.

Сонымен қатар сіз корреляциялық ішкі сұраныстарды (correlated subquery) қолдана аласыз, олардың нәтижелері негізгі сұрауда таңдалған жолдарға байланысты болады.

Мысал

Студентің аты-жөні мен тобының нөмірі емес, атын шығарайық.

Студенттің аты-жөні, топ нөмірі *students* кестесінде берілген. Ал топ нөмірі сәйкес топтың атауы *class* кестесінде берілген.



3 - сызба. *class* пен *students* кестелерінің байланысының сызбасы

students кестесіндегі жазбалар

id	lastname	firstname	iin	age	class	phone
2	Aitpayeva	Dariga	010306123456	21	7	87000000 00
3	Aryn	Sagadat	011128123456	21	7	87000000 00
4	Baimukhanova	Leyla	020616123456	20	1	87000000 00
5	Ersainov	Asan	020721123456	20	1	87000000 00
7	Meyram	Talgat	021010123456	20	2	87000000 00
8	Temirbayqyzy	Madina	010518123456	21	6	87000000 00

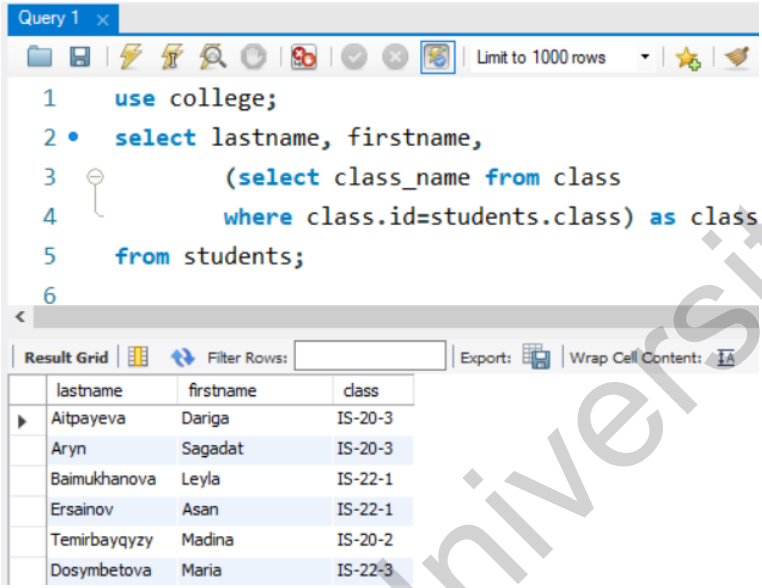
class кестесіндегі жазбалар

Id	Class_name	Course	count
1	IS-22-1	1	23
2	IS-22-2	1	23
3	IS-22-3	1	1
4	IS-22-3	1	21
5	IS-20-1	1	23
6	IS-20-2	2	10
7	IS-20-3	2	23
8	IS-20-5	2	16
9	IS-19-1	3	19
10	IS-19-2	3	11

students.class өрісінде студенттің тобының нөмірі көрсетілген. Бұл нөмірі *class.id* (class кестесі) өрісіне сәйкес келеді.

Осы кестелерден әр студент және оның тобы (топ атауы) туралы ақпаратты алайық.

```
SELECT lastname, firstname,
        (SELECT class_name FROM class
        WHERE class.id=students.class) as class
FROM students;
```



192 - сурет. Ішкі сұранысты қолдану

Мұндағы ішкі сұраныс

```
SELECT class_name FROM class
WHERE class.id=students.class
```

class кестесінен студент тобының нөміріне сәйкес топтың атауын алады. Алынған мәндер бағанына *class* атауы берілген (as class).

Нәтижесі:

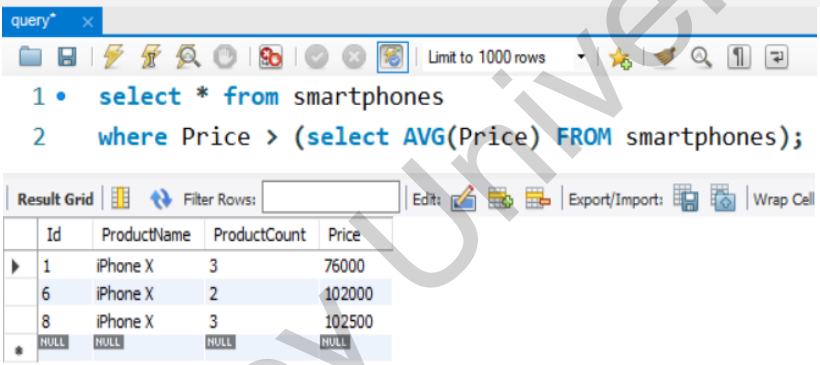
Aitpayeva	Dariga	IS-20-3
Aryn	Sagadat	IS-20-3
Baimukhanova	Leyla	IS-22-1
Ersainov	Asan	IS-22-1
Meyram	Talgat	IS-22-2
Temirbayqyzy	Madina	IS-20-2

4.8 SQL негізгі операторларындағы ішкі сұраныстар

SELECT өрнегінде ішкі сұраныстарды төрт жолмен енгізуге болады:

1. WHERE шартында
2. HAVING шартында
3. FROM өрнегінде кесте ретінде
4. SELECT өрнегіндегі өрістің сипаттамасы ретінде

```
SELECT * FROM smartphones
WHERE Price > (SELECT AVG(Price) FROM smartphones);
```



193 - сурет. SELECT өрнегінде ішкі сұраныс

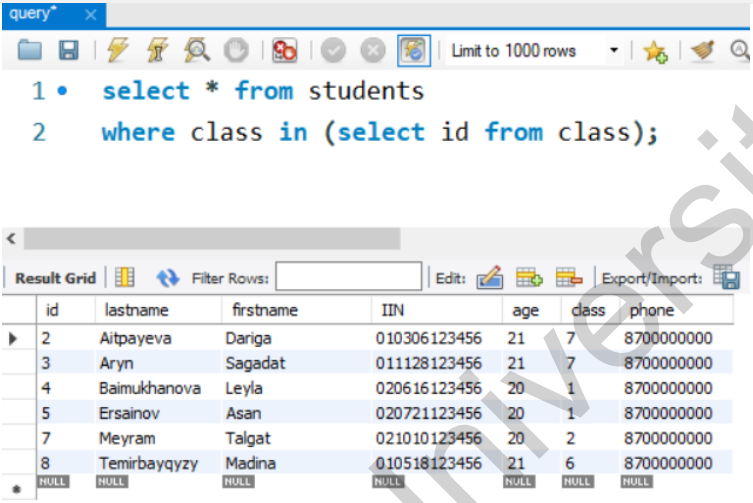
Мұндағы SELECT AVG(Price) FROM smartphones;– ішкі сұраныс, smartphones кестесіндегі смартфондар бағаларының орташа мәнін анықтайды. WHERE Price > (SELECT AVG(Price) FROM smartphones); арқылы бағасы орташа бағадан жоғары смартфон тізімін аламыз.

IN операторы

Көбінесе ішкі сұраныстар мәндер жиынынан таңдайтын IN операторымен бірге қолданылады. Ішкі сұраныс қажетті мәндер жиынтығын бере алады.

Мысалы, *class* кестесінде топта оқитын студенттерді таңдаймыз.

```
SELECT * FROM students
WHERE class IN (SELECT id FROM class);
```



id	lastname	firstname	IIN	age	class	phone
2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
7	Meyram	Talgat	021010123456	20	2	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

194 - сурет. SELECT өрнегінде ішкі сұраныс

Мысалы, *class* кестесінде топта оқымайтын студенттерді таңдаймыз.

```
SELECT * FROM students
WHERE class NOT IN (SELECT id FROM class);
```

Мәндер жиынтығын алу

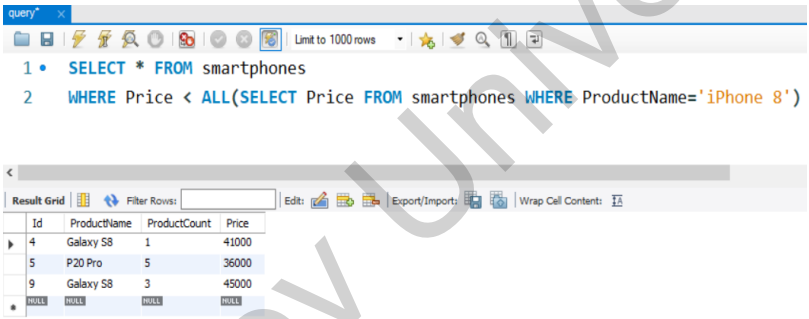
Салыстыру операторларын қолданған кезде сұраныстар бір скалярлық мәнді қайтаруы керек. Бірақ кейде мәндер жиынтығын алу қажет болады. Салыстыру операторларын қолданған кезде ішкі сұраныстар мәндер жиынтығын қайтаруы үшін оның алдында келесі операторлардың бірін қолдану керек: ALL, SOME немесе ANY.

ALL кілттік сөзін қолданған кезде салыстыру операциясындағы шарт ішкі сұраныспен қайтарылатын барлық мәндерге қатысты болуы керек.

Мысалы, iPhone 8 смартфонының бағасынан аз смартфондар тізімін алу.

Кестеде iPhone 8 екі рет кездесейді, 2 және 10 жазбалар.

1	iPhone X	3	76000
2	*iPhone 8*	*2*	*51000*
3	Galaxy S9	2	56000
4	Galaxy S8	1	41000
5	P20 Pro	5	36000
6	iPhone X	2	102000
7	Galaxy S9	2	59000
8	iPhone X	3	102500
9	Galaxy S8	3	45000
10	*iPhone 8*	*3*	*60500*



195 - сурет. ALL кілттік сөзін қолдану

```
SELECT * FROM smartphones
WHERE Price < ALL(SELECT Price FROM smartphones WHERE ProductName='iPhone 8')
```

4	Galaxy S8	1	41000
5	P20 Pro	5	36000
9	Galaxy S8	3	45000

Ішкі сұраныс

```
SELECT Price FROM smartphones WHERE ProductName='iPhone 8'
```

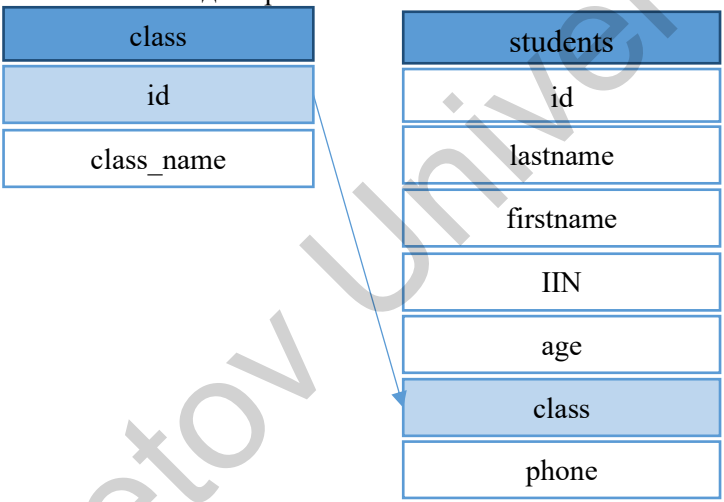
екі мәнді алады: 51000, 60500

ANY (сондай ақ SOME):

Өрнек	Эквиваленті
x > ANY (1, 2)	x > 1
x < ANY (1, 2)	x < 2
x = ANY (1, 2)	x IN (1, 2)
x <> ANY (1, 2)	(x <> 1) OR (x <> 2)

Өрісті ішкі сұраныс арқылы алу

Студенттің аты-жөні, топ нөмірі *students* кестесінде берілген. Ал *students* кестесіндегі топ нөміріне сәйкес топтың атауы *class* кестесінде берілген.



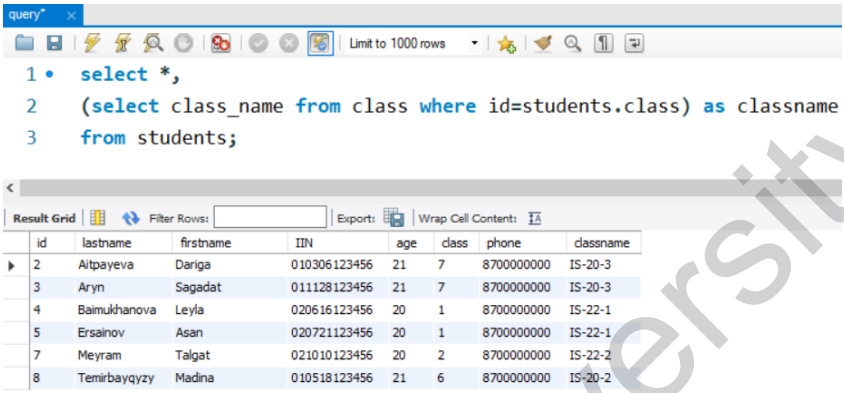
4-сызба. *students* пен *class* кестелерінің байланысы

Бір сұраныс арқылы студент туралы барлық ақпартатты (*students.**) және студенттің тобының атауын (*class.class_name*) алайық.

class кестесінен студент тобының нөміріне сәйкес топтың атауын алады. Алынған мәндер бағанына *classname* атауы берілген (*AS classname*).

```
SELECT *,
(SELECT class_name FROM class WHERE id=students.class)
AS classname
```

FROM students;



id	lastname	firstname	IIN	age	class	phone	classname
2	Aitpayeva	Dariga	010306123456	21	7	8700000000	IS-20-3
3	Aryn	Sagadat	011128123456	21	7	8700000000	IS-20-3
4	Baimukhanova	Leyla	020616123456	20	1	8700000000	IS-22-1
5	Ersainov	Asan	020721123456	20	1	8700000000	IS-22-1
7	Meyram	Talgat	021010123456	20	2	8700000000	IS-22-2
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000	IS-20-2

196 - сурет. Ішкі сұраныс арқылы өріс мәндерін алу

SELECT *, - students кестесіндегі барлық бағандар

Ішкі сұраныс

SELECT class_name FROM class WHERE id=students.class

class кестесінен студент тобының нөміріне сәйкес топтың атауын алады. Алынған мәндер бағанына *classname* атауы берілген (*AS classname*).

2	Aitpayeva	Dariga	010306123456	21	7	8700000000	IS-20-3
3	Aryn	Sagadat	011128123456	21	7	8700000000	IS-20-3
4	Baimukhano-va	Leyla	020616123456	20	1	8700000000	IS-22-1
5	Ersainov	Asan	020721123456	20	1	8700000000	IS-22-1
7	Meyram	Talgat	021010123456	20	2	8700000000	IS-22-2
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000	IS-20-2

INSERT командасындағы ішкі сұраныстар

INSERT командасында ішкі сұраныстар бағандардың біріне енгізілген мәнді анықтау үшін қолданылуы мүмкін:

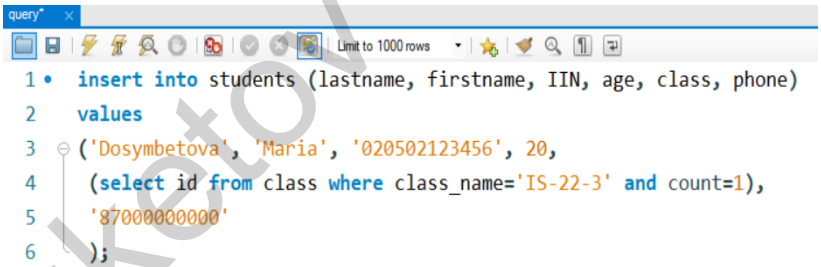
IS-22-3 тобына оқитын *students* кестесіне жаңа студентті қосайық:

students кестесі

2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
7	Meyram	Talgat	021010123456	20	2	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000

class кестесі

1	IS-22-1	1	23
2	IS-22-2	1	23
3	IS-22-3	1	1
4	IS-22-3	1	21
5	IS-20-1	1	23
6	IS-20-2	2	10
7	IS-20-3	2	23
8	IS-20-5	2	16
9	IS-19-1	3	19
10	IS-19-2	3	11

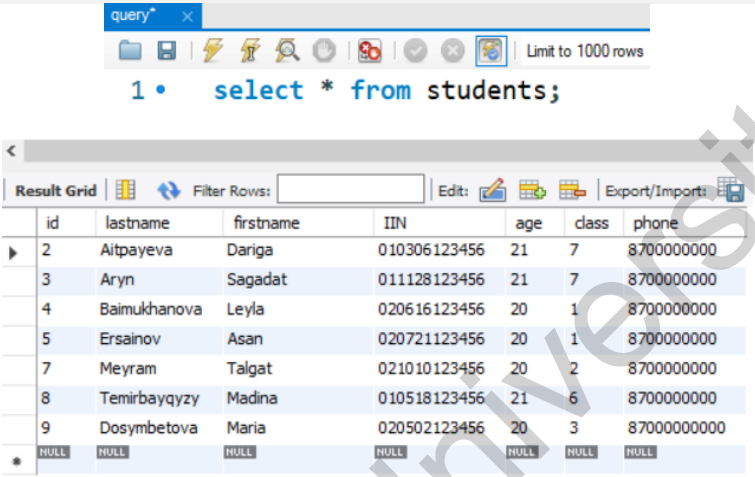


197 - сурет. INSERT командасындағы ішкі сұраныс

```
INSERT INTO students (lastname, firstname, IIN, age, class, phone)
VALUES
('Dosymbetova', 'Maria', '020502123456', 20,
 (SELECT id FROM class WHERE class_name='IS-22-3' and count=1),
 '87000000000'
 );
```

Топтың нөмірін ішкі сұраныс арқылы алдық:

```
SELECT id FROM class WHERE class_name='IS-22-3' and count=1
```



id	lastname	firstname	IIN	age	class	phone
2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
7	Meyram	Talgat	021010123456	20	2	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
9	Dosymbetova	Maria	020502123456	20	3	87000000000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

198 - сурет. *students* кестесі

students кестесі

2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
7	Meyram	Talgat	021010123456	20	2	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
9	Dosymbetova	Maria	020502123456	20	3	8700000000

UPDATE командасындағы ішкі сұраныстар

UPDATE командасына:

1) SET операторыдағы мән ретінде
2) WHERE өрнегінде шарттың бір бөлігіретінде қолданылады.

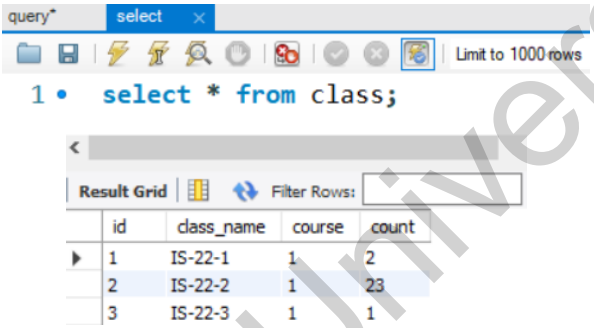
IS-22-1 тобында кесте бойынша 23 студент оқиды. Шын мәнінде бұл топтағы студенттер санын анықтап, студенттер санын (*class.count*) өзгертейік.

IS-22-1 тобының id нөмірі – 1

```
UPDATE class
```

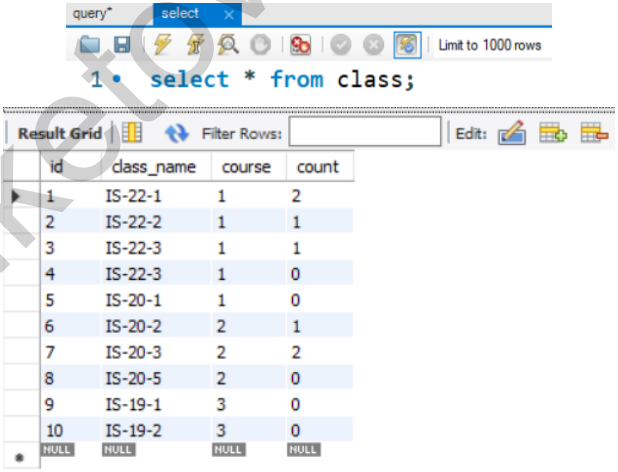
```
    SET count= (SELECT count(*) FROM students WHERE class=1)
    WHERE id=1;
```

Ішкі сұраныс SELECT count(*) FROM students WHERE class=1 арқылы *students* кестесінен 1 топта неше студент бар екенін анықтаймыз. Анықталған мәнді IS-22-1 тобындағы студенттер саны ретінде орнатамыз (SET count=…).



199 - сурет. *class* кестесі

Осы әдіспен барлық топтағы студенттер санын өзгертіп шығамыз.



200 - сурет. *class* кестесі

DELETE командасында сұраныс

DELETE командасында ішкі сұраныстар шарттың бөлігі ретінде де қолданылады

'IS-22-2' тобында оқитын барлық студенттерді *students* кестесінен өшіріп тастаймыз:

```
DELETE FROM students
WHERE class=(SELECT Id FROM class WHERE class_name='IS-22-2');
```

4.9 EXISTS операторы

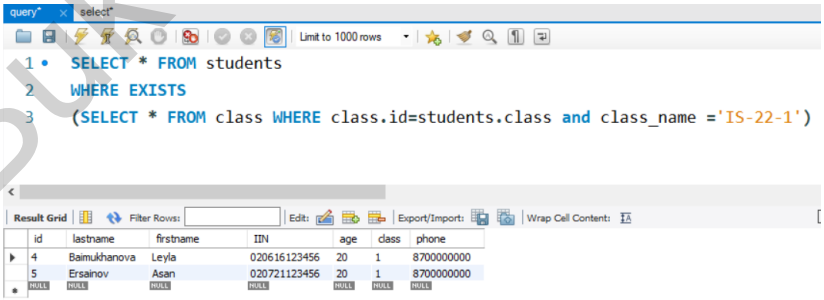
EXISTS операторы ішкі сұраныстың қандай да бір мәнді қайтаратынын тексереді. Әдетте, бұл оператор кестеде кем дегенде бір жол белгілі бір шартты қанағаттандыратындығын көрсету үшін қолданылады.

Операторды қолдану келесі ресми синтаксиске ие:

```
WHERE [NOT] EXISTS (ішкі сұраныс)
```

'IS-22-1' оқитын студенттер тізімін алу:

```
SELECT * FROM students
WHERE EXISTS
(SELECT * FROM class WHERE class.id=students.class and class_name ='IS-22-1')
```

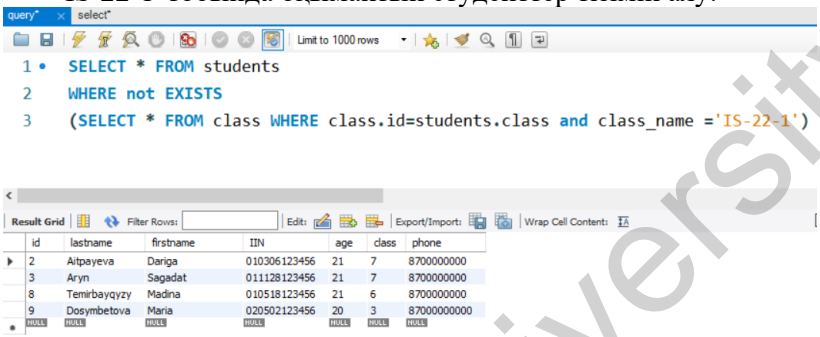


201 - сурет. EXISTS операторын қолдану

Нәтижесі:

4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000

'IS-22-1' тобында оқымайтын студенттер тізімін алу:



202 - сурет. EXISTS операторын қолдану

```
SELECT * FROM students
WHERE NOT EXISTS
(SELECT * FROM class WHERE class.id=students.class and class_name ='IS-22-1')
```

Нәтижесі:

2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
9	Dosymbetova	Maria	020502123456	20	3	87000000000

Егер біз, керісінше, кестеде шартты қанағаттандырмайтын жолдар бар-жоғын білгіміз келсе, онда NOT EXISTS қолдануға болады. Мысалы, *class* кестесінде студенттері жоқ топтарды алу:

```
SELECT * FROM class
WHERE NOT EXISTS (SELECT * FROM students WHERE class.Id = students.class)
```

```
SELECT * FROM class
WHERE NOT EXISTS (SELECT * FROM students WHERE class.Id = students.class)
```

id	class_name	course	count
2	IS-22-2	1	1

203 - сурет. EXISTS операторын қолдану

Айта кету керек, мұндай нәтижеге қол жеткізу үшін IN операторында қолдануға болады:

```
SELECT *
FROM class
WHERE Id NOT IN (SELECT class FROM students)
```

id	class_name	course	count
2	IS-22-2	1	1

204 - сурет. IN операторын қолдану

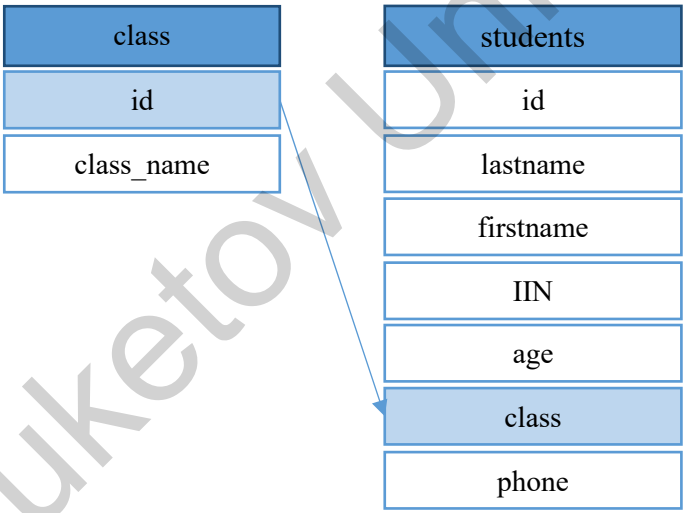
5 тарау. Кестелерді біріктіру

5.1 Кестелерді айқындалмаған түрде біріктіру

Көбінесе бір сұраныспен бірнеше байланысқан кестеден деректерді алу қажеттілігі туындайды. Кестелерде бір-біріне байланысты өрістер болуы керек. Негізгі кестенің бастапқы кілті және тәуелді кестенің сыртқы кілті бірігу үшін қолданылады. Бірнеше кестелердегі деректерді бір сұраныспен алу үшін кестелерді біріктіру әдістері қолданылады.

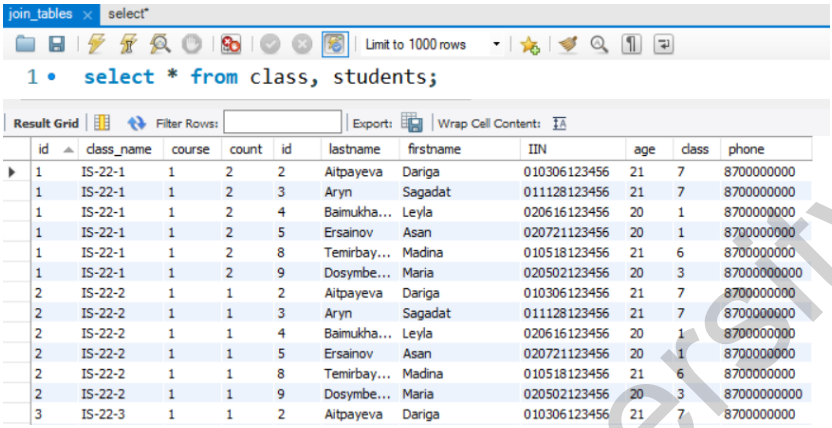
Қарапайым бірақ ең көп таралмаған кестелердің анық емес бірігуін қарастырайық.

Бір-бірімен байланысқан *class* және *students* кестелері бар делік. Кестелердің құрылымы төмендегі сызбада берілген.



5 сызба. *class* және *students* кестелерінің байланыс сызбасы

class кестесінде топтар тізімі, *students* кестесінде студенттердің тізімі берілген. Бұл кестелер бір бірімен топтың нөмірі өрісімен байланысады: *class.id = students.class*. Бір бірімен байланысқан *class* және *students* кестелерінен деректерді алудың ең қарапайым сұранысы:



id	class_name	course	count	id	lastname	firstname	IIN	age	class	phone
1	IS-22-1	1	2	2	Aitpayeva	Dariga	010306123456	21	7	8700000000
1	IS-22-1	1	2	3	Aryn	Sagadat	011128123456	21	7	8700000000
1	IS-22-1	1	2	4	Baimukha...	Leyla	020616123456	20	1	8700000000
1	IS-22-1	1	2	5	Ersainov	Asan	020721123456	20	1	8700000000
1	IS-22-1	1	2	8	Temirbay...	Madina	010518123456	21	6	8700000000
1	IS-22-1	1	2	9	Dosymbe...	Maria	020502123456	20	3	87000000000
2	IS-22-2	1	1	2	Aitpayeva	Dariga	010306123456	21	7	8700000000
2	IS-22-2	1	1	3	Aryn	Sagadat	011128123456	21	7	8700000000
2	IS-22-2	1	1	4	Baimukha...	Leyla	020616123456	20	1	8700000000
2	IS-22-2	1	1	5	Ersainov	Asan	020721123456	20	1	8700000000
2	IS-22-2	1	1	8	Temirbay...	Madina	010518123456	21	6	8700000000
2	IS-22-2	1	1	9	Dosymbe...	Maria	020502123456	20	3	87000000000
3	IS-22-3	1	1	2	Aitpayeva	Dariga	010306123456	21	7	8700000000

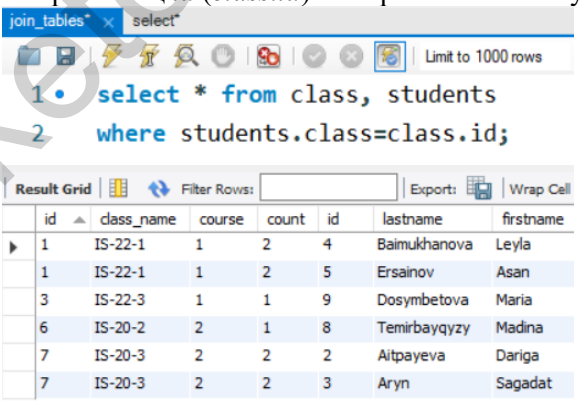
205 - сурет. Екі кестеден бағандарды алу

```
SELECT * FROM class, students;
```

Бірақ мұндай сұраныстан дұрыс нәтиже алынбайды. Әр студенттің сәйкес топ атауы дұрыс шықпайды. Әр студенттің оқитын тобының атауын дұрыс шығару үшін WHERE операторын пайдаланамыз.

Әр студент жайлы және оның тобы туралы ақпаратты алу сұранысы

Студенттің тобының нөмірі (*students.class*) *class* кестесіндегі әр топтың *id* (*class.id*) нөмеріне сәйкес келуі қажет.



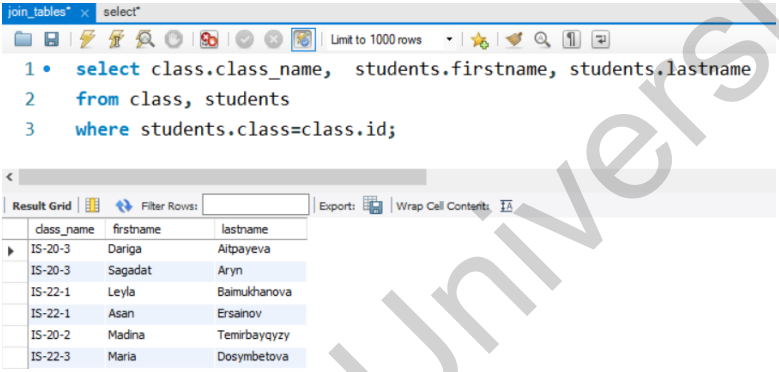
id	class_name	course	count	id	lastname	firstname
1	IS-22-1	1	2	4	Baimukhanova	Leyla
1	IS-22-1	1	2	5	Ersainov	Asan
3	IS-22-3	1	1	9	Dosymbetova	Maria
6	IS-20-2	2	1	8	Temirbayqyzy	Madina
7	IS-20-3	2	2	2	Aitpayeva	Dariga
7	IS-20-3	2	2	3	Aryn	Sagadat

206 - сурет. Екі байланысқан кестеден деректерді алу

```
SELECT * FROM class, students
WHERE students.class=class.id;
```

Жекелеген өрістерді (бағандарды) ғана алу

Бірнеше кестеден өрістерді таңдау үшін SELECT операторынан кейін өрістер кесте атауымен бірге жазылуы керек. Кесте атауынан кейін нүкте қойылып, өріс атауы жазылады.



207 - сурет. Екі байланысқан кестеден өрістерді таңдап алу

```
SELECT
  class.class_name, students.firstname, students.lastname

FROM class, students
WHERE students.class=class.id;
```

class кестесінен *class_name* өрісі, *students* кестесінен *firstname, lastname* өрістері алынды.

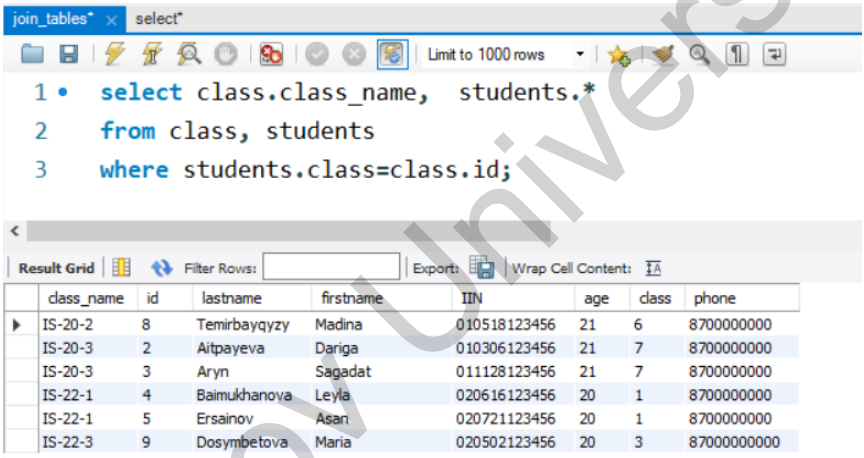
Нәтиже:

IS-20-3	Dariga	Aitpayeva
IS-20-3	Sagadat	Aryn
IS-22-1	Leyla	Baimukhanova
IS-22-1	Asan	Ersainov
IS-20-2	Madina	Temirbayqyzy
IS-22-3	Maria	Dosymbetova

class кестесінен тек топ атын, ал *students* кестесінен барлық өрістерді алу:

```
SELECT class.class_name,  students.*
FROM class, students
WHERE students.class=class.id;
```

*students.** - жұлдызша белгісі *students* кестесіндегі барлық өрістерді алуды білдіреді.



class_name	id	lastname	firstname	IIN	age	class	phone
IS-20-2	8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
IS-20-3	2	Aitpayeva	Dariga	010306123456	21	7	8700000000
IS-20-3	3	Aryn	Sagadat	011128123456	21	7	8700000000
IS-22-1	4	Baimukhanova	Leyla	020616123456	20	1	8700000000
IS-22-1	5	Ersainov	Asan	020721123456	20	1	8700000000
IS-22-3	9	Dosymbetova	Maria	020502123456	20	3	87000000000

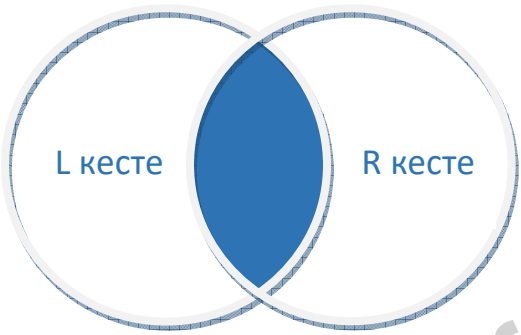
208 - сурет. Екі байланысқан кестеден өрістерді таңдап алу

5.2 JOIN операторы

5.1 бөлімде байланысқан кестелерден деректерді алу үшін бірігудің анықталмаған түрін қарастырдық. Бірақ, әдетте, кестелерді біріктірудің кең таралған тәсілі JOIN операторы арқылы байланыстыру болып табылады.

MySQL қолданылатын JOIN операторларының түрлері:

- INNER JOIN: Екі немесе бірнеше өзара байланысқан кестелерден тек сәйкес жазбаларды алады.



209 - сурет. INNER JOIN сызбасы

```
SELECT L.*, R.*
FROM L
[INNER] JOIN R
ON L.key = R.key
```

Мұндағы, L – сол жақтағы кесте, R – оң жақтағы кесте, L.* - сол жақтағы кестенің өрістері, R.* - оң жақтағы кестенің өрістер.

- LEFT [OUTER] JOIN: Сол жақтағы (FROM кейін көрсетілген кесте) барлық жазбаларды, ал оң жақтағы кестеден сол жақтағы сәйкес келетін жазбаларды ғана алады.

210 - сурет. LEFT JOIN сызбасы

```
SELECT L.*, R.*
FROM L
LEFT JOIN R
```

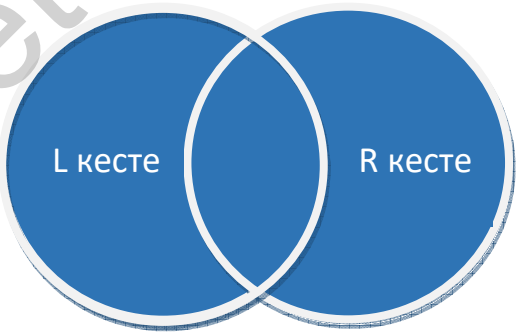
```
ON L.key = R.key
```

- RIGHT [OUTER] JOIN: Оң (JOIN кейін көрсетілген кесте) жақтағы барлық жазбаларды, ал оң жақтағы кестеден оң жақтағы сәйкес келетін жазбаларды ғана алады.

211 сурет. RIGHT JOIN сызбасы

```
SELECT L.*, R.*
FROM L
RIGHT JOIN R
ON L.key = R.key
```

FULL OUTER JOIN операторы: кестелердегі барлық жазбаларды қайтарады. Бірақ MySQL-де бұл оператор жоқ.



212 - сурет. FULL OUTER JOIN сызбасы

INNER JOIN операторы

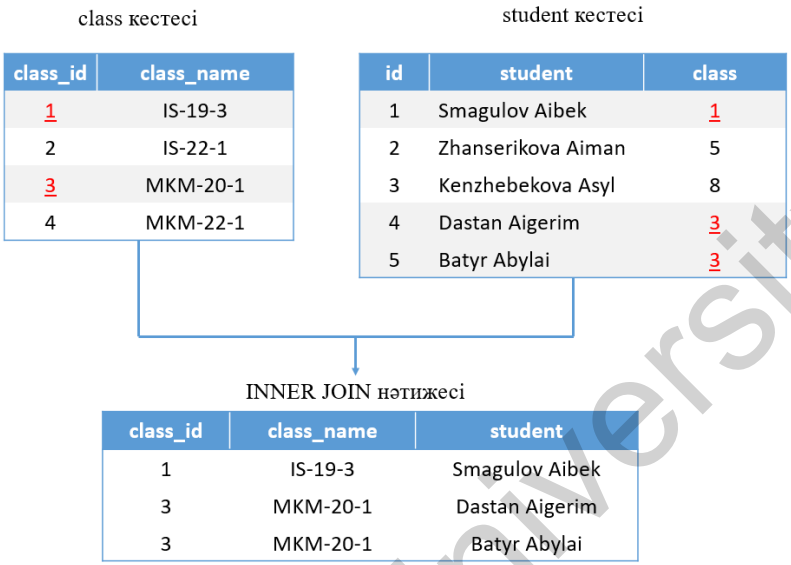
Егер кестелердің байланыстырушы өрістерінде бірдей мәндер болса, екі кестенің жазбаларын алады.

INNER JOIN операторының жалпы синтаксисі:

```
SELECT бағандар
FROM кесте1
[INNER] JOIN кесте2
ON шарт1
[[INNER] JOIN кесте3
ON шарт2]
```

SELECT сөзінен кейін барлық кестелерден өрістері тізбектеліп жазылады. Өрістер атауы кесте атауымен бірге көрсетіледі. Себебі, кестелерде атаулары бірдей өрістер болуы мүмкін. Кесте атауы мен өріс атауы нүкте (.) арқылы жазылады, мысалы *class.name – class* кестесіндегі *name* өрісі, *students.name – students* кестесіндегі *name* өрісі. INNER JOIN операторынан кейін екінші кестенің атауы көрсетіледі. INNER JOIN орнына тек JOIN деп көрсетуге болады. MySQL үнсіздік келісім бойынша JOIN деп жазылса, оны INNER JOIN деп түсінеді. ON кілттік сөзінен кейін байланысу шарты көрсетіледі. Бұл шарт екі кестенің ортақ өрістерін көрсетеді. Көп жағдайда негізгі кестенің бастапқы кілті және тәуелді кестенің сыртқы кілті бірігу үшін қолданылады.

INNER JOIN операторының жұмысын сызба ретінде қарастырайық.



6-сызба. INNER JOIN нәтижесі

```
SELECT class.class_id, class.class_name, student.student
FROM class
JOIN student
ON student.class=class.class_id;
```

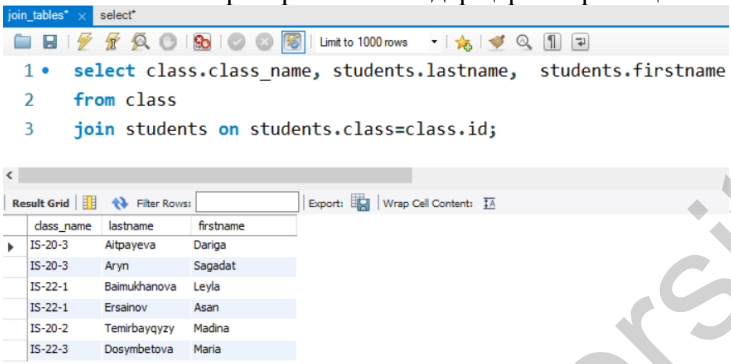
class кестесінде топтар тізімі, ал студенттер тізімі *student* кестесінде берілген. *student* мен *class* кестелерін байланыстырушы өріс – топтың нөмірі (*class.class_id* = *student.class*). Егер кестелердің байланыстырушы өрістерінде бірдей мәндер болса, екі кестенің жазбаларын алады.

INNER JOIN операторының нәтижесінде келесі жазбалар шығарылмады:

– Нөмері (*class_id*) 2 және 4 топтарда оқитын студенттердің тізімі *student* кестесінде жоқ.

– Нөмірі (*id*) студенттердің топ нөмері 5 және 8. Бұл номердегі топтар *class* кестесінде жоқ.

INNER JOIN операторына мысалдар қарастырайық.

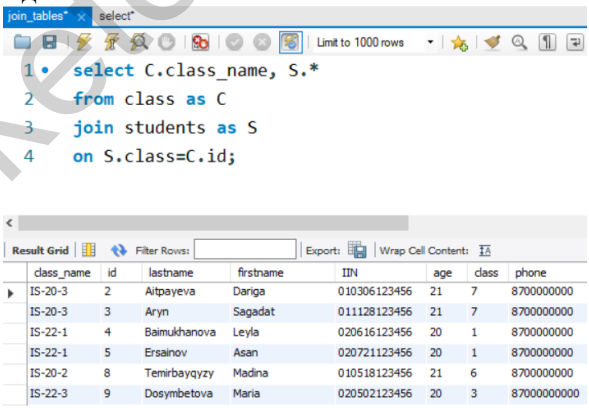


213 - сурет. [INNER] JOIN операторын қолдану

```
SELECT class.class_name, students.lastname, students.firstname
FROM class
JOIN students
ON students.class=class.id;
```

мұнда *class* – негізгі кесте, *students* – тәуелді кесте. class кестесінен *class_name*, *students* кестесінен *lastname* және *firstname* өрістері таңдалынды.

Кодты қысқарту үшін кесте атауы үшін AS кілттік сөз арқылы көрсетілген қысқартылған атаулар (бүркеншік) қолданылады.



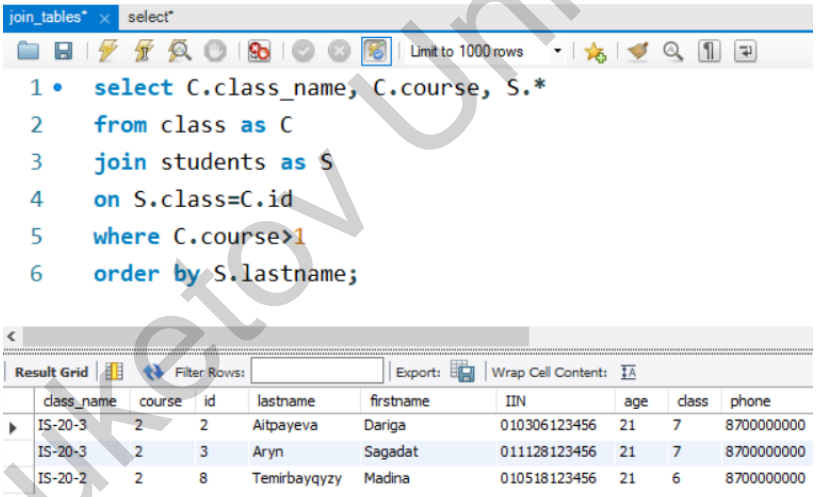
214 - сурет. [INNER] JOIN операторын қолдану

```
SELECT C.class_name, S.*
FROM class AS C
JOIN students AS S
ON S.class=C.id;
```

Кестелерді біріктіруде сүзгілеу немесе сұрыптау әдістерін қолдана аламыз.

2, 3 курс топта оқитын студенттер туралы ақпарат алу:

```
SELECT C.class_name, C.course, S.*
FROM class AS C
JOIN students AS S
ON S.class=C.id
WHERE C.course>1
ORDER BY S.lastname;
```

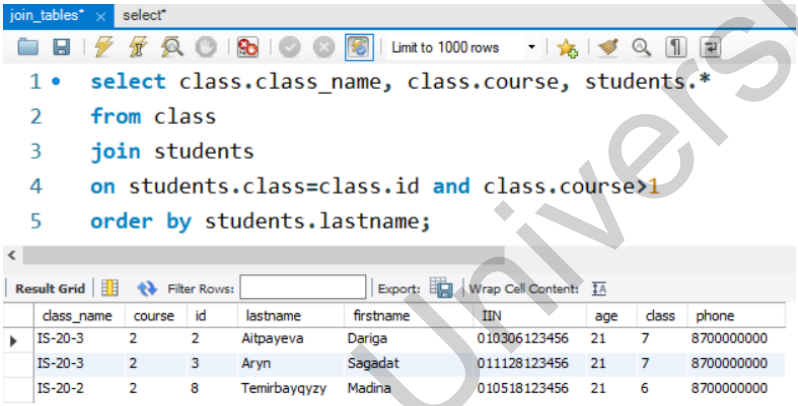


215 - сурет. [INNER] JOIN операторын қолдану

ON кілттік сөзінен кейінгі шарттар құрамы жағынан күрделі болуы мүмкін.

2, 3 курс топта оқитын студенттер туралы ақпарат алу.

```
SELECT class.class_name, class.course, students.*
FROM class
JOIN students
ON students.class=class.id and class.course>1
ORDER BY students.lastname;
```



216 - сурет. [INNER] JOIN операторын қолдану

JOIN операторын пайдаланған кезде кестелерді қосу процесі ресурстарды қажет ететінін ескеру керек, сондықтан тек деректер қажет кестелер қосылуы керек. Кестелер неғұрлым көп қосылса, соғұрлым өнімділік төмендейді.

5.3 OUTER JOIN операторы

Алдыңғы бөлімде INNER JOIN немесе кестелердің ішкі байланысын қарастырдық. Сонымен қатар, MySQL-де сыртқы байланыс немесе OUTER JOIN деп аталатын әдісті қолдана аламыз. INNER JOIN-нан айырмашылығы, сыртқы байланыс қосылысқа қатысатын бір немесе екі кестенің барлық жолдарын қайтарады.

OUTER JOIN жалпы синтаксисі

```
SELECT бағандар
FROM кесте1
   {LEFT|RIGHT} [OUTER] JOIN кесте2 ON шарт1
   [{LEFT|RIGHT} [OUTER] JOIN кесте3 ON шарт2]...
```

JOIN операторының алдында қосылу түрін анықтайтын LEFT немесе RIGHT кілттік сөздерінің бірі көрсетіледі:

- LEFT: бірінші немесе сол жақ кестедегі барлық жолдарды қамтиды
- RIGHT: екінші немесе оң жақ кестеден барлық жолдарды қамтиды

Сондай-ақ, OUTER кілттік сөзі JOIN операторының алдында көрсетілуі мүмкін, бірақ оны қолдану міндетті емес. JOIN-ден кейін қосылатын кесте көрсетіледі, содан кейін қосылу шарты жазылады.

LEFT OUTER JOIN

class кестесі

class_id	class_name
1	IS-19-3
2	IS-22-1
3	MKM-20-1
4	MKM-22-1

student кестесі

id	student	class
1	Smagulov Aibek	1
2	Zhanserikova Aiman	5
3	Kenzhebekova Asyl	8
4	Dastan Aigerim	3
5	Batyr Abylai	3

LEFT JOIN нәтижесі

class_id	class_name	student
1	IS-19-3	Smagulov Aibek
2	IS-22-1	NULL
3	MKM-20-1	Dastan Aigerim
3	MKM-20-1	Batyr Abylai
4	MKM-22-1	NULL

7 сызба. LEFT OUTER JOIN нәтижесі

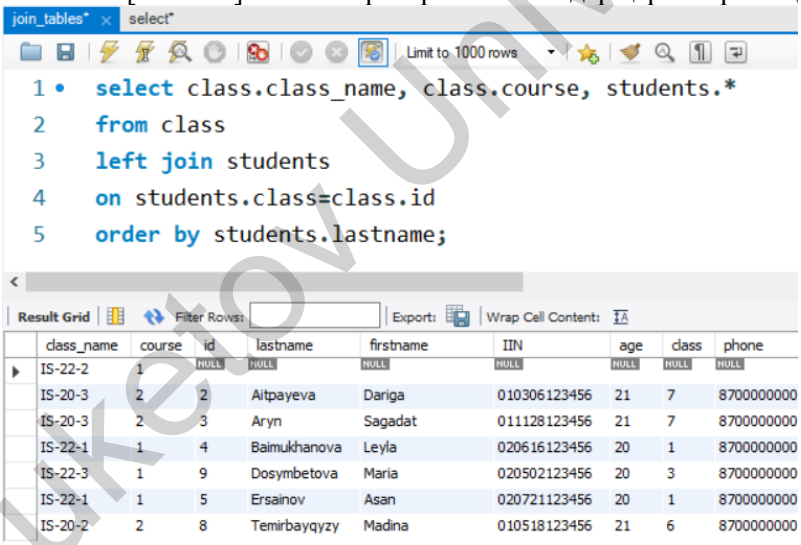
```
SELECT class.class_id, class.class_name, student.student
FROM class
LEFT JOIN student
ON student.class=class.class_id;
```

student мен *class* кестелерін байланыстырушы өріс – топтың нөмірі *(class.class_id = student.class).*

class – сол жақтағы кесте, *student* –оң жақтағы кесте

LEFT [OUTER] JOIN нәтижесінде *class* кестесіндегі барлық жазбалар және *class.class_id* өрісіне мәні сәйкес келетін *student* кестесінен жазбалар шығарылды. Егер топта оқитын студенттер болмаса, тек топ атауы шығарылады бірақ студент атауы NULL мәнімен толтырылады.

LEFT [OUTER] JOIN операторына мысалдар қарастырайық.



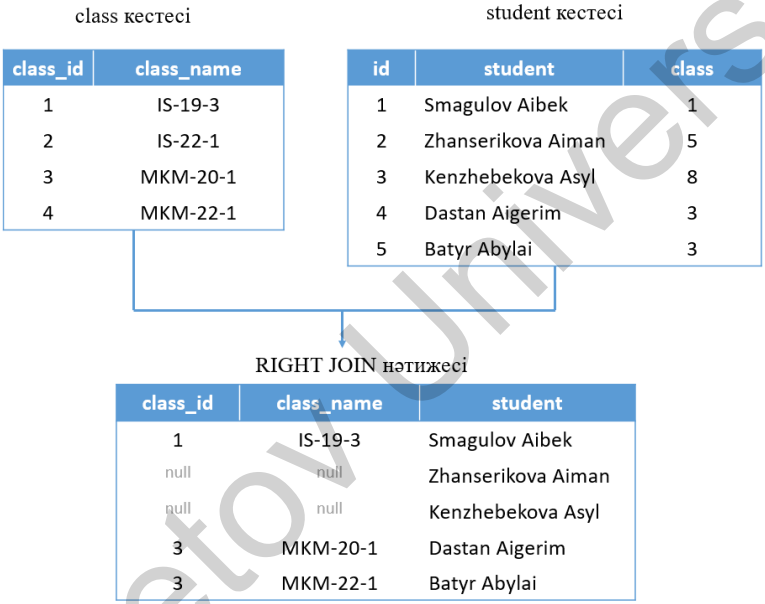
class_name	course	id	lastname	firstname	IIN	age	class	phone
IS-22-2	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL
IS-20-3	2	2	Aitpayeva	Dariga	010306123456	21	7	8700000000
IS-20-3	2	3	Aryn	Sagadat	011128123456	21	7	8700000000
IS-22-1	1	4	Baimukhanova	Leyla	020616123456	20	1	8700000000
IS-22-3	1	9	Dosymbetova	Maria	020502123456	20	3	8700000000
IS-22-1	1	5	Ersainov	Asan	020721123456	20	1	8700000000
IS-20-2	2	8	Temirbayqyzy	Madina	010518123456	21	6	8700000000

217 - сурет. LEFT [OUTER] JOIN операторын қолдану

```
SELECT class.class_name, class.course, students.*
FROM class
LEFT join students
ON students.class=class.id
ORDER BY students.lastname;
```

LEFT JOIN жағдайында MySQL алдымен *class* кестесінен барлық топтарды таңдайды, содан кейін *students.class=class.id* шарты арқылы студенттердің топ нөмерімен сәйкестендіреді, әр топта оқитын студенттер тізімін алады. IS-22-2 тобында студент жоқ, сол себепті бұл топтағы *students* кестесіне тиісті бағандар үшін NULL мәндері орнатылады.

RIGHT [OUTER] JOIN пайдалану



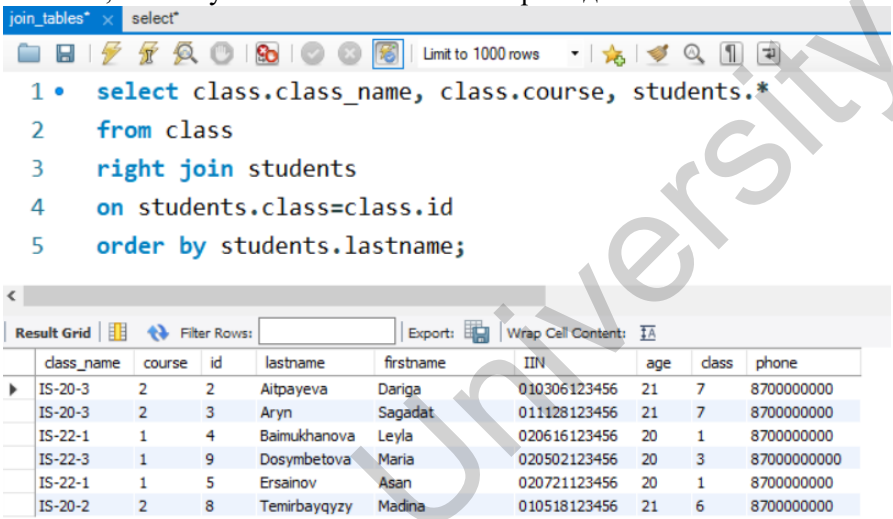
8 сызба. RIGHT [OUTER] JOIN нәтижесінің сызбасы

```
SELECT class.class_id, class.class_name, student.student
FROM class
RIGHT JOIN student
ON student.class=class.class_id;
```

student мен *class* кестелерін байланыстырушы өріс – топтың нөмірі (*class.class_id = student.class*).

class – сол жақтағы кесте, *student* – оң жақтағы кесте

RIGHT [OUTER] JOIN нәтижесінде *student* кестесіндегі барлық жазбалар және *student.class* өрісіне мәні сәйкес келетін *class* кестесінен жазбалар шығарылды. Егер студенттің оқитын тобының номеріне сәйкес келетін *class* кестесінде жазба болмаса, топ атауы NULL мәнімен толтырылады.



218 - сурет. RIGHT [OUTER] JOIN пайдалану

MySQL-де FULL [OUTER] JOIN операторы жоқ. Бірақ оны UNION операторымен көмегімен алмастыруға болады.

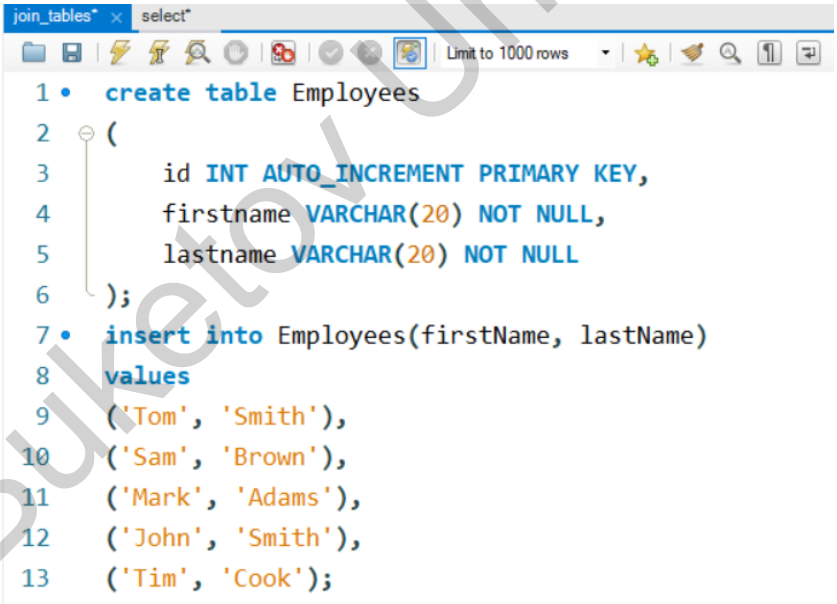
5.4 UNION операторы

UNION операторы екі бірдей іріктеуді біріктіруге мүмкіндік береді. Бұл іріктеу әртүрлі кестелерден немесе бір кестеден болуы мүмкін. Біріктірудің ресми синтаксисі:

```
SELECT бағандар1
FROM кесте1
UNION [ALL] SELECT бағандар2 FROM кесте2
[UNION [ALL] SELECT бағандарN FROM кестеN]
```

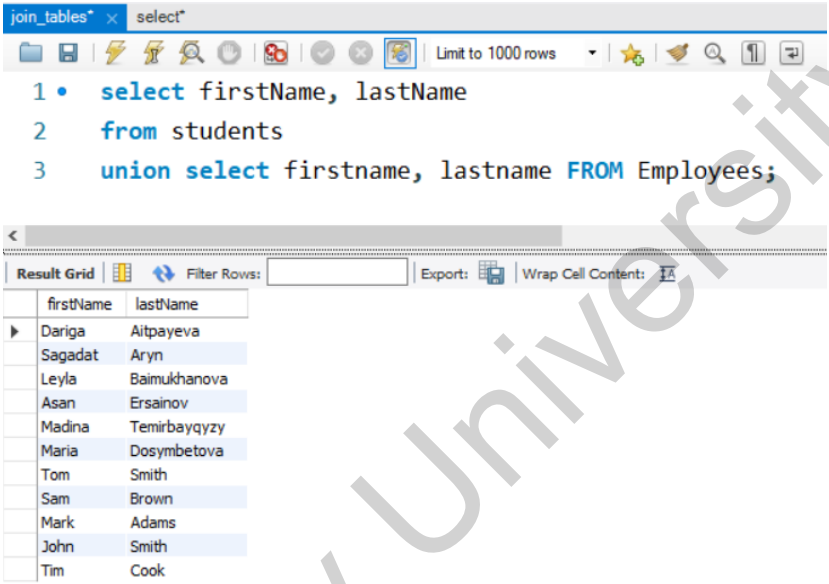
college деректер қорына тағы бір кесте қосайық. Кесте атауы – *employees*, қызметшілер туралы ақпараттар сақталады.

```
CREATE TABLE employees
(
   id INT AUTO_INCREMENT PRIMARY KEY,
   firstName VARCHAR(20) NOT NULL,
   lastName VARCHAR(20) NOT NULL
);
insert into employees(firstName, lastName)
values
('Tom', 'Smith'),
('Sam', 'Brown'),
('Mark', 'Adams'),
('John', 'Smith'),
('Tim', 'Cook');
```



219 - сурет. *employees* кестесін құру және деректер қосу

UNION операторы көмегімен *employees* және *students* кестесінен барлық адамдың аты-жөні сақталған өрістерді алайық.



220 - сурет. Бірдей бағандары бар кестелерден деректерді алу

UNION операторы арқылы екі немесе бірнеше байланысқан кестелерден жазбаларды алуға болады. Бұл оператор FULL OUTER JOIN операторын алмастырады. Екі байланысқан кестелерден UNION операторын қолданып жазбаларды алудың нәтижесі 9-сызбада көрсетілген.

Жалпы синтаксисі:

Екі байланысқан кестеден жазбаларды алу

```
SELECT * FROM кесте1
LEFT JOIN кесте2 ON кесте1.key = кесте2.key
UNION
SELECT * FROM кесте1
RIGHT JOIN кесте2 ON кесте1.key = кесте2.key
```

class кестесі

class_id	class_name
1	IS-19-3
2	IS-22-1
3	MKM-20-1
4	MKM-22-1

student кестесі

id	student	class
1	Smagulov Aibek	1
2	Zhanserikova Aiman	5
3	Kenzhebekova Asyl	8
4	Dastan Aigerim	3
5	Batyr Abylai	3

UNION нәтижесі

class_id	class_name	student
1	IS-19-3	Smagulov Aibek
2	IS-22-1	null
null	null	Zhanserikova Aiman
null	null	Kenzhebekova Asyl
3	MKM-20-1	Dastan Aigerim
3	MKM-22-1	Batyr Abylai
4	MKM-22-1	null
null	null	Zhanserikova Aiman
null	null	Kenzhebekova Asyl

9-сызба. UNION операторын нәтижесі

UNION операторын қолданып *students* пен *class* кестелерінен жазбаларды алайық. *student* мен *class* кестелерін байланыстырушы өріс – топтың нөмірі (*class.class_id = student.class*).

```
SELECT * FROM students
LEFT JOIN class ON students.class = class.id
UNION
SELECT * FROM students
RIGHT JOIN class ON students.class = class.id
```

```
1 • SELECT * FROM students
2   LEFT JOIN class ON students.class = class.id
3   UNION
4   SELECT * FROM students
5   RIGHT JOIN class ON students.class = class.id
```

id	lastname	firstname	IIN	age	class	phone	id	class_name	course	count
2	NULL	NULL	010202000000	22	NULL	NULL	NULL	NULL	NULL	NULL
3	Aryn	Sagadat	011128123456	21	7	8700000000	7	IS-20-3	2	4
4	Baimukhanova	Leyla	020616123456	20	1	8700000000	1	IS-22-1	1	2
5	Ersainov	Asan	020721123456	20	1	8700000000	1	IS-22-1	1	2
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000	6	IS-20-2	2	1
9	Dosymbetova	Maria	020502123456	20	3	87000000000	3	IS-22-3	1	1
16	Kengesova	Zhaina	010200123456	21	7	87001234567	7	IS-20-3	2	4
17	Asylova	Aina	000821123456	21	7	87012345678	7	IS-20-3	2	4
18	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
19	Aryn	Sagadat	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	2	IS-22-2	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	11	IS-21-1	2	0

221 - сурет. UNION операторы арқылы екі кесетден жазбаларды алу

5.5 Ұсыныстар мен уақытша кестелер. Кестелердің көшірмесін жасау

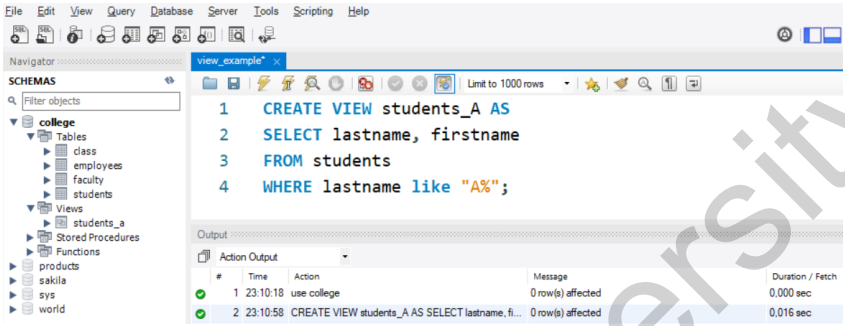
Ұсыныстар – SELECT операторы арқылы алынған жазбаларды сақтайтын (сұраныс нәтижесі) виртуалды кесте. Ағылшынша атауы – Views. Кейде көрініс деп те аударылады.

Ұсыныста құрудың жалпы синтаксисі:

```
CREATE [OR REPLACE] VIEW ұсыныс_атауы AS
SELECT өріс1, өріс2, ...
FROM кесте_атауы
WHERE шарт;
```

[OR REPLACE] кілттік сөзі егер ұсыныс деректер қорында бар болса, құрылған жаңа ұсыныспен алмастырады.

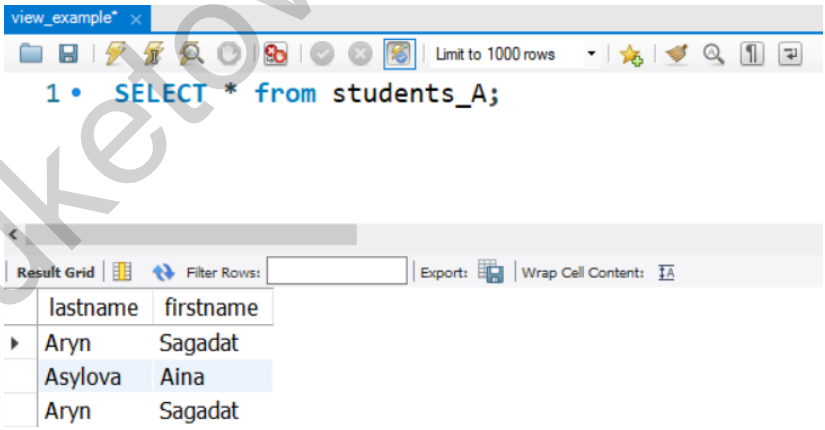
Мысалы, *students* кестесінен тегі А әріпінен басталатын студенттер тізімінен ұсыныс құру



222 - сурет. *students_A* ұсынысын құру

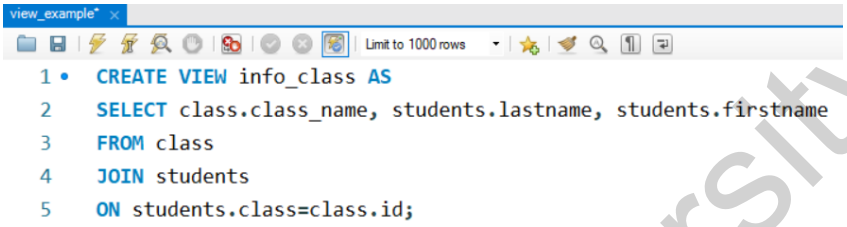
```
CREATE VIEW students_A AS
SELECT lastname, firstname
FROM students
WHERE lastname like "A%";
```

Құрылған ұсыныстан жазбаларды кесте сияқты SELECT операторы арқылы алып отырамыз.



223 - сурет. Ұсыныстан жазбаларды алу

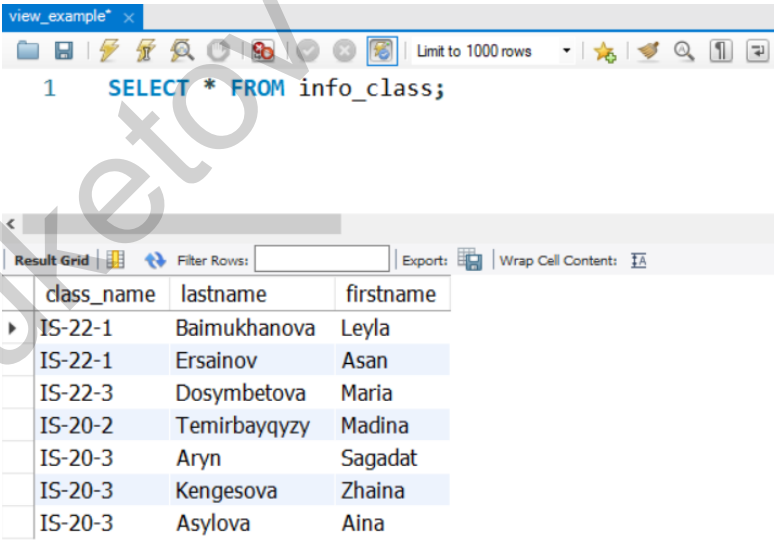
Күрделі сұраныстарды ұсыныс ретінде сақтауға болады. Екі кестенден біріктіру арқылы алынған деректерді пайдаланып ұсыныс құрамыз (223-сурет).



224 - сурет. Ұсыныс құру

```
CREATE VIEW info_class AS
SELECT class.class_name, students.lastname,
students.firstname
FROM class
JOIN students
ON students.class=class.id;
```

Нәтижесі:



class_name	lastname	firstname
IS-22-1	Baimukhanova	Leyla
IS-22-1	Ersainov	Asan
IS-22-3	Dosymbetova	Maria
IS-20-2	Temirbayqyzy	Madina
IS-20-3	Aryn	Sagadat
IS-20-3	Kengesova	Zhaina
IS-20-3	Asylova	Aina

225 - сурет. Ұсыныстан деректерді алу

Ұсынысты өшіру

Ұсынысты өшірудің синтаксисі

```
DROP VIEW  [IF EXISTS] ұсыныс_атауы;
```

Мысалы

```
DROP VIEW students_A;
```

Уақытша кестелер құру

Уақытша кестелер деректер қорынан күрделі сұраныстар арқылы алынған деректерді сақтау үшін қолданылады, мысалы, JOIN және UNION сұраныстарының нәтижесі.

Құрылған уақытша кестелер байланыс (сеанс) жабылғанға дейін қол жетімді болады.

Уақытша кесте құрудың жалпы синтаксисі

```
CREATE TEMPORARY TABLE кесте_атауы (
  өріс_1, өріс_2, ...,
);
```

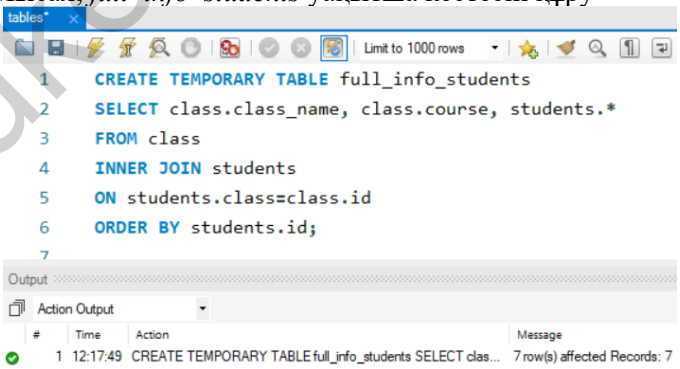
немесе

```
CREATE TEMPORARY TABLE кесте_атауы (
  сұраныс
);
```

Уақытша кестені жою

```
DROP TEMPORARY TABLE кесте_атауы;
```

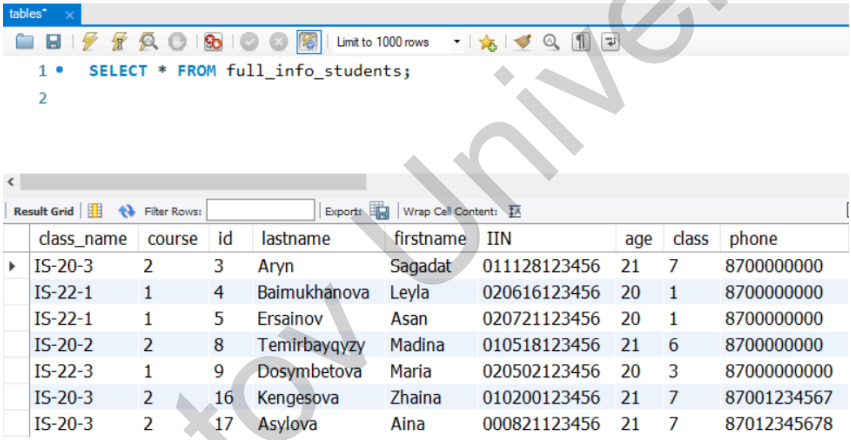
Мысал, *full_info_students* уақытша кестесін құру



226 - сурет. *full_info_students* уақытша кестесін құру

```
CREATE TEMPORARY TABLE full_info_students
SELECT class.class_name, class.course, students.*
FROM class
INNER JOIN students
ON students.class=class.id
ORDER BY students.id;
```

Бұл уақытша кесте students және class кестелірінен JOIN операторы арқылы алынған жазбаларды сақтайды.



class_name	course	id	lastname	firstname	IIN	age	class	phone
IS-20-3	2	3	Aryn	Sagadat	011128123456	21	7	8700000000
IS-22-1	1	4	Baimukhanova	Leyla	020616123456	20	1	8700000000
IS-22-1	1	5	Ersainov	Asan	020721123456	20	1	8700000000
IS-20-2	2	8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
IS-22-3	1	9	Dosymbetova	Maria	020502123456	20	3	87000000000
IS-20-3	2	16	Kengesova	Zhaina	010200123456	21	7	87001234567
IS-20-3	2	17	Asylova	Aina	000821123456	21	7	87012345678

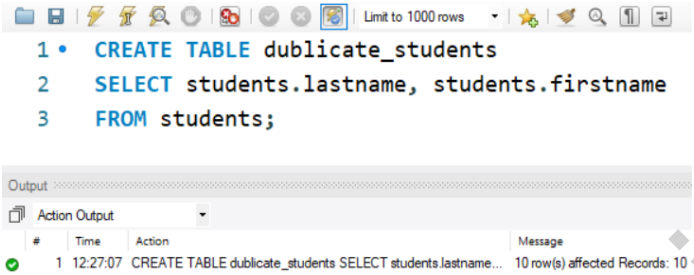
227 - сурет. Уақытша кестеден жазбаларды алу

Кестені көшірмесін жасау

MySQL-де деректер қорында бар кестенің көшірмесін жасау мүмкіндігі бар. Ол үшін CREATE TABLE және SELECT операторларын пайдаланамыз. CREATE TABLE операторынан кейін жаңа кестенің аты жазылады, кесте атауынан кейін көшірілетін кестеден деректер SELECT операторы арқылы алынады.

Жалпы синтаксисі:

```
CREATE TABLE жаңа_кесте_атауы
SELECT өріс1, өріс2, өріс3
FROM кесте_атауы;
```

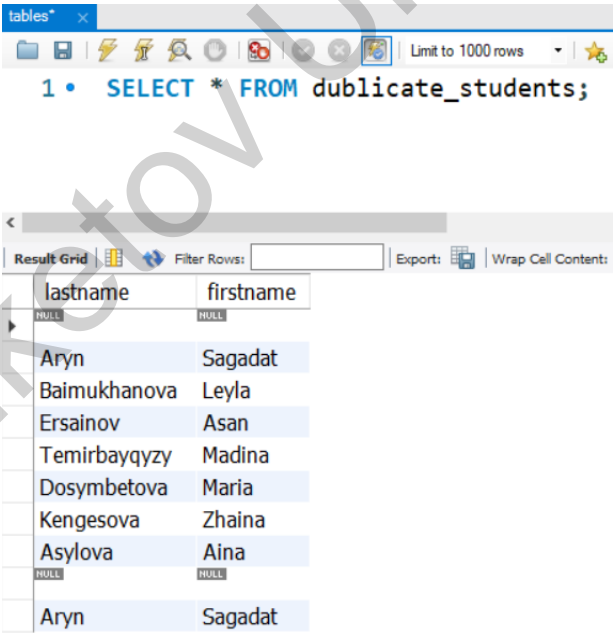


228 - сурет. *dublicate_students* кестесін құру

```
CREATE TABLE dublicate_students
SELECT students.lastname, students.firstname
FROM students;
```

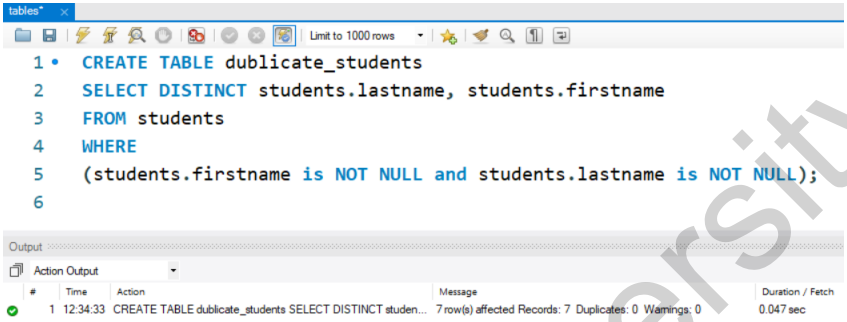
students кестесінен *lastname, firstname* өрістері *dublicate_students* кестесінен көшіріледі.

Нәтижесі:



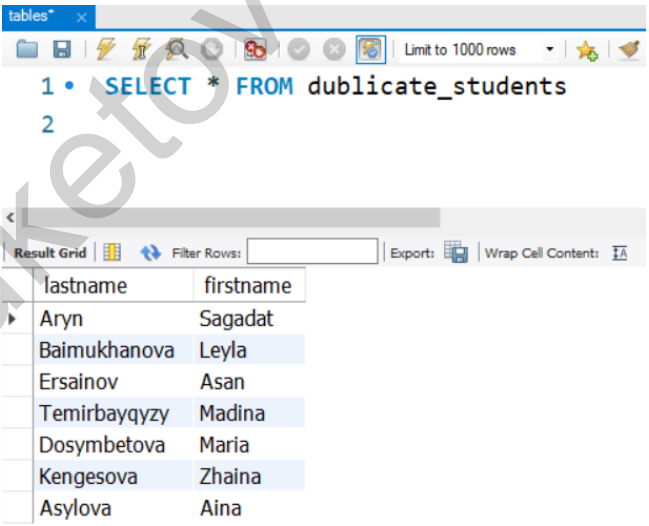
229 - сурет. Жазбаларды алу

students кестесінен қайталанатын және мәндері NULL болатын жазбаларды алып тастап, жаңа кесте құрайық.



230 - сурет. Кестеден жазбаларды көшіріп, жаңа кесте құру

```
CREATE TABLE dublicate_students
SELECT DISTINCT students.lastname, students.firstname
FROM students
WHERE
(students.firstname is NOT NULL and students.lastname is
NOT NULL);
```



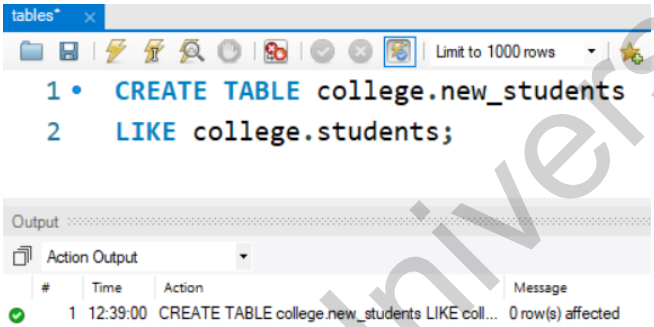
231 - сурет. Кестеден жазбаларды алу

Бір кестенің өрістерін ғана (жазбаларды емес) көшіріп кесте құру үшін CREATE TABLE және LIKE операторлары қолданылады.

Жалпы синтаксисі:

```
CREATE TABLE жаңа_кесте_атауы LIKE кесте_атауы;
```

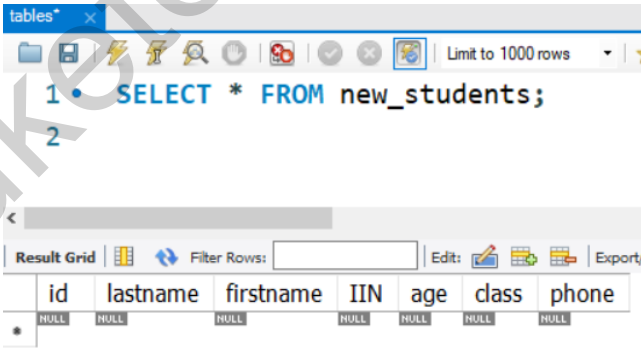
students кестесінің өрістерін көшіріп new_students кестесін құрамыз.



232 - сурет. Бірдей бағандары бар кестелерден деректерді алу

```
CREATE TABLE college.new_students LIKE college.students;
```

Құрылған *new_students* кестесінен жазбаларды алсақ, бос кесте аламыз.

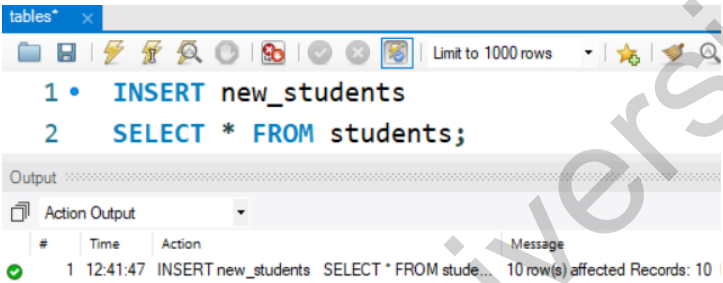


233 – сурет. Кестеден жазбаларды алу

Бір кестенің жазбаларын басқа кестеге қою үшін INSERT және SELECT операторылары қолданылады.

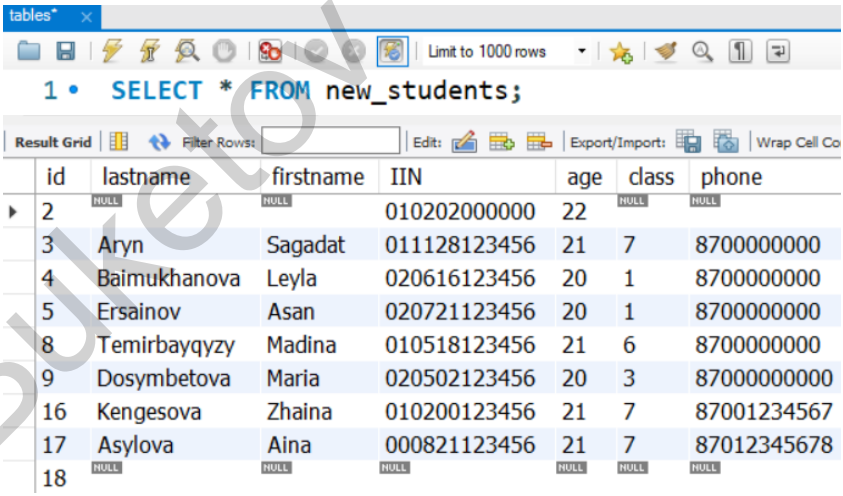
Мысалы, *new_students* кестесіне *students* кестесіндегі жазбаларды қосу.

```
INSERT new_students
SELECT * FROM students;
```



234 - сурет. *new_students* кестесіне *students* кестесіндегі жазбаларды қосу

new_students кестесіндегі жазбаларды алайық.



id	lastname	firstname	IIN	age	class	phone
2	NULL	NULL	010202000000	22	NULL	NULL
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
9	Dosymbetova	Maria	020502123456	20	3	87000000000
16	Kengesova	Zhaina	010200123456	21	7	87001234567
17	Asylova	Aina	000821123456	21	7	87012345678
18	NULL	NULL	NULL	NULL	NULL	NULL

235 - сурет. Кестеден жазбаларды алу

6 тарау. Ішкі программалар

6.1 Айнымалылар

MySQL-де айнымалылардың үш түрі бар:

1) Қолданушы анықтаған айнымалылар
2) Локальді айнымалылар
3) Жүйелік айнымалылар

Қолданушы анықтаған айнымалылар атауы (идентификаторы) @ символынан басталады, мысалы, *@name, @age*.

Қолданушы анықтаған айнымалылардың атауында қаріп регистрі маңызды емес, *@name* мен *@NAME* бір айнымалы болып табылады. Айнымалы атауының ұзындығы 64 символдан аспауы керек.

Айнымалыларды жариялау және инициализациялау үшін SET және SELECT қолданылады:

```
SET @айнымалы_атауы=мән;
SELECT @айнымалы_атауы:=мән;
```

Айнымалыға мән беру үшін = және := меншіктеу операторлары қолданылады.

Мысалы:

```
SET @name='arman';
SELECT @name;
```

Локальді айнымалылар – сақталатын ішкі бағдарлама (процедура мен функция, триггер)

Локальді айнымалыларды жариялау үшін DECLARE кілттік сөзі қолданылады.

Локальді айнымалыларды жариялау

```
DECLARE айнымалы_атауы дерек_типі [DEFAULT мән];
```

Мысалы, процедуда локальді айнымалыларды қолдану

```
DECLARE total_price Oct(8,2) DEFAULT 0.0;
DECLARE a,b,c INT DEFAULT 0;
DELIMITER $$
Create Procedure Test()
```

```
    BEGIN
      DECLARE A INT DEFAULT 100;
      DECLARE B INT;
      DECLARE C INT;
      DECLARE D INT;
      SET B = 90;
      SET C = 45;
      SET D = A + B - C;
      SELECT A, B, C, D;
    END
  $$
```

Жүйелік айнымалылар

Жүйелік айнымалылар – жүйеде алдын ала анықталған мәндері бар айнымалылар.

Жүйелік айнымалылар тізімін алу

```
SHOW VARIABLES;
```

Атауында '%wait_timeout%' тіркесі бар Жүйелік айнымалылар тізімін алу

```
SHOW VARIABLES LIKE '%wait_timeout%';
```

Жүйелік айнымалының мәнін экранға шығару

```
SELECT @@key_buffer_size;
```

6.2 Кірістірілген функциялар

MySQL өте үлкен кірістірілген функциялар жиынын ұсынады. Оларды built-in функциялар деп атайды.

MySQL кірістірілген функцияларын бірнеше топқа бөлуге болады.

- Математикалық функциялар
- Агрегаттық функциялар
- Жол функциялары
- Күн мен уақыт функциялары
- Жүйелік функциялар

12-кесте

Жолдық типтегі мәндерге қолданылатын функциялар тізімі

№	Функция	Сипаттамасы	Мысал
1	ASCII*(string)*	Жолдың бірінші символының ASCII кодын қайтарады	SELECT ASCII("a") AS Code; *Нәтижесі: 65* SELECT ASCII(lastname) AS Code FROMstudents;
2	CHAR_LENGTH (*string*) CHARACTER_LENGTH(*string*)	Жолдағы символдарын санын анықтайды	SELECT CHAR_LENGTH("MySQL") AS Length; *Нәтижесі: 5* SELECT CHARACTER_LENGTH(lastname) AS Length FROM students;
3	CONCAT(*expression1*, *expression2*, *expression3*,...)	Екі немесе бірнеше жолдарды қосады	SELECT CONCAT("My", "SQL") AS Result; *Нәтижесі: MySQL* SELECT CONCAT(lastname, " ", firstname) AS fullname FROM students; *Нәтижесі: студенттің тегі мен атын қосып шығарады, бағанға fullname деп атауы беріледі.*

№	Функция	Сипаттамасы	Мысал
4	CONCAT_WS(*separator*, *expression1*, *expression2*, *expression3*,...)	Екі немесе бірнеше жолдарды қосады. Жолдар арасына separator символы қойылады	SELECT CONCAT_WS("*", "My", "SQL") AS Result; *Нәтижесі: My*SQL* SELECT CONCAT_WS(" ", lastname, firstname) AS fullname FROM students; *Нәтижесі:студенттің тегі мен атының арасына бос орын белгісі қосып шығарылады, бағанға fullname деп атауы беріледі.*
5	INSERT(*string*, *position*, *number*, *string2*)	*Бірінші жолдың (string)position символынан бастап, number символды екінші жолмен (string2) алмастырады*	SELECT INSERT("MSSQL",1, 2, "My"); *Нәтижесі: MySQL* SELECT INSERT("mysql.kz",7, 2,"com"); *Нәтижесі: mysql.com*
6	INSTR(*string1*, *string2*)	*Екінші жолдың бірінші жолдағы кездескен алғашқы орнын (индексін) анықтайды*	SELECT INSTR("MySQL", "S") ASresult; *Нәтижесі: 3*

№	Функция	*Сипаттамасы*	Мысал
7	LCASE(*text*)	*Жолдың символдарын кіші регистрге айналдырады*	SELECT LCASE("MySQL"); *Нәтижесі: mysql*
8	LEFT(*string*, *number_of_chars*)	*Жолдан (*string) алғашқы number_of_chars символды ғана алады.	SELECT LEFT("MySQL 8.0", 5) AS result; *Нәтижесі: MySQL* SELECT LEFT(CustomerName, 5) ASresult FROM Customers;
9	LENGTH(*string*)	*Жолдың ұзындығы (символдар саны)*	SELECT LENGTH("MySQL") ASresult; *Нәтижесі: 5*
10	LOCATE(*substring*, *string*, *start*)	*Returns the position of the first occurrence of a substring in a string*	SELECT LOCATE("y", "MySQL") ASresult; *Нәтижесі: 2* SELECT LOCATE("SQL", "MySQL", 3) ASresult; *Нәтижесі: 3*
11	LOWER(*text*)	*Жолдың символдарын кіші регистрге айналдырады*	SELECT LOWER("MySQL "); *Нәтижесі: mysql* SELECT LOWER(CustomerName) ASresult FROM Customers;
№	Функция	*Сипаттамасы*	Мысал

12	LPAD(*string*, *length*, *lpad_string*)	*Жолдың ұзындығы* length орнатады, егер *Жолдың нақты ұзындығы* length кіші болса, жолдың сол жағынан lpad_string жолымен толтырады	SELECT LPAD("MySQL", 10, "*"); *Нәтижесі:* *****MySQL* SELECT LPAD(CustomerName, 30, "") AS result FROM Customers;
13	LTRIM(*string*)	*Жолдың сол жағыннан бос орындарды алып тастайды*	SELECT LTRIM(" MySQL ") ASresult; *Нәтижесі: MySQL*
14	MID(*string*, *start*, *length*)	*Жолдан* start символынан бастап, length символды алады	SELECT MID("MySQL", 3, 3) AS result; *Нәтижесі: SQL* SELECT MID(CustomerName, 2, 5) ASresult FROM Customers;
15	POSITION(*substring* IN *string*)	*Символдың немесе тіркестің берілген жолда нешінші индексте тұрғанын анықтайды*	SELECT POSITION("y"IN"MySQL") AS result; *Нәтижесі: 2* SELECT POSITION ("a"IN CustomerName) FROM Customers;

№	Функция	*Сипаттамасы*	Мысал
16	REPEAT(*string*, *number*)	*String жолды number рет қайталап жазады*	SELECT REPEAT("MySQL", 3); *Нәтижесі: MySQLMySQLMySQL* SELECT REPEAT(CustomerName, 2) FROM Customers;
17	REPLACE(*string*, *substring*, *new_string*)	*string жолдағы substring тіркесін new_string тіркесімен алмастырады.* Қаріптің регистрі ескеріледі.	SELECT REPLACE("MS SQL", "MS ", "My"); *Нәтижесі: MySQL*
18	REVERSE(*string*)	Жолды кері қайтарады	SELECT REVERSE("LQS "); *Нәтижесі: SQL* SELECT REVERSE(CustomerName) FROM Customers;
19	RIGHT(*string*, *number_of_chars*)	Жолдың сол жағынан number_of_chars символды алады	SELECT RIGHT("MySQL Workbench", 9) AS result; *Нәтижесі: Workbench* SELECT RIGHT(CustomerName, 5) ASresult FROM Customers;

№	Функция	Сипаттамасы	Мысал
20	RPAD(*string*, *length*, *rpad_string*)	Жолдың ұзындығы *length орнатады, егер* Жолдың нақты ұзындығы *length кіші болса, жолдың оң жағынан rpad_string жолымен толтырады*	SELECT RPAD("SQL", 10, "*"); *Нәтижесі: SQL*******
21	RTRIM(*string*)	Жолдың оң жағынан бастап бірден көп болған бос орындарды жояды	SELECT RTRIM("SQL ") AS result; *Нәтижесі: SQL*
22	STRCMP(*string1*, *string2*)	Екі жолды салыстырады. Екі жол бірдей болса 0 мәні, әр түрлі болса -1 мәні алынады.	SELECT STRCMP("SQL", "SQL"); *Нәтижесі: 0*
23	SUBSTR(*string*, *start*, *length*)	*string жолдан start басман, length символды бөліп алады*	SELECT SUBSTR("querySQL", 6, 3) AS result; *Нәтижесі: SQL*
24	SUBSTRING(*string*, *start*, *length*)	*string жолдан start басман, length символды бөліп алады*	SELECT SUBSTRING(CustomerName, 2, 5) AS ExtractString FROM Customers;
25	SUBSTRING_INDEX(*string*, *delimiter*, *number*)	Жолдан delimiter символына дейінгі жолды бөліп алады. *Number неше жол тіркесі бөлінуі керек екен көрсетеді*	SELECT SUBSTRING_INDEX("www.mysql.com", ".", 1); *Нәтижесі: www* SELECTSUBSTRING_INDEX("www.mysql.com", ".", 2); *Нәтижесі: www.mysql*

№	Функция	Сипаттамасы	Мысал
26	TRIM(*string*)	Жолдың басындағы және соңындағы бос орындарды жояды	SELECT TRIM(' SQL ') AS result; *Нәтижесі: SQL*
27	UCASE(*text*)	Қаріпті үлкен регистрге айналдырады	SELECT UCASE("sql"); *Нәтижесі: SQL*
28	UPPER(*text*)	Қаріпті үлкен регистрге айналдырады	SELECT UPPER("sql"); *Нәтижесі: SQL*

13-кесте

Сандық типтегі мәндерге қолданылатын функциялар тізімі

№	Функция	Сипаттамасы	Мысал
1	ABS(number)	Абсолют мәнді қайтарады (санның модулі)	SELECT ABS(-43); *Нәтижесі: 43*
2	ACOS(number)	Санның арккосинусын есептейді	SELECT ACOS(0.37); *Нәтижесі: 1.191787*
3	ASIN(number)	Санның арксинусын есептейді	SELECT ASIN(-0.7); *Нәтижесі: -0.77539749661753*
4	ATAN(number)	Санның арктангенсін есептейді	SELECT ATAN(2.5);
5	AVG(expression)	Орташа мәнді есептейді	SELECT AVG(Price) FROM smartphones;
6	CEIL(number) CEILING(number)	number-ден үлкен бүтін санды қайтарады	SELECT CEIL(78.65); *Нәтижесі: 79* SELECT CEILING(78.45); *Нәтижесі: 79*

№	Функция	Сипаттамасы	Мысал
7	COS(number)	Косинусты (cos) есептейді	SELECT COS(2); *Нәтижесі: -0.4161468365*
8	COT(number)	Котангенсті (ctg) есептейді	SELECT COT(7); *Нәтижесі: 1.1475154224*
9	COUNT(expression)	Сұранысқа байланысты жазбалар санын есептейді	SELECT COUNT(lastname) FROM students;
10	x DIV y	Бүтін бөлуді есептейді	SELECT 10 DIV 3; *Нәтижесі: 3*
11	EXP(number)	e санының дәрежесін есептейді e (2.718281…)	SELECT EXP(1); *Нәтижесі: 2.718281*
12	FLOOR(number)	Санның бүтін бөлігін қайтарады	SELECT FLOOR(29.84); *Нәтижесі: 29*
13	GREATEST(arg1, arg2, arg3, …)	Тізімдегі ең үлкен мәнді қайтарады	SELECT GREATEST(13,7,47); *Нәтижесі: 47*
14	LEAST(arg1, arg2, arg3, …)	Тізімдегі ең кіші мәнді қайтарады	SELECT LEAST(13, 7, 47); *Нәтижесі: 7*
15	LN(number)	Натурал логарифмді есептейді	SELECT LN(2); *Нәтижесі: 0.69314718*
16	LOG(number) LOG(base, number)	Логарифмді есептейді	SELECT LOG(2); *Нәтижесі: 0.69314718* SELECT LOG(3, 9); *Нәтижесі: 2*

№	Функция	Сипаттамасы	Мысал
17	LOG10(number)	Ондық логарифмді есептейді	SELECT LOG10(10); *Нәтижесі: 1*
18	MAX(expression)	Мәндер жиымынан үлкен мәнді қайтарады	SELECT MAX(Price) AS MaxPrice FROM smartphones; *Нәтижесі: smartphones кестесіндегі Price өрісіндегі үлкен мәнді анықтайды*
19	MIN(expression)	Мәндер жиымынан кіші мәнді қайтарады	SELECT MIN(Price) AS MinPrice FROM smartphones; *Нәтижесі: smartphones кестесіндегі Price өрісіндегі кіші мәнді анықтайды*
20	MOD(x, y) x MOD y x % y	Қалдықты есептейді	SELECT MOD(18, 4); *Нәтижесі: 2* SELECT 18 % 4; *Нәтижесі: 2*
21	PI()	π санын қайтарады	SELECT PI(); *Нәтижесі: 3.141593*
22	POW(x, y)	хсанының удәрежесін есептейді	SELECT POW(3, 3); *Нәтижесі: 27*
23	POWER(x, y)	хсанының удәрежесін есептейді	SELECT POWER(4, 2); *Нәтижесі: 16*

№	Функция	Сипаттамасы	Мысал
24	RADIANS(number)	Бұрышты радианға айналдырады	SELECT RADIANS(180); *Нәтижесі: 3.141592653589793*
25	RAND(seed)	0 мен 1 арасындағы кездейсоқ нақты санды қайтарады	SELECT RAND();
26	ROUND(number, decimals)	Санды дөңгелейтейді	SELECT ROUND(135.375, 2); *Нәтижесі:135.37* SELECT ProductName, Price, ROUND(Price, 1) AS RoundedPrice FROM smartphones;
27	SIGN(number)	Санның белгісін қайтарады Егер number > 0 болса 1, егер number = 0 болса 0, егер number < 0 болса -1 қайтарады	SELECT SIGN(255.5); *Нәтижесі:1*
28	SIN(number)	Санның синусын есептейді	SELECT SIN(90); *Нәтижесі: 0.8939966636005579*
29	SQRT(number)	Квадратты түбірді есептейді	SELECT SQRT(64); *Нәтижесі: 8*
30	SUM(expression)	Қосындыны есептейді	SELECT SUM(Price) AS Total FROM smartphones;
31	TAN(number)	Санның тангенсін есептейді	SELECT TAN(90); *Нәтижесі: -1.995200412208242*

14-кесте

Уақыт үшін қолданылатын функциялар

№	Функция	Сипаттамасы	Мысал
1	ADDDATE(*date*, INTERVAL *value addunit*) ADDDATE(*date*, *days*)	Көрсетілген күнге күнді қосады	SELECT ADDDATE("2023-02-19", INTERVAL 10 DAY); *Нәтижесі: 2023-03-01* SELECT ADDDATE("2022-05-21 09:34:21", INTERVAL -3 HOUR); *Нәтижесі: 2022-05-21 06:34:21*
2	ADDTIME(*datetime*, *addtime*)	Көрсетілген күнге уақытты қосады	SELECT ADDTIME("2022-04-05 08:10:21", "2:10:5"); *Нәтижесі: 2022-04-05 10:20:26*
3	CURRENT_DATE()	Ағымдағы күнді (датаны) қайтарады	SELECT CURRENT_DATE();
4	CURRENT_TIME()	Ағымдағы уақытты қайтарады	SELECT CURRENT_TIME();
5	CURRENT_TIMESTAMP()	Ағымдағы күн мен уақытты қайтарады	SELECT CURRENT_TIMESTAMP();
6	CURTIME()	Ағымдағы уақытты қайтарады	SELECT CURTIME();

№	Функция	Сипаттамасы	Мысал
7	DATE(*expression*)	Уақыт көрсетілген мәтіннен күнді бөліп алады	SELECT DATE("2017-06-15 09:34:21"); *Нәтижесі: 2017-06-15*
8	DATEDIFF(*date1*, *date2*)	Екі уақыттың арасындағы айырмашылықты есептейді	SELECTDATEDIFF("2023-02-05", "2023-01-29"); *Нәтижесі: 7*
9	DATE_FORMAT (*date*, *format*)	Уақытты көрсетілген формат бойынша форматтайды	SELECT DATE_FORMAT("2022-02-28", "%Y"); *Нәтижесі: 2022*
10	NOW()	Ағымдағы күн мен уақытты қайтарады	SELECT NOW();

6.3 Сақталатын процедуралар мен функциялар

Сақталатын бағдарламалар (процедуралар мен функцияларға) MySQL-дің 5.0 нұсқасынан бастап енгізілді. Сақталған процедуралар – серверде сақтауға болатын SQL өрнектерінің жиынтығы. Клиентке сұранысты қайта жіберудің қажеті жоқ, тек сақталған бағдарламаны шақыру қажет.

Сақталатын процедуралар – бұл серверде құрастыруға және сақтауға болатын SQL командаларының жиынтығы. Осылайша, жиі қолданылатын сұранысты сақтаудың орнына, клиенттер тиісті сақталған процедураға сілтеме жасайды.

SQL сұраныстары (;) таңбасымен аяқталады. Бірақ бұл MySQL сақталған процедуралары мен функцияларда қателіктер туғызады. Өйткені оның көптеген операторлары болуы мүмкін және әрқайсысы нүктелі үтірмен (;) аяқталуы керек. Сақталған процедуралар мен функцияларды SQL сұраныстарынан бөліп көрсету үшін DELIMITER $$ кілттік сөзі жазылады.

DELIMITER $$ кілттік сөзінен кейін процедура не функцияның сипаттамасы жазылады және $$ символымен аяқталады.

Мысал, процедураны сипаттау

```
USE database_name;

DELIMITER $$
CREATE PROCEDURE procedure_name()
BEGIN
        SELECT * FROM table_name;
END;
$$
```

Сақталатын процедуралар

Сақталатын процедураның жалпы жазылу синтаксисі:

```
CREATE PROCEDURE процедура_атауы
[параметрлер]
BEGIN
declaration_section
executable_section
END;
```

– Процедура бірнеше параметрді қабылдай алады. Параметрлердің 3 типі бар:

IN – параметр процедураға сілтеме жасай алады. Параметр мәнін процедура арқылы өзгерту мүмкін емес.

OUT – параметр процедураға сілтеме жасай алмайды, бірақ параметрдің мәні процедура арқылы өзгеруі мүмкін.

IN OUT – параметр процедураға сілтеме жасай алады және параметрдің мәні процедура арқылы өзгеруі мүмкін.

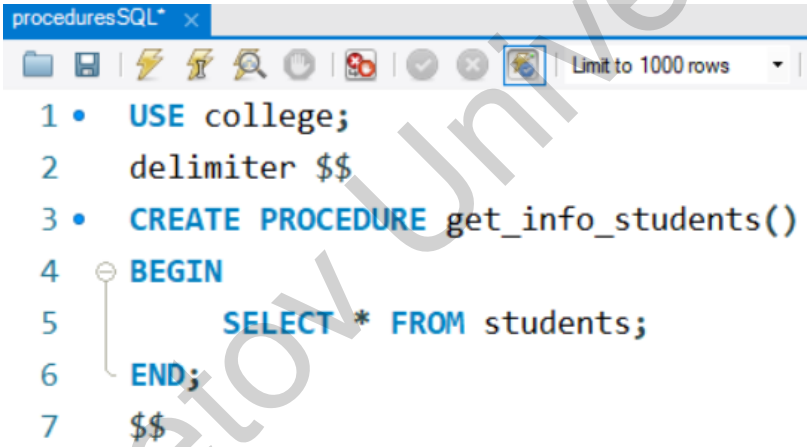
– executable_section – орындалатын операторлар;

– declaration_section – процедурадағы локальді айнымалыларды жариялайтын бөлігі.

Мысал 6.2.1

college деректер қорында параметрсіз *get_info_students()* процедурасын құру. Бұл процедура *students* кестесінен барлық өрістер мен жазбаларды шығарады.

```
USE college;
DELIMITER $$
CREATE PROCEDURE get_info_students()
BEGIN
    SELECT * FROM students;
END;
$$
```

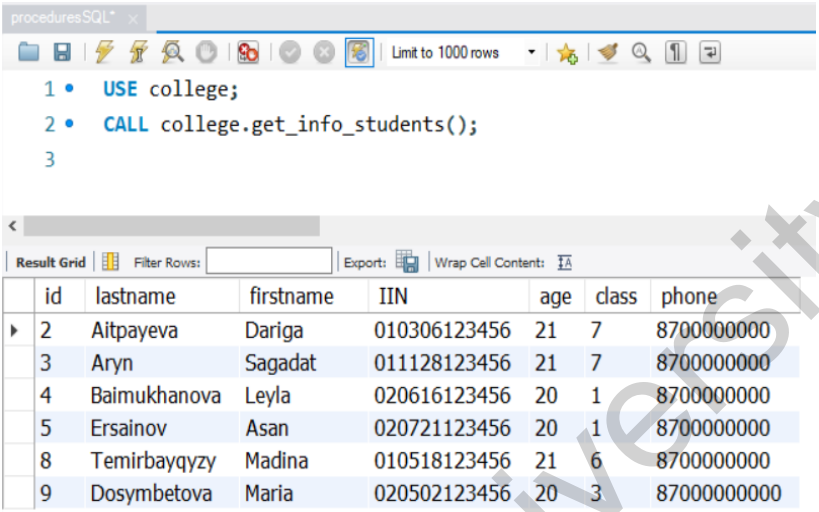


236 - сурет. Параметрсіз процедура құру

Процедураны орындал үшін, оны CALL операторы арқылы шақырады.

Құрылған *get_info_students()* процедурасын шақыру

```
CALL college.get_info_students();
```



id	lastname	firstname	IIN	age	class	phone
2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
9	Dosymbetova	Maria	020502123456	20	3	87000000000

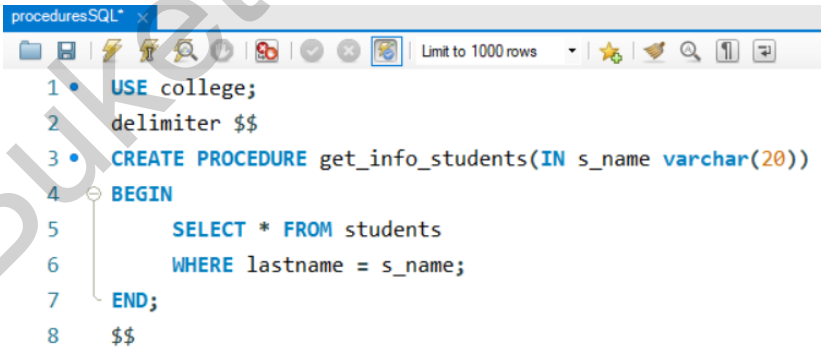
237 - сурет. Процедураны шақыру

Параметрмен процедура құру

Мысал 6.2.2

*get_info_students (*IN s_name VARCHAR(20)*)* процедурасында бір ғана VARCHAR(20) типті *s_name* параметрі бар. Бұл параметр жолдық (VARCHAR) мәндерді қабылдайды.

Құрылған процедура students кестесінен тегі *s_name*-ге тең болатын жазбаларды шығарады.

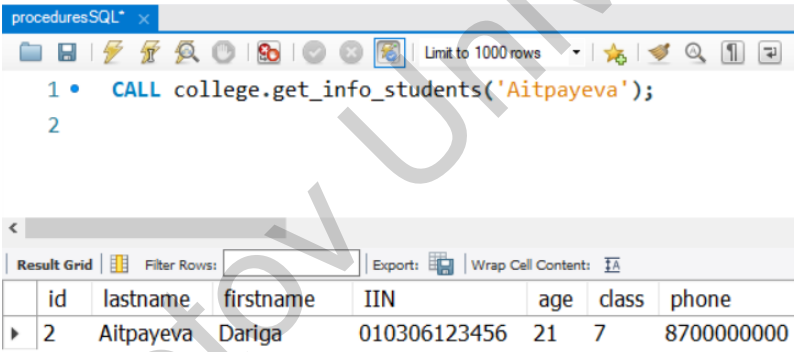


238 - сурет. IN параметрмен процедура құру

```
USE college;
DELIMITER $$
CREATE PROCEDURE get_info_students(IN s_name varchar(20))
BEGIN
    SELECT * FROM students
   WHERE lastname = s_name;
END;
$$
```

get_info_students(IN s_name VARCHAR(20)) функциясын шақыру. *s_name* параметріне 'Aitpayeva' мәні беріледі.

```
CALL college.get_info_students('Aitpayeva');
```



239 - сурет. Процедураны шақыру

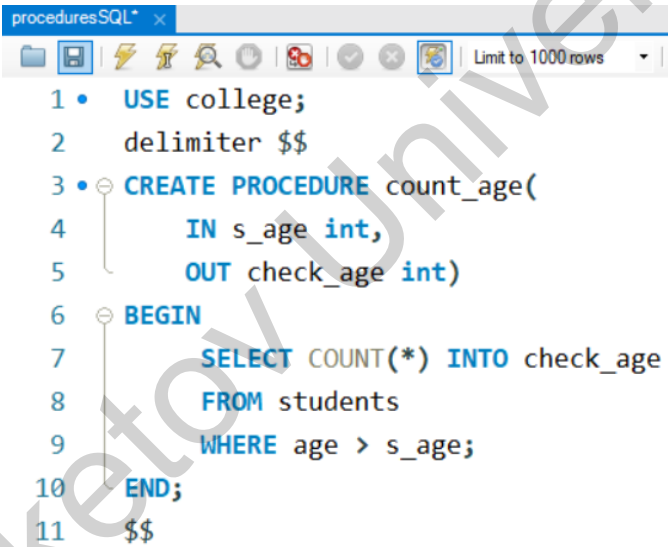
OUT параметрмен процедура құру

Мысал 6.2.3

count_age(IN s_age int, OUT check_age int) процедурасын құрайық. Бұл процедура жасы *s_age*-ден үлкен студенттердің санын есептейді. Процедура шақырғанда, *s_age* параметріне аргумент (сандық мән) беріледі. Процедура нәтижені *check_age* параметріне INTO кілттік сөзі арқылы береді.

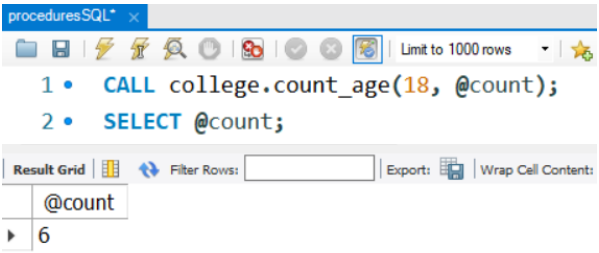
```
USE college;
DELIMITER $$
```

```
CREATE PROCEDURE count_age
(
    IN s_age int,
   OUT check_age int
)
BEGIN
     SELECT COUNT(*) INTO check_age
    FROM students
    WHERE age > s_age;
END;
$$
```



240 - сурет. count_age(IN s_age int, OUT check_age int) процедурасын құру

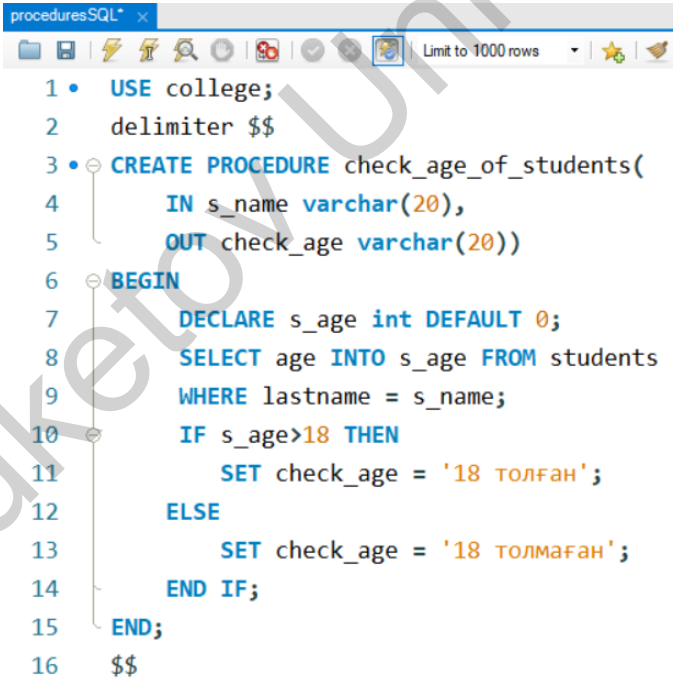
Процедураны шақырамыз. *s_age* айнымалысына 18 мәні беріледі. Процедура students кестесінен жасы *s_age* –ден үлкен студенттердің санын анықтайды және оны *check_age* айнымалысына меншіктейді. Ал процедурадағы *check_age* айнымалысының мәні *count* айнымалысында беріледі. count айнымалысының мәнін SELECT арқылы экранға шығарамыз.



241 - сурет. count_age(IN s_age int, OUT check_age int) процедурасын шақыру

Мысал 6.2.4

check_age_of_students процедурасын құрамыз. *students* кестесінде тегі s_name болатын студенттің жасы 18 жасқа толғанын тексереді. Нәтижені *check_age* параметріне береді.



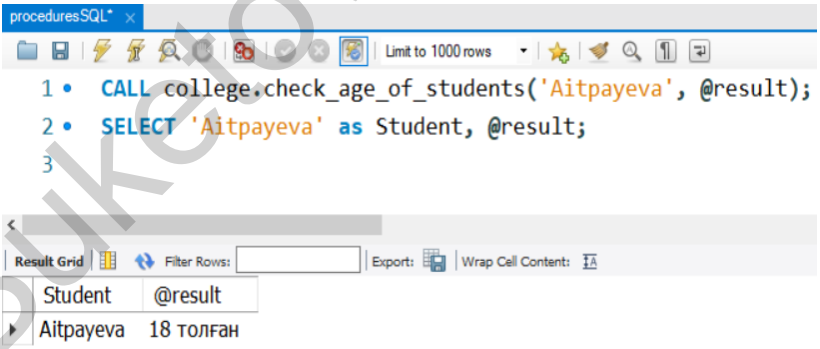
242 - сурет. check_age_of_students() процедурасын құру

```
USE college;
DELIMITER $$
CREATE PROCEDURE check_age_of_students(
    IN s_name varchar(20),
   OUT check_age varchar(20))
BEGIN
     DECLARE s_age int DEFAULT 0;
    SELECT age INTO s_age FROM students
    WHERE lastname = s_name;
    IF s_age>18 THEN
            SET check_age = '18 толған';
    ELSE
            SET check_age = '18 толмаған';
    END IF;
END;
$$
```

Процедураны шақыру

```
CALL college.check_age_of_students('Aitpayeva', @result);

SELECT 'Aitpayeva' as Student, @result;
```

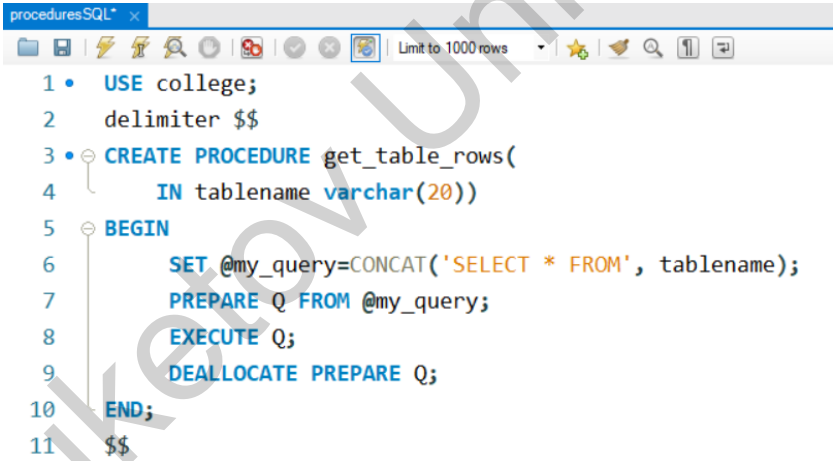


243 - сурет. Процедураны шақыру және нәтижені шығару

Мысал 6.2.5

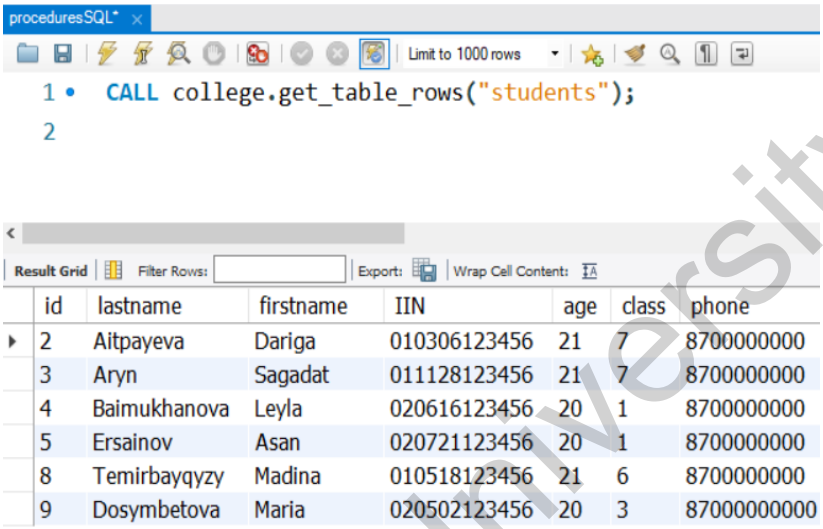
Процедураға параметр ретінде кесте атауын беру. Процедура арқылы көрсетілген кестедегі жазбаларды алу.

```
USE college;
DELIMITER $$
CREATE PROCEDURE get_table_rows(
   IN tablename varchar(20))
BEGIN
    SET @my_query=CONCAT('SELECT * FROM', tablename);
   PREPARE Q FROM @my_query;
   EXECUTE Q;
   DEALLOCATE PREPARE Q;
END;
$$
```



244 - сурет. get_table_rows(IN tablename varchar(20)) процедурасын құру

Процедураны шақыру



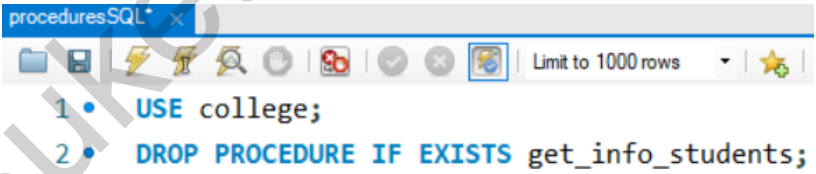
id	lastname	firstname	IIN	age	class	phone
2	Aitpayeva	Dariga	010306123456	21	7	8700000000
3	Aryn	Sagadat	011128123456	21	7	8700000000
4	Baimukhanova	Leyla	020616123456	20	1	8700000000
5	Ersainov	Asan	020721123456	20	1	8700000000
8	Temirbayqyzy	Madina	010518123456	21	6	8700000000
9	Dosymbetova	Maria	020502123456	20	3	87000000000

245 - сурет. Процедураны шақыру

Процедураны өшіру

Процедураны өшіру үшін DROP PROCEDURE операторы қолданылады.

```
DROP PROCEDURE IF EXISTS get_info_students;
```



246 - сурет. Процедураны өшіру

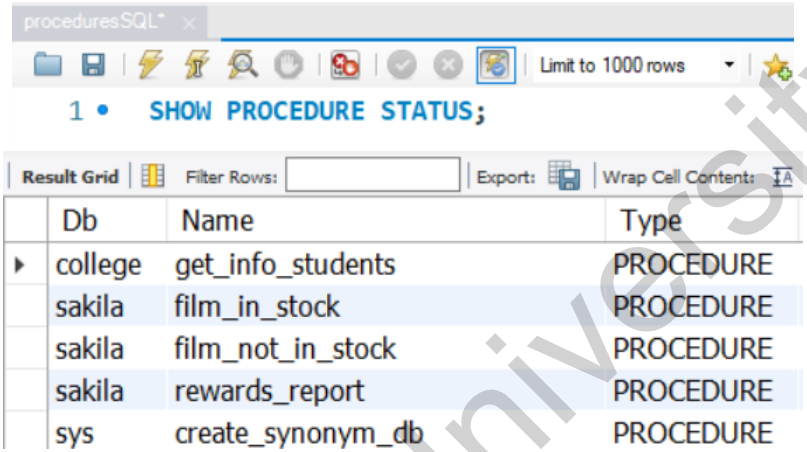
Барлық процедуралар тізімін алу

Жалпы синтаксисі

```
SHOW PROCEDURE STATUS [LIKE 'pattern' | WHERE search_condition]
```

Сервердегі барлық деректер қорындағы процедуралар тізімін алу.

```
SHOW PROCEDURE STATUS;
```

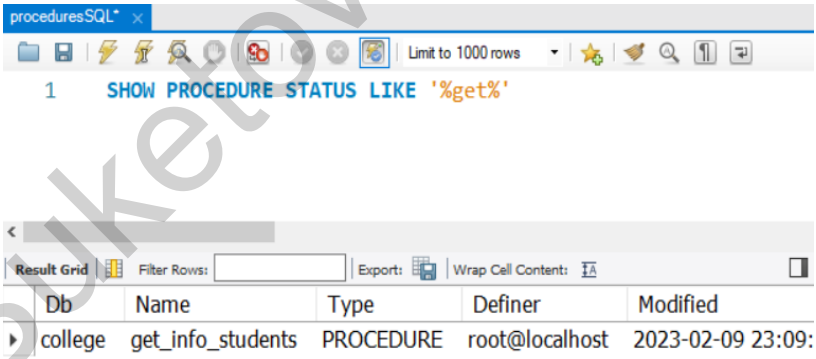


Db	Name	Type
college	get_info_students	PROCEDURE
sakila	film_in_stock	PROCEDURE
sakila	film_not_in_stock	PROCEDURE
sakila	rewards_report	PROCEDURE
sys	create_synonym_db	PROCEDURE

247 - сурет. Сервердегі барлық процедуралар тізімін алу

Атауында *get* тіркесі бар процедуралар тізімін алу

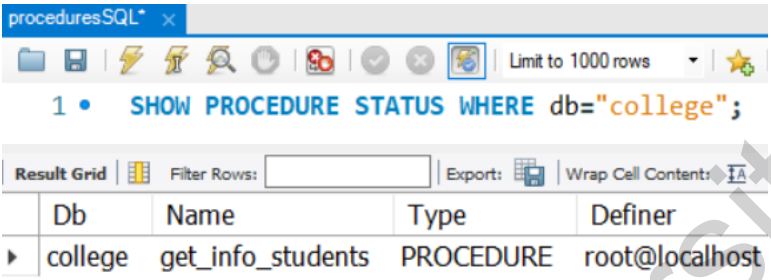
```
SHOW PROCEDURE STATUS LIKE '%get%'
```



Db	Name	Type	Definer	Modified
college	get_info_students	PROCEDURE	root@localhost	2023-02-09 23:09:

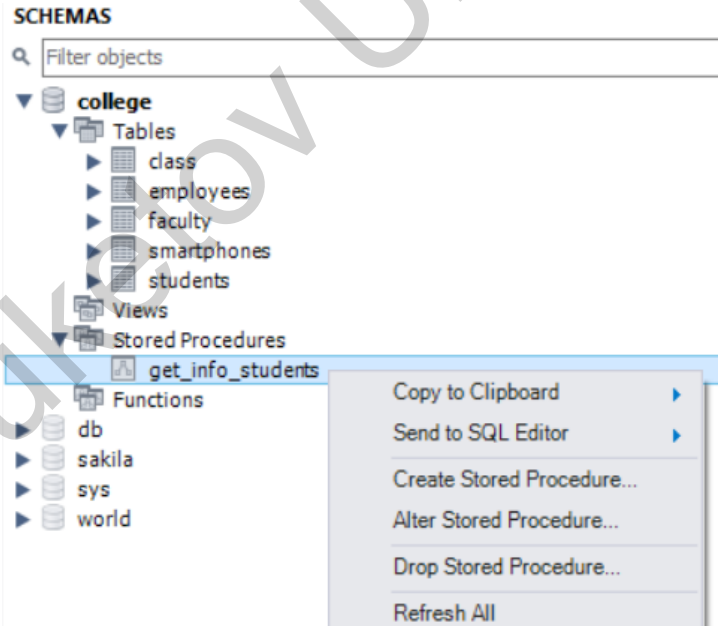
248 - сурет. Атауында *get* тіркесі бар процедуралар тізімі

college деректер қорындағы процедуралар тізімін алу



249 - сурет. *college* деректер қорындағы процедуралар тізімін алу

Деректер қорындағы процедураларды Навигация-SCHEMAS тақтасынан көре аламыз. SCHEMAS тақтасы арқылы процедураларды басқара аламыз: жаңа процедура құру, өзгерту, өшіру, орындауға жіберу.



250 - сурет. Навигация-SCHEMAS тақтасы

Сақталатын функциялар

MySQL-де функция - бұл параметрлерді беруге және мәнді қайтаруға болатын сақталған бағдарлама.

Жалпы синтаксисі

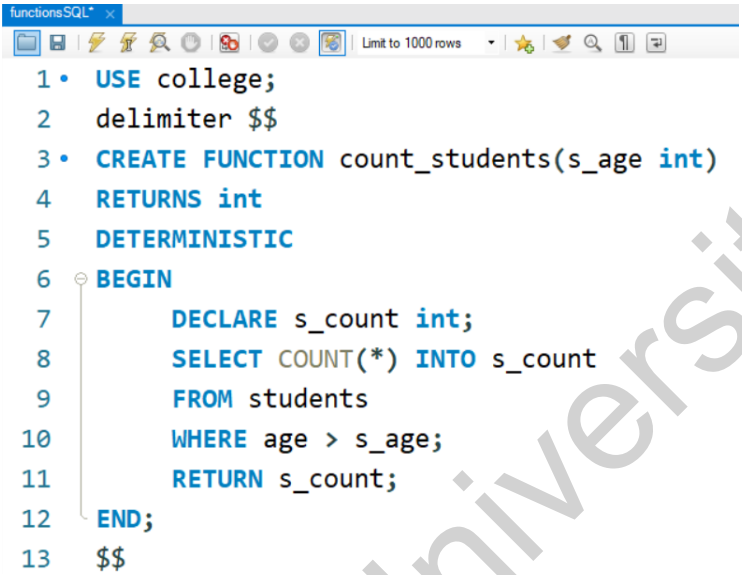
```
CREATE FUNCTION функция_атауы
[ (параметр дерек_типі ]
RETURNS қайтаратын_ дерек_типі
[NOT] DETERMINISTIC
BEGIN
    айнымалыларды_жариялау
    операторлар
END;
```

Параметры немесе аргументтер:

1) функция_атауы – сақталатын функция атауы
2) параметр – фукнцияға берілетін мәндер. Функцияға берілетін параметрлер IN параметрлер болып табылады.
3) қайтаратын_ дерек_типі – функция қайтаратын мәннің типі
4) айнымалыларды_жариялау – локальді айнымалылар жариялау
5) операторлар – функцияда орындалатын операторлар

Мысал 6.2.6

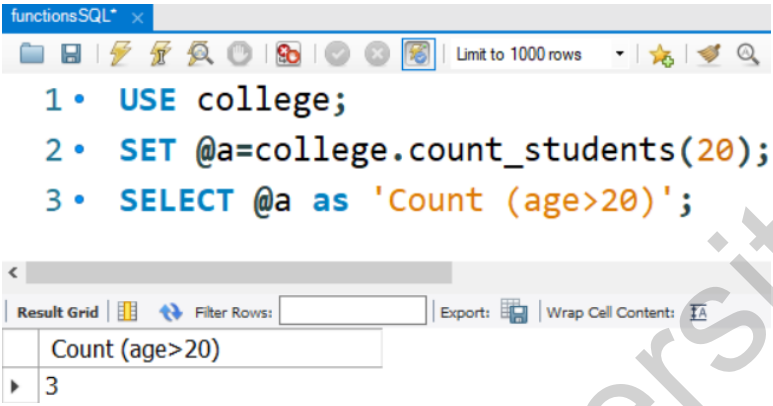
count_students(s_age int) функциясын құру – бұл функция *students* кестесінен жасы *s_age* үлкен студенттердің санын қайтарады.



251 - сурет. *count_students(s_age int)* функциясын құру

```
USE college;
DELIMITER $$
CREATE FUNCTION count_students(s_age int)
RETURNS int
DETERMINISTIC
BEGIN
    DECLARE s_count int;
    SELECT COUNT(*) INTO s_count
   FROM students
   WHERE age > s_age;
   RETURN s_count;
END;
$$
```

count_students(s_age int) функцияны шақыру. *s* мәні *age* параметріне 20 санын береміз. Функцияның нәтижесін *a* айнымалысына меншіктейміз. *a* айнымалысының мәнін SELECT операторы арқылы шығарамыз.



252 - сурет. college.count_students(s_age int) функциясын шақыру

```
USE college;
SET @a=college.count_students(20);
SELECT @a as 'Count (age>20)';
```

Мысал 6.2.7

Calculate(price int, discount int) функциясын құру. Бұл функция тауардың құнын жеңілдікпен есептейді.

```
USE college;
DELIMITER $$
CREATE FUNCTION Calculate(price int, discount int)
RETURNS float
DETERMINISTIC
BEGIN
        DECLARE final_price float;
        SET final_price=price - price/100*discount;
   RETURN final_price;
END;
$$
```

price – тауар құны
discount – жеңілдік (пайызбен)
final_price – жеңілдікпен есептелген тауар құны

```sql
USE college;
delimiter $$
CREATE FUNCTION Calculate(price int, discount int)
RETURNS float
DETERMINISTIC
BEGIN
    DECLARE final_price float;
    SET final_price=price - price/100*discount;
    RETURN final_price;
END;
$$
```

253 - сурет. *Calculate(price int, discount int)* функциясын құру

Функцияны шақыру

```sql
USE college;
SELECT
    Id, ProductName, Price,
    Calculate(Price, 20) as 'Discount price'
FROM smartphones;
```

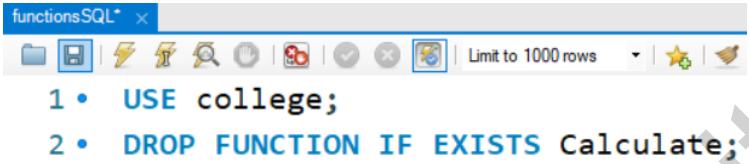
```sql
USE college;
SELECT
    Id, ProductName, Price,
    Calculate(Price, 20) as 'Discount price'
FROM smartphones;
```

Id	ProductName	Price	Discount price
1	iPhone X	76000	60800
2	iPhone 8	51000	40800
3	Galaxy S9	56000	44800
4	Galaxy S8	41000	32800
5	P20 Pro	36000	28800
6	iPhone X	102000	81600
7	Galaxy S9	59000	47200
8	iPhone X	102500	82000
9	Galaxy S8	45000	36000
10	iPhone 8	60500	48400

254 - сурет. Функцияны шақыру

Функцияны өшіру

```
DROP FUNCTION [IF EXISTS] функция_атауы;
```



255 - сурет. Функцияны өшіру

6.4 Триггерлер

MySQL жүйесіндегі триггер – INSERT, DELETE немесе UPDATE операторы орындалуына дейін және кейін автоматты түрде шақырылатын пайдаланушы анықтайтын SQL оператор (лар). Триггер коды кестемен байланысты және кесте жойылған кезде жойылады. Триггер MySQL жүйесінің 5.0.2 нұсқасынан бастап қолданылды.

Триггердің жалпы жазылу синтаксисі

```
DELIMITER $$
CREATE TRIGGER [триггер_атауы]
[триггер уақыты] [триггер оқиғасы/әдісі]
ON [кесте_атауы]
FOR EACH ROW
    [операторлар]
$$;
```

Триггерлерді басқа SQL сұраныстарынан бөлу үшін DELIMITER $$ кілттік сөзі қолданылады. DELIMITER $$ сөзінен кейін триггер сипатталады. Триггердің сипатталуы $$ белгісімен аяқталады.

[триггер уақыты]: триггердің орындалу уақыты. Before – дейін, After – кейін

[триггер оқиғасы/әдісі]: INSERT, UPDATE және DELETE.

[кесте_атауы]: кесте атауы, кез келген триггер кестемен байланысты болуы керек.

FOR EACH ROW: кестенің әр жолына триггерді қолдану үшін қолданылады.

[операторлар]: триггер блогы. Триггер блогы BEGIN басталып, END кілттік сөзімен аяқталады.

BEFORE INSERT триггерін құру

BEFORE INSERT - кестеге жазба қосатын INSERT операторына дейін орындалады.

Мысал 6.3.1

smartphones кестесі үшін BEFORE INSERT триггерін құру.

Триггер атауы: *check_price*

Бұл триггер *smartphones* кестесіне жаңа тауарды енгізуден бұрын орындалады. Егер тауардың бағасын *price<=0* енгізсе, қателік (SQLSTATE '45000') туындайды. *'Бағаны дұрыс енгізіңіз'* хабарламасы шығарылады.

```
USE products;
DELIMITER $$
CREATE TRIGGER check_price
BEFORE INSERT
ON smartphones
FOR EACH ROW
        IF NEW.price<=0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Бағаныдұрысенгізіңіз';
        END IF
$$
```

```
before_insert_trigger*
Limit to 1000 rows
1  USE products;
2  DELIMITER $$
3  CREATE TRIGGER check_price
4  BEFORE INSERT
5  ON smartphones
6  FOR EACH ROW
7  IF NEW.price<=0 THEN
8  SIGNAL SQLSTATE '45000'
9  SET MESSAGE_TEXT = 'Бағаны дұрыс енгізіңіз';
10 END IF
11 $$
```

256 - сурет. *check_price* (BEFORE INSERT) триггерін құру

Сигнал беру үшін SQLSTATE мәніне «45000» пайдаланылған, бұл «unhandled user-defined exception» (бұл «анықталмаған пайдаланушы қателігі») дегенді білдіреді.

Триггер жұмысын тексерейік. *smartphones* кестесіне жаңа жазба қосамыз. Жаңа тауар бағасын 0 деп енгізейік.

```
query_smartphones*
Limit to 1000 rows
1  INSERT INTO `products`.`smartphones`
2  (`ProductName`,`ProductCount`,`Price`)
3  VALUES
4  ("Nokia Limua", 2, 0);
5
```

```
Output
Action Output
#  Time      Action                                                  Message
1  14:44:19  INSERT INTO `products`.`smartphones` (`Pro...          Error Code: 1644. Бағаны дұрыс енгізіңіз
```

257 - сурет. *check_price* BEFORE INSERT триггерін құру

Сұранысты орындауда қателік туындайды. Кестеге жаңа жазба қосылған жоқ.

AFTER INSERT – кестеге INSERT операторы арқылы жазба қосылғаннан кейін орындалады. *students* кестесіне AFTER INSERT триггерін құрамыз.

Триггер атауы: *change_count*

Егер *students* кестесіне жаңа студент жазбасын (аты-жөні, тобы) қосамыз. *class* кестесінде енгізілген студент оқитын топтың студенттер саны 1 санға артуы керек.

class

Limit to 1000 rows

```
1 •  SELECT * FROM college.class;
```

Result Grid | Filter Rows: | Edit:

id	class_name	course	count
1	IS-22-1	1	2
2	IS-22-2	1	1
3	IS-22-3	1	1
6	IS-20-2	2	1
7	IS-20-3	2	3
NULL	NULL	NULL	NULL

258 - сурет. *class* кестесі

after_insert_trigger

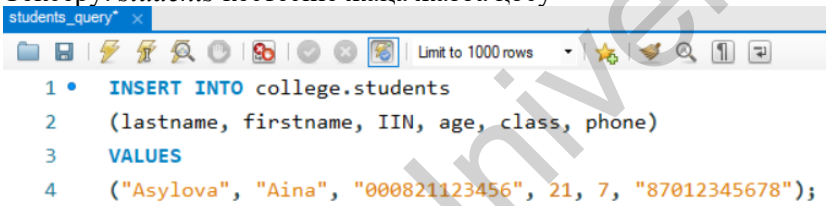
Limit to 1000 rows

```
1  •  USE college;
2     DELIMITER $$
3  •  CREATE TRIGGER change_count
4     AFTER INSERT
5     ON students
6     FOR EACH ROW
7     UPDATE college.class
8     SET class.count=class.count+1
9     WHERE NEW.class=class.id;
10    $$
```

259 - сурет. *change_count* (AFTER INSERT) триггерін құру

```
USE college;
DELIMITER $$
CREATE TRIGGER change_count
AFTER INSERT
ON students
        FOR EACH ROW
        UPDATE class
        SET count=count+1
WHERE NEW.class=id;
$$
```

Тексеру: *students* кестесіне жаңа жазба қосу

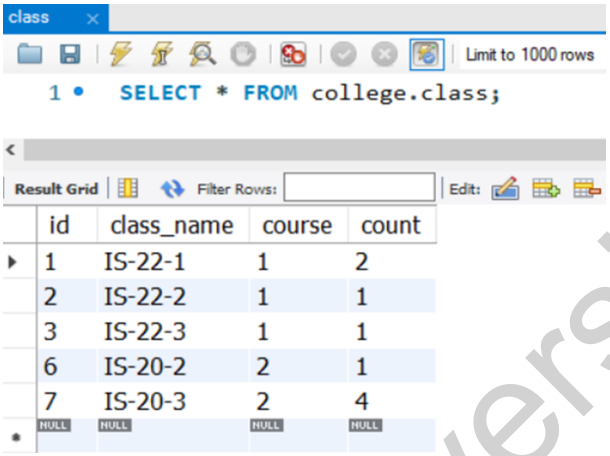


260 - сурет. *students* кестесіне жаңа жазба қосу

```
INSERT INTO college.students
(lastname, firstname, IIN, age, class, phone)
VALUES
("Asylova", "Aina", "000821123456", 21, 7, "87012345678");
```

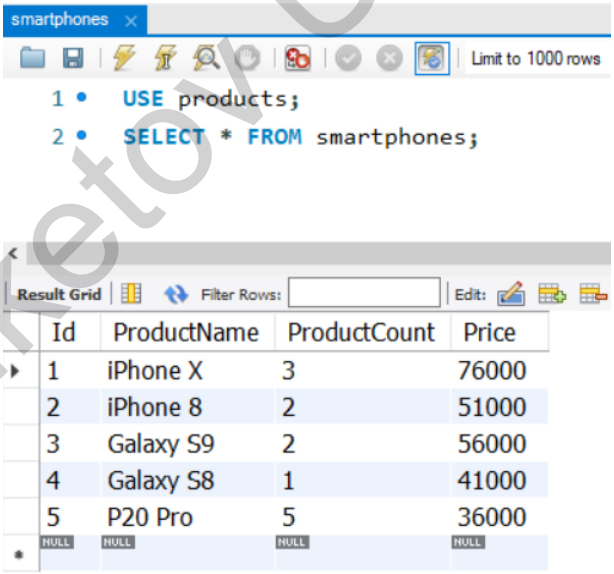
Нәтижесі:

Бұл студент нөмері *7* болатын топқа қосылады. Бұл нөмер *class* кестесіндегі *IS-20-3* тобына сәйкес. *IS-20-3* тобындағы студенттер саны бір санға артты.



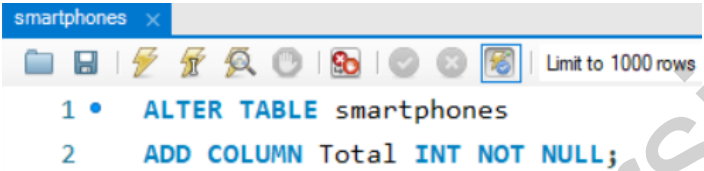
261 - сурет. *class* кестесі

Келесі Before UPDATE триггеріне мысалды қарастырмастан бұрын, мысалда қолданылатын *smartphones* кестесіне жаңа өріс қосайық.

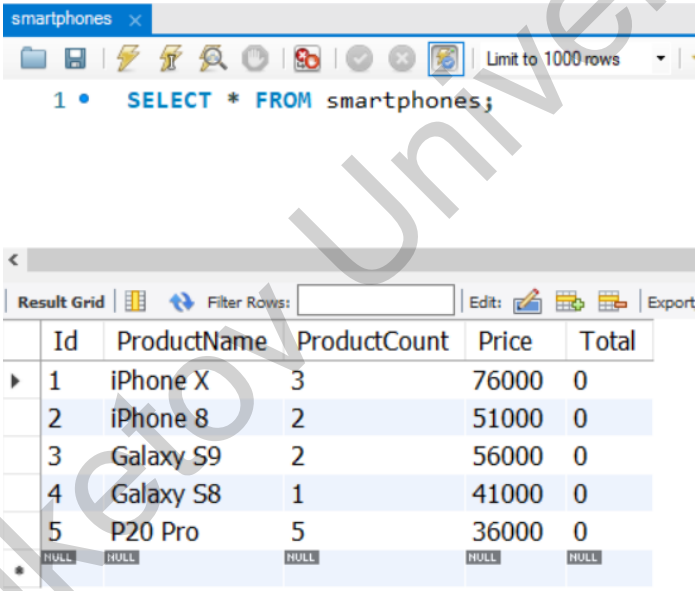


262 - сурет. *smartphones* кестесіне өріс қосу

smartphones кестесіне *Total* өрісін қосамыз. Бұл өріс мәні тауардың жалпы құнын (тауар саны мен тауар құнының көбейтіндісі) сақтайды.



263 - сурет. *smartphones* кестесіне *Total* өрісін қосу

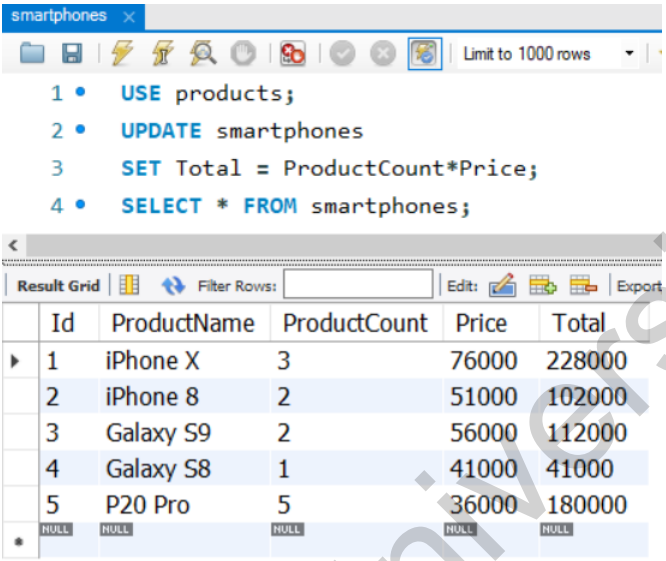


Id	ProductName	ProductCount	Price	Total
1	iPhone X	3	76000	0
2	iPhone 8	2	51000	0
3	Galaxy S9	2	56000	0
4	Galaxy S8	1	41000	0
5	P20 Pro	5	36000	0
NULL	NULL	NULL	NULL	NULL

264 - сурет. *smartphones* кестесі

Қосылған жаңа өрістің мәнін толтырамыз.

```
USE products;
UPDATE smartphones
SET Total = ProductCount*Price;
SELECT * FROM smartphones;
```



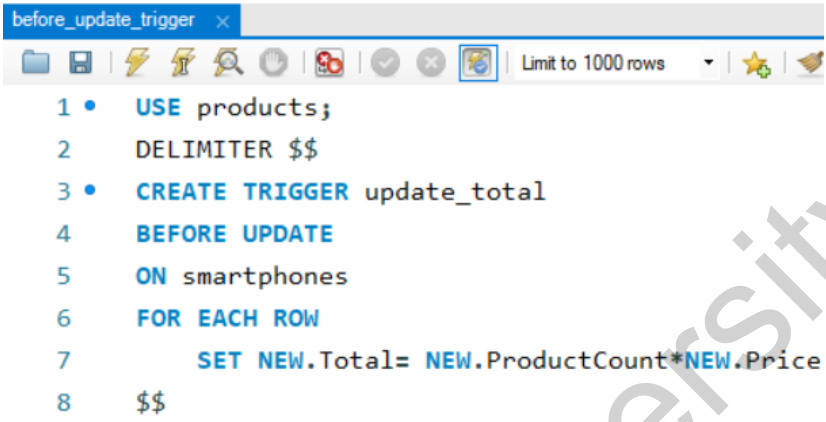
265 - сурет. *smartphones* кестесіндегі *Total* өрісін толтыру

Before UPDATE триггерін құру

Before UPDATE – бұл триггер кестені жаңалаудан бұрын, қандай да бір жазбаны өзгертуден бұрын орындалады.

Тауар санын немесе бағасын өзгерткенде тауардың жалпы құны автоматты түрде есептелуі керек.

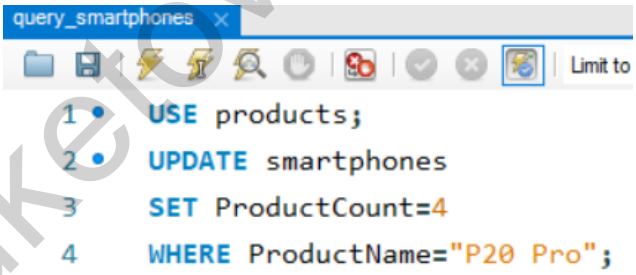
```
USE products;
DELIMITER $$
CREATE TRIGGER update_total
BEFORE UPDATE
ON smartphones
FOR EACH ROW
   SET NEW.Total= NEW.ProductCount*NEW.Price
$$
```



266 - сурет. *update_total* (BEFORE UPDATE) триггерін құру

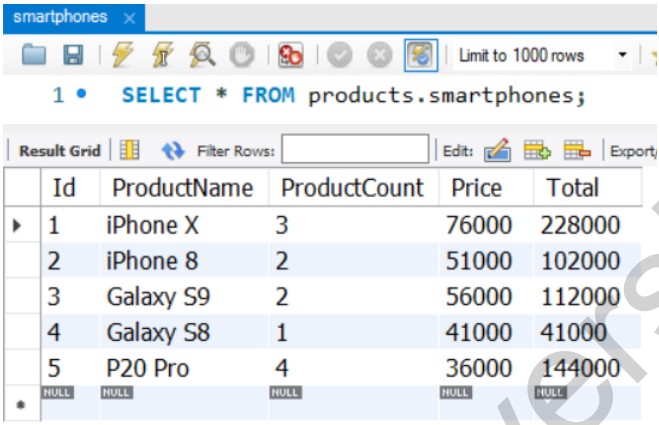
ProductName="P20 Pro" болатын тауардың санын өзгертеміз.

```
USE products;
UPDATE smartphones
SET ProductCount=4
WHERE ProductName="P20 Pro";
```



267 - сурет. Тауар санын өзгерту

Нәтижесі:

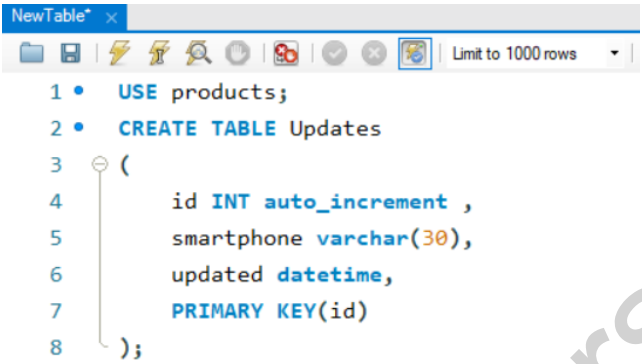


268 - сурет. *Total* өрісіде автоматты түрде өзгерді (264-суретпен салыстырыңыз).

AFTER UPDATE триггерін құру

AFTER UPDATE триггерін құрмас бұрын жаңа *Updates* кестесін құрайық. Бұл кестеде *smartphones* кестесінде өзгеріс енгізілген, өзгертілген уақыты тіркеліп отырады.

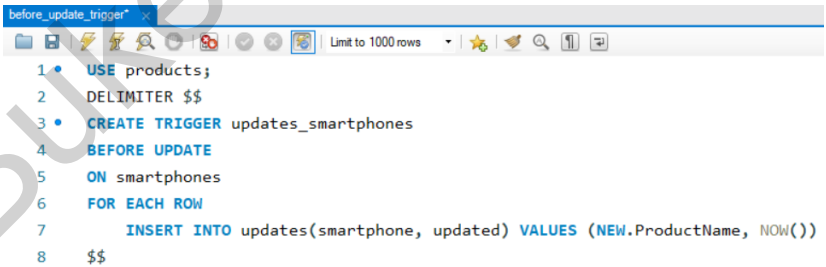
```
USE products;
CREATE TABLE Updates
(
id INT auto_increment ,
   smartphone VARCHAR(30),
   updated DATETIME,
   PRIMARY KEY(id)
);
```



269 - сурет. *Updates* кестесін құру

smartphones кестесінде өзгеріс енгізілсе, *updates_smartphones* триггері *updates* кестесіне тауар аты мен өзгеріс уақытын қосып отырады.

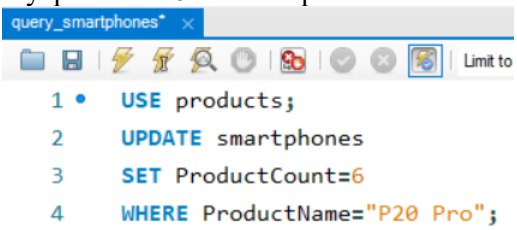
```
USE products;
DELIMITER $$
CREATE TRIGGER updates_smartphones
BEFORE UPDATE
ON smartphones
FOR EACH ROW
        INSERT INTO updates(smartphone, updated) VALUES
(NEW.ProductName, NOW())
$$
```



270 - сурет. *updates_smartphones* триггерін құру

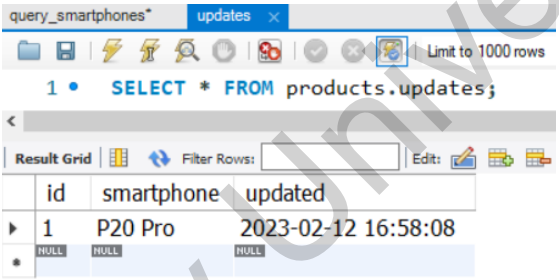
Триггер жұмысын тексеріп көрейік.

smartphones кестесіндегі жазбаны *ProductName* = «P20 Pro» өзгертеміз. Тауар санына 6 мәнін берейік.



271 - сурет. Жазбаны өзгерту

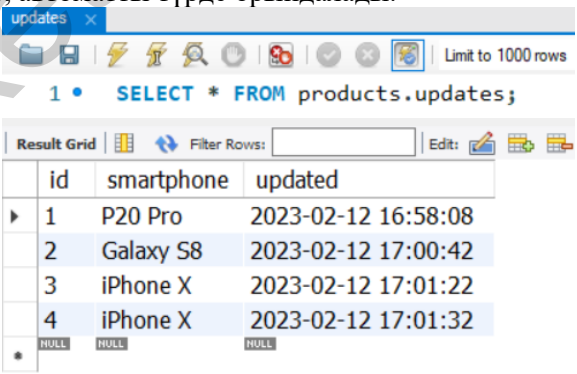
Нәтижесі:



272 - сурет. *updates* кестесі

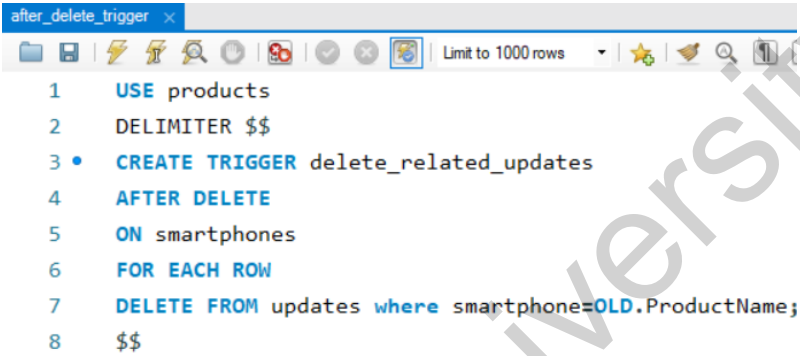
AFTER DELETE триггерін құру

AFTER DELETE триггерін кестеден қандай да бір жазба өшірілгенде, автоматты түрде орындалады.



273 - сурет. *updates* кестесі

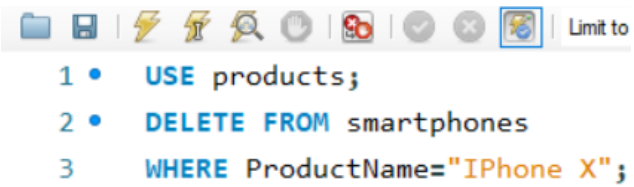
Егер *smartphones* кестесінен қандай да бір тауар жойылса, сол тауарға қатысты мәліметтер *updates* кестесінен жазбалар жойылуы керек.
Ол үшін *delete_related_updates* триггерін құрамыз.



274 - сурет. *delete_related_updates* (AFTER DELETE) триггерін құру

```
USE products
DELIMITER $$
CREATE TRIGGER delete_related_updates
AFTER DELETE
ON smartphones
FOR EACH ROW
        DELETE FROM updates where
smartphone=OLD.ProductName;
$$
```

Құрылған тригтгер жұмысын тексереміз. Smartphones кестесінен "IPhone X" жазбасын өшіріп тастаймыз.

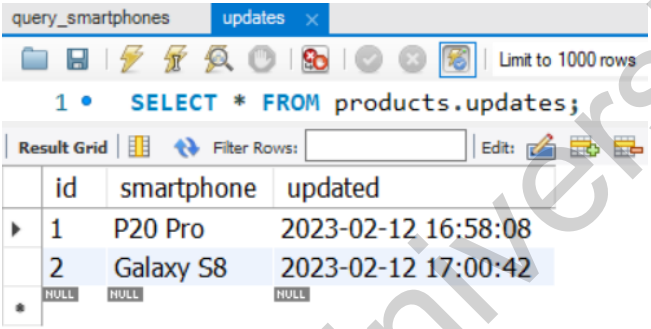


275 - сурет. *smartphones* кестесінен "IPhone X" жазбасын өшіру

```
USE products;
DELETE FROM smartphones
WHERE ProductName="IPhone X";
```

Нәтижесі:

```
SELECT * FROM products.updates;
```



276 - сурет. *updates* кестесі

BEFORE DELETE триггері

smartphones кестесінен тауар жойылса, *updates* кестесіне жойылған тауар туралы ақпарат (DELETE: тауар аты, уақыты) жазылып отыруы керек. Ол үшін *delete_updates* триггерін құрайық.

```
USE products
DELIMITER $$
CREATE TRIGGER delete_updates
BEFORE DELETE
ON smartphones
FOR EACH ROW
        INSERT INTO updates(smartphone, updated) VALUES
(CONCAT("DELETE:", OLD.ProductName), NOW())
$$
```

```
before_delete_trigger
Limit to 1000 rows
1  USE products
2  DELIMITER $$
3  CREATE TRIGGER delete_updates
4  BEFORE DELETE
5  ON smartphones
6  FOR EACH ROW
7      INSERT INTO updates(smartphone, updated) VALUES (CONCAT("DELETE:", OLD.ProductName), NOW())
8  $$
```

277 - сурет. BEFORE DELETE триггерін құру

Тексеру

```
query_smartphones*
Limit to 1000 rows
1  USE products;
2  DELETE FROM smartphones
3  WHERE ProductName="P20 Pro";
```

278 - сурет. *smartphones* кестесінен *"P20 Pro"* жазбасын өшіру

Нәтижесі:

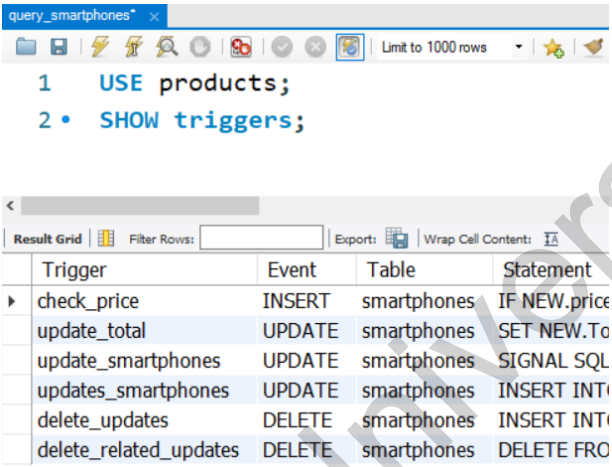
```
query_smartphones*  updates
Limit to 1000 rows
1  SELECT * FROM products.updates;
Result Grid | Filter Rows: | Edit:
```

id	smartphone	updated
2	Galaxy S8	2023-02-12 17:00:42
5	DELETE:P20 Pro	2023-02-12 17:21:13
NULL	NULL	NULL

279 - сурет. *updates* кестесі

Деректер қорындағы триггерлерді көрсету

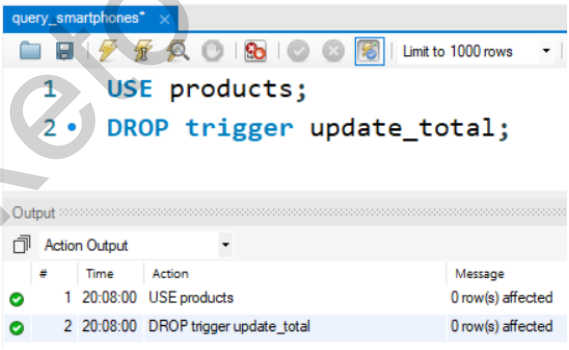
```
USE products;
SHOW triggers;
```



280 - сурет. *products* кестесіндегі триггерлер тізімі

Триггерді өшіру

```
DROP trigger тригге_атауы;
```



281 - сурет. Триггерді жою

Деректер қорын құруға арналған тапсырмалар

Тапсырма №1

1. Деректер қорын құру:ЖОО студенттерінің оқу үлгерімі.
2. Деректер қоры келесі байланысқан кестелерден тұрады: факультеттер, кафедралар, оқу топтары, студенттер, үлгерім парақтары.
3. Факультет кестесіниңбағандар: факультет атауы, деканның аты-жөні, кабинет нөмірі, ғимарат нөмірі, телефон нөмірі.
4. Кафедра кестесінің бағандары: кафедра аты, факультет, меңгеруші аты, кабинет нөмірі, ғимарат нөмірі, телефон нөмірі, оқытушылар саны.
5. Оқу топтары кестесінің бағандары: топ атауы, қабылданған жылы, оқу курсы, топтағы студенттер саны.
6. Оқушылар кестесінің бағандары: студент, тегі, аты, әкесінің аты, топ, туған жылы, жынысы, мекен-жайы, қаласы, телефоны.
7. Баға парағы кестесінің келесі атрибуттары бар: топ, студент, пән, баға.
8. Кестелерді деректермен толтырыңыз.
9. CSV файлдарынан жазбаларды импорттаңыз.
10. Деректер қорының ER моделін құрыңыз.
11. Кестелерден жазбаларды алу үшін сұраныстар құрыңыз.
12. JOIN және UNION операторлары арқылы байланысқа кестелерден жазбаларды алыңыз.
13. Сұраныстар нәтижесін ұсыныстар мен уақытша кестелерде сақтаңыз.
14. Деректер қорына сақталатын процедуралар мен функциялар қосыңыз.
15. Кестелер үшін триггерлер құрыңыз.

Тапсырма №2

1. Деректер қорын құру: Супермаркеттің ақпараттық жүйесі.
2. Деректер қоры келесі байланысқан кестелерден тұрады: бөлімдер, қызметкерлер, тауарлар, тауарларды сату, лауазымдар.

3. Бөлімдер кестесінің келесі бағандары бар: бөлім атауы, сатушылар саны, зал нөмірі.

4. Қызметкерлер кестесінде келесі атрибуттар бар: тегі, аты, әкесінің аты, бөлім, туған жылы, жұмыс істеген жылы, еңбек өтілі, лауазымы, жынысы, мекенжайы, қаласы, телефоны.

5. Лауазымдар кестесінің келесі өрістері бар: лауазым атауы, жалақы.

6. Тауарлар кестесінің келесі өрістері бар: тауар атауы, бөлім, өндірген ел, сақтау шарттары, жарамдылық мерзімі.

7. Тауарларды сату кестесінің келесі өрістері бар: сатушы болып табылатын қызметкер, тауардың күні, уақыты, саны, бағасы, сомасы.

8. Кестелерді деректермен толтырыңыз.

9. CSV файлдарынан жазбаларды импорттаңыз.

10.Деректер қорының ER моделін құрыңыз.

11.Кестелерден жазбаларды алу үшін сұраныстар құрыңыз.

12.JOIN және UNION операторлары арқылы байланысқа кестелерден жазбаларды алыңыз.

13.Сұраныстар нәтижесін ұсыныстар мен уақытша кестелерде сақтаңыз.

14.Деректер қорына сақталатын процедуралар мен функциялар қосыңыз.

15.Кестелер үшін триггерлер құрыңыз.

Тапсырма №3

1. Деректер қорын құру: кітапхананың ақпараттық жүйесі.

2. Деректер қоры келесі кестелерден тұрады: кітапханалар, кітапхана қоры, әдебиеттер түрі, қызметкерлер, қорды толықтыру.

3. Кітапхана кестесінің бағандары: аты, мекенжайы, қаласы.

4. Кітапхана қоры кестесінің бағандары: қор атауы, кітапхана, кітаптар саны, журналдар саны, газеттер саны, жинақтар саны, диссертациялар саны, авторефераттар саны.

5. Әдебиет түрлері кестесінің бағандары: тип атауы.

6. Қызметкерлер кестесінде келесі бағандар бар: қызметкердің тегі, кітапханасы, қызметі, туған жылы, жұмыс істеген жылы, білімі, жалақысы.

7. Қорды толықтыру кестесінің келесі атрибуттары болады: қор, қызметкер, күні, әдебиет көзінің атауы, әдебиет түрі, баспасы, шыққан күні, дана саны.

8. Кестелерді деректермен толтырыңыз.

9. CSV файлдарынан жазбаларды импорттаңыз.

10. Деректер қорының ER моделін құрыңыз.

11. Кестелерден жазбаларды алу үшін сұраныстар құрыңыз.

12. JOIN және UNION операторлары арқылы байланысқа кестелерден жазбаларды алыңыз.

13. Сұраныстар нәтижесін ұсыныстар мен уақытша кестелерде сақтаңыз.

14. Деректер қорына сақталатын процедуралар мен функциялар қосыңыз.

15. Кестелер үшін триггерлер құрыңыз.

Тапсырма №4

1. Деректер қорын құру: туристік фирманың ақпараттық жүйесі.

2. Деректер қоры келесі кестелерден тұрады: пансионаттар, турлар, клиенттер, жолдамалар, тұрғын үй түрі.

3. Пансионаттар кестесінің келесі бағандары бар: пансионаттың атауы, мекен-жайы, қаласы, елі, телефоны, аумақтың сипаттамасы, бөлмелер саны, бассейннің болуы, медициналық қызметтердің болуы, СПА-ның болуы, теңізге дейінгі қашықтық.

4. Тұрғын үй кестесінің бағандары: атауы (үй, бунгало, пәтер, 1-бөлмелі, 2-бөлмелі және т.б.), тұрғын үй санаты (люкс, жартылай люкс және т.б), бір күндік жалға алу бағасы.

5. Тур кестесінің бағандары: турдың атауы (Еуропа, Орталық Азия, Тибет және т.б.), қону орны (қонақ үй, қонақүй, шатыр және т.б.), тамақ түрі (бір реттік , екі мезгіл, үш мезгіл тамақ, таңғы ас), бір күндік тур бағасы.

6. Клиенттер кестесінде келесі бағандар бар: тегі, аты, әкесінің аты, төлқұжат мәліметтері, туған күні, мекен-жайы, қаласы, телефоны.

7. Жолдамалар кестесінде келесі бағандар бар: клиент, пансионат, үй түрі, келу күні, кету күні, балалардың болуы,

медициналық сақтандырудың болуы, адам саны, бағасы, сомасы.

8. Кестелерді деректермен толтырыңыз.

9. CSV файлдарынан жазбаларды импорттаңыз.

10.Деректер қорының ER моделін құрыңыз.

11.Кестелерден жазбаларды алу үшін сұраныстар құрыңыз.

12.JOIN және UNION операторлары арқылы байланысқа кестелерден жазбаларды алыңыз.

13.Сұраныстар нәтижесін ұсыныстар мен уақытша кестелерде сақтаңыз.

14.Деректер қорына сақталатын процедуралар мен функциялар қосыңыз.

15.Кестелер үшін триггерлер құрыңыз.

Тапсырма №5

1. Деректер қорын құру: қалалық автокәсіпорынның ақпараттық жүйесі.

2. Деректер келесі кестелерден тұрады: көліктер, жүргізушілер, маршруттар, жолсеріктер, жөндеу.

3. Автокөлік кестесінің келесі бағандары бар: көліктің атауы (автобустар, таксилер, тұрақты маршруттық таксилер, басқа жолаушылар көлігі, жүк көлігі және т.б.), жұмыс уақытының саны, жүгіріс, жөндеу саны, сипаттамалары.

4. Маршрут кестесінің келесі атрибуттары бар: маршрут атауы, көлік, жүргізуші, жұмыс кестесі.

5. Жүргізушілер кестесінде келесі атрибуттар бар: тегі, аты, әкесінің аты, туған жылы, жұмыс істеген жылы, еңбек өтілі, лауазымы, жынысы, мекенжайы, қаласы, телефоны.

6. Қызмет көрсетуші персонал кестесінің келесі атрибуттары болады: лауазымы (техниктер, дәнекерлеушілер, слесарлар, құрастырушылар және т.б.), тегі, аты, әкесінің аты, туған жылы, жұмыс істеген жылы, еңбек өтілі, жынысы, мекенжайы, қаласы, телефоны. .

7. Жөндеу кестесінің келесі бағандары бар: жөндеу гаражының атауы, жөнделетін көлік құралы, жөндеу түрі, алынған күні, жөндеуден кейін берілген күні, жөндеу нәтижесі, жөндеуді жүзеге асыратын персонал.

8. Кестелерді деректермен толтырыңыз.

9. CSV файлдарынан жазбаларды импорттаңыз.

10.Деректер қорының ER моделін құрыңыз.

11.Кестелерден жазбаларды алу үшін сұраныстар құрыңыз.

12.JOIN және UNION операторлары арқылы байланысқа кестелерден жазбаларды алыңыз.

13.Сұраныстар нәтижесін ұсыныстар мен уақытша кестелерде сақтаңыз.

14.Деректер қорына сақталатын процедуралар мен функциялар қосыңыз.

15.Кестелер үшін триггерлер құрыңыз.

Тапсырма №6

1. Деректер қорын құру: емхананың ақпараттық жүйесі.

2. Деректер қоры келесі кестелерден тұрады: дәрігерлер, науқастар, ауру тарихы, бөлімшелер, қызметкерлер.

3. Бөлімше кестесінің келесі бағандары бар: бөлімшенің атауы (хирургия, терапия, неврология және т.б.), қабат, бөлме нөмірлері, меңгерушінің аты-жөні.

4. Дәрігерлер кестесінің бағандары: тегі, аты, әкесінің аты, лауазымы, жұмыс өтілі, ғылыми атағы, мекенжайы, өзі жұмыс істейтін бөлімшенің нөмірі.

5. Пациенттер кестесінде келесі атрибуттар бар: тегі, аты, әкесінің аты, мекен-жайы, қаласы, жасы, жынысы.

6. Диагноздар кестесінде келесі атрибуттар бар: диагноздың атауы, аурудың белгілері, емдеу мерзімі, тағайындаулар.

7. Ауру тарихы кестесінде келесі атрибуттар бар: науқас, дәрігер, диагноз, емдеу, ауру күні, емдеу күні, емдеу түрі (амбулаторлық, стационарлық).

8. Кестелерді деректермен толтырыңыз.

9. CSV файлдарынан жазбаларды импорттаңыз.

10.Деректер қорының ER моделін құрыңыз.

11.Кестелерден жазбаларды алу үшін сұраныстар құрыңыз.

12.JOIN және UNION операторлары арқылы байланысқа кестелерден жазбаларды алыңыз.

13.Сұраныстар нәтижесін ұсыныстар мен уақытша кестелерде сақтаңыз.

14.Деректер қорына сақталатын процедуралар мен функциялар қосыңыз.

15.Кестелер үшін триггерлер құрыңыз.

Тапсырма №7

1. Деректер қорын құру: фильмдер ақпараттық жүйесі.
2. Деректер қоры келесі кестелерден тұрады: фильмдер, жанрлар, киностудиялар, режиссерлер.
3. Фильмдер кестесінің келесі бағандары бар: фильм атауы, фильм жанры, жылы, режиссер, киностудияс, сипаттамасы, рейтинг.
4. Жанрлар кестесінің келесі бағандары бар: жанр атауы, жанр сипаттамасы
5. Киностудиялар кестесінің келесі бағандары бар: атауы, құрылған жылы, директоры, мекен-жайы, телефон нөмірі.
6. Режиссерлер: аты-жөні, телефон нөмірі, мекен-жайы.
7. Кестелерді деректермен толтырыңыз.
8. CSV файлдарынан жазбаларды импорттаңыз.
9. Деректер қорының ER моделін құрыңыз.
10. Кестелерден жазбаларды алу үшін сұраныстар құрыңыз.
11. JOIN және UNION операторлары арқылы байланысқа кестелерден жазбаларды алыңыз.
12. Сұраныстар нәтижесін ұсыныстар мен уақытша кестелерде сақтаңыз.
13. Деректер қорына сақталатын процедуралар мен функциялар қосыңыз.
14. Кестелер үшін триггерлер құрыңыз.

Тапсырма №8

1. Деректер қорын құру: дәріханалар желісі ақпараттық жүйесі.
2. Деректер қоры келесі кестелерден тұрады: дәріханалар, дәрі-дәрмектер, жеткізушілер, категориялар, өндіруші.
3. Дәріханалар кестесінің келесі бағандары бар: дәріхана нөмірі, мекен-жайы, телефон нөмірі, дәріханашы.
4. Дәрі-дәрмек кестесінің келесі бағандары бар: атауы, бренд, өндіруші, бағасы, бар болуы, сипаттамасы.

5. Жеткізушілер кестесінің келесі бағандары бар: атауы, мекен-жайы, телефон нөмірі.

6. Категорялар кестесінің келесі бағандары бар: атауы, сипаттамасы.

7. Өндіруші кестесінің келесі бағандары бар: өндіруші компания атауы, мекен-жайы.

8. Кестелерді деректермен толтырыңыз.

9. CSV файлдарынан жазбаларды импорттаңыз.

10.Деректер қорының ER моделін құрыңыз.

11.Кестелерден жазбаларды алу үшін сұраныстар құрыңыз.

12.JOIN және UNION операторлары арқылы байланысқа кестелерден жазбаларды алыңыз.

13.Сұраныстар нәтижесін ұсыныстар мен уақытша кестелерде сақтаңыз.

14.Деректер қорына сақталатын процедуралар мен функциялар қосыңыз.

15.Кестелер үшін триггерлер құрыңыз.

Тапсырма №9

1. Деректер қорын құру: балабақшаның ақпараттық жүйесі.

2. Деректер қоры келесі кестелерден тұрады: топтар, тәрбиеленушілер, қызметкерлер, тауарлар (жиһаз, электронды құрылғылар, тұрмыстық заттар).

3. Топтар кестесінің келесі бағандары бар: топ атауы, топтағы тәрбиеленушілер саны, сипаттамасы.

4. Тәрбиеленушілер кестесінің бағандары: аты-жөні, құжат нөмірі, жасы, жынысы, тобы, ата-анасы, ата-анасының нөмірі.

5. Қызметкерлер кестесінде келесі атрибуттар бар: аты-жөні, қызметі, топ нөмірі, телефон нөмірі, мекен-жайы.

6. Тауарлар кестесінде келесі атрибуттар бар: тауар аты, артикул, топ, сипаттамасы.

7. Кестелерді деректермен толтырыңыз.

8. CSV файлдарынан жазбаларды импорттаңыз.

9. Деректер қорының ER моделін құрыңыз.

10.Кестелерден жазбаларды алу үшін сұраныстар құрыңыз.

11.JOIN және UNION операторлары арқылы байланысқа кестелерден жазбаларды алыңыз.

12.Сұраныстар нәтижесін ұсыныстар мен уақытша кестелерде сақтаңыз.

13.Деректер қорына сақталатын процедуралар мен функциялар қосыңыз.

14.Кестелер үшін триггерлер құрыңыз.

Тапсырма №10

1. Деректер қорын құру: Зергерлік дүкендер желісінің ақпараттық жүйесі.

2. Деректер қоры келесі кестелерден тұрады: дүкендер, зергерлік бұйымдар, қызметкерлер, жеткізушілер, өндіруші компания.

3. Дүкендер кестесінің келесі бағандары бар: дүкен нөмірі, мекен-жайы, телефон нөмірі, қызметкер нөмірі.

4. Зергерлік бұйымдар кестесінің келесі бағандары бар: атауы, бұйым түрі, материал, жеткізуші, өндіруші компания нөмірі, дүкен нөмірі.

5. Жеткізушілер кестесінің келесі бағандары бар: атауы, мекен-жайы, телефон нөмірі.

6. Өндіруші компаниялар кестесінің келесі бағандары бар: атауы, мекен-жайы, телефон нөмірі.

7. Қызметкерлер кестесінің келесі бағандары бар: аты-жөні, телефон нөмірі, мекен-жайы, құжат нөмірі.

8. Кестелерді деректермен толтырыңыз.

9. CSV файлдарынан жазбаларды импорттаңыз.

10.Деректер қорының ER моделін құрыңыз.

11.Кестелерден жазбаларды алу үшін сұраныстар құрыңыз.

12.JOIN және UNION операторлары арқылы байланысқа кестелерден жазбаларды алыңыз.

13.Сұраныстар нәтижесін ұсыныстар мен уақытша кестелерде сақтаңыз.

14.Деректер қорына сақталатын процедуралар мен функциялар қосыңыз.

15.Кестелер үшін триггерлер құрыңыз.

Тест сұрақтары

1. Реляциялық мәліметтер қорының моделін ұсынған ғалым
A) Дак Энгельбарт
B) Деннис Ритчи
C) Эдгар Кодд
D) Бьёрн Страуструп
E) Тим Бернерс-Ли
2. Кілттік өріс....
A) қайталанбайтын ерекше өріс
B) 8 байттан аспайтын өріс
C) жолдық типті кез келген өріс
D) қайталанатын өріс
E) автоматты түрде толтырылатын өріс
3. Барлық жазбаларды алу
A) SELECT ALL FROM table;
B) SELECT RECORDS FROM table;
C) SELECT * FROM table;
D) SHOW ALL(records) FROM table;
E) SHOW * FROM table;
4. Кесте келесі өрістерден тұрады: ID, Name, Address, Country. Елдердің тізімін алу. Жазбалар қайталанбауы керек.
A) SELECT Country FROM table;
B) SELECT UNIQUE Country FROM table;
C) SHOW Country FROM table;
D) SELECT DISTINCT(Country) FROM table;
E) SHOW DISTINCT(Country) FROM table;
5. Кесте келесі өрістерден тұрады: ID, Name, Address, Country. USA және UK тұратын қолданушылардың тізімін шығарыңыз.
A) SELECT * FROM table WHERE Country=”UK” and Country=”USA”;
B) SELECT * FROM table WHERE Country=”UK” , ”USA”;
C) SELECT * FROM table WHERE Country=”UK” or Country=”USA”;
D) SELECT * FROM table Country=”UK” or Country=”USA”;
E) SELECT * FROM table WHERE Country (”UK” , ”USA”);

6. Кесте келесі өрістерден тұрады: ID, Name, Address, Country. USA және UK тұрмайтын қолданушылардың тізімін шығарыңыз.

A) SELECT * FROM table WHERE Country='UK' and Country='USA';

B) SELECT * FROM table NOT WHERE Country='UK' , 'USA';

C) SELECT * FROM table Country='UK' or Country='USA';

D) SELECT * FROM table WHERE NOT(Country='UK' or Country='USA');

E) SELECT * FROM table WHERE Country NOT ('UK', 'USA');

7. Сұрыптау әдісі

A) ORDER өріс

B) SORT BY өріс

C) ORDER BY өріс

D) SORT өріс

E) HAVING өріс

8. Сұрыптау әдісі, өсу реті бойынша сұрыптау

A) ORDER BY өріс DESC

B) ORDER BY өріс ASC

C) ORDER BY ASC өріс

D) SORT BY өріс ASC

E) өріс ASC

9. Сұрыптау әдісі, кему реті бойынша сұрыптау

A) ORDER BY өріс ASC

B) ORDER BY өріс

C) ORDER BY өріс DESC

D) SORT BY өріс ASC

E) REVERSE өріс

10. SELECT * FROM Customers ORDER BY Country ASC, CustomerName DESC;

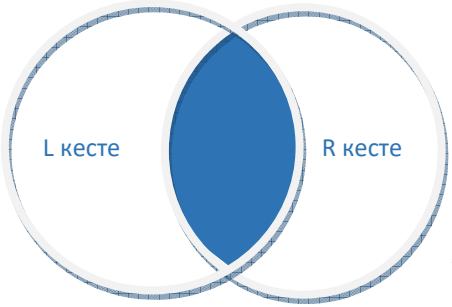
A) Customers кестесінен барлық өрістерді алу. Country өрісінің кему реті бойынша сұрыптау.

B) Customers кестесінен барлық өрістерді алу. CustomerName өрісінің өсу реті бойынша сұрыптау.

C) Customers кестесінен барлық өрістерді алу. Country өрісінің кему реті бойынша және CustomerName өрісінің кему реті бойынша сұрыптау.
D) қате сұраныс
E) Customers кестесінен барлық өрістерді алу. Country өрісінің өсу реті бойынша және CustomerName өрісінің кему реті бойынша сұрыптау.
11. my_db деректер қорын ағымдық дерекқор ретінде орнату
A) drop database my_db
B) update my_db
C) use my_db
D) select my_db
E) my_db
12. WHERE CustomerName LIKE 'a%'
A) аты а әріпінен аяқталатын тұтынушылар
B) аты а әріпінен басталатын тұтынушылар
C) аты а әріпінен басталатын және екі символдан тұратын тұтынушылар
D) атында а әріпі бар тұтынушылар
E) аты кіші а әріпінен басталатын тұтынушылар
13. WHERE CustomerName LIKE '_r%'
A) аты r әріпінен аяқталатын тұтынушылар
B) аты r әріпінен басталатын тұтынушылар
C) аты r әріпінен басталатын және екі символдан тұратын тұтынушылар
D) атының екінші әрісі r болатын тұтынушылар
E) атында r әріпі бар тұтынушылар
14. SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;
A) Бағасы 10 мен 20 аралығында тауарлар
B) Бағасы 10 мен 20 аралығындағы емес тауарлар
C) Бағасы 10 мен 20-дан жоғары тауарлар
D) Бағасы 20-дан жоғары тауарлар
E) қате сұраныс
15. SELECT CustomerName, CONCAT(Address,', ',City) AS Address FROM Customers; Нәтижесінде қандай өрістер шығарылады?

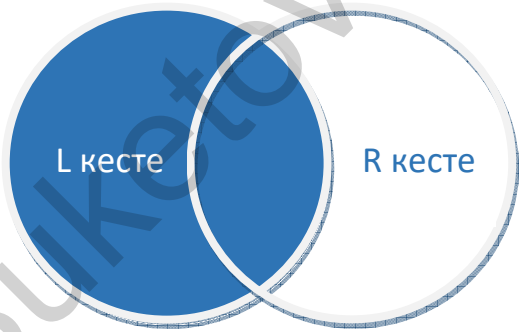
A) CustomerName, Address
B) CustomerName, Address, City
C) CustomerName, Address, City, Address
D) CustomerName, CONCAT(Address,', ',City), Address
E) CustomerName, CONCAT(Address,', ',City)

16. Сызбада қандай қосылу JOIN түрі көрсетілген?



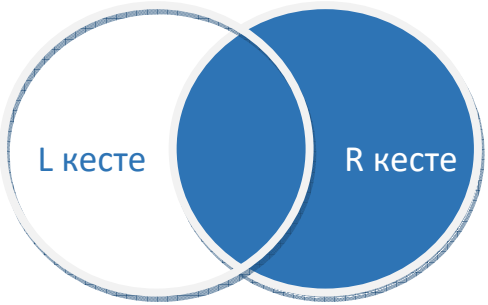
A) LEFT (OUTER) Join
B) RIGHT (OUTER) Join
C) FULL (OUTER) Join
D) INNER Join
E) SELF Join

17. Сызбада қандай қосылу JOIN түрі көрсетілген?



A) INNER JOIN
B) LEFT (OUTER) JOIN
C) RIGHT (OUTER) JOIN
D) FULL (OUTER) JOIN
E) SELF JOIN

18. Сызбада қандай қосылу JOIN түрі көрсетілген?



A) INNER Join
B) LEFT (OUTER) Join
C) RIGHT (OUTER) Join
D) FULL (OUTER) Join
E) SELF Join

19. Сұранысты толықтырыңыз. SELECT Orders.OrderID, Customers.CustomerName FROM Orders INNER JOIN Customers ____ Orders.CustomerID = Customers.CustomerID;

A) WHERE
B) WHEN
C) ON
D) IF
E) LIKE

20. Сұранысты толықтырыңыз. Екі кестеде де сәйкестік бар барлық жазбаларды таңдау үшін дұрыс JOIN түрін таңдаңыз. SELECT * FROM Orders ____ ON Orders.CustomerID=Customers.CustomerID;

A) LEFT JOIN Customers
B) RIGHT JOIN Customers
C) FULL JOIN Customers
D) INNER JOIN Customers
E) SELF JOIN Customers

21. Тұтынушылар кестесіндегі барлық жазбаларды және Тапсырыстар кестесіндегі барлық сәйкестіктерді таңдау үшін дұрыс JOIN сөйлемін таңдаңыз. SELECT * FROM Orders ____ ON Orders.CustomerID=Customers.CustomerID;

A) FULL JOIN Customers

B) SELF JOIN Customers
C) INNER JOIN Customers
D) LEFT JOIN Customers
E) RIGHT JOIN Customers
22. Жаңа жазба қосу әдісі
A) ADD
B) INSERT NEW
C) INSERT RECORD
D) INSERT INTO
E) CONCAT
23. Жазбаны өзгерту әдісі
A) SET
B) EDIT
C) UPDATE
D) CHANGE
E) MODIFY
24. Өріске мән орнату
A) UPDATE
B) EDIT
C) CHANGE
D) SET
E) MODIFY
25. Барлық жазбаларды өшіру
A) DELETE FROM Customers
B) DEL FROM Customers
C) REMOVE FROM Customers
D) TRUNCATE FROM Customers
E) REMOVE ALL FROM Customers
26. Жаңа деректер қорын құру әдісі
A) CREATE DATABASE databasename;
B) DATABASE databasename;
C) INSERT DATABASE databasename;
D) NEW DATABASE databasename;
E) CREATE databasename;
27. Деректер қорын өшіру әдісі
A) CREATE DATABASE databasename;
B) REMOVE DATABASE databasename;
C) DROP DATABASE databasename;

D) DELETE DATABASE databasename;
E) ERASE DATABASE databasename;
28. SQL көмегімен «Persons» кестесіндегі жазбалар санын қалай қайтаруға болады?
A) SELECT COLUMNS(*) FROM Persons
B) SELECT NO(*) FROM Persons
C) SELECT COUNT(*) FROM Persons
D) SELECT LEN(*) FROM Persons
E) SELECT (*) FROM Persons
29. SELECT * FROM Orders LIMIT 10;
A) кездейсоқ 10 жазбаны көрсету
B) 10 саны бар барлық тапсырыстарды көрсету
C) 10 өрісті көрсету
D) алғашқы 10 жазбаны көрсету
E) Соңғы 10 жазба
30. Агрегаттық функциялар
A) Бір мәнді санау және қайтару үшін жолдар жинағын өңдеу, жиынтық мәндерді алу үшін пайдаланылады.
B) мәндерді сүзетін функциялар
C) мәндерді сұрыптайтын функциялар
D) барлық мәндерді қосатын функциялар
E) деректерді өзгерту функциялары
31. Нәтижесі: SELECT MIN(PRICE), MAX(PRICE) FROM Orders;
A) сұраныс қате, MIN мен MAX функциялары жоқ
B) қате сұраныс, екі функцияны бір уақытта пайдалану мүмкін емес
C) нәтижесінде ең төменгіден максимумға дейін сұрыпталған бағалар тізімі
D) ең төменгі және максималды баға
E) price бағанын кему ретінмен сұрыптау
32. GROUP BY функциясы дұрыс қолданылған сұранысты таңдаңыз:
A) SELECT seller_id, COUNT(*) FROM Orders GROUP BY seller_id;
B) SELECT COUNT(*) FROM Orders GROUP ON seller_id;
C) SELECT seller_id, count(*) FROM Orders GROUP seller_id;

D) SELECT COUNT(*) FROM Orders GROUP seller_id;
E) SELECT COUNT(*) FROM Orders GROUPBY seller_id;
33. Ең қымбат құны бар тапсырыс туралы ақпаратты көрсететін ішкі сұранысты пайдаланып дұрыс жазылған сұранысты таңдаңыз:
A) SELECT COUNT(*) FROM Orders
B) SELECT * FROM Orders WHERE price = max
C) SELECT * FROM Orders WHERE price = (SELECT BIG(price) FROM Orders)
D) SELECT price = max FROM Orders
E) SELECT * FROM Orders WHERE price = (SELECT MAX(price) FROM Orders)
34. JOIN көмегімен құрастырылған дұрыс сұранысты көрсетіңіз. Бұл сұраныс тапсырыс ID нөмері, тұтынушы мен сатушының атын көрсетеді:
A) SELECT Orders.id, Orders.customer_name, Sellers.id FROM Orders LEFT JOIN ON Sellers AND Orders.seller_id = Sellers.id;
B) SELECT id AND customer_name AND seller_id FROM Orders LEFT JOIN Sellers ON seller_id = id;
C) SELECT Orders.id, Orders.customer_name, Sellers.id FROM Orders JOIN Sellers WHEN Orders.seller_id = Sellers.id;
D) SELECT id, customer_name FROM Orders, Sellers seller_id = Sellers.id;
E) SELECT Orders.id, Orders.customer_name, Sellers.id FROM Orders LEFT JOIN Sellers ON Orders.seller_id = Sellers.id;
35. SQL-де ағымдағы күнді қалай алуға болады?
A) SELECT now();
B) SELECT year();
C) SELECT year(now());
D) SELECT year from Date;
E) SELECT year(today);
36. Кестедегі өрісті жою
A) ALTER TABLE Users DROP COLUMN name;
B) TRUNCATE Users DROP COLUMN name;
C) DROP Users COLUMN name;
D) SELECT Users DROP COLUMN name;
E) ALTER TABLE Users DELETE COLUMN name;

37. Топтастырудан кейін сүзу шарттарын орнату үшін қандай кілт сөзді пайдалануға болады?

A) WHERE
B) SELECT
C) COUNT
D) HAVING
E) IF

38. Кесте атауына бүркеншік (псевдоним) ат қою үшін қандай түйінді сөзді қолдануға болады?

A) AS
B) LIKE
C) IS
D) ALIAS
E) SET

39. Өріс мәндері ерекше қайталанбауы үшін, өріске берілетін атрибут?

A) INDEX
B) AUTO_INCREMENT
C) DEFAULT
D) UNIQUE
E) NOT NULL

40. Ағымдағы кестенің table құрылымын көру

A) SELECT * FROM table
B) DESCRIBE table
C) SHOW TABLE STRUCTURE table
D) DROP TABLE table
E) SHOW table

№	жауабы	№	жауабы	№	жауабы	№	жауабы
1	C	11	C	21	D	31	D
2	A	12	B	22	D	32	A
3	C	13	D	23	C	33	E
4	D	14	A	24	D	34	E
5	C	15	A	25	A	35	A
6	D	16	D	26	A	36	C
7	C	17	B	27	C	37	D
8	B	18	C	28	C	38	A
9	C	19	C	29	D	39	D
10	E	20	D	30	A	40	B

Ағылшынша-қазақша сөздік

№	*Ағылшын тілінде*	*Қазақ тілінде*
1	1 row in set	жиында 1 қатар/жол
2	1 row affected	1 жол өзгертілді
3	adjust	реттеу
4	after	кейін
5	all	барлығы
6	alter table	кестені өзгерту
7	any	кез келген
8	Application Programming Interface (API)	қосымшаның программалық интерфейсі
9	apply	қолдану
10	architecture	архитектура
11	attribute	атрибут
12	authentification method	аутентификация әдісі
13	auto increment	автоматты түрде өсуі
14	average, avg()	орташа мән
15	before	дейін
16	begin	бастау
17	between	Аралығы
18	built-in	кірістірілген
19	call	шақыру
20	can't drop database 'college';	«college» дерекқорын жою мүмкін емес
21	check	тексеру
22	client	клиент
23	client-server	клиент-серверлік
24	column	баған
25	command line client	mysql командалық жол
26	commit	міндеттеу
27	concatenate (concat)	біріктіру
28	configuration	конфигурация
29	connection succeeded	қосылыс (байланыс) сәтті аяқталды

30	connectors	қосқыштар
31	constraint	шектеу
32	control	бақылау
33	cortege	кортеж
34	count	саны
35	create	құру
36	create database	деректер қорын құру
37	create function	функция құру
38	create table	кесте құру
39	create user	пайдаланушы құру
40	crossplatform	кроссплатформалық
41	current	ағымдағы
42	data	дерек
43	database	деректер қоры
44	database doesn't exist	деректер қоры жоқ
45	database management systems	деректер қорын басқару жүйесі
46	database schema	деректер қорының схемасы
47	declaration_section	Жариялау бөлімі
48	declare	жариялау
49	default	үнсіздік жағдайда, үнсіздік бойынша
50	definition	анықтама
51	delete	өшіру
52	delimiter	бөлгіш
53	deny	бас тарту, болдырмау
54	describe	сипаттау
55	deterministic	детерминистік
56	diagram	диаграмма
57	distinct	анықталған, әр түрлік
58	domain	домен
59	drop	өшіру
60	drop function	функцияны өшіру
61	drop procedure	процедураны өшіру
62	duration	ұзақтығы
63	end	соңы

66	executable_section	орындалатын бөлімі
67	execute	орындау
68	export	экспорттау
69	extra	қосымша
70	filter	сүзгілеу
71	flield	өріс
	foreign key	сыртқы кілт
72	full	толық
73	function	функция
74	grant	тағындау, ұсыну
75	group by	топтау
76	having	жазбаларды топтау үшін шарт қою үшін қолданылады
77	identified by	анықталған
78	if	егер
79	in	тізімдермен жұмыс істегенде қолданылатын оператор. Қандай да бір мәннің бір тізімде бар екенін тексереді 5 in (5, 6, 7, 8) – true
80	inner join	ішкі бірігу
81	insert	қою (жаңа жазба қосу)
82	install	орнату
83	installer	орнатушы
84	join	байланыс
85	key	кілт
86	length	ұзындығы
88	like	сұраныстарда шарт қою үшін пайдаланылатын кілттік сөз like a% - а әріпінен басталатын сөздер
89	manipulation	манипуляция
90	model	модель
91	modify column	бағанды өзгерту
92	navigator	навигатор
93	now	ағымдағы уақыт

94	null	нөлдік мән (мәні жоқ)
95	order by	сұрыптау (реттеу)
96	outer join	сыртқы байланыс
97	output	шығару, нәтиже
98	password	құпиясөз
99	primary key	бастапқы (алғашқы) кілт
100	privileges	артықшылықтар, бірақ оқулық құқықтар деп жазылды
101	procedure	процедура
102	queries	сұраныстар
103	query	сұраныс
104	query editor	сұраныстар редакторы
105	rdbms (relational database management system)	реляциялық деректер қорын басқару жүйесі
106	recommended	қолдануға ұсынылады
107	record	жазба
108	references	сілтемелер
109	relation	байланыстар
110	relational database	реляциялық деректер қоры
111	replace	орын ауыстыру
112	return	қайтару
113	revoke	кері алу
114	rollback	қайтару
115	row	жол
116	rows affected	жолдар өзгертілді
117	savepoint	сақтау нүктесі
118	schemas	схемалар
119	script	скрипт, SQL тілінде жазылған сұраныстар
120	select	таңдау
121	server	сервер
122	set	орнату (мән орнату немесе меншіктеу)
123	set password	құпиясөзді орнату
124	set up	орнату

125	show columns	бағандарды көрсету
126	show databases	деректер қорын көрсету
127	show tables	кестелерді көрсету
128	structured query language	құрылымды сұраныс тілі
129	table	кесте
130	temporary table	уақытша кесте
131	time	уақыт
134	trigger	триггер
135	tuple	кортеж
136	type	тип
137	type and networking	түр және желі
138	unique	ерекше, қайталанбайтын
140	update	жаңарту
141	use	қолдану
142	user	пайдаланушы
143	username	пайдаланушы аты
144	views	ұсыныстар немесе көріністер
146	where	қайда, сұраныстарда шарт қою үшін пайдаланылатын кілттік сөз

SQL сұраныстар

Деректер қорын құру
CREATE DATABASE деректер_қорының_атауы;
CREATE DATABASE [IF NOT EXISTS]
деректер_қорының_атауы;

Деректер қорын пайдалану
USE деректер_қорының_атауы;

Деректер қорын өшіру
DROP DATABASE [IF EXISTS] деректер_қорының_атауы;

Деректер қорын көру
SHOW DATABASES;

Кесте құру
CREATE TABLE кесте_атауы(
өріс1 дерек_типі [атрибуттар],
өріс2 дерек_типі [атрибуттар],
. . .
өрісN дерек_типі [атрибуттар]
);

Бар кестені көшіріп, жаңа кесте құру
CREATE TABLE [IF NOT EXISTS] жаңа_кесте_атауы
SELECT өріс1, өріс2, ..., өрісN
FROM бар_кесте_атауы;
[WHERE шарт];

Бар кестені көшіріп, жаңа кесте құру
CREATE TABLE жаңа_кесте_атауы LIKE original_table;

Бар кестені көшіріп, жаңа кесте құру
INSERT жаңа_кесте_атауы SELECT * FROM original_table;

Кестелердің тізімі
SHOW TABLES;
SHOW TABLES FROM деректер_қорының_атауы;
SHOW TABLES IN деректер_қорының_атауы;

Кестедегі өрістерді көру
SHOW COLUMNS FROM кесте_атауы;

Кестенің өрістерінің сипаттамасы
DESCRIBE|DESC кесте_атауы;

Кестедегі барлық өрістерді және жазбаларды алу
SELECT * FROM кесте_атауы;

Жазбаларды өшіру
TRUNCATE TABLE кесте_атауы;

Кестені өшіру
DROP TABLE кесте_аты;
DROP TABLE [IF EXISTS] кесте_атауы;

Кестені өзгерту
ALTER TABLE кесте_атауы
{ ADD баған_атауы дерек_типі [атрибуттар] |
DROP COLUMN баған_атауы |
MODIFY COLUMN баған_атауы дерек_типі [атрибуттар] |
ALTER COLUMN баған_атауы SET DEFAULT үнсіздік_бойынша_мән |
ADD [CONSTRAINT] шектеу_аты |
DROP [CONSTRAINT] шектеу_аты}

Кестеге бағанды|өрісті қосу
ALTER TABLE кесте_аты
ADD COLUMN баған_атауы;

Кестеден бағанды|өрісті өшіру
ALTER TABLE кесте_аты
DROP COLUMN баған_атауы;

Кестеге шектеуді қосу
ALTER TABLE кесте_аты
ADD CONSTRAINT шектеу_атауы;

Кестеден шектеуді өшіру
ALTER TABLE кесте_аты
DROP CONSTRAINT шектеу_атауы;

Кестенің атауын өзгерту
ALTER TABLE кесте_аты RENAME кестенің_жаңа_атауы

Кестедегі өрістің атауын өзгерту
ALTER TABLE кесте_аты
RENAME баған_атауы ТО жаңа_баған_атауы;

Кестеге жазба қосу
INSERT[INTO] кесте_атауы [(өрістер_тізімі)]
VALUES(мән1, мән2, ... мәнN)

Кестедегі жазбаны өзгерту
UPDATE кесте_атауы
SET өріс1 = жаңа_мән1,
өріс2 = жаңа_мән2,
...
өрісN = жаңа_мән
[WHERE шарт]

Кестеден жазбаны өшіру
DELETE FROM кесте_атауы
[WHERE шарт]

Қайталатын жазбаларды алып тастау
SELECT DISTINCT(өріс_атауы)FROM кесте_атауы;

WHERE операторы
WHERE баған [NOT] IN(мәндер жиыны)

WHERE баған [NOT] BETWEEN бастапқы_мәні AND соңғы_мәні
WHERE баған [NOT] LIKE жол_үлгісі
WHERE баған [NOT] REGEXP тұрақты өрнек
WHERE баған IS NULL;
WHERE баған IS NOT NULL;

Сұрыптау | Өсу реті бойынша
SELECT бағандар FROM кесте_атауы
ORDER BY баған [ASC];

Сұрыптау | Кему реті бойынша
SELECT бағандар FROM кесте_атауы
ORDER BY баған DESC;

LIMIT операторы
SELECT * FROM кесте_атауы
LIMIT n;
n – жазба саны

SELECT * FROM кесте_атауы
LIMIT start, n;
start – жазба нөмірі
n – жазба саны

Жазбалар саны
SELECT COUNT(баған) FROMкесте_атауы;

Ең кіші және үлкен мәндер
SELECT MIN(баған), MAX(баған) FROM кесте_атауы;

Жазбаларды топтау
SELECT бағандар
FROM кесте_атауы
[WHERE шарттар]
GROUP BY баған;

Жазбаларды топтау

```
SELECT бағандар
FROM кесте_атауы
[WHERE шарттар]
GROUP BY баған;
HAVING топтау_үшін_қойылатын_шарт
ORDER BY баған
```

Байланысқан екі кестеден деректерді алу

```
SELECT * FROM кесте1, кесте2
WHERE кесте1.key = кесте2.key;
```

Байланысқан екі кестеден деректерді алу
INNER JOIN

```
SELECT L.*, R.*
FROM L
[INNER] JOIN R
ON L.key = R.key
```

Байланысқан екі кестеден деректерді алу
INNER JOIN

```
SELECT бағандар
FROM кесте1
[INNER] JOIN кесте2
ON шарт1
[[INNER] JOIN кесте3
ON шарт2]
```

Байланысқан екі кестеден деректерді алу
RIGHT JOIN

```
SELECT L.*, R.*
FROM L
RIGHT JOIN R
ON L.key = R.key
```

OUTER JOIN

```
SELECT бағандар
FROM кесте1
   {LEFT|RIGHT} [OUTER] JOIN кесте2 ON шарт1
```

[{LEFT|RIGHT} [OUTER] JOIN кесте3 ON шарт2]...

Ұсыныстарды құру
CREATE [OR REPLACE] VIEW ұсыныс_атауы AS
SELECT өріс1, өріс2, ...
FROM кесте_атауы
[WHERE шарт];

Ұсыныстыөшіру
DROP VIEW [IF EXISTS] ұсыныс_атауы;

Уақытша кесте құру
CREATE TEMPORARY TABLE кесте_атауы (
өріс_1, өріс_2, ...,
);
немесе
CREATE TEMPORARY TABLE кесте_атауы (
сұраныс
);

Уақытша кестені жою

DROP TEMPORARY TABLE кесте_атауы;

Кесте көшірмесін құру (барлық өрістері мен жазбаларын көшіру)
CREATE TABLE жаңа_кесте_атауы
SELECT өріс1, өріс2, өріс3
FROM кесте_атауы
[WHERE шарт];

Кесте көшірмесін құру (өрістерді ғана көшіру)
CREATE TABLE жаңа_кесте_атауы LIKE кесте_атауы;

Бір кестенің жазбаларын басқа кестеге көшіру
INSERT кесте2
SELECT өрістер FROM кесте1

[WHERE шарт];

Сақталатын процедура құру
DELIMITER $$
CREATE PROCEDURE процедура_атауы
[IN параметрлер, OUT параметрлер]
BEGIN
	айнымалыларды_жариялау
	операторлар
END;
$$

Сақталатын процедураны шақыру
CALL процедура_атауы(параметрлер);

Сақталатын процедураны шақыру
DROP PROCEDURE [IF EXISTS] процедура_атауы;

Сақталатын функция құру
DELIMITER $$
CREATE FUNCTION функция_атауы (
параметрлер
)
RETURNS тип
[NOT] DETERMINISTIC
BEGIN
	операторлар
END
$$

Сақталатын функцияны өшіру
DROP FUNCTION [IF EXISTS] функция_атауы;

Барлық сақталатын процедуралар тізімі
SHOW PROCEDURE STATUS;

Сақталатын процедуралар тізімі
SHOW PROCEDURE STATUS [LIKE 'жол үлгісі'|WHERE шарт]

SHOW PROCEDURE STATUS WHERE db='деректер_қорының_атауы'

Триггерді құру
DELIMITER $$
CREATE TRIGGER [TRIGGER_NAME]
[TRIGGER TIME] [TRIGGER EVENT]
ON [TABLE]
FOR EACH ROW
[TRIGGER BODY]
$$;

Триггерді өшіру
DROP TRIGGER [IF EXISTS] триггер_атауы;

Триггерлер тізімі
SHOW TRIGGERS;
SHOW TRIGGERS LIKE 'жол_үлгісі';
SHOW TRIGGERS WHERE шарт;
SHOW TRIGGERS FROM 'деректер_қорының_атауы';
SHOW TRIGGERS FROM 'деректер_қорының_атауы' WHERE table = 'кесте_атауы';
SHOW TRIGGERS IN 'деректер_қорының_атауы';

Пайдаланушы құру
CREATE USER [IF NOT EXISTS] пайдаланушы_аты IDENTIFIED BY 'құпиясөз';

Пайдаланушы құпиясөзін ауыстыру
ALTER USER 'пайдаланушы_аты'@'хост_атауы' IDENTIFIED BY 'жаңа_құпиясөз'
немесе
SET PASSWORD FOR 'пайдаланушы_аты'@'хост_атауы'= PASSWORD ('жаңа_құпия_кілт');

Пайдаланушыны өшіру
DROP USER 'пайдаланушы_аты'@'хост_атауы';

Пайдаланушыға құқықтар беру
GRANT құқықтар_түрі
ON объектілер
TO пайдаланушы_аты;

Пайдаланушы құқықтарын алып тастау
REVOKE құқықтар_тізімі
ON объект
FROM пайдаланушы_аты;

Сервердегі барлық пайдаланушылар тізімі
SELECT USER FROM mysql.user;

Пайдаланылған әдебиеттер тізімі

1) MySQL Documentation https://dev.mysql.com/doc/
2) MySQL Workbench Reference Manual. https://downloads.mysql.com/docs/workbench-en.pdf
3) Сейед Тахагхогхи, Хью Е. Вильямс. Руководство по MySQL / Пер. с англ. — М. : Издательство «Русская редакция» ; 2007. — 544 стр. : ил. ISBN 978–5–7502–0319–2
4) Шварц Б., Зайцев П., Ткаченко В. MySQL по максимуму. 3-е изд. – СПб.: Питер, 2018. – 864 с.: ил. – (Серия «Бестселлеры O'Reilly»). ISBN 978-5-4461-0696-7
5) М. В. Кузнецов, И. В. Симдянов. MySQL на примерах / – СПб.: БХВ-Петербург, 2007. — 592 с.: ил. + CD-ROM. ISBN 978-5-9775-0066-1
6) Виктор Гольцман. MySQL 5.0. Библиотека программиста. Питер; Санкт-Петербург; 2010. ISBN 978-5-49807-135-0
7) Куликов С. Работа с MySQL, MS SQL Server и Oracle в примерах © EPAM Systems, RD Dep, 2018. — 547 с.
8) Гровф, Джеймс Р., Вайнберг, Пол Н., Оппелъ, Эндрю Дж. SQL: полное руководство, 3-е изд. : Пер. с англ. - М.: ООО «И.Д. Вильяме», 2015. - 960 с. : ил. - Парал. тит. англ. ISBN 978-5-8459-1654-9 (рус.)
9) Дюбуа П. MySQL. Сборник рецептов. – Пер. с англ. – СПб: Символ-Плюс, 2006. – 1056 с., ил. ISBN 5-93286-070-7
10) Васвани В. MySQL: использование и администрирование = MySQL Database Usage & Administration. - М.: «Питер», 2011. – 368 с. - ISBN 978-5-459-00264-5.
11) Мотев А. А. Уроки MySQL. Самоучитель. — СПб.: БХВ-Петербург, 2006. — 208 с.: ил. ISBN 5-94157-658-7
12) Joel Murach. Murach's MySQL. 2nd edition. 2015, Mike Murach & Associates, Inc.
13) Цвелой В.А. Разработка баз данных в среде MySQL: учебно-практическое пособие для студентов специальности 230103 «Автоматизированные системы обработки информации и управления» всех форм обучения /– Омск: АНО ВПО «Омский экономический институт», 2012. – 132 с.

14) Кузнецов М.В., Симдянов И.В. MySQL 5. – СПб.: БХВ-Петербург, 2006. – 1024 с.: ил.
15) Люк Веллинг, Лора Томсон. MySQL. Учебное пособие. – М.: Вильямс, 2005.
16) MySQL Cookbook , Third Edition by Paul DuBois Copyright © 2014 Paul DuBois and O'Reilly Media, Inc.. All rights reserved.
17) Allen G. Taylor. SQL For Dummies®, 8th Edition Published by: John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com Copyright © 2013 by John Wiley & Sons, Inc., Hoboken, New Jersey
18) Карпова И.П. Базы данных. Учебное пособие. – Московский государственный институт электроники и математики (Технический университет). – М., 2009.
19) Кузин А.В., Левонисова С.В. Базы данных: учеб. Пособие для студ. Высш.учеб.заведений. – 5-е изд., испр. – М.:Издательский дом «Академия», 2012. – 320 с.
20) Дэлер Эльмар, Харди Дирк, Троссман Хуберт. Базы данных: Учебник / Пер. с немецкого. – Нур-Султан: Фолиант, 2019. – 184 с.
21) Утепбергенов И.Т., Сагындыкова Ш.Н. Ақпараттық жүйелердегі деректер қоры: Оқу құралы (жоғары оқу орындарының «Ақпараттық жүйелер» мамандығы студенттеріне арналған)/ И.Т.Утепбергенов, Ш.Н.Сагындыкова. – Алматы: АЭжБУ, 2016. – 157 б.: кесте – 1, ил. – 35, әдеб. көрсеткіші – 25 атау.
22) Кляйн К., Кляйн Д., Хант Б. SQL. Справочник, 3-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2010. – 656 с., ил. ISBN 9785932861653
23) Алан Бьюли. Изучаем SQL. СПб: Символ-Плюс. 2007. – 308 с.
24) Бейли Л. Б41 Изучаем SQL. — СПб.: Питер, 2012. — 592 с.: ил.
25) Молинаро Э. SQL. Сборник рецептов. – Пер. с англ. – СПб: Символ!Плюс, 2009. – 672 с., ил.

Оқу басылымы

Сланбекова Асылзат Ермановна,
Каменова Шынар Кайкеновна,
Тогисова Акерке Бакитбековна

ДЕРЕКТЕР ҚОРЫНЫҢ НЕГІЗДЕРІ

Оқу құралы

Авторлық түпнұсқадан басылды

Басуға қол қойылған күні 30.06.2023 ж.
Пішімі 60х84 1/16. Ксерокстік қағаз. Times гарнитурасы.
Шартты баспа табағы 16,37 б.т.
Таралымы 70 дана. Тапсырыс № 74.

Академик Е.А. Бөкетов атындағы Қарағанды университеті
100024, Қарағанды қ., Университет к-сі, 28.

Академик Е.А. Бөкетов атындағы Қарағанды университетінің
Баспасында басылып шықты
Тел. (7212) 35-63-16. E-mail: izd_kargu@mail.ru