INDEX

EX.NO:1 **INSTALLATION AND PRACTICE OF SIMPLE MySql**

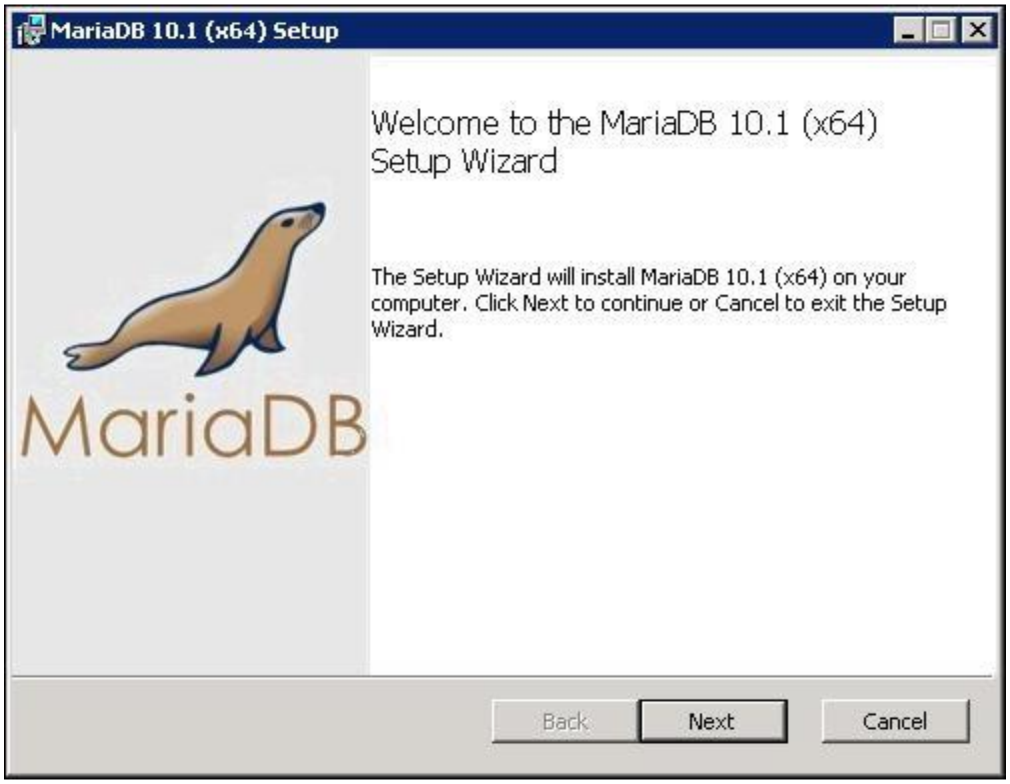
DATE: 28-**0**2-2021 **COMMANDS**

AIM:

To write the procedure to how to install mariaDB in the required system and to keep the snapshots of the screen and connect to mariaDB

PROCEDURE:

1. Download the MSI package from https://downloads.mariadb.org/ location. First click on the series that we want (whatever is the current stable version, most likely), then locate the Windows 64-bit or Windows 32-bit MSI package. For most Windows PCs, the 64-bit MSI package is probably the one that we want, especially if we have more than 4 GB of RAM. If you're unsure, the 32-bit package will work on both 32-bit and 64-bit Windows computers.
2. Once the download has finished, launch the MSI installer by double-clicking on it. Depending on the local Windows settings, you may be promoted to launch the installer automatically. The installer will walk us through installing MariaDB.


MariaDB 10.1 (x64) Setup

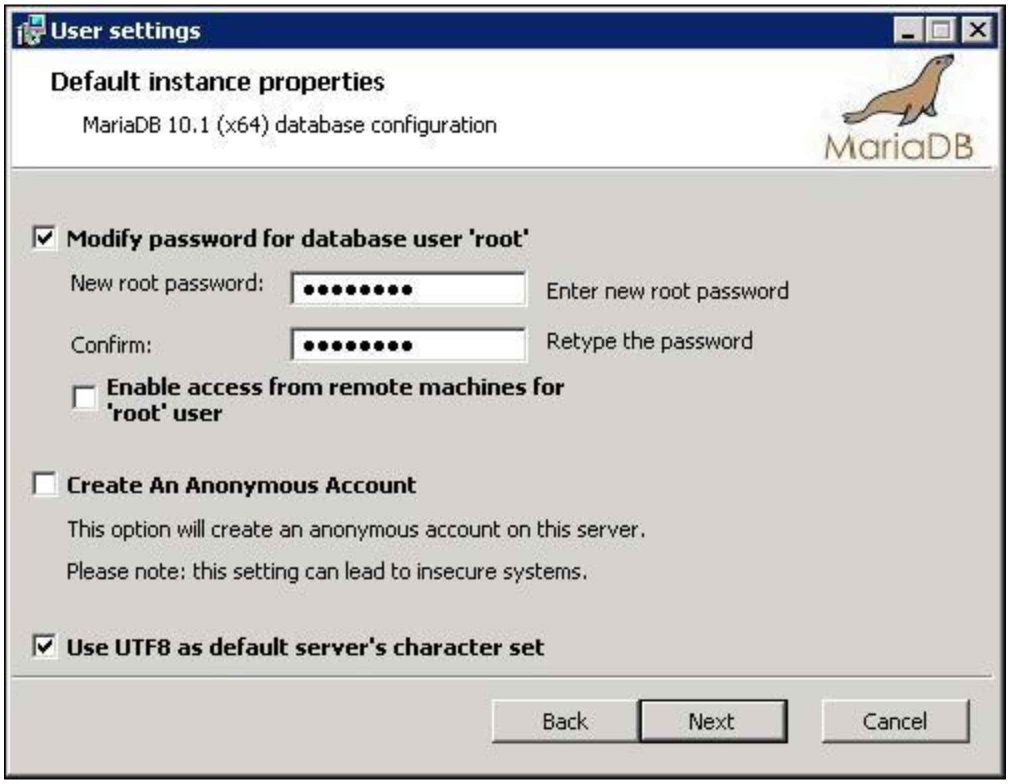
Welcome to the MariaDB 10.1 (x64) Setup Wizard

The Setup Wizard will install MariaDB 10.1 (x64) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

3. If we are installing MariaDB for the first time, we must be sure to set the MariaDB `root` user password when prompted. This is done by checking the **Modify password for database user 'root'** checkbox and then filling in our chosen password two times in the provided textboxes.

4. Unless you need to, don't check the **Enable access from remote machines for 'root' user** or the **Create An Anonymous Account** checkboxes. We'll cover creating regular user accounts in Chapter 4, **Administering MariaDB**.

5. The **Use UTF8 as the default server's character set** checkbox is unchecked by default, but it's a good idea to check it, as shown in the following screenshot:


User settings

Default instance properties

MariaDB 10.1 (x64) database configuration

☑ Modify password for database user 'root'

New root password: ●●●●●●●●● Enter new root password

Confirm: ●●●●●●●●● Retype the password

☐ Enable access from remote machines for 'root' user

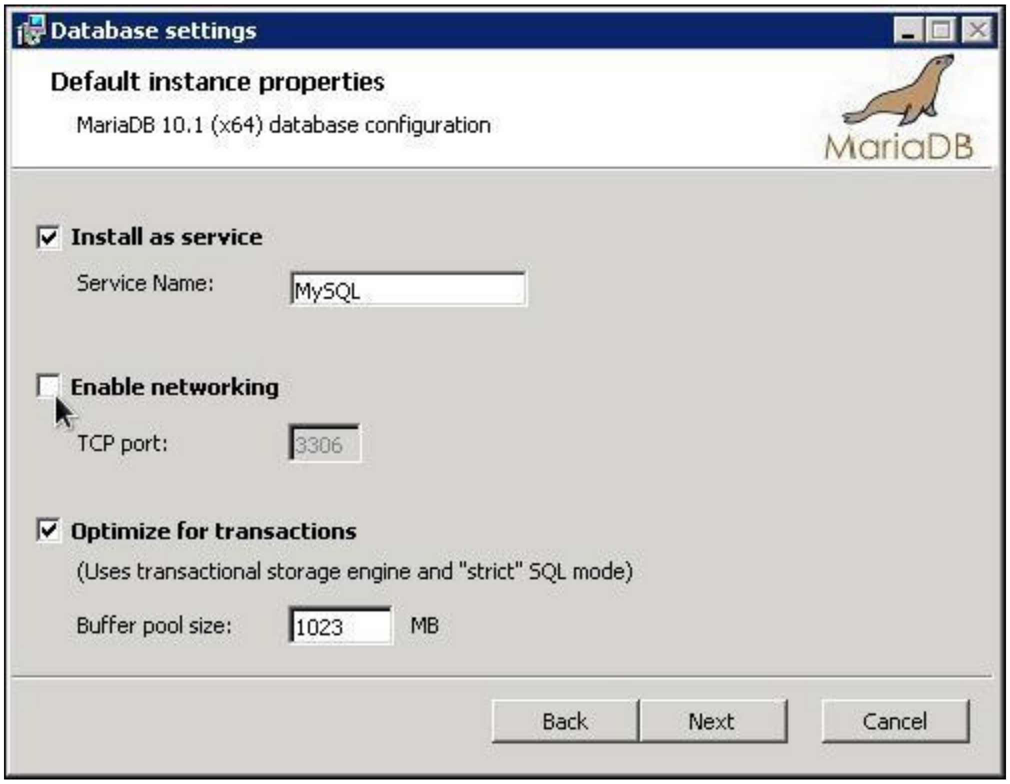
☐ Create An Anonymous Account

This option will create an anonymous account on this server.

Please note: this setting can lead to insecure systems.

☑ Use UTF8 as default server's character set

6. The **Install as service** box is checked by default, and it is recommended to keep it that way so that MariaDB starts up when the computer is booted.
7. The **Service Name** textbox has the default value `MySQL` for compatibility reasons, but we can rename it if we like. This name is what Windows uses to identify the running service, and it does not affect MariaDB so, it is okay to rename or keep it as the default name.
8. Check the **Enable networking** option if you need to access your databases from a different computer. If you don't need remote access, it's best to uncheck this box. As with the service name, there is a default TCP port, number `3306`, which we can change if we want to, but it is usually best to stick with the default unless there is a specific reason not to.
9. The **Optimize for transactions** checkbox is also checked by default. This is the recommended setting, as shown here:


Database settings

Default instance properties

MariaDB 10.1 (x64) database configuration

☑ Install as service

Service Name: MySQL

☐ Enable networking

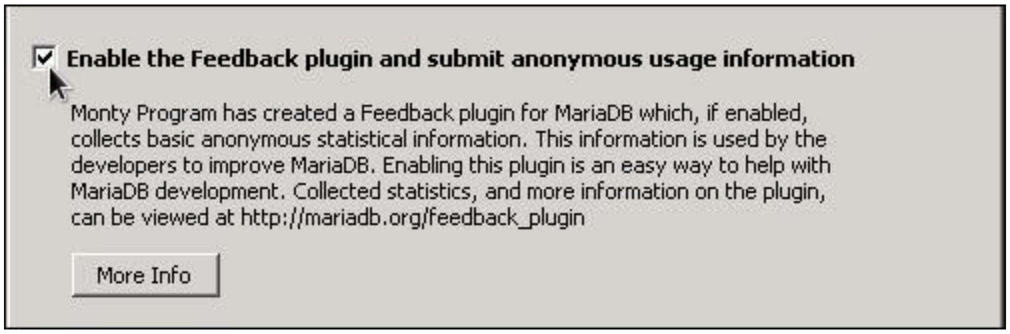
TCP port: 3306

☑ Optimize for transactions

(Uses transactional storage engine and "strict" SQL mode)

Buffer pool size: 1023 MB

10. One easy way to help the MariaDB developers is to check the **Enable the Feedback plugin** checkbox, as shown in the following screenshot. When enabled, the feedback plugin submits anonymous usage information to the MariaDB Foundation. This information includes things such as what plugins are enabled, how much memory MariaDB uses, and the operating system that we are using. MariaDB developers use this information to guide MariaDB development.


☑ Enable the Feedback plugin and submit anonymous usage information

Monty Program has created a Feedback plugin for MariaDB which, if enabled, collects basic anonymous statistical information. This information is used by the developers to improve MariaDB. Enabling this plugin is an easy way to help with MariaDB development. Collected statistics, and more information on the plugin, can be viewed at http://mariadb.org/feedback_plugin

11. There are other settings that we can make through the installer. All of them can be changed later by editing the `my.ini` file. We will be covering this in Chapter 2, **Configuring MariaDB**, so we don't need to worry about them right away.

12. If our version of Windows has user account control enabled, a pop-up window will appear during the installation asking if we want to allow the installer to install MariaDB. For obvious reasons, we will need to click on **Yes**.

13. Once the installation is complete, there will be a `MariaDB` folder added to the start or the programs menu. There will be various links under this, including one to the `mysql` command-line client application.

14. Eventually, we will be presented with a dialog box containing an installation complete message and a **Finish** button. At this point, MariaDB is installed and running on our Windows-based computer. Congratulations! Click on **Finish** to quit the installer.

RESULT: The installation of mariaDB is done and the snapshots of screen are kept.

Exercise No:2 **SIMPLE MySQL COMMANDS**

Date: 05**-0**3**-**2021

AIM:

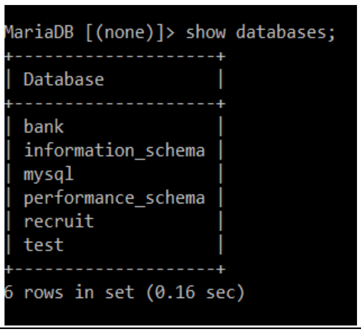
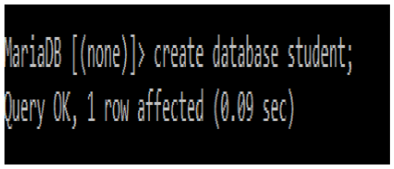
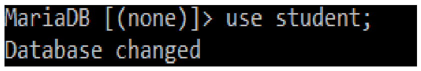
To execute simple MySQL commands by connecting to MariaDB.

SYSTEM REQUIREMENTS:

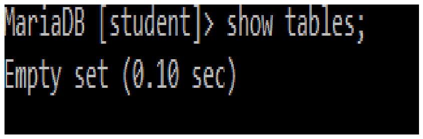
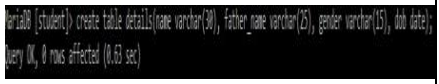
MariaDB version 10.2.12

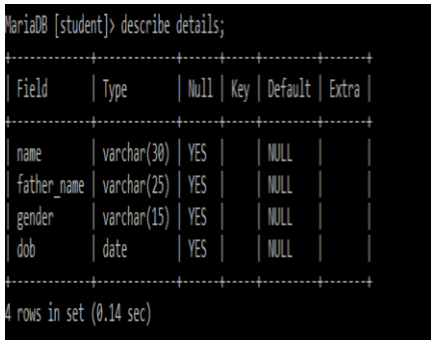
OS Windows (64 bit)

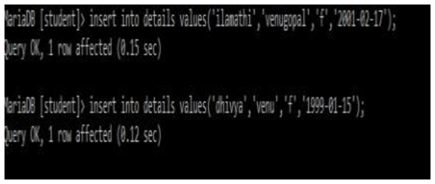
PROCEDURE:

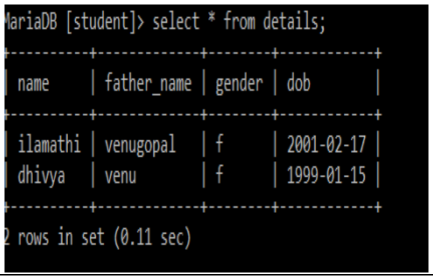
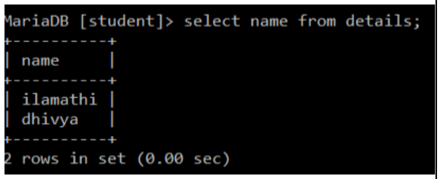
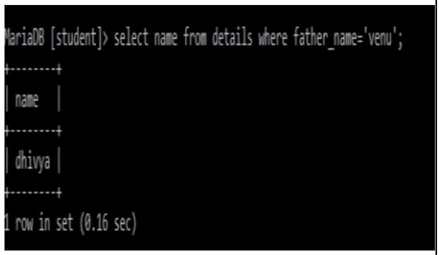
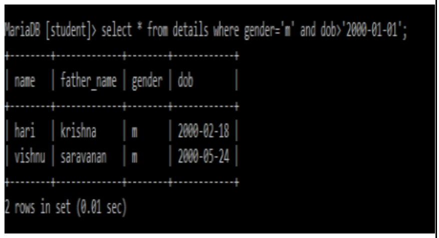
GENERAL SYNTAX	**COMMANDS**	**OUTPUT**
Show{DATABASES| SCHEMAS} LIKE ‘pattern’|WHERE Expr]	Show databases;	 MariaDB [(none)]> show databases; +--------------------+ | Database | +--------------------+ | bank | | information_schema | | mysql | | performance_schema | | recruit | | test | +--------------------+ 6 rows in set (0.16 sec)
Create{database | schema } [if not exists] db_name *create_specification+… Create specification: [default]CHARACTER SET[=] charset_name| [default] collate[=] collation_name| default encryption [=] ,‘y’|’n’-	Create database database_name;	 MariaDB [(none)]> create database student; Query OK, 1 row affected (0.09 sec)
Use db_name	Use database_name;	 MariaDB [(none)]> use student; Database changed

Show[extended] [full] tables [{from|IN}db_name] *LIKE ‘pattern’| WHERE Expr]	Show tables;	

CREATE[TEMPORARY] TABLE[IF NOT EXISTS]tbl_name(create_ Definition,….) [table_options] [partition_options]	CREATE TABLE Table_name (column 1 Datatype, column 2 datatype, column 3 datatype,…..);	 `MariaDB [student]> show tables;` `Empty set (0.10 sec)`
Describe table_name	Describe table_name;	 `MariaDB [student]> create table details(name varchar(30), father_name varchar(25), gender varchar(15), dob date);` `Query OK, 0 rows affected (0.63 sec)`

Insert{low_priority|Delayed|High_priority] [ignore] [into]tbl_name [partition (partition_name[partition Name+…)+ *(COL_name*col_name+…)+ {{VALUES|VALUES} (value_list)*(value_list)+…. Values row_constructor_list}[as row_alias *(col_alias*col_alias+….)++ [ON DUPLICATE KEY UPDATE assignment_list]	Insert into table_name values(value1,value2, value3,…….);	 `MariaDB [student]> describe details;`
		`+-------------+-------------+------+-----+---------+-------+`
		`| Field       | Type        | Null | Key | Default | Extra |`
		`+-------------+-------------+------+-----+---------+-------+`
		`| name        | varchar(30) | YES  |     | NULL    |       |`
		`| father_name | varchar(25) | YES  |     | NULL    |       |`
		`| gender      | varchar(15) | YES  |     | NULL    |       |`
		`| dob         | date        | YES  |     | NULL    |       |`
		`+-------------+-------------+------+-----+---------+-------+`
		`4 rows in set (0.14 sec)`

Select * from orig_tbl;	Select * from table_name;	 `MariaDB [student]> insert into details values('ilamathi','venugopal','f','2001-02-17');`
		`Query OK, 1 row affected (0.15 sec)`
		
		`MariaDB [student]> insert into details values('dhivya','venu','f','1999-01-15');`
		`Query OK, 1 row affected (0.12 sec)`

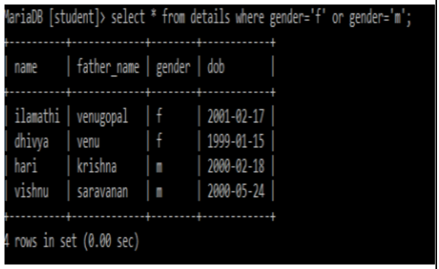
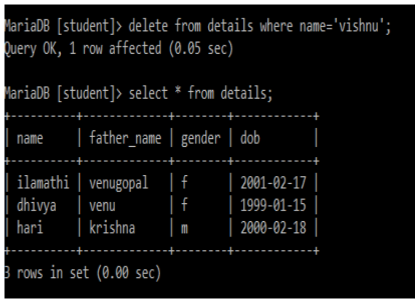
Select column1,column2,… from Tablename	Select column1, column 2, from table_name;	
[all|distinct| distinct row] [HIGH_PRIORITY][STRAIGHT_ JOIN][SQL_SMALL_RESULT] [SQL_BIG_RESULT][SQL_ BUFFER_RESULT][SQL_NO_ CACHE][SQL_CALC_FOUND_ ROWS] select_expr[select_ Expr]..[into_option][from table-references[partition partion_list]][WHERE where_condition]	Select column1, column2… from table_name WHERE condition;	
Select [ALL|DISTINCT|DISTINCT ROW+*HIGH_PRIORITY+… [STRAIGHT_JOIN][SQL_SMALL_ RESULT][SQL_BIG_ RESULT][SQL_BUFFER_RESULT] [SQL_NO_CACHE]	Select column1,column2,.. from table_name where condition1 and condition2 and condition3..	
Select [ALL|DISTINCT|DISTINCT ROW][HIGH_PRIORITY][STRAIGHT JOIN][SQL_SMALL_RESULT][SQL_ BIG_RESULT][SQL_BUFFER_ RESULT]	Select column1,column2,.. from table_name where condition1 OR condition2 OR condition3..	

```
MariaDB [student]> select * from details;
+----------+-------------+--------+------------+
| name     | father_name | gender | dob        |
+----------+-------------+--------+------------+
| ilamathi | venugopal   | f      | 2001-02-17 |
| dhivya   | venu        | f      | 1999-01-15 |
+----------+-------------+--------+------------+
2 rows in set (0.11 sec)
```

```
MariaDB [student]> select name from details;
+----------+
| name     |
+----------+
| ilamathi |
| dhivya   |
+----------+
2 rows in set (0.00 sec)
```

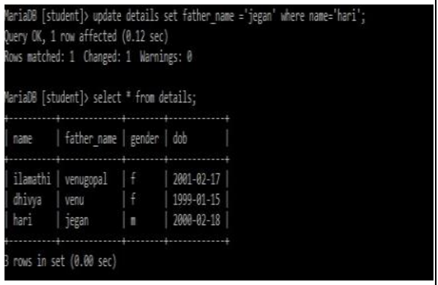
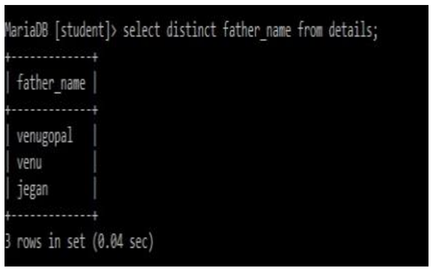
```
MariaDB [student]> select name from details where father_name='venu';
+--------+
| name   |
+--------+
| dhivya |
+--------+
1 row in set (0.16 sec)
```

```
MariaDB [student]> select * from details where gender='m' and dob>'2000-01-01';
+--------+-------------+--------+------------+
| name   | father_name | gender | dob        |
+--------+-------------+--------+------------+
| hari   | krishna     | m      | 2000-02-18 |
| vishnu | saravanan   | m      | 2000-05-24 |
+--------+-------------+--------+------------+
2 rows in set (0.01 sec)
```

[SQL_NO_CACHE][SQL_CALC_ FOUND_ROWS] Select_expr[select_expr][into_option] [from table_references[partition Partition_list]][where where_ Condition1] or condition2 or Condition3		
Select [ALL|DISTINCT|DISTINCT ROW][HIGH_PRIORITY][STRAIGHT JOIN][SQL_SMALL_RESULT][SQL_ BIG_RESULT][SQL_BUFFER_RESULT] [SQL_NO_CACHE][SQL_CALC_ FOUND_ROWS] Select_expr[select_expr][into_option] [from table_references[partition Partition_list]][where where not condition]	Select column1,column2,.. from table_name where not condition	
DELETE [LOW_PRIORITY][QUICK] [IGNORE] FROM tbl_name [[as] Tbl_alias][PARTITION (partition_name[partition_name)] [where where_ condition] *order by…..+*LIMIT	Delete from table_name WHERE Condition;	

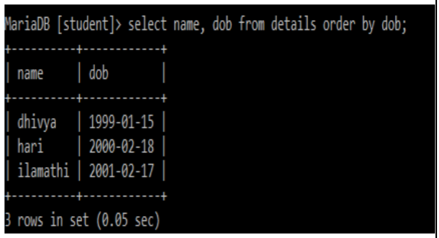
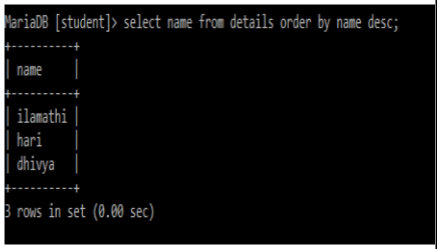
```
MariaDB [student]> select * from details where gender='f' or gender='m';
+----------+-------------+--------+------------+
| name     | father_name | gender | dob        |
+----------+-------------+--------+------------+
| ilamathi | venugopal   | f      | 2001-02-17 |
| dhivya   | venu        | f      | 1999-01-15 |
| hari     | krishna     | m      | 2000-02-18 |
| vishnu   | saravanan   | m      | 2000-05-24 |
+----------+-------------+--------+------------+
4 rows in set (0.00 sec)
```

```
MariaDB [student]> delete from details where name='vishnu';
Query OK, 1 row affected (0.05 sec)

MariaDB [student]> select * from details;
+----------+-------------+--------+------------+
| name     | father_name | gender | dob        |
+----------+-------------+--------+------------+
| ilamathi | venugopal   | f      | 2001-02-17 |
| dhivya   | venu        | f      | 1999-01-15 |
| hari     | krishna     | m      | 2000-02-18 |
+----------+-------------+--------+------------+
3 rows in set (0.00 sec)
```

row_count]		
Update[low_priority][ignore] Table_reference SET assignment_list [WHERE where_condition][ORDER BY….+*LIMIT row_count+	Update table_name set column1= value1, column2= value2,… WHERE condition;	
Select DISTINCT c1,c2,c3 from t1	Select DISTINCT Column1, column2,.. From table_name;	

```
MariaDB [student]> update details set father_name ='jegan' where name='hari';
Query OK, 1 row affected (0.12 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [student]> select * from details;
+----------+-------------+--------+------------+
| name     | father_name | gender | dob        |
+----------+-------------+--------+------------+
| ilamathi | venugopal   | f      | 2001-02-17 |
| dhivya   | venu        | f      | 1999-01-15 |
| hari     | jegan       | m      | 2000-02-18 |
+----------+-------------+--------+------------+
3 rows in set (0.00 sec)
```

```
MariaDB [student]> select distinct father_name from details;
+-------------+
| father_name |
+-------------+
| venugopal   |
| venu        |
| jegan       |
+-------------+
3 rows in set (0.04 sec)
```

Select pk, key_part 1, key_part 2 from t1 ORDER by key_part1, key_ Part2	Select column1, Column2, from table_name order by Column1, column 2,…	 ```MariaDB [student]> select name, dob from details order by dob;``` ```+----------+------------+``` ```| name	dob |``` ```+----------+------------+``` ```| dhivya	1999-01-15 |``` ```| hari	2000-02-18 |``` ```| ilamathi	2001-02-17 |``` ```+----------+------------+``` ```3 rows in set (0.05 sec)```
Select * from t1 order by key_part1 Desc, key_part2 desc;	Select column 1, column 2,.. From table_name Order by column 1, column 2,….DESC;	 ```MariaDB [student]> select name from details order by name desc;``` ```+----------+``` ```| name |``` ```+----------+``` ```| ilamathi |``` ```| hari |``` ```| dhivya |``` ```+----------+``` ```3 rows in set (0.00 sec)```

Result:

Thus the above simple MySQL commands by connecting to MariaDB was executed successfully.

Exercise No:3 **SIMPLE MYSQL COMMANDS**

Date: 12-03-2021 **PRACTISE EXERCISE:1**

AIM:

To write and execute MySQL queries.

SYSTEM REQUIREMENTS:

MariaDB version 10.2.12

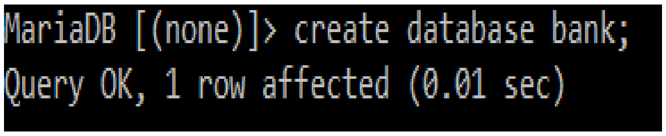
OS Windows (64 bit)

QUERY 1:

Create a bank database.

Create database bank;

OUTPUT:



```
MariaDB [(none)]> create database bank;
Query OK, 1 row affected (0.01 sec)
```

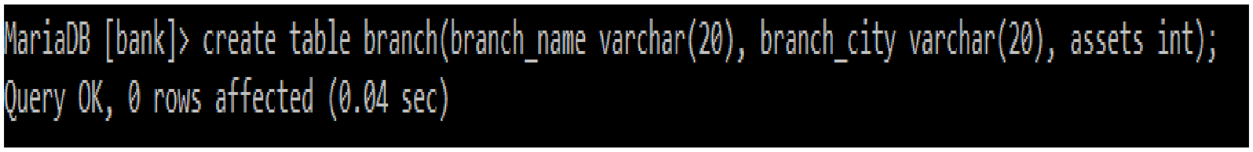
QUERY 2:

Create the following tables

2.1 : Write a SQL statement to create a table branch with column attribute branch_name, branch_city, assets.

Create table branch(branch_name varchar(20), branch_city varchar(20), assets int(20));

OUTPUT:



```
MariaDB [bank]> create table branch(branch_name varchar(20), branch_city varchar(20), assets int);
Query OK, 0 rows affected (0.04 sec)
```

2.2 : Insert values for branch as follows:

Branch relation:

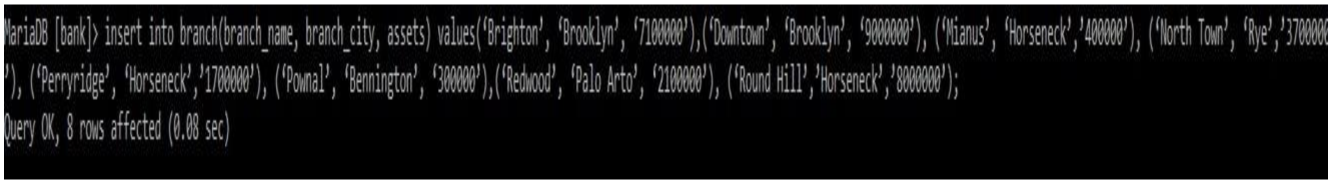
Branch_name	**Branch_city**	**assets**
Brighton	Brooklyn	**7100000**
Downtown	Brooklyn	**9000000**
Mianus	Horseneck	**400000**
North Town	Rye	**3700000**
Perryridge	Horseneck	**1700000**
Pownal	Bennington	**300000**
Redwood	Palo Arto	**2100000**
Round Hill	Horseneck	**8000000**

Insert into branch(branch_name, branch_city, assets) values(‘Brighton’, ‘Brooklyn’, ‘7100000’),

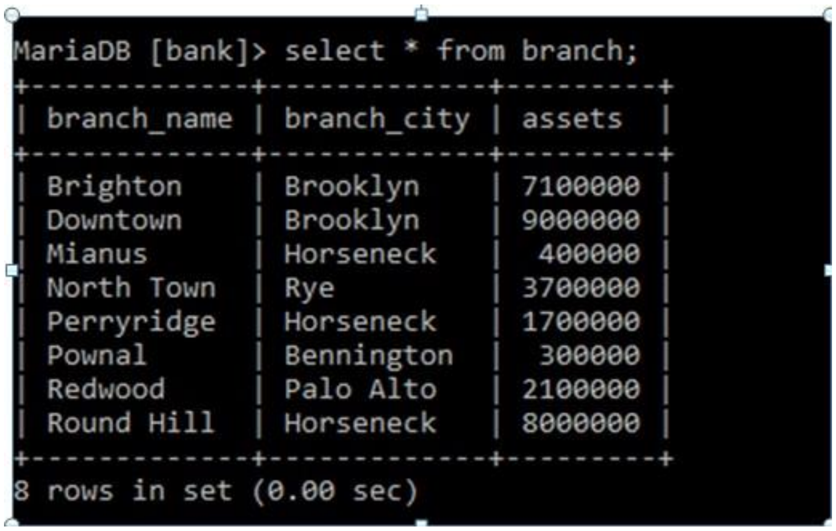
(‘Downtown’, ‘Brooklyn’, ‘9000000’), (‘Mianus’, ‘Horseneck’,’400000’), (‘North Town’, ‘Rye’,’3700000’), (‘Perryridge’, ‘Horseneck’,’1700000’), (‘Pownal’, ‘Bennington’, ‘300000’),

(‘Redwood’, ‘Palo Arto’, ‘2100000’), (‘Round Hill’,’Horseneck’,’8000000’).

OUTPUT:



```
MariaDB [bank]> insert into branch(branch_name, branch_city, assets) values(‘Brighton’, ‘Brooklyn’, ‘7100000’),(‘Downtown’, ‘Brooklyn’, ‘9000000’), (‘Mianus’, ‘Horseneck’,’400000’), (‘North Town’, ‘Rye’,’370000
’), (‘Perryridge’, ‘Horseneck’,’1700000’), (‘Pownal’, ‘Bennington’, ‘300000’),(‘Redwood’, ‘Palo Arto’, ‘2100000’), (‘Round Hill’,’Horseneck’,’8000000’);
Query OK, 8 rows affected (0.08 sec)
```

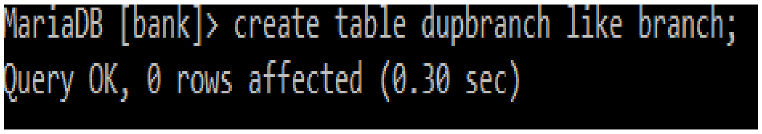


```
MariaDB [bank]> select * from branch;
+-------------+-------------+---------+
| branch_name | branch_city | assets  |
+-------------+-------------+---------+
| Brighton    | Brooklyn    | 7100000 |
| Downtown    | Brooklyn    | 9000000 |
| Mianus      | Horseneck   |  400000 |
| North Town  | Rye         | 3700000 |
| Perryridge  | Horseneck   | 1700000 |
| Pownal      | Bennington  |  300000 |
| Redwood     | Palo Alto   | 2100000 |
| Round Hill  | Horseneck   | 8000000 |
+-------------+-------------+---------+
8 rows in set (0.00 sec)
```

2.3 : Create a duplicate branch table dupbranch using like and check for the non existence of the table.

Create table dupbranch like branch;

OUTPUT:

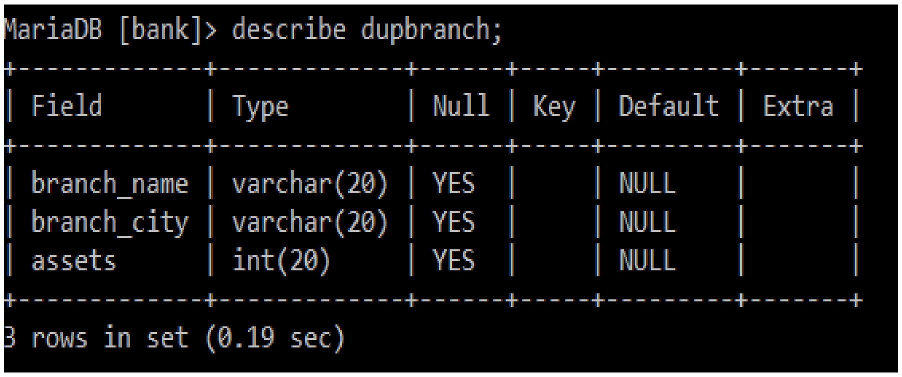


```
MariaDB [bank]> create table dupbranch like branch;
Query OK, 0 rows affected (0.30 sec)
```

2.4 : Check with describe command the structure of table dupbranch.

DESCRIBE dupbranch;

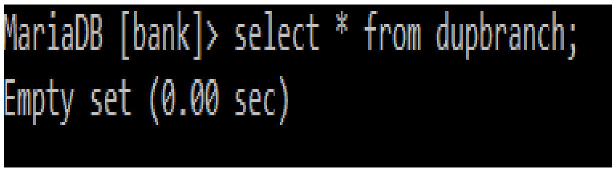
OUTPUT:


```
MariaDB [bank]> describe dupbranch;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| branch_name | varchar(20) | YES  |     | NULL    |       |
| branch_city | varchar(20) | YES  |     | NULL    |       |
| assets      | int(20)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.19 sec)
```

2.5 : Check with the command select * from dupbranch and note down what you observe.

Select * from dupbranch

OUTPUT:


```
MariaDB [bank]> select * from dupbranch;
Empty set (0.00 sec)
```

2.6 : Set not null constraints to branch_name, branch_city, assets.

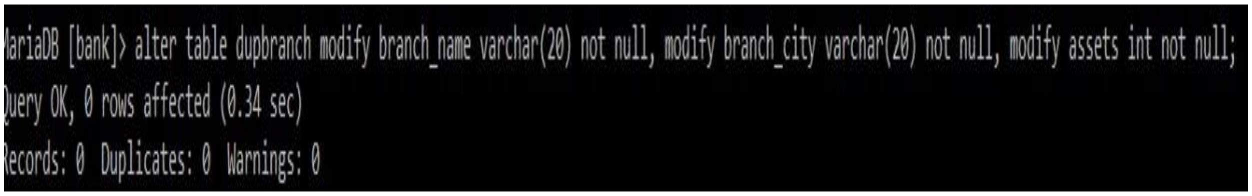
Alter table dupbranch,

modify branch_name varchar(20) not null,

modify branch_city varchar(20) not null,

modify assets int not null;

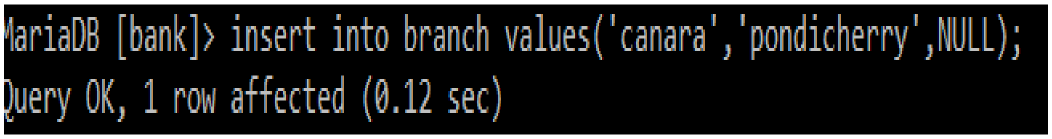
OUTPUT:


```
MariaDB [bank]> alter table dupbranch modify branch_name varchar(20) not null, modify branch_city varchar(20) not null, modify assets int not null;
Query OK, 0 rows affected (0.34 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

2.7 : Try to insert values into branch leaving any one of the above null and not you see as the result.

Insert into branch values('canara','pondicherry',NULL);

OUTPUT:

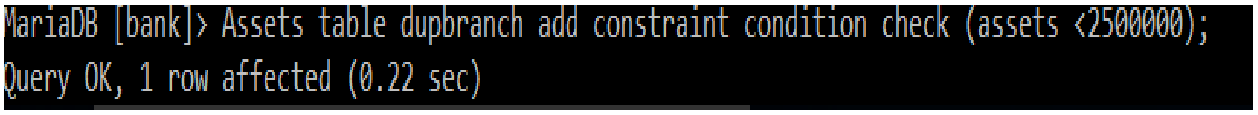


```
MariaDB [bank]> insert into branch values('canara','pondicherry',NULL);
Query OK, 1 row affected (0.12 sec)
```

2.8 : Write a SQL statement to check whether the assets amount exceeding the upper limit 2500000.

Alter table dupbranch add constraint condition check (assets <2500000);

OUTPUT:

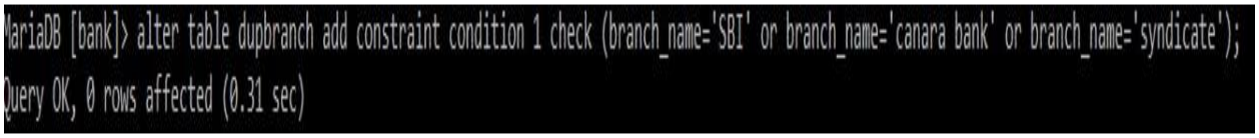


```
MariaDB [bank]> Assets table dupbranch add constraint condition check (assets <2500000);
Query OK, 1 row affected (0.22 sec)
```

2.9 : Now in branch table the branch name should be SBI, canara bank and syndicate bank. No other bank should be entered.

Alter table dupbranch add constraint condition 2 check (branch_name= 'SBI' or branch_name= 'canara bank' or branch_name= 'syndicate');

OUTPUT:

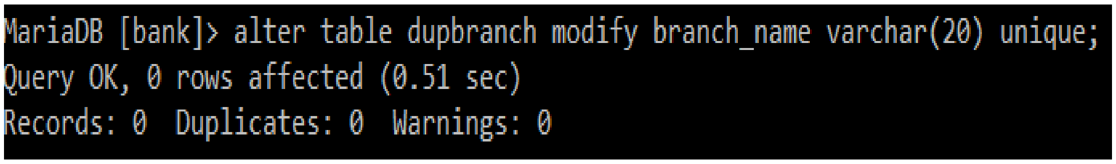


```
MariaDB [bank]> alter table dupbranch add constraint condition 1 check (branch_name='SBI' or branch_name='canara bank' or branch_name='syndicate');
Query OK, 0 rows affected (0.31 sec)
```

2.10 : In the table no branch name should unique and not be repeated.

alter table dupbranch modify branch_name varchar(20) unique;

OUTPUT:

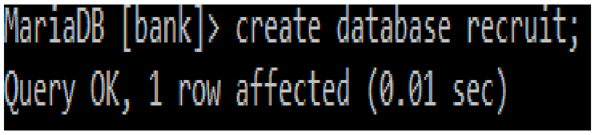


```
MariaDB [bank]> alter table dupbranch modify branch_name varchar(20) unique;
Query OK, 0 rows affected (0.51 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

2.11 : Create new database recruit.

Create database recruit;

OUTPUT:

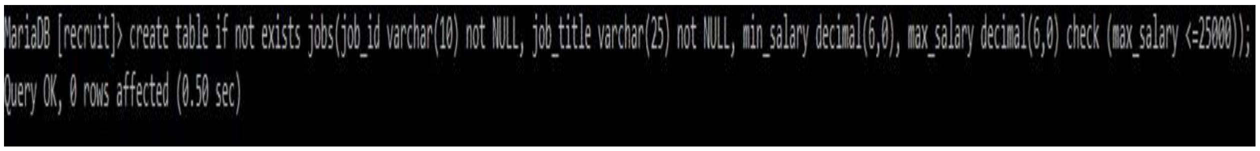


```
MariaDB [bank]> create database recruit;
Query OK, 1 row affected (0.01 sec)
```

2.12 : Write a SQL statement to create a table named jobs including column job_id, job_title, min_salary, max_salary and check whether the max_salary amount execessing the upper limit 25000.

Create table if not exists jobs(job_id varchar(20) not NULL, job_title varchar(25) not NULL, min_salary decimal(6,0), max_salary decimal (6,0) check (max_salary <=25000));

OUTPUT:

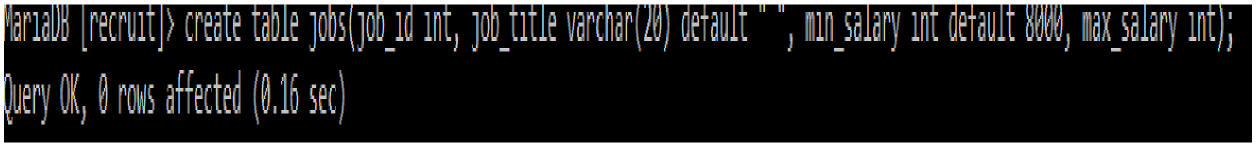


```
MariaDB [recruit]> create table if not exists jobs(job_id varchar(10) not NULL, job_title varchar(25) not NULL, min_salary decimal(6,0), max_salary decimal(6,0) check (max_salary <=25000));
Query OK, 0 rows affected (0.50 sec)
```

2.13 : Write a SQL statement to create a table named jobs including column job_id, job_title, min_salary and max_salary and make sure that, the default value for job_title is blank and min_salary is 8000 andmax_salary is NULL will be entered automatically at the time of insertionif no value assigned for the specified column.

Create table jobs(job_id int, job_title varchar(20) default " ",min_salary int default 8000, max_salary int);

OUTPUT:

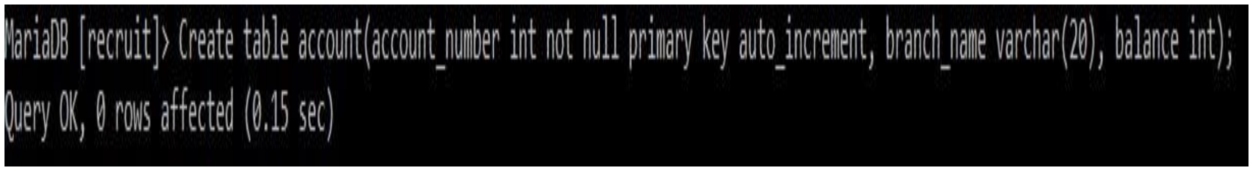


```
MariaDB [recruit]> create table jobs(job_id int, job_title varchar(20) default " ", min_salary int default 8000, max_salary int);
Query OK, 0 rows affected (0.16 sec)
```

2.14 : Create a table account under bank database with attribute account_number, branch_name and balance with account_number as a key and it must be incremented automatically.

Create table account(account_number int not null primary key auto-increment, branch_name varchar(20), balance int);

OUTPUT:

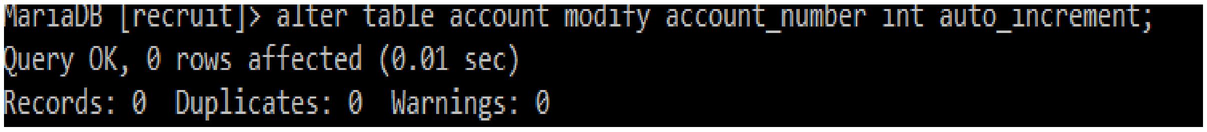


```
MariaDB [recruit]> Create table account(account_number int not null primary key auto_increment, branch_name varchar(20), balance int);
Query OK, 0 rows affected (0.15 sec)
```

2.15 : Write a SQL command to create account table with the following attributes account_number , branch_name and balance with account_number on the key attribute and the account_number should be an automatically incremented value.

Alter table account modify account_no int auto increment;

OUTPUT:

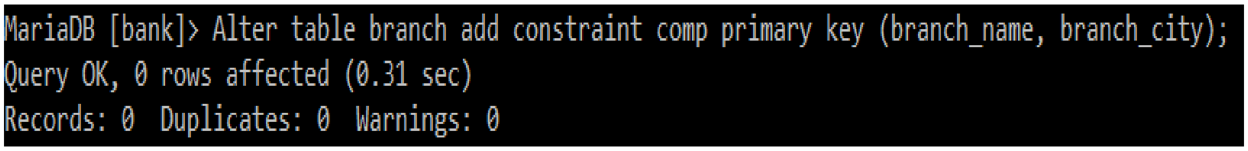


```
MariaDB [recruit]> alter table account modify account_number int auto_increment;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

2.16 : Write a SQL command to set branch_name and branch_city as primary key.

Alter table branch add constraint comp primary key (branch_name, branch_city);

OUTPUT:



```
MariaDB [bank]> Alter table branch add constraint comp primary key (branch_name, branch_city);
Query OK, 0 rows affected (0.31 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Result:

Thus the above MySQL queries was executed successfully.

QUIZ QUESTIONS:

1. What is database?

A database is a collection references of related data. In otherwords a database is an electronic system that allows data to be easily accessed, manipulated and updated. They are managed using a DBMS.

2. Write a short note on what is MariaDB?

MariaDB is an open source relational database management system (DBMS) that is a compatible drop in replacement for the widely used MySQL database technology. The developers devised MariaDB in 2009 in response to Oracle corp's acquisition of MySQL.

3. What is the difference between unique key and primary key?

PRIMARY KEY:

It makes the table row unique(i.e, there can't be 2 row with exact same key)

There can be only one primary key in a database table.

They will not accept NULL values.

UNIQUE KEY:

It makes the table column in a table row unique(i.e no:2 table row may have the same exact value.

They accept one NULL value.

4. Can the structure of the table be altered after creating the table?

Yes, the structure of the table can be altered. The SQL ALTER TABLE command is used to modify the definition of a table by modifying the definition of its column.

Example: ALTER TABLE stud

ADD COLUMN dept VARCHAR(10);

5. What command is used to delete a table from the database?

The command used to delete a table from the database is DROP TABLE table_name;

Example: DROP TABLE stud;

6. How is SQL, MariaDB and MySQL different?

1. MySQL used by few organizations like us Navy, GitHub, Tesla, Netflix, Facebook and Spotify.

While MariaDB used by few organizations like Wikipedia, Google, Gaigslist, Red Hat, Fedora.

2. MySQL is a open source relational database management system(RDBMS) which is similar to other relational databases. Constraints of MySQL uses tables, roles, triggers, views as the core components to work with.

While MariaDB are similar to MySQL and the databases but it permits you to change without needing to change the applications as the data structures and the data will never require changing.

7. In which version of MySQL and which operating system you are using in DBMS lab?

We are using “MySQL 10.2” version of 64 bit in “windows” operating system in DBMS lab.

Exercise no:4 PRACTISE EXERCISE:2

Date: 19-03-2021

AIM:

To write and execute MySQL queries.

SYSTEM REQUIREMENTS:

MariaDB Version 10.2.12

OS Windows (64 bit)

QUERIES:

QUERY1:

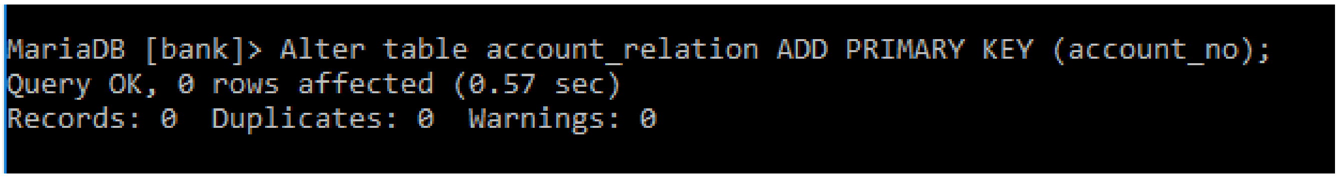
Under bank database create the following tables and insert values using load data command. Set account number as primary key in account table.

Account_number	Branch_name	Balance
A-101	Downtown	500
A-215	Mianus	700
A-102	Perryridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Redwood	700
A-217	Brighton	750

```
Load data local infile ‘c:/dhinesh/account.text’ into table account_relation;
```

```
Alter table account_relation add primary key (account_no);
```

OUTPUT:



```
MariaDB [bank]> Alter table account_relation ADD PRIMARY KEY (account_no);
Query OK, 0 rows affected (0.57 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

QUERY 2:

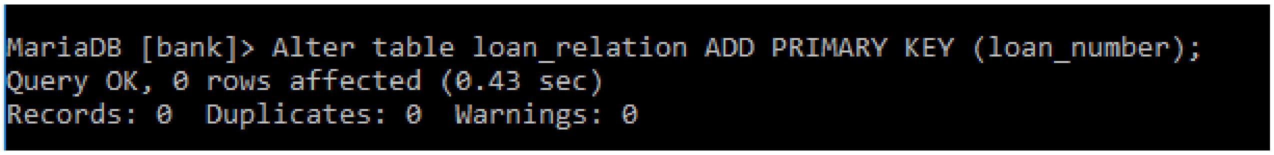
Set Loan number as primary key in table loan.

Loan_number	Branch_name	amount
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

Load data local infile 'c:/dhinesh/loan' into table loan_relation;

Alter table loan_relation add primary key(loan_number);

OUTPUT:



```
MariaDB [bank]> Alter table loan_relation ADD PRIMARY KEY (loan_number);
Query OK, 0 rows affected (0.43 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

QUERY3:

Set loan number of borrower table as foreign key, customer name loan number as primary key of borrower table.

Customer_name	Loan_number
Adame	L-16
Curry	L-93

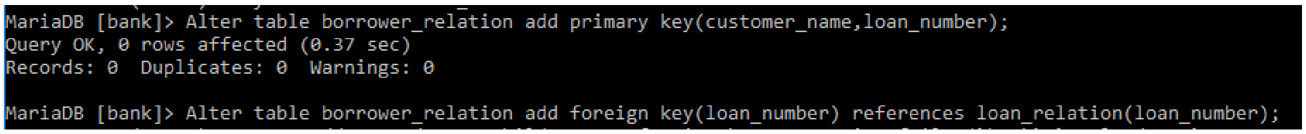
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

Load data local infile ‘c:/borrower.text’ insert into borrower table;

Alter table borrower_relation add foreign key(loan_number) references loan_relation(loan_number);

Alter table borrower_relation add primary key(custumor_name,loan_number);

OUTPUT:



```
MariaDB [bank]> Alter table borrower_relation add primary key(customer_name,loan_number);
Query OK, 0 rows affected (0.37 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [bank]> Alter table borrower_relation add foreign key(loan_number) references loan_relation(loan_number);
```

QUERY 4:

Set account number and customer name of depositor table as primary key and account number of depositor table as foreign key.

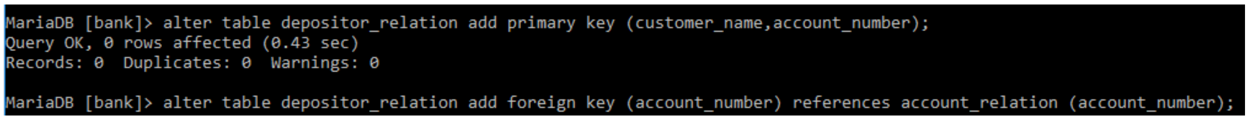
Customer_name	**Account_number**
Hayes	**A-102**
Johnson	**A-101**
Johnson	**A-201**
Jones	**A-217**
Lindsay	**A-222**
Smith	**A-215**
Turner	**A-305**

load data local infile ‘c:/deposit.txt’ insert into depositor_relation;

alter table depositor_relation add primary key (customer_name,account_number);

alter table depositor_relation add foreign key (account_number) references account_relation (account_number);

OUTPUT:



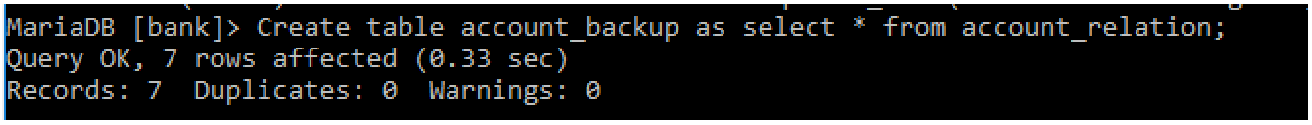
```
MariaDB [bank]> alter table depositor_relation add primary key (customer_name,account_number);
Query OK, 0 rows affected (0.43 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [bank]> alter table depositor_relation add foreign key (account_number) references account_relation (account_number);
```

QUERY 5:

Create a backup of account_table account_backup using “as select”.

Create table account_backup as select * from account_relation;

OUTPUT:



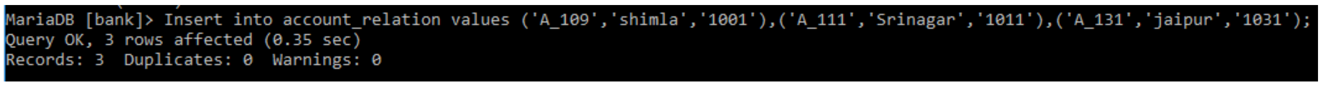
```
MariaDB [bank]> Create table account_backup as select * from account_relation;
Query OK, 7 rows affected (0.33 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

QUERY 6:

Try to insert 3 new accounts into account table using single insert statement.

Insert into account_relation values (‘A_109’,’shimla’,’1001’),(‘A_111’,’Srinagar’,’1011’),(’A_131’,’jaipur’,’1031’);

OUTPUT:



```
MariaDB [bank]> Insert into account_relation values ('A_109','shimla','1001'),('A_111','Srinagar','1011'),('A_131','jaipur','1031');
Query OK, 3 rows affected (0.35 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

QUERY 7:

Write a SQL Statement to change balance of customer whose name is ‘ram’ by 11000.

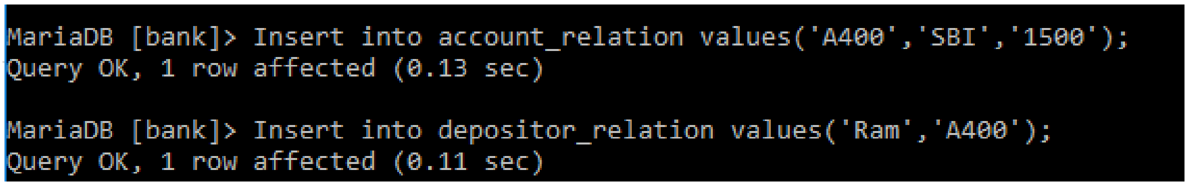
Insert into customer_relation values(‘ram’,’sene’,’bangalore’,’N/A’);

Insert into account_relation values(‘A400’,’SBI’,’1500’);

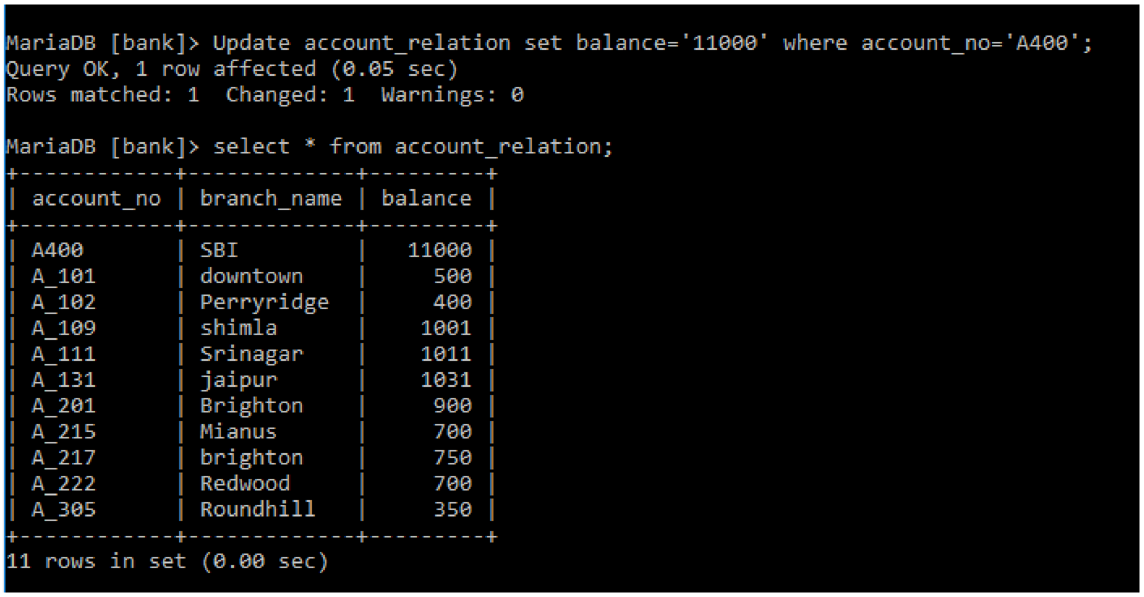
Insert into depositor_relation values(‘Ram’,’A400’);

Update account_relation set balance=’11000’ where account_number=’A400’;

OUTPUT:



```
MariaDB [bank]> Insert into account_relation values('A400','SBI','1500');
Query OK, 1 row affected (0.13 sec)

MariaDB [bank]> Insert into depositor_relation values('Ram','A400');
Query OK, 1 row affected (0.11 sec)
```



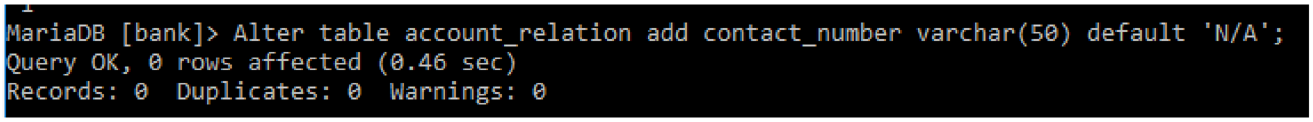
```
MariaDB [bank]> Update account_relation set balance='11000' where account_no='A400';
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [bank]> select * from account_relation;
+------------+-------------+---------+
| account_no | branch_name | balance |
+------------+-------------+---------+
| A400       | SBI         |   11000 |
| A_101      | downtown    |     500 |
| A_102      | Perryridge  |     400 |
| A_109      | shimla      |    1001 |
| A_111      | Srinagar    |    1011 |
| A_131      | jaipur      |    1031 |
| A_201      | Brighton    |     900 |
| A_215      | Mianus      |     700 |
| A_217      | brighton    |     750 |
| A_222      | Redwood     |     700 |
| A_305      | Roundhill   |     350 |
+------------+-------------+---------+
11 rows in set (0.00 sec)
```

QUERY 8:

Insert a new column into contact number in customer table and set for all customers the contact as N/A.

Alter table account_relation add contact_number int(50) default 'N/A';

OUTPUT:



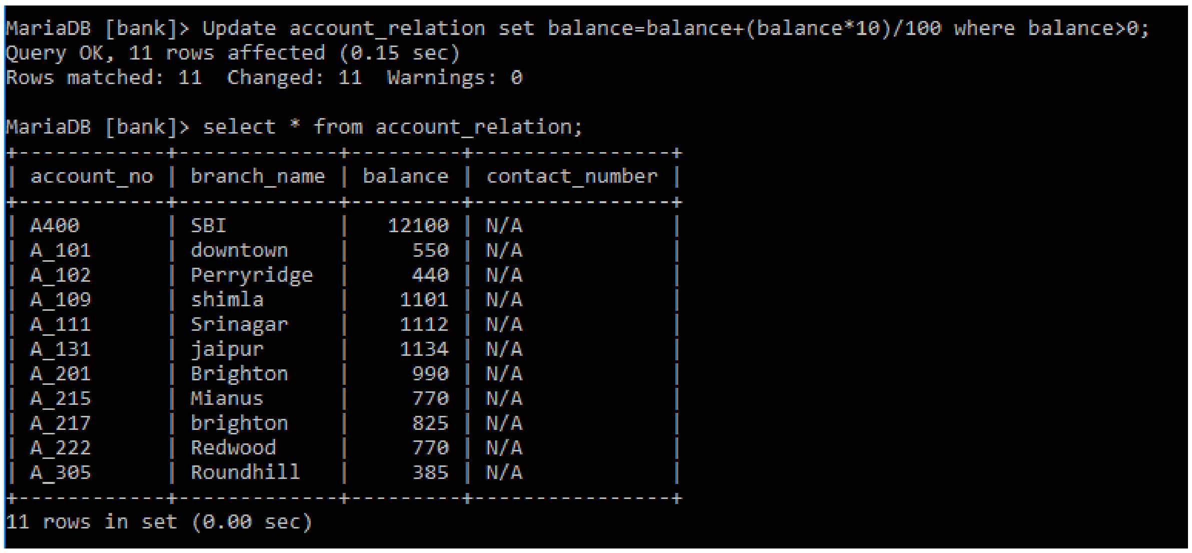
```
MariaDB [bank]> Alter table account_relation add contact_number varchar(50) default 'N/A';
Query OK, 0 rows affected (0.46 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

QUERY 9:

Increase the account balance of all account by 10% of the actual balance.

Update account_relation set balance=balance+(balance*10)/100 where balance>0;

OUTPUT:



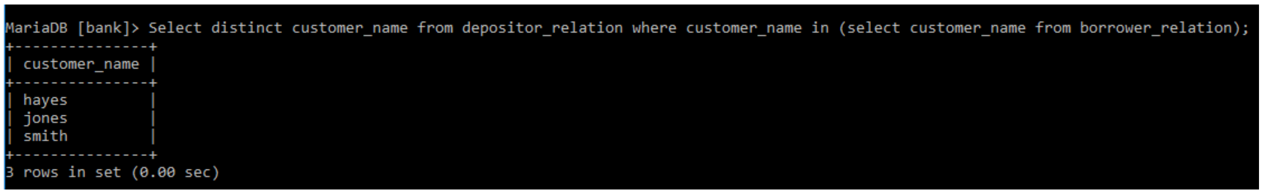
```
MariaDB [bank]> Update account_relation set balance=balance+(balance*10)/100 where balance>0;
Query OK, 11 rows affected (0.15 sec)
Rows matched: 11  Changed: 11  Warnings: 0

MariaDB [bank]> select * from account_relation;
+------------+-------------+---------+----------------+
| account_no | branch_name | balance | contact_number |
+------------+-------------+---------+----------------+
| A400       | SBI         |   12100 | N/A            |
| A_101      | downtown    |     550 | N/A            |
| A_102      | Perryridge  |     440 | N/A            |
| A_109      | shimla      |    1101 | N/A            |
| A_111      | Srinagar    |    1112 | N/A            |
| A_131      | jaipur      |    1134 | N/A            |
| A_201      | Brighton    |     990 | N/A            |
| A_215      | Mianus      |     770 | N/A            |
| A_217      | brighton    |     825 | N/A            |
| A_222      | Redwood     |     770 | N/A            |
| A_305      | Roundhill   |     385 | N/A            |
+------------+-------------+---------+----------------+
11 rows in set (0.00 sec)
```

QUERY 10:

List the customers who have an account and a loan . select distinct customer_name from depositor_relation where costumer_name in (select customer_name from borrower_relation);

Select distinct customer_name from depositor_relation where customer_name in (select customer_name from borrower_relation);

OUTPUT:



```
MariaDB [bank]> Select distinct customer_name from depositor_relation where customer_name in (select customer_name from borrower_relation);
+---------------+
| customer_name |
+---------------+
| hayes         |
| jones         |
| smith         |
+---------------+
3 rows in set (0.00 sec)
```

RESULT: Thus the above MySQL queries was executed successfully.

QUIZ QUESTIONS:

1. What is the difference between Primary key and foreign key?

Primary key	Foreign key
Primary key uniquely identify a record in the tables	Foreign key is a field in the table that primary key in another table.
Primary key can’t accept null values.	Foreign key can accept multiple null value.
By default, primary key is clustered index and data is the database table is physically organized in the sequence of clustered index.	Foreign key do not automatically create an index, clustered or non-clustered. You can manually create an index on foreign key.
We can have only one primary key in a table.	We can have more than one foreign key in a table.

2. What is refrential integrity?

Refrential Integrity(R.I) is a relational database concept, which states that table relationship must always be consistant. In otherwords, any foreign key field must agree with the primary key thatis refluenced by the foreign key. Then, any primary key field changes must be applied to all foreign keys, or not at all the same restriction also applied to foreign keys in that any updates(but not necessarily deletion) must be propagated to the primary parent key.

3. Are the operators BETWEEN and IN allowed in MySQL? If allowed what is the difference?

The operators BETWEEN and IN are allowed in MySQL. The BETWEEN operator selects a range of data between two values. The values can be number, text, etc.

Select * from table_name where column_name between value 1 and value 2.

The IN operator allows you to specify the multiple values.

SYNTAX:

Select * from table_name where column_name is (‘value1’, ‘value2’);

EXERCISE NO:5 PRACTISE EXERCISE:3

DATE: 26-03-2021

AIM:

To write and execute MySQL queries.

SYSTEM REQUIREMENTS:

MariaDB Version 10.2.12

OS Windows (64 bit)

QUERIES:

QUERY1:

Create an employee under the database recruit. Employee(employee_id, first_name, last_name, email, phone_number, hire_date, job_id, salary, commission, manager_id, department_id) Department(department_id, manager_id, department_name). Set primary and foreign key references for employee, department and job table.

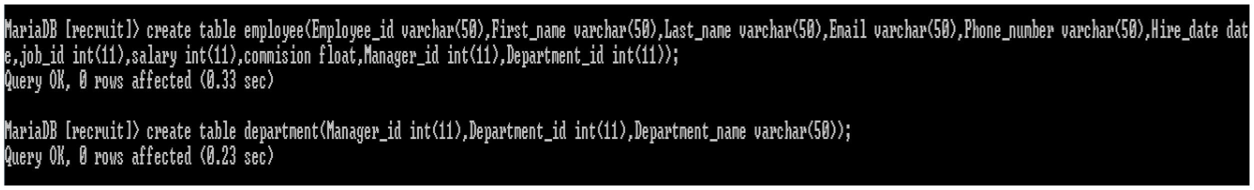
alter table jobs add primary key(Job_id);

alter table employee add primary key(Manager_id,Department_id);

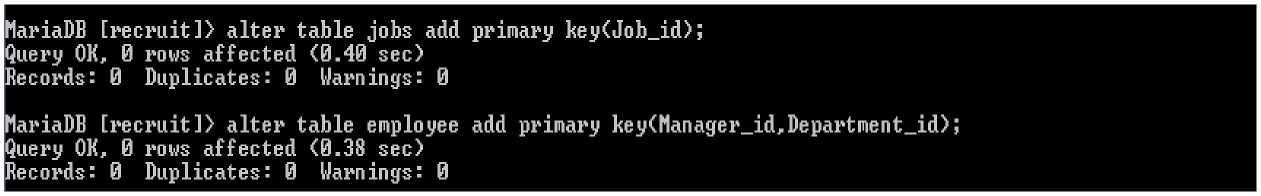
alter table employee add foreign key(Job_id) references jobs(Job_id);

alter table department add foreign key(Manager_id,Department_id) references employee(Manager_id,Department_id);

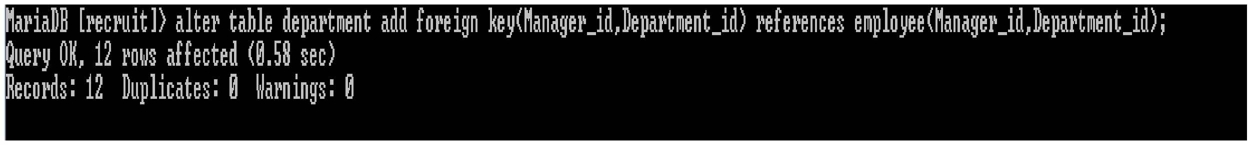
OUTPUT:



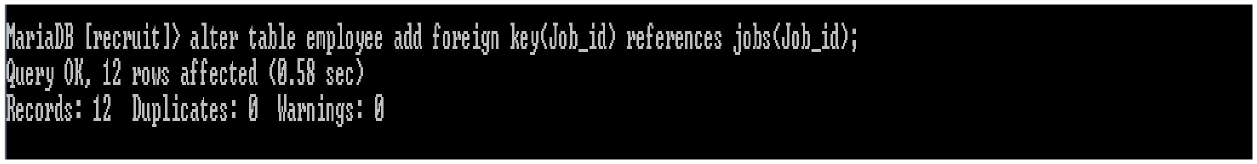
```
MariaDB [recruit]> create table employee(Employee_id varchar(50),First_name varchar(50),Last_name varchar(50),Email varchar(50),Phone_number varchar(50),Hire_date dat
e,job_id int(11),salary int(11),commision float,Manager_id int(11),Department_id int(11));
Query OK, 0 rows affected (0.33 sec)

MariaDB [recruit]> create table department(Manager_id int(11),Department_id int(11),Department_name varchar(50));
Query OK, 0 rows affected (0.23 sec)
```



```
MariaDB [recruit]> alter table jobs add primary key(Job_id);
Query OK, 0 rows affected (0.40 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [recruit]> alter table employee add primary key(Manager_id,Department_id);
Query OK, 0 rows affected (0.38 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



```
MariaDB [recruit]> alter table department add foreign key(Manager_id,Department_id) references employee(Manager_id,Department_id);
Query OK, 12 rows affected (0.58 sec)
Records: 12  Duplicates: 0  Warnings: 0
```



```
MariaDB [recruit]> alter table employee add foreign key(Job_id) references jobs(Job_id);
Query OK, 12 rows affected (0.58 sec)
Records: 12  Duplicates: 0  Warnings: 0
```

QUERY 2:

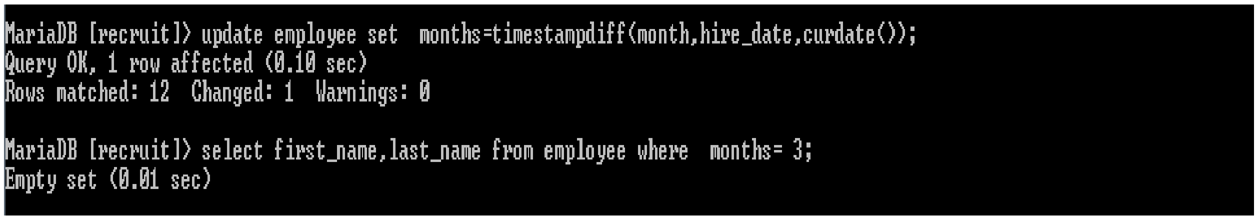
List the employees who have joined 3 months ago from the current date

alter table employee add months int;

update employee set months=timestampdiff(month,hire_date,curdate());

select first_name,last_name from employee where months= 3;

OUTPUT:



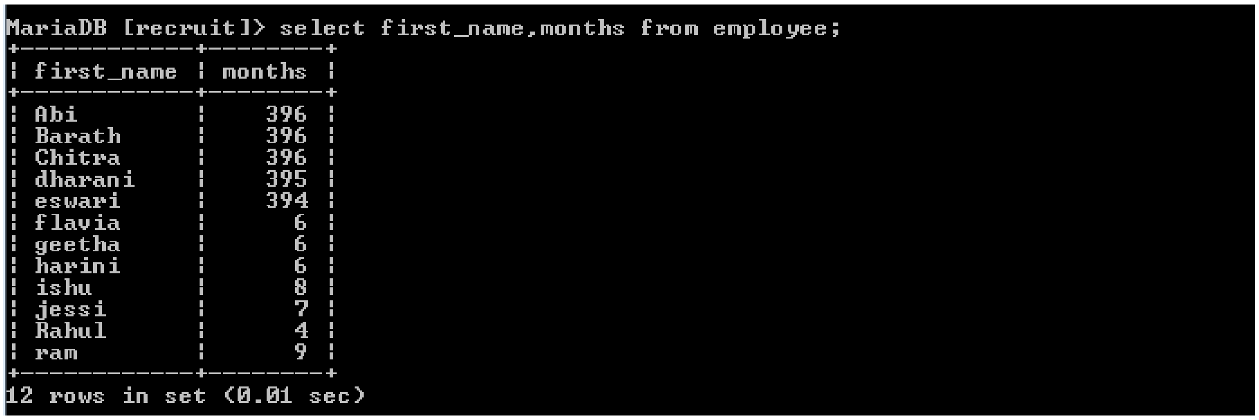
```
MariaDB [recruit]> update employee set  months=timestampdiff(month,hire_date,curdate());
Query OK, 1 row affected (0.10 sec)
Rows matched: 12  Changed: 1  Warnings: 0

MariaDB [recruit]> select first_name,last_name from employee where  months= 3;
Empty set (0.01 sec)
```

QUERY 3:

List the number of months the employees have worked for the organization

select first_name,months from employee;

OUTPUT:



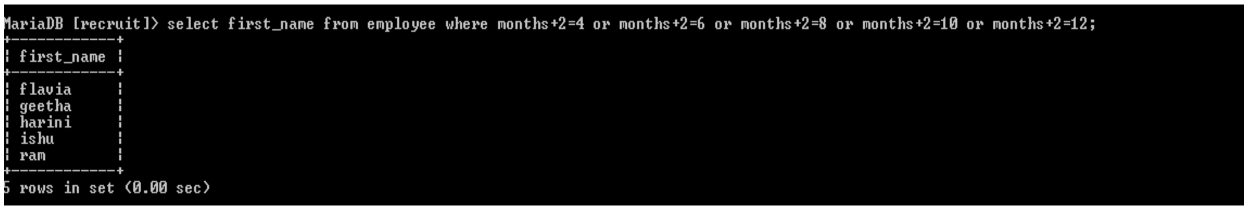
```
MariaDB [recruit]> select first_name,months from employee;
+------------+--------+
| first_name | months |
+------------+--------+
| Abi        |    396 |
| Barath     |    396 |
| Chitra     |    396 |
| dharani    |    395 |
| eswari     |    394 |
| flavia     |      6 |
| geetha     |      6 |
| harini     |      6 |
| ishu       |      8 |
| jessi      |      7 |
| Rahul      |      4 |
| ram        |      9 |
+------------+--------+
12 rows in set (0.01 sec)
```

QUERY 4:

List the employee who will get increment in next two months.

select first_name from employee where months+2=4 or months+2=6 or months+2=8 or months+2=10 or months+2=12;

OUTPUT:



```
MariaDB [recruit]> select first_name from employee where months+2=4 or months+2=6 or months+2=8 or months+2=10 or months+2=12;
+------------+
| first_name |
+------------+
| flavia     |
| geetha     |
| harini     |
| ishu       |
| ram        |
+------------+
5 rows in set (0.00 sec)
```

QUERY 5:

List the employees service length.

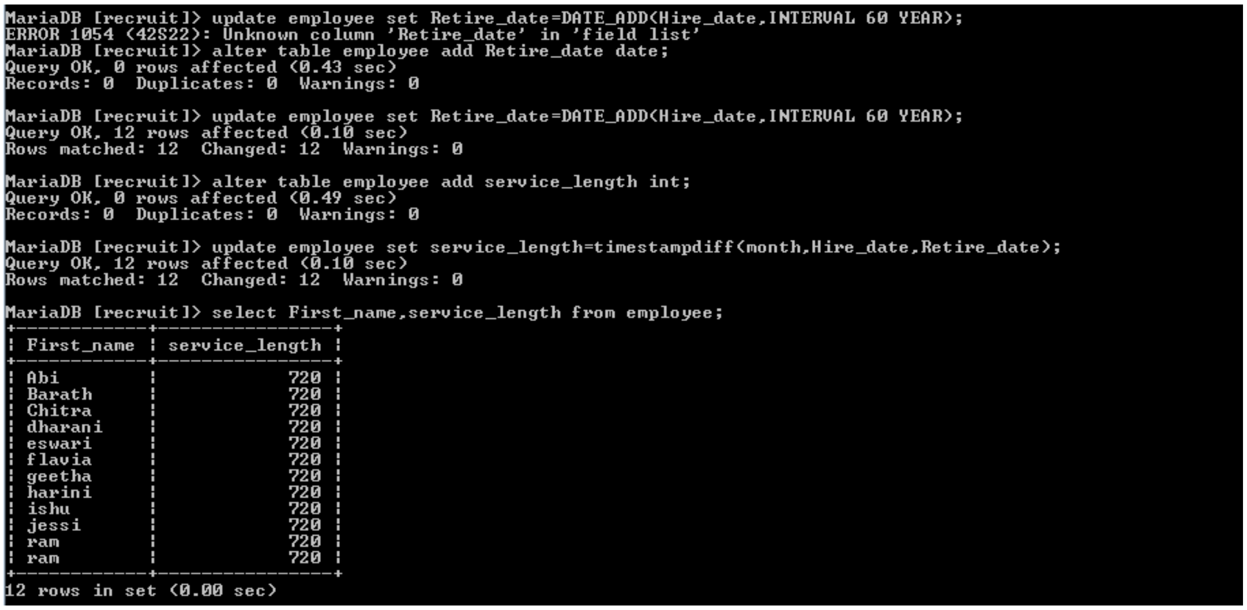
update employee set Retire_date=DATE_ADD(Hire_date,INTERVAL 60 YEAR);

alter table employee add service_length int;

update employee set service_length=timestampdiff(month,Hire_date,Retire_date);

select First_name,service_length from employee;

OUTPUT:



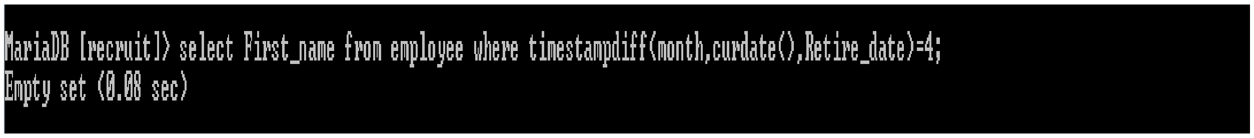
```
MariaDB [recruit]> update employee set Retire_date=DATE_ADD(Hire_date,INTERVAL 60 YEAR);
ERROR 1054 (42S22): Unknown column 'Retire_date' in 'field list'
MariaDB [recruit]> alter table employee add Retire_date date;
Query OK, 0 rows affected (0.43 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [recruit]> update employee set Retire_date=DATE_ADD(Hire_date,INTERVAL 60 YEAR);
Query OK, 12 rows affected (0.10 sec)
Rows matched: 12  Changed: 12  Warnings: 0

MariaDB [recruit]> alter table employee add service_length int;
Query OK, 0 rows affected (0.49 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [recruit]> update employee set service_length=timestampdiff(month,Hire_date,Retire_date);
Query OK, 12 rows affected (0.10 sec)
Rows matched: 12  Changed: 12  Warnings: 0

MariaDB [recruit]> select First_name,service_length from employee;
+------------+----------------+
| First_name | service_length |
+------------+----------------+
| Abi        |            720 |
| Barath     |            720 |
| Chitra     |            720 |
| dharani    |            720 |
| eswari     |            720 |
| flavia     |            720 |
| geetha     |            720 |
| harini     |            720 |
| ishu       |            720 |
| jessi      |            720 |
| ram        |            720 |
| ram        |            720 |
+------------+----------------+
12 rows in set (0.00 sec)
```

QUERY 6:

List the employees who will retire in four months.

select First_name from employee where timestampdiff(month,curdate(),Retire_date)=4;

OUTPUT:



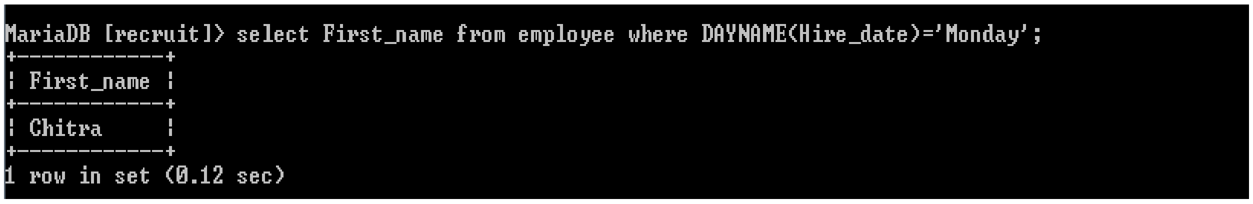
```
MariaDB [recruit]> select First_name from employee where timestampdiff(month,curdate(),Retire_date)=4;
Empty set (0.08 sec)
```

QUERY 7:

List the employees who have joined duty on Monday.

select First_name from employee where DAYNAME(Hire_date)='Monday';

OUTPUT:



```
MariaDB [recruit]> select First_name from employee where DAYNAME(Hire_date)='Monday';
+------------+
| First_name |
+------------+
| Chitra     |
+------------+
1 row in set (0.12 sec)
```

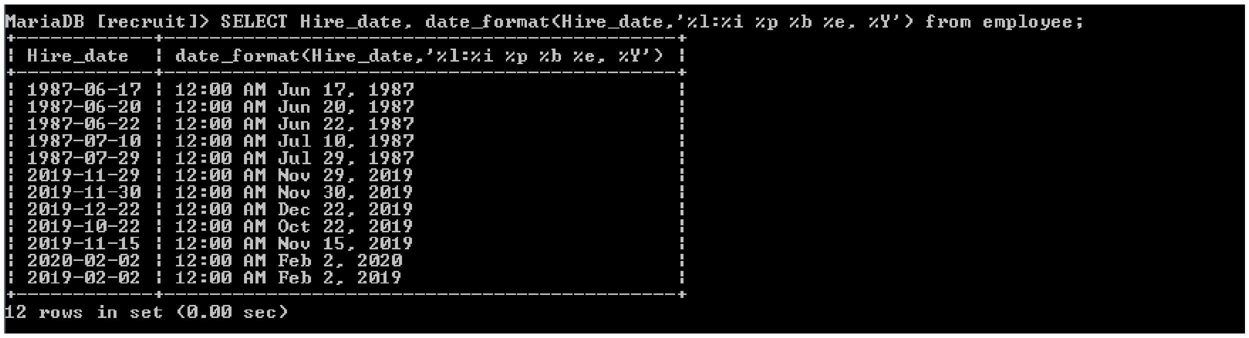
QUERY 8:

Display the hire date in the following formats

•12:00 AM Sep 5, 2014

SELECT Hire_date, date_format(Hire_date,'%l:%i %p %b %e, %Y') from employee;

OUTPUT:

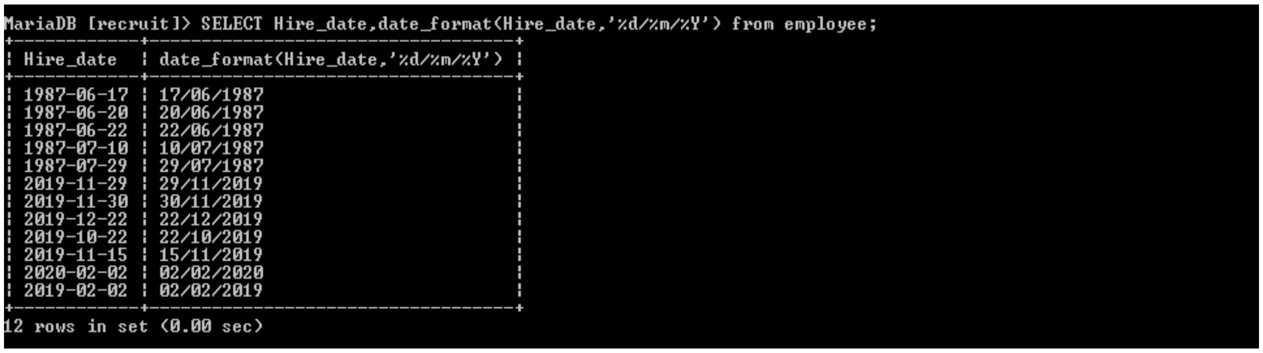


```
MariaDB [recruit]> SELECT Hire_date, date_format(Hire_date,'%l:%i %p %b %e, %Y') from employee;
+------------+-------------------------------------------+
| Hire_date  | date_format(Hire_date,'%l:%i %p %b %e, %Y') |
+------------+-------------------------------------------+
| 1987-06-17 | 12:00 AM Jun 17, 1987                     |
| 1987-06-20 | 12:00 AM Jun 20, 1987                     |
| 1987-06-22 | 12:00 AM Jun 22, 1987                     |
| 1987-07-10 | 12:00 AM Jul 10, 1987                     |
| 1987-07-29 | 12:00 AM Jul 29, 1987                     |
| 2019-11-29 | 12:00 AM Nov 29, 2019                     |
| 2019-11-30 | 12:00 AM Nov 30, 2019                     |
| 2019-12-22 | 12:00 AM Dec 22, 2019                     |
| 2019-10-22 | 12:00 AM Oct 22, 2019                     |
| 2019-11-15 | 12:00 AM Nov 15, 2019                     |
| 2020-02-02 | 12:00 AM Feb 2, 2020                      |
| 2019-02-02 | 12:00 AM Feb 2, 2019                      |
+------------+-------------------------------------------+
12 rows in set (0.00 sec)
```

- **05/09/2014**

SELECT Hire_date,date_format(Hire_date,'%d/%m/%Y') from employee;

OUTPUT:

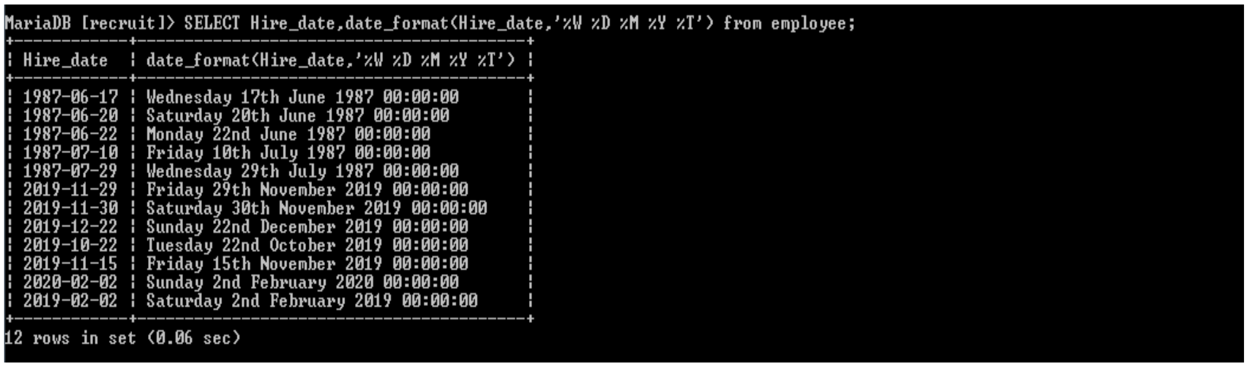


```
MariaDB [recruit]> SELECT Hire_date,date_format(Hire_date,'%d/%m/%Y') from employee;
+------------+------------------------------------+
| Hire_date  | date_format(Hire_date,'%d/%m/%Y') |
+------------+------------------------------------+
| 1987-06-17 | 17/06/1987                         |
| 1987-06-20 | 20/06/1987                         |
| 1987-06-22 | 22/06/1987                         |
| 1987-07-10 | 10/07/1987                         |
| 1987-07-29 | 29/07/1987                         |
| 2019-11-29 | 29/11/2019                         |
| 2019-11-30 | 30/11/2019                         |
| 2019-12-22 | 22/12/2019                         |
| 2019-10-22 | 22/10/2019                         |
| 2019-11-15 | 15/11/2019                         |
| 2020-02-02 | 02/02/2020                         |
| 2019-02-02 | 02/02/2019                         |
+------------+------------------------------------+
12 rows in set (0.00 sec)
```

- **Thursday 4th September 2014 00:00:00**

SELECT Hire_date,date_format(Hire_date,'%W %D %M %Y %T') from employee;

OUTPUT:



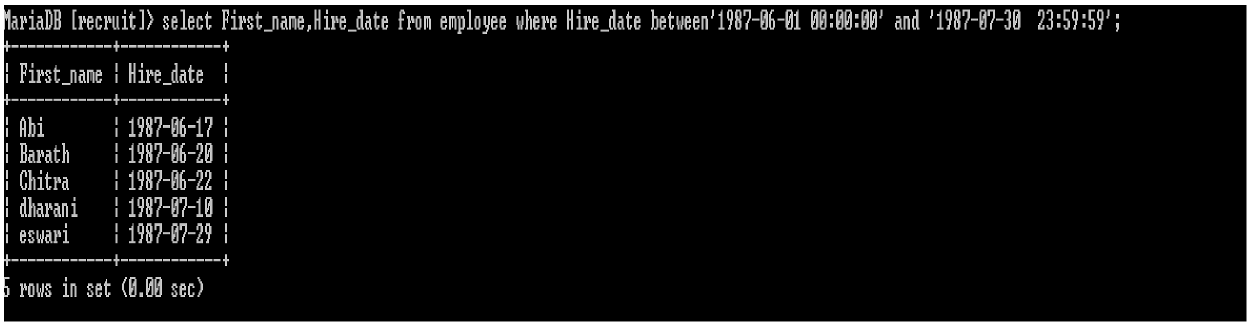
```
MariaDB [recruit]> SELECT Hire_date,date_format(Hire_date,'%W %D %M %Y %T') from employee;
+------------+------------------------------------------+
| Hire_date  | date_format(Hire_date,'%W %D %M %Y %T') |
+------------+------------------------------------------+
| 1987-06-17 | Wednesday 17th June 1987 00:00:00        |
| 1987-06-20 | Saturday 20th June 1987 00:00:00         |
| 1987-06-22 | Monday 22nd June 1987 00:00:00           |
| 1987-07-10 | Friday 10th July 1987 00:00:00           |
| 1987-07-29 | Wednesday 29th July 1987 00:00:00        |
| 2019-11-29 | Friday 29th November 2019 00:00:00       |
| 2019-11-30 | Saturday 30th November 2019 00:00:00     |
| 2019-12-22 | Sunday 22nd December 2019 00:00:00       |
| 2019-10-22 | Tuesday 22nd October 2019 00:00:00       |
| 2019-11-15 | Friday 15th November 2019 00:00:00       |
| 2020-02-02 | Sunday 2nd February 2020 00:00:00        |
| 2019-02-02 | Saturday 2nd February 2019 00:00:00      |
+------------+------------------------------------------+
12 rows in set (0.06 sec)
```

QUERY 9:

Write a query to get the first name and hire date from employees table where hire date between '1987-06-01' and '1987-07-30'.

select First_name,Hire_date from employee where Hire_date between'1987-06-01 00:00:00' and '1987-07-30 23:59:59';

OUTPUT:



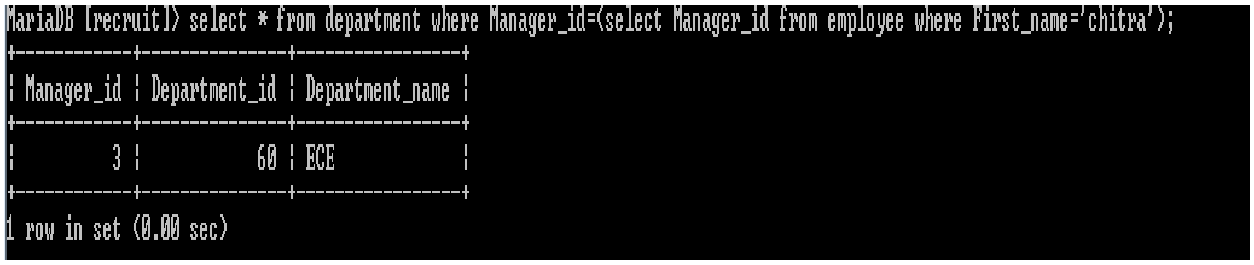
```
lariaDB [recruit]> select First_name,Hire_date from employee where Hire_date between'1987-06-01 00:00:00' and '1987-07-30  23:59:59';
+------------+------------+
| First_name | Hire_date  |
+------------+------------+
| Abi        | 1987-06-17 |
| Barath     | 1987-06-20 |
| Chitra     | 1987-06-22 |
| dharani    | 1987-07-10 |
| eswari     | 1987-07-29 |
+------------+------------+
 rows in set (0.00 sec)
```

QUERY 10:

List the employees with ‘chitra’ their manager and department name.

select * from department where Manager_id=(select Manager_id from employee where First_name='chitra');

OUTPUT:



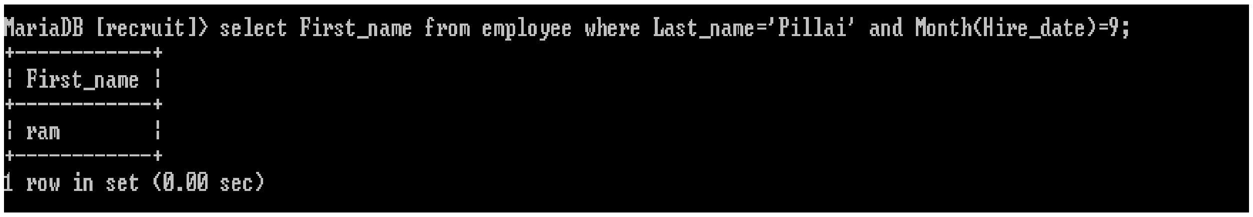
```
MariaDB [recruit]> select * from department where Manager_id=(select Manager_id from employee where First_name='chitra');
+------------+---------------+-----------------+
| Manager_id | Department_id | Department_name |
+------------+---------------+-----------------+
|          3 |            60 | ECE             |
+------------+---------------+-----------------+
1 row in set (0.00 sec)
```

QUERY 11:

List the employee whose name ends with “pillai: and who have joined the production department in the month of September.

select First_name from employee where Last_name='Pillai' and Month(Hire_date)=9;

OUTPUT:



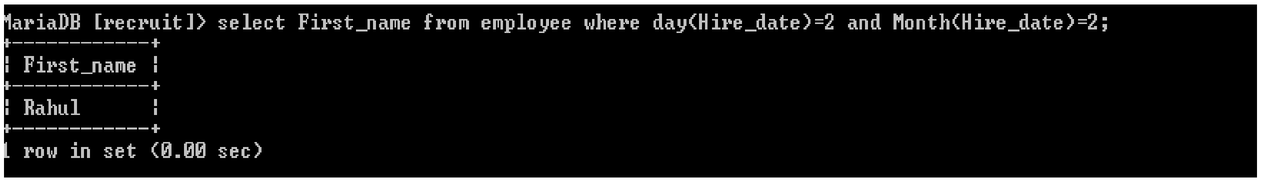
```
MariaDB [recruit]> select First_name from employee where Last_name='Pillai' and Month(Hire_date)=9;
+------------+
| First_name |
+------------+
| ram        |
+------------+
1 row in set (0.00 sec)
```

QUERY 12:

List the employees who have joined on second of February month.

select First_name from employee where day(Hire_date)=2 and Month(Hire_date)=2;

OUTPUT:



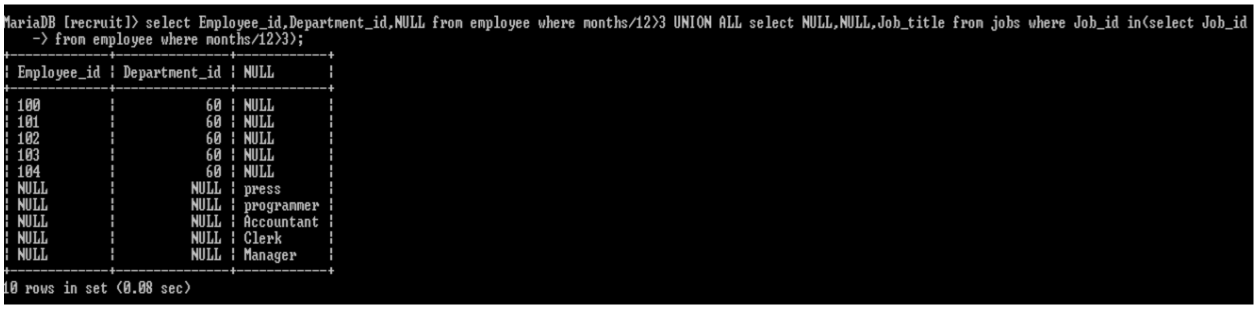
```
MariaDB [recruit]> select First_name from employee where day(Hire_date)=2 and Month(Hire_date)=2;
+------------+
| First_name |
+------------+
| Rahul      |
+------------+
1 row in set (0.00 sec)
```

QUERY 13:

List the employee_id, department_id , job_title who have service greater than 3 years.

select Employee_id,Department_id,NULL from employee where months/12>3 UNION ALL select NULL,NULL,Job_title from jobs where Job_id in(select Job_id from employee where months/12>3);

OUTPUT:



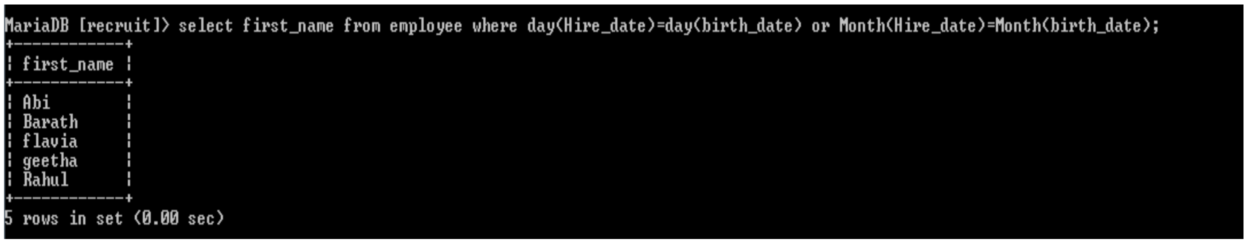
```
MariaDB [recruit]> select Employee_id,Department_id,NULL from employee where months/12>3 UNION ALL select NULL,NULL,Job_title from jobs where Job_id in(select Job_id
    -> from employee where months/12>3);
+-------------+---------------+------------+
| Employee_id | Department_id | NULL       |
+-------------+---------------+------------+
| 100         |            60 | NULL       |
| 101         |            60 | NULL       |
| 102         |            60 | NULL       |
| 103         |            60 | NULL       |
| 104         |            60 | NULL       |
| NULL        |          NULL | press      |
| NULL        |          NULL | programmer |
| NULL        |          NULL | Accountant |
| NULL        |          NULL | Clerk      |
| NULL        |          NULL | Manager    |
+-------------+---------------+------------+
10 rows in set (0.08 sec)
```

QUERY 14:

List the employees who have been hired on their birth dates or birth month.

select first_name from employee where day(Hire_date)=day(birth_date) or Month(Hire_date)=Month(birth_date);

OUTPUT:



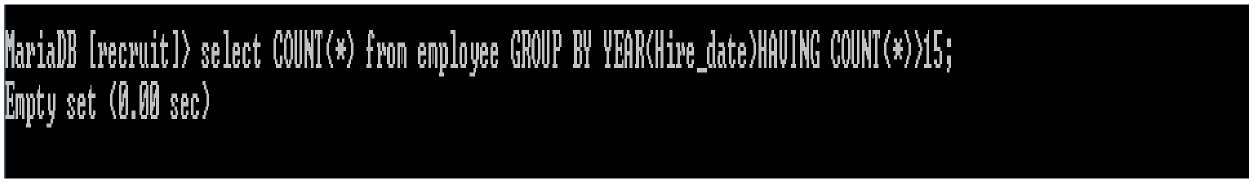
```
MariaDB [recruit]> select first_name from employee where day(Hire_date)=day(birth_date) or Month(Hire_date)=Month(birth_date);
+------------+
| first_name |
+------------+
| Abi        |
| Barath     |
| flavia     |
| geetha     |
| Rahul      |
+------------+
5 rows in set (0.00 sec)
```

QUERY 15:

List the years where more than 15 employees have been hired.

select COUNT(*) from employee GROUP BY YEAR(Hire_date)HAVING COUNT(*)>15;

OUTPUT:



```
MariaDB [recruit]> select COUNT(*) from employee GROUP BY YEAR(Hire_date)HAVING COUNT(*)>15;
Empty set (0.00 sec)
```

RESULT:

Thus the above MySQL queries was executed successfully.

QUIZ QUESTIONS:

1. What is the default port for MySQL server?

The default port for the MySQL server is TCP 3306. The port can also be used for MariaDB database server.

2. Can you elaborate on BLOB and TEXT in MySQL?

A BLOB is a binary large object that can hold a variable amount of data. The four BLOB types are TINYBLOB, BLOB, MEDIUMBLOB AND LONGBLOB. BLOB values are treated as binary strings (byte strings). The maximum theoretical size for BLOB is 2 GB.

Text is the family of column type intended as high-capacity character storage. The four TEXT types are TINYTEXT, MEDIUMTEXT, TEXT, LONGTEXT. The maximum theoretical size is 2 GB. It can contain both single-byte and multiple byte characters.

3. What is the difference between the NVL function, IFNULL function and the ISNULL function? Give examples

NVL function is specific to Oracle server, it replaces null value with other value.

Example: Select nvl(exp1, exp2) it will replace exp1 to exp2 if exp1 is null.

Ifnull function does the same but it is supported only in MySQL server.

Isnull function does the same in SQL server.

4. What datatype you will use to store pictures in database?

BLOB or TEXT- A field with a maximum length of 65535 characters BLOB3 are Binary Large Objects and are used to store large amounts of binary data, such as images or other type of files. For example, a digital file containing a picture video or a song can be stored in a database using BLOB.

5. Write MySQL Queries to illustrate the use of following function.

ADDDATE: ADDDATE() adds a time value with a date.

ADDDATE(date, INTERVAL expr unit);

Here date-date value

INTERVAL-keyword

Expr-a date or datetime expression.

ADDTIME:

ADDTIME() returns a time or datetime after adding a time value with time or datetime.

ADDTIME(expr1, expr2);

MINUTE:

MINUTE() returns a minute from time or datetime value.

MINUTE(time 1);

Where time 1 is time.

DATE:

The DATE() extracts the date part from a datetime expression.

DATE(expression);

MONTH:

The MONTH() returns the month part for a given date

MONTH(date);

EXERCISE NO:06 PRACTISE EXERCISE-04

DATE: 03-07-2021

AIM:

To write and execute MySQL queries.

SYSTEM REQUIREMENTS:

MariaDB Version 10.2.12

OS Windows (64 bit)

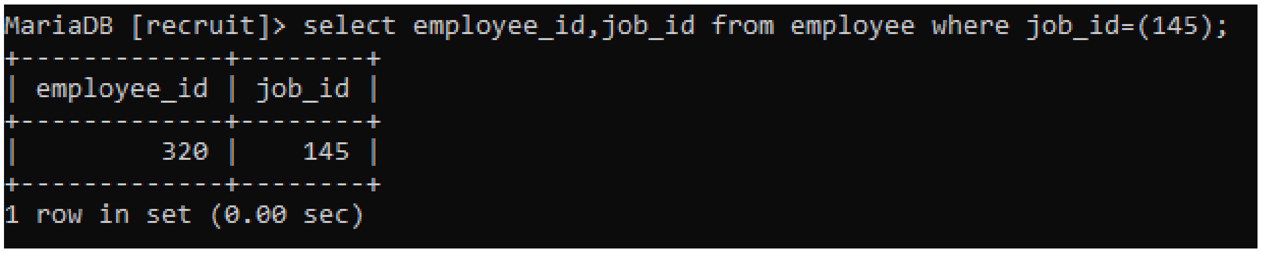
QUERIES:

QUERY 1:

List the employee Id who work under a particular jobID.

MariaDB [recruit]> select employee_id,job_id from employee where job_id=(145);

OUTPUT:



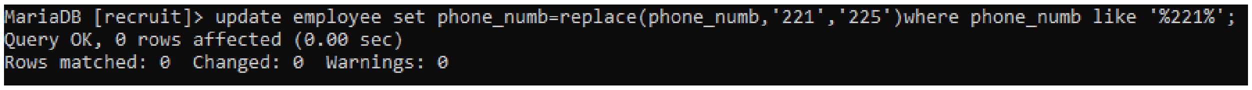
```
MariaDB [recruit]> select employee_id,job_id from employee where job_id=(145);
+-------------+--------+
| employee_id | job_id |
+-------------+--------+
|         320 |    145 |
+-------------+--------+
1 row in set (0.00 sec)
```

QUERY2:

Write a query to update the portion of phone-number in the employees table, within the phone number the substring ‘221’ will be replaced by ‘225’.

MariaDB [recruit]> update employee set phone_number= replace(phone_number,'221','225')wherephone_number like '%221%';

OUTPUT:



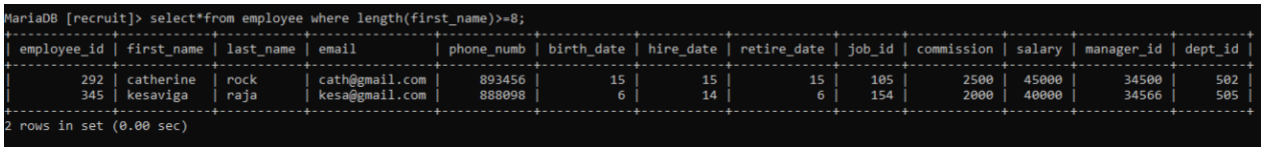
```
MariaDB [recruit]> update employee set phone_numb=replace(phone_numb,'221','225')where phone_numb like '%221%';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

QUERY 3:

Write a query to get the details of employee where firstname is the longest.

MariaDB [recruit]> select*from employee where length(first_name)>=8;

OUTPUT:



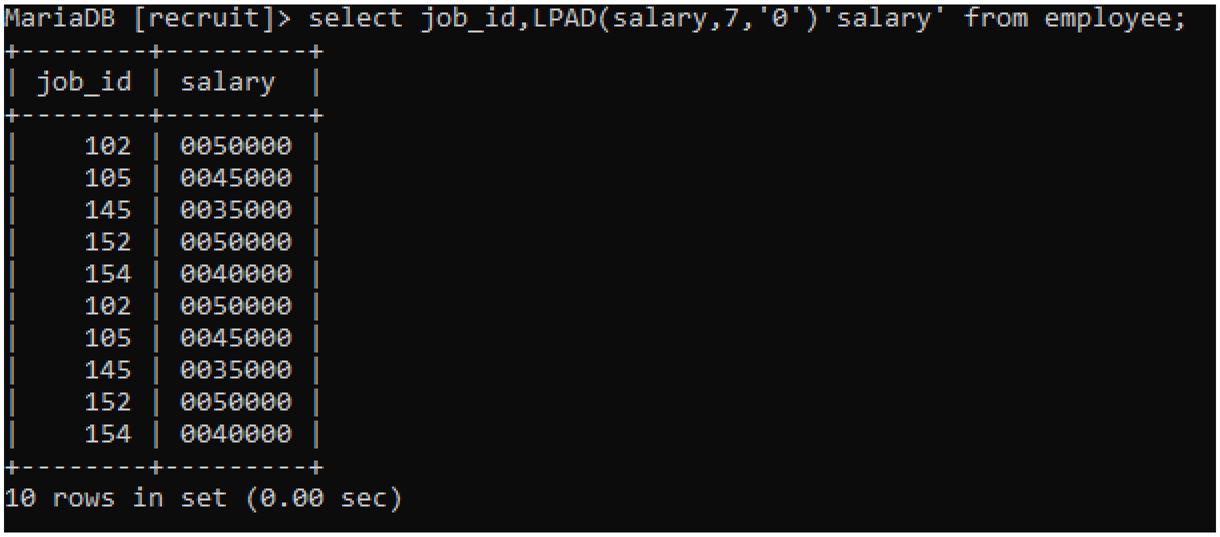
```
MariaDB [recruit]> select*from employee where length(first_name)>=8;
+-------------+------------+-----------+----------------+------------+------------+-----------+-------------+--------+------------+--------+------------+---------+
| employee_id | first_name | last_name | email          | phone_numb | birth_date | hire_date | retire_date | job_id | commission | salary | manager_id | dept_id |
+-------------+------------+-----------+----------------+------------+------------+-----------+-------------+--------+------------+--------+------------+---------+
|         292 | catherine  | rock      | cath@gmail.com |     893456 |         15 |        15 |          15 |    105 |       2500 |  45000 |      34500 |     502 |
|         345 | kesaviga   | raja      | kesa@gmail.com |     888098 |          6 |        14 |           6 |    154 |       2000 |  40000 |      34566 |     505 |
+-------------+------------+-----------+----------------+------------+------------+-----------+-------------+--------+------------+--------+------------+---------+
2 rows in set (0.00 sec)
```

QUERY 4:

Write a query to display leading zeroes before maximum and minimum salary.

MariaDB *recruit+> select job_id, LPAD(max_salary,7,’0’)’salary’ from employee;

OUTPUT:

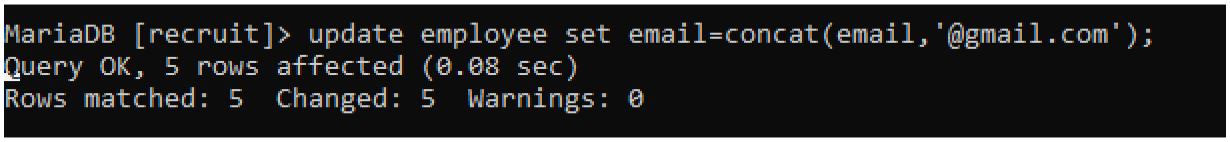

```
MariaDB [recruit]> select job_id,LPAD(salary,7,'0')'salary' from employee;
+--------+---------+
| job_id | salary  |
+--------+---------+
|    102 | 0050000 |
|    105 | 0045000 |
|    145 | 0035000 |
|    152 | 0050000 |
|    154 | 0040000 |
|    102 | 0050000 |
|    105 | 0045000 |
|    145 | 0035000 |
|    152 | 0050000 |
|    154 | 0040000 |
+--------+---------+
10 rows in set (0.00 sec)
```

QUERY 5:

Write a query to append ‘@gmail.com’ to email field.

MariaDB [recruit]> update employee set email=concat(email,'@gmail.com');

OUTPUT:

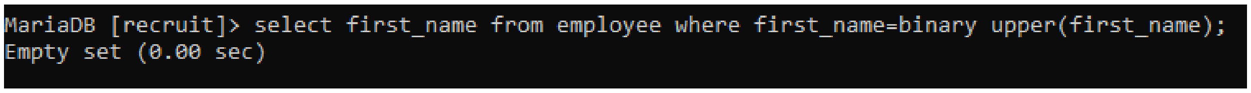

```
MariaDB [recruit]> update employee set email=concat(email,'@gmail.com');
Query OK, 5 rows affected (0.08 sec)
Rows matched: 5  Changed: 5  Warnings: 0
```

QUERY 6:

Write a query to find all employees where first names are in upper case.

MariaDB [recruit]> select first_name from employee where first_name=binary upper(first_name);

OUTPUT:

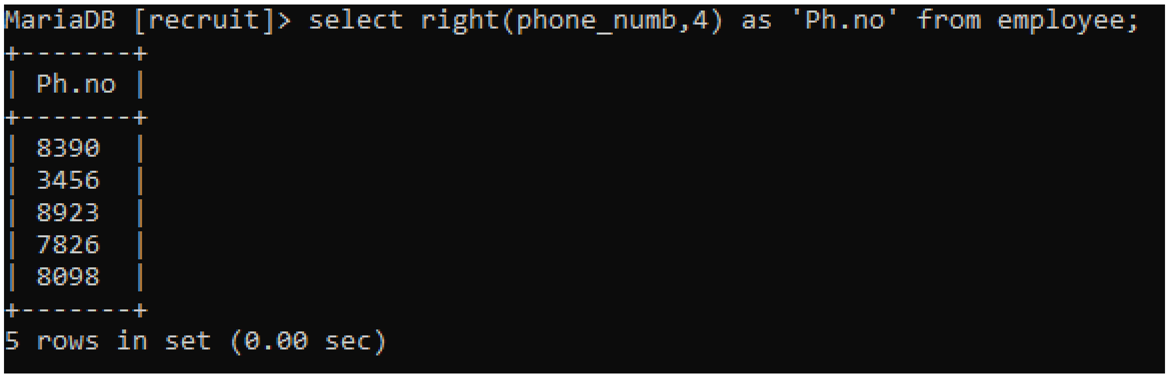

```
MariaDB [recruit]> select first_name from employee where first_name=binary upper(first_name);
Empty set (0.00 sec)
```

QUERY 7:

Write a query to extract last 4 characters of phone number.

MariaDB [recruit]> select right(phone_number,4) as 'Ph.no' from employee;

OUTPUT:

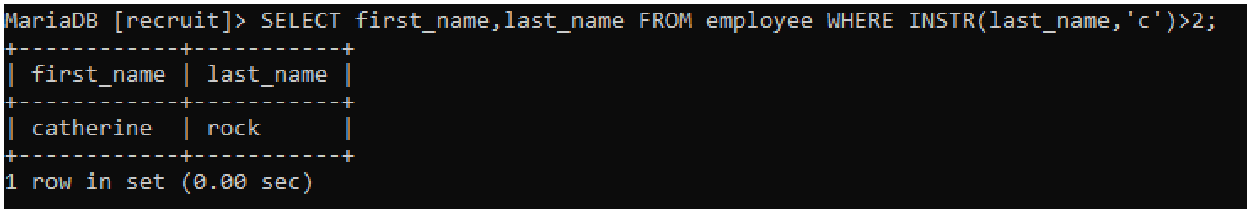

```
MariaDB [recruit]> select right(phone_numb,4) as 'Ph.no' from employee;
+-------+
| Ph.no |
+-------+
| 8390  |
| 3456  |
| 8923  |
| 7826  |
| 8098  |
+-------+
5 rows in set (0.00 sec)
```

QUERY 8:

Write a query to display the length of firstname for employee where last name contain character c after 2nd position.

MariaDB [recruit]> SELECT first_name, last_name FROM employee WHERE INSTR(last_name,'C') > 2;

OUTPUT:

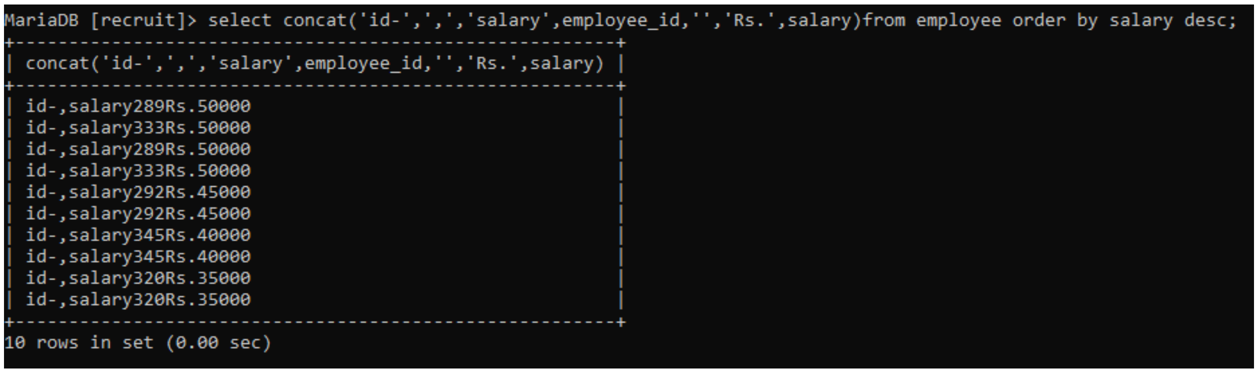

```
MariaDB [recruit]> SELECT first_name,last_name FROM employee WHERE INSTR(last_name,'c')>2;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| catherine  | rock      |
+------------+-----------+
1 row in set (0.00 sec)
```

QUERY 9:

List the employee_id , salary prefixed by “rs” in the descending order of values.

MariaDB [recruit]> select concat('id -',',','salary ',employee_id,' ','Rs. ',salary) from employee order by salary desc;

OUTPUT:



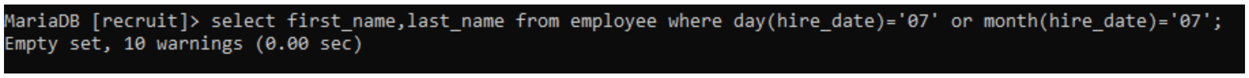
```
MariaDB [recruit]> select concat('id-',',','salary',employee_id,'','Rs.',salary)from employee order by salary desc;
+-------------------------------------------------------+
| concat('id-',',','salary',employee_id,'','Rs.',salary) |
+-------------------------------------------------------+
| id-,salary289Rs.50000                                  |
| id-,salary333Rs.50000                                  |
| id-,salary289Rs.50000                                  |
| id-,salary333Rs.50000                                  |
| id-,salary292Rs.45000                                  |
| id-,salary292Rs.45000                                  |
| id-,salary345Rs.40000                                  |
| id-,salary345Rs.40000                                  |
| id-,salary320Rs.35000                                  |
| id-,salary320Rs.35000                                  |
+-------------------------------------------------------+
10 rows in set (0.00 sec)
```

QUERY 10:

Write a query to display the employees firstname, lastname , who hired on seventh day of any month or seventh month in any year.

MariaDB [recruit]> select first_name,last_name from employee where day(hire_date)='07' or month(hire_date)='07';

OUTPUT:



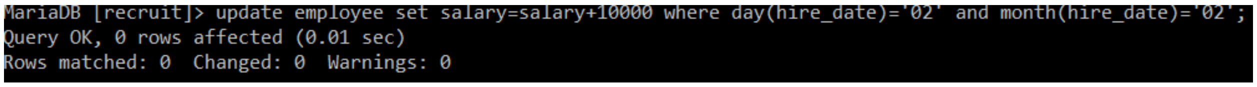
```
MariaDB [recruit]> select first_name,last_name from employee where day(hire_date)='07' or month(hire_date)='07';
Empty set, 10 warnings (0.00 sec)
```

QUERY 11:

Write a query to update incentive of Rs.10000 who have born on 2nd October.

Update employee set salary=salary+10000 where day(hire_date)=’02’ and month(hire_date)=’02’;

OUTPUT:



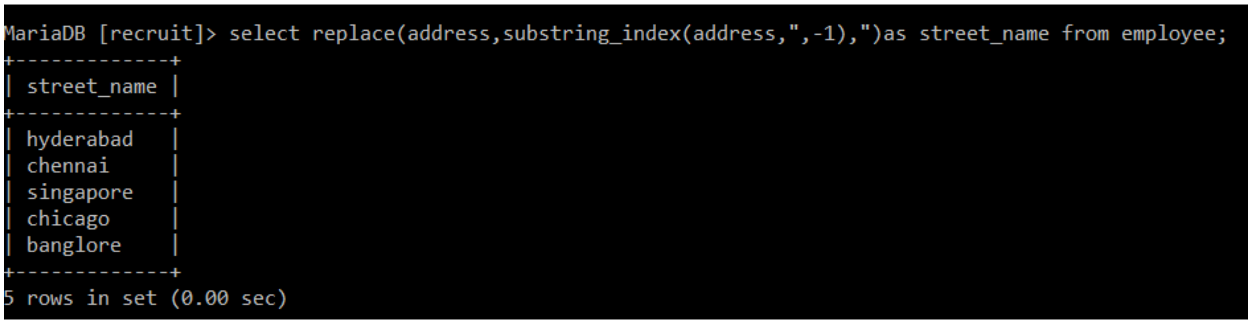
```
MariaDB [recruit]> update employee set salary=salary+10000 where day(hire_date)='02' and month(hire_date)='02';
Query OK, 0 rows affected (0.01 sec)
Rows matched: 0  Changed: 0  Warnings: 0
```

QUERY 12:

Write a query to get streetname of street address.

Select replace(address,substring_index(address,”,-1),”)as street_name from employee;

OUTPUT:



```
MariaDB [recruit]> select replace(address,substring_index(address,",-1),")as street_name from employee;
+-------------+
| street_name |
+-------------+
| hyderabad   |
| chennai     |
| singapore   |
| chicago     |
| banglore    |
+-------------+
5 rows in set (0.00 sec)
```

RESULT:

Thus the above MySQL Queries was executed successfully.

EXERCISE NO:07 **PRACTISE EXERCISE-05**

DATE: 10-07-2021

AIM:

To write and execute MySQL queries.

SYSTEM REQUIREMENTS:

MariaDB version 10.2.12

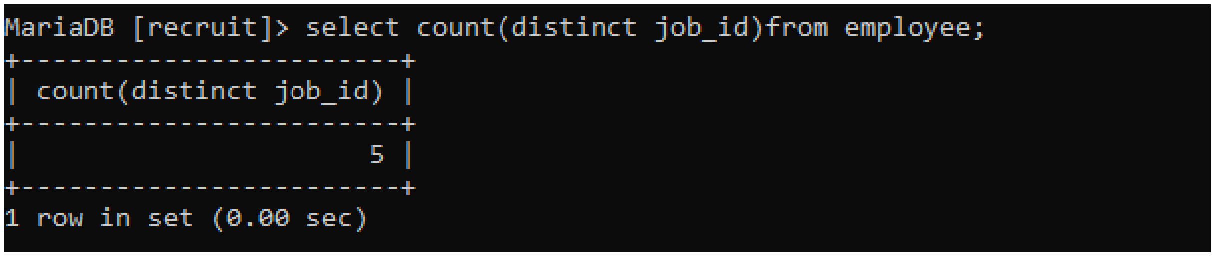
OS Windows (64 bit)

QUERY 1:

Write a query to list the number of jobs available in the employees table.

MariaDB [recruit]> select count(distinct job_id)from employee;

OUTPUT:



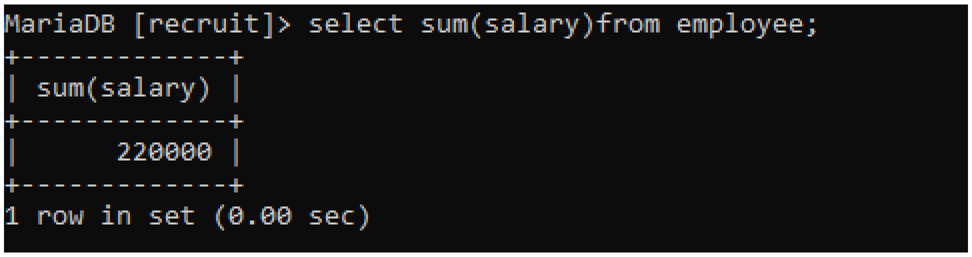
```
MariaDB [recruit]> select count(distinct job_id)from employee;
+------------------------+
| count(distinct job_id) |
+------------------------+
|                      5 |
+------------------------+
1 row in set (0.00 sec)
```

QUERY 2:

Write a query to get the total salaries payable to employees in each category for a month and the total salary.

MariaDB [recruit]> select sum(salary)from employee;

OUTPUT:



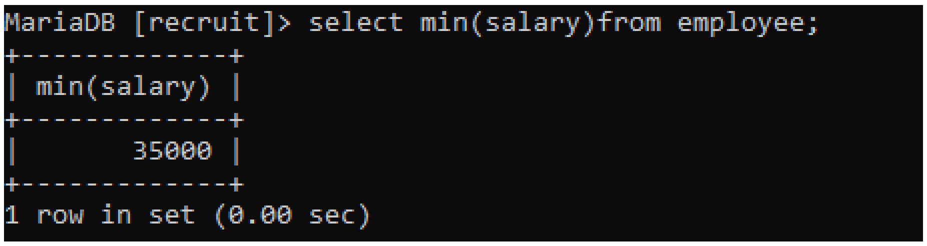
```
MariaDB [recruit]> select sum(salary)from employee;
+-------------+
| sum(salary) |
+-------------+
|      220000 |
+-------------+
1 row in set (0.00 sec)
```

QUERY 3:

Write a query to get the minimum salary from employees table.

MariaDB [recruit]> select min(salary)from employee;

OUTPUT:



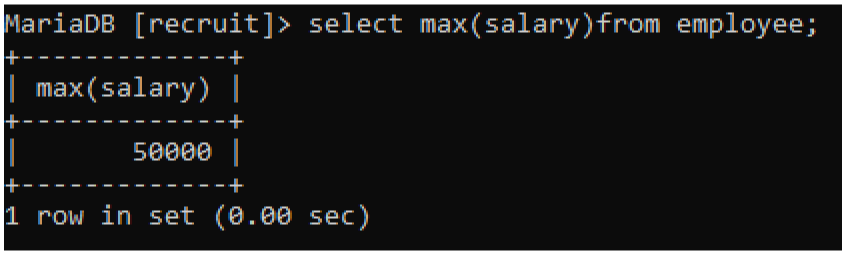
```
MariaDB [recruit]> select min(salary)from employee;
+-------------+
| min(salary) |
+-------------+
|       35000 |
+-------------+
1 row in set (0.00 sec)
```

QUERY 4:

Write a query to get the maximum salary of an employee working as a Programmer.

MariaDB [recruit]> select max(salary)from employee;

OUTPUT:



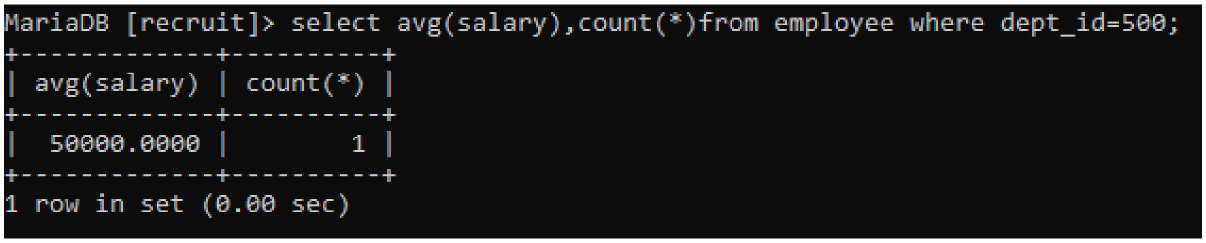
```
MariaDB [recruit]> select max(salary)from employee;
+-------------+
| max(salary) |
+-------------+
|       50000 |
+-------------+
1 row in set (0.00 sec)
```

QUERY 5:

Write a query to get the average salary spent towards a department per year.

MariaDB [recruit]> select avg(salary),count(*) from employee where dept_id=500;

OUTPUT:



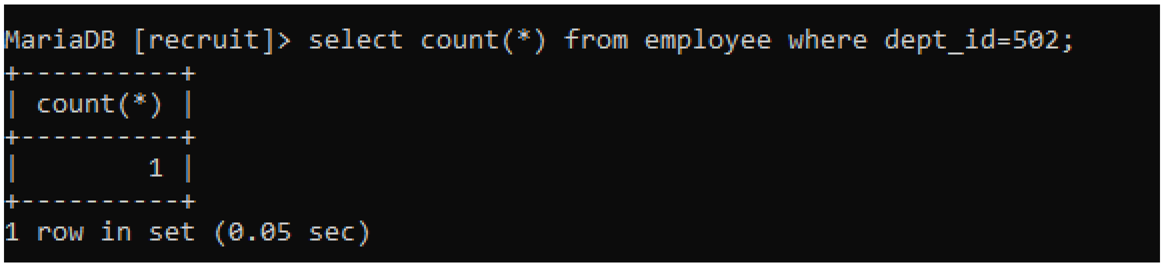
```
MariaDB [recruit]> select avg(salary),count(*)from employee where dept_id=500;
+-------------+----------+
| avg(salary) | count(*) |
+-------------+----------+
|  50000.0000 |        1 |
+-------------+----------+
1 row in set (0.00 sec)
```

QUERY 6:

Write a query to print the number of employees working in the department 90 with designation manager and programmer.

select count(*) from employee where dept_id=90;

OUTPUT:

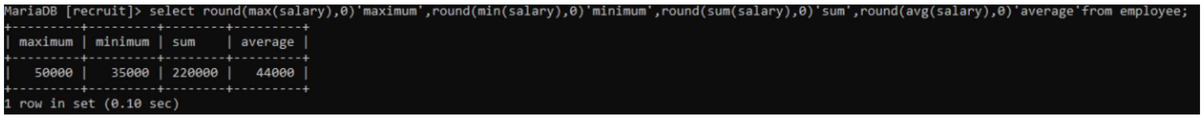


```
MariaDB [recruit]> select count(*) from employee where dept_id=502;
+----------+
| count(*) |
+----------+
|        1 |
+----------+
1 row in set (0.05 sec)
```

QUERY 7:

> **Write a query to get the highest, lowest, sum, and average salary of all employees.**
> MariaDB [recruit]> select round(max(salary),0) 'maximum', round(min(salary),0) 'minimum', round(sum(salary),0) 'sum',round(avg(salary),0) 'average' from employee;

OUTPUT:

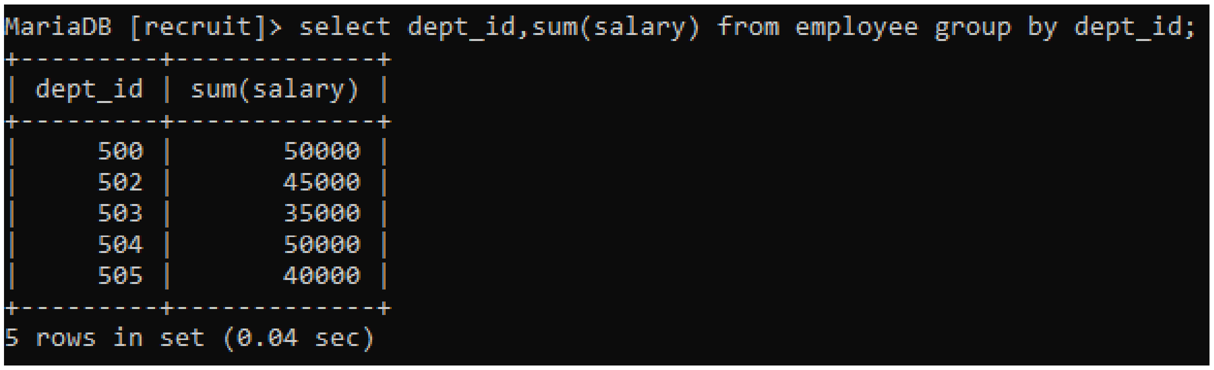


```
MariaDB [recruit]> select round(max(salary),0)'maximum',round(min(salary),0)'minimum',round(sum(salary),0)'sum',round(avg(salary),0)'average'from employee;
+---------+---------+--------+---------+
| maximum | minimum | sum    | average |
+---------+---------+--------+---------+
|   50000 |   35000 | 220000 |   44000 |
+---------+---------+--------+---------+
1 row in set (0.10 sec)
```

QUERY 8:

> **Write a query to get the department ID and the total salary payable in each department.**
> MariaDB [recruit]> select department_id,sum(salary) from employee group by dept_id;

OUTPUT:



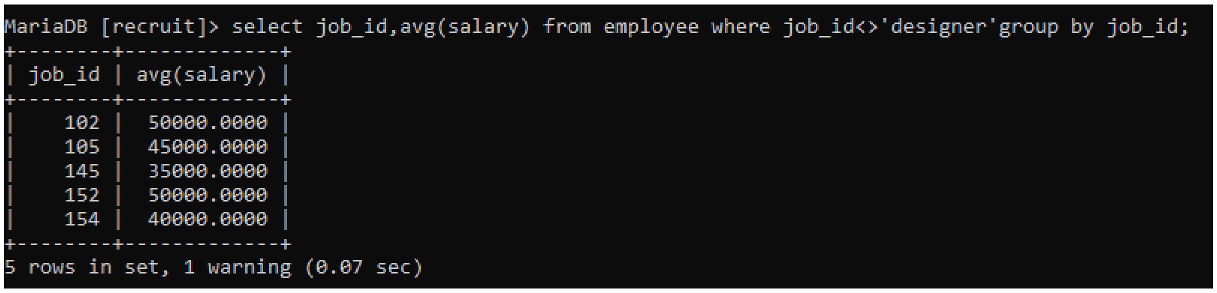
```
MariaDB [recruit]> select dept_id,sum(salary) from employee group by dept_id;
+---------+-------------+
| dept_id | sum(salary) |
+---------+-------------+
|     500 |       50000 |
|     502 |       45000 |
|     503 |       35000 |
|     504 |       50000 |
|     505 |       40000 |
+---------+-------------+
5 rows in set (0.04 sec)
```

QUERY 9:

Write a query to get the average salary for each job ID excluding programmer

MariaDB [recruit]> select job_id, avg(salary) from employee where job_id<> 'designer' group by job_id;

OUTPUT:



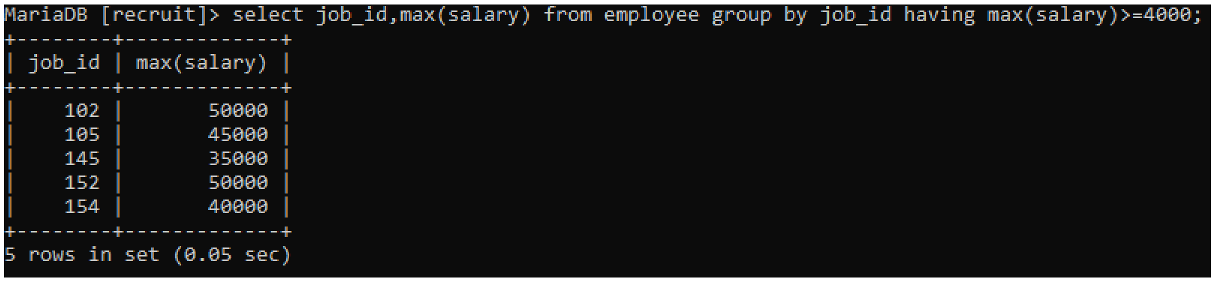
```
MariaDB [recruit]> select job_id,avg(salary) from employee where job_id<>'designer'group by job_id;
+--------+-------------+
| job_id | avg(salary) |
+--------+-------------+
|    102 |  50000.0000 |
|    105 |  45000.0000 |
|    145 |  35000.0000 |
|    152 |  50000.0000 |
|    154 |  40000.0000 |
+--------+-------------+
5 rows in set, 1 warning (0.07 sec)
```

QUERY 10:

Write a query to get the job ID and maximum salary of the employees where maximum salary is greater than or equal to $4000

MariaDB [recruit]> select job_id ,max(salary) from employee group by job_id having max(salary)>=4000;

OUTPUT:



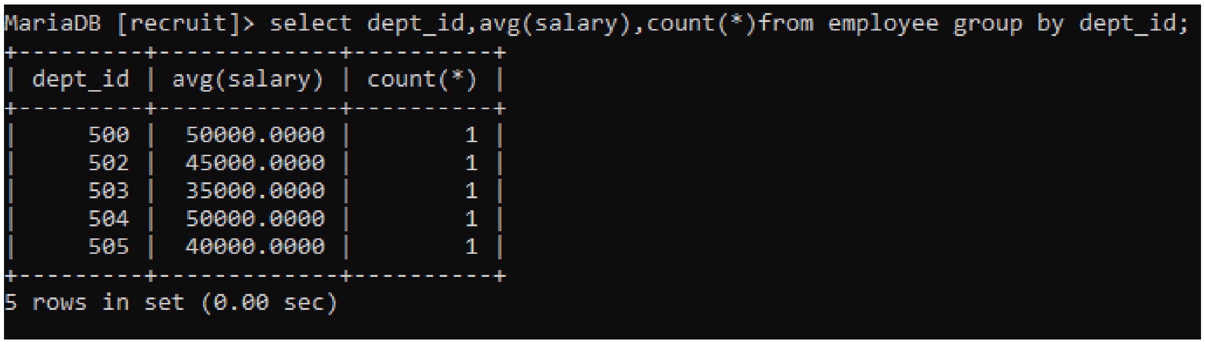
```
MariaDB [recruit]> select job_id,max(salary) from employee group by job_id having max(salary)>=4000;
+--------+-------------+
| job_id | max(salary) |
+--------+-------------+
|    102 |       50000 |
|    105 |       45000 |
|    145 |       35000 |
|    152 |       50000 |
|    154 |       40000 |
+--------+-------------+
5 rows in set (0.05 sec)
```

QUERY 11:

Write a query to get the average salary for all departments employing more than 10 employees

select department_id,avg(salary), count(*) from employee group by department_id;

OUTPUT:



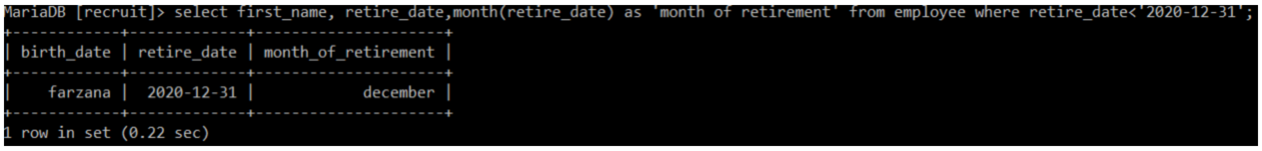
```
MariaDB [recruit]> select dept_id,avg(salary),count(*)from employee group by dept_id;
+---------+-------------+----------+
| dept_id | avg(salary) | count(*) |
+---------+-------------+----------+
|     500 |  50000.0000 |        1 |
|     502 |  45000.0000 |        1 |
|     503 |  35000.0000 |        1 |
|     504 |  50000.0000 |        1 |
|     505 |  40000.0000 |        1 |
+---------+-------------+----------+
5 rows in set (0.00 sec)
```

QUERY 12:

List the employees who will retire in the coming two years and the print the employee names who will retire in each month.

MariaDB [recruit]> selectfirst_name,retire_date,month(retire_date) as ‘month of retirement’ from employee where retire_date<’2021-12-31’;

OUTPUT:



```
MariaDB [recruit]> select first_name, retire_date,month(retire_date) as 'month of retirement' from employee where retire_date<'2020-12-31';
+------------+-------------+---------------------+
| birth_date | retire_date | month_of_retirement |
+------------+-------------+---------------------+
|    farzana |  2020-12-31 |            december |
+------------+-------------+---------------------+
1 row in set (0.22 sec)
```

RESULT:

Thus the above MySQL Queries was executed successfully.

QUIZ QUESTIONS:

1. What is the difference between count and count-distinct.

COUNT, COUNT DISTINCT are stored aggregate functions in SQL. They all will give a one row answer.

All these commands were introduced in SQL 1999.

COUNT	COUNT- DISTINCT
It returns total count of tuples satisfying expression specified using WHERE COUNT considers only those tuples that have a value. NULL values are not considered.	It returns the total number of distinct tuples satisfying the condition specified using WHERE. It only considers once if more than one tuple has equal values. COUNT DISTINCT considers only those tuples that have a value. NULL values are not considered.

2. What aggregate functions are available for calculating standard deviation and variance.

SQL anywhere supports two versions of variance and standard derivation functions, a sampling version, and a population version. Following are the standard derivation and variance functions offered in SQL.

STDDEV function

STDDEV pop function

STDDEV SAMP function

VARIANCE function

VAR-POP function

VAR-SAMP function

STDDEV pop function:

This function computes the standard deviation of a population consisting of a numeric expression as a DOUBLE.

STDDEV function:

This function is an alias for the STDDEV-STAMP function.

STDDEV-STAMP:

This function computes the standard deviation of a sample consisting of a numeric expression, as a DOUBLE.

VARIANCE function:

This function is as alias for the VAR-SAMP function.

VAR-POP function:

This function computes the statistical variance of a population consisting of a numeric expression, as a DOUBLE.

Example: The following statement lists the average and variance in the number of items per order in different time periods.

Select year(ship(date) as year, Quarter(shipdate) As Quarter, AVG(Quantity) as Average, VAR-POP(quantity) as variance FROM salesorder items.

GROUP BY Year, Quarter;

VAR-SAMP function:

It returns to the sample variance of the sample variance of a given expression.

3.How group by and having clauses are used?

The HAVING Clauses used in the select statement to specify filter conditions for a group of rows or aggregate. The HAVING clause is often used within the GROUP BY clause to filter groups based on a specified condition.

SYNTAX:

```
 Select
    Select_list
From
    Table_name
Where
    Search_condition
Group by
    Group_by_expression
Having
    Group_condition;
```

4.

❖ **BIT_AND()**

MySQL BIT_AND() function returns the bitwise AND of all bit in a given expression.

The calculation is performed on 64 bit precession. If this function doesnot find a matching row, it returns 18446744073759551615

SYNTAX:

Select book_id , BIT_AND('book_price') as BITS from book_most group by book_id;

❖ **BIT_OR()**

MySQL BIT_OR() function returns the bitwise OR of all bits ina given expression the calculation is performed on 64 bit precession. IF this function doesn't find a matching rows, it return 0.

SYNTAX:

Select book_id , BIT_OR('book_price') as BITS from book_most group by book_id;

❖ **BIT_XOR()**

MySQL BIT_XOR() function returns the bitwise XOR of all bits in a given expression. The calculation is performed on 64 bit precession. If this function doesn't find a matching rows, it return 0.

SYNTAX:

Select book_id , BIT_XOR('book_price') as BITS from book_most group by book_id;

❖ **GROUP_CONCAT()**

MySQL GROUP_CONCAT() function returns a string with concatenated non_NULL when there are no NULL values.

SYNTAX:

Select pub_id, group_CONCAT(cat_id) from book_most group by pub_id;

EXERCISE NO:08 **PRACTISE EXERCISE-06**

DATE: 17-**07**-2021

AIM:

To write and execute MySQL queries.

SYSTEM REQUIREMENTS:

MariaDB Version 10.2.12

OS Windows (64 bit)

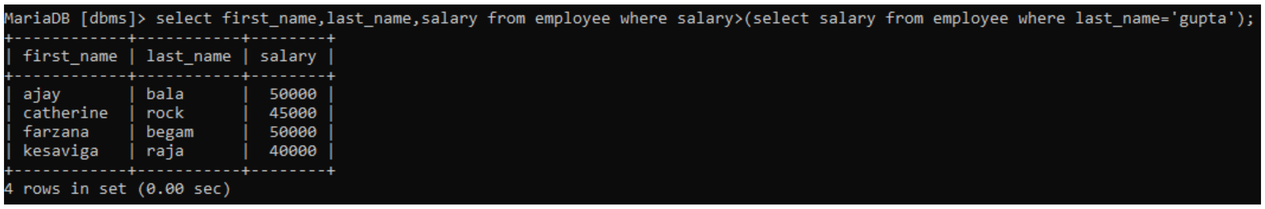
QUERIES:

QUERY 1:

Write a query to find the name(first_name, last_name) and the salary of the employees who have a higher salary that the employee whose last_name-'naidu'.

MariaDB [recruit]> select first_name,last_name,salary from employee where salary>(select salary from employee where last_name ='naidu');

OUTPUT:



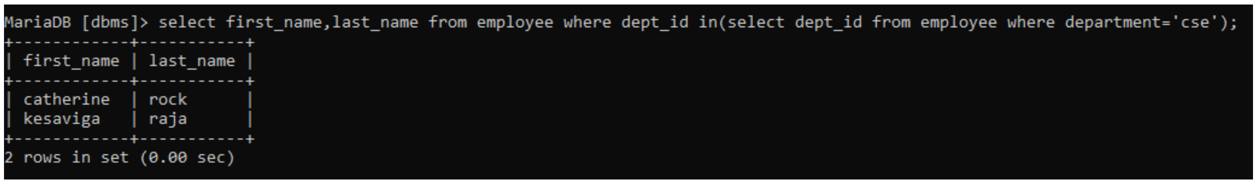
```
MariaDB [dbms]> select first_name,last_name,salary from employee where salary>(select salary from employee where last_name='gupta');
+------------+-----------+--------+
| first_name | last_name | salary |
+------------+-----------+--------+
| ajay       | bala      |  50000 |
| catherine  | rock      |  45000 |
| farzana    | begam     |  50000 |
| kesaviga   | raja      |  40000 |
+------------+-----------+--------+
4 rows in set (0.00 sec)
```

QUERY 2:

Write a query to find the name(first_name, last_name) of all employees who works in IT dept.

MariaDB [recruit]> select first_name,last_name from employee where department_id in (select department_id from department where department_name ='cse');

OUTPUT:



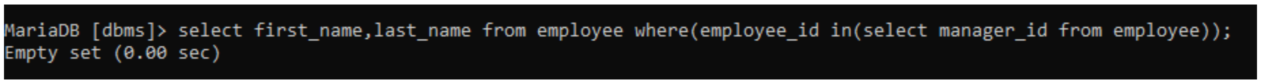
```
MariaDB [dbms]> select first_name,last_name from employee where dept_id in(select dept_id from employee where department='cse');
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| catherine  | rock      |
| kesaviga   | raja      |
+------------+-----------+
2 rows in set (0.00 sec)
```

QUERY 3:

Write a query to find the name(first_name, last_name) of the employees who are managers.

MariaDB [recruit]> select first_name,last_name from employee where (employee_id in (select manager_id from employee));

OUTPUT:



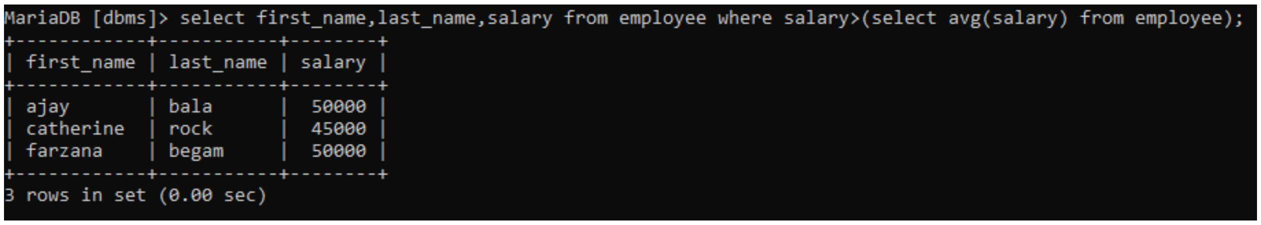
```
MariaDB [dbms]> select first_name,last_name from employee where(employee_id in(select manager_id from employee));
Empty set (0.00 sec)
```

QUERY 4:

Write a query to find the name(first_name,last_name) and salary of the employees whose salary is greater than the average salary of the IT Dept.

MariaDB [recruit]> select first_name,last_name,salary from employee where salary>(select avg(salary) from employee);

OUTPUT:



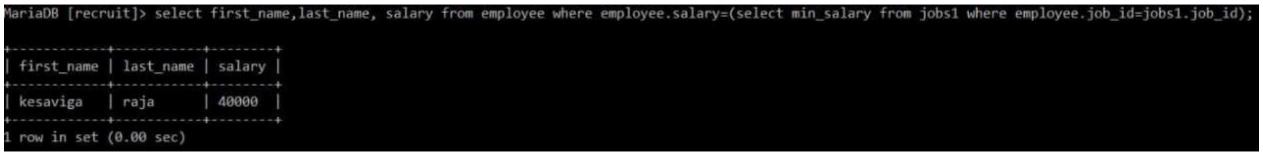
```
MariaDB [dbms]> select first_name,last_name,salary from employee where salary>(select avg(salary) from employee);
+------------+-----------+--------+
| first_name | last_name | salary |
+------------+-----------+--------+
| ajay       | bala      |  50000 |
| catherine  | rock      |  45000 |
| farzana    | begam     |  50000 |
+------------+-----------+--------+
3 rows in set (0.00 sec)
```

QUERY 5:

Write a query to find the name(first_name, last_name) and salary of the employees whose salary is equal to min_salary for their job grade.

MariaDB [recruit]> select first_name,last_name,salary from employee where employee.salary =(select min_salary from jobs1 where employee.job_id= jobs1.job_id);

OUTPUT:



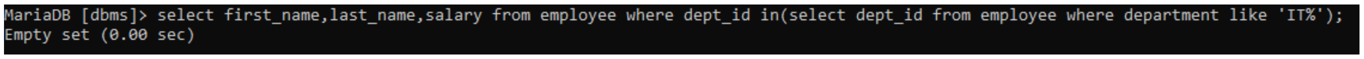
```
MariaDB [recruit]> select first_name,last_name, salary from employee where employee.salary=(select min_salary from jobs1 where employee.job_id=jobs1.job_id);

+------------+-----------+--------+
| first_name | last_name | salary |
+------------+-----------+--------+
| kesaviga   | raja      | 40000  |
+------------+-----------+--------+
1 row in set (0.00 sec)
```

QUERY 6:

Write a query to find the name(first_name, last_name) and salary of the employees who earns more than the avg salary and works in any of the IT Departments.

MariaDB [recruit]> select first_name,last_name,salary from employee where department_id in (select department_id from department where dept_name like 'IT%');

OUTPUT:



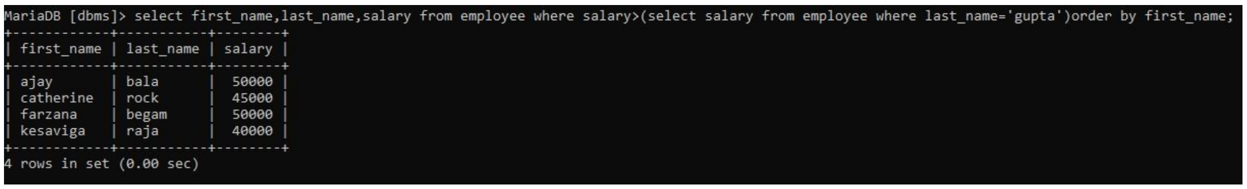
```
MariaDB [dbms]> select first_name,last_name,salary from employee where dept_id in(select dept_id from employee where department like 'IT%');
Empty set (0.00 sec)
```

QUERY 7:

Write a query to find the name(first_name, last_name) and salary of the employees who serve as a manager and who earns more than the earning of Mr.Ram.

MariaDB [recruit]> select first_name,last_name,salary from employee where salary>(select salary from employee where last_name ='gupta') order by first_name;

OUTPUT:



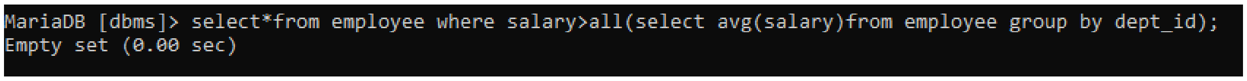
```
MariaDB [dbms]> select first_name,last_name,salary from employee where salary>(select salary from employee where last_name='gupta')order by first_name;
+------------+-----------+--------+
| first_name | last_name | salary |
+------------+-----------+--------+
| ajay       | bala      |  50000 |
| catherine  | rock      |  45000 |
| farzana    | begam     |  50000 |
| kesaviga   | raja      |  40000 |
+------------+-----------+--------+
4 rows in set (0.00 sec)
```

QUERY 8:

Write a query to find the name(first_name, last_name) and salary of the employees whose salary is greater than the average salary of all the departments.

MariaDB [recruit]> select*from employee where salary>all(select avg(salary) from employee group by dept_id);

OUTPUT:



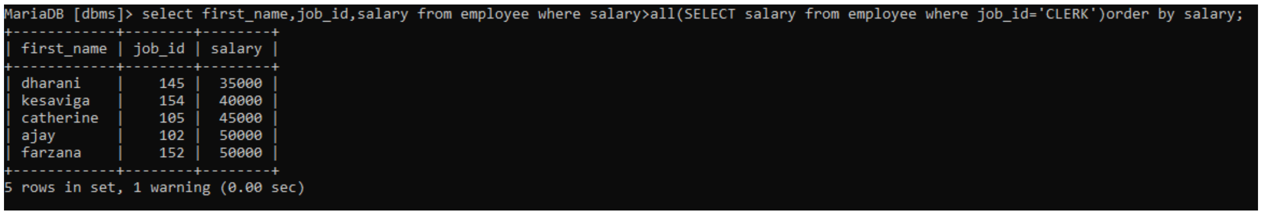
```
MariaDB [dbms]> select*from employee where salary>all(select avg(salary)from employee group by dept_id);
Empty set (0.00 sec)
```

QUERY 9:

Write a query to find the name and salary of the employees who earn a salary that is higher than the salary of all the clerk (JOB_ID ='clerk'). Sort the results of the salary of the lowest to highest.

MariaDB [recruit]> select first_name,job_id,salary from employee where salary>all(SELECT salary from employee where job_id='CLERK')order by salary;

OUTPUT:



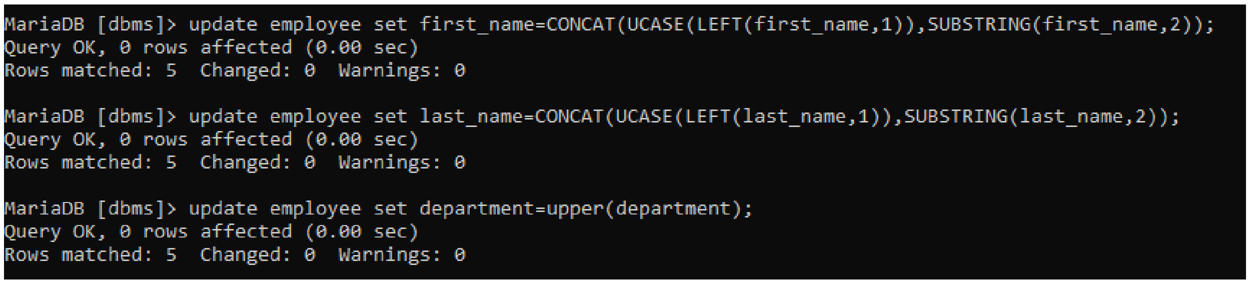
```
MariaDB [dbms]> select first_name,job_id,salary from employee where salary>all(SELECT salary from employee where job_id='CLERK')order by salary;
+------------+--------+--------+
| first_name | job_id | salary |
+------------+--------+--------+
| dharani    |    145 |  35000 |
| kesaviga   |    154 |  40000 |
| catherine  |    105 |  45000 |
| ajay       |    102 |  50000 |
| farzana    |    152 |  50000 |
+------------+--------+--------+
5 rows in set, 1 warning (0.00 sec)
```

QUERY 10:

Change the employee names to have first letter of first name and last_name in upper case and department name all capital letters.

MariaDB [recruit]> update employee set first_name=CONCAT(UCASE(LEFT(first_name,1)),SUBSTRING(first_name,2));

OUTPUT:



```
MariaDB [dbms]> update employee set first_name=CONCAT(UCASE(LEFT(first_name,1)),SUBSTRING(first_name,2));
Query OK, 0 rows affected (0.00 sec)
Rows matched: 5  Changed: 0  Warnings: 0

MariaDB [dbms]> update employee set last_name=CONCAT(UCASE(LEFT(last_name,1)),SUBSTRING(last_name,2));
Query OK, 0 rows affected (0.00 sec)
Rows matched: 5  Changed: 0  Warnings: 0

MariaDB [dbms]> update employee set department=upper(department);
Query OK, 0 rows affected (0.00 sec)
Rows matched: 5  Changed: 0  Warnings: 0
```

RESULT:

Thus the above MySQL Queries was executed successfully.

QUIZ QUESTIONS:

- **ASCII( )**

  Select first_name, ASCII(first_name) as “ASCII value of 1 *character “ from employee where ASCII(first_name)<70;

- **CHAR( )**

  Select CHAR(67,72,65,82);
- **CHAR_LENGTH( )**

  Select first_name, CHAR_LENGTH(first_name) as ‘character length’ from employee where CHAR_LENGTH(first_name)>20;
- **CHARACTER_LENGTH( )**

  Select first_name, CHARACTER_LENGTH(first_name) as ‘character length’ from employee where CHARACTER_LENGTH(first_name)>20;

- **CONCAT( )**

  Select CONCAT(first_name,’→’,last_name)from employee;
- **CONCAT_WS( )**

  Select CONCAT_WS(‘,’,employee_id, first_name, email, salary) from employee where employee_id< >’e1’;
- **FORMAT( )**

  Select first_name, FORMAT(salary,4) from employee where salary >’1000’;
- **HEX( )**

  Select HEX(‘Q’);
- **INSERT( )**

  Select INSERT(employee_id, 4,0’/’)from employee where salary=’2000’;
- **INSTR( )**

  Select first_name, INSTR(first_name,’an’)from employee where INSTR(first_name,’an’)>0;
- **LCASE( )**

Select first_name, LCASE(first_name) from employee where employee_id< >'2';

- **LEFT( )**
  Select first_name, LEFT(first_name) from employee;
- **LENGTH( )**
  Select first_name, LENGTH(first_name) from employee;
- **LIKE( )**
  Select * from employee where first_name like 'a%';
- **LOCATE( )**
  Select first_name, LOCATE('at',first_name) from employee where LOCATE('at',first_name)>0;
- **LOWER( )**
  Select first_name, LOWER(first_name)from employee where salary< >'2000';
- **LPAD( )**
  Select LPAD(last_name,25-(length(last_name)),'**)from employee;
- **LTRIM( )**
  Update employee set first_name=LTRIM(first_name);
- **MAKE_SET( )**
  Select MAKE_SET('4','hello','nice','world');
- **MATCH( )**
  Select * from employees where MATCH(employee_id, first_name)('search terms ' in natural language mode);
- **MID( )**
  Select first_name, MID(first_name,4,5) from employee where employee_id< >1;
- **NOT LIKE( )**
  Select first_name, last_name from employee where first_name NOT LIKE='W%';
- **NOT REGEXP( )**
  Select employee_id, first_name from employee where first_name not regexp'^sh';
- **ORD( )**

Select ORD(“hello”);

- **POSITION( )**

  Select position(“ll”,N,”hello”);

- **REGEXP( )**

  Select * from employee where first_name REGEXP ‘^w’;

- **REGEXP_INSTR( )**

  Select employee_id, first_name, REGEXP_INSTR(first_name,’a|e|i|o|u’) as first-occurrence from employee;

- **REGEXP_LIKE( )**

  Select last_name from employee where REGEXP_LIKE(last_name,’^A(+)’);

- **REGEXP_REPLACE( )**

  Select employee_id,last_name, REGEXP_REPLACE(last_name, ’a|e|i|o|u’,”c”) as result from employee;

- **REGEXP_SUBSTR( )**

  Select employee_id, last_name, REGEXP_SUBSTR(last_name, ’a|e|i|o|u’) as “first vowel” from employee;

- **REPEAT( )**

  Select REPEAT(‘A’,2);

- **REPLACE( )**

  Select first_name, last_name, REPLACE(last_name,’sharan’,’sekar’) from employee where last_name=’k’;

- **REVERSE( )**

  Select first_name, last_name, REVERSE(last_name) from employee where last_name=’pillai’;

- **RIGHT( )**

  Select first_name, RIGHT(first_name,7) from employee where employee_id=2;

- **RLIKE( )**

  Select * from employee where first_name RLIKE ‘^w’;

- **RPAD( )**

  Select first_name RPAD(first_name,25,’*’)from employee_id=2;

- **RTIM( )**

Select RTIM(first_name),last_name from employee where employee_id='500';

- **SOUNDEX( )**
  Select SOUNDEX('helloworld');
- **SOUNDS_LIKE( )**
  Select * from employee where first_name SOUNDSLIKE 'sudipto';
- **STRCMP( )**
  STRCMP('helloworld','helloworld');
- **SUBSTR( )**
  Select first_name, SUBSTR(first_name,4,5)from employee where last_name='c';
- **TO_BASE64( )**
  Select TO_BASE64("hello");
- **TRIM( )**
  Select TRIM(LEADING '0' from '000123');
- **UCASE( )**
  Select UCASE('hello'),UPPER('hello');
- **UNHEX( )**
  Select UNHEX('WWW');
- **UPPER( )**
  Select first_name,UPPER(first_name) from employee WHERE employee_id < > '10';
- **WEIGHT_STRING( )**
  Select WEIGHT_STRING('cat');

Exercise No:9 **PRACTICE EXERCISE-7**

Date: 24-07-2021

AIM:

To write and execute MySQL queries.

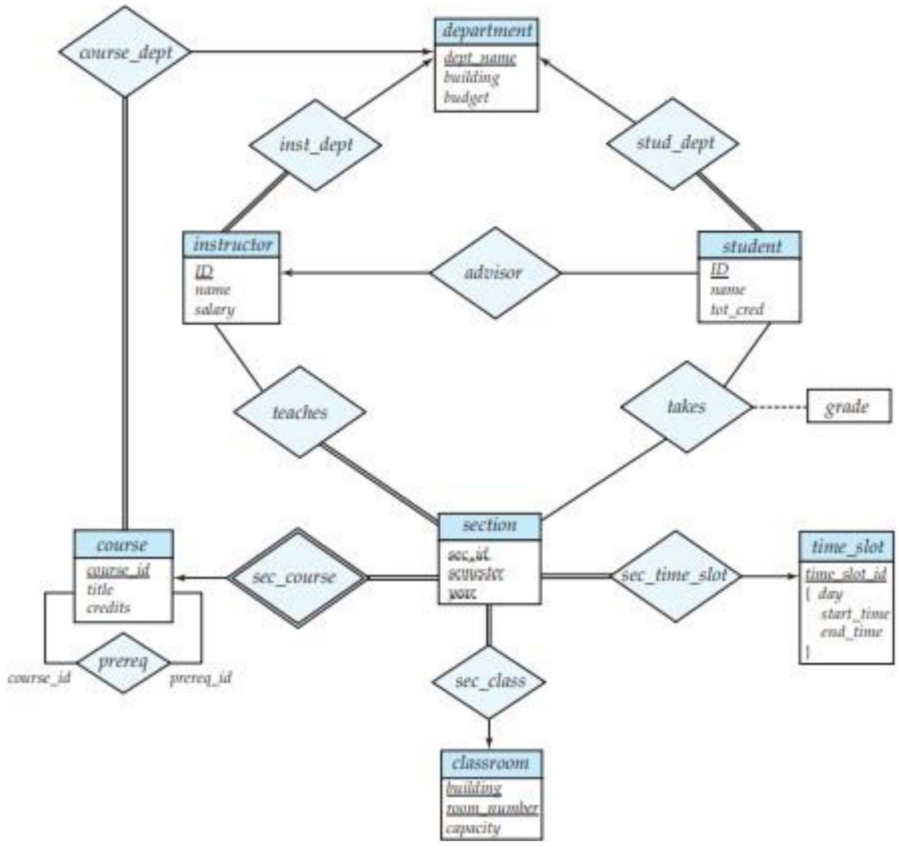
SYSTEM REQUIREMENTS:

MariaDB version 10.2.12

OS windows 8(64 bit)

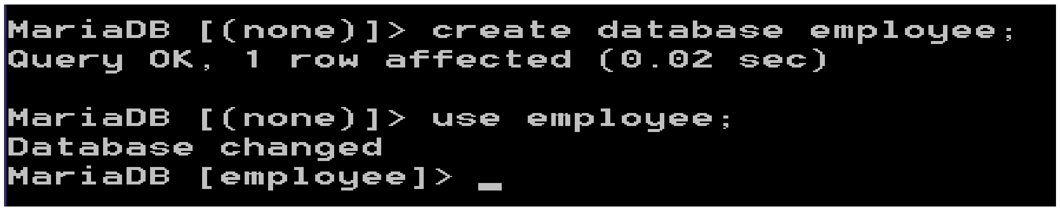
Reference : chapter 3 of silberschatz sixth edition Work from home to solve these queries. Store the results of the queries in Notepad and you can show output when you come back

For this ER diagram create necessary tables and store data necessary for solving the given queries in mysql.



Employee database:

create database employee;



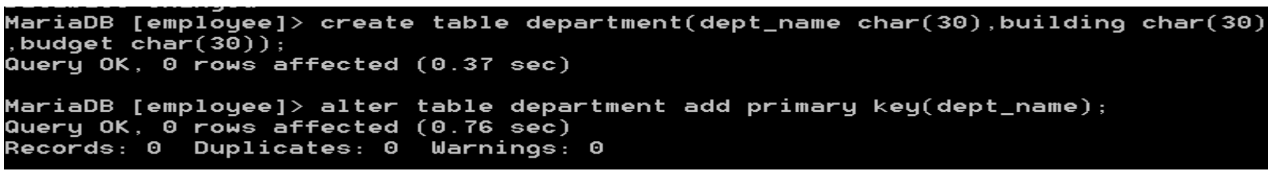
```
MariaDB [(none)]> create database employee;
Query OK, 1 row affected (0.02 sec)

MariaDB [(none)]> use employee;
Database changed
MariaDB [employee]> _
```

CREATE TABLE QUERIES:

- **DEPARTMENT:**

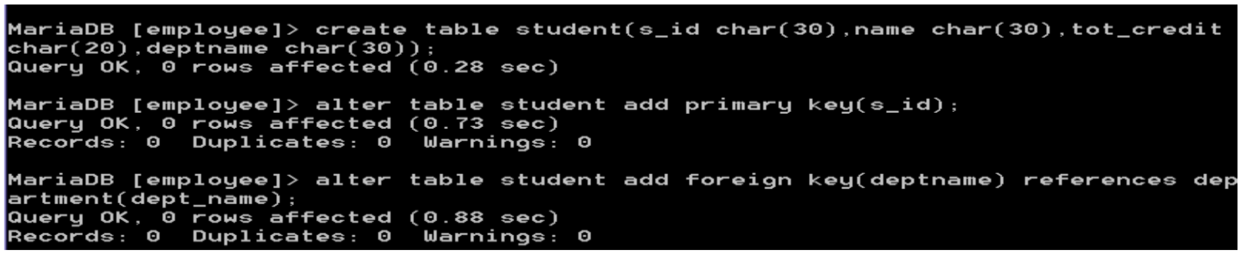
  i. createtable department(dept_name char(30),building char(30),budget char(30));

  ii. altertable department addprimarykey(dept_name);



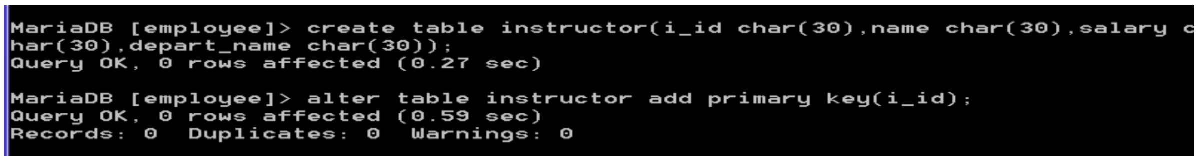
```
MariaDB [employee]> create table department(dept_name char(30),building char(30)
,budget char(30));
Query OK, 0 rows affected (0.37 sec)

MariaDB [employee]> alter table department add primary key(dept_name);
Query OK, 0 rows affected (0.76 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **STUDENT:**
  i. createtable student(s_id char(30),name char(30),tot_credit char(20),deptname char(30));
  ii. altertable student addprimarykey(s_id);
  iii. altertable student addforeignkey(deptname) references department(dept_name);



```
MariaDB [employee]> create table student(s_id char(30),name char(30),tot_credit
char(20),deptname char(30));
Query OK, 0 rows affected (0.28 sec)

MariaDB [employee]> alter table student add primary key(s_id);
Query OK, 0 rows affected (0.73 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table student add foreign key(deptname) references dep
artment(dept_name);
Query OK, 0 rows affected (0.88 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

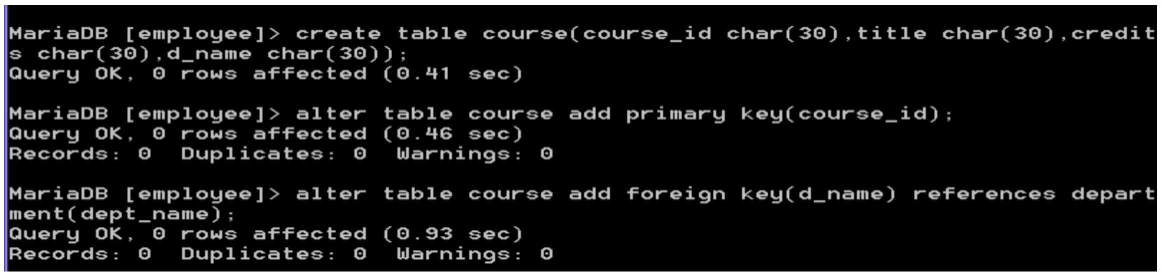
- **INSTRUCTOR:**
  i. createtable instructor(i_id char(30),name char(30),salary char(30),depart_name char(30));
  ii. altertable instructor addprimarykey(i_id);



```
MariaDB [employee]> create table instructor(i_id char(30),name char(30),salary c
har(30),depart_name char(30));
Query OK, 0 rows affected (0.27 sec)

MariaDB [employee]> alter table instructor add primary key(i_id);
Query OK, 0 rows affected (0.59 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **COURSE:**

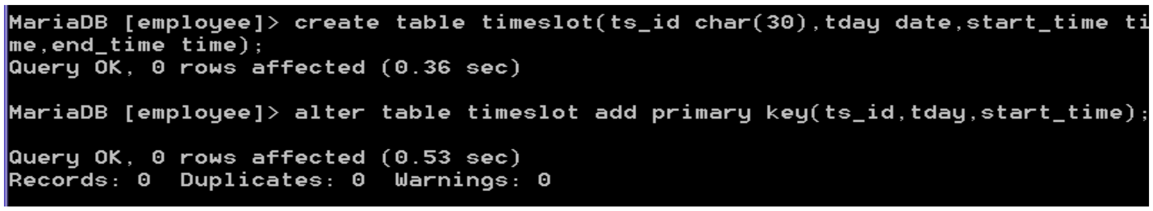
i. createtable course(course_id char(30),title char(30),credits char(30),d_name char(30));
ii. altertable course addprimarykey(course_id);
iii. altertable course addforeignkey(d_name) references department(dept_name);



```
MariaDB [employee]> create table course(course_id char(30),title char(30),credit
s char(30),d_name char(30));
Query OK, 0 rows affected (0.41 sec)

MariaDB [employee]> alter table course add primary key(course_id);
Query OK, 0 rows affected (0.46 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table course add foreign key(d_name) references depart
ment(dept_name);
Query OK, 0 rows affected (0.93 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **TIMESLOT:**

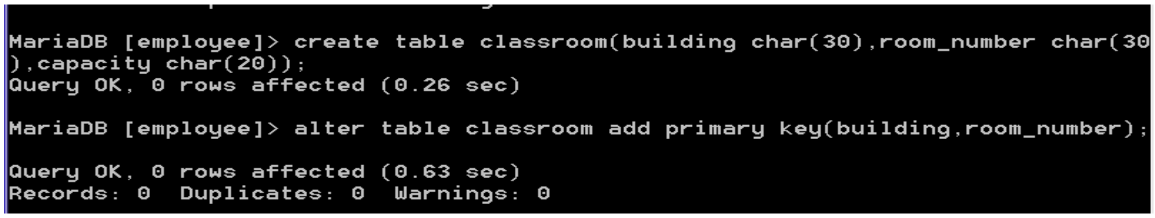
i. createtable timeslot(ts_id char(30),tday date,start_time time,end_time time);
ii. altertable timeslot addprimarykey(ts_id,tday,start_time);



```
MariaDB [employee]> create table timeslot(ts_id char(30),tday date,start_time ti
me,end_time time);
Query OK, 0 rows affected (0.36 sec)

MariaDB [employee]> alter table timeslot add primary key(ts_id,tday,start_time);

Query OK, 0 rows affected (0.53 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

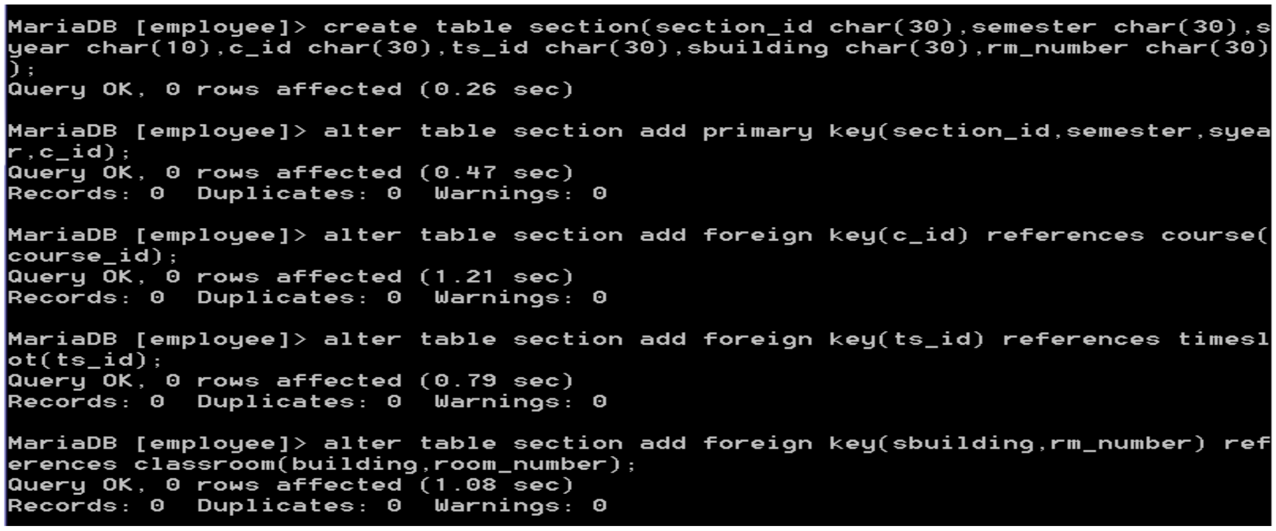
- **CLASSROOM:**

i. createtable classroom(building char(30),room_number char(30),capacity char(20));
ii. altertable classroom addprimarykey(building,room_number);



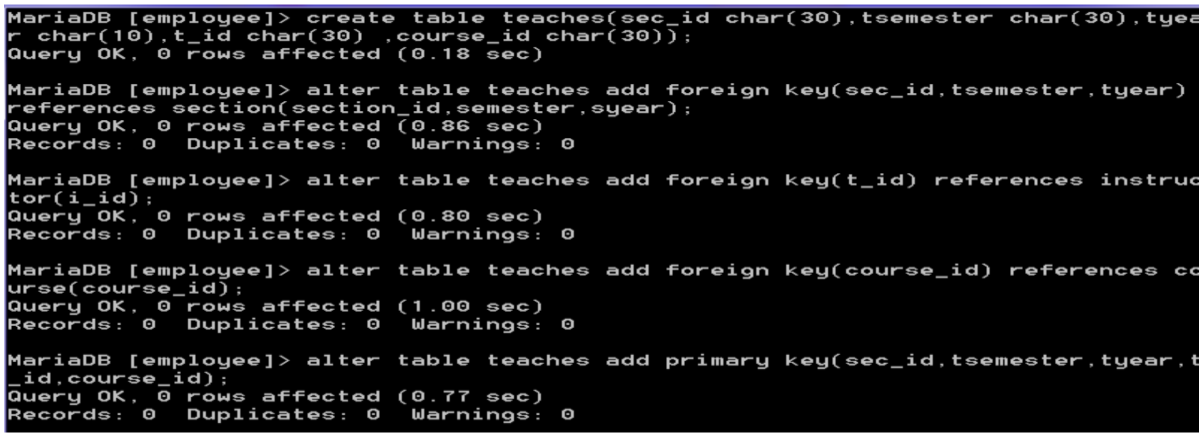
```
MariaDB [employee]> create table classroom(building char(30),room_number char(30
),capacity char(20));
Query OK, 0 rows affected (0.26 sec)

MariaDB [employee]> alter table classroom add primary key(building,room_number);

Query OK, 0 rows affected (0.63 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **SECTION:**
  1. createtable section(section_id char(30),semester char(30),syear char(10),c_id char(30),ts_id char(30),sbuilding char(30),rm_number char(30));
  2. altertable section addprimarykey(section_id,semester,syear,c_id);
  3. altertable section addforeignkey(c_id) references course(course_id);
  4. altertable section addforeignkey(ts_id) references timeslot(ts_id);
  5. altertable section addforeignkey(sbuilding,rm_number) references classroom(building,room_number);



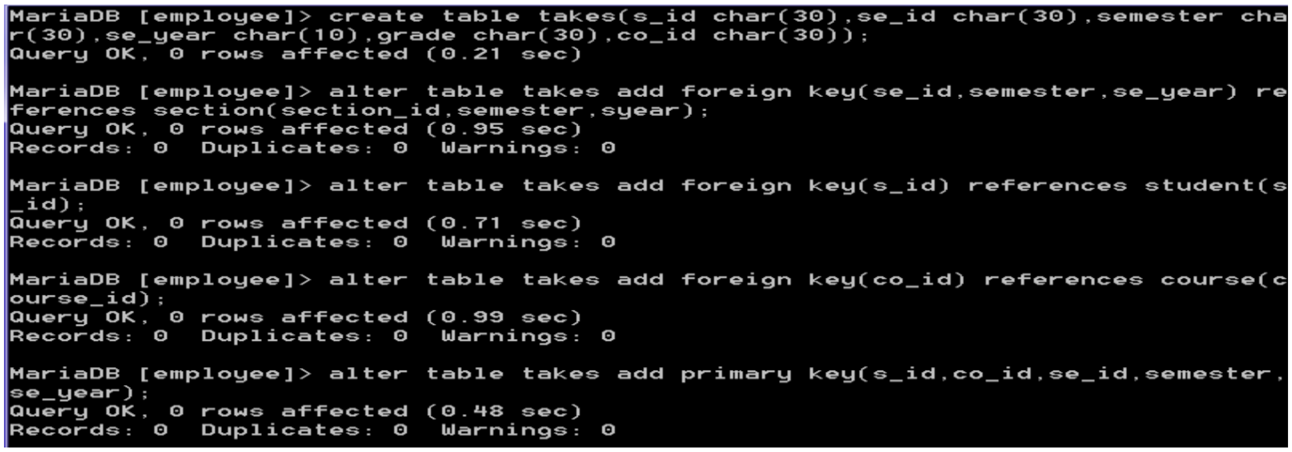
```
MariaDB [employee]> create table section(section_id char(30),semester char(30),s
year char(10),c_id char(30),ts_id char(30),sbuilding char(30),rm_number char(30)
);
Query OK, 0 rows affected (0.26 sec)

MariaDB [employee]> alter table section add primary key(section_id,semester,syea
r,c_id);
Query OK, 0 rows affected (0.47 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table section add foreign key(c_id) references course(
course_id);
Query OK, 0 rows affected (1.21 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table section add foreign key(ts_id) references timesl
ot(ts_id);
Query OK, 0 rows affected (0.79 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table section add foreign key(sbuilding,rm_number) ref
erences classroom(building,room_number);
Query OK, 0 rows affected (1.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **TEACHES:**
  1. createtable teaches(sec_id char(30),tsemester char(30),tyear char(10),t_id char(30) ,course_id char(30));
  2. altertable teaches addforeignkey(sec_id,tsemester,tyear) references section(section_id,semester,syear);
  3. altertable teaches addforeignkey(t_id) references instructor(i_id);
  4. altertable teaches addforeignkey(course_id) references course(course_id);
  5. altertable teaches addprimarykey(sec_id,tsemester,tyear,t_id,course_id);



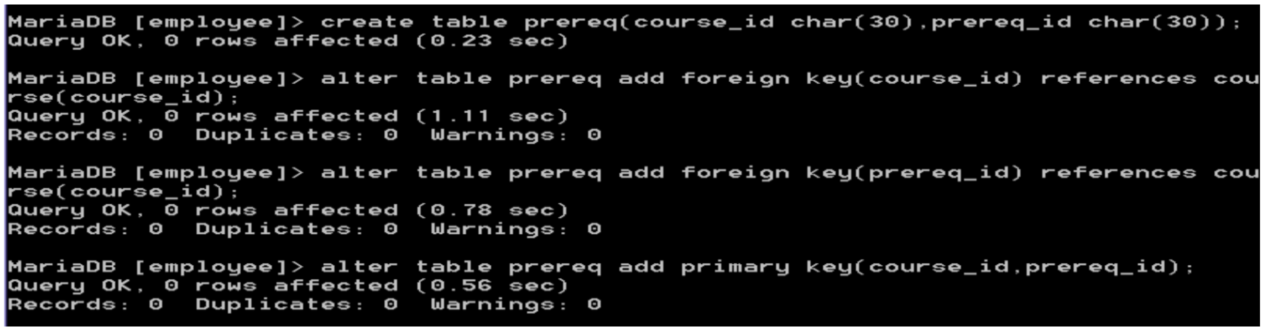
```
MariaDB [employee]> create table teaches(sec_id char(30),tsemester char(30),tyea
r char(10),t_id char(30) ,course_id char(30));
Query OK, 0 rows affected (0.18 sec)

MariaDB [employee]> alter table teaches add foreign key(sec_id,tsemester,tyear)
references section(section_id,semester,syear);
Query OK, 0 rows affected (0.86 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table teaches add foreign key(t_id) references instruc
tor(i_id);
Query OK, 0 rows affected (0.80 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table teaches add foreign key(course_id) references co
urse(course_id);
Query OK, 0 rows affected (1.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table teaches add primary key(sec_id,tsemester,tyear,t
_id,course_id);
Query OK, 0 rows affected (0.77 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **TAKES:**
  i. createtable takes(s_id char(30),se_id char(30),semester char(30),se_year char(10),grade char(30),co_id char(30));
  ii. altertable takes addforeignkey(se_id,semester,se_year) references section(section_id,semester,syear);
  iii. altertable takes addforeignkey(s_id) references student(s_id);
  iv. altertable takes addforeignkey(co_id) references course(course_id);
  v. altertable takes addprimarykey(s_id,co_id,se_id,semester,se_year);

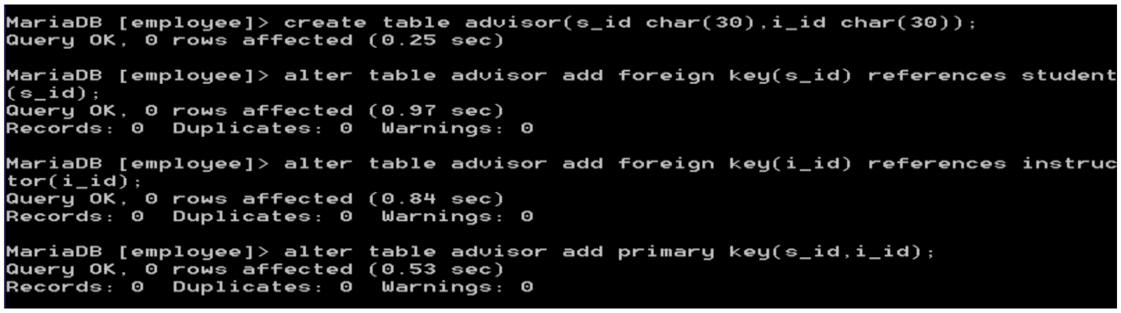


```
MariaDB [employee]> create table takes(s_id char(30),se_id char(30),semester cha
r(30),se_year char(10),grade char(30),co_id char(30));
Query OK, 0 rows affected (0.21 sec)

MariaDB [employee]> alter table takes add foreign key(se_id,semester,se_year) re
ferences section(section_id,semester,syear);
Query OK, 0 rows affected (0.95 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table takes add foreign key(s_id) references student(s
_id);
Query OK, 0 rows affected (0.71 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table takes add foreign key(co_id) references course(c
ourse_id);
Query OK, 0 rows affected (0.99 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table takes add primary key(s_id,co_id,se_id,semester,
se_year);
Query OK, 0 rows affected (0.48 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **PREREQ:**
  i. createtable prereq(course_id char(30),prereq_id char(30));
  ii. altertable prereq addforeignkey(course_id) references course(course_id);
  iii. altertable prereq addforeignkey(prereq_id) references course(course_id);
  iv. altertable prereq addprimarykey(course_id,prereq_id);

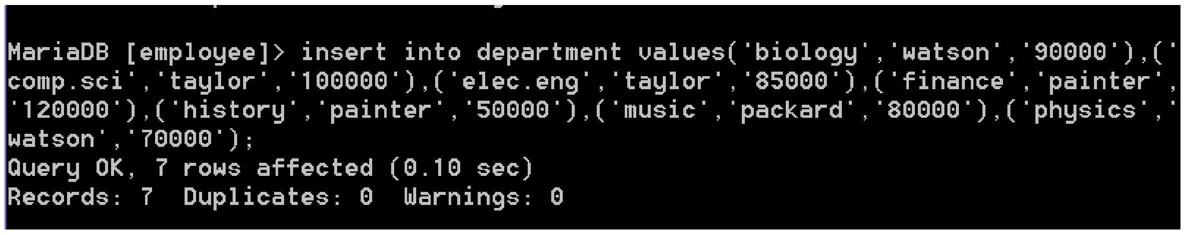

```
MariaDB [employee]> create table prereq(course_id char(30),prereq_id char(30));
Query OK, 0 rows affected (0.23 sec)

MariaDB [employee]> alter table prereq add foreign key(course_id) references cou
rse(course_id);
Query OK, 0 rows affected (1.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table prereq add foreign key(prereq_id) references cou
rse(course_id);
Query OK, 0 rows affected (0.78 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table prereq add primary key(course_id,prereq_id);
Query OK, 0 rows affected (0.56 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- **ADVISOR:**
  i. createtable advisor(s_id char(30),i_id char(30));
  ii. altertable advisor addforeignkey(s_id) references student(s_id);
  iii. altertable advisor addforeignkey(i_id) references instructor(i_id);
  iv. altertable advisor addprimarykey(s_id,i_id);


```
MariaDB [employee]> create table advisor(s_id char(30),i_id char(30));
Query OK, 0 rows affected (0.25 sec)

MariaDB [employee]> alter table advisor add foreign key(s_id) references student
(s_id);
Query OK, 0 rows affected (0.97 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table advisor add foreign key(i_id) references instruc
tor(i_id);
Query OK, 0 rows affected (0.84 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [employee]> alter table advisor add primary key(s_id,i_id);
Query OK, 0 rows affected (0.53 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

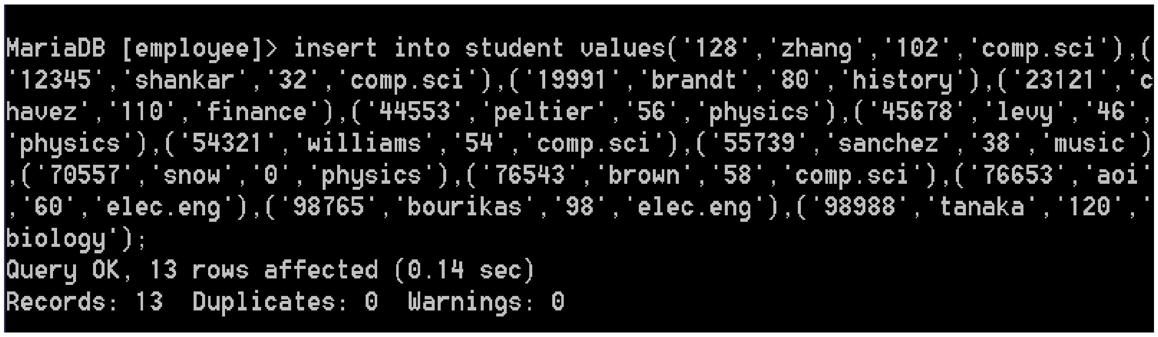
INSERT INTO QUERIES:

insertinto department values('biology','watson','90000'),('comp.sci','taylor','100000'),('elec.eng','taylor','85000'),('finance','painter','120000'),('history','painter','50000'),('music','packard','80000'),('physics','watson','70000');



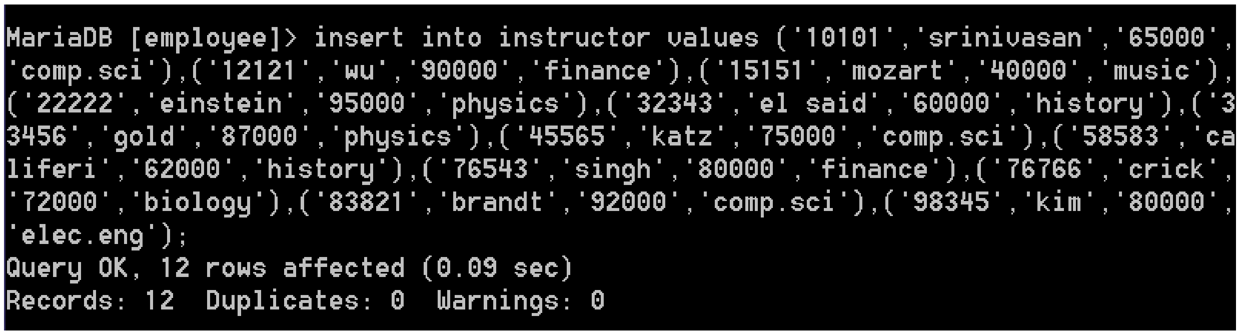
```
MariaDB [employee]> insert into department values('biology','watson','90000'),('
comp.sci','taylor','100000'),('elec.eng','taylor','85000'),('finance','painter',
'120000'),('history','painter','50000'),('music','packard','80000'),('physics','
watson','70000');
Query OK, 7 rows affected (0.10 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

insertinto student values('128','zhang','102','comp.sci'),('12345','shankar','32','comp.sci'),('19991','brandt','80','history'),('23121','chavez','110','finance'),('44553','peltier','56','physics'),('45678','levy','46','physics'),('54321','williams','54','comp.sci'),('55739','sanchez','38','music'),('70557','snow','0','physics'),('76543','brown','58','comp.sci'),('76653','aoi','60','elec.eng'),('98765','bourikas','98','elec.eng'),('98988','tanaka','120','biology');



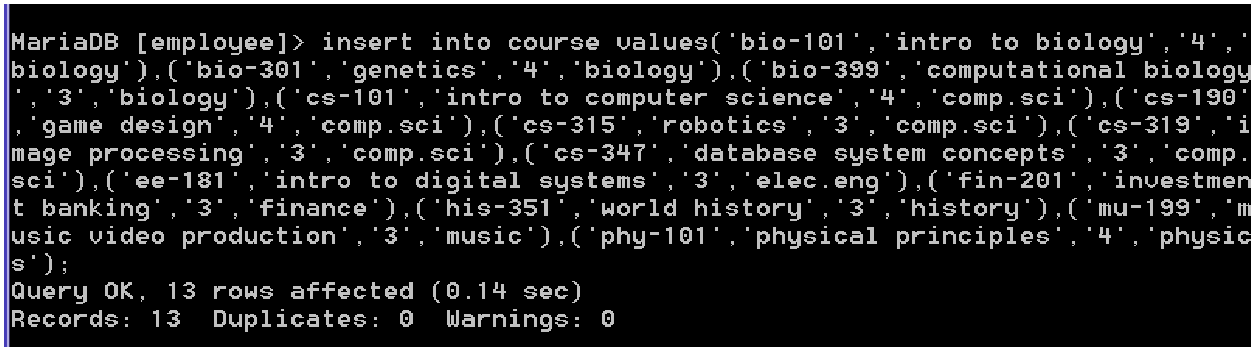
```
MariaDB [employee]> insert into student values('128','zhang','102','comp.sci'),(
'12345','shankar','32','comp.sci'),('19991','brandt','80','history'),('23121','c
havez','110','finance'),('44553','peltier','56','physics'),('45678','levy','46',
'physics'),('54321','williams','54','comp.sci'),('55739','sanchez','38','music')
,('70557','snow','0','physics'),('76543','brown','58','comp.sci'),('76653','aoi'
,'60','elec.eng'),('98765','bourikas','98','elec.eng'),('98988','tanaka','120','
biology');
Query OK, 13 rows affected (0.14 sec)
Records: 13  Duplicates: 0  Warnings: 0
```

insertinto instructor values ('10101','srinivasan','65000','comp.sci'),('12121','wu','90000','finance'),('15151','mozart','40000','music'),('22222','einstein','95000','physics'),('32343','el said','60000','history'),('33456','gold','87000','physics'),('45565','katz','75000','comp.sci'),('58583','califeri','62000','history'),('76543','singh','80000','finance'),('76766','crick','72000','biology'),('83821','brandt','92000','comp.sci'),('98345','kim','80000','elec.eng');



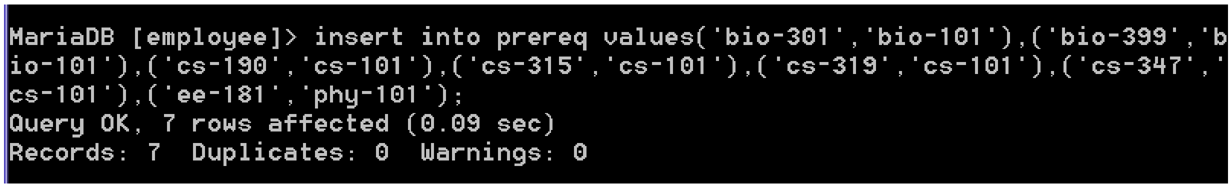
```
MariaDB [employee]> insert into instructor values ('10101','srinivasan','65000',
'comp.sci'),('12121','wu','90000','finance'),('15151','mozart','40000','music'),
('22222','einstein','95000','physics'),('32343','el said','60000','history'),('3
3456','gold','87000','physics'),('45565','katz','75000','comp.sci'),('58583','ca
liferi','62000','history'),('76543','singh','80000','finance'),('76766','crick',
'72000','biology'),('83821','brandt','92000','comp.sci'),('98345','kim','80000',
'elec.eng');
Query OK, 12 rows affected (0.09 sec)
Records: 12  Duplicates: 0  Warnings: 0
```

insertinto course values('bio-101','intro to biology','4','biology'),('bio-301','genetics','4','biology'),('bio-399','computational biology','3','biology'),('cs-101','intro to computer science','4','comp.sci'),('cs-190','game design','4','comp.sci'),('cs-315','robotics','3','comp.sci'),('cs-319','image processing','3','comp.sci'),('cs-347','database system concepts','3','comp.sci'),('ee-181','intro to digital systems','3','elec.eng'),('fin-201','investment banking','3','finance'),('his-351','world history','3','history'),('mu-199','music video production','3','music'),('phy-101','physical principles','4','physics');



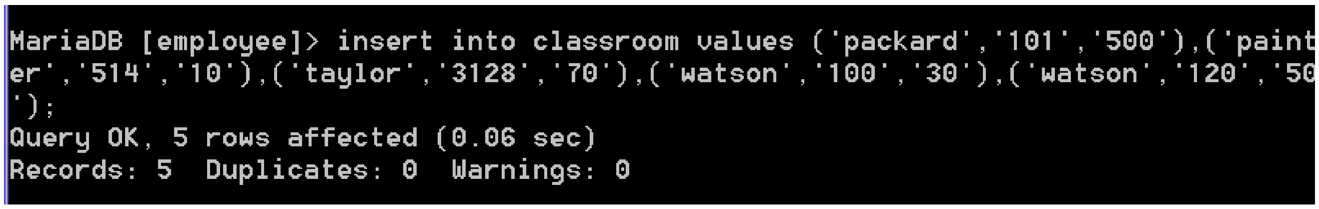
```
MariaDB [employee]> insert into course values('bio-101','intro to biology','4','
biology'),('bio-301','genetics','4','biology'),('bio-399','computational biology
','3','biology'),('cs-101','intro to computer science','4','comp.sci'),('cs-190'
,'game design','4','comp.sci'),('cs-315','robotics','3','comp.sci'),('cs-319','i
mage processing','3','comp.sci'),('cs-347','database system concepts','3','comp.
sci'),('ee-181','intro to digital systems','3','elec.eng'),('fin-201','investmen
t banking','3','finance'),('his-351','world history','3','history'),('mu-199','m
usic video production','3','music'),('phy-101','physical principles','4','physic
s');
Query OK, 13 rows affected (0.14 sec)
Records: 13  Duplicates: 0  Warnings: 0
```

insertinto prereq values('bio-301','bio-101'),('bio-399','bio-101'),('cs-190','cs-101'),('cs-315','cs-101'),('cs-319','cs-101'),('cs-347','cs-101'),('ee-181','phy-101');



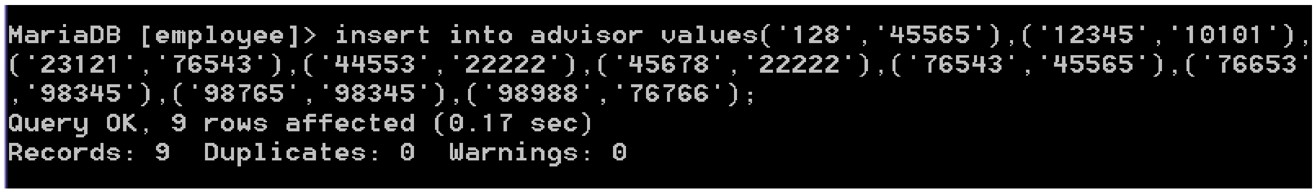
```
MariaDB [employee]> insert into prereq values('bio-301','bio-101'),('bio-399','b
io-101'),('cs-190','cs-101'),('cs-315','cs-101'),('cs-319','cs-101'),('cs-347','
cs-101'),('ee-181','phy-101');
Query OK, 7 rows affected (0.09 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

insertinto classroom values
('packard','101','500'),('painter','514','10'),('taylor','3128','70'),('watson','100','30'),
('watson','120','50');



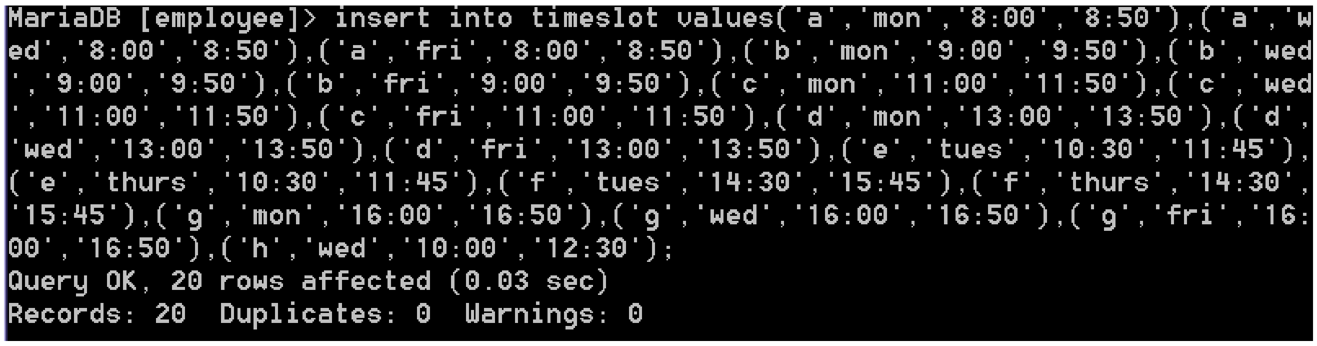
```
MariaDB [employee]> insert into classroom values ('packard','101','500'),('paint
er','514','10'),('taylor','3128','70'),('watson','100','30'),('watson','120','50
');
Query OK, 5 rows affected (0.06 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

insertinto advisor
values('128','45565'),('12345','10101'),('23121','76543'),('44553','22222'),('45678',
'22222'),('76543','45565'),('76653','98345'),('98765','98345'),('98988','76766');



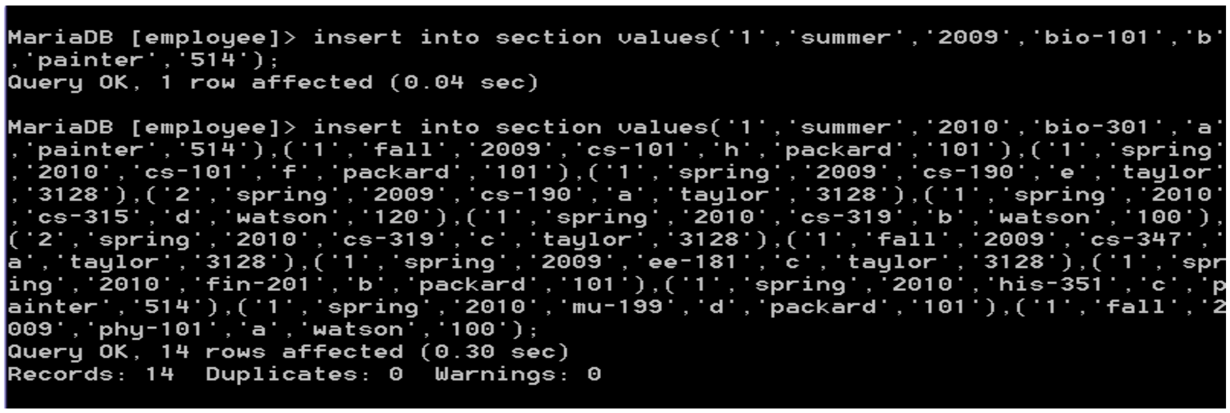
```
MariaDB [employee]> insert into advisor values('128','45565'),('12345','10101'),
('23121','76543'),('44553','22222'),('45678','22222'),('76543','45565'),('76653'
,'98345'),('98765','98345'),('98988','76766');
Query OK, 9 rows affected (0.17 sec)
Records: 9  Duplicates: 0  Warnings: 0
```

insertinto timeslot
values('a','mon','8:00','8:50'),('a','wed','8:00','8:50'),('a','fri','8:00','8:50'),('b','mon',
'9:00','9:50'),('b','wed','9:00','9:50'),('b','fri','9:00','9:50'),('c','mon','11:00','11:50'),(
'c','wed','11:00','11:50'),('c','fri','11:00','11:50'),('d','mon','13:00','13:50'),('d','wed',
'13:00','13:50'),('d','fri','13:00','13:50'),('e','tues','10:30','11:45'),('e','thurs','10:30','
11:45'),('f','tues','14:30','15:45'),('f','thurs','14:30','15:45'),('g','mon','16:00','16:50'
),('g','wed','16:00','16:50'),('g','fri','16:00','16:50'),('h','wed','10:00','12:30');



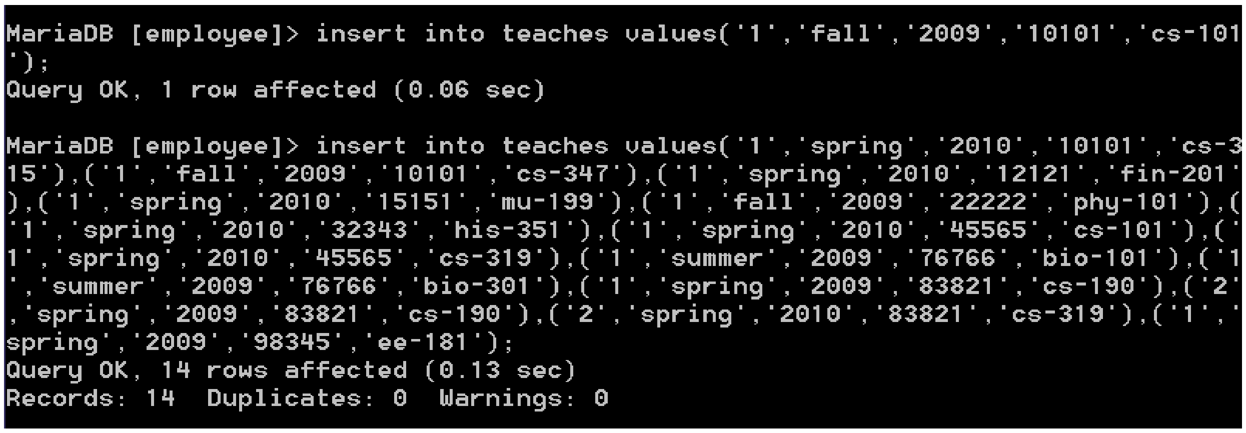
```
MariaDB [employee]> insert into timeslot values('a','mon','8:00','8:50'),('a','w
ed','8:00','8:50'),('a','fri','8:00','8:50'),('b','mon','9:00','9:50'),('b','wed
','9:00','9:50'),('b','fri','9:00','9:50'),('c','mon','11:00','11:50'),('c','wed
','11:00','11:50'),('c','fri','11:00','11:50'),('d','mon','13:00','13:50'),('d',
'wed','13:00','13:50'),('d','fri','13:00','13:50'),('e','tues','10:30','11:45'),
('e','thurs','10:30','11:45'),('f','tues','14:30','15:45'),('f','thurs','14:30',
'15:45'),('g','mon','16:00','16:50'),('g','wed','16:00','16:50'),('g','fri','16:
00','16:50'),('h','wed','10:00','12:30');
Query OK, 20 rows affected (0.03 sec)
Records: 20  Duplicates: 0  Warnings: 0
```

insertinto section values('1','summer','2009','bio-101','b','painter','514');
insertinto section values('1','summer','2010','bio-301','a','painter','514'),('1','fall','2009','cs-101','h','packard','101'),('1','spring','2010','cs-101','f','packard','101'),('1','spring','2009','cs-190','e','taylor','3128'),('2','spring','2009','cs-190','a','taylor','3128'),('1','spring','2010','cs-315','d','watson','120'),('1','spring','2010','cs-319','b','watson','100'),('2','spring','2010','cs-319','c','taylor','3128'),('1','fall','2009','cs-347','a','taylor','3128'),('1','spring','2009','ee-181','c','taylor','3128'),('1','spring','2010','fin-201','b','packard','101'),('1','spring','2010','his-351','c','painter','514'),('1','spring','2010','mu-199','d','packard','101'),('1','fall','2009','phy-101','a','watson','100');



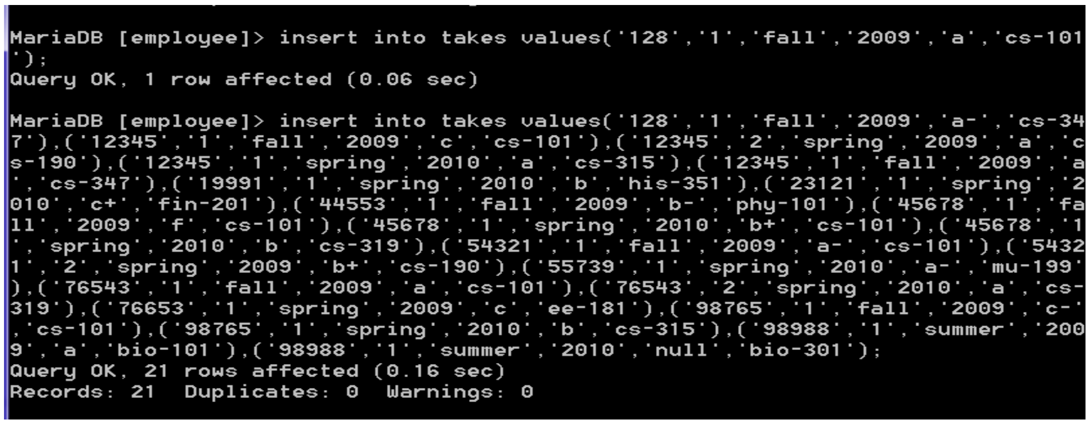
```
MariaDB [employee]> insert into section values('1','summer','2009','bio-101','b'
,'painter','514');
Query OK, 1 row affected (0.04 sec)

MariaDB [employee]> insert into section values('1','summer','2010','bio-301','a'
,'painter','514'),('1','fall','2009','cs-101','h','packard','101'),('1','spring'
,'2010','cs-101','f','packard','101'),('1','spring','2009','cs-190','e','taylor'
,'3128'),('2','spring','2009','cs-190','a','taylor','3128'),('1','spring','2010'
,'cs-315','d','watson','120'),('1','spring','2010','cs-319','b','watson','100'),
('2','spring','2010','cs-319','c','taylor','3128'),('1','fall','2009','cs-347','
a','taylor','3128'),('1','spring','2009','ee-181','c','taylor','3128'),('1','spr
ing','2010','fin-201','b','packard','101'),('1','spring','2010','his-351','c','p
ainter','514'),('1','spring','2010','mu-199','d','packard','101'),('1','fall','2
009','phy-101','a','watson','100');
Query OK, 14 rows affected (0.30 sec)
Records: 14  Duplicates: 0  Warnings: 0
```

insertinto teaches values('1','fall','2009','10101','cs-101');
insertinto teaches values('1','spring','2010','10101','cs-315'),('1','fall','2009','10101','cs-347'),('1','spring','2010','12121','fin-201'),('1','spring','2010','15151','mu-199'),('1','fall','2009','22222','phy-101'),('1','spring','2010','32343','his-351'),('1','spring','2010','45565','cs-101'),('1','spring','2010','45565','cs-319'),('1','summer','2009','76766','bio-101'),('1','summer','2009','76766','bio-301'),('1','spring','2009','83821','cs-190'),('2','spring','2009','83821','cs-190'),('2','spring','2010','83821','cs-319'),('1','spring','2009','98345','ee-181');



```
MariaDB [employee]> insert into teaches values('1','fall','2009','10101','cs-101
');
Query OK, 1 row affected (0.06 sec)

MariaDB [employee]> insert into teaches values('1','spring','2010','10101','cs-3
15'),('1','fall','2009','10101','cs-347'),('1','spring','2010','12121','fin-201'
),('1','spring','2010','15151','mu-199'),('1','fall','2009','22222','phy-101'),(
'1','spring','2010','32343','his-351'),('1','spring','2010','45565','cs-101'),('
1','spring','2010','45565','cs-319'),('1','summer','2009','76766','bio-101'),('1
','summer','2009','76766','bio-301'),('1','spring','2009','83821','cs-190'),('2'
,'spring','2009','83821','cs-190'),('2','spring','2010','83821','cs-319'),('1','
spring','2009','98345','ee-181');
Query OK, 14 rows affected (0.13 sec)
Records: 14  Duplicates: 0  Warnings: 0
```

insertinto takes values('128','1','fall','2009','a','cs-101');
insertinto takes values('128','1','fall','2009','a-','cs-347'),('12345','1','fall','2009','c','cs-101'),('12345','2','spring','2009','a','cs-190'),('12345','1','spring','2010','a','cs-315'),('12345','1','fall','2009','a','cs-347'),('19991','1','spring','2010','b','his-351'),('23121','1','spring','2010','c+','fin-201'),('44553','1','fall','2009','b-','phy-101'),('45678','1','fall','2009','f','cs-101'),('45678','1','spring','2010','b+','cs-101'),('45678','1','spring','2010','b','cs-319'),('54321','1','fall','2009','a-','cs-101'),('54321','2','spring','2009','b+','cs-190'),('55739','1','spring','2010','a-','mu-199'),('76543','1','fall','2009','a','cs-101'),('76543','2','spring','2010','a','cs-319'),('76653','1','spring','2009','c','ee-181'),('98765','1','fall','2009','c-','cs-101'),('98765','1','spring','2010','b','cs-315'),('98988','1','summer','2009','a','bio-101'),('98988','1','summer','2010','null','bio-301');



```
MariaDB [employee]> insert into takes values('128','1','fall','2009','a','cs-101
');
Query OK, 1 row affected (0.06 sec)

MariaDB [employee]> insert into takes values('128','1','fall','2009','a-','cs-34
7'),('12345','1','fall','2009','c','cs-101'),('12345','2','spring','2009','a','c
s-190'),('12345','1','spring','2010','a','cs-315'),('12345','1','fall','2009','a
','cs-347'),('19991','1','spring','2010','b','his-351'),('23121','1','spring','2
010','c+','fin-201'),('44553','1','fall','2009','b-','phy-101'),('45678','1','fa
ll','2009','f','cs-101'),('45678','1','spring','2010','b+','cs-101'),('45678','1
','spring','2010','b','cs-319'),('54321','1','fall','2009','a-','cs-101'),('5432
1','2','spring','2009','b+','cs-190'),('55739','1','spring','2010','a-','mu-199'
),('76543','1','fall','2009','a','cs-101'),('76543','2','spring','2010','a','cs-
319'),('76653','1','spring','2009','c','ee-181'),('98765','1','fall','2009','c-'
,'cs-101'),('98765','1','spring','2010','b','cs-315'),('98988','1','summer','200
9','a','bio-101'),('98988','1','summer','2010','null','bio-301');
Query OK, 21 rows affected (0.16 sec)
Records: 21  Duplicates: 0  Warnings: 0
```

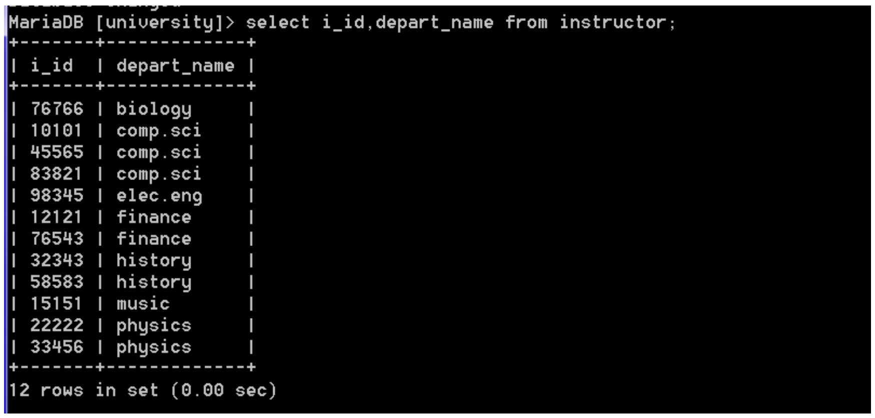
QUERIES:

1. **Find the department names of all instructors.**

QUERY:

select i_id,depart_name from instructor;

OUTPUT:



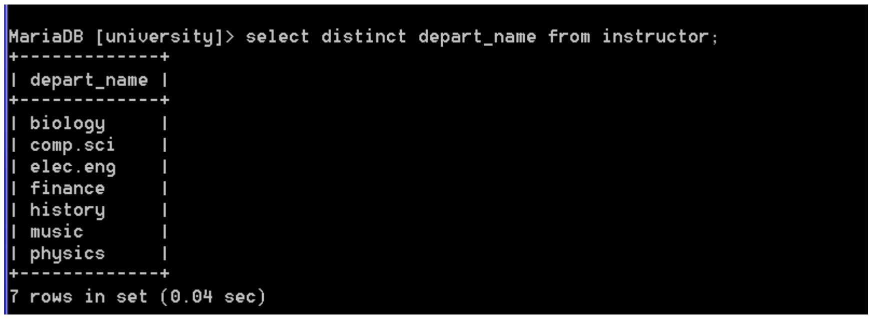
```
MariaDB [university]> select i_id,depart_name from instructor;
+-------+-------------+
| i_id  | depart_name |
+-------+-------------+
| 76766 | biology     |
| 10101 | comp.sci    |
| 45565 | comp.sci    |
| 83821 | comp.sci    |
| 98345 | elec.eng    |
| 12121 | finance     |
| 76543 | finance     |
| 32343 | history     |
| 58583 | history     |
| 15151 | music       |
| 22222 | physics     |
| 33456 | physics     |
+-------+-------------+
12 rows in set (0.00 sec)
```

2. **Refine previous query to eliminate duplicates (Question : What happens if " all" is used with select)**

QUERY:

selectdistinct depart_name from instructor;

OUTPUT:



```
MariaDB [university]> select distinct depart_name from instructor;
+-------------+
| depart_name |
+-------------+
| biology     |
| comp.sci    |
| elec.eng    |
| finance     |
| history     |
| music       |
| physics     |
+-------------+
7 rows in set (0.04 sec)
```

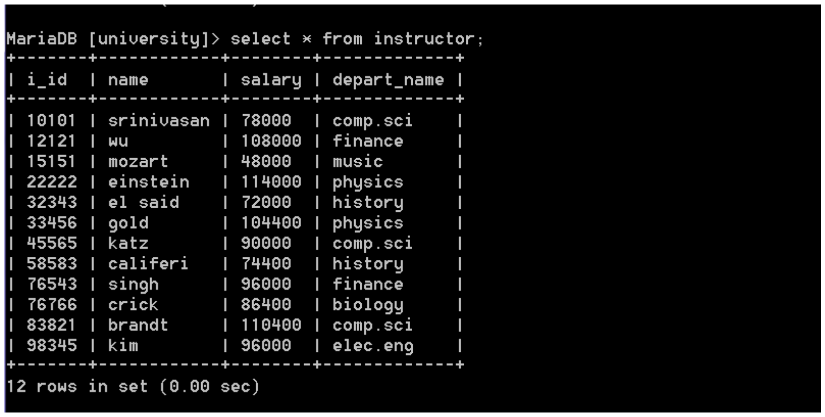
3. **Write query to raise 20% raise in salary for all instructors**

QUERY:

update instructor set salary=salary+(salary*20/100);

select * from instructor;

OUTPUT:



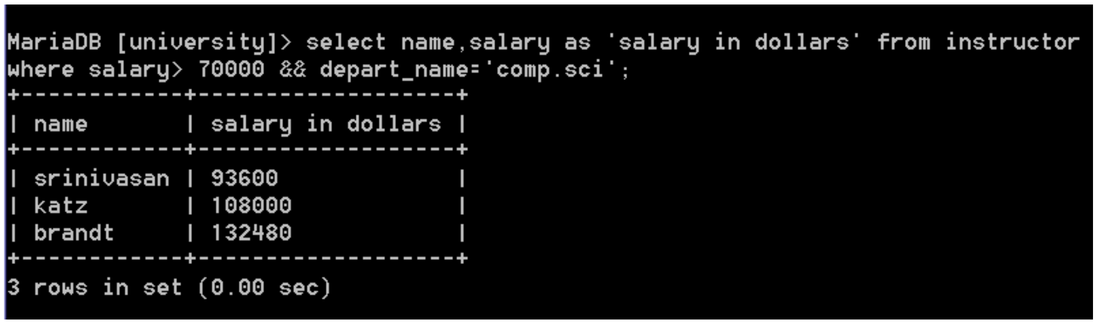
```
MariaDB [university]> select * from instructor;
+-------+------------+--------+-------------+
| i_id  | name       | salary | depart_name |
+-------+------------+--------+-------------+
| 10101 | srinivasan | 78000  | comp.sci    |
| 12121 | wu         | 108000 | finance     |
| 15151 | mozart     | 48000  | music       |
| 22222 | einstein   | 114000 | physics     |
| 32343 | el said    | 72000  | history     |
| 33456 | gold       | 104400 | physics     |
| 45565 | katz       | 90000  | comp.sci    |
| 58583 | califeri   | 74400  | history     |
| 76543 | singh      | 96000  | finance     |
| 76766 | crick      | 86400  | biology     |
| 83821 | brandt     | 110400 | comp.sci    |
| 98345 | kim        | 96000  | elec.eng    |
+-------+------------+--------+-------------+
12 rows in set (0.00 sec)
```

4. Find thenames of all instructors in the Computer Science department who have salarygreater than $70,000

QUERY:

select name,salary as 'salary in dollars' from instructor where salary> 70000 && depart_name='comp.sci';

OUTPUT:



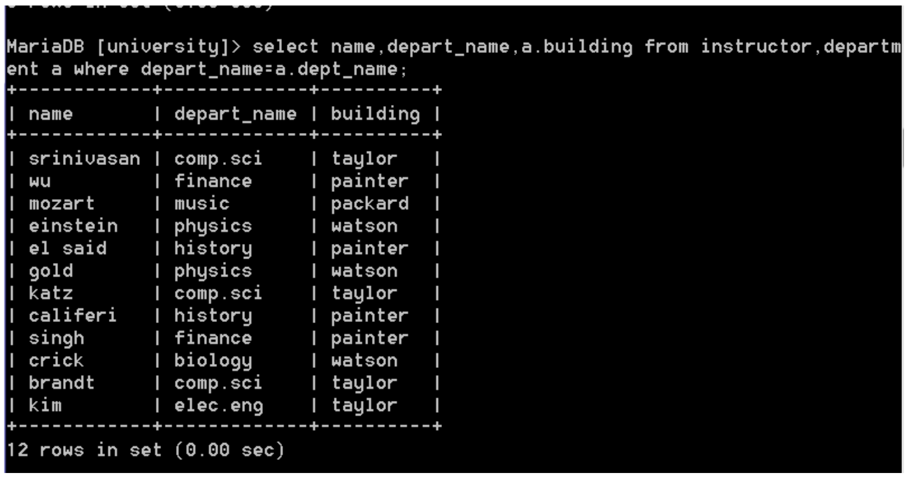
```
MariaDB [university]> select name,salary as 'salary in dollars' from instructor
where salary> 70000 && depart_name='comp.sci';
+------------+-------------------+
| name       | salary in dollars |
+------------+-------------------+
| srinivasan | 93600             |
| katz       | 108000            |
| brandt     | 132480            |
+------------+-------------------+
3 rows in set (0.00 sec)
```

5. Retrieve the namesof all instructors, along with their department names and department buildingname

QUERY:

select name,depart_name,a.building from instructor,department a where depart_name=a.dept_name;

OUTPUT:



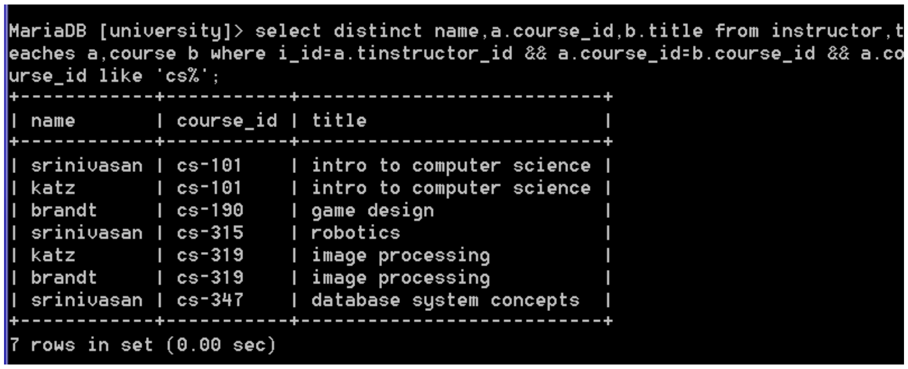
```
MariaDB [university]> select name,depart_name,a.building from instructor,departm
ent a where depart_name=a.dept_name;
+------------+-------------+----------+
| name       | depart_name | building |
+------------+-------------+----------+
| srinivasan | comp.sci    | taylor   |
| wu         | finance     | painter  |
| mozart     | music       | packard  |
| einstein   | physics     | watson   |
| el said    | history     | painter  |
| gold       | physics     | watson   |
| katz       | comp.sci    | taylor   |
| califeri   | history     | painter  |
| singh      | finance     | painter  |
| crick      | biology     | watson   |
| brandt     | comp.sci    | taylor   |
| kim        | elec.eng    | taylor   |
+------------+-------------+----------+
12 rows in set (0.00 sec)
```

6. List instructor names and course identifiers for instructors in the Computer Science department

QUERY:

selectdistinct name,a.course_id,b.title from instructor,teaches a,course b where i_id=a.tinstructor_id && a.course_id=b.course_id && a.course_id like 'cs%';

OUTPUT:



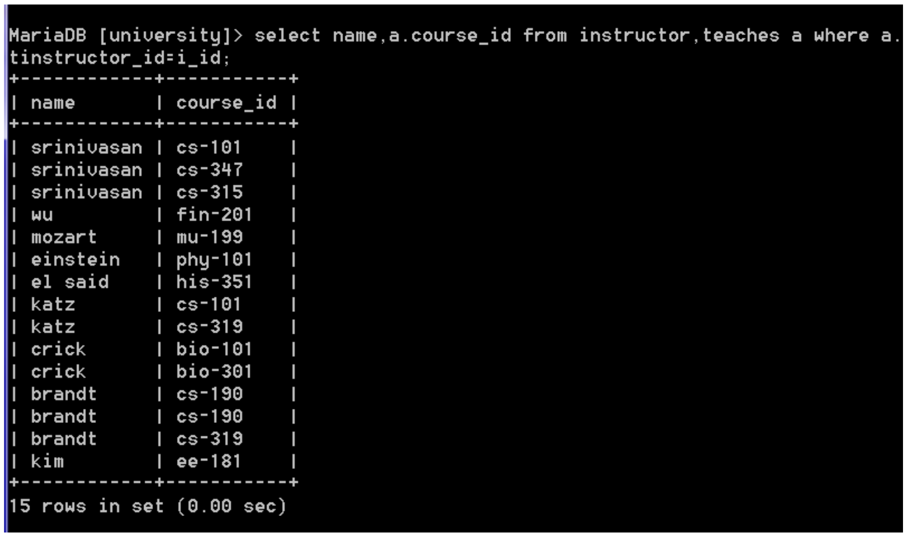
```
MariaDB [university]> select distinct name,a.course_id,b.title from instructor,t
eaches a,course b where i_id=a.tinstructor_id && a.course_id=b.course_id && a.co
urse_id like 'cs%';
+------------+-----------+--------------------------+
| name       | course_id | title                    |
+------------+-----------+--------------------------+
| srinivasan | cs-101    | intro to computer science |
| katz       | cs-101    | intro to computer science |
| brandt     | cs-190    | game design              |
| srinivasan | cs-315    | robotics                 |
| katz       | cs-319    | image processing         |
| brandt     | cs-319    | image processing         |
| srinivasan | cs-347    | database system concepts |
+------------+-----------+--------------------------+
7 rows in set (0.00 sec)
```

7. For all instructors in the university who have taughtsome course, find their names and the course ID of all courses they taught

QUERY:

select name,a.course_id from instructor,teaches a where a.tinstructor_id=i_id;

OUTPUT:



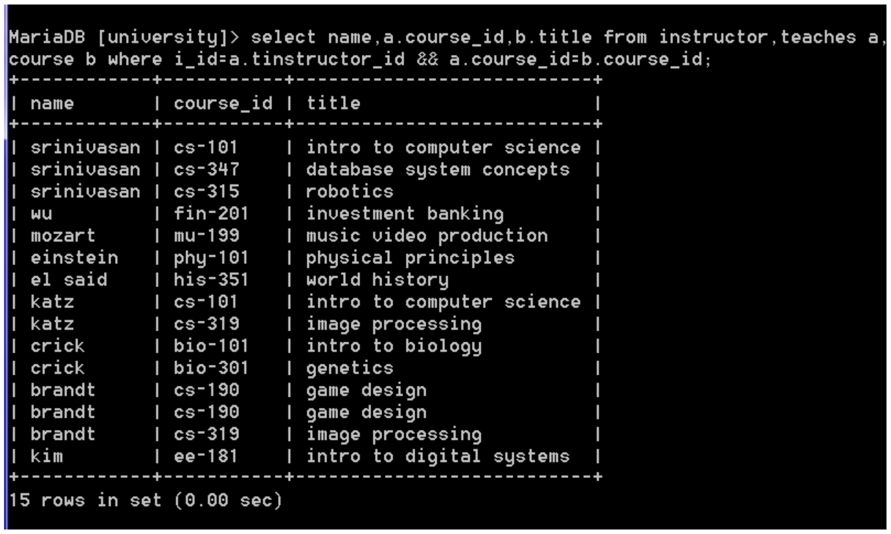
```
MariaDB [university]> select name,a.course_id from instructor,teaches a where a.
tinstructor_id=i_id;
+------------+-----------+
| name       | course_id |
+------------+-----------+
| srinivasan | cs-101    |
| srinivasan | cs-347    |
| srinivasan | cs-315    |
| wu         | fin-201   |
| mozart     | mu-199    |
| einstein   | phy-101   |
| el said    | his-351   |
| katz       | cs-101    |
| katz       | cs-319    |
| crick      | bio-101   |
| crick      | bio-301   |
| brandt     | cs-190    |
| brandt     | cs-190    |
| brandt     | cs-319    |
| kim        | ee-181    |
+------------+-----------+
15 rows in set (0.00 sec)
```

8. List the names of instructorsalong with the the titles of courses that they teach.

QUERY:

select name,a.course_id,b.title from instructor,teaches a,course b where i_id=a.tinstructor_id && a.course_id=b.course_id;

OUTPUT:



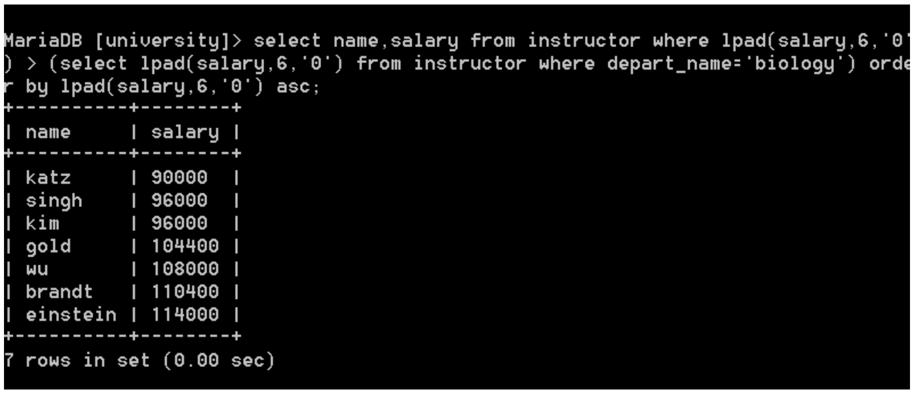
```
MariaDB [university]> select name,a.course_id,b.title from instructor,teaches a,
course b where i_id=a.tinstructor_id && a.course_id=b.course_id;
+------------+-----------+---------------------------+
| name       | course_id | title                     |
+------------+-----------+---------------------------+
| srinivasan | cs-101    | intro to computer science |
| srinivasan | cs-347    | database system concepts  |
| srinivasan | cs-315    | robotics                  |
| wu         | fin-201   | investment banking        |
| mozart     | mu-199    | music video production    |
| einstein   | phy-101   | physical principles       |
| el said    | his-351   | world history             |
| katz       | cs-101    | intro to computer science |
| katz       | cs-319    | image processing          |
| crick      | bio-101   | intro to biology          |
| crick      | bio-301   | genetics                  |
| brandt     | cs-190    | game design               |
| brandt     | cs-190    | game design               |
| brandt     | cs-319    | image processing          |
| kim        | ee-181    | intro to digital systems  |
+------------+-----------+---------------------------+
15 rows in set (0.00 sec)
```

9. **Find thenames of all instructors whose salary is greater than at least one instructor in the Biology department**

QUERY:

select name,salary from instructor wherelpad(salary,6,'0') > (selectlpad(salary,6,'0') from instructor where depart_name='biology') orderbylpad(salary,6,'0') asc;

OUTPUT:



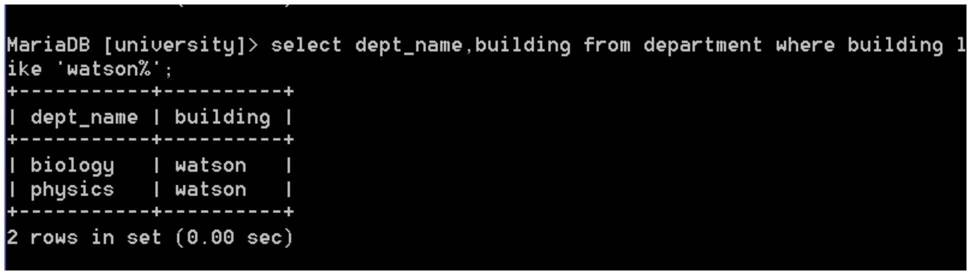
```
MariaDB [university]> select name,salary from instructor where lpad(salary,6,'0'
) > (select lpad(salary,6,'0') from instructor where depart_name='biology') orde
r by lpad(salary,6,'0') asc;
+----------+--------+
| name     | salary |
+----------+--------+
| katz     | 90000  |
| singh    | 96000  |
| kim      | 96000  |
| gold     | 104400 |
| wu       | 108000 |
| brandt   | 110400 |
| einstein | 114000 |
+----------+--------+
7 rows in set (0.00 sec)
```

10. **Find the names of all departments whose building name includes the substring'Watson'**

QUERY:

select dept_name,building from department where building like 'watson%';

OUTPUT:



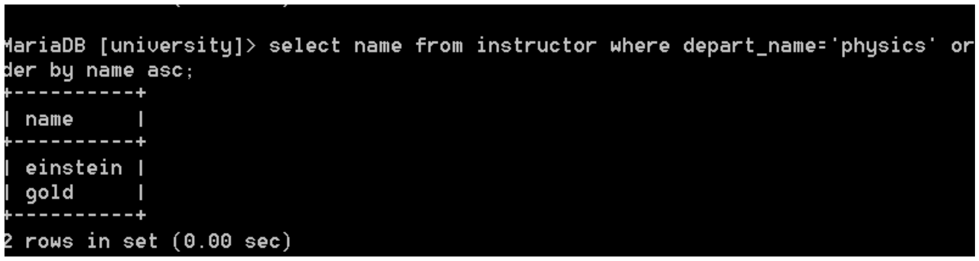
```
MariaDB [university]> select dept_name,building from department where building l
ike 'watson%';
+-----------+----------+
| dept_name | building |
+-----------+----------+
| biology   | watson   |
| physics   | watson   |
+-----------+----------+
2 rows in set (0.00 sec)
```

11. **list in alphabetic order all instructors in the Physicsdepartment**

QUERY:

select name from instructor where depart_name='physics' orderby name asc;

OUTPUT:



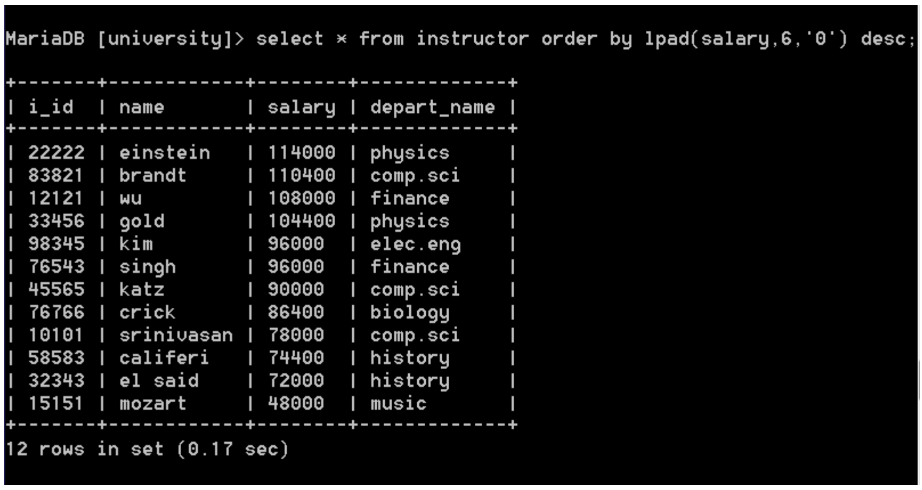
```
MariaDB [university]> select name from instructor where depart_name='physics' or
der by name asc;
+----------+
| name     |
+----------+
| einstein |
| gold     |
+----------+
2 rows in set (0.00 sec)
```

12. **list the entire instructor relation in descending order of salary.**

QUERY:

select * from instructor orderbylpad(salary,6,'0') desc;

OUTPUT:



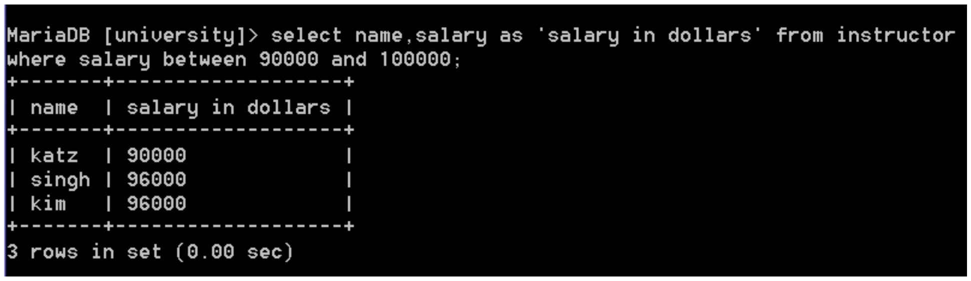
```
MariaDB [university]> select * from instructor order by lpad(salary,6,'0') desc;

+-------+------------+--------+-------------+
| i_id  | name       | salary | depart_name |
+-------+------------+--------+-------------+
| 22222 | einstein   | 114000 | physics     |
| 83821 | brandt     | 110400 | comp.sci    |
| 12121 | wu         | 108000 | finance     |
| 33456 | gold       | 104400 | physics     |
| 98345 | kim        | 96000  | elec.eng    |
| 76543 | singh      | 96000  | finance     |
| 45565 | katz       | 90000  | comp.sci    |
| 76766 | crick      | 86400  | biology     |
| 10101 | srinivasan | 78000  | comp.sci    |
| 58583 | califeri   | 74400  | history     |
| 32343 | el said    | 72000  | history     |
| 15151 | mozart     | 48000  | music       |
+-------+------------+--------+-------------+
12 rows in set (0.17 sec)
```

13. **find the names of instructors with salaryamounts between $90,000 and $100,000**

QUERY:

select name,salary as 'salary in dollars' from instructor where salary between 90000 and 100000;

OUTPUT:



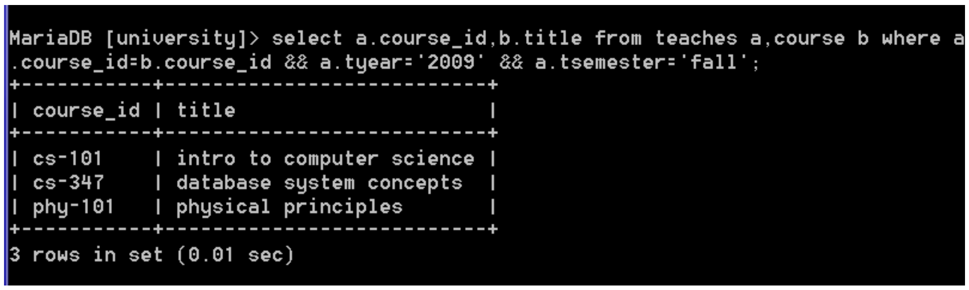
```
MariaDB [university]> select name,salary as 'salary in dollars' from instructor
where salary between 90000 and 100000;
+-------+-------------------+
| name  | salary in dollars |
+-------+-------------------+
| katz  | 90000             |
| singh | 96000             |
| kim   | 96000             |
+-------+-------------------+
3 rows in set (0.00 sec)
```

14. **List all courses taught in the Fall 2009 semester**

QUERY:

select a.course_id,b.title from teaches a,course b where a.course_id=b.course_id && a.tyear='2009' && a.tsemester='fall';

OUTPUT:



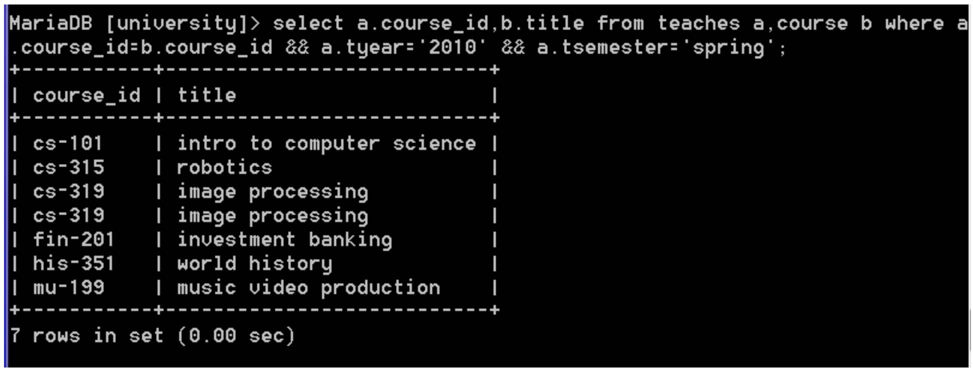
```
MariaDB [university]> select a.course_id,b.title from teaches a,course b where a
.course_id=b.course_id && a.tyear='2009' && a.tsemester='fall';
+-----------+---------------------------+
| course_id | title                     |
+-----------+---------------------------+
| cs-101    | intro to computer science |
| cs-347    | database system concepts  |
| phy-101   | physical principles       |
+-----------+---------------------------+
3 rows in set (0.01 sec)
```

15. **List all courses taught in the Spring 2010 semester**

QUERY:

OUTPUT:

select a.course_id,b.title from teaches a,course b where a.course_id=b.course_id && a.tyear='2010' && a.tsemester='spring';



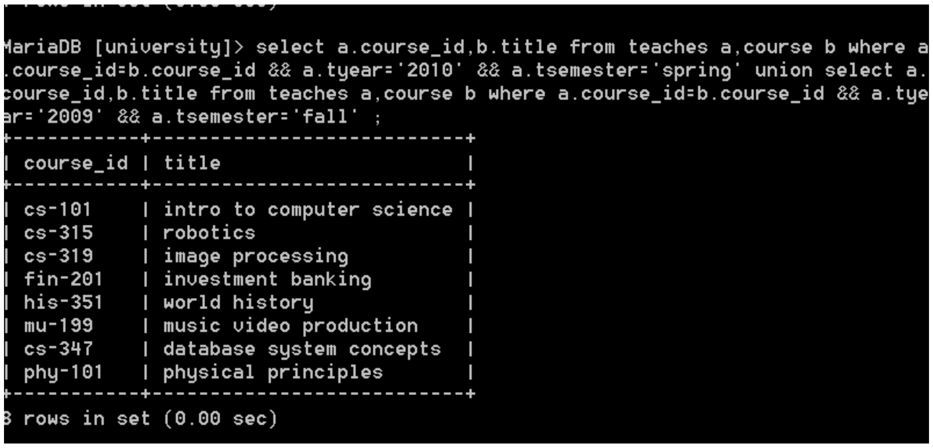
```
MariaDB [university]> select a.course_id,b.title from teaches a,course b where a
.course_id=b.course_id && a.tyear='2010' && a.tsemester='spring';
+-----------+---------------------------+
| course_id | title                     |
+-----------+---------------------------+
| cs-101    | intro to computer science |
| cs-315    | robotics                  |
| cs-319    | image processing          |
| cs-319    | image processing          |
| fin-201   | investment banking        |
| his-351   | world history             |
| mu-199    | music video production    |
+-----------+---------------------------+
7 rows in set (0.00 sec)
```

16. **To find the set of all courses taught either in Fall 2009 or in Spring 2010(Try using union and union all.)**

QUERY:

select a.course_id,b.title from teaches a,course b where a.course_id=b.course_id && a.tyear='2010' && a.tsemester='spring' unionselect a.course_id,b.title from teaches a,course b where a.course_id=b.course_id && a.tyear='2009' && a.tsemester='fall' ;

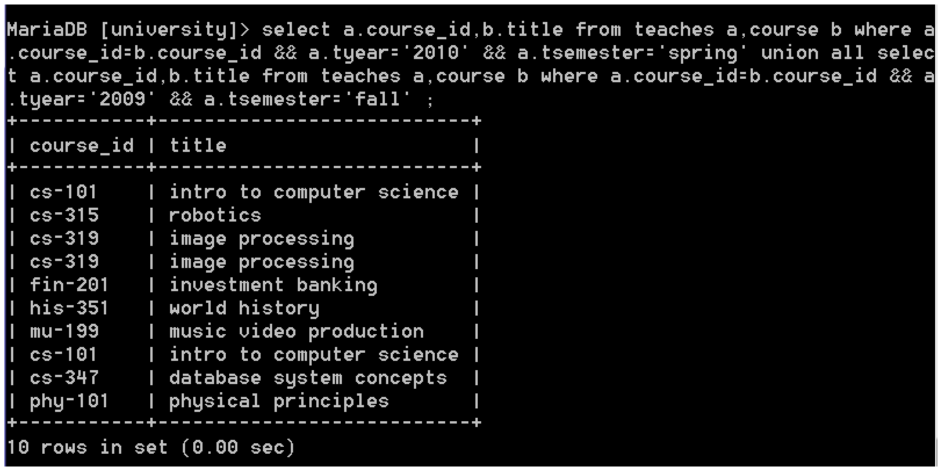
OUTPUT:



```
MariaDB [university]> select a.course_id,b.title from teaches a,course b where a
.course_id=b.course_id && a.tyear='2010' && a.tsemester='spring' union select a.
course_id,b.title from teaches a,course b where a.course_id=b.course_id && a.tye
ar='2009' && a.tsemester='fall' ;
+-----------+--------------------------+
| course_id | title                    |
+-----------+--------------------------+
| cs-101    | intro to computer science |
| cs-315    | robotics                 |
| cs-319    | image processing         |
| fin-201   | investment banking       |
| his-351   | world history            |
| mu-199    | music video production   |
| cs-347    | database system concepts |
| phy-101   | physical principles      |
+-----------+--------------------------+
8 rows in set (0.00 sec)
```

select a.course_id,b.title from teaches a,course b where a.course_id=b.course_id && a.tyear='2010' && a.tsemester='spring' unionallselect a.course_id,b.title from teaches a,course b where a.course_id=b.course_id && a.tyear='2009' && a.tsemester='fall' ;

OUTPUT:



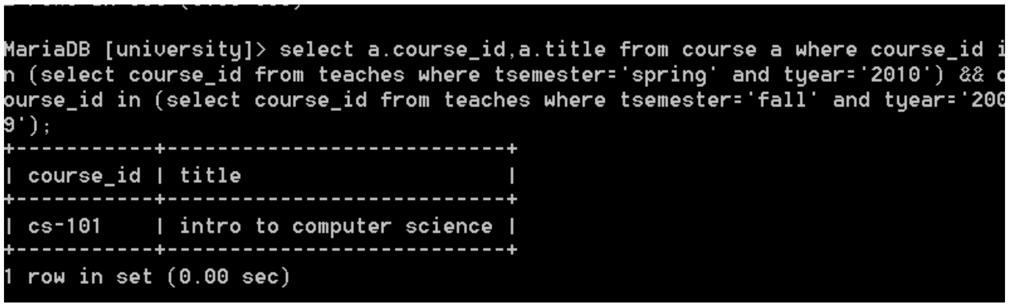
```
MariaDB [university]> select a.course_id,b.title from teaches a,course b where a
.course_id=b.course_id && a.tyear='2010' && a.tsemester='spring' union all selec
t a.course_id,b.title from teaches a,course b where a.course_id=b.course_id && a
.tyear='2009' && a.tsemester='fall' ;
+-----------+--------------------------+
| course_id | title                    |
+-----------+--------------------------+
| cs-101    | intro to computer science |
| cs-315    | robotics                 |
| cs-319    | image processing         |
| cs-319    | image processing         |
| fin-201   | investment banking       |
| his-351   | world history            |
| mu-199    | music video production   |
| cs-101    | intro to computer science |
| cs-347    | database system concepts |
| phy-101   | physical principles      |
+-----------+--------------------------+
10 rows in set (0.00 sec)
```

17. To find the set of all courses taught in the Fall 2009 as well as in Spring 2010(Try using intersect and intersect all)

QUERY:

select a.course_id,a.title from course a where course_id in (select course_id from teaches where tsemester='spring' and tyear='2010') && course_id in (select course_id from teaches where tsemester='fall' and tyear='2009');

OUTPUT:



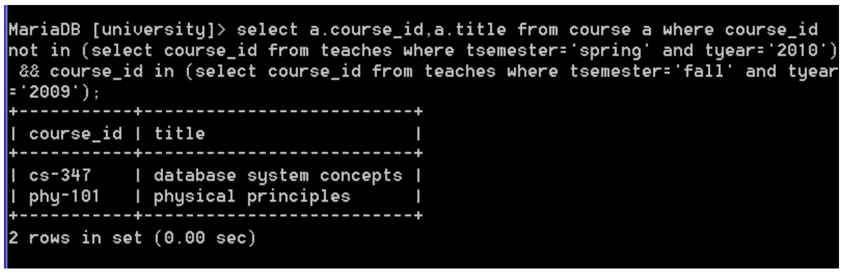
```
MariaDB [university]> select a.course_id,a.title from course a where course_id i
n (select course_id from teaches where tsemester='spring' and tyear='2010') && c
ourse_id in (select course_id from teaches where tsemester='fall' and tyear='200
9');
+-----------+--------------------------+
| course_id | title                    |
+-----------+--------------------------+
| cs-101    | intro to computer science |
+-----------+--------------------------+
1 row in set (0.00 sec)
```

20. To find all courses taught in the Fall 2009 semester but not in the Spring 2010semester(use except and except all)

QUERY:

select a.course_id,a.title from course a where course_id notin (select course_id from teaches where tsemester='spring' and tyear='2010') && course_id in (select course_id from teaches where tsemester='fall' and tyear='2009');

OUTPUT:



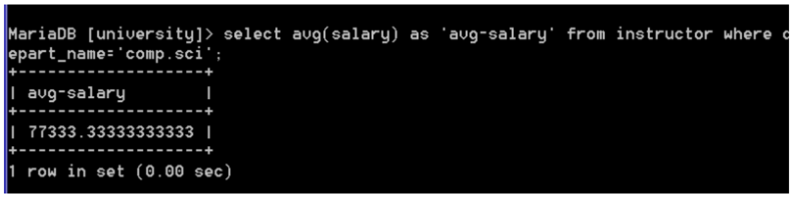
```
MariaDB [university]> select a.course_id,a.title from course a where course_id
not in (select course_id from teaches where tsemester='spring' and tyear='2010')
 && course_id in (select course_id from teaches where tsemester='fall' and tyear
='2009');
+-----------+-------------------------+
| course_id | title                   |
+-----------+-------------------------+
| cs-347    | database system concepts |
| phy-101   | physical principles     |
+-----------+-------------------------+
2 rows in set (0.00 sec)
```

21. Find the average salary of instructors in the Computer Science department."(Modify the query to use avg-salary as the columname for the average salary calculated.)

QUERY:

selectavg(salary) as 'avg-salary' from instructor where depart_name='comp.sci';

OUTPUT:



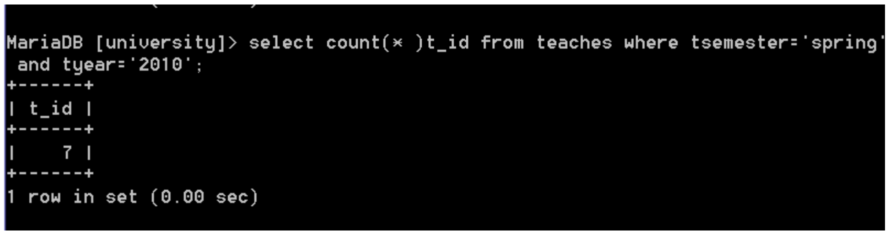
```
MariaDB [university]> select avg(salary) as 'avg-salary' from instructor where d
epart_name='comp.sci';
+-------------------+
| avg-salary        |
+-------------------+
| 77333.33333333333 |
+-------------------+
1 row in set (0.00 sec)
```

22. Find thetotal number of instructors who teach a course in the Spring 2010 semester."(Execute using count * ,count distinct)

QUERY:

selectcount(*)t_id from teaches where tsemester='spring' and tyear='2010';

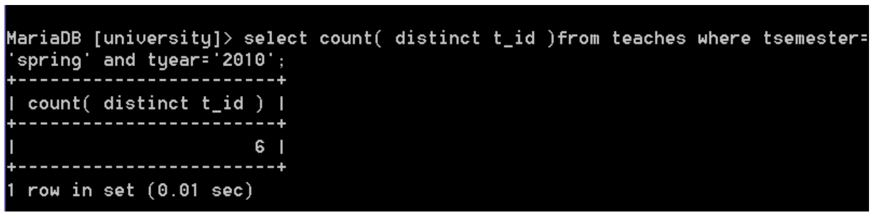
OUTPUT:



```
MariaDB [university]> select count(* )t_id from teaches where tsemester='spring'
 and tyear='2010';
+------+
| t_id |
+------+
|    7 |
+------+
1 row in set (0.00 sec)
```

selectcount(distinct t_id)from teaches where tsemester='spring' and tyear='2010';

OUTPUT:



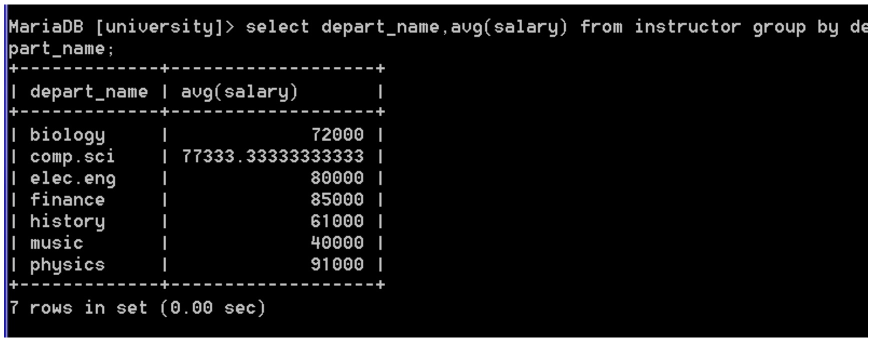
```
MariaDB [university]> select count( distinct t_id )from teaches where tsemester=
'spring' and tyear='2010';
+------------------------+
| count( distinct t_id ) |
+------------------------+
|                      6 |
+------------------------+
1 row in set (0.01 sec)
```

23. Find the average salary in each department. (use group by)

QUERY:

select depart_name,avg(salary) from instructor groupby depart_name;

OUTPUT:



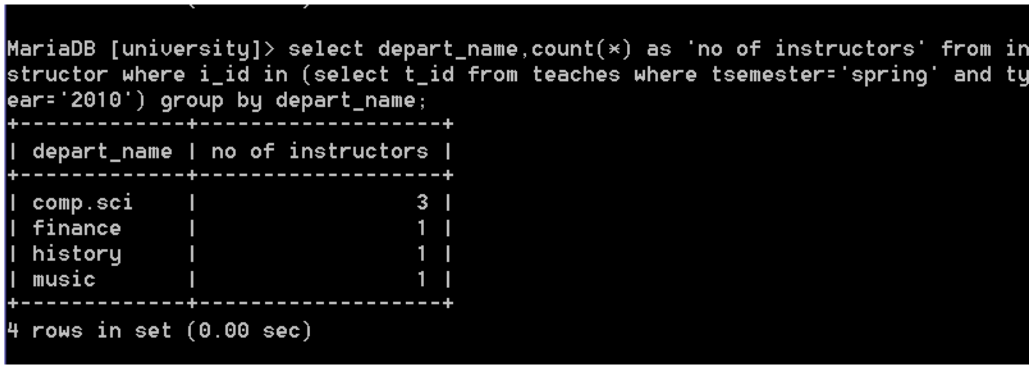
```
MariaDB [university]> select depart_name,avg(salary) from instructor group by de
part_name;
+-------------+-------------------+
| depart_name | avg(salary)       |
+-------------+-------------------+
| biology     |             72000 |
| comp.sci    | 77333.33333333333 |
| elec.eng    |             80000 |
| finance     |             85000 |
| history     |             61000 |
| music       |             40000 |
| physics     |             91000 |
+-------------+-------------------+
7 rows in set (0.00 sec)
```

24. Find the number of instructors in each department who teach a course in theSpring 2010 semester

QUERY:

select depart_name,count(*) as 'no of instructors' from instructor where i_id in (select t_id from teaches where tsemester='spring' and tyear='2010') groupby depart_name;

OUTPUT:



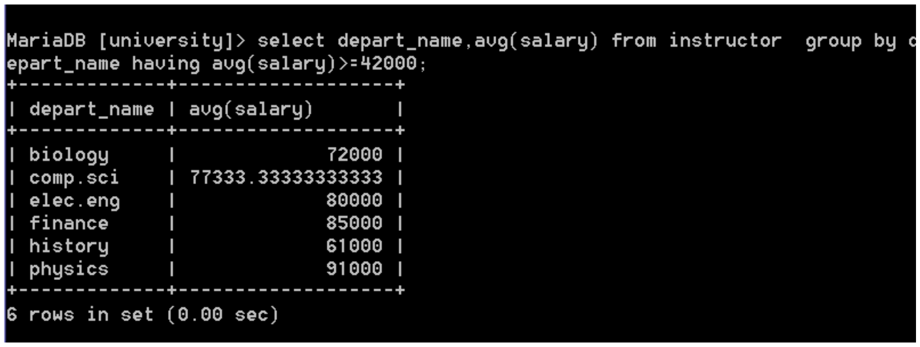
```
MariaDB [university]> select depart_name,count(*) as 'no of instructors' from in
structor where i_id in (select t_id from teaches where tsemester='spring' and ty
ear='2010') group by depart_name;
+-------------+-------------------+
| depart_name | no of instructors |
+-------------+-------------------+
| comp.sci    |                 3 |
| finance     |                 1 |
| history     |                 1 |
| music       |                 1 |
+-------------+-------------------+
4 rows in set (0.00 sec)
```

25. List departments where theaverage salary of the instructors is more than $42,000

QUERY:

select depart_name,avg(salary) from instructor groupby depart_name havingavg(salary)>=42000;

OUTPUT:



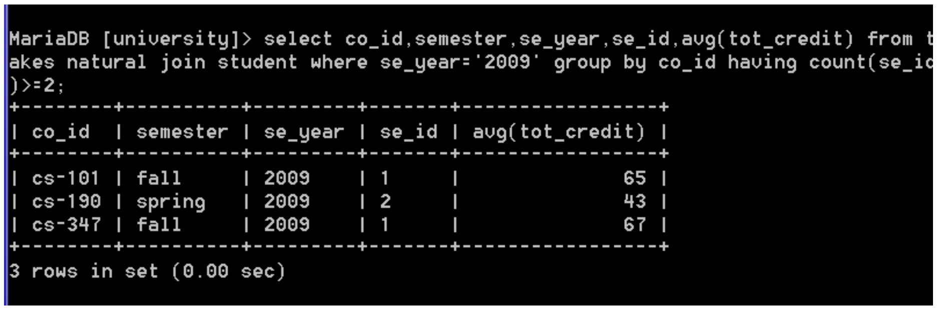
```
MariaDB [university]> select depart_name,avg(salary) from instructor  group by d
epart_name having avg(salary)>=42000;
+-------------+--------------------+
| depart_name | avg(salary)        |
+-------------+--------------------+
| biology     |              72000 |
| comp.sci    | 77333.333333333333 |
| elec.eng    |              80000 |
| finance     |              85000 |
| history     |              61000 |
| physics     |              91000 |
+-------------+--------------------+
6 rows in set (0.00 sec)
```

26. For each course section offered in 2009, find theaverage total credits (tot cred) of all students enrolled in the section, if the sectionhad at least 2 students.

QUERY:

select co_id,semester,se_year,se_id,avg(tot_credit) from takes naturaljoin student where se_year='2009' groupby co_id havingcount(se_id)>=2;

OUTPUT:



```
MariaDB [university]> select co_id,semester,se_year,se_id,avg(tot_credit) from t
akes natural join student where se_year='2009' group by co_id having count(se_id
)>=2;
+--------+----------+---------+-------+-----------------+
| co_id  | semester | se_year | se_id | avg(tot_credit) |
+--------+----------+---------+-------+-----------------+
| cs-101 | fall     | 2009    | 1     |              65 |
| cs-190 | spring   | 2009    | 2     |              43 |
| cs-347 | fall     | 2009    | 1     |              67 |
+--------+----------+---------+-------+-----------------+
3 rows in set (0.00 sec)
```

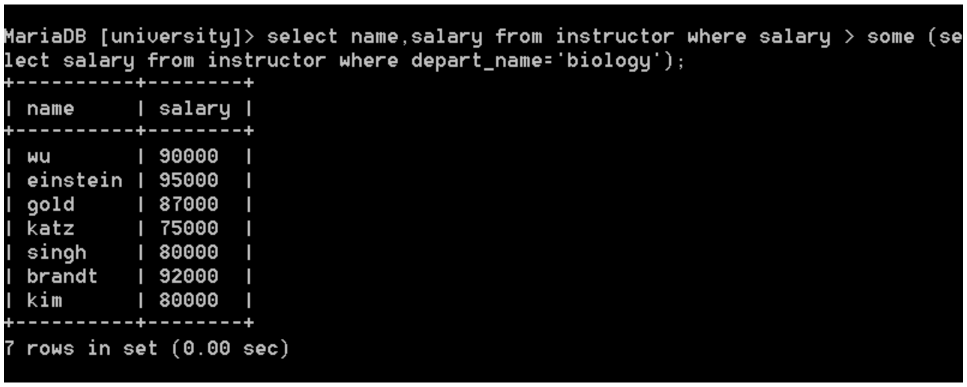
27. already done

28. Find the names of all instructors whose salary is greater than at least oneinstructor in the Biology department."(Execute using set comparision, " some" to solve the queries)

QUERY:

select name,salary from instructor where salary >some (select salary from instructor where depart_name='biology');

OUTPUT:



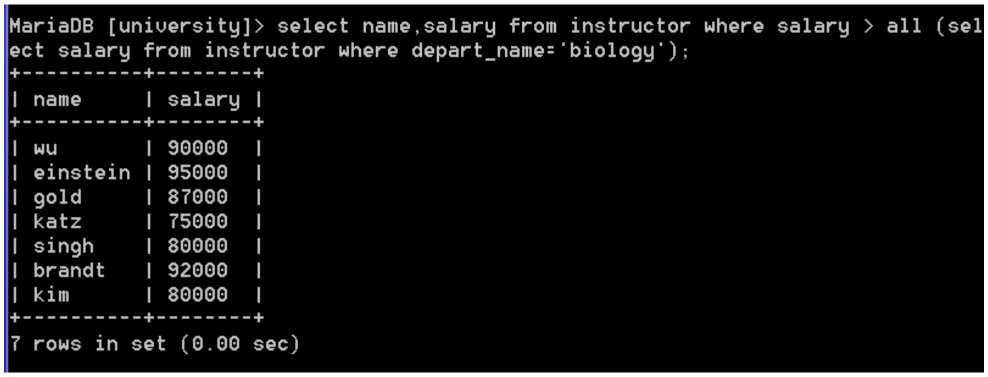
```
MariaDB [university]> select name,salary from instructor where salary > some (se
lect salary from instructor where depart_name='biology');
+----------+--------+
| name     | salary |
+----------+--------+
| wu       | 90000  |
| einstein | 95000  |
| gold     | 87000  |
| katz     | 75000  |
| singh    | 80000  |
| brandt   | 92000  |
| kim      | 80000  |
+----------+--------+
7 rows in set (0.00 sec)
```

29. **Find the names of all instructorsthat have a salary value greater than that of each instructor in the Biology department. (Use "all" to solve the query)**

QUERY:

select name,salary from instructor where salary >all (select salary from instructor where depart_name='biology');

OUTPUT:



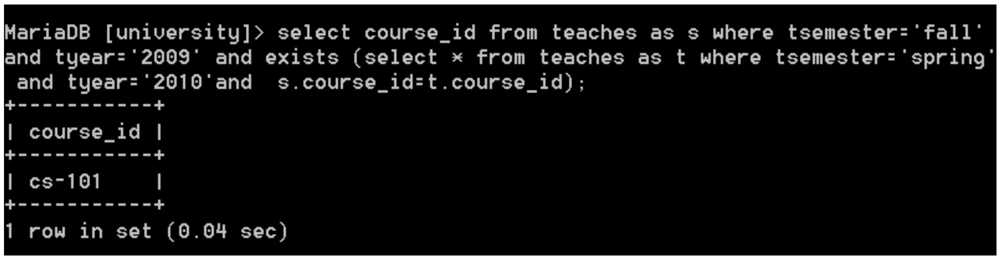
```
MariaDB [university]> select name,salary from instructor where salary > all (sel
ect salary from instructor where depart_name='biology');
+----------+--------+
| name     | salary |
+----------+--------+
| wu       | 90000  |
| einstein | 95000  |
| gold     | 87000  |
| katz     | 75000  |
| singh    | 80000  |
| brandt   | 92000  |
| kim      | 80000  |
+----------+--------+
7 rows in set (0.00 sec)
```

30. **Find all courses taught in boththe Fall 2009 semester and in the Spring 2010 semester" (use exists)**

QUERY:

select course_id from teaches as s where tsemester='fall' and tyear='2009' andexists (select * from teaches as t where tsemester='spring' and tyear='2010'and s.course_id=t.course_id);

OUTPUT:



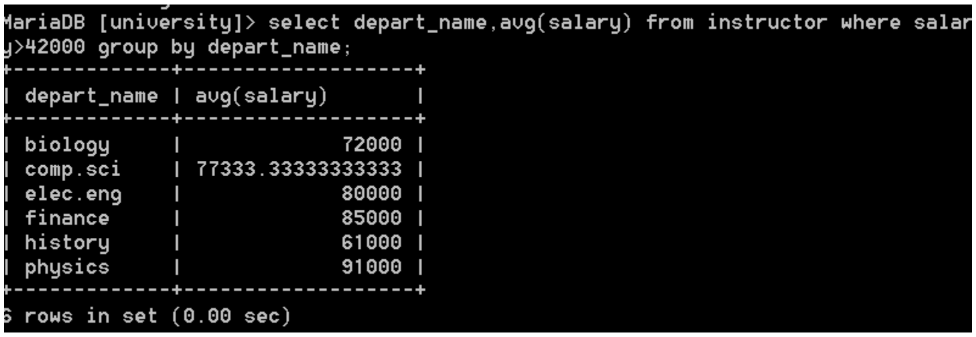
```
MariaDB [university]> select course_id from teaches as s where tsemester='fall'
and tyear='2009' and exists (select * from teaches as t where tsemester='spring'
 and tyear='2010'and  s.course_id=t.course_id);
+-----------+
| course_id |
+-----------+
| cs-101    |
+-----------+
1 row in set (0.04 sec)
```

31. Find the average instructors' salaries of those departments where the average salary is greater than $42,000."(Write query without using "having" clause)

QUERY:

select depart_name,avg(salary) from instructor where salary>42000 groupby depart_name;

OUTPUT:



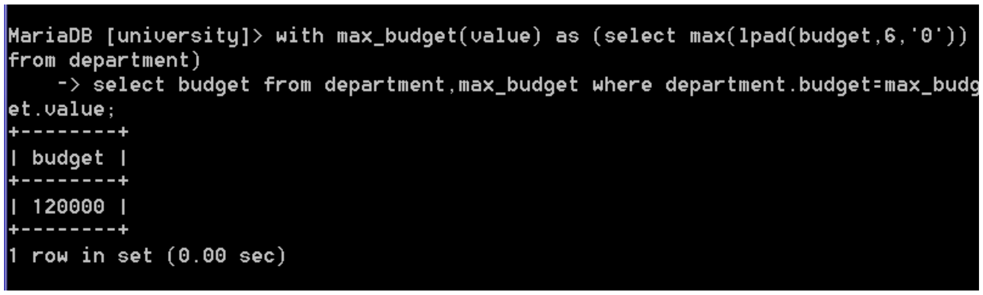
```
lariaDB [university]> select depart_name,avg(salary) from instructor where salar
y>42000 group by depart_name;
+-------------+------------------+
| depart_name | avg(salary)      |
+-------------+------------------+
| biology     |            72000 |
| comp.sci    | 77333.3333333333 |
| elec.eng    |            80000 |
| finance     |            85000 |
| history     |            61000 |
| physics     |            91000 |
+-------------+------------------+
6 rows in set (0.00 sec)
```

32. Find those departments with the maximum budget.(Solve using "with" clause)

QUERY:

with max_budget(value) as (selectmax(lpad(budget,6,'0')) from department)
select budget from department,max_budget where
department.budget=max_budget.value;

OUTPUT:



```
MariaDB [university]> with max_budget(value) as (select max(lpad(budget,6,'0'))
from department)
    -> select budget from department,max_budget where department.budget=max_budg
et.value;
+--------+
| budget |
+--------+
| 120000 |
+--------+
1 row in set (0.00 sec)
```

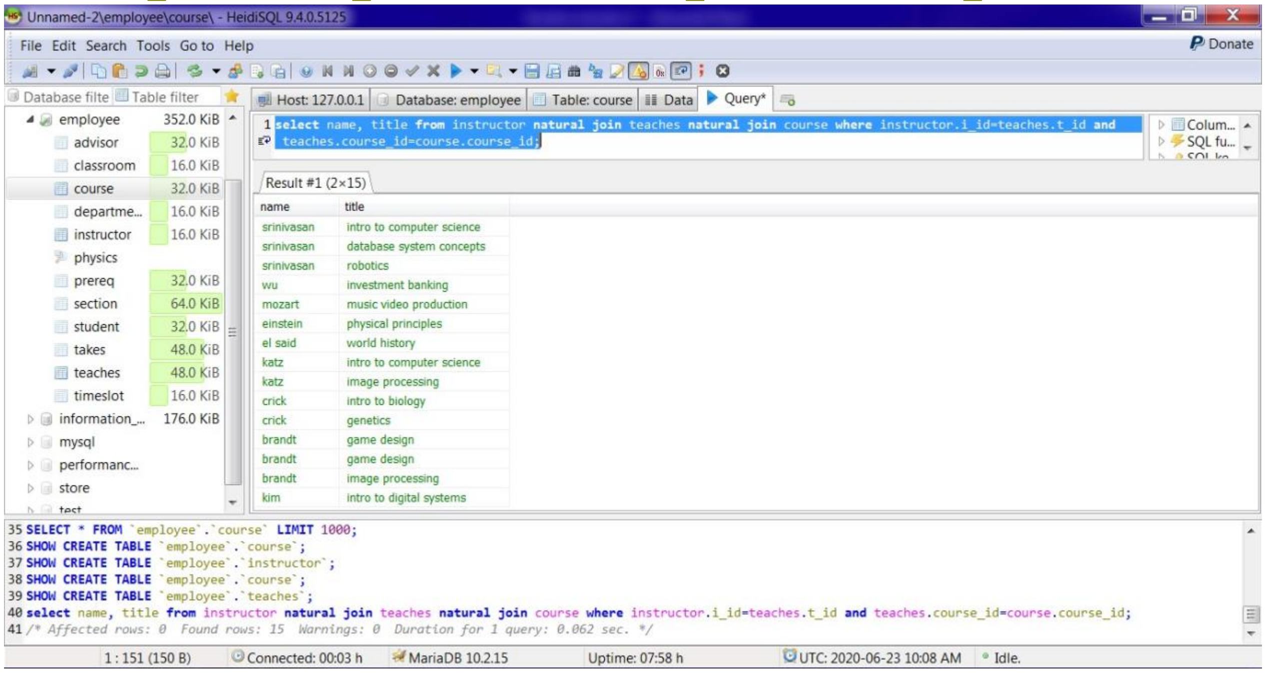
RESULT:

Thus the queries for university exercise was executed and verified successfully.

QUIZ QUESTIONS:

1. What is the output of this query

- **“select name, title from instructor natural join teaches natural join course;”**
- **select**name,title**from**instructor**naturaljoin**teaches**naturaljoin**course**where**instructor.i_id=teaches.t_id**and**teaches.course_id=course.course_id;



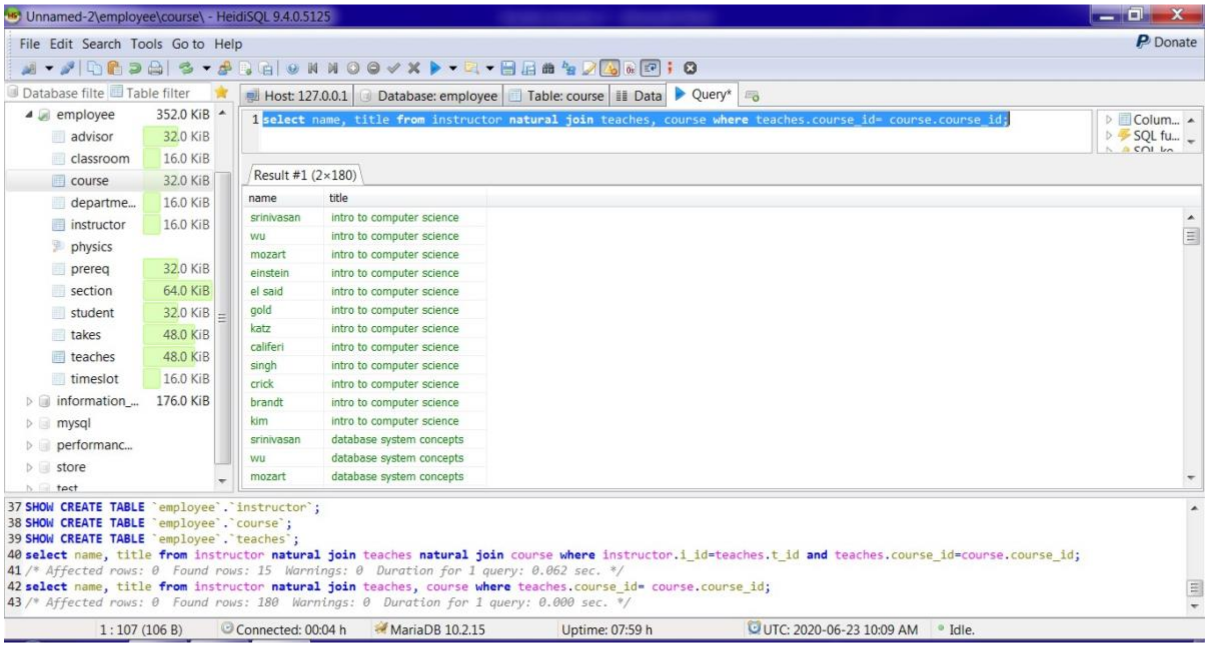
```
select name, title from instructor natural join teaches natural join course where instructor.i_id=teaches.t_id and
 teaches.course_id=course.course_id;
```

Result #1 (2×15)

name	title
srinivasan	intro to computer science
srinivasan	database system concepts
srinivasan	robotics
wu	investment banking
mozart	music video production
einstein	physical principles
el said	world history
katz	intro to computer science
katz	image processing
crick	intro to biology
crick	genetics
brandt	game design
brandt	game design
brandt	image processing
kim	intro to digital systems

```
SELECT * FROM `employee`.`course` LIMIT 1000;
SHOW CREATE TABLE `employee`.`course`;
SHOW CREATE TABLE `employee`.`instructor`;
SHOW CREATE TABLE `employee`.`course`;
SHOW CREATE TABLE `employee`.`teaches`;
select name, title from instructor natural join teaches natural join course where instructor.i_id=teaches.t_id and teaches.course_id=course.course_id;
/* Affected rows: 0  Found rows: 15  Warnings: 0  Duration for 1 query: 0.062 sec. */
```

- **“select name, title from instructor natural join teaches, course where teaches.course id= course.course id;”**



```
select name, title from instructor natural join teaches, course where teaches.course_id= course.course_id;
```

Result #1 (2×180)

name	title
srinivasan	intro to computer science
wu	intro to computer science
mozart	intro to computer science
einstein	intro to computer science
el said	intro to computer science
gold	intro to computer science
katz	intro to computer science
califeri	intro to computer science
singh	intro to computer science
crick	intro to computer science
brandt	intro to computer science
kim	intro to computer science
srinivasan	database system concepts
wu	database system concepts
mozart	database system concepts

```
SHOW CREATE TABLE `employee`.`instructor`;
SHOW CREATE TABLE `employee`.`course`;
SHOW CREATE TABLE `employee`.`teaches`;
select name, title from instructor natural join teaches natural join course where instructor.i_id=teaches.t_id and teaches.course_id=course.course_id;
/* Affected rows: 0  Found rows: 15  Warnings: 0  Duration for 1 query: 0.062 sec. */
select name, title from instructor natural join teaches, course where teaches.course_id= course.course_id;
/* Affected rows: 0  Found rows: 180  Warnings: 0  Duration for 1 query: 0.000 sec. */
```

1 : 107 (106 B) Connected: 00:04 h MariaDB 10.2.15 Uptime: 07:59 h UTC: 2020-06-23 10:09 AM Idle.

2. “select name, title from (instructor natural join teaches) join course using (course id);” 4. What type of strings will be matched

like ’ab\%cd%’

like ’ab\\cd%’

selectname,titlefrom(instructornaturaljointeaches)joincourseusing(course_id)WHEREtitlelike'ab\%cd%';
null

selectname,titlefrom(instructornaturaljointeaches)joincourseusing(course_id)WHEREtitlelike'ab\\%cd%';

null

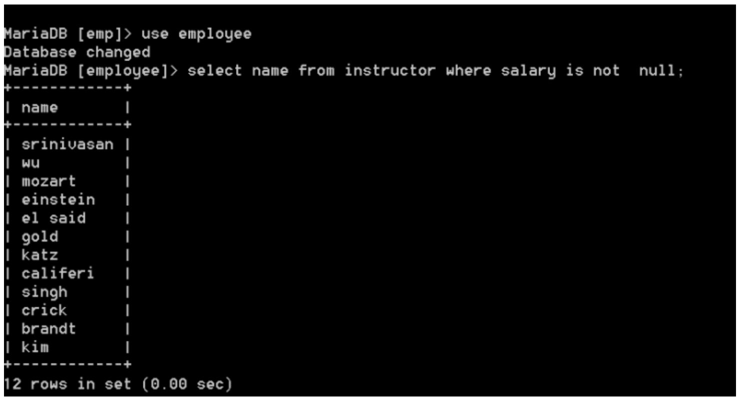
3. Write the output of the query

- **select name from instructor where salary is null;**

selectnamefrominstructorwheresalaryisnull;
output:null

- **select name from instructor where salary is not null;**

selectname**from**instructor**where**salary**is** not **null**;



```
MariaDB [emp]> use employee
Database changed
MariaDB [employee]> select name from instructor where salary is not  null;
+------------+
| name       |
+------------+
| srinivasan |
| wu         |
| mozart     |
| einstein   |
| el said    |
| gold       |
| katz       |
| califeri   |
| singh      |
| crick      |
| brandt     |
| kim        |
+------------+
12 rows in set (0.00 sec)
```

4.How do all aggregation functions work with NULL and Boolean values?

AVG:

```
AVG([DISTINCT] expression)  [OVER (...)]
```

Description

Returns the average of non-NULL input values, or NaN if the input contains a NaN.

Supported Argument Types

Any numeric input type, such as INT64. Note that, for floating point input types, the return result is non-deterministic, which means you might receive a different result each time you use this function.

Optional Clauses

The clauses are applied *in the following order*:

1. OVER: Specifies a window. See Analytic Functions. This clause is currently incompatible with all other clauses within AVG().

2. DISTINCT: Each distinct value of expression is aggregated only once into the result.

Returned Data Types

- NUMERIC if the input type is NUMERIC.
- FLOAT64

Examples

```
SELECT AVG(x) as avg
FROMUNNEST([0, 2, 4, 4, 5]) as x;

+.......+
| avg |
+.......+
| 3  |
+-----+

SELECTAVG(DISTINCT x) AS avg
FROM UNNEST([0, 2, 4, 4, 5]) AS x;

+.........+
| avg  |
+------+
| 2.75 |
+.........+

SELECT
  x,
  AVG(x) OVER (ORDERBY x ROWS BETWEEN1 PRECEDING ANDCURRENT ROW) AS
avg
FROMUNNEST([0, 2, NULL, 4, 4, 5]) AS x;

+.........+.........+
| x    | avg |
+.........+.........+
| NULL | NULL |
| 0   | 0   |
| 2   | 1   |
| 4   | 3   |
| 4   | 4   |
| 5   | 4.5 |
+-----+-----+
```

BIT_AND

```
BIT_AND(expression)
```

Description

Performs a bitwise AND operation on expression and returns the result.

Supported Argument Types

- INT64

Returned Data Types

INT64

Examples

```
SELECT BIT_AND(x) as bit_and FROMUNNEST([0xF001, 0x00A1]) as x;

+---------+
| bit_and |
+---------+
| 1       |
+---------+
```

BIT_OR

```
BIT_OR(expression)
```

Description

Performs a bitwise OR operation on expression and returns the result.

Supported Argument Types

- INT64

Returned Data Types

INT64

Examples

```
SELECT BIT_OR(x) as bit_or FROMUNNEST([0xF001, 0x00A1]) as x;

+.........+
| bit_or |
+.........+
| 61601  |
+.........+
```

BIT_XOR

```
BIT_XOR([DISTINCT] expression)
```

Description

Performs a bitwise XOR operation on expression and returns the result.

Supported Argument Types

- INT64

Optional Clause

DISTINCT: Each distinct value of expression is aggregated only once into the result.

Returned Data Types

INT64

Examples

```
SELECT BIT_XOR(x) AS bit_xor FROMUNNEST([5678, 1234]) AS x;

+............+
| bit_xor |
+............+
| 4860    |
+............+

SELECT BIT_XOR(x) AS bit_xor FROMUNNEST([1234, 5678, 1234]) AS x;

+............+
| bit_xor |
+............+
| 5678    |
+............+

SELECT BIT_XOR(DISTINCT x) AS bit_xor FROM UNNEST([1234, 5678, 1234]) AS x;

+............+
| bit_xor |
+............+
| 4860    |
+............+
```

COUNT

1. COUNT(*) [OVER (...)]

2. COUNT([DISTINCT] expression) [OVER (...)]

Description

1. Returns the number of rows in the input.

2. Returns the number of rows with expression evaluated to any value other than NULL.

Supported Argument Types

expression can be any data type.

Optional Clauses

The clauses are applied *in the following order*:

1. OVER: Specifies a window. See Analytic Functions.
2. DISTINCT: Each distinct value of expression is aggregated only once into the result.

Return Data Types

INT64

Examples

```
SELECT
  COUNT(*) AS count_star,
COUNT(DISTINCT x) AS count_dist_x
FROM UNNEST([1, 4, 4, 5]) AS x;

+------------+--------------+
| count_star | count_dist_x |
+------------+--------------+
| 4          | 3            |
+------------+--------------+

SELECT
  x,
  COUNT(*) OVER (PARTITION BY MOD(x, 3)) AS count_star,
COUNT(DISTINCT x) OVER (PARTITION BY MOD(x, 3)) AS count_dist_x
FROM UNNEST([1, 4, 4, 5]) AS x;

+------+------------+--------------+
| x    | count_star | count_dist_x |
+------+------------+------------+
| 1    | 3          | 2          |
| 4    | 3          | 2          |
| 4    | 3          | 2          |
| 5    | 1          | 1          |
+------+------------+------------+

SELECT
  x,
  COUNT(*) OVER (PARTITION BY MOD(x, 3)) AS count_star,
COUNT(x) OVER (PARTITION BY MOD(x, 3)) AS count_x
FROM UNNEST([1, 4, NULL, 4, 5]) AS x;


+------+------------+---------+
| x    | count_star | count_x |
+------+------------+---------+
| NULL | 1          | 0       |
| 1    | 3          | 3       |
| 4    | 3          | 3       |
| 4    | 3          | 3       |
| 5    | 1          | 1       |
+------+------------+---------+
```

COUNTIF

```
COUNTIF(expression)  [OVER (...)]
```

Description

Returns the count of TRUE values for expression. Returns 0 if there are zero input rows, or if expression evaluates to FALSE or NULL for all rows.

Supported Argument Types

BOOL

Optional Clause

OVER: Specifies a window. See Analytic Functions.

Return Data Types

INT64

Examples

```
SELECT COUNTIF(x<0) AS num_negative, COUNTIF(x>0) AS num_positive
FROMUNNEST([5, -2, 3, 6, -10, -7, 4, 0]) AS x;

+--------------+--------------+
| num_negative | num_positive |
+--------------+--------------+
| 3            | 4            |
+--------------+--------------+

SELECT
  x,
  COUNTIF(x<0) OVER (ORDERBY ABS(x) ROWS BETWEEN1 PRECEDING AND1
FOLLOWING) AS num_negative
FROM UNNEST([5, -2, 3, 6, -10, NULL, -7, 4, 0]) AS x;

+------+--------------+
| x    | num_negative |
+------+--------------+
| NULL | 0            |
| 0    | 1            |
| -2   | 1            |
| 3    | 1            |
| 4    | 0            |
| 5    | 0            |
| 6    | 1            |
| -7   | 2            |
| -10  | 2            |
+------+--------------+
```

LOGICAL_AND

```
LOGICAL_AND(expression)
```

Description

Returns the logical AND of all non-NULL expressions. Returns NULL if there are zero input rows or expression evaluates to NULL for all rows.

Supported Argument Types

BOOL

Return Data Types

BOOL

Examples

```
SELECT LOGICAL_AND(x) AS logical_and FROMUNNEST([true, false, true]) AS x;

+-------------+
| logical_and |
+-------------+
| false       |
+-------------+
```

LOGICAL_OR

```
LOGICAL_OR(expression)
```

Description

Returns the logical OR of all non-NULL expressions. Returns NULL if there are zero input rows or expression evaluates to NULL for all rows.

Supported Argument Types

BOOL

Return Data Types

BOOL

Examples

```
SELECT LOGICAL_OR(x) AS logical_or FROMUNNEST([true, false, true]) AS x;

+------------+
| logical_or |
+------------+
| true       |
+------------+
```

MAX

```
MAX(expression)  [OVER (...)]
```

Description

Returns the maximum value of non-NULL expressions. Returns NULL if there are zero input rows or expression evaluates to NULL for all rows. Returns NaN if the input contains a NaN.

Supported Argument Types

Any data type except: ARRAY STRUCT

Optional Clause

OVER: Specifies a window. See Analytic Functions.

Return Data Types

Same as the data type used as the input values.

Examples

```
SELECT MAX(x) AS max
FROMUNNEST([8, 37, 4, 55]) AS x;
```

```
+-----+
| max |
+-----+
| 55  |
+-----+

SELECT x, MAX(x) OVER (PARTITION BY MOD(x, 2)) AS max
FROMUNNEST([8, NULL, 37, 4, NULL, 55]) AS x;

+------+------+
| x    | max  |
+------+------+
| NULL | NULL |
| NULL | NULL |
| 8    | 8    |
| 4    | 8    |
| 37   | 55   |
| 55   | 55   |
+------+------+
```

MIN

```
MIN(expression)  [OVER (...)]
```

Description

Returns the minimum value of non-NULL expressions. Returns NULL if there are zero input rows or expression evaluates to NULL for all rows. Returns NaN if the input contains a NaN.

Supported Argument Types

Any data type except: ARRAY STRUCT

Optional Clause

OVER: Specifies a window. See Analytic Functions.

Return Data Types

Same as the data type used as the input values.

Examples

```
SELECT MIN(x) AS min
FROMUNNEST([8, 37, 4, 55]) AS x;

+.......+
| min |
+.......+
| 4  |
+-----+

SELECT x, MIN(x) OVER (PARTITION BY MOD(x, 2)) AS min
FROMUNNEST([8, NULL, 37, 4, NULL, 55]) AS x;

+.........+.........+
| x    | min  |
+.........+.........+
| NULL | NULL |
| NULL | NULL |
| 8    | 4    |
| 4    | 4    |
| 37   | 37   |
| 55   | 37   |
+.........+.........+
```

5. What are correlated subquery?

Correlated subqueries are used for row-by-row processing. Each subquery is executed once for every row of the outer query. A correlated subquery is evaluated once for each row processed by the parent statement. The parent statement can be a **SELECT**, **UPDATE**, or **DELETE** statement.

EXAMPLE of Correlated Subqueries **:** Find all the employees who earn more than the average salary in their department.

SELECT last_name, salary, department_id FROM employees outer WHERE salary (SELECT AVG(salary) FROM employee WHERE department_id =outer.department_id);

6. What are scalar subqueries?

A scalar subquery is a subquery that selects only one column or expression and returns one row. A scalar subquery is a subquery that selects only one column or expression and returns one row. A scalar subquery is a subquery that selects only one column or expression and returns one row.

7. Write queries using natural join and without natural join?

Queries with natural join :

i.select id,aval1,cval1 frrom table111 natural join table113;

ii. select table111.id,table111.aval1,table113 .cval1 from table111 inner join table113 on table111.id=table113.id;

iii. select id ,aval1, cval1 from table111 natural join table113 where table111.aval1>200;

iv. select id,aval1,cval1 from table111 natural join table113 natural join table114 where table111.aval1>200;

Queries without natural joins:

i.select* from t1 left join(t2,t3,t4) on (t2 .a=t1.a and t2.bt1.b and t4.c=t1.c);

ii. select * from t1 left join(t2 cross join t3 creoss join t4) on (t2.a=t1.a and t3.b=t1.b and t4.c=t1.c);

iii. select t1,name,t2.salary from employee t1 inner join info t2 on t1.name=t2.name;

8. What is outer join?

Outer joins. When performing an inner **join**, rows from either table that are unmatched in the other table are not returned. In an **outer join**, unmatched rows in one or both tables can be returned. There are a few types of **outer joins**

1. Left outer join (also known as left join): this join returns all the rows from left table combine with the matching rows of the right table. If you get no matching in the right table it returns NULL values.

2. Right outer join (also known as right join): this join returns all the rows from right table are combined with the matching rows of left table .If you get no column matching in the left table .it returns null value.

9. Can having work without group by?

We can use the having clause with the transact_SQL extension that allows us to omit the group by clause from a queries that includes an aggregate in its select list. These scalar aggregate functions calculate values for the tables as a single group not for groups within the table.

In this example the group by vlause is omitted which makes the aggregatev functions calculate a value for the entire table.The having clauseexcludes non-matching rows from the result group.

```
 Select pub_id,count(pub_id) from publishers having pub_id<'1000';
```

10. When an 'in' can be used?

The IN operator allows us to determine if a specified value matches any value in a set of values or returned by a subquery .

The following illustrate the system of the IN operator.

```
 Select column1,column2,… from table_name where (expr|column_1) in
('value1','value2',…);
```

Exercise No:10 **PRACTICE EXERCISE-8**

DATE:31-07-2021

AIM:

To write and execute MySQL queries.

SYSTEM REQUIREMENTS:

MariaDB version 10.2.12

OS windows 8(64 bit)

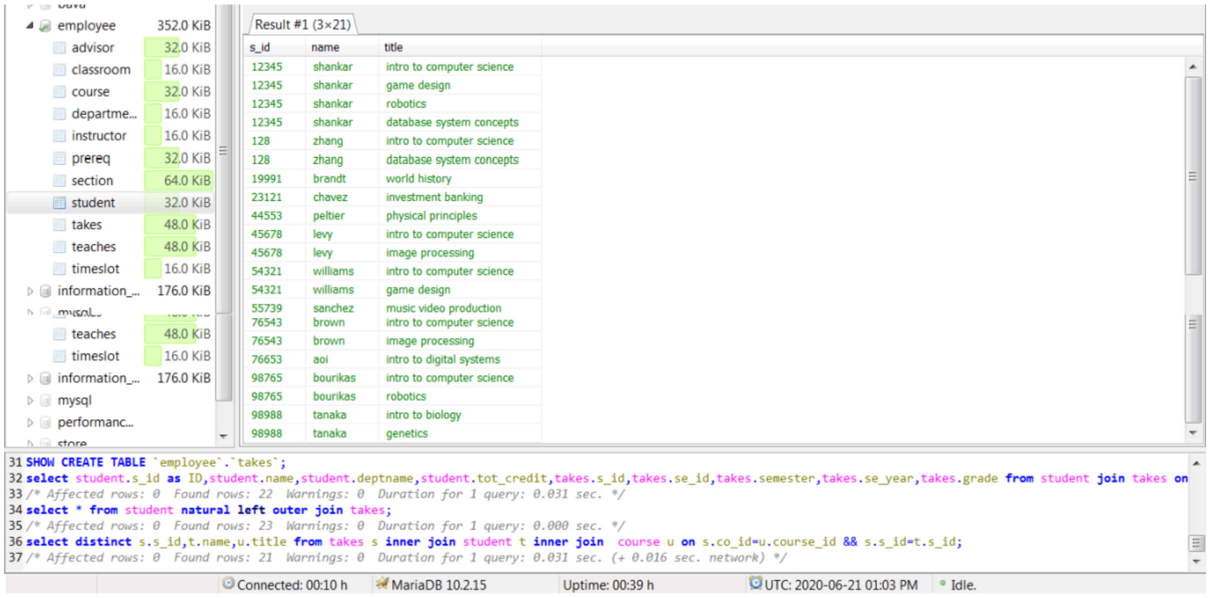
QUERIES:

1. List the courses taken by student(using on condition)

QUERY:

selectdistinct s.s_id,t.name,u.title **from** takes s **innerjoin** student t **innerjoin** course u **on** s.co_id=u.course_id && s.s_id=t.s_id;

OUTPUT:



s_id	name	title
12345	shankar	intro to computer science
12345	shankar	game design
12345	shankar	robotics
12345	shankar	database system concepts
128	zhang	intro to computer science
128	zhang	database system concepts
19991	brandt	world history
23121	chavez	investment banking
44553	peltier	physical principles
45678	levy	intro to computer science
45678	levy	image processing
54321	williams	intro to computer science
54321	williams	game design
55739	sanchez	music video production
76543	brown	intro to computer science
76543	brown	image processing
76653	aoi	intro to digital systems
98765	bourikas	intro to computer science
98765	bourikas	robotics
98988	tanaka	intro to biology
98988	tanaka	genetics

```
SHOW CREATE TABLE `employee`.`takes`;
select student.s_id as ID,student.name,student.deptname,student.tot_credit,takes.s_id,takes.se_id,takes.semester,takes.se_year,takes.grade from student join takes on
/* Affected rows: 0  Found rows: 22  Warnings: 0  Duration for 1 query: 0.031 sec. */
select * from student natural left outer join takes;
/* Affected rows: 0  Found rows: 23  Warnings: 0  Duration for 1 query: 0.000 sec. */
select distinct s.s_id,t.name,u.title from takes s inner join student t inner join  course u on s.co_id=u.course_id && s.s_id=t.s_id;
/* Affected rows: 0  Found rows: 21  Warnings: 0  Duration for 1 query: 0.031 sec. (+ 0.016 sec. network) */
```

2. What is the result of the following query:

select student.ID as ID, name, dept name, tot cred, course id, sec id, semester, year, grade

from student join takes on student.ID= takes.ID;

How it is different from result of Query 1?

How it is different from join without on condition?

What is the output of the following queries?

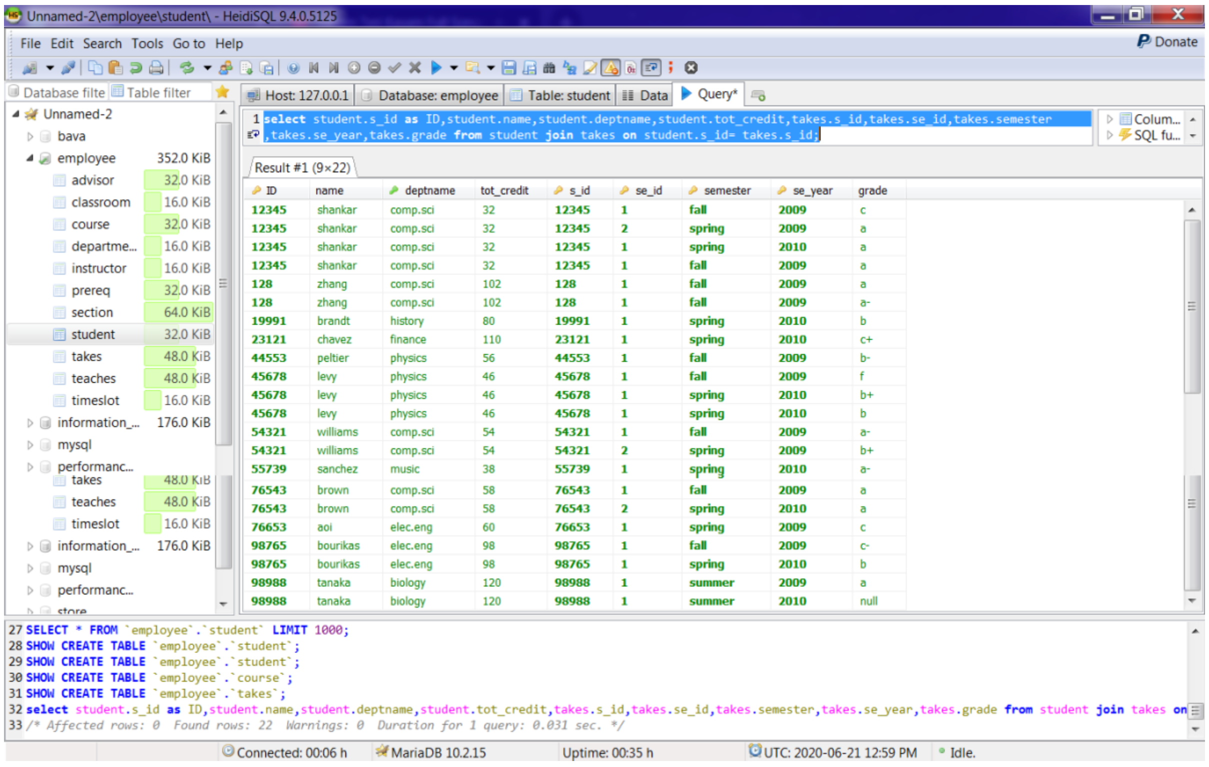
select * from student natural left outer join takes;

select *from takes natural right outer joinstudent;

QUERY:

select student.s_id **as** ID,student.name,student.deptname,student.tot_credit,takes.s_id,takes.se_id,takes.semester,takes.se_year,takes.grade **from** student **join** takes **on** student.s_id= takes.s_id;

OUTPUT:



```
select student.s_id as ID,student.name,student.deptname,student.tot_credit,takes.s_id,takes.se_id,takes.semester
,takes.se_year,takes.grade from student join takes on student.s_id= takes.s_id;
```

Result #1 (9×22)

ID	name	deptname	tot_credit	s_id	se_id	semester	se_year	grade
12345	shankar	comp.sci	32	12345	1	fall	2009	c
12345	shankar	comp.sci	32	12345	2	spring	2009	a
12345	shankar	comp.sci	32	12345	1	spring	2010	a
12345	shankar	comp.sci	32	12345	1	fall	2009	a
128	zhang	comp.sci	102	128	1	fall	2009	a
128	zhang	comp.sci	102	128	1	fall	2009	a-
19991	brandt	history	80	19991	1	spring	2010	b
23121	chavez	finance	110	23121	1	spring	2010	c+
44553	peltier	physics	56	44553	1	fall	2009	b-
45678	levy	physics	46	45678	1	fall	2009	f
45678	levy	physics	46	45678	1	spring	2010	b+
45678	levy	physics	46	45678	1	spring	2010	b
54321	williams	comp.sci	54	54321	1	fall	2009	a-
54321	williams	comp.sci	54	54321	2	spring	2009	b+
55739	sanchez	music	38	55739	1	spring	2010	a-
76543	brown	comp.sci	58	76543	1	fall	2009	a
76543	brown	comp.sci	58	76543	2	spring	2010	a
76653	aoi	elec.eng	60	76653	1	spring	2009	c
98765	bourikas	elec.eng	98	98765	1	fall	2009	c-
98765	bourikas	elec.eng	98	98765	1	spring	2010	b
98988	tanaka	biology	120	98988	1	summer	2009	a
98988	tanaka	biology	120	98988	1	summer	2010	null

```
SELECT * FROM `employee`.`student` LIMIT 1000;
SHOW CREATE TABLE `employee`.`student`;
SHOW CREATE TABLE `employee`.`student`;
SHOW CREATE TABLE `employee`.`course`;
SHOW CREATE TABLE `employee`.`takes`;
select student.s_id as ID,student.name,student.deptname,student.tot_credit,takes.s_id,takes.se_id,takes.semester,takes.se_year,takes.grade from student join takes on
/* Affected rows: 0  Found rows: 22  Warnings: 0  Duration for 1 query: 0.031 sec. */
```

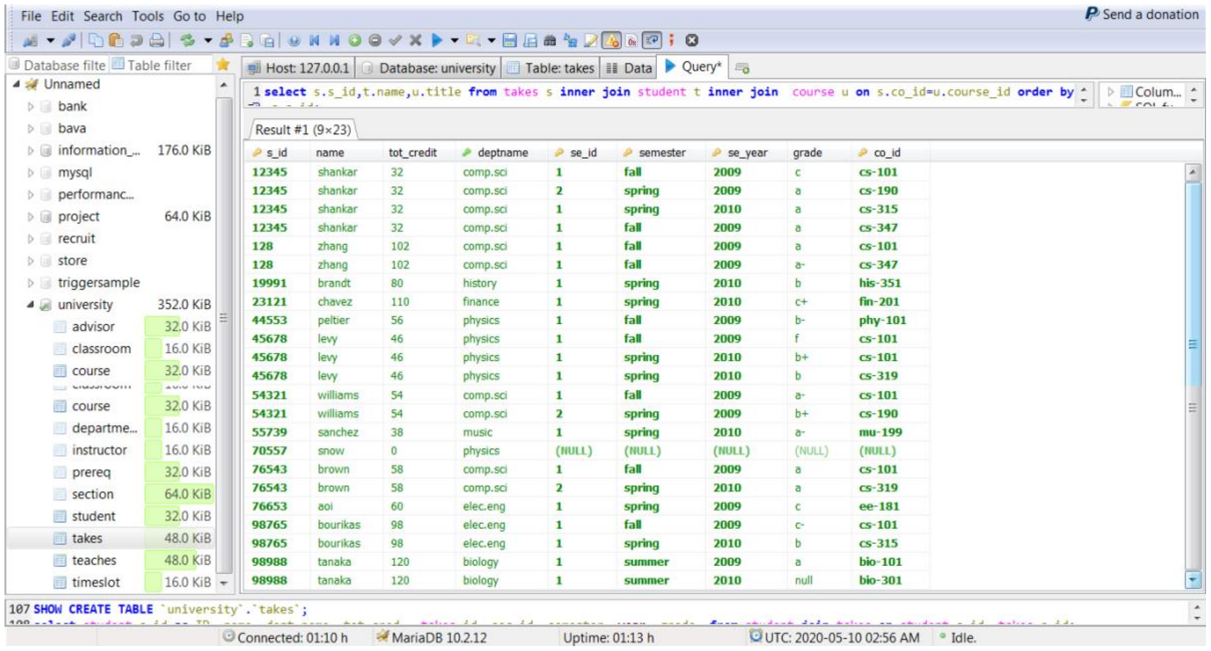
Connected: 00:06 h   MariaDB 10.2.15   Uptime: 00:35 h   UTC: 2020-06-21 12:59 PM   Idle.

How it is different from result of Query 1?

Here the student who have failed and repeated the course also appears.And the in the 1st query only once s_id attribute appears whereas here it appears twice.

select * from student natural left outer join takes;

OUTPUT:



```
1 select s.s_id,t.name,u.title from takes s inner join student t inner join  course u on s.co_id=u.course_id order by
```

Result #1 (9×23)

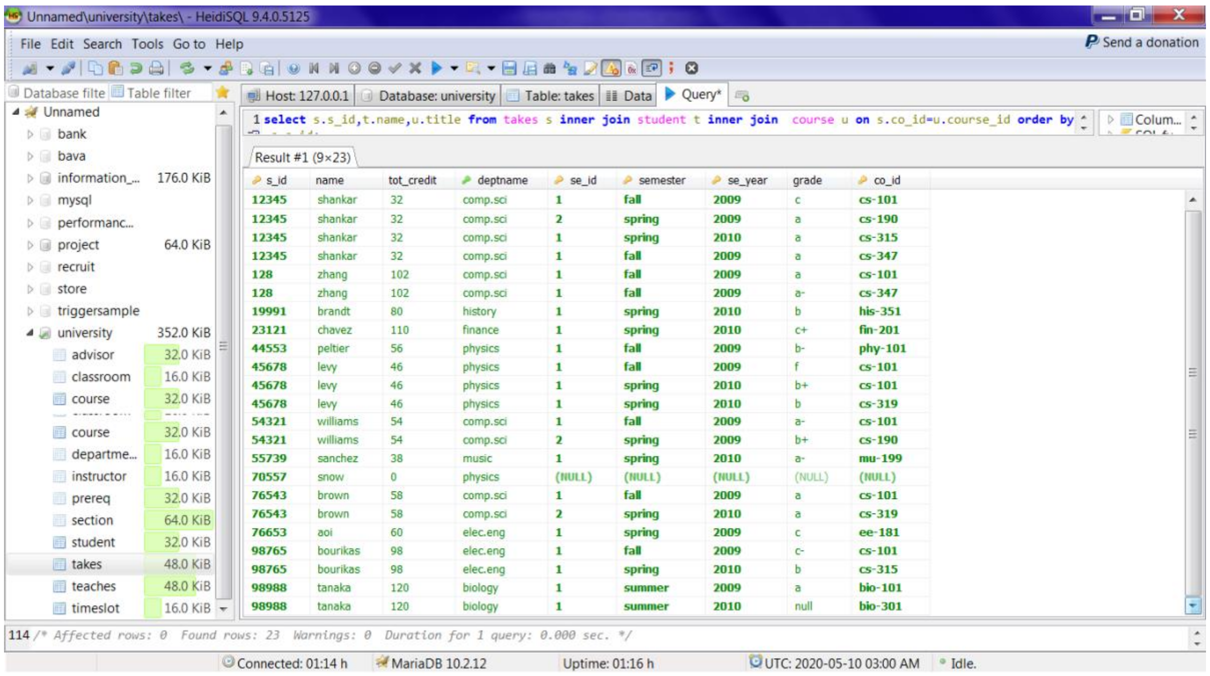
s_id	name	tot_credit	deptname	se_id	semester	se_year	grade	co_id
12345	shankar	32	comp.sci	1	fall	2009	c	cs-101
12345	shankar	32	comp.sci	2	spring	2009	a	cs-190
12345	shankar	32	comp.sci	1	spring	2010	a	cs-315
12345	shankar	32	comp.sci	1	fall	2009	a	cs-347
128	zhang	102	comp.sci	1	fall	2009	a	cs-101
128	zhang	102	comp.sci	1	fall	2009	a-	cs-347
19991	brandt	80	history	1	spring	2010	b	his-351
23121	chavez	110	finance	1	spring	2010	c+	fin-201
44553	peltier	56	physics	1	fall	2009	b-	phy-101
45678	levy	46	physics	1	fall	2009	f	cs-101
45678	levy	46	physics	1	spring	2010	b+	cs-101
45678	levy	46	physics	1	spring	2010	b	cs-319
54321	williams	54	comp.sci	1	fall	2009	a-	cs-101
54321	williams	54	comp.sci	2	spring	2009	b+	cs-190
55739	sanchez	38	music	1	spring	2010	a-	mu-199
70557	snow	0	physics	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
76543	brown	58	comp.sci	1	fall	2009	a	cs-101
76543	brown	58	comp.sci	2	spring	2010	a	cs-319
76653	aoi	60	elec.eng	1	spring	2009	c	ee-181
98765	bourikas	98	elec.eng	1	fall	2009	c-	cs-101
98765	bourikas	98	elec.eng	1	spring	2010	b	cs-315
98988	tanaka	120	biology	1	summer	2009	a	bio-101
98988	tanaka	120	biology	1	summer	2010	null	bio-301

```
107 SHOW CREATE TABLE `university`.`takes`;
```

Connected: 01:10 h   MariaDB 10.2.12   Uptime: 01:13 h   UTC: 2020-05-10 02:56 AM   Idle.

select *from takes natural right outer join student;

OUTPUT:



```
1 select s.s_id,t.name,u.title from takes s inner join student t inner join  course u on s.co_id=u.course_id order by
```

Result #1 (9×23)

s_id	name	tot_credit	deptname	se_id	semester	se_year	grade	co_id
12345	shankar	32	comp.sci	1	fall	2009	c	cs-101
12345	shankar	32	comp.sci	2	spring	2009	a	cs-190
12345	shankar	32	comp.sci	1	spring	2010	a	cs-315
12345	shankar	32	comp.sci	1	fall	2009	a	cs-347
128	zhang	102	comp.sci	1	fall	2009	a	cs-101
128	zhang	102	comp.sci	1	fall	2009	a-	cs-347
19991	brandt	80	history	1	spring	2010	b	his-351
23121	chavez	110	finance	1	spring	2010	c+	fin-201
44553	peltier	56	physics	1	fall	2009	b-	phy-101
45678	levy	46	physics	1	fall	2009	f	cs-101
45678	levy	46	physics	1	spring	2010	b+	cs-101
45678	levy	46	physics	1	spring	2010	b	cs-319
54321	williams	54	comp.sci	1	fall	2009	a-	cs-101
54321	williams	54	comp.sci	2	spring	2009	b+	cs-190
55739	sanchez	38	music	1	spring	2010	a-	mu-199
70557	snow	0	physics	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
76543	brown	58	comp.sci	1	fall	2009	a	cs-101
76543	brown	58	comp.sci	2	spring	2010	a	cs-319
76653	aoi	60	elec.eng	1	spring	2009	c	ee-181
98765	bourikas	98	elec.eng	1	fall	2009	c-	cs-101
98765	bourikas	98	elec.eng	1	spring	2010	b	cs-315
98988	tanaka	120	biology	1	summer	2009	a	bio-101
98988	tanaka	120	biology	1	summer	2010	null	bio-301

```
114 /* Affected rows: 0  Found rows: 23  Warnings: 0  Duration for 1 query: 0.000 sec. */
```

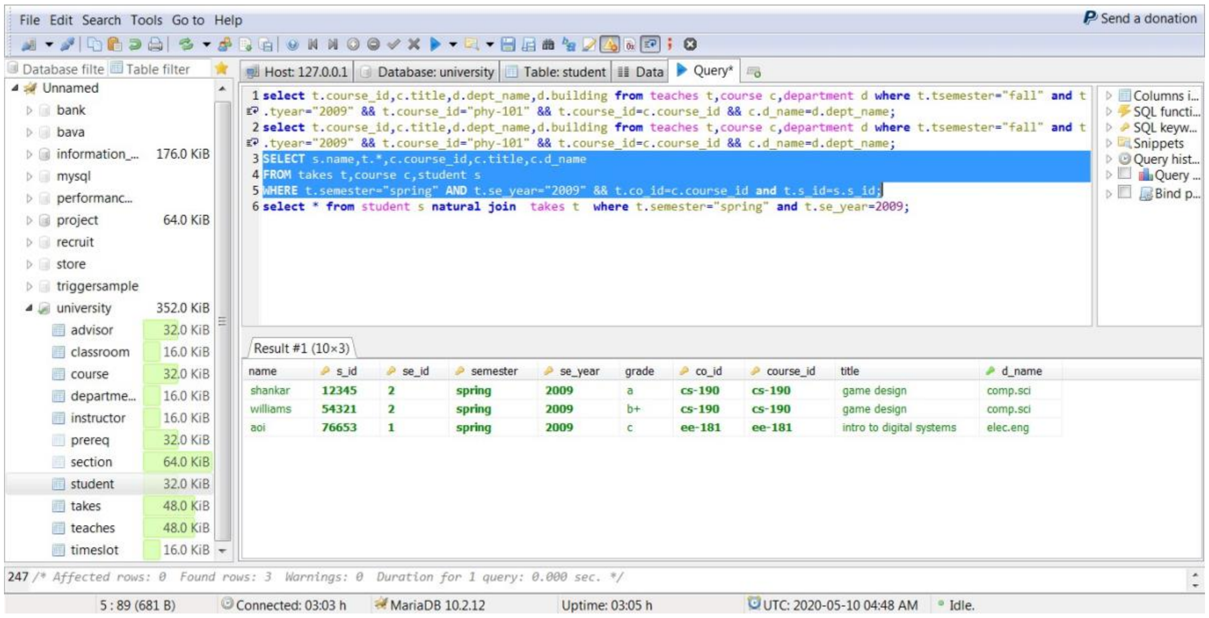
Connected: 01:14 h MariaDB 10.2.12 Uptime: 01:16 h UTC: 2020-05-10 03:00 AM Idle.

3. Write a query to Display a list of all students in the Comp. Sci. department, along with the course sections, if any, that they have taken in Spring 2009; all course sections fromSpring 2009 must be displayed, even if no student from the Comp. Sci. department has taken the course section.

QUERY:

SELECT s.name,t.*,c.course_id,c.title,c.d_name **FROM** takes t,course c,student s **WHERE** t.semester="spring" **AND** t.se_year="2009" && t.co_id=c.course_id **and** t.s_id=s.s_id;

OUTPUT:



```
select t.course_id,c.title,d.dept_name,d.building from teaches t,course c,department d where t.tsemester="fall" and t
.tyear="2009" && t.course_id="phy-101" && t.course_id=c.course_id && c.d_name=d.dept_name;
select t.course_id,c.title,d.dept_name,d.building from teaches t,course c,department d where t.tsemester="fall" and t
.tyear="2009" && t.course_id="phy-101" && t.course_id=c.course_id && c.d_name=d.dept_name;
SELECT s.name,t.*,c.course_id,c.title,c.d_name
FROM takes t,course c,student s
WHERE t.semester="spring" AND t.se_year="2009" && t.co_id=c.course_id and t.s_id=s.s_id;
select * from student s natural join  takes t  where t.semester="spring" and t.se_year=2009;
```

Result #1 (10×3)

name	s_id	se_id	semester	se_year	grade	co_id	course_id	title	d_name
shankar	12345	2	spring	2009	a	cs-190	cs-190	game design	comp.sci
williams	54321	2	spring	2009	b+	cs-190	cs-190	game design	comp.sci
aoi	76653	1	spring	2009	c	ee-181	ee-181	intro to digital systems	elec.eng

```
247 /* Affected rows: 0  Found rows: 3  Warnings: 0  Duration for 1 query: 0.000 sec. */
```

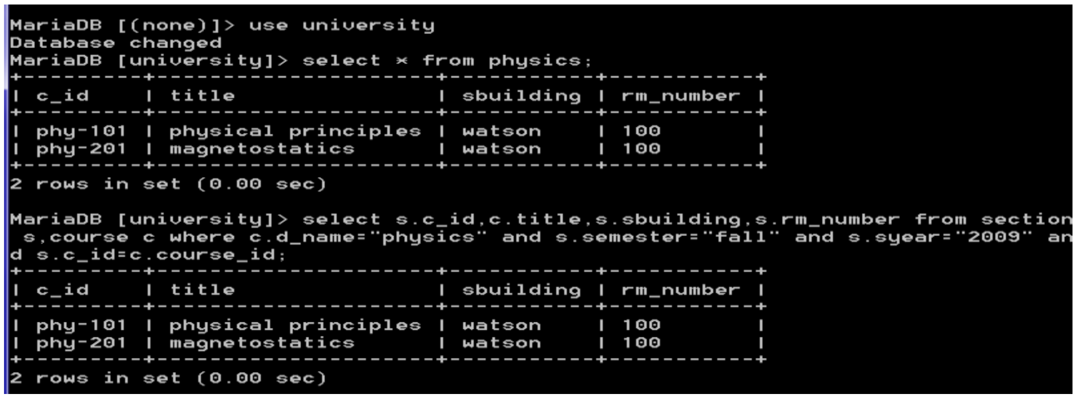
5 : 89 (681 B)   Connected: 03:03 h   MariaDB 10.2.12   Uptime: 03:05 h   UTC: 2020-05-10 04:48 AM   Idle.

4. Write a query to list of all course sections offered by the Physicsdepartment in the Fall 2009 semester,with the building and room number of each section.

QUERY:

select s.c_id,c.title,s.sbuilding,s.rm_number **from** section s,course c **where** c.d_name="physics" **and** s.semester="fall" **and** s.syear="2009" **and** s.c_id=c.course_id;

OUTPUT:



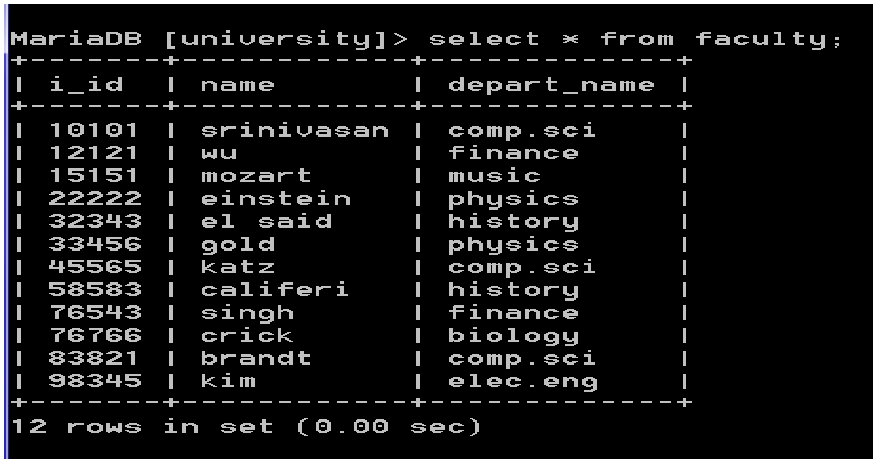
```
MariaDB [(none)]> use university
Database changed
MariaDB [university]> select * from physics;
+---------+--------------------+-----------+-----------+
| c_id    | title              | sbuilding | rm_number |
+---------+--------------------+-----------+-----------+
| phy-101 | physical principles | watson    | 100       |
| phy-201 | magnetostatics      | watson    | 100       |
+---------+--------------------+-----------+-----------+
2 rows in set (0.00 sec)

MariaDB [university]> select s.c_id,c.title,s.sbuilding,s.rm_number from section
 s,course c where c.d_name="physics" and s.semester="fall" and s.syear="2009" an
d s.c_id=c.course_id;
+---------+--------------------+-----------+-----------+
| c_id    | title              | sbuilding | rm_number |
+---------+--------------------+-----------+-----------+
| phy-101 | physical principles | watson    | 100       |
| phy-201 | magnetostatics      | watson    | 100       |
+---------+--------------------+-----------+-----------+
2 rows in set (0.00 sec)
```

5. Create a view named faculty to list faculty id,name and department

QUERY:

i. **createview** faculty **asselect** i_id,name,depart_name **from** instructor;
ii. **select** * **from** faculty;

OUTPUT:



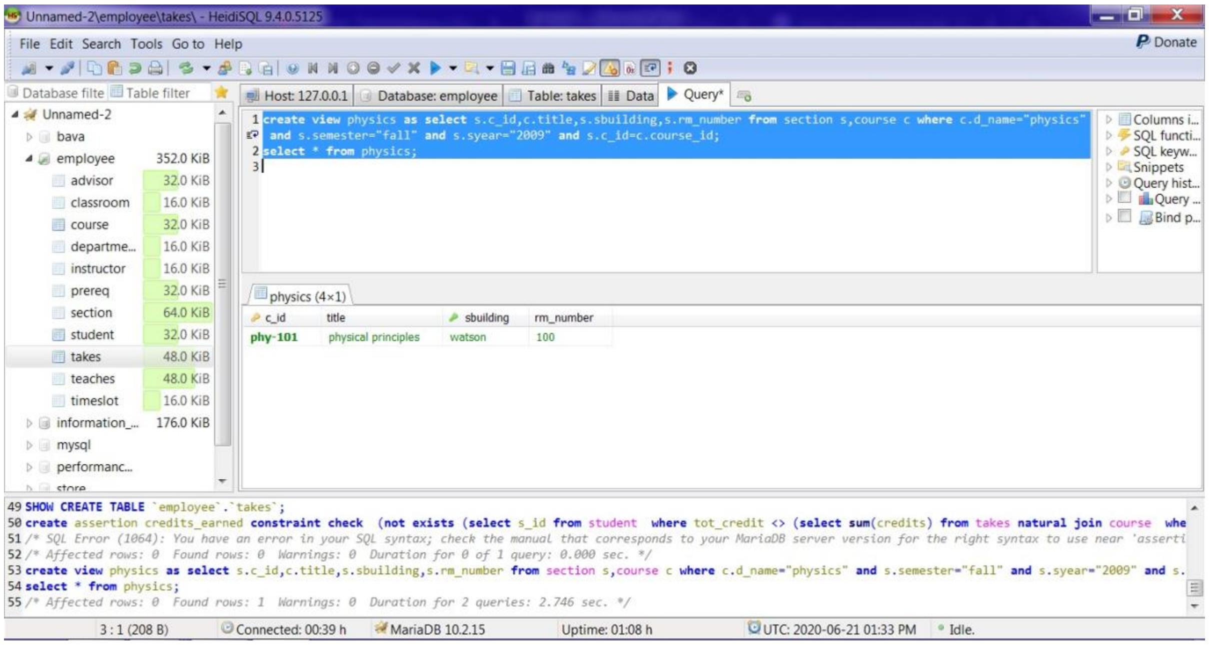
```
MariaDB [university]> select * from faculty;
+-------+------------+-------------+
| i_id  | name       | depart_name |
+-------+------------+-------------+
| 10101 | srinivasan | comp.sci    |
| 12121 | wu         | finance     |
| 15151 | mozart     | music       |
| 22222 | einstein   | physics     |
| 32343 | el said    | history     |
| 33456 | gold       | physics     |
| 45565 | katz       | comp.sci    |
| 58583 | califeri   | history     |
| 76543 | singh      | finance     |
| 76766 | crick      | biology     |
| 83821 | brandt     | comp.sci    |
| 98345 | kim        | elec.eng    |
+-------+------------+-------------+
12 rows in set (0.00 sec)
```

6. Create a view physics fall 2009 to list of all course sections offered by the Physicsdepartment in the Fall 2009 semester,with the building and room number of each section.

QUERY:

i. **createview** physics **asselect** s.c_id,c.title,s.sbuilding,s.rm_number **from** section s,course c **where** c.d_name="physics" **and** s.semester="fall" **and** s.syear="2009" **and** s.c_id=c.course_id;
ii. **select** * **from** physics;

OUTPUT:



```
create view physics as select s.c_id,c.title,s.sbuilding,s.rm_number from section s,course c where c.d_name="physics"
 and s.semester="fall" and s.syear="2009" and s.c_id=c.course_id;
select * from physics;
```

physics (4×1)

c_id	title	sbuilding	rm_number
phy-101	physical principles	watson	100

```
49 SHOW CREATE TABLE `employee`.`takes`;
50 create assertion credits_earned constraint check  (not exists (select s_id from student  where tot_credit <> (select sum(credits) from takes natural join course  whe
51 /* SQL Error (1064): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'asserti
52 /* Affected rows: 0  Found rows: 0  Warnings: 0  Duration for 0 of 1 query: 0.000 sec. */
53 create view physics as select s.c_id,c.title,s.sbuilding,s.rm_number from section s,course c where c.d_name="physics" and s.semester="fall" and s.syear="2009" and s.
54 select * from physics;
55 /* Affected rows: 0  Found rows: 1  Warnings: 0  Duration for 2 queries: 2.746 sec. */
```

Insert or delete some tuples in course and section tables .

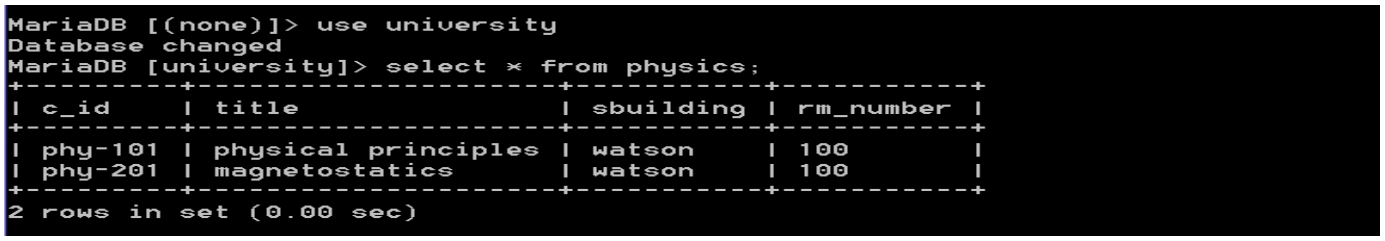
insert into course values(‘phy-201’,’’magnetostatics’,’4’,’physics’),(‘phy-203’,’electrostatics’,’4’,’physics’);

insertinto section **values** (‘1’,’fall’,’2009’,’phy-201’,’a’,’watson’,’100’),('1','fall','2010','phy-203','a','watson','100');

Run select * from physics fall 2009 and note down the results:

select * from physics;

OUTPUT:

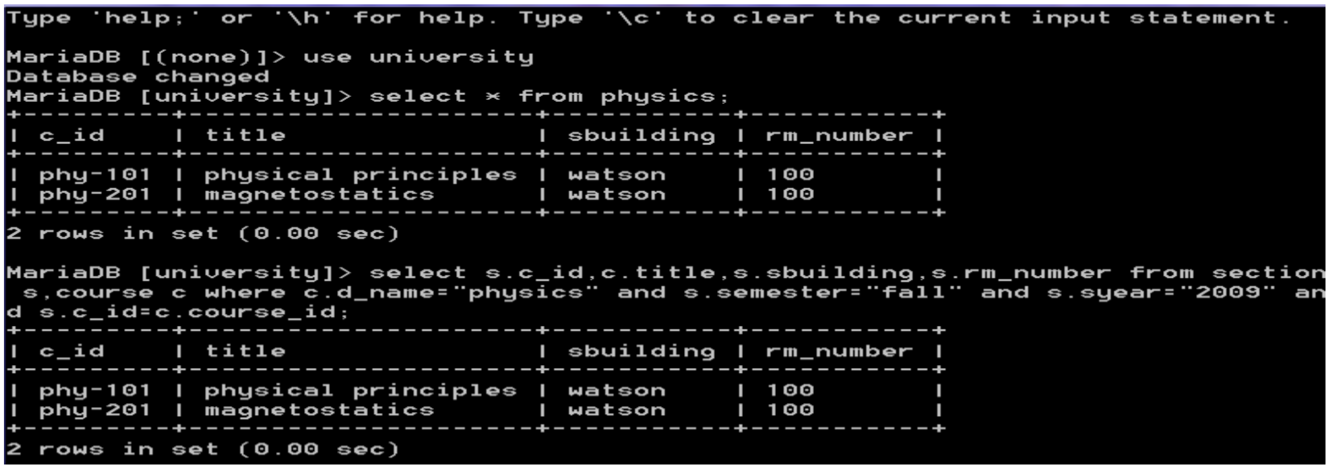


```
MariaDB [(none)]> use university
Database changed
MariaDB [university]> select * from physics;
+---------+---------------------+-----------+-----------+
| c_id    | title               | sbuilding | rm_number |
+---------+---------------------+-----------+-----------+
| phy-101 | physical principles | watson    | 100       |
| phy-201 | magnetostatics      | watson    | 100       |
+---------+---------------------+-----------+-----------+
2 rows in set (0.00 sec)
```

Execute Q4 and note down the results

select s.c_id,c.title,s.sbuilding,s.rm_number **from** section s,course c **where** c.d_name="physics" **and** s.semester="fall" **and** s.syear="2009" **and** s.c_id=c.course_id;

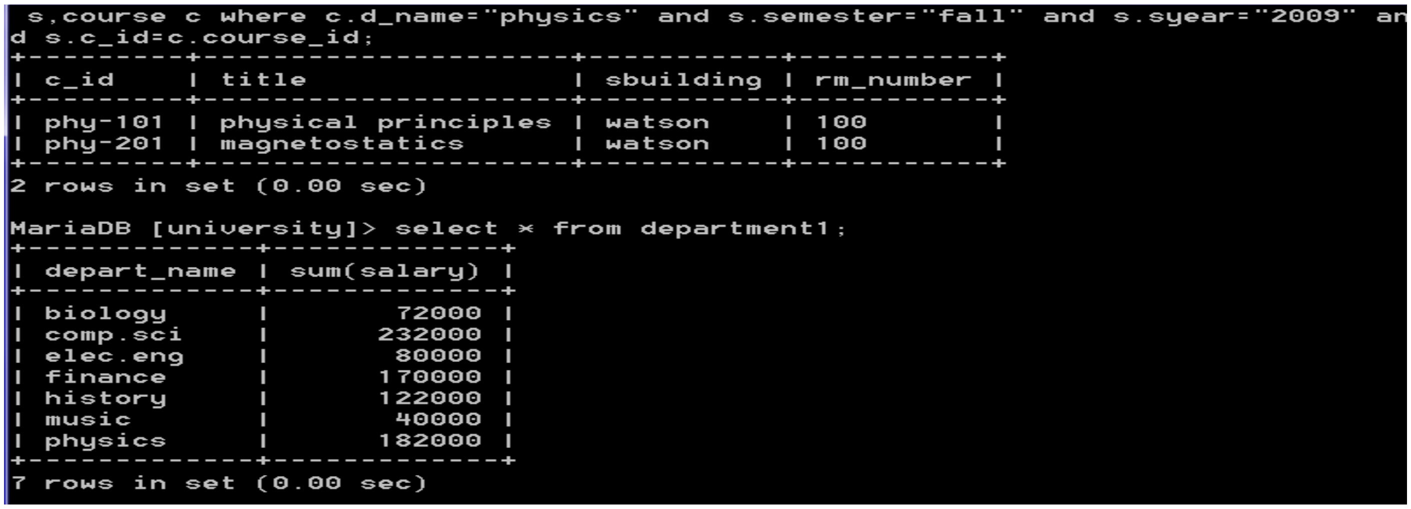
OUTPUT:



```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use university
Database changed
MariaDB [university]> select * from physics;
+---------+---------------------+-----------+-----------+
| c_id    | title               | sbuilding | rm_number |
+---------+---------------------+-----------+-----------+
| phy-101 | physical principles | watson    | 100       |
| phy-201 | magnetostatics      | watson    | 100       |
+---------+---------------------+-----------+-----------+
2 rows in set (0.00 sec)

MariaDB [university]> select s.c_id,c.title,s.sbuilding,s.rm_number from section
 s,course c where c.d_name="physics" and s.semester="fall" and s.syear="2009" an
d s.c_id=c.course_id;
+---------+---------------------+-----------+-----------+
| c_id    | title               | sbuilding | rm_number |
+---------+---------------------+-----------+-----------+
| phy-101 | physical principles | watson    | 100       |
| phy-201 | magnetostatics      | watson    | 100       |
+---------+---------------------+-----------+-----------+
2 rows in set (0.00 sec)
```

7. What is the out put of the following queries
a) create view departments total salary(dept name, total salary) as select dept name, sum (salary) from instructor group by dept name;

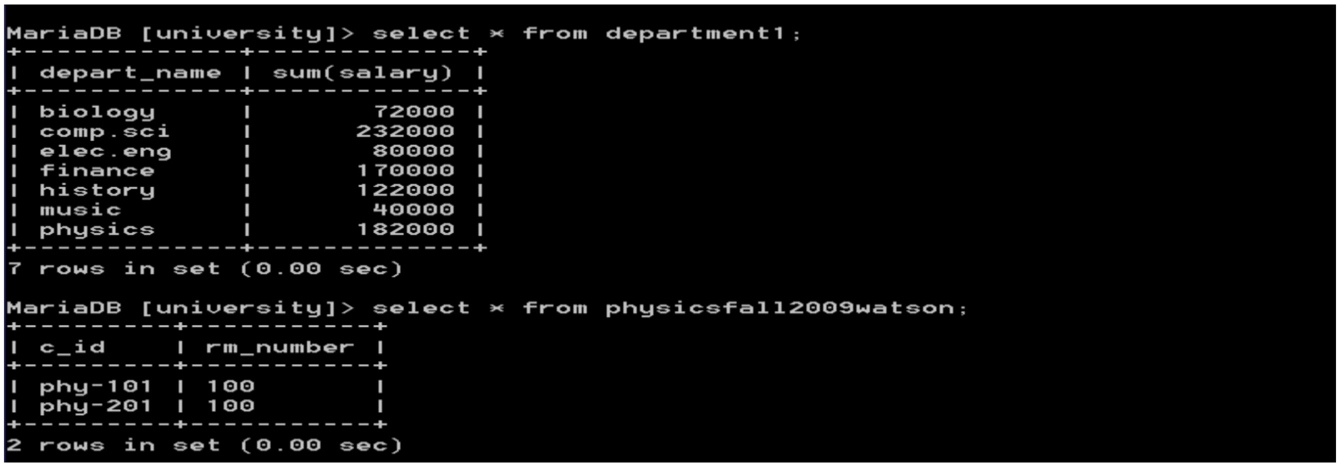
createview department1 **as**
select depart_name, **sum**(salary) **from** instructor **groupby** depart_name;
select * **from** department1;
OUTPUT:


```
 s,course c where c.d_name="physics" and s.semester="fall" and s.syear="2009" an
d s.c_id=c.course_id;
+---------+---------------------+-----------+-----------+
| c_id    | title               | sbuilding | rm_number |
+---------+---------------------+-----------+-----------+
| phy-101 | physical principles | watson    | 100       |
| phy-201 | magnetostatics      | watson    | 100       |
+---------+---------------------+-----------+-----------+
2 rows in set (0.00 sec)

MariaDB [university]> select * from department1;
+-------------+-------------+
| depart_name | sum(salary) |
+-------------+-------------+
| biology     |       72000 |
| comp.sci    |      232000 |
| elec.eng    |       80000 |
| finance     |      170000 |
| history     |      122000 |
| music       |       40000 |
| physics     |      182000 |
+-------------+-------------+
7 rows in set (0.00 sec)
```

create view physics fall 2009 watson as select course id, room number from physics fall 2009 where building= ’Watson’;

QUERY:

createview physicsfall2009watson **asselect** c_id, rm_number
from physics **where** sbuilding= "Watson";
select * **from** physicsfall2009watson;
OUTPUT:

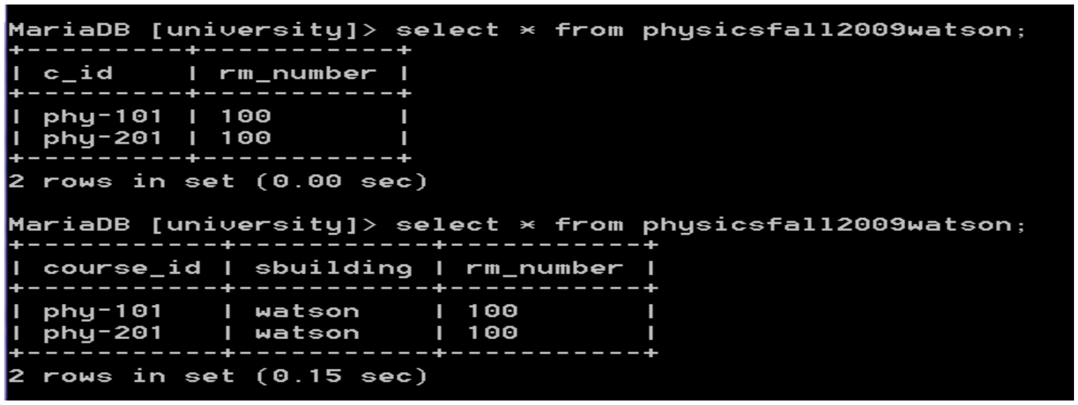

```
MariaDB [university]> select * from department1;
+-------------+-------------+
| depart_name | sum(salary) |
+-------------+-------------+
| biology     |       72000 |
| comp.sci    |      232000 |
| elec.eng    |       80000 |
| finance     |      170000 |
| history     |      122000 |
| music       |       40000 |
| physics     |      182000 |
+-------------+-------------+
7 rows in set (0.00 sec)

MariaDB [university]> select * from physicsfall2009watson;
+---------+-----------+
| c_id    | rm_number |
+---------+-----------+
| phy-101 | 100       |
| phy-201 | 100       |
+---------+-----------+
2 rows in set (0.00 sec)
```

QUERY:

alterview physicsfall2009watson **as**

select course.course_id,sbuilding,rm_number **from** course , section
where course.course_id = section.c_id **and** course.d_name = 'Physics'
and section.semester = 'Fall' **and** section.syear = '2009' && sbuilding= 'watson';

select * **from** physicsfall2009watson;
OUTPUT:



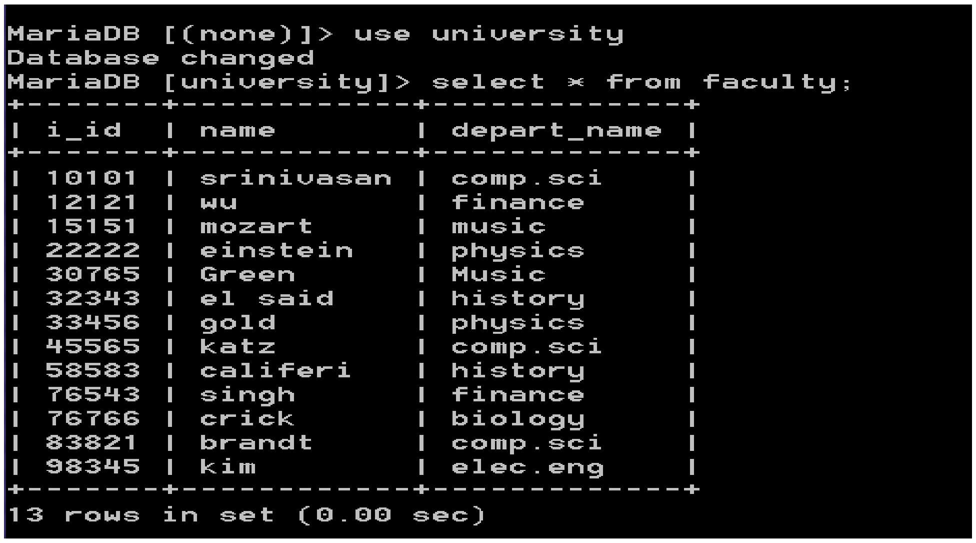
```
MariaDB [university]> select * from physicsfall2009watson;
+---------+-----------+
| c_id    | rm_number |
+---------+-----------+
| phy-101 | 100       |
| phy-201 | 100       |
+---------+-----------+
2 rows in set (0.00 sec)

MariaDB [university]> select * from physicsfall2009watson;
+-----------+-----------+-----------+
| course_id | sbuilding | rm_number |
+-----------+-----------+-----------+
| phy-101   | watson    | 100       |
| phy-201   | watson    | 100       |
+-----------+-----------+-----------+
2 rows in set (0.15 sec)
```

8. What happens if the following query is executed?

QUERY:

insertinto faculty **values** ('30765', 'Green', 'Music');

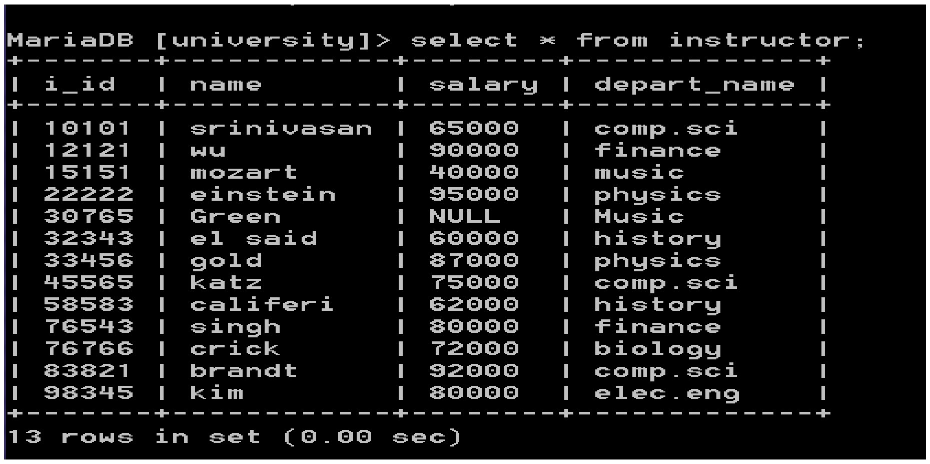
Affected **rows**: 0 **Foundrows**: 1 **Warnings**: 0 Duration **for** 1 **query**: 0.000 sec. */
OUTPUT:



```
MariaDB [(none)]> use university
Database changed
MariaDB [university]> select * from faculty;
+-------+------------+------------+
| i_id  | name       | depart_name |
+-------+------------+------------+
| 10101 | srinivasan | comp.sci    |
| 12121 | wu         | finance     |
| 15151 | mozart     | music       |
| 22222 | einstein   | physics     |
| 30765 | Green      | Music       |
| 32343 | el said    | history     |
| 33456 | gold       | physics     |
| 45565 | katz       | comp.sci    |
| 58583 | califeri   | history     |
| 76543 | singh      | finance     |
| 76766 | crick      | biology     |
| 83821 | brandt     | comp.sci    |
| 98345 | kim        | elec.eng    |
+-------+------------+------------+
13 rows in set (0.00 sec)
```

Select * from instructor

OUTPUT:



```
MariaDB [university]> select * from instructor;
+-------+------------+--------+-------------+
| i_id  | name       | salary | depart_name |
+-------+------------+--------+-------------+
| 10101 | srinivasan | 65000  | comp.sci    |
| 12121 | wu         | 90000  | finance     |
| 15151 | mozart     | 40000  | music       |
| 22222 | einstein   | 95000  | physics     |
| 30765 | Green      | NULL   | Music       |
| 32343 | el said    | 60000  | history     |
| 33456 | gold       | 87000  | physics     |
| 45565 | katz       | 75000  | comp.sci    |
| 58583 | califeri   | 62000  | history     |
| 76543 | singh      | 80000  | finance     |
| 76766 | crick      | 72000  | biology     |
| 83821 | brandt     | 92000  | comp.sci    |
| 98345 | kim        | 80000  | elec.eng    |
+-------+------------+--------+-------------+
13 rows in set (0.00 sec)
```

9. Create the following view instructor_info
create view instructor_info as select ID, name, building from instructor, department where instructor.dept name= department.dept name;

After successful creation of the instructor_info view insert the following tuple
insert into instructor _info values (’69987’, ’White’, ’Taylor’);

How this insertion is relected in the base tables instructor and department tables? Write your inference.

QUERY:

createview instructor_info **asselect** i_id, name, d.building **from** instructor, department d
where instructor.depart_name= d.dept_name;

Affected **rows**: 0 **Foundrows**: 0 **Warnings**: 0 Duration **for** 1 **query**: 0.172 sec. */

insert into instructor _info values (’69987’, ’White’, ’Taylor’);

ERROR 1394 (HY000): Can not insert into join view 'university.instructor_info' without fields list

insert into instructor_info (ID,name,building) values ('69987', 'White', 'Taylor');
ERROR 1393 (HY000): Can not modify more than one base table through a join view 'university.instructor_info

RESULT:

Thus the queries for university exercise was executed and verified successfully.

QUIZ QUESTIONS:

1. What are integrity constraints?

Integrity constraints are a set of rules......Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected. Thus, integrity constraint is used to guard against accidental damage to the database. Integrity Constraints are used to apply business rules for the database tables. The constraints available in SQL are Foreign Key, Not Null, Unique, Check.

2. What is referential integrity?

This constraint identifies any column referencing the PRIMARY KEY in another table. It establishes a relationship between two columns in the same table or between different tables. One or more columns can be defined as Foreign key.

3. What is materialized view?

A materialized view is a database object that contains the results of a query. The FROM clause of the query can name tables, views, and other materialized views. Collectively these objects are called master tables (a replication term) or detail tables (a data warehousing term). Materialized View is a physical copy, picture or snapshot of the base table.

4. What is the difference between relation(table) and view relation(table)?

In a relational database, the table is a relation because it stores the relation between data in its column-row format. The columns are the table's attributes, and the rows represent the data records. A single row is known as a tuple.

A view is a virtual table. A view consists of rows and columns just like a table. The difference between a view and a table is that views are definitions built on top of other tables (or views), and do not hold data themselves. If data is changing in the underlying table, the same change is reflected in the view.

5. **What is an assertion?**

An assertion is a piece of SQL which makes sure a condition is satisfied or it stops action being taken on a database object. It could mean locking out the whole table or even the whole database. Assertions are not linked to specific tables in the database and not linked to specific events.

An assertion is a statement in SQL that ensures a certain condition will always exist in the database. Assertions are like column and table constraints, except that they are specified separately from table definitions. However, assertions are checked only when UPDATE or INSERT actions are performed against the table.

6. **What is the difference between Cartesian product,natural join,left outerjoin,right outerjoin?**

CARTESIAN PRODUCT :

The Cartesian product, also referred to as a cross-join, returns all the rows in all the tables listed in the query. Each row in the first table is paired with all the rows in the second table. This happens when there is no relationship defined between the two tables.

NATURAL JOIN :

A NATURAL JOIN is a JOIN operation that creates an implicit join clause for you based on the common columns in the two tables being joined. Common columns are columns that have the same name in both tables. A Natural Join is where 2 tables are joined on the basis of all common columns. A NATURAL JOIN can be an INNER join, a LEFT OUTER join, or a RIGHT OUTER join.

LEFT OUTER JOIN :

SQL LEFT JOIN is used to combine the two tables together. LEFT JOIN selects all records from left table and also selects all matching records from the right table. And, LEFT JOIN selects all records from left table, even though there are no matching records in the right table. In this, all selected right column values will be returned as NULL. LEFT JOIN is also called as a LEFT OUTER JOIN.

RIGHT OUTER JOIN :

The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1). The result is NULL from the left side, when there is no match.

7. What is the necessary condition for joining two tables?

A JOIN clause is used to combine rows from two or more tables, based on a related column between them. Joins allow you to link data from two or more tables together into a single query result—from one single SELECT statement. A “Join” can be recognized in a SQL SELECT statement if it has more than one table after the FROM keyword.

- **(INNER) JOIN**: Returns records that have matching values in both tables
- **LEFT (OUTER) JOIN:** Returns all records from the left table, and the matched records from the right table
- **RIGHT (OUTER) JOIN:** Returns all records from the right table, and the matched records from the left table
- **FULL (OUTER) JOIN:** Returns all records when there is a match in either left or right table

MINI PROJECT

DATE:31-08-2021

E-COMMERCE DATABASE MANAGEMENT SYSTEM

OBJECTIVES

The Prime Objective of our database project is to design a robust E-commerce database by performing operations such as

- Viewing orders
- Placing orders
- Updating database
- Reviewing products
- Maintaining data consistency across tables

Using features such as :

- Triggers
- Stored procedures
- Functions
- Transactions

PROPOSED BY

19CS1076	19CS1113	19CS1100
N.L.S HARSHA	UTKARSH	SAURABH KISHOR

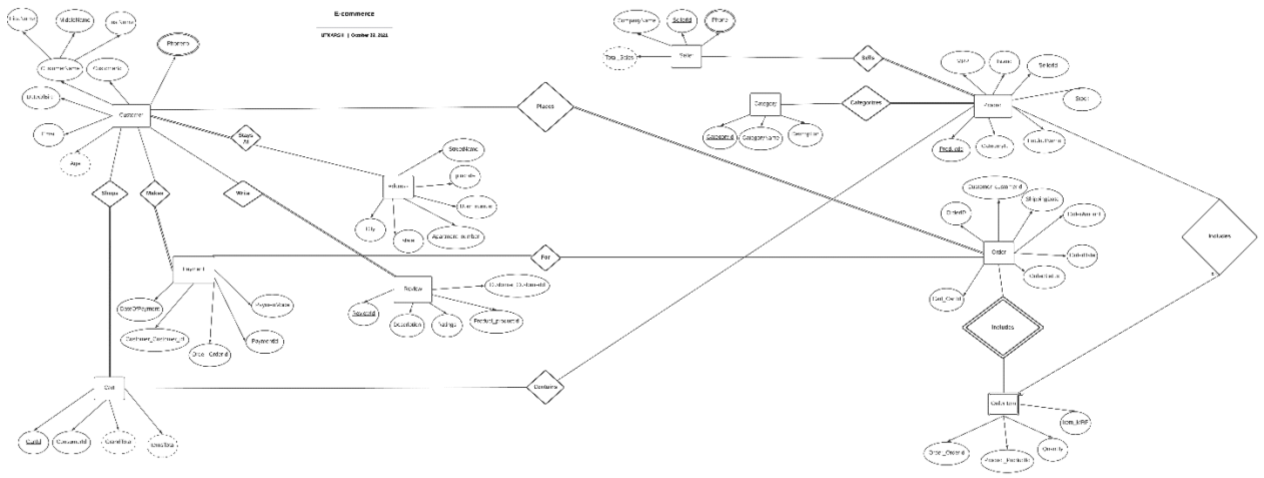
REQUIREMENTS

- This platform provides an interface to sellers and customers to buy and sell products.Each seller has a unique id using which it can access its portfolio and add products,interact with customers.
- In turn customers using their unique id can buy products.It also allows **customers to be sellers and vice versa.**
- Each product can range from different categories from which customers will be able to **sort the products** according to their category.
- Customers can see the ratings and reviews of the product and can compare different products based on them and can choose the best one.
- For each customer a cart is maintained in which customers can add their wishlist of products and can buy whenever required.
- While buying a product customers will be given different payment options from which one can be chosen.
- Once the order is placed customers can keep track of their order and can cancel the order if it's necessary(Terms and Conditions applied).

- A seller can have different categories of products.Seller can add or delete products and can also update the stock of the products.
- Sellers can keep track of the sales ,can receive feedback from customers through the interface and can improve the product quality etc if required.

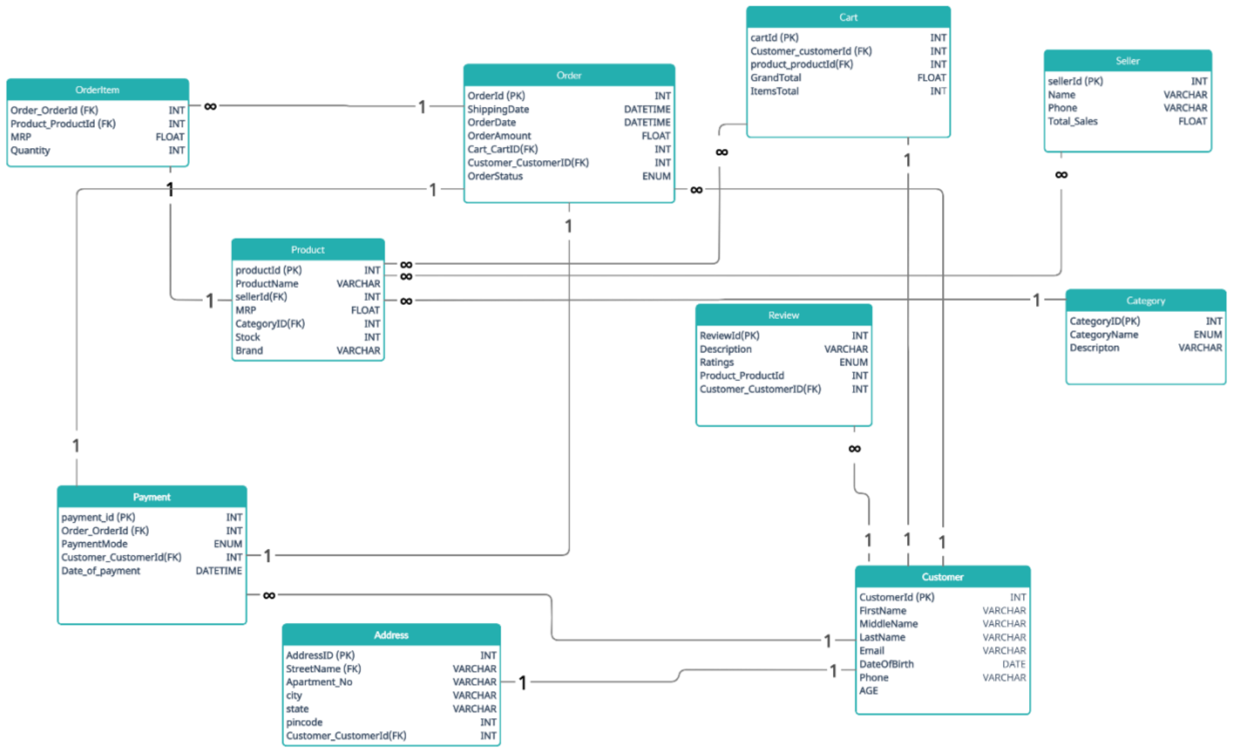
- To process an order, one should check whether those items are in stock.
- If items are in stock, they need to be reserved so that they go in hands of those who have expressed them in wishlist/order.
- Once ordered the available quantity must be reduced to reflect the correct value in the stock.
- Any items not in stock cannot be sanctioned; this requires confirmation from the seller.
- The customer needs to be informed as to which items are in stock (and can be shipped immediately) and which are cancelled.

ER-DIAGRAM:



https://lucid.app/lucidchart/invitations/accept/inv_ad5807b2-7e25-4239-998a-93024e33f98c

RELATIONAL SCHEMA:



Entities and their Attributes

ENTITIES	ATTRIBUTES	ATTRIBUTE TYPE	Entity Type
Customer	Customer_CustomerId Name Email DateOfBirth Phone Age	Simple Composite Simple Simple Multivalued Derived	Strong
Order	OrderId ShippingDate OrderDate OrderAmount Cart_CartID	Simple Simple Simple Simple Simple	Strong
OrderItem	Order_OrderId (PK) Product_ProductId(FK) MRP Quantity	Simple Simple Simple Simple	Weak
Product	productId (PK) ProductName(FK) sellerId MRP CategoryID Stock Brand	Simple Simple Simple Simple Simple Simple Simple	Strong

Review	ReviewId(PK) Description Ratings Product_ProductId Customer_CustomerID(FK)	Simple Simple Simple Simple simple	Strong
Cart	cartId (PK) Customer_customerId(FK) GrandTotal ItemsTotal	Simple Simple Derived Derived	Strong
Category	CategoryID(PK) CategoryName DESCRIPTION	Simple Simple Simple	Strong
seller	sellerId (PK) Name Phone Total_Sales	Simple Simple Multivalued Derived	Strong
Payment	payment_id (PK) Order_OrderId(FK) PaymentMode Customer_CustomerId Date_of_payment	Simple Simple Simple Simple Simple	Strong

Entities and Relations

Entities	Relation	Cardinality	Type of participation
Customer Address	Stays At	One To One	Total Partial
Customer Cart	Shops	One To One	Partial Total
Customer Order	Places	One To Many	Partial Total
Customer Payment	Makes	One To Many	Partial Total
Customer Review	Write	One To Many	Partial Total
Seller Product	Sells	Many To Many	Partial Total
Category Product	Categorizes	One To Many	Partial Total
Cart Product	Contains	Many To Many	Partial Partial

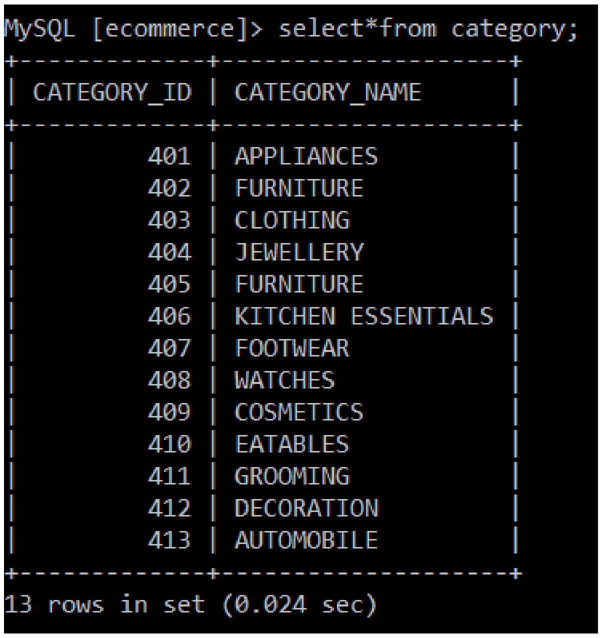
Product OrderItem	Includes	One To Many	Partial Total
Order OrderItem	Includes	One To One	Partial total
Payment Order	For	One To One	Total Total

QUERIES ON THE ABOVE RELATIONAL SCHEMA

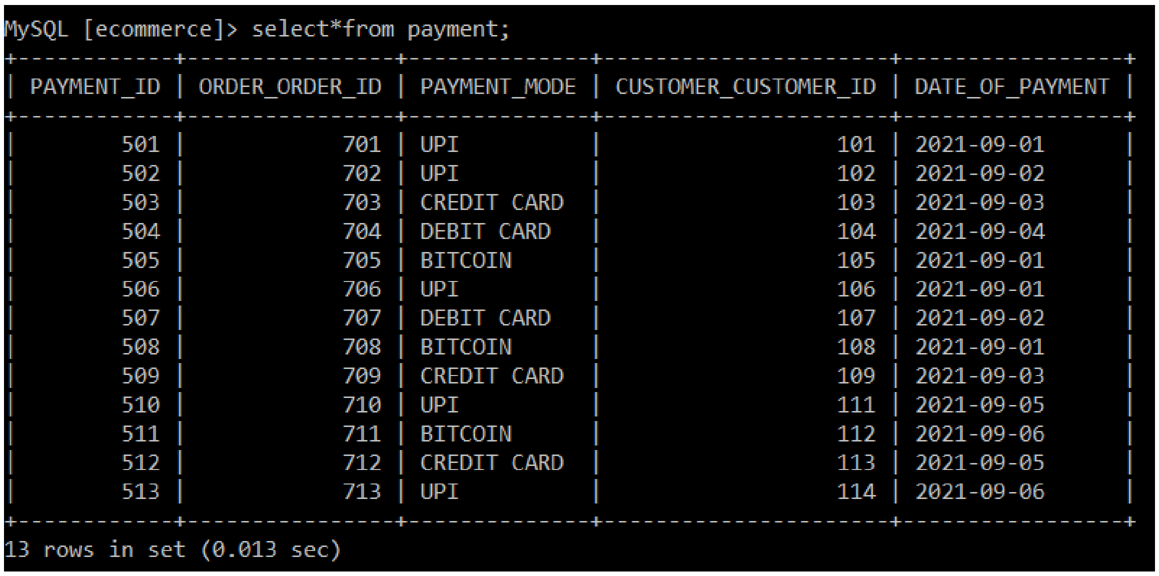
1.Stored procedure for the details of the customer.
2.View for getting sales by category of products.
3.Using triggers to update the no.of products as soon as the payment is made.
4.Stored procedure for getting order history.
6.

- Check whether the specified customer exists
- IF NOT EXISTS add him/her
- COMMIT the info
- Fetch the customer id
- INSERT a row to Order tables
- If unable to do so,ROLLBACK;
- Fetch the new orderid in orders table
- INSERT row to the order table for every product ordered
- If adding tuples to orderitems fails ROLL BACK all tuples of products added for and the tuple in order row

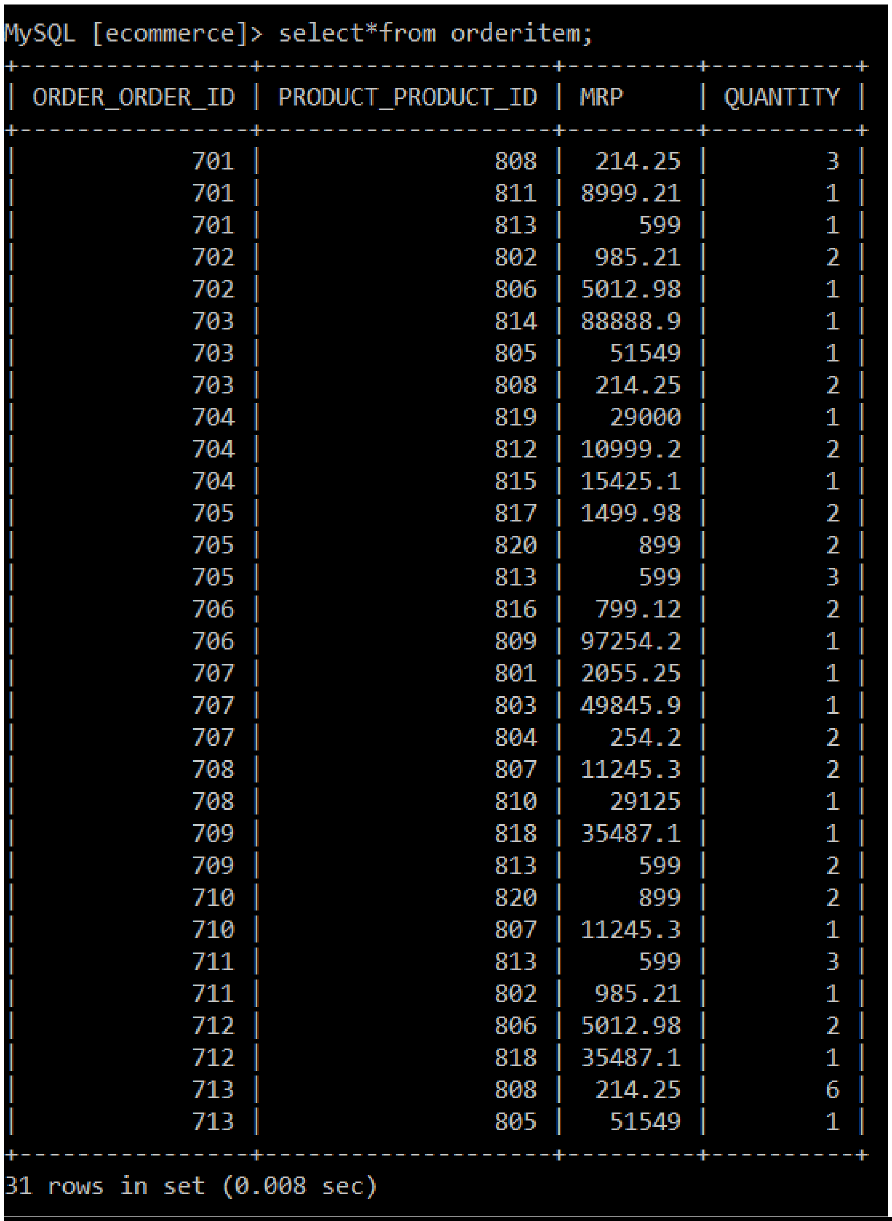
TABLES:



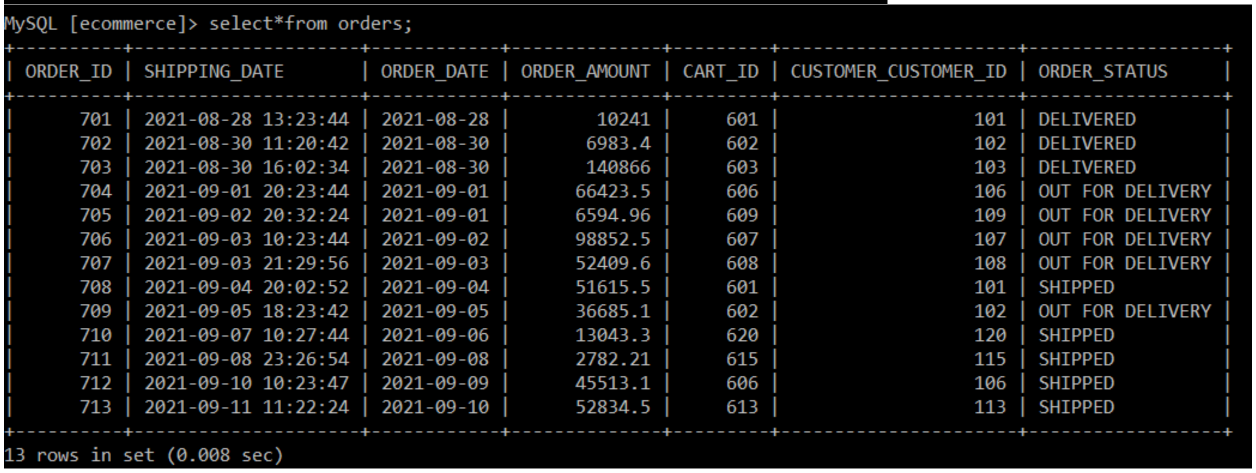
```
MySQL [ecommerce]> select*from category;
+-------------+--------------------+
| CATEGORY_ID | CATEGORY_NAME      |
+-------------+--------------------+
|         401 | APPLIANCES         |
|         402 | FURNITURE          |
|         403 | CLOTHING           |
|         404 | JEWELLERY          |
|         405 | FURNITURE          |
|         406 | KITCHEN ESSENTIALS |
|         407 | FOOTWEAR           |
|         408 | WATCHES            |
|         409 | COSMETICS          |
|         410 | EATABLES           |
|         411 | GROOMING           |
|         412 | DECORATION         |
|         413 | AUTOMOBILE         |
+-------------+--------------------+
13 rows in set (0.024 sec)
```



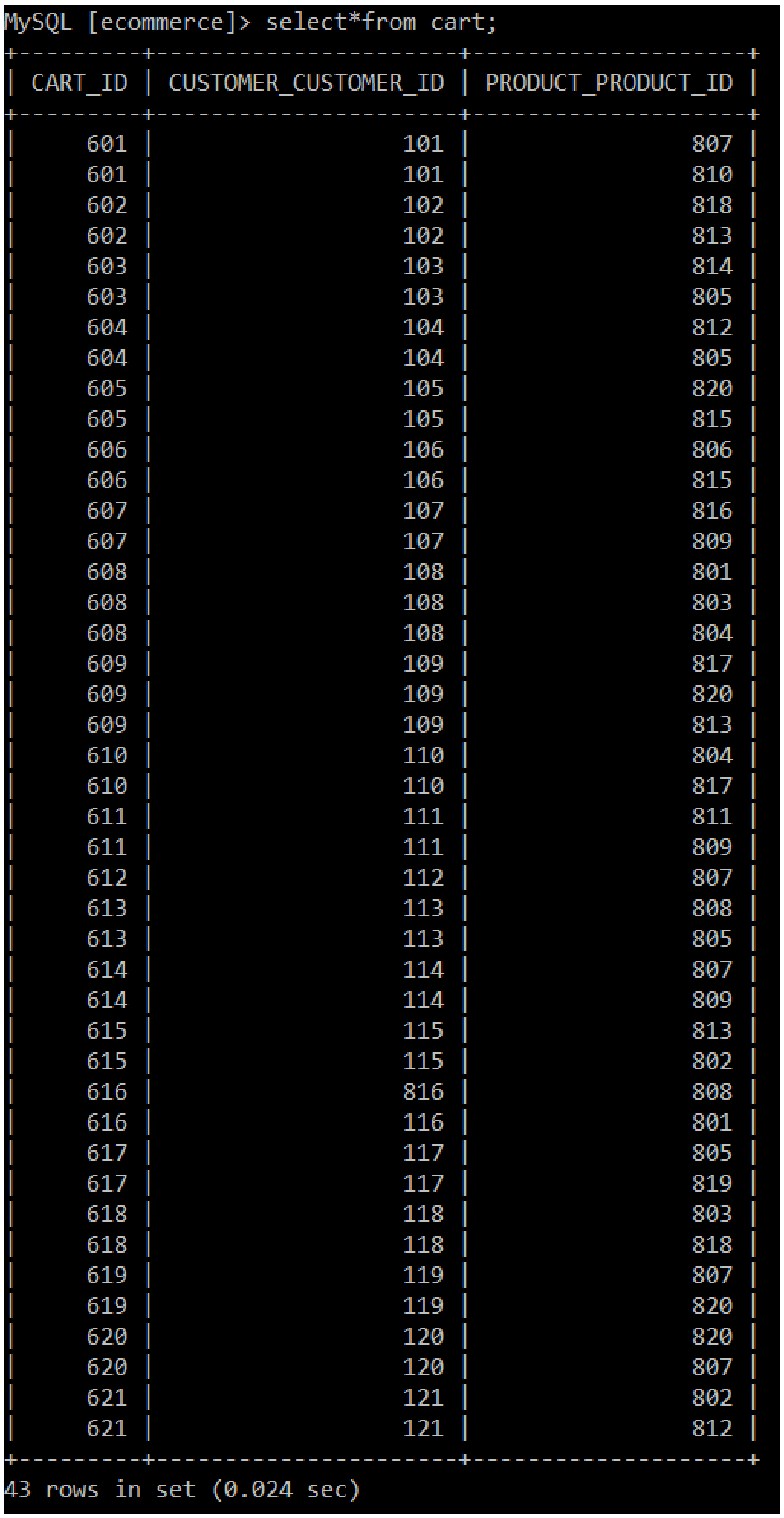
```
MySQL [ecommerce]> select*from payment;
+------------+----------------+--------------+----------------------+-----------------+
| PAYMENT_ID | ORDER_ORDER_ID | PAYMENT_MODE | CUSTOMER_CUSTOMER_ID | DATE_OF_PAYMENT |
+------------+----------------+--------------+----------------------+-----------------+
|        501 |            701 | UPI          |                  101 | 2021-09-01      |
|        502 |            702 | UPI          |                  102 | 2021-09-02      |
|        503 |            703 | CREDIT CARD  |                  103 | 2021-09-03      |
|        504 |            704 | DEBIT CARD   |                  104 | 2021-09-04      |
|        505 |            705 | BITCOIN      |                  105 | 2021-09-01      |
|        506 |            706 | UPI          |                  106 | 2021-09-01      |
|        507 |            707 | DEBIT CARD   |                  107 | 2021-09-02      |
|        508 |            708 | BITCOIN      |                  108 | 2021-09-01      |
|        509 |            709 | CREDIT CARD  |                  109 | 2021-09-03      |
|        510 |            710 | UPI          |                  111 | 2021-09-05      |
|        511 |            711 | BITCOIN      |                  112 | 2021-09-06      |
|        512 |            712 | CREDIT CARD  |                  113 | 2021-09-05      |
|        513 |            713 | UPI          |                  114 | 2021-09-06      |
+------------+----------------+--------------+----------------------+-----------------+
13 rows in set (0.013 sec)
```



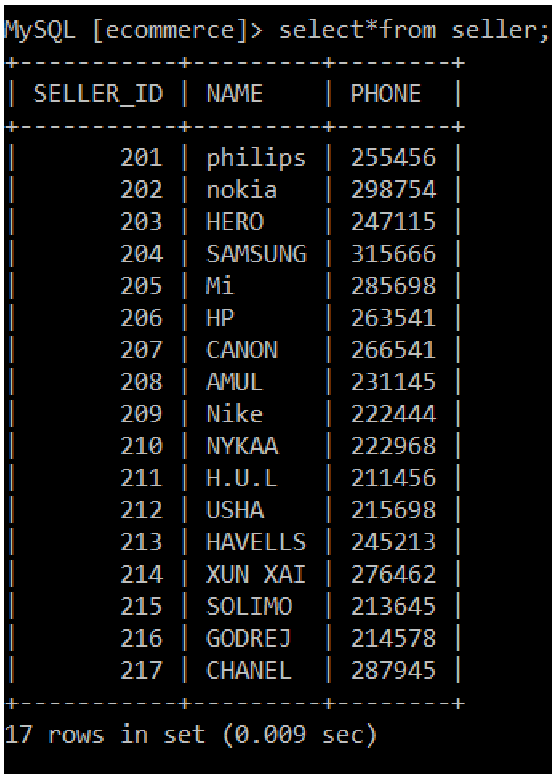
```
MySQL [ecommerce]> select*from orderitem;
+----------------+--------------------+---------+----------+
| ORDER_ORDER_ID | PRODUCT_PRODUCT_ID | MRP     | QUANTITY |
+----------------+--------------------+---------+----------+
|            701 |                808 |  214.25 |        3 |
|            701 |                811 | 8999.21 |        1 |
|            701 |                813 |     599 |        1 |
|            702 |                802 |  985.21 |        2 |
|            702 |                806 | 5012.98 |        1 |
|            703 |                814 | 88888.9 |        1 |
|            703 |                805 |   51549 |        1 |
|            703 |                808 |  214.25 |        2 |
|            704 |                819 |   29000 |        1 |
|            704 |                812 | 10999.2 |        2 |
|            704 |                815 | 15425.1 |        1 |
|            705 |                817 | 1499.98 |        2 |
|            705 |                820 |     899 |        2 |
|            705 |                813 |     599 |        3 |
|            706 |                816 |  799.12 |        2 |
|            706 |                809 | 97254.2 |        1 |
|            707 |                801 | 2055.25 |        1 |
|            707 |                803 | 49845.9 |        1 |
|            707 |                804 |   254.2 |        2 |
|            708 |                807 | 11245.3 |        2 |
|            708 |                810 |   29125 |        1 |
|            709 |                818 | 35487.1 |        1 |
|            709 |                813 |     599 |        2 |
|            710 |                820 |     899 |        2 |
|            710 |                807 | 11245.3 |        1 |
|            711 |                813 |     599 |        3 |
|            711 |                802 |  985.21 |        1 |
|            712 |                806 | 5012.98 |        2 |
|            712 |                818 | 35487.1 |        1 |
|            713 |                808 |  214.25 |        6 |
|            713 |                805 |   51549 |        1 |
+----------------+--------------------+---------+----------+
31 rows in set (0.008 sec)
```



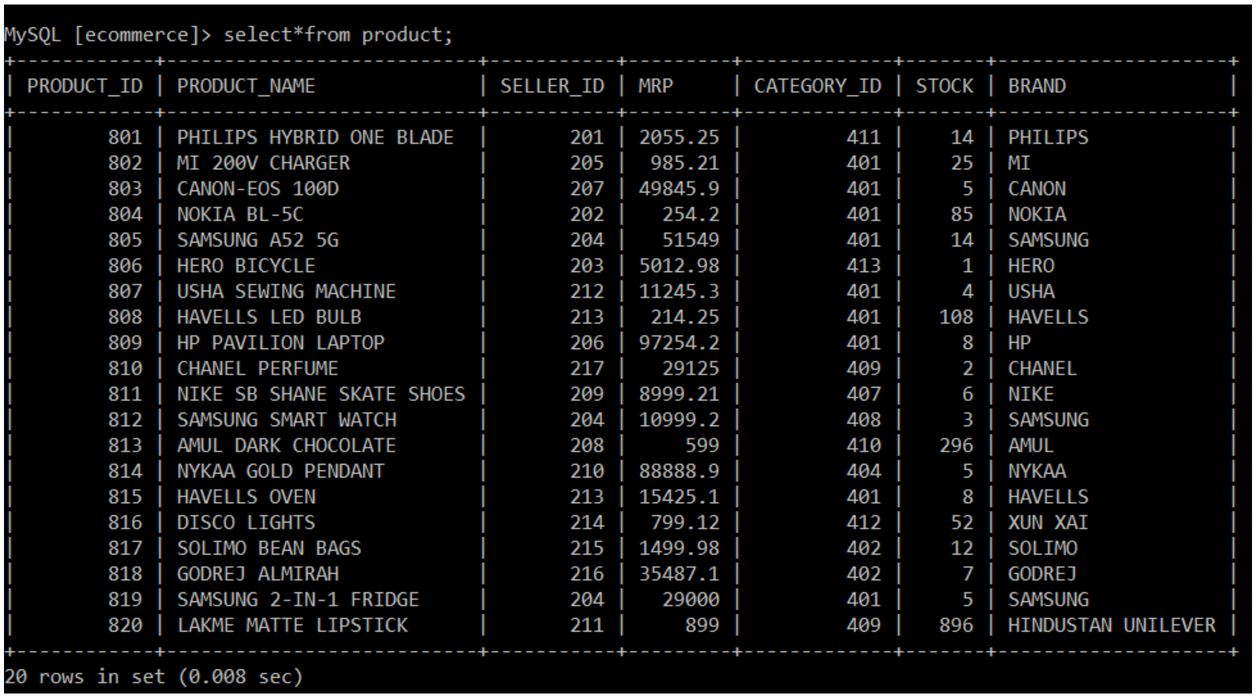
```
MySQL [ecommerce]> select*from orders;
+----------+---------------------+------------+--------------+---------+----------------------+------------------+
| ORDER_ID | SHIPPING_DATE       | ORDER_DATE | ORDER_AMOUNT | CART_ID | CUSTOMER_CUSTOMER_ID | ORDER_STATUS     |
+----------+---------------------+------------+--------------+---------+----------------------+------------------+
|      701 | 2021-08-28 13:23:44 | 2021-08-28 |        10241 |     601 |                  101 | DELIVERED        |
|      702 | 2021-08-30 11:20:42 | 2021-08-30 |       6983.4 |     602 |                  102 | DELIVERED        |
|      703 | 2021-08-30 16:02:34 | 2021-08-30 |       140866 |     603 |                  103 | DELIVERED        |
|      704 | 2021-09-01 20:23:44 | 2021-09-01 |      66423.5 |     606 |                  106 | OUT FOR DELIVERY |
|      705 | 2021-09-02 20:32:24 | 2021-09-01 |      6594.96 |     609 |                  109 | OUT FOR DELIVERY |
|      706 | 2021-09-03 10:23:44 | 2021-09-02 |      98852.5 |     607 |                  107 | OUT FOR DELIVERY |
|      707 | 2021-09-03 21:29:56 | 2021-09-03 |      52409.6 |     608 |                  108 | OUT FOR DELIVERY |
|      708 | 2021-09-04 20:02:52 | 2021-09-04 |      51615.5 |     601 |                  101 | SHIPPED          |
|      709 | 2021-09-05 18:23:42 | 2021-09-05 |      36685.1 |     602 |                  102 | OUT FOR DELIVERY |
|      710 | 2021-09-07 10:27:44 | 2021-09-06 |      13043.3 |     620 |                  120 | SHIPPED          |
|      711 | 2021-09-08 23:26:54 | 2021-09-08 |      2782.21 |     615 |                  115 | SHIPPED          |
|      712 | 2021-09-10 10:23:47 | 2021-09-09 |      45513.1 |     606 |                  106 | SHIPPED          |
|      713 | 2021-09-11 11:22:24 | 2021-09-10 |      52834.5 |     613 |                  113 | SHIPPED          |
+----------+---------------------+------------+--------------+---------+----------------------+------------------+
13 rows in set (0.008 sec)
```



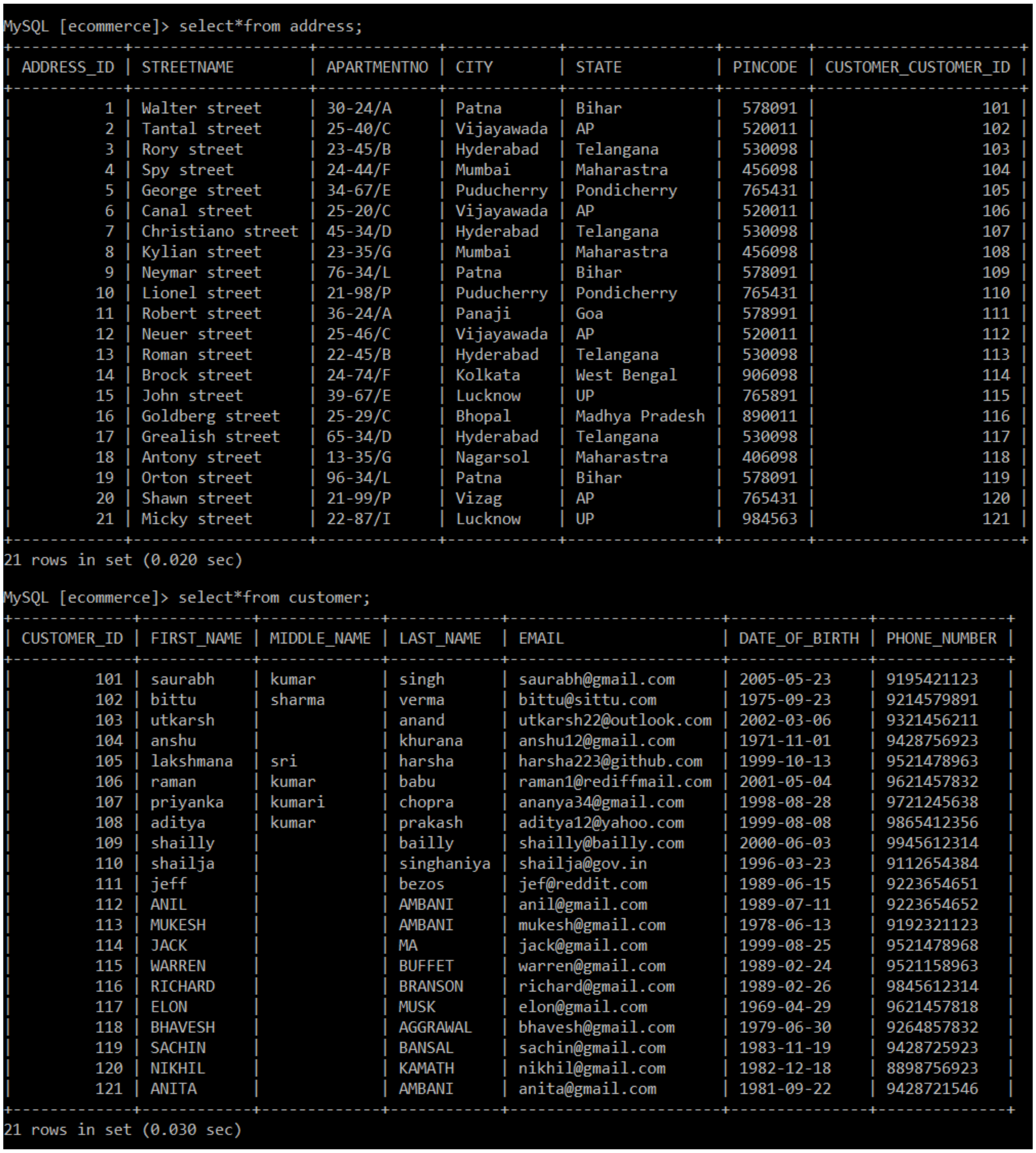
```
MySQL [ecommerce]> select*from cart;
+---------+----------------------+---------------------+
| CART_ID | CUSTOMER_CUSTOMER_ID | PRODUCT_PRODUCT_ID |
+---------+----------------------+---------------------+
|     601 |                  101 |                 807 |
|     601 |                  101 |                 810 |
|     602 |                  102 |                 818 |
|     602 |                  102 |                 813 |
|     603 |                  103 |                 814 |
|     603 |                  103 |                 805 |
|     604 |                  104 |                 812 |
|     604 |                  104 |                 805 |
|     605 |                  105 |                 820 |
|     605 |                  105 |                 815 |
|     606 |                  106 |                 806 |
|     606 |                  106 |                 815 |
|     607 |                  107 |                 816 |
|     607 |                  107 |                 809 |
|     608 |                  108 |                 801 |
|     608 |                  108 |                 803 |
|     608 |                  108 |                 804 |
|     609 |                  109 |                 817 |
|     609 |                  109 |                 820 |
|     609 |                  109 |                 813 |
|     610 |                  110 |                 804 |
|     610 |                  110 |                 817 |
|     611 |                  111 |                 811 |
|     611 |                  111 |                 809 |
|     612 |                  112 |                 807 |
|     613 |                  113 |                 808 |
|     613 |                  113 |                 805 |
|     614 |                  114 |                 807 |
|     614 |                  114 |                 809 |
|     615 |                  115 |                 813 |
|     615 |                  115 |                 802 |
|     616 |                  816 |                 808 |
|     616 |                  116 |                 801 |
|     617 |                  117 |                 805 |
|     617 |                  117 |                 819 |
|     618 |                  118 |                 803 |
|     618 |                  118 |                 818 |
|     619 |                  119 |                 807 |
|     619 |                  119 |                 820 |
|     620 |                  120 |                 820 |
|     620 |                  120 |                 807 |
|     621 |                  121 |                 802 |
|     621 |                  121 |                 812 |
+---------+----------------------+---------------------+
43 rows in set (0.024 sec)
```



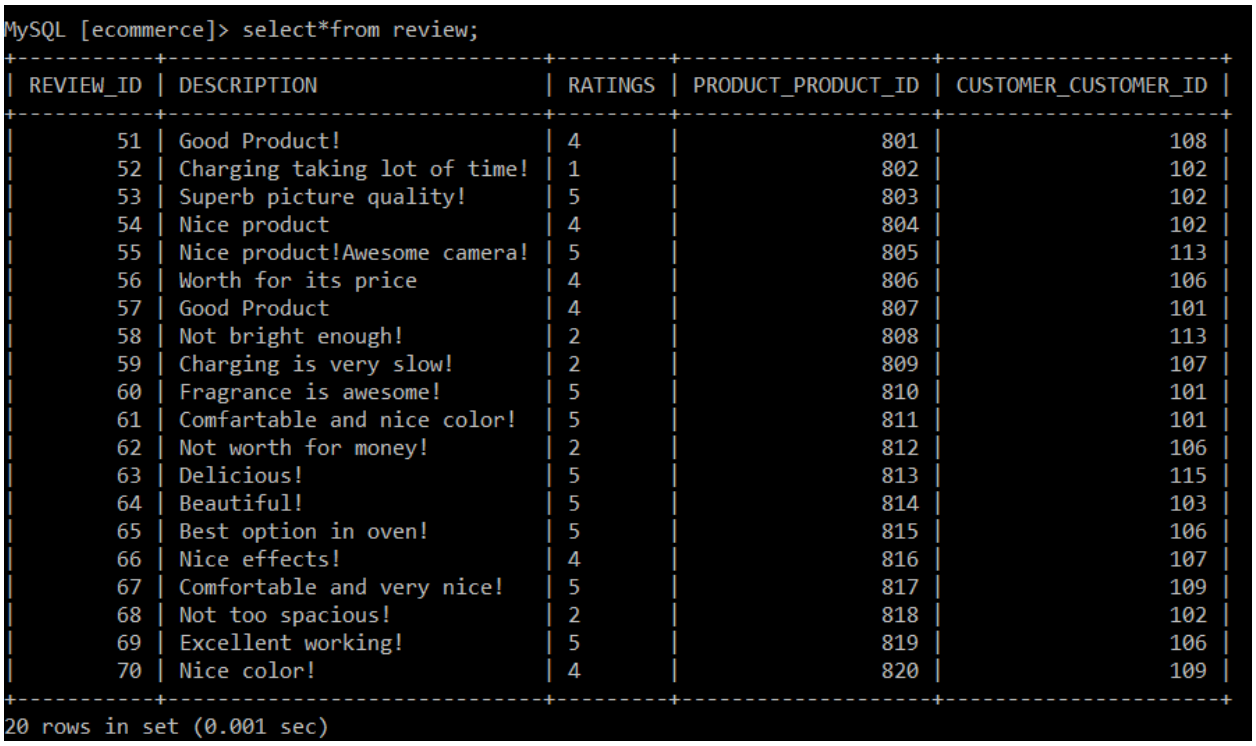
```
MySQL [ecommerce]> select*from seller;
+-----------+----------+--------+
| SELLER_ID | NAME     | PHONE  |
+-----------+----------+--------+
|       201 | philips  | 255456 |
|       202 | nokia    | 298754 |
|       203 | HERO     | 247115 |
|       204 | SAMSUNG  | 315666 |
|       205 | Mi       | 285698 |
|       206 | HP       | 263541 |
|       207 | CANON    | 266541 |
|       208 | AMUL     | 231145 |
|       209 | Nike     | 222444 |
|       210 | NYKAA    | 222968 |
|       211 | H.U.L    | 211456 |
|       212 | USHA     | 215698 |
|       213 | HAVELLS  | 245213 |
|       214 | XUN XAI  | 276462 |
|       215 | SOLIMO   | 213645 |
|       216 | GODREJ   | 214578 |
|       217 | CHANEL   | 287945 |
+-----------+----------+--------+
17 rows in set (0.009 sec)
```



```
MySQL [ecommerce]> select*from product;
+------------+---------------------------+-----------+---------+-------------+-------+--------------------+
| PRODUCT_ID | PRODUCT_NAME              | SELLER_ID | MRP     | CATEGORY_ID | STOCK | BRAND              |
+------------+---------------------------+-----------+---------+-------------+-------+--------------------+
|        801 | PHILIPS HYBRID ONE BLADE  |       201 | 2055.25 |         411 |    14 | PHILIPS            |
|        802 | MI 200V CHARGER           |       205 |  985.21 |         401 |    25 | MI                 |
|        803 | CANON-EOS 100D            |       207 | 49845.9 |         401 |     5 | CANON              |
|        804 | NOKIA BL-5C               |       202 |   254.2 |         401 |    85 | NOKIA              |
|        805 | SAMSUNG A52 5G            |       204 |   51549 |         401 |    14 | SAMSUNG            |
|        806 | HERO BICYCLE              |       203 | 5012.98 |         413 |     1 | HERO               |
|        807 | USHA SEWING MACHINE       |       212 | 11245.3 |         401 |     4 | USHA               |
|        808 | HAVELLS LED BULB          |       213 |  214.25 |         401 |   108 | HAVELLS            |
|        809 | HP PAVILION LAPTOP        |       206 | 97254.2 |         401 |     8 | HP                 |
|        810 | CHANEL PERFUME            |       217 |   29125 |         409 |     2 | CHANEL             |
|        811 | NIKE SB SHANE SKATE SHOES |       209 | 8999.21 |         407 |     6 | NIKE               |
|        812 | SAMSUNG SMART WATCH       |       204 | 10999.2 |         408 |     3 | SAMSUNG            |
|        813 | AMUL DARK CHOCOLATE       |       208 |     599 |         410 |   296 | AMUL               |
|        814 | NYKAA GOLD PENDANT        |       210 | 88888.9 |         404 |     5 | NYKAA              |
|        815 | HAVELLS OVEN              |       213 | 15425.1 |         401 |     8 | HAVELLS            |
|        816 | DISCO LIGHTS              |       214 |  799.12 |         412 |    52 | XUN XAI            |
|        817 | SOLIMO BEAN BAGS          |       215 | 1499.98 |         402 |    12 | SOLIMO             |
|        818 | GODREJ ALMIRAH            |       216 | 35487.1 |         402 |     7 | GODREJ             |
|        819 | SAMSUNG 2-IN-1 FRIDGE     |       204 |   29000 |         401 |     5 | SAMSUNG            |
|        820 | LAKME MATTE LIPSTICK      |       211 |     899 |         409 |   896 | HINDUSTAN UNILEVER |
+------------+---------------------------+-----------+---------+-------------+-------+--------------------+
20 rows in set (0.008 sec)
```



```
MySQL [ecommerce]> select*from address;
+------------+-------------------+-------------+------------+----------------+---------+----------------------+
| ADDRESS_ID | STREETNAME        | APARTMENTNO | CITY       | STATE          | PINCODE | CUSTOMER_CUSTOMER_ID |
+------------+-------------------+-------------+------------+----------------+---------+----------------------+
|          1 | Walter street     | 30-24/A     | Patna      | Bihar          |  578091 |                  101 |
|          2 | Tantal street     | 25-40/C     | Vijayawada | AP             |  520011 |                  102 |
|          3 | Rory street       | 23-45/B     | Hyderabad  | Telangana      |  530098 |                  103 |
|          4 | Spy street        | 24-44/F     | Mumbai     | Maharastra     |  456098 |                  104 |
|          5 | George street     | 34-67/E     | Puducherry | Pondicherry    |  765431 |                  105 |
|          6 | Canal street      | 25-20/C     | Vijayawada | AP             |  520011 |                  106 |
|          7 | Christiano street | 45-34/D     | Hyderabad  | Telangana      |  530098 |                  107 |
|          8 | Kylian street     | 23-35/G     | Mumbai     | Maharastra     |  456098 |                  108 |
|          9 | Neymar street     | 76-34/L     | Patna      | Bihar          |  578091 |                  109 |
|         10 | Lionel street     | 21-98/P     | Puducherry | Pondicherry    |  765431 |                  110 |
|         11 | Robert street     | 36-24/A     | Panaji     | Goa            |  578991 |                  111 |
|         12 | Neuer street      | 25-46/C     | Vijayawada | AP             |  520011 |                  112 |
|         13 | Roman street      | 22-45/B     | Hyderabad  | Telangana      |  530098 |                  113 |
|         14 | Brock street      | 24-74/F     | Kolkata    | West Bengal    |  906098 |                  114 |
|         15 | John street       | 39-67/E     | Lucknow    | UP             |  765891 |                  115 |
|         16 | Goldberg street   | 25-29/C     | Bhopal     | Madhya Pradesh |  890011 |                  116 |
|         17 | Grealish street   | 65-34/D     | Hyderabad  | Telangana      |  530098 |                  117 |
|         18 | Antony street     | 13-35/G     | Nagarsol   | Maharastra     |  406098 |                  118 |
|         19 | Orton street      | 96-34/L     | Patna      | Bihar          |  578091 |                  119 |
|         20 | Shawn street      | 21-99/P     | Vizag      | AP             |  765431 |                  120 |
|         21 | Micky street      | 22-87/I     | Lucknow    | UP             |  984563 |                  121 |
+------------+-------------------+-------------+------------+----------------+---------+----------------------+
21 rows in set (0.020 sec)

MySQL [ecommerce]> select*from customer;
+-------------+------------+-------------+------------+-----------------------+---------------+--------------+
| CUSTOMER_ID | FIRST_NAME | MIDDLE_NAME | LAST_NAME  | EMAIL                 | DATE_OF_BIRTH | PHONE_NUMBER |
+-------------+------------+-------------+------------+-----------------------+---------------+--------------+
|         101 | saurabh    | kumar       | singh      | saurabh@gmail.com     | 2005-05-23    | 9195421123   |
|         102 | bittu      | sharma      | verma      | bittu@sittu.com       | 1975-09-23    | 9214579891   |
|         103 | utkarsh    |             | anand      | utkarsh22@outlook.com | 2002-03-06    | 9321456211   |
|         104 | anshu      |             | khurana    | anshu12@gmail.com     | 1971-11-01    | 9428756923   |
|         105 | lakshmana  | sri         | harsha     | harsha223@github.com  | 1999-10-13    | 9521478963   |
|         106 | raman      | kumar       | babu       | raman1@rediffmail.com | 2001-05-04    | 9621457832   |
|         107 | priyanka   | kumari      | chopra     | ananya34@gmail.com    | 1998-08-28    | 9721245638   |
|         108 | aditya     | kumar       | prakash    | aditya12@yahoo.com    | 1999-08-08    | 9865412356   |
|         109 | shailly    |             | bailly     | shailly@bailly.com    | 2000-06-03    | 9945612314   |
|         110 | shailja    |             | singhaniya | shailja@gov.in        | 1996-03-23    | 9112654384   |
|         111 | jeff       |             | bezos      | jef@reddit.com        | 1989-06-15    | 9223654651   |
|         112 | ANIL       |             | AMBANI     | anil@gmail.com        | 1989-07-11    | 9223654652   |
|         113 | MUKESH     |             | AMBANI     | mukesh@gmail.com      | 1978-06-13    | 9192321123   |
|         114 | JACK       |             | MA         | jack@gmail.com        | 1999-08-25    | 9521478968   |
|         115 | WARREN     |             | BUFFET     | warren@gmail.com      | 1989-02-24    | 9521158963   |
|         116 | RICHARD    |             | BRANSON    | richard@gmail.com     | 1989-02-26    | 9845612314   |
|         117 | ELON       |             | MUSK       | elon@gmail.com        | 1969-04-29    | 9621457818   |
|         118 | BHAVESH    |             | AGGRAWAL   | bhavesh@gmail.com     | 1979-06-30    | 9264857832   |
|         119 | SACHIN     |             | BANSAL     | sachin@gmail.com      | 1983-11-19    | 9428725923   |
|         120 | NIKHIL     |             | KAMATH     | nikhil@gmail.com      | 1982-12-18    | 8898756923   |
|         121 | ANITA      |             | AMBANI     | anita@gmail.com       | 1981-09-22    | 9428721546   |
+-------------+------------+-------------+------------+-----------------------+---------------+--------------+
21 rows in set (0.030 sec)
```



```
MySQL [ecommerce]> select*from review;
+-----------+------------------------------+---------+--------------------+---------------------+
| REVIEW_ID | DESCRIPTION                  | RATINGS | PRODUCT_PRODUCT_ID | CUSTOMER_CUSTOMER_ID |
+-----------+------------------------------+---------+--------------------+---------------------+
|        51 | Good Product!                | 4       |                801 |                 108 |
|        52 | Charging taking lot of time! | 1       |                802 |                 102 |
|        53 | Superb picture quality!      | 5       |                803 |                 102 |
|        54 | Nice product                 | 4       |                804 |                 102 |
|        55 | Nice product!Awesome camera! | 5       |                805 |                 113 |
|        56 | Worth for its price          | 4       |                806 |                 106 |
|        57 | Good Product                 | 4       |                807 |                 101 |
|        58 | Not bright enough!           | 2       |                808 |                 113 |
|        59 | Charging is very slow!       | 2       |                809 |                 107 |
|        60 | Fragrance is awesome!        | 5       |                810 |                 101 |
|        61 | Comfartable and nice color!  | 5       |                811 |                 101 |
|        62 | Not worth for money!         | 2       |                812 |                 106 |
|        63 | Delicious!                   | 5       |                813 |                 115 |
|        64 | Beautiful!                   | 5       |                814 |                 103 |
|        65 | Best option in oven!         | 5       |                815 |                 106 |
|        66 | Nice effects!                | 4       |                816 |                 107 |
|        67 | Comfortable and very nice!   | 5       |                817 |                 109 |
|        68 | Not too spacious!            | 2       |                818 |                 102 |
|        69 | Excellent working!           | 5       |                819 |                 106 |
|        70 | Nice color!                  | 4       |                820 |                 109 |
+-----------+------------------------------+---------+--------------------+---------------------+
20 rows in set (0.001 sec)
```

<u>QUERIES:</u>

1.Customers to find products with highest ratings for a given category.

MySQL [ecommerce]> SELECT PRODUCT.PRODUCT_ID,PRODUCT_NAME,REVIEW.RATINGS FROM PRODUCT JOIN CATEGORY ON PRODUCT.CATEGORY_ID=CATEGORY.CATEGORY_ID JOIN REVIEW ON PRODUCT.PRODUCT_ID=REVIEW.PRODUCT_PRODUCT_ID WHERE REVIEW.RATINGS=(SELECT MAX(REVIEW.RATINGS) FROM PRODUCT JOIN CATEGORY ON PRODUCT.CATEGORY_ID=CATEGORY.CATEGORY_ID JOIN REVIEW ON PRODUCT.PRODUCT_ID=REVIEW.PRODUCT_PRODUCT_ID WHERE CATEGORY.CATEGORY_NAME='APPLIANCES') AND CATEGORY.CATEGORY_NAME='APPLIANCES';

<u>OUTPUT:</u>

```
+------------+----------------------+---------+
| PRODUCT_ID | PRODUCT_NAME         | RATINGS |
+------------+----------------------+---------+
|        803 | CANON-EOS 100D       | 5       |
|        805 | SAMSUNG A52 5G        | 5       |
|        815 | HAVELLS OVEN          | 5       |
|        819 | SAMSUNG 2-IN-1 FRIDGE | 5       |
+------------+-----------------------+---------+
3 rows in set (0.030 sec)
```

2.Customers filter out the products according to their brand and price.

```
MySQL [ecommerce]> SELECT PRODUCT_ID,PRODUCT_NAME,MRP FROM
PRODUCT WHERE BRAND='SAMSUNG' AND MRP BETWEEN 10000 AND 30000;
```

OUTPUT:

```
+------------+-----------------------+---------+
| PRODUCT_ID | PRODUCT_NAME          | MRP     |
+------------+-----------------------+---------+
|        812 | SAMSUNG SMART WATCH   | 10999.2 |
|        819 | SAMSUNG 2-IN-1 FRIDGE |   29000 |
+------------+-----------------------+---------+
2 rows in set (0.009 sec)
```

3.Customers compare the products based on their ratings and reviews.

```
DELIMITER //
CREATE PROCEDURE p5 (pr1 INT,pr2 INT)
  BEGIN
   SELECT
PRODUCT.PRODUCT_ID,PRODUCT.PRODUCT_NAME,REVIEW.RATINGS,REVIEW.DE
SCRIPTION FROM REVIEW JOIN PRODUCT
   ON REVIEW.PRODUCT_PRODUCT_ID=PRODUCT.PRODUCT_ID WHERE
PRODUCT.PRODUCT_ID=pr1 OR PRODUCT.PRODUCT_ID=pr2;
  END;//
```

OUTPUT:

```
MySQL [ecommerce]> call p5(802,804);
```

```
+------------+-----------------+---------+-----------------------------+
| PRODUCT_ID | PRODUCT_NAME    | RATINGS | DESCRIPTION                 |
+------------+-----------------+---------+-----------------------------+
|        802 | MI 200V CHARGER | 1       | Charging taking lot of time! |
|        804 | NOKIA BL-5C     | 4       | Nice product                |
+------------+-----------------+---------+-----------------------------+
2 rows in set (0.027 sec)
```

4.List the orders which are to be delivered at a particular pincode.

```
MySQL [ecommerce]> SELECT
ORDERS.ORDER_ID,ORDERS.ORDER_DATE,ORDERS.CUSTOMER_CUSTOMER_ID,C
ONCAT(CUSTOMER.FIRST_NAME,' ',CUSTOMER.MIDDLE_NAME,'
',CUSTOMER.LAST_NAME) AS CUSTOMER_NAME,ADDRESS.PINCODE FROM
CUSTOMER JOIN ORDERS ON
CUSTOMER.CUSTOMER_ID=ORDERS.CUSTOMER_CUSTOMER_ID JOIN ADDRESS
ON CUSTOMER.CUSTOMER_ID=ADDRESS.CUSTOMER_CUSTOMER_ID WHERE
ORDERS.ORDER_STATUS='OUT FOR DELIVERY' AND ADDRESS.PINCODE='520011';
```

<u>OUTPUT:</u>

```
+----------+------------+----------------------+--------------------+---------+
| ORDER_ID | ORDER_DATE | CUSTOMER_CUSTOMER_ID | CUSTOMER_NAME      | PINCODE |
+----------+------------+----------------------+--------------------+---------+
|      709 | 2021-09-05 |          102         | bittu sharma verma |520011|
|      704 | 2021-09-01 |          106         | raman kumar babu   | 520011|
+----------+------------+----------------------+--------------------+---------+
2 rows in set (0.000 sec)
```

5.List the product whose sale is the highest on a particular day.

```
DELIMITER //
CREATE PROCEDURE query(O_DATE DATE)
  BEGIN
  DECLARE CNT INT DEFAULT 801;
  DECLARE MAXID INT DEFAULT 820;
  DECLARE TEMP INT DEFAULT 0;
  DECLARE MAX_CNT INT default 0;
  DECLARE MAXP_ID INT;
  WHILE CNT<=MAXID DO
  select sum(ORDERITEM.quantity) INTO TEMP from orderitem JOIN ORDERS ON
ORDERS.ORDER_ID=ORDERITEM.ORDER_ORDER_ID where
ORDERITEM.product_product_id=CNT AND ORDERS.ORDER_DATE=O_DATE;
  IF TEMP > MAX_CNT THEN
  SET MAX_CNT = TEMP;
  SET MAXP_ID = CNT;
  END IF;
  set CNT =CNT+1;
  END WHILE;
SELECT
ORDERS.ORDER_DATE,ORDERITEM.PRODUCT_PRODUCT_ID,PRODUCT.PRODUCT_
NAME,SUM(ORDERITEM.QUANTITY) AS SALE_COUNT FROM ORDERS JOIN
ORDERITEM ON ORDERS.ORDER_ID=ORDERITEM.ORDER_ORDER_ID
  JOIN PRODUCT ON
PRODUCT.PRODUCT_ID=ORDERITEM.PRODUCT_PRODUCT_ID WHERE
ORDERITEM.PRODUCT_PRODUCT_ID=MAXP_ID AND
ORDERS.ORDER_DATE=O_DATE;
  END;//
```

```
MySQL [ecommerce]> call query('2021-09-01');
```

<u>OUTPUT:</u>

```
+------------+--------------------+------------------+------------+
| ORDER_DATE | PRODUCT_PRODUCT_ID | PRODUCT_NAME     | SALE_COUNT |
+------------+--------------------+------------------+------------+
| 2021-09-01 |                817 | SOLIMO BEAN BAGS |          2 |
+------------+--------------------+------------------+------------+
1 row in set (0.011 sec)

Query OK, 0 rows affected (0.017 sec)

MySQL [ecommerce]> call query('2021-09-10');
```

<u>OUTPUT:</u>

```
+------------+--------------------+------------------+------------+
| ORDER_DATE | PRODUCT_PRODUCT_ID | PRODUCT_NAME     | SALE_COUNT |
+------------+--------------------+------------------+------------+
| 2021-09-10 |                808 | HAVELLS LED BULB |          6 |
+------------+--------------------+------------------+------------+
1 row in set (0.002 sec)

Query OK, 0 rows affected (0.008 sec)
```

6.Write a procedure to calculate total order amount of all orders.

```
DELIMITER //
CREATE PROCEDURE p1 ()
  BEGIN
    DECLARE CNT INT DEFAULT 701;
    DECLARE MAXID INT DEFAULT 713;
    WHILE CNT<=MAXID DO
     UPDATE ORDERS SET ORDER_AMOUNT=(SELECT SUM(MRP*QUANTITY) FROM
ORDERITEM WHERE ORDER_ORDER_ID=CNT) WHERE ORDER_ID=CNT;
     SET CNT=CNT+1;
    END WHILE;
  END;//
```

7.List how many times a particular customer bought from different sellers.

```
MySQL [ecommerce]> select
orderitem.order_order_id,orders.customer_customer_id,orderitem.product_pro
duct_id,product.product_name,product.seller_id,count(product.seller_id),seller.n
ame from orders join orderitem on orders.order_id=orderitem.order_order_id
join product on product.product_id=orderitem.product_product_id join seller on
seller.seller_id=product.seller_id where orders.customer_customer_id='106'
group by product.seller_id order by orderitem.product_product_id;
```

OUTPUT:

```
+----------------+----------------------+--------------------+-----------------------+-----------+--------------------------+---------+
| order_order_id | customer_customer_id | product_product_id | product_name          | seller_id | count(product.seller_id) | name    |
+----------------+----------------------+--------------------+-----------------------+-----------+--------------------------+---------+
|        712        |            106             |            806         | HERO BICYCLE         |      203 |                 1                | HERO    |
|        704        |            106             |            815         | HAVELLS OVEN         |      213 |                 1                | HAVELLS |
|        712        |            106             |            818         | GODREJ ALMIRAH       |      216 |                 1                | GODREJ  |
|        704        |            106             |            819         | SAMSUNG 2-IN-1 FRIDGE |      204 |           2                | SAMSUNG |
+----------------+---------------------+--------------------+----------------------+-----------+-------------------------+---------+
4 rows in set (0.001 sec)
```

8.List all the orders whose payment mode is Credit Card and yet to be delivered.

MySQL [ecommerce]> SELECT ORDERS.ORDER_ID,ORDERS.CUSTOMER_CUSTOMER_ID,ORDERS.ORDER_STATUS, PAYMENT.PAYMENT_MODE FROM ORDERS JOIN PAYMENT ON ORDERS.ORDER_ID=PAYMENT.ORDER_ORDER_ID WHERE PAYMENT.PAYMENT_MODE='CREDIT CARD' AND ORDERS.ORDER_STATUS='SHIPPED';

OUTPUT:

```
+----------+----------------------+--------------+--------------+
| ORDER_ID | CUSTOMER_CUSTOMER_ID | ORDER_STATUS | PAYMENT_MODE |
+----------+----------------------+--------------+--------------+
|      712 |                  106 | SHIPPED      | CREDIT CARD  |
+----------+----------------------+--------------+--------------+
1 row in set (0.000 sec)
```

9.List all orders of customers whose total amount is less than 10000.

MySQL [ecommerce]> select*from orders where order_amount<'10000';

OUTPUT:

```
+----------+---------------------+------------+--------------+---------+----------------------+------------------+
| ORDER_ID | SHIPPING_DATE       | ORDER_DATE | ORDER_AMOUNT | CART_ID | CUSTOMER_CUSTOMER_ID | ORDER_STATUS     |
+----------+---------------------+------------+--------------+---------+----------------------+------------------+
|      702 | 2021-08-30 11:20:42 | 2021-08-30 |       6983.4 |     602 |                  102 | DELIVERED        |
|      705 | 2021-09-02 20:32:24 | 2021-09-01 |      6594.96 |     609 |                  109 | OUT FOR DELIVERY |
|      711 | 2021-09-08 23:26:54 | 2021-09-08 |      2782.21 |     615 |                  115 | SHIPPED          |
+----------+---------------------+------------+--------------+---------+----------------------+------------------+
3 rows in set (0.008 sec)
```

10.List the product and its seller which has the highest stock.

```
MySQL [ecommerce]> select product_id,product_name,seller_id,brand,stock
from product where stock=(select max(stock) from product);
```

OUTPUT:

```
+------------+----------------------+-----------+--------------------+-------+
| product_id | product_name         | seller_id | brand              | stock |
+------------+----------------------+-----------+--------------------+-------+
|        820 | LAKME MATTE LIPSTICK |       211 | HINDUSTAN UNILEVER |   896 |
+------------+----------------------+-----------+--------------------+-------+
1 row in set (0.000 sec)
```