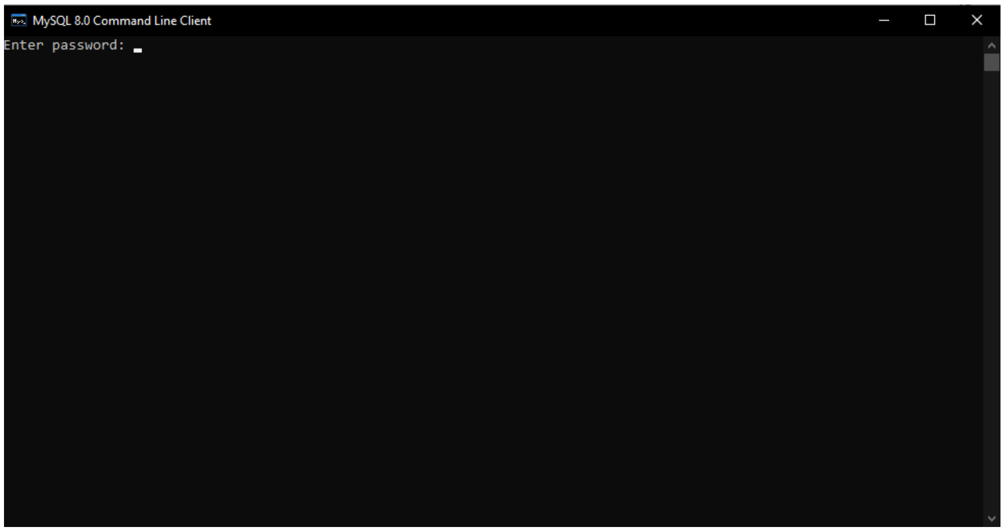
How to Start?

Step 1 - Download MySQL Community Server using this link - https://dev.mysql.com/downloads/mysql/

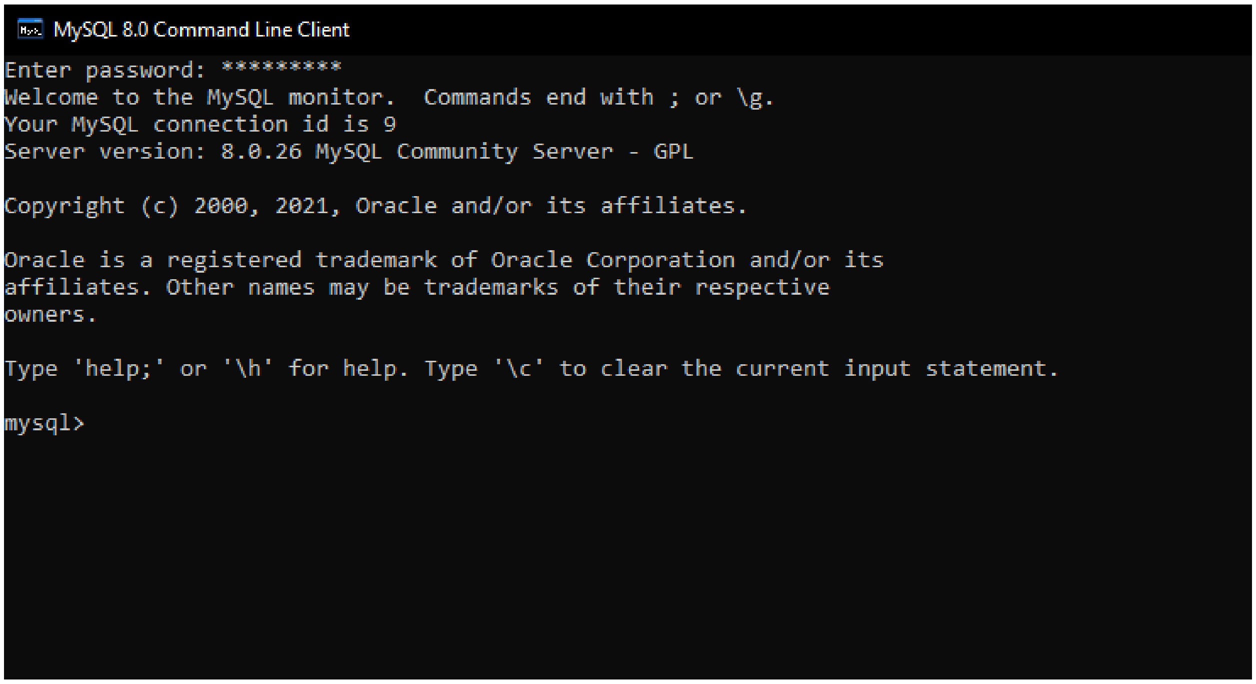
Step 2 - Follow the setup steps and set root password for mysql

Step 3 - Open MySQL Command Line Client



```
Enter password: _
```

Step 4 - Enter root password



```
Enter password: *********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

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

mysql>
```

Step 5 - Begin the fun

Database Management

Create Database

Command to create database : CREATE DATABASE [IF NOT EXISTS] database_name;

```
mysql> create database if not exists trial;
Query OK, 1 row affected (0.04 sec)
```

The if not exists phrase is used to check whether a database already pre-existed or not. If the same name database already existed then it would not have created the database.

Select Database

Command to select/activate database : USE <database_name>;

```
mysql> use trial;
Database changed
```

Show Databases

Command to show databases : SHOW DATABASES;

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| fulfillment_centre |
| information_schema |
| mysql              |
| orderdatabase      |
| performance_schema |
| sakila             |
| student_report     |
| sys                |
| trial              |
| university         |
| world              |
+--------------------+
11 rows in set (0.01 sec)
```

Drop Database

Command to drop database : DROP DATABASE [IF EXISTS] database_name;

The if exists phrase is not mandatory and is used as a safety measure. If table does not exist and we pass the command to drop it then, the command will do nothing.

```
mysql> drop database trial;
Query OK, 0 rows affected (0.18 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| fulfillment_centre |
| information_schema |
| mysql              |
| orderdatabase      |
| performance_schema |
| sakila             |
| student_report     |
| sys                |
| university         |
| world              |
+--------------------+
10 rows in set (0.01 sec)
```

Table Management and Views

Create Table

```
Command to create table : CREATE TABLE [IF NOT EXISTS] table_name(
   column_definition1,
   column_definition2,
   ........,
   table_constraints
);

mysql> create table emp_table(
   -> id int,
   -> name varchar(45),
   -> occupation varchar(35),
   -> age int
   -> );
Query OK, 0 rows affected (0.11 sec)
```

Alter Table

Add Columns

```
Command to add column : ALTER TABLE table_name
ADD new_column_name column_definition
[ FIRST | AFTER column_name ];

mysql> alter table emp_table
```

```
    -> add emp_sal int
    -> after age;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Modify Columns

Command to modify column : ALTER TABLE table_name
MODIFY column_name column_definition
[FIRST | AFTER column_name];

```
mysql> alter table emp_table
    -> modify id varchar(10);
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Drop Columns

Command to modify column : ALTER TABLE table_name
DROP COLUMN column_name;

```
mysql> alter table emp_table
    -> drop column age;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Rename Columns

Command to rename column : ALTER TABLE table_name
CHANGE COLUMN old_name new_name column_definition
[FIRST | AFTER column_name]

```
mysql> alter table emp_table
    -> change column id emp_id varchar(10);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Rename Table

Command to rename table : ALTER TABLE table_name
RENAME TO new_table_name;

```
mysql> alter table emp_table
    -> rename to employees;
Query OK, 0 rows affected (0.04 sec)
```

Show Tables

Command to rename table : SHOW TABLES;

```
mysql> show tables;
+-----------------+
| Tables_in_trial |
+-----------------+
| employees       |
+-----------------+
1 row in set (0.04 sec)
```

Describe Tables

Command to rename table : {DESCRIBE | DESC} table_name;

```
mysql> desc employees;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| emp_id     | varchar(10) | YES  |     | NULL    |       |
| name       | varchar(45) | YES  |     | NULL    |       |
| occupation | varchar(35) | YES  |     | NULL    |       |
| emp_sal    | int         | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

Copy Tables

Command to copy table : CREATE TABLE IF NOT EXISTS duplicate_table
SELECT * FROM original_table;

```
mysql> create table if not exists emp_info
    -> select * from employees;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Drop Tables

Command to drop table : DROP TABLE [IF EXISTS] table_name;

```
mysql> drop table if exists employees;
Query OK, 0 rows affected (0.04 sec)
```

Create Views

Command to create views : CREATE [OR REPLACE] VIEW view_name AS
SELECT columns
FROM tables
[WHERE conditions];

OR REPLACE: It is optional. It is used when a VIEW already exists. If you do not specify this clause and the VIEW already exists, the CREATE VIEW statement will return an error.

WHERE conditions: It is also optional. It specifies the conditions that must be met for the records to be included in the VIEW.

```
mysql> create view emp_personal_info as
    -> select name
    -> from emp_info;
Query OK, 0 rows affected (0.02 sec)
```

See Created Views

Command to see created views : SELECT * FROM view_name;

```
mysql> select * from emp_personal_info;
Empty set (0.01 sec)
```

Update Views

Command to update views : ALTER VIEW view_name AS
SELECT columns
FROM table
[WHERE conditions];

```
mysql> alter view emp_personal_info as
    -> select name
    -> from emp_info
    -> where name like "A%";                                          (Names starting with A)
Query OK, 0 rows affected (0.02 sec)
```

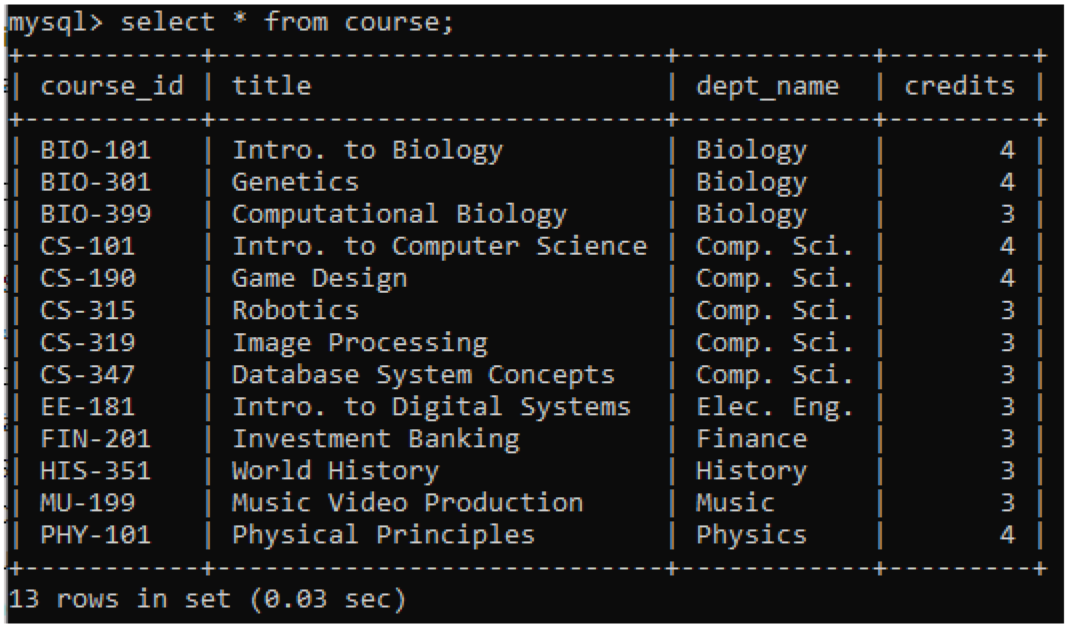
Drop Views

Command to drop views : DROP VIEW [IF EXISTS] view_name;

```
mysql> drop view emp_personal_info;
Query OK, 0 rows affected (0.02 sec)
```

Keys

Course Table



```
mysql> select * from course;
+-----------+----------------------------+------------+---------+
| course_id | title                      | dept_name  | credits |
+-----------+----------------------------+------------+---------+
| BIO-101   | Intro. to Biology          | Biology    |       4 |
| BIO-301   | Genetics                   | Biology    |       4 |
| BIO-399   | Computational Biology      | Biology    |       3 |
| CS-101    | Intro. to Computer Science | Comp. Sci. |       4 |
| CS-190    | Game Design                | Comp. Sci. |       4 |
| CS-315    | Robotics                   | Comp. Sci. |       3 |
| CS-319    | Image Processing           | Comp. Sci. |       3 |
| CS-347    | Database System Concepts   | Comp. Sci. |       3 |
| EE-181    | Intro. to Digital Systems  | Elec. Eng. |       3 |
| FIN-201   | Investment Banking         | Finance    |       3 |
| HIS-351   | World History              | History    |       3 |
| MU-199    | Music Video Production     | Music      |       3 |
| PHY-101   | Physical Principles        | Physics    |       4 |
+-----------+----------------------------+------------+---------+
13 rows in set (0.03 sec)
```

Unique Key

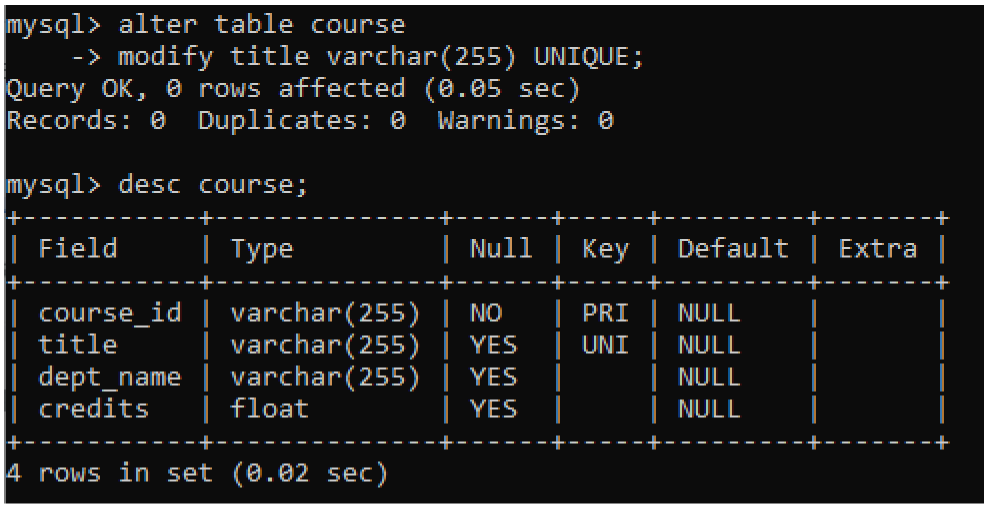
Command :

While creating a table:

```
CREATE TABLE table_name(
    col1 datatype,
    col2 datatype UNIQUE,
    ...
);
```

While altering a table:

```
ALTER TABLE <table_name>
[MODIFY | ADD CONSTRAINT] <col_name> <col_type> UNIQUE;
```



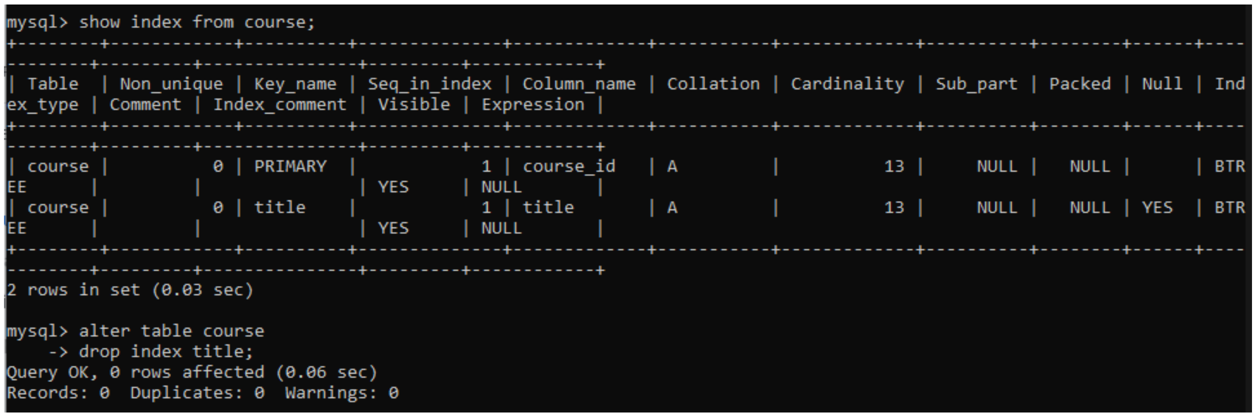
```
mysql> alter table course
    -> modify title varchar(255) UNIQUE;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc course;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| course_id | varchar(255) | NO   | PRI | NULL    |       |
| title     | varchar(255) | YES  | UNI | NULL    |       |
| dept_name | varchar(255) | YES  |     | NULL    |       |
| credits   | float        | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

Command to drop the Unique Key:

First: SHOW INDEX FROM <table_name>;

Note the name of the constraint in the index table.

Second: ALTER TABLE table_name DROP INDEX constraint_name;



```
mysql> show index from course;
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+----
--------+---------+---------------+---------+------------+
| Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Ind
ex_type | Comment | Index_comment | Visible | Expression |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+----
--------+---------+---------------+---------+------------+
| course |          0 | PRIMARY  |            1 | course_id   | A         |          13 |     NULL |   NULL |      | BTR
EE      |         |               | YES     | NULL       |
| course |          0 | title    |            1 | title       | A         |          13 |     NULL |   NULL | YES  | BTR
EE      |         |               | YES     | NULL       |
+--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+----
--------+---------+---------------+---------+------------+
2 rows in set (0.03 sec)

mysql> alter table course
    -> drop index title;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Primary Key

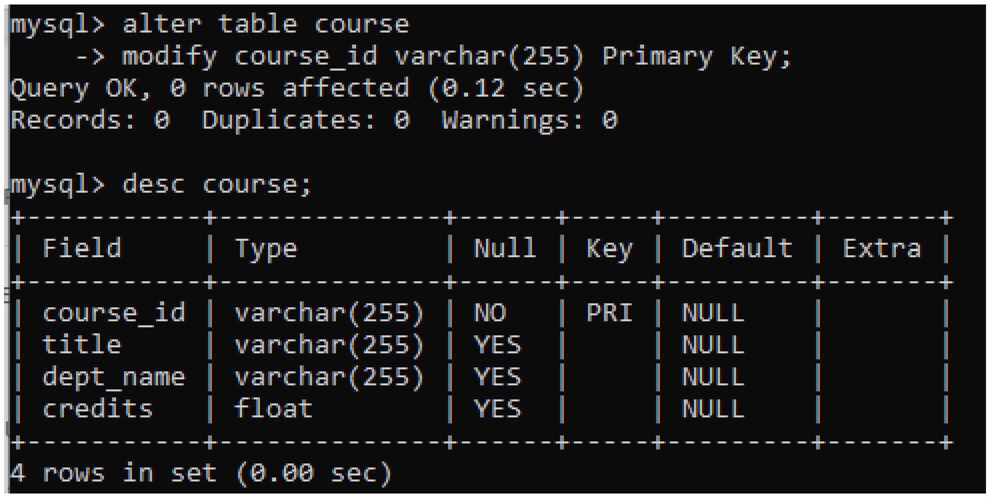
Command :

While creating a table:

```
CREATE TABLE table_name(
    col1 datatype PRIMARY KEY,
    col2 datatype,
    ...
);
```

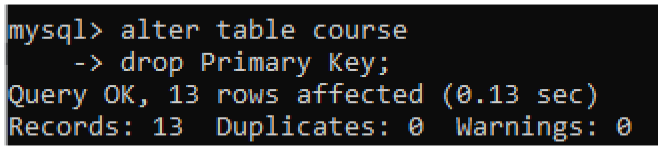
While altering a table:

```
ALTER TABLE <table_name>
[MODIFY | ADD CONSTRAINT] <col_name> <col_type> PRIMARY KEY
```



```
mysql> alter table course
    -> modify course_id varchar(255) Primary Key;
Query OK, 0 rows affected (0.12 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc course;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| course_id | varchar(255) | NO   | PRI | NULL    |       |
| title     | varchar(255) | YES  |     | NULL    |       |
| dept_name | varchar(255) | YES  |     | NULL    |       |
| credits   | float        | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```
;

Command to drop Primary Key: ALTER TABLE table_name DROP PRIMARY KEY;



```
mysql> alter table course
    -> drop Primary Key;
Query OK, 13 rows affected (0.13 sec)
Records: 13  Duplicates: 0  Warnings: 0
```

Foreign Key

```
Command : [CONSTRAINT <constraint_name>]
   FOREIGN KEY <foreign_key_name> (<col_name, …>)
   REFERENCES <parent_tbl_name> (<col_name,...>)
   [ON DELETE referenceOption]
   [ON UPDATE referenceOption]
```

```
Using alter table: ALTER TABLE table_name
   ADD [CONSTRAINT <constraint_name>] FOREIGN KEY
   <foreign_key_name> (column_name, ...)
   REFERENCES table_name (column_name,...)
   [ON DELETE referenceOption]
   [ON UPDATE referenceOption]
```

Description of command:

constraint_name: It specifies the name of the foreign key constraint. If we have not provided the constraint name, MySQL generates its name automatically.

col_name: It is the name of the column that will be converted to a foreign key.

parent_tbl_name: It specifies the name of a parent table followed by column names that reference the foreign key columns.

Refrence_option: It is used to ensure that the foreign key maintains referential integrity using ON DELETE and ON UPDATE clause between parent and child table.

MySQL contains five different referential options, which are given below:

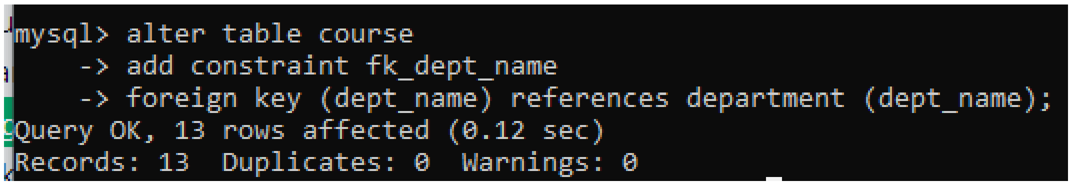
CASCADE: It is used when we delete or update any row from the parent table, the values of the matching rows in the child table will be deleted or updated automatically.

SET NULL: It is used when we delete or update any row from the parent table, the values of the foreign key columns in the child table are set to NULL.

RESTRICT: It is used when we delete or update any row from the parent table that has a matching row in the reference(child) table, MySQL does not allow us to delete or update rows in the parent table.

NO ACTION: It is similar to RESTRICT but it has one difference, it checks referential integrity after trying to modify the table.

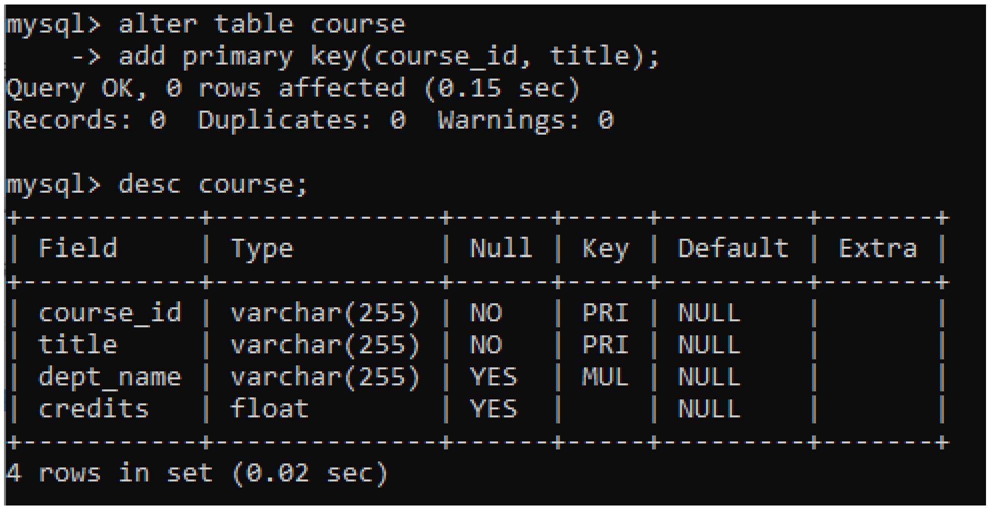
SET DEFAULT: The MySQL parser recognizes this action. However, the InnoDB and NDB tables both rejected this action.



```
mysql> alter table course
    -> add constraint fk_dept_name
    -> foreign key (dept_name) references department (dept_name);
Query OK, 13 rows affected (0.12 sec)
Records: 13  Duplicates: 0  Warnings: 0
```

Composite Key

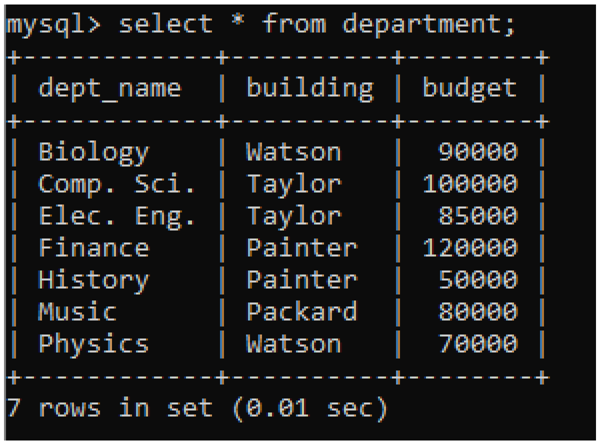
Command : PRIMARY KEY(<col_name_1>, <col_name_2>,....)



```
mysql> alter table course
    -> add primary key(course_id, title);
Query OK, 0 rows affected (0.15 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc course;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| course_id | varchar(255) | NO   | PRI | NULL    |       |
| title     | varchar(255) | NO   | PRI | NULL    |       |
| dept_name | varchar(255) | YES  | MUL | NULL    |       |
| credits   | float        | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

Clauses and Filtering

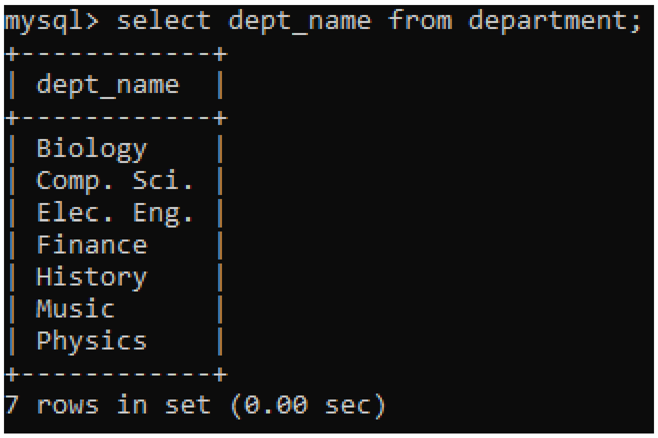
Department Table



```
mysql> select * from department;
+------------+----------+--------+
| dept_name  | building | budget |
+------------+----------+--------+
| Biology    | Watson   |  90000 |
| Comp. Sci. | Taylor   | 100000 |
| Elec. Eng. | Taylor   |  85000 |
| Finance    | Painter  | 120000 |
| History    | Painter  |  50000 |
| Music      | Packard  |  80000 |
| Physics    | Watson   |  70000 |
+------------+----------+--------+
7 rows in set (0.01 sec)
```

From

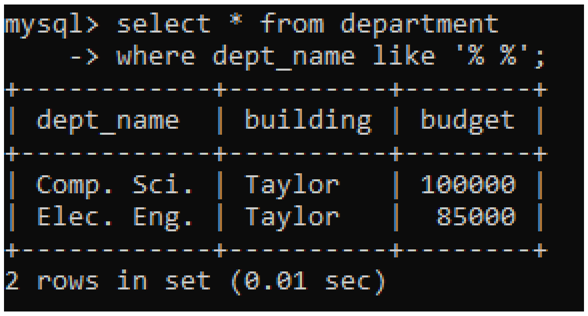
Command : SELECT statements FROM <table_name>;



```
mysql> select dept_name from department;
+------------+
| dept_name  |
+------------+
| Biology    |
| Comp. Sci. |
| Elec. Eng. |
| Finance    |
| History    |
| Music      |
| Physics    |
+------------+
7 rows in set (0.00 sec)
```

Where

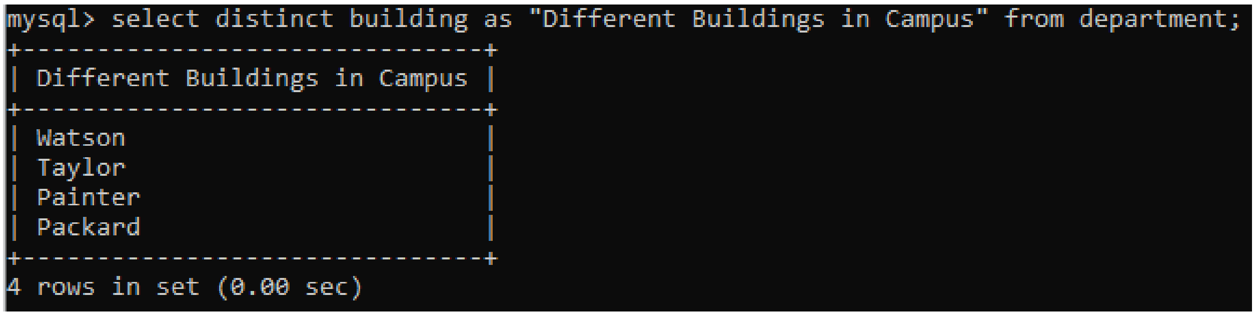
Command : WHERE conditions;



```
mysql> select * from department
    -> where dept_name like '% %';
+------------+----------+--------+
| dept_name  | building | budget |
+------------+----------+--------+
| Comp. Sci. | Taylor   | 100000 |
| Elec. Eng. | Taylor   |  85000 |
+------------+----------+--------+
2 rows in set (0.01 sec)
```

Distinct

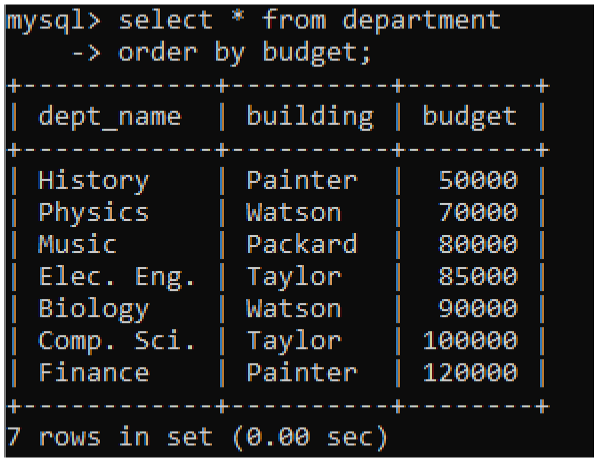
Command : SELECT DISTINCT expressions
FROM tables
[WHERE conditions];



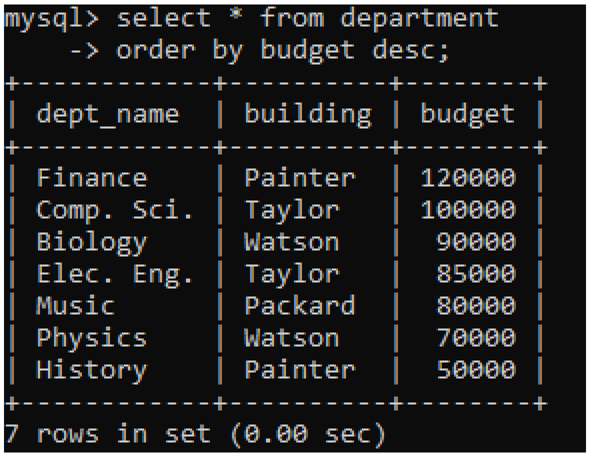
```
mysql> select distinct building as "Different Buildings in Campus" from department;
+-------------------------------+
| Different Buildings in Campus |
+-------------------------------+
| Watson                        |
| Taylor                        |
| Painter                       |
| Packard                       |
+-------------------------------+
4 rows in set (0.00 sec)
```

Order By

Command : ORDER BY <column_name> [desc]



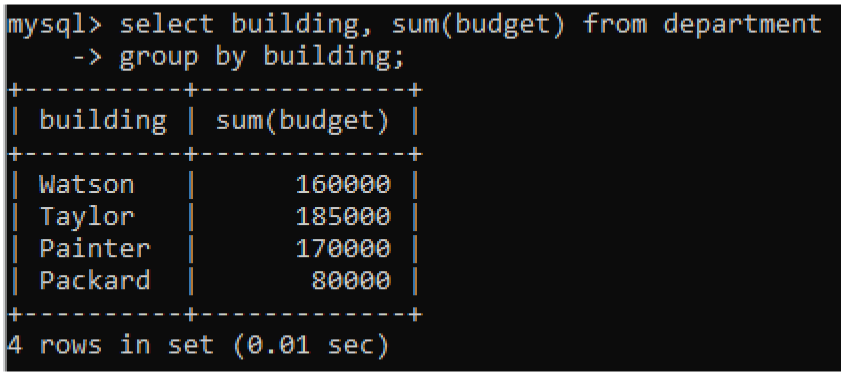
```
mysql> select * from department
    -> order by budget;
+------------+----------+--------+
| dept_name  | building | budget |
+------------+----------+--------+
| History    | Painter  |  50000 |
| Physics    | Watson   |  70000 |
| Music      | Packard  |  80000 |
| Elec. Eng. | Taylor   |  85000 |
| Biology    | Watson   |  90000 |
| Comp. Sci. | Taylor   | 100000 |
| Finance    | Painter  | 120000 |
+------------+----------+--------+
7 rows in set (0.00 sec)
```



```
mysql> select * from department
    -> order by budget desc;
+------------+----------+--------+
| dept_name  | building | budget |
+------------+----------+--------+
| Finance    | Painter  | 120000 |
| Comp. Sci. | Taylor   | 100000 |
| Biology    | Watson   |  90000 |
| Elec. Eng. | Taylor   |  85000 |
| Music      | Packard  |  80000 |
| Physics    | Watson   |  70000 |
| History    | Painter  |  50000 |
+------------+----------+--------+
7 rows in set (0.00 sec)
```

Group By

Command : GROUP BY <column_name>

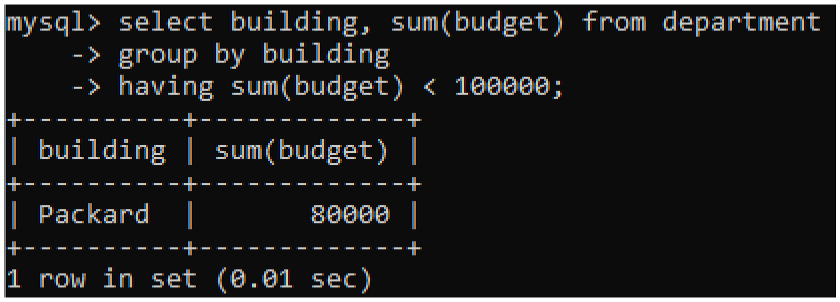


```
mysql> select building, sum(budget) from department
    -> group by building;
+----------+-------------+
| building | sum(budget) |
+----------+-------------+
| Watson   |      160000 |
| Taylor   |      185000 |
| Painter  |      170000 |
| Packard  |       80000 |
+----------+-------------+
4 rows in set (0.01 sec)
```

Having

Command : HAVING conditions;

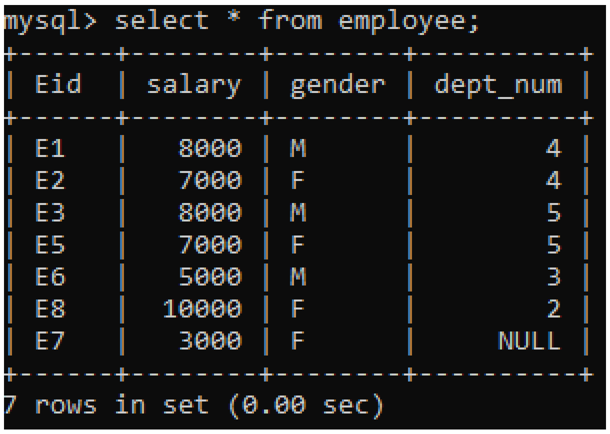
Here conditions are mainly on aggregate functions since where clause does not handle aggregate functions.



```
mysql> select building, sum(budget) from department
    -> group by building
    -> having sum(budget) < 100000;
+----------+-------------+
| building | sum(budget) |
+----------+-------------+
| Packard  |       80000 |
+----------+-------------+
1 row in set (0.01 sec)
```

Conditions

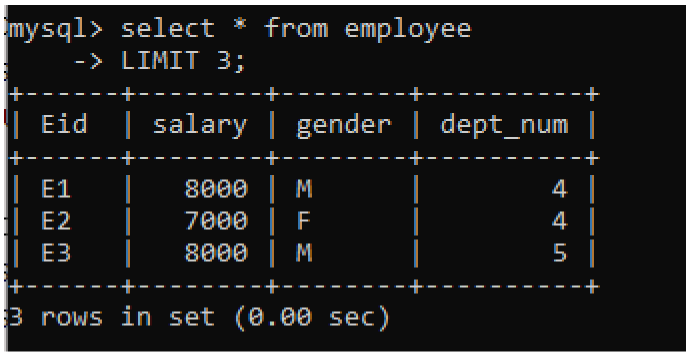
Employee Table:



```
mysql> select * from employee;
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E1   |   8000 | M      |        4 |
| E2   |   7000 | F      |        4 |
| E3   |   8000 | M      |        5 |
| E5   |   7000 | F      |        5 |
| E6   |   5000 | M      |        3 |
| E8   |  10000 | F      |        2 |
| E7   |   3000 | F      |     NULL |
+------+--------+--------+----------+
7 rows in set (0.00 sec)
```

LIMIT

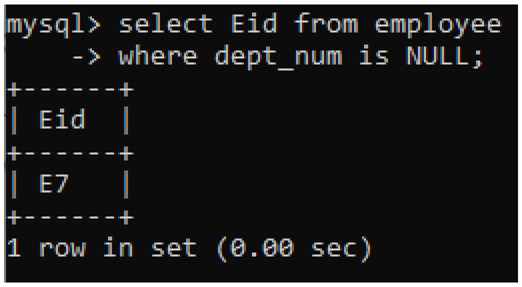
Command : LIMIT <number_of_desired_rows>



```
mysql> select * from employee
    -> LIMIT 3;
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E1   |   8000 | M      |        4 |
| E2   |   7000 | F      |        4 |
| E3   |   8000 | M      |        5 |
+------+--------+--------+----------+
3 rows in set (0.00 sec)
```

IS NULL

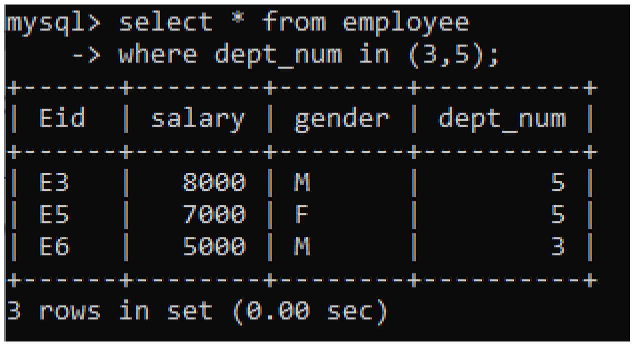
Command : expression IS NULL



```
mysql> select Eid from employee
    -> where dept_num is NULL;
+------+
| Eid  |
+------+
| E7   |
+------+
1 row in set (0.00 sec)
```

IN

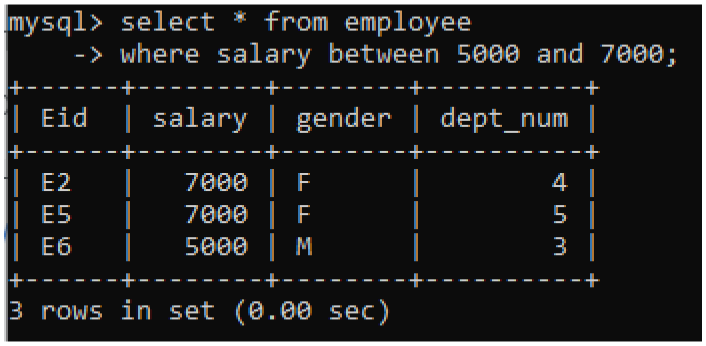
Command : expression IN (value1, value2, ... value_n);



```
mysql> select * from employee
    -> where dept_num in (3,5);
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E3   |   8000 | M      |        5 |
| E5   |   7000 | F      |        5 |
| E6   |   5000 | M      |        3 |
+------+--------+--------+----------+
3 rows in set (0.00 sec)
```

BETWEEN

Command : expression BETWEEN value1 AND value2;



```
mysql> select * from employee
    -> where salary between 5000 and 7000;
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E2   |   7000 | F      |        4 |
| E5   |   7000 | F      |        5 |
| E6   |   5000 | M      |        3 |
+------+--------+--------+----------+
3 rows in set (0.00 sec)
```

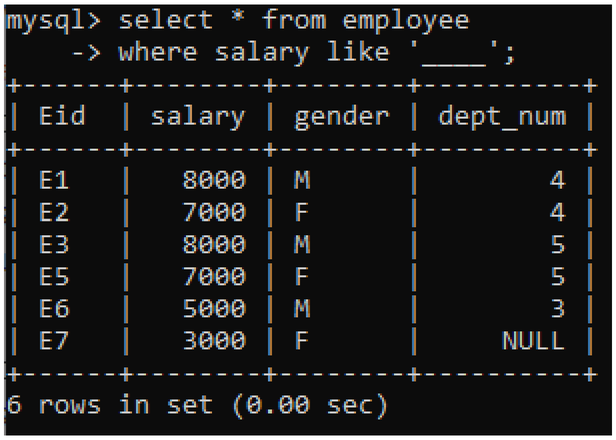
LIKE

Command : expression LIKE pattern

Here the pattern is always mentioned in single quotes (‘’). The pattern can include all kinds of characters with two special characters (called wild cards) having a particular meaning.

1. _ - Underscore - denotes a single character
2. % - Percentage - denotes multiple characters

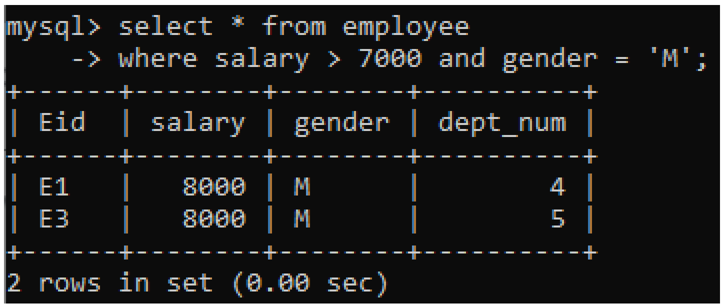
Let's enter the pattern with 4 underscores meaning 4 digits only.



```
mysql> select * from employee
    -> where salary like '____';
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E1   |   8000 | M      |        4 |
| E2   |   7000 | F      |        4 |
| E3   |   8000 | M      |        5 |
| E5   |   7000 | F      |        5 |
| E6   |   5000 | M      |        3 |
| E7   |   3000 | F      |     NULL |
+------+--------+--------+----------+
6 rows in set (0.00 sec)
```

AND

Command : WHERE condition1
AND condition2
...
AND condition_n;

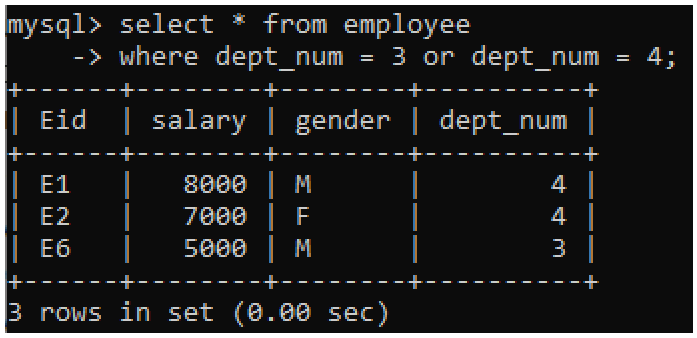


```
mysql> select * from employee
    -> where salary > 7000 and gender = 'M';
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E1   |   8000 | M      |        4 |
| E3   |   8000 | M      |        5 |
+------+--------+--------+----------+
2 rows in set (0.00 sec)
```

OR

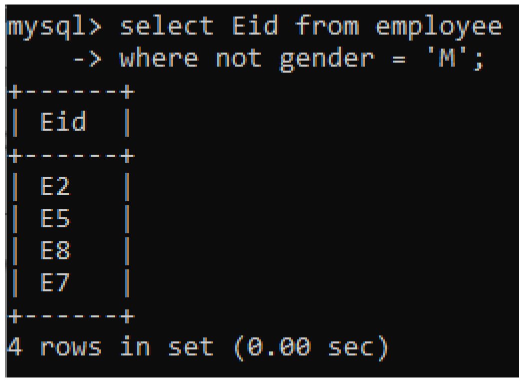
Command : WHERE condition1
OR condition2

...
OR condition_n;


```
mysql> select * from employee
    -> where dept_num = 3 or dept_num = 4;
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E1   |   8000 | M      |        4 |
| E2   |   7000 | F      |        4 |
| E6   |   5000 | M      |        3 |
+------+--------+--------+----------+
3 rows in set (0.00 sec)
```

NOT

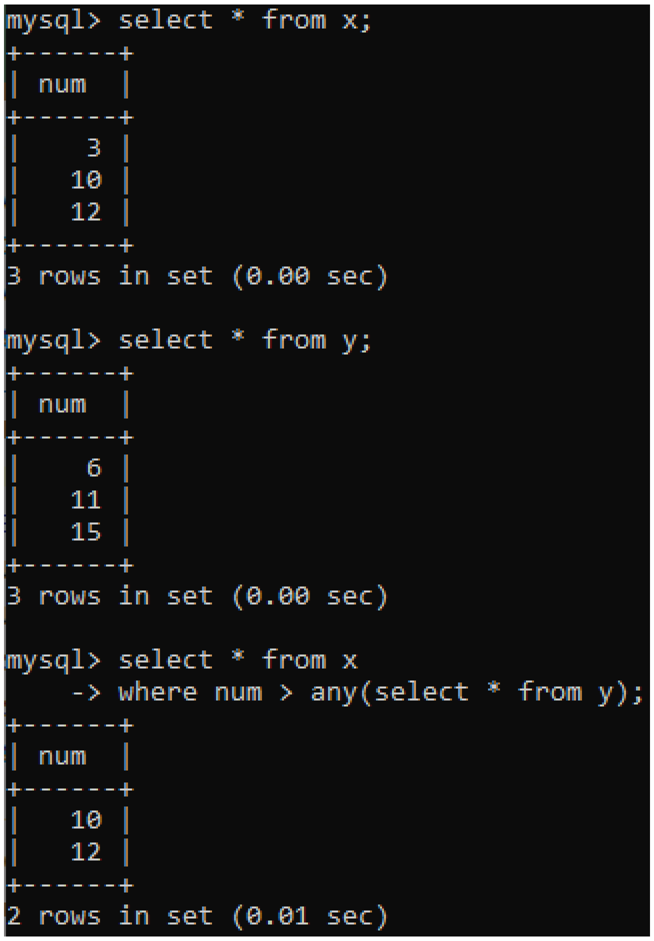
Command : NOT condition


```
mysql> select Eid from employee
    -> where not gender = 'M';
+------+
| Eid  |
+------+
| E2   |
| E5   |
| E8   |
| E7   |
+------+
4 rows in set (0.00 sec)
```

ANY

Command : operand comparison_operator ANY (subquery)

Comparison operator : =(equal), >(greater than), <(less than), >=(greater than equal to), <=(less than equal to), != (not equal to).

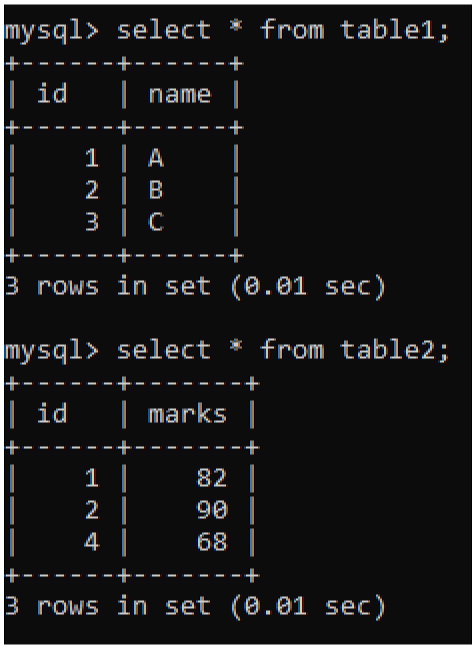


```
mysql> select * from x;
+------+
| num  |
+------+
|    3 |
|   10 |
|   12 |
+------+
3 rows in set (0.00 sec)

mysql> select * from y;
+------+
| num  |
+------+
|    6 |
|   11 |
|   15 |
+------+
3 rows in set (0.00 sec)

mysql> select * from x
    -> where num > any(select * from y);
+------+
| num  |
+------+
|   10 |
|   12 |
+------+
2 rows in set (0.01 sec)
```

Joins

MySQL JOINS are used with the SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.

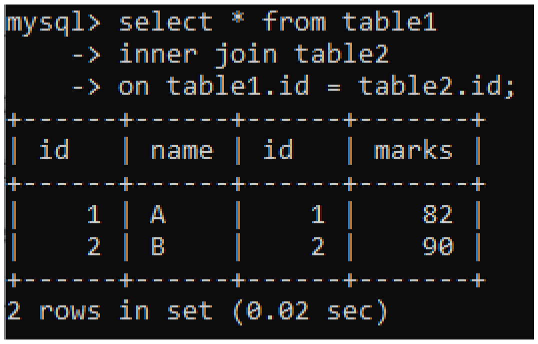
Table 1 and Table 2



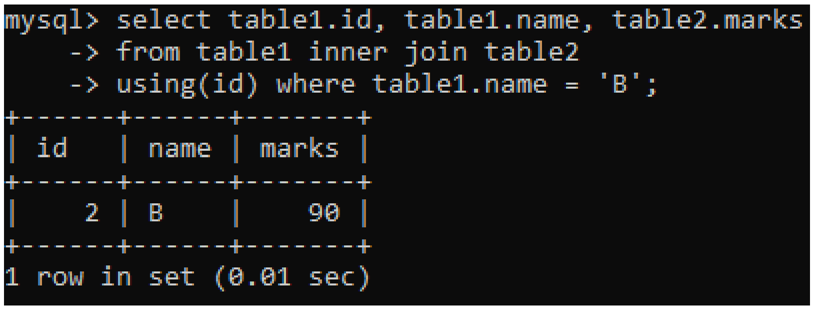
```
mysql> select * from table1;
+------+------+
| id   | name |
+------+------+
|    1 | A    |
|    2 | B    |
|    3 | C    |
+------+------+
3 rows in set (0.01 sec)

mysql> select * from table2;
+------+-------+
| id   | marks |
+------+-------+
|    1 |    82 |
|    2 |    90 |
|    4 |    68 |
+------+-------+
3 rows in set (0.01 sec)
```

Inner Join

Command : SELECT columns
FROM table1
INNER JOIN table2 <ON condition1 | USING (condition1.column)>
INNER JOIN table3 <ON condition2 | USING (condition2.column)>
...;



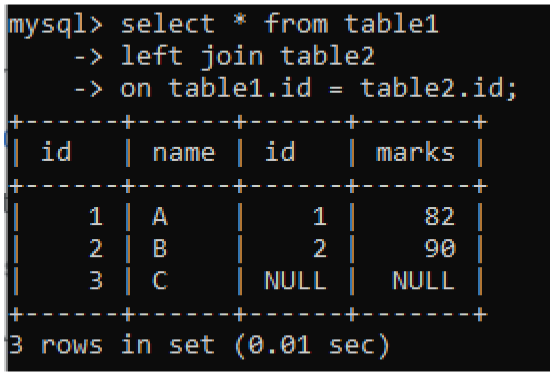
```
mysql> select * from table1
    -> inner join table2
    -> on table1.id = table2.id;
+------+------+------+-------+
| id   | name | id   | marks |
+------+------+------+-------+
|    1 | A    |    1 |    82 |
|    2 | B    |    2 |    90 |
+------+------+------+-------+
2 rows in set (0.02 sec)
```



```
mysql> select table1.id, table1.name, table2.marks
    -> from table1 inner join table2
    -> using(id) where table1.name = 'B';
+------+------+-------+
| id   | name | marks |
+------+------+-------+
|    2 | B    |    90 |
+------+------+-------+
1 row in set (0.01 sec)
```

Left Outer Join (Left Join)

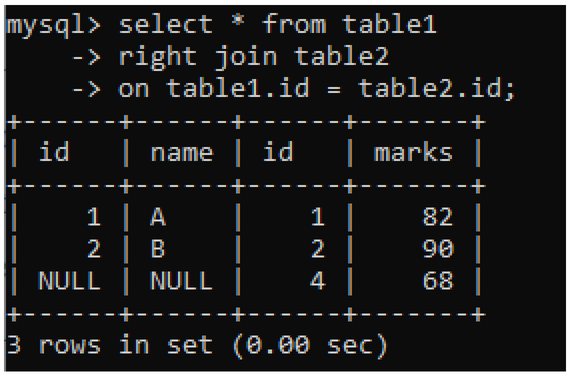
Command : SELECT columns
FROM table1
LEFT [OUTER] JOIN table2
ON Join_Condition;



```
mysql> select * from table1
    -> left join table2
    -> on table1.id = table2.id;
+------+------+------+-------+
| id   | name | id   | marks |
+------+------+------+-------+
|    1 | A    |    1 |    82 |
|    2 | B    |    2 |    90 |
|    3 | C    | NULL |  NULL |
+------+------+------+-------+
3 rows in set (0.01 sec)
```

Right Outer Join (Right Join)

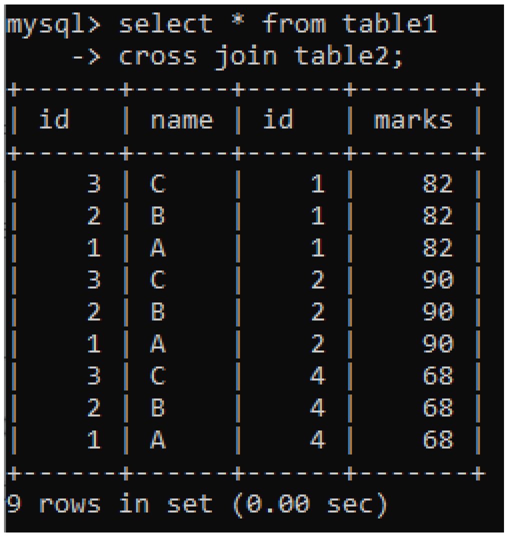
Command : SELECT column_list
FROM Table1
RIGHT [OUTER] JOIN Table2
ON join_condition;



```
mysql> select * from table1
    -> right join table2
    -> on table1.id = table2.id;
+------+------+------+-------+
| id   | name | id   | marks |
+------+------+------+-------+
|    1 | A    |    1 |    82 |
|    2 | B    |    2 |    90 |
| NULL | NULL |    4 |    68 |
+------+------+------+-------+
3 rows in set (0.00 sec)
```

Cross Join

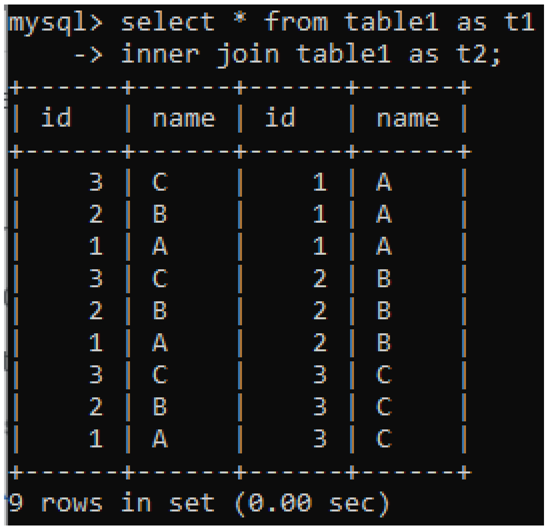
Command : SELECT column-lists
FROM table1
CROSS JOIN table2;



```
mysql> select * from table1
    -> cross join table2;
+------+------+------+-------+
| id   | name | id   | marks |
+------+------+------+-------+
|    3 | C    |    1 |    82 |
|    2 | B    |    1 |    82 |
|    1 | A    |    1 |    82 |
|    3 | C    |    2 |    90 |
|    2 | B    |    2 |    90 |
|    1 | A    |    2 |    90 |
|    3 | C    |    4 |    68 |
|    2 | B    |    4 |    68 |
|    1 | A    |    4 |    68 |
+------+------+------+-------+
9 rows in set (0.00 sec)
```

Self Join

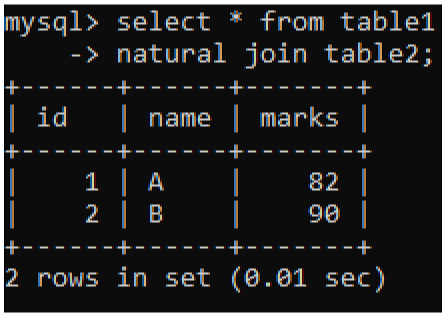
Command : Select … FROM student AS S1
INNER JOIN student AS S2;



```
mysql> select * from table1 as t1
    -> inner join table1 as t2;
+------+------+------+------+
| id   | name | id   | name |
+------+------+------+------+
|    3 | C    |    1 | A    |
|    2 | B    |    1 | A    |
|    1 | A    |    1 | A    |
|    3 | C    |    2 | B    |
|    2 | B    |    2 | B    |
|    1 | A    |    2 | B    |
|    3 | C    |    3 | C    |
|    2 | B    |    3 | C    |
|    1 | A    |    3 | C    |
+------+------+------+------+
9 rows in set (0.00 sec)
```

Natural Join

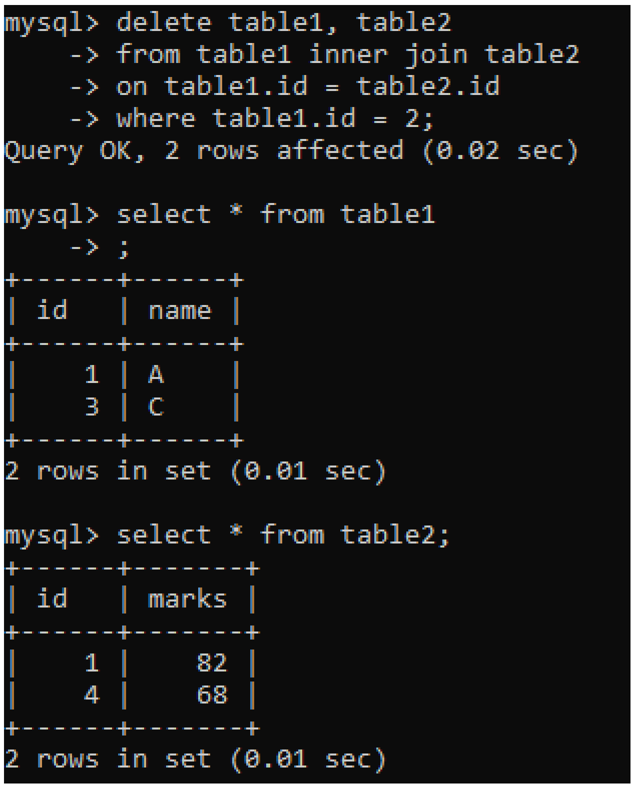
Command : SELECT [column_names | *]
FROM table_name1
NATURAL JOIN table_name2;



```
mysql> select * from table1
    -> natural join table2;
+------+------+-------+
| id   | name | marks |
+------+------+-------+
|    1 | A    |    82 |
|    2 | B    |    90 |
+------+------+-------+
2 rows in set (0.01 sec)
```

Delete Join

Command : DELETE target table
FROM   table1
<INNER | LEFT> JOIN table2
ON table1.joining_column= table2.joining_column
WHERE condition;

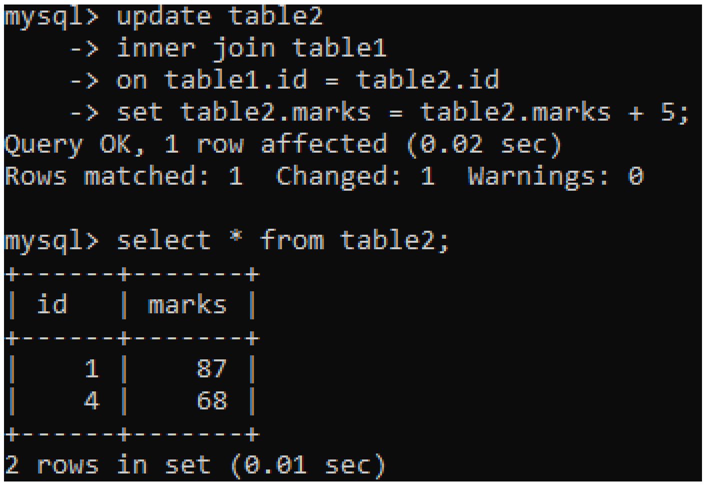


```
mysql> delete table1, table2
    -> from table1 inner join table2
    -> on table1.id = table2.id
    -> where table1.id = 2;
Query OK, 2 rows affected (0.02 sec)

mysql> select * from table1
    -> ;
+------+------+
| id   | name |
+------+------+
|    1 | A    |
|    3 | C    |
+------+------+
2 rows in set (0.01 sec)

mysql> select * from table2;
+------+-------+
| id   | marks |
+------+-------+
|    1 |    82 |
|    4 |    68 |
+------+-------+
2 rows in set (0.01 sec)
```

Update Join

Command : UPDATE Tab1, Tab2
[INNER JOIN | LEFT JOIN] Tab1

ON Tab1.C1 = Tab2.C1
SET update_expressions
WHERE Condition;



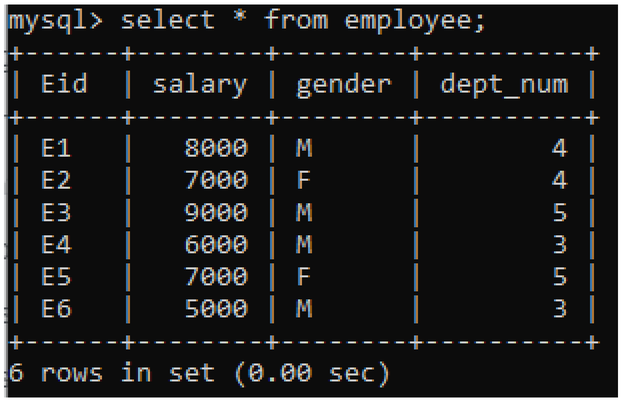
```
mysql> update table2
    -> inner join table1
    -> on table1.id = table2.id
    -> set table2.marks = table2.marks + 5;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from table2;
+------+-------+
| id   | marks |
+------+-------+
|    1 |    87 |
|    4 |    68 |
+------+-------+
2 rows in set (0.01 sec)
```

Triggers

It is a special type of stored procedure that is invoked automatically in response to an event. Each trigger is associated with a table, which is activated on any DML statement such as INSERT, UPDATE, or DELETE

Basic Command : (BEFOR | AFTER) table_name (INSERT | UPDATE | DELETE)

Employee Table



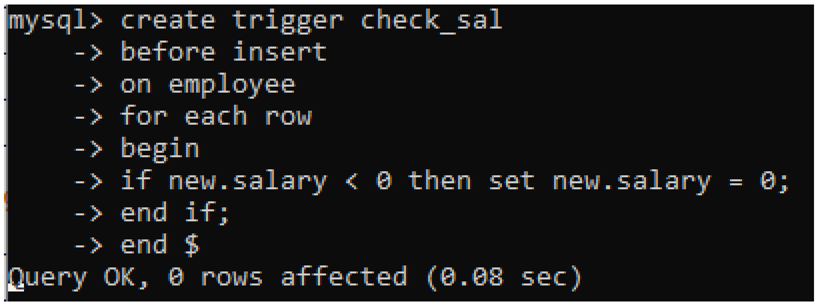
```
mysql> select * from employee;
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E1   |   8000 | M      |        4 |
| E2   |   7000 | F      |        4 |
| E3   |   9000 | M      |        5 |
| E4   |   6000 | M      |        3 |
| E5   |   7000 | F      |        5 |
| E6   |   5000 | M      |        3 |
+------+--------+--------+----------+
6 rows in set (0.00 sec)
```

Create Trigger

Command : DELIMITER <symbol like $$, $, && etc.>

Command: CREATE TRIGGER trigger_name trigger_time trigger_event
ON table_name FOR EACH ROW
BEGIN
    --variable declarations ending with ;
    --trigger code ending ending with ;
END <same_delimiter_symbol>

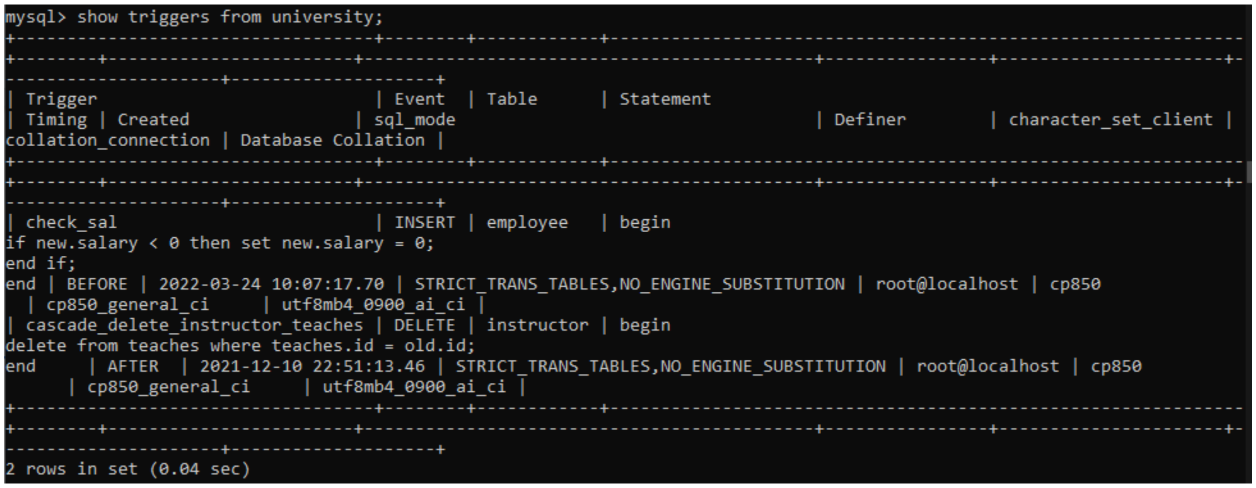
After the above code is executed, enter the below mentioned code to revert back the delimiter symbol to ;
Command : DELIMITER ;



```
mysql> create trigger check_sal
    -> before insert
    -> on employee
    -> for each row
    -> begin
    -> if new.salary < 0 then set new.salary = 0;
    -> end if;
    -> end $
Query OK, 0 rows affected (0.08 sec)
```

Show Trigger

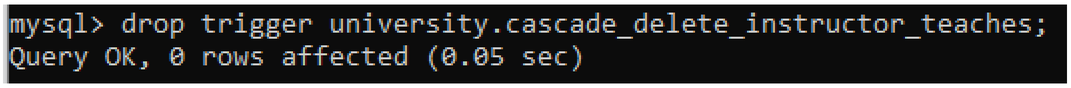
Command : SHOW TRIGGERS [FROM <database_name>] [LIKE <pattern>] [WHERE <search_conditions>];



```
mysql> show triggers from university;
+------------------------------------+--------+-----------+----------------------------------------------------------------
+--------+-------------------------+-------------------------------------------+-----------------+---------------------+-
--------------------+-------------------+
| Trigger                            | Event  | Table     | Statement
| Timing | Created                 | sql_mode                                  | Definer         | character_set_client |
collation_connection | Database Collation |
+------------------------------------+--------+-----------+----------------------------------------------------------------
+--------+-------------------------+-------------------------------------------+-----------------+---------------------+-
--------------------+-------------------+
| check_sal                          | INSERT | employee  | begin
if new.salary < 0 then set new.salary = 0;
end if;
end | BEFORE | 2022-03-24 10:07:17.70 | STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION | root@localhost | cp850
  | cp850_general_ci     | utf8mb4_0900_ai_ci |
| cascade_delete_instructor_teaches | DELETE | instructor | begin
delete from teaches where teaches.id = old.id;
end     | AFTER  | 2021-12-10 22:51:13.46 | STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION | root@localhost | cp850
      | cp850_general_ci     | utf8mb4_0900_ai_ci |
+------------------------------------+--------+-----------+----------------------------------------------------------------
+--------+-------------------------+-------------------------------------------+-----------------+---------------------+-
--------------------+-------------------+
2 rows in set (0.04 sec)
```

Drop Trigger

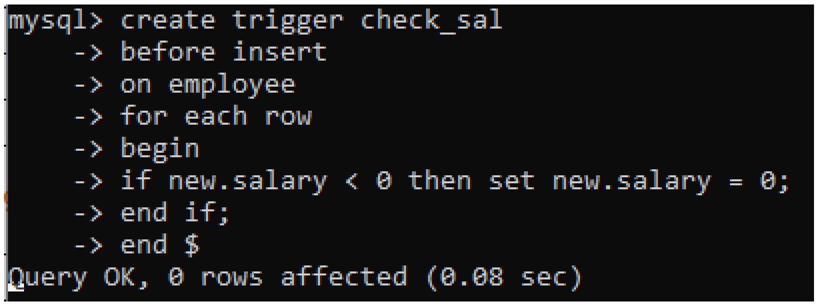
Command : DROP TRIGGER [IF EXISTS] [databse_name.]trigger_name;

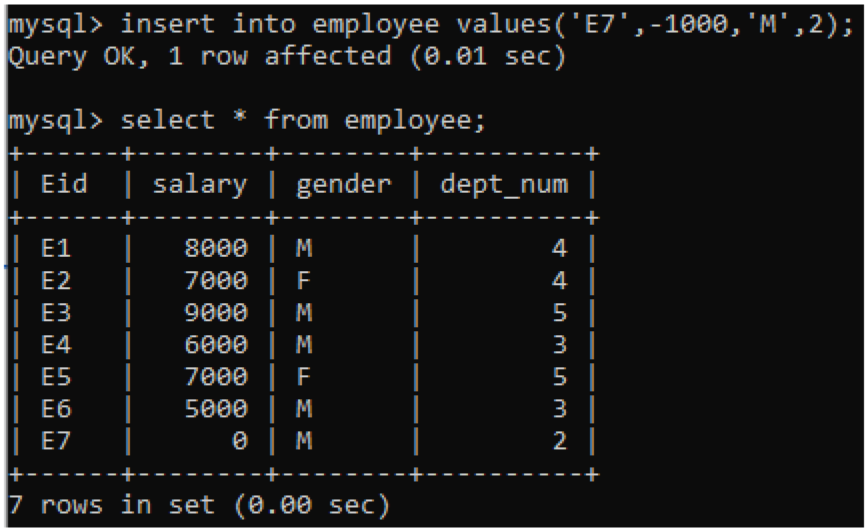

```
mysql> drop trigger university.cascade_delete_instructor_teaches;
Query OK, 0 rows affected (0.05 sec)
```

Insert Trigger

Before Insert

Command : CREATE TRIGGER trigger_name
BEFORE INSERT
ON table_name FOR EACH ROW
Trigger_body <end_symbol>


```
mysql> create trigger check_sal
    -> before insert
    -> on employee
    -> for each row
    -> begin
    -> if new.salary < 0 then set new.salary = 0;
    -> end if;
    -> end $
Query OK, 0 rows affected (0.08 sec)
```

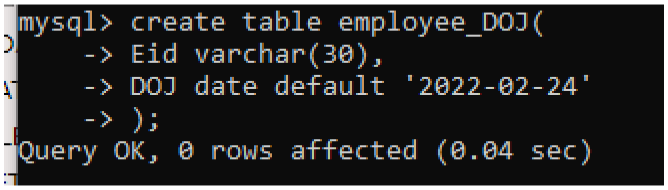

```
mysql> insert into employee values('E7',-1000,'M',2);
Query OK, 1 row affected (0.01 sec)

mysql> select * from employee;
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E1   |   8000 | M      |        4 |
| E2   |   7000 | F      |        4 |
| E3   |   9000 | M      |        5 |
| E4   |   6000 | M      |        3 |
| E5   |   7000 | F      |        5 |
| E6   |   5000 | M      |        3 |
| E7   |      0 | M      |        2 |
+------+--------+--------+----------+
7 rows in set (0.00 sec)
```

After Insert

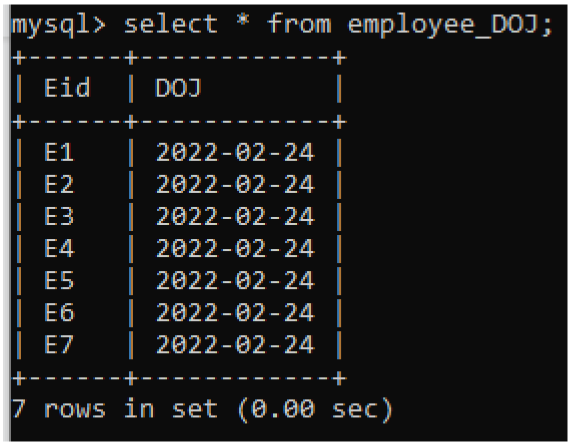
Command : CREATE TRIGGER trigger_name
AFTER INSERT
ON table_name FOR EACH ROW

trigger_body <end_symbol>

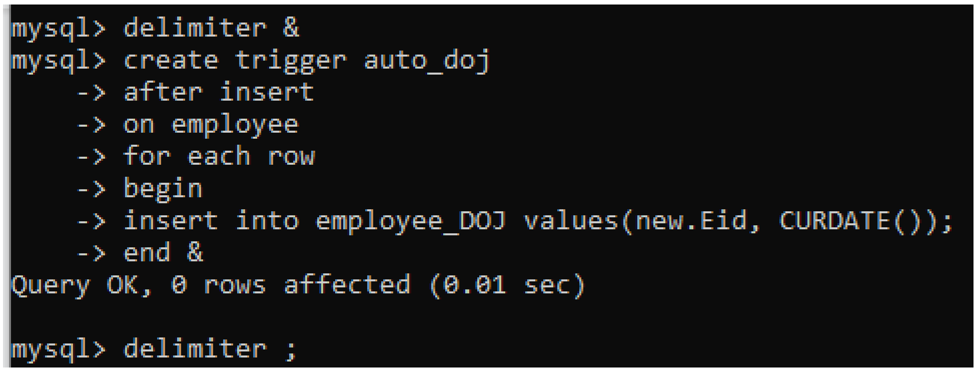
First we create a new table, employee_DOJ



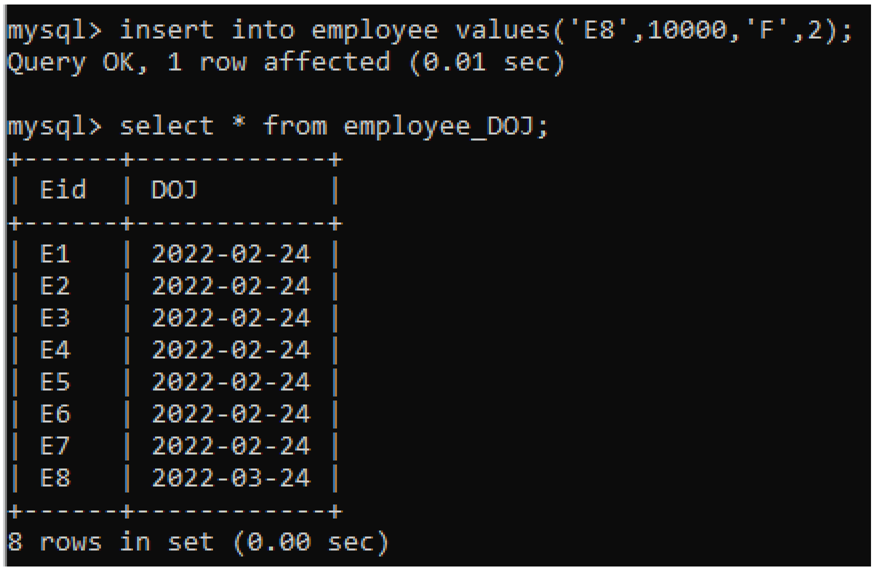
```
mysql> create table employee_DOJ(
    -> Eid varchar(30),
    -> DOJ date default '2022-02-24'
    -> );
Query OK, 0 rows affected (0.04 sec)
```



```
mysql> select * from employee_DOJ;
+------+------------+
| Eid  | DOJ        |
+------+------------+
| E1   | 2022-02-24 |
| E2   | 2022-02-24 |
| E3   | 2022-02-24 |
| E4   | 2022-02-24 |
| E5   | 2022-02-24 |
| E6   | 2022-02-24 |
| E7   | 2022-02-24 |
+------+------------+
7 rows in set (0.00 sec)
```



```
mysql> delimiter &
mysql> create trigger auto_doj
    -> after insert
    -> on employee
    -> for each row
    -> begin
    -> insert into employee_DOJ values(new.Eid, CURDATE());
    -> end &
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter ;
```

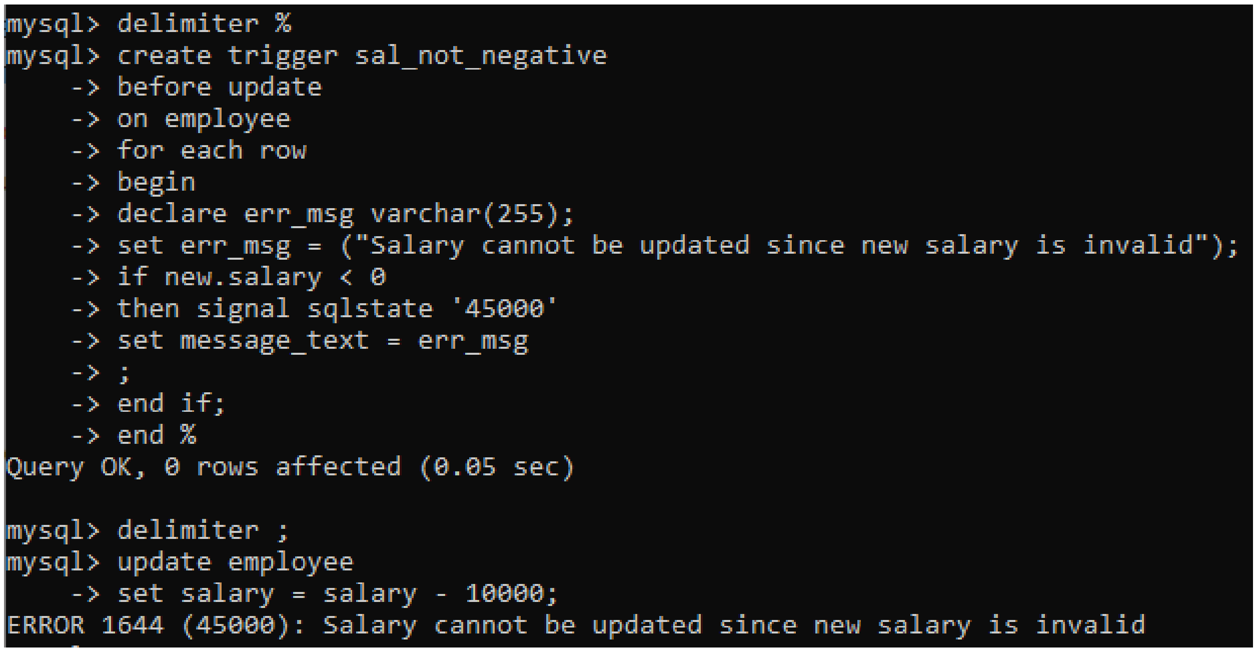


```
mysql> insert into employee values('E8',10000,'F',2);
Query OK, 1 row affected (0.01 sec)

mysql> select * from employee_DOJ;
+------+------------+
| Eid  | DOJ        |
+------+------------+
| E1   | 2022-02-24 |
| E2   | 2022-02-24 |
| E3   | 2022-02-24 |
| E4   | 2022-02-24 |
| E5   | 2022-02-24 |
| E6   | 2022-02-24 |
| E7   | 2022-02-24 |
| E8   | 2022-03-24 |
+------+------------+
8 rows in set (0.00 sec)
```

Update Trigger

Before Update

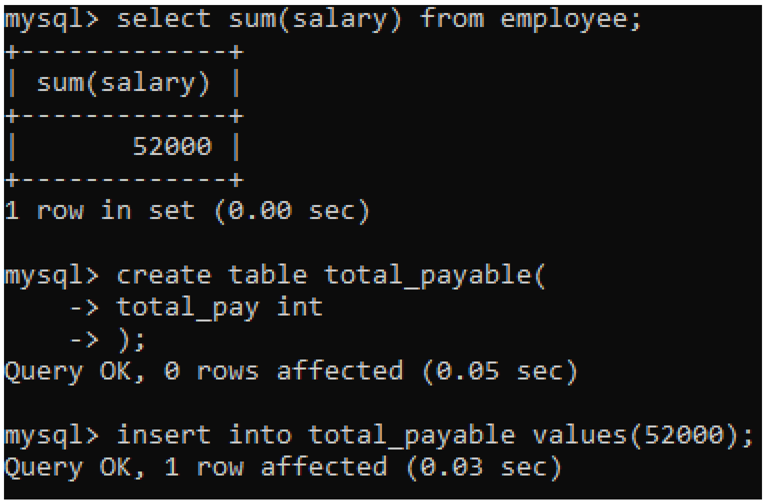
Command : CREATE TRIGGER trigger_name
BEFORE UPDATE
ON table_name FOR EACH ROW
trigger_body ;



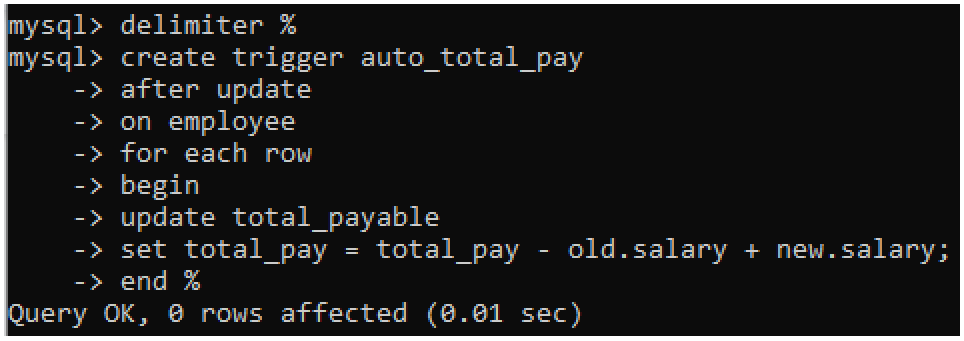
```
mysql> delimiter %
mysql> create trigger sal_not_negative
    -> before update
    -> on employee
    -> for each row
    -> begin
    -> declare err_msg varchar(255);
    -> set err_msg = ("Salary cannot be updated since new salary is invalid");
    -> if new.salary < 0
    -> then signal sqlstate '45000'
    -> set message_text = err_msg
    -> ;
    -> end if;
    -> end %
Query OK, 0 rows affected (0.05 sec)

mysql> delimiter ;
mysql> update employee
    -> set salary = salary - 10000;
ERROR 1644 (45000): Salary cannot be updated since new salary is invalid
```

After Update

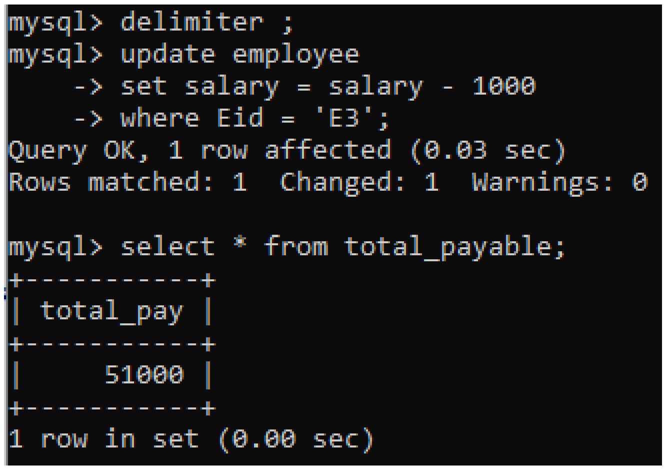
Command : CREATE TRIGGER trigger_name
AFTER UPDATE
ON table_name FOR EACH ROW
trigger_body ;



```
mysql> select sum(salary) from employee;
+-------------+
| sum(salary) |
+-------------+
|       52000 |
+-------------+
1 row in set (0.00 sec)

mysql> create table total_payable(
    -> total_pay int
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> insert into total_payable values(52000);
Query OK, 1 row affected (0.03 sec)
```



```
mysql> delimiter %
mysql> create trigger auto_total_pay
    -> after update
    -> on employee
    -> for each row
    -> begin
    -> update total_payable
    -> set total_pay = total_pay - old.salary + new.salary;
    -> end %
Query OK, 0 rows affected (0.01 sec)
```

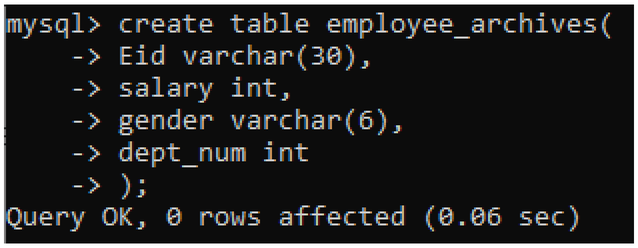


```
mysql> delimiter ;
mysql> update employee
    -> set salary = salary - 1000
    -> where Eid = 'E3';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from total_payable;
+-----------+
| total_pay |
+-----------+
|     51000 |
+-----------+
1 row in set (0.00 sec)
```

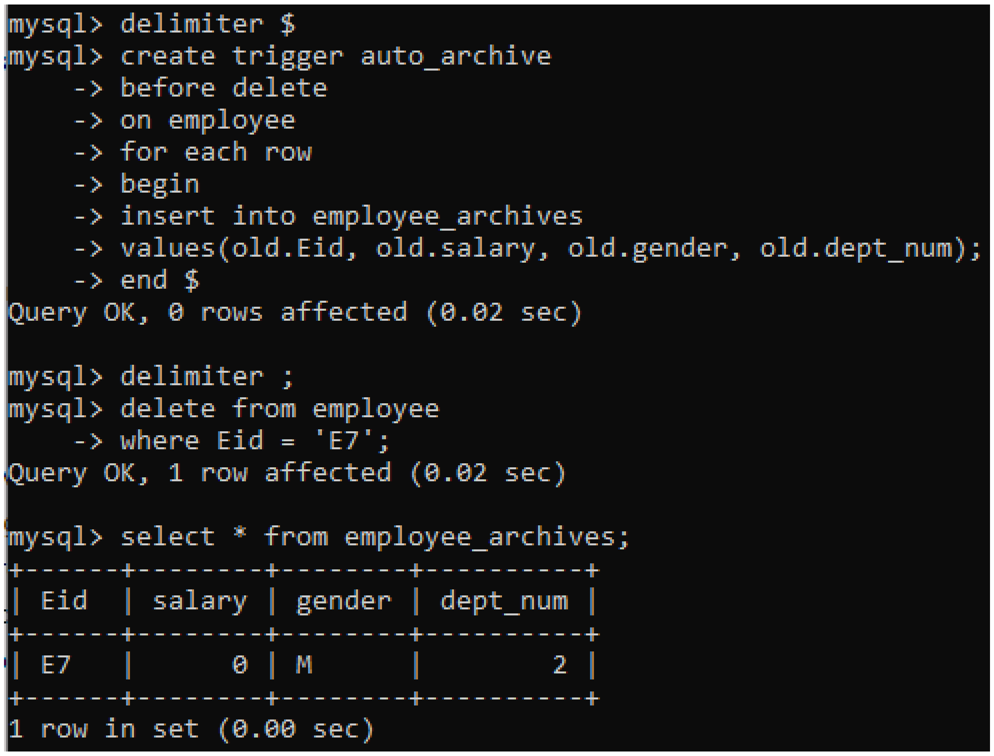
Delete Trigger

Before Delete

Command : CREATE TRIGGER trigger_name
BEFORE DELETE
ON table_name FOR EACH ROW
Trigger_body ;



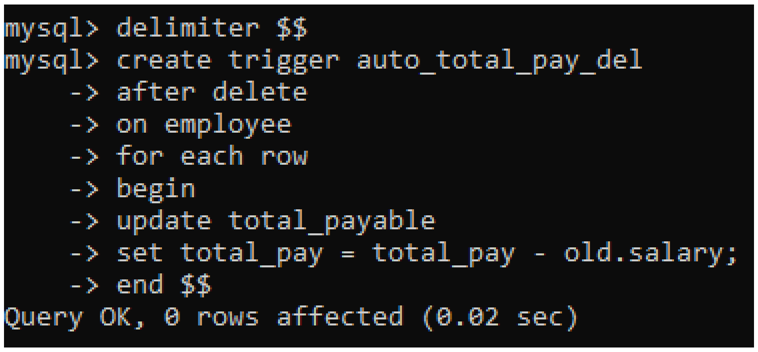
```
mysql> create table employee_archives(
    -> Eid varchar(30),
    -> salary int,
    -> gender varchar(6),
    -> dept_num int
    -> );
Query OK, 0 rows affected (0.06 sec)
```



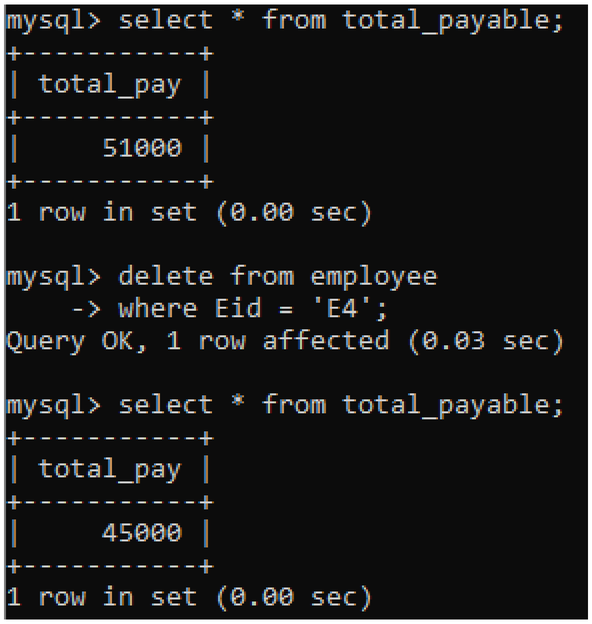
```
mysql> delimiter $
mysql> create trigger auto_archive
    -> before delete
    -> on employee
    -> for each row
    -> begin
    -> insert into employee_archives
    -> values(old.Eid, old.salary, old.gender, old.dept_num);
    -> end $
Query OK, 0 rows affected (0.02 sec)

mysql> delimiter ;
mysql> delete from employee
    -> where Eid = 'E7';
Query OK, 1 row affected (0.02 sec)

mysql> select * from employee_archives;
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E7   |      0 | M      |        2 |
+------+--------+--------+----------+
1 row in set (0.00 sec)
```

After Delete

Command : CREATE TRIGGER trigger_name
AFTER DELETE
ON table_name FOR EACH ROW
Trigger_body ;



```
mysql> delimiter $$
mysql> create trigger auto_total_pay_del
    -> after delete
    -> on employee
    -> for each row
    -> begin
    -> update total_payable
    -> set total_pay = total_pay - old.salary;
    -> end $$
Query OK, 0 rows affected (0.02 sec)
```

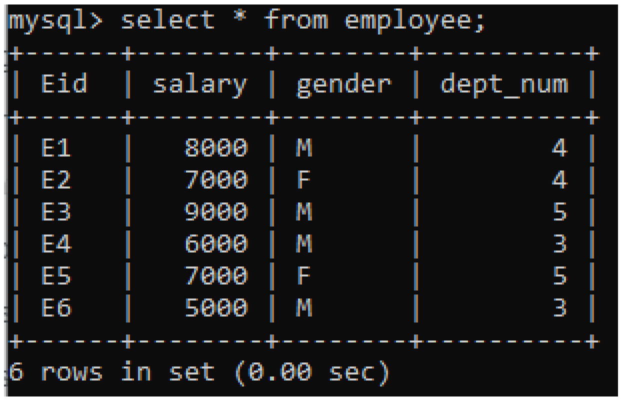


```
mysql> select * from total_payable;
+-----------+
| total_pay |
+-----------+
|     51000 |
+-----------+
1 row in set (0.00 sec)

mysql> delete from employee
    -> where Eid = 'E4';
Query OK, 1 row affected (0.03 sec)

mysql> select * from total_payable;
+-----------+
| total_pay |
+-----------+
|     45000 |
+-----------+
1 row in set (0.00 sec)
```

Aggregate Functions

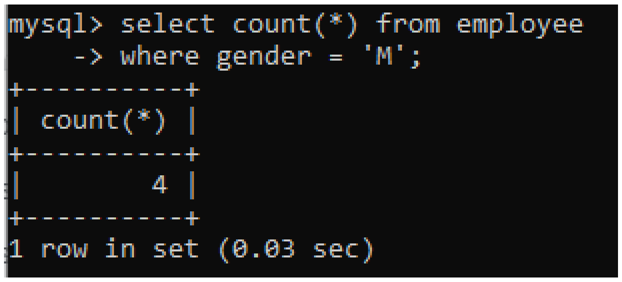
MySQL's aggregate function is used to perform calculations on multiple values and return the result in a single value like the average of all values, the sum of all values, and maximum & minimum value among certain groups of values.

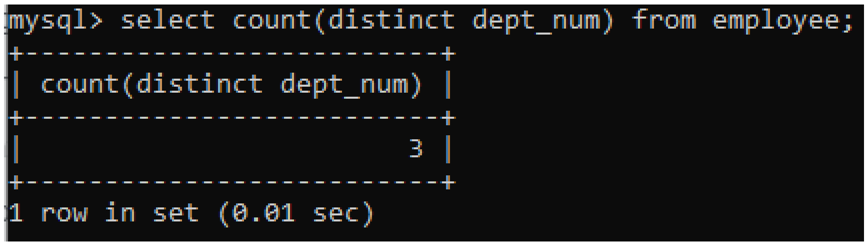
Employee Table


```
mysql> select * from employee;
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E1   |   8000 | M      |        4 |
| E2   |   7000 | F      |        4 |
| E3   |   9000 | M      |        5 |
| E4   |   6000 | M      |        3 |
| E5   |   7000 | F      |        5 |
| E6   |   5000 | M      |        3 |
+------+--------+--------+----------+
6 rows in set (0.00 sec)
```

Count

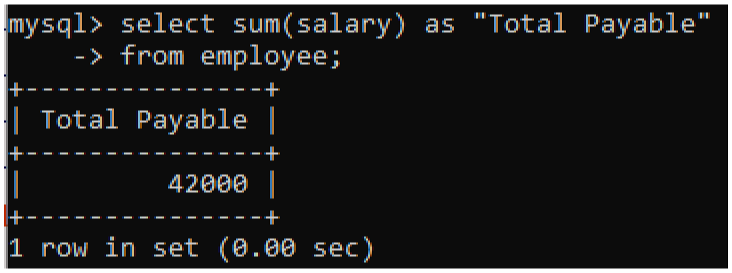
Command : SELECT COUNT (aggregate_expression)
FROM table_name
[WHERE conditions];


```
mysql> select count(*) from employee
    -> where gender = 'M';
+----------+
| count(*) |
+----------+
|        4 |
+----------+
1 row in set (0.03 sec)
```

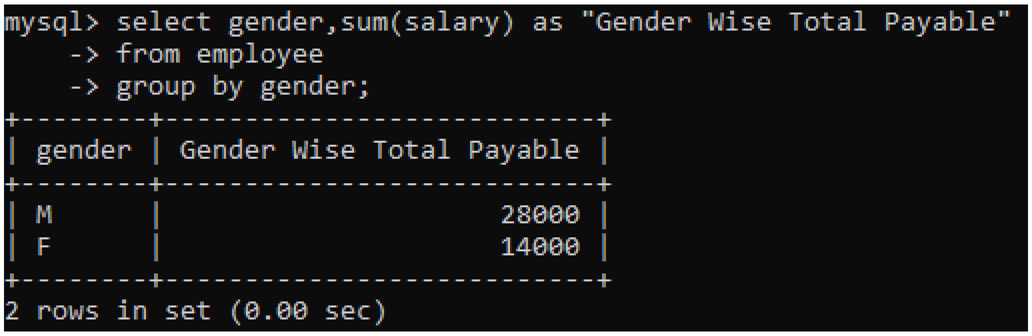

```
mysql> select count(distinct dept_num) from employee;
+--------------------------+
| count(distinct dept_num) |
+--------------------------+
|                        3 |
+--------------------------+
1 row in set (0.01 sec)
```

Sum

Command : SELECT SUM(aggregate_expression)
FROM tables
[WHERE conditions];



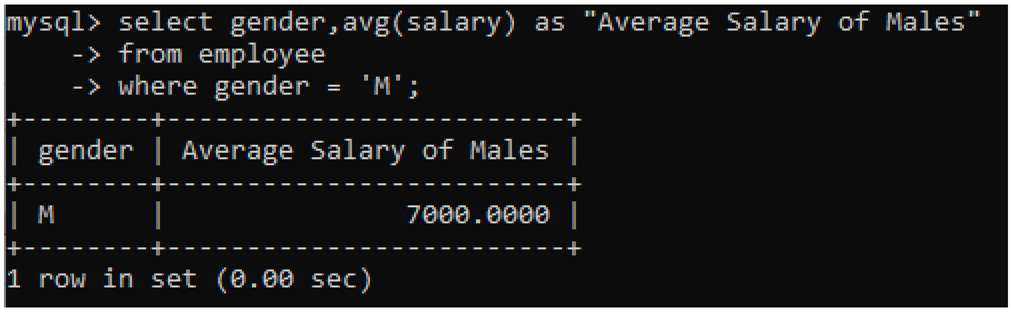
```
mysql> select sum(salary) as "Total Payable"
    -> from employee;
+---------------+
| Total Payable |
+---------------+
|         42000 |
+---------------+
1 row in set (0.00 sec)
```



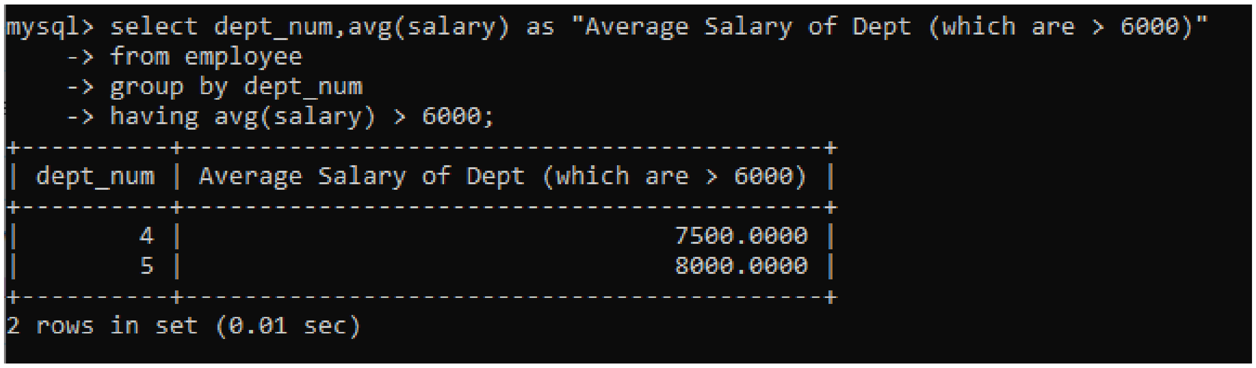
```
mysql> select gender,sum(salary) as "Gender Wise Total Payable"
    -> from employee
    -> group by gender;
+--------+---------------------------+
| gender | Gender Wise Total Payable |
+--------+---------------------------+
| M      |                     28000 |
| F      |                     14000 |
+--------+---------------------------+
2 rows in set (0.00 sec)
```

Avg

Command : SELECT AVG(aggregate_expression)
FROM tables
[WHERE conditions];



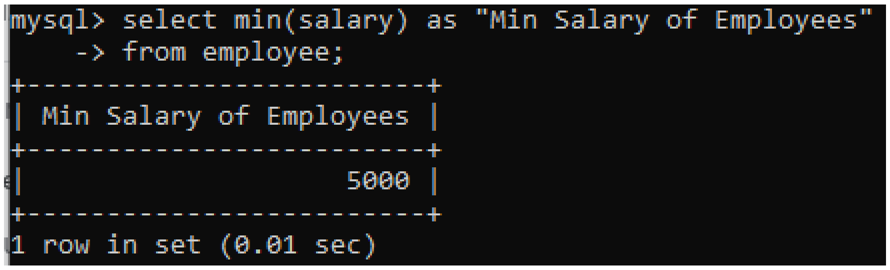
```
mysql> select gender,avg(salary) as "Average Salary of Males"
    -> from employee
    -> where gender = 'M';
+--------+-------------------------+
| gender | Average Salary of Males |
+--------+-------------------------+
| M      |               7000.0000 |
+--------+-------------------------+
1 row in set (0.00 sec)
```



```
mysql> select dept_num,avg(salary) as "Average Salary of Dept (which are > 6000)"
    -> from employee
    -> group by dept_num
    -> having avg(salary) > 6000;
+----------+-------------------------------------------+
| dept_num | Average Salary of Dept (which are > 6000) |
+----------+-------------------------------------------+
|        4 |                                 7500.0000 |
|        5 |                                 8000.0000 |
+----------+-------------------------------------------+
2 rows in set (0.01 sec)
```

Min

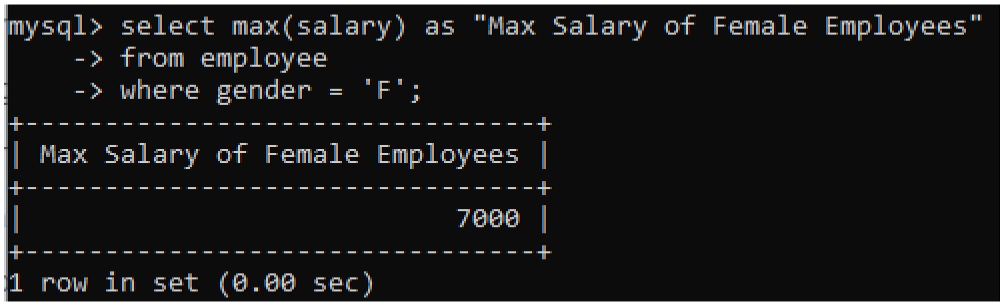
Command : SELECT MIN (aggregate_expression)
FROM table_name(s)
[WHERE conditions];



```
mysql> select min(salary) as "Min Salary of Employees"
    -> from employee;
+-------------------------+
| Min Salary of Employees |
+-------------------------+
|                    5000 |
+-------------------------+
1 row in set (0.01 sec)
```

Max

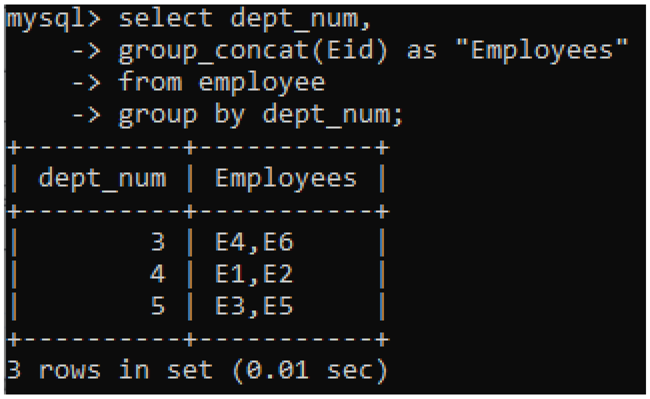
Command : SELECT MAX (aggregate_expression)
FROM table_name(s)
[WHERE conditions];



```
mysql> select max(salary) as "Max Salary of Female Employees"
    -> from employee
    -> where gender = 'F';
+--------------------------------+
| Max Salary of Female Employees |
+--------------------------------+
|                           7000 |
+--------------------------------+
1 row in set (0.00 sec)
```

Group_concat

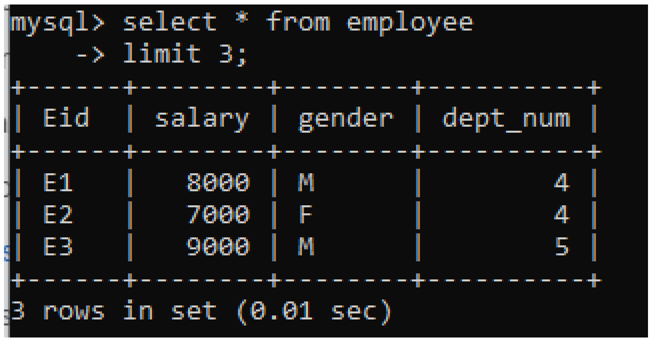
Command : SELECT c1, c2, ..., cN
GROUP_CONCAT (
  [DISTINCT] c_name1
  [ORDER BY]
  [SEPARATOR])
FROM table_name GROUP BY c_name2;



```
mysql> select dept_num,
    -> group_concat(Eid) as "Employees"
    -> from employee
    -> group by dept_num;
+----------+-----------+
| dept_num | Employees |
+----------+-----------+
|        3 | E4,E6     |
|        4 | E1,E2     |
|        5 | E3,E5     |
+----------+-----------+
3 rows in set (0.01 sec)
```

First

Command : SELECT column_name
FROM table_name
LIMIT <first_n_rows>;



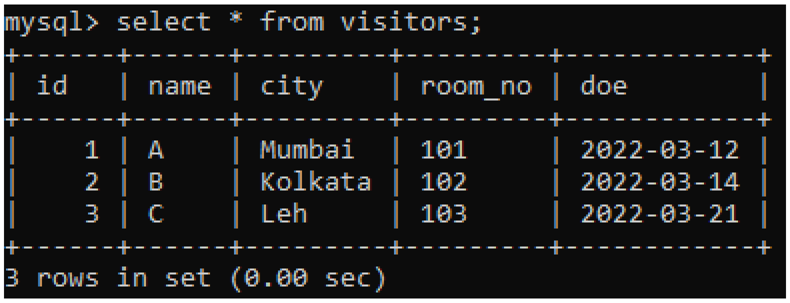
```
mysql> select * from employee
    -> limit 3;
+------+--------+--------+----------+
| Eid  | salary | gender | dept_num |
+------+--------+--------+----------+
| E1   |   8000 | M      |        4 |
| E2   |   7000 | F      |        4 |
| E3   |   9000 | M      |        5 |
+------+--------+--------+----------+
3 rows in set (0.01 sec)
```

Set Operations

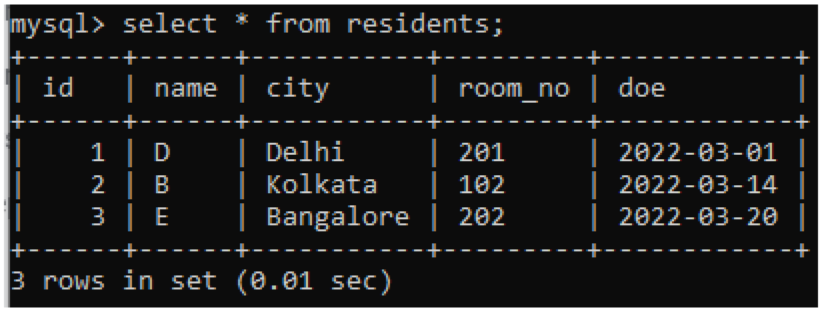
The SET Operators in MySQL are basically used to combine the result of more than 1 select statement and return the output as a single result set. In SQL, there are 2 types of set operators. They are as follows:

1. UNION: It is used to combine two or more result sets into a single set, without duplicates.
2. UNION ALL: It is used to combine two or more result sets into a single set, including duplicates.

Visitors Table:

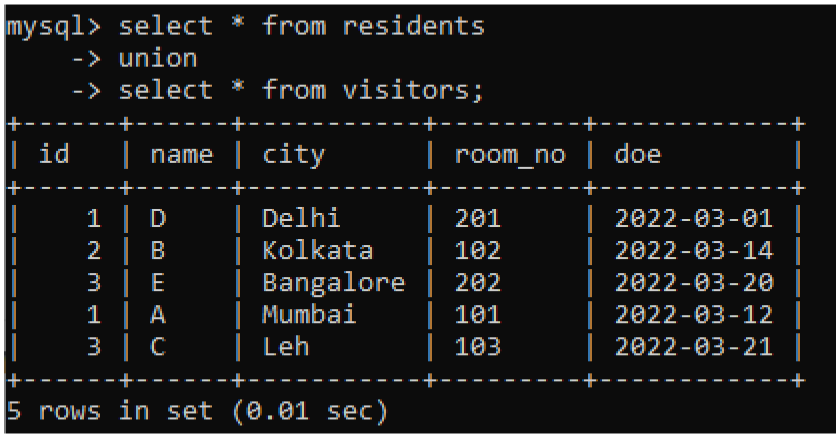

```
mysql> select * from visitors;
+------+------+---------+---------+------------+
| id   | name | city    | room_no | doe        |
+------+------+---------+---------+------------+
|    1 | A    | Mumbai  | 101     | 2022-03-12 |
|    2 | B    | Kolkata | 102     | 2022-03-14 |
|    3 | C    | Leh     | 103     | 2022-03-21 |
+------+------+---------+---------+------------+
3 rows in set (0.00 sec)
```

Residents Table:


```
mysql> select * from residents;
+------+------+-----------+---------+------------+
| id   | name | city      | room_no | doe        |
+------+------+-----------+---------+------------+
|    1 | D    | Delhi     | 201     | 2022-03-01 |
|    2 | B    | Kolkata   | 102     | 2022-03-14 |
|    3 | E    | Bangalore | 202     | 2022-03-20 |
+------+------+-----------+---------+------------+
3 rows in set (0.01 sec)
```

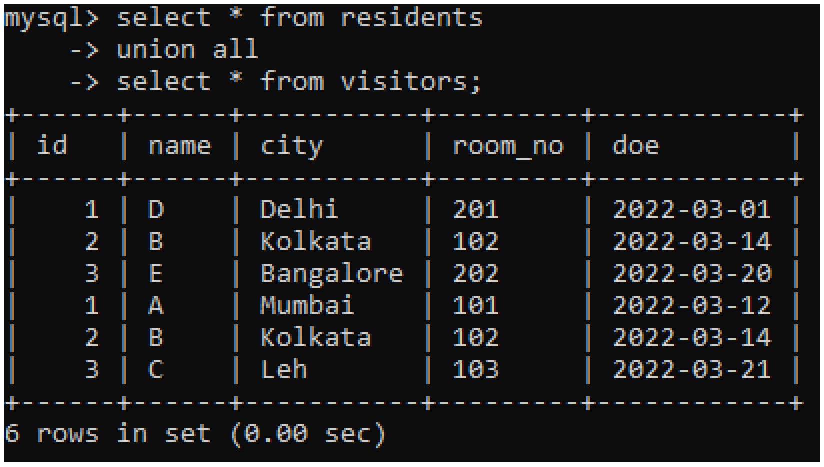
Union

Command : SELECT statement_1
UNION
SELECT statement_2;


```
mysql> select * from residents
    -> union
    -> select * from visitors;
+------+------+-----------+---------+------------+
| id   | name | city      | room_no | doe        |
+------+------+-----------+---------+------------+
|    1 | D    | Delhi     | 201     | 2022-03-01 |
|    2 | B    | Kolkata   | 102     | 2022-03-14 |
|    3 | E    | Bangalore | 202     | 2022-03-20 |
|    1 | A    | Mumbai    | 101     | 2022-03-12 |
|    3 | C    | Leh       | 103     | 2022-03-21 |
+------+------+-----------+---------+------------+
5 rows in set (0.01 sec)
```

Union All

Command : SELECT statement_1
UNION ALL
SELECT statement_2;



```
mysql> select * from residents
    -> union all
    -> select * from visitors;
+------+------+-----------+---------+------------+
| id   | name | city      | room_no | doe        |
+------+------+-----------+---------+------------+
|    1 | D    | Delhi     | 201     | 2022-03-01 |
|    2 | B    | Kolkata   | 102     | 2022-03-14 |
|    3 | E    | Bangalore | 202     | 2022-03-20 |
|    1 | A    | Mumbai    | 101     | 2022-03-12 |
|    2 | B    | Kolkata   | 102     | 2022-03-14 |
|    3 | C    | Leh       | 103     | 2022-03-21 |
+------+------+-----------+---------+------------+
6 rows in set (0.00 sec)
```

Data Handling

The prerequisite to use this is to have a .sql file which can be worked upon. To generate .sql file data in case you do not have it available with you, please visit this site - https://www.mockaroo.com/.

In case data is csv format, use https://www.convertcsv.com/csv-to-sql.htm to convert data into .sql format.

Import from .sql

First we have to create a table with the exact same description as that of the columns in the csv file.

```
mysql> create table student_info(
    -> id int Primary Key,
    -> first_name varchar(50),
    -> last_name varchar(50),
    -> email varchar(50),
    -> gender varchar(50),
    -> dev_marks decimal(5,2),
    -> ai_marks decimal(5,2),
    -> cyber_marks decimal(5,2),
    -> cp_marks decimal(5,2)
    -> );
Query OK, 0 rows affected (0.29 sec)
```

Once the table is created we have to put in this command: source absolute_path_of_sql_file;

```
mysql> source C:/Users/atten/Downloads/student_info.sql;
Query OK, 0 rows affected (0.06 sec)

mysql> show tables;
+-----------------+
| Tables_in_trial |
+-----------------+
| emp_info        |
| student_info    |
+-----------------+
2 rows in set (0.01 sec)

mysql> desc student_info;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| id          | int          | YES  |     | NULL    |       |
| first_name  | varchar(50)  | YES  |     | NULL    |       |
| last_name   | varchar(50)  | YES  |     | NULL    |       |
| email       | varchar(50)  | YES  |     | NULL    |       |
| gender      |varchar(50)   | YES  |     | NULL    |       |
| dev_marks   | decimal(5,2) | YES  |     | NULL    |       |
| ai_marks    | decimal(5,2) | YES  |     | NULL    |       |
| cyber_marks | decimal(5,2) | YES  |     | NULL    |       |
| cp_marks    | decimal(5,2) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
9 rows in set (0.01 sec)

mysql> select * from student_info;
+------+----------------+-----------------+----------------------------------+------------+-----------+----------+---
----------+----------+
| id   | first_name     | last_name       | email                            | gender     | dev_marks |
ai_marks | cyber_marks | cp_marks |
+------+----------------+-----------------+----------------------------------+------------+-----------+----------+---
----------+----------+
|    1 | Giulietta      | Faulkener       | gfaulkener0@is.gd                | Female     |      2.42 |
67.95 |       99.53 |    85.34 |
|    2 | Joey           | Gilder          | jgilder1@hp.com                  | Male       |     34.50 |    98.60 |
91.03 |    31.68 |
|    3 | Helsa          | Hradsky         | hhradsky2@seesaa.net             | Male       |     54.88 |
3.93 |       45.40 |    75.21 |
|    4 | Hagan          | Normanell       | hnormanell3@berkeley.edu          | Male        |     66.22 |
53.40 |       15.13 |    31.71 |
.
.
.
.
.
```
The entire table is printed.

Export to .sql

Command in terminal : mysqldump -u my_username -p database_name > output_file_path

```
$ mysqldump -u root -p trial > ./Users/atten/OneDrive/Desktop/trial.sql
```

Python + SQL

Install and import connector

In case you use Python on your local system, then you have to install the mysql connector using the terminal.

Command - $ python -m pip install mysql-connector-python

In case you use online resources like Google Colab, you can directly move on to importing the library.

Command - import mysql.connector

Create connection

```
Command - mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword”,
  [database="mydatabase"]
)
```

Use connection

To use the connection, first we have to create a cursor using the following command

Python Statement : mycursor = mydb.cursor()

Then, to execute the actual mysql command we execute the following:

Python Statement : mycursor.execute(“<entire mysql command without ending semicolon”)

To make the changes final:

Python Statement : mydb.commit()

If the mysql command returns a result which can be displayed, after the mycursor.execute command, we run the following:

Python Statement : myresult = mycursor.fetchall()

Python Statement : for x in myresult: print(x)

To enter the sql command which is multiple lines:

```
sql = "Line 1 \
  Line 2 \
  Line 3 \
  Line 4"

mycursor.execute(sql)

myresult = mycursor.fetchall()

for x in myresult:
  print(x)
```

Also to prevent sql injection error complications and confusions, we should denote the strings in the sql command in single quotes (‘’) and those of python in double quotes (“”).

Exercises

Accounts Schema

1. Create a table Accounts
a. having (receipt_no, receipt_desc, payment_mode, amount, status)
b. With “receipt_no” being the primary key

c. Check constraint on amount => (amount > 1000)
d. Default constraint on payment_mode => (default value = “Cash”)

receipt_no	receipt_desc	payment_mode	amount	status
101	Tuition Fee	Cash	17000	Paid
102	Development Fee	Cash	5000	Due
103	Exam Fee	Cheque	1500	Paid
104	Tuition Fee	Online	24000	Due
105	Exam Fee	Cheque	1500	Paid

2. Perform following operations on Accounts
a. Update “Exam Fee” amounts to 2000 for all records in the table.
b. Set “status” = paid for receipt_no = 102
c. If payment_mode = “cash” then increase the amount by 1000

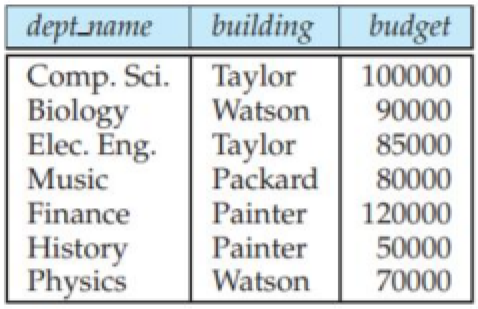
3. Perform following operations on Accounts
a. Delete record for receipt_no = 105
b. Delete all records where status = paid

University Schema

1. Create the following Database Schemas:

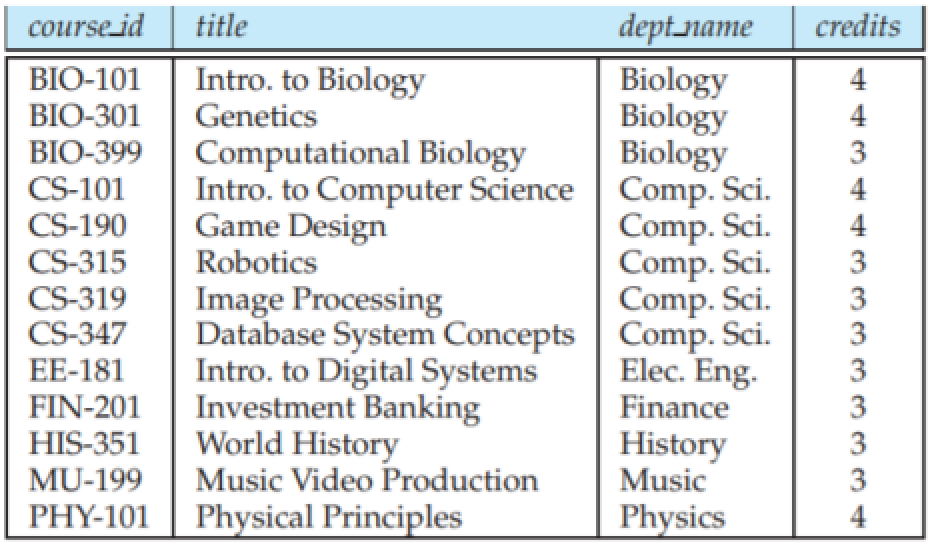
Department(Dept_name, building, budget)
Course(Course_ID, Title, Dept_name, Credits)
Instructor(ID, name, Dept_name, Salary)
Section(CourseID, SectionID, Semester, Year, Building, Room_no, Time_slot_ID)
Teachers(ID, Course_ID, Section_ID, Semester, Year)

2. Insert the following tuples into Department Table



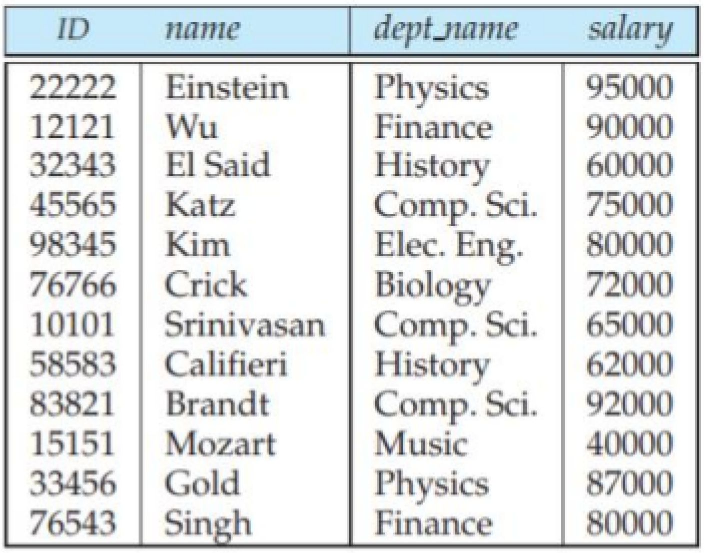
dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

3. Insert the following tuples into Course Table



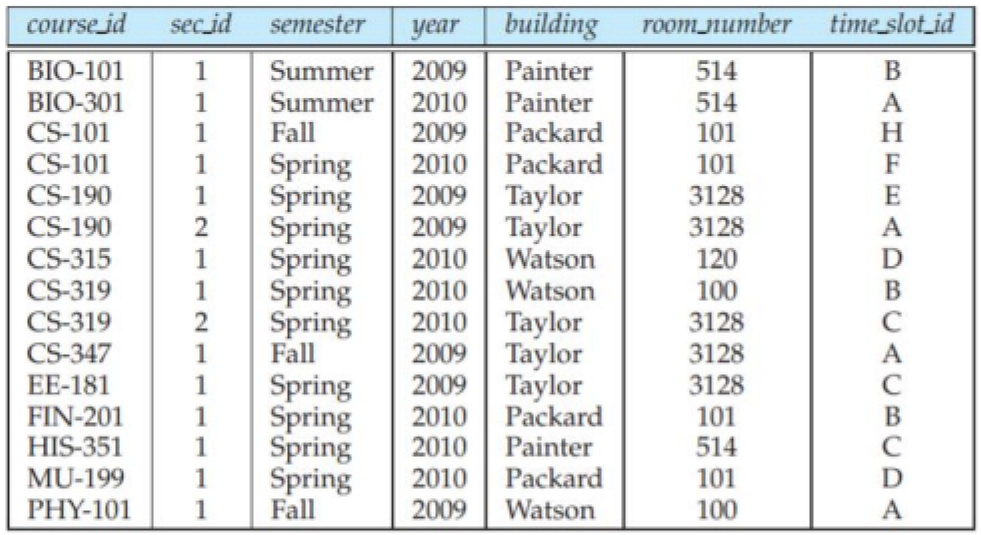
course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

4. Insert the following tuples into the Instructor table



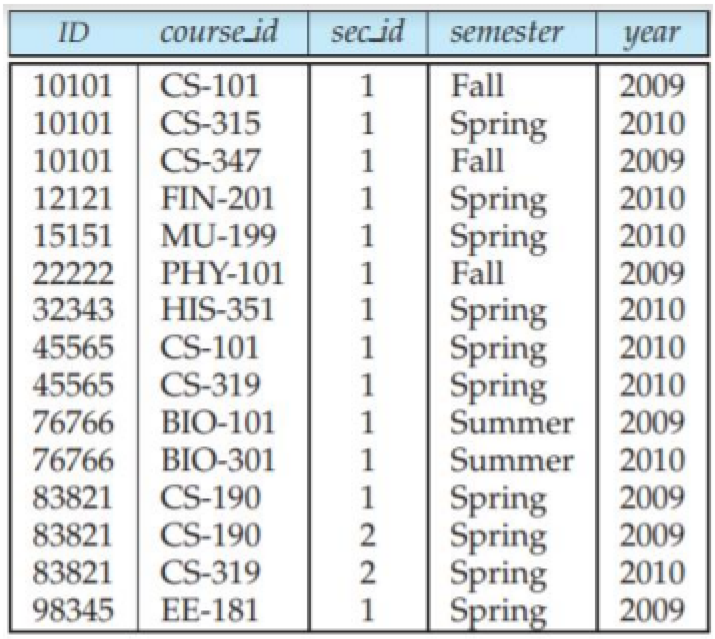
ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

5. Insert the following tuples into the Section table



course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2009	Painter	514	B
BIO-301	1	Summer	2010	Painter	514	A
CS-101	1	Fall	2009	Packard	101	H
CS-101	1	Spring	2010	Packard	101	F
CS-190	1	Spring	2009	Taylor	3128	E
CS-190	2	Spring	2009	Taylor	3128	A
CS-315	1	Spring	2010	Watson	120	D
CS-319	1	Spring	2010	Watson	100	B
CS-319	2	Spring	2010	Taylor	3128	C
CS-347	1	Fall	2009	Taylor	3128	A
EE-181	1	Spring	2009	Taylor	3128	C
FIN-201	1	Spring	2010	Packard	101	B
HIS-351	1	Spring	2010	Painter	514	C
MU-199	1	Spring	2010	Packard	101	D
PHY-101	1	Fall	2009	Watson	100	A

6. Insert the following tuples into the Teachers table



ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2009
10101	CS-315	1	Spring	2010
10101	CS-347	1	Fall	2009
12121	FIN-201	1	Spring	2010
15151	MU-199	1	Spring	2010
22222	PHY-101	1	Fall	2009
32343	HIS-351	1	Spring	2010
45565	CS-101	1	Spring	2010
45565	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	Summer	2010
83821	CS-190	1	Spring	2009
83821	CS-190	2	Spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

7. Using the Course Table
a) Select courses offered by the computer science department.
b) Select courses offered by the computer science department where credits=4.
c) Get courses which has a substring ‘ro’ in title
d) Get courses where title is made of 3 words

8. Using the Instructor Table

a) Select all course_id where room_num are between 100-500.
b) Select course_Id, room_number from Section table. Room Number should be in increasing order.
c) How many rows are there when you apply the like operator on building with ‘ant’ and building with ’son’.
d) Select all course_id which were offered in summer or spring of 2009
e) Select name of all Instructors in Comp. Sci. or Physics department
f) Select Distinct department from Instructor
g) Find the name of all Instructors in Comp. Sci. department with Salary greater than 70000
h) Rename salary to monthly_sal
i) Find department name, total salary corresponding to each Department.
j) List the number of distinct departments available in the instructor table
k) List the total salary payable to all the instructors.
l) Find the Instructor name whose salary is minimum.

9. In the University Schema
a) Make course_id as the primary key in the course table.
b) Make dept_name as the primary key in the department table.
c) Create a foreign key (dept_name) in the course table referencing department (dept_name).
d) Add CHECK constraint in course table with condition (credits > 0);

10. Using the Section Table
(a) Find the number of distinct buildings in the campus?
(b) Find the number of buildings where course_id starts with “cs”.
(c) Find the semester and year having the max number of courses.

11. Using the Department Table
(a) What is the average budget allotted to each department.
(b) How many departments have a higher budget than average ?
(c) How many departments have a lower budget than the average budget ?

12. In the University Schema, using join:
(a) Display course_name, room_number and semester of all the courses in university.
(b) Get name and year of courses taught in 2009.
(c) Display name of all instructors along with their course_id (null if no course assigned)
(d) Find the instructor's names who work in the Taylor building.
(e) Find all courses which are taught in the Spring Semester.
(f) Count the number of courses which are taught in ‘Biology’ Department and room number ‘514’.

13. In the University Schema,
(a) Find the set of all courses taught either in Fall 2009 or in Spring 2010, or both
(b) Find all the courses taught in the Fall 2009 semester but not in the Spring 2010

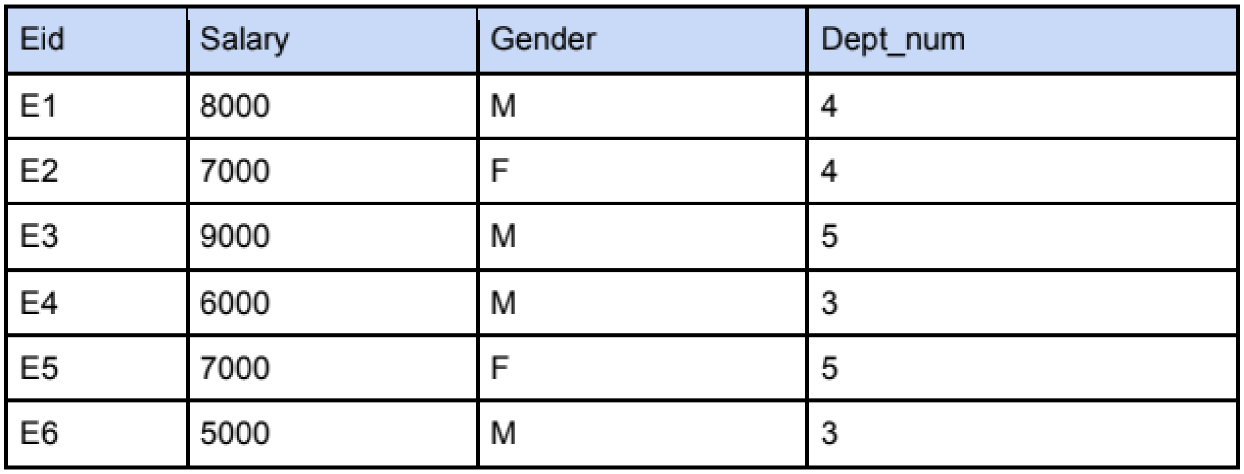
Semester
(c) Find the names of all instructors whose salary is greater than at least one instructor in the Physics department
(d) Find those departments for which the average salary is greater than or equal to all average salaries
(e) Display a list of all instructors, showing their ID, name, and the number of sections that they have taught. Make sure to show the number of sections as 0 for instructors who have not taught any section. Your query should use an outer join, and should not use scalar subqueries.

14. Create three views as “Spring”, “Fall”, “Summer” from the section table which contains “Course_ID, Sec_ID, Year, Building, Room_no” of the respective seasons.
(a) Show all table names before and after creating views.
(b) Add two new rows in each view and display all records before and after insertion.
(c) Delete the 1’st record from each view and display all records before and after the operation.
(d) Drop all 3 Created Views and show all Tables before and after the operation.

15. Create a view as “CSE” for “Comp.Sci.” Department which contains “Course_name, Credit, Teacher_ID, Semester, Year, Building, Room_no, Time_slot”.

16. Consider University Database
(a) Create a trigger on the instructor, which performs the on delete cascade on teachers table when a record in the instructor table is deleted.
(b) Create a trigger on the instructor, which deducts Rs. 200 from salary each time a record is inserted.
(c) Drop the first trigger

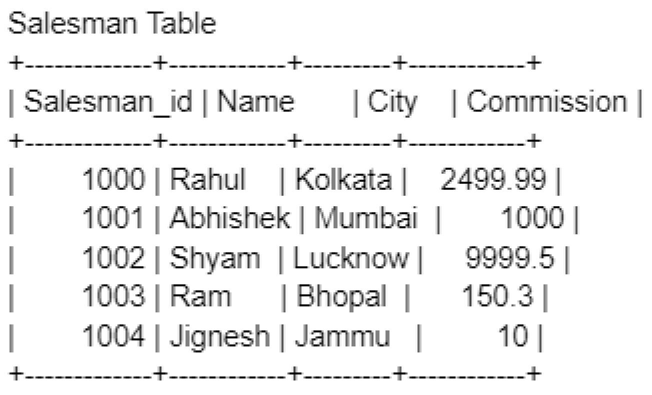
Employee Schema



Eid	Salary	Gender	Dept_num
E1	8000	M	4
E2	7000	F	4
E3	9000	M	5
E4	6000	M	3
E5	7000	F	5
E6	5000	M	3

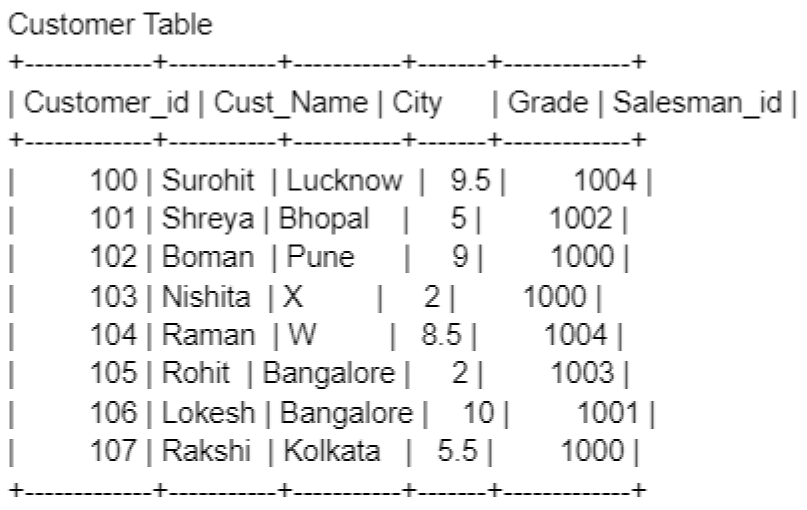
1. Using the above table,
(a) Retrieve the Eid of the person who gets the highest salary?
(b) Retrieve the Eid of the person who gets the maximum salary for each department?
(c) Retrieve Dept_num for which average salary of female employees of each department is more than the average salary of the male employee in them?

Company Schema



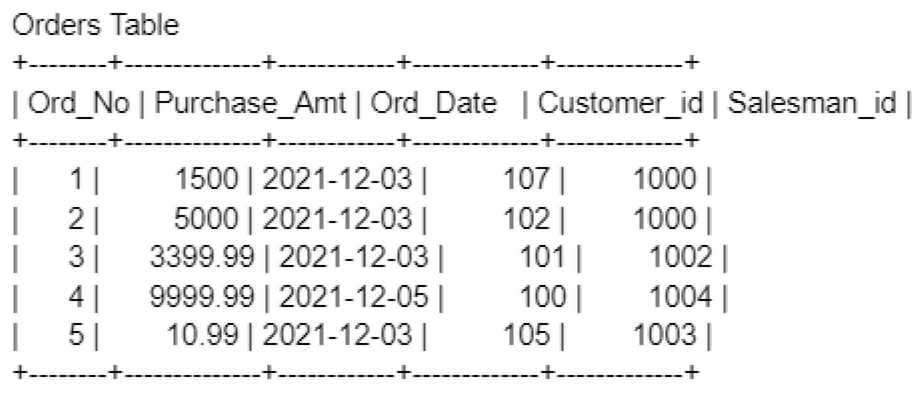
Salesman Table

```
+-------------+------------+---------+------------+
| Salesman_id | Name       | City    | Commission |
+-------------+------------+---------+------------+
|        1000 | Rahul      | Kolkata |    2499.99 |
|        1001 | Abhishek   | Mumbai  |       1000 |
|        1002 | Shyam      | Lucknow |     9999.5 |
|        1003 | Ram        | Bhopal  |      150.3 |
|        1004 | Jignesh    | Jammu   |         10 |
+-------------+------------+---------+------------+
```



Customer Table

```
+-------------+-----------+-----------+-------+-------------+
| Customer_id | Cust_Name | City      | Grade | Salesman_id |
+-------------+-----------+-----------+-------+-------------+
|         100 | Surohit   | Lucknow   |   9.5 |        1004 |
|         101 | Shreya    | Bhopal    |     5 |        1002 |
|         102 | Boman     | Pune      |     9 |        1000 |
|         103 | Nishita   | X         |     2 |        1000 |
|         104 | Raman     | W         |   8.5 |        1004 |
|         105 | Rohit     | Bangalore |     2 |        1003 |
|         106 | Lokesh    | Bangalore |    10 |        1001 |
|         107 | Rakshi    | Kolkata   |   5.5 |        1000 |
+-------------+-----------+-----------+-------+-------------+
```



Orders Table

```
+--------+--------------+------------+-------------+-------------+
| Ord_No | Purchase_Amt | Ord_Date   | Customer_id | Salesman_id |
+--------+--------------+------------+-------------+-------------+
|      1 |         1500 | 2021-12-03 |         107 |        1000 |
|      2 |         5000 | 2021-12-03 |         102 |        1000 |
|      3 |      3399.99 | 2021-12-03 |         101 |        1002 |
|      4 |      9999.99 | 2021-12-05 |         100 |        1004 |
|      5 |        10.99 | 2021-12-03 |         105 |        1003 |
+--------+--------------+------------+-------------+-------------+
```

Using the above database:
(a) Count the customers with grades above Bangalore's average.
(b) Find the name and numbers of all salesmen who had more than one customer.
(c) List all salesmen and indicate those who have and don't have customers in their Cities, using union operation
(d) Create a view that finds the salesman who has the customer with the highest order of a day.
(e) Demonstrate the DELETE operation by removing the salesman with id 1000. All his orders must also be deleted.