SQL

DAY - 1

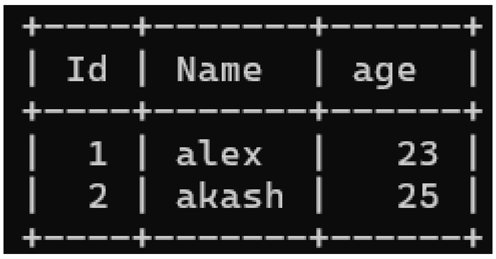
Create database Ds;

Use Ds;

Create table Emp(Id int Not null, Name char(10) not null, age int check (age>18));

insert into emp values(1,"alex",23),(2,"akash",25);

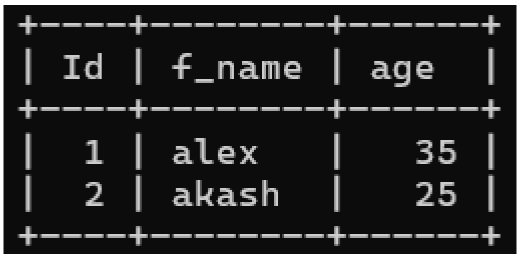
select * from emp;



```
+----+-------+------+
| Id | Name  | age  |
+----+-------+------+
|  1 | alex  |   23 |
|  2 | akash |   25 |
+----+-------+------+
```

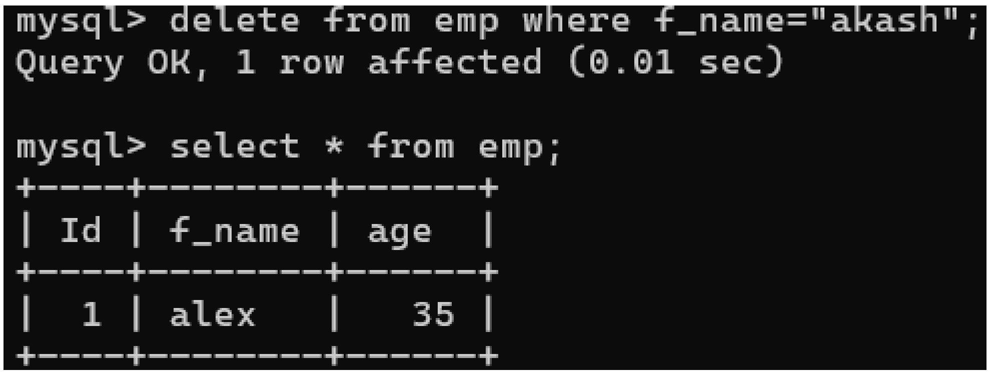
alter table emp rename column name to f_name;

update emp set age=35 where f_name="alex";



```
+----+--------+------+
| Id | f_name | age  |
+----+--------+------+
|  1 | alex   |   35 |
|  2 | akash  |   25 |
+----+--------+------+
```

Deleting a record:

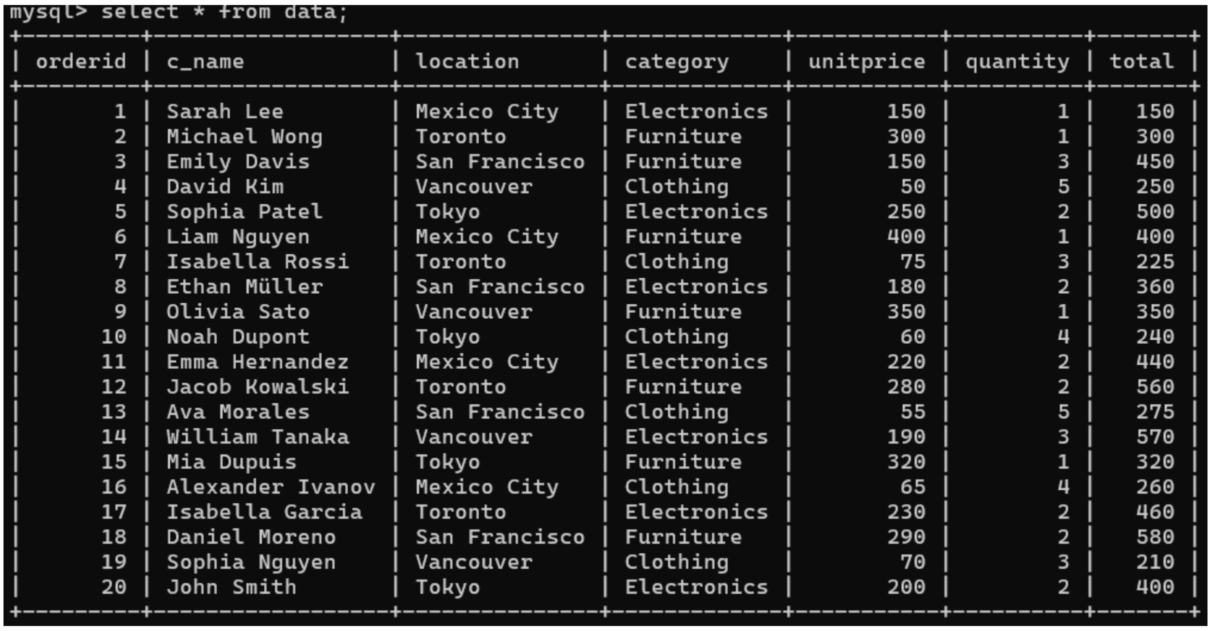


```
mysql> delete from emp where f_name="akash";
Query OK, 1 row affected (0.01 sec)

mysql> select * from emp;
+----+--------+------+
| Id | f_name | age  |
+----+--------+------+
|  1 | alex   |   35 |
+----+--------+------+
```

DAY - 2

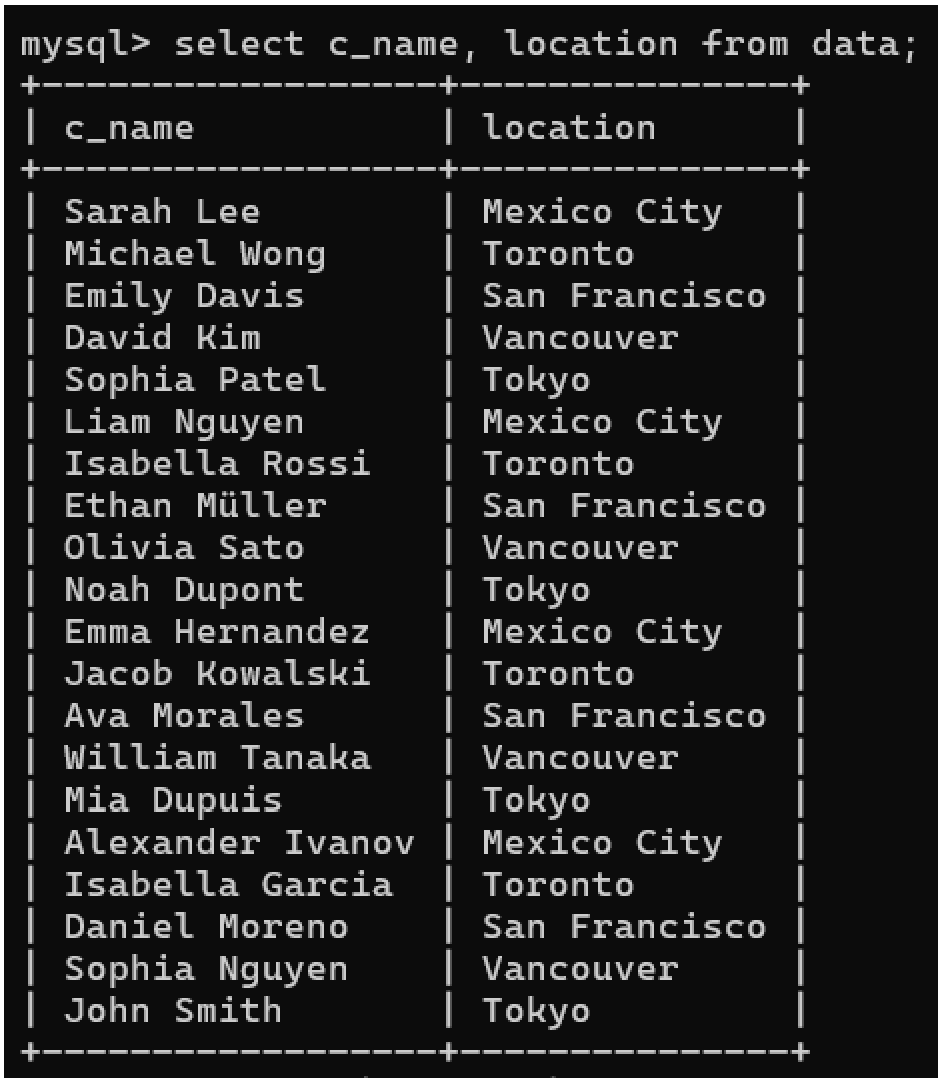
```
2) create table data(orderid int primary key,c_name varchar(25) not null,location varchar(15)
not null,category varchar(20) not null,unitprice int not null,quantity int not null,total int not
null);

insert into data values(1, 'Sarah Lee', 'Mexico City', 'Electronics', 150, 1, 150),
    -> (2, 'Michael Wong', 'Toronto', 'Furniture', 300, 1, 300),
    -> (3, 'Emily Davis', 'San Francisco', 'Furniture', 150, 3, 450),
    -> (4, 'David Kim', 'Vancouver', 'Clothing', 50, 5, 250),
    -> (5, 'Sophia Patel', 'Tokyo', 'Electronics', 250, 2, 500),
    -> (6, 'Liam Nguyen', 'Mexico City', 'Furniture', 400, 1, 400),
    -> (7, 'Isabella Rossi', 'Toronto', 'Clothing', 75, 3, 225),
    -> (8, 'Ethan Müller', 'San Francisco', 'Electronics', 180, 2, 360),
    -> (9, 'Olivia Sato', 'Vancouver', 'Furniture', 350, 1, 350),
    -> (10, 'Noah Dupont', 'Tokyo', 'Clothing', 60, 4, 240),
    -> (11, 'Emma Hernandez', 'Mexico City', 'Electronics', 220, 2, 440),
    -> (12, 'Jacob Kowalski', 'Toronto', 'Furniture', 280, 2, 560),
    -> (13, 'Ava Morales', 'San Francisco', 'Clothing', 55, 5, 275),
    -> (14, 'William Tanaka', 'Vancouver', 'Electronics', 190, 3, 570),
    -> (15, 'Mia Dupuis', 'Tokyo', 'Furniture', 320, 1, 320),
    -> (16, 'Alexander Ivanov', 'Mexico City', 'Clothing', 65, 4, 260),
    -> (17, 'Isabella Garcia', 'Toronto', 'Electronics', 230, 2, 460),
    -> (18, 'Daniel Moreno', 'San Francisco', 'Furniture', 290, 2, 580),
    -> (19, 'Sophia Nguyen', 'Vancouver', 'Clothing', 70, 3, 210),
    -> (20, 'John Smith', 'Tokyo', 'Electronics', 200, 2, 400);
```

Viewing table:



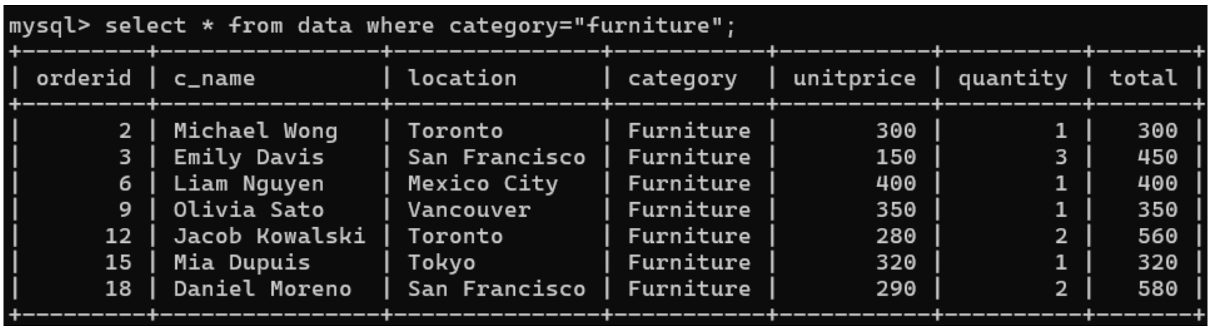
```
mysql> select * from data;
+---------+------------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name           | location      | category    | unitprice | quantity | total |
+---------+------------------+---------------+-------------+-----------+----------+-------+
|       1 | Sarah Lee        | Mexico City   | Electronics |       150 |        1 |   150 |
|       2 | Michael Wong     | Toronto       | Furniture   |       300 |        1 |   300 |
|       3 | Emily Davis      | San Francisco | Furniture   |       150 |        3 |   450 |
|       4 | David Kim        | Vancouver     | Clothing    |        50 |        5 |   250 |
|       5 | Sophia Patel     | Tokyo         | Electronics |       250 |        2 |   500 |
|       6 | Liam Nguyen      | Mexico City   | Furniture   |       400 |        1 |   400 |
|       7 | Isabella Rossi   | Toronto       | Clothing    |        75 |        3 |   225 |
|       8 | Ethan Müller     | San Francisco | Electronics |       180 |        2 |   360 |
|       9 | Olivia Sato      | Vancouver     | Furniture   |       350 |        1 |   350 |
|      10 | Noah Dupont      | Tokyo         | Clothing    |        60 |        4 |   240 |
|      11 | Emma Hernandez   | Mexico City   | Electronics |       220 |        2 |   440 |
|      12 | Jacob Kowalski   | Toronto       | Furniture   |       280 |        2 |   560 |
|      13 | Ava Morales      | San Francisco | Clothing    |        55 |        5 |   275 |
|      14 | William Tanaka   | Vancouver     | Electronics |       190 |        3 |   570 |
|      15 | Mia Dupuis       | Tokyo         | Furniture   |       320 |        1 |   320 |
|      16 | Alexander Ivanov | Mexico City   | Clothing    |        65 |        4 |   260 |
|      17 | Isabella Garcia  | Toronto       | Electronics |       230 |        2 |   460 |
|      18 | Daniel Moreno    | San Francisco | Furniture   |       290 |        2 |   580 |
|      19 | Sophia Nguyen    | Vancouver     | Clothing    |        70 |        3 |   210 |
|      20 | John Smith       | Tokyo         | Electronics |       200 |        2 |   400 |
+---------+------------------+---------------+-------------+-----------+----------+-------+
```

1)Customer name and location:



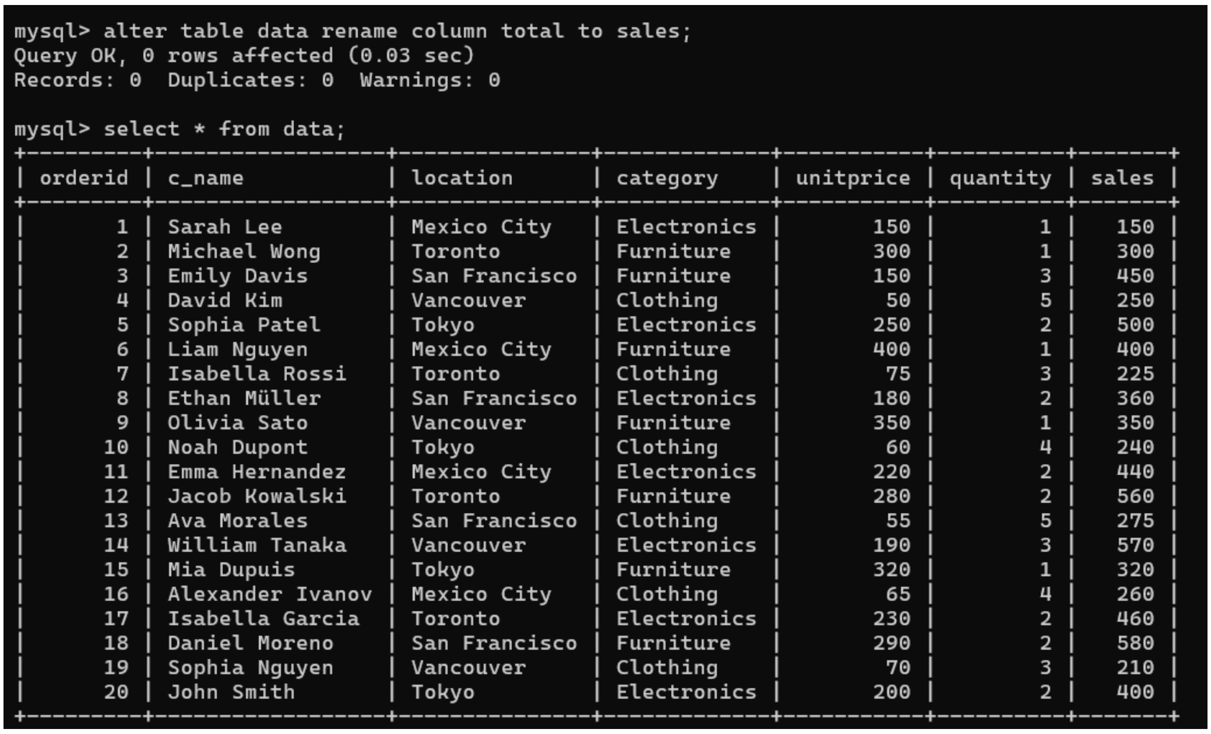
```
mysql> select c_name, location from data;
+------------------+---------------+
| c_name           | location      |
+------------------+---------------+
| Sarah Lee        | Mexico City   |
| Michael Wong     | Toronto       |
| Emily Davis      | San Francisco |
| David Kim        | Vancouver     |
| Sophia Patel     | Tokyo         |
| Liam Nguyen      | Mexico City   |
| Isabella Rossi   | Toronto       |
| Ethan Müller     | San Francisco |
| Olivia Sato      | Vancouver     |
| Noah Dupont      | Tokyo         |
| Emma Hernandez   | Mexico City   |
| Jacob Kowalski   | Toronto       |
| Ava Morales      | San Francisco |
| William Tanaka   | Vancouver     |
| Mia Dupuis       | Tokyo         |
| Alexander Ivanov | Mexico City   |
| Isabella Garcia  | Toronto       |
| Daniel Moreno    | San Francisco |
| Sophia Nguyen    | Vancouver     |
| John Smith       | Tokyo         |
+------------------+---------------+
```

2)All data for furniture:



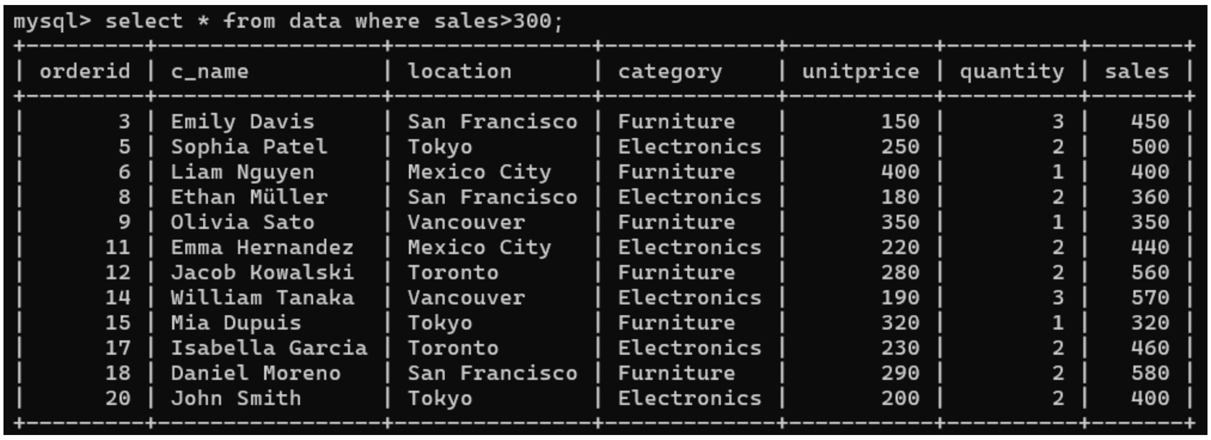
```
mysql> select * from data where category="furniture";
+---------+------------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name           | location      | category    | unitprice | quantity | total |
+---------+------------------+---------------+-------------+-----------+----------+-------+
|       2 | Michael Wong     | Toronto       | Furniture   |       300 |        1 |   300 |
|       3 | Emily Davis      | San Francisco | Furniture   |       150 |        3 |   450 |
|       6 | Liam Nguyen      | Mexico City   | Furniture   |       400 |        1 |   400 |
|       9 | Olivia Sato      | Vancouver     | Furniture   |       350 |        1 |   350 |
|      12 | Jacob Kowalski   | Toronto       | Furniture   |       280 |        2 |   560 |
|      15 | Mia Dupuis       | Tokyo         | Furniture   |       320 |        1 |   320 |
|      18 | Daniel Moreno    | San Francisco | Furniture   |       290 |        2 |   580 |
+---------+------------------+---------------+-------------+-----------+----------+-------+
```

3)Rename total into sales:



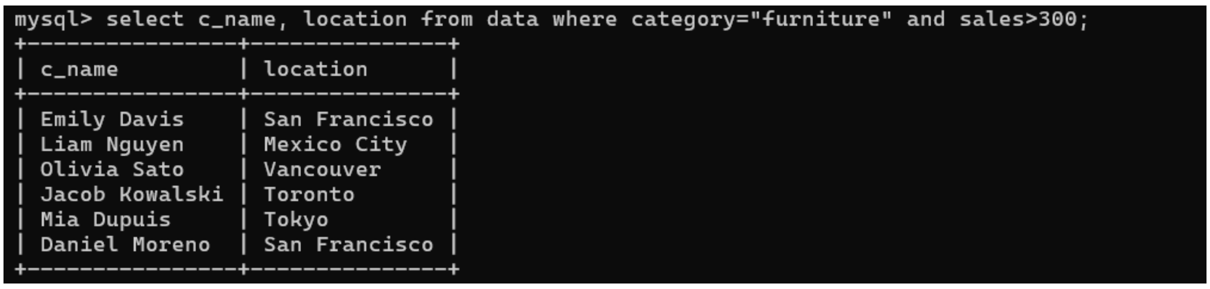
```
mysql> alter table data rename column total to sales;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from data;
+---------+------------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name           | location      | category    | unitprice | quantity | sales |
+---------+------------------+---------------+-------------+-----------+----------+-------+
|       1 | Sarah Lee        | Mexico City   | Electronics |       150 |        1 |   150 |
|       2 | Michael Wong     | Toronto       | Furniture   |       300 |        1 |   300 |
|       3 | Emily Davis      | San Francisco | Furniture   |       150 |        3 |   450 |
|       4 | David Kim        | Vancouver     | Clothing    |        50 |        5 |   250 |
|       5 | Sophia Patel     | Tokyo         | Electronics |       250 |        2 |   500 |
|       6 | Liam Nguyen      | Mexico City   | Furniture   |       400 |        1 |   400 |
|       7 | Isabella Rossi   | Toronto       | Clothing    |        75 |        3 |   225 |
|       8 | Ethan Müller     | San Francisco | Electronics |       180 |        2 |   360 |
|       9 | Olivia Sato      | Vancouver     | Furniture   |       350 |        1 |   350 |
|      10 | Noah Dupont      | Tokyo         | Clothing    |        60 |        4 |   240 |
|      11 | Emma Hernandez   | Mexico City   | Electronics |       220 |        2 |   440 |
|      12 | Jacob Kowalski   | Toronto       | Furniture   |       280 |        2 |   560 |
|      13 | Ava Morales      | San Francisco | Clothing    |        55 |        5 |   275 |
|      14 | William Tanaka   | Vancouver     | Electronics |       190 |        3 |   570 |
|      15 | Mia Dupuis       | Tokyo         | Furniture   |       320 |        1 |   320 |
|      16 | Alexander Ivanov | Mexico City   | Clothing    |        65 |        4 |   260 |
|      17 | Isabella Garcia  | Toronto       | Electronics |       230 |        2 |   460 |
|      18 | Daniel Moreno    | San Francisco | Furniture   |       290 |        2 |   580 |
|      19 | Sophia Nguyen    | Vancouver     | Clothing    |        70 |        3 |   210 |
|      20 | John Smith       | Tokyo         | Electronics |       200 |        2 |   400 |
+---------+------------------+---------------+-------------+-----------+----------+-------+
```

4)All data where sales above 300:



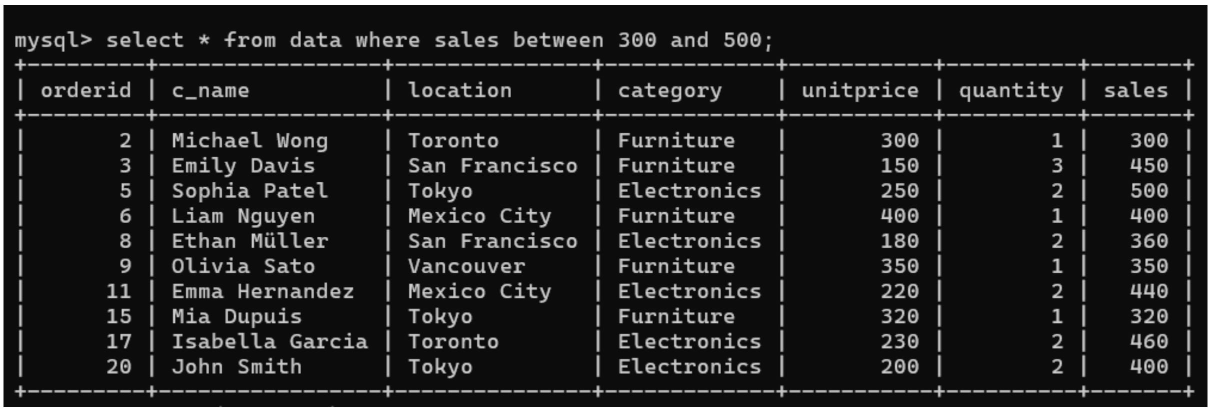
```
mysql> select * from data where sales>300;
+---------+------------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name           | location      | category    | unitprice | quantity | sales |
+---------+------------------+---------------+-------------+-----------+----------+-------+
|       3 | Emily Davis      | San Francisco | Furniture   |       150 |        3 |   450 |
|       5 | Sophia Patel     | Tokyo         | Electronics |       250 |        2 |   500 |
|       6 | Liam Nguyen      | Mexico City   | Furniture   |       400 |        1 |   400 |
|       8 | Ethan Müller     | San Francisco | Electronics |       180 |        2 |   360 |
|       9 | Olivia Sato      | Vancouver     | Furniture   |       350 |        1 |   350 |
|      11 | Emma Hernandez   | Mexico City   | Electronics |       220 |        2 |   440 |
|      12 | Jacob Kowalski   | Toronto       | Furniture   |       280 |        2 |   560 |
|      14 | William Tanaka   | Vancouver     | Electronics |       190 |        3 |   570 |
|      15 | Mia Dupuis       | Tokyo         | Furniture   |       320 |        1 |   320 |
|      17 | Isabella Garcia  | Toronto       | Electronics |       230 |        2 |   460 |
|      18 | Daniel Moreno    | San Francisco | Furniture   |       290 |        2 |   580 |
|      20 | John Smith       | Tokyo         | Electronics |       200 |        2 |   400 |
+---------+------------------+---------------+-------------+-----------+----------+-------+
```

5)Select c_name and location for category furniture and sales above 300:



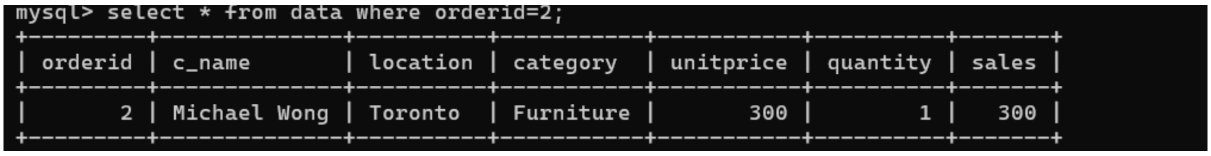
```
mysql> select c_name, location from data where category="furniture" and sales>300;
+----------------+---------------+
| c_name         | location      |
+----------------+---------------+
| Emily Davis    | San Francisco |
| Liam Nguyen    | Mexico City   |
| Olivia Sato    | Vancouver     |
| Jacob Kowalski | Toronto       |
| Mia Dupuis     | Tokyo         |
| Daniel Moreno  | San Francisco |
+----------------+---------------+
```

6)All data for sales from 300 to 500:



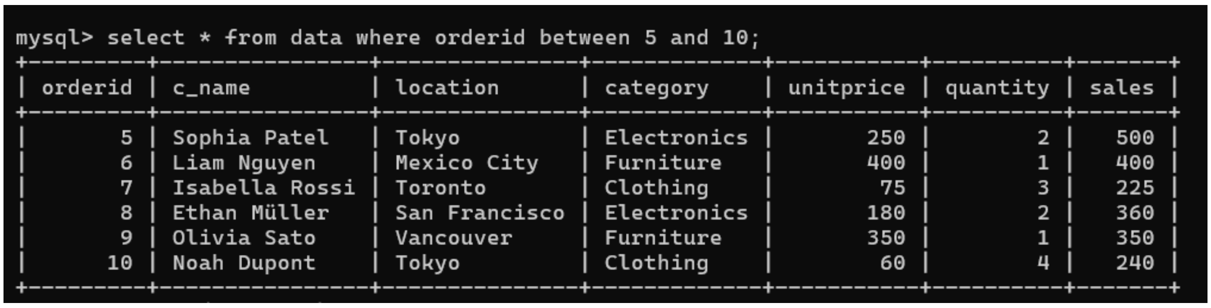
```
mysql> select * from data where sales between 300 and 500;
+---------+-----------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name          | location      | category    | unitprice | quantity | sales |
+---------+-----------------+---------------+-------------+-----------+----------+-------+
|       2 | Michael Wong    | Toronto       | Furniture   |       300 |        1 |   300 |
|       3 | Emily Davis     | San Francisco | Furniture   |       150 |        3 |   450 |
|       5 | Sophia Patel    | Tokyo         | Electronics |       250 |        2 |   500 |
|       6 | Liam Nguyen     | Mexico City   | Furniture   |       400 |        1 |   400 |
|       8 | Ethan Müller    | San Francisco | Electronics |       180 |        2 |   360 |
|       9 | Olivia Sato     | Vancouver     | Furniture   |       350 |        1 |   350 |
|      11 | Emma Hernandez  | Mexico City   | Electronics |       220 |        2 |   440 |
|      15 | Mia Dupuis      | Tokyo         | Furniture   |       320 |        1 |   320 |
|      17 | Isabella Garcia | Toronto       | Electronics |       230 |        2 |   460 |
|      20 | John Smith      | Tokyo         | Electronics |       200 |        2 |   400 |
+---------+-----------------+---------------+-------------+-----------+----------+-------+
```

7)All data for order id 2:



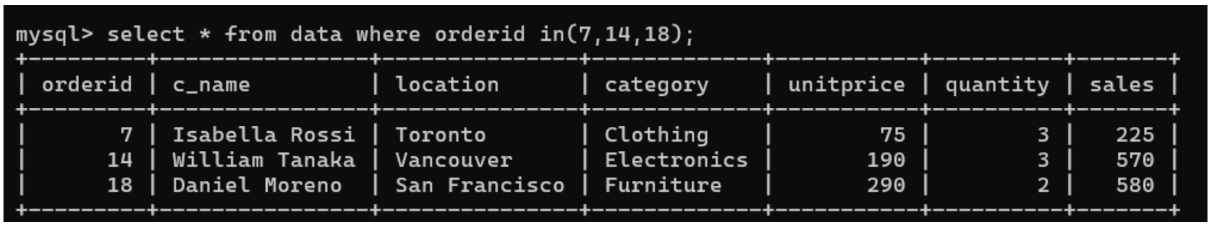
```
mysql> select * from data where orderid=2;
+---------+--------------+----------+-----------+-----------+----------+-------+
| orderid | c_name       | location | category  | unitprice | quantity | sales |
+---------+--------------+----------+-----------+-----------+----------+-------+
|       2 | Michael Wong | Toronto  | Furniture |       300 |        1 |   300 |
+---------+--------------+----------+-----------+-----------+----------+-------+
```

8)All data for order id from 5 to 10:



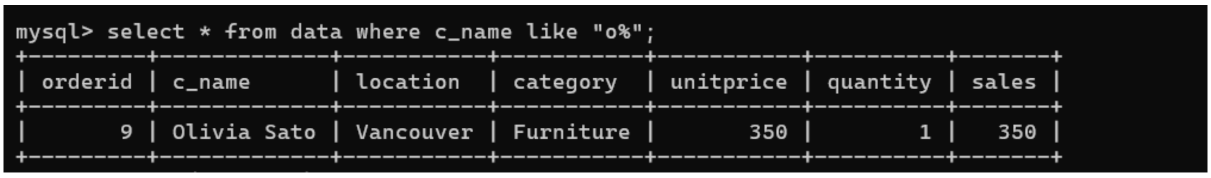
```
mysql> select * from data where orderid between 5 and 10;
+---------+----------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name         | location      | category    | unitprice | quantity | sales |
+---------+----------------+---------------+-------------+-----------+----------+-------+
|       5 | Sophia Patel   | Tokyo         | Electronics |       250 |        2 |   500 |
|       6 | Liam Nguyen    | Mexico City   | Furniture   |       400 |        1 |   400 |
|       7 | Isabella Rossi | Toronto       | Clothing    |        75 |        3 |   225 |
|       8 | Ethan Müller   | San Francisco | Electronics |       180 |        2 |   360 |
|       9 | Olivia Sato    | Vancouver     | Furniture   |       350 |        1 |   350 |
|      10 | Noah Dupont    | Tokyo         | Clothing    |        60 |        4 |   240 |
+---------+----------------+---------------+-------------+-----------+----------+-------+
```

9)1All data for order id 7,14,18:



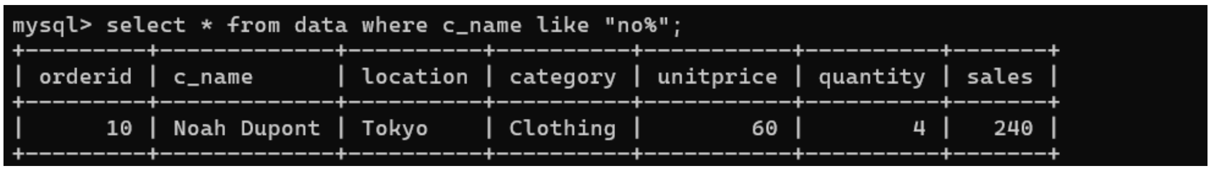
```
mysql> select * from data where orderid in(7,14,18);
+---------+----------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name         | location      | category    | unitprice | quantity | sales |
+---------+----------------+---------------+-------------+-----------+----------+-------+
|       7 | Isabella Rossi | Toronto       | Clothing    |        75 |        3 |   225 |
|      14 | William Tanaka | Vancouver     | Electronics |       190 |        3 |   570 |
|      18 | Daniel Moreno  | San Francisco | Furniture   |       290 |        2 |   580 |
+---------+----------------+---------------+-------------+-----------+----------+-------+
```

10)All data whose customer name starts with “o”:



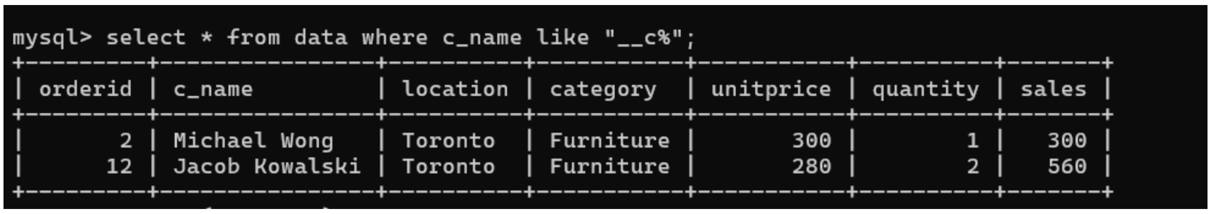
```
mysql> select * from data where c_name like "o%";
+---------+-------------+-----------+-----------+-----------+----------+-------+
| orderid | c_name      | location  | category  | unitprice | quantity | sales |
+---------+-------------+-----------+-----------+-----------+----------+-------+
|       9 | Olivia Sato | Vancouver | Furniture |       350 |        1 |   350 |
+---------+-------------+-----------+-----------+-----------+----------+-------+
```

11)1All data whose customer name starts with “no”:



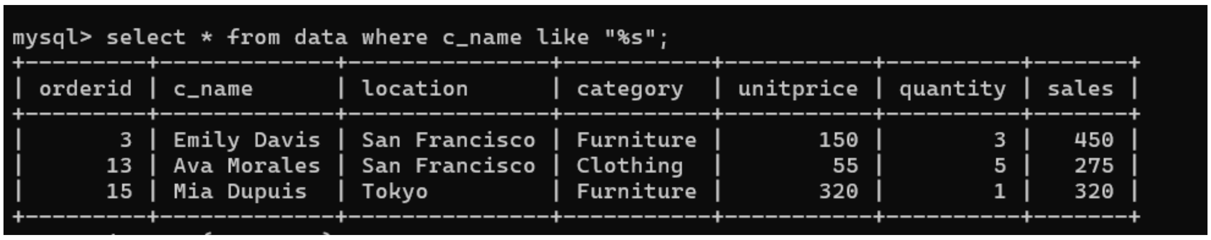
```
mysql> select * from data where c_name like "no%";
+---------+-------------+----------+----------+-----------+----------+-------+
| orderid | c_name      | location | category | unitprice | quantity | sales |
+---------+-------------+----------+----------+-----------+----------+-------+
|      10 | Noah Dupont | Tokyo    | Clothing |        60 |        4 |   240 |
+---------+-------------+----------+----------+-----------+----------+-------+
```

12)All data whose customer third letter start with “c”:



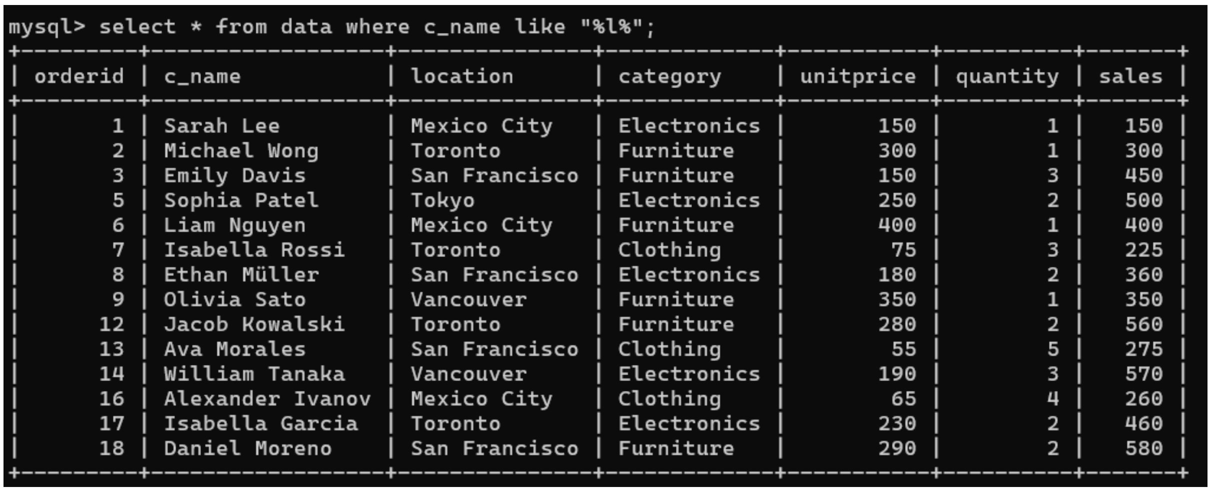
```
mysql> select * from data where c_name like "__c%";
+---------+----------------+----------+-----------+-----------+----------+-------+
| orderid | c_name         | location | category  | unitprice | quantity | sales |
+---------+----------------+----------+-----------+-----------+----------+-------+
|       2 | Michael Wong   | Toronto  | Furniture |       300 |        1 |   300 |
|      12 | Jacob Kowalski | Toronto  | Furniture |       280 |        2 |   560 |
+---------+----------------+----------+-----------+-----------+----------+-------+
```

13)All data whose customer name ends with “s”:



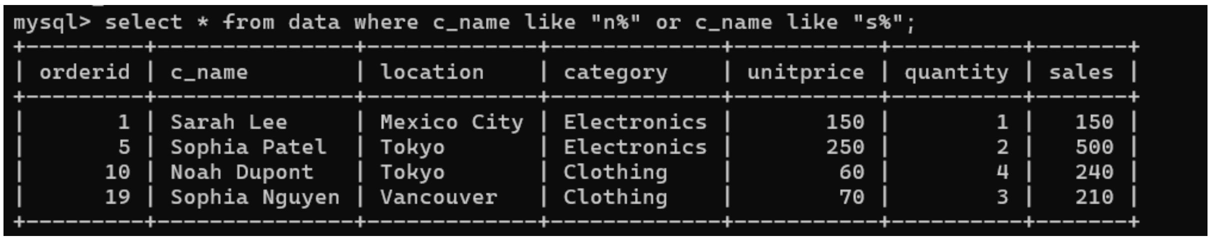
```
mysql> select * from data where c_name like "%s";
+---------+-------------+---------------+-----------+-----------+----------+-------+
| orderid | c_name      | location      | category  | unitprice | quantity | sales |
+---------+-------------+---------------+-----------+-----------+----------+-------+
|       3 | Emily Davis | San Francisco | Furniture |       150 |        3 |   450 |
|      13 | Ava Morales | San Francisco | Clothing  |        55 |        5 |   275 |
|      15 | Mia Dupuis  | Tokyo         | Furniture |       320 |        1 |   320 |
+---------+-------------+---------------+-----------+-----------+----------+-------+
```

14)All data whose customer name has “L” in it:



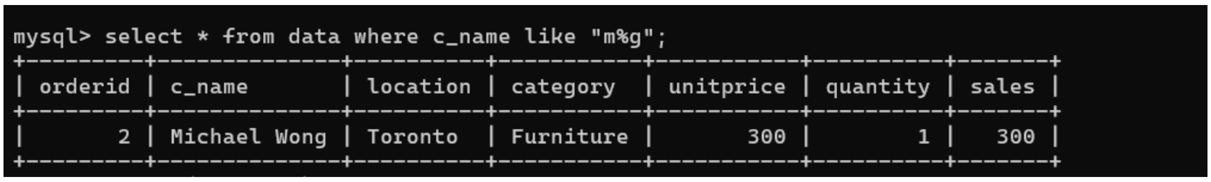
```
mysql> select * from data where c_name like "%l%";
+---------+-------------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name            | location      | category    | unitprice | quantity | sales |
+---------+-------------------+---------------+-------------+-----------+----------+-------+
|       1 | Sarah Lee         | Mexico City   | Electronics |       150 |        1 |   150 |
|       2 | Michael Wong      | Toronto       | Furniture   |       300 |        1 |   300 |
|       3 | Emily Davis       | San Francisco | Furniture   |       150 |        3 |   450 |
|       5 | Sophia Patel      | Tokyo         | Electronics |       250 |        2 |   500 |
|       6 | Liam Nguyen       | Mexico City   | Furniture   |       400 |        1 |   400 |
|       7 | Isabella Rossi    | Toronto       | Clothing    |        75 |        3 |   225 |
|       8 | Ethan Müller      | San Francisco | Electronics |       180 |        2 |   360 |
|       9 | Olivia Sato       | Vancouver     | Furniture   |       350 |        1 |   350 |
|      12 | Jacob Kowalski    | Toronto       | Furniture   |       280 |        2 |   560 |
|      13 | Ava Morales       | San Francisco | Clothing    |        55 |        5 |   275 |
|      14 | William Tanaka    | Vancouver     | Electronics |       190 |        3 |   570 |
|      16 | Alexander Ivanov  | Mexico City   | Clothing    |        65 |        4 |   260 |
|      17 | Isabella Garcia   | Toronto       | Electronics |       230 |        2 |   460 |
|      18 | Daniel Moreno     | San Francisco | Furniture   |       290 |        2 |   580 |
+---------+-------------------+---------------+-------------+-----------+----------+-------+
```

15)All data whose customer name starts with s or n



```
mysql> select * from data where c_name like "n%" or c_name like "s%";
+---------+---------------+-------------+-------------+-----------+----------+-------+
| orderid | c_name        | location    | category    | unitprice | quantity | sales |
+---------+---------------+-------------+-------------+-----------+----------+-------+
|       1 | Sarah Lee     | Mexico City | Electronics |       150 |        1 |   150 |
|       5 | Sophia Patel  | Tokyo       | Electronics |       250 |        2 |   500 |
|      10 | Noah Dupont   | Tokyo       | Clothing    |        60 |        4 |   240 |
|      19 | Sophia Nguyen | Vancouver   | Clothing    |        70 |        3 |   210 |
+---------+---------------+-------------+-------------+-----------+----------+-------+
```

16)All data whose customer name starts with m and ends with g:

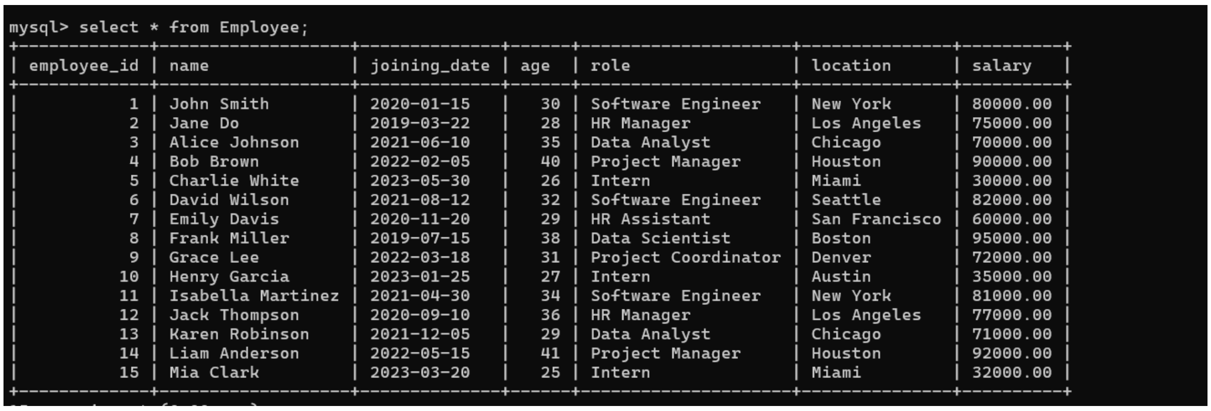


```
mysql> select * from data where c_name like "m%g";
+---------+--------------+---------+-----------+-----------+----------+-------+
| orderid | c_name       | location | category | unitprice | quantity | sales |
+---------+--------------+---------+-----------+-----------+----------+-------+
|       2 | Michael Wong | Toronto | Furniture |       300 |        1 |   300 |
+---------+--------------+---------+-----------+-----------+----------+-------+
```

TASK

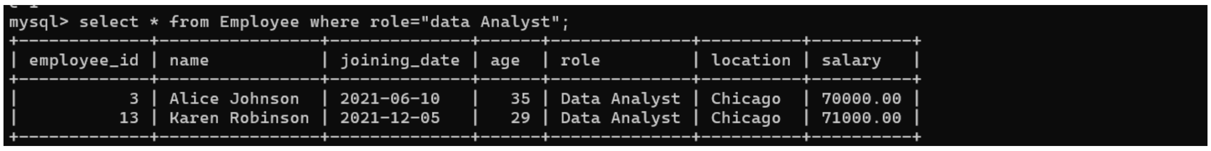
```
3) CREATE TABLE Employee (
    employee_id INT PRIMARY KEY,
    name VARCHAR(50),
    joining_date DATE,
    age INT,
    role VARCHAR(50),
    location VARCHAR(50),
    salary DECIMAL(10,2)
);

INSERT INTO Employee (employee_id, name, joining_date, age, role, location, salary)
VALUES
(1, 'John Smith', '2020-01-15', 30, 'Software Engineer', 'New York', 80000.00),
(2, 'Jane Do', '2019-03-22', 28, 'HR Manager', 'Los Angeles', 75000.00),
(3, 'Alice Johnson', '2021-06-10', 35, 'Data Analyst', 'Chicago', 70000.00),
(4, 'Bob Brown', '2022-02-05', 40, 'Project Manager', 'Houston', 90000.00),
(5, 'Charlie White', '2023-05-30', 26, 'Intern', 'Miami', 30000.00),
(6, 'David Wilson', '2021-08-12', 32, 'Software Engineer', 'Seattle', 82000.00),
(7, 'Emily Davis', '2020-11-20', 29, 'HR Assistant', 'San Francisco', 60000.00),
(8, 'Frank Miller', '2019-07-15', 38, 'Data Scientist', 'Boston', 95000.00),
(9, 'Grace Lee', '2022-03-18', 31, 'Project Coordinator', 'Denver', 72000.00),
(10, 'Henry Garcia', '2023-01-25', 27, 'Intern', 'Austin', 35000.00),
(11, 'Isabella Martinez', '2021-04-30', 34, 'Software Engineer', 'New York', 81000.00),
(12, 'Jack Thompson', '2020-09-10', 36, 'HR Manager', 'Los Angeles', 77000.00),
(13, 'Karen Robinson', '2021-12-05', 29, 'Data Analyst', 'Chicago', 71000.00),
(14, 'Liam Anderson', '2022-05-15', 41, 'Project Manager', 'Houston', 92000.00),
(15, 'Mia Clark', '2023-03-20', 25, 'Intern', 'Miami', 32000.00);
```

Viewing the table:



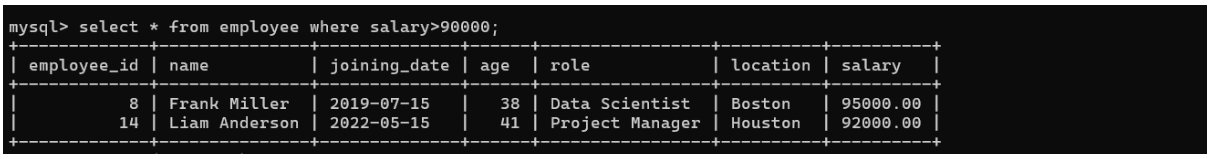
```
mysql> select * from Employee;
+-------------+-------------------+--------------+------+---------------------+---------------+----------+
| employee_id | name              | joining_date | age  | role                | location      | salary   |
+-------------+-------------------+--------------+------+---------------------+---------------+----------+
|           1 | John Smith        | 2020-01-15   |   30 | Software Engineer   | New York      | 80000.00 |
|           2 | Jane Do           | 2019-03-22   |   28 | HR Manager          | Los Angeles   | 75000.00 |
|           3 | Alice Johnson     | 2021-06-10   |   35 | Data Analyst        | Chicago       | 70000.00 |
|           4 | Bob Brown         | 2022-02-05   |   40 | Project Manager     | Houston       | 90000.00 |
|           5 | Charlie White     | 2023-05-30   |   26 | Intern              | Miami         | 30000.00 |
|           6 | David Wilson      | 2021-08-12   |   32 | Software Engineer   | Seattle       | 82000.00 |
|           7 | Emily Davis       | 2020-11-20   |   29 | HR Assistant        | San Francisco | 60000.00 |
|           8 | Frank Miller      | 2019-07-15   |   38 | Data Scientist      | Boston        | 95000.00 |
|           9 | Grace Lee         | 2022-03-18   |   31 | Project Coordinator | Denver        | 72000.00 |
|          10 | Henry Garcia      | 2023-01-25   |   27 | Intern              | Austin        | 35000.00 |
|          11 | Isabella Martinez | 2021-04-30   |   34 | Software Engineer   | New York      | 81000.00 |
|          12 | Jack Thompson     | 2020-09-10   |   36 | HR Manager          | Los Angeles   | 77000.00 |
|          13 | Karen Robinson    | 2021-12-05   |   29 | Data Analyst        | Chicago       | 71000.00 |
|          14 | Liam Anderson     | 2022-05-15   |   41 | Project Manager     | Houston       | 92000.00 |
|          15 | Mia Clark         | 2023-03-20   |   25 | Intern              | Miami         | 32000.00 |
+-------------+-------------------+--------------+------+---------------------+---------------+----------+
```

1)Data of employees working as data analyst:



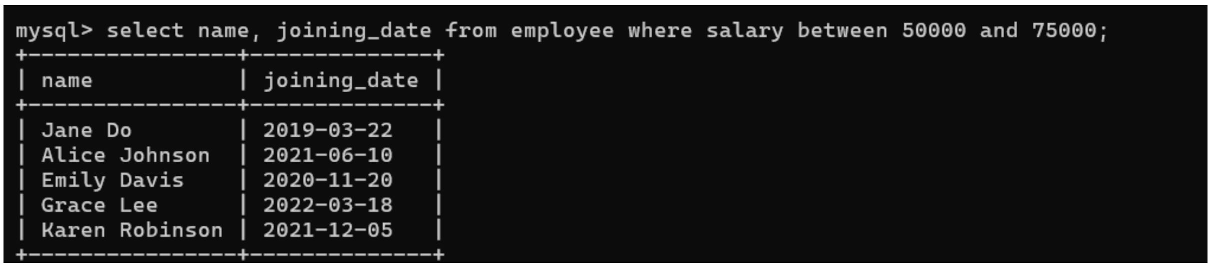
```
mysql> select * from Employee where role="data Analyst";
+-------------+----------------+--------------+------+--------------+----------+----------+
| employee_id | name           | joining_date | age  | role         | location | salary   |
+-------------+----------------+--------------+------+--------------+----------+----------+
|           3 | Alice Johnson  | 2021-06-10   |   35 | Data Analyst | Chicago  | 70000.00 |
|          13 | Karen Robinson | 2021-12-05   |   29 | Data Analyst | Chicago  | 71000.00 |
+-------------+----------------+--------------+------+--------------+----------+----------+
```

2)Employee details where salary is above 90000:



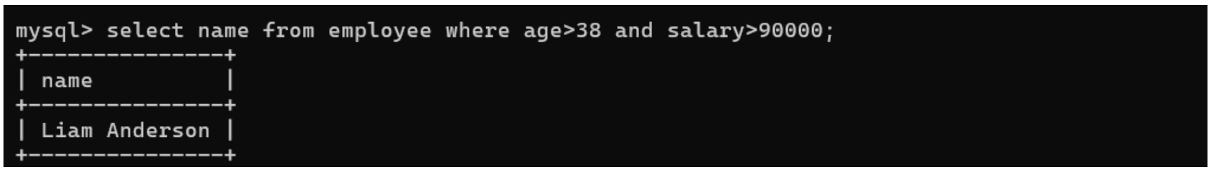
```
mysql> select * from employee where salary>90000;
+-------------+---------------+--------------+------+-----------------+----------+----------+
| employee_id | name          | joining_date | age  | role            | location | salary   |
+-------------+---------------+--------------+------+-----------------+----------+----------+
|           8 | Frank Miller  | 2019-07-15   |   38 | Data Scientist  | Boston   | 95000.00 |
|          14 | Liam Anderson | 2022-05-15   |   41 | Project Manager | Houston  | 92000.00 |
+-------------+---------------+--------------+------+-----------------+----------+----------+
```

3)Employee names and joining dates where the salary is between 50000 and 75000:



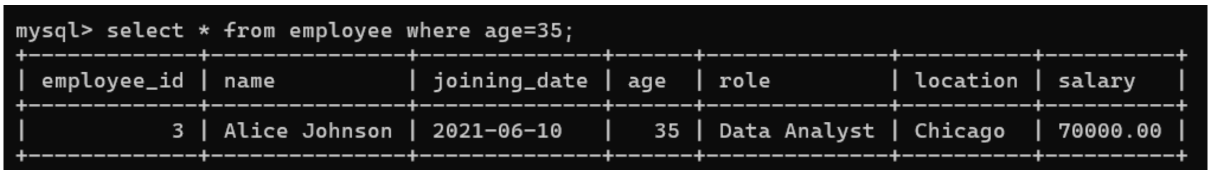
```
mysql> select name, joining_date from employee where salary between 50000 and 75000;
+----------------+--------------+
| name           | joining_date |
+----------------+--------------+
| Jane Do        | 2019-03-22   |
| Alice Johnson  | 2021-06-10   |
| Emily Davis    | 2020-11-20   |
| Grace Lee      | 2022-03-18   |
| Karen Robinson | 2021-12-05   |
+----------------+--------------+
```

4)Employee name whose age is above 38 and salary is above 90000:



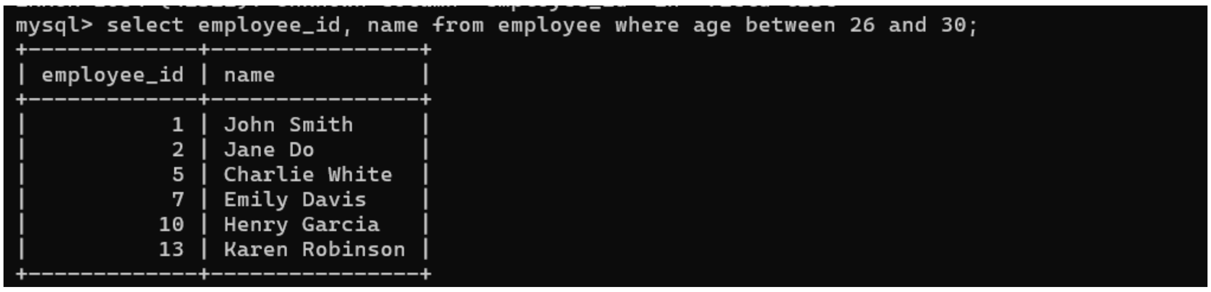
```
mysql> select name from employee where age>38 and salary>90000;
+---------------+
| name          |
+---------------+
| Liam Anderson |
+---------------+
```

5) Employee details where the age is 35:



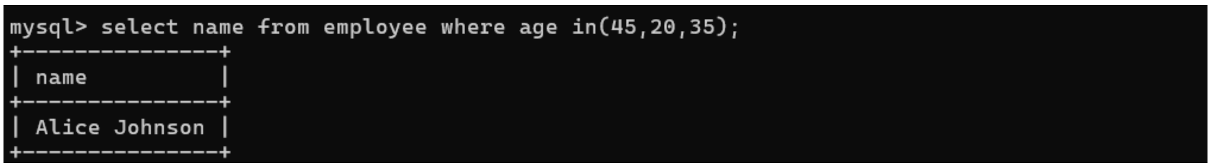
```
mysql> select * from employee where age=35;
+-------------+---------------+--------------+------+--------------+----------+----------+
| employee_id | name          | joining_date | age  | role         | location | salary   |
+-------------+---------------+--------------+------+--------------+----------+----------+
|           3 | Alice Johnson | 2021-06-10   |   35 | Data Analyst | Chicago  | 70000.00 |
+-------------+---------------+--------------+------+--------------+----------+----------+
```

6)Employee id and name where the age is between 26 and 30:



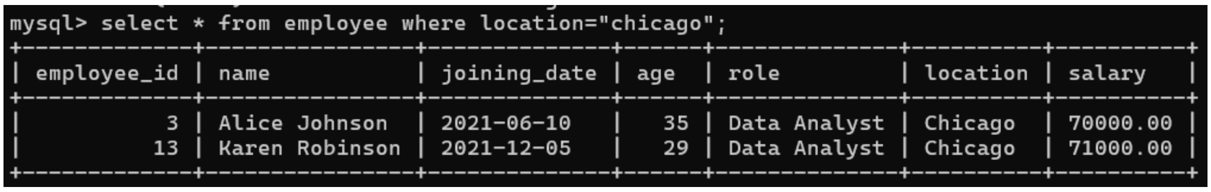
```
mysql> select employee_id, name from employee where age between 26 and 30;
+-------------+----------------+
| employee_id | name           |
+-------------+----------------+
|           1 | John Smith     |
|           2 | Jane Do        |
|           5 | Charlie White  |
|           7 | Emily Davis    |
|          10 | Henry Garcia   |
|          13 | Karen Robinson |
+-------------+----------------+
```

7)Employee names whose age is 45,20,35:



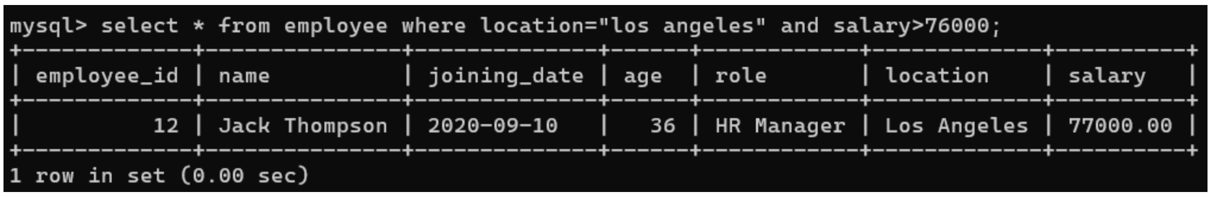
```
mysql> select name from employee where age in(45,20,35);
+---------------+
| name          |
+---------------+
| Alice Johnson |
+---------------+
```

8)Employee details who are working in Chicago:



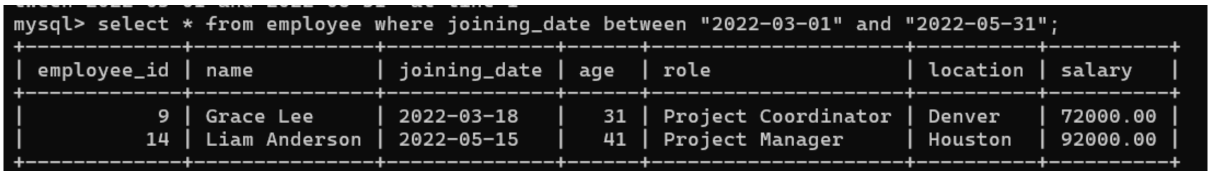
```
mysql> select * from employee where location="chicago";
+-------------+----------------+--------------+------+--------------+----------+----------+
| employee_id | name           | joining_date | age  | role         | location | salary   |
+-------------+----------------+--------------+------+--------------+----------+----------+
|           3 | Alice Johnson  | 2021-06-10   |   35 | Data Analyst | Chicago  | 70000.00 |
|          13 | Karen Robinson | 2021-12-05   |   29 | Data Analyst | Chicago  | 71000.00 |
+-------------+----------------+--------------+------+--------------+----------+----------+
```

9)Employee details who are working in los angeles and have a salary above 76000:



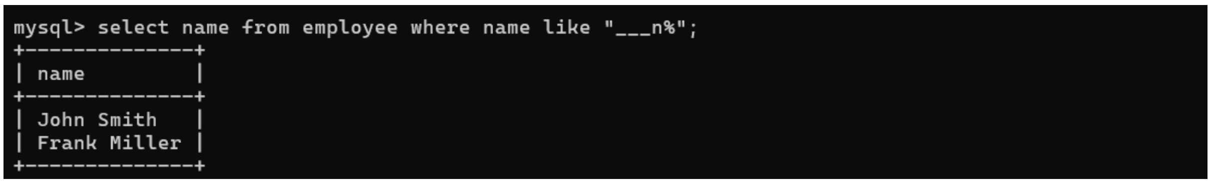
```
mysql> select * from employee where location="los angeles" and salary>76000;
+-------------+---------------+--------------+------+------------+-------------+----------+
| employee_id | name          | joining_date | age  | role       | location    | salary   |
+-------------+---------------+--------------+------+------------+-------------+----------+
|          12 | Jack Thompson | 2020-09-10   |   36 | HR Manager | Los Angeles | 77000.00 |
+-------------+---------------+--------------+------+------------+-------------+----------+
1 row in set (0.00 sec)
```

10)Employee who joined between March 1 2022 and May 31 2022:



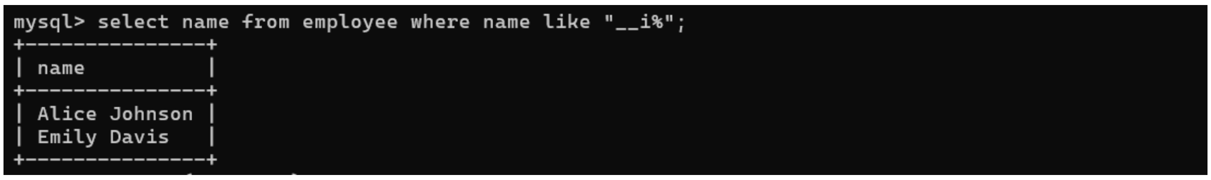
```
mysql> select * from employee where joining_date between "2022-03-01" and "2022-05-31";
+-------------+---------------+--------------+------+---------------------+----------+----------+
| employee_id | name          | joining_date | age  | role                | location | salary   |
+-------------+---------------+--------------+------+---------------------+----------+----------+
|           9 | Grace Lee     | 2022-03-18   |   31 | Project Coordinator | Denver   | 72000.00 |
|          14 | Liam Anderson | 2022-05-15   |   41 | Project Manager     | Houston  | 92000.00 |
+-------------+---------------+--------------+------+---------------------+----------+----------+
```

11)Employee names where the fourth letter is “n”:



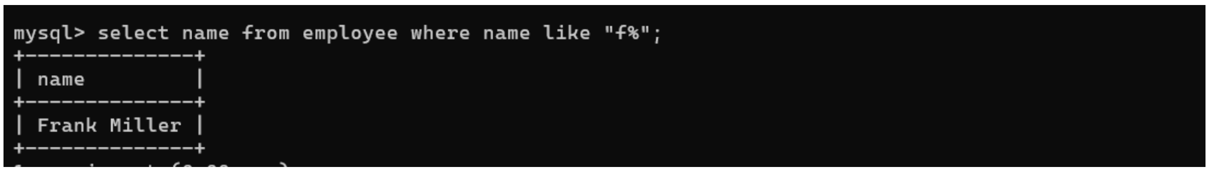
```
mysql> select name from employee where name like "___n%";
+--------------+
| name         |
+--------------+
| John Smith   |
| Frank Miller |
+--------------+
```

12)Employee names where the third letter is ”i”:



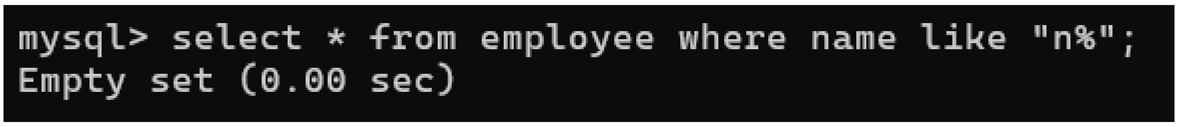
```
mysql> select name from employee where name like "__i%";
+---------------+
| name          |
+---------------+
| Alice Johnson |
| Emily Davis   |
+---------------+
```

13)Employee names that start with “F”:



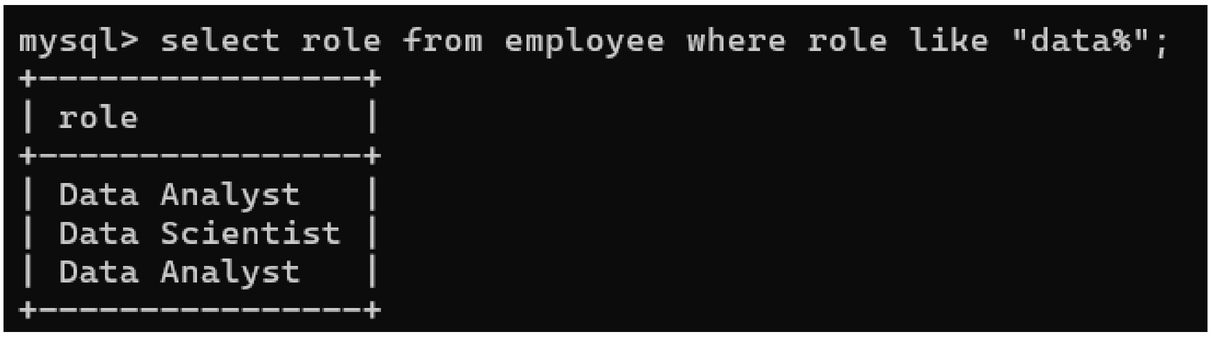
```
mysql> select name from employee where name like "f%";
+--------------+
| name         |
+--------------+
| Frank Miller |
+--------------+
```

14)All employee details where the name starts with ”N”:



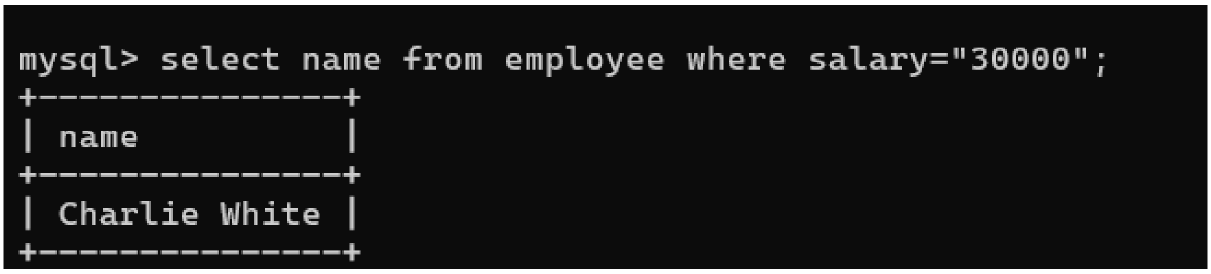
```
mysql> select * from employee where name like "n%";
Empty set (0.00 sec)
```

15)All roles that start with “Data”:



```
mysql> select role from employee where role like "data%";
+----------------+
| role           |
+----------------+
| Data Analyst   |
| Data Scientist |
| Data Analyst   |
+----------------+
```

16)Employee names whose salary is 30000:

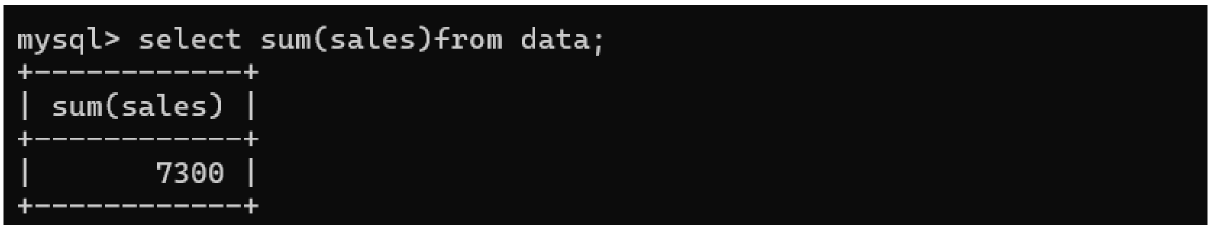


```
mysql> select name from employee where salary="30000";
+---------------+
| name          |
+---------------+
| Charlie White |
+---------------+
```

DAY -3

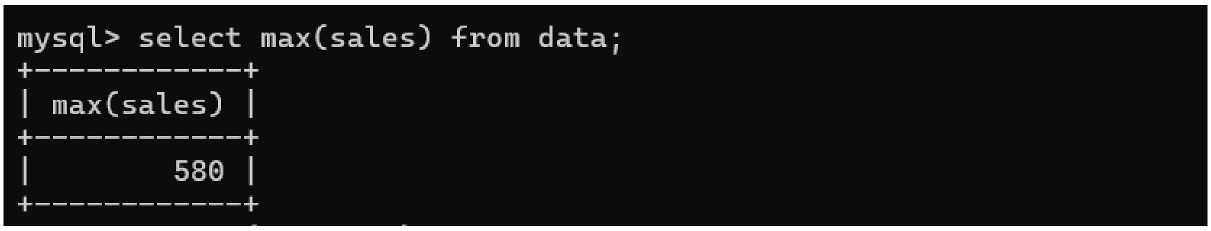
Data Table:

1)Total sales:



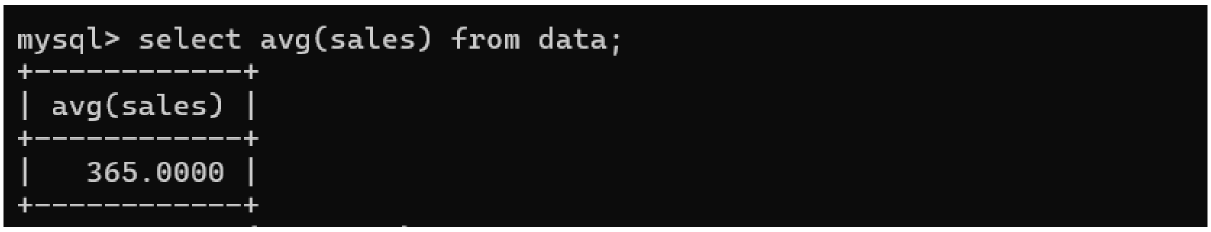
```
mysql> select sum(sales)from data;
+------------+
| sum(sales) |
+------------+
|       7300 |
+------------+
```

2)Maximum Sales:



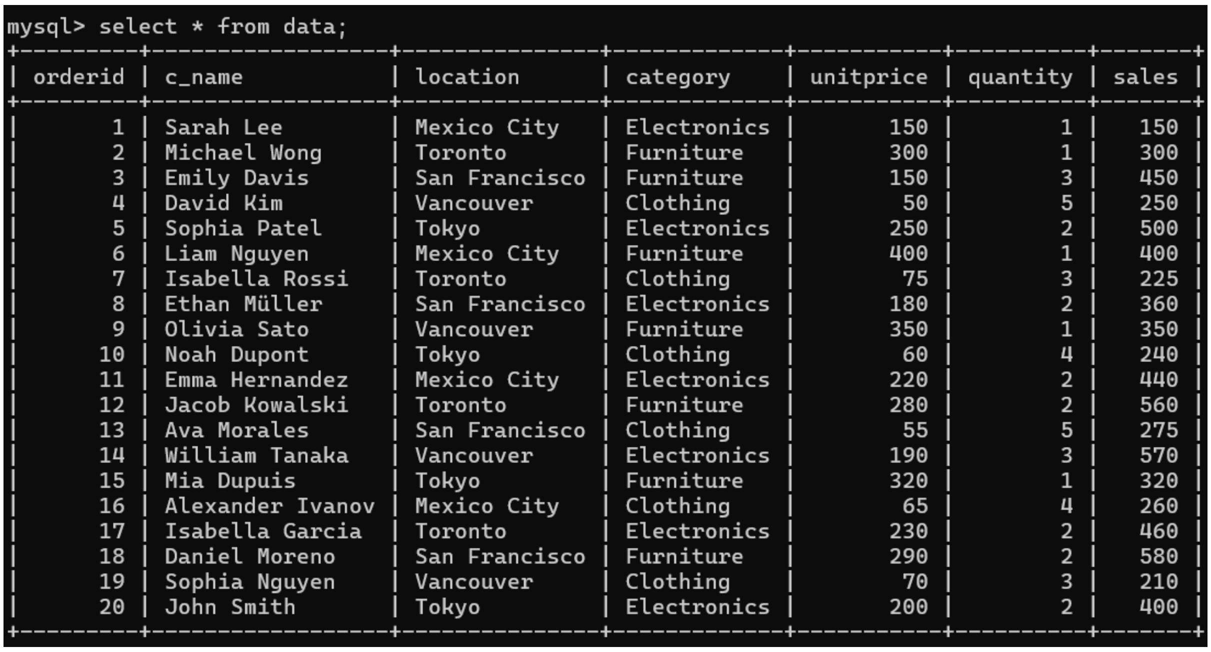
```
mysql> select max(sales) from data;
+------------+
| max(sales) |
+------------+
|        580 |
+------------+
```

3)Average Sales:



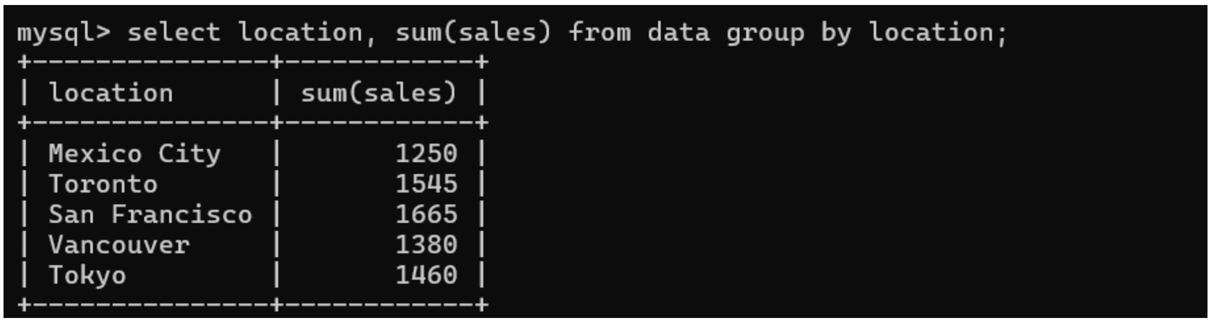
```
mysql> select avg(sales) from data;
+------------+
| avg(sales) |
+------------+
|   365.0000 |
+------------+
```

Viewing Table:



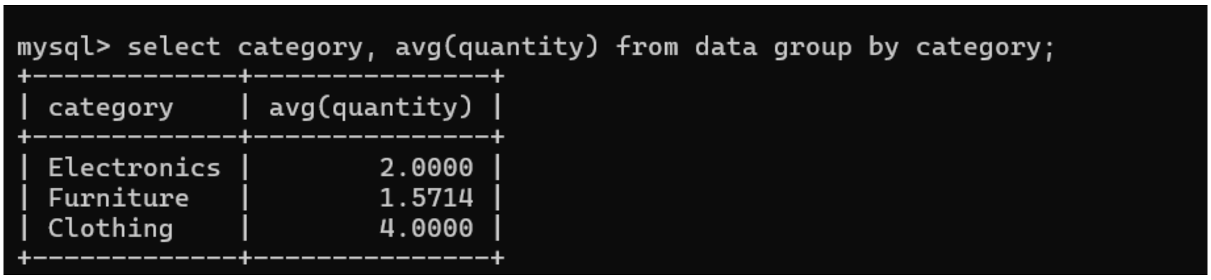
```
mysql> select * from data;
+---------+-------------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name            | location      | category    | unitprice | quantity | sales |
+---------+-------------------+---------------+-------------+-----------+----------+-------+
|       1 | Sarah Lee         | Mexico City   | Electronics |       150 |        1 |   150 |
|       2 | Michael Wong      | Toronto       | Furniture   |       300 |        1 |   300 |
|       3 | Emily Davis       | San Francisco | Furniture   |       150 |        3 |   450 |
|       4 | David Kim         | Vancouver     | Clothing    |        50 |        5 |   250 |
|       5 | Sophia Patel      | Tokyo         | Electronics |       250 |        2 |   500 |
|       6 | Liam Nguyen       | Mexico City   | Furniture   |       400 |        1 |   400 |
|       7 | Isabella Rossi    | Toronto       | Clothing    |        75 |        3 |   225 |
|       8 | Ethan Müller      | San Francisco | Electronics |       180 |        2 |   360 |
|       9 | Olivia Sato       | Vancouver     | Furniture   |       350 |        1 |   350 |
|      10 | Noah Dupont       | Tokyo         | Clothing    |        60 |        4 |   240 |
|      11 | Emma Hernandez    | Mexico City   | Electronics |       220 |        2 |   440 |
|      12 | Jacob Kowalski    | Toronto       | Furniture   |       280 |        2 |   560 |
|      13 | Ava Morales       | San Francisco | Clothing    |        55 |        5 |   275 |
|      14 | William Tanaka    | Vancouver     | Electronics |       190 |        3 |   570 |
|      15 | Mia Dupuis        | Tokyo         | Furniture   |       320 |        1 |   320 |
|      16 | Alexander Ivanov  | Mexico City   | Clothing    |        65 |        4 |   260 |
|      17 | Isabella Garcia   | Toronto       | Electronics |       230 |        2 |   460 |
|      18 | Daniel Moreno     | San Francisco | Furniture   |       290 |        2 |   580 |
|      19 | Sophia Nguyen     | Vancouver     | Clothing    |        70 |        3 |   210 |
|      20 | John Smith        | Tokyo         | Electronics |       200 |        2 |   400 |
+---------+-------------------+---------------+-------------+-----------+----------+-------+
```

4)Total sales for each location:



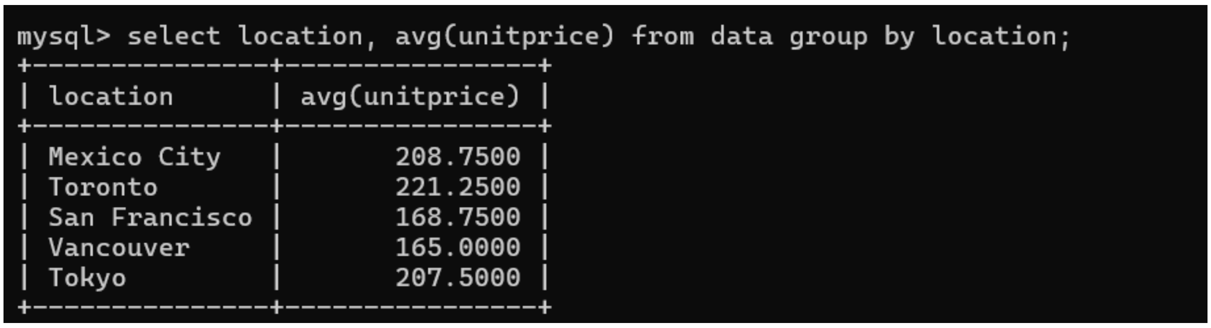
```
mysql> select location, sum(sales) from data group by location;
+---------------+------------+
| location      | sum(sales) |
+---------------+------------+
| Mexico City   |       1250 |
| Toronto       |       1545 |
| San Francisco |       1665 |
| Vancouver     |       1380 |
| Tokyo         |       1460 |
+---------------+------------+
```

5)Average quantity for each category:



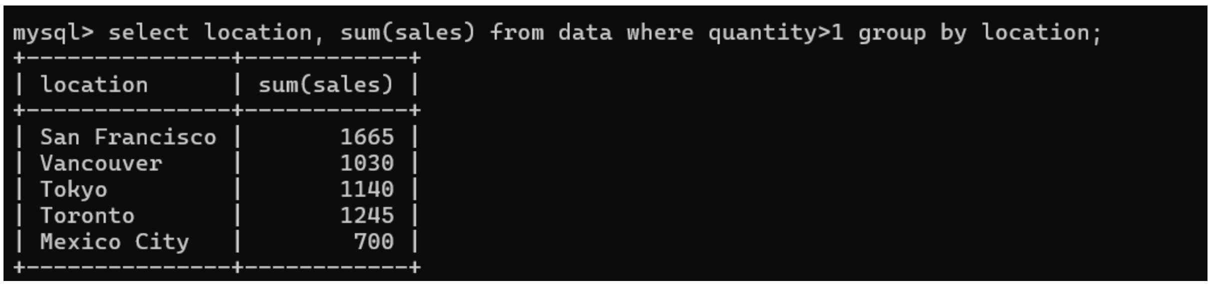
```
mysql> select category, avg(quantity) from data group by category;
+-------------+---------------+
| category    | avg(quantity) |
+-------------+---------------+
| Electronics |        2.0000 |
| Furniture   |        1.5714 |
| Clothing    |        4.0000 |
+-------------+---------------+
```

6)Average unit price for each location:



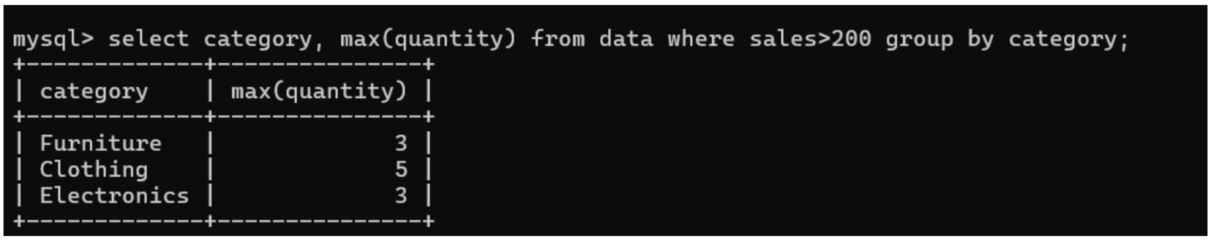
```
mysql> select location, avg(unitprice) from data group by location;
+---------------+----------------+
| location      | avg(unitprice) |
+---------------+----------------+
| Mexico City   |       208.7500 |
| Toronto       |       221.2500 |
| San Francisco |       168.7500 |
| Vancouver     |       165.0000 |
| Tokyo         |       207.5000 |
+---------------+----------------+
```

7)Total sales for each location for quantity greater than 1:



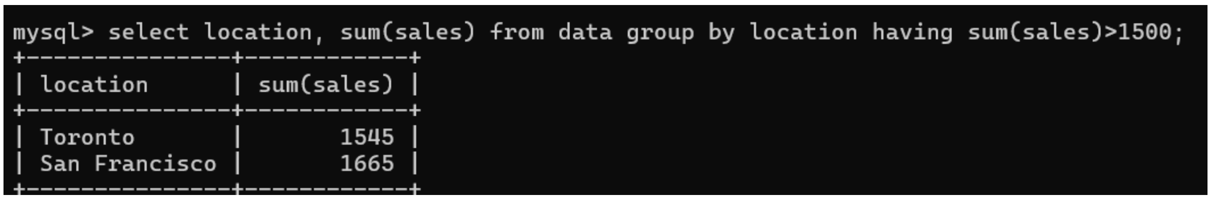
```
mysql> select location, sum(sales) from data where quantity>1 group by location;
+---------------+------------+
| location      | sum(sales) |
+---------------+------------+
| San Francisco |       1665 |
| Vancouver     |       1030 |
| Tokyo         |       1140 |
| Toronto       |       1245 |
| Mexico City   |        700 |
+---------------+------------+
```

8)Maximum quantity for each category for sales above 200:



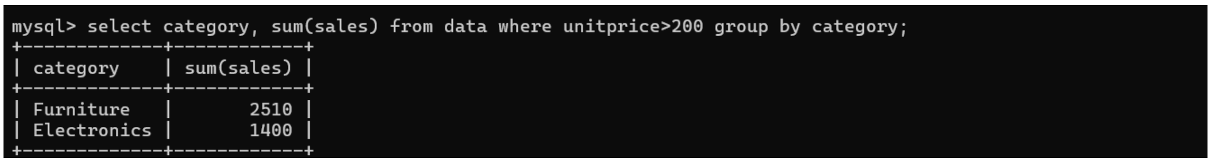
```
mysql> select category, max(quantity) from data where sales>200 group by category;
+-------------+---------------+
| category    | max(quantity) |
+-------------+---------------+
| Furniture   |             3 |
| Clothing    |             5 |
| Electronics |             3 |
+-------------+---------------+
```

9)Total sales for each location whose total sales is above 1500:



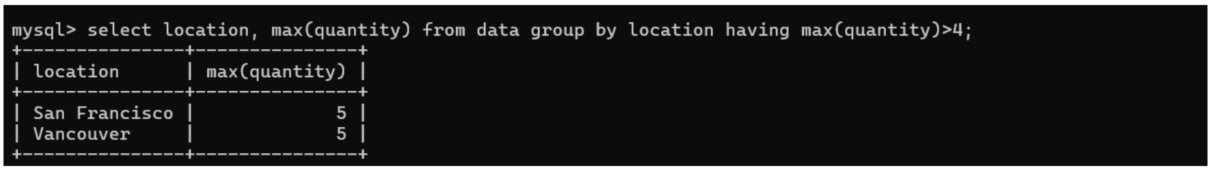
```
mysql> select location, sum(sales) from data group by location having sum(sales)>1500;
+---------------+------------+
| location      | sum(sales) |
+---------------+------------+
| Toronto       |       1545 |
| San Francisco |       1665 |
+---------------+------------+
```

10)Total sales for each category for unit price greater than 200:



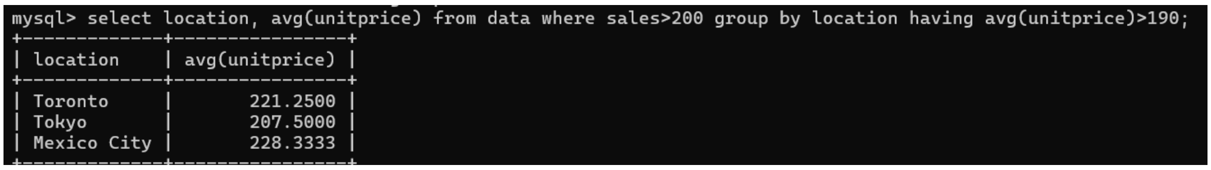
```
mysql> select category, sum(sales) from data where unitprice>200 group by category;
+-------------+------------+
| category    | sum(sales) |
+-------------+------------+
| Furniture   |       2510 |
| Electronics |       1400 |
+-------------+------------+
```

11)Maximum quantity for each location whose maximum quantity is greater than 4:



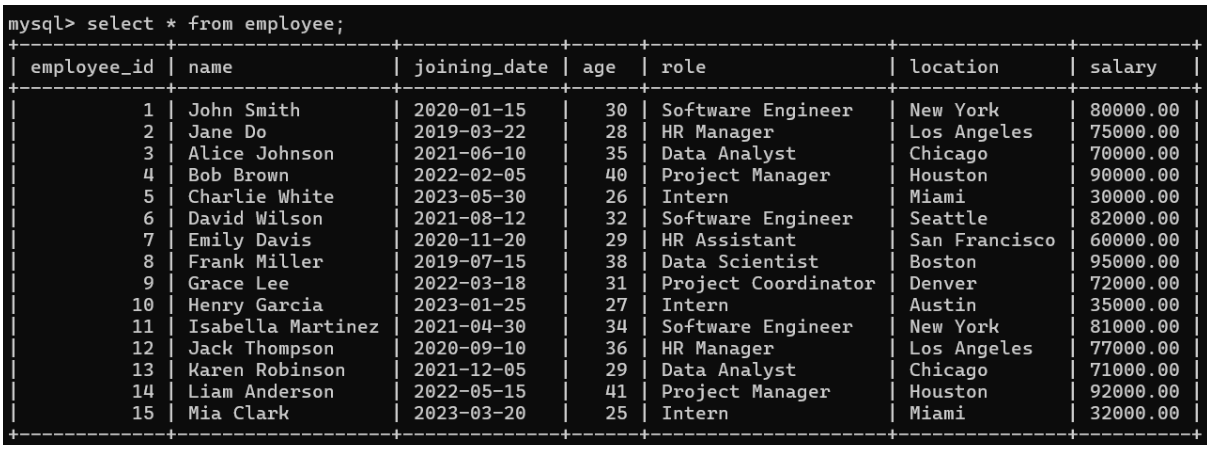
```
mysql> select location, max(quantity) from data group by location having max(quantity)>4;
+---------------+---------------+
| location      | max(quantity) |
+---------------+---------------+
| San Francisco |             5 |
| Vancouver     |             5 |
+---------------+---------------+
```

12)Average unit price for each location whose sales is above 200 and average unit price is above 190:



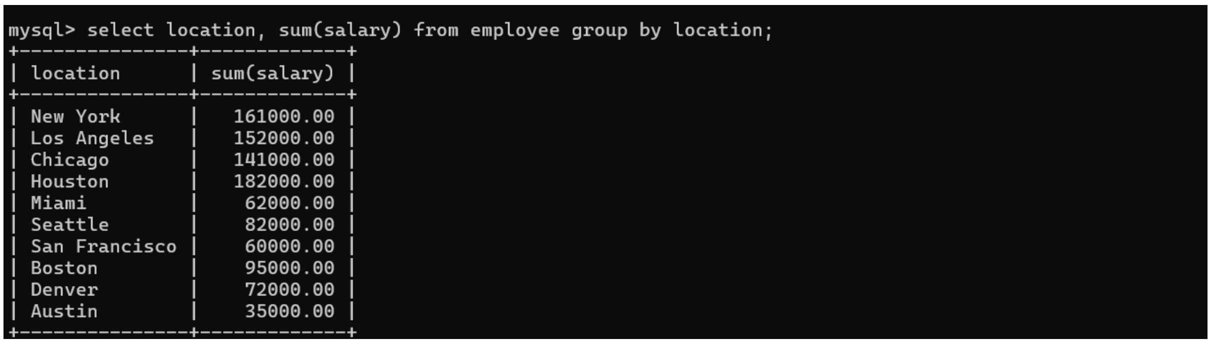
```
mysql> select location, avg(unitprice) from data where sales>200 group by location having avg(unitprice)>190;
+-------------+----------------+
| location    | avg(unitprice) |
+-------------+----------------+
| Toronto     |       221.2500 |
| Tokyo       |       207.5000 |
| Mexico City |       228.3333 |
+-------------+----------------+
```

Employee table:



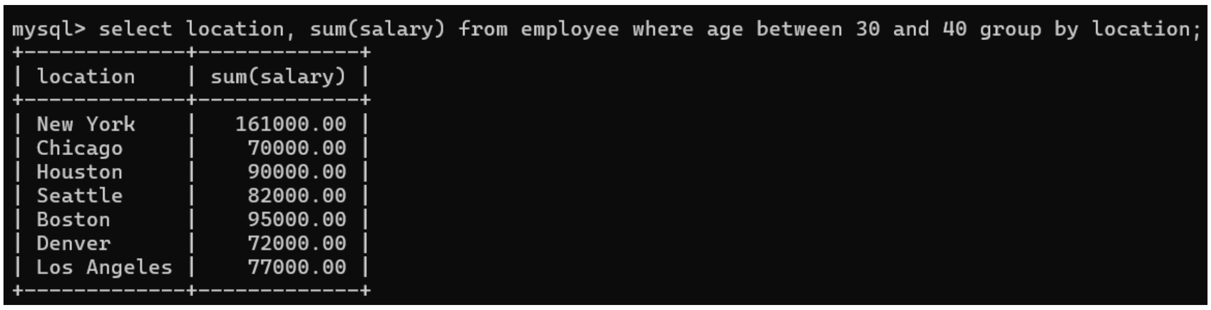
```
mysql> select * from employee;
+-------------+-------------------+--------------+------+---------------------+---------------+----------+
| employee_id | name              | joining_date | age  | role                | location      | salary   |
+-------------+-------------------+--------------+------+---------------------+---------------+----------+
|           1 | John Smith        | 2020-01-15   |   30 | Software Engineer   | New York      | 80000.00 |
|           2 | Jane Do           | 2019-03-22   |   28 | HR Manager          | Los Angeles   | 75000.00 |
|           3 | Alice Johnson     | 2021-06-10   |   35 | Data Analyst        | Chicago       | 70000.00 |
|           4 | Bob Brown         | 2022-02-05   |   40 | Project Manager     | Houston       | 90000.00 |
|           5 | Charlie White     | 2023-05-30   |   26 | Intern              | Miami         | 30000.00 |
|           6 | David Wilson      | 2021-08-12   |   32 | Software Engineer   | Seattle       | 82000.00 |
|           7 | Emily Davis       | 2020-11-20   |   29 | HR Assistant        | San Francisco | 60000.00 |
|           8 | Frank Miller      | 2019-07-15   |   38 | Data Scientist      | Boston        | 95000.00 |
|           9 | Grace Lee         | 2022-03-18   |   31 | Project Coordinator | Denver        | 72000.00 |
|          10 | Henry Garcia      | 2023-01-25   |   27 | Intern              | Austin        | 35000.00 |
|          11 | Isabella Martinez | 2021-04-30   |   34 | Software Engineer   | New York      | 81000.00 |
|          12 | Jack Thompson     | 2020-09-10   |   36 | HR Manager          | Los Angeles   | 77000.00 |
|          13 | Karen Robinson    | 2021-12-05   |   29 | Data Analyst        | Chicago       | 71000.00 |
|          14 | Liam Anderson     | 2022-05-15   |   41 | Project Manager     | Houston       | 92000.00 |
|          15 | Mia Clark         | 2023-03-20   |   25 | Intern              | Miami         | 32000.00 |
+-------------+-------------------+--------------+------+---------------------+---------------+----------+
```

1)Total salary for each location:



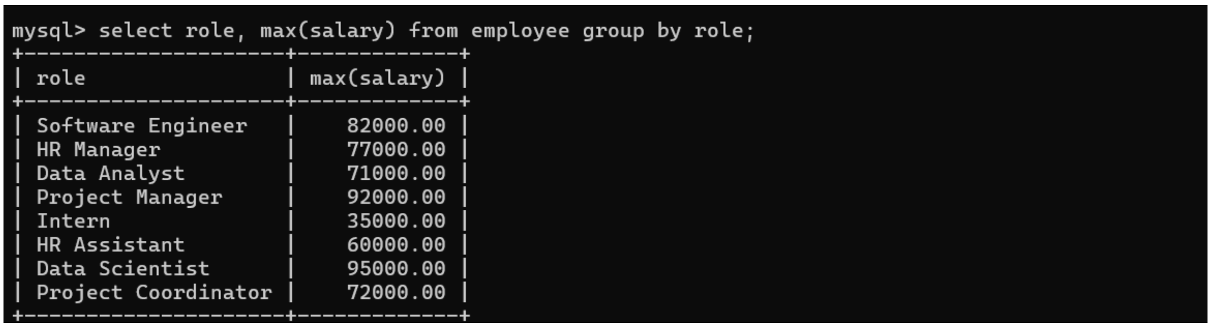
```
mysql> select location, sum(salary) from employee group by location;
+---------------+-------------+
| location      | sum(salary) |
+---------------+-------------+
| New York      |   161000.00 |
| Los Angeles   |   152000.00 |
| Chicago       |   141000.00 |
| Houston       |   182000.00 |
| Miami         |    62000.00 |
| Seattle       |    82000.00 |
| San Francisco |    60000.00 |
| Boston        |    95000.00 |
| Denver        |    72000.00 |
| Austin        |    35000.00 |
+---------------+-------------+
```

2)Total salary for each location for age from 30 to 40:



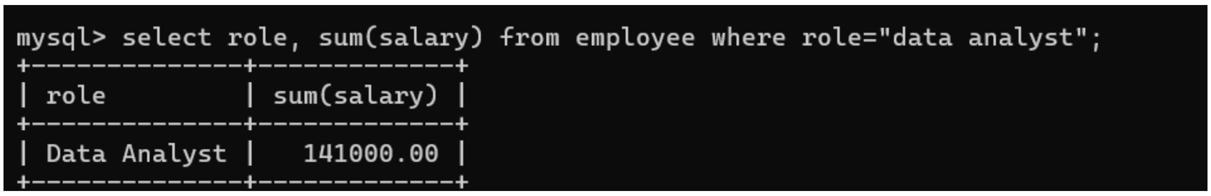
```
mysql> select location, sum(salary) from employee where age between 30 and 40 group by location;
+-------------+-------------+
| location    | sum(salary) |
+-------------+-------------+
| New York    |   161000.00 |
| Chicago     |    70000.00 |
| Houston     |    90000.00 |
| Seattle     |    82000.00 |
| Boston      |    95000.00 |
| Denver      |    72000.00 |
| Los Angeles |    77000.00 |
+-------------+-------------+
```

3)Maximum salary paid for each role:



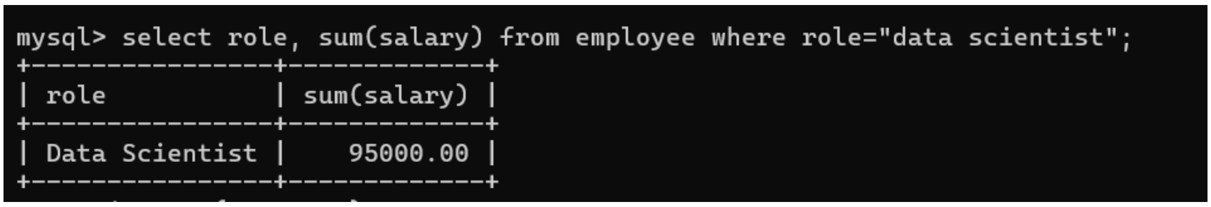
```
mysql> select role, max(salary) from employee group by role;
+---------------------+-------------+
| role                | max(salary) |
+---------------------+-------------+
| Software Engineer   |    82000.00 |
| HR Manager          |    77000.00 |
| Data Analyst        |    71000.00 |
| Project Manager     |    92000.00 |
| Intern              |    35000.00 |
| HR Assistant        |    60000.00 |
| Data Scientist      |    95000.00 |
| Project Coordinator |    72000.00 |
+---------------------+-------------+
```

4)Total salary for Data Analyst:



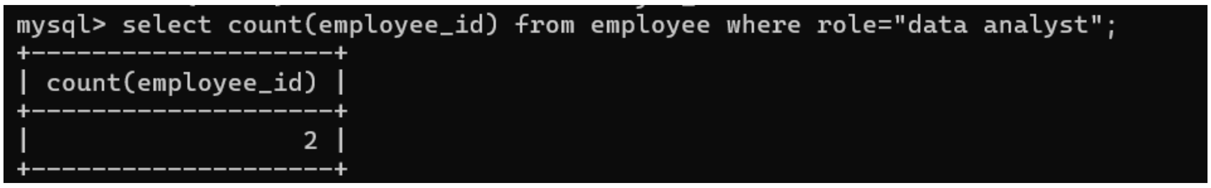
```
mysql> select role, sum(salary) from employee where role="data analyst";
+--------------+-------------+
| role         | sum(salary) |
+--------------+-------------+
| Data Analyst |   141000.00 |
+--------------+-------------+
```

5)Total salary for Data Scientist:



```
mysql> select role, sum(salary) from employee where role="data scientist";
+----------------+-------------+
| role           | sum(salary) |
+----------------+-------------+
| Data Scientist |    95000.00 |
+----------------+-------------+
```

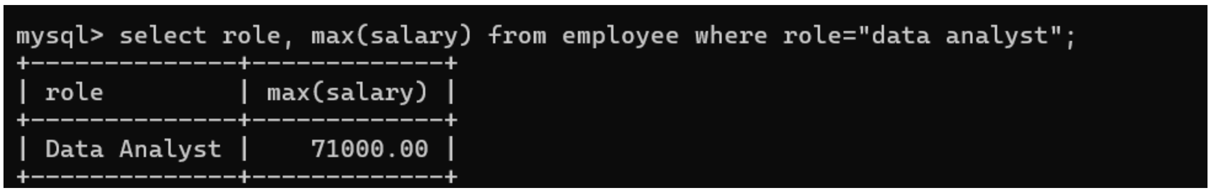
6)How many employees working as data analyst:



```
mysql> select count(employee_id) from employee where role="data analyst";
+--------------------+
| count(employee_id) |
+--------------------+
|                  2 |
+--------------------+
```

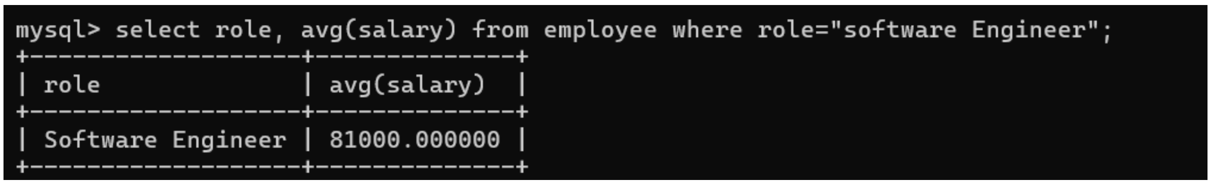
Task

1)Maximum salary paying for data analyst:



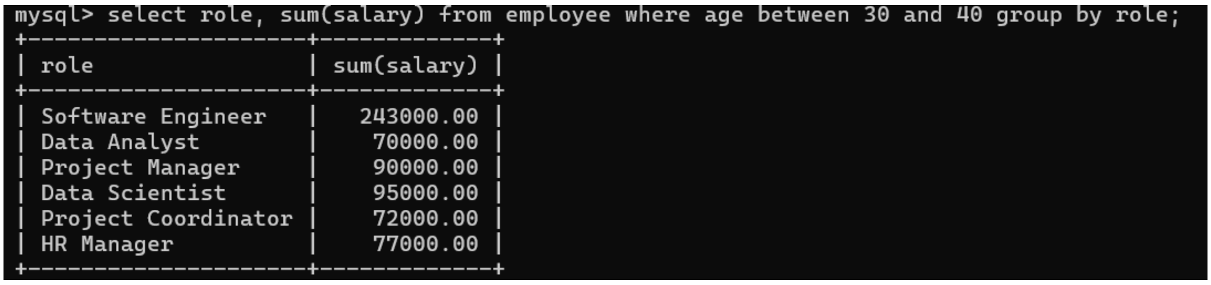
```
mysql> select role, max(salary) from employee where role="data analyst";
+--------------+-------------+
| role         | max(salary) |
+--------------+-------------+
| Data Analyst |    71000.00 |
+--------------+-------------+
```

2)Average salary paying for software engineer:



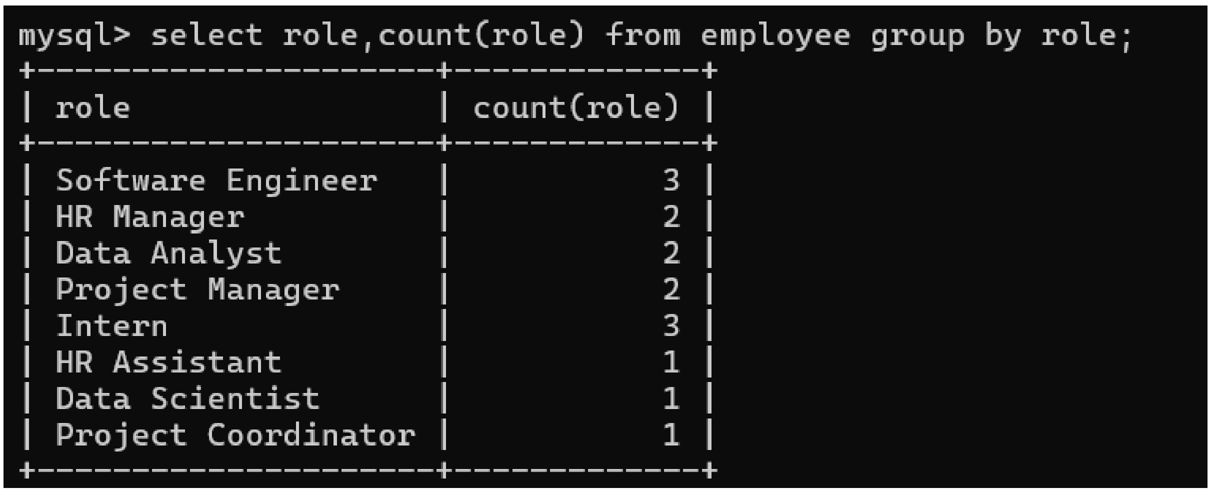
```
mysql> select role, avg(salary) from employee where role="software Engineer";
+-------------------+--------------+
| role              | avg(salary)  |
+-------------------+--------------+
| Software Engineer | 81000.000000 |
+-------------------+--------------+
```

3)Total salary paying for each role for age 30 to 40:



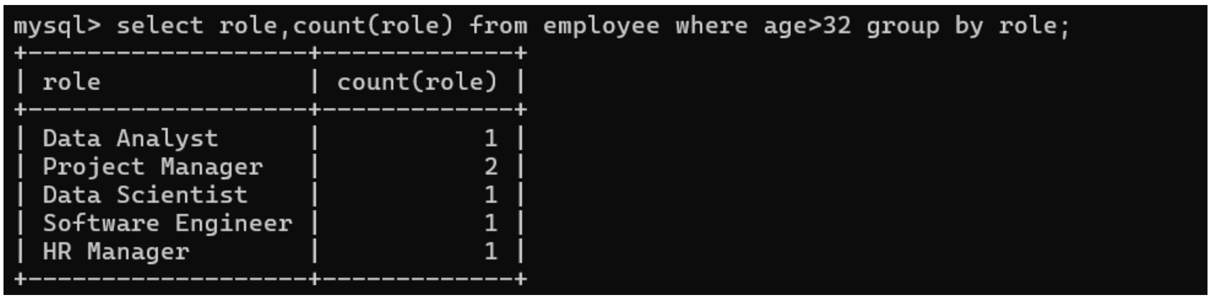
```
mysql> select role, sum(salary) from employee where age between 30 and 40 group by role;
+---------------------+-------------+
| role                | sum(salary) |
+---------------------+-------------+
| Software Engineer   |   243000.00 |
| Data Analyst        |    70000.00 |
| Project Manager     |    90000.00 |
| Data Scientist      |    95000.00 |
| Project Coordinator |    72000.00 |
| HR Manager          |    77000.00 |
+---------------------+-------------+
```

4)How many employees working under each role?



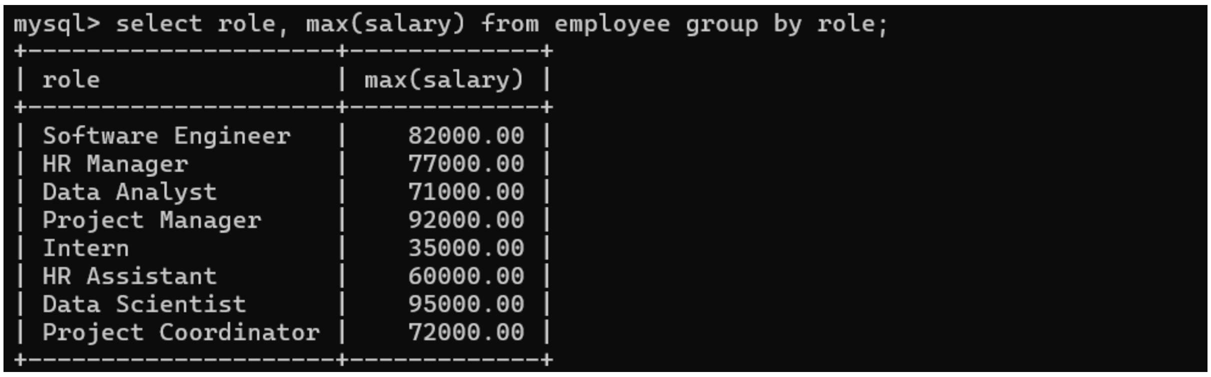
```
mysql> select role,count(role) from employee group by role;
+---------------------+-------------+
| role                | count(role) |
+---------------------+-------------+
| Software Engineer   |           3 |
| HR Manager          |           2 |
| Data Analyst        |           2 |
| Project Manager     |           2 |
| Intern              |           3 |
| HR Assistant        |           1 |
| Data Scientist      |           1 |
| Project Coordinator |           1 |
+---------------------+-------------+
```

5) How many employees working under each role for age above 32?



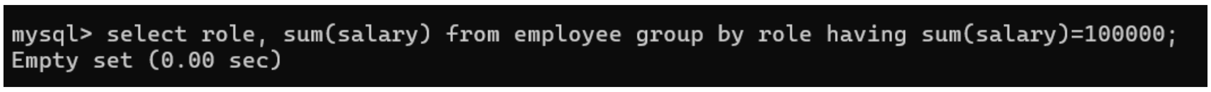
```
mysql> select role,count(role) from employee where age>32 group by role;
+-------------------+-------------+
| role              | count(role) |
+-------------------+-------------+
| Data Analyst      |           1 |
| Project Manager   |           2 |
| Data Scientist    |           1 |
| Software Engineer |           1 |
| HR Manager        |           1 |
+-------------------+-------------+
```

6)Maximum salary paying for each role:



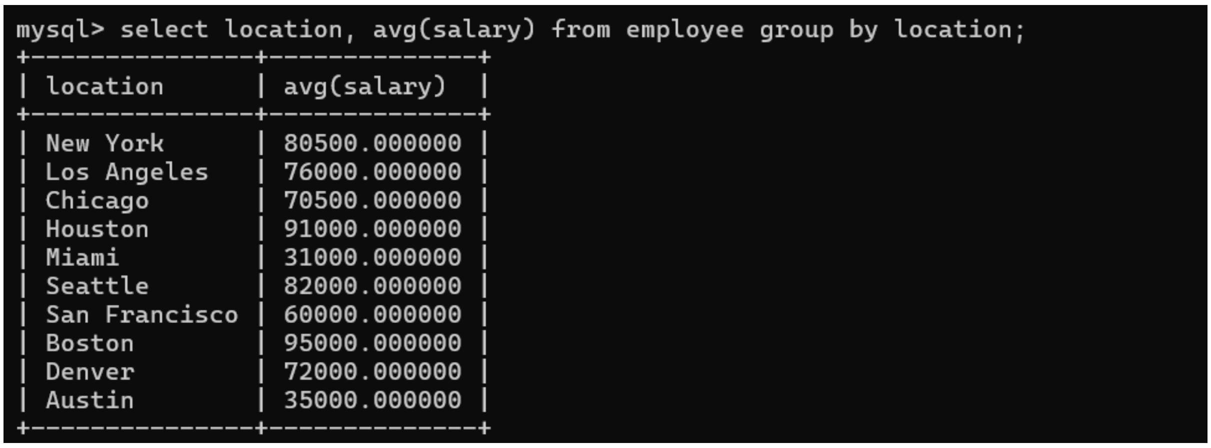
```
mysql> select role, max(salary) from employee group by role;
+---------------------+-------------+
| role                | max(salary) |
+---------------------+-------------+
| Software Engineer   |    82000.00 |
| HR Manager          |    77000.00 |
| Data Analyst        |    71000.00 |
| Project Manager     |    92000.00 |
| Intern              |    35000.00 |
| HR Assistant        |    60000.00 |
| Data Scientist      |    95000.00 |
| Project Coordinator |    72000.00 |
+---------------------+-------------+
```

7)Total salary for each role whose total salary is 1 lakh:



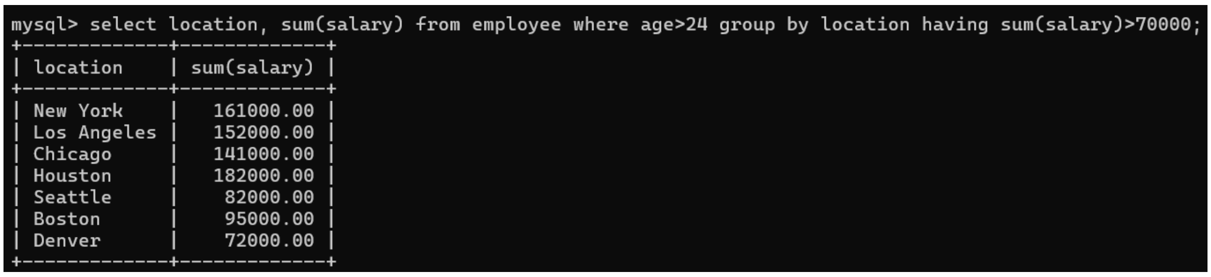
```
mysql> select role, sum(salary) from employee group by role having sum(salary)=100000;
Empty set (0.00 sec)
```

8)Average salary paying for each location:



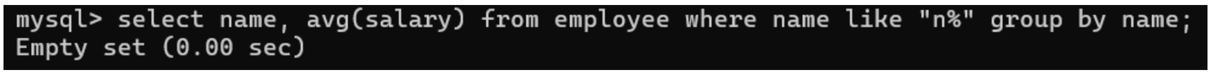
```
mysql> select location, avg(salary) from employee group by location;
+---------------+--------------+
| location      | avg(salary)  |
+---------------+--------------+
| New York      | 80500.000000 |
| Los Angeles   | 76000.000000 |
| Chicago       | 70500.000000 |
| Houston       | 91000.000000 |
| Miami         | 31000.000000 |
| Seattle       | 82000.000000 |
| San Francisco | 60000.000000 |
| Boston        | 95000.000000 |
| Denver        | 72000.000000 |
| Austin        | 35000.000000 |
+---------------+--------------+
```

9)Total salary for each location for age above 24 total salary above 70000:



```
mysql> select location, sum(salary) from employee where age>24 group by location having sum(salary)>70000;
+-------------+-------------+
| location    | sum(salary) |
+-------------+-------------+
| New York    |   161000.00 |
| Los Angeles |   152000.00 |
| Chicago     |   141000.00 |
| Houston     |   182000.00 |
| Seattle     |    82000.00 |
| Boston      |    95000.00 |
| Denver      |    72000.00 |
+-------------+-------------+
```

10)Average salary for employee name start with n:



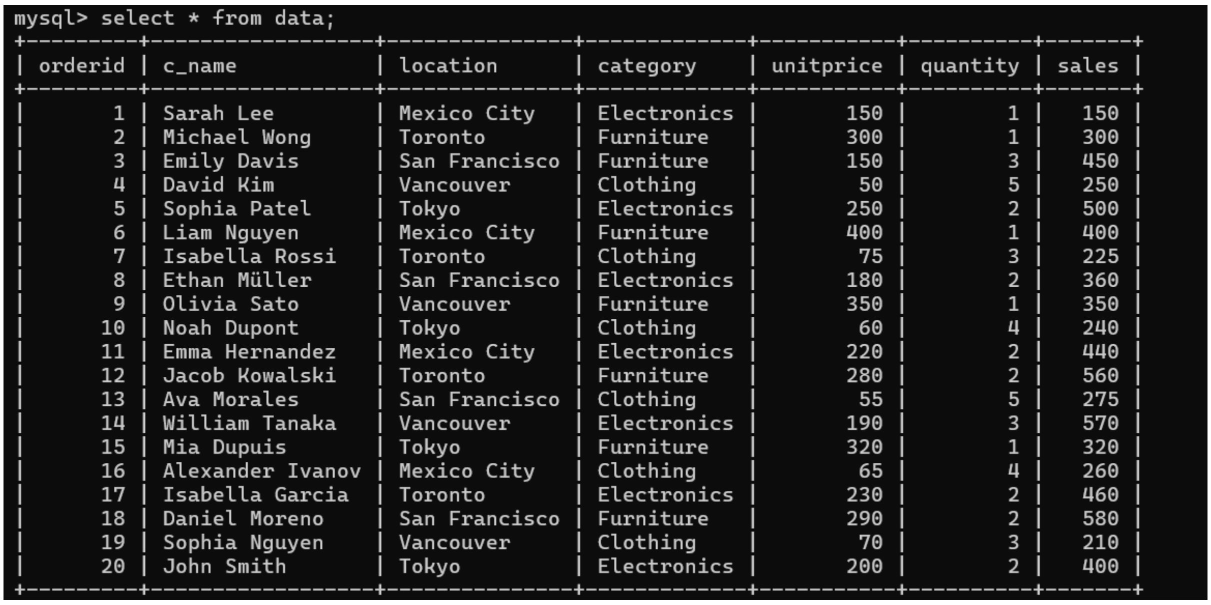
```
mysql> select name, avg(salary) from employee where name like "n%" group by name;
Empty set (0.00 sec)
```

DAY 4

ORDER BY:

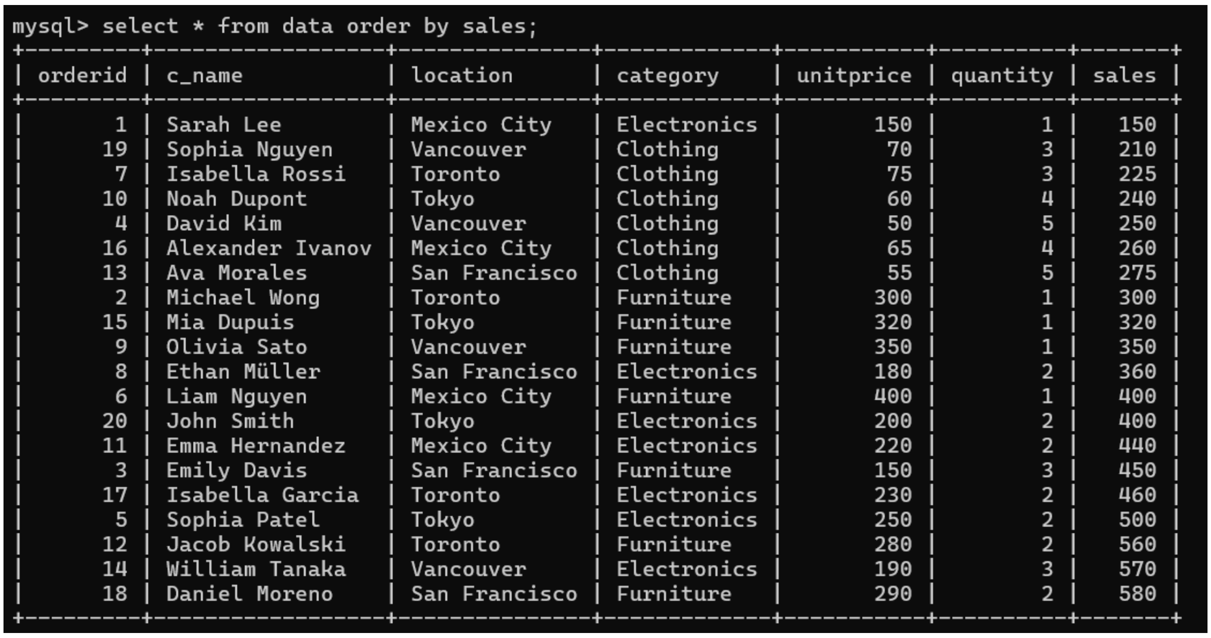
Arranging in ascending or descending order

Viewing the Table:



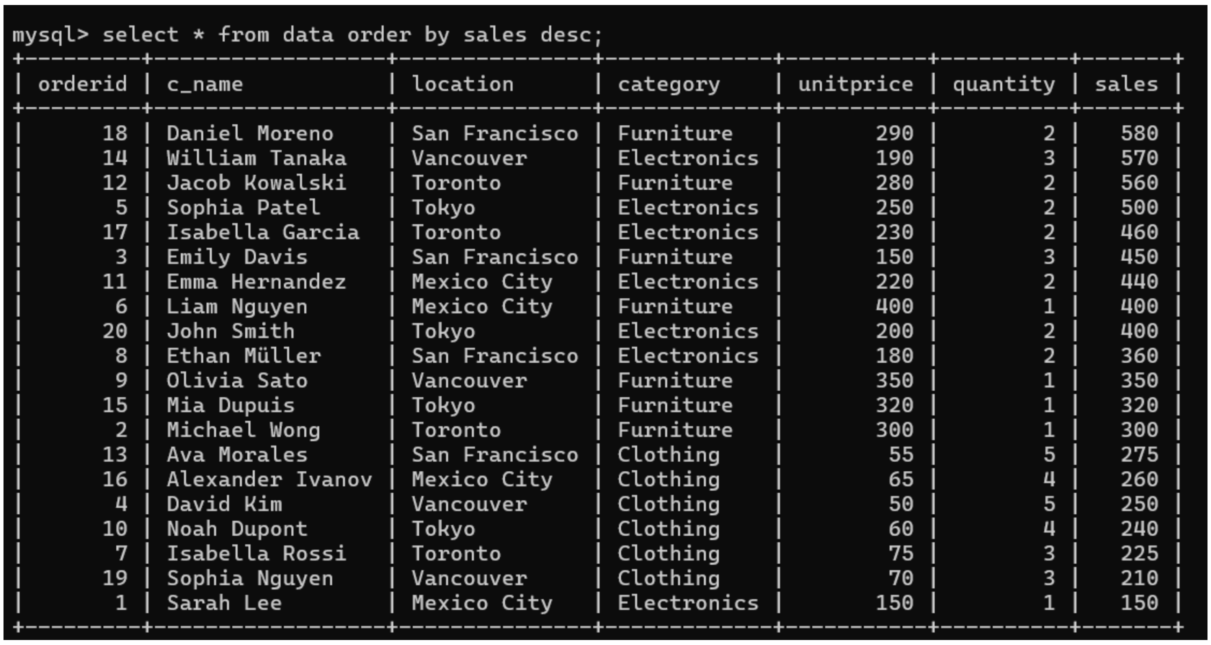
```
mysql> select * from data;
+---------+------------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name           | location      | category    | unitprice | quantity | sales |
+---------+------------------+---------------+-------------+-----------+----------+-------+
|       1 | Sarah Lee        | Mexico City   | Electronics |       150 |        1 |   150 |
|       2 | Michael Wong     | Toronto       | Furniture   |       300 |        1 |   300 |
|       3 | Emily Davis      | San Francisco | Furniture   |       150 |        3 |   450 |
|       4 | David Kim        | Vancouver     | Clothing    |        50 |        5 |   250 |
|       5 | Sophia Patel     | Tokyo         | Electronics |       250 |        2 |   500 |
|       6 | Liam Nguyen      | Mexico City   | Furniture   |       400 |        1 |   400 |
|       7 | Isabella Rossi   | Toronto       | Clothing    |        75 |        3 |   225 |
|       8 | Ethan Müller     | San Francisco | Electronics |       180 |        2 |   360 |
|       9 | Olivia Sato      | Vancouver     | Furniture   |       350 |        1 |   350 |
|      10 | Noah Dupont      | Tokyo         | Clothing    |        60 |        4 |   240 |
|      11 | Emma Hernandez   | Mexico City   | Electronics |       220 |        2 |   440 |
|      12 | Jacob Kowalski   | Toronto       | Furniture   |       280 |        2 |   560 |
|      13 | Ava Morales      | San Francisco | Clothing    |        55 |        5 |   275 |
|      14 | William Tanaka   | Vancouver     | Electronics |       190 |        3 |   570 |
|      15 | Mia Dupuis       | Tokyo         | Furniture   |       320 |        1 |   320 |
|      16 | Alexander Ivanov | Mexico City   | Clothing    |        65 |        4 |   260 |
|      17 | Isabella Garcia  | Toronto       | Electronics |       230 |        2 |   460 |
|      18 | Daniel Moreno    | San Francisco | Furniture   |       290 |        2 |   580 |
|      19 | Sophia Nguyen    | Vancouver     | Clothing    |        70 |        3 |   210 |
|      20 | John Smith       | Tokyo         | Electronics |       200 |        2 |   400 |
+---------+------------------+---------------+-------------+-----------+----------+-------+
```

Ascending Order:



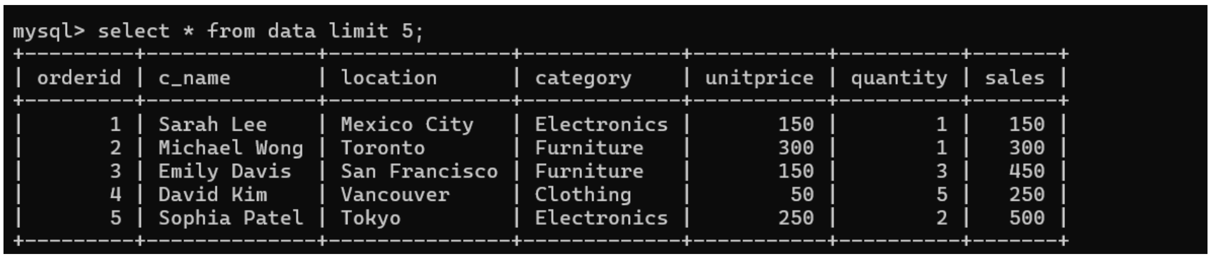
```
mysql> select * from data order by sales;
+---------+------------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name           | location      | category    | unitprice | quantity | sales |
+---------+------------------+---------------+-------------+-----------+----------+-------+
|       1 | Sarah Lee        | Mexico City   | Electronics |       150 |        1 |   150 |
|      19 | Sophia Nguyen    | Vancouver     | Clothing    |        70 |        3 |   210 |
|       7 | Isabella Rossi   | Toronto       | Clothing    |        75 |        3 |   225 |
|      10 | Noah Dupont      | Tokyo         | Clothing    |        60 |        4 |   240 |
|       4 | David Kim        | Vancouver     | Clothing    |        50 |        5 |   250 |
|      16 | Alexander Ivanov | Mexico City   | Clothing    |        65 |        4 |   260 |
|      13 | Ava Morales      | San Francisco | Clothing    |        55 |        5 |   275 |
|       2 | Michael Wong     | Toronto       | Furniture   |       300 |        1 |   300 |
|      15 | Mia Dupuis       | Tokyo         | Furniture   |       320 |        1 |   320 |
|       9 | Olivia Sato      | Vancouver     | Furniture   |       350 |        1 |   350 |
|       8 | Ethan Müller     | San Francisco | Electronics |       180 |        2 |   360 |
|       6 | Liam Nguyen      | Mexico City   | Furniture   |       400 |        1 |   400 |
|      20 | John Smith       | Tokyo         | Electronics |       200 |        2 |   400 |
|      11 | Emma Hernandez   | Mexico City   | Electronics |       220 |        2 |   440 |
|       3 | Emily Davis      | San Francisco | Furniture   |       150 |        3 |   450 |
|      17 | Isabella Garcia  | Toronto       | Electronics |       230 |        2 |   460 |
|       5 | Sophia Patel     | Tokyo         | Electronics |       250 |        2 |   500 |
|      12 | Jacob Kowalski   | Toronto       | Furniture   |       280 |        2 |   560 |
|      14 | William Tanaka   | Vancouver     | Electronics |       190 |        3 |   570 |
|      18 | Daniel Moreno    | San Francisco | Furniture   |       290 |        2 |   580 |
+---------+------------------+---------------+-------------+-----------+----------+-------+
```

Descending Order:



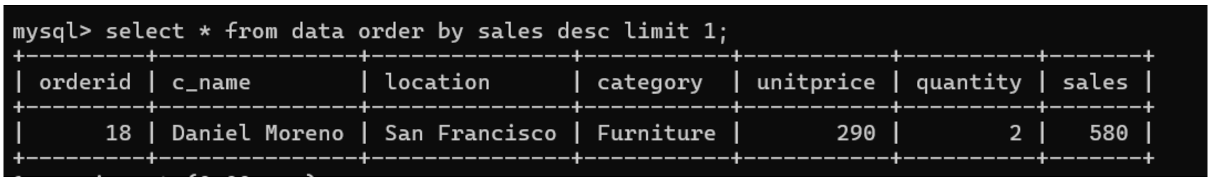
```
mysql> select * from data order by sales desc;
+---------+------------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name           | location      | category    | unitprice | quantity | sales |
+---------+------------------+---------------+-------------+-----------+----------+-------+
|      18 | Daniel Moreno    | San Francisco | Furniture   |       290 |        2 |   580 |
|      14 | William Tanaka   | Vancouver     | Electronics |       190 |        3 |   570 |
|      12 | Jacob Kowalski   | Toronto       | Furniture   |       280 |        2 |   560 |
|       5 | Sophia Patel     | Tokyo         | Electronics |       250 |        2 |   500 |
|      17 | Isabella Garcia  | Toronto       | Electronics |       230 |        2 |   460 |
|       3 | Emily Davis      | San Francisco | Furniture   |       150 |        3 |   450 |
|      11 | Emma Hernandez   | Mexico City   | Electronics |       220 |        2 |   440 |
|       6 | Liam Nguyen      | Mexico City   | Furniture   |       400 |        1 |   400 |
|      20 | John Smith       | Tokyo         | Electronics |       200 |        2 |   400 |
|       8 | Ethan Müller     | San Francisco | Electronics |       180 |        2 |   360 |
|       9 | Olivia Sato      | Vancouver     | Furniture   |       350 |        1 |   350 |
|      15 | Mia Dupuis       | Tokyo         | Furniture   |       320 |        1 |   320 |
|       2 | Michael Wong     | Toronto       | Furniture   |       300 |        1 |   300 |
|      13 | Ava Morales      | San Francisco | Clothing    |        55 |        5 |   275 |
|      16 | Alexander Ivanov | Mexico City   | Clothing    |        65 |        4 |   260 |
|       4 | David Kim        | Vancouver     | Clothing    |        50 |        5 |   250 |
|      10 | Noah Dupont      | Tokyo         | Clothing    |        60 |        4 |   240 |
|       7 | Isabella Rossi   | Toronto       | Clothing    |        75 |        3 |   225 |
|      19 | Sophia Nguyen    | Vancouver     | Clothing    |        70 |        3 |   210 |
|       1 | Sarah Lee        | Mexico City   | Electronics |       150 |        1 |   150 |
+---------+------------------+---------------+-------------+-----------+----------+-------+
```

LIMIT:



```
mysql> select * from data limit 5;
+---------+--------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name       | location      | category    | unitprice | quantity | sales |
+---------+--------------+---------------+-------------+-----------+----------+-------+
|       1 | Sarah Lee    | Mexico City   | Electronics |       150 |        1 |   150 |
|       2 | Michael Wong | Toronto       | Furniture   |       300 |        1 |   300 |
|       3 | Emily Davis  | San Francisco | Furniture   |       150 |        3 |   450 |
|       4 | David Kim    | Vancouver     | Clothing    |        50 |        5 |   250 |
|       5 | Sophia Patel | Tokyo         | Electronics |       250 |        2 |   500 |
+---------+--------------+---------------+-------------+-----------+----------+-------+
```

1)Sales in descending order with limit 1:

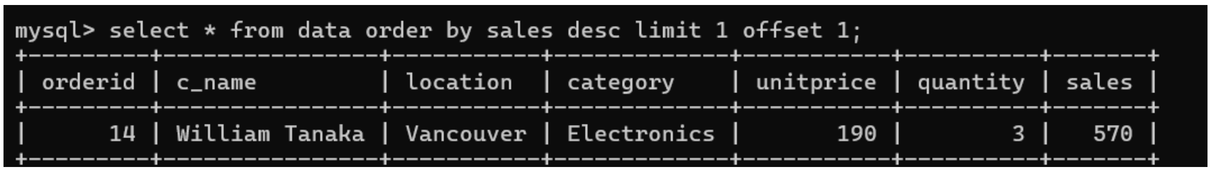


```
mysql> select * from data order by sales desc limit 1;
+---------+--------------+---------------+-----------+-----------+----------+-------+
| orderid | c_name       | location      | category  | unitprice | quantity | sales |
+---------+--------------+---------------+-----------+-----------+----------+-------+
|      18 | Daniel Moreno | San Francisco | Furniture |       290 |        2 |   580 |
+---------+--------------+---------------+-----------+-----------+----------+-------+
```

OFFSET:

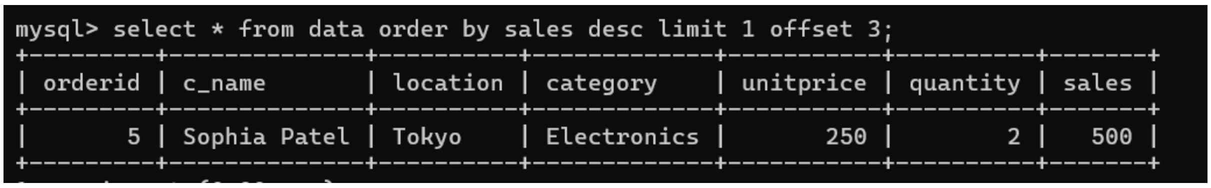
Removing above records:

1)Which Location has highest total sales regarding quantity?



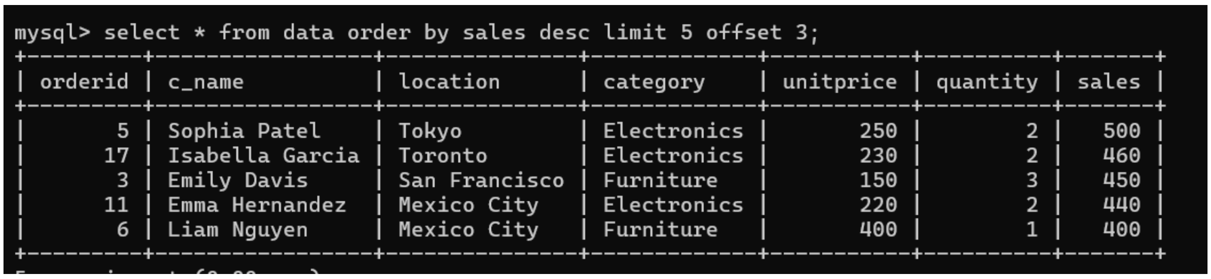
```
mysql> select * from data order by sales desc limit 1 offset 1;
+---------+----------------+-----------+-------------+-----------+----------+-------+
| orderid | c_name         | location  | category    | unitprice | quantity | sales |
+---------+----------------+-----------+-------------+-----------+----------+-------+
|      14 | William Tanaka | Vancouver | Electronics |       190 |        3 |   570 |
+---------+----------------+-----------+-------------+-----------+----------+-------+
```

2) Which Location has second highest total sales regarding quantity?



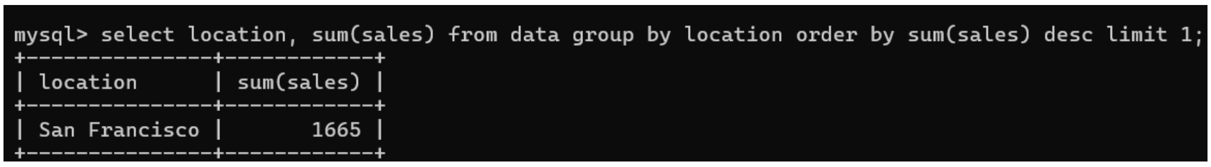
```
mysql> select * from data order by sales desc limit 1 offset 3;
+---------+-------------+----------+-------------+-----------+----------+-------+
| orderid | c_name      | location | category    | unitprice | quantity | sales |
+---------+-------------+----------+-------------+-----------+----------+-------+
|       5 | Sophia Patel | Tokyo    | Electronics |       250 |        2 |   500 |
+---------+-------------+----------+-------------+-----------+----------+-------+
```

3)Limit 5 and offset 3:



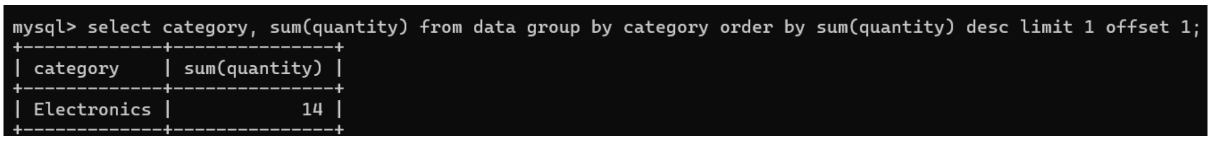
```
mysql> select * from data order by sales desc limit 5 offset 3;
+---------+-----------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name          | location      | category    | unitprice | quantity | sales |
+---------+-----------------+---------------+-------------+-----------+----------+-------+
|       5 | Sophia Patel    | Tokyo         | Electronics |       250 |        2 |   500 |
|      17 | Isabella Garcia | Toronto       | Electronics |       230 |        2 |   460 |
|       3 | Emily Davis     | San Francisco | Furniture   |       150 |        3 |   450 |
|      11 | Emma Hernandez  | Mexico City   | Electronics |       220 |        2 |   440 |
|       6 | Liam Nguyen     | Mexico City   | Furniture   |       400 |        1 |   400 |
+---------+-----------------+---------------+-------------+-----------+----------+-------+
```

4)Highest total sales for location:



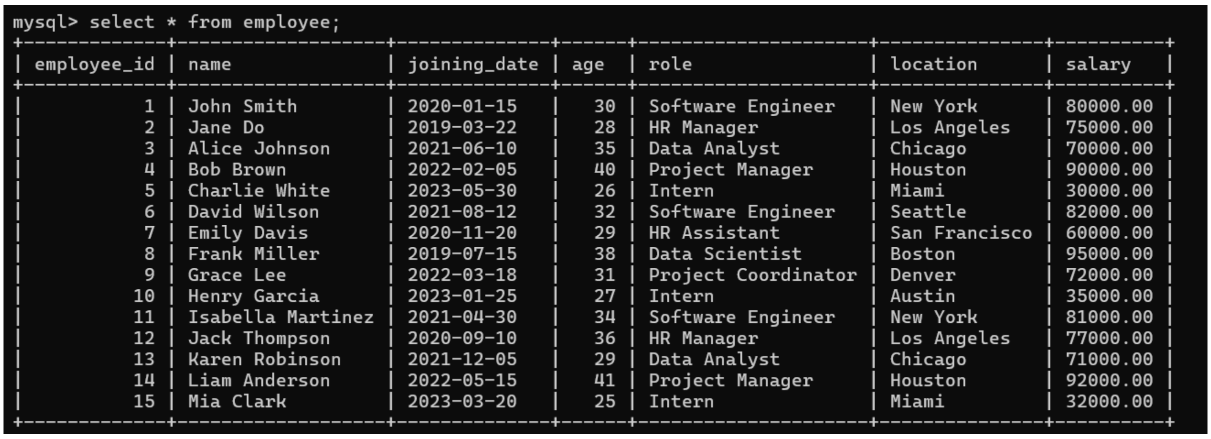
```
mysql> select location, sum(sales) from data group by location order by sum(sales) desc limit 1;
+---------------+------------+
| location      | sum(sales) |
+---------------+------------+
| San Francisco |       1665 |
+---------------+------------+
```

5)Second highest total quantity for category:



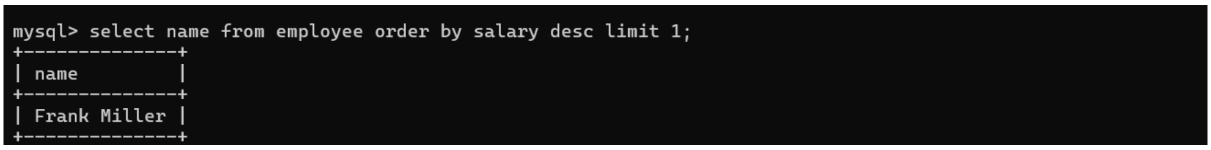
```
mysql> select category, sum(quantity) from data group by category order by sum(quantity) desc limit 1 offset 1;
+-------------+---------------+
| category    | sum(quantity) |
+-------------+---------------+
| Electronics |            14 |
+-------------+---------------+
```

Employee Table:



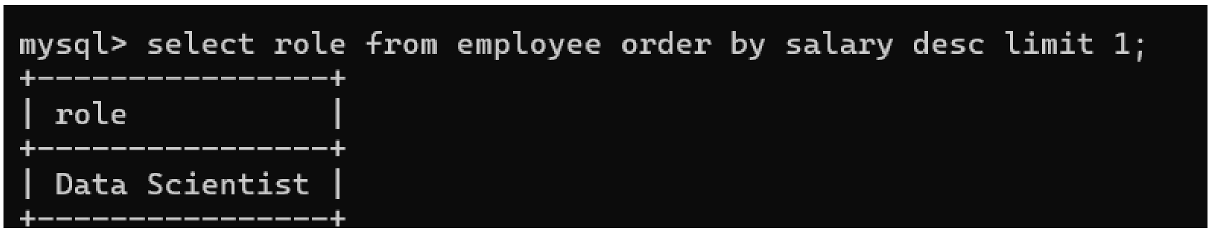
```
mysql> select * from employee;
+-------------+-------------------+--------------+-----+---------------------+---------------+----------+
| employee_id | name              | joining_date | age | role                | location      | salary   |
+-------------+-------------------+--------------+-----+---------------------+---------------+----------+
|           1 | John Smith        | 2020-01-15   |  30 | Software Engineer   | New York      | 80000.00 |
|           2 | Jane Do           | 2019-03-22   |  28 | HR Manager          | Los Angeles   | 75000.00 |
|           3 | Alice Johnson     | 2021-06-10   |  35 | Data Analyst        | Chicago       | 70000.00 |
|           4 | Bob Brown         | 2022-02-05   |  40 | Project Manager     | Houston       | 90000.00 |
|           5 | Charlie White     | 2023-05-30   |  26 | Intern              | Miami         | 30000.00 |
|           6 | David Wilson      | 2021-08-12   |  32 | Software Engineer   | Seattle       | 82000.00 |
|           7 | Emily Davis       | 2020-11-20   |  29 | HR Assistant        | San Francisco | 60000.00 |
|           8 | Frank Miller      | 2019-07-15   |  38 | Data Scientist      | Boston        | 95000.00 |
|           9 | Grace Lee         | 2022-03-18   |  31 | Project Coordinator | Denver        | 72000.00 |
|          10 | Henry Garcia      | 2023-01-25   |  27 | Intern              | Austin        | 35000.00 |
|          11 | Isabella Martinez | 2021-04-30   |  34 | Software Engineer   | New York      | 81000.00 |
|          12 | Jack Thompson     | 2020-09-10   |  36 | HR Manager          | Los Angeles   | 77000.00 |
|          13 | Karen Robinson    | 2021-12-05   |  29 | Data Analyst        | Chicago       | 71000.00 |
|          14 | Liam Anderson     | 2022-05-15   |  41 | Project Manager     | Houston       | 92000.00 |
|          15 | Mia Clark         | 2023-03-20   |  25 | Intern              | Miami         | 32000.00 |
+-------------+-------------------+--------------+-----+---------------------+---------------+----------+
```

1)Find which employee receiving highest salary?



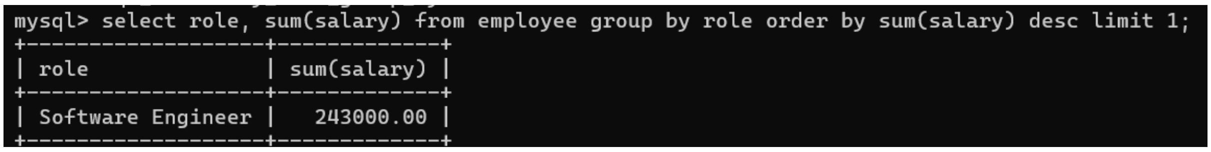
```
mysql> select name from employee order by salary desc limit 1;
+--------------+
| name         |
+--------------+
| Frank Miller |
+--------------+
```

2)Which role receiving highest salary?



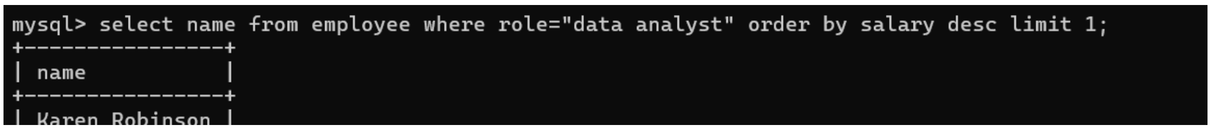
```
mysql> select role from employee order by salary desc limit 1;
+----------------+
| role           |
+----------------+
| Data Scientist |
+----------------+
```

3)Which role has highest total salary?



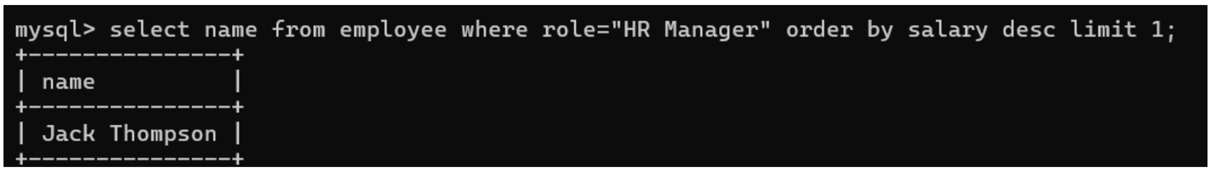
```
mysql> select role, sum(salary) from employee group by role order by sum(salary) desc limit 1;
+-------------------+-------------+
| role              | sum(salary) |
+-------------------+-------------+
| Software Engineer |   243000.00 |
+-------------------+-------------+
```

4)Which employee has highest salary in data analyst?



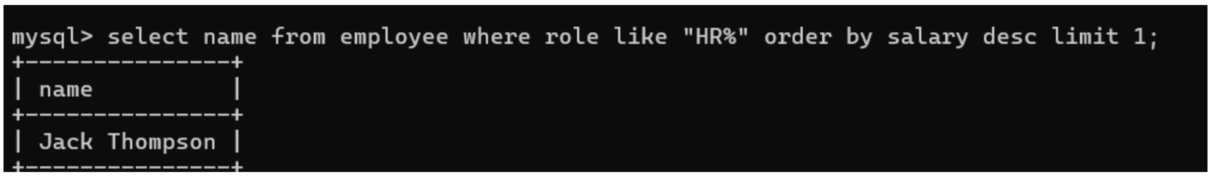
```
mysql> select name from employee where role="data analyst" order by salary desc limit 1;
+----------------+
| name           |
+----------------+
| Karen Robinson |
```

5)Which employee has highest salary in HR Manager?



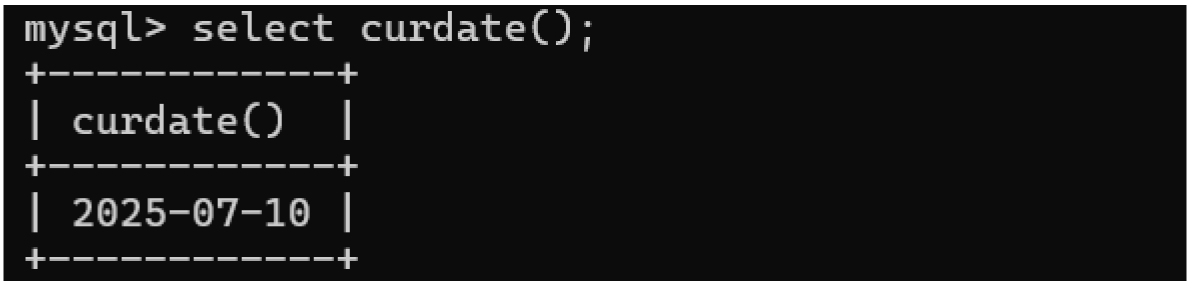
```
mysql> select name from employee where role="HR Manager" order by salary desc limit 1;
+---------------+
| name          |
+---------------+
| Jack Thompson |
+---------------+
```

5)Which employee has highest salary in HR Department?

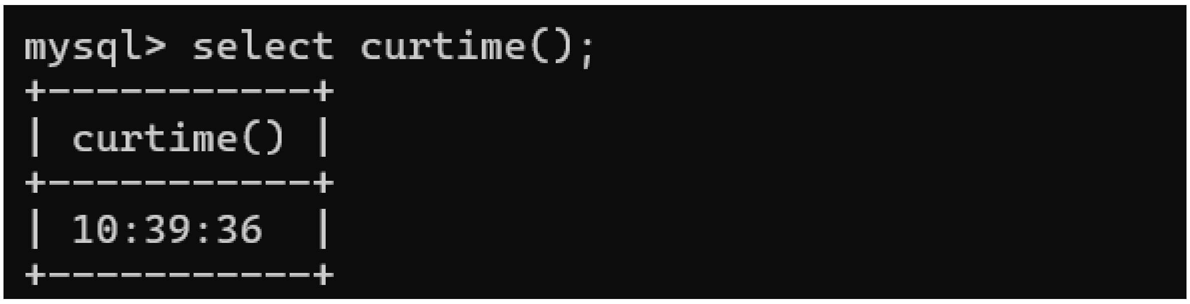

```
mysql> select name from employee where role like "HR%" order by salary desc limit 1;
+---------------+
| name          |
+---------------+
| Jack Thompson |
+---------------+
```

DATE FUNCTION:

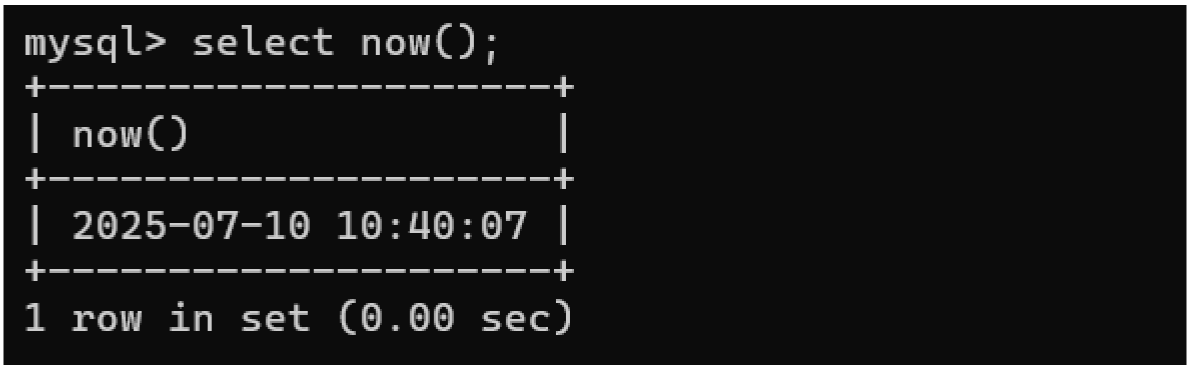
To find date:


```
mysql> select curdate();
+------------+
| curdate()  |
+------------+
| 2025-07-10 |
+------------+
```

To find time:


```
mysql> select curtime();
+-----------+
| curtime() |
+-----------+
| 10:39:36  |
+-----------+
```

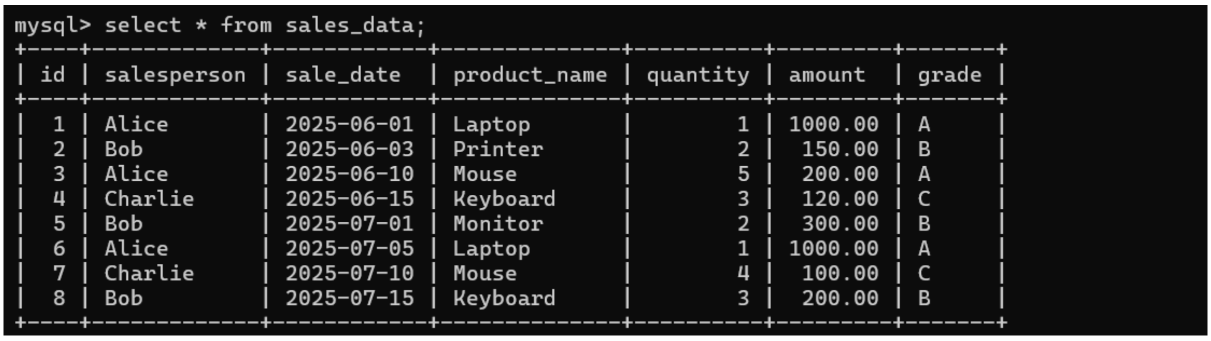
To find date and time:


```
mysql> select now();
+---------------------+
| now()               |
+---------------------+
| 2025-07-10 10:40:07 |
+---------------------+
1 row in set (0.00 sec)
```

TASK-3:

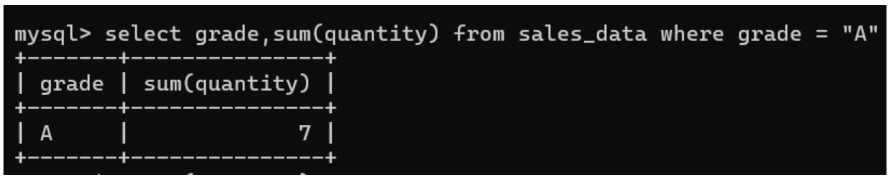
```
CREATE TABLE sales_data (
    id INT PRIMARY KEY,
    salesperson VARCHAR(50),
    sale_date DATE,
    product_name VARCHAR(100),
    quantity INT,
    amount DECIMAL(10, 2),
    grade CHAR(1)
);
INSERT INTO sales_data (id, salesperson, sale_date, product_name, quantity, amount, grade)
VALUES
(1, 'Alice',   '2025-06-01', 'Laptop',     1, 1000.00, 'A'),
(2, 'Bob',     '2025-06-03', 'Printer',    2,  150.00, 'B'),
(3, 'Alice',   '2025-06-10', 'Mouse',      5,  200.00, 'A'),
(4, 'Charlie', '2025-06-15', 'Keyboard',   3,  120.00, 'C'),
(5, 'Bob',     '2025-07-01', 'Monitor',    2,  300.00, 'B'),
(6, 'Alice',   '2025-07-05', 'Laptop',     1, 1000.00, 'A'),
(7, 'Charlie', '2025-07-10', 'Mouse',      4,  100.00, 'C'),
(8, 'Bob',     '2025-07-15', 'Keyboard',   3,  200.00, 'B');
```

Viewing the table:



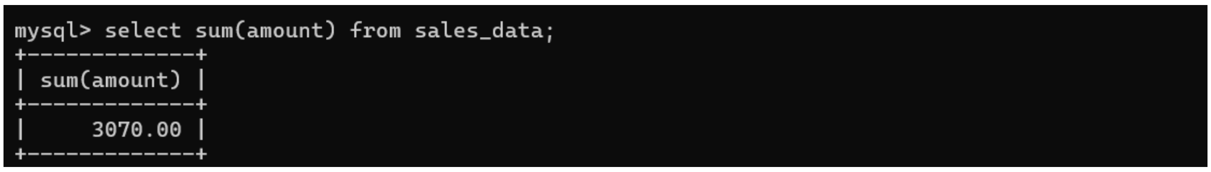
```
mysql> select * from sales_data;
+----+-------------+------------+--------------+----------+---------+-------+
| id | salesperson | sale_date  | product_name | quantity | amount  | grade |
+----+-------------+------------+--------------+----------+---------+-------+
|  1 | Alice       | 2025-06-01 | Laptop       |        1 | 1000.00 | A     |
|  2 | Bob         | 2025-06-03 | Printer      |        2 |  150.00 | B     |
|  3 | Alice       | 2025-06-10 | Mouse        |        5 |  200.00 | A     |
|  4 | Charlie     | 2025-06-15 | Keyboard     |        3 |  120.00 | C     |
|  5 | Bob         | 2025-07-01 | Monitor      |        2 |  300.00 | B     |
|  6 | Alice       | 2025-07-05 | Laptop       |        1 | 1000.00 | A     |
|  7 | Charlie     | 2025-07-10 | Mouse        |        4 |  100.00 | C     |
|  8 | Bob         | 2025-07-15 | Keyboard     |        3 |  200.00 | B     |
+----+-------------+------------+--------------+----------+---------+-------+
```

1)Total quantity sold out in grade A:



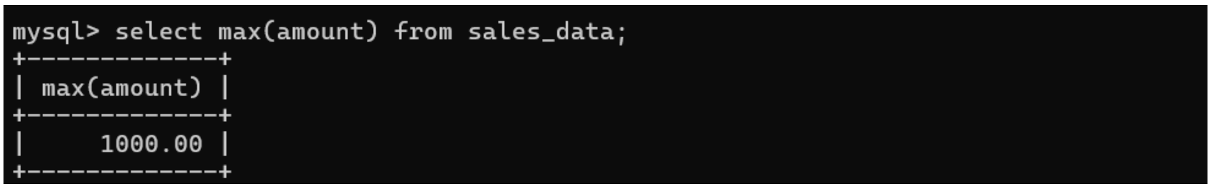
```
mysql> select grade,sum(quantity) from sales_data where grade = "A"
+-------+---------------+
| grade | sum(quantity) |
+-------+---------------+
| A     |             7 |
+-------+---------------+
```

2)Total amount sold out:



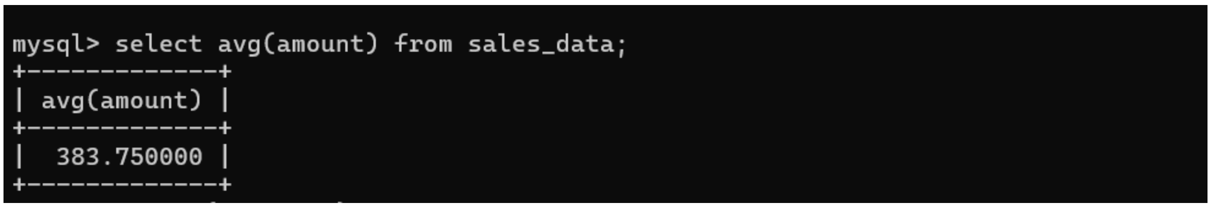
```
mysql> select sum(amount) from sales_data;
+-------------+
| sum(amount) |
+-------------+
|     3070.00 |
+-------------+
```

3)Maximum amount sold out:



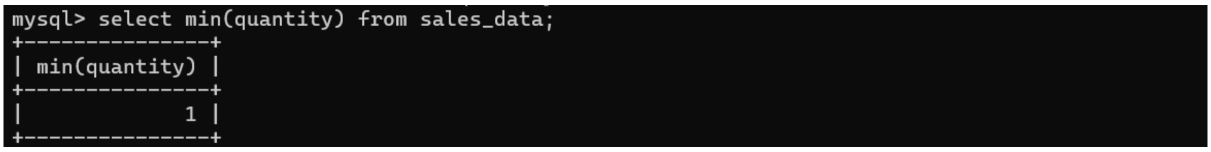
```
mysql> select max(amount) from sales_data;
+-------------+
| max(amount) |
+-------------+
|     1000.00 |
+-------------+
```

4)Average amount sold out:



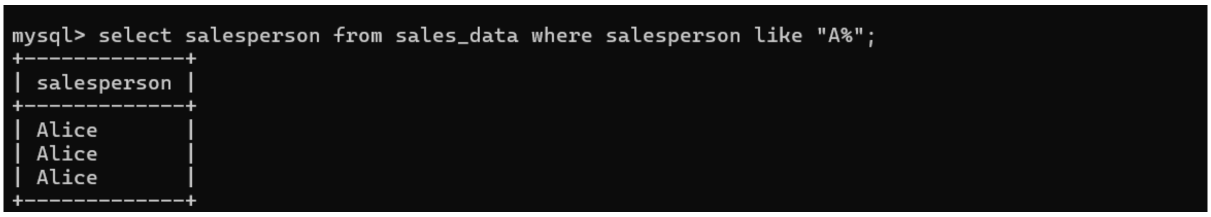
```
mysql> select avg(amount) from sales_data;
+-------------+
| avg(amount) |
+-------------+
|  383.750000 |
+-------------+
```

5)Minimum quantity:

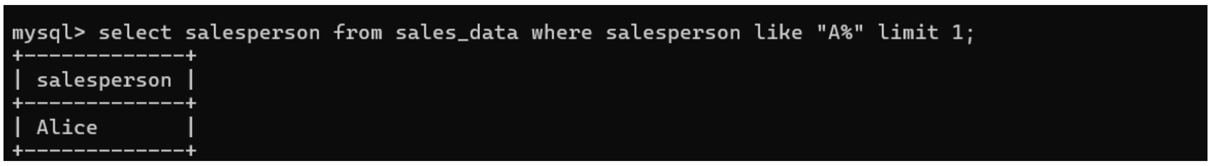


```
mysql> select min(quantity) from sales_data;
+---------------+
| min(quantity) |
+---------------+
|             1 |
+---------------+
```

6)Sales person whose name start with A:

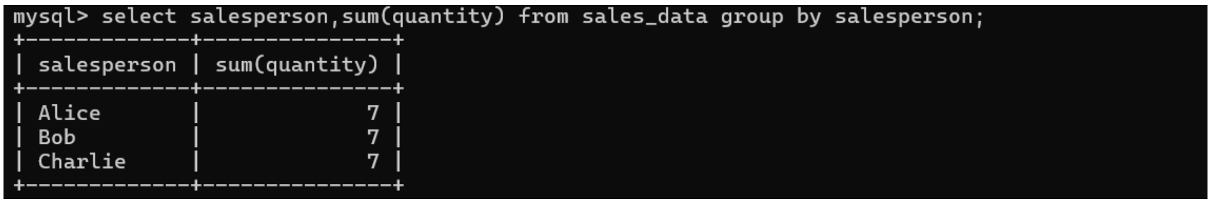


```
mysql> select salesperson from sales_data where salesperson like "A%";
+-------------+
| salesperson |
+-------------+
| Alice       |
| Alice       |
| Alice       |
+-------------+
```



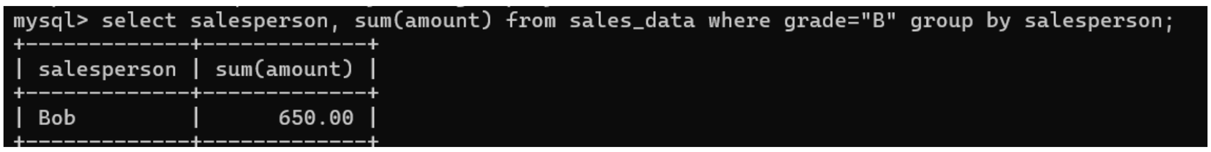
```
mysql> select salesperson from sales_data where salesperson like "A%" limit 1;
+-------------+
| salesperson |
+-------------+
| Alice       |
+-------------+
```

7)How many quantity sold out by each sales person



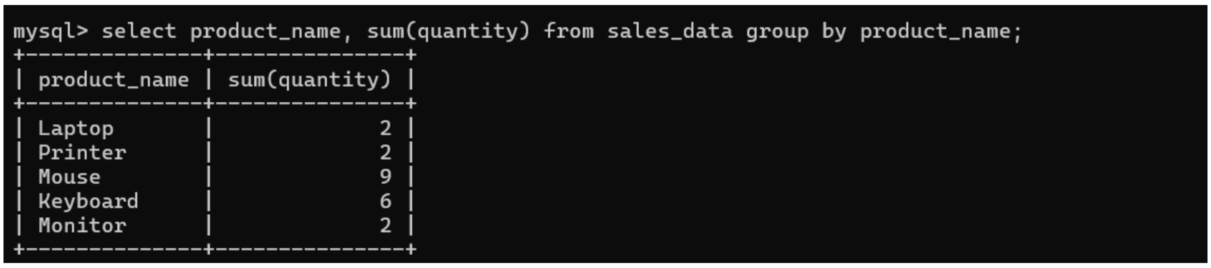
```
mysql> select salesperson,sum(quantity) from sales_data group by salesperson;
+-------------+---------------+
| salesperson | sum(quantity) |
+-------------+---------------+
| Alice       |             7 |
| Bob         |             7 |
| Charlie     |             7 |
+-------------+---------------+
```

8)Total amount sold out by each sales person for grade B:



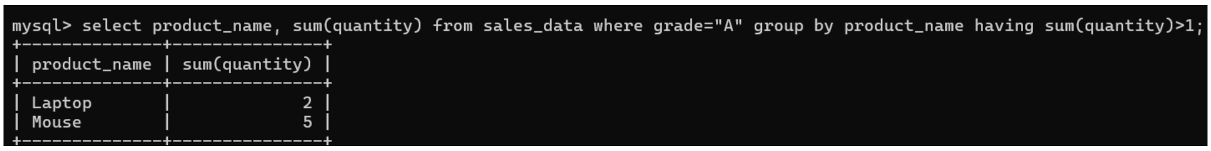
```
mysql> select salesperson, sum(amount) from sales_data where grade="B" group by salesperson;
+-------------+-------------+
| salesperson | sum(amount) |
+-------------+-------------+
| Bob         |      650.00 |
+-------------+-------------+
```

9)Find total quantity sold out for each product:



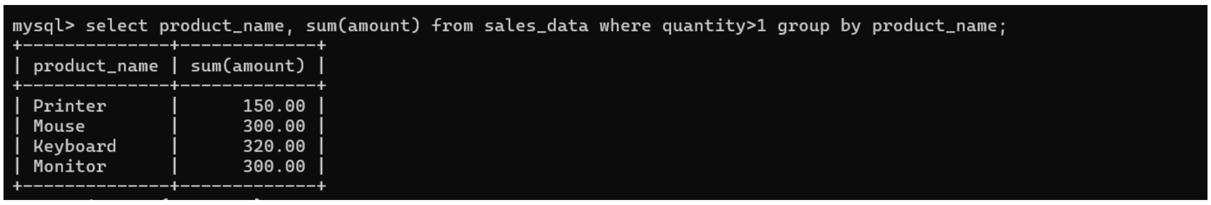
```
mysql> select product_name, sum(quantity) from sales_data group by product_name;
+--------------+---------------+
| product_name | sum(quantity) |
+--------------+---------------+
| Laptop       |             2 |
| Printer      |             2 |
| Mouse        |             9 |
| Keyboard     |             6 |
| Monitor      |             2 |
+--------------+---------------+
```

10)Find total quantity for each product name for grade A and total quantity greater than 1:



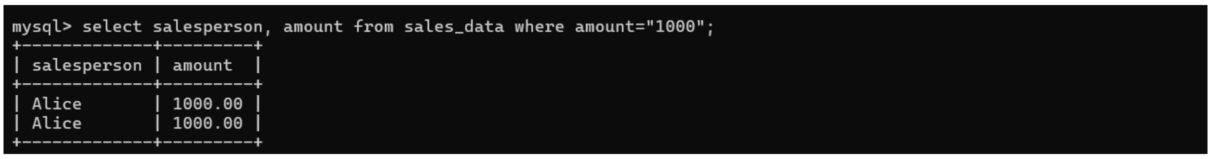
```
mysql> select product_name, sum(quantity) from sales_data where grade="A" group by product_name having sum(quantity)>1;
+--------------+---------------+
| product_name | sum(quantity) |
+--------------+---------------+
| Laptop       |             2 |
| Mouse        |             5 |
+--------------+---------------+
```

11)Find total amount for each product name for quantity >1:



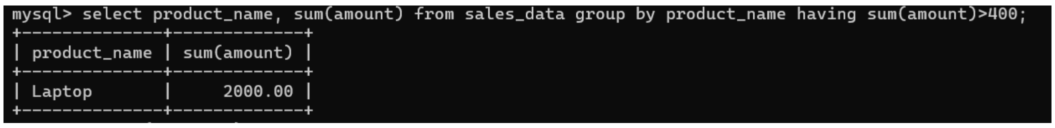
```
mysql> select product_name, sum(amount) from sales_data where quantity>1 group by product_name;
+--------------+-------------+
| product_name | sum(amount) |
+--------------+-------------+
| Printer      |      150.00 |
| Mouse        |      300.00 |
| Keyboard     |      320.00 |
| Monitor      |      300.00 |
+--------------+-------------+
```

12)Find salesperson who sold out amount 1000:



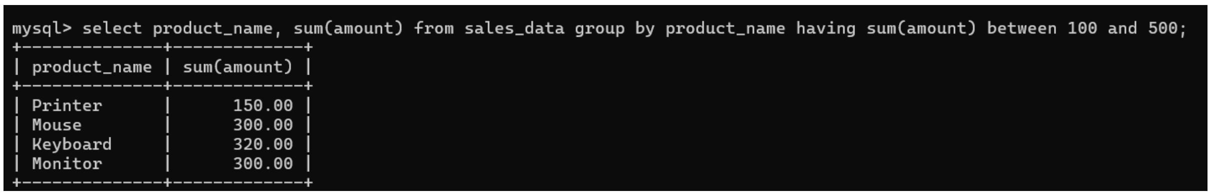
```
mysql> select salesperson, amount from sales_data where amount="1000";
+-------------+---------+
| salesperson | amount  |
+-------------+---------+
| Alice       | 1000.00 |
| Alice       | 1000.00 |
+-------------+---------+
```

13)Find total amount for each product whose total amount greater than 400:



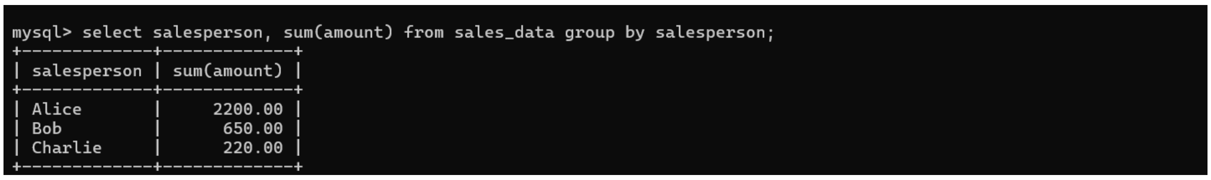
```
mysql> select product_name, sum(amount) from sales_data group by product_name having sum(amount)>400;
+--------------+-------------+
| product_name | sum(amount) |
+--------------+-------------+
| Laptop       |     2000.00 |
+--------------+-------------+
```

14)Find total amount for each product name whose total amount from 100 to 500:



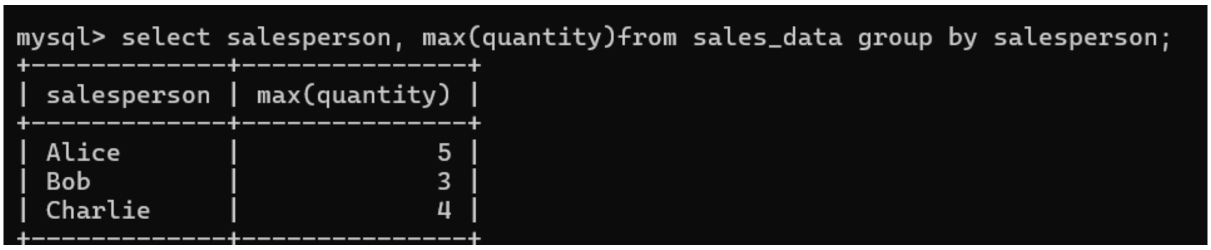
```
mysql> select product_name, sum(amount) from sales_data group by product_name having sum(amount) between 100 and 500;
+--------------+-------------+
| product_name | sum(amount) |
+--------------+-------------+
| Printer      |      150.00 |
| Mouse        |      300.00 |
| Keyboard     |      320.00 |
| Monitor      |      300.00 |
+--------------+-------------+
```

15)Find total amount for each sales person:



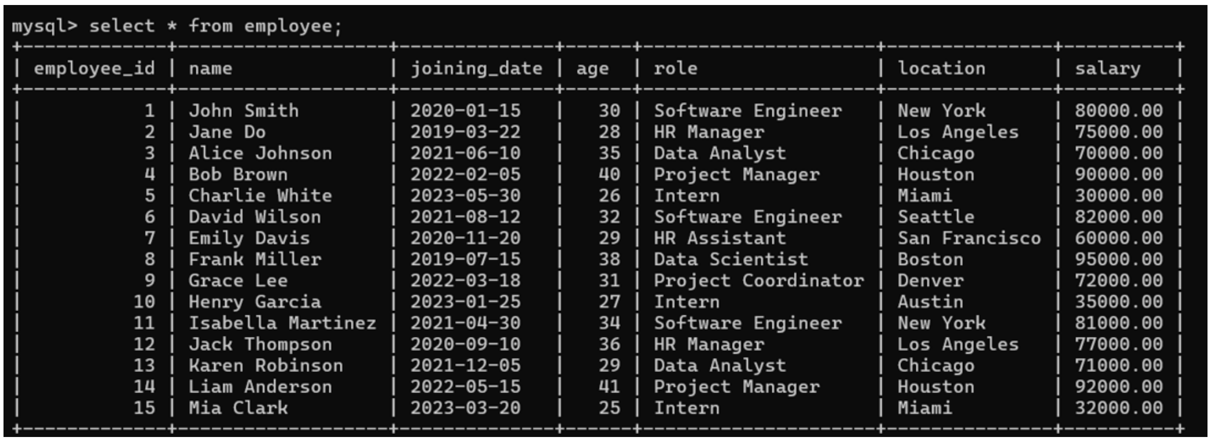
```
mysql> select salesperson, sum(amount) from sales_data group by salesperson;
+-------------+-------------+
| salesperson | sum(amount) |
+-------------+-------------+
| Alice       |     2200.00 |
| Bob         |      650.00 |
| Charlie     |      220.00 |
+-------------+-------------+
```

16)Find maximum quantity for each sales person:



```
mysql> select salesperson, max(quantity)from sales_data group by salesperson;
+-------------+---------------+
| salesperson | max(quantity) |
+-------------+---------------+
| Alice       |             5 |
| Bob         |             3 |
| Charlie     |             4 |
+-------------+---------------+
```

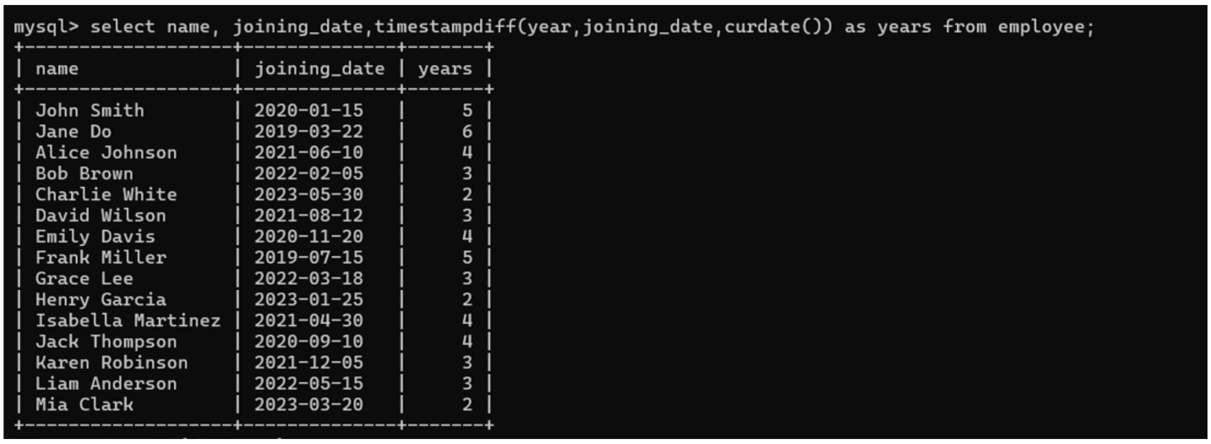
DAY 5



```
mysql> select * from employee;
+-------------+-------------------+--------------+-----+---------------------+---------------+----------+
| employee_id | name              | joining_date | age | role                | location      | salary   |
+-------------+-------------------+--------------+-----+---------------------+---------------+----------+
|           1 | John Smith        | 2020-01-15   |  30 | Software Engineer   | New York      | 80000.00 |
|           2 | Jane Do           | 2019-03-22   |  28 | HR Manager          | Los Angeles   | 75000.00 |
|           3 | Alice Johnson     | 2021-06-10   |  35 | Data Analyst        | Chicago       | 70000.00 |
|           4 | Bob Brown         | 2022-02-05   |  40 | Project Manager     | Houston       | 90000.00 |
|           5 | Charlie White     | 2023-05-30   |  26 | Intern              | Miami         | 30000.00 |
|           6 | David Wilson      | 2021-08-12   |  32 | Software Engineer   | Seattle       | 82000.00 |
|           7 | Emily Davis       | 2020-11-20   |  29 | HR Assistant        | San Francisco | 60000.00 |
|           8 | Frank Miller      | 2019-07-15   |  38 | Data Scientist      | Boston        | 95000.00 |
|           9 | Grace Lee         | 2022-03-18   |  31 | Project Coordinator | Denver        | 72000.00 |
|          10 | Henry Garcia      | 2023-01-25   |  27 | Intern              | Austin        | 35000.00 |
|          11 | Isabella Martinez | 2021-04-30   |  34 | Software Engineer   | New York      | 81000.00 |
|          12 | Jack Thompson     | 2020-09-10   |  36 | HR Manager          | Los Angeles   | 77000.00 |
|          13 | Karen Robinson    | 2021-12-05   |  29 | Data Analyst        | Chicago       | 71000.00 |
|          14 | Liam Anderson     | 2022-05-15   |  41 | Project Manager     | Houston       | 92000.00 |
|          15 | Mia Clark         | 2023-03-20   |  25 | Intern              | Miami         | 32000.00 |
+-------------+-------------------+--------------+-----+---------------------+---------------+----------+
```

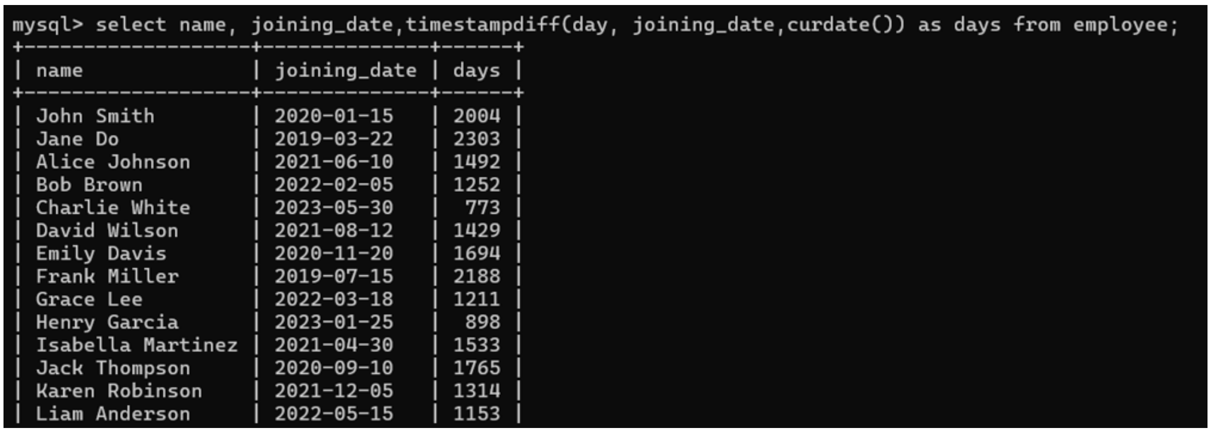
Timestampdiff():

1)How many years my employees are working?



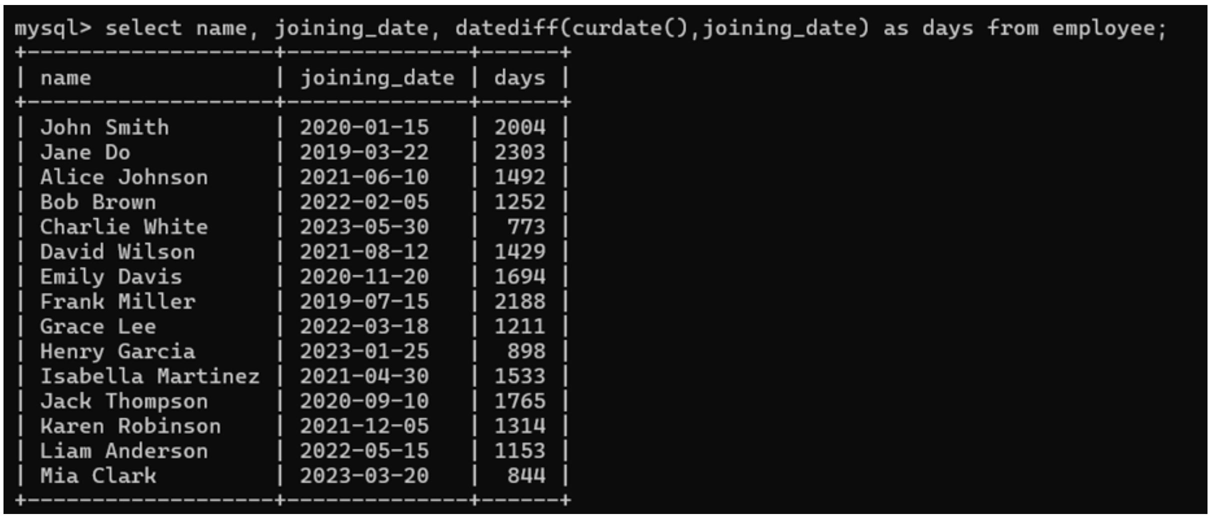
```
mysql> select name, joining_date,timestampdiff(year,joining_date,curdate()) as years from employee;
+-------------------+--------------+-------+
| name              | joining_date | years |
+-------------------+--------------+-------+
| John Smith        | 2020-01-15   |     5 |
| Jane Do           | 2019-03-22   |     6 |
| Alice Johnson     | 2021-06-10   |     4 |
| Bob Brown         | 2022-02-05   |     3 |
| Charlie White     | 2023-05-30   |     2 |
| David Wilson      | 2021-08-12   |     3 |
| Emily Davis       | 2020-11-20   |     4 |
| Frank Miller      | 2019-07-15   |     5 |
| Grace Lee         | 2022-03-18   |     3 |
| Henry Garcia      | 2023-01-25   |     2 |
| Isabella Martinez | 2021-04-30   |     4 |
| Jack Thompson     | 2020-09-10   |     4 |
| Karen Robinson    | 2021-12-05   |     3 |
| Liam Anderson     | 2022-05-15   |     3 |
| Mia Clark         | 2023-03-20   |     2 |
+-------------------+--------------+-------+
```

2)How many days my employees are working?



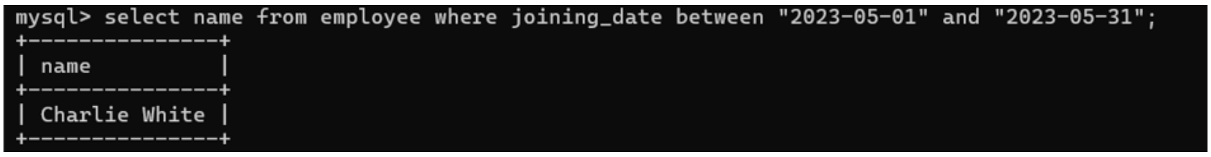
```
mysql> select name, joining_date,timestampdiff(day, joining_date,curdate()) as days from employee;
+-------------------+--------------+------+
| name              | joining_date | days |
+-------------------+--------------+------+
| John Smith        | 2020-01-15   | 2004 |
| Jane Do           | 2019-03-22   | 2303 |
| Alice Johnson     | 2021-06-10   | 1492 |
| Bob Brown         | 2022-02-05   | 1252 |
| Charlie White     | 2023-05-30   |  773 |
| David Wilson      | 2021-08-12   | 1429 |
| Emily Davis       | 2020-11-20   | 1694 |
| Frank Miller      | 2019-07-15   | 2188 |
| Grace Lee         | 2022-03-18   | 1211 |
| Henry Garcia      | 2023-01-25   |  898 |
| Isabella Martinez | 2021-04-30   | 1533 |
| Jack Thompson     | 2020-09-10   | 1765 |
| Karen Robinson    | 2021-12-05   | 1314 |
| Liam Anderson     | 2022-05-15   | 1153 |
```

Datediff()- To find days(minus the dates):



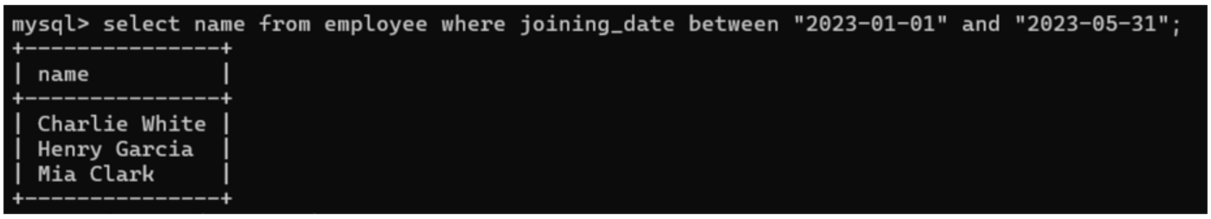
```
mysql> select name, joining_date, datediff(curdate(),joining_date) as days from employee;
+-------------------+--------------+------+
| name              | joining_date | days |
+-------------------+--------------+------+
| John Smith        | 2020-01-15   | 2004 |
| Jane Do           | 2019-03-22   | 2303 |
| Alice Johnson     | 2021-06-10   | 1492 |
| Bob Brown         | 2022-02-05   | 1252 |
| Charlie White     | 2023-05-30   |  773 |
| David Wilson      | 2021-08-12   | 1429 |
| Emily Davis       | 2020-11-20   | 1694 |
| Frank Miller      | 2019-07-15   | 2188 |
| Grace Lee         | 2022-03-18   | 1211 |
| Henry Garcia      | 2023-01-25   |  898 |
| Isabella Martinez | 2021-04-30   | 1533 |
| Jack Thompson     | 2020-09-10   | 1765 |
| Karen Robinson    | 2021-12-05   | 1314 |
| Liam Anderson     | 2022-05-15   | 1153 |
| Mia Clark         | 2023-03-20   |  844 |
+-------------------+--------------+------+
```

1)I need employees name who joined on 2023 May:



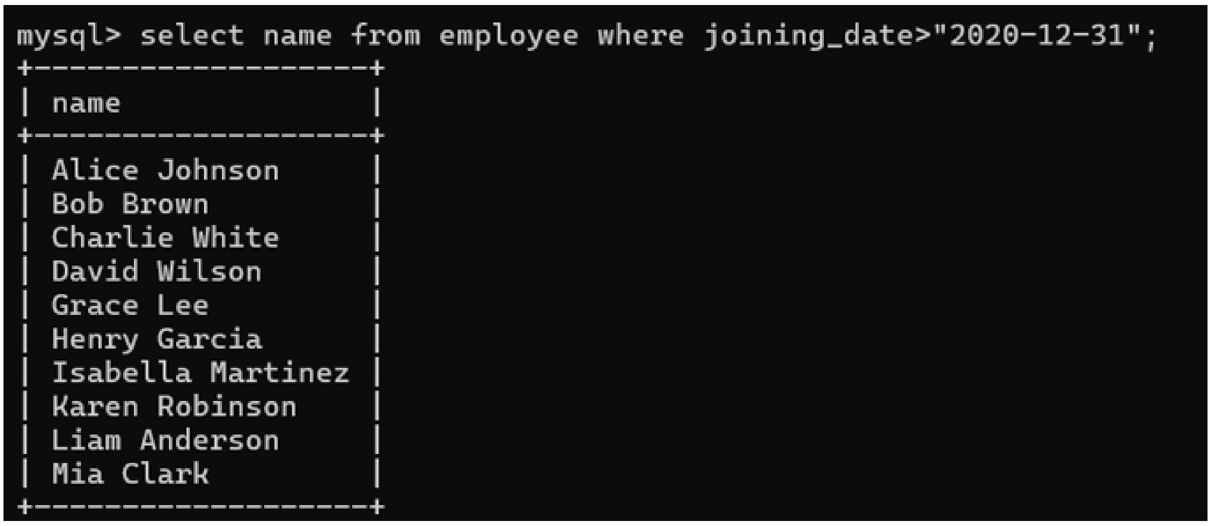
```
mysql> select name from employee where joining_date between "2023-05-01" and "2023-05-31";
+---------------+
| name          |
+---------------+
| Charlie White |
+---------------+
```

2) I need employees name who joined on 2023 January , 2023 May:

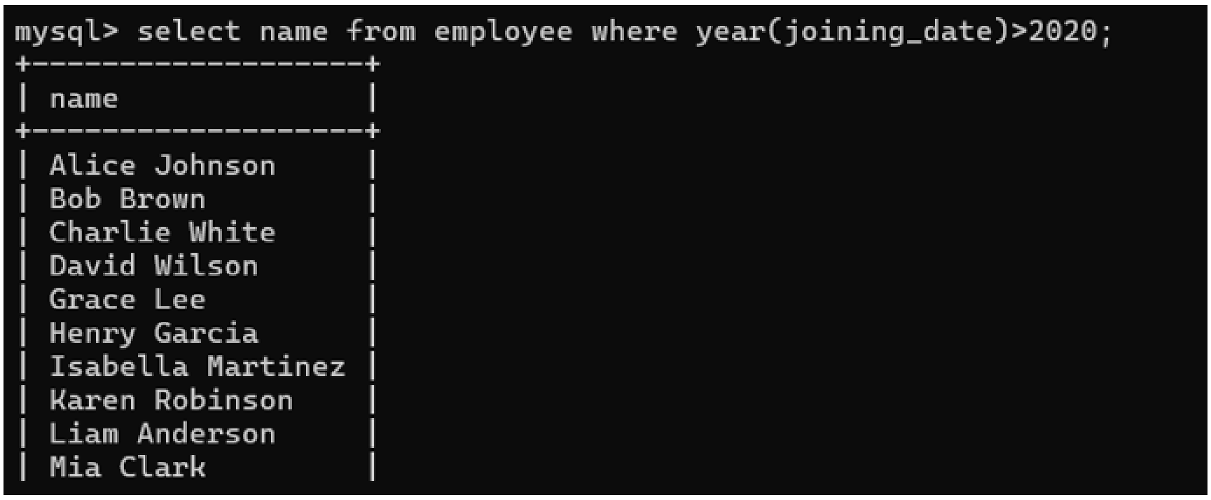


```
mysql> select name from employee where joining_date between "2023-01-01" and "2023-05-31";
+---------------+
| name          |
+---------------+
| Charlie White |
| Henry Garcia  |
| Mia Clark     |
+---------------+
```

4)I need to find employee name who joined after 2020:



```
mysql> select name from employee where joining_date>"2020-12-31";
+--------------------+
| name               |
+--------------------+
| Alice Johnson      |
| Bob Brown          |
| Charlie White      |
| David Wilson       |
| Grace Lee          |
| Henry Garcia       |
| Isabella Martinez  |
| Karen Robinson     |
| Liam Anderson      |
| Mia Clark          |
+--------------------+
```



```
mysql> select name from employee where year(joining_date)>2020;
+--------------------+
| name               |
+--------------------+
| Alice Johnson      |
| Bob Brown          |
| Charlie White      |
| David Wilson       |
| Grace Lee          |
| Henry Garcia       |
| Isabella Martinez  |
| Karen Robinson     |
| Liam Anderson      |
| Mia Clark          |
```

NORMALIZATION & DENORMALIZATION:

DENORMALIZATION → Single Table

Disadvantages:

Doesn’t able to maintain table

Anomalies

1. Insert
2. Update
3. Delete

Data Redundancy

NORMALIZATION → Multiple Tables

It is the process of designing a database effectively that we can avoid data redundancy.

It contains 2 or multiple tables and avoid the anomalies.

CANDIDATE KEY:

Set of columns which uniquely identify a column.

NON KEY COLUMN:

All tables other than candidate key or primary key.

PARTIAL DEPENDENCY:

A column partially depends on candidate key.

1NF:

Every column/attribute must have single value.
Each row should be unique not mandatory to have primary key.

2NF:

If a non key column is partially dependent on candidate key split them into separate table.

All non key attributes must be fully dependent on candidate key.

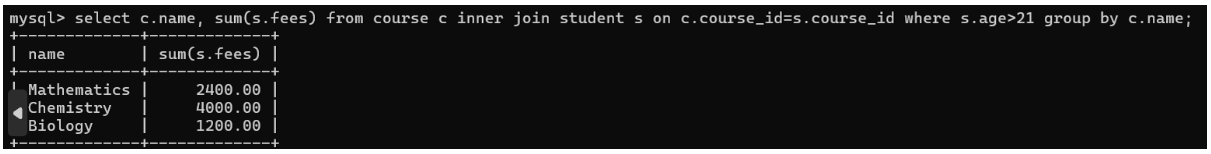
DAY 6

TYPES OF JOIN

Inner join- Return common records from tables
Outer join- Return common and uncommon records from tables
Left join- Return common and uncommon from left table
Right join- Return common and uncommon from right table
Self join-

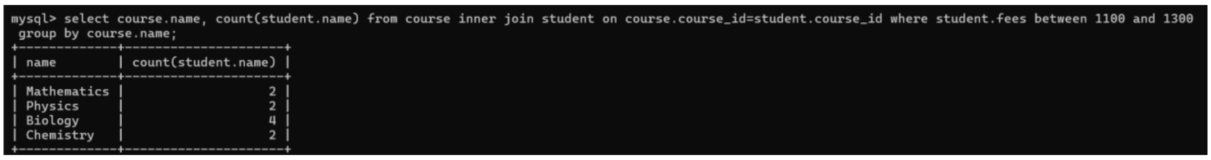
DAY 7:

1) I need to find total fees collected for each course whose student age > 21.



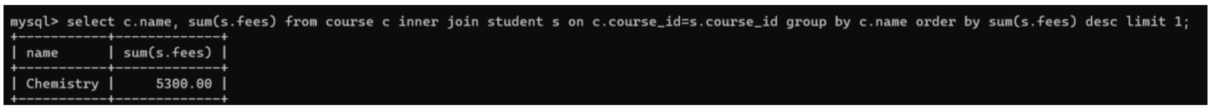
```
mysql> select c.name, sum(s.fees) from course c inner join student s on c.course_id=s.course_id where s.age>21 group by c.name;
+-------------+-------------+
| name        | sum(s.fees) |
+-------------+-------------+
| Mathematics |     2400.00 |
 Chemistry   |     4000.00 |
 Biology     |     1200.00 |
+-------------+-------------+
```

2) I need to find how many students studying each course who has fees between 1100-1300



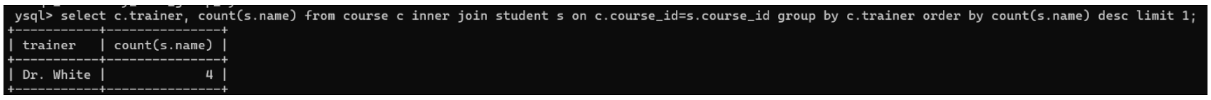
```
mysql> select course.name, count(student.name) from course inner join student on course.course_id=student.course_id where student.fees between 1100 and 1300
 group by course.name;
+-------------+---------------------+
| name        | count(student.name) |
+-------------+---------------------+
| Mathematics |                   2 |
| Physics     |                   2 |
| Biology     |                   4 |
| Chemistry   |                   2 |
+-------------+---------------------+
```

3) I need to find which course collecting highest amount of total fees.



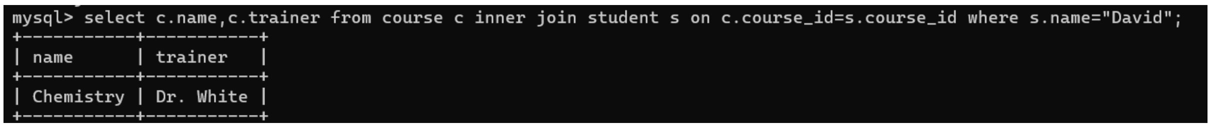
```
mysql> select c.name, sum(s.fees) from course c inner join student s on c.course_id=s.course_id group by c.name order by sum(s.fees) desc limit 1;
+-----------+-------------+
| name      | sum(s.fees) |
+-----------+-------------+
| Chemistry |     5300.00 |
+-----------+-------------+
```

4) I need to find which trainer has maximum no. of. Students.



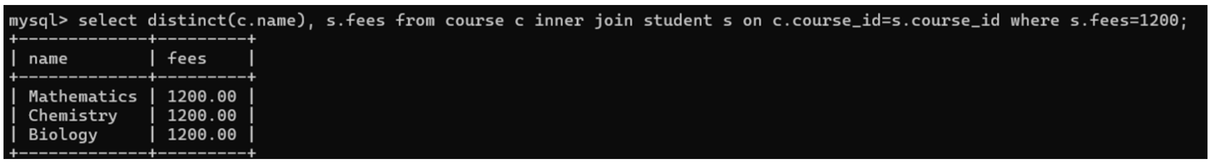
```
ysql> select c.trainer, count(s.name) from course c inner join student s on c.course_id=s.course_id group by c.trainer order by count(s.name) desc limit 1;
+-----------+---------------+
| trainer   | count(s.name) |
+-----------+---------------+
| Dr. White |             4 |
+-----------+---------------+
```

5) I need to find the course name and trainer name for student David.



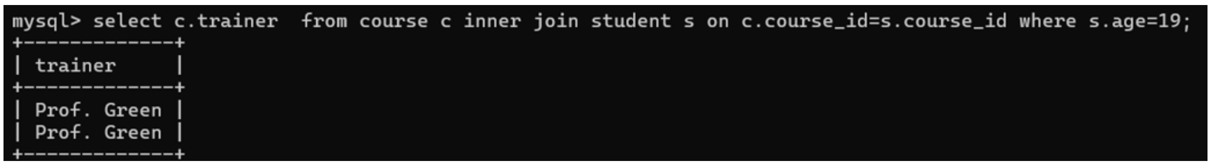
```
mysql> select c.name,c.trainer from course c inner join student s on c.course_id=s.course_id where s.name="David";
+-----------+-----------+
| name      | trainer   |
+-----------+-----------+
| Chemistry | Dr. White |
+-----------+-----------+
```

6) I need to find course name which are collecting fees 1200



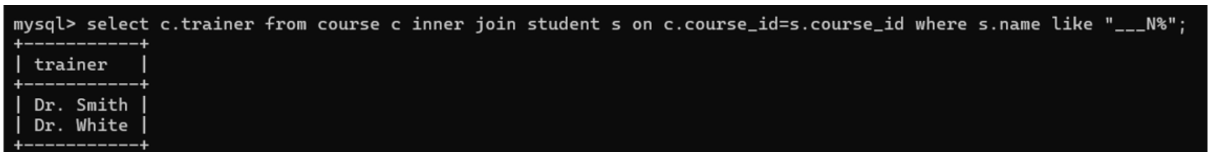
```
mysql> select distinct(c.name), s.fees from course c inner join student s on c.course_id=s.course_id where s.fees=1200;
+-------------+---------+
| name        | fees    |
+-------------+---------+
| Mathematics | 1200.00 |
| Chemistry   | 1200.00 |
| Biology     | 1200.00 |
+-------------+---------+
```

7) I need to find trainer name who are teaching student age 19.



```
mysql> select c.trainer  from course c inner join student s on c.course_id=s.course_id where s.age=19;
+-------------+
| trainer     |
+-------------+
| Prof. Green |
| Prof. Green |
+-------------+
```

8) I need to find trainer name for student fourth letter start with N.



```
mysql> select c.trainer from course c inner join student s on c.course_id=s.course_id where s.name like "___N%";
+-----------+
| trainer   |
+-----------+
| Dr. Smith |
| Dr. White |
+-----------+
```

CREATE TABLE Customers (
  Customer_id INT PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100),
  address VARCHAR(255)
);

INSERT INTO Customers (customer_id,name, email, address) VALUES
(101,'Alice Johnson', 'alice.johnson@example.com', '123 Maple St.'),
(102,'Bob Smith', 'bob.smith@example.com', '456 Oak St.'),
(103,'Charlie Brown', 'charlie.brown@example.com', '789 Pine St.'),
(104,'David Wilson', 'david.wilson@example.com', '321 Elm St.'),
(105,'Eva Green', 'eva.green@example.com', '654 Cedar St.')

CREATE TABLE Products (
  Product_id INT PRIMARY KEY,
  name VARCHAR(100),
  price DECIMAL(10, 2)
);

INSERT INTO Products (product_id,name, price) VALUES
(201,'Laptop', 1200.00),
(202,'Smartphone', 800.00),
(203,'Tablet', 400.00),
(204,'Headphones', 150.00),
(205,'Smartwatch', 250.00);

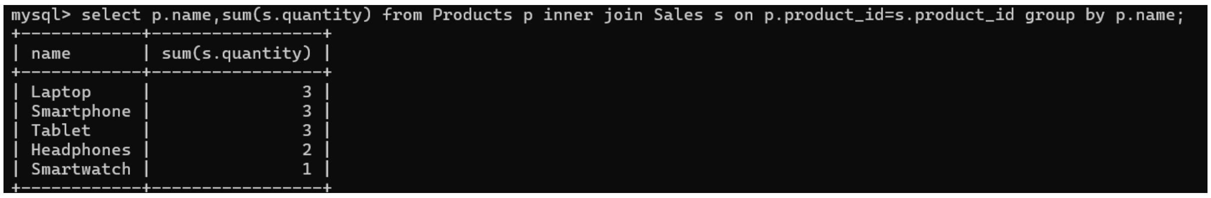
create table Sales(order_id int not null,customer_id int not null,product_id int not null,sales_date date,quantity int not null , total int not null, foreign key (Customer_id) references customers(Customer_id),foreign key(product_id)
 references products(product_id));

INSERT INTO sales (order_id,customer_id, product_id, sales_date, quantity, total) VALUES
(31,101, 201, '2024-01-15', 1, 1200.00), -- Alice bought 1 Laptop
(32, 102, 202,'2024-01-16', 2, 1600.00), -- Alice bought 2 Smartphones

(33, 102,201, '2024-01-17', 1, 1200.00), -- Bob bought 1 Laptop
(34,102, 203, '2024-01-18', 3, 1200.00), -- Bob bought 3 Tablets
(35, 103,202, '2024-01-19', 1, 800.00),   -- Charlie bought 1 Smartphone
(36, 103,204, '2024-01-20', 2, 300.00),   -- Charlie bought 2 Headphones
(37, 104,205, '2024-01-21', 1, 250.00),   -- David bought 1 Smartwatch
(38, 105,201, '2024-01-22', 1,1200.00);

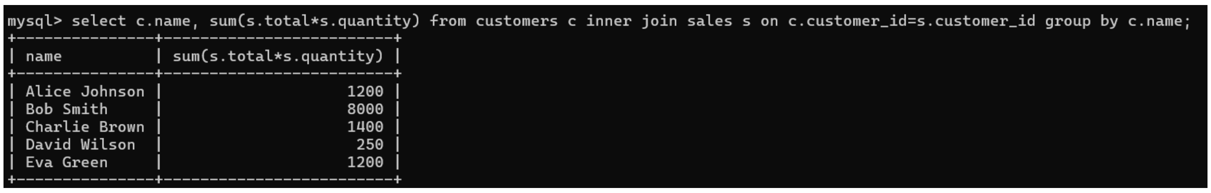
TASK

1) I need to find the total quantity sold for each product name.



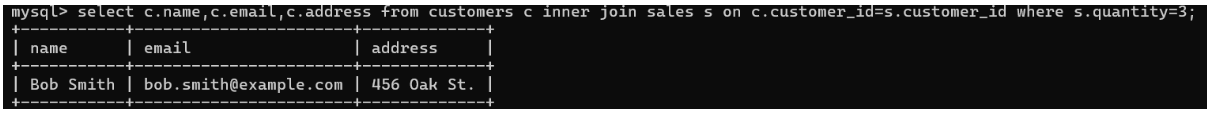
```
mysql> select p.name,sum(s.quantity) from Products p inner join Sales s on p.product_id=s.product_id group by p.name;
+------------+-----------------+
| name       | sum(s.quantity) |
+------------+-----------------+
| Laptop     |               3 |
| Smartphone |               3 |
| Tablet     |               3 |
| Headphones |               2 |
| Smartwatch |               1 |
+------------+-----------------+
```

2) I need to find the total sales amount for each customer name.



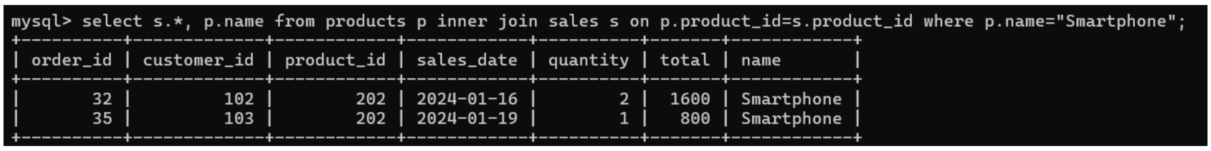
```
mysql> select c.name, sum(s.total*s.quantity) from customers c inner join sales s on c.customer_id=s.customer_id group by c.name;
+---------------+-------------------------+
| name          | sum(s.total*s.quantity) |
+---------------+-------------------------+
| Alice Johnson |                    1200 |
| Bob Smith     |                    8000 |
| Charlie Brown |                    1400 |
| David Wilson  |                     250 |
| Eva Green     |                    1200 |
+---------------+-------------------------+
```

3) I need to find customer name, email id and address of those who purchased a quantity of 3.



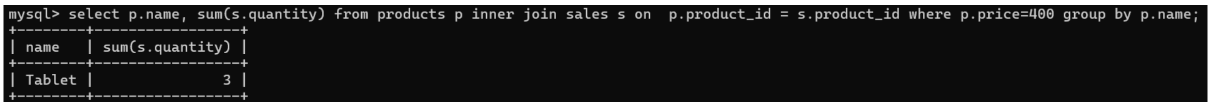
```
mysql> select c.name,c.email,c.address from customers c inner join sales s on c.customer_id=s.customer_id where s.quantity=3;
+-----------+-----------------------+-------------+
| name      | email                 | address     |
+-----------+-----------------------+-------------+
| Bob Smith | bob.smith@example.com | 456 Oak St. |
+-----------+-----------------------+-------------+
```

4) I need to find the sales data for the product named smartphone.



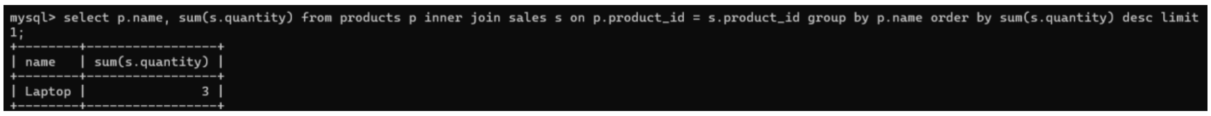
```
mysql> select s.*, p.name from products p inner join sales s on p.product_id=s.product_id where p.name="Smartphone";
+----------+-------------+------------+------------+----------+-------+------------+
| order_id | customer_id | product_id | sales_date | quantity | total | name       |
+----------+-------------+------------+------------+----------+-------+------------+
|       32 |         102 |        202 | 2024-01-16 |        2 |  1600 | Smartphone |
|       35 |         103 |        202 | 2024-01-19 |        1 |   800 | Smartphone |
+----------+-------------+------------+------------+----------+-------+------------+
```

5) I need to find the total quantity sold for products with a price of 400.



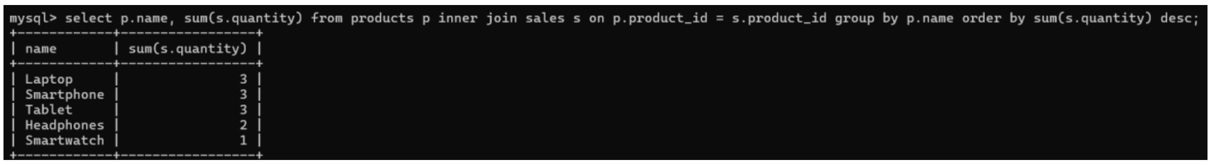
```
mysql> select p.name, sum(s.quantity) from products p inner join sales s on  p.product_id = s.product_id where p.price=400 group by p.name;
+--------+-----------------+
| name   | sum(s.quantity) |
+--------+-----------------+
| Tablet |               3 |
+--------+-----------------+
```

6) I need to find which product was sold in the highest quantity.



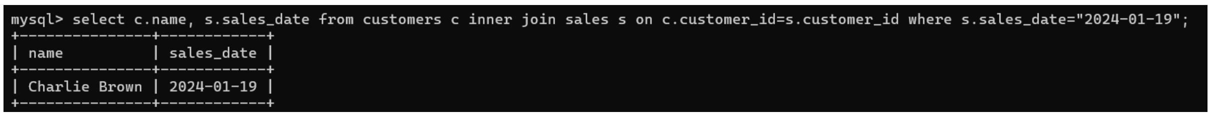
```
mysql> select p.name, sum(s.quantity) from products p inner join sales s on p.product_id = s.product_id group by p.name order by sum(s.quantity) desc limit
1;
+--------+-----------------+
| name   | sum(s.quantity) |
+--------+-----------------+
| Laptop |               3 |
+--------+-----------------+
```

7) I need to find which product has the highest total quantity sold (same as 6 if you meant by multiple orders, clarify).



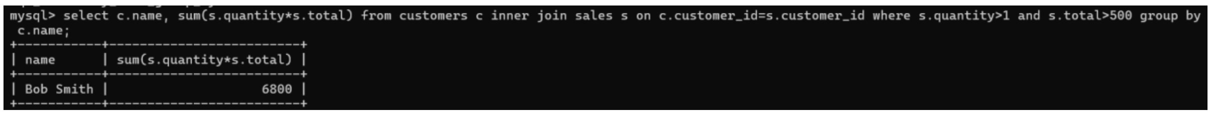
```
mysql> select p.name, sum(s.quantity) from products p inner join sales s on p.product_id = s.product_id group by p.name order by sum(s.quantity) desc;
+------------+-----------------+
| name       | sum(s.quantity) |
+------------+-----------------+
| Laptop     |               3 |
| Smartphone |               3 |
| Tablet     |               3 |
| Headphones |               2 |
| Smartwatch |               1 |
+------------+-----------------+
```

8) I need to find which customer bought a product on the sales date "2024-01-19".



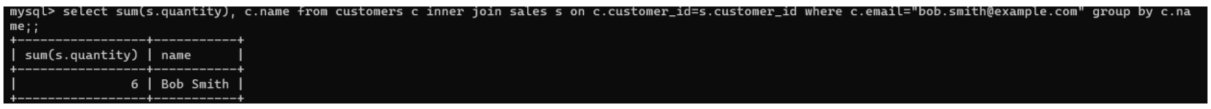
```
mysql> select c.name, s.sales_date from customers c inner join sales s on c.customer_id=s.customer_id where s.sales_date="2024-01-19";
+---------------+------------+
| name          | sales_date |
+---------------+------------+
| Charlie Brown | 2024-01-19 |
+---------------+------------+
```

9) I need to find the total sales amount for each customer where quantity grade>1 and sales grade>500.



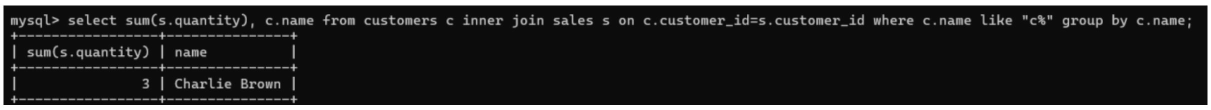
```
mysql> select c.name, sum(s.quantity*s.total) from customers c inner join sales s on c.customer_id=s.customer_id where s.quantity>1 and s.total>500 group by
 c.name;
+-----------+-------------------------+
| name      | sum(s.quantity*s.total) |
+-----------+-------------------------+
| Bob Smith |                    6800 |
+-----------+-------------------------+
```

10) I need to find the total quantity purchased by the customer with the email id bob.smith@example.com.



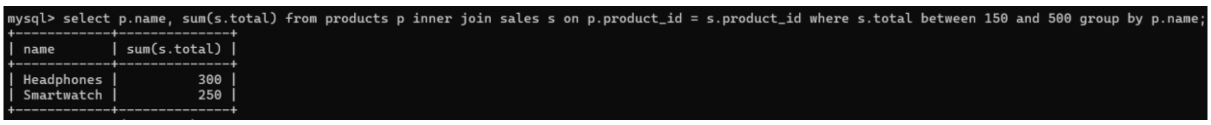
```
mysql> select sum(s.quantity), c.name from customers c inner join sales s on c.customer_id=s.customer_id where c.email="bob.smith@example.com" group by c.na
me;;
+-----------------+-----------+
| sum(s.quantity) | name      |
+-----------------+-----------+
|               6 | Bob Smith |
+-----------------+-----------+
```

11) I need to find total quantity purchased by customers whose names start with C.



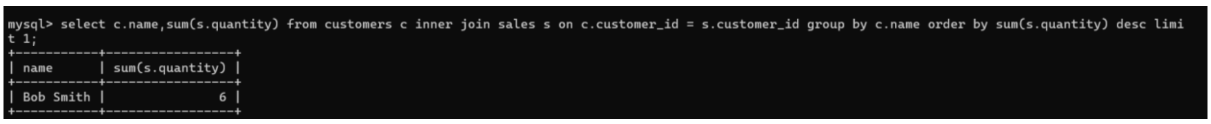
```
mysql> select sum(s.quantity), c.name from customers c inner join sales s on c.customer_id=s.customer_id where c.name like "c%" group by c.name;
+-----------------+---------------+
| sum(s.quantity) | name          |
+-----------------+---------------+
|               3 | Charlie Brown |
+-----------------+---------------+
```

12) I need to find the product names whose total price is between 150 and 500.



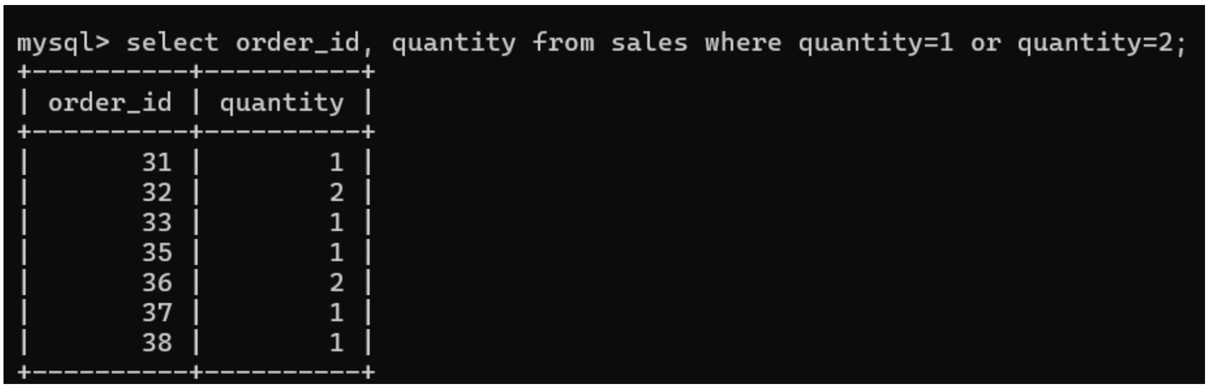
```
mysql> select p.name, sum(s.total) from products p inner join sales s on p.product_id = s.product_id where s.total between 150 and 500 group by p.name;
+------------+--------------+
| name       | sum(s.total) |
+------------+--------------+
| Headphones |          300 |
| Smartwatch |          250 |
+------------+--------------+
```

13) I need to find which customer bought the highest amount of products.



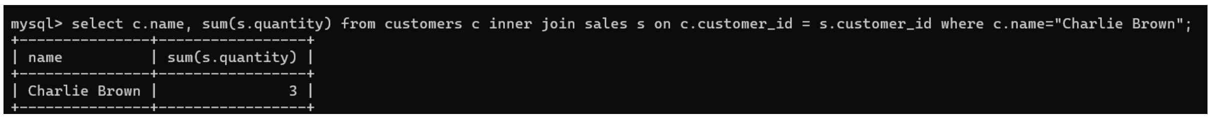
```
mysql> select c.name,sum(s.quantity) from customers c inner join sales s on c.customer_id = s.customer_id group by c.name order by sum(s.quantity) desc limi
t 1;
+-----------+-----------------+
| name      | sum(s.quantity) |
+-----------+-----------------+
| Bob Smith |               6 |
+-----------+-----------------+
```

14) I need to find the order ID s where the quantity is 1 or 2.



```
mysql> select order_id, quantity from sales where quantity=1 or quantity=2;
+----------+----------+
| order_id | quantity |
+----------+----------+
|       31 |        1 |
|       32 |        2 |
|       33 |        1 |
|       35 |        1 |
|       36 |        2 |
|       37 |        1 |
|       38 |        1 |
+----------+----------+
```

15) I need to find total quantity purchased by Charlie brown.



```
mysql> select c.name, sum(s.quantity) from customers c inner join sales s on c.customer_id = s.customer_id where c.name="Charlie Brown";
+---------------+-----------------+
| name          | sum(s.quantity) |
+---------------+-----------------+
| Charlie Brown |               3 |
+---------------+-----------------+
```

DAY 8:

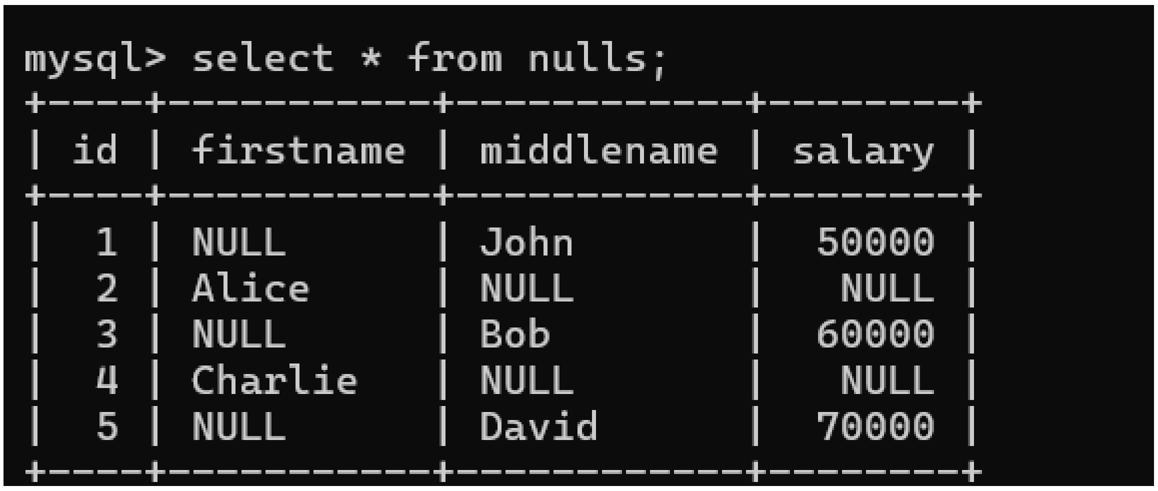
To handle null values we use

Is

Is not null

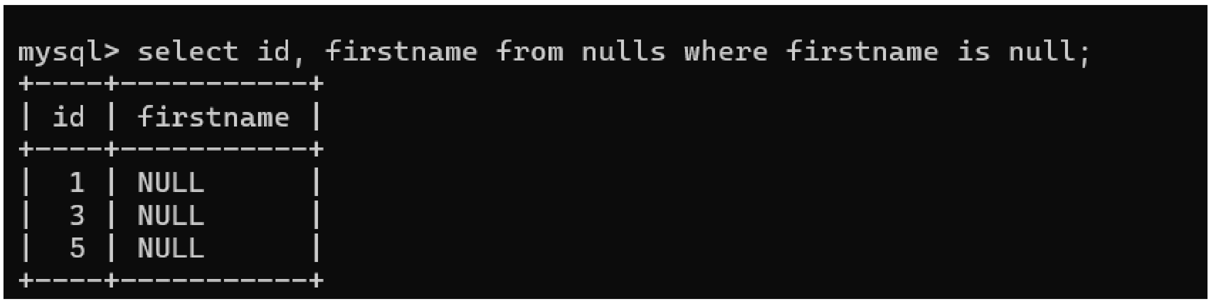
If null

coalesce



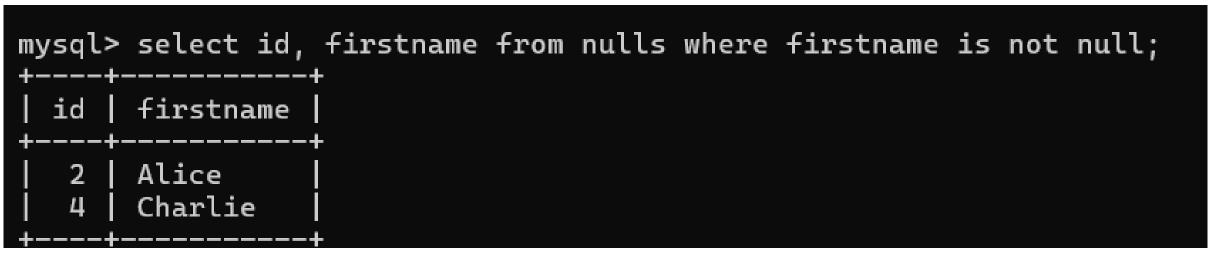
```
mysql> select * from nulls;
+----+-----------+------------+--------+
| id | firstname | middlename | salary |
+----+-----------+------------+--------+
|  1 | NULL      | John       |  50000 |
|  2 | Alice     | NULL       |   NULL |
|  3 | NULL      | Bob        |  60000 |
|  4 | Charlie   | NULL       |   NULL |
|  5 | NULL      | David      |  70000 |
+----+-----------+------------+--------+
```

1)Select id, first name from nulls where first name is null



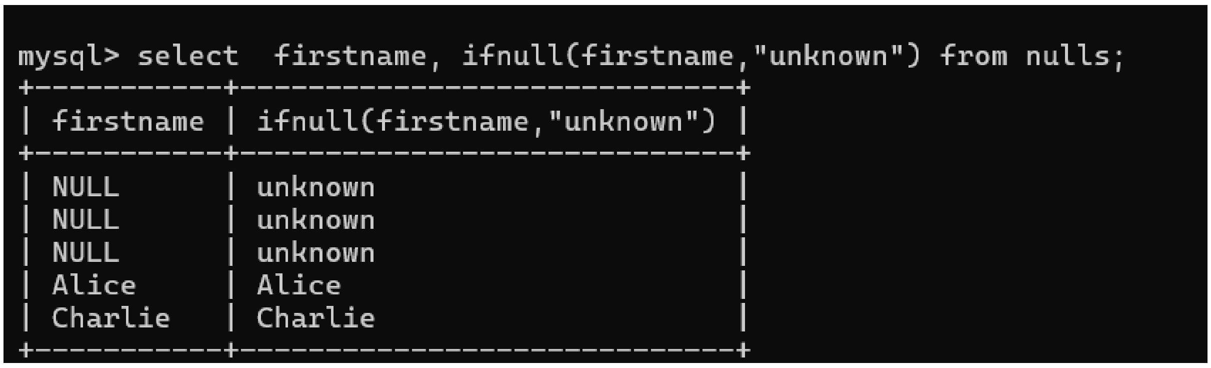
```
mysql> select id, firstname from nulls where firstname is null;
+----+-----------+
| id | firstname |
+----+-----------+
|  1 | NULL      |
|  3 | NULL      |
|  5 | NULL      |
+----+-----------+
```

2)select id, first name from nulls where first name is not null

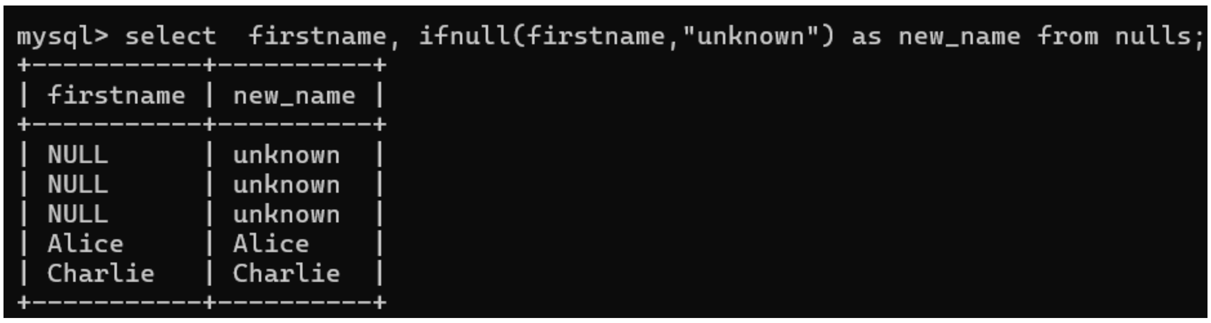


```
mysql> select id, firstname from nulls where firstname is not null;
+----+-----------+
| id | firstname |
+----+-----------+
|  2 | Alice     |
|  4 | Charlie   |
+----+-----------+
```

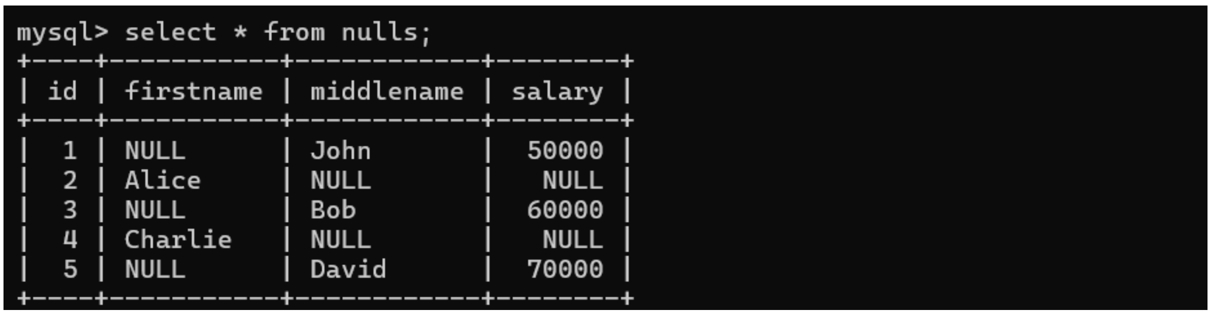
3)To fill unknown column



```
mysql> select  firstname, ifnull(firstname,"unknown") from nulls;
+-----------+-----------------------------+
| firstname | ifnull(firstname,"unknown") |
+-----------+-----------------------------+
| NULL      | unknown                     |
| NULL      | unknown                     |
| NULL      | unknown                     |
| Alice     | Alice                       |
| Charlie   | Charlie                     |
+-----------+-----------------------------+
```



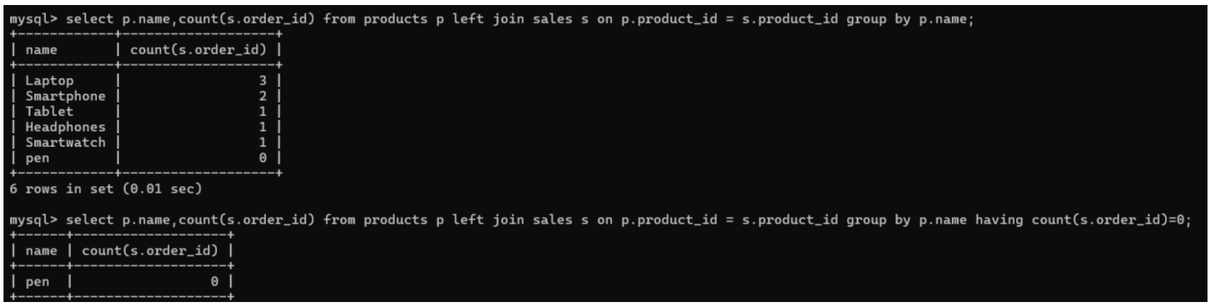
```
mysql> select  firstname, ifnull(firstname,"unknown") as new_name from nulls;
+-----------+----------+
| firstname | new_name |
+-----------+----------+
| NULL      | unknown  |
| NULL      | unknown  |
| NULL      | unknown  |
| Alice     | Alice    |
| Charlie   | Charlie  |
+-----------+----------+
```



```
mysql> select * from nulls;
+----+-----------+------------+--------+
| id | firstname | middlename | salary |
+----+-----------+------------+--------+
|  1 | NULL      | John       |  50000 |
|  2 | Alice     | NULL       |   NULL |
|  3 | NULL      | Bob        |  60000 |
|  4 | Charlie   | NULL       |   NULL |
|  5 | NULL      | David      |  70000 |
+----+-----------+------------+--------+
```

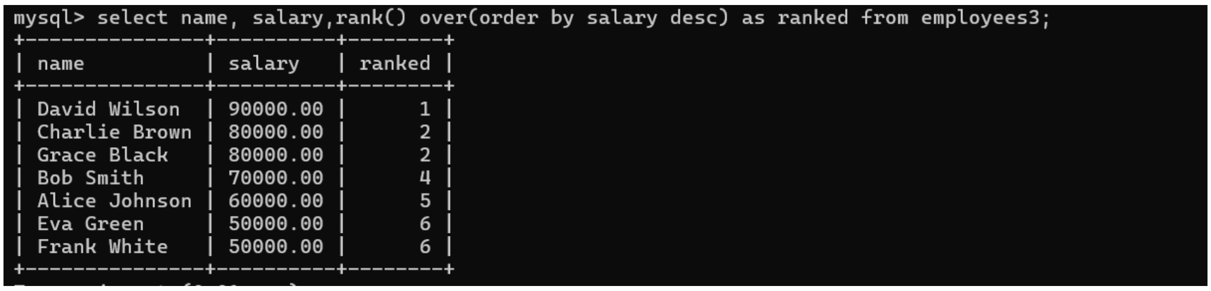
Product sales table:

To fetch uncommon values:



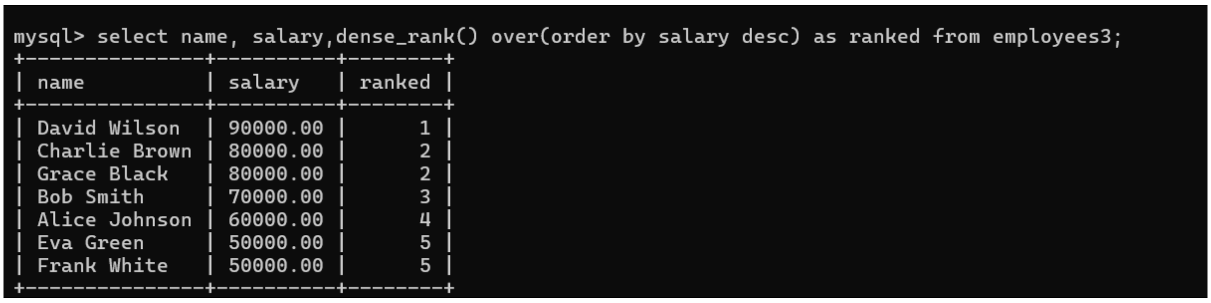
```
mysql> select p.name,count(s.order_id) from products p left join sales s on p.product_id = s.product_id group by p.name;
+------------+-------------------+
| name       | count(s.order_id) |
+------------+-------------------+
| Laptop     |                 3 |
| Smartphone |                 2 |
| Tablet     |                 1 |
| Headphones |                 1 |
| Smartwatch |                 1 |
| pen        |                 0 |
+------------+-------------------+
6 rows in set (0.01 sec)

mysql> select p.name,count(s.order_id) from products p left join sales s on p.product_id = s.product_id group by p.name having count(s.order_id)=0;
+------+-------------------+
| name | count(s.order_id) |
+------+-------------------+
| pen  |                 0 |
+------+-------------------+
```

RANK



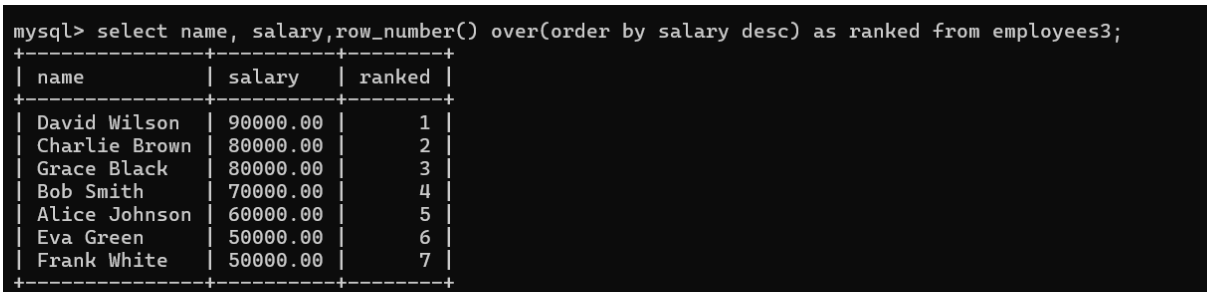
```
mysql> select name, salary,rank() over(order by salary desc) as ranked from employees3;
+---------------+----------+--------+
| name          | salary   | ranked |
+---------------+----------+--------+
| David Wilson  | 90000.00 |      1 |
| Charlie Brown | 80000.00 |      2 |
| Grace Black   | 80000.00 |      2 |
| Bob Smith     | 70000.00 |      4 |
| Alice Johnson | 60000.00 |      5 |
| Eva Green     | 50000.00 |      6 |
| Frank White   | 50000.00 |      6 |
+---------------+----------+--------+
```

DENSE RANK



```
mysql> select name, salary,dense_rank() over(order by salary desc) as ranked from employees3;
+---------------+----------+--------+
| name          | salary   | ranked |
+---------------+----------+--------+
| David Wilson  | 90000.00 |      1 |
| Charlie Brown | 80000.00 |      2 |
| Grace Black   | 80000.00 |      2 |
| Bob Smith     | 70000.00 |      3 |
| Alice Johnson | 60000.00 |      4 |
| Eva Green     | 50000.00 |      5 |
| Frank White   | 50000.00 |      5 |
+---------------+----------+--------+
```

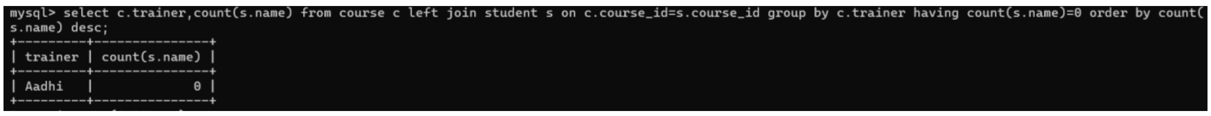
ROW NUMBER



```
mysql> select name, salary,row_number() over(order by salary desc) as ranked from employees3;
+---------------+----------+--------+
| name          | salary   | ranked |
+---------------+----------+--------+
| David Wilson  | 90000.00 |      1 |
| Charlie Brown | 80000.00 |      2 |
| Grace Black   | 80000.00 |      3 |
| Bob Smith     | 70000.00 |      4 |
| Alice Johnson | 60000.00 |      5 |
| Eva Green     | 50000.00 |      6 |
| Frank White   | 50000.00 |      7 |
+---------------+----------+--------+
```

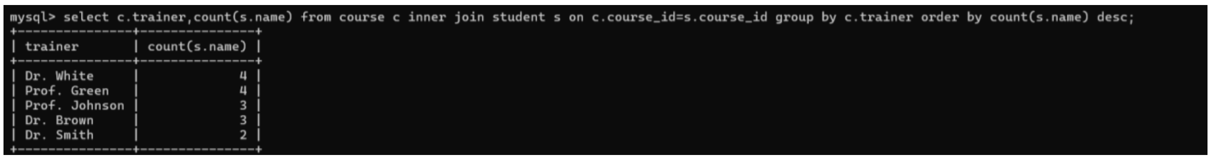
TASK

1) I need to find the trainer name who are not assigned single student



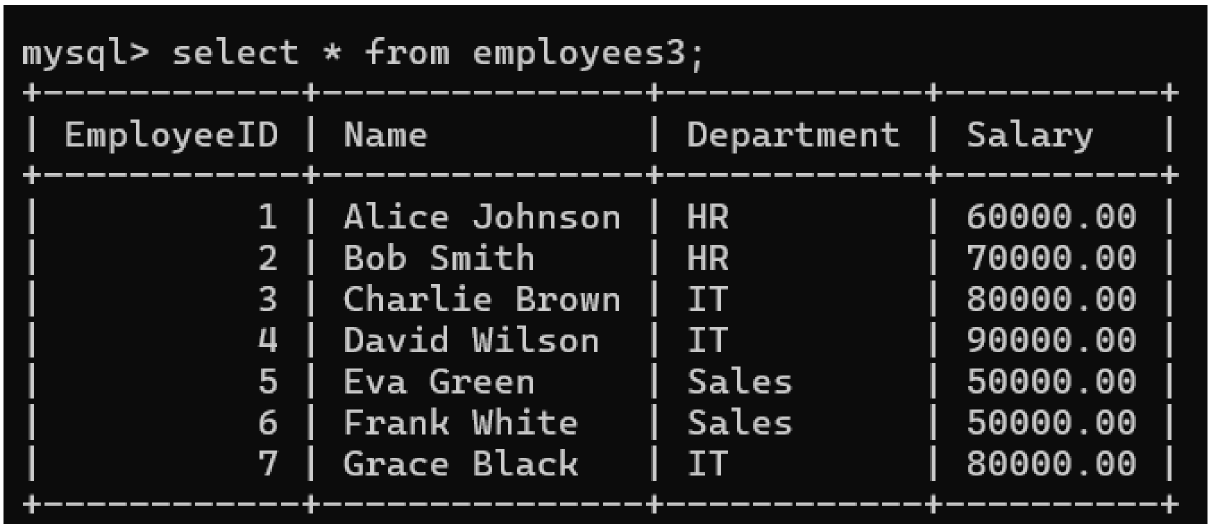
```
mysql> select c.trainer,count(s.name) from course c left join student s on c.course_id=s.course_id group by c.trainer having count(s.name)=0 order by count(
s.name) desc;
+---------+---------------+
| trainer | count(s.name) |
+---------+---------------+
| Aadhi   |             0 |
+---------+---------------+
```

2)I need to find the trainer name who are assigned students



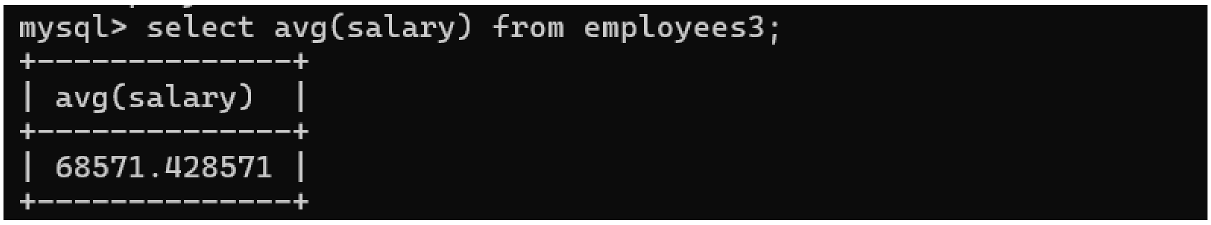
```
mysql> select c.trainer,count(s.name) from course c inner join student s on c.course_id=s.course_id group by c.trainer order by count(s.name) desc;
+---------------+---------------+
| trainer       | count(s.name) |
+---------------+---------------+
| Dr. White     |             4 |
| Prof. Green   |             4 |
| Prof. Johnson |             3 |
| Dr. Brown     |             3 |
| Dr. Smith     |             2 |
+---------------+---------------+
```

DAY 9

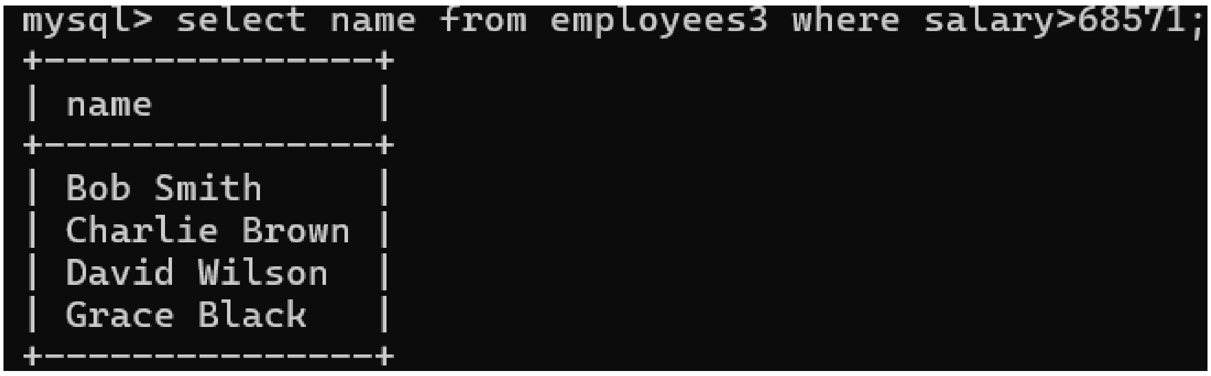


```
mysql> select * from employees3;
+------------+---------------+------------+----------+
| EmployeeID | Name          | Department | Salary   |
+------------+---------------+------------+----------+
|          1 | Alice Johnson | HR         | 60000.00 |
|          2 | Bob Smith     | HR         | 70000.00 |
|          3 | Charlie Brown | IT         | 80000.00 |
|          4 | David Wilson  | IT         | 90000.00 |
|          5 | Eva Green     | Sales      | 50000.00 |
|          6 | Frank White   | Sales      | 50000.00 |
|          7 | Grace Black   | IT         | 80000.00 |
+------------+---------------+------------+----------+
```

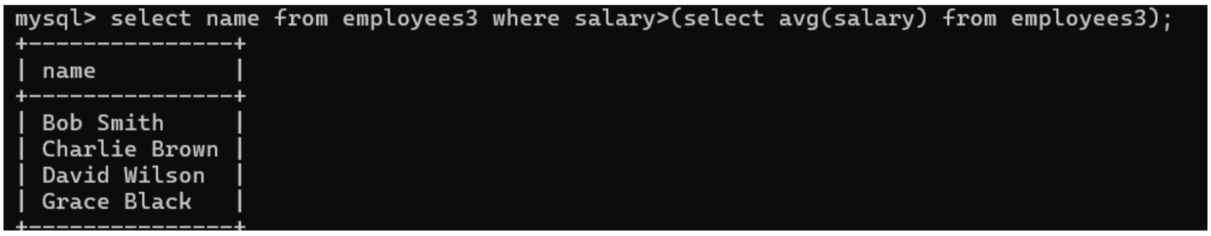
1)I need to find employee name who is getting salary greater than the average salary.



```
mysql> select avg(salary) from employees3;
+--------------+
| avg(salary)  |
+--------------+
| 68571.428571 |
+--------------+
```

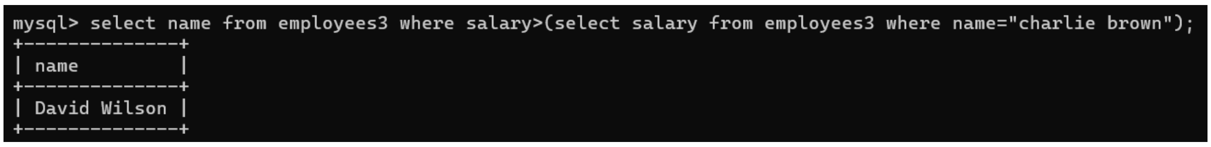


```
mysql> select name from employees3 where salary>68571;
+---------------+
| name          |
+---------------+
| Bob Smith     |
| Charlie Brown |
| David Wilson  |
| Grace Black   |
+---------------+
```



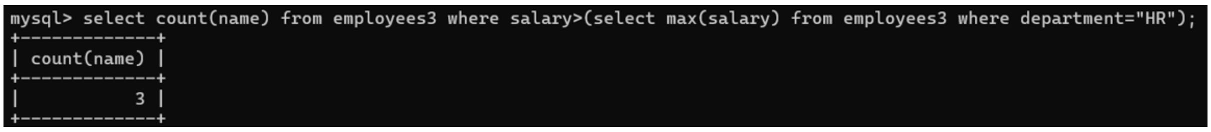
```
mysql> select name from employees3 where salary>(select avg(salary) from employees3);
+---------------+
| name          |
+---------------+
| Bob Smith     |
| Charlie Brown |
| David Wilson  |
| Grace Black   |
+---------------+
```

2) I need to find employee name who is getting salary greater than salary of Charlie brown.



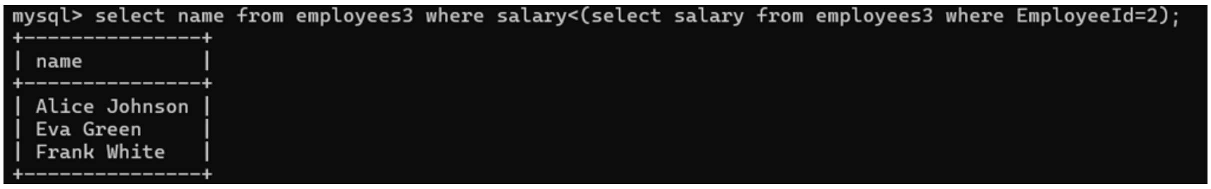
```
mysql> select name from employees3 where salary>(select salary from employees3 where name="charlie brown");
+--------------+
| name         |
+--------------+
| David Wilson |
+--------------+
```

3)I need to find count of employees who is getting salary > maximum salary of HR Department.



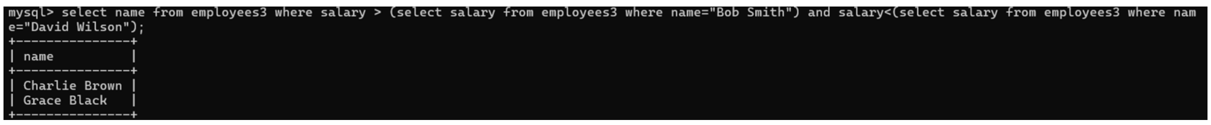
```
mysql> select count(name) from employees3 where salary>(select max(salary) from employees3 where department="HR");
+-------------+
| count(name) |
+-------------+
|           3 |
+-------------+
```

4) I need to find employees name who is getting salary less than the salary of employee id 2.



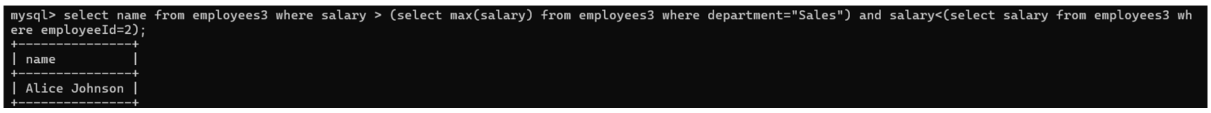
```
mysql> select name from employees3 where salary<(select salary from employees3 where EmployeeId=2);
+---------------+
| name          |
+---------------+
| Alice Johnson |
| Eva Green     |
| Frank White   |
+---------------+
```

5)I need to find employee name who is getting salary greater than bob smith and less than David Wilson.



```
mysql> select name from employees3 where salary > (select salary from employees3 where name="Bob Smith") and salary<(select salary from employees3 where nam
e="David Wilson");
+---------------+
| name          |
+---------------+
| Charlie Brown |
| Grace Black   |
+---------------+
```

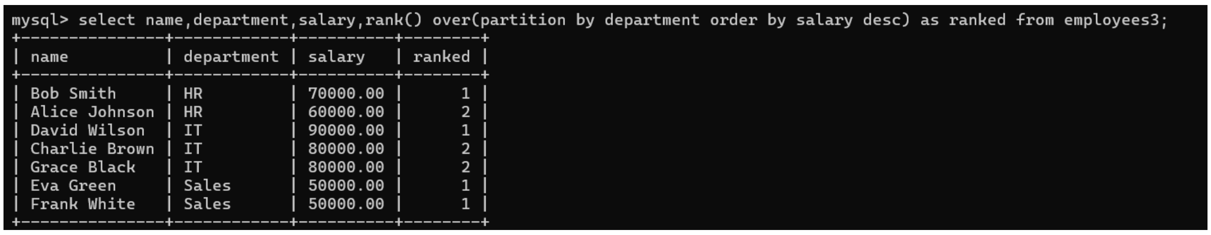
6)I need to find employee name who is getting salary > max salary of sales department and less than salary of employee id 2.



```
mysql> select name from employees3 where salary > (select max(salary) from employees3 where department="Sales") and salary<(select salary from employees3 wh
ere employeeId=2);
+---------------+
| name          |
+---------------+
| Alice Johnson |
+---------------+
```

PARTITION

To rank department wise:



```
mysql> select name,department,salary,rank() over(partition by department order by salary desc) as ranked from employees3;
+---------------+------------+----------+--------+
| name          | department | salary   | ranked |
+---------------+------------+----------+--------+
| Bob Smith     | HR         | 70000.00 |      1 |
| Alice Johnson | HR         | 60000.00 |      2 |
| David Wilson  | IT         | 90000.00 |      1 |
| Charlie Brown | IT         | 80000.00 |      2 |
| Grace Black   | IT         | 80000.00 |      2 |
| Eva Green     | Sales      | 50000.00 |      1 |
| Frank White   | Sales      | 50000.00 |      1 |
+---------------+------------+----------+--------+
```

TASK

1)Why do we use SQL?

We use SQL to store, retrive, manipulate in relational database.

2)Difference between My SQL and Ms SQL.

MySQL is open-source and free to use, while MS SQL is a commercial product with licensing costs. MySQL is known for its cross-platform compatibility and flexibility, while MS SQL is closely integrated with the Microsoft ecosystem.

3)Can you tell the coding order of SQL

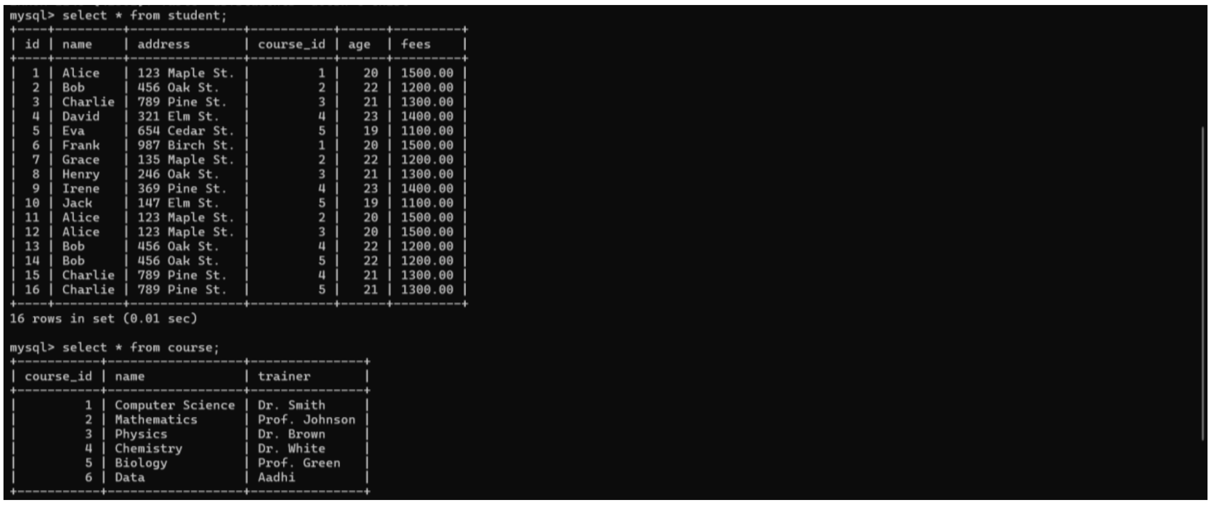
Select
From
Where
Group by
Having
Order by
Limit

4)Difference between where and having.

Where is to filter out columns from the table whereas having is to filter from the aggregate function.

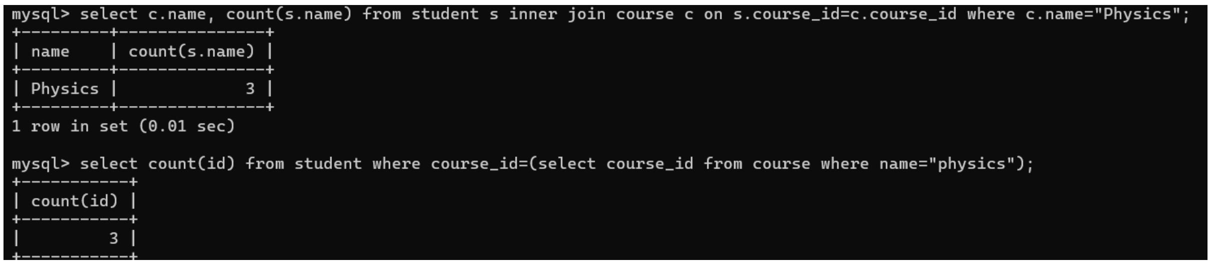
DAY 10

Viewing tha table:



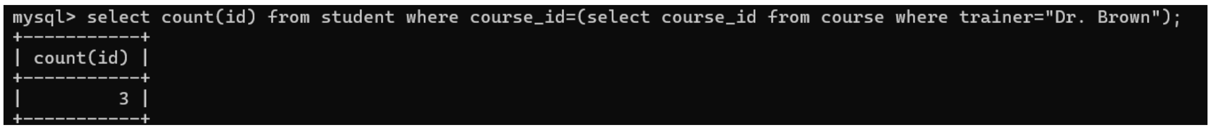
```
mysql> select * from student;
+----+---------+----------------+-----------+------+---------+
| id | name    | address        | course_id | age  | fees    |
+----+---------+----------------+-----------+------+---------+
|  1 | Alice   | 123 Maple St.  |         1 |   20 | 1500.00 |
|  2 | Bob     | 456 Oak St.    |         2 |   22 | 1200.00 |
|  3 | Charlie | 789 Pine St.   |         3 |   21 | 1300.00 |
|  4 | David   | 321 Elm St.    |         4 |   23 | 1400.00 |
|  5 | Eva     | 654 Cedar St.  |         5 |   19 | 1100.00 |
|  6 | Frank   | 987 Birch St.  |         1 |   20 | 1500.00 |
|  7 | Grace   | 135 Maple St.  |         2 |   22 | 1200.00 |
|  8 | Henry   | 246 Oak St.    |         3 |   21 | 1300.00 |
|  9 | Irene   | 369 Pine St.   |         4 |   23 | 1400.00 |
| 10 | Jack    | 147 Elm St.    |         5 |   19 | 1100.00 |
| 11 | Alice   | 123 Maple St.  |         2 |   20 | 1500.00 |
| 12 | Alice   | 123 Maple St.  |         3 |   20 | 1500.00 |
| 13 | Bob     | 456 Oak St.    |         4 |   22 | 1200.00 |
| 14 | Bob     | 456 Oak St.    |         5 |   22 | 1200.00 |
| 15 | Charlie | 789 Pine St.   |         4 |   21 | 1300.00 |
| 16 | Charlie | 789 Pine St.   |         5 |   21 | 1300.00 |
+----+---------+----------------+-----------+------+---------+
16 rows in set (0.01 sec)

mysql> select * from course;
+-----------+-------------------+---------------+
| course_id | name              | trainer       |
+-----------+-------------------+---------------+
|         1 | Computer Science  | Dr. Smith     |
|         2 | Mathematics       | Prof. Johnson |
|         3 | Physics           | Dr. Brown     |
|         4 | Chemistry         | Dr. White     |
|         5 | Biology           | Prof. Green   |
|         6 | Data              | Aadhi         |
+-----------+-------------------+---------------+
```

1)I need to find number of students studying physics.



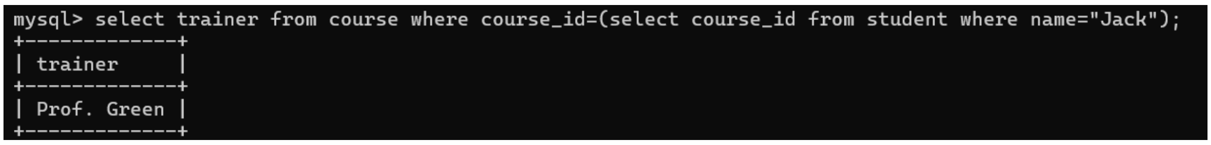
```
mysql> select c.name, count(s.name) from student s inner join course c on s.course_id=c.course_id where c.name="Physics";
+---------+---------------+
| name    | count(s.name) |
+---------+---------------+
| Physics |             3 |
+---------+---------------+
1 row in set (0.01 sec)

mysql> select count(id) from student where course_id=(select course_id from course where name="physics");
+-----------+
| count(id) |
+-----------+
|         3 |
+-----------+
```

2)I need to find how many students studying under dr. brown.



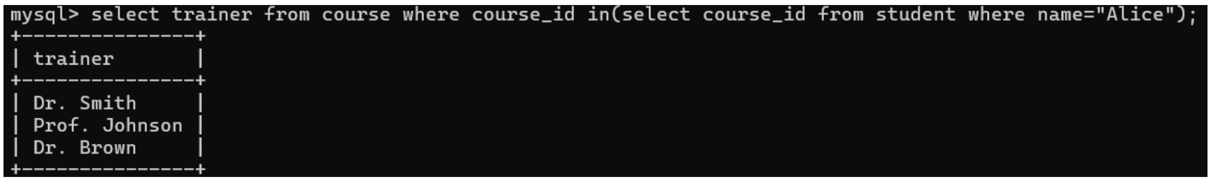
```
mysql> select count(id) from student where course_id=(select course_id from course where trainer="Dr. Brown");
+-----------+
| count(id) |
+-----------+
|         3 |
+-----------+
```

3)I need trainer name for student Jack.



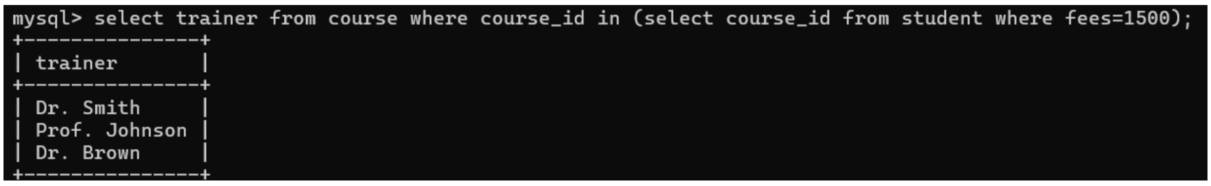
```
mysql> select trainer from course where course_id=(select course_id from student where name="Jack");
+-------------+
| trainer     |
+-------------+
| Prof. Green |
+-------------+
```

4)I need trainer for alice. (It returns more than 1 value so w use in instead of =)



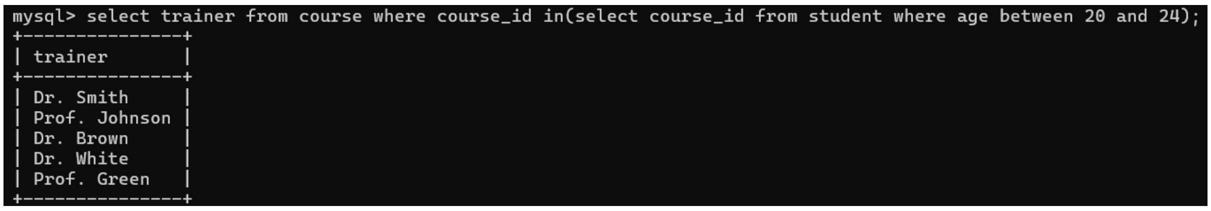
```
mysql> select trainer from course where course_id in(select course_id from student where name="Alice");
+---------------+
| trainer       |
+---------------+
| Dr. Smith     |
| Prof. Johnson |
| Dr. Brown     |
+---------------+
```

5) I need to find trainer name who is collecting fees 1500.



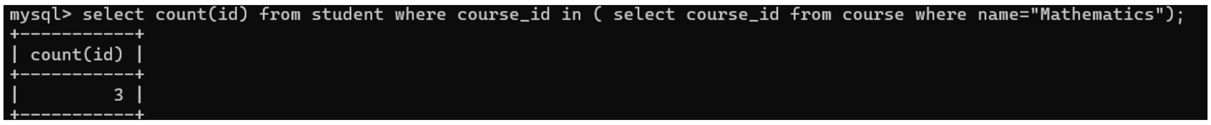
```
mysql> select trainer from course where course_id in (select course_id from student where fees=1500);
+----------------+
| trainer        |
+----------------+
| Dr. Smith      |
| Prof. Johnson  |
| Dr. Brown      |
+----------------+
```

6)I need to find trainer name who is teaching the student whose age is between 20 to 24.



```
mysql> select trainer from course where course_id in(select course_id from student where age between 20 and 24);
+---------------+
| trainer       |
+---------------+
| Dr. Smith     |
| Prof. Johnson |
| Dr. Brown     |
| Dr. White     |
| Prof. Green   |
+---------------+
```

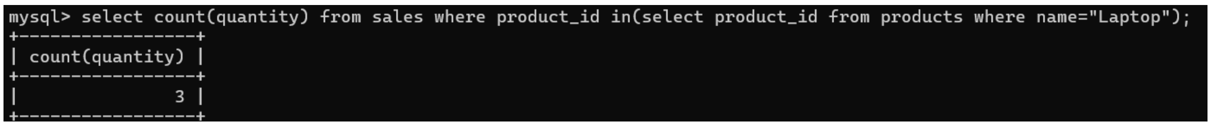
7)I need to find how many student study maths.



```
mysql> select count(id) from student where course_id in ( select course_id from course where name="Mathematics");
+-----------+
| count(id) |
+-----------+
|         3 |
+-----------+
```

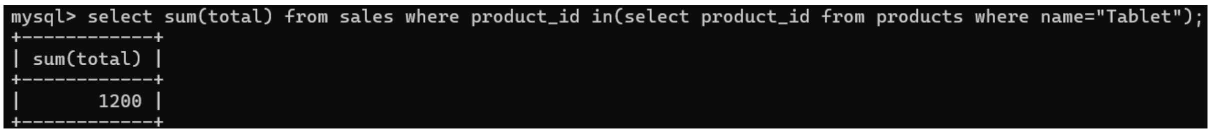
TASK

1)I need total quantity sold for laptop.



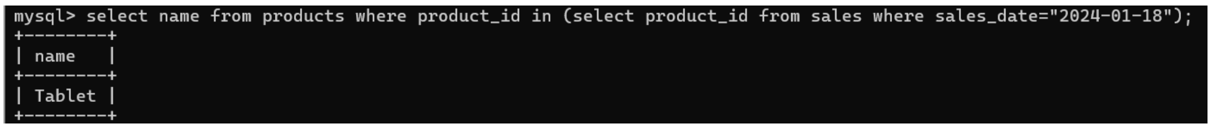
```
mysql> select count(quantity) from sales where product_id in(select product_id from products where name="Laptop");
+-----------------+
| count(quantity) |
+-----------------+
|               3 |
+-----------------+
```

2)I need total sales for tablet.



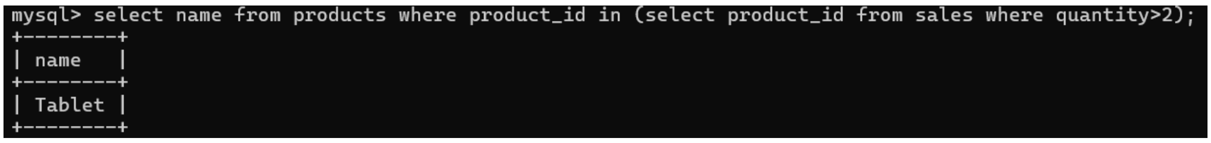
```
mysql> select sum(total) from sales where product_id in(select product_id from products where name="Tablet");
+------------+
| sum(total) |
+------------+
|       1200 |
+------------+
```

3)I need product name that sold out on 2024-01-18.



```
mysql> select name from products where product_id in (select product_id from sales where sales_date="2024-01-18");
+--------+
| name   |
+--------+
| Tablet |
+--------+
```

4)I need product name sold out more than quantity 2.



```
mysql> select name from products where product_id in (select product_id from sales where quantity>2);
+--------+
| name   |
+--------+
| Tablet |
+--------+
```

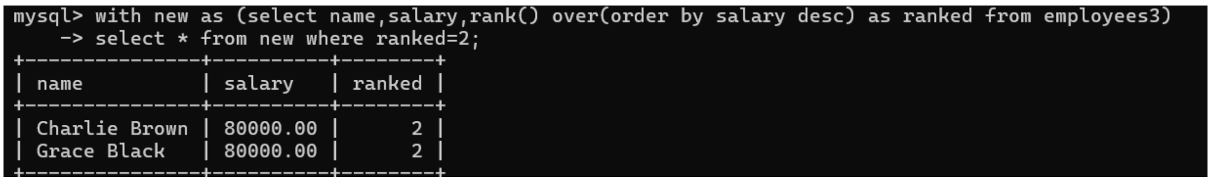
DAY 11

CTE-Common Table Expression:

It is a temporary table

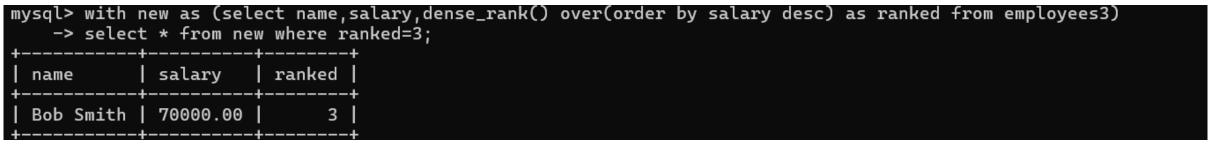
It is created by using WITH clause

1)I need rank 2 from employees3:



```
mysql> with new as (select name,salary,rank() over(order by salary desc) as ranked from employees3)
    -> select * from new where ranked=2;
+---------------+----------+--------+
| name          | salary   | ranked |
+---------------+----------+--------+
| Charlie Brown | 80000.00 |      2 |
| Grace Black   | 80000.00 |      2 |
+---------------+----------+--------+
```

2)Rank 3



```
mysql> with new as (select name,salary,dense_rank() over(order by salary desc) as ranked from employees3)
    -> select * from new where ranked=3;
+-----------+----------+--------+
| name      | salary   | ranked |
+-----------+----------+--------+
| Bob Smith | 70000.00 |      3 |
+-----------+----------+--------+
```

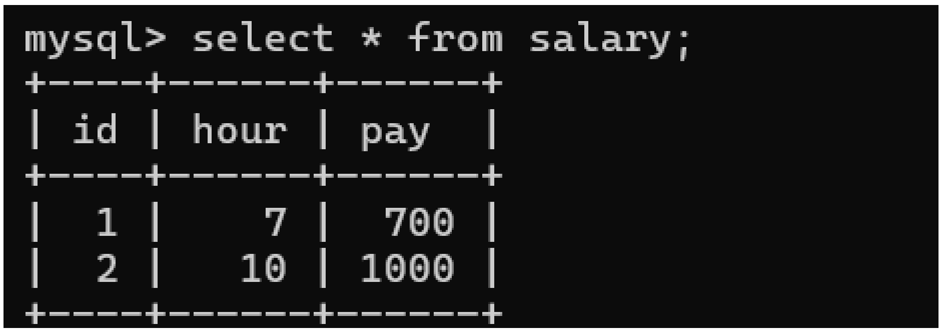
TRIGGER:

To do multiple operations or events at the same time.

Using "create" to create triggers

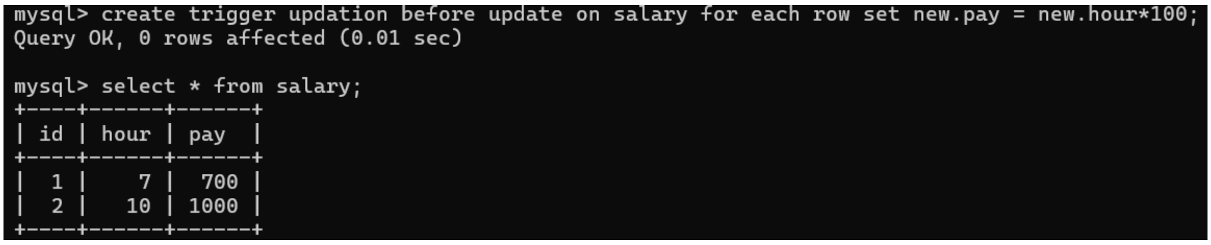
Keywords: NEW OLD BEFORE AFTER

Query: Create trigger sugan before update on salary for each row set new.pay = new.hour*100;



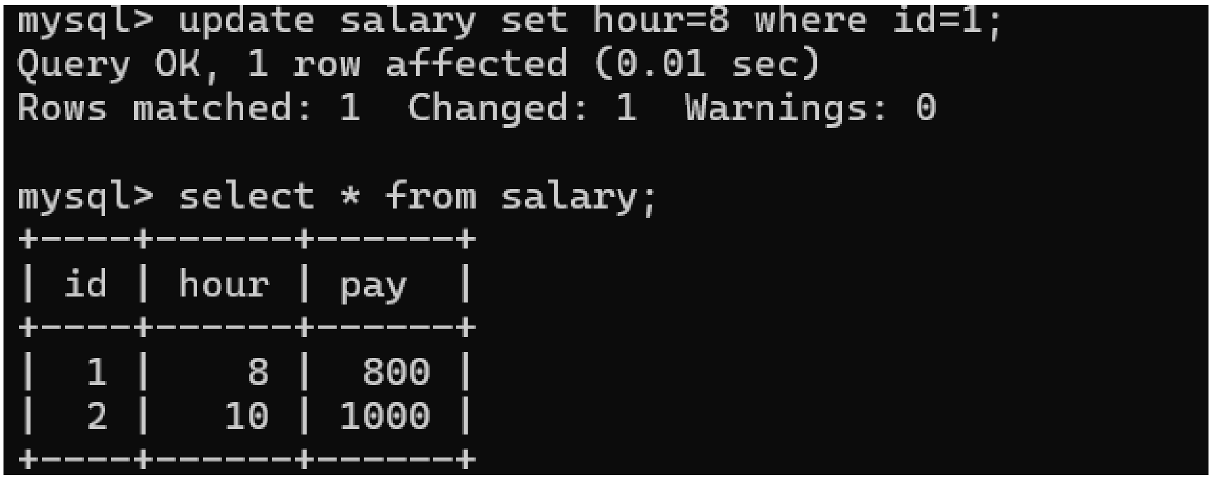
```
mysql> select * from salary;
+----+------+------+
| id | hour | pay  |
+----+------+------+
|  1 |    7 |  700 |
|  2 |   10 | 1000 |
+----+------+------+
```

CREATING TRIGGER

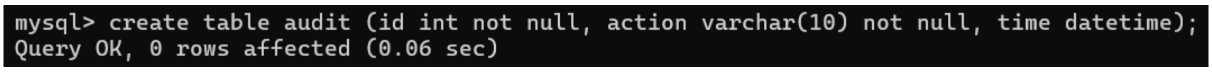


```
mysql> create trigger updation before update on salary for each row set new.pay = new.hour*100;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from salary;
+----+------+------+
| id | hour | pay  |
+----+------+------+
|  1 |    7 |  700 |
|  2 |   10 | 1000 |
+----+------+------+
```

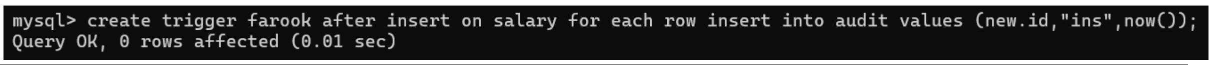
Updating the values:



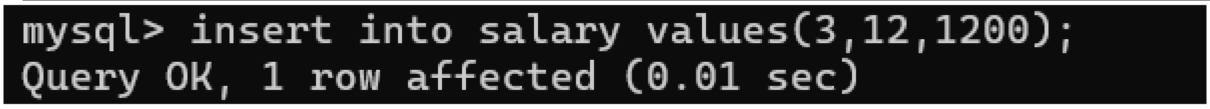
```
mysql> update salary set hour=8 where id=1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from salary;
+----+------+------+
| id | hour | pay  |
+----+------+------+
|  1 |    8 |  800 |
|  2 |   10 | 1000 |
+----+------+------+
```



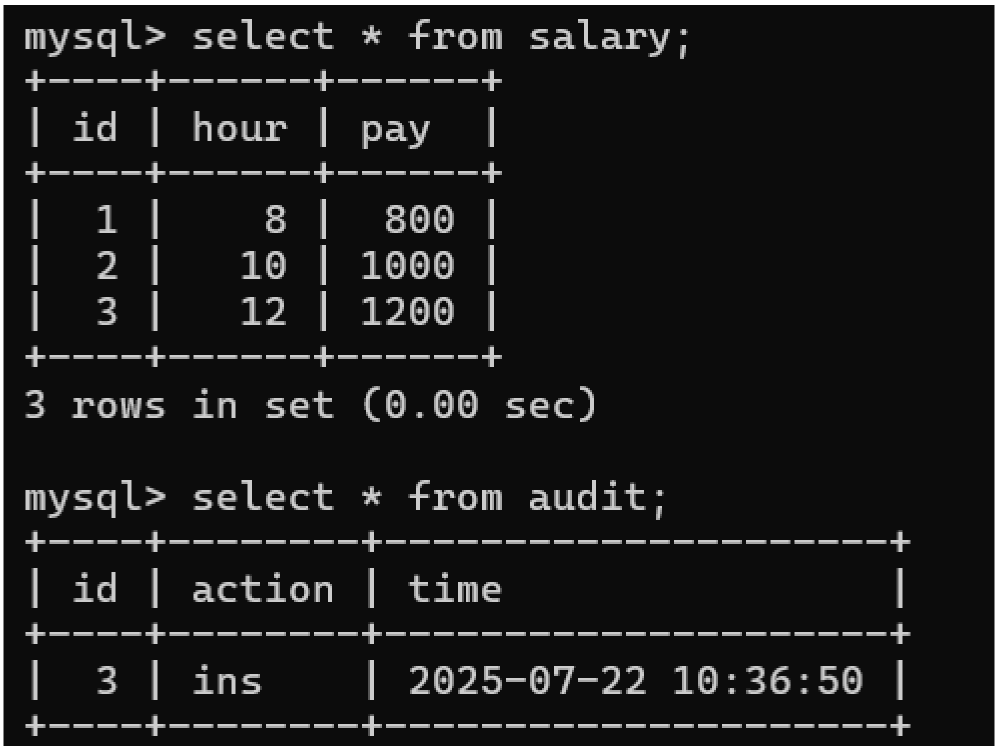
```
mysql> create table audit (id int not null, action varchar(10) not null, time datetime);
Query OK, 0 rows affected (0.06 sec)
```



```
mysql> create trigger farook after insert on salary for each row insert into audit values (new.id,"ins",now());
Query OK, 0 rows affected (0.01 sec)
```



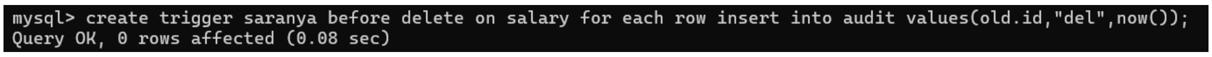
```
mysql> insert into salary values(3,12,1200);
Query OK, 1 row affected (0.01 sec)
```



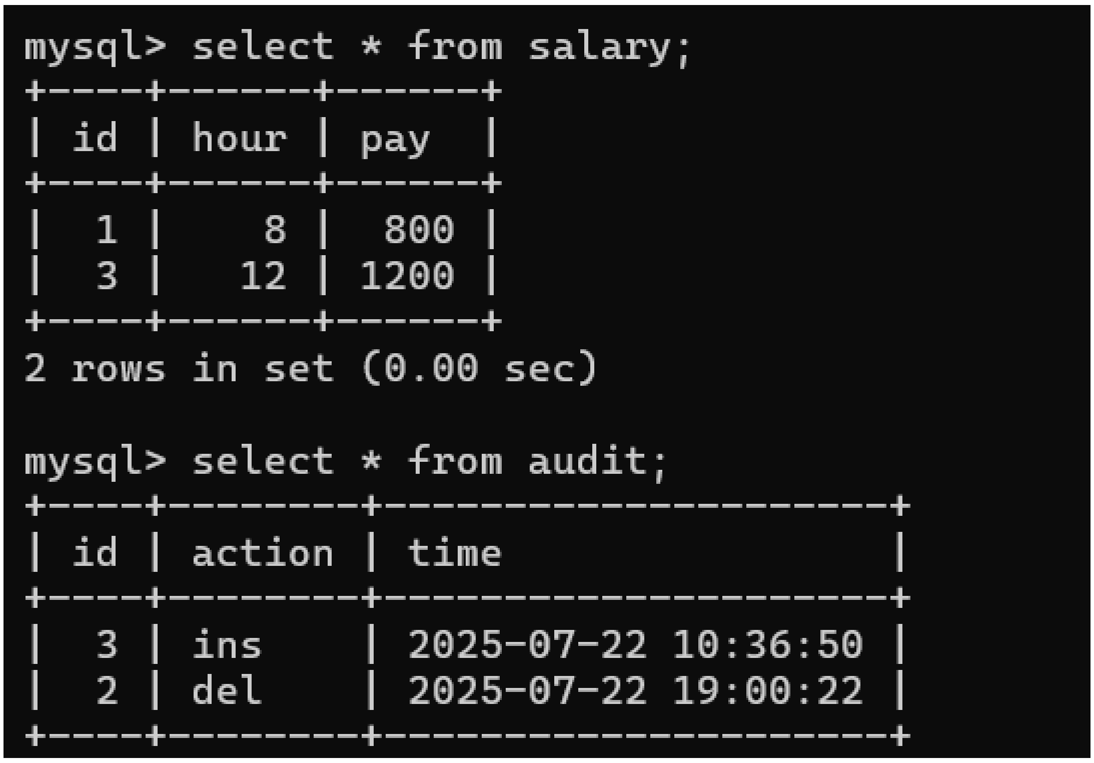
```
mysql> select * from salary;
+----+------+------+
| id | hour | pay  |
+----+------+------+
|  1 |    8 |  800 |
|  2 |   10 | 1000 |
|  3 |   12 | 1200 |
+----+------+------+
3 rows in set (0.00 sec)

mysql> select * from audit;
+----+--------+---------------------+
| id | action | time                |
+----+--------+---------------------+
|  3 | ins    | 2025-07-22 10:36:50 |
+----+--------+---------------------+
```

TASK

To delete a row and view it in audit table:



```
mysql> create trigger saranya before delete on salary for each row insert into audit values(old.id,"del",now());
Query OK, 0 rows affected (0.08 sec)
```



```
mysql> select * from salary;
+----+------+------+
| id | hour | pay  |
+----+------+------+
|  1 |    8 |  800 |
|  3 |   12 | 1200 |
+----+------+------+
2 rows in set (0.00 sec)

mysql> select * from audit;
+----+--------+---------------------+
| id | action | time                |
+----+--------+---------------------+
|  3 | ins    | 2025-07-22 10:36:50 |
|  2 | del    | 2025-07-22 19:00:22 |
+----+--------+---------------------+
```

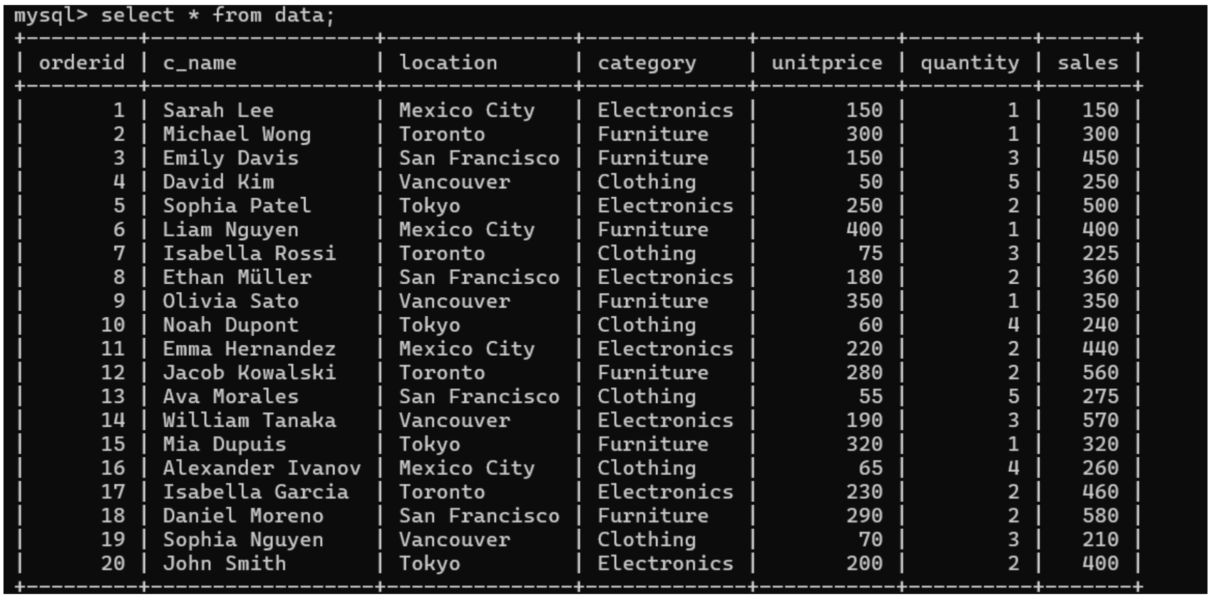
DAY 12:

VIEW - VIRTUAL TABLE

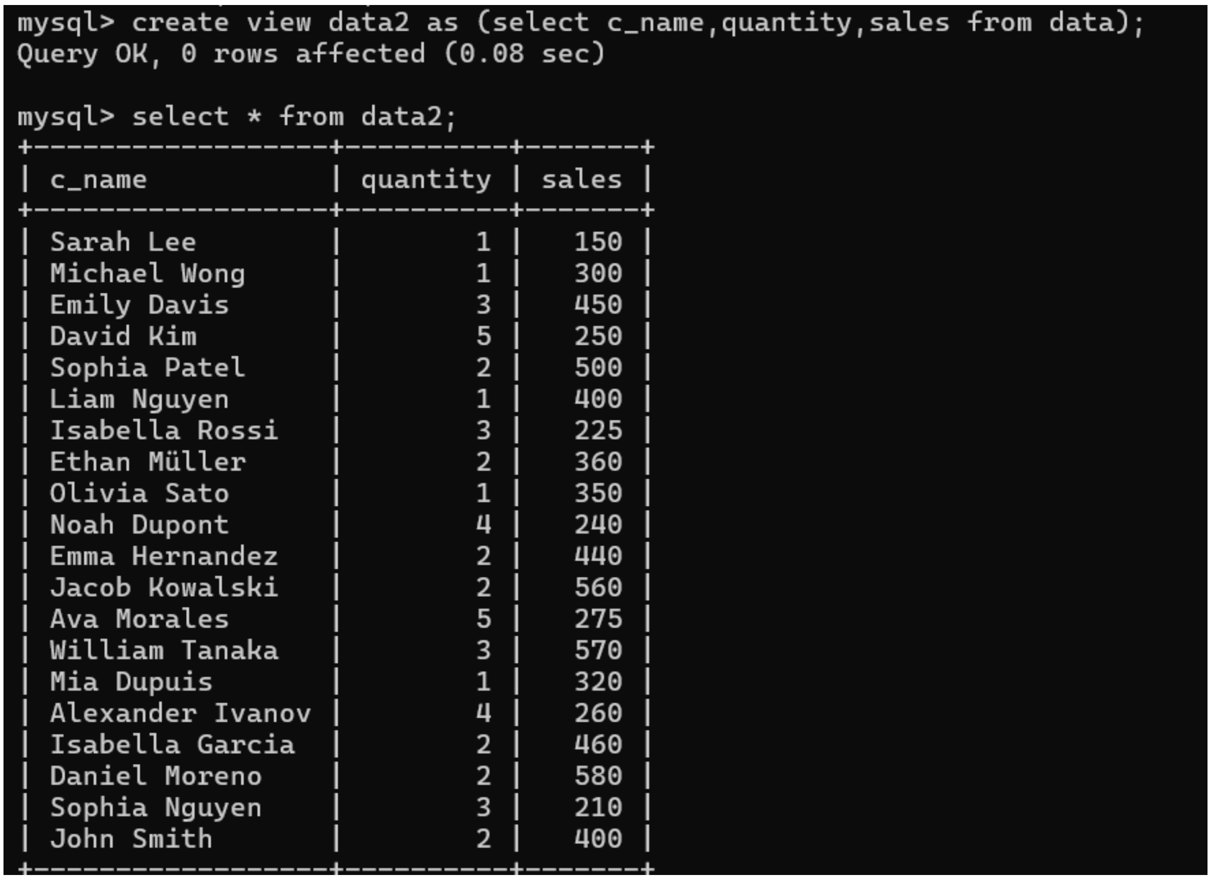
Using CREATE we can create view

Used for security purpose

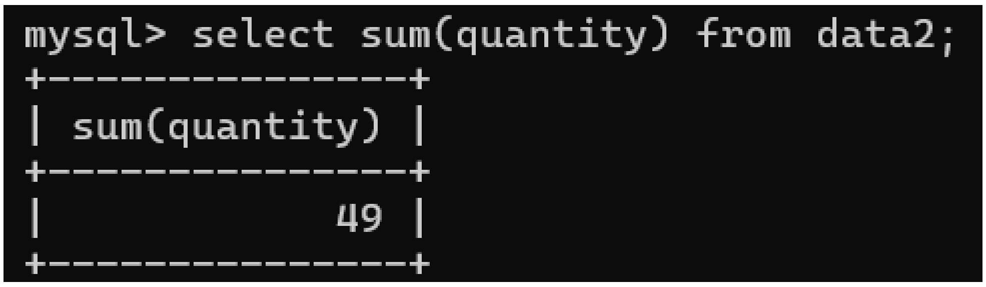
We can retrieve the columns we needed



```
mysql> select * from data;
+---------+------------------+---------------+-------------+-----------+----------+-------+
| orderid | c_name           | location      | category    | unitprice | quantity | sales |
+---------+------------------+---------------+-------------+-----------+----------+-------+
|       1 | Sarah Lee        | Mexico City   | Electronics |       150 |        1 |   150 |
|       2 | Michael Wong     | Toronto       | Furniture   |       300 |        1 |   300 |
|       3 | Emily Davis      | San Francisco | Furniture   |       150 |        3 |   450 |
|       4 | David Kim        | Vancouver     | Clothing    |        50 |        5 |   250 |
|       5 | Sophia Patel     | Tokyo         | Electronics |       250 |        2 |   500 |
|       6 | Liam Nguyen      | Mexico City   | Furniture   |       400 |        1 |   400 |
|       7 | Isabella Rossi   | Toronto       | Clothing    |        75 |        3 |   225 |
|       8 | Ethan Müller     | San Francisco | Electronics |       180 |        2 |   360 |
|       9 | Olivia Sato      | Vancouver     | Furniture   |       350 |        1 |   350 |
|      10 | Noah Dupont      | Tokyo         | Clothing    |        60 |        4 |   240 |
|      11 | Emma Hernandez   | Mexico City   | Electronics |       220 |        2 |   440 |
|      12 | Jacob Kowalski   | Toronto       | Furniture   |       280 |        2 |   560 |
|      13 | Ava Morales      | San Francisco | Clothing    |        55 |        5 |   275 |
|      14 | William Tanaka   | Vancouver     | Electronics |       190 |        3 |   570 |
|      15 | Mia Dupuis       | Tokyo         | Furniture   |       320 |        1 |   320 |
|      16 | Alexander Ivanov | Mexico City   | Clothing    |        65 |        4 |   260 |
|      17 | Isabella Garcia  | Toronto       | Electronics |       230 |        2 |   460 |
|      18 | Daniel Moreno    | San Francisco | Furniture   |       290 |        2 |   580 |
|      19 | Sophia Nguyen    | Vancouver     | Clothing    |        70 |        3 |   210 |
|      20 | John Smith       | Tokyo         | Electronics |       200 |        2 |   400 |
+---------+------------------+---------------+-------------+-----------+----------+-------+
```

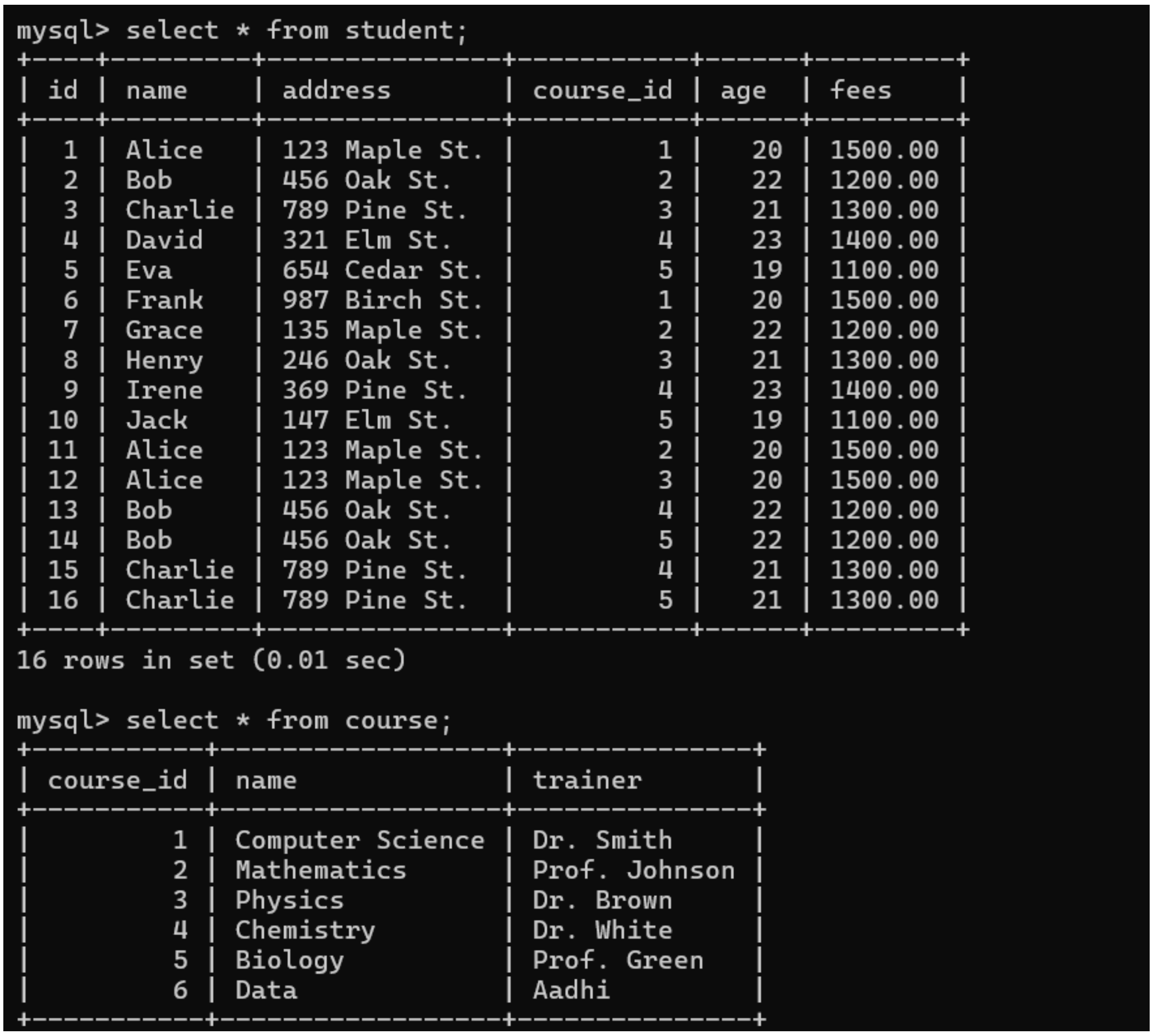


```
mysql> create view data2 as (select c_name,quantity,sales from data);
Query OK, 0 rows affected (0.08 sec)

mysql> select * from data2;
+-------------------+----------+-------+
| c_name            | quantity | sales |
+-------------------+----------+-------+
| Sarah Lee         |        1 |   150 |
| Michael Wong      |        1 |   300 |
| Emily Davis       |        3 |   450 |
| David Kim         |        5 |   250 |
| Sophia Patel      |        2 |   500 |
| Liam Nguyen       |        1 |   400 |
| Isabella Rossi    |        3 |   225 |
| Ethan Müller      |        2 |   360 |
| Olivia Sato       |        1 |   350 |
| Noah Dupont       |        4 |   240 |
| Emma Hernandez    |        2 |   440 |
| Jacob Kowalski    |        2 |   560 |
| Ava Morales       |        5 |   275 |
| William Tanaka    |        3 |   570 |
| Mia Dupuis        |        1 |   320 |
| Alexander Ivanov  |        4 |   260 |
| Isabella Garcia   |        2 |   460 |
| Daniel Moreno     |        2 |   580 |
| Sophia Nguyen     |        3 |   210 |
| John Smith        |        2 |   400 |
+-------------------+----------+-------+
```



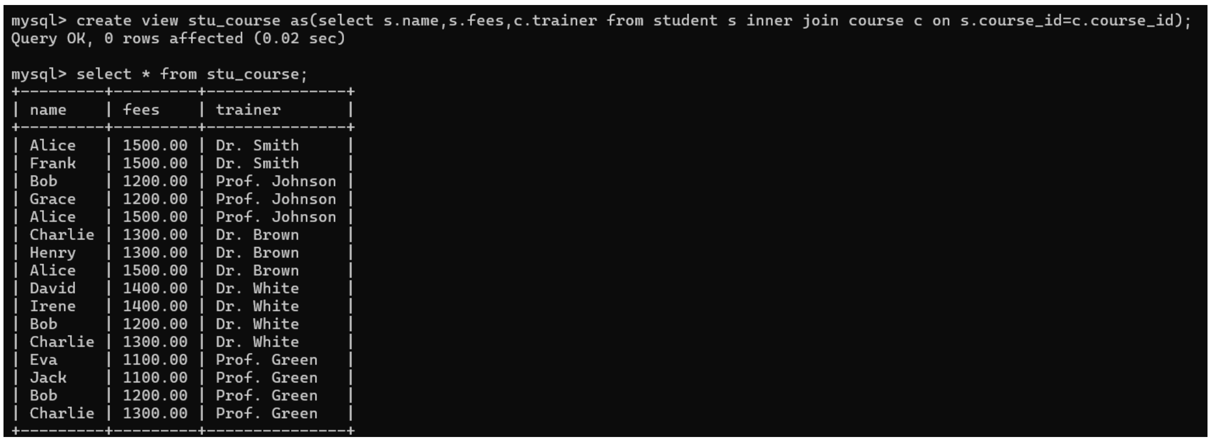
```
mysql> select sum(quantity) from data2;
+---------------+
| sum(quantity) |
+---------------+
|            49 |
+---------------+
```

VIEWING TE TABLE



```
mysql> select * from student;
+----+---------+---------------+-----------+------+---------+
| id | name    | address       | course_id | age  | fees    |
+----+---------+---------------+-----------+------+---------+
|  1 | Alice   | 123 Maple St. |         1 |   20 | 1500.00 |
|  2 | Bob     | 456 Oak St.   |         2 |   22 | 1200.00 |
|  3 | Charlie | 789 Pine St.  |         3 |   21 | 1300.00 |
|  4 | David   | 321 Elm St.   |         4 |   23 | 1400.00 |
|  5 | Eva     | 654 Cedar St. |         5 |   19 | 1100.00 |
|  6 | Frank   | 987 Birch St. |         1 |   20 | 1500.00 |
|  7 | Grace   | 135 Maple St. |         2 |   22 | 1200.00 |
|  8 | Henry   | 246 Oak St.   |         3 |   21 | 1300.00 |
|  9 | Irene   | 369 Pine St.  |         4 |   23 | 1400.00 |
| 10 | Jack    | 147 Elm St.   |         5 |   19 | 1100.00 |
| 11 | Alice   | 123 Maple St. |         2 |   20 | 1500.00 |
| 12 | Alice   | 123 Maple St. |         3 |   20 | 1500.00 |
| 13 | Bob     | 456 Oak St.   |         4 |   22 | 1200.00 |
| 14 | Bob     | 456 Oak St.   |         5 |   22 | 1200.00 |
| 15 | Charlie | 789 Pine St.  |         4 |   21 | 1300.00 |
| 16 | Charlie | 789 Pine St.  |         5 |   21 | 1300.00 |
+----+---------+---------------+-----------+------+---------+
16 rows in set (0.01 sec)

mysql> select * from course;
+-----------+-------------------+---------------+
| course_id | name              | trainer       |
+-----------+-------------------+---------------+
|         1 | Computer Science  | Dr. Smith     |
|         2 | Mathematics       | Prof. Johnson |
|         3 | Physics           | Dr. Brown     |
|         4 | Chemistry         | Dr. White     |
|         5 | Biology           | Prof. Green   |
|         6 | Data              | Aadhi         |
+-----------+-------------------+---------------+
```

We can retrive the needed columns



```
mysql> create view stu_course as(select s.name,s.fees,c.trainer from student s inner join course c on s.course_id=c.course_id);
Query OK, 0 rows affected (0.02 sec)

mysql> select * from stu_course;
+---------+---------+---------------+
| name    | fees    | trainer       |
+---------+---------+---------------+
| Alice   | 1500.00 | Dr. Smith     |
| Frank   | 1500.00 | Dr. Smith     |
| Bob     | 1200.00 | Prof. Johnson |
| Grace   | 1200.00 | Prof. Johnson |
| Alice   | 1500.00 | Prof. Johnson |
| Charlie | 1300.00 | Dr. Brown     |
| Henry   | 1300.00 | Dr. Brown     |
| Alice   | 1500.00 | Dr. Brown     |
| David   | 1400.00 | Dr. White     |
| Irene   | 1400.00 | Dr. White     |
| Bob     | 1200.00 | Dr. White     |
| Charlie | 1300.00 | Dr. White     |
| Eva     | 1100.00 | Prof. Green   |
| Jack    | 1100.00 | Prof. Green   |
| Bob     | 1200.00 | Prof. Green   |
| Charlie | 1300.00 | Prof. Green   |
+---------+---------+---------------+
```

SET OPERATORS

Combines the result of multiple SQL Queries into single result set

- UNION
- UNION ALL
- INTERSECT
- EXCEPT

UNION remove duplicates combining two tables.

UNION ALL: It will give duplicates

INTERSECT: Gives common records

EXCEPT: No common records and give record on left table.

TABLE:

create table first(name char(10) not null,age int not null,salary int);

create table second(name char(10) not null,age int not null,salary int);

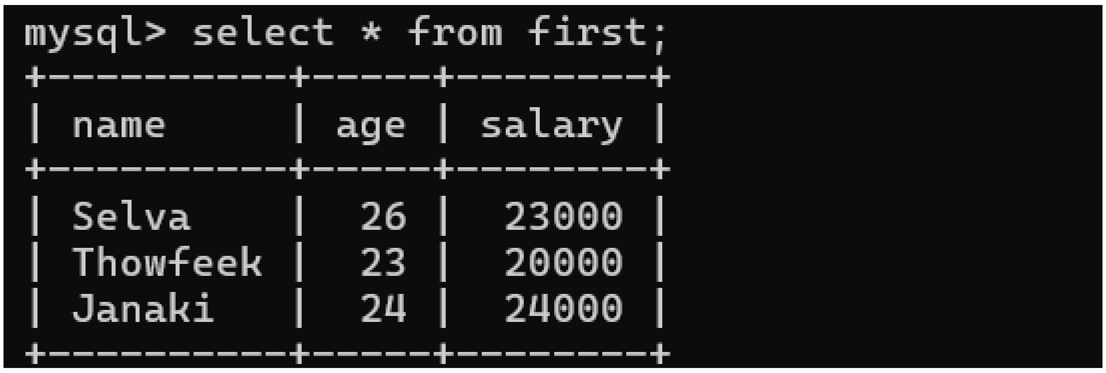
create table three(name char(10) not null,age int not null,salary int);

insert into first values("Selva",26,23000),("Thowfeek",23,20000),("Janaki",24,24000);

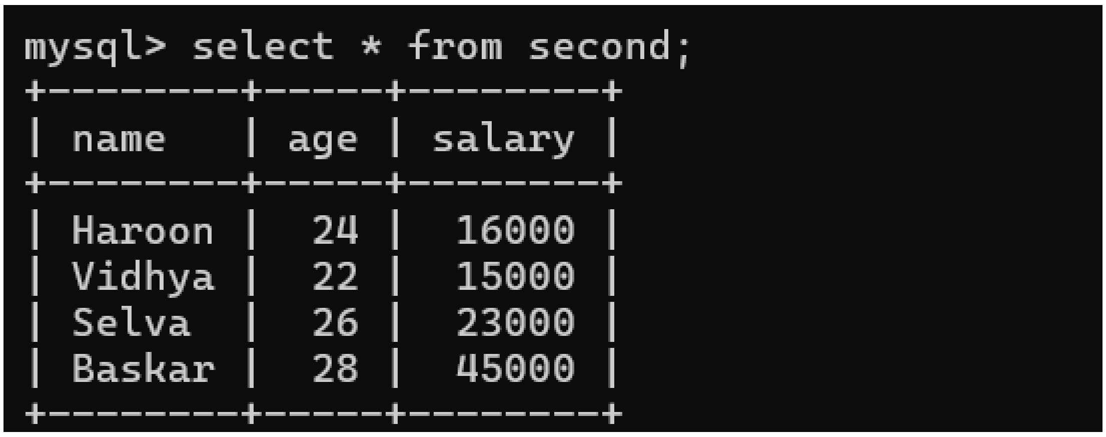
insert into Second values("Haroon",24,16000),("Vidhya",22,15000),("Selva",26,23000),("Baskar",28,45000);

insert into Three values('Mani',22,17000),('Lavanya',23,18000),('Nitheesh',24,19000);

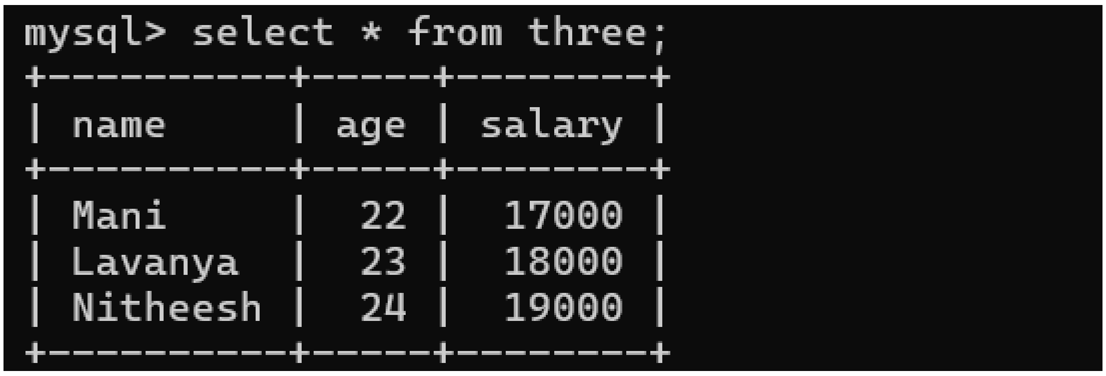
VIEWING TABLES:



```
mysql> select * from first;
+----------+-----+--------+
| name     | age | salary |
+----------+-----+--------+
| Selva    |  26 |  23000 |
| Thowfeek |  23 |  20000 |
| Janaki   |  24 |  24000 |
+----------+-----+--------+
```

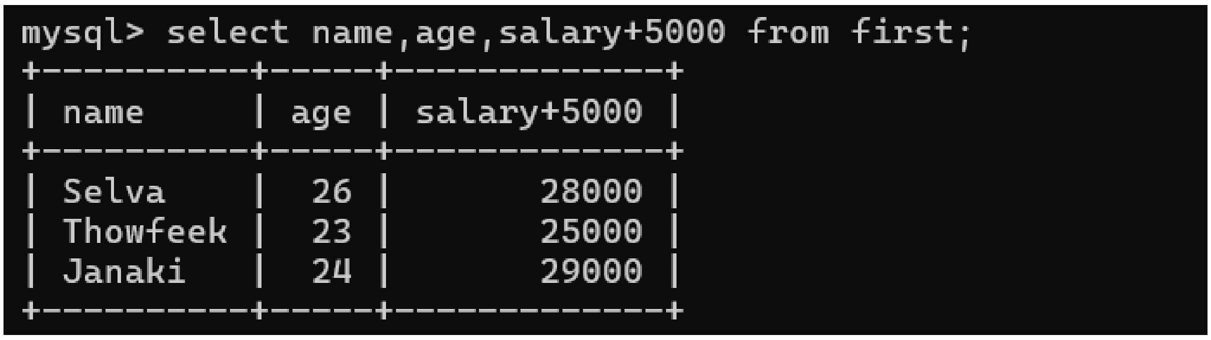


```
mysql> select * from second;
+--------+-----+--------+
| name   | age | salary |
+--------+-----+--------+
| Haroon |  24 |  16000 |
| Vidhya |  22 |  15000 |
| Selva  |  26 |  23000 |
| Baskar |  28 |  45000 |
+--------+-----+--------+
```

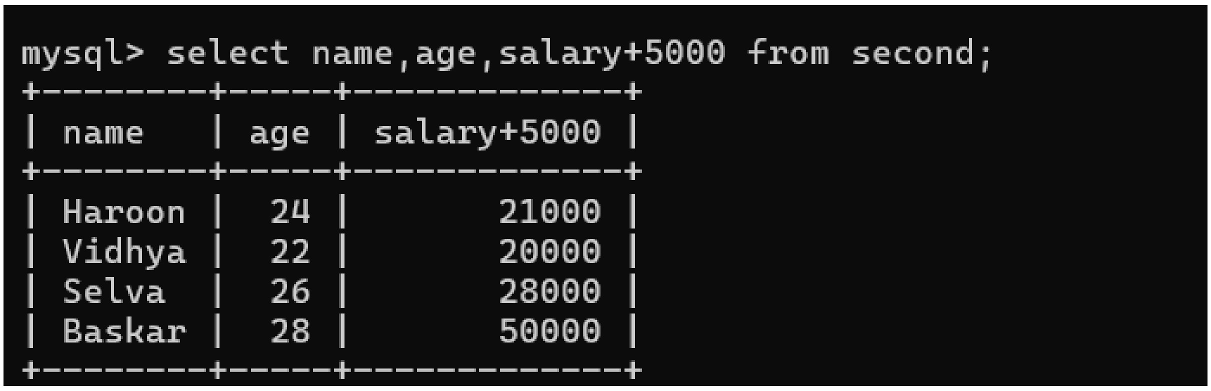


```
mysql> select * from three;
+----------+-----+--------+
| name     | age | salary |
+----------+-----+--------+
| Mani     |  22 |  17000 |
| Lavanya  |  23 |  18000 |
| Nitheesh |  24 |  19000 |
+----------+-----+--------+
```

Selecting and making operations:

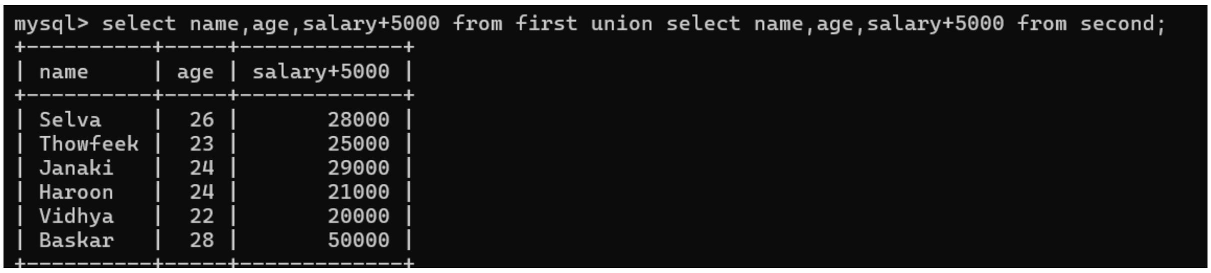


```
mysql> select name,age,salary+5000 from first;
+----------+-----+-------------+
| name     | age | salary+5000 |
+----------+-----+-------------+
| Selva    |  26 |       28000 |
| Thowfeek |  23 |       25000 |
| Janaki   |  24 |       29000 |
+----------+-----+-------------+
```



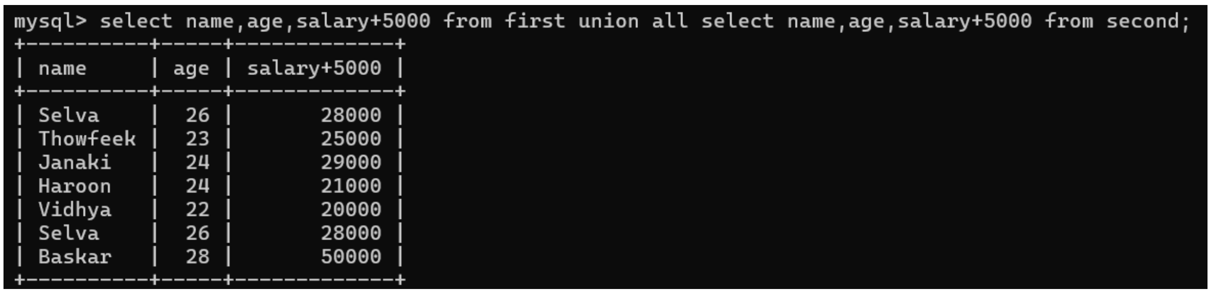
```
mysql> select name,age,salary+5000 from second;
+--------+-----+-------------+
| name   | age | salary+5000 |
+--------+-----+-------------+
| Haroon |  24 |       21000 |
| Vidhya |  22 |       20000 |
| Selva  |  26 |       28000 |
| Baskar |  28 |       50000 |
+--------+-----+-------------+
```

UNION: 1&2 table: “**Remove duplicates”**



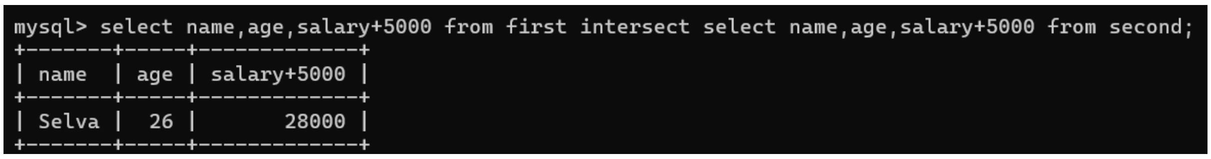
```
mysql> select name,age,salary+5000 from first union select name,age,salary+5000 from second;
+----------+-----+-------------+
| name     | age | salary+5000 |
+----------+-----+-------------+
| Selva    |  26 |       28000 |
| Thowfeek |  23 |       25000 |
| Janaki   |  24 |       29000 |
| Haroon   |  24 |       21000 |
| Vidhya   |  22 |       20000 |
| Baskar   |  28 |       50000 |
+----------+-----+-------------+
```

UNION ALL :1&2 table: **“Give duplicates”**



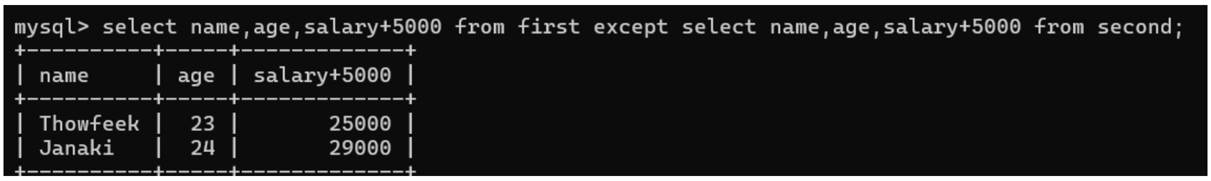
```
mysql> select name,age,salary+5000 from first union all select name,age,salary+5000 from second;
+----------+-----+-------------+
| name     | age | salary+5000 |
+----------+-----+-------------+
| Selva    |  26 |       28000 |
| Thowfeek |  23 |       25000 |
| Janaki   |  24 |       29000 |
| Haroon   |  24 |       21000 |
| Vidhya   |  22 |       20000 |
| Selva    |  26 |       28000 |
| Baskar   |  28 |       50000 |
+----------+-----+-------------+
```

INTERSECT:1&2 table: **“Gives common records only”**



```
mysql> select name,age,salary+5000 from first intersect select name,age,salary+5000 from second;
+-------+-----+-------------+
| name  | age | salary+5000 |
+-------+-----+-------------+
| Selva |  26 |       28000 |
+-------+-----+-------------+
```

EXCEPT:1&2 table: **“Remove common records and give left table”**



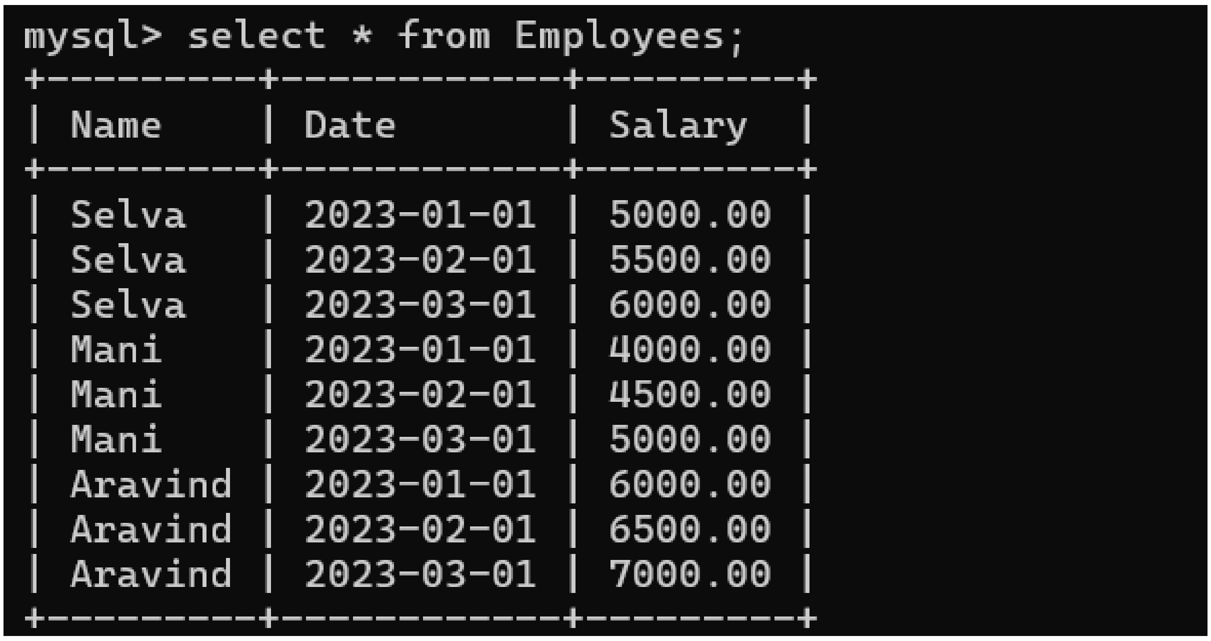
```
mysql> select name,age,salary+5000 from first except select name,age,salary+5000 from second;
+----------+-----+-------------+
| name     | age | salary+5000 |
+----------+-----+-------------+
| Thowfeek |  23 |       25000 |
| Janaki   |  24 |       29000 |
+----------+-----+-------------+
```

LEAD LAG

LEAD-To compare next row in a table

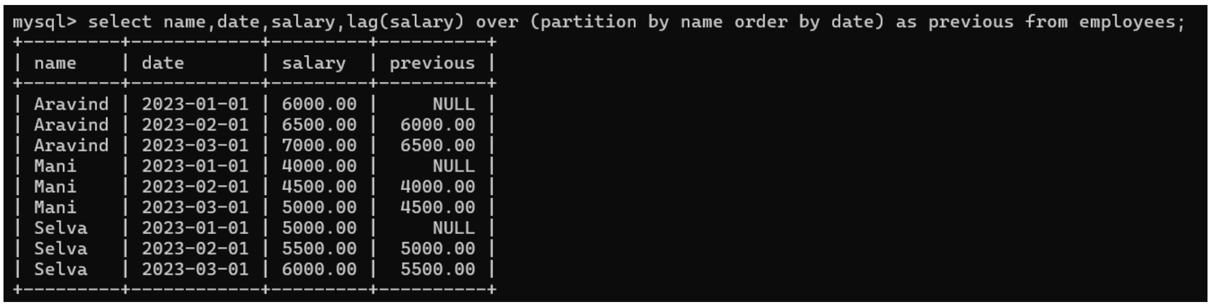
LAG-To compare previous row in a table

Viewing table



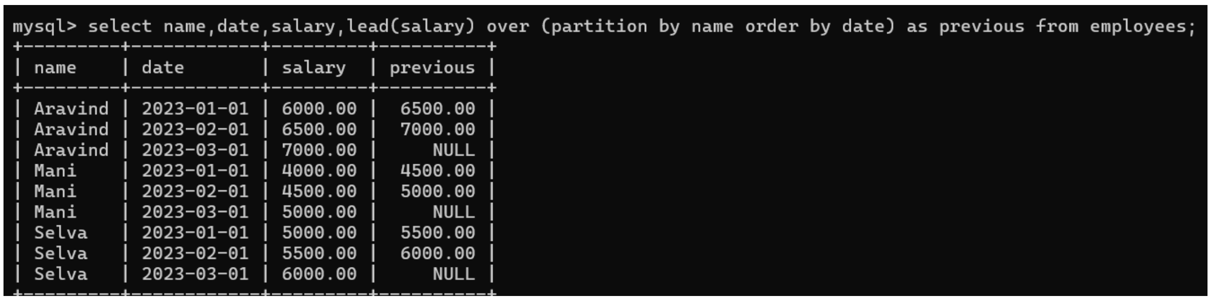
```
mysql> select * from Employees;
+---------+------------+---------+
| Name    | Date       | Salary  |
+---------+------------+---------+
| Selva   | 2023-01-01 | 5000.00 |
| Selva   | 2023-02-01 | 5500.00 |
| Selva   | 2023-03-01 | 6000.00 |
| Mani    | 2023-01-01 | 4000.00 |
| Mani    | 2023-02-01 | 4500.00 |
| Mani    | 2023-03-01 | 5000.00 |
| Aravind | 2023-01-01 | 6000.00 |
| Aravind | 2023-02-01 | 6500.00 |
| Aravind | 2023-03-01 | 7000.00 |
+---------+------------+---------+
```

Using lag :



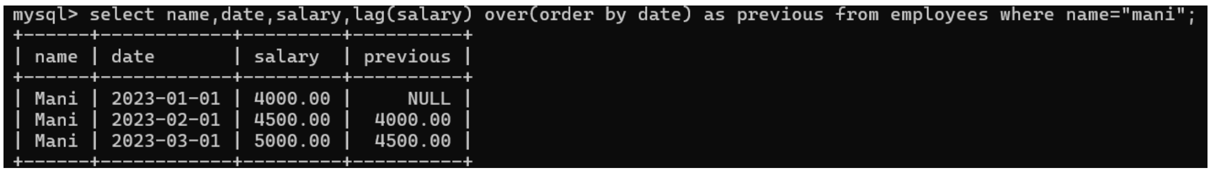
```
mysql> select name,date,salary,lag(salary) over (partition by name order by date) as previous from employees;
+---------+------------+---------+----------+
| name    | date       | salary  | previous |
+---------+------------+---------+----------+
| Aravind | 2023-01-01 | 6000.00 |     NULL |
| Aravind | 2023-02-01 | 6500.00 |  6000.00 |
| Aravind | 2023-03-01 | 7000.00 |  6500.00 |
| Mani    | 2023-01-01 | 4000.00 |     NULL |
| Mani    | 2023-02-01 | 4500.00 |  4000.00 |
| Mani    | 2023-03-01 | 5000.00 |  4500.00 |
| Selva   | 2023-01-01 | 5000.00 |     NULL |
| Selva   | 2023-02-01 | 5500.00 |  5000.00 |
| Selva   | 2023-03-01 | 6000.00 |  5500.00 |
+---------+------------+---------+----------+
```

Using Lead:



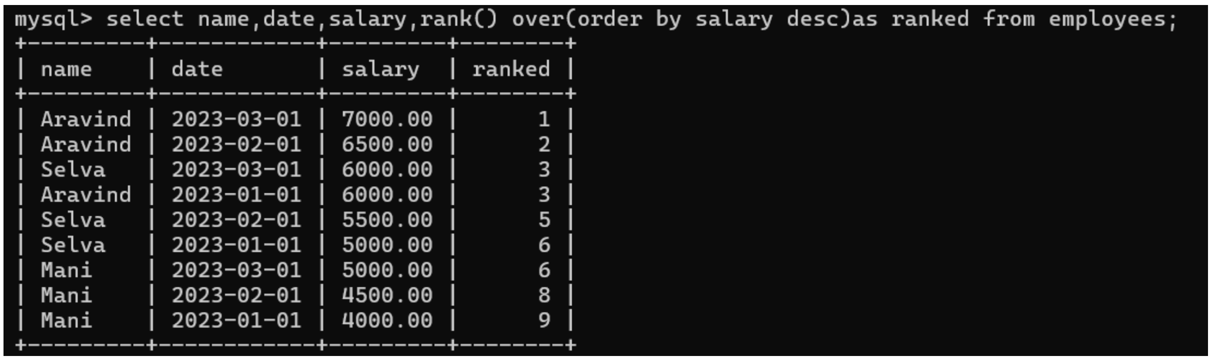
```
mysql> select name,date,salary,lead(salary) over (partition by name order by date) as previous from employees;
+---------+------------+---------+----------+
| name    | date       | salary  | previous |
+---------+------------+---------+----------+
| Aravind | 2023-01-01 | 6000.00 |  6500.00 |
| Aravind | 2023-02-01 | 6500.00 |  7000.00 |
| Aravind | 2023-03-01 | 7000.00 |     NULL |
| Mani    | 2023-01-01 | 4000.00 |  4500.00 |
| Mani    | 2023-02-01 | 4500.00 |  5000.00 |
| Mani    | 2023-03-01 | 5000.00 |     NULL |
| Selva   | 2023-01-01 | 5000.00 |  5500.00 |
| Selva   | 2023-02-01 | 5500.00 |  6000.00 |
| Selva   | 2023-03-01 | 6000.00 |     NULL |
+---------+------------+---------+----------+
```

1)Retriving data of mani

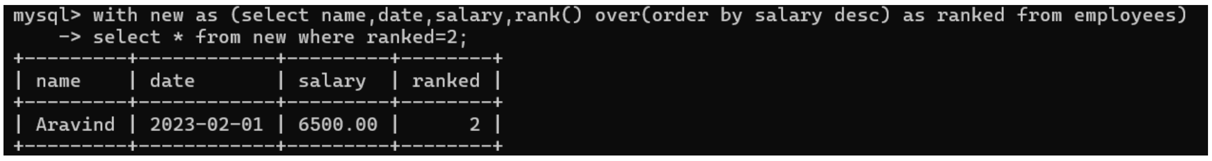


```
mysql> select name,date,salary,lag(salary) over(order by date) as previous from employees where name="mani";
+------+------------+---------+----------+
| name | date       | salary  | previous |
+------+------------+---------+----------+
| Mani | 2023-01-01 | 4000.00 |     NULL |
| Mani | 2023-02-01 | 4500.00 |  4000.00 |
| Mani | 2023-03-01 | 5000.00 |  4500.00 |
+------+------------+---------+----------+
```

2) I need to find the second highest salary getter.(Always use rank and CTE):



```
mysql> select name,date,salary,rank() over(order by salary desc)as ranked from employees;
+---------+------------+---------+--------+
| name    | date       | salary  | ranked |
+---------+------------+---------+--------+
| Aravind | 2023-03-01 | 7000.00 |      1 |
| Aravind | 2023-02-01 | 6500.00 |      2 |
| Selva   | 2023-03-01 | 6000.00 |      3 |
| Aravind | 2023-01-01 | 6000.00 |      3 |
| Selva   | 2023-02-01 | 5500.00 |      5 |
| Selva   | 2023-01-01 | 5000.00 |      6 |
| Mani    | 2023-03-01 | 5000.00 |      6 |
| Mani    | 2023-02-01 | 4500.00 |      8 |
| Mani    | 2023-01-01 | 4000.00 |      9 |
+---------+------------+---------+--------+
```



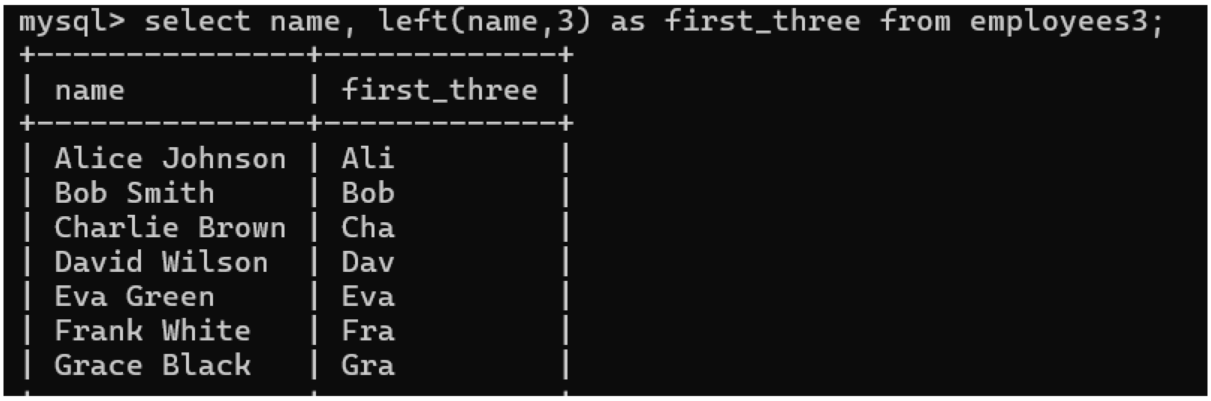
```
mysql> with new as (select name,date,salary,rank() over(order by salary desc) as ranked from employees)
    -> select * from new where ranked=2;
+---------+------------+---------+--------+
| name    | date       | salary  | ranked |
+---------+------------+---------+--------+
| Aravind | 2023-02-01 | 6500.00 |      2 |
+---------+------------+---------+--------+
```

DAY 13

STRING FUNCTIONS

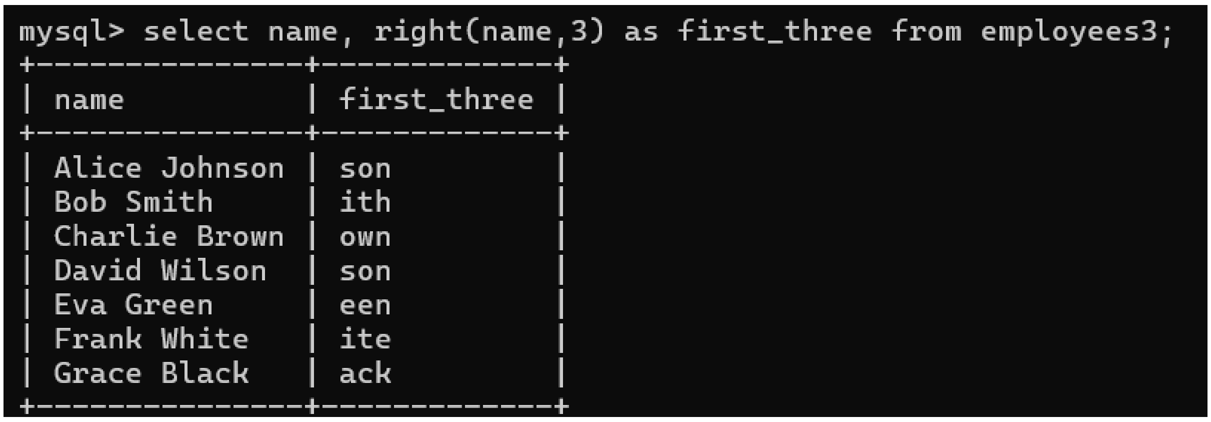
- left
- right
- substring_index
- concat
- replace

1)LEFT() : To get letters from left side:



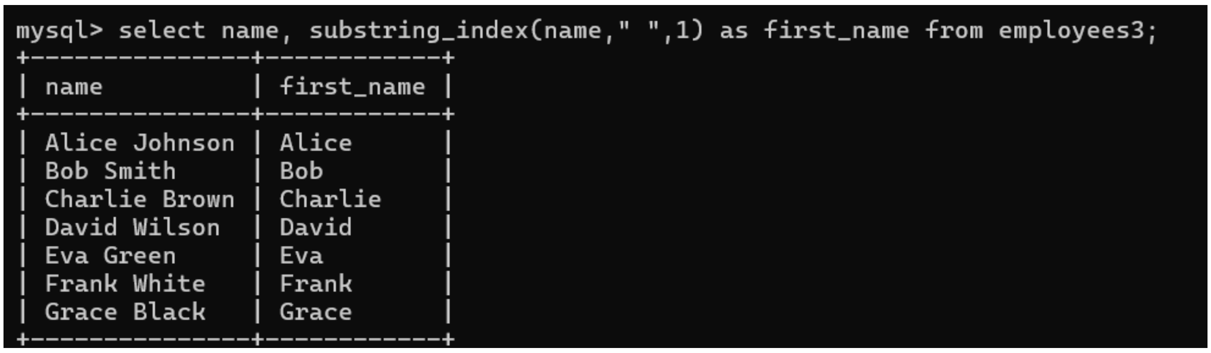
```
mysql> select name, left(name,3) as first_three from employees3;
+---------------+-------------+
| name          | first_three |
+---------------+-------------+
| Alice Johnson | Ali         |
| Bob Smith     | Bob         |
| Charlie Brown | Cha         |
| David Wilson  | Dav         |
| Eva Green     | Eva         |
| Frank White   | Fra         |
| Grace Black   | Gra         |
```

2)RIGHT() : To get last letter of name

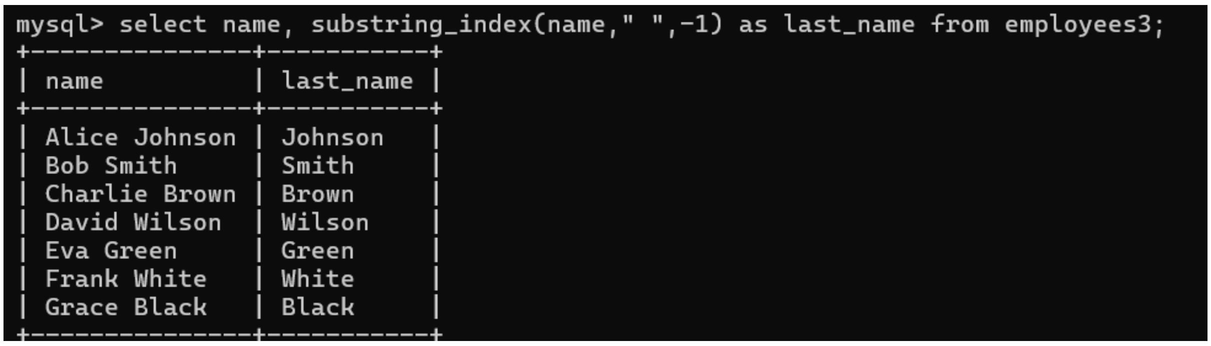


```
mysql> select name, right(name,3) as first_three from employees3;
+---------------+-------------+
| name          | first_three |
+---------------+-------------+
| Alice Johnson | son         |
| Bob Smith     | ith         |
| Charlie Brown | own         |
| David Wilson  | son         |
| Eva Green     | een         |
| Frank White   | ite         |
| Grace Black   | ack         |
+---------------+-------------+
```

3)SUBSTRING_INDEX() : To fetch first name and last name:

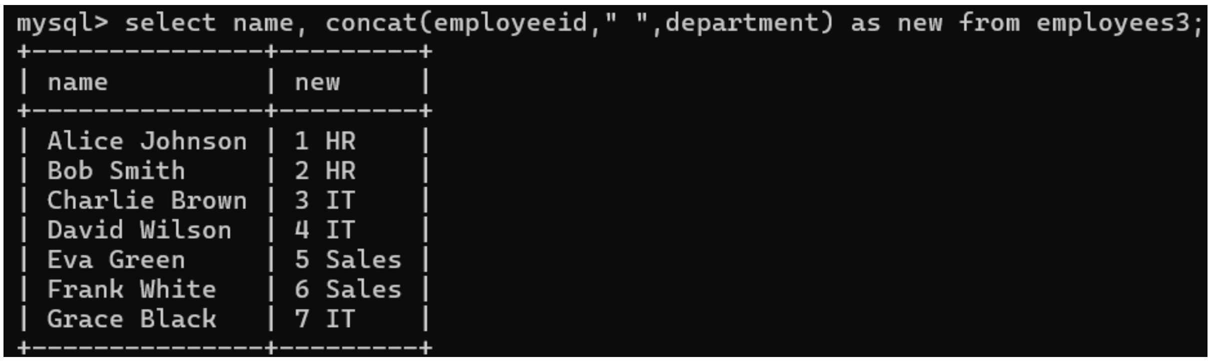


```
mysql> select name, substring_index(name," ",1) as first_name from employees3;
+---------------+------------+
| name          | first_name |
+---------------+------------+
| Alice Johnson | Alice      |
| Bob Smith     | Bob        |
| Charlie Brown | Charlie    |
| David Wilson  | David      |
| Eva Green     | Eva        |
| Frank White   | Frank      |
| Grace Black   | Grace      |
+---------------+------------+
```



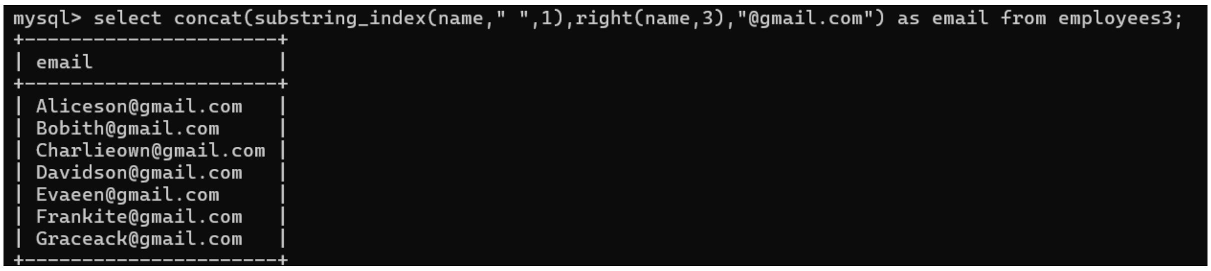
```
mysql> select name, substring_index(name," ",-1) as last_name from employees3;
+---------------+-----------+
| name          | last_name |
+---------------+-----------+
| Alice Johnson | Johnson   |
| Bob Smith     | Smith     |
| Charlie Brown | Brown     |
| David Wilson  | Wilson    |
| Eva Green     | Green     |
| Frank White   | White     |
| Grace Black   | Black     |
+---------------+-----------+
```

4)Concat(): To join two column values into one



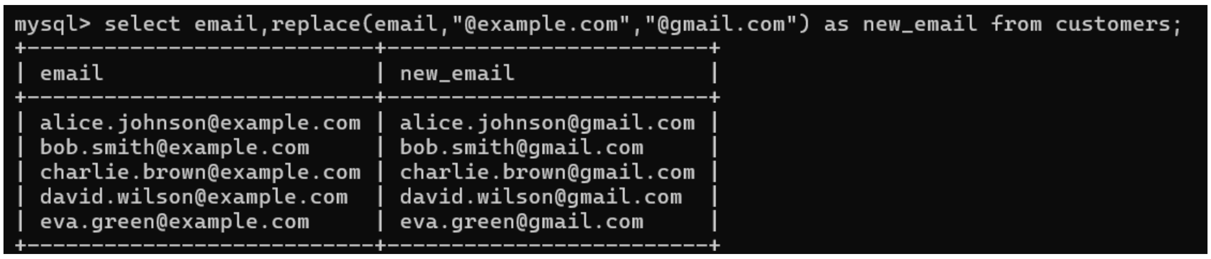
```
mysql> select name, concat(employeeid," ",department) as new from employees3;
+---------------+---------+
| name          | new     |
+---------------+---------+
| Alice Johnson | 1 HR    |
| Bob Smith     | 2 HR    |
| Charlie Brown | 3 IT    |
| David Wilson  | 4 IT    |
| Eva Green     | 5 Sales |
| Frank White   | 6 Sales |
| Grace Black   | 7 IT    |
+---------------+---------+
```

1)I need first name and last three letters with @gmail.com:



```
mysql> select concat(substring_index(name," ",1),right(name,3),"@gmail.com") as email from employees3;
+----------------------+
| email                |
+----------------------+
| Aliceson@gmail.com   |
| Bobith@gmail.com     |
| Charlieown@gmail.com |
| Davidson@gmail.com   |
| Evaeen@gmail.com     |
| Frankite@gmail.com   |
| Graceack@gmail.com   |
+----------------------+
```

5)REPLACE() :To replace the words in the column



```
mysql> select email,replace(email,"@example.com","@gmail.com") as new_email from customers;
+---------------------------+-------------------------+
| email                     | new_email               |
+---------------------------+-------------------------+
| alice.johnson@example.com | alice.johnson@gmail.com |
| bob.smith@example.com     | bob.smith@gmail.com     |
| charlie.brown@example.com | charlie.brown@gmail.com |
| david.wilson@example.com  | david.wilson@gmail.com  |
| eva.green@example.com     | eva.green@gmail.com     |
+---------------------------+-------------------------+
```

DAY 14

STORED PROCEDURE

- In and out parameter

Delimiter ##
Create procedure procedure_name (in id int)
Begin
Select * from data where ordered=id;
End ##
Call procedure_name ();
Delimiter ;

DELIMITER ##
We use DELIMITER ## to avoid confusion with semicolons inside the SQL block.
Tells the tool: "Don't treat ; as the end of a command; wait until you see ##."

After finishing the procedure, we reset to normal with : **DELIMITER ;**

BEGIN

Start a block of SQL statements

END End the block

## End of full SQL command

To declare a variable : set @variable_name=0;