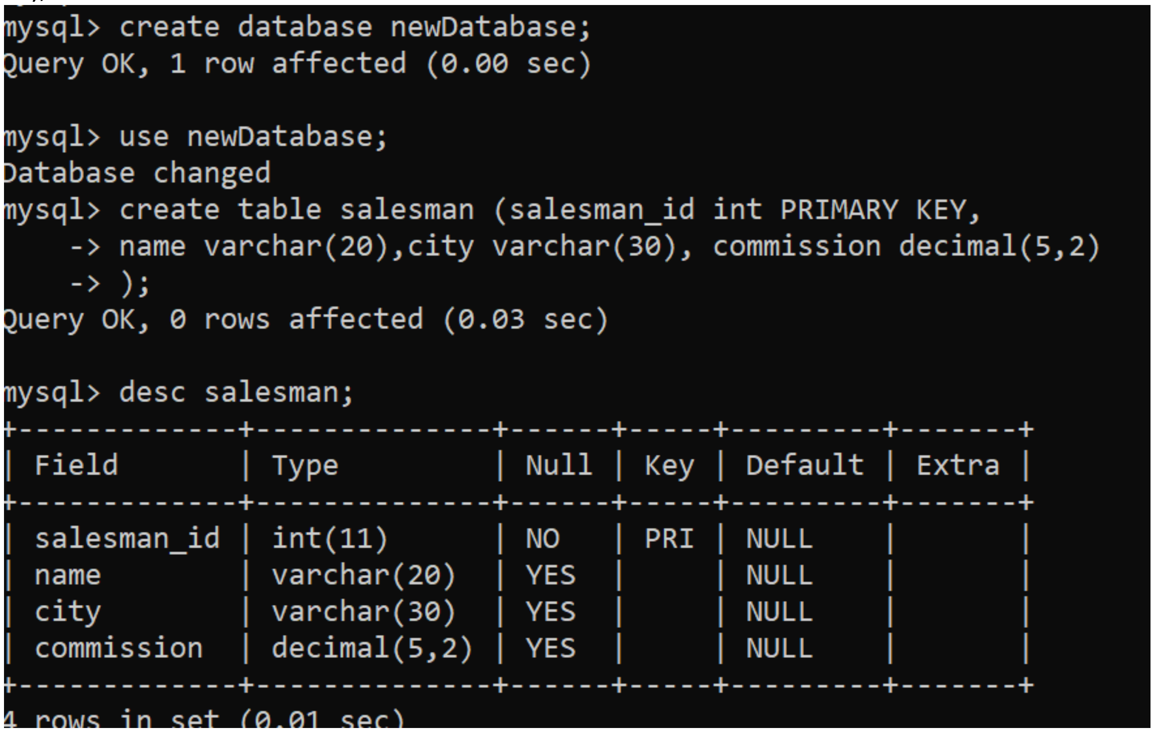
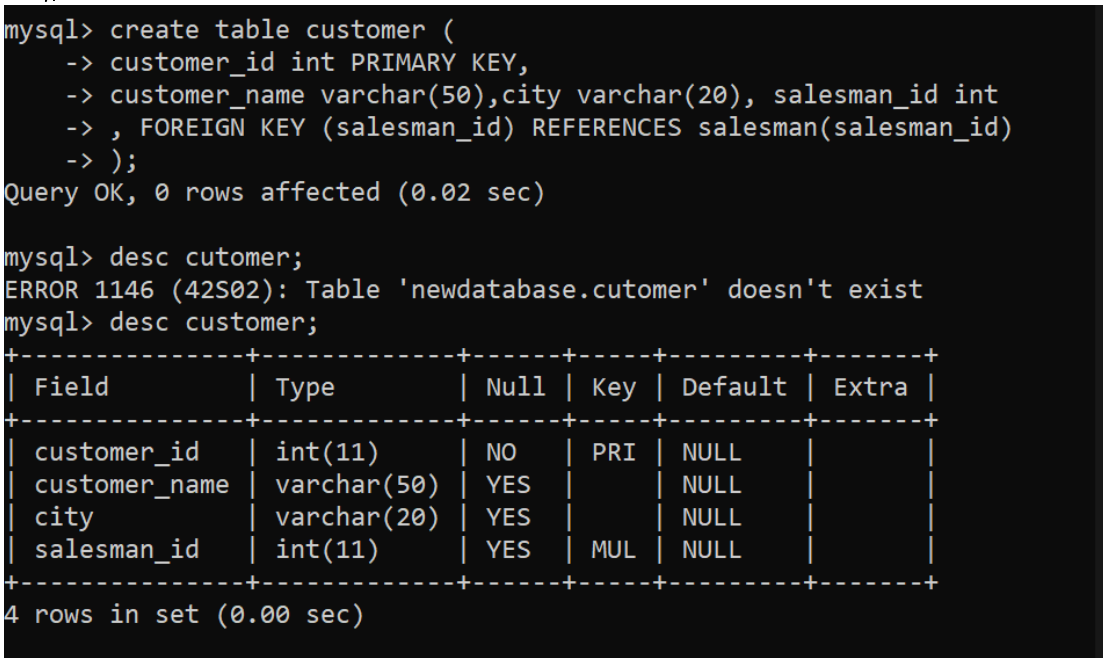
PRACTICAL 1

```
create table salesman (salesman_id int PRIMARY KEY,
   name varchar(20),city varchar(30), commission decimal(5,2)
   );
```



```
mysql> create database newDatabase;
Query OK, 1 row affected (0.00 sec)

mysql> use newDatabase;
Database changed
mysql> create table salesman (salesman_id int PRIMARY KEY,
    -> name varchar(20),city varchar(30), commission decimal(5,2)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> desc salesman;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| salesman_id | int(11)      | NO   | PRI | NULL    |       |
| name        | varchar(20)  | YES  |     | NULL    |       |
| city        | varchar(30)  | YES  |     | NULL    |       |
| commission  | decimal(5,2) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

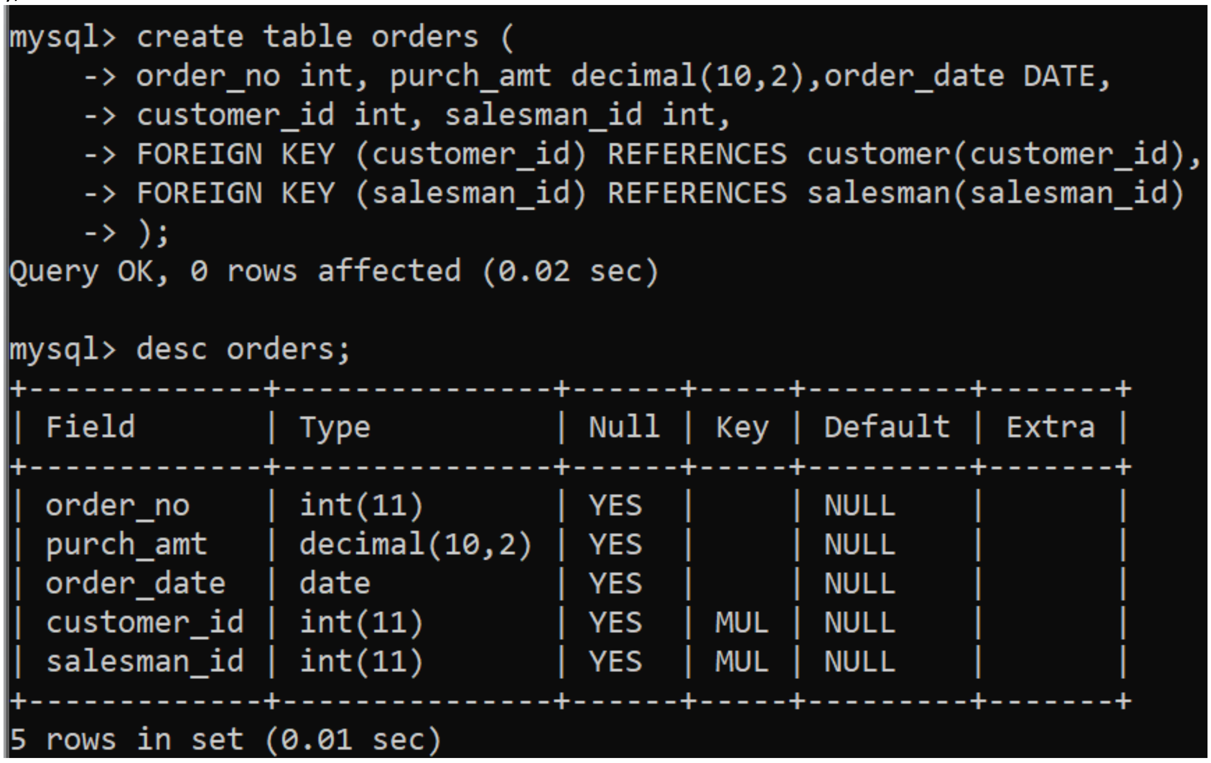
```
create table customer (
  customer_id int PRIMARY KEY,
  customer_name varchar(50),city varchar(20), salesman_id int
  , FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
  );
```



```
mysql> create table customer (
    -> customer_id int PRIMARY KEY,
    -> customer_name varchar(50),city varchar(20), salesman_id int
    -> , FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> desc cutomer;
ERROR 1146 (42S02): Table 'newdatabase.cutomer' doesn't exist
mysql> desc customer;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| customer_id   | int(11)     | NO   | PRI | NULL    |       |
| customer_name | varchar(50) | YES  |     | NULL    |       |
| city          | varchar(20) | YES  |     | NULL    |       |
| salesman_id   | int(11)     | YES  | MUL | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

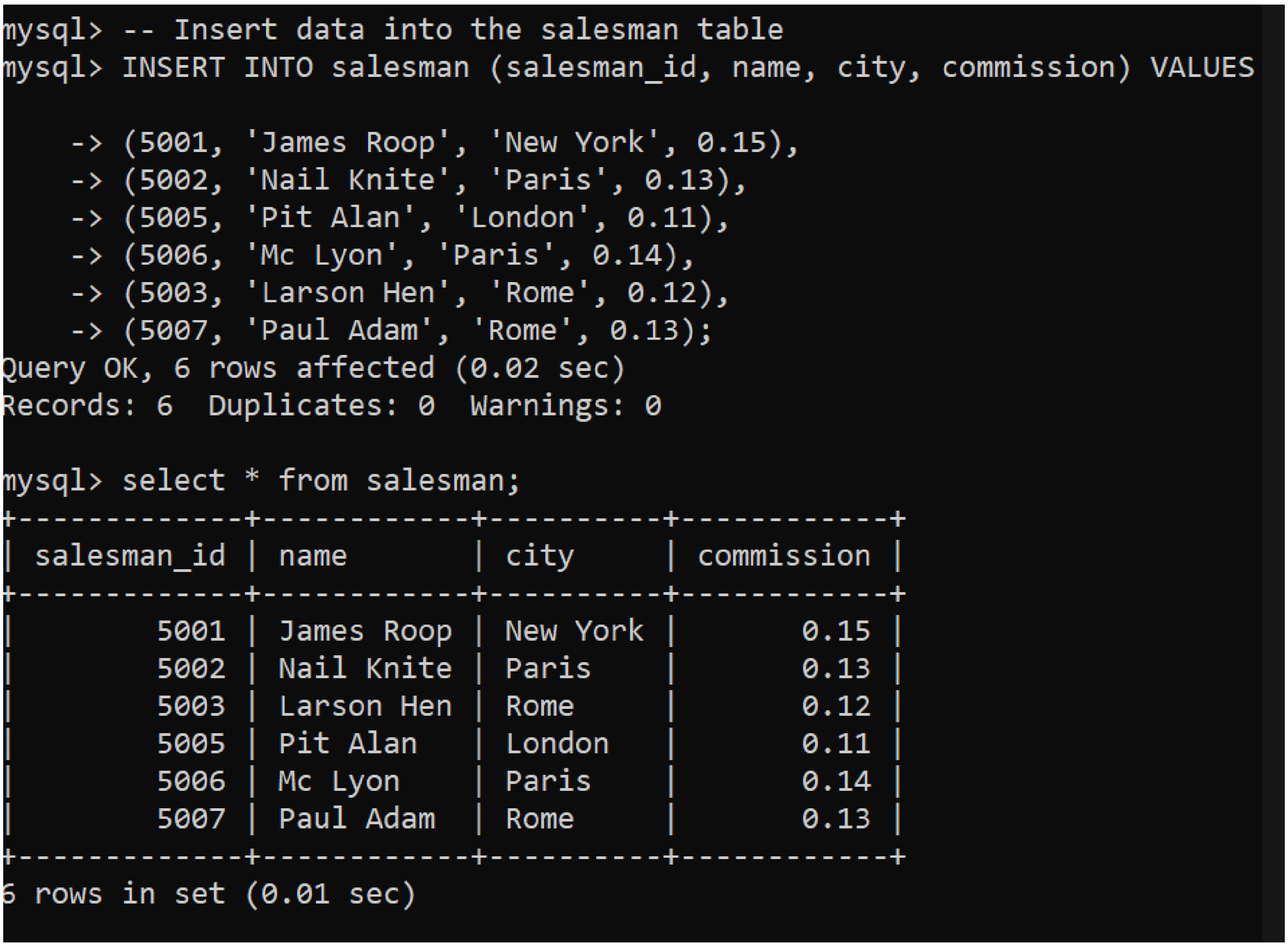
```
create table orders (
  order_no int, purch_amt decimal(10,2),order_date DATE,
  customer_id int, salesman_id int,
  FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
  FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
```

);



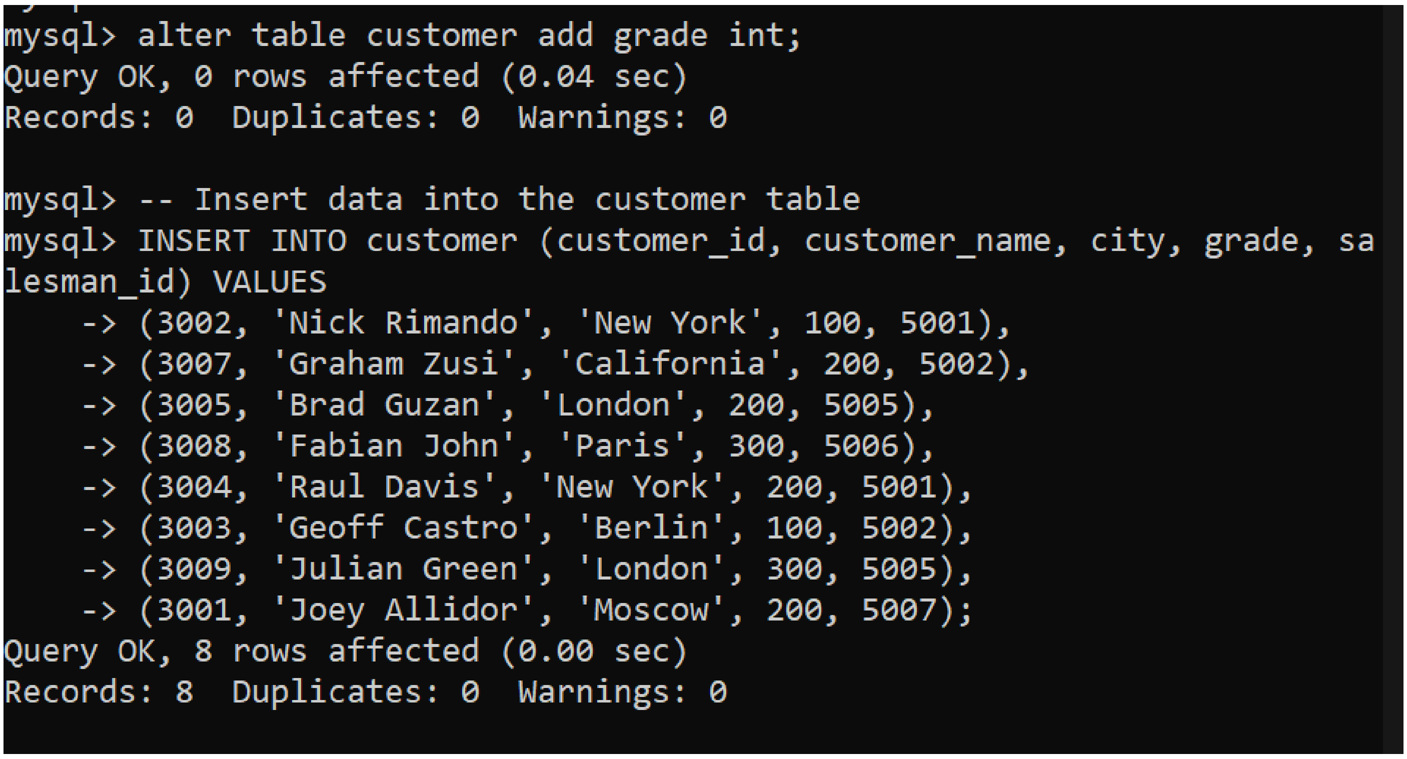
```
mysql> create table orders (
    -> order_no int, purch_amt decimal(10,2),order_date DATE,
    -> customer_id int, salesman_id int,
    -> FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
    -> FOREIGN KEY (salesman_id) REFERENCES salesman(salesman_id)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> desc orders;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| order_no    | int(11)       | YES  |     | NULL    |       |
| purch_amt   | decimal(10,2) | YES  |     | NULL    |       |
| order_date  | date          | YES  |     | NULL    |       |
| customer_id | int(11)       | YES  | MUL | NULL    |       |
| salesman_id | int(11)       | YES  | MUL | NULL    |       |
+-------------+---------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

```
-- Insert data into the salesman table
INSERT INTO salesman (salesman_id, name, city, commission) VALUES
(5001, 'James Roop', 'New York', 0.15),
(5002, 'Nail Knite', 'Paris', 0.13),
(5005, 'Pit Alan', 'London', 0.11),
(5006, 'Mc Lyon', 'Paris', 0.14),
(5003, 'Larson Hen', 'Rome', 0.12),
(5007, 'Paul Adam', 'Rome', 0.13);
```

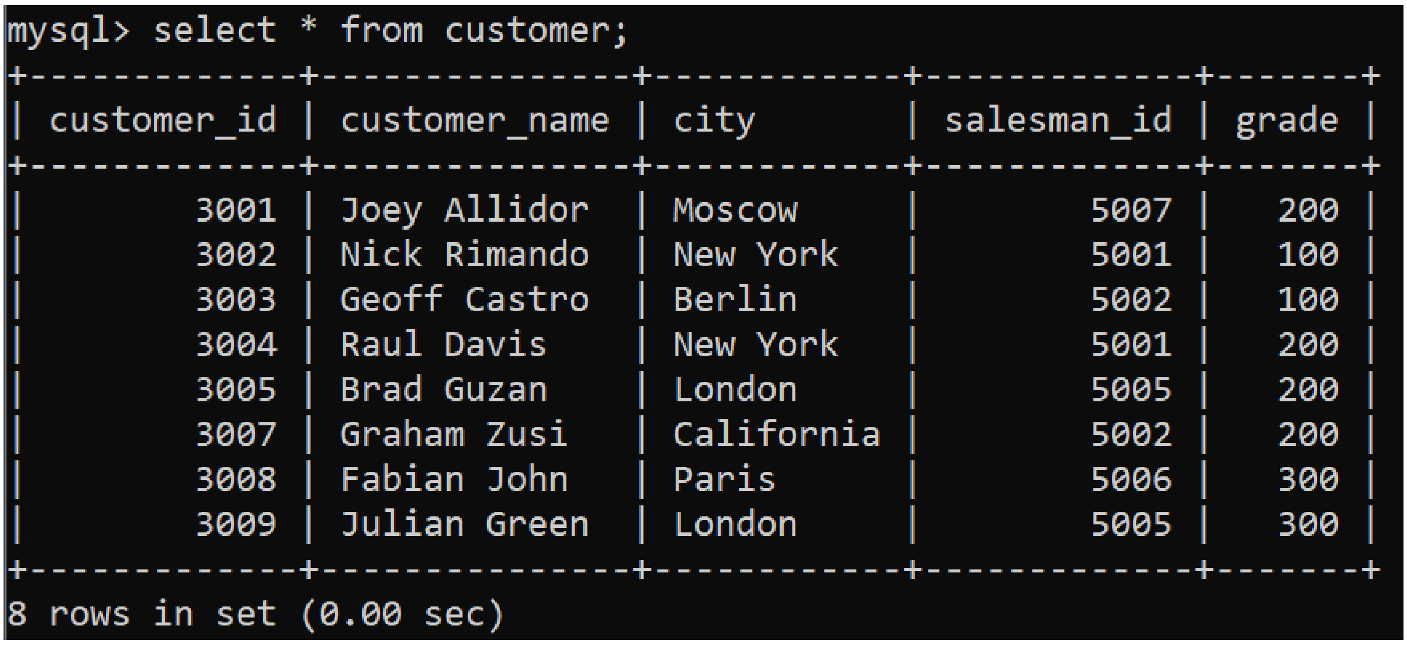


```
mysql> -- Insert data into the salesman table
mysql> INSERT INTO salesman (salesman_id, name, city, commission) VALUES

    -> (5001, 'James Roop', 'New York', 0.15),
    -> (5002, 'Nail Knite', 'Paris', 0.13),
    -> (5005, 'Pit Alan', 'London', 0.11),
    -> (5006, 'Mc Lyon', 'Paris', 0.14),
    -> (5003, 'Larson Hen', 'Rome', 0.12),
    -> (5007, 'Paul Adam', 'Rome', 0.13);
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from salesman;
+-------------+------------+----------+------------+
| salesman_id | name       | city     | commission |
+-------------+------------+----------+------------+
|        5001 | James Roop | New York |       0.15 |
|        5002 | Nail Knite | Paris    |       0.13 |
|        5003 | Larson Hen | Rome     |       0.12 |
|        5005 | Pit Alan   | London   |       0.11 |
|        5006 | Mc Lyon    | Paris    |       0.14 |
|        5007 | Paul Adam  | Rome     |       0.13 |
+-------------+------------+----------+------------+
6 rows in set (0.01 sec)
```

```
-- Insert data into the customer table
INSERT INTO customer (customer_id, customer_name, city, grade, salesman_id) VALUES
(3002, 'Nick Rimando', 'New York', 100, 5001),
(3007, 'Graham Zusi', 'California', 200, 5002),
(3005, 'Brad Guzan', 'London', 200, 5005),
(3008, 'Fabian John', 'Paris', 300, 5006),
(3004, 'Raul Davis', 'New York', 200, 5001),
(3003, 'Geoff Castro', 'Berlin', 100, 5002),
(3009, 'Julian Green', 'London', 300, 5005),
(3001, 'Joey Allidor', 'Moscow', 200, 5007);
```

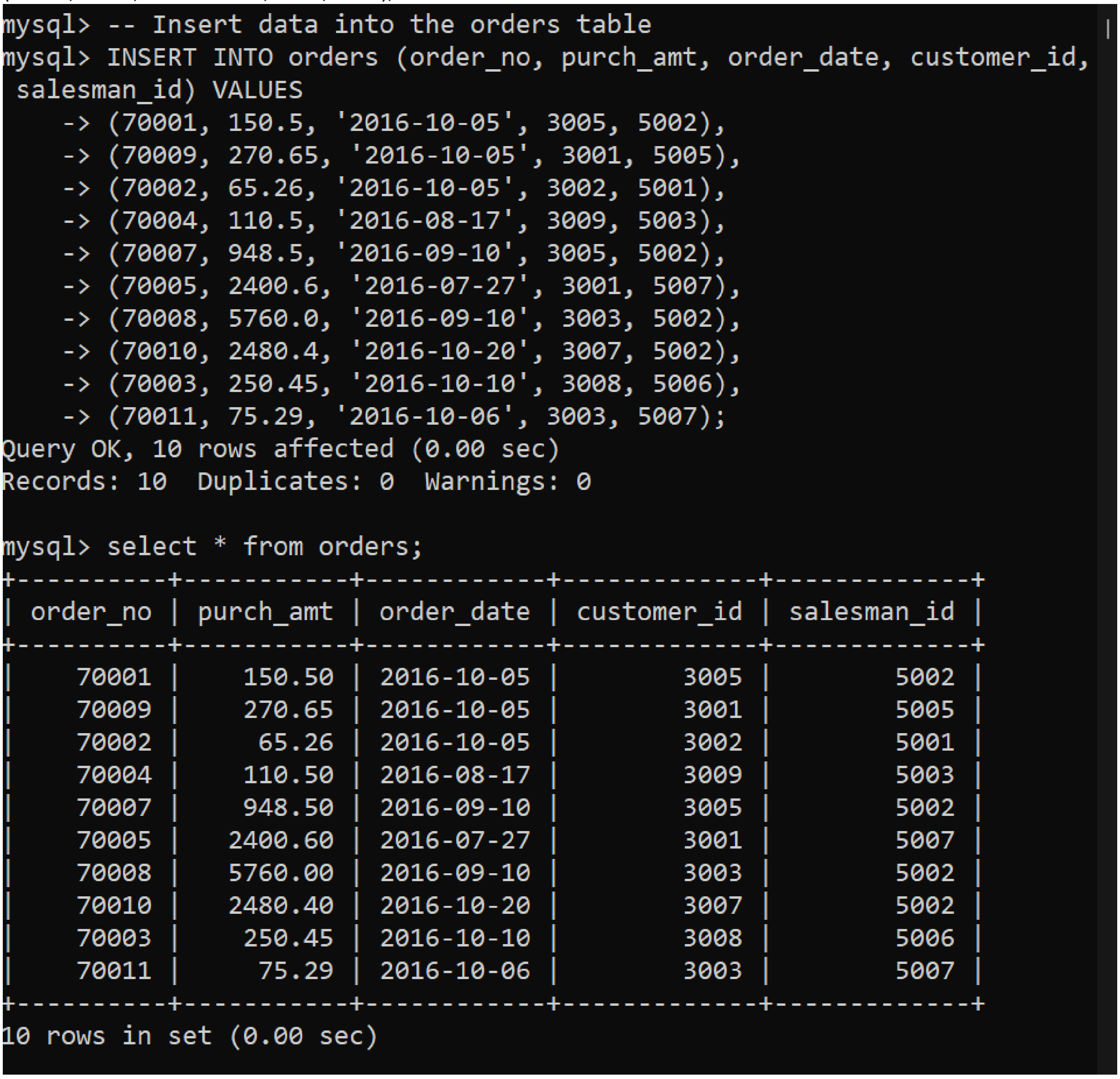


```
mysql> alter table customer add grade int;
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> -- Insert data into the customer table
mysql> INSERT INTO customer (customer_id, customer_name, city, grade, sa
lesman_id) VALUES
    -> (3002, 'Nick Rimando', 'New York', 100, 5001),
    -> (3007, 'Graham Zusi', 'California', 200, 5002),
    -> (3005, 'Brad Guzan', 'London', 200, 5005),
    -> (3008, 'Fabian John', 'Paris', 300, 5006),
    -> (3004, 'Raul Davis', 'New York', 200, 5001),
    -> (3003, 'Geoff Castro', 'Berlin', 100, 5002),
    -> (3009, 'Julian Green', 'London', 300, 5005),
    -> (3001, 'Joey Allidor', 'Moscow', 200, 5007);
Query OK, 8 rows affected (0.00 sec)
Records: 8  Duplicates: 0  Warnings: 0
```



```
mysql> select * from customer;
+-------------+---------------+------------+-------------+-------+
| customer_id | customer_name | city       | salesman_id | grade |
+-------------+---------------+------------+-------------+-------+
|        3001 | Joey Allidor  | Moscow     |        5007 |   200 |
|        3002 | Nick Rimando  | New York   |        5001 |   100 |
|        3003 | Geoff Castro  | Berlin     |        5002 |   100 |
|        3004 | Raul Davis    | New York   |        5001 |   200 |
|        3005 | Brad Guzan    | London     |        5005 |   200 |
|        3007 | Graham Zusi   | California |        5002 |   200 |
|        3008 | Fabian John   | Paris      |        5006 |   300 |
|        3009 | Julian Green  | London     |        5005 |   300 |
+-------------+---------------+------------+-------------+-------+
8 rows in set (0.00 sec)
```

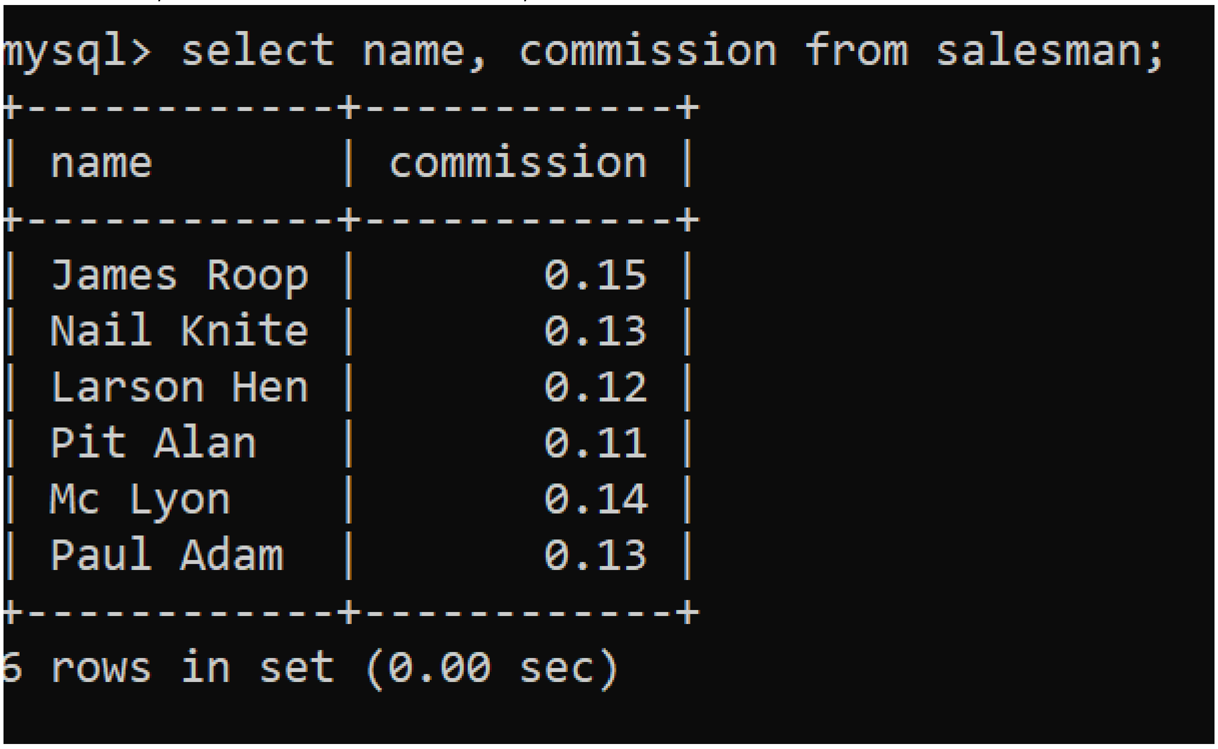
```
-- Insert data into the orders table
INSERT INTO orders (order_no, purch_amt, order_date, customer_id, salesman_id) VALUES
(70001, 150.5, '2016-10-05', 3005, 5002),
(70009, 270.65, '2016-10-05', 3001, 5005),
(70002, 65.26, '2016-10-05', 3002, 5001),
(70004, 110.5, '2016-08-17', 3009, 5003),
(70007, 948.5, '2016-09-10', 3005, 5002),
(70005, 2400.6, '2016-07-27', 3001, 5007),
(70008, 5760.0, '2016-09-10', 3003, 5002),
(70010, 2480.4, '2016-10-20', 3007, 5002),
(70003, 250.45, '2016-10-10', 3008, 5006),
(70011, 75.29, '2016-10-06', 3003, 5007);
```



```
mysql> -- Insert data into the orders table
mysql> INSERT INTO orders (order_no, purch_amt, order_date, customer_id,
 salesman_id) VALUES
    -> (70001, 150.5, '2016-10-05', 3005, 5002),
    -> (70009, 270.65, '2016-10-05', 3001, 5005),
    -> (70002, 65.26, '2016-10-05', 3002, 5001),
    -> (70004, 110.5, '2016-08-17', 3009, 5003),
    -> (70007, 948.5, '2016-09-10', 3005, 5002),
    -> (70005, 2400.6, '2016-07-27', 3001, 5007),
    -> (70008, 5760.0, '2016-09-10', 3003, 5002),
    -> (70010, 2480.4, '2016-10-20', 3007, 5002),
    -> (70003, 250.45, '2016-10-10', 3008, 5006),
    -> (70011, 75.29, '2016-10-06', 3003, 5007);
Query OK, 10 rows affected (0.00 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from orders;
+----------+-----------+------------+-------------+-------------+
| order_no | purch_amt | order_date | customer_id | salesman_id |
+----------+-----------+------------+-------------+-------------+
|    70001 |    150.50 | 2016-10-05 |        3005 |        5002 |
|    70009 |    270.65 | 2016-10-05 |        3001 |        5005 |
|    70002 |     65.26 | 2016-10-05 |        3002 |        5001 |
|    70004 |    110.50 | 2016-08-17 |        3009 |        5003 |
|    70007 |    948.50 | 2016-09-10 |        3005 |        5002 |
|    70005 |   2400.60 | 2016-07-27 |        3001 |        5007 |
|    70008 |   5760.00 | 2016-09-10 |        3003 |        5002 |
|    70010 |   2480.40 | 2016-10-20 |        3007 |        5002 |
|    70003 |    250.45 | 2016-10-10 |        3008 |        5006 |
|    70011 |     75.29 | 2016-10-06 |        3003 |        5007 |
+----------+-----------+------------+-------------+-------------+
10 rows in set (0.00 sec)
```

1. Display name and commission for all the salesmen.

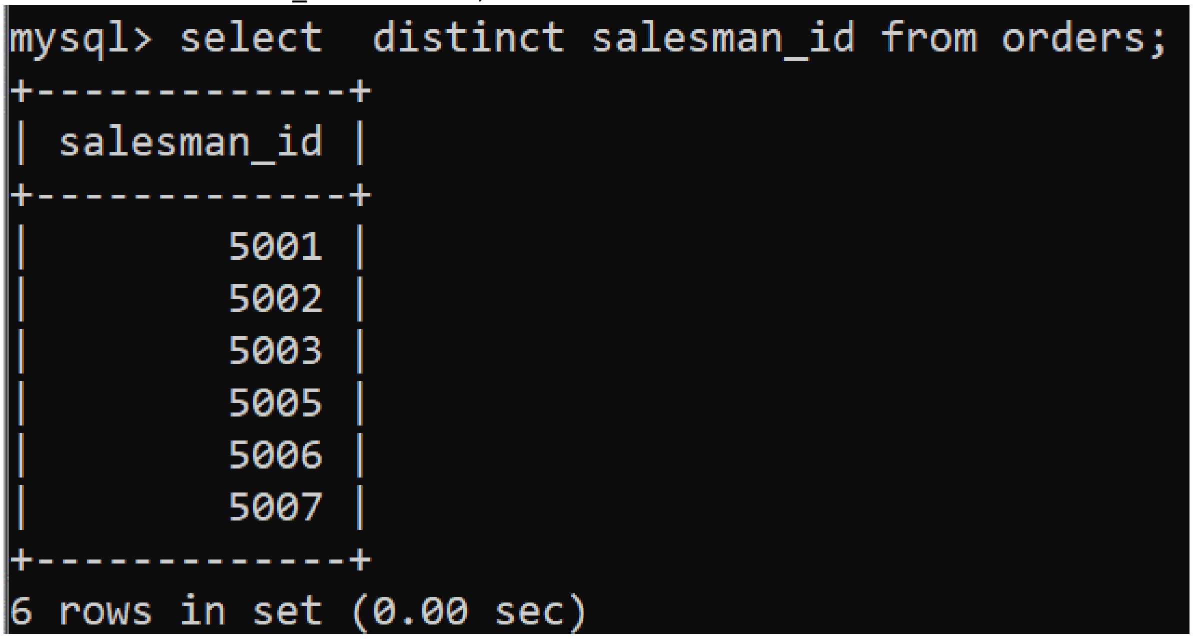
select name, commission from salesman;



```
mysql> select name, commission from salesman;
+------------+------------+
| name       | commission |
+------------+------------+
| James Roop |       0.15 |
| Nail Knite |       0.13 |
| Larson Hen |       0.12 |
| Pit Alan   |       0.11 |
| Mc Lyon    |       0.14 |
| Paul Adam  |       0.13 |
+------------+------------+
6 rows in set (0.00 sec)
```

2. Retrieve salesman id of all salesmen from orders table without any repeats.

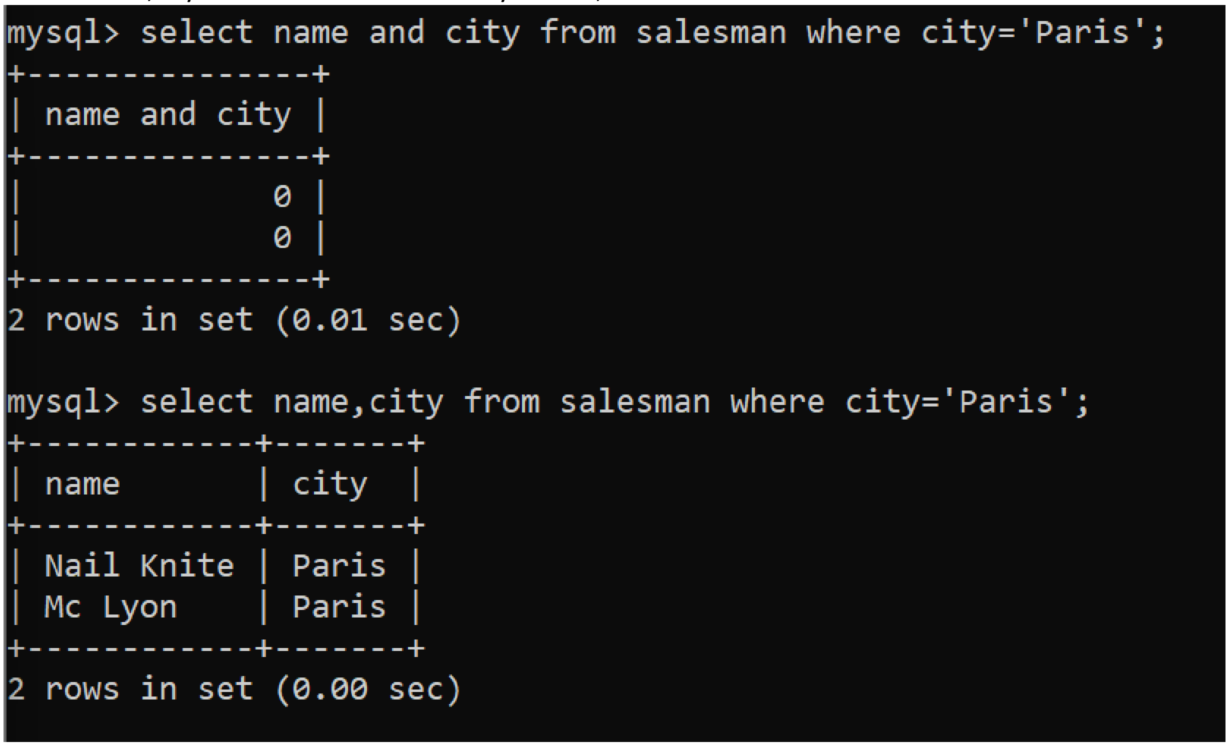
select distinct salesman_id from orders;



```
mysql> select  distinct salesman_id from orders;
+-------------+
| salesman_id |
+-------------+
|        5001 |
|        5002 |
|        5003 |
|        5005 |
|        5006 |
|        5007 |
+-------------+
6 rows in set (0.00 sec)
```

3. Display names and city of salesman, who belongs to the city of Paris.

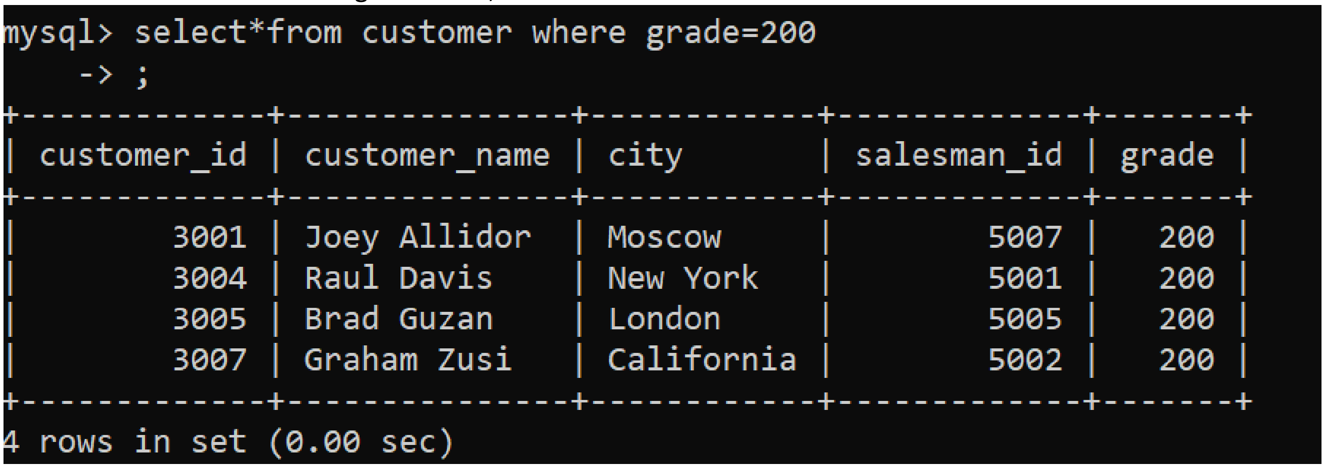
select name,city from salesman where city='Paris';



```
mysql> select name and city from salesman where city='Paris';
+---------------+
| name and city |
+---------------+
|             0 |
|             0 |
+---------------+
2 rows in set (0.01 sec)

mysql> select name,city from salesman where city='Paris';
+------------+-------+
| name       | city  |
+------------+-------+
| Nail Knite | Paris |
| Mc Lyon    | Paris |
+------------+-------+
2 rows in set (0.00 sec)
```

4. Display all the information for those customers with a grade of 200.

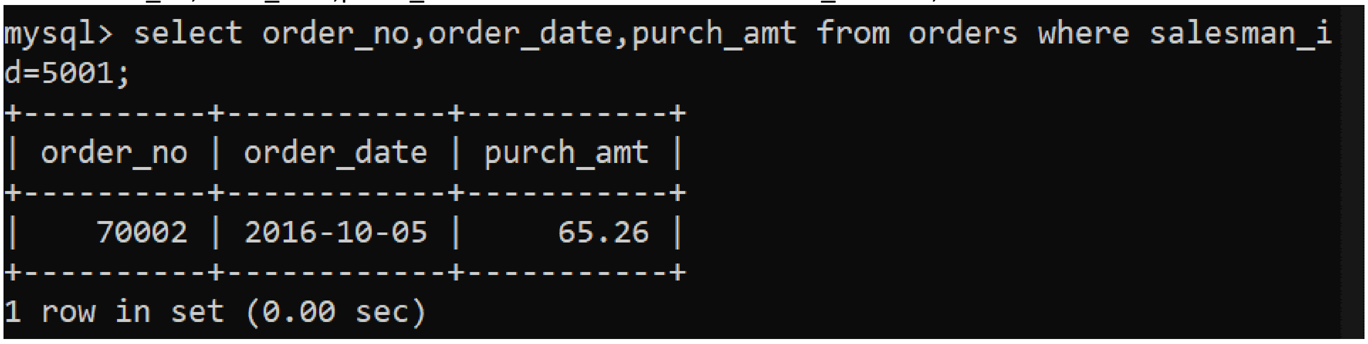
select*from customer where grade=200 ;



```
mysql> select*from customer where grade=200
    -> ;
+-------------+---------------+------------+-------------+-------+
| customer_id | customer_name | city       | salesman_id | grade |
+-------------+---------------+------------+-------------+-------+
|        3001 | Joey Allidor  | Moscow     |        5007 |   200 |
|        3004 | Raul Davis    | New York   |        5001 |   200 |
|        3005 | Brad Guzan    | London     |        5005 |   200 |
|        3007 | Graham Zusi   | California |        5002 |   200 |
+-------------+---------------+------------+-------------+-------+
4 rows in set (0.00 sec)
```

5. Display the order number, order date and the purchase amount for order(s) which will be delivered by the salesman with ID 5001

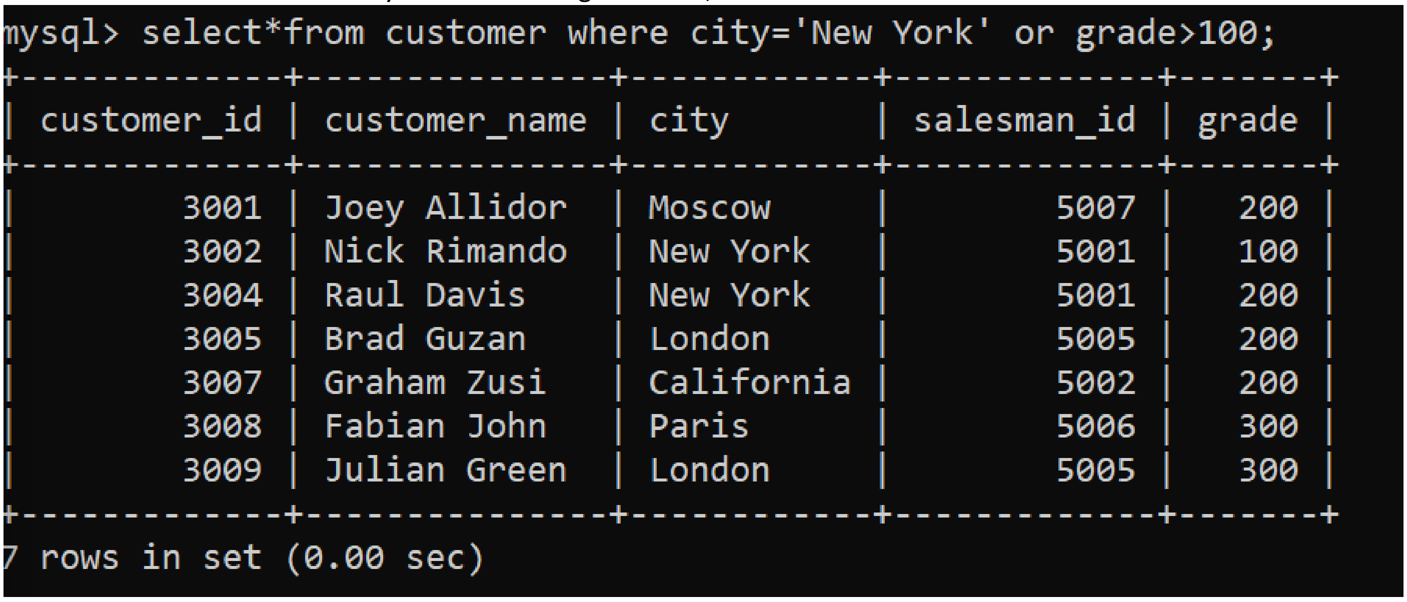
select order_no,order_date,purch_amt from orders where salesman_id=5001;



```
mysql> select order_no,order_date,purch_amt from orders where salesman_i
d=5001;
+----------+------------+-----------+
| order_no | order_date | purch_amt |
+----------+------------+-----------+
|    70002 | 2016-10-05 |     65.26 |
+----------+------------+-----------+
1 row in set (0.00 sec)
```

6. Display all the customers, who are either belongs to the city New York or not had a grade above 100.

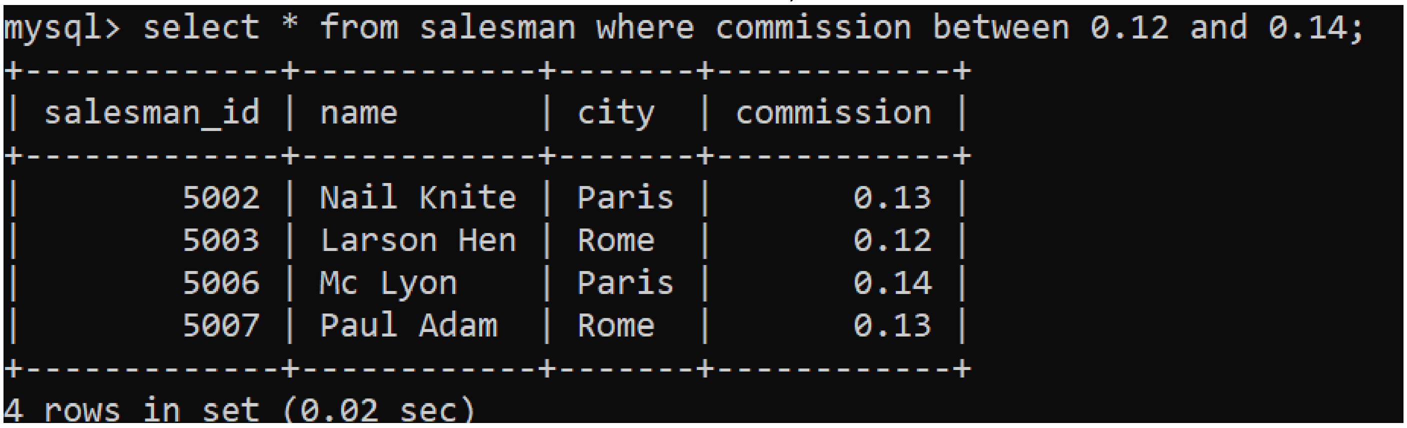
select*from customer where city='New York' or grade>100;



```
mysql> select*from customer where city='New York' or grade>100;
+-------------+---------------+------------+-------------+-------+
| customer_id | customer_name | city       | salesman_id | grade |
+-------------+---------------+------------+-------------+-------+
|        3001 | Joey Allidor  | Moscow     |        5007 |   200 |
|        3002 | Nick Rimando  | New York   |        5001 |   100 |
|        3004 | Raul Davis    | New York   |        5001 |   200 |
|        3005 | Brad Guzan    | London     |        5005 |   200 |
|        3007 | Graham Zusi   | California |        5002 |   200 |
|        3008 | Fabian John   | Paris      |        5006 |   300 |
|        3009 | Julian Green  | London     |        5005 |   300 |
+-------------+---------------+------------+-------------+-------+
7 rows in set (0.00 sec)
```

7. Find those salesmen with all information who gets the commission within a range of 0.12 and 0.14.

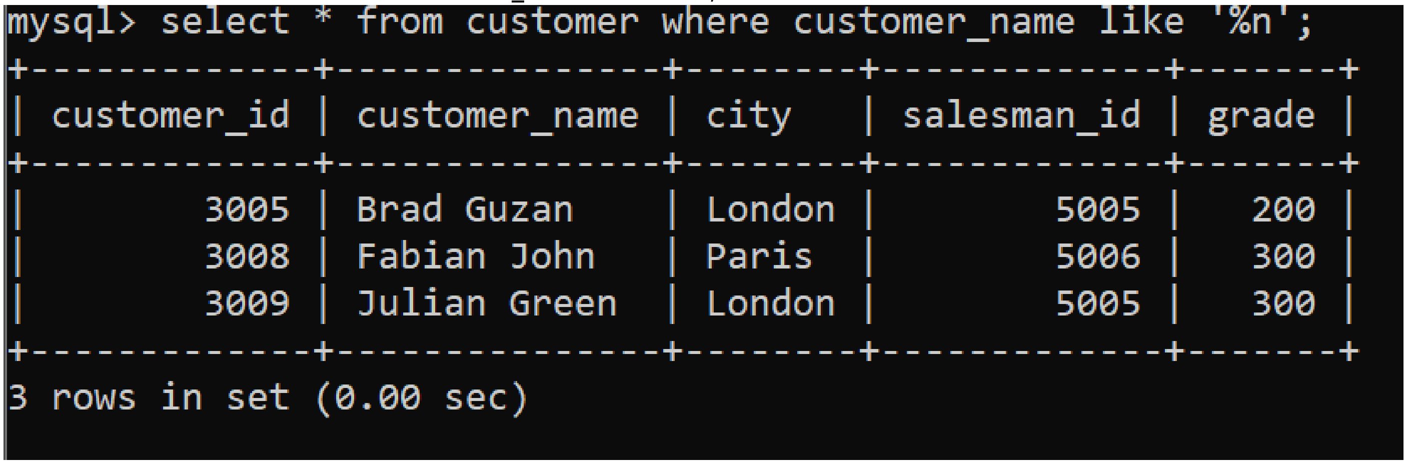
select * from salesman where commission between 0.12 and 0.14;



```
mysql> select * from salesman where commission between 0.12 and 0.14;
+-------------+------------+-------+------------+
| salesman_id | name       | city  | commission |
+-------------+------------+-------+------------+
|        5002 | Nail Knite | Paris |       0.13 |
|        5003 | Larson Hen | Rome  |       0.12 |
|        5006 | Mc Lyon    | Paris |       0.14 |
|        5007 | Paul Adam  | Rome  |       0.13 |
+-------------+------------+-------+------------+
4 rows in set (0.02 sec)
```

8. Find all those customers with all information whose names are ending with the letter 'n'.

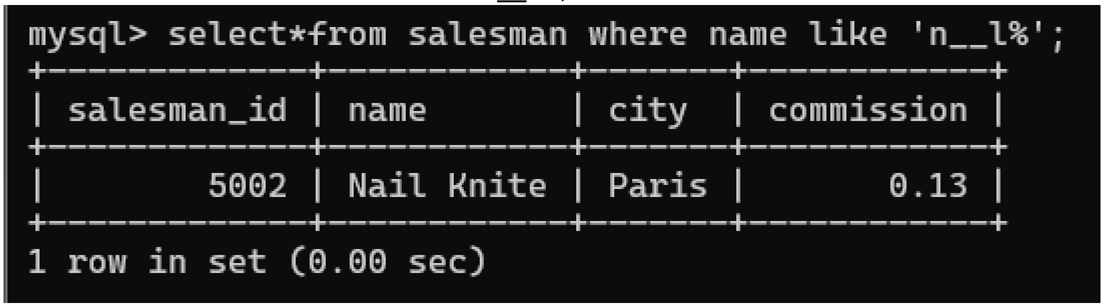
select * from customer where customer_name like '%n';



```
mysql> select * from customer where customer_name like '%n';
+-------------+---------------+--------+-------------+-------+
| customer_id | customer_name | city   | salesman_id | grade |
+-------------+---------------+--------+-------------+-------+
|        3005 | Brad Guzan    | London |        5005 |   200 |
|        3008 | Fabian John   | Paris  |        5006 |   300 |
|        3009 | Julian Green  | London |        5005 |   300 |
+-------------+---------------+--------+-------------+-------+
3 rows in set (0.00 sec)
```

9. Find those salesmen with all information whose name containing the 1st character is 'N' and the 4th character is 'l' and rests may be any character.

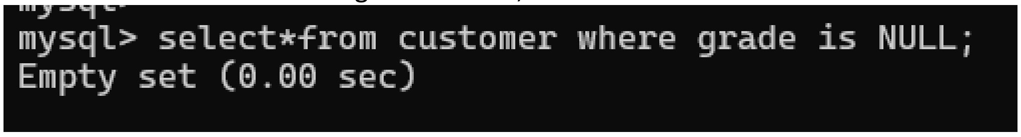
select*from salesman where name like 'n__l%';



```
mysql> select*from salesman where name like 'n__l%';
+-------------+------------+-------+------------+
| salesman_id | name       | city  | commission |
+-------------+------------+-------+------------+
|        5002 | Nail Knite | Paris |       0.13 |
+-------------+------------+-------+------------+
1 row in set (0.00 sec)
```

10. Find that customer with all information who does not get any grade except NULL.

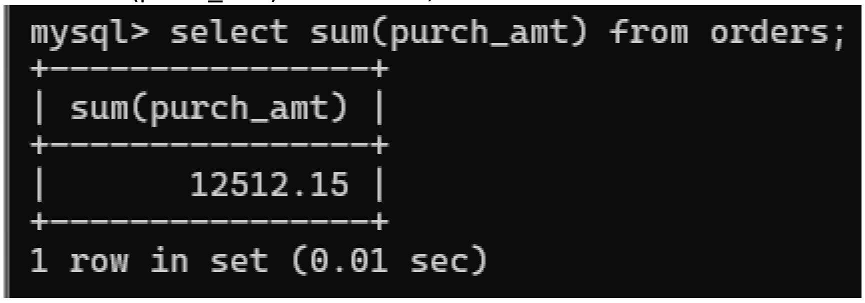
select*from customer where grade is NULL;



```
mysql> select*from customer where grade is NULL;
Empty set (0.00 sec)
```

11. Find the total purchase amount of all orders.

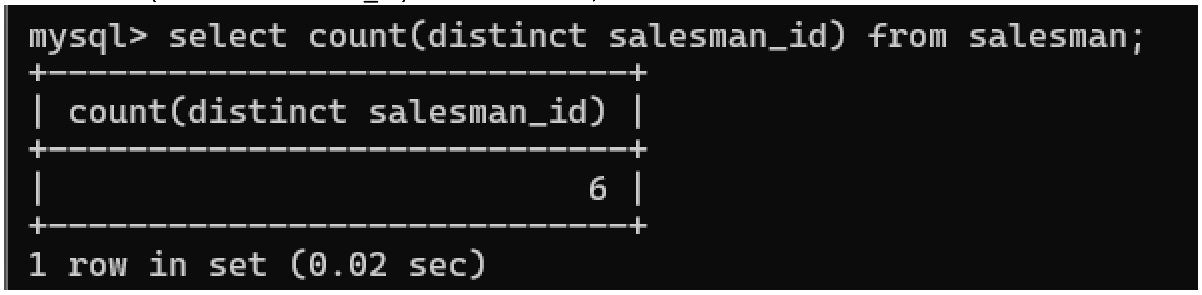
select sum(purch_amt) from orders;



```
mysql> select sum(purch_amt) from orders;
+----------------+
| sum(purch_amt) |
+----------------+
|       12512.15 |
+----------------+
1 row in set (0.01 sec)
```

12. Find the number of salesman currently listing for all of their customers.

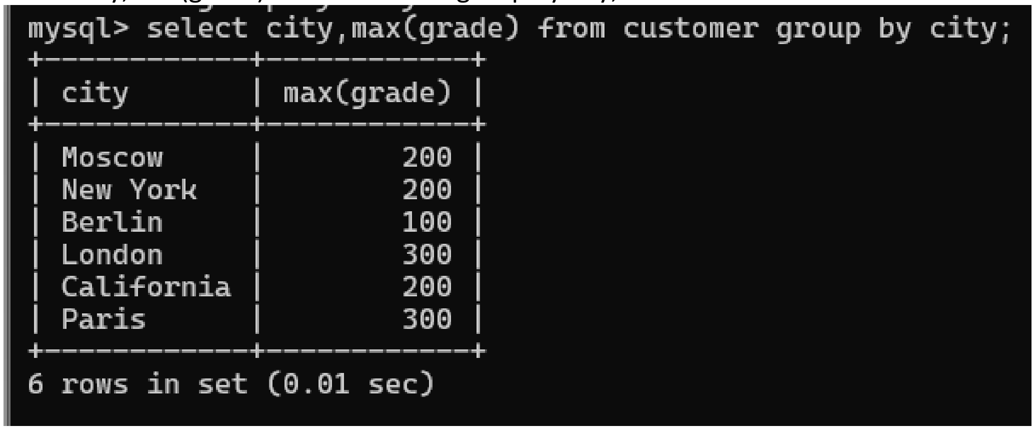
select count(distinct salesman_id) from salesman;



```
mysql> select count(distinct salesman_id) from salesman;
+-----------------------------+
| count(distinct salesman_id) |
+-----------------------------+
|                           6 |
+-----------------------------+
1 row in set (0.02 sec)
```

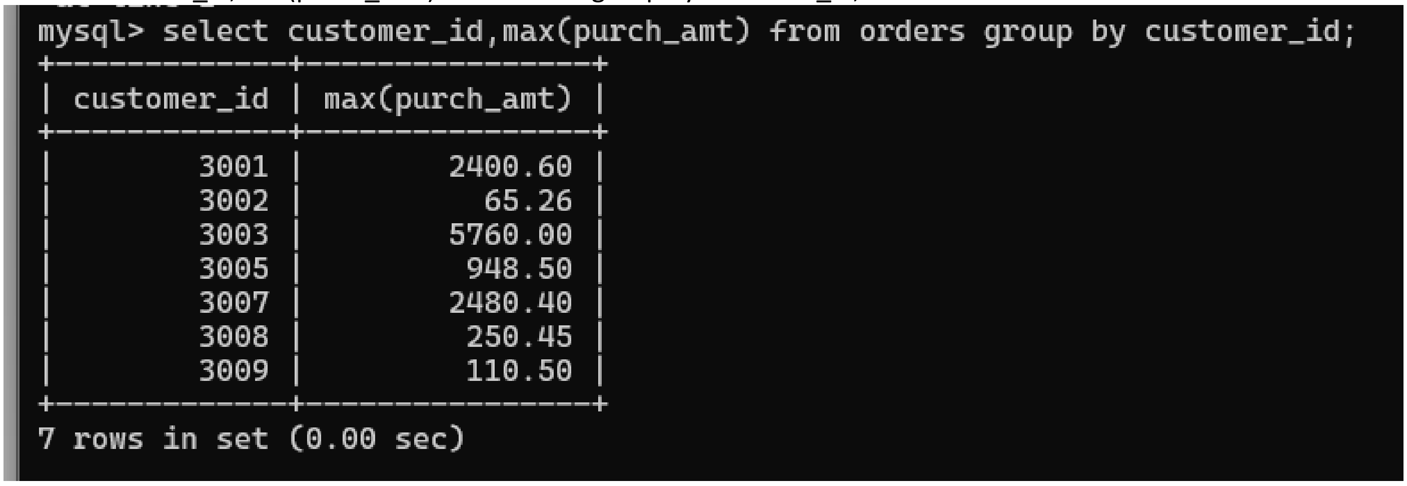
13. Find the highest grade for each of the cities of the customers.

select city,max(grade) from customer group by city;



```
mysql> select city,max(grade) from customer group by city;
+------------+------------+
| city       | max(grade) |
+------------+------------+
| Moscow     |        200 |
| New York   |        200 |
| Berlin     |        100 |
| London     |        300 |
| California |        200 |
| Paris      |        300 |
+------------+------------+
6 rows in set (0.01 sec)
```

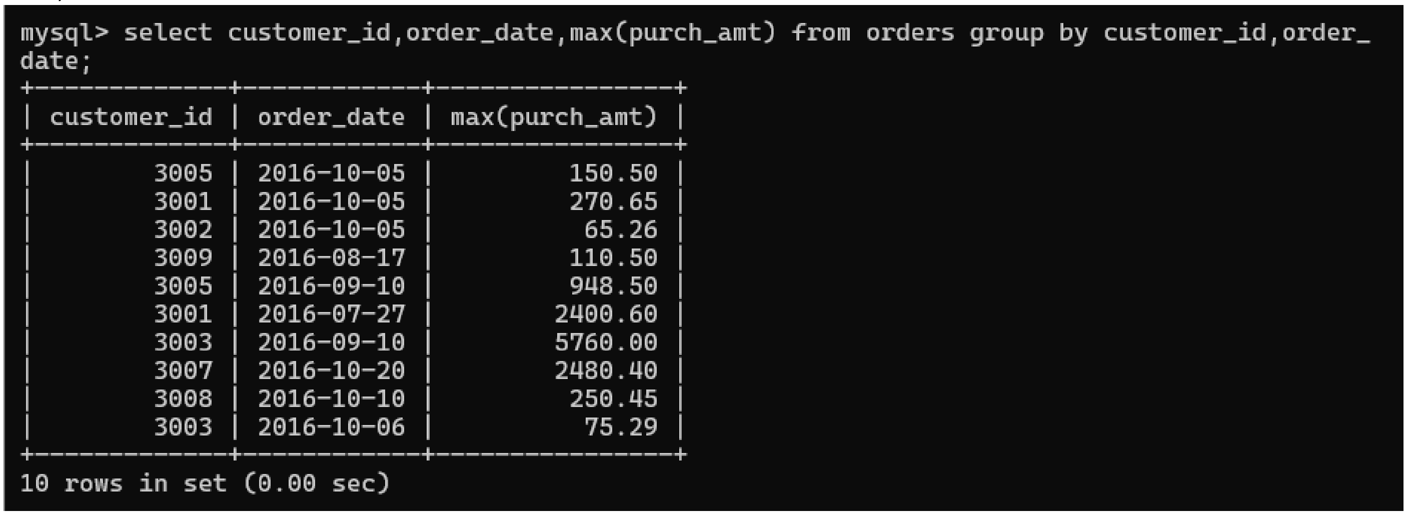
14. Find the highest purchase amount ordered by each customer with their ID and highest purchase amount.
select customer_id,max(purch_amt) from orders group by customer_id;



```
mysql> select customer_id,max(purch_amt) from orders group by customer_id;
+-------------+----------------+
| customer_id | max(purch_amt) |
+-------------+----------------+
|        3001 |        2400.60 |
|        3002 |          65.26 |
|        3003 |        5760.00 |
|        3005 |         948.50 |
|        3007 |        2480.40 |
|        3008 |         250.45 |
|        3009 |         110.50 |
+-------------+----------------+
7 rows in set (0.00 sec)
```

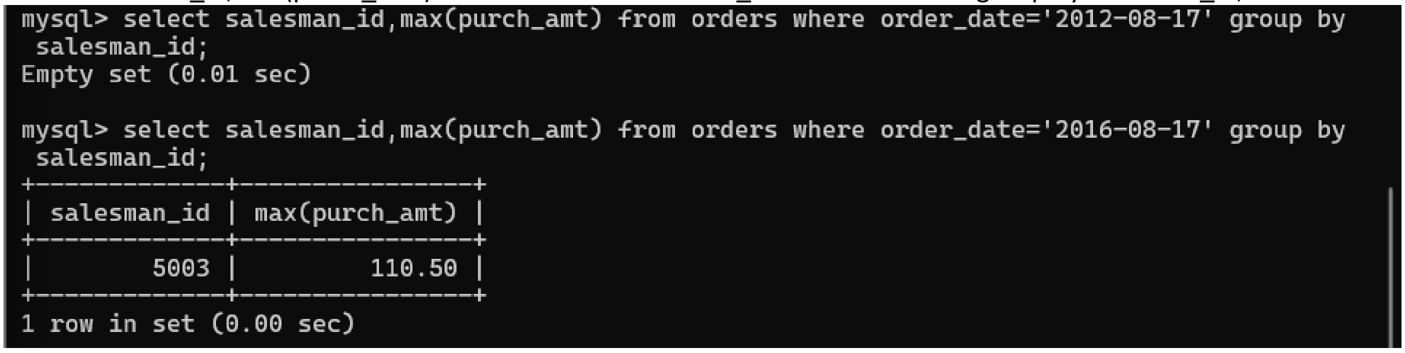
15. Find the highest purchase amount ordered by each customer on a particular date with their ID, order date and highest purchase amount.

select customer_id,order_date,max(purch_amt) from orders group by customer_id,order_
date;



```
mysql> select customer_id,order_date,max(purch_amt) from orders group by customer_id,order_
date;
+-------------+------------+----------------+
| customer_id | order_date | max(purch_amt) |
+-------------+------------+----------------+
|        3005 | 2016-10-05 |         150.50 |
|        3001 | 2016-10-05 |         270.65 |
|        3002 | 2016-10-05 |          65.26 |
|        3009 | 2016-08-17 |         110.50 |
|        3005 | 2016-09-10 |         948.50 |
|        3001 | 2016-07-27 |        2400.60 |
|        3003 | 2016-09-10 |        5760.00 |
|        3007 | 2016-10-20 |        2480.40 |
|        3008 | 2016-10-10 |         250.45 |
|        3003 | 2016-10-06 |          75.29 |
+-------------+------------+----------------+
10 rows in set (0.00 sec)
```

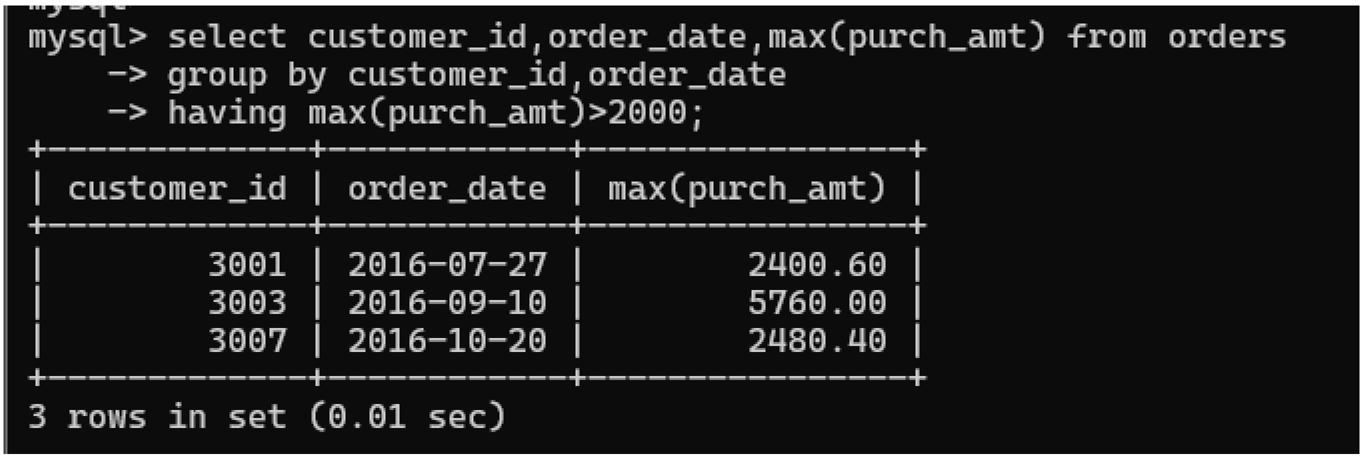
16. Find the highest purchase amount on a date '2012-08-17' for each salesman with their ID.
select salesman_id,max(purch_amt) from orders where order_date='2012-08-17' group by salesman_id;



```
mysql> select salesman_id,max(purch_amt) from orders where order_date='2012-08-17' group by
 salesman_id;
Empty set (0.01 sec)

mysql> select salesman_id,max(purch_amt) from orders where order_date='2016-08-17' group by
 salesman_id;
+-------------+----------------+
| salesman_id | max(purch_amt) |
+-------------+----------------+
|        5003 |         110.50 |
+-------------+----------------+
1 row in set (0.00 sec)
```

17. Find the highest purchase amount with their customer ID and order date, for only those customers who have the highest purchase amount in a day is more than 2000.

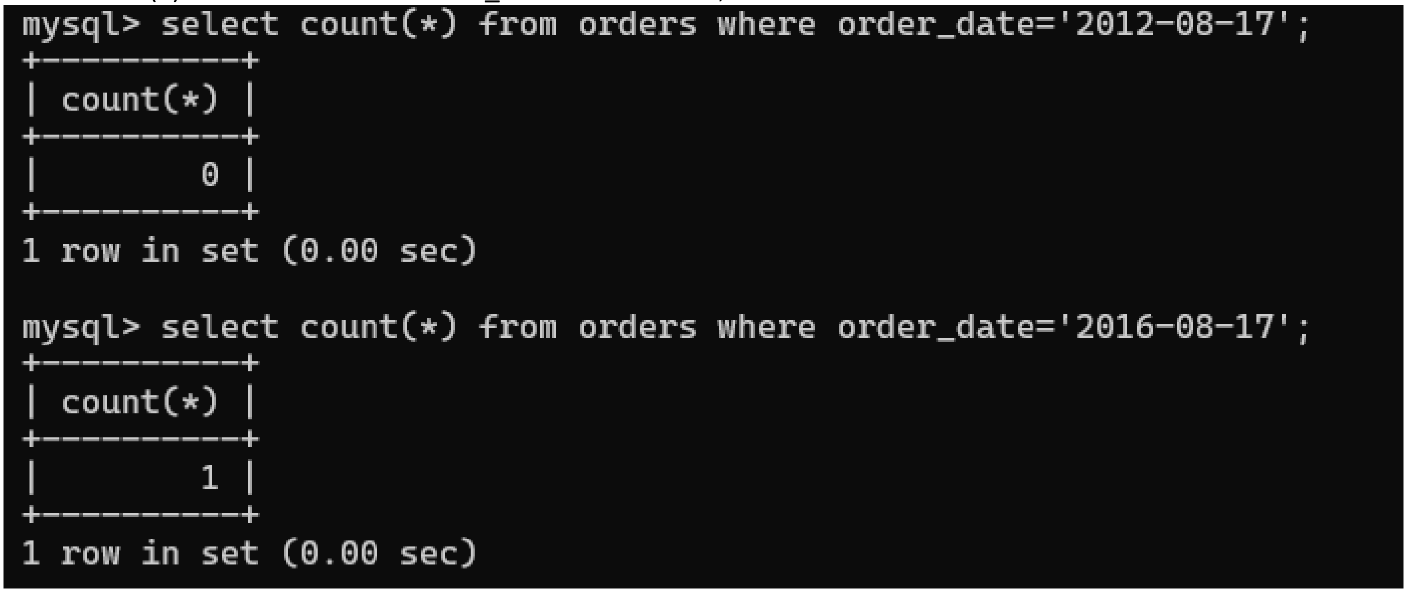
```
select customer_id,order_date,max(purch_amt) from orders
  -> group by customer_id,order_date
  -> having max(purch_amt)>2000;
```



```
mysql> select customer_id,order_date,max(purch_amt) from orders
    -> group by customer_id,order_date
    -> having max(purch_amt)>2000;
+-------------+------------+----------------+
| customer_id | order_date | max(purch_amt) |
+-------------+------------+----------------+
|        3001 | 2016-07-27 |        2400.60 |
|        3003 | 2016-09-10 |        5760.00 |
|        3007 | 2016-10-20 |        2480.40 |
+-------------+------------+----------------+
3 rows in set (0.01 sec)
```

18. Write a SQL statement that counts all orders for a date August 17th, 2012.

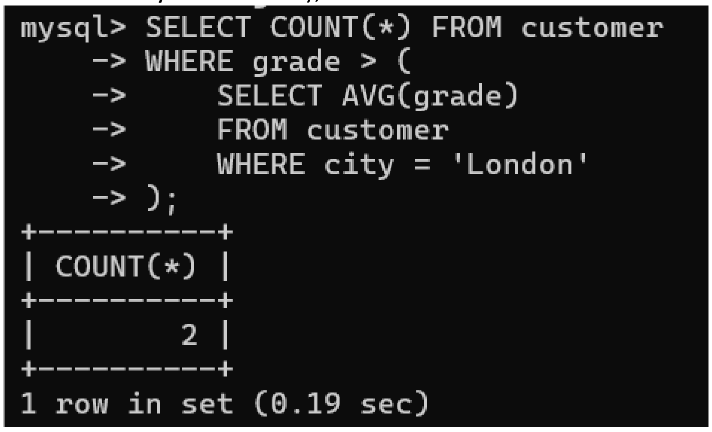
select count(*) from orders where order_date='2012-08-17';



```
mysql> select count(*) from orders where order_date='2012-08-17';
+----------+
| count(*) |
+----------+
|        0 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from orders where order_date='2016-08-17';
+----------+
| count(*) |
+----------+
|        1 |
+----------+
1 row in set (0.00 sec)
```

19. Count the customers with grades above Lonodon’s average.

```
SELECT COUNT(*) FROM customer
WHERE grade > (
   SELECT AVG(grade)
   FROM customer
   WHERE city = 'London' );
```

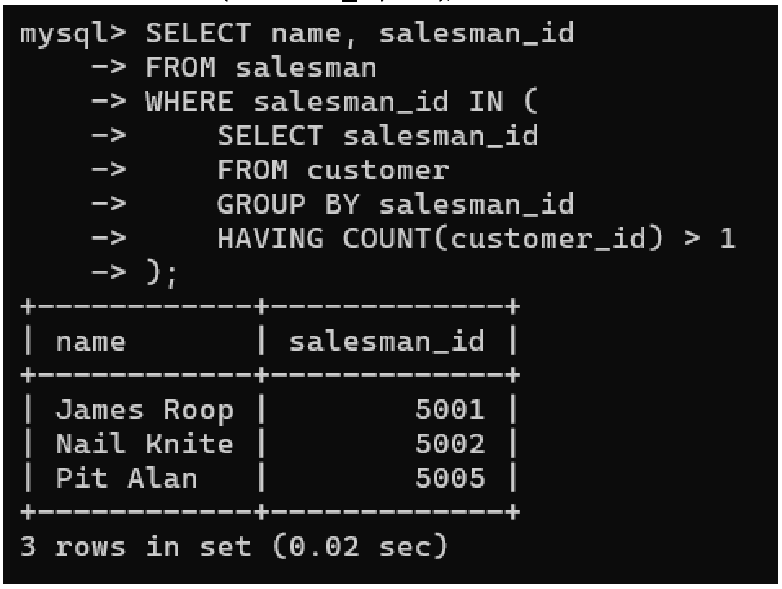


```
mysql> SELECT COUNT(*) FROM customer
    -> WHERE grade > (
    ->     SELECT AVG(grade)
    ->     FROM customer
    ->     WHERE city = 'London'
    -> );
+----------+
| COUNT(*) |
+----------+
|        2 |
+----------+
1 row in set (0.19 sec)
```

20. Find the name and numbers of all salesmen who had more than one customer.

(USING SUBQUERY)

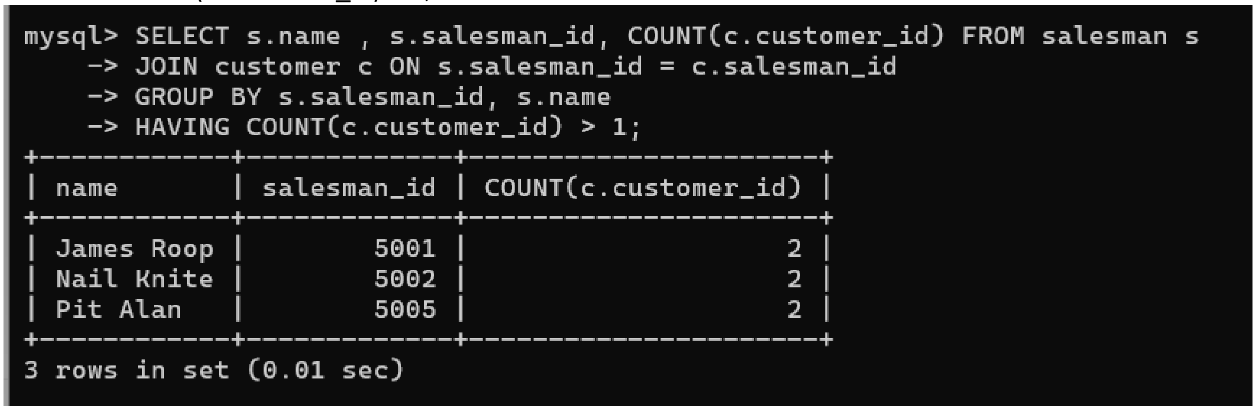
```
SELECT name, salesman_id
FROM salesman
WHERE salesman_id IN (
   SELECT salesman_id
   FROM customer
   GROUP BY salesman_id
   HAVING COUNT(customer_id) > 1 );
```



```
mysql> SELECT name, salesman_id
    -> FROM salesman
    -> WHERE salesman_id IN (
    ->     SELECT salesman_id
    ->     FROM customer
    ->     GROUP BY salesman_id
    ->     HAVING COUNT(customer_id) > 1
    -> );
+------------+-------------+
| name       | salesman_id |
+------------+-------------+
| James Roop |        5001 |
| Nail Knite |        5002 |
| Pit Alan   |        5005 |
+------------+-------------+
3 rows in set (0.02 sec)
```

(USING JOINS)

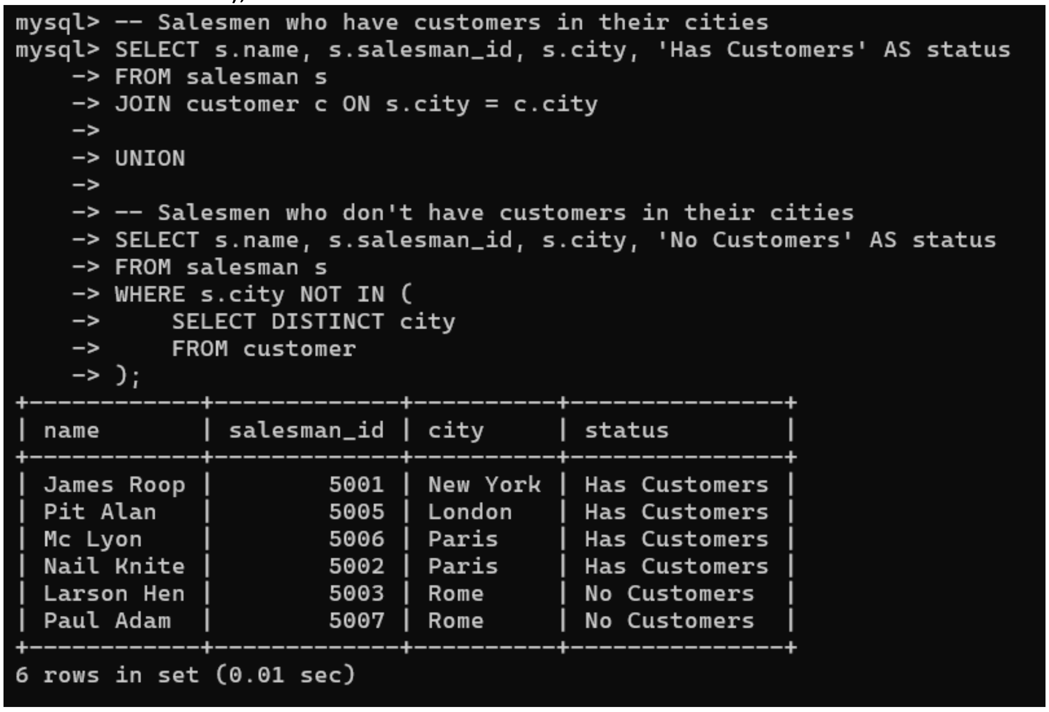
```
SELECT s.name , s.salesman_id, COUNT(c.customer_id) FROM salesman s
JOIN customer c ON s.salesman_id = c.salesman_id
GROUP BY s.salesman_id, s.name
HAVING COUNT(c.customer_id) > 1;
```



```
mysql> SELECT s.name , s.salesman_id, COUNT(c.customer_id) FROM salesman s
    -> JOIN customer c ON s.salesman_id = c.salesman_id
    -> GROUP BY s.salesman_id, s.name
    -> HAVING COUNT(c.customer_id) > 1;
+------------+-------------+----------------------+
| name       | salesman_id | COUNT(c.customer_id) |
+------------+-------------+----------------------+
| James Roop |        5001 |                    2 |
| Nail Knite |        5002 |                    2 |
| Pit Alan   |        5005 |                    2 |
+------------+-------------+----------------------+
3 rows in set (0.01 sec)
```

21. List all salesmen and indicate those who have and don’t have customers in their cities (Use UNION operation.)

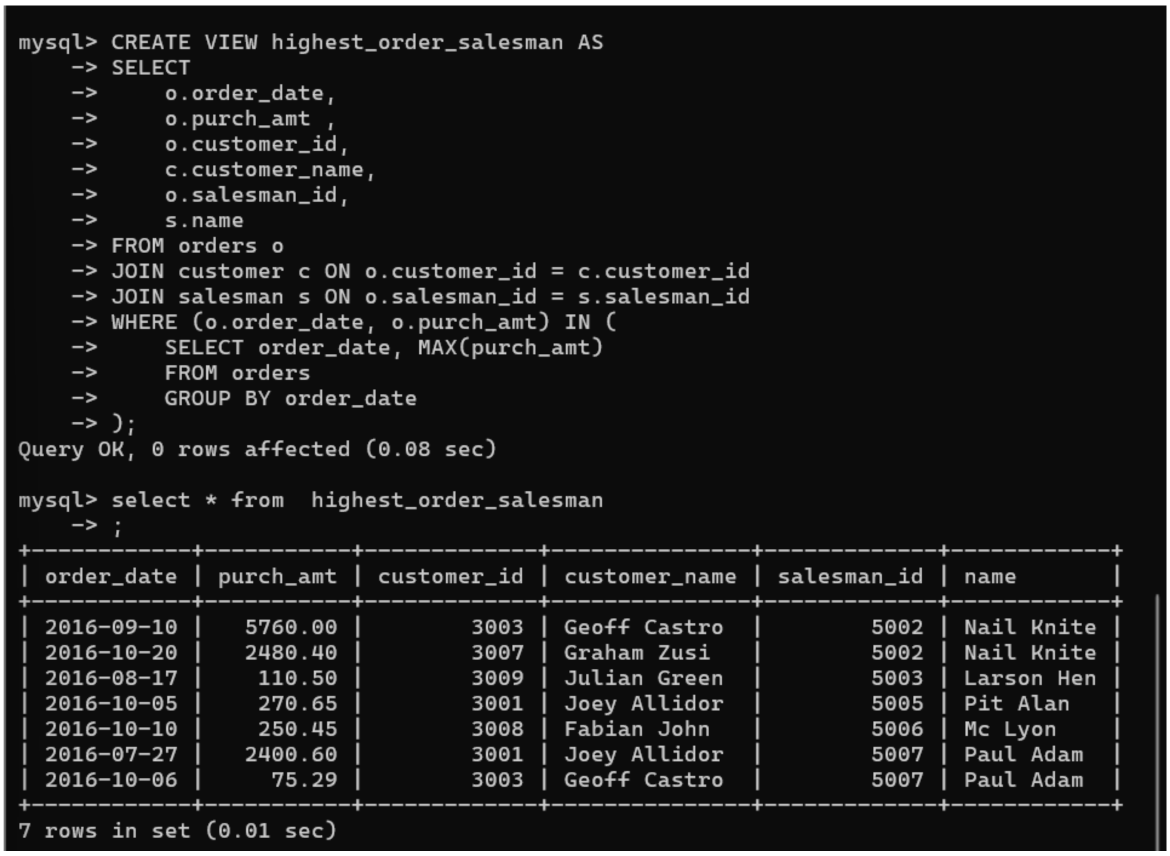
```
-- Salesmen who have customers in their cities
SELECT s.name, s.salesman_id, s.city, 'Has Customers' AS status
FROM salesman s
JOIN customer c ON s.city = c.city
UNION
-- Salesmen who don't have customers in their cities
SELECT s.name, s.salesman_id, s.city, 'No Customers' AS status
FROM salesman s
WHERE s.city NOT IN (
   SELECT DISTINCT city
   FROM customer );
```



```
mysql> -- Salesmen who have customers in their cities
mysql> SELECT s.name, s.salesman_id, s.city, 'Has Customers' AS status
    -> FROM salesman s
    -> JOIN customer c ON s.city = c.city
    ->
    -> UNION
    ->
    -> -- Salesmen who don't have customers in their cities
    -> SELECT s.name, s.salesman_id, s.city, 'No Customers' AS status
    -> FROM salesman s
    -> WHERE s.city NOT IN (
    ->     SELECT DISTINCT city
    ->     FROM customer
    -> );
+------------+-------------+----------+---------------+
| name       | salesman_id | city     | status        |
+------------+-------------+----------+---------------+
| James Roop |        5001 | New York | Has Customers |
| Pit Alan   |        5005 | London   | Has Customers |
| Mc Lyon    |        5006 | Paris    | Has Customers |
| Nail Knite |        5002 | Paris    | Has Customers |
| Larson Hen |        5003 | Rome     | No Customers  |
| Paul Adam  |        5007 | Rome     | No Customers  |
+------------+-------------+----------+---------------+
6 rows in set (0.01 sec)
```

22. Create a view that finds the salesman who has the customer with the highest order of a day.

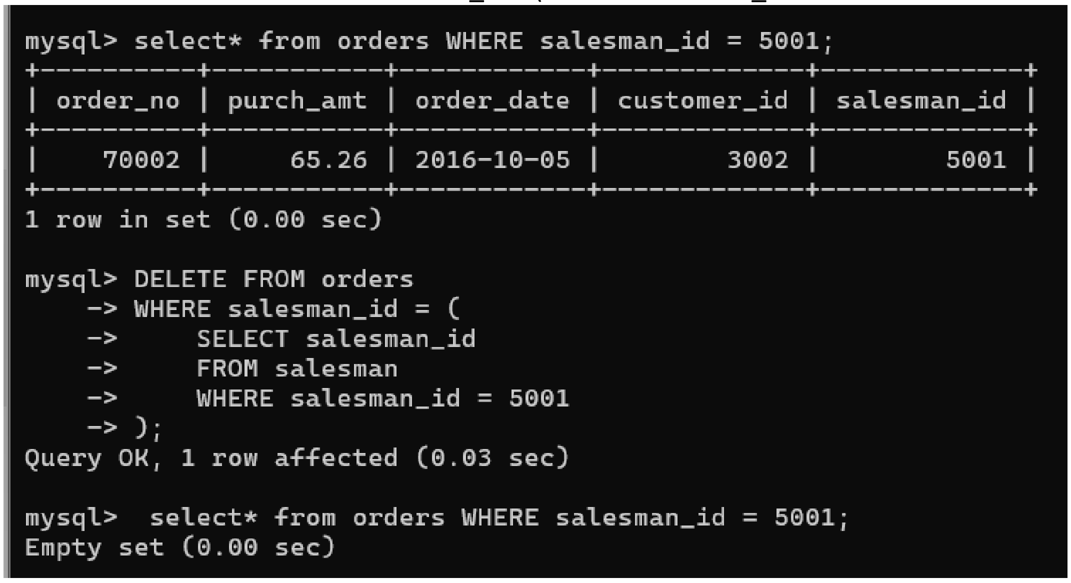
```
CREATE VIEW highest_order_salesman AS
SELECT
   o.order_date,
   o.purch_amt ,
   o.customer_id,
   c.customer_name,
   o.salesman_id,
   s.name
FROM orders o
JOIN customer c ON o.customer_id = c.customer_id
JOIN salesman s ON o.salesman_id = s.salesman_id
WHERE (o.order_date, o.purch_amt) IN (
   SELECT order_date, MAX(purch_amt)
   FROM orders
   GROUP BY order_date );

select * from  highest_order_salesman ;
```



```
mysql> CREATE VIEW highest_order_salesman AS
    -> SELECT
    ->     o.order_date,
    ->     o.purch_amt ,
    ->     o.customer_id,
    ->     c.customer_name,
    ->     o.salesman_id,
    ->     s.name
    -> FROM orders o
    -> JOIN customer c ON o.customer_id = c.customer_id
    -> JOIN salesman s ON o.salesman_id = s.salesman_id
    -> WHERE (o.order_date, o.purch_amt) IN (
    ->     SELECT order_date, MAX(purch_amt)
    ->     FROM orders
    ->     GROUP BY order_date
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> select * from  highest_order_salesman
    -> ;
+------------+-----------+-------------+---------------+-------------+------------+
| order_date | purch_amt | customer_id | customer_name | salesman_id | name       |
+------------+-----------+-------------+---------------+-------------+------------+
| 2016-09-10 |   5760.00 |        3003 | Geoff Castro  |        5002 | Nail Knite |
| 2016-10-20 |   2480.40 |        3007 | Graham Zusi   |        5002 | Nail Knite |
| 2016-08-17 |    110.50 |        3009 | Julian Green  |        5003 | Larson Hen |
| 2016-10-05 |    270.65 |        3001 | Joey Allidor  |        5005 | Pit Alan   |
| 2016-10-10 |    250.45 |        3008 | Fabian John   |        5006 | Mc Lyon    |
| 2016-07-27 |   2400.60 |        3001 | Joey Allidor  |        5007 | Paul Adam  |
| 2016-10-06 |     75.29 |        3003 | Geoff Castro  |        5007 | Paul Adam  |
+------------+-----------+-------------+---------------+-------------+------------+
7 rows in set (0.01 sec)
```

23. Demonstrate the DELETE operation by removing salesman with id 5001. All his orders must also be deleted

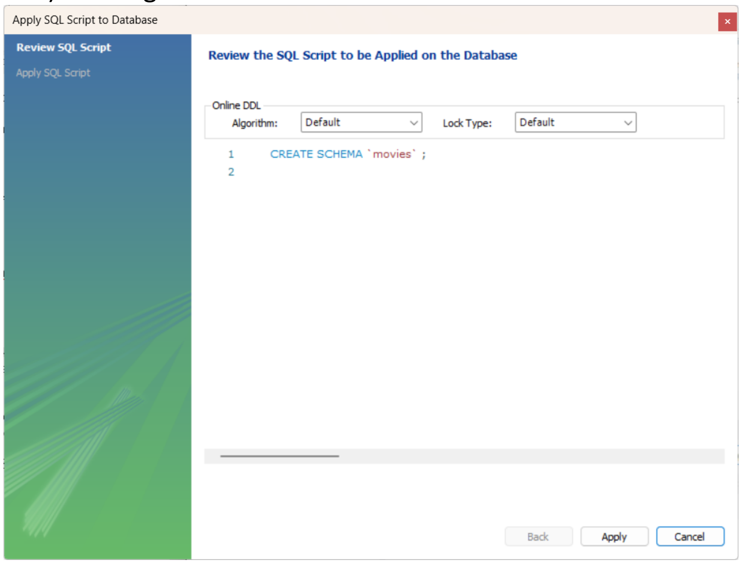
DELETE FROM orders WHERE salesman_id = (SELECT salesman_id FROM salesman WHERE salesman_id = 5001);



```
mysql> select* from orders WHERE salesman_id = 5001;
+----------+-----------+------------+-------------+-------------+
| order_no | purch_amt | order_date | customer_id | salesman_id |
+----------+-----------+------------+-------------+-------------+
|    70002 |     65.26 | 2016-10-05 |        3002 |        5001 |
+----------+-----------+------------+-------------+-------------+
1 row in set (0.00 sec)

mysql> DELETE FROM orders
    -> WHERE salesman_id = (
    ->     SELECT salesman_id
    ->     FROM salesman
    ->     WHERE salesman_id = 5001
    -> );
Query OK, 1 row affected (0.03 sec)

mysql>  select* from orders WHERE salesman_id = 5001;
Empty set (0.00 sec)
```

PRACTICAL 2

2.1)Creating Database “Movies”



Review the SQL Script to be Applied on the Database

Online DDL
Algorithm: Default
Lock Type: Default

```
CREATE SCHEMA `movies` ;
```

Creating table “Actor”

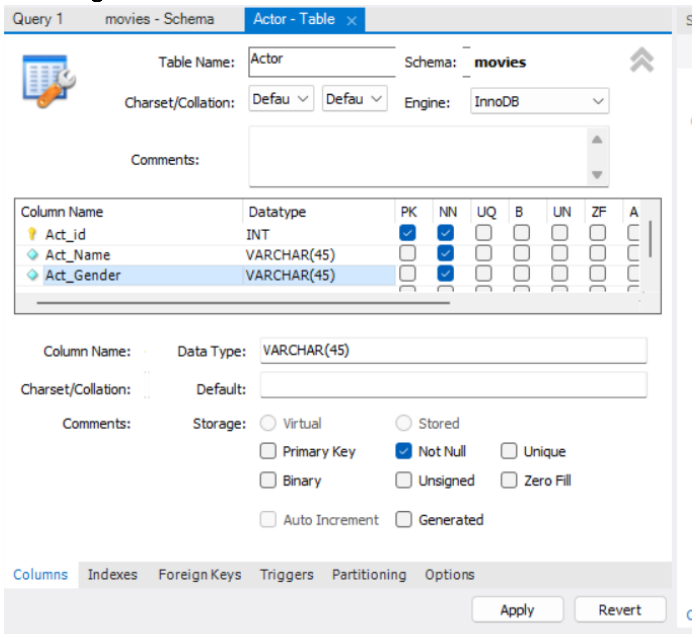
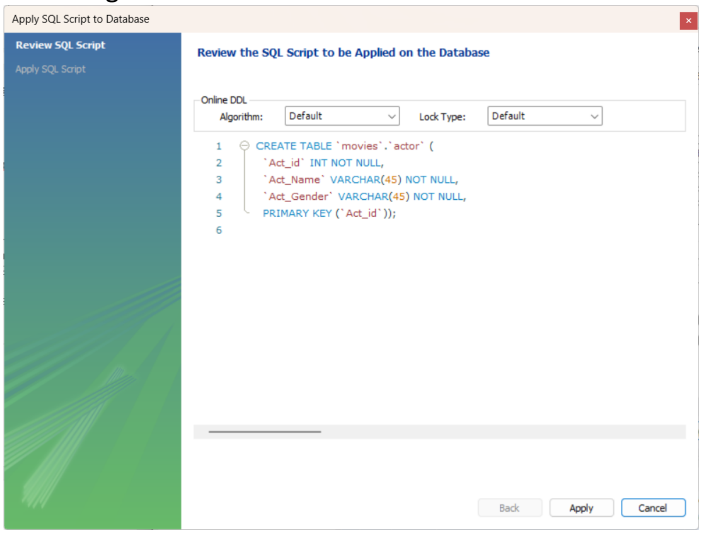


Table Name: Actor
Schema: movies
Charset/Collation: Defau Defau
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	A
Act_id	INT	✓	✓					
Act_Name	VARCHAR(45)		✓					
Act_Gender	VARCHAR(45)		✓					

Column Name:
Data Type: VARCHAR(45)
Charset/Collation:
Default:
Comments:
Storage: Virtual Stored
Primary Key [✓ Not Null] Unique
Binary Unsigned Zero Fill
Auto Increment Generated

Executing the command



Review the SQL Script to be Applied on the Database

Online DDL
Algorithm: Default
Lock Type: Default

```
CREATE TABLE `movies`.`actor` (
  `Act_id` INT NOT NULL,
  `Act_Name` VARCHAR(45) NOT NULL,
  `Act_Gender` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Act_id`));
```

Creating table “Movie”

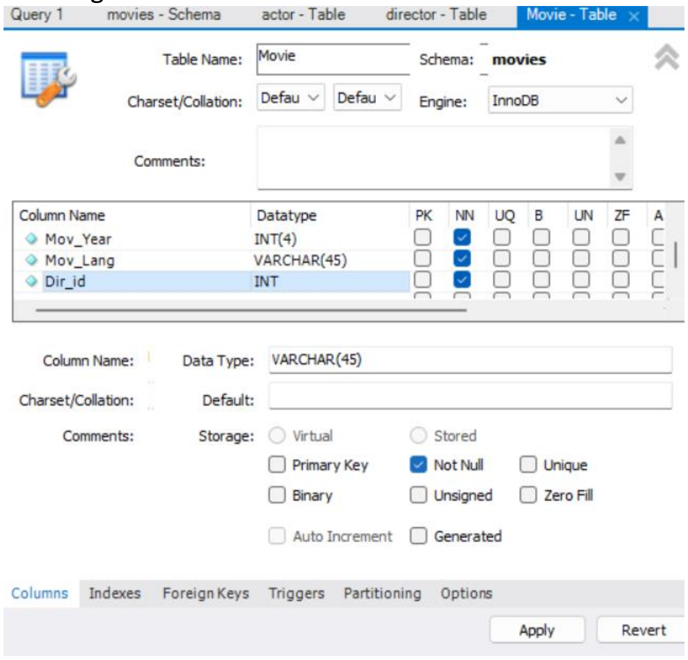


Table Name: Movie
Schema: movies
Charset/Collation: Defau Defau
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	A
Mov_Year	INT(4)	☐	✓	☐	☐	☐	☐	☐
Mov_Lang	VARCHAR(45)	☐	✓	☐	☐	☐	☐	☐
Dir_id	INT	☐	✓	☐	☐	☐	☐	☐

Column Name:
Data Type: VARCHAR(45)
Charset/Collation:
Default:
Comments:
Storage: Virtual Stored
☐ Primary Key ✓ Not Null ☐ Unique
☐ Binary ☐ Unsigned ☐ Zero Fill
☐ Auto Increment ☐ Generated

Adding a foreign key

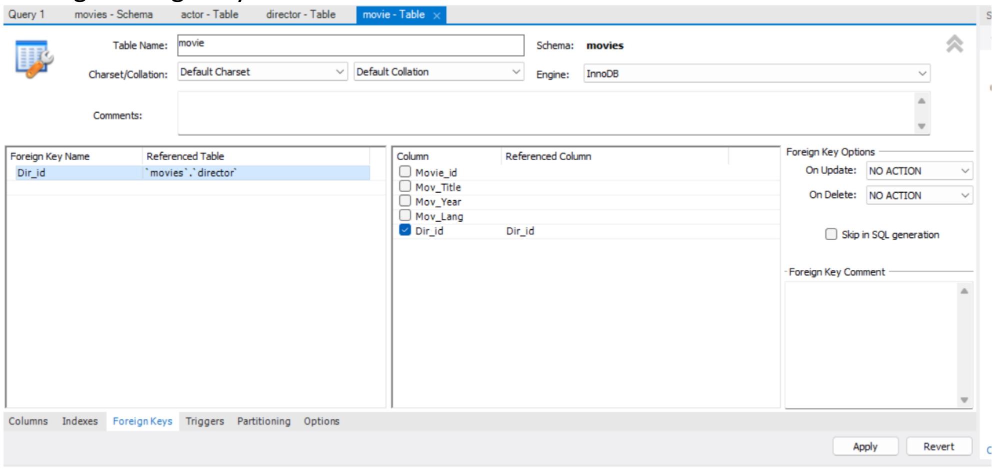


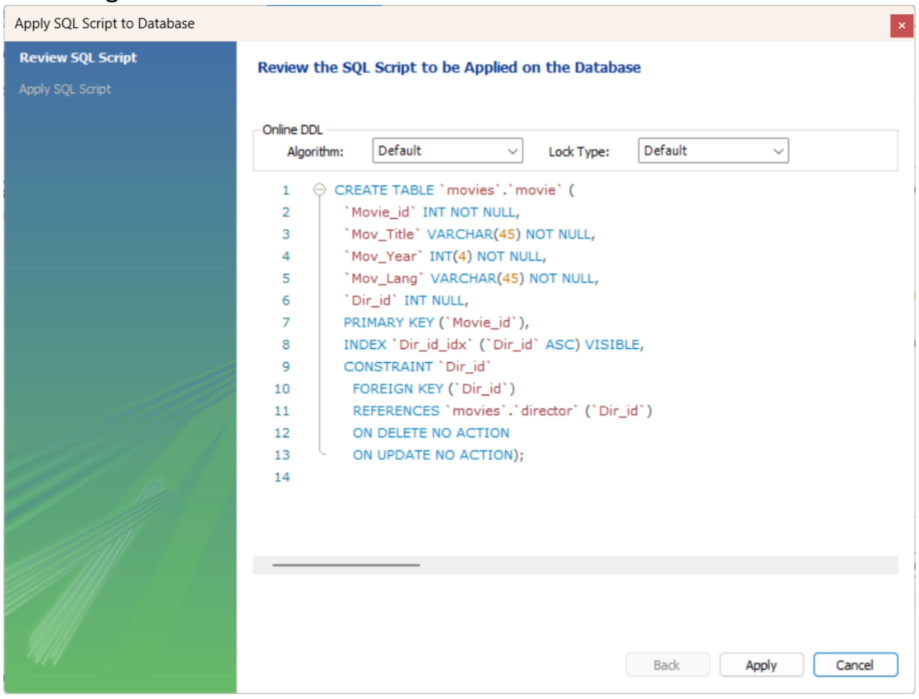
Table Name: movie
Schema: movies
Charset/Collation: Default Charset Default Collation
Engine: InnoDB
Comments:

Foreign Key Name	Referenced Table
Dir_id	`movies`.`director`

Column	Referenced Column
☐ Movie_id	
☐ Mov_Title	
☐ Mov_Year	
☐ Mov_Lang	
✓ Dir_id	Dir_id

Foreign Key Options
On Update: NO ACTION
On Delete: NO ACTION
☐ Skip in SQL generation
Foreign Key Comment

Executing the command



Apply SQL Script to Database

Review the SQL Script to be Applied on the Database

Online DDL
Algorithm: Default
Lock Type: Default

```
CREATE TABLE `movies`.`movie` (
  `Movie_id` INT NOT NULL,
  `Mov_Title` VARCHAR(45) NOT NULL,
  `Mov_Year` INT(4) NOT NULL,
  `Mov_Lang` VARCHAR(45) NOT NULL,
  `Dir_id` INT NULL,
  PRIMARY KEY (`Movie_id`),
  INDEX `Dir_id_idx` (`Dir_id` ASC) VISIBLE,
  CONSTRAINT `Dir_id`
    FOREIGN KEY (`Dir_id`)
    REFERENCES `movies`.`director` (`Dir_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

Creating table "Movie_Cast"

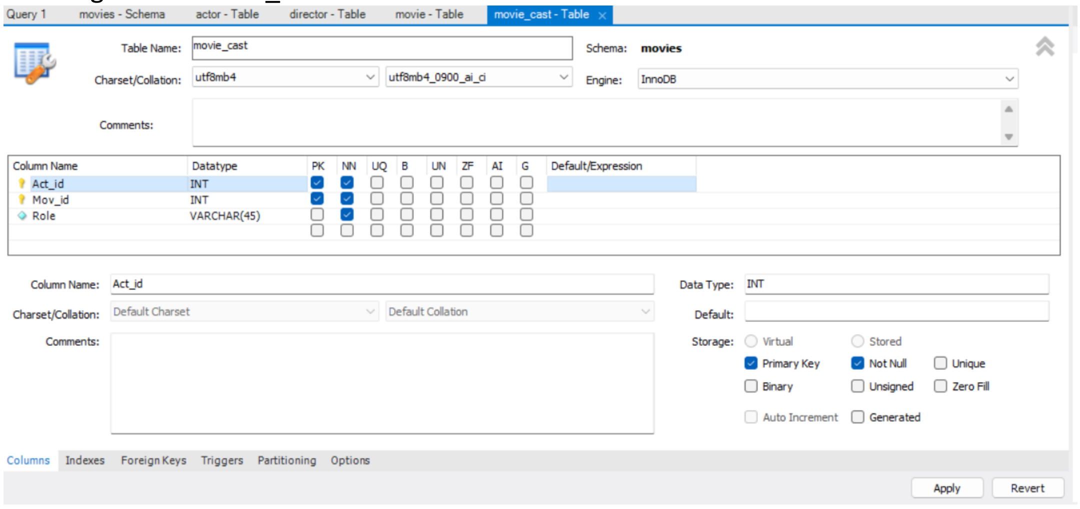


Table Name: movie_cast
Schema: movies
Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Act_id	INT	✓	✓							
Mov_id	INT	✓	✓							
Role	VARCHAR(45)		✓							

Column Name: Act_id
Data Type: INT
Charset/Collation: Default Charset / Default Collation
Default:
Comments:
Storage: Virtual / Stored
[✓ Primary Key] [✓ Not Null] Unique
Binary Unsigned Zero Fill
Auto Increment Generated

Adding a foreign key

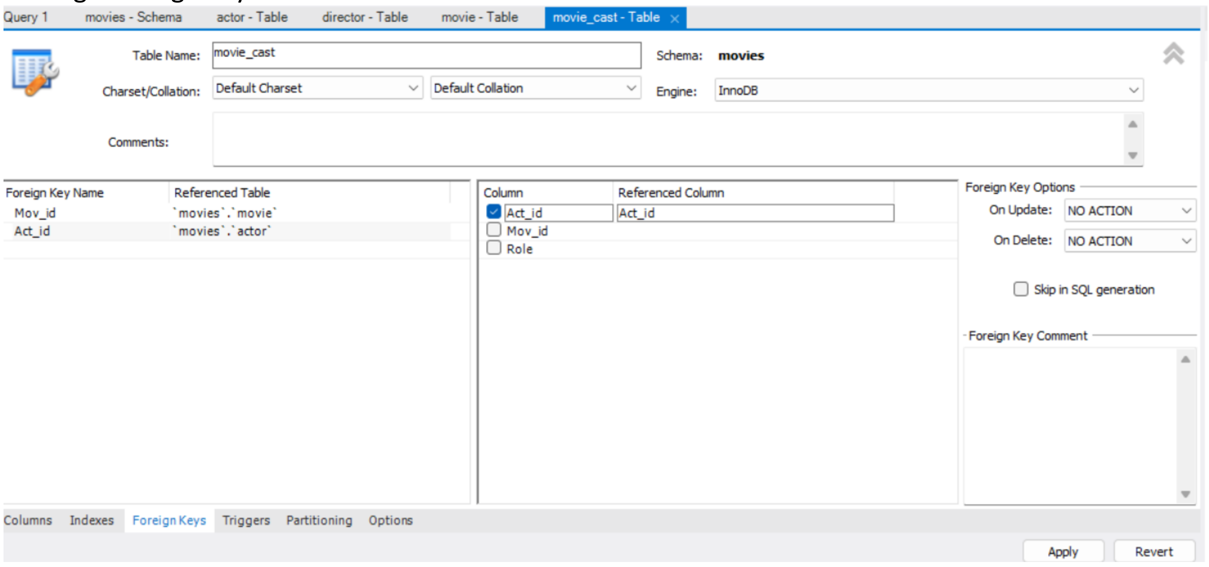


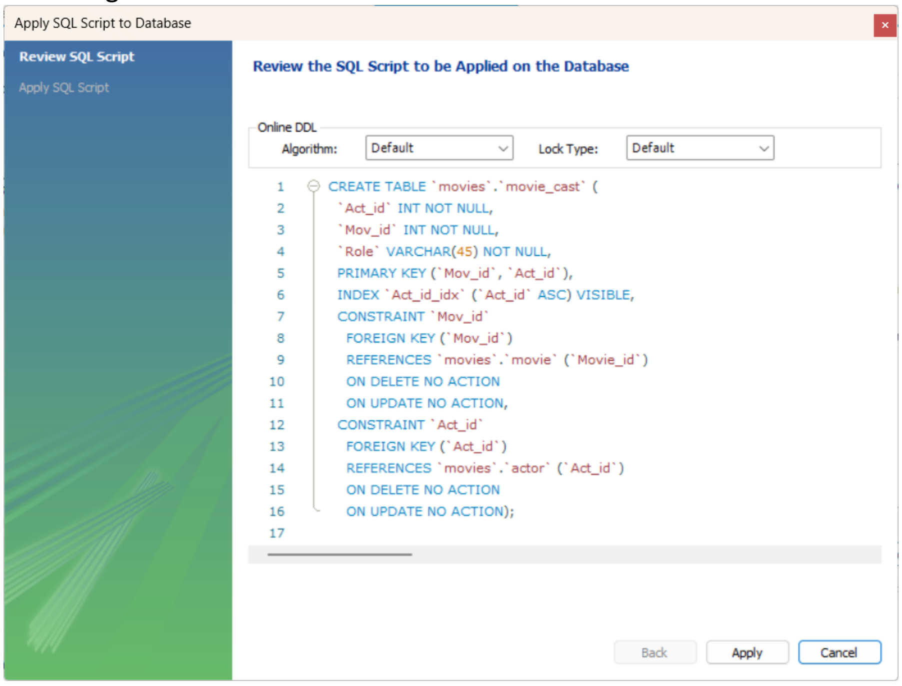
Table Name: movie_cast
Schema: movies
Charset/Collation: Default Charset / Default Collation
Engine: InnoDB
Comments:

Foreign Key Name	Referenced Table
Mov_id	`movies`.`movie`
Act_id	`movies`.`actor`

Column	Referenced Column
✓ Act_id	Act_id
Mov_id	
Role	

Foreign Key Options
On Update: NO ACTION
On Delete: NO ACTION
Skip in SQL generation
Foreign Key Comment

Executing the command



Review the SQL Script to be Applied on the Database

Online DDL
Algorithm: Default
Lock Type: Default

```
CREATE TABLE `movies`.`movie_cast` (
  `Act_id` INT NOT NULL,
  `Mov_id` INT NOT NULL,
  `Role` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Mov_id`, `Act_id`),
  INDEX `Act_id_idx` (`Act_id` ASC) VISIBLE,
  CONSTRAINT `Mov_id`
    FOREIGN KEY (`Mov_id`)
    REFERENCES `movies`.`movie` (`Movie_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `Act_id`
    FOREIGN KEY (`Act_id`)
    REFERENCES `movies`.`actor` (`Act_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

Creating table “Ratings”

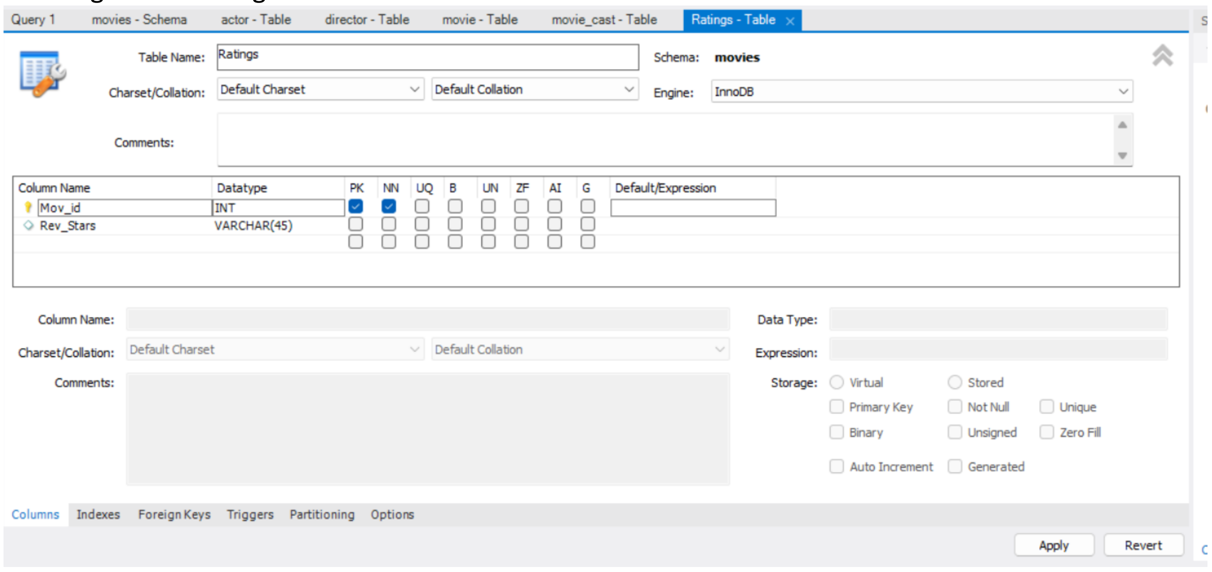


Table Name: Ratings
Schema: movies
Charset/Collation: Default Charset, Default Collation
Engine: InnoDB

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Mov_id	INT	✓	✓							
Rev_Stars	VARCHAR(45)									

Adding a foreign key

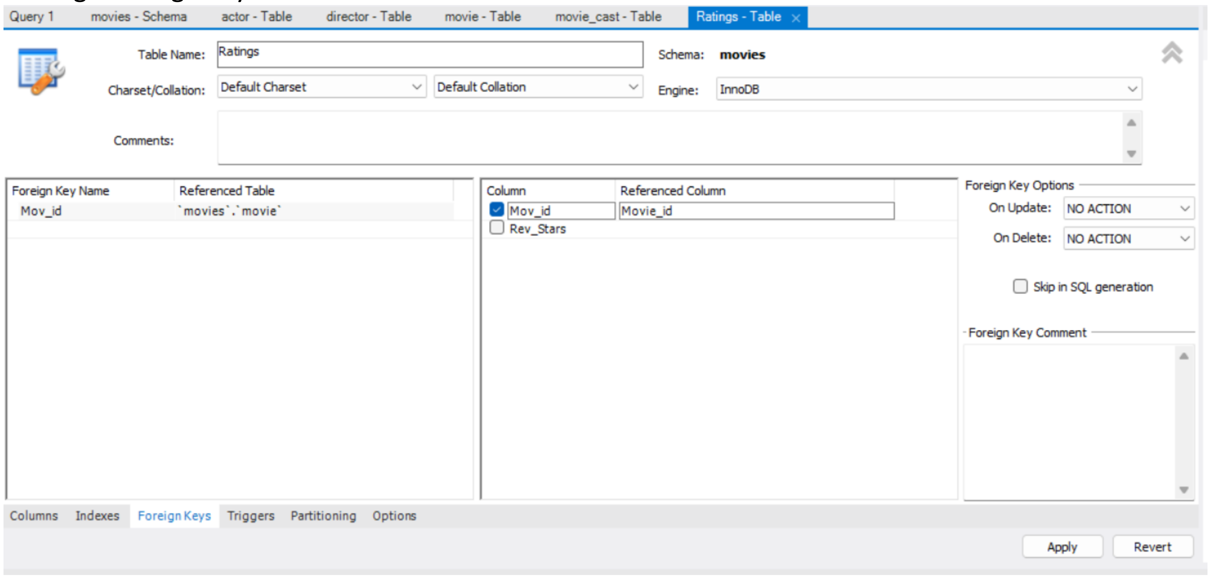


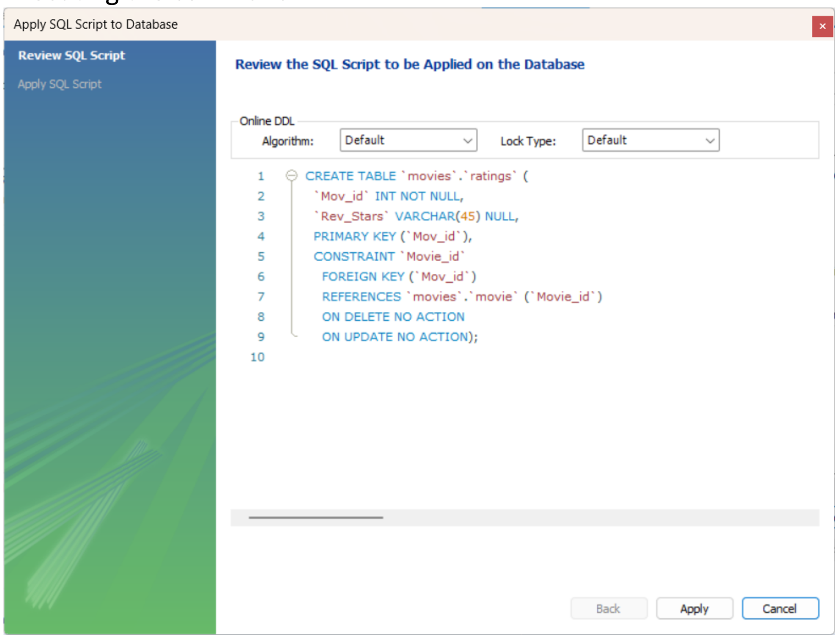
Table Name: Ratings
Schema: movies
Charset/Collation: Default Charset, Default Collation
Engine: InnoDB

Foreign Key Name	Referenced Table
Mov_id	`movies`.`movie`

Column	Referenced Column
✓ Mov_id	Movie_id
Rev_Stars	

Foreign Key Options
On Update: NO ACTION
On Delete: NO ACTION
Skip in SQL generation
Foreign Key Comment

Executing the command



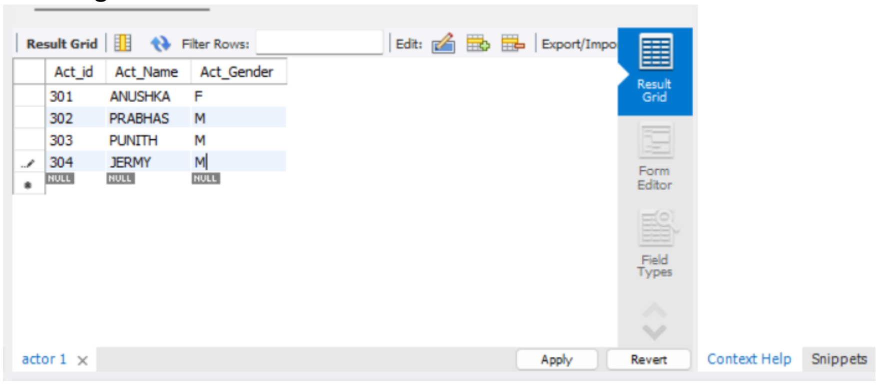
Apply SQL Script to Database

Review the SQL Script to be Applied on the Database

Online DDL
Algorithm: Default
Lock Type: Default

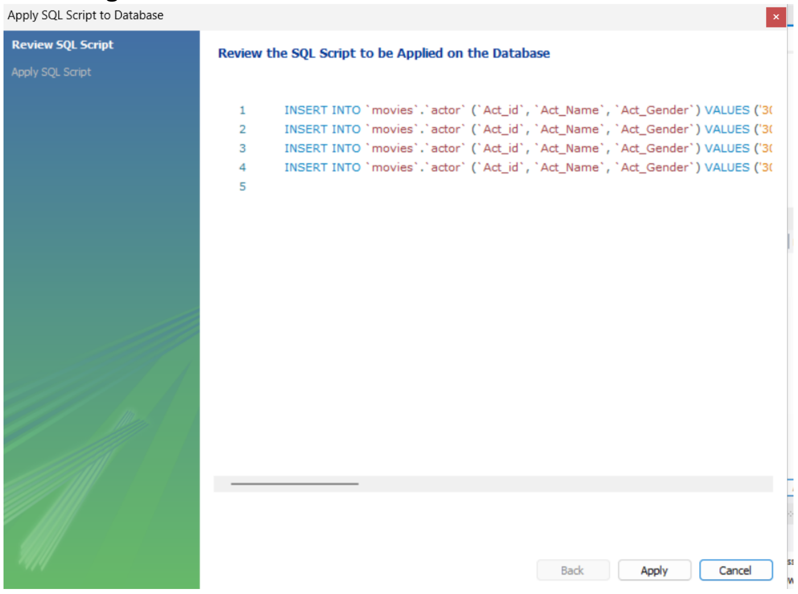
```
CREATE TABLE `movies`.`ratings` (
  `Mov_id` INT NOT NULL,
  `Rev_Stars` VARCHAR(45) NULL,
  PRIMARY KEY (`Mov_id`),
  CONSTRAINT `Movie_id`
    FOREIGN KEY (`Mov_id`)
    REFERENCES `movies`.`movie` (`Movie_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

Inserting data into “Actor”



Act_id	Act_Name	Act_Gender
301	ANUSHKA	F
302	PRABHAS	M
303	PUNITH	M
304	JERMY	M
NULL	NULL	NULL

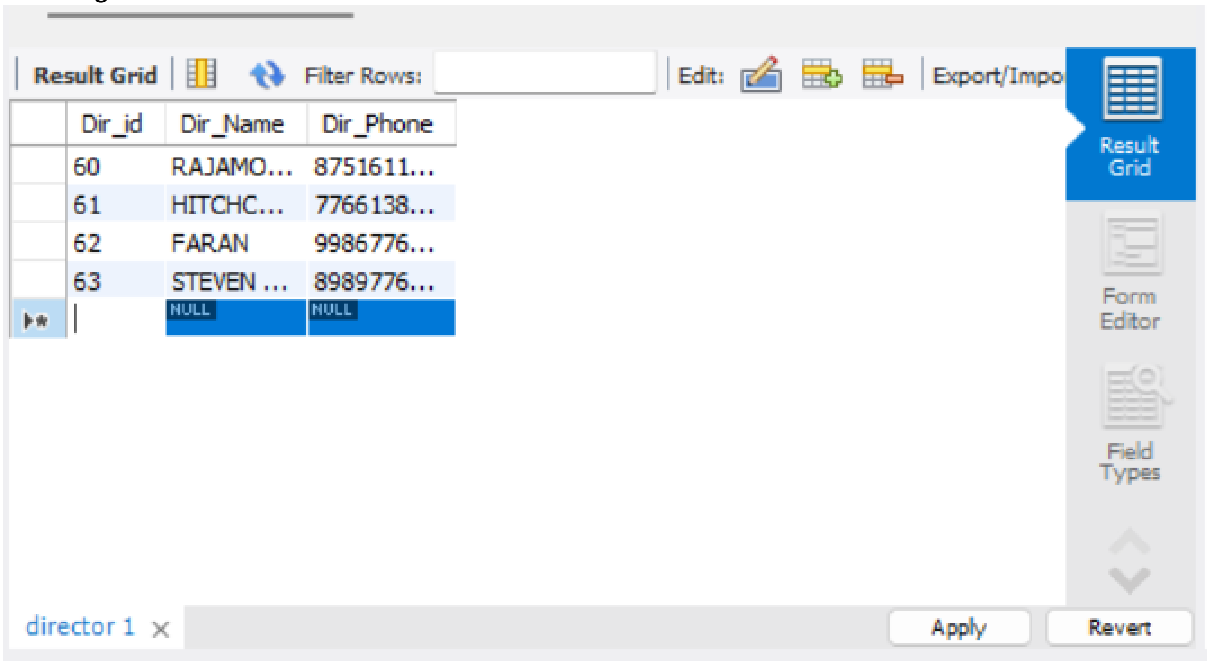
Executing the command



Review the SQL Script to be Applied on the Database

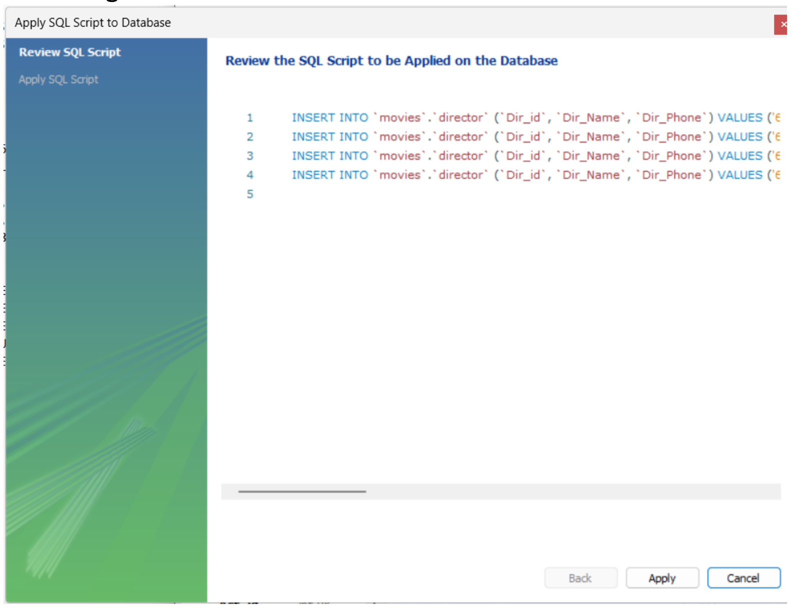
```
INSERT INTO `movies`.`actor` (`Act_id`, `Act_Name`, `Act_Gender`) VALUES ('3(
INSERT INTO `movies`.`actor` (`Act_id`, `Act_Name`, `Act_Gender`) VALUES ('3(
INSERT INTO `movies`.`actor` (`Act_id`, `Act_Name`, `Act_Gender`) VALUES ('3(
INSERT INTO `movies`.`actor` (`Act_id`, `Act_Name`, `Act_Gender`) VALUES ('3(
```

Inserting data into “Director”



Dir_id	Dir_Name	Dir_Phone
60	RAJAMO...	8751611...
61	HITCHC...	7766138...
62	FARAN	9986776...
63	STEVEN ...	8989776...
	NULL	NULL

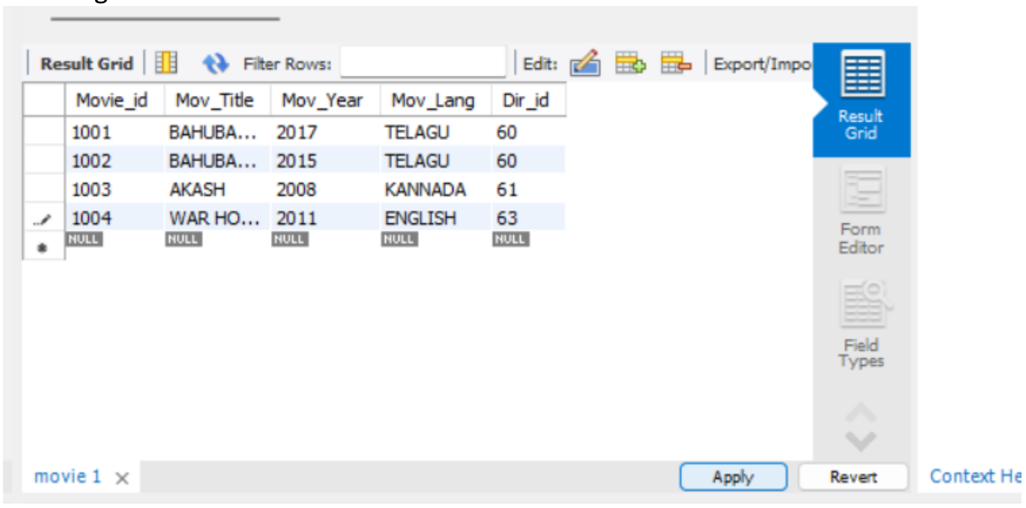
Executing the command



Review the SQL Script to be Applied on the Database

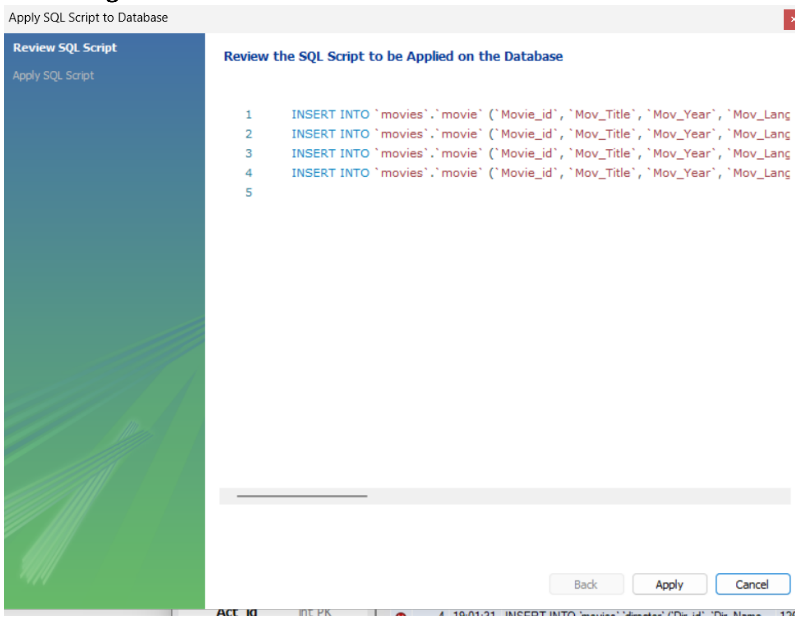
```
INSERT INTO `movies`.`director` (`Dir_id`, `Dir_Name`, `Dir_Phone`) VALUES ('6
INSERT INTO `movies`.`director` (`Dir_id`, `Dir_Name`, `Dir_Phone`) VALUES ('6
INSERT INTO `movies`.`director` (`Dir_id`, `Dir_Name`, `Dir_Phone`) VALUES ('6
INSERT INTO `movies`.`director` (`Dir_id`, `Dir_Name`, `Dir_Phone`) VALUES ('6
```

Inserting data into “Movie”



Movie_id	Mov_Title	Mov_Year	Mov_Lang	Dir_id
1001	BAHUBA...	2017	TELAGU	60
1002	BAHUBA...	2015	TELAGU	60
1003	AKASH	2008	KANNADA	61
1004	WAR HO...	2011	ENGLISH	63
NULL	NULL	NULL	NULL	NULL

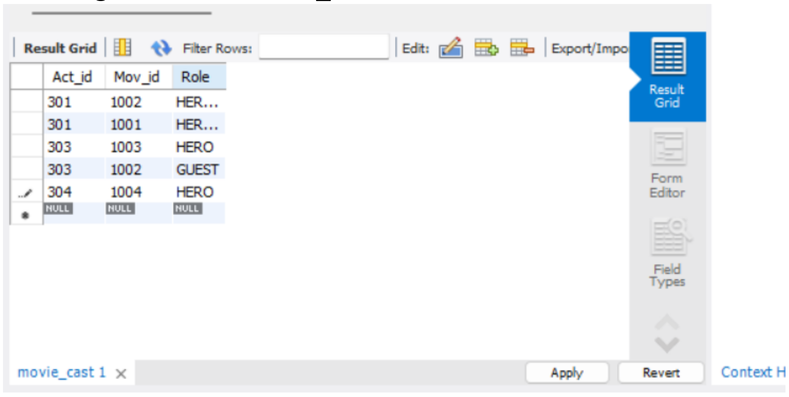
Executing the command



Review the SQL Script to be Applied on the Database

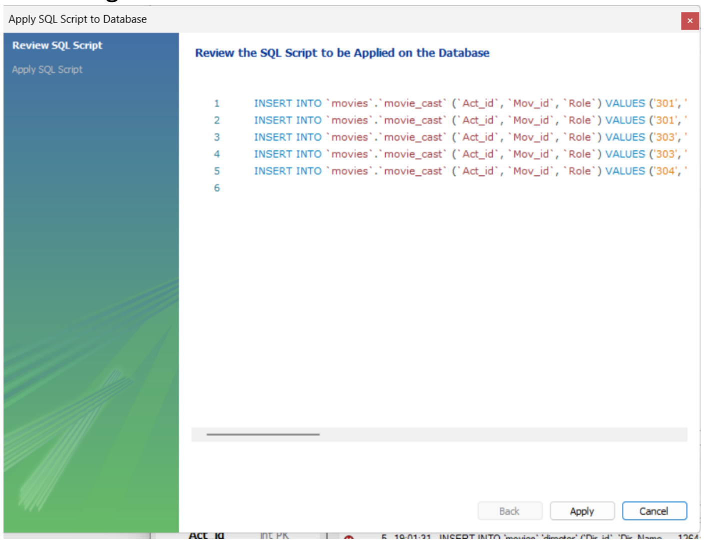
```
INSERT INTO `movies`.`movie` (`Movie_id`, `Mov_Title`, `Mov_Year`, `Mov_Lang
INSERT INTO `movies`.`movie` (`Movie_id`, `Mov_Title`, `Mov_Year`, `Mov_Lang
INSERT INTO `movies`.`movie` (`Movie_id`, `Mov_Title`, `Mov_Year`, `Mov_Lang
INSERT INTO `movies`.`movie` (`Movie_id`, `Mov_Title`, `Mov_Year`, `Mov_Lang
```

Inserting data into “Movie_Cast”



Act_id	Mov_id	Role
301	1002	HER...
301	1001	HER...
303	1003	HERO
303	1002	GUEST
304	1004	HERO
NULL	NULL	NULL

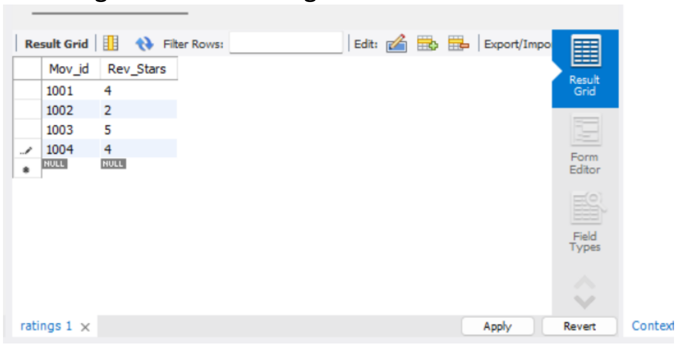
Executing the command



Review the SQL Script to be Applied on the Database

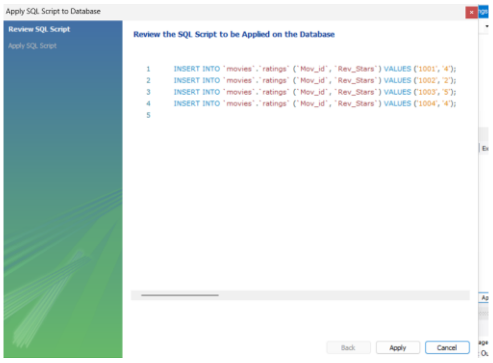
```
INSERT INTO `movies`.`movie_cast` (`Act_id`, `Mov_id`, `Role`) VALUES ('301', '
INSERT INTO `movies`.`movie_cast` (`Act_id`, `Mov_id`, `Role`) VALUES ('301', '
INSERT INTO `movies`.`movie_cast` (`Act_id`, `Mov_id`, `Role`) VALUES ('303', '
INSERT INTO `movies`.`movie_cast` (`Act_id`, `Mov_id`, `Role`) VALUES ('303', '
INSERT INTO `movies`.`movie_cast` (`Act_id`, `Mov_id`, `Role`) VALUES ('304', '
```

Inserting data into “Ratings”



Mov_id	Rev_Stars
1001	4
1002	2
1003	5
1004	4
NULL	NULL

Executing the command

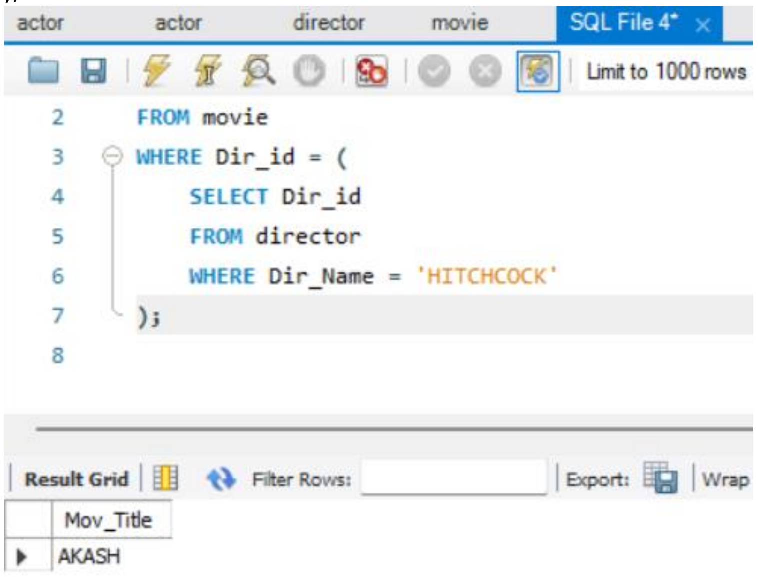


Review the SQL Script to be Applied on the Database

```
INSERT INTO `movies`.`ratings` (`Mov_id`, `Rev_Stars`) VALUES ('1001', '4');
INSERT INTO `movies`.`ratings` (`Mov_id`, `Rev_Stars`) VALUES ('1002', '2');
INSERT INTO `movies`.`ratings` (`Mov_id`, `Rev_Stars`) VALUES ('1003', '5');
INSERT INTO `movies`.`ratings` (`Mov_id`, `Rev_Stars`) VALUES ('1004', '4');
```

1. List the titles of all movies directed by ‘Hitchcock’.

```
SELECT Mov_Title
FROM movie
WHERE Dir_id = (
   SELECT Dir_id
   FROM director
   WHERE Dir_Name = 'HITCHCOCK'
);
```

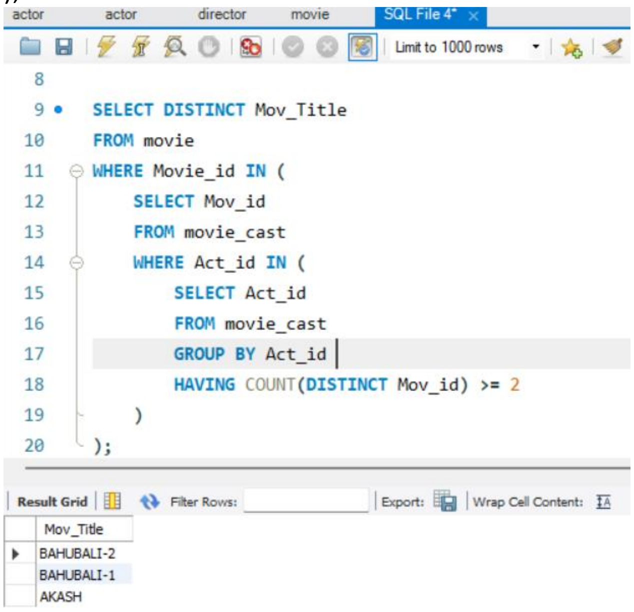


```
FROM movie
WHERE Dir_id = (
    SELECT Dir_id
    FROM director
    WHERE Dir_Name = 'HITCHCOCK'
);
```

Mov_Title
AKASH

2. Find the movie names where one or more actors acted in two or more movies.

```
SELECT DISTINCT Mov_Title
FROM movie
WHERE Movie_id IN (
   SELECT Mov_id
   FROM movie_cast
   WHERE Act_id IN (
      SELECT Act_id
      FROM movie_cast
      GROUP BY Act_id
      HAVING COUNT(DISTINCT Mov_id) >= 2
   )
);
```

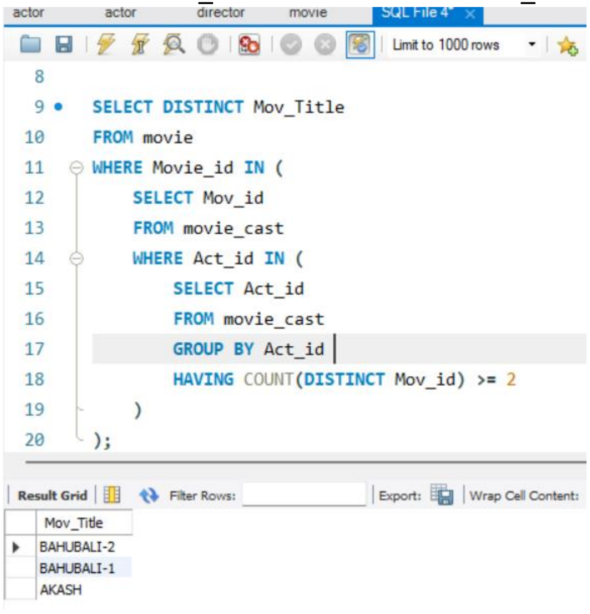


```
SELECT DISTINCT Mov_Title
FROM movie
WHERE Movie_id IN (
    SELECT Mov_id
    FROM movie_cast
    WHERE Act_id IN (
        SELECT Act_id
        FROM movie_cast
        GROUP BY Act_id
        HAVING COUNT(DISTINCT Mov_id) >= 2
    )
);
```

Mov_Title
BAHUBALI-2
BAHUBALI-1
AKASH

3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

```
SELECT DISTINCT a.Act_Name
FROM actor a
JOIN movie_cast mc1 ON a.Act_id = mc1.Act_id
JOIN movie m1 ON mc1.Mov_id = m1.Movie_id
JOIN movie_cast mc2 ON a.Act_id = mc2.Act_id
JOIN movie m2 ON mc2.Mov_id = m2.Movie_id
WHERE m1.Mov_Year < 2000 AND m2.Mov_Year > 2015;
```

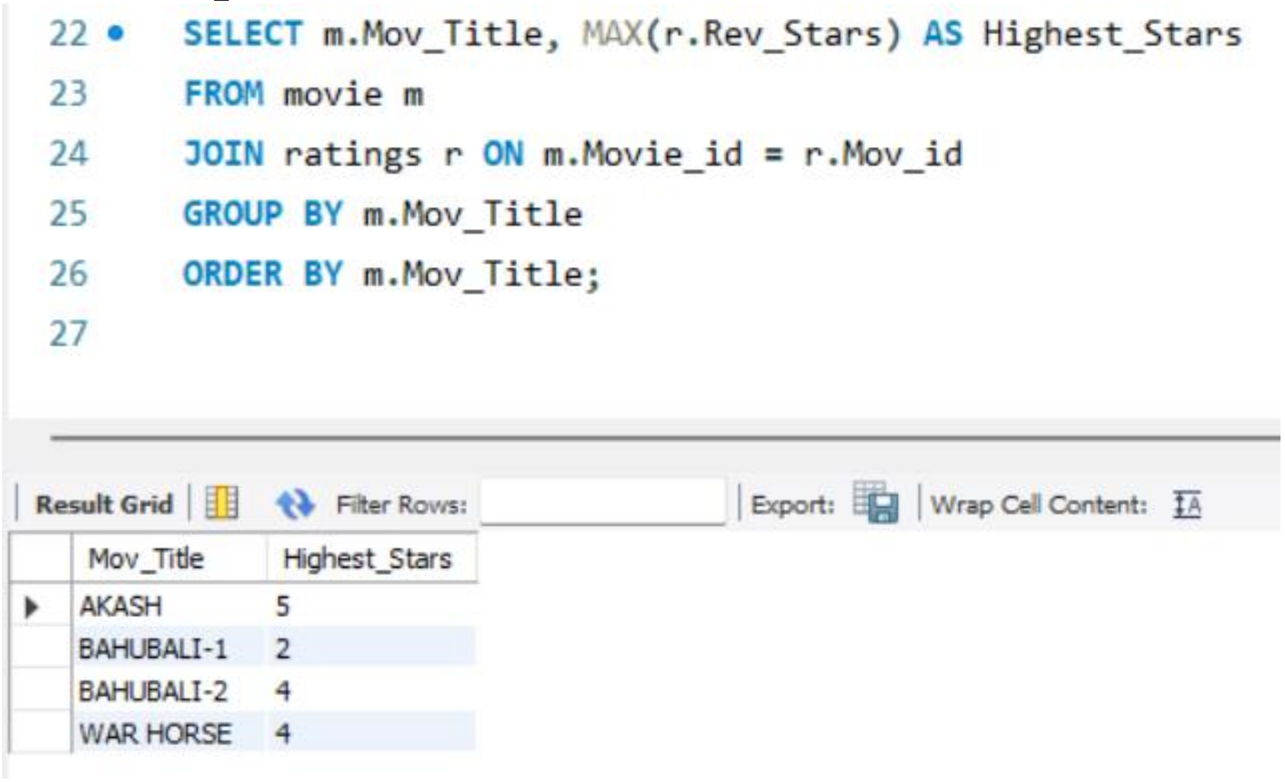


```
SELECT DISTINCT Mov_Title
FROM movie
WHERE Movie_id IN (
    SELECT Mov_id
    FROM movie_cast
    WHERE Act_id IN (
        SELECT Act_id
        FROM movie_cast
        GROUP BY Act_id
        HAVING COUNT(DISTINCT Mov_id) >= 2
    )
);
```

Mov_Title
BAHUBALI-2
BAHUBALI-1
AKASH

4. Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.

```
SELECT m.Mov_Title, MAX(r.Rev_Stars) AS Highest_Stars
FROM movie m
JOIN ratings r ON m.Movie_id = r.Mov_id
GROUP BY m.Mov_Title
ORDER BY m.Mov_Title;
```

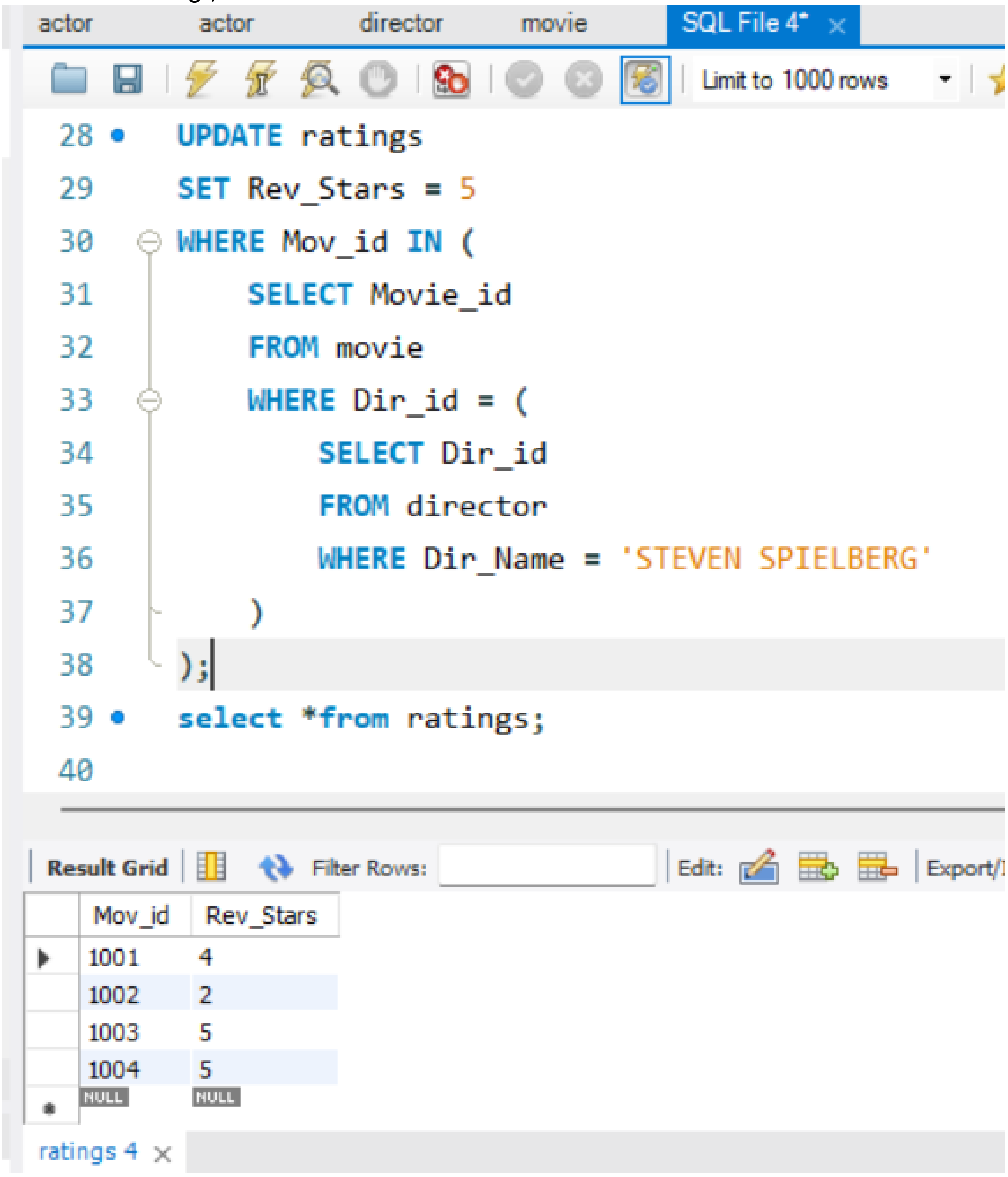


```
SELECT m.Mov_Title, MAX(r.Rev_Stars) AS Highest_Stars
FROM movie m
JOIN ratings r ON m.Movie_id = r.Mov_id
GROUP BY m.Mov_Title
ORDER BY m.Mov_Title;
```

Mov_Title	Highest_Stars
AKASH	5
BAHUBALI-1	2
BAHUBALI-2	4
WAR HORSE	4

5. Update rating of all movies directed by ‘Steven Spielberg’ to 5.

```
UPDATE ratings
SET Rev_Stars = 5
WHERE Mov_id IN (
   SELECT Movie_id
   FROM movie
   WHERE Dir_id = (
      SELECT Dir_id
      FROM director
      WHERE Dir_Name = 'STEVEN SPIELBERG'
   )
);
select *from ratings;
```



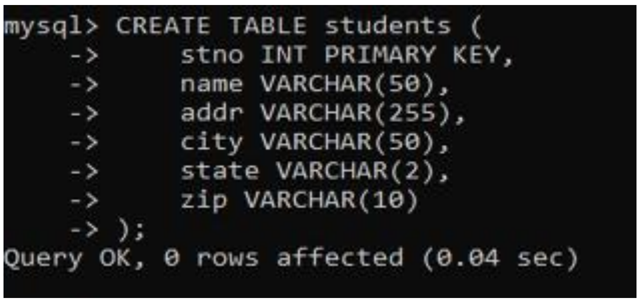
```
UPDATE ratings
SET Rev_Stars = 5
WHERE Mov_id IN (
    SELECT Movie_id
    FROM movie
    WHERE Dir_id = (
        SELECT Dir_id
        FROM director
        WHERE Dir_Name = 'STEVEN SPIELBERG'
    )
);
select *from ratings;
```

Mov_id	Rev_Stars
1001	4
1002	2
1003	5
1004	5
NULL	NULL

2.2 Design ERD for the following schema and execute the following Queries on it:

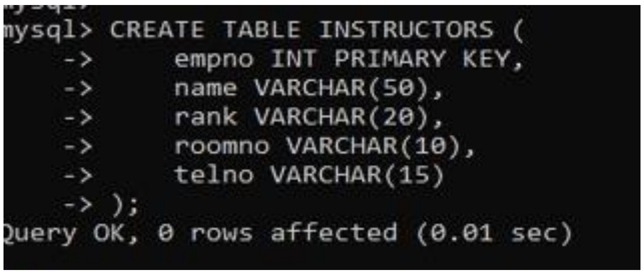
Code:

```
CREATE TABLE students (   stno INT
PRIMARY KEY,    name
VARCHAR(50),    addr
VARCHAR(255),    city
VARCHAR(50),    state VARCHAR(2),
zip VARCHAR(10)
);
```



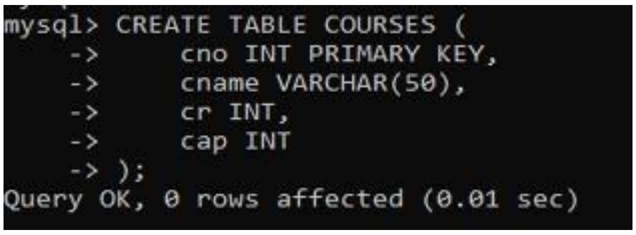
```
mysql> CREATE TABLE students (
    ->     stno INT PRIMARY KEY,
    ->     name VARCHAR(50),
    ->     addr VARCHAR(255),
    ->     city VARCHAR(50),
    ->     state VARCHAR(2),
    ->     zip VARCHAR(10)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
CREATE TABLE INSTRUCTORS (   empno INT
PRIMARY KEY,    name VARCHAR(50),
rank VARCHAR(20),    roomno
VARCHAR(10),    telno VARCHAR(15)
);
```



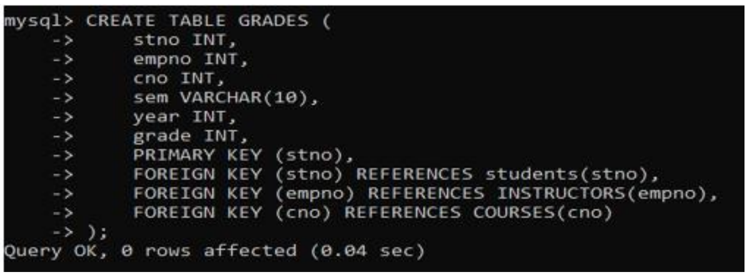
```
mysql> CREATE TABLE INSTRUCTORS (
    ->     empno INT PRIMARY KEY,
    ->     name VARCHAR(50),
    ->     rank VARCHAR(20),
    ->     roomno VARCHAR(10),
    ->     telno VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

```
CREATE TABLE COURSES (
    cno INT PRIMARY KEY,
    cname VARCHAR(50),
    cr INT,    cap
INT
);
```



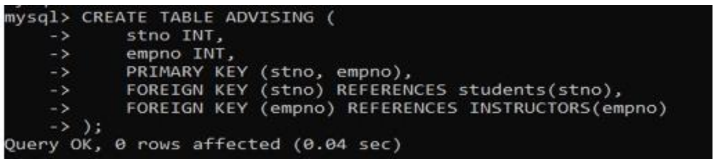
```
mysql> CREATE TABLE COURSES (
    ->     cno INT PRIMARY KEY,
    ->     cname VARCHAR(50),
    ->     cr INT,
    ->     cap INT
    -> );
Query OK, 0 rows affected (0.01 sec)
```

```
CREATE TABLE GRADES (
   stno INT,
   empno INT,   cno INT,
   sem VARCHAR(10),
   year INT,   grade INT,
   PRIMARY KEY (stno),
   FOREIGN KEY (stno) REFERENCES students(stno),
   FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno),
   FOREIGN KEY (cno) REFERENCES COURSES(cno)
);
```



```
mysql> CREATE TABLE GRADES (
    ->     stno INT,
    ->     empno INT,
    ->     cno INT,
    ->     sem VARCHAR(10),
    ->     year INT,
    ->     grade INT,
    ->     PRIMARY KEY (stno),
    ->     FOREIGN KEY (stno) REFERENCES students(stno),
    ->     FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno),
    ->     FOREIGN KEY (cno) REFERENCES COURSES(cno)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
CREATE TABLE ADVISING (
   stno INT,   empno INT,
   PRIMARY KEY (stno, empno),
   FOREIGN KEY (stno) REFERENCES students(stno),
   FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno)
);
```



```
mysql> CREATE TABLE ADVISING (
    ->     stno INT,
    ->     empno INT,
    ->     PRIMARY KEY (stno, empno),
    ->     FOREIGN KEY (stno) REFERENCES students(stno),
    ->     FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

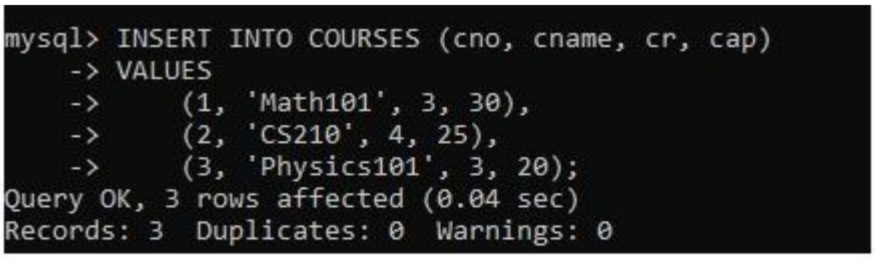
INSERT INTO COURSES (cno, cname, cr, cap)

VALUES

(1, 'Math101', 3, 30),

(2, 'CS210', 4, 25),

(3, 'Physics101', 3, 20);



```
mysql> INSERT INTO COURSES (cno, cname, cr, cap)
    -> VALUES
    ->     (1, 'Math101', 3, 30),
    ->     (2, 'CS210', 4, 25),
    ->     (3, 'Physics101', 3, 20);
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

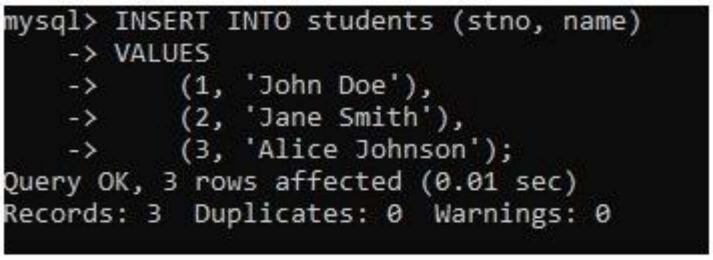
INSERT INTO students (stno, name)

VALUES

(1, 'John Doe'),

(2, 'Jane Smith'),

(3, 'Alice Johnson');



```
mysql> INSERT INTO students (stno, name)
    -> VALUES
    ->     (1, 'John Doe'),
    ->     (2, 'Jane Smith'),
    ->     (3, 'Alice Johnson');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

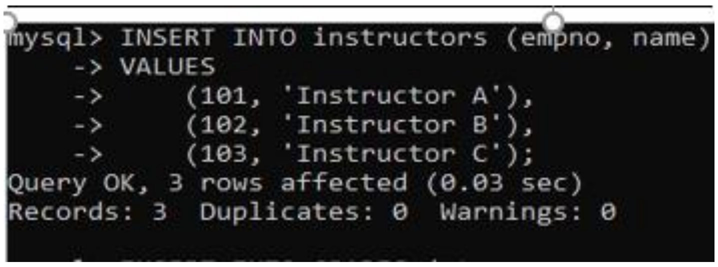
INSERT INTO instructors (empno, name)

VALUES

(101, 'Instructor A'),

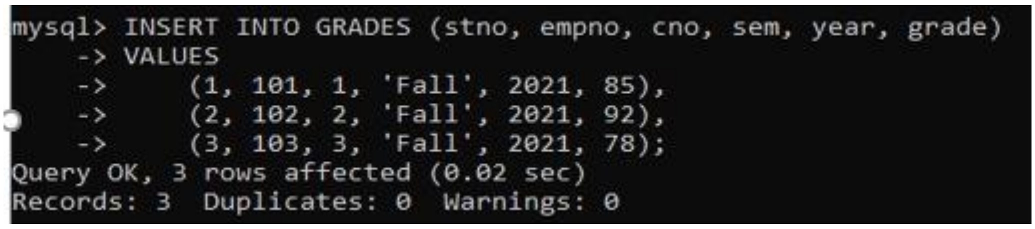
(102, 'Instructor B'),

(103, 'Instructor C');



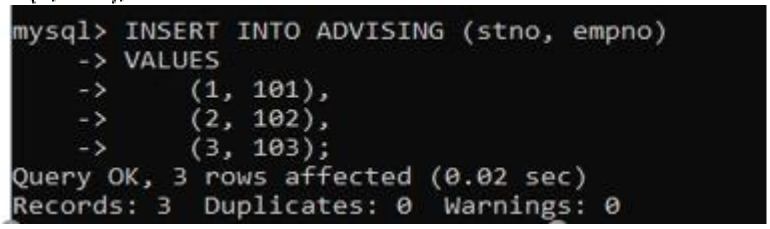
```
mysql> INSERT INTO instructors (empno, name)
    -> VALUES
    ->     (101, 'Instructor A'),
    ->     (102, 'Instructor B'),
    ->     (103, 'Instructor C');
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

INSERT INTO GRADES (stno, empno, cno, sem, year, grade)
VALUES
(1, 101, 1, 'Fall', 2021, 85),
(2, 102, 2, 'Fall', 2021, 92),
(3, 103, 3, 'Fall', 2021, 78);



```
mysql> INSERT INTO GRADES (stno, empno, cno, sem, year, grade)
    -> VALUES
    ->     (1, 101, 1, 'Fall', 2021, 85),
    ->     (2, 102, 2, 'Fall', 2021, 92),
    ->     (3, 103, 3, 'Fall', 2021, 78);
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

INSERT INTO ADVISING (stno, empno)
VALUES
(1, 101),
(2, 102),
(3, 103);

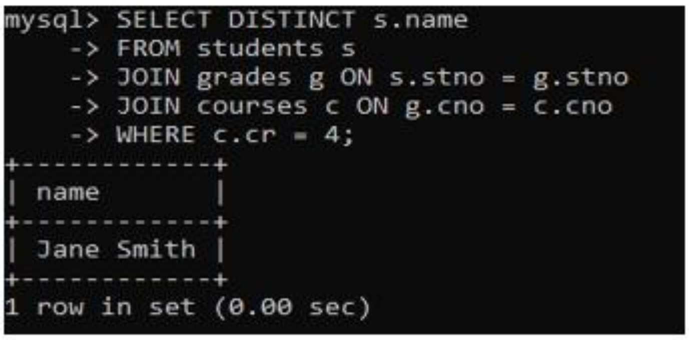


```
mysql> INSERT INTO ADVISING (stno, empno)
    -> VALUES
    ->     (1, 101),
    ->     (2, 102),
    ->     (3, 103);
Query OK, 3 rows affected (0.02 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

1. **Find the names of students who took some four-credit courses.**

Code:

SELECT DISTINCT s.name
FROM students s
JOIN grades g ON s.stno = g.stno
JOIN courses c ON g.cno = c.cno
WHERE c.cr = 4; **Output:**



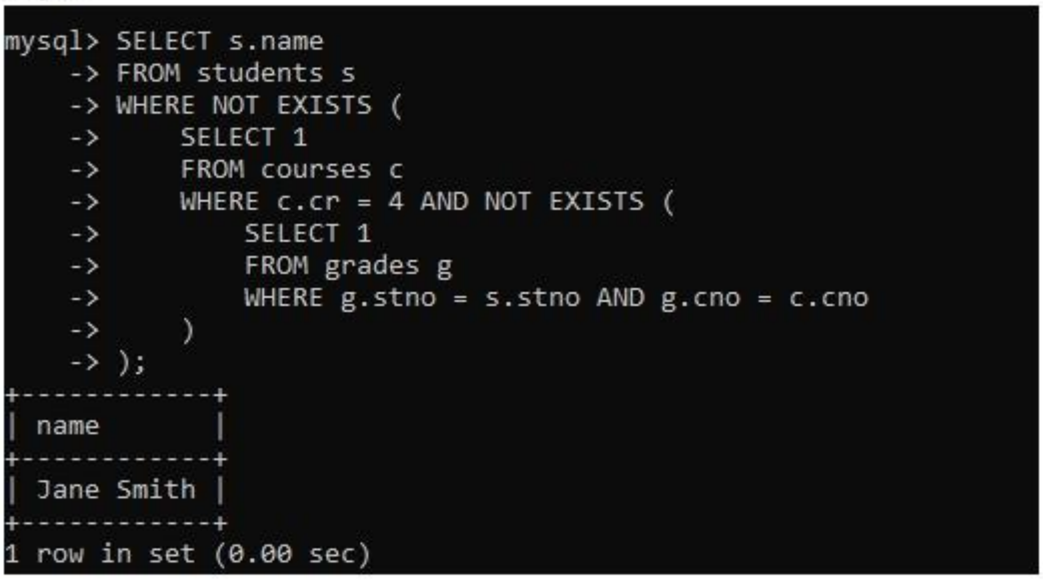
```
mysql> SELECT DISTINCT s.name
    -> FROM students s
    -> JOIN grades g ON s.stno = g.stno
    -> JOIN courses c ON g.cno = c.cno
    -> WHERE c.cr = 4;
+------------+
| name       |
+------------+
| Jane Smith |
+------------+
1 row in set (0.00 sec)
```

2. Find the names of students who took every four-credit course.

Code:

```
SELECT s.name
FROM students s
WHERE NOT EXISTS (
  SELECT 1
  FROM courses c
  WHERE c.cr = 4 AND NOT EXISTS (
    SELECT 1
    FROM grades g
    WHERE g.stno = s.stno AND g.cno = c.cno
  )
);
```

Output:

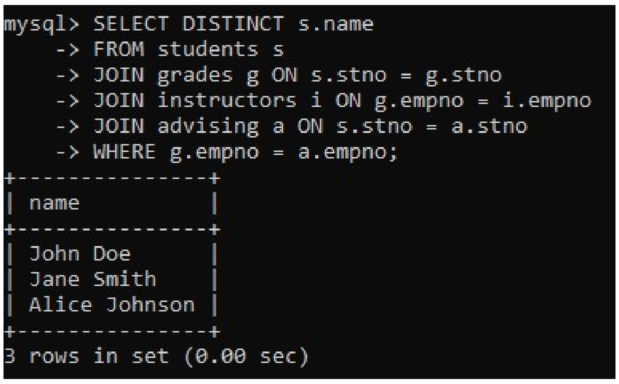


```
mysql> SELECT s.name
    -> FROM students s
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM courses c
    ->     WHERE c.cr = 4 AND NOT EXISTS (
    ->         SELECT 1
    ->         FROM grades g
    ->         WHERE g.stno = s.stno AND g.cno = c.cno
    ->     )
    -> );
+------------+
| name       |
+------------+
| Jane Smith |
+------------+
1 row in set (0.00 sec)
```

3. Find the names of students who took a course with an instructor who is also their advisor.

Code:

```
SELECT DISTINCT s.name
FROM students s
JOIN grades g ON s.stno = g.stno
JOIN instructors i ON g.empno = i.empno
JOIN advising a ON s.stno = a.stno
WHERE g.empno = a.empno;
```
Output:



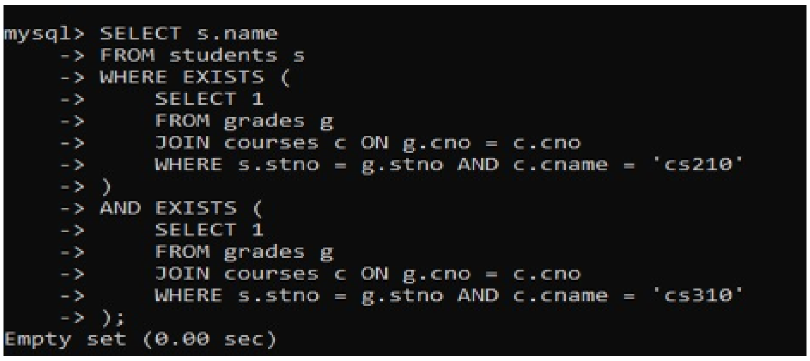
```
mysql> SELECT DISTINCT s.name
    -> FROM students s
    -> JOIN grades g ON s.stno = g.stno
    -> JOIN instructors i ON g.empno = i.empno
    -> JOIN advising a ON s.stno = a.stno
    -> WHERE g.empno = a.empno;
+---------------+
| name          |
+---------------+
| John Doe      |
| Jane Smith    |
| Alice Johnson |
+---------------+
3 rows in set (0.00 sec)
```

4. Find the names of students who took cs210 and cs310.

Code:

```
SELECT s.name
FROM students s
WHERE EXISTS (
   SELECT 1
   FROM grades g
   JOIN courses c ON g.cno = c.cno
   WHERE s.stno = g.stno AND c.cname = 'cs210'
)
AND EXISTS (
   SELECT 1
   FROM grades g
   JOIN courses c ON g.cno = c.cno
   WHERE s.stno = g.stno AND c.cname = 'cs310'
);
```

Output:



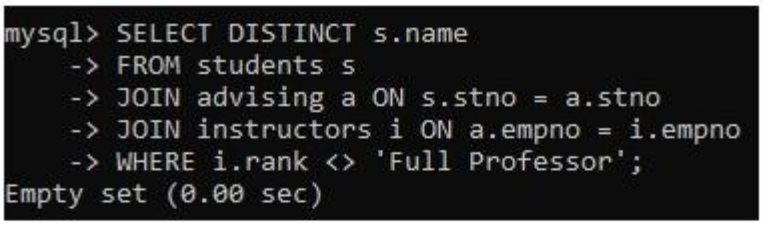
```
mysql> SELECT s.name
    -> FROM students s
    -> WHERE EXISTS (
    ->     SELECT 1
    ->     FROM grades g
    ->     JOIN courses c ON g.cno = c.cno
    ->     WHERE s.stno = g.stno AND c.cname = 'cs210'
    -> )
    -> AND EXISTS (
    ->     SELECT 1
    ->     FROM grades g
    ->     JOIN courses c ON g.cno = c.cno
    ->     WHERE s.stno = g.stno AND c.cname = 'cs310'
    -> );
Empty set (0.00 sec)
```

5. Find the names of all students whose advisor is not a full professor.

Code:

```
SELECT DISTINCT s.name
FROM students s
JOIN advising a ON s.stno = a.stno
JOIN instructors i ON a.empno = i.empno
WHERE i.rank <> 'Full Professor';
```

Output:

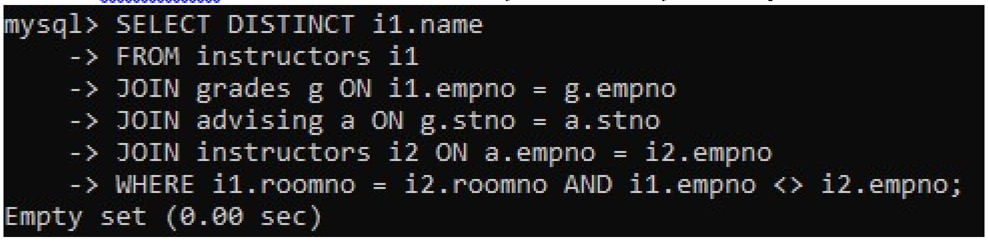


```
mysql> SELECT DISTINCT s.name
    -> FROM students s
    -> JOIN advising a ON s.stno = a.stno
    -> JOIN instructors i ON a.empno = i.empno
    -> WHERE i.rank <> 'Full Professor';
Empty set (0.00 sec)
```

6. **Find instructors who taught students who are advised by another instructor who shares the same room.**

Code:

SELECT DISTINCT i1.name
FROM instructors i1
JOIN grades g ON i1.empno = g.empno
JOIN advising a ON g.stno = a.stno
JOIN instructors i2 ON a.empno = i2.empno
WHERE i1.roomno = i2.roomno AND i1.empno <> i2.empno; **Output:**

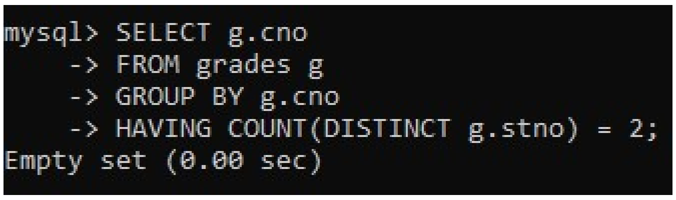


```
mysql> SELECT DISTINCT i1.name
    -> FROM instructors i1
    -> JOIN grades g ON i1.empno = g.empno
    -> JOIN advising a ON g.stno = a.stno
    -> JOIN instructors i2 ON a.empno = i2.empno
    -> WHERE i1.roomno = i2.roomno AND i1.empno <> i2.empno;
Empty set (0.00 sec)
```

7. **Find course numbers for courses that enroll exactly two students**

Code:

SELECT g.cno
FROM grades g
GROUP BY g.cno
HAVING COUNT(DISTINCT g.stno) = 2; **Output:**



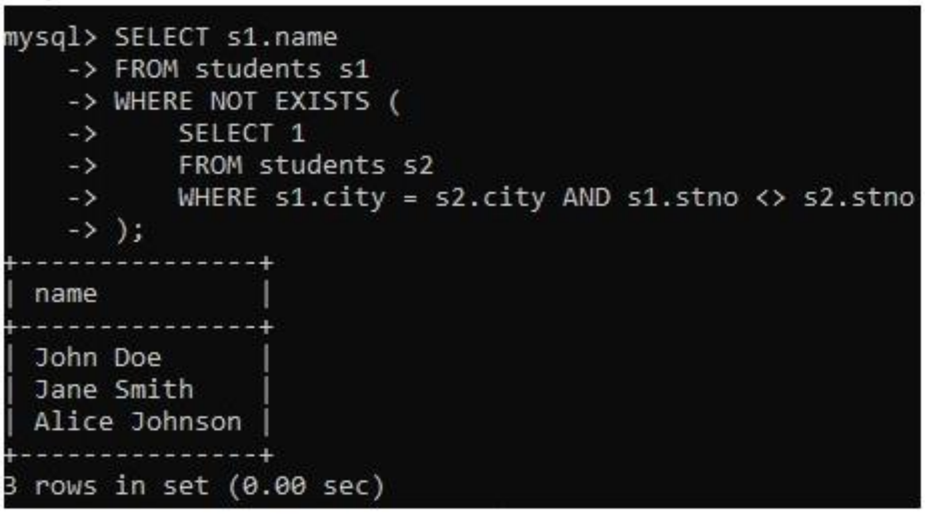
```
mysql> SELECT g.cno
    -> FROM grades g
    -> GROUP BY g.cno
    -> HAVING COUNT(DISTINCT g.stno) = 2;
Empty set (0.00 sec)
```

8. Find the names of all students for whom no other student lives in the same city.

Code:

```
SELECT s1.name
FROM students s1
WHERE NOT EXISTS (
  SELECT 1
  FROM students s2
  WHERE s1.city = s2.city AND s1.stno <> s2.stno
);
```

Output:



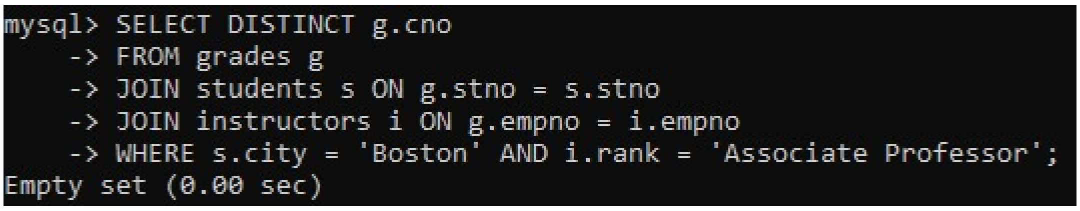
```
mysql> SELECT s1.name
    -> FROM students s1
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM students s2
    ->     WHERE s1.city = s2.city AND s1.stno <> s2.stno
    -> );
+---------------+
| name          |
+---------------+
| John Doe      |
| Jane Smith    |
| Alice Johnson |
+---------------+
3 rows in set (0.00 sec)
```

9. Find course numbers of courses taken by students who live in Boston and which are taught by an associate professor.

Code:

```
SELECT DISTINCT g.cno
FROM grades g
JOIN students s ON g.stno = s.stno
JOIN instructors i ON g.empno = i.empno
WHERE s.city = 'Boston' AND i.rank = 'Associate Professor';
```

Output:

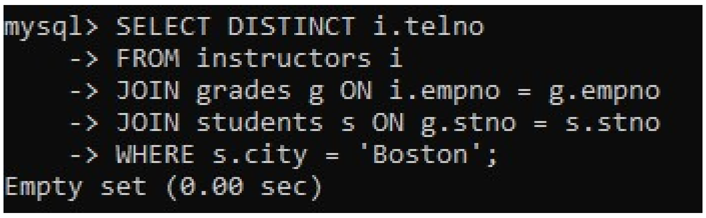


```
mysql> SELECT DISTINCT g.cno
    -> FROM grades g
    -> JOIN students s ON g.stno = s.stno
    -> JOIN instructors i ON g.empno = i.empno
    -> WHERE s.city = 'Boston' AND i.rank = 'Associate Professor';
Empty set (0.00 sec)
```

10. Find the telephone numbers of instructors who teach a course taken by any student who lives in Boston.

Code:

```
SELECT DISTINCT i.telno
FROM instructors i
JOIN grades g ON i.empno = g.empno
JOIN students s ON g.stno = s.stno
WHERE s.city = 'Boston'; Output:
```



```
mysql> SELECT DISTINCT i.telno
    -> FROM instructors i
    -> JOIN grades g ON i.empno = g.empno
    -> JOIN students s ON g.stno = s.stno
    -> WHERE s.city = 'Boston';
Empty set (0.00 sec)
```