MANVI JALAWADIYA
L_018

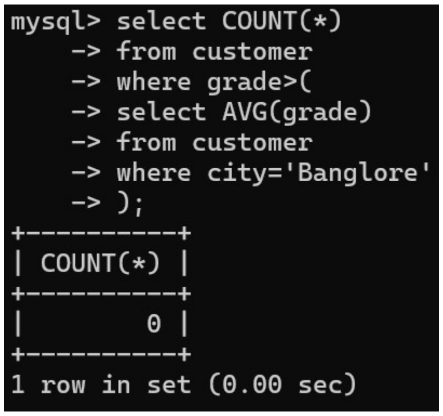
PRACTICAL 2

AIM: Subquery-join operations on Relational Schema.

1. Using Practical 1

1. Count the customers with grades above Bangalore's average.

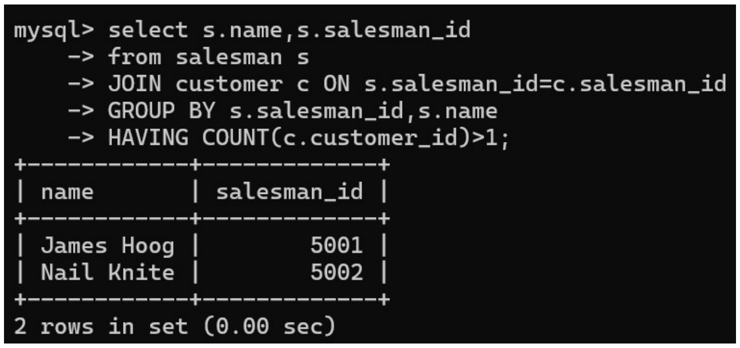
Select COUNT(*) from
customer where
grade>(select
AVG(grade) from
customer where
city='Banglore'
);



```
mysql> select COUNT(*)
    -> from customer
    -> where grade>(
    -> select AVG(grade)
    -> from customer
    -> where city='Banglore'
    -> );
+----------+
| COUNT(*) |
+----------+
|        0 |
+----------+
1 row in set (0.00 sec)
```

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

```
select s.name,s.salesman_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
    -> 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 |
+------------+-------------+
| James Hoog |        5001 |
| Nail Knite |        5002 |
+------------+-------------+
2 rows in set (0.00 sec)
```

3. **List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)** select s.salesman_id,s.name,'Has Customers' As customer_status from salesman s

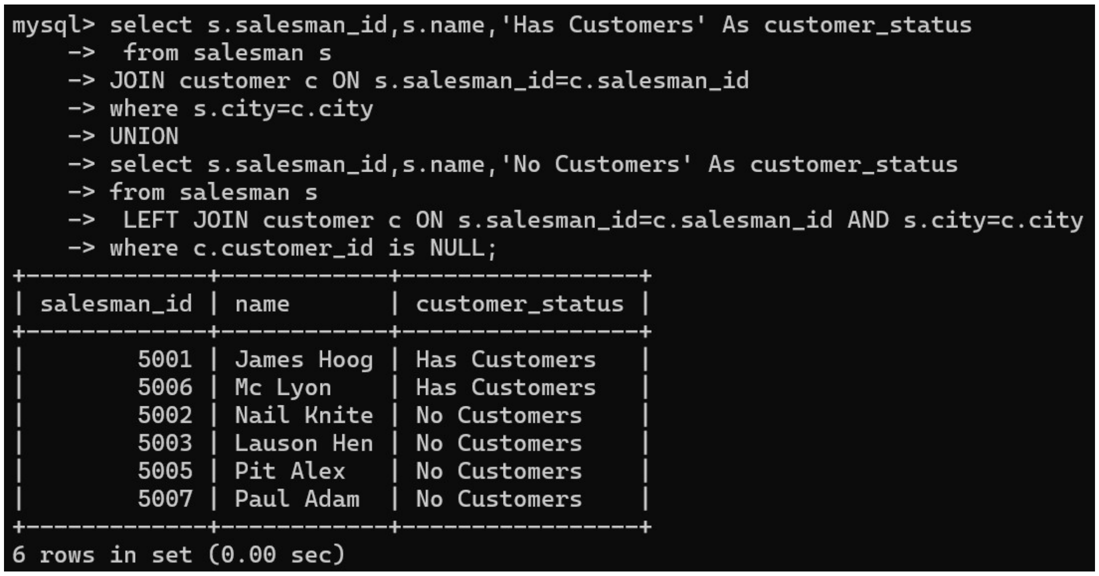
JOIN customer c ON s.salesman_id=c.salesman_id where

s.city=c.city UNION select s.salesman_id,s.name,'No

Customers' As customer_status from salesman s

LEFT JOIN customer c ON s.salesman_id=c.salesman_id AND s.city=c.city

where c.customer_id is NULL;



```
mysql> select s.salesman_id,s.name,'Has Customers' As customer_status
    ->  from salesman s
    -> JOIN customer c ON s.salesman_id=c.salesman_id
    -> where s.city=c.city
    -> UNION
    -> select s.salesman_id,s.name,'No Customers' As customer_status
    -> from salesman s
    ->  LEFT JOIN customer c ON s.salesman_id=c.salesman_id AND s.city=c.city
    -> where c.customer_id is NULL;
+-------------+------------+-----------------+
| salesman_id | name       | customer_status |
+-------------+------------+-----------------+
|        5001 | James Hoog | Has Customers   |
|        5006 | Mc Lyon    | Has Customers   |
|        5002 | Nail Knite | No Customers    |
|        5003 | Lauson Hen | No Customers    |
|        5005 | Pit Alex   | No Customers    |
|        5007 | Paul Adam  | No Customers    |
+-------------+------------+-----------------+
6 rows in set (0.00 sec)
```

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

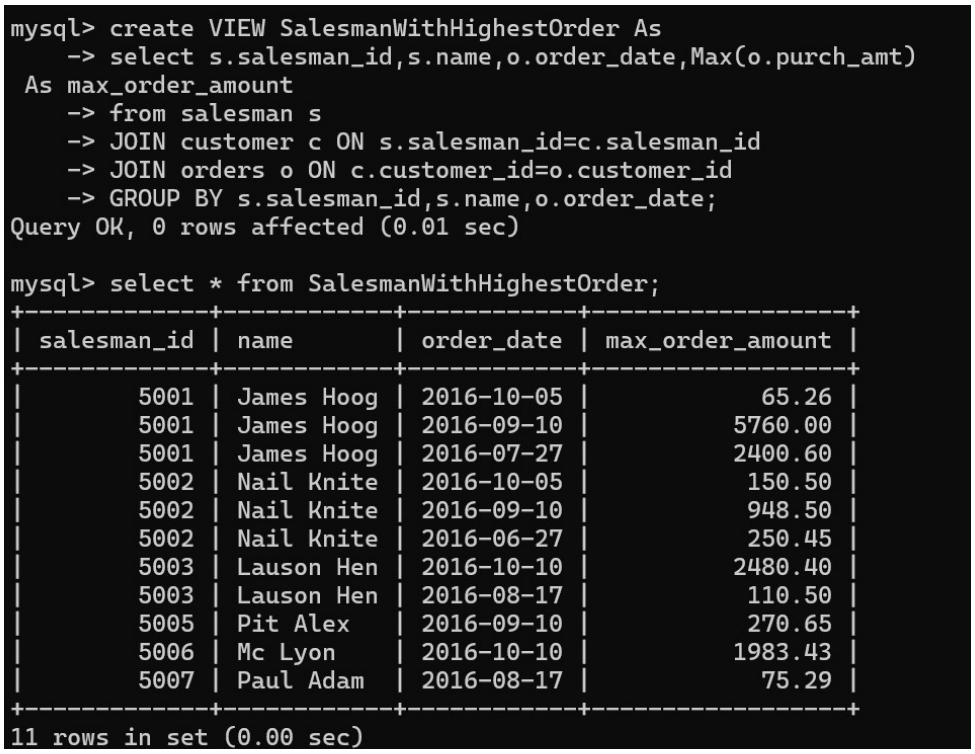
create VIEW SalesmanWithHighestOrder As select s.salesman_id,s.name,o.order_date,Max(o.purch_amt) As max_order_amount from salesman s

JOIN customer c ON s.salesman_id=c.salesman_id

JOIN orders o ON c.customer_id=o.customer_id

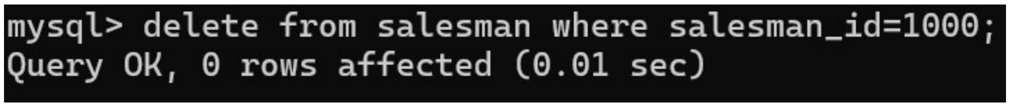
GROUP BY s.salesman_id,s.name,o.order_date;

select * from SalesmanWithHighestOrder;

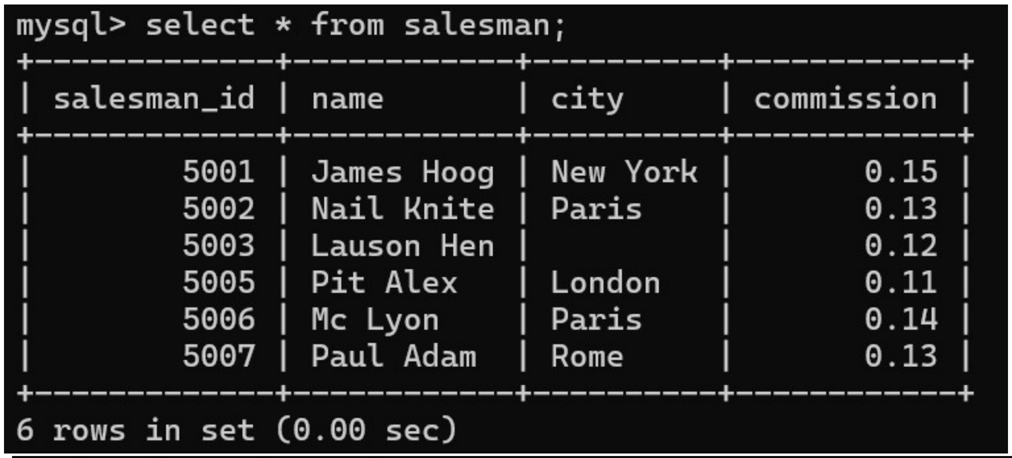


```
mysql> create VIEW SalesmanWithHighestOrder As
    -> select s.salesman_id,s.name,o.order_date,Max(o.purch_amt)
 As max_order_amount
    -> from salesman s
    -> JOIN customer c ON s.salesman_id=c.salesman_id
    -> JOIN orders o ON c.customer_id=o.customer_id
    -> GROUP BY s.salesman_id,s.name,o.order_date;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from SalesmanWithHighestOrder;
+-------------+------------+------------+------------------+
| salesman_id | name       | order_date | max_order_amount |
+-------------+------------+------------+------------------+
|        5001 | James Hoog | 2016-10-05 |            65.26 |
|        5001 | James Hoog | 2016-09-10 |          5760.00 |
|        5001 | James Hoog | 2016-07-27 |          2400.60 |
|        5002 | Nail Knite | 2016-10-05 |           150.50 |
|        5002 | Nail Knite | 2016-09-10 |           948.50 |
|        5002 | Nail Knite | 2016-06-27 |           250.45 |
|        5003 | Lauson Hen | 2016-10-10 |          2480.40 |
|        5003 | Lauson Hen | 2016-08-17 |           110.50 |
|        5005 | Pit Alex   | 2016-09-10 |           270.65 |
|        5006 | Mc Lyon    | 2016-10-10 |          1983.43 |
|        5007 | Paul Adam  | 2016-08-17 |            75.29 |
+-------------+------------+------------+------------------+
11 rows in set (0.00 sec)
```

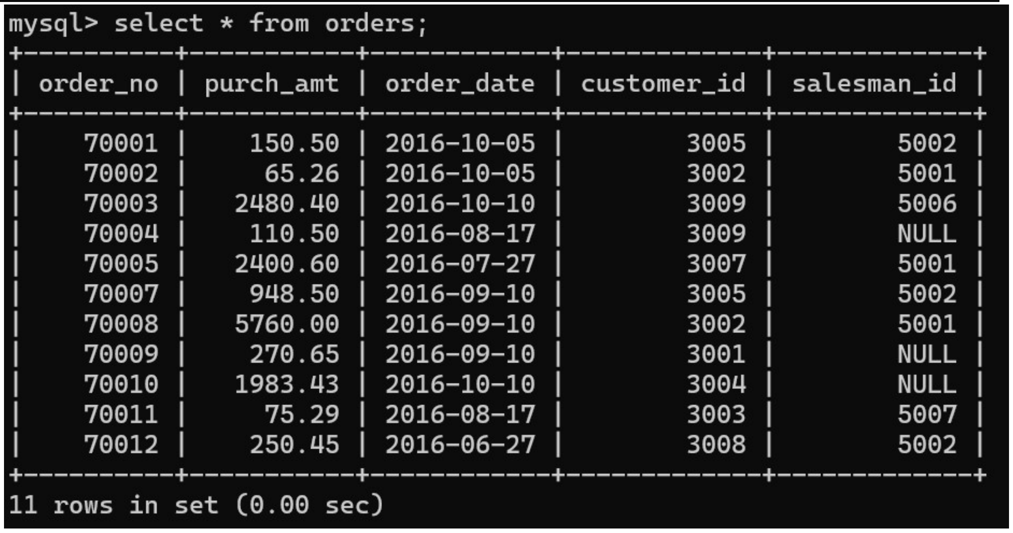
5. **Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.** delete from salesman where salesman_id=1000; select * from salesman; select * from orders;



```
mysql> delete from salesman where salesman_id=1000;
Query OK, 0 rows affected (0.01 sec)
```



```
mysql> select * from salesman;
+-------------+------------+----------+------------+
| salesman_id | name       | city     | commission |
+-------------+------------+----------+------------+
|        5001 | James Hoog | New York |       0.15 |
|        5002 | Nail Knite | Paris    |       0.13 |
|        5003 | Lauson Hen |          |       0.12 |
|        5005 | Pit Alex   | London   |       0.11 |
|        5006 | Mc Lyon    | Paris    |       0.14 |
|        5007 | Paul Adam  | Rome     |       0.13 |
+-------------+------------+----------+------------+
6 rows in set (0.00 sec)
```



```
mysql> select * from orders;
+----------+-----------+------------+-------------+-------------+
| order_no | purch_amt | order_date | customer_id | salesman_id |
+----------+-----------+------------+-------------+-------------+
|    70001 |    150.50 | 2016-10-05 |        3005 |        5002 |
|    70002 |     65.26 | 2016-10-05 |        3002 |        5001 |
|    70003 |   2480.40 | 2016-10-10 |        3009 |        5006 |
|    70004 |    110.50 | 2016-08-17 |        3009 |        NULL |
|    70005 |   2400.60 | 2016-07-27 |        3007 |        5001 |
|    70007 |    948.50 | 2016-09-10 |        3005 |        5002 |
|    70008 |   5760.00 | 2016-09-10 |        3002 |        5001 |
|    70009 |    270.65 | 2016-09-10 |        3001 |        NULL |
|    70010 |   1983.43 | 2016-10-10 |        3004 |        NULL |
|    70011 |     75.29 | 2016-08-17 |        3003 |        5007 |
|    70012 |    250.45 | 2016-06-27 |        3008 |        5002 |
+----------+-----------+------------+-------------+-------------+
11 rows in set (0.00 sec)
```

MANVI JALAWADIYA
L_018

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

Consider the schema for Movie Database:

ACTOR (Act_id, Act_Name, Act_Gender)

DIRECTOR (Dir_id, Dir_Name, Dir_Phone)

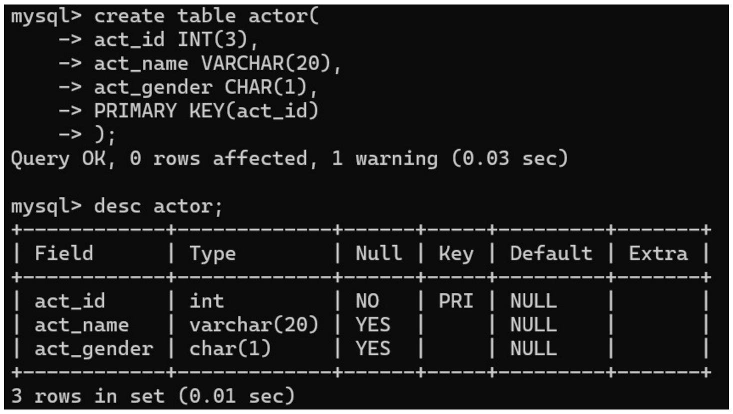
MOVIES (Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id)

MOVIE_CAST (Act_id, Mov_id, Role)

RATING (Mov_id, Rev_Stars)

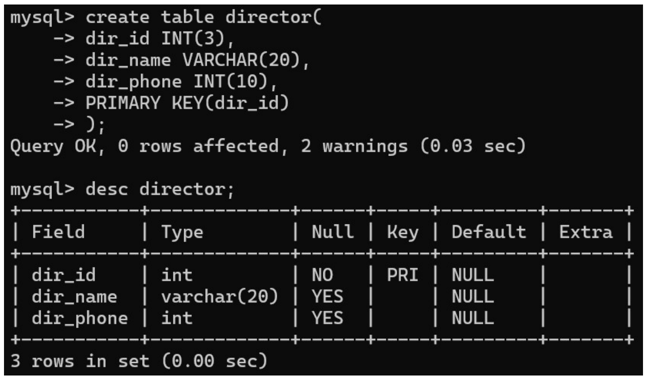
Create tables for actor, director, movies, movie_cast, rating.

```
create table actor(     act_id
INT(3),     act_name
VARCHAR(20),
act_gender CHAR(1),
   PRIMARY KEY(act_id)
   );
```



```
mysql> create table actor(
    -> act_id INT(3),
    -> act_name VARCHAR(20),
    -> act_gender CHAR(1),
    -> PRIMARY KEY(act_id)
    -> );
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> desc actor;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| act_id     | int         | NO   | PRI | NULL    |       |
| act_name   | varchar(20) | YES  |     | NULL    |       |
| act_gender | char(1)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

```
create table director(     dir_id
INT(3),                  dir_name
VARCHAR(20),
dir_phone INT(10),
   PRIMARY KEY(dir_id)
   );
```



```
mysql> create table director(
    -> dir_id INT(3),
    -> dir_name VARCHAR(20),
    -> dir_phone INT(10),
    -> PRIMARY KEY(dir_id)
    -> );
Query OK, 0 rows affected, 2 warnings (0.03 sec)

mysql> desc director;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| dir_id    | int         | NO   | PRI | NULL    |       |
| dir_name  | varchar(20) | YES  |     | NULL    |       |
| dir_phone | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

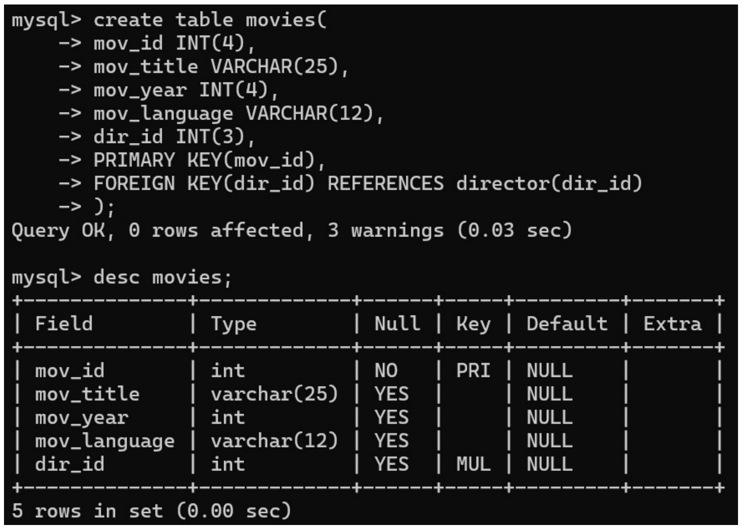
create table movies(mov_id INT(4), mov_title VARCHAR(25), mov_year INT(4), mov_language VARCHAR(12),

dir_id INT(3),

PRIMARY KEY(mov_id),

FOREIGN KEY(dir_id) REFERENCES director(dir_id)

);



```
mysql> create table movies(
    -> mov_id INT(4),
    -> mov_title VARCHAR(25),
    -> mov_year INT(4),
    -> mov_language VARCHAR(12),
    -> dir_id INT(3),
    -> PRIMARY KEY(mov_id),
    -> FOREIGN KEY(dir_id) REFERENCES director(dir_id)
    -> );
Query OK, 0 rows affected, 3 warnings (0.03 sec)

mysql> desc movies;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| mov_id       | int         | NO   | PRI | NULL    |       |
| mov_title    | varchar(25) | YES  |     | NULL    |       |
| mov_year     | int         | YES  |     | NULL    |       |
| mov_language | varchar(12) | YES  |     | NULL    |       |
| dir_id       | int         | YES  | MUL | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

create table movie_cast(

act_id INT(3),

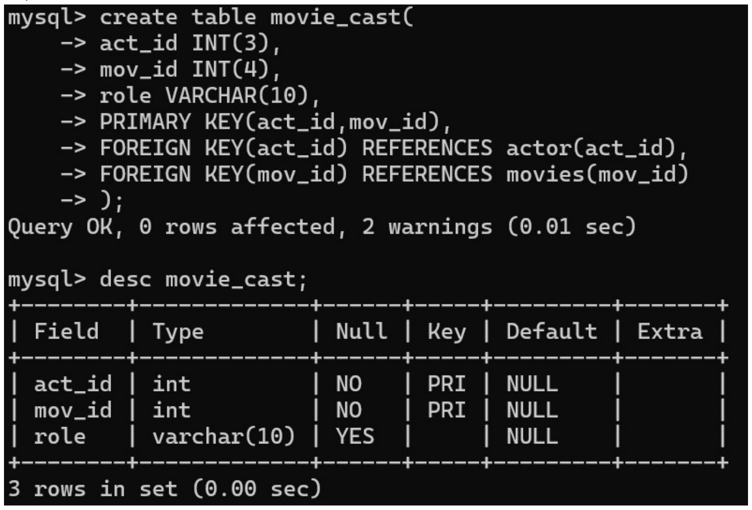
mov_id INT(4), role VARCHAR(10),

PRIMARY KEY(act_id,mov_id),

FOREIGN KEY(act_id) REFERENCES actor(act_id),

FOREIGN KEY(mov_id) REFERENCES movies(mov_id)

);



```
mysql> create table movie_cast(
    -> act_id INT(3),
    -> mov_id INT(4),
    -> role VARCHAR(10),
    -> PRIMARY KEY(act_id,mov_id),
    -> FOREIGN KEY(act_id) REFERENCES actor(act_id),
    -> FOREIGN KEY(mov_id) REFERENCES movies(mov_id)
    -> );
Query OK, 0 rows affected, 2 warnings (0.01 sec)

mysql> desc movie_cast;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| act_id | int         | NO   | PRI | NULL    |       |
| mov_id | int         | NO   | PRI | NULL    |       |
| role   | varchar(10) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

create table rating(

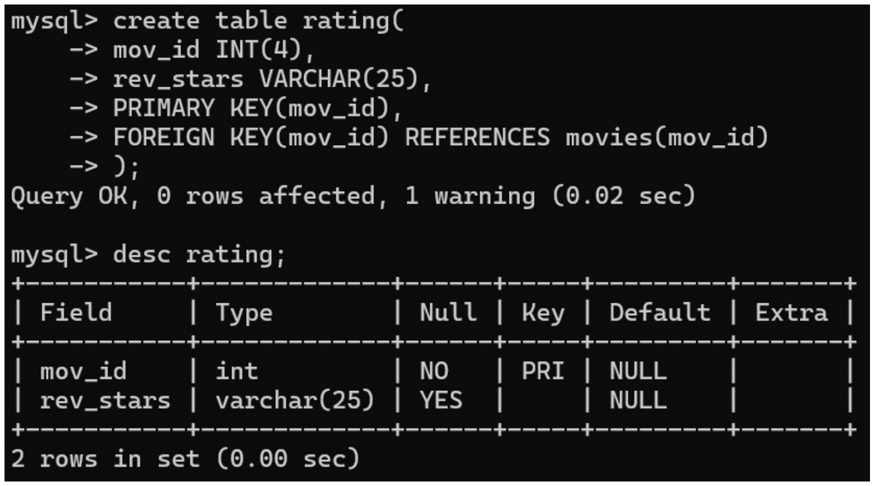
mov_id INT(4), rev_stars

VARCHAR(25),

PRIMARY KEY(mov_id),

FOREIGN KEY(mov_id) REFERENCES movies(mov_id)

);



```
mysql> create table rating(
    -> mov_id INT(4),
    -> rev_stars VARCHAR(25),
    -> PRIMARY KEY(mov_id),
    -> FOREIGN KEY(mov_id) REFERENCES movies(mov_id)
    -> );
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> desc rating;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| mov_id    | int         | NO   | PRI | NULL    |       |
| rev_stars | varchar(25) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

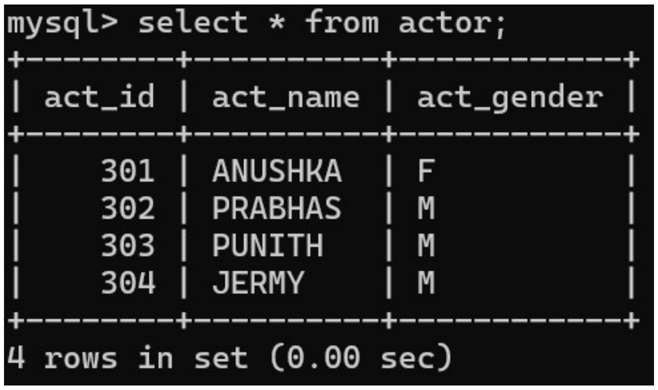
Insert values into tables.

insert into actor values(301,'ANUSHKA','F');

insert into actor values(302,'PRABHAS','M');

insert into actor values(303,'PUNITH','M');

insert into actor values(304,'JERMY','M');



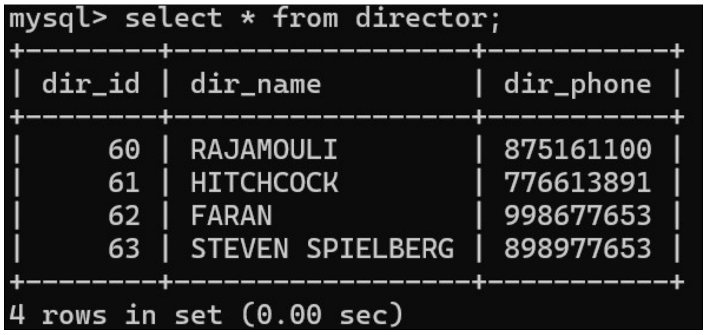
```
mysql> select * from actor;
+--------+----------+------------+
| act_id | act_name | act_gender |
+--------+----------+------------+
|    301 | ANUSHKA  | F          |
|    302 | PRABHAS  | M          |
|    303 | PUNITH   | M          |
|    304 | JERMY    | M          |
+--------+----------+------------+
4 rows in set (0.00 sec)
```

insert into director values(60,'RAJAMOULI', 875161100); insert

into director values(61,'HITCHCOCK', 776613891); insert into

director values(62,'FARAN', 998677653); insert into director

values(63,'STEVEN SPIELBERG', 898977653);



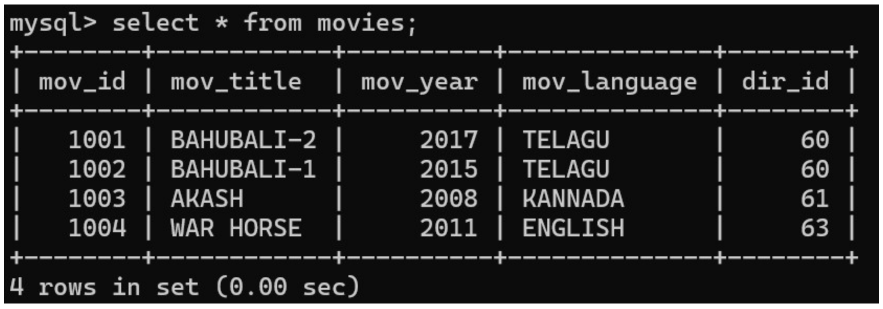
```
mysql> select * from director;
+--------+------------------+-----------+
| dir_id | dir_name         | dir_phone |
+--------+------------------+-----------+
|     60 | RAJAMOULI        | 875161100 |
|     61 | HITCHCOCK        | 776613891 |
|     62 | FARAN            | 998677653 |
|     63 | STEVEN SPIELBERG | 898977653 |
+--------+------------------+-----------+
4 rows in set (0.00 sec)
```

insert into movies values(1001,'BAHUBALI-2', 2017, 'TELAGU', 60);

insert into movies values(1002,'BAHUBALI-1', 2015, 'TELAGU', 60);

insert into movies values(1003,'AKASH', 2008, 'KANNADA', 61);

insert into movies values(1004,'WAR HORSE', 2011, 'ENGLISH', 63);



```
mysql> select * from movies;
+--------+------------+----------+--------------+--------+
| mov_id | mov_title  | mov_year | mov_language | dir_id |
+--------+------------+----------+--------------+--------+
|   1001 | BAHUBALI-2 |     2017 | TELAGU       |     60 |
|   1002 | BAHUBALI-1 |     2015 | TELAGU       |     60 |
|   1003 | AKASH      |     2008 | KANNADA      |     61 |
|   1004 | WAR HORSE  |     2011 | ENGLISH      |     63 |
+--------+------------+----------+--------------+--------+
4 rows in set (0.00 sec)
```

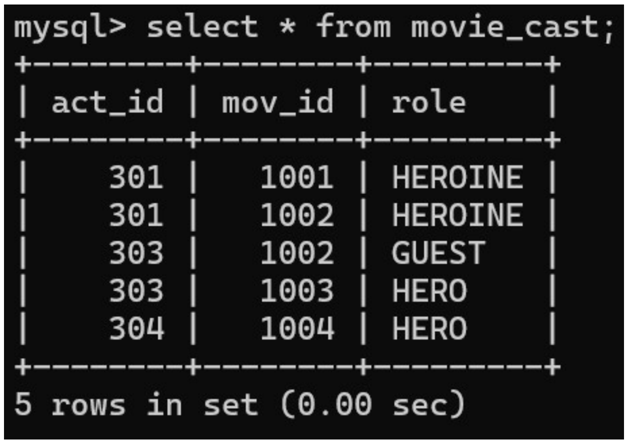
insert into movie_cast values(301, 1002, 'HEROINE');

insert into movie_cast values(301, 1001, 'HEROINE');

insert into movie_cast values(303, 1003, 'HERO');

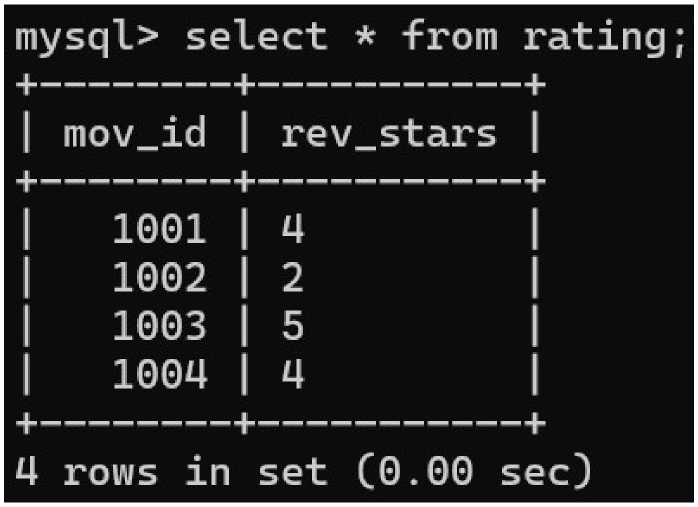
insert into movie_cast values(303, 1002, 'GUEST');

insert into movie_cast values(304, 1004, 'HERO');



```
mysql> select * from movie_cast;
+--------+--------+---------+
| act_id | mov_id | role    |
+--------+--------+---------+
|    301 |   1001 | HEROINE |
|    301 |   1002 | HEROINE |
|    303 |   1002 | GUEST   |
|    303 |   1003 | HERO    |
|    304 |   1004 | HERO    |
+--------+--------+---------+
5 rows in set (0.00 sec)
```

insert into rating values(1001,4); insert
into rating values(1002,2); insert into
rating values(1003,5); insert into
rating values(1004,4);



```
mysql> select * from rating;
+--------+-----------+
| mov_id | rev_stars |
+--------+-----------+
|   1001 | 4         |
|   1002 | 2         |
|   1003 | 5         |
|   1004 | 4         |
+--------+-----------+
4 rows in set (0.00 sec)
```

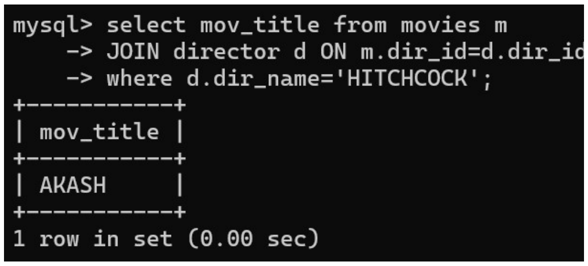
Write SQL queries to

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

select mov_title from movies m JOIN

director d ON m.dir_id=d.dir_id

where d.dir_name='HITCHCOCK';



```
mysql> select mov_title from movies m
    -> JOIN director d ON m.dir_id=d.dir_ic
    -> where d.dir_name='HITCHCOCK';
+-----------+
| mov_title |
+-----------+
| AKASH     |
+-----------+
1 row in set (0.00 sec)
```

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

   DISTINCT m.mov_title from movies m

   JOIN movie_cast mc ON

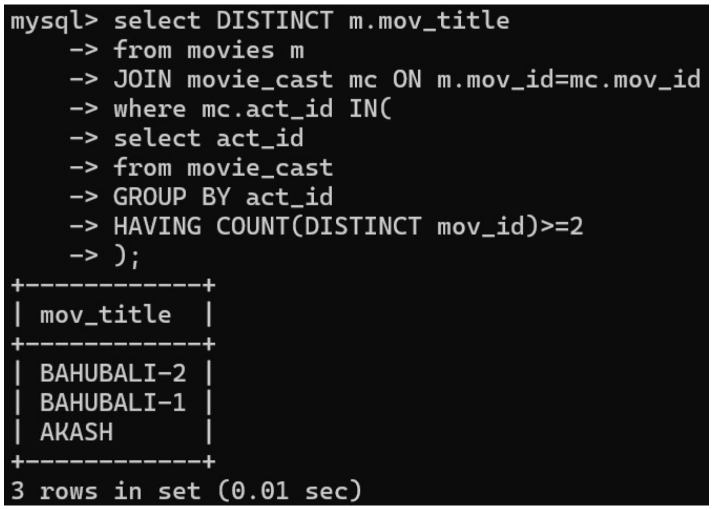
m.mov_id=mc.mov_id where mc.act_id IN(

select act_id from movie_cast

   GROUP BY act_id

   HAVING COUNT(DISTINCT mov_id)>=2

   );



```
mysql> select DISTINCT m.mov_title
    -> from movies m
    -> JOIN movie_cast mc ON m.mov_id=mc.mov_id
    -> where mc.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 rows in set (0.01 sec)
```

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

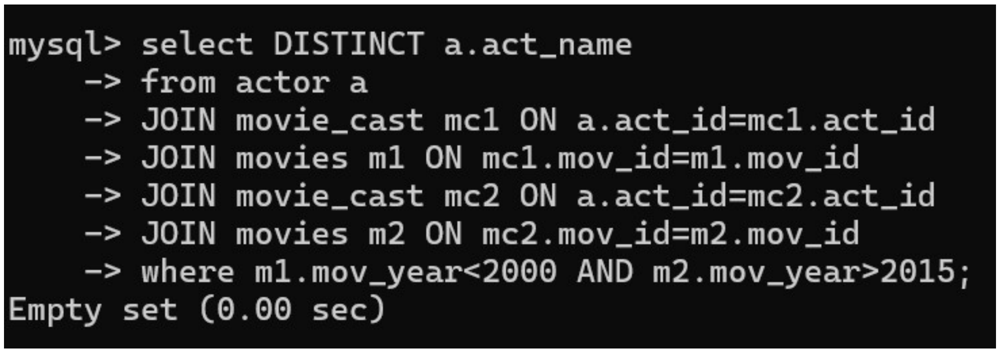
JOIN movie_cast mc1 ON a.act_id=mc1.act_id

JOIN movies m1 ON mc1.mov_id=m1.mov_id

JOIN movie_cast mc2 ON a.act_id=mc2.act_id

JOIN movies m2 ON mc2.mov_id=m2.mov_id

where m1.mov_year<2000 AND m2.mov_year>2015;



```
mysql> select DISTINCT a.act_name
    -> from actor a
    -> JOIN movie_cast mc1 ON a.act_id=mc1.act_id
    -> JOIN movies m1 ON mc1.mov_id=m1.mov_id
    -> JOIN movie_cast mc2 ON a.act_id=mc2.act_id
    -> JOIN movies m2 ON mc2.mov_id=m2.mov_id
    -> where m1.mov_year<2000 AND m2.mov_year>2015;
Empty set (0.00 sec)
```

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,r.rev_stars,(select

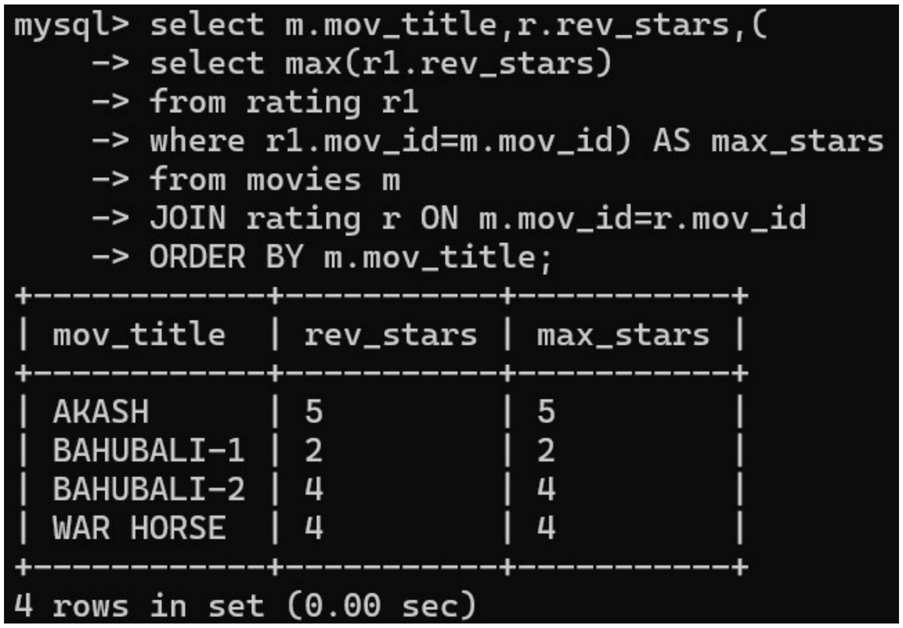
max(r1.rev_stars)   from rating r1   where

r1.mov_id=m.mov_id) AS max_stars   from

movies m

JOIN rating r ON m.mov_id=r.mov_id

ORDER BY m.mov_title;

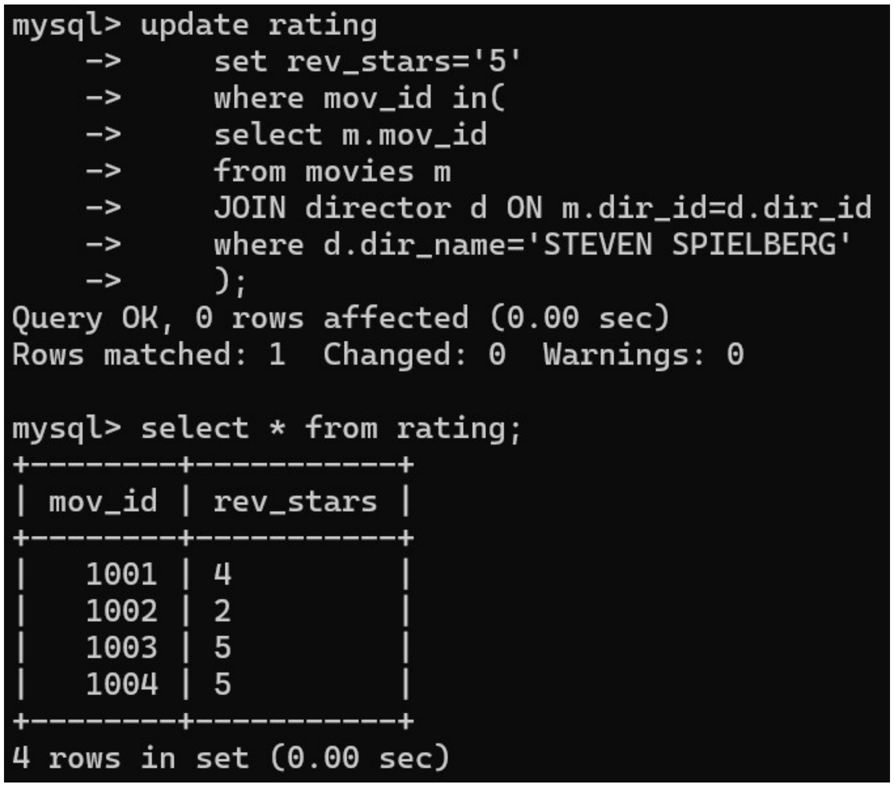


```
mysql> select m.mov_title,r.rev_stars,(
    -> select max(r1.rev_stars)
    -> from rating r1
    -> where r1.mov_id=m.mov_id) AS max_stars
    -> from movies m
    -> JOIN rating r ON m.mov_id=r.mov_id
    -> ORDER BY m.mov_title;
+------------+-----------+-----------+
| mov_title  | rev_stars | max_stars |
+------------+-----------+-----------+
| AKASH      | 5         | 5         |
| BAHUBALI-1 | 2         | 2         |
| BAHUBALI-2 | 4         | 4         |
| WAR HORSE  | 4         | 4         |
+------------+-----------+-----------+
4 rows in set (0.00 sec)
```

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

```
update rating     set

rev_stars='5'

where mov_id in(

select m.mov_id

from movies m

   JOIN director d ON m.dir_id=d.dir_id

where d.dir_name='STEVEN SPIELBERG'

   );
```

```
select * from rating;
```

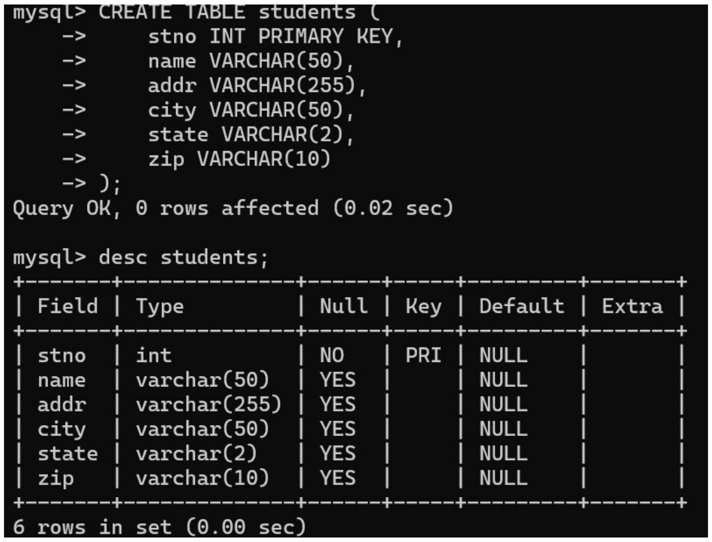


```
mysql> update rating
    ->     set rev_stars='5'
    ->     where mov_id in(
    ->     select m.mov_id
    ->     from movies m
    ->     JOIN director d ON m.dir_id=d.dir_id
    ->     where d.dir_name='STEVEN SPIELBERG'
    ->     );
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> select * from rating;
+--------+-----------+
| mov_id | rev_stars |
+--------+-----------+
|   1001 | 4         |
|   1002 | 2         |
|   1003 | 5         |
|   1004 | 5         |
+--------+-----------+
4 rows in set (0.00 sec)
```

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

Create tables.

```
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 (0.02 sec)

mysql> desc students;
+-------+--------------+------+-----+---------+-------+
| Field | Type         | Null | Key | Default | Extra |
+-------+--------------+------+-----+---------+-------+
| stno  | int          | NO   | PRI | NULL    |       |
| name  | varchar(50)  | YES  |     | NULL    |       |
| addr  | varchar(255) | YES  |     | NULL    |       |
| city  | varchar(50)  | YES  |     | NULL    |       |
| state | varchar(2)   | YES  |     | NULL    |       |
| zip   | varchar(10)  | YES  |     | NULL    |       |
+-------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

CREATE TABLE INSTRUCTORS (

empno INT PRIMARY KEY,

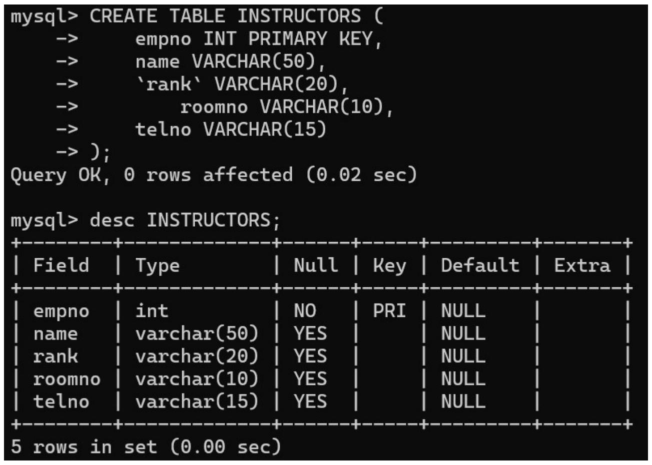
name VARCHAR(50), `rank`

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 (0.02 sec)

mysql> desc INSTRUCTORS;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| empno  | int         | NO   | PRI | NULL    |       |
| name   | varchar(50) | YES  |     | NULL    |       |
| rank   | varchar(20) | YES  |     | NULL    |       |
| roomno | varchar(10) | YES  |     | NULL    |       |
| telno  | varchar(15) | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

CREATE TABLE COURSES (

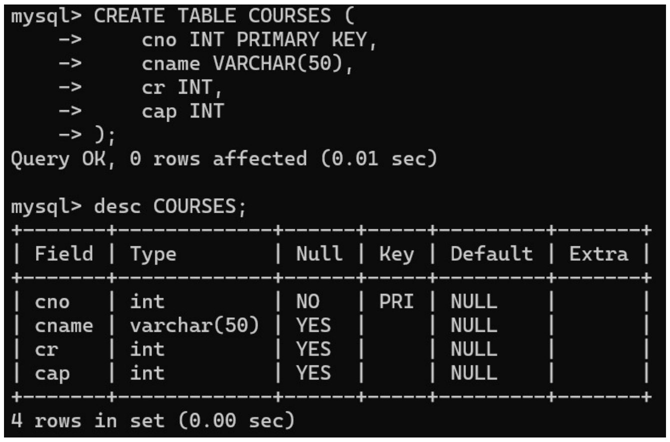
cno INT PRIMARY KEY,

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)

mysql> desc COURSES;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| cno   | int         | NO   | PRI | NULL    |       |
| cname | varchar(50) | YES  |     | NULL    |       |
| cr    | int         | YES  |     | NULL    |       |
| cap   | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

CREATE TABLE GRADES (

stno INT,

empno INT,

cno INT,

sem VARCHAR(10),

year INT,

grade INT,

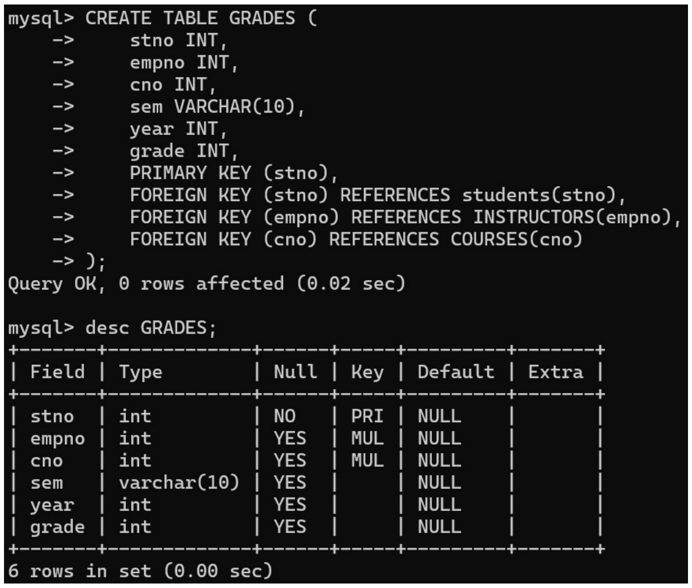
PRIMARY KEY (stno),

FOREIGN KEY (stno) REFERENCES students(stno),

FOREIGN KEY (empno) REFERENCES INSTRUCTORS(empno),

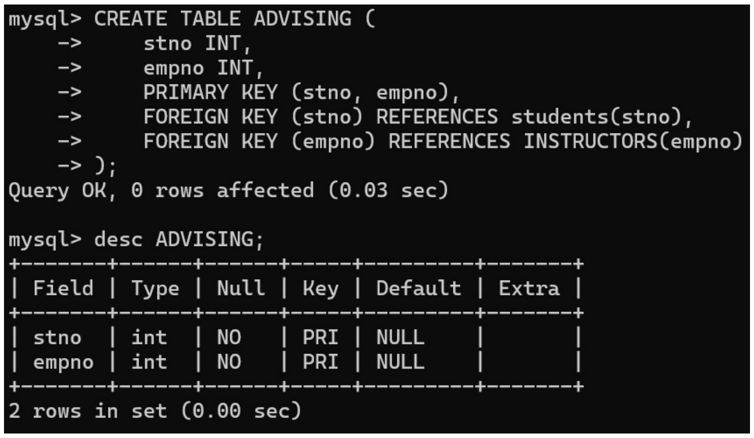
FOREIGN KEY (cno) REFERENCES COURSES(cno)

);



```
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 (0.02 sec)

mysql> desc GRADES;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| stno  | int         | NO   | PRI | NULL    |       |
| empno | int         | YES  | MUL | NULL    |       |
| cno   | int         | YES  | MUL | NULL    |       |
| sem   | varchar(10) | YES  |     | NULL    |       |
| year  | int         | YES  |     | NULL    |       |
| grade | int         | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
6 rows in set (0.00 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 (0.03 sec)

mysql> desc ADVISING;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| stno  | int  | NO   | PRI | NULL    |       |
| empno | int  | NO   | PRI | NULL    |       |
+-------+------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Insert values into tables.

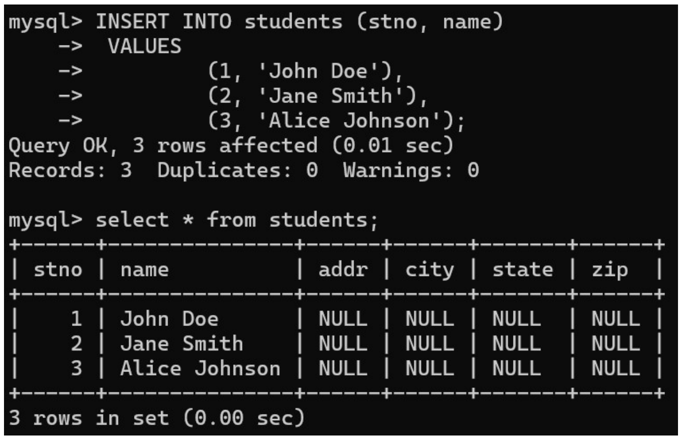
INSERT INTO students (stno, name)

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

mysql> select * from students;
+------+---------------+------+------+-------+------+
| stno | name          | addr | city | state | zip  |
+------+---------------+------+------+-------+------+
|    1 | John Doe      | NULL | NULL | NULL  | NULL |
|    2 | Jane Smith    | NULL | NULL | NULL  | NULL |
|    3 | Alice Johnson | NULL | NULL | NULL  | NULL |
+------+---------------+------+------+-------+------+
3 rows in set (0.00 sec)
```

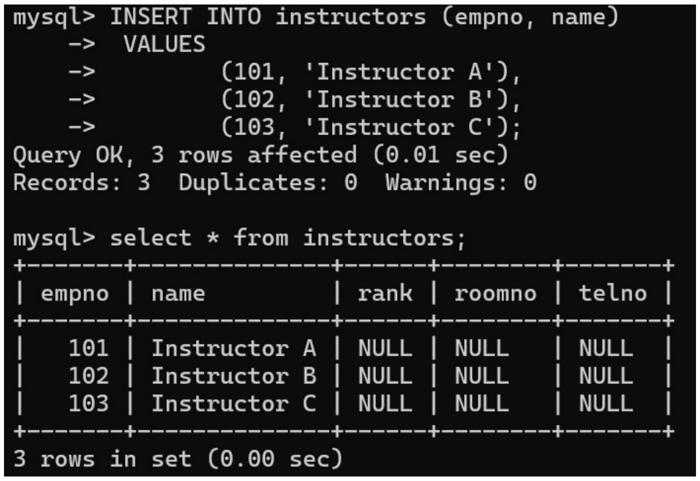
INSERT INTO instructors (empno, name)

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 (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from instructors;
+-------+--------------+------+--------+-------+
| empno | name         | rank | roomno | telno |
+-------+--------------+------+--------+-------+
|   101 | Instructor A | NULL | NULL   | NULL  |
|   102 | Instructor B | NULL | NULL   | NULL  |
|   103 | Instructor C | NULL | NULL   | NULL  |
+-------+--------------+------+--------+-------+
3 rows in set (0.00 sec)
```

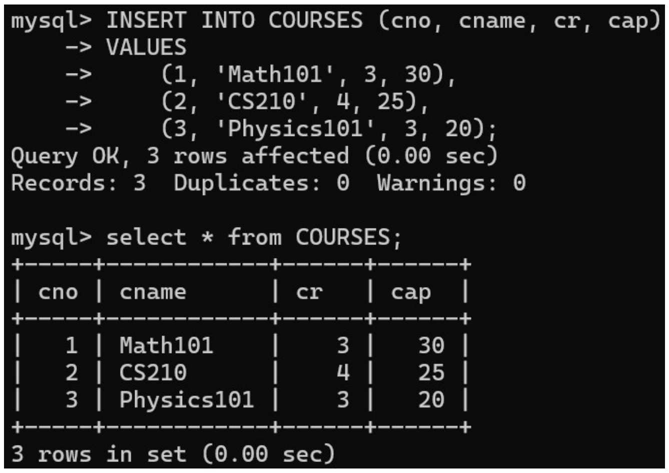
INSERT INTO COURSES (cno, cname, cr, cap)

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 (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from COURSES;
+-----+------------+------+------+
| cno | cname      | cr   | cap  |
+-----+------------+------+------+
|   1 | Math101    |    3 |   30 |
|   2 | CS210      |    4 |   25 |
|   3 | Physics101 |    3 |   20 |
+-----+------------+------+------+
3 rows in set (0.00 sec)
```

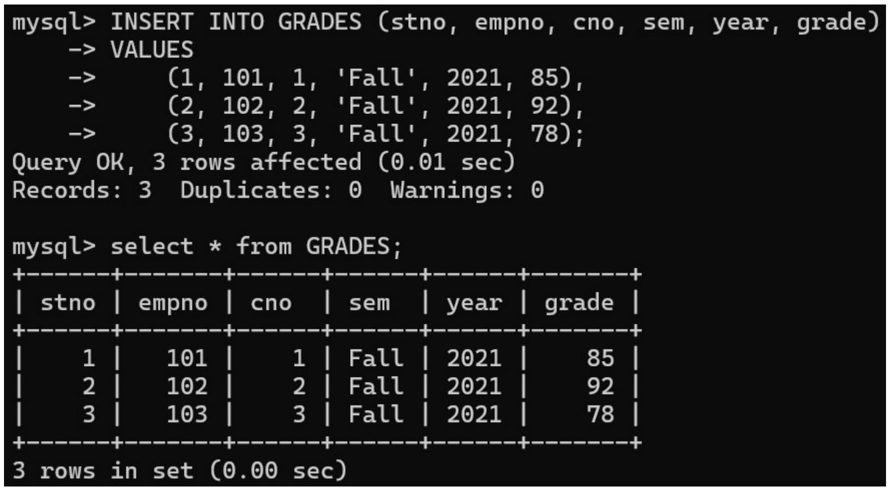
INSERT INTO GRADES (stno, empno, cno, sem, year, grade)

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 (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from GRADES;
+------+-------+------+------+------+-------+
| stno | empno | cno  | sem  | year | grade |
+------+-------+------+------+------+-------+
|    1 |   101 |    1 | Fall | 2021 |    85 |
|    2 |   102 |    2 | Fall | 2021 |    92 |
|    3 |   103 |    3 | Fall | 2021 |    78 |
+------+-------+------+------+------+-------+
3 rows in set (0.00 sec)
```

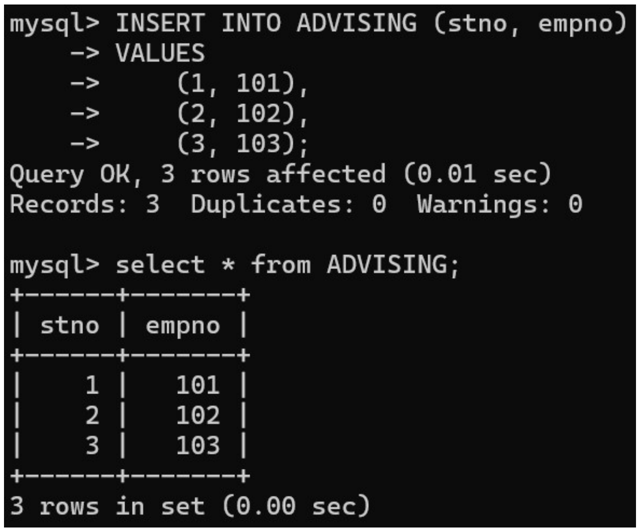
INSERT INTO ADVISING (stno, empno)

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 (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from ADVISING;
+------+-------+
| stno | empno |
+------+-------+
|    1 |   101 |
|    2 |   102 |
|    3 |   103 |
+------+-------+
3 rows in set (0.00 sec)
```

MANVI JALAWADIYA
L_018

For even roll numbers (any 10)

1. **Find the names of students who took only four-credit courses.** select s.name

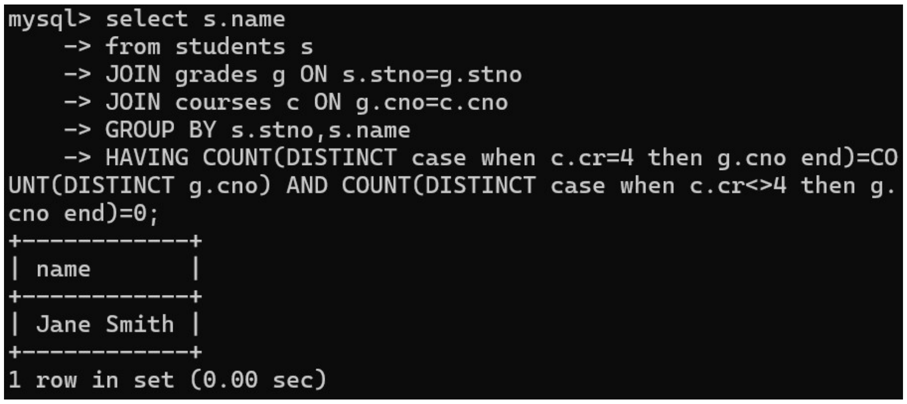
   from students s

   JOIN grades g ON s.stno=g.stno

   JOIN courses c ON g.cno=c.cno

   GROUP BY s.stno,s.name

   HAVING COUNT(DISTINCT case when c.cr=4 then g.cno end)=COUNT(DISTINCT g.cno) AND COUNT(DISTINCT case when c.cr<>4 then g.cno end)=0;



```
mysql> select s.name
    -> from students s
    -> JOIN grades g ON s.stno=g.stno
    -> JOIN courses c ON g.cno=c.cno
    -> GROUP BY s.stno,s.name
    -> HAVING COUNT(DISTINCT case when c.cr=4 then g.cno end)=CO
UNT(DISTINCT g.cno) AND COUNT(DISTINCT case when c.cr<>4 then g.
cno end)=0;
+------------+
| name       |
+------------+
| Jane Smith |
+------------+
1 row in set (0.00 sec)
```

2. **Find the names of students who took no four-credit courses.** select s.name

   from students s   where NOT EXISTS(

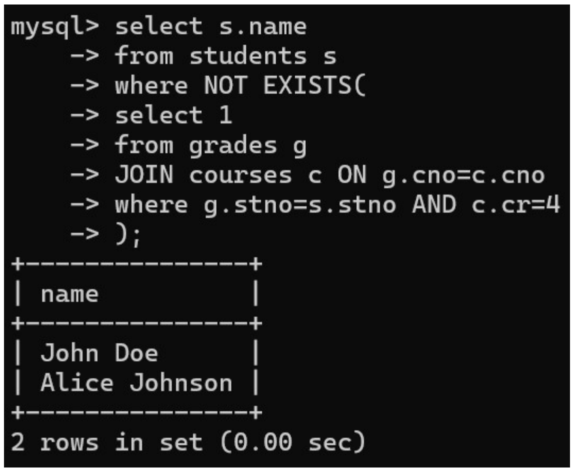
   select 1

   from grades g

   JOIN courses c ON g.cno=c.cno

   where g.stno=s.stno AND c.cr=4

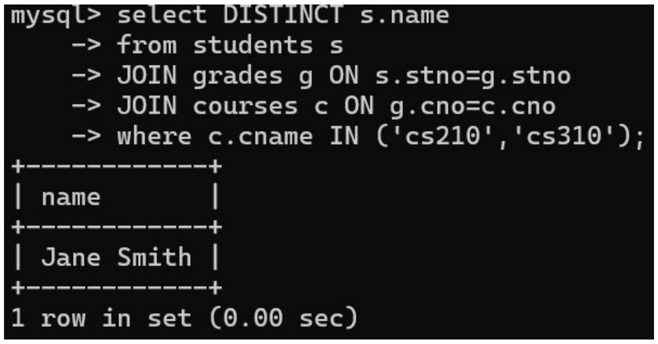
   );



```
mysql> select s.name
    -> from students s
    -> where NOT EXISTS(
    -> select 1
    -> from grades g
    -> JOIN courses c ON g.cno=c.cno
    -> where g.stno=s.stno AND c.cr=4
    -> );
+---------------+
| name          |
+---------------+
| John Doe      |
| Alice Johnson |
+---------------+
2 rows in set (0.00 sec)
```

3. **Find the names of students who took cs210 or cs310.**

```
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.cname IN ('cs210','cs310');
```



```
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.cname IN ('cs210','cs310');
+------------+
| name       |
+------------+
| Jane Smith |
+------------+
1 row in set (0.00 sec)
```

4. **Find names of all students who have a cs210 grade higher than the highest grade given in cs310 and did not take any course with Prof. Evans.**

```
SELECT DISTINCT s.name
   FROM students s
   JOIN grades g1 ON s.stno = g1.stno
```

JOIN courses c1 ON g1.cno = c1.cno

WHERE c1.cname = 'cs210' AND g1.grade > (

SELECT MAX(g2.grade)

FROM grades g2

JOIN courses c2 ON g2.cno = c2.cno

WHERE c2.cname = 'cs310'

)

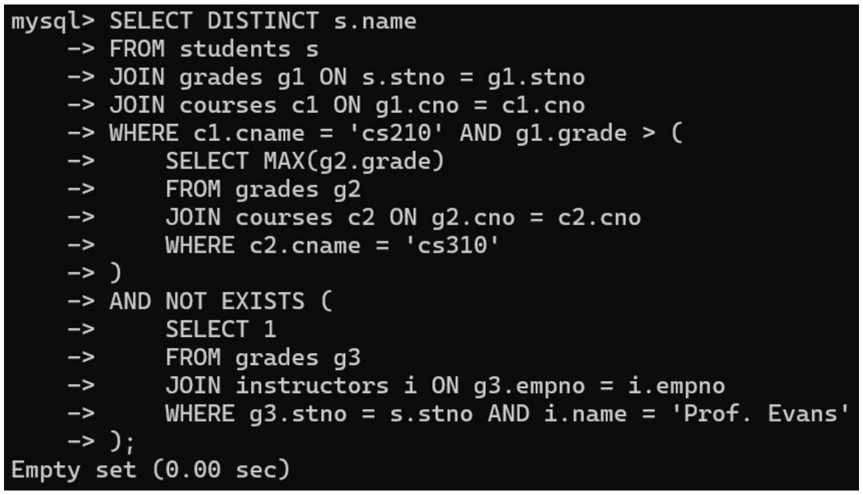
AND NOT EXISTS (

SELECT 1

FROM grades g3

JOIN instructors i ON g3.empno = i.empno

WHERE g3.stno = s.stno AND i.name = 'Prof. Evans'



```
mysql> SELECT DISTINCT s.name
    -> FROM students s
    -> JOIN grades g1 ON s.stno = g1.stno
    -> JOIN courses c1 ON g1.cno = c1.cno
    -> WHERE c1.cname = 'cs210' AND g1.grade > (
    ->     SELECT MAX(g2.grade)
    ->     FROM grades g2
    ->     JOIN courses c2 ON g2.cno = c2.cno
    ->     WHERE c2.cname = 'cs310'
    -> )
    -> AND NOT EXISTS (
    ->     SELECT 1
    ->     FROM grades g3
    ->     JOIN instructors i ON g3.empno = i.empno
    ->     WHERE g3.stno = s.stno AND i.name = 'Prof. Evans'
    -> );
Empty set (0.00 sec)
```

5. **Find course numbers for courses that enrol at least two students; solve the same query for courses that enroll at least three students.**

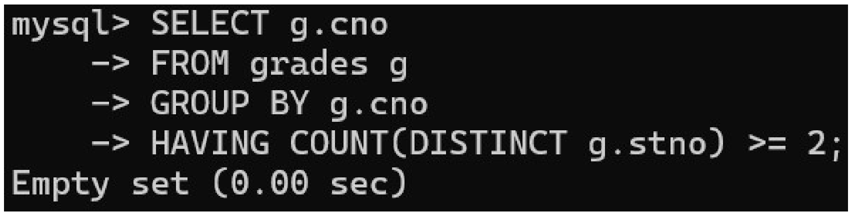
For courses with at least 2 students.

SELECT g.cno

FROM grades g

GROUP BY g.cno

HAVING COUNT(DISTINCT g.stno) >= 2;



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

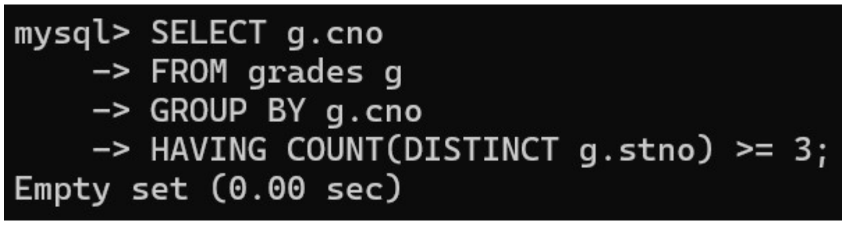
For courses with at least 3 students.

SELECT g.cno

-> FROM grades g

-> GROUP BY g.cno

-> HAVING COUNT(DISTINCT g.stno) >= 3;



```
mysql> SELECT g.cno
    -> FROM grades g
    -> GROUP BY g.cno
    -> HAVING COUNT(DISTINCT g.stno) >= 3;
Empty set (0.00 sec)
```

6. Find the names of students who obtained the highest grade in cs210.

select s.name

from students s

JOIN grades g ON s.stno=g.stno

JOIN courses c ON g.cno=c.cno where

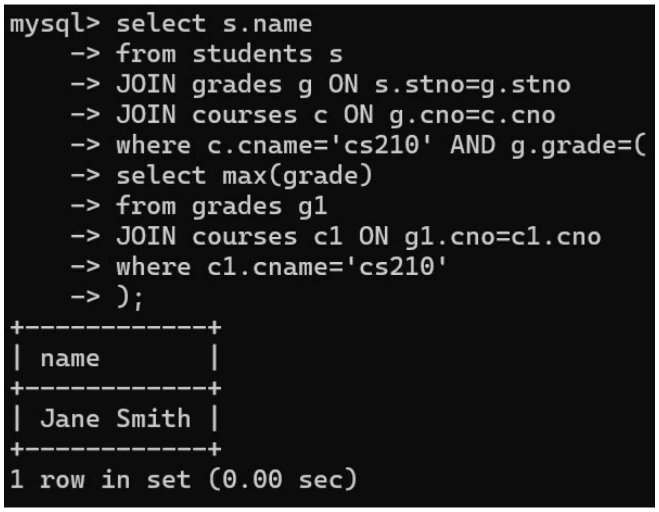
c.cname='cs210' AND g.grade=(select

max(grade) from grades g1

JOIN courses c1 ON g1.cno=c1.cno

where c1.cname='cs210'

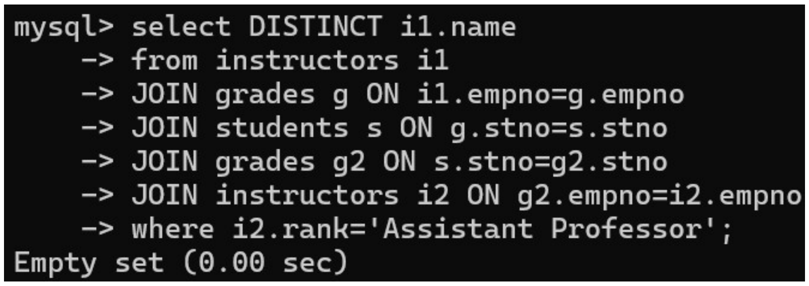
);



```
mysql> select s.name
    -> from students s
    -> JOIN grades g ON s.stno=g.stno
    -> JOIN courses c ON g.cno=c.cno
    -> where c.cname='cs210' AND g.grade=(
    -> select max(grade)
    -> from grades g1
    -> JOIN courses c1 ON g1.cno=c1.cno
    -> where c1.cname='cs210'
    -> );
+------------+
| name       |
+------------+
| Jane Smith |
+------------+
1 row in set (0.00 sec)
```

7. **Find the names of instructors who teach courses attended by students who took a course with an instructor who is an assistant professor.**

```
select DISTINCT i1.name
   from instructors i1
   JOIN grades g ON i1.empno=g.empno
   JOIN students s ON g.stno=s.stno
   JOIN grades g2 ON s.stno=g2.stno
   JOIN instructors i2 ON g2.empno=i2.empno
where i2.rank='Assistant Professor';
```

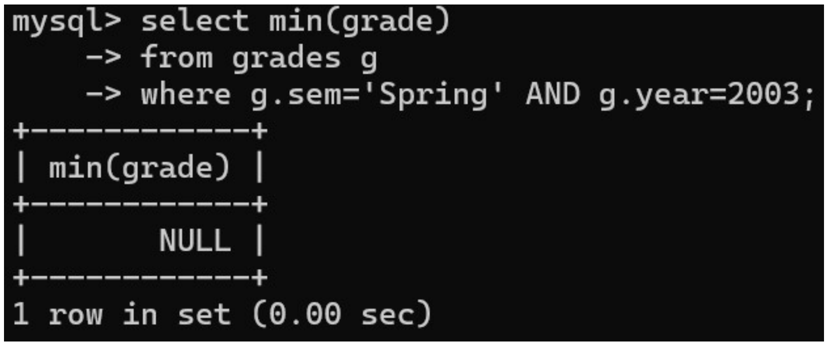


```
mysql> select DISTINCT i1.name
    -> from instructors i1
    -> JOIN grades g ON i1.empno=g.empno
    -> JOIN students s ON g.stno=s.stno
    -> JOIN grades g2 ON s.stno=g2.stno
    -> JOIN instructors i2 ON g2.empno=i2.empno
    -> where i2.rank='Assistant Professor';
Empty set (0.00 sec)
```

8. **Find the lowest grade of a student who took a course during the spring of 2003.**

select min(grade) from grades g where

g.sem='Spring' AND g.year=2003;



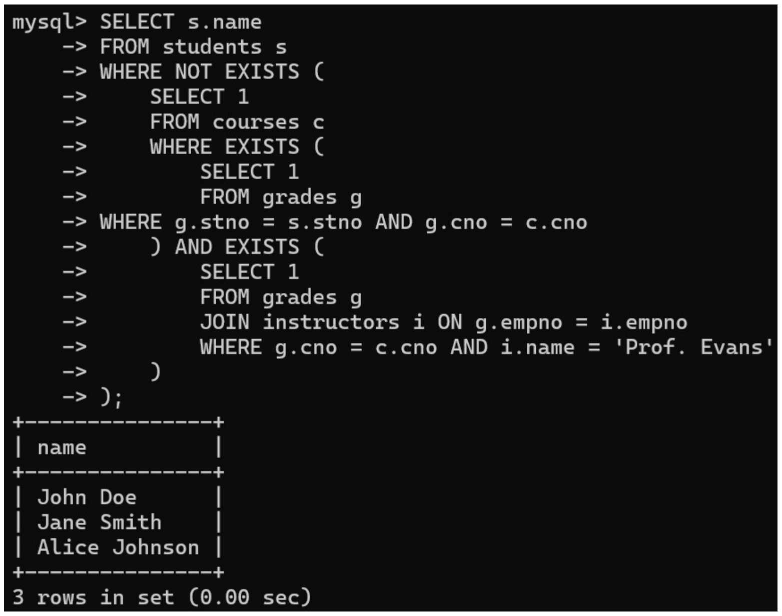
```
mysql> select min(grade)
    -> from grades g
    -> where g.sem='Spring' AND g.year=2003;
+------------+
| min(grade) |
+------------+
|       NULL |
+------------+
1 row in set (0.00 sec)
```

9. **Find the names for students such that if prof. Evans teaches a course, then the student takes that course (although not necessarily with prof. Evans).**

```
SELECT s.name
  FROM students s
  WHERE NOT EXISTS (
    SELECT 1
    FROM courses c
    WHERE EXISTS (
      SELECT 1
      FROM grades g
      WHERE g.stno = s.stno AND g.cno = c.cno
    ) AND EXISTS (
      SELECT 1
      FROM grades g
      JOIN instructors i ON g.empno = i.empno
      WHERE g.cno = c.cno AND i.name = 'Prof. Evans'
```

)

    );



```
mysql> SELECT s.name
    -> FROM students s
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM courses c
    ->     WHERE EXISTS (
    ->         SELECT 1
    ->         FROM grades g
    -> WHERE g.stno = s.stno AND g.cno = c.cno
    ->     ) AND EXISTS (
    ->         SELECT 1
    ->         FROM grades g
    ->         JOIN instructors i ON g.empno = i.empno
    ->         WHERE g.cno = c.cno AND i.name = 'Prof. Evans'
    ->     )
    -> );
+---------------+
| name          |
+---------------+
| John Doe      |
| Jane Smith    |
| Alice Johnson |
+---------------+
3 rows in set (0.00 sec)
```

10. Find the names of students whose advisor did not teach them any course.

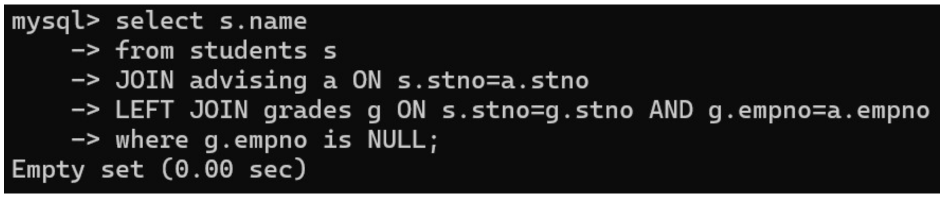
select s.name

from students s

    JOIN advising a ON s.stno=a.stno

    LEFT JOIN grades g ON s.stno=g.stno AND g.empno=a.empno

where g.empno is NULL;



```
mysql> select s.name
    -> from students s
    -> JOIN advising a ON s.stno=a.stno
    -> LEFT JOIN grades g ON s.stno=g.stno AND g.empno=a.empno
    -> where g.empno is NULL;
Empty set (0.00 sec)
```

11. Find the names of students who have failed all their courses (failing is defined as a grade less than 60).

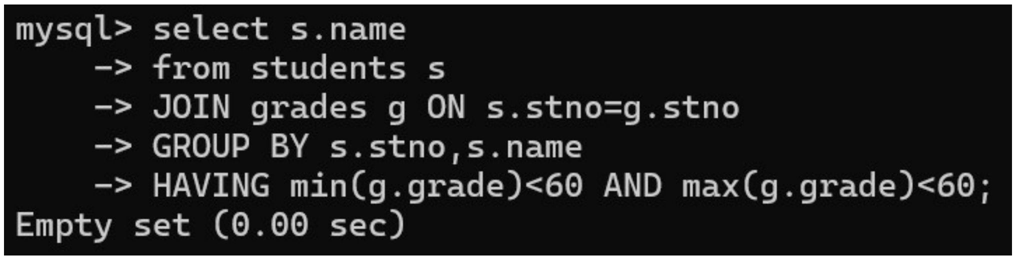
select s.name

from students s

JOIN grades g ON s.stno=g.stno

GROUP BY s.stno,s.name

HAVING min(g.grade)<60 AND max(g.grade)<60;



```
mysql> select s.name
    -> from students s
    -> JOIN grades g ON s.stno=g.stno
    -> GROUP BY s.stno,s.name
    -> HAVING min(g.grade)<60 AND max(g.grade)<60;
Empty set (0.00 sec)
```

12. Find the highest grade of a student who never took cs110.

select max(g.grade)

from grades g   where

g.stno NOT in(

select g2.stno   from

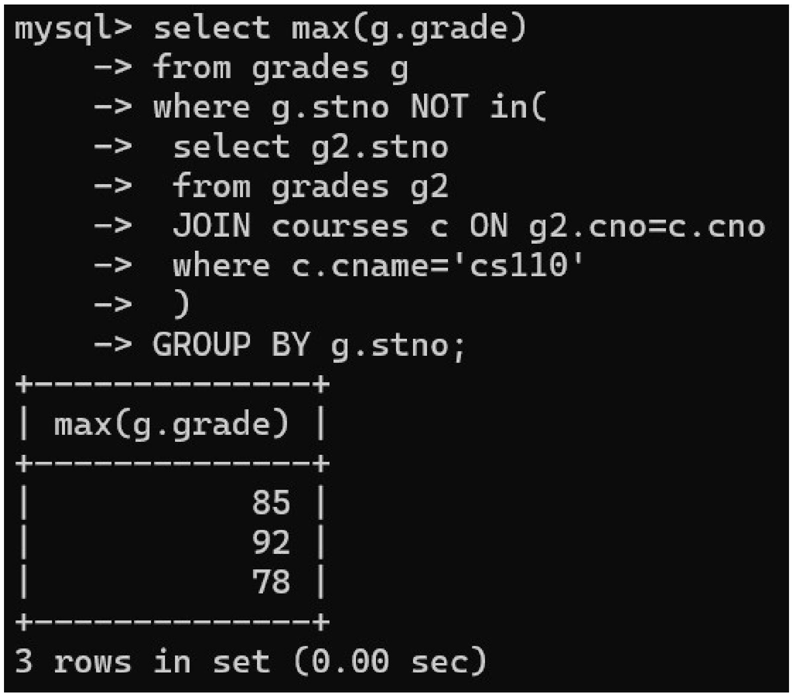
grades g2

JOIN courses c ON g2.cno=c.cno

where c.cname='cs110'

)

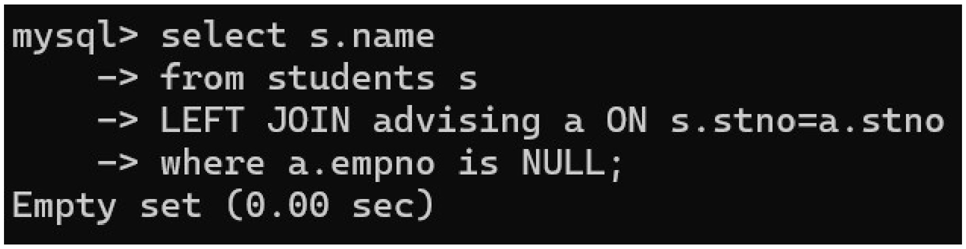
GROUP BY g.stno;



```
mysql> select max(g.grade)
    -> from grades g
    -> where g.stno NOT in(
    ->  select g2.stno
    ->  from grades g2
    ->  JOIN courses c ON g2.cno=c.cno
    ->  where c.cname='cs110'
    ->  )
    -> GROUP BY g.stno;
+--------------+
| max(g.grade) |
+--------------+
|           85 |
|           92 |
|           78 |
+--------------+
3 rows in set (0.00 sec)
```

13. Find the names of students who do not have an advisor.

```
select s.name

from students s

    LEFT JOIN advising a ON s.stno=a.stno

where a.empno is NULL;
```



```
mysql> select s.name
    -> from students s
    -> LEFT JOIN advising a ON s.stno=a.stno
    -> where a.empno is NULL;
Empty set (0.00 sec)
```

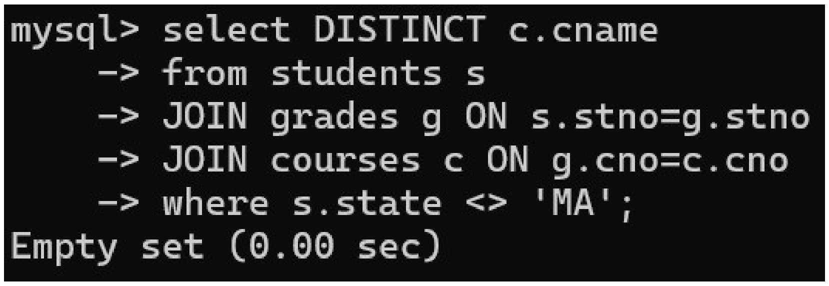
14. Find names of courses taken by students who do not live in Massachusetts (MA).

```
select DISTINCT c.cname

    from students s
```

JOIN grades g ON s.stno=g.stno

JOIN courses c ON g.cno=c.cno

where s.state <> 'MA';



```
mysql> select DISTINCT c.cname
    -> from students s
    -> JOIN grades g ON s.stno=g.stno
    -> JOIN courses c ON g.cno=c.cno
    -> where s.state <> 'MA';
Empty set (0.00 sec)
```