Practical 2

1. **USING (practical 1)**

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

```
mysql> select count(*) from customer where grade > (select avg(grade) from
customer where city = 'New York');
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.01 sec)
```

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

```
mysql> select salesman_id, name from salesman a where 1<(select count(*) from
customer where salesman_id = a.salesman_id);
+-------------+------------+
| salesman_id | name       |
+-------------+------------+
|        5001 | James Hoog |
|        5002 | Nail Knite |
+-------------+------------+
2 rows in set (0.01 sec)
```

3. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

```
mysql> delete from orders where salesman_id = 5001;
Query OK, 3 rows affected (0.04 sec)
mysql> delete from customer where salesman_id = 5001;
Query OK, 2 rows affected (0.01 sec)

mysql> delete from salesman where salesman_id = 5001;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> select count(*) from customer where grade > (select avg(grade) from customer where city = 'New York');
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.01 sec)
```

```
mysql> select salesman_id, name from salesman a where 1<(select count(*) from customer where salesman_id = a.salesman_id);
+-------------+------------+
| salesman_id | name       |
+-------------+------------+
|        5001 | James Hoog |
|        5002 | Nail Knite |
+-------------+------------+
2 rows in set (0.01 sec)
```

```
mysql> select * from salesman;
+--------------+------------+---------+------------+
| salesman_id | name       | city    | commission |
+--------------+------------+---------+------------+
|        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 |
+--------------+------------+---------+------------+
5 rows in set (0.00 sec)

mysql> select * from orders;
+----------+-----------+------------+-------------+-------------+
| order_no | purch_amt | order_date | customer_id | salesman_id |
+----------+-----------+------------+-------------+-------------+
|    70001 |     150.5 | 2012-10-05 |        3005 |        5002 |
|    70003 |    2480.4 | 2012-10-10 |        3009 |        NULL |
|    70004 |     110.5 | 2012-08-17 |        3009 |        NULL |
|    70007 |     948.5 | 2012-09-10 |        3005 |        5002 |
|    70009 |    270.65 | 2012-09-10 |        3001 |        NULL |
|    70010 |   1983.43 | 2012-10-10 |        3004 |        5006 |
|    70011 |     75.29 | 2012-08-17 |        3003 |        5007 |
|    70012 |    250.45 | 2012-06-27 |        3008 |        5002 |
+----------+-----------+------------+-------------+-------------+
8 rows in set (0.00 sec)

mysql> select * from customer;
+-------------+---------------+------------+-------+-------------+
| customer_id | customer_name | city       | grade | salesman_id |
+-------------+---------------+------------+-------+-------------+
|        3001 | Brad Guzan    | London     |  NULL |        NULL |
|        3003 | Jozy Altidor  | Moncow     |   200 |        5007 |
|        3004 | Fabian Johns  | Paris      |   300 |        5006 |
|        3005 | Graham Zusi   | California |   200 |        5002 |
|        3008 | Julian Green  | London     |   300 |        5002 |
|        3009 | Geoff Camero  | Berlin     |   100 |        NULL |
+-------------+---------------+------------+-------+-------------+
6 rows in set (0.00 sec)
```

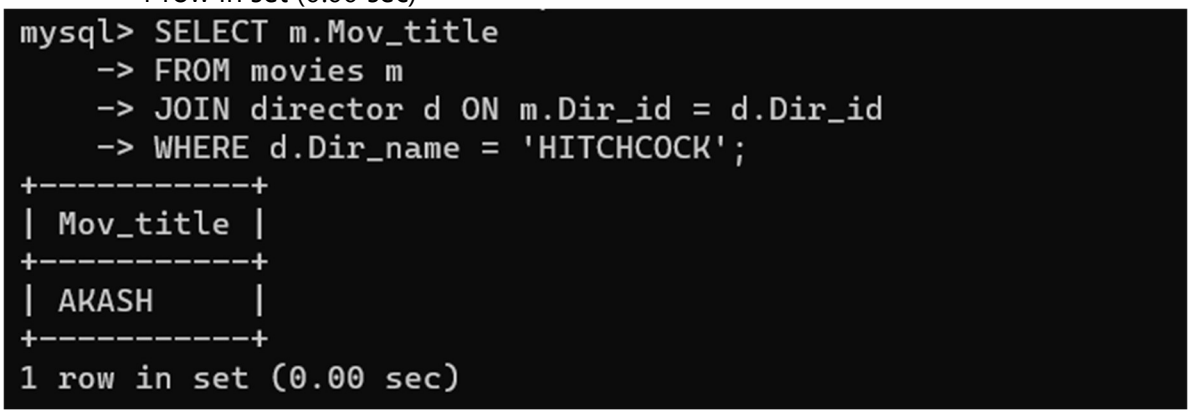
Design ERD for the following schema and execute the following Queries on it:

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)

Write SQL queries to

1. List the titles of all movies directed by 'Hitchcock'.

```
mysql> SELECT m.Mov_title
    -> FROM movies m
    -> JOIN director d ON m.Dir_id = d.Dir_id
    -> 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.

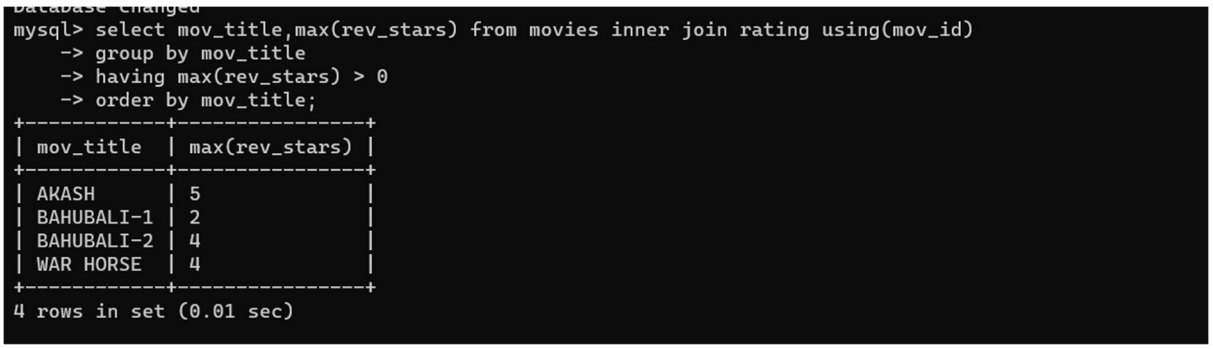
3. List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).

```
mysql> SELECT DISTINCT a.Act_id, a.Act_name
    -> FROM actor a
    -> JOIN movie_cast mc ON a.Act_id = mc.Act_id
    -> JOIN movies m ON mc.Mov_id = m.Mov_id
    -> WHERE m.Mov_year < 2000 OR m.Mov_year > 2015;
+--------+----------+
| Act_id | Act_name |
+--------+----------+
|    301 | Anushka  |
+--------+----------+
1 row in set (0.01 sec)
```

```
mysql> SELECT DISTINCT a.Act_id, a.Act_name
    -> FROM actor a
    -> JOIN movie_cast mc ON a.Act_id = mc.Act_id
    -> JOIN movies m ON mc.Mov_id = m.Mov_id
    -> WHERE m.Mov_year < 2000 OR m.Mov_year > 2015;
+--------+----------+
| Act_id | Act_name |
+--------+----------+
|    301 | Anushka  |
+--------+----------+
1 row in set (0.01 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.

```
mysql> select mov_title,max(rev_stars) from movies inner join rating using(mov_id)
   -> group by mov_title
   -> having max(rev_stars) > 0
   -> order by mov_title;
+------------+---------------+
| mov_title  | max(rev_stars) |
+------------+---------------+
| AKASH      | 5             |
| BAHUBALI-1 | 2             |
| BAHUBALI-2 | 4             |
| WAR HORSE  | 4             |
+------------+---------------+
4 rows in set (0.01 sec)
```



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

```
mysql> update rating set rev_stars=5 where mov_id in(select mov_id from movies
where
dir_id in (select dir_id from director where dir_name='STEVEN SPIELBERG'));
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 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)

For even rollnumbers(any 10)

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

```
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
    -> AND g.cno NOT IN(
    -> select cno
    -> from courses
    -> where cr != 4);
+-----------------+
| name            |
+-----------------+
| Edwards P.David |
| Mixon Leatha    |
| Pierce Richard  |
| Rawlings Jerry  |
| Prior Lorraine  |
| Lewis Jerry     |
+-----------------+
6 rows in set (0.01 sec)
```

```
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
    -> AND g.cno NOT IN(
    -> select cno
    -> from courses
    -> where cr != 4);
+-----------------+
| name            |
+-----------------+
| Edwards P.David |
| Mixon Leatha    |
| Pierce Richard  |
| Rawlings Jerry  |
| Prior Lorraine  |
| Lewis Jerry     |
+-----------------+
6 rows in set (0.01 sec)
```

2. Find the names of students who took no four-credit courses.

```
mysql> select distinct s.name
   -> from students s
   -> where s.stno NOT IN (
   -> select distinct g.stno
   -> from grades g
   -> join courses c ON g.cno = c.cno
   -> where c.cr = 4);
+---------------+
| name          |
+---------------+
| Grogan A. Mary |
| Mclane Sandy   |
| Novak Roland   |
+---------------+
3 rows in set (0.00 sec)
```

```
mysql> select distinct s.name
    -> from students s
    -> where s.stno NOT IN (
    -> select distinct g.stno
    -> from grades g
    -> join courses c ON g.cno = c.cno
    -> where c.cr = 4);
+----------------+
| name           |
+----------------+
| Grogan A. Mary |
| Mclane Sandy   |
| Novak Roland   |
+----------------+
3 rows in set (0.00 sec)
```

3. Find the names of students who took cs210 or cs310.
   mysql> select name from students where stno in (select stno from grades where cno='cs210' or cno='cs310');
   +----------------+
   | name |
   +----------------+
   | Edwards P.David |
   | Mixon Leatha |
   | Pierce Richard |
   | Prior Lorraine |
   | Lewis Jerry |
   +----------------+
   5 rows in set (0.01 sec)

```
mysql> select name from students where stno in (select stno from grades where cno='cs210' or cno='cs310');
+----------------+
| name           |
+----------------+
| Edwards P.David |
| Mixon Leatha   |
| Pierce Richard |
| Prior Lorraine |
| Lewis Jerry    |
+----------------+
5 rows in set (0.01 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.

   mysql> select s.name
   -> from students s
   -> where s.stno IN(
   -> select g1.stno
   -> from grades g1
   -> where g1.cno = 'cs210'

```
    -> AND g1.grade > (
    -> select max(g2.grade)
    -> from grades g2
    -> where g2.cno = 'cs310'
    -> ))
    -> AND s.stno NOT IN(
    -> select g3.stno
    -> from grades g3
    -> join instructors i ON g3.empno = i.empno
    -> where i.name = 'Evans Robert');
Empty set (0.01 sec)
```

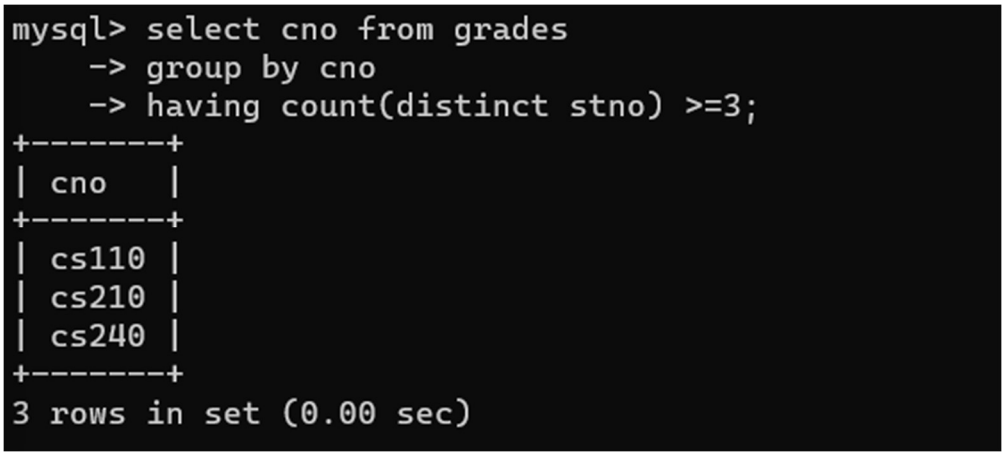
```
mysql> select s.name
    -> from students s
    -> where s.stno IN(
    -> select g1.stno
    -> from grades g1
    -> where g1.cno = 'cs210'
    -> AND g1.grade > (
    -> select max(g2.grade)
    -> from grades g2
    -> where g2.cno = 'cs310'
    -> ))
    -> AND s.stno NOT IN(
    -> select g3.stno
    -> from grades g3
    -> join instructors i ON g3.empno = i.empno
    -> where i.name = 'Evans Robert');
Empty set (0.01 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.

```
mysql> select cno from grades
    -> group by cno
    -> having count(distinct stno) >=2;
+-------+
| cno   |
+-------+
| cs110 |
| cs210 |
| cs240 |
| cs310 |
| cs410 |
+-------+
5 rows in set (0.01 sec)
```

```
mysql> select cno from grades
    -> group by cno
    -> having count(distinct stno) >=2;
+-------+
| cno   |
+-------+
| cs110 |
| cs210 |
| cs240 |
| cs310 |
| cs410 |
+-------+
5 rows in set (0.01 sec)
```

```
mysql> select cno from grades
   -> group by cno
   -> having count(distinct stno) >=3;
+-------+
| cno   |
+-------+
| cs110 |
| cs210 |
| cs240 |
+-------+
3 rows in set (0.00 sec)
```



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

```
mysql> SELECT s.name
   -> FROM students s
   -> JOIN grades g ON s.stno = g.stno
   -> WHERE g.cno = 'cs210' AND g.grade = (SELECT MAX(grade) FROM grades
WHERE cno = 'cs210');
+----------------+
| name           |
```

```
+-----------------+
| Edwards P.David |
| Pierce Richard  |
+-----------------+
2 rows in set (0.00 sec)
```

```
mysql> SELECT s.name
    -> FROM students s
    -> JOIN grades g ON s.stno = g.stno
    -> WHERE g.cno = 'cs210' AND g.grade = (SELECT MAX(grade) FROM grades WHERE cno = 'cs210');
+-----------------+
| name            |
+-----------------+
| Edwards P.David |
| Pierce Richard  |
+-----------------+
2 rows in set (0.00 sec)
```

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

```
mysql> select s.name
   -> from students s
   -> where NOT EXISTS (
   -> select 1
   -> from advising a
   -> where a.stno = s.stno
   -> AND NOT EXISTS (
   -> select 1
   -> from grades g
   -> where g.stno = a.stno
   -> AND g.empno = a.empno
   -> ));
+-----------------+
| name            |
+-----------------+
| Edwards P.David |
| Grogan A. Mary  |
| Prior Lorraine  |
| Lewis Jerry     |
+-----------------+
4 rows in set (0.01 sec)
```

```
mysql> select s.name
    -> from students s
    -> where NOT EXISTS (
    -> select 1
    -> from advising a
    -> where a.stno = s.stno
    -> AND NOT EXISTS (
    -> select 1
    -> from grades g
    -> where g.stno = a.stno
    -> AND g.empno = a.empno
    -> ));
+-----------------+
| name            |
+-----------------+
| Edwards P.David |
| Grogan A. Mary  |
| Prior Lorraine  |
| Lewis Jerry     |
+-----------------+
4 rows in set (0.01 sec)
```

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

```
mysql> select max(grade) AS highest_grade
   -> from grades
   -> where stno NOT IN (
   -> select stno
   -> from grades
   -> where cno = 'cs110');
+---------------+
| highest_grade |
+---------------+
|           100 |
+---------------+
1 row in set (0.00 sec)
```

```
mysql> select max(grade) AS highest_grade
    -> from grades
    -> where stno NOT IN (
    -> select stno
    -> from grades
    -> where cno = 'cs110');
+---------------+
| highest_grade |
+---------------+
|           100 |
+---------------+
1 row in set (0.00 sec)
```

9. Find course numbers for courses that enroll exactly two students.

```
mysql> SELECT cno
-> FROM grades
-> GROUP BY cno
-> HAVING COUNT(DISTINCT stno) = 2;
+-------+
| cno |
+-------+
| cs310 |
| cs410 |
+-------+
2 rows in set (0.00 sec)
```

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

```
mysql> SELECT s.name
-> FROM students s
-> WHERE NOT EXISTS (
-> SELECT 1
-> FROM advising a
-> WHERE a.stno = s.stno
-> AND NOT EXISTS (
-> SELECT 1
-> FROM grades g
-> WHERE g.stno = a.stno
-> AND g.empno = a.empno
-> )
-> );
+------------------+
| name |
+------------------+
| edwards p. david |
| Grogan A. Mary |
| Prior Lorraine |
| Lewis Jerry |
+------------------+
4 rows in set (0.00 sec)
```