Rehmah Ahmed Batki
MSC DSAI L010

Practical - 2

20-12-24

Group-by and Sub-Queries

Write-up: -

- DBMS Architecture
- DML
- DCL
- ERD
- Components of ERD
- Relationships in ERD
- SUBQUERY and Types of Subquery
- GROUPBY and HAVING
- Join and Types of join

Q1) USING (practical - 1)

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

```
mysql> SELECT COUNT(*) AS customers_above_avg_grade
    -> FROM customer
    -> WHERE grade > (
    ->     SELECT AVG(grade)
    ->     FROM customer
    ->     WHERE city = 'Bangalore'
    -> );
+---------------------------+
| customers_above_avg_grade |
+---------------------------+
|                         0 |
+---------------------------+
1 row in set (0.03 sec)
```

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

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

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

```
mysql> SELECT s.salesman_id, s.name, s.city, 'Has Customers' AS 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, s.city, 'No Customers' AS status
    -> FROM salesman s
    -> LEFT JOIN customer c ON s.salesman_id = c.salesman_id
    -> WHERE s.city = s.city
    -> AND c.customer_id IS NULL;
```

```
+-------------+------------+----------+---------------+
| salesman_id | name       | city     | status        |
+-------------+------------+----------+---------------+
|        5001 | James Hooq | New York | Has Customers |
|        5006 | Mc Lyon    | Paris    | Has Customers |
|        5005 | Pit Alex   | London   | No Customers  |
+-------------+------------+----------+---------------+
3 rows in set (0.01 sec)
```

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

```
mysql> CREATE VIEW highest_order_per_day AS
    -> SELECT o.order_date,
    ->        o.salesman_id,
    ->        o.customer_id,
    ->        o.purch_amt
    -> FROM `order` 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 `order`
    ->     GROUP BY order_date
    -> );
Query OK, 0 rows affected (0.02 sec)
```

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

```
mysql> DELETE FROM `order`
    -> WHERE salesman_id = 5001;
Query OK, 3 rows affected (0.01 sec)
```

Rehmah Ahmed Batki

MSC DSAI L010

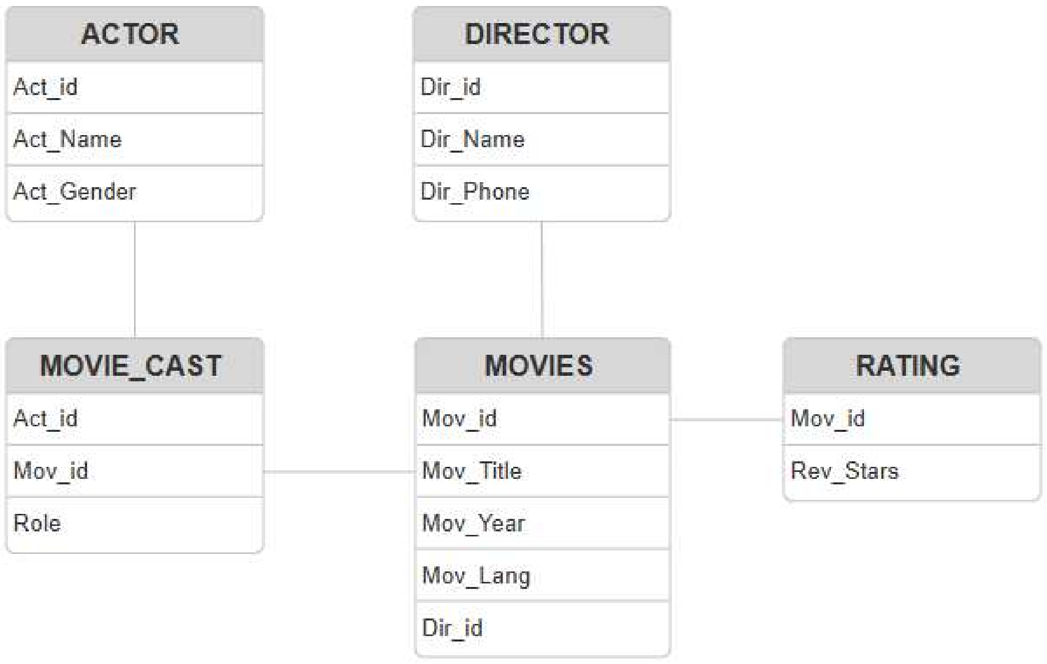
```
mysql> select * from salesman;
+-------------+------------+----------+------------+
| salesman_id | name       | city     | commission |
+-------------+------------+----------+------------+
|        5001 | James Hooq | New York |       0.15 |
|        5002 | Nail Knite | Paris    |       0.13 |
|        5003 | Lauson Hen | NULL     |       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 `order`;
+----------+-----------+------------+-------------+-------------+
| order_no | purch_amt | order_date | customer_id | salesman_id |
+----------+-----------+------------+-------------+-------------+
|    70001 |     150.5 | 2016-10-05 |        3005 |        5002 |
|    70003 |    2480.4 | 2016-10-10 |        3009 |        NULL |
|    70004 |     110.5 | 2016-08-17 |        3009 |        NULL |
|    70007 |     948.5 | 2016-09-10 |        3005 |        5002 |
|    70009 |    270.65 | 2016-09-10 |        3001 |        NULL |
|    70010 |   1983.43 | 2016-10-10 |        3004 |        5006 |
|    70011 |     75.29 | 2016-08-17 |        3003 |        5007 |
|    70012 |    250.45 | 2016-06-27 |        3008 |        5002 |
+----------+-----------+------------+-------------+-------------+
8 rows in set (0.00 sec)
```

Rehmah Ahmed Batki
MSC DSAI L010

Q2) 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)



```
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 (0.02 sec)

mysql> INSERT INTO ACTOR VALUES (301,'ANUSHKA','F');
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO ACTOR VALUES (302, 'PRABHAS', 'M');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ACTOR VALUES (303, 'PUNITH', 'M');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO ACTOR VALUES (304, 'JERMY', 'M');
Query OK, 1 row affected (0.00 sec)

mysql> CREATE TABLE DIRECTOR (
    ->     DIR_ID INT(3),
    ->     DIR_NAME VARCHAR(20),
    ->     DIR_PHONE BIGINT(10),
    ->     PRIMARY KEY (DIR_ID)
    -> );
Query OK, 0 rows affected, 2 warnings (0.01 sec)

mysql> INSERT INTO DIRECTOR VALUES (60, 'RAJAMOULI', 8751611001);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO DIRECTOR VALUES (61, 'HITCHCOCK', 7766138911);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO DIRECTOR VALUES (62, 'FARAN', 9986776531);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO DIRECTOR VALUES (63, 'STEVEN SPIELBERG',
8989776530);
Query OK, 1 row affected (0.00 sec)

mysql> CREATE TABLE MOVIES (
    ->     MOV_ID INT(4),
    ->     MOV_TITLE VARCHAR(25),
    ->     MOV_YEAR INT(4),
    ->     MOV_LANG 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 (0.02 sec)

mysql> INSERT INTO MOVIES VALUES (1001, 'BAHUBALI-2', 2017,
'TELAGU', 60);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIES VALUES (1002, 'BAHUBALI-1', 2015,
'TELAGU', 60);
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO MOVIES VALUES (1003, 'AKASH', 2008, 'KANNADA',
61);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIES VALUES (1004, 'WAR HORSE', 2011,
'ENGLISH', 63);
Query OK, 1 row affected (0.00 sec)

mysql> CREATE TABLE MOVIE_CAST (
    ->     ACT_ID INT(3),
   ->     MOV_ID INT(4),
   ->     ROLE_NAME VARCHAR(30),
   ->     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 (0.03 sec)

mysql> INSERT INTO MOVIE_CAST VALUES (301, 1002, 'HEROINE');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIE_CAST VALUES (301, 1001, 'HEROINE');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO MOVIE_CAST VALUES (303, 1003, 'HERO');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIE_CAST VALUES (303, 1002, 'GUEST');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO MOVIE_CAST VALUES (304, 1004, 'HERO');
Query OK, 1 row affected (0.00 sec)

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> INSERT INTO RATING VALUES (1001, 4);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO RATING VALUES (1002, 2);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO RATING VALUES (1003, 5);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO RATING VALUES (1004, 4);
Query OK, 1 row affected (0.01 sec)

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)

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

```
mysql> select * from movies;
+--------+------------+----------+----------+--------+
| MOV_ID | MOV_TITLE  | MOV_YEAR | MOV_LANG | 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)

mysql> select * from director;
+--------+------------------+------------+
| DIR_ID | DIR_NAME         | DIR_PHONE  |
+--------+------------------+------------+
|     60 | RAJAMOULI        | 8751611001 |
|     61 | HITCHCOCK        | 7766138911 |
|     62 | FARAN            | 9986776531 |
|     63 | STEVEN SPIELBERG | 8989776530 |
+--------+------------------+------------+
4 rows in set (0.00 sec)
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'.

```
mysql> SELECT MOV_TITLE FROM movies WHERE DIR_ID = (SELECT
DIR_ID FROM director WHERE 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.

```
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).

```
mysql> SELECT DISTINCT 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_NAME |
+----------+
| ANUSHKA  |
+----------+
1 row in 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.

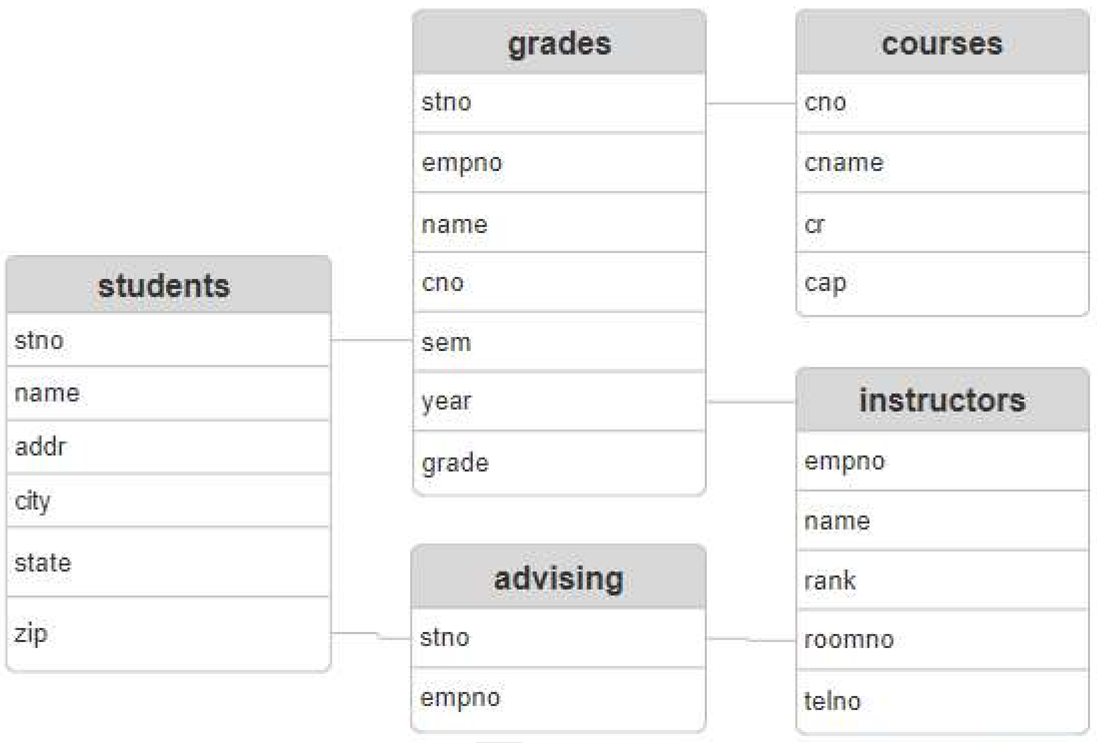
```
mysql> SELECT m.MOV_TITLE, r.REV_STARS
    -> FROM movies m
    -> JOIN rating r ON m.MOV_ID = r.MOV_ID
    -> WHERE r.REV_STARS IS NOT NULL
    -> ORDER BY m.MOV_TITLE;
+------------+-----------+
| MOV_TITLE  | REV_STARS |
+------------+-----------+
| AKASH      | 5         |
| BAHUBALI-1 | 2         |
| BAHUBALI-2 | 4         |
| WAR HORSE  | 4         |
+------------+-----------+
4 rows in set (0.00 sec)
```

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

```
mysql> UPDATE rating r
    -> JOIN movies m ON r.MOV_ID = m.MOV_ID
    -> JOIN director d ON m.DIR_ID = d.DIR_ID
    -> SET r.REV_STARS = 5
    -> WHERE d.DIR_NAME = 'STEVEN SPIELBERG';
Query OK, 1 row affected (0.02 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)
```

Q3) Design ERD for the following schema and execute the following Queries on it:



```
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)

mysql> CREATE TABLE INSTRUCTORS (
    ->     empno INT PRIMARY KEY,
    ->     name VARCHAR(50),
    ->     `rank` VARCHAR(50),
    ->     roomno VARCHAR(10),
    ->     telno VARCHAR(15)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE COURSES (
    ->     cno INT PRIMARY KEY,
    ->     cname VARCHAR(50),
    ->     cr INT,
    ->     cap INT
    -> );
Query OK, 0 rows affected (0.02 sec)

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 (0.03 sec)

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> INSERT INTO students (stno, name, addr, city, state, zip) values
(1, 'Rehmah', 'Mumbai', 'Thane', 'MH', '401107'),
(2, 'Tom', 'Mumbai', 'Thane', 'MH', '401107'),
(3, 'someoneelse', 'Mumbai', 'Thane', 'MH', '401107')
;
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

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

mysql> insert into instructors (empno, name, `rank`, roomno, telno) values
(101, 'abc', 'A', 'R1', '112233'),
(102, 'efg', 'B', 'R2', '112233'),
(103, 'xyz', 'C', 'R3', '112233')
;
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

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> INSERT INTO ADVISING (stno, empno) VALUES
(1, 101),
(2, 102),
(3, 103);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from students;
+------+-------------+--------+-------+-------+--------+
| stno | name        | addr   | city  | state | zip    |
+------+-------------+--------+-------+-------+--------+
|    1 | Rehmah      | Mumbai | Thane | MH    | 401107 |
|    2 | Tom         | Mumbai | Thane | MH    | 401107 |
|    3 | someoneelse | Mumbai | Thane | MH    | 401107 |
+------+-------------+--------+-------+-------+--------+
3 rows in set (0.00 sec)

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)

mysql> select * from instructors;
+-------+------+------+--------+--------+
| empno | name | rank | roomno | telno  |
+-------+------+------+--------+--------+
|   101 | abc  | A    | R1     | 112233 |
|   102 | efg  | B    | R2     | 112233 |
|   103 | xyz  | C    | R3     | 112233 |
+-------+------+------+--------+--------+
3 rows in set (0.00 sec)

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)

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

For even roll numbers(any 10)

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

```
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 SUM(CASE WHEN c.cr != 4 THEN 1 ELSE 0 END) = 0;
+--------+
| name   |
+--------+
| Tom |
+--------+
1 row in set (0.00 sec)
```

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

```
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 SUM(CASE WHEN c.cr = 4 THEN 1 ELSE 0 END) = 0;
+-------------+
| name        |
+-------------+
| Rehmah      |
| someoneelse |
+-------------+
2 rows in set (0.00 sec)
```

3. Find the names of students who took cs210 or 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   |
+--------+
| Tom |
+--------+
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 Physics101 and did not take any course with 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
    -> JOIN grades g2 ON s.stno = g2.stno
    -> JOIN courses c2 ON g2.cno = c2.cno
    -> LEFT JOIN advising a ON s.stno = a.stno
    -> LEFT JOIN instructors i ON a.empno = i.empno
    -> WHERE c1.cname = 'CS210'
    ->   AND g1.grade > (
    ->       SELECT MAX(g.grade)
    ->       FROM grades g
    ->       JOIN courses c ON g.cno = c.cno
    ->       WHERE c.cname = 'Physics101'
    ->   )
    ->   AND (i.name != 'Prof. Evans' OR i.name IS NULL);
+--------+
| name   |
+--------+
| Tom |
+--------+
1 row in 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.

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

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.

```
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(g1.grade)
    ->       FROM grades g1
    ->       JOIN courses c1 ON g1.cno = c1.cno
    ->       WHERE c1.cname = 'CS210'
    ->   );
+------+
| name |
+------+
| Tom  |
+------+
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.

```
mysql> SELECT DISTINCT i.name
    -> FROM instructors i
    -> JOIN advising a ON i.empno = a.empno
    -> JOIN grades g ON a.stno = g.stno
    -> JOIN courses c ON g.cno = c.cno
    -> WHERE c.cno IN (
    ->     SELECT c1.cno
    ->     FROM instructors i1
    ->     JOIN advising a1 ON i1.empno = a1.empno
    ->     JOIN grades g1 ON a1.stno = g1.stno
    ->     JOIN courses c1 ON g1.cno = c1.cno
    ->     WHERE i1.rank = 'A'
    -> );
+------+
| name |
+------+
| abc  |
+------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT MIN(g.grade) AS lowest_grade
    -> FROM grades g
    -> JOIN courses c ON g.cno = c.cno
    -> WHERE g.sem = 'Spring'
    ->   AND g.year = 2003;
```

```
+--------------+
| lowest_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 DISTINCT s.name
FROM students s
WHERE NOT EXISTS (
    SELECT 1
    FROM instructors i
    JOIN advising a ON i.empno = a.empno
    JOIN grades g ON a.stno = g.stno
    JOIN courses c ON g.cno = c.cno
    WHERE i.name = 'Prof. Evans'
      AND c.cno NOT IN (
          SELECT cno
          FROM grades
          WHERE stno = s.stno
      )
);
```

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

```
mysql> SELECT DISTINCT s.name
    -> FROM students s
    -> WHERE NOT EXISTS (
    ->     SELECT 1
    ->     FROM instructors i
    ->     JOIN advising a ON i.empno = a.empno
    ->     JOIN grades g ON a.stno = g.stno
    ->     JOIN courses c ON g.cno = c.cno
    ->     WHERE i.name = 'Prof. Evans'
    ->       AND c.cno NOT IN (
    ->           SELECT cno
    ->           FROM grades
    ->           WHERE stno = s.stno
    ->       )
    -> );
+-------------+
| name        |
+-------------+
| Rehmah      |
| Tom         |
| someoneelse |
+-------------+
3 rows in set (0.01 sec)
```

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

```
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 g.grade >= 60
    -> );
Empty set (0.00 sec)
```

12. Find the highest grade of a student who never took cs210.

```
mysql> SELECT MAX(g.grade) AS highest_grade
    -> FROM grades g
    -> WHERE g.stno NOT IN (
    ->     SELECT DISTINCT g1.stno
    ->     FROM grades g1
    ->     JOIN courses c ON g1.cno = c.cno
    ->     WHERE c.cname = 'CS210'
    -> );
+---------------+
| highest_grade |
+---------------+
|            85 |
+---------------+
1 row 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;
```

14. Find names of courses taken by students who do not live in Massachusetts (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';
+------------+
| cname      |
+------------+
| Math101    |
| CS210      |
| Physics101 |
+------------+
3 rows in set (0.00 sec)
```