Practical2: Subquery-join operations on Relational Schema

USING (practical 1)

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

```
mysql> select count(customer_id) as customer_count from customer
    -> where grade > (
    -> select avg(grade) from customer where city='Bangalore');
+----------------+
| customer_count |
+----------------+
|              0 |
+----------------+
1 row in set (0.00 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
    -> CROSS JOIN customer c
    -> on s.salesman_id = c.salesman_id
    -> group by s.salesman_id,s.name
    -> having count(c.customer_id)>1;
+-------------+------------+
| salesman_id | name       |
+-------------+------------+
|        5001 | James Hoog |
|        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 where exists(
    -> select 1 from customer c where s.salesman_id=c.salesman_id and s.city=c.city)
    -> UNION
    -> select s.salesman_id,s.name,s.city,'No customers' as status
    -> from salesman s where not exists(
    -> select 1 from customer c where s.salesman_id=c.salesman_id and s.city=c.city);
+------------+-----------+----------+--------------+
| salesman_id | name      | city     | status       |
+------------+-----------+----------+--------------+
|        5001 | James Hoog | New York | Has customers |
|        5006 | Mc Lyon    | Paris    | Has customers |
|        5002 | Nail Knite | Paris    | No customers  |
|        5003 | Lauson Hen |          | No customers  |
|        5005 | Pit Alex   | London   | No customers  |
|        5007 | Paul Adam  | Rome     | 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.

```
mysql> CREATE VIEW Salesman_Highest_Order AS
    -> SELECT o.salesman_id, s.name AS salesman_name, o.customer_id, c.customer_name AS customer_name,
    ->        o.order_date, o.purch_amt
    -> FROM orders o
    -> JOIN salesman s ON o.salesman_id = s.salesman_id
    -> JOIN customer c ON o.customer_id = c.customer_id
    -> WHERE o.purch_amt = (
    ->     SELECT MAX(purch_amt)
    ->     FROM orders
    ->     WHERE order_date = o.order_date
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> select * from Salesman_Highest_Order;
+-------------+--------------+-------------+---------------+------------+-----------+
| salesman_id | salesman_name | customer_id | customer_name | order_date | purch_amt |
+-------------+--------------+-------------+---------------+------------+-----------+
|        5001 | James Hoog    |        3002 | Nick Rimando  | 2016-09-10 |      5760 |
|        5002 | Nail Knite    |        3005 | Graham Zusi   | 2016-10-05 |     150.5 |
|        5001 | James Hoog    |        3007 | Brad Davis    | 2016-07-27 |    2400.6 |
|        5002 | Nail Knite    |        3008 | Julian Green  | 2016-06-27 |    250.45 |
+-------------+--------------+-------------+---------------+------------+-----------+
4 rows in set (0.01 sec)
```

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

```
mysql> delete from orders where salesman_id = 1000;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from customer where salesman_id = 1000;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from salesman where salesman_id = 1000;
Query OK, 0 rows affected (0.00 sec)
```

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)

```
mysql> create table actor(act_id int,act_name varchar(10),
    -> act_gender varchar(5),PRIMARY KEY (act_id));
Query OK, 0 rows affected (0.07 sec)

mysql> insert into actor values(301,'Anushka','F');
Query OK, 1 row affected (0.01 sec)

mysql> insert into actor values(302,'Prabhas','M');
Query OK, 1 row affected (0.01 sec)

mysql> insert into actor values(303,'Punith','M');
Query OK, 1 row affected (0.01 sec)

mysql> insert into actor values(304,'Jermy','M');
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> create table director(dir_id int,dir_name varchar(10),
    -> dir_phone char(10), PRIMARY KEY(dir_id));
Query OK, 0 rows affected (0.03 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 (0.01 sec)

mysql> insert into director values(62,'Faran',9986776531);
Query OK, 1 row affected (0.01 sec)

mysql> insert into director values(63,'Steven Spielberg',8989776530);
ERROR 1406 (22001): Data too long for column 'dir_name' at row 1
mysql> insert into director values(63,'Steven',8989776530);
Query OK, 1 row affected (0.01 sec)

mysql> select * from director;
+--------+-----------+------------+
| dir_id | dir_name  | dir_phone  |
+--------+-----------+------------+
|     60 | Rajamouli | 8751611001 |
|     61 | Hitchcock | 7766138911 |
|     62 | Faran     | 9986776531 |
|     63 | Steven    | 8989776530 |
+--------+-----------+------------+
```

```
mysql> create table movies(mov_id int,mov_title varchar(20),mov_year int,mov
_lang varchar(12),dir_id int,
    -> PRIMARY KEY(mov_id),
    -> FOREIGN KEY (dir_id) REFERENCES director(dir_id));
Query OK, 0 rows affected (0.04 sec)

mysql> insert into movies values(1001,'Bahubali-2',2017,'Telugu',60);
Query OK, 1 row affected (0.01 sec)

mysql> insert into movies values(1002,'Bahubali-1',2015,'Telugu',60);
Query OK, 1 row affected (0.01 sec)

mysql> insert into movies values(1003,'Akash',2008,'Kannada',61);
Query OK, 1 row affected (0.01 sec)

mysql> insert into movies values(1004,'War horse',2011,'English',63);
Query OK, 1 row affected (0.01 sec)

mysql> select * from movies;
+--------+------------+----------+----------+--------+
| mov_id | mov_title  | mov_year | mov_lang | dir_id |
+--------+------------+----------+----------+--------+
|   1001 | Bahubali-2 |     2017 | Telugu   |     60 |
|   1002 | Bahubali-1 |     2015 | Telugu   |     60 |
|   1003 | Akash      |     2008 | Kannada  |     61 |
|   1004 | War horse  |     2011 | English  |     63 |
+--------+------------+----------+----------+--------+
```

```
mysql> create table rating(mov_id int,rev_stars varchar(5),
    -> PRIMARY KEY(mov_id),
    -> FOREIGN KEY(mov_id) REFERENCES movies(mov_id));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into rating values(1001,4);
Query OK, 1 row affected (0.01 sec)

mysql> insert into rating values(1002,2);
Query OK, 1 row affected (0.01 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 rating;
+--------+-----------+
| mov_id | rev_stars |
+--------+-----------+
|   1001 | 4         |
|   1002 | 2         |
|   1003 | 5         |
|   1004 | 4         |
+--------+-----------+
```

```
mysql> create table movie_cast(act_id int,mov_id int,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 (0.05 sec)

mysql> insert into movie_cast values(301,1002,'Heroine');
Query OK, 1 row affected (0.01 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 (0.01 sec)

mysql> insert into movie_cast values(303,1002,'Guest');
Query OK, 1 row affected (0.01 sec)

mysql> insert into movie_cast values(304,1004,'Hero');
Query OK, 1 row affected (0.01 sec)

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    |
+--------+--------+---------+
```

Shriyanshi Maurya | L024

Write SQL queries to

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

```
mysql> select m.mov_title from movies m
    -> CROSS 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.

```
mysql> select distinct m.mov_title from movies m
    -> JOIN movie_cast mc1 on m.mov_id = mc1.mov_id
    -> where mc1.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 (0.00 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_id,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.

```
mysql> select m.mov_title,r.rev_stars as no_of_stars, max(r.rev_stars) as max_stars from movies m
    -> join rating r on m.mov_id = r.mov_id
    -> where r.rev_stars >1
    -> group by m.mov_id
    -> order by m.mov_title;
+-----------+------------+-----------+
| mov_title | no_of_stars | max_stars |
+-----------+------------+-----------+
| Akash     | 5          | 5         |
| Bahubali-1 | 2          | 2         |
| Bahubali-2 | 4          | 4         |
| War horse | 4          | 4         |
+-----------+------------+-----------+
```

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

```
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");
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

3. 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 (0.03 sec)
```

```
mysql> create table instructors(empno int PRIMARY KEY,
    -> name varchar(20),emp_rank varchar(20),roomno varchar(10),
    -> telno varchar(15));
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> create table courses ( cno int PRIMARY KEY,
    -> cname varchar(20), cr int, cap int);
Query OK, 0 rows affected (0.03 sec)

mysql> create table grades(stno int,empno int, cno int, sem varchar(20),
    -> 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 (0.06 sec)

mysql> create table advising(stno int,empno int,PRIMARY KEY(stno,empno),
    -> FOREIGN KEY(empno) REFERENCES instructors(empno),
    -> FOREIGN KEY(stno) REFERENCES students(stno));
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> select * from students;
+------+------------------+-----------------+------------+-------+------+
| stno | name             | addr            | city       | state | zip  |
+------+------------------+-----------------+------------+-------+------+
| 1011 | Edwards P. David | 10 REd Rd.      | Newton     | MA    | 2159 |
| 2415 | Grogan A. MAry   | 8 Walnut ST.    | Malden     | MA    | 2148 |
| 2661 | Mixon Leatha     | 100 School ST.  | Brookline  | MA    | 2146 |
| 2890 | McLane Sandy     | 30 Case Rd.     | Boston     | MA    | 2122 |
| 3442 | Noval Roland     | 42 Beacon St.   | Nashua     | NH    | 3060 |
| 3566 | Pierce Richard   | 70 Park St.     | Brookline  | MA    | 2146 |
| 4022 | Prior Lorraine   | 8 Beacon St.    | Boston     | MA    | 2125 |
| 5544 | Rawlings Jerry   | 15 PLeasant Dr. | Boston     | MA    | 2115 |
| 5571 | Lewis Jerry      | 1 Main Rd.      | Providence | RI    | 2904 |
+------+------------------+-----------------+------------+-------+------+
9 rows in set (0.01 sec)
```

```
mysql> select * from instructors;
+-------+----------------+-------------+--------+-------+
| empno | name           | emp_rank    | roomno | telno |
+-------+----------------+-------------+--------+-------+
|    19 | Evana Robert   | Professor   | 82     | 7122  |
|    23 | Exxon George   | Professor   | 90     | 9101  |
|    56 | Sawyer Kathy   | Assoc.Prof. | 91     | 5110  |
|   126 | Davis William  | Assoc.Prof. | 72     | 5411  |
|   234 | Will Samuel    | Assoc.Prof. | 90     | 7024  |
+-------+----------------+-------------+--------+-------+
5 rows in set (0.00 sec)
```

```
mysql> select * from courses;
+-------+-----------------------------+------+------+
| cno   | cname                       | cr   | cap  |
+-------+-----------------------------+------+------+
| cs110 | Introduction to computing   |    4 |  120 |
| cs210 | Computer Programming        |    4 |  100 |
| cs240 | Computer architecture       |    3 |  100 |
| cs310 | Data structures             |    3 |   60 |
| cs350 | Higher level languages      |    3 |   50 |
| cs410 | Software engineering        |    3 |   40 |
| cs460 | Graphics                    |    3 |   30 |
+-------+-----------------------------+------+------+
7 rows in set (0.00 sec)
```

```
mysql> select * from grades;
+------+-------+-------+--------+------+-------+
| stno | empno | cno   | sem    | year | grade |
+------+-------+-------+--------+------+-------+
| 1011 |    19 | cs110 | Fail   | 2001 |    40 |
| 2661 |    19 | cs110 | Fail   | 2001 |    80 |
| 3566 |    19 | cs110 | Fail   | 2001 |    95 |
| 5544 |    19 | cs110 | Fail   | 2001 |   100 |
| 1011 |    23 | cs110 | Spring | 2002 |    75 |
| 4022 |    23 | cs110 | Spring | 2002 |    60 |
| 3566 |    19 | cs240 | Spring | 2002 |   100 |
| 5571 |    19 | cs240 | Spring | 2002 |    50 |
| 2415 |    19 | cs240 | Spring | 2002 |   100 |
| 3442 |   234 | cs410 | Spring | 2002 |    60 |
| 5571 |   234 | cs410 | Spring | 2002 |    80 |
| 1011 |    19 | cs210 | Fail   | 2002 |    90 |
| 2661 |    19 | cs210 | Fail   | 2002 |    70 |
| 3566 |    19 | cs210 | Fail   | 2002 |    90 |
| 5571 |    19 | cs210 | Spring | 2003 |    85 |
| 4022 |    19 | cs210 | Spring | 2003 |    70 |
| 5544 |    56 | cs240 | Spring | 2003 |    70 |
| 1011 |    56 | cs240 | Spring | 2003 |    90 |
| 4022 |    56 | cs240 | Spring | 2003 |    80 |
| 2661 |   234 | cs310 | Spring | 2003 |   100 |
| 4022 |   234 | cs310 | Spring | 2003 |    75 |
+------+-------+-------+--------+------+-------+
21 rows in set (0.00 sec)
```

```
mysql> select * from advising;
+------+-------+
| stno | empno |
+------+-------+
| 1011 |    19 |
| 2415 |    19 |
| 2661 |    23 |
| 2890 |    23 |
| 5544 |    23 |
| 3442 |    56 |
| 3566 |   126 |
| 4022 |   234 |
| 5571 |   234 |
+------+-------+
9 rows in set (0.00 sec)
```

For odd roll numbers(any 10)

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

```
mysql> select distinct s.name,s.stno from students s
    -> JOIN grades g on s.stno = g.stno
    -> JOIN courses c on g.cno = c.cno
    -> where c.cr = 4 and s.stno % 2 = 1 ;
+-------------------+------+
| name              | stno |
+-------------------+------+
| Edwards P. David  | 1011 |
| Mixon Leatha      | 2661 |
| Lewis Jerry       | 5571 |
+-------------------+------+
3 rows in set (0.00 sec)
```

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

```
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 s.stno % 2 = 1
    -> group by s.stno,s.name
    -> having count(distinct c.cno) = (select count(*) from courses where cr=4);
+-----------------+
| name            |
+-----------------+
| Edwards P. David |
| Mixon Leatha    |
+-----------------+
2 rows in set (0.01 sec)
```

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

```
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 and i.empno = a.empno
    -> where s.stno % 2 =1;
+-----------------+
| name            |
+-----------------+
| Edwards P. David |
| Grogan A. MAry  |
| Lewis Jerry     |
+-----------------+
```

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

```
mysql> select distinct s.name from students s
    -> JOIN grades g1 on s.stno = g1.stno
    -> JOIN courses c1 on g1.cno = c1.cno and c1.cno = "cs210"
    -> JOIN grades g2 on s.stno = g2.stno
    -> JOIN courses c2 on g2.cno = c2.cno and c2.cno = "cs310"
    -> where s.stno % 2 =1;
+-------------+
| name        |
+-------------+
| Mixon Leatha |
+-------------+
```

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

```
mysql> select distinct s.name,s.stno from students s
    -> join advising a on s.stno = a.stno
    -> join instructors i on a.empno = i.empno
    -> where i.emp_rank <> 'Professor' and s.stno % 2 =1;
+--------------+------+
| name         | stno |
+--------------+------+
| Lewis Jerry | 5571 |
+--------------+------+
```

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

```
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 advising a on s.stno = a.stno
    -> JOIN instructors i2 on a.empno = i2.empno
    -> where i1.roomno = i2.roomno and i1.empno != i2.empno and s.stno%2=1;
+-------------+
| name        |
+-------------+
| Will Samuel |
+-------------+
```

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

```
mysql> select g.cno from grades g
    -> join students s on g.stno = s.stno
    -> where s.stno%2=1 group by g.cno
    -> having count(distinct g.stno) = 2 ;
+-------+
| cno   |
+-------+
| cs110 |
+-------+
```

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

```
mysql> select s1.name from students s1
    -> where s1.stno % 2 =1 AND NOT EXISTS(
    -> select 1 from students s2
    -> where s1.city = s2.city and s1.stno <> s2.stno);
+------------------+
| name             |
+------------------+
| Edwards P. David |
| Grogan A. MAry   |
| Lewis Jerry      |
+------------------+
```

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

```
mysql> select g.cno from grades g
    -> JOIN students s on g.stno = s.stno
    -> JOIN instructors i on g.empno and i.empno
    -> where s.stno % 2 =1 and s.city = 'Boston' and i.emp_rank='Assoc.Prof.';
Empty set (0.01 sec)
```

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

```
mysql> select i.name,i.telno from instructors i
    -> join grades g on i.empno = g.empno
    -> join students s on g.stno = s.stno
    -> where s.stno%2=1 and s.city = 'Boston';
Empty set (0.00 sec)
```

11. Find names of students who took every course taken by Richard Pierce.

```
mysql> select distinct s1.name from students s1
    -> join grades g1 on s1.stno = g1.stno
    -> join courses c1 on g1.cno = c1.cno
    -> where s1.stno%2=1
    -> AND NOT EXISTS (
    -> select 1 from grades g2
    -> join students s2 on g2.stno=s2.stno
    -> join courses c2 on g2.cno = c2.cno
    -> where s2.name = 'Richard Pierce' and c2.cno = c1.cno
    -> AND NOT EXISTS(
    -> select 1 from grades g3 where g3.stno = s1.stno and g3.cno = c2.cno));
+-----------------+
| name            |
+-----------------+
| Edwards P. David |
| Grogan A. MAry   |
| Mixon Leatha     |
| Lewis Jerry      |
+-----------------+
```

12. Find the names of students who took only one course.

```
mysql> select s.name from students s
    -> join grades g on s.stno = g.stno
    -> group by s.stno,s.name
    -> having count(g.cno) = 1 and s.stno % 2 =1;
+-----------------+
| name            |
+-----------------+
| Grogan A. MAry  |
+-----------------+
```

13. Find the names of instructors who teach no course.

```
mysql> select i.name from instructors i
    -> left join grades g on i.empno = g.empno
    -> where g.cno is NULL;
+---------------+
| name          |
+---------------+
| Davis William |
+---------------+
```

14. Find the names of the instructors who taught only one course during the spring semester of 2001.

```
mysql> select distinct i.name from instructors i
    -> join grades g on i.empno = g.empno
    -> where g.sem = 'Spring' and g.year = 2001
    -> group by i.empno , i.name
    -> having count(g.cno)=1;
Empty 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
    -> where s.stno % 2 = 0
    -> group by s.stno,s.name
    -> having count(distinct c.cr)=1 and max(c.cr)=4;
Empty set (0.01 sec)
```

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

```
mysql> select s.name from students s where s.stno%2 =0
    -> AND 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         |
+--------------+
| McLane Sandy |
| Noval Roland |
+--------------+
```

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

```
mysql> select s.name,c.cno from students s
    ->  join grades g on s.stno = g.stno
    -> join courses c on g.cno = c.cno
    -> where s.stno % 2 = 0 and (c.cno = "cs210" or c.cno = "cs310");
+---------------+-------+
| name          | cno   |
+---------------+-------+
| Pierce Richard | cs210 |
| Prior Lorraine | cs210 |
| Prior Lorraine | cs310 |
+---------------+-------+
```

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
    -> join grades g1 on s.stno = g1.stno
    -> where s.stno%2 =0 and g1.cno = "cs210" and g1.grade > (
    -> select max(g2.grade) from grades g2 where g2.cno = "cs310")
    -> and s.stno not in (
    -> select distinct 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 g.cno from grades g
    -> join students s on g.stno = s.stno
    -> where s.stno%2 =0
    -> group by g.cno
    -> having count(distinct g.stno)>=2;
+-------+
| cno   |
+-------+
| cs110 |
| cs210 |
| cs240 |
+-------+
```

```
mysql> select g.cno from grades g
    -> join students s on g.stno = s.stno
    -> where s.stno%2 =0
    -> group by g.cno
    -> having count(distinct g.stno)>=3;
+-------+
| cno   |
+-------+
| cs110 |
| cs240 |
+-------+
```

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 s.stno%2=0 and g.cno = "cs210" and g.grade = (
    -> select max(g1.grade) from grades g1 where g1.cno = "cs210");
+----------------+
| name           |
+----------------+
| Pierce Richard |
+----------------+
```

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 i1.name from instructors i1
    -> join grades g1 on i1.empno = g1.empno
    -> join grades g2 on g1.stno = g2.stno
    -> join instructors i2 on g2.empno = i2.empno
    -> join students s on g1.stno = s.stno
    -> where i2.emp_rank = 'Assoc.Prof.' and s.stno % 2 = 0;
+--------------+
| name         |
+--------------+
| Evana Robert |
| Sawyer Kathy |
| Exxon George |
| Will Samuel  |
+--------------+
```

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

```
mysql> select min(grade) as lowest_grade from grades g
    -> join students s on g.stno = s.stno
    -> where sem="Spring" and year=2003 and s.stno%2=0;
+--------------+
| lowest_grade |
+--------------+
|           70 |
+--------------+
```

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).

```
mysql> select distinct s.name from students s where s.stno%2 = 0 and not exists(
    -> select 1 from courses c where not exists(
    -> select 1 from grades g where g.stno = s.stno and g.cno = c.cno and exists(
    -> select 1 from instructors i where i.empno = g.empno
    -> and i.name = 'Evans Robert')));
Empty set (0.00 sec)
```

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

```
mysql> select distinct s.name from students s
    -> join advising a on s.stno = a.stno where s.stno%2=0 and not exists(
    -> select 1 from grades g where g.stno = s.stno and g.empno = a.empno);
+---------------+
| name          |
+---------------+
| McLane Sandy  |
| Noval Roland  |
| Pierce Richard |
| Rawlings Jerry |
+---------------+
```

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

```
mysql> select distinct s.name from students s
    -> where s.stno%2=0 and not exists (
    -> select 1 from grades g where g.stno = s.stno and g.grade >= 60);
+-------------+
| name        |
+-------------+
| McLane Sandy |
+-------------+
```

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

```
mysql> select max(g.grade) as highest_grade from grades g
    -> join students s on g.stno = s.stno
    -> where s.stno%2=0 and g.cno != "cs110";
+---------------+
| highest_grade |
+---------------+
|           100 |
+---------------+
```

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

```
mysql> select s.name from students s
    -> left join advising a on s.stno = a.stno
    -> where s.stno%2=0 and a.empno is null;
Empty set (0.00 sec)
```

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

```
mysql> select distinct c.cname from courses c
    -> join grades g on c.cno = g.cno
    -> join students s on g.stno = s.stno
    -> where s.stno%2 = 0 and s.state != 'MA';
+----------------------+
| cname                |
+----------------------+
| Software engineering |
+----------------------+
```