ADBMS Practical 2

Using practical 1

1. Count the customers with grades above New York's average.

```
mysql> select count(customer_id) from customer where grade > (select avg(grade) from customer where city = 'new york');
+--------------------+
| count(customer_id) |
+--------------------+
|                  5 |
+--------------------+
1 row in set (0.00 sec)
```

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

```
mysql> select salesman_id,name from salesman where salesman_id in (select salesman_id from customer group by salesman_id
 having count(customer_id)>1);
+-------------+------------+
| 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 = 5006;
Query OK, 1 row affected (0.00 sec)

mysql> delete from customer where salesman_id = 5006;
Query OK, 1 row affected (0.01 sec)

mysql> delete from salesman where salesman_id = 5006;
Query OK, 1 row affected (0.00 sec)
```

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

Actor

```
mysql> CREATE TABLE ACTOR (
    -> ACT_ID INT (3) PRIMARY KEY,
    -> ACT_NAME VARCHAR (20),
    -> ACT_GENDER CHAR (1)
    -> )
```

```
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 (0.00 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)
```

Director

mysql> CREATE TABLE DIRECTOR (

-> DIR_ID INT (3) PRIMARY KEY,

-> DIR_NAME VARCHAR (20),

-> DIR_PHONE INT (10)

->);

```
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)
```

```
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)
```

Movies

mysql> CREATE TABLE MOVIES (

-> MOV_ID INT (4) PRIMARY KEY,

-> MOV_TITLE VARCHAR (25),

-> MOV_YEAR INT (4),

-> MOV_LANG VARCHAR (12),

-> DIR_ID INT (3),

-> FOREIGN KEY (DIR_ID) REFERENCES DIRECTOR (DIR_ID)

->);

```
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_LANG  | varchar(12) | YES  |     | NULL    |       |
| DIR_ID    | int         | YES  | MUL | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
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)
```

Movie_cast

mysql> CREATE TABLE MOVIE_CAST (

-> ACT_ID INT (3) ,

-> MOV_ID INT (4) ,

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

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

Rating

```
mysql> CREATE TABLE RATING (
    -> MOV_ID INT (4),
    -> REV_STARS VARCHAR (25),
    -> PRIMARY KEY (MOV_ID),
    -> FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID)
    -> )
```

```
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)
```

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

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 mov_id,mov_title from movies where mov_id in (select mov_id from movie_cast where act_id in (select act_id
 from movie_cast group by act_id having count(act_id)>=2));
+--------+------------+
| mov_id | mov_title  |
+--------+------------+
|   1001 | BAHUBALI-2 |
|   1002 | BAHUBALI-1 |
|   1003 | AKASH      |
+--------+------------+
3 rows in set (0.00 sec)
```

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

```
mysql> select movies.mov_title,movie_cast.act_id from movie_cast inner join movies on movies.mov_id = movie_cast.mov_id
where mov_year between 2000 and 2015;
+------------+--------+
| mov_title  | act_id |
+------------+--------+
| BAHUBALI-1 |    301 |
| BAHUBALI-1 |    303 |
| AKASH      |    303 |
| WAR HORSE  |    304 |
+------------+--------+
4 rows in set (0.00 sec)
```

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

```
mysql> select mov_title,mov_id from movies where mov_id in (select mov_id from rating where rev_stars >=4);
+------------+--------+
| mov_title  | mov_id |
+------------+--------+
| BAHUBALI-2 |   1001 |
| AKASH      |   1003 |
| WAR HORSE  |   1004 |
+------------+--------+
3 rows in set (0.00 sec)
```

```
mysql> select movies.mov_title,rating.rev_stars from rating inner join movies on movies.mov_id = rating.mov_id where rev
_stars >=4;
+------------+-----------+
| mov_title  | rev_stars |
+------------+-----------+
| BAHUBALI-2 | 4         |
| AKASH      | 5         |
| WAR HORSE  | 4         |
+------------+-----------+
3 rows in set (0.00 sec)
```

```
mysql> select movies.mov_title,rating.rev_stars from rating inner join movies on movies.mov_id = rating.mov_id where rev
_stars >=4 and rating.rev_stars = (select max(rev_stars) from rating where mov_id = rating.mov_id);
+-----------+-----------+
| mov_title | rev_stars |
+-----------+-----------+
| AKASH     | 5         |
+-----------+-----------+
1 row in set (0.00 sec)
```

```
mysql> select movies.mov_title,rating.rev_stars from rating inner join movies on movies.mov_id = rating.mov_id where rev
_stars >=4 order by mov_title;
+------------+-----------+
| mov_title  | rev_stars |
+------------+-----------+
| AKASH      | 5         |
| BAHUBALI-2 | 4         |
| WAR HORSE  | 4         |
+------------+-----------+
3 rows in set (0.00 sec)
```

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

```
mysql> update rating set rev_stars = 5 where mov_id = (select mov_id from movies where dir_id in (select dir_id from dir
ector where dir_name = 'steven spielberg'));
Query OK, 1 row affected (0.01 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)
```

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

Students

```
mysql> CREATE TABLE students (
    ->  stno INT PRIMARY KEY,
    ->  name VARCHAR(50),
    ->  addr VARCHAR(255),
    ->  city VARCHAR(50),
```

-> state VARCHAR(2),

-> zip VARCHAR(10)

->);

```
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 (0.01 sec)
```

Instructors

mysql> CREATE TABLE INSTRUCTORS (

-> empno INT PRIMARY KEY,

-> name VARCHAR(50),

-> ranks VARCHAR(20),

-> roomno VARCHAR(10),

-> telno VARCHAR(15)

->);

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

Courses

mysql> CREATE TABLE COURSES (

-> cno INT PRIMARY KEY,

-> cname VARCHAR(50),

-> cr INT,

-> cap INT

->)

```
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)
```

Grades

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

```
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)
```

Advising

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

```
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)
```

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

```
mysql> select students.stno,students.name,courses.cname from students inner join grades on students.stno = grades.stno i
nner join courses on grades.cno = courses.cno where courses.cr = 4;
+------+-----------------+--------------------------+
| stno | name            | cname                    |
+------+-----------------+--------------------------+
| 1011 | Edwards P. David | introduction to computing |
| 2661 | mixon leatha     | introduction to computing |
| 3566 | pierce richard   | introduction to computing |
| 5544 | rawlings jerry   | introduction to computing |
| 1011 | Edwards P. David | introduction to computing |
| 4022 | prior lorraine   | introduction to computing |
| 1011 | Edwards P. David | computer programming      |
| 2661 | mixon leatha     | computer programming      |
| 3566 | pierce richard   | computer programming      |
| 5571 | lewis jerry      | computer programming      |
| 4022 | prior lorraine   | computer programming      |
+------+-----------------+--------------------------+
11 rows in set (0.00 sec)
```

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

```
mysql> select students.name,grades.stno from students inner join grades on students.stno = grades.stno where grades.cno
=310 and 210;
+----------------+------+
| name           | stno |
+----------------+------+
| mixon leatha   | 2661 |
| prior lorraine | 4022 |
+----------------+------+
2 rows in set (0.00 sec)
```

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

```
mysql> select students.stno,students.name,grades.empno,advising.empno from students inner join grades on students.stno =
 grades.stno inner join advising on grades.stno = advising.stno where grades.empno = advising.empno;
+------+-----------------+-------+-------+
| stno | name            | empno | empno |
+------+-----------------+-------+-------+
| 1011 | Edwards P. David |    19 |    19 |
| 1011 | Edwards P. David |    19 |    19 |
| 2415 | grogan a. mary   |    19 |    19 |
| 4022 | prior lorraine   |   234 |   234 |
| 5571 | lewis jerry      |   234 |   234 |
+------+-----------------+-------+-------+
5 rows in set (0.00 sec)
```

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

```
mysql> select students.stno, students.name,advising.empno,instructors.ranks from students inner join advising on student
s.stno = advising.stno inner join instructors on advising.empno = instructors.empno where instructors.ranks != 'professo
r';
+------+---------------+-------+-------------+
| stno | name          | empno | ranks       |
+------+---------------+-------+-------------+
| 3442 | novak roland  |    56 | assoc. prof. |
| 3566 | pierce richard |   126 | assoc. prof. |
| 4022 | prior lorraine |   234 | assit. prof. |
| 5571 | lewis jerry   |   234 | assit. prof. |
+------+---------------+-------+-------------+
4 rows in set (0.00 sec)
```

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

```
mysql> select courses.cno from courses inner join grades on courses.cno = grades.cno group by grades.cno having count(gr
ades.stno)=2;
+-----+
| cno |
+-----+
| 410 |
| 310 |
+-----+
2 rows in set (0.00 sec)
```

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

```
mysql> select name,city from students where city not in (select city from students group by city having count(city)>1);
+------------------+------------+
| name             | city       |
+------------------+------------+
| Edwards P. David | newton     |
| grogan a. mary   | malden     |
| novak roland     | nashua     |
| lewis jerry      | providence |
+------------------+------------+
4 rows in set (0.00 sec)
```

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

```
mysql> select grades.cno,students.city,instructors.ranks from students inner join grades on students.stno = grades.stno
inner join instructors on grades.empno = instructors.empno where students.city='boston' and instructors.ranks = 'assoc.
prof.';
+------+--------+--------------+
| cno  | city   | ranks        |
+------+--------+--------------+
|  240 | boston | assoc. prof. |
|  240 | boston | assoc. prof. |
+------+--------+--------------+
2 rows in set (0.00 sec)
```

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

```
mysql> select instructors.telno from instructors inner join grades on instructors.empno = grades.empno inner join studen
ts on grades.stno = students.stno where students.city = 'boston';
+-------+
| telno |
+-------+
| 9101  |
| 7122  |
| 5110  |
| 7024  |
| 7122  |
| 5110  |
+-------+
6 rows in set (0.00 sec)
```

9. Find names of students who took every course taken by Will Samuel.

```
mysql> select students.name, courses.cname from students inner join grades on students.stno = grades.stno inner join cou
rses on grades.cno = courses.cno inner join instructors on grades.empno = instructors.empno where instructors.name = 'wi
ll samuel';
+----------------+----------------------+
| name           | cname                |
+----------------+----------------------+
| novak roland   | software engineering |
| lewis jerry    | software engineering |
| mixon leatha   | data structure       |
| prior lorraine | data structure       |
+----------------+----------------------+
4 rows in set (0.00 sec)
```

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

```
mysql> select instructors.name,grades.sem, grades.year from instructors inner join grades on instructors.empno = grades.
empno where grades.sem = 'spring' and grades.year = 2001;
+--------------+--------+------+
| name         | sem    | year |
+--------------+--------+------+
| exxon george | spring | 2001 |
+--------------+--------+------+
1 row in set (0.00 sec)
```

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

```
mysql> select stno,name from students where stno in (select stno from grades group by stno having count(cno) = 1);
+------+----------------+
| stno | name           |
+------+----------------+
| 2415 | grogan a. mary |
| 3442 | novak roland   |
+------+----------------+
2 rows in set (0.00 sec)
```

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

```
mysql> select empno,name from instructors where empno not in (select empno from grades group by empno);
+-------+---------------+
| empno | name          |
+-------+---------------+
|   126 | davis william |
+-------+---------------+
1 row in set (0.00 sec)
```