1.Create a view called ***TNS*** containing title-name-stars triples, where the movie (title) was reviewed by a reviewer (name) and received the rating (stars). Then referencing only view ***TNS*** and table *Movie*, write a SQL query that returns the lastest year of any movie reviewed by Chris Jackson. You may assume movie names are unique.

```
mysql> create or replace view tns as select title,name,stars from movie,reviewer,rating where movie.mid=rating.mid and reviewer.rid=rating.rid;
Query OK, 0 rows affected (0.12 sec)

mysql> select *from tns;
+-------------------------+------------------+-------+
| title                   | name             | stars |
+-------------------------+------------------+-------+
| Gone with the Wind      | Sarah Martinez   |     2 |
| Gone with the Wind      | Sarah Martinez   |     4 |
| Snow White              | Daniel Lewis     |     4 |
| The Sound of Music      | Brittany Harris  |     2 |
| Raiders of the Lost Ark | Brittany Harris  |     4 |
| Raiders of the Lost Ark | Brittany Harris  |     2 |
| Gone with the Wind      | Mike Anderson    |     3 |
| The Sound of Music      | Chris Jackson    |     3 |
| E.T.                    | Chris Jackson    |     2 |
| Raiders of the Lost Ark | Chris Jackson    |     4 |
| Avatar                  | Elizabeth Thomas |     3 |
| Snow White              | Elizabeth Thomas |     5 |
| Avatar                  | James Cameron    |     5 |
| E.T.                    | Ashley White     |     3 |
+-------------------------+------------------+-------+
14 rows in set (0.06 sec)

mysql> select tns.title,year from tns,movie where tns.name='Chris Jackson' and tns.title=movie.title order by year desc limit 1;
+-------+------+
| title | year |
+-------+------+
| E.T.  | 1982 |
+-------+------+
1 row in set (0.01 sec)
```

2. Referencing view ***TNS*** from Exercise 1 and no other tables, create a view ***RatingStats*** containing each movie title that has at least one rating, the number of ratings it received, and its average rating. Then referencing view ***RatingStats*** and no other tables, write a SQL query to find the title of the highest-average-rating movie with at least three ratings.

```
mysql> select title from ratingstats where average_rating in (select max(average_rating) from ratingstats where title in (select title from ratingstats where number_of_ratings>=3));
+-------------------------+
| title                   |
+-------------------------+
| Raiders of the Lost Ark |
+-------------------------+
1 row in set (0.01 sec)

mysql>
```

```
mysql> create or replace view ratingstats as select title,count(*) as number_of_ratings,avg(stars) as average_rating from tns group by title having count(*)>=1;
Query OK, 0 rows affected (0.07 sec)

mysql> select *from ratingstats;
+-------------------------+-------------------+----------------+
| title                   | number_of_ratings | average_rating |
+-------------------------+-------------------+----------------+
| Gone with the Wind      |                 3 |         3.0000 |
| Snow White              |                 2 |         4.5000 |
| The Sound of Music      |                 2 |         2.5000 |
| Raiders of the Lost Ark |                 3 |         3.3333 |
| E.T.                    |                 2 |         2.5000 |
| Avatar                  |                 2 |         4.0000 |
+-------------------------+-------------------+----------------+
6 rows in set (0.01 sec)
```

3. Create a view ***Favorites*** containing rID-mID pairs, where the reviewer with rID gave the movie with mID the highest rating he or she gave any movie. Then referencing only view ***Favorites*** and tables ***Movie*** and ***Reviewer***, write a SQL query to return reviewer-reviewer-movie triples where the two (different) reviewers have the movie as their favorite. Return each pair once, i.e., don't return a pair and its inverse.

```
mysql> create or replace view favourites as select rid,mid from rating where (rid,stars) in (select rid,max(stars) as ma from rating group by rid);
Query OK, 0 rows affected (0.06 sec)

mysql>  select *from favourites;
+------+------+
| rid  | mid  |
+------+------+
|  201 |  101 |
|  202 |  106 |
|  203 |  108 |
|  204 |  101 |
|  205 |  108 |
|  206 |  106 |
|  207 |  107 |
|  208 |  104 |
+------+------+
8 rows in set (0.02 sec)

mysql> select r.name,a.name,title from favourites as f inner join favourites as q on f.mid=q.mid and f.rid!=q.rid inner join reviewer as r on f.rid=r.rid inner join reviewer as a on q.rid=a.rid inner join movie
as m on m.mid=f.mid and q.rid<f.rid;
+-----------------+----------------+------------------------+
| name            | name           | title                  |
+-----------------+----------------+------------------------+
| Mike Anderson   | Sarah Martinez | Gone with the Wind     |
| Elizabeth Thomas | Daniel Lewis   | Snow White             |
| Chris Jackson   | Brittany Harris | Raiders of the Lost Ark |
+-----------------+----------------+------------------------+
3 rows in set (0.00 sec)

mysql>
```

4. Create a view ***REVIEW*** *containing* reviewer name, movie title, stars, and ratingDate. Also, sort the data, first by reviewer name, then by movie title, and lastly by number of stars.

```
mysql> create or replace view review as  select name,title,stars,ratingdate from reviewer,movie,rating where movie.mid=rating.mid and reviewer.rid=rating.rid order by name,title,stars;
Query OK, 0 rows affected (0.04 sec)

mysql> select *from review
    -> ^C
mysql> select *from review;
+-----------------+------------------------+-------+------------+
| name            | title                  | stars | ratingdate |
+-----------------+------------------------+-------+------------+
| Ashley White    | E.T.                   |     3 | 2011-01-02 |
| Brittany Harris | Raiders of the Lost Ark |     2 | 2011-01-30 |
| Brittany Harris | Raiders of the Lost Ark |     4 | 2011-01-12 |
| Brittany Harris | The Sound of Music     |     2 | 2011-01-20 |
| Chris Jackson   | E.T.                   |     2 | 2011-01-22 |
| Chris Jackson   | Raiders of the Lost Ark |     4 | NULL       |
| Chris Jackson   | The Sound of Music     |     3 | 2011-01-27 |
| Daniel Lewis    | Snow White             |     4 | NULL       |
| Elizabeth Thomas | Avatar                 |     3 | 2011-01-15 |
| Elizabeth Thomas | Snow White             |     5 | 2011-01-19 |
| James Cameron   | Avatar                 |     5 | 2011-01-20 |
| Mike Anderson   | Gone with the Wind     |     3 | 2011-01-09 |
| Sarah Martinez  | Gone with the Wind     |     2 | 2011-01-22 |
| Sarah Martinez  | Gone with the Wind     |     4 | 2011-01-27 |
+-----------------+------------------------+-------+------------+
14 rows in set (0.02 sec)
```

5. Create a view ***stars*** containing title,For each movie that has at least one rating, find the highest number of stars that movie received. Return the movie title and number of stars. Sort by movie title.

```
mysql> create or replace view stars as select title,max(stars) from movie,rating where movie.mid=rating.mid group by rating.mid order by movie.title;
Query OK, 0 rows affected (0.07 sec)

mysql> select *from stars;
+-------------------------+-------------+
| title                   | max(stars) |
+-------------------------+-------------+
| Avatar                  |           5 |
| E.T.                    |           3 |
| Gone with the Wind      |           4 |
| Raiders of the Lost Ark |           4 |
| Snow White              |           5 |
| The Sound of Music      |           3 |
+-------------------------+-------------+
6 rows in set (0.01 sec)

mysql>
```

6. Create a view ***pairs*** *containing names of both reviewers.* For all pairs of reviewers such that both reviewers gave a rating to the same movie, return the names of both reviewers. Eliminate duplicates, don't pair reviewers with themselves, and include each pair only once. For each pair, return the names in the pair in alphabetical order.

```
mysql> create or replace view pairs as select distinct r1.name as reviewer_1,r2.name as reviewer_2 from rating as ra1,rating as ra2,reviewer as r1,reviewer as r2 where ra1.mid=ra2.mid and ra1.rid=r1.rid and ra2.
rid=r2.rid and r1.name<r2.name order by r1.name,r2.name;
Query OK, 0 rows affected (0.06 sec)

mysql> select *from pairs;
+-----------------+-----------------+
| reviewer_1      | reviewer_2      |
+-----------------+-----------------+
| Ashley White    | Chris Jackson   |
| Brittany Harris | Chris Jackson   |
| Daniel Lewis    | Elizabeth Thomas |
| Elizabeth Thomas | James Cameron  |
| Mike Anderson   | Sarah Martinez  |
+-----------------+-----------------+
5 rows in set (0.01 sec)

mysql>
```

7. Create a view ***director*** *containing title, director.* Some directors directed more than one movie. For all such directors, return the titles of all movies directed by them, along with the director name. Sort by director name, then movie title.

```
mysql> create or replace view director as select title,director from movie where director=any(select director from movie group by director having count(*)>1) order by director,title;
Query OK, 0 rows affected (0.11 sec)

mysql> select *from director;
+-------------------------+-----------------+
| title                   | director        |
+-------------------------+-----------------+
| Avatar                  | James Cameron   |
| Titanic                 | James Cameron   |
| E.T.                    | Steven Spielberg |
| Raiders of the Lost Ark | Steven Spielberg |
+-------------------------+-----------------+
4 rows in set (0.01 sec)
```

8. Create a view **HighLow** containing title, For each movie, return the title and the 'rating spread', that is, the difference between highest and lowest ratings given to that movie. Sort by rating spread from highest to lowest, then by movie title.

```
mysql> create or replace view highlow as select title,max(stars)-min(stars) as ratingspread from movie,rating where movie.mid=rating.mid group by rating.mid order by ratingspread desc,title asc;
Query OK, 0 rows affected (0.10 sec)

mysql> select *from highlow;
+-------------------------+--------------+
| title                   | ratingspread |
+-------------------------+--------------+
| Avatar                  |            2 |
| Gone with the Wind      |            2 |
| Raiders of the Lost Ark |            2 |
| E.T.                    |            1 |
| Snow White              |            1 |
| The Sound of Music      |            1 |
+-------------------------+--------------+
6 rows in set (0.00 sec)

mysql>
```