Course Number and Name: CSE 4308 Database Management Systems Lab	
Student Name: Nafisa Maliyat	**Student ID:** 200042133
Report Submission Date: September, 2022	**Name of Lab Instructor:** Md. Bakhtiar Hasan, Lecturer, CSE Zannatun Naim Sristy, Lecturer, CSE

Lab 4: Advanced Data Manipulation

Overview:

This lab required us to perform queries on a given .sql file. The queries required multiple sub queries at times to get the result. The same result was also obtained using two methods at times, as some of the problem statements specified.
On the next pages, I have mentioned the following :

- the problem statement,
- the problem analysis,
- problems faced (if any) during solution of the tasks, and
- the queries and their results on the SQL command line.

Task 1

Problem Statement:

Find the name of the actors/actresses that are also directors (with and without 'intersect' clause).

Analysis of the problem:

The problem statement required us to find the people that are both actors and directors i.e. the people that whose data is in both actor and director table. Using intersect between both tables is one way of doing it. Writing conditions to match the first and last names of the entries in both tables will produce the same result as using 'intersect' clause, given that there is no director and actor who are different people but has the same full name.

Any problems faced and how it was solved:

There were no problems faced since the query was straightforward and simple.

Results:

with intersect:

```
SQL> SELECT DIRECTOR.DIR_FIRSTNAME, DIRECTOR.DIR_LASTNAME
  2  FROM DIRECTOR
  3  INTERSECT
  4  SELECT ACT_FIRSTNAME, ACT_LASTNAME
  5  FROM ACTOR;

DIR_FIRSTNAME        DIR_LASTNAME
-------------------- --------------------
Kevin                Spacey
Orson                Welles
Woody                Allen
```

without intersect :

```
SQL> SELECT DIRECTOR.DIR_FIRSTNAME, DIRECTOR.DIR_LASTNAME
  2  FROM DIRECTOR, ACTOR
  3  WHERE DIRECTOR.DIR_FIRSTNAME = ACTOR.ACT_FIRSTNAME
  4        AND DIRECTOR.DIR_LASTNAME = ACTOR.ACT_LASTNAME;

DIR_FIRSTNAME        DIR_LASTNAME
-------------------- --------------------
Woody                Allen
Kevin                Spacey
Orson                Welles
```

Task 2

Problem Statement:

Find the list of all the first names stored in the database.

Analysis of the problem:

The list of all the first names stored, which requires to select first names from actor and director table (since these are the only tables with first names) without selecting the duplicate entries in both tables. Use of 'distinct' ensured only unique names were selected.

Any problems faced and how it was solved:

There were no problems faced since the query was straightforward and simple.

Results:

```
SQL> (
  2      SELECT DISTINCT ACTOR.ACT_FIRSTNAME AS FIRST_NAME
  3      FROM ACTOR
  4      UNION
  5      SELECT DISTINCT DIRECTOR.DIR_FIRSTNAME AS FIRST_NAME
  6      FROM DIRECTOR
  7  )
  8  MINUS
  9  (
 10      SELECT DISTINCT DIRECTOR.DIR_FIRSTNAME
 11      FROM DIRECTOR, ACTOR
 12      WHERE DIRECTOR.DIR_FIRSTNAME = ACTOR.ACT_FIRSTNAME
 13      AND DIRECTOR.DIR_LASTNAME = ACTOR.ACT_LASTNAME
 14  );
```

```
FIRST_NAME
--------------------
Al
Alfred
Ali
Andrei
Bryan
Christian
Christopher
Claire
Danny
David
Deborah
```

```
FIRST_NAME
--------------------
Dev
Eddie
Ewan
F. Murray
Felicity
Frank
George
Gus
Harrison
Hayao
Jack
```

```
FIRST_NAME
--------------------
Jackie
James
Jennifer
John
Jon
Kate
Lana
Maggie
Mark
Michael
Milos
```

```
FIRST_NAME
--------------------
Nicole
Paul
Peter
Raoul
Richard
Ridley
Robert
Robin
Roman
Sam
Shelley
```

```
FIRST_NAME
--------------------
Sigourney
Stanley
Stephen
Susan
Tim

49 rows selected.
```

Task 3

Problem Statement:

Find the movie titles that did not receive any ratings (with and without 'minus' clause).

Analysis of the problem:

This required the entries in movie that are not in the rating table. Where minus was not allowed, 'not in' was used to produce the same result.

Any problems faced and how it was solved:

There were no problems faced since the query was straightforward and simple.

Results:

with minus:

```
SQL> SELECT MOV_ID
  2  FROM MOVIE
  3  WHERE MOV_ID NOT IN (SELECT MOV_ID FROM RATING);

    MOV_ID
----------
       932
       930
       927
       931
       929
       928
       926

7 rows selected.
```

without minus:

```
SQL> (SELECT MOV_ID
  2  FROM MOVIE)
  3  MINUS
  4  (SELECT MOV_ID
  5  FROM RATING);

    MOV_ID
----------
       926
       927
       928
       929
       930
       931
       932

7 rows selected.
```

Task 4

Problem Statement:

Find the average rating of all movies.

Analysis of the problem:

The result is the average of all the values of the attribute rev_stars in the rating table.

Any problems faced and how it was solved:

There were no problems faced since the query was straightforward and simple.

Results:

```
SQL> SELECT AVG(REV_STARS) AS AVG_RATING
  2  FROM RATING;

AVG_RATING
----------
       6.8
```

Task 5

Problem Statement:

Find the minimum rating for each movie and display them in descending order of rating.

Analysis of the problem:

First the group by is used on the mov_id attribute on rating and the aggregate function min was applied on rev_stars. Finally, order by was used to display them in descending order according to the minimum rev_stars.

Any problems faced and how it was solved:

There were no problems faced since the query was straightforward and simple.

Results:

```
SQL> SELECT MOV_ID, MIN(REV_STARS) AS MIN_RATING
  2  FROM RATING
  3  GROUP BY MOV_ID
  4  ORDER BY MIN_RATING DESC;
```

MOV_ID	MIN_RATING
933	7
924	6
917	6
913	5
909	5
916	5
903	5
905	4
904	4
901	4
902	4

MOV_ID	MIN_RATING
919	4
910	3
914	3
915	3
925	3
906	3
923	3
907	3
918	3
911	3
920	3

MOV_ID	MIN_RATING
921	3
922	2
908	2
912	1

26 rows selected.

Task 6

Problem Statement:

Find the last name of actors/actresses and the number of ratings received by the movies that they played a role in.

Analysis of the problem:

The actors' last names who had a movie that also had a rating and the number of rating of the movies of those actors were the objective.

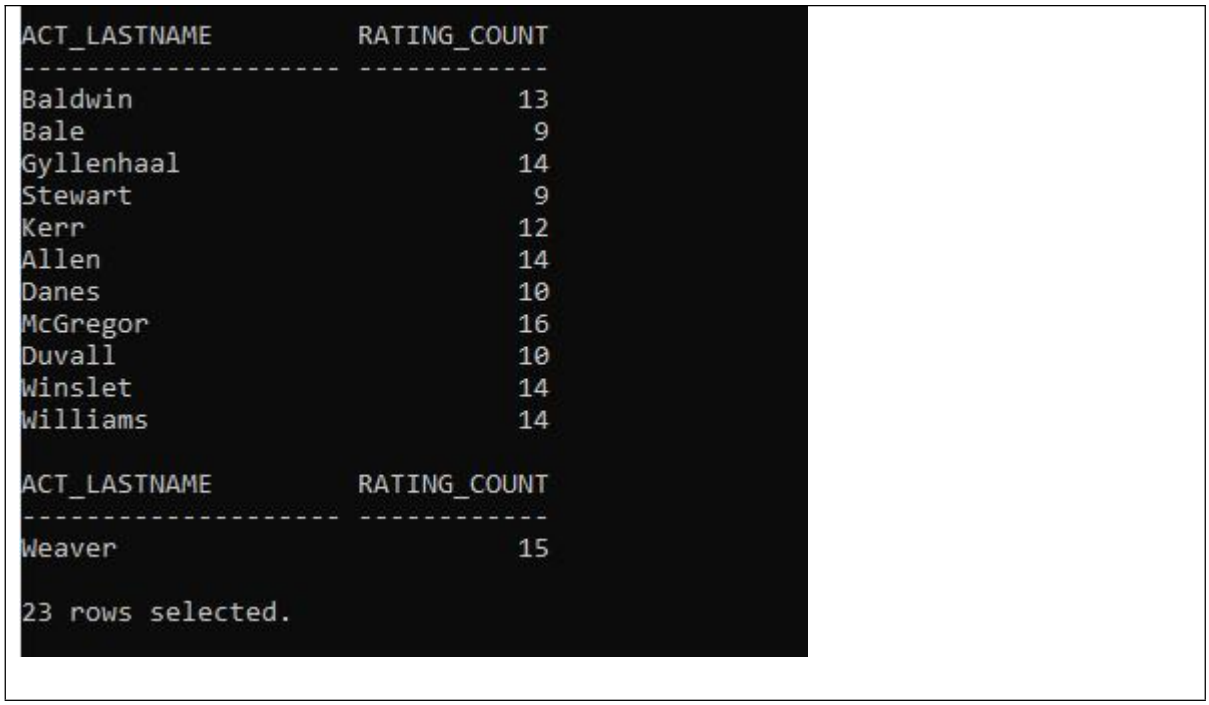
Any problems faced and how it was solved:

Since three tables were involved, knowing which data had to match was a bit confusing. This was solved using natural join to join all three tables to get a table containing actors who had movies that also had ratings. From there, group by was used to form groups according to act_id and the count of rev_stars for each act_id was easily obtained.

Results:

```
SQL> SELECT ACT_LASTNAME, COUNT(RATING.REV_STARS) AS RATING_COUNT
  2  FROM ((ACTOR NATURAL JOIN CASTS) NATURAL JOIN RATING)
  3  GROUP BY ACT_ID, ACT_LASTNAME;

ACT_LASTNAME         RATING_COUNT
-------------------- ------------
Spacey                         18
De Niro                        10
Abraham                        12
Kidman                         13
Wahlberg                       12
Robbins                        13
OToole                         11
Nicholson                      10
Voight                         11
Patel                           9
Ford                           13
```



```
ACT_LASTNAME         RATING_COUNT
-------------------- ------------
Baldwin                        13
Bale                            9
Gyllenhaal                     14
Stewart                         9
Kerr                           12
Allen                          14
Danes                          10
McGregor                       16
Duvall                         10
Winslet                        14
Williams                       14

ACT_LASTNAME         RATING_COUNT
-------------------- ------------
Weaver                         15

23 rows selected.
```

Task 7

Problem Statement:

Find the last name and average runtime of movies of different actors/actresses. Do not include any actor/actress who worked with 'James Cameron' (with and without 'having' clause).

Analysis of the problem:

Firstly, a distinct list of actors was used to get act_lastname and a sub query was used to find the average runtime of the movies performed by that actor. The sub query selected all the mov_time of the movie provided that the mov_id was associated with that particular actor in the casts table and an average function was used.
From these results, some of the rows were eliminated if found to be common with a second query which contained list of actors whose id were associated with the dir_id (in the direction table) that had name 'James Cameron' in the director table.
For having, the first query was grouped by act_id and the second query was in the having clause. In without having, the first query was made using matching of act_id from casts and act_id and where clause was used instead of having.

Any problems faced and how it was solved:

Using group by, act_lastname could not be selected because it was not within the group by function. This was solved using a dummy function max(act_lastname) which would return the same thing since the act_lastname in group formed by act_id was same.
The number of rows in the query without 'having' clause was 29 while the second was 28 and upon further inspection, a name (Spacey) was in the list twice. Using 'distinct' for this solved the problem.

Results:

without having:

```
SQL> SELECT DISTINCT ACT_LASTNAME, (SELECT AVG(MOV_TIME)
  2                                 FROM MOVIE, CASTS
  3                                 WHERE MOVIE.MOV_ID = CASTS.MOV_ID
  4                                 AND CASTS.ACT_ID = ACTOR.ACT_ID
  5                                ) AS AVG_RUNTIME
  6  FROM ACTOR, CASTS
  7  WHERE ACTOR.ACT_ID = CASTS.ACT_ID AND
  8  ACTOR.ACT_LASTNAME NOT IN
  9  (
 10      SELECT ACTOR.ACT_LASTNAME
 11      FROM ACTOR, CASTS, DIRECTION, DIRECTOR
 12      WHERE CASTS.ACT_ID = ACTOR.ACT_ID
 13      AND CASTS.MOV_ID = DIRECTION.MOV_ID
 14      AND DIRECTION.DIR_ID =  DIRECTOR.DIR_ID
 15      AND DIRECTOR.DIR_FIRSTNAME = 'JAMES'
 16      AND DIRECTOR.DIR_LASTNAME = 'CAMERON'
 17  );
```

ACT_LASTNAME	AVG_RUNTIME
Stewart	128
Voight	109
OToole	216
Baldwin	106
Jones	123
Winslet	194
Wahlberg	155
Weaver	137
Patel	120
Williams	126
Duvall	146

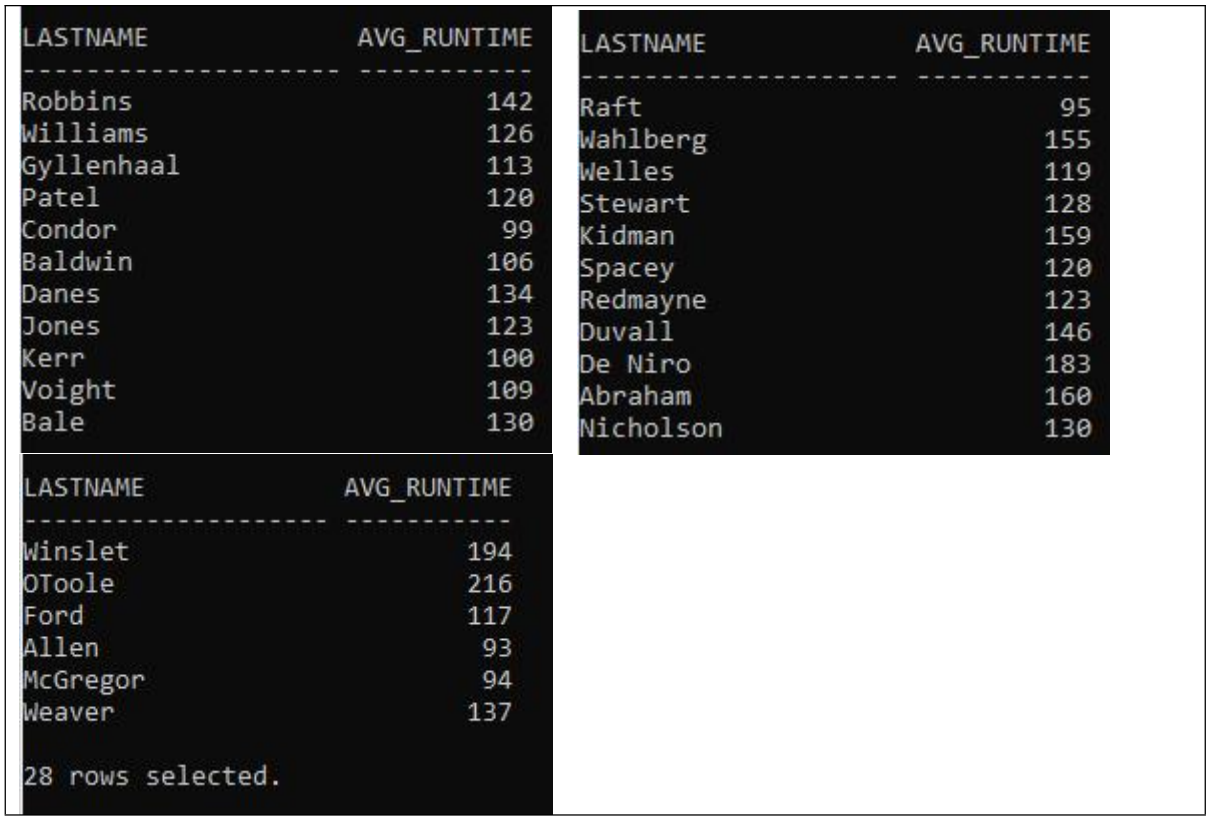
ACT_LASTNAME	AVG_RUNTIME
De Niro	183
Condor	99
Abraham	160
Ford	117
Raft	95
Bale	130
Gyllenhaal	113
Danes	134
Nicholson	130
Allen	93
Robbins	142

ACT_LASTNAME	AVG_RUNTIME
McGregor	94
Kidman	159
Spacey	120
Welles	119
Redmayne	123
Kerr	100

28 rows selected.

With having:

```
SQL> SELECT MAX(ACT_LASTNAME) AS LASTNAME, AVG(MOV_TIME) AS AVG_RUNTIME
  2  FROM ((ACTOR NATURAL JOIN CASTS)
  3          NATURAL JOIN MOVIE)
  4  GROUP BY ACT_ID
  5  HAVING ACT_ID NOT IN (SELECT ACT_ID
  6                        FROM ((ACTOR NATURAL JOIN CASTS)
  7                        NATURAL JOIN DIRECTION)
  8                        NATURAL JOIN DIRECTOR
  9                        WHERE DIR_FIRSTNAME = 'JAMES'
 10                        AND DIR_LASTNAME = 'CAMERON'
 11  );
```



LASTNAME	AVG_RUNTIME
Robbins	142
Williams	126
Gyllenhaal	113
Patel	120
Condor	99
Baldwin	106
Danes	134
Jones	123
Kerr	100
Voight	109
Bale	130

LASTNAME	AVG_RUNTIME
Raft	95
Wahlberg	155
Welles	119
Stewart	128
Kidman	159
Spacey	120
Redmayne	123
Duvall	146
De Niro	183
Abraham	160
Nicholson	130

LASTNAME	AVG_RUNTIME
Winslet	194
OToole	216
Ford	117
Allen	93
McGregor	94
Weaver	137

28 rows selected.

Task 8

Problem Statement:

Find the first name and last name of the director of the movie having the highest average rating (with and without 'all' clause).

Analysis of the problem:

First a list of average rating of each movie is produced from rating table. Next the maximum value of the average rating is obtained using max aggregate function. The list of first and last names of the directors who had directed a movie is obtained whose directed movie has that maximum average rating.
Using all, the average rating of each movie is checked to find the average rating that is greater or equal to all of the average rating .Without all, the average rating is matched against the max of the average rating.

Any problems faced and how it was solved:

Since there were multiple sub queries, it was difficult to organize the order. The problem was solved by thinking in steps and writing the smaller sub queries and keep nesting them one by one.

Results:

with all:

```
SQL> SELECT DIRECTOR.DIR_FIRSTNAME, DIRECTOR.DIR_LASTNAME
  2  FROM DIRECTOR, DIRECTION
  3  WHERE DIRECTION.DIR_ID = DIRECTOR.DIR_ID
  4  AND DIRECTION.MOV_ID IN
  5  (SELECT MOV_ID
  6          FROM RATING
  7          GROUP BY MOV_ID
  8          HAVING AVG(REV_STARS) = (SELECT MAX(AVG_RATING)
  9                                   FROM
 10                                   (
 11                                       SELECT MOV_ID, AVG(REV_STARS) AS AVG_RATING
 12                                       FROM RATING
 13                                       GROUP BY MOV_ID)
 14                                   )
 15  );

DIR_FIRSTNAME        DIR_LASTNAME
-------------------- --------------------
Stanley              Kubrick
```

without all:

```
SQL> SELECT DIRECTOR.DIR_FIRSTNAME, DIRECTOR.DIR_LASTNAME
  2  FROM DIRECTOR, DIRECTION
  3  WHERE DIRECTION.DIR_ID = DIRECTOR.DIR_ID
  4  AND DIRECTION.MOV_ID IN
  5  (SELECT MOV_ID
  6          FROM RATING
  7          GROUP BY MOV_ID
  8          HAVING AVG(REV_STARS) >= ALL (SELECT AVG_RATING
  9                                       FROM
 10                                       (
 11                                           SELECT MOV_ID, AVG(REV_STARS) AS AVG_RATING
 12                                           FROM RATING
 13                                           GROUP BY MOV_ID)
 14                                       )
 15  );

DIR_FIRSTNAME        DIR_LASTNAME
-------------------- --------------------
Stanley              Kubrick
```

Task 9

Problem Statement:

Find all the movie related information of movies acted and directed by the same person.

Analysis of the problem:

First the dir_id of those directors were selected who are also actors by matching their names. Next all movie related information is selected of those movies whose mov_id was associated with those dir_id (found by query in the first part) in the direction table.

Any problems faced and how it was solved:

Since there were many selections, the view of the obtained result was broken to accommodate all the information. This problem could not be solved.

Results:

```
SQL> SELECT MOV_ID, MOV_TITLE AS TITLE, MOV_YEAR AS YEAR,
  2  MOV_LANGUAGE AS LANG, MOV_RELEASEDATE AS RELEASEDATE,
  3  MOV_COUNTRY AS COUNTRY
  4  FROM MOVIE
  5  WHERE MOV_ID IN
  6  (
  7      SELECT MOV_ID
  8      FROM DIRECTION
  9      WHERE DIR_ID IN
 10          (
 11              SELECT DIRECTOR.DIR_ID
 12              FROM DIRECTOR, ACTOR
 13              WHERE DIRECTOR.DIR_FIRSTNAME = ACTOR.ACT_FIRSTNAME
 14              AND DIRECTOR.DIR_LASTNAME = ACTOR.ACT_LASTNAME
 15          )
 16  );
```

```
    MOV_ID TITLE                                                    YEAR
---------- ---------------------------------------------------- ----------
LANG                           RELEASEDA COUNTRY
------------------------------ --------- ------------------------------
       911 Annie Hall                                               1977
English                        20-APR-77 USA

       923 Beyond the Sea                                           2004
English                        26-NOV-04 UK

       932 Citizen Kane                                             1941
English                        05-SEP-41 USA
```

Task 10

Problem Statement:

Find the title and average rating of the movies that have average rating more than 7 (with and without using 'having' clause).

Analysis of the problem:

First the mov_id was chosen from rating table whose average rating was more than 7 and the mov_id was used to find the mov_title from movie table.
Using having, it was easily done by grouping the mov_id in the rating table and applying the condition using having.
Without having, the same thing was done with more steps. The mov_id and average rating was chosen after applying group by. Next only those mov_id was selected whose average rating matched the condition. This was done using where clause which could be used because the table produced in the subquery had the columns mov_id and avg_rating.

Any problems faced and how it was solved:

At first, it seemed that group by could not be used since having could not be used but later, with the help of online resources, it was discovered that if the average rating and movie id was considered as a table, conditions could be imposed on average rating using 'where' clause. Thus nested queries were used to accomplish this.

Results:

with having:

```
SQL> SELECT MOV_TITLE
  2  FROM MOVIE
  3  WHERE MOV_ID IN
  4  (
  5       SELECT MOV_ID
  6       FROM RATING
  7       GROUP BY MOV_ID
  8       HAVING AVG(REV_STARS) > 7
  9  );

MOV_TITLE
--------------------------------------------------
Deliverance
Good Will Hunting
The Shining
Chinatown
The Shawshank Redemption
Eyes Wide Shut
Avatar
Braveheart

8 rows selected.
```

without having:

```
SQL> SELECT MOV_TITLE
  2  FROM MOVIE
  3  WHERE MOV_ID IN
  4  (
  5      SELECT MOV_ID
  6      FROM (SELECT AVG_RATING, MOV_ID
  7            FROM (SELECT AVG(REV_STARS) AS AVG_RATING, MOV_ID
  8                  FROM RATING
  9                  GROUP BY MOV_ID)
 10            WHERE AVG_RATING>7)
 11  );

MOV_TITLE
----------------------------------------------------
Deliverance
Good Will Hunting
The Shining
Chinatown
The Shawshank Redemption
Eyes Wide Shut
Avatar
Braveheart

8 rows selected.
```

Task 11

Problem Statement:

Find the title of the movies having average rating higher than the average rating of all the movies.

Analysis of the problem:

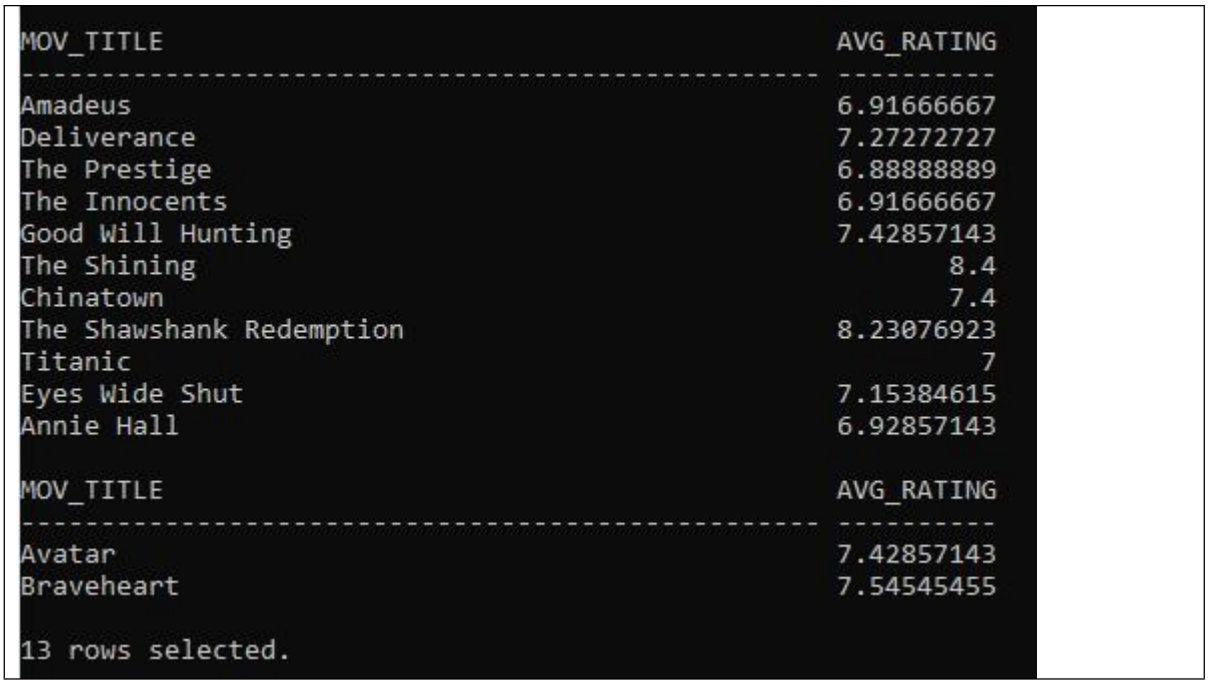
First the average rating of all movies was found (also done in task 4). This was used to produce a list of mov_id whose average rating was higher than this value.
Next the movie's title and average rating was selected. Average rating was selected using a sub-query and mov_id of the rating's mov_id and movie's mov_id was matched so average of the rating of a particular movie would be done and not all the movies.
From this list of movie name and average mov_id, only those were selected that had their mov_id in the list produced in the first step.

Any problems faced and how it was solved:

In this problem, selecting the average rating was the problem since using group by would not allow movie title or any other attribute to be selected that were not grouped by. This was solved using sub query and this method was then followed for the rest of the previous queries as needed.

Results:

```
SQL> SELECT MOVIE.MOV_TITLE, (SELECT AVG(REV_STARS)
  2                       FROM RATING
  3                       WHERE RATING.MOV_ID = MOVIE.MOV_ID) AS AVG_RATING
  4  FROM MOVIE
  5  WHERE MOV_ID IN
  6  (SELECT MOV_ID
  7  FROM RATING
  8  GROUP BY MOV_ID
  9  HAVING AVG(REV_STARS) > (SELECT AVG(REV_STARS)
 10                          FROM RATING));
```



```
MOV_TITLE                                          AVG_RATING
-------------------------------------------------- ----------
Amadeus                                            6.91666667
Deliverance                                        7.27272727
The Prestige                                       6.88888889
The Innocents                                      6.91666667
Good Will Hunting                                  7.42857143
The Shining                                               8.4
Chinatown                                                 7.4
The Shawshank Redemption                           8.23076923
Titanic                                                    7
Eyes Wide Shut                                     7.15384615
Annie Hall                                         6.92857143

MOV_TITLE                                          AVG_RATING
-------------------------------------------------- ----------
Avatar                                             7.42857143
Braveheart                                         7.54545455

13 rows selected.
```

Task 12

Problem Statement:

Find the title and average rating of the movies without using the group by statement.

Analysis of the problem:

From the movie and rating table, the mov_title (who had a rating) and their average rating was selected.
This was done without grouping by taking the same table rating twice and assigning them alias R and R2 where the mov_id matched in both table (which produces the effect of grouping) and this selected mov_id matched the mov_id from movie table so the correct mov_title was selected. From this query tables with average ratings of a particular mov_id was produced.
Finally only those mov_title was included with ratings present in the rating table otherwise, mov_title with no titles would show up with blank in average rating.

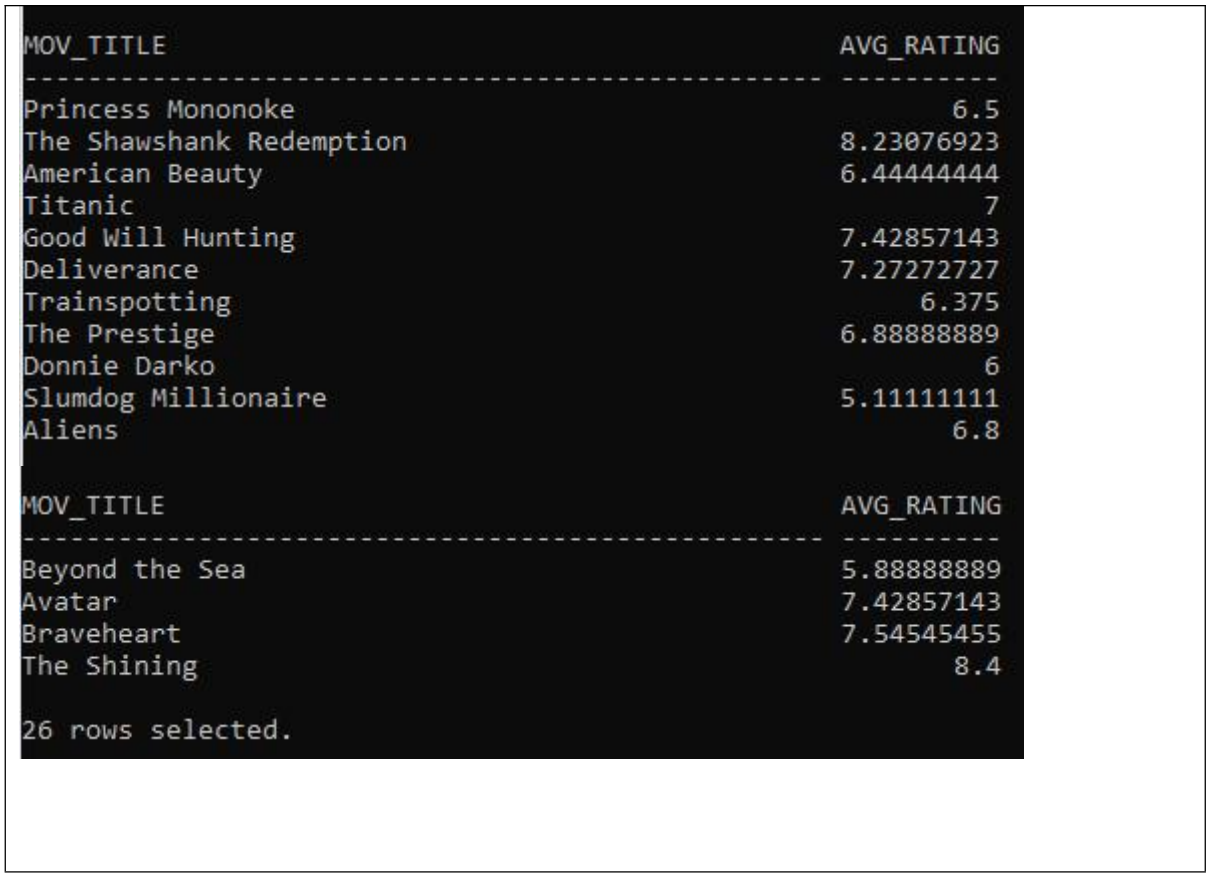
Any problems faced and how it was solved:

The problems regarding this task was how to do it without group by. Internet resources were used to solve the problem.

Results:

```
SQL> SELECT   MOVIE.MOV_TITLE, (SELECT AVG(R.REV_STARS)
  2                             FROM RATING R, RATING R2
  3                             WHERE R.MOV_ID = R2.MOV_ID
  4                             AND R.MOV_ID = MOVIE.MOV_ID) AS AVG_RATING
  5  FROM MOVIE
  6  WHERE MOVIE.MOV_ID IN (SELECT MOV_ID
  7                         FROM RATING);
```

MOV_TITLE	AVG_RATING
Vertigo	6.66666667
The Innocents	6.91666667
Lawrence of Arabia	6.63636364
The Deer Hunter	6.3
Amadeus	6.91666667
Blade Runner	6.23076923
Eyes Wide Shut	7.15384615
The Usual Suspects	6.07692308
Chinatown	7.4
Boogie Nights	6.16666667
Annie Hall	6.92857143



```
MOV_TITLE                                          AVG_RATING
-------------------------------------------------- ----------
Princess Mononoke                                         6.5
The Shawshank Redemption                           8.23076923
American Beauty                                    6.44444444
Titanic                                                    7
Good Will Hunting                                  7.42857143
Deliverance                                        7.27272727
Trainspotting                                          6.375
The Prestige                                       6.88888889
Donnie Darko                                               6
Slumdog Millionaire                                5.11111111
Aliens                                                   6.8

MOV_TITLE                                          AVG_RATING
-------------------------------------------------- ----------
Beyond the Sea                                     5.88888889
Avatar                                             7.42857143
Braveheart                                         7.54545455
The Shining                                              8.4

26 rows selected.
```

Task 13

Problem Statement:

Find the actresses with the same first name.

Analysis of the problem:

Only those act_firstname in the actor table were selected that appeared more than once. This was done by grouping the first names and checking if their count was more than one (if they appeared more than once).

Any problems faced and how it was solved:

There were no problems faced since the query was straightforward and simple.

Results:

```
SQL> SELECT ACT_FIRSTNAME
  2  FROM ACTOR
  3  WHERE ACT_GENDER = 'F'
  4  GROUP BY ACT_FIRSTNAME
  5  HAVING COUNT(ACT_FIRSTNAME)>1;

ACT_FIRSTNAME
--------------------
Kate
Jennifer
```

Task 14

Problem Statement:

Find the title and maximum rating of the movies that has at least 10 reviews and has a female actress. One of the reviewers of the movie should be 'Neal Wruck'. Do not include any movie that received less than 4 stars rating or any movies from directors that have did not direct more than one movie.

Analysis of the problem:

The problem can be broken into conditions:

- **Movie title and maximum rating of the movies that has at least 10 reviews**: using movie table and selecting the maximum rating of those movies that has count of rev_stars of at least 10. This is done using group by function to select the maximum rating and mov_id and then nesting it to select the rating of those movies that matches the mov_id of the mov_title selected.
- **Has a female actress**: the mov_id is checked to see if it is in the list of movies that had female actresses (using cast and actor table).
- **One of the reviewers of the movie should be 'Neal Wruck'**: the mov_id is checked to see if it is in the list of movies directed by Neal Wruck. (using rating table to find rev_id and checking corresponding rev_name in the reviewer table).
- **Do not include any movie that received less than 4 stars rating**: using 'not in' clause to ensure that mov_id is not in the list of those movies that has a less than 4 star rating (using rating table to produce a list of mov_id that has a less than 4 star rating).
- **Do not include any movies from directors that have did not direct more than one movie**: First those dir_id with a count of mov_id more than one are selected (using direction table and group by function). Next the movie id of those dir_id is obtained and the original mov_id is checked to ensure it is in that list.

Finally the mov_id that satisfies all of these (using 'and') and its corresponding maximum rating is the result.

Any problems faced and how it was solved:

The problem statement stated a lot of conditions. The interconnection of tables made it all a bit confusing to decide where to start. This was solved by breaking down the problem (as shown above) which made it easier to think.

Results:

```
SQL> --AT LEAST TEN REVIEWS
SQL> SELECT MOV_TITLE, (SELECT REV_RATING
  2                     FROM
  3                     (SELECT MAX(REV_STARS) AS REV_RATING, MOV_ID
  4                     FROM RATING
  5                     GROUP BY MOV_ID
  6                     HAVING COUNT(REV_STARS) >= 10)
  7                     WHERE MOV_ID = MOVIE.MOV_ID) AS MAX_RATING
  8  FROM MOVIE
  9  --FEMALE ACTRESS
 10  WHERE MOV_ID IN (SELECT MOV_ID
 11                   FROM CASTS
 12                   WHERE ACT_ID IN (SELECT ACT_ID
 13                   FROM ACTOR
 14                   WHERE ACT_GENDER='F'))
 15  --ONE REVIEWER NEAL WRUCK
 16  AND MOV_ID IN (SELECT MOV_ID
 17                   FROM RATING
 18                   WHERE REV_ID IN (SELECT REV_ID
 19                                    FROM REVIEWER
 20                                    WHERE REV_NAME = 'Neal Wruck'))
 21  --NOT ANY THAT RECEIVE LESS THAN FOUR STARS
 22  AND MOV_ID NOT IN (SELECT MOV_ID
 23                     FROM RATING
 24                     WHERE REV_STARS<4)
 25  --DIRECTOR DIRECTED AT LEAST ONE MOVIE
 26  AND MOV_ID IN (SELECT MOV_ID
 27                   FROM DIRECTION
 28                   WHERE DIR_ID IN (SELECT DIR_ID
 29                                    FROM DIRECTION
 30                                    GROUP BY DIR_ID
 31                                    HAVING COUNT(MOV_ID)>1 ));

MOV_TITLE                                          MAX_RATING
-------------------------------------------------- ----------
The Shining                                                10
```