Std.name : AHMED M. S. ALBREEM

Std.id : 210041258

COURSE.NO : CSE4308

DATE OF SOLUTION : 14-09-2023

ASSIGNMENT 5 OF DBMS LAB

The commend 1 of assignment 5

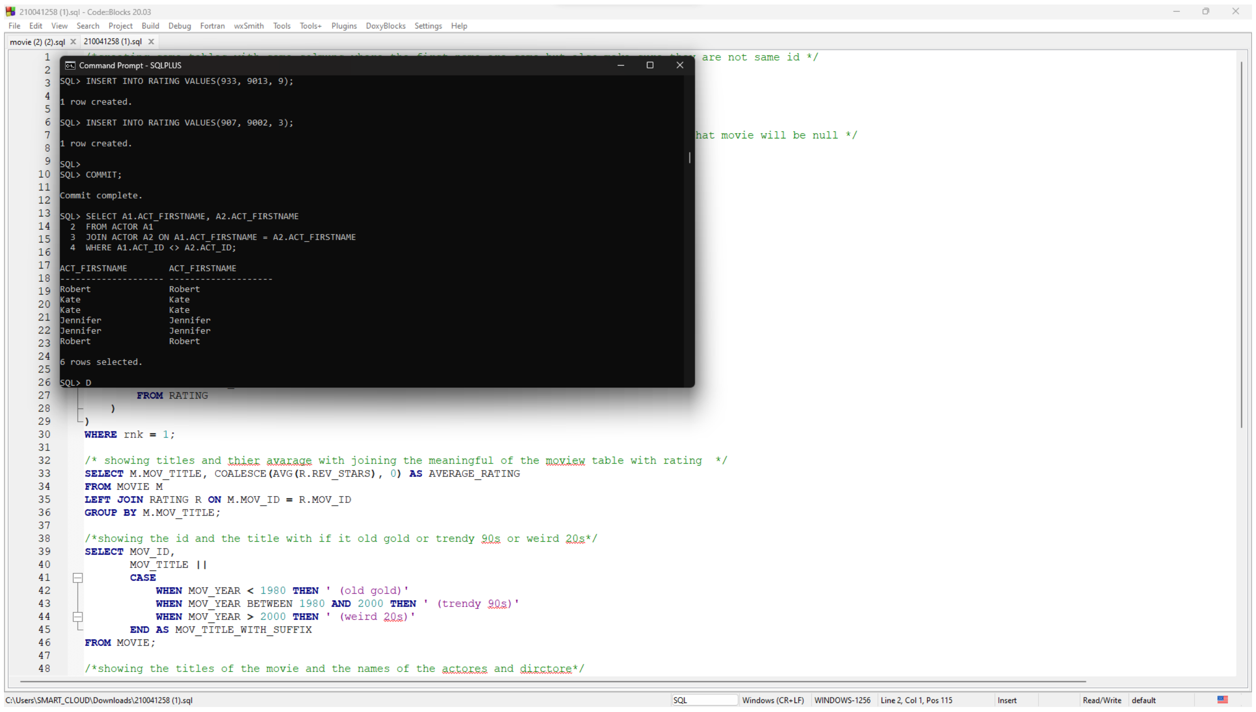
Commend Request : Duplicate Actor First Names:

Code: /*creating same tables with same colmuns where the first name are same but also make sure they are not same id */

```
SELECT A1.ACT_FIRSTNAME, A2.ACT_FIRSTNAME
FROM ACTOR A1
JOIN ACTOR A2 ON A1.ACT_FIRSTNAME = A2.ACT_FIRSTNAME
WHERE A1.ACT_ID <> A2.ACT_ID;
```

Explanation :This query retrieves pairs of actors with the same first name but different IDs.
It selects the ACT_FIRSTNAME from the ACTOR table for two instances, A1 and A2, and ensures that their ACT_ID values are not the same.

Screenshot:



```
SQL> INSERT INTO RATING VALUES(933, 9013, 9);

1 row created.

SQL> INSERT INTO RATING VALUES(907, 9002, 3);

1 row created.

SQL>
SQL> COMMIT;

Commit complete.

SQL> SELECT A1.ACT_FIRSTNAME, A2.ACT_FIRSTNAME
  2  FROM ACTOR A1
  3  JOIN ACTOR A2 ON A1.ACT_FIRSTNAME = A2.ACT_FIRSTNAME
  4  WHERE A1.ACT_ID <> A2.ACT_ID;

ACT_FIRSTNAME        ACT_FIRSTNAME
-------------------- --------------------
Robert               Robert
Kate                 Kate
Kate                 Kate
Jennifer             Jennifer
Jennifer             Jennifer
Robert               Robert

6 rows selected.

SQL> D
```

```
 are not same id */
hat movie will be null */
            FROM RATING
        )
)
WHERE rnk = 1;

/* showing titles and thier avarage with joining the meaningful of the moview table with rating  */
SELECT M.MOV_TITLE, COALESCE(AVG(R.REV_STARS), 0) AS AVERAGE_RATING
FROM MOVIE M
LEFT JOIN RATING R ON M.MOV_ID = R.MOV_ID
GROUP BY M.MOV_TITLE;

/*showing the id and the title with if it old gold or trendy 90s or weird 20s*/
SELECT MOV_ID,
       MOV_TITLE ||
       CASE
           WHEN MOV_YEAR < 1980 THEN ' (old gold)'
           WHEN MOV_YEAR BETWEEN 1980 AND 2000 THEN ' (trendy 90s)'
           WHEN MOV_YEAR > 2000 THEN ' (weird 20s)'
       END AS MOV_TITLE_WITH_SUFFIX
FROM MOVIE;

/*showing the titles of the movie and the names of the actores and dirctore*/
```

C:\Users\SMART_CLOUD\Downloads\210041258 (1).sql SQL Windows (CR+LF) WINDOWS-1256 Line 2, Col 1, Pos 115 Insert Read/Write default

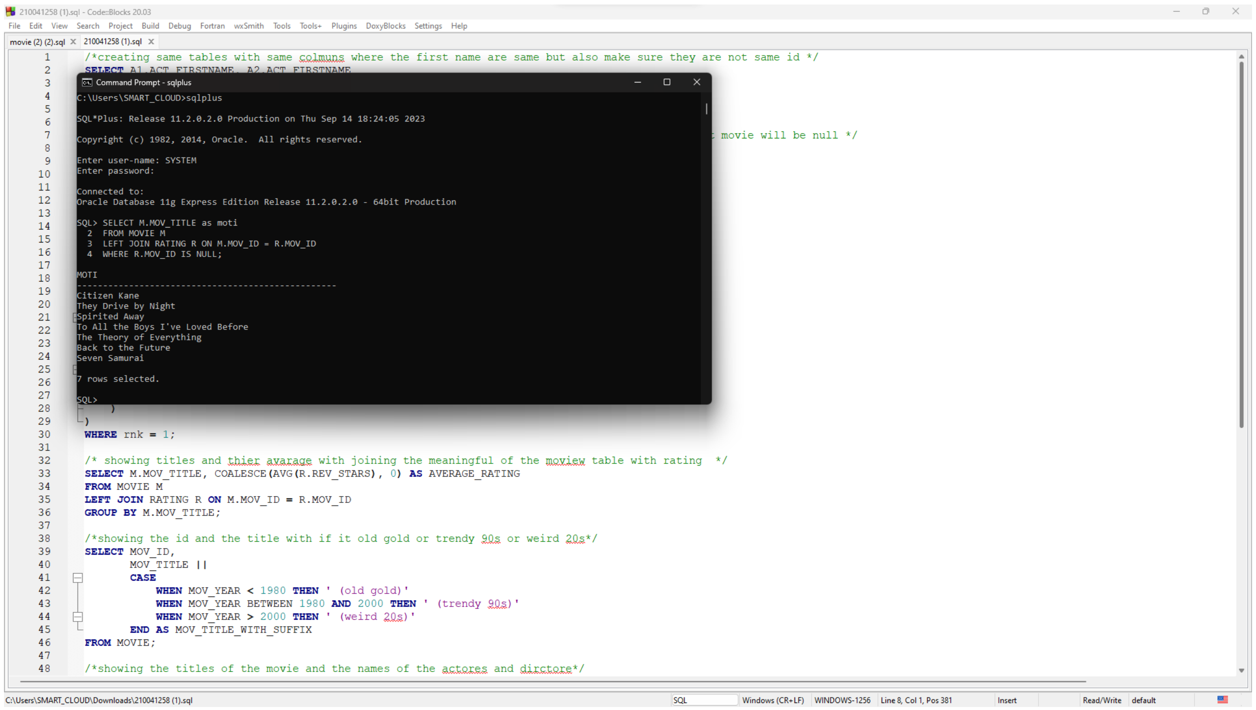
The commend 2 of assignment 5

Commend Request : Movies Without Ratings:

Code: /* selecting the movie name from movie table with take the commin from inside it where id of that movie will be null */
SELECT M.MOV_TITLE as moti
FROM MOVIE M
LEFT JOIN RATING R ON M.MOV_ID = R.MOV_ID
WHERE R.MOV_ID IS NULL;

Explanation: This query retrieves movie titles from the MOVIE table where the associated movie has no ratings.
It uses a LEFT JOIN with the RATING table and checks for cases where the MOV_ID in the RATING table is null.

Screenshot:



```
/*creating same tables with same colmuns where the first name are same but also make sure they are not same id */
SELECT A1.ACT_FIRSTNAME, A2.ACT_FIRSTNAME
```

```
movie will be null */
```

```
C:\Users\SMART_CLOUD>sqlplus

SQL*Plus: Release 11.2.0.2.0 Production on Thu Sep 14 18:24:05 2023

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Enter user-name: SYSTEM
Enter password:

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> SELECT M.MOV_TITLE as moti
  2  FROM MOVIE M
  3  LEFT JOIN RATING R ON M.MOV_ID = R.MOV_ID
  4  WHERE R.MOV_ID IS NULL;

MOTI
--------------------------------------------------
Citizen Kane
They Drive by Night
Spirited Away
To All the Boys I've Loved Before
The Theory of Everything
Back to the Future
Seven Samurai

7 rows selected.

SQL>
```

```
    )
)
WHERE rnk = 1;

/* showing titles and thier avarage with joining the meaningful of the moview table with rating  */
SELECT M.MOV_TITLE, COALESCE(AVG(R.REV_STARS), 0) AS AVERAGE_RATING
FROM MOVIE M
LEFT JOIN RATING R ON M.MOV_ID = R.MOV_ID
GROUP BY M.MOV_TITLE;

/*showing the id and the title with if it old gold or trendy 90s or weird 20s*/
SELECT MOV_ID,
       MOV_TITLE ||
       CASE
           WHEN MOV_YEAR < 1980 THEN ' (old gold)'
           WHEN MOV_YEAR BETWEEN 1980 AND 2000 THEN ' (trendy 90s)'
           WHEN MOV_YEAR > 2000 THEN ' (weird 20s)'
       END AS MOV_TITLE_WITH_SUFFIX
FROM MOVIE;

/*showing the titles of the movie and the names of the actores and dirctore*/
```

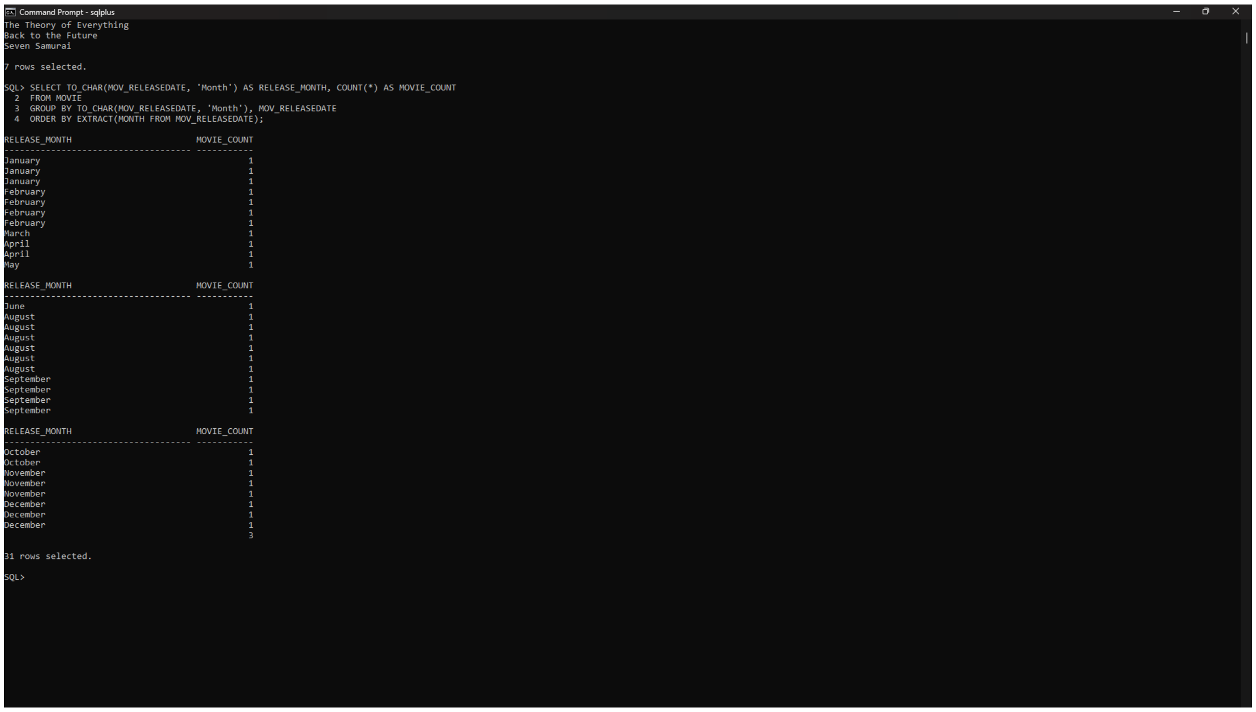
210041258 (1).sql - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
movie (2) (2).sql
210041258 (1).sql
Command Prompt - sqlplus
C:\Users\SMART_CLOUD\Downloads\210041258 (1).sql
SQL
Windows (CR+LF)
WINDOWS-1256
Line 8, Col 1, Pos 381
Insert
Read/Write
default

The commend 3 of assignment 5

Commend Request : Movie Release Months and Counts:
Code:
/* we will take the month realse by (MOV_RELEASEDATE)*/
SELECT TO_CHAR(MOV_RELEASEDATE, 'Month') AS RELEASE_MONTH, COUNT(*) AS MOVIE_COUNT
FROM MOVIE
GROUP BY TO_CHAR(MOV_RELEASEDATE, 'Month'), MOV_RELEASEDATE
ORDER BY EXTRACT(MONTH FROM MOV_RELEASEDATE);

Explanation: This query counts and lists the number of movies released in each month.
It extracts the month from the MOV_RELEASEDATE column, groups the results by month, and orders them by the month's numeric value.

Screenshot:



```
The Theory of Everything
Back to the Future
Seven Samurai

7 rows selected.

SQL> SELECT TO_CHAR(MOV_RELEASEDATE, 'Month') AS RELEASE_MONTH, COUNT(*) AS MOVIE_COUNT
     FROM MOVIE
     GROUP BY TO_CHAR(MOV_RELEASEDATE, 'Month'), MOV_RELEASEDATE
     ORDER BY EXTRACT(MONTH FROM MOV_RELEASEDATE);

RELEASE_MONTH                        MOVIE_COUNT
------------------------------------ -----------
January                                        1
January                                        1
January                                        1
February                                       1
February                                       1
February                                       1
February                                       1
March                                          1
April                                          1
April                                          1
May                                            1

RELEASE_MONTH                        MOVIE_COUNT
------------------------------------ -----------
June                                           1
August                                         1
August                                         1
August                                         1
August                                         1
August                                         1
August                                         1
September                                      1
September                                      1
September                                      1
September                                      1

RELEASE_MONTH                        MOVIE_COUNT
------------------------------------ -----------
October                                        1
October                                        1
November                                       1
November                                       1
November                                       1
December                                       1
December                                       1
December                                       1
                                               3

31 rows selected.

SQL>
```

The commend 4 of assignment 5

Commend Request : Find the months between the release date of the first movie and the last movie directed by
‘James Cameron’.

Code: SELECT MONTHS_BETWEEN(

```
    (
        SELECT MAX(MOV_RELEASEDATE)
        FROM MOVIE
        WHERE MOV_ID IN (
            SELECT MOV_ID
            FROM DIRECTION
```

```
          WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR WHERE DIR_FIRSTNAME =
'James' AND DIR_LASTNAME = 'Cameron')
        )
    ),
    (
        SELECT MIN(MOV_RELEASEDATE)
        FROM MOVIE
        WHERE MOV_ID IN (
          SELECT MOV_ID
          FROM DIRECTION
          WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR WHERE DIR_FIRSTNAME =
'James' AND DIR_LASTNAME = 'Cameron')
        )
    )
) AS MONTHS_DIFFERENCE
FROM DUAL;

SELECT MONTHS_BETWEEN(
    (
        SELECT MAX(MOV_RELEASEDATE)
        FROM MOVIE
        WHERE MOV_ID IN (
          SELECT MOV_ID
          FROM DIRECTION
          WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR WHERE DIR_FIRSTNAME =
'James' AND DIR_LASTNAME = 'Cameron')
        )
    ),
    (
        SELECT MIN(MOV_RELEASEDATE)
        FROM MOVIE
        WHERE MOV_ID IN (
          SELECT MOV_ID
          FROM DIRECTION
          WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR WHERE DIR_FIRSTNAME =
'James' AND DIR_LASTNAME = 'Cameron')
        )
    )
) AS MONTHS_DIFFERENCE
FROM DUAL;
```

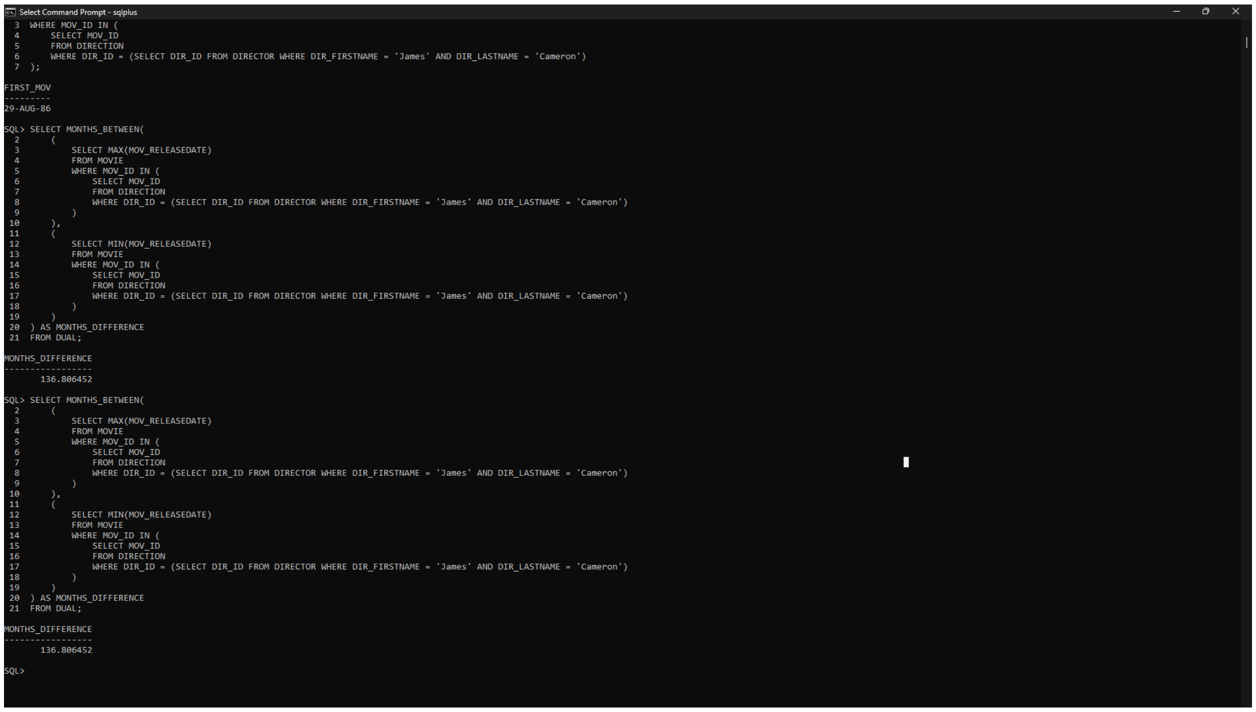
Explanation: Find the release date of the first James Cameron movie:

We first figure out the director ID for 'James Cameron.'
Then, we look at all movies directed by him and find the earliest release date among those movies.
Find the release date of the last James Cameron movie:

Similar to the first step, but we find the latest release date among his movies.
Calculate the difference in months:

We subtract the release date of the first movie from the release date of the last movie to get the time gap in months
Selecting the id of the id mof the movie of the spacic director after that find the max and the min mum of the date of these movies after that seleting two subquary , where by month between the max to min as shown below difference : 136.80

Screenshot:



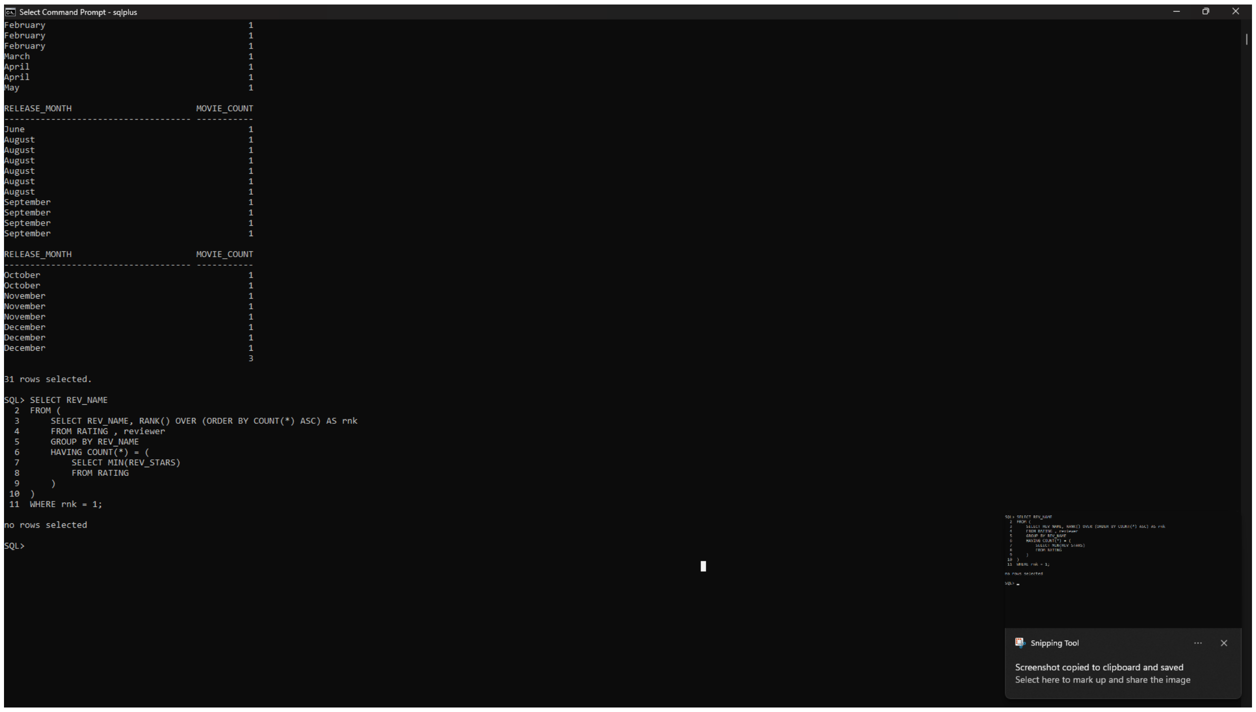
```
  3  WHERE MOV_ID IN (
  4      SELECT MOV_ID
  5      FROM DIRECTION
  6      WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR WHERE DIR_FIRSTNAME = 'James' AND DIR_LASTNAME = 'Cameron')
  7  );

FIRST_MOV
---------
29-AUG-86

SQL> SELECT MONTHS_BETWEEN(
  2      (
  3          SELECT MAX(MOV_RELEASEDATE)
  4          FROM MOVIE
  5          WHERE MOV_ID IN (
  6              SELECT MOV_ID
  7              FROM DIRECTION
  8              WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR WHERE DIR_FIRSTNAME = 'James' AND DIR_LASTNAME = 'Cameron')
  9          )
 10      ),
 11      (
 12          SELECT MIN(MOV_RELEASEDATE)
 13          FROM MOVIE
 14          WHERE MOV_ID IN (
 15              SELECT MOV_ID
 16              FROM DIRECTION
 17              WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR WHERE DIR_FIRSTNAME = 'James' AND DIR_LASTNAME = 'Cameron')
 18          )
 19      )
 20  ) AS MONTHS_DIFFERENCE
 21  FROM DUAL;

MONTHS_DIFFERENCE
-----------------
       136.806452

SQL> SELECT MONTHS_BETWEEN(
  2      (
  3          SELECT MAX(MOV_RELEASEDATE)
  4          FROM MOVIE
  5          WHERE MOV_ID IN (
  6              SELECT MOV_ID
  7              FROM DIRECTION
  8              WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR WHERE DIR_FIRSTNAME = 'James' AND DIR_LASTNAME = 'Cameron')
  9          )
 10      ),
 11      (
 12          SELECT MIN(MOV_RELEASEDATE)
 13          FROM MOVIE
 14          WHERE MOV_ID IN (
 15              SELECT MOV_ID
 16              FROM DIRECTION
 17              WHERE DIR_ID = (SELECT DIR_ID FROM DIRECTOR WHERE DIR_FIRSTNAME = 'James' AND DIR_LASTNAME = 'Cameron')
 18          )
 19      )
 20  ) AS MONTHS_DIFFERENCE
 21  FROM DUAL;

MONTHS_DIFFERENCE
-----------------
       136.806452

SQL>
```

The commend 5 of assignment 5

Commend Request : Lowest-Rated Reviewer
Code: /* ranking rank over */

```sql
SELECT REV_NAME
FROM (
    SELECT REV_NAME, RANK() OVER (ORDER BY COUNT(*) ASC) AS rnk
    FROM RATING , reviewer
    GROUP BY REV_NAME
    HAVING COUNT(*) = (
        SELECT MIN(REV_STARS)
        FROM RATING
    )
)
WHERE rnk = 1;
```

Explanation: This query finds the reviewer with the lowest average rating and selects their name.
It ranks reviewers based on the count of their ratings and selects the one with the minimum rating using a subquery.
Screenshot:



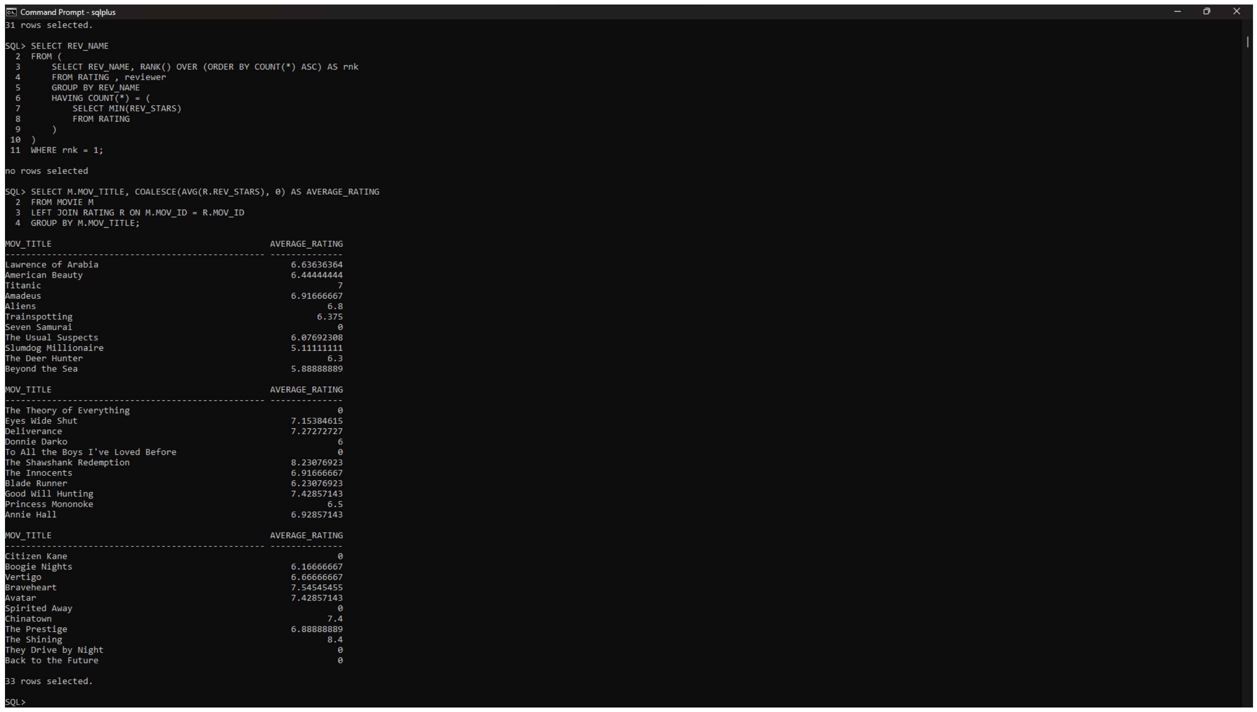
```
February                                       1
February                                       1
February                                       1
March                                          1
April                                          1
April                                          1
May                                            1

RELEASE_MONTH                        MOVIE_COUNT
------------------------------------ -----------
June                                           1
August                                         1
August                                         1
August                                         1
August                                         1
August                                         1
August                                         1
September                                      1
September                                      1
September                                      1
September                                      1

RELEASE_MONTH                        MOVIE_COUNT
------------------------------------ -----------
October                                        1
October                                        1
November                                       1
November                                       1
November                                       1
December                                       1
December                                       1
December                                       1
                                               3

31 rows selected.

SQL> SELECT REV_NAME
  2  FROM (
  3      SELECT REV_NAME, RANK() OVER (ORDER BY COUNT(*) ASC) AS rnk
  4      FROM RATING , reviewer
  5      GROUP BY REV_NAME
  6      HAVING COUNT(*) = (
  7          SELECT MIN(REV_STARS)
  8          FROM RATING
  9      )
 10  )
 11  WHERE rnk = 1;

no rows selected

SQL>
```

Snipping Tool
Screenshot copied to clipboard and saved
Select here to mark up and share the image

The commend 6 of assignment 5

Commend Request : Movie Titles with Average Ratings
Code:/* showing titles and thier avarage with joining the meaningful of the moview table with rating */
SELECT M.MOV_TITLE, COALESCE(AVG(R.REV_STARS), 0) AS AVERAGE_RATING
FROM MOVIE M
LEFT JOIN RATING R ON M.MOV_ID = R.MOV_ID
GROUP BY M.MOV_TITLE;

Explanation: This query displays movie titles along with their average ratings.
It performs a left join between the MOVIE and RATING tables and calculates the average rating for each movie.
Screenshot:



```
31 rows selected.

SQL> SELECT REV_NAME
     FROM (
         SELECT REV_NAME, RANK() OVER (ORDER BY COUNT(*) ASC) AS rnk
         FROM RATING , reviewer
         GROUP BY REV_NAME
         HAVING COUNT(*) = (
             SELECT MIN(REV_STARS)
             FROM RATING
         )
     )
     WHERE rnk = 1;

no rows selected

SQL> SELECT M.MOV_TITLE, COALESCE(AVG(R.REV_STARS), 0) AS AVERAGE_RATING
     FROM MOVIE M
     LEFT JOIN RATING R ON M.MOV_ID = R.MOV_ID
     GROUP BY M.MOV_TITLE;

MOV_TITLE                                          AVERAGE_RATING
-------------------------------------------------- --------------
Lawrence of Arabia                                     6.63636364
American Beauty                                        6.44444444
Titanic                                                         7
Amadeus                                                6.91666667
Aliens                                                        6.8
Trainspotting                                               6.375
Seven Samurai                                                   0
The Usual Suspects                                     6.07692308
Slumdog Millionaire                                    5.11111111
The Deer Hunter                                               6.3
Beyond the Sea                                         5.88888889

MOV_TITLE                                          AVERAGE_RATING
-------------------------------------------------- --------------
The Theory of Everything                                        0
Eyes Wide Shut                                         7.15384615
Deliverance                                            7.27272727
Donnie Darko                                                    6
To All the Boys I've Loved Before                               0
The Shawshank Redemption                               8.23076923
The Innocents                                          6.91666667
Blade Runner                                           6.23076923
Good Will Hunting                                      7.42857143
Princess Mononoke                                             6.5
Annie Hall                                             6.92857143

MOV_TITLE                                          AVERAGE_RATING
-------------------------------------------------- --------------
Citizen Kane                                                    0
Boogie Nights                                          6.16666667
Vertigo                                                6.66666667
Braveheart                                             7.54545455
Avatar                                                 7.42857143
Spirited Away                                                   0
Chinatown                                                     7.4
The Prestige                                           6.88888889
The Shining                                                   8.4
They Drive by Night                                             0
Back to the Future                                              0

33 rows selected.

SQL>
```

The commend 7 of assignment 5

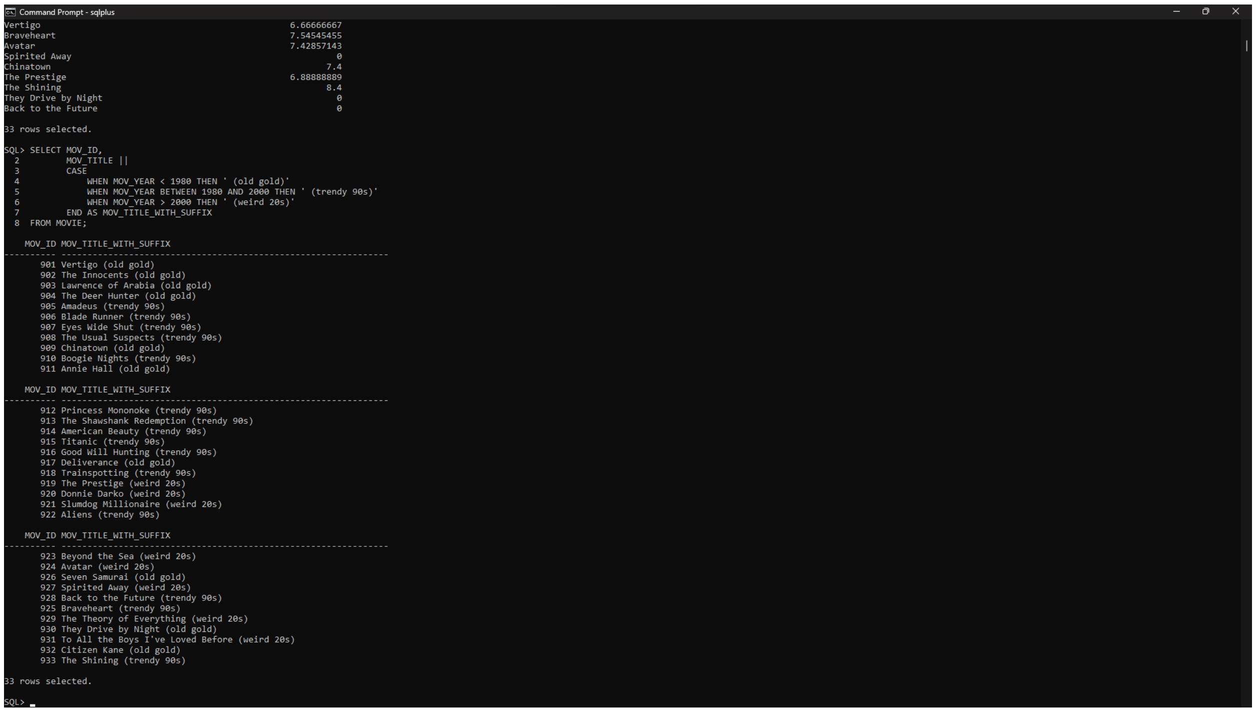
Commend Request : Movie Titles with Year Suffix

Code:/*showing the id and the title with if it old gold or trendy 90s or weird 20s*/

```
SELECT MOV_ID,
       MOV_TITLE ||
       CASE
           WHEN MOV_YEAR < 1980 THEN ' (old gold)'
           WHEN MOV_YEAR BETWEEN 1980 AND 2000 THEN ' (trendy 90s)'
           WHEN MOV_YEAR > 2000 THEN ' (weird 20s)'
       END AS MOV_TITLE_WITH_SUFFIX
FROM MOVIE;
```

Explanation: This query adds a suffix to movie titles based on their release years.
It categorizes movies into "old gold," "trendy 90s," or "weird 20s" based on the MOV_YEAR column.

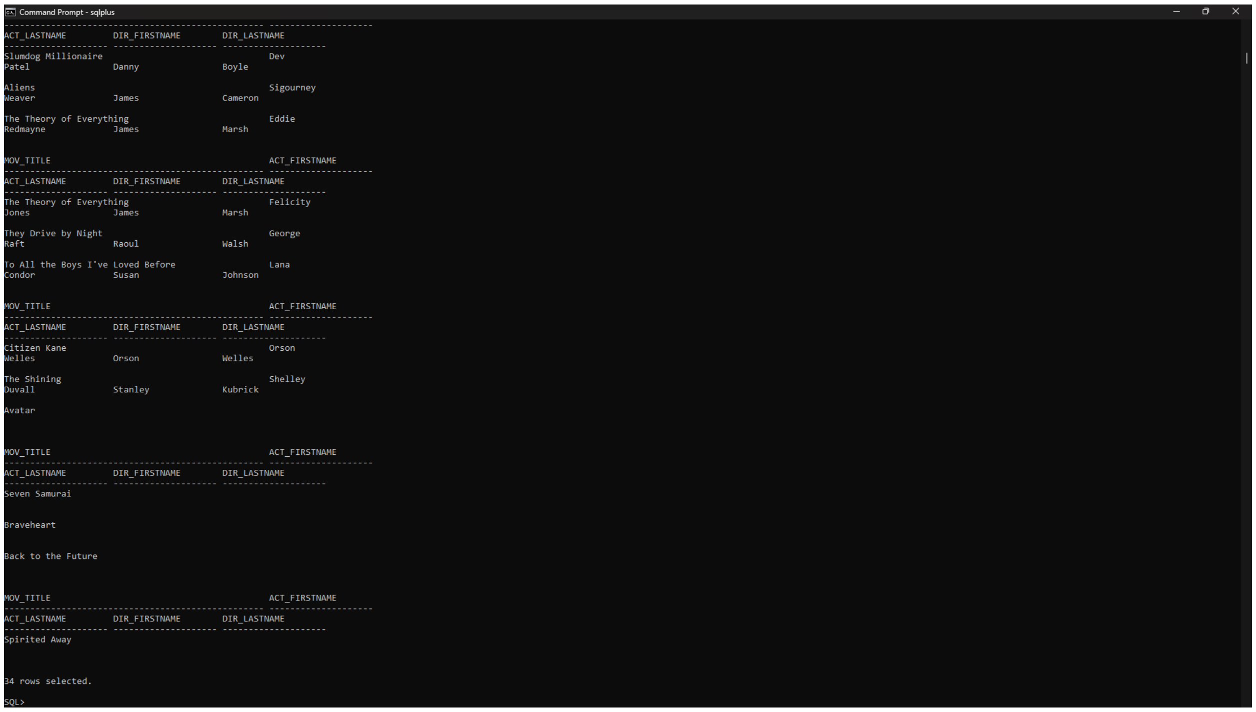
Screenshot:



```
Vertigo                                            6.66666667
Braveheart                                         7.54545455
Avatar                                             7.42857143
Spirited Away                                               0
Chinatown                                                 7.4
The Prestige                                       6.88888889
The Shining                                               8.4
They Drive by Night                                         0
Back to the Future                                          0

33 rows selected.

SQL> SELECT MOV_ID,
           MOV_TITLE ||
           CASE
               WHEN MOV_YEAR < 1980 THEN ' (old gold)'
               WHEN MOV_YEAR BETWEEN 1980 AND 2000 THEN ' (trendy 90s)'
               WHEN MOV_YEAR > 2000 THEN ' (weird 20s)'
           END AS MOV_TITLE_WITH_SUFFIX
     FROM MOVIE;

    MOV_ID MOV_TITLE_WITH_SUFFIX
---------- ---------------------------------------------------------------
       901 Vertigo (old gold)
       902 The Innocents (old gold)
       903 Lawrence of Arabia (old gold)
       904 The Deer Hunter (old gold)
       905 Amadeus (trendy 90s)
       906 Blade Runner (trendy 90s)
       907 Eyes Wide Shut (trendy 90s)
       908 The Usual Suspects (trendy 90s)
       909 Chinatown (old gold)
       910 Boogie Nights (trendy 90s)
       911 Annie Hall (old gold)

    MOV_ID MOV_TITLE_WITH_SUFFIX
---------- ---------------------------------------------------------------
       912 Princess Mononoke (trendy 90s)
       913 The Shawshank Redemption (trendy 90s)
       914 American Beauty (trendy 90s)
       915 Titanic (trendy 90s)
       916 Good Will Hunting (trendy 90s)
       917 Deliverance (old gold)
       918 Trainspotting (trendy 90s)
       919 The Prestige (weird 20s)
       920 Donnie Darko (weird 20s)
       921 Slumdog Millionaire (weird 20s)
       922 Aliens (trendy 90s)

    MOV_ID MOV_TITLE_WITH_SUFFIX
---------- ---------------------------------------------------------------
       923 Beyond the Sea (weird 20s)
       924 Avatar (weird 20s)
       926 Seven Samurai (old gold)
       927 Spirited Away (weird 20s)
       928 Back to the Future (trendy 90s)
       925 Braveheart (trendy 90s)
       929 The Theory of Everything (weird 20s)
       930 They Drive by Night (old gold)
       931 To All the Boys I've Loved Before (weird 20s)
       932 Citizen Kane (old gold)
       933 The Shining (trendy 90s)

33 rows selected.

SQL>
```

The commend 8 of assignment 5

Commend Request : Movie Titles with Actors and Directors
Code:/*showing the titles of the movie and the names of the actores and dirctore*/
SELECT M.MOV_TITLE, A.ACT_FIRSTNAME, A.ACT_LASTNAME, D.DIR_FIRSTNAME, D.DIR_LASTNAME
FROM MOVIE M
LEFT JOIN CASTS C ON M.MOV_ID = C.MOV_ID
LEFT JOIN ACTOR A ON C.ACT_ID = A.ACT_ID
LEFT JOIN DIRECTION DI ON M.MOV_ID = DI.MOV_ID
LEFT JOIN DIRECTOR D ON DI.DIR_ID = D.DIR_ID;
Explanation: This query retrieves movie titles along with the names of actors and directors involved in those movies.
It uses LEFT JOIN operations with the CASTS, ACTOR, DIRECTION, and DIRECTOR tables to associate movies with their cast and directors.
Screenshot:

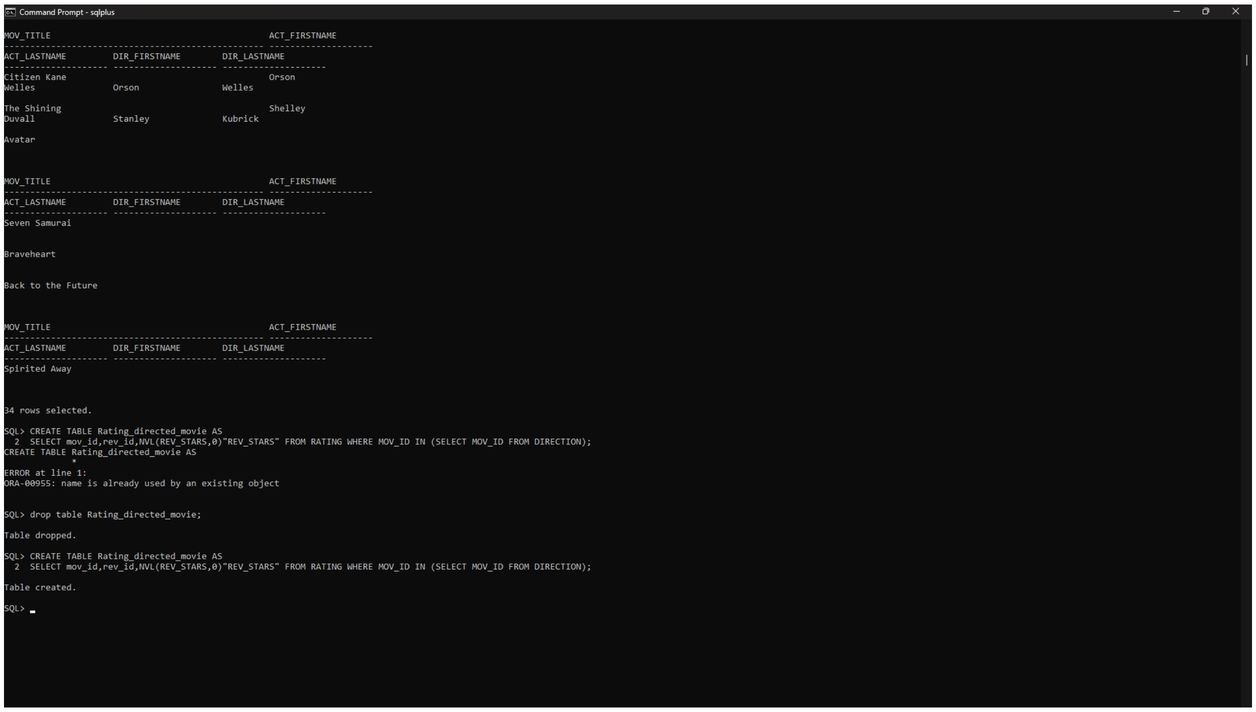


```
-------------------------------------------------- --------------------
ACT_LASTNAME         DIR_FIRSTNAME        DIR_LASTNAME
-------------------- -------------------- --------------------
Slumdog Millionaire                                Dev
Patel                Danny                Boyle

Aliens                                             Sigourney
Weaver               James                Cameron

The Theory of Everything                           Eddie
Redmayne             James                Marsh


MOV_TITLE                                          ACT_FIRSTNAME
-------------------------------------------------- --------------------
ACT_LASTNAME         DIR_FIRSTNAME        DIR_LASTNAME
-------------------- -------------------- --------------------
The Theory of Everything                           Felicity
Jones                James                Marsh

They Drive by Night                                George
Raft                 Raoul                Walsh

To All the Boys I've Loved Before                  Lana
Condor               Susan                Johnson


MOV_TITLE                                          ACT_FIRSTNAME
-------------------------------------------------- --------------------
ACT_LASTNAME         DIR_FIRSTNAME        DIR_LASTNAME
-------------------- -------------------- --------------------
Citizen Kane                                       Orson
Welles               Orson                Welles

The Shining                                        Shelley
Duvall               Stanley              Kubrick

Avatar



MOV_TITLE                                          ACT_FIRSTNAME
-------------------------------------------------- --------------------
ACT_LASTNAME         DIR_FIRSTNAME        DIR_LASTNAME
-------------------- -------------------- --------------------
Seven Samurai


Braveheart


Back to the Future



MOV_TITLE                                          ACT_FIRSTNAME
-------------------------------------------------- --------------------
ACT_LASTNAME         DIR_FIRSTNAME        DIR_LASTNAME
-------------------- -------------------- --------------------
Spirited Away



34 rows selected.

SQL>
```

The commend 9 of assignment 5

Commend Request : Creating a New Table (Rating_directed_movie):
Code:/* CREATING TABLE SUB AS TABLE */
CREATE TABLE Rating_directed_movie AS

SELECT mov_id,rev_id,NVL(REV_STARS,0)"REV_STARS" FROM RATING WHERE MOV_ID IN (SELECT MOV_ID FROM DIRECTION);
Explanation: This SQL command creates a new table called Rating_directed_movie based on a subset of data from the RATING table.
It selects specific columns (MOV_ID, REV_ID, REV_STARS) from the RATING table where the MOV_ID matches those in the DIRECTION table.
Screenshot:

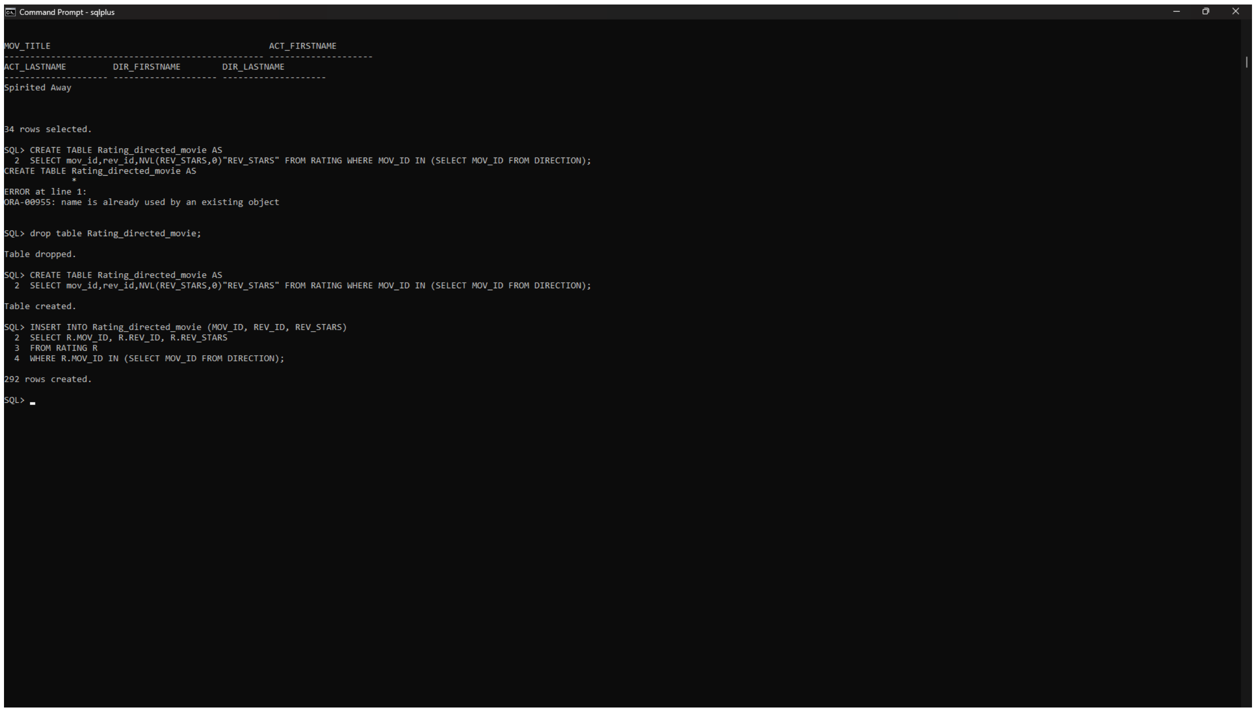


```
Command Prompt - sqlplus

MOV_TITLE                                          ACT_FIRSTNAME
-------------------------------------------------- --------------------
ACT_LASTNAME         DIR_FIRSTNAME        DIR_LASTNAME
-------------------- -------------------- --------------------
Citizen Kane                                       Orson
Welles               Orson                Welles

The Shining                                        Shelley
Duvall               Stanley              Kubrick

Avatar


MOV_TITLE                                          ACT_FIRSTNAME
-------------------------------------------------- --------------------
ACT_LASTNAME         DIR_FIRSTNAME        DIR_LASTNAME
-------------------- -------------------- --------------------
Seven Samurai


Braveheart


Back to the Future


MOV_TITLE                                          ACT_FIRSTNAME
-------------------------------------------------- --------------------
ACT_LASTNAME         DIR_FIRSTNAME        DIR_LASTNAME
-------------------- -------------------- --------------------
Spirited Away



34 rows selected.

SQL> CREATE TABLE Rating_directed_movie AS
  2  SELECT mov_id,rev_id,NVL(REV_STARS,0)"REV_STARS" FROM RATING WHERE MOV_ID IN (SELECT MOV_ID FROM DIRECTION);
CREATE TABLE Rating_directed_movie AS
             *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL> drop table Rating_directed_movie;

Table dropped.

SQL> CREATE TABLE Rating_directed_movie AS
  2  SELECT mov_id,rev_id,NVL(REV_STARS,0)"REV_STARS" FROM RATING WHERE MOV_ID IN (SELECT MOV_ID FROM DIRECTION);

Table created.

SQL> _
```

The commend 10 of assignment 5

Commend Request : Inserting Data into the New Table
Code:/* INSERTION CREATED TABLE WHICH DATA WILL BE MOVED FROM RATING TO Rating_directed_movie */
INSERT INTO Rating_directed_movie (MOV_ID, REV_ID, REV_STARS)
SELECT R.MOV_ID, R.REV_ID, R.REV_STARS
FROM RATING R
WHERE R.MOV_ID IN (SELECT MOV_ID FROM DIRECTION);
Explanation: This command inserts data into the Rating_directed_movie table.
It selects the same columns (MOV_ID, REV_ID, REV_STARS) from the RATING table and inserts them into the new table.

Screenshot:

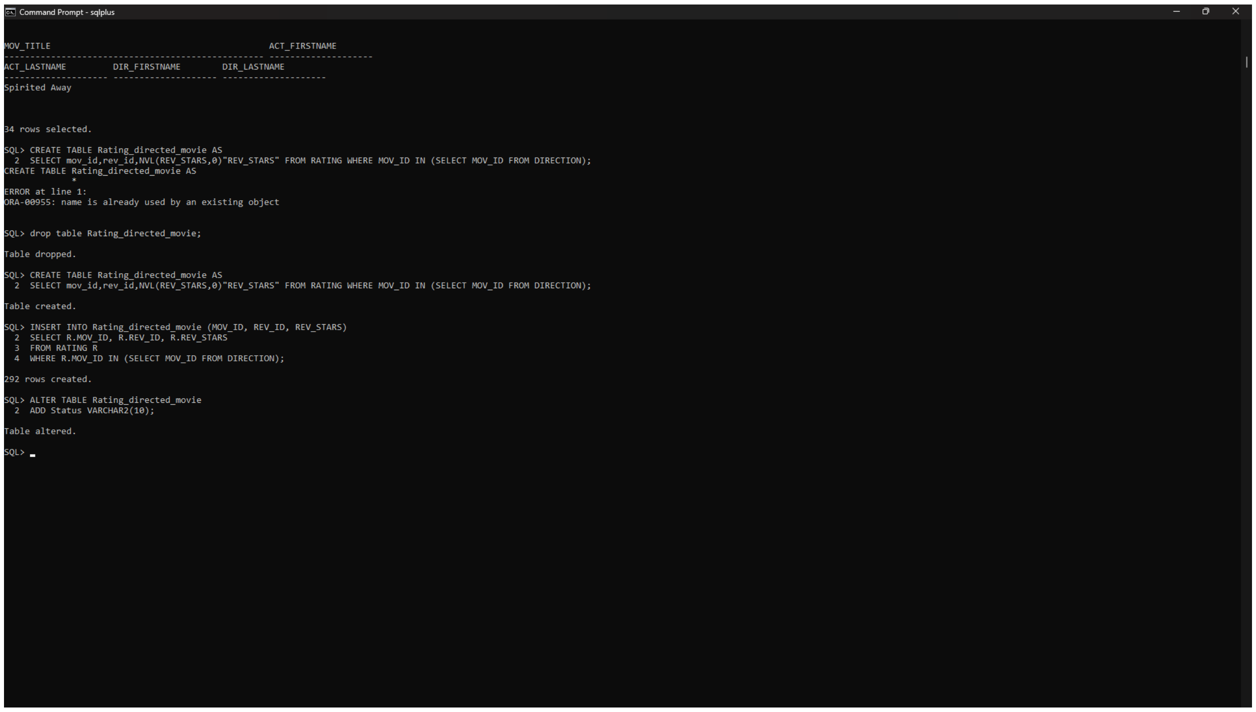


```
MOV_TITLE                                          ACT_FIRSTNAME
-------------------------------------------------- --------------------
ACT_LASTNAME         DIR_FIRSTNAME        DIR_LASTNAME
-------------------- -------------------- --------------------
Spirited Away



34 rows selected.

SQL> CREATE TABLE Rating_directed_movie AS
  2  SELECT mov_id,rev_id,NVL(REV_STARS,0)"REV_STARS" FROM RATING WHERE MOV_ID IN (SELECT MOV_ID FROM DIRECTION);
CREATE TABLE Rating_directed_movie AS
             *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL> drop table Rating_directed_movie;

Table dropped.

SQL> CREATE TABLE Rating_directed_movie AS
  2  SELECT mov_id,rev_id,NVL(REV_STARS,0)"REV_STARS" FROM RATING WHERE MOV_ID IN (SELECT MOV_ID FROM DIRECTION);

Table created.

SQL> INSERT INTO Rating_directed_movie (MOV_ID, REV_ID, REV_STARS)
  2  SELECT R.MOV_ID, R.REV_ID, R.REV_STARS
  3  FROM RATING R
  4  WHERE R.MOV_ID IN (SELECT MOV_ID FROM DIRECTION);

292 rows created.

SQL> _
```

The commend 11 of assignment 5

Commend Request : Adding a New Column to the New Table
Code:/*ADDING NEW COLUMN INSIDE THE NEW TABLE */
ALTER TABLE Rating_directed_movie
ADD Status VARCHAR2(10);
Explanation: This command adds a new column called Status to the Rating_directed_movie table with a data type of VARCHAR2(10).

Screenshot:



```
MOV_TITLE                                          ACT_FIRSTNAME
-------------------------------------------------- --------------------
ACT_LASTNAME         DIR_FIRSTNAME        DIR_LASTNAME
-------------------- -------------------- --------------------
Spirited Away



34 rows selected.

SQL> CREATE TABLE Rating_directed_movie AS
  2  SELECT mov_id,rev_id,NVL(REV_STARS,0)"REV_STARS" FROM RATING WHERE MOV_ID IN (SELECT MOV_ID FROM DIRECTION);
CREATE TABLE Rating_directed_movie AS
             *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL> drop table Rating_directed_movie;

Table dropped.

SQL> CREATE TABLE Rating_directed_movie AS
  2  SELECT mov_id,rev_id,NVL(REV_STARS,0)"REV_STARS" FROM RATING WHERE MOV_ID IN (SELECT MOV_ID FROM DIRECTION);

Table created.

SQL> INSERT INTO Rating_directed_movie (MOV_ID, REV_ID, REV_STARS)
  2  SELECT R.MOV_ID, R.REV_ID, R.REV_STARS
  3  FROM RATING R
  4  WHERE R.MOV_ID IN (SELECT MOV_ID FROM DIRECTION);

292 rows created.

SQL> ALTER TABLE Rating_directed_movie
  2  ADD Status VARCHAR2(10);

Table altered.

SQL> _
```

The commend 12 of assignment 5

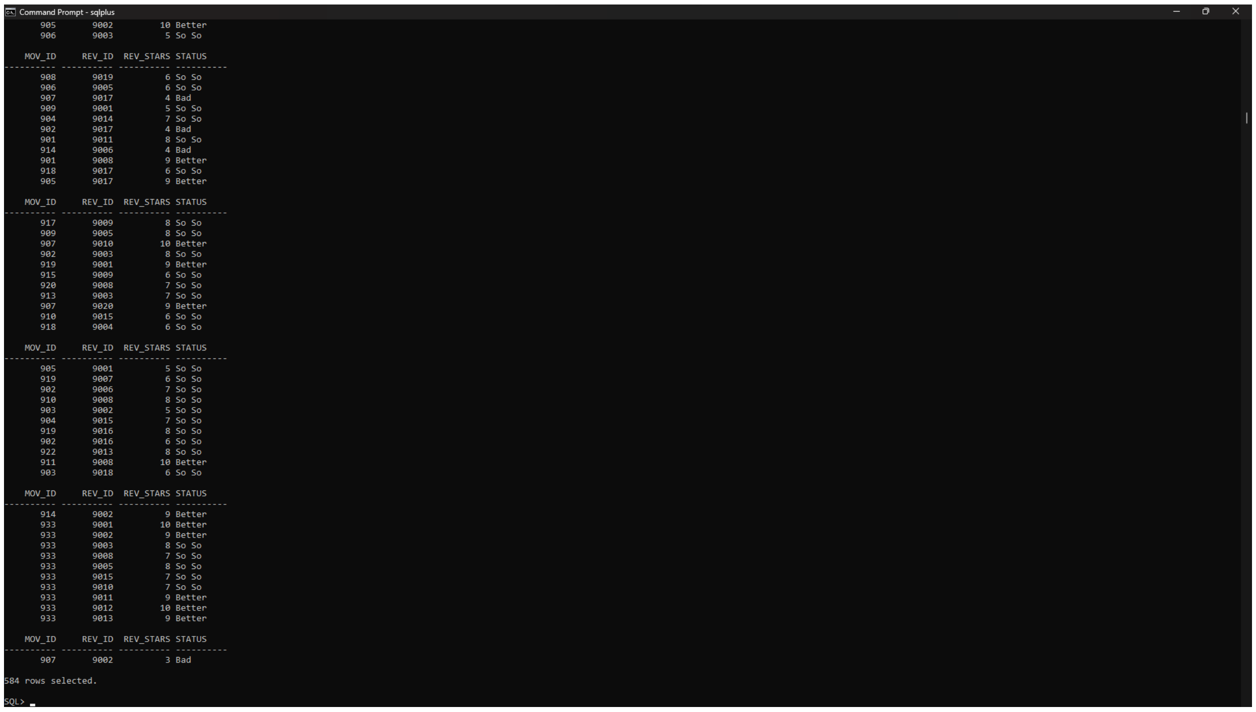
Commend Request : For each rating if it is greater than the overall rating average+2 then set the Status ’Better’, if
less than the overall rating average-2 then set the Status ’Bad’ else ’So So’
Code:

```
/* UPDATE THE TABLE OF RATING DIRECTED MOVIE VALUES FOR NEW COLUMNS FOR
Rating_directed_movie WHICH USING CASE CONDITION COMPARING WITH
REV_STARTS*/
UPDATE Rating_directed_movie
SET Status =
  CASE
    WHEN REV_STARS > (SELECT AVG(REV_STARS) FROM Rating_directed_movie) + 2
THEN 'Better'
    WHEN REV_STARS < (SELECT AVG(REV_STARS) FROM Rating_directed_movie) - 2
THEN 'Bad'
    ELSE 'So So'
  END;
```

Explanation: This query identifies directors who have directed movies with an average rating above a certain threshold.

It selects the names of directors whose movies' average ratings are higher than a specified value, using subqueries to calculate the averages and compare them to the threshold.

Screenshot:



```
       905       9002         10 Better
       906       9003          5 So So

    MOV_ID     REV_ID  REV_STARS STATUS
---------- ---------- ---------- ----------
       908       9019          6 So So
       906       9005          6 So So
       907       9017          4 Bad
       909       9001          5 So So
       904       9014          7 So So
       902       9017          4 Bad
       901       9011          8 So So
       914       9006          4 Bad
       901       9008          9 Better
       918       9017          6 So So
       905       9017          9 Better

    MOV_ID     REV_ID  REV_STARS STATUS
---------- ---------- ---------- ----------
       917       9009          8 So So
       909       9005          8 So So
       907       9010         10 Better
       902       9003          8 So So
       919       9001          9 Better
       915       9009          6 So So
       920       9008          7 So So
       913       9003          7 So So
       907       9020          9 Better
       910       9015          6 So So
       918       9004          6 So So

    MOV_ID     REV_ID  REV_STARS STATUS
---------- ---------- ---------- ----------
       905       9001          5 So So
       919       9007          6 So So
       902       9006          7 So So
       910       9008          8 So So
       903       9002          5 So So
       904       9015          7 So So
       919       9016          8 So So
       902       9016          6 So So
       922       9013          8 So So
       911       9008         10 Better
       903       9018          6 So So

    MOV_ID     REV_ID  REV_STARS STATUS
---------- ---------- ---------- ----------
       914       9002          9 Better
       933       9001         10 Better
       933       9002          9 Better
       933       9003          8 So So
       933       9008          7 So So
       933       9005          8 So So
       933       9015          7 So So
       933       9010          7 So So
       933       9011          9 Better
       933       9012         10 Better
       933       9013          9 Better

    MOV_ID     REV_ID  REV_STARS STATUS
---------- ---------- ---------- ----------
       907       9002          3 Bad

584 rows selected.

SQL> _
```