

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

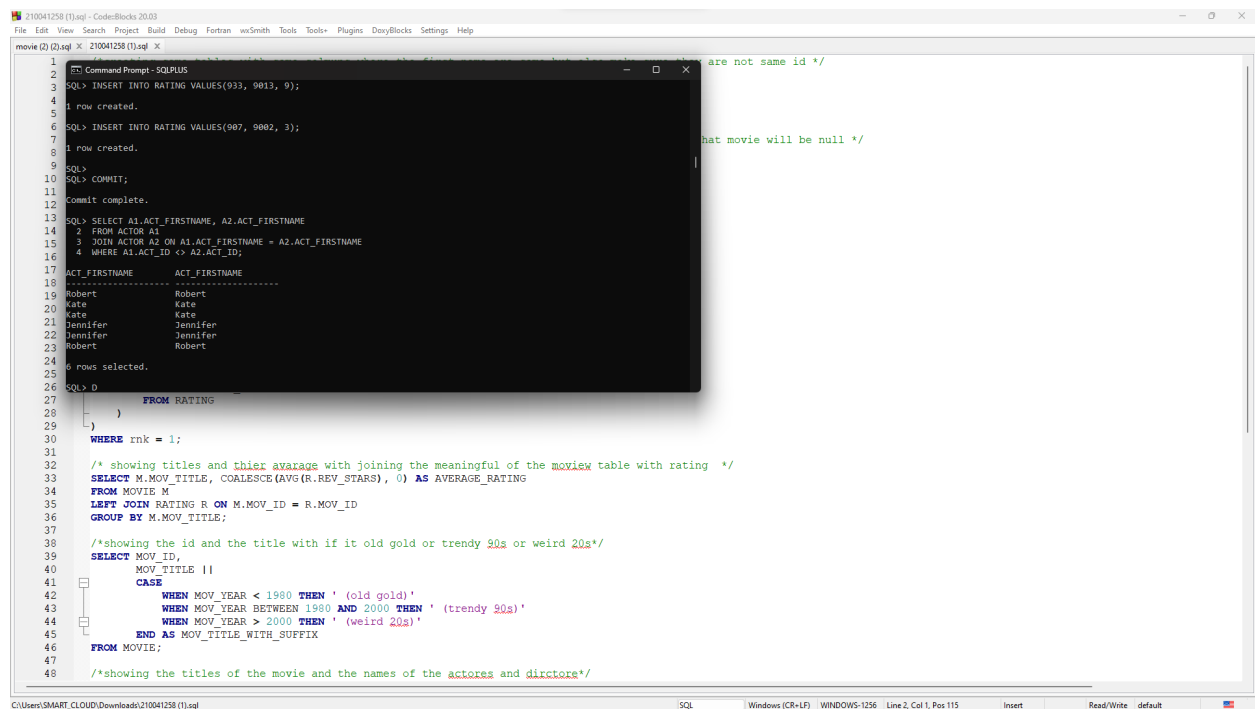
Commend Request : Duplicate Actor First Names:

Code: /*creating same tables with same colmunns 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:



```
1 2
3 SQL> INSERT INTO RATING VALUES(933, 9813, 9);
4 1 row created.
5
6 SQL> INSERT INTO RATING VALUES(987, 9802, 3);
7 1 row created.
8
9 SQL> COMMIT;
10 Commit complete.
11
12 SQL> SELECT A1.ACT_FIRSTNAME, A2.ACT_FIRSTNAME
13 2 FROM ACTOR A1
14 3 JOIN ACTOR A2 ON A1.ACT_FIRSTNAME = A2.ACT_FIRSTNAME
15 4 WHERE A1.ACT_ID <> A2.ACT_ID;
16
17 ACT_FIRSTNAME    ACT_FIRSTNAME
18 -----
19 Robert           Robert
20 Kate             Kate
21 Jennifer         Jennifer
22 Jennifer         Jennifer
23 Robert           Robert
24 4 rows selected.
25
26 SQL> D
27
28 FROM RATING
29
30 WHERE rnk = 1;
31
32 /* showing titles and their average with joining the meaningful of the movie table with rating */
33 SELECT M.MOV_TITLE, COALESCE(AVG(R.REV_STARS), 0) AS AVERAGE_RATING
34 FROM MOVIE M
35 LEFT JOIN RATING R ON M.MOV_ID = R.MOV_ID
36 GROUP BY M.MOV_TITLE;
37
38 /*showing the id and the title with if it old gold or trendy 90s or weird 20s*/
39 SELECT MOV_ID,
40 MOV_TITLE ||
41 CASE
42 WHEN MOV_YEAR < 1980 THEN ' (old gold)'
43 WHEN MOV_YEAR BETWEEN 1980 AND 2000 THEN ' (trendy 90s)'
44 WHEN MOV_YEAR > 2000 THEN ' (weird 20s)'
45 END AS MOV_TITLE_WITH_SUFFIX
46 FROM MOVIE;
47
48 /*showing the titles of the movie and the names of the actores and directors*/
```

The command 2 of assignment 5

Command 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:

```
File Edit View Search Project Build Debug Fortran wsSmith Tools Plugins DvayBlocks Settings Help
210041258 (1)sql> CodeBlocks 2023
movie (2) sql X 210041258 (1)sql X

1 /*creating same tables with same columns where the first name are same but also make sure they are not same id */
2 SELECT A1 A1T FIRSTNAME, A2 A2T FIRSTNAME
3
4 C:\Users\SMART_CLOUD>sqlplus
5
6 SQL*Plus: Release 11.2.0.2.0 Production on Thu Sep 14 18:24:05 2023
7
8 Copyright (c) 1982, 2014, Oracle. All rights reserved.
9
10 Enter user-name: SYSTEM
11 Enter password:
12
13 Connected to:
14 Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
15
16 SQL> SELECT M.MOV_TITLE as motl
17       2 FROM MOVIE M
18     3 LEFT JOIN RATING R ON M.MOV_ID = R.MOV_ID
19     4 WHERE R.MOV_ID IS NULL;
20
21 MOTI
22 -----
23 Citizen Kane
24 They Drive by Night
25 Spirited Away
26 To All the Boys I've Loved Before
27 The Theory of Everything
28 Back to the Future
29 Seven Samurai
30
31 7 rows selected.
32
33 SQL>
34 )
35
36 WHERE rnk = 1;
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968

```

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:

```
SQL Command Prompt - sqlplus
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
2 FROM MOVIE
3 GROUP BY TO_CHAR(MOV_RELEASEDATE, 'Month'), MOV_RELEASEDATE
4 ORDER BY EXTRACT(MONTH FROM MOV_RELEASEDATE);

RELEASE_MONTH      MOVIE_COUNT
-----
January            1
January            1
January            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
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           3

11 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:

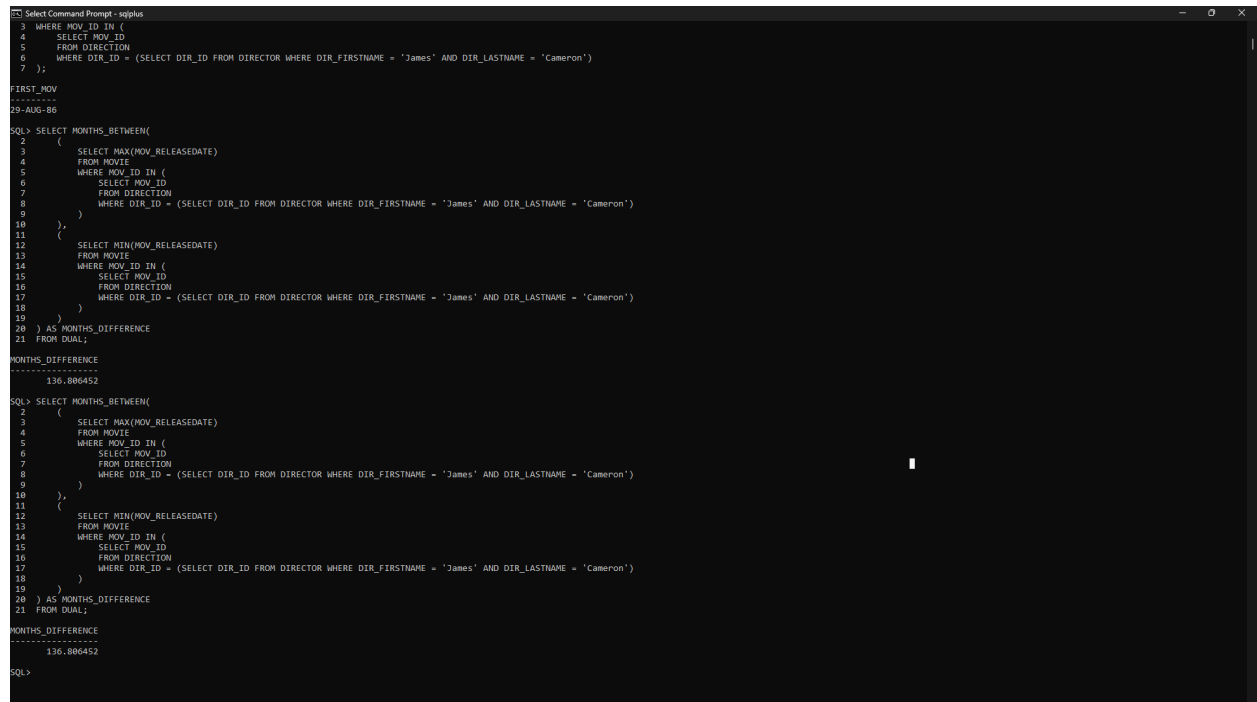
Similar to the first step, but we find the latest release date among his movies.

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:



```
SQL> SELECT MOV_ID IN (
  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 */

```

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:

```

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>

```

31 rows selected.

RELEASE_MONTH	MOVIE_COUNT
February	1
February	1
February	1
March	1
April	1
April	1
May	1
June	1
August	1
August	1
August	1
August	1
August	1
September	1
September	1
September	1
September	1
October	1
October	1
November	1
November	1
November	1
December	1
December	1
December	1
December	3

The commend 6 of assignment 5

Commend Request : Movie Titles with Average Ratings

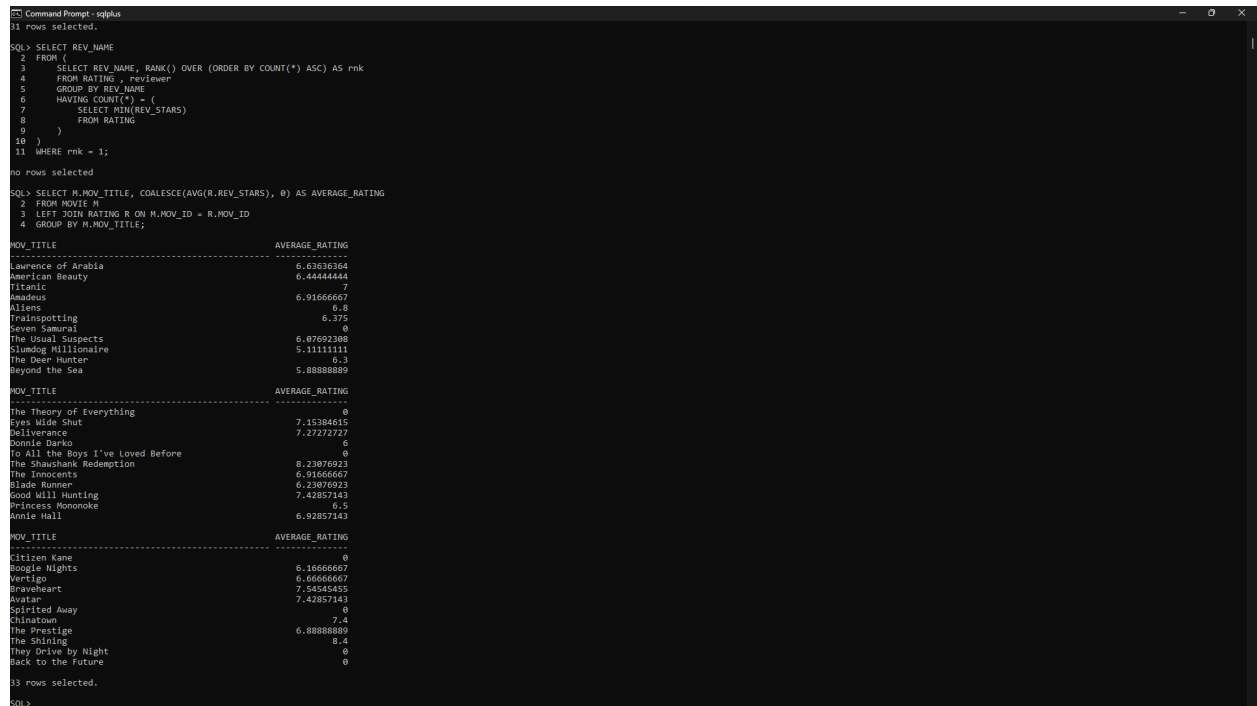
Code:/* showing titles and thier avarage with joining the meaningful of the moview table with rating */

```
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:



```
Command Prompt - sqlplus
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> SELECT M.MOV_TITLE, COALESCE(AVG(R.REV_STARS), 0) AS AVERAGE_RATING
2 FROM MOVIE M
3 LEFT JOIN RATING R ON M.MOV_ID = R.MOV_ID
4 GROUP BY M.MOV_TITLE;

MOV_TITLE                                AVERAGE_RATING
-----
Lawrence of Arabia                        6.63636364
American Beauty                          6.44444444
Titanic                                   7
Anadeus                                   6.91666667
Alien                                     6.8
Trainspotting                            6.375
Seven Samurai                            0
The Usual Suspects                       6.87692308
Gladiator 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
Bellverence                             7.27272727
Donnie Darko                             6
10 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
Dunnie Hall                             6.92857143

MOV_TITLE                                AVERAGE_RATING
-----
Citizen Kane                            0
Boogie Nights                           6.16666667
Vertigo                                 6.86666667
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

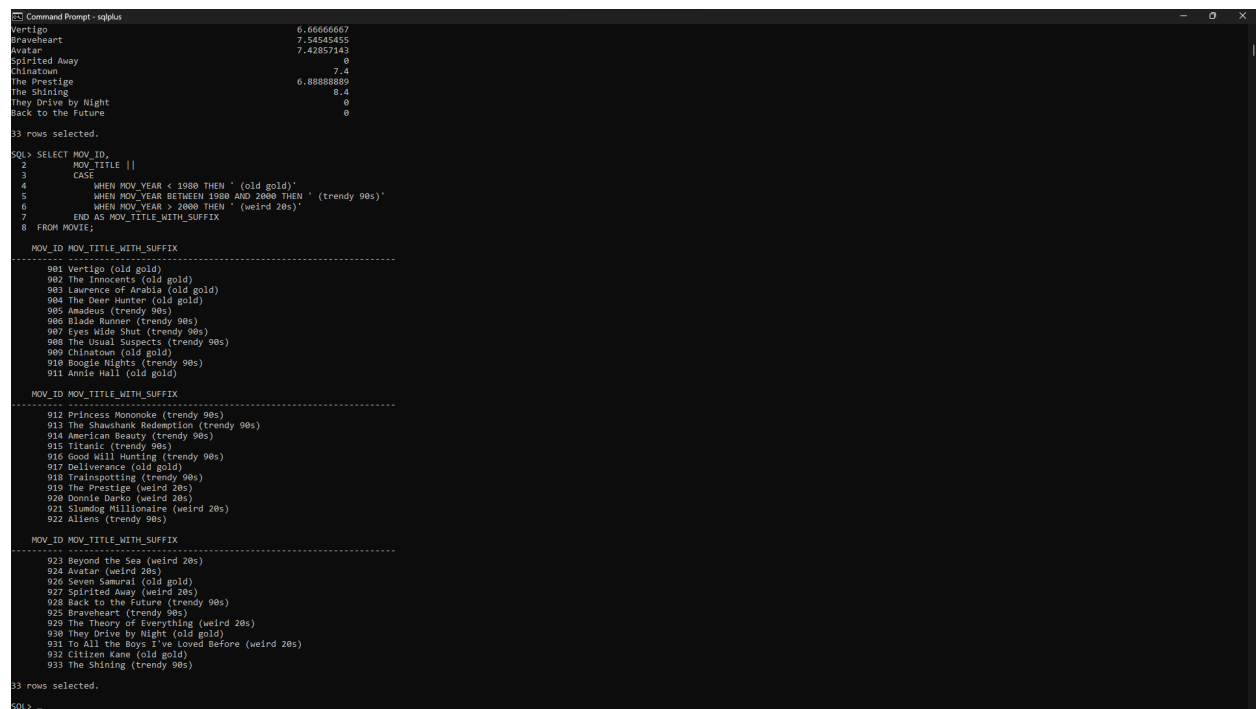
Code:/*showing the id and the title with if it old gold or trendy 90s or weird 20s*/

```
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:



```
Command Prompt - sqlplus  
Vertigo 6.66666667  
Braveheart 7.54545455  
Avatar 7.4287143  
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,  
2 MOV_TITLE ||  
3 CASE  
4 WHEN MOV_YEAR < 1980 THEN ' (old gold)'  
5 WHEN MOV_YEAR BETWEEN 1980 AND 2000 THEN ' (trendy 90s)'  
6 WHEN MOV_YEAR > 2000 THEN ' (weird 20s)'  
7 END AS MOV_TITLE_WITH_SUFFIX  
8 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)  
925 Seven Samurai (old gold)  
926 Spirited Away (weird 20s)  
927 Back to the Future (trendy 90s)  
928 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

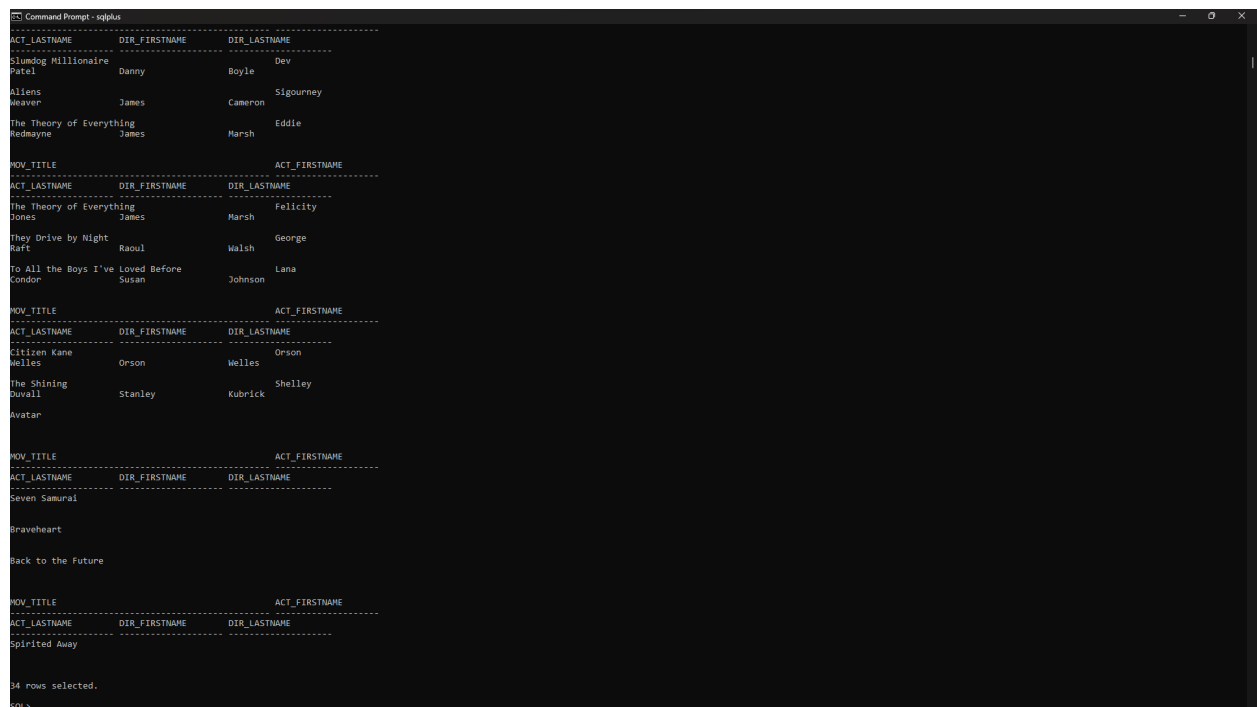
Code:/*showing the titles of the movie and the names of the actores and dirctore*/

```
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:



```
Command Prompt - sqlplus
-----
ACT_LASTNAME      DIR_FIRSTNAME      DIR_LASTNAME
-----
Blundog Millionaire      Danny      Boyle      Dev
Patel
Aliens                  James      Cameron      Sigourney
Weaver
The Theory of Everything      James      Marsh      Eddie
Redmayne
MOV_TITLE      ACT_FIRSTNAME
-----
ACT_LASTNAME      DIR_FIRSTNAME      DIR_LASTNAME
-----
The Theory of Everything      James      Marsh      Felicity
Jones
They Drive by Night      Raoul      Walsh      George
Raft
To All the Boys I've Loved Before      Susan      Johnson      Lana
Condor
MOV_TITLE      ACT_FIRSTNAME
-----
ACT_LASTNAME      DIR_FIRSTNAME      DIR_LASTNAME
-----
Citizen Kane      Orson      Welles      Orson
Welles
The Shining      Stanley      Kubrick      Shelley
Duvall
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 */

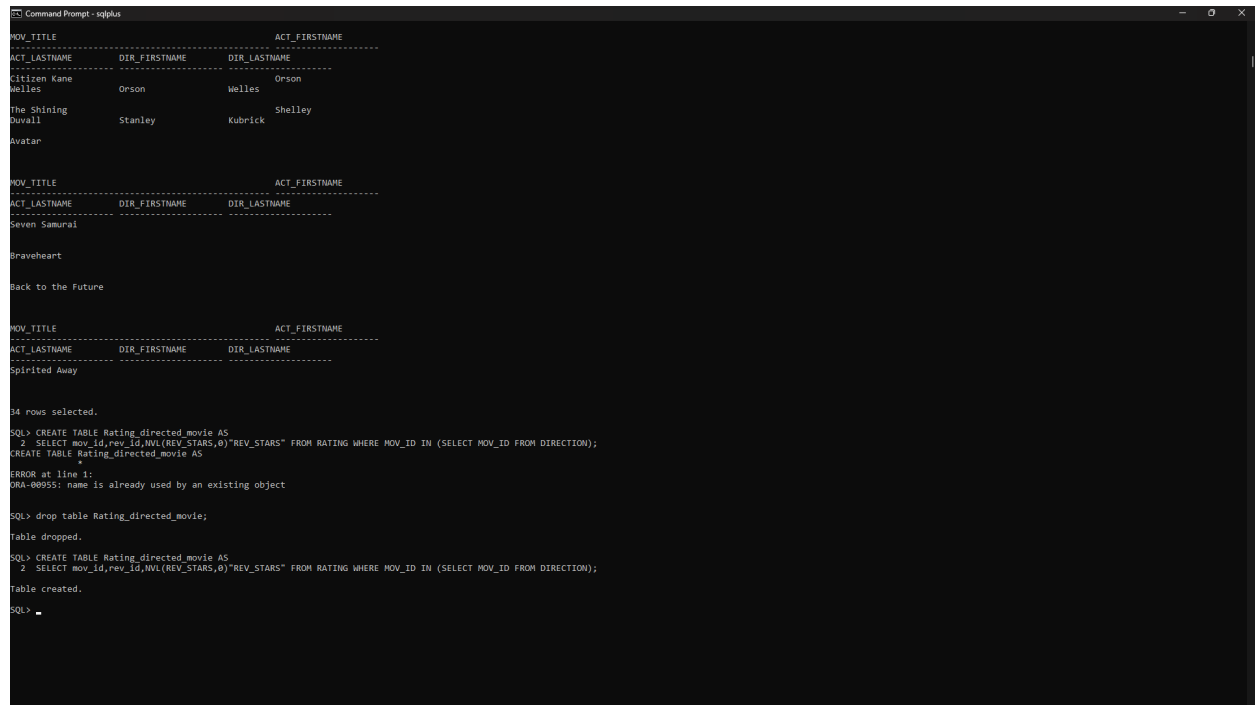
CREATE TABLE Rating_directed_movie AS

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:



```
SQL Command Prompt - sqlplus

MOV_TITLE          ACT_FIRSTNAME
-----
ACT_LASTNAME      DIR_FIRSTNAME  DIR_LASTNAME
-----
Citizen Kane      Orson          Orson
Welles            Orson          Welles
The Shining       Stanley        Kubrick
Duvall            Stanley        Shelley
Avatar            Stanley        Shelley

MOV_TITLE          ACT_FIRSTNAME
-----
ACT_LASTNAME      DIR_FIRSTNAME  DIR_LASTNAME
-----
Seven Samurai     Stanley        Kubrick
Braveheart         Stanley        Kubrick
Back to the Future Stanley        Kubrick

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

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:

```
SQL> SELECT MOV_TITLE, ACT_FIRSTNAME, ACT_LASTNAME, DIR_FIRSTNAME, DIR_LASTNAME
FROM RATING
WHERE MOV_ID = 11;
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
3
4
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 command 11 of assignment 5

Command 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:

```
Command Prompt - sqlplus

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.

Screenshot:

```

Command Prompt - sqls
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
907 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 6 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>

```