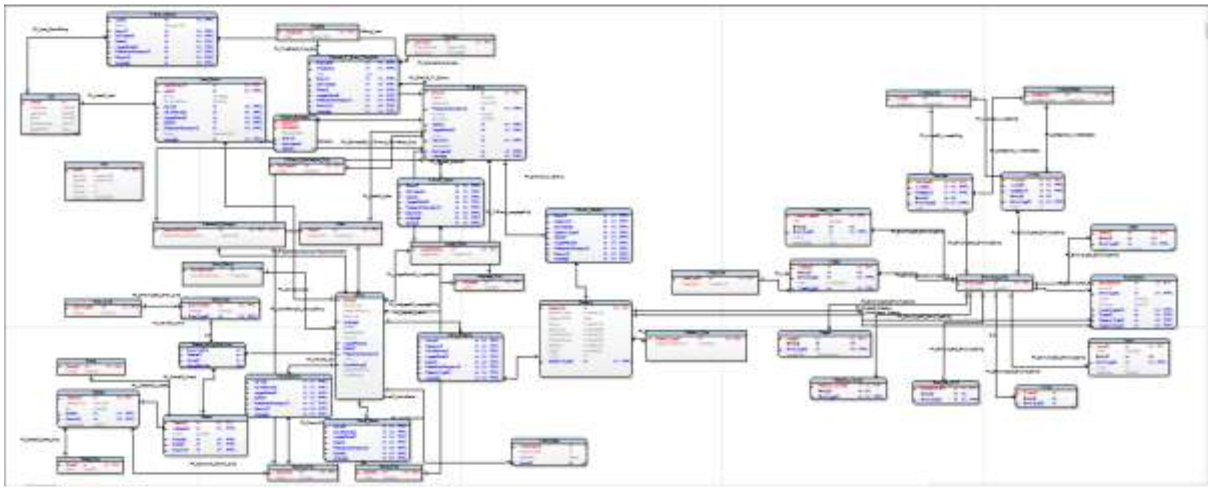
Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

Database Design and Data Management - Fall 2016

Oracle Database final Project – Modelling IMDB Database

ER model:



PROCEDURES

1.Display Award Winners based on Year and Award Category:

```
CREATE OR REPLACE PROCEDURE USP_AWARD_WINNERS

(Input_year IN Winners.AwardYear%TYPE,

Input_awardID IN Awards_Cnfg.awardID%TYPE )

IS

Cursor winner_cursor

IS

SELECT ac2.CategoryName,

 w.AwardYear,

CASE

WHEN w.EntityTypeID = 1 THEN m.MovieName
```

```
ELSE c.CelebrityName

END Winner

FROM Winners w

JOIN Awards_Cnfg ac

ON ac.AwardID = w.AwardID

JOIN Award_Category ac2

ON ac2.CategoryID = w.CategoryID

LEFT JOIN Movie m

ON m.MovieID = w.EntityID

LEFT JOIN Celebrity c

ON c.CelebrityID =w.EntityID

WHERE w.AwardYear = Input_year AND w.AwardID = Input_awardID;


cName Award_Category. CategoryName %TYPE;

inpYear Winners.AwardYear%TYPE;

winner Movie.MovieName %TYPE;

BEGIN

open winner_cursor;

loop

fetch winner_cursor into

cName,

inpYear,

winner;

exit when winner_cursor %notfound;

DBMS_OUTPUT.PUT_LINE('Category : ' || cName);

DBMS_OUTPUT.PUT_LINE('Year : ' || inpYear);
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

DBMS_OUTPUT.PUT_LINE('Winner ' || winner);

end loop;

close winner_cursor;

COMMIT;

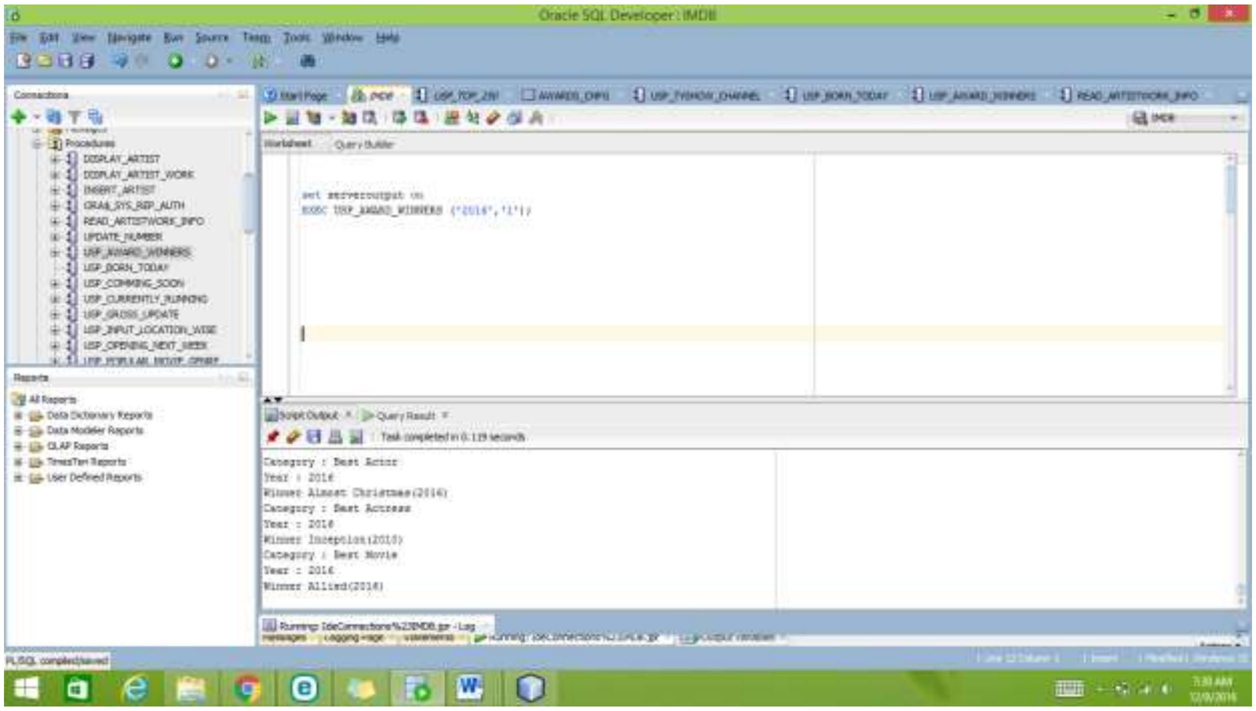
END USP_AWARD_WINNERS;

INPUT:

set serveroutput on;

EXEC USP_AWARD_WINNERS ('2016','1');

OUTPUT:



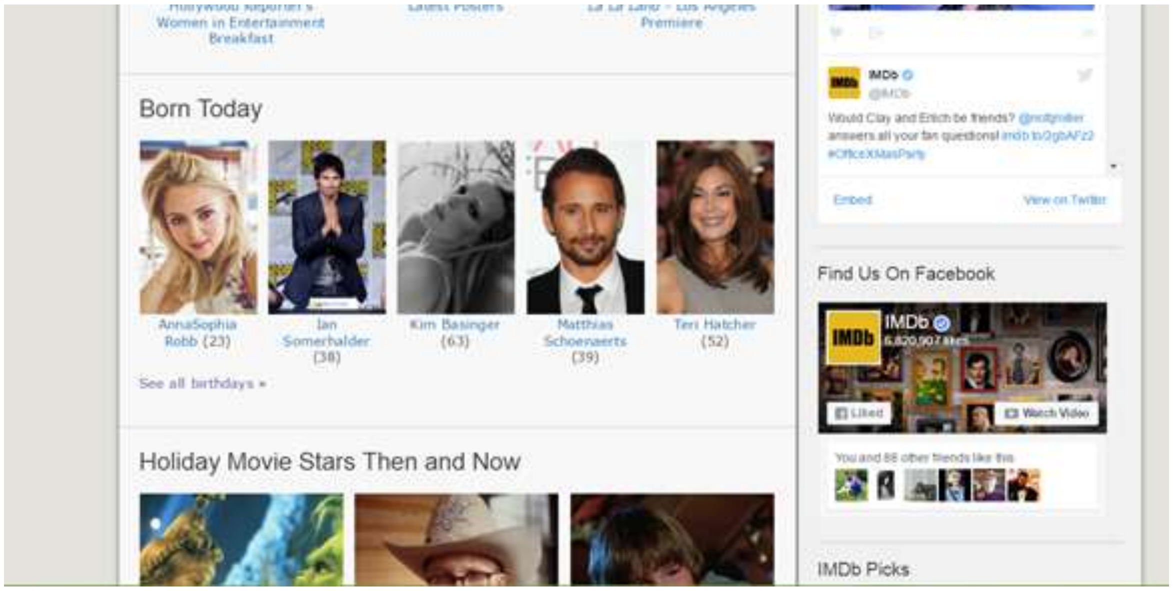
```
set serveroutput on
EXEC USP_AWARD_WINNERS ('2016','1');
```

Task completed in 0.119 seconds

```
Category : Best Actor
Year : 2016
Winner Almost Christmas(2016)
Category : Best Actress
Year : 2016
Winner Inception(2010)
Category : Best Movie
Year : 2016
Winner Allied(2016)
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

2.Display Celebrity Born Today and their Age:



```
create or replace PROCEDURE USP_BORN_TODAY

IS

Cursor born_today_cursor IS

SELECT  CelebrityName,AlternateName,PersonalQuote,Round(TRUNC(MONTHS_BETWEEN(SYSDATE,
celebrityDOB))/12,-1) As Age

from Celebrity where extract( day from CelebrityDOB)=extract (day from sysdate) AND extract(month
from CELEBRITYDOB

) =extract(month from sysdate);

A_Age int;

A_CelebrityName celebrity.CelebrityName%TYPE;

A_AlternateName CELEBRITY.ALTERNATENAME%TYPE;

A_PersonalQuote CELEBRITY.PERSONALQUOTE%TYPE;

BEGIN

open born_today_cursor;

loop

fetch born_today_cursor into
```

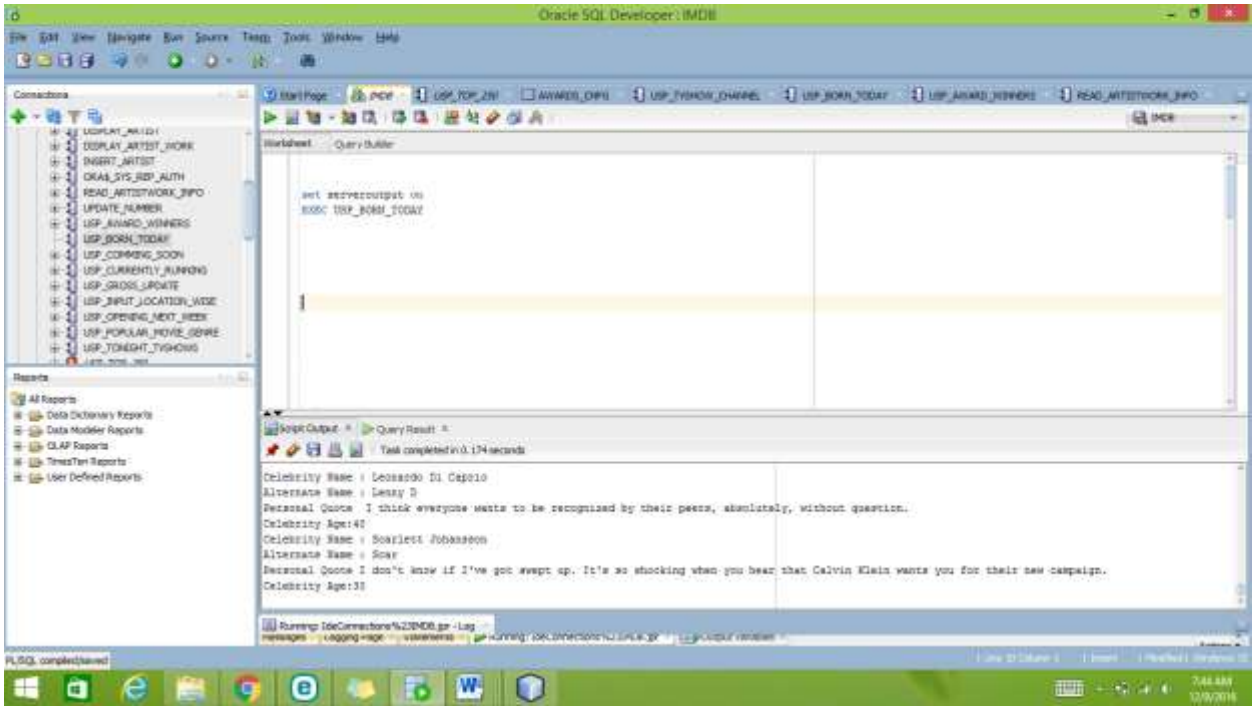
```
A_CelebrityName,

A_AlternateName,

A_PersonalQuote,

A_Age;

exit when born_today_cursor%notfound;

DBMS_OUTPUT.PUT_LINE('Celebrity Name : ' ||A_CelebrityName );

DBMS_OUTPUT.PUT_LINE('Alternate Name : ' ||A_AlternateName );

DBMS_OUTPUT.PUT_LINE('Personal Quote ' || A_PersonalQuote);

DBMS_OUTPUT.PUT_LINE('Celebrity Age:'|| A_Age );

end loop;

close born_today_cursor;

COMMIT;

END USP_BORN_TODAY;
```

INPUT:

```
set serveroutput on

EXEC USP_BORN_TODAY
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

OUTPUT:



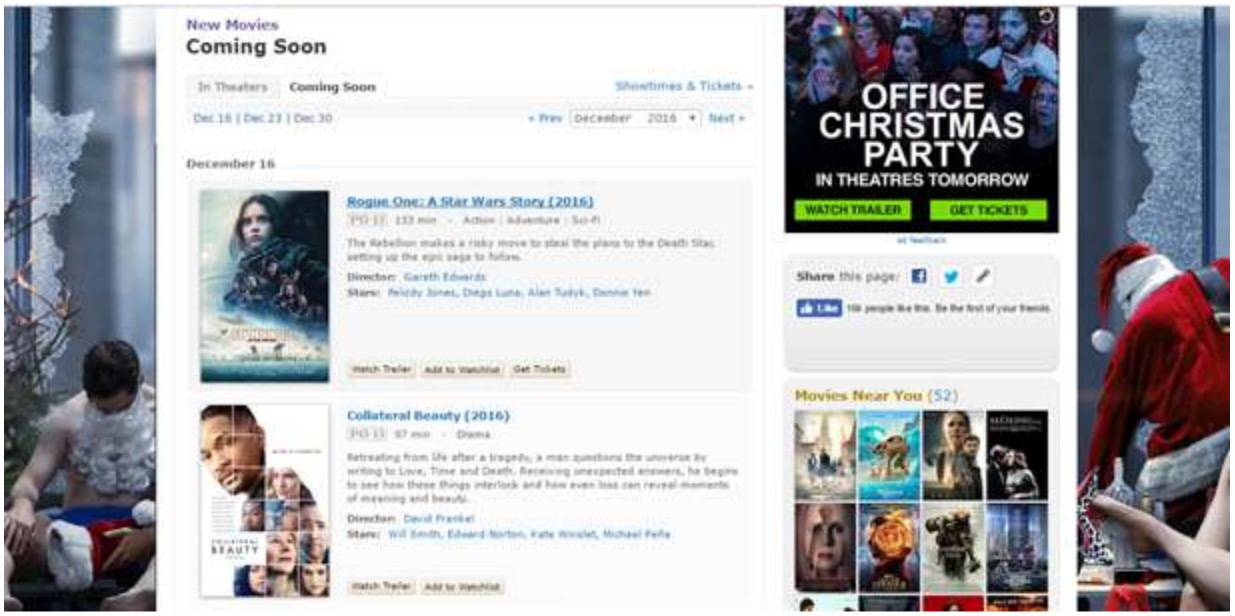
```
set serveroutput on
EXEC USP_BORN_TODAY
```

Task completed in 0.174 seconds

```
Celebrity Name : Leonardo Di Caprio
Alternate Name : Lenny D
Personal Quote  I think everyone wants to be recognized by their peers, absolutely, without question.
Celebrity Age:42
Celebrity Name : Scarlett Johansson
Alternate Name : Scar
Personal Quote I don't know if I've got swept up. It's so shocking when you hear that Calvin Klein wants you for their new campaign.
Celebrity Age:33
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

3.Display Movie List Coming Soon


New Movies
Coming Soon
In Theaters | Coming Soon | Showtimes & Tickets »
Dec 16 | Dec 23 | Dec 30 « Prev December 2016 Next »

December 16

Rogue One: A Star Wars Story (2016)
PG-13 133 min - Action | Adventure | Sci-Fi
The Rebellion makes a risky move to steal the plans to the Death Star, setting up the epic saga to follow.
Director: Gareth Edwards
Stars: Felicity Jones, Diego Luna, Alan Tudyk, Donnie Yen
Watch Trailer | Add to Watchlist | Get Tickets

Collateral Beauty (2016)
PG-13 97 min - Drama
Retreating from life after a tragedy, a man questions the universe by writing to Love, Time and Death. Receiving unexpected answers, he begins to see how these things interlock and how even loss can reveal moments of meaning and beauty.
Director: David Frankel
Stars: Will Smith, Edward Norton, Kate Winslet, Michael Peña
Watch Trailer | Add to Watchlist

```
create or replace PROCEDURE USP_COMMING_SOON

IS

Cursor movie_cursor IS

select distinct MovieID,MovieName,Runtime ,Storyline,ReleaseDate,listagg(RTRIM(name),',') WITHIN GROUP (ORDER BY MOVIENAME)  As Genre

FROM  (SELECT Movie.MovieID,MovieName,RunTime,GCnfg.Name,StoryLine,ReleaseDate from Movie Movie

Join Movie_Genre Genre ON Movie.MovieID=Genre.MovieID

Join Genre_Cnfg GCnfg on Genre.GenreID=GCnfg.GenreID  where ReleaseDate > (select sysdate + 7 from dual))

GROUP BY MovieID,MovieName,Runtime ,Storyline,ReleaseDate;


A_MovieID Movie.MovieID%TYPE;

A_MovieName  Movie.MovieName%TYPE;

A_Runtime  Movie.Runtime%TYPE;
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

```
A_Storyline  Movie.Storyline%TYPE;

A_ReleaseDate  Movie.ReleaseDate%TYPE;

A_Genre  varchar2(100);

BEGIN

open movie_cursor;

loop

fetch movie_cursor into

A_MovieID,

A_MovieName,

A_Runtime,

A_Storyline,

A_ReleaseDate,

A_Genre;

exit when movie_cursor%notfound;

DBMS_OUTPUT.PUT_LINE('Movie ID : ' ||A_MovieID );

DBMS_OUTPUT.PUT_LINE('Movie Name : ' ||A_MovieName );

DBMS_OUTPUT.PUT_LINE('Run Time: ' || A_Runtime);


DBMS_OUTPUT.PUT_LINE('StoryLine :'|| A_Storyline );

DBMS_OUTPUT.PUT_LINE('Release Date: ' || A_ReleaseDate);

DBMS_OUTPUT.PUT_LINE('Genre: ' || A_Genre);

end loop;

close movie_cursor;

COMMIT;

END USP_COMMING_SOON;
```

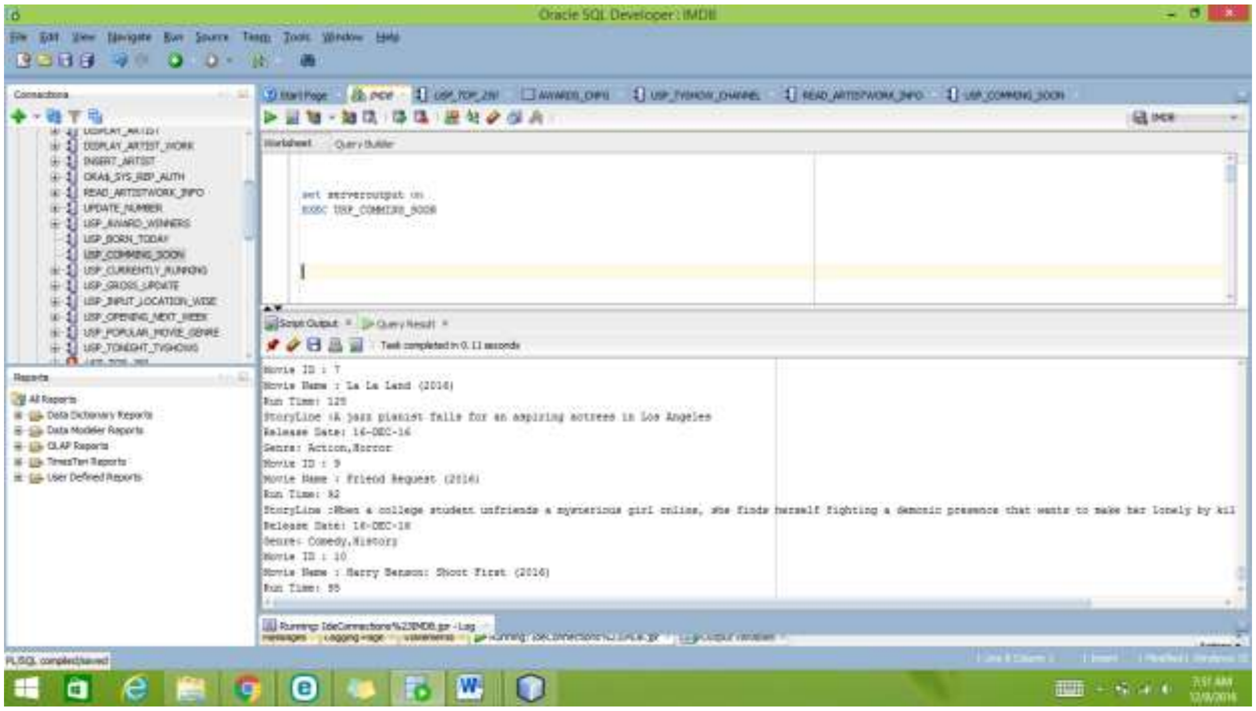
Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

INPUT:

set serveroutput on

EXEC USP_COMMING_SOON

OUTPUT:



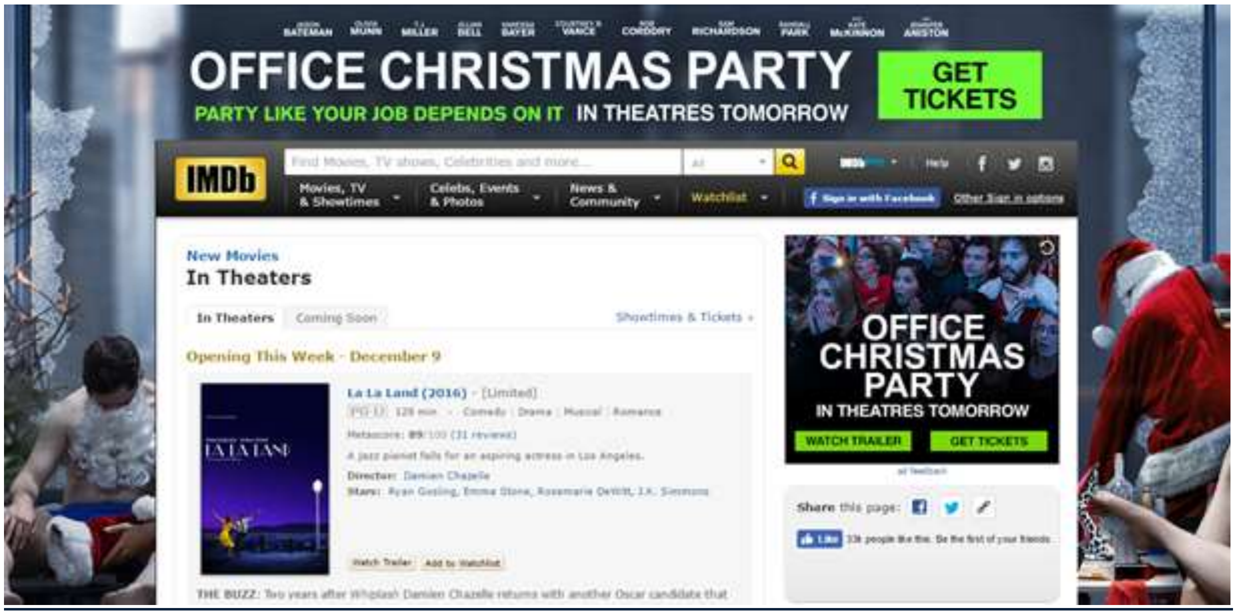
```
set serveroutput on
EXEC USP_COMMING_SOON
```

Task completed in 0.11 seconds

```
Movie ID : 7
Movie Name : La La Land (2016)
Run Time: 128
StoryLine :A jazz pianist falls for an aspiring actress in Los Angeles
Release Date: 16-DEC-16
Genre: Action,Horror
Movie ID : 9
Movie Name : Friend Request (2016)
Run Time: 92
StoryLine :When a college student unfriends a mysterious girl online, she finds herself fighting a demonic presence that wants to make her lonely by kil
Release Date: 16-DEC-18
Genre: Comedy,History
Movie ID : 10
Movie Name : Harry Benson: Shoot First (2016)
Run Time: 95
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

4.Display Movie List Opening This Week:



New Movies
In Theaters

In Theaters | Coming Soon | Showtimes & Tickets »

Opening This Week - December 9

La La Land (2016) - [Limited]
PG-13 | 128 min - Comedy | Drama | Musical | Romance
Metascore: 89/100 (31 reviews)
A jazz pianist falls for an aspiring actress in Los Angeles.
Director: Damien Chazelle
Stars: Ryan Gosling, Emma Stone, Rosemarie DeWitt, J.K. Simmons
Watch Trailer | Add to Watchlist

THE BUZZ: Two years after Whiplash Damien Chazelle returns with another Oscar candidate that

```
create or replace PROCEDURE USP_OPENING_NEXT_WEEK

IS

Cursor movie_cursor IS

select distinct MovieID,MovieName,Runtime ,Storyline,ReleaseDate,listagg(name,'') WITHIN GROUP
(ORDER BY MOVIENAME)  As Genre

FROM  (SELECT Movie.MovieID,MovieName,RunTime,GCnfg.Name,StoryLine,ReleaseDate from Movie
Movie

Join Movie_Genre Genre ON Movie.MovieID=Genre.MovieID

Join Genre_Cnfg GCnfg on Genre.GenreID=GCnfg.GenreID  where ReleaseDate BETWEEN  (select
sysdate from dual) AND (select sysdate + 7 from dual))

GROUP BY MovieID,MovieName,Runtime ,Storyline,ReleaseDate;

A_MovieID Movie.MovieID%TYPE;

A_MovieName  Movie.MovieName%TYPE;

A_Runtime  Movie.Runtime%TYPE;

A_Storyline  Movie.Storyline%TYPE;
```

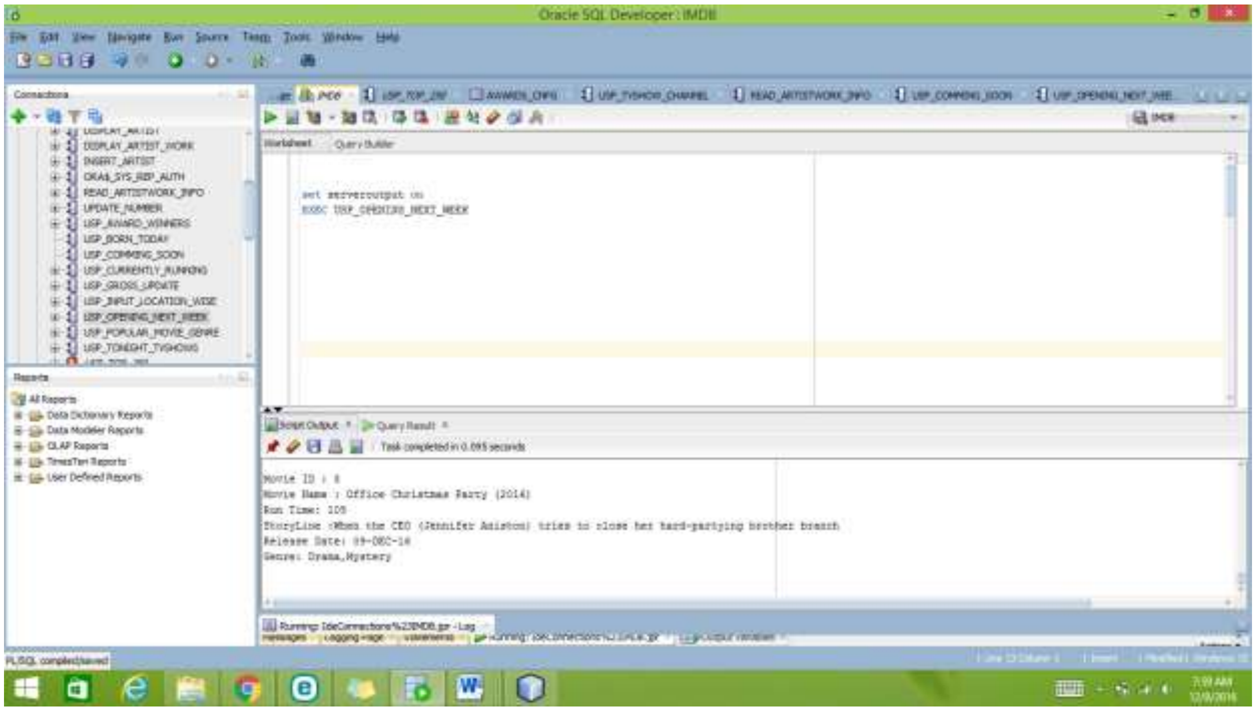
```
A_ReleaseDate  Movie.ReleaseDate%TYPE;

A_Genre  varchar2(100);

BEGIN

open movie_cursor;

loop

fetch movie_cursor into

A_MovieID,

A_MovieName,

A_Runtime,

A_Storyline,

A_ReleaseDate,

A_Genre;

exit when movie_cursor%notfound;


DBMS_OUTPUT.PUT_LINE('Movie ID : ' ||A_MovieID );

DBMS_OUTPUT.PUT_LINE('Movie Name : ' ||A_MovieName );

DBMS_OUTPUT.PUT_LINE('Run Time: ' || A_Runtime);

DBMS_OUTPUT.PUT_LINE('StoryLine :'|| A_Storyline );

DBMS_OUTPUT.PUT_LINE('Release Date: ' || A_ReleaseDate);

DBMS_OUTPUT.PUT_LINE('Genre: ' || A_Genre);

end loop;

close movie_cursor;

COMMIT;

END USP_OPENING_NEXT_WEEK;
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

INPUT:

```
set serveroutput on

EXEC USP_OPENING_NEXT_WEEK
```

OUTPUT:



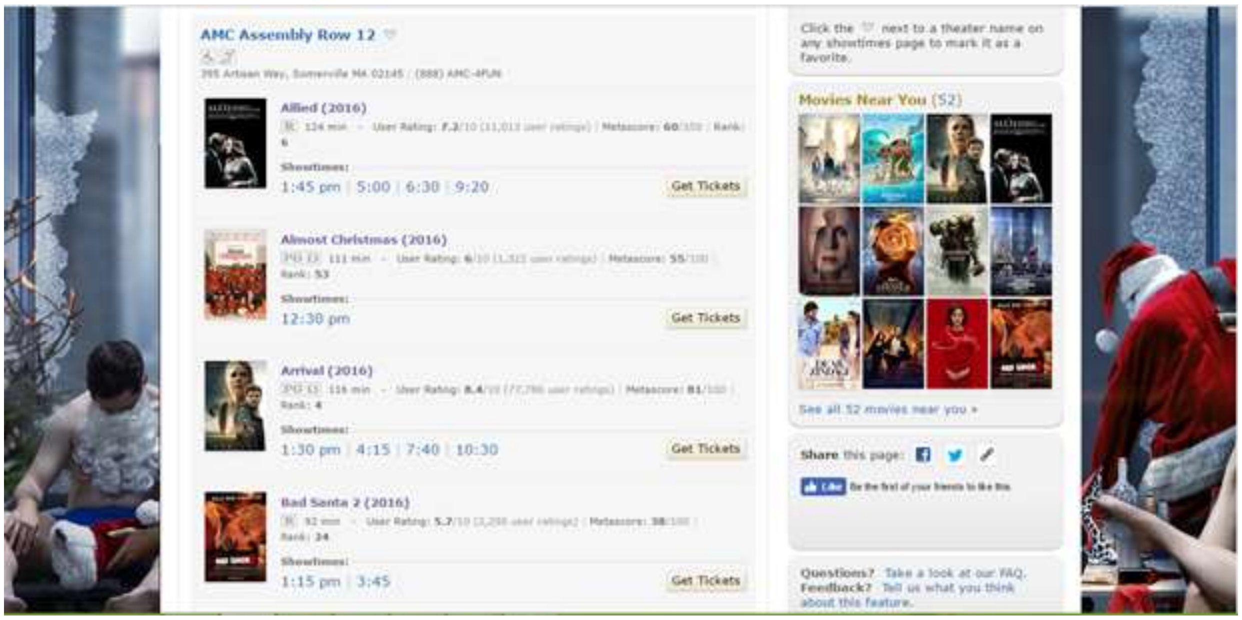
```
set serveroutput on
EXEC USP_OPENING_NEXT_WEEK
```

Task completed in 0.095 seconds

```
Movie ID : 8
Movie Name : Office Christmas Party (2016)
Run Time: 105
StoryLine :When the CEO (Jennifer Aniston) tries to close her hard-partying brother branch
Release Date: 09-DEC-16
Genre: Drama,Mystery
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

5.Display Movies Currently Running in Theatre:



AMC Assembly Row 12

295 Artisan Way, Somerville MA 02145 | (888) AMC-4FUN

Allied (2016)

R 124 min - User Rating: 7.2/10 (11,013 user ratings) | Metascore: 60/100 | Rank: 6

Showtimes:

1:45 pm | 5:00 | 6:30 | 9:20 Get Tickets

Almost Christmas (2016)

PG-13 111 min - User Rating: 6/10 (1,315 user ratings) | Metascore: 55/100 | Rank: 53

Showtimes:

12:30 pm Get Tickets

Arrival (2016)

PG-13 116 min - User Rating: 8.4/10 (77,786 user ratings) | Metascore: 81/100 | Rank: 4

Showtimes:

1:30 pm | 4:15 | 7:40 | 10:30 Get Tickets

Bad Santa 2 (2016)

R 92 min - User Rating: 5.7/10 (3,298 user ratings) | Metascore: 38/100 | Rank: 24

Showtimes:

1:15 pm | 3:45 Get Tickets

Click the ♡ next to a theater name on any showtimes page to mark it as a favorite.

Movies Near You (52)

See all 52 movies near you »

Share this page:

Like Be the first of your friends to like this.

Questions? Take a look at our FAQ.
Feedback? Tell us what you think about this feature.

create or replace PROCEDURE USP_CURRENTLY_RUNNING

IS

Cursor theartre_cursor IS

SELECT theatreID ,theatrename,theateraddress,moviename,runtime,metacriticreview,movieratingname,listagg(timeslot,'') WITHIN GROUP (ORDER BY MOVIENAME)

FROM (select theatre.TheatreID,theatre.TheatREName,

RTRIM(fr.AddressLine)|| ','||RTRIM(fr.City)|| ','||RTRIM(fs.StateName)|| ','||RTRIM(fc.Name)||','||RTRIM(fr.PostalCode) AS TheaterAddress,

movie.MovieName,movie.RunTime,movie.MetacriticReview,mr.MovieRatingName,ts.Timeslot from THEATRE

Join Theatre_Movie_Showtime tms on theatre.TheatreID=tms.TheatreID

Join Movie on movie.MovieID=tms.MovieID

Join Showtimes ts on ts.ShowtimesID=tms.ShowtimesID

Join Movie_Rating mr on mr.MovieRatingID=movie.MovieRatingID

Join Address fr on fr.AddressID=theatre.AddressID

```
Join State_Cnfg fs on fs.StateID=fr.StateID

Join Country_Cnfg fc on fc.CountryID=fr.CountryID

order by theatre.TheatreID)

GROUP BY
theatreID,theatrename,theateraddress,moviename,runtime,metacriticreview,movieratingname;

A_theatreID THEATRE.theatreID%TYPE;

A_theatreName  THEATRE.theatreName%TYPE;

A_Address  varchar2(200);

A_CityName  ADDRESS.City%TYPE;

A_MovieName  movie.MovieName%TYPE;

A_RunTime  movie.RunTime%TYPE;

A_MetacriticReview   movie.MetacriticReview%TYPE;

A_MovieRatingName  Movie_Rating.MovieRatingName%TYPE;

A_Timeslot varchar2(200) ;

BEGIN

open theartre_cursor;

loop

fetch theartre_cursor into
A_theatreID,A_theatreName,A_Address,A_MovieName,A_RunTime,A_MetacriticReview,
A_MovieRatingName,A_Timeslot;

exit when theartre_cursor%notfound;

DBMS_OUTPUT.PUT_LINE('Theatre ID : ' ||A_theatreID );

DBMS_OUTPUT.PUT_LINE('Theatre Name : ' ||A_theatreName );

DBMS_OUTPUT.PUT_LINE('Theatre AddressLine: ' || A_Address);

DBMS_OUTPUT.PUT_LINE('Movie Information :' );

DBMS_OUTPUT.PUT_LINE('Movie Name: ' || A_MovieName);

DBMS_OUTPUT.PUT_LINE('Run Time: ' || A_RunTime);

DBMS_OUTPUT.PUT_LINE('Metacritic Review: ' || A_MetacriticReview);
```

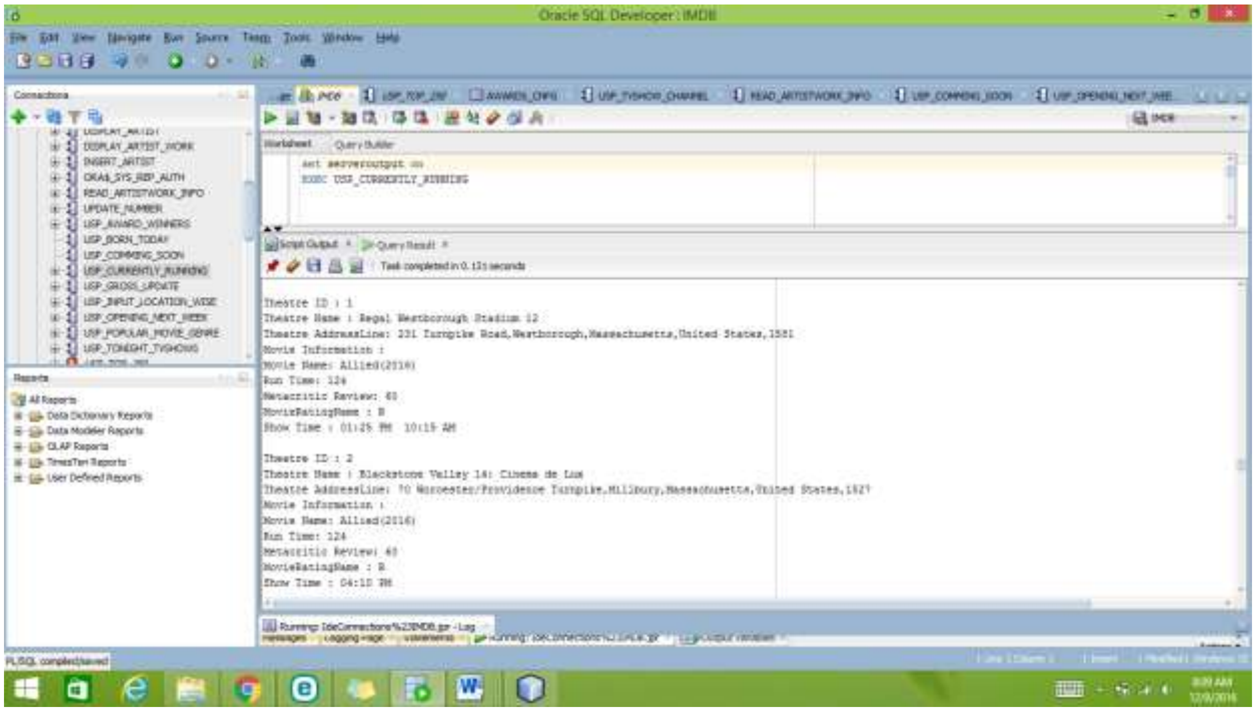
Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

```
DBMS_OUTPUT.PUT_LINE('MovieRatingName : ' || A_MovieRatingName);

DBMS_OUTPUT.PUT_LINE('Show Time : ' || A_Timeslot);

end loop;

close theartre_cursor;

COMMIT;

END USP_CURRENTLY_RUNNING;
```

INPUT:

```
set serveroutput on

EXEC USP_CURRENTLY_RUNNING
```

OUTPUT:

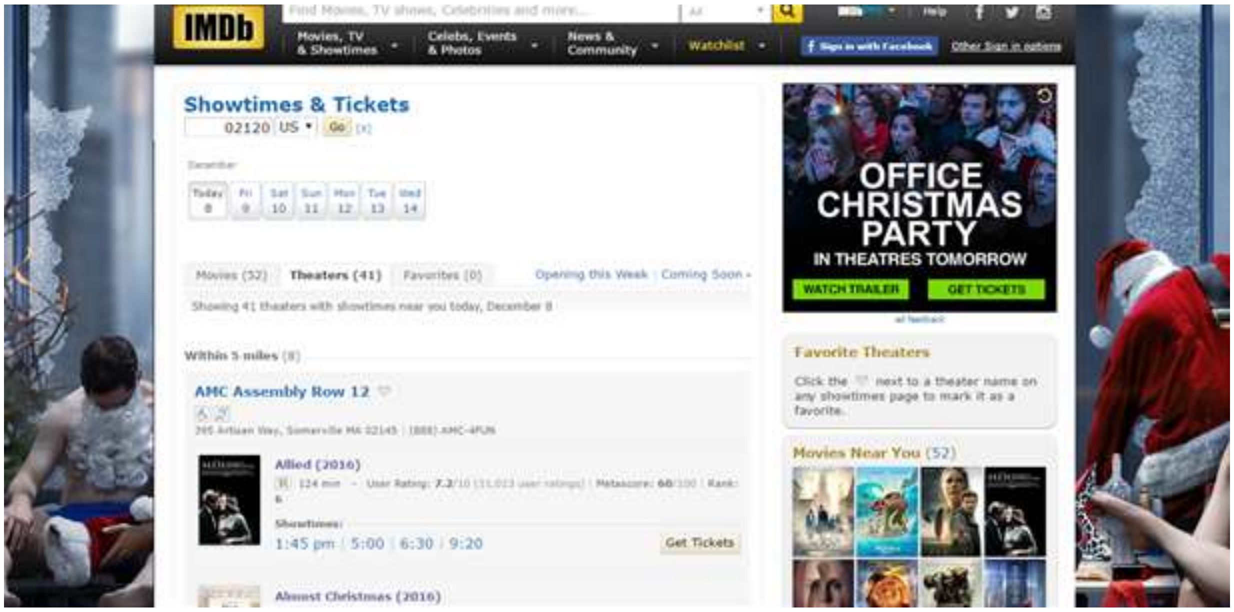


```
set serveroutput on
EXEC USP_CURRENTLY_RUNNING
```

```
Theatre ID : 1
Theatre Name : Regal Westborough Stadium 12
Theatre AddressLine: 231 Turnpike Road,Westborough,Massachusetts,United States,1581
Movie Information :
Movie Name: Allied(2016)
Run Time: 124
Metacritic Review: 60
MovieRatingName : R
Show Time : 01:25 PM  10:15 AM

Theatre ID : 2
Theatre Name : Blackstone Valley 14: Cinema de Lux
Theatre AddressLine: 70 Worcester/Providence Turnpike,Millbury,Massachusetts,United States,1527
Movie Information :
Movie Name: Allied(2016)
Run Time: 124
Metacritic Review: 60
MovieRatingName : R
Show Time : 04:10 PM
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

6.Display Theater and Movie List Based on ZIP CODE And Country As Input:



```
create or replace PROCEDURE USP_INPUT_LOCATION_WISE

(zipCode IN ADDRESS.PostalCode%TYPE,


country_name IN Country_Cnfg.Name%TYPE)

IS

Cursor theartre_cursor IS


SELECT theatreID
,theatrename,theateraddress,moviename,runtime,metacriticreview,movieratingname,listagg(timeslot,''
) WITHIN GROUP (ORDER BY MOVIENAME)

FROM  (

select theatre.TheatreID,theatre.TheatREName,

RTRIM(fr.AddressLine)|| ','||RTRIM(fr.City)|| ','||RTRIM(fs.StateName)||
','||RTRIM(fc.Name)||','||RTRIM(fr.PostalCode) AS TheaterAddress,
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

```
 movie.MovieName,movie.RunTime,movie.MetacriticReview,mr.MovieRatingName,ts.Timeslot from
THEATRE

Join Theatre_Movie_Showtime tms on  theatre.TheatreID=tms.TheatreID

Join Movie  on movie.MovieID=tms.MovieID

Join Showtimes ts on ts.ShowtimesID=tms.ShowtimesID

Join Movie_Rating mr on mr.MovieRatingID=movie.MovieRatingID

Join Address fr on fr.AddressID=theatre.AddressID

Join State_Cnfg fs on fs.StateID=fr.StateID

Join Country_Cnfg fc on fc.CountryID=fr.CountryID

where fr.PostalCode=zipCode  AND fc.Name=country_Name

order by theatre.TheatreID)

GROUP BY
theatreID,theatrename,theateraddress,moviename,runtime,metacriticreview,movieratingname;


A_theatreID THEATRE.theatreID%TYPE;

A_theatreName  THEATRE.theatreName%TYPE;

A_Address  varchar2(200);

A_MovieName  movie.MovieName%TYPE;

A_RunTime  movie.RunTime%TYPE;

A_MetacriticReview   movie.MetacriticReview%TYPE;

A_MovieRatingName  Movie_Rating.MovieRatingName%TYPE;

A_Timeslot varchar2(200) ;


BEGIN

open theartre_cursor;

loop
```

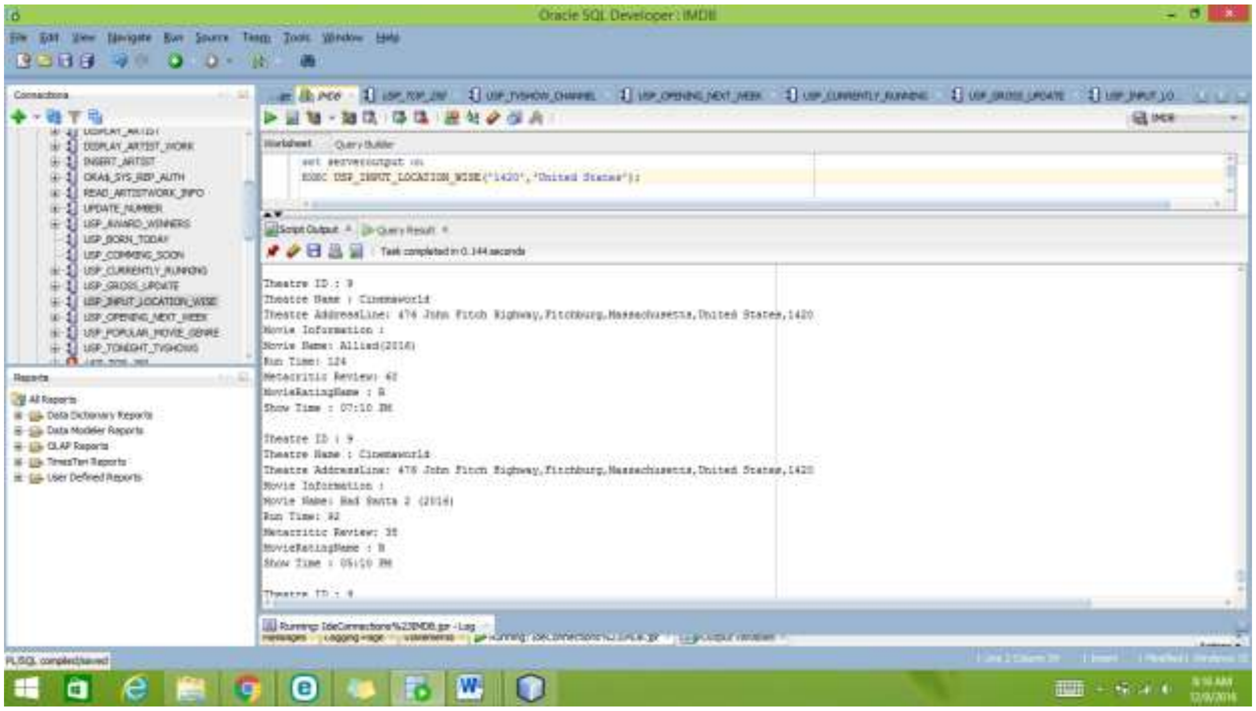
```
fetch theartre_cursor into
A_theatreID,A_theatreName,A_Address,A_MovieName,A_RunTime,A_MetacriticReview,
A_MovieRatingName,A_Timeslot;

exit when theartre_cursor%notfound;


DBMS_OUTPUT.PUT_LINE('Theatre ID : ' ||A_theatreID );

DBMS_OUTPUT.PUT_LINE('Theatre Name : ' ||A_theatreName );

DBMS_OUTPUT.PUT_LINE('Theatre AddressLine: ' || A_Address);

DBMS_OUTPUT.PUT_LINE('Movie Information :' );

DBMS_OUTPUT.PUT_LINE('Movie Name: ' || A_MovieName);

DBMS_OUTPUT.PUT_LINE('Run Time: ' || A_RunTime);

DBMS_OUTPUT.PUT_LINE('Metacritic Review: ' || A_MetacriticReview);

DBMS_OUTPUT.PUT_LINE('MovieRatingName : ' || A_MovieRatingName);

DBMS_OUTPUT.PUT_LINE('Show Time : ' || A_Timeslot|| chr(10) );


end loop;

close theartre_cursor;

COMMIT;

END USP_INPUT_LOCATION_WISE;
```

INPUT:

```
set serveroutput on

EXEC USP_INPUT_LOCATION_WISE('1420','United States');
```

OUTPUT:

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

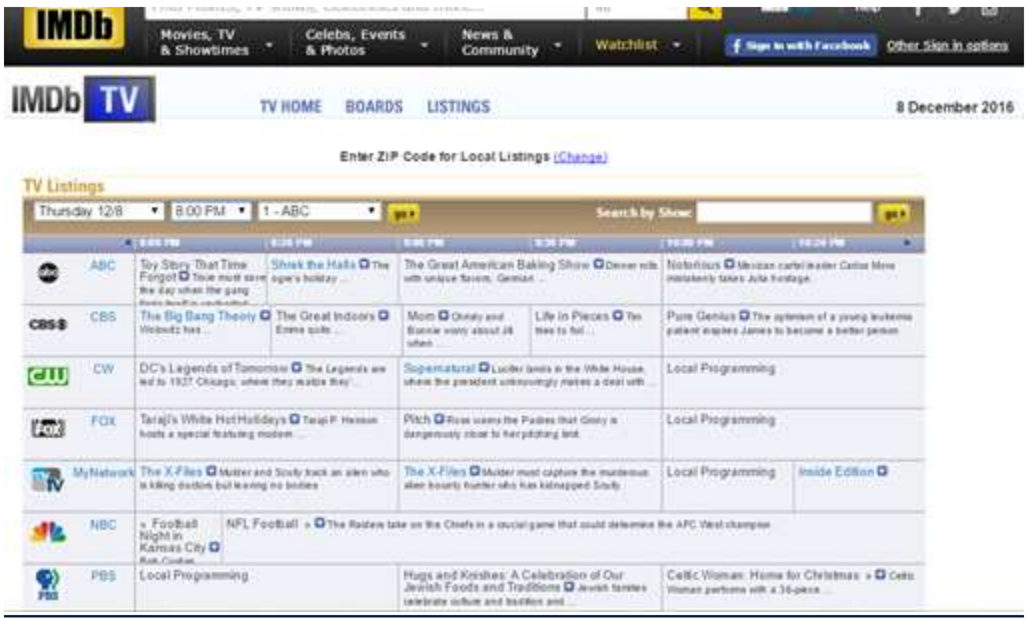


```
set serveroutput on
EXEC USP_INPUT_LOCATION_WISE('1420','United States');
```

Task completed in 0.144 seconds

```
Theatre ID : 9
Theatre Name : Cinemaworld
Theatre AddressLine: 476 John Fitch Highway,Fitchburg,Massachusetts,United States,1420
Movie Information :
Movie Name: Allied(2016)
Run Time: 124
Metacritic Review: 62
MovieRatingName : R
Show Time : 07:10 PM

Theatre ID : 9
Theatre Name : Cinemaworld
Theatre AddressLine: 476 John Fitch Highway,Fitchburg,Massachusetts,United States,1420
Movie Information :
Movie Name: Bad Santa 2 (2016)
Run Time: 92
Metacritic Review: 35
MovieRatingName : R
Show Time : 05:10 PM

Theatre ID : 9
```

7. Display TV Shows and Time Based on Input Date,Time,Channel Name :



8 December 2016

Enter ZIP Code for Local Listings (Change)

TV Listings

Thursday 12/8 | 8:00 PM | 1 - ABC | Search by Show:

	8:00 PM	8:30 PM	9:00 PM	9:30 PM	10:00 PM	10:30 PM
ABC	Toy Story That Time Forgot	Shrek the Halls	The Great American Baking Show		Notorious	
CBS	The Big Bang Theory	The Great Indoors	Mom	Life in Pieces	Pure Genius	
CW	DC's Legends of Tomorrow		Supernatural		Local Programming	
FOX	Taraji's White Hot Holidays		Pitch		Local Programming	
MyNetwork	The X-Files		The X-Files		Local Programming	Inside Edition
NBC	Football Night in Kansas City	NFL Football				
PBS	Local Programming		Hugs and Knishes: A Celebration of Our Jewish Foods and Traditions		Celtic Woman: Home for Christmas	

create or replace PROCEDURE USP_TVSHOW_CHANNEL

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

```
(Input_Date IN Channel_TV_Show_Time_Slot.SHOWDate%TYPE,
input_time IN TIMESLOT.SLOT%Type,
Input_Channnel IN Channels.ChannelName%TYPE
)
IS
Cursor TONIGHT_TVSHOW_CURSOR IS

SELECT
ts.SHOWName AS ShowName,
c.ChannelName,
ctsts.SHOWDate,
t.SLOT
 FROM TV_Shows ts
 JOIN Channel_TV_Show_Time_Slot ctsts
 ON ts.ShowID = ctsts.ShowID
 JOIN Channels c
 ON ctsts.ChannelID = c.ChannelID
 JOIN Timeslot t
 ON ctsts.TimeSlotID = t.TimeSlotID
 WHERE c.ChannelName = Input_Channnel AND ctsts.SHOWDate = Input_Date AND t.SLOT=input_time;
inpShowName  TV_Shows.SHOWName%TYPE;
inpChannelName Channels.ChannelName%TYPE;
inpDate CHANNEL_TV_SHOW_TIME_SLOT.SHOWDATE%TYPE;
inpslot TIMESLOT.SLOT%TYPE;

BEGIN
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

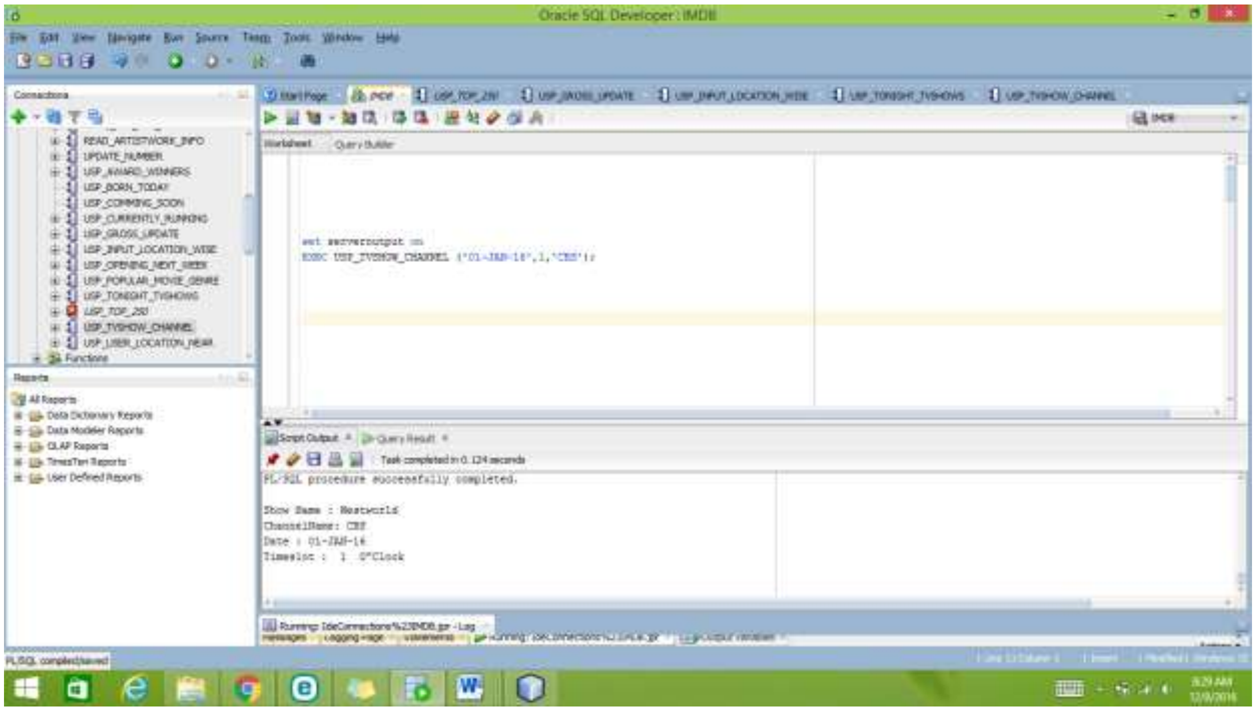
```
open TONIGHT_TVSHOW_CURSOR;

loop


fetch TONIGHT_TVSHOW_CURSOR into

inpShowName,

inpChannelName,

inpDate,

INPSLOT;

exit when TONIGHT_TVSHOW_CURSOR %notfound;

DBMS_OUTPUT.PUT_LINE('Show Name : ' || inpShowName );

DBMS_OUTPUT.PUT_LINE('ChannelName: ' || inpChannelName);

DBMS_OUTPUT.PUT_LINE('Date : ' || inpDate);

DBMS_OUTPUT.PUT_LINE('Timeslot :  ' || inpslot ||'  '||'O''Clock');

end loop;

close TONIGHT_TVSHOW_CURSOR;

COMMIT;

END USP_TVSHOW_CHANNEL;
```

INPUT:

```
set serveroutput on;

EXEC USP_TVSHOW_CHANNEL ('01-JAN-16',1,'CBS');
```

OUTPUT:

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



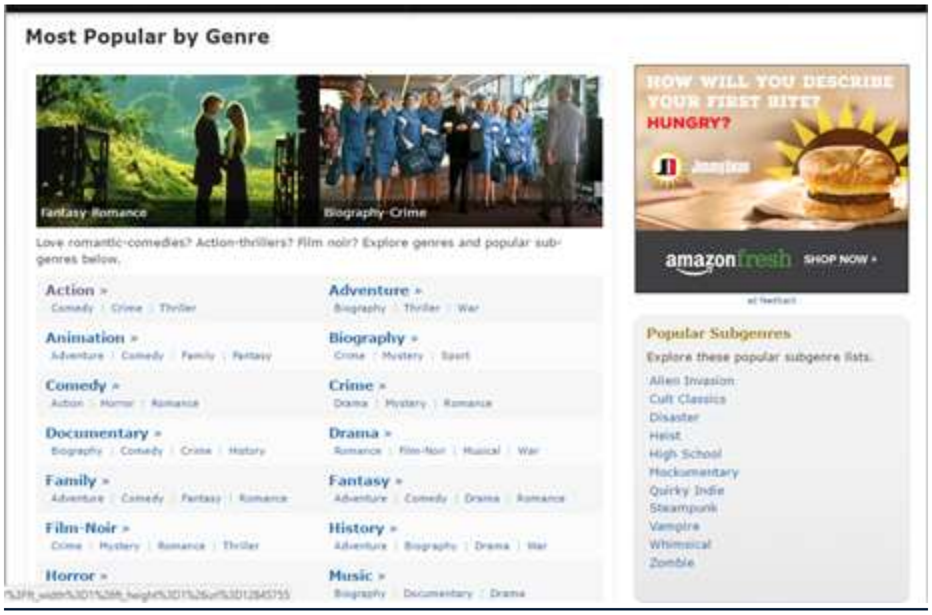
```
set serveroutput on
EXEC USP_TVSHOW_CHANNEL ('01-JAN-16',1,'CBS');
```

Task completed in 0.124 seconds

```
PL/SQL procedure successfully completed.

Show Name : Westworld
ChannelName: CBS
Date : 01-JAN-16
Timeslot :  1  O'Clock
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

8. Most Popular Movie By Genre:



Most Popular by Genre

Love romantic-comedies? Action-thrillers? Film noir? Explore genres and popular sub-genres below.

Action » Comedy | Crime | Thriller
Adventure » Biography | Thriller | War
Animation » Adventure | Comedy | Family | Fantasy
Biography » Crime | Mystery | Sport
Comedy » Action | Horror | Romance
Crime » Drama | Mystery | Romance
Documentary » Biography | Comedy | Crime | History
Drama » Romance | Film-Noir | Musical | War
Family » Adventure | Comedy | Fantasy | Romance
Fantasy » Adventure | Comedy | Drama | Romance
Film-Noir » Crime | Mystery | Romance | Thriller
History » Adventure | Biography | Drama | War
Horror »
Music » Biography | Documentary | Drama

Popular Subgenres

Explore these popular subgenre lists.

Alien Invasion
Cult Classics
Disaster
Heist
High School
Mockumentary
Quirky Indie
Steampunk
Vampire
Whimsical
Zombie

create or replace PROCEDURE USP_POPULAR_MOVIE_GENRE(Input_Genre IN GENRE_CNFG.NAME%TYPE)

IS

Cursor popular_movie_cursor IS

```
SELECT m.MovieName,
       m.MetacriticReview,
       m.StoryLine,
       gc.Name GenreName
FROM Movie m
JOIN Movie_Genre mg
ON m.MovieID = mg.MovieID
JOIN Genre_Cnfg gc
```

```
ON mg.GenreID = gc.GenreID

WHERE gc.Name = Input_Genre

ORDER BY m.Views DESC;


popMovieName movie.MovieName %TYPE;

review movie.MetacriticReview %TYPE;

story movie.StoryLine%TYPE;

genre Genre_Cnfg.name%TYPE;

BEGIN

open popular_movie_cursor;

loop

fetch popular_movie_cursor into

popMovieName,

review,

story,

genre;

exit when popular_movie_cursor %notfound;


DBMS_OUTPUT.PUT_LINE('Movie Name : ' ||popMovieName);

DBMS_OUTPUT.PUT_LINE('Metacritic Review : ' || review);

DBMS_OUTPUT.PUT_LINE('Story ' || story );

DBMS_OUTPUT.PUT_LINE('Genre:'|| genre|| chr(10) );

end loop;

close popular_movie_cursor;

COMMIT;

END USP_POPULAR_MOVIE_GENRE;
```

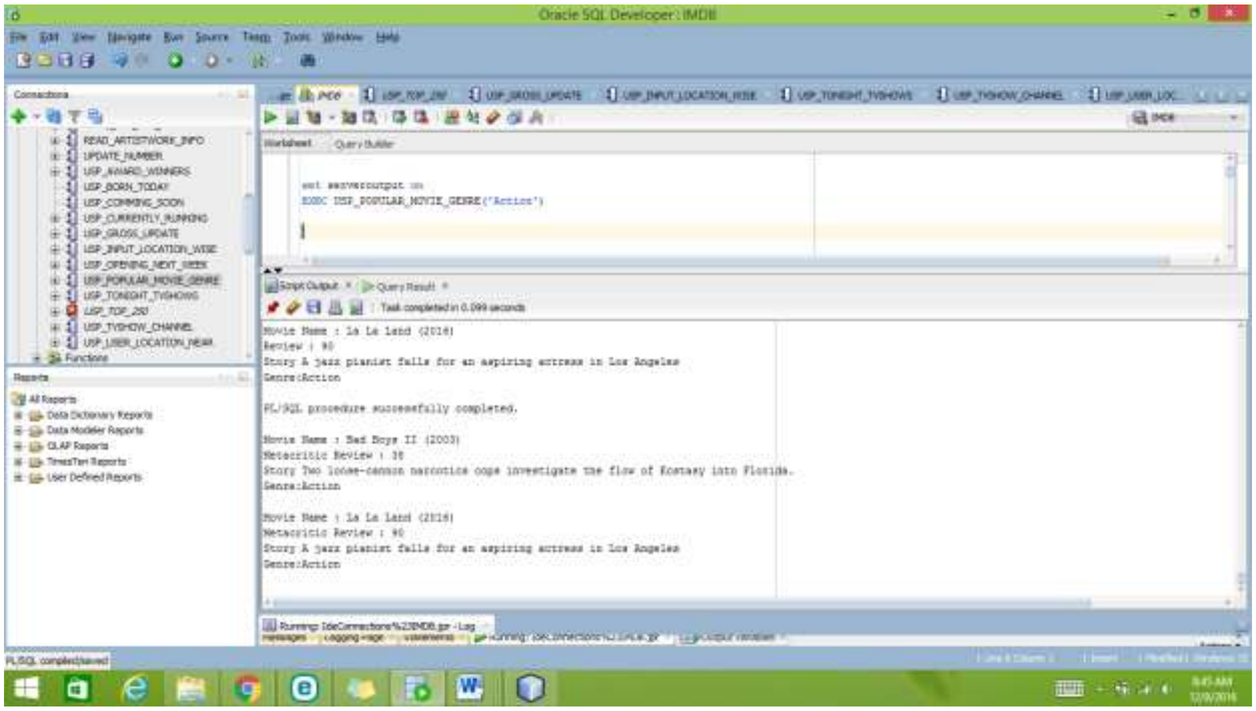
Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

INPUT:

set serveroutput on

EXEC USP_POPULAR_MOVIE_GENRE('Action')

OUTPUT:



```
set serveroutput on
EXEC USP_POPULAR_MOVIE_GENRE('Action')
```

```
Movie Name : La La Land (2016)
Review : 90
Story A jazz pianist falls for an aspiring actress in Los Angeles
Genre:Action

PL/SQL procedure successfully completed.

Movie Name : Bad Boys II (2003)
Metacritic Review : 38
Story Two loose-cannon narcotics cops investigate the flow of Ecstasy into Florida.
Genre:Action

Movie Name : La La Land (2016)
Metacritic Review : 90
Story A jazz pianist falls for an aspiring actress in Los Angeles
Genre:Action
```

9. Recently Reviewed Movie By User:

CREATE OR REPLACE PROCEDURE USP_RECENTLY_REVIEWED

(user_ID IN INTEGER)

IS

Cursor Recent_cursor IS

select
ud.USERID,FIRSTNAME,LASTNAME,EMAIL,PHONENUMBER,DATEOFBIRTH,CITY,listagg(movie.MOVIENAME,',') WITHIN GROUP (ORDER BY FIRSTNAME) As MovieList

from user_details ud

join user_review ur ON ur.USERID=ud.USERID

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

```
 join movie on movie.movieID=ur.movieID

 GROUP BY ud.USERID,FIRSTNAME,LASTNAME,EMAIL,PHONENUMBER,DATEOFBIRTH,CITY having
ud.userid=user_ID;

A_USERID user_details.userID%TYPE;

A_FIRSTNAME  user_details.FIRSTNAME%TYPE;

A_LASTNAME  user_details.LASTNAME%TYPE;

A_EMAIL  user_details.EMAIL%TYPE;

A_PHONENUMBER  user_details.PHONENUMBER%TYPE;

A_DATEOFBIRTH  user_details.DATEOFBIRTH%TYPE;

A_CITY  user_details.city%TYPE;

A_Movielist varchar2(300);


BEGIN

open Recent_cursor;

loop

fetch Recent_cursor into

A_USERID,

A_FIRSTNAME,

A_LASTNAME,

A_EMAIL,

A_PHONENUMBER,

A_DATEOFBIRTH,

A_CITY,

A_Movielist;

exit when Recent_cursor%notfound;


DBMS_OUTPUT.PUT_LINE('User Id : ' ||A_USERID );
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

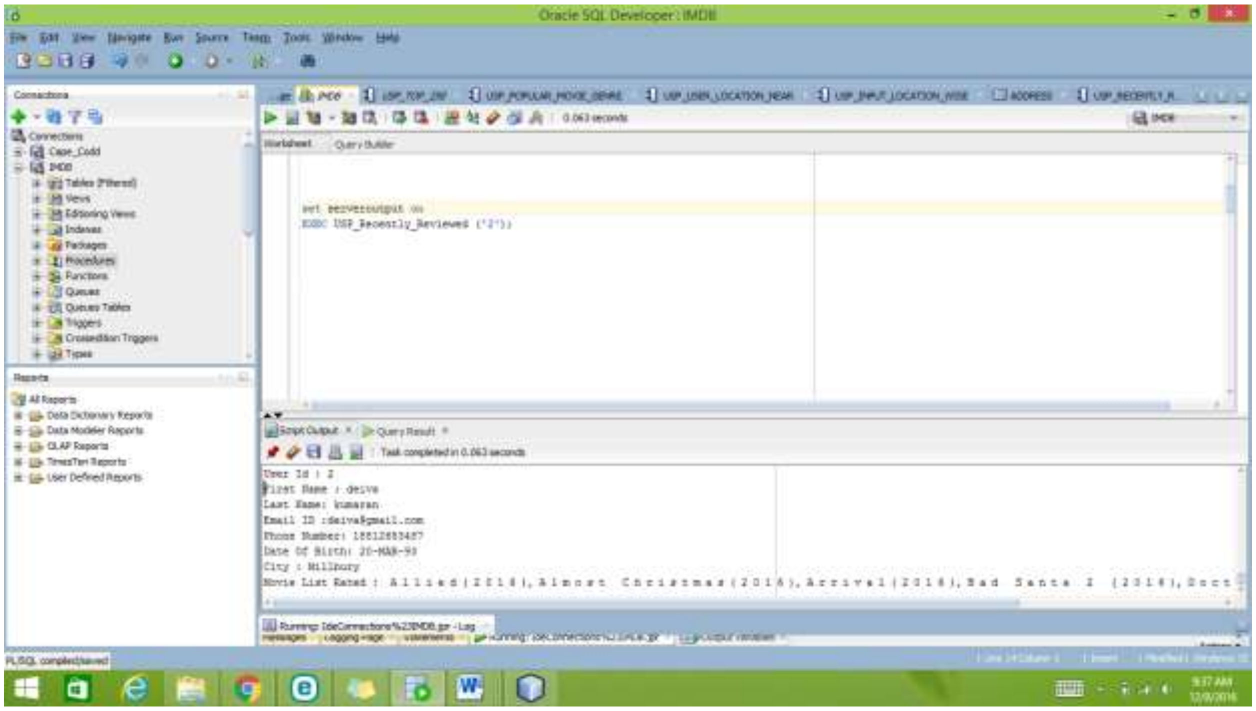
```
DBMS_OUTPUT.PUT_LINE('First Name : ' ||A_FIRSTNAME );

DBMS_OUTPUT.PUT_LINE('Last Name: ' || A_LASTNAME);

DBMS_OUTPUT.PUT_LINE('Email ID :' || A_EMAIL);

DBMS_OUTPUT.PUT_LINE('Phone Number: ' || A_PHONENUMBER);

DBMS_OUTPUT.PUT_LINE('Date Of Birth: ' || A_DATEOFBIRTH);

DBMS_OUTPUT.PUT_LINE('City : ' || A_CITY);

DBMS_OUTPUT.PUT_LINE('Movie List Rated : ' || A_Movielist);

end loop;

close Recent_cursor;

COMMIT;

END USP_RECENTLY_REVIEWED;
```

INPUT:

```
set serveroutput on

EXEC USP_Recently_Reviewed ('2');
```

OUTPUT:

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



```
set serveroutput on
EXEC USP_Recently_Reviewed ('2');
```

```
User Id : 2
First Name : deiva
Last Name: kumaran
Email ID :deiva@gmail.com
Phone Number: 18612883487
Date Of Birth: 20-MAR-93
City : Millbury
Movie List Rated :  A l l i e d ( 2 0 1 6 ), A l m o s t   C h r i s t m a s ( 2 0 1 6 ), A r r i v a l ( 2 0 1 6 ), B a d   S a n t a   2   ( 2 0 1 6 ), D o c t
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

Functions:

1.Gross Movie Profit

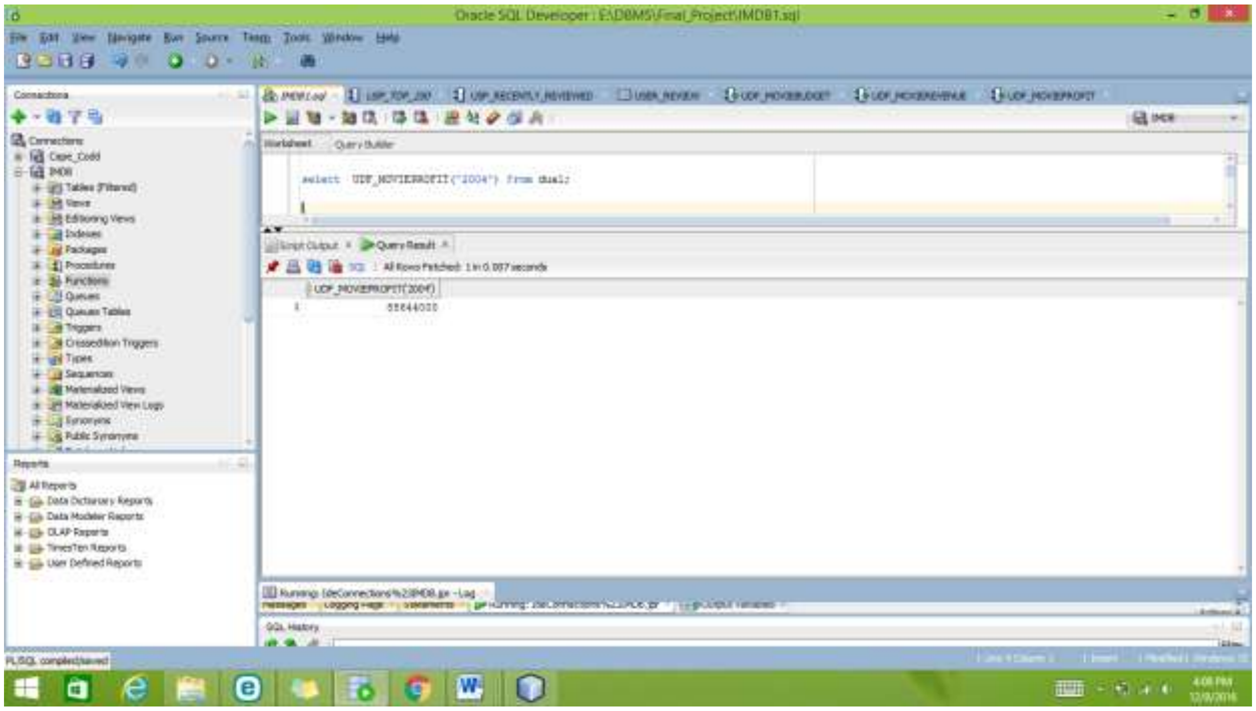
```
CREATE OR REPLACE Function udf_MovieProfit
  ( year IN INTEGER )
  RETURN float
IS
  profit float;
  cursor c1 is
    SELECT GROSS_REVENUE - BUDGET profit
    FROM MOVIE
    WHERE EXTRACT(YEAR FROM ReleaseDate) = year;
BEGIN
profit := 0;
FOR movie_rec in c1
LOOP
profit := profit + movie_rec.profit;
END LOOP;
RETURN profit;
END;
```

INPUT:

```
select  UDF_MOVIEPROFIT('2004') from dual;
```

OUTPUT:

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



```
select UDF_MOVIEPROFIT('2004') from dual;
```

```
All Rows Fetched: 1 in 0.007 seconds

  UDF_MOVIEPROFIT('2004')
1                 55644000
```

<u>2.Function To Display Gross Budget Across all Movies In Particular Year:</u>

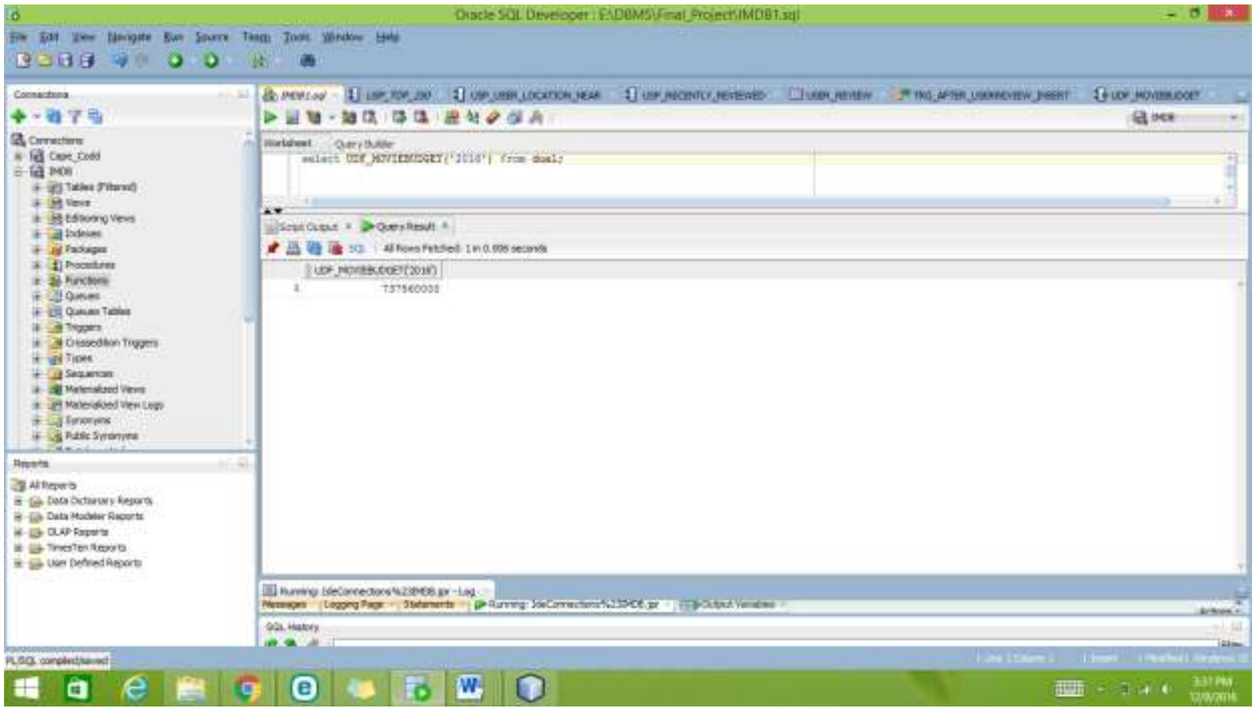
```
CREATE OR REPLACE Function udf_MovieBudget
   ( year IN INTEGER )
   RETURN float
IS
   total_budget float;

   cursor c1 is
     SELECT Budget
     FROM MOVIE
     WHERE EXTRACT(YEAR FROM ReleaseDate) = year;

BEGIN

   total_budget := 0;

   FOR movie_rec in c1
   LOOP
      total_budget := total_budget + movie_rec.Budget;
   END LOOP;

   RETURN total_budget;

END;
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

INPUT:

select UDF_MOVIEBUDGET('2016') from dual;

Output:



```
select UDF_MOVIEBUDGET('2016') from dual;
```

All Rows Fetched: 1 in 0.008 seconds

```
     UDF_MOVIEBUDGET('2016')
1                  737560000
```

3. Function To Display Gross RevenueAcross all Movies In Particular Year:

```
CREATE OR REPLACE Function udf_MovieRevenue
   ( year IN INTEGER )
   RETURN float
IS
   total_revenue float;
  cursor c1 is
     SELECT GROSS_REVENUE
     FROM MOVIE
```

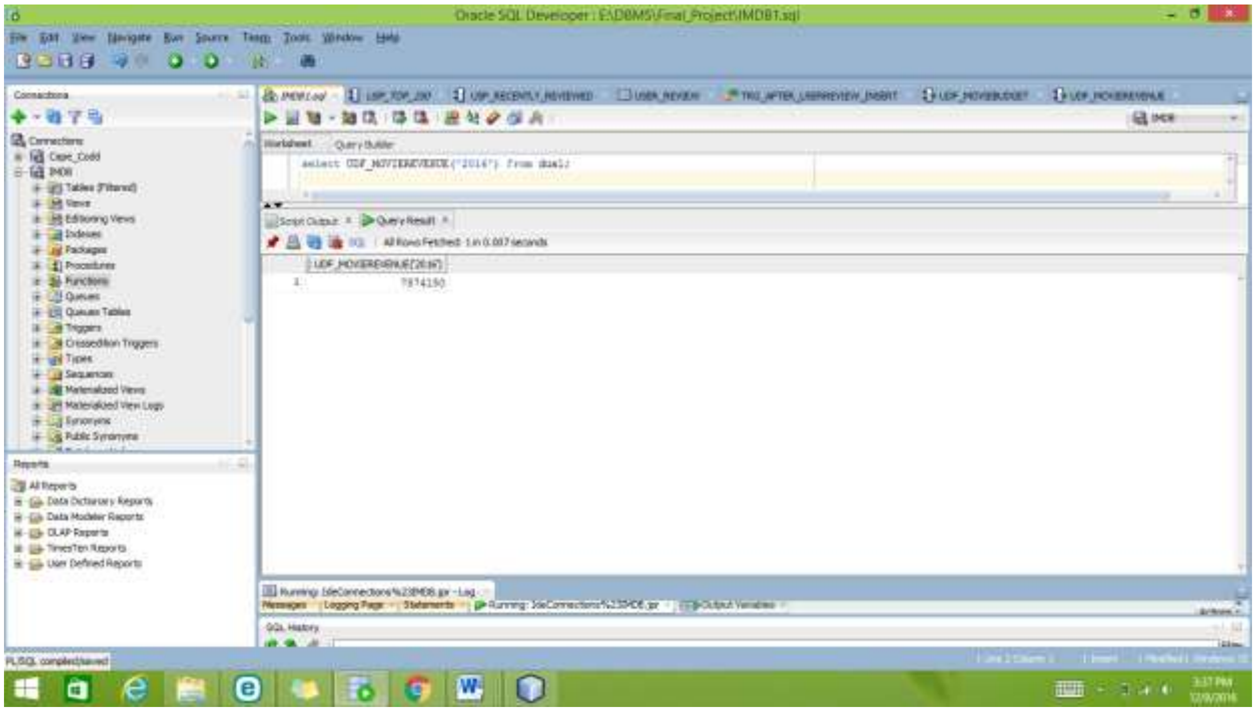
Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

```
    WHERE EXTRACT(YEAR FROM ReleaseDate) = year;
BEGIN
  total_revenue := 0;
  FOR movie_rec in c1
  LOOP
  total_revenue := total_revenue + movie_rec.Gross_Revenue;
  END LOOP;
  RETURN total_revenue;
 END;
```

INPUT:

select UDF_MOVIEREVENUE('2016') from dual;

OUTPUT:



```
select UDF_MOVIEREVENUE('2016') from dual;
```

All Rows Fetched: 1 in 0.007 seconds

```
     UDF_MOVIEREVENUE('2016')
1                     7874150
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

4.Function to count Movie List on Particular Year

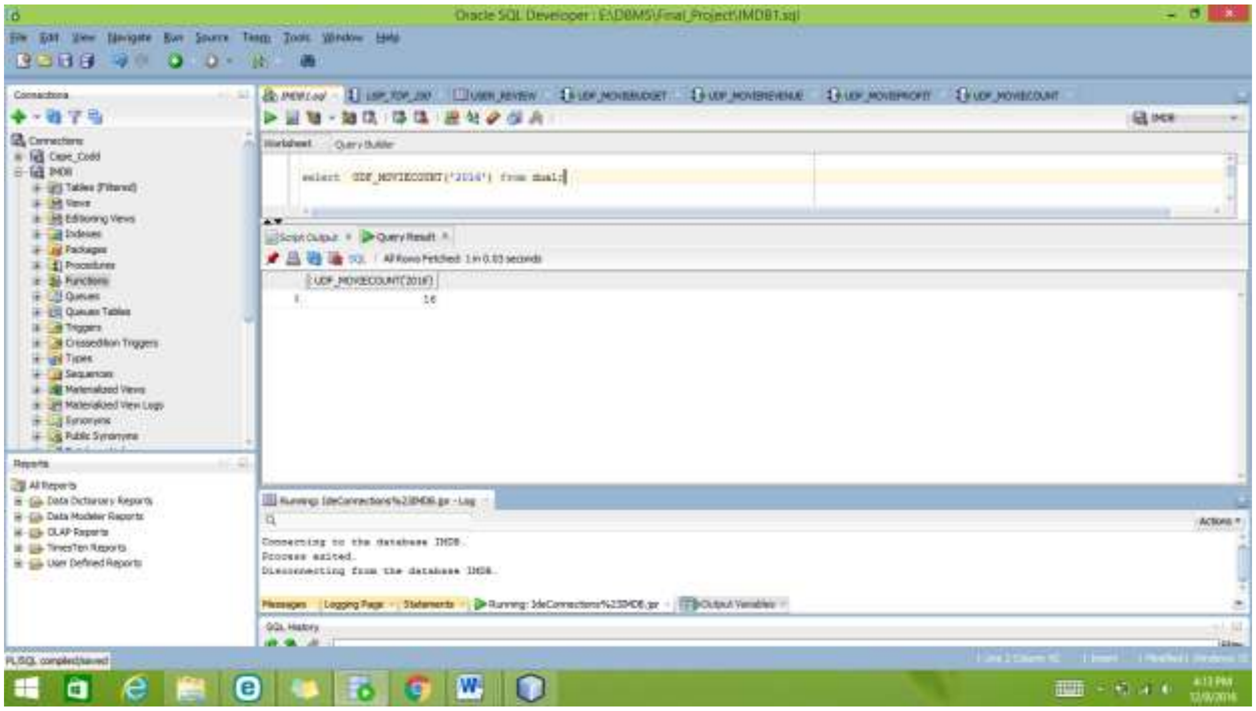
```
CREATE OR REPLACE Function udf_MovieCount

  ( year IN INTEGER )

  RETURN int

IS

  mov_count INTEGER;

  cursor c1 is

    SELECT COUNT(*) movie_count

    FROM MOVIE

    WHERE EXTRACT(YEAR FROM ReleaseDate) = year;

BEGIN

  mov_count := 0;

  FOR movie_rec in c1

  LOOP

     mov_count := mov_count + movie_rec.movie_count;

  END LOOP;

  RETURN mov_count;

END;
```

INPUT:

```
select  UDF_MOVIECOUNT('2016') from dual;
```

OUTPUT:

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



```
select UDF_MOVIECOUNT('2016') from dual;
```

All Rows Fetched: 1 in 0.03 seconds

```
  UDF_MOVIECOUNT(2016)
1                   16
```

```
Connecting to the database IMDB.
Process exited.
Disconnecting from the database IMDB.
```

<u>5.Function To display Most Popular Based on Number of view:</u>

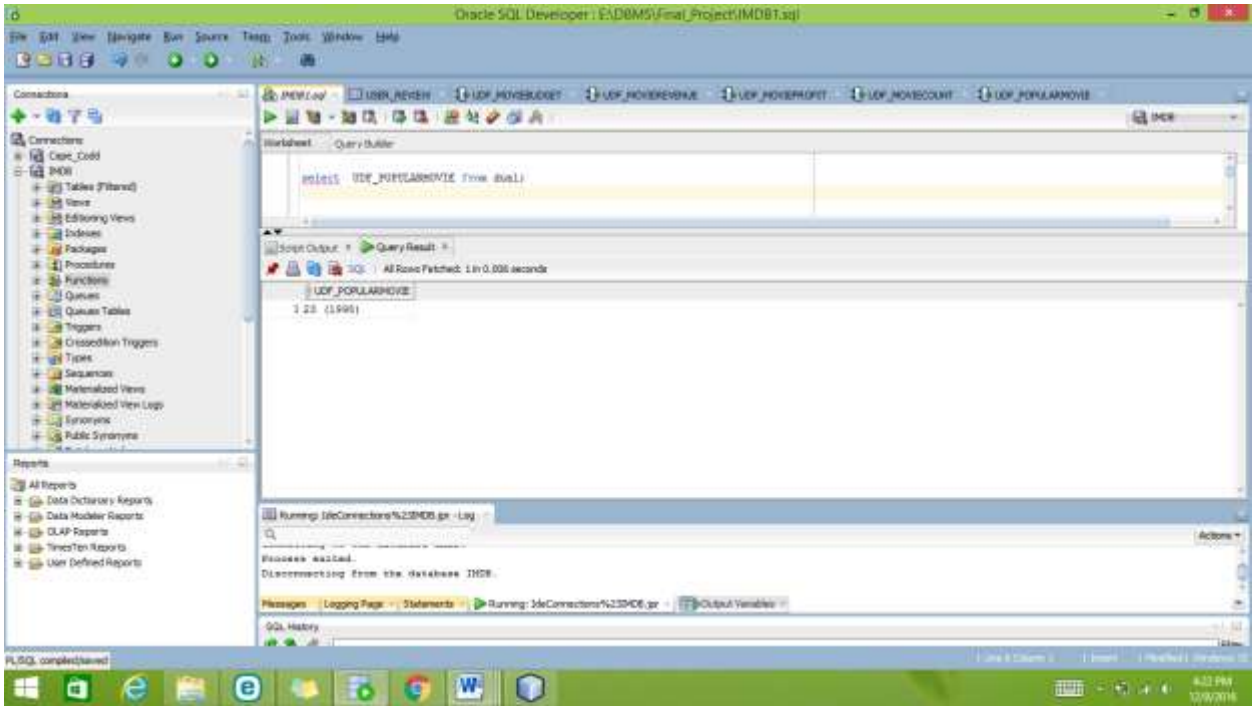
```
CREATE OR REPLACE Function udf_PopularMovie
  RETURN varchar2
IS
  movie varchar2(50);
BEGIN
    SELECT MovieName into movie
    FROM MOVIE where Views = (select max(Views) from movie);
  RETURN movie;
END;
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

INPUT:

select UDF_POPULARMOVIE from dual;

OUTPUT:



```
select  UDF_POPULARMOVIE from dual;
```

All Rows Fetched: 1 in 0.006 seconds

```
   UDF_POPULARMOVIE
1  23 (1998)
```

```
Process exited.
Disconnecting from the database IMDB.
```

<u>6.Function To display Best Based on Metacritic Review:</u>

```
CREATE OR REPLACE Function udf_BestMovie

   RETURN varchar2

IS

   movie varchar2(50);


BEGIN


     SELECT MovieName into movie
```

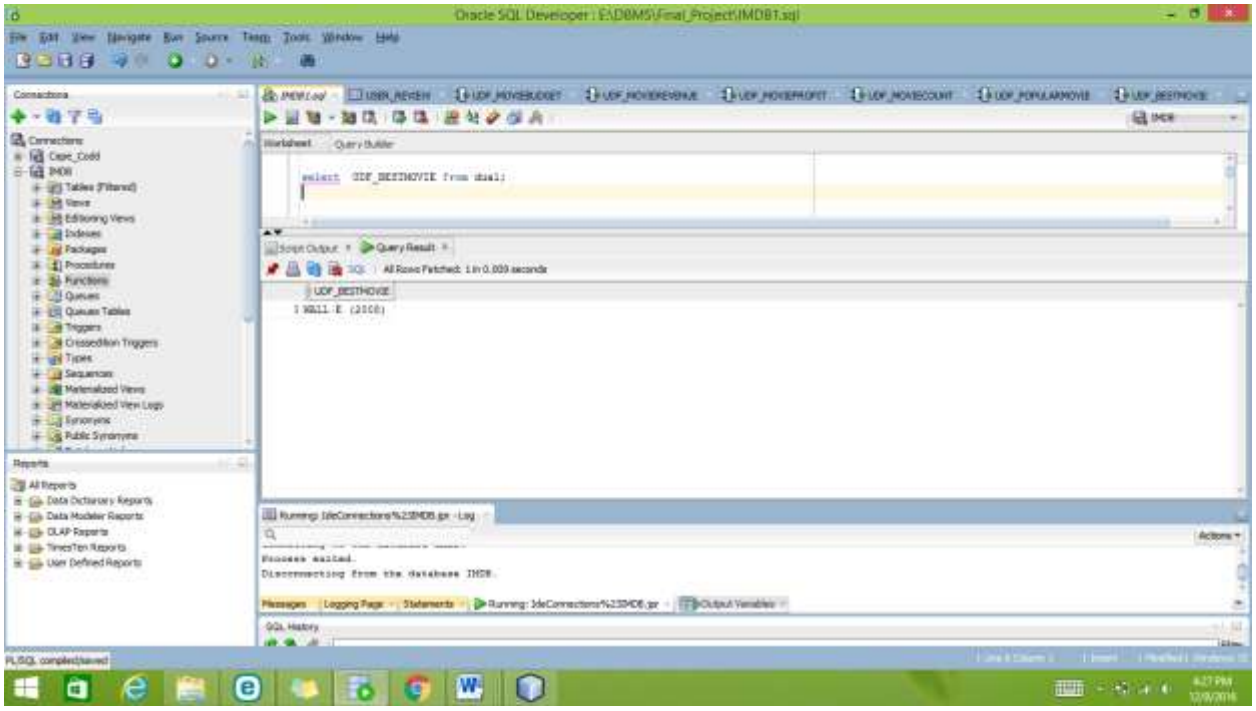
Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

```
   FROM MOVIE where METACRITICREVIEW = (select max(METACRITICREVIEW) from movie);


  RETURN movie;


END;
```

INPUT:

```
select  UDF_BESTMOVIE from dual;
```

OUTPUT:



```
select  UDF_BESTMOVIE from dual;
```

All Rows Fetched: 1 in 0.009 seconds

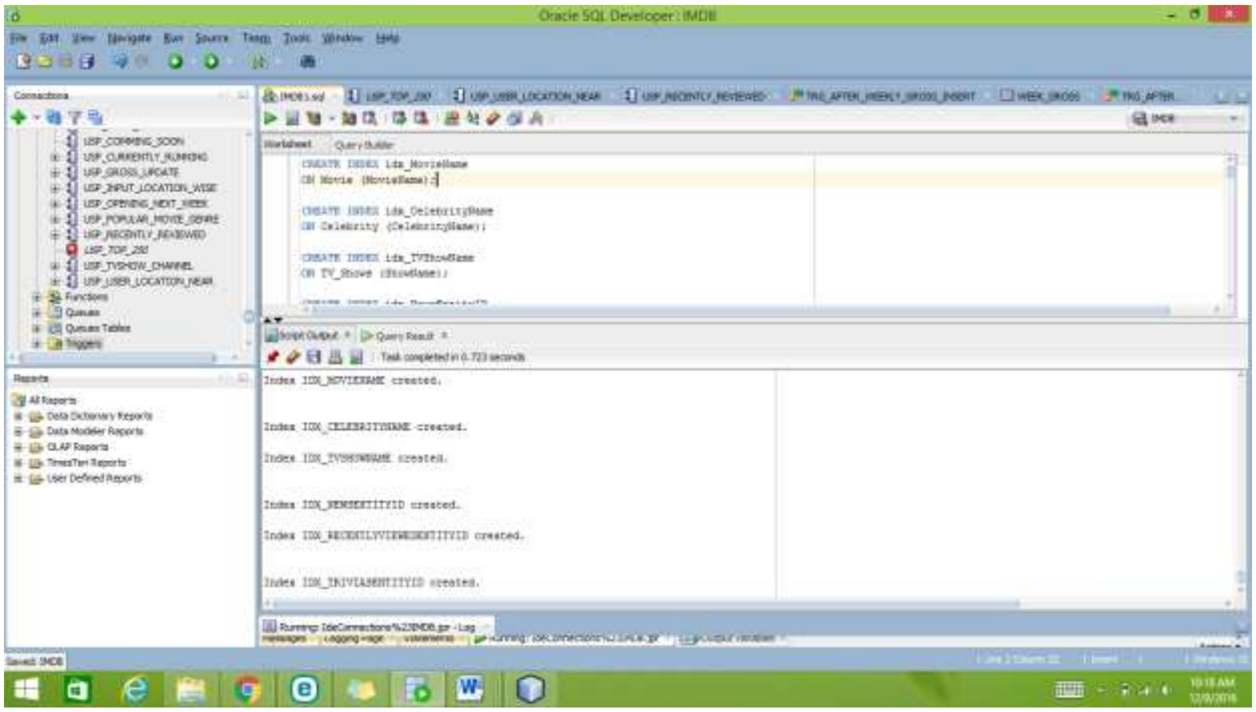
```
  UDF_BESTMOVIE
1 WALL·E (2008)
```

```
Process exited.
Disconnecting from the database IMDB.
```

INDEXES:

```
CREATE INDEX idx_MovieName
ON Movie (MovieName);

CREATE INDEX idx_CelebrityName
ON Celebrity (CelebrityName);

CREATE INDEX idx_TVShowName
ON TV_Shows (ShowName);
```

```
CREATE INDEX idx_NewsEntityID
ON News (EntityID);

CREATE INDEX idx_RecentlyViewedEntityID
ON Recently_Viewed (EntityID);

CREATE INDEX idx_TriviasEntityID
ON Trivias (EntityID);

CREATE INDEX idx_SoundtracksEntityID
ON Soundtracks (EntityID);

CREATE INDEX idx_VideosEntityID
ON Videos (EntityID);

CREATE INDEX idx_AmazonVideosEntityID
ON Amazon_Videos (EntityID);
```



```
CREATE INDEX idx_MovieName
ON Movie (MovieName);

CREATE INDEX idx_CelebrityName
ON Celebrity (CelebrityName);

CREATE INDEX idx_TVShowName
ON TV_Shows (ShowName);
```

```
Index IDX_MOVIENAME created.


Index IDX_CELEBRITYNAME created.

Index IDX_TVSHOWNAME created.


Index IDX_NEWSENTITYID created.

Index IDX_RECENTLYVIEWEDENTITYID created.


Index IDX_TRIVIASENTITYID created.
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

TRIGGER

1.TO UPDATE GROSS REVENUE FROM WEEKLY GROSS:

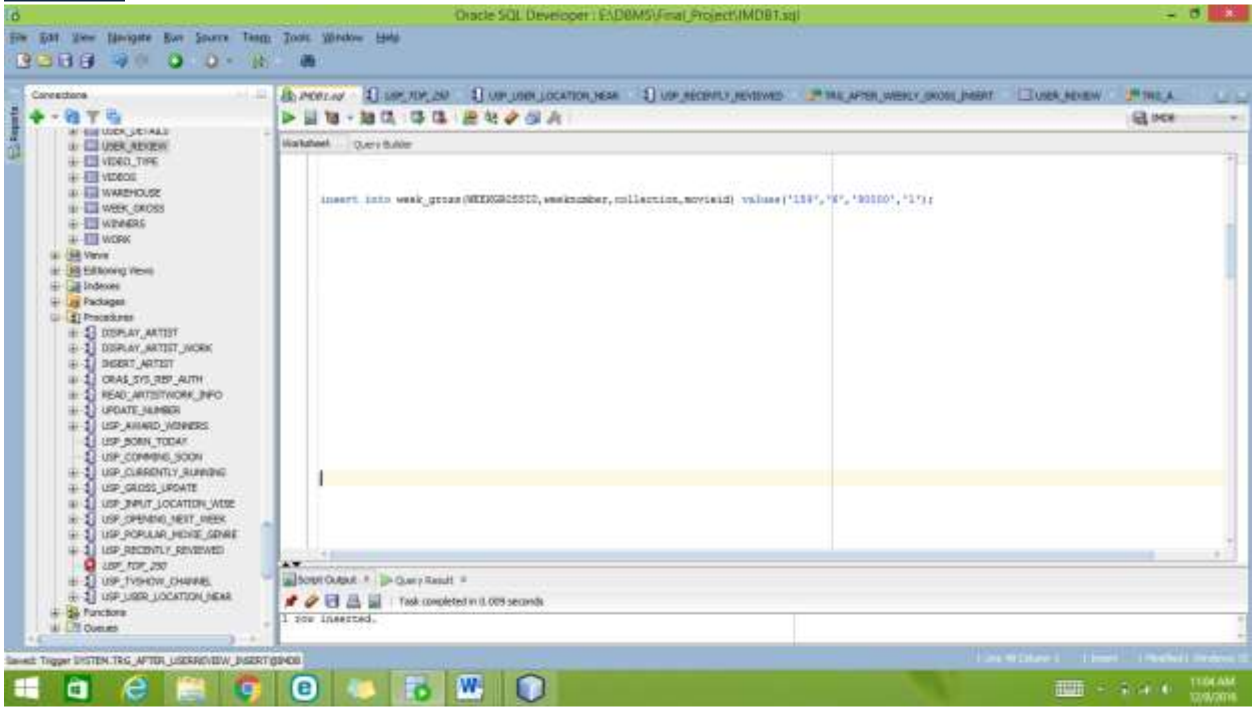
```
CREATE OR REPLACE TRIGGER TRG_AFTER_WEEKLY_GROSS_INSERT

AFTER INSERT ON Week_Gross

FOR EACH ROW

DECLARE

 mID int;

 gross float;

 TotalGross float;


BEGIN


   gross := :new.Collection;


   select Gross_Revenue into TotalGross from Movie where MovieID = :new.MovieID;


   UPDATE Movie SET Gross_Revenue = TotalGross+gross WHERE MovieID = :new.MovieID;


END;
```

INPUT:

```
insert into week_gross(WEEKGROSSID,weeknumber,collection,movieid) values('158','5','90000','1');
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

<u>OUTPUT:</u>



```
insert into week_gross(WEEKGROSSID,weeknumber,collection,movieid) values('159','4','80000','1');
```

Task completed in 0.009 seconds

```
1 row inserted.
```

<u>2.INSERT AFTER TRIGGER TO UPDATE IMDB_RATING Based on User Rating:</u>

```
CREATE OR REPLACE TRIGGER TRG_AFTER_USERREVIEW_INSERT

AFTER INSERT

  ON User_Review

  FOR EACH ROW


DECLARE

  mID int;

  rating float;


BEGIN

 select AVG(IMDB_Rating) into rating from Movie where MovieID = :new.MovieID;

  rating := rating + :new.Rating;

  rating := rating/2;
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

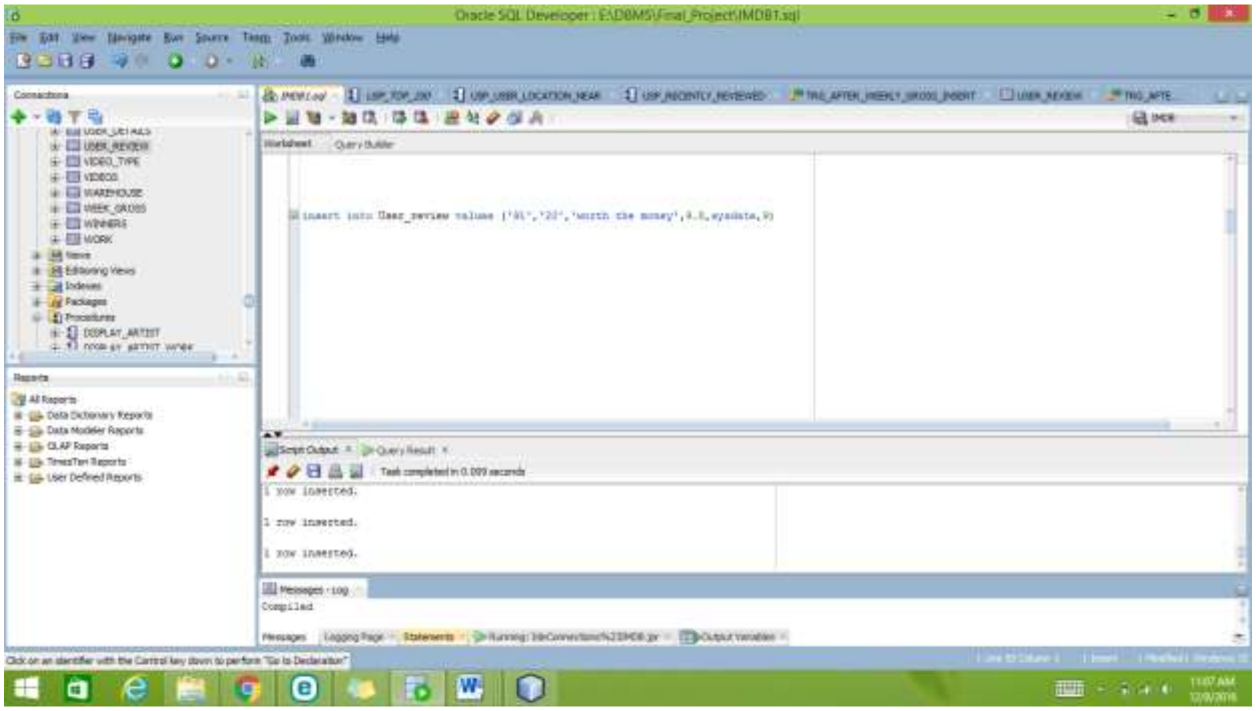
UPDATE Movie SET IMDB_Rating = rating WHERE MovieID = :new.MovieID;

END;

INPUT:

insert into User_review values ('91','20','worth the money',9.8,sysdate,9);

OUTPUT:



```
insert into User_review values ('91','20','worth the money',9.8,sysdate,9)
```

Task completed in 0.099 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.
```

```
Compiled
```

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

QUERIES:

1. Top amazon videos by views

SELECT

CASE EntityTypeID

WHEN 1 THEN MovieName

ELSE ShowName

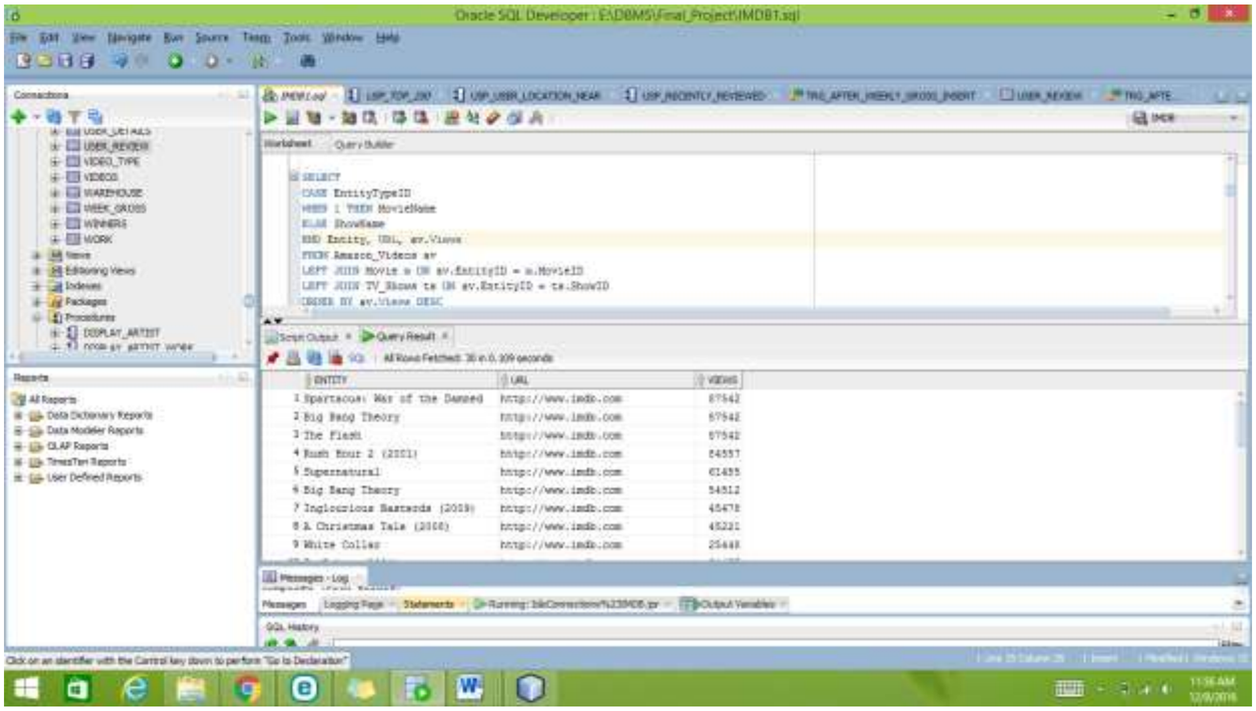
END Entity, URL, av.Views

FROM Amazon_Videos av

LEFT JOIN Movie m ON av.EntityID = m.MovieID

LEFT JOIN TV_Shows ts ON av.EntityID = ts.ShowID

ORDER BY av.Views DESC



```
All Rows Fetched: 30 in 0.309 seconds

  ENTITY                           URL                    VIEWS
1 Spartacus: War of the Damned     http://www.imdb.com    87542
2 Big Bang Theory                  http://www.imdb.com    87542
3 The Flash                        http://www.imdb.com    87542
4 Rush Hour 2 (2001)               http://www.imdb.com    84557
5 Supernatural                     http://www.imdb.com    61455
6 Big Bang Theory                  http://www.imdb.com    54512
7 Inglourious Basterds (2009)      http://www.imdb.com    45478
8 A Christmas Tale (2008)          http://www.imdb.com    45221
9 White Collar                     http://www.imdb.com    25448
```

2. Top amazon videos by IMDB rating

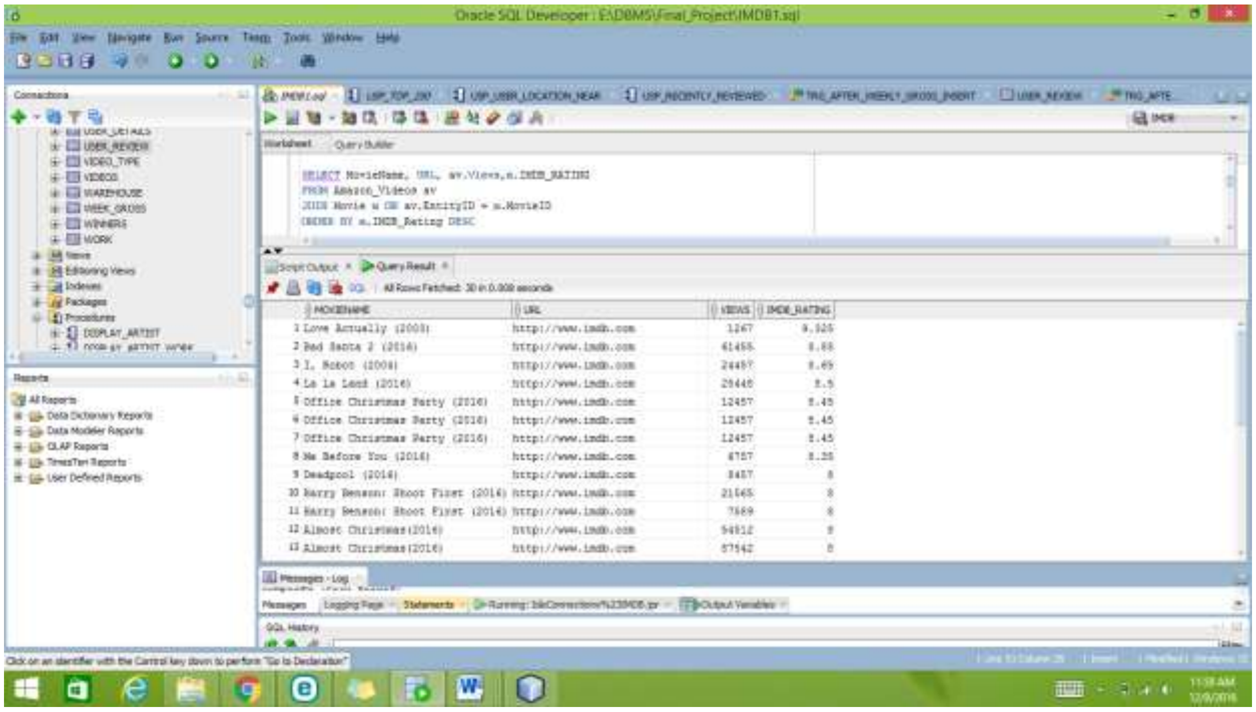
Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan

SELECT MovieName, URL, av.Views,m.IMDB_RATING

FROM Amazon_Videos av

JOIN Movie m ON av.EntityID = m.MovieID

ORDER BY m.IMDB_Rating DESC



```
SELECT MovieName, URL, av.Views,m.IMDB_RATING
FROM Amazon_Videos av
JOIN Movie m ON av.EntityID = m.MovieID
ORDER BY m.IMDB_Rating DESC
```

All Rows Fetched: 30 in 0.008 seconds

```
   MOVIENAME                         URL                    VIEWS  IMDB_RATING
 1 Love Actually (2003)              http://www.imdb.com     1267        9.325
 2 Bad Santa 2 (2016)                http://www.imdb.com    61455         8.88
 3 I, Robot (2008)                   http://www.imdb.com    24457         8.69
 4 La La Land (2016)                 http://www.imdb.com    25446          8.5
 5 Office Christmas Party (2016)     http://www.imdb.com    12457         8.45
 6 Office Christmas Party (2016)     http://www.imdb.com    12457         8.45
 7 Office Christmas Party (2016)     http://www.imdb.com    12457         8.45
 8 Me Before You (2016)              http://www.imdb.com     4757         8.25
 9 Deadpool (2016)                   http://www.imdb.com     8457            8
10 Barry Benson: Shoot First (2016)  http://www.imdb.com    21565            8
11 Barry Benson: Shoot First (2016)  http://www.imdb.com     7569            8
12 Almost Christmas(2016)            http://www.imdb.com    54512            8
13 Almost Christmas(2016)            http://www.imdb.com    57542            8
```

3. List all DVDs

SELECT

CASE EntityTypeID

WHEN 1 THEN m.MovieName

ELSE ts.ShowName

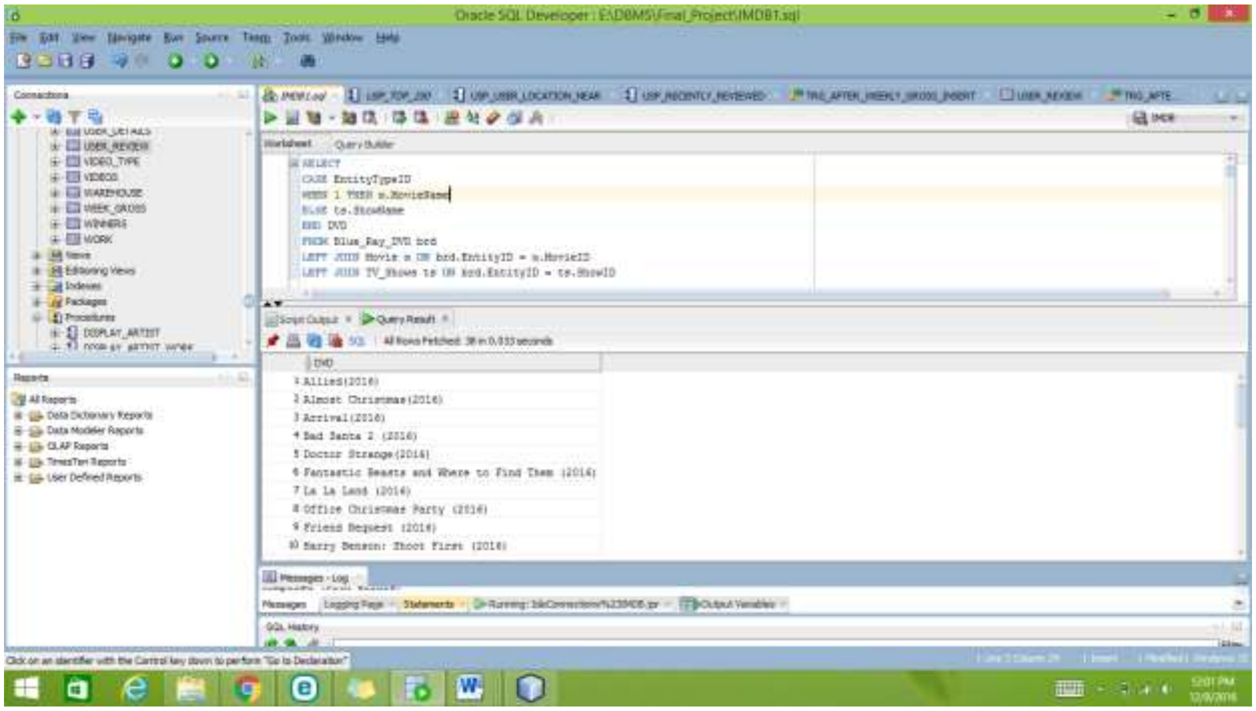
END DVD

FROM Blue_Ray_DVD brd

LEFT JOIN Movie m ON brd.EntityID = m.MovieID

LEFT JOIN TV_Shows ts ON brd.EntityID = ts.ShowID

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



```
SELECT
CASE EntityTypeID
WHEN 1 THEN m.MovieName
ELSE ts.ShowName
END DVD
FROM Blue_Ray_DVD brd
LEFT JOIN Movie m ON brd.EntityID = m.MovieID
LEFT JOIN TV_Shows ts ON brd.EntityID = ts.ShowID
```

All Rows Fetched: 38 in 0.033 seconds

```
   DVD
1  Allied(2016)
2  Almost Christmas(2016)
3  Arrival(2016)
4  Bad Santa 2 (2016)
5  Doctor Strange(2016)
6  Fantastic Beasts and Where to Find Them (2016)
7  La La Land (2016)
8  Office Christmas Party (2016)
9  Friend Request (2016)
10 Harry Benson: Shoot First (2016)
```

4. List all videos

SELECT

CASE EntityTypeID

WHEN 1 THEN m.MovieName

WHEN 2 THEN c.CelebrityName

ELSE ShowName

END Video, VideoURL, vt.VideoName

FROM Videos v

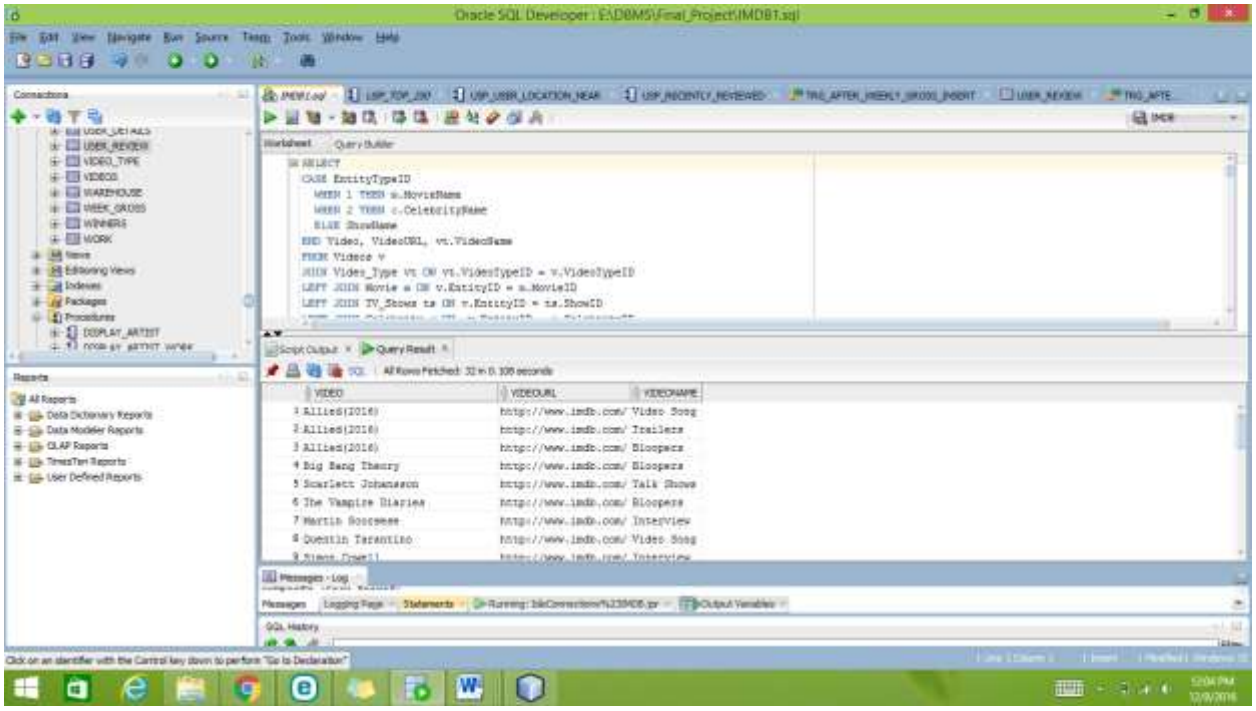
JOIN Video_Type vt ON vt.VideoTypeID = v.VideoTypeID

LEFT JOIN Movie m ON v.EntityID = m.MovieID

LEFT JOIN TV_Shows ts ON v.EntityID = ts.ShowID

LEFT JOIN Celebrity c ON v.EntityID = c.CelebrityID

JOIN Details.Video_Type vt ON vt.VideoTypeID = v.VideoTypeID



```
SELECT
  CASE EntityTypeID
    WHEN 1 THEN m.MovieName
    WHEN 2 THEN c.CelebrityName
    ELSE ShowName
  END Video, VideoURL, vt.VideoName
  FROM Videos v
  JOIN Video_Type vt ON vt.VideoTypeID = v.VideoTypeID
  LEFT JOIN Movie m ON v.EntityID = m.MovieID
  LEFT JOIN TV_Shows ts ON v.EntityID = ts.ShowID
```

All Rows Fetched: 32 in 0.108 seconds

```
  VIDEO                 VIDEOURL                VIDEONAME
1 Allied(2016)          http://www.imdb.com/    Video Song
2 Allied(2016)          http://www.imdb.com/    Trailers
3 Allied(2016)          http://www.imdb.com/    Bloopers
4 Big Bang Theory       http://www.imdb.com/    Bloopers
5 Scarlett Johansson    http://www.imdb.com/    Talk Shows
6 The Vampire Diaries   http://www.imdb.com/    Bloopers
7 Martin Scorsese       http://www.imdb.com/    Interview
8 Quentin Tarantino     http://www.imdb.com/    Video Song
9 Simon Cowell          http://www.imdb.com/    Interview
```

5.Latest movie trailers

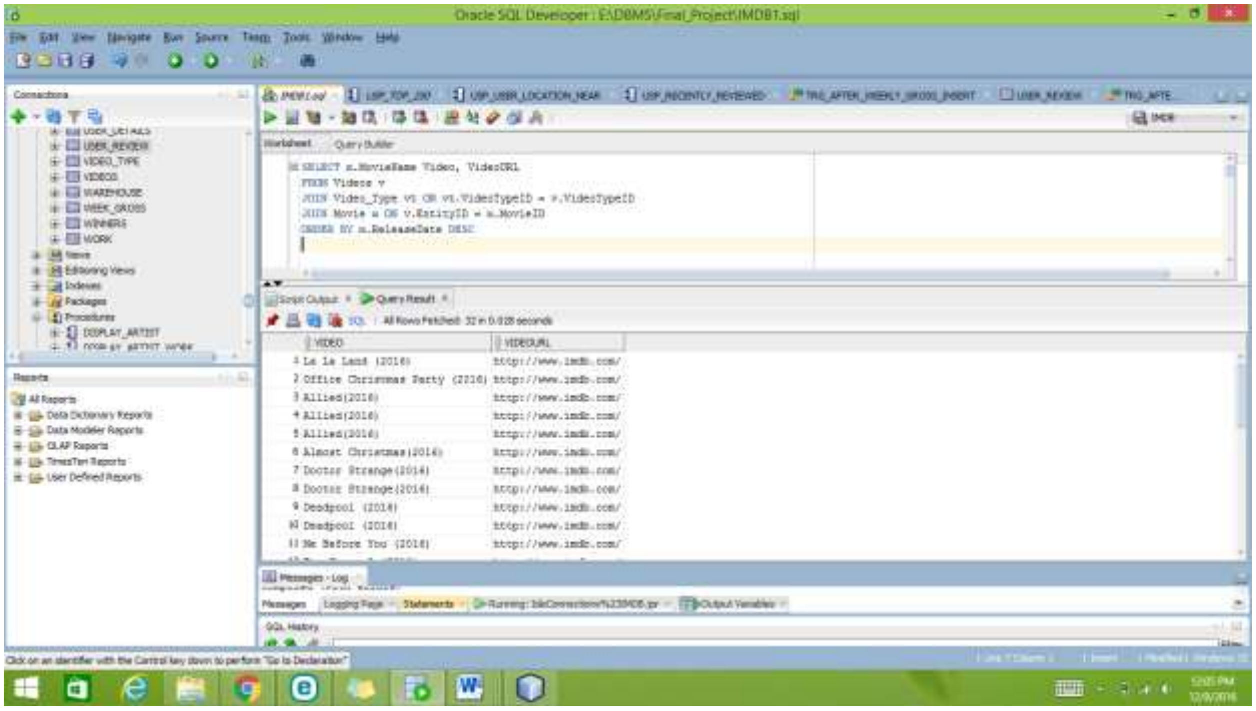
SELECT m.MovieName Video, VideoURL
FROM Videos v

JOIN Video_Type vt ON vt.VideoTypeID = v.VideoTypeID

JOIN Movie m ON v.EntityID = m.MovieID

ORDER BY m.ReleaseDate DESC

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



```
SELECT m.MovieName Video, VideoURL
FROM Videos v
JOIN Video_Type vt ON vt.VideoTypeID = v.VideoTypeID
JOIN Movie m ON v.EntityID = m.MovieID
ORDER BY m.ReleaseDate DESC
```

All Rows Fetched: 32 in 0.028 seconds

```
    VIDEO                         VIDEOURL
 1  La La Land (2016)             http://www.imdb.com/
 2  Office Christmas Party (2016) http://www.imdb.com/
 3  Allied(2016)                  http://www.imdb.com/
 4  Allied(2016)                  http://www.imdb.com/
 5  Allied(2016)                  http://www.imdb.com/
 6  Almost Christmas(2016)        http://www.imdb.com/
 7  Doctor Strange(2016)          http://www.imdb.com/
 8  Doctor Strange(2016)          http://www.imdb.com/
 9  Deadpool (2016)               http://www.imdb.com/
10  Deadpool (2016)               http://www.imdb.com/
11  Me Before You (2016)          http://www.imdb.com/
```

6.Details of all soundtracks

SELECT TrackName,

CASE EntityTypeID

WHEN 1 THEN m.MovieName

ELSE ShowName

END Entity, s.Playtime, c.CelebrityName MusicDirector, c2.CelebrityName Singer

FROM Soundtracks s

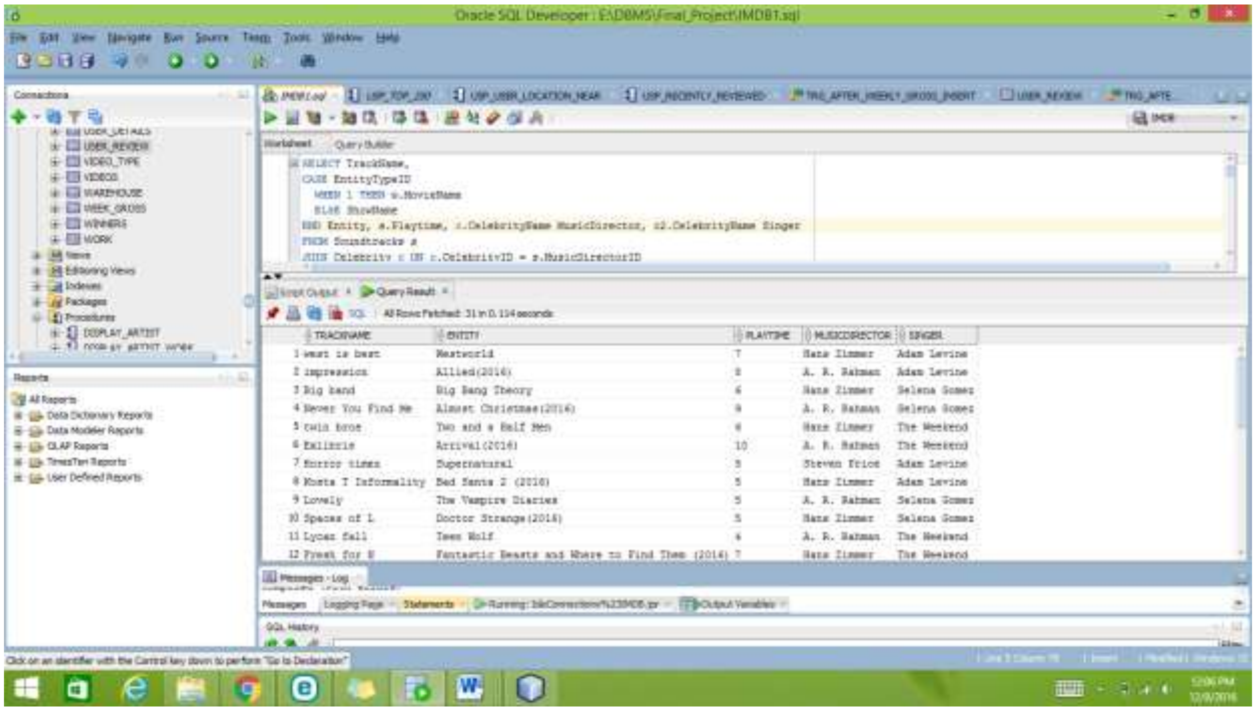
JOIN Celebrity c ON c.CelebrityID = s.MusicDirectorID

JOIN Celebrity c2 ON c2.CelebrityID = s.SingerID

LEFT JOIN Movie m ON s.EntityID = m.MovieID

LEFT JOIN TV_Shows ts ON s.EntityID = ts.ShowID

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



```
SELECT TrackName,
CASE EntityTypeID
  WHEN 1 THEN m.MovieName
  ELSE ShowName
END Entity, s.Playtime, c.CelebrityName MusicDirector, c2.CelebrityName Singer
FROM Soundtracks s
JOIN Celebrity c ON c.CelebrityID = s.MusicDirectorID
```

All Rows Fetched: 31 in 0.114 seconds

```
   TRACKNAME           ENTITY                                         PLAYTIME  MUSICDIRECTOR  SINGER
 1 want is best        Westworld                                      7         Hans Zimmer    Adam Levine
 2 impression          Allied(2016)                                   8         A. R. Rahman   Adam Levine
 3 Big band            Big Bang Theory                                6         Hans Zimmer    Selena Gomez
 4 Never You Find Me   Almost Christmas(2016)                         9         A. R. Rahman   Selena Gomez
 5 twin bros           Two and a Half Men                             6         Hans Zimmer    The Weekend
 6 Exlirrie            Arrival(2016)                                  10        A. R. Rahman   The Weekend
 7 Horror times        Supernatural                                   5         Steven Price   Adam Levine
 8 Kosta T Informality Bad Santa 2 (2016)                             5         Hans Zimmer    Adam Levine
 9 Lovely              The Vampire Diaries                            5         A. R. Rahman   Selena Gomez
10 Spaces of L         Doctor Strange(2016)                           5         Hans Zimmer    Selena Gomez
11 Lycan fall          Teen Wolf                                      4         A. R. Rahman   The Weekend
12 Frest for H         Fantastic Beasts and Where to Find Them (2016) 7         Hans Zimmer    The Weekend
```

7.TOP 250 MOVIES BASED ON RANKING OR IMDB REVIEW:

Select

Dense_RANK() OVER (ORDER BY Cast(avg(Rating) AS float) desc) AS Rank ,

MovieName,Round(Cast(avg(Rating) AS float),1) As IMDBRating

from Movie

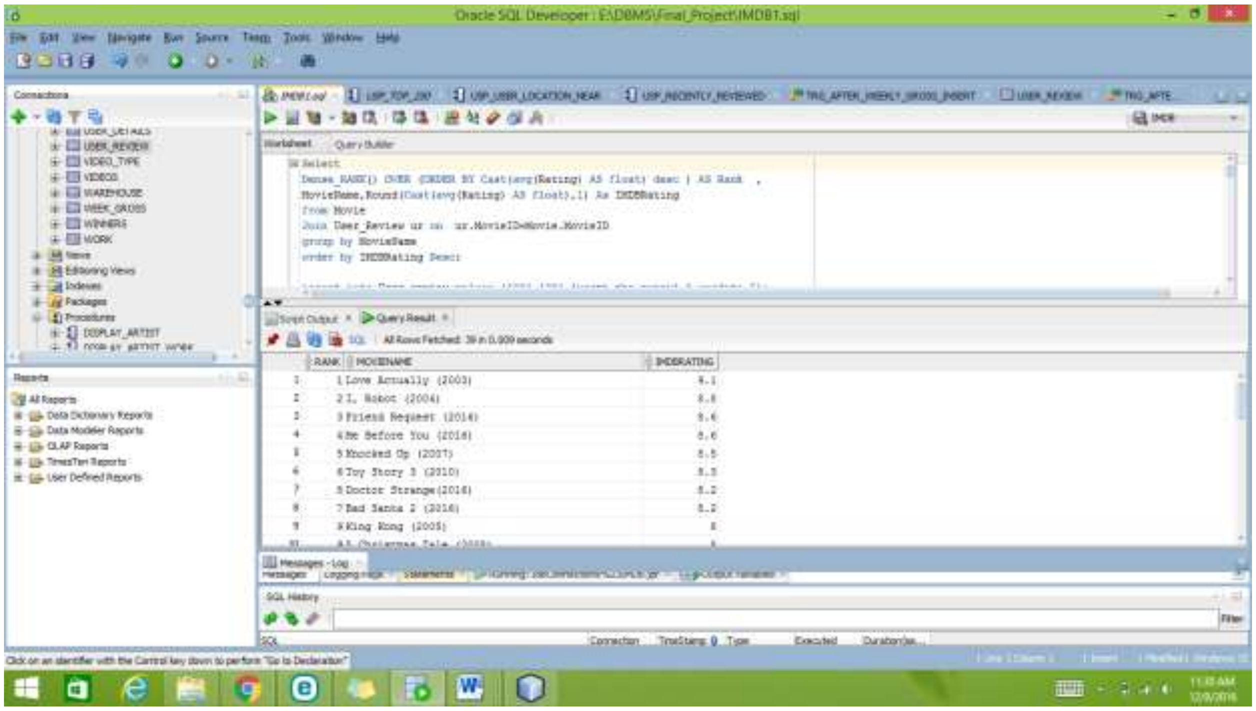
Join User_Review ur on ur.MovieID=Movie.MovieID

WHERE ROWNUM <=250

group by MovieName

order by IMDBRating Desc ;

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



```
Select
Dense_RANK() OVER (ORDER BY Cast(avg(Rating) AS float) desc ) AS Rank  ,
MovieName,Round(Cast(avg(Rating) AS float),1) As IMDBRating
from Movie
Join User_Review ur on  ur.MovieID=Movie.MovieID
group by MovieName
order by IMDBRating Desc;
```

All Rows Fetched: 39 in 0.909 seconds

	RANK	MOVIENAME	IMDBRATING
1	1	Love Actually (2003)	9.1
2	2	I, Robot (2004)	8.8
3	3	Friend Request (2016)	8.6
4	4	Me Before You (2016)	8.6
5	5	Knocked Up (2007)	8.5
6	6	Toy Story 3 (2010)	8.3
7	7	Doctor Strange (2016)	8.2
8	7	Bad Santa 2 (2016)	8.2
9	8	King Kong (2005)	8

8.TOP 250 MOVIES BASED ON NUMBER OF USER RATING:

Select

Dense_RANK() OVER (ORDER BY Cast(avg(Rating) AS float)desc) AS Rank ,

MovieName,Round(Cast(avg(Rating) AS float),1) As IMDBRating,count(*) TotalUserReview from Movie

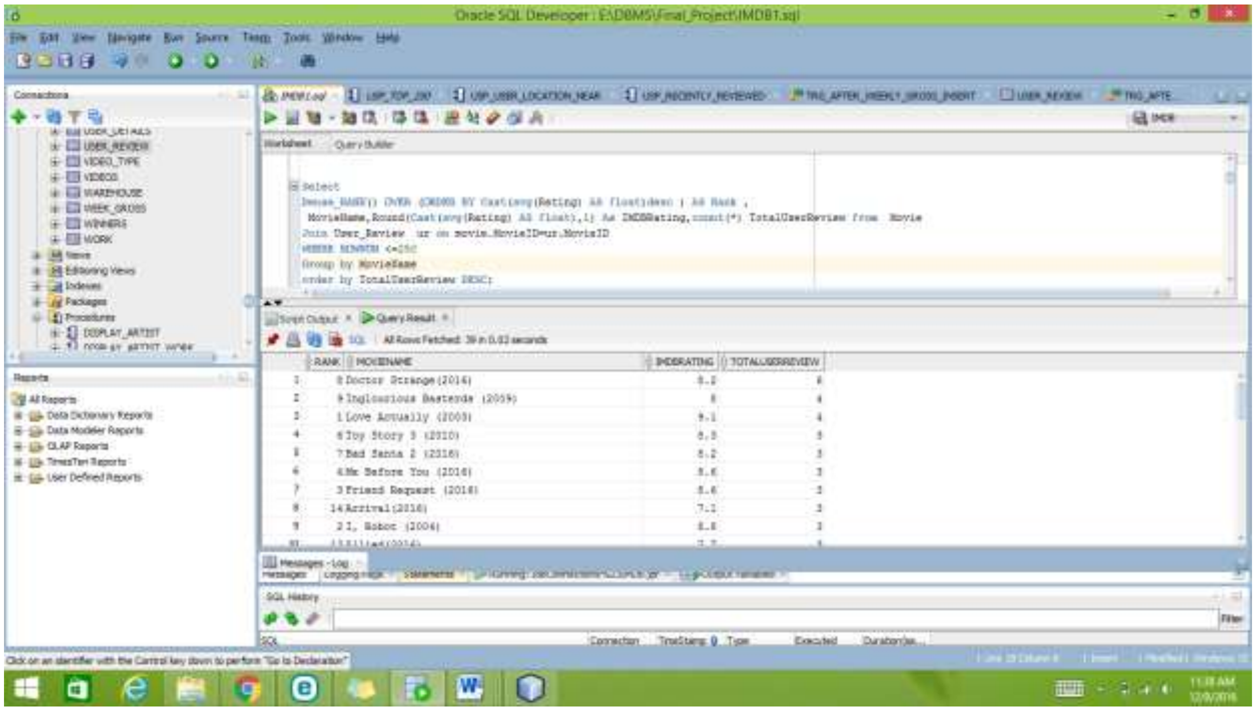
Join User_Review ur on movie.MovieID=ur.MovieID

WHERE ROWNUM <=250

Group by MovieName

order by TotalUserReview DESC;

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



```
Select
Dense_RANK() OVER (ORDER BY Cast(avg(Rating) AS float)desc ) AS Rank ,
 MovieName,Round(Cast(avg(Rating) AS float),1) As IMDBRating,count(*) TotalUserReview from  Movie
Join User_Review  ur on movie.MovieID=ur.MovieID
WHERE ROWNUM <=250
Group by MovieName
order by TotalUserReview DESC;
```

```
All Rows Fetched: 39 in 0.03 seconds

   RANK  MOVIENAME                          IMDBRATING  TOTALUSERREVIEW
1     8 Doctor Strange(2016)                       8.2                6
2     9 Inglourious Basterds (2009)                  8                4
3     1 Love Actually (2003)                       9.1                4
4     6 Toy Story 3 (2010)                         8.3                3
5     7 Bad Santa 2 (2016)                         8.2                3
6     4 Me Before You (2016)                       8.6                3
7     3 Friend Request (2016)                      8.6                3
8    14 Arrival(2016)                              7.1                3
9     2 I, Robot (2004)                            8.8                3
```

9.TOP 250 MOVIES BASED ON Release Date:

Select

Dense_RANK() OVER (ORDER BY Cast(avg(Rating) AS float)desc) AS Rank ,

MovieName,ReleaseDate,Round(Cast(avg(Rating) AS float),1) As IMDBRating

from Movie movie

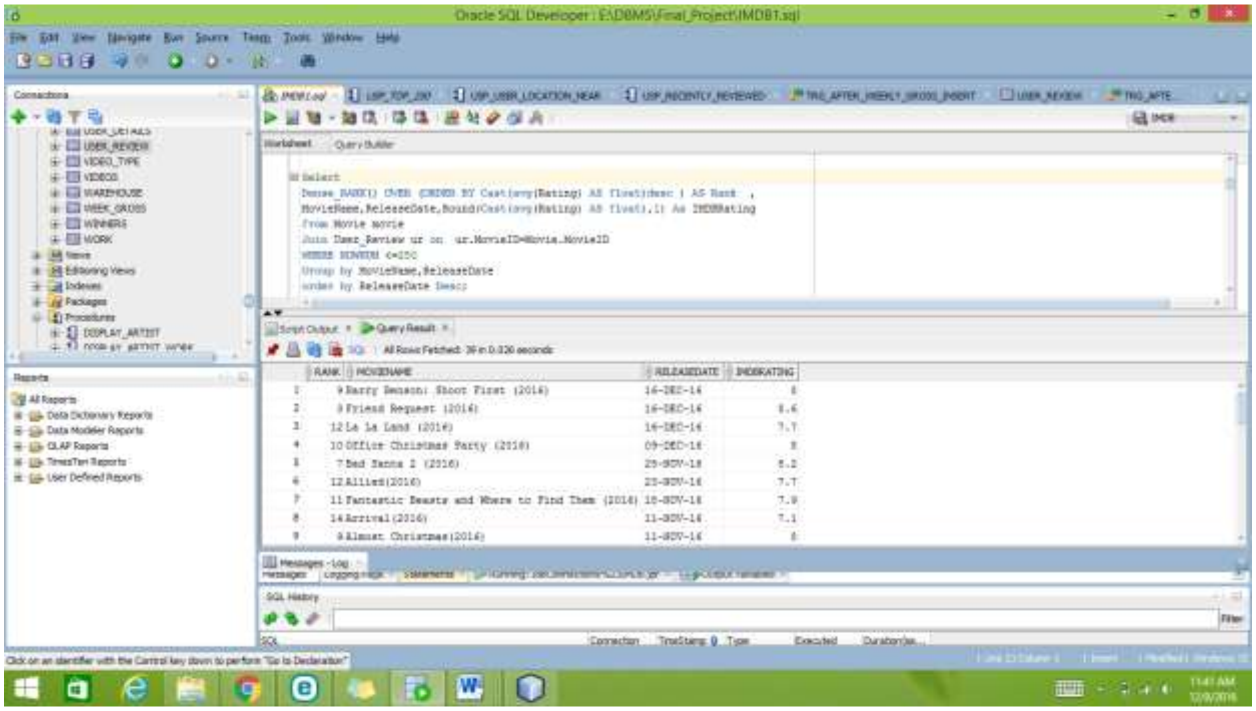
Join User_Review ur on ur.MovieID=Movie.MovieID

WHERE ROWNUM <=250

Group by MovieName,ReleaseDate

order by ReleaseDate Desc;

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



```
Select
Dense_RANK() OVER (ORDER BY Cast(avg(Rating) AS float)desc ) AS Rank  ,
MovieName,ReleaseDate,Round(Cast(avg(Rating) AS float),1) As IMDBRating
from Movie movie
Join User_Review ur on  ur.MovieID=Movie.MovieID
WHERE ROWNUM <=250
Group by MovieName,ReleaseDate
order by ReleaseDate Desc;
```

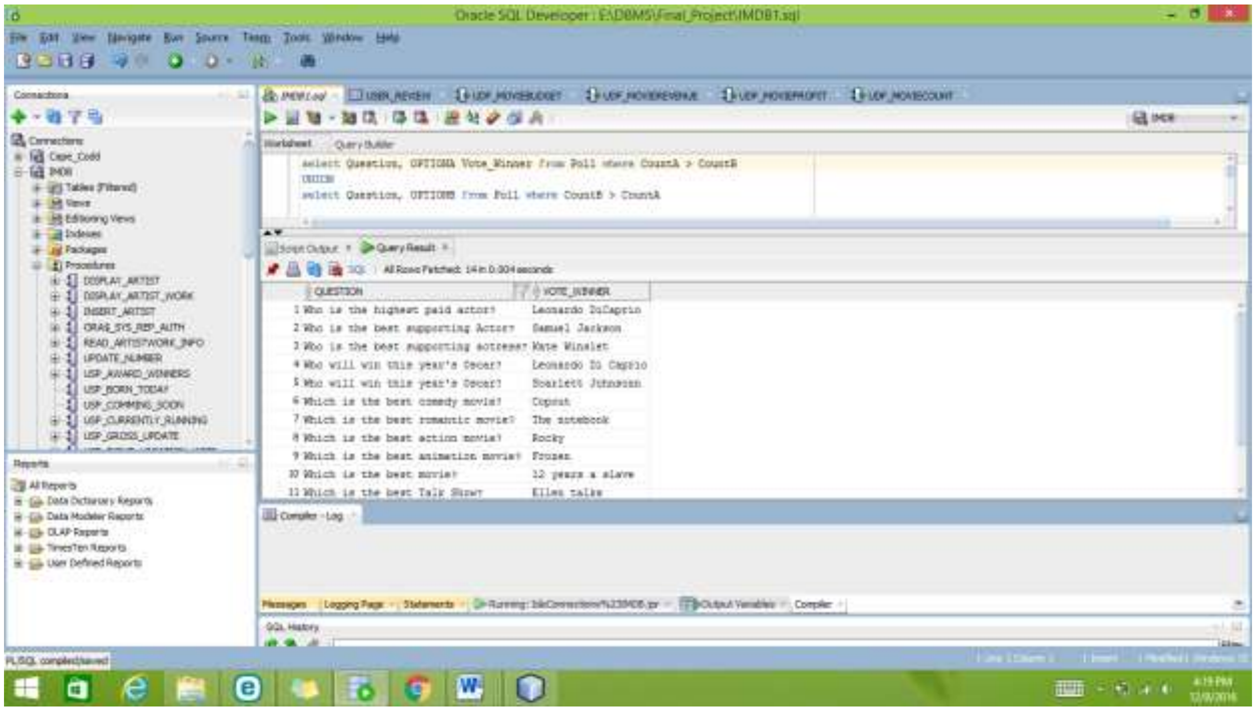
All Rows Fetched: 39 in 0.320 seconds

```
   RANK MOVIENAME                                     RELEASEDATE  IMDBRATING
1     9 Barry Benson: Shoot First (2016)              16-DEC-16             8
2     3 Friend Request (2016)                         16-DEC-16           8.6
3    12 La La Land (2016)                             16-DEC-16           7.7
4    10 Office Christmas Party (2016)                 09-DEC-16             8
5     7 Bad Santa 2 (2016)                            25-NOV-16           8.2
6    12 Allied(2016)                                  23-NOV-16           7.7
7    11 Fantastic Beasts and Where to Find Them (2016) 18-NOV-16          7.9
8    14 Arrival(2016)                                 11-NOV-16           7.1
9     9 Almost Christmas(2016)                        11-NOV-16             8
```

10.POLL WINNER:

select Question, OPTIONA Vote_Winner from Poll where CountA > CountB
UNION
select Question, OPTIONB from Poll where CountB > CountA

Deivakumaran Dhanasegaran
Aswathnarayan
Pranesh Ramakrishnan



```
select Question, OPTIONA Vote_Winner from Poll where CountA > CountB
UNION
select Question, OPTIONB from Poll where CountB > CountA
```

All Rows Fetched: 14 in 0.004 seconds

```
   QUESTION                              VOTE_WINNER
 1 Who is the highest paid actor?        Leonardo DiCaprio
 2 Who is the best supporting Actor?     Samuel Jackson
 3 Who is the best supporting actress?   Kate Winslet
 4 Who will win this year's Oscar?       Leonardo Di Caprio
 5 Who will win this year's Oscar?       Scarlett Johnson
 6 Which is the best comedy movie?       Copout
 7 Which is the best romantic movie?     The notebook
 8 Which is the best action movie?       Rocky
 9 Which is the best animation movie?    Frozen
10 Which is the best movie?              12 years a slave
11 Which is the best Talk Show?          Ellen talks
```