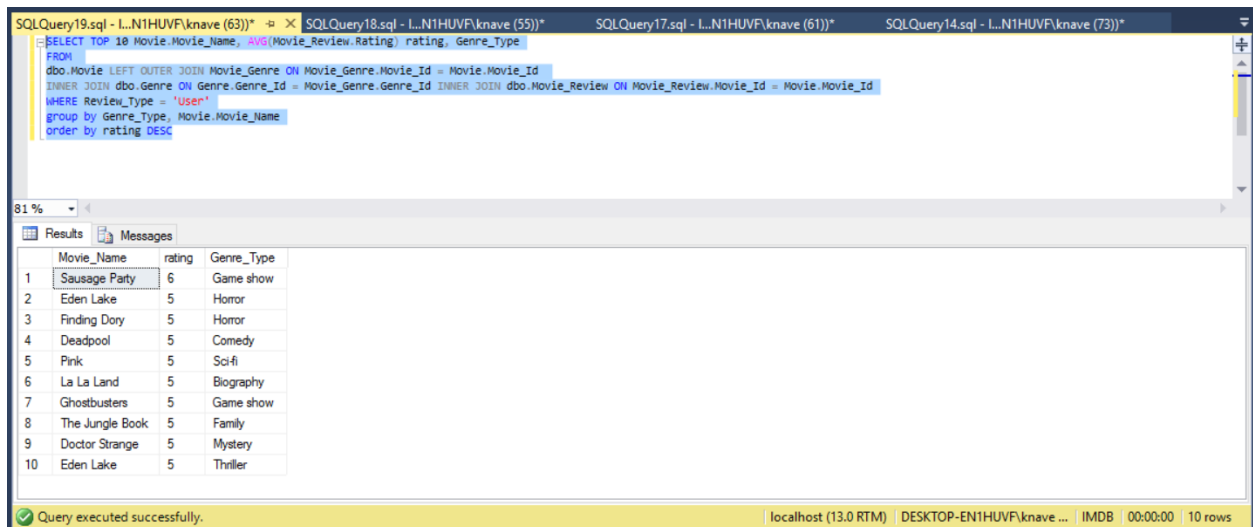


Query 1

To find top 10 movies with respect to User rating and Genre Type of movies.

##

```
SELECT TOP 10 Movie.Movie_Name, AVG(Movie_Review.Rating) rating, Genre_Type
FROM
dbo.Movie LEFT OUTER JOIN Movie_Genre ON Movie_Genre.Movie_Id = Movie.Movie_Id
INNER JOIN dbo.Genre ON Genre.Genre_Id = Movie_Genre.Genre_Id INNER JOIN dbo.Movie_Review
ON Movie_Review.Movie_Id = Movie.Movie_Id
WHERE Review_Type = 'User'
group by Genre_Type, Movie.Movie_Name
order by rating desc
```



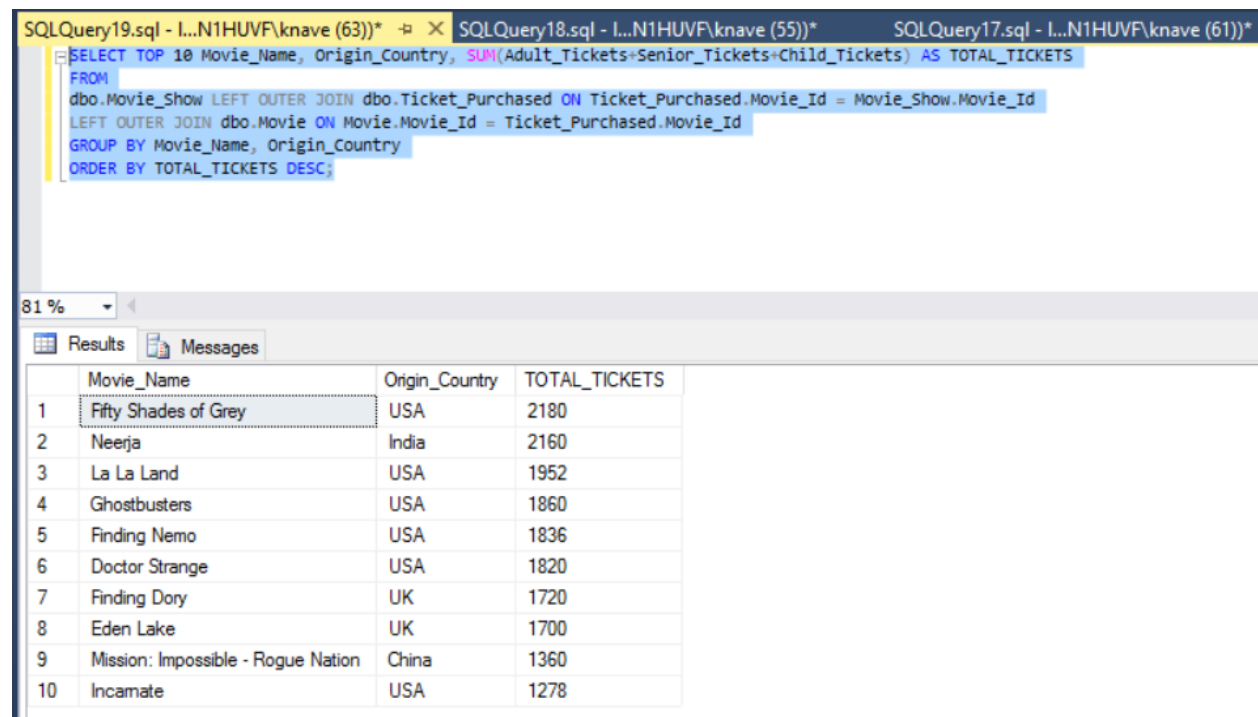
	Movie_Name	rating	Genre_Type
1	Sausage Party	6	Game show
2	Eden Lake	5	Horror
3	Finding Dory	5	Horror
4	Deadpool	5	Comedy
5	Pink	5	Sci-fi
6	La La Land	5	Biography
7	Ghostbusters	5	Game show
8	The Jungle Book	5	Family
9	Doctor Strange	5	Mystery
10	Eden Lake	5	Thriller

Query executed successfully. | localhost (13.0 RTM) | DESKTOP-EN1HUVF\knaive ... | IMDB | 00:00:00 | 10 rows

Query 2

Top 10 movies watched by User's

```
SELECT TOP 10 Movie_Name, Origin_Country, SUM(Adult_Tickets+Senior_Tickets+Child_Tickets) AS  
TOTAL_TICKETS  
FROM  
dbo.Movie_Show LEFT OUTER JOIN dbo.Ticket_Purchased ON Ticket_Purchased.Movie_Id =  
Movie_Show.Movie_Id  
LEFT OUTER JOIN dbo.Movie ON Movie.Movie_Id = Ticket_Purchased.Movie_Id  
GROUP BY Movie_Name, Origin_Country  
ORDER BY TOTAL_TICKETS DESC;
```



The screenshot shows a SQL Server Enterprise Manager window with three tabs: 'SQLQuery19.sql - I...N1HUVF\knave (63))*', 'SQLQuery18.sql - I...N1HUVF\knave (55))*', and 'SQLQuery17.sql - I...N1HUVF\knave (61))*'. The 'SQLQuery19.sql' tab is active, displaying the following SQL query:

```
SELECT TOP 10 Movie_Name, Origin_Country, SUM(Adult_Tickets+Senior_Tickets+Child_Tickets) AS TOTAL_TICKETS  
FROM  
dbo.Movie_Show LEFT OUTER JOIN dbo.Ticket_Purchased ON Ticket_Purchased.Movie_Id = Movie_Show.Movie_Id  
LEFT OUTER JOIN dbo.Movie ON Movie.Movie_Id = Ticket_Purchased.Movie_Id  
GROUP BY Movie_Name, Origin_Country  
ORDER BY TOTAL_TICKETS DESC;
```

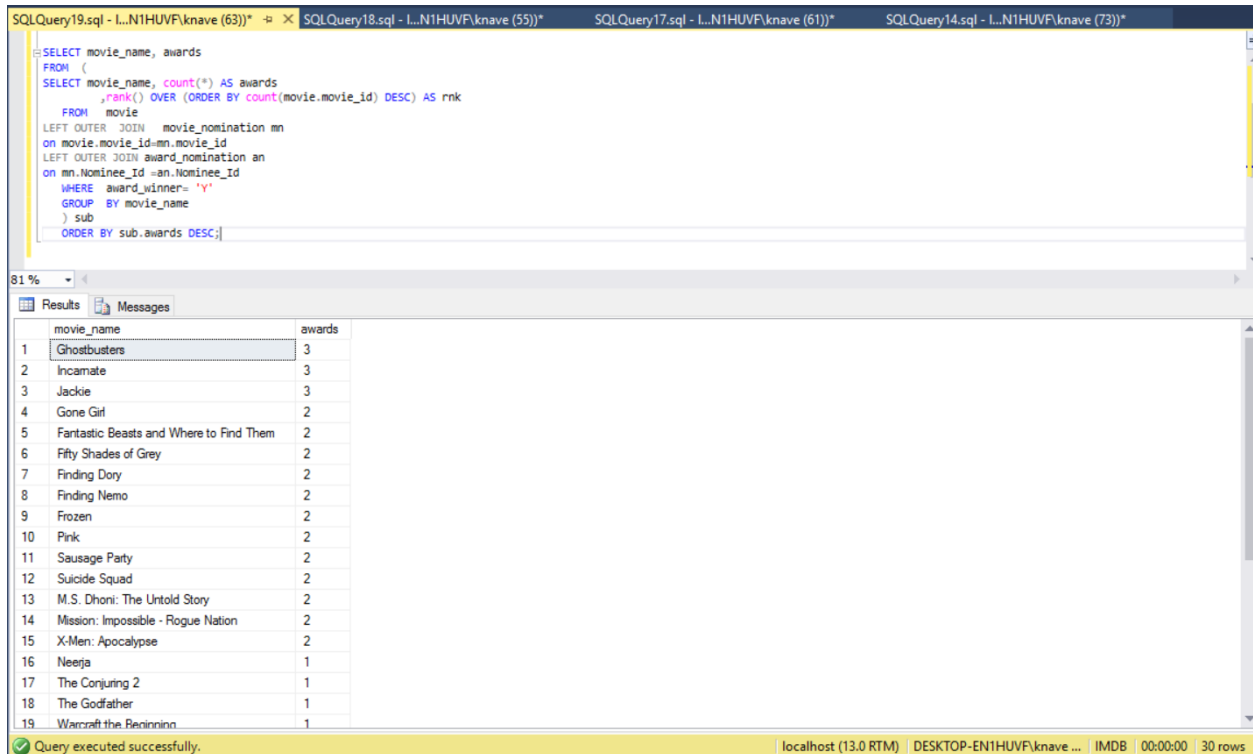
Below the query editor, the 'Results' tab is selected, showing a table with 10 rows and 3 columns: 'Movie_Name', 'Origin_Country', and 'TOTAL_TICKETS'. The table is sorted by 'TOTAL_TICKETS' in descending order. The zoom level is set to 81%.

	Movie_Name	Origin_Country	TOTAL_TICKETS
1	Fifty Shades of Grey	USA	2180
2	Neerja	India	2160
3	La La Land	USA	1952
4	Ghostbusters	USA	1860
5	Finding Nemo	USA	1836
6	Doctor Strange	USA	1820
7	Finding Dory	UK	1720
8	Eden Lake	UK	1700
9	Mission: Impossible - Rogue Nation	China	1360
10	Incarnate	USA	1278

Query 3

To find most awarded Movies

```
SELECT movie_name, awards
FROM (
SELECT movie_name, count(*) AS awards
      ,rank() OVER (ORDER BY count(movie.movie_id) DESC) AS rnk
  FROM movie
LEFT OUTER JOIN movie_nomination mn
on movie.movie_id=mn.movie_id
LEFT OUTER JOIN award_nomination an
on mn.Nominee_Id =an.Nominee_Id
  WHERE award_winner= 'Y'
  GROUP BY movie_name
) sub
ORDER BY sub.awards DESC;
```



The screenshot shows a SQL Server Enterprise Manager interface. At the top, there are several tabs for different queries. The active query is 'SQLQuery19.sql - L...N1HUVF\knave (63))'. The query text is displayed in the main editor. Below the editor, the 'Results' pane shows the output of the query as a table with two columns: 'movie_name' and 'awards'. The results are ordered by the number of awards in descending order. The first row is 'Ghostbusters' with 3 awards, followed by 'Incarnate', 'Jackie', and 'Gone Girl', each with 2 awards. The table continues with more movies and their respective award counts. At the bottom of the interface, a status bar indicates that the query was executed successfully and shows the number of rows returned (30 rows).

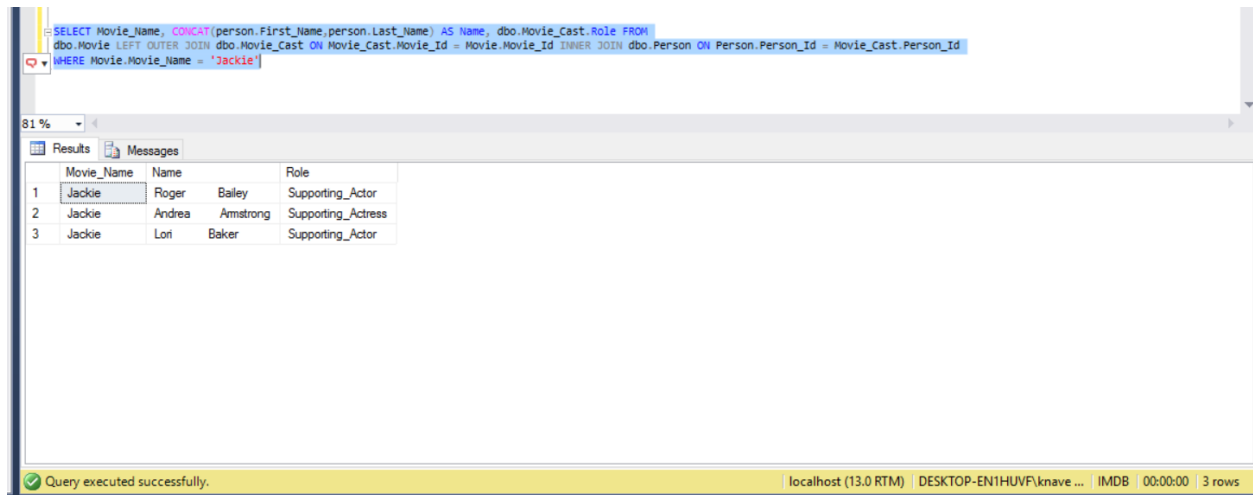
movie_name	awards
1 Ghostbusters	3
2 Incarnate	3
3 Jackie	3
4 Gone Girl	2
5 Fantastic Beasts and Where to Find Them	2
6 Fifty Shades of Grey	2
7 Finding Dory	2
8 Finding Nemo	2
9 Frozen	2
10 Pink	2
11 Sausage Party	2
12 Suicide Squad	2
13 M.S. Dhoni: The Untold Story	2
14 Mission: Impossible - Rogue Nation	2
15 X-Men: Apocalypse	2
16 Neerja	1
17 The Conjuring 2	1
18 The Godfather	1
19 Warcraft the Rebirth	1

Query executed successfully. localhost (13.0 RTM) | DESKTOP-EN1HUVF\knave ... | IMDB | 00:00:00 | 30 rows

Query 4

This query fetches the Cast of a movie.

```
SELECT Movie_Name, CONCAT(person.First_Name, person.Last_Name) AS Name,  
dbo.Movie_Cast.Role FROM  
dbo.Movie LEFT OUTER JOIN dbo.Movie_Cast ON Movie_Cast.Movie_Id = Movie.Movie_Id INNER  
JOIN dbo.Person ON Person.Person_Id = Movie_Cast.Person_Id  
WHERE Movie.Movie_Name = 'Jackie'
```



The screenshot shows a SQL query execution window. The query is displayed in the top pane, and the results are shown in the bottom pane. The results table has three columns: Movie_Name, Name, and Role. There are three rows of data.

	Movie_Name	Name	Role
1	Jackie	Roger Bailey	Supporting_Actor
2	Jackie	Andrea Armstrong	Supporting_Actress
3	Jackie	Lori Baker	Supporting_Actor

Query executed successfully. localhost (13.0 RTM) DESKTOP-ENTHUVF\knave ... IMDB 00:00:00 3 rows

Triggers

Whenever a User buys tickets, the total number of tickets available will be reduced by deleting the number of tickets bought by the User from the Available tickets column.

```
CREATE TRIGGER trig_Update_Tickets
ON DBO.Ticket_Purchased
AFTER INSERT
AS
Begin
Declare @sentckt int
Declare @adtckt int
Declare @chldtckt int
Declare @scId int
Declare @theatreId int
Declare @MovId int
Declare @shwId int
Declare @tcktSold int
DECLARE @tckt_avail INT

Select @sentckt = Senior_Tickets, @adtckt = Adult_Tickets, @chldtckt = Child_Tickets,
@scId = Screen_Id, @theatreId = Theatre_Id,
@MovId = Movie_Id, @shwId = Show_Id
FROM inserted;

SET @tcktSold = @sentckt + @adtckt + @chldtckt;

Update Movie_Show SET Tickets_Available = Tickets_Available - @tcktSold where Theatre_Id
= @theatreId and Screen_Id = @scId and Movie_Id = @MovId and Show_Id = @shwId
END
```

Procedure #1

This procedure takes Actress_Name as the Input parameter and displays all the Movies she has acted as an Actress.

```
USE [IMDB]
GO
/***** Object:  StoredProcedure [dbo].[PrintMovieData]    Script Date: 12/8/2016 3:19:32
PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[PrintMovieData]
@fname char(25)
AS
BEGIN
SET NOCOUNT ON;
DECLARE @first_name char(25)
DECLARE @lName char(25)
DECLARE @movie_id INT
DECLARE @movieName CHAR(30)
DECLARE @desc varchar(100);
DECLARE moviecursor CURSOR FOR
select first_name, last_name, Movie_Name, Description from person join movie_cast on
person.person_id=movie_cast.person_id
inner join movie on movie_cast.movie_id=movie.movie_id where movie_cast.role=
'Actress'AND person.First_Name = @fname;

if @fname NOT IN(Select First_Name from Person)
BEGIN
RAISERROR('Invalid Entry',16,1);
return;
END
else

OPEN moviecursor;
FETCH NEXT FROM moviecursor;
WHILE @@FETCH_STATUS=0
BEGIN
FETCH NEXT FROM moviecursor into @first_name,@lName,@movieName,@desc;
PRINT '*****Details*****'
PRINT '*****Movie_Id*****'
PRINT @movie_id;
PRINT '*****Name*****'
PRINT CONCAT(@first_name,@lName);
PRINT '*****Movie Name*****'
PRINT @movieName;
PRINT '*****Description*****'
PRINT @desc;

END
CLOSE moviecursor;
DEALLOCATE moviecursor;
```

Procedure #2

This procedure takes City Name as an input parameter and displays all the Theatres in that particular City

```
USE [IMDB]
GO
/***** Object:  StoredProcedure [dbo].[Theatre_Location]    Script Date: 12/8/2016
3:21:01 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER procedure [dbo].[Theatre_Location]
(@city char(25))
AS
BEGIN
SET NOCOUNT ON;
DECLARE @theatre_id int;
DECLARE @theatre_name char(30);
DECLARE theatrecursor CURSOR FOR select theatre_id,name from Theatre inner join Location
on theatre.location_id=location.location_id WHERE City = @city

if @city NOT IN(Select city from Location)
BEGIN
RAISERROR('Invalid Entry',16,1);
return;
END
else

OPEN theatrecursor;
FETCH NEXT FROM theatrecursor;
WHILE @@FETCH_STATUS=0
BEGIN
FETCH NEXT FROM theatrecursor into @theatre_id,@theatre_name;
PRINT '*****Details*****'
PRINT '*****Theatre_Id*****'
PRINT @theatre_id;
PRINT '*****Name*****'
PRINT @theatre_name;
PRINT '*****City*****'
PRINT @city;

END
CLOSE theatrecursor;
DEALLOCATE theatrecursor;
END;
```