Individual Project on Movies Database

Introduction:

Most of the people loves to watch movies, therefore, I have decided to analyze **Movies** database and answer queries that comes to our mind for example, in which year Joker movie was released and who was the lead actor in the movie? It will also give us the ratings (in numbers) given by a reviewer for a particular movie.

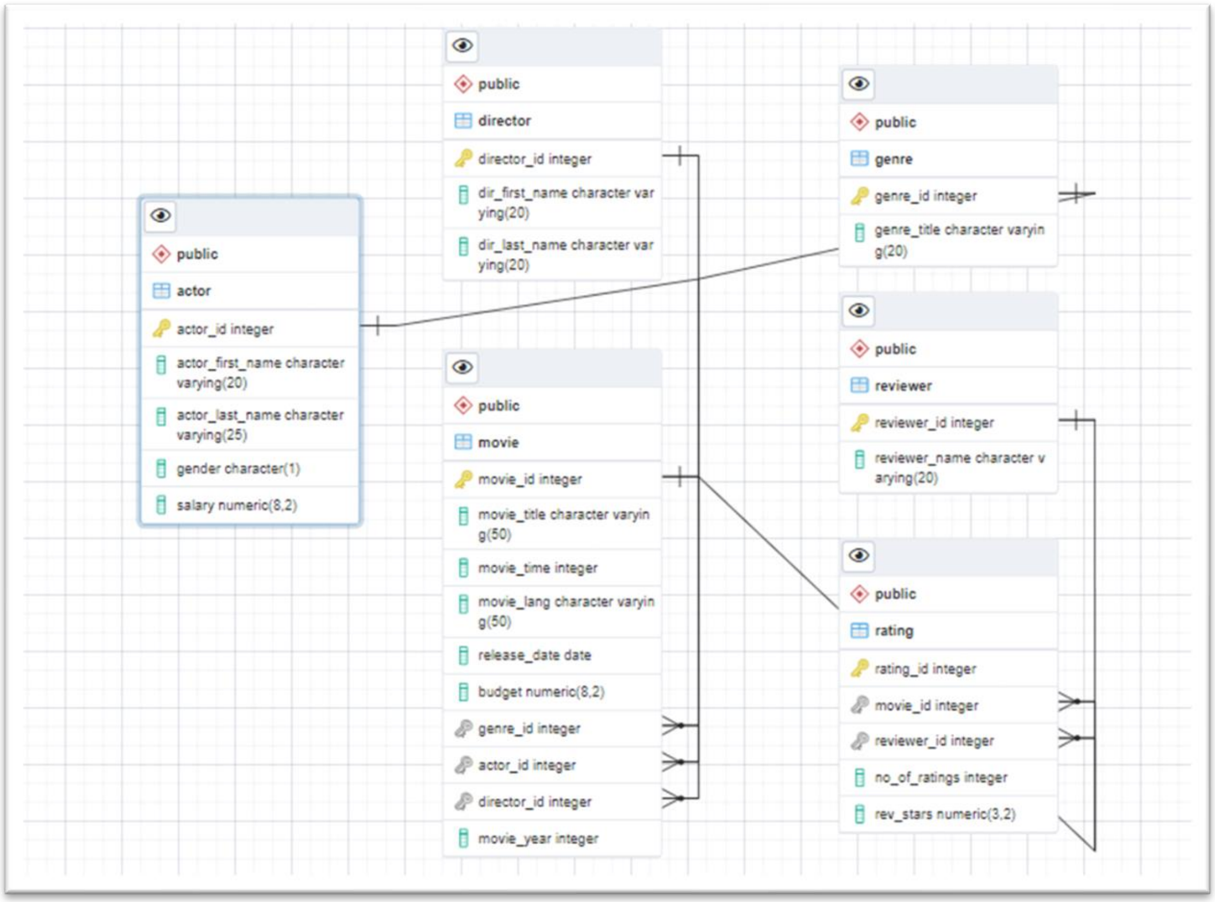
Goal of the project:

As already mentioned, I want to analyze this data to answer various queries related to movies and ratings given by various reviewers. I will aim to answer the following questions:

1) Find when the movie 'American Beauty' released. Return movie release year.
2) Find all reviewers who have rated 7 or more stars to their rating.
3) Find all movies whose movie titles include the words 'Boogie Nights'. Sort the result-set in ascending order by movie year.
4) Write a SQL query to find the movies who got ratings below 7. Return reviewer name, movie title, and number of stars for those movies.
5) Write a SQL query to find those reviewers who rated more than two movies. Group the result set on reviewer's name, movie title.

6) Write a SQL query to compute the average time and count number of movies for each genre. Return genre title, average time and number of movies for each genre.
7) Write a SQL query to find the movies released before 1st January 1989. Sort the result-set in descending order by date of release. Return movie title, release year, date of release, duration, and first and last name of the director.
8) Write a SQL query to find those movies, which have received highest number of stars and name starts with "Am". Group the result set on movie title and sorts the result-set in ascending order by movie title. Return movie title and maximum number of review stars.
9) Create a **VIEW** to fetch all the actors who acted in movies whose budget was less than 6000000. Fetch columns movie_title, actor_id, actor_first_name, actor_last_name and budget.
10) Find the reviewer's name, movie title, and stars within range 6-8 in an order that movie title will come first, then by number of stars, and lastly by reviewer name. (Use **CTE**)

Relational Schema:



MOVIES SCHEMA

Dataset:

This database will have 6 tables in total. Below are the different tables with brief description and I analyzed this database using ***PostgreSQL***.

1) ***Actor:***

   a. actor_id – this is a unique ID for each actor and will be the primary key for this table
   b. actor_first_name – this is the first name of each actor
   c. actor_last_name – this is the last name of each actor
   d. gender – this is the gender of each actor
   e. salary-this is the salary of each actor

2) ***Genre***:

   a. genre_id – this is a unique ID for each genre and will be the primary key for this table
   b. genre_title – this is the description of the genre

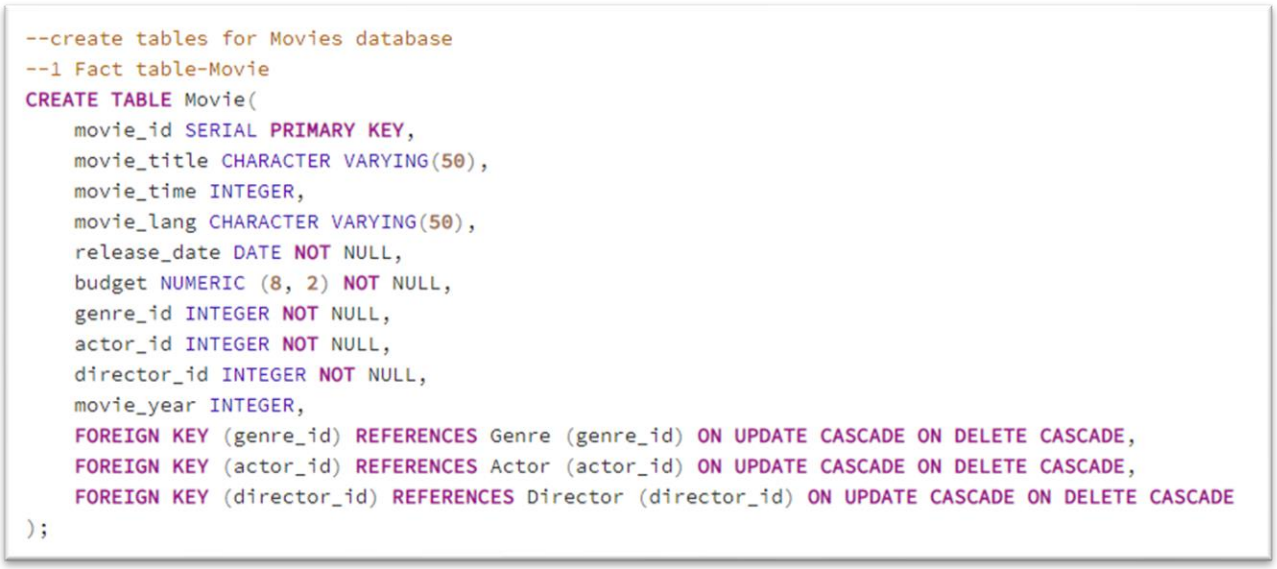
3) ***Director***:
   a. director_id-this is a unique ID for each director and will be the primary key for this table
   b. director_first_name- this is the first name of the director
   c. director_last_name- this is the last name of the director

4) Movie:
   a. movie_id – this is the unique ID for each movie and will be the primary key for this table
   b. movie_title – this column represents the name of the movie
   c. movie_time– this is the year of making the movie
   d. movie_lang– duration of the movie i.e., how long it was running
   e. release_date– the language in which movie was casted
   f. budget– this is the release date of the movie
   g. genre_id-this is the ID of the genre, which is referencing the genre_id column of the table Genre and will be the foreign key in this table
   h. actor_id- this is the ID of the actor, which is referencing the actor_id column of the table Actor and will be the foreign key in this table
   i. director_id- this is the ID of the director, which is referencing the director_id column of the table Director and will be the foreign key in this table
   j. movie_year – this is the year of making the movie

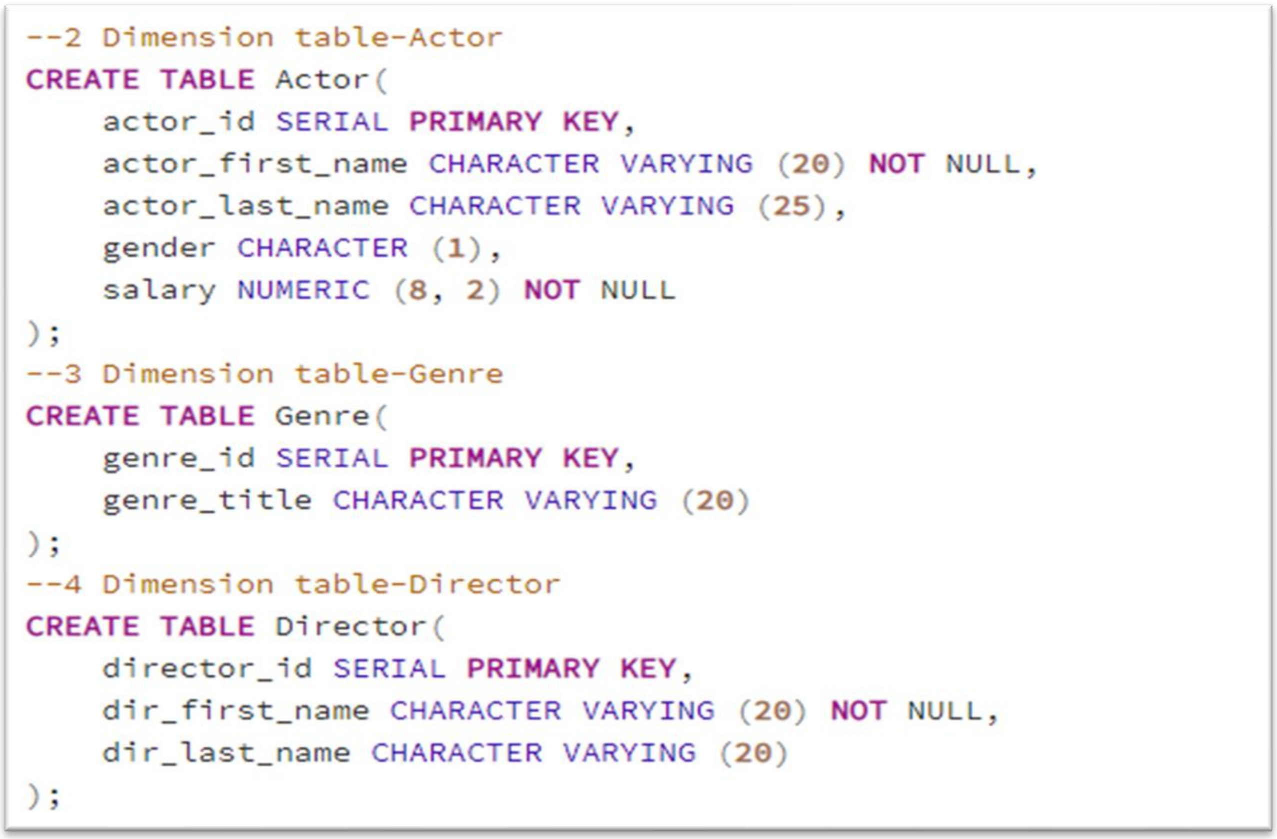
5) Reviewer:
   a. reviewer_id – this is the unique ID for each reviewer and will be the primary key for this table
   b. reviewer_name – this is the name of the reviewer

6) Rating:
   a. rating_id – this is the unique ID for each rating and will be the primary key for this table
   b. movie_id –this is the ID of the movie, which is referencing the movie_id column of the table Movie and will be the foreign key in this table
   c. reviewer_id – this is the ID of the reviewer, which is referencing the reviewer_id column of the table Reviewer and will be the foreign key in this table
   d. rev_stars – this indicates how many stars a reviewer rated for a review of a movie
   e. no_of_ratings – this indicates how many ratings a movie achieved till date

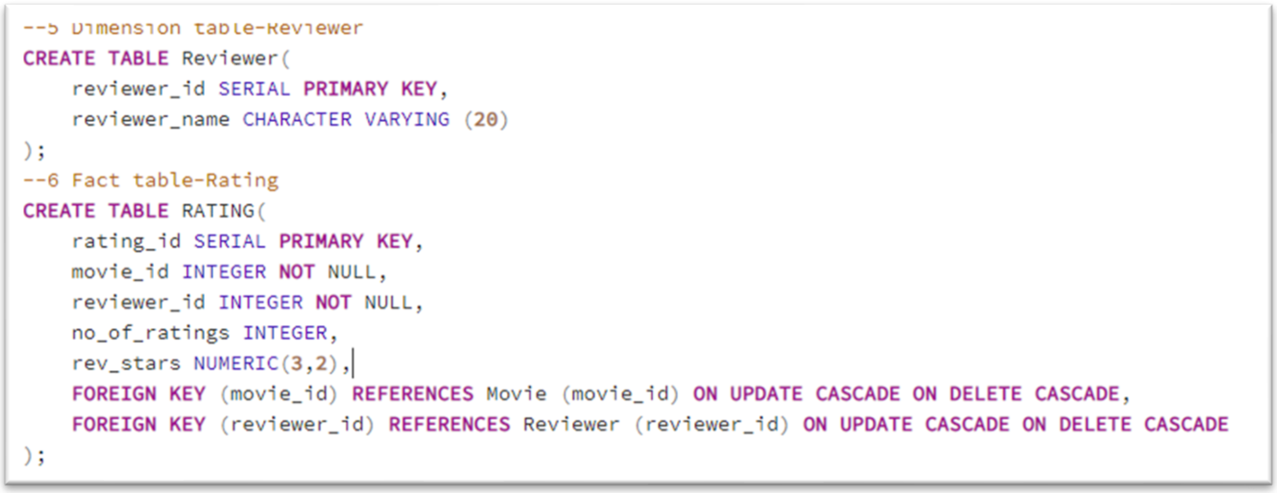
CREATE QUERIES:



```sql
--create tables for Movies database
--1 Fact table-Movie
CREATE TABLE Movie(
    movie_id SERIAL PRIMARY KEY,
    movie_title CHARACTER VARYING(50),
    movie_time INTEGER,
    movie_lang CHARACTER VARYING(50),
    release_date DATE NOT NULL,
    budget NUMERIC (8, 2) NOT NULL,
    genre_id INTEGER NOT NULL,
    actor_id INTEGER NOT NULL,
    director_id INTEGER NOT NULL,
    movie_year INTEGER,
    FOREIGN KEY (genre_id) REFERENCES Genre (genre_id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (actor_id) REFERENCES Actor (actor_id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (director_id) REFERENCES Director (director_id) ON UPDATE CASCADE ON DELETE CASCADE
);
```



```sql
--2 Dimension table-Actor
CREATE TABLE Actor(
    actor_id SERIAL PRIMARY KEY,
    actor_first_name CHARACTER VARYING (20) NOT NULL,
    actor_last_name CHARACTER VARYING (25),
    gender CHARACTER (1),
    salary NUMERIC (8, 2) NOT NULL
);
--3 Dimension table-Genre
CREATE TABLE Genre(
    genre_id SERIAL PRIMARY KEY,
    genre_title CHARACTER VARYING (20)
);
--4 Dimension table-Director
CREATE TABLE Director(
    director_id SERIAL PRIMARY KEY,
    dir_first_name CHARACTER VARYING (20) NOT NULL,
    dir_last_name CHARACTER VARYING (20)
);
```



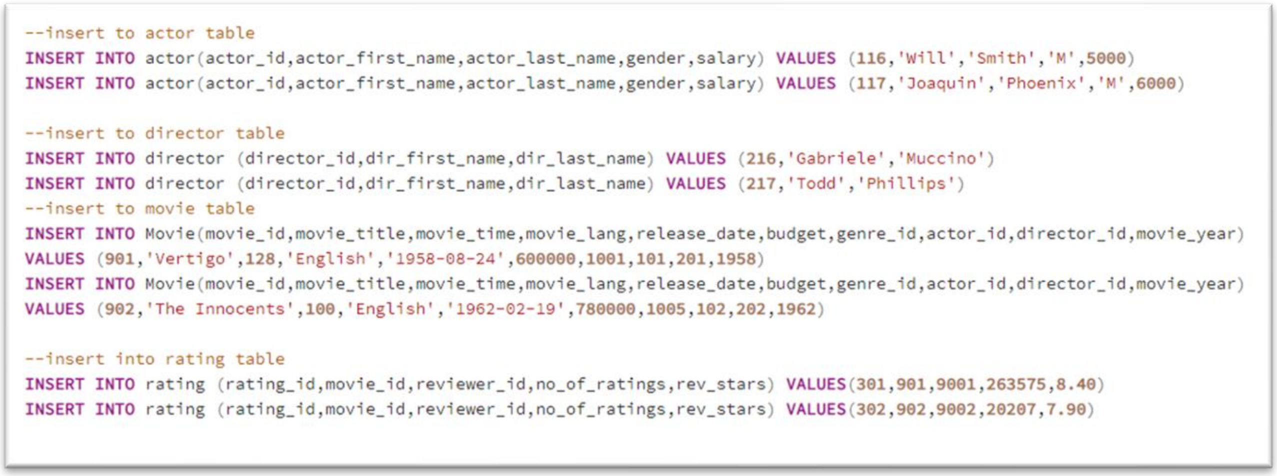
```
--5 Dimension table-Reviewer
CREATE TABLE Reviewer(
    reviewer_id SERIAL PRIMARY KEY,
    reviewer_name CHARACTER VARYING (20)
);
--6 Fact table-Rating
CREATE TABLE RATING(
    rating_id SERIAL PRIMARY KEY,
    movie_id INTEGER NOT NULL,
    reviewer_id INTEGER NOT NULL,
    no_of_ratings INTEGER,
    rev_stars NUMERIC(3,2),
    FOREIGN KEY (movie_id) REFERENCES Movie (movie_id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (reviewer_id) REFERENCES Reviewer (reviewer_id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

INSERT QUERIES:

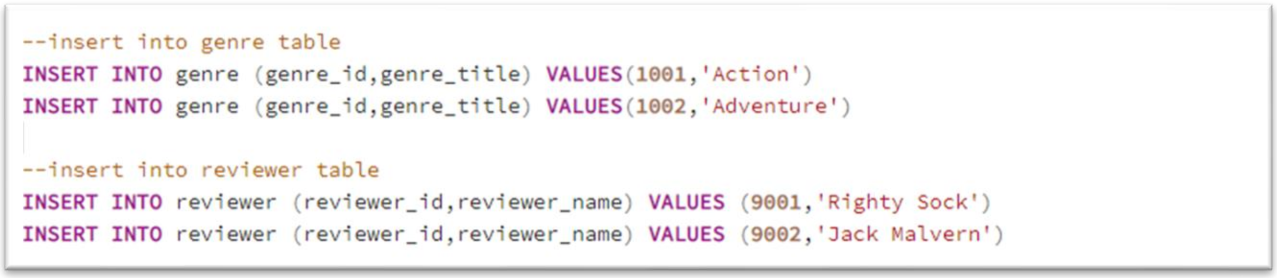
No of rows inserted in each table:

1) Actor-17
2) Director-17
3) Movie-30
4) Rating-30
5) Genre-13
6) Reviewer-15

Only 2 queries are shown here for reference for each table:



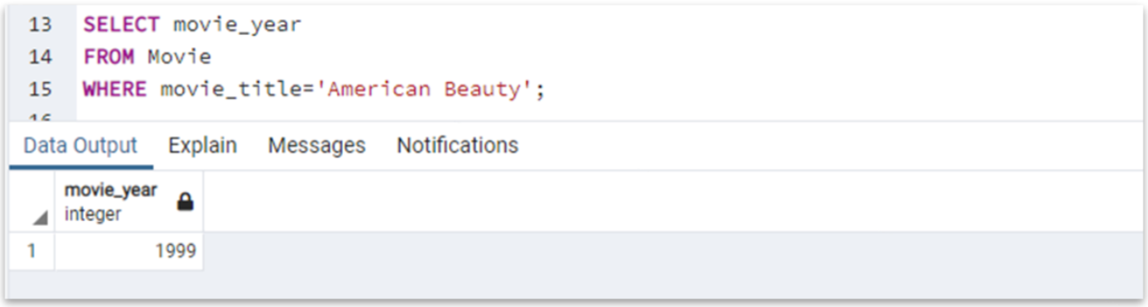
```
--insert to actor table
INSERT INTO actor(actor_id,actor_first_name,actor_last_name,gender,salary) VALUES (116,'Will','Smith','M',5000)
INSERT INTO actor(actor_id,actor_first_name,actor_last_name,gender,salary) VALUES (117,'Joaquin','Phoenix','M',6000)

--insert to director table
INSERT INTO director (director_id,dir_first_name,dir_last_name) VALUES (216,'Gabriele','Muccino')
INSERT INTO director (director_id,dir_first_name,dir_last_name) VALUES (217,'Todd','Phillips')
--insert to movie table
INSERT INTO Movie(movie_id,movie_title,movie_time,movie_lang,release_date,budget,genre_id,actor_id,director_id,movie_year)
VALUES (901,'Vertigo',128,'English','1958-08-24',600000,1001,101,201,1958)
INSERT INTO Movie(movie_id,movie_title,movie_time,movie_lang,release_date,budget,genre_id,actor_id,director_id,movie_year)
VALUES (902,'The Innocents',100,'English','1962-02-19',780000,1005,102,202,1962)

--insert into rating table
INSERT INTO rating (rating_id,movie_id,reviewer_id,no_of_ratings,rev_stars) VALUES(301,901,9001,263575,8.40)
INSERT INTO rating (rating_id,movie_id,reviewer_id,no_of_ratings,rev_stars) VALUES(302,902,9002,20207,7.90)
```



```
--insert into genre table
INSERT INTO genre (genre_id,genre_title) VALUES(1001,'Action')
INSERT INTO genre (genre_id,genre_title) VALUES(1002,'Adventure')

--insert into reviewer table
INSERT INTO reviewer (reviewer_id,reviewer_name) VALUES (9001,'Righty Sock')
INSERT INTO reviewer (reviewer_id,reviewer_name) VALUES (9002,'Jack Malvern')
```

Analysis of questions:

1. Find when the movie ‘American Beauty’ released. Return movie release year.
   The query and output for this question is: In this query, conditional operator **WHERE** is used.

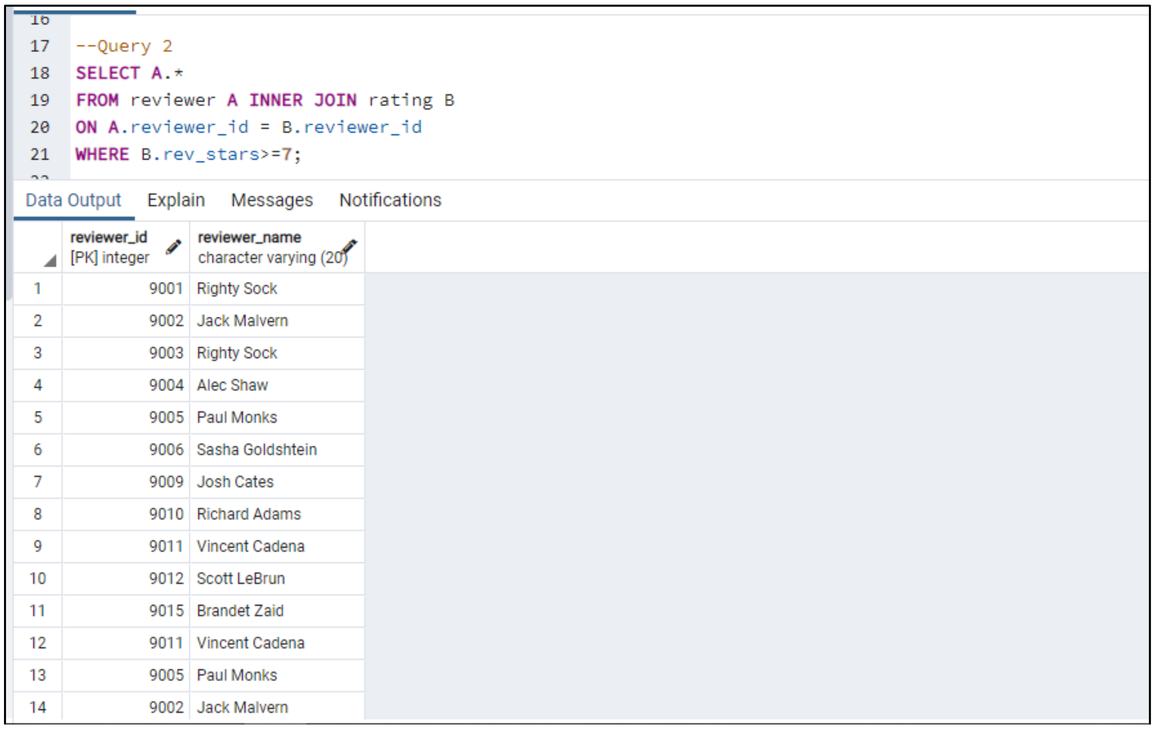


```
SELECT movie_year
FROM Movie
WHERE movie_title='American Beauty';
```

Data Output | Explain | Messages | Notifications

```
   movie_year
   integer
1        1999
```

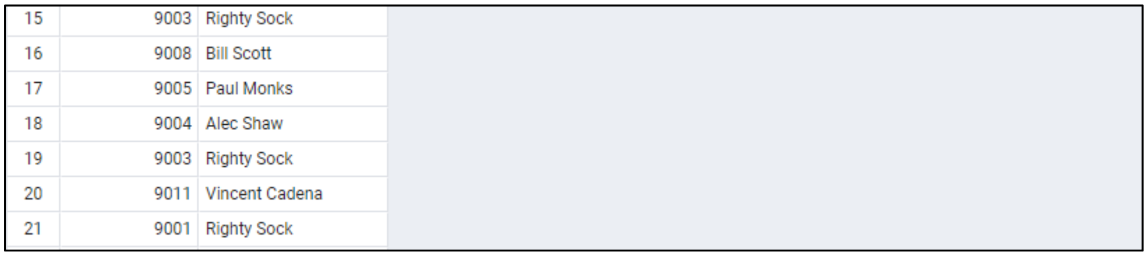
2. Find all reviewers who have rated 7 or more stars to their rating.   In this query **JOIN** is used.



```
--Query 2
SELECT A.*
FROM reviewer A INNER JOIN rating B
ON A.reviewer_id = B.reviewer_id
WHERE B.rev_stars>=7;
```

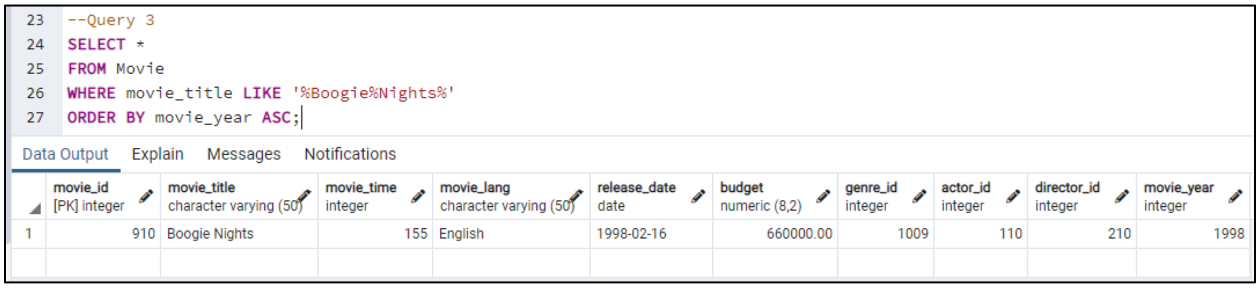
Data Output | Explain | Messages | Notifications

```
    reviewer_id    reviewer_name
    [PK] integer   character varying (20)
1          9001    Righty Sock
2          9002    Jack Malvern
3          9003    Righty Sock
4          9004    Alec Shaw
5          9005    Paul Monks
6          9006    Sasha Goldshtein
7          9009    Josh Cates
8          9010    Richard Adams
9          9011    Vincent Cadena
10         9012    Scott LeBrun
11         9015    Brandet Zaid
12         9011    Vincent Cadena
13         9005    Paul Monks
14         9002    Jack Malvern
```



15	9003	Righty Sock
16	9008	Bill Scott
17	9005	Paul Monks
18	9004	Alec Shaw
19	9003	Righty Sock
20	9011	Vincent Cadena
21	9001	Righty Sock

3. Find all movies whose movie titles include the words 'Boogie Nights'. Sort the result-set in ascending order by movie year. In this query **LIKE** and **ORDER BY** operators are used.

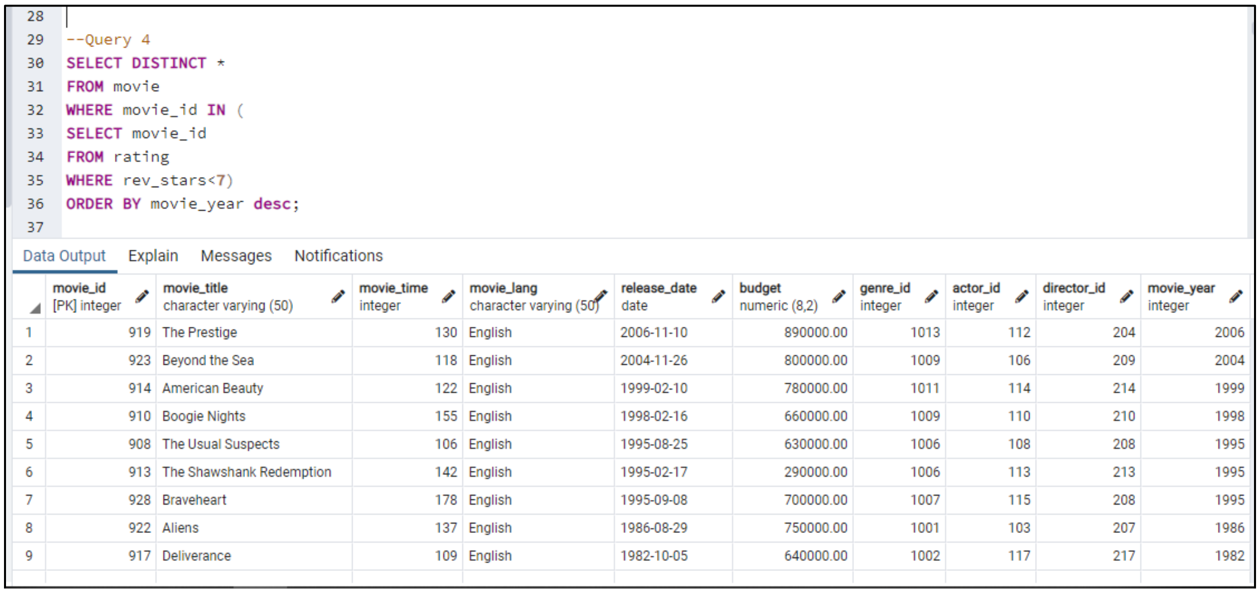


```sql
--Query 3
SELECT *
FROM Movie
WHERE movie_title LIKE '%Boogie%Nights%'
ORDER BY movie_year ASC;
```

Data Output   Explain   Messages   Notifications

	movie_id [PK] integer	movie_title character varying (50)	movie_time integer	movie_lang character varying (50)	release_date date	budget numeric (8,2)	genre_id integer	actor_id integer	director_id integer	movie_year integer
1	910	Boogie Nights	155	English	1998-02-16	660000.00	1009	110	210	1998

4. Write a SQL query to find the movies who got ratings below 7. Return result in decreasing order of year. In this query **SUBQUERY** is used.

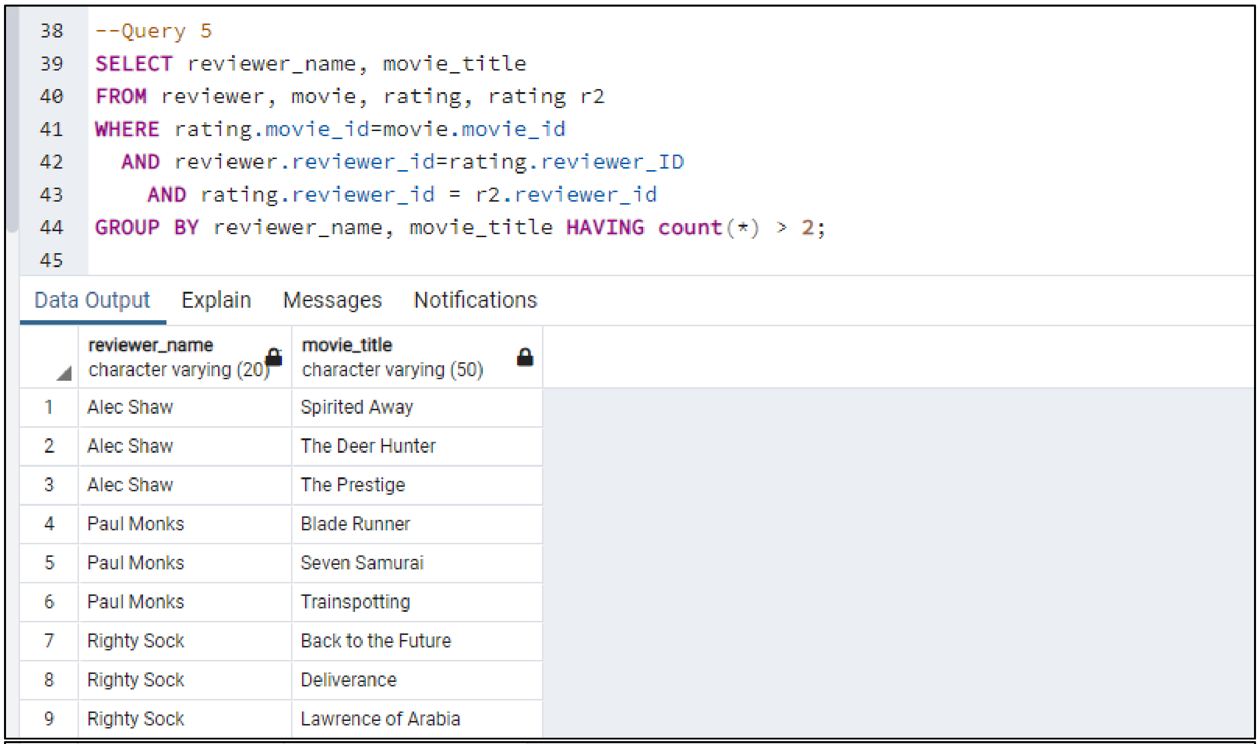


```sql

--Query 4
SELECT DISTINCT *
FROM movie
WHERE movie_id IN (
SELECT movie_id
FROM rating
WHERE rev_stars<7)
ORDER BY movie_year desc;

```

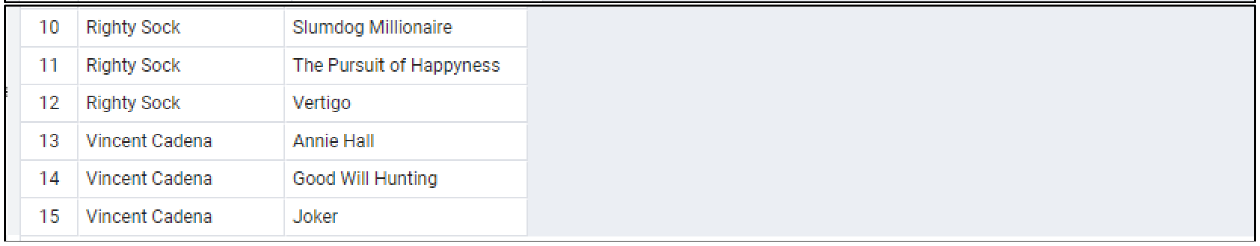
Data Output   Explain   Messages   Notifications

	movie_id [PK] integer	movie_title character varying (50)	movie_time integer	movie_lang character varying (50)	release_date date	budget numeric (8,2)	genre_id integer	actor_id integer	director_id integer	movie_year integer
1	919	The Prestige	130	English	2006-11-10	890000.00	1013	112	204	2006
2	923	Beyond the Sea	118	English	2004-11-26	800000.00	1009	106	209	2004
3	914	American Beauty	122	English	1999-02-10	780000.00	1011	114	214	1999
4	910	Boogie Nights	155	English	1998-02-16	660000.00	1009	110	210	1998
5	908	The Usual Suspects	106	English	1995-08-25	630000.00	1006	108	208	1995
6	913	The Shawshank Redemption	142	English	1995-02-17	290000.00	1006	113	213	1995
7	928	Braveheart	178	English	1995-09-08	700000.00	1007	115	208	1995
8	922	Aliens	137	English	1986-08-29	750000.00	1001	103	207	1986
9	917	Deliverance	109	English	1982-10-05	640000.00	1002	117	217	1982

5. Write a SQL query to find those reviewers who rated more than two movies. Group the result set on reviewer's name, movie title. In this query **JOINS**, **GROUP BY** and **HAVING** clause are used.

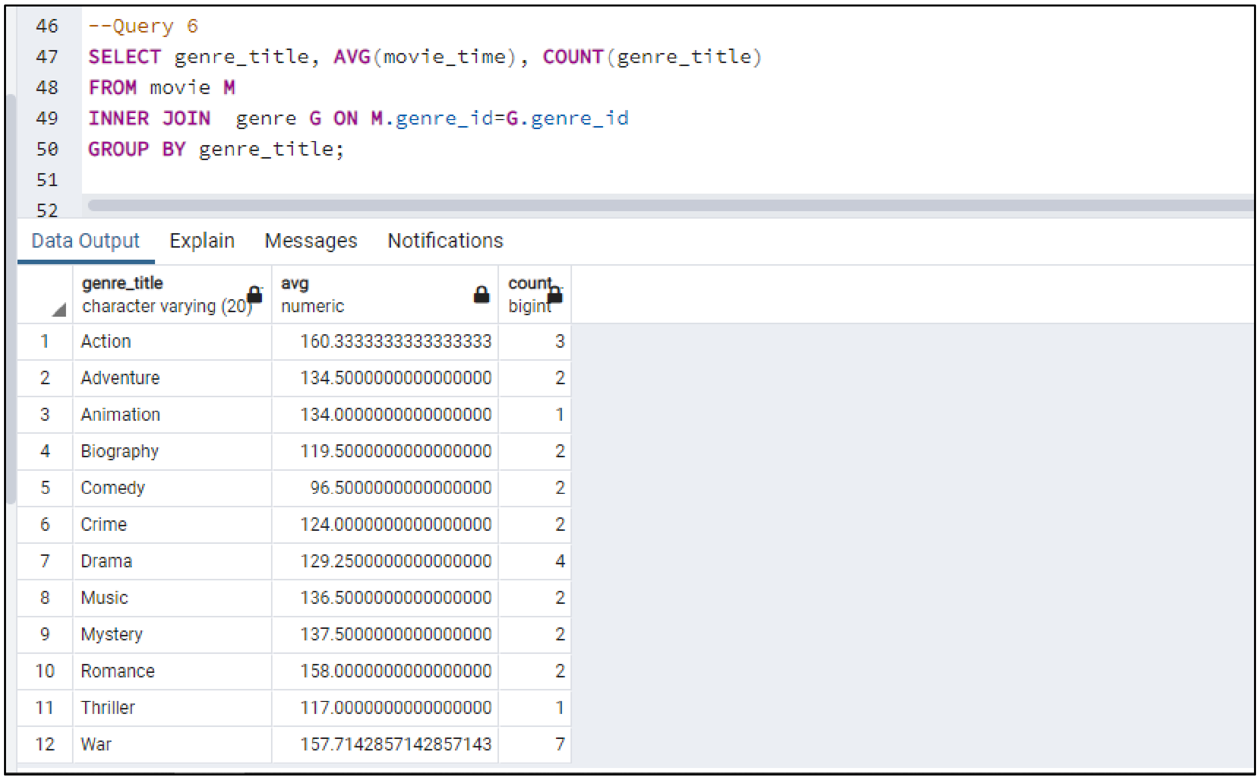


```
--Query 5
SELECT reviewer_name, movie_title
FROM reviewer, movie, rating, rating r2
WHERE rating.movie_id=movie.movie_id
  AND reviewer.reviewer_id=rating.reviewer_ID
    AND rating.reviewer_id = r2.reviewer_id
GROUP BY reviewer_name, movie_title HAVING count(*) > 2;

```



	reviewer_name character varying (20)	movie_title character varying (50)
1	Alec Shaw	Spirited Away
2	Alec Shaw	The Deer Hunter
3	Alec Shaw	The Prestige
4	Paul Monks	Blade Runner
5	Paul Monks	Seven Samurai
6	Paul Monks	Trainspotting
7	Righty Sock	Back to the Future
8	Righty Sock	Deliverance
9	Righty Sock	Lawrence of Arabia
10	Righty Sock	Slumdog Millionaire
11	Righty Sock	The Pursuit of Happyness
12	Righty Sock	Vertigo
13	Vincent Cadena	Annie Hall
14	Vincent Cadena	Good Will Hunting
15	Vincent Cadena	Joker

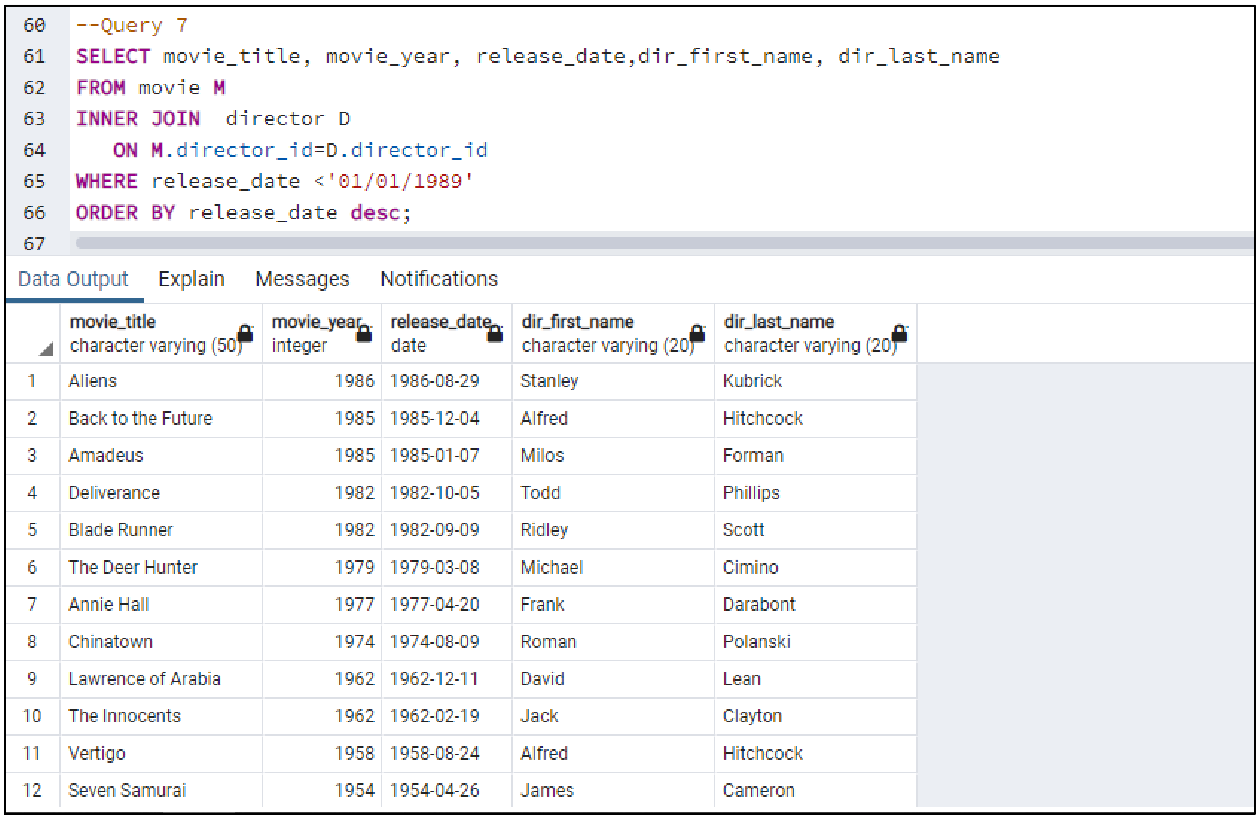
6. Write a SQL query to compute the average time and count number of movies for each genre. Return genre title, average time and number of movies for each genre. In this query Aggregate functions **AVERAGE** and **COUNT** are used.



```
--Query 6
SELECT genre_title, AVG(movie_time), COUNT(genre_title)
FROM movie M
INNER JOIN  genre G ON M.genre_id=G.genre_id
GROUP BY genre_title;
```

	genre_title character varying (20)	avg numeric	count bigint
1	Action	160.3333333333333333	3
2	Adventure	134.5000000000000000	2
3	Animation	134.0000000000000000	1
4	Biography	119.5000000000000000	2
5	Comedy	96.5000000000000000	2
6	Crime	124.0000000000000000	2
7	Drama	129.2500000000000000	4
8	Music	136.5000000000000000	2
9	Mystery	137.5000000000000000	2
10	Romance	158.0000000000000000	2
11	Thriller	117.0000000000000000	1
12	War	157.7142857142857143	7

7. Write a SQL query to find the movies released before 1st January 1989. Sort the result-set in descending order by date of release. Return movie title, release year, date of release, and first and last name of the director. In this query Less than**(<) operator** is used along with **JOIN**.

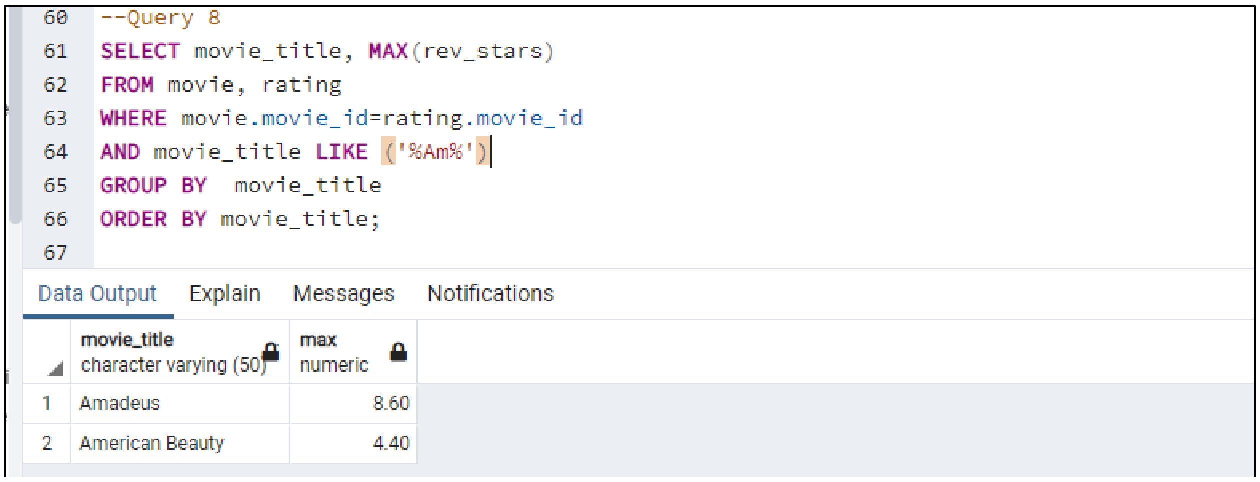


```
--Query 7
SELECT movie_title, movie_year, release_date,dir_first_name, dir_last_name
FROM movie M
INNER JOIN  director D
   ON M.director_id=D.director_id
WHERE release_date <'01/01/1989'
ORDER BY release_date desc;
```

Data Output Explain Messages Notifications

	movie_title character varying (50)	movie_year integer	release_date date	dir_first_name character varying (20)	dir_last_name character varying (20)
1	Aliens	1986	1986-08-29	Stanley	Kubrick
2	Back to the Future	1985	1985-12-04	Alfred	Hitchcock
3	Amadeus	1985	1985-01-07	Milos	Forman
4	Deliverance	1982	1982-10-05	Todd	Phillips
5	Blade Runner	1982	1982-09-09	Ridley	Scott
6	The Deer Hunter	1979	1979-03-08	Michael	Cimino
7	Annie Hall	1977	1977-04-20	Frank	Darabont
8	Chinatown	1974	1974-08-09	Roman	Polanski
9	Lawrence of Arabia	1962	1962-12-11	David	Lean
10	The Innocents	1962	1962-02-19	Jack	Clayton
11	Vertigo	1958	1958-08-24	Alfred	Hitchcock
12	Seven Samurai	1954	1954-04-26	James	Cameron

8. Write a SQL query to find those movies, which have received highest number of stars and name starts with “Am”. Group the result set on movie title and sorts the result-set in ascending order by movie title. Return movie title and maximum number of review stars. In this query **MAX** and **LIKE** are used.

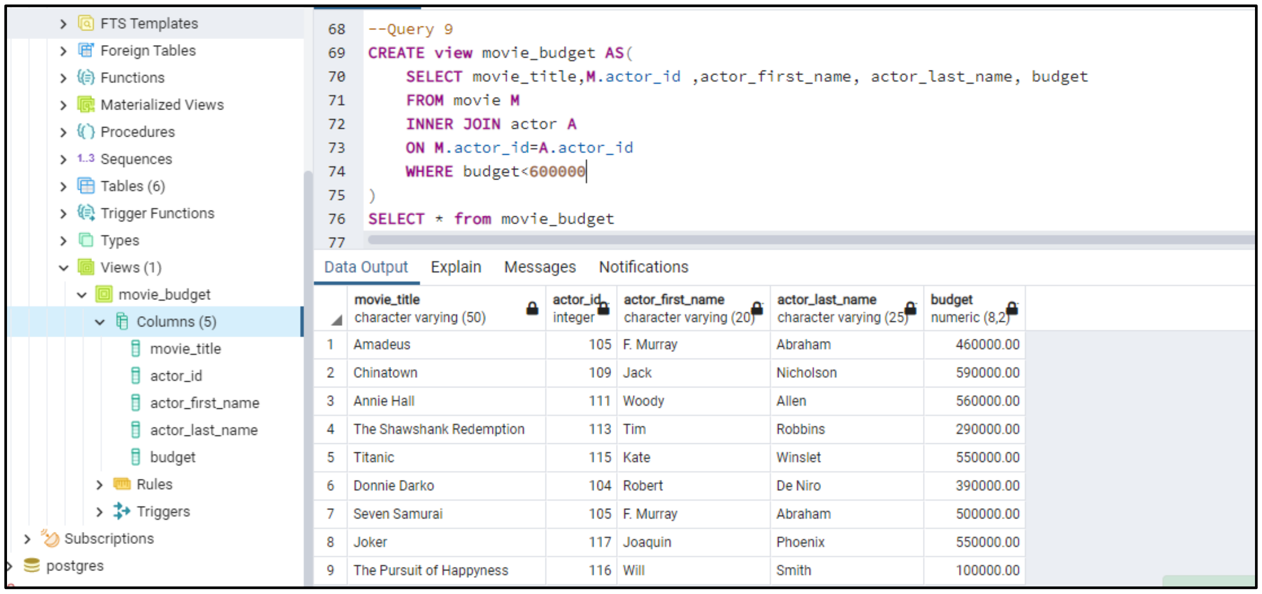


```
--Query 8
SELECT movie_title, MAX(rev_stars)
FROM movie, rating
WHERE movie.movie_id=rating.movie_id
AND movie_title LIKE ('%Am%')
GROUP BY  movie_title
ORDER BY movie_title;
```

Data Output Explain Messages Notifications

	movie_title character varying (50)	max numeric
1	Amadeus	8.60
2	American Beauty	4.40

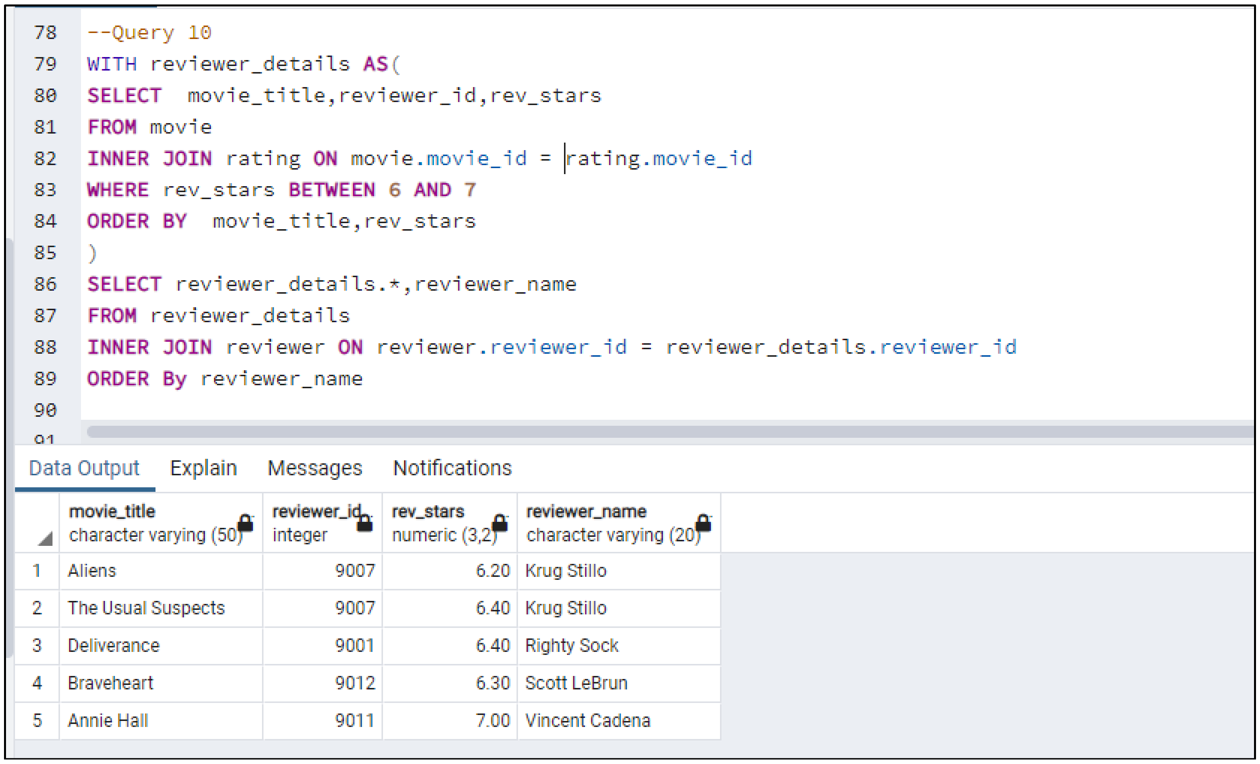
9. Create a **VIEW** to fetch all the actors who acted in movies whose budget was less than 6000000. Fetch columns movie_title, actor_id, actor_first_name, actor_last_name and budget.



```sql
--Query 9
CREATE view movie_budget AS(
    SELECT movie_title,M.actor_id ,actor_first_name, actor_last_name, budget
    FROM movie M
    INNER JOIN actor A
    ON M.actor_id=A.actor_id
    WHERE budget<600000
)
SELECT * from movie_budget
```

	movie_title character varying (50)	actor_id integer	actor_first_name character varying (20)	actor_last_name character varying (25)	budget numeric (8,2)
1	Amadeus	105	F. Murray	Abraham	460000.00
2	Chinatown	109	Jack	Nicholson	590000.00
3	Annie Hall	111	Woody	Allen	560000.00
4	The Shawshank Redemption	113	Tim	Robbins	290000.00
5	Titanic	115	Kate	Winslet	550000.00
6	Donnie Darko	104	Robert	De Niro	390000.00
7	Seven Samurai	105	F. Murray	Abraham	500000.00
8	Joker	117	Joaquin	Phoenix	550000.00
9	The Pursuit of Happyness	116	Will	Smith	100000.00

10. Find the reviewer's name, movie title, and stars within range 6-8 in an order that movie title will come first, then by number of stars, and lastly by reviewer name. (Use **CTE**)



```sql
--Query 10
WITH reviewer_details AS(
SELECT  movie_title,reviewer_id,rev_stars
FROM movie
INNER JOIN rating ON movie.movie_id = rating.movie_id
WHERE rev_stars BETWEEN 6 AND 7
ORDER BY  movie_title,rev_stars
)
SELECT reviewer_details.*,reviewer_name
FROM reviewer_details
INNER JOIN reviewer ON reviewer.reviewer_id = reviewer_details.reviewer_id
ORDER By reviewer_name
```

	movie_title character varying (50)	reviewer_id integer	rev_stars numeric (3,2)	reviewer_name character varying (20)
1	Aliens	9007	6.20	Krug Stillo
2	The Usual Suspects	9007	6.40	Krug Stillo
3	Deliverance	9001	6.40	Righty Sock
4	Braveheart	9012	6.30	Scott LeBrun
5	Annie Hall	9011	7.00	Vincent Cadena

SET OF QUERIES USED:

```
-- Query1

SELECT movie_year

FROM Movie

WHERE movie_title='American Beauty';
```

```
--Query 2

SELECT A.*

FROM reviewer A INNER JOIN rating B

ON A.reviewer_id = B.reviewer_id
```

```
--Query 3

SELECT *

FROM Movie

WHERE movie_title LIKE '%Boogie%Nights%'
```

```
--Query 4

SELECT DISTINCT *

FROM movie

WHERE movie_id IN (

SELECT movie_id

FROM rating

WHERE rev_stars<7)

ORDER BY movie_year desc;

WHERE B.rev_stars>=7;
```

```
--Query 5

SELECT reviewer_name, movie_title

FROM reviewer, movie, rating, rating r2

WHERE rating.movie_id=movie.movie_id

  AND reviewer.reviewer_id=rating.reviewer_ID

    AND rating.reviewer_id = r2.reviewer_id

GROUP BY reviewer_name, movie_title HAVING
count(*) > 2;

ORDER BY movie_year ASC;
```

```
--Query 6

SELECT genre_title, AVG(movie_time),
COUNT(genre_title)

FROM movie M

INNER JOIN  genre G ON M.genre_id=G.genre_id

GROUP BY genre_title;

WHERE B.rev_stars>=7;
```

```
--Query 7

SELECT movie_title, movie_year,
release_date,dir_first_name, dir_last_name

FROM movie M

INNER JOIN  director D

   ON M.director_id=D.director_id

WHERE release_date <'01/01/1989'

ORDER BY release_date desc;
```

```
--Query 8

SELECT movie_title, MAX(rev_stars)

FROM movie, rating

WHERE movie.movie_id=rating.movie_id

AND movie_title LIKE ('%Am%')

GROUP BY  movie_title

ORDER BY movie_title;
```

```
--Query 9

CREATE view movie_budget AS(

        SELECT movie_title,M.actor_id
,actor_first_name, actor_last_name, budget

        FROM movie M

        INNER JOIN actor A

        ON M.actor_id=A.actor_id

        WHERE budget<600000

)

SELECT * from movie_budget
```

```
--Query 10

WITH reviewer_details AS(

SELECT  movie_title,reviewer_id,rev_stars

FROM movie

INNER JOIN rating ON movie.movie_id = rating.movie_id

WHERE rev_stars BETWEEN 6 AND 7

ORDER BY  movie_title,rev_stars

)

SELECT reviewer_details.*,reviewer_name

FROM reviewer_details

INNER JOIN reviewer ON reviewer.reviewer_id = reviewer_details.reviewer_id

ORDER By reviewer_name
```