DATABASE PROJECT PART-2

By: Pratyusha Sanapathi

- **Finalize the dataset(s) that you proposed in Part I and finalize your 5-10 different tables from your dataset(s). And if you need to change some of your dataset resources, mention the reason. Describe how you are normalizing the original dataset if you create multiple tables.**

Data Finalizing:

- I've finalized total 9 datasets for my database. The raw datasets contain the data of a movie management system.
- My dataset contained redundant data and so I normalized with the help of Python's panda tools.
- Python's pandas tool is a high-level data manipulation tool. Prior to building the database and filling up the relations, the movie management system database could be easily created and efficiently used.

Normalization:

In my raw dataset, I noticed that there are two columns which are similar in a single table. So, using Pandas library in Python, I have separated them into two different tables along with their respective attributes to make the dataset more efficient and readable.
I've imported csv file as data frame named df, read the csv file and printed the dataframe.

```
In [23]:  import pandas as pd
          df = pd.read_csv(r'C:\Users\Pratyusha Sanapathi\Desktop\Assignments\DB\movie_data\raw_data.csv')

In [24]:  df
```

Out[24]:

	dir_id	mov_id	dir_id.1	dir_fname	dir_lname
0	201	901	201	Alfred	Hitchcock
1	202	902	202	Jack	Clayton
2	203	903	203	David	Lean
3	204	904	204	Michael	Cimino
4	205	905	205	Milos	Forman
...	...	...	...	...	...
97	298	998	298	Cindy	Girling
98	299	999	299	Ricky	Poulsom
99	300	1000	300	Sal	Scothern
100	301	1001	301	Konstance	Dyett
101	302	1002	302	Fax	Shobrook

102 rows × 5 columns

I've used the below command to find out duplicity between two columns.

```
In [31]: df['dir_id'].equals(df['dir_id.1'])
Out[31]: True
```

After this, I've segregated the raw dataset into two different tables and kept them in two data frame (df1, df2) by following the process which is mentioned below.

```
df1 = df[['dir_id', 'mov_id']]
df2 = df[['dir_id.1', 'dir_fname', 'dir_lname']]
```

```
df1
```

]:

	dir_id	mov_id
0	201	901
1	202	902
2	203	903
3	204	904
4	205	905
...	...	...
97	298	998
98	299	999
99	300	1000
100	301	1001
101	302	1002

102 rows × 2 columns

```
df2
```

]:

	dir_id.1	dir_fname	dir_lname
0	201	Alfred	Hitchcock
1	202	Jack	Clayton
2	203	David	Lean
3	204	Michael	Cimino
4	205	Milos	Forman
...	...	...	...
97	298	Cindy	Girling
98	299	Ricky	Poulsom
99	300	Sal	Scothern
100	301	Konstance	Dyett
101	302	Fax	Shobrook

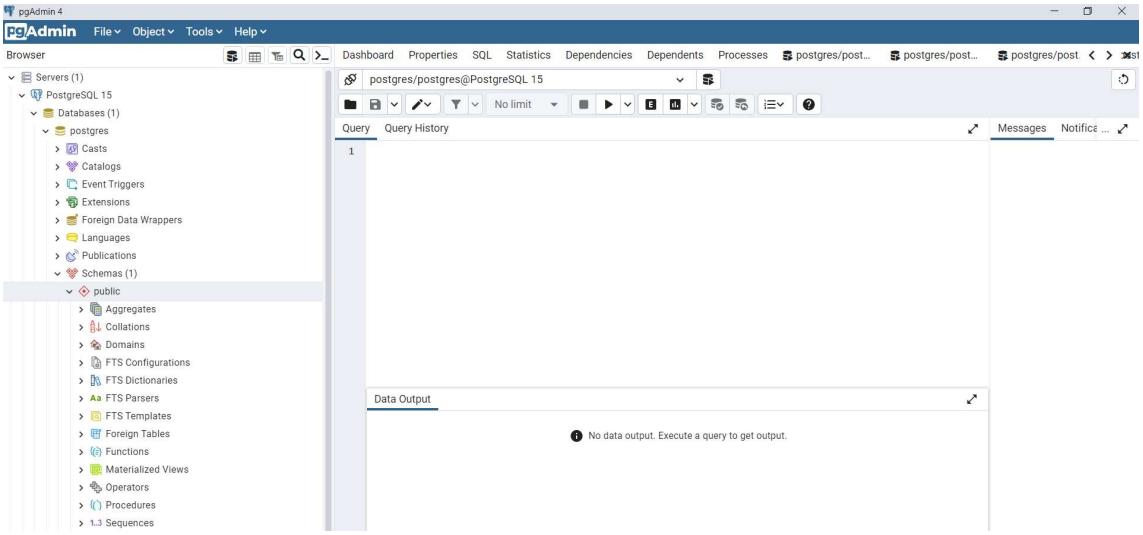
102 rows × 3 columns

● **[5 points] Show that you have successfully installed Postgres and you can use pgAdmin or psql on your local machine or virtual machine (you can provide a screenshot).**

PostgreSQL Installation

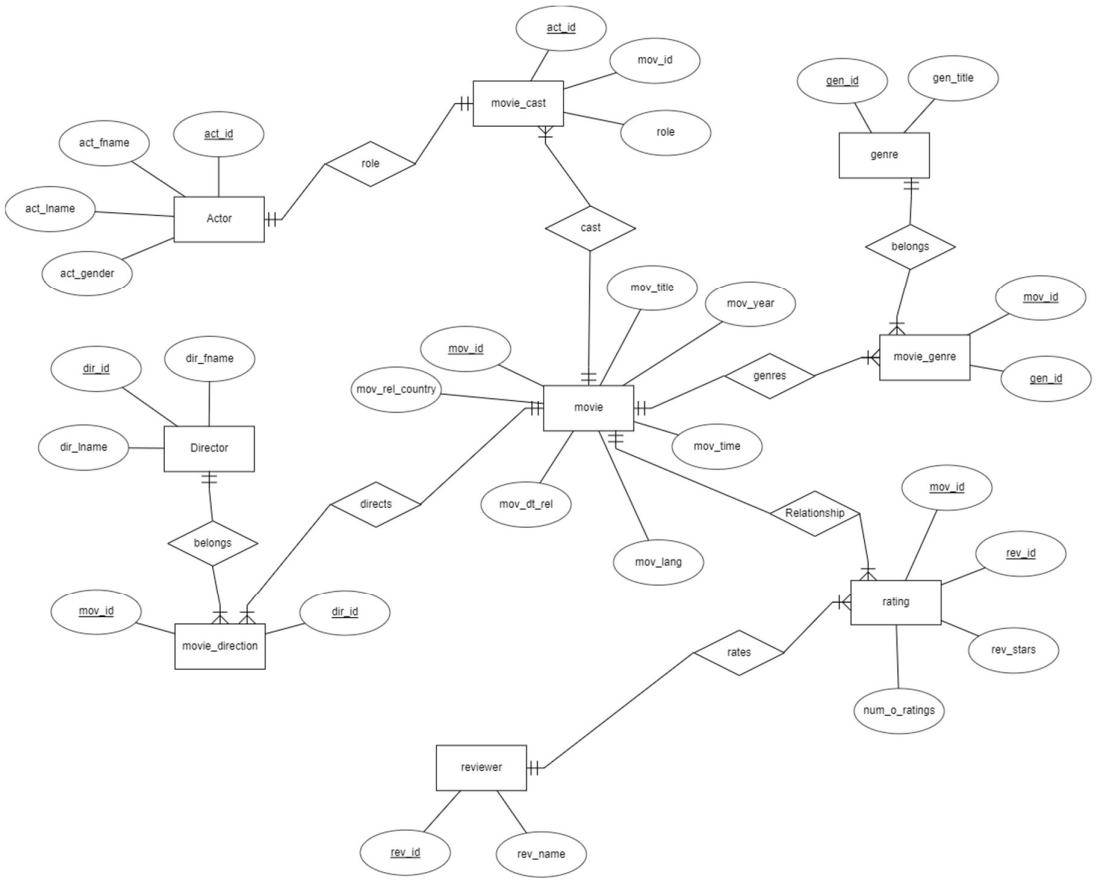
```
Server [localhost]: localhost
Database [postgres]: postgres
Port [5432]: 5432
Username [postgres]: postgres
Password for user postgres:
psql (15.0)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# SELECT version();
                           version
-----------------------------------------------------------
 PostgreSQL 15.0, compiled by Visual C++ build 1914, 64-bit
(1 row)
```



● [20 points] ER diagram.

ER Diagram:



● [25 points] Database schema designed with your designed tables, attribute declaration, primary/foreign keys, views, or temporary tables etc. (You should demonstrate a variety of schema and SQL features such as a variety of data types, keys, foreign keys, different cardinalities).

Database Schema:

Actor Table:

actor(act_id INT, act_fname VARCHAR, act_lname VARCHAR, act_gender VARCHAR)

act_id is the primary key

Director Table:

director(dir_id INT, dir_fname VARCHAR, dir_lname VARCHAR)

dir_id is the primary key

Genre Table:

genre(gen_id INT, gen_title VARCHAR)

gen_id is the primary key

Movie Table:

movie(mov_id INT, mov_title VARCHAR, mov_year INT, mov_time INT, mov_lang VARCHAR, mov_dt_rel DATE, mov_rel_country VARCHAR)

mov_id is the primary key

Reviewer Table:

reviewer(rev_id INT, rev_name VARCHAR)

rev_id is the primary key

Movie_cast Table:

movie_cast(act_id INT, mov_id INT, role VARCHAR)

act_id is the primary key

- act_id is the foreign key references the actor(act_id)
- mov_id is the foreign key references the movie(mov_id)

Movie_direction Table:

movie_direction(dir_id INT, mov_id INT)

dir_id and mov_id will work together as primary key

- dir_id is the foreign key references the director(dir_id)
- mov_id is the foreign key references the movie(mov_id)

Movie_genre Table:

movie_genre(mov_id INT, gen_id INT)

mov_id and gen_id will work together as primary key

- mov_id is the foreign key references the movie(mov_id)
- gen_id is the foreign key references the genre(gen_id)

Rating Table:

rating(mov_id INT,rev_id INT, rev_stars FLOAT,num_o_ratings INT)

- rev_id is the foreign key references the reviewer(rev_id)
- mov_id is the foreign key references the movie(mov_id)

● [45 points] Data loading process and data preprocessing and cleaning. You should have effective data entry and avoid large amounts of manual data

entry. Please describe what you have done to clean your data fully, what you have considered in cleaning, and how you have chosen to load and import your data in your tables.

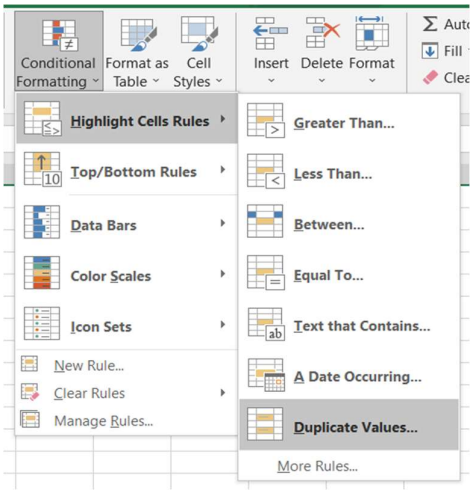
Data pre-processing and cleaning:

Any type of processing done on raw data to get it ready for another data processing operation is referred to as data pre-processing, which is a part of data preparation. The tasks involved in data preparation and filtering can take a long time to process. Raw data frequently has uneven formatting and is insufficient.

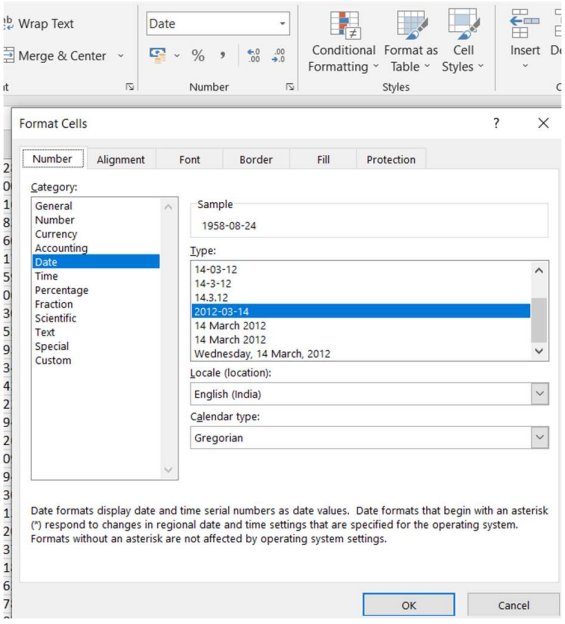
The type and format of data that we import from an external data source isn't necessarily under our control. Frequently, the data needs to be cleaned up before we can evaluate it. So, after analysing my dataset which is the information related to Movies, I understood that there's necessity for data pre-processing. In order to overcome that, I chose excel for this process as Excel includes a ton of tools that helped me to collect data in the exact manner that I need for implementing this project.

Using the Remove Duplicates dialog box, I have removed duplicate rows from a table. In other cases, we might have to modify one or more columns by applying a formula to change the imported values into fresh ones. For instance, in my case, I have constructed a new column and used a formula to clean the data by removing trailing spaces. Sometimes data isn't even organized in a tabular manner, in which case we will need a method to convert it to a tabular format. So according to the bad data in my dataset, I used TRANSPOSE function and got the data for a column in one of the tables which can count under transforming and rearranging columns and rows. Fixing dates and times and converting them to correct format is another change that I implemented here. Below are some of the examples that I used to correct my dataset.

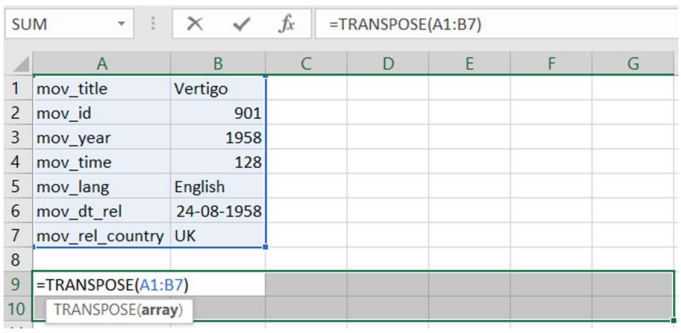
Removing duplicate rows

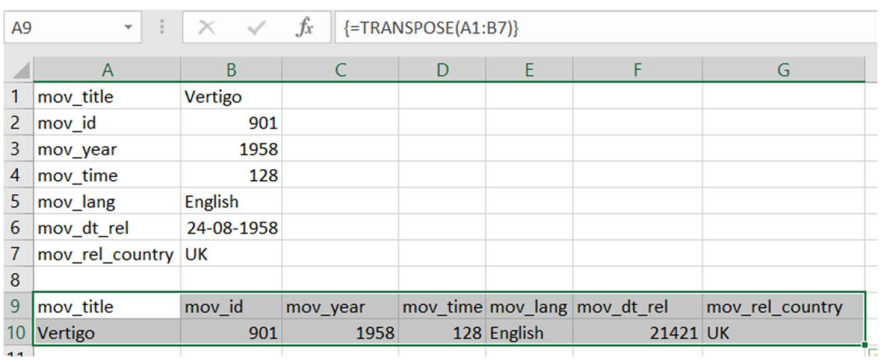


Changing the Date format



Transforming and rearranging the rows and columns as required





Data Loading Process:

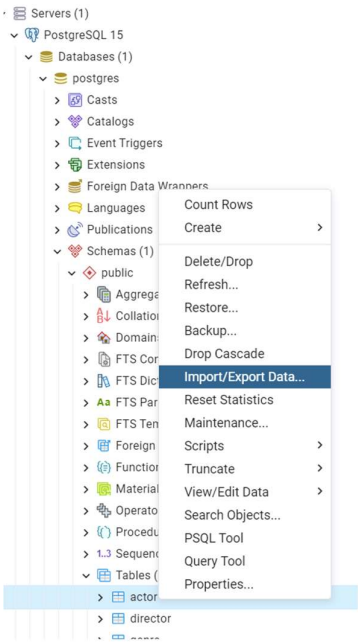
A web-based Graphical User Interface (GUI) management tool called pgAdmin is used to interact with relational databases like Postgres and its offshoots on both local and distant servers. I have pushed the data by creating tables (using SQL queries) in the Query tool console as per the requirement and imported the related csv files from the local device which contains the data.

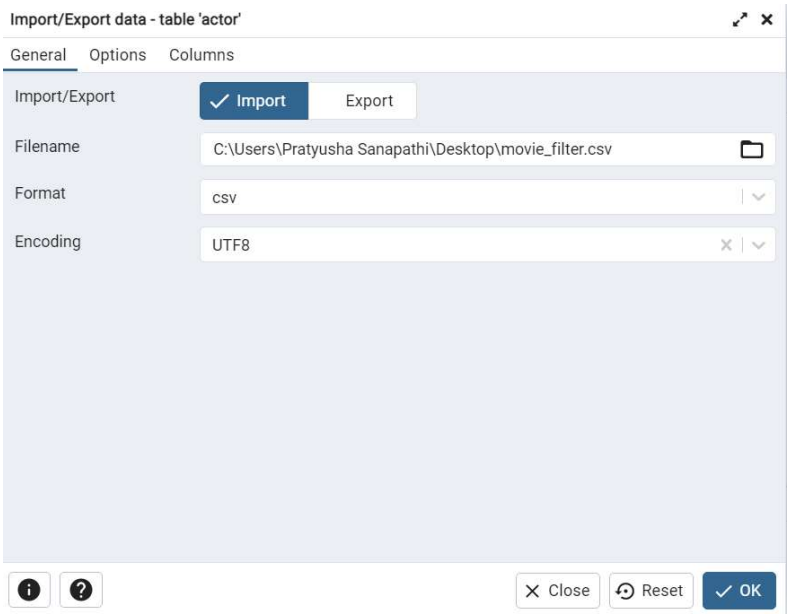
postgres/postgres@PostgreSQL 15

No limit

Query | Query History

```sql
CREATE table actor(
act_id int,
act_fname varchar,
act_lname varchar,
PRIMARY KEY(act_id)
);
```





In the above manner, I have created 9 tables and loaded the data.

● [5 points] Screenshots of the first five rows of all the populated tables. Your screenshots need to show sufficiently show your working environment as well.

<u>Screenshots of Tables:</u>

Actor Table:

```
select * from actor;
```

Data Output

	act_id [PK] integer	act_fname character varying	act_lname character varying	act_gender character varying
1	101	James	Stewart	M
2	102	Deborah	Kerr	F
3	103	Peter	OToole	M
4	104	Robert	DeNiro	M
5	106	Harrison	Ford	M

Director Table:

```
select * from director;
```

Data Output

	dir_id [PK] integer	dir_fname character varying	dir_lname character varying
1	201	Alfred	Hitchcock
2	202	Jack	Clayton
3	203	David	Lean
4	204	Michael	Cimino
5	205	Milos	Forman

Genre Table:

```
select * from genre;
```

Data Output

	gen_id [PK] integer	gen_title character varying
1	1001	Action
2	1002	Adventure
3	1003	Animation
4	1004	Biography
5	1005	Comedy

Movie Table:

```
select * from movie;
```

Data Output

	mov_title character varying	mov_id [PK] integer	mov_year integer	mov_time integer	mov_lang character varying	mov_dt_rel date	mov_rel_country character varying
1	Vertigo	901	1958	128	English	1958-08-24	UK
2	The Innocents	902	1961	100	English	1962-02-19	SW
3	Lawrence of Ara...	903	1962	216	English	1962-12-11	UK
4	The Deer Hunter	904	1978	183	English	1979-03-08	UK
5	Amadeus	905	1984	160	English	1985-01-07	UK

Moive_cast Table:

```
select * from movie_cast;
```

Data Output

	act_id integer	mov_id integer	role character varying
1	101	901	John Scottie Fer...
2	102	902	Miss Giddens
3	103	903	T.E. Lawrence
4	104	904	Michael
5	106	905	Rick Deckard

Movie_direction Table:

```
select * from movie_direction;
```

Data Output

	dir_id integer	mov_id integer
1	201	901
2	202	902
3	203	903
4	204	904
5	205	905

Movie_genre Table:

```
select * from movie_genre;
```

Data Output

	gen_id integer	mov_id integer
1	1001	901
2	1002	902
3	1003	903
4	1004	904
5	1005	905

Rating Table:

```
select * from rating;
```

Data Output

	mov_id integer	rev_id integer	rev_stars double precision	num_o_ratings integer
1	901	9001	8.4	263575
2	902	9002	7.9	20207
3	903	9003	8.3	202778
4	904	9004	8.2	484746
5	905	9005	7.3	546908

Reviewer Table:

```
select * from reviewer;
```

Data Output

	rev_id [PK] integer	rev_name character varying
1	9001	Righty Sock
2	9002	Jack Malvern
3	9003	Flagrant Barones...
4	9004	Alec Shaw
5	9005	Jane Fusedale

● [100 points] Your 10 English questions and 10 query execution of those questions with screenshots of the first five rows of output and the total number of rows in the result. Design a variety of complex questions, as mentioned earlier.

<u>Queries:</u>

1) Write a SQL query to find those movies that have been released in countries other than the United Kingdom. Return movie title, movie year, movie time, and date of release, releasing country.

(*Changed Question. Reason:* *The question I've mentioned in part-1 is way too simple. In order to make the query a bit complicated, I changed it with some extra conditions.*)

ANS:

```
SELECT mov_title, mov_year, mov_time,

mov_dt_rel AS Date_of_Release,

mov_rel_country AS Releasing_Country

FROM movie

WHERE mov_rel_country<>'UK';
```

```sql
SELECT mov_title, mov_year, mov_time,
mov_dt_rel AS Date_of_Release,
mov_rel_country AS Releasing_Country
FROM movie
WHERE mov_rel_country<>'UK';
```

Data Output

	mov_title character varying	mov_year integer	mov_time integer	date_of_release date	releasing_country character varying
1	The Innocents	1961	100	1962-02-19	SW
2	Annie Hall	1977	93	1977-04-20	USA
3	Seven Samurai	1954	207	1954-04-26	JP
4	Strange Behaviou...	1994	121	1979-08-13	RU
5	Doulos, Le	1984	91	1958-11-27	ID

Total rows: 77 of 77

2) Write a SQL query to find out which actors have not appeared in any movies between 1967 and 1989 (Begin and end values are included.). Return actor first name, last name, movie title and release year.

(*Changed Question. Reason:* *In order to cover the topics mentioned in the part-2 document which I wasn't aware while working for part-1, I've changed the question to make a query with joining more than two tables.*)

ANS:

```sql
SELECT act_fname, act_lname, mov_title, mov_year

FROM actor

JOIN movie_cast

ON actor.act_id=movie_cast.act_id

JOIN movie

ON movie_cast.mov_id=movie.mov_id

WHERE mov_year NOT BETWEEN 1967 and 1989;
```

```sql
SELECT act_fname, act_lname, mov_title, mov_year
FROM actor
JOIN movie_cast
ON actor.act_id=movie_cast.act_id
JOIN movie
ON movie_cast.mov_id=movie.mov_id
WHERE mov_year NOT BETWEEN 1967 and 1989;
```

Data Output

	act_fname character varying	act_lname character varying	mov_title character varying	mov_year integer
1	James	Stewart	Vertigo	1958
2	Deborah	Kerr	The Innocents	1961
3	Peter	OToole	Lawrence of Arabia	1962
4	Jack	Nicholson	Eyes Wide Shut	1999
5	Mark	Wahlberg	The Usual Suspects	1995

Total rows: 85 of 85

3) Create a query to show reviewers who have rated more than 6 starts.

ANS:

SELECT reviewer.rev_name

FROM reviewer, rating

WHERE rating.rev_id = reviewer.rev_id

AND rating.rev_stars>=6

AND reviewer.rev_name IS NOT NULL;

Query Query History

```sql
SELECT reviewer.rev_name
FROM reviewer, rating
WHERE rating.rev_id = reviewer.rev_id
AND rating.rev_stars>=6
AND reviewer.rev_name IS NOT NULL;
```

Data Output

	rev_name character varying
1	Righty Sock
2	Jack Malvern
3	Flagrant Barones...
4	Alec Shaw
5	Jane Fusedale

Total rows: 63 of 63

4) Create a query to show the movie names which got more than 9 stars as rating.

Ans:

SELECT mov_title

FROM movie m JOIN rating r

ON m.mov_id = r.mov_id

WHERE r.rev_stars > 9;

Query Query History

```
SELECT mov_title
FROM movie m JOIN rating r
ON m.mov_id = r.mov_id
WHERE r.rev_stars > 9;
```

Data Output

	mov_title character varying
1	Thief of Damascus
2	Homesman, The
3	High and Dry
4	Donovan's Brain
5	Vamps

Total rows: 8 of 8

5) Create a query to show the movie name(s) which is in 'Japanese' language and released in the year 1997.

Ans:

SELECT mov_title

FROM movie

WHERE mov_year=1997 AND mov_lang = 'Japanese';

```
SELECT mov_title
FROM movie
WHERE mov_year=1997 AND mov_lang = 'Japanese';
```

Data Output

	mov_title character varying
1	Princess Monono...

Total rows: 1 of 1

6) Create a query to determine the year(s) in which there was at least one movie that go a rating of at least three stars. Sort them in descending order.

Ans:

SELECT DISTINCT mov_year

FROM movie

WHERE mov_id IN (

SELECT mov_id

FROM rating

WHERE rev_stars>3)

ORDER BY mov_year DESC;

Query Query History

```
SELECT DISTINCT mov_year
FROM movie
WHERE mov_id IN (
SELECT mov_id
FROM rating
WHERE rev_stars>3)
ORDER BY mov_year DESC;
```

Data Output

	mov_year integer
1	2013
2	2012
3	2011
4	2010
5	2009

Total rows: 36 of 36

7) Write a SQL query to find the actor whose first name is 'Tim' and last name is 'Robbins'. Return director first name, last name, movie title.

ANS:

SELECT dir_fname, dir_lname, mov_title

FROM actor

JOIN movie_cast

ON actor.act_id=movie_cast.act_id

JOIN movie_direction

ON movie_cast.mov_id=movie_direction.mov_id

JOIN director

ON movie_direction.dir_id=director.dir_id

JOIN movie

ON movie.mov_id=movie_direction.mov_id

WHERE act_fname='Tim'

AND act_lname='Robbins';

Query Query History

```
SELECT dir_fname, dir_lname, mov_title
FROM actor
JOIN movie_cast
  ON actor.act_id=movie_cast.act_id
JOIN movie_direction
  ON movie_cast.mov_id=movie_direction.mov_id
JOIN director
  ON movie_direction.dir_id=director.dir_id
JOIN movie
  ON movie.mov_id=movie_direction.mov_id
WHERE act_fname='Tim'
  AND act_lname='Robbins';
```

Data Output

	dir_fname character varying	dir_lname character varying	mov_title character varying
1	Woody	Allen	Annie Hall

Total rows: 1 of 1

8) Write a query to show the details of an actor whose first name starts with 'D' and last name starts with 'K'.

(*Changed Question. Reason:* *Modified the question to make a query that includes LIKE condition as mentioned in the document.*)

ANS:

SELECT * FROM actor

WHERE act_fname LIKE 'D%' AND act_lname LIKE 'K%';

```
SELECT * FROM actor
WHERE act_fname LIKE 'D%' AND act_lname LIKE 'K%';
```

Data Output

	act_id [PK] integer	act_fname character varying	act_lname character varying	act_gender character varying
1	102	Deborah	Kerr	F

Total rows: 1 of 1

9) Create a query to show the movie name, released year of the movie which got the lowest rating starts.

ANS:

SELECT mov_title, mov_year, rev_stars, mov_rel_country

FROM movie

NATURAL JOIN rating

WHERE rev_stars = (

SELECT MIN(rev_stars)

FROM rating

);

```
SELECT mov_title, mov_year, rev_stars, mov_rel_country
FROM movie
NATURAL JOIN rating
WHERE rev_stars = (
SELECT MIN(rev_stars)
FROM rating
);
```

Data Output

	mov_title character varying	mov_year integer	rev_stars double precision	mov_rel_country character varying
1	Strange Behaviou...	1994	1.3	RU

Total rows: 1 of 1

10) Create a query to find the years when most of the Action Movies made. Count the number of generic title and compute their average rating. Group the result set on movie release year, generic title. Return movie year, generic title, number of generic title and average rating.

ANS:

SELECT mov_year,gen_title,count(gen_title), avg(rev_stars)
FROM movie
NATURAL JOIN movie_genre
NATURAL JOIN genre
NATURAL JOIN rating
WHERE gen_title='Mystery'
GROUP BY mov_year,gen_title;

```
SELECT mov_year,gen_title,count(gen_title), avg(rev_stars)
FROM movie
NATURAL JOIN movie_genre
NATURAL JOIN genre
NATURAL JOIN rating
WHERE gen_title='Mystery'
GROUP BY mov_year,gen_title;
```

Data Output

	mov_year integer	gen_title character varying	count bigint	avg double precision
1	1997	Mystery	1	8.4

Total rows: 1 of 1