Nirnoy Ghosh

imdb.jpg

Analysis on IMDB useful libraries with sql

Explanation

SQLite : - SQLite database is integrated with the application that accesses the database. The applications interact with the SQLite database read and write directly from the database files stored on disk.

- SQLite does NOT require a server to run.
- Refer at https://www.sqlite.org/index.html

sqlite_image.jpg

PANDAS : - Pandas provide high performance, fast, easy to use data structures and data analysis tools for manipulating numeric data and time series. Pandas is built on the numpy library and written in languages like Python, Cython, and C. In pandas, we can import data from various file formats like JSON, SQL, Microsoft Excel, etc.

NUMPY: - Numpy is an Python library,It is used for scientific computing in python.It contains a collection of tools and techniques that can be used to solve on computers with mathematical model of problems. - High performance and multi-dimensional areay objects. - High level mathematical functions. - matrices

Data Visualizations

- Data Visualization is the graphic representation of data. It converts a huge dataset into small graphs, thus aids in data analysis and predictions

MATPLOTLIB

- It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python. It is used for creating statical interferences and plotting 2D graphs of arrays

SEABORN:

- It is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy. It is built on the roof of Matplotlib and is considered as a superset of the Matplotlib library. It helps in visualizing univariate and bivariate data

```
# Import Pandas and NumPy

import numpy as np
import pandas as pd

# Import Libraries for plotting    # Visualizations
```

```
import matplotlib.pyplot as plt
import seaborn as sns



#pip install db-sqlite3
#SQLite in general is a server-less database that you can use within
almost all programming languages including Python.
import sqlite3


# Supress Warnings
```

We import the sqlite3 module and then create a connection object which will connect us to the database and will let us execute the SQL statements.

- sqlite3.connect(Database)
- **cursor()** - we can use the cursor object to call the execute() method to execute any SQL queries.

STEPS

To query data in an SQLite database from Python, you use these steps:

- First, establish a connection to the SQLite database by creating a Connection object.
- Next, create a Cursor object using the cursor method of the Connection object.
- Then, execute a SELECT statement.
- After that, call the fetchall() method of the cursor object to fetch the data.
- Finally, loop the cursor and process each row individually.

```
mycon = sqlite3.connect(r"C:\Users\ASUS\Downloads\movies.sqlite")
cur = mycon.cursor()
```

QUESTIONS

Question 1: Can you get all data about movies?

```
cur.execute('SELECT * FROM movies')
#use the cursor object to call the execute() method to execute any SQL
queries.
movies = cur.fetchall()

# #movies -> if u print movies , all the data shown in list manner,
can't understanadble

# #Note - fetchall() --> fetch all rows for the current query, if no
rows avaliable then,it returns an empty list.

movies = pd.DataFrame(movies, columns = ['id', 'original_title',
'budget', 'popularity', 'release_date', 'revenue', 'title',
'vote_average', 'vote_count', 'overview', 'tagline', 'uid',
```

```
'director_id'])
# #Creating a dataframe, A dataframe contains rows and columns , It is a Two-dimensional, size-mutable, potentially heterogeneous tabular data.



print('Shape of Movies data :',movies.shape)
print('\n\n\n')
movies.head(3)
# #Displays first 3 rows in the table.
```

```
Shape of Movies data : (4773, 13)
```

```
      id                                original_title     budget
popularity  \
0  43597                                     Avatar  237000000
150
1  43598  Pirates of the Caribbean: At World's End  300000000
139
2  43599                                    Spectre  245000000
107

  release_date     revenue
title  \
0   2009-12-10  2787965087                                       Avatar

1   2007-05-19   961000000  Pirates of the Caribbean: At World's End

2   2015-10-26   880674609                                     Spectre


   vote_average  vote_count  \
0           7.2       11800
1           6.9        4500
2           6.3        4466

                                            overview  \
0  In the 22nd century, a paraplegic Marine is di...
1  Captain Barbossa, long believed to be dead, ha...
2  A cryptic message from Bond’s past sends him o...

                                        tagline     uid  director_id

0                    Enter the World of Pandora.   19995         4762

1  At the end of the world, the adventure begins.     285         4763
```

```
2                           A Plan No One Escapes  206647         4764

mycon = sqlite3.connect(r"C:\Users\ASUS\Downloads\movies.sqlite")
import pandas as pd
movies_df = pd.read_sql('SELECT * FROM movies', mycon)
movies_df.head()

      id                                original_title     budget
popularity  \
0  43597                                     Avatar  237000000
150
1  43598  Pirates of the Caribbean: At World's End  300000000
139
2  43599                                    Spectre  245000000
107
3  43600                     The Dark Knight Rises  250000000
112
4  43601                                  John Carter  260000000
43

  release_date     revenue
title  \
0   2009-12-10  2787965087                                     Avatar

1   2007-05-19   961000000  Pirates of the Caribbean: At World's End

2   2015-10-26   880674609                                    Spectre

3   2012-07-16  1084939099                       The Dark Knight Rises

4   2012-03-07   284139100                                  John Carter


   vote_average  vote_count  \
0           7.2       11800
1           6.9        4500
2           6.3        4466
3           7.6        9106
4           6.1        2124

                                          overview  \
0  In the 22nd century, a paraplegic Marine is di...
1  Captain Barbossa, long believed to be dead, ha...
2  A cryptic message from Bond’s past sends him o...
3  Following the death of District Attorney Harve...
4  John Carter is a war-weary, former military ca...

                                       tagline     uid  director_id

0                    Enter the World of Pandora.   19995         4762
```

```
1  At the end of the world, the adventure begins.     285         4763

2                           A Plan No One Escapes  206647         4764

3                                     The Legend Ends   49026         4765

4            Lost in our world, found in another.   49529         4766
```

```
movies_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4773 entries, 0 to 4772
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   id              4773 non-null   int64  
 1   original_title  4773 non-null   object 
 2   budget          4773 non-null   int64  
 3   popularity      4773 non-null   int64  
 4   release_date    4773 non-null   object 
 5   revenue         4773 non-null   int64  
 6   title           4773 non-null   object 
 7   vote_average    4773 non-null   float64
 8   vote_count      4773 non-null   int64  
 9   overview        4770 non-null   object 
 10  tagline         3951 non-null   object 
 11  uid             4773 non-null   int64  
 12  director_id     4773 non-null   int64  
dtypes: float64(1), int64(7), object(5)
memory usage: 484.9+ KB
```

```
movies.isnull().sum()
```

```
id                  0
original_title      0
budget              0
popularity          0
release_date        0
revenue             0
title               0
vote_average        0
vote_count          0
overview            3
tagline           822
uid                 0
director_id         0
dtype: int64
```

Question 2: How do you get all data about directors?

```
query = 'SELECT * FROM directors'
directors_df = pd.read_sql(query, mycon)
directors_df
```

```
                    name    id  gender      uid department
0          James Cameron  4762       2     2710  Directing
1         Gore Verbinski  4763       2     1704  Directing
2             Sam Mendes  4764       2       39  Directing
3     Christopher Nolan  4765       2      525  Directing
4        Andrew Stanton  4766       2        7  Directing
...                 ...   ...     ...      ...        ...
2344       Shane Carruth  7106       2    76624  Directing
2345   Neill Dela Llana  7107       0  1174437  Directing
2346         Scott Smith  7108       0  1219158  Directing
2347         Daniel Hsia  7109       2   208138  Directing
2348   Brian Herzlinger  7110       2    85563  Directing

[2349 rows x 5 columns]
```

```
directors_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2349 entries, 0 to 2348
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   name        2349 non-null   object
 1   id          2349 non-null   int64 
 2   gender      2349 non-null   int64 
 3   uid         2349 non-null   int64 
 4   department  2349 non-null   object
dtypes: int64(3), object(2)
memory usage: 91.9+ KB
```

Question 3: Check how many movies are present in iMDB.

```
cur.execute('SELECT COUNT (id) from movies;')
#use the cursor object to call the execute() method to execute any SQL
queries.
count = cur.fetchall()
print("the number of movies is present in IMDB is  :",count[0])
#Note - fetchall() --> fetch all rows for the current query, if no
rows avaliable then,it returns an empty list.
```

```
the number of movies is present in IMDB is  : (4773,)
```

```
##ASSIGNMENT

# dataframes to be used for pandas - movies_df, directors_df
# for sql table names are movies and directors
```

Question 4: Find these 3 directors: James Cameron ; Luc Besson ; John Woo

```
query = 'SELECT * FROM directors where name in ("James Cameron","Luc
Besson" , "John Woo")'
directors_df1 = pd.read_sql(query, mycon)
directors_df1
```

	name	id	gender	uid	department
0	James Cameron	4762	2	2710	Directing
1	John Woo	4893	2	11401	Directing
2	Luc Besson	4949	2	59	Directing

Question 5: Find all directors with name starting with Steven.

```
query = 'SELECT * FROM directors where name like "Steven%"'
directors_Steven = pd.read_sql(query, mycon)
directors_Steven
```

	name	id	gender	uid	department
0	Steven Spielberg	4799	2	488	Directing
1	Steven Soderbergh	4909	2	1884	Directing
2	Steven Brill	5013	2	32593	Directing
3	Steven Zaillian	5117	2	2260	Directing
4	Steven Quale	5216	2	93214	Directing
5	Steven Seagal	5221	2	23880	Directing
6	Steven E. de Souza	5390	2	1726	Directing
7	Steven Shainberg	5803	2	67795	Directing
8	Steven R. Monroe	6713	2	88039	Directing

Question 6: Count female directors.

```
query = 'SELECT * FROM directors where gender=1'
directors_fem = pd.read_sql(query, mycon)
directors_fem
```

	name	id	gender	uid	department
0	Brenda Chapman	4801	1	59803	Directing
1	Lilly Wachowski	4805	1	9339	Directing
2	Jennifer Yuh Nelson	4853	1	142312	Directing
3	Kathryn Bigelow	4970	1	14392	Directing
4	Nancy Meyers	4978	1	17698	Directing
..	...	...	...	...	...
145	Marianna Palka	7027	1	77775	Directing
146	Ricki Stern	7028	1	74105	Directing
147	Lynn Shelton	7049	1	90492	Directing
148	Julie Davis	7066	1	87033	Directing
149	Lena Dunham	7075	1	139135	Directing

```
[150 rows x 5 columns]
```

Question 7: Find the name of the 10th first women directors?

```
query = 'SELECT name FROM directors WHERE gender=1 ORDER BY ID LIMIT 1
OFFSET 10 '
directors_10_fem = pd.read_sql(query, mycon)
print("10th first women director is ",directors_10_fem["name"][0])
```

```
10th first women director is  Karyn Kusama
```

```
#Alternative
query = 'SELECT name FROM directors WHERE gender=1 ORDER BY ID LIMIT
10 '
directors_10_fe = pd.read_sql(query, mycon)
directors_10_fe
```

```
                  name
0       Brenda Chapman
1      Lilly Wachowski
2  Jennifer Yuh Nelson
3      Kathryn Bigelow
4         Nancy Meyers
5          Jill Culton
6           Mimi Leder
7         Vicky Jenson
8         Betty Thomas
9       Angelina Jolie
```

Question 8: What are the 3 most popular movies?

```
query = 'SELECT popularity,original_title,revenue,vote_count FROM
movies ORDER BY popularity DESC LIMIT 3'
directors_fem = pd.read_sql(query, mycon)
print("Three most popular movies are:")
directors_fem
```

```
Three most popular movies are:
```

```
   popularity original_title     revenue  vote_count
0         875        Minions  1156730962        4571
1         724   Interstellar   675120017       10867
2         514       Deadpool   783112979       10995
```

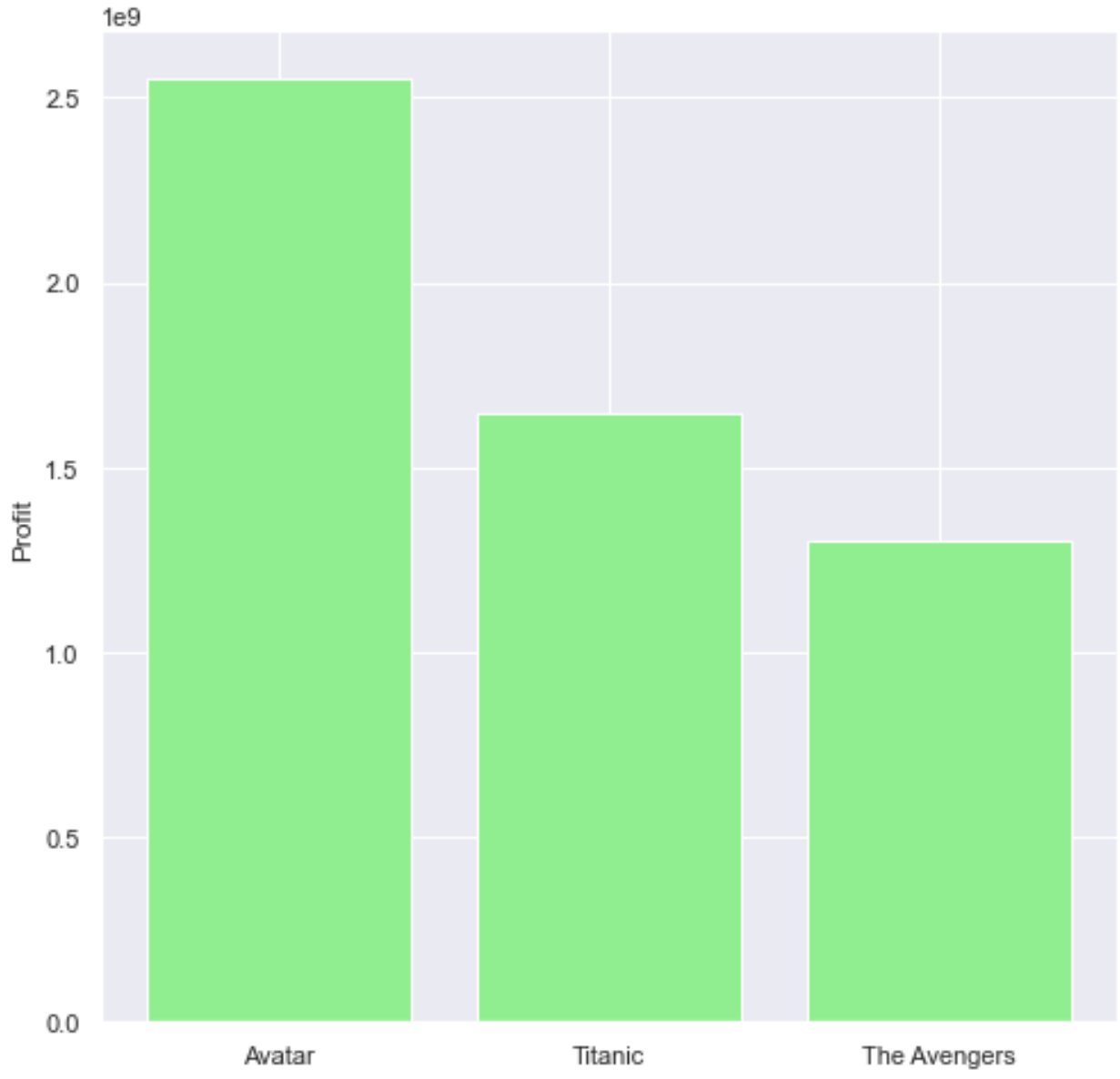
Question 9: What are the 3 most bankable movies?

```
query = """SELECT original_title,(revenue-budget) as
profit,popularity,vote_count
FROM movies
ORDER BY revenue DESC LIMIT 3"""
bankable_movies = pd.read_sql(query, mycon)
#print("Three most bankable movies are:")
print('\033[1m' +"Three most bankable movies are:" + '\033[0m')
plt.bar(bankable_movies["original_title"],bankable_movies["profit"],co
```

```
lor="lightgreen")
plt.ylabel("Profit")
#plt.xticks(rotation="90")
bankable_movies
```

```
Three most bankable movies are:
```

	original_title	profit	popularity	vote_count
0	Avatar	2550965087	150	11800
1	Titanic	1645034188	100	7562
2	The Avengers	1299557910	144	11776




```
movies
```

```
         id                                original_title     budget
popularity  \
0     43597                                        Avatar  237000000
150
1     43598  Pirates of the Caribbean: At World's End  300000000
139
2     43599                                       Spectre  245000000
107
3     43600                      The Dark Knight Rises  250000000
112
4     43601                                   John Carter  260000000
43
...     ...                                       ...        ...
...
4768  48395                                   El Mariachi     220000
14
4769  48396                                     Newlyweds       9000
0
4770  48397                   Signed, Sealed, Delivered          0
1
4771  48398                            Shanghai Calling          0
0
4772  48399                           My Date with Drew          0
1

     release_date     revenue
title  \
0      2009-12-10  2787965087
Avatar
1      2007-05-19   961000000  Pirates of the Caribbean: At World's
End
2      2015-10-26   880674609
Spectre
3      2012-07-16  1084939099                    The Dark Knight
Rises
4      2012-03-07   284139100                               John
Carter
...           ...         ...                                  ..
.
4768   1992-09-04     2040920                               El
Mariachi
4769   2011-12-26           0
Newlyweds
4770   2013-10-13           0                 Signed, Sealed,
Delivered
4771   2012-05-03           0                          Shanghai
Calling
4772   2005-08-05           0                        My Date with
Drew
```

```
      vote_average  vote_count  \
0              7.2       11800
1              6.9        4500
2              6.3        4466
3              7.6        9106
4              6.1        2124
...            ...         ...
4768           6.6         238
4769           5.9           5
4770           7.0           6
4771           5.7           7
4772           6.3          16

                                                  overview  \
0     In the 22nd century, a paraplegic Marine is di...
1     Captain Barbossa, long believed to be dead, ha...
2     A cryptic message from Bond’s past sends him o...
3     Following the death of District Attorney Harve...
4     John Carter is a war-weary, former military ca...
...                                                 ...
4768  El Mariachi just wants to play his guitar and ...
4769  A newlywed couple's honeymoon is upended by th...
4770  "Signed, Sealed, Delivered" introduces a dedic...
4771  When ambitious New York attorney Sam is sent t...
4772  Ever since the second grade when he first saw ...

                                              tagline     uid
director_id
0                           Enter the World of Pandora.   19995
4762
1        At the end of the world, the adventure begins.     285
4763
2                                 A Plan No One Escapes  206647
4764
3                                      The Legend Ends   49026
4765
4                  Lost in our world, found in another.   49529
4766
...                                                ...     ...
...
4768  He didn't come looking for trouble, but troubl...    9367
5097
4769  A newlywed couple's honeymoon is upended by th...   72766
6485
4770                                               None  231617
7108
4771                           A New Yorker in Shanghai  126186
7109
4772                                               None   25975
```

```
7110

[4773 rows x 13 columns]
```

Question 10: What is the most awarded average vote since the January 1st, 2000?

```
query = """SELECT original_title,max(vote_average) as vote_avg
FROM movies
WHERE release_date>=2000-01-01
"""
most_award_movies = pd.read_sql(query, mycon)
most_award_movies

# mostly occuerd vote average.
```

```
     original_title  vote_avg
0  Stiff Upper Lips      10.0
```

```
#Alternative
query = """SELECT original_title,vote_average
FROM movies WHERE release_date>='2000-01-01' ORDER BY vote_average
DESC LIMIT 1
"""
awarded_movies = pd.read_sql(query, mycon)
awarded_movies
```

```
  original_title  vote_average
0      Sardaarji           9.5
```

JOINS

Inner join left join right join full outer join

- Explanation

Inner join : - Wherever the data matching between the two tables only thouse records will displayed . (Only matched in both tables)

Left join : - All the data which is present on the left side will displayed, irrespective it having matching entry will displays or if it does not match entry on another table

Right join : - (vice-versa) as Left join From right table all records will displayas, irrespective But in left table who are matching are be displayed

full outer join : - It is a combination of Inner join , Left join, Right join. **Full outer join** shows all matching records,

- It will displays all matching records as well as It will display non matching records On left table and right table.
- It will show every thing on what tables contains

NOTE

- Left join display all records in left table

- Right join display all records in right table
- Inner join display only matching records from tables

directors_df.rename(columns = {'id':'director_id'}, inplace = True)
m_tab=directors_df.merge(movies, how="inner", on="director_id").head()

Question 11: Which movie(s) were directed by Brenda Chapman?

```
query = """
SELECT movies.original_title, directors.name
FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
WHERE name='Brenda Chapman'
"""
awarded_movies = pd.read_sql(query, mycon)
awarded_movies
```

```
  original_title            name
0          Brave  Brenda Chapman
```

Question 12: Whose director made the most movies?

```
query = """
SELECT name,count(name) as no_of_movies
FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
GROUP BY director_id
ORDER BY count(name) DESC limit 1
"""
awarded_movies = pd.read_sql(query, mycon)
print(awarded_movies["name"][0],"made",awarded_movies["no_of_movies"]
[0],"movies and it is the highest no of movies made by a director till
now")
awarded_movies
```

```
Steven Spielberg made 27 movies and it is the highest no of movies
made by a director till now
```

```
               name  no_of_movies
0  Steven Spielberg            27
```

Question 13: Whose director is the most bankable?

```
query = """
SELECT name, revenue,vote_count
FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
GROUP BY revenue
```

```
ORDER BY revenue DESC limit 1
"""
revenue_movies = pd.read_sql(query, mycon)
print(revenue_movies)
print("So, the most bankable director is:",'\033[1m'
+revenue_movies["name"][0]+'\033[0m',"\n\n")
```

```
            name     revenue  vote_count
0  James Cameron  2787965087       11800
So, the most bankable director is: James Cameron
```

```
query = """
SELECT name, revenue,vote_count
FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
GROUP BY revenue
ORDER BY revenue DESC
"""
print('\033[1m' +"Bankable directors List:"+'\033[0m')
revenue_movies = pd.read_sql(query, mycon)
revenue_movies
```

```
Bankable directors List:
```

	name	revenue	vote_count
0	James Cameron	2787965087	11800
1	James Cameron	1845034188	7562
2	Joss Whedon	1519557910	11776
3	Colin Trevorrow	1513528810	8662
4	James Wan	1506249360	4176
...	...	...	...
3290	Norman Jewison	11	29
3291	Louis Morneau	10	35
3292	Rachel Perkins	7	6
3293	Tony Maylam	5	63
3294	Roland Joffé	0	34

```
[3295 rows x 3 columns]
```

Analysis on IMDB DATA SET

Budget Analysis

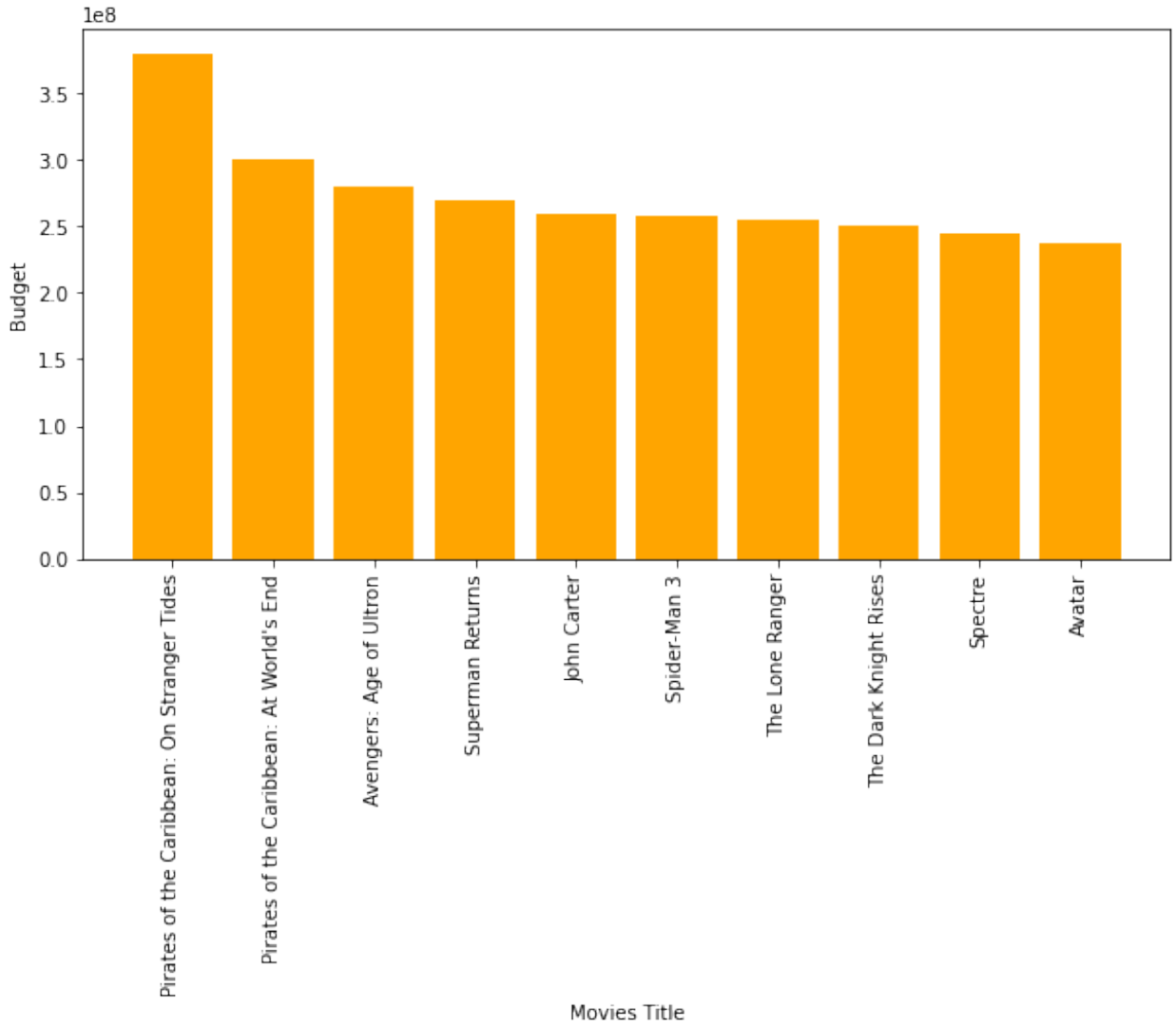
Question - List out Top 10 highest budget made movies

```
query = """
SELECT directors.name,movies.original_title,movies.budget
FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
GROUP BY budget
ORDER BY budget DESC limit 10
"""
budget_movies = pd.read_sql(query, mycon)
print("Top 10 highest budget made movies::")
budget_movies
```

```
Top 10 highest budget made movies::

                name                                  original_title
budget
0       Rob Marshall  Pirates of the Caribbean: On Stranger Tides
380000000
1     Gore Verbinski     Pirates of the Caribbean: At World's End
300000000
2        Joss Whedon                       Avengers: Age of Ultron
280000000
3       Bryan Singer                               Superman Returns
270000000
4     Andrew Stanton                                    John Carter
260000000
5          Sam Raimi                                   Spider-Man 3
258000000
6     Gore Verbinski                                 The Lone Ranger
255000000
7  Christopher Nolan                          The Dark Knight Rises
250000000
8         Sam Mendes                                       Spectre
245000000
9      James Cameron                                        Avatar
237000000
```

```
fig = plt.figure(figsize = (10, 5))
plt.bar(budget_movies["original_title"] ,budget_movies["budget"],color
='orange')
plt.xticks(rotation=90)
plt.xlabel("Movies Title")
plt.ylabel("Budget")
plt.show()
```



Note :

The most Expensive Table Contains List Of Movies With Highest Production Costs.

For Your Information, **Pirates of the Caribbean: On Stranger Tides**, Directed By **Rob Marshall**,is the Most Production Costs, It's **USD 380,000,000.**

Popularity Analysis

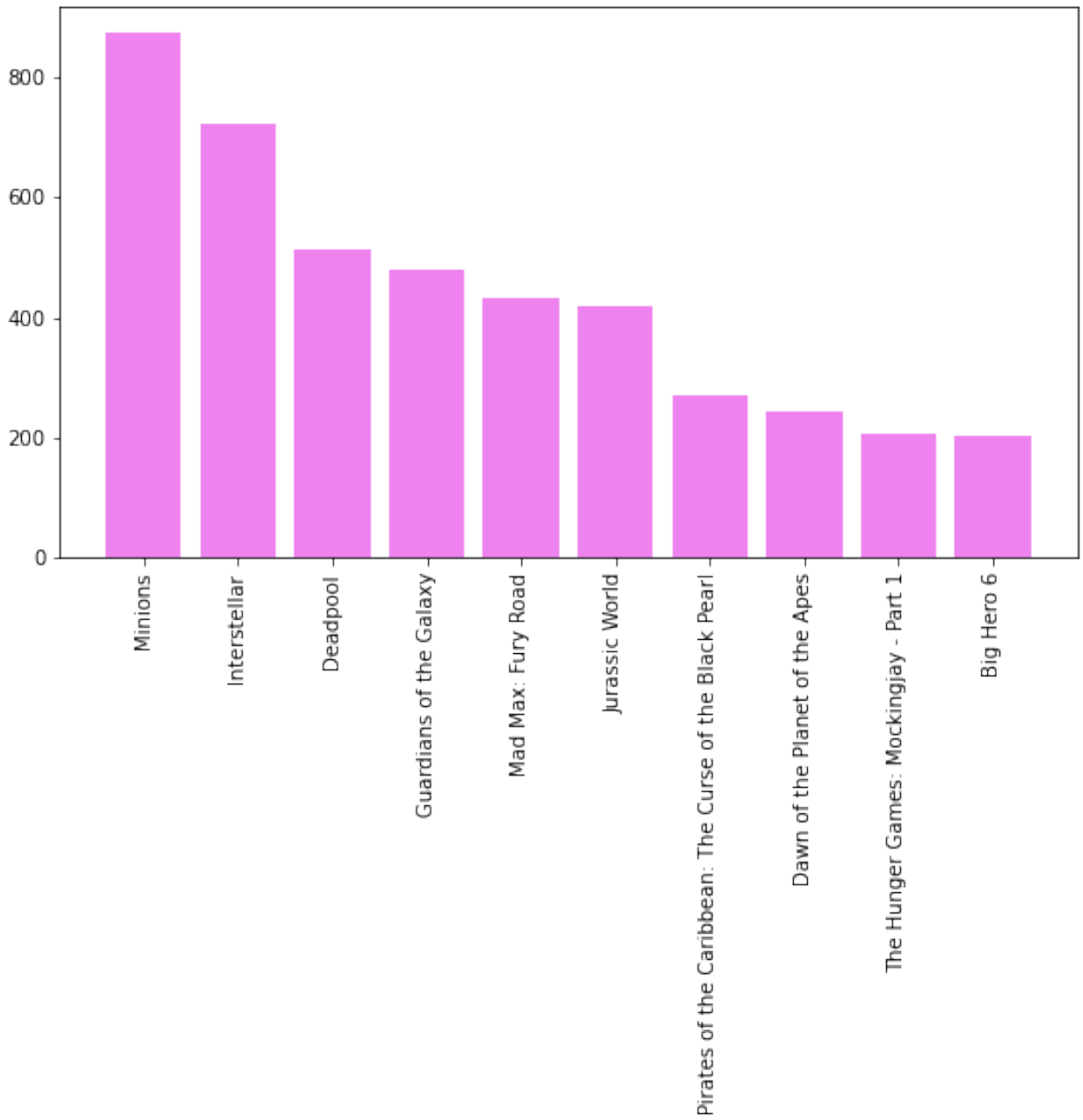
Question- List out Top 10 popularity on movies

```
query = """
SELECT
movies.popularity,directors.name,movies.original_title,movies.budget
FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
```

```
GROUP BY popularity
ORDER BY movies.popularity DESC limit 10
"""
popular_movies = pd.read_sql(query, mycon)
print("Top 10 highest popular movies::")

fig = plt.figure(figsize = (9, 5))
plt.bar(popular_movies["original_title"] ,popular_movies["popularity"]
,color ='violet')
plt.xticks(rotation=90)
plt.show()
popular_movies
```

```
Top 10 highest popular movies::
```



	popularity	name
0	875	Kyle Balda
1	724	Christopher Nolan
2	514	Tim Miller
3	481	James Gunn
4	434	George Miller
5	418	Colin Trevorrow
6	271	Gore Verbinski
7	243	Matt Reeves
8	206	Francis Lawrence
9	203	Chris Williams

```
                                       original_title     budget
0                                             Minions   74000000
1                                        Interstellar  165000000
2                                            Deadpool   58000000
3                                Guardians of the Galaxy  170000000
4                                     Mad Max: Fury Road  150000000
5                                        Jurassic World  150000000
6  Pirates of the Caribbean: The Curse of the Bla...  140000000
7                        Dawn of the Planet of the Apes  170000000
8                The Hunger Games: Mockingjay - Part 1  125000000
9                                          Big Hero 6  165000000
```

Note:

Most Popular Table Shows 10 Movies With The Highest Ranking By Popularity.

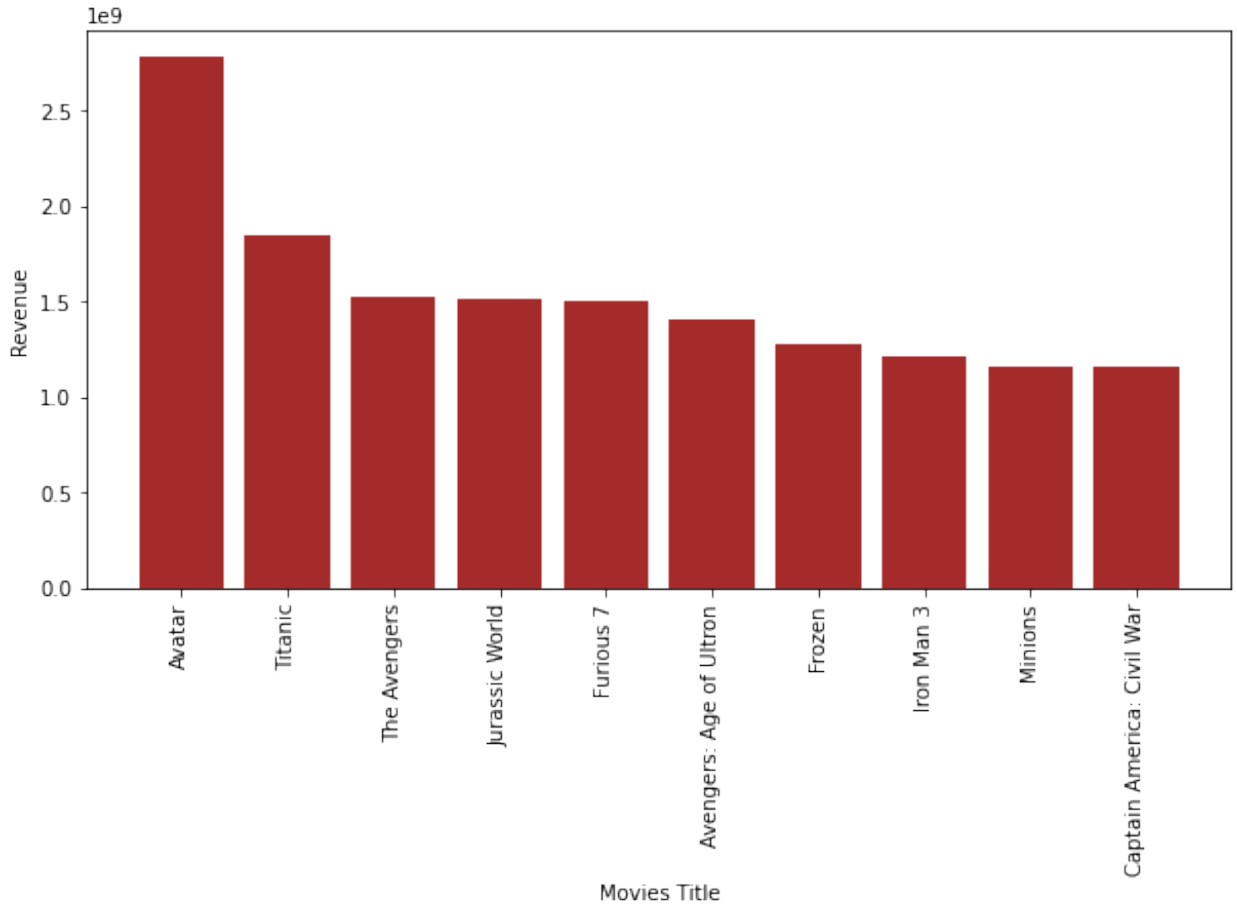
Currently, **Minions Directed By Kyle Balda** Got The First Place With Score 875, Beating Interstellar As The Closest Competitor, Deadpool And The Other Competitors.

Revenu Analysis

Question - What are the top 10 Revenue movies

```
query = """
SELECT original_title, revenue,vote_count,name as Director
FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
GROUP BY revenue
ORDER BY revenue DESC limit 10
"""
print('\033[1m' +" Top 10 revenue movies:"+'\033[0m')
revenue_movies = pd.read_sql(query, mycon)
fig = plt.figure(figsize = (10, 5))
plt.bar(revenue_movies["original_title"] ,revenue_movies["revenue"],co
lor ='brown')
plt.xticks(rotation=90)
plt.xlabel("Movies Title")
plt.ylabel("Revenue")
plt.show()
revenue_movies
```

```
 Top 10 revenue movies:
```



	original_title	revenue	vote_count	Director
0	Avatar	2787965087	11800	James Cameron
1	Titanic	1845034188	7562	James Cameron
2	The Avengers	1519557910	11776	Joss Whedon
3	Jurassic World	1513528810	8662	Colin Trevorrow
4	Furious 7	1506249360	4176	James Wan
5	Avengers: Age of Ultron	1405403694	6767	Joss Whedon
6	Frozen	1274219009	5295	Chris Buck
7	Iron Man 3	1215439994	8806	Shane Black
8	Minions	1156730962	4571	Kyle Balda
9	Captain America: Civil War	1153304495	7241	Anthony Russo

Note :

The mostProfit Table Shows The 10 Movies Got The Highest Gross Profit When Compared To The Other Movies.

It Turns Out That **Avatar** Directed By **James Cameron** Got A Gross Profit **USD 2,787,965,087**. It's Make Avatar In **First Place**, Followed By Titanic, The Avengers, Jurassic World, And Others.

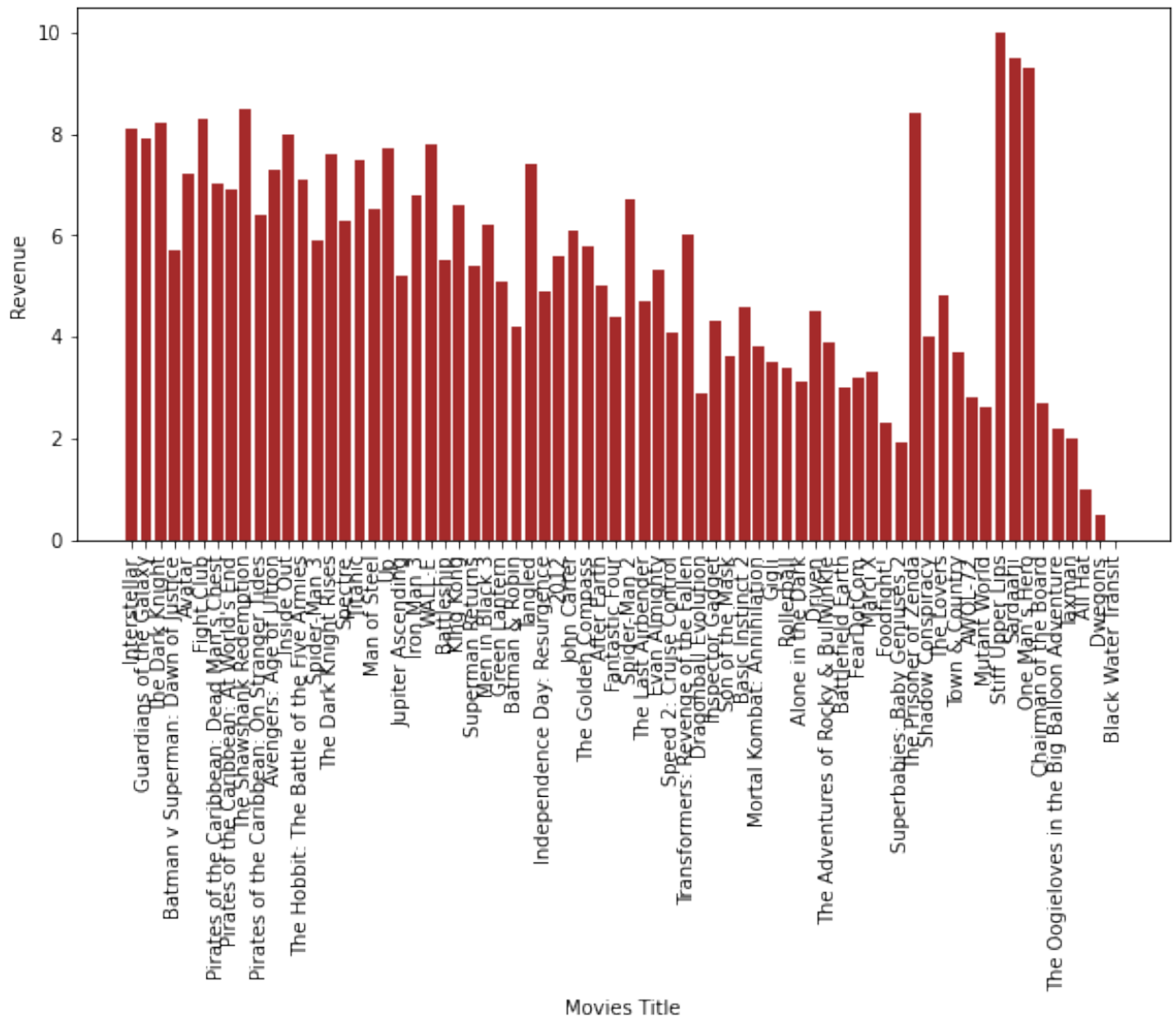
From The Table, It Can Be Seen That James Cameron Make Two Movies To Fill The First And Second Place On The Rankings.

Question - Display most popular movies with vote_average

```
query = """
SELECT original_title as Movie_Name,popularity,vote_average,name as
Director
FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
GROUP BY vote_average
ORDER BY popularity DESC
"""
print('\033[1m' +" Most popular movies:"+'\033[0m')
popular_movies = pd.read_sql(query, mycon)

fig = plt.figure(figsize = (10, 5))
plt.bar(popular_movies["Movie_Name"] ,popular_movies["vote_average"],c
olor ='brown')
plt.xticks(rotation=90)
plt.xlabel("Movies Title")
plt.ylabel("Revenue")
plt.show()
popular_movies
```

```
 Most popular movies:
```



	Movie_Name	popularity	vote_average
0	Interstellar	724	8.1
1	Guardians of the Galaxy	481	7.9
2	The Dark Knight	187	8.2
3	Batman v Superman: Dawn of Justice	155	5.7
4	Avatar	150	7.2
..	...	...	...
65	The Oogieloves in the Big Balloon Adventure	0	2.2
66	Taxman	0	

```
2.0
67                                        All Hat           0
1.0
68                                        Dwegons           0
0.5
69                             Black Water Transit           0
0.0

              Director
0   Christopher Nolan
1          James Gunn
2   Christopher Nolan
3         Zack Snyder
4        James Cameron
..                ...
65    Matthew Diamond
66          Avi Nesher
67  Leonard Farlinger
68           Tom Walsh
69           Tony Kaye

[70 rows x 4 columns]
```

popular_movies

```
                                       Movie_Name  popularity
vote_average  \
0                                     Interstellar         724
8.1
1                           Guardians of the Galaxy         481
7.9
2                                  The Dark Knight         187
8.2
3              Batman v Superman: Dawn of Justice         155
5.7
4                                           Avatar         150
7.2
..                                             ...         ...
...
65  The Oogieloves in the Big Balloon Adventure           0
2.2
66                                          Taxman           0
2.0
67                                        All Hat           0
1.0
68                                        Dwegons           0
0.5
69                             Black Water Transit           0
0.0
```

```
             Director
0   Christopher Nolan
1          James Gunn
2   Christopher Nolan
3         Zack Snyder
4       James Cameron
..                ...
65    Matthew Diamond
66          Avi Nesher
67  Leonard Farlinger
68           Tom Walsh
69           Tony Kaye

[70 rows x 4 columns]
```

Note :

From the mostProfit Table, It Can Be Seen That The Movies In The First And Second Rank Is A Movies Directed By James Cameron.

In The Table moviesByJamesCameron Shows All Movies Directed By James Cameron.

Voting Analysis

Question - List out Particular movie for Voting average and votecount

```
query = """
SELECT original_title ,vote_average,vote_count
FROM movies
LEFT JOIN directors
ON movies.director_id=directors.id
ORDER by vote_average DESC
"""
vote = pd.read_sql(query, mycon)
vote
```

```
                          original_title  vote_average  vote_count
0                     Stiff Upper Lips          10.0           1
1                 Dancer, Texas Pop. 81          10.0           1
2                              Sardaarji           9.5           2
3                       One Man's Hero            9.3           2
4              The Shawshank Redemption           8.5        8205
...                                 ...           ...         ...
4768            The Legend of God's Gun           0.0           0
4769  Her Cry: La Llorona Investigation           0.0           0
4770                         Dutch Kills           0.0           0
4771               Stories of Our Lives           0.0           0
4772       Sanctuary: Quite a Conundrum           0.0           0
```

```
[4773 rows x 3 columns]
```

Note :

BestVote Table Contains The Best Movies By Vote. Stiff Upper Lips, Directed By Gary Sinyor And Dancer, Texas Pop. 81 Directed by the MacCanlies Was Successfully Becomes the Best Movies By Vote With Perfect Score, 10.

However, There Are Anomalies. It Turned Out That The Two Movies Were Only Vote By One Person. So, Likely to Affect The Assessment Results..

CORRELATION

- **Correlation** is also a statistical technique that determines how the change of one variable related to another variable affects the relationship.
- it defines the degree of relation between two variables. There exist three types of correlations - positive and negative, and zero correlations.

The main intention of Seaborn heatmap is to visualize the correlation matrix of data for feature selection to solve business problems Python data visualization seaborn library has a powerful function that is called sns.heatmap(). annot — **annot** when set to True, the correlation values become visible on the colored cells linewidth- **linewidth** Width of the lines that will divide each cell.

```
# plot a heatmap of correlation
#corr() is used to create the correlation matrix.
#You'll have to make sure that all the elements in the matrix are of
numeric type.
#If they are not of the numeric type you'll have to add or concat them
explicitly.

sns.set(rc = {'figure.figsize': (8, 8)})
sns.heatmap(movies[['budget', 'popularity', 'revenue', 'vote_average',
'director_id']].corr(),
            annot = True, fmt = '.2f', linewidth = 1);
```

Note On correlation :

The Heat Map Graph Shows The Effect Between One Variable To The Other Variables.

In The Graph, It Is Clear That budget Has The Greatest Impact on revenue with 0.73.

Meanwhile, popularity Is Quite Impact On revenue, And director_id Which Means director_name Also Has A Correlation With revenue, Even Though It Is Small.

However, vote_average Has A Very Small Correlation With revenue.

Directors

Question- List out the director names with number of movies and revenue

```
query = """
SELECT name,count(name),sum(revenue) as total_revenue
FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
GROUP BY director_id
ORDER BY count(name) desc
"""
total_movies = pd.read_sql(query, mycon)
total_movies
```

	name	count(name)	total_revenue
0	Steven Spielberg	27	9147393164
1	Woody Allen	21	669101038
2	Clint Eastwood	20	2512058888
3	Martin Scorsese	20	1956635998
4	Spike Lee	16	340618771
...	...	...	...
2344	Brenda Chapman	1	538983207
2345	Peter Sohn	1	331926147
2346	Lee Unkrich	1	1066969703
2347	Dan Scanlon	1	743559607
2348	Byron Howard	1	591794936

```
[2349 rows x 3 columns]
```

Note :

The Most ProfitableDirector Table Shows The Directors With The Highest Gross Profit.

Currently, **Steven Spielberg** is the Director With The Most Gross Profit, with **USD 9,147,393,164** from his **27** movies.

Then, **Peter Jackson** With USD **6,498,642,820**, **James Cameron** With USD **5,883,569,439**, And The Other Directors.

Question - Display all the number of movies for particular director and revenue .

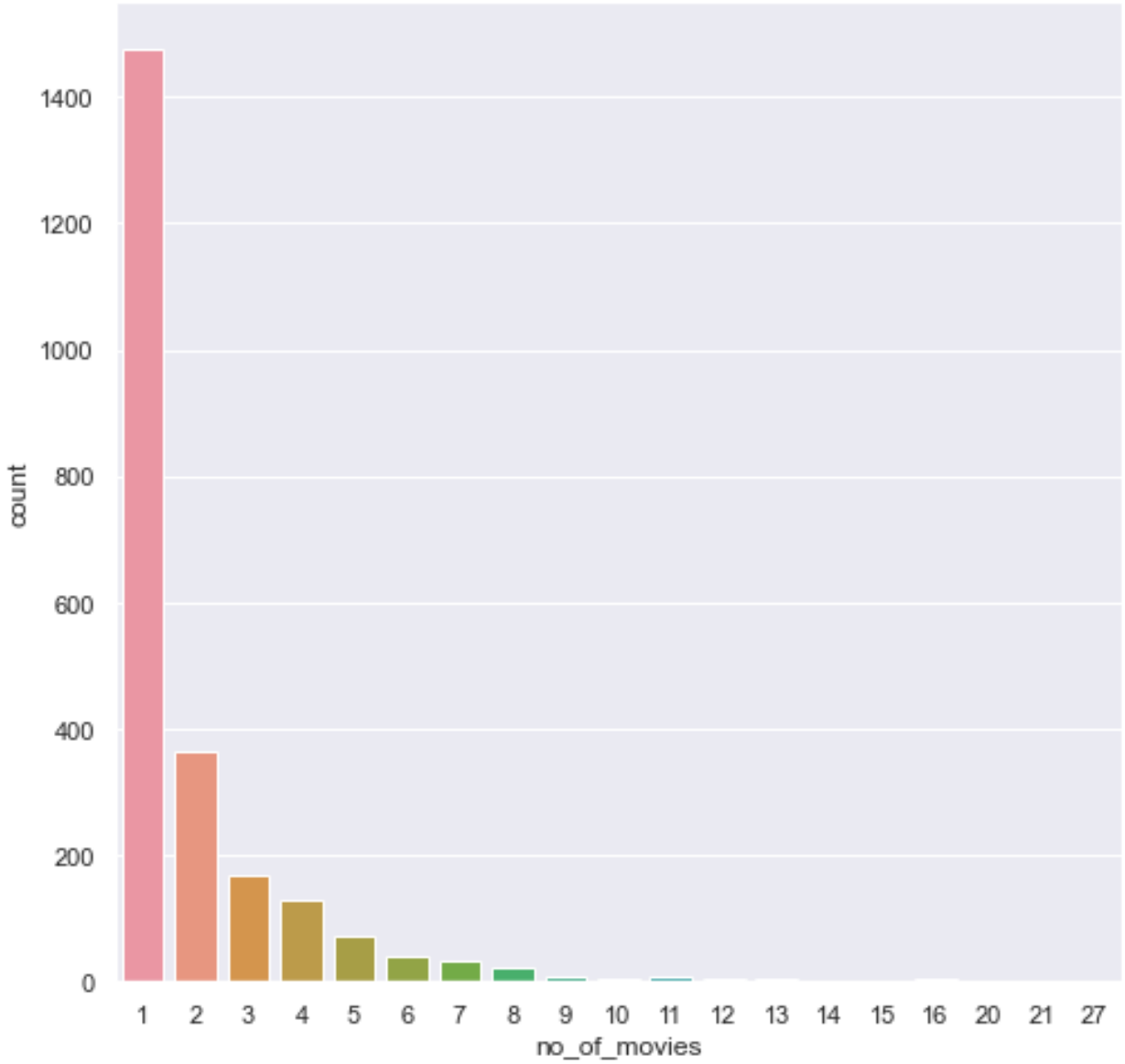
```
query = """
SELECT name,count(name)as no_of_movies,sum(revenue) as total_revenue
FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
GROUP BY director_id
ORDER BY no_of_movies desc
"""
```

```
total_movies = pd.read_sql(query, mycon)
total_movies
sns.countplot(data = total_movies,x="no_of_movies")

total_movies
```

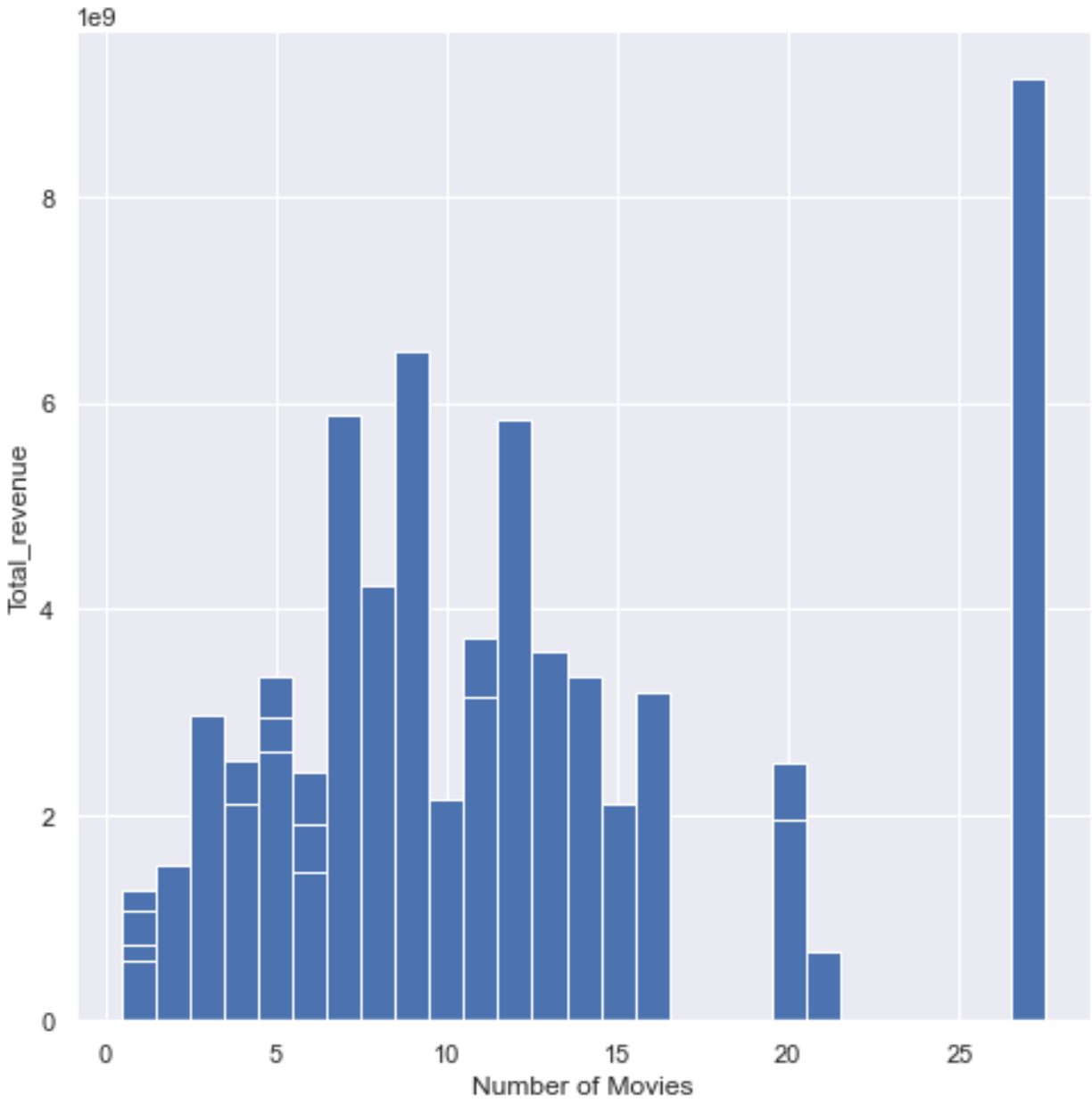
	name	no_of_movies	total_revenue
0	Steven Spielberg	27	9147393164
1	Woody Allen	21	669101038
2	Clint Eastwood	20	2512058888
3	Martin Scorsese	20	1956635998
4	Spike Lee	16	340618771
...	...	...	...
2344	Brenda Chapman	1	538983207
2345	Peter Sohn	1	331926147
2346	Lee Unkrich	1	1066969703
2347	Dan Scanlon	1	743559607
2348	Byron Howard	1	591794936

[2349 rows x 3 columns]




```
# no_of_movies V/S total_revenue

x = total_movies["no_of_movies"]
y = total_movies["total_revenue"]
# plot
fig,ax = plt.subplots()
ax.bar(x, y, width=1, edgecolor="white")
plt.xlabel("Number of Movies")
plt.ylabel("Total_revenue")
plt.show()
```



Note :

The mostProductiveDirector Table Contains The Name Of The Director And The Number Of Movies It Has Directed By Him.

Steven Spielberg Is The Most Productive Director With 27 Movies Titles He Has Directed. Meanwhile, Woody Allen Has Only 21 Movies Titles.

Then, Martin Scorsese And Clint Eastwood With 20 Movies Titles.

Question- By doing Director analysis We know Steven Spielberg is the highest revenue, so list the Steven Spielberg movies info

```
query = """
SELECT * FROM movies
LEFT JOIN directors
ON directors.id = movies.director_id
WHERE name = "Steven Spielberg"
ORDER BY revenue DESC
"""
ss_movies = pd.read_sql(query, mycon)
ss_movies
```

```
       id                                     original_title
budget  \
0   44272                                      Jurassic Park
63000000
1   46564                          E.T. the Extra-Terrestrial
10500000
2   43650  Indiana Jones and the Kingdom of the Crystal S...
185000000
3   43782                                   War of the Worlds
132000000
4   44225                                Saving Private Ryan
70000000
5   44603                 Indiana Jones and the Last Crusade
48000000
6   46406                                               Jaws
7000000
7   45682                            Raiders of the Lost Ark
18000000
8   43787                           The Adventures of Tintin
130000000
9   43872                                   Minority Report
102000000
10  44480                                Catch Me If You Can
52000000
11  45294               Indiana Jones and the Temple of Doom
28000000
12  45415                                   Schindler's List
22000000
13  45684                 Close Encounters of the Third Kind
20000000
14  44169                                               Hook
70000000
15  44230                                             Lincoln
65000000
16  43960                          A.I. Artificial Intelligence
100000000
17  44105                        The Lost World: Jurassic Park
73000000
18  44118                                     The Terminal
60000000
```

```
19  43772                                                The BFG
140000000
20  44182                                              War Horse
66000000
21  44784                                       Bridge of Spies
40000000
22  46117                                     The Color Purple
15000000
23  44125                                               Munich
70000000
24  44808                                              Amistad
36000000
25  45107                                                 1941
35000000
26  46603                           Twilight Zone: The Movie
10000000

    popularity release_date    revenue  \
0           40   1993-06-11  920100000
1           56   1982-04-03  792910554
2           75   2008-05-21  786636033
3           48   2005-06-28  591739379
4           76   1998-07-24  481840909
5           80   1989-05-24  474171806
6           50   1975-06-18  470654000
7           68   1981-06-12  389925971
8           89   2011-10-25  371940071
9           65   2002-06-20  358372926
10          73   2002-12-25  352114312
11          66   1984-05-23  333000000
12         104   1993-11-29  321365567
13          52   1977-11-16  303788635
14          33   1991-12-11  300854823
15          36   2012-11-09  275293450
16          34   2001-06-29  235926552
17           2   1997-05-23  229074524
18          57   2004-06-17  219417255
19          44   2016-06-01  183345589
20          29   2011-12-25  177584879
21          48   2015-10-15  165478348
22          17   1985-12-18  146292009
23          29   2005-12-22  130358911
24           3   1997-12-03   74000000
25          10   1979-12-13   31755742
26          12   1983-06-24   29450919

                                                 title  vote_average  \
0                                     Jurassic Park           7.6
1                         E.T. the Extra-Terrestrial           7.3
```

2	Indiana Jones and the Kingdom of the Crystal S...	5.7
3	War of the Worlds	6.2
4	Saving Private Ryan	7.9
5	Indiana Jones and the Last Crusade	7.6
6	Jaws	7.5
7	Raiders of the Lost Ark	7.7
8	The Adventures of Tintin	6.7
9	Minority Report	7.1
10	Catch Me If You Can	7.7
11	Indiana Jones and the Temple of Doom	7.1
12	Schindler's List	8.3
13	Close Encounters of the Third Kind	7.2
14	Hook	6.6
15	Lincoln	6.7
16	A.I. Artificial Intelligence	6.8
17	The Lost World: Jurassic Park	6.2
18	The Terminal	7.0
19	The BFG	6.0
20	War Horse	7.0
21	Bridge of Spies	7.2
22	The Color Purple	7.7
23	Munich	6.9
24	Amistad	6.8
25	1941	5.6
26	Twilight Zone: The Movie	6.2

	vote_count	overview
0	4856	A wealthy entrepreneur secretly creates a them...
1	3269	After a gentle alien becomes stranded on Earth...
2	2495	Set during the Cold War, the Soviets – led by ...
3	2322	Ray Ferrier is a divorced dockworker and less-...
4	5048	As U.S. troops storm the beaches of Normandy, ...
5	3152	When Dr. Henry Jones Sr. suddenly goes missing...
6	2542	An insatiable great white shark terrorizes the...
7	3854	When Dr. Indiana Jones – the tweed-suited prof...
8	2061	Intrepid young reporter, Tintin and his loyal ...
9	2608	John Anderton is a top 'Precrime' cop in the l...
10	3795	A true story about Frank Abagnale Jr. who, bef...
11	2781	After arriving in India, Indiana Jones is aske...
12	4329	The true story of how businessman Oskar Schind...
13	1098	After an encounter with UFOs, a line worker fe...
14	1532	The boy who wasn't supposed to grow up—Peter P...
15	1429	A revealing drama that focuses on the 16th Pre...
16	1974	A robotic boy, the first programmed to love, D...
17	2487	Four years after Jurassic Park's genetically b...
18	1910	Viktor Navorski is a man without a country; hi...
19	1000	The BFG is no ordinary bone-crunching giant. H...
20	992	Follows a young man named Albert and his horse...
21	2583	During the Cold War, the Soviet Union captures...

```
22         338  An epic tale spanning forty years in the life ...
23         696  During the 1972 Olympic Games in Munich, eleve...
24         316  In 1839, the slave ship Amistad set sail from ...
25         143  It's been six days since the attack on Pearl H...
26         161  Four directors collaborated to remake four epi...

                                                  tagline     uid
director_id  \
0        An adventure 65 million years in the making.     329
4799
1   He is afraid. He is alone. He is three million...     601
4799
2                       The adventure continues . . .     217
4799
3                                They're already here.      74
4799
4                                The mission is a man.     857
4799
5   The man with the hat is back. And this time, h...      89
4799
6                              Don't go in the water.     578
4799
7   Indiana Jones - the new hero from the creators...      85
4799
8   This year, discover how far adventure will tak...   17578
4799
9     The system is perfect until it comes after you.     180
4799
10                    The true story of a real fake.     640
4799
11  If adventure has a name... it must be Indiana ...      87
4799
12    Whoever saves one life, saves the world entire.     424
4799
13                                  We are not alone.     840
4799
14                         What if Peter Pan grew up?     879
4799
15  With the moral courage and fierce determinatio...   72976
4799
16  David is 11 years old. He weighs 60 pounds. He...     644
4799
17                             Something has survived.     330
4799
18                                   Life is waiting.     594
4799
19      The world is more giant than you can imagine.  267935
4799
20  Separated by War. Tested by Battle. Bound by F...   57212
```

```
4799
21  In the shadow of war, one man showed the world...  296098
4799
22   It's about life. It's about love. It's about us.     873
4799
23  The world was watching in 1972 as 11 Israeli a...     612
4799
24  Freedom is not given. It is our right at birth...   11831
4799
25                        Paranoia meets pandemonium.   11519
4799
26  You're travelling through another dimension. A...   15301
4799
```

	name	id	gender	uid	department
0	Steven Spielberg	4799	2	488	Directing
1	Steven Spielberg	4799	2	488	Directing
2	Steven Spielberg	4799	2	488	Directing
3	Steven Spielberg	4799	2	488	Directing
4	Steven Spielberg	4799	2	488	Directing
5	Steven Spielberg	4799	2	488	Directing
6	Steven Spielberg	4799	2	488	Directing
7	Steven Spielberg	4799	2	488	Directing
8	Steven Spielberg	4799	2	488	Directing
9	Steven Spielberg	4799	2	488	Directing
10	Steven Spielberg	4799	2	488	Directing
11	Steven Spielberg	4799	2	488	Directing
12	Steven Spielberg	4799	2	488	Directing
13	Steven Spielberg	4799	2	488	Directing
14	Steven Spielberg	4799	2	488	Directing
15	Steven Spielberg	4799	2	488	Directing
16	Steven Spielberg	4799	2	488	Directing
17	Steven Spielberg	4799	2	488	Directing
18	Steven Spielberg	4799	2	488	Directing
19	Steven Spielberg	4799	2	488	Directing
20	Steven Spielberg	4799	2	488	Directing
21	Steven Spielberg	4799	2	488	Directing
22	Steven Spielberg	4799	2	488	Directing
23	Steven Spielberg	4799	2	488	Directing
24	Steven Spielberg	4799	2	488	Directing
25	Steven Spielberg	4799	2	488	Directing
26	Steven Spielberg	4799	2	488	Directing

Note :

From Two Tables Above, mostProfitableDirector And mostProductiveDirector, It Can Be Seen That Steven Spielberg Is Always In The First Place.

It Can Be Understood That Steven Spielberg Has Produced Many And Incredible Movies That Are Shows In The Table MoviesByStevenSpielberg Above.