Importing the libraries

```
In [1]: import pandas as pd # For data manipulation and cleaning
        import numpy as np # For mathematical operations
        import sqlite3 as sq # Server-less database
```

1. **Establishing the connection and creating cursor**

```
In [2]: db=("C:/Users/Hanan Fouzer/Desktop/movie.sqlite")
        database=sq.connect(db)
        mycursor=database.cursor()
```

2. **Getting to know the tables present in the dataset**

```
In [3]: mycursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
        tables=mycursor.fetchall()
        tables

        # We will be using the movies table and directors table
```

```
Out[3]: [('movies',), ('sqlite_sequence',), ('directors',)]
```

3. **Creating DataFrames for both movies table and directors table**

```
In [4]: # For movies table

        mycursor.execute("""
        SELECT * FROM movies;
        """)

        # Using fetchall function
        data=mycursor.fetchall()

        # Creating the dataframe
        df=pd.DataFrame(data,columns=["ID","Original_Title","Budget","Popularity","Release_Date","Revenue","Title","Vote_Average",
                                      "Rating_count","Overview","Tagline","UID","Director_ID"])
```

```
In [10]: # For directors table

         mycursor.execute("""
         SELECT * FROM directors;
         """)

         # Using fetchall function
         data2=mycursor.fetchall()

         df2=pd.DataFrame(data2,columns=["Name","Director_ID","Gender","UID","Department"])
         df2
```

Data Exploration:

01. Using data frames

In [6]:
```
# For movies dataframe

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   Rating_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
```

In [9]:
```
# For directors dataframe

df2.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   Director_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
```

02. Using SQL + Python

In [30]:
```
# Checking how many movies are present in the movies table

# Using Aggregations
mycursor.execute("""
SELECT COUNT(*) FROM movies;
""")

# Using index to avoid the value being in a tuple
count=mycursor.fetchall()
print(f"The total number of movies present are {count[0][0]}")
```

```
The total number of movies present are 4773
```

In [12]:
```
#Counting the female directors where 1 is for female and 2 is for male

mycursor.execute("""
SELECT COUNT(*) FROM directors WHERE Gender=1;
""")

Count_fe=mycursor.fetchall()
Count_fe
```

Out[12]:
```
[(150,)]
```

In [31]:
```
# Counting the number of males

mycursor.execute("""
SELECT COUNT(*) FROM directors WHERE Gender=2;
""")

Count_ma=mycursor.fetchall()
Count_ma
```

Out[31]:
```
[(1574,)]
```

Data Analysis:

01. Top 3 popular movies

In [13]:
```
# Finding 3 most popular movies

# Using order by and limit on popularity column
mycursor.execute("""
SELECT Title FROM movies ORDER BY Popularity DESC LIMIT 3;
""")

# Using indexing to make it more clear
popular=mycursor.fetchall()
print(f"The 3 most popular movies are {popular[0][0]},{popular[1][0]},{popular[2][0]}")
```

```
The 3 most popular movies are Minions,Interstellar,Deadpool
```

02. Top 3 budget movies

In [16]:
```
# The 3 most bankable(highest budget) movies

# Using Order by
mycursor.execute("""
SELECT Title FROM movies ORDER BY Budget DESC LIMIT 3;
""")

bankable=mycursor.fetchall()
print(f"The 3 highest budget movies are:\n{bankable[0][0]},\n{bankable[1][0]},\n{bankable[2][0]}")
```

```
The 3 highest budget movies are:
Pirates of the Caribbean: On Stranger Tides,
Pirates of the Caribbean: At World's End,
Avengers: Age of Ultron
```

03. The most awarded average vote movie since Jan 1st,2000?

In [17]:
```
# The most awarded average vote movie since Jan 1st,2000

# Using WHERE on release_date column and order by for vote_average column

mycursor.execute("""
SELECT Title FROM movies WHERE Release_Date > '2000-01-01' ORDER BY Vote_Average DESC LIMIT 1;
""")

rated=mycursor.fetchall()
print (f"The most voted movie since 2000s is {rated[0][0]}")
```

```
The most voted movie since 2000s is Sardaarji
```

04. The director with the most number of movies

In [19]:
```
# Finding the director who has made the most movies

# Using joins, group by,order by and limit
mycursor.execute('''
SELECT Name FROM directors JOIN movies on movies.Director_ID=directors.ID GROUP BY Director_ID ORDER BY COUNT(Name)
DESC LIMIT 1;''')

most=mycursor.fetchall()
print(f"The director who has made the most movies is {most[0][0]}")
```

```
The director who has made the most movies is Steven Spielberg
```

05. The director with the highest budget for movies

In [20]:
```
# Find the director who is the most bankable (Highest budget)

mycursor.execute('''
SELECT Name FROM directors JOIN movies on movies.Director_ID=directors.ID GROUP BY Director_ID ORDER BY SUM(Budget)
DESC LIMIT 1;''')

director_bankable=mycursor.fetchall()
print(f"The most bankable director is {director_bankable[0][0]}")
```

```
The most bankable director is Steven Spielberg
```

06. Top 05 revenue making movies

In [38]:
```
# Find the top 5 revenue making movies

# Using Order by and Limit

mycursor.execute('''
SELECT TITLE FROM movies ORDER BY Revenue DESC LIMIT 5''')

Top_5_revenue=mycursor.fetchall()
Top_5_revenue
```

Out[38]:
```
[('Avatar',),
 ('Titanic',),
 ('The Avengers',),
 ('Jurassic World',),
 ('Furious 7',)]
```

07. Top 5 directors based on the revenue.

In [42]:
```
# Naming top 05 directors with the number of movies & revenue and putting it in a dataframe

# Using aggregation, joins, order by and limit
mycursor.execute('''
SELECT Name,COUNT(Title),SUM(Revenue) FROM movies JOIN directors ON movies.Director_ID=directors.ID
GROUP BY Director_ID ORDER BY SUM(REVENUE) DESC LIMIT 5;''')

Top_5_directors_revenue=mycursor.fetchall()

df_Top_5_directors_revenue=pd.DataFrame(Top_5_directors_revenue,columns=["Name","Movies_Made",
                                                                         "Total_Revenue"])
df_Top_5_directors_revenue
```

Out[42]:

	Name	Movies_Made	Total_Revenue
0	Steven Spielberg	27	9147393164
1	Peter Jackson	9	6498642820
2	James Cameron	7	5883569439
3	Michael Bay	12	5832524638
4	Christopher Nolan	8	4227483234

08. Top 5 directors based on the no. of movies made.

In [43]:
```
# Naming all the directors with no of movies and revenue,here the director who has the highest made movies
# appears first and so on

# Using aggregations, joins, group by, order by and limit
mycursor.execute('''
SELECT Name,COUNT(Title),SUM(Revenue) FROM movies JOIN directors ON movies.Director_ID=directors.ID
GROUP BY Director_ID ORDER BY COUNT(Title) DESC LIMIT 5;''')

Top_5_directors_movies=mycursor.fetchall()

df_Top_5_directors_movies=pd.DataFrame(Top_5_directors_movies,columns=["Name","Movies_Made",
                                                                       "Total_Revenue"])
df_Top_5_directors_movies
```

Out[43]:

	Name	Movies_Made	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