```python
#Importing libraries
import pandas as pd
import zipfile
import matplotlib.pyplot as plt
import sqlite3
import seaborn as sns


#loading the Movie Gross data(csv)
bom_file_path = r'C:\Users\Lenovo\Downloads\p2\NF\
bom.movie_gross.csv.csv'
movie_gross_df = pd.read_csv(bom_file_path)
bom_data=movie_gross_df.head()
bom_data
```

```
                                         title studio  domestic_gross  
\
0                                  Toy Story 3     BV     415000000.0   

1                   Alice in Wonderland (2010)     BV     334200000.0   

2  Harry Potter and the Deathly Hallows Part 1     WB     296000000.0   

3                                    Inception     WB     292600000.0   

4                          Shrek Forever After   P/DW     238700000.0   


  foreign_gross  year  
0     652000000  2010  
1     691300000  2010  
2     664300000  2010  
3     535700000  2010  
4     513900000  2010  
```

```python
# loading the imdb data and connecting to the SQLite database
imdb_zip_path = r'C:\Users\Lenovo\Downloads\p2\IMDB\im.db'
# connect to the SQLite database
conn = sqlite3.connect(r'C:\Users\Lenovo\Downloads\p2\IMDB\im.db')

# query the tables: loading the movie basics and ratings tables
movie_basics_df =pd.read_sql_query('SELECT * FROM movie_basics;',
conn)
movie_ratings_df = pd.read_sql_query('SELECT * FROM movie_ratings;',
conn )
#clocse the SQLite connection
conn.close()

# Displaying first 10 rowws of each dataframe
# the loaded movie basics
movie_basics_df.head(10)
```

```
    movie_id                    primary_title
original_title  \
0  tt0063540                        Sunghursh
Sunghursh
1  tt0066787  One Day Before the Rainy Season             Ashad Ka Ek
Din
2  tt0069049       The Other Side of the Wind  The Other Side of the
Wind
3  tt0069204                  Sabse Bada Sukh             Sabse Bada
Sukh
4  tt0100275         The Wandering Soap Opera       La Telenovela
Errante
5  tt0111414                      A Thin Life                 A Thin
Life
6  tt0112502                          Bigfoot
Bigfoot
7  tt0137204                  Joe Finds Grace             Joe Finds
Grace
8  tt0139613                       O Silêncio                  O
Silêncio
9  tt0144449           Nema aviona za Zagreb       Nema aviona za
Zagreb

   start_year  runtime_minutes                       genres
0        2013            175.0           Action,Crime,Drama
1        2019            114.0              Biography,Drama
2        2018            122.0                        Drama
3        2018              NaN                 Comedy,Drama
4        2017             80.0         Comedy,Drama,Fantasy
5        2018             75.0                       Comedy
6        2017              NaN              Horror,Thriller
7        2017             83.0  Adventure,Animation,Comedy
8        2012              NaN        Documentary,History
9        2012             82.0                    Biography
```

```
#Inspecting the datasets
movie_basics_df.info(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 146144 entries, 0 to 146143
Data columns (total 6 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   movie_id         146144 non-null  object
 1   primary_title    146144 non-null  object
 2   original_title   146123 non-null  object
 3   start_year       146144 non-null  int64
 4   runtime_minutes  114405 non-null  float64
 5   genres           140736 non-null  object
```

```
dtypes: float64(1), int64(1), object(4)
memory usage: 6.7+ MB
```

```
movie_basics_df.describe()
```

```
          start_year  runtime_minutes
count  146144.000000    114405.000000
mean     2014.621798        86.187247
std         2.733583       166.360590
min      2010.000000         1.000000
25%      2012.000000        70.000000
50%      2015.000000        87.000000
75%      2017.000000        99.000000
max      2115.000000     51420.000000
```

```
# the loaded movie ratings
movie_ratings_df.head(10)
```

```
     movie_id  averagerating  numvotes
0  tt10356526            8.3        31
1  tt10384606            8.9       559
2   tt1042974            6.4        20
3   tt1043726            4.2     50352
4   tt1060240            6.5        21
5   tt1069246            6.2       326
6   tt1094666            7.0      1613
7   tt1130982            6.4       571
8   tt1156528            7.2       265
9   tt1161457            4.2       148
```

```
movie_ratings_df.info(10)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 73856 entries, 0 to 73855
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   movie_id       73856 non-null  object
 1   averagerating  73856 non-null  float64
 2   numvotes       73856 non-null  int64
dtypes: float64(1), int64(1), object(1)
memory usage: 1.7+ MB
```

```
movie_ratings_df.describe()
```

```
       averagerating      numvotes
count   73856.000000  7.385600e+04
mean        6.332729  3.523662e+03
std         1.474978  3.029402e+04
min         1.000000  5.000000e+00
25%         5.500000  1.400000e+01
```

```
50%         6.500000  4.900000e+01
75%         7.400000  2.820000e+02
max        10.000000  1.841066e+06

#Data cleaning and preparation
#checking for missing values
missing_values=(movie_gross_df.isnull().sum())
missing_values_1=(movie_basics_df.isnull().sum())
missing_values_2=(movie_ratings_df.isnull().sum())

# Droping rows with missing values
movie_basics_df.dropna(subset=['original_title'], inplace= True)
movie_basics_df.dropna(subset=['runtime_minutes'], inplace= True)
movie_basics_df.dropna(subset=['genres'], inplace= True)
movie_gross_df.dropna(subset=['studio'], inplace= True)
movie_gross_df.dropna(subset=['foreign_gross'], inplace= True)
movie_gross_df.dropna(subset=['domestic_gross'], inplace= True)

#All missing values dropped
print(missing_values)
print(missing_values_1)
print(missing_values_2)

title                0
studio               5
domestic_gross      28
foreign_gross     1350
year                 0
dtype: int64
movie_id               0
primary_title          0
original_title        21
start_year             0
runtime_minutes    31739
genres              5408
dtype: int64
movie_id         0
averagerating    0
numvotes         0
dtype: int64

#REMOVING duplicates
print(movie_gross_df.duplicated().sum())
print(movie_basics_df.duplicated().sum())
print(movie_ratings_df.duplicated().sum())

0
0
0
```

```
print('columns in movie_basics_df')
print(movie_basics_df.columns)
```

```
columns in movie_basics_df
Index(['movie_id', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres'],
      dtype='object')
```

```
print('columns in bom_data')
print(bom_data.columns)
```

```
columns in bom_data
Index(['title', 'studio', 'domestic_gross', 'foreign_gross', 'year'],
dtype='object')
```

```
# Merging Datasets for EDA
merged_data = pd.merge(movie_basics_df, bom_data, left_on =
'movie_id', right_on ='title',  how ='inner')
merged_data_1= pd.merge(merged_data, bom_data, left_on =
'primary_title', right_on ='title',  how ='inner')
print(merged_data.head())
```

```
Empty DataFrame
Columns: [movie_id, primary_title, original_title, start_year,
runtime_minutes, genres, title, studio, domestic_gross, foreign_gross,
year]
Index: []
```

```
print(merged_data_1.head())
```

```
Empty DataFrame
Columns: [movie_id, primary_title, original_title, start_year,
runtime_minutes, genres, title_x, studio_x, domestic_gross_x,
foreign_gross_x, year_x, title_y, studio_y, domestic_gross_y,
foreign_gross_y, year_y]
Index: []
```

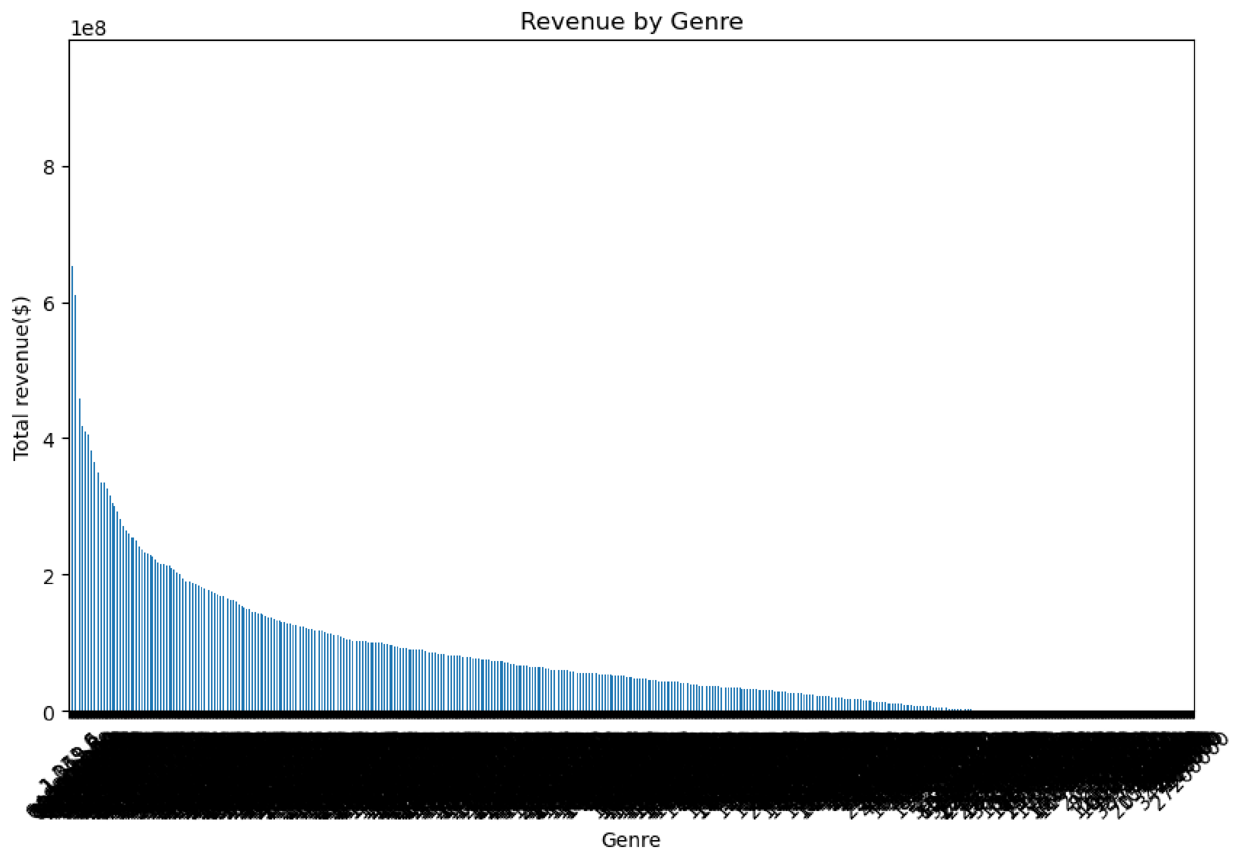
```
print(merged_data.columns)
```

```
Index(['movie_id', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres', 'title', 'studio',
'domestic_gross',
       'foreign_gross', 'year'],
      dtype='object')
```

```
print(merged_data.head())
```

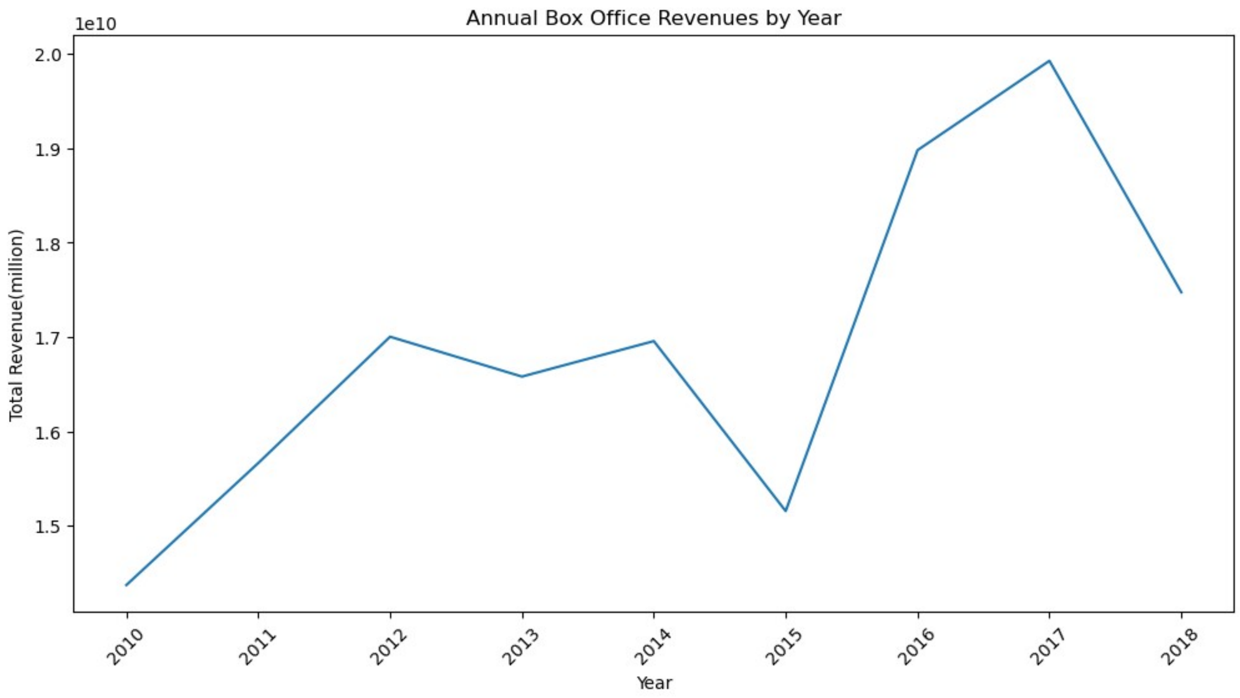
```
Empty DataFrame
Columns: [movie_id, primary_title, original_title, start_year,
runtime_minutes, genres, title, studio, domestic_gross, foreign_gross,
year]
Index: []
```

```
#Aggregate gross earnings by genre
genre_gross = merged_data_1.groupby('movie_id')
['foreign_gross_y'].sum().reset_index()
genre_gross.sort_values(by='foreign_gross_y', ascending=False,
inplace=True)

print(genre_gross.head())
print(genre_gross.columns)

Empty DataFrame
Columns: [movie_id, foreign_gross_y]
Index: []
Index(['movie_id', 'foreign_gross_y'], dtype='object')

print(genre_gross.head())  # Check the first few rows of the DataFrame
print(genre_gross.isnull().sum())  # Check for null values

Empty DataFrame
Columns: [movie_id, foreign_gross_y]
Index: []
movie_id           0
foreign_gross_y    0
dtype: int64

print(genre_gross.dtypes)

movie_id           object
foreign_gross_y    object
dtype: object

genre_gross['foreign_gross_y'] =
pd.to_numeric(genre_gross['foreign_gross_y'], errors='coerce')

#Group by Genre and total revenue
genre_revenue= movie_gross_df.groupby('foreign_gross')
['domestic_gross'].sum().sort_values(ascending=False)
genre_revenue.plot(kind='bar', figsize=(10,6))
plt.title('Revenue by Genre')
plt.xlabel('Genre')
plt.ylabel('Total revenue($)')
plt.xticks(rotation=45)
plt.show()
```



```
annual_revenue = movie_gross_df.groupby('year')
['foreign_gross'].sum().reset_index()
print(annual_revenue.head())

   year                                      foreign_gross
0  2010  65200000069130000066430000053570000051390000003...
1  2011  96050000077140000080460000043090000048530000005...
2  2012  89550000080420000063680000071810000071590000005...
3  2013  87570000080580000060270000070000000044030000005...
4  2014  85860000070090000044020000051710000041820000005...

movie_gross_df

                                            title      studio
\
0                                    Toy Story 3          BV

1                     Alice in Wonderland (2010)          BV

2    Harry Potter and the Deathly Hallows Part 1          WB

3                                      Inception          WB

4                            Shrek Forever After        P/DW
```

```
...                                                      ...           ...
3275                                            I Still See You           LGF
3286                                      The Catcher Was a Spy           IFC
3309                                                 Time Freak    Grindstone
3342  Reign of Judges: Title of Liberty - Concept Short  Darin Southa
3353            Antonio Lopez 1970: Sex Fashion & Disco            FM

      domestic_gross foreign_gross  year
0        415000000.0     652000000  2010
1        334200000.0     691300000  2010
2        296000000.0     664300000  2010
3        292600000.0     535700000  2010
4        238700000.0     513900000  2010
...              ...           ...   ...
3275          1400.0       1500000  2018
3286        725000.0        229000  2018
3309         10000.0        256000  2018
3342         93200.0          5200  2018
3353         43200.0         30000  2018

[2007 rows x 5 columns]

movie_gross_df['year']= pd.to_numeric(movie_gross_df['year'], errors =
'coerce')
movie_gross_df['foreign_gross']=
pd.to_numeric(movie_gross_df['foreign_gross'], errors='coerce')
movie_gross_df['year']=pd.to_numeric(movie_gross_df['year'],
errors='coerce')

movie_gross_df['foreign_gross']=
pd.to_numeric(movie_gross_df['foreign_gross'], errors='coerce')
annual_revenue = movie_gross_df.groupby('year')
['foreign_gross'].sum().reset_index()
#plotting the trend
plt.figure(figsize=(12,6))
sns.lineplot(data= annual_revenue, x= 'year', y='foreign_gross')
plt.title('Annual Box Office Revenues by Year')
plt.xlabel('Year')
plt.ylabel('Total Revenue(million)')
plt.xticks(rotation=45)
plt.show()
```


Annual Box Office Revenues by Year
1e10
2.0
1.9
1.8
1.7
1.6
1.5
Total Revenue(million)
2010
2011
2012
2013
2014
2015
2016
2017
2018
Year