Final Project Submission

Please fill out:

- Student name: JASMINE NJERI
- Student pace: FULL TIME HYBRID
- Scheduled project review date/time:
- Instructor name: ANTONNY MUIKO
- Blog post URL:

1.0 BUSINESS UNDERSTANDING

2.0 OBJECTIVE

The main objective is to explore the type of films which are doing best in the box office and use findings to decide what type of movie to create

In [ ]:
```
# Importing relevant libraries
import sqlite3
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [ ]:
```
# Connecting to sqlite database
path = r"C:\Users\HP\Documents\Jas_labs\Movie-Recommendation-Analysis-Phase-2-Project-\
conn = sqlite3.connect(path)
```

In [ ]:
```
# Listing down  all the tables in the database
pd.read_sql(
"""
SELECT name FROM sqlite_master WHERE type='table';
""", conn)
```

Out[ ]:

	name
0	movie_basics
1	directors
2	known_for
3	movie_akas
4	movie_ratings
5	persons
6	principals
7	writers

READ THE CSV FILE

In [ ]:
```
# Loading movie_gross file into a dataframe
movie_gross_path = r"C:\Users\HP\Documents\Jas_labs\dsc-phase-2-project-v3-1\zippedData
movie_gross_df = pd.read_csv(movie_gross_path)
movie_gross_df
```

Out[ ]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010
...	...	...	...	...	...
3382	The Quake	Magn.	6200.0	NaN	2018
3383	Edward II (2018 re-release)	FM	4800.0	NaN	2018
3384	El Pacto	Sony	2500.0	NaN	2018
3385	The Swan	Synergetic	2400.0	NaN	2018
3386	An Actor Prepares	Grav.	1700.0	NaN	2018

3387 rows × 5 columns

3.0 EXPLORING THE DATA

3.1 For SQLite database

In [ ]:
```
# Calling all the data in the movie_ratings table
movie_ratings_df = pd.read_sql("""
SELECT *
FROM movie_ratings
""",conn)
movie_ratings_df
```

Out[ ]:

	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
...	...	...	...
73851	tt9805820	8.1	25
73852	tt9844256	7.5	24

	movie_id	averagerating	numvotes
73853	tt9851050	4.7	14
73854	tt9886934	7.0	5
73855	tt9894098	6.3	128

73856 rows × 3 columns

In [ ]:
```
movie_basics_df = pd.read_sql("""
SELECT *
FROM movie_basics
""",conn)
movie_basics_df
```

Out[ ]:

	movie_id	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy
...	...	...	...	...	...	...
146139	tt9916538	Kuambil Lagi Hatiku	Kuambil Lagi Hatiku	2019	123.0	Drama
146140	tt9916622	Rodolpho Teóphilo - O Legado de um Pioneiro	Rodolpho Teóphilo - O Legado de um Pioneiro	2015	NaN	Documentary
146141	tt9916706	Dankyavar Danka	Dankyavar Danka	2013	NaN	Comedy
146142	tt9916730	6 Gunn	6 Gunn	2017	116.0	None
146143	tt9916754	Chico Albuquerque - Revelações	Chico Albuquerque - Revelações	2013	NaN	Documentary

146144 rows × 6 columns

3.2 For CSV File

In [ ]:
```
movie_gross_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3387 entries, 0 to 3386
Data columns (total 5 columns):
```

```
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   title           3387 non-null   object 
 1   studio          3382 non-null   object 
 2   domestic_gross  3359 non-null   float64
 3   foreign_gross   2037 non-null   object 
 4   year            3387 non-null   int64  
dtypes: float64(1), int64(1), object(3)
memory usage: 132.4+ KB
```

In [ ]:
```
# Display the column names
movie_gross_df.columns
```

Out[ ]:
```
Index(['title', 'studio', 'domestic_gross', 'foreign_gross', 'year'], dtype='object')
```

In [ ]:
```
# Display the first few rows
movie_gross_df.head()
```

Out[ ]:

	title	studio	domestic_gross	foreign_gross	year
0	Toy Story 3	BV	415000000.0	652000000	2010
1	Alice in Wonderland (2010)	BV	334200000.0	691300000	2010
2	Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	664300000	2010
3	Inception	WB	292600000.0	535700000	2010
4	Shrek Forever After	P/DW	238700000.0	513900000	2010

4.0 DATA CLEANING

4.1 Checking for missing values in the dataframe

In [ ]:
```
movie_gross_df.isnull().sum()
```

Out[ ]:
```
title               0
studio              5
domestic_gross     28
foreign_gross    1350
year                0
dtype: int64
```

4.2 Filling in missing values

In [ ]:
```
# Filling in missing values in studio column with 'unknown'
movie_gross_df['studio'].fillna('Unknown', inplace=True)
```

In [ ]:
```
# Filling in missing values in domestic gross using median
movie_gross_df['domestic_gross'] = movie_gross_df['domestic_gross'].fillna(movie_gross_
```

In [ ]:
```
# Since foreign_gross has a lot of missing values, we can use the predictive modelling 
print(movie_gross_df[movie_gross_df["foreign_gross"].isna()].describe())
```

```
       domestic_gross         year
count    1.350000e+03  1350.000000
mean     1.620968e+06  2014.660000
std      5.106640e+06     2.109834
min      1.000000e+02  2010.000000
```

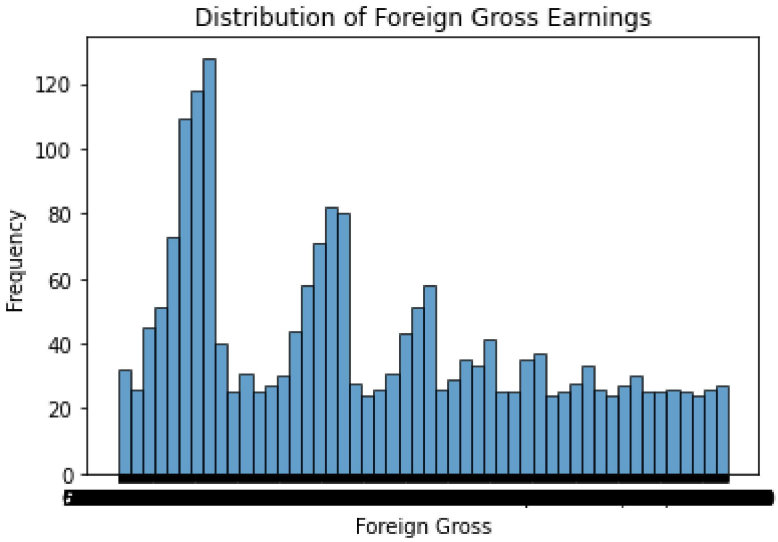
```
25%      4.197500e+04  2013.000000
50%      1.800000e+05  2015.000000
75%      1.000000e+06  2016.000000
max      6.860000e+07  2018.000000
```

In [ ]:
```
movie_gross_df.isnull().sum()
```

Out[ ]:
```
title               0
studio              0
domestic_gross      0
foreign_gross    1350
year                0
dtype: int64
```

In [ ]:
```
# Plot the distribution of foreign_gross
plt.hist(movie_gross_df['foreign_gross'].dropna(), bins=50, edgecolor='k', alpha=0.7)
plt.title('Distribution of Foreign Gross Earnings')
plt.xlabel('Foreign Gross')
plt.ylabel('Frequency')
plt.show()


# Print summary statistics
print(movie_gross_df['foreign_gross'].describe())
```




```
count        2037
unique       1204
top       1200000
freq           23
Name: foreign_gross, dtype: object
```

In [ ]:
```
# Filling in missing values in the foreign_gross column using the mode
movie_gross_df['foreign_gross'].fillna(movie_gross_df['foreign_gross'].mode()[0], inpla
```

In [ ]:
```
# Checking for missing values
movie_gross_df.isnull().sum()
```

Out[ ]:
```
title            0
studio           0
domestic_gross   0
foreign_gross    0
year             0
dtype: int64
```

In [ ]:
```
# Checking the datatype of foreign_gross column
print(movie_gross_df["foreign_gross"].dtype)
```

```
object
```

In [ ]:
```
non_numeric_values = movie_gross_df[~pd.to_numeric(movie_gross_df["foreign_gross"], err
print(non_numeric_values)
```

```
                          title studio  domestic_gross foreign_gross  year
1872  Star Wars: The Force Awakens     BV     936700000.0       1,131.6  2015
1873               Jurassic World   Uni.     652300000.0       1,019.4  2015
1874                    Furious 7   Uni.     353000000.0       1,163.0  2015
2760       The Fate of the Furious   Uni.     226000000.0       1,010.0  2017
3079        Avengers: Infinity War     BV     678800000.0       1,369.5  2018
```

In [ ]:
```
# Changing the datatype of the foreign_gross column
movie_gross_df["foreign_gross"] = pd.to_numeric(movie_gross_df["foreign_gross"], errors
```

In [ ]:
```
print(movie_gross_df["foreign_gross"].dtype)
```

```
float64
```

In [ ]:
```
movie_gross_df["foreign_gross"].mean()
```

Out[ ]:
```
45575372.14163217
```

In [ ]:
```
movie_gross_df["foreign_gross"].describe()
```

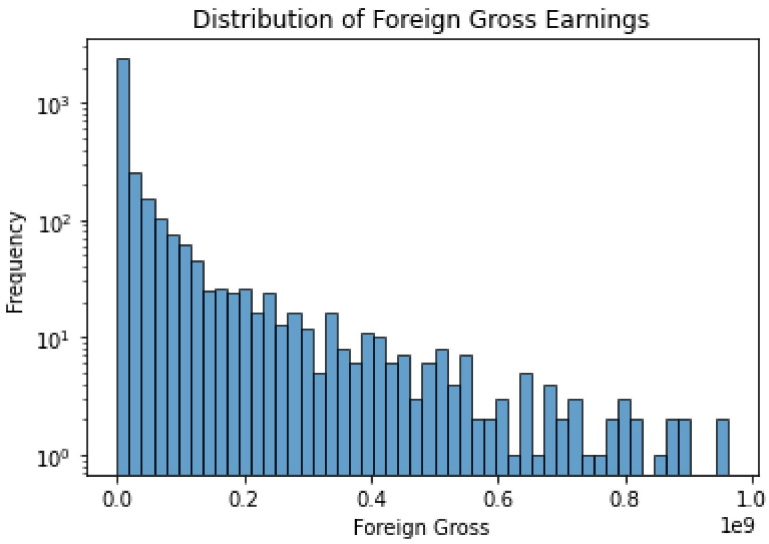
Out[ ]:
```
count    3.382000e+03
mean     4.557537e+07
std      1.125641e+08
min      6.000000e+02
25%      1.200000e+06
50%      1.500000e+06
75%      2.920000e+07
max      9.605000e+08
Name: foreign_gross, dtype: float64
```

In [ ]:
```
movie_gross_df["foreign_gross"].tail(100)
```

Out[ ]:
```
3287    1200000.0
3288    1200000.0
3289    1200000.0
3290    1200000.0
3291    1200000.0
          ...
3382    1200000.0
3383    1200000.0
3384    1200000.0
3385    1200000.0
3386    1200000.0
Name: foreign_gross, Length: 100, dtype: float64
```

In [ ]:
```
# Plot the distribution of foreign_gross
plt.hist(movie_gross_df['foreign_gross'].dropna(), bins=50, edgecolor='k', alpha=0.7)
plt.title('Distribution of Foreign Gross Earnings')
plt.xlabel('Foreign Gross')
plt.ylabel('Frequency')
plt.yscale('log')
plt.show()
```

```
# Print summary statistics
print(movie_gross_df['foreign_gross'].describe())
```




```
count    3.382000e+03
mean     4.557537e+07
std      1.125641e+08
min      6.000000e+02
25%      1.200000e+06
50%      1.500000e+06
75%      2.920000e+07
max      9.605000e+08
Name: foreign_gross, dtype: float64
```

4.3 EXPLORING MOVIE_BASICS AND MOVIE_RATINGS IN IM.DB

In [ ]:
```
#Checking the info in movie_ratingss_df
movie_ratings_df.info()
```

```
<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
```

In [ ]:
```
# Checking info in movie_basicss_df.
movie_basics_df.info()
```

```
<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
```

In [ ]:
```
movie_ratings_df.describe()
```

Out[ ]:

	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

In [ ]:
```
movie_basics_df.describe()
```

Out[ ]:

	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

In [ ]:
```
# Checking for null values in movie_basics_df
movie_basics_df.isnull().sum()
```

Out[ ]:
```
movie_id              0
primary_title         0
original_title       21
start_year            0
runtime_minutes    31739
genres              5408
dtype: int64
```

In [ ]:
```
# Checking for null values in movie_ratings_df
movie_ratings_df.isnull().sum()
```

Out[ ]:
```
movie_id         0
averagerating    0
numvotes         0
dtype: int64
```

In [ ]:
```
# Checking for duplicates in movie_ratings_df
movie_ratings_df.duplicated().sum()
```

Out[ ]:
```
0
```

From the cells above, you can tell movie_ratings_df does not contain any null values hence no cleaning needed.

4.3.1 EXPLORING MOVIE_BASICS_DF

In [ ]:
```
#Checking the info in movie_basics_df
movie_ratings_df.info()
```

```
<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
```

In [ ]:
```
movie_ratings_df.describe()
```

Out[ ]:

	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

In [ ]:
```
# Checking for missing values in movie_basics_df
movie_ratings_df.isnull().sum()
```

Out[ ]:
```
movie_id         0
averagerating    0
numvotes         0
dtype: int64
```

In [ ]:
```
# Checking the percentage of missing values in movie_basics_df
(movie_ratings_df.isnull().sum()/len(movie_ratings_df))*100
```

Out[ ]:
```
movie_id         0.0
averagerating    0.0
numvotes         0.0
dtype: float64
```

Since the percentage of missing values in genres and original_title is very minimal, dropping it wont have a major effect

In [ ]:
```
movie_basics_df = movie_basics_df.dropna(subset = ["genres", "original_title"])
```

In [ ]:
```
movie_basics_df.isnull().sum()
```

Out[ ]:
```
movie_id              0
primary_title         0
original_title        0
start_year            0
runtime_minutes    28502
genres                0
dtype: int64
```

In [ ]:
```
movie_basics_df["runtime_minutes"]
```

Out[ ]:
```
0         175.0
1         114.0
2         122.0
3           NaN
4          80.0
          ...
146138      NaN
146139    123.0
146140      NaN
146141      NaN
146143      NaN
Name: runtime_minutes, Length: 140734, dtype: float64
```

In [ ]:
```
# Filling the null values using median
movie_basics_df["runtime_minutes"] = movie_basics_df["runtime_minutes"].fillna(movie_ba
```

```
<ipython-input-56-169bbc722849>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  movie_basics_df["runtime_minutes"] = movie_basics_df["runtime_minutes"].fillna(movie_b
asics_df["runtime_minutes"].median())
```

In [ ]:
```
movie_basics_df.isnull().sum()
```

Out[ ]:
```
movie_id           0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
dtype: int64
```

Saving the cleaned data into csv for visualisations in tableau

In [ ]:
```
# Saving cleaned Dataframes
movie_gross_df.to_csv('cleaned_movie_gross_df', index=False)
movie_basics_df.to_csv('cleaned_movie_basics_df', index=False)
movie_ratings_df.to_csv('cleaned_movie_ratings_df', index=False)
```

In [ ]:
```
# Merge movie_ratings_df with movie_basics_df on 'movie_id'
merged_df = pd.merge(movie_ratings_df, movie_basics_df, on='movie_id', how='inner')
```

```
merged_df
```

Out[ ]:

	movie_id	averagerating	numvotes	primary_title	original_title	start_year	runtime_minutes	
0	tt10356526	8.3	31	Laiye Je Yaarian	Laiye Je Yaarian	2019	117.0	
1	tt10384606	8.9	559	Borderless	Borderless	2019	87.0	
2	tt1042974	6.4	20	Just Inès	Just Inès	2010	90.0	
3	tt1043726	4.2	50352	The Legend of Hercules	The Legend of Hercules	2014	99.0	Ac
4	tt1060240	6.5	21	Até Onde?	Até Onde?	2011	73.0	
...	...	...	...	...	...	...	...	
73047	tt9805820	8.1	25	Caisa	Caisa	2018	84.0	
73048	tt9844256	7.5	24	Code Geass: Lelouch of the Rebellion - Glorifi...	Code Geass: Lelouch of the Rebellion Episode III	2018	120.0	
73049	tt9851050	4.7	14	Sisters	Sisters	2019	87.0	
73050	tt9886934	7.0	5	The Projectionist	The Projectionist	2019	81.0	
73051	tt9894098	6.3	128	Sathru	Sathru	2019	129.0	

73052 rows × 8 columns

In [ ]:
```
# Changing the merged_df ro csv
merged_df.to_csv('cleaned_merged_df', index=False)
```

In [ ]:
```
# Merging merged_df with movie_gross_df
final_merged_df = pd.merge(merged_df, movie_gross_df, left_on='primary_title', right_on
```

In [ ]:
```
print(final_merged_df.isnull().sum())
```

```
movie_id           0
averagerating      0
numvotes           0
primary_title      0
original_title     0
start_year         0
runtime_minutes    0
genres             0
title              0
studio             0
domestic_gross     0
foreign_gross      4
year               0
dtype: int64
```

In [ ]:
```
#   Dropping the primary_title and original_title
final_merged_df.drop(columns = ["primary_title", "original_title"], inplace = True)
```

In [ ]:
```
# Saving the final merged data to csv
final_merged_df.to_csv("clean_merged_movie_data.csv", index=False)
```

In [ ]:
```

```