

# Greenlight Analytics — Phase 2 Project

## Executive Summary

This notebook analyzes movie industry data from Box Office Mojo and IMDb (via SQLite) to guide strategic decisions for a new film studio.

The analysis focuses on three key business questions:

1. **Which genres and runtimes are most successful?**
2. **Do ratings correlate with revenue?**
3. **How do domestic vs. foreign markets vary by studio?**

## Key Insights

- Action, Adventure, and Fantasy films consistently generate the highest box office revenues.
- Studios differ significantly in their domestic vs. foreign revenue mix — release strategies matter.
- Runtimes between 95–120 minutes correlate with stronger ratings and solid box office performance.

## Recommendations

1. Prioritize high-performing genres.
2. Balance domestic and international release strategies.
3. Greenlight films with strong audience signals (ratings, genre fit, runtime).

## Data Loading

We combine multiple sources:

- **Box Office Mojo:** CSV with domestic and foreign grosses.
- **IMDb (SQLite):** movie basics and ratings tables.

The datasets are merged using common keys such as `title / primary_title` , `year / start_year` , and `movie_id` .

```
In [1]: import pandas as pd
import sqlite3
import numpy as np

# Load Box Office Mojo data
bom = pd.read_csv('bom.movie_gross.csv.gz')
#bom.head()
```

```
In [2]: # Load IMDb SQLite tables
path = "im.db"
conn = sqlite3.connect(path)
basics = pd.read_sql("SELECT * FROM movie_basics", conn)
ratings = pd.read_sql("SELECT * FROM movie_ratings", conn)

# Merge Box Office Mojo with IMDb basics
merged_1_2 = pd.merge(bom, basics, left_on=['title', 'year'], right_on=['primary_title', 'year'])

# Merge with ratings
df = pd.merge(merged_1_2, ratings, on='movie_id', how='left')
df.tail(5)
```

Out[2]:

	title	studio	domestic_gross	foreign_gross	year	movie_id	primary_title	original_title
1868	Girls vs Gangsters	WGUSA	37100.0	NaN	2018	tt7870578	Girls vs Gangsters	Gui mi :
1869	The Workshop	Strand	22100.0	NaN	2018	tt7405478	The Workshop	The Worksho
1870	A Paris Education	KL	21600.0	NaN	2018	tt6593240	A Paris Education	Me provinciale
1871	The Quake	Magn.	6200.0	NaN	2018	tt6523720	The Quake	Skjelve
1872	An Actor Prepares	Grav.	1700.0	NaN	2018	tt5718046	An Actor Prepares	An Acto Prepare



## Data Cleaning

Before analysis, we perform several cleaning steps:

- Remove duplicate rows.
- Handle missing values by dropping rows with critical nulls.
- Convert column types (e.g., `foreign_gross`).
- Fill missing `foreign_gross` values with 0.
- Create a new `total_gross` column combining domestic and foreign grosses.

```
In [3]: # Check for duplicates
df.duplicated().sum()

# Drop rows missing essential values
df.dropna(subset=['averagerating', 'numvotes', 'studio', 'runtime_minutes', 'genres'],

# Convert foreign_gross to numeric
df['foreign_gross'] = pd.to_numeric(df['foreign_gross'], errors='coerce')

# Replace missing values with 0
df['foreign_gross'].fillna(0, inplace=True)

# Add total_gross column
df['total_gross'] = df['domestic_gross'] + df['foreign_gross']
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1832 entries, 0 to 1872
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   title                  1832 non-null   object
1   studio                 1832 non-null   object
2   domestic_gross         1832 non-null   float64
3   foreign_gross          1832 non-null   float64
4   year                   1832 non-null   int64
5   movie_id               1832 non-null   object
6   primary_title          1832 non-null   object
7   original_title         1832 non-null   object
8   start_year             1832 non-null   int64
9   runtime_minutes        1832 non-null   float64
10  genres                 1832 non-null   object
11  averagerating          1832 non-null   float64
12  numvotes               1832 non-null   float64
13  total_gross            1832 non-null   float64
dtypes: float64(6), int64(2), object(6)
memory usage: 214.7+ KB
```

## Question 1: Which Genres and Runtimes Are Most Successful?

We explore revenue trends by genre and runtime.

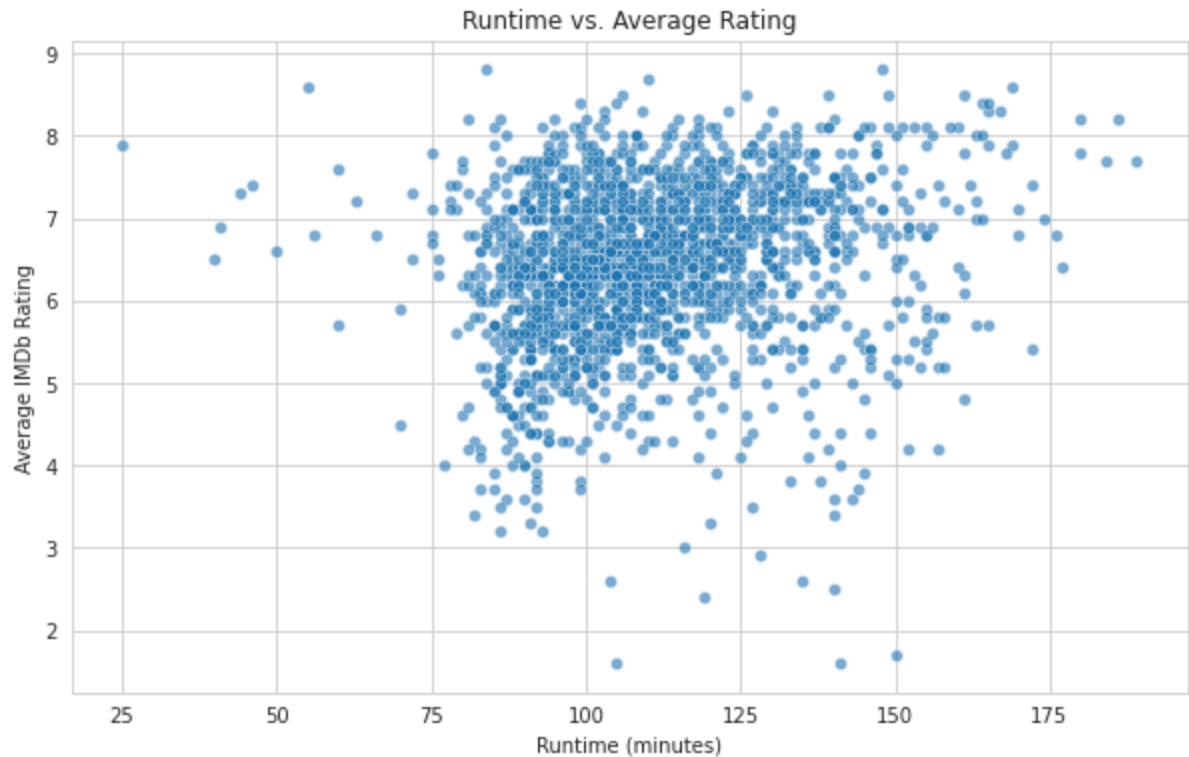
```
In [4]: import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')

# Revenue by genre
genre_gross = df.groupby('genres')['total_gross'].agg(['mean', 'median', 'count'])
genre_gross.head(10)
```

Out[4]:

	mean	median	count
genres			
Adventure,Fantasy	7.040333e+08	956000000.0	3
Adventure,Drama,Sci-Fi	6.537500e+08	653750000.0	2
Action,Adventure,Sci-Fi	6.120711e+08	605499999.0	45
Action,Comedy,Mystery	5.441000e+08	544100000.0	1
Action,Adventure,Fantasy	4.565960e+08	364700000.0	28
Biography,Drama,Musical	4.350000e+08	435000000.0	1
Adventure,Mystery,Sci-Fi	4.034000e+08	403400000.0	1
Adventure,Animation,Comedy	4.033133e+08	310650000.0	68
Action,Adventure,Thriller	3.923999e+08	277200000.0	16
Adventure,Family,Fantasy	3.849875e+08	249650000.0	8

```
In [5]: # Runtime vs. rating
plt.figure(figsize=(10,6))
sns.scatterplot(data=df, x='runtime_minutes', y='averagerating', alpha=0.6)
plt.title('Runtime vs. Average Rating')
plt.xlabel('Runtime (minutes)')
plt.ylabel('Average IMDb Rating')
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



## Genres with Mean Gross > \$100M (Numbered X-axis)

To improve readability, genres are represented by numbers on the x-axis. A separate mapping dictionary shows the number-to-genre correspondence.

```

In [6]: # Filtered data for mean gross > $100M
filtered_mean = genre_gross[genre_gross['mean'] > 100_000_000]
genre_mapping_mean = {i: genre for i, genre in enumerate(filtered_mean.index)}

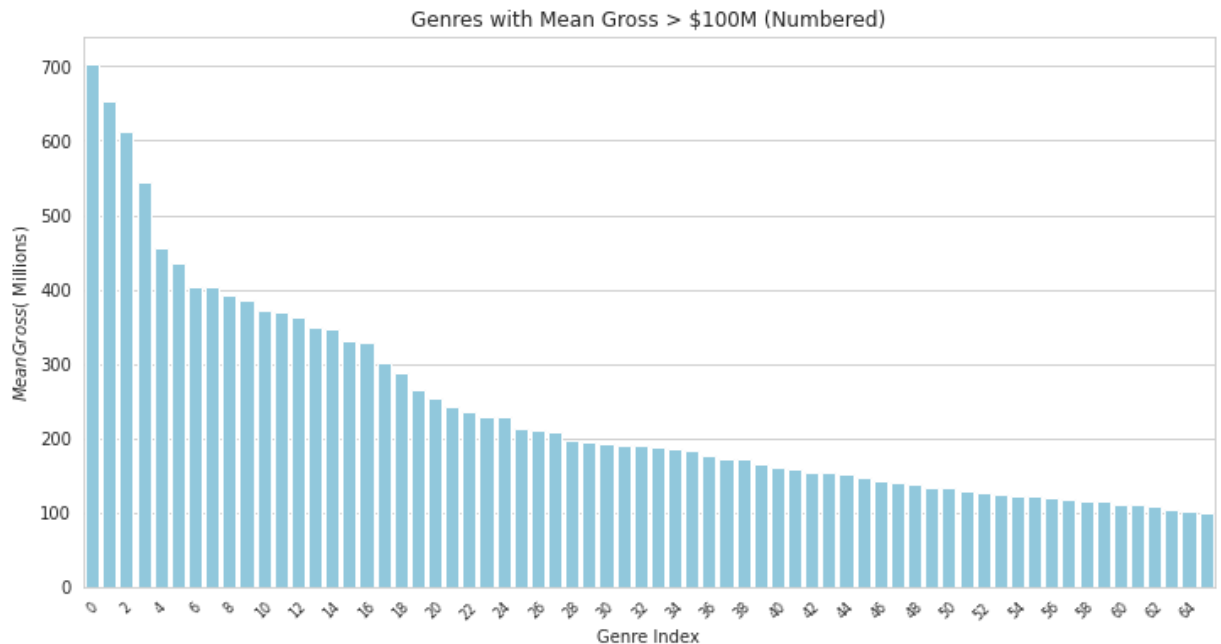
plt.figure(figsize=(12,6))
sns.barplot(x=list(genre_mapping_mean.keys()), y=filtered_mean['mean']/1e6, color='teal')

plt.xticks(ticks=range(0, len(genre_mapping_mean), 2), labels=list(genre_mapping_mean.keys()))
plt.title('Genres with Mean Gross > $100M (Numbered)')
plt.ylabel('$Mean Gross ($ Millions)')
plt.xlabel('Genre Index')
plt.ticklabel_format(style='plain', axis='y')
plt.show()

# print full mapping
# genre_mapping_mean

# Show only top 10 mappings
{k: genre_mapping_mean[k] for k in list(genre_mapping_mean.keys())[:10]}

```



```

Out[6]: {0: 'Adventure,Fantasy',
1: 'Adventure,Drama,Sci-Fi',
2: 'Action,Adventure,Sci-Fi',
3: 'Action,Comedy,Mystery',
4: 'Action,Adventure,Fantasy',
5: 'Biography,Drama,Musical',
6: 'Adventure,Mystery,Sci-Fi',
7: 'Adventure,Animation,Comedy',
8: 'Action,Adventure,Thriller',
9: 'Adventure,Family,Fantasy'}

```

## Genres with Median Gross > \$100M (Numbered X-axis)

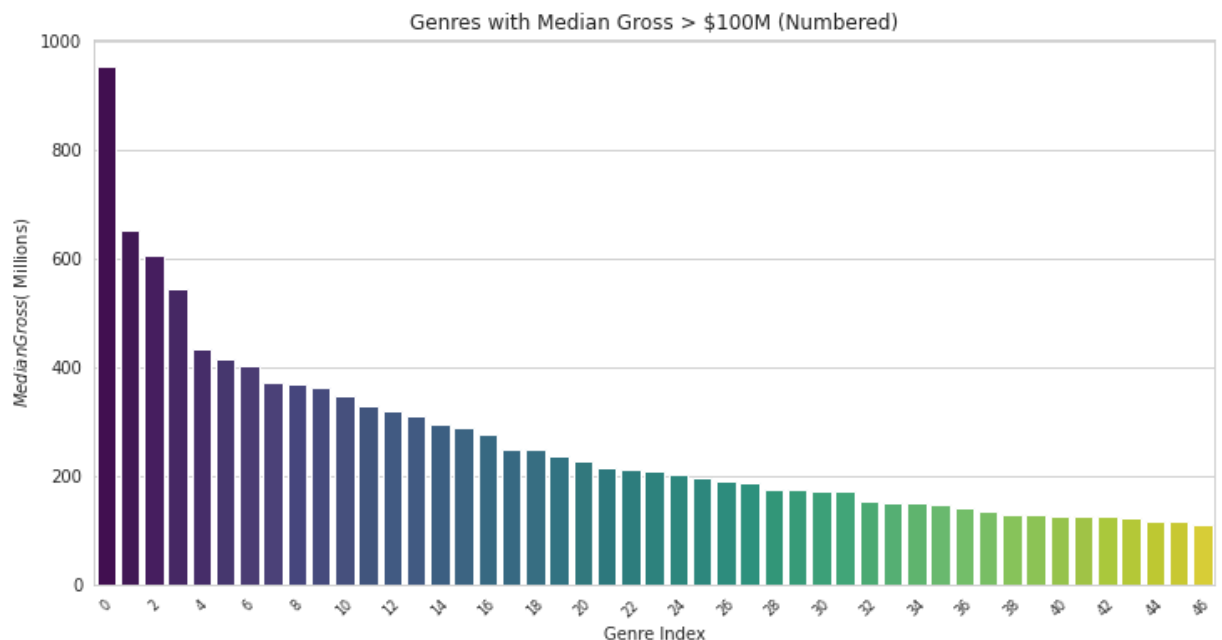
```
In [7]: # Filtered data for median gross > $100M
filtered_median = genre_gross[genre_gross['median'] > 100_000_000].sort_values('median')
genre_mapping_median = {i: genre for i, genre in enumerate(filtered_median.index)}

plt.figure(figsize=(12,6))
sns.barplot(x=list(genre_mapping_median.keys()), y=filtered_median['median']/1e6)

plt.xticks(ticks=range(0, len(genre_mapping_median), 2), labels=list(genre_mapping_median.keys()[::2]))
plt.title('Genres with Median Gross > $100M (Numbered)')
plt.ylabel('$Median Gross ($ Millions)')
plt.xlabel('Genre Index')
plt.ticklabel_format(style='plain', axis='y')
plt.show()

# print full mapping
# genre_mapping_median

# Show only top 10 mappings
{k: genre_mapping_median[k] for k in list(genre_mapping_median.keys())[:10]}
```



```
Out[7]: {0: 'Adventure,Fantasy',
1: 'Adventure,Drama,Sci-Fi',
2: 'Action,Adventure,Sci-Fi',
3: 'Action,Comedy,Mystery',
4: 'Biography,Drama,Musical',
5: 'Animation,Comedy,Family',
6: 'Adventure,Mystery,Sci-Fi',
7: 'Action,Adventure,Animation',
8: 'Action,Sci-Fi',
9: 'Action,Adventure,Fantasy'}
```

## Question 2: Do Ratings Correlate with Revenue?

We use scatter plots to examine the relationship between ratings and total revenue.

```
In [8]: plt.figure(figsize=(10,6))
sns.regplot(data=df, x='averagerating', y='total_gross', scatter_kws={'alpha':0.5})
plt.yscale('log')
plt.title('IMDb Rating vs. Total Gross')
plt.xlabel('Average IMDb Rating')
plt.ylabel('$Total Gross (log scale)')
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```

```
-----
AttributeError                                Traceback (most recent call last)
~/anaconda3/envs/learn-env/lib/python3.8/site-packages/matplotlib/axes/_base.
py in ticklabel_format(self, axis, style, scilimits, useOffset, useLocale, us
eMathText)
    3259             if is_sci_style is not None:
-> 3260                 axis.major.formatter.set_scientific(is_sci_style)
    3261             if scilimits is not None:
```

**AttributeError:** 'LogFormatterSciNotation' object has no attribute 'set\_scientific'

The above exception was the direct cause of the following exception:

```
AttributeError                                Traceback (most recent call last)
<ipython-input-8-3b547f13f000> in <module>
      5 plt.xlabel('Average IMDb Rating')
      6 plt.ylabel('$Total Gross (log scale)')
----> 7 plt.ticklabel_format(style='plain', axis='y')
```

## Question 3: How Do Domestic vs. Foreign Markets Vary by Studio?

We compare studio-level performance across domestic and foreign revenues.

```
In [ ]: studio_revenue = df.groupby('studio')[['domestic_gross', 'foreign_gross']].sum().s
studio_revenue
```



```
In [ ]: # Stacked bar chart: domestic vs. foreign revenue
studio_revenue[['domestic_gross', 'foreign_gross']].div(1e6).plot(kind='bar', stacked=True)
plt.title('Top 10 Studios: Domestic vs. Foreign Gross')
plt.ylabel('$Gross ($ Millions)')
plt.xlabel('Studio')
plt.xticks(rotation=45)
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```