Phase 1 Project Submission

Please fill out:

- Student name: Shilton Soi
- Student pace: part time
- Scheduled project review date/time:
- Instructor name: William Okomba, Noah Kandie, Samuel Mwangi
- Blog post URL: https://github.com/ShiltonTK/phase_1_project.git

---

Overview

Business Problem

Microsoft sees all the big companies creating original video content and they want to get in on the fun. They have decided to create a new movie studio, but they don’t know anything about creating movies. exploring what types of films are currently doing the best at the box office. You must then translate those findings into actionable insights that the head of Microsoft's new movie studio can use to help decide what type of films to create.

Data Understanding & Preparation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/
ipykernel_22566/555797462.py:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major
release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type,
and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at
https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

Getting data from the csv files

```
basics_df =
pd.read_csv('/Users/shilton/Documents/phase_1_project/datasets/imdb.ti
tle.basics.csv')
ratings_df =
pd.read_csv('/Users/shilton/Documents/phase_1_project/datasets/title.r
atings.csv')
gross_df =
pd.read_csv('/Users/shilton/Documents/phase_1_project/datasets/bom.mov
ie_gross.csv')
```

Merging data from the three datasets

```
df = pd.merge(basics_df,ratings_df, on = 'tconst', how = 'right')
df.head(5)
```

```
       tconst            primary_title           original_title
start_year  \
0  tt10356526         Laiye Je Yaarian         Laiye Je Yaarian
2019
1  tt10384606               Borderless               Borderless
2019
2   tt1042974                Just Inès                Just Inès
2010
3   tt1043726  The Legend of Hercules  The Legend of Hercules
2014
4   tt1060240                Até Onde?                Até Onde?
2011

   runtime_minutes                    genres  averagerating  numvotes

0            117.0                   Romance            8.3        31

1             87.0               Documentary            8.9       559

2             90.0                     Drama            6.4        20

3             99.0  Action,Adventure,Fantasy            4.2     50352

4             73.0         Mystery,Thriller            6.5        21
```

```
df.shape
```

```
(73856, 8)
```

```
merged_df = pd.merge(df,gross_df, on = 'primary_title', how = 'right')
merged_df.head(5)
```

```
      tconst                               primary_title  \
0  tt0435761                                  Toy Story 3
1        NaN                   Alice in Wonderland (2010)
```

```
2        NaN  Harry Potter and the Deathly Hallows Part 1
3  tt1375666                                    Inception
4  tt0892791                           Shrek Forever After

       original_title  start_year  runtime_minutes  \
0          Toy Story 3      2010.0            103.0
1                  NaN         NaN              NaN
2                  NaN         NaN              NaN
3            Inception      2010.0            148.0
4  Shrek Forever After      2010.0             93.0

                       genres  averagerating   numvotes studio  \
0  Adventure,Animation,Comedy            8.3   682218.0     BV
1                         NaN            NaN        NaN     BV
2                         NaN            NaN        NaN     WB
3     Action,Adventure,Sci-Fi            8.8  1841066.0     WB
4  Adventure,Animation,Comedy            6.3   167532.0   P/DW

   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
```

```
merged_df.shape
```

```
(3815, 12)
```

```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3815 entries, 0 to 3814
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   tconst           3025 non-null   object
 1   primary_title    3815 non-null   object
 2   original_title   3025 non-null   object
 3   start_year       3025 non-null   float64
 4   runtime_minutes  2978 non-null   float64
 5   genres           3018 non-null   object
 6   averagerating    3025 non-null   float64
 7   numvotes         3025 non-null   float64
 8   studio           3810 non-null   object
 9   domestic_gross   3782 non-null   float64
 10  foreign_gross    2311 non-null   object
 11  year             3815 non-null   int64
dtypes: float64(5), int64(1), object(6)
memory usage: 357.8+ KB
```

```
merged_df.describe()
```

```
        start_year  runtime_minutes  averagerating      numvotes  \
count  3025.000000      2978.000000    3025.000000  3.025000e+03
mean   2013.783140       107.225991       6.458612  6.173183e+04
std       2.466558        20.077436       1.011553  1.255487e+05
min    2010.000000         3.000000       1.600000  5.000000e+00
25%    2012.000000        94.000000       5.900000  2.113000e+03
50%    2014.000000       105.000000       6.600000  1.310900e+04
75%    2016.000000       118.000000       7.100000  6.294200e+04
max    2019.000000       272.000000       9.200000  1.841066e+06

       domestic_gross         year
count    3.782000e+03  3815.000000
mean     2.872771e+07  2013.987418
std      6.619343e+07     2.488221
min      1.000000e+02  2010.000000
25%      1.210000e+05  2012.000000
50%      1.450000e+06  2014.000000
75%      2.900000e+07  2016.000000
max      9.367000e+08  2018.000000
```

Data Cleaning

Removing unnecessary columns

```
merged_df.drop(columns = 'original_title', inplace = True)

merged_df.drop(columns = 'start_year', inplace = True)

merged_df.drop(columns = 'foreign_gross', inplace = True)
```

I will be using domestic gross, therefore I will not require the foreign gross column

Checking for and dropping missing values

```
merged_df.isna().sum()
```

```
tconst             790
primary_title        0
runtime_minutes    837
genres             797
averagerating      790
numvotes           790
studio               5
domestic_gross      33
year                 0
dtype: int64
```

```
merged_df.dropna(subset = ['averagerating', 'domestic_gross',
'genres', 'studio', 'runtime_minutes'], inplace = True)
merged_df.isna().sum()
```

```
tconst             0
primary_title      0
runtime_minutes    0
genres             0
averagerating      0
numvotes           0
studio             0
domestic_gross     0
year               0
dtype: int64
```

```
merged_df.shape
```

```
(2950, 9)
```

Checking for duplicates

```
df.duplicated().sum()
```

```
0
```

No duplicates found

Feature Engineering

Defining the rating scale

- 0 - 1.9: Poor
- 2 - 3.9: Fair
- 4 - 5.9: Average
- 6 - 7.9: Good
- 8 - 10: Excellent

```
labels = ["Poor", "Fair", "Average", "Good", "Excellent"]

merged_df["rating"] = pd.cut(merged_df.averagerating, (0, 2, 4, 6, 8,
10),  labels=labels)
merged_df['rating'].value_counts()
```

```
rating
Good         2031
Average       762
Excellent      92
Fair           60
Poor            5
Name: count, dtype: int64
```

Defining film type by runtime

- 0 - 39: Short film
- 40 - 59: Featurette
- 60 - 129: Standard feature
- 120 - 179: Long film
- 180 - 280: Extended

```
labels = ["Short film", "Featurette", "Standard feature", "Long film",
"Extended"]

merged_df["film_type"] = pd.cut(merged_df.runtime_minutes, (0, 39, 59,
129, 179, 280),  labels=labels)
merged_df['film_type'].value_counts()
```

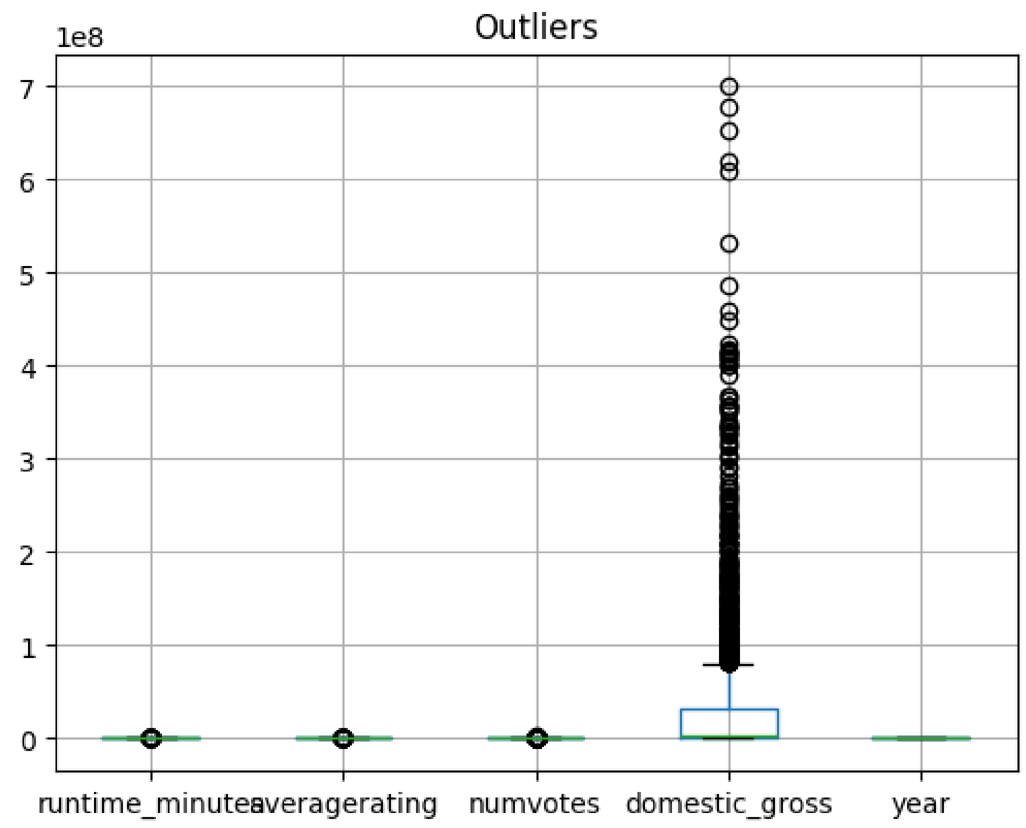
```
film_type
Standard feature    2530
Long film            381
Featurette            28
Extended               9
Short film             2
Name: count, dtype: int64
```

Outlier Handling

Checking for outliers

```
merged_df.boxplot()
plt.title('Outliers')
```

```
Text(0.5, 1.0, 'Outliers')
```

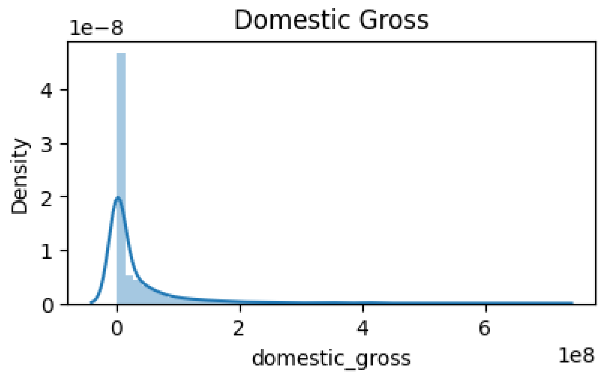


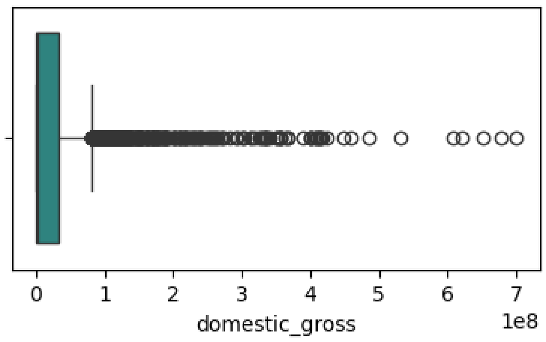
distribution plots for the domestic gross column

```
plt.figure(figsize=(10,5))
plt.subplot(2,2,1)
sns.distplot(merged_df['domestic_gross'])
plt.title('Domestic Gross')
plt.subplot(2,2,2)
sns.boxplot(merged_df['domestic_gross'], orient = 'h', palette =
'viridis')

plt.show()
```

```
/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/
ipykernel_22566/2434905955.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
  sns.distplot(merged_df['domestic_gross'])
/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/ipykernel_22566/24349
05955.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(merged_df['domestic_gross'], orient = 'h', palette =
'viridis')
```





Z score

setting upper and lower limit

```
ugross = merged_df['domestic_gross'].mean() +
3*merged_df['domestic_gross'].std()
print("Upper limit gross =",ugross)
lgross = merged_df['domestic_gross'].mean() -
3*merged_df['domestic_gross'].std()
print("Lower limit gross =",lgross)

Upper limit gross = 232024129.97461376
Lower limit gross = -170622003.6139358
```

Entries outside the interquartile range

```
merged_df[merged_df['domestic_gross'] > ugross].head(10)

        tconst                                                primary_title
runtime_minutes  \
0    tt0435761                                                   Toy Story 3
103.0
3    tt1375666                                                     Inception
148.0
4    tt0892791                                           Shrek Forever After
```

```
93.0
5    tt1325004                     The Twilight Saga: Eclipse
124.0
6    tt1228705                                      Iron Man 2
124.0
8    tt1323594                                   Despicable Me
95.0
372  tt1399103                 Transformers: Dark of the Moon
154.0
373  tt1298650  Pirates of the Caribbean: On Stranger Tides
136.0
378  tt1411697                           The Hangover Part II
102.0
822  tt1074638                                         Skyfall
143.0

                         genres  averagerating   numvotes studio  \
0    Adventure,Animation,Comedy            8.3   682218.0     BV
3       Action,Adventure,Sci-Fi            8.8  1841066.0     WB
4    Adventure,Animation,Comedy            6.3   167532.0   P/DW
5       Adventure,Drama,Fantasy            5.0   211733.0   Sum.
6       Action,Adventure,Sci-Fi            7.0   657690.0   Par.
8       Animation,Comedy,Family            7.7   464511.0   Uni.
372     Action,Adventure,Sci-Fi            6.2   366409.0   P/DW
373    Action,Adventure,Fantasy            6.6   447624.0     BV
378              Comedy,Mystery            6.5   432800.0     WB
822   Action,Adventure,Thriller            7.8   592221.0   Sony

     domestic_gross  year     rating         film_type
0       415000000.0  2010  Excellent  Standard feature
3       292600000.0  2010  Excellent         Long film
4       238700000.0  2010       Good  Standard feature
5       300500000.0  2010    Average  Standard feature
6       312400000.0  2010       Good  Standard feature
8       251500000.0  2010       Good  Standard feature
372     352400000.0  2011       Good         Long film
373     241100000.0  2011       Good         Long film
378     254500000.0  2011       Good  Standard feature
822     304400000.0  2012       Good         Long film
```

Dropping outliers

```
merged_df = merged_df.loc[(merged_df["domestic_gross"] <= ugross) &
(merged_df["domestic_gross"] >=lgross)]

largest_value = merged_df['domestic_gross'].max()
least_value = merged_df['domestic_gross'].min()
```

```
print(largest_value)
print(least_value)

229000000.0
100.0
```

Explorer Data Analysis

Selecting the top 10 genres by average rating

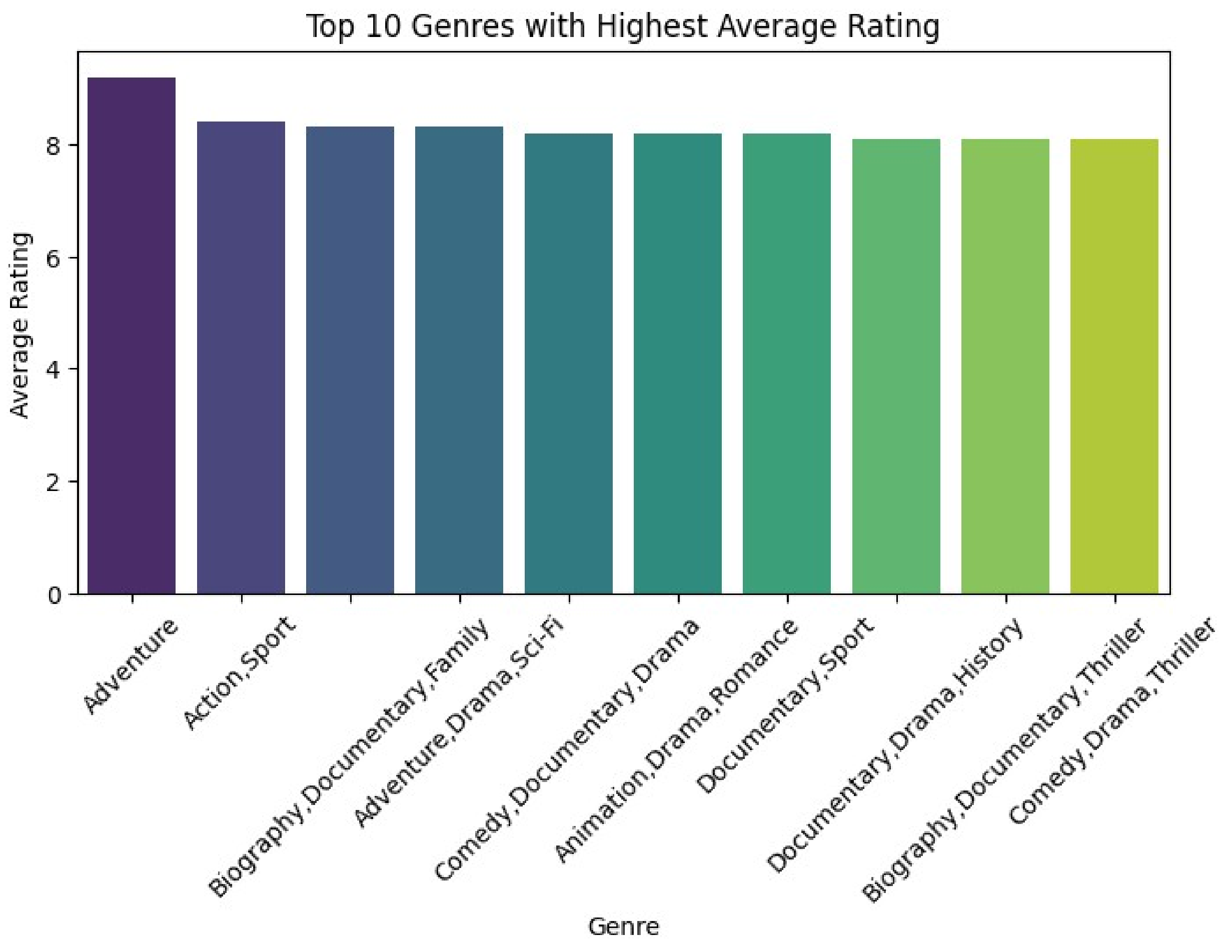
```
# Group by Genre and calculate the average rating
groupmerged_df = merged_df.groupby('genres')
['averagerating'].mean().reset_index()

# Sort by average rating in descending order
sortgroupmerged_df = groupmerged_df.sort_values(by='averagerating',
ascending=False)

# Select the top 10 genres
top_10_genres = sortgroupmerged_df.head(10)
top_10_genres
```

```
                               genres  averagerating
76                          Adventure            9.2
74                       Action,Sport            8.4
143     Biography,Documentary,Family            8.3
104             Adventure,Drama,Sci-Fi            8.3
170          Comedy,Documentary,Drama            8.2
130           Animation,Drama,Romance            8.2
241                 Documentary,Sport            8.2
227         Documentary,Drama,History            8.1
148   Biography,Documentary,Thriller            8.1
181             Comedy,Drama,Thriller            8.1
```

```
plt.figure(figsize=(8, 4))
sns.barplot(x='genres', y='averagerating', data=top_10_genres,
palette='viridis')
plt.xlabel('Genre')
plt.ylabel('Average Rating')
plt.title('Top 10 Genres with Highest Average Rating')
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.show()
```

```
/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/
ipykernel_22566/3216531453.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.
```

```
  sns.barplot(x='genres', y='averagerating', data=top_10_genres,
palette='viridis')
```



Selecting the top 10 genres by average gross

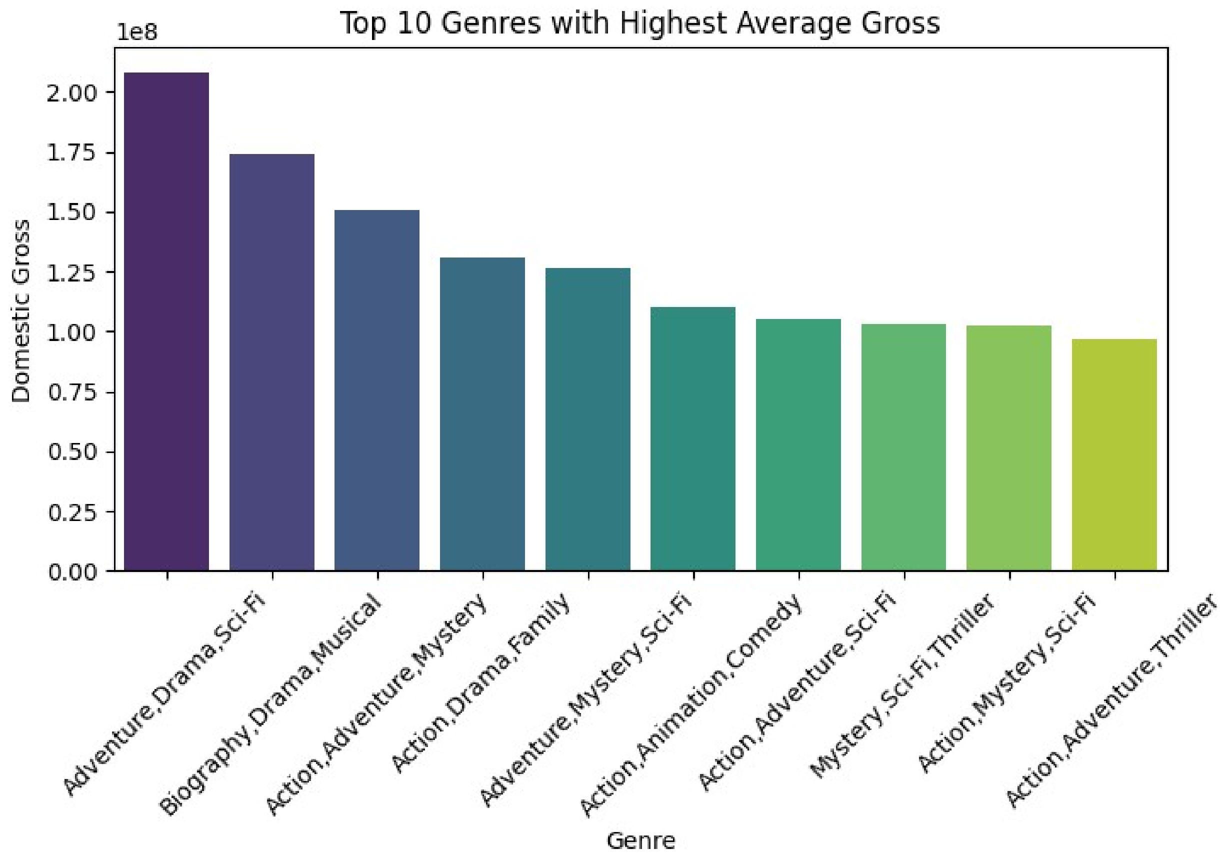
```
# Group by Genre and calculate the average gross
groupmergedgross_df = merged_df.groupby('genres')
['domestic_gross'].mean().reset_index()

# Sort by average gross in descending order
sortgroupmergedgross_df =
groupmergedgross_df.sort_values(by='domestic_gross', ascending=False)

# Select the top 10 genres
top_10_genres_gross = sortgroupmergedgross_df.head(10)
top_10_genres_gross

                        genres  domestic_gross
104     Adventure,Drama,Sci-Fi    2.082000e+08
154    Biography,Drama,Musical    1.743000e+08
10    Action,Adventure,Mystery    1.509000e+08
```

```
45          Action,Drama,Family    1.310500e+08
114   Adventure,Mystery,Sci-Fi    1.265000e+08
15     Action,Animation,Comedy    1.099333e+08
11     Action,Adventure,Sci-Fi    1.050885e+08
307    Mystery,Sci-Fi,Thriller    1.031000e+08
68       Action,Mystery,Sci-Fi    1.024000e+08
12   Action,Adventure,Thriller    9.671238e+07
```

Bar graph for top 10 genres with highest average gross

```
plt.figure(figsize=(8, 4))
sns.barplot(x='genres', y='domestic_gross', data=top_10_genres_gross,
palette='viridis')
plt.xlabel('Genre')
plt.ylabel('Domestic Gross')
plt.title('Top 10 Genres with Highest Average Gross')
plt.xticks(rotation=45)  # Rotate x-axis labels for better visibility
plt.show()
```

```
/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/
ipykernel_22566/2900559918.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='genres', y='domestic_gross',
data=top_10_genres_gross, palette='viridis')
```



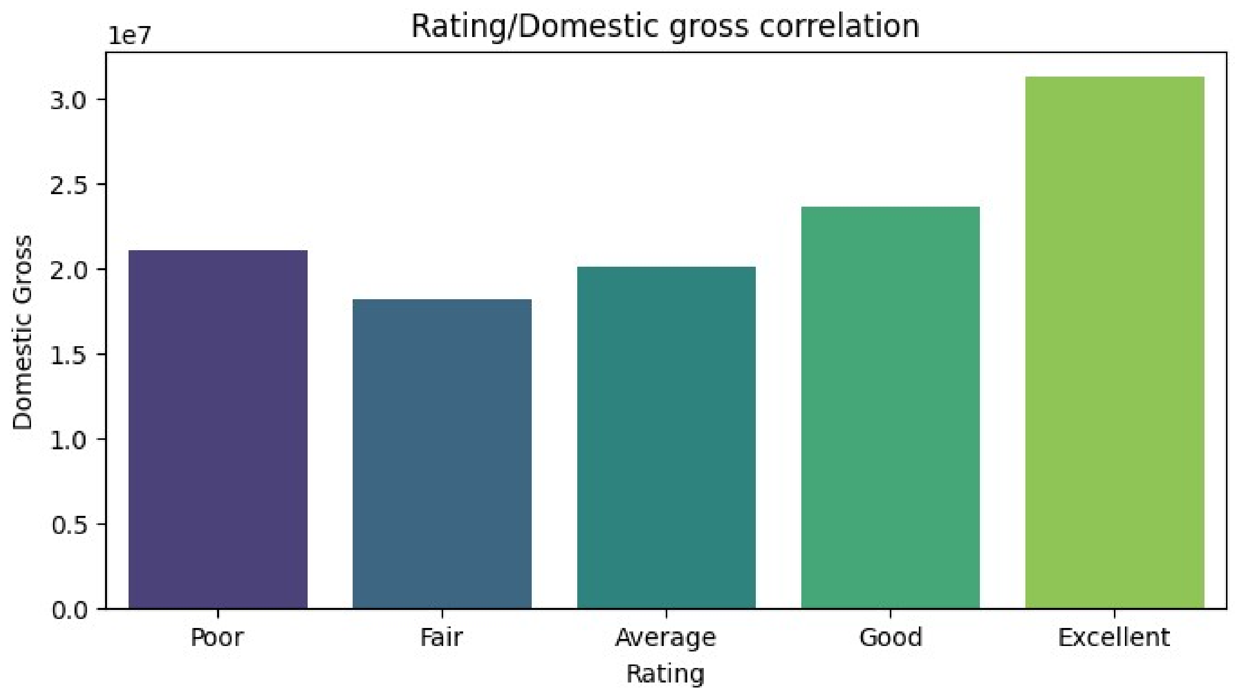
Average gross across the ratings

```
# Group by rating and calculate the average gross
grouprg_df = merged_df.groupby('rating')
['domestic_gross'].mean().reset_index()

# Sort by average gross in descending order
sortgrouprg_df = grouprg_df.sort_values(by='domestic_gross',
ascending=False)

sortgrouprg_df
```

```
/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/
ipykernel_22566/852649187.py:2: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  grouprg_df = merged_df.groupby('rating')
['domestic_gross'].mean().reset_index()
```

```
      rating  domestic_gross
4  Excellent    3.123855e+07
3       Good    2.358553e+07
0       Poor    2.107500e+07
```

```
2    Average    2.006543e+07
1       Fair    1.822048e+07

plt.figure(figsize=(8, 4))
sns.barplot(x='rating', y='domestic_gross', data=sortgrouprg_df,
palette='viridis')
plt.xlabel('Rating')
plt.ylabel('Domestic Gross')
plt.title('Rating/Domestic gross correlation')
plt.show()

/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/
ipykernel_22566/3583766134.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='rating', y='domestic_gross', data=sortgrouprg_df,
palette='viridis')
```



Top 5 Studios domestic gross

```
# Group by studio and calculate the average gross
groupstudio_df = merged_df.groupby('studio')
['domestic_gross'].mean().reset_index()

# Sort by average gross in descending order
```

```
sortgroupstudio_df = groupstudio_df.sort_values(by='domestic_gross',
ascending=False)

# Select top 5 studios
top_5_studio_gross = sortgroupstudio_df.head(5)
top_5_studio_gross

    studio  domestic_gross
144   P/DW    1.364750e+08
32      BV    8.497121e+07
119    MGM    8.300000e+07
153   Par.    7.751509e+07
81     Fox    7.498571e+07
```

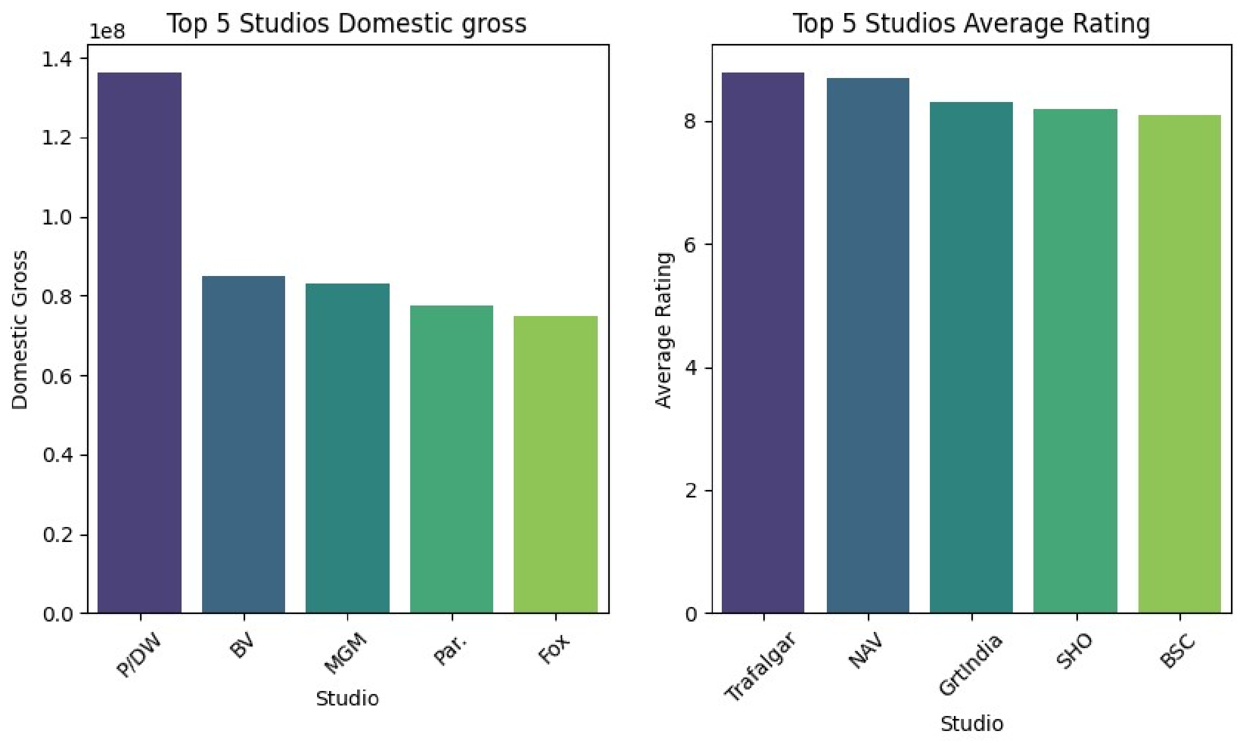
Top 5 Studios rating

```
# Group by studio and calculate the average rating
groupstudiorr_df = merged_df.groupby('studio')
['averagerating'].mean().reset_index()

# Sort by  average rating in descending order
sortgroupstudiorr_df =
groupstudiorr_df.sort_values(by='averagerating', ascending=False)

#Select top 5 studios
top_5_studio_rating = sortgroupstudiorr_df.head(5)
top_5_studio_rating

        studio  averagerating
188  Trafalgar            8.8
130        NAV            8.7
93    GrtIndia            8.3
170        SHO            8.2
30         BSC            8.1
```

Plottting graphs for studio rating and gross

```
plt.figure(figsize=(10, 5))
plt.subplot(1,2,1)
sns.barplot(x='studio', y='domestic_gross', data=top_5_studio_gross,
palette='viridis')
plt.xlabel('Studio')
plt.ylabel('Domestic Gross')
plt.title('Top 5 Studios Domestic gross')
plt.xticks(rotation=45)
plt.subplot(1,2,2)
sns.barplot(x='studio', y='averagerating', data=top_5_studio_rating,
palette='viridis')
plt.xlabel('Studio')
plt.ylabel('Average Rating')
plt.title('Top 5 Studios Average Rating')
plt.xticks(rotation=45)
plt.show()
```

```
/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/
ipykernel_22566/2074911572.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='studio', y='domestic_gross', data=top_5_studio_gross,
palette='viridis')
/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/ipykernel_22566/20749
11572.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='studio', y='averagerating', data=top_5_studio_rating,
palette='viridis')
```



Film Type performance in terms of rating and gross

```
# Group by runtime and calculate the average rating
groupruntime_df = merged_df.groupby('film_type')
```

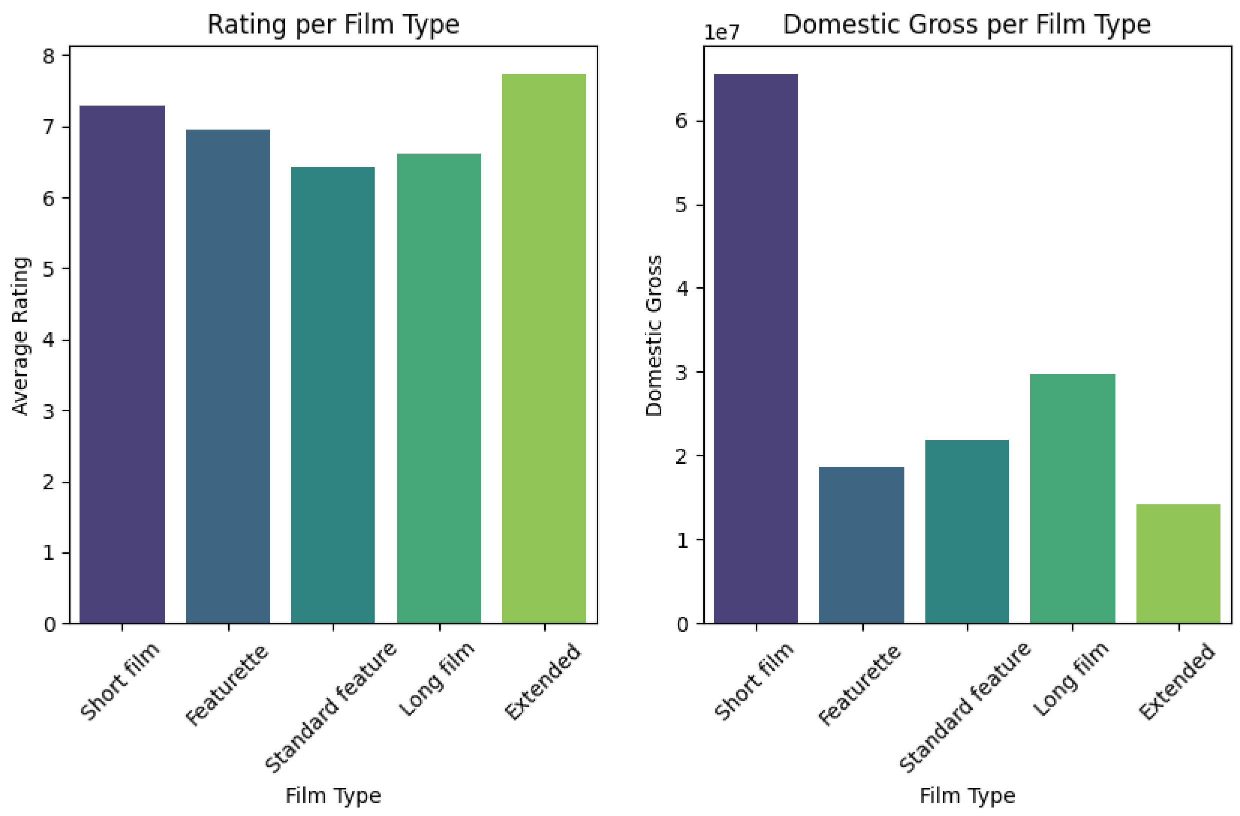
```
['averagerating'].mean().reset_index()

# Sort by  average rating in descending order
sortgroupruntime_df = groupruntime_df.sort_values(by='averagerating',
ascending=False)
sortgroupruntime_df

/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/
ipykernel_22566/1657010974.py:2: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  groupruntime_df = merged_df.groupby('film_type')
['averagerating'].mean().reset_index()

          film_type  averagerating
4          Extended       7.744444
0        Short film       7.300000
1        Featurette       6.948148
3         Long film       6.611207
2  Standard feature       6.416038

# Group by runtime and calculate the average gross
groupruntimegr_df = merged_df.groupby('film_type')
['domestic_gross'].mean().reset_index()

# Sort by  average gross in descending order
sortgroupruntimegr_df =
groupruntimegr_df.sort_values(by='domestic_gross', ascending=False)
sortgroupruntimegr_df

/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/
ipykernel_22566/3901180404.py:2: FutureWarning: The default of
observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  groupruntimegr_df = merged_df.groupby('film_type')
['domestic_gross'].mean().reset_index()

          film_type  domestic_gross
0        Short film    6.555000e+07
3         Long film    2.975962e+07
2  Standard feature    2.184090e+07
1        Featurette    1.860430e+07
4          Extended    1.420819e+07
```

Plotting graphs for film type rating and gross

```
plt.figure(figsize=(10, 5))
plt.subplot(1,2,1)
```

```python
sns.barplot(x='film_type', y='averagerating',
data=sortgroupruntime_df, palette='viridis')
plt.xlabel('Film Type')
plt.ylabel('Average Rating')
plt.title('Rating per Film Type')
plt.xticks(rotation=45)
plt.subplot(1,2,2)
sns.barplot(x='film_type', y='domestic_gross',
data=sortgroupruntimegr_df, palette='viridis')
plt.xlabel('Film Type')
plt.ylabel('Domestic Gross')
plt.title('Domestic Gross per Film Type')
plt.xticks(rotation=45)
plt.show()
```

```
/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/
ipykernel_22566/41428131.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='film_type', y='averagerating',
data=sortgroupruntime_df, palette='viridis')
/var/folders/41/82wcd3b12yvd24mnnbgvs2900000gn/T/ipykernel_22566/41428
131.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='film_type', y='domestic_gross',
data=sortgrouprutimegr_df, palette='viridis')
```


Rating per Film Type
Average Rating
Film Type
Short film
Featurette
Standard feature
Long film
Extended
Domestic Gross per Film Type
1e7
Domestic Gross