

THE TMDB MOVIES DATA CSV DATA SET.

DATA UNDERSTANDING .

```
In [152]: # Importing and reading the data.
df = pd.read_csv("tmdb.movies.csv.gz", index_col = 0)
df.head()
```

```
Out[152]:
```

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_ave
0	[12, 14, 10751]	12444	en	Harry Potter and the Deathly Hallows: Part 1	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	
2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	

```
In [153]: # Check the data types and number of rows and columns
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26517 entries, 0 to 26516
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   genre_ids             26517 non-null  object
1   id                    26517 non-null  int64
2   original_language     26517 non-null  object
3   original_title        26517 non-null  object
4   popularity            26517 non-null  float64
5   release_date          26517 non-null  object
6   title                 26517 non-null  object
7   vote_average          26517 non-null  float64
8   vote_count            26517 non-null  int64
dtypes: float64(2), int64(2), object(5)
memory usage: 2.0+ MB
None
```

This data consists of float, integer and object as data types.

```
In [154]: #finding the shape of the data
df.shape
```

```
Out[154]: (26517, 9)
```

This data consists of 26517 rows and 9 columns.

```
In [155]: # Check the summary statistics of each column
print(df.describe())
```

	id	popularity	vote_average	vote_count
count	26517.000000	26517.000000	26517.000000	26517.000000
mean	295050.153260	3.130912	5.991281	194.224837
std	153661.615648	4.355229	1.852946	960.961095
min	27.000000	0.600000	0.000000	1.000000
25%	157851.000000	0.600000	5.000000	2.000000
50%	309581.000000	1.374000	6.000000	5.000000
75%	419542.000000	3.694000	7.000000	28.000000
max	608444.000000	80.773000	10.000000	22186.000000

The table shows summary statistics for four variables - id, popularity, vote_average, and vote_count - with a sample size of 26,517.

count: The count column shows the number of non-missing values for each variable. In this case, there are 26,517 non-missing values for each variable.

mean: The mean column shows the average value for each variable. For example, the mean value for the "popularity" variable is 3.130912. This means that, on average, the movies in the dataset have a popularity score of 3.130912.

std: The std column shows the standard deviation of the values for each variable. For example, the standard deviation of the "popularity" variable is 4.355229. This means that the popularity scores for movies in the dataset vary widely, with some movies having very high popularity scores and others having very low popularity scores.

min: The min column shows the minimum value for each variable. For example, the minimum value for the "vote_average" variable is 0. This means that there are movies in the dataset with a vote average score of 0.

25%, 50%, 75%: These columns show the quartiles of the distribution of values for each variable. For example, the 25th percentile of the "vote_count" variable is 2, which means that 25% of the movies in the dataset have a vote count of 2 or less. Similarly, the 50th percentile (median) of the "popularity" variable is 1.374, which means that 50% of the movies in the dataset have a popularity score of 1.374 or less.

max: The max column shows the maximum value for each variable. For example, the maximum value for the "vote_count" variable is 22,186. This means that there is at least one movie in the dataset with a very high vote count.

DATA CLEANING.

```
In [156]: #checking for null values in the data.  
print(df.isnull().sum())
```

```
genre_ids      0  
id             0  
original_language  0  
original_title  0  
popularity     0  
release_date   0  
title          0  
vote_average   0  
vote_count     0  
dtype: int64
```

The data does not contain any null values.

```
In [157]: # Check for duplicates  
print(df.duplicated().value_counts())
```

```
False    25497  
True      1020  
dtype: int64
```

The table shows the count of two categories - False and True - for a variable. There are 25,497 values of False and 1,020 values of True for the variable in the dataset. The data has 1020 duplicated values.

```
In [158]: #viewing the duplicated values by title.
df[df.duplicated(keep = False)].sort_values(by = 'title')
```

Out[158]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vot
9191	[99]	95383	en	\$ellebrity	1.420	2013-01-11	\$ellebrity	
6315	[99]	95383	en	\$ellebrity	1.420	2013-01-11	\$ellebrity	
20070	[99, 36, 10770]	430364	en	'85: The Greatest Team in Pro Football History	0.600	2018-01-29	'85: The Greatest Team in Pro Football History	
26340	[99, 36, 10770]	430364	en	'85: The Greatest Team in Pro Football History	0.600	2018-01-29	'85: The Greatest Team in Pro Football History	
18016	[18, 10749]	416691	en	1 Night	5.409	2017-02-10	1 Night	
...
21273	[18]	326382	es	Zama	5.671	2017-09-30	Zama	
15061	[10751, 16]	94196	fr	Zarafa	2.705	2012-11-11	Zarafa	
5888	[10751, 16]	94196	fr	Zarafa	2.705	2012-11-11	Zarafa	
25188	[10752, 10751, 36]	472553	en	Zoo	2.550	2018-06-08	Zoo	
21676	[10752, 10751, 36]	472553	en	Zoo	2.550	2018-06-08	Zoo	

2016 rows × 9 columns



```
In [159]: #viewing the duplicated values by id.
duplicates = df[df.duplicated(subset=['id'], keep=False)]
duplicates
```

Out[159]:

	genre_ids	id	original_language	original_title	popularity	release_date	title \
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story
10	[16, 35, 10751]	863	en	Toy Story 2	22.698	1999-11-24	Toy Story 2
43	[35, 10749]	239	en	Some Like It Hot	14.200	1959-03-18	Some Like It Hot
54	[12, 28, 878]	20526	en	TRON: Legacy	13.459	2010-12-10	TRON: Legacy
56	[35, 16, 10751]	9994	en	The Great Mouse Detective	13.348	1986-07-02	The Great Mouse Detective
...
26481	[35, 18]	270805	en	Summer League	0.600	2013-03-18	Summer League
26485	[27, 53]	453259	en	Devils in the Darkness	0.600	2013-05-15	Devils in the Darkness
26504	[27, 35, 27]	534282	en	Head	0.600	2015-03-28	Head
26510	[99]	495045	en	Fail State	0.600	2018-10-19	Fail State
26511	[99]	492837	en	Making Filmmakers	0.600	2018-04-07	Making Filmmakers

2016 rows × 9 columns



```
In [160]: #sorting the duplicated values by id.
duplicates_sorted = duplicates.sort_values(['id'])
duplicates_sorted
```

Out[160]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	v
20626	[16, 10751, 14]	129	ja	千と千尋の神隠し	32.043	2002-09-20	Spirited Away	
14173	[16, 10751, 14]	129	ja	千と千尋の神隠し	32.043	2002-09-20	Spirited Away	
43	[35, 10749]	239	en	Some Like It Hot	14.200	1959-03-18	Some Like It Hot	
24000	[35, 10749]	239	en	Some Like It Hot	14.200	1959-03-18	Some Like It Hot	
20639	[28, 53, 878]	280	en	Terminator 2: Judgment Day	24.604	1991-07-03	Terminator 2: Judgment Day	
...
17071	[27]	560717	en	Requiem	0.600	2015-01-01	Requiem	
23685	[35]	564441	en	Adopting Trouble	0.600	2016-04-08	Adopting Trouble	
20461	[35]	564441	en	Adopting Trouble	0.600	2016-04-08	Adopting Trouble	
23785	[99]	572012	en	Harvested Alive - 10 Years of Investigations	0.600	2016-11-28	Harvested Alive	
20569	[99]	572012	en	Harvested Alive - 10 Years of Investigations	0.600	2016-11-28	Harvested Alive	

2016 rows × 9 columns



```
In [161]: #dropping the duplicates.
df.drop_duplicates(keep = "first", inplace = True)
```

```
In [162]: #checking to see if we have any remaining duplicates.
print(df.duplicated().value_counts())
```

```
False    25497
dtype: int64
```

This data set no longer contains any duplicated values.

In [163]: *# Get information about the data types and non-null values in each column*
df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25497 entries, 0 to 26516
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   genre_ids              25497 non-null  object
1   id                     25497 non-null  int64
2   original_language      25497 non-null  object
3   original_title          25497 non-null  object
4   popularity              25497 non-null  float64
5   release_date            25497 non-null  object
6   title                   25497 non-null  object
7   vote_average            25497 non-null  float64
8   vote_count              25497 non-null  int64
dtypes: float64(2), int64(2), object(5)
memory usage: 1.9+ MB
```

In [164]: *#dropping the original_language and original_title columns from the data frame*
df= df.drop(['original_language','original_title'], axis=1)

In [165]: df.shape

Out[165]: (25497, 7)

The data frame now has fewer columns = 7.

In [166]: df['release_date'] = pd.to_datetime(df['release_date'])
df['year'] = df['release_date'].dt.year
df['month'] = df['release_date'].dt.month
df['day'] = df['release_date'].dt.day

In [167]: df.dtypes

Out[167]: genre_ids object
id int64
popularity float64
release_date datetime64[ns]
title object
vote_average float64
vote_count int64
year int64
month int64
day int64
dtype: object

In [168]: `df.head()`

Out[168]:

	genre_ids	id	popularity	release_date	title	vote_average	vote_count	year	month
0	[12, 14, 10751]	12444	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788	2010	11
1	[14, 12, 16, 10751]	10191	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	2010	3
2	[12, 28, 878]	10138	28.515	2010-05-07	Iron Man 2	6.8	12368	2010	5
3	[16, 35, 10751]	862	28.005	1995-11-22	Toy Story	7.9	10174	1995	11
4	[28, 878, 12]	27205	27.920	2010-07-16	Inception	8.3	22186	2010	7

In [169]: `import warnings`

`warnings.filterwarnings("ignore")`

Code that generates a DeprecationWarning will be ignored.

In [170]: *#Viewing the data.*

`df.head()`

Out[170]:

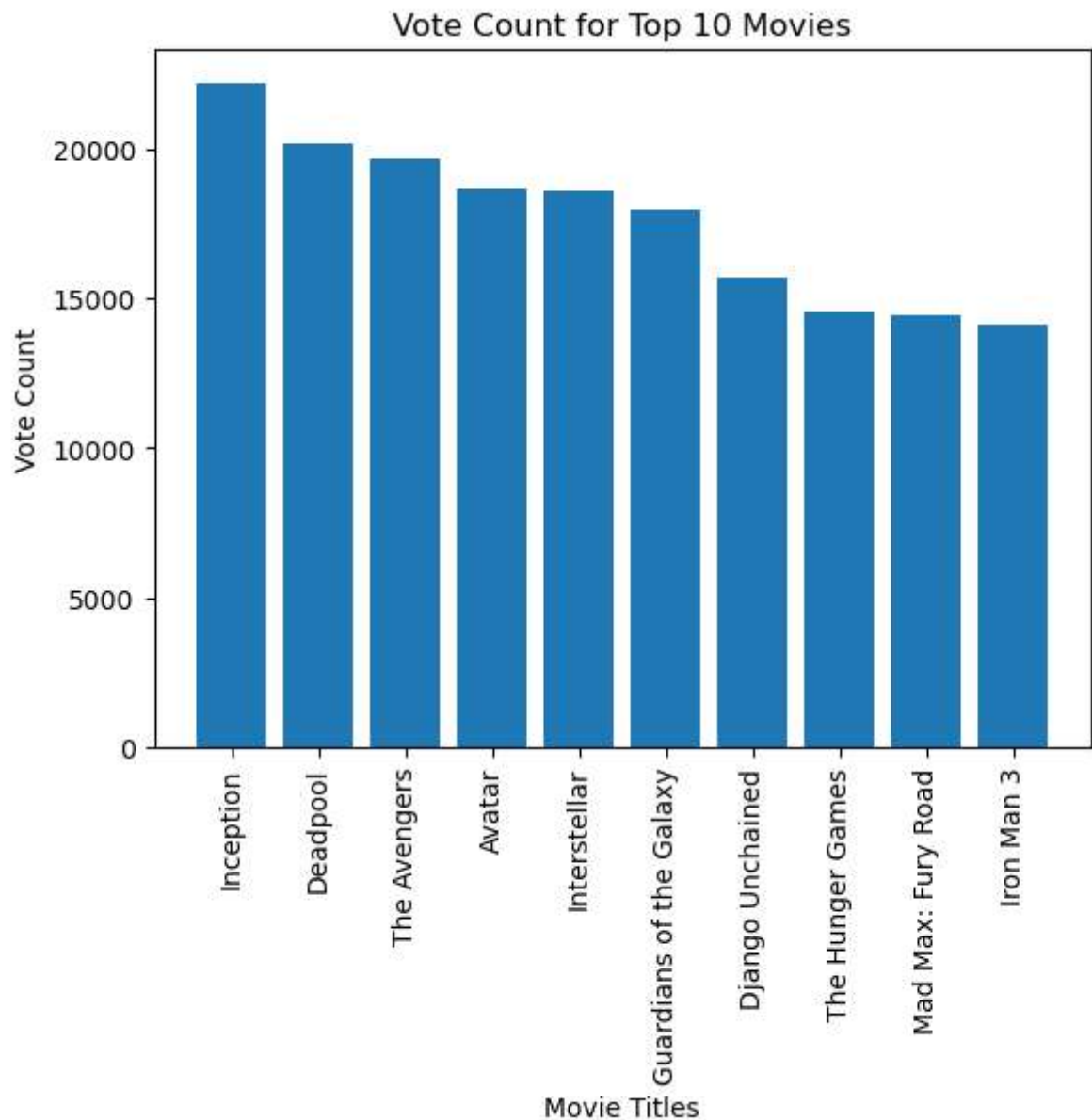
	genre_ids	id	popularity	release_date	title	vote_average	vote_count	year	month
0	[12, 14, 10751]	12444	33.533	2010-11-19	Harry Potter and the Deathly Hallows: Part 1	7.7	10788	2010	11
1	[14, 12, 16, 10751]	10191	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	2010	3
2	[12, 28, 878]	10138	28.515	2010-05-07	Iron Man 2	6.8	12368	2010	5
3	[16, 35, 10751]	862	28.005	1995-11-22	Toy Story	7.9	10174	1995	11
4	[28, 878, 12]	27205	27.920	2010-07-16	Inception	8.3	22186	2010	7

DATA ANALYSIS AND VISUALIZATION.

Top Ten Movies as per the Vote Count.

```
In [171]: # Sort the DataFrame by vote count and select the top 10 movies
top_movies = df.sort_values(by='vote_count', ascending=False).head(10)

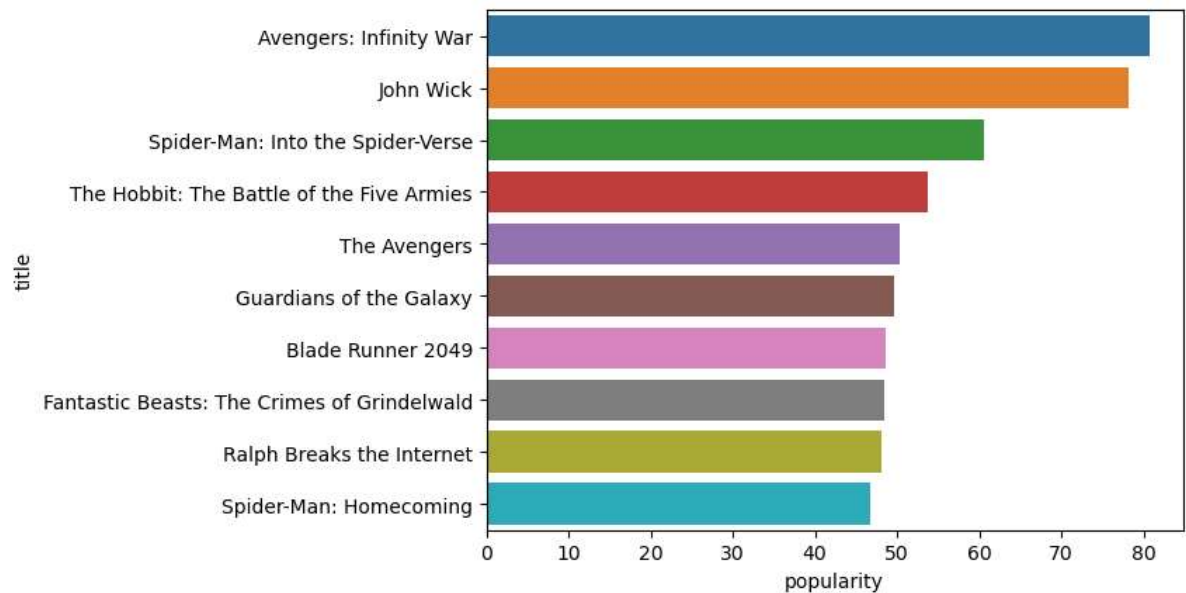
# Create a histogram of the vote count for the top 10 movies
plt.bar(top_movies['title'], top_movies['vote_count'])
plt.xticks(rotation=90)
plt.xlabel('Movie Titles')
plt.ylabel('Vote Count')
plt.title('Vote Count for Top 10 Movies')
plt.show()
```



The bar graph displays the most popular movies as per the vote count. The most popular movie was Inception followed by Deadpool, The Avengers, Avatar, Interstellar, guardians of the Galaxy, Django Unchained, The hunger Games, Mad Max: Fury Road and Iron Man 3 in that order.

Top ten Movies as per Popularity.

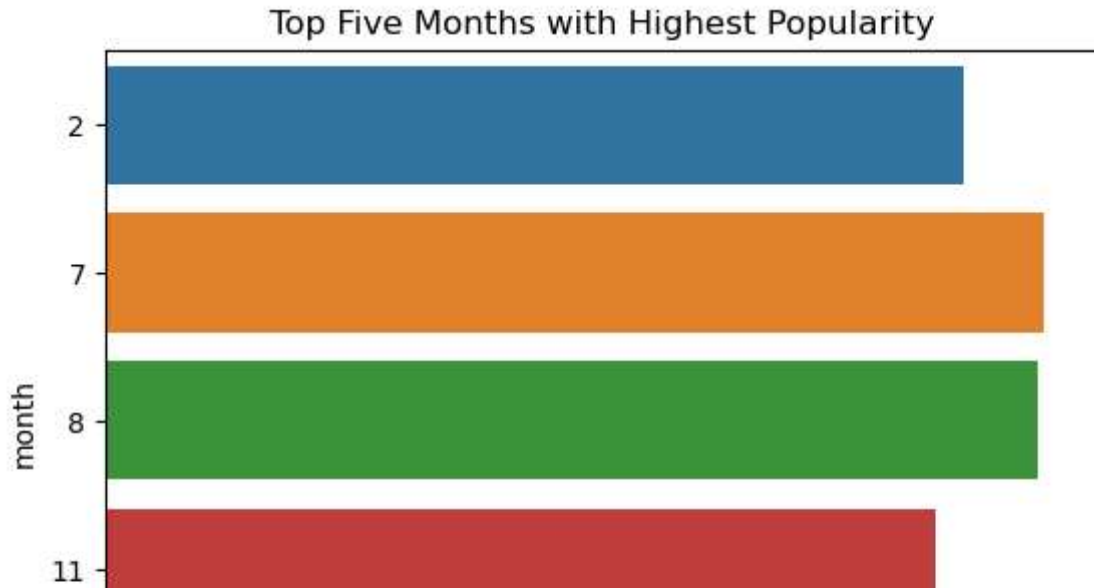
```
In [172]: #plotting bargraph using seaborn of movies vs popularity.  
top_movies2 = df.sort_values(by='popularity', ascending=False).head(10)  
sns.barplot(y='title', x='popularity', data=top_movies2, orient='h');
```



The bar graph displays the most popular movies as per the popularity index.. The most popular movie was the AVengers: infinity War followed by john Wick, Spider_Man: Into the Spider-Verse, The Hobbit: The battle of the Five Armies, Th Avengers, Guardians of the Galaxy, Blade Runner 2049, Fantastic Beasts: The crimes of Grindelwald, Ralph Breaks the Internet and Spider-Man:Homecoming in that order.

```
In [173]: #plotting bar graph displaying top ten month with the highest popularity of movies
top_months = df.groupby('month')['popularity'].mean().reset_index().sort_values(ascending=False)
sns.barplot(y='month', x='popularity', data=top_months, orient='h')
plt.title('Top Five Months with Highest Popularity')
```

```
Out[173]: Text(0.5, 1.0, 'Top Five Months with Highest Popularity')
```



This plot displays the top five months with the highest popularity. Movies released in december were the most popular followed by movies released in July, August, february and November in that order.

FINDINGS.

1. December is the most appropriate time to release new movies as shown by the popularity of movies released in december.
2. The top three most popular movies are:

AVengers: infinity War.

john Wick.

Spider_Man: Into the Spider-Verse.

3. The most voted for movies as per the vote count include the Inception, Deadpool and The Avengers.

RECOMMENDATIONS.

1. Microsoft should major in producing movies in December.
2. Microsoft should produce the following movies:

Avengers: Infinity War.

John Wick.

BOM.MOVIE_GROSS.CSV.gz

DATA UNDERSTANDING.

```
In [174]: # Importing and reading the data.
bom_movies_gross_df = pd.read_csv("bom.movie_gross.csv.gz", index_col = 0)
bom_movies_gross_df.head()
```

Out[174]:

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

```
In [175]: #Previewing the data shape
bom_movies_gross_df.shape
```

Out[175]: (3387, 4)

The data consists of 3387 rows and 4 columns.

In [176]: *#Getting information about the bom_movies_gross*
 bom_movies_gross_df.info()

```
<class 'pandas.core.frame.DataFrame'>
Index: 3387 entries, Toy Story 3 to An Actor Prepares
Data columns (total 4 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   studio          3382 non-null   object
1   domestic_gross  3359 non-null   float64
2   foreign_gross   2037 non-null   object
3   year            3387 non-null   int64
dtypes: float64(1), int64(1), object(2)
memory usage: 132.3+ KB
```

The data consists of object , float and integer data types.

In [177]: *#checking the summary statistics of each column.*
 bom_movies_gross_df.describe()

Out[177]:

	domestic_gross	year
count	3.359000e+03	3387.000000
mean	2.874585e+07	2013.958075
std	6.698250e+07	2.478141
min	1.000000e+02	2010.000000
25%	1.200000e+05	2012.000000
50%	1.400000e+06	2014.000000
75%	2.790000e+07	2016.000000
max	9.367000e+08	2018.000000

This is a summary of a dataset with two columns: domestic_gross and year.

The domestic_gross column represents the total gross earnings of a movie in the domestic (US) market, and the year column represents the year in which the movie was released.

Here are some observations about the summary statistics:

There are 3,359 movies in the dataset. The average domestic gross earnings for a movie in the dataset is approximately \$28.7 million.

The standard deviation of the domestic gross earnings is approximately \$67 million, indicating that the earnings are spread out over a wide range.

The minimum domestic gross earnings for a movie in the dataset is \$100, indicating that there are some very low-grossing movies in the dataset.

The 25th percentile of the domestic gross earnings is \$120,000, indicating that 25% of the movies in the dataset earned less than this amount.

The median (50th percentile) of the domestic gross earnings is \$1.4 million, indicating that half of the movies in the dataset earned less than this amount and half earned more.

The 75th percentile of the domestic gross earnings is \$27.9 million, indicating that 75% of the movies in the dataset earned less than this amount.

The maximum domestic gross earnings for a movie in the dataset is \$936.7 million, indicating that there are some very high-grossing movies in the dataset.

The dataset covers the years 2010 through 2018, with most of the movies released in 2013 and 2014.

DATA CLEANING.

```
In [178]: #Finding missing values.  
bom_movies_gross_df.isnull().sum()
```

```
Out[178]: studio          5  
domestic_gross        28  
foreign_gross       1350  
year                  0  
dtype: int64
```

The studio column has non-null values for 5 rows.

The domestic_gross column has non-null values for 28 rows.

The foreign_gross column has non-null values for 1350 rows.

The year column has non-null values for 0 rows, which may indicate that the column is empty or missing.

```
In [179]: #filling missing values.  
bom_movies_gross_df['domestic_gross'].fillna(bom_movies_gross_df['domestic_gross'])  
bom_movies_gross_df['studio'].fillna('Missing', inplace = True)  
  
bom_movies_gross_df.isna().sum()
```

```
Out[179]: studio          0  
domestic_gross        0  
foreign_gross       1350  
year                  0  
dtype: int64
```

```
In [180]: #dropping foreign gross row  
bom_movies_gross_df.drop('foreign_gross', axis=1, inplace=True)
```

```
In [181]: #previewing null values.
bom_movies_gross_df.isna().sum()
```

```
Out[181]: studio          0
domestic_gross      0
year                0
dtype: int64
```

```
In [182]: #checking for the range of values and outliers.
bom_movies_gross_df['domestic_gross'].unique()
```

```
Out[182]: array([4.150e+08, 3.342e+08, 2.960e+08, ..., 2.070e+04, 1.290e+04,
                2.400e+03])
```

```
In [183]: #checking for the range of values and outliers.
bom_movies_gross_df['year'].unique()
```

```
Out[183]: array([2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018], dtype=int64)
```

```
In [184]: #Grouping.
studio_gross_title = bom_movies_gross_df.groupby("title")["domestic_gross"].sum()
print(studio_gross_title)
```

	title	domestic_gross
0	'71	1300000.0
1	1,000 Times Good Night	53900.0
2	10 Cloverfield Lane	72100000.0
3	10 Years	203000.0
4	1001 Grams	11000.0
...
3381	Zoolander 2	28800000.0
3382	Zootopia	341300000.0
3383	[Rec] 2	27800.0
3384	mother!	17800000.0
3385	xXx: The Return of Xander Cage	44900000.0

[3386 rows x 2 columns]

```
In [185]: studio_gross = bom_movies_gross_df.groupby("studio")["domestic_gross"].sum().reset_index()
print(studio_gross)
```

	studio	domestic_gross
0	3D	6100000.0
1	A23	164200.0
2	A24	324194200.0
3	ADC	248200.0
4	AF	2142900.0
..
253	XL	458000.0
254	YFG	1100000.0
255	Yash	33031400.0
256	Zee	1100000.0
257	Zeit.	5663500.0

[258 rows x 2 columns]

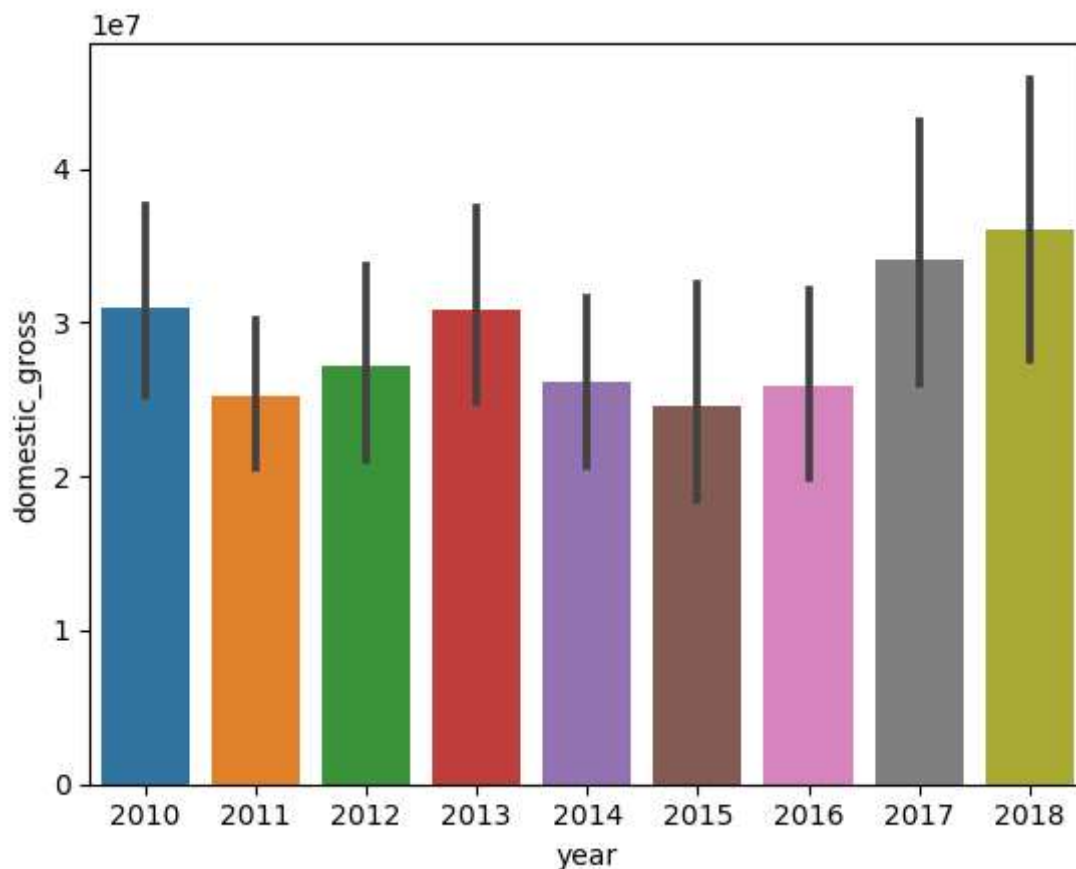
```
In [186]: bom_movies_gross_df.head()
```

```
Out[186]:
```

	studio	domestic_gross	year
title			
Toy Story 3	BV	415000000.0	2010
Alice in Wonderland (2010)	BV	334200000.0	2010
Harry Potter and the Deathly Hallows Part 1	WB	296000000.0	2010
Inception	WB	292600000.0	2010
Shrek Forever After	P/DW	238700000.0	2010

DATA ANALYSIS AND VISUALIZATION.

```
In [187]: #Linegraph
sns.barplot(x='year', y='domestic_gross', data=bom_movies_gross_df)
plt.show()
```



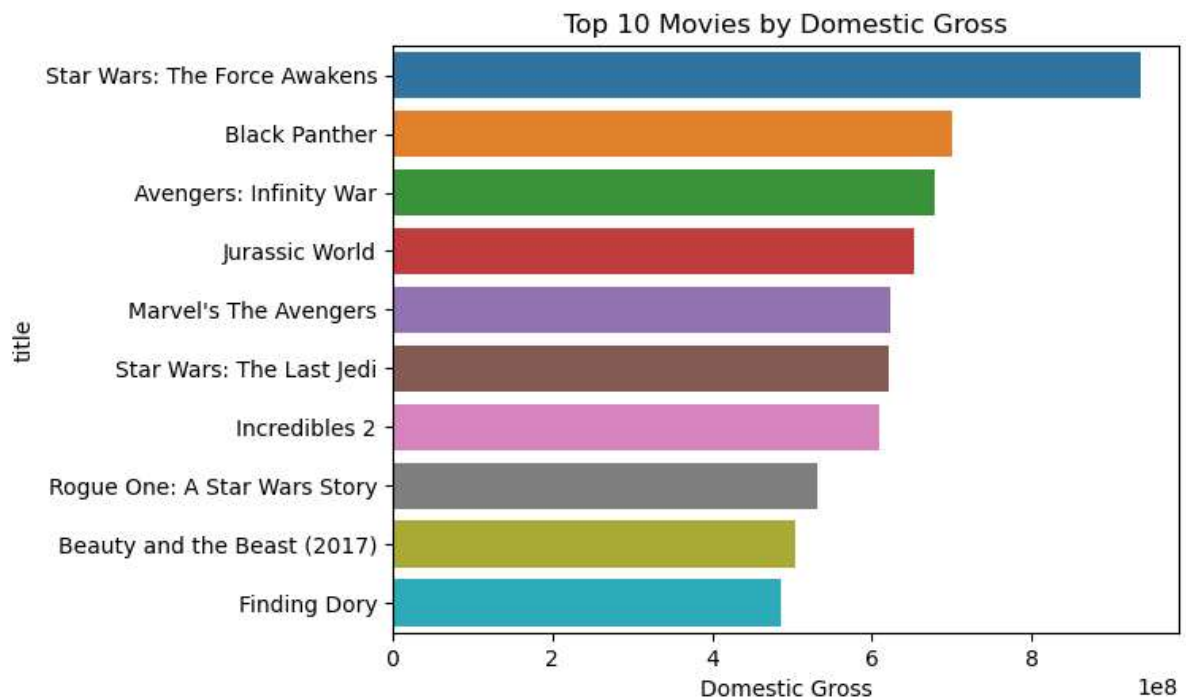
The above is a line graph that shows the trend in average domestic gross earnings for movies over the years in the dataset. The year 2018 generated the highest domestic gross income followed by 2017, 2013, 2010, 2012, 2014, 2016, 2011 and 2015 in that order.


```
In [188]: # Sort the dataset by domestic gross in descending order
bom_movies_gross_df = bom_movies_gross_df.reset_index()

sorted_data = bom_movies_gross_df.sort_values('domestic_gross', ascending=False)

# Selecting the top 10 movies by domestic gross
top_10 = sorted_data.head(10)

# Create the bar chart
sns.barplot(x="domestic_gross", y="title", data = top_10, orient = "h")
plt.xlabel("Domestic Gross")
plt.ylabel("title")
plt.title("Top 10 Movies by Domestic Gross")
plt.show()
```



The above is a horizontal bar chart of the top 10 movies by domestic gross, with the movie titles on the y-axis and the domestic gross on the x-axis. Star Wars: The Force Awakens generated the highest domestic gross revenue followed by Black Panther, Avengers: Infinity War, Jurassic World, Marvel's The Avengers, Star Wars: The Last Jedi, Incredibles 2, Rogue One: A Star Wars Story, Beauty and the Beast(2017) while the least was generated by Finding Dory.

FINDINGS

Domestic gross revenue was generated highest in the year 2018, then 2017, 2013, 2010, 2012, 2014, 2016, 2011 with the least being in 2015.

RECOMMENDATIONS

I would recommend Microsoft to produce Star Wars:The Force Awakens, Black Panther and Avengers: Infinity War movies since they generated the highest domestic gross income.