Project: TMDB movie Data Analysis

Table of Contents

Introduction

Dataset Description

> In this project, we need to examine the TMDB movie dataset and share what we discover. We will use Python tools such as NumPy, pandas, and Matplotlib to simplify our analysis of the TMDB movie data. This dataset has details on 10,000 movies from the Movie Database (TMDb), covering aspects like user ratings and revenue. It includes information on various fields such as 'cast', 'genres', and 'characters'.

- **id**: Unique movie identifier.
- **imdb_id**: IMDB code specific to the movie.
- **popularity**: Metric indicating the movie's popularity.
- **budget**: Amount spent to produce the movie.
- **revenue**: Earnings from the movie.
- **original_title**: The movie's original name.
- **cast**: Leading actors involved in the movie.
- **homepage**: Official website of the movie.
- **director**: The movie's director.
- **tagline**: A catchy phrase representing the movie.
- **keywords**: Terms associated with the movie.
- **overview**: A brief summary of the movie.
- **runtime**: Total duration of the movie in minutes.
- **genres**: Categories describing the movie's style and content.
- **production_companies**: Firms that produced the movie.
- **release_date**: When the movie was first released.
- **vote_count**: Total votes received by the movie.
- **vote_average**: Average rating given to the movie.
- **release_year**: Year the movie was released.
- **budget_adj**: Movie's budget adjusted for inflation.
- **revenue_adj**: Movie's revenue adjusted for inflation.

Question(s) for Analysis

```
Q1. What are the most common movie genres?
Q2. Who are the most cast actors?
Q3. What production company produces the most movies?
Q4. What are the top movies in terms of profit?
Q5. What are the top movies based on popularity?
Q6. What are the top movies based on viewer rating?
Q7. What are the most common keywords?
Q8. Is the budget related to a higher average vote?
Q9. what's the correlation between runtime and vote average, budget and popularity?
Q10. Who are the most successful directors?
Q11. How did the runtime of movies change over the years? What Movie has the longest runtime? what movie has the shortest runtime? what's the aver
```

```
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Data Wrangling

- Loading the Data
- Exploring the Data
- Data Cleaning
  - Check if the data is clean, remove columns that are not needed, look for missing values (NAN values) and correct them, etc.

```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.5
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->
```

Loading the Data

```
# Load the dataset
# Importing the movie dataset
dataset_location = 'tmdb-movies.csv'
data = pd.read_csv(dataset_location)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-3-1cbcf29327e7> in <cell line: 4>()
      2 # Importing the movie dataset
      3 dataset_location = 'tmdb-movies.csv'
----> 4 data = pd.read_csv(dataset_location)

4 frames
/usr/local/lib/python3.10/dist-packages/pandas/io/common.py in get_handle(path_or_buf, mode, encoding, compression, memory_map,
is_text, errors, storage_options)
    871         if ioargs.encoding and "b" not in ioargs.mode:
    872             # Encoding
--> 873             handle = open(
    874                 handle,
    875                 ioargs.mode,

FileNotFoundError: [Errno 2] No such file or directory: 'tmdb-movies.csv'
```

Next steps: Explain error

Exploring the Data

```
# Show initial and final rows of the movie data
data.head() # Display the first few entries
```

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	direc
0	135397	tt0369610	32.985763	150000000	1513528810	Jurassic World	Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...	http://www.jurassicworld.com/	C Trevor
1	76341	tt1392190	28.419936	150000000	378436354	Mad Max: Fury Road	Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...	http://www.madmaxmovie.com/	Geo Mi
2	262500	tt2908446	13.112507	110000000	295238201	Insurgent	Shailene Woodley|Theo James|Kate Winslet|Ansel...	http://www.thedivergentseries.movie/#insurgent	Rob Schwen
3	140607	tt2488496	11.173104	200000000	2068178225	Star Wars: The Force Awakens	Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...	http://www.starwars.com/films/star-wars-episod...	Abra
4	168259	tt2820852	9.335014	190000000	1506249360	Furious 7	Vin Diesel|Paul Walker|Jason Statham|Michelle ...	http://www.furious7.com/	Jan W

5 rows × 21 columns

```
# Display the last few entries of the dataset
data.tail()
```

	id	imdb_id	popularity	budget	revenue	original_title	cast	homepage	director	tagline	...	overvie
10861	21	tt0060371	0.080598	0	0	The Endless Summer	Michael Hynson|Robert August|Lord 'Tally Ho' B...	NaN	Bruce Brown	NaN	...	Th Endles Summe by Bruc Brown, one of
10862	20379	tt0060472	0.065543	0	0	Grand Prix	James Garner|Eva Marie Saint|Yves Montand|Tosh...	NaN	John Frankenheimer	Cinerama sweeps YOU into a drama of speed and ...	...	Grand Pr driver Pe Aron fired by h te
10863	39768	tt0060161	0.065141	0	0	Beregis Avtomobilya	Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...	NaN	Eldar Ryazanov	NaN	...	A insuranc agent wh moonligh as carthie
10864	21449	tt0061177	0.064317	0	0	What's Up, Tiger Lily?	Tatsuya Mihashi|Akiko Wakabayashi|Mie Hama|Joh...	NaN	Woody Allen	WOODY ALLEN STRIKES BACK!	...	In com Wooc Allen's fil debut, h took the
10865	22293	tt0060666	0.035919	19000	0	Manos: The Hands of Fate	Harold P. Warren|Tom Neyman|John Reynolds|Dian...	NaN	Harold P. Warren	It's Shocking! It's Beyond Your Imagination!	...	A fami gets lo on th road ar stumble up

5 rows × 21 columns

```
# Output the total count of rows and columns in the dataset
print(f"There are {data.shape[0]:,} rows and {data.shape[1]:,} columns in the dataset.")
```

```
There are 10,866 rows and 21 columns in the dataset.
```

```
# Generate statistical summary for the dataset's numerical columns
data.describe()
```

	id	popularity	budget	revenue	runtime	vote_count	vote_average	release_year	budget_adj	rever
count	10866.000000	10866.000000	1.086600e+04	1.086600e+04	10866.000000	10866.000000	10866.000000	10866.000000	1.086600e+04	1.0866
mean	66064.177434	0.646441	1.462570e+07	3.982332e+07	102.070863	217.389748	5.974922	2001.322658	1.755104e+07	5.1364
std	92130.136561	1.000185	3.091321e+07	1.170035e+08	31.381405	575.619058	0.935142	12.812941	3.430616e+07	1.4463
min	5.000000	0.000065	0.000000e+00	0.000000e+00	0.000000	10.000000	1.500000	1960.000000	0.000000e+00	0.0000
25%	10596.250000	0.207583	0.000000e+00	0.000000e+00	90.000000	17.000000	5.400000	1995.000000	0.000000e+00	0.0000
50%	20669.000000	0.383856	0.000000e+00	0.000000e+00	99.000000	38.000000	6.000000	2006.000000	0.000000e+00	0.0000
75%	75610.000000	0.713817	1.500000e+07	2.400000e+07	111.000000	145.750000	6.600000	2011.000000	2.085325e+07	3.3697
max	417859.000000	32.985763	4.250000e+08	2.781506e+09	900.000000	9767.000000	9.200000	2015.000000	4.250000e+08	2.8271

1. Popularity:

Popularity scores in the dataset range widely from nearly 0 to 32.99. However, most movies have low popularity, with an average score of about 0.65.

2. Revenue:

Revenues range from 0 dollars many movies made no money to about 2.78 billion dollars. This shows that while some movies earn huge amounts, many do not make any revenue.

3. Vote Count:

Votes per movie vary greatly from 10 to 9,767, indicating that some movies are much more popular with viewers than others.

```
# Count the unique values in each column
unique_values_count = data.nunique()
unique_values_count
```

```
id                      10865
imdb_id                 10855
popularity              10814
budget                    557
revenue                  4702
original_title          10571
cast                    10719
homepage                 2896
director                 5067
tagline                  7997
keywords                 8804
overview                10847
runtime                   247
genres                   2039
production_companies     7445
release_date             5909
vote_count               1289
vote_average               72
release_year               56
budget_adj               2614
revenue_adj              4840
dtype: int64
```

```
# Calculate the total count of missing values per column
missing_values_count = data.isnull().sum()
missing_values_count
```

```
id                  0
imdb_id            10
popularity          0
budget              0
revenue             0
original_title      0
cast               76
```

```
homepage                7930
director                  44
tagline                 2824
keywords                1493
overview                   4
runtime                    0
genres                    23
production_companies    1030
release_date               0
vote_count                 0
vote_average               0
release_year               0
budget_adj                 0
revenue_adj                0
dtype: int64
```

Columns with many unique values compared to the number of rows are high-cardinality categorical features. Columns with few unique values are likely categorical.

```
# Display the data types and check for missing values in each column
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   id                    10866 non-null  int64  
 1   imdb_id               10856 non-null  object 
 2   popularity            10866 non-null  float64
 3   budget                10866 non-null  int64  
 4   revenue               10866 non-null  int64  
 5   original_title        10866 non-null  object 
 6   cast                  10790 non-null  object 
 7   homepage              2936 non-null   object 
 8   director              10822 non-null  object 
 9   tagline               8042 non-null   object 
 10  keywords              9373 non-null   object 
 11  overview              10862 non-null  object 
 12  runtime               10866 non-null  int64  
 13  genres                10843 non-null  object 
 14  production_companies  9836 non-null   object 
 15  release_date          10866 non-null  object 
 16  vote_count            10866 non-null  int64  
 17  vote_average          10866 non-null  float64
 18  release_year          10866 non-null  int64  
 19  budget_adj            10866 non-null  float64
 20  revenue_adj           10866 non-null  float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

Some columns have missing values: cast, homepage, director, tagline, keywords, overview, genres, and production_companies. All data types are correct except for release_date and release_year. I will keep the datatype of release_year as it is. The release_date will be dropped later in the analysis because it is not important for this study.

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Cancel upload

Data Cleaning

- Based on my observations from the first 5 rows, and from previous datasets we need to do the following:
  1. The columns id, imdb_id, homepage, budget_adj, revenue adj are useless, hence, we need to delete them
  2. remove dublicates
  3. check for NAN values
  4. replace null values with NAN

Dropping Unimportant Columns, duplicates, and null values.

I will drop the following columns because they are not important or needed for this analysis: 'id', 'imdb_id', 'homepage', 'tagline', 'overview', 'vote_count', 'budget_adj', and 'revenue_adj'.

```
# Remove columns that are not significant for our analysis
columns_to_drop = ['id', 'imdb_id', 'homepage', 'tagline', 'overview', 'vote_count', 'budget_adj', 'revenue_adj', 'release_date']
data.drop(columns=columns_to_drop, axis=1, inplace=True)

# Display the first few rows after dropping columns
data.head()
```

```
# Calculate the number of duplicate rows in the dataset
duplicate_entries_count = data.duplicated().sum()
duplicate_entries_count
```

...

```
# Remove duplicate rows from the dataset
data.drop_duplicates(inplace=True)

# Verify that all duplicate rows have been removed
remaining_duplicates = data.duplicated().sum()
remaining_duplicates
```

...

```
# Drop rows with missing values in critical columns
data = data.dropna(subset=['genres','director', 'cast'])
```

```
data.info()
```

...

```
# Convert specified object type columns to string type
cols_to_convert = ['production_companies', 'genres', 'keywords', 'director', 'original_title', 'cast']
data[cols_to_convert] = data[cols_to_convert].astype(str)
```

...

```
data.info()
```

...

Changing columns to string makes sure all entries, even numbers and NaN values, are treated as strings in those columns.

The keywords and production_companies columns have null values, but since I might not need these columns, I'll leave them as they are (impute them).

```
# Save the data before removing 'production_companies' and 'keywords' columns
# These columns will be used only in two specific research questions

# Create a copy of the data for company-related analysis
company_data = data.copy()

# Drop the 'keywords' column from the company_data
company_data.drop('keywords', axis=1, inplace=True)

# Remove rows with missing values in the 'production_companies' column
company_data.dropna(subset=['production_companies'], inplace=True)

# Remove NaN values from company_data
company_data = company_data[company_data != 'nan']

##########################################################################

# Create a copy of the data
keywords_data = data.copy()

# Drop the 'production_companies' column from the keywords_data
keywords_data.drop('production_companies', axis=1, inplace=True)

# Remove rows with missing values in the 'keywords' column
keywords_data.dropna(subset=['keywords'], inplace=True)
# Remove NaN values from keywords_data
keywords_data = keywords_data[keywords_data != 'nan']
```

```
# Verify the changes by checking for missing values in the modified datasets
print("Missing values in company_data:")
print(company_data.isnull().sum())

print("\nMissing values in keywords_data:")
print(keywords_data.isnull().sum())
```

•••

We save the data before removing the 'production_companies' and 'keywords' columns. These columns will be used only for two research questions. They are removed because they have many missing values. Dropping these rows would affect other columns, so we will handle the questions with the missing rows imputed.

```
data.drop(['production_companies','keywords'], axis = 1, inplace=True)
```

...

```
#Number of missing values in each column.
data.isnull().sum()
```

...

```
# Remove NaN values
data = data[data != 'nan']
```

Review dataset

```
# Display the first 10 rows of the dataset to get an overview
data.head(10)
```

```
# Get the dimensions of the dataset (number of rows and columns)
dataset_dimensions = data.shape

# Display the dimensions
dataset_dimensions
```

...

```
# Plot the distribution of the numerical features in the dataset
data.hist(figsize=(15,10));
```

The popularity column's distribution, seen in the summary statistics, is skewed to the right. This is also true for the budget, revenue, vote_count, vote_average, and runtime columns. The release_year column is skewed to the left, showing that more movies were made or released from the 2000s to 2015 compared to earlier years.

Exploratory Data Analysis

- Relationship Between Independent Features
- Total Number of Movies Produced by Year
- Total Number of Movies Produced by Genre

> Now that we've cleaned the data, we can start exploring it. In this section, we'll calculate statistics and make visualizations to answer the research questions from the Introduction.

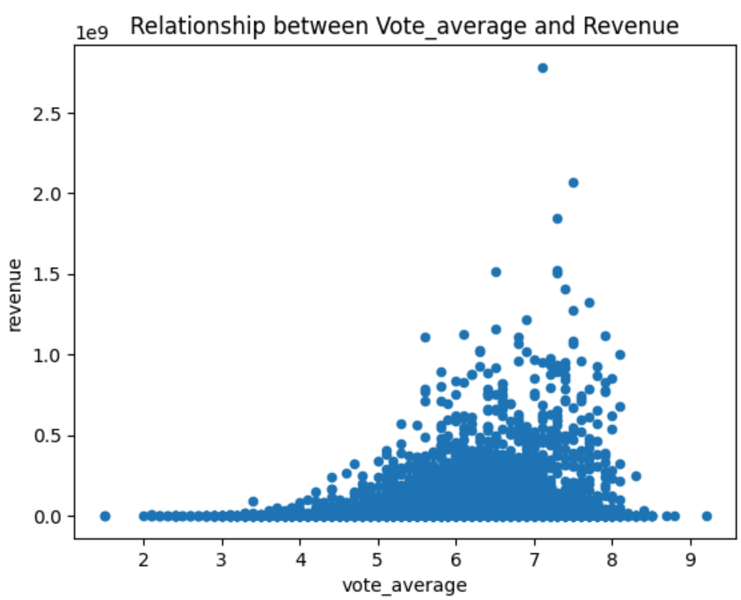
General Questions

Q1. What is the relationship Between Independent Features

```
# Plots the relationship between two variables as a scatter plot.

#     Args:
#     data : DataFrame containing the data.
#     x_axis : Column name for the x-axis.
#     y_axis : Column name for the y-axis (default is 'revenue').
#     plot_kind : Type of plot (default is 'scatter').
#     title_prefix : Prefix for the title (default is 'Relationship between').

def plot_relationship(data, x_axis, y_axis='revenue', plot_kind='scatter', title_prefix='Relationship between'):
    data.plot(x=x_axis, y=y_axis, kind=plot_kind)
    plt.title(f'{title_prefix} {x_axis.capitalize()} and {y_axis.capitalize()}')
    plt.show()


# #Relationship between popularity and revenue.


plot_relationship(data, 'popularity')
```

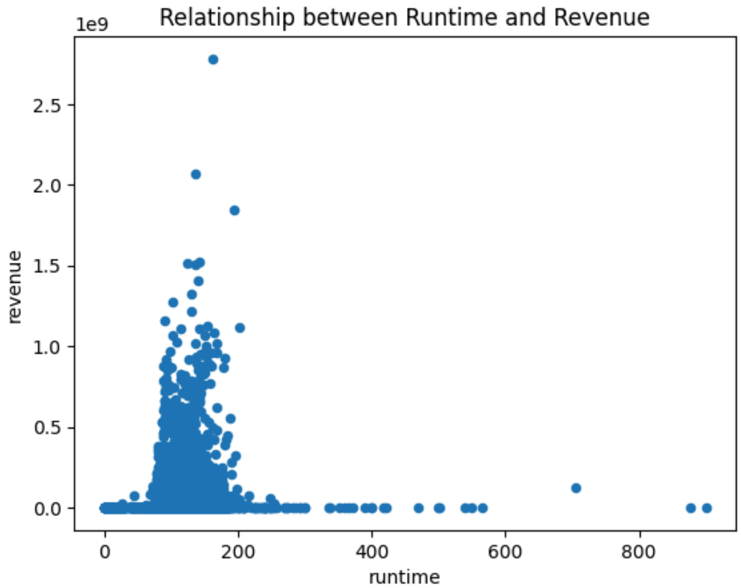
1. There is some positive correlation between popularity and revenue as this plot shows that the correlation is not that strong. This will be investigated further later in the analysis.
2. X-Axis (Popularity): Represents the independent variable, popularity. This axis measures how popular the instances (e.g., movies) are, with values ranging from 0 to above 30.
3. Y-Axis (Revenue): Represents the dependent variable, revenue. This axis measures the revenue generated by the instances, with values up to around 2.5 billion.

```
# #Relationship between vote_average and revenue.

plot_relationship(data, 'vote_average')
```



1. There is some positive correlation between vote_average and revenue. This will be investigated further later in the analysis.
2. X-Axis (Vote Average): Represents the independent variable, vote average. This axis measures the average rating given to the instances (e.g., movies), with values ranging from 2 to 9.
3. Y-Axis (Revenue): Represents the dependent variable, revenue. This axis measures the revenue generated by the instances, with values up to around 2.5 billion.

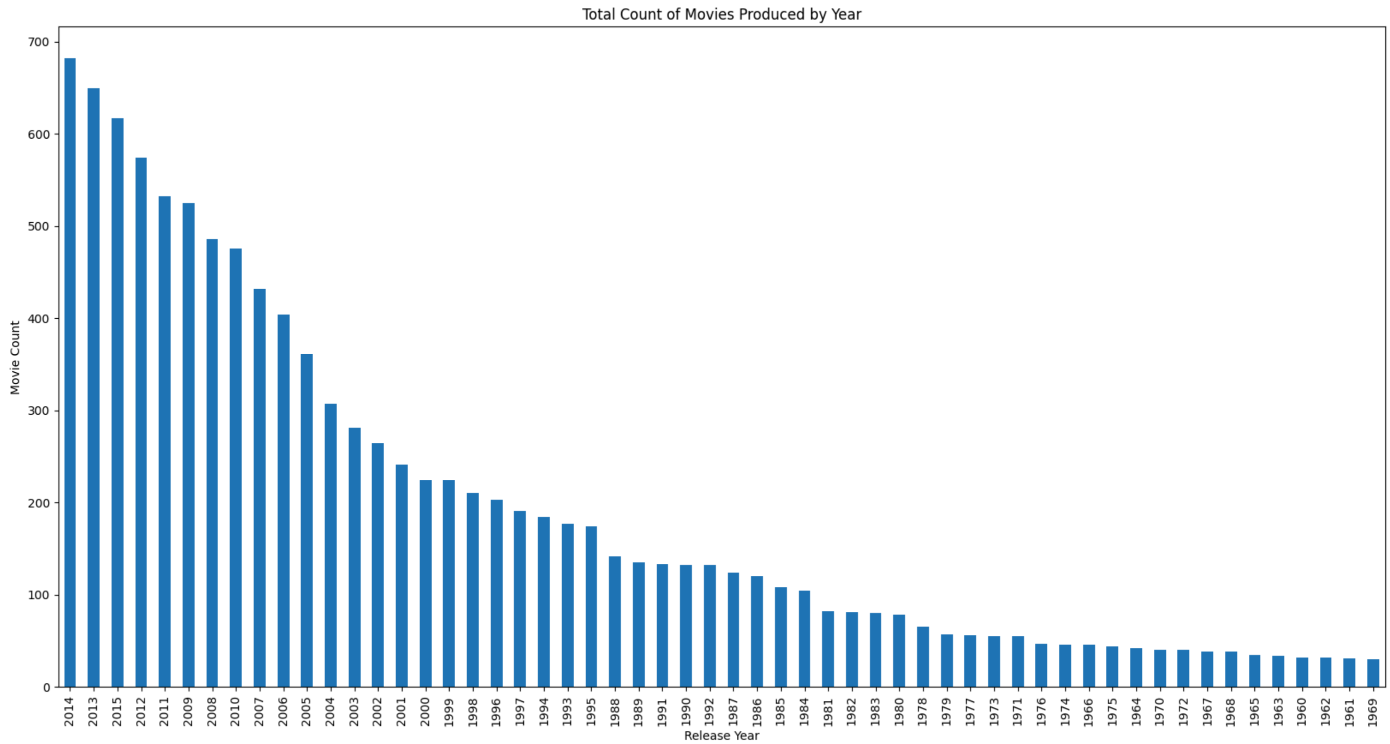
```
# #Relationship between runtime and revenue.

plot_relationship(data, 'runtime')
```



1. There is no correlation between runtime and revenue. This will be investigated further later in the analysis.
2. X-Axis (Runtime): Represents the independent variable, runtime. This axis measures the duration of the instances (e.g., movies) in minutes, with values ranging from 0 to over 800 minutes.
3. Y-Axis (Revenue): Represents the dependent variable, revenue. This axis measures the revenue generated by the instances, with values up to around 2.5 billion.

Q2. What is the total Number of Movies Produced by Year?

```
#Value counts of movies for each year.
data.release_year.value_counts().plot(kind = 'bar', figsize = (20, 10));
plt.xlabel('Release Year');
plt.ylabel('Movie Count')
plt.title('Total Count of Movies Produced by Year');
```



The total amount of movies produced by year has been increasing steadily over the years. The year 2014 recorded the most number of movies released/produced.

Research Questions

Research Question 1: What are the most common movie genres?

↳ *6 cells hidden*

Research Question 2: Who are the most cast actors?

[ ] ↳ *14 cells hidden*

Research Question 3: What production company produces the most movies?

[ ] ↳ *3 cells hidden*

Research Question 4: What are the top movies in terms of profit?

[ ] ↳ *9 cells hidden*

> Research Question 5: What are the top movies based on popularity?

[ ] ↳ *11 cells hidden*

> Research Question 6: What are the top movies based on viewer rating?

[ ] ↳ *13 cells hidden*

> Research Question 7: What are the most common keywords?

[ ] ↳ *6 cells hidden*

> Research question 8: Is the budget related to a higher average vote?

[ ] ↳ *2 cells hidden*

> Research Question 9: what's the correlation between runtime and each of popularity, rating, popularity?

[ ] ↳ *4 cells hidden*

> Research Question 10: Who are the most successful directors?

```
Most successful director is the one who generated the most revenue
```

[ ] ↳ *3 cells hidden*

> Research Question 11: How did the runtime of movies change over the years? What Movie has the longest runtime? what movie has the shortest runtime? what's the average movie runtime?

[ ] ↳ *13 cells hidden*

Conclusions

```
1. Key columns like 'cast' and 'director' have missing values. 'Release_date' will be dropped as it's irrelevant to the analysis.
2. Popularity, budget, and revenue distributions are skewed right, indicating more entries from 2000 to 2015.
3. Weak positive correlations exist between popularity and revenue, and between vote_average and revenue.
4. The most movies were produced in 2014.
5. Drama, comedy, and thriller are the most frequent genres.
6. Robert De Niro, Samuel L. Jackson, and Bruce Willis are the most featured actors.
7. Universal Pictures, Warner Bros., and Paramount Pictures are the top movie producers.
8. 'Avatar', 'Star Wars', and 'Titanic' are the most profitable films, showing that high budgets don't always lead to high profits.
9. The longest movie recorded is 'The Story of Film: An Odyssey' at 900 minutes, and the average movie runtime is 102 minutes, which has been decr
```

Data limitations:

1. **Viewer Bias**: Viewer counts vary significantly across movies. Consequently, movies with fewer viewers may show disproportionately high ratings, which could skew perceived quality assessments.
2. **Duration Variability**: The dataset includes movies shorter than 40 minutes, classified as short films. Their inclusion could distort analyses