Visualization of the Data

After cleaning the dataset, I imported it and used it to visualize it to produce insights that will be of use and help to the client, Microsoft.

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

In [2]:
```
df = pd.read_csv('cleaned_data.csv',)
df
```

	0	production_budget	worldwide_gross	original_title	start_year	runtime_minutes	genr
0	0	80000000.0	247023808.0	The Courier	2012	95.0	Action,Crime,Dran
1	1	80000000.0	226739416.0	Pelé: Birth of a Legend	2016	107.0	Biography,Drama,Spo
2	2	80000000.0	168311558.0	Bibliothèque Pascal	2010	105.0	Dran
3	3	80000000.0	241200000.0	The Experiment	2010	96.0	Drama,Thrill
4	4	80000000.0	221468935.0	Stadilaista tangoa etsimässä	2010	90.0	Documentary,Mus
...	...	...	...	...	...	...	
2110	2172	7000.0	841926.0	Pet	2016	94.0	Horror,Thrill
2111	2173	7000.0	71644.0	Stag Night of the Dead	2010	81.0	Action,Comedy,Horr

In [3]:
```
df = df.drop(columns=['Unnamed: 0'])
```

In [4]:
```
df.columns
```

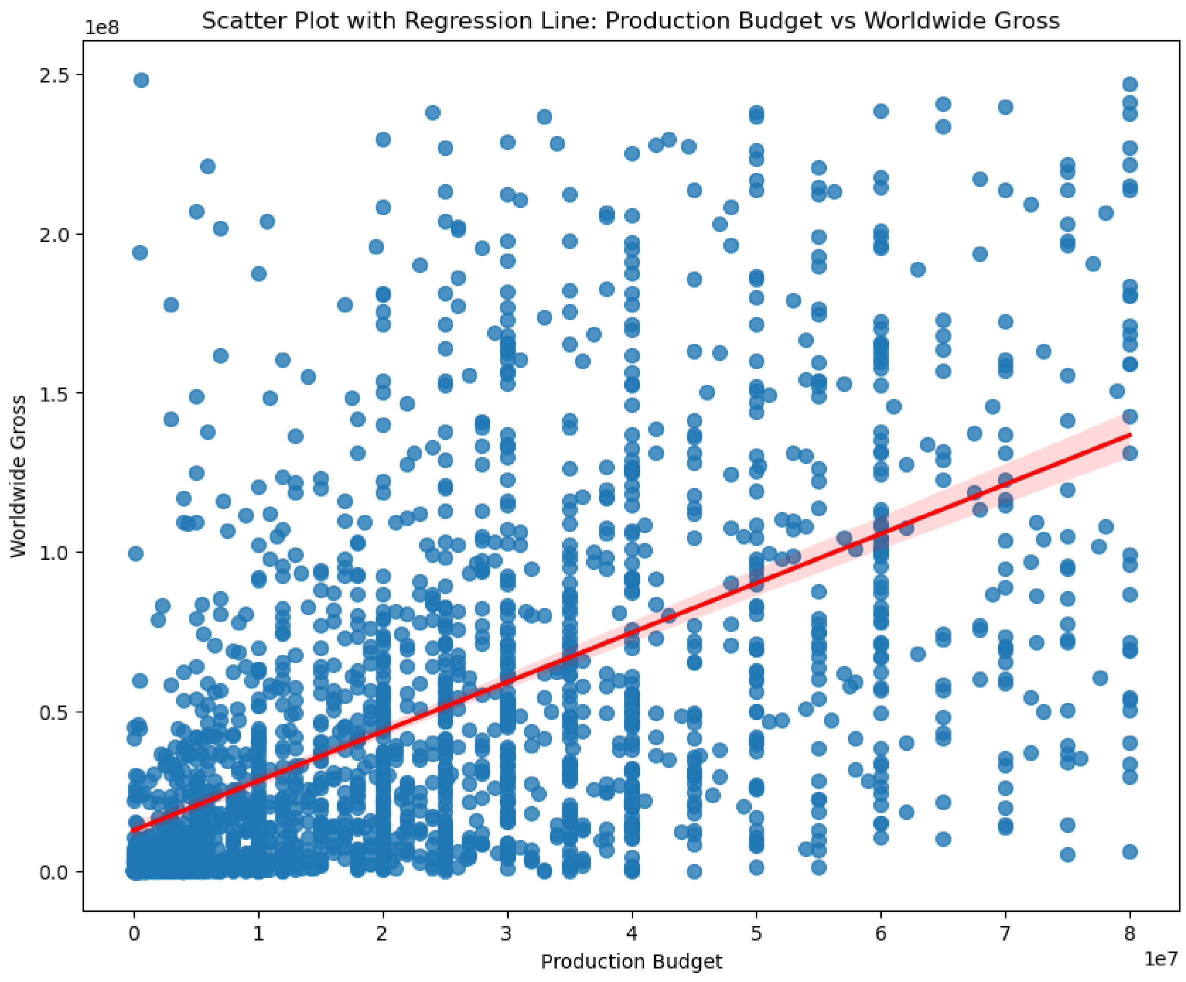
Out[4]:
```
Index(['production_budget', 'worldwide_gross', 'original_title', 'start_year',
       'runtime_minutes', 'genres'],
      dtype='object')
```

In [5]:
```
# Calculate the correlation matrix
correlation = df['production_budget'].corr(df['worldwide_gross'])

# Display the correlation coefficient
print(f"Correlation between 'production_budget' and 'worldwide_gross': {correlation:.2f}")
```

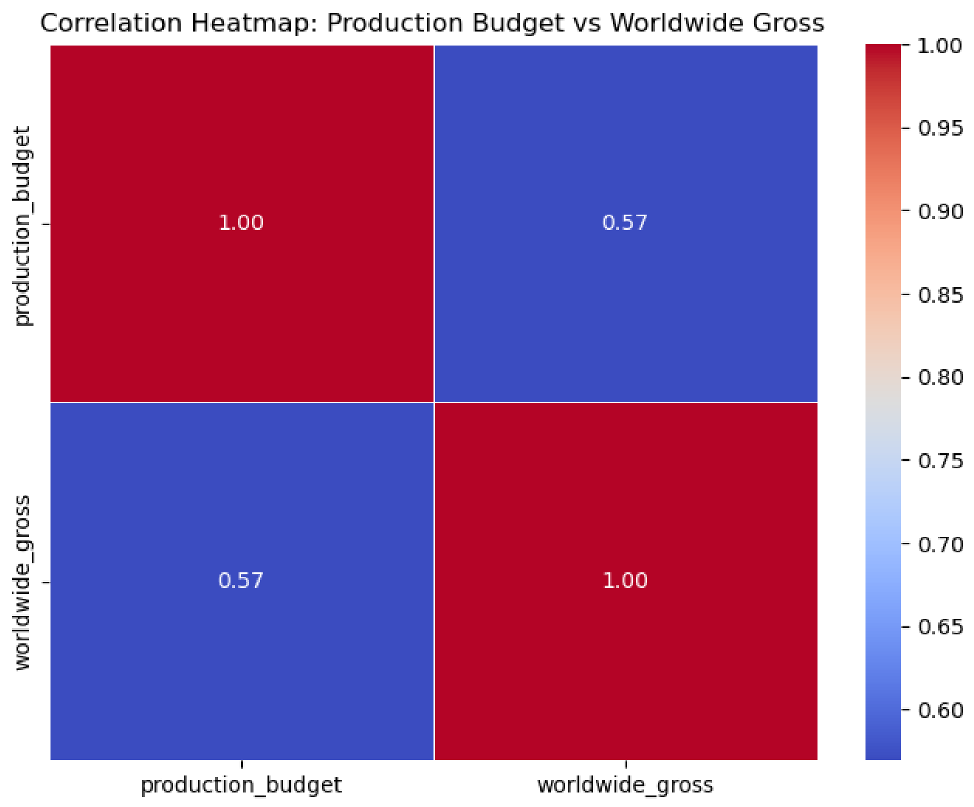
```
Correlation between 'production_budget' and 'worldwide_gross': 0.57
```

In [6]:

```
# Exploring the relationship between Production Budget and Worldwide Gross using a scatter plot and a
# the trend
plt.figure(figsize=(10, 8))
sns.regplot(x='production_budget', y='worldwide_gross', data=df, scatter_kws={'s': 50}, line_kws={'co

# Set labels and title
plt.title('Scatter Plot with Regression Line: Production Budget vs Worldwide Gross')
plt.xlabel('Production Budget')
plt.ylabel('Worldwide Gross')

# Display the plot
plt.show();
# As Production budget increases, so does the Worldwide Gross increase
```



In [7]:
```
# Creating a DataFrame with only production_budget and worldwide_gross to understand their correlatio
selected_columns = ['production_budget', 'worldwide_gross']
subset_df = df[selected_columns]

# Calculate the correlation matrix for the selected columns
correlation_matrix_subset = subset_df.corr()

# Create a heatmap using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix_subset, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

# Set labels and title
plt.title('Correlation Heatmap: Production Budget vs Worldwide Gross')
# Display the plot
plt.show()
# A positive correlation exists between Production Budget and Worldwide Gross, which is a good thing
# Production budget leads to an increase in Worldwide Gross
```



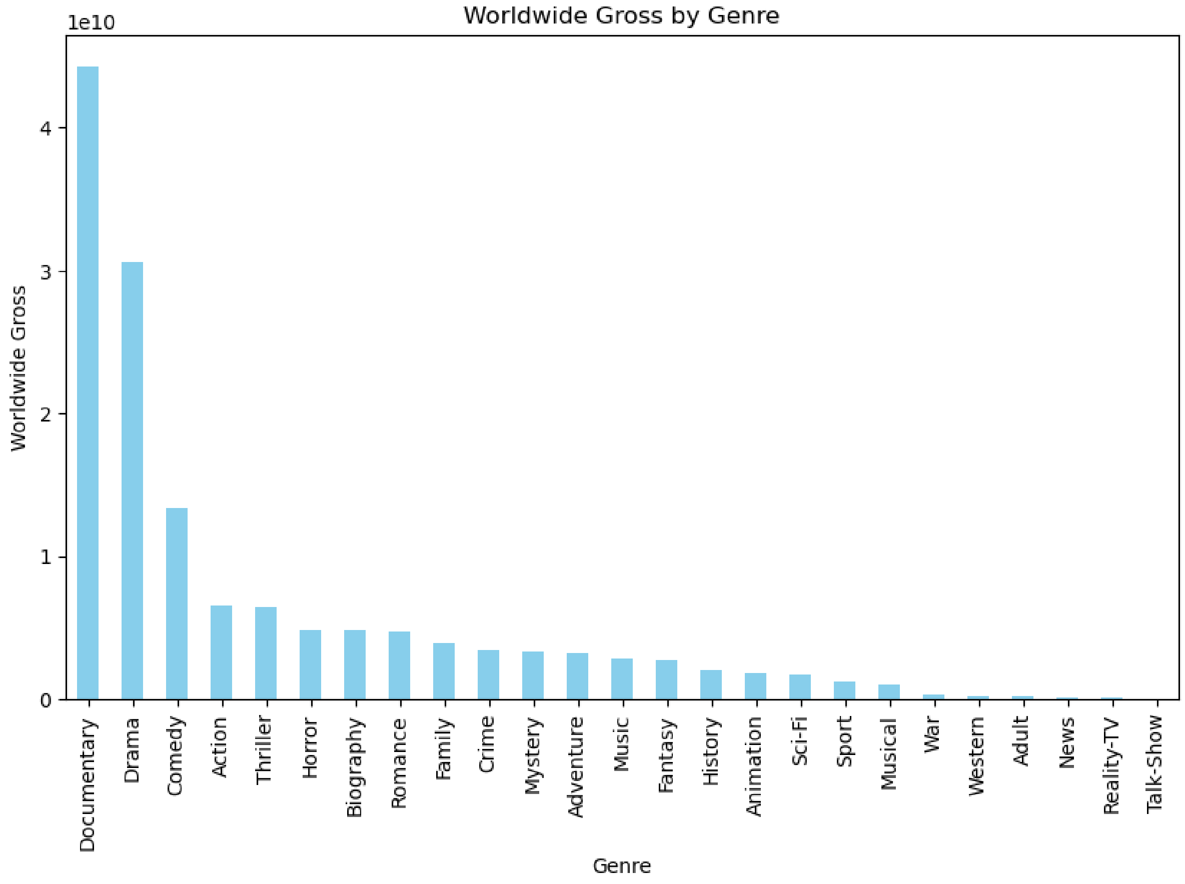
In [8]:
```
print(df.columns)
```

```
Index(['production_budget', 'worldwide_gross', 'original_title', 'start_year',
       'runtime_minutes', 'genres'],
      dtype='object')
```

In [9]:
```
# Determining the top 3 genres with high worldwide gross generation
# Split genres into separate rows
df_genres = df['genres'].str.split(',', expand=True).stack().reset_index(level=1, drop=True).to_frame
df_expanded = df_genres.join(df.drop('genres', axis=1))

# Group by genre and calculate the sum of worldwide_gross for each genre
genre_worldwide_gross = df_expanded.groupby('genre')['worldwide_gross'].sum()

# Find the top 3 genres
top_genres = genre_worldwide_gross.nlargest(3)

print("Top 3 genres with high worldwide gross:")
print(top_genres)
# Top 3 genres with high worldwide gross generation are Documentary, Drama and Comedy.
```

```
Top 3 genres with high worldwide gross:
genre
Documentary    4.424609e+10
Drama          3.062846e+10
Comedy         1.343596e+10
Name: worldwide_gross, dtype: float64
```

In [10]:
```
# Create a bar plot to measure genres with highest worldwide gross.
plt.figure(figsize=(10, 6))
genre_worldwide_gross.sort_values(ascending=False).plot(kind='bar', color='skyblue')
plt.title('Worldwide Gross by Genre')
plt.xlabel('Genre')
plt.ylabel('Worldwide Gross')
plt.show();
# According to genre classification by Worldwide Gross, Documentaries seem to be pooling massive Worl
# by drama and comedy
```



In [11]:
```
# Calculate the total number of movies per genre
genre_counts = df['genres'].value_counts().reset_index()
genre_counts.columns = ['Genre', 'Total Movies']

# Rank genres based on the total number of movies
genre_counts['Rank'] = genre_counts['Total Movies'].rank(ascending=False)
genre_counts.head(6)
```

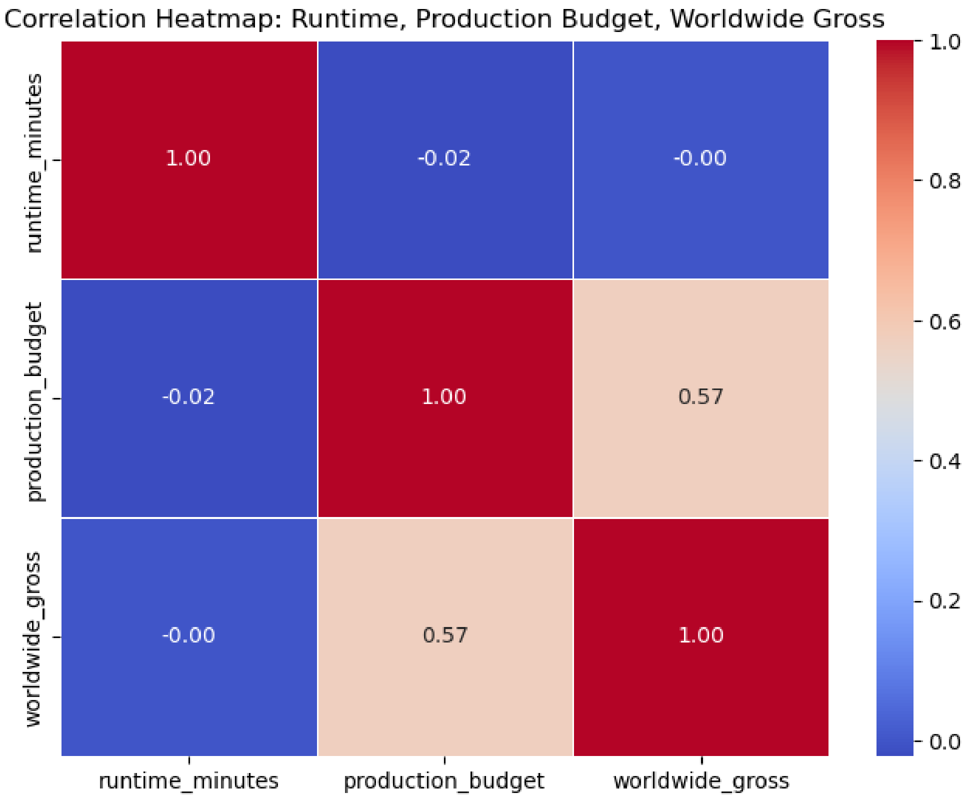
Out[11]:

	Genre	Total Movies	Rank
0	Documentary	717	1.0
1	Drama	321	2.0
2	Comedy	116	3.0
3	Comedy,Drama	62	4.0
4	Horror	60	5.0
5	Thriller	49	6.0

In [12]:
```
# Calculate the correlation matrix
correlation_matrix = df[['runtime_minutes', 'production_budget', 'worldwide_gross']].corr()

# Display the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)
```

```
Correlation Matrix:
                   runtime_minutes  production_budget  worldwide_gross
runtime_minutes           1.000000          -0.022727        -0.002747
production_budget        -0.022727           1.000000         0.569324
worldwide_gross          -0.002747           0.569324         1.000000
```

In [13]:
```
# Determining the correlation between runtime and Production Budget and Worldwide Gross
# Create a heatmap using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

# Set labels and title
plt.title('Correlation Heatmap: Runtime, Production Budget, Worldwide Gross')
plt.show()
# There is a negative correlation between runtime minutes and Worldwide Gross, and similarly, a negat
# exists between runtime minutes and Production Budget.
```



In [14]:
```
# Calculating the profit margins based on worldwide gross and production budget and add it to the Dat
df['profit_margin'] = ((df['worldwide_gross'] - df['production_budget']) / df['worldwide_gross']) * 1
df
```

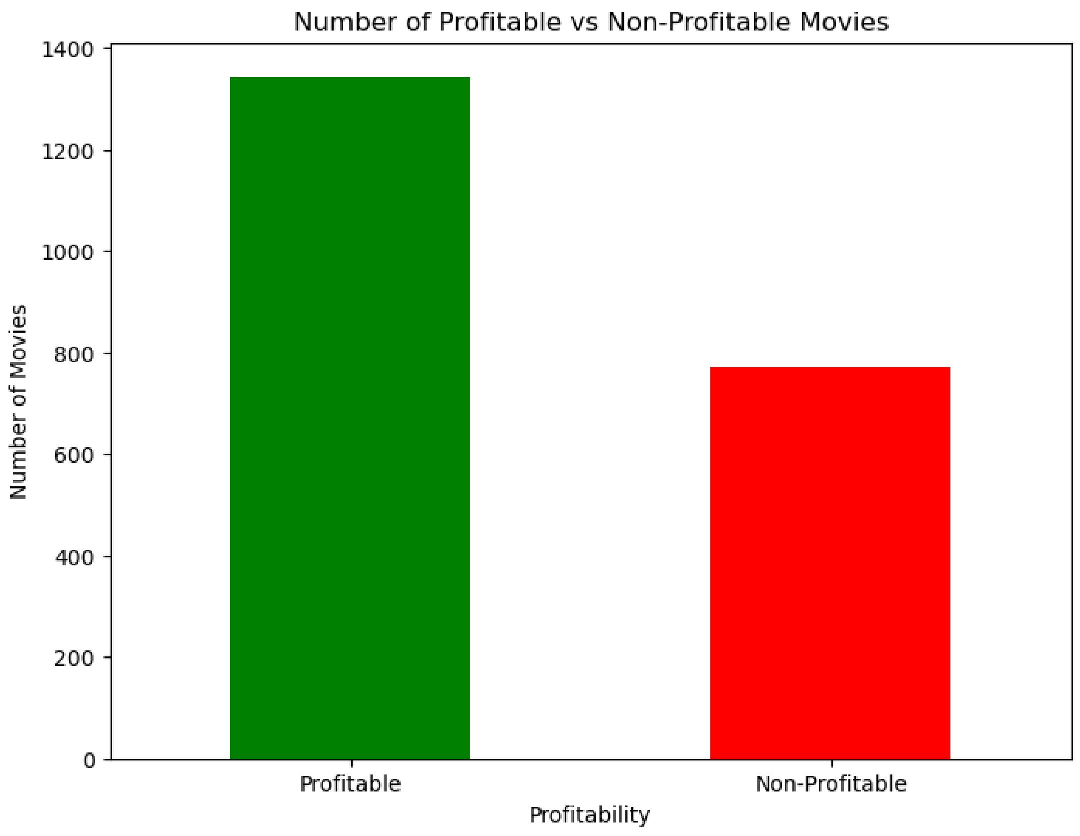
Out[14]:

	production_budget	worldwide_gross	original_title	start_year	runtime_minutes	genres	profit_margi
0	80000000.0	247023808.0	The Courier	2012	95.0	Action,Crime,Drama	67.61445
1	80000000.0	226739416.0	Pelé: Birth of a Legend	2016	107.0	Biography,Drama,Sport	64.71720
2	80000000.0	168311558.0	Bibliothèque Pascal	2010	105.0	Drama	52.46909
3	80000000.0	241200000.0	The Experiment	2010	96.0	Drama,Thriller	66.83250
4	80000000.0	221468935.0	Stadilaista tangoa etsimässä	2010	90.0	Documentary,Music	63.87755
...	...	...	...	...	...	...	
2110	7000.0	841926.0	Pet	2016	94.0	Horror,Thriller	99.16857
2111	7000.0	71644.0	Stag Night of the Dead	2010	81.0	Action,Comedy,Horror	90.22946
2112	7000.0	900.0	Mr. Nice	2010	121.0	Biography,Comedy,Crime	-677.77777
2113	6000.0	240495.0	Marley	2012	144.0	Biography,Documentary,Music	97.50514
2114	5000.0	1338.0	Welcome to the Rileys	2010	110.0	Drama	-273.69207

2115 rows × 7 columns

In [15]:

```
# Categorizing movies as profitable or non-profitable
df['profitable'] = df['profit_margin'] > 0

# Count the number of profitable and non-profitable movies
profitable_counts = df['profitable'].value_counts()

# Create a bar plot
plt.figure(figsize=(8, 6))
profitable_counts.plot(kind='bar', color=['green', 'red'])
plt.title('Number of Profitable vs Non-Profitable Movies')
plt.xlabel('Profitability')
plt.ylabel('Number of Movies')
plt.xticks([0, 1], ['Profitable', 'Non-Profitable'], rotation=0)
plt.show()
```



In [16]:
```
# Determining the ROI: Calculating the ROI using production_budget and worldwide_gross then adding it
df['roi'] = ((df['worldwide_gross'] - df['production_budget']) / df['production_budget']) * 100
df
```

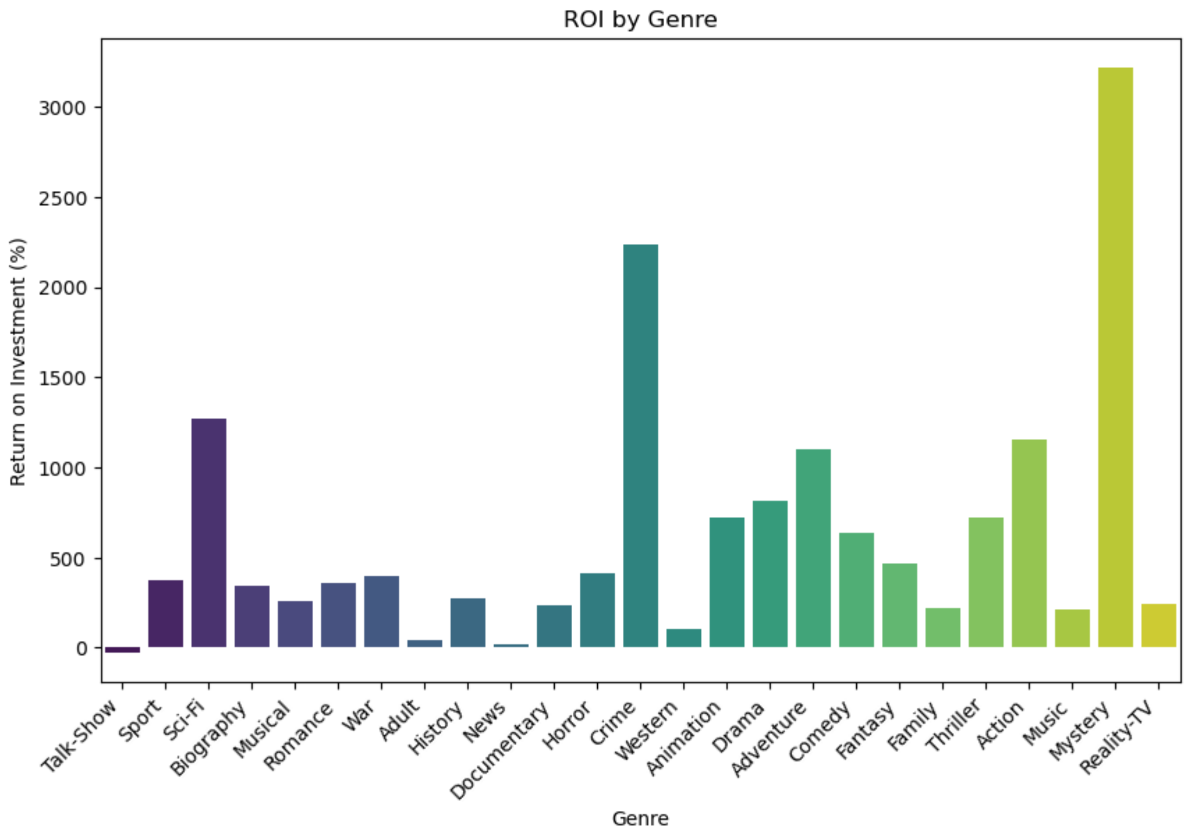
Out[16]:

	production_budget	worldwide_gross	original_title	start_year	runtime_minutes	genres	profit_margi
0	80000000.0	247023808.0	The Courier	2012	95.0	Action,Crime,Drama	67.61445
1	80000000.0	226739416.0	Pelé: Birth of a Legend	2016	107.0	Biography,Drama,Sport	64.71720
2	80000000.0	168311558.0	Bibliothèque Pascal	2010	105.0	Drama	52.46909
3	80000000.0	241200000.0	The Experiment	2010	96.0	Drama,Thriller	66.83250
4	80000000.0	221468935.0	Stadilaista tangoa etsimässä	2010	90.0	Documentary,Music	63.87755
...	...	...	...	...	...	...	
2110	7000.0	841926.0	Pet	2016	94.0	Horror,Thriller	99.16857
2111	7000.0	71644.0	Stag Night of the Dead	2010	81.0	Action,Comedy,Horror	90.22946
2112	7000.0	900.0	Mr. Nice	2010	121.0	Biography,Comedy,Crime	-677.77777
2113	6000.0	240495.0	Marley	2012	144.0	Biography,Documentary,Music	97.50514
2114	5000.0	1338.0	Welcome to the Rileys	2010	110.0	Drama	-273.69207

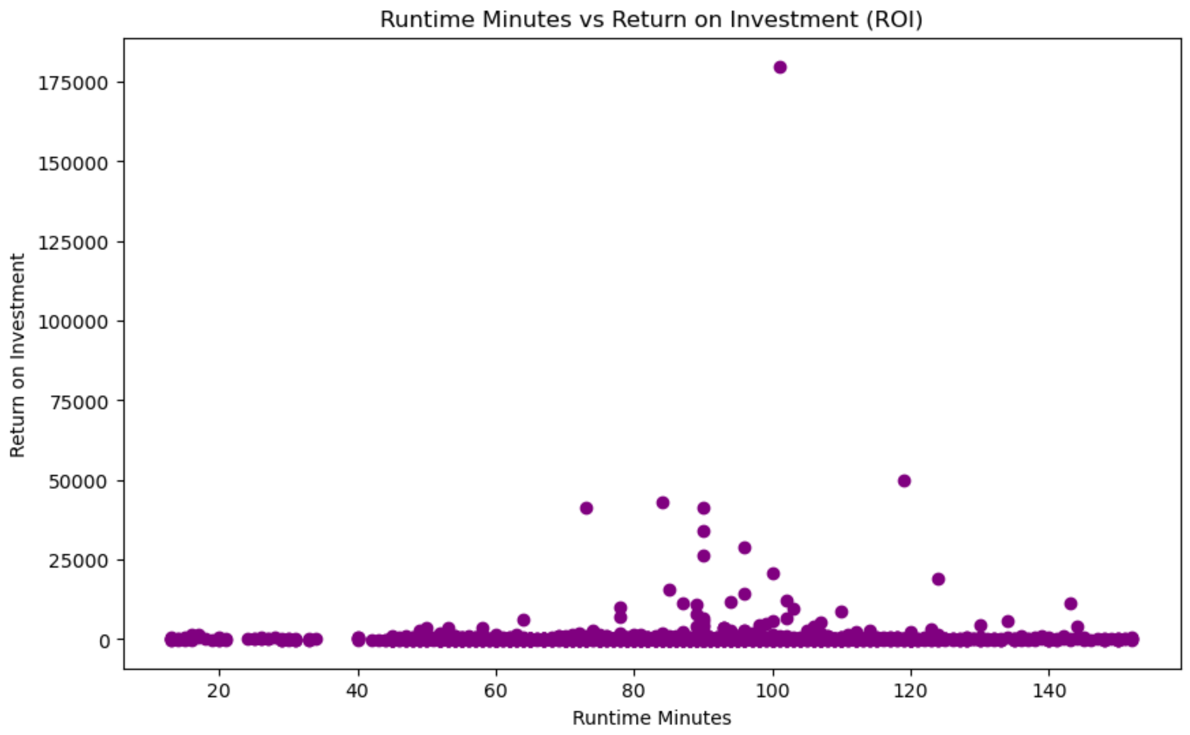
2115 rows × 9 columns

In [17]:
```
# Determining the return on investment by genre type to find out which genre should be considered for
# Splitting genres into separate rows
df_genres = df['genres'].str.split(',', expand=True).stack().reset_index(level=1, drop=True).to_frame
df_expanded = df_genres.join(df.drop('genres', axis=1))

# Create a bar plot with sorted ROI values (largest to smallest)
plt.figure(figsize=(10, 6))
sns.barplot(x='genre', y='roi', data=df_expanded, ci=None, palette='viridis', order=df_expanded.group
plt.title('ROI by Genre')
plt.xlabel('Genre')
plt.ylabel('Return on Investment (%)')
plt.xticks(rotation=45, ha='right')
plt.show()
# Top 3 genres that MS can consider to invest in are Sci-Fi, Crime, and Mystery, where Mystery movies
```

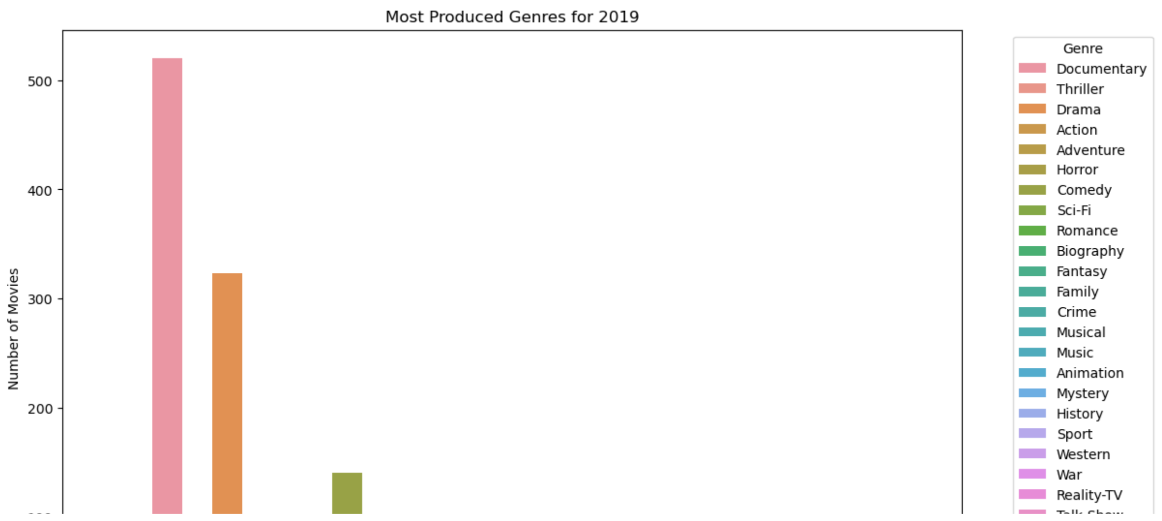
```
C:\Users\DELL\AppData\Local\Temp\ipykernel_20568\3891381107.py:8: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='genre', y='roi', data=df_expanded, ci=None, palette='viridis', order=df_expanded.gr
oupby('genre')['roi'].median().sort_values(ascending=False).index[::-1])
```



In [22]:
```python
# Determining the relationship between movie runtime minutes and ROI by scatter plot analysis
plt.figure(figsize=(10, 6))
plt.scatter(df['runtime_minutes'], df['roi'], color='purple')
plt.title('Runtime Minutes vs Return on Investment (ROI)')
plt.xlabel('Runtime Minutes')
plt.ylabel('Return on Investment')
plt.show()
# There is a higher ROI on movies whose runtime minutes are within 80 - 120 minutes
```

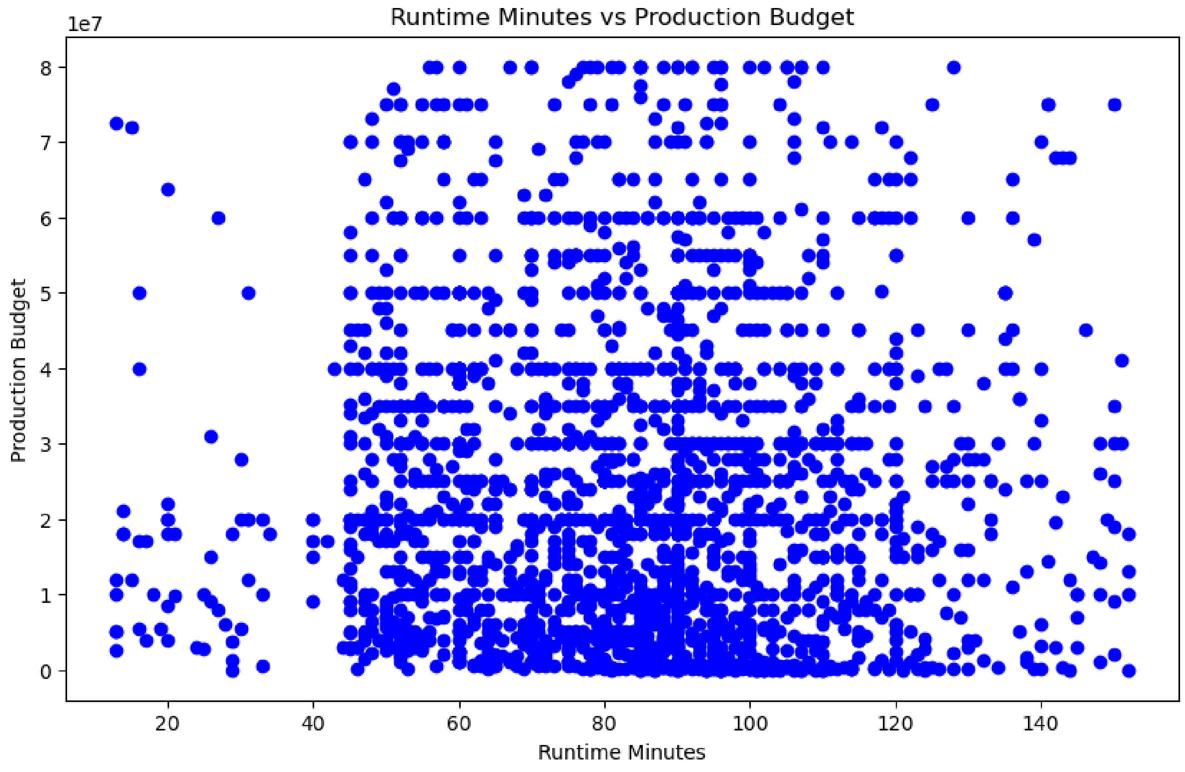


In [19]:
```python
# Determing which year had the highest production of genres
# Split genres into separate rows
df_genres = df['genres'].str.split(',', expand=True).stack().reset_index(level=1, drop=True).to_frame
df_expanded = df_genres.join(df.drop('genres', axis=1))

# Choose the topmost year
top_year = df['start_year'].value_counts().idxmax()

# Filter data for the top year
df_top_year = df_expanded[df_expanded['start_year'] == top_year]

# Create a countplot
plt.figure(figsize=(12, 8))
sns.countplot(data=df_top_year, x='start_year', hue='genre')
plt.title(f'Most Produced Genres for {top_year}')
plt.xlabel('Year')
plt.ylabel('Number of Movies')
plt.legend(title='Genre', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



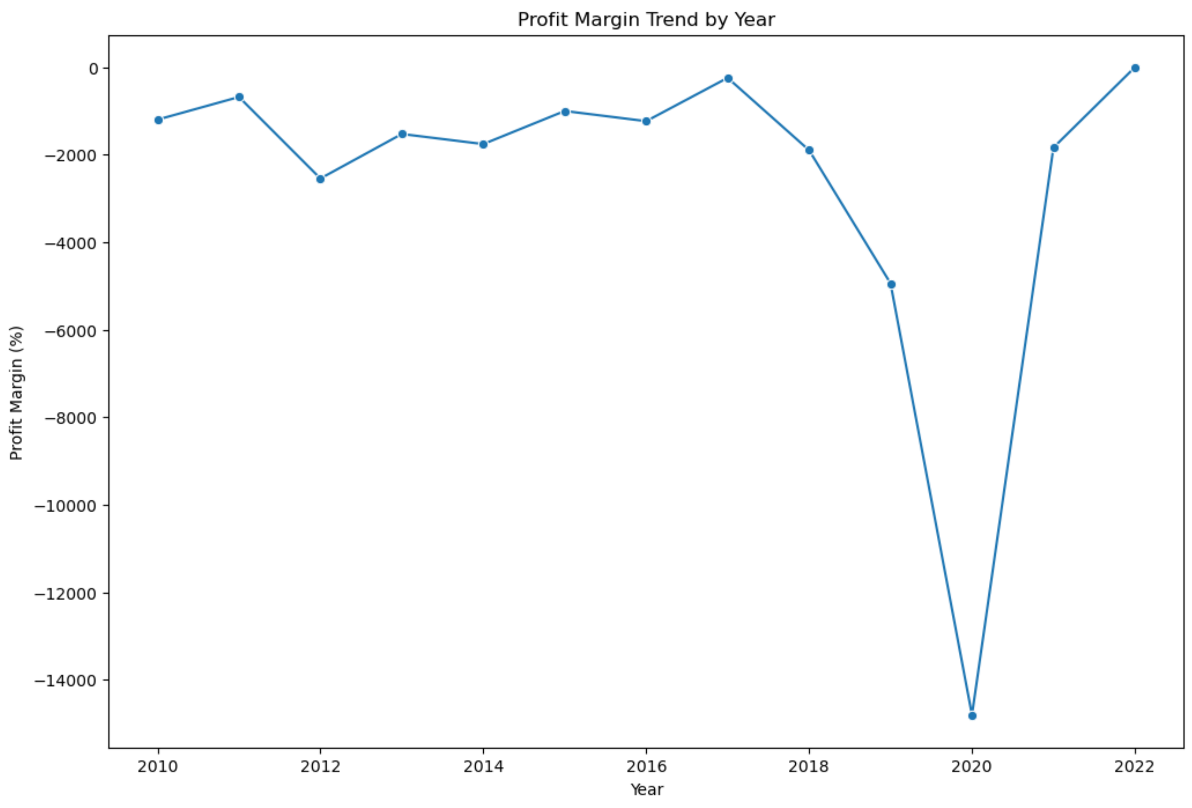
In [20]:

```
# Determining the relationship between movie runtime minutes and ROI by scatter plot analysis
plt.figure(figsize=(10, 6))
plt.scatter(df['runtime_minutes'], df['production_budget'], color='blue')
plt.title('Runtime Minutes vs Production Budget')
plt.xlabel('Runtime Minutes')
plt.ylabel('Production Budget')
plt.show();
# Shorter films have a low production budget
# Longer films have a high production budget
# Most of the films lying between 80 - 120 mins have an relatively average producton budget
```



In [21]:
```
# Determining the trend in profit margin of the movies across the years
# Create a line plot to show the profit margin trend by year
plt.figure(figsize=(12, 8))
sns.lineplot(data=df, x='start_year', y='profit_margin', marker='o', ci=None)
plt.title('Profit Margin Trend by Year')
plt.xlabel('Year')
plt.ylabel('Profit Margin (%)')
plt.show()
# A stability on profit margin is seen between years 2010 to 2018. A sharp drop in profit margin note
# and then picks at 2022. By assumption, the Covid-19 pandemic may have been the major cause of the s
# However, MS can proceed to invest in movie production as the trend shows positive increment accordi
```

```
C:\Users\DELL\AppData\Local\Temp\ipykernel_20568\466270572.py:4: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.lineplot(data=df, x='start_year', y='profit_margin', marker='o', ci=None)
```



These are the usable visualizations of the data