Author: Diana Jerusha Olulo

Student name: Diana Jerusha Olulo

Student pace: part time

Instructor name: Noah K./ William O. / Samuel G.

Overview:

This project analyzes movie data to make informed decisions for Microsoft's venture into creating a new movie studio. The analysis encompasses three key areas: determining the target worldwide gross and production budget, setting an optimal runtime for Microsoft's movies, and selecting genres based on their popularity. These decisions aim to equip Microsoft with the necessary insights to increase the likelihood of producing successful movies.

Business problem

Microsoft has made the strategic decision to enter the film industry and embark on producing their own films. In pursuit of this new venture, this analysis seeks to provide valuable advice to Microsoft by offering insights and informed decisions on how they should approach their entry into the filmmaking domain.

Questions Addressed by the Analysis:

1. What is the relationship for Microsoft's worldwide gross and production budget?
2. What should be the optimal runtime for Microsoft's movies?
3. Which genres should Microsoft prioritize for production based on their popularity?

Data Understanding

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

In [2]:
```

df1 = pd.read_csv("ZippedData/tmdb.movies.csv", index_col=0) #converting the file to a dataframe
df1.head()#print out the first five lines
```

Out[2]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count
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	7.7	10788
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610
2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368
3	[16, 35, 10751]	862	en	Toy Story	28.005	1995-11-22	Toy Story	7.9	10174
4	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186

In [3]:
```
df2 = pd.read_csv("ZippedData/title.basics.csv") #converting the file to a dataframe
df2.head()#print out the first five lines
```

Out[3]:

	tconst	primary_title	original_title	start_year	runtime_minutes	genres
0	tt0063540	Sunghursh	Sunghursh	2013	175.0	Action,Crime,Drama
1	tt0066787	One Day Before the Rainy Season	Ashad Ka Ek Din	2019	114.0	Biography,Drama
2	tt0069049	The Other Side of the Wind	The Other Side of the Wind	2018	122.0	Drama
3	tt0069204	Sabse Bada Sukh	Sabse Bada Sukh	2018	NaN	Comedy,Drama
4	tt0100275	The Wandering Soap Opera	La Telenovela Errante	2017	80.0	Comedy,Drama,Fantasy

In [4]:
```
df3 = pd.read_csv("ZippedData/tn.movie_budgets.csv") #converting the file to a dataframe
df3.head()#print out the first five lines
```

Out[4]:

	id	release_date	movie	production_budget	domestic_gross	worldwide_gross
0	1	Dec 18, 2009	Avatar	$425,000,000	$760,507,625	$2,776,345,279
1	2	May 20, 2011	Pirates of the Caribbean: On Stranger Tides	$410,600,000	$241,063,875	$1,045,663,875
2	3	Jun 7, 2019	Dark Phoenix	$350,000,000	$42,762,350	$149,762,350
3	4	May 1, 2015	Avengers: Age of Ultron	$330,600,000	$459,005,868	$1,403,013,963
4	5	Dec 15, 2017	Star Wars Ep. VIII: The Last Jedi	$317,000,000	$620,181,382	$1,316,721,747

Data Merging

In [5]:
```
merged_df = pd.merge(df1, df2)# merging df1 and df2
merged_df.head()#printing the first five lines
```

Out[5]:

	genre_ids	id	original_language	original_title	popularity	release_date	title	vote_average	vote_count	tconst	primary_
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	7.7	10788	tt0926084	Harry Po and Dea Hallows:
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	tt0892769	How to T Your Dra
2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	tt1228705	Iron Ma
3	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	tt1375666	Incep
4	[12, 14, 10751]	32657	en	Percy Jackson & the Olympians: The Lightning T...	26.691	2010-02-11	Percy Jackson & the Olympians: The Lightning T...	6.1	4229	tt0814255	P Jackso Olympi Lightning

In [6]:
```
joined_df = merged_df.merge(df3, left_index=True, right_index=True, how='outer')
joined_df.head()
```

							Part 1				
1	[14, 12, 16, 10751]	10191	en	How to Train Your Dragon	28.734	2010-03-26	How to Train Your Dragon	7.7	7610	tt0892769	Ho Yo
2	[12, 28, 878]	10138	en	Iron Man 2	28.515	2010-05-07	Iron Man 2	6.8	12368	tt1228705	I
3	[28, 878, 12]	27205	en	Inception	27.920	2010-07-16	Inception	8.3	22186	tt1375666	
4	[12, 14, 10751]	32657	en	Percy Jackson & the Olympians: The Lightning T...	26.691	2010-02-11	Percy Jackson & the Olympians: The Lightning T...	6.1	4229	tt0814255	C Lig

In [7]:
```
joined_df.shape
```

Out[7]: (21078, 20)

Data cleaning

In [8]:
```
df=joined_df
df.shape
```

Out[8]: (21078, 20)

In [9]:
```
df.columns
```

Out[9]:
```
Index(['genre_ids', 'id_x', 'original_language', 'original_title',
       'popularity', 'release_date_x', 'title', 'vote_average', 'vote_count',
       'tconst', 'primary_title', 'start_year', 'runtime_minutes', 'genres',
       'id_y', 'release_date_y', 'movie', 'production_budget',
       'domestic_gross', 'worldwide_gross'],
      dtype='object')
```

In [10]:
```
#dropping uneccessary columns
df.drop([ 'genre_ids', 'original_title', 'release_date_x', 'id_x', 'id_y', 'release_date_y', 'domestic_gross',
```

In [11]:
```
df.shape
```

Out[11]:
```
(21078, 12)
```

In [12]:
```
#checking the information about the dataseet ie. no of rows, columns, datatype, memory usage, null valuesetc
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 21078 entries, 0 to 21077
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   original_language  21078 non-null  object 
 1   popularity         21078 non-null  float64
 2   title              21078 non-null  object 
 3   vote_average       21078 non-null  float64
 4   tconst             21078 non-null  object 
 5   primary_title      21078 non-null  object 
 6   start_year         21078 non-null  int64  
 7   runtime_minutes    19452 non-null  float64
 8   genres             20779 non-null  object 
 9   movie              5782 non-null   object 
 10  production_budget  5782 non-null   object 
 11  worldwide_gross    5782 non-null   object 
dtypes: float64(3), int64(1), object(8)
memory usage: 2.1+ MB
```

In [13]:
```
#checking the proportiion of missing values
df.isnull().mean()
```

Out[13]:
```
original_language    0.000000
popularity           0.000000
title                0.000000
vote_average         0.000000
tconst               0.000000
primary_title        0.000000
start_year           0.000000
runtime_minutes      0.077142
genres               0.014185
movie                0.725686
production_budget    0.725686
worldwide_gross      0.725686
dtype: float64
```

In [14]:
```
#dropping missing values in runtime_minutes since the percentage is small
df = df.dropna(subset=['runtime_minutes', 'genres'])
```

In [46]:
```
#replacing missing values with the word missing
df['movie'] = df['movie'].fillna('missing')
warnings.filterwarnings('ignore')
```

In [16]:
```
df['movie'].isna().mean()
```

Out[16]:
```
0.0
```

In [17]:
```
# removing dollar signs from the column
df['production_budget'] = df['production_budget'].str.replace('$', '')
#removing commas
df['production_budget'] = df['production_budget'].str.replace(',', '')
#filling missing values with zero
df['production_budget'] = df['production_budget'].fillna(0)
# Convert column 'A' to integer type
df['production_budget'] = df['production_budget'].astype(int)#changing the data type
warnings.filterwarnings('ignore', category= RuntimeWarning)
```

In [18]:
```
df['worldwide_gross'] = df['worldwide_gross'].str.replace('$', '')
df['worldwide_gross'] = df['worldwide_gross'].str.replace(',', '')
df['worldwide_gross'] = df['worldwide_gross'].fillna(0)

# Convert column 'A' to integer type
df['worldwide_gross'] = df['worldwide_gross'].astype(float)
```

In [19]:
```
df.isna().mean()#checking for missing values
```

Out[19]:
```
original_language    0.0
popularity           0.0
title                0.0
vote_average         0.0
tconst               0.0
primary_title        0.0
start_year           0.0
runtime_minutes      0.0
genres               0.0
movie                0.0
production_budget    0.0
worldwide_gross      0.0
dtype: float64
```

In [20]:
```
#checking for duplicates in any columns
df.duplicated().any()
```

Out[20]:
```
True
```

In [21]:
```
#dropping duplicate values and keeping the first entry among the duplicates
df.drop_duplicates(inplace=True, keep='first')
```

In [22]:
```
#Getting to knwo the statistics summary of the data
df.describe()
```

Out[22]:

	popularity	vote_average	start_year	runtime_minutes	production_budget	worldwide_gross
count	18467.00000	18467.000000	18467.000000	18467.000000	1.846700e+04	1.846700e+04
mean	3.80749	5.750615	2014.177939	91.356961	8.888205e+06	2.600016e+07
std	4.91130	1.733414	2.522028	25.494119	2.666385e+07	1.037226e+08
min	0.60000	0.000000	2010.000000	1.000000	0.000000e+00	0.000000e+00
25%	0.65400	4.900000	2012.000000	83.000000	0.000000e+00	0.000000e+00
50%	1.72600	5.900000	2014.000000	90.000000	0.000000e+00	0.000000e+00
75%	5.64500	6.800000	2016.000000	100.000000	1.250000e+06	8.639550e+04
max	80.77300	10.000000	2020.000000	1834.000000	4.250000e+08	2.776345e+09

In [23]:
```
df = df[(df['production_budget'] != 0)]# removing rows with the missing values which are denoted by zero
df = df[(df['worldwide_gross'] != 0)]
```

In [24]:
```
df.shape# checking the numer of rows and columns available in hte dataset
```

Out[24]:
```
(4853, 12)
```

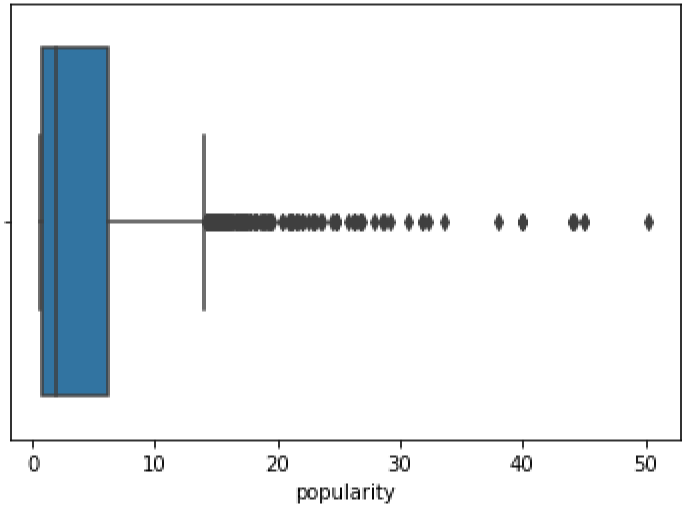
In [25]:
```
# Position of the Outlier values
x = np.where(df['runtime_minutes']>150)
print(x)
```

```
(array([ 261,  322,  391,  409,  473,  474,  478,  479,  604,  876,  919,
         953,  957,  961, 1515, 1520, 1522, 1700, 1796, 1839, 1909, 1914,
        1919, 1924, 1929, 1979, 2009, 2074, 2075, 2096, 2098, 2107, 2271,
        2364, 2492, 2605, 2612, 2683, 2687, 2786, 2828, 2832, 2834, 2939,
        2943, 2946, 2950, 3007, 3129, 3390, 3408, 3492, 3506, 3510, 3568,
        3844, 3849, 3853, 3882, 3946, 4135, 4361, 4392, 4764, 4770, 4775,
        4835], dtype=int64),)
```

plotting the boxplot that vividly shows the outliers

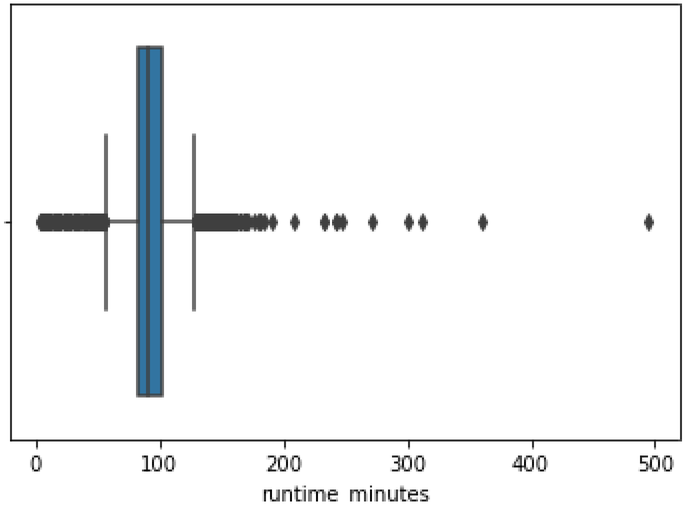
In [26]:
```
sns.boxplot(df['popularity'])# plotting the boxplot that vividly shows the outliers
```

Out[26]: <AxesSubplot:xlabel='popularity'>



In [27]:
```
sns.boxplot(df['runtime_minutes'])
```

Out[27]: <AxesSubplot:xlabel='runtime_minutes'>



Removing Outliers

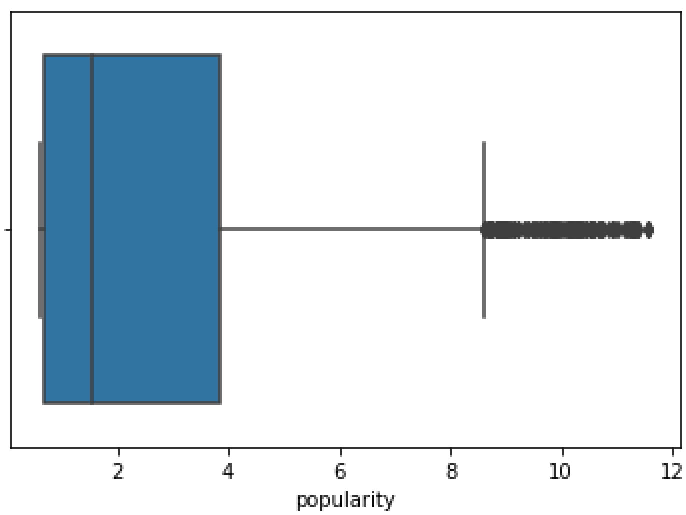
In [28]:
```
#Creating a function to remove the ouliers using the IQR method
def remove_outliers_iqr(df, column):
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
```

In [29]:
```
# Remove outliers in the following columns using the function created above
df = remove_outliers_iqr(df, 'runtime_minutes')
df = remove_outliers_iqr(df, 'production_budget')
df = remove_outliers_iqr(df, 'worldwide_gross')
df = remove_outliers_iqr(df, 'popularity')
```

Visualisations after removing the outliers

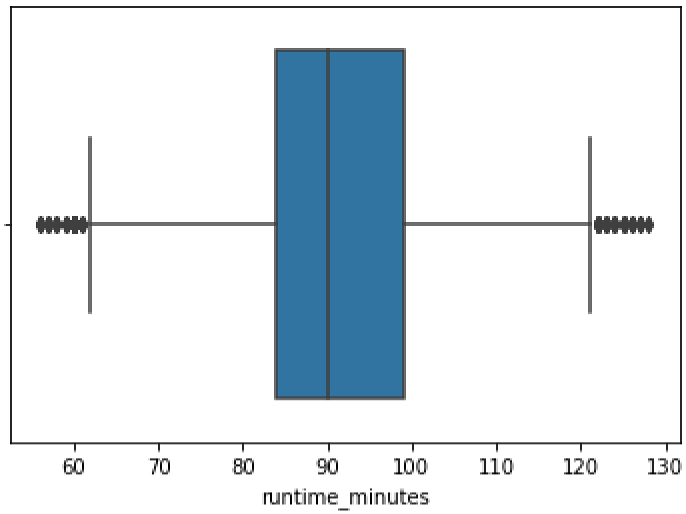
In [30]:
```
sns.boxplot(df['popularity'])
```

Out[30]: <AxesSubplot:xlabel='popularity'>



In [31]:
```
sns.boxplot(df['runtime_minutes'])
```

Out[31]: <AxesSubplot:xlabel='runtime_minutes'>



In [32]:
```
df.describe()
```

Out[32]:

	popularity	vote_average	start_year	runtime_minutes	production_budget	worldwide_gross
count	3454.000000	3454.000000	3454.000000	3454.000000	3.454000e+03	3.454000e+03
mean	2.803139	5.559873	2012.102490	91.269543	2.002213e+07	3.896244e+07
std	2.749238	1.621166	2.355448	12.667780	1.990075e+07	4.559862e+07
min	0.600000	0.000000	2010.000000	56.000000	5.000000e+03	2.600000e+01
25%	0.665250	4.800000	2011.000000	84.000000	5.000000e+06	4.024722e+06
50%	1.552000	5.600000	2011.000000	90.000000	1.400000e+07	2.015804e+07
75%	3.836000	6.600000	2012.000000	99.000000	3.000000e+07	5.891821e+07
max	11.571000	10.000000	2020.000000	128.000000	9.000000e+07	1.879959e+08

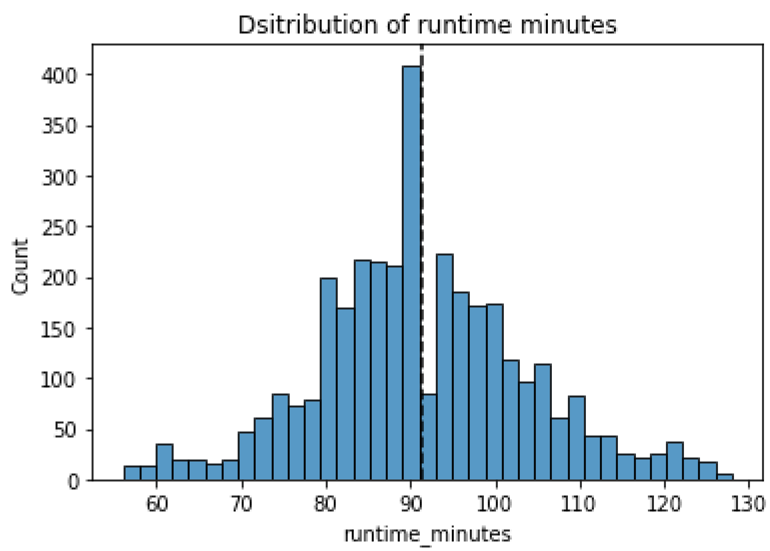
In [33]:
```
df.shape
```

Out[33]: (3454, 12)

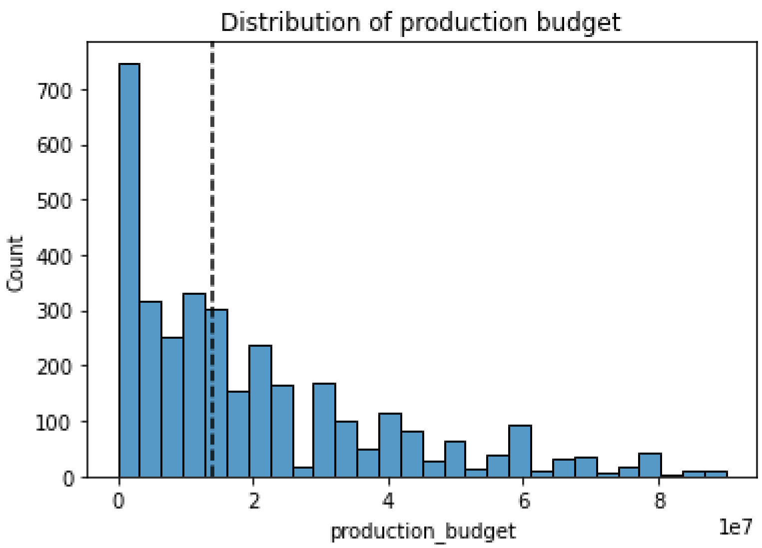
Data Visualization

Univariate Analysis

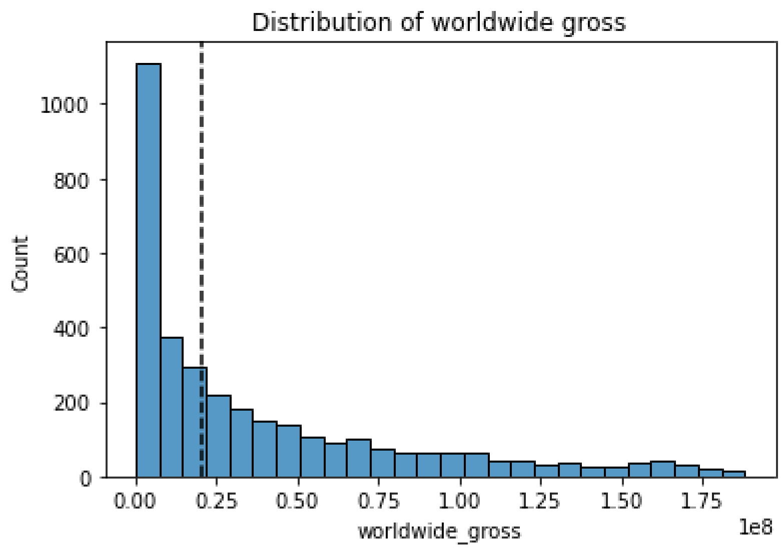
In [34]:
```
#Average duration of a movie that people tend to watch
sns.histplot(data=df['runtime_minutes'],bins='auto') # creates a histogram using Seaborn's histplot() functior
average_runtime_minutes = df['runtime_minutes'].mean() #calculates the mean of the runtime_minutes column
#This line below adds a vertical line at the position of the mean runtime
plt.axvline(average_runtime_minutes, color='black', linestyle='dashed', linewidth=1.5, label=f'Mean Runtime: {
plt.title('Dsitribution of runtime minutes');
```



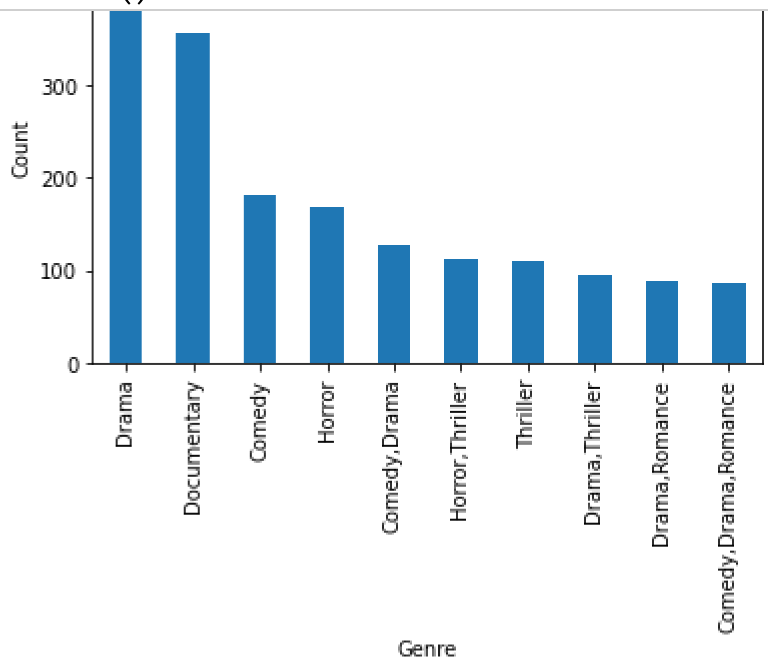
In [35]:
```
sns.histplot(data=df['production_budget'],bins='auto')
average_production_budget = df['production_budget'].median()
plt.axvline(average_production_budget, color='black', linestyle='dashed', linewidth=1.5, label=f'Mean Runtime:
plt.title('Distribution of production budget');
```



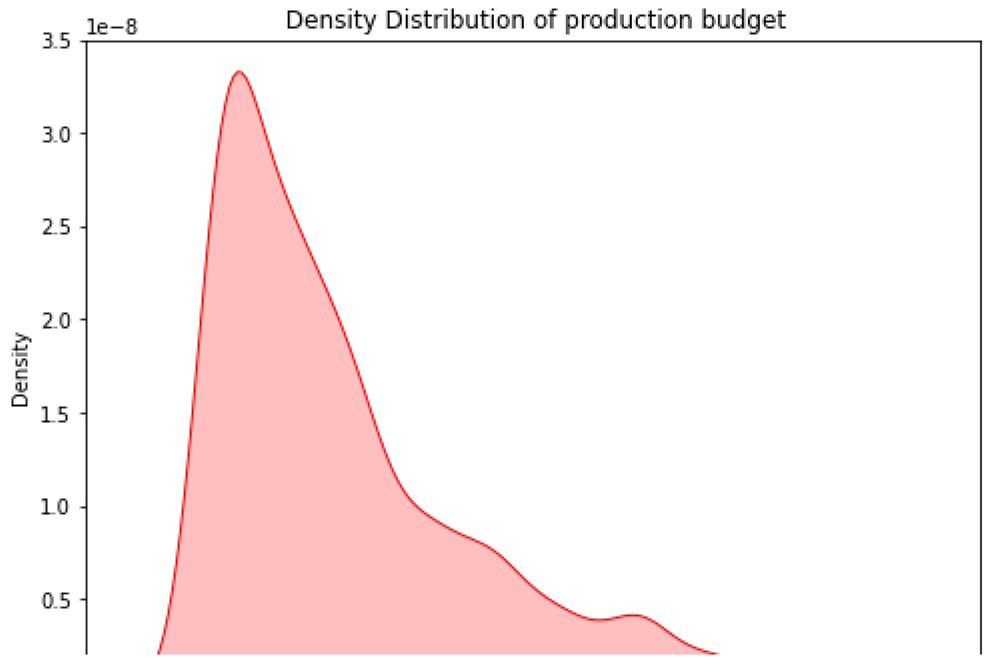
In [36]:
```
sns.histplot(data=df['worldwide_gross'],bins='auto')
average_worldwide_gross = df['worldwide_gross'].median()
plt.axvline(average_worldwide_gross, color='black', linestyle='dashed', linewidth=1.5, label=f'Mean Runtime: {
plt.title('Distribution of worldwide gross');
```



In [37]:
```
# Get the count of each genre
genre_counts = df['genres'].value_counts()

# Get the top 10 genres
top_10_genres = genre_counts.head(10)

# Create a bar plot of the top 10 genres
top_10_genres.plot(kind='bar') # plotting a bar graph

# Add a title and labels to the plot
plt.title('Frequency distribution of different genres')
plt.xlabel('Genre')
plt.ylabel('Count')

# Show the plot
plt.show()
```

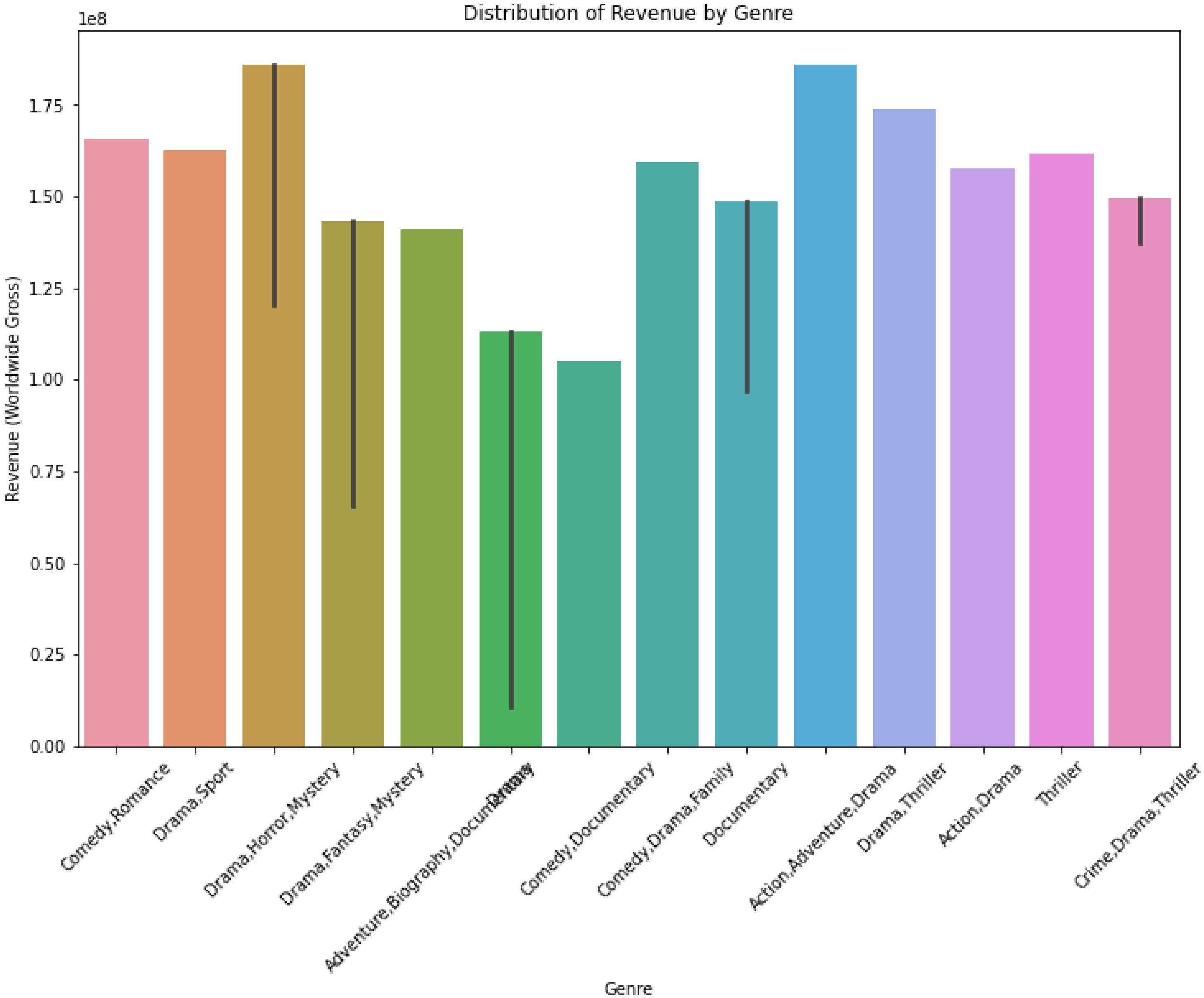


In [38]:
```
# Plot density plot
plt.figure(figsize=(8, 6))
sns.kdeplot(df['production_budget'], shade=True, color='red') # creates a kernel density estimate (KDE) plot
plt.title(' Density Distribution of production budget')
plt.xlabel('production budget')
plt.ylabel('Density')
plt.show() # displays the plot
```

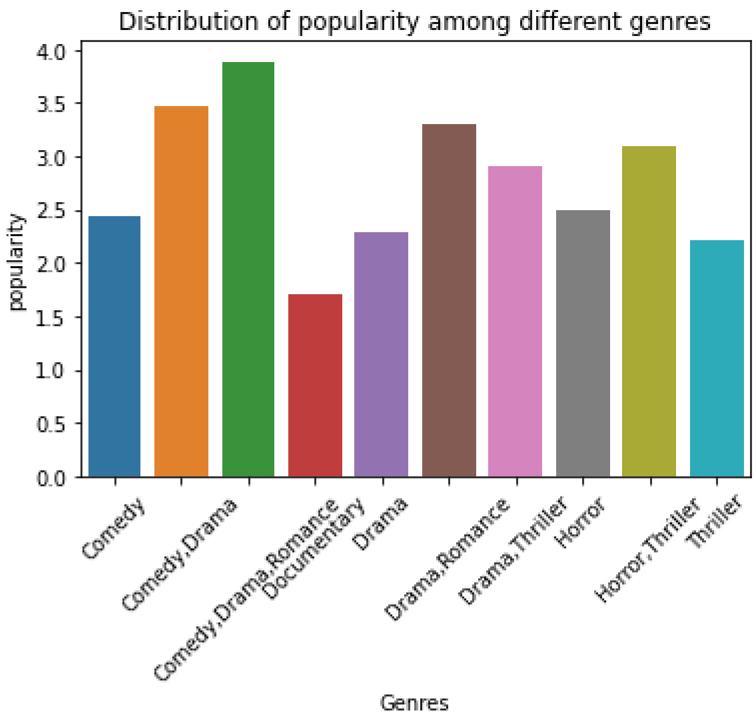


Bivariate Analysis

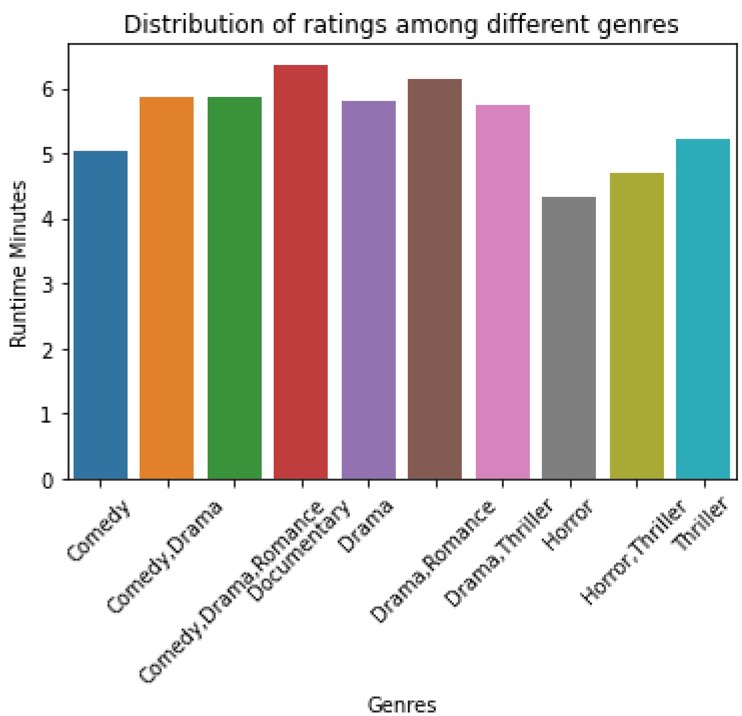
In [39]:
```
#Which genres tend to have the highest revenue?
# Plotting bar plot
genre = df['genres'].head(20)
plt.figure(figsize=(12, 8))
sns.barplot(x=genre, y='worldwide_gross', data=df, estimator=max)  # Use max as the estimator to show the high
plt.title('Distribution of Revenue by Genre')
plt.xlabel('Genre')
plt.ylabel('Revenue (Worldwide Gross)')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```


Distribution of Revenue by Genre
1e8
Revenue (Worldwide Gross)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Comedy,Romance
Drama,Sport
Drama,Horror,Mystery
Drama,Fantasy,Mystery
Adventure,Biography,Documentary
Drama
Comedy,Documentary
Comedy,Drama,Family
Documentary
Action,Adventure,Drama
Drama,Thriller
Action,Drama
Thriller
Crime,Drama,Thriller
Genre

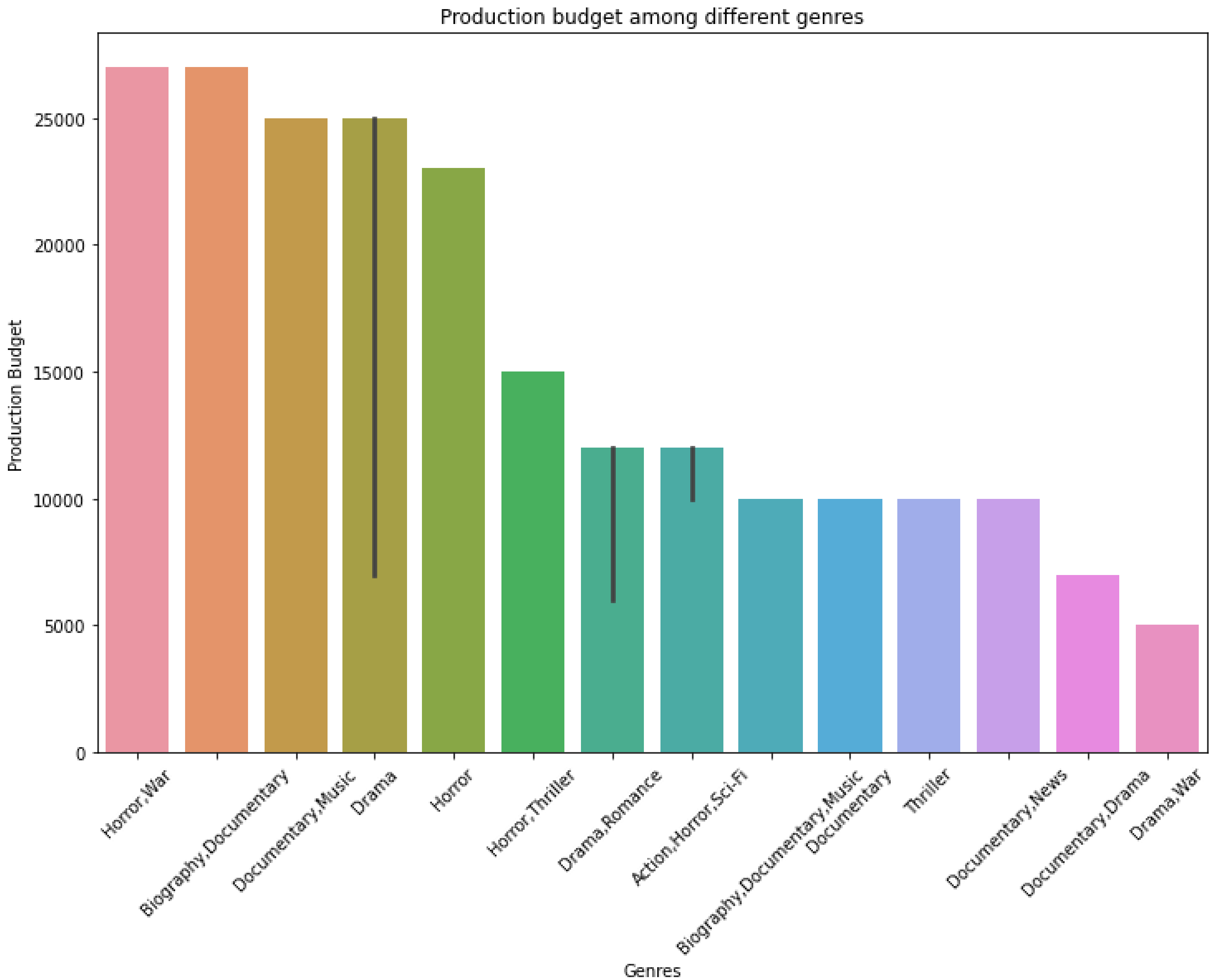
In [40]:
```
# Get the top 10 genres by frequency
top_genres = df['genres'].value_counts().head(10).index.tolist()

# Filter the dataframe to only include rows with one of the top 10 genres
top_genre_df = df[df['genres'].isin(top_genres)]

# Group the data by genres and calculate the mean runtime for each genre
genre_means = top_genre_df.groupby('genres')['popularity'].mean()

# Create a bar plot to visualize the mean runtime for each genre
sns.barplot(x=genre_means.index, y=genre_means.values)
plt.title('Distribution of popularity among different genres')
plt.xlabel('Genres')
plt.ylabel('popularity')
plt.xticks(rotation=45)
plt.show()
```



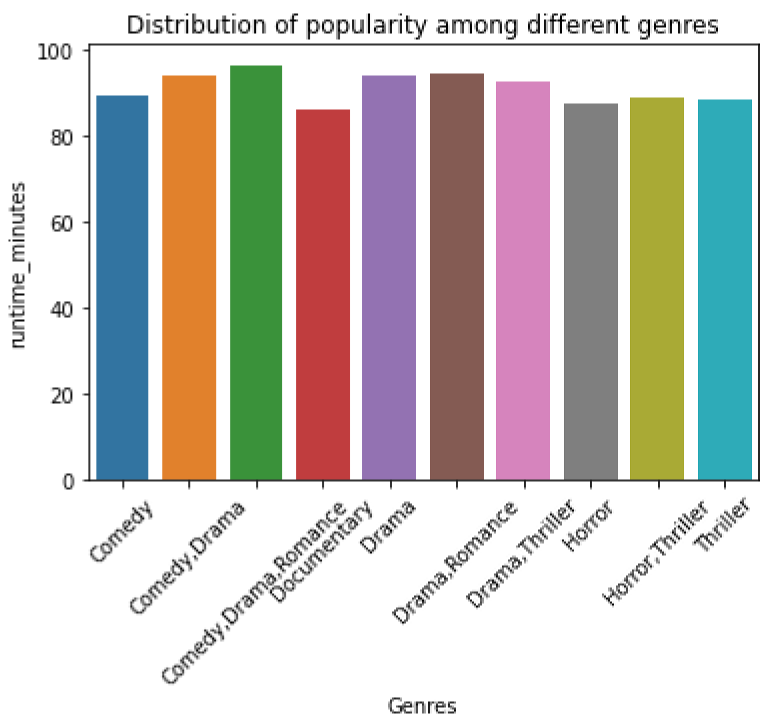
In [41]:
```
#Highest rated genre
# Get the top 10 genres by frequency
top_genres = df['genres'].value_counts().head(10).index.tolist()

# Filter the dataframe to only include rows with one of the top 10 genres
top_genre_df = df[df['genres'].isin(top_genres)]

# Group the data by genres and calculate the voteaverage for each genre
genre_means = top_genre_df.groupby('genres')['vote_average'].mean()

# Create a bar plot to visualize the mean runtime for each genre
sns.barplot(x=genre_means.index, y=genre_means.values)
plt.title('Distribution of ratings among different genres')
plt.xlabel('Genres')
plt.ylabel('Runtime Minutes')
plt.xticks(rotation=45)
plt.show()
```



In [42]:
```
# Plotting bar plot
genre = df['genres'].tail(20)
plt.figure(figsize=(12, 8))
sns.barplot(x=genre, y='production_budget', data=df, estimator=max)  # Use max as the estimator to show the hi
plt.title('Production budget among different genres')
plt.xlabel('Genres')
plt.ylabel('Production Budget')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```


Production budget among different genres
Production Budget
0
5000
10000
15000
20000
25000
Horror,War
Biography,Documentary
Documentary,Music
Drama
Horror
Horror,Thriller
Drama,Romance
Action,Horror,Sci-Fi
Biography,Documentary,Music
Documentary
Thriller
Documentary,News
Documentary,Drama
Drama,War
Genres

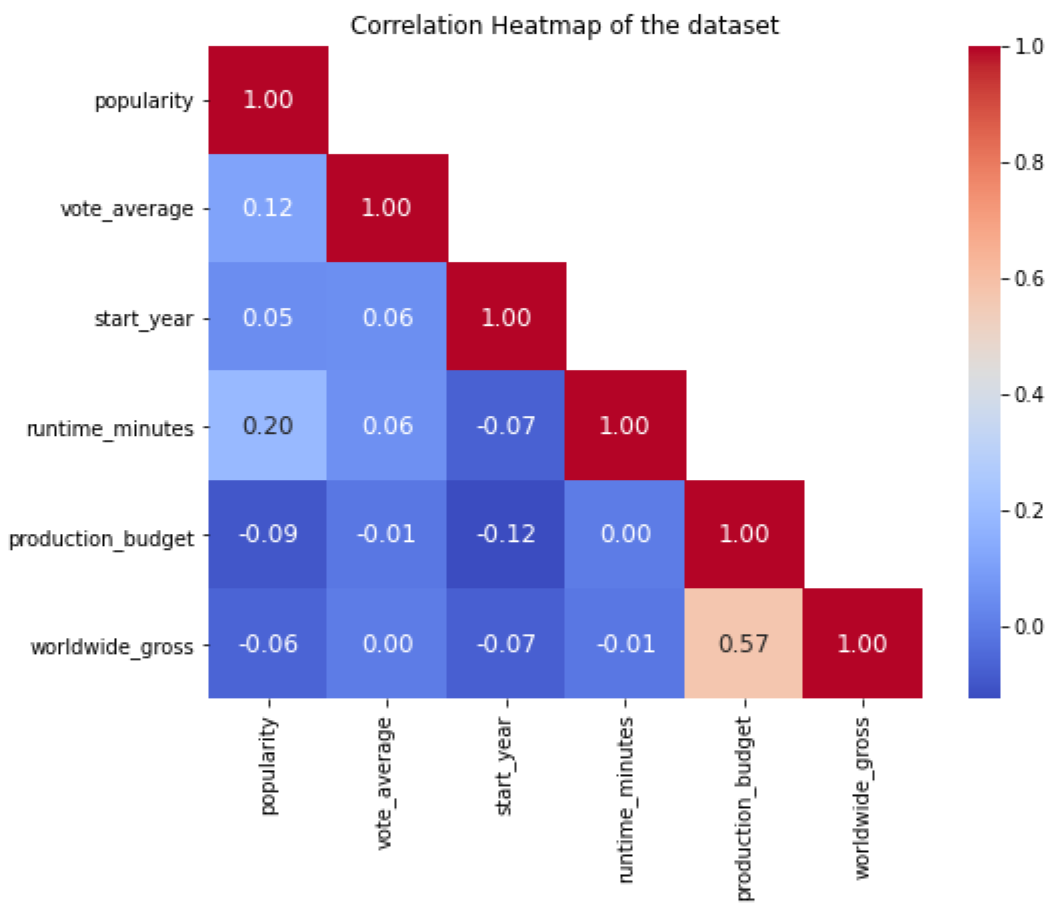
In [43]:
```
#Relationship between popularity and genre
#Highest rated genre
# Get the top 10 genres by frequency
top_genres = df['genres'].value_counts().head(10).index.tolist()

# Filter the dataframe to only include rows with one of the top 10 genres
top_genre_df = df[df['genres'].isin(top_genres)]

# Group the data by genres and calculate the voteaverage for each genre
genre_means = top_genre_df.groupby('genres')['runtime_minutes'].mean()

# Create a bar plot to visualize the mean runtime for each genre
sns.barplot(x=genre_means.index, y=genre_means.values)
plt.title('Distribution of popularity among different genres')
plt.xlabel('Genres')
plt.ylabel('runtime_minutes')
plt.xticks(rotation=45)
plt.show()
```



In [ ]:

Multivariate Analysis

In [44]:
```
# Calculate correlation matrix
correlation_matrix = df.corr()
# Create a mask to hide the upper triangle
mask = np.triu(np.ones_like(correlation_matrix), k=1)
# Plot heatmap
plt.figure(figsize=(8, 6))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", annot_kws={"size": 12}, mask=mask)
plt.title('Correlation Heatmap of the dataset')
plt.show()
```



In [47]:
```
joined_df.to_csv('cleaned_df.csv')#exporting data
```

Conclusion

The analysis leads to the following recommendations:

1. Microsoft should aim for movie runtimes between 90 and 125 minutes.
2. The production budget should fall within the range of 15 to 30 million dollars.
3. Microsoft should prioritize producing more comedy, drama, action, and adventure movies, as they are popular and generate high revenue.