DSC Phase One Project

Importing Libraries

In [15]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sqlite3
```

Connecting to SQLite and Loading Tables

In [16]:
```
import sqlite3
import pandas as pd

# Connect to the SQLite database and load relevant tables
dataset_folder_path = r"C:\Users\PC\Desktop\Moringa Projects\DSC_Projects\phase

# Connect to the SQLite database
conn = sqlite3.connect(f'{dataset_folder_path}/im.db')

# List all tables in the database
tables = pd.read_sql("SELECT name FROM sqlite_master WHERE type='table';", conn
print("Tables in the database:")
print(tables)

# Load the movie_basics and movie_ratings tables if they exist
if 'movie_basics' in tables['name'].values:
    movie_basics = pd.read_sql("SELECT * FROM movie_basics", conn)
    print("\nMovie Basics Columns (IMDB)")
    print(movie_basics.columns)
else:
    print("Table 'movie_basics' does not exist in the database.")

if 'movie_ratings' in tables['name'].values:
    movie_ratings = pd.read_sql("SELECT * FROM movie_ratings", conn)
    print("\nMovie Ratings Columns (IMDB)")
    print(movie_ratings.columns)
else:
    print("Table 'movie_ratings' does not exist in the database.")

# Close the connection
conn.close()
```

```
Tables in the database:
           name
0   movie_basics
1      directors
2      known_for
3     movie_akas
4  movie_ratings
5        persons
6     principals
7        writers

Movie Basics Columns (IMDB)
Index(['movie_id', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres'],
      dtype='object')

Movie Ratings Columns (IMDB)
Index(['movie_id', 'averagerating', 'numvotes'], dtype='object')
```

Loading Datasets

In [17]:
```
#Path to the datasets folder

dataset_folder_path = r"C:\Users\PC\Desktop\Moringa Projects\DSC_Projects\phase

bom_movie_gross = pd.read_csv(f'{dataset_folder_path}/bom.movie_gross.csv')
tmdb_movies = pd.read_csv(f'{dataset_folder_path}/tmdb.movies.csv')
tn_movie_budgets = pd.read_csv(f'{dataset_folder_path}/tn.movie_budgets.csv')
rt_movie_info = pd.read_csv(f'{dataset_folder_path}/rt.movie_info.tsv', sep='\t
rt_reviews = pd.read_csv(f'{dataset_folder_path}/rt.reviews.tsv', sep='\t', enc
```

Merging Datasets

In [18]:
```
# Merge movie_basics with movie_ratings on the correct key
imdb_data = pd.merge(movie_basics, movie_ratings, on='movie_id', how='left')

print("\nIMDB Data Columns")
print(imdb_data.columns)
```

```
IMDB Data Columns
Index(['movie_id', 'primary_title', 'original_title', 'start_year',
       'runtime_minutes', 'genres', 'averagerating', 'numvotes'],
      dtype='object')
```

Inspecting Columns and Displaying Initial Rows

In [19]:
```
# Display the first few rows of each dataset
print("BOM Movie Gross")
print(bom_movie_gross.head())
print("\nTMDB Movies")
print(tmdb_movies.head())
print("\nTN Movie Budgets")
print(tn_movie_budgets.head())
print("\nRT Movie Info")
print(rt_movie_info.head())
print("\nRT Reviews")
print(rt_reviews.head())
print("\nMovie Basics (IMDB)")
print(movie_basics.head())
print("\nMovie Ratings (IMDB)")
print(movie_ratings.head())
print("\nIMDB Data (Merged)")
print(imdb_data.head())
```

```
BOM Movie Gross
                                              title studio  domestic_gross  \
0                                     Toy Story 3     BV     415000000.0
1                       Alice in Wonderland (2010)     BV     334200000.0
2  Harry Potter and the Deathly Hallows Part 1     WB     296000000.0
3                                       Inception     WB     292600000.0
4                           Shrek Forever After   P/DW     238700000.0

  foreign_gross  year
0     652000000  2010
1     691300000  2010
2     664300000  2010
3     535700000  2010
4     513900000  2010

TMDB Movies
   Unnamed: 0            genre_ids     id original_language  \
0           0      [12, 14, 10751]  12444                en
1           1  [14, 12, 16, 10751]  10191                en
```

Cleaning TN Movie Budgets

In [20]:
```
# Clean tn_movie_budgets
tn_movie_budgets['production_budget'] = tn_movie_budgets['production_budget'].r
tn_movie_budgets['domestic_gross'] = tn_movie_budgets['domestic_gross'].replace
tn_movie_budgets['worldwide_gross'] = tn_movie_budgets['worldwide_gross'].repla
```

Merging Datasets - bom_movie_gross and tn_movie_budgets

In [21]:
```
# Mergeing bom_movie_gross with tn_movie_budgets on movie title
merged_data = pd.merge(tn_movie_budgets, bom_movie_gross, left_on='movie', righ
```

Cleaning TMDB Movies

In [22]:
```
# Extracting and cleaning genre information from tmdb_movies
def convert_genre_ids(genre_ids):
    if isinstance(genre_ids, str):
        return eval(genre_ids)
    return genre_ids

tmdb_movies['genre_ids'] = tmdb_movies['genre_ids'].apply(convert_genre_ids)
```

Genre Analysis

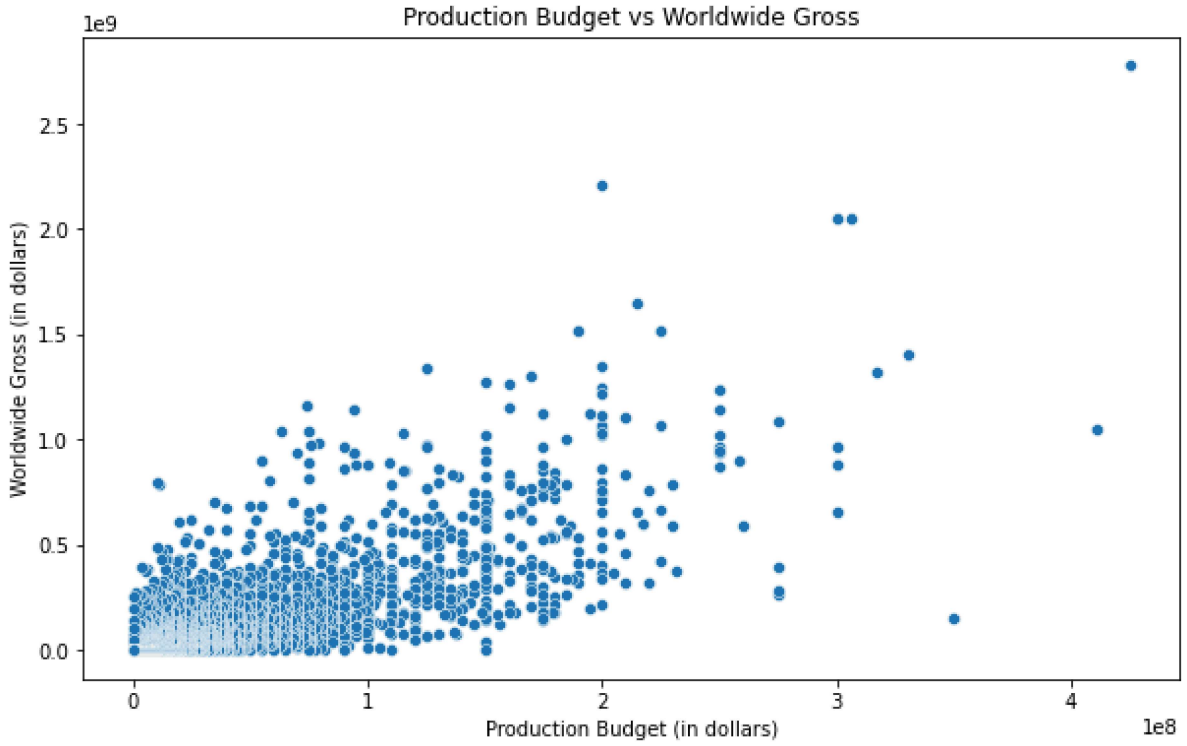
In [23]:
```
# Flatten genre_ids and count occurrences for genre analysis
all_genres = [genre for sublist in tmdb_movies['genre_ids'] for genre in sublis
genre_counts = pd.Series(all_genres).value_counts()
```

Average Rating by Genre

In [24]:
```
# Calculating average rating by genre using TMDB data only
average_ratings_by_genre = tmdb_movies.explode('genre_ids').groupby('genre_ids
average_ratings_by_genre = average_ratings_by_genre.sort_values(by='vote_averag
```

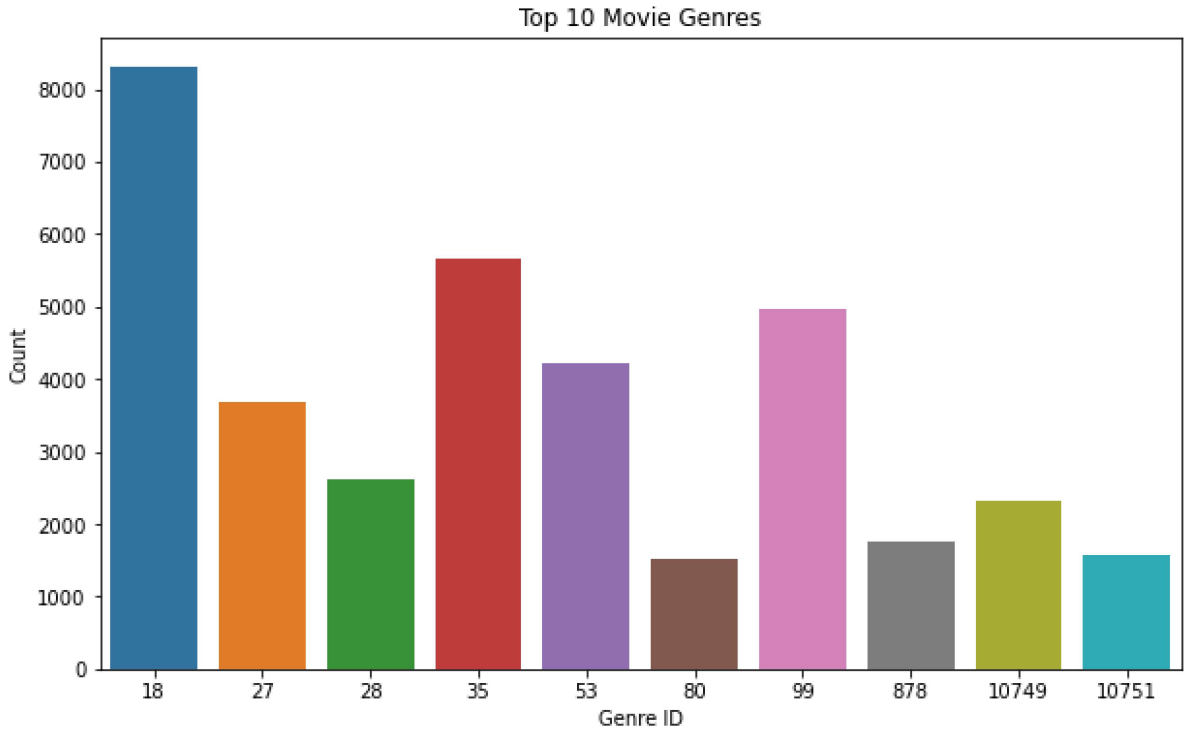
Visualization - Scatter Plot of Production Budget vs Worldwide Gross

In [25]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

#Scatter Plot of Production Budget vs Worldwide Gross
plt.figure(figsize=(10, 6))
sns.scatterplot(data=merged_data, x='production_budget', y='worldwide_gross')
plt.title('Production Budget vs Worldwide Gross')
plt.xlabel('Production Budget (in dollars)')
plt.ylabel('Worldwide Gross (in dollars)')
plt.savefig('Images/scatter_plot_budget_vs_gross.png')
plt.show()
```



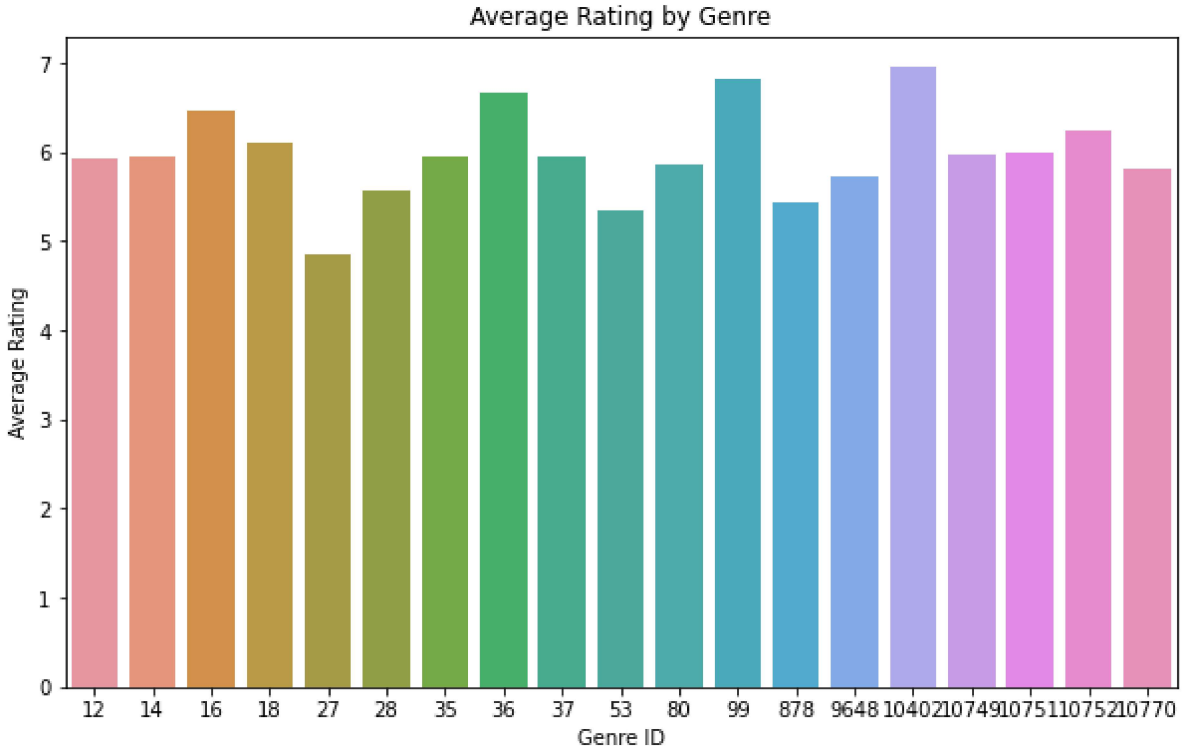
Visualization - Bar Plot of Top 10 Movie Genres

In [26]:
```
# Visualization - Bar Plot of Top 10 Movie Genres
plt.figure(figsize=(10, 6))
sns.barplot(x=genre_counts.index[:10], y=genre_counts.values[:10])
plt.title('Top 10 Movie Genres')
plt.xlabel('Genre ID')
plt.ylabel('Count')
plt.savefig('Images/bar_plot_top_10_movie_genres.png')
plt.show()
```



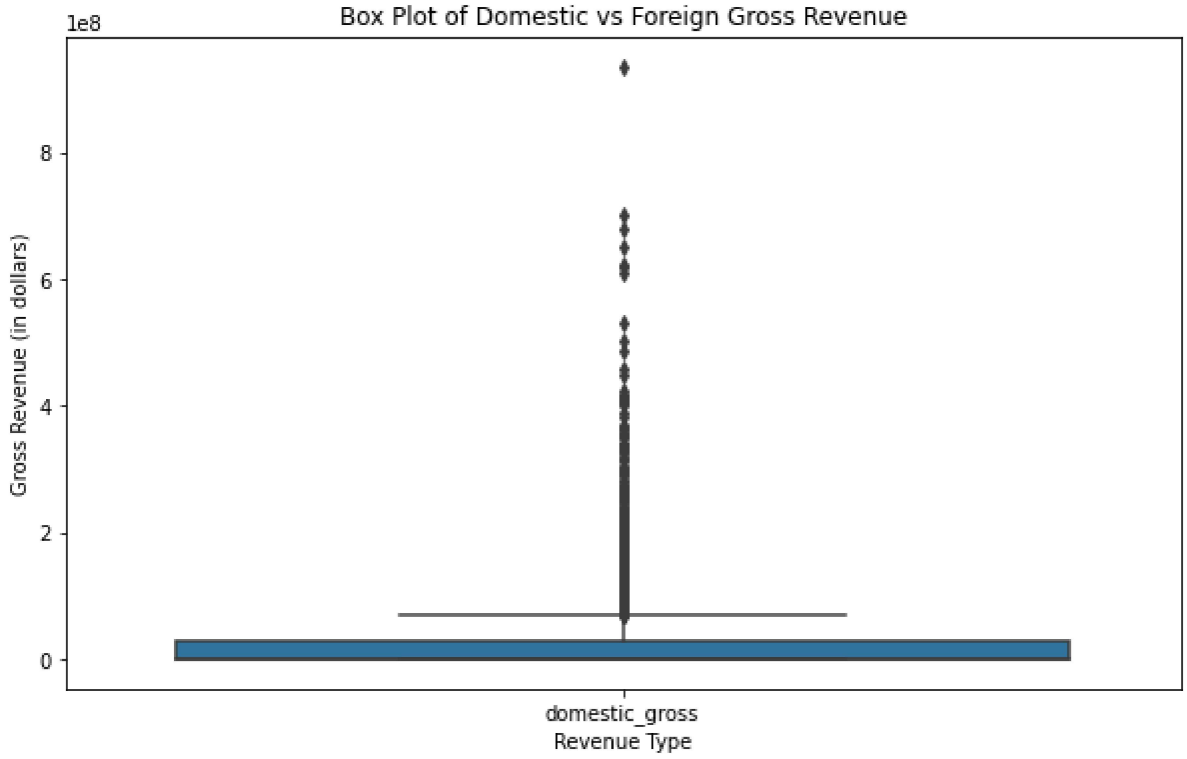
Visualization - Bar Plot of Average Rating by Genre

In [27]:
```
# Visualization - Bar Plot of Average Rating by Genre
plt.figure(figsize=(10, 6))
sns.barplot(x=average_ratings_by_genre['genre_ids'], y=average_ratings_by_genre
plt.title('Average Rating by Genre')
plt.xlabel('Genre ID')
plt.ylabel('Average Rating')
plt.savefig('Images/bar_plot_average_rating_by_genre.png')
plt.show()
```



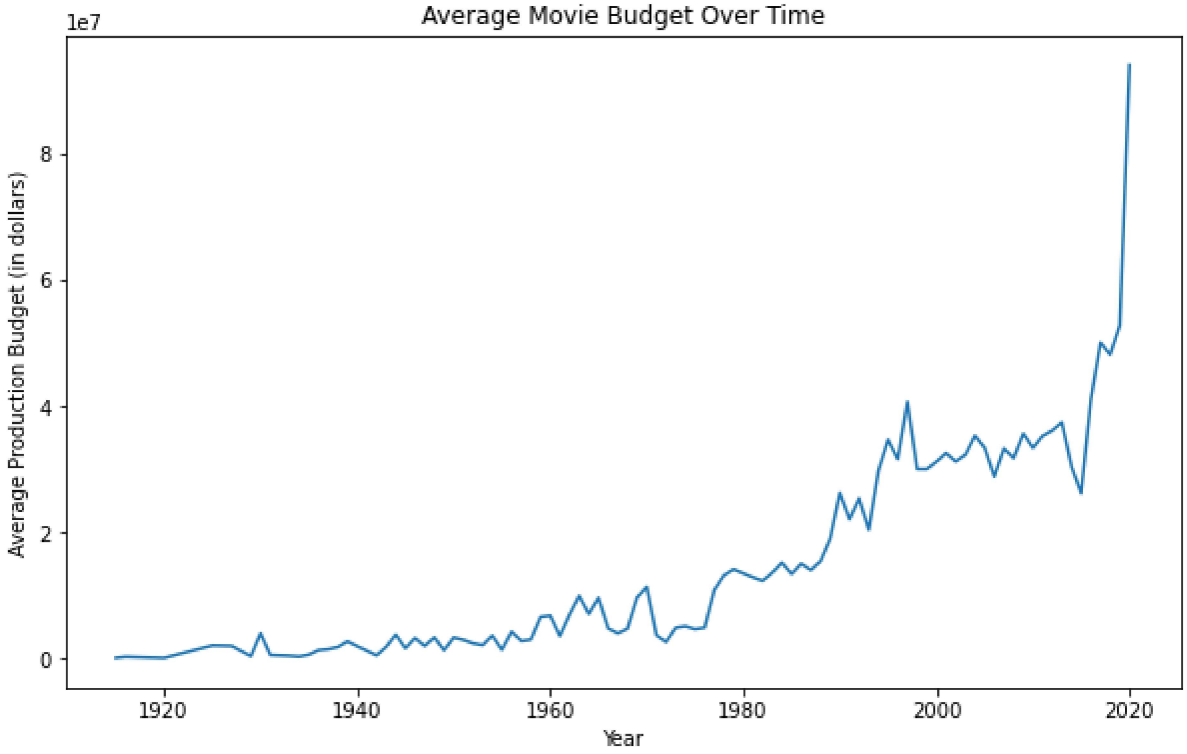
Visualization - Box Plot of Domestic vs Foreign Gross Revenue

In [28]:
```
# Visualization - Box Plot of Domestic vs Foreign Gross Revenue
plt.figure(figsize=(10, 6))
sns.boxplot(data=bom_movie_gross[['domestic_gross', 'foreign_gross']])
plt.title('Box Plot of Domestic vs Foreign Gross Revenue')
plt.xlabel('Revenue Type')
plt.ylabel('Gross Revenue (in dollars)')
plt.savefig('Images/box_plot_domestic_foreign_gross.png')
plt.show()
```



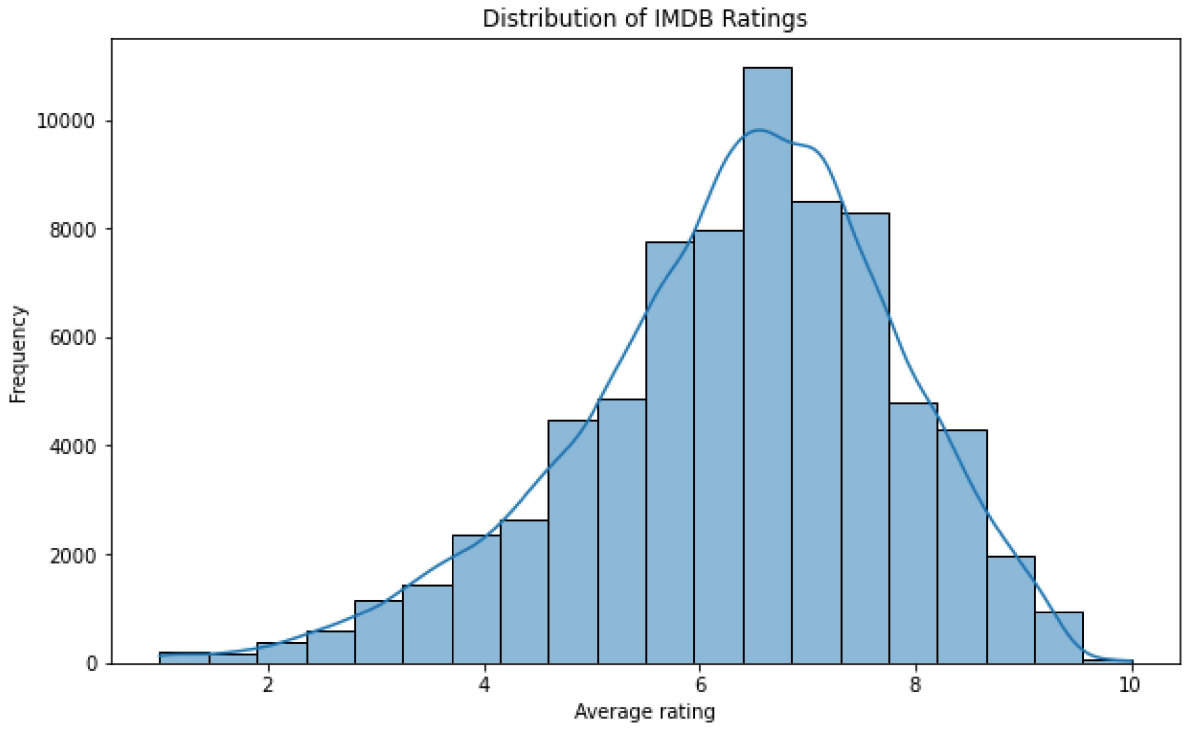
Visualization- Line Plot of Average Movie Budget Over Time

In [29]:
```
# Converting release_date to datetime
tn_movie_budgets['release_date'] = pd.to_datetime(tn_movie_budgets['release_dat

# Calculating average production budget by year
average_budget_by_year = tn_movie_budgets.groupby(tn_movie_budgets['release_dat

# average budget over time
plt.figure(figsize=(10, 6))
sns.lineplot(data=average_budget_by_year, x='release_date', y='production_budge
plt.title('Average Movie Budget Over Time')
plt.xlabel('Year')
plt.ylabel('Average Production Budget (in dollars)')
plt.savefig('Images/average_movie_budget_over_time.png')
plt.show()
```



Visualization - Distribution of IMDB Ratings

In [30]:
```
average_rating_column = 'averagerating'

# Visualization - Distribution of IMDB Ratings
plt.figure(figsize=(10, 6))
sns.histplot(imdb_data[average_rating_column], bins=20, kde=True)
plt.title('Distribution of IMDB Ratings')
plt.xlabel('Average rating')
plt.ylabel('Frequency')
plt.savefig('Images/distribution_of_imdb_ratings.png')
plt.show()
```



Business Recommendations

In [21]:

```
recommendations = """

High-Budget Blockbusters: Action and Adventure
Our analysis reveals that movies with substantial production budgets tend to ac

Captivating Family Audiences
Genre analysis shows that family-oriented genres such as Animation, Family, and

Capitalize on Franchises and Sequels
Movies that are part of well-known franchises or are sequels tend to perform ex

Global Marketing Strategy
Certain movies achieve significant success internationally, even if their domes

Strategic Budget Planning
Monitoring trends in production budgets over time can provide valuable insights
"""

print(recommendations)
```

```
High-Budget Blockbusters: Action and Adventure
Our analysis reveals that movies with substantial production budgets tend to
achieve higher global box office revenues. In particular, the Action and Adve
nture genres are consistently among the top performers in terms of gross earn
ings.

Captivating Family Audiences
Genre analysis shows that family-oriented genres such as Animation, Family, a
nd Adventure are perennially popular. Investing in these genres can attract a
wide audience base, especially families, thereby enhancing box office succes
s.

Capitalize on Franchises and Sequels
Movies that are part of well-known franchises or are sequels tend to perform
exceptionally well at the box office. Strategic investments in creating or ac
quiring successful franchises can ensure a reliable revenue stream.

Global Marketing Strategy
Certain movies achieve significant success internationally, even if their dom
estic performance is only moderate. Developing marketing strategies that effe
ctively target both domestic and international markets is essential for maxim
izing revenue.

Strategic Budget Planning
Monitoring trends in production budgets over time can provide valuable insigh
ts for planning future projects. Regularly reviewing and optimizing budget al
locations can lead to more efficient use of resources and better financial ou
tcomes for new projects.
```