Analyzing Global Movie Trends with Big Data: Genre, Ratings, and Regional Availability

Team Name: The V-Engineers

Team Members:

Bhuvaneswari Devi Ghanta
Harini Beeram
Manush Kumar Kachakayala

Section: 02
Project ID: 44517

Contents

1 Project Overview

1.1 Project Idea

This project analyzes a large movie dataset to understand patterns in release trends, genre popularity, rating distribution, and geographic availability. Using PySpark for big data processing and Matplotlib for visual representation, the project aims to provide insights into trends across different genres, release years, and audience ratings.

2 Technology Summary

The project leverages the following technologies:

- **PySpark:** To handle the large dataset and perform distributed data processing efficiently.
- **Matplotlib:** For visualizing findings such as trends in release years, genres, and ratings.
- **Python:** For data manipulation, data cleaning, and integration of PySpark and Matplotlib.

3 Architecture Diagram

The architecture diagram for this project outlines the data flow through various stages. Here is a breakdown of the stages:

- **Data Ingestion:** Load the dataset into PySpark for high-performance data processing.
- **Data Cleaning:** Handle missing values and standardize data formats (e.g., convert release years to integers).
- **Data Analysis:** Analyze data to identify trends and patterns.
  - **Trend Analysis:** Analyze release year trends and genre popularity.
  - **Rating Analysis:** Investigate IMDb rating distributions across genres.
  - **Regional Availability:** Study movie distribution across available countries.
- **Visualization:** Use Matplotlib to create charts for trends, genre distribution, and rating patterns.

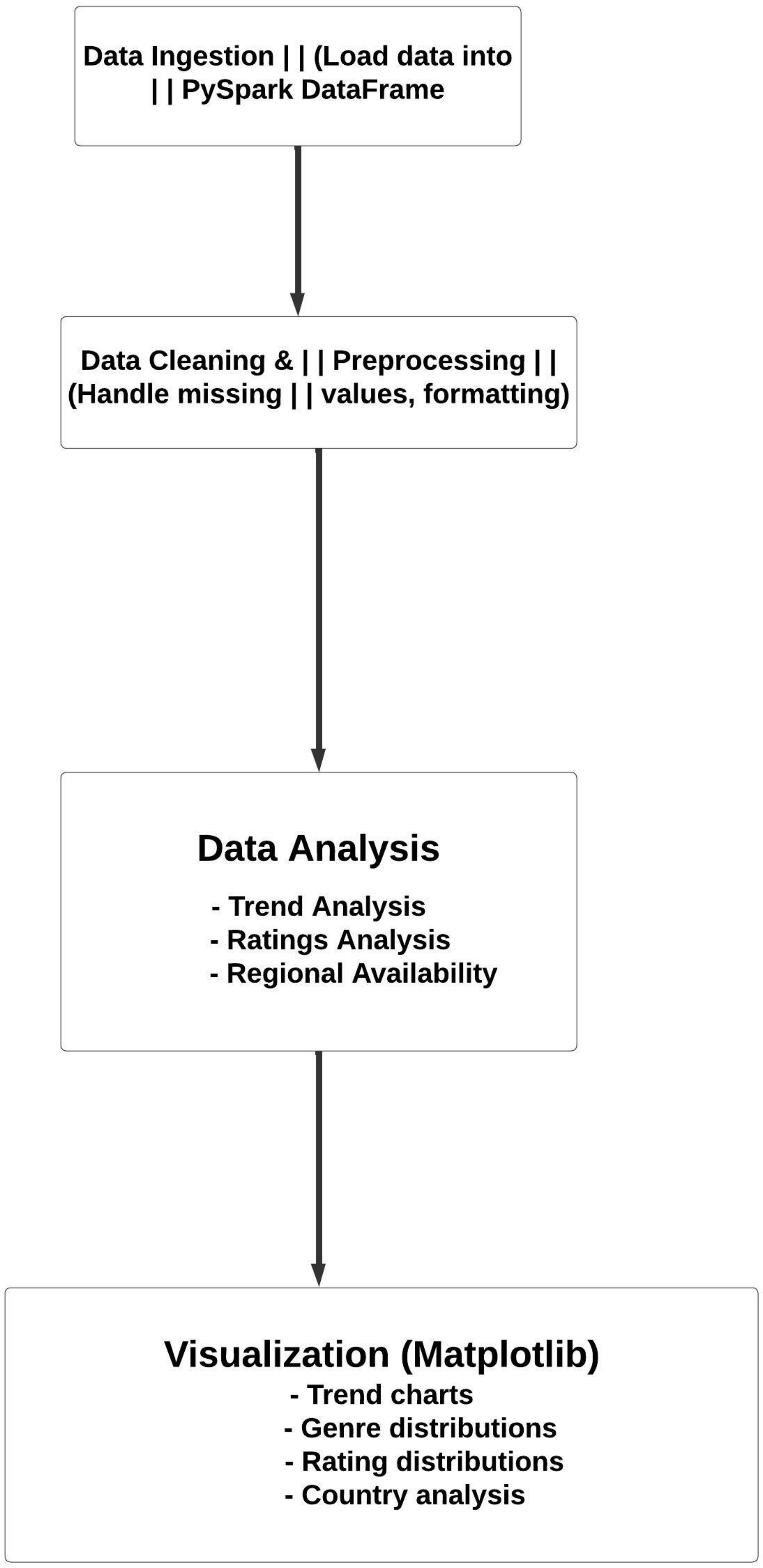

Data Ingestion | | (Load data into | | PySpark DataFrame
Data Cleaning & | | Preprocessing | | (Handle missing | | values, formatting)
Data Analysis
- Trend Analysis
- Ratings Analysis
- Regional Availability
Visualization (Matplotlib)
- Trend charts
- Genre distributions
- Rating distributions
- Country analysis

Figure 1: Architecture Diagram

4 Architecture Summary

4.1 Data Ingestion

The data ingestion phase involves loading the dataset into PySpark to leverage its distributed processing capabilities. This stage ensures efficient data processing and handling of large datasets.

4.2 Data Cleaning

During this phase, missing values are addressed, and data formats are standardized for consistency. For example, release years may be converted to integers, and genres may be formatted uniformly.

4.3 Data Analysis

The data analysis stage includes three major areas:

1. **Trend Analysis:** Identify release year trends and the popularity of genres over time.
2. **Rating Analysis:** Calculate average ratings for each genre and examine IMDb rating distributions.
3. **Regional Availability:** Analyze the distribution of movies across different countries.

4.4 Visualization

Visual representations will be generated using Matplotlib to illustrate trends, genre distribution, and rating patterns, providing insights into global movie trends.

article graphicx float

5 Project Goals

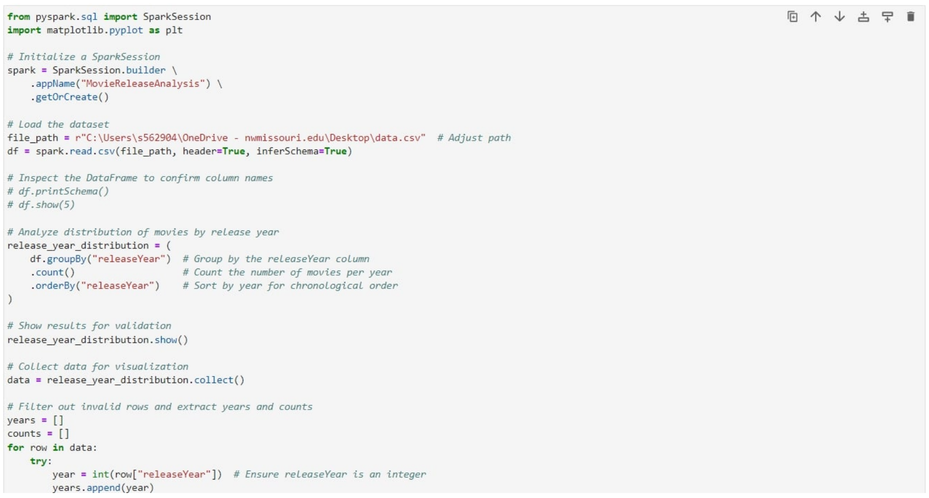
The project has the following goals:

1. **Analyze Movie Release Trends by Year:** Examine how the number of movies released has varied over the years.
2. **Determine Genre Popularity:** Identify and visualize the most common movie genres.
3. **Explore Ratings by Genre:** Calculate the average IMDb rating for each genre to highlight differences in audience preferences.
4. **Investigate Country Availability Patterns:** Analyze the distribution of movies by the countries where they are available.
5. **Study IMDb Rating Distribution:** Visualize the distribution of IMDb ratings across all movies.
6. **Analyze High-Engagement Movies:** Identify movies with high IMDb vote counts and explore their characteristics, such as genre and release year.

6 Result Summary

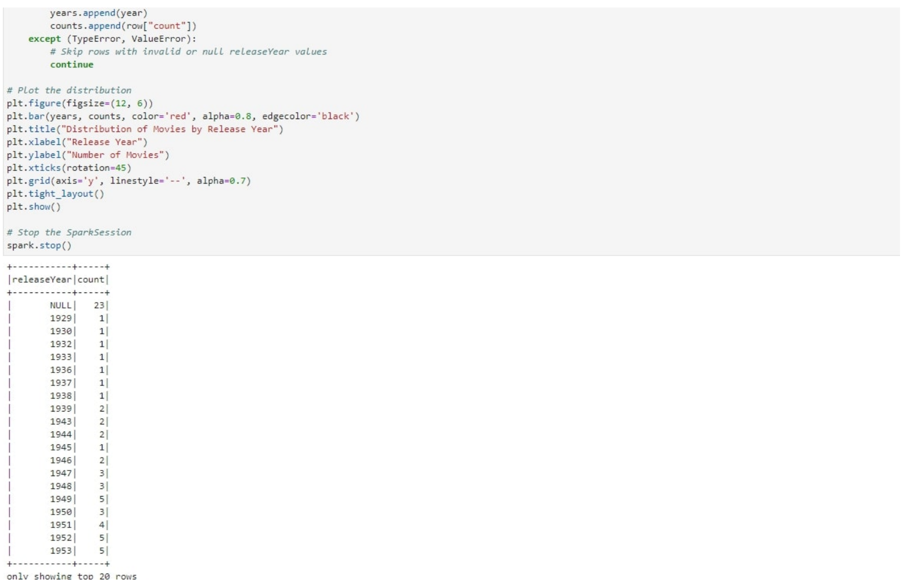
In this section, we summarize the key findings of the project:

- **Analyze Movie Release Trends by Year:** The analysis of movie release years revealed significant trends in the production volume, with a clear spike in the early 2000s. This suggests increased interest in movie production globally, possibly driven by digital distribution.



```python
from pyspark.sql import SparkSession
import matplotlib.pyplot as plt

# Initialize a SparkSession
spark = SparkSession.builder \
    .appName("MovieReleaseAnalysis") \
    .getOrCreate()

# Load the dataset
file_path = r"C:\Users\s562904\OneDrive - nwmissouri.edu\Desktop\data.csv"  # Adjust path
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Inspect the DataFrame to confirm column names
# df.printSchema()
# df.show(5)

# Analyze distribution of movies by release year
release_year_distribution = (
    df.groupBy("releaseYear")  # Group by the releaseYear column
    .count()                   # Count the number of movies per year
    .orderBy("releaseYear")    # Sort by year for chronological order
)

# Show results for validation
release_year_distribution.show()

# Collect data for visualization
data = release_year_distribution.collect()

# Filter out invalid rows and extract years and counts
years = []
counts = []
for row in data:
    try:
        year = int(row["releaseYear"])  # Ensure releaseYear is an integer
        years.append(year)
```

Figure 2: Movie Release Trends by Year



```
        years.append(year)
        counts.append(row["count"])
    except (TypeError, ValueError):
        # Skip rows with invalid or null releaseYear values
        continue

# Plot the distribution
plt.figure(figsize=(12, 6))
plt.bar(years, counts, color='red', alpha=0.8, edgecolor='black')
plt.title("Distribution of Movies by Release Year")
plt.xlabel("Release Year")
plt.ylabel("Number of Movies")
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Stop the SparkSession
spark.stop()
```

```
+-----------+-----+
|releaseYear|count|
+-----------+-----+
|       NULL|   23|
|       1929|    1|
|       1930|    1|
|       1932|    1|
|       1933|    1|
|       1936|    1|
|       1937|    1|
|       1938|    1|
|       1939|    2|
|       1943|    2|
|       1944|    2|
|       1945|    1|
|       1946|    2|
|       1947|    3|
|       1948|    3|
|       1949|    5|
|       1950|    3|
|       1951|    4|
|       1952|    5|
|       1953|    5|
+-----------+-----+
only showing top 20 rows
```

Figure 3: Movie Release Trends by Year

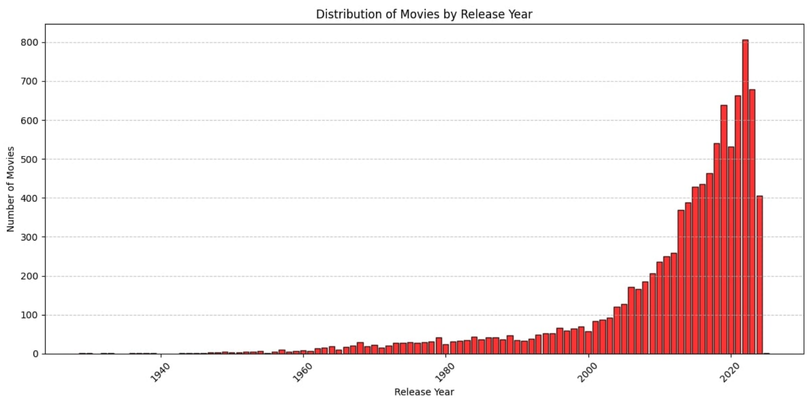
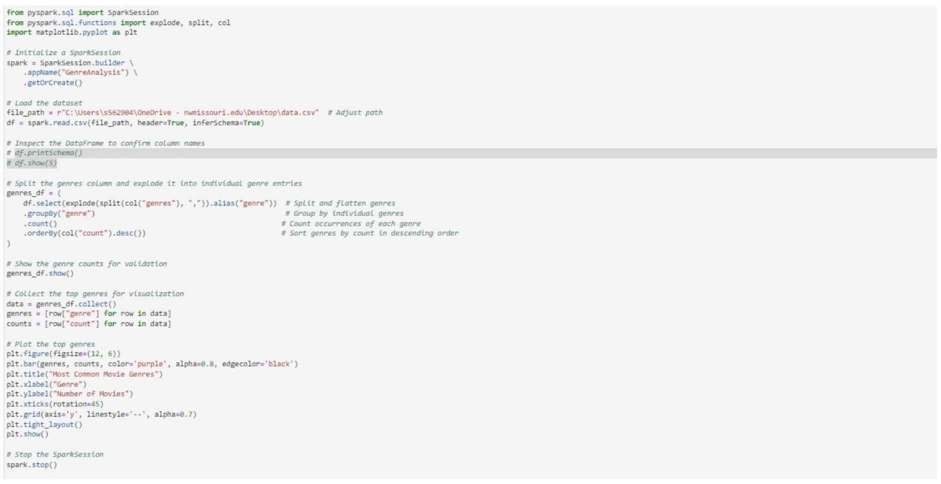


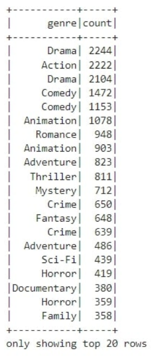
Figure 4: Movie Release Trends by Year

- **Determine Genre Popularity:** The most common genres were found to be Drama, Comedy, and Action, with significant variations over time. The distribution of genres has evolved, with certain genres (e.g., Sci-Fi) gaining popularity in more recent years.



```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, split, col
import matplotlib.pyplot as plt

# Initialize a SparkSession
spark = SparkSession.builder \
    .appName("GenreAnalysis") \
    .getOrCreate()

# Load the dataset
file_path = r"C:\Users\s562904\OneDrive - nwmissouri.edu\Desktop\data.csv"  # Adjust path
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Inspect the DataFrame to confirm column names
# df.printSchema()
# df.show(5)

# Split the genres column and explode it into individual genre entries
genres_df = (
    df.select(explode(split(col("genres"), ",")).alias("genre"))  # Split and flatten genres
    .groupBy("genre")                                              # Group by individual genres
    .count()                                                      # Count occurrences of each genre
    .orderBy(col("count").desc())                                 # Sort genres by count in descending order
)

# Show the genre counts for validation
genres_df.show()

# Collect the top genres for visualization
data = genres_df.collect()
genres = [row["genre"] for row in data]
counts = [row["count"] for row in data]

# Plot the top genres
plt.figure(figsize=(12, 6))
plt.bar(genres, counts, color='purple', alpha=0.8, edgecolor='black')
plt.title("Most Common Movie Genres")
plt.xlabel("Genre")
plt.ylabel("Number of Movies")
plt.xticks(rotation=45)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Stop the SparkSession
spark.stop()
```

Figure 5: Genre Popularity Trends



```
+-----------+-----+
|      genre|count|
+-----------+-----+
|      Drama| 2244|
|     Action| 2222|
|      Drama| 2104|
|     Comedy| 1472|
|     Comedy| 1153|
|  Animation| 1078|
|    Romance|  948|
|  Animation|  903|
|  Adventure|  823|
|   Thriller|  811|
|    Mystery|  712|
|      Crime|  650|
|    Fantasy|  648|
|      Crime|  639|
|  Adventure|  486|
|     Sci-Fi|  439|
|     Horror|  419|
|Documentary|  380|
|     Horror|  359|
|     Family|  358|
+-----------+-----+
only showing top 20 rows
```

Figure 6: Genre Popularity Trends

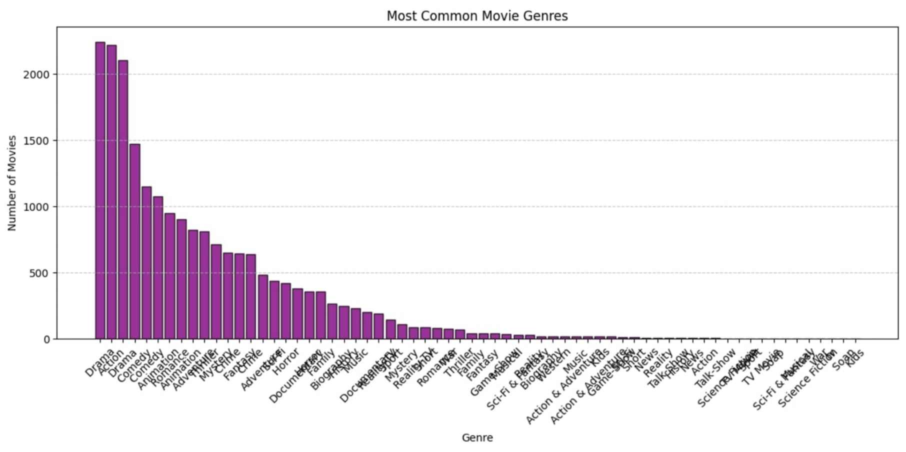
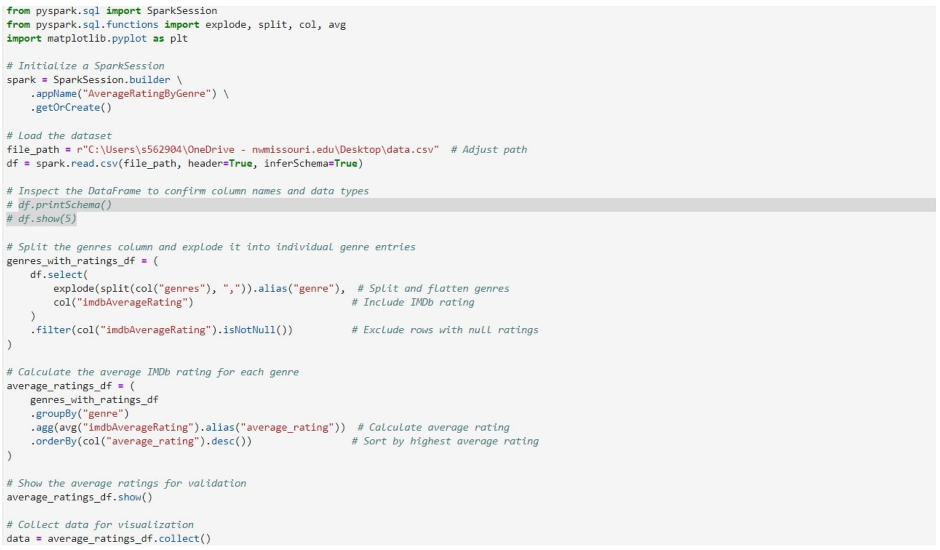


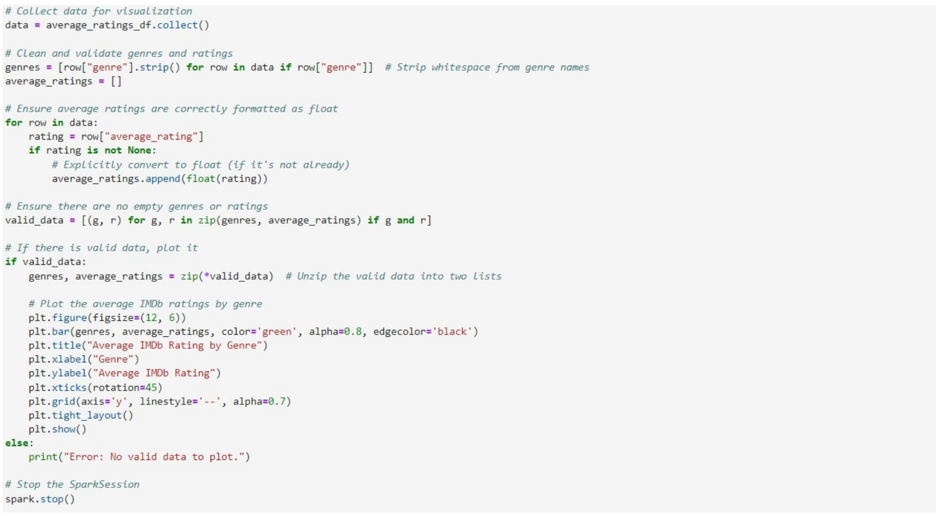
Figure 7: Genre Popularity Trends

- **Explore Ratings by Genre:** IMDb ratings tend to be normally distributed, with most movies clustering around a rating of 6 to 8. However, a few outliers with extremely high or low ratings were also observed.



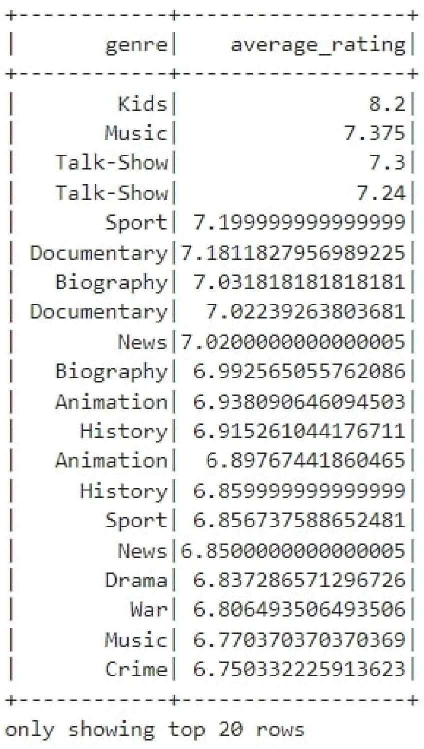
```
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, split, col, avg
import matplotlib.pyplot as plt

# Initialize a SparkSession
spark = SparkSession.builder \
    .appName("AverageRatingByGenre") \
    .getOrCreate()

# Load the dataset
file_path = r"C:\Users\s562904\OneDrive - nwmissouri.edu\Desktop\data.csv"  # Adjust path
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Inspect the DataFrame to confirm column names and data types
# df.printSchema()
# df.show(5)

# Split the genres column and explode it into individual genre entries
genres_with_ratings_df = (
    df.select(
        explode(split(col("genres"), ",")).alias("genre"),  # Split and flatten genres
        col("imdbAverageRating")                            # Include IMDb rating
    )
    .filter(col("imdbAverageRating").isNotNull())          # Exclude rows with null ratings
)

# Calculate the average IMDb rating for each genre
average_ratings_df = (
    genres_with_ratings_df
    .groupBy("genre")
    .agg(avg("imdbAverageRating").alias("average_rating"))  # Calculate average rating
    .orderBy(col("average_rating").desc())                  # Sort by highest average rating
)

# Show the average ratings for validation
average_ratings_df.show()

# Collect data for visualization
data = average_ratings_df.collect()
```

Figure 8: Explore Ratings by Genre



```python
# Collect data for visualization
data = average_ratings_df.collect()

# Clean and validate genres and ratings
genres = [row["genre"].strip() for row in data if row["genre"]]  # Strip whitespace from genre names
average_ratings = []

# Ensure average ratings are correctly formatted as float
for row in data:
    rating = row["average_rating"]
    if rating is not None:
        # Explicitly convert to float (if it's not already)
        average_ratings.append(float(rating))

# Ensure there are no empty genres or ratings
valid_data = [(g, r) for g, r in zip(genres, average_ratings) if g and r]

# If there is valid data, plot it
if valid_data:
    genres, average_ratings = zip(*valid_data)  # Unzip the valid data into two lists

    # Plot the average IMDb ratings by genre
    plt.figure(figsize=(12, 6))
    plt.bar(genres, average_ratings, color='green', alpha=0.8, edgecolor='black')
    plt.title("Average IMDb Rating by Genre")
    plt.xlabel("Genre")
    plt.ylabel("Average IMDb Rating")
    plt.xticks(rotation=45)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()
else:
    print("Error: No valid data to plot.")

# Stop the SparkSession
spark.stop()
```

Figure 9: Explore Ratings by Genre



```
+------------+------------------+
|       genre|    average_rating|
+------------+------------------+
|        Kids|               8.2|
|       Music|             7.375|
|   Talk-Show|               7.3|
|   Talk-Show|              7.24|
|       Sport| 7.199999999999999|
| Documentary|7.1811827956989225|
|   Biography| 7.031818181818181|
| Documentary|  7.02239263803681|
|        News|7.0200000000000005|
|   Biography| 6.992565055762086|
|   Animation| 6.938090646094503|
|     History| 6.915261044176711|
|   Animation|  6.89767441860465|
|     History| 6.859999999999999|
|       Sport| 6.856737588652481|
|        News|6.8500000000000005|
|       Drama| 6.837286571296726|
|         War| 6.806493506493506|
|       Music| 6.770370370370369|
|       Crime| 6.750332225913623|
+------------+------------------+
only showing top 20 rows
```

Figure 10: Explore Ratings by Genre

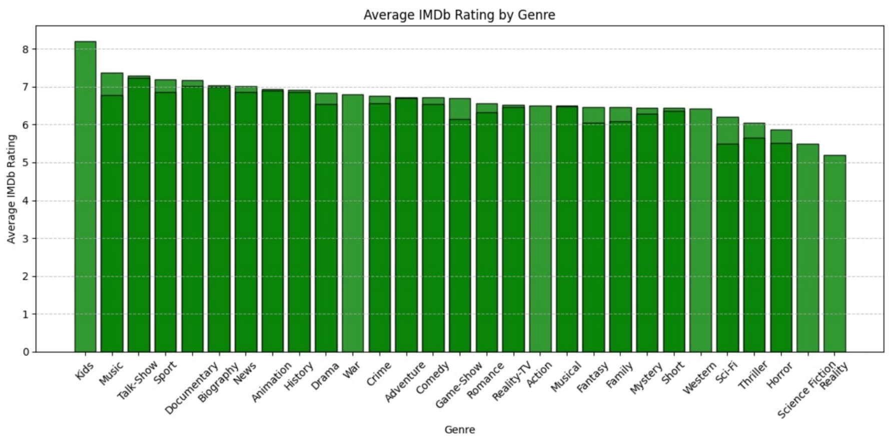
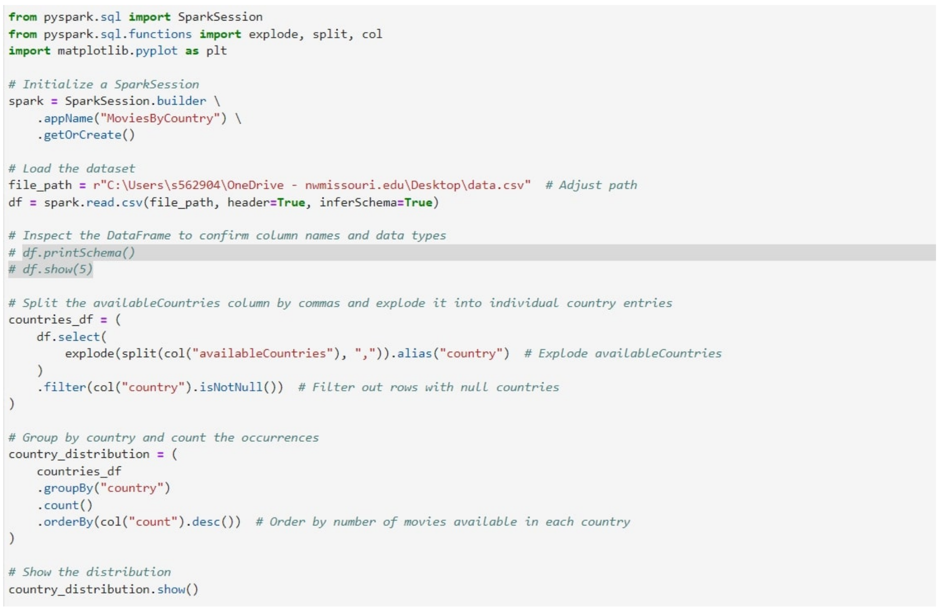


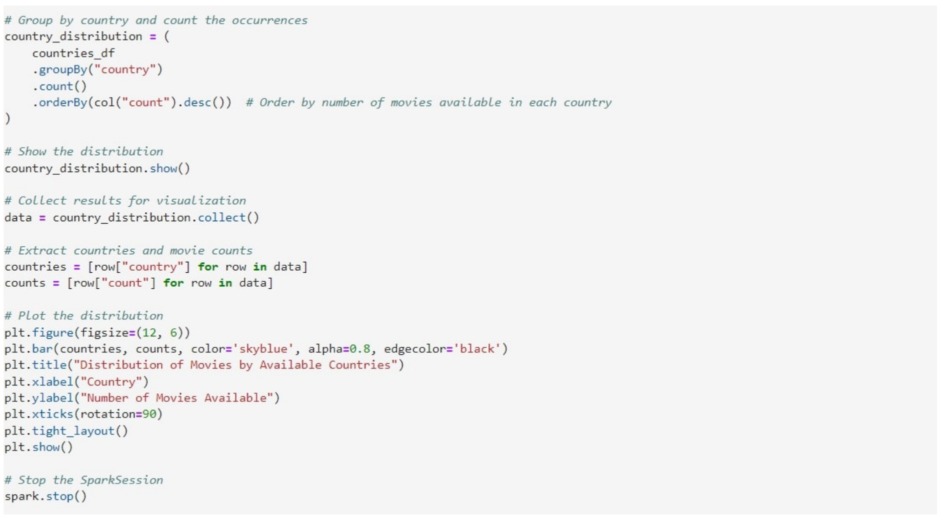
Figure 11: Explore Ratings by Genre

- **Investigate Country Availability Patterns:** A clear pattern emerged showing that movies are most commonly available in the United States, followed by European countries, indicating regional disparities in movie distribution.



```
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, split, col
import matplotlib.pyplot as plt

# Initialize a SparkSession
spark = SparkSession.builder \
    .appName("MoviesByCountry") \
    .getOrCreate()

# Load the dataset
file_path = r"C:\Users\s562904\OneDrive - nwmissouri.edu\Desktop\data.csv"  # Adjust path
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Inspect the DataFrame to confirm column names and data types
# df.printSchema()
# df.show(5)

# Split the availableCountries column by commas and explode it into individual country entries
countries_df = (
    df.select(
        explode(split(col("availableCountries"), ",")).alias("country")  # Explode availableCountries
    )
    .filter(col("country").isNotNull())  # Filter out rows with null countries
)

# Group by country and count the occurrences
country_distribution = (
    countries_df
    .groupBy("country")
    .count()
    .orderBy(col("count").desc())  # Order by number of movies available in each country
)

# Show the distribution
country_distribution.show()
```

Figure 12: Investigate Country Availability Patterns



```python
# Group by country and count the occurrences
country_distribution = (
    countries_df
    .groupBy("country")
    .count()
    .orderBy(col("count").desc())  # Order by number of movies available in each country
)

# Show the distribution
country_distribution.show()

# Collect results for visualization
data = country_distribution.collect()

# Extract countries and movie counts
countries = [row["country"] for row in data]
counts = [row["count"] for row in data]

# Plot the distribution
plt.figure(figsize=(12, 6))
plt.bar(countries, counts, color='skyblue', alpha=0.8, edgecolor='black')
plt.title("Distribution of Movies by Available Countries")
plt.xlabel("Country")
plt.ylabel("Number of Movies Available")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

# Stop the SparkSession
spark.stop()
```

Figure 13: Investigate Country Availability Patterns

```
+-------+-----+
|country|count|
+-------+-----+
|     JP| 7178|
|     US| 2634|
|     US|  255|
|    221|    1|
+-------+-----+
```

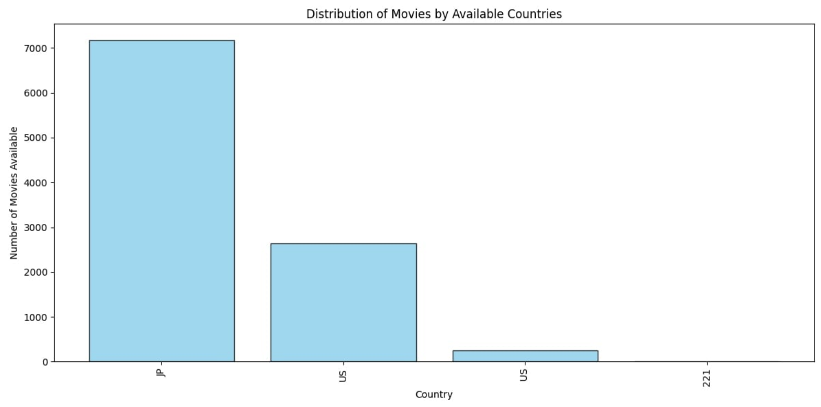
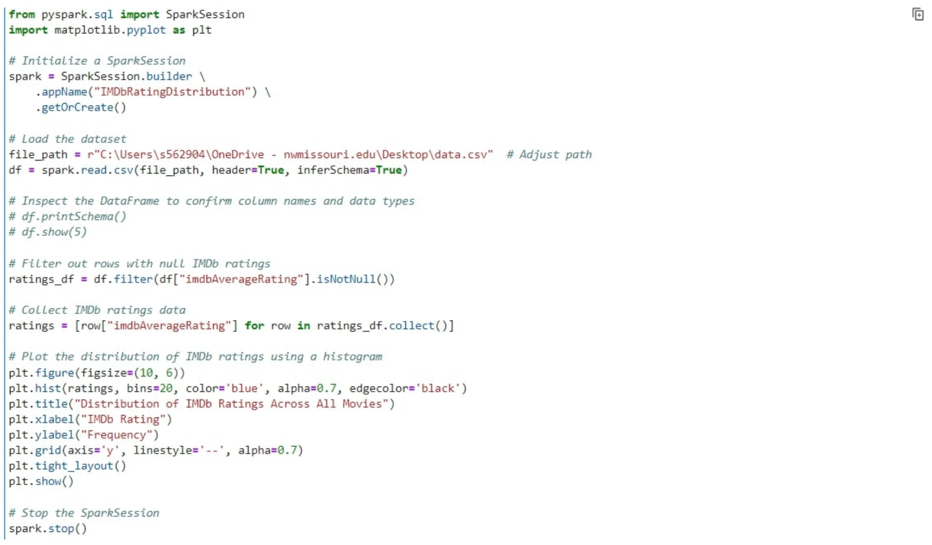


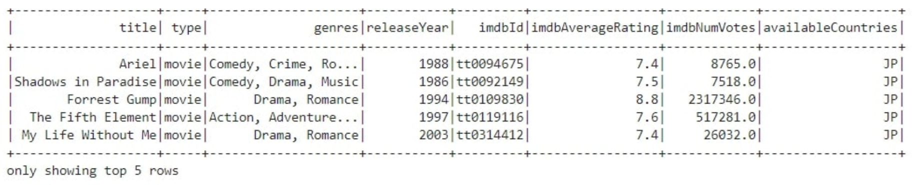
Figure 14: Investigate Country Availability Patterns

- **Study IMDb Rating Distribution:**IMDb ratings exhibited a normal distribution, with most movies clustering around ratings of 6 to 8. A few outliers with extremely high or low ratings were also identified, representing either critically acclaimed masterpieces or widely criticized productions.



```python
from pyspark.sql import SparkSession
import matplotlib.pyplot as plt

# Initialize a SparkSession
spark = SparkSession.builder \
    .appName("IMDbRatingDistribution") \
    .getOrCreate()

# Load the dataset
file_path = r"C:\Users\s562904\OneDrive - nwmissouri.edu\Desktop\data.csv"  # Adjust path
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Inspect the DataFrame to confirm column names and data types
# df.printSchema()
# df.show(5)

# Filter out rows with null IMDb ratings
ratings_df = df.filter(df["imdbAverageRating"].isNotNull())

# Collect IMDb ratings data
ratings = [row["imdbAverageRating"] for row in ratings_df.collect()]

# Plot the distribution of IMDb ratings using a histogram
plt.figure(figsize=(10, 6))
plt.hist(ratings, bins=20, color='blue', alpha=0.7, edgecolor='black')
plt.title("Distribution of IMDb Ratings Across All Movies")
plt.xlabel("IMDb Rating")
plt.ylabel("Frequency")
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Stop the SparkSession
spark.stop()
```

Figure 15: High IMDb Vote Movies by Genre



```
+------------------+-----+-------------------+-----------+---------+-----------------+-------------+------------------+
|             title| type|             genres|releaseYear|   imdbId|imdbAverageRating|imdbNumVotes|availableCountries|
+------------------+-----+-------------------+-----------+---------+-----------------+-------------+------------------+
|             Ariel|movie|Comedy, Crime, Ro...|       1988|tt0094675|              7.4|       8765.0|                JP|
|Shadows in Paradise|movie|Comedy, Drama, Music|       1986|tt0092149|              7.5|       7518.0|                JP|
|      Forrest Gump|movie|     Drama, Romance|       1994|tt0109830|              8.8|    2317346.0|                JP|
|  The Fifth Element|movie|Action, Adventure...|       1997|tt0119116|              7.6|     517281.0|                JP|
| My Life Without Me|movie|     Drama, Romance|       2003|tt0314412|              7.4|      26032.0|                JP|
+------------------+-----+-------------------+-----------+---------+-----------------+-------------+------------------+
only showing top 5 rows
```

Figure 16: Characteristics of High-Vote Movies

```
Number of IMDb Ratings Collected: 8644
```

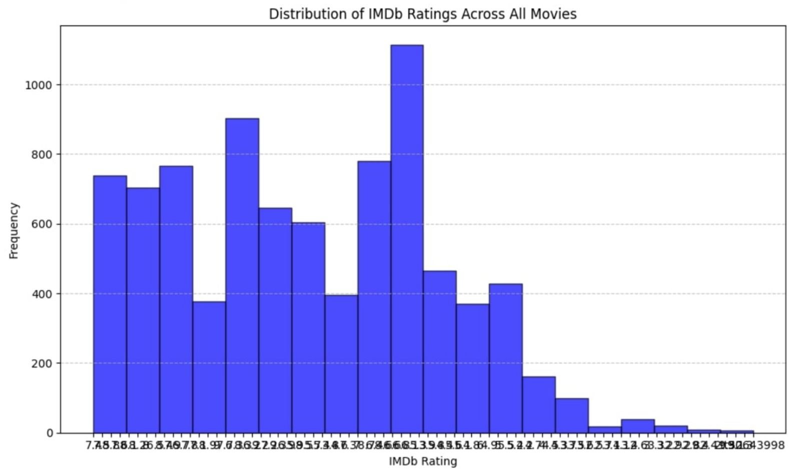
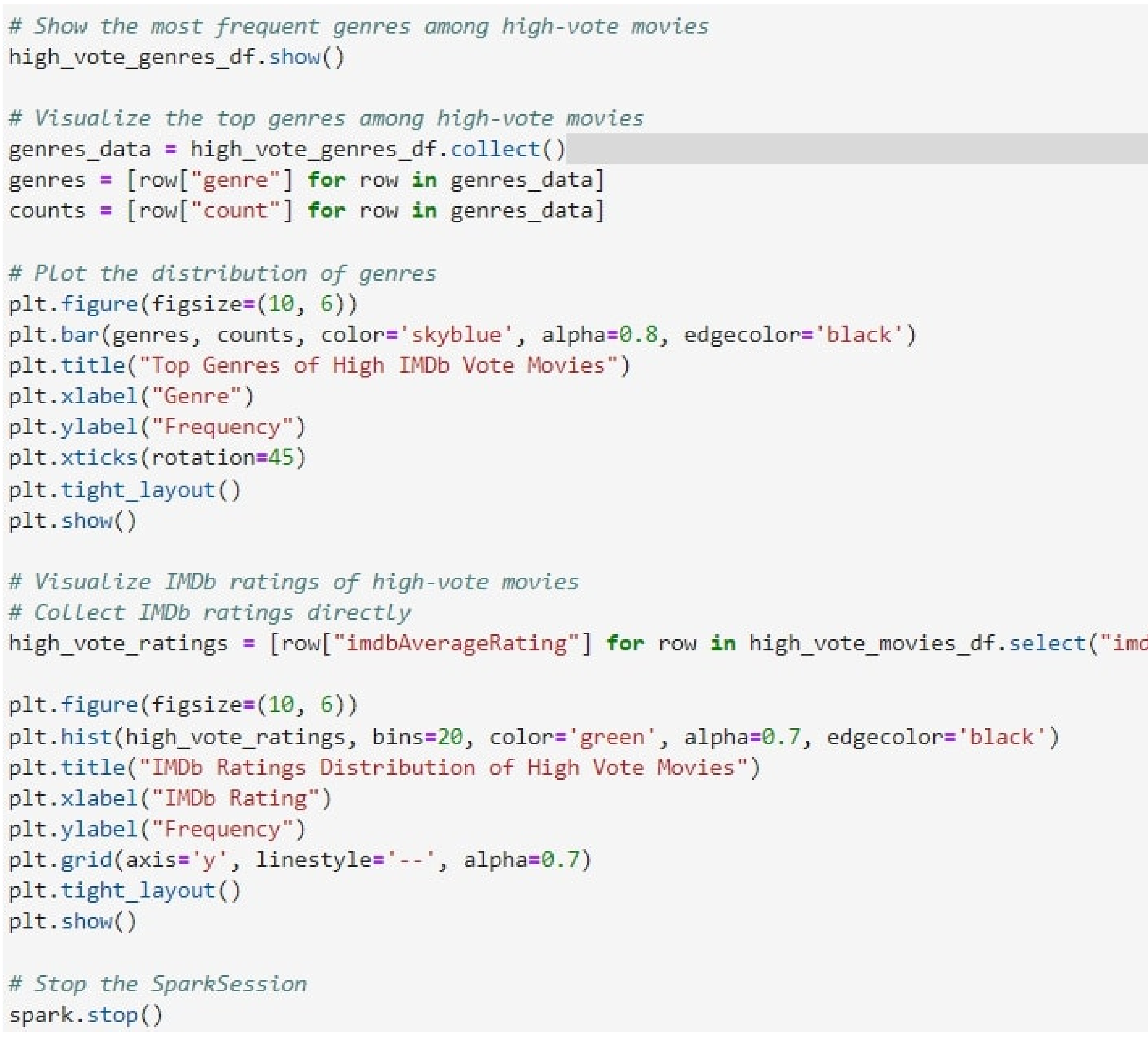


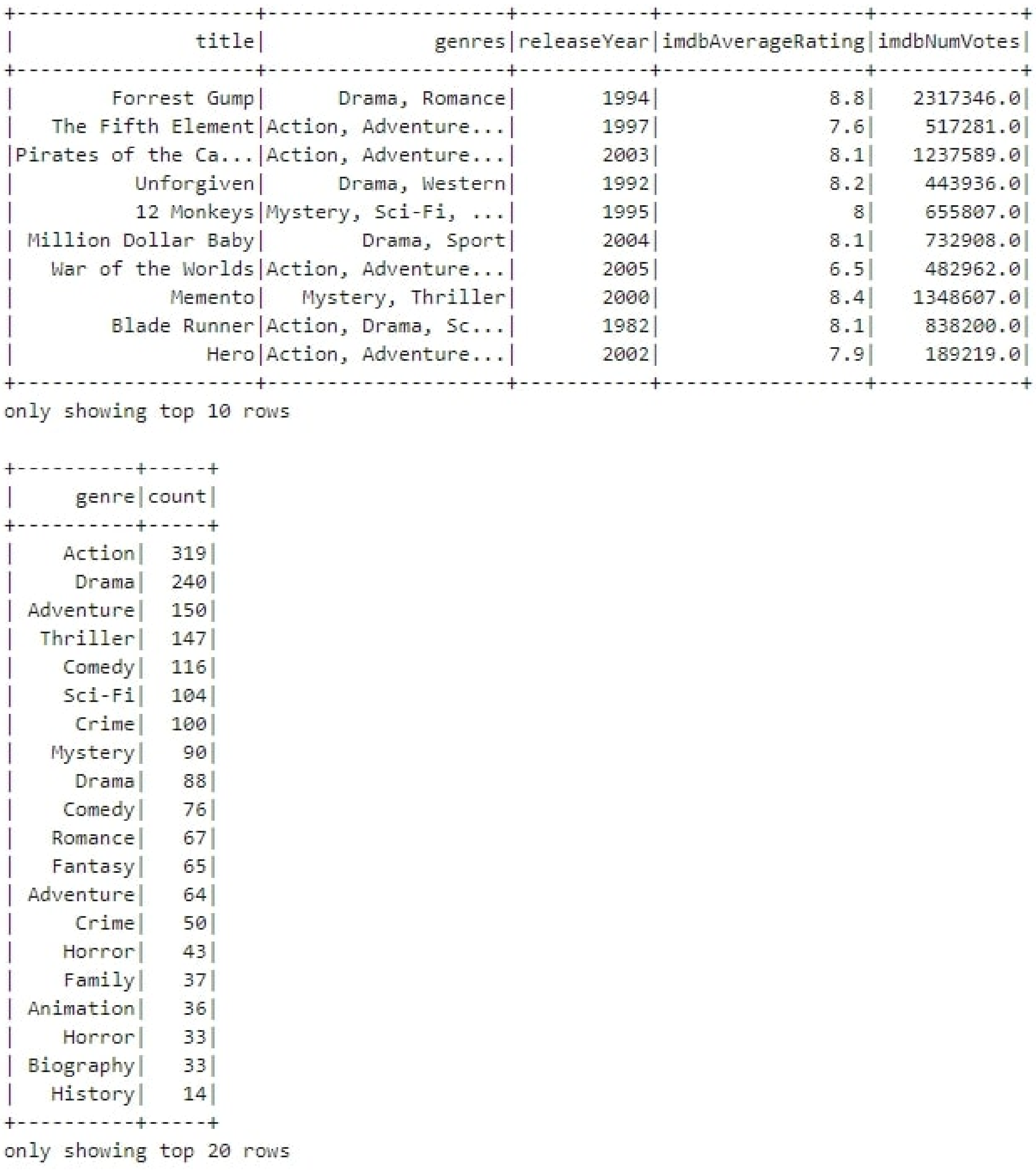
Figure 17: High-Vote Movies Over Time

- **Analyze High-Engagement Movies:**Movies with high IMDb vote counts were typically major releases in genres such as Action, Adventure, and Drama. These films often had wide distribution, significant marketing campaigns, and strong global impact, making them stand out as audience favorites



```
# Show the most frequent genres among high-vote movies
high_vote_genres_df.show()

# Visualize the top genres among high-vote movies
genres_data = high_vote_genres_df.collect()
genres = [row["genre"] for row in genres_data]
counts = [row["count"] for row in genres_data]

# Plot the distribution of genres
plt.figure(figsize=(10, 6))
plt.bar(genres, counts, color='skyblue', alpha=0.8, edgecolor='black')
plt.title("Top Genres of High IMDb Vote Movies")
plt.xlabel("Genre")
plt.ylabel("Frequency")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Visualize IMDb ratings of high-vote movies
# Collect IMDb ratings directly
high_vote_ratings = [row["imdbAverageRating"] for row in high_vote_movies_df.select("imd

plt.figure(figsize=(10, 6))
plt.hist(high_vote_ratings, bins=20, color='green', alpha=0.7, edgecolor='black')
plt.title("IMDb Ratings Distribution of High Vote Movies")
plt.xlabel("IMDb Rating")
plt.ylabel("Frequency")
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

# Stop the SparkSession
spark.stop()
```

Figure 18: Analyze High-Engagement Movies



```
+--------------------+-------------------+-----------+----------------+------------+
|               title|             genres|releaseYear|imdbAverageRating|imdbNumVotes|
+--------------------+-------------------+-----------+----------------+------------+
|        Forrest Gump|     Drama, Romance|       1994|             8.8|   2317346.0|
|   The Fifth Element|Action, Adventure...|       1997|             7.6|    517281.0|
|Pirates of the Ca...|Action, Adventure...|       2003|             8.1|   1237589.0|
|          Unforgiven|     Drama, Western|       1992|             8.2|    443936.0|
|          12 Monkeys|Mystery, Sci-Fi, ...|       1995|               8|    655807.0|
|  Million Dollar Baby|       Drama, Sport|       2004|             8.1|    732908.0|
|   War of the Worlds|Action, Adventure...|       2005|             6.5|    482962.0|
|             Memento|  Mystery, Thriller|       2000|             8.4|   1348607.0|
|        Blade Runner|Action, Drama, Sc...|       1982|             8.1|    838200.0|
|                Hero|Action, Adventure...|       2002|             7.9|    189219.0|
+--------------------+-------------------+-----------+----------------+------------+
only showing top 10 rows

+----------+-----+
|     genre|count|
+----------+-----+
|    Action|  319|
|     Drama|  240|
| Adventure|  150|
|  Thriller|  147|
|    Comedy|  116|
|    Sci-Fi|  104|
|     Crime|  100|
|   Mystery|   90|
|     Drama|   88|
|    Comedy|   76|
|   Romance|   67|
|   Fantasy|   65|
| Adventure|   64|
|     Crime|   50|
|    Horror|   43|
|    Family|   37|
| Animation|   36|
|    Horror|   33|
| Biography|   33|
|   History|   14|
+----------+-----+
only showing top 20 rows
```

Figure 19: Analyze High-Engagement Movies

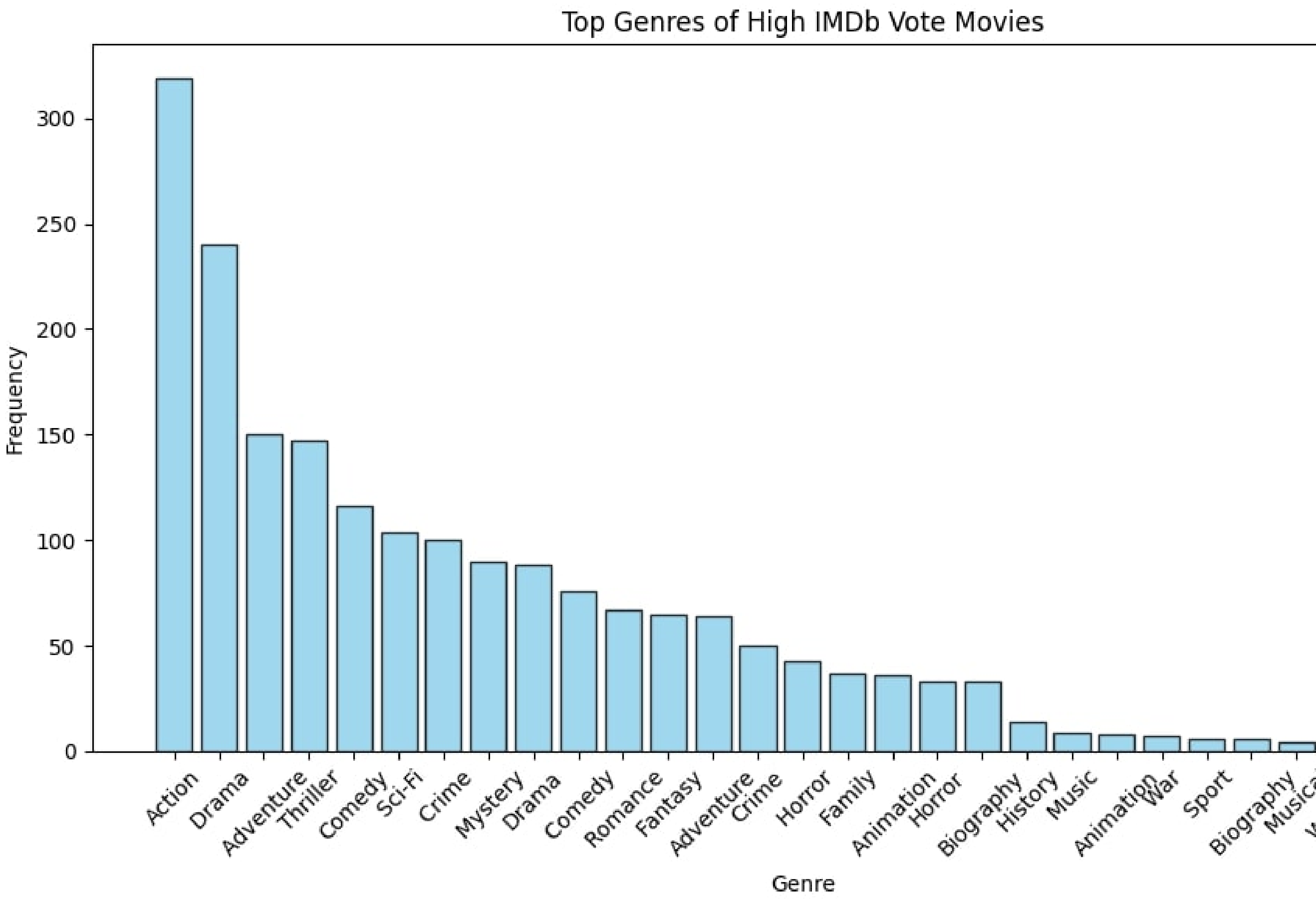


Figure 20: Analyze High-Engagement Movies

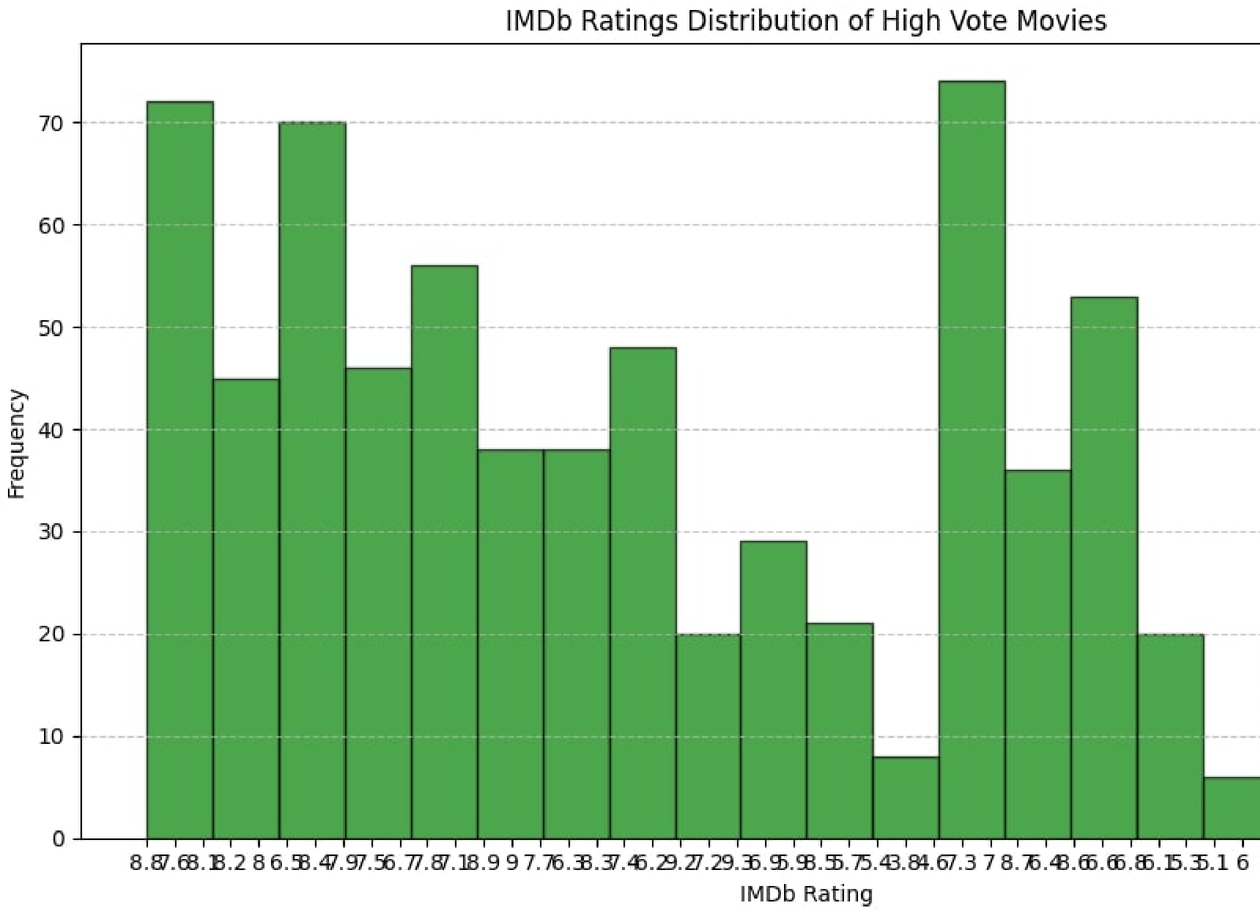


Figure 21: Analyze High-Engagement Movies

7 Conclusion

The project has provided a comprehensive understanding of global movie trends using big data analysis. By leveraging PySpark for distributed data processing and Matplotlib for visualization, we were able to uncover significant trends in movie release patterns, genre popularity, IMDb ratings, and regional availability. The key conclusions drawn from this analysis are:

- Movie production has significantly increased over the years, with a notable rise in the early 2000s.
- Certain genres like Drama and Comedy remain the most popular across different regions, though trends indicate a rise in the popularity of Action and Sci-Fi genres in more recent years.
- The IMDb rating distribution shows a concentration around average ratings, with a few outliers exhibiting extreme ratings.
- The availability of movies is more common in certain countries, especially the United States, pointing to regional disparities in movie distribution.
- High-vote movies tend to have a broader global appeal and often come from top-tier genres.

This analysis provides valuable insights that can help stakeholders in the film industry make data-driven decisions on movie production, distribution, and audience targeting.

8 Citation

For citing this project, please refer to the following:

(a) Git-Hub: `https://github.com/s562904/BigData-Project.git`

(b) Data Set : `https://www.kaggle.com/datasets/octopusteam/full-hulu-dataset`

(c) Matplotlib : `https://matplotlib.org/`