Top 10000 Popular Movies Dataset

Team Name: Data Pulse

Team Members:

Jahnavi Chava - S566948
Kolan Harshini Reddy - S564892
Deepthi Mekhala - S565441

Section: 03
Project ID: 44517

Contents

1 Project Overview

1.1 Project Idea

Recommendation systems have become integral to various platforms such as Netflix, Amazon Prime, YouTube, and online shopping sites, helping users discover content based on their preferences. This dataset provides a valuable starting point for developing recommendation algorithms. Sourced from the official API provided by TMDB, it contains data for 10,000 popular movies, each rated by users on the TMDB platform. The dataset includes key attributes such as movie ID, original language, popularity, release date, average rating, vote count, genre, overview, revenue, runtime, and tagline. The data serves as an ideal foundation for building and testing recommendation systems, enabling researchers and developers to create algorithms that can suggest movies based on various criteria, such as user ratings, genre, and popularity.

2 Technology Summary

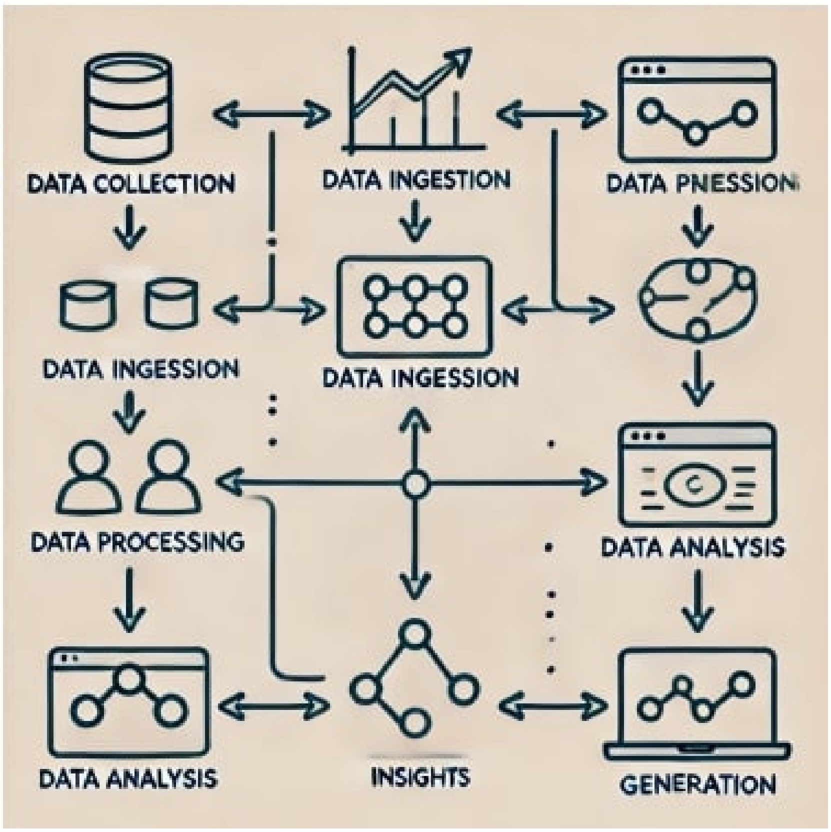
The project leverages the following technologies:

- **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 Summary

The architecture for this project follows a structured approach to ensure efficient data collection, processing, analysis, and visualization. The stages are as follows:

article graphicx



- **Data Collection:** Gather raw sales data from multiple sources such as transactional databases, CRM systems, or CSV files.
- **Data Ingestion:** Import the raw data into a data lake or distributed storage system.
- **Data Processing:** Clean, transform, and structure the data for further analysis using Apache Spark for distributed data processing.
- **Data Analysis:** Extract meaningful patterns and trends from the processed data.
- **Insights Generation:** Visualize findings and provide actionable insights to stakeholders. Tools for this stage include:
  - **BI Tools:** Tableau, Power BI for interactive dashboards.
  - **Visualization Libraries:** Python's Matplotlib, Seaborn, or Plotly for creating visual reports.
  - **Predictive Analytics:** Scikit-learn for machine learning models to forecast sales trends or customer behavior.

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

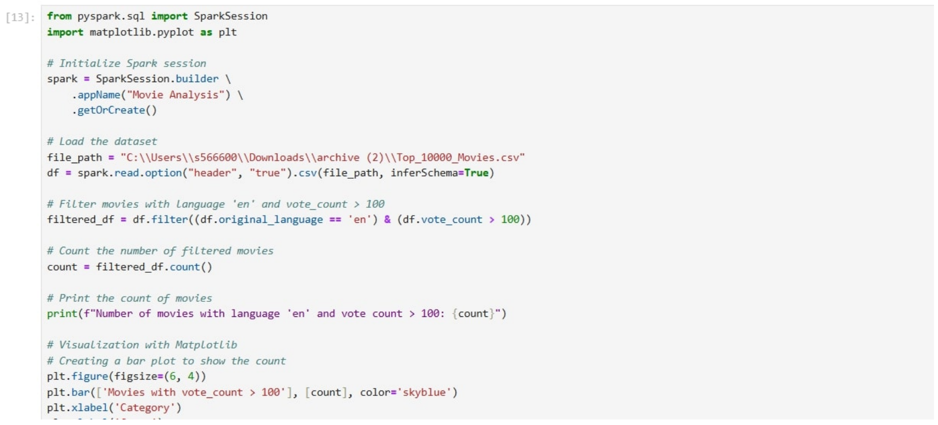
The project has the following goals:

1. **Find Movies with Language ”en” and Vote Count Greater Than 100**
2. **Calculate the Average Revenue of All Movies**
3. **Calculate the Average Revenue of All Movies**
4. **Find the Number of Different Languages in the Dataset.**
5. **Find the Top 10 Most Popular Movies Across All Languages**
6. **Calculate the Average Vote for Movies in Languages: en, no, ru, and pl apply visualization like pie chart**

6 Result Summary

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

- **Goal1** To find movies with the language ”en” (English) and a vote count greater than 100, we filter the dataset based on two conditions. First, we select movies where the language is ”en”. Second, we choose movies that have more than 100 votes, meaning they are more popular. This can be done using a data processing tool like PySpark or Pandas. Once the filtering is done, we get a list of movies that meet both conditions. These filtered movies can then be used for further analysis or recommendations. This process helps in focusing on popular English-language movies with enough ratings to be relevant.



```
[13]: from pyspark.sql import SparkSession
import matplotlib.pyplot as plt

# Initialize Spark session
spark = SparkSession.builder \
    .appName("Movie Analysis") \
    .getOrCreate()

# Load the dataset
file_path = "C:\\Users\\s566600\\Downloads\\archive (2)\\Top_10000_Movies.csv"
df = spark.read.option("header", "true").csv(file_path, inferSchema=True)

# Filter movies with language 'en' and vote_count > 100
filtered_df = df.filter((df.original_language == 'en') & (df.vote_count > 100))

# Count the number of filtered movies
count = filtered_df.count()

# Print the count of movies
print(f"Number of movies with language 'en' and vote count > 100: {count}")

# Visualization with Matplotlib
# Creating a bar plot to show the count
plt.figure(figsize=(6, 4))
plt.bar(['Movies with vote_count > 100'], [count], color='skyblue')
plt.xlabel('Category')
```

Figure 1: goal 1

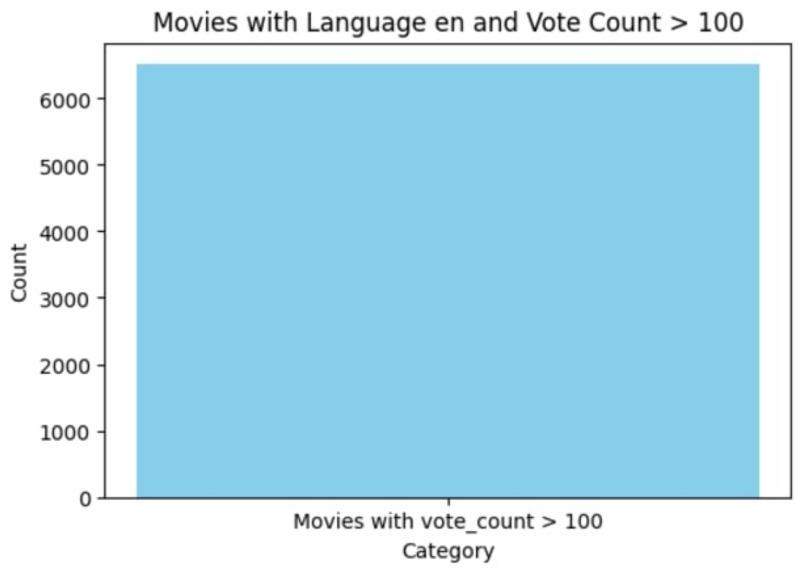
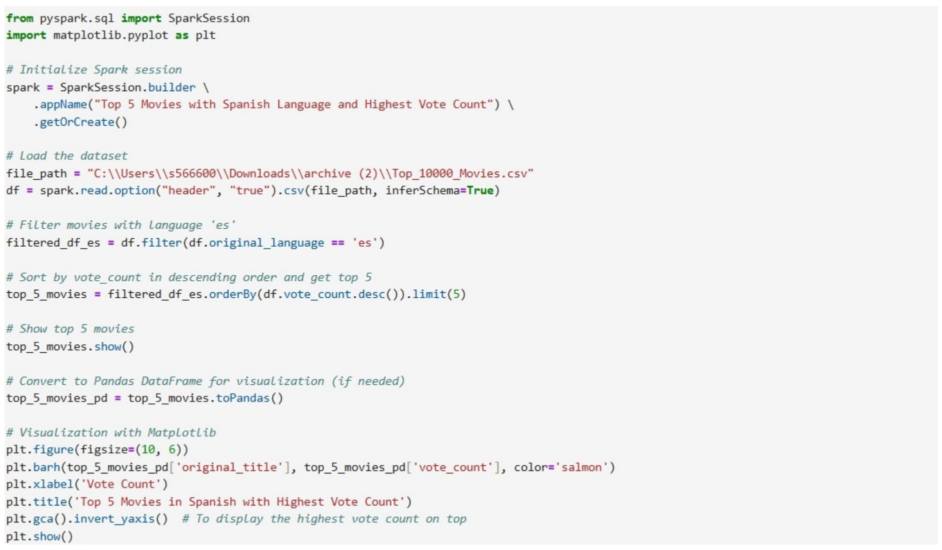


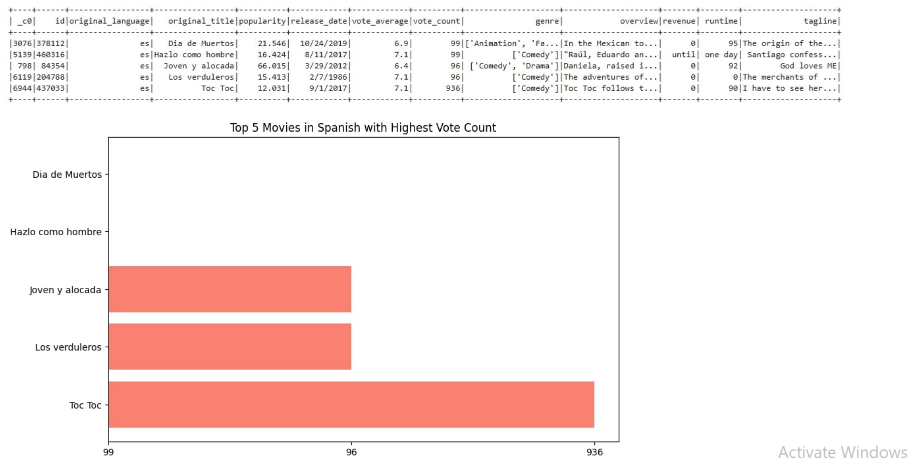
Figure 2: goal 1

- **Goal 2** To calculate the average revenue of all movies in the dataset, the process involves three key steps. First, the total revenue of all the movies needs to be summed up. Next, the total number of movies in the dataset is determined. Finally, the average revenue is calculated by dividing the total revenue by the number of movies. This average revenue provides insight into the financial performance of the movies in the dataset, helping to understand how much, on average, each movie has earned.



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

# Initialize Spark session
spark = SparkSession.builder \
    .appName("Top 5 Movies with Spanish Language and Highest Vote Count") \
    .getOrCreate()

# Load the dataset
file_path = "C:\\Users\\s566600\\Downloads\\archive (2)\\Top_10000_Movies.csv"
df = spark.read.option("header", "true").csv(file_path, inferSchema=True)

# Filter movies with language 'es'
filtered_df_es = df.filter(df.original_language == 'es')

# Sort by vote_count in descending order and get top 5
top_5_movies = filtered_df_es.orderBy(df.vote_count.desc()).limit(5)

# Show top 5 movies
top_5_movies.show()

# Convert to Pandas DataFrame for visualization (if needed)
top_5_movies_pd = top_5_movies.toPandas()

# Visualization with Matplotlib
plt.figure(figsize=(10, 6))
plt.barh(top_5_movies_pd['original_title'], top_5_movies_pd['vote_count'], color='salmon')
plt.xlabel('Vote Count')
plt.title('Top 5 Movies in Spanish with Highest Vote Count')
plt.gca().invert_yaxis()  # To display the highest vote count on top
plt.show()
```

Figure 3: goal 2

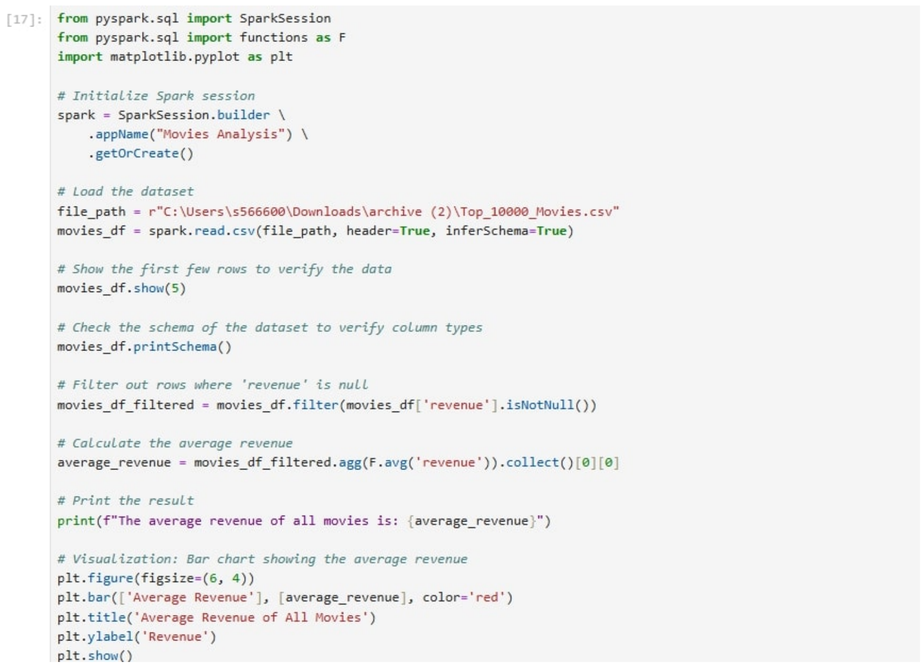


```
+----+------+-----------------+-----------------+----------+------------+------------+----------+-------------------+-------------------+-------+--------+--------------------+
| _c0|    id|original_language|   original_title|popularity|release_date|vote_average|vote_count|              genre|           overview|revenue| runtime|             tagline|
+----+------+-----------------+-----------------+----------+------------+------------+----------+-------------------+-------------------+-------+--------+--------------------+
|3076|378112|               es|   Dia de Muertos|    21.546|  10/24/2019|         6.9|        99|['Animation', 'Fa...|In the Mexican to...|      0|      95|The origin of the...|
|5139|460316|               es|Hazlo como hombre|    16.424|   8/11/2017|         7.1|        99|        ['Comedy']|"Raúl, Eduardo an...|  until| one day| Santiago confess...|
| 798| 84354|               es|  Joven y alocada|    66.015|   3/29/2012|         6.4|        96| ['Comedy', 'Drama']|Daniela, raised i...|      0|      92|         God loves ME|
|6119|204788|               es|   Los verduleros|    15.413|    2/7/1986|         7.1|        96|        ['Comedy']|The adventures of...|      0|       0|The merchants of ...|
|6944|437033|               es|          Toc Toc|    12.031|    9/1/2017|         7.1|       936|        ['Comedy']|Toc Toc follows t...|      0|      90|I have to see her...|
+----+------+-----------------+-----------------+----------+------------+------------+----------+-------------------+-------------------+-------+--------+--------------------+
```

Activate Windows

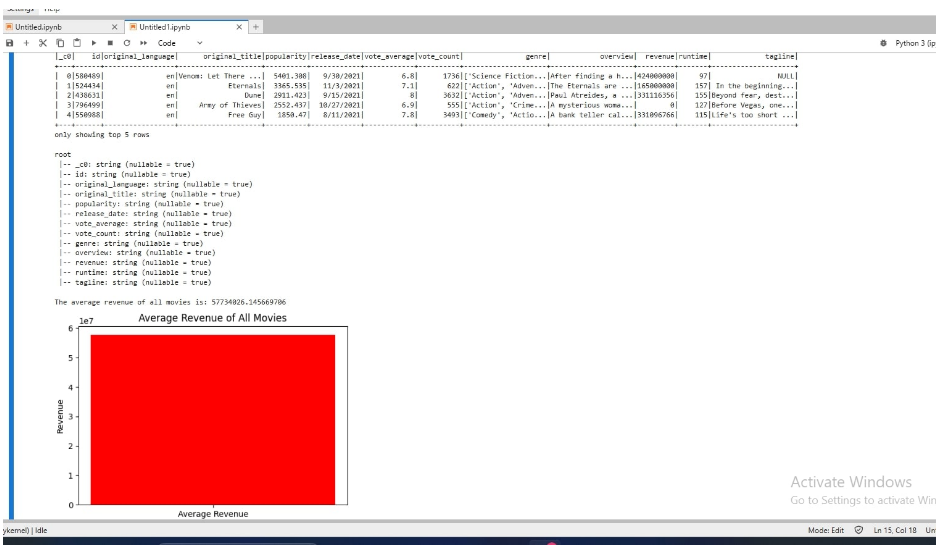
Figure 4: goal 2

- **Goal 3** To calculate the average revenue of all movies in the dataset, you need to sum up the revenue of each movie and then divide that total by the number of movies. This process involves first gathering the revenue data for all movies, then counting the total number of movies in the dataset, and finally calculating the average by dividing the total revenue by the count of movies. The result gives an understanding of the average financial earnings per movie, which can provide insights into the overall revenue trends within the dataset.



```
[17]: from pyspark.sql import SparkSession
      from pyspark.sql import functions as F
      import matplotlib.pyplot as plt

      # Initialize Spark session
      spark = SparkSession.builder \
          .appName("Movies Analysis") \
          .getOrCreate()

      # Load the dataset
      file_path = r"C:\Users\s566600\Downloads\archive (2)\Top_10000_Movies.csv"
      movies_df = spark.read.csv(file_path, header=True, inferSchema=True)

      # Show the first few rows to verify the data
      movies_df.show(5)

      # Check the schema of the dataset to verify column types
      movies_df.printSchema()

      # Filter out rows where 'revenue' is null
      movies_df_filtered = movies_df.filter(movies_df['revenue'].isNotNull())

      # Calculate the average revenue
      average_revenue = movies_df_filtered.agg(F.avg('revenue')).collect()[0][0]

      # Print the result
      print(f"The average revenue of all movies is: {average_revenue}")

      # Visualization: Bar chart showing the average revenue
      plt.figure(figsize=(6, 4))
      plt.bar(['Average Revenue'], [average_revenue], color='red')
      plt.title('Average Revenue of All Movies')
      plt.ylabel('Revenue')
      plt.show()
```

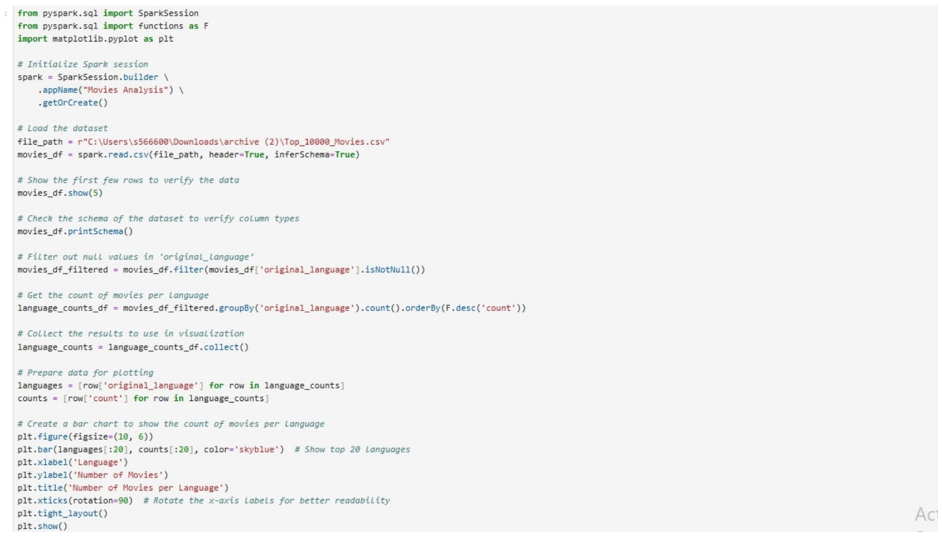
Figure 5: goal 3



```
|_c0|    id|original_language|      original_title|popularity|release_date|vote_average|vote_count|               genre|            overview|  revenue|runtime|             tagline|
+---+------+-----------------+--------------------+----------+------------+------------+----------+--------------------+--------------------+---------+-------+--------------------+
|  0|580489|               en|Venom: Let There ...|  5401.308|   9/30/2021|         6.8|      1736|['Science Fiction...|After finding a h...|424000000|     97|                NULL|
|  1|524434|               en|            Eternals|  3365.535|   11/3/2021|         7.1|       622|['Action', 'Adven...|The Eternals are ...|165000000|    157| In the beginning...|
|  2|438631|               en|                Dune|  2911.423|   9/15/2021|           8|      3632|['Action', 'Adven...|Paul Atreides, a ...|331116356|    155|Beyond fear, dest...|
|  3|796499|               en|     Army of Thieves|  2552.437|  10/27/2021|         6.9|       555|['Action', 'Crime...|A mysterious woma...|        0|    127|Before Vegas, one...|
|  4|550988|               en|            Free Guy|   1850.47|   8/11/2021|         7.8|      3493|['Comedy', 'Actio...|A bank teller cal...|331096766|    115|Life's too short ...|
+---+------+-----------------+--------------------+----------+------------+------------+----------+--------------------+--------------------+---------+-------+--------------------+
only showing top 5 rows

root
 |-- _c0: string (nullable = true)
 |-- id: string (nullable = true)
 |-- original_language: string (nullable = true)
 |-- original_title: string (nullable = true)
 |-- popularity: string (nullable = true)
 |-- release_date: string (nullable = true)
 |-- vote_average: string (nullable = true)
 |-- vote_count: string (nullable = true)
 |-- genre: string (nullable = true)
 |-- overview: string (nullable = true)
 |-- revenue: string (nullable = true)
 |-- runtime: string (nullable = true)
 |-- tagline: string (nullable = true)

The average revenue of all movies is: 57734026.145669706
```

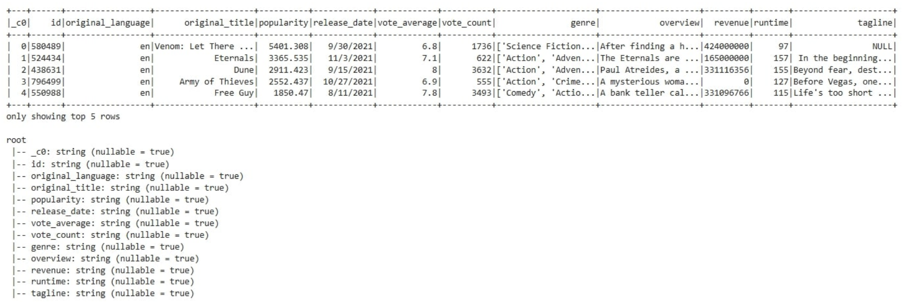
Figure 6: goal 3

- **Goal 4** To find the number of different languages in the dataset, you need to identify the unique values in the ”originallanguage” column. This involves extracting all the language codes from the dataset and then counting the distinct ones. The result will give you the total number of different languages represented in the dataset, providing insights into the linguistic diversity of the movies included. This can be useful for analyzing the global reach and language distribution of the movies.



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

# Initialize Spark session
spark = SparkSession.builder \
    .appName("Movies Analysis") \
    .getOrCreate()

# Load the dataset
file_path = r"C:\Users\s566600\Downloads\archive (2)\Top_10000_Movies.csv"
movies_df = spark.read.csv(file_path, header=True, inferSchema=True)

# Show the first few rows to verify the data
movies_df.show(5)

# Check the schema of the dataset to verify column types
movies_df.printSchema()

# Filter out null values in 'original_language'
movies_df_filtered = movies_df.filter(movies_df['original_language'].isNotNull())

# Get the count of movies per language
language_counts_df = movies_df_filtered.groupBy('original_language').count().orderBy(F.desc('count'))

# Collect the results to use in visualization
language_counts = language_counts_df.collect()

# Prepare data for plotting
languages = [row['original_language'] for row in language_counts]
counts = [row['count'] for row in language_counts]

# Create a bar chart to show the count of movies per language
plt.figure(figsize=(10, 6))
plt.bar(languages[:20], counts[:20], color='skyblue')  # Show top 20 languages
plt.xlabel('Language')
plt.ylabel('Number of Movies')
plt.title('Number of Movies per Language')
plt.xticks(rotation=90)  # Rotate the x-axis labels for better readability
plt.tight_layout()
plt.show()
```

Figure 7: goal 4



```
+---+------+-----------------+--------------------+----------+------------+------------+----------+--------------------+--------------------+---------+-------+--------------------+
|_c0|    id|original_language|      original_title|popularity|release_date|vote_average|vote_count|               genre|            overview|  revenue|runtime|             tagline|
+---+------+-----------------+--------------------+----------+------------+------------+----------+--------------------+--------------------+---------+-------+--------------------+
|  0|580489|               en|Venom: Let There ...|  5401.308|   9/30/2021|         6.8|      1736|['Science Fiction...|After finding a h...|424000000|     97|                NULL|
|  1|524434|               en|            Eternals|  3365.535|   11/3/2021|         7.1|       622|['Action', 'Adven...|The Eternals are ...|165000000|    157| In the beginning...|
|  2|438631|               en|                Dune|  2911.423|   9/15/2021|           8|      3632|['Action', 'Adven...|Paul Atreides, a ...|331116356|    155|Beyond fear, dest...|
|  3|796499|               en|     Army of Thieves|  2552.437|  10/27/2021|         6.9|       555|['Action', 'Crime...|A mysterious woma...|        0|    127|Before Vegas, one...|
|  4|550988|               en|            Free Guy|   1850.47|   8/11/2021|         7.8|      3493|['Comedy', 'Actio...|A bank teller cal...|331096766|    115|Life's too short ...|
+---+------+-----------------+--------------------+----------+------------+------------+----------+--------------------+--------------------+---------+-------+--------------------+
only showing top 5 rows

root
 |-- _c0: string (nullable = true)
 |-- id: string (nullable = true)
 |-- original_language: string (nullable = true)
 |-- original_title: string (nullable = true)
 |-- popularity: string (nullable = true)
 |-- release_date: string (nullable = true)
 |-- vote_average: string (nullable = true)
 |-- vote_count: string (nullable = true)
 |-- genre: string (nullable = true)
 |-- overview: string (nullable = true)
 |-- revenue: string (nullable = true)
 |-- runtime: string (nullable = true)
 |-- tagline: string (nullable = true)
```

Figure 8: goal 4

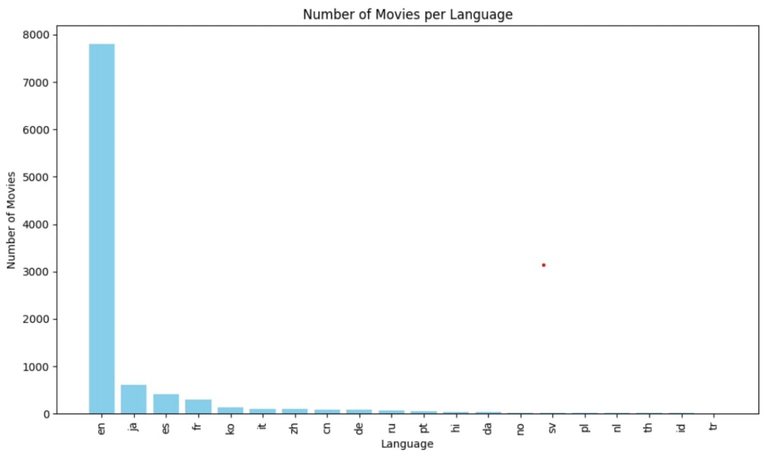
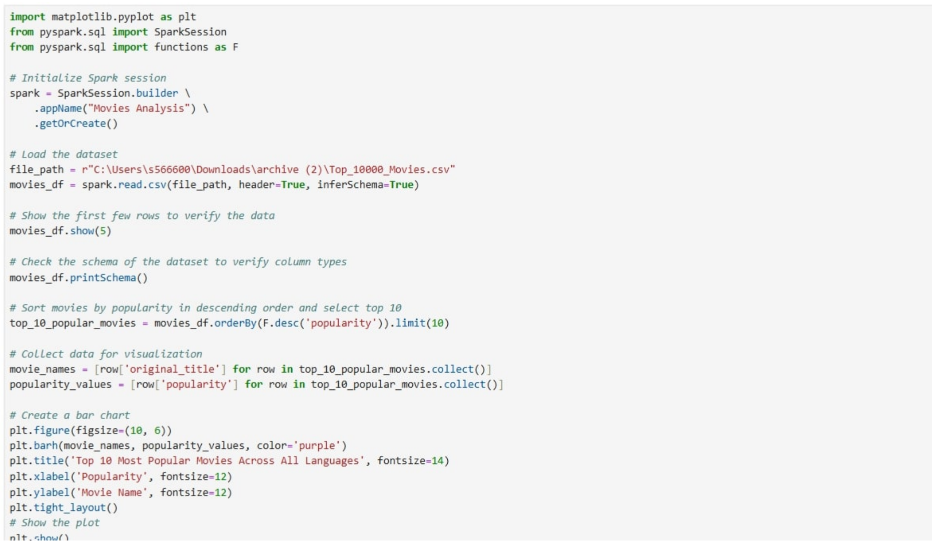


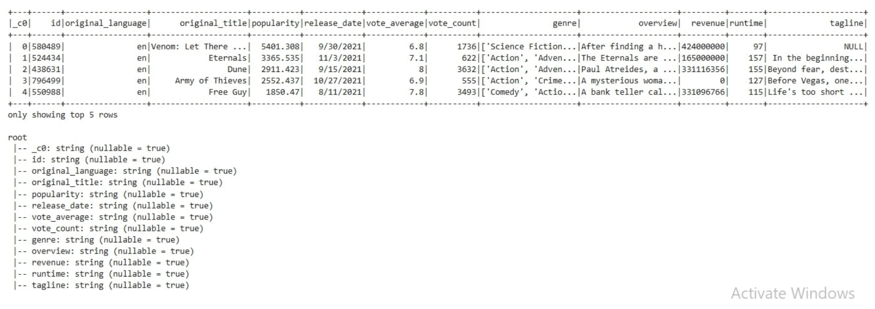
Figure 9: goal 4

- **Goal 5** To find the top 10 most popular movies across all languages, you need to sort the movies based on their popularity scores in descending order. By identifying the ”popularity” column, you can arrange the movies from the highest to the lowest popularity. After sorting, select the top 10 movies with the highest popularity values. This process helps in identifying the most popular movies, regardless of language, offering insights into global trends and preferences. First image



```
import matplotlib.pyplot as plt
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

# Initialize Spark session
spark = SparkSession.builder \
    .appName("Movies Analysis") \
    .getOrCreate()

# Load the dataset
file_path = r"C:\Users\s566600\Downloads\archive (2)\Top_10000_Movies.csv"
movies_df = spark.read.csv(file_path, header=True, inferSchema=True)

# Show the first few rows to verify the data
movies_df.show(5)

# Check the schema of the dataset to verify column types
movies_df.printSchema()

# Sort movies by popularity in descending order and select top 10
top_10_popular_movies = movies_df.orderBy(F.desc('popularity')).limit(10)

# Collect data for visualization
movie_names = [row['original_title'] for row in top_10_popular_movies.collect()]
popularity_values = [row['popularity'] for row in top_10_popular_movies.collect()]

# Create a bar chart
plt.figure(figsize=(10, 6))
plt.barh(movie_names, popularity_values, color='purple')
plt.title('Top 10 Most Popular Movies Across All Languages', fontsize=14)
plt.xlabel('Popularity', fontsize=12)
plt.ylabel('Movie Name', fontsize=12)
plt.tight_layout()
# Show the plot
plt.show()
```

Figure 10: goal 5



```
+---+------+-----------------+--------------------+----------+------------+------------+----------+--------------------+--------------------+---------+-------+--------------------+
|_c0|    id|original_language|      original_title|popularity|release_date|vote_average|vote_count|               genre|            overview|  revenue|runtime|             tagline|
+---+------+-----------------+--------------------+----------+------------+------------+----------+--------------------+--------------------+---------+-------+--------------------+
|  0|580489|               en|Venom: Let There ...|  5401.308|   9/30/2021|         6.8|      1736|['Science Fiction...|After finding a h...|424000000|     97|                NULL|
|  1|524434|               en|            Eternals|  3365.535|   11/3/2021|         7.1|       622|['Action', 'Adven...|The Eternals are ...|165000000|    157| In the beginning...|
|  2|438631|               en|                Dune|  2911.423|   9/15/2021|           8|      3632|['Action', 'Adven...|Paul Atreides, a ...|331116356|    155|Beyond fear, dest...|
|  3|796499|               en|     Army of Thieves|  2552.437|  10/27/2021|         6.9|       555|['Action', 'Crime...|A mysterious woma...|        0|    127|Before Vegas, one...|
|  4|550988|               en|            Free Guy|   1850.47|   8/11/2021|         7.8|      3493|['Comedy', 'Actio...|A bank teller cal...|331096766|    115|Life's too short ...|
+---+------+-----------------+--------------------+----------+------------+------------+----------+--------------------+--------------------+---------+-------+--------------------+
only showing top 5 rows

root
 |-- _c0: string (nullable = true)
 |-- id: string (nullable = true)
 |-- original_language: string (nullable = true)
 |-- original_title: string (nullable = true)
 |-- popularity: string (nullable = true)
 |-- release_date: string (nullable = true)
 |-- vote_average: string (nullable = true)
 |-- vote_count: string (nullable = true)
 |-- genre: string (nullable = true)
 |-- overview: string (nullable = true)
 |-- revenue: string (nullable = true)
 |-- runtime: string (nullable = true)
 |-- tagline: string (nullable = true)
```

Figure 11: goal 5

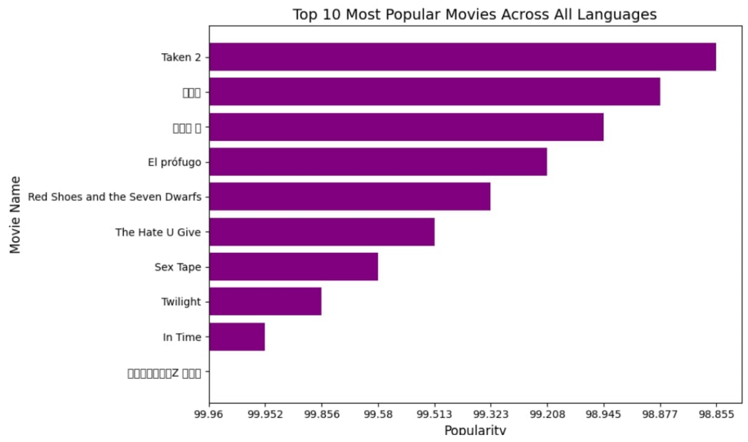
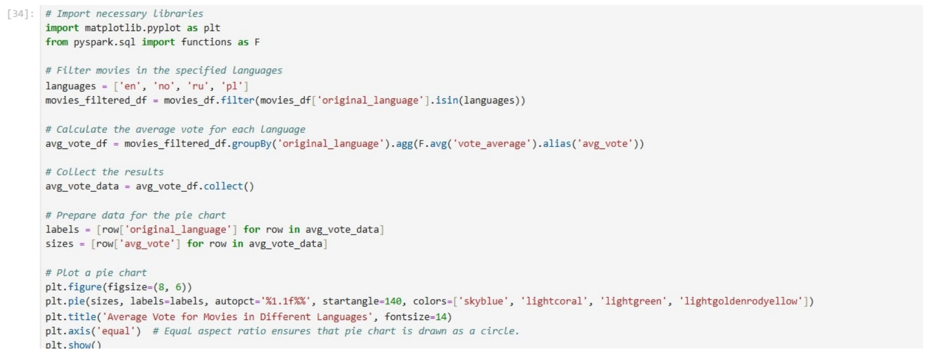


Figure 12: goal 5

- **Goal 6** To calculate the average vote for movies in the languages ”en” (English), ”no” (Norwegian), ”ru” (Russian), and ”pl” (Polish), you would first filter the dataset for these specific languages. Then, you would calculate the average vote for each language by averaging the values in the ”vote average” column for each language group. Once the averages are calculated, you can visualize the results using a pie chart to represent the distribution of average votes across these languages. The pie chart would provide a clear visual comparison of the average ratings for each language group, highlighting any differences or trends. First image



```
[34]: # Import necessary libraries
      import matplotlib.pyplot as plt
      from pyspark.sql import functions as F

      # Filter movies in the specified languages
      languages = ['en', 'no', 'ru', 'pl']
      movies_filtered_df = movies_df.filter(movies_df['original_language'].isin(languages))

      # Calculate the average vote for each language
      avg_vote_df = movies_filtered_df.groupBy('original_language').agg(F.avg('vote_average').alias('avg_vote'))

      # Collect the results
      avg_vote_data = avg_vote_df.collect()

      # Prepare data for the pie chart
      labels = [row['original_language'] for row in avg_vote_data]
      sizes = [row['avg_vote'] for row in avg_vote_data]

      # Plot a pie chart
      plt.figure(figsize=(8, 6))
      plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, colors=['skyblue', 'lightcoral', 'lightgreen', 'lightgoldenrodyellow'])
      plt.title('Average Vote for Movies in Different Languages', fontsize=14)
      plt.axis('equal')  # Equal aspect ratio ensures that pie chart is drawn as a circle.
      plt.show()
```

Figure 13: goal 6

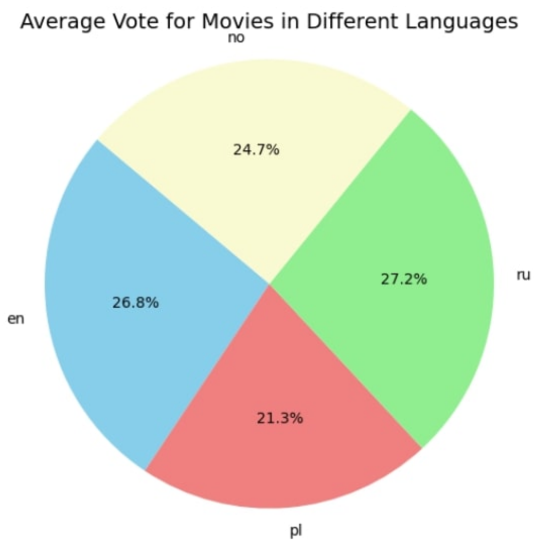


Figure 14: goal 6

Conclusion

The dataset provides valuable information on 10,000 popular movies based on TMDB ratings. With columns such as movie ID, language, title, popularity, release date, ratings, genre, revenue, and runtime, this dataset serves as an ideal starting point for building recommendation systems. The addition of the overview, revenue, runtime, and tagline columns enhances the dataset's usefulness for deeper analysis and model building. This structured data can be leveraged to create algorithms that suggest movies based on various features like genre, ratings, and popularity.

7 Citation

For citing this project, please refer to the following:

1. Dataset: Top 10000 Popular Movies Dataset
2. Python Libraries: pandas, numpuy, matplotlib, seaborn, plotly.
3. Dataset source: `https://www.kaggle.com/datasets/omkarborikar/top-10000-popular-movies`
4. Tools: Jupyter Notebook, GitHub.
5. GitHub link: `https://github.com/harshinikolan/BigData-Project`