Data Visualization Track Requirements

Data and Delivery (20 points)

- The dataset contains at least 100 unique records. (5 points)
- A database is used to house the data (SQL, MongoDB, SQLite, etc.). (5 points)
- The GitHub repo has a README.md that includes the following: (10 points)
    - An overview of the project and its purpose
    - Instructions on how to use and interact with the project
    - At least one paragraph summarizing efforts for ethical considerations made in the project
    - References for the data source(s)
    - References for any code used that is not your own

Visualizations (25 points)

- A minimum of three unique views present the data. (10 points)
- The visualizations are presented in a clear, digestible manner. (5 points)
- The data story is easy to interpret for users of all levels. (10 points)

Usability (30 points)

- The script, notebook, or webpage created to showcase data visualizations runs without error. (10 points)
- A Python or JavaScript library not shown in class is used in the project. (10 points)
- The project includes some level of user-driven interaction, conforming to one of the following designs: (10 points)
    - HTML menus, dropdowns, and/or textboxes to display JavaScript-powered visualizations
    - Flask backend with interactive API routes that serve back Python or JavaScript created plots
    - Visualizations created from user-selected filtered data

Below is a Python code to help you meet the project requirements, including visualizations, using the dataset you provided.

1. Dataset Summary and Analysis:

python

Copy code

```
import pandas as pd


# Load the dataset
file_path = '/mnt/data/imdb_top_1000.csv'
df = pd.read_csv(file_path)


# Display summary of the dataset
summary = df.describe(include='all')
print(summary)
```

2. Visualizations:

1. Distribution of IMDb Ratings (Histogram):

python

Copy code

```
import matplotlib.pyplot as plt


# Plot histogram for IMDb ratings
plt.hist(df['IMDB_Rating'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of IMDb Ratings')
plt.xlabel('IMDb Rating')
plt.ylabel('Frequency')
plt.show()
```

2. Gross Earnings vs. IMDb Rating (Scatter Plot):

python

Copy code

```
# Remove any rows with missing or invalid data for 'Gross' and 'IMDB_Rating'
df_cleaned = df.dropna(subset=['Gross', 'IMDB_Rating'])

# Convert Gross to numeric, removing commas and non-numeric characters
df_cleaned['Gross'] = df_cleaned['Gross'].replace({',': '', '$': ''}, regex=True).astype(float)

# Scatter plot for IMDb Rating vs Gross
plt.scatter(df_cleaned['IMDB_Rating'], df_cleaned['Gross'], alpha=0.5, color='green')
plt.title('IMDb Rating vs Gross Earnings')
plt.xlabel('IMDb Rating')
plt.ylabel('Gross Earnings (in USD)')
plt.show()
```

3. Top Genres by Count (Bar Chart):

python

Copy code

```
# Extract top genres
genre_counts = df['Genre'].str.split(',').explode().value_counts().head(10)

# Plot top 10 genres
genre_counts.plot(kind='bar', color='orange', edgecolor='black')
plt.title('Top 10 Genres by Frequency')
plt.xlabel('Genre')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.show()
```

3. Usability and Interaction:

You can create an interactive web application using **Flask** or **Dash**. Below is a simple example using **Flask** for a user-driven interactive interface. This would allow users to select a genre and view the corresponding movies:

Flask Backend Example:

python

Copy code

```
from flask import Flask, render_template, request
import pandas as pd

app = Flask(__name__)

# Load the dataset
df = pd.read_csv('/path/to/imdb_top_1000.csv')

@app.route('/')
def index():
    genres = df['Genre'].str.split(',').explode().unique()
    return render_template('index.html', genres=genres)

@app.route('/filter', methods=['GET'])
def filter_by_genre():
    genre = request.args.get('genre')
    filtered_df = df[df['Genre'].str.contains(genre, na=False)]
    return render_template('filter_results.html', tables=[filtered_df.to_html(classes='data')])

if __name__ == '__main__':
    app.run(debug=True)
```

HTML Template (index.html):

html

Copy code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Filter Movies by Genre</title>
</head>
<body>
  <h1>Select a Genre to Filter Movies</h1>
  <form action="/filter" method="get">
    <label for="genre">Choose a genre:</label>
    <select name="genre" id="genre">
      {% for genre in genres %}
        <option value="{{ genre }}">{{ genre }}</option>
      {% endfor %}
    </select>
    <button type="submit">Filter</button>
  </form>
</body>
</html>
```

This will allow users to select a genre and see the filtered movie data.

---

4. Adding Ethical Considerations to README.md:

Example paragraph for ethical considerations:

markdown

Copy code

```
## Ethical Considerations
```

The data used in this project is publicly available and sourced from IMDb, which is a widely recognized platform for movie ratings and reviews. We ensured that the data used in this project adheres to open data standards and respects user privacy, as it does not include any personally identifiable information. Any references to actors, directors, or other individuals are publicly available from movie credits and are used solely for analytical purposes. The project was developed with the intent to provide insights and visualizations that can aid users in exploring movie trends.

---

5. GitHub README.md Example:

markdown

Copy code

IMDb Top 1000 Movies Analysis

Overview

This project aims to analyze the top 1000 movies on IMDb based on various parameters like IMDb ratings, gross earnings, and genres. The dataset provides insights into movie trends, and visualizations have been created to explore these insights in an interactive manner.

How to Use

1. Clone the repository.

2. Install the required dependencies using:

  ```bash

  pip install -r requirements.txt

3. Run the Flask app using:

bash

Copy code

python app.py

4. Access the interactive visualizations by navigating to http://127.0.0.1:5000/.

**Ethical Considerations**
  ```

The data used in this project is publicly available and sourced from IMDb. No personally identifiable information has been used, and all references to individuals (actors, directors) are from publicly accessible credits. We prioritize transparency and respect for privacy in all aspects of this project.

References

- IMDb dataset: IMDb Top 1000 Movies
- Code inspiration and external libraries: Flask Documentation, Matplotlib