

Movie Ratings Dataset Exploration with Pivot Tables

# Project Description:

* This project analyzes a movie ratings dataset using Python (Pandas) and builds an interactive web application with **Streamlit**. The system provides insights into user preferences, most popular movies, genre performance, and ratings distribution. It makes use of **pivot tables, grouping, filtering, and aggregation**. Additional features include exporting results into CSV files and generating visual insights with charts.  
    
  The interactive webpage allows users to:
* Filter by genre, user, or rating range.
* Explore pivot tables (average ratings, counts, etc.).
* Visualize results with bar charts and pie charts.
* Export cleaned datasets and pivot tables for further use.

**BY:- Name:** Unnati M P

**Institution:** GSSS Institute of Engineering and Technology for Women

**Batch:** 2023 – 2027

**Branch:** Artificial Intelligence and Data Science (AI&DS)

# Index

1. Project Overview
2. Use-Case Explanations
3. Algorithms / Approach
4. UML Diagrams
5. Front-End (Interface) Design
6. Setup Instructions
7. Code & Explanation
8. Screenshots of Output
9. Closure / Bibliography

# Detailed Explanation

## Overview

The project demonstrates the use of pivot tables and visualizations in analyzing movie ratings data. It integrates data from Movies, Ratings, and Users datasets to produce meaningful insights. The **Movie Rating Analysis** project is designed to explore and analyze movie rating datasets by leveraging **Python (Pandas, Matplotlib, Seaborn)** and an interactive **Streamlit web application**. The system integrates three key datasets — **movies**, **users**, and **ratings** — to provide comprehensive insights into audience preferences, genre popularity, and rating behavior.

The project begins with **data preprocessing**, where missing values are handled, duplicates are removed, and datasets are merged into a single cleaned dataset. Next, derived attributes such as **Rating Categories (High/Medium/Low)** are introduced to enhance analysis.

The application implements multiple **pivot tables and visualizations** that highlight important use-cases, including:

* Average rating per movie, genre, and user.
* Count of ratings per movie to identify popular titles.
* Distribution of ratings across categories (High/Medium/Low).
* Interactive filtering by genre, user, and rating range.

# 📖 Use-Case Explanations for Movie Rating Analysis Project

This section provides a detailed explanation of the major **use-cases** implemented in the project. Each use-case highlights the **goal**,and **outcome**, supported by pivot tables, visualizations, and export features.

## 🎥 ****Use-Case 1: Average Rating per Movie****

* **Goal**: To compute the mean rating of each movie and understand which movies are most loved by viewers.
* **Methodology**:
  + A **pivot table** is created with Title as the index and the average of Rating as values.
  + A bar chart is used to visualize the ratings per movie.
* **Outcome**:
  + Enables identification of movies with consistently high audience approval.
  + Helps filter movies by average rating ≥ 4 to highlight critically appreciated titles.

## 🎭 ****Use-Case 2: Average Rating per Genre****

* **Goal**: To analyze how different genres perform in terms of viewer ratings.
* **Methodology**:
  + A **pivot table** groups movies by Genre and calculates the average rating.
  + Visualization uses a **horizontal bar chart** with genres on the Y-axis.
* **Outcome**:
  + Helps discover the most popular genres.
  + Useful for recommendations, such as suggesting genres with higher engagement and satisfaction.

## 👤 ****Use-Case 3: Average Rating per User****

* **Goal**: To study user behavior in rating movies and check for bias or consistency.
* **Methodology**:
  + A pivot table groups by UserID to calculate each user’s average given rating.
  + A **line chart** visualizes variations among users.
* **Outcome**:
  + Helps identify generous users (who often rate higher) and critical users (who rate lower).
  + Assists in detecting rating anomalies for further recommendation refinement.

## ⭐ ****Use-Case 4: Count of Ratings per Movie (Most Rated)****

* **Goal**: To identify movies that received the highest number of ratings, representing popularity.
* **Methodology**:
  + A pivot table counts the number of ratings per Title.
  + A **bar plot** (Top 10) visualizes the most-rated movies.
* **Outcome**:
  + Indicates popularity, even if not always high-rated.
  + Useful for identifying trending or widely viewed movies.

## 🥧 ****Use-Case 5: Ratings Distribution (High/Medium/Low)****

* **Goal**: To categorize ratings into High (≥4), Medium (≥3 and <4), and Low (<3) and understand their distribution.
* **Methodology**:
  + A new column RatingCategory classifies each rating.
  + A **pie chart** displays the percentage distribution of these categories.
* **Outcome**:
  + Provides insights into overall satisfaction trends.
  + Shows whether the dataset is skewed toward positive or negative ratings.

## 📂 ****Use-Case 6: Export of Data and Results****

* **Goal**: To allow users to save processed results for further use.
* **Methodology**:
  + Export buttons are implemented to save pivot tables and cleaned datasets (CSV format).
  + Available exports include:
    - movie\_avg\_ratings.csv
    - genre\_avg\_ratings.csv
    - user\_avg\_ratings.csv
    - movie\_rating\_counts.csv
    - ratings\_distribution.csv
    - cleaned\_movie\_ratings.csv
* **Outcome**:
  + Enhances usability for researchers and analysts.
  + Provides offline access for deeper study or reporting.

# ⚙️ ****Algorithms / Approach****

The **Movie Rating Analysis** applies a structured **data analytics pipeline** with systematic steps to process, analyze, and visualize the dataset. The approach can be summarized as follows:

### ****1. Data Collection & Loading****

* Three datasets were provided:
  + **movies.csv** → Contains movie titles and genres.
  + **ratings.csv** → Contains user ratings for movies.
  + **users.csv** → Contains user information (e.g., UserID, demographics if available).
* These datasets were imported into the system using **Pandas** for efficient handling of structured data.

### ****2. Data Preprocessing****

* **Merging Datasets**:
  + First, ratings.csv was merged with movies.csv using MovieID.
  + Then, the result was merged with users.csv using UserID.
  + This produced a **cleaned and unified dataset** containing movies, ratings, and user details.
* **Handling Missing Values**:
  + Checked for null values in all datasets.
  + Retained NaN values for ratings if found (optional fill with 0).
* **Removing Duplicates**: Ensured each (UserID, MovieID) pair is unique.

### ****3. Feature Engineering****

* Introduced a derived column **RatingCategory**:
  + **High** → Ratings ≥ 4
  + **Medium** → Ratings = 3
  + **Low** → Ratings ≤ 2
* This categorization allowed easy visualization of audience sentiment.

### ****4. Analytical Use-Cases (Pivot Tables & Grouping)****

Implemented **pivot tables** and **group-by operations** to support the following insights:

1. **Average Rating per Movie** → Identifies overall reception of each film.
2. **Average Rating per Genre** → Shows which genres are more liked by users.
3. **Average Rating per User** → Tracks rating behavior of individuals.
4. **Count of Ratings per Movie** → Identifies most-rated (popular) movies.

### ****5. Visualization****

* Used **Seaborn** and **Matplotlib** for visually appealing charts.
* Plots included:
  + **Bar Charts** → For Top 5 movies, genre averages, and most-rated movies.
  + **Pie Chart** → For distribution of High/Medium/Low ratings.
  + **Line Chart** → For average rating per user.
* Adopted **color palettes (crest, viridis, mako)** for aesthetic consistency.

### ****6. Interactivity (Streamlit Integration)****

* **Sidebar Filters** → Genre, User, Rating Range.
* **Expandable Previews** → Raw datasets preview.
* **Export Options** → Ability to save pivot tables and final cleaned dataset as CSV.
* **Logo Integration** → A custom Movie Rating Analysis logo added for branding.

### ****7. Output & Export****

* Exported CSV files for:
  + movie\_avg\_ratings.csv
  + genre\_avg\_ratings.csv
  + user\_avg\_ratings.csv
  + movie\_rating\_counts.csv
  + cleaned\_movie\_ratings.csv
* Ensured project outputs can be reused for external analysis and reporting.

# 🎯 ****UML Diagrams****

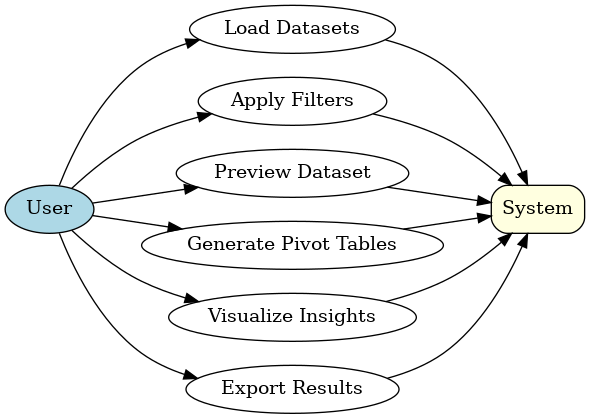
## ****1. Use-Case Diagram****

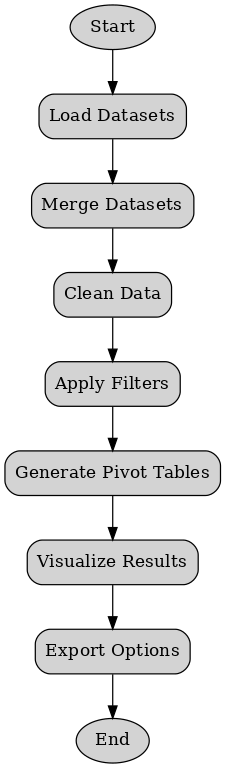
This diagram shows the **interactions between the user and the system**.

**Actors:**

* **End User (Student/Analyst)** → explores ratings, applies filters, exports results.
* **System (Streamlit Web App)** → loads, merges, analyzes, visualizes, and exports datasets.

**Main Use-Cases:**

* Upload/Load Datasets (movies, ratings, users)
* Apply Filters (genre, user, rating range)
* View Dataset Preview
* Generate Pivot Tables (avg rating per movie/genre/user, rating counts)
* Visualize Insights (bar charts, pie charts, line charts)
* Export Results (CSV files, cleaned dataset)  
    
  



## ****2. Activity Diagram****

Represents the **workflow of your system** (step-by-step data flow).

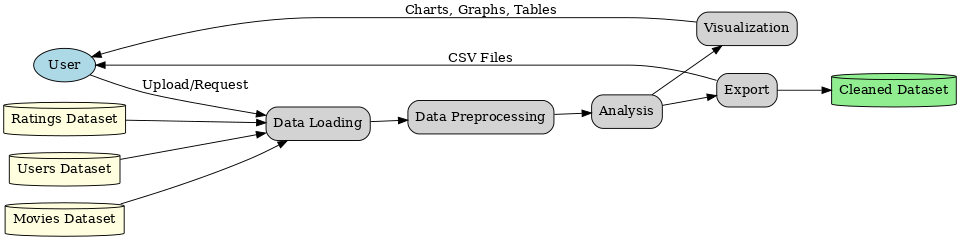
**Flow:**

1. Start
2. Load Datasets (movies, ratings, users)
3. Merge Datasets (ratings + movies + users)
4. Data Cleaning (check missing values, duplicates,   
   derive RatingCategory)
5. Apply Filters (genre, user, rating range)
6. Generate Pivot Tables
7. Visualize Results (charts & graphs)
8. Export Options (CSV, cleaned dataset)
9. End

## ****3. Data Flow Diagram****

Shows how **data moves** between components.

**Entities:**

* **User** → Provides input (filters, dataset selection).
* **System Modules:**
  + Data Loading Module (reads CSVs)
  + Data Preprocessing Module (clean, merge, transform)
  + Analysis Module (pivot tables, grouping)
  + Visualization Module (charts & graphs)
  + Export Module (CSV output).
* **Datastores:**
  + Movies Dataset
  + Ratings Dataset
  + Users Dataset
  + Cleaned & Processed Dataset

## 🎨 Front-End (Interface) Design

### Overview

The front-end of this project is implemented using **Streamlit**, which provides an interactive web-based interface. It allows users to:

* Upload and preview datasets (movies.csv, ratings.csv, users.csv).
* Filter movies by genre, user, and rating range.
* View pivot tables and visual insights (average ratings, most-rated movies, etc.).
* Export processed results into CSV files.
* Visualize insights using bar charts, pie charts, and line charts.

### Key UI Components

1. **Title**
   * Project title: “🎬 Movie Ratings Dataset Exploration with Pivot Tables”
2. **Dataset Preview Section**
   * Expandable panels (st.expander) to view:
     + Movies dataset
     + Ratings dataset
     + Users dataset
3. **Sidebar Filters**
   * Genre selection (selectbox)
   * User selection (selectbox)
   * Rating range slider (slider)
4. **Interactive Insights Section**
   * Movies with average rating ≥ 4.0
   * Top 5 movies by ratings (table + bar chart)
   * Active users (who rated more than 5 movies)
   * Highest & lowest rated movies
5. **Pivot Tables Section**
   * Average rating per movie
   * Average rating per genre
   * Average rating per user
   * Count of ratings per movie
6. **Visual Insights Section**
   * Bar charts for top-rated movies
   * Pie chart for rating distribution (High/Medium/Low)
   * Bar chart for genre-wise averages
7. **Export Options**
   * Export individual pivot tables
   * Export filtered dataset
   * Export full cleaned dataset

### SETUP INSTUCTIONS

**📂 Project Structure**

movie\_ratings\_project/

│

├── analysis.py # Data analysis with Pandas

├── app.py # Streamlit interactive app

├── movies.csv # Sample dataset (movies)

├── ratings.csv # Sample dataset (ratings)

├── movie\_avg\_ratings.csv # Exported pivot table (movies)

├── genre\_avg\_ratings.csv # Exported pivot table (genres)

├── user\_avg\_ratings.csv # Exported pivot table (users)

├── cleaned\_movie\_ratings.csv # Final cleaned + merged dataset

├── requirements.txt # Python dependencies

└── README.md # Project documentation

**SETTING UP START:**

**1️⃣ Clone or Download the Project**

Unzip the project folder or clone it from GitHub (if hosted).

**2️⃣ Setup Virtual Environment**

Open terminal inside the project folder and run:

python -m venv venv

venv\Scripts\activate # Windows

source venv/bin/activate # Mac/Linux

**3️⃣ Install Dependencies**

pip install -r requirements.txt

**4️⃣ Run Data Analysis (Optional)**

python analysis.py

**5️⃣ Run the Web App**

streamlit run app.py

**🛠️ Technologies Used**

* Python 3.12
* Pandas
* Streamlit
* Matplotlib & Seaborn

# Code

Click here to view full source code: [app\_code.txt](../../../gsss_sic/project/movie_ratings_project/app_code.txt)  
-ctrl+click to follow the code

# ****Explanation of the Code****

The project is implemented in **Python** using the **Streamlit framework** for building an interactive web application. It integrates **Pandas** for data handling, **Seaborn/Matplotlib** for visualization.

Below is the structured explanation with code snippets.

## ****1. Importing Libraries****

import pandas as pd

import streamlit as st

import matplotlib.pyplot as plt

import seaborn as sns

* **pandas** → Data manipulation and analysis.
* **streamlit** → Web application framework for interactive dashboards.
* **matplotlib.pyplot** & **seaborn** → For data visualization and prettier charts.

## ****2. Loading Data****

movies = pd.read\_csv("movies.csv")

ratings = pd.read\_csv("ratings.csv")

users = pd.read\_csv("users.csv")

* movies.csv: Movie details (titles, genres).
* ratings.csv: Ratings provided by users.
* users.csv: User demographic details.

Data is loaded into DataFrames for further processing.

## ****3. Merging Datasets****

ratings\_movies = pd.merge(ratings, movies, on="MovieID", how="inner")

merged = pd.merge(ratings\_movies, users, on="UserID", how="inner")

* **Step 1**: Merge ratings with movies on MovieID.
* **Step 2**: Merge the result with users on UserID.
* Final merged DataFrame contains user, movie, rating, and genre details in one place.

## ****4. Creating Derived Columns****

merged["RatingCategory"] = merged["Rating"].apply(

lambda x: "High" if x >= 4 else ("Medium" if x >= 3 else "Low")

)

* Ratings are categorized as:
  + **High** (≥ 4)
  + **Medium** (3 ≤ rating < 4)
  + **Low** (< 3)
* This helps in analyzing overall distribution.

## ****5. Pivot Tables (Pre-Computed Insights)****

movie\_rating\_counts = (

merged.pivot\_table(index="Title", values="Rating", aggfunc="count")

.reset\_index()

.rename(columns={"Rating": "RatingCount"})

)

movie\_avg\_ratings = (

merged.pivot\_table(index="Title", values="Rating", aggfunc="mean")

.reset\_index()

.rename(columns={"Rating": "AvgRating"})

)

movie\_rating\_counts["IsPopular"] = movie\_rating\_counts["RatingCount"].apply(

lambda x: "Yes" if x > 10 else "No"

)

* **Count of Ratings per Movie** → How many times each movie was rated.
* **Average Rating per Movie** → Mean rating for each movie.
* **Popularity Flag** → Movies with more than 10 ratings are labeled as "Yes" (popular).

## ****6. Streamlit UI Setup****

st.title("🎬 Movie Ratings Dataset Exploration with Pivot Tables")

st.header("Dataset Preview")

with st.expander("Show Movies Dataset"):

st.dataframe(movies)

with st.expander("Show Ratings Dataset"):

st.dataframe(ratings)

with st.expander("Show Users Dataset"):

st.dataframe(users)

* The app displays a **title** and expandable dataset previews.
* Users can quickly explore raw data without exporting it.

## ****7. Filters (Sidebar Controls)****

st.sidebar.header("🔎 Filters")

genres = movies["Genre"].unique().tolist()

selected\_genre = st.sidebar.selectbox("Select Genre", ["All"] + genres)

users\_list = merged["UserID"].unique().tolist()

selected\_user = st.sidebar.selectbox("Select User", ["All"] + users\_list)

min\_rating, max\_rating = st.sidebar.slider(

"Select Rating Range", 1.0, 5.0, (1.0, 5.0)

)

* **Genre filter** → Select specific genres or all.
* **User filter** → Focus on ratings by a specific user.
* **Rating range filter** → Restrict ratings between chosen values.

👉 These filters update the dataset dynamically.

## ****8. Filtered Dataset****

filtered\_data = merged.copy()

if selected\_genre != "All":

filtered\_data = filtered\_data[filtered\_data["Genre"] == selected\_genre]

if selected\_user != "All":

filtered\_data = filtered\_data[filtered\_data["UserID"] == selected\_user]

filtered\_data = filtered\_data[

(filtered\_data["Rating"] >= min\_rating) & (filtered\_data["Rating"] <= max\_rating)

]

st.header("🎯 Filtered Dataset")

st.dataframe(filtered\_data.head(20))

* Creates a copy of merged dataset.
* Applies user-selected filters.
* Displays first 20 rows in a table format.

## ****9. Interactive Insights****

### a) Movies with Avg Rating ≥ 4.0

highly\_rated = filtered\_data.groupby("Title")["Rating"].mean().reset\_index()

highly\_rated = highly\_rated[highly\_rated["Rating"] >= 4.0]

st.dataframe(highly\_rated)

Shows movies with strong user approval.

### b) Top 5 Movies by Ratings (Pivot + Chart)

top5 = (

merged.groupby("Title")["Rating"]

.count()

.sort\_values(ascending=False)

.head(5)

.reset\_index()

.rename(columns={"Rating": "RatingCount"})

)

* Pivot table for **Top 5 most rated movies**.
* Displayed in **table** and **bar chart** with annotation labels.

### c) Highest & Lowest Rated Movies

avg\_ratings = filtered\_data.groupby("Title")["Rating"].mean().reset\_index()

highest = avg\_ratings.sort\_values(by="Rating", ascending=False).head(1)

lowest = avg\_ratings.sort\_values(by="Rating", ascending=True).head(1)

Displays extremes in ratings.

## ****10. Pivot Tables & Graphs****

* **Average Rating per Movie**

pivot\_movie\_avg = merged.pivot\_table(index="Title", values="Rating", aggfunc="mean").reset\_index()

st.bar\_chart(pivot\_movie\_avg.set\_index("Title"))

* **Average Rating per Genre**

pivot\_genre\_avg = merged.pivot\_table(index="Genre", values="Rating", aggfunc="mean").reset\_index()

sns.barplot(x="Rating", y="Genre", data=pivot\_genre\_avg, palette="viridis")

* **Average Rating per User**

pivot\_user\_avg = merged.pivot\_table(index="UserID", values="Rating", aggfunc="mean").reset\_index()

st.line\_chart(pivot\_user\_avg.set\_index("UserID"))

* **Count of Ratings per Movie**

pivot\_movie\_count = (

merged.pivot\_table(index="Title", values="Rating", aggfunc="count")

.reset\_index()

.rename(columns={"Rating": "RatingCount"})

)

sns.barplot(x="RatingCount", y="Title", data=pivot\_movie\_count.sort\_values(by="RatingCount", ascending=False).head(10))

## ****11. Visual Insights (Pie Chart)****

rating\_dist = merged["RatingCategory"].value\_counts()

colors = sns.color\_palette("crest", len(rating\_dist))

ax1.pie(

rating\_dist,

labels=rating\_dist.index,

autopct="%1.1f%%",

startangle=90,

colors=colors,

wedgeprops={"edgecolor": "white"}

)

* Shows distribution of High, Medium, Low ratings in a clean pie chart.

## ****12. Export Options****

The app provides CSV exports for every major analysis:

top5.to\_csv("top5\_movies\_by\_ratings.csv", index=False)

pivot\_movie\_avg.to\_csv("movie\_avg\_ratings.csv", index=False)

pivot\_genre\_avg.to\_csv("genre\_avg\_ratings.csv", index=False)

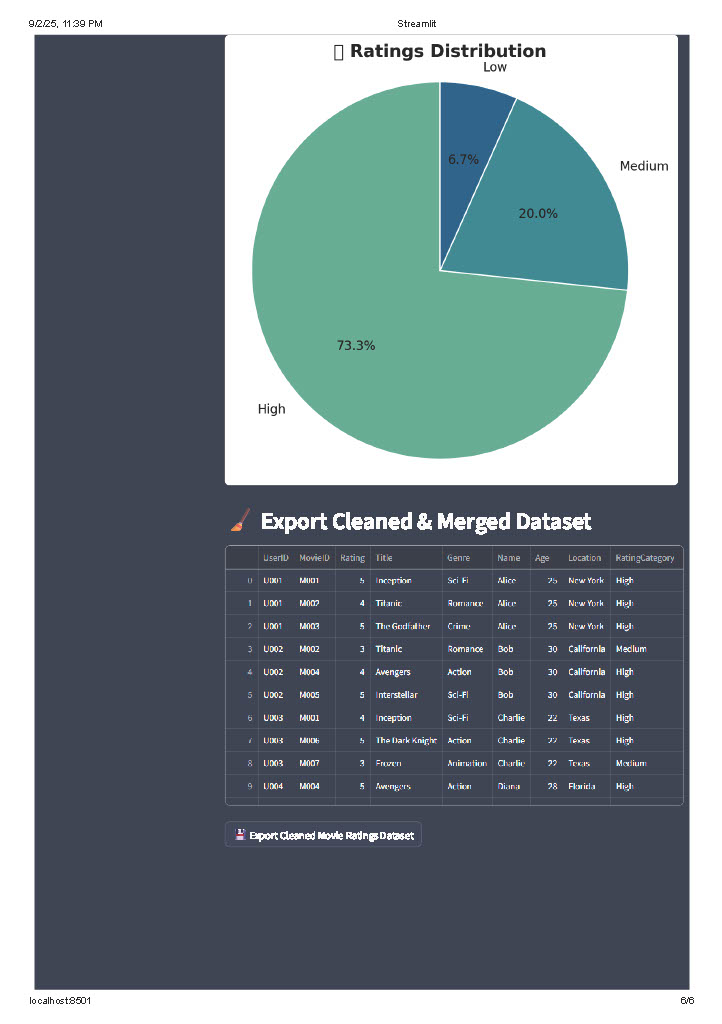
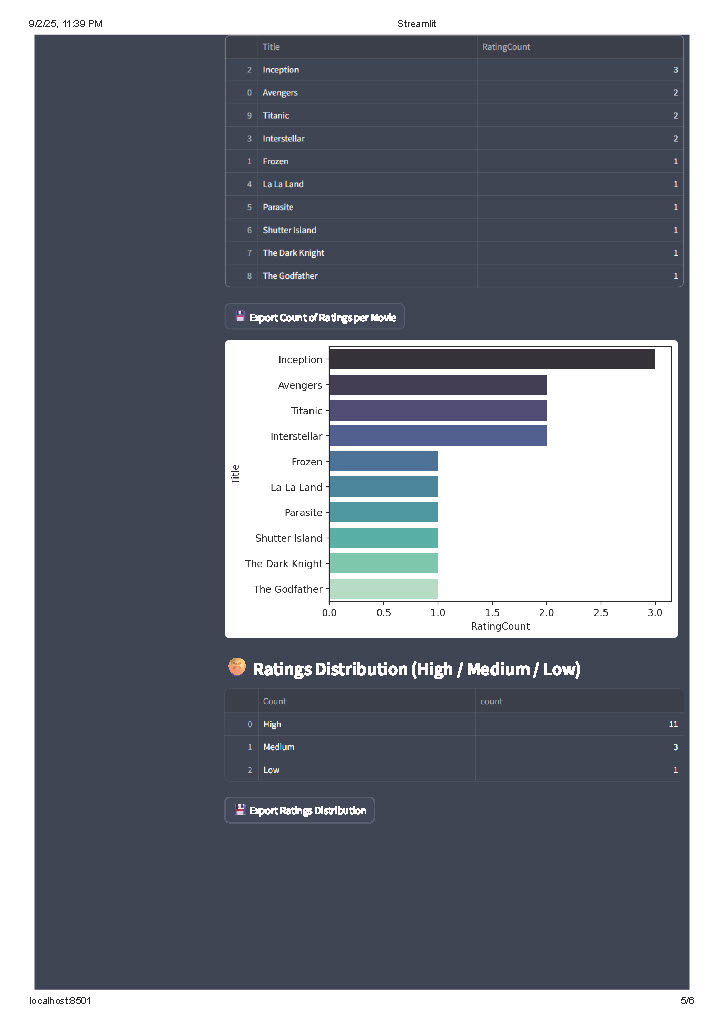
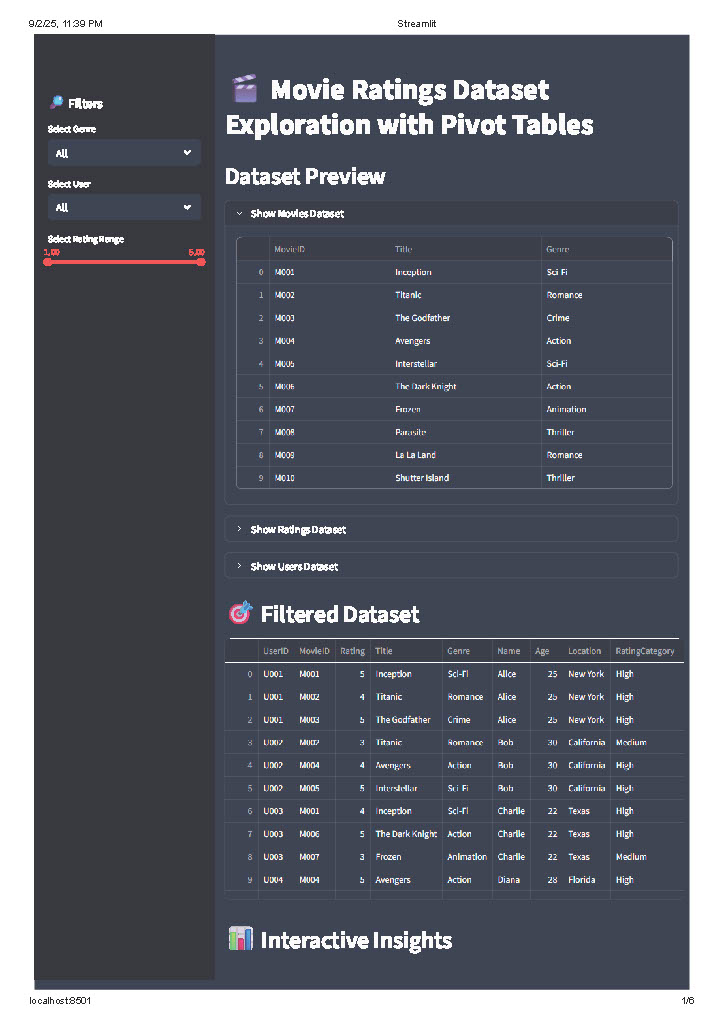
pivot\_user\_avg.to\_csv("user\_avg\_ratings.csv", index=False)

pivot\_movie\_count.to\_csv("movie\_rating\_counts.csv", index=False)

merged.to\_csv("cleaned\_movie\_ratings.csv", index=False)

This ensures **reusability** of results in Excel

# Screenshots of the Output

Screenshots of the dashboard, graphs, and exported files will be attached here.  
  


### Closure (Conclusion & Bibliography)

**Conclusion:**  
The project successfully demonstrates the use of **Pandas and Streamlit** to perform dataset exploration interactively. By using pivot tables, grouping, and aggregation, users can identify key insights like popular movies, top-rated genres, and user activity. The Streamlit interface makes it user-friendly and professional.

**Bibliography:**

* Python Pandas Documentation
* Streamlit Documentation
* Matplotlib & Seaborn Libraries
* Dataset inspired by MovieLens

Dataset: Sample dataset prepared for analysis.  
Tools: VS Code, Streamlit, Draw.io (for UML diagrams).