Netflix Movie Recommendations

Netflix offers thousands of movies and TV shows, which can make it hard for users to decide what to watch. To solve this problem, this project focuses on creating a movie recommendation system that suggests movies based on user preferences.

Using the Netflix Movies & TV Shows dataset, the project explores data such as genres, ratings, and release years to understand viewing trends. Then, techniques like TF-IDF (Term Frequency–Inverse Document Frequency) and cosine similarity are used to recommend movies that are similar to what a user has already watched or liked.

Objective or Problem statement

The goal of this project is to develop an intelligent Netflix movie recommendation system that helps users discover content aligned with their viewing preferences. By analyzing and processing the Netflix Movies & TV Shows dataset, the project aims to uncover patterns in user behavior, identify popular genres, and generate personalized recommendations.

The system will address the challenge of information overload on streaming platforms, where users often struggle to find relevant content among thousands of titles. Through data cleaning, exploratory analysis, and machine learning techniques such as TF-IDF vectorization and cosine similarity, the project seeks to improve user satisfaction by suggesting movies and TV shows that match their interests and historical preferences.

Tools and Technologies Used

- **Python** – for data analysis, preprocessing, and building the recommendation system
- **Pandas** – for data manipulation and cleaning
- **NumPy** – for numerical computations
- **Matplotlib** and **Seaborn** – for data visualization and exploratory analysis
- **Scikit-learn** – for implementing TF-IDF and cosine similarity
- **VS Code** – as the coding and analysis environment
- **CSV Dataset (Netflix Movies & TV Shows)** – as the primary data source

Dataset Used

The dataset used for this project is the Netflix Movies & TV Shows dataset sourced from Kaggle. It contains information about over 8,800 records of movies and TV shows available on Netflix, including details such as title, director, cast, country, release year, rating, and genre.
This dataset provides both categorical and textual data**,** which is ideal for building a content-based recommendation system using techniques like TF-IDF and cosine similarity.

Methodology

- **Data Collection**

✓ The Netflix Movies & TV Shows dataset was imported from Kaggle. It includes details such as title, genre, cast, director, and release year for analysis.

- **Data Cleaning**

✓ Duplicate records and missing values were handled. Text fields like titles and genres were standardized to maintain consistency.

- **Data Preprocessing**

✓ Combined multiple text features (title, cast, genre) into a single field to prepare the dataset for feature extraction.

- **Feature Extraction**

✓ Used the TF-IDF technique to convert text data into numerical vectors, enabling similarity calculations between movies.

- **Model Building**

✓ Implemented cosine similarity to find and recommend movies or shows similar to the one selected by the user.

- **Visualization**

  - ✓ Generated charts and trend analysis using Python libraries (Matplotlib, Seaborn) to better understand genre distribution and ratings.

- **Output / Deployment**

  - ✓ The final system provides personalized Netflix movie recommendations based on content similarity.

Features Implemented/ Key Visuals

- ❖ **Movie Recommendation System** – Built a content-based recommendation model using TF-IDF and cosine similarity to suggest movies similar to the user’s choice.
- ❖ **Genre Analysis** – Visualized the most popular genres and their frequency using bar charts and count plots.
- ❖ **Rating and Year Trends** – Analyzed how movie releases and user ratings have changed over different years.
- ❖ **Data Insights Dashboard** – Displayed visual insights such as top genres, release year distribution, and type-wise (Movie/TV Show) count using Python visualization libraries.
- ❖ **User Personalization** – Generated customized movie suggestions based on user viewing patterns and preferences.

Results/ Key Insights

- ✓ The model successfully recommends movies and TV shows similar to the user’s selected title based on genre, cast, and description.
- ✓ Analysis revealed that Drama, Comedy, and Action are the most common genres on Netflix.
- ✓ Most content was released between 2015–2021, showing a rapid growth in Netflix’s content library during this period.
- ✓ The majority of Netflix content comes from the United States, India, and the United Kingdom, highlighting regional dominance.
- ✓ The recommendation system improved user experience by providing accurate and relevant movie suggestions using TF-IDF and cosine similarity.

Challenges Faced

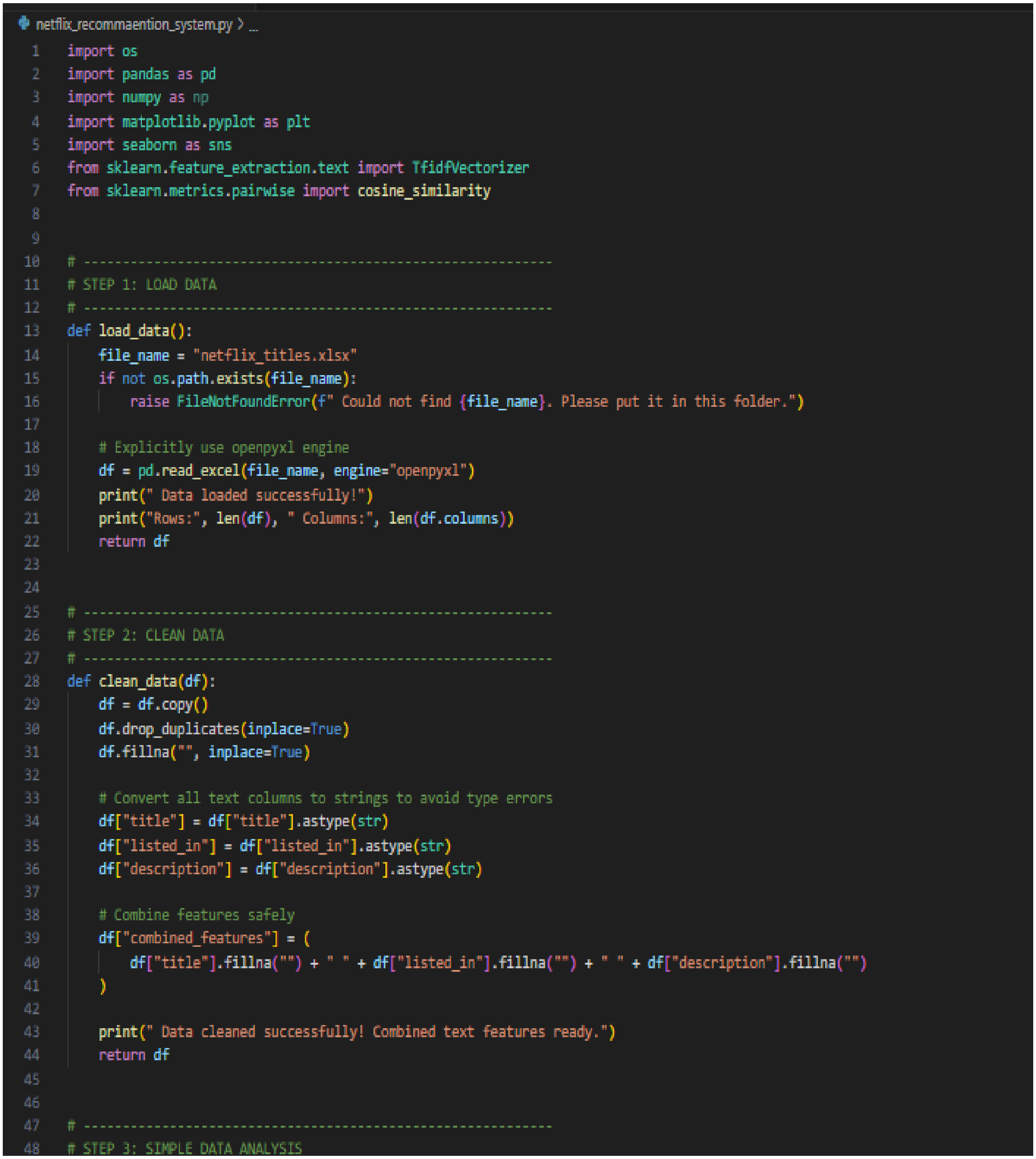
- ✓ Faced missing and duplicate data in the dataset, which was solved through data cleaning and preprocessing using Python (Pandas).
- ✓ Encountered text formatting issues in titles and genres, handled by converting text to lowercase and removing extra spaces.
- ✓ Dealt with high dimensionality during TF-IDF vectorization, resolved by limiting the number of features and optimizing cosine similarity calculations.
- ✓ Difficulty in balancing recommendations across genres was improved by combining multiple text attributes like cast, director, and genre for better accuracy.

Conclusion

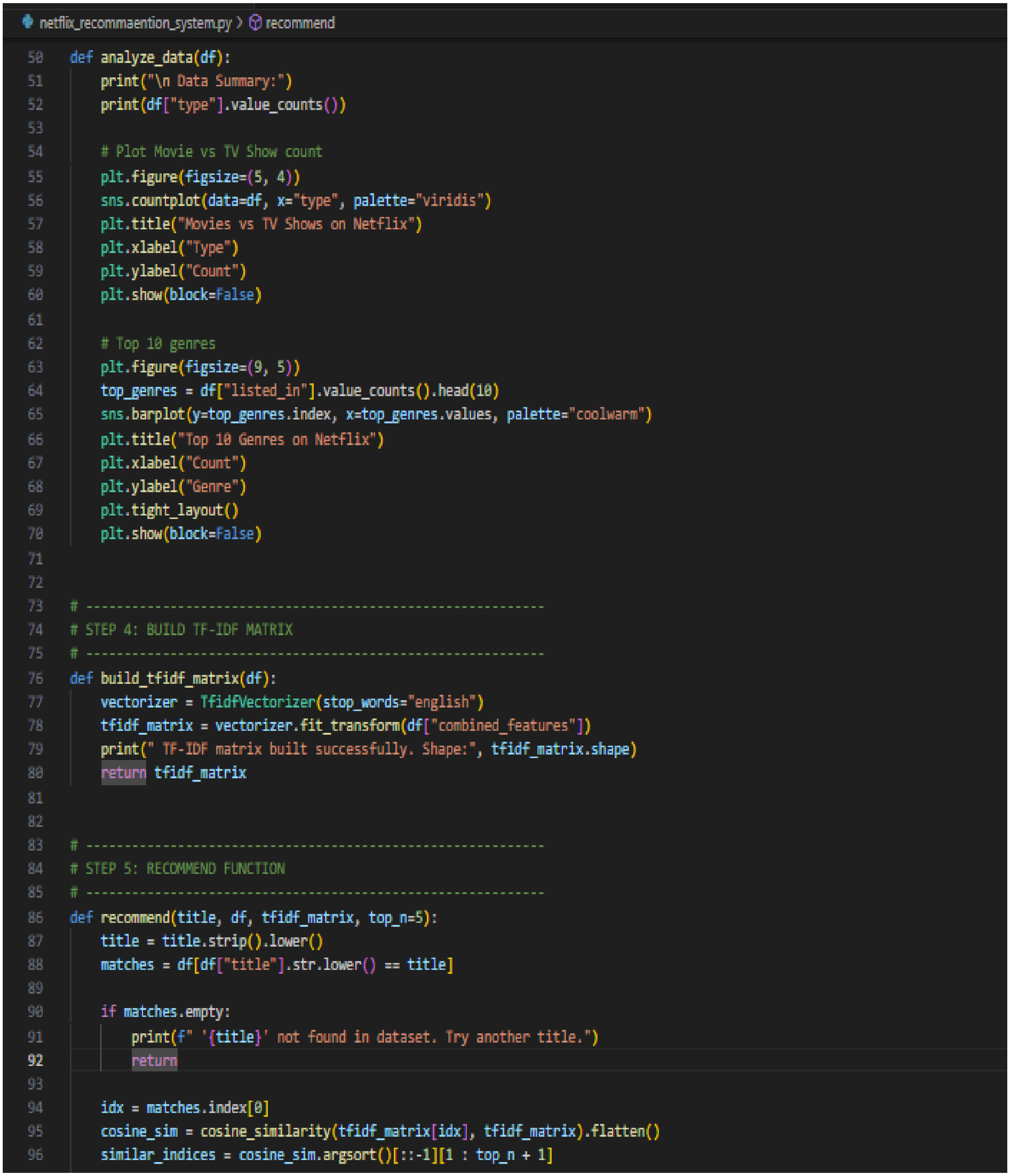
This project provided valuable insights into how data-driven techniques can be applied to personalize user experiences on streaming platforms. By cleaning, exploring, and analyzing the Netflix dataset, I was able to understand viewer preferences, popular genres, and trends across different years. Implementing TF-IDF and cosine similarity techniques enabled the creation of a functional movie recommendation system that suggests content aligned with user interests.

This project helped improve my skills in data cleaning**,** EDA**,** and storytelling using data. It also strengthened my understanding of machine learning concepts like similarity measures and recommendation algorithms. Overall, this project was an excellent opportunity to combine analytical thinking with practical implementation to solve a real-world problem effectively.

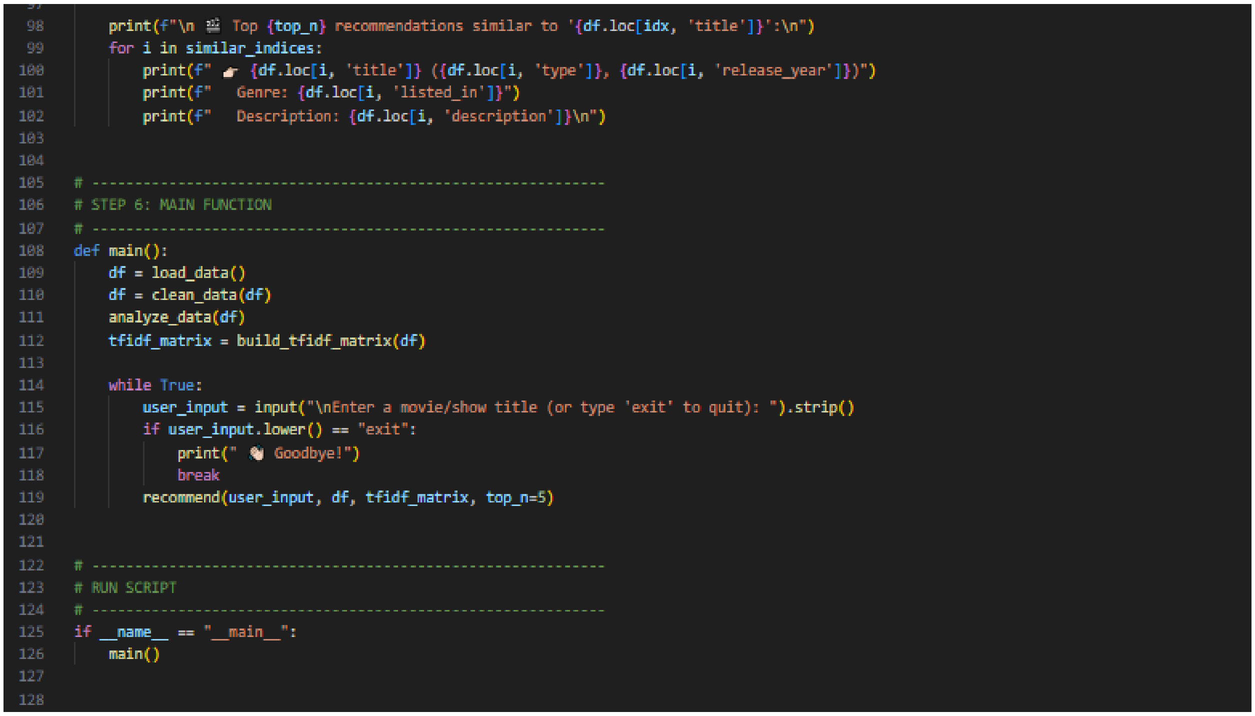
Screenshots



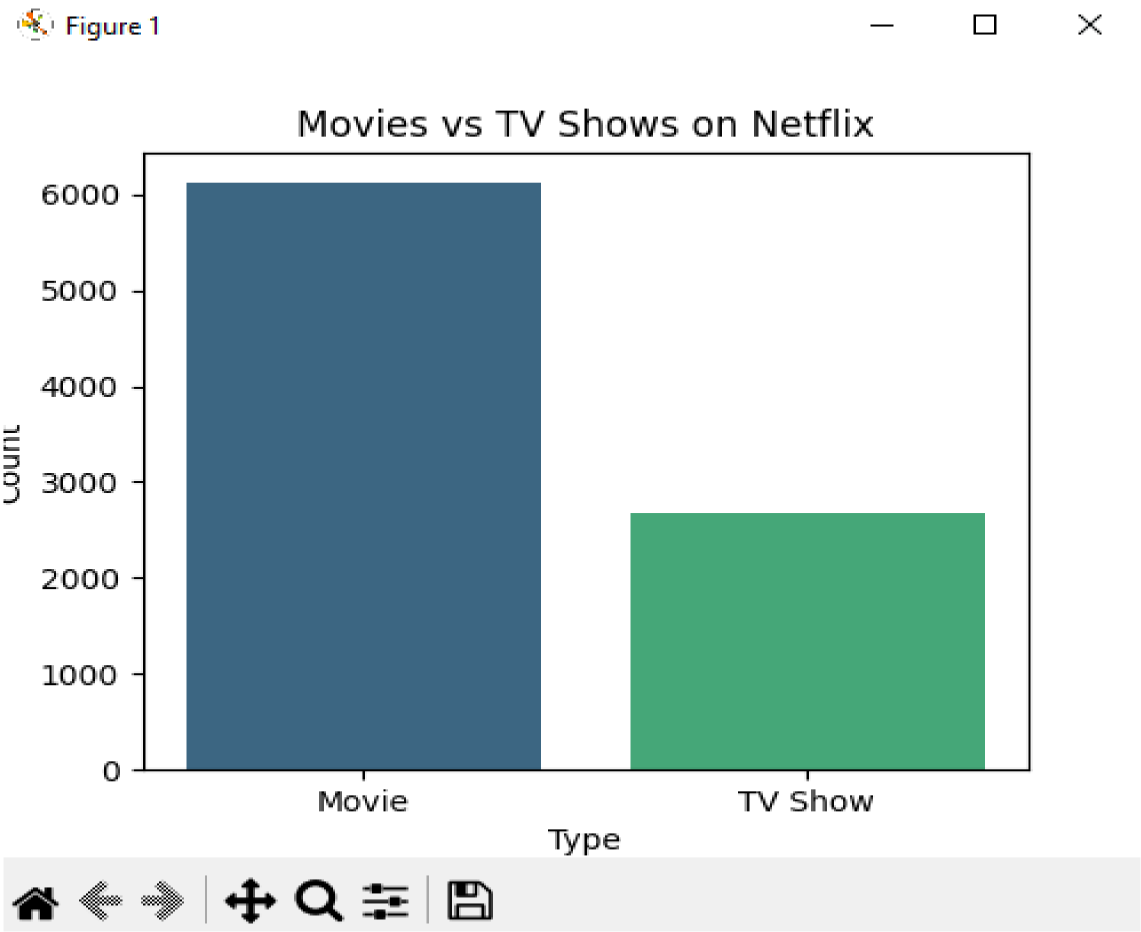
```python
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity


# ------------------------------------------------------------
# STEP 1: LOAD DATA
# ------------------------------------------------------------
def load_data():
    file_name = "netflix_titles.xlsx"
    if not os.path.exists(file_name):
        raise FileNotFoundError(f" Could not find {file_name}. Please put it in this folder.")

    # Explicitly use openpyxl engine
    df = pd.read_excel(file_name, engine="openpyxl")
    print(" Data loaded successfully!")
    print("Rows:", len(df), " Columns:", len(df.columns))
    return df


# ------------------------------------------------------------
# STEP 2: CLEAN DATA
# ------------------------------------------------------------
def clean_data(df):
    df = df.copy()
    df.drop_duplicates(inplace=True)
    df.fillna("", inplace=True)

    # Convert all text columns to strings to avoid type errors
    df["title"] = df["title"].astype(str)
    df["listed_in"] = df["listed_in"].astype(str)
    df["description"] = df["description"].astype(str)

    # Combine features safely
    df["combined_features"] = (
        df["title"].fillna("") + " " + df["listed_in"].fillna("") + " " + df["description"].fillna("")
    )

    print(" Data cleaned successfully! Combined text features ready.")
    return df


# ------------------------------------------------------------
# STEP 3: SIMPLE DATA ANALYSIS
```

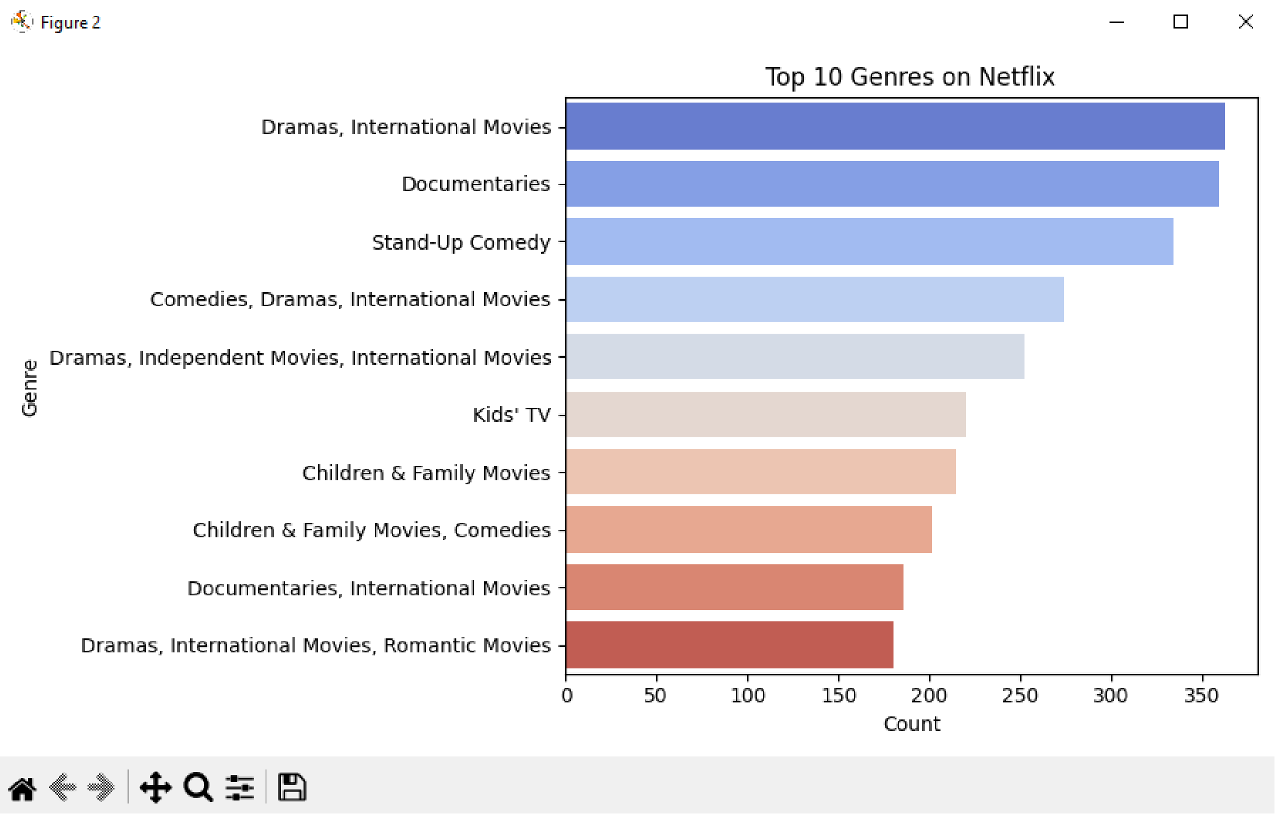


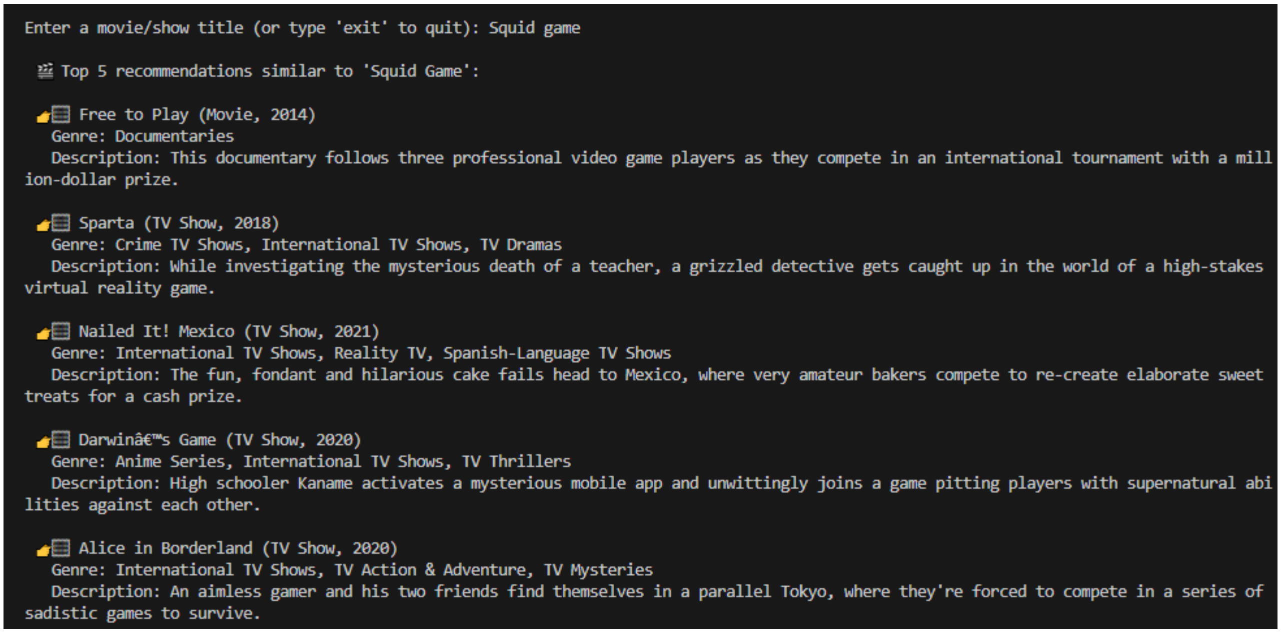
```python
def analyze_data(df):
    print("\n Data Summary:")
    print(df["type"].value_counts())

    # Plot Movie vs TV Show count
    plt.figure(figsize=(5, 4))
    sns.countplot(data=df, x="type", palette="viridis")
    plt.title("Movies vs TV Shows on Netflix")
    plt.xlabel("Type")
    plt.ylabel("Count")
    plt.show(block=False)

    # Top 10 genres
    plt.figure(figsize=(9, 5))
    top_genres = df["listed_in"].value_counts().head(10)
    sns.barplot(y=top_genres.index, x=top_genres.values, palette="coolwarm")
    plt.title("Top 10 Genres on Netflix")
    plt.xlabel("Count")
    plt.ylabel("Genre")
    plt.tight_layout()
    plt.show(block=False)


# -------------------------------------------------------------
# STEP 4: BUILD TF-IDF MATRIX
# -------------------------------------------------------------
def build_tfidf_matrix(df):
    vectorizer = TfidfVectorizer(stop_words="english")
    tfidf_matrix = vectorizer.fit_transform(df["combined_features"])
    print(" TF-IDF matrix built successfully. Shape:", tfidf_matrix.shape)
    return tfidf_matrix


# -------------------------------------------------------------
# STEP 5: RECOMMEND FUNCTION
# -------------------------------------------------------------
def recommend(title, df, tfidf_matrix, top_n=5):
    title = title.strip().lower()
    matches = df[df["title"].str.lower() == title]

    if matches.empty:
        print(f" '{title}' not found in dataset. Try another title.")
        return

    idx = matches.index[0]
    cosine_sim = cosine_similarity(tfidf_matrix[idx], tfidf_matrix).flatten()
    similar_indices = cosine_sim.argsort()[::-1][1 : top_n + 1]
```



```python
    print(f"\n 🎬 Top {top_n} recommendations similar to '{df.loc[idx, 'title']}':\n")
    for i in similar_indices:
        print(f" 👉 {df.loc[i, 'title']} ({df.loc[i, 'type']}, {df.loc[i, 'release_year']})")
        print(f"   Genre: {df.loc[i, 'listed_in']}")
        print(f"   Description: {df.loc[i, 'description']}\n")


# ------------------------------------------------------------
# STEP 6: MAIN FUNCTION
# ------------------------------------------------------------
def main():
    df = load_data()
    df = clean_data(df)
    analyze_data(df)
    tfidf_matrix = build_tfidf_matrix(df)

    while True:
        user_input = input("\nEnter a movie/show title (or type 'exit' to quit): ").strip()
        if user_input.lower() == "exit":
            print(" 👋 Goodbye!")
            break
        recommend(user_input, df, tfidf_matrix, top_n=5)


# ------------------------------------------------------------
# RUN SCRIPT
# ------------------------------------------------------------
if __name__ == "__main__":
    main()

```







```
Enter a movie/show title (or type 'exit' to quit): Squid game

 🎬 Top 5 recommendations similar to 'Squid Game':

 👉🎞 Free to Play (Movie, 2014)
   Genre: Documentaries
   Description: This documentary follows three professional video game players as they compete in an international tournament with a mill
ion-dollar prize.

 👉🎞 Sparta (TV Show, 2018)
   Genre: Crime TV Shows, International TV Shows, TV Dramas
   Description: While investigating the mysterious death of a teacher, a grizzled detective gets caught up in the world of a high-stakes
virtual reality game.

 👉🎞 Nailed It! Mexico (TV Show, 2021)
   Genre: International TV Shows, Reality TV, Spanish-Language TV Shows
   Description: The fun, fondant and hilarious cake fails head to Mexico, where very amateur bakers compete to re-create elaborate sweet
treats for a cash prize.

 👉🎞 Darwinâ€™s Game (TV Show, 2020)
   Genre: Anime Series, International TV Shows, TV Thrillers
   Description: High schooler Kaname activates a mysterious mobile app and unwittingly joins a game pitting players with supernatural abi
lities against each other.

 👉🎞 Alice in Borderland (TV Show, 2020)
   Genre: International TV Shows, TV Action & Adventure, TV Mysteries
   Description: An aimless gamer and his two friends find themselves in a parallel Tokyo, where they're forced to compete in a series of
sadistic games to survive.
```

Link to GitHub

GitHub: https://github.com/Yashu-teach/Netflix-Movie-Recommendations-Project