TMDB Movie Data Analysis using Pandas and APIs

Project Overview

This project challenges you to build a **movie data analysis pipeline** using Python and Pandas. You will fetch movie-related data from an API, clean and transform the dataset, and implement **key performance indicators (KPIs)**.

This is **not a group project**, meaning you will design the workflow, structure the analysis, and implement the required calculations independently.

Project Objectives

- **API Data Extraction**: Fetch movie data from a movie database API.
- **Data Cleaning & Transformation**: Process and structure the data for analysis.
- **Exploratory Data Analysis (EDA)**: Perform an initial exploration to understand trends.
- **Advanced Filtering & Ranking**: Identify the best and worst movies based on financial and popularity metrics.
- **Franchise & Director Analysis**: Assess how franchises and directors perform over time.
- **Visualization & Insights**: Present key findings using visualizations.

Project Steps

Step 1: Fetch Movie Data from API

- Choose a **movie database API** (e.g., TMDb).

  Fetch movies with ID: movie_id = [0, 299534, 19995, 140607, 299536, 597, 135397,

  420818, 24428, 168259, 99861, 284054, 12445,

  181808, 330457, 351286, 109445, 321612, 260513]

- Store the data as a Pandas **DataFrame**.

NB: Read the API documentation to understand the nature of the data and how it's organized.

Step 2: Data Cleaning and Preprocessing

Data Preparation & Cleaning

1. **Drop** irrelevant columns: ['adult', 'imdb_id', 'original_title', 'video', 'homepage'].
2. **Evaluate** JSON-like columns (['belongs_to_collection', 'genres', 'production_countries', 'production_companies', 'spoken_languages']).
3. **Extract** and clean key data points:
   - Collection name (belongs_to_collection)

- Genre names (genres → separate multiple genres with "|").
- Spoken languages (spoken_languages → separate with "|").
- Production countries (production_countries → separate with "|").
- Production companies (production_companies → separate with "|").

4. **Inspect** extracted columns using value_counts() to identify anomalies.

Handling Missing & Incorrect Data

5. **Convert** column datatypes:
    - 'budget', 'id', 'popularity' → Numeric (set invalid values to NaN).
    - 'release_date' → Convert to **datetime**.
    - etc
6. **Replace unrealistic values**:
    - Budget/Revenue/Runtime = **0** → Replace with NaN or infer from similar movies.
    - Convert 'budget' and 'revenue' **to million USD**.
    - Movies with **vote_count = 0** → Analyze their **vote_average** and adjust accordingly.
    - 'overview' and 'tagline' → Replace known placeholders (e.g., 'No Data') with NaN.
7. **Remove duplicates** and drop rows with unknown 'id' or 'title'.
8. **Keep** only rows where at least **10 columns have non-NaN values**.
9. **Filter** to include only 'Released' movies, then drop 'status'.

Reorder & Finalize DataFrame

10. **Reorder columns**:

['id', 'title', 'tagline', 'release_date', 'genres', 'belongs_to_collection',

'original_language', 'budget_musd', 'revenue_musd', 'production_companies',

'production_countries', 'vote_count', 'vote_average', 'popularity', 'runtime',

'overview', 'spoken_languages', 'poster_path', 'cast', 'cast_size', 'director', 'crew_size']

11. **Reset index**.

Step 3: KPI Implementation & Analysis

Identify the Best/Worst Performing Movies

1. **Filter and rank movies** based on:
   - **Highest Revenue**
   - **Highest Budget**
   - **Highest Profit** (Revenue - Budget)
   - **Lowest Profit** (Revenue - Budget)
   - **Highest ROI** (Revenue / Budget) *(only movies with Budget ≥ 10M)*
   - **Lowest ROI** *(only movies with Budget ≥ 10M)*
   - **Most Voted Movies**
   - **Highest Rated Movies** *(only movies with ≥10 votes)*
   - **Lowest Rated Movies** *(only movies with ≥10 votes)*
   - **Most Popular Movies**

Define a User-Defined Function (UDF) to streamline ranking operations.

Advanced Movie Filtering & Search Queries

2. **Filter the dataset for specific queries**:
   - **Search 1:** *Find the best-rated* **Science Fiction Action** *movies starring Bruce Willis (sorted by Rating - highest to lowest)*.
   - **Search 2:** *Find movies starring Uma Thurman, directed by Quentin Tarantino (sorted by runtime - shortest to longest)*.

Franchise vs. Standalone Movie Performance

3. **Compare movie franchises (belongs_to_collection) vs. standalone movies** in terms of:
   - **Mean Revenue**
   - **Median ROI**
   - **Mean Budget Raised**
   - **Mean Popularity**
   - **Mean Rating**

Most Successful Franchises & Directors

4. **Find the Most Successful Movie Franchises** based on:
    - **Total number of movies in franchise**
    - **Total & Mean Budget**
    - **Total & Mean Revenue**
    - **Mean Rating**
5. **Find the Most Successful Directors** based on:
    - **Total Number of Movies Directed**
    - **Total Revenue**
    - **Mean Rating**

Step 4: Data Visualization

Use **Pandas, Matplotlib** to visualize:

- **Revenue vs. Budget Trends**
- **ROI Distribution by Genre**
- **Popularity vs. Rating**
- **Yearly Trends in Box Office Performance**
- **Comparison of Franchise vs. Standalone Success**

Project Deliverables

- **Complete Workflow**: Jupyter Notebook or Python script covering data processing, analysis, and results.
- **Final Report**: Summary of key insights, methodology, and conclusions.
- **Efficient Code**: Clean, modular, and optimized for performance.
- **Git Best Practices**: Proper version control with clear commits and organized structure.