**IMDb 2024 Data Scraping and Visualization Project — Full Documentation**

**1. Project Overview**

This project automates extracting, cleaning, storing, and visually exploring IMDb’s 2024 feature movies. It involves:

* Web scraping dynamic content with Selenium
* Parallel HTTP requests for movie details (genres and duration)
* Data cleaning and transformation
* Storage in CSV, SQLite, and PostgreSQL databases
* An interactive Streamlit dashboard for filtering and visual analysis

**2. Script: data\_scraping\_and\_store.py**

**2.1 Imports & Setup**

* Imports Selenium and support modules for web scraping dynamic content from IMDb’s filtered movie search page.
* Uses BeautifulSoup for HTML parsing.
* Uses **concurrent.futures.ThreadPoolExecutor** for parallel HTTP requests to speed up detail data fetching.
* Uses SQLAlchemy for PostgreSQL connection and pandas for data manipulation.

python

CHROMEDRIVER\_PATH = r'...' *# Configure ChromeDriver path.*

IMDB\_SEARCH\_URL = '...' *# IMDb filtered URL for 2024 feature films.*

OUTPUT\_DIR = "IMDB\_2024\_by\_genre"

os.makedirs(OUTPUT\_DIR, exist\_ok=True)

**2.2 Helper Function: parse\_vote\_count**

* Converts IMDb vote string like **"1.2K"**, **"900"**, **"2M"** into integer vote counts.

**2.3 Function: scrape\_search\_results(limit=300)**

* Opens IMDb search results page with Selenium.
* Repeatedly clicks the “50 more” button to load more movies until either:
  + The specified **limit** number of movies is reached
  + Or no more movies available.
* Parses each movie’s title, detail URL, IMDb rating, and vote count from the loaded page source.
* Returns list of movie dicts.

**2.4 Function: fetch\_genres\_and\_duration\_from\_url(movie)**

* Given a movie dictionary (with URL), fetches the movie’s detail page using **requests**.
* Parses genres from chips in interests section (**data-testid="interests"**) using BeautifulSoup.
* Parses duration from a specific **<ul>** list inside an identified div (class **sc-f9ad6c98-0 bqDcCk**) as **"1h 51m"** string.
* Returns genres string and duration string.

**2.5 Function: parallel\_genre\_duration\_fetch(movies, max\_workers=10)**

* Fetches all movies’ genres and durations **in parallel** using a ThreadPoolExecutor for improved speed.
* Updates each movie dict with fetched **"Genre"** and **"Duration"** keys.

**2.6 Functions: clean\_title and duration\_to\_minutes**

* **clean\_title**: Removes leading numbering from titles (e.g., **"1. The Life of Chuck"** → **"The Life of Chuck"**).
* **duration\_to\_minutes**: Parses durations like **"1h 51m"** into total minutes (e.g., **111**).

**2.7 Function: save\_by\_major\_genre(movies)**

* Defines a fixed set of **major genres** like Action, Drama, Comedy, etc.
* For each movie (with potentially multiple genres), assigns it exclusively to the first matched major genre.
* Saves one CSV file per major genre folder with all movies assigned uniquely; ensures no duplicate movies in multiple genres.

**2.8 Function: save\_to\_postgres**

* Saves the full pandas DataFrame (built from the movies list) into a PostgreSQL database using SQLAlchemy and PostgreSQL driver.

**2.9 Main Flow: main()**

* Calls **scrape\_search\_results** with **limit**.
* Calls **parallel\_genre\_duration\_fetch** to add genres and durations.
* Cleans and normalizes data (titles, durations in minutes, fill missing values).
* Saves combined CSV, genre CSVs, SQLite DB (if enabled), and PostgreSQL DB.
* Prints logs on progress.

**3. Script: imdb\_streamlit\_app.py**

**3.1 Imports & Setup**

* Imports **streamlit**, **pandas**, **matplotlib**, **seaborn**, **plotly**, and other data viz libs.

**3.2 Data Loading and Expansion**

* Loads the cleaned CSV (**imdb\_2024\_all\_movies.csv**) with caching for performance.
* Expands movie genres by splitting multi-genre strings, creating a dataframe with one genre per row (**GenreList**).

**3.3 Sidebar Filters**

* Provides multi-select filters for:
  + Movie duration ranges (**<2 hr**, **2-3 hr**, **>3 hr**)
  + Minimum IMDb rating
  + Minimum vote count
  + Movie genres
* Applies these filters in combination to the main dataframe.

**3.4 Visualizations**

The app dynamically generates various charts/tables on the filtered data:

1. **Top 10 Movies by Rating and Votes:** Table of top movies by rating and vote count.
2. **Genre Distribution:** Bar chart counting movies per genre.
3. **Average Duration by Genre:** Horizontal bar chart of avg movie length by genre.
4. **Voting Trends by Genre:** Bar chart of avg voting counts per genre.
5. **Rating Distribution:** Histogram and boxplot of movie ratings.
6. **Genre-Based Rating Leaders:** Table showing top-rated movies per major genre.
7. **Most Popular Genres by Voting:** Pie chart of total votes per genre.
8. **Duration Extremes:** Tables showing shortest and longest movies.
9. **Ratings by Genre Heatmap:** Heatmap comparing avg genre ratings.
10. **Correlation Analysis:** Scatter plot of votes vs ratings with regression line.

**3.5 Filtered Data Display**

* Shows the filtered movie data interactively as a table for detailed inspection.

**4. How It All Works Together**

* **Data scraped and cleaned** in **data\_scraping\_and\_store.py**, producing enriched CSV and DB.
* **Streamlit app loads cleaned data** and lets user slice and dice via its dashboard.
* The dashboard and pipeline combined provide rich insights and interactivity over the IMDb 2024 dataset.

**5. Technologies Used**

* **Web Scraping:** Selenium, Requests, BeautifulSoup
* **Data Engineering:** Pandas, NumPy, SQLAlchemy
* **Databases:** PostgreSQL, SQLite
* **Visualization & Dashboard:** Streamlit, Plotly, Matplotlib, Seaborn
* **Python Environment:** Recommended Python 3.9+, ChromeDriver installed for Selenium

**6. Next Steps / Improvements**

* Resume scraping on failures
* Switch to async detail fetching for greater speed
* Add more visualizations (e.g., time series trends if date data added)
* Deploy Streamlit app on cloud for broader access
* Add user authentication and save personalized filters

**Summary**

This project delivers a full-stack data solution from raw IMDb scraping through clean storage and rich exploration with interactive visualizations.