**Project 1 (Module 2) Documentation**

**Data Source:** The Movie Database (TMDb) API

**Technologies Used:** Python, Pandas, NumPy, Matplotlib, Seaborn, Requests

**Date Completed:** October 24, 2025

**Executive Summary**

This project involved building a comprehensive movie data analysis pipeline that extracted, cleaned, analyzed, and visualized data from The Movie Database (TMDb) API. The analysis covered 18 movies across various franchises and genres, focusing on performance metrics, franchise comparisons, and director success patterns. Project was broken into section to effectively work on each of the sections

**Key Objectives Achieved:**

* Successfully fetched and processed movie data from TMDb API
* Implemented comprehensive data cleaning and transformation pipeline
* Calculated key performance indicators (KPIs) for movie success
* Analyzed franchise vs. standalone movie performance
* Identified top-performing franchises and directors
* Created visualizations to communicate findings

**Section 1: Data Acquisition & Preparation**

**1.1 API Integration**

**Objective:** Fetch movie data from TMDb API for specified movie IDs.

**Implementation:**

* Registered for TMDb API key and configured environment variables
* Implemented fetch\_movie() function to retrieve movie details
* Fetched data for 18 movies using the /movie/{movie\_id} endpoint
* Implemented rate limiting (0.25s delay) to respect API constraints
* Added error handling for failed requests

Movie IDs Fetched: [0, 299534, 19995, 140607, 299536, 597, 135397, 420818, 24428, 168259, 99861, 284054, 12445, 181808, 330457, 351286, 109445, 321612, 260513]

**Result:** Successfully fetched 18 movies with complete metadata

**1.2 Data Cleaning & Transformation**

**Steps Performed:**

**Step 1: Drop Irrelevant Columns**

Removed columns not needed for analysis:

* adult
* imdb\_id
* original\_title
* video
* homepage

**Step 2: Parse Nested JSON Columns**

Extracted and flattened complex nested data structures:

* **belongs\_to\_collection:** Extracted collection name
* **genres:** Extracted genre names, joined with ‘|’
* **spoken\_languages:** Extracted language names, joined with ‘|’
* **production\_countries:** Extracted country names, joined with ‘|’
* **production\_companies:** Extracted company names, joined with ‘|’

**Step 3: Data Type Conversions**

* Converted budget, revenue, popularity to numeric (invalid → NaN)
* Converted release\_date to datetime format
* Handled missing and invalid values appropriately

**Step 4: Handle Unrealistic Values**

* Replaced budget/revenue/runtime = 0 with NaN
* Created new columns: **budget\_musd** and **revenue\_musd** (converted to millions USD)
* Replaced placeholder text ("No Data", empty strings) in overview/tagline with NaN

**Step 5: Data Quality Checks**

* Removed duplicate rows based on movie ID
* Dropped rows with missing ID or title
* Kept only rows with ≥10 non-null columns
* Filtered for movies with status = "Released"
* Removed the Status Column after Filtering

**Step 6: Fetch Cast & Crew Data**

Made additional API calls to /movie/{movie\_id}/credits endpoint:

* Extracted cast names and cast size
* Extracted director names and crew size
* Added columns: cast, cast\_size, director, crew\_size

**Step 7: Reorder Columns & Reset Index**

Final column order:

['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']

**1.3 Final Dataset Summary**

**Clean Dataset Statistics:**

* Total Movies: 18
* Total Columns: 22
* Data Quality: All movies have complete core information
* Saved to: tmdb\_cleaned\_data\_with\_credits.csv

**Section 2: Exploratory Analysis & KPI Computation**

**2.1 Performance Ranking (Best/Worst Movies)**

**Objective:** Identify top and bottom performing movies across multiple metrics.

**Calculated Metrics:**

* **Profit (M USD):** Revenue - Budget
* **ROI (Return on Investment):** Revenue / Budget

**User-Defined Function (UDF)**

Created rank\_movies() function with parameters:

* **rank\_by:** Column to rank by
* **top\_n:** Number of movies to return
* **ascending:** Sort order
* **filter\_condition:** Optional filters (e.g., budget ≥ 10M)
* **display\_columns:** Columns to show in results

**Rankings Performed:**

1. Top 10 Highest Revenue Movies
2. Top 10 Highest Budget Movies
3. Top 10 Highest Profit Movies
4. Top 10 Lowest Profit Movies (Biggest Losses)
5. Top 10 Highest ROI Movies (Budget ≥ $10M)
6. Top 10 Lowest ROI Movies (Budget ≥ $10M)
7. Top 10 Most Voted Movies
8. Top 10 Highest Rated Movies (Vote Count ≥ 10)
9. Top 10 Lowest Rated Movies (Vote Count ≥ 10)
10. Top 10 Most Popular Movies

**2.2 Advanced Filtering & Search Queries**

**Search Query 1:** Best-rated Science Fiction Action movies starring Bruce Willis

* Filter: Cast contains "Bruce Willis"
* Filter: Genres contain both "Science Fiction" AND "Action"
* Sort: By rating (highest to lowest)

**Search Query 2:** Movies starring Uma Thurman, directed by Quentin Tarantino

* Filter: Cast contains "Uma Thurman"
* Filter: Director is "Quentin Tarantino"
* Sort: By runtime (shortest to longest)

**Note:** Due to the limited dataset size (18 movies), these specific search criteria did not return results. The search functions were implemented and tested successfully.

**2.3 Franchise vs. Standalone Comparison**

**Objective:** Compare performance of franchise movies vs. standalone movies.

**Comparison Metrics:**

| **Metric** | **Description** |
| --- | --- |
| Number of Movies | Count of movies in each category |
| Mean Revenue | Average revenue in millions USD |
| Median ROI | Median return on investment |
| Mean Budget | Average budget in millions USD |
| Mean Popularity | Average popularity score |
| Mean Rating | Average vote average (out of 10) |

**Key Insights:**

* Franchise movies typically generate higher revenue than standalone films
* Franchise movies receive higher budgets on average
* ROI comparison reveals efficiency differences between categories
* Popularity and ratings vary between franchise and standalone movies

**2.4 Top Franchises & Directors**

**Franchise Analysis**

**Metrics Calculated:**

* Number of movies in each franchise
* Total budget per franchise
* Mean budget per movie
* Total revenue per franchise
* Mean revenue per movie
* Mean rating per franchise

**Rankings Created:**

1. Top franchises by total revenue
2. Top franchises by number of movies
3. Top franchises by total budget
4. Top franchises by mean budget per movie
5. Top franchises by mean revenue per movie
6. Top franchises by mean rating

**Director Analysis**

**Metrics Calculated:**

* Number of movies directed
* Total revenue generated
* Mean revenue per movie
* Mean rating across all movies

**Rankings Created:**

1. Top directors by total revenue
2. Top directors by number of movies
3. Top directors by mean revenue per movie
4. Top directors by mean rating

**Note:** Multiple directors per movie were handled by splitting director strings and creating separate records for each director.

**Section 3: Visualization & Reporting**

**3.1 Visualizations Created**

**The following visualizations were implemented using Matplotlib and Seaborn:**

1. **Revenue vs. Budget Scatter Plot**

* Shows relationship between movie budgets and revenues
* Includes trend line for correlation analysis
* Color-coded by franchise/standalone movies

1. **ROI Distribution by Top 10 Genres**

* Box plot ROI spread across genres
* Identifies which genres provide best return on investment

1. **Popularity vs. Rating Scatter Plot**

* Analyzes relationship between popularity and average ratings
* Identifies movies that are popular but low-rated or vice versa

1. **Yearly Box Office Trends**

* Line and bar charts showing total revenue per release year and other presentations such number of movies by Year , Budget and Revenue Trends over the Years
* Reveals temporal trends in box office performance

1. **Franchise vs. Standalone Comparison Charts**

* Bar charts comparing mean revenue, rating, budget, etc.
* Visual representation of performance differences

**3.2 Key Insights & Findings**

**Discoveries:**

* **Revenue Patterns:** Identified highest-grossing movies and factors contributing to box office success
* **Budget Efficiency:** Analyzed which movies achieved best ROI relative to their budgets
* **Franchise Power:** Demonstrated performance differences between franchise and standalone films
* **Director Impact:** Revealed which directors consistently deliver successful movies
* **Genre Trends:** Identified which genre combinations perform best financially
* **Rating vs. Popularity:** Explored the relationship between critical reception and audience engagement

**Methodology**

**4.1 Data Collection**

* **Source:** The Movie Database (TMDb) API
* **Endpoints Used:**
  + /movie/{movie\_id} - Main movie details
  + /movie/{movie\_id}/credits - Cast and crew information
* **Sample Size:** 18 movies
* **Authentication:** API key stored in environment variables

**4.2 Data Processing Pipeline**

1. **Extraction:** API calls with error handling and rate limiting
2. **Transformation:** JSON parsing, data type conversions, feature engineering
3. **Cleaning:** Missing value handling, duplicate removal, data validation
4. **Enrichment:** Calculated fields (profit, ROI)
5. **Analysis:** Aggregations, rankings, comparisons
6. **Visualization:** Charts and graphs for insights communication

**4.3 Tools & Technologies**

| **Tool/Library** | **Purpose** |
| --- | --- |
| Python 3.x | Primary programming language |
| Pandas | Data manipulation and analysis |
| NumPy | Numerical computations |
| Requests | API calls and HTTP requests |
| Matplotlib | Data visualization |
| Seaborn | Statistical data visualization |
| python-dotenv | Environment variable management |
| Jupyter Notebook | Interactive development environment |

**Conclusion**

This project successfully demonstrated a complete data analysis pipeline from data acquisition through insight generation. Key accomplishments include:

* Built robust API integration with proper error handling
* Implemented comprehensive data cleaning pipeline
* Created reusable User-Defined Functions for analysis
* Calculated meaningful KPIs (profit, ROI) for movie performance
* Performed comparative analysis (franchise vs. standalone)
* Identified top-performing franchises and directors
* Created visualizations to communicate findings effectively
* Documented methodology

**Appendix: Project Deliverables**

**Files Produced**

1. **tmdb\_raw\_data.csv** - Raw data from initial API fetch
2. **tmdb\_cleaned\_data.csv** - Cleaned dataset after Section 1
3. **tmdb\_cleaned\_data\_with\_credits.csv** - Final dataset with cast/crew data
4. **Section1: Data\_Preparation.ipynb** - Jupyter notebook for data acquisition and cleaning
5. **Section 2: KPI\_Implementation\_Analysis.ipynb** - Jupyter notebook for exploratory analysis and KPIs
6. **Section 3: tmdb\_visualization.ipynb** - Jupyter notebook for visualizations
7. **TMDB\_Final\_Report.docx** - This comprehensive report document

**Acknowledgments**

This project utilized the following resources:

* **The Movie Database (TMDb):** For providing comprehensive movie data via their API
* **Python Community:** For developing and maintaining the libraries used (Pandas, NumPy, Matplotlib, etc.)
* **TMDb API Documentation:** <https://developers.themoviedb.org/3>

**Note:** This product uses the TMDb API but is not endorsed or certified by TMDb.

**References**

1. The Movie Database (TMDb) API Documentation. Available at: https://developers.themoviedb.org/3
2. McKinney, W. (2022). Python for Data Analysis, 3rd Edition. O'Reilly Media.
3. VanderPlas, J. (2016). Python Data Science Handbook. O'Reilly Media.
4. Pandas Documentation. Available at: https://pandas.pydata.org/docs/
5. Matplotlib Documentation. Available at: https://matplotlib.org/stable/contents.html
6. Seaborn Documentation. Available at: https://seaborn.pydata.org/