**SENG8081 - Case Studies**

**Big Data Solution Architecture**

**Section 1**

Netflix Movies and TV shows till 2025

**A close up of a logo

AI-generated content may be incorrect.**

**Team 2**

**Team Members:**

Veera Rani

Isha Patel

Yesha Panchal

Nandhakumar Balaji

**Git Hub Repository**

<https://github.com/SENG8081/SENG8081-S25-Team2>

**Abstract**

Managing and analyzing content data is essential for understanding viewer preferences and content trends in the fast-growing world of online streaming. This project presents a Python-based approach to organizing and analyzing Netflix’s movie and TV show catalog using data sourced from TMDb (The Movie Database) and Kaggle.

The system collects detailed information, including titles, genres, release dates, ratings, and content descriptions. The data is cleaned, formatted, and stored in a structured database for smooth querying and trend analysis. The backend uses Python to connect to TMDb's API and retrieve the latest content details, ensuring the dataset remains current and relevant.

Key features of the system include genre-based categorization, rating distribution tracking, and content release timeline analysis. This dataset and system can be used to build recommendation engines, explore content trends over time, and support content-based marketing strategies.

By combining historical and live content data, this project delivers a strong base for deep insights into Netflix’s evolving media library and enhances the potential for personalized recommendations and media analytics.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Content** | **Page No.** |
| 1 | Introduction | 4 |
| 2 | System Overview | 4 |
| 3 | Data Research and Integration  * Types of Data * Data Sources | 5 |
| 4 | Data Collection | 6 |
| 5 | Database Connection and Data Insertion Process | 12 |
| 6 | Data Storage and Maintenance Database: SQL Server (Sales\_Analysis)Data LoadingStorage NeedsStorage SolutionsData Retention PoliciesData Backup and Disaster Recovery | 16 |
| 7 | Data Quality | 24 |
| **8** | **Data Documentation and Metadata Management** | **25** |
| **9** | **Data Dictionary** | **26** |
| 10 | Project Timeline | 27 |
| 11 | Reference | 28 |

# **Introduction**

# This project aims to conduct a detailed analysis of Netflix's content library, focusing on historical and current data related to movies and TV shows. It explores patterns in content genres, release trends, viewer ratings, and other metadata to understand how Netflix's catalog has evolved. The main objective is to uncover insights into content popularity, genre distribution, and viewer engagement that can support data-driven decisions and recommendations. The project uses a clean and processed dataset from Kaggle, integrates real-time updates from the TMDb API, and builds visualizations and models to highlight key trends.

**System Components for Netflix Movies & TV Shows Project**

* **Python Backend**: Used for collecting data from files and APIs, cleaning data, and preparing it for storage.
* **Real-Time API**: Fetches the latest movie and TV show details to keep the dataset updated.
* **Historical Dataset**: Contains movie and TV show data collected earlier from Kaggle and other sources.
* **Database**: Microsoft SQL Server stores all cleaned and detailed Netflix movies and TV shows data.
* **Dashboard/Visualization**: Tools like Tableau can be used to show reports and charts from the stored data.

**Data Research and Integration**

**Data Sources:**

**TMDb API – Real-Time Content Data:** The TMDb (The Movie Database) API is used to fetch real-time information on Netflix content. This includes newly added movies and TV shows, updated ratings, genre tags, and detailed descriptions. Access to this API allows the system to stay up to date with Netflix’s ever-changing library, supporting accurate content tracking and personalized recommendation features.

**API Link:** <https://api.themoviedb.org/3>

**Key Steps for Utilizing the TMDb API:**

* Register on TMDb to get an API key for access.
* Use Python libraries such as requests or official SDKs to connect and fetch data.
* Schedule periodic data synchronization to keep your content database updated in real time.
* Store new and updated records into the existing database following the same format used for historical records.

1. **Kaggle Netflix Dataset – Historical Content Data:**  
   This Kaggle dataset includes Netflix’s complete catalog of movies and TV shows up to 2025. It includes content titles, genres, release dates, durations, content types (movie/show), and ratings. This dataset serves as the historical foundation for analyzing past trends, tracking the growth of Netflix’s library, and studying genre popularity over time.

**Kaggle Dataset Link:** <https://www.kaggle.com/datasets/bhargavchirumamilla/netflix-movies-and-tv-shows-till-2025>

**Key Steps for Utilizing the Kaggle Dataset:**

* Download the dataset and load it into your Python environment using a library like pandas.
* Design a database table structure that matches the dataset’s columns (title, type, release year, genre, etc.).
* Insert the historical records into the database for analysis and future comparison with real-time updates.

**Data Management and Integration:**

Once data from both sources (Kaggle and TMDb API) is available, they are integrated into a single system:

**Database Schema:** A unified database schema is created to handle both historical and real-time Netflix content data. Tables are organized to store content titles, genres, release dates, ratings, descriptions, and types (movie or TV show). This structure ensures clean, consistent data ready for analysis or application use.

**Synchronization:** The system performs scheduled synchronization with the TMDb API to update records and fetch new entries regularly. Depending on how frequently Netflix updates its content, this synchronization can be scheduled daily or weekly. During this process, the system checks for changes in ratings, newly added shows, or updated descriptions, and then updates the database accordingly.

In summary, this project combines a historical dataset with real-time content updates from TMDb. The integration ensures the system always has the latest and most complete Netflix content data. This supports accurate trend analysis, improves recommendations, and allows for better understanding of viewer preferences over time.

**Type of data**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Description** | **Purpose** |
| **Title Metadata** | Movie/TV show title, type (Movie/Series), release year, duration | Analyze content catalog growth, identify trends by content type and release period |
| **Genre Data** | Includes customer ID, location (country), and potentially segmentation information | Understand geographic distribution, loyalty, churn rate |
| **Cast & Crew Data** | Actor, director information | Evaluate star impact and frequent collaborations |
| **Country Data** | Country of origin and availability | Track regional content and localization |
| **Rating Data** | IMDb/TMDb ratings and votes | Measure popularity and content quality |
| **Top 10 Data** | Weekly most-watched rankings | Identify high-performing content |

## 

# **Data Collection**

Python file for Data Collection from different sources and data cleaning

**1. Data Cleaning Process:**

Cleaning the raw Netflix datasets was essential for making them reliable, consistent, and analysis-ready. The process was applied to both TV Shows and Movies datasets using Python (primarily pandas, numpy, and re). Below is a step-by-step breakdown:

**2.Missing Values Handling**

**Purpose:**Missing values, especially in key columns like cast, director, and genres, can lead to incomplete or biased insights.

**Actions Taken:**

* Columns like cast, director, genres, and description were filled with "Unknown" to maintain dataset completeness.
* Rows with too many missing values (i.e., less than 5 non-null entries) were entirely removed to preserve data quality.

**Why it matters:**

Prevents analysis tools from crashing or giving misleading results when encountering NaN.

**A screen shot of a computer program

AI-generated content may be incorrect.**

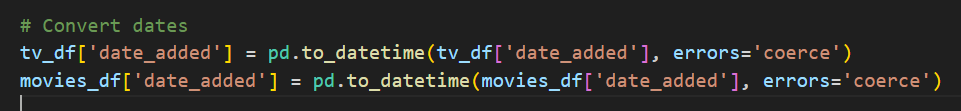
**3.Date Conversion**

**Purpose:**Dates must be in a proper format for any time-based analysis (e.g., trend over years or seasons).

**Actions Taken:**

* The column date\_added was converted to datetime format using:
* Invalid date entries were automatically turned into NaT (Not a Time), which can be filtered later.

**Why it matters:**Allows for filtering, grouping, or visualizing content based on release date (e.g., annual release trends).

****

**4.Duplicate Removal**

**Purpose:**Duplicate records inflate counts and distort metrics like total titles, revenue, or profit.

**Actions Taken:**

* Used drop\_duplicates() to remove any fully duplicated rows from the datasets.

**Why it matters:**Guarantees each movie/TV show record is counted only once in analyses like genre distribution or budget sums.

**A black background with white text

AI-generated content may be incorrect.**

**5.String Standardization**

**Purpose:**Inconsistent casing, spacing, or formatting in string fields leads to fragmented groupings during analysis.

**Actions Taken:**

* **Standardized title fields:**

Removes extra spaces and capitalizes titles (e.g., " breaking bad " → "Breaking Bad")

* **Lowercased the language field:**

**Ensures "English", "ENGLISH" and "english" are treated the same**

**Why it matters:**Improves grouping accuracy in reports (e.g., grouping by language or title-based sorting).

**A computer screen shot of text

AI-generated content may be incorrect.**

**6.Numeric Imputation & Profit Calculation**

**Purpose:**Revenue and budget values are essential for calculating profitability and content investment patterns.

**Actions Taken:**

* Filled missing values in budget and revenue with 0, assuming no data implies zero (conservatively).

**python**

* Created a new column profit:

**Why it matters:**Profitability is a key metric for evaluating content performance, especially in cost-benefit analysis.

A screen shot of a computer code

AI-generated content may be incorrect.

**7.Emoji and Symbol Removal**

**Purpose:**Emojis and non-standard Unicode characters (e.g., ©, ™, emojis) can break visualizations or crash pipelines.

**Actions Taken:**

* Defined a regex pattern to detect:
  + Emojis
  + Foreign or special characters
* Applied it across each row to filter out affected entries.

**Result:**Removed rows that contained emojis or abnormal characters in any field.

**Why it matters:**Ensures clean, consistent textual data for visual tools like Tableau or Power BI, which may not handle these characters gracefully.

A screen shot of a computer code

AI-generated content may be incorrect.

**8.Final Outcome of Cleaning**

* TV Shows: 16,000 records removed
* Movies: 16,000 records removed  
  (primarily due to emojis, duplicates, and missing critical data)

**Cleaned files:**

* netflix\_tv\_shows\_final\_clean.csv
* netflix\_movies\_final\_clean.csv

**9. Data Export**

Final cleaned datasets were saved to:

* netflix\_tv\_shows\_final\_clean.csv
* netflix\_movies\_final\_clean.csv

These files are now free from duplicates, missing critical fields, and non-standard Unicode artifacts, making them suitable for:

* Genre analysis
* Profitability trends
* Language-based content segmentation
* Release pattern trends over years

**10. Tools Used**

* **Python Libraries:** pandas, NumPy, requests, pandas, pyodbc, openpyxl
* **Storage:** Local CSV output (can be extended to SQL Server or cloud DB)
* **Regex Logic:** Used for detecting emojis and special symbols

A diagram of a software system

AI-generated content may be incorrect.

**Database Connection and Data Insertion Process**

**Connecting Python to SQL Server**

To connect Python to SQL Server, the project used the pyodbc library — a powerful and widely-used tool for ODBC (Open Database Connectivity) connections.

**A computer screen with text and images

AI-generated content may be incorrect.**

**Explanation:**

* **DRIVER**: Specifies the ODBC driver installed on your system.
* **SERVER**: Name of your SQL Server instance (e.g., YourMachineName\\SQLEXPRESS).
* **DATABASE**: The SQL database you want to use (NetflixDB).
* **Trusted\_Connection=yes**: Uses Windows authentication (no need to enter SQL username/password).

**Loading Cleaned Data from CSV:**

Before inserting into SQL Server, cleaned data is loaded from saved .csv files using pandas:

**A screen shot of a computer

AI-generated content may be incorrect.**

**3. Inserting Data into SQL Server**

**General Approach:**

* Connect to the database.
* Create a cursor object to execute SQL commands.
* Loop through each row of the DataFrame.
* Insert records using cursor.execute().
* Commit the transaction to save changes.

A computer screen shot of a program code

AI-generated content may be incorrect.

**Finalizing the Process**

After all insertions:

A black background with white text

AI-generated content may be incorrect.

**Database Schema Used**

Example Tables:

A computer screen shot of text

AI-generated content may be incorrect.

# **Data Storage and Maintenance**

## **Database: SQL Server (Sales\_Analysis)**

**Tables Created** using the below SQL script:

* **Cleaned\_Netflix\_Movies\_Data(**id, title, original\_language, release\_date, popularity, vote\_average, vote\_count, overview)
* **Netflix\_Movies\_Detailed** (show\_id, type, title, director, cast, country, date\_added, release\_year, rating, duration, genres, language, description, popularity, vote\_count, vote\_average, budget, revenue, profit)
* **Cleaned\_Netflix\_TV\_Shows** (id, name, original\_language, first\_air\_date, popularity, vote\_average, vote\_count)
* **Netflix\_TV\_Shows\_Detailed** (show\_id, type, title, director, cast, country, date\_added, release\_year, rating, duration, genres, languag, description, popularity, vote\_count, vote\_average, num\_seasons)

-- Table 1: Cleaned Netflix Movies

CREATE TABLE Cleaned\_Netflix\_Movies\_Data (

id INT,

title NVARCHAR(500),

original\_language VARCHAR(10),

release\_date DATE,

popularity FLOAT,

vote\_average FLOAT,

vote\_count INT,

overview NVARCHAR(MAX)

);

-- Table 2: Detailed Movies

CREATE TABLE Netflix\_Movies\_Detailed (

show\_id INT,

type VARCHAR(50),

title NVARCHAR(500),

director NVARCHAR(500),

cast NVARCHAR(MAX),

country NVARCHAR(255),

date\_added DATE,

release\_year INT,

rating VARCHAR(50),

duration VARCHAR(50),

genres NVARCHAR(500),

language VARCHAR(50),

description NVARCHAR(MAX),

popularity FLOAT,

vote\_count INT,

vote\_average FLOAT,

budget BIGINT,

revenue BIGINT,

profit BIGINT

);

-- Table 3: Cleaned TV Shows

CREATE TABLE Cleaned\_Netflix\_TV\_Shows (

id INT,

name NVARCHAR(500),

original\_language VARCHAR(10),

first\_air\_date DATE,

popularity FLOAT,

vote\_average FLOAT,

vote\_count INT

);

-- Table 4: Detailed TV Shows

CREATE TABLE Netflix\_TV\_Shows\_Detailed (

show\_id INT,

type VARCHAR(50),

title NVARCHAR(500),

director NVARCHAR(500),

cast NVARCHAR(MAX),

country NVARCHAR(255),

date\_added DATE,

release\_year INT,

rating VARCHAR(50),

duration VARCHAR(50),

genres NVARCHAR(500),

language VARCHAR(50),

description NVARCHAR(MAX),

popularity FLOAT,

vote\_count INT,

vote\_average FLOAT,

num\_seasons INT

);

## **Data Loading**

* Used python file to connect and insert into SQL Server.
* Mapped Brand from product CSV to BrandID via lookup dictionary.
* Applied error handling for missing brands, type conversions, and duplicate entries.
* Integrity constraints maintained using primary and foreign keys.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

## **Storage Needs**

For Netflix content analysis, storage must handle:

* High-volume metadata (titles, cast, genres)
* Time-series data (release dates, viewership trends)
* Enriched API data (TMDb ratings, images)  
  Ensure integrity/security for sensitive data (viewer ratings, proprietary content info)  
  Support fast queries for:
* Content recommendations
* Trend analysis (genre popularity over time)

## **Storage Solutions**

Cleaned Netflix data stored in **SQL Server**  
Normalized tables:

* Content (core metadata)
* Genres/ContentGenres (categorization)
* Countries/ContentCountries (production regions)

**Pre-processing in Python:**

* Fixed missing cast/director data
* Standardized date formats
* Removed duplicate titles
* Enriched with TMDb API data

## **Data Retention Policies**

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Retention Period** | **Reason** |
| Core content metadata | Indefinite | Catalog analysis |
| Viewer ratings | 5 years | Trend analysis |
| API-sourced metadata | 1 year + snapshot | License compliance |

## **Data Backup and Disaster Recovery**

* A **daily or real-time backup system** is essential to protect business-critical retail data from:
* System crashes
* Ransomware or malicious attacks
* Human errors or accidental deletion
* A **disaster recovery plan** must be in place to:
  + Restore SQL Server databases from backups
  + Re-deploy customer-facing systems like dashboards and APIs
  + Notify internal teams and mitigate impact on ongoing sales
* Regular **system maintenance** should ensure:
  + Database indexes are optimized
  + Storage drives are not nearing full capacity
  + Latest **security patches and updates** are applied to both the database and the hosting infrastructure.

# **Data Quality**

## 

## **Data Cleaning Methods**

Data Exploration

**1. Data Quality (Netflix Dataset)**

**Issues Identified:**

| **Data Quality Category** | **Examples in Netflix Dataset** |
| --- | --- |
| **Missing Values** | director, cast, country, and date\_added have nulls |
| **Duplicates** | show\_id may be duplicated in rare cases |
| **Inconsistencies** | Variations in country and rating (e.g., "United States" vs "USA") |
| **Outliers** | Some unusual release\_year values (like before 1920 or after 2025) |
| **Data Redundancy** | Repetition in genres within listed\_in |
| **Invalid Types** | Dates stored as strings, requiring type conversion |

Handling Missing Values

| **Strategy** | **Description** | **When to Use** |
| --- | --- | --- |
| **Deletion** | Remove rows or columns with too many missing values | If many nulls and not critical to analysis |
| **Imputation (Filling)** | Replace nulls with estimated values (mean, mode, forward fill, etc.) | If data is important and missing rate is manageable |
| **Placeholder values** | Replace with "Unknown" or "Not Available" | For categorical fields like director, country |

### Data Transformation

* Apply transformations such as normalization or standardization to ensure consistency and comparability across variables.
* Transform categorical variables into numerical representations using techniques like one-hot encoding or label encoding.

## 

## **Data Accuracy Assurance**

### Error Detection

* Use data profiling techniques column analysis, pattern analysis and descriptive statistics to detect errors or anomalies in the dataset.
* Look for patterns of inconsistency or unusual values that may indicate data quality issues.

### Data Cleaning Tools

* **pandas** for manual cleaning (.dropna(), .fillna(), .duplicated())
* **OpenRefine** for bulk text cleaning and clustering
* **Pyjanitor** or **Dataprep** for chaining transformation tasks

## **Data Documentation and Metadata Management**

### Metadata Creation

* Data documentation involves recording detailed information about a dataset—its structure, meaning, source, and any cleaning or transformation steps. Metadata management ensures that each data field is clearly defined and traceable.
* It helps maintain **data quality**, improves **transparency**, and ensures **consistent use** across teams or projects. Well-documented metadata includes variable names, data types, permissible values, and data lineage.

### **Data Dictionary**

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Definition | Data Type | Permissible Values |
| show\_id | Unique identifier for each show or movie | String | Alphanumeric, non-null (e.g., "s1", "s2", ...) |
| type | Type of content | String | "Movie", "TV Show" |
| title | Title of the movie or TV show | String | Text, non-null |
| director | Name(s) of the director(s) | String | Text, may be null or "Not Available" |
| cast | Main cast members | String | Text, may be null or "Not Available" |
| duration | Runtime or number of seasons | String | e.g., "90 min", "1 Season", "4 Seasons" |
| description | Summary or synopsis of the | String | Text, non-null |

# **Project Timeline:**

|  |  |  |
| --- | --- | --- |
| **Date** | **Deliverable** | **Responsible** |
| May 29 | Resource finding (API and dataset) | Veera, Yesha, Isha, Nandakumar |
| Jun 06 | Collected data from API and planned, cleaning | Veera, Nandakumar |
| Jun 20 | Stored data in the database | Yesha, Isha, Veera |
| Jun 22 | Midterm report | Isha, Yesha, Veera, Nandakumar |
| Jul 05 | Further cleaning of data |  |
| Jul 20 | Testing and solved errors |  |
| Jul 29 | System Diagram |  |
| Jul 29 | Analysis & visualization |  |
| Aug 02 | Final Report |  |

**Reference:**

*Bhargavchirumamilla. (2025, March 4). Netflix Movies and TV Shows till 2025 [Dataset]. Kaggle.* [*https://www.kaggle.com/datasets/bhargavchirumamilla/netflix-movies-and-tv-shows-till-2025*](https://www.kaggle.com/datasets/bhargavchirumamilla/netflix-movies-and-tv-shows-till-2025)

*The Movie Database (TMDb). (n.d.). TMDb API – Real-Time Content Data.* [*https://api.themoviedb.org/3*](https://api.themoviedb.org/3)

*pandas development team. (n.d.). pandas documentation – pandas 2.3.0.* [*https://pandas.pydata.org/docs*](https://pandas.pydata.org/docs)

*Kleehammer, M. (2024, October 16). pyodbc [Python package]. PyPI.* [*https://pypi.org/project/pyodbc/*](https://pypi.org/project/pyodbc/)

*Python Software Foundation. (n.d.). Python 3 documentation.* [*https://docs.python.org/3/*](https://docs.python.org/3/)

*SQLAlchemy. (n.d.). SQLAlchemy documentation.* [*https://www.sqlalchemy.org/*](https://www.sqlalchemy.org/)