## Contents

# Netflix Movie Recommendation System

## Introduction

This project aims to develop a personalized Netflix movie recommendation system. The system combines an **ETL (Extract, Transform, Load) process** for data management in a Netflix **data warehouse** with a **machine learning model** to predict movie ratings and provide recommendations. The purpose of the data warehouse is to consolidate data for analysis, while the machine learning model predicts user ratings and recommends movies based on user preferences. The project leverages **Azure Blob Storage** for data storage and **Python** for data preprocessing and model deployment.

---

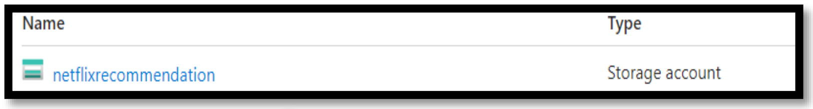
## 1. Azure Blob Storage for Data Storage and Ingestion

In this project, Azure Blob Storage was used to securely store the raw data files, including `DimUsers.csv`, `movies.csv`, and `ratings.csv`. Azure Blob Storage provides a scalable and cost-efficient solution for managing large datasets, which is essential for the Netflix recommendation system.

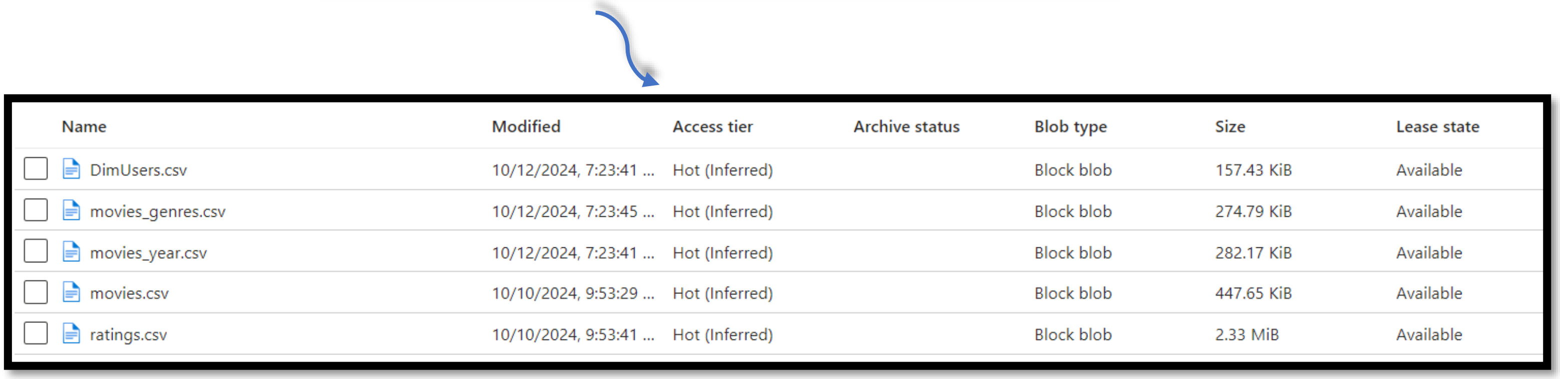
### 1. Storing Data in Azure Blob Storage

The following steps were followed to configure Azure Blob Storage for the project:

- **Creating a Blob Storage Container**: We created a container within an Azure Blob Storage account to store the CSV files containing user, movie, and ratings data. Containers in Blob Storage act as directories where files can be uploaded and managed.
- **Uploading the Data**: The data files were uploaded to the Blob Storage container, ensuring that they are accessible for further processing in Azure services. The files can be accessed via shared access signatures (SAS) for secure data transfer.



| Name | Type |
|---|---|
| netflixrecommendation | Storage account |



| Name | Modified | Access tier | Archive status | Blob type | Size | Lease state |
|---|---|---|---|---|---|---|
| DimUsers.csv | 10/12/2024, 7:23:41 ... | Hot (Inferred) | | Block blob | 157.43 KiB | Available |
| movies_genres.csv | 10/12/2024, 7:23:45 ... | Hot (Inferred) | | Block blob | 274.79 KiB | Available |
| movies_year.csv | 10/12/2024, 7:23:41 ... | Hot (Inferred) | | Block blob | 282.17 KiB | Available |
| movies.csv | 10/10/2024, 9:53:29 ... | Hot (Inferred) | | Block blob | 447.65 KiB | Available |
| ratings.csv | 10/10/2024, 9:53:41 ... | Hot (Inferred) | | Block blob | 2.33 MiB | Available |

### 2. Data Ingestion Using Azure Data Factory

To automate the process of data ingestion from Azure Blob Storage into the SQL Database for further analysis and machine learning, I utilized **Azure Data Factory** (ADF). ADF allows the orchestration of data movement and transformation at scale. The following steps outline the process:

- **Creating a Linked Service**: I created a Linked Service in Azure Data Factory to connect to the Blob Storage. This enables ADF to access the data stored in the Blob Storage.
- **Copying Data to SQL Server**: A Data Pipeline was set up in ADF to copy the raw CSV files from Blob Storage into an Azure SQL Database. This pipeline consists of:
    1. **Source Dataset**: A dataset referencing the CSV files stored in Blob Storage.
    2. **Sink Dataset**: A dataset that maps the target SQL tables in the SQL database (DimUsers, DimMovies, FactRatings).
    3. **Copy Activity**: The copy activity facilitates the transfer of data from the Blob Storage files into the SQL Server tables, ensuring a smooth integration between storage and the database.

This process significantly automates the data ingestion process, ensuring the datasets are always up-to-date and ready for analysis and machine learning tasks.

### 3 Benefits of Using Azure Blob Storage

- **Scalability**: Azure Blob Storage provides virtually unlimited storage capacity, making it suitable for large-scale data storage needs for Netflix movie recommendations.
- **Security**: The data was secured using shared access signatures (SAS), ensuring only authorized access.
- **Efficiency**: The combination of Azure Blob Storage and Azure Data Factory ensures efficient and automated data ingestion, reducing manual work and ensuring data consistency across platforms.

## 2. ETL Process for Netflix Data Warehouse

### 2.1 Purpose of the Data Warehouse

The Netflix data warehouse consolidates data from various sources to support business intelligence, analytics, and reporting activities. It focuses on movie ratings, user demographics, and subscription details to enable decision-making and strategic planning.

### 2.2 Schema Design

The star schema is implemented for the Netflix data warehouse, which includes a central fact table and several dimension tables:

- **Fact Table**:
    - **FactRatings**: Contains user ratings for movies.
- **Dimension Tables**:
    - **DimUsers**: Stores user information such as subscription type, country, age, and gender.
    - **DimMovies**: Contains movie details such as movie ID, title, and genres.

## 2.3 ETL Process Overview

The ETL process includes the following steps:

1. **Extract**: Data is extracted from CSV files including 'DimUsers.csv', 'movies.csv', and 'ratings.csv'.
2. **Transform**: Data is cleaned, validated, and structured according to the schema.
3. **Load**: The transformed data is loaded into respective dimension and fact tables in SQL Server.

## 2.4 SQL Scripts and Queries

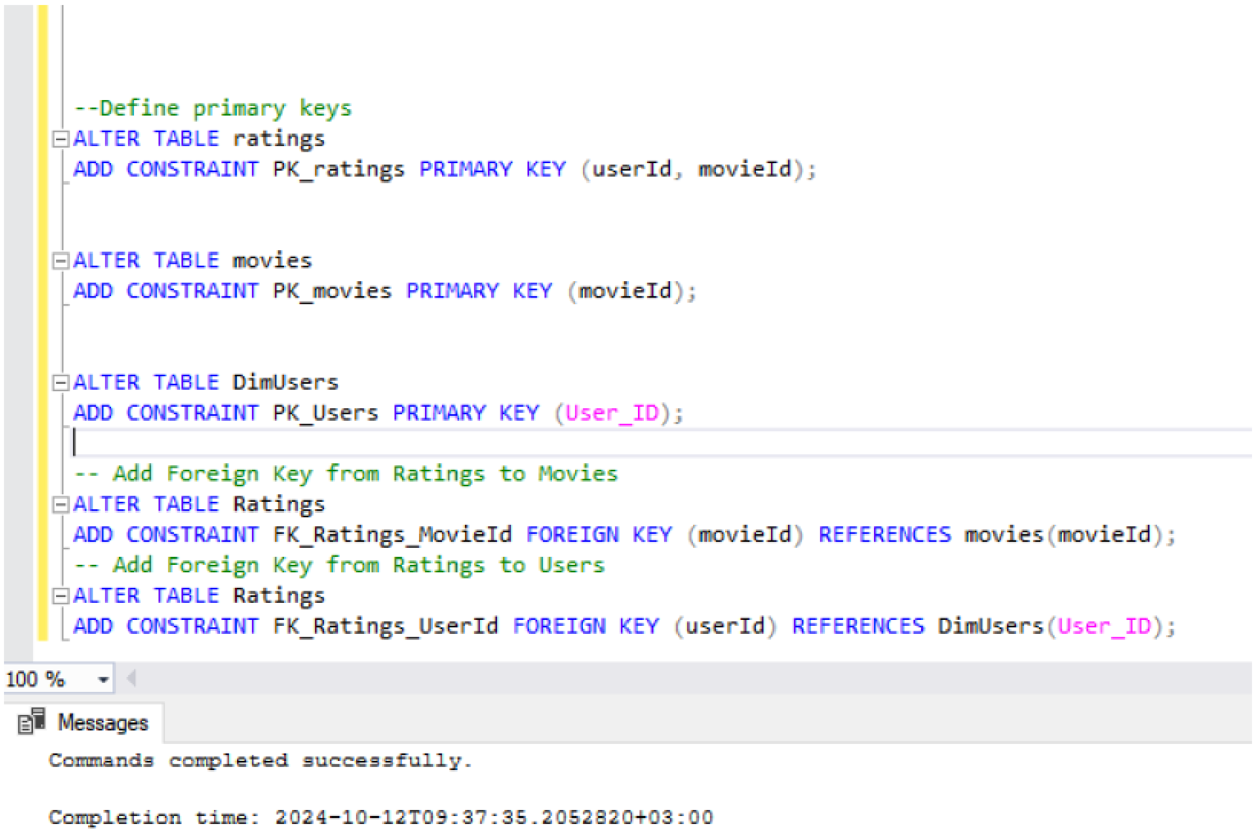
### 2.4.1 Uploading CSV Data

The following SQL command is used to upload CSV data into the database:

sql
```
BULK INSERT DimUsers FROM 'C:\Users\hanin\Downloads\DimUsers.csv'
WITH (FIELDTERMINATOR = ',', ROWTERMINATOR = '\n', FIRSTROW = 2);

BULK INSERT DimMovies FROM 'C:\Users\hanin\Downloads\movies.csv'
WITH (FIELDTERMINATOR = ',', ROWTERMINATOR = '\n', FIRSTROW = 2);

BULK INSERT FactRatings FROM 'C:\Users\hanin\Downloads\ratings.csv'
WITH (FIELDTERMINATOR = ',', ROWTERMINATOR = '\n', FIRSTROW = 2);
```

### 2.4.2 Primary Key and Foreign Key Constraints

sql
```
-- Define Primary Keys
ALTER TABLE DimUsers ADD CONSTRAINT PK_DimUsers PRIMARY KEY (User_ID);
ALTER TABLE DimMovies ADD CONSTRAINT PK_Movies PRIMARY KEY (movieId);
ALTER TABLE FactRatings ADD CONSTRAINT PK_FactRatings PRIMARY KEY (userId, movieId);

-- Define Foreign Keys
ALTER TABLE FactRatings ADD CONSTRAINT FK_Ratings_UserId FOREIGN KEY (userId) REFERENCES
DimUsers(User_ID);
ALTER TABLE FactRatings ADD CONSTRAINT FK_Ratings_MovieId FOREIGN KEY (movieId)
REFERENCES DimMovies(movieId);
```



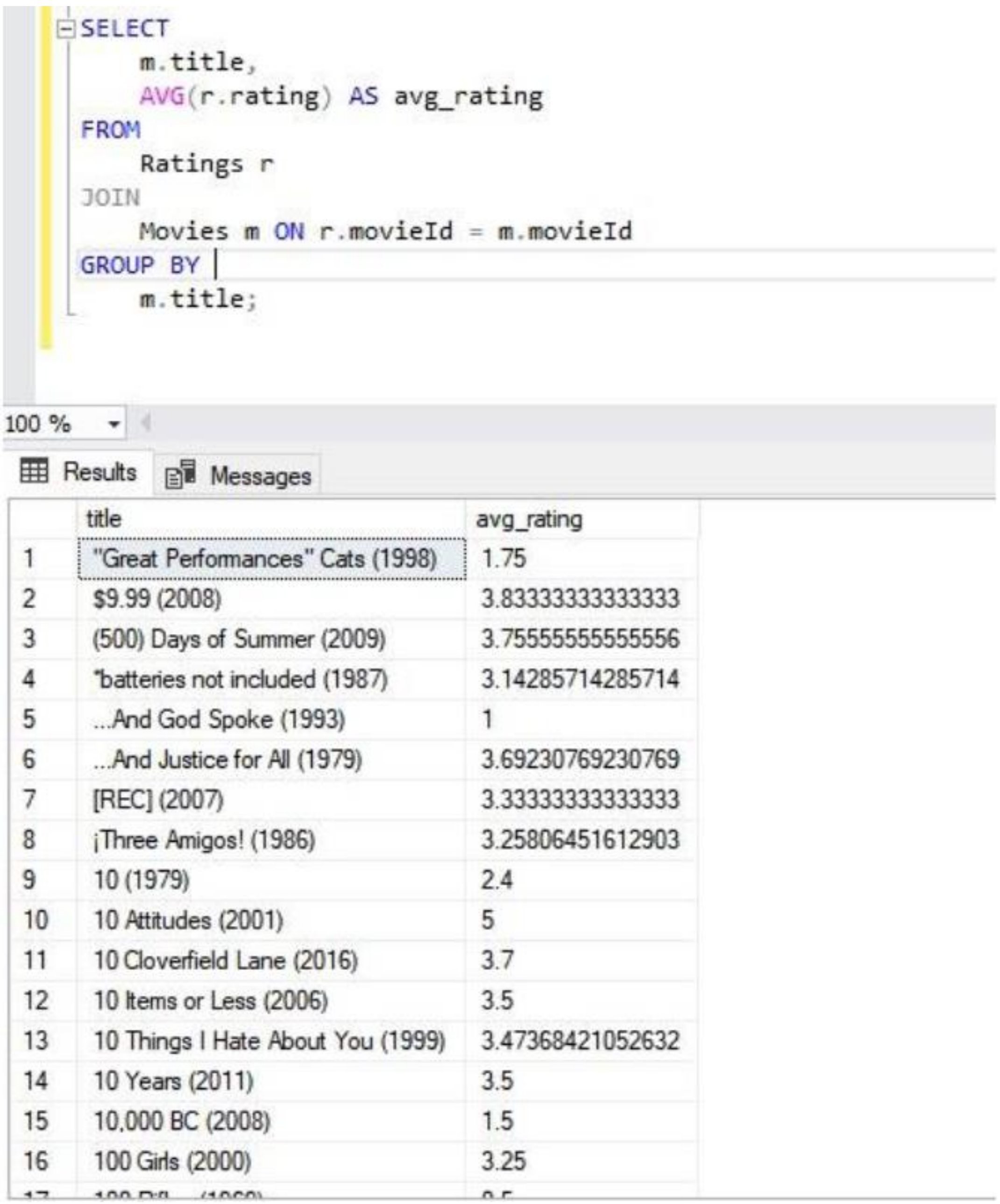
```
--Define primary keys
ALTER TABLE ratings
ADD CONSTRAINT PK_ratings PRIMARY KEY (userId, movieId);


ALTER TABLE movies
ADD CONSTRAINT PK_movies PRIMARY KEY (movieId);


ALTER TABLE DimUsers
ADD CONSTRAINT PK_Users PRIMARY KEY (User_ID);

-- Add Foreign Key from Ratings to Movies
ALTER TABLE Ratings
ADD CONSTRAINT FK_Ratings_MovieId FOREIGN KEY (movieId) REFERENCES movies(movieId);
-- Add Foreign Key from Ratings to Users
ALTER TABLE Ratings
ADD CONSTRAINT FK_Ratings_UserId FOREIGN KEY (userId) REFERENCES DimUsers(User_ID);
```

```
Commands completed successfully.

Completion time: 2024-10-12T09:37:35.2052820+03:00
```

## 2.5 Testing the Data Warehouse

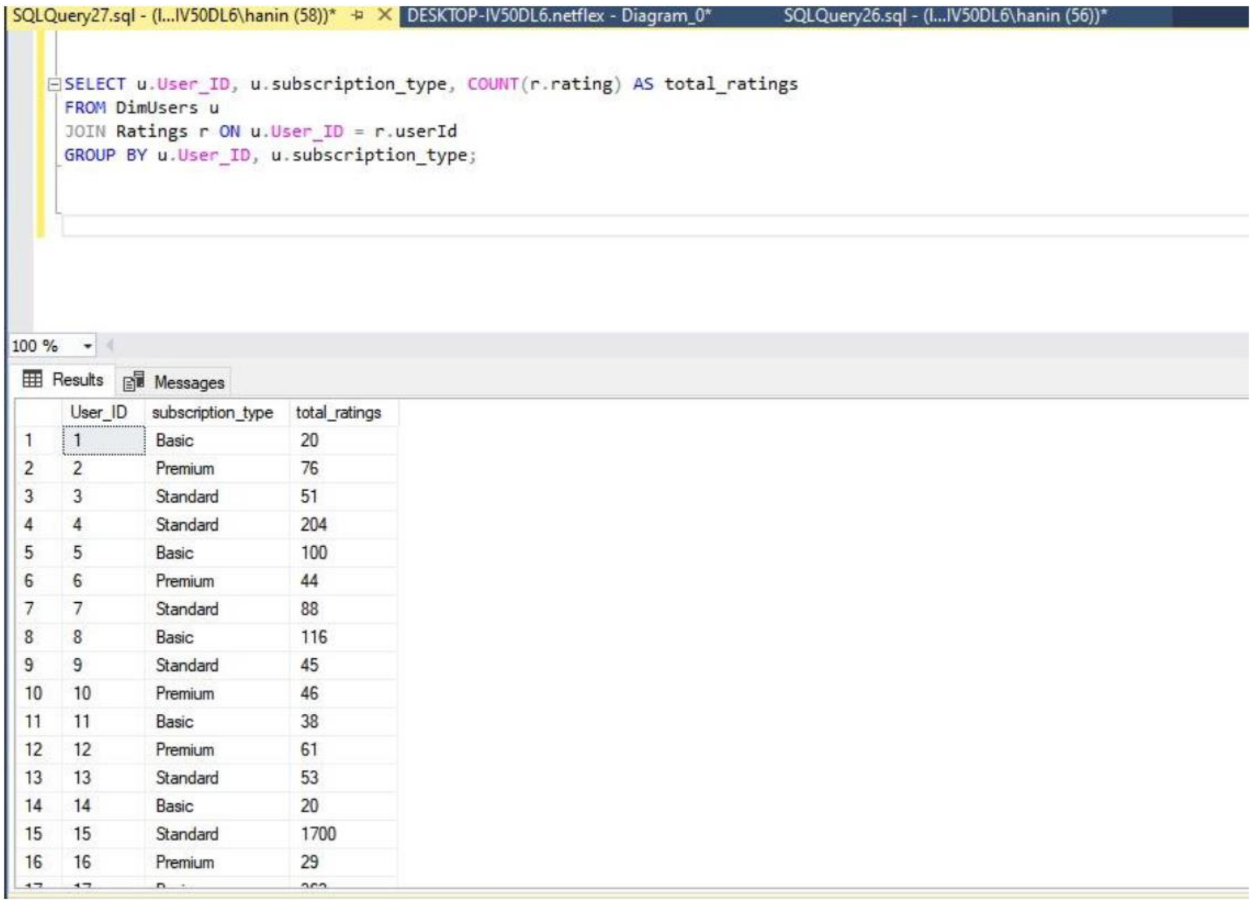
To test the data warehouse, the following SQL query retrieves the average rating for each movie along with the movie title:

```
sql
SELECT m.title, AVG(r.rating) AS avg_rating
FROM FactRatings r
JOIN DimMovies m ON r.movieId = m.movieId
GROUP BY m.title;
```



```sql
SELECT
    m.title,
    AVG(r.rating) AS avg_rating
FROM
    Ratings r
JOIN
    Movies m ON r.movieId = m.movieId
GROUP BY
    m.title;
```

| | title | avg_rating |
|---|---|---|
| 1 | "Great Performances" Cats (1998) | 1.75 |
| 2 | $9.99 (2008) | 3.83333333333333 |
| 3 | (500) Days of Summer (2009) | 3.75555555555556 |
| 4 | *batteries not included (1987) | 3.14285714285714 |
| 5 | ...And God Spoke (1993) | 1 |
| 6 | ...And Justice for All (1979) | 3.69230769230769 |
| 7 | [REC] (2007) | 3.33333333333333 |
| 8 | ¡Three Amigos! (1986) | 3.25806451612903 |
| 9 | 10 (1979) | 2.4 |
| 10 | 10 Attitudes (2001) | 5 |
| 11 | 10 Cloverfield Lane (2016) | 3.7 |
| 12 | 10 Items or Less (2006) | 3.5 |
| 13 | 10 Things I Hate About You (1999) | 3.47368421052632 |
| 14 | 10 Years (2011) | 3.5 |
| 15 | 10,000 BC (2008) | 1.5 |
| 16 | 100 Girls (2000) | 3.25 |


SQLQuery27.sql - (l...IV50DL6\hanin (58))*
DESKTOP-IV50DL6.netflex - Diagram_0*
SQLQuery26.sql - (l...IV50DL6\hanin (56))*
100 %
Results
Messages

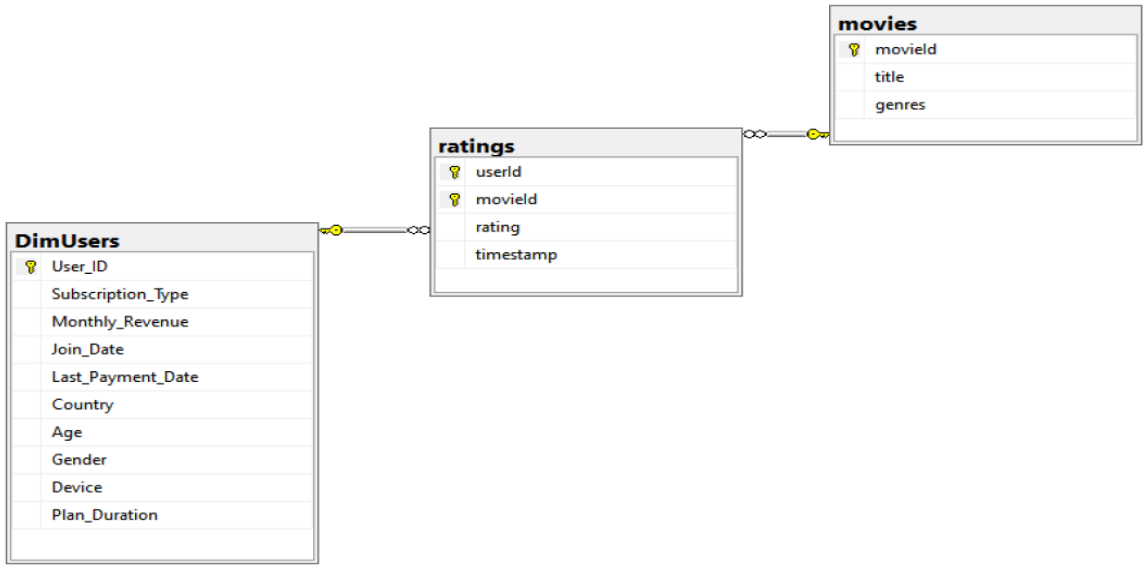
```
SELECT u.User_ID, u.subscription_type, COUNT(r.rating) AS total_ratings
FROM DimUsers u
JOIN Ratings r ON u.User_ID = r.userId
GROUP BY u.User_ID, u.subscription_type;
```

| | User_ID | subscription_type | total_ratings |
|---|---|---|---|
| 1 | 1 | Basic | 20 |
| 2 | 2 | Premium | 76 |
| 3 | 3 | Standard | 51 |
| 4 | 4 | Standard | 204 |
| 5 | 5 | Basic | 100 |
| 6 | 6 | Premium | 44 |
| 7 | 7 | Standard | 88 |
| 8 | 8 | Basic | 116 |
| 9 | 9 | Standard | 45 |
| 10 | 10 | Premium | 46 |
| 11 | 11 | Basic | 38 |
| 12 | 12 | Premium | 61 |
| 13 | 13 | Standard | 53 |
| 14 | 14 | Basic | 20 |
| 15 | 15 | Standard | 1700 |
| 16 | 16 | Premium | 29 |

Associated Azure Storage Link: [https://netflixrecommendation.blob.core.windows.net/netflix-data?sp=r&st=2024-10-12T16:26:33Z&se=2024-11-30T01:26:33Z&spr=https&sv=2022-11-02&sr=c&sig=jrJyXaMttjGkGOYhbTniPDAippHSJUpr0h3%2BJ0Y6yvc%3D](https://netflixrecommendation.blob.core.windows.net/netflix-data?sp=r&st=2024-10-12T16:26:33Z&se=2024-11-30T01:26:33Z&spr=https&sv=2022-11-02&sr=c&sig=jrJyXaMttjGkGOYhbTniPDAippHSJUpr0h3%2BJ0Y6yvc%3D)

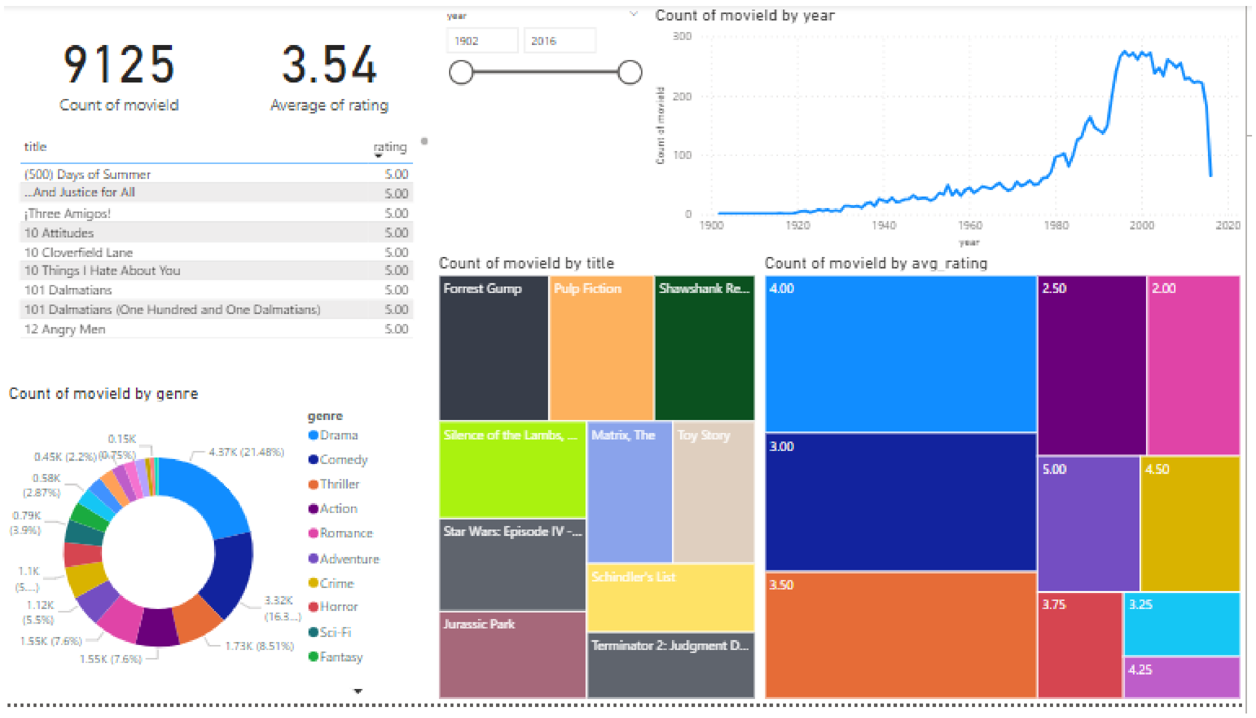
## 2.6 Conclusion

This ETL process documentation outlines the purpose, schema design, and SQL scripts for the Netflix data warehouse. The successful implementation allows efficient data analysis and reporting.



## 3. Visualization

The data source from SSMS.



| title | rating |
|---|---|
| (500) Days of Summer | 5.00 |
| ...And Justice for All | 5.00 |
| ¡Three Amigos! | 5.00 |
| 10 Attitudes | 5.00 |
| 10 Cloverfield Lane | 5.00 |
| 10 Things I Hate About You | 5.00 |
| 101 Dalmatians | 5.00 |
| 101 Dalmatians (One Hundred and One Dalmatians) | 5.00 |
| 12 Angry Men | 5.00 |

## 3.1 Overall Metrics:

Total Number of Movies Rated: 9,125 movies have been rated in the dataset.

Average Rating: The overall average rating across all movies is 3.54. This indicates a generally positive sentiment in the dataset, with the majority of ratings being above the midpoint (2.5).

### 3.2 Top-Rated Movies:

The "Count of Movied by Title" chart highlights some iconic films such as Forrest Gump, Pulp Fiction, and The Shawshank Redemption, which received high numbers of ratings.

These movies are classics that have had a lasting cultural impact, often appearing on "best of" lists, indicating that they resonate with a large audience.

### 3.3 Genre Preferences:

The pie chart of movie counts by genre shows that Drama is the most common genre, followed by Comedy and Thriller.

The high number of ratings for these genres suggests a preference for more emotionally engaging and diverse narratives, while genres like Fantasy and Sci-Fi are less represented.

### 3.4 Average Rating by Movie:

The "Count of Movied by Title" chart highlights some iconic films such as Forrest Gump, Pulp Fiction, and The Shawshank Redemption, which received high numbers of ratings.

These movies are classics that have had a lasting cultural impact, often appearing on "best of" lists, indicating that they resonate with a large audience.

### 3.5 Count of Movies by Year

The "Count of Movies by Year" chart shows a steady increase in movie production over time, with a significant surge in the late 20th century.

Associated PowerBI link: https://app.powerbi.com/links/x3kay7w7j0?ctid=0bc92751-071a-4e2c-a48b-633206fef374&pbi_source=linkShare&bookmarkGuid=fb3bbaa1-e1dc-478f-a68a-1e3b401d92f8

Google Colab link:

https://colab.research.google.com/drive/1kqfm0G48gpLpTX0dKyYW7qt5tkjs_IAG?usp=sharing

## 4. Machine Learning Model and Algorithms

### 4.1 Model Overview

The Netflix Movie Recommendation System is built using a machine learning model that predicts user ratings for movies and recommends films that are likely to be enjoyed by the user.

The model utilizes collaborative filtering techniques and neural network-based algorithms to predict the ratings a user might give to unrated movies, thereby enabling personalized recommendations. The key components of this system include data preprocessing, model training, and the recommendation engine.

## 4.2 Data Preprocessing

The data preprocessing stage is crucial to ensuring the model can accurately predict movie ratings. The original datasets for users, movies, and ratings are loaded from Azure Blob Storage. Since the machine learning model requires categorical data, user IDs and movie IDs are transformed into numerical codes using Pandas astype('category').cat.codes. This ensures that each unique user and movie is represented by an integer, simplifying the input for the model.

The dataset is then split into training and testing sets using an 80/20 ratio. This allows the model to be trained on one portion of the data while its accuracy is validated on unseen data.

## 4.3 Neural Collaborative Filtering (NCF) Model

The recommendation system employs a Neural Collaborative Filtering (NCF) model, which combines collaborative filtering techniques with deep learning to predict user ratings.

The NCF model is constructed using the following components:

1. **Embedding Layers**:
    - One embedding layer for user IDs and another for movie IDs, which convert these categorical variables into dense vectors of size 50. These embeddings allow the model to learn latent factors that represent user and movie characteristics.
2. **Concatenation Layer**:
    - The user and movie embeddings are concatenated to form a combined input vector that represents both entities. This combined vector serves as input to the hidden layers of the neural network.
3. **Hidden Layers**:
    - Two fully connected layers (Dense layers) with 128 and 64 neurons are used to learn interactions between the user and movie embeddings. Batch normalization and dropout layers are applied between these hidden layers to prevent overfitting and stabilize learning.
4. **Output Layer**:
    - A single neuron output layer predicts the rating the user is likely to give to a specific movie. The model is trained to minimize the mean squared error (MSE) between predicted and actual ratings.
5. **Optimizer**:
    - The model is optimized using the AdamW optimizer, a variant of Adam that incorporates weight decay for better generalization. The learning rate is set to 0.001.

## 4.4 Model Training

The model is trained on the training data for 10 epochs using a batch size of 256. During training, the model learns to predict ratings by minimizing the MSE loss. The input to the model consists of two arrays: one for user IDs and one for movie IDs. The target output is the corresponding rating for each user-movie pair. The model achieves generalization by using a validation split of 10%, allowing the model to be evaluated on unseen data during training.

### 4.5 Recommendation Engine

Once the model is trained, it is used to generate personalized movie recommendations for each user. The recommendation process consists of the following steps:

1. **Filtering Rated Movies**:
   - For a given user, the model filters out movies that the user has already rated. This ensures that only unseen movies are considered for recommendation.
2. **Predicting Ratings**:
   - The model predicts ratings for the unrated movies by inputting the user ID and the list of unwatched movie IDs into the trained NCF model.
3. **Generating Recommendations**:
   - The predicted ratings are sorted in descending order, and the top N movies with the highest predicted ratings are selected for recommendation. These recommendations are merged with the movie dataset to provide details such as movie titles and genres.

### 4.6 Deployment

The recommendation system is deployed using Gradio, a Python library for creating user interfaces for machine learning models. The interface allows users to input their user ID, and the system returns a list of recommended movies along with predicted ratings. This makes it easy for users to interact with the model and receive personalized recommendations in real-time.

Associated Link:
https://colab.research.google.com/drive/1B3zzWCkWp3B28i7bIuht5ZVQd7aLYFlk?usp=sharing

## Conclusion

This project developed a personalized Netflix movie recommendation system using an ETL process, data visualization, and a machine learning model. The data warehouse in SQL Server and visualizations in Power BI provided insights into movie ratings and trends. The Neural Collaborative Filtering model predicted user preferences, enabling personalized recommendations.

Using Azure Blob Storage and Python, the system ensures scalability and performance, demonstrating the power of data science and machine learning in improving user experiences on platforms like Netflix.