**Task 3: SQL Database Implementation Documentation**

**Project:** StreamFlix Subscription Analytics

**Purpose**

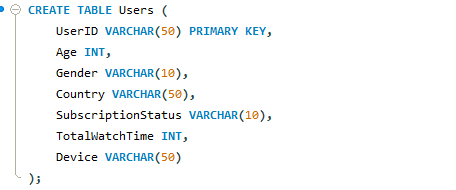
This documentation outlines the implementation of SQL tables for Users, Movies, and Ratings to support StreamFlix's transition from free to paid subscriptions by analyzing viewer behaviour.

1. **Creating Tables with Appropriate Fields, Primary Keys, and Constraints**

To store user and movie data effectively, we designed and implemented three main tables: **Users**, **Movies**, and **Ratings**. Below are the SQL commands used to create these tables.

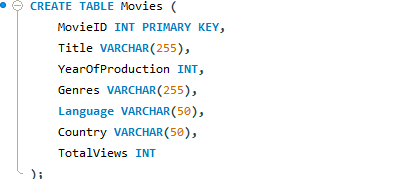
**Users Table**

This table stores user details, including their subscription status.



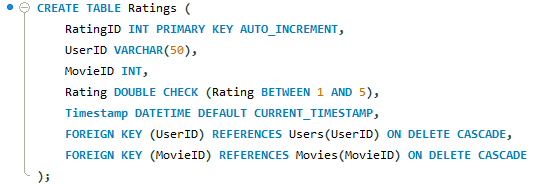
**Movies Table**

This table contains details about movies such as genre, year of production and country .

****

**Ratings Table**

The Ratings table connects Users and Movies, allowing users to rate movies.

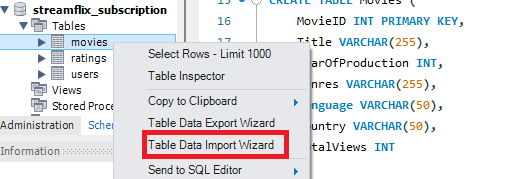


The **ON DELETE CASCADE** in the ratings table ensures that when a referenced movie or user is deleted, all related ratings are automatically removed, preventing orphaned records and reducing manual cleanup.

A **CHECK constraint** enforces that the ratings remain within a valid range (e.g., 0 to 5) to maintain data accuracy.

1. **Importing Data into SQL**

The data for the three tables (Users, Movies, and Ratings) was provided and successfully imported into SQL using the Import Wizard. This method allowed for an efficient and user-friendly way to load data without manually writing SQL import commands**.**

****

**3. Implementing the Ratings Table**

This has already been defined in the schema (Copy\_of\_Ratings), with appropriate columns like RatingID, UserID, MovieID, Rating, and Timestamp.

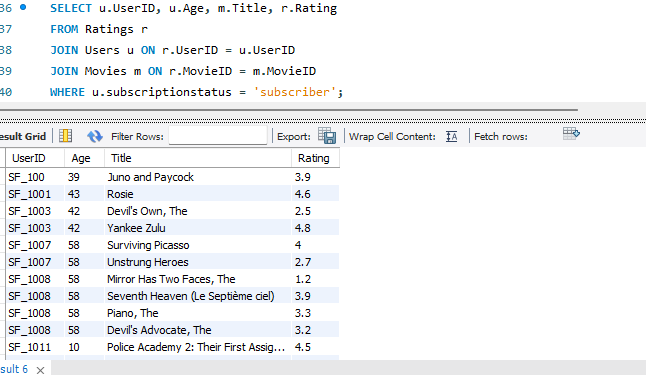
1. **Linking Users and Movies Using Foreign Keys**

The Ratings table establishes a many-to-many relationship between Users and Movies by using user\_id and movie\_id as foreign keys, ensuring data integrity.

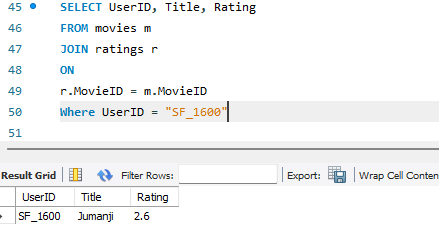
1. **Testing Relationships with SQL Queries**

To verify relationships between tables, the following SQL queries were executed:

* Retrieve ratings for movies rated by subscribed users



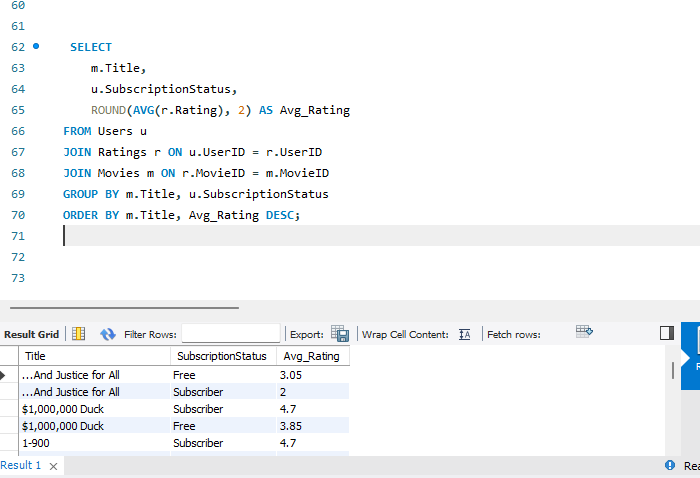
* Retrieve movies rated by a specific user



1. **Next Steps**

* Validate the database schema and relationships.
* Test SQL queries for correctness and performance.

1. **Retrieving Average rating per movies group by subscription status**



This query retrieves **average ratings per movie** grouped by **subscription status** (Paid vs. Free).

**Technical Breakdown:**

* **Joins three tables**:
  + Users Contains SubscriptionStatus
  + Ratings Stores Rating given by users
  + Movies Contains Title for each MovieID
* **Groups by** Movie Title and Subscription Status.
* **Calculates the average rating** per category.
* **Orders results** by highest-rated movies.

1. **Conclusion**

The successful implementation of these tables ensures a structured database that supports analyzing user engagement and movie preferences for Streamflix's transition to a paid subscription model.