Course: Advanced Database Techniques

Course Code: CAP570

Project Title: -

Subscription Based Streaming Service Database

Submitted by

1. Siddharth Kumar (12404553),Grpup-9
2. Ritesh Singh Kushwaha(12410009) ,Grpup-9
3. Amritanshu Kumar(12413909) ,Grpup-9

Submitted to

Dr. Heena Khanna
UID: 31463
Assistant Professor, SCA, LPU

Acknowledgment

We would like to express our sincere gratitude to **Dr. Heena Khanna(31463), Assistant Professor, School of Computer Applications, Lovely Professional University**, for her guidance and continuous supervision, as well as for providing essential information and unwavering support throughout the project. Her consistent direction and willingness to share her extensive knowledge enabled us to gain a deep understanding of the project, which greatly assisted us in completing our tasks on time.

We would also like to extend our heartfelt thanks to all the individuals who graciously participated in the interviews and surveys conducted during the project. Their experiences and insights provided a practical understanding of how database systems are used in real-world applications, which greatly enriched the quality of this project.

Finally, we would like to thank our families and friends for their constant encouragement and support during this journey. This project would not have been possible without the contribution of everyone involved.

Thank you all.

Table of Contents

1. INTRODUCTION

In the era of digital entertainment, online streaming platforms have become an integral part of modern life, providing users with a wide variety of content at their fingertips. These platforms rely on robust database management systems to handle vast amounts of data while ensuring seamless user experiences.

[1] This report aims at describing the creation of a database system that has been created for an online streaming firm. The actual design of the database structure replicates those of front-end platforms similar to Netflix, Amazon Prime Video, and Disney Plus Hotstar, proving how such platforms organize their massive datasets. By understanding and simulating their database structures, this project aims to explore effective data organization techniques and user behavior analysis to offer better content recommendations and services.

These core platforms rely on a robust database management system that facilitates essential services such as user management, subscription handling, content categorization, device compatibility, and payment processing. Additionally, these systems enable in-depth analysis of user behavior, which informs business strategies such as content acquisition and recommendation algorithms.

This project is a proof of concept – that the basic principles and methods used in today's streaming companies can be applied and efficiently course data. Thus, it will help to close the gap between the top-level theories and practical implementation as well as provide the response time, performance, and valuable user-oriented analysis.

2. PROJECT OVERVIEW

The proposed refers to creating a template for a database management system that corresponds to an online streaming platform services similar to Netflix, Amazon Prime Video, Disney + Hotstar etc. This database has to perform a number of key tasks including user registration, subscription control, content delivery, device identification, watch-history tracking, payments, and content licensing. The objectives of the project are to sort the data effectively and optimize the interaction of users with the platform while making decisions based on the results of analyzing the behavior of users of the platform.

2.1 Core Concept [2]

The central objective of this database is to provide an efficient and scalable system that can manage the following key aspects:

- User Management: Involves handling information concerning users and their activities and providing users with interesting content.
- Subscription Management: To make it function properly, users must be allowed to choose, promote, and restore subscription plans and monitor their lifecycle.
- Content Streaming: Provide a way of categorizing and presenting content in different categories, genres, and languages to suit the user.
- Payment Processing: Record payments so that the bills would be properly updated and subscriptions to be renewed punctually.
- Device and Watch History Management: Devices for streaming should be tracked and a history of users' watches should be recorded carefully in order to show relevant suggestions.
- Content Licensing: Track licensing information based on region and access available to the content material.

2.2 Scope of the Project

The project covers the development of a robust and normalized relational database that:

- It can handle multiple users and the users can choose the subscription plan. The platform logs users' video-watching histories, so it can provide more user-targeted content.
- Responsible for content information processes such as the type of content provided, genre of content as well as content ratings.

- It maintains records of linked devices for each user for multi-streaming compatibility.
- Enables smooth payment operations for subscription reactivation and improvements.
- Controls the licensing information to meet the regional distribution rights governing content distribution.

2.3 Entities and Attributes

- Users:- user_id (PK), first_name, last_name, email, password, phone_number, address, created_at, status
- Subscription Plans:- plan_id (PK), plan_name, monthly_cost, resolution, max_devices, created_at, status
- Subscriptions:- subscription_id, user_id, plan_id, start_date, end_date, status, auto |_id
- Contents:- content_id (PK), title, content_type, genre, release_date, language, duration_minutes, rating, description, age_rating, added_on
- Devices:- device_id (Primary key), user_id (foreign key), name, type, operating system, date registered
- Watch History:- history_id (PK), user_id (FK), content_id (FK), watched_at, progress, completed
- Payments:- payment_id (primary), user_id (foreign), subscription_id (foreign), amount, payment_type, payment_time, payment_status
- Content Licensing:- License_id (Primary), content_id (Foreign), license_start, license_end, region

2.4 Relationships Between Entities

- **Users**
  - A **user** can have **multiple subscriptions**:

    Relationship: One-to-Many

    Keys: user_id (PK in Users) → user_id (FK in Subscriptions)
  - A **user** can have **multiple watch histories**:

    Relationship: One-to-Many

    Keys: user_id (PK in Users) → user_id (FK in Watch History)
  - A **user** can have **multiple payments**:

    Relationship: One-to-Many

    Keys: user_id (PK in Users) → user_id (FK in Payments)

- **Subscription Plans**
  - A **subscription plan** can be subscribed to by **many users**:

    Relationship: One-to-Many

    Keys: plan_id (PK in Subscription Plans) → plan_id (FK in Subscriptions)
- **Subscriptions**
  - A **subscription** links a **user** to a **subscription plan**:

    Relationship: Many-to-One

    Keys: user_id (FK in Subscriptions) → user_id (PK in Users)

    plan_id (FK in Subscriptions) → plan_id (PK in Subscription Plans)
  - A **Subscriptions** can have **multiple devices**:

    Relationship: One-to-Many

    Keys: subscription_id (PK in Subscriptions) → subscription_id (FK in Devices)
- **Contents**
  - A **content item** can appear in **multiple watch histories**:

    Relationship: One-to-Many

    Keys: content_id (PK in Contents) → content_id (FK in Watch History)
  - A **content item** can have **multiple content licenses**:

    Relationship: One-to-Many

    Keys: content_id (PK in Contents) → content_id (FK in Content Licensing)
- **Devices**
  - A **device** is linked to **subscriptions**:

    Relationship: Many-to-One

    Keys: subscription_id (FK in Devices) → subscription_id (PK in Subscriptions)
- **Watch History**
  - A **watch history** links a **user** to a **content item**:

    Relationship: Many-to-One

    Keys: user_id (FK in Watch History) → user_id (PK in Users)

    content_id (FK in Watch History) → content_id (PK in Contents)
- **Payments**
  - A **payment** is linked to a **user** and a **subscription**:

    Relationship: Many-to-One

    Keys: user_id (FK in Payments) → user_id (PK in Users)

    subscription_id (FK in Payments) → subscription_id (PK in Subscriptions)
- **Content Licensing**
  - A **content item** can have **multiple licenses** across different **regions**:

    Relationship: One-to-Many

    Keys: content_id (PK in Contents) → content_id (FK in Content Licensing)

3. PROJECT OBJECTIVE

- Develop a structured database for the online streaming platform Development of a structured and integrated database that will cover user, content and subscription details, payments, watching history, and content rights for an online streaming platform.

- Implement the entire database using Microsoft SQL Server (MSSQL), leveraging its relational database capabilities to manage structured data efficiently, enforce data integrity, and provide robust query optimization for large datasets.

- Design a data model that is affordable at scale (e.g., millions of users; thousands of pieces of content; and extended viewing history).

- Structure an effective means of processing subscriptions and plans, subscription plans with easy ways for people to select a preferred plan, and track his/her payment history and renewals.

- Evaluate the option to incorporate a system to remember a user's watch history to analyze data and get insights about the user's behavior and interests catering to their needs.

- Develop a Regions Sales Constraints system for the content licensable that addresses legal and contractual restrictions of content distribution in a given region.

- Design a system that gives a list of recommendations to the user with regards to series they would like to watch next, series that they might like based on their watch history or even ratings given.

- Develop an efficient way of handling devices and their registration for streaming services to enhance how user experiences content streaming across the multiple devices such as smart mobiles devices, laptops, smart TVs etc.

- Establish an effective mechanism for organizing users and their preferences, subscription and associated devices for easy use.

4. PROBLEM STATEMENT

The core problem the database system is solving is that of an efficient management of large amounts of user data, subscriptions, and media content for a Subscription-Based Streaming Service. Platforms like Netflix, Hulu, and Spotify all contend with humongous data in terms of users, their preferences, content metadata, subscription plans, and payment transactions in such a way that it ensures access quick and high performance with guaranteed data integrity.

5. REQUIREMENT GATHERING

5.1 Interviews and Surveys

5.1.1 Objective:

The objective of the interviews and questionnaires would be to know how real-world streaming services and similar systems handle vast databases of humongous size containing user details, subscription data, and content management. Of utmost importance are points of significant challenges, needs, and best practices that are emphasized within streaming service database management.

5.1.2 Method:

The students conduct interviews with IT personnel who work with or manage databases for subscription-based services like streaming services. Surveys were conducted with target end-users, including video streaming service subscribers, to determine what features would be important to manage profiles, preferences, and subscriptions. Professionals were asked questions regarding the performance, security, and scalability of current systems, but actual end-users were surveyed to find out what features are the most critical to them from a database viewpoint. Key Findings:

- Scalability and Data Performance: According to respondents, this has been the main issue-large stores of user data and media content, which must be accessed in an exceptionally fast manner with low latency. The database design must support efficient queries for content search, user history, and subscription tracking.

- Subscription Management. Subscription status and plan change, renewal, and cancellation tracking were pointed out as one of the very key requirements. Linking users to specific plans, managing users' payment history, and ensuring correct billing were cited as critical functionalities.

- Content Management: System administrators realized that an arranged list of the contents with features such as genres, year of publication, cast, and availability of the content is necessary. Many services track user ratings or reviews regarding the content; mostly stored in the database to support personalized suggestions.

- User Behavior and Preferences: The most significant interest in this group was the viewable history of the user, enabling users to construct a watchlist, and for receiving personalized recommendations. Storing and analyzing user activity data are the main decisions in building a personal user experience.

5.2 Website Exploration [3]

5.2.1 Objective:

We sought to investigate the currently provided streaming services to understand how these structure their databases to store and manage large-scale content and user information. This involved an examination of database schema, user management systems, and subscription models. Web Services Analyzed are Netflix, Spotify, Hulu , Amazon Prime Video.

5.2.2 Content Structure:

All the services employ an extensive metadata model for content, including content ID, title, year of release, genre, ratings, cast and crew, and even user reviews. A relational database is applied to categorize content as belonging to multiple genres and enables easy access for search and recommendation algorithms.

5.2.3 Subscription Models:

Subscription management, for instance, involves Netflix and Spotify with significant information related to subscription types-Basic, Standard, Premium-date of renewal of subscription, and mode of payment. Databases allow upgrading or degrading, suspending, or even cancelling accounts while keeping track of history of payments.

5.2.4 Payment and Billing:

All platforms track payment transactions, amount paid, and billing cycles. Payments are tied to user accounts with the option to capture renewal or failure statuses for reporting purposes.

6. ER DIAGRAM AND RELATIONAL SCHEMA

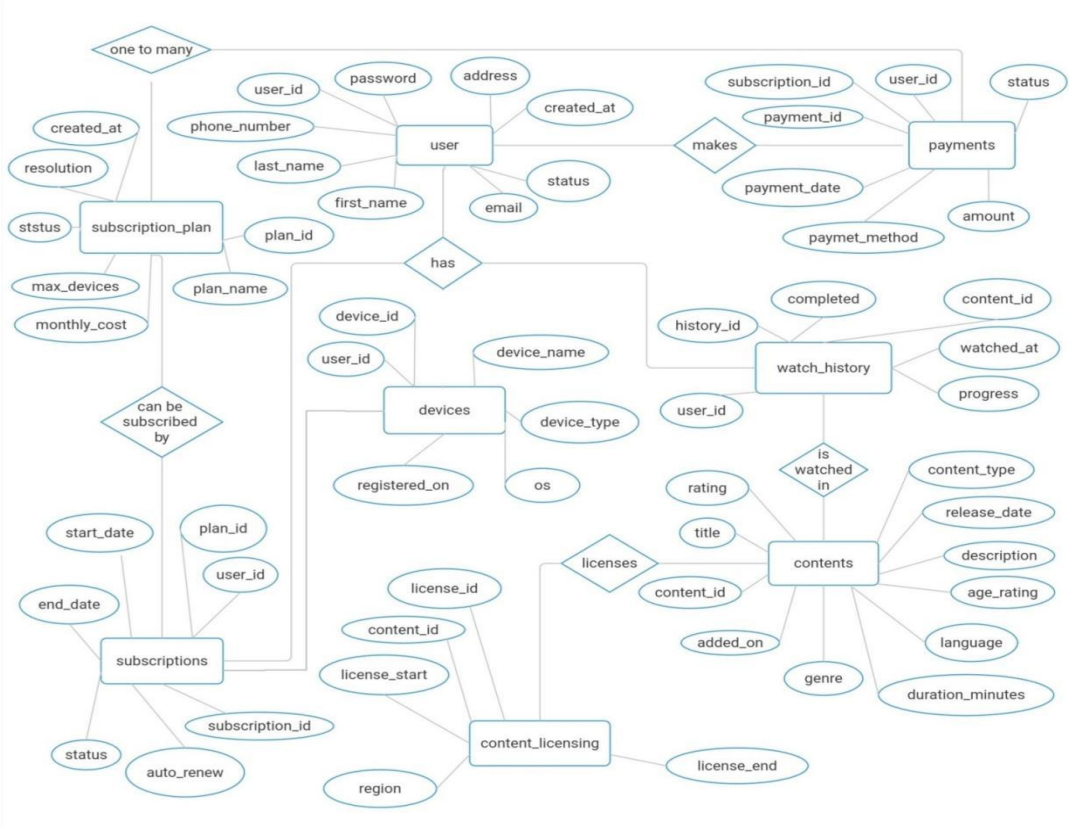

Fig.1 ER Diagram

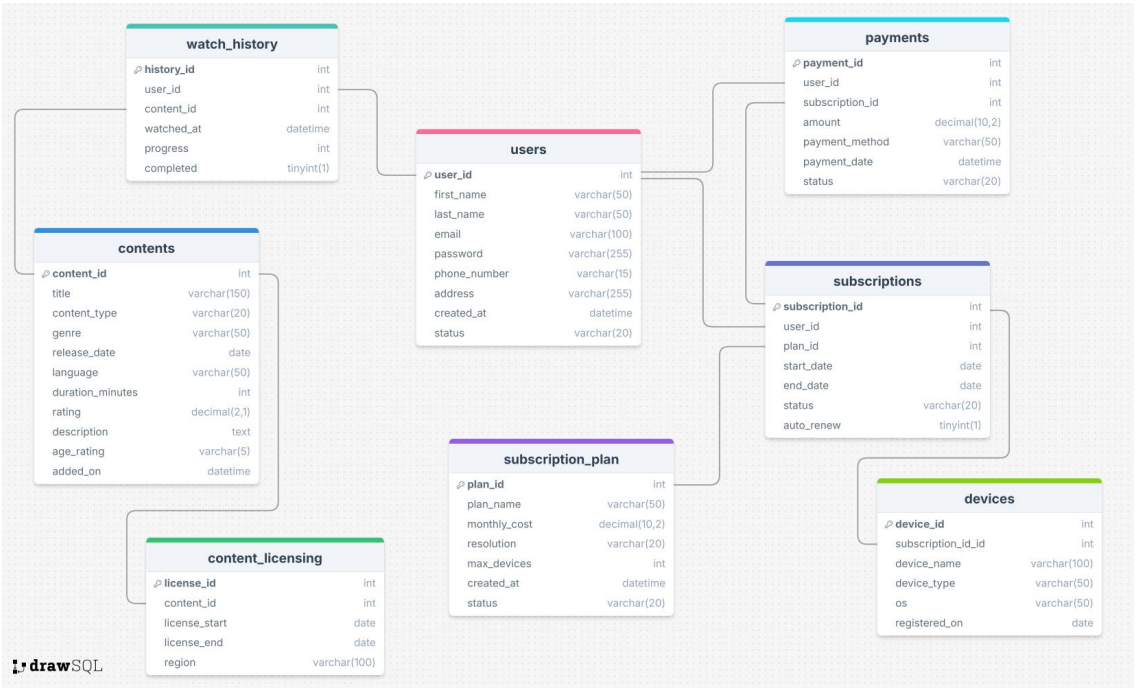

Fig.2 Database Schema

7. DATABASE DESIGN AND NORMALIZATION

7.1 Table Descriptions

- **Table 1:** Users
    - Attributes: user_id, first_name, last_name, email, password, phone_number, address, created_at, status.
    - Primary Key: user_id
    - **Description:** Stores information about the users registered on the platform.

- **Table 2:** Subscription Plan Table
    - Attributes: plan_id, Movie plan_name, monthly_cost, resolution, max_devices, created_at, status.
    - Primary Key: plan_id
    - **Description:** Stores information about the different subscription plans offered.

- **Table 3:** Subscriptions Table
    - Attributes: subscription_id, user_id, plan_id, start_date, end_date, status, auto_renew
    - Primary Key: subscription_id
    - **Description:** Tracks user subscriptions and their associated plans.

- **Table 4:** Content Table
    - Attributes: content_id, title, content_type, genre, release_date, language, duration_minutes, rating, description, age_rating, added_on
    - Primary Key: content_id
    - **Description:** Stores all available content on the platform, such as movies and TV shows.

- **Table 5:** Watch History Table
    - Attributes: history_id, user_id, content_id, watched_at, progress_percent, completed
    - Primary Key: history_id
    - **Description:** Tracks the viewing history of users.

- **Table 6:** Payments Table
    - Attributes: payment_id, user_id, subscription_id, amount, payment_method, payment_date, status
    - Primary Key: payment_id
    - **Description:** Tracks payment transactions for subscriptions.

- **Table 7:** Content Licensing Table
    - Attributes: license_id, license_start, license_end, region
    - Primary Key: license_id
    - **Description:** Tracks licensing agreements for content on the platform.

- **Table 8:** Devices Table
    - Attributes: device_id, subscription_id, device_name, device_type, os, registered_on
    - Primary Key: device_id
    - **Description:** Tracks devices used by users for accessing the platform.

7.2 Normalization Process

- **1NF (First Normal Form):** Ensure no repeating groups.

  Users Table: Contains atomic attributes like first_name, last_name, email, etc. There are no repeating groups or multiple values in a single column.
  Watch History Table: Each record contains a unique history_id with atomic values like watched_at, progress, and completed.
  Payments Table: Each record has a unique payment_id and atomic attributes like amount and payment_date.

- **2NF (Second Normal Form):** Remove partial dependencies.

  Subscriptions Table: Links subscription_id to user_id and plan_id. Attributes like start_date and end_date depend fully on the subscription_id.
  Payments Table: Links payment_id to user_id and subscription_id. Attributes like amount and payment_date depend fully on the payment_id.

- **3NF (Third Normal Form):** Eliminate transitive dependencies.

  Users Table: Attributes like first_name, last_name, email, and phone_number are directly dependent on user_id. No attribute is indirectly dependent on user_id through another attribute.
  Contents Table: Attributes like title, content_type, release_date, and genre are directly dependent on content_id.
  Subscriptions Table: The plan details (plan_id) are stored in the Subscription Plan table, avoiding redundancy and ensuring no transitive dependencies.

8. SQL QUERIES AND DATABASE OPERATIONS

8.1 Create Table

- Creating Table Users

```
CREATE TABLE users (
user_id INT PRIMARY KEY NOT NULL IDENTITY(2000,1),
first_name VARCHAR(50) NOT NULL,
last_name VARCHAR(50) NOT NULL,
email VARCHAR(100) NOT NULL,
password VARCHAR(255) NOT NULL,
phone_number VARCHAR(15) NOT NULL,
address VARCHAR(255) NOT NULL,
created_at DATETIME NOT NULL,
status VARCHAR(20) NOT NULL
);
```

- Creating Table Subscription Plan

```
Subscription Plan Table
CREATE TABLE subscription_plan (
plan_id INT PRIMARY KEY NOT NULL IDENTITY(7000,1),
plan_name VARCHAR(50) NOT NULL,
monthly_cost DECIMAL(10, 2) NOT NULL,
resolution VARCHAR(20) NOT NULL,
max_devices INT NOT NULL,
created_at DATETIME NOT NULL,
status VARCHAR(20) NOT NULL
);
```

- Creating Table Subscriptions

```
CREATE TABLE subscriptions (
subscription_id INT PRIMARY KEY NOT NULL IDENTITY(9000,1),
user_id INT NOT NULL,
plan_id INT NOT NULL,
start_date DATE NOT NULL,
end_date DATE NOT NULL,
status VARCHAR(20) NOT NULL,
auto_renew TINYINT NOT NULL
);
```

- Creating Table Containts

```
CREATE TABLE contents(
content_id INT PRIMARY KEY NOT NULL IDENTITY(5000,1),
title VARCHAR(150) NOT NULL,
content_type VARCHAR(20) NOT NULL,
genre VARCHAR(50) NOT NULL,
release_date DATE NOT NULL,
language VARCHAR(50) NOT NULL,
duration_minutes INT NOT NULL,
rating DECIMAL(2, 1) NOT NULL,
```

```
description TEXT NOT NULL,
age_rating VARCHAR(5) NOT NULL,
added_on DATETIME NOT NULL
);
```

- Creating Table Watch History

```
CREATE TABLE watch_history (
history_id INT PRIMARY KEY NOT NULL IDENTITY(8000,1),
user_id INT NOT NULL,
content_id INT NOT NULL,
watched_at DATETIME NOT NULL,
progress_percent INT NOT NULL,
completed TINYINT NOT NULL
);
```

- Creating Table Payments

```
CREATE TABLE payments (
payment_id INT PRIMARY KEY NOT NULL IDENTITY(3000,1),
user_id INT NOT NULL,
subscription_id INT NOT NULL,
amount DECIMAL(10, 2) NOT NULL,
payment_method VARCHAR(50) NOT NULL,
payment_date DATETIME NOT NULL,
status VARCHAR(20) NOT NULL
);
```

- Creating Table Content Licencing

```
CREATE TABLE content_licensing (
license_id INT PRIMARY KEY NOT NULL IDENTITY(4000,1),
content_id INT NOT NULL,
license_start DATE NOT NULL,
license_end DATE NOT NULL,
region VARCHAR(100) NOT NULL
);
```

- Creating Table Devices

```
CREATE TABLE devices(
device_id INT PRIMARY KEY NOT NULL IDENTITY(6000,1),
subscription_id INT NOT NULL,
device_name VARCHAR(100) NOT NULL,
device_type VARCHAR(50) NOT NULL,
os VARCHAR(50) NOT NULL,
registered_on DATE NOT NULL
);
```

8.2 Insert Data

- Inserting Data Into Users Table

```
INSERT INTO users (first_name, last_name, email, password, phone_number, address, created_at, status)
VALUES
('Rahul', 'Sharma', 'rahul.sharma@example.com', 'password123', '+919876543210', 'Mumbai, Maharashtra',
'2023-01-01 10:00:00', 'active'),
('Sneha', 'Patil', 'sneha.patil@example.com', 'password123', '+919876543211', 'Bengaluru, Karnataka', '2023-
01-02 11:00:00', 'active'),
('Vikram', 'Rao', 'vikram.rao@example.com', 'password123', '+919876543212', 'Hyderabad, Telangana',
'2023-01-03 12:00:00', 'active'),
('Aisha', 'Khan', 'aisha.khan@example.com', 'password123', '+919876543213', 'Chennai, Tamil Nadu', '2023-
01-04 13:00:00', 'active'),
('Manoj', 'Verma', 'manoj.verma@example.com', 'password123', '+919876543214', 'Connaught Place, Delhi',
'2023-01-05 14:00:00', 'active'),
('Priya', 'Nair', 'priya.nair@example.com', 'password123', '+919876543215', 'Kochi, Kerala', '2023-01-06
15:00:00', 'active'),
('Ravi', 'Iyer', 'ravi.iyer@example.com', 'password123', '+919876543216', 'Kolkata, West Bengal', '2023-01-
07 16:00:00', 'active'),
('Anita', 'Singh', 'anita.singh@example.com', 'password123', '+919876543217', 'Pune, Maharashtra', '2023-
01-08 17:00:00', 'active'),
('Rajesh', 'Gupta', 'rajesh.gupta@example.com', 'password123', '+919876543218', 'Noida, Uttar Pradesh',
'2023-01-09 18:00:00', 'active'),
('Deepika', 'Desai', 'deepika.desai@example.com', 'password123', '+919876543219', 'Ahmedabad, Gujarat',
'2023-01-10 19:00:00', 'active');
```

- Inserting Data Into Subscription Plan Table

```
INSERT INTO subscription_plan (plan_name, monthly_cost, resolution, max_devices, created_at, status)
VALUES
('Basic', 199.00, '480p', 1, '2023-01-01 10:00:00', 'active'),
('Standard', 499.00, '1080p', 2, '2023-01-02 11:00:00', 'active'),
('Premium', 799.00, '4K', 4, '2023-01-03 12:00:00', 'active'),
('Mobile', 149.00, '480p', 1, '2023-01-04 13:00:00', 'active'),
('Family Plan', 999.00, '4K', 5, '2023-01-05 14:00:00', 'active'),
('Annual Basic', 1999.00, '480p', 1, '2023-01-06 15:00:00', 'active'),
('Annual Standard', 4999.00, '1080p', 2, '2023-01-07 16:00:00', 'active'),
('Annual Premium', 7999.00, '4K', 4, '2023-01-08 17:00:00', 'active'),
('Student Plan', 99.00, '720p', 1, '2023-01-09 18:00:00', 'active'),
('Corporate Plan', 1499.00, '1080p', 10, '2023-01-10 19:00:00', 'active');
```

- Inserting Data Into Subscriptions Table

```
INSERT INTO subscriptions (user_id, plan_id, start_date, end_date, status, auto_renew)
VALUES
(2001, 7001, '2023-01-01', '2023-02-01', 'active', 1),
(2002, 7002, '2023-01-15', '2023-02-15', 'active', 1),
(2003, 7003, '2023-02-01', '2023-03-01', 'active', 1),
(2004, 7004, '2023-03-01', '2023-04-01', 'active', 0),
(2005, 7005, '2023-04-01', '2023-05-01', 'active', 1),
(2006, 7006, '2023-05-01', '2024-05-01', 'active', 0),
(2007, 7007, '2023-06-01', '2024-06-01', 'active', 1),
(2008, 7008, '2023-07-01', '2024-07-01', 'active', 1),
(2009, 7009, '2023-08-01', '2023-09-01', 'active', 0),
```

- Inserting Data Into Content Table

```
INSERT INTO contents (title, content_type, genre, release_date, language, duration_minutes, rating, description, age_rating, added_on)
VALUES
('3 Idiots', 'Movie', 'Comedy, Drama', '2009-12-25', 'Hindi', 170, 8.4, 'Three friends navigate life at an engineering college, challenging the traditional education system.', 'U', '2023-01-01 12:00:00'),
('Sacred Games', 'Series', 'Crime, Thriller', '2018-07-06', 'Hindi', 50, 8.6, 'A Mumbai police officer uncovers a conspiracy while chasing a notorious gangster.', 'A', '2023-01-02 13:00:00'),
('Bahubali: The Beginning', 'Movie', 'Action, Drama', '2015-07-10', 'Telugu', 159, 8.1, 'A prince overcomes obstacles to reclaim his kingdom and legacy.', 'U/A', '2023-01-03 14:00:00'),
('KGF: Chapter 1', 'Movie', 'Action, Crime', '2018-12-21', 'Kannada', 156, 8.2, 'A power-hungry man rises from the slums to rule the gold mafia in Kolar.', 'U/A', '2023-01-04 15:00:00'),
('Paatal Lok', 'Series', 'Crime, Thriller', '2020-05-15', 'Hindi', 45, 7.8, 'A cynical inspector investigates a high-profile case that takes him to the dark underbelly of society.', 'A', '2023-01-05 16:00:00'),
('Dangal', 'Movie', 'Biography, Drama', '2016-12-23', 'Hindi', 161, 8.3, 'A former wrestler trains his daughters to become world-class wrestlers.', 'U', '2023-01-06 17:00:00'),
('Jallikattu', 'Movie', 'Thriller, Drama', '2019-09-06', 'Malayalam', 95, 7.7, 'A village descends into chaos when a buffalo escapes and runs wild.', 'A', '2023-01-07 18:00:00'),
('The Family Man', 'Series', 'Action, Thriller', '2019-09-20', 'Hindi', 50, 8.5, 'A middle-class man works secretly as a spy for an intelligence agency.', 'A', '2023-01-08 19:00:00'),
('Super Deluxe', 'Movie', 'Drama, Thriller', '2019-03-29', 'Tamil', 176, 8.4, 'Four people experience life-changing events on the same day.', 'A', '2023-01-09 20:00:00'),
('Pushpa: The Rise', 'Movie', 'Action, Drama', '2021-12-17', 'Telugu', 179, 7.6, 'A laborer rises through the ranks of a smuggling syndicate in the forests of India.', 'U/A', '2023-01-10 21:00:00');
```

- Inserting Data Into Watch History Table

```
INSERT INTO watch_history (user_id, content_id, watched_at, progress_percent, completed)
VALUES
(2001, 5001, '2023-01-01 18:30:00', 100, 1),
(2002, 5002, '2023-01-15 19:45:00', 50, 0),
(2003, 5003, '2023-02-01 20:20:00', 100, 1),
(2004, 5004, '2023-03-01 21:10:00', 75, 0),
(2005, 5005, '2023-04-01 22:30:00', 100, 1),
(2006, 5006, '2023-05-01 23:40:00', 100, 1),
(2007, 5007, '2023-06-01 00:50:00', 20, 0),
(2008, 5008, '2023-07-01 01:15:00', 100, 1),
(2009, 5009, '2023-08-01 02:05:00', 100, 1)
(2010, 5010, '2023-09-01 03:25:00', 45, 0);
```

- Inserting Data Into Payments Table

```
INSERT INTO payments (user_id, subscription_id, amount, payment_method, payment_date, status)
VALUES
(2001, 9001, 199.00, 'Credit Card', '2023-01-01 10:30:00', 'successful'),
(2002, 9002, 499.00, 'Debit Card', '2023-01-15 11:45:00', 'successful'),
(2003, 9003, 799.00, 'UPI', '2023-02-01 12:20:00', 'successful'),
(2004, 9004, 149.00, 'Net Banking', '2023-03-01 13:10:00', 'successful'),
(2005, 9005, 999.00, 'Credit Card', '2023-04-01 14:30:00', 'successful'),
(2006, 9006, 1999.00, 'Debit Card', '2023-05-01 15:40:00', 'successful'),
(2007, 9007, 4999.00, 'UPI', '2023-06-01 16:50:00', 'successful'),
(2008, 9008, 7999.00, 'Net Banking', '2023-07-01 17:15:00', 'successful'),
(2009, 9009, 99.00, 'Credit Card', '2023-08-01 18:05:00', 'successful'),
(2010, 9010, 1499.00, 'UPI', '2023-09-01 19:25:00', 'successful');
```

- Inserting Data Into Content Licensing Table

```
INSERT INTO content_licensing (license_start, license_end, region)
VALUES
('2023-01-01', '2025-01-01', 'India'),
('2023-01-15', '2025-01-15', 'India, Pakistan, Bangladesh'),
('2023-02-01', '2024-02-01', 'India, Sri Lanka, Nepal'),
('2023-03-01', '2026-03-01', 'India'),
('2023-04-01', '2024-04-01', 'India, Nepal'),
('2023-05-01', '2025-05-01', 'India, Bhutan'),
('2023-06-01', '2024-06-01', 'India, Sri Lanka, Bangladesh'),
('2023-07-01', '2025-07-01', 'India, Pakistan'),
('2023-08-01', '2026-08-01', 'India'),
('2023-09-01', '2025-09-01', 'India, Nepal, Bhutan');
```

- Inserting Data Into Devices Table

```
INSERT INTO devices (subscription_id, device_name, device_type, os, registered_on)
VALUES
('iPhone 12', 'Smartphone', 'iOS', '2023-01-01'),
('Galaxy A51', 'Smartphone', 'Android', '2023-01-02'),
('HP pavillion', 'Laptop', 'Windows 10', '2023-01-03'),
('MacBook Pro', 'Laptop', 'macOS', '2023-01-04'),
('Fire Stick', 'Streaming Device', 'Fire OS', '2023-01-05'),
('Samsung TV', 'Smart TV', 'Tizen OS', '2023-01-06'),
('iPad', 'Tablet', 'iPadOS', '2023-01-07'),
('Google Pixel', 'Smartphone', 'Android', '2023-01-08'),
('Lenovo ideapad', 'Laptop', 'Linux', '2023-01-09'),
('Deepika Roku', 'Streaming Device', 'Roku OS', '2023-01-10');
```

8.3 Queries for Data Retrieval

8.3.1 Function

- Non Parametrized Function To fetch first name of all the users

```
create function dbo.getfirstname6()
returns table
as
return  (Select first_name from dbo.users);
GO

select*from dbo.getfirstname6()
```

- Parametrized Function to get the list of subscription plans whose monthly cost is less than or equal to the given amount

```
create function dbo.getsubplans(@cost int)
returns table
as
return  (Select plan_name, monthly_cost, max_devices
from dbo.subscription_plan
where monthly_cost = @cost or monthly_cost<@cost);
GO
```

```
select*from dbo.getsubplans(500)
```

8.3.2 Stored Procedures

- Non Parametrized SP to get the details of all the users

```
create proc spgetusersdetails
as
begin
select*from dbo.users
end
go

EXEC  spgetusersdetails
```

- Parametrized SP to get the contents and its details as per the given language

```
create proc spgetcontents @Language varchar(40)
as
begin
select title, content_type, genre, language, duration_minutes, rating, description from
dbo.contents
where language like @Language;
end
go

 EXEC spgetcontents @Language='Hindi'
```

- Try Catch

```
CREATE PROCEDURE GetUsersWithWatchHistory
   @HistoryId INT
AS
BEGIN
   BEGIN TRY
      SELECT
         U.user_id,
         CONCAT(U.first_name, ' ', U.last_name) AS full_name,
         U.email,
         S.status,
         WH.history_id AS watch_history_id,
         WH.content_id,
         WH.watched_at,
         WH.progress_percent,
         WH.completed
      FROM
         users U
      INNER JOIN subscriptions S ON U.user_id = S.user_id
      INNER JOIN watch_history WH ON U.user_id = WH.user_id
      WHERE
         WH.history_id = @HistoryId;

      PRINT 'Query executed successfully.';
   END TRY
```

```
    BEGIN CATCH

        DECLARE @ErrorMessage NVARCHAR(4000) = ERROR_MESSAGE();
        DECLARE @ErrorSeverity INT = ERROR_SEVERITY();
        DECLARE @ErrorState INT = ERROR_STATE();

        PRINT @ErrorMessage;
        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
    END CATCH
END;

EXEC GetUsersWithWatchHistory @HistoryId = 8002
```

8.3.3 Cursor

- To show the details of all the subscription plans

```
Declare @Planname Varchar(50), @Cost decimal, @Resolution varchar(40), @Maxdevices
int

Declare SubscriptionPlanCursor Cursor for
select plan_name, monthly_cost, resolution, max_devices from dbo.subscription_plan

open SubscriptionPlanCursor;

Fetch next from SubscriptionPlanCursor into @Planname, @Cost, @Resolution,
@Maxdevices ;
while @@FETCH_STATUS=0
begin
print 'Plan Name: ' + Cast(@Planname as varchar(40)) +
    ', Monthly Cost: ' + Cast(@Cost as varchar(10))+
        ', Resolution: ' + Cast(@Resolution as varchar(10))+
        ', Maximum Devices: ' + Cast(@Maxdevices as varchar(10));

fetch next from SubscriptionPlanCursor into @Planname, @Cost, @Resolution,
@Maxdevices ;
end
close SubscriptionPlanCursor;

Deallocate SubscriptionPlanCursor;
```

8.3.4 Normal Quaries

- Retrieve All Users and Their Subscription Details

```
SELECT sp.plan_name, COUNT(s.subscription_id) AS active_subscriptions
FROM [subscription_plan] sp
JOIN subscriptions s ON sp.plan_id = s.plan_id
WHERE s.status = 'Active'
GROUP BY sp.plan_name;
```

- Find Users with Active Subscriptions

```
SELECT u.user_id, u.first_name, u.last_name, sp.plan_name
FROM users u
JOIN subscriptions s ON u.user_id = s.user_id
JOIN [subscription_plan] sp ON s.plan_id = sp.plan_id
WHERE s.status = 'Active';
```

- Retrieve Users' Watch History

```
SELECT u.user_id, u.first_name, u.last_name, c.title AS content_title, wh.watched_at,
wh.progress_percent, wh.completed
FROM users u
JOIN watch_history wh ON u.user_id = wh.user_id
JOIN contents c ON wh.content_id = c.content_id
ORDER BY u.user_id;
```

- Get Total Number of Active Subscriptions per Plan

```
SELECT sp.plan_name, COUNT(s.subscription_id) AS active_subscriptions
FROM [subscription_Plan] sp
JOIN subscriptions s ON sp.plan_id = s.plan_id
WHERE s.status = 'Active'
GROUP BY sp.plan_name;
```

- List Content Watched by Each User in the Last 3 year

```
SELECT u.user_id, u.first_name, u.last_name, c.title AS content_title, wh.watched_at
FROM users u
JOIN watch_history wh ON u.user_id = wh.user_id
JOIN contents c ON wh.content_id = c.content_id
WHERE wh.watched_at >= DATEADD(MONTH, -36, GETDATE())
ORDER BY wh.watched_at DESC;
```

- Calculate Total Revenue from All Payments

```
SELECT SUM(amount) AS total_revenue
FROM payments
WHERE status = 'successful';
```

- Get Subscription Renewal Information for Users

```
SELECT u.user_id, u.first_name, u.last_name, sp.plan_name, s.end_date
FROM users u
JOIN subscriptions s ON u.user_id = s.user_id
JOIN [subscription_plan] sp ON s.plan_id = sp.plan_id
WHERE s.auto_renew = 1;
```

9. CONCLUSION

This project aims at designing and implementing a strong relational database management system (RDBMS) of an online streaming platform employing Microsoft SQL Server (MSSQL). By doing this, we were able to show what the fundamental features are to mimic a platform similar to Netflix or Amazon Prime Video, ranging from user administration, subscription tracking, and payment processing to personalized content recommendation.

The design of the database was highly pragmatic and specific to the scalability requirements of the application that will use it, able to service millions of users and thousands of content items. Normalization was effective in removing data inconsistencies and the elimination of data redundancy, the incorporation of SQL Queries, stored procedures as well as functions made the system operational.

It also focused on practical use cases like handling licenses for content from a particular region and watching history for users and recommendations. MSSQL has been a good platform for managing vast data with a rich set of sophisticated built-in attributes of indexes and implementing queries with the optimum level of efficiency.

In general, the project ties theoretical concepts underlying databases with real-world applications of such theories. This case shows how an anchored database can propel a streamlined user-oriented online service while providing scalability and dependability. This proof of concept may be used as a starting point for constructing incrementally more complex systems as the need arises and as new problems and developments in the domain of streaming services present themselves.

REFERENCES

[1] https://arxiv.org/pdf/2106.00932

[2] https://www.sciencedirect.com/topics/computer-science/streaming service.

[3] https://www.actowizsolutions.com/case-study-insights-ott-platform-data-scraping.php

https://github.com/rahulkumariit/Movies-on-OTT-platform