The Video Streaming Database
MEQ-15: Database Design Project

Winter 2020

Vanier college

Designed and Developed by Aleksandr Mitrofanov

TABLE OF CONTENTS

INTRODUCTION

This document provides an overview of the Video Streaming database developed by the Aleksandr Mitrofanov for the final project in Database Design Project class at Vanier college. The following sections outline the database application, lists the entities that the database stores and provides some SQL queries that were used to build the database. The document concludes with a reflection of the project.

DATABASE APPLICATION

The VSDB will contain a set of movies and TV Shows. Each movie and tv show will contain at least a description, rating (see ratings symbol table), rating components (zero or more of the movie rating components – see rating component table), genre, score (1-10, based on user scores), cover picture (path to the jpg file with the movie/show poster), actors, directors, release year and cost to distribute the show. TV Shows will be split in multiple seasons, each season having multiple episodes. Each Movie and each TV Show episode will store the path to files where the show file is stored for each resolution (thus we have multiple resolutions for each move). Resolutions may vary from 480p, 720p, 1080p, 1440p, 4K and 8K. Not all shows have all resolutions, but all shows have at least one. The VSDB will contain also all the information needed for the video-streaming web platform clients to login to their account. Part of the clients' accounts it will store at least the login, client name, client address, age, address, payment option (credit card, debit account, paypal, amazon pay and moneygram). Note that each payment option may need different information from the client. Payments will be made monthly. If a user didn't have the payment up to date his subscription will be marked as inactive. For each account there may be 1 to 5 profiles created. Each profile will contain a profile name and the icon name associated with that profile (this icon will be displayed on the webpage). The VSDB will store for each movie/show a count on how many distinct users (profiles) that watched the movie/show. Web clients will be able to search movies/shows in the database based on their name, genre, year and score. Beside web clients the VSDB will also contain admin users that are able to maintain the movie/show database. These admin users will login through another web application where they'll be able to search movies/shows by ID and edit these. Of course, they'll also be able to insert and delete movie/shows. For audit purposes the admin users will be able to view who last modified/created/deleted a movie/show and when. Accounting team will use another web application in order to view deposits and costs associated with the Video Streaming platform. People from the accounting team will be identified based on their login name and password. These users will be able to generate and view profit and loss report generated each month (total distribution cost for the shows added in the given month and total client's payments that month). Management team will have access to a special reporting application that will give them different reports including: given a movie display viewing history, given a client return access history, given a movie return its profitability.

SYSTEM DESCRIPTION AND FUNCTIONALITIES

The DBMS will be a database with a user interface that will allow VSDB data to be viewed, stored and updated using SQL queries to the database. These will allow the store to maintain and have easy and efficient access to accurate and up-to-date records relevant to the VSDB information. The system will facilitate the VSDB by updating database tables. It will also allow the VSDB to easily look for a client and all sort of queries by using complex queries.

SYSTEM FUNCTIONALITY

The system is mainly a DBMS build around a SQL database and a Web-based GUI. The system has several main functions and those will be delivered.

- The system will let a user connect to the database with the mean of a username and password

-The system will let a user search or see a movie information

- The system will allow a user to subscribe to watch the content.

- The system will allow a user to update their personal customer information

- The system will allow the administrator to add and update a movies ant tv shows to the database

- The system will allow the accountant to generate and view profit and loss report generated each month

- The system will allow the manager will have access to a special reporting application that will give them different reports

- The system does not perform any form of transactions. The payment information is only saved on the DB.

Database Design

Relational data model

Movies (Title: string, Release year: integer, Distribute cost: integer, Score: float,

Description: string, Cover picture: string, Date of distribution: string)

TV Shows (Title: string, Release year: integer, Distribute cost: integer, Score: float, Description: string, Cover picture: string, Date of distribution: string)

Genres (Title: string)

Rating components (Title: string, Description: string)

Rating symbols (Rating symbol: string, Description: string)

Actors (Name: string, Date of birth: date)

Directors (Name: string)

Resolutions (Title: string)

Client accounts (Name: string, Age: integer, Address: string, Username: string,

Password: string, Status: boolean)

Profiles (Name: string, Icon path: string)

Payments (Payment date: date, Payment information: string, Payment amount: decimal)

Subscriptions (Title: string, Price: float)

Administrators (Login: string, Password: string)

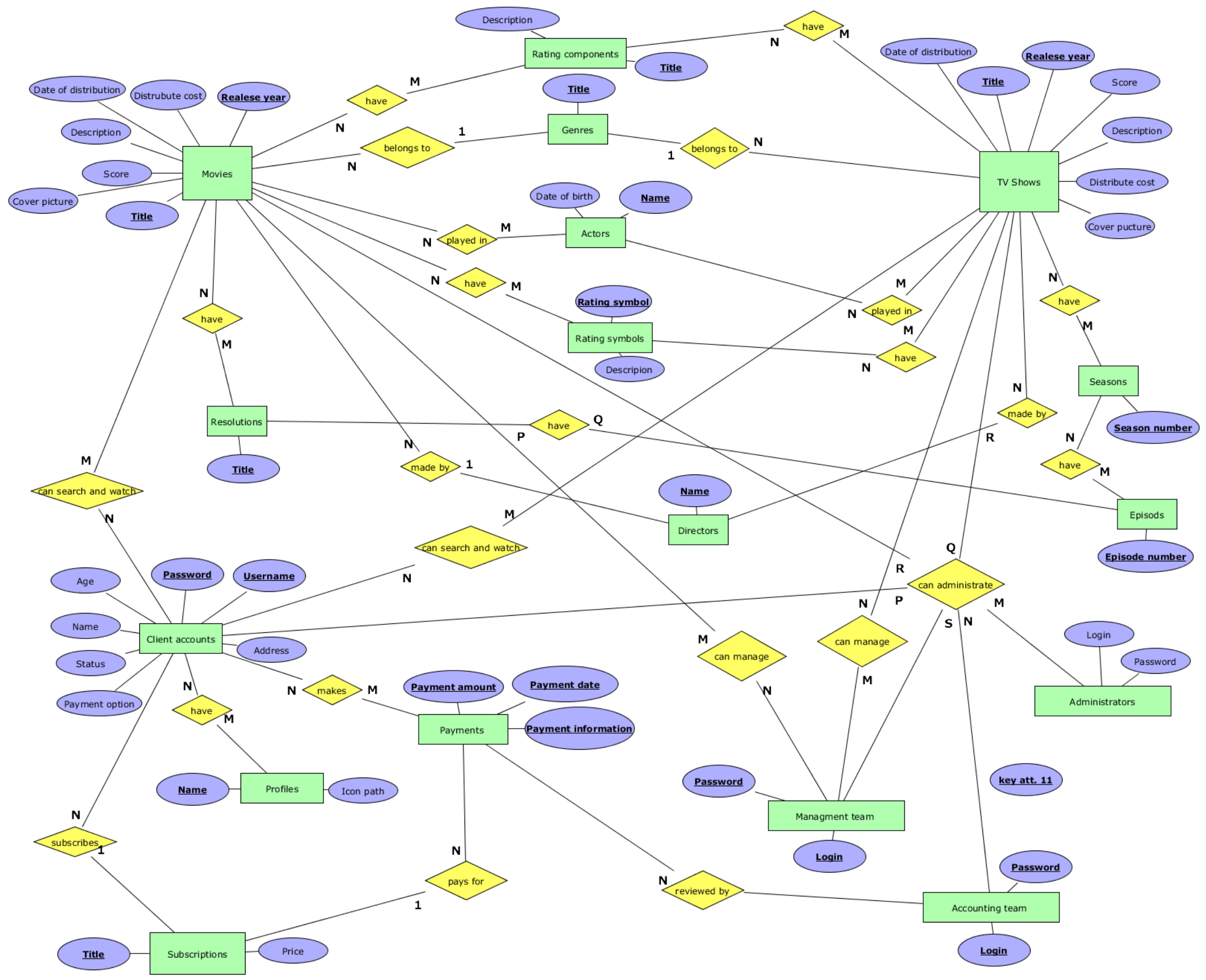
Accounting team (Login: string, Password: string)

Management team (Login: string, Password: string)

Seasons (Season number: Integer)

Episodes (Episode number: Integer)

ER Diagram


Description
Rating components
Title
have
Date of distribution
Realese year
Title
Score
Description
Distribute cost
Cover pucture
TV Shows
Date of distribution
Distrubute cost
Realese year
Description
Score
Cover picture
Title
Movies
have
belongs to
Title
Genres
belongs to
Date of birth
Name
Actors
played in
have
Rating symbol
Rating symbols
Descripion
played in
have
have
Seasons
Season number
made by
have
Episods
Episode number
Resolutions
Title
have
made by
Name
Directors
can search and watch
can search and watch
can administrate
Age
Password
Username
Name
Client accounts
Address
Status
Payment option
have
makes
Payment amount
Payment date
Payments
Payment information
Name
Profiles
Icon path
can manage
can manage
Login
Password
Administrators
Password
Managment team
Login
key att. 11
subscribes
pays for
reviewed by
Password
Accounting team
Login
Title
Subscriptions
Price
M
N
1
N
1
N
M
N
M
N
M
N
N
M
M
N
N
M
N
M
N
Q
P
N
1
R
N
M
M
N
Q
R
P
N
S
N
M
M
N
M
N
N
N
M
N
M
N
1
N
1
N

DBMS Selection.

I’ve chose MySQL because it has following benefits:

- Scalability and Flexibility
- High Performance
- High Availability
- Robust Transactional Support
- Web and Data Warehouse Strengths
- Strong Data Protection
- Comprehensive Application Development
- Management Ease
- Open Source Freedom and 24 x 7 Support
- Lowest Total Cost of Ownership

NORMALIZATION

Upon designing the ER Diagram and Relational Data Model, the process of Normalization was adopted to assess any deficiencies in the derived tables. The goal was to develop third normal form for all tables. In order to achieve third normal form, each table must be free from multi values attributes and transient dependencies and must have full functional dependencies. This section outlines all the functional dependencies for the final relations.

Movies_and_TV_Shows

- ID → Title
- ID → Realese_year

ID → Distribute_cost

- ID → Date_of_distribution
- ID → Score
- ID → Description

-ID → Cover_Picture

- ID → Director

-ID → Genre

Attributes: ID, Title, Realese_year, Distribute_cost, Date_of_distribution, Score, Description,

Cover_Picture , Director, Genre .

The Movies_and_TV_Shows table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Rating_components

-ID→Description

The Rating_components table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Description

Rating_symbols

-ID→Rating_symbol

-ID→Description

The Rating_symbols table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Rating_symbol,Description

Actors

-ID→Name

-ID→Date_of_birth

The Actors table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Name, Date_of_birth

Resolutions

-ID→Title

The Resolutions table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID,T itle

Client_accounts

- ID → Name

- ID → Age

ID → Address

- ID → Payment_option

- ID → Username

- ID → Password

-ID → Status

The Client_accounts table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Name, Age, Address, Payment_option, Username, Password, Status

Profiles

- ID → Name
- ID → Icon_path
- ID → Client_ID

The Profiles table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Name, Icon_path, Client_ID

Foreign key: Client_ID

Subscriptions

- ID → Title
- ID → Price

The Subscriptions table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Title ,Price

Payments

- ID → Payment_date
- ID → Payment_information

ID → Payment_amount

- ID → Client_ID
- ID → Subcsription_ID

The Payments table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Payment_date, Payment_information, Payment_amount, Client_ID, Subcsription_ID

Foreign key: Client_ID, Subcsription_ID

Roles

- ID → Title

The Roles table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID ,Title

Stuff_accounts

- ID → Name
- ID → Password

ID → Role_ID

The Stuff_accounts table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Name, Password, Role_ID

Foreign key: Role_ID

Movies_resolutions

- Movie_ID,Resolution_ID → File_Path

The Movies_resolutions table is in third normal form. All of the attributes are uniquely identified by composite key Movie_ID,Resolution_ID and no transient dependencies exist.

Attributes: Movie_ID, Resolutions_ID, File_path

Foreign key: Resolutions_ID, Movie_ID

Movies_and_TV_Shows_actors

It has only composite primary key Movies_and_TV_Shows_ID, Actor_ID. The Movies_and_TV_Shows_actors table is in third normal form.

Attributes: Movies_and_TV_Shows_ID, Actor_ID

Foreign key: Movies_and_TV_Shows_ID, Actor_ID

Movies_and_TV_Shows_symbols

It has only composite primary key Movies_and_TV_Shows_ID, Rating_symbol_ID. The Movies_and_TV_Shows_symbols table is in third normal form.

Attributes: Movies_and_TV_Shows_ID, Rating_symbol_ID

Foreign key: Movies_and_TV_Shows_ID, Rating_symbol_ID

Movies_and_TV_Shows_rating_components

It has only composite primary key Movie_ID, Rating_components_ID. The Movies_and_TV_Shows_rating_components table is in third normal form.

Attributes: Movie_ID, Rating_components_ID

Foreign key: Movie_ID, Rating_components_ID

TV_Show_seasons

- ID → TV_Show_ID
- ID → Season_number

The TV_Show_seasons table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, TV_Show_ID, Season_number

Foreign key: TV_Show_ID

TV_shows_episodes

- ID → Episode_Number
- ID → Season_ID

The TV_shows_episodes table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Episode_Number, Season_ID

Foreign key: TV_Show_ID

TV_Show_resolutions

- Resolutions_ID, Episode_ID→ File_Path

The TV_Show_resolutions table is in third normal form. All of the attributes are uniquely identified by composite key Resolutions_ID, Episode_ID and no transient dependencies exist.

Attributes: Resolutions_ID, Episode_ID, File_path

Foreign key: Resolutions_ID, Episode_ID

Movies_sessions

- ID → Movie_ID
- ID → Client_ID

ID → Watched_at

The Movies_sessions table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Movie_ID, Client_ID, Watched_at

Foreign key: Movie_ID, Client_ID

TV_Show_sessions

- ID → TV_Show_episode_ID
- ID → Client_ID

ID → Watched_at

The TV_Show_sessions table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, TV_Show_episode_ID, Client_ID, Watched_at

Foreign key: TV_Show_episode_ID, Client_ID

clients_logins

- ID → Client_ID
- ID → Logged_in

ID → Logged_out

The clients_logins table is in third normal form. All of the attributes are uniquely identified by the ID and no transient dependencies exist.

Attributes: ID, Client_ID, Logged_in, Logged_out

Foreign key: Client_ID

data_history

- ID, revision→ User
- ID, revision→ action
- ID, revision→ dt_datetime

The data_history table is in third normal form. All of the attributes are uniquely identified by the ID, revision and no transient dependencies exist.

Attributes: ID, revision,User,action,dt_datetime

CONSTRAINTS

The three main types of integrity constraints are: Key, Entity integrity and Referential integrity constraints.

Movies_and_TV_Shows

Key: ID is the primary key and must hold "unique" values. Values cannot be repeated for ID.
Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Title, Realese_year, Distribute_cost, Date_of_distribution, Score, Description,Cover_Picture , Director, Genre can't be null.

Referential Integrity: None.

No foreign keys in Movies_and_TV_Shows table.

Rating_components

Key: ID is the primary key and must hold "unique" values. Values cannot be repeated for ID.
Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Description can't be null.

Referential Integrity: None.

No foreign keys in Rating_components table.

Rating_symbols

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID.
Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Rating_symbol, Description can’t be null.

Referential Integrity: None.

No foreign keys in Rating_symbols table.

Actors

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID.
Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Name, Date_of_birth can’t be null.

Referential Integrity: None.

No foreign keys in Actors table.

Resolutions

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID.
Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Title can’t be null.

Referential Integrity: None.

No foreign keys in Resolutions table.

Client_accounts

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID.
Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Name, Age, Address, Payment_option, Username, Password, Status can’t be null.

Referential Integrity: None.

No foreign keys in Client_accounts table.

Profiles

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Name, Icon_path, Client_ID can’t be null.

Referential Integrity: Client_ID references Client_accounts table’s ID. Client_ID cannot be NULL and must be a value that already exists in the parent table.

Subscriptions

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Title, Price can’t be null.

Referential Integrity: None.

No foreign keys in Subscriptions table.

Payments

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Payment_date, Payment_information, Payment_amount,Client_ID, Subcsription_ID can’t be null.

Referential Integrity: Client_ID references Client_accounts table’s ID, Subcsription_ID references Subscriptions’s ID. Client_id, Subcsription_ID cannot be NULL and must be a value that already exists in the parent table.

Roles

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Title can’t be null.

Referential Integrity: None.

No foreign keys in Subscriptions table.

Stuff_accounts

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Name, Password, Role_ID can’t be null.

Referential Integrity: Role_ID references Roles table’s ID. Role_ID cannot be NULL and must be a value that already exists in the parent table.

Movies_resolutions

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Movie_ID, Resolutions_ID, File_path can’t be null.

Referential Integrity: Resolutions_ID references Resolutions table’s ID, Movie_ID references Movies_and_TV_Shows table’s ID. Resolutions_ID , Movie_ID cannot be NULL and must be a value that already exists in the parent table.

Movies_and_TV_Shows_actors

Key: Movies_and_TV_Shows_ID and Actor_ID is the composite primary key and must hold “unique” values. Values cannot be repeated for ID.

Entity Integrity: Movies_and_TV_Shows_ID and Actor_ID is the primary key and hence cannot be a NULL value.

Referential Integrity: Movies_and_TV_Shows_ID references Movies_and_TV_Shows table’s ID, Actor_ID references Actors table’s ID. Movies_and_TV_Shows_ID and Actor_ID cannot be NULL and must be a value that already exists in the parent table.

Movies_and_TV_Shows_symbols

Key: Movies_and_TV_Shows_ID and Rating_symbol_ID is the composite primary key and must hold “unique” values. Values cannot be repeated for ID.

Entity Integrity: Movies_and_TV_Shows_ID and Rating_symbol_ID is the primary key and hence cannot be a NULL value.

Referential Integrity: Movies_and_TV_Shows_ID references Movies_and_TV_Shows table’s ID, Rating_symbol_ID references Rating_symbols table’s ID. Movies_and_TV_Shows_ID and Rating_symbol_ID cannot be NULL and must be a value that already exists in the parent table.

Movies_and_TV_Shows_rating_components

Key: Movie_ID and Rating_components_ID is the composite primary key and must hold "unique" values. Values cannot be repeated for ID.

Entity Integrity: Movie_ID and Rating_components_ID is the primary key and hence cannot be a NULL value.

Referential Integrity: Movie_ID references Movies_and_TV_Shows table's ID, Rating_components_ID references Rating_components table's ID. Movie_ID and Rating_components_ID cannot be NULL and must be a value that already exists in the parent table.

TV_Show_seasons

Key: ID is the primary key and must hold "unique" values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

TV_Show_ID, Season_number can't be null.

Referential Integrity: TV_Show_ID references Movies_and_TV_Shows table's ID. TV_Show_ID cannot be NULL and must be a value that already exists in the parent table.

TV_shows_episodes

Key: ID is the primary key and must hold "unique" values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Episode_Number, Season_ID can't be null.

Referential Integrity: Season_ID references TV_Show_seasons table's ID. Season_ID cannot be NULL and must be a value that already exists in the parent table.

TV_Show_resolutions

Key: Resolutions_ID and Episode_ID is the composite primary key and must hold "unique" values. Values cannot be repeated for ID.

Entity Integrity: File_path cannot be a NULL value.

Referential Integrity: Resolutions_ID references Resolutions table's ID, Episode_ID references TV_shows_episods table's ID. Resolutions_ID and Episode_ID cannot be NULL and must be a value that already exists in the parent table.

Movies_sessions

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Movie_ID, Client_ID, Watched_at can’t be null.

Referential Integrity: Movie_ID references Movies_and_TV_Shows table’s ID, Client_ID references Profiles table’s ID. Movie_ID and Client_ID cannot be NULL and must be a value that already exists in the parent table.

TV_Show_sessions

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

TV_Show_episode_ID,Client_ID, Watched_at can’t be null.

Referential Integrity: TV_Show_episode_ID references TV_shows_episods table’s ID, Client_ID references Profiles table’s ID. TV_shows_episods and Client_ID cannot be NULL and must be a value that already exists in the parent table.

clients_logins

Key: ID is the primary key and must hold “unique” values. Values cannot be repeated for ID. Entity Integrity: ID is the primary key and hence cannot be a NULL value.

Client_ID, Logged_in can’t be null.

Referential Integrity: Client_ID references client_accounts table’s ID. Client_ID cannot be NULL and must be a value that already exists in the parent table.

data_history

Key: ID and revision is the composite primary key and must hold “unique” values. Values cannot be repeated for ID and revision.

Entity Integrity: User,action, dt_datetime cannot be a NULL value.

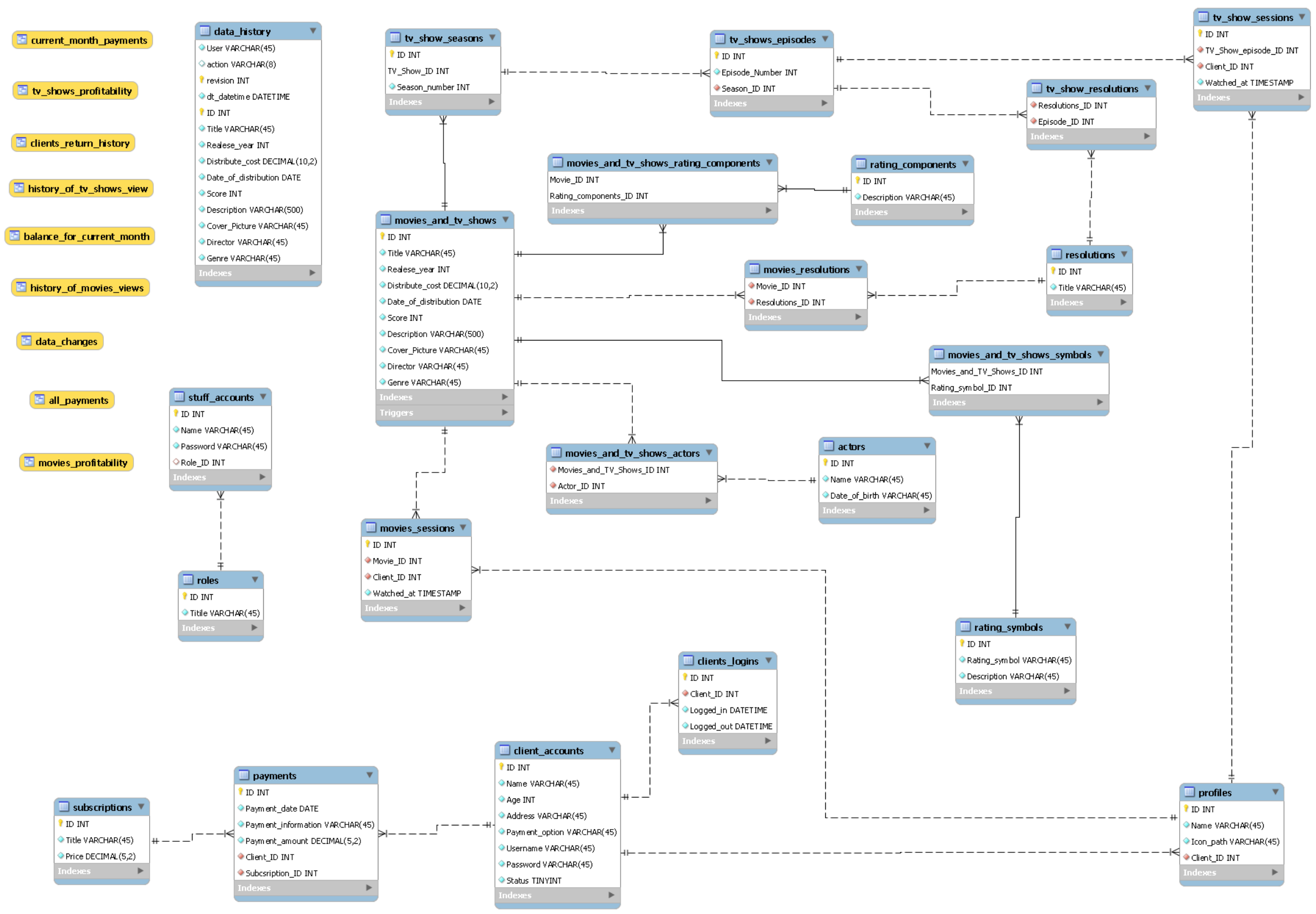
Referential Integrity: None.

No foreign keys in Subscriptions table.

RELATIONSHIPS

- One Movie_and_TV_Show can have one or many Resolutions
- One TV_Show_Episode can have one or many Resolutions
- One Movie_and_TV_Show can have 0 or many Rating_components
- One Movie_and_TV_Show can have one Rating_symbol
- One Movie_and_TV_Show can have one or many Actors
- One Movie_and_TV_Show can have one Director
- One Client's account can have one or many clients_logins
- One Client's account can have 0 or many payments
- One Client's account can have one or many clients_logins
- One Client's account can have 0 or many subscriptions
- One Client's account can have one or many profiles
- One Profile can have 0 or many Movies_sessions
- One Profile can have 0 or many TV_Show_sessions
- One subscription can have 0 or many Payments
- One Staff_account can have only one Role.
- One Role can have 0 or many Stuff_accounts
- One Movie_and_TV_Show can have one or many TV_Show_seasons
- One TV_Show_season can have one or many TV_Show _episodes

Physical model



Implementation and loading

DBMS configuration parameters

```
abort-slave-event-count           0
allow-suspicious-udfs             FALSE
archive                           ON
auto-increment-increment          1
auto-increment-offset             1
autocommit                        TRUE
automatic-sp-privileges           TRUE
back-log                          80
basedir                           /home/jon/bin/mysql-5.6/
...
tmpdir                            /tmp
transaction-alloc-block-size      8192
transaction-isolation             REPEATABLE-READ
transaction-prealloc-size         4096
transaction-read-only             FALSE
updatable-views-with-limit        YES
verbose                           TRUE
wait-timeout                      28800
```

Create database and tables

```
CREATE SCHEMA IF NOT EXISTS `VSDB` DEFAULT CHARACTER SET utf8;
USE `VSDB`;

CREATE TABLE IF NOT EXISTS `Movies_and_TV_Shows` (
  `ID` INT NOT NULL,  `Title` VARCHAR(45) NOT NULL,  `Realese_year` INT NOT NULL,
  `Distribute_cost` DECIMAL(10,2) NOT NULL,  `Date_of_distribution` DATE NOT NULL,
  `Score` INT NOT NULL,  `Description` VARCHAR(500) NOT NULL,  `Cover_Picture` VARCHAR(45) NOT NULL,
  `Director` VARCHAR(45) NOT NULL,  `Genre` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`));

CREATE TABLE IF NOT EXISTS `Rating_components` (
  `ID` INT NOT NULL,  `Description` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`),  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE);

CREATE TABLE IF NOT EXISTS `Rating_symbols` (
  `ID` INT NOT NULL,  `Rating_symbol` VARCHAR(45) NOT NULL,  `Description` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`),  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE);

CREATE TABLE IF NOT EXISTS `Actors` (
  `ID` INT NOT NULL,  `Name` VARCHAR(45) NOT NULL,  `Date_of_birth` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`),  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE);

CREATE TABLE IF NOT EXISTS `Resolutions` (
  `ID` INT NOT NULL,  `Title` VARCHAR(45) NOT NULL,  PRIMARY KEY (`ID`),
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE);

CREATE TABLE IF NOT EXISTS `Client_accounts` (
  `ID` INT NOT NULL,  `Name` VARCHAR(45) NOT NULL,  `Age` INT NOT NULL,  `Address` VARCHAR(45) NOT
NULL,  `Payment_option` VARCHAR(45) NOT NULL,  `Username` VARCHAR(45) NOT NULL,
  `Password` VARCHAR(45) NOT NULL,  `Status` TINYINT NOT NULL,  PRIMARY KEY (`ID`),
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE,  UNIQUE INDEX `Username_UNIQUE` (`Username` ASC)
VISIBLE);
```

```
CREATE TABLE IF NOT EXISTS `Profiles` (
  `ID` INT NOT NULL,  `Name` VARCHAR(45) NOT NULL,  `Icon_path` VARCHAR(45) NOT NULL,
  `Client_ID` INT NOT NULL,  PRIMARY KEY (`ID`),  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE,
  INDEX `Clients_idx` (`Client_ID` ASC) VISIBLE,  CONSTRAINT `FK_Profiles_Clients`
    FOREIGN KEY (`Client_ID`)    REFERENCES `Client_accounts` (`ID`)
    ON DELETE CASCADE    ON UPDATE CASCADE);

CREATE TABLE IF NOT EXISTS `Subscriptions` (
  `ID` INT NOT NULL,  `Title` VARCHAR(45) NOT NULL,  `Price` DECIMAL(5,2) NOT NULL,
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE,  PRIMARY KEY (`ID`));

CREATE TABLE IF NOT EXISTS `Payments` (
  `ID` INT NOT NULL,  `Payment_date` DATE NOT NULL,  `Payment_information` VARCHAR(45) NOT NULL,
  `Payment_amount` DECIMAL(5,2) NOT NULL,  `Client_ID` INT NOT NULL,  `Subcsription_ID` INT NOT NULL,
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE,  PRIMARY KEY (`ID`),
  INDEX `Clients_idx` (`Client_ID` ASC) VISIBLE,  INDEX `Subscriptions_idx` (`Subcsription_ID` ASC) VISIBLE,
   CONSTRAINT `FK_Payments_Client_accounts`    FOREIGN KEY (`Client_ID`)
    REFERENCES `Client_accounts` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE,
  CONSTRAINT `FK_Payments_Subscriptions`    FOREIGN KEY (`Subcsription_ID`)
    REFERENCES `Subscriptions` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE);

CREATE TABLE IF NOT EXISTS `Roles` (
  `ID` INT NOT NULL,  `Title` VARCHAR(45) NOT NULL,  PRIMARY KEY (`ID`),  UNIQUE INDEX `ID_UNIQUE` (`ID`
ASC) VISIBLE);



CREATE TABLE IF NOT EXISTS `Stuff_accounts` (
  `ID` INT NOT NULL,  `Name` VARCHAR(45) NOT NULL,  `Password` VARCHAR(45) NOT NULL,
  `Role_ID` INT NULL,  PRIMARY KEY (`ID`),  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE,
  UNIQUE INDEX `Name_UNIQUE` (`Name` ASC) VISIBLE,  INDEX `Roles_idx` (`Role_ID` ASC) VISIBLE,
  CONSTRAINT `FK_Staff_Roles`    FOREIGN KEY (`Role_ID`)    REFERENCES `Roles` (`ID`)
    ON DELETE CASCADE    ON UPDATE CASCADE);

CREATE TABLE IF NOT EXISTS `Movies_resolutions` (
  `Movie_ID` INT NOT NULL,  `Resolutions_ID` INT NOT NULL,  `File_path` VARCHAR(45) NOT NULL,
  INDEX `Resolutions_idx` (`Resolutions_ID` ASC) VISIBLE,   PRIMARY KEY (`Movie_ID`, `Resolutions_ID`),
  CONSTRAINT `FK_Movie_resolutions_Movies`    FOREIGN KEY (`Resolutions_ID`)
```

```
    REFERENCES `Resolutions` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE,
  CONSTRAINT `FK_Movies_resolutions_resolutions`    FOREIGN KEY (`Movie_ID`)
    REFERENCES `Movies_and_TV_Shows` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE);

CREATE TABLE IF NOT EXISTS `Movies_and_TV_Shows_actors` (
  `Movies_and_TV_Shows_ID` INT NOT NULL,  `Actor_ID` INT NOT NULL,
  INDEX `Actor_idx` (`Actor_ID` ASC) VISIBLE,
  CONSTRAINT `FK_Actors_Movie`    FOREIGN KEY (`Movies_and_TV_Shows_ID`)
    REFERENCES `Movies_and_TV_Shows` (`ID`)   ON DELETE CASCADE    ON UPDATE CASCADE,
  CONSTRAINT `FK_Actor_Movis_actors`    FOREIGN KEY (`Actor_ID`)    REFERENCES `Actors` (`ID`)
    ON DELETE CASCADE    ON UPDATE CASCADE);

CREATE TABLE IF NOT EXISTS `Movies_and_TV_Shows_symbols` (
  `Movies_and_TV_Shows_ID` INT NOT NULL,  `Rating_symbol_ID` INT NOT NULL,
  PRIMARY KEY (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`),
  INDEX `Ratings_idx` (`Rating_symbol_ID` ASC) VISIBLE,
   CONSTRAINT `FK_Movies_symbols`    FOREIGN KEY (`Movies_and_TV_Shows_ID`)
    REFERENCES `Movies_and_TV_Shows` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE,
  CONSTRAINT `FK_Movies_Ratings`    FOREIGN KEY (`Rating_symbol_ID`)
    REFERENCES `Rating_symbols` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE);



CREATE TABLE IF NOT EXISTS `Movies_and_TV_Shows_rating_components` (
  `Movie_ID` INT NOT NULL,  `Rating_components_ID` INT NOT NULL,  PRIMARY KEY (`Movie_ID`,
`Rating_components_ID`),  INDEX `Rating_components_idx` (`Rating_components_ID` ASC) VISIBLE,
  CONSTRAINT `FK_Movies_rating_component`    FOREIGN KEY (`Movie_ID`)
    REFERENCES `Movies_and_TV_Shows` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE,
  CONSTRAINT `FK_Rating_components`    FOREIGN KEY (`Rating_components_ID`)
    REFERENCES `Rating_components` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE);

CREATE TABLE IF NOT EXISTS `TV_Show_seasons` (
  `ID` INT NOT NULL,  `TV_Show_ID` INT NOT NULL,  `Season_number` INT NOT NULL,
  PRIMARY KEY (`ID`, `TV_Show_ID`),  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE,
  INDEX `TV Shows_idx` (`TV_Show_ID` ASC) VISIBLE,
  CONSTRAINT `FK_Seasons_TV Shows`    FOREIGN KEY (`TV_Show_ID`)
    REFERENCES `Movies_and_TV_Shows` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS `TV_shows_episodes` (
  `ID` INT NOT NULL,  `Episode_Number` INT NOT NULL,  `Season_ID` INT NOT NULL,
  PRIMARY KEY (`ID`),  INDEX `Seasons_idx` (`Season_ID` ASC) VISIBLE,
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE,
   CONSTRAINT `FK_Seasons_TV Shows`    FOREIGN KEY (`Season_ID`)
    REFERENCES ` TV_Show_seasons` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE);

CREATE TABLE IF NOT EXISTS `TV_Show_resolutions` (
  `Resolutions_ID` INT NOT NULL,  `Episode_ID` INT NOT NULL,  `File_path` VARCHAR(45) NOT NULL,
  INDEX `Resolutions_idx` (`Resolutions_ID` ASC) VISIBLE,
  INDEX `Episode_resolution_idx` (`Episode_ID` ASC) VISIBLE,
  CONSTRAINT `FK_TV_Shows_Resolutions`    FOREIGN KEY (`Resolutions_ID`)
    REFERENCES `Resolutions` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE,
  CONSTRAINT `FK_TV_Shows_Resolutions_Episode`    FOREIGN KEY (`Episode_ID`)
    REFERENCES `TV_shows_episodes` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE);

CREATE TABLE IF NOT EXISTS `Movies_sessions` (
  `ID` INT NOT NULL,  `Movie_ID` INT NOT NULL,  `Client_ID` INT NOT NULL,
  `Watched_at` TIMESTAMP NOT NULL,  PRIMARY KEY (`ID`),
  INDEX `Movies_idx` (`Movie_ID` ASC) VISIBLE,  INDEX `Users_idx` (`Client_ID` ASC) VISIBLE,
   CONSTRAINT `FK_Movies_sessions_Movies`    FOREIGN KEY (`Movie_ID`)
    REFERENCES `Movies_and_TV_Shows` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE,
  CONSTRAINT `FK_Movies_sessions_Profiles`    FOREIGN KEY (`Client_ID` )
    REFERENCES `Profiles` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE);

CREATE TABLE IF NOT EXISTS `TV_Show_sessions` (
  `ID` INT NOT NULL,  `TV_Show_episode_ID` INT NOT NULL,  `Client_ID` INT NOT NULL,
  `Watched_at` TIMESTAMP NOT NULL,
  PRIMARY KEY (`ID`),  INDEX `Users_idx` (`Client_ID` ASC) VISIBLE,
  INDEX `TV Shows_idx` (`TV_Show_episode_ID` ASC) VISIBLE,
   CONSTRAINT `FK_TV_Shows_sessions`    FOREIGN KEY (`TV_Show_episode_ID`)
    REFERENCES `TV_shows_episodes` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE,
  CONSTRAINT `FK_Users_TV_Shows`    FOREIGN KEY (`Client_ID`)
    REFERENCES `Profiles` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE);
```

```
CREATE TABLE IF NOT EXISTS data_history LIKE movies_and_tv_shows;
ALTER TABLE data_history MODIFY COLUMN ID int NOT NULL,
   DROP PRIMARY KEY, ENGINE = MyISAM, ADD User VARCHAR(45) NOT NULL FIRST,
   ADD action VARCHAR(8) DEFAULT 'insert' after User,
   ADD revision INT NOT NULL AUTO_INCREMENT AFTER action,
   ADD dt_datetime DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP AFTER revision,
   ADD PRIMARY KEY (ID, revision);

CREATE TABLE `clients_logins` (
  `ID` INT NOT NULL,  `Client_ID` INT NOT NULL,  `Logged_in` DATETIME NOT NULL,
  `Logged_out` DATETIME ,  PRIMARY KEY (`ID`),
  UNIQUE INDEX `ID_UNIQUE` (`ID` ASC) VISIBLE,
  INDEX `FK_Client_logins_Client_Accounts_idx` (`Client_ID` ASC) VISIBLE,
  CONSTRAINT `FK_Client_logins_Client_Accounts`    FOREIGN KEY (`Client_ID`)
    REFERENCES `client_accounts` (`ID`)    ON DELETE CASCADE    ON UPDATE CASCADE);
```

Implement External Schema

Procedure to search movies/shows in the database based on their name, genre, year and score:

```
DROP procedure IF EXISTS `search_content`;

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `search_content`(name VARCHAR(45),genre VARCHAR(45),year int, score int)

BEGIN

SELECT M.Title,M.Genre, M.Realese_year,M.Genre, M.Description,M.Director, A.Name, RC.Description, RS.Rating_symbol,

RS.Description

FROM movies_and_tv_shows M, movies_and_tv_shows_actors MA, actors A, rating_components RC,rating_symbols RS,

  movies_and_tv_shows_rating_components MRC, movies_and_tv_shows_symbols MRS

WHERE M.ID = MA.Movies_and_TV_Shows_ID AND MA.Actor_ID = A.ID AND M.Title = IFNULL(name, M.Title) AND M.Genre = IFNULL(genre, M.Genre)

AND M.Realese_year = IFNULL(year,M.Realese_year ) AND M.Score = IFNULL(score, M.Score)

AND M.ID =MRC.Movie_ID AND MRC.Rating_components_ID=RC.ID AND M.ID = MRS.Movies_and_TV_Shows_ID AND MRS.Rating_symbol_ID = RS.ID;

END$$

DELIMITER ;
```

Procedure for search content by ID for Administrator:

```
DROP procedure IF EXISTS `search_content_by_ID`;

DELIMITER $$

CREATE PROCEDURE `search_by_ID` (ID int)

BEGIN

SELECT * FROM movies_and_tv_shows M WHERE

M.ID = IFNULL(ID,M.ID);

END$$

DELIMITER ;
```

Procedures and views for administrator to manipulate with data:

```
DROP procedure IF EXISTS `insert_in_Movies`;

DELIMITER $$

CREATE PROCEDURE `insert_in_Movies` (ID int, Title VARCHAR(45),Realese_year int,

  Distribute_cost DECIMAL(10,2) ,Date_of_distribution DATE ,Score INT ,Description VARCHAR(500) ,

  Cover_Picture VARCHAR(45) ,Director VARCHAR(45) ,Genre VARCHAR(45) )

BEGIN

INSERT INTO movies_and_tv_shows  (ID, Title, Realese_year, Distribute_cost, Date_of_distribution, Score,
Description, Cover_Picture, Director, Genre)

VALUES (ID, Title, Realese_year, Distribute_cost, Date_of_distribution, Score, Description, Cover_Picture, Director,
Genre);

END$$

DELIMITER ;



DROP procedure IF EXISTS `delete_by_ID`;

DELIMITER $$

CREATE PROCEDURE `delete_by_ID` (ID INT)

BEGIN

DELETE FROM movies_and_tv_shows M WHERE

M.ID = ID;

END$$

DELIMITER ;


DROP procedure IF EXISTS `edit_movie_by_ID`;

DELIMITER $$

 PROCEDURE `edit_movie_by_ID`(ID int, Title VARCHAR(45),Realese_year int,

  Distribute_cost DECIMAL(10,2) ,Date_of_distribution DATE ,Score INT ,Description VARCHAR(500) ,

  Cover_Picture VARCHAR(45) ,Director VARCHAR(45) ,Genre VARCHAR(45))

BEGIN

UPDATE movies_and_tv_shows M SET

 M.Title = Title, M.Realese_year = Realese_year,M.Distribute_cost = Distribute_cost,

 M.Date_of_distribution = Date_of_distribution,M.Score = Score,M.Description = Description,

 M.Cover_Picture = Cover_Picture,M.Director = Director,M.Genre = Genre WHERE M.ID = ID;

END$$

DELIMITER ;
```

Triggers for table data_history:

```
DROP TRIGGER IF EXISTS movies_and_tv_shows__ai;

DROP TRIGGER IF EXISTS movies_and_tv_shows__au;

DROP TRIGGER IF EXISTS movies_and_tv_shows__bd;


CREATE TRIGGER movies_and_tv_shows__ai AFTER INSERT ON movies_and_tv_shows FOR EACH ROW
   INSERT INTO data_history SELECT current_user(), 'insert', NULL, NOW(), M.*
   FROM movies_and_tv_shows AS M WHERE M.ID = NEW.ID;


CREATE TRIGGER movies_and_tv_shows__au AFTER UPDATE ON movies_and_tv_shows FOR EACH ROW
   INSERT INTO data_history SELECT current_user(),'update', NULL, NOW(), M.*
   FROM movies_and_tv_shows AS M WHERE M.ID = NEW.ID;


CREATE TRIGGER movies_and_tv_shows__bd BEFORE DELETE ON movies_and_tv_shows FOR EACH ROW
   INSERT INTO data_history SELECT current_user(),'delete', NULL, NOW(), M.*
   FROM movies_and_tv_shows AS M WHERE M.ID = OLD.ID;
```

View for data_history table:

```
CREATE  OR REPLACE VIEW `data_changes` AS
SELECT * FROM data_history;
```

Roles creation:

```
CREATE ROLE 'Administrator', 'Accountant', 'Manager';
GRANT ALL ON VSDB.* TO 'Administrator';
GRANT SELECT ON VSDB.* TO 'Accountant';
GRANT SELECT ON VSDB. *  TO 'Manager';
CREATE USER 'Lex_luter'@'localhost' IDENTIFIED BY 'superman7777';
CREATE USER 'Mia_Sorvino'@'localhost' IDENTIFIED BY 'Witch666';
CREATE USER 'Chris_Pratt'@'localhost' IDENTIFIED BY 'Guardian999';
GRANT 'Administrator' TO 'Lex_luter'@'localhost';
GRANT 'Accountant' TO 'Mia_Sorvino'@'localhost';
GRANT 'Manager' TO 'Chris_Pratt'@'localhost';
```

Procedures and views for accountant:

```
CREATE VIEW `current_month_payments` AS
   SELECT     SUM(`payments`.`Payment_amount`) AS `SUM of payments in current month`
   FROM   `payments`    WHERE   ((MONTH(`payments`.`Payment_date`) = MONTH(CURDATE()))
        AND (YEAR(`payments`.`Payment_date`) = YEAR(CURDATE())));



CREATE  OR REPLACE VIEW `all_payments` AS
SELECT  SUM(`payments`.`Payment_amount`) AS `SUM of all payments`    FROM   `payments`;


CREATE  OR REPLACE VIEW `balance_for_current_month` AS
SELECT   SUM(M.Distribute_cost)-SUM(P.Payment_amount) AS `Balance in current month`
   FROM     payments P, movies_and_tv_shows M
WHERE   ((MONTH(P.Payment_date) = MONTH(CURDATE())) AND (YEAR(P.Payment_date) = YEAR(CURDATE()))
AND  (MONTH(M.Date_of_distribution) = MONTH(CURDATE()))
AND  (YEAR(M.Date_of_distribution) = YEAR(CURDATE())));



DROP procedure IF EXISTS `payments_for_month`;
DELIMITER $$
CREATE PROCEDURE `payments_for_month` (Month INT,Year INT)
BEGIN
SELECT
      SUM(`payments`.`Payment_amount`) AS `SUM of payments `    FROM     `payments`
   WHERE    ((MONTH(`payments`.`Payment_date`) = Month)   AND (YEAR(`payments`.`Payment_date`) = Year));
END$$
DELIMITER ;


DROP procedure IF EXISTS `balance_for_month`;
DELIMITER $$
CREATE PROCEDURE `balance_for_month`(Month INT, Year INT)
BEGIN
SELECT (SELECT  SUM(Payment_amount)  FROM      payments
   WHERE   ((MONTH(Payment_date) = Month)  AND (YEAR(Payment_date) = Year)))
     -
(SELECT     SUM(Distribute_cost)     FROM     movies_and_tv_shows      WHERE
(MONTH(Date_of_distribution) = Month)   AND  (YEAR(Date_of_distribution) = Year))   AS `Balance in current month`;
```

```
END$$
DELIMITER ;



DROP procedure IF EXISTS `cost_of_distribution_for_month`;
DELIMITER $$
CREATE PROCEDURE `cost_of_distribution_for_month` (Month INT, Year INT)
BEGIN
SELECT SUM (Distribute_cost) AS `Sum of distribution costs `   FROM movies_and_tv_shows
   WHERE  MONTH(Date_of_distribution) = Month  AND  YEAR(Date_of_distribution) = Year;
END$$
DELIMITER ;
```

Procedures and views for management team:

```
DELIMITER $$
CREATE  PROCEDURE `most_viewed_movies`(Month INT, Day INT)
BEGIN
SELECT M.Title, (SELECT COUNT(MS.ID) FROM movies_sessions MS
WHERE MS.Movie_ID = M.ID  AND MONTH(MS.Watched_at)=Month AND  DAY(MS.Watched_at)=Day  ) AS views
FROM movies_and_tv_shows M WHERE M.Genre != 'TV Show' ORDER BY views DESC  LIMIT 10;
END$$
DELIMITER ;

DROP procedure IF EXISTS `most_viewed_tv_shows`;
DELIMITER $$
CREATE  PROCEDURE `most_viewed_tv_shows`(Month INT, Day INT)
BEGIN
SELECT M.Title, (SELECT DISTINCT COUNT(TVS.ID) FROM tv_show_sessions TVS, tv_show_seasons TV,
tv_shows_episodes TVE
WHERE TVS.TV_Show_episode_ID = TVE.ID AND TVE.Season_ID = TV.ID AND TV.TV_Show_ID = M.ID
AND MONTH(TVS.Watched_at)=Month AND  DAY(TVS.Watched_at)=Day  ) AS views
FROM movies_and_tv_shows M WHERE M.Genre = 'TV Show' ORDER BY views DESC  LIMIT 10;
END$$
DELIMITER ;
```

```
CREATE VIEW `history_of_tv_shows_view` AS
   SELECT `p`.`Name` AS `Client_name`, `m`.`Title` AS `Movie_title`, `tv`.`Season_number` AS `Season_number`,
      `tve`.`Episode_Number` AS `Episode_Number`,  `tvs`.`Watched_at` AS `Watched_at`
   FROM    ((((`tv_show_sessions` `tvs`  JOIN `movies_and_tv_shows` `m`)  JOIN `profiles` `p`)
      JOIN `tv_show_seasons` `tv`) JOIN `tv_shows_episodes` `tve`)
WHERE ((`m`.`ID` = `tv`.`TV_Show_ID`) AND (`tv`.`ID` = `tve`.`Season_ID`)
AND (`tve`.`ID` = `tvs`.`TV_Show_episode_ID`)      AND (`tvs`.`Client_ID` = `p`.`ID`));


CREATE VIEW `history_of_movies_views` AS
   SELECT    `p`.`Name` AS `Name_of_profile`,   `m`.`Title` AS `Movie_title`, `ms`.`Watched_at` AS `Watched_at`
   FROM ((`movies_sessions` `ms` JOIN `movies_and_tv_shows` `m`) JOIN `profiles` `p`)
   WHERE    ((`m`.`ID` = `ms`.`Movie_ID`)   AND (`ms`.`Client_ID` = `p`.`ID`));



CREATE VIEW `movies_profitability` AS
   SELECT DISTINCT   `m`.`Title` AS `Title`,   ((SELECT COUNT(`ms`.`ID`)
   FROM (`movies_sessions` `ms` JOIN `movies_and_tv_shows` `m`)
         WHERE   (`m`.`ID` = `ms`.`Movie_ID`)) / `m`.`Distribute_cost`) AS `Movies_profability`
   FROM   (`movies_and_tv_shows` `m`  JOIN `movies_sessions` `ms`)
   WHERE (`m`.`ID` = `ms`.`Movie_ID`);



CREATE VIEW `tv_shows_profitability` AS
   SELECT DISTINCT `m`.`Title` AS `Title`, ((SELECT  COUNT(`tvs`.`ID`)
         FROM (((`tv_show_sessions` `tvs`  JOIN `movies_and_tv_shows` `m`)  JOIN `tv_show_seasons` `tv`)
            JOIN `tv_shows_episodes` `tve`)
         WHERE    ((`m`.`ID` = `tv`.`TV_Show_ID`) AND (`tv`.`ID` = `tve`.`Season_ID`)
               AND (`tve`.`ID` = `tvs`.`TV_Show_episode_ID`))) / `m`.`Distribute_cost`) AS `Movies_profability`
   FROM    (((`tv_show_sessions` `tvs`  JOIN `movies_and_tv_shows` `m`) JOIN `tv_show_seasons` `tv`)
      JOIN `tv_shows_episods` `tve`)
  WHERE ((`m`.`ID` = `tv`.`TV_Show_ID`) AND (`tv`.`ID` = `tve`.`Season_ID`)
AND (`tve`.`ID` = `tvs`.`TV_Show_episode_ID`));




CREATE VIEW `clients_return_history` AS
```

```
 SELECT DISTINCT   `c`.`ID` AS `Client ID`, `c`.`Name` AS `Client name`,
    (SELECT    COUNT(*)  FROM     `clients_logins` `cl`
        WHERE  (`c`.`ID` = `cl`.`Client_ID`)
        GROUP BY `c`.`Name`) AS `log in times`
FROM     (`client_accounts` `c`  JOIN `clients_logins` `cl`)
WHERE      (`c`.`ID` = `cl`.`Client_ID`)
```

ADD permissions to do procedures to accountant and manager:

```
GRANT EXECUTE ON PROCEDURE balance_for_month TO 'Accountant';
GRANT EXECUTE ON PROCEDURE cost_of_distribution_for_month TO 'Accountant';
GRANT EXECUTE ON PROCEDURE payments_for_month TO 'Accountant';
GRANT EXECUTE ON PROCEDURE most_viewed_movies TO 'Manager';
GRANT EXECUTE ON PROCEDURE most_viewed_tv_shows TO 'Manager';
```

Load initial Data

```
INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`, `Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (1, 'Alexander', 2004, 50, '2020-03-01', 78, 'The story chronicles Alexander\'s path to becoming a living legend, from a youth fueled by dreams of myth, glory and adventure to his lonely death as a ruler of a vast Empire. Alexander is the incredible story of a life that united the Known World and proved if nothing else, fortune favors the bold.', 'E:\\app\\covers\\10.jpeg', ' Oliver Stone', 'Historic');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`, `Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (2, 'Ice Age 1', 2002, 50, '2019-12-01', 89, 'On Earth 20,000 years ago, everything was covered in ice. A group of friends, Manny, a mammoth, Diego, a saber tooth tiger, and Sid, a sloth encounter an Eskimo human baby. They must try to return the baby back to his tribe before a group of saber tooth tigers find him and eat him.', 'E:\\app\\covers\\1.jpeg', 'Chris Wedge', 'Animation');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`, `Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (3, 'Casino Royale', 2006, 50, '2020-03-01', 86, 'The plot has Bond on an assignment to bankrupt terrorist financier Le Chiffre in a high-stakes poker game at the Casino Royale in Montenegro; Bond falls in love with femme fatale Vesper Lynd, a treasury employee assigned to provide the money he needs for the game.', 'E:\\app\\covers\\2.jpeg', 'Martin Campbell', 'James Bond series');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`, `Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (4, 'Quantum of Solace', 2008, 50, '2020-02-10', 88, 'Quantum of Solace is a 2008 spy film and the twenty-second in the James Bond series produced by Eon Productions. ... In the film, Bond seeks revenge for the death of his lover, Vesper Lynd, and is assisted by Camille Montes, who is plotting revenge for the murder of her own family.', 'E:\\app\\covers\\3.jpeg', 'Marc Forster', 'James Bond series');
```

```sql
INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (5, 'Skyfall', 2012, 50, '2019-02-13', 96, 'Skyfall is
a 2012 spy film and the twenty-third in the James Bond series produced by Eon Productions. ... The story centres on
Bond investigating an attack on MI6; part of a plot by former agent Raoul Silva to discredit and kill M as revenge for
abandoning him.', 'E:\\app\\covers\\4.jpeg', 'Sam Mendes', 'James Bond series');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (6, 'Spectre', 2015, 50, '2020-02-10', 76, 'The
story sees Bond pitted against the global criminal organisation Spectre and their enigmatic leader Ernst Stavro Blofeld
(Christoph Waltz), who plans to launch a national surveillance network to mastermind criminal activities across the
globe.', 'E:\\app\\covers\\5.jpeg', 'Sam Mendes', 'James Bond series');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (7, '10.000 BC', 2008, 50, '2019-10-15', 78,
'10,000 BC is a 2008 American epic action-adventure film directed by Roland Emmerich, starring Steven Strait and
Camilla Belle. The film is set in the prehistoric era and depicts the journeys of a prehistoric tribe of mammoth hunters.',
'E:\\app\\covers\\6.jpeg', 'Roland Emmerich', 'Adventure');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (8, '2012', 2009, 50, '2020-01-15', 83, 'The plot
follows geologist Adrian Helmsley (Ejiofor), who discovers the Earth\'s crust is becoming unstable after a massive
solar flare caused by an alignment of the planets, and novelist Jackson Curtis (Cusack) as he attempts to bring his
family to safety as the world is destroyed by a series of extreme natural ...', 'E:\\app\\covers\\7.jpeg', 'Roland
Emmerich', 'Action');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (9, 'Bourne Identity', 2002, 50, '2020-02-08', 85,
'he Bourne Identity is a 2002 American action-thriller film based on Robert Ludlum\'s novel of the same name. It stars
Matt Damon as Jason Bourne, a man suffering from extreme memory loss and attempting to discover his true identity
amidst a clandestine conspiracy within the Central Intelligence Agency (CIA).', 'E:\\app\\covers\\8.jpeg', 'Paul
Greengrass', 'Action');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (10, 'Lord of the Rings I - The Fellowship', 2001,
50, '2020-01-16', 96, 'Set in Middle-earth, the story tells of the Dark Lord Sauron, who seeks the One Ring. The Ring
has found its way to the young hobbit Frodo Baggins. The fate of Middle-earth hangs in the balance as Frodo and
eight companions (who form the Fellowship of the Ring) begin their journey to Mount Doom in the land of Mordor, the
only place where the Ring can be destroyed.', 'E:\\app\\covers\\9.jpeg', 'Peter Jackson', 'Fantasy');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (11, 'Greys Anatomy', 2004, 50, '2020-01-10', 75,
'The fictional series focuses on the lives of surgical interns, residents, and attendings as they develop into seasoned
doctors while balancing personal and professional relationships. ... It revolves around the title character, Dr. Meredith
Grey, played by Ellen Pompeo.', 'E:\\app\\covers\\11.jpeg', 'Debbie Allen', 'TV Show');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (12, 'Breaking Bad', 2005, 50, '2020-02-10', 85,
'Premise. Set in Albuquerque, New Mexico, between 2008 and 2010, Breaking Bad follows Walter White, a meek high
school science teacher who transforms into a ruthless player in the local methamphetamine drug trade, driven by a
desire to provide for his family after being diagnosed with terminal lung cancer.', 'E:\\app\\covers\\12.jpeg', 'Vince
Gilligan', 'TV Show');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (13, '13 Reasons Why', 2006, 50, '2020-03-10',
74, 'Based on the best-selling book by Jay Asher, 13 Reasons Why follows teenager Clay Jensen as he returns home
from school to find a mysterious box with his name on it lying on his porch. Inside he discovers cassette tapes
recorded by Hannah Baker—his classmate and crush—who tragically committed suicide two weeks earlier.',
'E:\\app\\covers\\13.jpeg', 'Tom McCarthy', 'TV Show');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (14, 'Riverdale', 2008, 50, '2020-03-10', 63,
'Originally a story from Archie Comics which started in 1941, Riverdale centres around a group of high school students
who are shocked by the death of classmate, Jason Blossom. Together they unravel the secrets of Riverdale and who
really killed Jason.', 'E:\\app\\covers\\14.jpeg', 'Mädchen Amick', 'TV Show');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (15, 'Black Mirror', 2006, 50, '2020-03-10', 89,
'Black Mirror is a British dystopian science fiction anthology television series created by Charlie Brooker. He and
Annabel Jones are the programme\'s showrunners. It examines modern society, particularly with regard to the
unanticipated consequences of new technologies.', 'E:\\app\\covers\\15.jpeg', 'Charlie Brooker', 'TV Show');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (16, 'Supernaturals', 2004, 50, '2020-02-10', 89,
'Supernatural is an American dark fantasy television series created by Eric Kripke. ... Starring Jared Padalecki as Sam
Winchester and Jensen Ackles as Dean Winchester, the series follows the two brothers as they hunt demons, ghosts,
monsters, and other supernatural beings.', 'E:\\app\\covers\\16.jpeg', 'Erik Kripke', 'TV Show');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (17, 'Devs', 2019, 50, '2020-01-10', 81, 'In Devs,
an FX limited series, a young software engineer, Lily Chan, investigates the secret development division of her
employer, a cutting-edge tech company based in Silicon Valley, which she believes is behind the murder of her
boyfriend. ... The series is produced by FX Productions.', 'E:\\app\\covers\\17.jpeg', 'Alex Garland', 'TV Show');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (18, 'Westworld', 2015, 50, '2020-01-10', 95,
'Westworld is an exclusive theme park where those who can afford a ticket can live without limits. ... Partners Arnold
Weber and Robert Ford created lifelike robots that pass for humans called hosts. The hosts allow guests to live out
their fantasies (without harming humans) in the park.', 'E:\\app\\covers\\18.jpeg', 'Jonathan Nolan', 'TV Show');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (19, 'ER', 1992, 50, '2020-01-10', 83, 'Television.
ER follows the inner life of the emergency room (ER) of fictional County General Hospital in Chicago, Illinois, and
various critical issues faced by the room\'s physicians and staff..', 'E:\\app\\covers\\19.jpeg', 'Christopher Chulack', 'TV
Show');

INSERT INTO `Movies_and_TV_Shows` (`ID`, `Title`, `Realese_year`, `Distribute_cost`, `Date_of_distribution`,
`Score`, `Description`, `Cover_Picture`, `Director`, `Genre`) VALUES (20, 'Friends', 1994, 50, '2020-01-10', 87,
'Friends is a 90\'s Comedy TV show, based in Manhattan, about 6 friends who go through just about every life
experience imaginable together; love, marriage, divorce, children, heartbreaks, fights, new jobs and job losses and all
sorts of drama.', 'E:\\app\\covers\\20.jpeg', 'James Burrows', 'TV Show');



INSERT INTO `Rating_components` (`ID`, `Description`) VALUES (1, 'Drugs and substance abuse');

INSERT INTO `Rating_components` (`ID`, `Description`) VALUES (2, 'Violence');

INSERT INTO `Rating_components` (`ID`, `Description`) VALUES (3, 'Coerse Language');

INSERT INTO `Rating_components` (`ID`, `Description`) VALUES (4, 'Nudity and sexual content');



INSERT INTO `Rating_symbols` (`ID`, `Rating_symbol`, `Description`) VALUES (1, 'G', 'General Audiences');

INSERT INTO `Rating_symbols` (`ID`, `Rating_symbol`, `Description`) VALUES (2, 'PG', 'Parental Guidance
Suggested');

INSERT INTO `Rating_symbols` (`ID`, `Rating_symbol`, `Description`) VALUES (3, 'PG-13', 'Parents Strongly
Cautioned');

INSERT INTO `Rating_symbols` (`ID`, `Rating_symbol`, `Description`) VALUES (4, 'R', 'Restricted');

INSERT INTO `Rating_symbols` (`ID`, `Rating_symbol`, `Description`) VALUES (5, 'NC-17', 'Adults Only');
INSERT INTO `Rating_symbols` (`ID`, `Rating_symbol`, `Description`) VALUES (6, 'NR', 'Not Rated');


INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (1, 'Daniel Craig', '10.01.1972');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (2, 'John Cusac', '13.02.1964');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (3, 'Guy Hamilton', '25.04.1955');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (4, 'John Glen', '14.08.1975');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (5, 'Olga Kurulenko', '31.01.1981');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (6, 'Matt Damon', '15.06.1975');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (7, 'Viggo Mortensen', '7.05.1971');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (8, 'Colin Farrel', '12.11.1976');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (9, 'Liv Tyler', '5.04.1980');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (10, 'Camilla Belle', '23.08.1991');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (11, 'Ellene Pompeo', '10.09.1968');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (12, 'Braian Cranstone', '7.03.1956');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (13, 'Brandon Flynn', '5.08.1982');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (14, 'Lilly Rainhart', '18.03.1985');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (15, 'John Hamm', '25.10.1974');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (16, 'Jensen Ackles', '18.03.1980');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (17, 'Nick Offerman', '20.12.1971');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (18, 'Ad Harris', '23.07.1965');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (19, 'George Cloony', '2.06.1971');
INSERT INTO `Actors` (`ID`, `Name`, `Date_of_birth`) VALUES (20, 'Jannifer Aniston', '13.05.1974');


INSERT INTO `Resolutions` (`ID`, `Title`) VALUES (1, ' 480p');
INSERT INTO `Resolutions` (`ID`, `Title`) VALUES (2, '720p');
INSERT INTO `Resolutions` (`ID`, `Title`) VALUES (3, '1080p');
INSERT INTO `Resolutions` (`ID`, `Title`) VALUES (4, '1440p');
INSERT INTO `Resolutions` (`ID`, `Title`) VALUES (5, '4K');
INSERT INTO `Resolutions` (`ID`, `Title`) VALUES (6, '8K');


INSERT INTO `Client_accounts` (`ID`, `Name`, `Age`, `Address`, `Payment_option`, `Username`, `Password`,
`Status`) VALUES (1, 'Mike Simmons', 35, '4512 Dorchester street Montreal', 'Visa', 'Mike_Super', 'icandoit777', 1);
INSERT INTO `Client_accounts` (`ID`, `Name`, `Age`, `Address`, `Payment_option`, `Username`, `Password`,
`Status`) VALUES (2, 'Leila Moon', 29, '3625 Peel st. Montreal', 'Mastercard', 'LilM', 'mypassword', 0);
```

```
INSERT INTO `Client_accounts` (`ID`, `Name`, `Age`, `Address`, `Payment_option`, `Username`, `Password`,
`Status`) VALUES (3, 'Leo Dracula', 45, '666 Hell st. Toronto', 'PayPal', 'Dracleo', 'gotohell', 1);

INSERT INTO `Client_accounts` (`ID`, `Name`, `Age`, `Address`, `Payment_option`, `Username`, `Password`,
`Status`) VALUES (4, 'Kirk Nuke', 55, '2103 Corona st. Mexico city', 'Interac', 'KiNuke', 'adiosamigos', 1);

INSERT INTO `Client_accounts` (`ID`, `Name`, `Age`, `Address`, `Payment_option`, `Username`, `Password`,
`Status`) VALUES (5, 'Jane Doe', 33, '4503 Siracuz av. New York', 'Visa', 'JD9090', 'whoami', 0);



INSERT INTO `Profiles` (`ID`, `Name`, `Icon_path`, `Client_ID`) VALUES (1, 'Mike', 'E:\\app\\profiles\\mike.jpeg', 1);

INSERT INTO `Profiles` (`ID`, `Name`, `Icon_path`, `Client_ID`) VALUES (2, 'Leila77', 'E:\\app\\profiles\\lil.jpeg', 2);

INSERT INTO `Profiles` (`ID`, `Name`, `Icon_path`, `Client_ID`) VALUES (3, 'Leoleo', 'E:\\app\\profiles\\leo.jpeg', 3);

INSERT INTO `Profiles` (`ID`, `Name`, `Icon_path`, `Client_ID`) VALUES (4, 'Might Kirk',
'E:\\app\\profiles\\spartak.jpeg', 4);

INSERT INTO `Profiles` (`ID`, `Name`, `Icon_path`, `Client_ID`) VALUES (5, 'JaneSmith', 'E:\\app\\profiles\\jane.jpeg',
5);



INSERT INTO `Subscriptions` (`ID`, `Title`, `Price`) VALUES (1, 'Full pack high resolution', 99);

INSERT INTO `Subscriptions` (`ID`, `Title`, `Price`) VALUES (2, 'Movies high resolution', 59);

INSERT INTO `Subscriptions` (`ID`, `Title`, `Price`) VALUES (3, 'TV Shows high resolution', 59);

INSERT INTO `Subscriptions` (`ID`, `Title`, `Price`) VALUES (4, 'Full pack medium resolution', 59);

INSERT INTO `Subscriptions` (`ID`, `Title`, `Price`) VALUES (5, 'Movies medium resolution', 39);

INSERT INTO `Subscriptions` (`ID`, `Title`, `Price`) VALUES (6, 'TV Shows medium resolution', 39);



INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (1, '2020-08-14', '1234 5678 9874 0123', 99, 1, 1);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (2, '2020-08-10', '3214 5689 7412 0235', 59, 3, 3);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (3, '2020-08-05', '9625 5879 1254 3698', 59, 4, 4);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (4, '2020-07-25', '1234 5678 9874 0123', 99, 1, 1);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (5, '2020-07-22', '2589 3695 4589 2635', 59, 2, 2);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (6, '2020-07-20', '3214 5689 7412 0235', 59, 3, 3);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (7, '2020-07-15', '9625 5879 1254 3698', 59, 4, 4);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (8, '2020-07-07', 'mali@gmail.com', 39, 5, 5);
```

```
INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (9, '2020-06-21', '1234 5678 9874 0123', 99, 1, 1);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (10, '2020-06-20', '2589 3695 4589 2635', 59, 2, 2);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (11, '2020-06-15', '3214 5689 7412 0235', 59, 3, 3);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (12, '2020-06-13', '9625 5879 1254 3698', 59, 4, 4);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (13, '2020-06-05', 'mali@gmail.com', 39, 5, 5);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (14, '2020-05-21', '1234 5678 9874 0123', 99, 1, 1);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (15, '2020-05-13', '2589 3695 4589 2635', 59, 2, 2);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (16, '2020-05-08', '3214 5689 7412 0235', 59, 3, 3);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (17, '2020-05-05', '9625 5879 1254 3698', 59, 4, 4);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (18, '2020-05-01', 'mali@gmail.com', 39, 5, 5);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (19, '2020-04-13', '1234 5678 9874 0123', 99, 1, 1);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (20, '2020-04-07', '2589 3695 4589 2635', 59, 2, 2);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (21, '2020-03-14', '1234 5678 9874 0123', 99, 1, 1);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (22, '2020-03-10', '3214 5689 7412 0235', 59, 3, 3);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (23, '2020-03-05', '9625 5879 1254 3698', 59, 4, 4);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (24, '2020-02-25', '1234 5678 9874 0123', 99, 1, 1);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (25, '2020-02-22', '2589 3695 4589 2635', 59, 2, 2);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (26, '2020-02-20', '3214 5689 7412 0235', 59, 3, 3);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (27, '2020-02-15', '9625 5879 1254 3698', 59, 4, 4);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (28, '2020-02-07', 'mali@gmail.com', 39, 5, 5);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (29, '2020-01-21', '1234 5678 9874 0123', 99, 1, 1);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (30, '2020-01-20', '2589 3695 4589 2635', 59, 2, 2);
```

```
INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (31, '2020-01-15', '3214 5689 7412 0235', 59, 3, 3);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (32, '2020-01-13', '9625 5879 1254 3698', 59, 4, 4);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (33, '2020-01-05', 'mali@gmail.com', 39, 5, 5);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (34, '2019-12-21', '1234 5678 9874 0123', 99, 1, 1);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (35, '2019-12-13', '2589 3695 4589 2635', 59, 2, 2);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (36, '2019-12-08', '3214 5689 7412 0235', 59, 3, 3);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (37, '2019-12-05', '9625 5879 1254 3698', 59, 4, 4);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (38, '2019-12-01', 'mali@gmail.com', 39, 5, 5);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (39, '2019-11-13', '1234 5678 9874 0123', 99, 1, 1);

INSERT INTO `Payments` (`ID`, `Payment_date`, `Payment_information`, `Payment_amount`, `Client_ID`,
`Subcsription_ID`) VALUES (40, '2019-11-07', '2589 3695 4589 2635', 59, 2, 2);



INSERT INTO `Roles` (`ID`, `Title`) VALUES (1, 'Administrator');

INSERT INTO `Roles` (`ID`, `Title`) VALUES (2, 'Accountant');

INSERT INTO `Roles` (`ID`, `Title`) VALUES (3, 'Manager');



INSERT INTO `Stuff_accounts` (`ID`, `Name`, `Password`, `Role_ID`) VALUES (1, 'Lex_luter', 'superman7777', 1);

INSERT INTO `Stuff_accounts` (`ID`, `Name`, `Password`, `Role_ID`) VALUES (2, 'Mia_Sorvino', 'Witch666', 2);

INSERT INTO `Stuff_accounts` (`ID`, `Name`, `Password`, `Role_ID`) VALUES (3, 'Chris_Pratt', 'Guardian999', 3);



INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (1, 3, 'E:\\app\\films\\1.avi');

INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (1, 4, 'E:\\app\\films\\2.avi');

INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (2, 1, 'E:\\app\\films\\3.avi');

INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (2, 5, 'E:\\app\\films\\4.avi');

INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (3, 2, 'E:\\app\\films\\5.avi');

INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (4, 3, 'E:\\app\\films\\6.avi');

INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (4, 4, 'E:\\app\\films\\7.avi');
INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (5, 3, 'E:\\app\\films\\8.avi');
INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (6, 2, 'E:\\app\\films\\9.avi');
INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (6, 4, 'E:\\app\\films\\10.avi');
INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (7, 1, 'E:\\app\\films\\11.avi');
INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (7, 2, 'E:\\app\\films\\12.avi');
INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (8, 5, 'E:\\app\\films\\13.avi');
INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (8, 6, 'E:\\app\\films\\14.avi');
INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (9, 3, 'E:\\app\\films\\15.avi');
INSERT INTO `Movies_resolutions` (`Movie_ID`, `Resolutions_ID`, `File_path`) VALUES (10, 4,
'E:\\app\\films\\16.avi');


INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (1, 8);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (2, 3);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (3, 1);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (4, 5);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (5, 1);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (6, 1);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (7, 10);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (8, 2);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (9, 6);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (10, 7);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (7, 1);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (11, 11);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (12, 12);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (13, 13);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (14, 14);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (15, 15);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (16, 16);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (17, 17);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (18, 18);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (19, 19);
INSERT INTO `Movies_and_TV_Shows_actors` (`Movies_and_TV_Shows_ID`, `Actor_ID`) VALUES (20, 20);


INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (1,
4);
```

```
INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (2,
1);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (3,
3);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (4,
3);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (5,
3);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (6,
3);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (7,
2);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (8,
2);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (9,
3);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (10,
1);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (11,
3);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (12,
5);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (13,
3);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (14,
3);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (15,
5);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (16,
3);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (17,
3);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (18,
5);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (19,
2);

INSERT INTO `Movies_and_TV_Shows_symbols` (`Movies_and_TV_Shows_ID`, `Rating_symbol_ID`) VALUES (20,
2);
```

```
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (1, 2);

INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (3, 2);

INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (4, 2);

INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (5, 2);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (6, 2);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (8, 3);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (9, 2);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (12, 1);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (12, 2);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (12, 3);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (12, 4);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (15, 2);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (15, 3);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (15, 4);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (16, 2);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (16, 3);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (17, 2);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (18, 2);
INSERT INTO `Movies_and_TV_Shows_rating_components` (`Movie_ID`, `Rating_components_ID`) VALUES (18, 4);


INSERT INTO `TV_Show_seasons` (`ID`, `TV_Show_ID`, `Season_number`) VALUES (1, 11, 1);
INSERT INTO `TV_Show_seasons` (`ID`, `TV_Show_ID`, `Season_number`) VALUES (2, 12, 1);
INSERT INTO `TV_Show_seasons` (`ID`, `TV_Show_ID`, `Season_number`) VALUES (3, 13, 1);
INSERT INTO `TV_Show_seasons` (`ID`, `TV_Show_ID`, `Season_number`) VALUES (4, 14, 1);
INSERT INTO `TV_Show_seasons` (`ID`, `TV_Show_ID`, `Season_number`) VALUES (5, 15, 1);
INSERT INTO `TV_Show_seasons` (`ID`, `TV_Show_ID`, `Season_number`) VALUES (6, 16, 1);
INSERT INTO `TV_Show_seasons` (`ID`, `TV_Show_ID`, `Season_number`) VALUES (7, 17, 1);
INSERT INTO `TV_Show_seasons` (`ID`, `TV_Show_ID`, `Season_number`) VALUES (8, 18, 1);
INSERT INTO `TV_Show_seasons` (`ID`, `TV_Show_ID`, `Season_number`) VALUES (9, 19, 1);
INSERT INTO `TV_Show_seasons` (`ID`, `TV_Show_ID`, `Season_number`) VALUES (10, 20, 1);


INSERT INTO `TV_shows_episodes` (`ID`, `Episode_Number`, `Season_ID`) VALUES (1, 1, 1);
INSERT INTO `TV_shows_episodes` (`ID`, `Episode_Number`, `Season_ID`) VALUES (2, 1, 2);
INSERT INTO `TV_shows_episodes` (`ID`, `Episode_Number`, `Season_ID`) VALUES (3, 1, 3);
INSERT INTO `TV_shows_episodes` (`ID`, `Episode_Number`, `Season_ID`) VALUES (4, 1, 4);
INSERT INTO `TV_shows_episodes` (`ID`, `Episode_Number`, `Season_ID`) VALUES (5, 1, 5);
INSERT INTO `TV_shows_episodes` (`ID`, `Episode_Number`, `Season_ID`) VALUES (6, 1, 6);
INSERT INTO `TV_shows_episodes` (`ID`, `Episode_Number`, `Season_ID`) VALUES (7, 1, 7);
INSERT INTO `TV_shows_episodes` (`ID`, `Episode_Number`, `Season_ID`) VALUES (8, 1, 8);

INSERT INTO `TV_shows_episodes` (`ID`, `Episode_Number`, `Season_ID`) VALUES (9, 1, 9);

INSERT INTO `TV_shows_episodes` (`ID`, `Episode_Number`, `Season_ID`) VALUES (10, 1, 10);



INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (1, 1,
'E:\\app\\series\\1.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (2, 2,
'E:\\app\\series\\2.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (3, 3,
'E:\\app\\series\\3.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (4, 4,
'E:\\app\\series\\4.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (1, 5,
'E:\\app\\series\\5.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (2, 5,
'E:\\app\\series\\6.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (4, 6,
'E:\\app\\series\\7.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (1, 7,
'E:\\app\\series\\8.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (2, 7,
'E:\\app\\series\\9.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (1, 8,
'E:\\app\\series\\10.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (3, 8,
'E:\\app\\series\\11.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (3, 9,
'E:\\app\\series\\12.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (5, 9,
'E:\\app\\series\\13.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (1, 10,
'E:\\app\\series\\14.avi');

INSERT INTO `TV Show_resolutions` (`Resolutions_ID`, `Episode_ID`, `File_path`) VALUES (3, 10,
'E:\\app\\series\\15.avi');



INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (1, 1, 1, '2020-07-21
18:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (2, 2, 2, '2020-07-19
20:30:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (3, 3, 4, '2020-07-19
18:50:00');
```

```
INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (4, 4, 5, '2020-07-18
15:26:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (5, 5, 1, '2020-07-17
21:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (6, 6, 2, '2020-07-16
19:30:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (7, 7, 1, '2020-06-28
08:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (8, 8, 4, '2020-06-22
22:25:03');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (9, 9, 5, '2020-06-19
20:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (10, 10, 1, '2020-06-17
23:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (11, 1, 2, '2020-06-15
18:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (12, 2, 4, '2020-06-14
20:30:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (13, 3, 5, '2020-06-10
18:50:00');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (14, 4, 1, '2020-06-06
15:26:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (15, 5, 2, '2020-05-27
21:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (16, 6, 5, '2020-05-25
19:30:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (17, 7, 1, '2020-05-18
08:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (18, 8, 2, '2020-05-16
22:25:03');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (19, 9, 4, '2020-05-14
20:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (21, 1, 1, '2020-04-21
18:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (22, 2, 2, '2020-04-19
20:30:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (23, 3, 4, '2020-04-19
18:50:00');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (24, 4, 5, '2020-04-18
15:26:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (25, 5, 1, '2020-04-17
21:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (26, 6, 2, '2020-04-16
19:30:33');
```

```
INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (27, 7, 1, '2020-03-28
08:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (28, 8, 4, '2020-03-22
22:25:03');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (29, 9, 5, '2020-03-19
20:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (30, 10, 1, '2020-03-17
23:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (31, 1, 2, '2020-03-15
18:25:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (32, 2, 4, '2020-03-14
20:30:33');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (33, 3, 5, '2020-03-10
18:50:00');

INSERT INTO `Movies_sessions` (`ID`, `Movie_ID`, `Client_ID`, `Watched_at`) VALUES (34, 4, 1, '2020-03-06
15:26:33');



INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (1, 1, 1,
'2020-07-22 18:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (2, 2, 3,
'2020-07-21 20:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (3, 3, 4,
'2020-07-20 18:50:00');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (4, 4, 1,
'2020-07-17 15:26:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (5, 5, 3,
'2020-07-16 21:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (6, 6, 4,
'2020-07-14 19:30:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (7, 7, 4,
'2020-07-11 08:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (8, 8, 3,
'2020-07-10 22:25:03');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (9, 9, 1,
'2020-06-22 20:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (10, 10, 1,
'2020-06-17 23:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (11, 10, 4,
'2020-06-15 18:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (12, 9, 3,
'2020-06-13 20:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (13, 8, 4,
'2020-06-11 18:50:00');
```

```
INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (14, 7, 4,
'2020-05-17 15:26:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (15, 6, 3,
'2020-05-16 21:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (16, 5, 1,
'2020-05-14 19:30:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (17, 4, 1,
'2020-05-09 08:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (18, 3, 3,
'2020-05-07 22:25:03');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (19, 2, 3,
'2020-05-03 20:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (20, 1, 4,
'2020-05-01 23:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (21, 1, 1,
'2020-04-22 18:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (22, 2, 3,
'2020-04-21 20:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (23, 3, 4,
'2020-04-20 18:50:00');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (24, 4, 1,
'2020-04-17 15:26:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (25, 5, 3,
'2020-04-16 21:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (26, 6, 4,
'2020-04-14 19:30:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (27, 7, 4,
'2020-04-11 08:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (28, 8, 3,
'2020-04-10 22:25:03');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (29, 9, 1,
'2020-04-09 20:25:33');

INSERT INTO `TV_Show_sessions` (`ID`, `TV_Show_episode_ID`, `Client_ID`, `Watched_at`) VALUES (30, 10, 1,
'2020-04-05 23:25:33');


INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('3', '2', '2020-03-15 18:20:33',
'2020-03-15 19:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('4', '1', '2020-03-17 23:20:33',
'2020-03-18 01:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('5', '5', '2020-03-19 20:20:33',
'2020-03-19 22:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('6', '4', '2020-03-22 22:20:03',
'2020-03-22 23:20:03');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('7', '1', '2020-03-28 08:20:33',
'2020-03-28 12:20:33');
```

```
INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('8', '1', '2020-04-05 23:20:33',
'2020-04-06 01:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('9', '1', '2020-04-09 20:20:33',
'2020-04-09 22:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('10', '3', '2020-04-10 22:25:03',
'2020-04-10 23:25:03');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('11', '4', '2020-04-11 08:20:33',
'2020-04-11 11:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('12', '4', '2020-04-14 19:20:33',
'2020-04-14 22:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('13', '2', '2020-04-16 19:20:33',
'2020-04-16 21:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('14', '3', '2020-04-16 19:20:33',
'2020-04-16 21:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('15', '1', '2020-04-17 15:20:33',
'2020-04-17 19:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('16', '1', '2020-04-17 21:20:33',
'2020-04-17 23:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('17', '5', '2020-04-18 15:20:33',
'2020-04-18 17:20:33');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('18', '4', '2020-04-19 18:50:00',
'2020-04-19 20:50:00');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('19', '2', '2020-04-19 20:30:33',
'2020-04-19 21:40:00');

INSERT INTO `clients_logins` (`ID`, `Client_ID`, `Logged_in`, `Logged_out`) VALUES ('20', '4', '2020-04-20 18:40:00',
'2020-04-20 22:40:00');
```

DESIGN PATTERN

For my project, I used DAO (Data Access Object) design pattern to do the DBMS project. By using the DAO design pattern, I was able to abstract the detail of our application and communicate with the database indirectly. The design pattern provides a DAO layer, and the DAO layer handles all the database operations and communicates with the domain. The reason I chose to follow this design pattern is that if I want to change the underlying persistence mechanism, I can just use the DAO layer, without altering the business/domain layer.

SOFTWARE AND DATABASE DEVELOPMENT APPROACH

For my project, I used both spiral and agile software development methods. The spiral model helped me develop the project iteratively by incorporating feedback at each iteration stage and testing the software before the next incremental refinement. And the agile approach helped me to adapt to the specification and design changes quickly.

Testing and evaluation of project

Testing is the most crucial part of our software development, and we took the time to come up with test data to the software program by considering all the various use cases. All the test data are generated manually and uses the White Box Testing approach. The white box testing helped us to directly examine both the SQL program and the code we wrote in Java to be tested one code at a time. These means all branches of the code and SQL programs must be at least tested once but using many different possible cases.

The work distribution varied from time to time but I worked on certain aspect of the project. I split the work into three main areas. First area is database design, second is implementation of database and third is the design document. But I found this project very difficult for me and I understood that I didn’t do all requirements of the project.

References:

1. https://dev.mysql.com/
2. https://www.mysqltutorial.org/
3. https://stackoverflow.com/