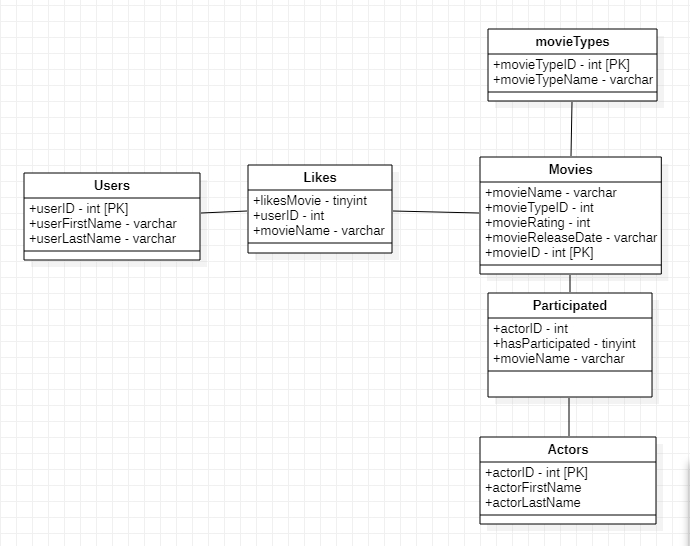
**Exercise 1**

1. A short description of the tables that should be created in the SQL Server and how each table relates to each other.



**Users** table: Holds all the users’ data.

**Movies** table: Holds all the movies’ data. I had to

**Actors** table: Holds all the actors’ data.

**Likes** table: Through a many-to-many relationship, connects the Users with the Movies table. When a user ‘likes’ a movie, a new row is created in the likes table, with the user’s ID, the liked movie’s name and the likesMovie tinyint value is turned to ‘1’ (indicates true). When the user unlikes a movie, the value is turned to ‘0’ (indicates false, and that the user had once liked the movie and decided to unlike it)

**Participated** table: Same logic as the Likes table but instead, for Actors and Movies.

**MovieTypes** table: Since the types of a movie are all specific (action, thriller etc.), it’s best to keep an ID for each one, to avoid mistakes such as adding a movie to a type that does not exist at all.

2) The queries needed for creating all the necessary tables (SQL).

**User table query:**

CREATE TABLE Users (  
    userID int,  
    userFirstName varchar(255),  
    userLastName varchar(255),  
);

**Movie table query:**

CREATE TABLE Movies (  
 movieName varchar(255),  
 movieType varchar(255),  
    rating int,  
    release Date date  
);

**Actor table query:**

CREATE TABLE Actors (  
 actorID int,  
 actorFirstName varchar(255),  
    actorLastName int,  
);

3) The query needed for displaying the information for each movie (name, type, rate, release date).

SELECT \*

FROM Movies;

4) The query that displays the information for each movie (name, type, rate, release date) that the user with ID = 100 likes. E.g. user 100 likes the movies Inception, Terminator 3 etc.

SELECT ‘User’, Users.userID, ‘likes the movie’, Movies.movieName

FROM Users

INNER JOIN Likes

ON Users.userID = Likes.UserID

INNER JOIN Movies

ON Movies.movieName = Likes.MovieName

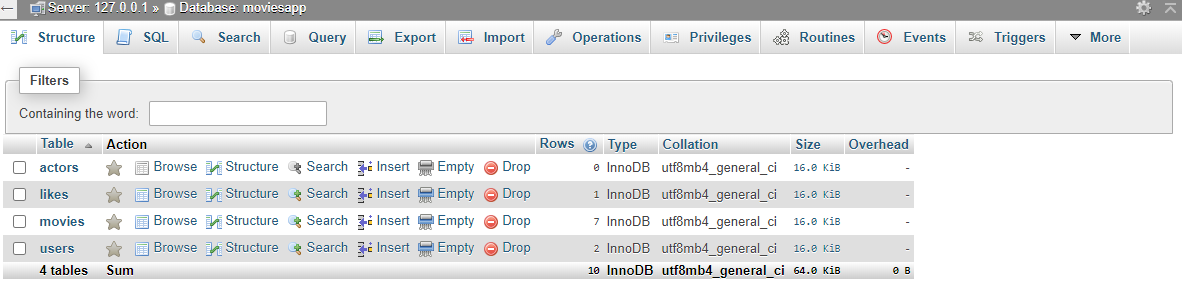
WHERE movieID=100 AND likesMovie=1;

5) A query that deletes all Action movies from the system.

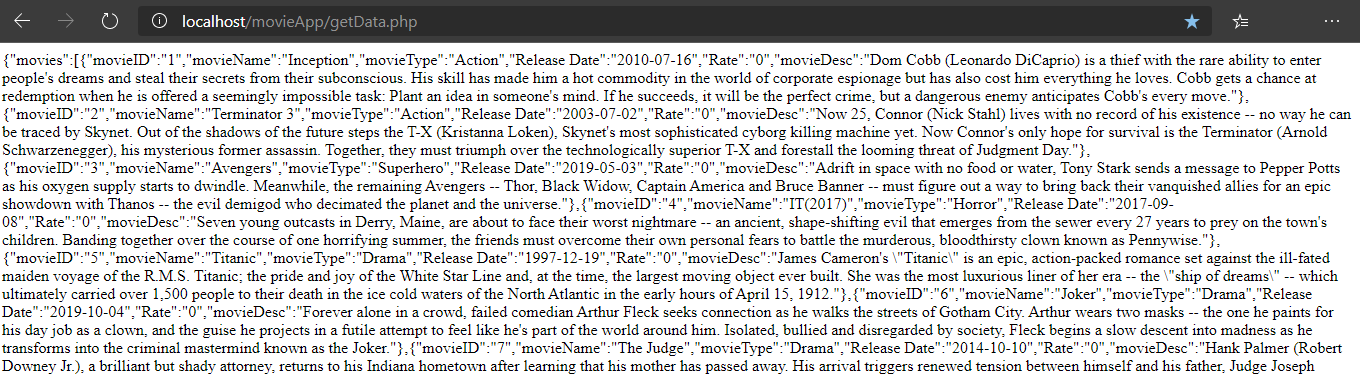
DELETE FROM Movies

WHERE movieType=’Action’;

**Exercise 2**

The way I approached the second exercise was by firstly actually creating the database in my local server in my laptop through PHPmyadmin. (used all the columns that were required) The moviesapp.sql file has also been included in the zip file and contains all the actual database that was used, along with its values etc.

Then, I created two PHP files. One PHP file which would be the connection to the database server and the other PHP file which would display all the information from the movies file in a Json format. The PHP files do not use the mysqlite method, but instead use PDO. Both files have also been included.



After generating the json file in a webpage, my initial thought was to import and read the getData.php file directly through Android Studio. I was unable to make the connection work (although when I entered the 10.2.2.0/movieApp/getData.php URL in Chrome **while in the emulator**, the link would work fine). What I decided to do instead, was to use the website <http://myjson.com/> to generate a link out of the copy pasted generated data from my getData.php file. This proved to worked just fine. The actual URL can be either found in the MainActivity java file inside Android studio or through the link <https://api.myjson.com/bins/kn2yu>

In Android studio, I used two layouts (activity\_main.xml and list\_movie.xml) and had two working scripts (MainAcitivity.java and httpHandler.java). The httpHandler script would test the connection and return the website as a String variable(in json format), in order for the MainActivity script to ‘deJSON’ it and list all the movies in a ListView. When you click in one of the movies in the ListView, a small pop up appears which displays all the movie’s information for some seconds.

**What did not work:** My initial plan was to only have only the names of the movies in ListView, and upon click on a movie name you would be redirected in another View to see all the details of the clicked movie. Some attempts to do this include creating a ‘third’ layout to display the details there, to decrypt the specific movie’s json format and display them there (notice the comments in MainActivity). I also had created a third PHP script which would display the specific details of one movie.

For any clarifications please contact me. The Projects folder contains the whole actual Android Studio file but I also included the files separately in case that’s easier for you.