Trivia!

William Wiemann

April 24th, 2021

# Introduction

The application I would like to build is a trivia game. This trivia game would be targeted to audiences, such as myself, who enjoy movies, board games, and video games. The database will store a series of attributes, each associated with the previous topics. For movies, we will have the actor first and last name, title, and year released. For the video game database, we will have the publisher, release date, genre, and title. For the board game database, we will have the title, genre, publisher, and country of origin. These are the three games we wish our user to play. Besides the user we will have an admin class, which allows the admin to remove, add, or edit existing information in the database. The user can only add record their first and last name, as well as their high score they received after finishing the game.

The user will expect to be able to navigate a series of ten questions. A series of question will be written, and randomly filled in by the attributes in the database. These questions will be different based on the category and the content that is in each database. With the series of 10 questions in each category, a series of four options will exist in the question as a multiple choice. These questions are randomly generated by the other existing attributes of the incorrect answers. Once the information answer has been received, the user will find out if they are correct or incorrect and receive ten points for a correct answer.

This is the ideal project from the existing project options due to the flexibility and universal nature of a database. With no experience with airplanes, even being in the Air Force, this trivia game will give me the knowledge and experience to work on other databases as needed. This will be a project I am proud to show-off for future jobs.

# Deliverable 1: Project Description:

The problem is they are not enough database trivia games that suit my personal needs. Video games, movies, and board games are all hot topics in 2021. A trivia game needs to exist to satisfy all our needs.

A database is the best way to interact with a trivia game such as the one above. Being able to add/edit and input a high score is key for any on-going developing trivia game. A database also allows us to have associate attributes for a single value such as the “title” of a movie. This allows us to create a universal question and essentially fill in the blank using the associated attributes.

This can not be done without anything other than a database. An array base game such as this, would be confusing, as one will most likely have to work with 2-dimensional arrays. I have personally done something like this, and I eventually stopped working on the project due to the repetitive annoyance of it.

## User Classes:

For our user classes, we would require two. The first one is for the User. The User can view and interact with the game, gaining points. The user is only allowed to add to the database at the very end, when it inputs the high score.

Our second user class would be an Admin class. An admin class allows us to have maintainability and control of our trivia game. This is the key to any game as longevity seems to be required in the game world. The main purpose of this class is to add, remove, and edit parts of the database.

## Stored Information:

Four our game, we will have four main “things” we need to keep track of. The first one is regarding our first game, movies. This would include the Actor, Title, and Year of a film. Our second is the participants along with the high score they received. The third is our next category, video games. This will include the publisher, release date, genre, and title. Lastly, the category is board games. This would include the genre, title, publisher, and country of origin.

## Database Operations Needed:

The user will manipulate the data of the database by having first having the option to select the category they wish to play video games, movies, or board games. The interaction within the specific games themselves itself will be a “fill in the blank” style approach. When the user has the option to select four options, only one of them being correct.

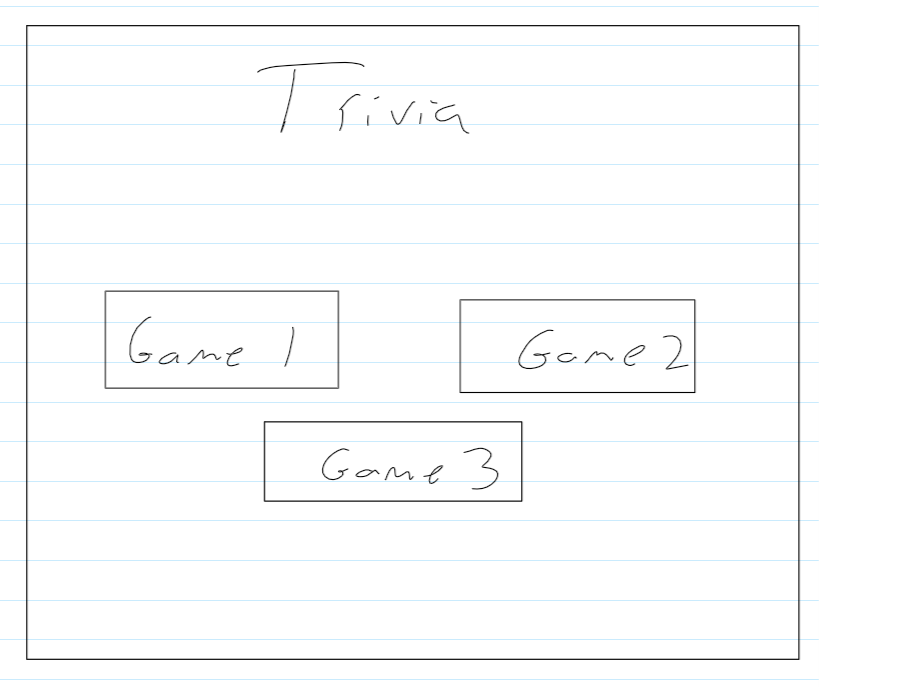
As a user, I want to be able to save my high score.

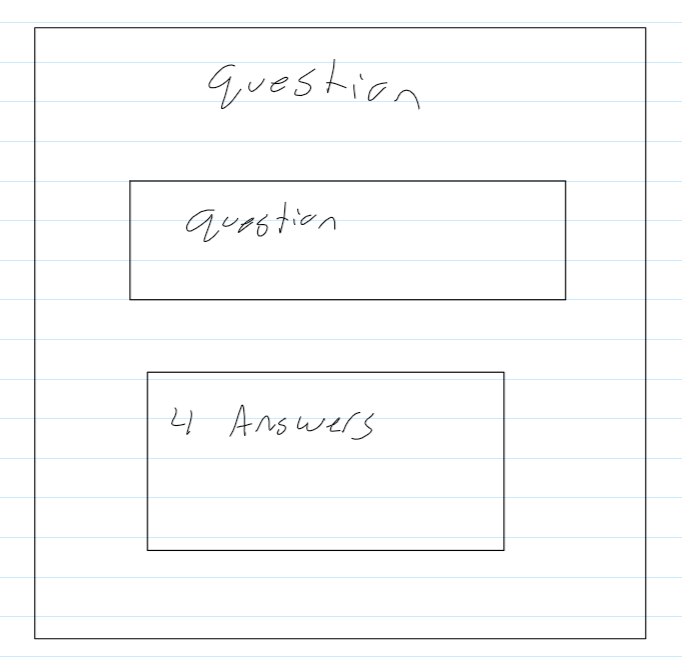
As a user, I want to play a variety of trivia games.

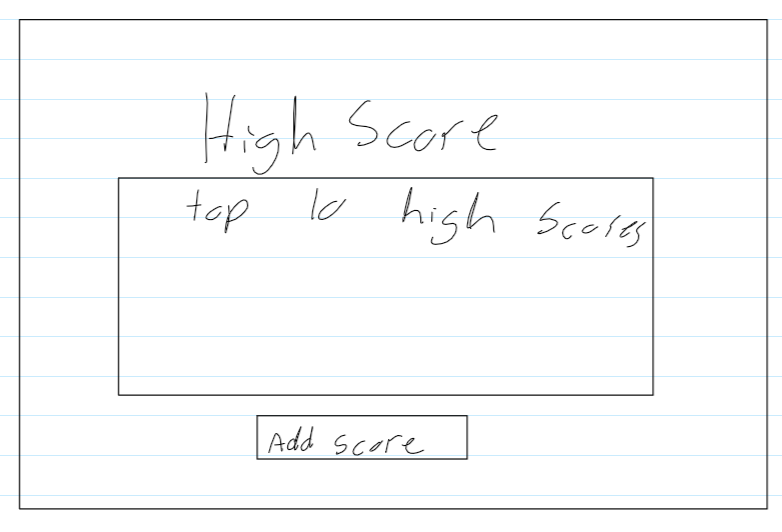
As a user, I want to see what my friend’s high score is.

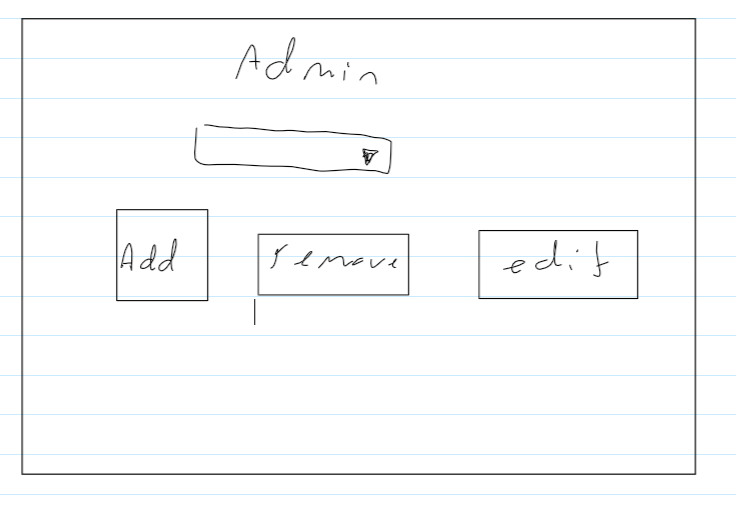
As a user, I want to be challenged by question with various answer options.

## Mockup of Application





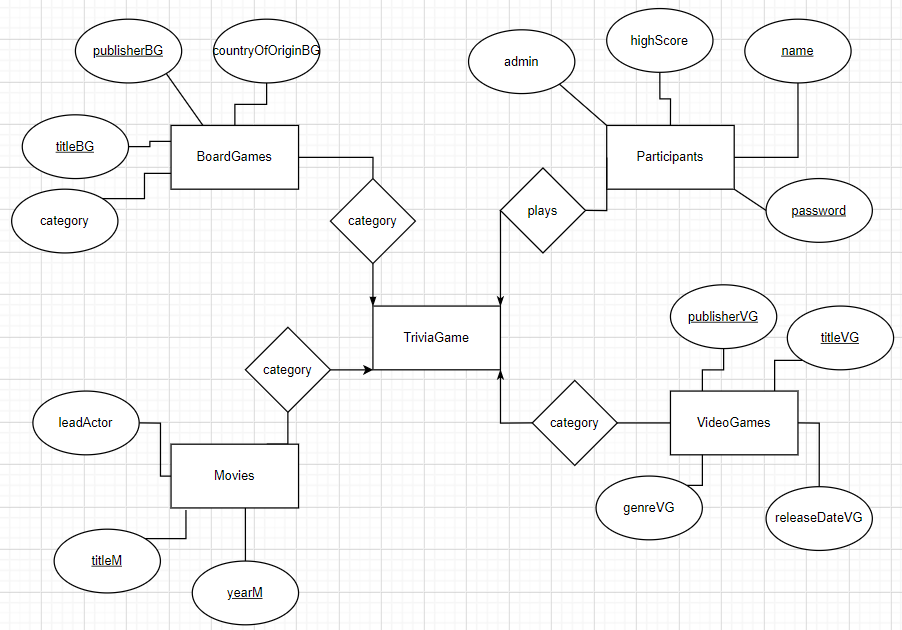




# *Deliverable 2:* Design of Relations

In this section, the design of the relational database is described. First an E/R diagram models our problem. We translate the model into a relational schema. Analysis demonstrates that the schema is in 3rd normal form.

## E/R Modeling



The conceptual design for the E/R model above, consists of the game, the categories, and the participant. The participant will have a primary key consisting of the name and the high score of the individual. The VideoGame will consist of four categories, publisher, title, genre, and releaseDate – with publisher and title being the primary keys. Movies will consist of the genre, title, and year – with the title and year being the primary key. BoardGames consists of genre, title, publisher, and countryOfOrigin – with publisher and title being the primary keys.

One of the main attributes I had to consider in my E/R model is avoiding redundancy. I originally planned the TriviaGame entity set to consist of set of attributes called “categories” – which would consist of the other four entity sets. I avoided adding this to reduce redundancy and avoid anomalies. Another way I reduced redundancy was labeling each attribute with the initials of the entity set. Since many of my entity sets can consist of the same attributes, I decided to define each uniquely. Another main attribute I focused on was the simplicity. I organized the E/R model and made it as clear and simple as it functionally can be.

## Relational Schema

CREATE TABLE Movies (

titleM CHAR(50),

yearM INT,

genreM CHAR(25),

PRIMARY KEY (titleM, yearM)

);

CREATE TABLE VideoGames (

titleVG CHAR(50),

releaseDateVG INT,

genreVG CHAR(25),

publisherVG CHAR(50),

PRIMARY KEY (titleVG, publisherVG)

);

CREATE TABLE BoardGames (

titleBG CHAR(40),

countryOfOriginBG CHAR(40),

genreBG CHAR(20),

publisherBG CHAR(50),

PRIMARY KEY (titleBG, publisherBG)

);

CREATE TABLE Participants (

highScore INT,

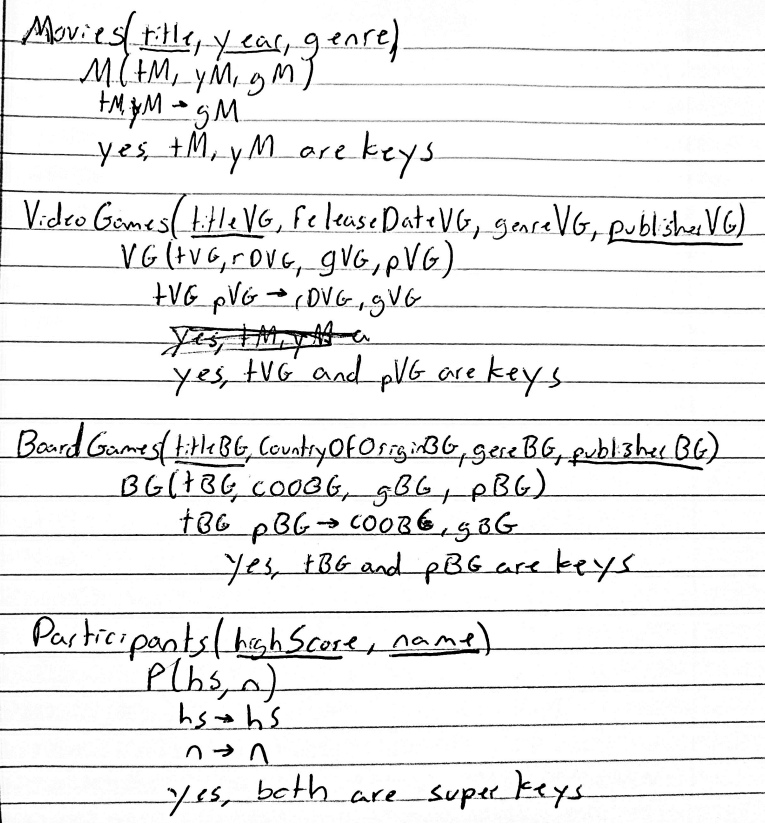
name CHAR(25),

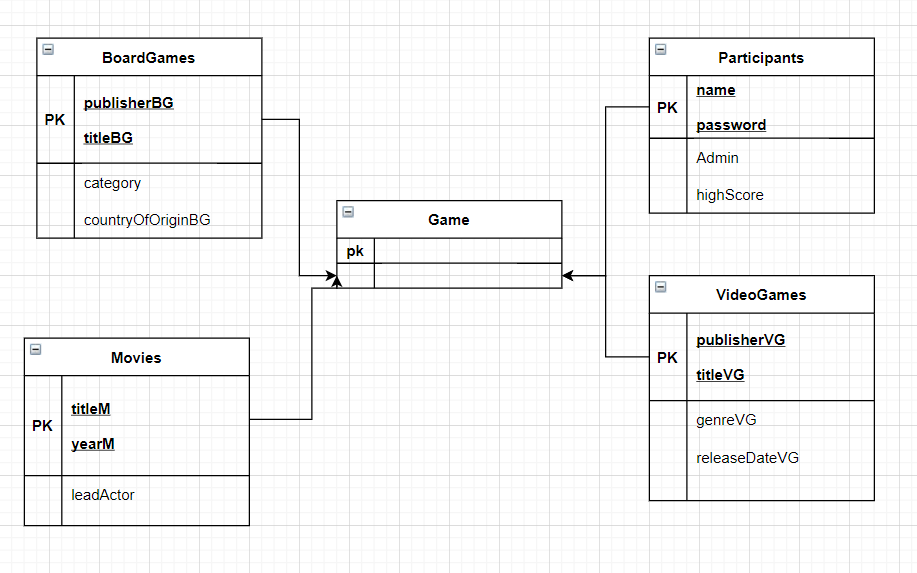
password CHAR(10),

admin BOOL,

PRIMARY KEY (name, password)

);





# Deliverable 3: Implementation of Database

This section discusses the translation of the design into an actual database in SQL. It focuses upon the database exclusively, and does not include detailed design/implementation of the application that utilizes the database.

## Definition of SQL Database Schema

This section presents the SQL schema for the database designed in the previous section.

### Database schema

***use TriviaGame;***

***CREATE TABLE MarvelMovies (***

***titleM CHAR(50),***

***releaseDateM INT,***

***leadActor CHAR(25),***

***PRIMARY KEY (titleM, releaseDateM)***

***);***

***CREATE TABLE VideoGames (***

***titleVG CHAR(50),***

***releaseDateVG INT,***

***genreVG CHAR(25),***

***publisherVG CHAR(50),***

***PRIMARY KEY (titleVG, publisherVG)***

***);***

***CREATE TABLE BoardGames (***

***titleBG CHAR(40),***

***countryOfOrigin CHAR(40),***

***category CHAR(20),***

***publisherBG CHAR(50),***

***PRIMARY KEY (titleBG, publisherBG)***

***);***

***CREATE TABLE Participants (***

***highScore INT,***

***name CHAR(25),***

***password CHAR(10),***

***admin BOOL,***

***PRIMARY KEY (name, password)***

***);***

### Sample Data

Board Game data insert

INSERT INTO `triviagame`.`boardgames` (`titleBG`, `countryOfOrigin`, `category`, `publisherBG`) VALUES ('Nemesis', 'Poland', 'Campaign', 'Awaken Realms');

INSERT INTO `triviagame`.`boardgames` (`titleBG`, `countryOfOrigin`, `category`, `publisherBG`) VALUES ('Catan', 'Germany', 'Resource-driven', 'KOSMOS');

INSERT INTO `triviagame`.`boardgames` (`titleBG`, `countryOfOrigin`, `category`, `publisherBG`) VALUES ('Dead of Winter', 'United States', 'Co-operative', 'Plaid Hat Games');

INSERT INTO `triviagame`.`boardgames` (`titleBG`, `countryOfOrigin`, `category`, `publisherBG`) VALUES ('Betrayal at House on the Hill', 'United States', 'Traitor-mechanic', 'Avalon Games');

INSERT INTO `triviagame`.`boardgames` (`titleBG`, `countryOfOrigin`, `category`, `publisherBG`) VALUES ('Axis and Allies', 'Australia', 'Wargame', 'Jedko Games');

INSERT INTO `triviagame`.`boardgames` (`titleBG`, `countryOfOrigin`, `category`, `publisherBG`) VALUES ('Terraforming Mars', 'Sweden', 'Economic', 'FryxGames');

INSERT INTO `triviagame`.`boardgames` (`titleBG`, `countryOfOrigin`, `category`, `publisherBG`) VALUES ('One Night Werewolf', 'United States', 'Party', 'Bezier Games');

Marvel Movies data insert

INSERT INTO `triviagame`.`marvelmovies` (`titleM`, `releaseDateM`, `leadActor`) VALUES ('Captain America: First Avenger', '2011', 'Chris Evans');

INSERT INTO `triviagame`.`marvelmovies` (`titleM`, `releaseDateM`, `leadActor`) VALUES ('Captain Marvel', '2019', 'Brie Larson');

INSERT INTO `triviagame`.`marvelmovies` (`titleM`, `releaseDateM`, `leadActor`) VALUES ('Iron Mane', '2008', 'Robert Downy Jr.');

INSERT INTO `triviagame`.`marvelmovies` (`titleM`, `releaseDateM`, `leadActor`) VALUES ('Doctor Stange', '2016', 'Benedict Cumberbatch');

INSERT INTO `triviagame`.`marvelmovies` (`titleM`, `releaseDateM`, `leadActor`) VALUES ('The Incredible Hulk', '2008', 'Edward Norton');

INSERT INTO `triviagame`.`marvelmovies` (`titleM`, `releaseDateM`, `leadActor`) VALUES ('Thor', '2011', 'Chris Hemsworth');

INSERT INTO `triviagame`.`marvelmovies` (`titleM`, `releaseDateM`, `leadActor`) VALUES ('Spider-man: Homecoming', '2017', 'Tom Holland');

INSERT INTO `triviagame`.`marvelmovies` (`titleM`, `releaseDateM`, `leadActor`) VALUES ('Black Panther', '2018', 'Chadwick Boseman');

Video Games data insert

INSERT INTO `triviagame`.`videogames` (`titleVG`, `releaseDateVG`, `genreVG`, `publisherVG`) VALUES ('Call of Duty: World at War', '2008', 'FPS', 'Activision');

INSERT INTO `triviagame`.`videogames` (`titleVG`, `releaseDateVG`, `genreVG`, `publisherVG`) VALUES ('Destiny 2', '2019', 'Looter-shooter', 'Bungie');

INSERT INTO `triviagame`.`videogames` (`titleVG`, `releaseDateVG`, `genreVG`, `publisherVG`) VALUES ('Binding of Isaac', '2011', 'Rogue-like', 'Edmund McMillen');

INSERT INTO `triviagame`.`videogames` (`titleVG`, `releaseDateVG`, `genreVG`, `publisherVG`) VALUES ('Total War: Warhammer', '2016', 'Grand-strategy', 'Sega');

INSERT INTO `triviagame`.`videogames` (`titleVG`, `releaseDateVG`, `genreVG`, `publisherVG`) VALUES ('Dragon Age: Inquisition', '2014', 'RPG', 'Electronic Arts');

INSERT INTO `triviagame`.`videogames` (`titleVG`, `releaseDateVG`, `genreVG`, `publisherVG`) VALUES ('Naturo to Boruto: Shinobi Striker', '2018', 'Fighting', 'Bandai');

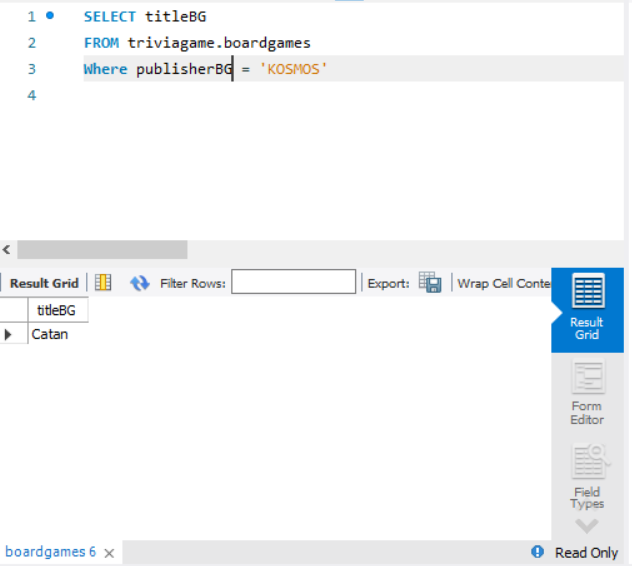
INSERT INTO `triviagame`.`videogames` (`titleVG`, `releaseDateVG`, `genreVG`, `publisherVG`) VALUES ('Devil May Cry 5', '2019', 'Hack and Slash', 'CAPCOM');

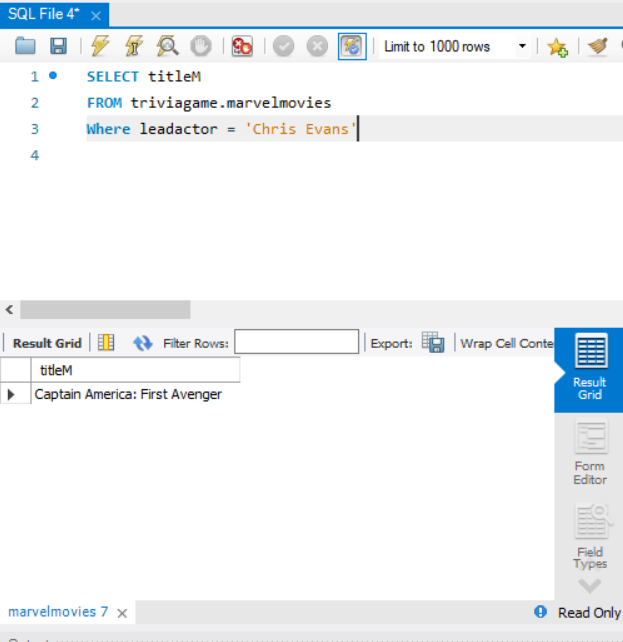
Participants data insert

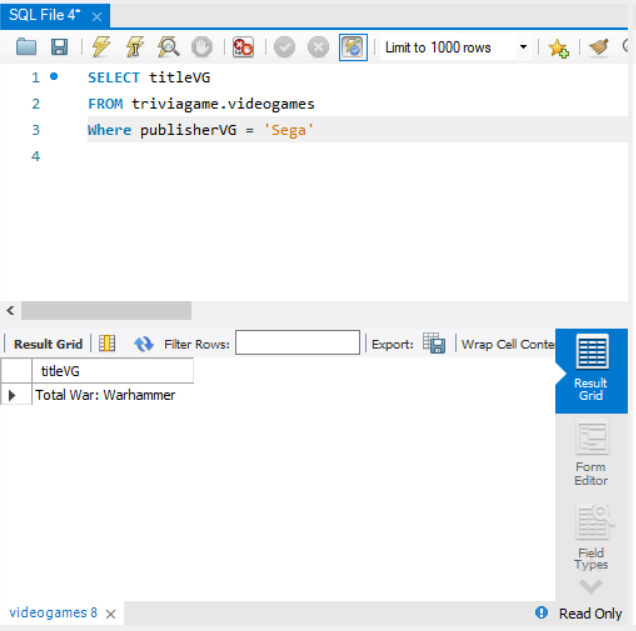
INSERT INTO `triviagame`.`participants` (`name`, `password`, `admin`) VALUES ('Stansbury', '12345', '0');

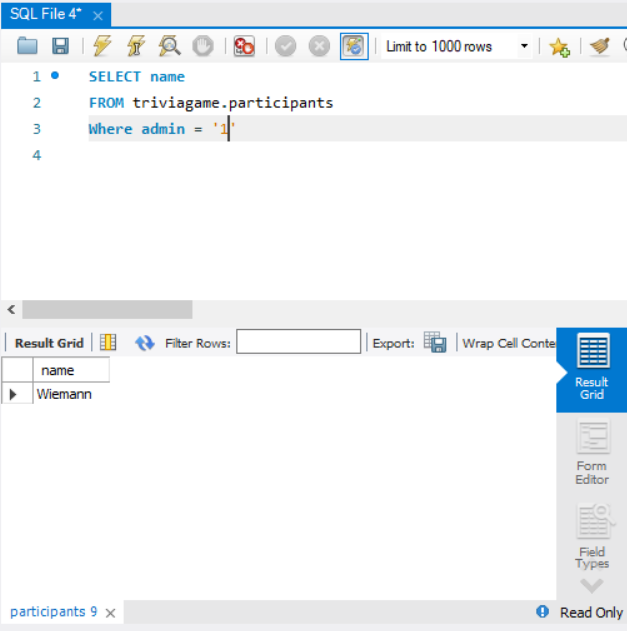
INSERT INTO `triviagame`.`participants` (`name`, `password`, `admin`) VALUES ('Wiemann', '12345', '1');

### Schema Test Results









## Database Operations

**Insert** (Admin/User):

Admin – Able to insert additional information such as marvel movies, board games, and video games. Also will allow to give other participants the admin status.

User – Create new user in a log-in format. This user will exist in the ‘participants’ table and will consist of a name and password. Once the game is completed, the high score will update according to whether it is higher than the previous high score.

**Delete** (Admin):

Admin – Delete rows from the database from marvel movies, board games, video games, and participants.

**Select** (Admin/User):

Admin – Allows the admin to play the game, which will select data according to the questions.

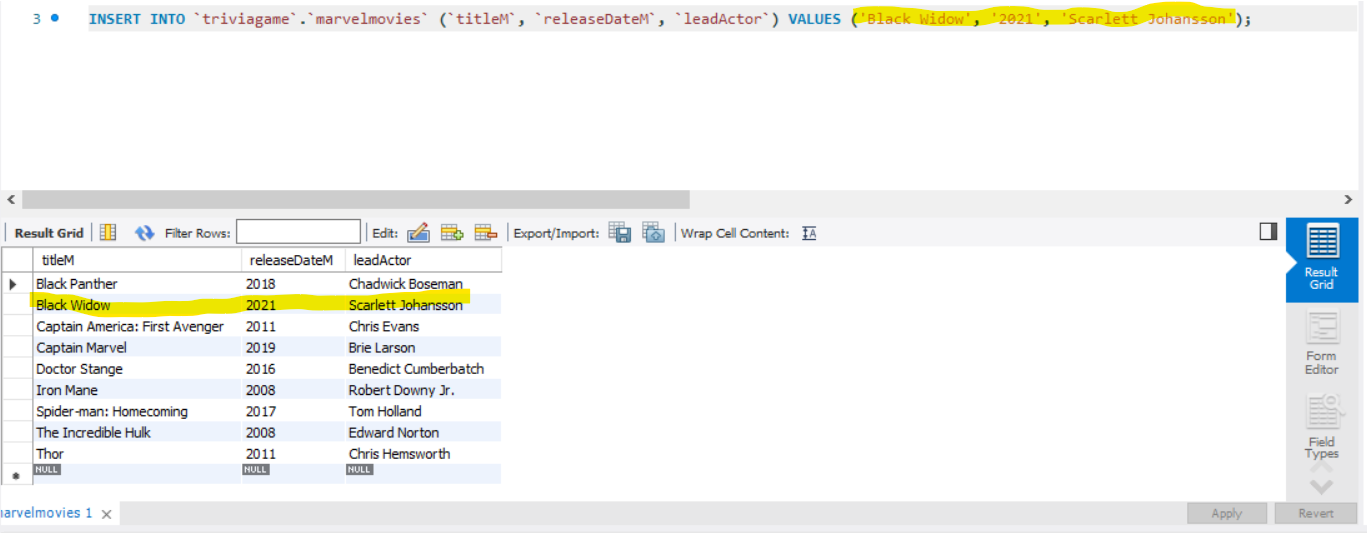
User - Allows the user to play the game, which will select data according to the questions.

## Operations Testing / Demonstration

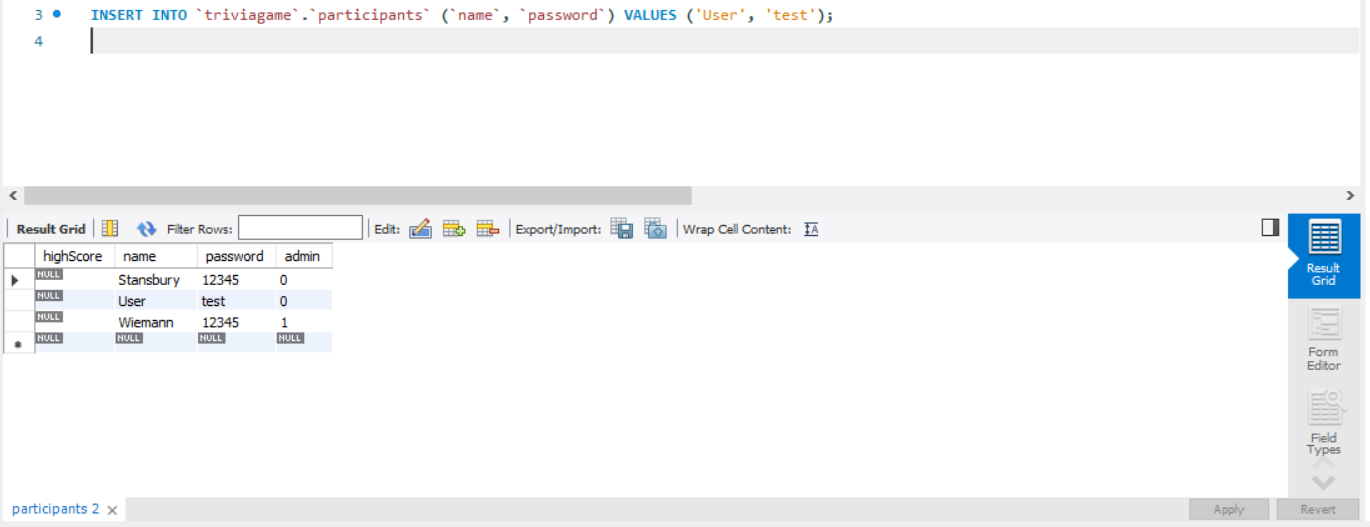
**Insert**:

Admin(creating a new row):

INSERT INTO `triviagame`.`marvelmovies` (`titleM`, `releaseDateM`, `leadActor`) VALUES (‘Black Widow’, '2021', 'Scarlett Johansson');



User (creating login):

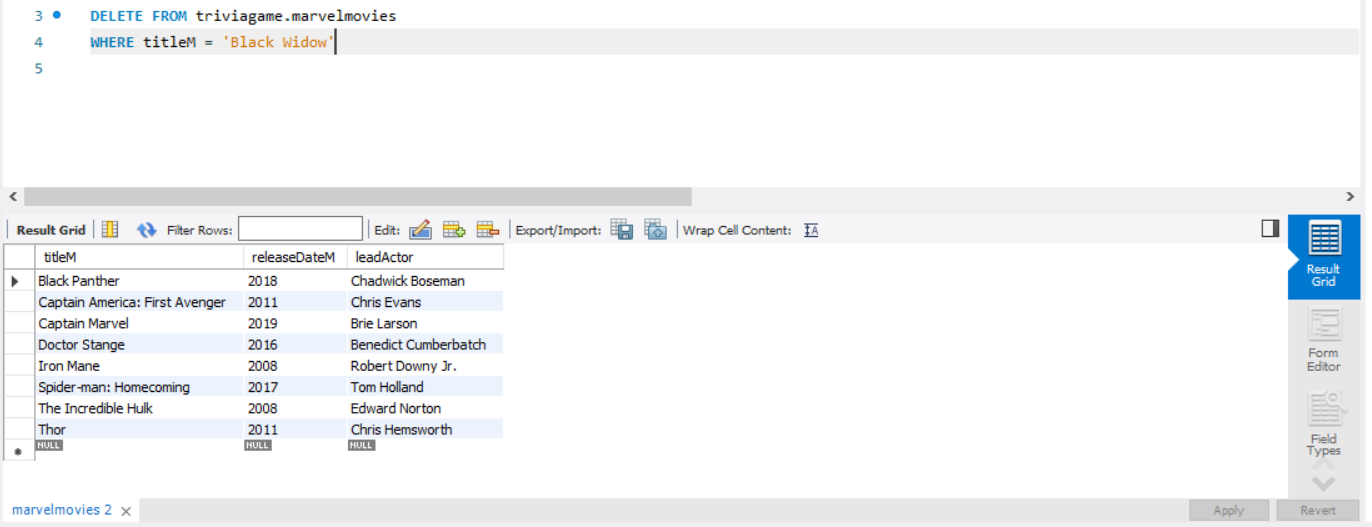
****

**Delete**:

Admin

DELETE FROM triviagame.marvelmovies

WHERE titleM = ‘Black Widow’



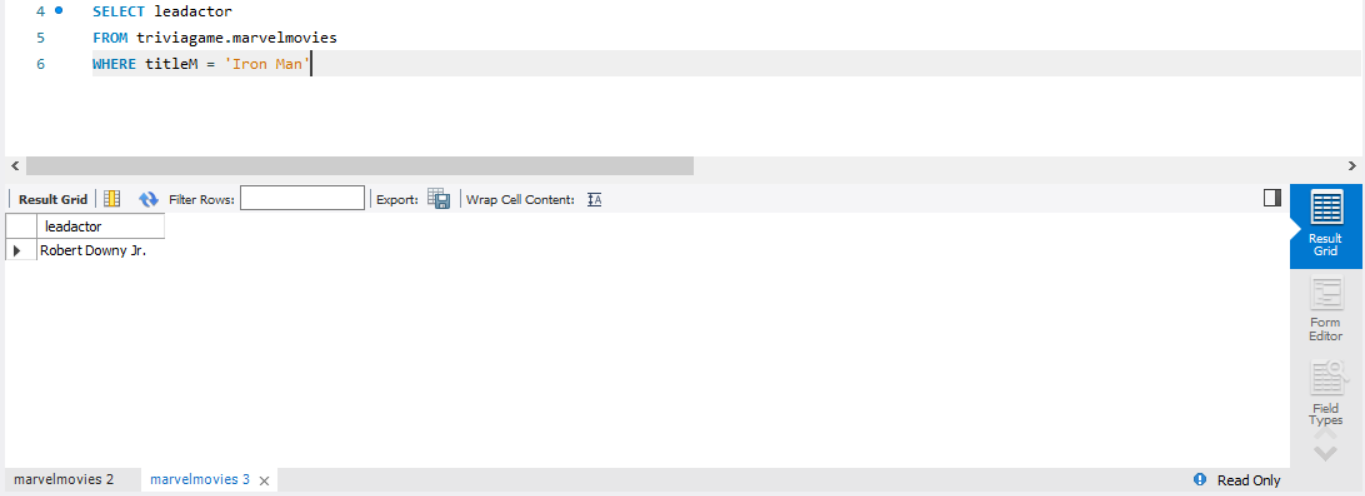
**Select**:

Both Admin and User

SELECT leadactor

FROM triviagame.marvelmovies

WHERE titleM = ‘Iron Man



Deliverable 4: Application Implementation

This section briefly describes the application that was developed. Appendix A contains a code listing of all applications. In this section, each source file is briefly described in the same order in which it is presented in the appendix.

## TriviaGame

|  |  |
| --- | --- |
| Description | For TriviaGame, the main purpose of this application is the JavaFX functionality. Here, we extend Application, allowing us to use JavaFX. We also set the MainMenu in this class. We do this by creating the menuButtons in the MainMenu class, then adding them to the main pane. |
| User Input | No user input |
| User Output | No user output |
| Database Operations | This source file does not have any database interaction. |

## MainMenu

|  |  |
| --- | --- |
| Description | The purpose of MainMenu is to manage and create the features that are used on our main menu. This class contains a smaller class called MenuButton. MenuButton adds our visual menu buttons, to make them unique. MenuButton consists code from previous project that I have wrote.  Other than the button functionality, here we use our admin status (from Participants.login()), to INSERT items into our database. |
| User Input | User input consists of having admin privileges and inputting new information in our database. |
| User Output | The output is used to add items to our database. |
| Database Operations | We use the INSERT function if the user has admin privileges. We are able to add new data to our existing categories, expanding our database. |

## GenericGame

|  |  |
| --- | --- |
| Description | GenericGame builds our questions, by using preset arrays consisting of various questions for each category, then grabbing information in our database. The information we select from our database, must have the attribute that we have gathered to complete our question. From there, we must gather three other attributes that are not associated with our question. |
| User Input | User select the questions and generates points that are added to their high score, from Participants.saveHighScore(). |
| User Output | Output that is given to the user, consists of whether or not a question is correct, the question itself, the questionNum, and all the answers. |
| Database Operations | The Database operations for this class is probably the biggest out of all of them. However, it is very repetitive, as the class is using the same thing to: grab a question and answer, then grab three incorrect answers from the database. (SELECT) |

## Participants

|  |  |
| --- | --- |
| Description | This class is meant to login, create user, and save high scores. This all occurs by interacting with the database, to select data such as name, password, admin status, and high score. This class is purely meant to manipulate Participants table. |
| User Input | To login and create an account, the user inputs a username and password. |
| User Output | The program will automatically set the admit status to , or 1 if it has been already established. |
| Database Operations | This class does SELECT, UPDATE, and INSERT. Insert is used when the user creates an account. Update is used to update the high score of the user. Finally, SELECT is used to grab the high scores and display them. |

1. **Conclusion**

With the creation of this program, the most challenging portion was starting to work with MySQL and Java. It was very intimidating as I was unsure where to start. After delving deep into the syntax, I was comfortable to start this project. I continued to learn and adapt my programming skills, to include the syntax for MySQL, as well as interacting with the database in general. This was a skill I was progressively seeing grow, which felt great. The final program is one I am very proud of. I am able to completely manipulate and use data without using the overwhelming and tedious File I/O, which I have been using previously.

I do believe this class would have gained a great deal from some lab interaction. The first portion of this class was confusing, but the final project has seem to complete the thought process of this data science class.

**References**

“All 23 Marvel Movies In Order: How To Watch MCU Movies Chronologically,” *Rotten Tomatoes Movie and TV News All 23 Marvel Movies In Order How To Watch MCU Movies Chronologically Comments*. [Online]. Available: https://editorial.rottentomatoes.com/guide/marvel-movies-in-order/. [Accessed: 02-Apr-2021].

*BoardGameGeek*. [Online]. Available: https://www.boardgamegeek.com/. [Accessed: 02-Apr-2021].

“Steam Store,” *Welcome to Steam*. [Online]. Available: http://store.steampowered.com/. [Accessed: 24-Apr-2021].