Purpose:

The purpose of this project was to make a GUI that interfaced with our phase 1 tables and database we made earlier this semester.

My Techniques/ Setup:

For this project, I used the SQL connector library in python which can be installed via PIP installer (or the default SQL installer). The connector library allows for intermediate connections to a SQL database as well as error reporting and constraint checking. This project also uses the python Time library for time reporting as well as the built-in tkinter library for the use of a GUI in python.

My Methods:

I used several methods to connect to the database in this project, they are as follows:

Submit():

Submits a query to the database and prints out the returned result. If there is an error with the user-submitted query, then the appropriate error is outputted directly from SQL.

deletePlayers():

Submits a request to the database to delete all Players from the players table. If there is an error in the process of deleting the players, then the appropriate error is outputted directly from SQL.

deletePlay():

Submits a request to the database to delete all Play from the Play table. If there is an error in the process of deleting the play, then the appropriate error is outputted directly from SQL.

deleteGame():

Submits a request to the database to delete all Games from the Games table. If there is an error in the process of deleting the Game, then the appropriate error is outputted directly from SQL.

readPlayers():

Opens the players.txt file, reads it line by line, parsing the lines into mini-arrays in one very large array (basically a 2D array) and then executes the SQL for each element in the large array.

readPlay():

Opens the play.txt file, reads it line by line, parsing the lines into mini-arrays in one very large array (basically a 2D array) and then executes the SQL for each element in the large array.

readGame():

Opens the game.txt file, reads it line by line, parsing the lines into mini-arrays in one very large array (basically a 2D array) and then executes the SQL for each element in the large array.

bulkPlayers():

Executes the sql ‘LOAD DATA INFILE’ query for players.txt. To get this to work on your computer the directory will need to be changed. (I also had to change the private file security setting in the root directory for mySQL.)

bulkPlay():

Executes the sql ‘LOAD DATA INFILE’ query for play.txt. To get this to work on your computer the directory will need to be changed. (I also had to change the private file security setting in the root directory for mySQL.)

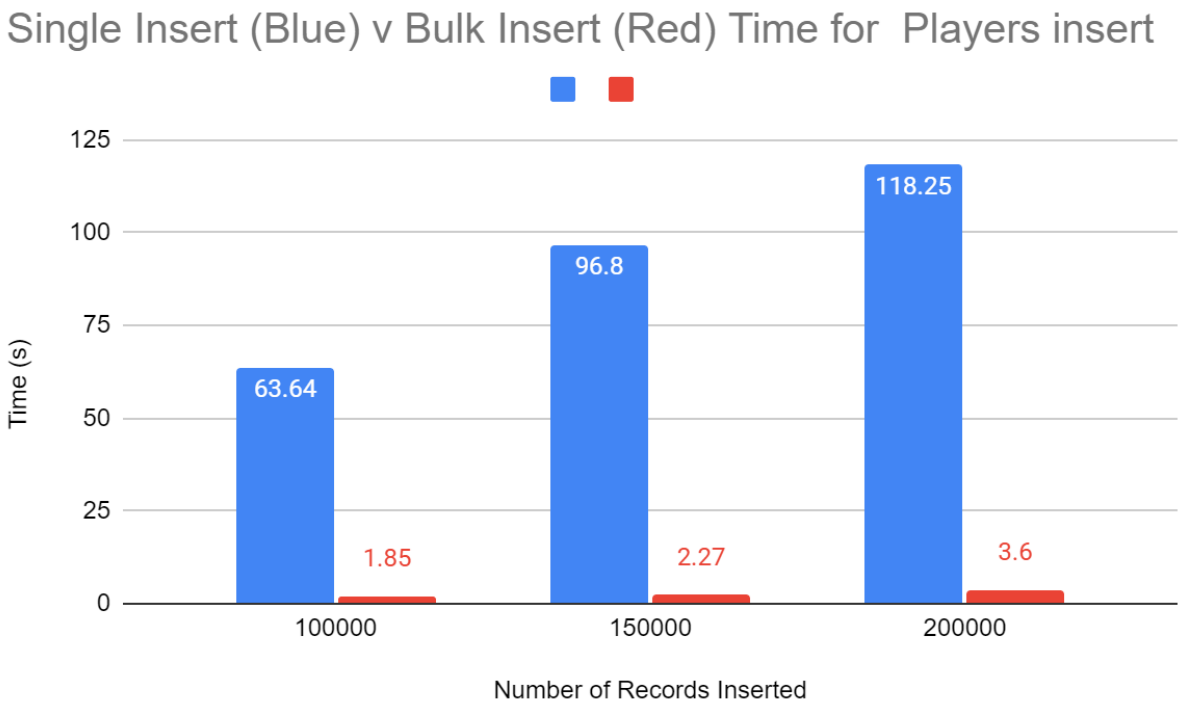
bulkGame():

Executes the sql ‘LOAD DATA INFILE’ query for games.txt. To get this to work on your computer the directory will need to be changed. (I also had to change the private file security setting in the root directory for mySQL.)

Analysis:

Files that my code ran on can be found in my submission zip folder as well as the code used to randomly generate the values that are being put into the tables / database.

Players Insert (Bulk v Single insert):



Play Insert (Bulk v Single insert):

Game Insert (Bulk v Single insert):

In all cases, Single insert takes significantly more time to execute compared to bulk insert. This is likely because we are parsing the files on our own, and inserting them one by one into the

database. On the other hand, the Bulk Insert is a built-in function of SQL and is likely highly optimized for inserting significantly large amounts of data.