Name: Brian Jon Stout
Date: 03JUNE2017
Current Module: SQL Development
Project Name: Sports_league

Project Goals

The goal of this project is to create tables for a database based on a sport; fill those tables with data about teams, players, and matches; and then create SQL scripts that query for specific sets of information based off that data.

Considerations

- The assignment specifies at a minimum what fields the table require
- The table for player attributes and the table game results both contain columns that exist in other tables (there's a relationship.)
- Table normalization

Initial Design

The database is populated using information from NHL.com and contains names and player information from actual NHL teams and players. The database is only loosely based on the actual game beyond basic goal tracking since implementing overtime losses, and shootout losses would have made some steps of the program insanely more complicated.

The database contains five tables; games, playerattributes, players, results, and teams.

The teams table contains a list of all the team names, has a unique number associated with each name acting as the primary key. The team name is limited to 30 character. The fields are teamId, and teamname.

The players table contains a list of all the players in the system. Information also in the table is their first and last name, the team they play on if they have one, and a jersey number (playernum) associated with that team which is required if they are on a team. The fields are playerId (primary), firstname, lastname, team, and playernum.

The playerattributes table contains additional information about a player. It contains a foreign key relating back to the playerId primary key in the players table set to cascade on updates or deletes so anything affecting the players table correlates back to the playerattributes table appropriately. The fields in this table are, playerId (foreign), age, height, and weight. Height is in inches and weight is in pounds.

The games table contains a list of games that have been played by the teams. It has a unique gameId for each one serving as the primary key. It also contains a datetime field called gameday which has the year-month-day hour-minute-second for when the game was played. It also has fields relating to teamId in the teams table which serves to keep track of which team

was home or away. I couldn't get foreign keys to work for these columns because they don't match up to the teamId column in the teams table. The fields in this table are, gameId, gameday, homeId, and visitId.

The results table contains a list of every game played by every team and the out come of that game. The table keeps track of three fields for this; pointsFor, which specifies how many points the team scored; pointsLost, which specifies how many points the other team scored on the; and result, which is a three letter string specifying the outcome of the match (WIN or LST.)

Data Flow

Each query is designed to meet the requirements in document provided by the teacher in class. All of the queries are individually stored in a sql script (stepXX.sql.) Scripts 1-5 set up the tables for the database. Script 6 populates the tables with the minimum required information to test the database. Script 7-22 query for specific information. Script 23 drops all the tables from the database correctly. Script 24 displays every single table in the database. Script 25 is a challenge which uses python to prompt the user for a table name, then outputs a one line SQL script which prints out all the rows in the table (if it exists.)

There is an extra script called step00.sql, which will reset the database fresh, and run up to step 6 populating the tables with information again. It also contains commented out code that runs the rest of the scripts in order, except for 24 and 23 since you can't print out tables if the tables get dropped before hand.

Certain queries which are designed to find the max value based off a group, all use the same trick. First you output a table which lists the comparison by each group, then you order it descending by the comparison, then you limit the output to only one entry, so only the highest value is displayed.

Communication Protocol

N/A

Potential Pitfalls

- Inputting incorrect information when hand jamming insertions
- Slow queries
- Selecting data from multiple tables
- Making sure a player has a jersey number if he has a team

Test Plan

User Test:

1. Run each step 0 - 6 individually to see if data all gets inserted into the table properly
2. Run each step after that individually to make sure proper output happens correctly
3. Run all the steps in a row to check for any errors

Test Cases:

1. Ran each table creation till the errors stopped occurring, and the expected results happened
2. Ran each step and adjusted queries till they worked properly
3. All the table creations, insertion, and queries run as expected

Conclusion

This project really showed how tedious database work can be. Jamming each of the values in individually is pretty annoying. Inserting based off a selection from an existing mySQL database might have been less time consuming in the long run, but more complicated to figure out at first. After the project it's safe to say that I could query for most data from a well designed database, but effectively setting one up is a little more complicated, and running efficient complicated selection is most likely beyond me.