Name: Daniel Ems
Date: 06/04/2017
Current Module: SQL
Project Name: Sports League

Project Goals:

The goal of this project was to successfully complete the steps required in the instructions. This involved creating tables and the appropriate relationships between each other. As well as, successfully querying the information requested.

Considerations:

1. What will be the design of your tables?
2. What will be the primary keys?
3. What will be the foreign keys?
4. Are the tables in 1NF?
5. Are the table in 2NF?
6. What naming conventions will be used?
7. What data will you populate?

Initial Design:

The initial design of the project involved reading through the requirements in the handout and breaking out each detail down to the column and record. On a separate sheet of paper, the tables were drawn, columns were labeled, and primary and foreign keys were identified. Once the tables and their information was identified, the data to be inserted was created. The reason the tables, keys, and relationships were identified prior to writing code was so any design flaws could be identified early. This ensures that as you write code you have access to all the information necessary as you make queries throughout your program.

Data Flow:

Throughout the program a total of five tables are created. In addition to the tables a total of ten views are created to aid in certain queries. Within the tables there are two constraints. Teams is a parent to players. The teamid in players serves as the foreign key to teams. This was established to ensure the players could be connected to their teams. The second is the player id serves as the foreign key from player info to the players table.

The views that are created serve the purpose of storing data in orders or groups that is not provided by the tables. This allows for additional queries after the data is isolated or parsed into an format easier to handle.

Potential Pitfalls:

The potential pitfalls are definitely within sub queries, and queries that involve two or more tables. This forced you to understand how to pick out the information you needed and use that as a key to access information from a second table. A large solution to this issue was views. Views allowed you to store information as a physical instances that you could then go back and query again. The importance here was ensuring you had enough unique identifiers to fulfill queries as you moved from table to table accessing the necessary information.

Test Plan:

User Test:

The test plans for each step varied in scope. The easier, and more strait forward steps were simple; just query the data. The tests cases that were most notable were complex queries that required information from multiple tables. An example of this can be the tallest player. While this is not the most complicated query, it serves to illustrate the approach taken to complete the query.

1. try and find the tallest player (select max(height) from playerinfo;).
2. try and attach a playerid to the output of step 1.

(select playerid, max(height) from playerinfo;.

3. Identify why this fails.
4. Try a sub query
   select playerid, height from (select max(height) height from playerinfo), playerinfo;
5. Try again (select playerid from playerinfo where height = (select max(height) from (playerinfo));
6. Try and identify player by name
7. hard coding playerid (select concat(p.firstname, " ", p.lastname) name, I.height from players p and playerinfo i where p.playerid = I.playerid and I.playerid = "hard coded";
8. combine both queries. SELECT CONCAT(p.firstname, " ", p.lastname) name, I.height FROM players p, playerinfo I WHERE p.playerid = I.playerid AND I.playerid = SELECT playerid FROM playerinfo WHERE height = (SELECT MAX(height) from playerinfo));

Test Cases:

The tests queries as employed were designed to take apart each query to their simplest form. The first question to ask oneself is "How do I access a single element needed?". Once you can pick out one piece of data, you then reach for a second. This approach is very helpful in making sure that you are constantly stepping through the query as needed. In addition, it allows you to develop sub queries as you pick through your data.

Conclusion:

The project did a great job of tying together SQL. Most of which is due to he need to perform numerous different tools, functions, and elements of SQL in order to accurately query information. This ranged from design concepts, constraints and keys, all the way to sub queries, joins and views.