League of Legends: Online Matchmaking Database System



Designed By: John Kryspin

Table of Contents

Executive Summary\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3

Entity Relationship Diagram\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4

itemBaseProperties\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5

currGameItems\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6

currentGame\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7

championBaseProperties\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8

Player\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9

proPlayer\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

proTeam\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11

matchScheduler\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11

matchHistory\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_12

gameStatistics\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_12

Player and Champion\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_13

Players In Game\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_13

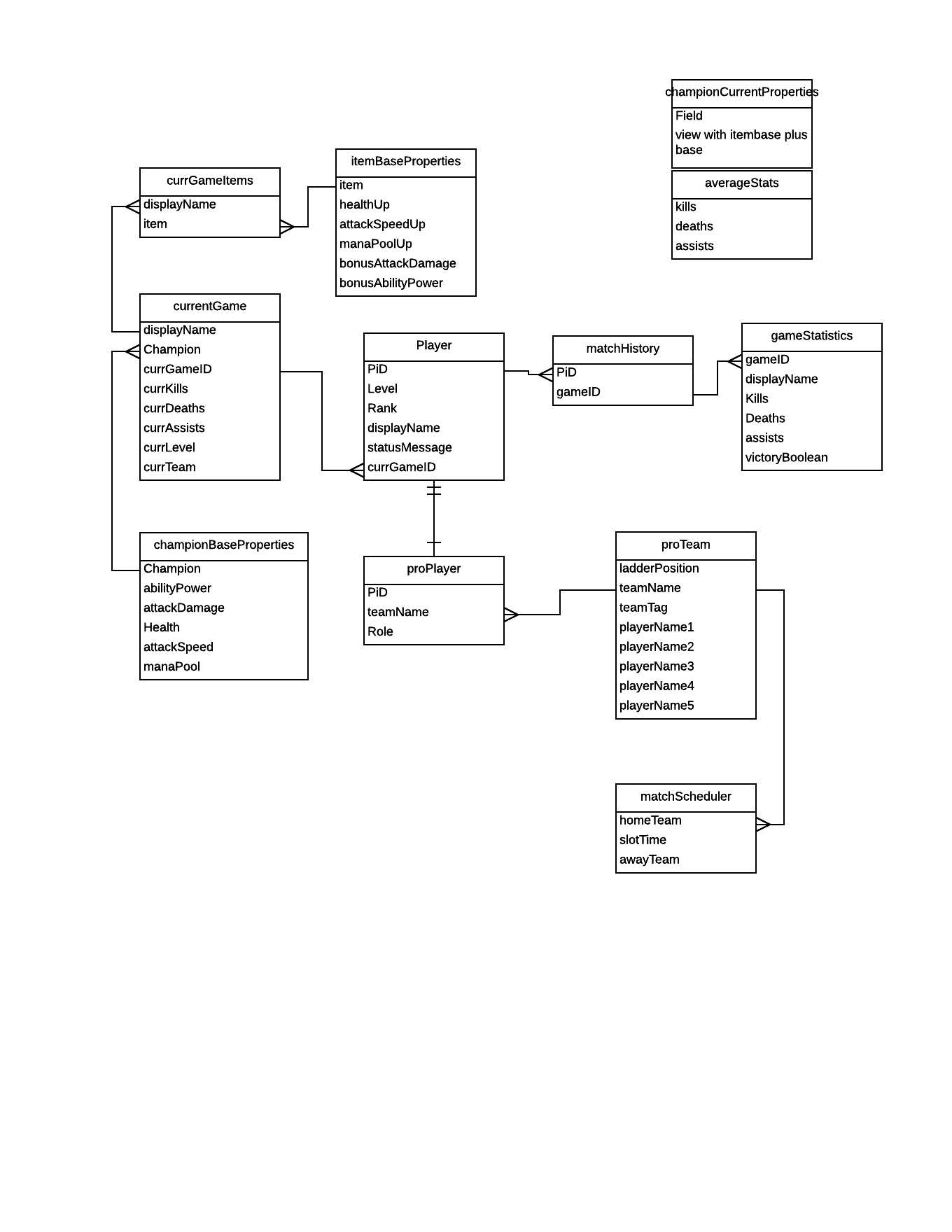
Current Champion\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_14

Security\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_15

Implementation notes/Know problems/Future Enhancements\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_16

Executive Summary

This entire document shows how League of Legends matchmaking in their online system works and how the data is stored in a way that makes sense and is easy to look through. The database shows players interactions with each other, their items, and their champions and has scheduling for pro teams.

Entity Relationship Diagram

Tables

**itemBaseProperties table**

This table lists every item in the game and the properties attached to it. Each champion can buy items to use in the current game and it affects their champion’s stats.

create table itemBaseProperties(healthUp int,

attackSpeedUp decimal,

manaPoolUp int,

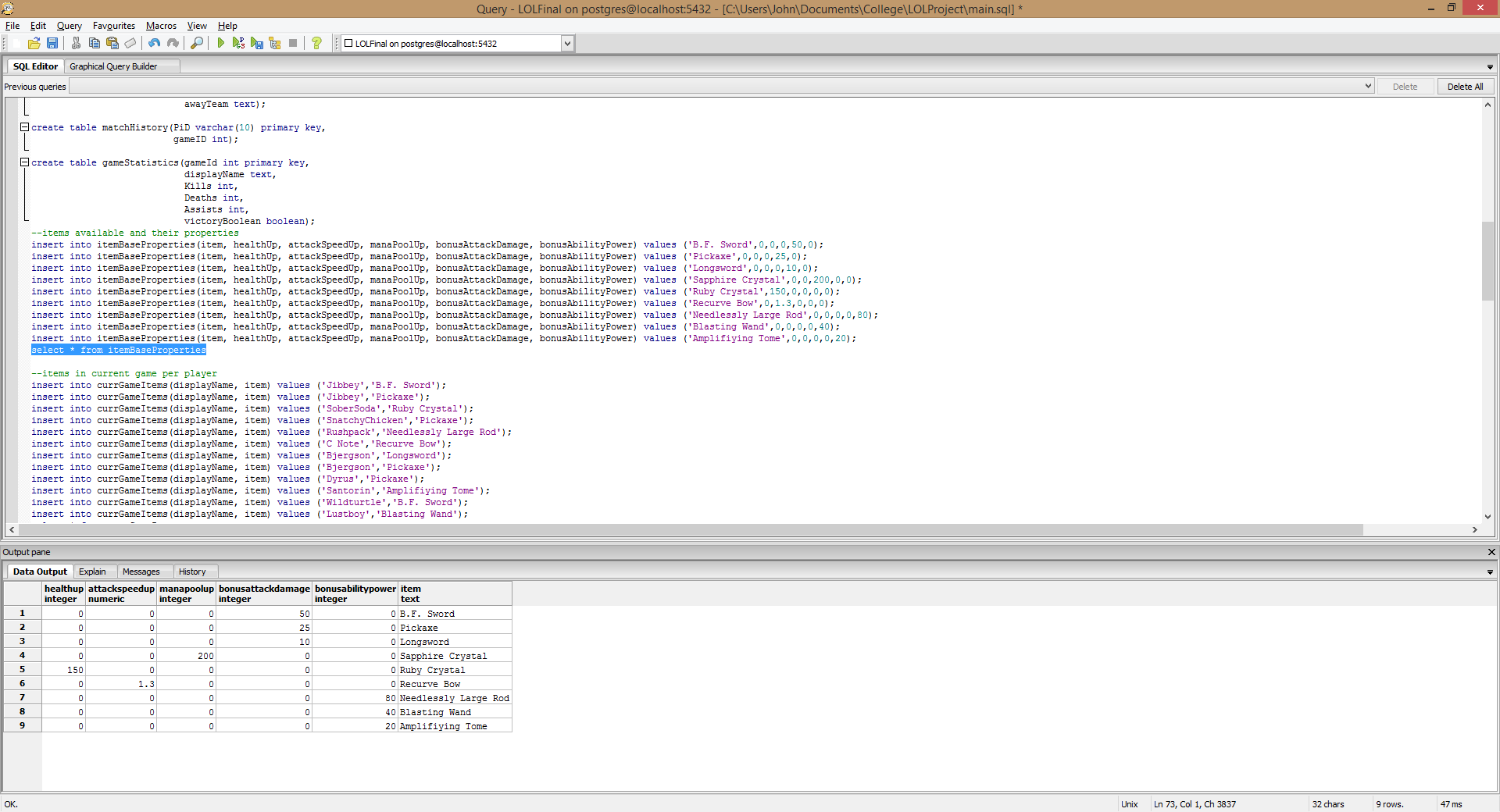
bonusAttackDamage int,

bonusAbilityPower int,

item text primary key);

Functional Dependencies: item-> healthUp,attackSpeedUp,manaPoolUp,bonusAttackDamage,bonusAbilityPower

Sample Data:



**currGameItems table**

This table uses each person’s display name and item as a composite key. This is necessary because one displayName can have multiple items which is why having just displayName as the primary key would not work.

create table currGameItems(displayName text,

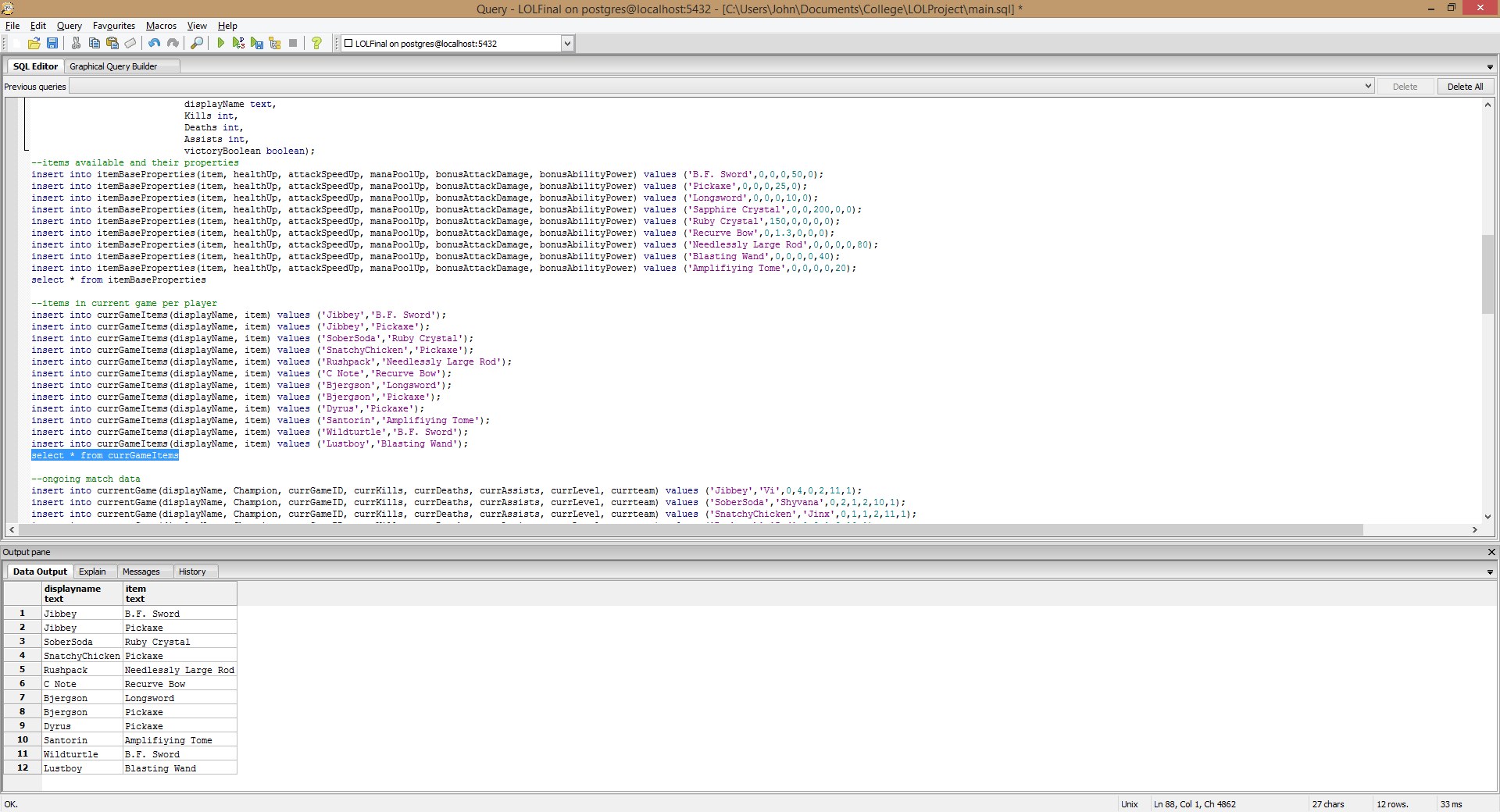
item text,

primary key (displayName, item)

);

Functional Dependencies: displayName,item->

Sample Data:



**currentGame table**

This tracks the current data in an ongoing match. This table would be updated every time a champion buys an item gets a kill or dies or gets an assist or gains a level. The example is just a snapshot in time for one game that is ongoing.

create table currentGame(displayName text,

Champion text,

currGameID int,

currKills int,

currDeaths int,

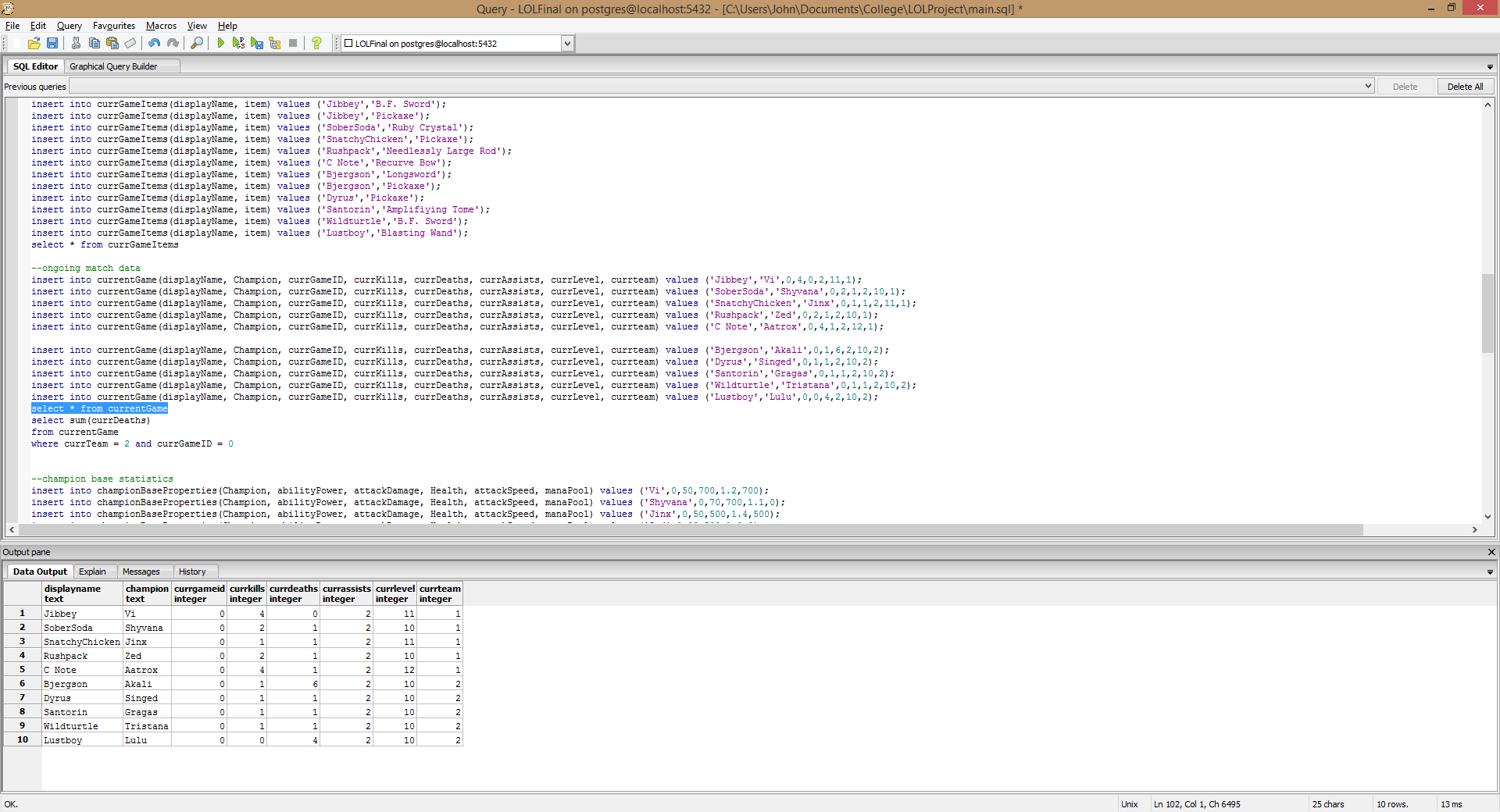
currAssists int,

currLevel int,

currTeam int,

primary key(currGameID, displayName));

Functional Dependencies: currGameID,displayName ->Champion, currKills, currDeaths, currAssists, currLevel, currTeam

Sample Data:

**championBaseProperties table**

This table shows the base stats of each champion in the entire game. This table would be static and not change but rather a view would be created to show a champions current statistics with items and such.

create table championBaseProperties(Champion text primary key,

abilityPower int,

attackDamage int,

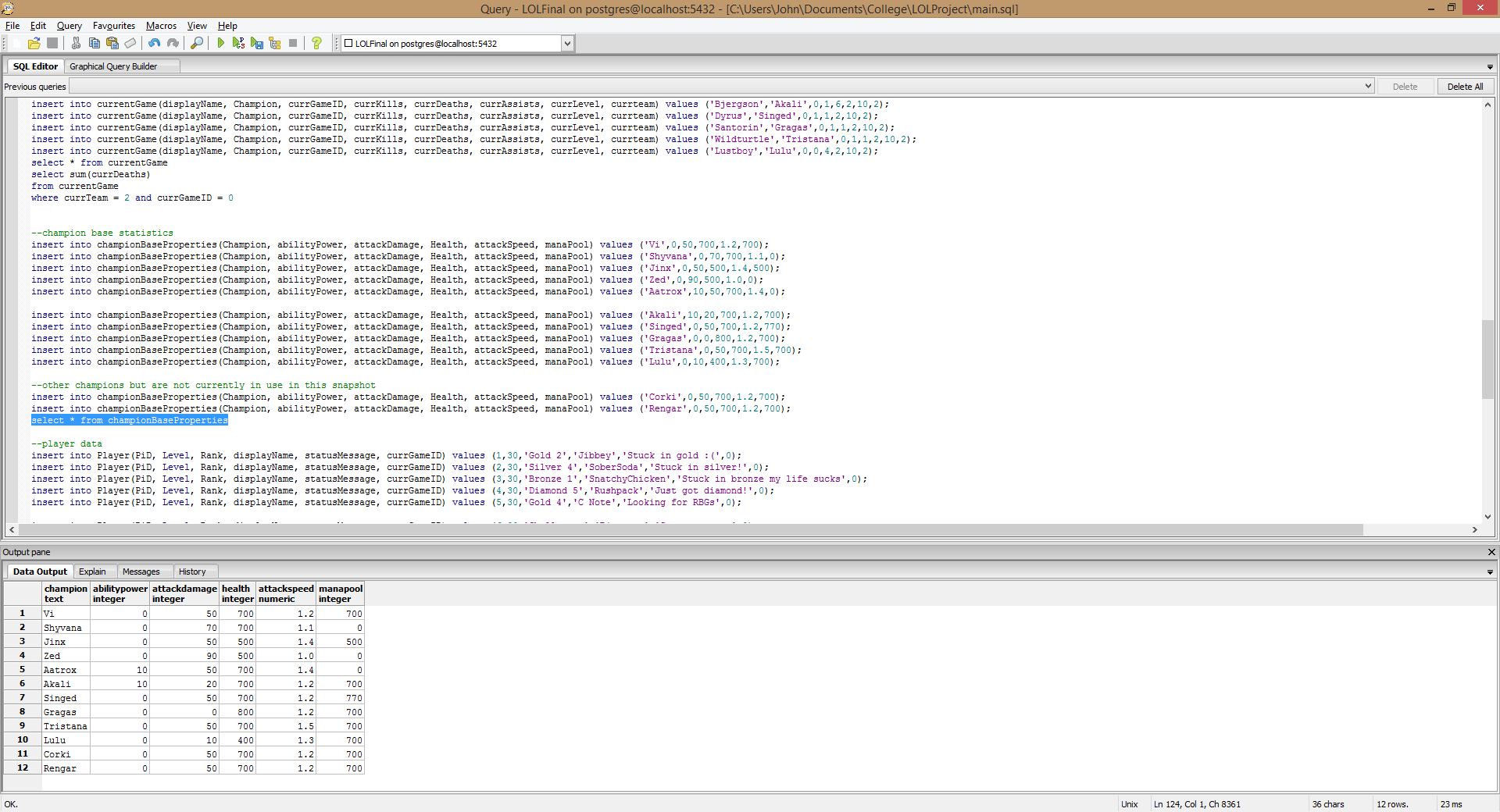
Health int,

attackSpeed decimal,

manaPool int);

Functional Dependencies: Champion -> abilityPower, attackDamage, Health, attackSpeed, manaPool

Sample Data:



**Player table**

This table tracks basic statistics about the player and shows if they are currently in a game.

create table Player(PiD varchar(10) primary key,

Level int,

Rank text,

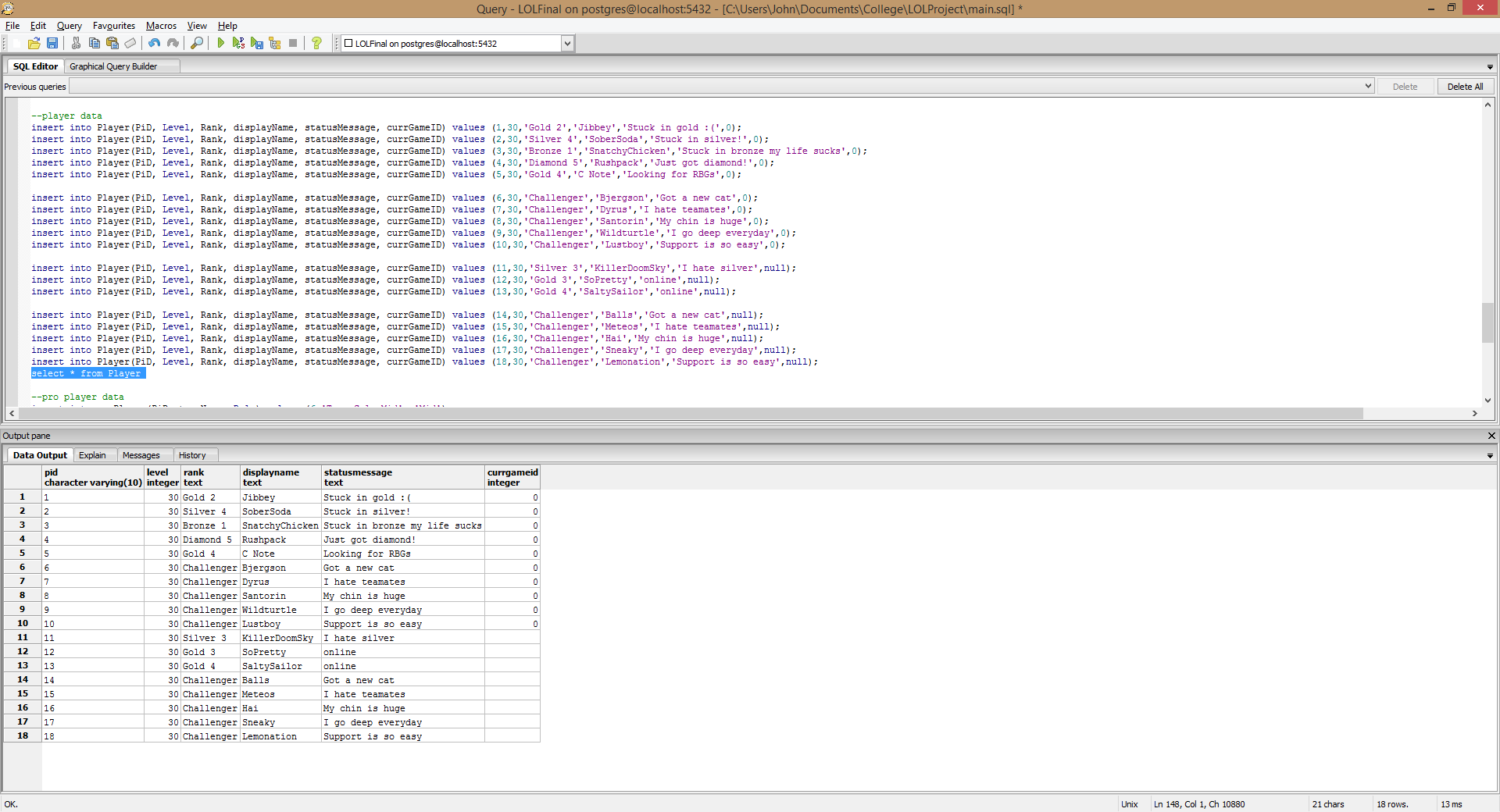
displayName text,

statusMessage text,

currGameID int);

Functional Dependencies: PiD-> Level, Rank, displayName, statusMessage, currGameID

Sample Data:



**proPlayer table**

This table inherits data from player because every proPlayer is a Player but not vice versa. It also shows which role they play in the team.

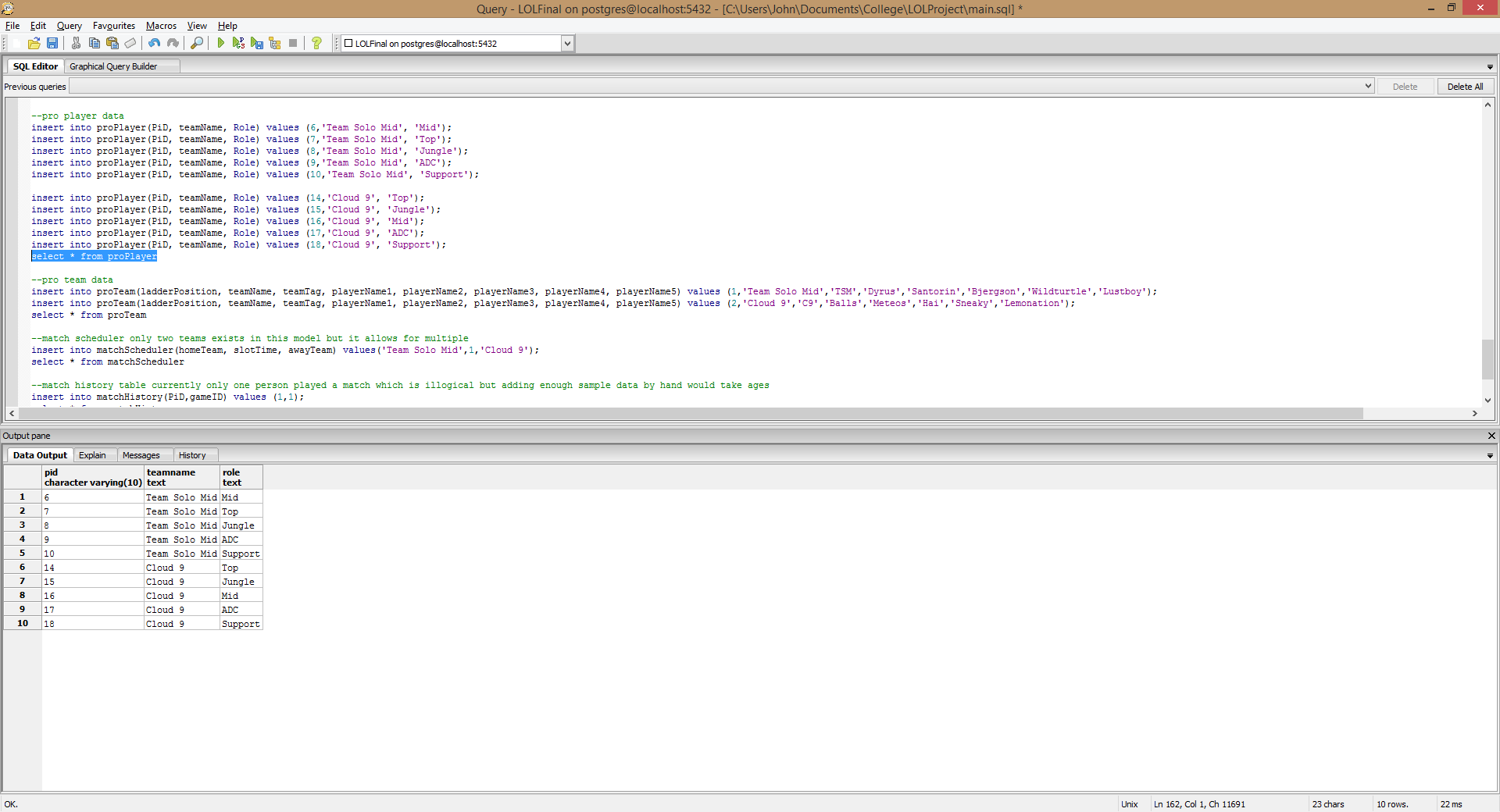
create table proPlayer(PiD varchar(10) primary key,

teamName text,

Role text);

Functional Dependencies: PiD->teamName,Role

Sample Data:



**proTeam table**

This table lists each professional team and their roster in the League Championship Series. Currently there are only two teams but adding more would work fine too.

create table proTeam(teamName text primary key,

ladderPosition int,

teamTag varchar(4),

playerName1 text,

playerName2 text,

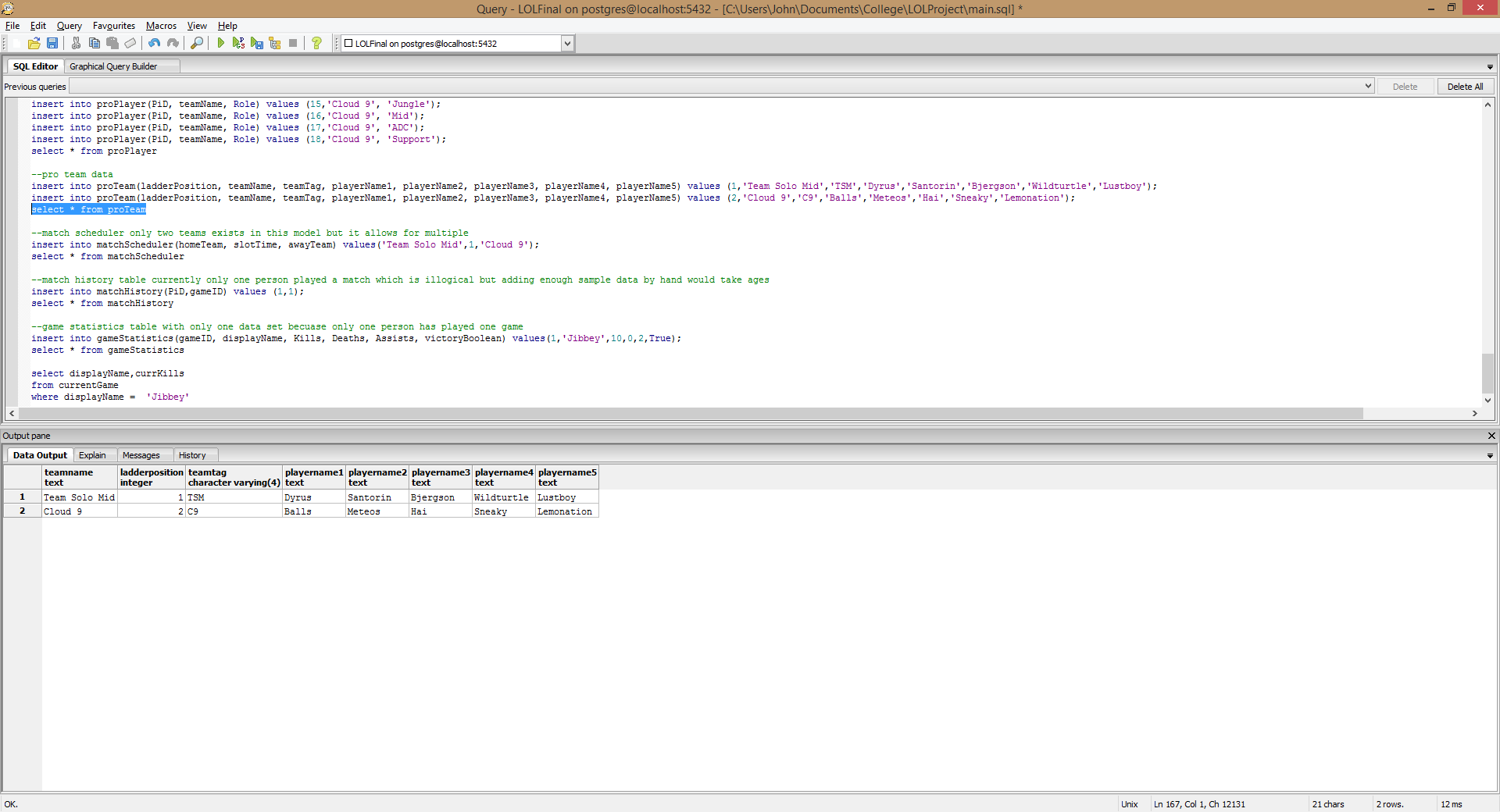
playerName3 text,

playerName4 text,

playerName5 text);

Functional Dependencies: teamName-> ladderPosition, teamTag, playerName1, playerName2, playerName3, playerName4, playerName5

Sample Data:



**matchScheduler table**

This table sets up matches between teams and users integer slots to determine the time they play. The sample data reflects one week.

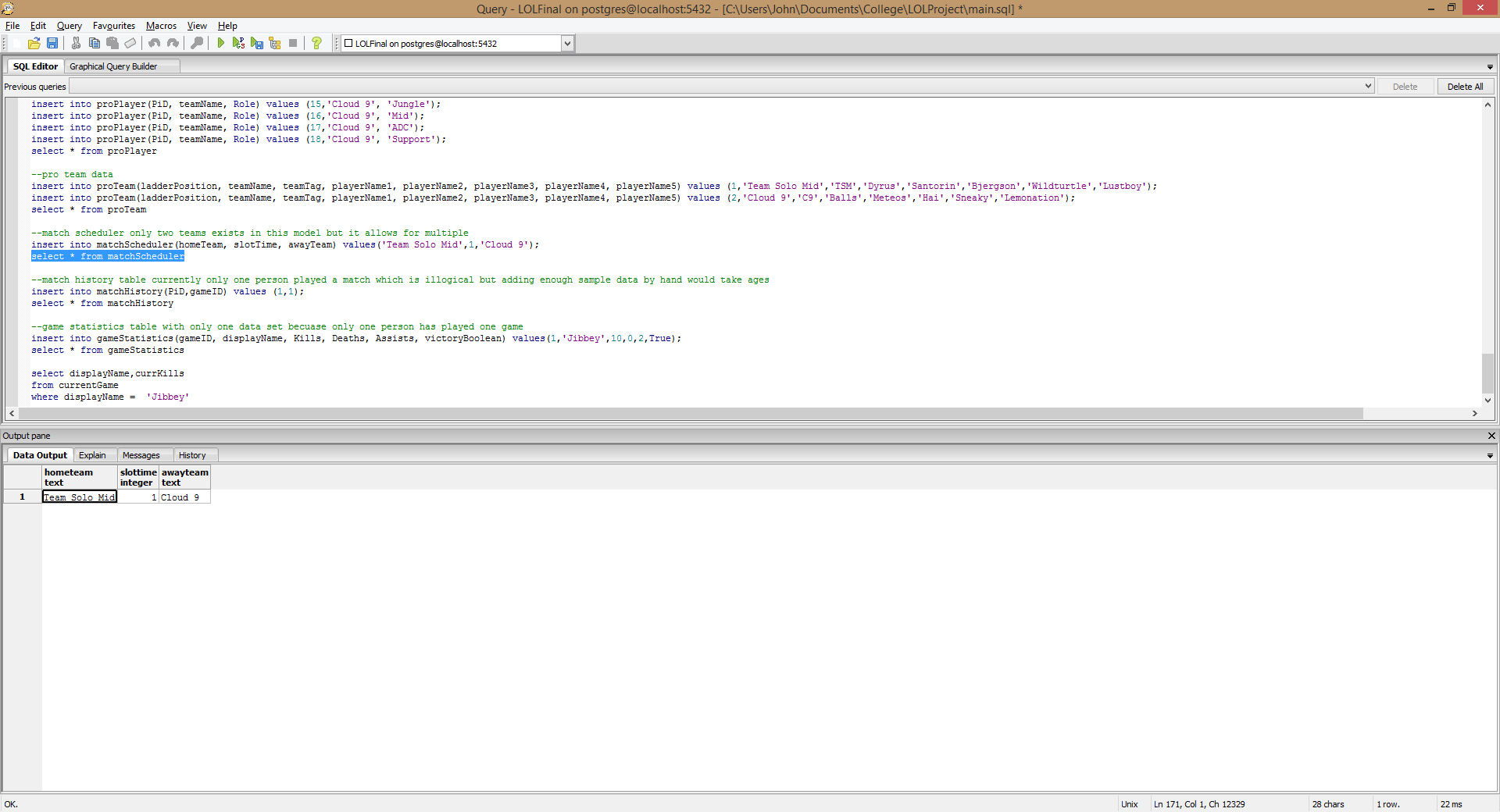
create table matchScheduler(homeTeam text primary key,

slotTime int,

awayTeam text);

Functional Dependencies homeTeam->slotTime, awayTeam

Sample Data:



**matchHistory table**

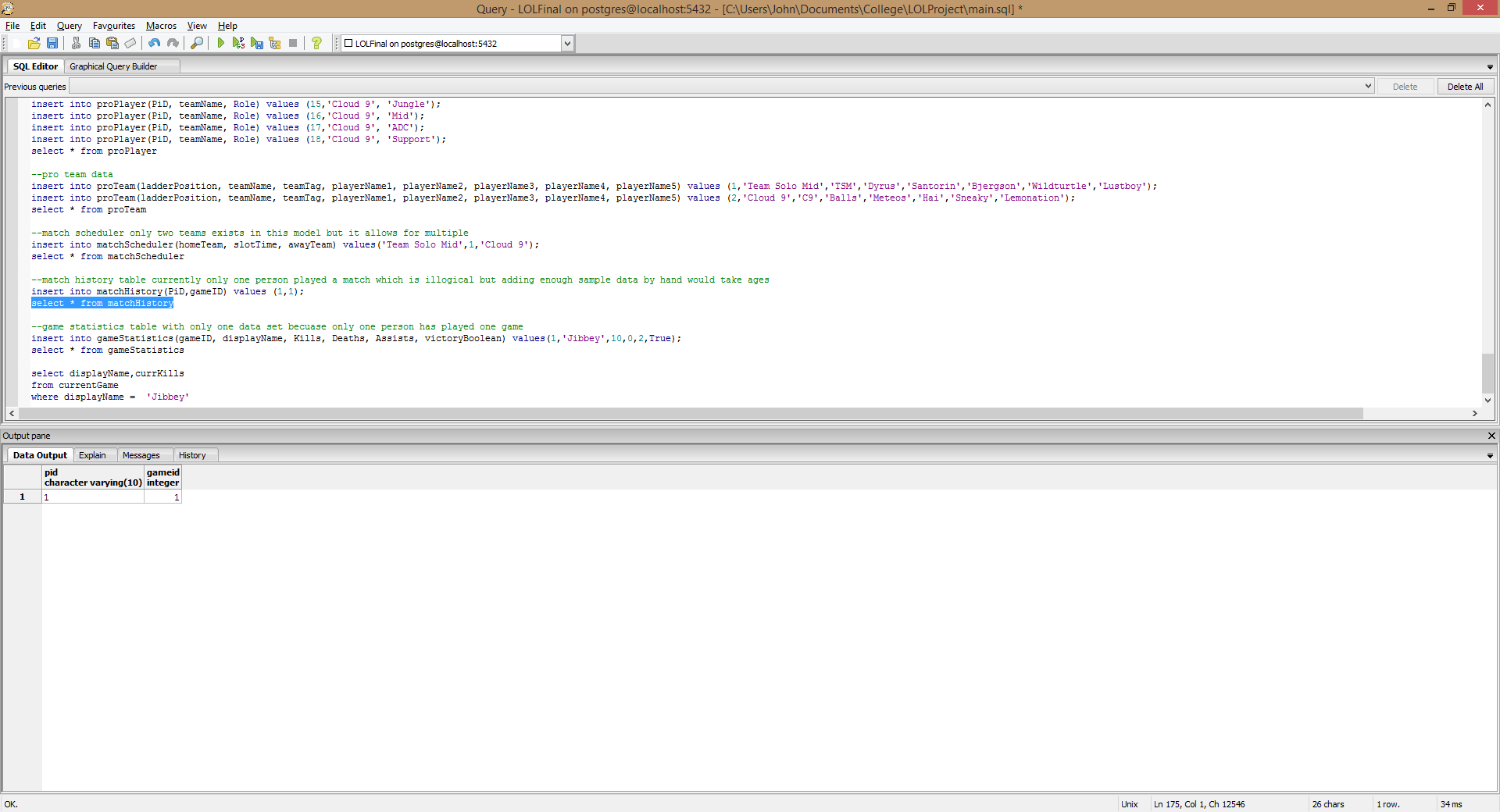
This table allows you to see how many games a player has played and links to another table which shows the stats for each game.

create table matchHistory(PiD varchar(10) primary key,

gameID int);

Functional Dependencies: PiD->gameID

Sample data:



**gameStatistics**

This table shows the details for each match a person has played and if they won or lost the match (boolean)

create table gameStatistics(gameId int primary key,

displayName text,

Kills int,

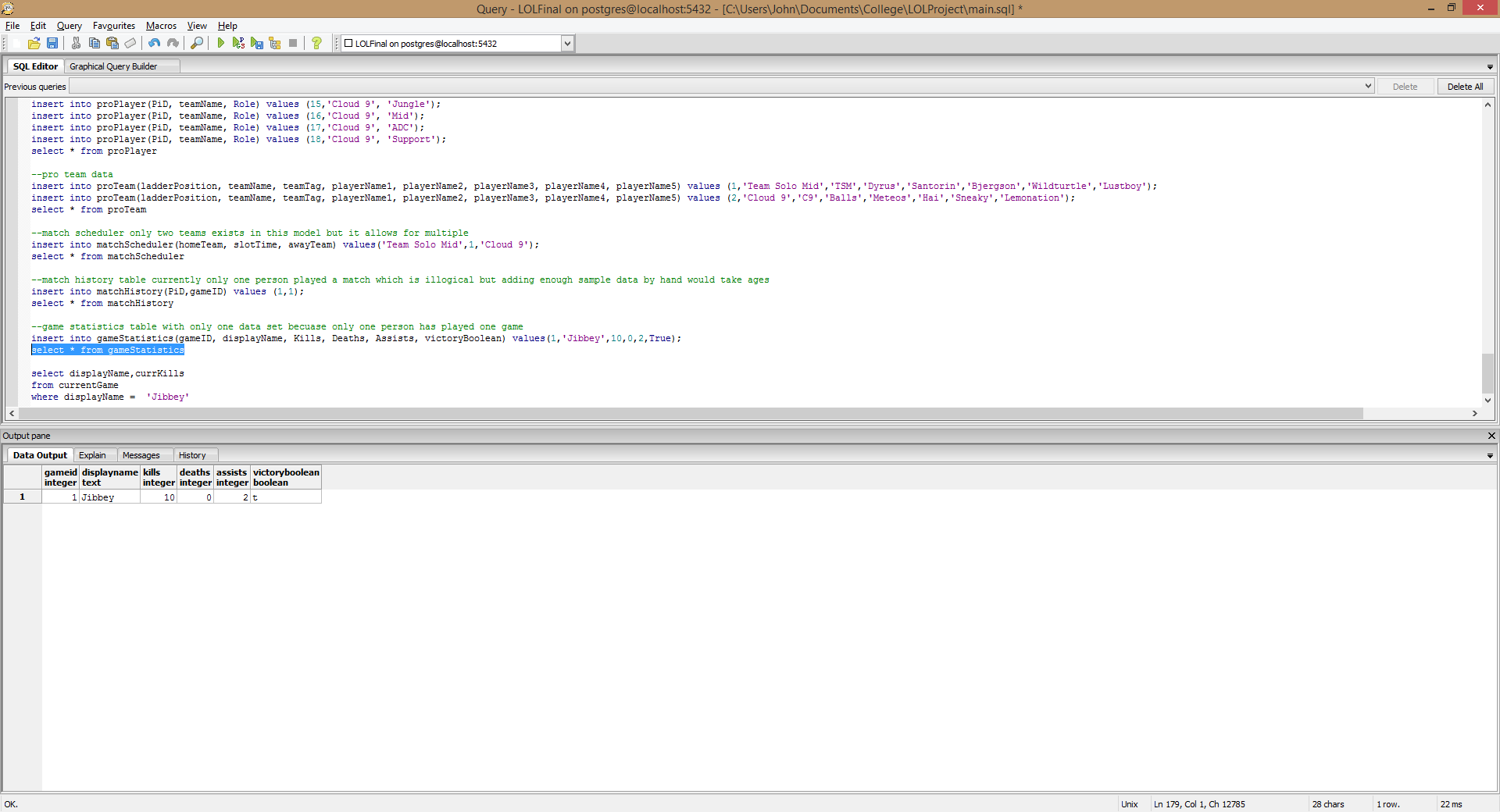
Deaths int,

Assists int,

victoryBoolean boolean);

Functional Dependencies: gameId-> diaplyName,Kills,Deaths,Assists,victoryBoolean

Sample Data:



**Views**

**PlayerAndChampion**

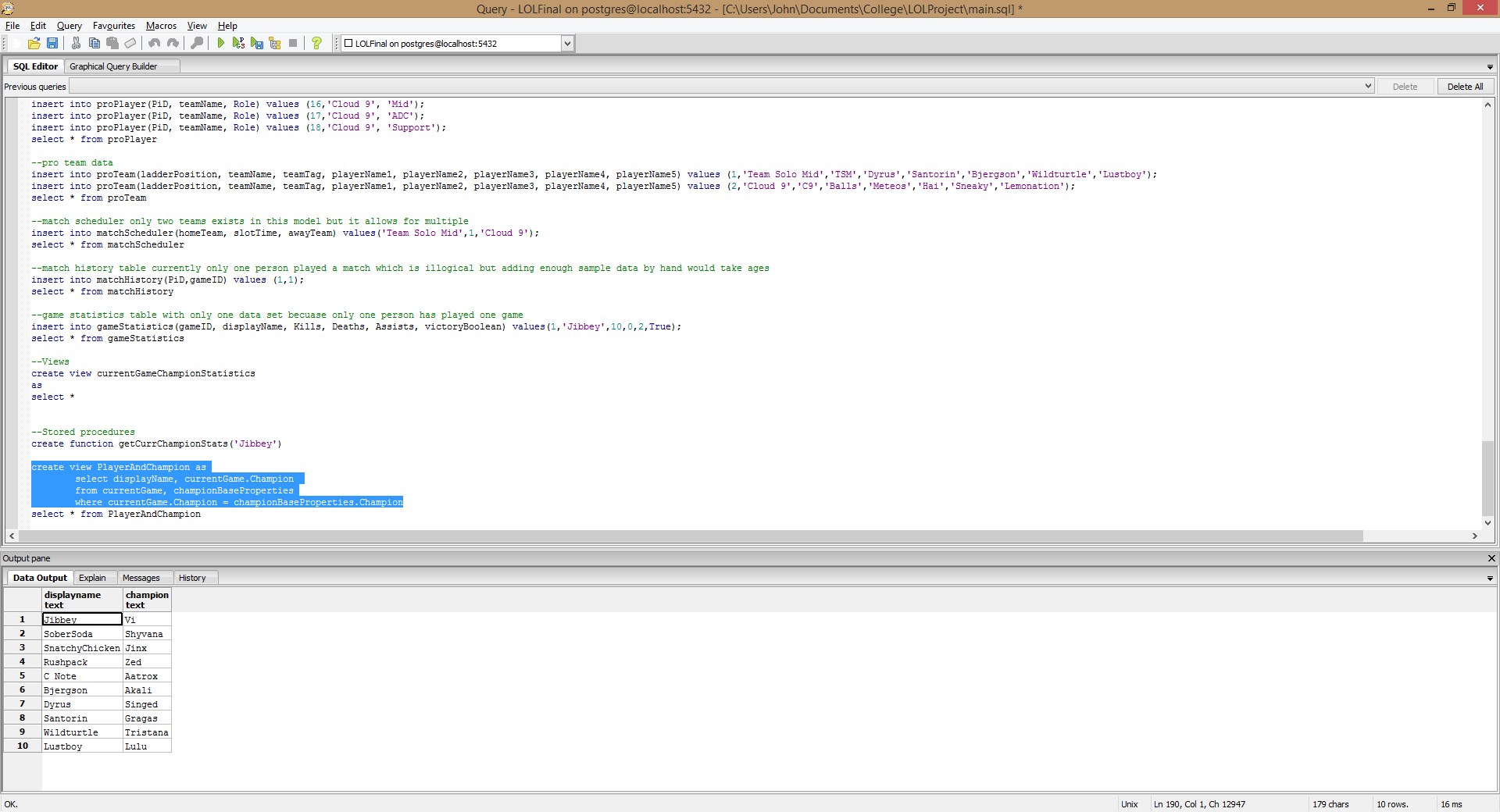
Shows the players and the champion they are playing

create view PlayerAndChampion as

select displayName, currentGame.Champion

from currentGame, championBaseProperties

where currentGame.Champion = championBaseProperties.Champion



**PlayersInGame**

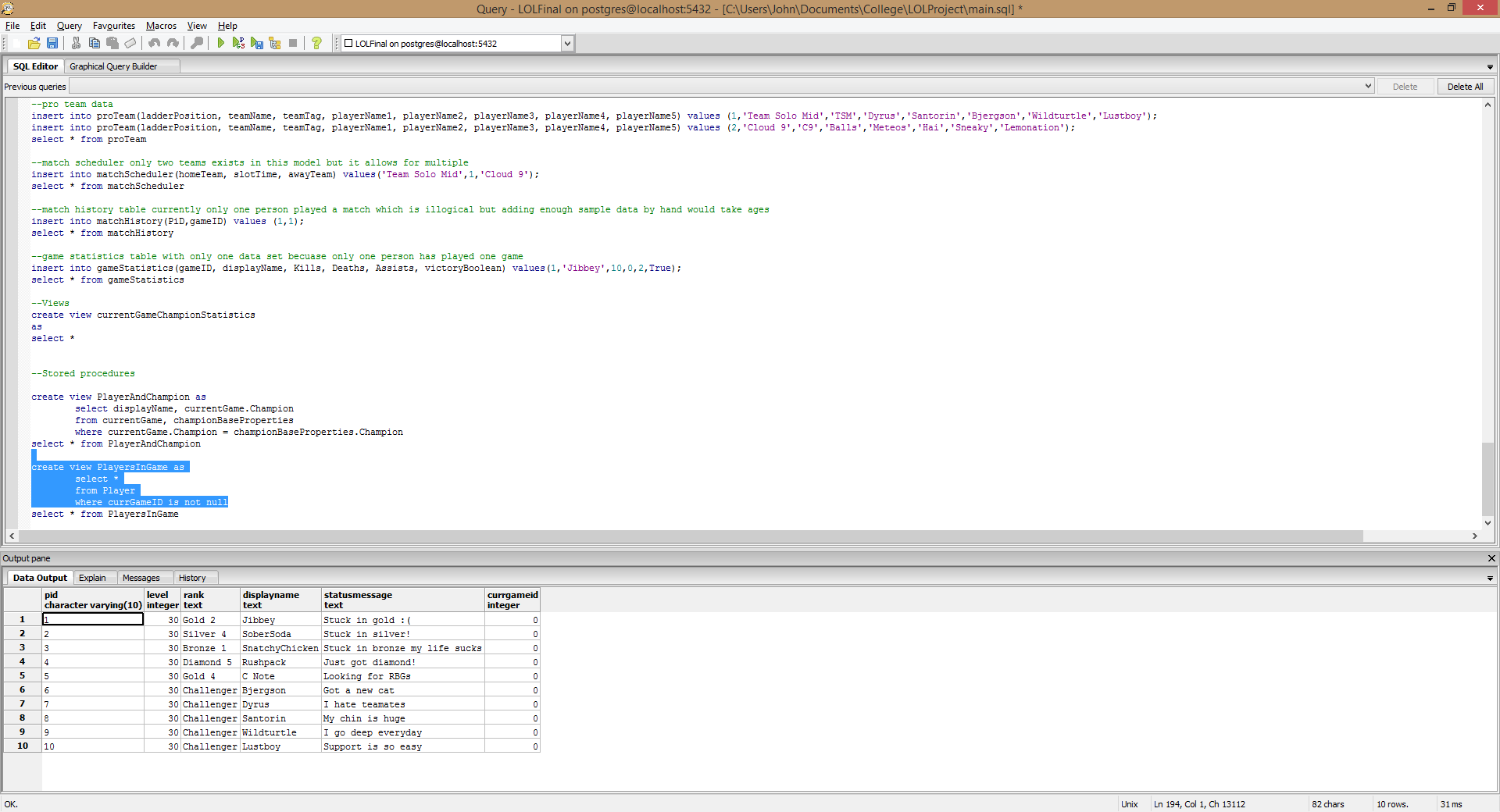
Shows only players who are currently in a game

create view PlayersInGame as

select \*

from Player

where currGameID is not null



**Stored Procedures**

**currChampion**

Finds a players current champion, takes in the players name as a parameter.

create or replace function currChampion(text, REFCURSOR) returns refcursor as

$$

declare displayNameVar text :=$1;

resultset REFCURSOR := $2;

begin

open resultset for

select displayName,Champion

from currentGame

where displayName = displayNameVar;

return resultset;

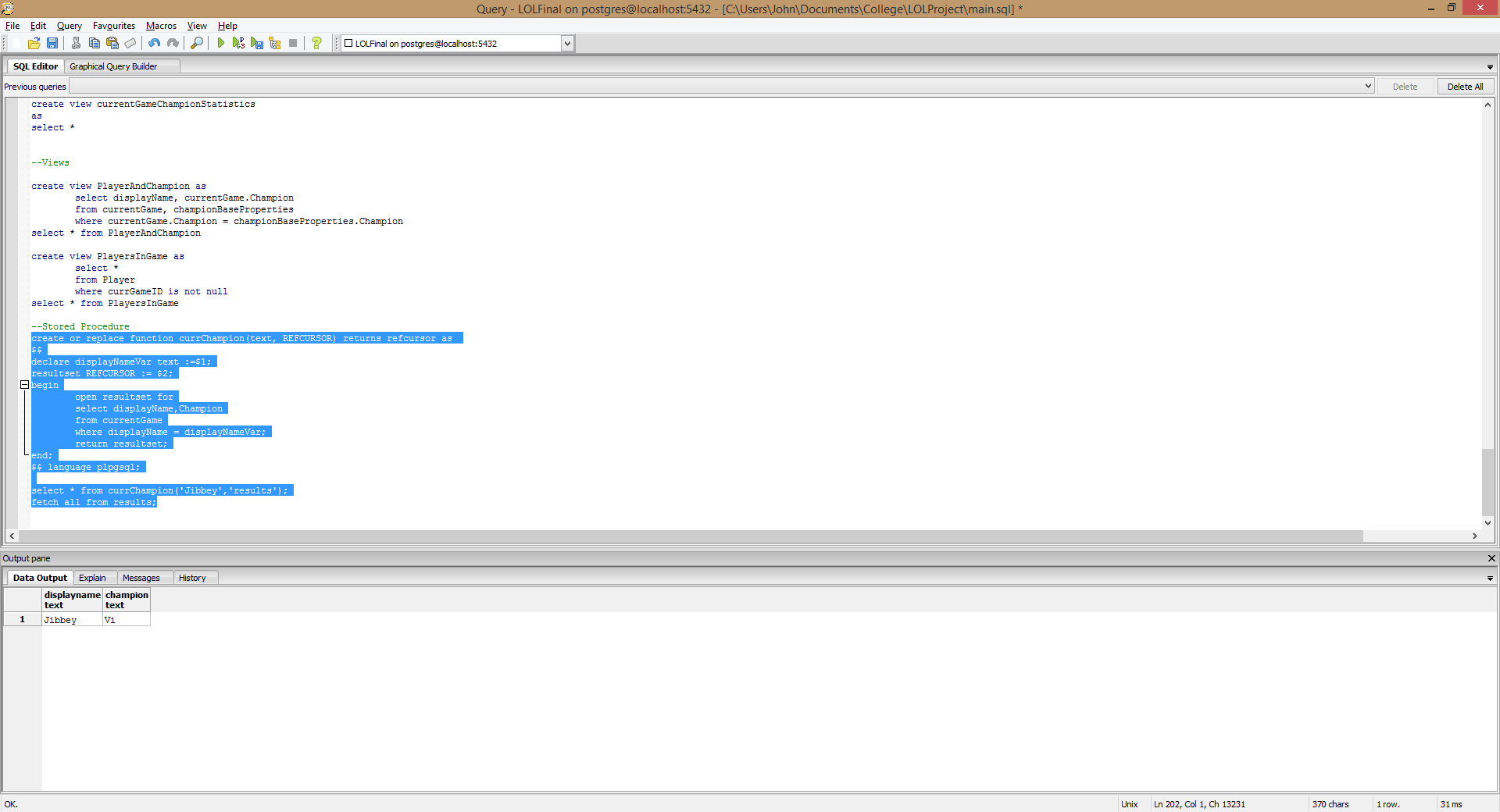
end;

$$ language plpgsql;

Sample Query:

select \* from currChampion('Jibbey','results');

fetch all from results;



**Security**

We cannot trust players to report their wins or loses so everything is handled by the administrator. But, LCS has a separate league going on which we need a role for. I create an LCSadmin role and he can only edit the proTeam and matchScheduler table so he cannot accidentally mess up the users public data.

CREATE ROLE admin;

GRANT SELECT, INSERT, UPDATE

ON ALL TABLES IN SCHEMA PUBLIC

TO admin;

CREATE ROLE LCSadmin;

GRANT SELECT, INSERT, UPDATE

ON proTeam, matchScheduler

To LCSadmin;

**Implementation Notes**

This system took a long time to implement. I barely scratched the surface with sample data and there could be millions of rows in these tables like there probably is a RIOT HQ. The biggest difficulty was setting up the currentGame table. This was hard because the data gets dropped once the game ends and is then pushed to match history which I have no idea how to implement but I believe that’s how it works. Putting in sample data took a long time to do, and I only used 12 champion of the 137 available that could all go into that table. I used my friends and I for the sample data and pro teams TSM and C9 for the pro team data. The stats are all realistic.

I don’t know of any know problems but I think that the matchScheduler could be more detailed and create time for certain teams and decide who is on what side (red or blue) per game. Also you could add best of threes into it somehow too.

There are many future enhancements that can be done. For one you could create a function that calculates the players current champion stats by adding the item data to the base stats data. I tried to do that but I could not figure it out after an hour of work on just that. I’m sure it is possible but I am not that proficient in SQL yet. There could be more matchHistory data I only have one game in there right now. This systems is only a starting point, the real system used by League of Legends is much more complex and detailed and I wish I could see there ER diagram to see how it actually works because it seems very complex after trying to make it simple in my project.