Level Down



By Richard Liao

Table of Contents

Executive Summary. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .2

Game Overview. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3

Database E/R Diagram. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .4

Database: Tables . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5-29

Database: Views. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .30

Database: Reports. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .33

Database: Stored Procedures. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36

Database: Triggers. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .37

Future Improvements. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .42

Known Issues. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43

Executive Summary

The goal of this database is to create a simple yet powerful way to store all actors and data in the game “Level Down”. While it is possible to store all of the parts of data as “objects” (i.e. Enemies and items) in Level Down, it is extremely inefficient and difficult to manage. A database is required to keep all objects and other events tracked such as high-scores or player data. This will allow an admin to utilize a special set of tools to manage “Level Down” and hopefully maintain balance where ever necessary.

Game Overview

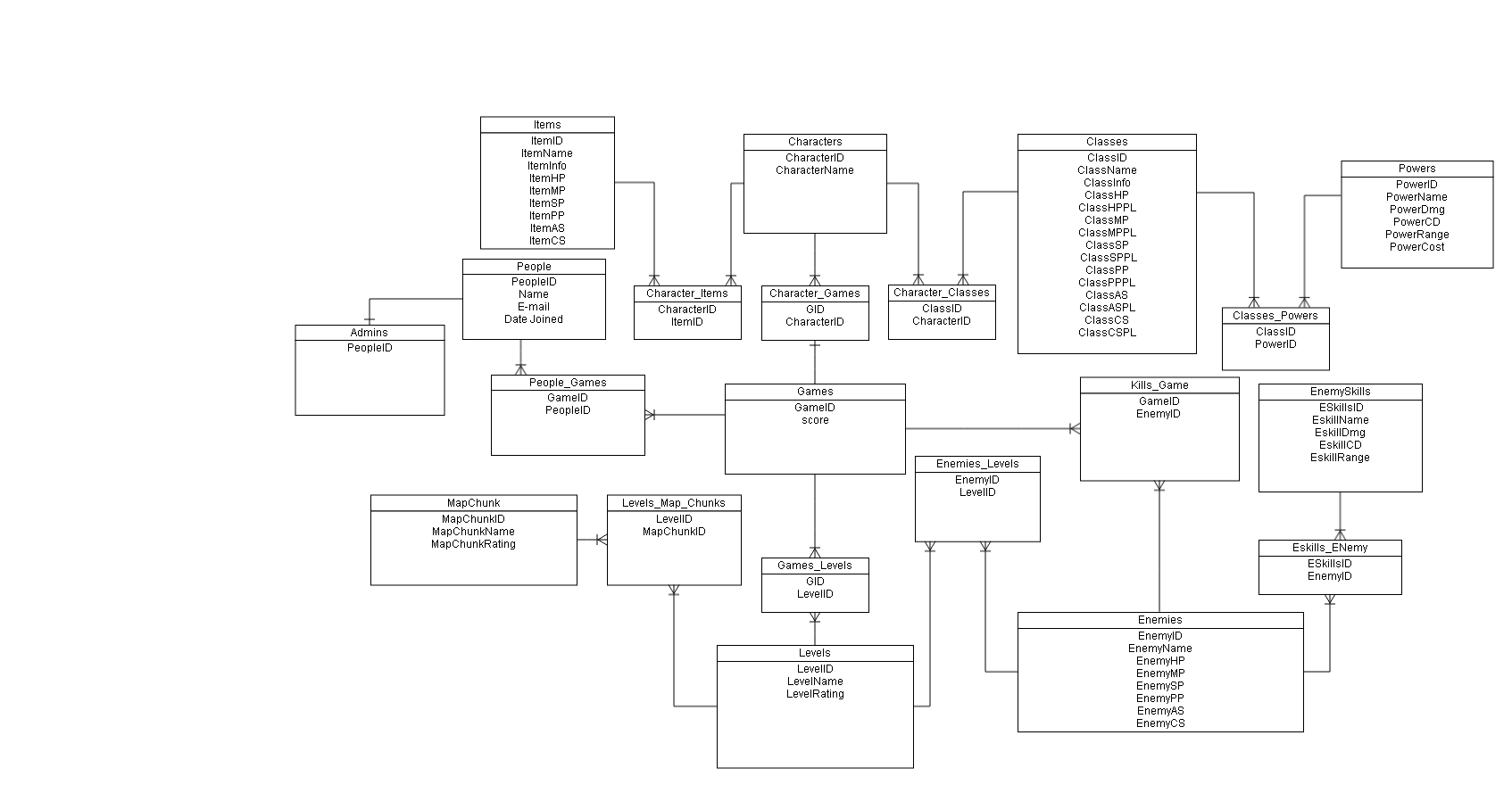
To fully understand this database and its existence, it is imperative the game is understood first. The premise of the game is as follows: You are the demon king, but instead of conquering and slaughtering the masses, you have opted to stay secluded and mostly uninvolved. This is a problem for the Hero’s Guild as they are not benefitting from the usual wartime economy. To “combat” your inaction, they have assaulted your castle in an attempted coup. Although you managed to repel the attack, the Guild Leader left you with a curse that drains your powers and a note detailing a back-up plan to start pillaging and pin the deaths on you, the demon king. You must now stop the guild’s leader before it is too late.

The main twist of the game is that instead of the normal convention of leveling up, the player will instead level down. The game is a rogue-like game; the player will have one chance to reach the end of a slightly randomized set of levels in order to beat the game. The player starts from their most powerful state and end up at their weakest. If they fail, their character is gone forever and they must try again. At the start, the player chooses up to 3 classes that make up a character. These characters start with a set of items and skills. Every time a player kills a boss or levels down, they will choose a power that they want to give up. However, whenever a power is lost by the player, the last boss gains it. This will force players to pick and choose powers based on their usefulness to their current item set, skill set, power, and what they want to sabotage the boss with. A power may eventually cost too much MP, so it may be beneficial to drop it because, while it does a large amount of damage, it is too costly to use and will hurt the final boss’ MP pool. Leveling down, instead of up, creates a new kind of strategy that can change on the fly.

The issue arises that a player might try to simply run past the level without killing anything. This is where the enemies play a large role. As the player proceeds, they will fight bosses that force them to lose a level (if they win). As such, their current equipment will no longer be useable as they have become too low level to use it. Enemies scattered through-out the level will drop gear, but also level the player down, creating a risk reward style of game play. Learning enemy attack patterns is much more important than anything else in the game; items and skills are unreliable.

Database: E/R Diagram

Note: This E/R Diagram is set up in a very awkward way; due to its size it had to be altered to fit in this document. It is still logically the same, even though the tables may not be presented in a way that is pleasing to the eye.



Database: Items Table

The purpose of this table is to store all the items that will be obtainable by the player. All items give the player HP(hit points), MP(Magic Points), SP(Spell power), PP(Physical power), AS(Attack Speed), and CS(Cast Speed). Items are tracked by a unique ID called ItemID, the primary key.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create table Items

(

ItemID SERIAL not null,

ItemName varchar (255),

ItemInfo varchar(255),

ItemHP int,

ItemMP int,

ItemSP int,

ItemPP int,

ItemAS int,

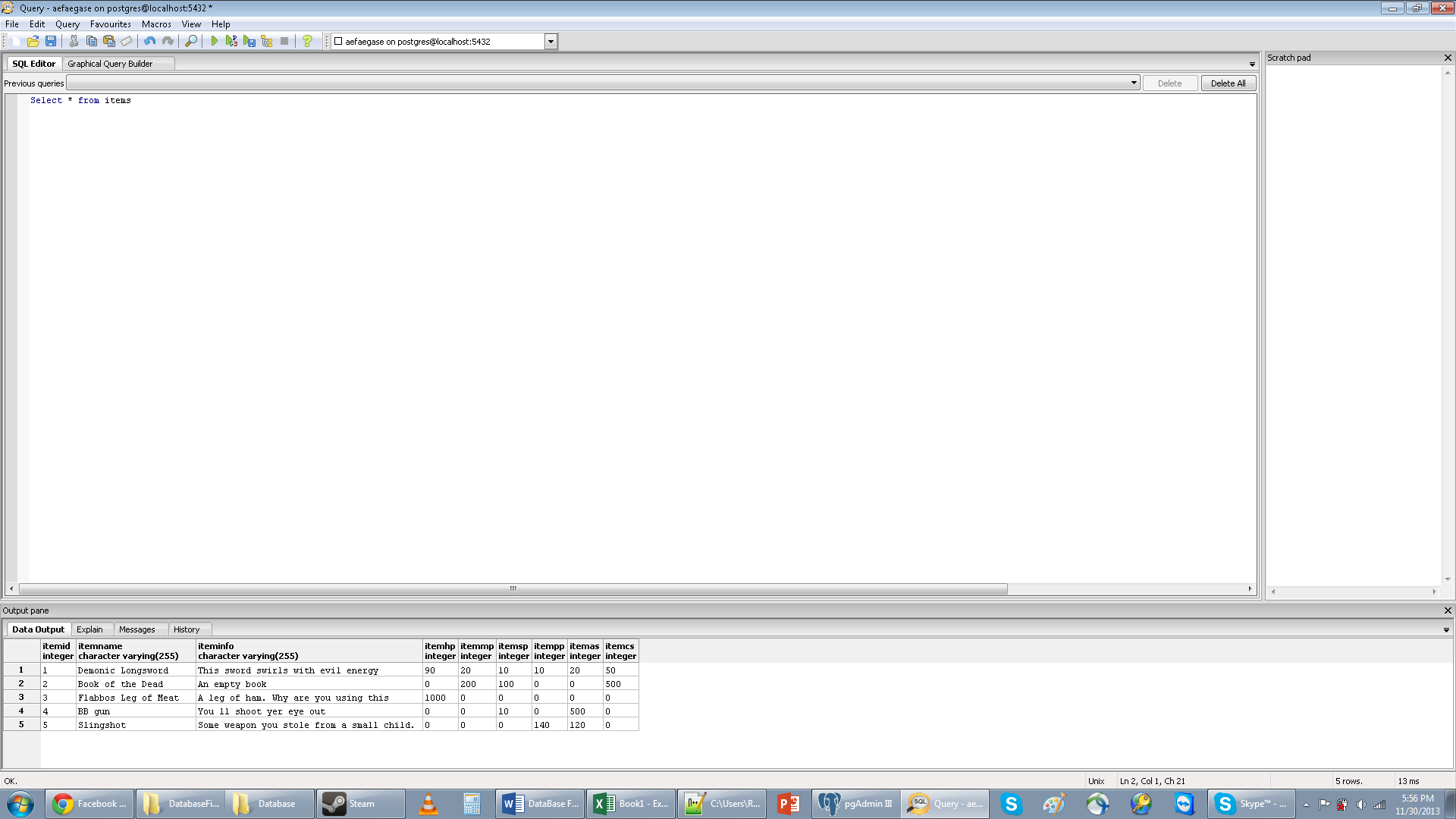
ItemCS int,

primary key (ItemID));

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ItemID🡪 ItemName, ItemInfo, ItemHP, ItemMP, ItemSP, ItemPP, ItemAS, ItemCS

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Characters table

This table will hold all names of characters. It will serve as the “hub” where characters and their information is stored.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create table Characters

(

CharacterID SERIAL not null,

CharacterName varchar (255) not null,

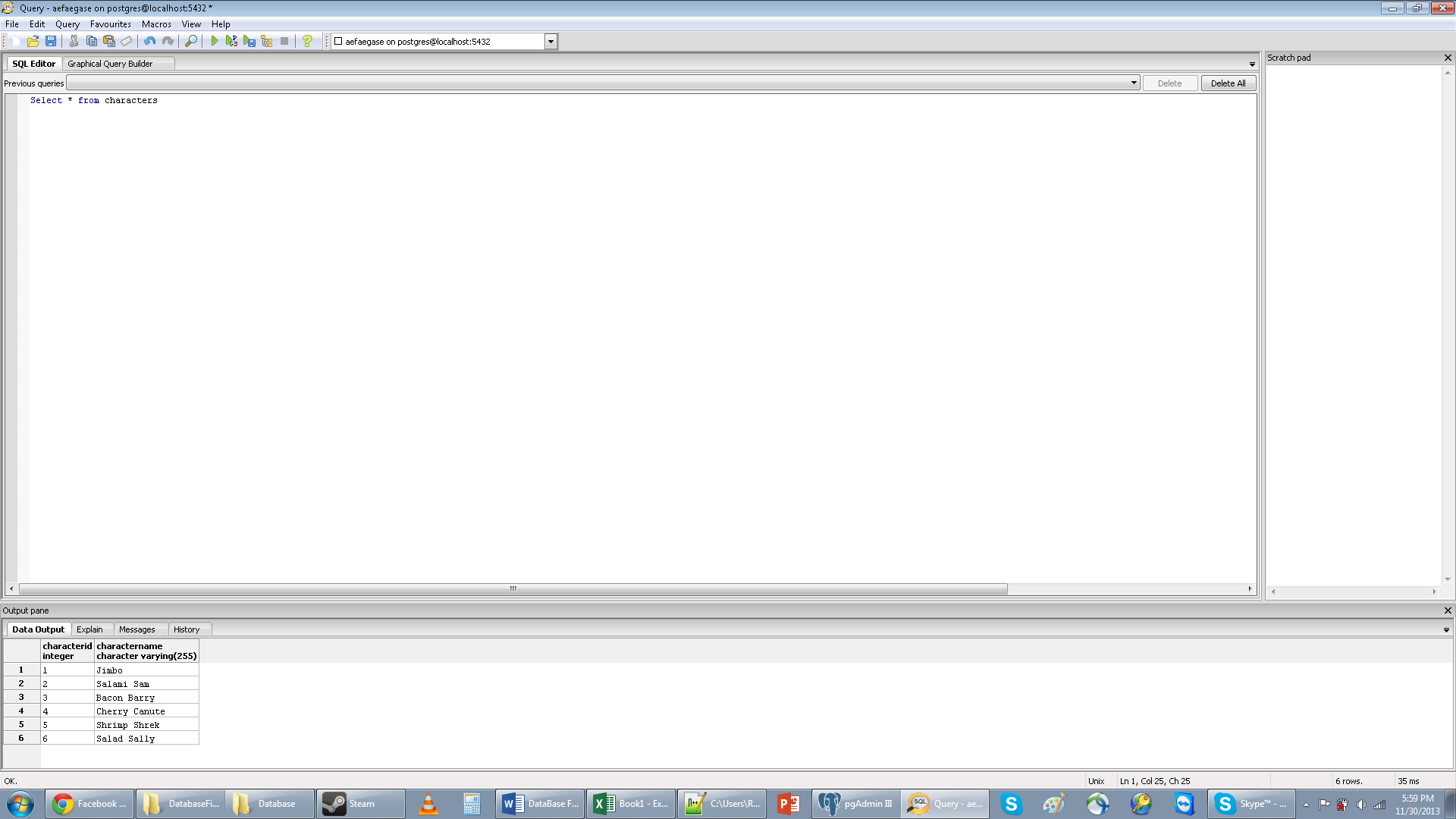
primary key (CharacterID)

);

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CharacterID🡪 CharacterName

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Classes Table

This table stores Classes and all their properties. Classes have HP, HP per level, MP, MP per level, etc.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create table Classes(

ClassID SERIAL not null,

ClassName varchar(255),

ClassInfo varchar(255),

ClassHP int,--Hitpoints

ClassHPPL int,--hitpoints per level (lost per level)

ClassMP int,

ClassMPPL int,

ClassSP int,

ClassSPPL int,

ClassPP int,

ClassPPPL int,

ClassAS int,

ClassASPL int,

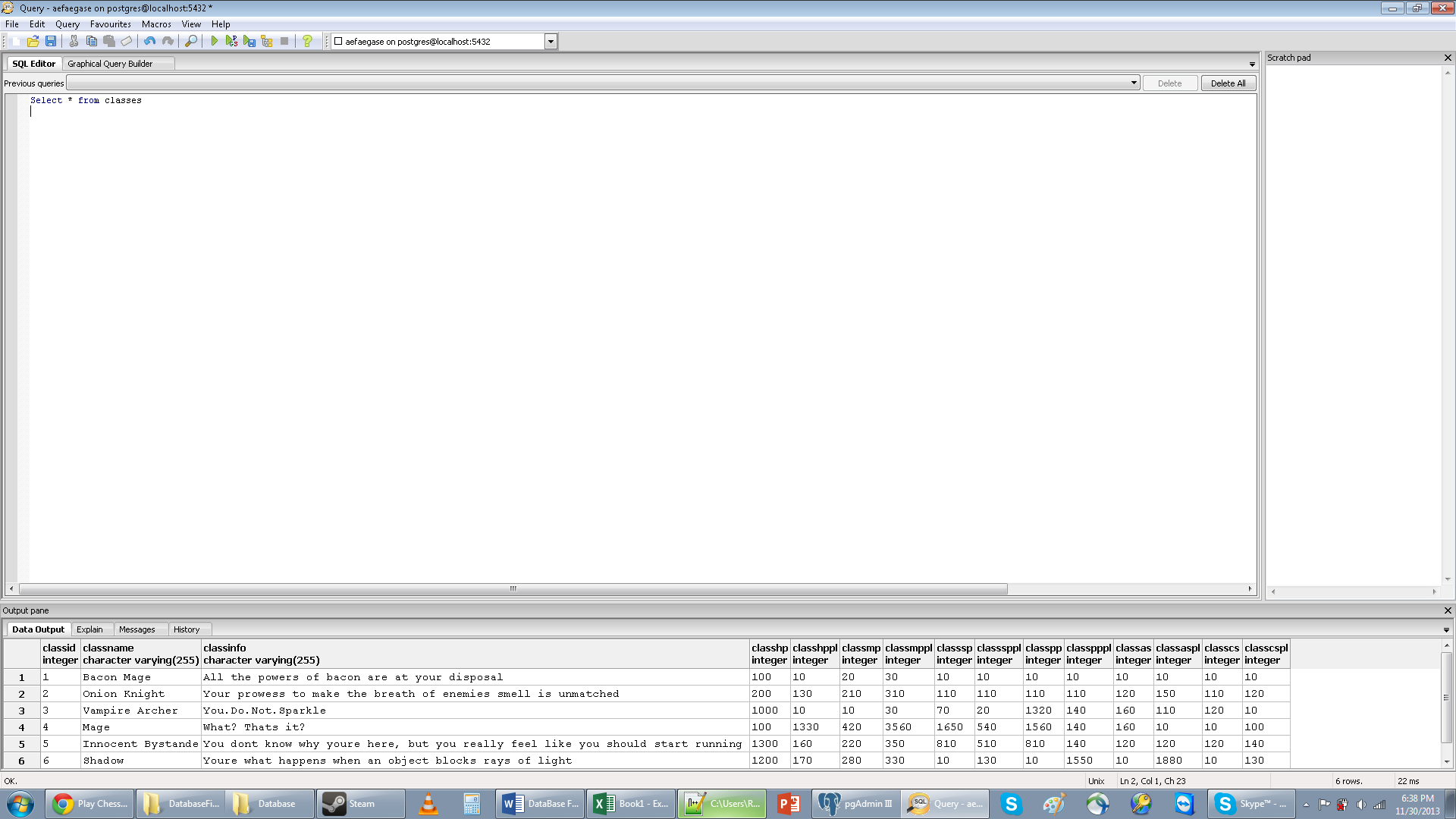
ClassCS int,

ClassCSPL int,

primary key (ClassID));

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ClassID🡪 ClassName, ClassInfo, ClassHP, ClassHPPL, ClassMP, ClassMPPL, ClassSP, ClassSPPL, . ClassPP, ClassPPPL, ClassAS, ClassASPL, ClassCS, ClassCSPL.

** Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Database: MapChunks Table

This table will hold the map chunks for use in the levels Table.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create table MapChunk(

MapChunkID SERIAL not null,

MapChunkName varchar(255),

MapChunkRating int,

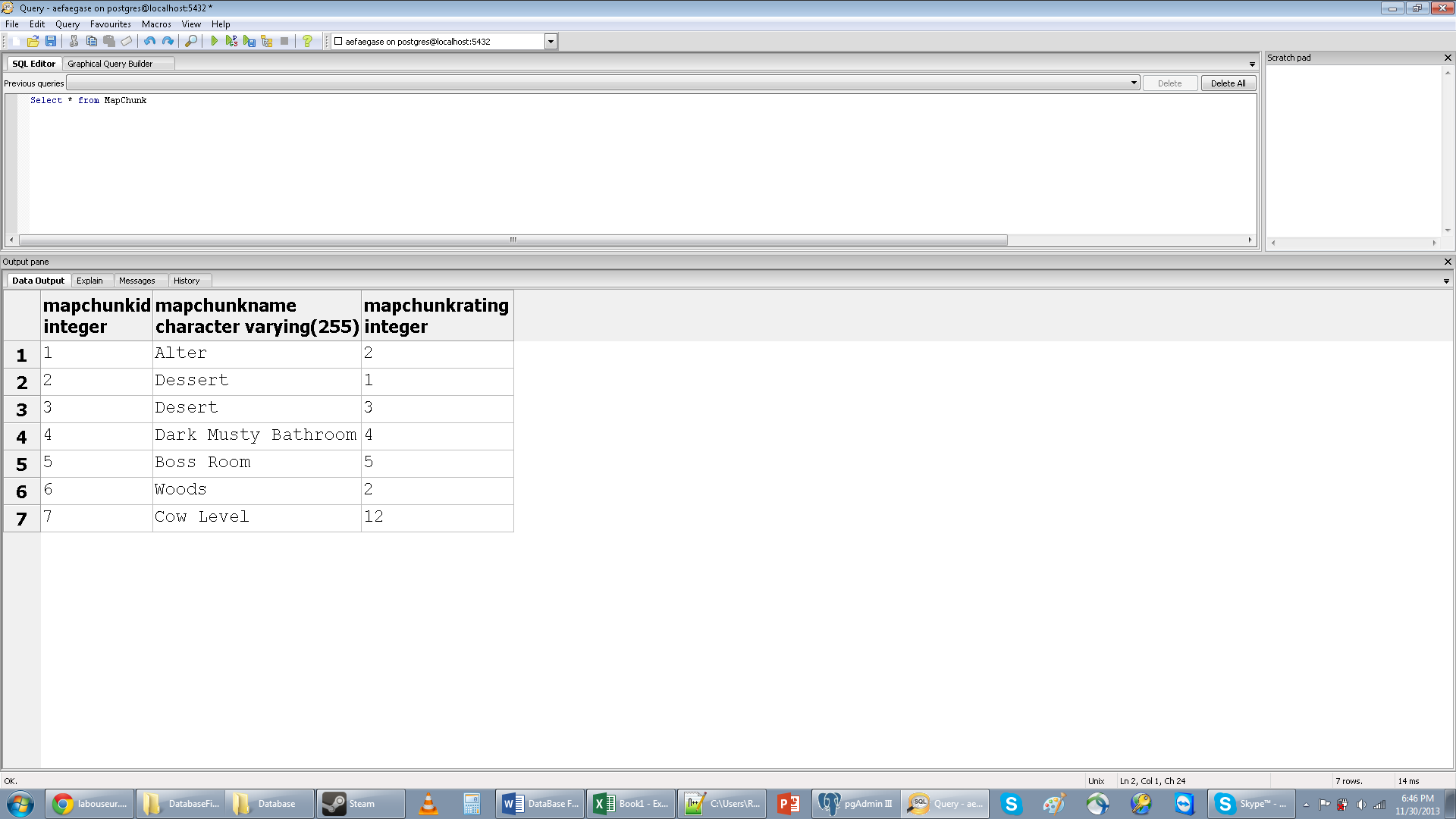
primary key(MapChunkID)

);

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MapChunkID🡪 MapChunkName, MapChunkRating

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Levels table

This table holds all the levels. Each level has a rating and will be organized as such in the actual game.

**SQL Create Statement\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

create table Levels(

LevelID Serial not null,

LevelName varchar(255),

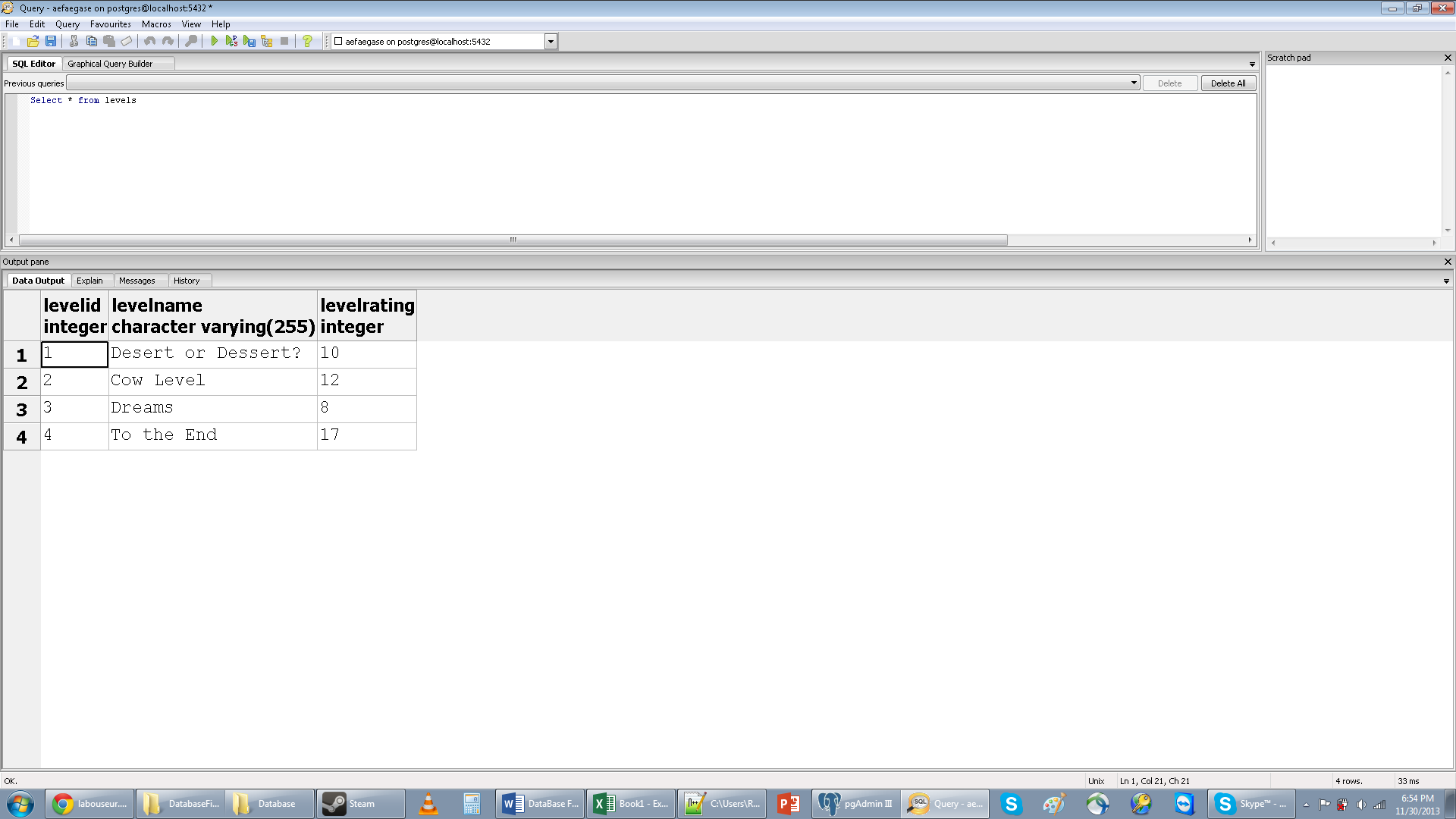
LevelRating int,

primary key(LevelID));

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LevelID🡪LevelName, LevelRating

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Games Table

This table will serve as a hub for all the game information.

**SQL Create Statement\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create table Games(

GameID Serial not null,

score int,

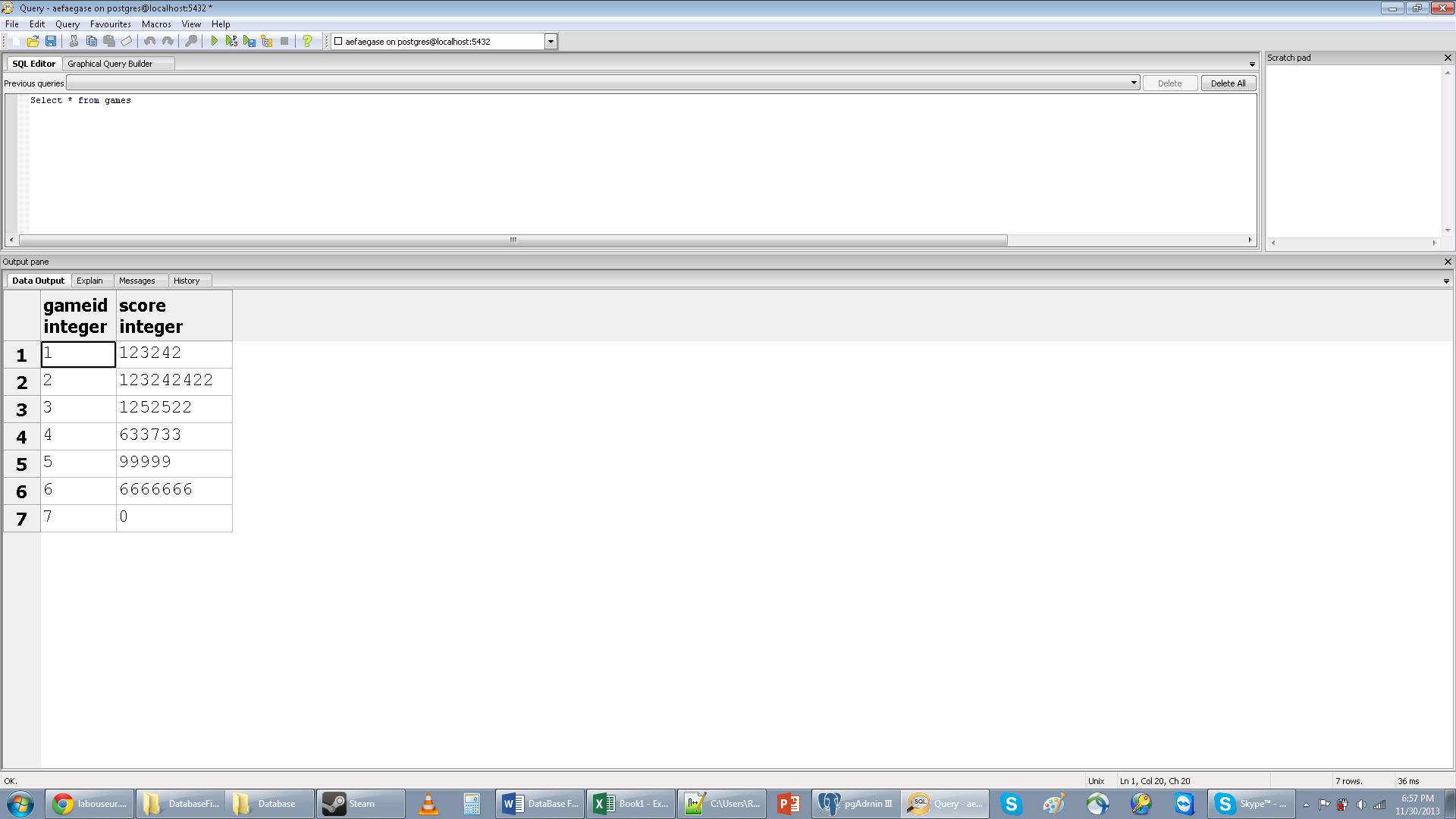
primary key(GameID)

);

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

GameID🡪 Score

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: People

This table holds all the people. People are all players, but not all players will be admins

**SQL Create Statement\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create table People(

PeopleID SERIAL not null,

Name varchar(255),

Email varchar(255),

JoinDate DATE,

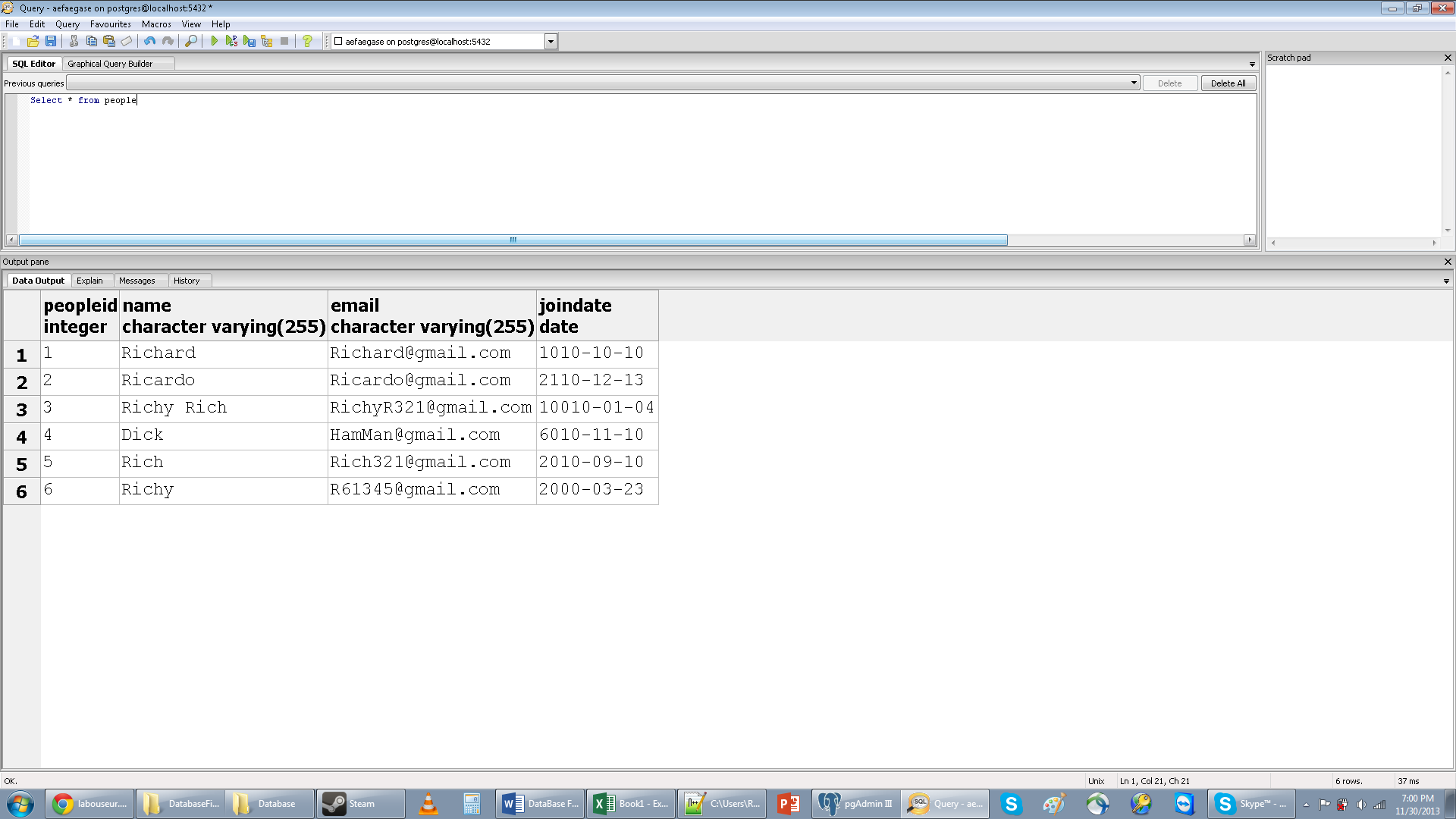
Primary Key(PeopleID)

);

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PeopleID🡪Name, Email, JoinDate

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



**Database: Admins**

The Admins table will hold the PeopleID of all the people will special permissions.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Create table Admins(

PeopleID Int,

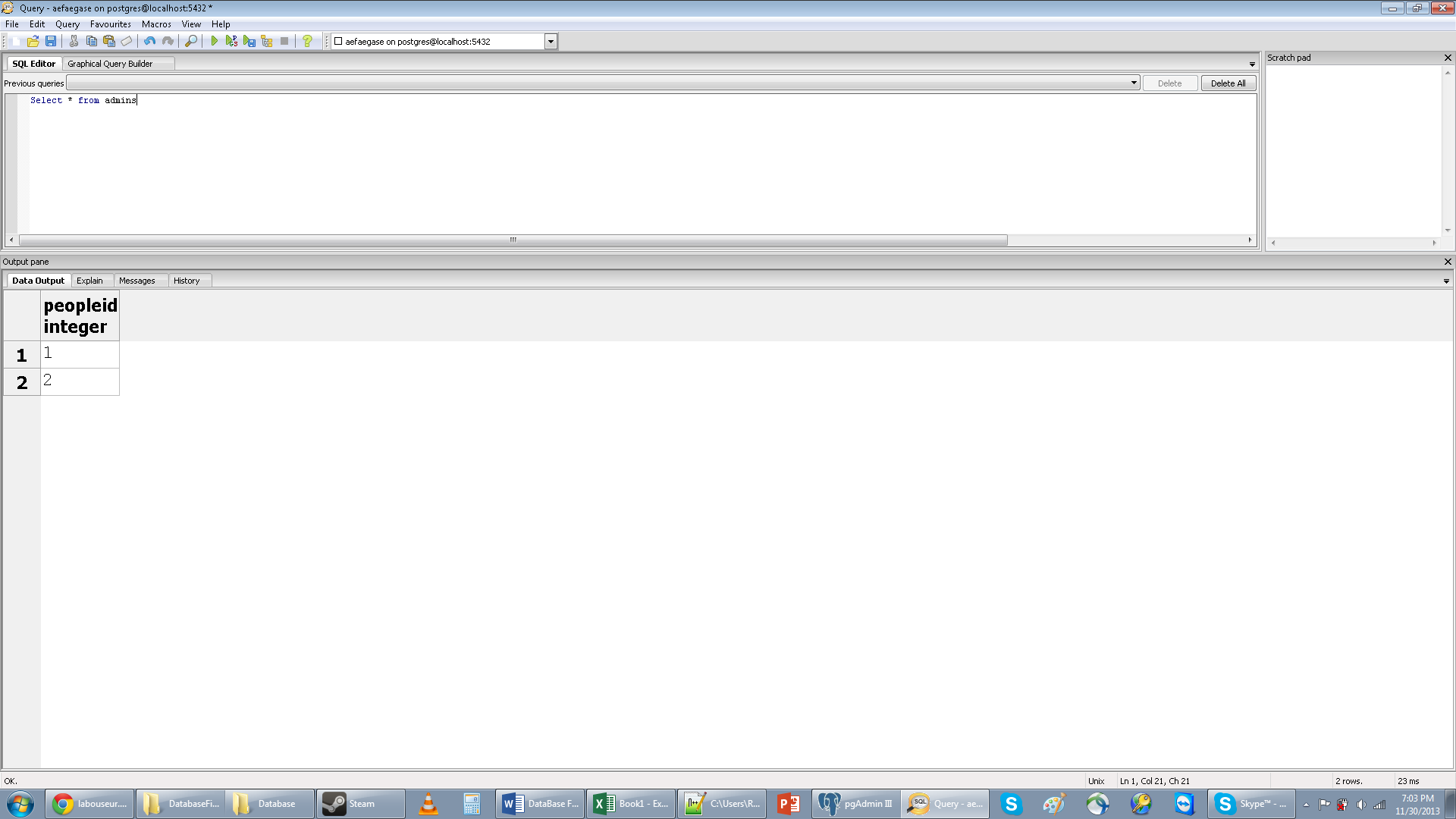
Foreign key (PeopleID) references People(PeopleID)

);

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PeopleID🡪

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Powers Table

This table contains all the powers that the player can use. Each power has damage, cool down, range, and MP cost

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Create table Powers(

PowerID SERIAL not null,

PowerName varchar(255),

PowerDmg int,

PowerCD int,

PowerRange int,

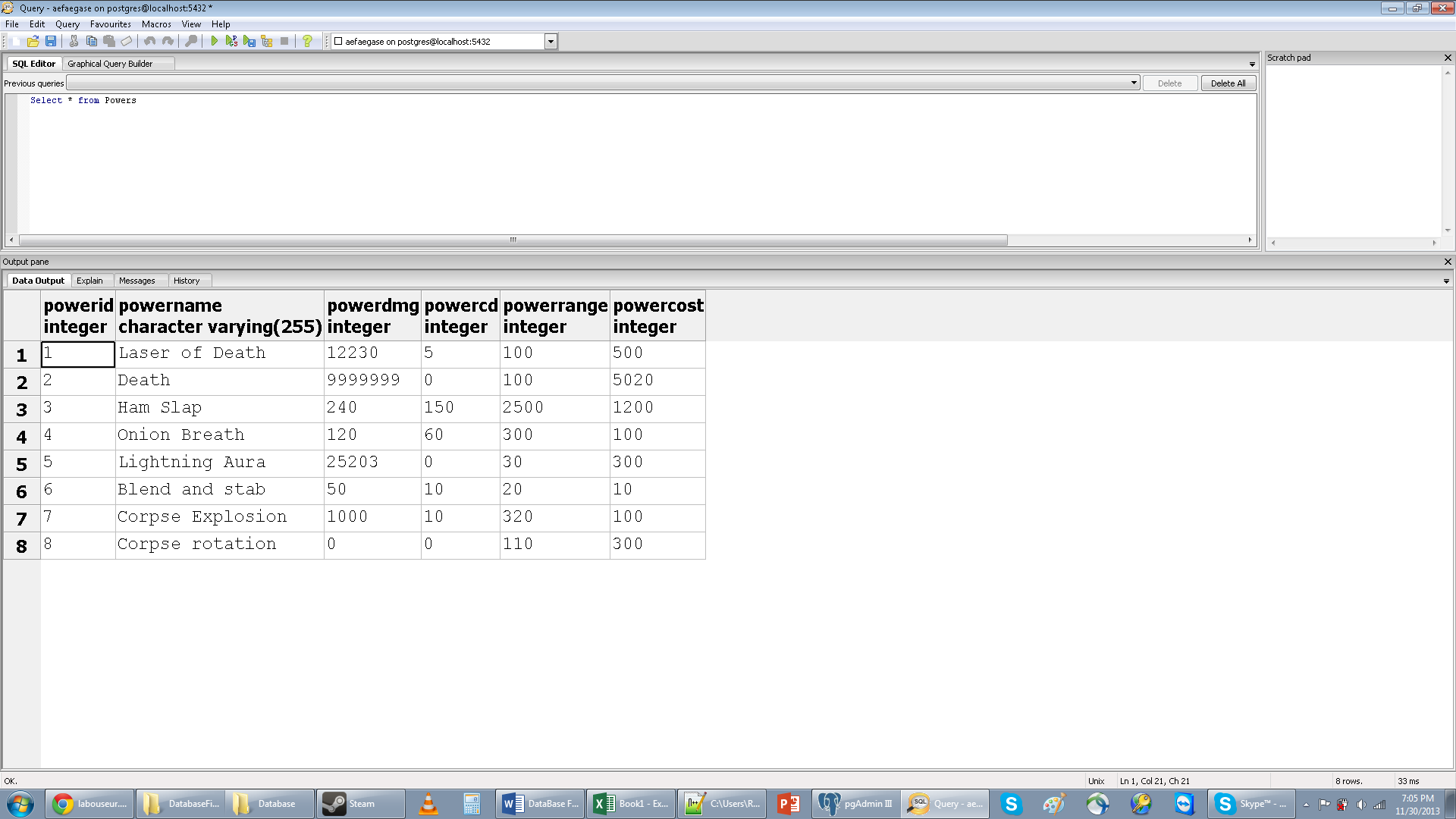
PowerCost int,

Primary key(PowerID));

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PowerID🡪 PowerName, PowerDmg, PowerCD, PowerRange, PowerCost

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Enemies Table

The Enemies table holds all the enemies that the player will fight in the game.

**SQL Create Statement\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create table Enemies(

EnemyID SERIAL not null,

EnemyName varchar(255),

EnemyInfo varchar(255),

EnemyType varchar(255),

ETypeHP int,

ETypeMP int,

ETypeSP int,

ETypePP int,

ETypeAS int,

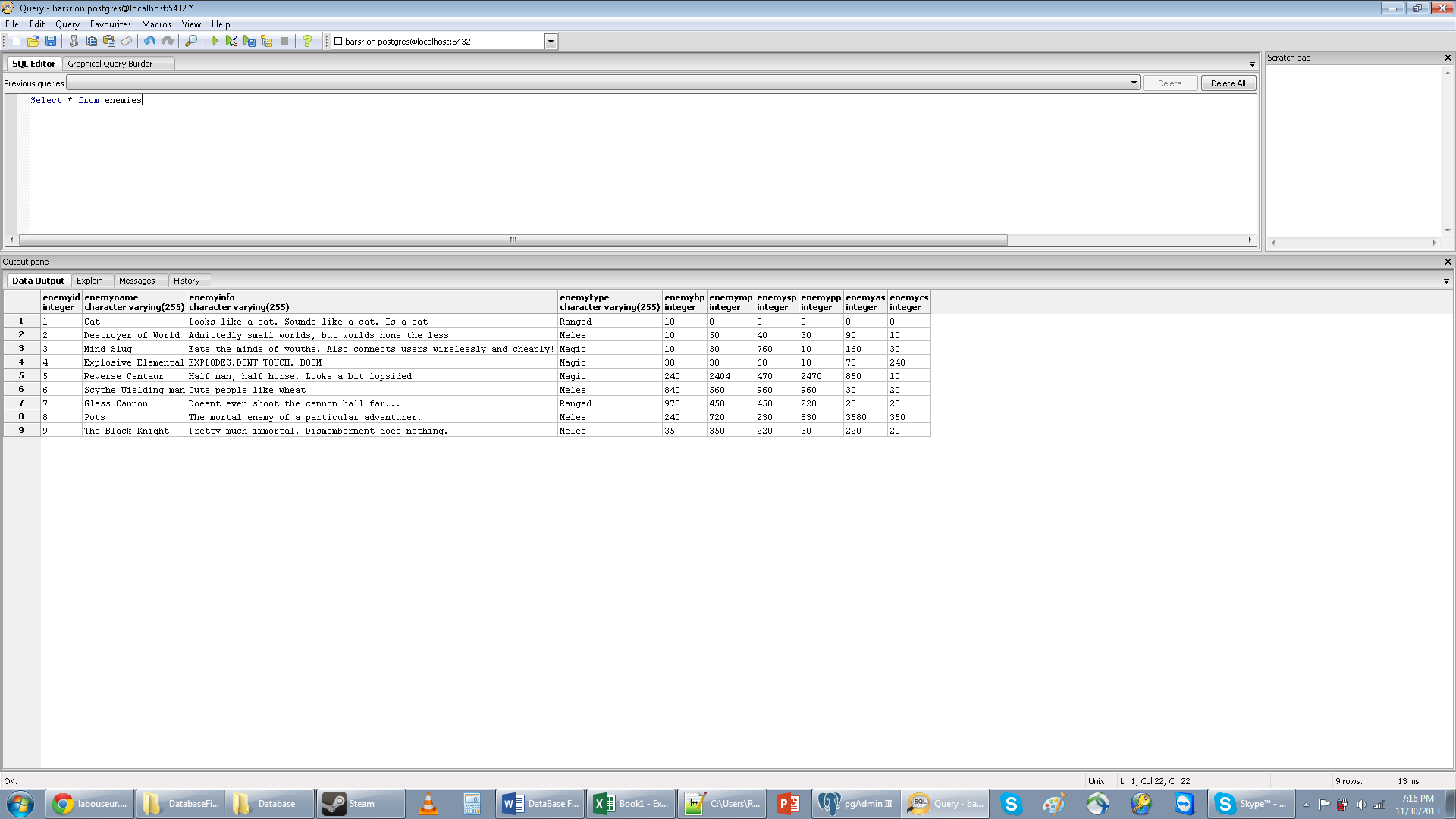
ETypeCS int,

Primary key(EnemyID)

);

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

EnemyID🡪EnemyName, EnemyInfo, EnemyType,

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Database: Enemy Skills Table

This table contains the skills that enemies can use.

**SQL Create Statement\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create table EnemySkills(

ESkillsID SERIAL not null,

ESkillsName varchar(255),

ESkillsDMG int,

ESkillsCD int,

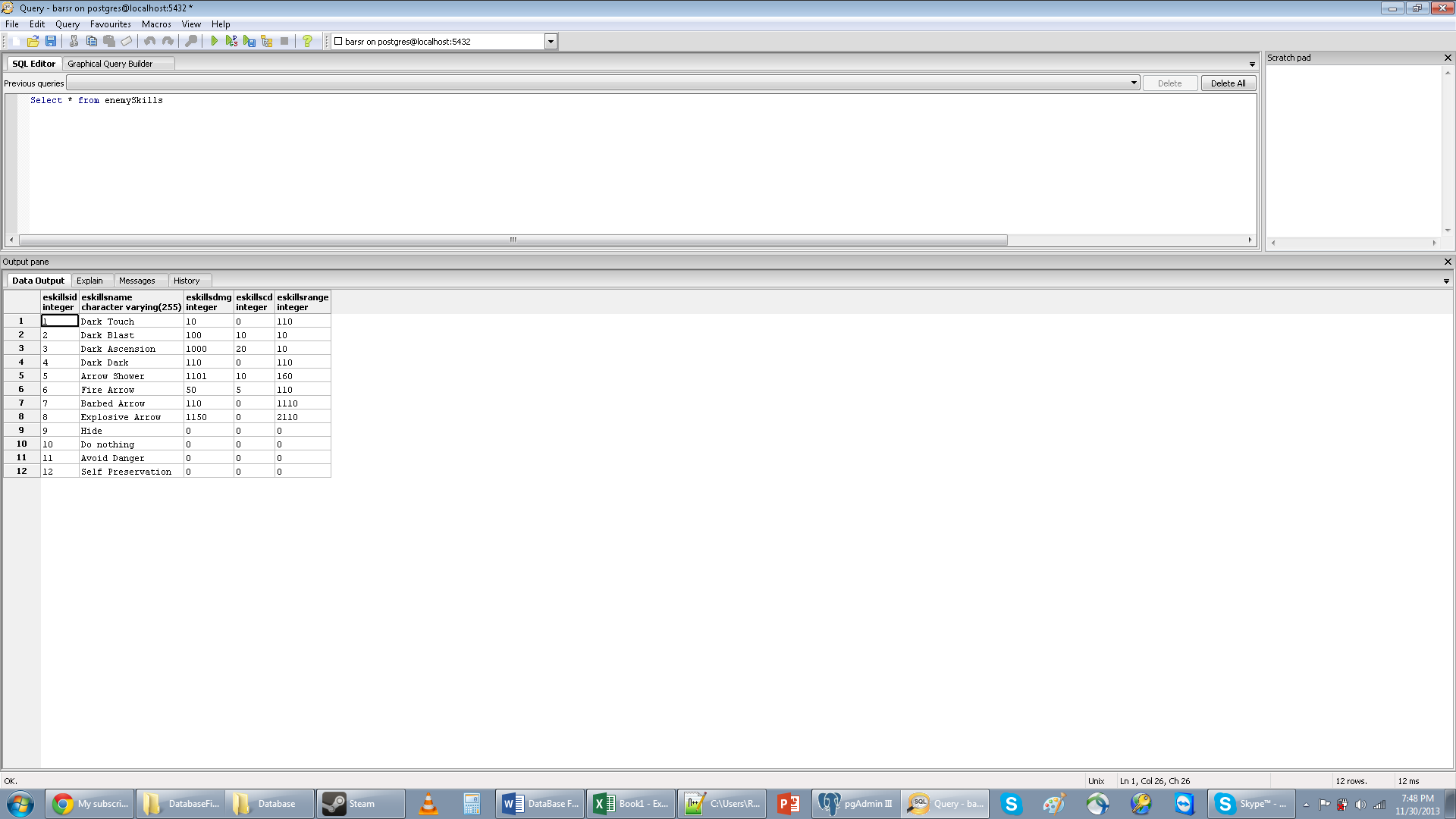
ESkillsRange int,

primary key(ESkillsID));

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ESkillsID🡪EskillsName, EskillsDMg, EskillsCD, EskillsRange

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Character\_Items Table

This table stores which characters have which items

**SQL Create Statement\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create table Character\_Items(

CharacterID int,

ItemID int,

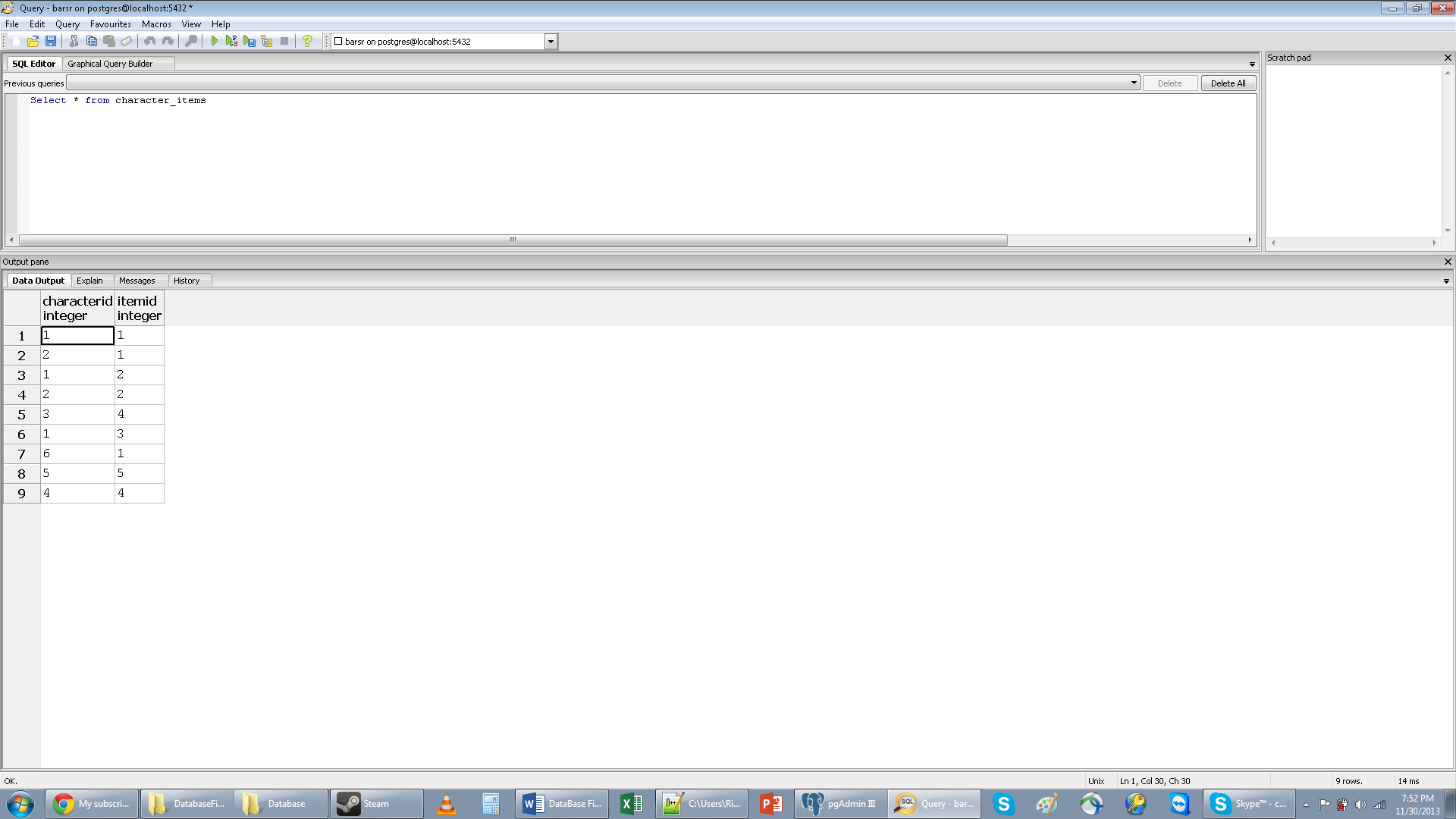
Foreign Key (CharacterID) References Characters(CharacterID),

Foreign Key(ItemID) References Items(ItemID));

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ItemID🡪

CharacterID🡪

**Sample Results\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Database: Character\_Classes Table

This table stores all the classes that each character has.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Create table Character\_classes(

CharacterID int,

ClassID int,

Foreign Key (CharacterID) References Characters(CharacterID),

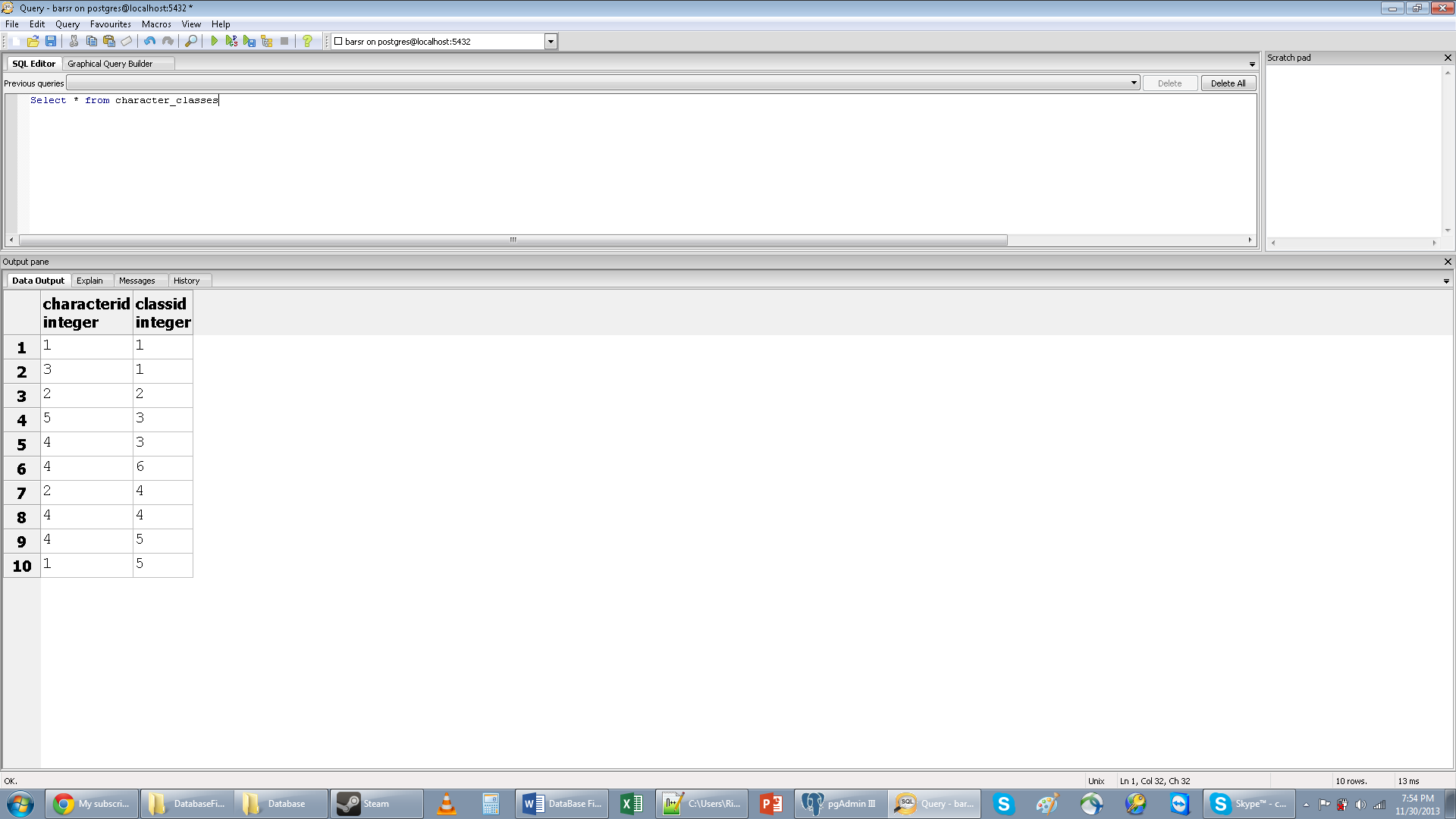
Foreign Key(ClassID) References Classes(ClassID));

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CharacterID🡪

ClassID🡪

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Character\_Games Table

This table stores all the games that characters have been in.

**SQL Create Statement\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create table Character\_Games(

CharacterID int,

GameID int unique,

Foreign Key (CharacterID) References Characters(CharacterID),

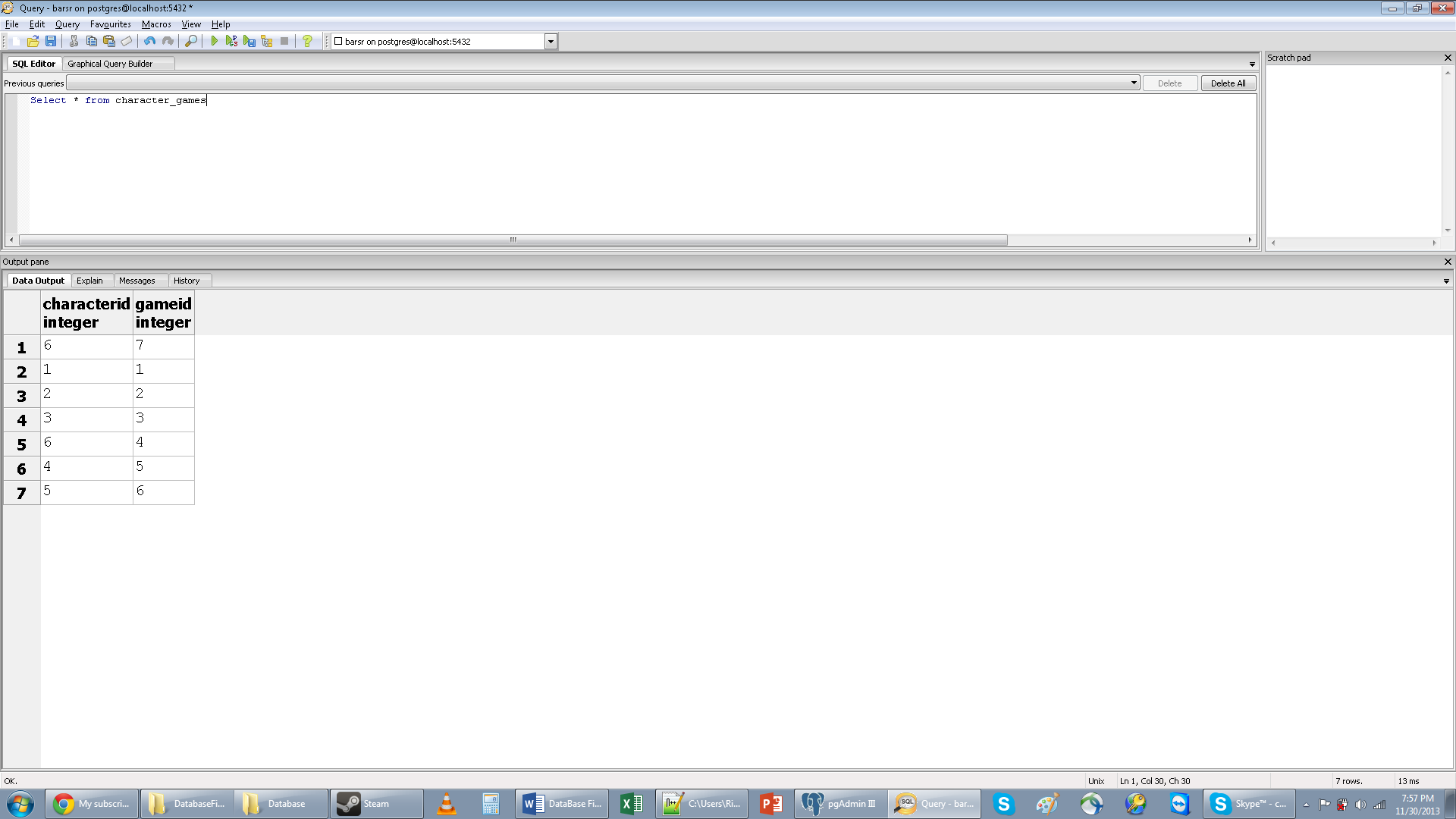
Foreign Key(GameID) References Games(GameID));

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CharacterID🡪

GameID🡪

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: People\_Games Table

The People\_Games table shows what people have been in what games.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Create table People\_Games(

GameID int unique,

PeopleID int,

Foreign Key (GameID) References Games(GameID),

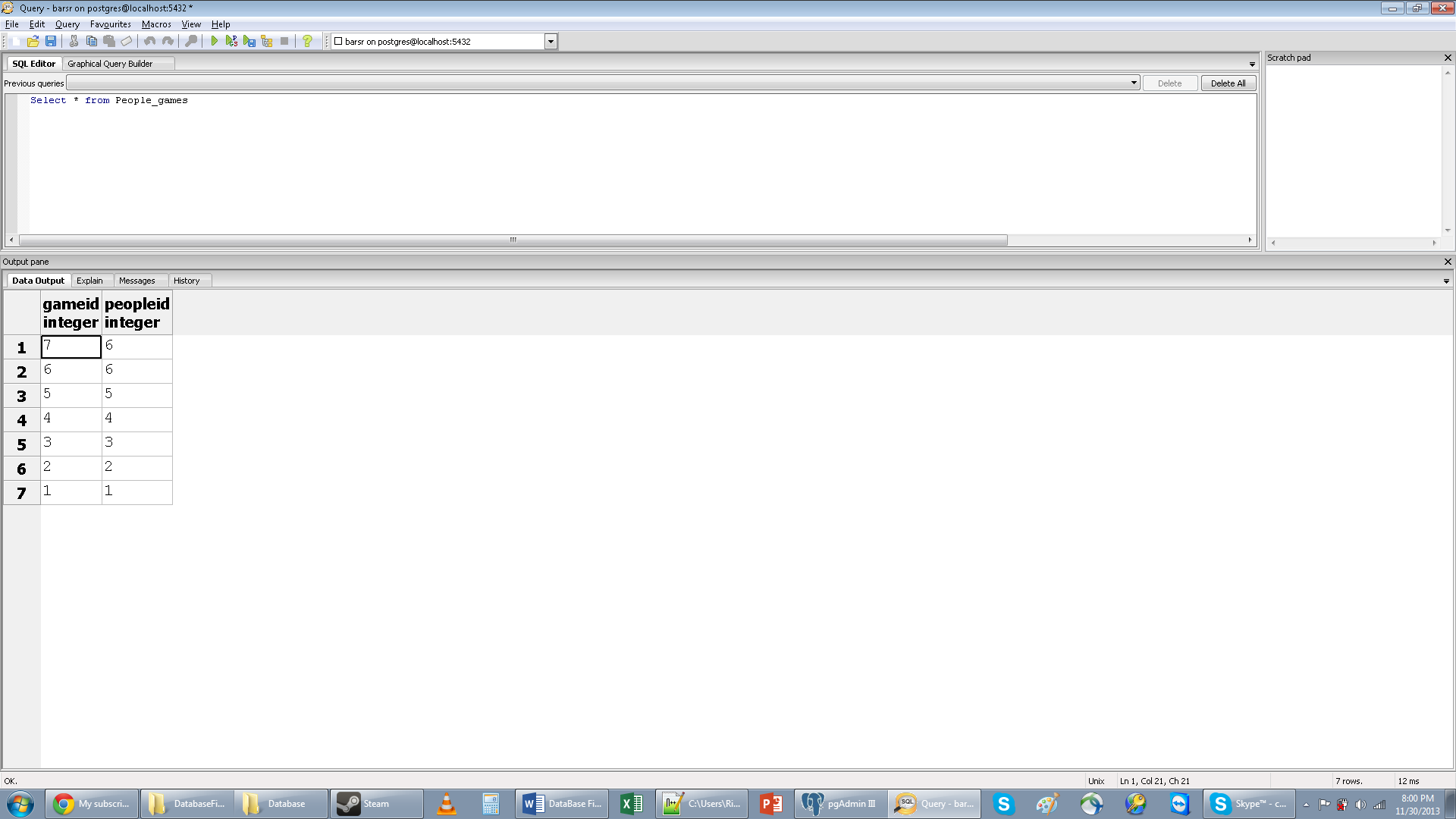
Foreign Key(PeopleID) References People(PeopleID));

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

GameID🡪

PeopleID🡪

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Games\_Levels Table

This table shows what levels are in what games.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Create table Games\_Levels(

GameID int,

LevelID int,

Foreign Key (GameID) References Games(GameID),

Foreign Key(LevelID) References Levels(LevelID)

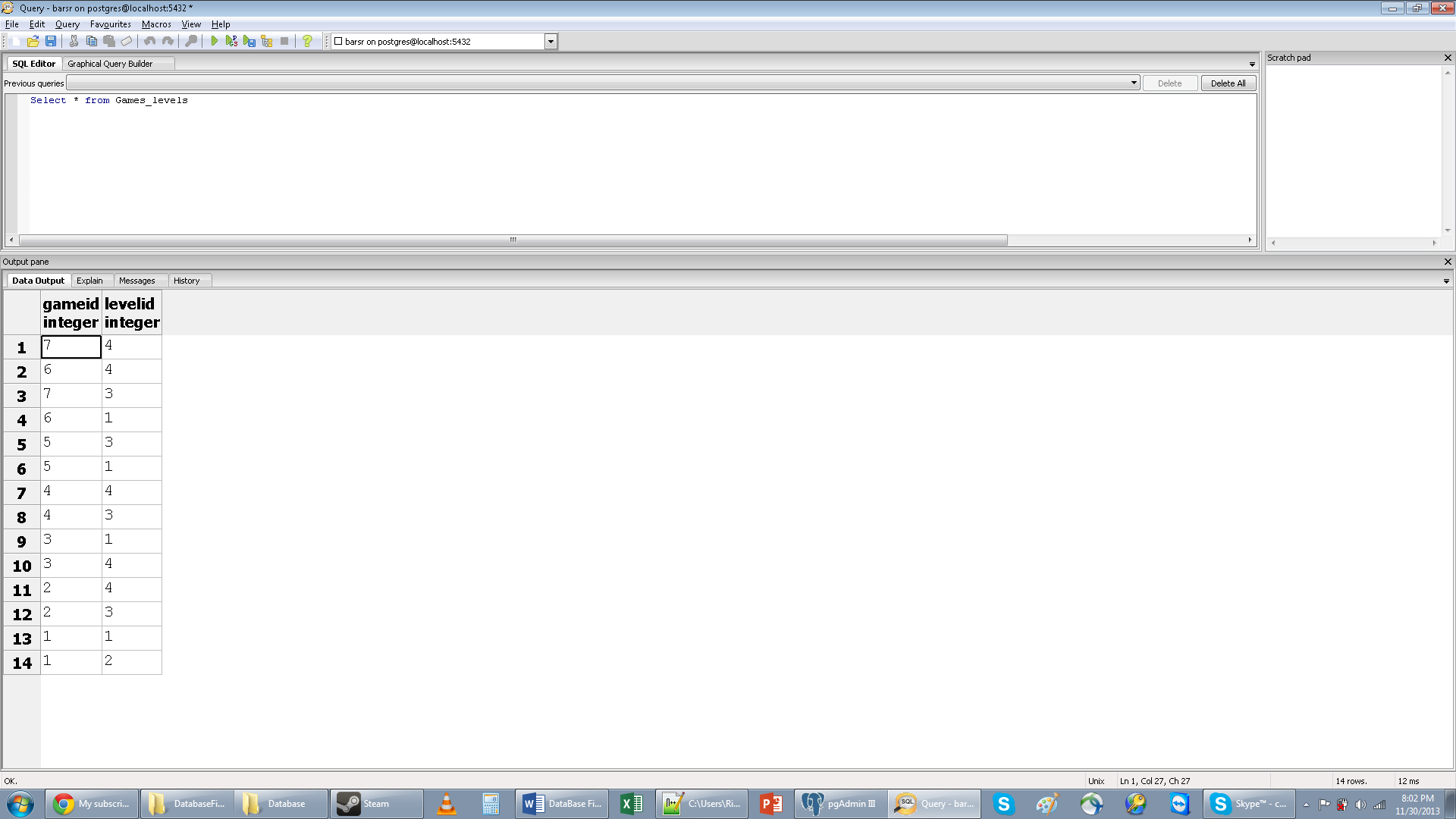
);

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

GameID🡪

LevelID🡪

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

****

Database: Classes\_Powers

This table holds all the powers that each class has.

**SQL Create Statement\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create table Classes\_Powers(

ClassID int,

PowerID int,

Foreign Key (ClassID) References Classes(ClassID),

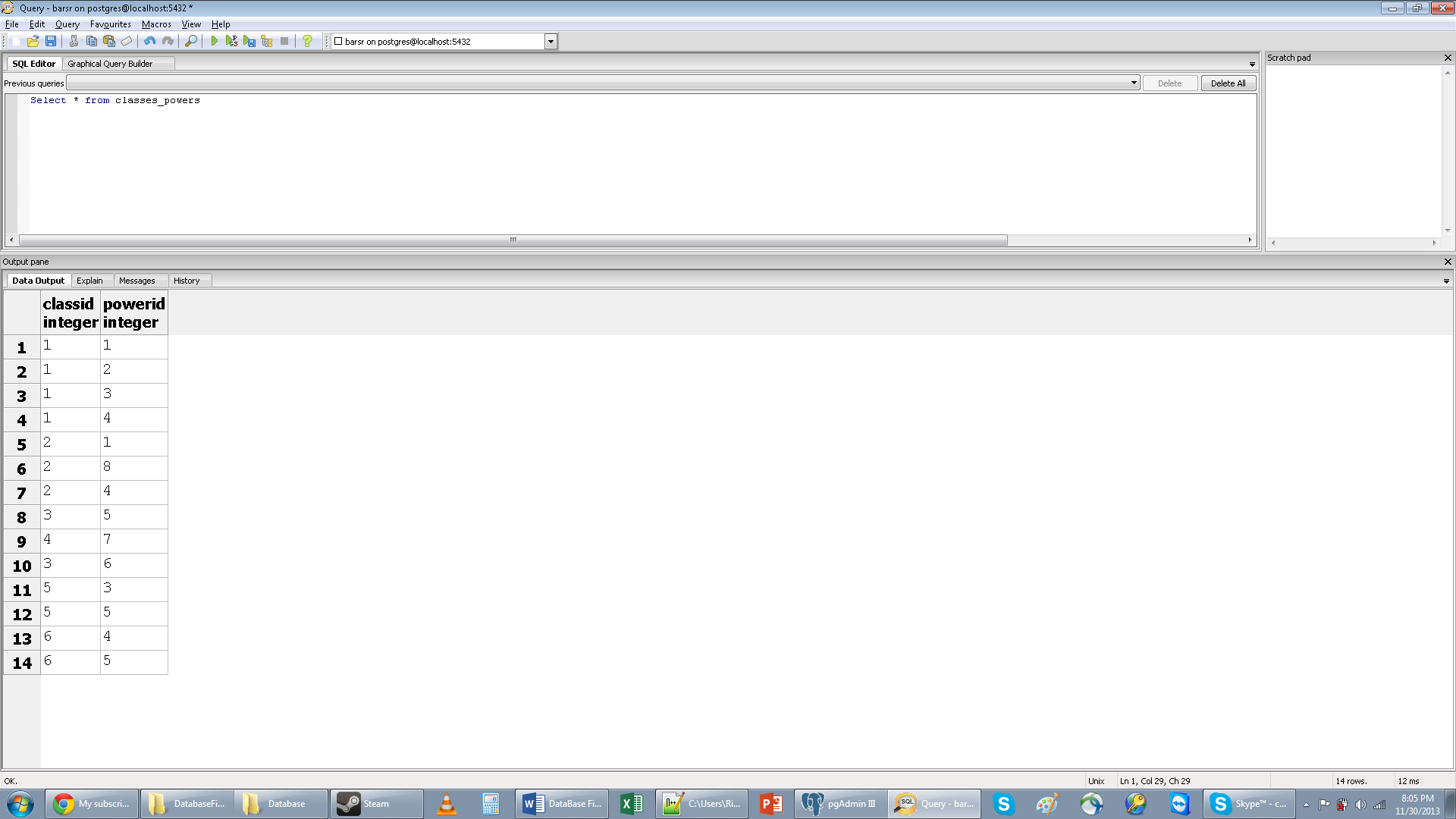
Foreign Key(PowerID) References Powers(PowerID));

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ClassId🡪

PowerID🡪

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Enemies\_Levels Table

This table contains all the levels that enemies will appear on

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Create table Enemies\_Levels(

EnemyID int,

LevelID int,

Foreign Key (EnemyID) References Enemies(EnemyID),

Foreign Key(LevelID) References Levels(LevelID)

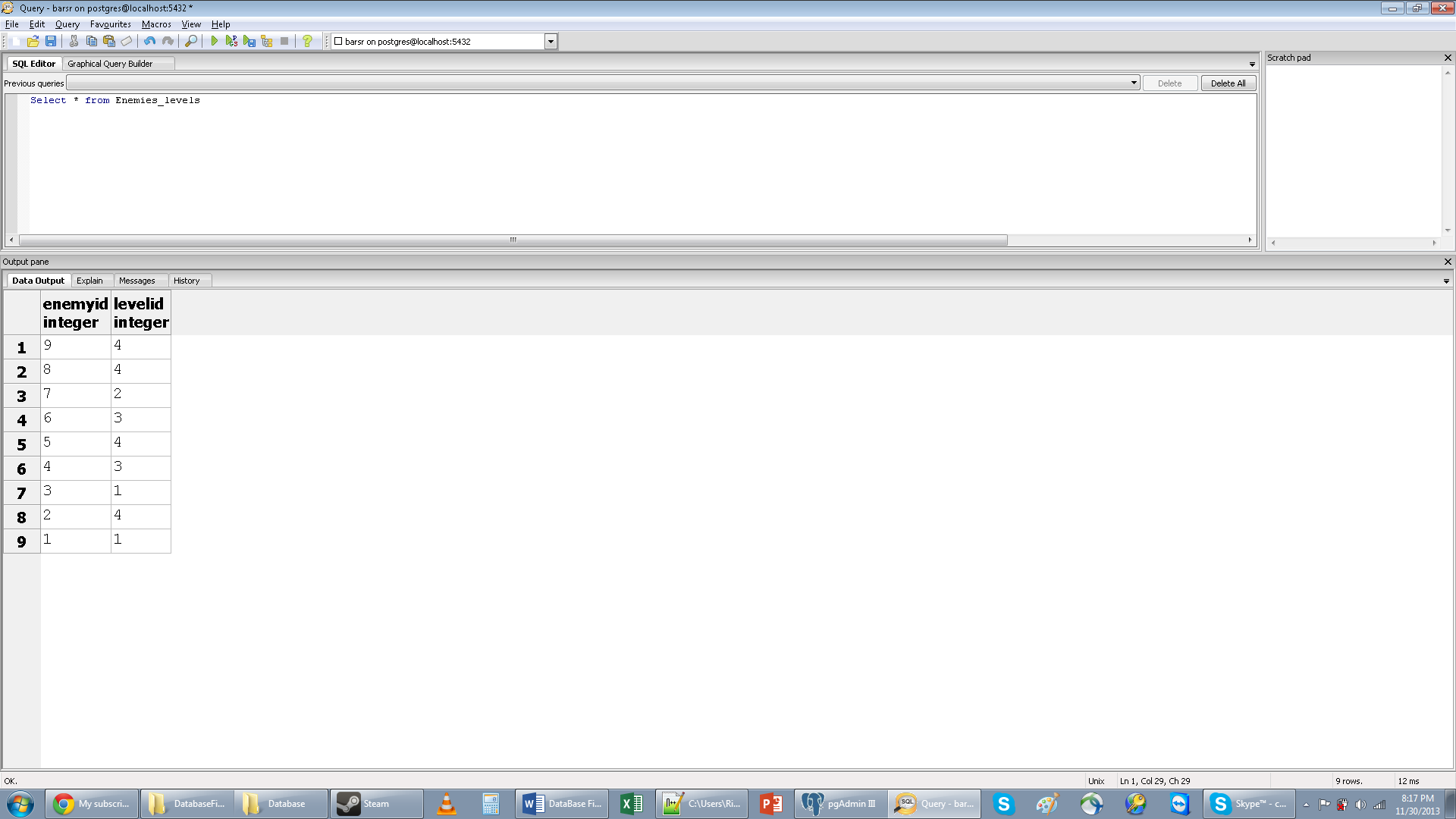
);

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

EnemyID🡪

Level🡪

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Kills\_Game Table

This table is used to show how many kills, what enemies, and which game they occurred in.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Create table Kills\_Game(

GameID int,

EnemyID int,

Foreign Key (GameID) References Games(GameID),

Foreign Key(EnemyID) References Enemies(EnemyID)

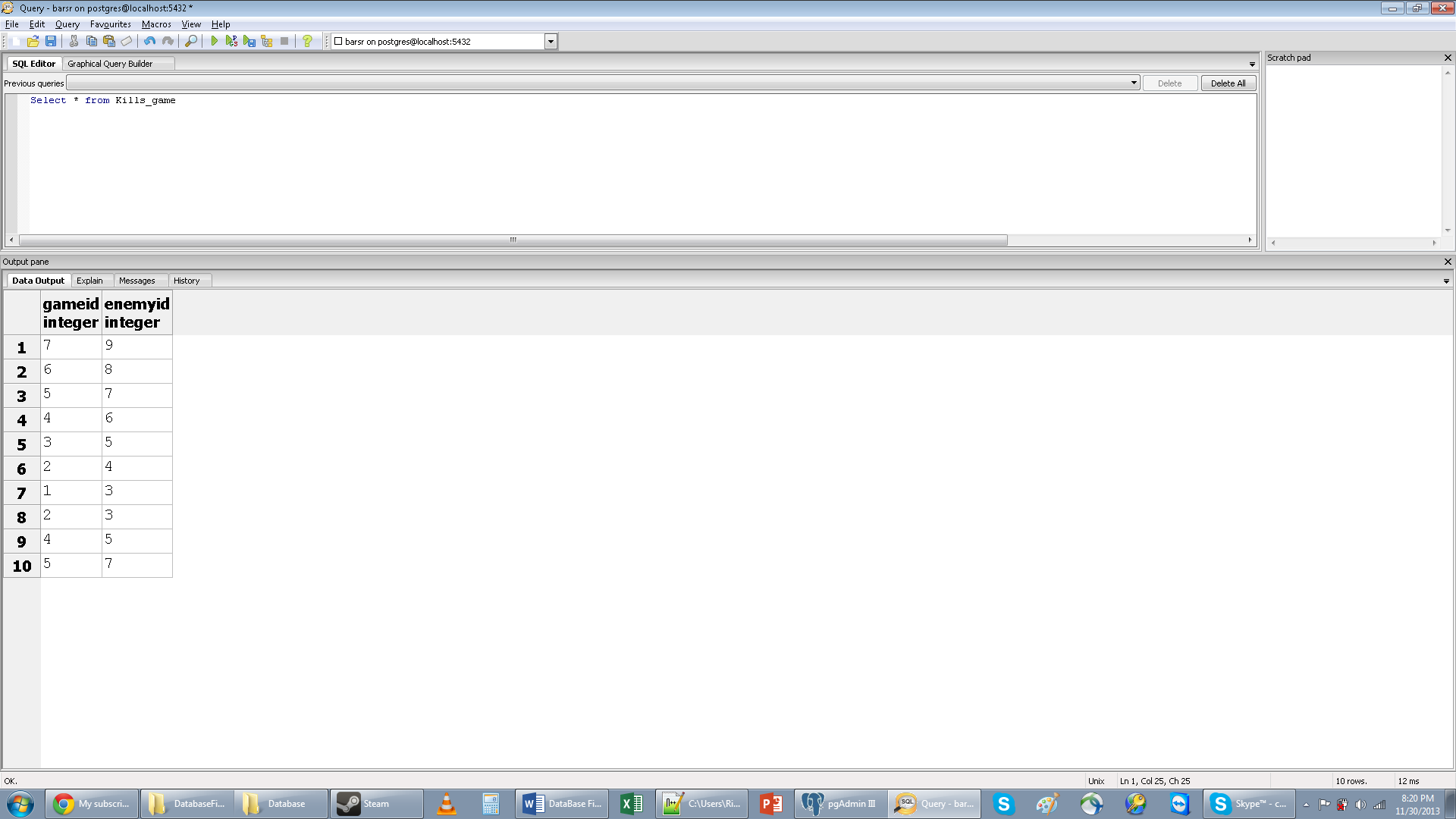
);

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

GameID🡪

EnemyID🡪

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Database: Levels\_Map\_ChunksTable

This table is used to show levels and the map chunks that make them up.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Create table Levels\_Map\_Chunks(

LevelID int,

MapChunkID int,

Foreign Key(LevelID) References Levels(LevelID),

Foreign Key(MapchunkID) References MapChunk(MapChunkID)

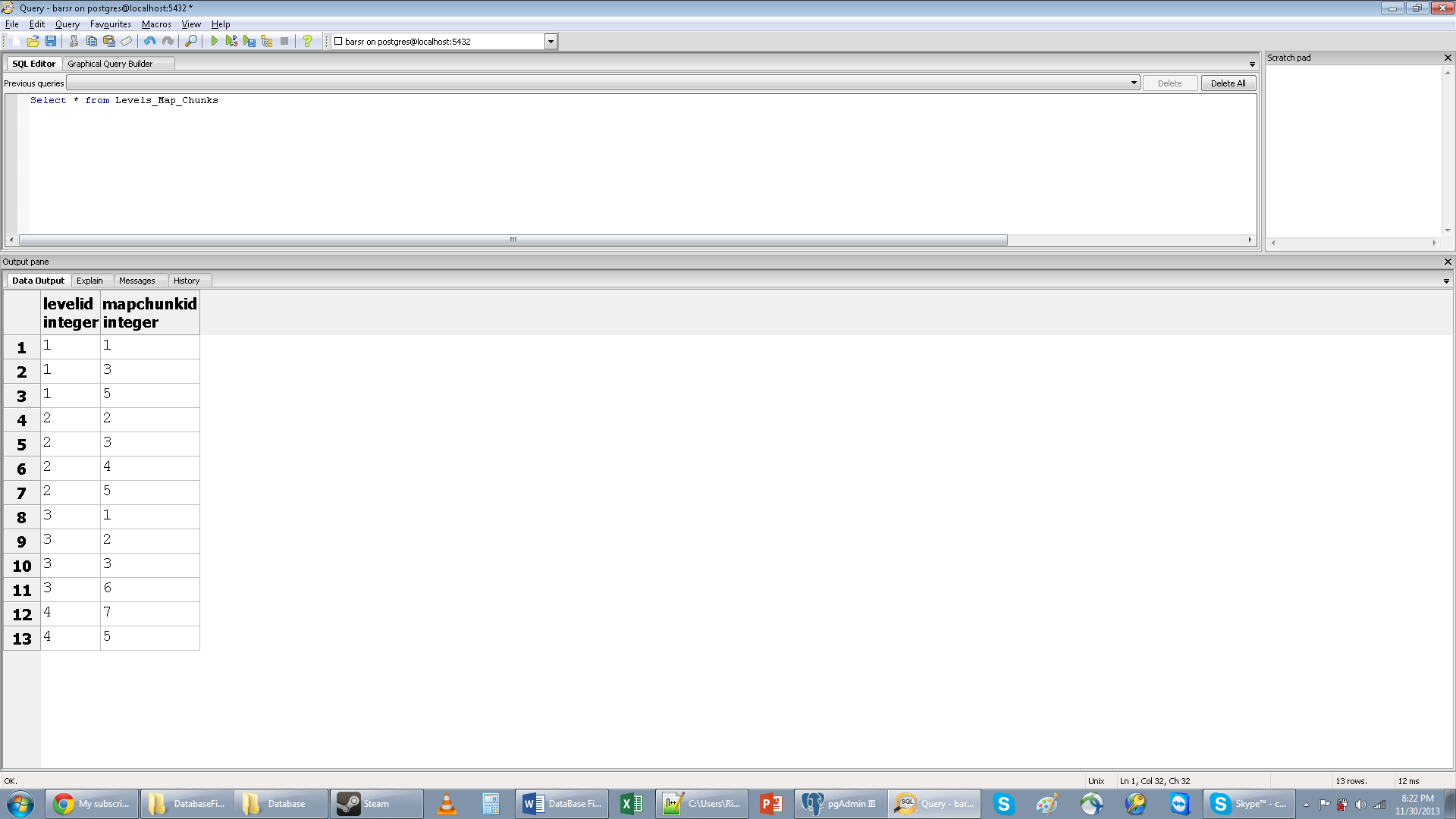
);

**Functional Dependency**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LevelID🡪

MapChunkID🡪

**Sample Data\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Views: Kill count and scores per game

This views help the user access data that is normally not easily used.

**SQL Create Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Drop View if exists KillsGame;

Create View KillsGame as( Select kg.GameID,g.score,Count(EnemyID) as KGcount

From Kills\_game kg, games g

where kg.GameID=g.gameID

Group by kg.GameID,g.score);

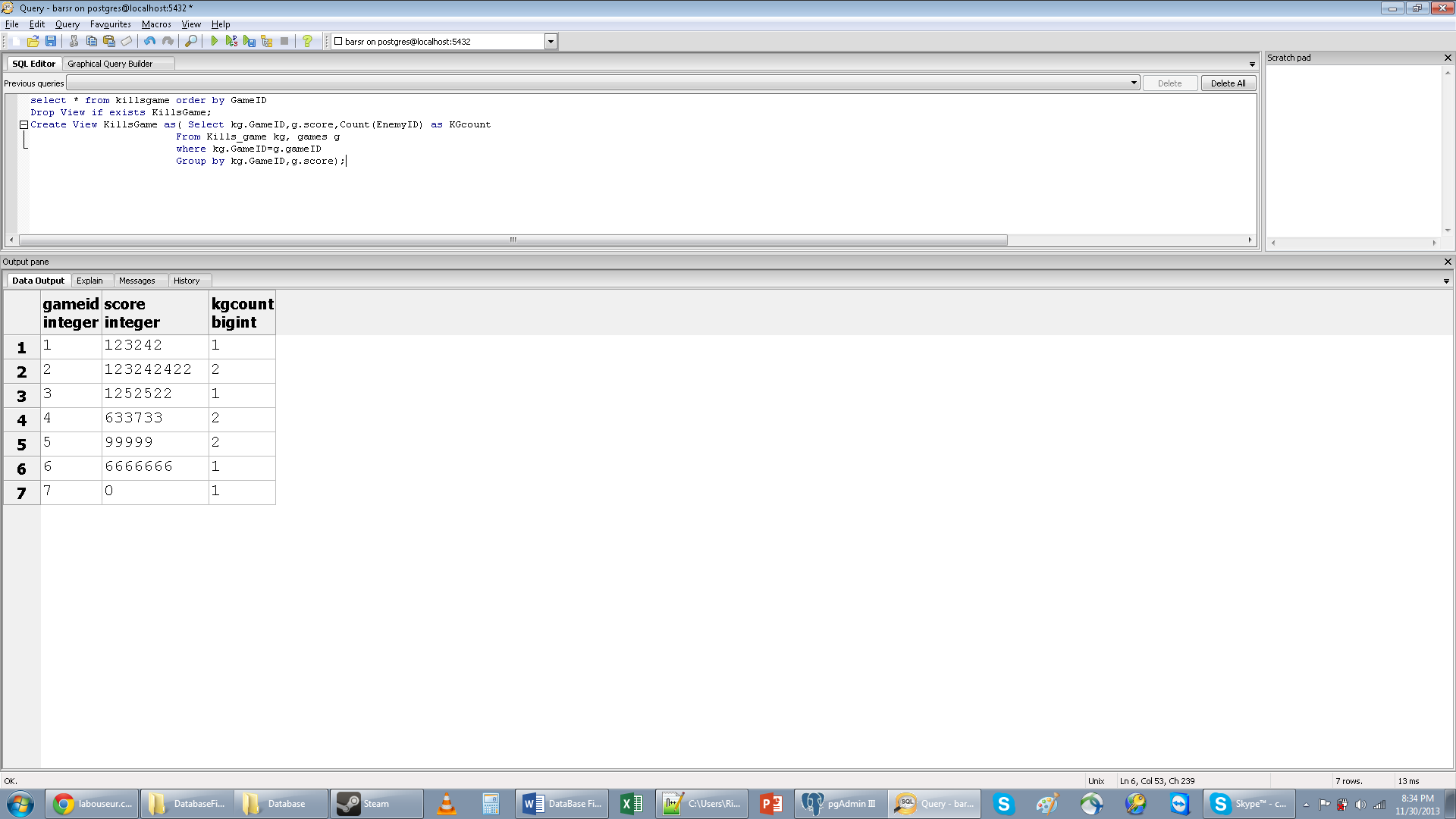
**SQL Query\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

select \*

from killsgame

Order By GameID desc

**Sample Results\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Views: Character info

This View lets admins see each character’s classes.

**SQL Create Statement\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Drop View if exists CharClass;

Create View CharClass as( SELECT distinct c.CharacterID,CharacterName, ClassName

From Classes CL, Characters C, Character\_Classes CC

Where C.CharacterID=CC.CharacterID

and CL.ClassID=CC.ClassID

Order By CharacterName Desc );

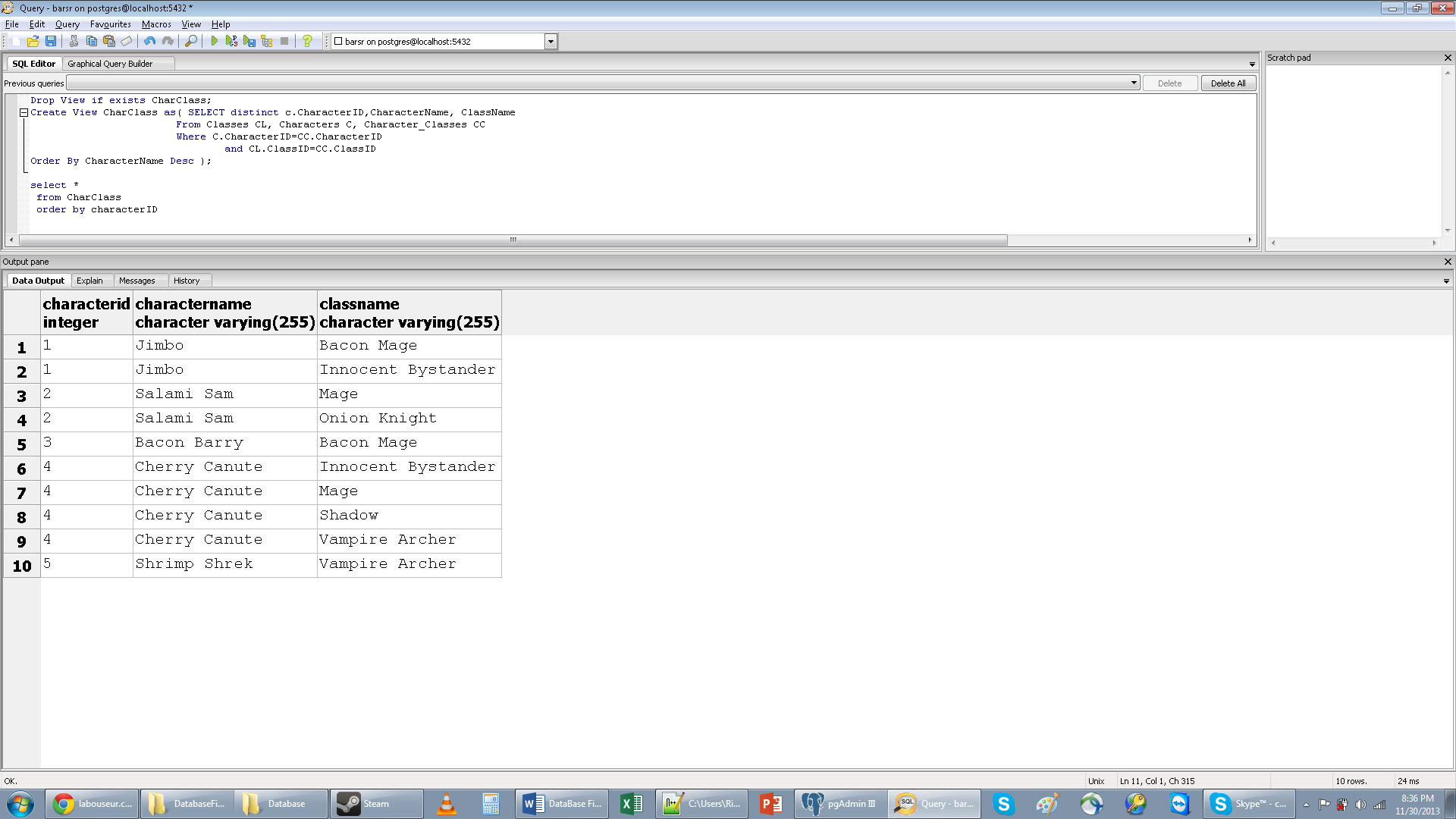
**SQL Query\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Select \*

From CharClass

Order by characterID asc

**Sample Results\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Views: Class info

This View lets admins see each character’s classes.

**SQL Create Statement\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Drop View if Exists CharItem;

Create View CharItem as(

Select C.CharacterID ,CharacterName, ItemName

From Items I, Characters c, Character\_Items CI

WHERE I.ItemID=CI.ItemID

and c.CharacterID=CI.CharacterID

Order By Charactername desc );

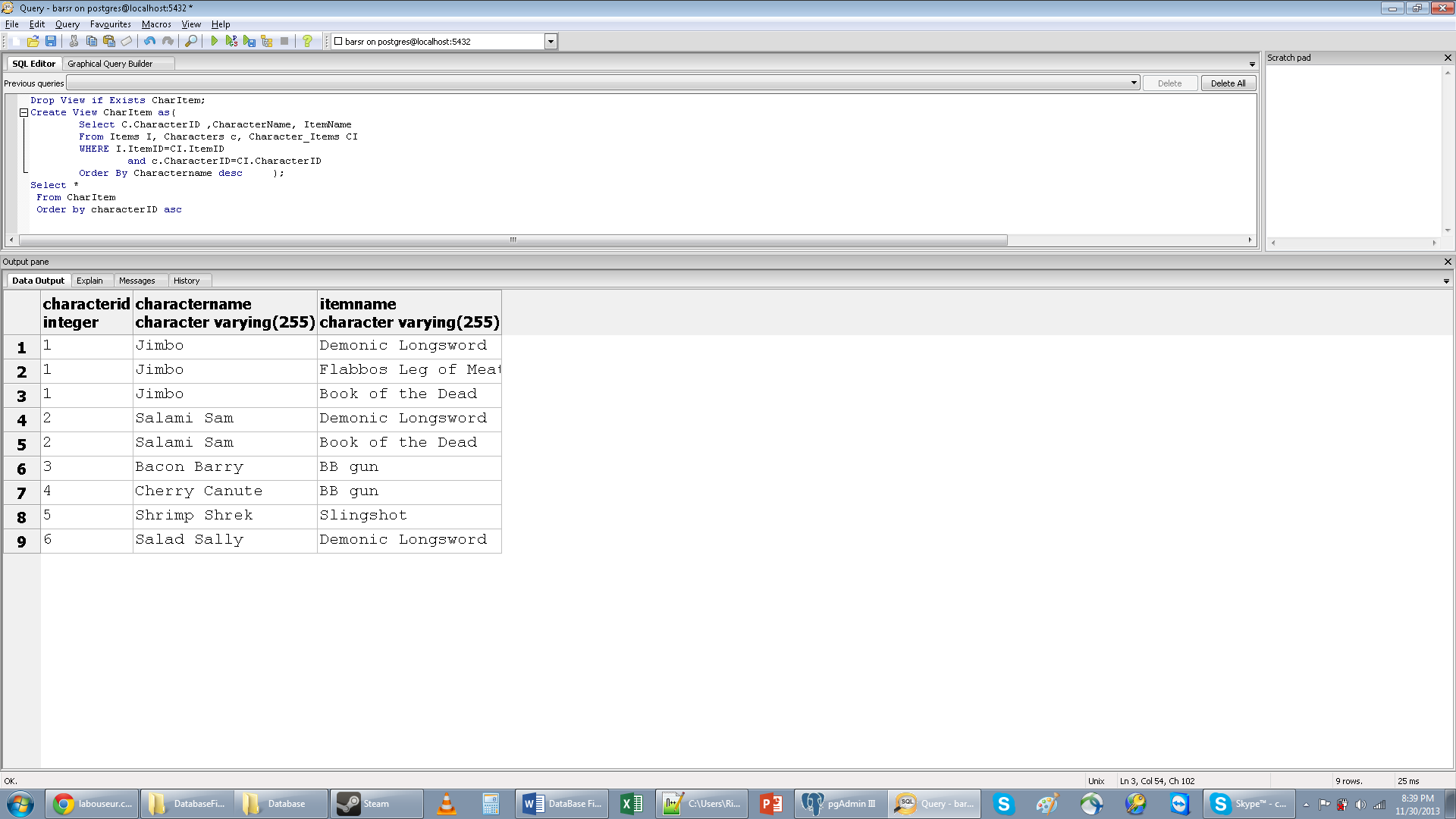
**SQL Query**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Select \*

From CharItem

Order by characterID asc

**Sample Results\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



Reports: Highscore board

This report illustrates how the highscore board will be populated.

**SQL Query\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Select P.Name, g.GameID, KillsGame.KgCount, score

From People p, Games g, People\_Games PG, KillsGame

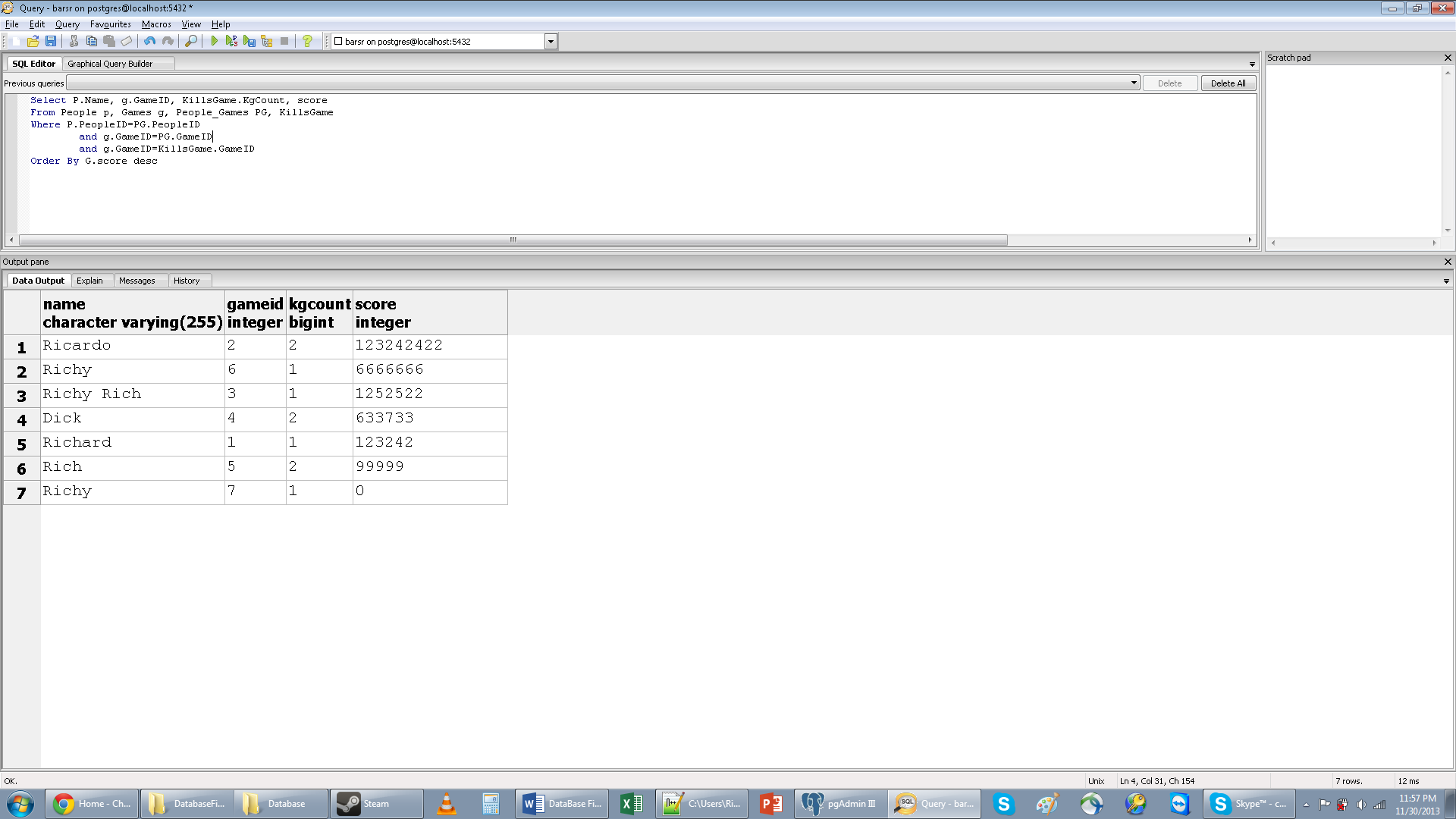
Where P.PeopleID=PG.PeopleID

and g.GameID=PG.GameID

and g.GameID=KillsGame.GameID

Order By G.GameID desc

**Sample Results**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Reports: Character Classes and Items

This report uses two previously create views to show all of the character’s classes and items.

**SQL Query\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

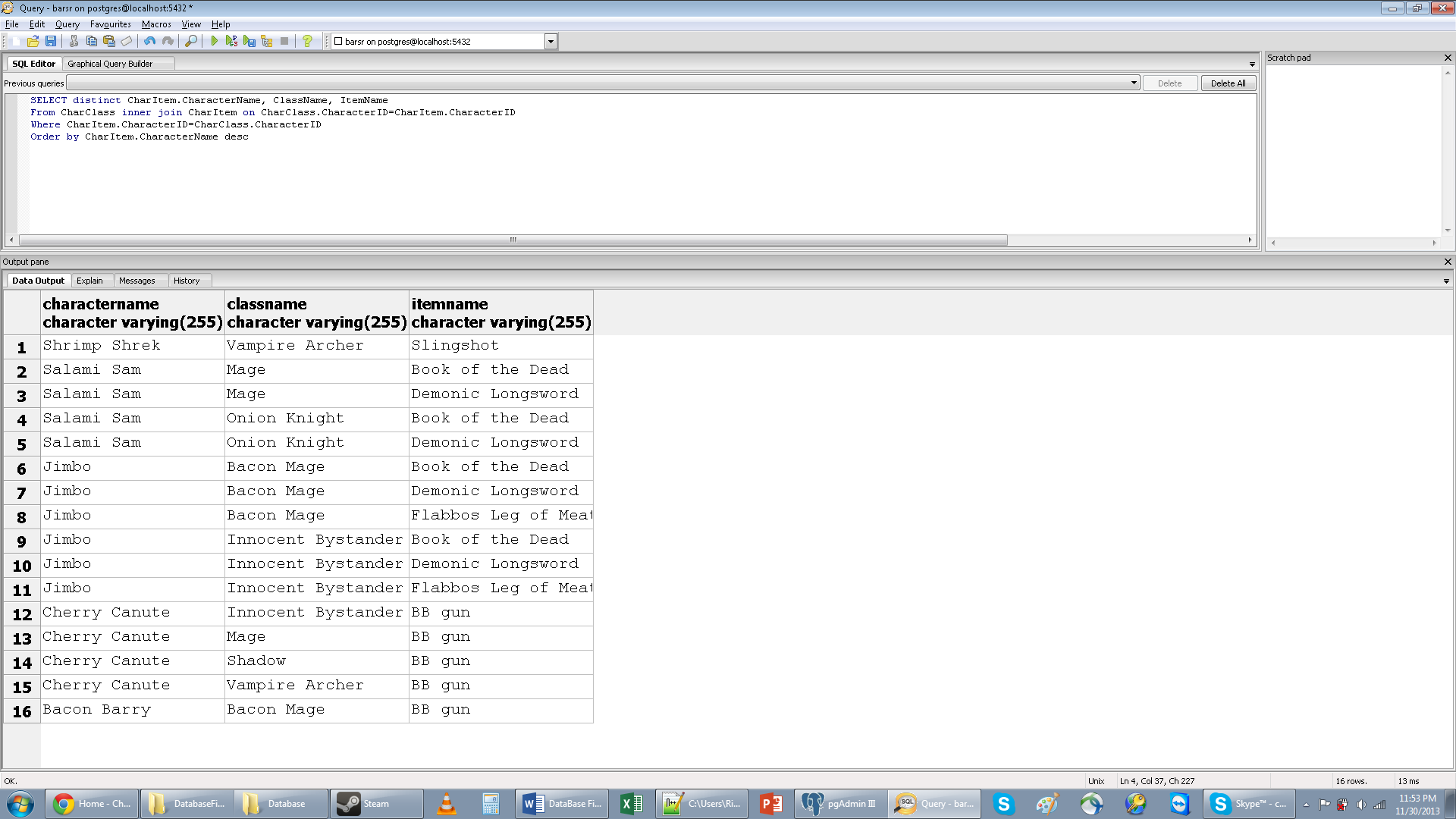
SELECT distinct CharItem.CharacterName, ClassName, ItemName

From CharClass inner join CharItem on CharClass.CharacterID=CharItem.CharacterID

Where CharItem.CharacterID=CharClass.CharacterID

Order by CharItem.CharacterName desc

**Sample Results**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Reports: Admins

This report illustrates how the highscore board will be populated.

**SQL Query\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Select \*

From People

Where PeopleID in(

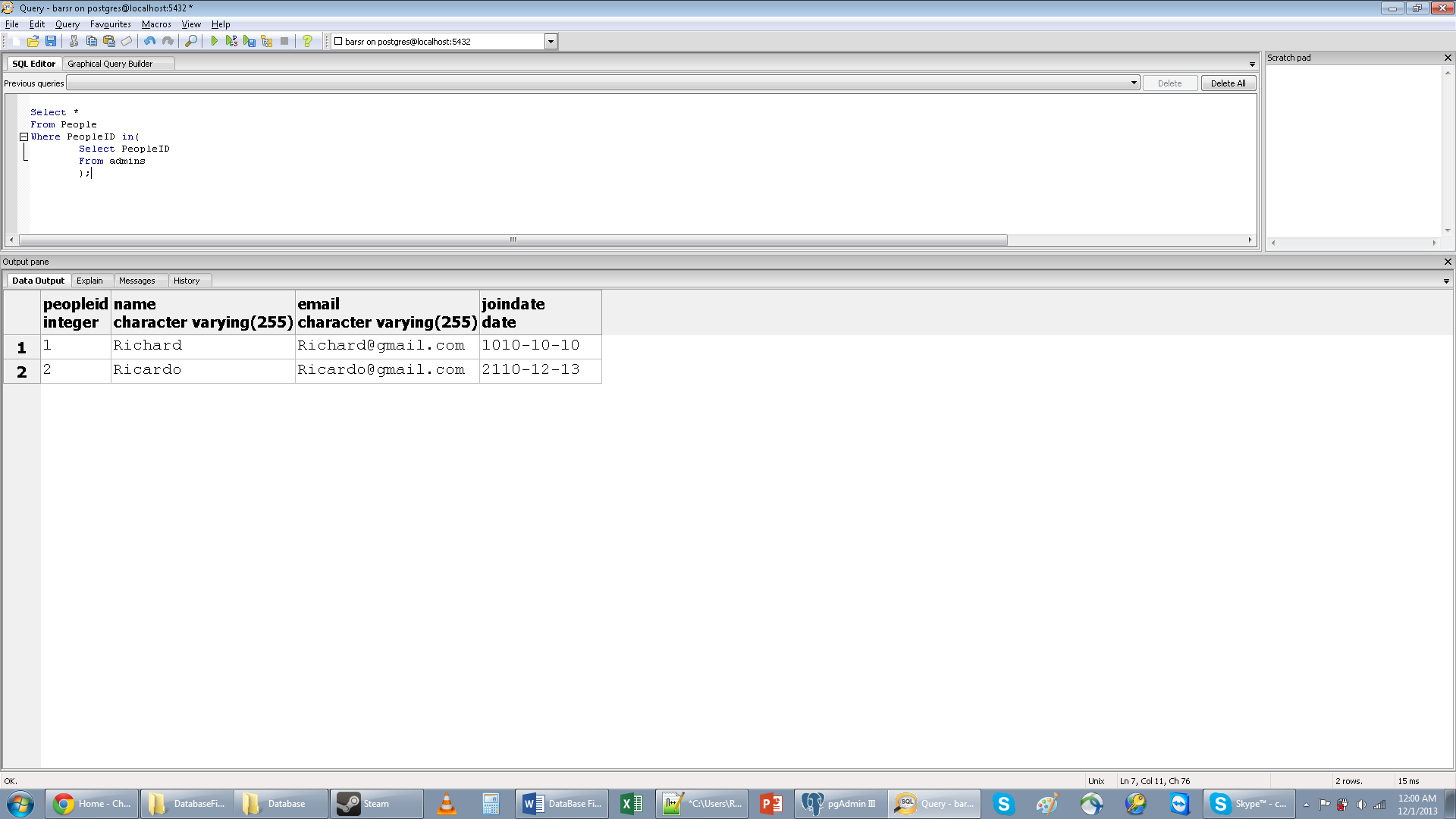
Select PeopleID

From admins

);

Order by PeopleID desc

**Sample Results**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Stored Procedures

This sample stored procedure will execute every time an update is applied to Classes\_Powers ensuring that no more than 6 powers are added to a single class.

**SQL Query\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

CREATE OR REPLACE FUNCTION CountClassPowers()

returns trigger as $$

Declare countpwr int =1;

BEGIN

loop

IF ((select count(CP.powerID)From Classes\_Powers CP Where ClassID= countpwr)<7)

THEN

DELETE from Classes\_Power

Where ClassID = countpwr;

END if;

countpwr=countpwr+1;

exit when countpwr < (select Max(ClassID) From Classes\_Powers);

end loop;

return new;

countpwr=0;

END

$$LANGUAGE plpgsql;

Triggers

This trigger will call CountClassPower() every time a power is added to a class.

**SQL Query\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create Trigger CountPowers

After Insert or Update

On Classes\_Powers

FOR EACH ROW Execute Procedure CountClassPowers();

Security

Although the end user’s interaction with the database is minimal and automatic in the future, security is still an issue. There are only two levels of security with the database: Admins and players. Admins have full access to the database while players can only view data in the database.

**SQL Grants(Admin)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create User admin With Password ‘admin’;

Revoke All on Items from admin;

Revoke All on admins from admin;

Revoke All on People from admin;

Revoke All on MapChunk from admin;

Revoke All on Character\_Items from admin;

Revoke All on People\_Games from admin;

Revoke All on Levels\_Map\_Chunks from admin;

Revoke All on Characters from admin;

Revoke All on Character\_Games from admin;

Revoke All on Games from admin;

Revoke All on Games\_Levels from admin;

Revoke All on Levels from admin;

Revoke All on Character\_Classes from admin;

Revoke All on Character\_Games from admin;

Revoke All on Classes from admin;

Revoke All on Kills\_Game from admin;

Revoke All on Enemies from admin;

Revoke All on Classes\_Powers from admin;

Revoke All on ESkills\_Enemies from admin;

Revoke All on Powers from admin;

Revoke All on EnemySkills from admin;

Grant insert, update, delete, select on Items to admin;

Grant insert, update, delete, select on admins to admin;

Grant insert, update, delete, select on People to admin;

Grant insert, update, delete, select on MapChunk to admin;

Grant insert, update, delete, select on Character\_Items to admin;

Grant insert, update, delete, select on People\_Games to admin;

Grant insert, update, delete, select on Levels\_Map\_Chunks to admin;

Grant insert, update, delete, select on Characters to admin;

Grant insert, update, delete, select on Character\_Games to admin;

Grant insert, update, delete, select on Games to admin;

Grant insert, update, delete, select on Games\_Levels to admin;

Grant insert, update, delete, select on Levels to admin;

Grant insert, update, delete, select on Character\_Classes to admin;

Grant insert, update, delete, select on Character\_Games to admin;

Grant insert, update, delete, select on Classes to admin;

Grant insert, update, delete, select on Kills\_Game to admin;

Grant insert, update, delete, select on Enemies to admin;

Grant insert, update, delete, select on Classes\_Powers to admin;

Grant insert, update, delete, select on ESkills\_Enemies to admin;

Grant insert, update, delete, select on Powers to admin;

Grant insert, update, delete, select on EnemySkills to admin;

**SQL Grants(Players)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Create User Player With Password 'pass';

Revoke All on Items from player;

Revoke All on admins from player;

Revoke All on People from player;

Revoke All on MapChunk from player;

Revoke All on Character\_Items from player;

Revoke All on People\_Games from player;

Revoke All on Levels\_Map\_Chunks from player;

Revoke All on Characters from player;

Revoke All on Character\_Games from player;

Revoke All on Games from player;

Revoke All on Games\_Levels from player;

Revoke All on Levels from player;

Revoke All on Character\_Classes from player;

Revoke All on Character\_Games from player;

Revoke All on Classes from player;

Revoke All on Kills\_Game from player;

Revoke All on Enemies from player;

Revoke All on Classes\_Powers from player;

Revoke All on ESkills\_Enemies from player;

Revoke All on Powers from player;

Revoke All on EnemySkills from player;

Grant select on Items to player;

Grant select on admins to player;

Grant select on Characters to player;

Grant select on Games to player;

Grant select on Levels to player;

Grant select on Enemies to player;

Grant select on Powers to player;

Grant select on EnemySkills to player;

Known Problems

* Map Chunks can be assigned to levels even if their ratings do not match
* Currently no way to know what Items will drop from enemies
* When more than 6 powers are added to a class all the powers associated with the class are dropped

Future Improvements

* Randomize map chunks to levels
* Randomize Character names based on class.
* Randomize items that enemies have/drop
* Show which level kills occurred on.