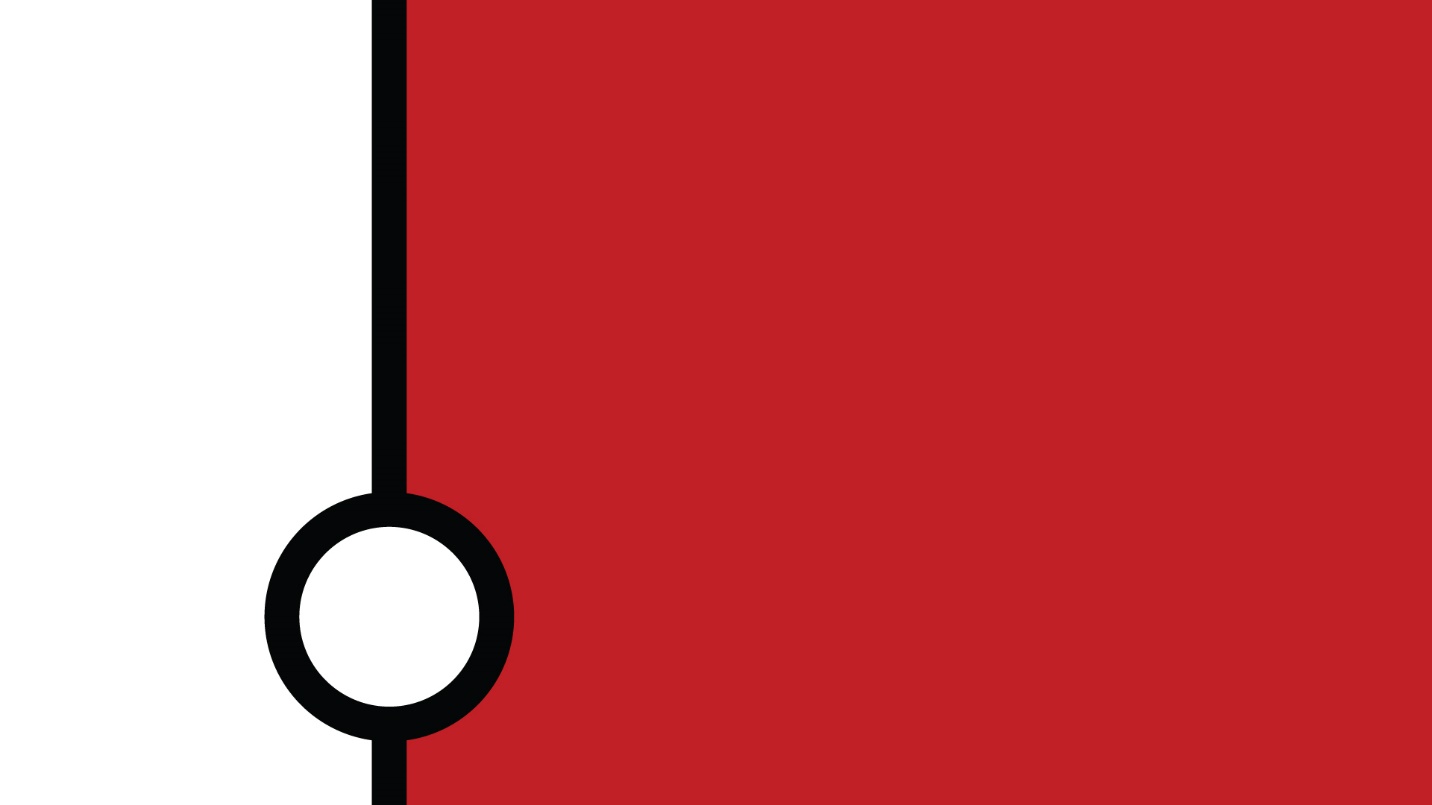


**Smart Pokédex**

**A Database Design Solution**

**By John Randis**

Contents

Executive Summary……………………………………………………….………………3

Entity Relationship Diagram………………………………………..….…….………4

Tables……………………………………………………………………………………….…..5

Views………………………………………………………………………………………..…17

Reports…………………………………………………………………………..…………….20

Stored Procedures……………………………………………………..………………..…21

Triggers……………………………………………………………………………………….22

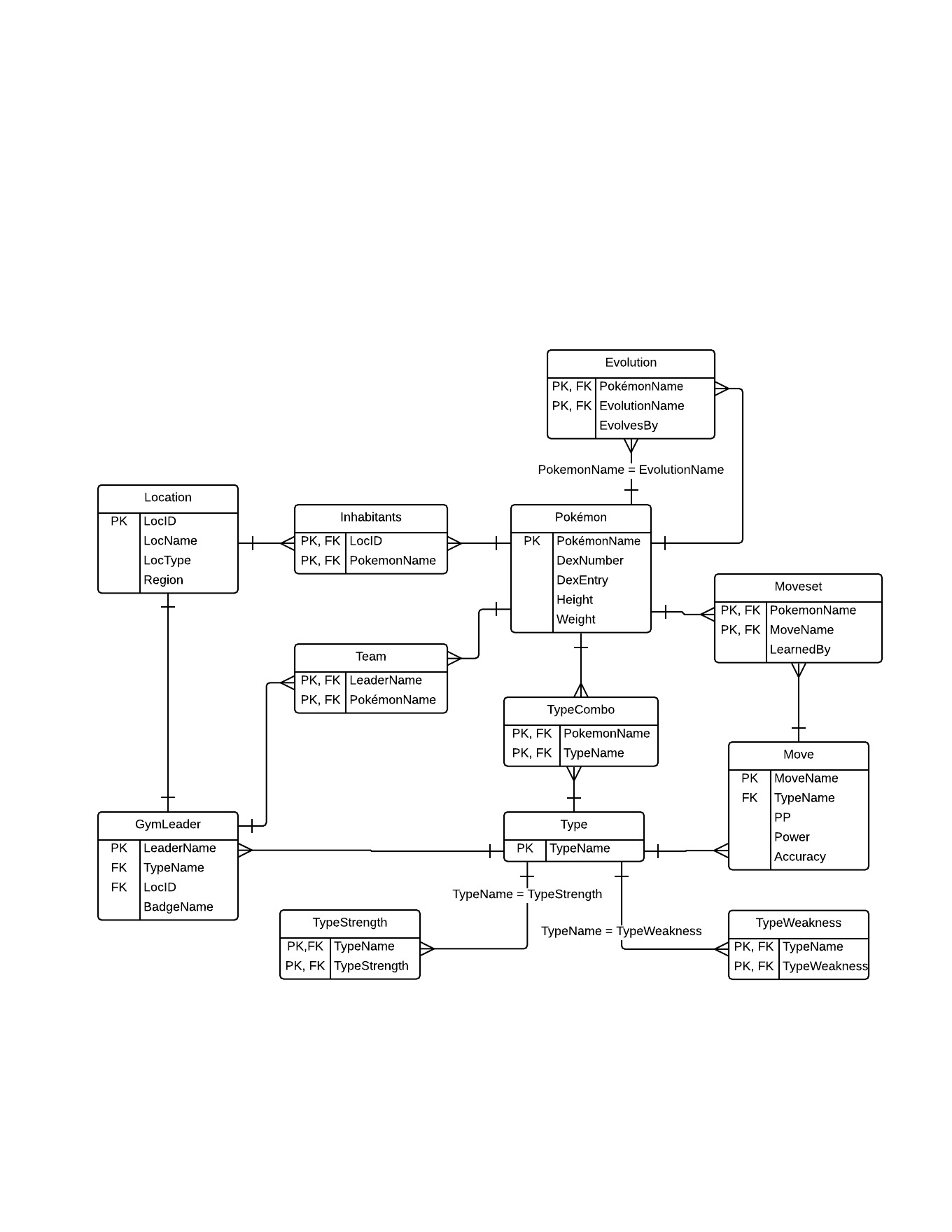
Security………………………………………………………….……………..….……….…23

Issues/Future Considerations………………………………………..……………...24

Executive Summary

The purpose of this database is to act as a resource for Pokémon trainers to manage discovered Pokémon. This database will include locations of Pokémon, their weaknesses, evolutions, and moves they learn, making it an invaluable resource no trainer should be caught without. Trainers can also store their Pokémon teams, as well as view and catalog the teams of gym leaders in order to be better prepared for a battle. This document includes the tables, functional dependencies, views, reports, procedures, and triggers that went in to creating the database. Using this Smart Pokédex, trainers will be able to keep an accurate and precise record of their Pokémon in order to become the very best.

Entity Relationship Diagram



Pokémon Table

There are hundreds of known Pokémon across the world and even more yet to be discovered. This table holds a record of each Pokémon, it’s dex number (order of discovery), a dex entry that provides a brief description of the Pokémon, and it’s height (m) and weight (kg).

Create Statement:

CREATE TABLE Pokemon (

PokemonName text not null,

DexNumber integer not null check (DexNumber > 0),

DexEntry text not null,

Height float not null check (Height > 0),

Weight float not null check (Weight > 0),

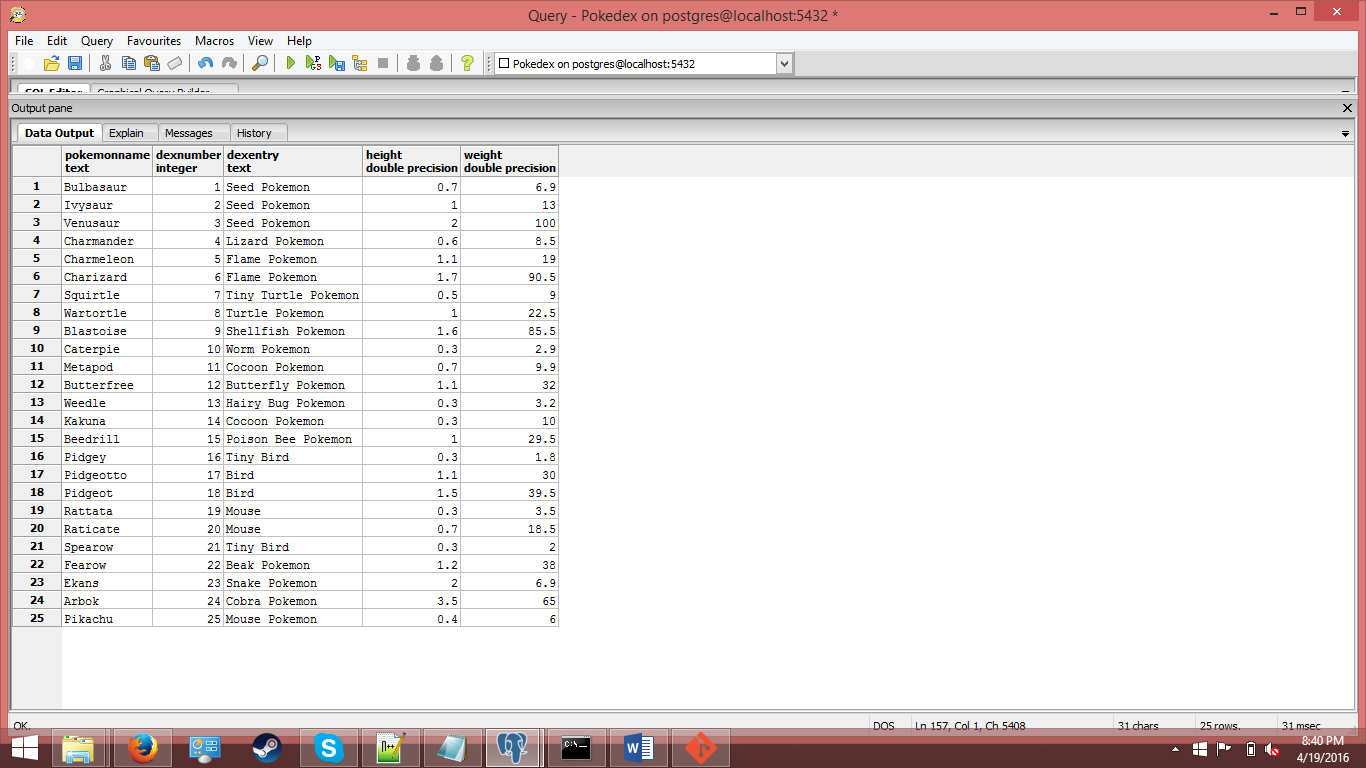
PRIMARY KEY (PokemonName)

);

Functional Dependencies:

PokemonName → DexNumber, DexEntry, Height, Weight

Sample Data:



Evolution Table

Pokémon that have the ability to evolve into new Pokémon will be added to this table. Users will be able to query the Pokémon’s name and get their evolution, as well as see what Pokémon come before in the evolutionary chain. Also includes the method by which the Pokémon evolves.

Create Statement:

CREATE TABLE Evolution (

PokemonName text references Pokemon(PokemonName) not null,

EvolutionName text references Pokemon(PokemonName) not null unique,

EvolvesBy EvolvesByEnum not null,

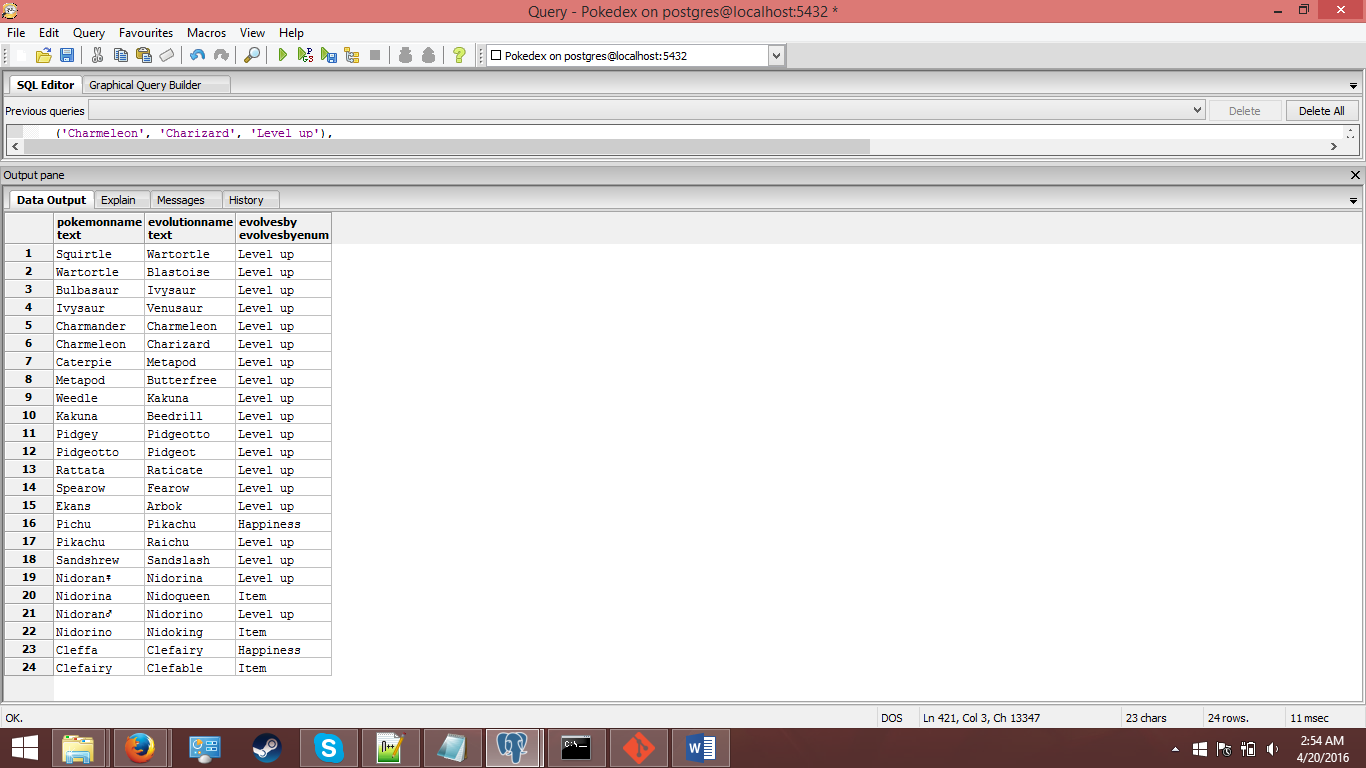
PRIMARY KEY (PokemonName, EvolutionName)

);

Functional Dependencies:

PokemonName → EvolutionName, EvolvesBy

Sample Data:



Location Table

The main purpose of the location table is so that users can query the Pokémon inhabitants at each location, in order to see where in the world a specific Pokémon resides.

Create Statement:

CREATE TABLE Location (

LocID serial not null,

LocName text not null,

LocType LocEnum,

Region RegionEnum,

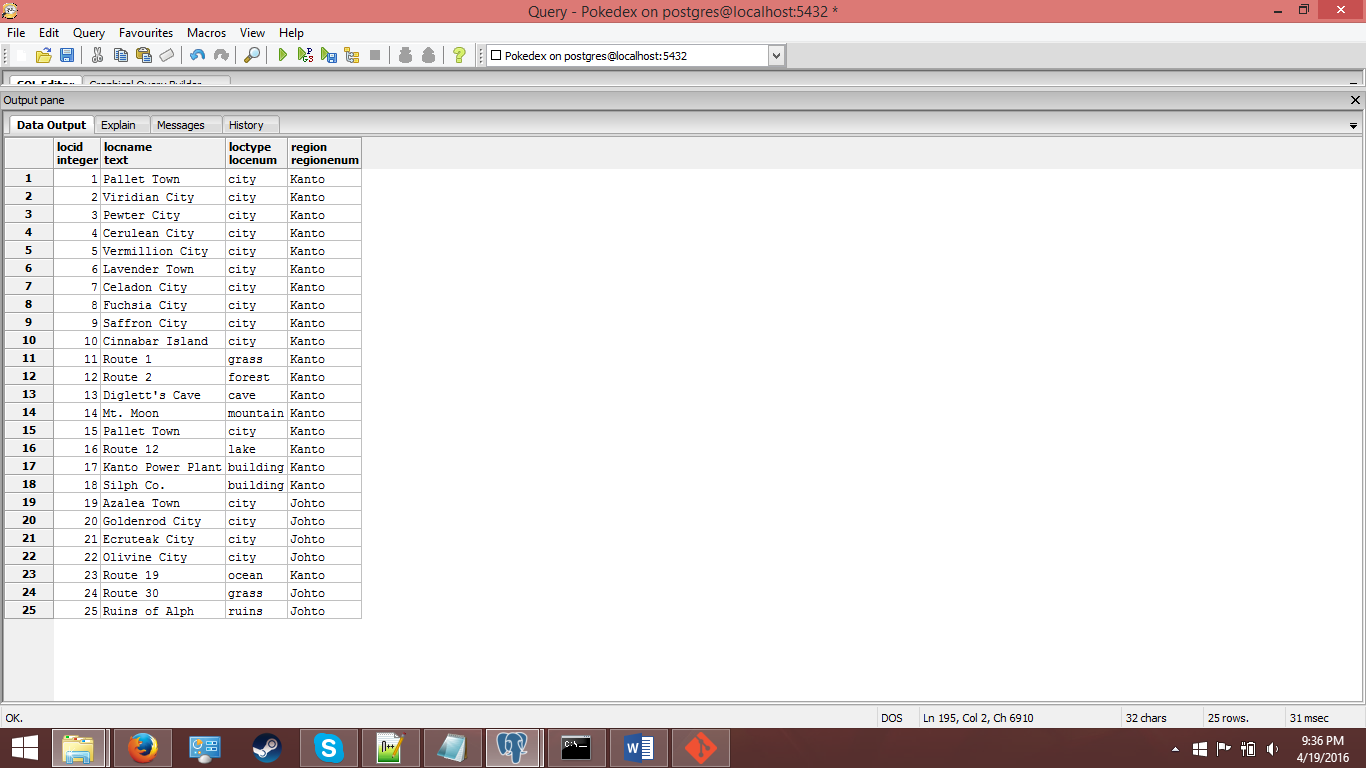
PRIMARY KEY (LocID)

);

Functional Dependencies:

LocID → LocName, LocType, Region

Sample Data:



Inhabitants Table

This table provides a link between Pokémon and Location. It stores all the locations of where a given Pokémon lives and where in the world it can be found.

Create Statement:

CREATE TABLE Inhabitants (

LocID integer not null references Location(LocID) not null,

PokemonName text references Pokemon(PokemonName) not null,

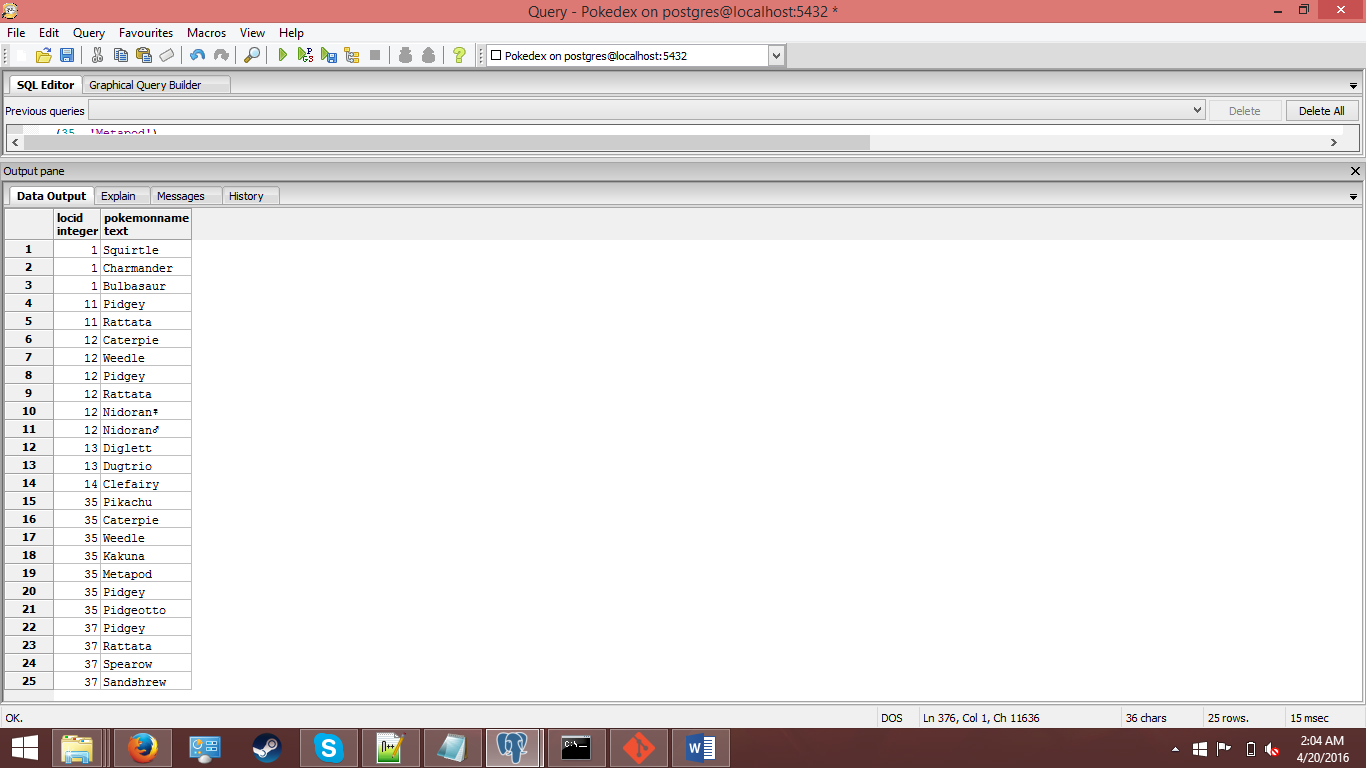
PRIMARY KEY (LocID, PokemonName)

);

Functional Dependencies:

LocID → PokemonName

Sample Data:



Type Table

An understanding of the elemental types is key for any trainer. Each type its own unique set of strengths and weaknesses to other types. Each Pokémon possesses at least one type, and each move has a type associated with it. Types are also adopted by gym leaders.

Create Statement:

CREATE TABLE Type (

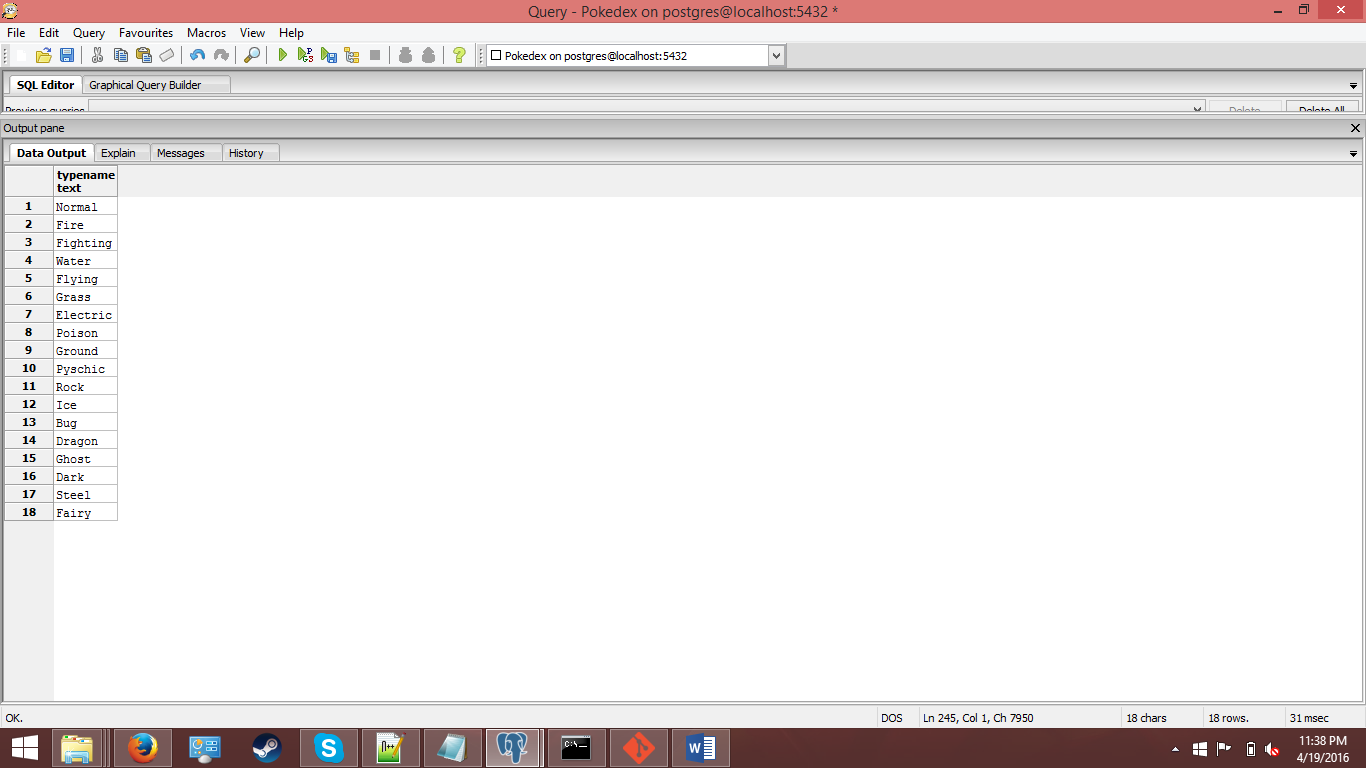
TypeName text not null unique,

PRIMARY KEY (TypeName)

);

Functional Dependencies: N/A

Sample Data:



Type Strength Table

A given type has its own specific set of types that it is strong against. That is, moves that of this type will be super-effect on Pokémon that belong to the types it is “strong” against. For instance, by querying the strengths of fire, we can see that fire-type moves will be super effective on grass, bug, ice, and steel type Pokémon.

Create Statement:

CREATE TABLE TypeStrength (

TypeName text not null references Type(TypeName) not null,

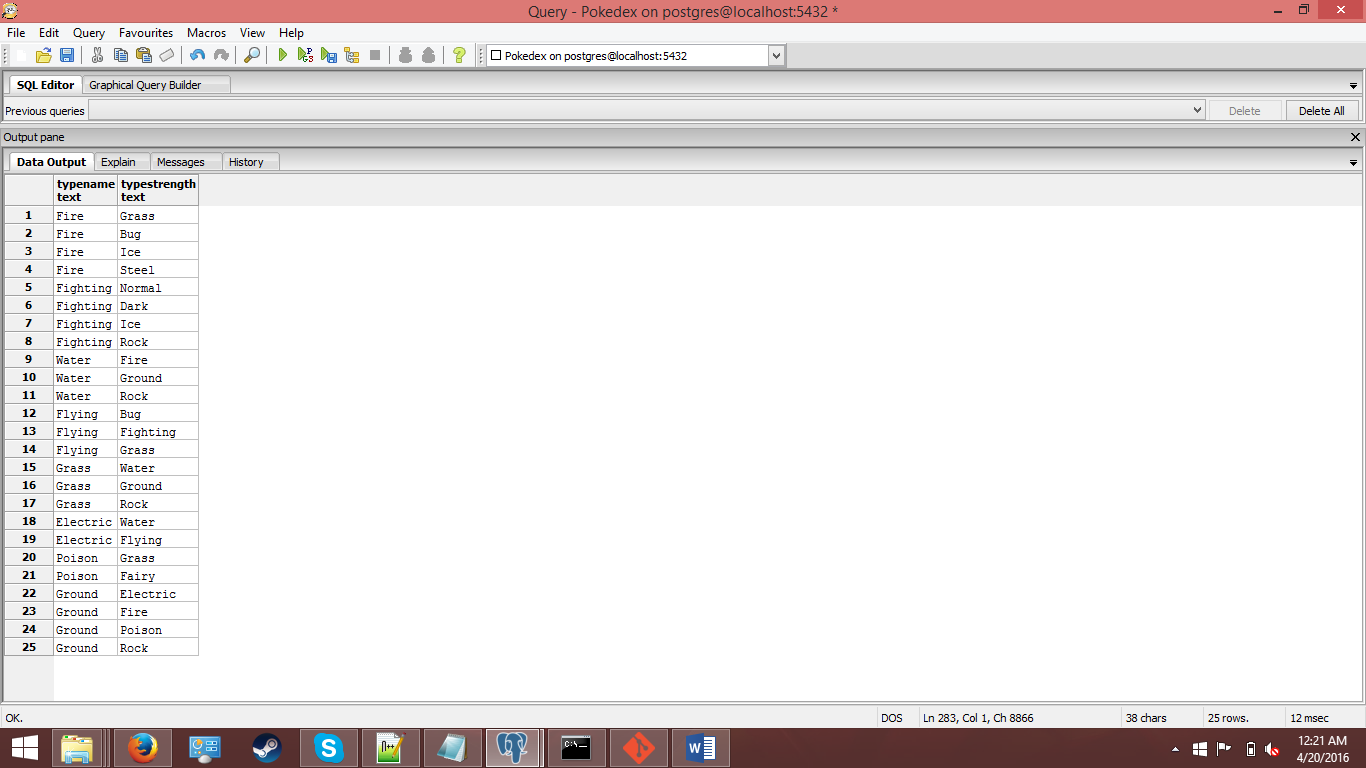
TypeStrength text not null references Type(TypeName) not null

);

Functional Dependencies:

TypeName → TypeStrength

Sample Data:



Type Weakness Table

A given type has its own specific set of types that it is weak against. Pokémon that belong to this type will be especially susceptible to moves possessing the type it is weak against. For instance, by querying the weaknesses of fire, we can see that water-type moves will be super effective on fire type Pokémon.

Create Statement:

CREATE TABLE TypeWeakness (

TypeName text not null references Type(TypeName) not null,

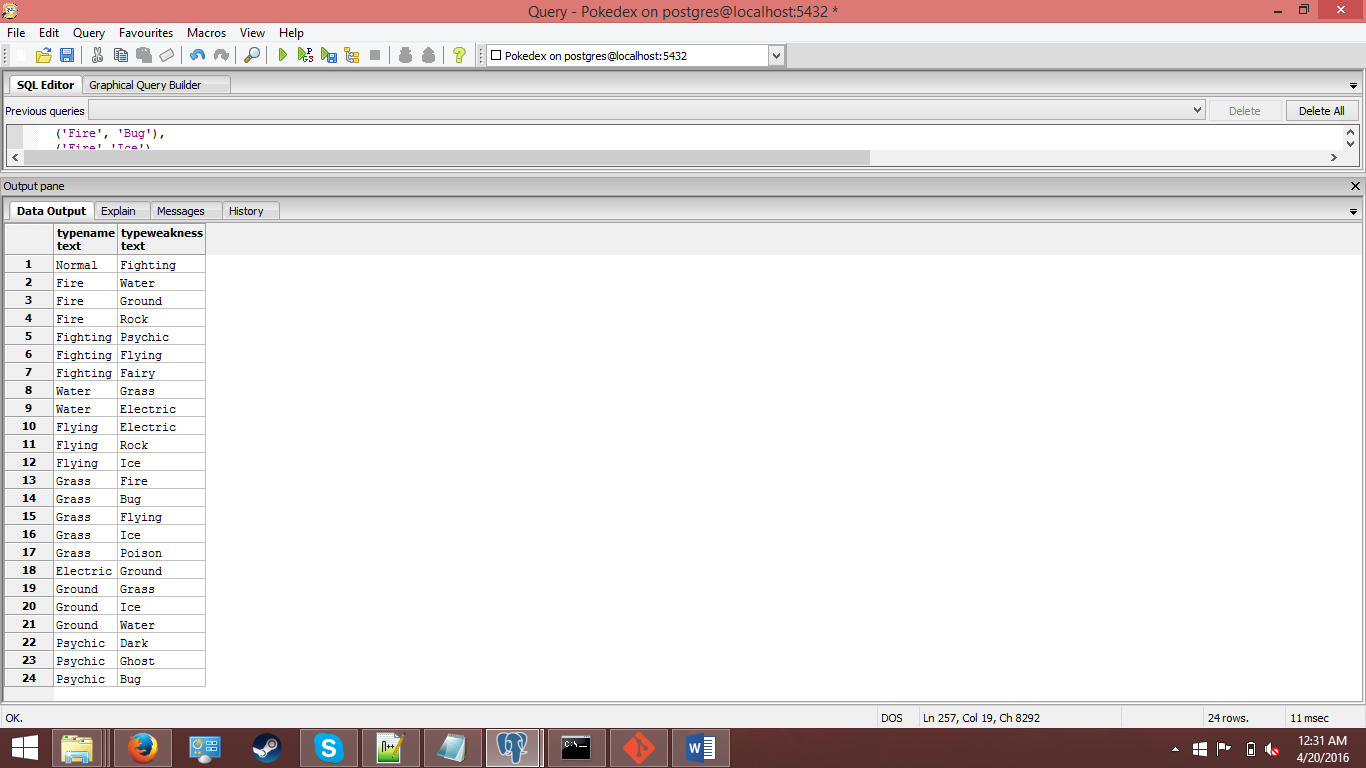
TypeWeakness text not null references Type(TypeName) not null

);

Functional Dependencies:

TypeName → TypeWeakness

Sample Data:



Type Combo Table

Each Pokémon can have up to two types. This table keeps a record every Pokémon in the database and the type or types they are affiliated with.

Create Statement:

CREATE TABLE TypeCombo (

PokemonName text references Pokemon(PokemonName) not null,

TypeName text not null references Type(TypeName) not null,

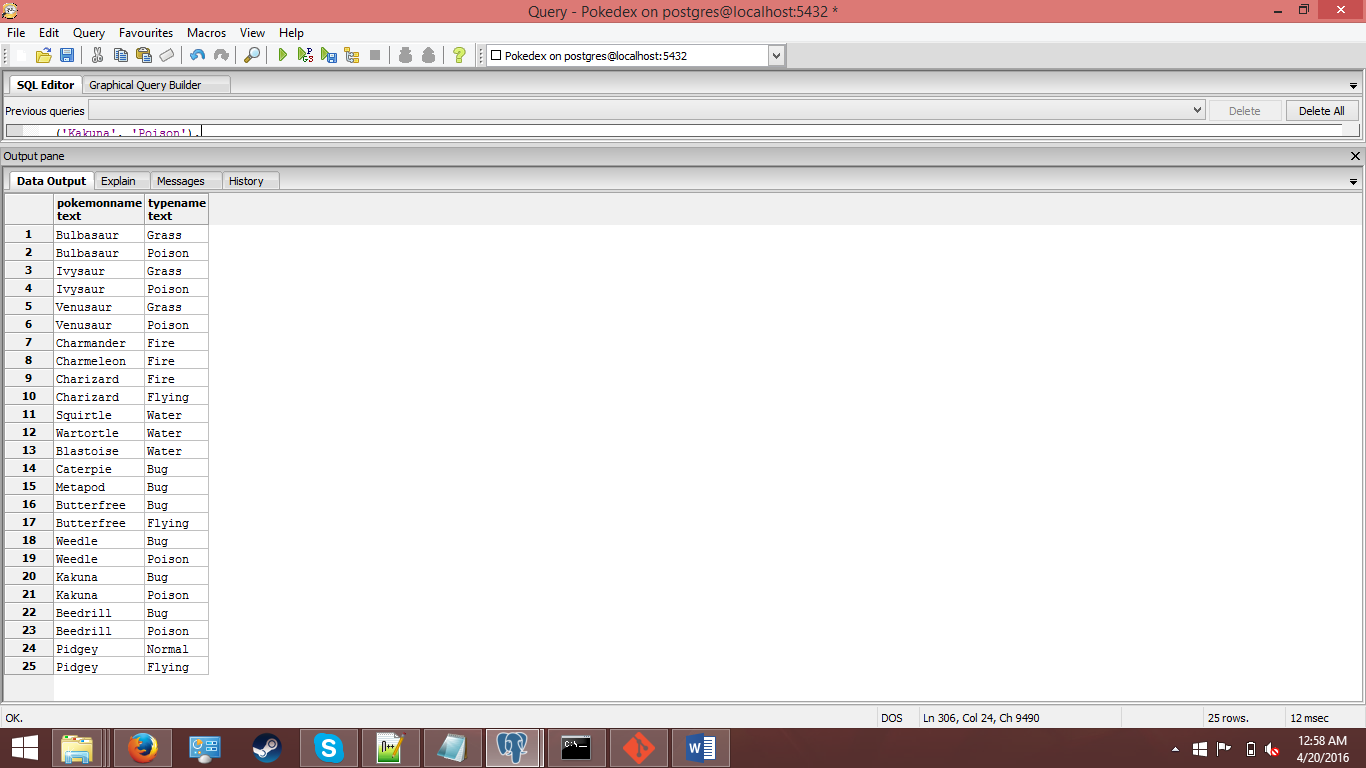
PRIMARY KEY (PokemonName, TypeName)

);

Functional Dependencies:

PokemonName → TypeName

Sample Data:



Move Table

Pokémon are able to learn a variety of moves to be used in battle. Moves all belong to an elemental type, and have base power and a base accuracy. Moves also possess PP, or power points, which determine how many times the move can be used.

Create Statement:

CREATE TABLE Move (

MoveName text not null,

TypeName text not null references Type(TypeName),

PP integer not null check (PP > 0),

Power integer check (Power > 0),

Accuracy integer check (Accuracy >= 0 and Accuracy <= 100),

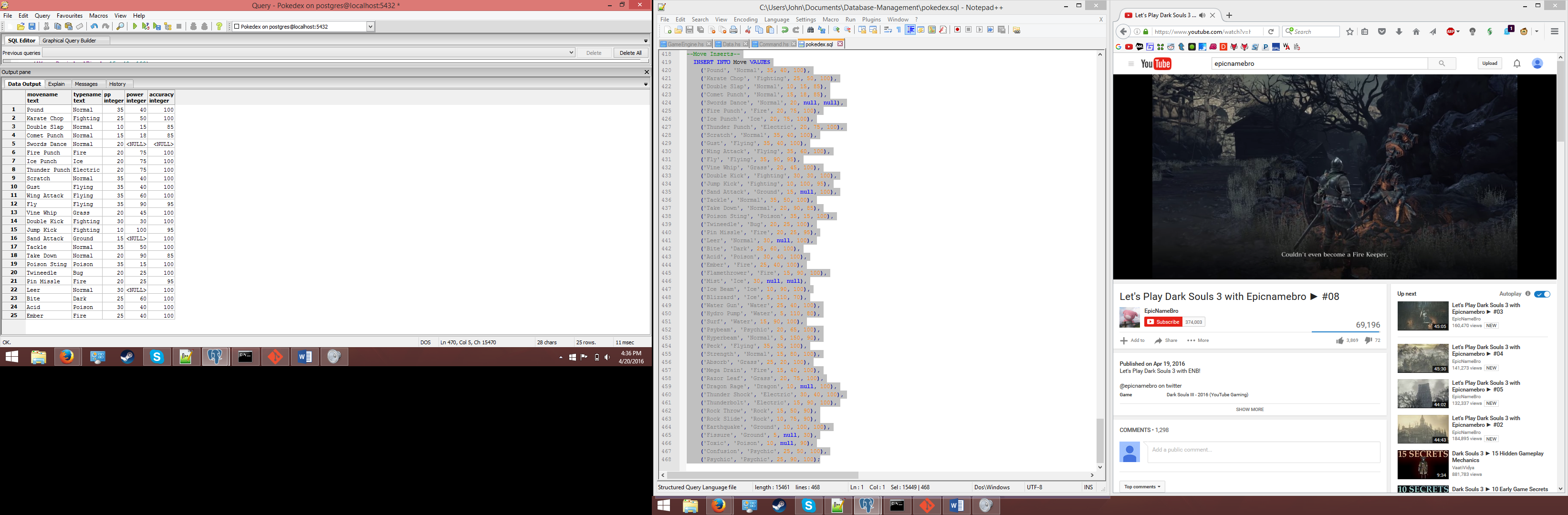
PRIMARY KEY (MoveName)

);

Functional Dependencies:

MoveName → TypeName, PP, Power, Accuracy

Sample Data:



Moveset Table

Each Pokemon has a wide set of moves it is able to learn, they either possess at birth by default, possess at birth by selective breeding, learn by leveling up, from a technical machine or hidden machine (TM/HM), or from a move tutor.

Create Statement:

CREATE TABLE Moveset (

PokemonName text not null references Pokemon(PokemonName),

MoveName text not null references Move(MoveName),

LearnedBy LearnedByEnum not null,

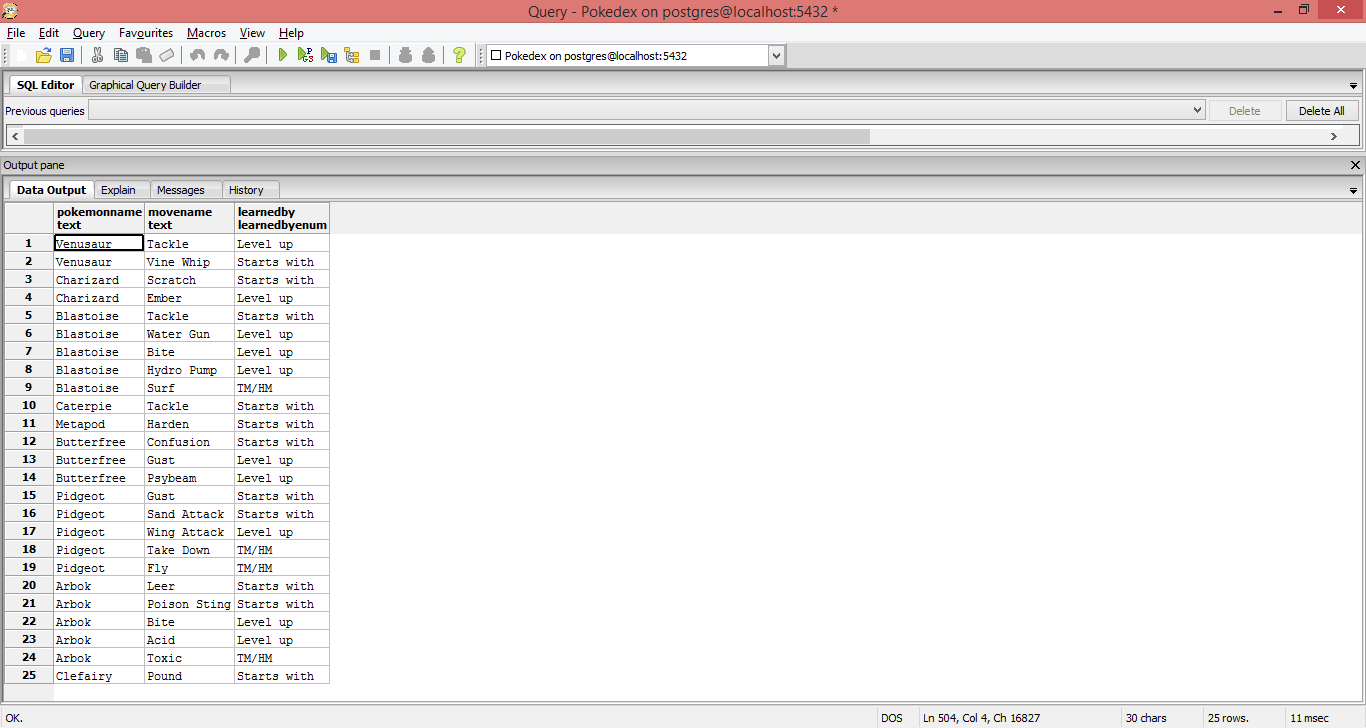
PRIMARY KEY (PokemonName, MoveName)

);

Functional Dependencies:

PokemonName → MoveName, LearnedBy

Sample Data:



Gym Leader Table

Gym leaders are the highest ranking trainers across the world, who exist for trainers to challenge and test their skill in order to obtain gym badges. Each gym leader affiliates themselves with a type. They only use Pokémon of this type and theme their gym and environment around this type in order to reflect their battling style. This table stores the leader’s name, their type, the location of their gym, and the name of their badge.

Create Statement:

CREATE TABLE GymLeader (

LeaderName text not null,

TypeName text references Type(TypeName),

LocID integer references Location(LocID) not null,

BadgeName text not null,

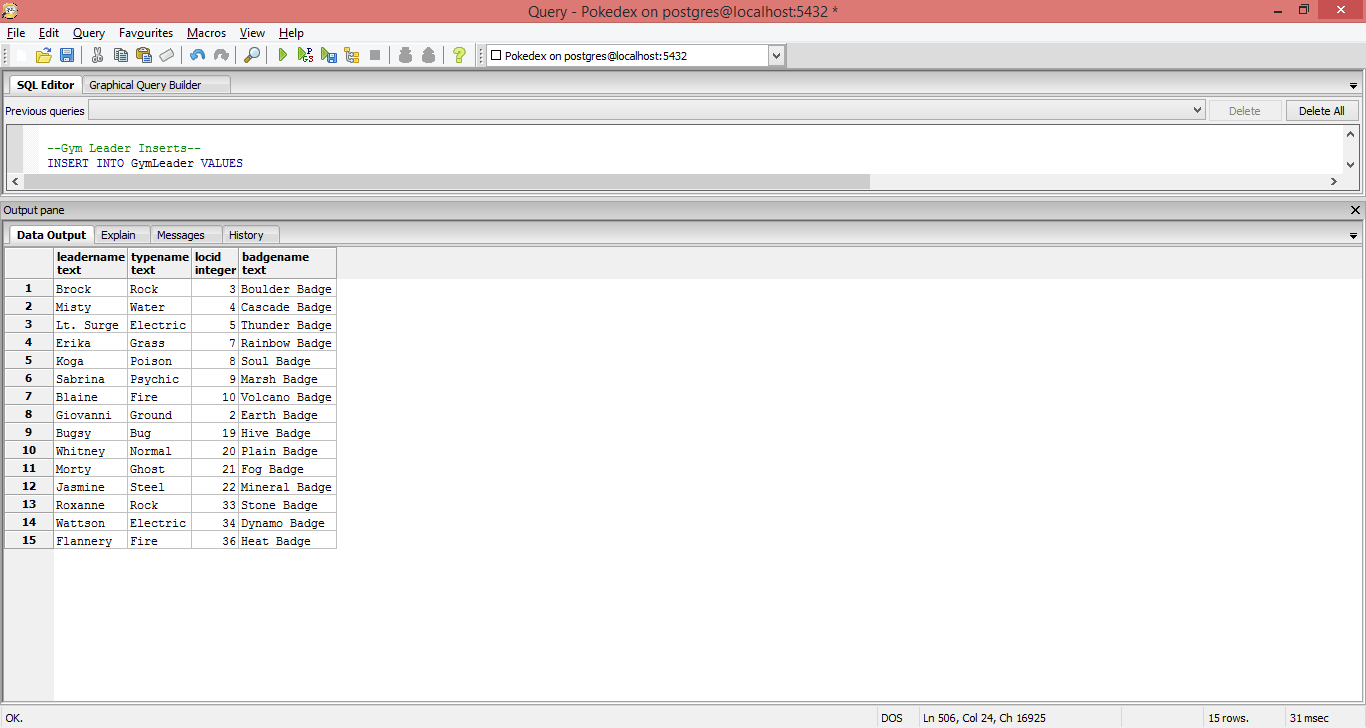
PRIMARY KEY (LeaderName)

);

Functional Dependencies:

LeaderName → TypeName, LocID, BadgeName

Sample Data:



Team Table

Each gym leader can have up to six Pokémon on their team. This table keeps track of the set of Pokémon that each leader will use in battle.

Create Statement:

CREATE TABLE Team (

LeaderName text not null references GymLeader(LeaderName) not null ,

PokemonName text references Pokemon(PokemonName) not null,

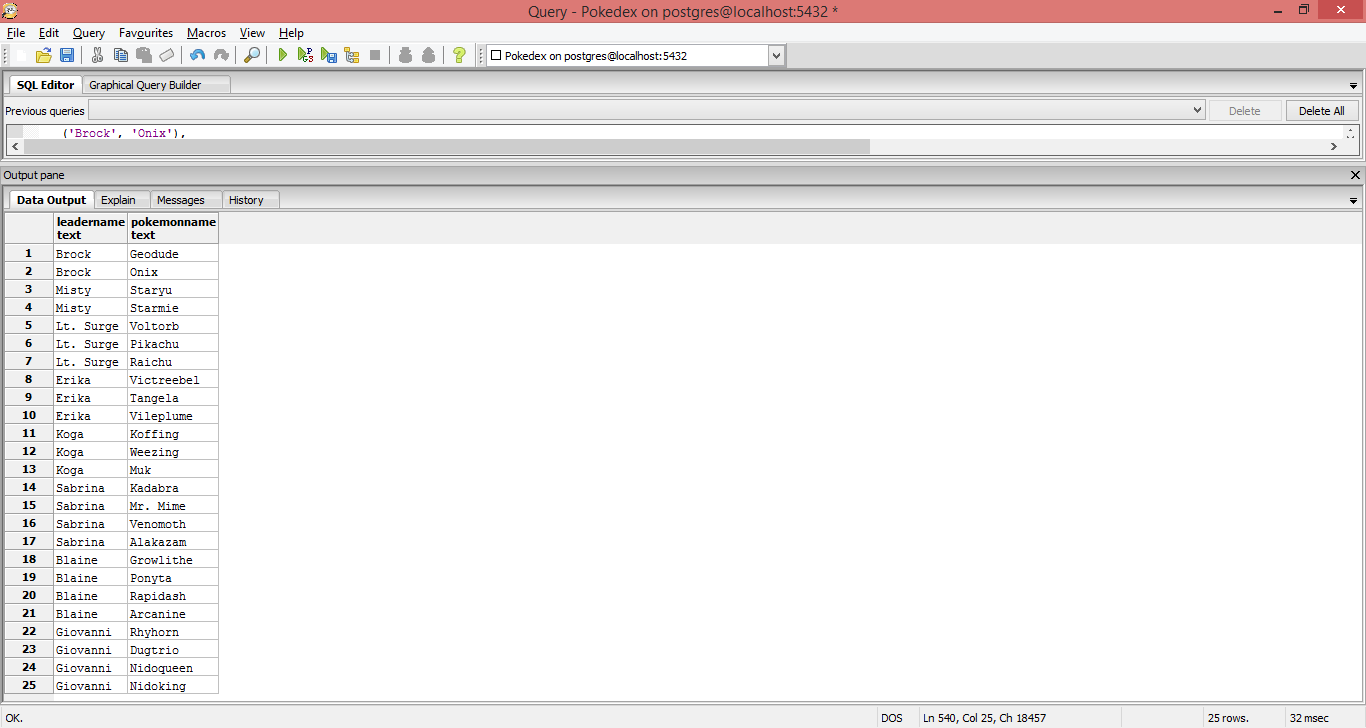
PRIMARY KEY (LeaderName, PokemonName)

);

Functional Dependencies:

LeaderName → PokemonName

Sample Data:



Gym Leader Locations View

This view shows all cities that contain a gym, and the gym leader that runs the gym in that city.

Create Statement:

CREATE VIEW GymLeaderLocations AS

SELECT Location.LocName,

GymLeader.LeaderName,

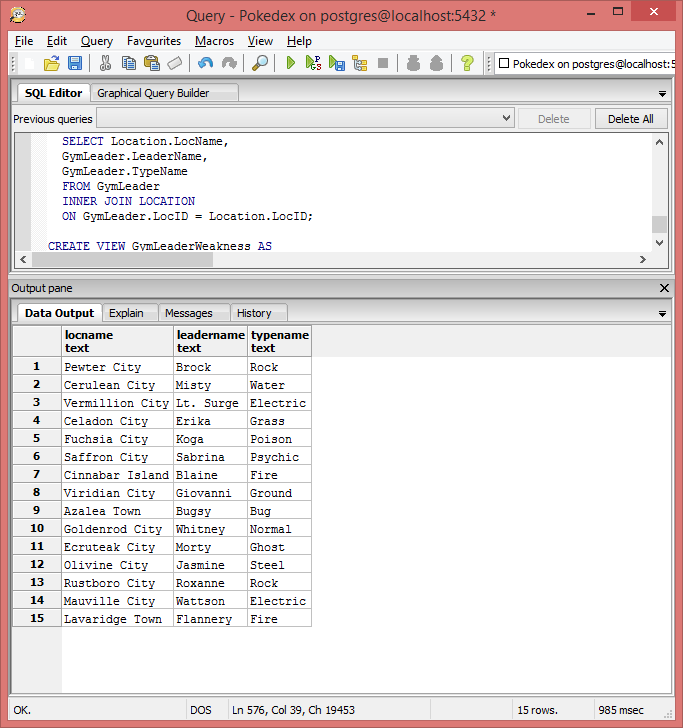
GymLeader.TypeName

FROM GymLeader

INNER JOIN Location

ON GymLeader.LocID = Location.LocID;

Sample Data:



Gym Leader Weakness View

This view will show each gym leader, their type, and all the weaknesses they have. This is essential for a trainer to be prepared for a battle with a gym leader.

Create Statement:

CREATE VIEW GymLeaderWeakness AS

Select GymLeader.LeaderName,

GymLeader.TypeName,

TypeWeakness.TypeWeakness

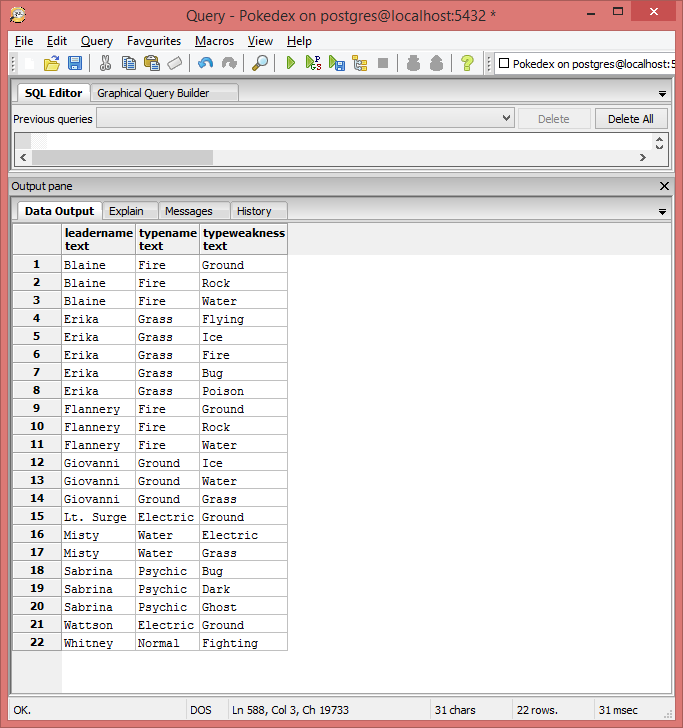
FROM GymLeader

INNER JOIN TypeWeakness

ON GymLeader.TypeName = TypeWeakness.TypeName

ORDER BY LeaderName ASC;

Sample Data:



Pokémon Location View

This view will display each Pokémon and the known locations they can be found. Useful resource for trainers who are looking where to find specific Pokémon.

Create Statement:

CREATE VIEW PokemonLocations AS

SELECT Inhabitants.PokemonName,

Location.LocName,

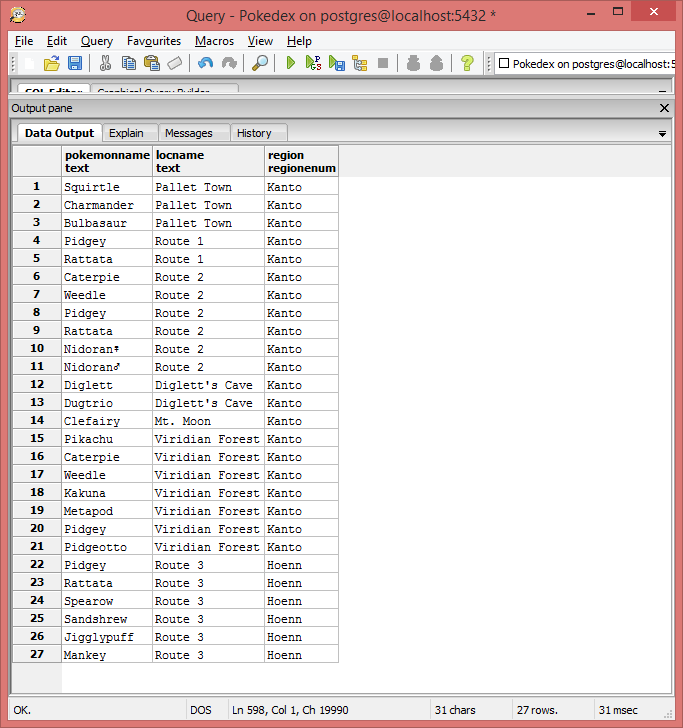
Location.Region

FROM Inhabitants

INNER JOIN Location

ON Inhabitants.LocID = Location.LocID;

Sample Data:



Strongest Type Report

This query will return the type which has the most appearances in the TypeStrengths table. In other words, the type which has the most strengths against other types.

Query:

SELECT TypeName,

COUNT(TypeName) AS TypeOccurence

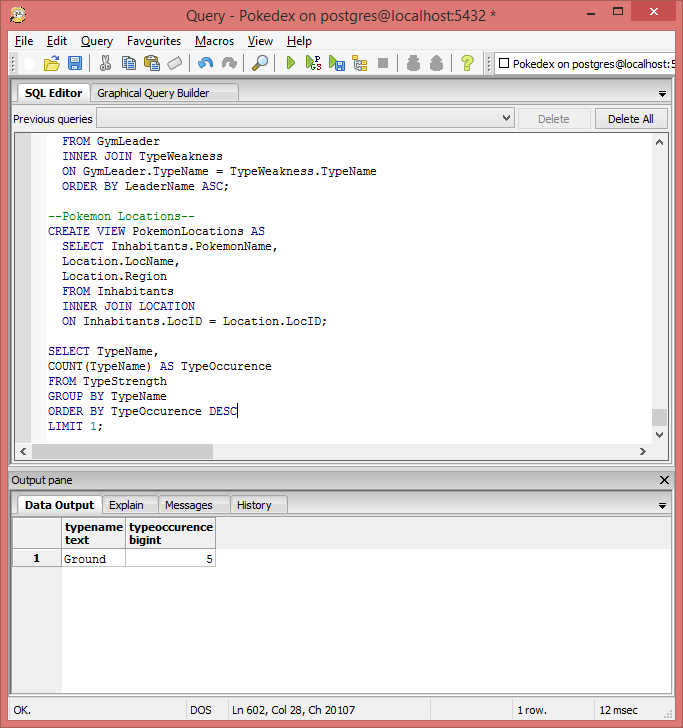
FROM TypeStrength

GROUP BY TypeName

ORDER BY TypeOccurence DESC

LIMIT 1;

Output:



Strongest Heavy Slam Users Report

Heavy Slam is a move that does damage based on the weight of the Pokémon. This query displays all the Pokémon that learn heavy slam, in order of heaviest to lightest.

Query:

SELECT PokemonName

FROM Pokemon

WHERE PokemonName in (

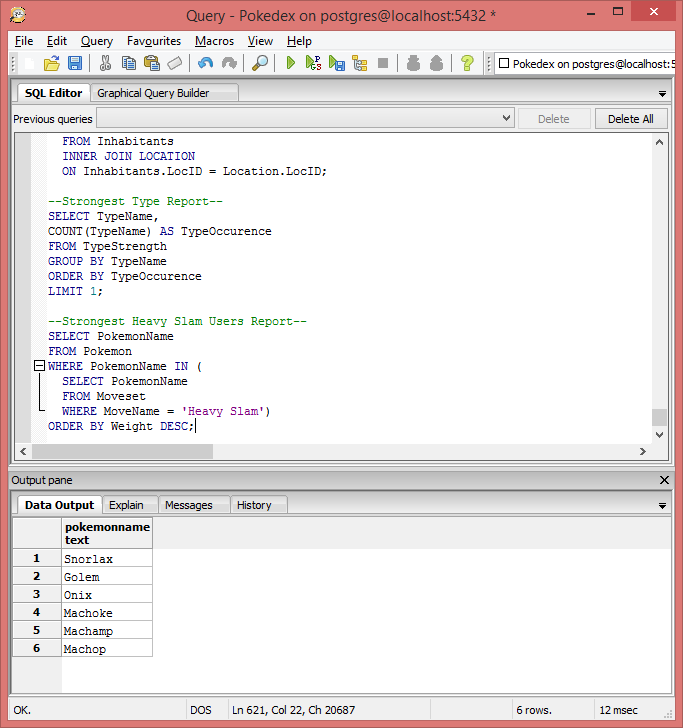
SELECT PokemonName

FROM Moveset

WHERE MoveName = 'Heavy Slam')

ORDER BY Weight DESC;

Output:



Moves of Prior Evolution Stored Procedure

Upon Evolution, Pokémon are able to retain any moves that were learned by their previous evolution. This function allows the user to specify an evolved Pokémon and returns all the moves that were able to be learned by its previous evolutions.

Function:

CREATE OR REPLACE FUNCTION get\_moves\_by\_prior\_evolution(text, REFCURSOR) RETURNS REFCURSOR AS

$$

DECLARE

evo text := $1;

resultset REFCURSOR := $2;

BEGIN

open resultset FOR

SELECT MoveName

FROM Moveset

WHERE PokemonName in (

SELECT PokemonName

FROM Evolution

WHERE EvolutionName = evo)

UNION

SELECT MoveName

FROM Moveset

WHERE PokemonName IN (

SELECT PokemonName

FROM Evolution

WHERE EvolutionName IN (

SELECT PokemonName

FROM Evolution

WHERE EvolutionName = evo));

RETURN resultset;

END;

$$

Language plpgsql; Insert Moves Upon Evolution Trigger

When a Pokémon is inserted into the evolution table, we can display the moves of their previous evolution. This will streamline inserting values into the moveset table.

CREATE TRIGGER set\_moves\_upon\_evolution

AFTER UPDATE ON Evolution

FOR EACH ROW EXECUTE PROCEDURE(PokemonName);

Trainer Role

While this database is designed as a resource to be used by trainers, they have no insertion privileges anywhere in the database. Trainers are able to query from any tables they like, but are not given access to update anything.

Role Creation:

CREATE ROLE Trainer;

GRANT SELECT ON

Pokemon, Evolution,

Inhabitants, Location,

Team, GymLeader,

Move, Moveset,

Type, TypeCombo,

TypeStrength, TypeWeakness

TO Trainer;

User Creation:

CREATE USER Red;

CREATE USER Blue;

CREATE USER Joey;

GRANT Trainer TO Red, Blue;

Leader Role

Gym Leaders have all the privileges trainers have, except they are also allowed the update the GymLeader and Team tables. They are allowed to insert into Team, in case a new Pokémon is added to their team.

Role Creation:

CREATE ROLE Leader;

GRANT SELECT ON

Pokemon, Evolution,

Inhabitants, Location,

Team, GymLeader,

Move, Moveset,

Type, TypeCombo,

TypeStrength, TypeWeakness

TO Leader;

GRANT UPDATE ON

GymLeader

TO Leader;

GRANT INSERT, UPDATE ON

Team

TO Leader;

User Creation:

CREATE USER Brock;

CREATE USER Misty;

CREATE USER Surge;

GRANT Leader TO Brock, Misty, Surge;

Administrator Role

Administrators are the creators of the Pokedex – Professor Oak, Professor Elm, and Bill. They have full privileges over everything in the database.

Role Creation:

CREATE ROLE Administrator;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO Administrator;

User Creation:

CREATE USER ProfessorOak;

CREATE USER ProfessorElm;

CREATE USER Bill;

GRANT Administrator TO ProfessorOak, ProfessorElm, Bill;

Known Issues

The following Issues are known in this database:

* While the trigger for setting moves upon evolution properly queries all the necessary moves, it will not execute themselves. To fix this one would have to edit or create a new stored procedure that uses insert statements into the moveset table in addition to the queries already in place.
* Some gym leaders possess more than one Pokémon on the same team. However, in the team table, LeaderName and PokemonName are both primary keys, so duplicate instances are not allowed. For example, while Koga has two Koffings, the database only records that he has one because the record for the second Koffing would not be unique. In order to fix this, we would need to create an arbitrary primary key, perhaps TeamID, to remove the primary key from PokemonName in the team table.
* Leaders are able to edit information not only about themselves, but other gym leaders. In order to fix this each gym leader would need his or her own table.

Future Enhancements

There are many features that I would have liked to include in this database, but could not due to time limitations. The following are implementations that could be included in the future:

* In its current form, the database shows the weaknesses for each type. However, an advanced implementation could show the weaknesses for each Pokémon. Pokémon with more than one type must have the aggregate strengths and weaknesses balanced out in order to display a dynamic and accurate record of strengths/weaknesses for each Pokémon. For instance, a user who owns a Bulbasaur may query the grass type Weakness and see that ground is weak to grass. However, Bulbsaur’s secondary typing, Poison, is weak to ground, making Bulbasaur take neutral damage from ground type moves.
* Another table that would be nice to be added in the future is an Abilities table. Pokémon can have up to 3 abilities each, each with their own specific effects.