Pokemon Database Project Final Report

Name: Deepa Subray Hegde
PSU ID: 954680127

Project Overview

This database idea is inspired by the famous 90's cartoon TV show called "Pokemon" which gets its name from "Pocket Monsters". It also has a series of video games that was developed by Game Freak and published by Nintendo.

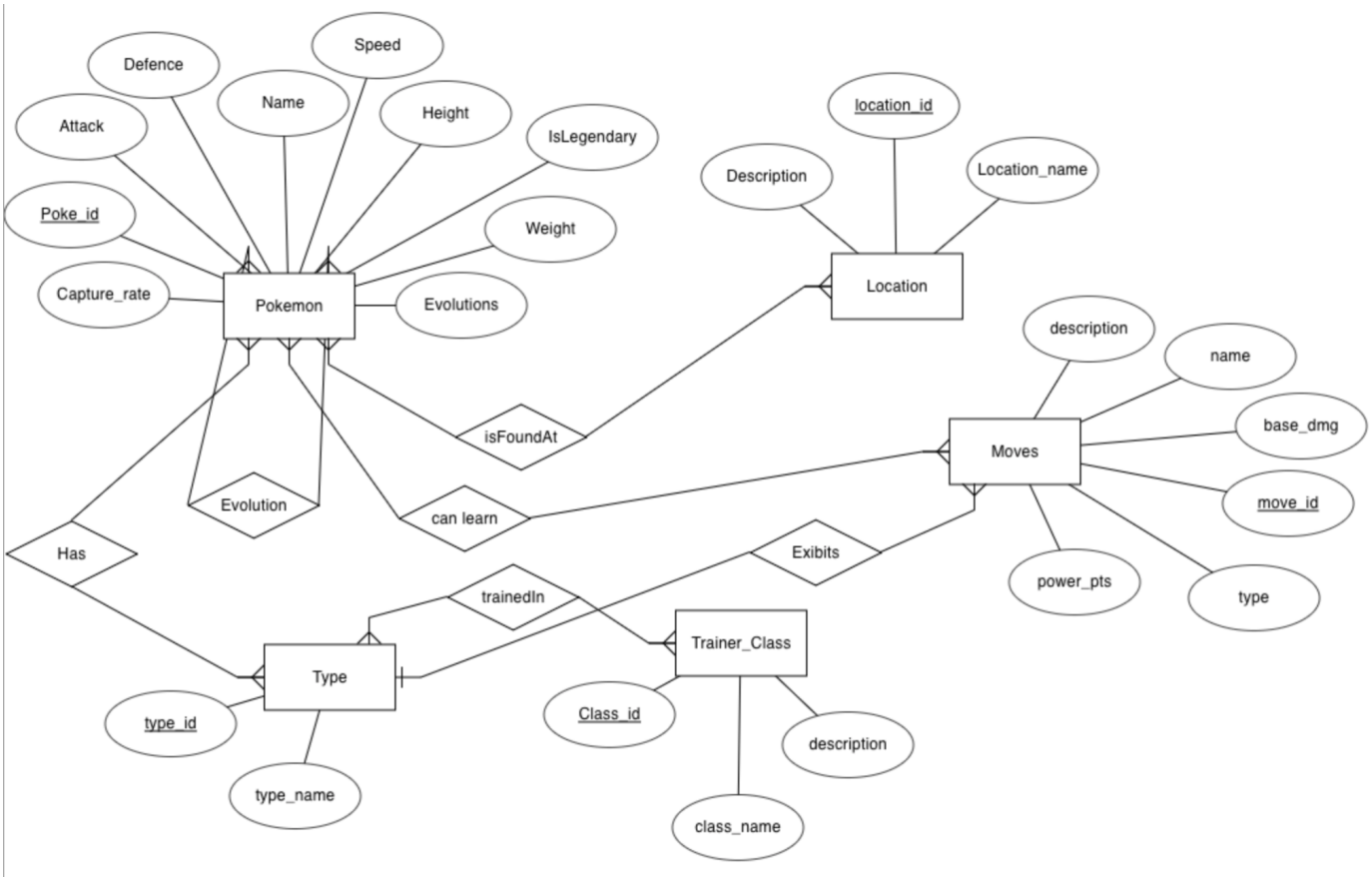
In this imaginative world, there are two main entities, Number one are the pokemons which are quirky monsters found throughout the world. They belong to a particular type (water, grass, fire, etc) and have their own unique skill set and statistics. The other entity are the trainers who travel to different places and try to capture these pokemon and put them in battle against other trainer's pokemon. This helps them to gain monetary benefits as well as gym badges that help them to become the very best pokemon trainer in the world. The database would be a replica of an item shown in the TV show called "Pokédex" which stores all the information about the pokemon, their types, moves and locations at which they are seen. As the TV show has become severely popular and has many different seasons, many new pokemons have been created. But for this project, I will be focusing on the original 151 pokemons which are found in the Kanto region and the database would revolve around the Generation I games (Red/Blue/Yellow) based on these 151 pokemons. (Some table examples include Pokemon, Trainer, moves, types, pokemon_location, pokemon_evolution_chain, etc)

Database Schema and Design

Pokemon Database Tables

1. **Pokemon** - The main table that contains details of all the Pokemons
2. **Type** - The domain table that stores details regarding different types that exist in the Pokemon world.
3. **Pokemon_Type_Rel** - This table provides information about the type that each Pokemon belongs to. Ex. Bulbasaur is a grass type as well as poison type Pokemon
4. **Pokemon_Evolutions** - This table stores information about Pokemon's prevolution
5. **Location** - This table will hold information about the location of the Pokemon
6. **Pokemon_Location_Rel** - This table contains information about the whereabouts of each Pokemon throughout the world.
7. **Moves** - This table contains information about different Pokemon moves.
8. **Pokemon_Moves_Rel** - This is a relation table between the pokemon table and the moves table to handle many to many relationship
9. **Trainer_Class** - This will have information about different trainer that are present in the Pokemon game
10. **Trainer_Class_Type_Rel** - This is a relation table between the trainer_class table and the type table to handle many to many relationship

ER Diagram



DDL SQL

```
CREATE TABLE Pokemon
(
        Poke_id   INT,
        Name   TEXT NOT NULL,
        Height   DECIMAL,
        Weight   DECIMAL,
        Capture_rate        INT,
        HP   INT,
        Attack   INT,
        Defense   INT,
        Special   INT,
        Speed   INT,
        Evolutions  INT,
        isLegendary  INT,
        PRIMARY KEY(Poke_id)
```

```
);

CREATE TABLE Type
(
        Type_id      SERIAL PRIMARY KEY,
        Type_name  VARCHAR(20)
);



CREATE TABLE Pokemon_Type_Rel
(
        pokemon_id   INT,
        type_id    INT,
        CONSTRAINT fk_pokemon FOREIGN KEY(pokemon_id) REFERENCES
        Pokemon(poke_id),
        CONSTRAINT fk_type FOREIGN KEY(type_id) REFERENCES Type(Type_id)
);

CREATE TABLE Location
(
        Location_id   SERIAL PRIMARY KEY,
        location_name  VARCHAR(100),
        description   TEXT
);


CREATE TABLE Pokemon_Location_Rel
(
        Pokemon_id     INT,
        location_id       INT,
        CONSTRAINT fk_pokemon FOREIGN KEY(pokemon_id) REFERENCES
        pokemon(poke_id),
        CONSTRAINT fk_location FOREIGN KEY(location_id) REFERENCES
        location(location_id)
);


CREATE TABLE Moves
(
        move_id      SERIAL PRIMARY KEY,
        name          VARCHAR(50),
        type          INT,
        base_dmg    INT,
```

```
        power_pts    INT,
        description   TEXT,
        CONSTRAINT fk_move_type FOREIGN KEY(type) REFERENCES type(type_id)
);

CREATE TABLE Pokemon_Moves_Rel
(
        pokemon_id        INT,
        move_id           INT,
        CONSTRAINT fk_pokemon FOREIGN KEY(pokemon_id) REFERENCES
        pokemon(poke_id),
        CONSTRAINT fk_move FOREIGN KEY(move_id) REFERENCES moves(move_id)
);

CREATE TABLE Pokemon_Evolutions
(
        pokemon_id        INT,
        pre_evolution_id    INT,
        isFinalEvolution     BOOL,
        CONSTRAINT fk_poke_pre_evol FOREIGN KEY(pre_evolution_id) REFERENCES
        Pokemon(Poke_id),
        CONSTRAINT fk_poke_id FOREIGN KEY(pokemon_id) REFERENCES
        Pokemon(Poke_id)
);


CREATE TABLE Trainer_Class
(
        class_id        INT   NOT NULL,
        class_name       VARCHAR(100),
        CONSTRAINT    trainer_class_pkey  PRIMARY KEY(class_id)
);


CREATE TABLE Trainer_Class_Type_Rel
(
        trainer_id       INT,
        type_id        INT,
        CONSTRAINT  fk_trainer FOREIGN KEY(trainer_id) REFERENCES
        trainer_class(class_id),
        CONSTRAINT   fk_type FOREIGN KEY(type_id) REFERENCES type(type_id)
);
```

Dataset references that are used to populate the tables

1. https://pokemondb.net/pokedex/game/red-blue-yellow
2. https://www.serebii.net/pokemon/gen1pokemon.shtml
3. https://www.kaggle.com/dizzypanda/gen-1-pokemon
4. https://www.kaggle.com/mariotormo/complete-pokemon-dataset-updated-090420

The above mentioned references had csv files which were downloaded and used for populating some of the tables.
The csv files were imported directly into the tables using phpPgAdmin.

Stored procedures are used to populate the 'rel' tables.

Below are the Stored Procedures used in this project

1. ```
   CREATE OR REPLACE PROCEDURE populate_pokemon_location_rel(poke_name varchar(100), loc_name varchar(100))
   LANGUAGE SQL
   AS $$
   INSERT INTO pokemon_location_rel (pokemon_id, location_id) VALUES ((SELECT poke_id FROM pokemon WHERE name=poke_name), (SELECT location_id FROM location WHERE location_name=loc_name));
   $$;
   ```

2. ```
   CREATE OR REPLACE PROCEDURE populate_pokemon_evolutions(pokename varchar(100), preevolpokename varchar(100), isfinalform bool)
   LANGUAGE SQL
   AS $$
   INSERT INTO pokemon_evolutions (pokemon_id, pre_evolution_id, isFinalEvolution)
   VALUES ((SELECT poke_id FROM pokemon WHERE name=pokename), (SELECT poke_id FROM pokemon WHERE name=preevolpokename), isfinalform);
   $$;
   ```

3. ```
   CREATE OR REPLACE PROCEDURE populate_pokemon_type_rel(pokename varchar(100), typename varchar(100))
   LANGUAGE SQL
   AS $$
   ```
   ```

```
INSERT INTO pokemon_type_rel (pokemon_id, type_id) VALUES ((SELECT poke_id
FROM pokemon WHERE name=pokename), (SELECT type_id FROM type WHERE
type_name=typename));
$$;
```

4.
```
CREATE OR REPLACE PROCEDURE populate_pokemon_moves_rel(pokeName text,
type1Name text, type2Name text)
LANGUAGE plpgsql
AS $$
DECLARE
type1_id INTEGER;
type2_id INTEGER;
type1_MoveCount      INTEGER := 3;
type2_MoveCount      INTEGER := 2;
pokeId           INTEGER;
moveId           INTEGER;

BEGIN
-- IF Pokemon has only 1 type then load all 5 moves of that type
IF (type2Name IS NULL) THEN
        type1_MoveCount := type1_MoveCount + type2_MoveCount;
        type2_MoveCount := 0;
END IF;

SELECT poke_id INTO pokeId FROM Pokemon WHERE name = pokeName;

SELECT type_id INTO type1_id FROM Type WHERE UPPER(type_name) =
UPPER(type1Name);

SELECT type_id INTO type2_id FROM Type WHERE UPPER(type_name) =
UPPER(type2Name);

FOR val IN 1..type1_MoveCount
LOOP
        SELECT move_id INTO moveId FROM Moves WHERE type = type1_id ORDER
        BY RANDOM() LIMIT 1;

        INSERT INTO Pokemon_moves_rel(pokemon_id, move_id) VALUES(pokeId,
        moveId);
END LOOP;

IF (type2_MoveCount != 0) THEN

        FOR val IN 1..type2_MoveCount
```

```
        LOOP
            SELECT move_id INTO moveId FROM Moves WHERE type = type2_id
            ORDER BY RANDOM() LIMIT 1;

            INSERT INTO Pokemon_moves_Rel(pokemon_id, move_id)
            VALUES(pokeId, moveId);
        END LOOP;
    END IF;

    END;
    $$;
```

The Stored Procedure are called and the tables are populated as shown below (sample statements)

```
CALL populate_pokemon_moves_rel('Bulbasaur','grass','poison');
CALL populate_pokemon_moves_rel('Ivysaur','grass','poison');
CALL populate_pokemon_moves_rel('Venusaur','grass','poison');
CALL populate_pokemon_moves_rel('Charmander','fire',NULL);
CALL populate_pokemon_moves_rel('Charmeleon','fire',NULL);
CALL populate_pokemon_moves_rel('Charizard','fire','flying');

CALL populate_pokemon_evolutions('Bulbasaur','',FALSE);
CALL populate_pokemon_evolutions('Ivysaur','Bulbasaur',FALSE);
CALL populate_pokemon_evolutions('Venusaur','Ivysaur',TRUE);
CALL populate_pokemon_evolutions('Charmander','',FALSE);

CALL populate_pokemon_location_rel('Caterpie','Route 25');
CALL populate_pokemon_location_rel('Caterpie','Viridian Forest');
CALL populate_pokemon_location_rel('Metapod','Route 24');
CALL populate_pokemon_location_rel('Metapod','Route 25');
CALL populate_pokemon_location_rel('Metapod','Viridian Forest');

CALL populate_pokemon_type_rel('Rocket', 'Psychic');
CALL populate_pokemon_type_rel('Rocket', 'Poison');
CALL populate_pokemon_type_rel('Rocket', 'Normal');
```

```
CALL populate_pokemon_type_rel('Rocket', 'Flying');
CALL populate_pokemon_type_rel('Green3', 'Flying');
```

Some of the tables are populated using simple INSERT INTO statements (sample statements)

```
INSERT INTO trainer_class VALUES (207, 'Pokemaniac');
INSERT INTO trainer_class VALUES (208, 'SuperNerd');
INSERT INTO trainer_class VALUES (209, 'Hiker');
INSERT INTO trainer_class VALUES (210, 'Biker');
INSERT INTO trainer_class VALUES (211, 'Burglar');

INSERT INTO Trainer_Class_Type_Rel VALUES (201, 14);
INSERT INTO Trainer_Class_Type_Rel VALUES (203, 8);
INSERT INTO Trainer_Class_Type_Rel VALUES (204, 6);
INSERT INTO Trainer_Class_Type_Rel VALUES (205, 4);
INSERT INTO Trainer_Class_Type_Rel VALUES (206, 7);
INSERT INTO Trainer_Class_Type_Rel VALUES (207, 2);

INSERT INTO Type (type_name) VALUES ('Bug');
INSERT INTO Type (type_name) VALUES ('Dragon');
INSERT INTO Type (type_name) VALUES ('Electric');
INSERT INTO Type (type_name) VALUES ('Fighting');
INSERT INTO Type (type_name) VALUES ('Fire');
```

20 SQL Queries

1. List all the names of Pokemon in the DB

   ```
   select name AS pokemon_names from pokemon;
   ```

pokemon_names
Bulbasaur
Ivysaur
Venusaur
Charmander
Charmeleon
Charizard
Squirtle
Wartortle
Blastoise
Caterpie
Metapod
Butterfree
Weedle
Kakuna
Beedrill
Pidgey
Pidgeotto
Pidgeot
Rattata
Raticate
Spearow
Fearow
Ekans
Arbok
Pikachu
Raichu

151 row(s)

Total runtime: 1.221 ms

SQL executed.

2. List the names of Pokemon living in a particular location

```
select name AS pokemon_name from pokemon p JOIN
pokemon_location_rel plr ON p.poke_id = plr.pokemon_id JOIN
location l ON l.location_id = plr.location_id WHERE l.location_name =
'Pallet Town';
```

pokemon_name
Poliwag
Tentacool
Goldeen

3 row(s)

Total runtime: 3.798 ms

SQL executed.

3. What is the average height and average weight of the pokemons that are present in the DB

```
select AVG(height) AS average_height, AVG(weight) AS
average_weight from pokemon;
```

average_height	average_weight
1.1947019867549669	45.9516556291390728

1 row(s)

Total runtime: 1.820 ms

SQL executed.

4. List all the pokemon that know a particular move

```
select p.name AS pokemon_name from pokemon p JOIN
pokemon_moves_rel pmr ON p.poke_id = pmr.pokemon_id JOIN
moves m ON m.move_id = pmr.move_id WHERE m.name =
'Confusion' GROUP BY p.name;
```

pokemon_name
Drowzee
Kadabra

2 row(s)

Total runtime: 3.347 ms

SQL executed.

5. Count all the pokemon who live in “Lavender Town”

```
select count(p.name) AS pokemon_name from pokemon p JOIN
pokemon_location_rel plr ON p.poke_id = plr.pokemon_id JOIN
location l ON l.location_id = plr.location_id WHERE l.location_name =
'Rock Tunnel';
```

pokemon_name
4

1 row(s)

Total runtime: 3.576 ms

SQL executed.

6. Find the predecessor name of a given pokemon

```
select p.name AS Predecessor from pokemon p WHERE p.poke_id
IN (select pre_evolution_id from pokemon_evolutions WHERE
pokemon_id IN (select poke_id from pokemon WHERE name =
'Charizard'));
```

predecessor
Charmeleon

1 row(s)

Total runtime: 1.888 ms

SQL executed.

7. Find the evolution name of a given pokemon

```
select p.name AS evolved_name from pokemon p WHERE p.poke_id
IN (select pokemon_id from pokemon_evolutions WHERE
pre_evolution_id IN (select poke_id from pokemon WHERE name =
'Pikachu'));
```

evolved_name
Raichu

1 row(s)

Total runtime: 1.740 ms

SQL executed.

8. Find the move that majority of the pokemon know

```
select name from moves JOIN (select move_id, COUNT(move_id) AS move_count from pokemon_moves_rel GROUP BY move_id ORDER BY move_count DESC LIMIT 1) x ON (x.move_id = moves.move_id);
```

name
Poisonpowder

1 row(s)

Total runtime: 4.472 ms

SQL executed.

9. How many types of pokemon are there in the DB

```
select COUNT(type_name) AS pokemon_type_count from type;
```

pokemon_type_count
15

1 row(s)

Total runtime: 1.173 ms

SQL executed.

10. List all the legendary pokemon present in the DB

```
select name AS pokemon_name from pokemon WHERE islegendary
= '1';
```

pokemon_name
Articuno
Zapdos
Moltres
Mewtwo
Mew

5 row(s)

Total runtime: 1.270 ms

SQL executed.

11. List all the trainers who have captured pokemon

```
select class_name from trainer_class WHERE class_id IN (select
trainer_id from trainer_class_type_rel GROUP BY trainer_id);
```

class_name
Youngster
Lass
Sailor
JrTrainerM
JrTrainerF
Pokemaniac
SuperNerd
Hiker
Biker
Burglar
Engineer
Fisher
Swimmer
CueBall
Gambler
Beauty
Psychic
Rocker

29 row(s)

Total runtime: 1.612 ms

SQL executed.

12. List all the pokemon that have more than 1 evolved form

```
select name AS pokemon_name from pokemon WHERE evolutions > 1;
```

pokemon_name
Bulbasaur
Ivysaur
Venusaur
Charmander
Charmeleon
Charizard
Squirtle
Wartortle
Blastoise
Caterpie
Metapod
Butterfree
Weedle
Kakuna
Beedrill
Pidgey
Pidgeotto
Pidgeot
Nidoran(F)

52 row(s)

Total runtime: 1.429 ms

SQL executed.

13. Find the pokemon which have the least base damage

```
select * from pokemon JOIN (select poke_id, SUM(attack) AS
base_damage from pokemon GROUP BY poke_id ORDER BY
base_damage ASC LIMIT 1) x ON (x.poke_id = pokemon.poke_id);
```

poke_id	name	height	weight	capture_rate	hp	attack	defense	special	speed	evolutions	islegendary	poke_id	base_damage
113	Chansey	1.1	34.6	30	250	5	5	105	50	0	0	113	5

1 row(s)

Total runtime: 3.839 ms

SQL executed.

14. Find the pokemon which has the least total base damage

```
select * from pokemon JOIN (select poke_id, SUM(attack + defense +
special + speed) AS total_base_damage from pokemon GROUP BY
poke_id ORDER BY total_base_damage ASC LIMIT 1) x ON
(x.poke_id = pokemon.poke_id);
```

poke_id	name	height	weight	capture_rate	hp	attack	defense	special	speed	evolutions	islegendary	poke_id	total_base_damage
39	Jigglypuff	0.5	5.5	170	115	45	20	25	20	1	0	39	110

1 row(s)

Total runtime: 3.103 ms

SQL executed.

15. List all the pokemon that have only one evolution

```
select name AS pokemon_name from pokemon WHERE evolutions =
1;
```

pokemon_name
Rattata
Raticate
Spearow
Fearow
Ekans
Arbok
Pikachu
Raichu
Sandshrew
Sandslash
Clefairy
Clefable
Vulpix
Ninetales
Jigglypuff
Wigglytuff

74 row(s)

Total runtime: 2.252 ms

SQL executed.

16. List all all trainers who belong to a particular type

```
select tc.class_name from trainer_class tc JOIN
trainer_class_type_rel ttr ON tc.class_id = ttr.trainer_id JOIN type t
ON t.type_id = ttr.type_id WHERE t.type_name = 'Ghost';
```

class_name
JrTrainerF
CueBall
Scientist

3 row(s)

Total runtime: 1.937 ms

SQL executed.

17. List all the pokemon who do not have any evolution

```
select p.name from pokemon p JOIN pokemon_evolutions pe ON
p.poke_id = pe.pokemon_id WHERE pe.pre_evolution_id is NULL
and pe.isfinalevolution = true;
```

name
Farfetchd
Onix
Lickitung
Chansey
Tangela
Kangaskhan
Scyther
Jynx
Electabuzz
Magmar
Pinsir
Tauros
Lapras
Ditto
Porygon
Aerodactyl
Snorlax
Articuno
Zapdos
Moltres
Mewtwo
Mew

22 row(s)

Total runtime: 4.001 ms

SQL executed.

18. Find the power points of the given move

```
select power_pts from moves WHERE name = 'Clamp';
```

power_pts
10

1 row(s)

Total runtime: 1.465 ms

SQL executed.

19. List all the pokemon and their evolution whose speed is 45 or less

```
select p.name from pokemon p full outer join pokemon_evolutions pe
on p.poke_id = pe.pokemon_id WHERE p.speed <= 45;
```

name
Bulbasaur
Squirtle
Caterpie
Metapod
Kakuna
Sandshrew
Nidoran(F)
Clefairy
Jigglypuff
Wigglytuff
Oddish
Gloom
Paras
Parasect
Venonat
Machop

38 row(s)

Total runtime: 1.720 ms

SQL executed.

20. List all the pokemon evolutions for pokemons who have capture rate 235 or more

```
select * from pokemon_evolutions pe right outer join pokemon p on p.poke_id = pe.pokemon_id WHERE p.capture_rate >= 235;
```

pokemon_id	pre_evolution_id	isfinalevolution	poke_id	name	height	weight	capture_rate	hp	attack	defense	special	speed	evolutions	islegendary
10	*NULL*	FALSE	10	Caterpie	0.3	2.9	255	45	30	35	20	45	2	0
13	*NULL*	FALSE	13	Weedle	0.3	3.2	255	40	35	30	20	50	2	0
16	*NULL*	FALSE	16	Pidgey	0.3	1.8	255	40	45	40	35	56	2	0
19	*NULL*	FALSE	19	Rattata	0.3	3.5	255	30	56	35	25	72	1	0
21	*NULL*	FALSE	21	Spearow	0.3	2	255	40	60	30	31	70	1	0
23	*NULL*	FALSE	23	Ekans	2	6.9	255	35	60	44	40	55	1	0
27	*NULL*	FALSE	27	Sandshrew	0.6	12	255	50	75	85	30	40	1	0
29	*NULL*	FALSE	29	Nidoran(F)	0.4	7	235	55	47	52	40	41	2	0
32	*NULL*	FALSE	32	Nidoran(M)	0.5	9	235	46	57	40	40	50	2	0
41	*NULL*	FALSE	41	Zubat	0.8	7.5	255	40	45	35	40	55	1	0
43	*NULL*	FALSE	43	Oddish	0.5	5.4	255	45	50	55	75	30	2	0
50	*NULL*	FALSE	50	Diglett	0.2	0.8	255	10	55	25	45	95	1	0
52	*NULL*	FALSE	52	Meowth	0.4	4.2	255	40	45	35	40	90	1	0
60	*NULL*	FALSE	60	Poliwag	0.6	12.4	255	40	50	40	40	90	2	0
69	*NULL*	FALSE	69	Bellsprout	0.7	4	255	50	75	35	70	40	2	0
74	*NULL*	FALSE	74	Geodude	0.4	20	255	40	80	100	30	20	2	0
129	*NULL*	FALSE	129	Magikarp	0.9	10	255	20	10	55	20	80	1	0

17 row(s)

Total runtime: 3.377 ms

SQL executed.