Pokemon: A Journey To a New Database!

BY: Jose Abreu

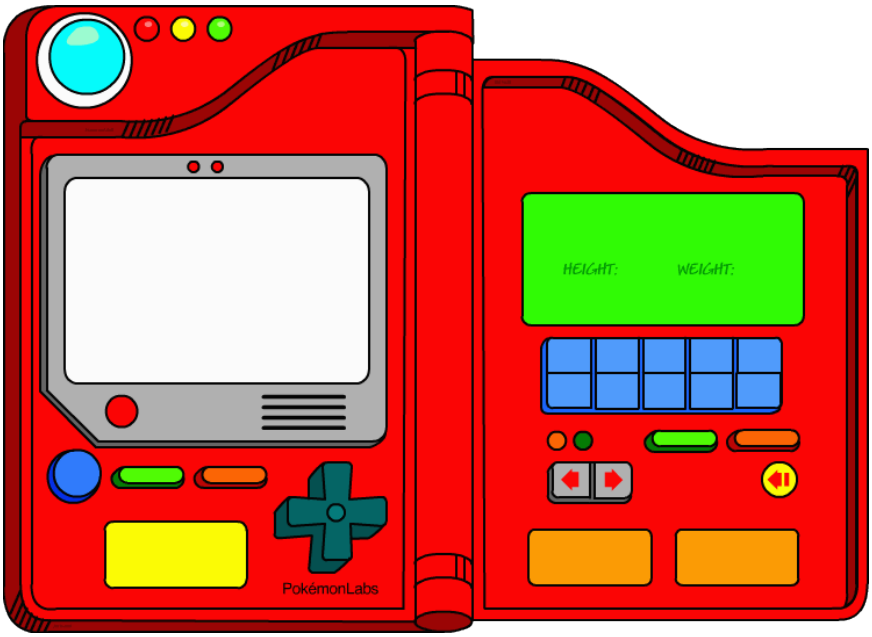


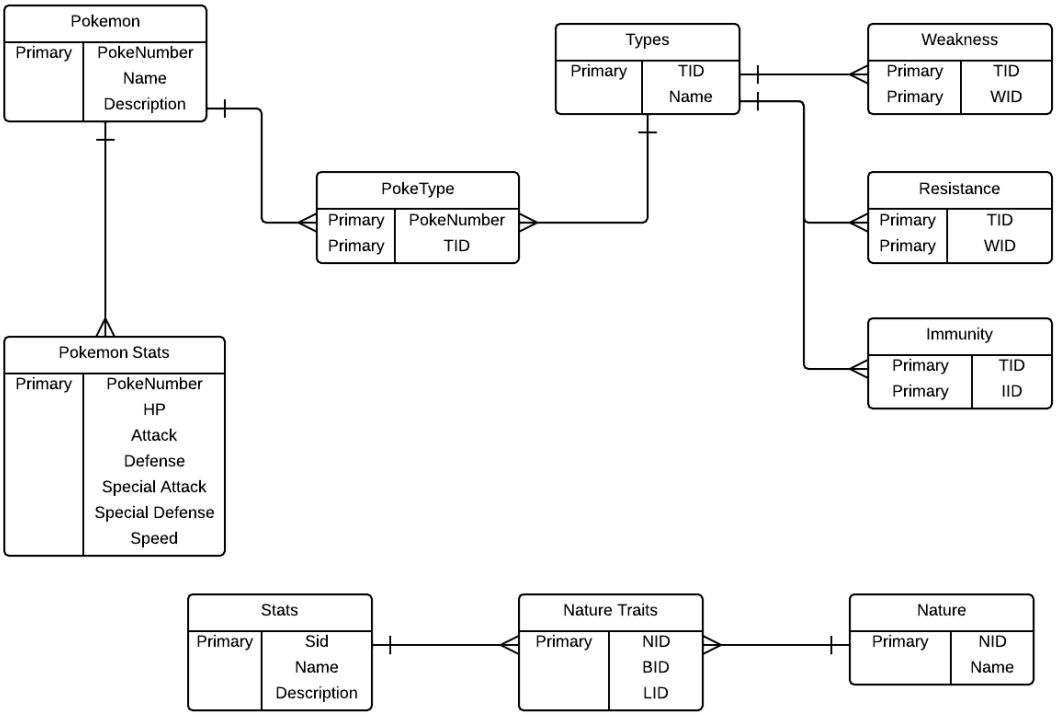
TABLE OF CONTENTS

Executive Summary

This documentation explains the design and the inner workings of a Pokémon Database. This database is used to see stats and description of every Pokémon as well as all 18 different types, their weaknesses, their resistances and any immunity that type may have. There are also descriptions of all 25 Natures, and what stats they increase and what stat they decrease. This can be used to build your ultimate team of Pokémon to show the world that you are the very best like no one ever was.

This database is made up of 10 different tables. The following Entity Relationship diagram will show how each table relates to one another, and how each piece of code will run its course one it is executed.

ENTITY RELATIONSHIP DIAGRAM



TABLES

POKEMON TABLE

The Pokémon Table holds general information for each Pokémon. It holds their Pokedex Number (Which is the Primary Key), their name, and a general description.

```
CREATE TABLE Pokemon (

PokeNumber integer not null,

Name      text,

Description text,

primary key(PokeNumber)

);
```

This is the output of the data:

	pokenumber integer	name text	description text
1	83	Farfetchd	The plant stalk it holds is its weapon. The stalk is used like a sword to cut all sorts of things.
2	91	Cloyster	Its shell is extremely hard. It cannot be shattered, even with a bomb. The shell opens only when it is attacking.
3	214	Heracross	This powerful Pokémon thrusts its prized horn under its enemies bellies, then lifts and throws them.
4	260	Swampert	It can swim while towing a large ship. It bashes down foes with a swing of its thick arms.
5	302	Sableye	It hides in the darkness of caves. Its diet of gems has transformed its eyes into gemstones.
6	303	Mawile	Attached to its head is a huge set of jaws formed by horns. It can chew through iron beams.
7	350	Milotic	Milotic is breathtakingly beautiful. Those that see it are said to forget their combative spirits.
8	354	Banette	A doll that became a Pokémon over its grudge from being junked. It seeks the child that disowned it.
9	542	Leavanny	Upon finding a small Pokémon, it weaves clothing for it from leaves by using the sticky silk secreted from its mouth.
10	556	Maractus	Arid regions are their habitat. They move rhythmically, making a sound similar to maracas.
11	635	Hydreigon	It responds to movement by attacking. This scary, three-headed Pokémon devours everything in its path!
12	666	Vivillon	Vivillon with many different patterns are found all over the world. These patterns are affected by the climate of their habitat.
13	693	Clawitzer	By expelling water from the nozzle in the back of its claw, it can move at a speed of 60 knots.
14	703	Carbink	Born from the temperatures and pressures deep underground, it fires beams from the stone in its head.

Functional Dependencies: PokeNumber → Name and Description

ELEMENT TYPE TABLE

This table contains all 18 types. Each type is given a Type ID (Tid) which is used as an artificial primary key.

CREATE TABLE ElementType (

tid integer not null, --tid is typeID--

Name text,

primary key(tid)

);

This is the output of the data:

	tid integer	name text
1	1	Normal
2	2	Fire
3	3	Water
4	4	Electric
5	5	Grass
6	6	Ice
7	7	Fighting
8	8	Poison
9	9	Ground
10	10	Flying
11	11	Psychic
12	12	Bug
13	13	Rock
14	14	Ghost
15	15	Dragon
16	16	Dark
17	17	Steel
18	18	Fairy

Functional Dependencies: TID → Name

POKEMON TYPE TABLE

This table shows the typing of every Pokémon listed in the Pokémon table. The primary key is a compound key of the PokeNumber from Pokémon and the TID from the Element Types table. This is done because each Pokémon can have more than one type.

CREATE TABLE PokemonType (

PokeNumber integer not null references Pokemon(PokeNumber),

tid integer references ElementType(tid),

primary key(PokeNumber, tid)

);

This is the following output:

	pokenumber integer	tid integer
1	83	1
2	83	10
3	91	3
4	91	6
5	214	12
6	214	7
7	260	3
8	260	9
9	302	16
10	302	14
11	303	17
12	303	18
13	350	3
14	354	14
15	542	12
16	542	5
17	635	16
18	635	15
19	666	12
20	666	10
21	693	3
22	703	13
23	703	18

Functional Dependencies: Pokemon.PokeNumber → PokeType.PokeNumber.

Types.TID → PokeType.TID

WEAKNESS

Each type has a certain weakness. This table is used to show what that weakness is. This table has the TID from Types and a WeaknessID(Wid) as a compound primary key. This is done because a type can have more than one weakness.

CREATE TABLE Weakness (

tid integer not null references ElementType(tid),

wid integer not null, --wid is Weakness ID--

primary key(tid,wid)

); This is the output:

	tid integer	wid integer
1	1	7
2	2	3
3	2	9
4	2	13
5	3	4
6	3	5
7	4	9
8	5	2
9	5	6
10	5	8
11	5	10
12	5	12
13	6	2
14	6	7
15	6	13
16	6	17
17	7	10
18	7	11
19	8	9
20	8	11
21	9	3
22	9	5
23	9	6
24	10	4
25	10	6
26	10	13
27	11	12
28	11	14
29	11	16
30	12	2
31	12	10
32	12	13
33	13	3
34	13	5
35	13	7
36	13	9
37	13	17
38	14	6
39	14	14
40	14	16
41	15	6
42	15	15
43	15	18
44	16	7
45	16	12
46	16	18
47	17	2
48	17	7
49	17	9
50	18	8
51	18	17

Functional Dependencies: ElementType.TID → Weakness.TID

RESISTANCE

Just as every type has its weakness, it also has a resistance. This table shows what type is resistant to what. As before, TID and Resistance (RID) are used as a compound key. This is done because types can have multiple resistances.

```
CREATE TABLE Resistance (
tid     integer not null references ElementType(tid),
rid     integer not null, --rid is Resistance ID--
primary key(tid,rid)
);
```

The output is far too big to be shown in this document so I will only show a piece of it:

	tid integer	rid integer
1	2	2
2	2	5
3	2	6
4	2	17
5	2	18
6	3	2
7	3	3
8	3	6
9	3	17
10	4	4
11	4	10
12	4	17
13	5	3
14	5	5

Functional Dependencies: ElementType.TID → Resistance.TID

IMMUNITY

Some types are immune to certain types. This table uses TID and ImmunityID (Iid) as a compound primary key because some types are immune to more than one type.

CREATE TABLE Immunity (

tid integer not null references ElementType(tid),

iid integer not null, --iid is Immunity ID--

primary key(tid,iid)

);

This is the output:

	tid integer	iid integer
1	1	14
2	9	4
3	10	9
4	14	1
5	14	7
6	16	11
7	17	8
8	18	15

Functional Dependencies: ElementType.Tid → Immunity.TID

POKEMON STATS TABLE

This table shows the base stats for every Pokémon that is in the Pokémon table. Each Pokémon is broken down into six different stats: HP, ATTACK, DEFENSE, SPECIAL ATTACK, SPECIAL DEFENSE and SPEED. The primary key for this table is PokeNumber from the Pokémon Table. There is no stat below 1 which is why I used the not null constraint.

```
CREATE TABLE PokemonStats (

PokeNumber        integer not null references Pokemon(PokeNumber),

HP                integer not null,

Attack            integer not null,

Defense                integer not null,

SpecialAttack     integer not null,

SpecialDefense          integer not null,

Speed             integer not null,

primary key(PokeNumber)

);
```

This is the output for the data:

	pokenumber integer	hp integer	attack integer	defense integer	specialattack integer	specialdefense integer	speed integer
1	83	52	65	55	58	62	60
2	91	50	95	180	85	45	70
3	214	80	125	75	40	95	85
4	260	100	110	90	85	90	60
5	302	50	75	75	65	65	50
6	303	50	85	85	55	55	50
7	350	95	60	79	100	125	81
8	354	64	115	65	83	63	65
9	542	75	103	80	70	80	92
10	635	92	105	90	125	90	98
11	666	80	52	50	90	50	89
12	693	71	73	88	120	89	59
13	703	50	50	150	50	150	50

Functional Dependencies: Pokemon.PokeNumber → PokemonStats.PokeNumber.

STATS

This table shows a breakdown of the six different stats as well as a StatID (sid) for each one. There is also a description of how each stat affects a Pokémon in battle. SID is used as an artificial primary key.

CREATE TABLE Stats (

Sid integer not null, --Sid stands for Stat ID--

Stat text,

Description text,

primary key(sid)

);

This is the output:

	sid integer	stat text	description text
1	1	HP	Hit Points. Determines how much damage a Pokémon can receive before fainting.
2	2	Attack	The Attack stat determines how much damage a Pokémon can deal using a physical move.
3	3	Defense	The Defense stat determines how much damage a Pokémon receives when it is hit with a physical move.
4	4	Special Attack	The Special Attack stat determines how much damage a Pokémon can deal using a special move.
5	5	Special Defense	The Special Defense stat determines how much damage a Pokémon receives when it is hit with a special move.
6	6	Speed	The Speed stat determines how quickly a Pokémon can act in battle.

Functional Dependencies: SID → Stat and Description

NATURES TABLE

Each Pokémon, when caught, will come with one of 25 different natures. Most natures raise one Stat and lower another. There are Natures however that do not raise or lower any stat. I created an NID as an artificial primary key for this table.

CREATE TABLE Nature (

nid integer not null, --Nid stands for Nature ID--

Name text,

primary key(nid)

);

This is the output:

	nid integer	name text
1	1	Hardy
2	2	Lonely
3	3	Brave
4	4	Adamant
5	5	Naughty
6	6	Bold
7	7	Docile
8	8	Relaxed
9	9	Impish
10	10	Lax
11	11	Timid
12	12	Hasty
13	13	Serious
14	14	Jolly
15	15	Naive
16	16	Modest
17	17	Mild
18	18	Quiet
19	19	Bashful
20	20	Rash
21	21	Calm
22	22	Gentle
23	23	Sassy
24	24	Careful
25	25	Quirky

Functional Dependicies: NID → Name

NATURE TRAITS TABLE

The purpose of this table is to show what stats are raised and lowered by each nature. The primary key is NID. Every nature that raises and lowers a stat has a BeneficialID(BID) and a LowerID(LID). If a Nature does not raise or lower a stat it will show as NULL.

CREATE TABLE NatureTraits (

Nid integer not null references Nature(nid),

Bid integer references Stats(sid), --Stands for Benefit ID. This will link with the Stat of the same number--

Lid integer references Stats(sid),

primary key(Nid)

);

This is the output:

	nid integer	bid integer	lid integer
1	1	<NULL>	<NULL>
2	2	2	3
3	3	2	6
4	4	2	4
5	5	2	5
6	6	3	2
7	7	<NULL>	<NULL>
8	8	3	6
9	9	3	4
10	10	3	5
11	11	6	2
12	12	6	3
13	13	<NULL>	<NULL>
14	14	6	4
15	15	6	5
16	16	4	2
17	17	4	3
18	18	4	6
19	19	<NULL>	<NULL>
20	20	4	5
21	21	5	2
22	22	5	3
23	23	5	6
24	24	5	4
25	25	<NULL>	<NULL>

VIEWS

POKEDEX

```
CREATE VIEW PokeDex

AS

SELECT      Pokemon.PokeNumber, Pokemon.Name, Pokemon.Description,

        PokemonStats.Hp, PokemonStats.Attack, PokemonStats.Defense,

        PokemonStats.SpecialAttack, PokemonStats.SpecialDefense, PokemonStats.Speed

from Pokemon

join PokemonStats

on Pokemon.PokeNumber = PokemonStats.PokeNumber

ORDER BY Pokemon.PokeNumber ASC;
```

This view will return a complete list of every Pokémon and their base stats.

	pokenumber integer	name text	description text	hp integer	attack integer	defense integer	specialattack integer	specialdefense integer	speed integer
1	83	Farfetchd	The plant	52	65	55	58	62	60
2	91	Cloyster	Its shell	50	95	180	85	45	70
3	214	Heracross	This powe:	80	125	75	40	95	85
4	260	Swampert	It can sw:	100	110	90	85	90	60
5	302	Sableye	It hides :	50	75	75	65	65	50
6	303	Mawile	Attached	50	85	85	55	55	50
7	350	Milotic	Milotic i:	95	60	79	100	125	81
8	354	Banette	A doll th:	64	115	65	83	63	65
9	542	Leavanny	Upon find:	75	103	80	70	80	92
10	635	Hydreigon	It respon:	92	105	90	125	90	98
11	666	Vivillon	Vivillon	80	52	50	90	50	89
12	693	Clawitzer	By expell:	71	73	88	120	89	59
13	703	Carbink	Born from	50	50	150	50	150	50

NATURES

CREATE VIEW Natures

AS

SELECT Nature.nid, Nature.name, S1.Stat as RaisedStat, S1.Description as RaisedDescription, S2.Stat as LoweredStat, S2.Description as LoweredDescription

from Nature

left join NatureTraits as N1

on N1.Nid = Nature.Nid

left join Stats as S1

on S1.Sid = N1.Bid

left join Stats as s2

on s2.Sid = N1.Lid

This table shows what stat each nature affects.

	nid integer	name text	raisedstat text	raiseddescription text	loweredstat text	lowereddescription text
1	1	Hardy	<NULL>	<NULL>	<NULL>	<NULL>
2	2	Lonely	Attack	The Attack stat det	Defense	The Defense stat
3	3	Brave	Attack	The Attack stat det	Speed	The Speed stat d
4	4	Adamant	Attack	The Attack stat det	Special At	The Special Atta
5	5	Naughty	Attack	The Attack stat det	Special De	The Special Defe
6	6	Bold	Defense	The Defense stat de	Attack	The Attack stat
7	7	Docile	<NULL>	<NULL>	<NULL>	<NULL>
8	8	Relaxed	Defense	The Defense stat de	Speed	The Speed stat d
9	9	Impish	Defense	The Defense stat de	Special At	The Special Atta
10	10	Lax	Defense	The Defense stat de	Special De	The Special Defe
11	11	Timid	Speed	The Speed stat dete	Attack	The Attack stat
12	12	Hasty	Speed	The Speed stat dete	Defense	The Defense stat
13	13	Serious	<NULL>	<NULL>	<NULL>	<NULL>
14	14	Jolly	Speed	The Speed stat dete	Special At	The Special Atta
15	15	Naive	Speed	The Speed stat dete	Special De	The Special Defe
16	16	Modest	Special A	The Special Attack	Attack	The Attack stat
17	17	Mild	Special A	The Special Attack	Defense	The Defense stat
18	18	Quiet	Special A	The Special Attack	Speed	The Speed stat d
19	19	Bashful	<NULL>	<NULL>	<NULL>	<NULL>
20	20	Rash	Special A	The Special Attack	Special De	The Special Defe
21	21	Calm	Special D	The Special Defense	Attack	The Attack stat
22	22	Gentle	Special D	The Special Defense	Defense	The Defense stat
23	23	Sassy	Special D	The Special Defense	Speed	The Speed stat d
24	24	Careful	Special D	The Special Defense	Special At	The Special Atta
25	25	Quirky	<NULL>	<NULL>	<NULL>	<NULL>

TYPE WEAKNESSES

CREATE VIEW TypeWeaknesses

AS

SELECT ElementType.Name, ElementType2.name as WeakAgainst

from ElementType

left join Weakness as Weakness1

on ElementType.tid = Weakness1.tid

left join ElementType as ElementType2

on ElementType2.tid = Weakness1.wid

This table shows the weaknesses of every type. The table is pretty big so I will only show a small part of it:

	name text	weakagainst text
1	Normal	Fighting
2	Fire	Rock
3	Fire	Ground
4	Fire	Water
5	Water	Grass
6	Water	Electric
7	Electric	Ground
8	Grass	Bug
9	Grass	Flying
10	Grass	Poison
11	Grass	Ice
12	Grass	Fire
13	Ice	Steel
14	Ice	Rock
15	Ice	Fighting
16	Ice	Fire
17	Fighting	Psychic
18	Fighting	Flying
19	Poison	Psychic
20	Poison	Ground
21	Ground	Ice
22	Ground	Grass
23	Ground	Water
24	Flying	Rock

RESISTANCE

CREATE VIEW TypeResists

AS

SELECT ElementType.Name, ElementType2.name as Resists

from ElementType

left join Resistance as Resistance1

on ElementType.tid = Resistance1.tid

left join ElementType as ElementType2

on ElementType2.tid = Resistance1.rid

This table shows the resistance of every type. It's big so I'll show a small part:

	name text	resists text
1	Steel	Normal
2	Rock	Normal
3	Dragon	Fire
4	Rock	Fire
5	Water	Fire
6	Fire	Fire
7	Dragon	Water
8	Grass	Water
9	Water	Water
10	Dragon	Electric
11	Electric	Electric
12	Steel	Grass
13	Dragon	Grass
14	Bug	Grass
15	Flying	Grass
16	Poison	Grass
17	Grass	Grass
18	Fire	Grass
19	Steel	Ice
20	Ice	Ice
21	Water	Ice
22	Fire	Ice
23	Fairy	Fighting
24	Bug	Fighting

IMMUNITY

CREATE VIEW Immunity

AS

SELECT ElementType.Name, COALESCE(ElementType2.name,'None') as Immune

from ElementType

left join Immunity as Immunity1

on ElementType.tid = Immunity1.tid

left join ElementType as ElementType2

on ElementType2.tid = Immunity1.iid

This table shows all of the immunities. Types that aren't immune to anything show up as None.

	name text	immune text
1	Normal	Ghost
2	Ground	Electric
3	Flying	Ground
4	Ghost	Normal
5	Ghost	Fighting
6	Dark	Psychic
7	Steel	Poison
8	Fairy	Dragon
9	Psychic	None
10	Bug	None
11	Fire	None
12	Dragon	None
13	Rock	None
14	Grass	None
15	Poison	None
16	Ice	None
17	Electric	None
18	Water	None
19	Fighting	None

SECURITY

The following pieces of code implement standard security for the database. An admin will be allowed to look, insert, update and delete data. The public can run queries on the data.

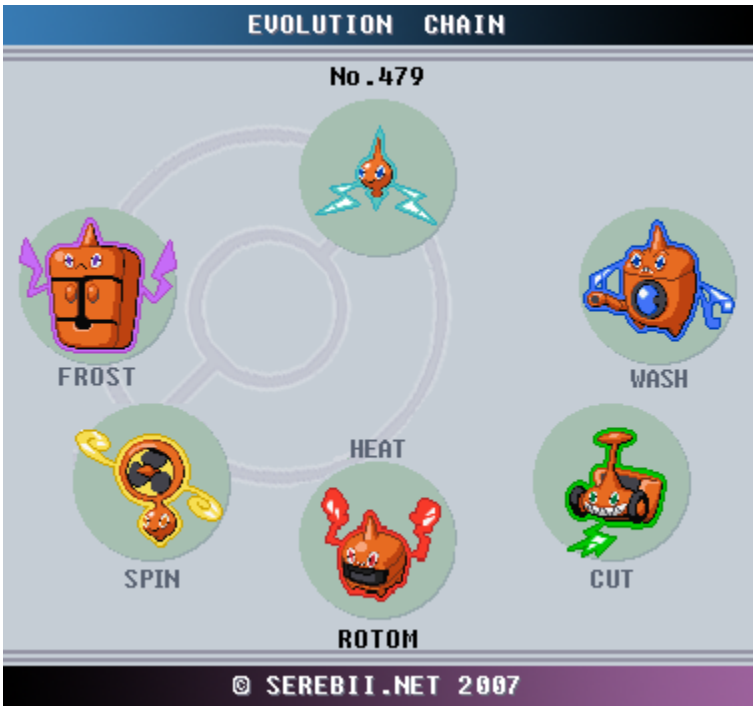
```
CREATE ROLE admin;

GRANT SELECT, INSERT, UPDATE

ON ALL TABLES IN SCHEMA PUBLIC

TO admin


CREATE ROLE Users;

GRANT SELECT

ON ALL TABLES IN SCHEMA PUBLIC

TO Users
```

IMPLEMENTATION

Implementing the data took far longer than I had originally thought. There is a lot of data that needs to be sorted through; you also want to make sure that you don't get any stat wrong so there was a lot of double guessing myself throughout the process. Thankfully it was on a subject that I enjoy, so it was not as mentally taxing as it would seem.

KNOWN PROBLEMS

I don't have any stored procedures. It is also troublesome that I won't be able to make sure that the base stats for every Pokémon is 100% correct when it is put in. Since each number is so specific I need to make sure that I am extra careful when inputting new entries. There is also another major issue. There are multiple Pokémon with multiple forms that still share the same Pokedex number. For example Rotom has six different forms that all share the same Pokedex number:



FUTURE ENHANCEMENTS

I would love to add all 718 Pokémon to this database. I chose a small random assortment of Pokémon due to how impossible it would be to add all the other Pokémon with such a small time frame. I also want to add moves that each Pokémon learns when they gain a level, as well as the

egg groups that each Pokémon belongs to. I would also like to find a way to add typing to each Pokémon's description. Although this could be easily done with a join I still need to take into account that Pokémon can be multiple types, causing it to show up twice during the output. I would also like to have the Pokémon's weaknesses, resistances and immunities show up properly. Trying this with joins caused a lot of incorrect data to be outputted. For example, Mawile is a Steel Type and a Fairy Type. Fairy is weak to Poison. But Steel is immune. Outputting data would show Mawile as weak and immune to Poison which is clearly incorrect. I feel that what I have so far is a great start, and if I ever do decide to add the rest of the Pokémon that I am missing it would be an easy enough task given enough time.