



## **FACULTY OF ENGINEERING AND APPLIED SCIENCE**

**SOFE 3700U - Data Management Systems**

**Project Final Report**

**Group 8**

**Tuesday, November 28 2017**

### **MEMBERS**

<b>#</b>	<b>First Name</b>	<b>Last Name</b>	<b>Student ID</b>
<b>1</b>	Jacob	Morra	100395426
<b>2</b>	Kevin	Apuyan	100561117
<b>3</b>	Raynosan	Edmond	100547264
<b>4</b>			

## **Abstract**

In this document, the project progression and analysis of a statistical database will be represented. This database will be relevant to the popular mobile-gaming application known as Pokemon GO, with the intention of creating a means of comparative play to increase player activity. The report will go over the related works used to assist in the data and concepts for the database and the conceptual design process (including requirement design, and component interactions used to create a relational schema and entity-relationship diagram) in initializing the project. This will be followed by the actual implementation of this project, including screenshots. Ultimately this project serves as a demonstration of using categorical data in order to develop a database which can be further developed to include features such as statistical analysis.

## **Introduction**

PokemonGO is a free-to-play, augmented reality mobile game for Android and iOS. It adapts concepts from previous iterations in the Pokemon series -- these games (starting on the GameBoy Color in 1995) have evolved with each new release, but feature many of the same core concepts, including (but not limited to) the following: a massive and continually growing collection of monsters for players to capture in-game, a catch mechanic whereby players throw various types of Pokeballs at Pokemon to capture them, a gym system whereby players can battle other trainers, and a trading mechanic whereby players could swap one Pokemon for another over a network. The Pokemon franchise extends its influence to card games, video games, action figures, television shows, and movies.

In July of 2016, the Pokemon brand took this to the next level by introducing a mobile-gaming application which allowed users to move pokemon-collecting from inside the game, to the physical world: Pokemon GO. Because of a combination of Pokemon's consistent popularity among gamers, and the immersion of augmented reality, the release of this application was met with intensity. For the first few weeks of its release, there was a notably increased amount of people outside using this application, particularly noticeable when large masses of people will be grouped in a particular area where the game notifies users of an increased chance of catching a pokemon. Unfortunately within a few weeks, popularity began to decline. This was due to multiple reasons including server connection issues (due to the unexpected reception of the application among the public), lack of new content, and exploits being realized within the game which allowed users to bypass certain restrictions. Although the active user base has decreased significantly, the game is still being played by millions around the world.

This report will outline our proposed implementation to PokemonGO - an integrated database which will keep track of specific user statistics both of the individual player, and of the players available online. Because of the gradual decrease in popularity, there has been less interaction between players compared

to when Pokemon GO first launched. This implementation aims to gradually restore this interaction by developing a means to display user statistics in comparison to others. It is hoped that this will motivate existing players and bring in new users due to the added competitive aspect with statistical comparison.

This report is organized as follows: First, related works are discussed in order to put our implementation into context with respect to existing Pokemon GO databases. Next, the conceptual design of the proposed implementation is discussed -- this section provides a visualization of the diagrams used in modelling the database developed. Following this, the implementation of the database is shown and discussed and includes screenshots of the actual database. Finally, concluding statements are made followed by references.

## Related Works

There are several Pokemon GO databases developed by both independent, and private developers since its release. The nature of Pokemon® is that there is an abundance of categorical information available. There are the type-combinations of pokemon, various moves for each pokemon, the moves themselves which have particular type-combinations, the possible evolutions of each pokemon, categorizing which types are strong against other types, and so on. Such a wealth of information allows for multiple types of databases to be developed, for multiple purposes. These purposes include: performing calculations, visualization of data, developing strategies based on queried data, and statistical presentation, to name a few. The implementation that we propose is the statistical implementation of user data and game data in order to allow for user comparison.

In order to develop this project, a few related works have been looked over in order to provide some referential concepts and development directions. The two works used were the databases found in: Pokemongodb.net[1], and Pokemongo.gamepress.gg[2].

Pokemongodb.net[1] serves as a statistical database containing information regarding the available pokemon in Pokemon GO. It includes categorical data for information such as the types of pokemon, pokemon eggs, moves, types of moves, views which represent the best pokemon based on particular criteria, as well as general guides included by the developer.

Pokemongo.gamepress.gg[2] expands upon the framework of pokemongodb.net[1]. It is at its core, a statistical database, however it has tabs which include much more in-depth information. There are tabs which perform calculations based on pokemon information, strategy-development tabs, tabs which include several guides for Pokemon GO, rankings, as well as the statistical data. This is an example of the

extent at which independent database developers can expand upon the available information from PokemonGO.

These are two of several databases developed by both independent, and private developers. Because of the nature of Pokemon GO, there exists several opportunities of database development with the variety of categorical information available.

## Conceptual Design

After the initial proposal for the PokemonGO comparative statistics database, we were to begin the design of the overall implementation. To do this, the requirements of the design were analyzed, followed up by the way in which the inner components would interact with themselves. After this general design was developed, the relational schema and entity-relationship diagram was made. These diagrams would also undergo changes as development progressed and requirements were updated.

The fundamental requirement of this database would be the trainer's account. This information would identify the user, and all other user-related data would be drawn from this. Initially, this would simply include the trainer's username, and the trainer's email in order to identify the user. General statistical data that is tied with the user would also be included in this in order to be used for the statistics entity. This includes the number of battles, number of pokemon, the user's level, and the distance walked.

Upon further research, it was realized that the login information and the trainer's account would have to be separate classes, due to the way Pokemon GO organized their data. Thus, a separate class was made for the login (which would include the username and the user's email), and a separate class for the trainer's account relevant to this login information.

Following this would be the classes for the pokemon-related data. There would be a class for Pokemon to start with. This would include the pokemon's name, the user, particular statistics relevant to the pokemon (such as HP, CP, IV, attack, defense, speed, etc.). There would be a class which contains the movesets for each pokemon, which would be identified with a particular pokemon. This class would include information such as the move type, its advantages & disadvantages, and so on. There would be a class for the move type, which provides more specific information about the moves including the actual damage, the DPS, the speed, the type, and so on. A class was made for possible evolutions of each pokemon, as

this itself was a form of categorical data relevant for most pokemon. This would include the name of the relevant pokemon, its evolution, and the in-game candies required for said evolution.

Finally, a statistics class was required for the purpose of this implementation. This class would contain all the comparative statistics of each user such as the trainer's level, distance walked, pokemon caught, total experience, and so on. Additional statistical categories were added as their relevance was realized.

After the requirements for the design were realized, the way in which the entities would interact with one another were analyzed. It was clear that the *login* would provide the trainer's user information to the *trainer account*. The *trainer account* would interact with both the *pokemon* and *statistics*. The trainer account would include the pokemon which the trainer has. The statistics is also simply a more specific representation of the trainer's information, so the trainer account would need to relay its data directly. *Pokemon* would relate with the *evolution* and *moveset*, as each pokemon would need to be identified with a particular evolution as well as its relevant moves. The *moveset* would interact with the *move type*, as the move type is a more specific mode of the moveset.

From here, the development of the relational schema and the entity-relationship diagram were initiated. Over the course of this, some additional changes were made to the requirements based on the information that was come across. The *login* information would have to include multiple layers of usernames because of the way PokemonGO has set up the login information for each user; it was later realized that this was implemented as a form of security for each user account. Initially statistics and *trainer\_account* were in one entity, but they were separated to remove complexity and redundancy between the two. The inclusion of battles won and battles lost was put into the statistics in order to add to comparative play, as well as the number of unique pokemon.



Changes in the ER diagram include updates to which entity was weak or strong, the relation-types between entities, the inclusion of composite attributes to remove redundancy, and the adjustment of primary keys dependent on the updates to how the components interacted with one another.

The following diagrams represents the final results of the relational schema and ER diagram.

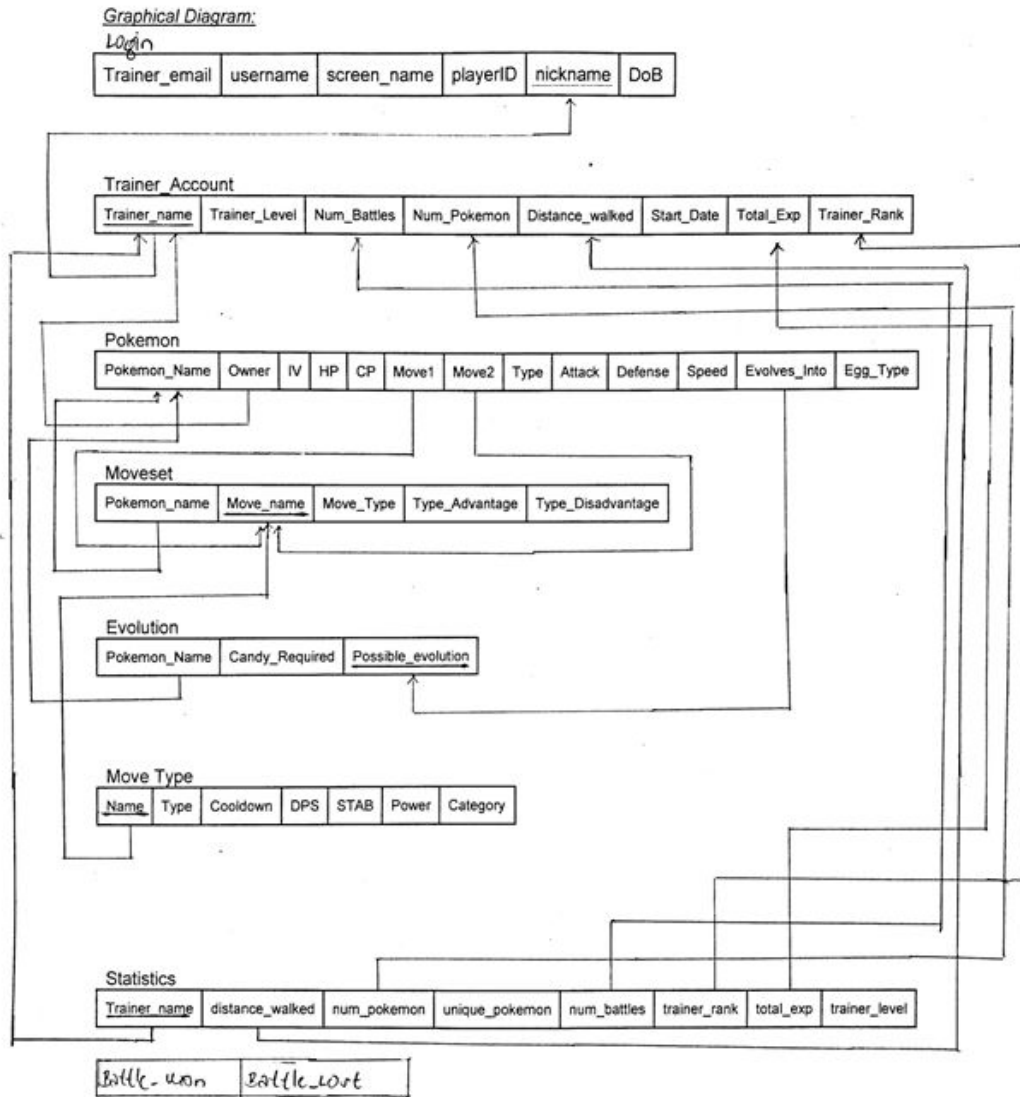


Figure 1: Relational Schema for PokemonGO statistics database

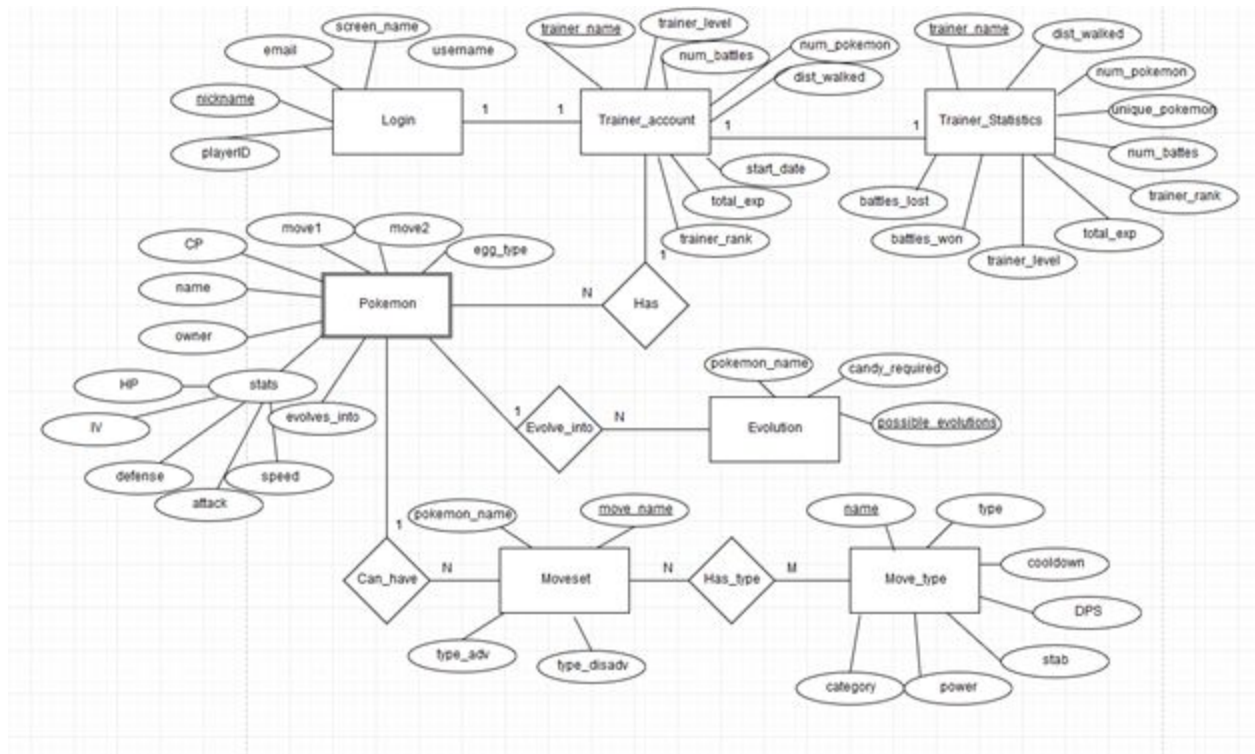



Figure 2: Entity-relationship diagram for Pokemon GO statistics database

## Implementation

**View 1: Select user with screenname 'username="5jakefire"'**

```
SELECT * FROM login WHERE nickname='5jakefire'
```




# PokemonGODatabase.net

Select user with screenname 'username="5jakefire"'

trainer_email	nickname	DoB
jakebombing@gmail.com	5jakefire	06/22/1991

**View 2: Find all trainers with more than 2 pokemon**

```
SELECT * FROM trainer_account WHERE num_Pokemon>2
```



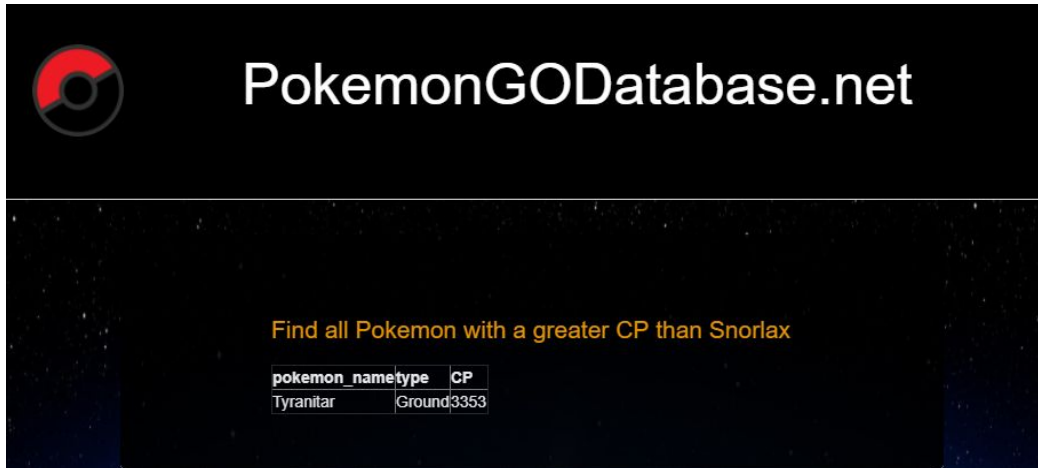
# PokemonGODatabase.net

Find all trainers with more than 2 pokemon'

trainer_name	trainer_level	num_battles	distance_walked	start_date	total_exp	trainer_rank	num_Pokemon
5jakefire	35	3249	3493	2016-06	5990023	23443	3
5jennifire	35	3550	3555	2016-06	6000000	23000	3
Nideak	40	523434	545345	2016-05	993420934203	221	5

### View 3: Find all Pokemon with a greater CP than Snorlax, group by type

```
SELECT pokemon_name, type, cp
FROM pokemon
WHERE cp > ANY ( SELECT cp FROM pokemon WHERE pokemon_name = 'Snorlax')
GROUP BY type
```

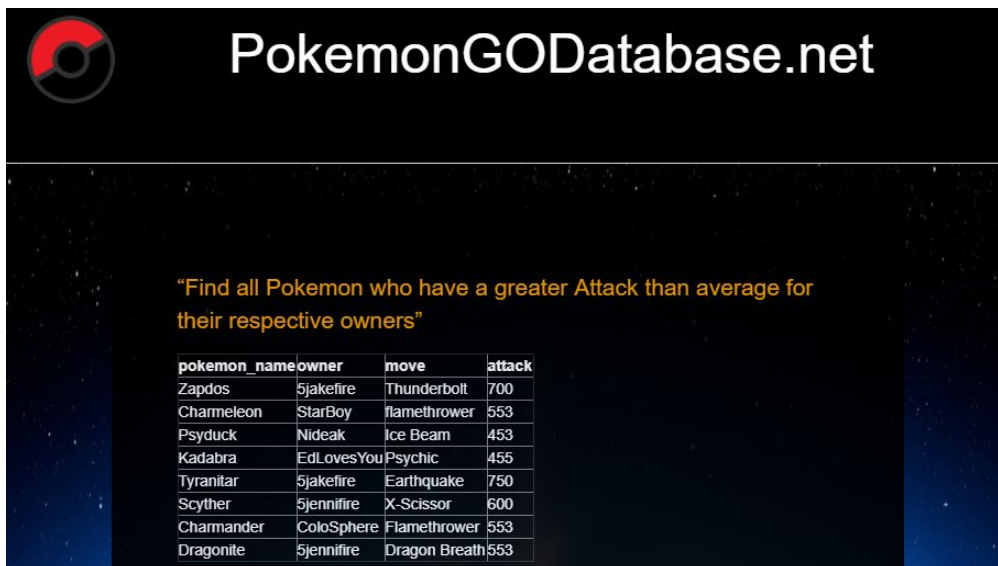


The screenshot shows the PokemonGODatabase.net website with a dark background. The title "PokemonGODatabase.net" is at the top. Below it, the query "Find all Pokemon with a greater CP than Snorlax" is displayed in orange text. A table shows the results of the query.

pokemon_name	type	CP
Tyranitar	Ground	3353

### View 4: Find all Pokemon who have a greater Attack than average for their respective owners

```
SELECT pokemon_name, owner, move2, attack
FROM Pokemon as P WHERE Attack > ( SELECT AVG(Attack)
                                     FROM Pokemon as P
                                     WHERE P.Owner = Owner)
```

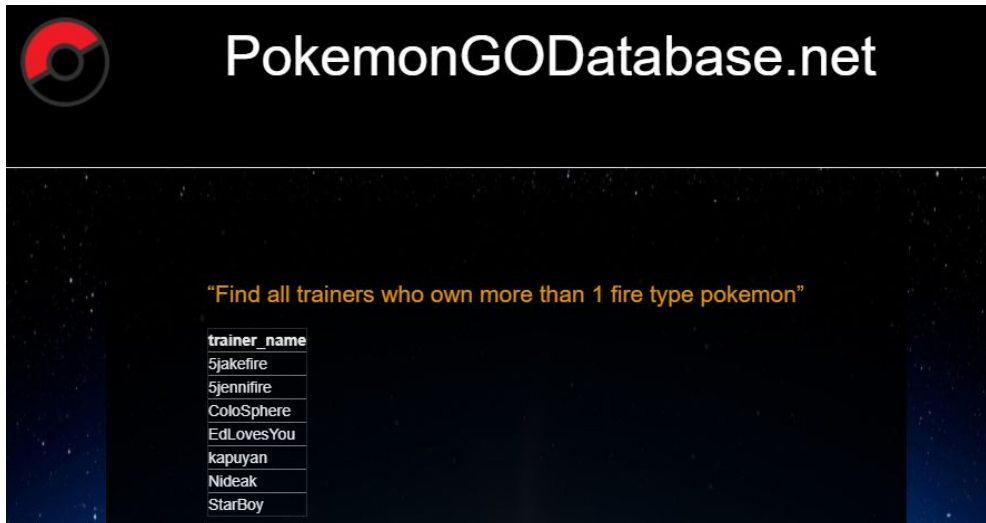


The screenshot shows the PokemonGODatabase.net website with a dark background. The title "PokemonGODatabase.net" is at the top. Below it, the query "Find all Pokemon who have a greater Attack than average for their respective owners" is displayed in orange text. A table shows the results of the query.

pokemon_name	owner	move	attack
Zapdos	5jakelire	Thunderbolt	700
Charmeleon	StarBoy	flamethrower	553
Psyduck	Nideak	Ice Beam	453
Kadabra	EdLovesYou	Psychic	455
Tyranitar	5jakelire	Earthquake	750
Scyther	5jennifire	X-Scissor	600
Charmander	ColoSphere	Flamethrower	553
Dragonite	5jennifire	Dragon Breath	553

### View 5: Find all trainers who own more than 1 fire type pokemon

```
SELECT DISTINCT trainer_name
FROM trainer_account, pokemon
WHERE (SELECT COUNT(*)
      FROM Trainer_Account as T, Pokemon as P
      WHERE T.Trainer_Name = P.Owner AND P.Type = 'Fire') > 1
```

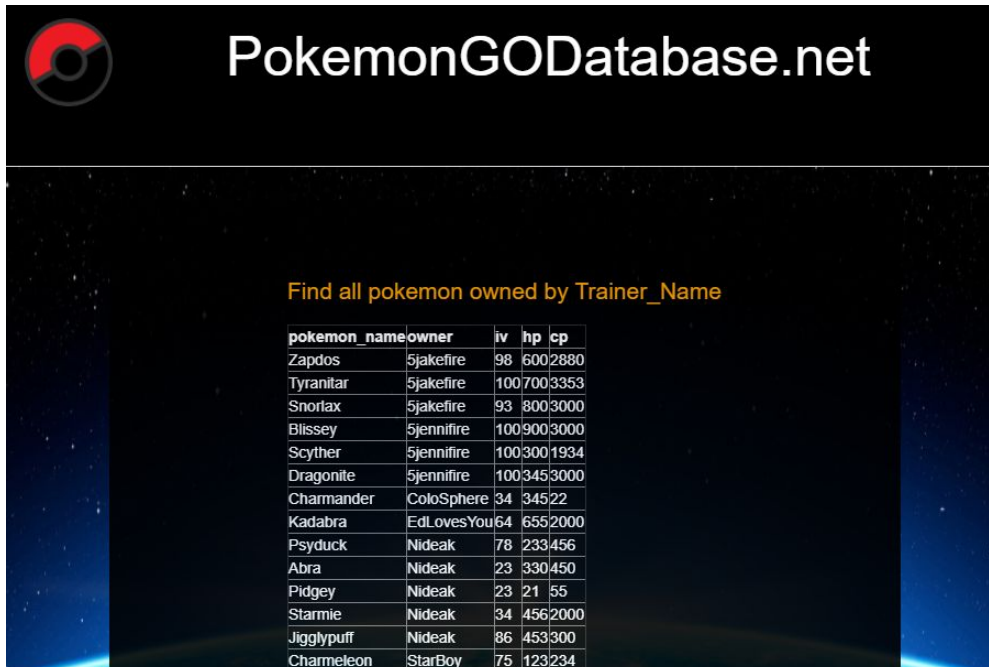


The screenshot shows the PokemonGODatabase.net website with a dark background and a starry sky. The website logo is in the top left corner. The main content area displays the query results for the query: "Find all trainers who own more than 1 fire type pokemon". The results are shown in a table with the following data:

trainer_name
5jakefire
5jennifire
ColoSphere
EdLovesYou
kapuyan
Nideak
StarBoy

### View 6: Find all pokemon owned by Trainer\_Name

```
SELECT pokemon_name, owner, iv, hp, cp
FROM pokemon
FULL JOIN trainer_account ON owner = trainer_account.trainer_name
```

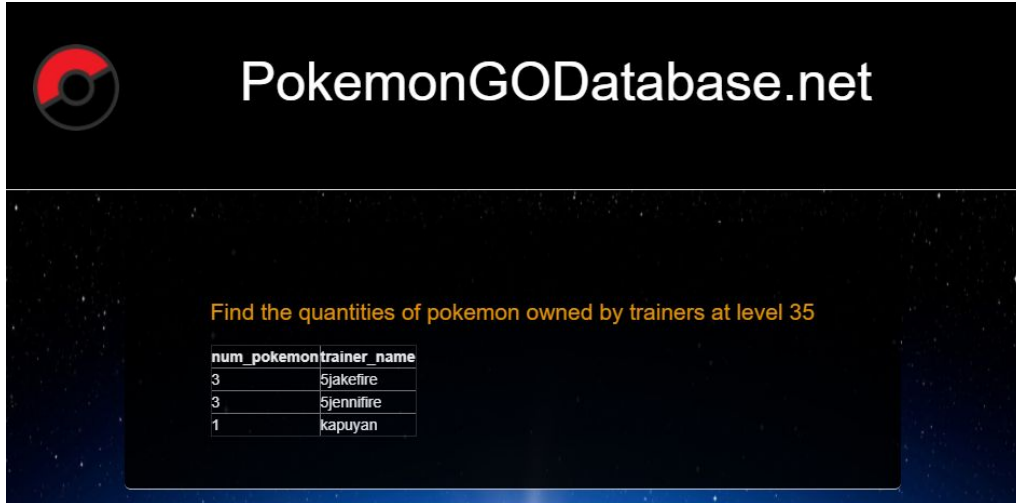


The screenshot shows the PokemonGODatabase.net website with a dark background and a starry sky. The website logo is in the top left corner. The main content area displays the query results for the query: "Find all pokemon owned by Trainer\_Name". The results are shown in a table with the following data:

pokemon_name	owner	iv	hp	cp
Zapdos	5jakefire	98	600	2880
Tyranitar	5jakefire	100	700	3353
Snorlax	5jakefire	93	800	3000
Blissey	5jennifire	100	900	3000
Scyther	5jennifire	100	300	1934
Dragonite	5jennifire	100	345	3000
Charmander	ColoSphere	34	345	22
Kadabra	EdLovesYou	64	655	2000
Psyduck	Nideak	78	233	456
Abra	Nideak	23	330	450
Pidgey	Nideak	23	21	55
Starmie	Nideak	34	456	2000
Jigglypuff	Nideak	86	453	300
Charmeleon	StarBoy	75	123	234

**View 7: “Find the quantities of pokemon for trainers at level 35”**

```
SELECT num_pokemon, trainer_name
FROM trainer_account
WHERE trainer_level = 35
```

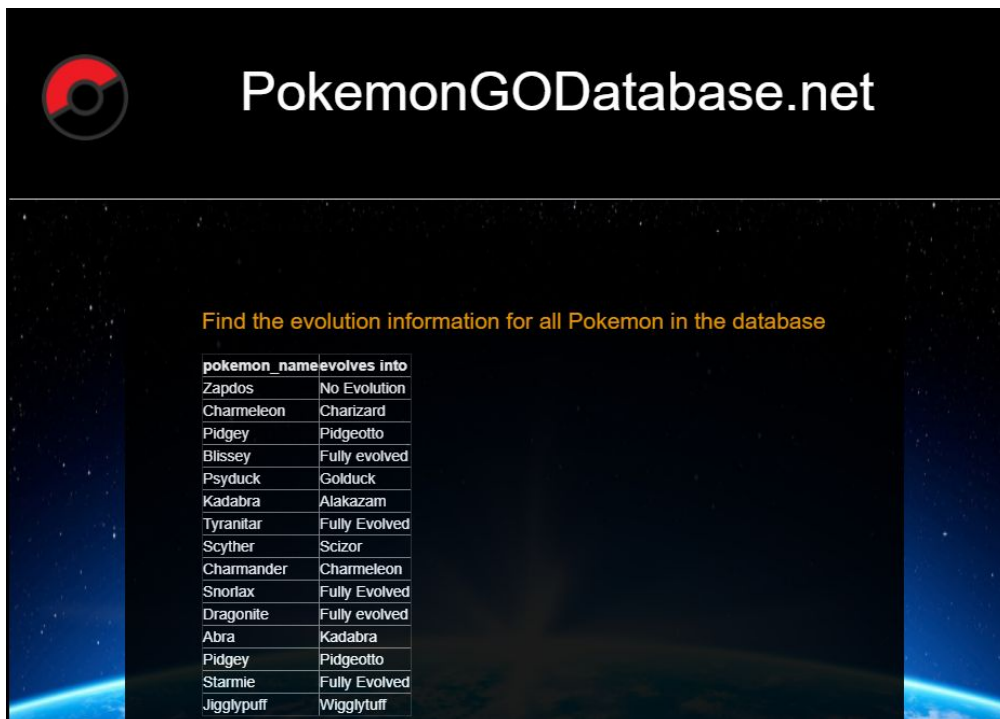


The screenshot shows the PokemonGO Database website with a dark background and a starry space theme. The website logo is in the top left. The main heading is "PokemonGO Database.net". Below it, a query is displayed: "Find the quantities of pokemon owned by trainers at level 35". The result is a table with two columns: "num\_pokemon" and "trainer\_name".

num_pokemon	trainer_name
3	5jakefire
3	5jennifire
1	kapuyan

**View 8: Find the evolution information for all Pokemon in the database**

```
SELECT pokemon.pokemon_name, evolves
FROM pokemon, evolution
WHERE pokemon.pokemon_name = evolution.pokemon_name AND pokemon.evolves =
evolution.possible_evolution
```



The screenshot shows the PokemonGO Database website with a dark background and a starry space theme. The website logo is in the top left. The main heading is "PokemonGO Database.net". Below it, a query is displayed: "Find the evolution information for all Pokemon in the database". The result is a table with two columns: "pokemon\_name" and "evolves into".

pokemon_name	evolves into
Zapdos	No Evolution
Charmeleon	Charizard
Pidgey	Pidgeotto
Blissey	Fully evolved
Psyduck	Golduck
Kadabra	Alakazam
Tyranitar	Fully Evolved
Scyther	Scizor
Charmander	Charmeleon
Snorlax	Fully Evolved
Dragonite	Fully evolved
Abra	Kadabra
Pidgey	Pidgeotto
Starmie	Fully Evolved
Jigglypuff	Wigglytuff

### View 9: Find the evolution information for all Pokemon in the database

```
SELECT pokemon.pokemon_name, pokemon.IV, pokemon.HP, pokemon.CP,  
trainer_account.trainer_name, trainer_account.trainer_level, moveset.move_name  
FROM (trainer_account  
INNER JOIN pokemon  
ON trainer_account.trainer_name = pokemon.owner)  
INNER JOIN moveset  
ON pokemon.pokemon_name = moveset.pokemon_name
```

### Find Pokemon name, IV, HP, CP, owner, owner level, and moves for all Pokemon

pokemon_name	IV	HP	CP	trainer_name	trainer_level	move_name
Zapdos	98	600	2880	5jakefire	35	Thunderbolt
Charmeleon	75	123	234	StarBoy	31	Fire Punch
Blissey	100	900	3000	5jennifire	35	Hyper Beam
Psyduck	78	233	456	Nideak	40	Ice Beam
Kadabra	64	655	2000	EdLovesYou	37	Psychic
Tyranitar	100	700	3353	5jakefire	35	Earthquake
Scyther	100	300	1934	5jennifire	35	X-Scissor
Charmander	34	345	22	ColoSphere	20	Flamethrower
Snorlax	93	800	3000	5jakefire	35	Hyper Beam
Dragonite	100	345	3000	5jennifire	35	Dragon Breath
Abra	23	330	450	Nideak	40	Psychic
Pidgey	23	21	55	Nideak	40	Aerial Ace
Pidgey	23	21	55	Nideak	40	Wing Attack
Starmie	34	456	2000	Nideak	40	Hydro Pump
Jigglypuff	86	453	300	Nideak	40	Puff Blast

### View 10: UNION of 2 queries

```
SELECT trainer_name  
FROM trainer_account  
WHERE num_battles > (SELECT num_battles  
FROM trainer_account  
WHERE trainer_name = 'ColoSphere')  
  
UNION  
SELECT trainer_name  
FROM trainer_account  
WHERE num_battles < (SELECT num_battles  
FROM trainer_account  
WHERE trainer_name = '5jennifire')
```



Find all Pokemon trainers with more battles won than ColoSphere or less battles played than 5jennifire

trainer_name
5jakefire
5jennifire
EdLovesYou
Nideak
StarBoy
ColoSphere

## Web Service Implementation -- SOAP/XML

Our web service implementation retrieves tabular data from the login table and outputs to an XML format. It automatically updates as the database is updated with player information. Please see dbtoxml.php for our implementation code. Below is a current view of the XML output for our database:

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<posts>
  <post>
    <trainer_name>5jakefire</trainer_name>
    <trainer_level>35</trainer_level>
    <num_battles>3249</num_battles>
    <distance_walked>3493</distance_walked>
    <start_date>2016-06</start_date>
    <total_exp>5990023</total_exp>
    <trainer_rank>23443</trainer_rank>
    <num_Pokemon>3</num_Pokemon>
  </post>
  <post>
    <trainer_name>5jennifire</trainer_name>
    <trainer_level>35</trainer_level>
    <num_battles>3550</num_battles>
    <distance_walked>3555</distance_walked>
    <start_date>2016-06</start_date>
    <total_exp>6000000</total_exp>
    <trainer_rank>23000</trainer_rank>
    <num_Pokemon>3</num_Pokemon>
  </post>
  <post>
    <trainer_name>ColoSphere</trainer_name>
    <trainer_level>20</trainer_level>
    <num_battles>332</num_battles>
    <distance_walked>332</distance_walked>
    <start_date>2017-07</start_date>
    <total_exp>33442</total_exp>
    <trainer_rank>32342345</trainer_rank>
    <num_Pokemon>1</num_Pokemon>
  </post>
  <post>
    <trainer_name>EdLovesYou</trainer_name>
    <trainer_level>37</trainer_level>
    <num_battles>34533</num_battles>
    <distance_walked>8573</distance_walked>
    <start_date>2016-06</start_date>
    <total_exp>7349234</total_exp>
    <trainer_rank>15034</trainer_rank>
    <num_Pokemon>2</num_Pokemon>
  </post>
  <post>
    <trainer_name>kapuyan</trainer_name>
    <trainer_level>35</trainer_level>
    <num_battles>242</num_battles>
    <distance_walked>113</distance_walked>
    <start_date>2016-07</start_date>
    <total_exp>500000</total_exp>
    <trainer_rank>30000</trainer_rank>
    <num_Pokemon>1</num_Pokemon>
  </post>
</posts>
```



## Conclusion

This database implementation has served as an attempt to increase the popularity of Pokemon GO by allowing for players to compare their in-game statistics with others. This has been done through the implementation of a comprehensive set of user statistics tables.

In the design phase of this project, the relevant schemas were created and ultimately converted into our implementation. We first went over the requirements for the proposed project. This was followed by further research, updating the requirements, and initializing the relationships between the inner components of the system. From here, a relational schema and entity-relationship diagram were developed, with further updates which allowed for a smooth flow of data within the system. Our implementation included the use of PHP, Bootstrap, MySQL via phpmyadmin, HTML/CSS, and XML (for our web service). We found that these technologies were sufficient for our needs, but potentially we would have benefitted from a better survey of all available technologies.

The content of this project is directly related to this course, as it is one which consists of layers of categorical data. This allows for the abundance in information in order to provide a sufficient implementation. As well, there are several related works available in which to provide reference and guidance as the work progresses, such as to put the knowledge learned from the course into practical use. The processes learned from this project can be used to develop databases for other categorical information, which is a constant need in today's industry of information technology.

Should this project be continued, future implementations would benefit from some of the following features: implementation of user inputs via buttons or drop-down boxes, such that a user could create custom table queries without requiring and technical knowledge; use of sessions, cookies, and user

login/logout to open up possibilities for more features; addition of more site features such as a forum, a tips page, and an additional database for all existing pokemon and their information.

## **References**

[1] "Pokémon Go Database", Pokemongodb.net , 2017. [Online]. Available: <http://www.pokemongodb.net/>. [Accessed: 10- Oct- 2017].

[2] "Pokemon GO GamePress", Pokemon GO GamePress , 2017. [Online]. Available: <https://pokemongo.gamepress.gg/>. [Accessed: 10- Oct- 2017].