**University of British Columbia, Vancouver**
Department of Computer Science

---

# CPSC 304 Project Cover Page

Milestone #: 4

Date: Nov 29, 2024

Group Number: 98

| Name | Student Number | CS Alias (Userid) | Preferred E-mail Address |
|---|---|---|---|
| Matthew Wan | 80099211 | k2m5q | mattwan1@student.ubc.ca |
| Joshua Wu | 61994414 | x3e0l | joshuawu2004@gmail.com |
| Albert Chang | 26234147 | b4g1b | alb12345@student.ubc.ca |

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Repository Link: https://github.students.cs.ubc.ca/CPSC304-2024W-T1/project_b4g1b_k2m5q_x3e0l

The SQL script is available from our project repository at: scripts/sql/insert_tables.sql

This script includes all the information and data needed to reload our tables.

# 2. Project Description:

Summary:

Our project is a centralized hub of information that allows users to access and contribute detailed information about the Pokémon video game series. Information about the game's Pokémon, trainers, gym leaders, moves, and more are to be tracked so users are able to develop strategies and keep track of the vast amount of information available in the Pokemon series. Users should be able to find information such as which trainers possess which Pokémon, how many Pokémon are in which region, or what the strongest moves are.

Schema Changes:

The only change that was done to our schema was having all the Pokemon traits like ID, Name, Evolution Items, etc. placed into one large table instead of breaking it down into smaller tables to normalize for BCNF. The reason we did this was because if we left each trait as its own individual table, for certain queries, we would have to make many joins to access all the information for Pokemon. We figured that having all the traits in one table would make the queries much more readable easy to maintain throughout the project. We also did not expect many null values in our project, making it more likely that we will be more space efficient as well. After discussing with our TA, we determined that this made the most sense.

# 3. Query Breakdown:

(2.1.1) One relation to insert on

app/backend/routes/albert_routes.js – Line: 27, 35, 44,

(2.1.2) Implement UPDATE on one relation

app/backend/routes/albert_routes.js – Line: 98, 107, 115, 141

(2.1.3) Implement DELETE on one relation

app/backend/routes/albert_routes.js – Line: 161

(2.1.4) Implement Selection on one relation

app/backend/routes/albert_routes.js – Line: 184

(2.1.5) One relation to project on

app/backend/routes/matt_routes.js – Line:: 9

(2.1.6) User should be able to see the gym leader's name, where they are located

app/backend/routes/josh_routes.js – Line: 9

(2.1.7) Aggregation with group by

Users should be able to count the number of Pokemon by region to be able to figure out how many new Pokemon are available to discover in a given region.

app/backend/routes/matt_routes.js – Line: 22

```
SELECT Pokemon.region_name AS region, COUNT(Pokemon.id) AS num_pokemon
FROM Pokemon
GROUP BY Pokemon.region_name;
```

(2.1.8) Aggregation with having

Users are able to find the min/max/avg power of moves with a specific type.

app/backend/routes/josh_routes.js – Line: 33

```
SELECT
move_has_type.type_name,
MIN(move.power),
MAX(move.power),
AVG(move.power)
FROM
move
JOIN move_has_type ON move.move_name = move_has_type.move_name
GROUP BY move_has_type.type_name
HAVING move_has_type.type_name = $1;
```

(2.1.9) Nested aggregation with group by

Users should be able to find the Pokemon type with the highest and lowest average power of their moves.

app/backend/routes/matt_routes.js – Line: 39

```
SELECT type, MAX(power)
FROM (
SELECT MT.type_name AS type, AVG(m.power) AS power
FROM Move M INNER JOIN move_has_type MT
ON M.move_name = MT.move_name
GROUP BY MT.type_name
) AS avg_power
ORDER BY power ${ord}
LIMIT 1;
```

(2.1.10) Division

Users should be able to find if a given list of Pokemon are used in a trainer's roster. For example you want to find out if a trainer you will need to face will have some Pokemon that will counter yours.

app/backend/routes/josh_routes.js – Line: 69

```
SELECT name
FROM significant_trainer
WHERE NOT EXISTS (
SELECT pokemon_name
FROM pokemon
WHERE pokemon_name IN (${placeHolders})
EXCEPT
SELECT pokemon.pokemon_name
FROM pokemon_roster
JOIN pokemon ON pokemon_roster.id = pokemon.id
WHERE pokemon_roster.name = significant_trainer.name
);
```