Project Description:

The main focus of our project is to describe and showcase information on video game characters called Pokemon and Trainers that can own Pokemon. This is done through tables containing attributes such as in-game statistics, types, moves, abilities, and evolution chains of Pokemon. Trainers are showcased through collections and inventory. Collections contain Pokemon owned by the trainer and inventory contains items owned by the Trainer. Additionally, information about the regions and routes where these pokemon and trainers can be found is also shown. The project successfully showcased our ability to turn an idea for database relation into an ER diagram, followed by a project schema on which data normalization was done to get SQL DDL statements and finally turned into a fully functional relational database system.

Our schema did not change from previous milestones. Schema diagram for our project is shown below:
Pokemon1(**to_pokedex_id**, pokemon_name, *hp*, *attack*, *defence*, *special_attack*, *special_defence*, *speed*, *from_pokedex_id*, *req_name*)
Pokemon2(**hp**, **attack**, **defence**, **special_attack**, **special_defence**, **speed**, total)
TrainerPokemon1(***pokedex_id***, *experience*, *leveling_group*, pet_name, height, weight, *collection_number*, *trainer_id*)
TrainerPokemon2(level, **experience**, **leveling_group**)
WildPokemon(***pokedex_id***, spawn_rate, spawn_weather, spawn_time)
EvolutionReq(**req_name**, method, threshold)
Type(weakness, resistance, **type_name**)
Move1(*move_effect*, move_scale, **move_name**)
Move2(move_category, **move_effect**)
Ability1(*ability_effect*, **ability_name**)
Ability2(**ability_effect**, ability_scale)
Trainer(trainer_name, rank, **trainer_id**, *region_name*)
Collection1(collection_name, *collection_category*, **collection_number**, ***trainer_id***)
Collection2(**collection_category**, collection_size)
Item(**item_name**, item_category, item_effect)
Region(**region_name**, climate, theme)
Route1(**route_name**, *terrain_type*)
Route2(difficulty_level, **terrain_type**)
hasType(***type_name***, ***pokedex_id***)
hasMove(***move_name***, ***pokedex_id***)
ableTo(***ability_name***, ***pokedex_id***)
hasItem(***item_name***, ***trainer_id***)
leadsTo(***region_name***, ***route_name***)
foundAt(***route_name***, ***pokedex_id***)

Further details for the schema can be accessed through pdfs for previous milestones located in project_u6m1z_z1r2b_z9s7o/info/prevMilestones. A separate pdf called PokemonSchema.pdf contains screenshots for each relation in a mock database instance.

SQL Queries and Location:

1. **Insert**
   Filename: project_u6m1z_z1r2b_z9s7o/appService.js
   Line number: 160, 172, 187, 201, 236, 260, 272, 287, 301, 315, 329
2. **Update**
   Filename: project_u6m1z_z1r2b_z9s7o/appService.js
   Line number: 358, 385, 434, 460, 510, 537, 564, 591, 618
3. **Delete**
   Filename: project_u6m1z_z1r2b_z9s7o/appService.js
   Line number: 631, 645, 659, 673, 687, 701, 715, 729
4. **Selection**
   Filename: project_u6m1z_z1r2b_z9s7o/appService.js
   Line number: 743, 776, 809
5. **Projection**
   Filename: project_u6m1z_z1r2b_z9s7o/appService.js
   Line number: 858, 883, 908, 920, 930, 940, 950, 960
6. **Join**
   Filename: project_u6m1z_z1r2b_z9s7o/appService.js
   Line number: 973, 996, 1021, 1041, 1063, 1089, 1109, 1129, 1149
7. **Aggregation with GROUP BY**
   Filename: project_u6m1z_z1r2b_z9s7o/appService.js
   Line number: 1173, 1196, 1214
   1173:

```
SELECT ht.type_name,
    AVG(p.hp) AS avg_hp,
    AVG(p.attack) AS avg_attack,
    AVG(p.defence) AS avg_defense,
    AVG(p.special_attack) AS avg_special_attack,
    AVG(p.special_defence) AS avg_special_defense,
    AVG(p.speed) AS avg_speed,
    AVG(p.total) AS avg_total,
    COUNT(*) AS num_pokemon
FROM hasType ht
JOIN Pokemon p ON ht.pokedex_id = p.pokedex_id
GROUP BY ht.type_name
```

Description: this query is used to get average stats(hp, attack, defence, special attack, special defence, speed and total) of pokemon for each type

1196:
SELECT r.region_name, COUNT(f.pokedex_id) AS num_pokemon
FROM foundAt f
JOIN Route1 ro ON f.route_name = ro.route_name
JOIN leadsTo l ON ro.route_name = l.route_name
JOIN Region r ON l.region_name = r.region_name
GROUP BY r.region_name
ORDER BY num_pokemon DESC
Description: this query is used to get the count of pokemon in each region and groups the result by region name in order starting with regions with higher pokemon count

1214:
SELECT region_name, COUNT(trainer_id) AS trainer_count
FROM Trainer
GROUP BY region_name
ORDER BY trainer_count DESC
Description: this query gets the number of trainers in each region and groups the result by region name in order starting with regions with higher number of trainers

8. **Aggregation with HAVING**
   Filename: project_u6m1z_z1r2b_z9s7o/appService.js
   Line number: 1231, 1246
   1231:
   SELECT pokedex_id, COUNT(route_name) AS num_routes
   FROM foundAt
   GROUP BY pokedex_id
   HAVING COUNT(route_name) > 1
   Description: this query gets wild pokemon that are found in more than one route

   1246:
   SELECT type_name, COUNT(*) AS pokemon_count
   FROM Pokemon p
   JOIN hasType ht ON ht.pokedex_id = p.pokedex_id
   GROUP BY type_name
   HAVING COUNT(*) > 3
   Description: this query gets types that strictly have more than 3 pokemon

9. **Nested aggregation with GROUP BY**

   Filename: project_u6m1z_z1r2b_z9s7o/appService.js
   Line number: 1264, 1280
   1264:

```
SELECT ht.type_name, AVG(p.total) AS total_avg
FROM Pokemon p
JOIN hasType ht ON ht.pokedex_id = p.pokedex_id
GROUP BY ht.type_name
HAVING AVG(p.total) <= (SELECT AVG(p1.total) FROM Pokemon p1)
```

   Description: this query gets pokemon types in which the average stats of pokemon are less than or equal to average stats of all Pokemon

   1280:

```
SELECT R.region_name, COUNT(F.pokedex_id) AS spawn_count
FROM leadsTo L
JOIN foundAt F ON L.route_name = F.route_name
JOIN Region R ON L.region_name = R.region_name
GROUP BY R.region_name
HAVING COUNT(F.pokedex_id) > (
    SELECT AVG(spawn_count) FROM (
       SELECT COUNT(pokedex_id) AS spawn_count
       FROM foundAt
       GROUP BY route_name
    )
)
```

   Description: this query gets regions where the average count of pokemon species is higher than the average count of pokemon species across all routes

10. **Division**

    Filename: project_u6m1z_z1r2b_z9s7o/appService.js
    Line number: 1305, 1324, 1343, 1363
    1305:

```
SELECT T.trainer_id
FROM Trainer T
WHERE NOT EXISTS (
   SELECT DISTINCT L.leveling_group FROM TrainerPokemon2 L
   MINUS
```

```
    SELECT DISTINCT TP.leveling_group FROM TrainerPokemon1 TP
    WHERE TP.trainer_id = T.trainer_id
)
```
Description: this query gets trainers that have pokemon in every leveling group

1324:
```
SELECT T.trainer_id
FROM Trainer T
WHERE NOT EXISTS (
    SELECT DISTINCT C.collection_category FROM Collection2 C
    MINUS
    SELECT DISTINCT C2.collection_category FROM Collection C2
    WHERE C2.trainer_id = T.trainer_id
)
```
Description: this query gets trainers that have at least one pokemon from every collection category

1343:
```
SELECT T.trainer_id
FROM Trainer T
WHERE NOT EXISTS (
    SELECT DISTINCT I.item_category FROM Item I
    MINUS
    SELECT DISTINCT I2.item_category FROM hasItem HI
    JOIN Item I2 ON HI.item_name = I2.item_name
    WHERE HI.trainer_id = T.trainer_id
)
```
Description: this query gets trainers that have at least one item from every item category

1363:
```
SELECT P.to_pokedex_id
FROM Pokemon1 P
WHERE NOT EXISTS (
    SELECT DISTINCT M.move_category FROM Move2 M
    MINUS
    SELECT DISTINCT M2.move_category FROM hasMove HM
    JOIN Move1 M1 ON HM.move_name = M1.move_name
    JOIN Move2 M2 ON M1.move_effect = M2.move_effect
    WHERE HM.pokedex_id = P.to_pokedex_id
)
```

Description: this query gets pokemon that can learn at least one move from every move category