THOMAS MORE

TALHA ARIF

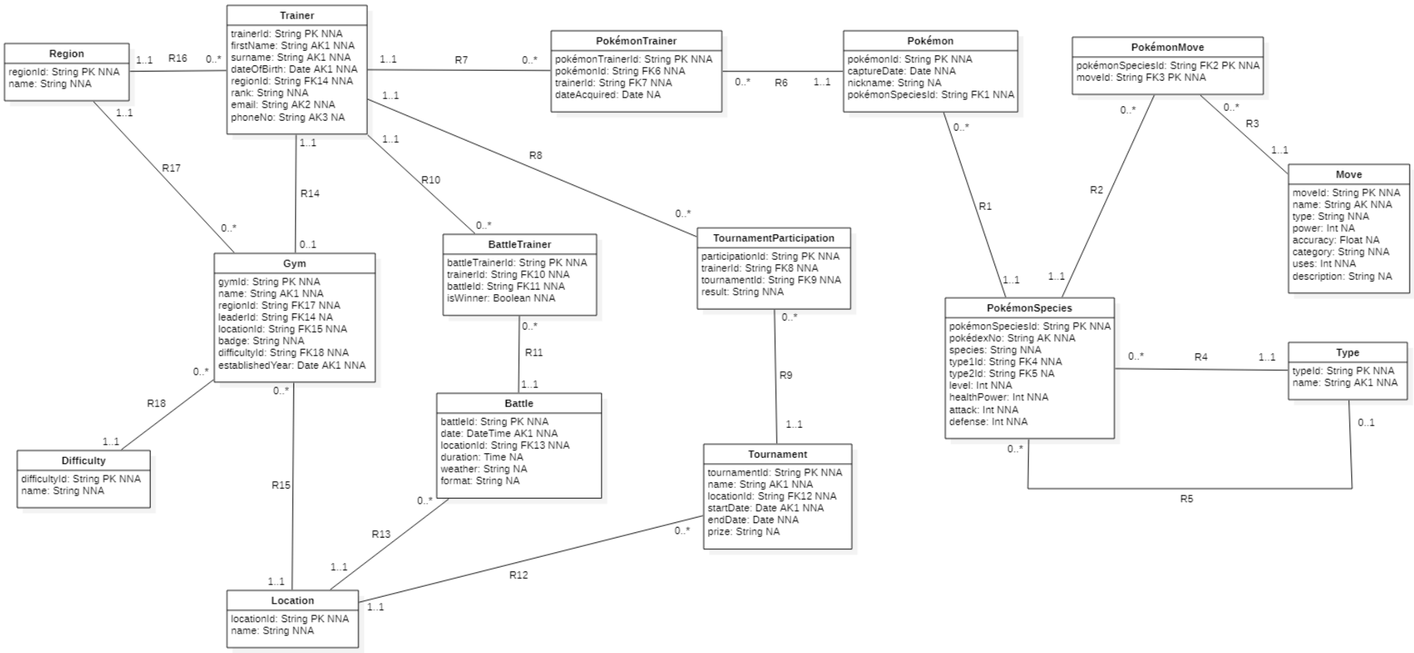
SQL Project

1ACS01

2025 - 2026

Inhoud

1. Assignment 1 – Data model



Core Entities:

1. Pokémon
2. PokémonSpecies
3. Move
4. Type
5. Trainer
6. Tournament
7. Battle
8. Gym
9. Location
10. Region
11. Difficulty

Association Entities:

1. PokémonMove (uses ‘*pokémonSpeciesId’* & ‘*moveId’* as **composite** primary keys)
2. PokémonTrainer (uses the **surrogate** key *‘pokémonTrainerId’* as primary key)
3. BattleTrainer (uses the **surrogate** key ‘*battleTrainerId*’ as primary key)
4. TournamentParticipation (uses the **surrogate** key: ‘*participationId*’ as primary key)

Relationships:

R1: One **PokémonSpecies** can be linked to zero or many **Pokémon**, but each Pokémon must belong to exactly one PokémonSpecies.
R2: One **PokémonSpecies** can appear in zero or many **PokémonMove** records (since multiple species can have the same moves), but each PokémonMove record must always link to exactly one PokémonSpecies.
R3: One **Move** can appear in zero or many **PokémonMove** records, but each PokémonMove record must always link to exactly one Move.
R4: One **Type** can be associated with zero or many **PokémonSpecies**, but each PokémonSpecies record must link to exactly one Type (type1).
R5: One **Type** can be associated (as a secondary type) with zero or many **PokémonSpecies**, but each PokémonSpecies may only link to zero or exactly one secondary Type (type2).

R6: One **Pokémon** can appear in zero or many **PokémonTrainer** records (due to ownership changes), but each PokémonTrainer record must link to exactly one Pokémon.
R7: One **Trainer** can own zero or many Pokémon, but each PokémonTrainer record must always link to exactly one Trainer.
R8: One **Trainer** can participate in zero or many **TournamentParticipation** records, but each TournamentParticipation record must always link to exactly one Trainer.
R9: One **Tournament** can have zero or many **TournamentParticipation** records, but each TournamentParticipation record must always link to exactly one Tournament.
R10: One **Trainer** can participate in zero or many **BattleTrainer** records, but each BattleTrainer record must always link to exactly one Trainer.
R11: One **Battle** can have zero or many **BattleTrainer** records, but each BattleTrainer record must always link to exactly one Battle.
R12: One **Location** can host zero or many **Tournaments**, but each Tournament must always be associated with exactly one Location.
R13: One **Location** can host zero or many **Battles**, but each Battle must always be associated with exactly one Location.
R14: One **Trainer** can lead zero or one **Gym**, but a Gym must always have exactly one Trainer as its leader.
R15: One **Location** can have zero or many **Gyms**, but each Gym must always link to exactly one Location.
R16: One **Region** can contain zero or many **Trainers**, but each Trainer must belong to exactly one Region.
R17: One **Region** can contain zero or many **Gyms**, but each Gym must belong to exactly one Region.
R18: One **Difficulty** level can apply to zero or many **Gyms**, but each Gym must have exactly one Difficulty level.

REFERENCES:

https://bulbapedia.bulbagarden.net/wiki/Type

2. Assignment 2 – Creating a database

```
DROP TABLE IF EXISTS type;
DROP TABLE IF EXISTS move;
DROP TABLE IF EXISTS region;
DROP TABLE IF EXISTS difficulty;
DROP TABLE IF EXISTS location;
DROP TABLE IF EXISTS pokémonSpecies;
DROP TABLE IF EXISTS pokémon;
DROP TABLE IF EXISTS pokémonMove;
DROP TABLE IF EXISTS trainer;
DROP TABLE IF EXISTS pokémonTrainer;
DROP TABLE IF EXISTS gym;
DROP TABLE IF EXISTS battle;
DROP TABLE IF EXISTS battleTrainer;
DROP TABLE IF EXISTS tournament;
DROP TABLE IF EXISTS tournamentParticipation;

DROP SCHEMA IF EXISTS Pokémon;
CREATE SCHEMA Pokémon;
USE Pokémon;

CREATE TABLE type (
     typeID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    name VARCHAR(25) NOT NULL
```

```
);

CREATE TABLE move (
    moveID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) UNIQUE NOT NULL,
    typeID INT NOT NULL,
    power INT,
    accuracy FLOAT,
    category VARCHAR(25) NOT NULL,
    uses INT NOT NULL,
    description VARCHAR(100),
    CONSTRAINT FK_move_type FOREIGN KEY (typeID) REFERENCES type(typeID)
);

CREATE TABLE region (
     regionID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    name VARCHAR(25) NOT NULL
);

CREATE TABLE difficulty (
     difficultyID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    name VARCHAR(25) NOT NULL
);

CREATE TABLE location (
     locationID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    name VARCHAR(25) NOT NULL
);

CREATE TABLE pokémonSpecies (
     pokémonSpeciesID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    pokédexNo INT UNIQUE NOT NULL,
    species VARCHAR(25) NOT NULL,
    type1ID INT NOT NULL,
    type2ID INT,
    level INT,
    healthPower INT NOT NULL,
    attack INT NOT NULL,
    defense INT NOT NULL,
    CONSTRAINT FK_pokémonSpecies_type1 FOREIGN KEY (type1ID) REFERENCES type(typeID),
    CONSTRAINT FK_pokémonSpecies_type2 FOREIGN KEY (type2ID) REFERENCES type(typeID)
);

CREATE TABLE pokémon (
     pokémonID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    captureDate DATE NOT NULL,
    nickname VARCHAR(25),
    pokémonSpeciesID INT NOT NULL,
    CONSTRAINT FK_pokémon_pokémonspecies FOREIGN KEY (pokémonSpeciesID) REFERENCES
pokémonSpecies(pokémonSpeciesID)
);

CREATE TABLE pokémonMove (
     pokémonSpeciesID INT NOT NULL,
    moveID INT NOT NULL,
    CONSTRAINT PK_pokémonMove PRIMARY KEY (pokémonSpeciesID, moveID),
    CONSTRAINT FK_pokémonMove_pokémonSpecies FOREIGN KEY (pokémonSpeciesID) REFERENCES
pokémonSpecies(pokémonSpeciesID),
    CONSTRAINT FK_pokémonMove_move FOREIGN KEY (moveID) REFERENCES move(moveID)
);

CREATE TABLE trainer (
     trainerID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    firstName VARCHAR(25) NOT NULL,
    surname VARCHAR(25) NOT NULL,
    dateOfBirth DATE NOT NULL,
    regionID INT NOT NULL,
     trainerRank VARCHAR(25) NOT NULL,
    email VARCHAR(50) NOT NULL UNIQUE,
    phoneNo VARCHAR(25) UNIQUE,
    CONSTRAINT AK_trainer UNIQUE (firstName, surname, dateOfBirth),
    CONSTRAINT FK_trainer_region FOREIGN KEY (regionID) REFERENCES region(regionID)
);

CREATE TABLE pokémonTrainer (
     pokémonTrainerID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    pokémonID INT NOT NULL,
    trainerID INT NOT NULL,
    dateAcquired DATE,
    CONSTRAINT FK_pokémonTrainer_pokémon FOREIGN KEY (pokémonID) REFERENCES
pokémon(pokémonID),
    CONSTRAINT FK_pokémonTrainer_trainer FOREIGN KEY (trainerID) REFERENCES trainer(trainerID)
);

CREATE TABLE gym (
     gymID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    name VARCHAR(25),
    regionID INT NOT NULL,
    leaderID INT,
    locationID INT NOT NULL,
    badge VARCHAR(25) NOT NULL,
    difficultyID INT NOT NULL,
    establishedYear DATE NOT NULL,
    CONSTRAINT AK_gym UNIQUE (name, establishedYear),
    CONSTRAINT FK_gym_region FOREIGN KEY (regionID) REFERENCES region(regionID),
    CONSTRAINT FK_gym_trainer FOREIGN KEY (leaderID) REFERENCES trainer(trainerID),
    CONSTRAINT FK_gym_location FOREIGN KEY (locationID) REFERENCES location(locationID),
    CONSTRAINT FK_gym_difficulty FOREIGN KEY (difficultyID) REFERENCES difficulty(difficultyID)
);

CREATE TABLE battle (
     battleID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    date DATETIME UNIQUE NOT NULL,
    locationID INT NOT NULL,
    duration TIME,
    weather VARCHAR(25),
    format VARCHAR(25),
    CONSTRAINT FK_battle_location FOREIGN KEY (locationID) REFERENCES location(locationID)
);

CREATE TABLE battleTrainer (
     battleTrainerID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
     trainerID INT NOT NULL,
    battleID INT NOT NULL,
    isWinner BOOL NOT NULL,
    CONSTRAINT FK_battleTrainer_trainer FOREIGN KEY (trainerID) REFERENCES trainer(trainerID),
    CONSTRAINT FK_battleTrainer_battleID FOREIGN KEY (battleID) REFERENCES battle(battleID)
);

CREATE TABLE tournament (
     tournamentID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    locationID INT NOT NULL,
    startDate DATE NOT NULL,
    endDate DATE NOT NULL,
    prize VARCHAR(25),
    CONSTRAINT AK_tournament UNIQUE (name, startDate),
    CONSTRAINT FK_tournament_location FOREIGN KEY (locationID) REFERENCES location(locationID)
);

CREATE TABLE tournamentParticipation (
     participationID INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    trainerID INT NOT NULL,
    tournamentID INT NOT NULL,
    result VARCHAR(25) NOT NULL,
    CONSTRAINT FK_tournamentParticipation_trainer FOREIGN KEY (trainerID) REFERENCES
trainer(trainerID),
    CONSTRAINT FK_tournamentParticipation_tournament FOREIGN KEY (tournamentID) REFERENCES
tournament(tournamentID)
);
```

3. Assignment 3 – Fill in data

```
INSERT INTO type (name)
VALUES
    ('Normal'),
   ('Fire'),
   ('Water'),
   ('Electric'),
   ('Grass'),
   ('Ice'),
   ('Fighting'),
   ('Poison'),
   ('Ground'),
   ('Flying'),
   ('Psychic'),
   ('Bug'),
   ('Rock'),
   ('Ghost'),
   ('Dragon'),
   ('Dark'),
   ('Steel'),
   ('Fairy');
```

```
INSERT INTO region (name)
VALUES
     ('Kanto'),
    ('Johto'),
    ('Hoenn'),
    ('Sinnoh'),
    ('Unova'),
    ('Kalos'),
    ('Alola'),
    ('Galar'),
    ('Paldea'),
    ('Orre'),
    ('Fiore'),
    ('Almia'),
    ('Ransei'),
    ('Pasio'),
    ('Sevii Islands');

INSERT INTO difficulty (name)
VALUES
     ('Beginner'),
    ('Easy'),
    ('Normal'),
    ('Intermediate'),
    ('Challenging'),
    ('Hard'),
    ('Expert'),
    ('Elite'),
    ('Champion'),
    ('Legendary'),
    ('Novice'),
    ('Advanced'),
    ('Master'),
    ('Extreme'),
    ('Impossible');

-- Kanto
INSERT INTO location (name) VALUES
    ('Pallet Town'),
    ('Cerulean City'),
    ('Lavender Town');

-- Johto
INSERT INTO location (name) VALUES
    ('New Bark Town'),
    ('Goldenrod City'),
    ('Ecruteak City');

-- Hoenn
INSERT INTO location (name) VALUES
    ('Littleroot Town'),
    ('Slateport City'),
    ('Fortree City');

-- Sinnoh
INSERT INTO location (name) VALUES
```

```
    ('Twinleaf Town'),
    ('Jubilife City'),
    ('Hearthome City');

-- Unova
INSERT INTO location (name) VALUES
    ('Nuvema Town'),
    ('Castelia City'),
    ('Nimbasa City');

-- Kalos
INSERT INTO location (name) VALUES
    ('Vaniville Town'),
    ('Lumiose City'),
    ('Shalour City');

-- Alola
INSERT INTO location (name) VALUES
    ('Iki Town'),
    ('Hau’oli City'),
    ('Heahea City');

-- Galar
INSERT INTO location (name) VALUES
    ('Postwick'),
    ('Motostoke'),
    ('Hammerlocke');

-- Paldea
INSERT INTO location (name) VALUES
    ('Mesagoza'),
    ('Cortondo'),
    ('Artazon');

-- Orre
INSERT INTO location (name) VALUES
    ('Agate Village'),
    ('Phenac City'),
    ('Pyrite Town');

-- Fiore
INSERT INTO location (name) VALUES
    ('Fiore Academy'),
    ('Fall City'),
    ('Lilycove Grove');

-- Almia
INSERT INTO location (name) VALUES
    ('Almia Town'),
    ('Techno City'),
    ('Great Canyon');

-- Ransei
INSERT INTO location (name) VALUES
    ('Chizu'),
    ('Osaka Castle'),
    ('Shirogane');

-- Pasio
INSERT INTO location (name) VALUES
    ('Pasio City'),
    ('Idol Hall'),
    ('Trainer Stadium');

-- Sevii Islands
INSERT INTO location (name) VALUES
    ('One Island'),
    ('Two Island'),
    ('Sevii Port');

INSERT INTO move (name, typeID, power, accuracy, category, uses, description)
VALUES
    ('Tackle', 1, 40, 100, 'Physical', 35, 'A basic physical attack.'),
    ('Flamethrower', 2, 90, 100, 'Special', 15, 'A powerful stream of fire.'),
    ('Hydro Pump', 3, 110, 80, 'Special', 5, 'Blast water at high pressure.'),
    ('Thunderbolt', 4, 90, 100, 'Special', 15, 'A strong electric attack.'),
    ('Vine Whip', 5, 45, 100, 'Physical', 25, 'Strikes the opponent with vines.'),
    ('Ice Beam', 6, 90, 100, 'Special', 10, 'Fires a chilling beam.'),
    ('Psychic', 11, 90, 100, 'Special', 10, 'Uses psychic power to attack.'),
    ('Shadow Ball', 14, 80, 100, 'Special', 15, 'Throws a shadowy blob at the opponent.'),
    ('Dragon Claw', 15, 80, 100, 'Physical', 15, 'Slashes the opponent with sharp claws.'),
    ('Rock Slide', 13, 75, 90, 'Physical', 10, 'Large rocks slide down onto the opponent.'),
    ('Earthquake', 9, 100, 100, 'Physical', 10, 'Causes an earthquake to damage all.'),
    ('Air Slash', 10, 75, 95, 'Special', 15, 'Cuts the enemy with a blade of wind.'),
    ('Iron Tail', 17, 100, 75, 'Physical', 15, 'Attacks with a hard metal tail.'),
    ('Dark Pulse', 16, 80, 100, 'Special', 15, 'Emits a wave of dark energy.'),
    ('Moonblast', 18, 95, 100, 'Special', 10, 'Blasts a powerful fairy energy wave.');

INSERT INTO pokémonSpecies (pokédexNo, species, type1ID, type2ID, level, healthPower, attack, defense)
VALUES
    (1, 'Bulbasaur', 5, 8, 5, 45, 49, 49),
    (4, 'Charmander', 2, NULL, 5, 39, 52, 43),
    (7, 'Squirtle', 3, NULL, 5, 44, 48, 65),
    (25, 'Pikachu', 3, NULL, 5, 35, 55, 40),
    (39, 'Jigglypuff', 1, 18, 5, 115, 45, 20),
    (52, 'Meowth', 1, NULL, 5, 40, 45, 35),
    (133, 'Eevee', 1, NULL, 5, 55, 55, 50),
    (150, 'Mewtwo', 11, NULL, 70, 106, 110, 90),
    (6, 'Charizard', 2, 10, 36, 78, 84, 78),
    (94, 'Gengar', 14, 8, 25, 60, 65, 60),
    (149, 'Dragonite', 15, 10, 55, 91, 134, 95),
    (131, 'Lapras', 3, 6, 25, 130, 85, 80),
    (143, 'Snorlax', 1, NULL, 30, 160, 110, 65),
    (59, 'Arcanine', 2, NULL, 30, 90, 110, 80),
    (248, 'Tyranitar', 13, 17, 55, 100, 134, 110);

INSERT INTO pokémon (captureDate, nickname, pokémonSpeciesID)
VALUES
    ('2025-01-10', 'Bulby', 1),
    ('2025-01-12', 'Char', 2),
    ('2025-01-15', 'Squirt', 3),
    ('2025-01-20', 'Sparky', 4),
    ('2025-01-22', 'Puffy', 5),
    ('2025-01-25', 'Meow', 6),
    ('2025-02-01', 'Eevee', 7),
    ('2025-02-05', 'MewtwoX', 8),
    ('2025-02-10', 'Flame', 9),
    ('2025-02-12', 'Gengy', 10),
    ('2025-02-15', 'Drago', 11),
    ('2025-02-18', 'Lap', 12),
    ('2025-02-20', 'Snorl', 13),
    ('2025-02-25', 'Arca', 14),
    ('2025-02-28', 'Tyra', 15);

INSERT INTO trainer (firstName, surname, dateOfBirth, regionID, trainerRank, email, phoneNo)
VALUES
    ('Ash', 'Ketchum', '1990-05-22', 1, 'Beginner', 'ash.ketchum@email.com', '080-1234-5678'),
    ('Misty', 'Waterflower', '1991-07-01', 1, 'Intermediate', 'misty@email.com', '090-2345-6789'),
    ('Brock', 'Stone', '1989-11-03', 1, 'Gym Leader', 'brock@email.com', '070-3456-7890'),
    ('Gary', 'Oak', '1990-08-15', 1, 'Elite', 'gary.oak@email.com', '080-4567-8901'),
    ('Dawn', 'Berlitz', '1992-03-20', 4, 'Beginner', 'dawn@email.com', '090-5678-9012'),
    ('May', 'Maple', '1991-10-10', 3, 'Intermediate', 'may@email.com', '070-6789-0123'),
    ('Max', 'Maple', '1995-02-18', 3, 'Beginner', 'max@email.com', '080-7890-1234'),
    ('Serena', 'Fontaine', '1992-06-25', 6, 'Intermediate', 'serena@email.com', '090-8901-2345'),
    ('Clemont', 'Citron', '1990-09-05', 6, 'Gym Leader', 'clemont@email.com', '070-9012-3456'),
    ('Kiawe', 'Lava', '1991-12-12', 7, 'Gym Leader', 'kiawe@email.com', '080-0123-4567'),
    ('Lillie', 'Lily', '1993-01-15', 7, 'Beginner', 'lillie@email.com', '090-1234-5678'),
    ('Hop', 'Smith', '1992-07-18', 8, 'Intermediate', 'hop@email.com', '070-2345-6789'),
    ('Marnie', 'Rock', '1993-11-22', 8, 'Gym Leader', 'marnie@email.com', '080-3456-7890'),
    ('Gloria', 'Rose', '1994-03-10', 9, 'Beginner', 'gloria@email.com', '090-4567-8901'),
    ('Victor', 'Cruz', '1988-05-30', 10, 'Elite', 'victor@email.com', '070-5678-9012');

INSERT INTO pokémonTrainer (pokémonID, trainerID, dateAcquired)
VALUES
    (1, 1, '2025-01-10'),
    (2, 1, '2025-01-12'),
    (3, 2, '2025-01-15'),
    (4, 2, '2025-01-20'),
    (5, 3, '2025-01-22'),
    (6, 3, '2025-01-25'),
    (7, 4, '2025-02-01'),
    (8, 4, '2025-02-05'),
    (9, 5, '2025-02-10'),
    (10, 5, '2025-02-12'),
    (11, 6, '2025-02-15'),
    (12, 6, '2025-02-18'),
    (13, 7, '2025-02-20'),
    (14, 7, '2025-02-25'),
    (15, 8, '2025-02-28');

INSERT INTO gym (name, regionID, leaderID, locationID, badge, difficultyID, establishedYear)
VALUES
    ('Pewter Gym', 1, 3, 1, 'Boulder Badge', 1, 1997),
    ('Cerulean Gym', 1, 2, 2, 'Cascade Badge', 2, 1997),
    ('Vermilion Gym', 1, 4, 3, 'Thunder Badge', 2, 1997),
    ('Celadon Gym', 1, 1, 4, 'Rainbow Badge', 3, 1997),
    ('Saffron Gym', 1, 5, 5, 'Marsh Badge', 3, 1997),
    ('Violet Gym', 2, 6, 6, 'Fog Badge', 1, 1998),
    ('Goldenrod Gym', 2, 7, 7, 'Plain Badge', 2, 1998),
    ('Ecruteak Gym', 2, 8, 8, 'Storm Badge', 3, 1998),
    ('Rustboro Gym', 3, 9, 9, 'Stone Badge', 1, 2000),
    ('Mauville Gym', 3, 10, 10, 'Dynamo Badge', 2, 2000),
    ('Hearthome Gym', 4, 11, 11, 'Relic Badge', 2, 2001),
    ('Sunyshore Gym', 4, 12, 12, 'Beacon Badge', 3, 2001),
    ('Nimbasa Gym', 5, 13, 13, 'Bolt Badge', 2, 2010),
    ('Driftveil Gym', 5, 14, 14, 'Mine Badge', 1, 2010),
    ('Hammerlocke Gym', 8, 15, 15, 'Tower Badge', 3, 2019);

INSERT INTO battle (date, locationID, duration, weather, format)
VALUES
    ('2025-03-01 10:00:00', 1, '00:30:00', 'Sunny', 'Single'),
    ('2025-03-02 14:00:00', 2, '00:45:00', 'Rain', 'Double'),
    ('2025-03-03 16:30:00', 3, '00:25:00', 'Sunny', 'Single'),
    ('2025-03-04 12:00:00', 4, '00:50:00', 'Fog', 'Double'),
    ('2025-03-05 09:00:00', 5, '00:35:00', 'Windy', 'Single'),
    ('2025-03-06 11:00:00', 6, '00:40:00', 'Rain', 'Single'),
    ('2025-03-07 15:00:00', 7, '01:00:00', 'Sunny', 'Double'),
    ('2025-03-08 13:30:00', 8, '00:20:00', 'Snow', 'Single'),
    ('2025-03-09 10:15:00', 9, '00:55:00', 'Sunny', 'Double'),
    ('2025-03-10 14:45:00', 10, '00:30:00', 'Windy', 'Single'),
    ('2025-03-11 09:30:00', 11, '00:40:00', 'Rain', 'Single'),
    ('2025-03-12 16:00:00', 12, '00:50:00', 'Sunny', 'Double'),
    ('2025-03-13 10:00:00', 13, '00:35:00', 'Fog', 'Single'),
    ('2025-03-14 15:15:00', 14, '00:45:00', 'Snow', 'Double'),
    ('2025-03-15 12:00:00', 15, '00:25:00', 'Sunny', 'Single');

INSERT INTO battleTrainer (trainerID, battleID, isWinner)
VALUES
    (1, 1, TRUE),
    (2, 1, FALSE),
    (3, 2, TRUE),
    (4, 2, FALSE),
    (5, 3, TRUE),
    (6, 3, FALSE),
    (7, 4, TRUE),
    (8, 4, FALSE),
    (9, 5, TRUE),
    (10, 5, FALSE),
    (11, 6, TRUE),
    (12, 6, FALSE),
    (13, 7, TRUE),
    (14, 7, FALSE),
    (15, 8, TRUE),
    (1, 8, FALSE),
    (2, 9, TRUE),
    (3, 9, FALSE),
    (4, 10, TRUE),
    (5, 10, FALSE),
    (6, 11, TRUE),
    (7, 11, FALSE),
    (8, 12, TRUE),
    (9, 12, FALSE),
    (10, 13, TRUE),
    (11, 13, FALSE),
    (12, 14, TRUE),
    (13, 14, FALSE),
    (14, 15, TRUE),
    (15, 15, FALSE);

INSERT INTO tournament (name, locationID, startDate, endDate, prize)
VALUES
    ('Kanto Regional', 1, '2025-06-01', '2025-06-03', 'Trophy & Coins'),
    ('Johto Invitational', 2, '2025-06-05', '2025-06-07', 'Rare Candy Pack'),
    ('Hoenn Open', 3, '2025-06-10', '2025-06-12', 'Legendary Token'),
    ('Sinnoh Championship', 4, '2025-06-15', '2025-06-17', 'Battle Points'),
    ('Unova Cup', 5, '2025-06-20', '2025-06-22', 'Master Ball'),
    ('Kalos Classic', 6, '2025-06-25', '2025-06-27', 'Trophy & Coins'),
    ('Alola Challenge', 7, '2025-07-01', '2025-07-03', 'Rare Candy Pack'),
    ('Galar League', 8, '2025-07-05', '2025-07-07', 'Legendary Token'),
    ('Johto Masters', 2, '2025-07-10', '2025-07-12', 'Battle Points'),
    ('Kanto Elite', 1, '2025-07-15', '2025-07-17', 'Master Ball'),
    ('Sinnoh Open', 4, '2025-07-20', '2025-07-22', 'Trophy & Coins'),
    ('Hoenn Championship', 3, '2025-07-25', '2025-07-27', 'Rare Candy Pack'),
    ('Unova Invitational', 5, '2025-08-01', '2025-08-03', 'Legendary Token'),
    ('Kalos Masters', 6, '2025-08-05', '2025-08-07', 'Battle Points'),
    ('Alola League', 7, '2025-08-10', '2025-08-12', 'Master Ball');

INSERT INTO tournamentParticipation (trainerID, tournamentID, result)
VALUES
    (1, 1, 'Winner'),
    (2, 1, 'Runner-up'),
    (3, 2, 'Semi-finalist'),
    (4, 2, 'Quarter-final'),
    (5, 3, 'Winner'),
    (6, 3, 'Runner-up'),
    (7, 4, 'Winner'),
    (8, 4, 'Runner-up'),
    (9, 5, 'Winner'),
    (10, 5, 'Runner-up'),
    (11, 6, 'Winner'),
    (12, 6, 'Runner-up'),
    (13, 7, 'Winner'),
    (14, 7, 'Runner-up'),
    (15, 8, 'Winner'),
    (1, 8, 'Runner-up'),
    (2, 9, 'Winner'),
    (3, 9, 'Runner-up'),
    (4, 10, 'Winner'),
    (5, 10, 'Runner-up'),
    (6, 11, 'Winner'),
    (7, 11, 'Runner-up'),
    (8, 12, 'Winner'),
    (9, 12, 'Runner-up'),
    (10, 13, 'Winner'),
    (11, 13, 'Runner-up'),
    (12, 14, 'Winner'),
    (13, 14, 'Runner-up'),
    (14, 15, 'Winner'),
    (15, 15, 'Runner-up');
```

REFERENCES:

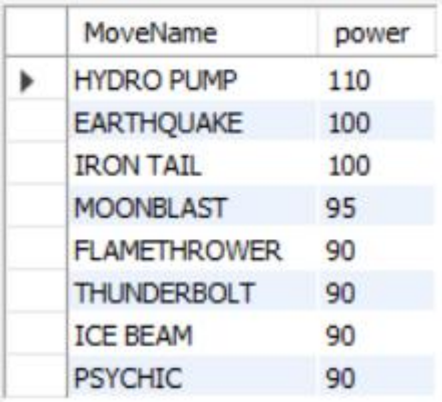
Used ChatGPT to fill in my tables with sample data.
Prompt:
“Give me sample data to populate my pokémon database. I have the following tables: type, move, region, difficulty, location, pokémonSpecies, pokémon, pokémonMove, trainer, pokémonTrainer, gym, battle, battleTrainer, tournament, and tournamentParticipation.”

4. Assignment 4 - WHERE + scalar functions

4.1. Query 1

Show all move names whose power is > 80 in UPPERCASE.

```
SELECT UPPER(name) AS MoveName, power
FROM move
WHERE power > 80;
```



MoveName	power
HYDRO PUMP	110
EARTHQUAKE	100
IRON TAIL	100
MOONBLAST	95
FLAMETHROWER	90
THUNDERBOLT	90
ICE BEAM	90
PSYCHIC	90

Content check:

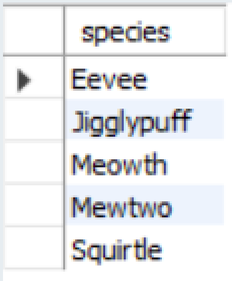
- ✓ **Operators (>)**
- □ In
- □ [not] Between
- □ [not] like (with % or _ and escape)
- □ Is [not] null
- □ And, or, not

- ✓ **String functions (UPPER)**
- □ Numeric functions
- □ Date and time functions
- □ Ifnull or coalesce
- □ Distinct
- □ Order by

4.2. Query 2

Show Pokémon species names that do not contain the letter ‘A’, ordered alphabetically.

```
SELECT species
FROM pokémonSpecies
WHERE species NOT LIKE '%a%'
ORDER BY species ASC;
```



	species
▶	Eevee
	Jigglypuff
	Meowth
	Mewtwo
	Squirtle

Content check:

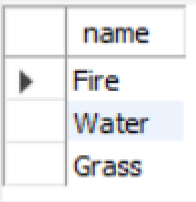
□ Operators (=, <, >, <>, …)
□ In
□ [not] Between
✓ **[not] like (with % or _ and escape)**
□ Is [not] null
□ And, or, not

□ String functions
□ Numeric functions
□ Date and time functions
□ Ifnull or coalesce
□ Distinct
✓ **Order by**

4.3. Query 3

Show distinct type names for types with IDs in 2, 3, or 5 (Fire, Water, Grass).

```
SELECT DISTINCT name
FROM type
WHERE typeID IN (2, 3, 5);
```



	name
▶	Fire
	Water
	Grass

Content check:

□ Operators (=, <, >, <>, …)
✓ **In**
□ [not] Between
□ [not] like (with % or _ and escape)

□ String functions
□ Numeric functions
□ Date and time functions
□ Ifnull or coalesce

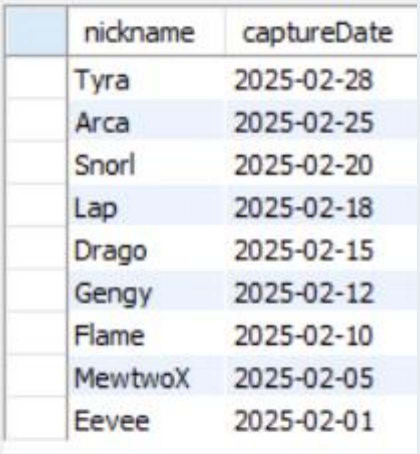
□ Is [not] null
□ And, or, not

✓ **Distinct**
□ Order by

4.4. Query 4

Display all Pokémon captured in February 2025, showing nickname and capture date, most recent first.

```
SELECT nickname, captureDate
FROM pokémon
WHERE YEAR(captureDate) = 2025 AND MONTH(captureDate) BETWEEN 2
AND 2
ORDER BY captureDate DESC;
```



nickname	captureDate
Tyra	2025-02-28
Arca	2025-02-25
Snorl	2025-02-20
Lap	2025-02-18
Drago	2025-02-15
Gengy	2025-02-12
Flame	2025-02-10
MewtwoX	2025-02-05
Eevee	2025-02-01

Content check:

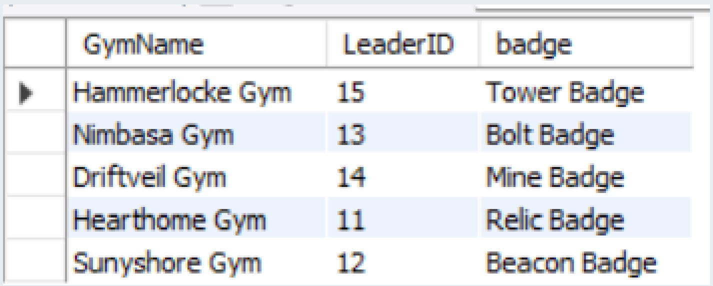
□ Operators (=, <, >, <>, …)
□ In
✓ **[not] Between**
□ [not] like (with % or _ and escape)
□ Is [not] null
✓ **And, or, not**

□ String functions
□ Numeric functions
✓ **Date and time functions**
□ Ifnull or coalesce
□ Distinct
□ Order by

4.5. Query 5

List all gyms that have a leader assigned (leaderID IS NOT NULL) and were established after 2000, showing the gym name, the leaderID, and the badge. Sort by established year descending.

```
SELECT name AS GymName, COALESCE(leaderID, 0) AS LeaderID, badge
FROM gym
WHERE leaderID IS NOT NULL AND establishedYear > 2000
ORDER BY establishedYear DESC;
```



GymName	LeaderID	badge
Hammerlocke Gym	15	Tower Badge
Nimbasa Gym	13	Bolt Badge
Driftveil Gym	14	Mine Badge
Hearthome Gym	11	Relic Badge
Sunyshore Gym	12	Beacon Badge

Content check:

☐ Operators (=, <, >, <>, …)
☐ In
☐ [not] Between
☐ [not] like (with % or _ and escape)
✓ **Is [not] null**
☐ And, or, not

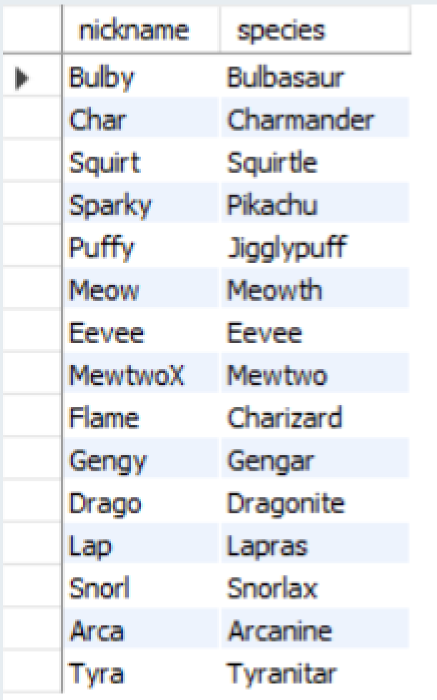
☐ String functions
☐ Numeric functions
☐ Date and time functions
✓ **Ifnull or coalesce**
☐ Distinct
☐ Order by

5. Assignment 5 - JOINS

5.1. Inner join between two tables

Show the nickname of each Pokémon together with the species it belongs to.

```
SELECT p.nickname, s.species
FROM pokémon p
JOIN pokémonSpecies s
ON p.pokémonSpeciesID = s.pokémonSpeciesID;
```

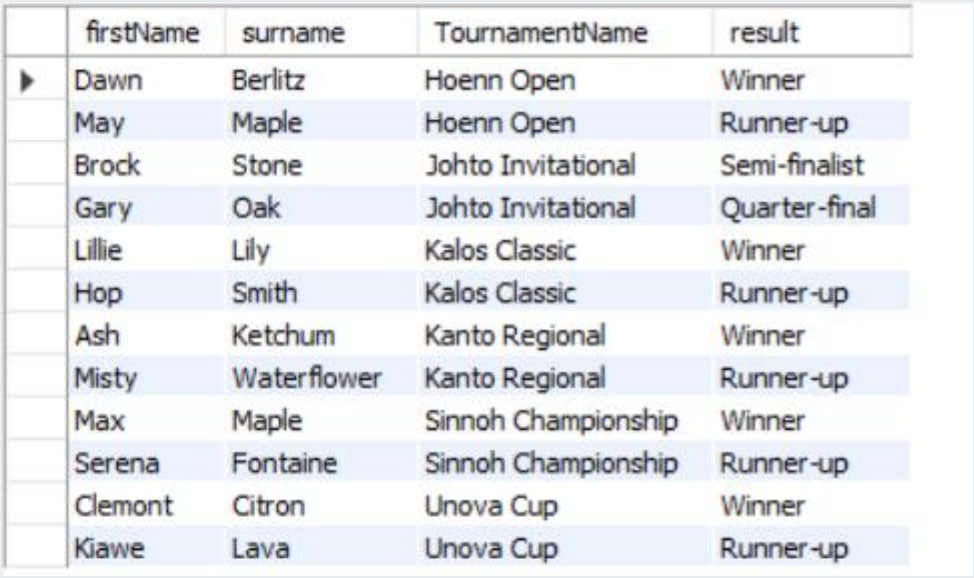


nickname	species
Bulby	Bulbasaur
Char	Charmander
Squirt	Squirtle
Sparky	Pikachu
Puffy	Jigglypuff
Meow	Meowth
Eevee	Eevee
MewtwoX	Mewtwo
Flame	Charizard
Gengy	Gengar
Drago	Dragonite
Lap	Lapras
Snorl	Snorlax
Arca	Arcanine
Tyra	Tyranitar

5.2. Inner join between more than two tables

List trainers who participated in tournaments starting in June 2025, along with the tournament name and result.

```
SELECT t.firstName, t.surname, tor.name AS TournamentName, tp.result
FROM trainer t
JOIN tournamentParticipation tp
ON t.trainerID = tp.trainerID
JOIN tournament tor
ON tp.tournamentID = tor.tournamentID
WHERE tor.startDate BETWEEN '2025-06-01' AND '2025-06-30';
```

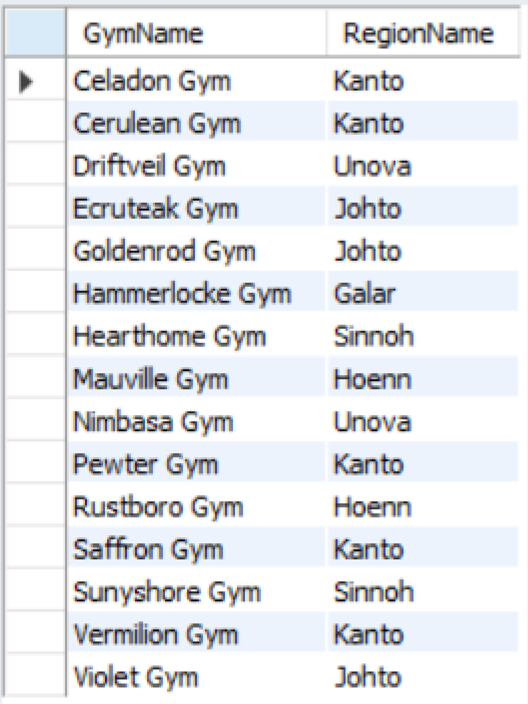


firstName	surname	TournamentName	result
Dawn	Berlitz	Hoenn Open	Winner
May	Maple	Hoenn Open	Runner-up
Brock	Stone	Johto Invitational	Semi-finalist
Gary	Oak	Johto Invitational	Quarter-final
Lillie	Lily	Kalos Classic	Winner
Hop	Smith	Kalos Classic	Runner-up
Ash	Ketchum	Kanto Regional	Winner
Misty	Waterflower	Kanto Regional	Runner-up
Max	Maple	Sinnoh Championship	Winner
Serena	Fontaine	Sinnoh Championship	Runner-up
Clemont	Citron	Unova Cup	Winner
Kiawe	Lava	Unova Cup	Runner-up

5.3. A left outer join

Show all gyms and their regions. If a gym has no region assigned, still display the gym. Order by gym name.

```
SELECT g.name AS GymName, r.name AS RegionName
FROM gym g
LEFT JOIN region r ON g.regionID = r.regionID
ORDER BY g.name;
```

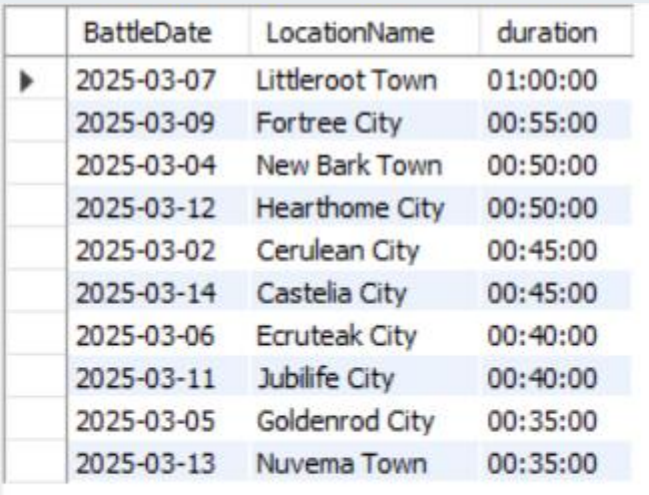


GymName	RegionName
Celadon Gym	Kanto
Cerulean Gym	Kanto
Driftveil Gym	Unova
Ecruteak Gym	Johto
Goldenrod Gym	Johto
Hammerlocke Gym	Galar
Hearthome Gym	Sinnoh
Mauville Gym	Hoenn
Nimbasa Gym	Unova
Pewter Gym	Kanto
Rustboro Gym	Hoenn
Saffron Gym	Kanto
Sunyshore Gym	Sinnoh
Vermilion Gym	Kanto
Violet Gym	Johto

5.4. A right outer join

Display all battles and their locations. If a location has no battles, still show the location. Only include battles longer than 30 minutes and order by duration descending.

```
SELECT DATE(b.date) AS BattleDate, l.name AS LocationName, b.duration
FROM battle b
RIGHT JOIN location l ON b.locationID = l.locationID
WHERE b.duration > '00:30:00'
ORDER BY b.duration DESC;
```



	BattleDate	LocationName	duration
▶	2025-03-07	Littleroot Town	01:00:00
	2025-03-09	Fortree City	00:55:00
	2025-03-04	New Bark Town	00:50:00
	2025-03-12	Hearthome City	00:50:00
	2025-03-02	Cerulean City	00:45:00
	2025-03-14	Castelia City	00:45:00
	2025-03-06	Ecruteak City	00:40:00
	2025-03-11	Jubilife City	00:40:00
	2025-03-05	Goldenrod City	00:35:00
	2025-03-13	Nuvema Town	00:35:00

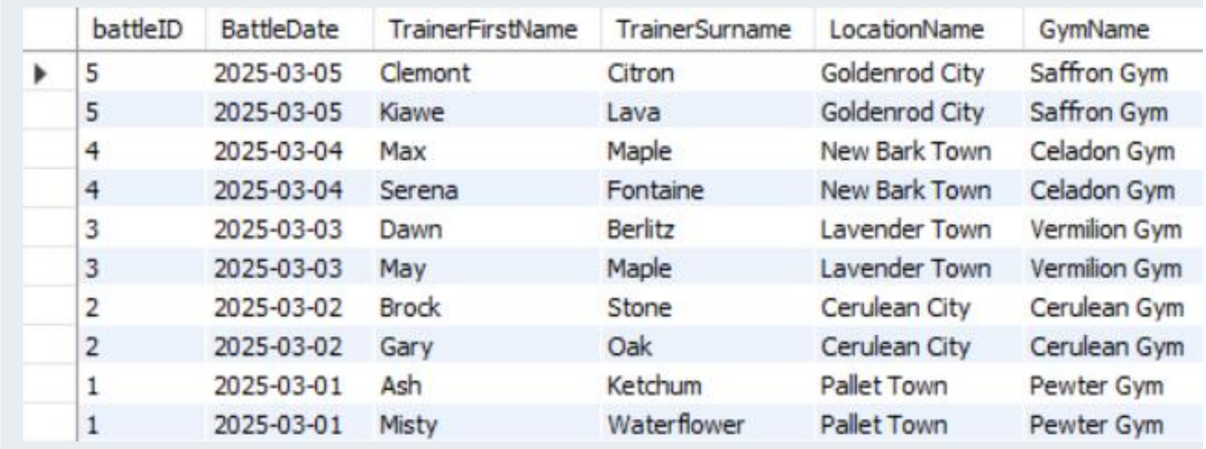
5.5. You can choose the fifth join.

Joins chosen:

- INNER JOIN => **battle** to **battleTrainer** to **trainer**
- LEFT JOIN => **battle** to **location** (include battles even if location info is missing)
- RIGHT JOIN => **location** to **gym** (include gyms even if no battles happened there)

Show battles in the Kanto region with trainer names, location names, and any gyms in those locations.

```
SELECT b.battleID, DATE(b.date) AS BattleDate, t.firstName AS
TrainerFirstName, t.surname AS TrainerSurname, l.name AS LocationName,
g.name AS GymName
FROM battle b
JOIN battleTrainer bt ON b.battleID = bt.battleID
JOIN trainer t ON bt.trainerID = t.trainerID
LEFT JOIN location l ON b.locationID = l.locationID
RIGHT JOIN gym g ON l.locationID = g.locationID
WHERE g.regionID = 1
ORDER BY b.date DESC;
```



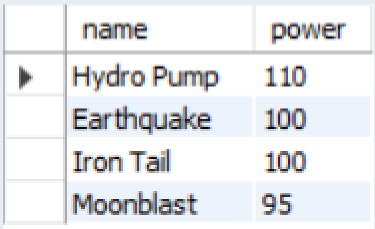
battleID	BattleDate	TrainerFirstName	TrainerSurname	LocationName	GymName
5	2025-03-05	Clemont	Citron	Goldenrod City	Saffron Gym
5	2025-03-05	Kiawe	Lava	Goldenrod City	Saffron Gym
4	2025-03-04	Max	Maple	New Bark Town	Celadon Gym
4	2025-03-04	Serena	Fontaine	New Bark Town	Celadon Gym
3	2025-03-03	Dawn	Berlitz	Lavender Town	Vermilion Gym
3	2025-03-03	May	Maple	Lavender Town	Vermilion Gym
2	2025-03-02	Brock	Stone	Cerulean City	Cerulean Gym
2	2025-03-02	Gary	Oak	Cerulean City	Cerulean Gym
1	2025-03-01	Ash	Ketchum	Pallet Town	Pewter Gym
1	2025-03-01	Misty	Waterflower	Pallet Town	Pewter Gym

6. Assignment 6 - Subqueries

6.1. Single-row subquery

Show all moves with higher power than the move "Thunderbolt".

```
SELECT name, power
FROM move
WHERE power > (
   SELECT power
   FROM move
   WHERE name = 'Thunderbolt'
);
```

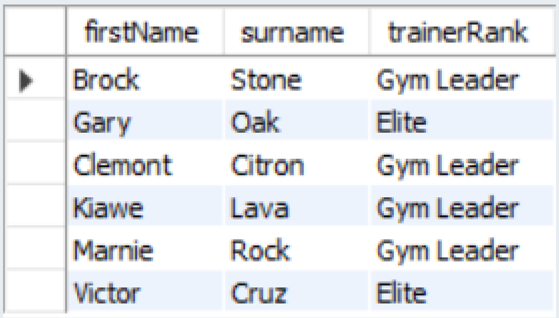


	name	power
▶	Hydro Pump	110
	Earthquake	100
	Iron Tail	100
	Moonblast	95

6.2. Multiple-row subquery

Find all Trainers who are ranked 'Gym Leader' or 'Elite'.

```
SELECT firstName, surname, trainerRank
FROM trainer
WHERE trainerRank IN (
   SELECT trainerRank
   FROM trainer
   WHERE trainerRank = 'Gym Leader' OR trainerRank = 'Elite'
);
```

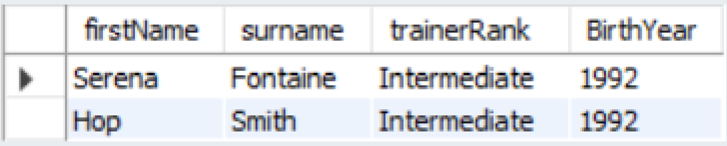


	firstName	surname	trainerRank
▶	Brock	Stone	Gym Leader
	Gary	Oak	Elite
	Clemont	Citron	Gym Leader
	Kiawe	Lava	Gym Leader
	Marnie	Rock	Gym Leader
	Victor	Cruz	Elite

6.3. Double key subquery

Find all the Trainers who have the same Trainer Rank and Birth Year as the trainer named ‘Serena’.

```
SELECT firstName, surname, trainerRank, YEAR(dateOfBirth) AS BirthYear
FROM trainer
WHERE (trainerRank, YEAR(dateOfBirth)) IN (
    SELECT trainerRank, YEAR(dateOfBirth)
    FROM trainer
    WHERE firstName = 'Serena'
);
```

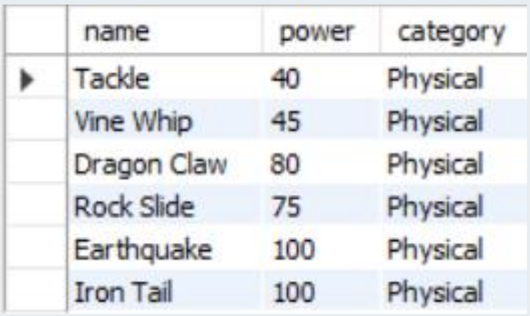


	firstName	surname	trainerRank	BirthYear
▶	Serena	Fontaine	Intermediate	1992
	Hop	Smith	Intermediate	1992

6.4. Nested subquery

Show all moves that have the same category as the move ‘Tackle’.

```
SELECT name, power, category
FROM move
WHERE category = (
    SELECT category
    FROM move
    WHERE name = (
        SELECT name
        FROM move
        WHERE name = 'Tackle'
    )
);
```

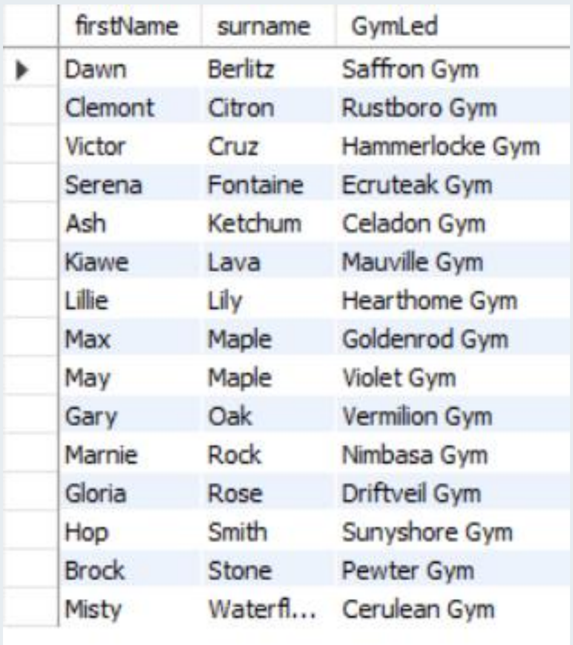


	name	power	category
▶	Tackle	40	Physical
	Vine Whip	45	Physical
	Dragon Claw	80	Physical
	Rock Slide	75	Physical
	Earthquake	100	Physical
	Iron Tail	100	Physical

6.5. Subquery in the select

Show each trainer’s name along with the gym they lead.

```
SELECT t.firstName, t.surname,
   (SELECT g.name
    FROM gym g
    WHERE g.leaderID = t.trainerID) AS GymLed
FROM trainer t
ORDER BY t.surname;
```



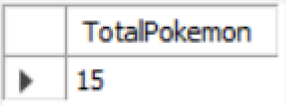
firstName	surname	GymLed
Dawn	Berlitz	Saffron Gym
Clemont	Citron	Rustboro Gym
Victor	Cruz	Hammerlocke Gym
Serena	Fontaine	Ecruteak Gym
Ash	Ketchum	Celadon Gym
Kiawe	Lava	Mauville Gym
Lillie	Lily	Hearthome Gym
Max	Maple	Goldenrod Gym
May	Maple	Violet Gym
Gary	Oak	Vermilion Gym
Marnie	Rock	Nimbasa Gym
Gloria	Rose	Driftveil Gym
Hop	Smith	Sunyshore Gym
Brock	Stone	Pewter Gym
Misty	Waterfl...	Cerulean Gym

7. Assignment 7 - Set functions

7.1. Count()

Count the total number of Pokémon in the database.

```
SELECT COUNT(*) AS TotalPokemon
FROM pokémon;
```

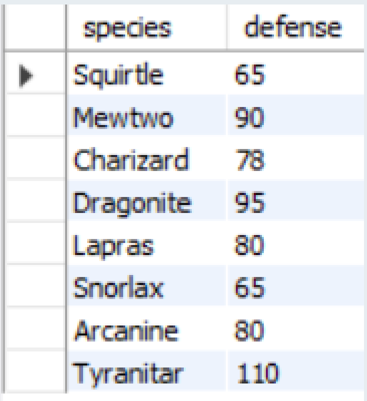


TotalPokemon
15

7.2. MIN() or MAX() or AVG() or SUM()

Find the names and defense values of all Pokémon species whose base defense is higher than the average defense of all species.

```
SELECT species, defense
FROM pokémonSpecies
WHERE defense > (
    SELECT AVG(defense)
    FROM pokémonSpecies
);
```



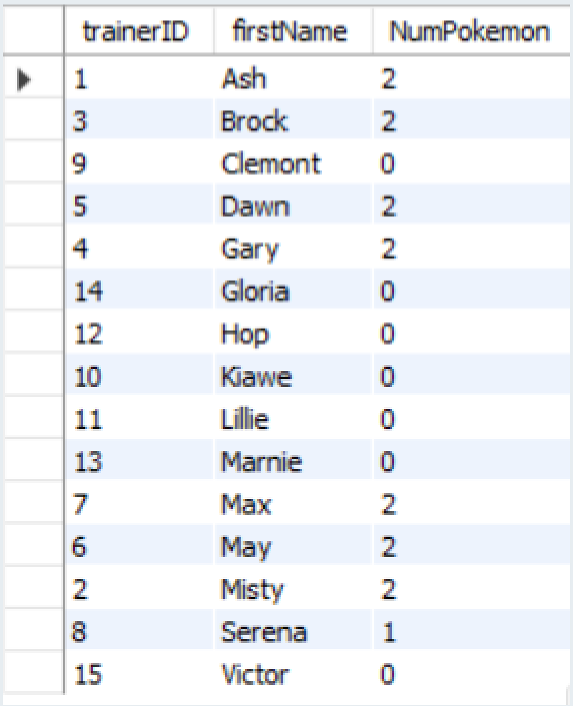
	species	defense
▶	Squirtle	65
	Mewtwo	90
	Charizard	78
	Dragonite	95
	Lapras	80
	Snorlax	65
	Arcanine	80
	Tyranitar	110

8. Assignment 8 - Correlated subqueries

8.1. Correlated subquery 1

Show trainers and the number of Pokémon they own.

```
SELECT t.trainerID, t.firstName,
   (SELECT COUNT(*)
    FROM pokémonTrainer pt
    WHERE pt.trainerID = t.trainerID) AS NumPokemon
FROM trainer t
ORDER BY t.firstName;
```

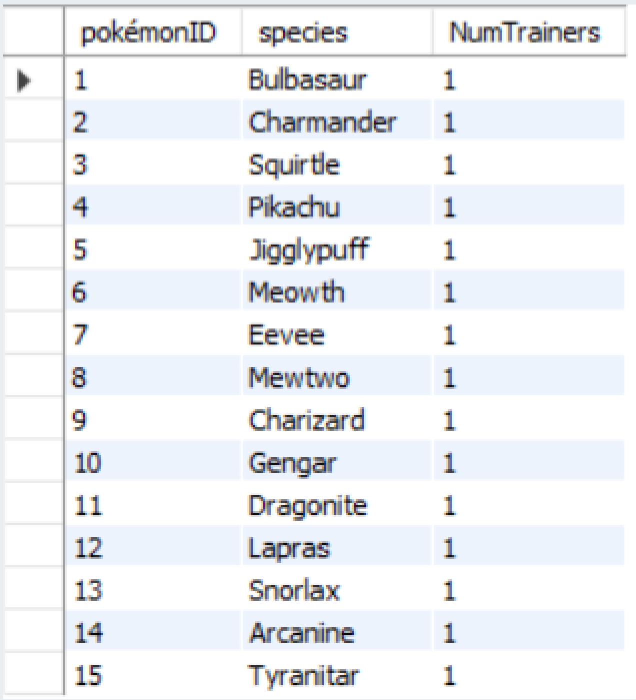


trainerID	firstName	NumPokemon
1	Ash	2
3	Brock	2
9	Clemont	0
5	Dawn	2
4	Gary	2
14	Gloria	0
12	Hop	0
10	Kiawe	0
11	Lillie	0
13	Marnie	0
7	Max	2
6	May	2
2	Misty	2
8	Serena	1
15	Victor	0

8.2. Correlated subquery 2

Show each Pokémon and the number of trainers that own it.

```
SELECT p.pokémonID, ps.species,
   (SELECT COUNT(*)
    FROM pokémonTrainer pt
    WHERE pt.pokémonID = p.pokémonID) AS NumTrainers
FROM pokémon p
JOIN pokémonSpecies ps ON p.pokémonSpeciesID = ps.pokémonSpeciesID
ORDER BY p.pokémonID;
```

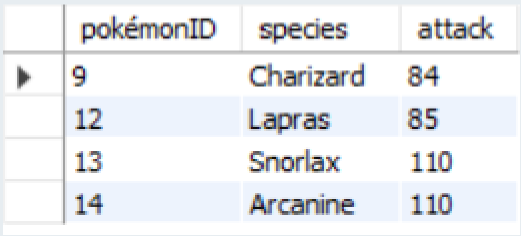


pokémonID	species	NumTrainers
1	Bulbasaur	1
2	Charmander	1
3	Squirtle	1
4	Pikachu	1
5	Jigglypuff	1
6	Meowth	1
7	Eevee	1
8	Mewtwo	1
9	Charizard	1
10	Gengar	1
11	Dragonite	1
12	Lapras	1
13	Snorlax	1
14	Arcanine	1
15	Tyranitar	1

8.3. Correlated subquery 3

Show Pokémon whose attack is higher than the average attack of their species.

```
SELECT p.pokémonID, ps.species, ps.attack
FROM pokémon p
JOIN pokémonSpecies ps ON p.pokémonSpeciesID = ps.pokémonSpeciesID
WHERE ps.attack > (
    SELECT AVG(ps2.attack)
    FROM pokémonSpecies ps2
    WHERE ps2.type1ID = ps.type1ID
);
```



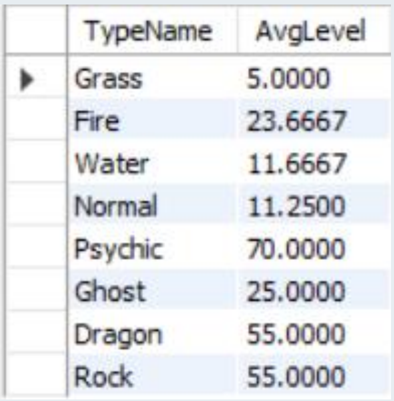
pokémonID	species	attack
9	Charizard	84
12	Lapras	85
13	Snorlax	110
14	Arcanine	110

9. Assignment 9 – Group by

9.1. Group by with one ore more set functions

Show the average level of each Pokémon type.

```
SELECT t.name AS TypeName, AVG(ps.level) AS AvgLevel
FROM pokémonSpecies ps
JOIN type t ON ps.type1ID = t.typeID
GROUP BY t.name;
```

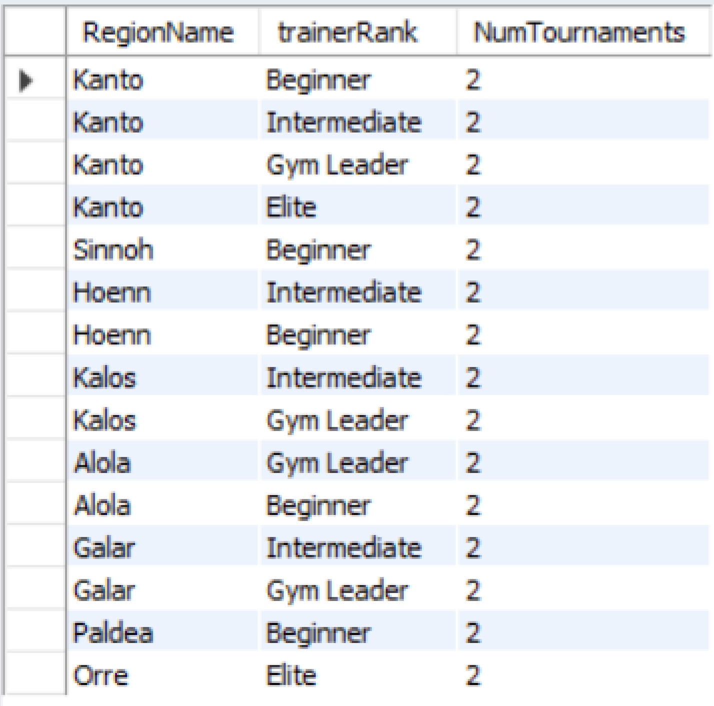


TypeName	AvgLevel
Grass	5.0000
Fire	23.6667
Water	11.6667
Normal	11.2500
Psychic	70.0000
Ghost	25.0000
Dragon	55.0000
Rock	55.0000

9.2. Group by on multiple columns

Count tournaments per region and which trainer rank participated most in them.

```
SELECT r.name AS RegionName, t.trainerRank, COUNT(tp.tournamentID) AS
NumTournaments
FROM tournamentParticipation tp
JOIN trainer t ON tp.trainerID = t.trainerID
JOIN region r ON t.regionID = r.regionID
GROUP BY r.name, t.trainerRank;
```

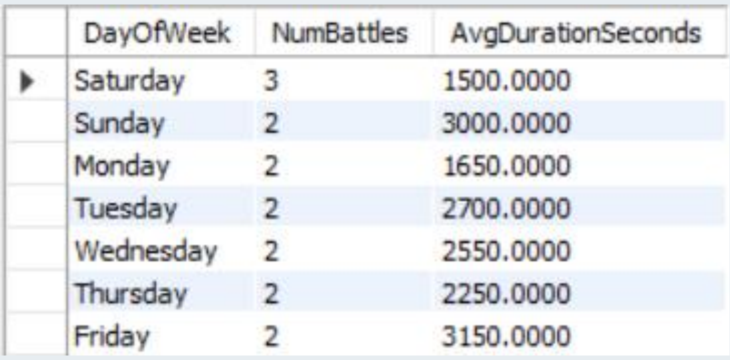


RegionName	trainerRank	NumTournaments
Kanto	Beginner	2
Kanto	Intermediate	2
Kanto	Gym Leader	2
Kanto	Elite	2
Sinnoh	Beginner	2
Hoenn	Intermediate	2
Hoenn	Beginner	2
Kalos	Intermediate	2
Kalos	Gym Leader	2
Alola	Gym Leader	2
Alola	Beginner	2
Galar	Intermediate	2
Galar	Gym Leader	2
Paldea	Beginner	2
Orre	Elite	2

9.3. Group by with an expression

Show the number of battles and the average duration (in seconds) for each day of the week.

```
SELECT DAYNAME(date) AS DayOfWeek, COUNT(battleID) AS NumBattles,
AVG(TIME_TO_SEC(duration)) AS AvgDurationSeconds
FROM battle
GROUP BY DAYNAME(date);
```

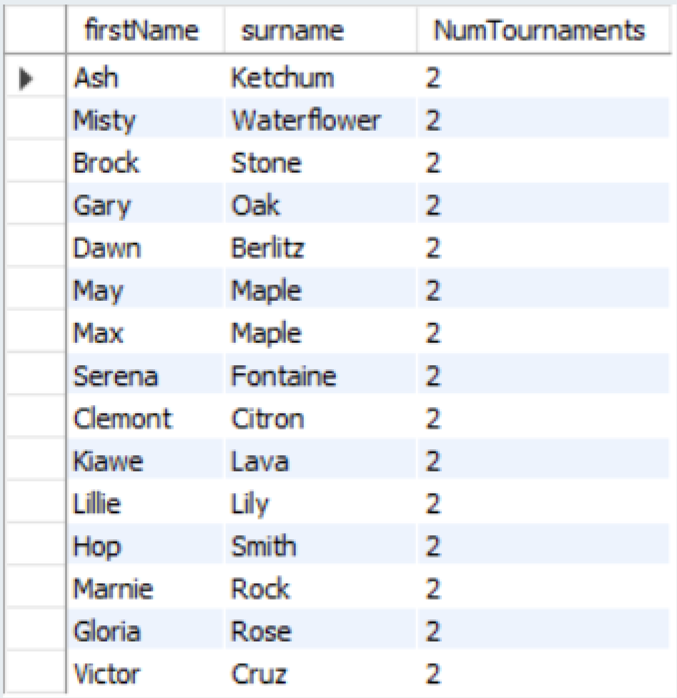


DayOfWeek	NumBattles	AvgDurationSeconds
Saturday	3	1500.0000
Sunday	2	3000.0000
Monday	2	1650.0000
Tuesday	2	2700.0000
Wednesday	2	2550.0000
Thursday	2	2250.0000
Friday	2	3150.0000

9.4. Group by with having

Show the trainers who participated in more than one tournament.

```
SELECT t.firstName, t.surname, COUNT(tp.tournamentID) AS
NumTournaments
FROM tournamentParticipation tp
JOIN trainer t ON tp.trainerID = t.trainerID
GROUP BY t.trainerID
HAVING COUNT(tp.tournamentID) > 1;
```

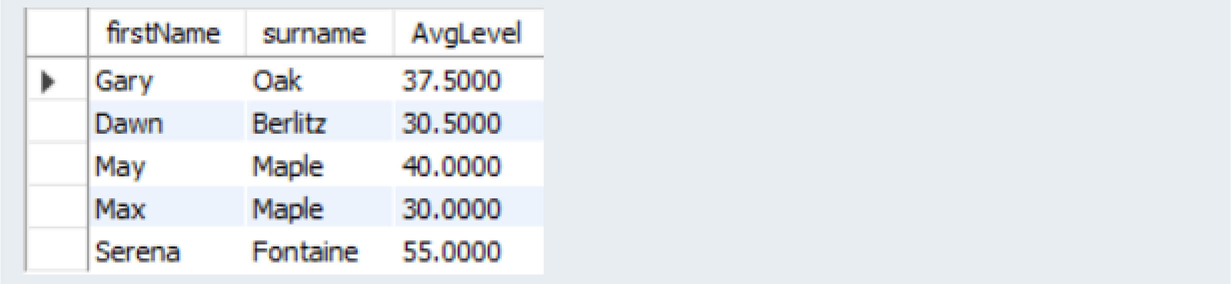


firstName	surname	NumTournaments
Ash	Ketchum	2
Misty	Waterflower	2
Brock	Stone	2
Gary	Oak	2
Dawn	Berlitz	2
May	Maple	2
Max	Maple	2
Serena	Fontaine	2
Clemont	Citron	2
Kiawe	Lava	2
Lillie	Lily	2
Hop	Smith	2
Marnie	Rock	2
Gloria	Rose	2
Victor	Cruz	2

9.5. Group by with having

Display the name along with the average level of the trainers whose Pokémon have an average level greater than 20.

```
SELECT t.firstName, t.surname, AVG(ps.level) AS AvgLevel
FROM pokémonTrainer pt
JOIN trainer t ON pt.trainerID = t.trainerID
JOIN pokémon p ON pt.pokémonID = p.pokémonID
JOIN pokémonSpecies ps ON p.pokémonSpeciesID = ps.pokémonSpeciesID
GROUP BY t.trainerID
HAVING AVG(ps.level) > 20;
```



firstName	surname	AvgLevel
Gary	Oak	37.5000
Dawn	Berlitz	30.5000
May	Maple	40.0000
Max	Maple	30.0000
Serena	Fontaine	55.0000

FOLLOW US

www.thomasmore.be
fb.com/ThomasMoreBE
#WeAreMore

THOMAS
MORE