University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ____1____

Date: ___2025.3.3_____

Group Number: _ 26_____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Astrid Zhang	49979776	d5c1m	zxyi0721@163.com
Ruiyang Zhang	12568747	z1q8i	rzhan101@student.ubc.ca
Zhaowei Cheng	56814577	v4p9a	zcheng32@student.ubc.ca

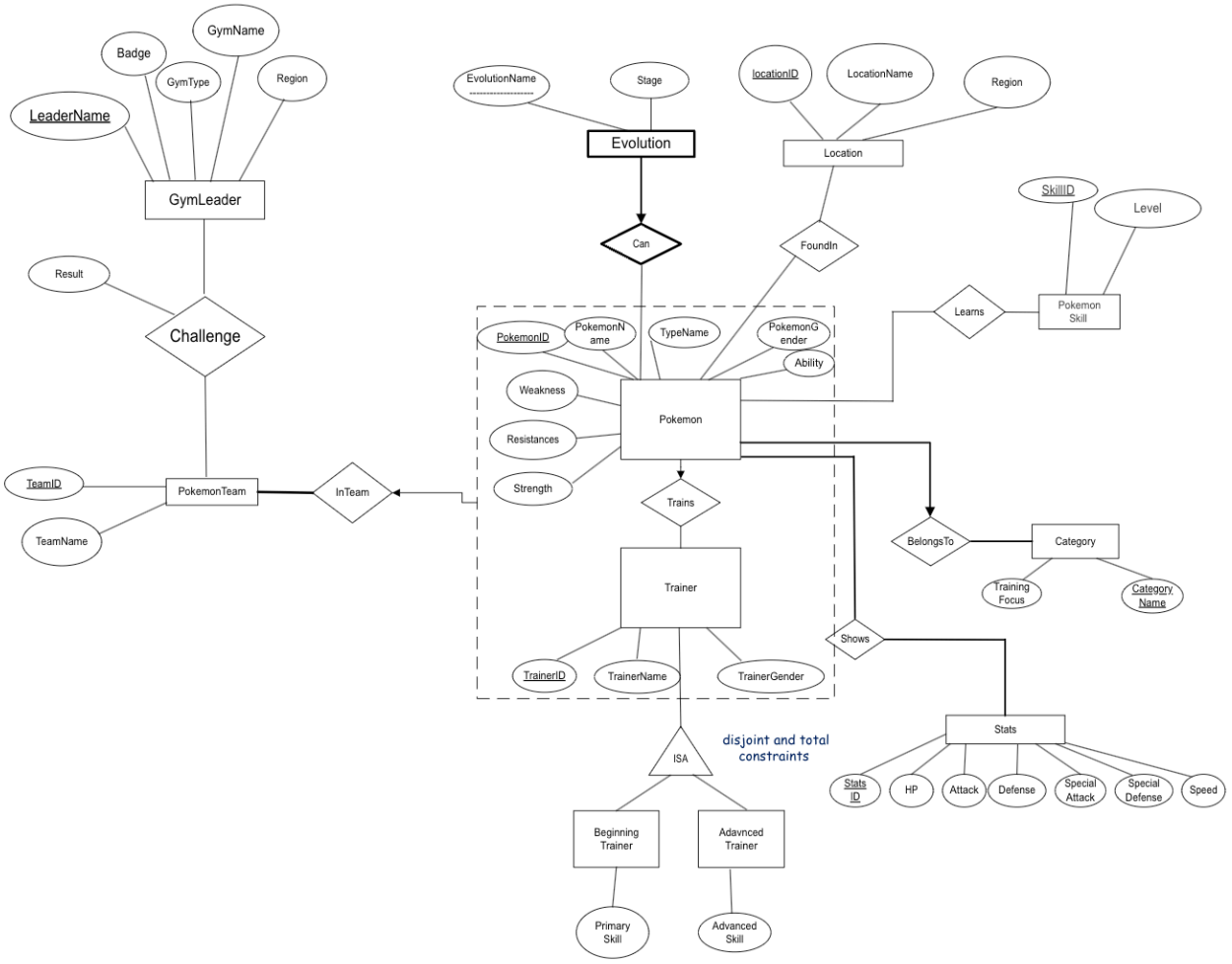
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address,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

2. Project Summary:

Our project focuses on developing a Pokemon data collection and game strategy management system using a relational database. The database models key aspects such as Pokemon species data, performance analysis, game mechanics, and team composition to help trainers optimize their gameplay. Users can store, retrieve, and analyze Pokemon attributes, evolution paths, and battle strategies, enhancing their decision-making in the game.

3. Updates on our ER-diagram



(1) We renamed the three “Has” relationships to “InTeam,” “BelongsTo,” and “Shows” respectively, providing more distinct and descriptive names for better clarity in the database design.

(2) We changed the relationship between Pokemon, Team, and Trainer from a ternary relationship to aggregation.

Initially, we want to use the ternary relationship to express that both each trainer and each Pokemon could only belong to only one team. However, we later realized that this unintentionally made the Pokemon-Trainer relationship one-to-one, which was not our intended design. Since we wanted each trainer to have multiple Pokemon while ensuring that each Pokemon belongs to only one trainer (many-to-one), we switched to aggregation to properly represent this structure.

(3) We are adding a new entity, GymLeader, and establishing a relationship between GymLeader and PokemonTeam. The GymLeader entity has five attributes: LeaderName, Badge, GymType, GymName, and Region, with LeaderName as the primary key. Each team can challenge a GymLeader to earn the corresponding Badge, better reflecting the mechanics of gym battles in the Pokemon world.

(4) We delete the Type entities and convert Type into an attribute of the Pokemon entity. Additionally, Weakness and Resistance will also be attributes of Pokemon because we realize that Weakness and Resistance should not be in the Type table and they are properties of individual Pokemon. This change simplifies the model and better represents these properties. To better express Pokemon property, we also add the Strength attribute to the Pokemon entity.

(5) To avoid ambiguity, we rename the Gender attribute in the Pokemon entity to PokemonGender and the Gender attribute in the Trainer entity to TrainerGender. Similarly, we rename the Name attribute in the Pokemon entity to PokemonName and the Name attribute in the Trainer entity to TrainerName.

4/5. Relational Schema & Functional Dependencies (FDs)

(Use blue font to express non-PK/CK FDs)

Trainer (TrainerID:VARCHAR(20), TrainerName: VARCHAR(20), TrainerGender: VARCHAR(20))
PK: TrainerID
CK: TrainerID
TrainerGender, TrainerName should be NOT NULL
FD:TrainerID → TrainerName, TrainerGender

BeginningTrainer (TrainerID:VARCHAR(20), PrimarySkill: VARCHAR(20))
PK: TrainerID
CK: TrainerID
FD:TrainerID → PrimarySkill

AdvancedTrainer (TrainerID:VARCHAR(20),AdvancedSkill: VARCHAR(20))
PK: TrainerID
CK: TrainerID
AdvancedSkill should be NOT NULL
FD:TrainerID → AdvancedSkill

PokemonTrains (PokemonID: VARCHAR(20), PokemonName:VARCHAR(20),
Weakness: VARCHAR(20), Resistance: VARCHAR(20),
TypeName:VARCHAR(20), PokemonGender:VARCHAR(20)
Ability: VARCHAR(20), **TrainerID**:VARCHAR(20),
Strength: VARCHAR(20)) reference Trainer
PK: PokemonID
CK:PokemonID, PokemonName (PokemonName should be UNIQUE and NOT NULL)
FK:TrainerID should be NOT NULL and ON DELETE CASCADE
(ensures that when a TrainerID is deleted from Trainer, all their associated records are automatically removed, preventing dangling records and maintaining referential integrity)
TypeName should be NOT NULL
FDs:
PokemonID → PokemonName, PokemonGender, Ability, TrainerID, Weakness, TypeName, Resistance, Strength
TypeName → Weakness
TypeName → Resistance
TypeName → Strength

InTeam (**TeamID**:VARCHAR(20), **PokemonID**:VARCHAR(20)) reference PokemonTeam, PokemonTrains
PK: TeamID, PokemonID
CK: TeamID, PokemonID
FK: TeamID, PokemonID
TeamID is both PK and FK, should be ON DELETE CASCADE
(ensures that when a TeamID is deleted from PokemonTeam, all their associated records are automatically removed, preventing dangling records and maintaining referential integrity)
PokemonID is both PK and FK, should be ON DELETE CASCADE
(ensures that when a PokemonID is deleted from PokemonTrains, all their associated records are automatically removed, preventing dangling records and maintaining referential integrity)
TeamID, PokemonID → (no other attributes)

BelongsTo (**PokemonID**:VARCHAR(20), **CategoryName**:VARCHAR(20)) reference PokemonTrains
PK: PokemonID, CategoryName
CK: PokemonID, CategoryName
FK: PokemonID, CategoryName
CategoryName is both PK and FK, ON DELETE CASCADE
(ensures that when a CategoryName is deleted from PokemonTrains, all their associated records in BelongsTo are automatically removed, preventing dangling records and maintaining referential integrity)
PokemonID is both PK and FK, ON DELETE CASCADE

(ensures that when a PokemonID is deleted from PokemonTrains, all their associated records in BelongsTo are automatically removed, preventing dangling records and maintaining referential integrity)
CategoryName, PokemonID → (no other attributes)

PokemonTeam (TeamID:VARCHAR(20), TeamName:VARCHAR(20))
PK:TeamID
CK:TeamID
TeamName should be NOT NULL
FD:TeamID → TeamName

EvolutionCan (EvolutionName:VARCHAR(20), Stage:INTEGER, **PokemonID**:VARCHAR(20)) reference PokemonTrains, InTeam
PK:EvolutionName, PokemonID
CK:EvolutionNam, PokemonID
FK:PokemonID
PokemonID is both PK and FK, use ON DELETE CASCADE
(ensures that when a PokemonID is deleted from PokemonTrains, all their associated records in EvolutionCan are automatically removed, preventing dangling records and maintaining referential integrity)
FD:EvolutionName, PokemonID → Stage

FoundIn(**PokemonID**:VARCHAR(20), **LocationID**:VARCHAR(20)) reference PokemonTrains, Location
PK: PokemonID, LocationID
CK:PokemonID, LocationID
FK:PokemonID, LocationID
PokemonID is both PK and FK, use ON DELETE CASCADE
(ensures that when a PokemonID is deleted from PokemonTrains, all their associated records in FoundIn are automatically removed, preventing dangling records and maintaining referential integrity)
LocationID is both PK and FK, use ON DELETE SET DEFAULT to be “unknown location”
(This ensures that Pokemon records are preserved even if the location they were found in is removed. example: If a location like "Safari Zone" is deleted, Pokemon found there should not be removed, their location should be updated to "unknown location")
(PokemonID, LocationID) → (no other attributes)

Location(LocationID:VARCHAR(20), LocationName:VARCHAR(20), Region:VARCHAR(20))
PK: LocationID
CK:LocationID
Region should be NOT NULL
LocationName should be NOT NULL
FD:LocationID → LocationName, Region

PokemonSkill(SkillID:VARCHAR(20), Level:INTEGER)
PK: SkillID
CK:SkillID
FD:SkillID → Level

Learns(**SkillID**:VARCHAR(20), **PokemonID**:VARCHAR(20))
reference PokemonTrains, PokemonSkill
PK:SkillID, PokemonID
CK:SkillID, PokemonID
FK:SkillID, PokemonID
PokemonID is both PK and FK, ON DELETE CASCADE
(ensures that when a PokemonID is deleted from PokemonTrains, all their associated records in Learns are automatically removed, preventing dangling records and maintaining referential integrity)
SkillID is both PK and FK, ON DELETE CASCADE
(ensures that when a SkillID is deleted from PokemonSkill, all their associated records in Learns are automatically removed, preventing dangling records and maintaining referential integrity)
(SkillID, PokemonID) → (no other attributes)

Category(TrainingFocus: VARCHAR(20), CategoryName:VARCHAR(20))
PK:CategoryName
CK:CategoryName
FD:CategoryName → TrainingFocus

Shows(**PokemonID**:VARCHAR(20), **StatsID**:VARCHAR(20)) reference PokemonTrains, Stats
PK:PokemonID, StatsID
CK:PokemonID, StatsID
FK:PokemonID, StatsID
PokemonID is both PK and FK, ON DELETE CASCADE
(ensures that when a PokemonID is deleted from PokemonTrains, all their associated records in Shows are automatically removed, preventing dangling records and maintaining referential integrity)
StatsID is both PK and FK, use ON DELETE CASCADE
(ensures that when a StatsID is deleted from Stats, all their associated records in Shows are automatically removed, preventing dangling records and maintaining referential integrity)
(PokemonID, StatsID) → (no other attributes)

Stats(StatsID:VARCHAR(20), HP: INTEGER, Attack:INTEGER, Defense:INTEGER, SpecialAttack:INTEGER, SpecialDefense:INTEGER, Speed:INTEGER)
PK:StatsID
CK:StatsID
FDs:HP, Attack, Defense, SpecialAttack, SpecialDefense, Speed should be NOT NULL
StatsID → HP, Attack, Defense, SpecialAttack, SpecialDefense, Speed

Challenge(Result: BOOLEAN, **LeaderName**: VARCHAR(20), **TeamID**:VARCHAR(20))
reference GymLeader, PokemonTeam
PK:LeaderName, TeamID
CK:LeaderName, TeamID
FK:LeaderName, TeamID
LeaderName is both PK and FK, ON DELETE CASCADE
(ensures that when a LeaderName is deleted from GymLeader, all their associated records in Challenge are automatically removed, preventing dangling records and maintaining referential integrity)
TeamID is both PK and FK, ON DELETE CASCADE
(ensures that when a TeamID is deleted from GymLeader, all their associated records in Challenge are automatically removed, preventing dangling records and maintaining referential integrity)
FDs:LeaderName, TeamID → Result

GymLeader(GymID: VARCHAR(20), LeaderName: VARCHAR(20), Badge: VARCHAR(20), GymType: VARCHAR(20), GymName: VARCHAR(20), Region: VARCHAR(20))
PK:LeaderName
CK:LeaderName
GymName should be NOT NULL
GymType should be NOT NULL
Region should be NOT NULL
Badge should be NOT NULL
FDs:LeaderName → Badge, GymType, GymName, Region
GymType, Region → Badge
GymName → Region
GymName → Badge

6. Normalization

Since all other tables are already in 3NF, we only need to decompose **GymLeader** and **PokemonTrains** to ensure they meet 3NF.

GymLeader (LeaderName, GymName, Badge, Region, GymType), The minimal key is LeaderName and the minimal cover is:

1. LeaderName → GymType

2. LeaderName → GymName

3. GymName → Badge

4. GymName → Region

5. GymType, Region → Badge

GymLeader (LeaderName, GymName, Badge, Region, GymType)

FD3 violates BCNF, decompose R to R1(GymName, Badge), R'(LeaderName, GymName, Type, Region)

FD4 violates BCNF, decompose R' to R2(GymName, Region), R3(LeaderName,GymName, Type)

To preserve the FD5 , we add R4(Type, Region, Badge)

Therefore, we have decomposed R into R1, R2, R3, R4.

R1 and R2 have the EXACT same key, we combined R1 with R2 into one table: RegionTypeBadge(Region, Type, Badge).
Also, we rename R4 and R3 to RegionTypeBadge(Region, Type, Badge) and GymLeader(LeaderName, Type, **GymName**) separately.

Finally, we get 3 tables:

(1) Gym(GymName, Badge, Region)

PK: GymName CK: GymName

(2) RegionTypeBadge(Region, Type, Badge)

PK: Region, Type CK: Region, Type

(3) GymLeader(LeaderName, Type, **GymName**) GymName reference Gym

PK: LeaderName, CK: LeaderName, FK: GymName reference Gym

PokemonTrains(PokemonID, PokemonName, TypeName, PokemonGender, Ability, Weakness, Resistance, Strength, **TrainerID**), The minimal key is PokemonID and the minimal cover is:

1. PokemonID → PokemonName

2. PokemonID → Type

3. PokemonID → PokemonGender

4. PokemonID → Ability

5. PokemonID → TrainerID

6. TypeName → Resistance

7. TypeName → Weakness

8. TypeName → Strength

R(PokemonID, PokemonName, TypeName, PokemonGender, Ability, Weakness, Resistance, Strength, **TrainerID**)

FD6 violates BCNF, decompose R to R'(PokemonID, PokemonName, TypeName, PokemonGender, Ability, Weakness, Strength, **TrainerID**) , R1(TypeName, Resistance)

FD7 violate BCNF, decompose R' to R2(TypeName, Weakness), R''(PokemonID, PokemonName, TypeName, PokemonGender, Ability, Strength, TrainerID)

FD8 violate BCNF, decompose R'' to R3(TypeName, Strength), R4(PokemonID, PokemonName, TypeName, PokemonGender, Ability, **TrainerID**)

Therefore, we have decomposed R into R1, R2, R3, R4

Because R1, R2, R3 have the EXACT same key, we combined R1, R2, R3 into one table.

Finally, we get 2 tables:

(1) PokemonTrains(PokemonID, PokemonName, **TypeName**, PokemonGender, Ability, **TrainerID**) TrainerID reference Trainer, TypeName reference Type

PK:PokemonID CK:PokemonID FK: TypeName reference Type

(2) Type (TypeName, Weakness, Resistance, Strength)

PK:TypeName CK:TypeName

7. The SQL DDL statements

Trainer (TrainerID:VARCHAR(20), TrainerName: VARCHAR(20), TrainerGender: VARCHAR(20))

```
CREATE TABLE Trainer (
   TrainerID VARCHAR(20),
   TrainerName VARCHAR(20) NOT NULL,
   TrainerGender VARCHAR(20) NOT NULL,
   PRIMARY KEY (TrainerID)
);

INSERT INTO Trainer (TrainerID, TrainerName, TrainerGender)
VALUES
   ('1', 'Raihan', 'F'),
   ('2', 'Clair', 'M'),
   ('3', 'Alain', 'M'),
   ('4', 'Iris', 'M'),
   ('5', 'Diantha', 'F'),
   ('6', 'Cynthia', 'F');
```

BeginningTrainer (TrainerID:VARCHAR(20), PrimarySkill: VARCHAR(20))

```
CREATE TABLE BeginningTrainer (
   TrainerID VARCHAR(20),
   PrimarySkill VARCHAR(20),
   PRIMARY KEY (TrainerID)
);

INSERT INTO BeginningTrainer (TrainerID, PrimarySkill)
VALUES
   ('8', 'Combat'),
   ('10', NULL),
   ('11', 'Survival'),
   ('12', NULL),
   ('13', 'Stealth'),
   ('14', NULL);
```

AdvancedTrainer (TrainerID:VARCHAR(20),AdvancedSkill: VARCHAR(20))

```
CREATE TABLE AdvancedTrainer (
    TrainerID VARCHAR(20),
    AdvancedSkill VARCHAR(20) NOT NULL,
    PRIMARY KEY (TrainerID)
);

INSERT INTO AdvancedTrainer (TrainerID, AdvancedSkill)
VALUES
    ('1', 'Charm'),
    ('2', 'Focus'),
    ('3', 'Intuition'),
    ('4', 'Perception'),
    ('5', 'Medicine'),
    ('6', 'Command');
```

InTeam (**TeamID**:VARCHAR(20), **PokemonID**:VARCHAR(20)) reference PokemonTeam, PokemonTrains

```
CREATE TABLE InTeam (
    TeamID VARCHAR(20),
    PokemonID VARCHAR(20),
    PRIMARY KEY (TeamID, PokemonID),
    FOREIGN KEY (TeamID) REFERENCES PokemonTeam(TeamID)
    ON DELETE CASCADE,
    FOREIGN KEY (PokemonID) REFERENCES Pokemon(PokemonID)
    ON DELETE CASCADE
);

INSERT INTO InTeam (TeamID, PokemonID)
VALUES
    ('1', '0175'),
    ('2', '0004'),
    ('3', '0005'),
    ('4', '0039'),
    ('5', '0025'),
    ('6', '0007');
```

BelongsTo (**PokemonID**:VARCHAR(20), **CategoryName**:VARCHAR(20)) reference PokemonTrains

```
CREATE TABLE BelongsTo (
 CategoryName VARCHAR(20),
 PokemonID VARCHAR(20),
 PRIMARY KEY (CategoryName, PokemonID),
 FOREIGN KEY (CategoryName) REFERENCES Category(CategoryName)
 ON DELETE CASCADE,
  FOREIGN KEY (PokemonID) REFERENCES Pokemon(PokemonID) ON
DELETE CASCADE
);

INSERT INTO BelongsTo (CategoryName, PokemonID)
VALUES
   ('Spike Ball', '0175'),
   ('Lizard', '0004'),
   ('Flame', '0005'),
   ('Balloon', '0039'),
   ('Mouse', '0025'),
   ('Tiny Turtle', '0007');
```

Gym(GymName, Badge, Region)

```
CREATE TABLE Gym (
   GymName VARCHAR(20) PRIMARY KEY,
   Badge VARCHAR(20) NOT NULL,
   Region VARCHAR(20) NOT NULL
);

INSERT INTO Gym (GymName, Badge, Region)
VALUES
   ('Pewter Gym', 'Boulder Badge', 'Kanto'),
   ('Cerulean Gym', 'Cascade Badge', 'Kanto'),
   ('Nacrene Gym', 'Basic Badge', 'Unova'),
   ('Aspertia Gym', 'Basic Badge', 'Unova'),
   ('Hulbury Stadium', 'Water Badge', 'Galar'),
   ('Cascarrafa Gym', 'Water Badge', 'Paldea');
```

RegionTypeBadge(Region, Type, Badge)

```
CREATE TABLE RegionTypeBadge (
   Region VARCHAR(20),
   Type VARCHAR(20),
   Badge VARCHAR(20) NOT NULL,
   PRIMARY KEY (Region, Type)
);

INSERT INTO RegionTypeBadge (Region, Type, Badge)
VALUES
   ('Kanto', 'Rock', 'Boulder Badge'),
   ('Kanto', 'Water', 'Cascade Badge'),
   ('Hoenn', 'Rock', 'Stone Badge'),
   ('Unova', 'Grass', 'Trio Badge'),
   ('Unova', 'Water', 'Trio Badge'),
   ('Unova', 'Fire', 'Trio Badge'),
   ('Unova', 'Normal', 'Basic Badge');
```

GymLeader(LeaderName, Type, **GymName**) GymName reference Gym

```
CREATE TABLE GymLeader (
   LeaderName VARCHAR(20) PRIMARY KEY,
   Type VARCHAR(20) NOT NULL,
   GymName VARCHAR(20) NOT NULL,
   FOREIGN KEY (GymName) REFERENCES Gym(GymName)
);

INSERT INTO GymLeader (LeaderName, Type, GymName)
VALUES
   ('Cilan', 'Grass', 'Striaton Gym'),
   ('Chili', 'Fire', 'Striaton Gym'),
   ('Cress', 'Water', 'Striaton Gym'),
   ('Nessa', 'Water', 'Hulbury Stadium'),
   ('Koga', 'Poison', 'Fuchsia Gym'),
   ('Janine', 'Poison', 'Fuchsia Gym');
```

PokemonTrains(PokemonID, PokemonName, **TypeName**, PokemonGender, Ability, **TrainerID**)
TrainerID reference Trainer, TypeName reference Type

```
CREATE TABLE PokemonTrains (
   PokemonID VARCHAR(20) PRIMARY KEY,
   PokemonName VARCHAR(20) NOT NULL UNIQUE,
   TypeName VARCHAR(20) NOT NULL,
   PokemonGender VARCHAR(20),
   Ability VARCHAR(20),
   TrainerID VARCHAR(20) NOT NULL,
   FOREIGN KEY (TypeName) REFERENCES Type(TypeName) ON
DELETE CASCADE
);

INSERT INTO PokemonTrains (PokemonID, PokemonName, TypeName,
PokemonGender,
Ability, TrainerID)
VALUES
   ('0175', 'Togepi', 'Fairy', 'F/M', 'Hustle', '1'),
   ('0004', 'Charmander', 'Fire', 'F/M', 'Blaze', '2'),
   ('0007', 'Squirtle', 'Water', 'F/M', 'Torrent', '3'),
   ('0025', 'Pikachu', 'Electric', 'F/M', 'Static', '3'),
   ('0035', 'Clefairy', 'Fairy', 'F/M', 'Cute Charm', '4'),
   ('0039', 'Jigglypuff', 'Fairy', 'F/M', 'Cute Charm', '4');
```

Type (TypeName, Weakness, Resistance, Strength)

```
CREATE TABLE Type (
   TypeName VARCHAR(20) PRIMARY KEY,
   Weakness VARCHAR(20),
   Resistance VARCHAR(20),
   Strength VARCHAR(20)
);

INSERT INTO Type (TypeName, Weakness, Resistance, Strength)
VALUES
   ('Grass', 'Fire', 'Water', 'Rock'),
   ('Fire', 'Water', 'Steel', 'Bug'),
   ('Water', 'Electric', 'Ice', 'Fire'),
   ('Normal', 'Fighting', 'Ghost', NULL),
```

```
    ('Fighting', 'Flying', 'Bug', 'Normal'),
    ('Ghost', 'Dark', 'Poison', 'Psychic');
```

PokemonTeam (TeamID:VARCHAR(20), TeamName:VARCHAR(20))

```
CREATE TABLE PokemonTeam (
    TeamID VARCHAR(20) PRIMARY KEY,
    TeamName VARCHAR(20) NOT NULL
);

INSERT INTO PokemonTeam (TeamID, TeamName)
VALUES
    ('0001', 'Alpha'),
    ('0002', 'Beta'),
    ('0003', 'Gamma'),
    ('0004', 'Delta'),
    ('0011', 'Doota'),
    ('0005', 'Epsilon');
```

EvolutionCan (EvolutionName:VARCHAR(20), Stage:INTEGER, **PokemonID**:VARCHAR(20)) reference PokemonTrains, InTeam

```
CREATE TABLE EvolutionCan (
    EvolutionName VARCHAR(20),
    Stage INTEGER,
    PokemonID VARCHAR(20),
    PRIMARY KEY (EvolutionName, PokemonID),
    FOREIGN KEY (PokemonID) REFERENCES Pokemon(PokemonID) ON
DELETE CASCADE
);

INSERT INTO EvolutionCan (EvolutionName, Stage, PokemonID)
VALUES
    ('Mega Venusaur', 1, '0003'),
    ('Gigantamax Venusaur', 2, '0003'),
    ('Mega Charizard X', 1, '0006'),
     ('Mega Charizard Y', 2, '0006'),
    ('Mega Charizard Z', 2, '0011'),
    ('Gigantamax Charizard', 3, '0006');
```

Location(LocationID:VARCHAR(20), LocationName:VARCHAR(20), Region:VARCHAR(20))

```
CREATE TABLE Location (
   LocationID VARCHAR(20) PRIMARY KEY,
   LocationName VARCHAR(20) NOT NULL,
   Region VARCHAR(20) NOT NULL UNIQUE
);

INSERT INTO Location (LocationID, LocationName, Region)
VALUES
   ('0001', 'Route 1', 'Kanto'),
   ('0002', 'Route 2', 'Kanto'),
   ('0090', 'Water Path', 'Kanto'),
   ('0091', 'Route 29', 'Johto'),
  ('0021', 'Route 30', ‘Kyoto),
   ('0351', 'Route 1', 'Unova');
```

FoundIn(**PokemonID**:VARCHAR(20), **LocationID**:VARCHAR(20)) reference PokemonTrains, Location

```
CREATE TABLE FoundIn (
   PokemonID VARCHAR(20),
   LocationID VARCHAR(20) DEFAULT 'UnknownLoc',
   PRIMARY KEY (PokemonID, LocationID),
   FOREIGN KEY (PokemonID) REFERENCES
PokemonTrains(PokemonID) ON DELETE CASCADE,
   FOREIGN KEY (LocationID) REFERENCES Location(LocationID) ON
DELETE SET DEFAULT
);

INSERT INTO FoundIn (PokemonID, LocationID)
VALUES
   ('0016', '0001'),
   ('0019', '0001'),
   ('0016', '0002'),
   ('0311', '0091'),
   ('0321', '0071'),
   ('0016', '0091');
```

PokemonSkill(SkillID:VARCHAR(20), Level:INTEGER)

```
CREATE TABLE PokemonSkill (
   SkillID VARCHAR(20) PRIMARY KEY,
   SkillLevel INTEGER
);
```

```
INSERT INTO PokemonSkill (SkillID, SkillLevel)
VALUES
   ('0002', 1),
  ('0006', 21),
   ('0003', 1),
   ('0004', 20),
   ('0005', 32),
   ('0007', 28);
```

Learns(**SkillID**:VARCHAR(20), **PokemonID**:VARCHAR(20))

```
CREATE TABLE Learn (
   SkillID VARCHAR(20),
   PokemonID VARCHAR(20),
   PRIMARY KEY (SkillID, PokemonID),
   FOREIGN KEY (PokemonID) REFERENCES
PokemonTrains(PokemonID) ON DELETE CASCADE,
   FOREIGN KEY (SkillID) REFERENCES PokemonSkill(SkillID) ON
DELETE CASCADE
);

INSERT INTO Learn (SkillID, PokemonID)
VALUES
   ('0002', '0043'),
   ('0002', '0044'),
   ('0002', '0045'),
   ('0003', '0745'),
   ('0830', '0745'),
   ('0880', '0795');
```

Category(TrainingFocus: VARCHAR(20), CategoryName:VARCHAR(20))

```
CREATE TABLE Category (
   CategoryName VARCHAR(20) PRIMARY KEY,
   TrainingFocus VARCHAR(20)
);

INSERT INTO Category (CategoryName, TrainingFocus)
VALUES
   ('Seed Pokemon', 'Grass-based Tactics'),
   ('Bird Pokemon', 'Aerial Maneuvers'),
   ('Mouse Pokemon', 'Speed and Agility'),
   ('Fox Pokemon', 'Illusion Mastery'),
   ('Bat Pokemon', 'Echolocation Tactics');
```

Shows(**PokemonID**:VARCHAR(20), **StatsID**:VARCHAR(20)) reference PokemonTrains, Stats

```
CREATE TABLE Shows (
   PokemonID VARCHAR(20),
   StatsID VARCHAR(20),
   PRIMARY KEY (PokemonID, StatsID),
   FOREIGN KEY (PokemonID) REFERENCES PokemonTrains
(PokemonID) ON DELETE CASCADE,
   FOREIGN KEY (StatsID) REFERENCES Stat(StatsID) ON DELETE
CASCADE
);

INSERT INTO Shows (PokemonID, StatsID)
VALUES
   ('0001', '0001'),
   ('0002', '0002'),
   ('0003', '0003'),
   ('0008', '0004'),
   ('0088', '0005'),
   ('0095', '0006');
```

Stats(StatsID:VARCHAR(20), HP: INTEGER, Attack:INTEGER, Defense:INTEGER, SpecialAttack:INTEGER, SpecialDefense:INTEGER, Speed:INTEGER)

```
CREATE TABLE Stats (
   StatsID VARCHAR(20) PRIMARY KEY,
   HP INTEGER,
   Attack INTEGER,
   Defense INTEGER,
   SpecialAttack INTEGER,
   SpecialDefense INTEGER,
   Speed INTEGER
);

INSERT INTO Stats (StatsID, HP, Attack, Defense, SpecialAttack,
SpecialDefense, Speed)
VALUES
   ('0001', 3, 3, 3, 4, 4, 3),
   ('0002', 4, 4, 4, 5, 5, 4),
   ('0003', 5, 5, 5, 6, 6, 5),
   ('0004', 3, 4, 3, 4, 3, 4),
   ('0009', 9, 4, 3, 4, 3, 4),
   ('0005', 4, 4, 4, 5, 4, 5);
```

Challenge(Result: BOOLEAN, **<u>LeaderName</u>**: VARCHAR(20), **<u>TeamID</u>**:VARCHAR(20)) reference GymLeader, PokemonTeam

```
CREATE TABLE Challenge (
   Result NUMBER(1), -- Replaced BOOLEAN with NUMBER(1)
   LeaderName VARCHAR(20),
   TeamID VARCHAR(20),
   PRIMARY KEY (LeaderName, TeamID),
   FOREIGN KEY (LeaderName) REFERENCES
   GymLeader(LeaderName),
   FOREIGN KEY (TeamID) REFERENCES PokemonTeam(TeamID)
);

INSERT INTO Challenge (Result, LeaderName, TeamID)
VALUES
   (1, 'Brock', '0001'),
   (1, 'Brock', '0002'),
   (0, 'Brock', '0003'),
   (1, 'Molly', '0009'),
   (1, 'Misty', '0003'),
   (1, 'Misty', '0001');
```

8. AI tools acknowledgment

We used Grammarly to improve sentence fluency and correct syntax mistakes.