University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2
Date: March 2, 2025
Group Number: 50

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Maya Dong	37406162	c0w0b	maya.mengya@gmail.com
Leo Kim	33416751	w4r0g	leo.k0922@gmail.com
Alan Wang	28413145	d9o7h	yinghsu@student.ubc.ca

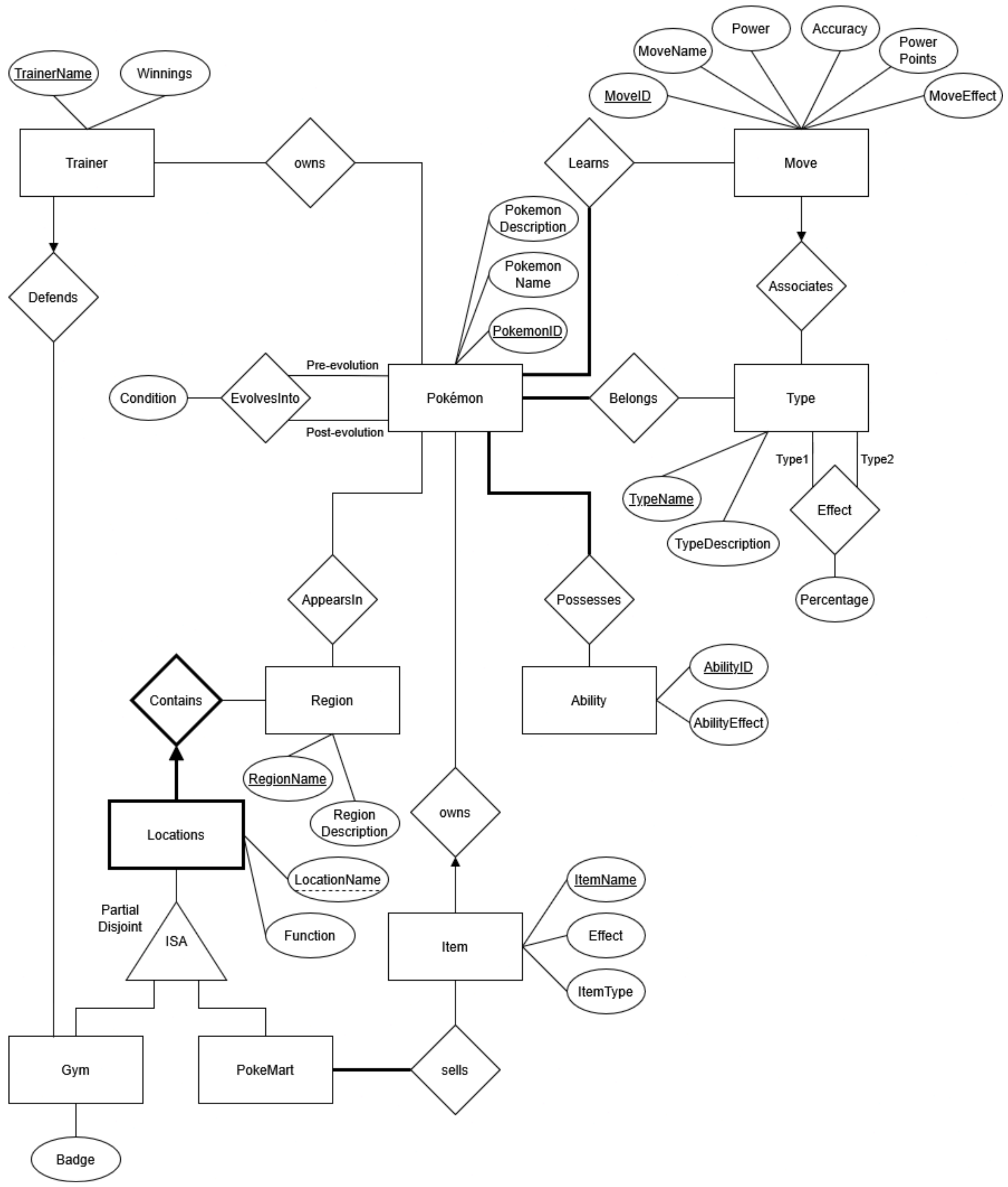
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address,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

Project Description:

The project is a database management system that organizes and catalogs Pokémon data. The domain of the project focuses on the Pokémon universe, specifically the various species, their moves, abilities, evolutions, and regional differences. It is designed for players and enthusiasts who want to explore detailed information about Pokémon, their interactions, and their evolution paths across different game versions and regions.

Declaration of AI tools: We did not use any AI tools for this portion of the project.



Modifications & Explanation:

- Move-Type relation was 1 to many, changed to many to 1 (because each Move can only have 1 Type, and each type can have many Moves)
- Pokemon evolution from many to 1 to many to many (Different pokemon can evolve into many different pokemons)
- Added relation Effect to entity Type to indicate how effective is one type to another (removed strength/weakness to type since each type can be strong and weak to multiple other types)

Schema

Primary keys: underlined
Foreign keys: **bold**

Entity/Relations	**Definition**
Pokemon	Pokemon(PokemonID: INTEGER, PokemonDescription: VARCHAR, PokemonName: VARCHAR) Candidate Key: PokemonName
Learns	Learns(**PokemonID: INTEGER, MoveID: INTEGER**) Constraints: MoveID is NOT NULL
Move_Associates	Move_Associates(MoveID: INTEGER, MoveName: VARCHAR, Power: INTEGER, Accuracy: INTEGER, PowerPoints: INTEGER, MoveEffect: VARCHAR, **TypeName: VARCHAR**) Candidate Key: MoveName
Type	Type(TypeName: VARCHAR, TypeDescription: VARCHAR)
Effect	Effect(**TypeName1: VARCHAR**, **TypeName2: VARCHAR**, Percentage: INTEGER)
Belongs	Belongs(**PokemonID: INTEGER, TypeName: VARCHAR)** Constraints: TypeName is NOT NULL
Possesses	Possesses(**PokemonID: INTEGER, AbilityID: INTEGER)** Constraints: AbilityID is NOT NULL
Ability	Ability(AbilityID: INTEGER, AbilityEffect: VARCHAR)
Item_Owns	Item_Owns(ItemName: VARCHAR, ItemEffect: VARCHAR, ItemType: VARCHAR, **PokemonID: INTEGER**)
Sells	Sells(**ItemName: VARCHAR**, **LocationName: VARCHAR, RegionName: VARCHAR**) Constraints: ItemName is NOT NULL

Pokemart	Pokemart(**LocationName: VARCHAR, RegionName: VARCHAR)**
Gym	Gym(**LocationName: VARCHAR**, **RegionName: VARCHAR,** Badge: VARCHAR)
Trainer_Defends	Trainer_Defends(TrainerName: VARCHAR, Winnings: INTEGER, **LocationName: VARCHAR, RegionName: VARCHAR**)
Location	Location(LocationName: VARCHAR, **RegionName: VARCHAR**, Function: VARCHAR) Constraints: RegionName is NOT NULL
Region	Region(RegionName: VARCHAR, RegionDescription: VARCHAR)
AppearsIn	AppearsIn(**RegionName: VARCHAR, PokemonID: INTEGER**)
Owns (for Trainer-Pokemon Relation)	Owns(**TrainerName: VARCHAR**, **PokemonID: INTEGER**)
EvolvesInto	EvolvesInto(**PreEvolutionID: INTEGER**, **PostEvolutionID: INTEGER**, Condition: VARCHAR)

Functional Dependencies

Entity/Relations	Functional Dependencies
Pokemon	PokemonID -> PokemonName, PokemonDescription PokemonName -> PokemonID, PokemonDescription
Learns	No FDs
Move_Associates	MoveID -> MoveName, Power, Accuracy, PowerPoints, MoveEffect, TypeName MoveName -> MoveID, Power, Accuracy, PowerPoints, MoveEffect, TypeName MoveEffect -> TypeName
Type	TypeName -> TypeDescription
Effect	TypeName1, TypeName2 -> Percentage
Belongs	No FDs
Possesses	No FDs
Ability	AbilityID -> AbilityEffect
Item_Owns	ItemName -> ItemEffect, ItemType, PokemonID ItemEffect -> ItemType
Sells	No FDs
Pokemart	No FDs
Gym	LocationName, RegionName -> Badge
Trainer_Defends	TrainerName -> Winnings, LocationName, RegionName
Location	LocationName -> RegionName, Function
Region	RegionName -> RegionDescription
AppearsIn	No FDs
Owns (for Trainer-Pokemon Relation)	No FDs
EvolvesInto	PreEvolutionID, PostEvolutionID -> Condition

Normalization

Primary keys: underlined
Foreign keys: **bold**

Entity/Relations	Normalization
Pokemon	Pokemon(PokemonID, PokemonDescription, PokemonName) PokemonID -> PokemonName, PokemonDescription PokemonName -> PokemonID, PokemonDescription FD1: PokemonID$^+$ = {PokemonID, PokemonDescription, PokemonName} ● PokemonID is a superkey for Pokemon FD2: PokemonName$^+$ = {PokemonID, PokemonDescription, PokemonName} ● PokemonName is a superkey for Pokemon All FD holds in Pokemon, thus it is in BCNF. Candidate Key: PokemonName
Learns	Learns(**PokemonID, MoveID**) No FDs, thus in BCNF
Move_Associates	Move_Associates(MoveID, MoveName, Power, Accuracy, PowerPoints, MoveEffect, **TypeName**) FD1: MoveID -> MoveName, Power, Accuracy, PowerPoints, MoveEffect, TypeName MoveID $^+$ = {MoveID , MoveName, Power, Accuracy, PowerPoints, MoveEffect, TypeName} ● MoveID is a superkey for Move_Associates FD2: MoveName -> MoveID, Power, Accuracy, PowerPoints, MoveEffect, TypeName MoveName $^+$ = {MoveID , MoveName, Power, Accuracy, PowerPoints, MoveEffect, TypeName} ● MoveName is a superkey for Move_Associates FD3: MoveEffect -> TypeName MoveEffect $^+$ = {MoveEffect, TypeName}

	● MoveEffect is not a superkey for Move_Associates FD3 does not hold for BCNF, thus we decompose on FD3 Move_Associates1(MoveID, MoveName, Power, Accuracy, PowerPoints, **MoveEffect**) Move_Associates2(MoveEffect, **TypeName**) All FD holds in Move_Associates1 & Move_Associates2, thus they are both in BCNF. Candidate Key: MoveName
Type	Type(TypeName, TypeDescription) TypeName -> TypeDescription TypeName $^+$ = {TypeName , TypeDescription} ● TypeName is a superkey for Type All FD holds in Type, thus it is in BCNF.
Effect	Effect(**TypeName1**, **TypeName2**, ~~Effect~~) Percentage TypeName1, TypeName2 -> Percentage TypeName1, TypeName2$^+$ = {TypeName1, TypeName2, Percentage} ● TypeName1, TypeName2 is a superkey for Effect All FD holds in Type, thus it is in BCNF.
Belongs	Belongs(**PokemonID, TypeName)** No FDs, thus in BCNF
Possesses	Possesses(**PokemonID, AbilityID)** No FDs, thus in BCNF
Ability	Ability(AbilityID, AbilityEffect) AbilityID -> AbilityEffect AbilityID $^+$ = {AbilityID , AbilityEffect} ● AbilityID is a superkey for Ability All FD holds in Ability, thus it is in BCNF.

Item_Owns	Item_Owns(ItemName, ItemEffect, ItemType, **PokemonID**) FD1: ItemName -> ItemEffect, ItemType, PokemonID ItemName $^{+}$ = {ItemName , ItemEffect, ItemType} ● ItemName is a superkey for Item_Owns FD2: ItemEffect -> ItemType Effect $^{+}$ = {ItemEffect, ItemType} ● ItemEffect is not a superkey for Item_Owns FD2 does not hold for BCNF, thus we decompose on FD2 Item_Owns1(ItemName, **ItemEffect**, **PokemonID**) Item_Owns2(ItemEffect, ItemType) All FD holds in Item_Owns1& Item_Owns2, thus they are both in BCNF.
Sells	Sells(**ItemName**, **LocationName, RegionName**) No FDs, thus in BCNF
Pokemart	Pokemart(**LocationName, RegionName)** No FDs, thus in BCNF
Gym	Gym(**LocationName**, **RegionName,** Badge) LocationName, RegionName -> Badge LocationName, RegionName $^{+}$ = {LocationName, RegionName, Badge} ● LocationName, RegionName is a superkey for Gym All FD holds in Gym, thus it is in BCNF.
Trainer_Defends	Trainer_Defends(TrainerName, Winnings, **LocationName, RegionName**) TrainerName -> Winnings, LocationName, RegionName TrainerName $^{+}$ = {TrainerName , Winnings, LocationName, RegionName} ● TrainerName is a superkey for Trainer_Defends All FD holds in Trainer_Defends, thus it is in BCNF.
Location	Location(LocationName, **RegionName**, Function) LocationName -> RegionName, Function

	LocationName $^+$ = {LocationName , RegionName, Function} ● LocationName is a superkey for Location All FD holds in Location, thus it is in BCNF.
Region	Region(<u>RegionName,</u> RegionDescription) RegionName -> RegionDescription RegionName $^+$ = {RegionName , RegionDescription} ● RegionName is a superkey for Region All FD holds in Region, thus it is in BCNF.
AppearsIn	AppearsIn(<u>**RegionName,**</u> <u>**PokemonID**</u><u>)</u> No FDs, thus in BCNF
Owns (for Trainer-Pokemon Relation)	Owns(<u>**TrainerName**</u>, <u>**PokemonID**</u>) No FDs, thus in BCNF
EvolvesInto	EvolvesInto(<u>**PreEvolutionID**</u>, <u>**PostEvolutionID**</u>, Condition) PreEvolutionID, PostEvolutionID -> Condition PreEvolutionID, PostEvolutionID$^+$ = {PreEvolutionID ,PostEvolutionID, Condition} ● PreEvolutionID, PostEvolutionID is a superkey for EvolvesInto All FD holds in Region, thus it is in BCNF.

SQL DDL Statements

Entity/Relationships	**SQL DDL Statements**
Pokemon	CREATE TABLE Pokemon (PokemonID INTEGER **PRIMARY KEY**, PokemonDescription VARCHAR, PokemonName VARCHAR **UNIQUE**);
Learns	CREATE TABLE Learns (PokemonID INTEGER, MoveID INTEGER **NOT NULL**, **PRIMARY KEY** (PokemonID, MoveID), **FOREIGN KEY** (PokemonID) **REFERENCES** Pokemon(PokemonID) ON DELETE CASCADE ON UPDATE CASCADE, **FOREIGN KEY** (MoveID) **REFERENCES** Move_Associates1(MoveID) ON DELETE CASCADE ON UPDATE CASCADE,); Note: many-many relationship constraint cannot be modelled with what we learned so far; we use **NOT NULL** to indicate the entity we want for constraint Note2: deleting Pokemon deletes this relation, same for Move
Move_Associates	CREATE TABLE Move_Associates1 (MoveID INTEGER **PRIMARY KEY**, MoveName VARCHAR **UNIQUE**, Power INTEGER, Accuracy INTEGER, PowerPoints INTEGER, MoveEffect VARCHAR, **FOREIGN KEY** (MoveEffect) **REFERENCES** Move_Associates2 (MoveEffect) ON DELETE NO ACTION ON UPDATE CASCADE,); Note: MoveEffect needs to be replaced before deleted CREATE TABLE Move_Associates2 (MoveEffect VARCHAR **PRIMARY KEY**, TypeName VARCHAR, **FOREIGN KEY** (TypeName) **REFERENCES** Type(TypeName) ON DELETE SET NULL

	ON UPDATE CASCADE); Note: Deleting Type keeps Move_Associates2 & 1, as each move can have or not have a type

Type	CREATE TABLE Type(
	TypeName VARCHAR **PRIMARY KEY**,
	TypeDescription VARCHAR
	);

Effect	CREATE TABLE Effect(TypeName1: VARCHAR, TypeName2: VARCHAR, Percentage: INTEGER, **PRIMARY KEY** (TypeName1, TypeName2), **FOREIGN KEY** (TypeName1) **REFERENCES** Type(TypeName) ON DELETE CASCADE ON UPDATE CASCADE, **FOREIGN KEY** (TypeName2) **REFERENCES** Type(TypeName) ON DELETE CASCADE ON UPDATE CASCADE); Note: Deleting a type deletes this relation
Belongs	CREATE TABLE Belongs(PokemonID INTEGER, TypeName VARCHAR **NOT NULL**, **PRIMARY KEY** (PokemonID, TypeName), **FOREIGN KEY** (PokemonID) **REFERENCES** Pokemon(PokemonID) ON DELETE CASCADE ON UPDATE CASCADE, **FOREIGN KEY** (TypeName) **REFERENCES** Type(TypeName) ON DELETE CASCADE ON UPDATE CASCADE); Note: many-many relationship constraint cannot be modelled with what we learned so far; we use **NOT NULL** to indicate the entity we want for constraint
Possesses	CREATE TABLE Possesses(PokemonID INTEGER,

	AbilityID INTEGER **NOT NULL**, **PRIMARY KEY** (PokemonID, AbilityID), **FOREIGN KEY** (PokemonID) **REFERENCES** Pokemon(PokemonID) ON DELETE CASCADE ON UPDATE CASCADE, **FOREIGN KEY** (AbilityID) **REFERENCES** Ability(AbilityID) ON DELETE CASCADE ON UPDATE CASCADE); Note: many-many relationship constraint cannot be modelled with what we learned so far; we use **NOT NULL** to indicate the entity we want for constraint

Ability	CREATE TABLE Ability(
	AbilityID INTEGER **PRIMARY KEY**,
	AbilityEffect VARCHAR
	);

Item_Owns	CREATE TABLE Item_Owns(ItemName: VARCHAR **PRIMARY KEY**, ItemEffect: VARCHAR, ItemType: VARCHAR, PokemonID: INTEGER, **FOREIGN KEY** (PokemonID) **REFERENCES** Pokemon(PokemonID) ON DELETE SET NULL ON UPDATE CASCADE); Note: “ON DELETE SET NULL” because item can exist without a pokemon holding it
Sells	CREATE TABLE Sells(ItemName VARCHAR **NOT NULL**, LocationName VARCHAR, RegionName VARCHAR, **PRIMARY KEY** (ItemName, LocationName, RegionName), **FOREIGN KEY** (ItemName) **REFERENCES** Item(ItemName) ON DELETE CASCADE ON UPDATE CASCADE, **FOREIGN KEY** (LocationName, RegionName) **REFERENCES** Pokemart(LocationName, RegionName) ON DELETE CASCADE

	ON UPDATE CASCADE); Note: deleting an item deletes relation, same for Pokemart Note: many-many relationship constraint cannot be modelled with what we learned so far; we use NOT NULL to indicate the entity we want for constraint
Pokemart	CREATE TABLE Pokemart(LocationName: VARCHAR, RegionName: VARCHAR, **PRIMARY KEY** (LocationName, RegionName), **FOREIGN KEY** (LocationName, RegionName) **REFERENCES** Location(LocationName, RegionName) ON DELETE CASCADE ON UPDATE CASCADE); Note: deleting a location deletes every Pokemart in the location
Gym	CREATE TABLE Gym(LocationName: VARCHAR, RegionName: VARCHAR, Badge: VARCHAR, **PRIMARY KEY** (LocationName, RegionName), **FOREIGN KEY** (LocationName, RegionName) **REFERENCES** Location(LocationName, RegionName) ON DELETE CASCADE ON UPDATE CASCADE); Note: deleting a location deletes every gym in the location
Trainer_Defends	CREATE TABLE Trainer_Defends(TrainerName: VARCHAR **PRIMARY KEY**, Winnings: INTEGER, LocationName: VARCHAR, RegionName: VARCHAR, **FOREIGN KEY** (LocationName, RegionName) **REFERENCES** Gym(LocationName, RegionName) ON DELETE SET NULL ON UPDATE CASCADE); Note: deleting a gym deletes sets NULL to trainer's relation to gym, since a trainer don't have to be defending a gym
Location	CREATE TABLE Location(LocationName: VARCHAR, RegionName: VARCHAR, Function: VARCHAR,

	PRIMARY KEY (LocationName, RegionName), **FOREIGN KEY** (RegionName) **REFERENCES** Region(RegionName) ON DELETE CASCADE ON UPDATE CASCADE); Note: deleting a Region deletes every Location in the Region

Region	CREATE TABLE Region(
	RegionName: VARCHAR **PRIMARY KEY**,
	RegionDescription: VARCHAR
	);

AppearsIn	CREATE TABLE AppearsIn(RegionName: VARCHAR, PokemonID: INTEGER, **PRIMARY KEY** (RegionName, PokemonID), **FOREIGN KEY** (RegionName) **REFERENCES** Region(RegionName) ON DELETE CASCADE ON UPDATE CASCADE, **FOREIGN KEY** (PokemonID) **REFERENCES** Pokemon(PokemonID) ON DELETE CASCADE ON UPDATE CASCADE); Note: deletes relation if pokemon is deleted, same for region
Owns (for Trainer-Pokemon Relation)	CREATE TABLE Owns (TrainerName: VARCHAR, PokemonID: INTEGER, **PRIMARY KEY** (TrainerName, PokemonID), **FOREIGN KEY** (TrainerName) **REFERENCES** Trainer(TrainerName) ON DELETE CASCADE ON UPDATE CASCADE, **FOREIGN KEY** (PokemonID) **REFERENCES** Pokemon(PokemonID) ON DELETE CASCADE ON UPDATE CASCADE); Note: deletes relation if pokemon is deleted, same for Trainer
EvolvesInto	CREATE TABLE EvolvesInto(

	PreEvolutionID: INTEGER, PostEvolutionID: INTEGER, Condition: VARCHAR **PRIMARY KEY** (PreEvolutionID, PostEvolutionID), **FOREIGN KEY** (PreEvolutionID) **REFERENCES** Pokemon(PokemonID) ON DELETE CASCADE ON UPDATE CASCADE, **FOREIGN KEY** (PostEvolutionID) **REFERENCES** Pokemon(PokemonID) ON DELETE CASCADE ON UPDATE CASCADE); Note: deletes relation if pokemon is deleted

SQL Inserts

Pokemon	
1	INSERT INTO Pokemon(PokemonID, PokemonDescription, PokemonName) VALUES (1, 'For some time after its birth, it uses the nutrients that are packed into the seed on its back in order to grow. ', 'Bulbasaur')
2	INSERT INTO Pokemon(PokemonID, PokemonDescription, PokemonName) VALUES (2, 'It has a preference for hot things. When it rains, steam is said to spout from the tip of its tail.', 'Charmander')
3	INSERT INTO Pokemon(PokemonID, PokemonDescription, PokemonName) VALUES (3, 'When its huge eyes light up, it leans forward and rams into its foe at full speed.', 'Squirtle')
4	INSERT INTO Pokemon(PokemonID, PokemonDescription, PokemonName) VALUES (4, 'Its plant blooms when it is absorbing solar energy. It stays on the move to seek sunlight.', 'Ivysaur')
5	INSERT INTO Pokemon(PokemonID, PokemonDescription, PokemonName) VALUES (5, 'Spits fire that is hot enough to melt boulders. Known to cause forest fires unintentionally.', 'Charizard')

Learns	
1	INSERT INTO Learns(PokemonID, MoveID) VALUES (1, 103)
2	INSERT INTO Learns(PokemonID, MoveID) VALUES (2, 101)
3	INSERT INTO Learns(PokemonID, MoveID) VALUES (3, 102)
4	INSERT INTO Learns(PokemonID, MoveID) VALUES (4, 103)
5	INSERT INTO Learns(PokemonID, MoveID)

	VALUES (5, 101)

Move_Associates1	

1	INSERT
	INTO Move_Associates1(MoveID, MoveName, Power, Accuracy, PowerPoints, MoveEffect)
	VALUES (101, 'Flamethrower', 90, 100, 15, 'Burns opponent')

2	INSERT INTO Move_Associates1(MoveID, MoveName, Power, Accuracy, PowerPoints, MoveEffect) VALUES (102, 'Hydro Pump', 110, 80, 5, 'High power water attack')
3	INSERT INTO Move_Associates1(MoveID, MoveName, Power, Accuracy, PowerPoints, MoveEffect) VALUES (103, 'Solar Beam', 120, 100, 10, 'Charges and fires on second turn')

4	INSERT
	INTO Move_Associates1(MoveID, MoveName, Power, Accuracy, PowerPoints, MoveEffect)
	VALUES (104, 'Thunderbolt', 90, 100, 15, 'May paralyze opponent')

5	INSERT
	INTO Move_Associates1(MoveID, MoveName, Power, Accuracy, PowerPoints, MoveEffect)
	VALUES (105, 'Psychic', 90, 100, 10, 'May lower opponent\'s special defense')

Move_Associates2	

1	INSERT
	INTO Move_Associates2(MoveEffect, TypeName)
	VALUES ('Burns opponent', 'Fire')

2	INSERT INTO Move_Associates2(MoveEffect, TypeName) VALUES ('High power water attack', 'Water')
3	INSERT INTO Move_Associates2(MoveEffect, TypeName) VALUES ('Charges and fires on second turn', 'Grass')

4	INSERT
	INTO Move_Associates2(MoveEffect, TypeName)
	VALUES ('May paralyze opponent', 'Electric')

5	INSERT

	INTO Move_Associates2(MoveEffect, TypeName) VALUES ('May lower opponent special defense', 'Psychic')

Type	
1	INSERT INTO Type(TypeName, TypeDescription) VALUES ('Fire', '*Fire* is one of the three basic elemental types along with Water and Grass')
2	INSERT INTO Type(TypeName, TypeDescription) VALUES ('Water', '*Water* is one of the three basic elemental types along with Fire and Grass')
3	INSERT INTO Type(TypeName, TypeDescription) VALUES ('Grass', '*Grass* is one of the three basic elemental types along with Fire and Water')
4	INSERT INTO Type(TypeName, TypeDescription) VALUES ('Electric', '*Electric* Pokémon are very good defensively, being weak only to Ground moves.')
5	INSERT INTO Type(TypeName, TypeDescription) VALUES ('Psychic', 'The *Psychic* type has few outright strengths, however, it also has few weaknesses.')

Effect	

1	INSERT
	INTO Effect(TypeName1, TypeName2, Percentage)
	VALUES ('Fire', 'Water', 50)

2	INSERT
	INTO Effect(TypeName1, TypeName2, Percentage)
	VALUES ('Fire', 'Electric', 50)

3	INSERT
	INTO Effect(TypeName1, TypeName2, Percentage)
	VALUES ('Fire', 'Fire', 50)

4	INSERT
	INTO Effect(TypeName1, TypeName2, Percentage)
	VALUES ('Grass', 'Water', 200)

5	INSERT
	INTO Effect(TypeName1, TypeName2, Percentage)
	VALUES ('Fire', 'Water', 25)

Belongs	

1	INSERT
	INTO Belongs(PokemonID, TypeName)
	VALUES (1, 'Grass')

2	INSERT
	INTO Belongs(PokemonID, TypeName)
	VALUES (2, 'Fire')

3	INSERT
	INTO Belongs(PokemonID, TypeName)
	VALUES (3, 'Water')

4	INSERT
	INTO Belongs(PokemonID, TypeName)
	VALUES (4, 'Grass')

5	INSERT
	INTO Belongs(PokemonID, TypeName)
	VALUES (5, 'Fire')

Ability	
1	INSERT INTO Ability(AbilityID, AbilityEffect) VALUES (201, 'Overgrow - Boosts Grass moves in a pinch')
2	INSERT INTO Ability(AbilityID, AbilityEffect) VALUES (202, 'Blaze - Boosts Fire moves in a pinch')
3	INSERT INTO Ability(AbilityID, AbilityEffect) VALUES (203, 'Torrent - Boosts Water moves in a pinch')
4	INSERT INTO Ability(AbilityID, AbilityEffect) VALUES (204, 'Static - May cause paralysis upon contact')

5	INSERT
	INTO Ability(AbilityID, AbilityEffect)
	VALUES (205, 'Levitate - Immune to Ground-type moves')

Item_owns	

1	INSERT
	INTO Item_Owns(ItemName, ItemEffect, ItemType, PokemonID)
	VALUES ('Potion', 'Restores 20 HP', 'Healing', NULL)

2	INSERT
	INTO Item_Owns(ItemName, ItemEffect, ItemType, PokemonID)
	VALUES('Fire Stone', 'Evolves Fire-type Pokemon', 'Evolution', 2)

3	INSERT
	INTO Item_Owns(ItemName, ItemEffect, ItemType, PokemonID)
	VALUES ('Water Stone', 'Evolves Water-type Pokemon', 'Evolution', 3)

4	INSERT
	INTO Item_Owns(ItemName, ItemEffect, ItemType, PokemonID)
	VALUES ('Thunder Stone', 'Evolves Electric-type Pokemon', 'Evolution', NULL)

5	INSERT INTO Item_Owns(ItemName, ItemEffect, ItemType, PokemonID) VALUES ('Rare Candy', 'Increases level by one', 'Level Up', NULL)

Sells	

1	INSERT
	INTO Sells(ItemName, LocationName, RegionName)
	VALUES ('Potion', 'Pewter City', 'Kanto')

2	INSERT
	INTO Sells(ItemName, LocationName, RegionName)
	VALUES ('Fire Stone', 'Cerulean City', 'Kanto')

3	INSERT
	INTO Sells(ItemName, LocationName, RegionName)
	VALUES ('Water Stone', 'Lavender Town', 'Kanto')

4	INSERT
	INTO Sells(ItemName, LocationName, RegionName)
	VALUES ('Thunder Stone', 'Vermilion City', 'Kanto')

5	INSERT
	INTO Sells(ItemName, LocationName, RegionName)
	VALUES ('Rare Candy', 'Celadon City', 'Kanto')

Pokemart	
1	INSERT INTO Pokemart(LocationName, RegionName)

	VALUES ('Pewter City', 'Kanto')

2	INSERT
	INTO Pokemart(LocationName, RegionName)
	VALUES ('Cerulean City', 'Kanto')

3	INSERT
	INTO Pokemart(LocationName, RegionName)
	VALUES ('Lavender Town', 'Kanto')

4	INSERT
	INTO Pokemart(LocationName, RegionName)
	VALUES ('Vermilion City', 'Kanto')

5	INSERT
	INTO Pokemart(LocationName, RegionName)
	VALUES ('Celadon City', 'Kanto')

Gym	

1	INSERT
	INTO Gym(LocationName, RegionName, Badge)
	VALUES ('Pewter City', 'Kanto', 'Boulder Badge')

2	INSERT
	INTO Gym(LocationName, RegionName, Badge)
	VALUES ('Cerulean City', 'Kanto', 'Cascade Badge')

3	INSERT
	INTO Gym(LocationName, RegionName, Badge)
	VALUES ('Vermilion City', 'Kanto', 'Thunder Badge')

4	INSERT
	INTO Gym(LocationName, RegionName, Badge)
	VALUES ('Celadon City', 'Kanto', 'Rainbow Badge')

5	INSERT
	INTO Gym(LocationName, RegionName, Badge)
	VALUES ('Fuchsia City', 'Kanto', 'Soul Badge')

Trainer_Defends	

1	INSERT
	INTO Trainer_Defends(TrainerName, Winnings, LocationName, RegionName)
	VALUES ('Brock', 500, 'Pewter City', 'Kanto')

2	INSERT INTO Trainer_Defends(TrainerName, Winnings, LocationName, RegionName)

	VALUES ('Misty', 600, 'Cerulean City', 'Kanto')

3	INSERT
	INTO Trainer_Defends(TrainerName, Winnings, LocationName, RegionName)
	VALUES ('Lt. Surge', 700, 'Vermilion City', 'Kanto')

4	INSERT
	INTO Trainer_Defends(TrainerName, Winnings, LocationName, RegionName)
	VALUES ('Erika', 800, 'Celadon City', 'Kanto')

5	INSERT
	INTO Trainer_Defends(TrainerName, Winnings, LocationName, RegionName)
	VALUES ('Koga', 900, 'Fuchsia City', 'Kanto')

Locations	

1	INSERT
	INTO Location(LocationName, RegionName, Function)
	VALUES ('Pewter City', 'Kanto', 'Gym, Pokemart, Museum')

2	INSERT
	INTO Location(LocationName, RegionName, Function)
	VALUES ('Cerulean City', 'Kanto', 'Gym, Pokemart, Bike Shop')

3	INSERT
	INTO Location(LocationName, RegionName, Function)
	VALUES ('Lavender Town', 'Kanto', 'Pokemart, Haunted Tower')

4	INSERT
	INTO Location(LocationName, RegionName, Function)
	VALUES ('Vermilion City', 'Kanto', 'Gym, Pokemart, Port')

5	INSERT
	INTO Location(LocationName, RegionName, Function)
	VALUES ('Celadon City', 'Kanto', 'Gym, Department Store, Casino')

Region	
1	INSERT INTO Region(RegionName, RegionDescription) VALUES ('Kanto', 'The first region in the Pokémon world, home to 151 species.')
2	INSERT INTO Region(RegionName, RegionDescription) VALUES ('Johto', 'A neighboring region with legendary Pokémon.')
3	INSERT INTO Region(RegionName, RegionDescription)

	VALUES ('Hoenn', 'A tropical region with diverse Pokémon species.')
4	INSERT INTO Region(RegionName, RegionDescription) VALUES ('Sinnoh', 'A cold northern region with ancient legends.')
5	INSERT INTO Region(RegionName, RegionDescription) VALUES ('Unova', 'A modernized region with industrial cities.')

AppearsIn	

1	INSERT
	INTO AppearsIn(RegionName, PokemonID)
	VALUES ('Kanto', 1)

2	INSERT
	INTO AppearsIn(RegionName, PokemonID)
	VALUES ('Kanto', 2)

3	INSERT
	INTO AppearsIn(RegionName, PokemonID)
	VALUES ('Kanto', 3)

4	INSERT
	INTO AppearsIn(RegionName, PokemonID)
	VALUES ('Kanto', 4)

5	INSERT
	INTO AppearsIn(RegionName, PokemonID)
	VALUES ('Kanto', 5)

Owns (for Trainer-Pokemon Relation)	

1	INSERT
	INTO Owns(TrainerName, PokemonID)
	VALUES ('Ash Ketchum', 1)

2	INSERT
	INTO Owns(TrainerName, PokemonID)
	VALUES ('Ash Ketchum', 2)

3	INSERT
	INTO Owns(TrainerName, PokemonID)
	VALUES ('Misty', 3)

4	INSERT INTO Owns(TrainerName, PokemonID)

	VALUES ('Brock', 4)

5	INSERT
	INTO Owns(TrainerName, PokemonID)
	VALUES ('Lt. Surge', 5)

EvolvesInto	

1	INSERT
	INTO EvolvesInto(PreEvolutionID, PostEvolutionID, Condition)
	VALUES (1, 4, 'Level 16')

2	INSERT
	INTO EvolvesInto(PreEvolutionID, PostEvolutionID, Condition)
	VALUES (2, 5, 'Level 16')

3	INSERT
	INTO EvolvesInto(PreEvolutionID, PostEvolutionID, Condition)
	VALUES (3, 6, 'Level 16')

4	INSERT
	INTO EvolvesInto(PreEvolutionID, PostEvolutionID, Condition)
	VALUES (4, 7, 'Level 32')

5	INSERT
	INTO EvolvesInto(PreEvolutionID, PostEvolutionID, Condition)
	VALUES (5, 8, 'Level 36')