University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2
Date: March 1, 2024
Group Number: 20

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Matthew Smith	51209682	v1d5r	magsubc@student.ubc.ca
Aman Prakash	88693502	l1u5t	amanprakashburnett@gmail.com
Danial Jaber	15766819	c1t4e	valentino.jaber@live.com

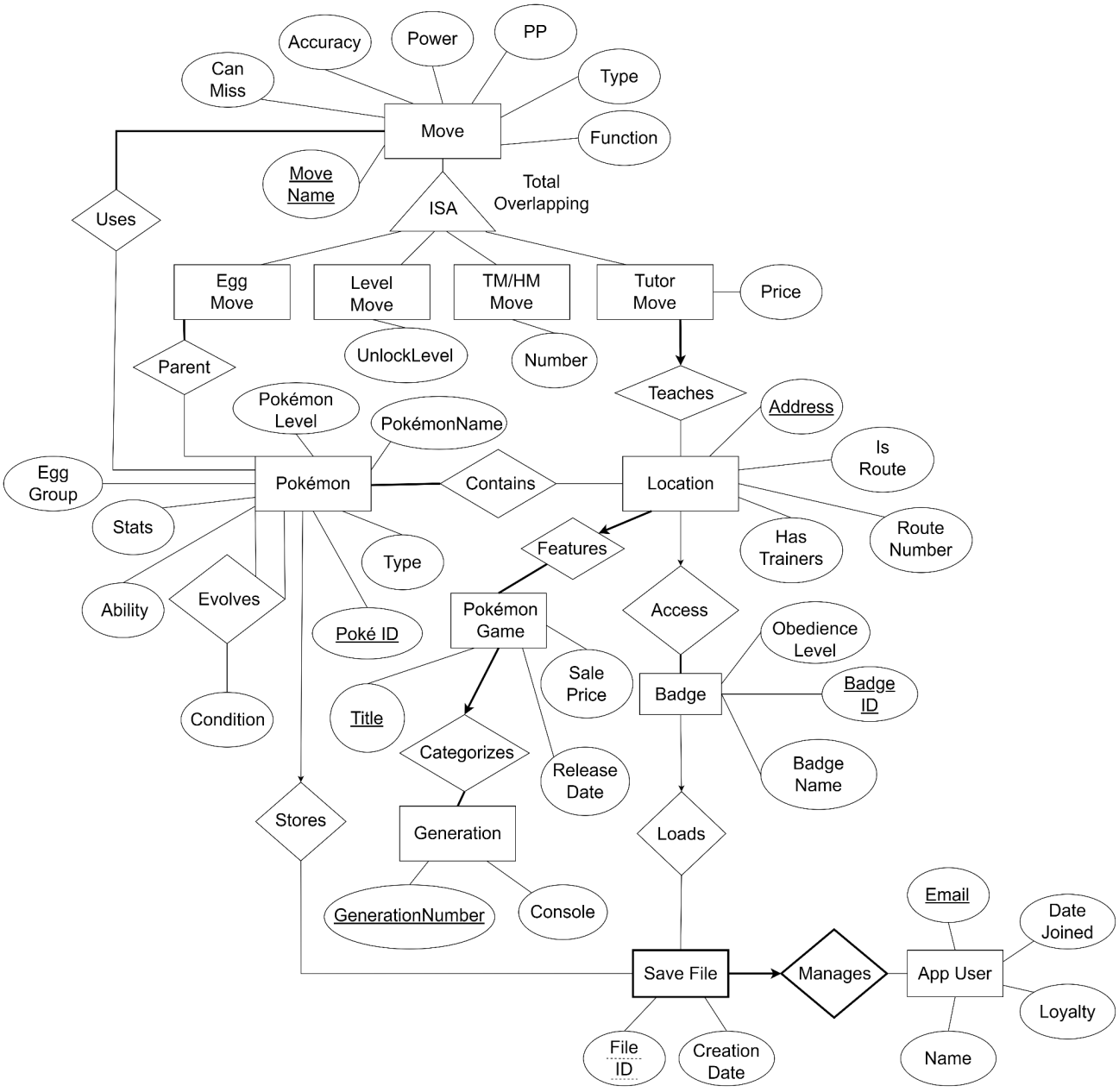
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, 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. A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.

The domain of our proposed application is Pokémon management. This domain mainly pertains to main-series Pokémon game playthroughs. Specifically, when a user is playing a Pokémon game, they might desire to track the traits and details of their currently caught Pokémon, as well as search what other Pokémon are available to them based on their current in-game progress. In essence, our application aims to serve as a thorough Pokédex and encyclopedia for a user's playthrough.

3. The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why



The following changes were made to our ER diagram,

1. Removed ISA relationships for Pokemon and Badge
    a. This was done in an approach to refactor the design for this iteration of the project. These ISA relationships were particularly troublesome due to the fact that they violated primary key constraints of ISA relationships. Removing them altogether and refactoring their functionality into Pokemon and Badge respectively provides a better solution.
2. Removed some unnecessary entities like Trainer
    a. As an attempt at refactoring again, Trainer was removed. This was mostly due to the fact that it is unnecessary for our project, and it provides no real use for the business logic and scope of our design.
3. Made SaveFile a weak entity of User
    a. In removing the previously mentioned ISA relationships, we are required to add either a weak entity or additional ISA to satisfy the milestone 1 requirements. Moreover, a SaveFile does not exist in the absence of an AppUser, so this weak entity relationship was a natural addition.
4. Added constraints on ISA relationship
    a. This is a small fix that addresses an issue our previous ER diagram had from milestone 1.
5. Removed ternary relationship
    a. Due to the complexity of the key and participation constraints needed between Pokemon, Location, and PokemonGame, the ternary relationship was insufficient for our modeling needs. We have instead opted to use two binary relationships with the desired constraints.
6. Added various attributes
    a. Upon completing milestone 2, we noticed that most of our tables were already in BCNF. As such, we added a few new non-trivial attributes to some entities, and consequently introduced some new FDs to go along with them.

4. The schema derived from your ER diagram (above). For translating the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:

a. List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute.

- AppUser (*Email*: string, *Name*: string, *Loyalty*: integer, *DateJoined*: date)
- SaveFileManages(*FileID*: integer, *CreationDate*: date, ***Email***: string)
- BadgeLoads(*BadgeID*: integer, *ObedienceLevel*: integer, *BadgeName* string, ***FileID***: integer, ***Email***: string)
- LocationFeaturesAccess(*Address*: string, ***BadgeID***: integer, ***Title***: string, *IsRoute*: boolean, *HasTrainers*: boolean, *RouteNumber*: integer)
- PokemonGameCategorizes (*Title*: string, *ReleaseDate*: string, ***GenerationNumber***: integer, *SalePrice*: real)

- Contains(***Address***: string, ***PokeID***: integer)
- PokémonStores (*PokémonLevel*: integer, *PokémonName*: string, *Ability*: string, *Type*: string, *PokéID*: integer, *Stats*: integer, *EggGroup*: string, ***FileID***: integer, ***Email***: string)
- TutorMoveTeaches (***MoveName***: string, *Price*: string, ***Address***: string)
- Evolves(***PokeIDfrom***: integer, ***PokeIDto***: integer, *Condition*: string)
- Parent(**MoveName**: string, **PokeID**: integer)
- Uses(**MoveName**: string, **PokeID**: integer)
- Move (*Accuracy*: integer, *Power*: integer, *PP*: integer, *Type*: string, *Function*: string, *MoveName*: string, *CanMiss*: boolean)
- EggMove (***MoveName***: string)
- LevelMove (***MoveName***: string, *UnlockLevel*: integer)
- TMHMMove (***MoveName***: string, *Number*: integer)
- Generation (***GenerationNumber***: integer, *Console*: string)

b. Specify the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain.

The following additional table constraints are tabulated below,

	PK	CKs	FKs	NOT NULL	UNIQUE
AppUser	Email	Email			

SaveFileManages	{
	FileID, Email
	}
		{
		FileID,
		Email
		}
			Email		

BadgeLoads	BadgeID	BadgeID	{
			FileID,
			Email
			}		

LocationFeaturesAccess	Address	Address	BadgeID, Title	Title, BadgeID	

PokemonGameCategorizes	Title	Title	GenerationNumber	GenerationNumber	
Generation	GenerationNumber	Generation Number			

PokemonStores	PokeID	PokeID	{
			FileID,
			Email
			}		

Contains	{
	PokeID,
	Address
	}
		{
		PokeID,
		Address
		}
			PokeID, Address	PokeID	

Parent	{
	MoveName,
	PokeID
	}
		{
		MoveName,
		PokeID
		}
			MoveName, PokeID	PokeID	

Uses	{
	PokeID,
	MoveName
	}
		{
		PokeID,
		MoveName
		}
			PokeID, MoveName	MoveName	

Move	MoveName	MoveName	MoveName		
EggMove	MoveName	MoveName	MoveName		
LevelMove	MoveName	MoveName	MoveName		
TMHMMove	MoveName	MoveName	MoveName		
TutorMoveTeaches	MoveName	MoveName	Address		

Evolves	{
	PokeIDfrom,
	PokeIDto
	}
		{
		PokeIDfrom,
		PokeIDto
		}
			PokeIDfrom, PokeIDto		

5. Functional Dependencies (FDs)

a. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key).

PKs and CKs are considered functional dependencies and should be included in the

list of FDs. You do not need to include trivial FDs such as A → A.

AppUser:
- Email → Name, DateJoined, Loyalty
- DateJoined → Loyalty

SaveFileManages:
- {Email, FileID} → CreationDate

BadgeLoads:
- BadgeID → ObedienceLevel, BadgeName, FileID, Email
- BadgeName → ObedienceLevel

LocationFeaturesAccess:
- Address → Title, BadgeID, IsRoute, HasTrainers, RouteNumber
- IsRoute → HasTrainers
- HasTrainers → IsRoute
- RouteNumber → IsRoute

PokemonGameCategorizes:
- Title → ReleaseDate, GenerationNumber, SalePrice
- ReleaseDate → SalePrice

Generation:
- GenerationNumber → Console

PokemonStores:
- PokeID → PokemonName, PokemonLevel, Ability, Type, Stats, EggGroup, FileID, Email
- PokemonName → Type, EggGroup

Evolves:
- {PokeIDfrom, PokeIDto} → Condition

Move:
- MoveName → Accuracy, Power, PP, Type, Function, CanMiss
- Function → CanMiss
- Accuracy → CanMiss

LevelMove:
- MoveName → UnlockLevel

TMHMMove:
- MoveName → Number

TutorMoveTeaches:

MoveName → Price, Address

Note that relations with only trivial FDs have been omitted from the above answer for sake of brevity.

6. Normalization

a. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization. You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown. The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, ...). ALL Tables must be listed, not only the ones post-normalization.

From above, there are a few tables that we need to renormalize in order to achieve BCNF. All tables above are already in BCNF except AppUser, BadgeLoads, LocationFeaturesAccess, PokemonGameCategorizes, PokemonStores, and Move. This is because of the presence of FDs with a left-hand-side that is not reliant on a candidate key. The following BCNF decompositions are derived.

1. AppUser (*Email*: string, *Name*: string, *Loyalty*: integer, *DateJoined*: date)

(1) Email → Name, DateJoined, Loyalty
(2) DateJoined → Loyalty

We decompose on FD (2) to arrive at two new relations,

AppUser1(*Email*: string, *Name*: string, ***DateJoined***: date)
AppUser2(*DateJoined*: date, *Loyalty*: integer)

Both of the derived tables are in BCNF– we are done.

2. BadgeLoads(*BadgeID*: integer, *ObedienceLevel*: integer, *BadgeName* string, ***FileID***: integer, ***Email***: string)

(1) BadgeID → ObedienceLevel, BadgeName, FileID, Email
(2) BadgeName → ObedienceLevel

We decompose on FD (2) to arrive at two new relations,

BadgeLoads1(*BadgeID*: integer, ***BadgeName***: string, ***FileID***: integer, ***Email***: string)
BadgeLoads2(*BadgeName*: string, *ObedienceLevel*: integer)

Both of the derived tables are in BCNF– we are done.

3. LocationFeaturesAccess(*Address*: string, ***BadgeID***: integer, ***Title***: string, *IsRoute*: boolean, *HasTrainers*: boolean, *RouteNumber*: integer)

(1) Address → Title, BadgeID, IsRoute, HasTrainers, RouteNumber
(2) IsRoute → HasTrainers
(3) HasTrainers → IsRoute
(4) RouteNumber → IsRoute

We decompose on FD (2) to arrive at,

LocationFeaturesAccess1(*Address*: string, ***BadgeID***: integer, ***Title***: string, ***IsRoute***: boolean, *RouteNumber*: integer)
LocationFeaturesAccess2(*IsRoute*: boolean, *HasTrainers*: boolean)

And then again on FD (4) to arrive at,

LocationFeaturesAccess3(*Address*: string, ***BadgeID***: integer, ***Title***: string, ***RouteNumber***: integer)
LocationFeaturesAccess2(*IsRoute*: boolean, *HasTrainers*: boolean)
LocationFeatureAccess4(*RouteNumber*: integer, ***IsRoute***: boolean)

We then rename the relations to arrive finally at,

LocationFeaturesAccess1(*Address*: string, ***BadgeID***: integer, ***Title***: string, ***RouteNumber***: integer)
LocationFeaturesAccess2(*IsRoute*: boolean, *HasTrainers*: boolean)
LocationFeaturesAccess3(*RouteNumber*: integer, ***IsRoute***: boolean)

4. PokemonGameCategorizes (*Title*: string, *ReleaseDate*: string, ***GenerationNumber***: integer, *SalePrice*: real)

(1) Title → ReleaseDate, GenerationNumber, SalePrice
(2) ReleaseDate → SalePrice

We decompose on FD (2) to arrive at,

PokemonGameCategorizes1 (*Title*: string, ***ReleaseDate***: date, ***GenerationNumber***: integer)
PokemonGameCategorizes2 (*ReleaseDate*: date, *SalePrice*: real)

5. PokémonStores (*PokémonLevel*: integer, *PokémonName*: string, *Ability*: string, *Type*: string, *PokéID*: integer, *Stats*: integer, *EggGroup*: string, ***FileID***: integer, ***Email***: string)

(1) PokeID → PokemonName, PokemonLevel, Ability, Type, Stats, EggGroup, FileID, Email
(2) PokemonName → Type, EggGroup

We decompose on FD (2) to arrive at,

PokemonStores1(*PokeID*: integer, ***PokemonName***: string, *PokemonLevel*: integer, Ability: string, *Stats*: integer, ***FileID***: integer, ***Email***: string)
PokemonStores2(*PokemonName*: string, *Type*: string, *EggGroup*: string)

6. Move(*Accuracy*: integer, *Power*: integer, *PP*: integer, *Type*: string, *Function*: string, *MoveName*: string, *CanMiss*: boolean)

(1) MoveName → Accuracy, Power, PP, Type, Function, CanMiss
(2) Function → CanMiss
(3) Accuracy → CanMiss

We first decompose on FD (2) to arrive at,

Move1(*Accuracy*: integer, *Power*: integer, *PP*: integer, *Type*: string, ***Function***: string, *MoveName*: string)
Move2(*Function*: string, *CanMiss*: boolean)

Both of the relations are in BCNF– we are done.

Altogether we then have the following new decomposed relations,

- AppUser1(*Email*: string, *Name*: string, ***DateJoined***: date)
- AppUser2(*DateJoined*: date, *Loyalty*: integer)
- SaveFileManages(*FileID*: integer, *CreationDate*: date, ***Email***: string)
- BadgeLoads1(*BadgeID*: integer, ***BadgeName***: string, ***FileID***: integer, ***Email***: string)
- BadgeLoads2(*BadgeName*: string, *ObedienceLevel*: integer)
- LocationFeaturesAccess1(*Address*: string, ***BadgeID***: integer, ***Title***: string, ***RouteNumber***: integer)
- LocationFeaturesAccess2(*IsRoute*: boolean, *HasTrainers*: boolean)
- LocationFeaturesAccess3(*RouteNumber*: integer, ***IsRoute***: boolean)
- PokemonGameCategorizes1 (*Title*: string, ***ReleaseDate***: date, ***GenerationNumber***: integer)

- PokemonGameCategorizes2 (*ReleaseDate*: date, *SalePrice*: real)
- Contains(***Address***: string, ***PokeID***: integer)
- PokemonStores1(*PokeID*: integer, ***PokemonName***: string, *PokemonLevel*: integer, Ability: string, *Stats*: integer, ***FileID***: integer, ***Email***: string)
- PokemonStores2(*PokemonName*: string, *Type*: string, *EggGroup*: string)
- TutorMoveTeaches (***MoveName***: string, *Price*: string, ***Address***: string)
- Evolves(***PokeIDfrom***: integer, ***PokeIDto***: integer, *Condition*: string)
- Parent(***MoveName***: string, ***PokeID***: integer)
- Uses(***MoveName***: string, ***PokeID***: integer)
- Move1 (*Accuracy*: integer, *Power*: integer, *PP*: integer, *Type*: string, ***Function***: string, *MoveName*: string)
- Move2 (*Function*: string, *CanMiss*: boolean)
- EggMove (***MoveName***: string)
- LevelMove (***MoveName***: string, *UnlockLevel*: integer)
- TMHMMove (***MoveName***: string, *Number*: integer)
- Generation (*GenerationNumber*: integer, *Console*: string)

The table constraints are summarized in the table below,

	PK	CKs	FKs	NOT NULL	UNIQUE
AppUser1	Email	Email	DateJoined		
AppUser2	DateJoined	DateJoined			

SaveFileManages	{
	FileID, Email
	}
		{
		FileID,
		Email
		}
			Email		

BadgeLoads1	BadgeID	BadgeID	{ FileID, Email }, BadgeName		
BadgeLoads2	BadgeName	BadgeName			
LocationFeaturesAccess 1	Address	Address	BadgeID, Title, RouteNumber	Title, BadgeID	
LocationFeaturesAccess 2	IsRoute	IsRoute, HasTrainers			HasTrainers
LocationFeaturesAccess	RouteNumb	RouteNumb	IsRoute		

3	er	er			
PokemonGameCategorizes1	Title	Title	GenerationNumber, ReleaseDate	GenerationNumber	
PokemonGameCategorizes2	ReleaseDate	ReleaseDate			
Generation	GenerationNumber	Generation Number			
PokemonStores1	PokeID	PokeID	{ FileID, Email }, PokemonName		
PokemonStores2	PokemonName	PokemonName			

Contains	{
	PokeID,
	Address
	}
		{
		PokeID,
		Address
		}
			PokeID, Address	PokeID	

Parent	{
	MoveName,
	PokeID
	}
		{
		MoveName,
		PokeID
		}
			MoveName, PokeID	PokeID	

Uses	{
	PokeID,
	MoveName
	}
		{
		PokeID,
		MoveName
		}
			PokeID, MoveName	MoveName	

Move1	MoveName	MoveName	Function		
Move2	Function	Function			
EggMove	MoveName	MoveName	MoveName		
LevelMove	MoveName	MoveName	MoveName		
TMHMMove	MoveName	MoveName	MoveName		
TutorMoveTeaches	MoveName	MoveName	Address, MoveName		

Evolves	{
	PokeIDfrom,
	PokeIDto
	}
		{
		PokeIDfrom,
		PokeIDto
		}
			PokeIDfrom, PokeIDto		

7. The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc.
Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use VARCHAR as the number of characters in a course name can vary greatly.

```
CREATE TABLE AppUser1(
  Email VARCHAR(254),
  Name VARCHAR(50),
  DateJoined DATE,
  PRIMARY KEY (Email),
  FOREIGN KEY (DateJoined) REFERENCES AppUser2 ON DELETE CASCADE
)

CREATE TABLE AppUser2(
  DateJoined DATE,
  Loyalty INTEGER,
  PRIMARY KEY (DateJoined)
)

CREATE TABLE SaveFileManages(
  FileID INTEGER,
  CreationDate DATE,
  Email VARCHAR(254),
  PRIMARY KEY (FileID, Email),
```

```
  FOREIGN KEY (Email) REFERENCES AppUser1 ON DELETE CASCADE
)

CREATE TABLE BadgeLoads1(
  BadgeID INTEGER,
  BadgeName VARCHAR(25),
  FileID INTEGER,
  Email VARCHAR(254),
  PRIMARY KEY (BadgeID),
  FOREIGN KEY (FileID, Email) REFERENCES SaveFile
    ON DELETE SET NULL
    ON UPDATE CASCADE,
  FOREIGN KEY (BadgeName) REFERENCES BadgeLoads2
    ON DELETE CASCADE
    ON UPDATE CASCADE
)

CREATE TABLE BadgeLoads2(
  ObedienceLevel INTEGER,
  BadgeName VARCHAR(25),
  PRIMARY KEY (BadgeName)
)


CREATE TABLE LocationFeaturesAccess1(
  Address VARCHAR(25),
  BadgeID INTEGER NOT NULL,
  Title VARCHAR(50) NOT NULL,
  RouteNumber INTEGER,
  PRIMARY KEY (Address),
  FOREIGN KEY (BadgeID) REFERENCES BadgeLoads1
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (RouteNumber) REFERENCES LocationFeaturesAccess3
    ON DELETE SET NULL
    ON UPDATE CASCADE,
  FOREIGN KEY (Title) REFERENCES PokemonGameCategorizes1
    ON DELETE CASCADE
    ON UPDATE CASCADE
)

CREATE TABLE LocationFeaturesAccess2(
  IsRoute BOOLEAN,
  HasTrainers BOOLEAN,
```

```
  UNIQUE (HasTrainers),
  PRIMARY KEY (IsRoute)
)

CREATE TABLE LocationFeaturesAccess3(
  RouteNumber INTEGER,
  IsRoute BOOLEAN,
  PRIMARY KEY (RouteNumber),
  FOREIGN KEY (IsRoute) REFERENCES LocationFeaturesAccess2
    ON DELETE CASCADE
    ON UPDATE CASCADE
)

CREATE TABLE PokemonGameCategorizes1(
  Title VARCHAR(50),
  ReleaseDate DATE,
  GenerationNumber INTEGER NOT NULL,
  PRIMARY KEY (Title),
  FOREIGN KEY (ReleaseDate) REFERENCES PokemonGameCategorizes2
    ON DELETE SET NULL
    ON UPDATE CASCADE,
  FOREIGN KEY (GenerationNumber) REFERENCES Generation
    ON DELETE NO ACTION
    ON UPDATE CASCADE
)

CREATE TABLE PokemonGameCategorizes2(
  ReleaseDate DATE,
  SalePrice REAL,
  PRIMARY KEY (ReleaseDate)
)

CREATE TABLE Contains(
  PokeID INTEGER NOT NULL,
  Address VARCHAR(25),
  PRIMARY KEY (PokeID, Address)
  FOREIGN KEY (PokeID) REFERENCES PokemonStores1
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (Address) REFERENCES LocationFeaturesAccess1
    ON DELETE CASCADE
    ON UPDATE CASCADE
)
```

```
CREATE TABLE PokemonStores1(
  PokeID INTEGER,
  PokemonName VARCHAR(25),
  PokemonLevel VARCHAR(25),
  Ability VARCHAR(15),
  Stats INTEGER,
  Type VARCHAR(8),
  FileID INTEGER,
  Email VARCHAR(254),
  PRIMARY KEY (PokeID),
  FOREIGN KEY (PokemonName) REFERENCES PokemonStores2
    ON DELETE SET NULL
    ON UPDATE CASCADE,
  FOREIGN KEY (FileID, Email) REFERENCES SaveFile
    ON DELETE CASCADE
    ON UPDATE CASCADE
)

CREATE TABLE PokemonStores2(
  PokemonName VARCHAR(25),
  Type VARCHAR(8),
  EggGroup VARCHAR(15),
  PRIMARY KEY (PokemonName)
)

CREATE TABLE Evolves(
  PokeIDFrom INTEGER,
  PokeIDTo INTEGER,
  Condition VARCHAR(100),
  PRIMARY KEY (PokeIDFrom, PokeIDTo),
  FOREIGN KEY (PokeIDFrom) REFERENCES PokemonStores1 (PokeID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (PokeIDTo) REFERENCES PokemonStores1 (PokeID)
    ON DELETE CASCADE
    ON UPDATE CASCADE
)

CREATE TABLE Parent(
  MoveName VARCHAR(25) NOT NULL,
  PokeID INTEGER,
  PRIMARY KEY (MoveName, PokeID),
  FOREIGN KEY (MoveName) REFERENCES EggMove
```

```
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (PokeID) REFERENCES PokemonStores1
    ON DELETE CASCADE
    ON UPDATE CASCADE
)

CREATE TABLE Uses(
  MoveName VARCHAR(25) NOT NULL,
  PokeID INTEGER,
  PRIMARY KEY (MoveName, PokeID),
  FOREIGN KEY (MoveName) REFERENCES Move
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (PokeID) REFERENCES PokemonStores1
    ON DELETE CASCADE
    ON UPDATE CASCADE
)

CREATE TABLE Move1(
  Accuracy INTEGER,
  Power INTEGER,
  PP INTEGER,
  Type VARCHAR(8),
  Function VARCHAR(15),
  MoveName VARCHAR(25),
  PRIMARY KEY (MoveName)
  FOREIGN KEY (Function) REFERENCES Move2
    ON DELETE SET NULL
    ON UPDATE CASCADE,
)

CREATE TABLE Move2(
  Function VARCHAR(15),
  CanMiss BOOLEAN,
  PRIMARY KEY (Function)
)

CREATE TABLE EggMove(
  MoveName VARCHAR(25),
  PRIMARY KEY (MoveName),
  FOREIGN KEY (MoveName) REFERENCES Move
    ON DELETE CASCADE
    ON UPDATE CASCADE
```

```
)

CREATE TABLE LevelMove(
  MoveName VARCHAR(25),
  UnlockLevel INTEGER,
  PRIMARY KEY (MoveName),
  FOREIGN KEY (MoveName) REFERENCES Move
    ON DELETE CASCADE
    ON UPDATE CASCADE
)

CREATE TABLE TMHMMove(
  MoveName VARCHAR(25),
  Number INTEGER,
  PRIMARY KEY (MoveName),
  FOREIGN KEY (MoveName) REFERENCES Move
    ON DELETE CASCADE
    ON UPDATE CASCADE
)

CREATE TABLE TutorMoveTeaches(
  MoveName VARCHAR(25),
  Price VARCHAR(25),
  Address VARCHAR(25),
  PRIMARY KEY (MoveName),
  FOREIGN KEY (MoveName) REFERENCES Move,
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (Address) REFERENCES LocationFeaturesAccess1
    ON DELETE NO ACTION
    ON UPDATE CASCADE
)

CREATE TABLE Generation(
  GenerationNumber INTEGER,
  Console VARCHAR(25),
  PRIMARY KEY (GenerationNumber)
)
```

8. INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later.

```
SET DATEFORMAT dmy;

INSERT INTO AppUser1(Email, Name, DateJoined)
VALUES      ('111@gmail.com', 'Ash', '01.05.2022'),
            ('222@gmail.com', 'Ash', '01.05.2022'),
            ('333@gmail.com', 'Wash', '12.05.1999'),
            ('444@gmail.com', 'epicman', '06.20.2000'),
            ('555@gmail.com', '555', '10.01.2023');

INSERT INTO AppUser2(DateJoined, Loyalty)
VALUES      ('01.05.2022', 2),
            ('01.06.2022', 2),
            ('12.05.1999', 25),
            ('06.20.2000', 24),
            ('10.01.2023', 1);

INSERT INTO SaveFileManages(FileID, CreationDate, Email)
VALUES      (1, '01.05.2022', '111@gmail.com'),
            (2, '01.05.2022', '111@gmail.com'),
            (1, '12.05.1999', '222@gmail.com'),
            (1, '06.20.2000', '444@gmail.com'),
            (1, '10.01.2023', '555@gmail.com');

INSERT INTO BadgeLoads1(BadgeID, BadgeName, FileID, Email)
VALUES      (1, 'Cascade', 1, '111@gmail.com'),
            (2, 'Rock', 3, 'abc@gmail.com'),
            (3, 'Rainbow', 2, '123@gmail.com'),
            (4, 'Cascade', 1, '111@gmail.com'),
            (5, 'Cascade', 6, '555@gmail.com');

INSERT INTO BadgeLoads2(BadgeName, ObedienceLevel)
VALUES      ('Cascade', 15),
            ( 'Rock', 25),
            ('Rainbow', 100),
            ( 'Spectre', 60),
            ( 'Blaze', 25);
```

```
INSERT INTO LocationFeaturesAccess1(Address, BadgeID, Title, RouteNumber)
VALUES      ('6200 University Blvd', 1, 'Scarlet and Violet', 1),
            ('1860 E Mall #301', 2, 'Brilliant Diamond', 2),
            ('6133 University Blvd', 3, 'Legends Arceus', 3),
            ('6138 Student Union Blvd', 4, 'Unite', 4),
            ('2053 Main Mall', 5, 'Shining Pearl', -1);

INSERT INTO LocationFeaturesAccess2(IsRoute, HasTrainer)
VALUES      (TRUE, TRUE),
            (FALSE, FALSE);

INSERT INTO LocationFeaturesAccess3(RouteNumber, IsRoute)
VALUES      (1, TRUE),
            (2, TRUE),
            (3, TRUE),
            (4, TRUE),
            (-1, FALSE);

INSERT INTO PokemonGameCategorizes1(Title, ReleaseDate, GenerationNumber)
VALUES      ('Scarlet and Violet', '01.05.2022', 1),
            ('Brilliant Diamond', '01.06.2022', 2),
            ('Legends Arceus', '12.05.1999', 3),
            ('Unite', '06.20.2000', 4),
            ('Shining Pearl', '10.01.2023', 5);

INSERT INTO PokemonGameCategorizes2(ReleaseDate, SalePrice)
VALUES      ('01.05.2022', 50.50),
            ('01.06.2022', 50.55),
            ('12.05.1999', 31.31),
            ('06.20.2000', 24.00),
            ('10.01.2023', 57.99);

INSERT INTO Contains(Address, PokeID)
VALUES      ('6200 University Blvd', 1),
            ('1860 E Mall #301', 2),
            ('6200 University Blvd', 3),
            ('6138 Student Union Blvd', 4),
            ('2053 Main Mall', 5);
```

```
INSERT INTO PokemonStores1(PokeID, PokemonName, PokemonLevel, Ability, Stats, FileID,
Email)
VALUES      (1, Charizard, 7, 'fly', 313131313131, 1, '555@gmail.com'),
            (2, Pikachu, 77, 'rizz', 310031293105, 2, '111@gmail.com'),
            (3, Squirtle, 777, 'dance', 312131093100, 1, '222@gmail.com'),
            (4, Wartortle, 16, 'float', 310031003100, 1, '222@gmail.com'),
            (5, Raichu, 771, 'throw', 310031003111, 2, '111@gmail.com'),
            (6, Charmander, 420, 'punch', 000000000000, 1, '555@gmail.com'),
            (7, Charmeleon, 8, 'think', 001122300011, 1, '555@gmail.com'),
            (8, Blastoise, 36, 'spray', 100000000000, 1, '222@gmail.com');

INSERT INTO PokemonStores2(PokemonName, Type, EggGroup)
VALUES      (Charizard, 'Fire', 'Dragon'),
            (Pikachu, 'Electric', 'Fairy'),
            (Squirtle, 'Water', 'Water1'),
            (Wartortle, 'Water', 'Water1'),
            (Raichu, 'Electric', 'Fairy'),
            (Charmander, 'Fire', 'Dragon'),
            (Charmeleon, 'Fire', 'Dragon'),
            (Blastoise, 'Water', 'Water1');

INSERT INTO TutorMoveTeaches(MoveName, Price, Address)
VALUES      ('Counter', '1 Diamond', '6200 University Blvd'),
            ('Body Slam', '3 Gold', '1860 E Mall #301'),
            ('Explosion', '2 Bronze', '6133 University Blvd'),
            ('Mega Kick', '7 Bronze', '6138 Student Union Blvd'),
            ('Mimic', '4 Dirt', '2053 Main Mall');

INSERT INTO Evolves(PokeIDfrom, PokeIDto, Condition)
VALUES      (1, 6, 'eats some food'),
            (2, 5, 'goes to therapy'),
            (3, 4, 'swims a bit'),
            (7, 6, 'starts a fire'),
            (4, 8, 'learns to swim');

INSERT INTO Parent(MoveName, PokeID)
VALUES      ('Counter', 1),
            ('Body Slam', 2),
            ('Explosion', 3),
            ('Mega Kick', 4),
            ('Mimic', 5),
            ('Pound', 6),
            ('Karate Chop', 7),
            ('Scratch', 8);
```

```
INSERT INTO Uses(MoveName, PokeID)
VALUES      ('Counter', 1),
            ('Counter', 2),
            ('Counter', 3),
            ('Counter', 4),
            ('Counter', 5),
            ('Counter', 6),
            ('Counter', 7),
            ('Counter', 8);

INSERT INTO Move1(Accuracy, Power, PP, Type, Function, MoveName)
VALUES      (100, 100, 15, 'Fire', 'Special', 'Flamethrower'),
            (85, 120, 10, 'Rock', 'Physical', 'Stone Edge'),
            (100, 0, 15, 'Psychic', 'Status', 'Calm Mind'),
            (100, 0, 15, 'Normal', 'Status', 'Swords Dance'),
            (100, 40, 25, 'Dragon', 'Special', 'Dragon Rage');

INSERT INTO Move2(Function, CanMiss)
VALUES      ('Status', FALSE),
            ('Physical', TRUE),
            ('Special', TRUE);

INSERT INTO Generation(GenerationNumber, Console)
VALUES      (1, 'GameBoy'),
            (2, 'GameBoy'),
            (3, 'GameBoy Advance'),
            (4, 'Nintendo DS'),
            (9, 'Nintendo Switch');

INSERT INTO TMHMMove(MoveName, Number)
VALUES      ('Bullet Seed', 9),
            ('Sunny Day', 11),
            ('Endure', 58),
            ('Rock Slide', 80),
            ('Cut', 1);

INSERT INTO LevelMove(MoveName, UnlockLevel)
VALUES      ('Flamethrower', 60),
            ('Water Gun', 6),
            ('Rock Throw', 60),
            ('Grass Knot', 25),
            ('Ancient Power', 34);
```

```
INSERT INTO EggMove(MoveName)
VALUES      ('Counter'),
            ('Body Slam'),
            ('Explosion'),
            ('Mega Kick'),
            ('Mimic'),
            ('Pound'),
            ('Karate Chop'),
            ('Scratch');
```

IMPORTANT NOTES:

We have included the following statement: SET DATEFORMAT dmy; at the top of the INSERT statements. This is to avoid ambiguity on the format we use for any dates in the INSERT statements.

We only insert two tuples into LocationFeaturesAccess2 because those are the only tuples that can exist based off of the domain of IsRoute.

We only insert three tuples into Move2 because those are the only tuples that can exist based off of the domain of Function (only “Status”, “Physical”, and “Special” exist).