**University of British Columbia, Vancouver**
Department of Computer Science

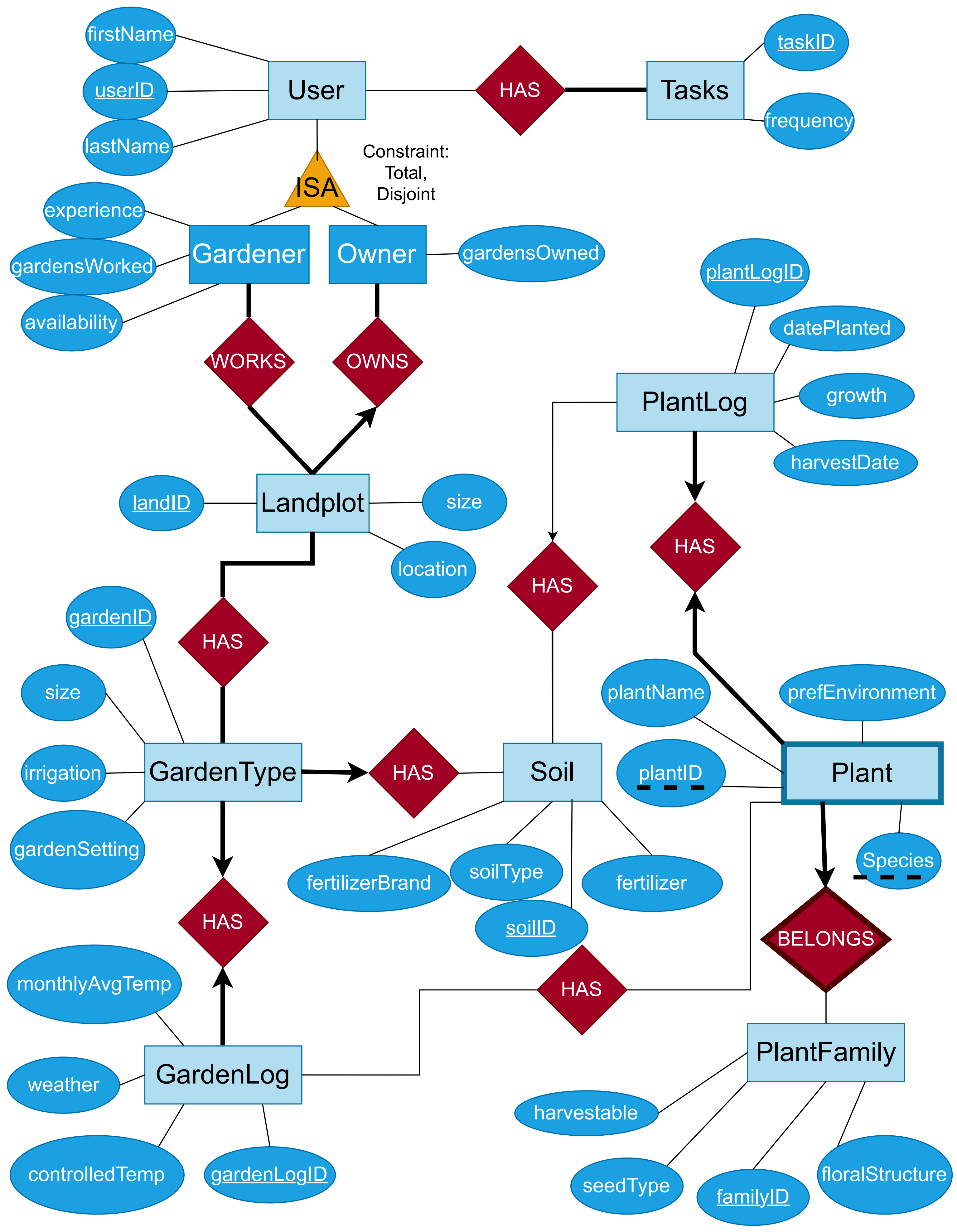
# CPSC 304 Project Cover Page

Milestone #: 2

Date: July 24, 2025

Group Number: 8

| Name | Student Number | CS Alias (Userid) | Preferred E-mail Address |
|---|---|---|---|
| Justin Galimpin | 59053306 | jgalimpi | jgalimpi@student.ubc.ca |
| Jacky Wang | 32227530 | wwang79 | jacky05wang@gmail.com |
| Ericson Ho | 94750692 | cyeho | cyeho@student.ubc.ca |


firstName
userID
lastName
User
HAS
Tasks
taskID
frequency
ISA
Constraint: Total, Disjoint
experience
gardensWorked
availability
Gardener
Owner
gardensOwned
WORKS
OWNS
plantLogID
datePlanted
PlantLog
growth
harvestDate
landID
Landplot
size
location
HAS
HAS
HAS
gardenID
size
irrigation
gardenSetting
GardenType
HAS
Soil
plantName
prefEnvironment
plantID
Plant
Species
fertilizerBrand
soilType
fertilizer
soilID
HAS
BELONGS
monthlyAvgTemp
weather
GardenLog
HAS
controlledTemp
gardenLogID
PlantFamily
harvestable
seedType
familyID
floralStructure

**2. A brief project description (2-3 sentences):**
Our project is a garden and plant management system that helps users, specifically garden owners and gardeners, track tasks, monitor plant growth, and record environmental conditions across land plots. The system models entities such as users, gardens, plants, soil types, and activity logs to support gardening practices, ensuring more organized and sustainable gardening practices.

**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.

We made three general changes to our ER diagram: (1) we added constraints to our Is-A relationship - namely, "Total Disjoint," and (2) we clarified the diagram by rearranging some of our relationships and entities. For instance, we simplified/removed relationships between entities that were redundant and/or unnecessary. One example is that we removed the relationship between "Soil" and "Plant"; this is because "PlantLog" already has a relationship with both of these entities, and as such, we can indirectly gain information about a Plant's Soil through its PlantLog. Lastly, we renamed some attributes (i.e., User(ID) to User(userID)) for additional clarity, while adding some to better describe and/or identify our entities (i.e., "soilID" in Soil and "harvestable" in PlantFamily - the latter of which assumes that a plant's harvestability is contingent on the family it is part of). Furthermore, the addition of *plantID* in *Plant* represents the *species* in number, and their combination is associated with a plant's partial identification, with each plant further defined by the family it belongs to.

**4. The schema derived from your ER diagram (above).**
For the translation of 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.
b. 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.

- User(<u>userID: INTEGER (PK)</u>, firstName: VARCHAR(20), lastName: VARCHAR(20))
  - Constraints: None

- Owner(<u>**userID: INTEGER (PK, FK)**</u>, gardensOwned: INTEGER)
  - Constraints:
    - gardensOwned NOT NULL

- Gardener(<u>**userID: INTEGER (PK, FK)**</u>, experience: VARCHAR(20), gardensWorked: INTEGER, availability: VARCHAR(20))
  - Constraints:
    - availability NOT NULL

- Tasks(taskID: INTEGER (PK), frequency: VARCHAR(20))
    - Constraints: None
- User_HAS_Task(**userID: INTEGER (PK, FK)**, **taskID: INTEGER (PK,FK)**)
    - Constraints: None
- Garden_WORKS_Landplot(**userID: INTEGER (PK, FK)**, **landID: INTEGER (PK, FK)**)
    - Constraints: None
- Owner_OWNS_Landplot(**userID: INTEGER (PK, FK)**, **landID: INTEGER (PK, FK)**)
    - Constraints: None
- Landplot(landID: INTEGER (PK), location: VARCHAR(20), size: FLOAT)
    - Constraints:
        - location NOT NULL
        - size NOT NULL
- GardenType(gardenID: INTEGER (PK), **soilID: INTEGER (FK)**, size: FLOAT, irrigation: VARCHAR(20), gardenSetting: VARCHAR(20))
    - Constraints:
        - soilID NOT NULL
- Landplot_HAS_GardenType(**landID: INTEGER (PK, FK)**, **gardenID: INTEGER (PK, FK)**)
    - Constraints: None
- GardenLog(gardenLogID: INTEGER (PK), **gardenID: INTEGER (FK)**, weather: VARCHAR (20), monthlyAvgTemp: FLOAT, controlledTemp: FLOAT)
    - Constraints: None
- Soil(soilID: INTEGER (PK), soilType: VARCHAR(20), fertilizer: VARCHAR(20), fertilizerBrand: VARCHAR(20))
    - Constraints:
        - soilType NOT NULL
- Plant(**plantID: INTEGER (PK, FK)**, **familyID: INTEGER (PK, FK),** plantName: VARCHAR(20), species: VARCHAR(20), prefEnvironment: VARCHAR(20))
    - Constraints:
        - species NOT NULL
        - prefEnvironment NOT NULL
        - plantName NOT NULL

- PlantLog(plantLogID: INTEGER (PK), **plantID: INTEGER (FK)**, **species: VARCHAR(20) (FK)**, **familyID: INTEGER (FK)**, **soilID: INTEGER (FK)**, datePlanted: DATE, growth: VARCHAR(20), harvestDate: VARCHAR(20))
    - Constraints:
        - datePlanted NOT NULL

- PlantFamily(familyID: INTEGER (PK), seedType: VARCHAR(20), floralStructure: VARCHAR(20), harvestable: BOOLEAN)
    - Constraints: None
        - seedType NOT NULL
        - floralStructure NOT NULL
        - boolean NOT NULL
        - harvestable NOT NULL

- GardenLog_HAS_Plant(**plantID: INTEGER (PK, FK)**, **gardenID: INTEGER (PK, FK)**)
    - Constraints: None

**5. Functional Dependencies (FDs)**
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.

**Note:** In your list of FDs, there must be some kind of valid FD other those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good normalization exercise. Your design must go through a normalization process. You do not need to have a non-PK/CK FD for each relation but be reasonable. If your TA feels that some non-PK/CK FDs have been omitted, your grade will be adjusted accordingly.

- User(userID: INTEGER (PK), firstName: VARCHAR(20), lastName: VARCHAR(20))
    - userID → firstName, lastName

- Owner(**userID: INTEGER (PK, FK)**, gardensOwned: INTEGER)
    - userID → gardensOwned

- Gardener(**userID: INTEGER (PK, FK)**, experience: VARCHAR(20), gardensWorked: INTEGER, availability: VARCHAR(20))
    - userID → experience, gardensWorked, availability

- Tasks(taskID: INTEGER (PK), frequency: VARCHAR(20))
    - taskID → frequency

- Landplot(landID: INTEGER (PK), location: VARCHAR(20), size: FLOAT)
  - landID → location, size

- GardenType(gardenID: INTEGER (PK), **soilID: INTEGER (FK)**, size: FLOAT, irrigation: VARCHAR(20), gardenSetting: VARCHAR(20))
  - gardenID → size, irrigation, gardenSetting, soilID
  - gardenSetting → irrigation

- GardenLog(gardenLogID: INTEGER (PK), **gardenID: INTEGER (FK)**, weather: VARCHAR (20), monthlyAvgTemp: FLOAT, controlledTemp: FLOAT)
  - gardenLogID → gardenID, weather, monthlyAvgTemp, controlledTemp

- Soil(soilID: INTEGER (PK), soilType: VARCHAR(20), fertilizer: VARCHAR(20), fertilizerBrand: VARCHAR(20))
  - soilID → soilType, fertilizer, fertilizerBrand
  - fertilizerBrand → fertilizer

- Plant(plantID: INTEGER (PK), species: VARCHAR(20) (PK), **familyID: INTEGER (PK, FK),** plantName: VARCHAR(20), prefEnvironment: VARCHAR(20))
  - plantID, species → plantName, prefEnvironment
  - species → familyID, prefEnvironment

- PlantLog(plantLogID: INTEGER (PK), **plantID: INTEGER (FK)**, **species: VARCHAR(20) (FK)**, **familyID: INTEGER (FK)**, **soilID: INTEGER (FK)**, datePlanted: DATE, growth: VARCHAR(20), harvestDate: VARCHAR(20))
  - plantLogID → plantID, species, familyID, soilID, datePlanted, growth, harvestDate

- PlantFamily(familyID: INTEGER (PK), seedType: VARCHAR(20), floralStructure: VARCHAR(20), harvestable: BOOLEAN)
  - familyID → seedType, floralStructure
  - seedType → harvestable

## 6. Normalization

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.

a) GardenType(<u>gardenID: INTEGER (PK)</u>, **soilID: INTEGER (FK)**, size: FLOAT, irrigation: VARCHAR(20), gardenSetting: VARCHAR(20))
   - Functional Dependencies:
     - gardenID → soilID, size, irrigation, gardenSetting
     - gardenSetting → irrigation **(violates BCNF)**

   **<u>Decomposed GardenType</u>**

   GardenType(<u>gardenID: INTEGER (PK)</u>, **soilID: INTEGER (FK)**, size: FLOAT, **gardenSetting: VARCHAR(20) (FK)**)
      PK: gardenID
      FK: soilID, gardenSetting

   GardenSetting(<u>gardenSetting: VARCHAR(20) (PK)</u>, irrigation: VARCHAR(20))
      PK: gardenSetting

b) Soil(<u>soilID: INTEGER (PK)</u>, soilType: VARCHAR(20), fertilizer: VARCHAR(20), fertilizerBrand: VARCHAR(20))
   - Functional Dependencies:
     - soilID → soilType, fertilizer, fertilizerBrand
     - fertilizerBrand → fertilizer **(violates BCNF)**

   **<u>Decomposed Soil</u>**

   Soil(<u>soilID: INTEGER (PK)</u>, soilType: VARCHAR(20), **fertilizerBrand: VARCHAR(20) (FK)**)
      PK: soilID
      FK: fertilizerBrand

   Fertilizer(<u>fertilizerBrand: VARCHAR(20) (PK)</u>, fertilizer: VARCHAR(20))
      PK: fertilizerBrand

c) PlantFamily(<u>familyID: INTEGER (PK)</u>, seedType: VARCHAR(20), floralStructure: VARCHAR(20), harvestable: BOOLEAN)
   - Functional Dependencies:
     - familyID → seedType, floralStructure
     - seedType → harvestable **(violates BCNF)**

**Decomposed PlantFamily**

PlantFamily(familyID: INTEGER (PK), **seedType: VARCHAR(20) (FK)**, floralStructure: VARCHAR(20))
PK: familyID
FK: seedType

seedType(seedType: VARCHAR(20) (PK), harvestable: BOOLEAN)
PK: seedType

d) Plant(plantID: INTEGER (PK), species: VARCHAR(20) (PK), **familyID: INTEGER (PK, FK),** plantName: VARCHAR(20), prefEnvironment: VARCHAR(20))
- Functional Dependencies:
  - plantID, species → plantName, prefEnvironment
  - species → familyID, prefEnvironment **(violates BCNF)**

**Decomposed Plant**

PlantInfo(species: VARCHAR(20) (PK), **familyID: INTEGER (FK)**, prefEnvironment: VARCHAR(20))
PK: species
FK: familyID

Plant(plantID: INTEGER (PK)**, species: VARCHAR(20) (PK, FK)**, plantName: VARCHAR(20))
PK: plantID, species
FK: species

**Table List:**

User(userID: INTEGER (PK), firstName: VARCHAR(20), lastName: VARCHAR(20))

Owner(**userID: INTEGER (PK, FK)**, gardensOwned: INTEGER)

Gardener(**userID: INTEGER (PK, FK)**, experience: VARCHAR(20), gardensWorked: INTEGER, availability: VARCHAR(20))

Tasks(taskID: INTEGER (PK), frequency: VARCHAR(20))

User_HAS_Task(**userID: INTEGER (PK, FK)**, **taskID: INTEGER (PK, FK)**)

Garden_WORKS_Landplot(**userID: INTEGER (PK, FK)**, **landID: INTEGER (PK, FK)**)

Owner_OWNS_Landplot(**userID: INTEGER (PK, FK)**, **landID: INTEGER (PK, FK)**)

Landplot(landID: INTEGER (PK), location: VARCHAR(20), size: FLOAT)

GardenType(gardenID: INTEGER (PK), **soilID: INTEGER (FK)**, size: FLOAT, **gardenSetting: VARCHAR(20) (FK)**)

GardenSetting(gardenSetting: VARCHAR(20) (PK), irrigation: VARCHAR(20))

Landplot_HAS_GardenType(**landID: INTEGER (PK, FK)**, **gardenID: INTEGER (PK, FK)**)

GardenLog(gardenLogID: INTEGER (PK), **gardenID: INTEGER (FK)**, weather: VARCHAR(20), monthlyAvgTemp: FLOAT, controlledTemp: FLOAT)

Soil(soilID: INTEGER (PK), soilType: VARCHAR(20), **fertilizerBrand: VARCHAR(20) (FK)**)

Fertilizer(fertilizerBrand: VARCHAR(20) (PK), fertilizer: VARCHAR(20))

Plant(plantID: INTEGER (PK)**, species: VARCHAR(20) (PK, FK),** plantName: VARCHAR(20))

PlantInfo(species: VARCHAR(20) (PK), **familyID: INTEGER (FK)**, prefEnvironment: VARCHAR(20))

PlantLog(plantLogID: INTEGER (PK), **plantID: INTEGER (FK), species: VARCHAR(20) (FK), soilID: INTEGER (FK)**, datePlanted: DATE, growth: VARCHAR(20), harvestDate: DATE)

PlantFamily(familyID: INTEGER (PK), **seedType: VARCHAR(20) (FK)**, floralStructure: VARCHAR(20))

SeedType(seedType: VARCHAR(20) (PK), harvestable: BOOLEAN)

GardenLog_HAS_Plant(**plantID: INTEGER (PK, FK)**, **species: VARCHAR(20) (PK, FK), gardenLogID: INTEGER (PK, FK)**)

**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 User (
        userID INTEGER,
        firstName VARCHAR(20),
        lastName VARCHAR(20),
        PRIMARY KEY (userID)
);

CREATE TABLE Owner (
        userID INTEGER,
        gardensOwned INTEGER NOT NULL,
        PRIMARY KEY (userID),
        FOREIGN KEY (userID) REFERENCES User(userID)
);

CREATE TABLE Gardener (
        userID INTEGER,
        experience VARCHAR(20),
        gardensWorked INTEGER,
        availability VARCHAR(20) NOT NULL,
        PRIMARY KEY (userID),
        FOREIGN KEY (userID) REFERENCES User(userID)
);

CREATE TABLE Tasks (
        taskID INTEGER,
        frequency VARCHAR(20),
        PRIMARY KEY (taskID)
);

CREATE TABLE User_HAS_Task (
        userID INTEGER,
        taskID INTEGER,
        PRIMARY KEY (userID, taskID),
        FOREIGN KEY (userID) REFERENCES User(userID),
        FOREIGN KEY (taskID) REFERENCES Tasks(taskID)
);
```

```
CREATE TABLE Landplot (
        landID INTEGER,
        location VARCHAR(20) NOT NULL,
        size FLOAT NOT NULL,
        PRIMARY KEY (landID)
);

CREATE TABLE Garden_WORKS_Landplot (
        userID INTEGER,
        landID INTEGER,
        PRIMARY KEY (userID, landID),
        FOREIGN KEY (userID) REFERENCES User(userID),
        FOREIGN KEY (landID) REFERENCES Landplot(landID)
);

CREATE TABLE Owner_OWNS_Landplot (
        userID INTEGER,
        landID INTEGER,
        PRIMARY KEY (userID, landID),
        FOREIGN KEY (userID) REFERENCES User(userID),
        FOREIGN KEY (landID) REFERENCES Landplot(landID)
);

CREATE TABLE Fertilizer (
        fertilizerBrand VARCHAR(20),
        fertilizer VARCHAR(20),
        PRIMARY KEY (fertilizerBrand)
);

CREATE TABLE Soil (
        soilID INTEGER,
        soilType VARCHAR(20) NOT NULL,
        fertilizerBrand VARCHAR(20),
        PRIMARY KEY (soilID),
        FOREIGN KEY (fertilizerBrand) REFERENCES Fertilizer(fertilizerBrand)
);

CREATE TABLE GardenSetting (
        gardenSetting VARCHAR(20),
        irrigation VARCHAR(20),
        PRIMARY KEY (gardenSetting)
);

CREATE TABLE GardenType (
        gardenID INTEGER,
        soilID INTEGER NOT NULL,
        size FLOAT,
        gardenSetting VARCHAR(20),
        PRIMARY KEY (gardenID),
        FOREIGN KEY (gardenSetting) REFERENCES GardenSetting(gardenSetting),
        FOREIGN KEY (soilID) REFERENCES Soil(soilID)
);

CREATE TABLE Landplot_HAS_GardenType (
        landID INTEGER,
        gardenID INTEGER,
        PRIMARY KEY (landID, gardenID),
        FOREIGN KEY (landID) REFERENCES Landplot(landID),
        FOREIGN KEY (gardenID) REFERENCES GardenType(gardenID)
);

CREATE TABLE GardenLog (
        gardenLogID INTEGER,
        gardenID INTEGER,
        weather VARCHAR(20),
        monthlyAvgTemp FLOAT,
        controlledTemp FLOAT,
        PRIMARY KEY (gardenLogID),
        FOREIGN KEY (gardenID) REFERENCES GardenType(gardenID)
);

CREATE TABLE SeedType (
        seedType VARCHAR(20),
        harvestable BOOLEAN NOT NULL,
        PRIMARY KEY (seedType)
);

CREATE TABLE PlantFamily (
        familyID INTEGER,
        seedType VARCHAR(20) NOT NULL,
        floralStructure VARCHAR(20) NOT NULL,
        PRIMARY KEY (familyID),
        FOREIGN KEY (seedType) REFERENCES SeedType(seedType)
);

CREATE TABLE PlantInfo (
        species VARCHAR(20),
        familyID INTEGER,
        prefEnvironment VARCHAR(20),
        PRIMARY KEY (species),
        FOREIGN KEY (familyID) REFERENCES PlantFamily(familyID)
);
```

```
CREATE TABLE Plant (
        plantID INTEGER,
        species VARCHAR(20) NOT NULL,
        plantName VARCHAR(20) NOT NULL,
        PRIMARY KEY (plantID, species),
        FOREIGN KEY (species) REFERENCES PlantInfo(species)
);

CREATE TABLE PlantLog (
        plantLogID INTEGER,
        plantID INTEGER,
        species VARCHAR(20) NOT NULL,
        soilID INTEGER,
        datePlanted DATE NOT NULL,
        growth VARCHAR(20),
        harvestDate DATE,
        PRIMARY KEY (plantLogID),
        FOREIGN KEY (plantID, species) REFERENCES Plant(plantID, species),
        FOREIGN KEY (soilID) REFERENCES Soil(soilID)
);

CREATE TABLE GardenLog_HAS_Plant (
        gardenLogID INTEGER,
        plantID INTEGER,
        species VARCHAR(20) NOT NULL,
        PRIMARY KEY (plantID, species, gardenLogID),
        FOREIGN KEY (plantID, species) REFERENCES Plant(plantID, species),
        FOREIGN KEY (gardenLogID) REFERENCES GardenLog(gardenLogID)
);
```

**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.

```
INSERT INTO User (userID, firstName, lastName) VALUES
(1, 'Ericson', 'Ho'),
(2, 'Justin', 'Galimpin'),
(3, 'Jacky', 'Wang'),
(4, 'John', 'Doe'),
(5, 'Michael', 'Jordan'),
(6, 'Bob', 'Smith'),
(7, 'LeBron', 'James'),
(8, 'Kevin', 'Levin'),
(9, 'Tom', 'Cruise'),
(10, 'Jackie', 'Chan');

INSERT INTO Owner (userID, gardensOwned) VALUES
(1, 2),
(2, 1),
(3, 3),
(4, 2),
(5, 0);

INSERT INTO Gardener (userID, experience, gardensWorked, availability) VALUES
(6, 'Expert', 5, 'Weekends'),
(7, 'Intermediate', 2, 'Weekdays'),
(8, 'Beginner', 1, 'Evenings'),
(9, 'Expert', 4, 'Weekdays'),
(10, 'Intermediate', 3, 'Mornings');

INSERT INTO Tasks (taskID, frequency) VALUES
(101, 'Daily'),
(102, 'Weekly'),
(103, 'Biweekly'),
(104, 'Monthly'),
(105, 'Annually');

INSERT INTO User_HAS_Task (userID, taskID) VALUES
(1, 101),
(2, 102),
(5, 103),
(6, 104),
(7, 105);
```

```
INSERT INTO Landplot (landID, location, size) VALUES
(201, 'Burnaby', 300.5),
(202, 'Richmond', 150.0),
(203, 'Vancouver', 250.0),
(204, 'Surrey', 100.0),
(205, 'NorthVancouver', 500.0);

INSERT INTO Owner_OWNS_Landplot (userID, landID) VALUES
(1, 201),
(2, 202),
(3, 203),
(4, 204),
(5, 205);

INSERT INTO Garden_WORKS_Landplot (userID, landID) VALUES
(6, 201),
(7, 202),
(8, 203),
(9, 204),
(10, 205);

INSERT INTO Fertilizer (fertilizerBrand, fertilizer) VALUES
('SyntheticFert', 'Compost'),
('EcoBoost', 'Manure'),
('SoilMix', 'Peat'),
('NatureRich', 'Organic'),
('FastGrow', 'Chemical');

INSERT INTO Soil (soilID, soilType, fertilizerBrand) VALUES
(401, 'Loamy', 'SyntheticFert'),
(402, 'Sandy', 'EcoBoost'),
(403, 'Clay', 'SoilMix'),
(404, 'Silty', 'NatureRich'),
(405, 'Peaty', 'FastGrow');

INSERT INTO GardenSetting (gardenSetting, irrigation) VALUES
('Urban', 'Drip'),
('Suburban', 'Sprinkler'),
('Natural', 'None');

INSERT INTO GardenType (gardenID, soilID, size, gardenSetting) VALUES
(301, 401, 300.5, 'Suburban'),
(302, 402, 150.0, 'Suburban'),
(303, 403, 250.0, 'Urban'),
(304, 404, 100.0, 'Suburban'),
(305, 405, 500.0, 'Natural');
```

```
INSERT INTO Landplot_HAS_GardenType (landID, gardenID) VALUES
(201, 301),
(202, 302),
(203, 303),
(204, 304),
(205, 305);

INSERT INTO GardenLog (gardenLogID, gardenID, weather, monthlyAvgTemp,
controlledTemp) VALUES
(501, 301, 'Sunny', 22.5, 21.0),
(502, 302, 'Rainy', 18.0, 17.5),
(503, 303, 'Cloudy', 20.0, 19.5),
(504, 304, 'Windy', 23.0, 22.0),
(505, 305, 'Foggy', 19.0, 18.0);

INSERT INTO SeedType (seedType, harvestable) VALUES
('Seed', TRUE),
('Bulb', FALSE),
('Tuber', TRUE),
('Rhizome', FALSE),
('Seed', TRUE);

INSERT INTO PlantFamily (familyID, seedType, floralStructure) VALUES
(601, 'Seed', 'Simple'),
(602, 'Bulb', 'Simple'),
(603, 'Tuber', 'Simple'),
(604, 'Rhizome', 'Simple'),
(605, 'Seed', 'Complex');

INSERT INTO PlantInfo (species, familyID, prefEnvironment) VALUES
('Rose', 601, 'Mild'),
('Tulip', 602, 'Cool'),
('Lily', 603, 'Warm'),
('Fern', 604, 'Shady'),
('Cactus', 605, 'Dry');

INSERT INTO Plant (plantID, species, plantName) VALUES
(701, 'Rose', 'Rose'),
(702, 'Tulip', 'Tulip'),
(703, 'Lily', 'Lily'),
(704, 'Fern', 'Fern'),
(705, 'Cactus', 'Cactus');
```

```
INSERT INTO PlantLog (plantLogID, plantID, species, soilID, datePlanted, growth,
harvestDate) VALUES
(601, 701, 'Rose', 401, DATE '2025-04-01', 'Blooming', DATE '2025-05-01'),
(602, 702, 'Tulip', 402, DATE '2025-03-15', 'Seedling', null),
(603, 703, 'Lily', 403, DATE '2025-02-20', 'Growing', DATE '2025-04-20'),
(604, 704, 'Fern', 404, DATE '2025-01-10', 'Dormant', null),
(605, 705, 'Cactus', 405, DATE '2025-05-05', 'Flowering', DATE '2025-07-20');

INSERT INTO GardenLog_HAS_Plant (gardenLogID, plantID, species) VALUES
(501, 701, 'Rose'),
(502, 702, 'Tulip'),
(503, 703, 'Lily'),
(504, 704, 'Fern'),
(505, 705, 'Cactus');
```