CPSC 304 Project Cover Page

Milestone #: 2

Date: Friday Oct 20, 2023

Group Number: 127

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Madeleine Penner	57844268	D0b3b	madeleine.penner@yahoo.com
Kratika Rathi	38763710	c3l3v	kratikar2011@gmail.com
Will Beaulieu	24994386	e4v4v	willbeau02@gmail.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

1. A completed cover page (template on Canvas)

Attached

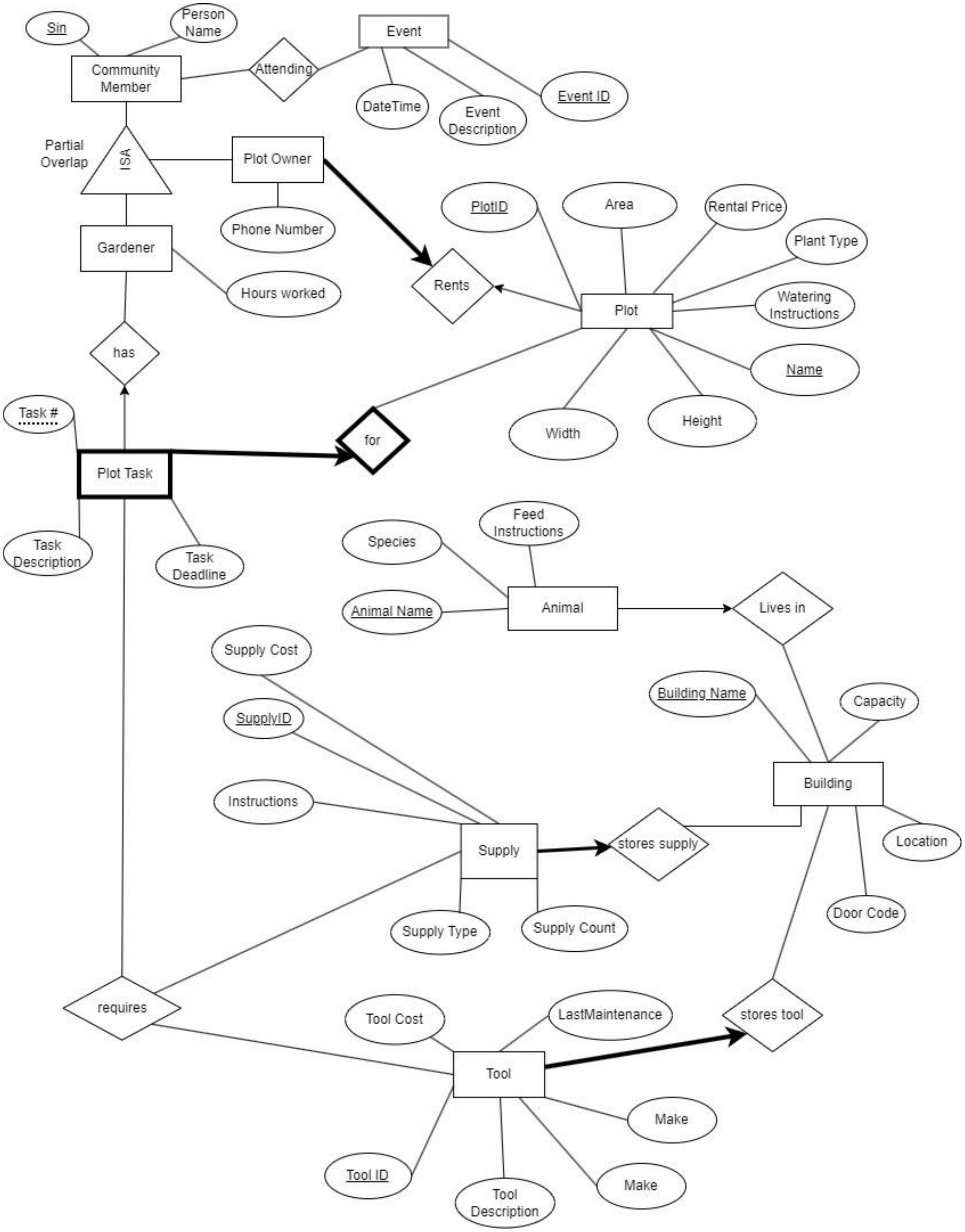
2. A brief (~2-3 sentences) summary of your project.

This project deals with a community garden management system. The database will allow for community members to be gardeners, plot owners, or both. Plot owners are responsible for managing their plot and will be able to create tasks for gardeners to complete on their plot.

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 have made many changes from our ER diagram one. We began to work through milestone 2 and we realized that we designed our ER diagram to already be free of redundancy. We've added and removed numerous entities from our ER diagram. Here's the major changes we've made:

- We removed the attribute 'Address' from the entity 'Plot Owner'.
- Added event entity for tracking farm events (Workshops, parties, etc)
- Removed schedule entity (We can store everything we need without it)
- Made plot tasks belong directly to gardeners due to removal of schedule
- Revamped tool so that each entry in the tool table will represent an individual tool entity rather than a type of tool. Subsequently removed count from stores tool relationship.
- Revamped Seed. Renamed to supply to store a wider variety of supplies.
- Removed Plant as its attributes work better as attributes of plot. We don't care to track each individual plant, we only want to know the type of plant on a plot, and how to care for that plant.
- Turned Shed entity into building entity to represent more than just sheds.
- Added animal entity to track animals living at the community garden, how to care for them, and where they live.
- Replaced the specialization attribute of Gardener entity with hours worked, as we felt hours worked was a more useful attribute to track.


Sin
Person Name
Community Member
Attending
Event
DateTime
Event Description
Event ID
Partial Overlap
ISA
Plot Owner
Phone Number
Gardener
Hours worked
Rents
PlotID
Area
Rental Price
Plant Type
Plot
Watering Instructions
Name
Width
Height
has
Task #
Plot Task
for
Task Description
Task Deadline
Species
Feed Instructions
Animal Name
Animal
Lives in
Supply Cost
SupplyID
Building Name
Capacity
Instructions
Building
Supply
stores supply
Location
Door Code
Supply Type
Supply Count
requires
Tool Cost
LastMaintenance
stores tool
Tool
Make
Tool ID
Tool Description
Make

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. 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.

Primary keys are underlined. Foreign keys are bolded.

Plot (PlotID: Integer, Area: Integer, RentalPrice: Integer, Width: Integer, Height: Integer, PlantType: varchar (50), WateringInstructions: varchar (200), **SIN**: char (9))

Plot Owner (**SIN**: char (9), Phone#: char (10), **PlotID**: Integer) -- Phone# is a candidate key

Gardener (**SIN**: char (9), HoursWorked: Integer)

CommunityMember (SIN: char (9), PersonName: varchar (50))

Event (EventID : Integer, EventDescription: varchar (200), DT: Date)

Attending (**EventID:** Integer**,** **SIN:** char (9))

PlotTask (Task#: Integer, **PlotID**: Integer, TaskDescription: varchar (200), Deadline: Date, **SIN:** char (9))

Supply (SupplyID: Integer, SupplyCost: Integer, SupplyType: varchar (20), Instructions: varchar (200), SupplyCount: Integer, **BuildingName:** varchar (50)) -- Building Name must not be null

Tool (ToolID: Integer, Model: varchar (50), Make: varchar (50), ToolDescription: varchar (200), ToolCost: Integer, LastMaintenance: Date, **BuildingName:** varchar (50)) -- Building Name must not be null

Building (BuildingName: varchar (50), Capacity: Integer, BuildingLocation: varchar (50), DoorCode: Char (4))

Requires (**Task#:** Integer, **PlotID** : Integer, **SupplyID:** varchar (20), **ToolID**: Integer)

Animal (AnimalName: varchar (20), FeedInstructions: varchar (200), Species: varchar (50), **BuildingName:** varchar (50))

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.

PlotID → Width, Height, Area, RentalPrice, PlantType, WateringInstructions, SIN
PlantType → WateringInstructions
Width, Height → Area, RentalPrice
Area → RentalPrice
SIN → PlotID, PersonName
SIN → HoursWorked
SIN → Phone#
Phone# → SIN, PlotID
EventID → EventDescription, DT
Task#, PlotID → TaskDescription, Deadline, SIN
SupplyID → SupplyCost, SupplyType, Instructions, SupplyCount, BuildingName
SupplyType → Instructions, SupplyCost
ToolID → Model, Make, ToolDescription, ToolCost, LastMaintenance, BuildingName
Make, Model → ToolDescription, ToolCost
BuildingName → Capacity, BuildingLocation, DoorCode
AnimalName → FeedInstructions, Species, BuildingName
Species → FeedInstructions

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.

Finding Minimal Basis

PlotID → Width
PlotID → Height
~~PlotID → Area~~
~~PlotID → RentalPrice~~
PlotID → PlantType
~~PlotID → WateringInstructions~~
PlotID → SIN
PlantType → WateringInstructions
Width, Height → Area
~~Width, Height → RentalPrice~~
Area → RentalPrice
SIN → Phone#
SIN → PlotID
Phone# → SIN
Phone# → PlotID
SIN → HoursWorked
SIN → PersonName
EventID → EventDescription
EventID → DT
Task#, PlotID → TaskDescription
Task#, PlotID → Deadline
Task#, PlotID → SIN
~~SupplyID → SupplyCost~~
SupplyID → SupplyType
~~SupplyID → Instructions~~
SupplyID → SupplyCount
SupplyID → BuildingName
SupplyType → Instructions
SupplyType → SupplyCost
ToolID → Model
ToolID → Make
~~ToolID → ToolDescription~~
~~ToolID → ToolCost~~
ToolID → LastMaintenance
ToolID → BuildingName
Make, Model → ToolDescription
Make, Model → ToolCost
BuildingName → Capacity
BuildingName → BuildingLocation
BuildingName → DoorCode
~~AnimalName → FeedInstructions~~

AnimalName → Species
AnimalName → BuildingName
Species → FeedInstructions

Post Reduction:

PlotID → Width, Height, PlantType, SIN
PlantType → WateringInstructions
Width, Height → Area
Area → RentalPrice
SIN → Phone#, PlotID
Phone# → SIN, PlotID
SIN → HoursWorked, PersonName
EventID → EventDescription, DT
Task#, PlotID → TaskDescription, Deadline, SIN
SupplyID → SupplyType, SupplyCount, BuildingName
SupplyType → Instructions, SupplyCost
ToolID → Model, Make, LastMaintenance, BuildingName
Make, Model → ToolDescription, ToolCost
BuildingName → Capacity, BuildingLocation, DoorCode
AnimalName → Species, BuildingName
Species → FeedInstructions

Decomposing

Plot:
Plot (PlotID: Integer, Area: Integer, RentalPrice: Integer, Width: Integer, Height: Integer, PlantType: varchar (50), WateringInstructions: varchar (200), **SIN**: char (9))

The following FDs violate 3nf so let's decompose.
PlantType → WateringInstructions
Width, Height → Area
Area → RentalPrice

Decompose over PlantType → WateringInstructions
Plot (PlotID: Integer, Area: Integer, RentalPrice: Integer, Width: Integer, Height: Integer, **PlantType**: varchar (50), **SIN**: char (9))
PlantInstructions(PlantType: varchar (50), WateringInstructions, varchar (200))

Decompose Plot over Area → RentalPrice
Plot (PlotID: Integer, Width: Integer, Height: Integer, PlantType: varchar (50), **SIN**: char (9), **Area:** Integer)
PlotPrices (Area: Integer, RentalPrice: Integer)

Decompose Plot over Width, Height
Plot (PlotID: Integer, **Width**: Integer, **Height**: Integer, PlantType: varchar (50), **SIN**: char (9))
Areas (Width: Integer, Height: Integer, **Area:** Integer**)**
PlotPrices (Area: Integer, RentalPrice: Integer)

Supply:
Supply (SupplyID: Integer, SupplyCost: Integer, SupplyType: varchar (20), Instructions: varchar (200), SupplyCount: Integer, **BuildingName:** varchar (50)) -- Building Name must not be null

The following FDs violate 3nf so let's decompose.
SupplyType → Instructions, SupplyCost

Decompose over SupplyType → Instructions, SupplyCost
Supply (SupplyID: Integer, **SupplyType**: varchar (20), **BuildingName:** varchar (50)) -- Building Name must not be null
SupplyInformation (SupplyType: varchar (20), SupplyCost: Integer, Instructions: varchar (200))

Tool:
Tool (ToolID: Integer, Model: varchar (50), Make: varchar (50), ToolDescription: varchar (200), ToolCost: Integer, LastMaintenance: Date, **BuildingName:** varchar (50)) -- Building Name must not be null

The following FDs violate 3nf so let's decompose.
Make, Model → ToolDescription, ToolCost

Decompose over Make, Model → ToolDescription, ToolCost

Tool (ToolID: Integer, **Model**: varchar (50), **Make**: varchar (50), LastMaintenance: Date, **BuildingName:** varchar (50)) -- Building Name must not be null
ToolInfo (Model: varchar (50), Make: varchar (50), ToolDescription: varchar (200), ToolCost: Integer)

Animal:
Animal (AnimalName: varchar (20), FeedInstructions: varchar (200), Species: varchar (50), **BuildingName:** varchar (50))
The following FDs violate 3nf so let's decompose.
Species → FeedInstructions

Decompose over Species → FeedInstructions

Animal (AnimalName: varchar (20), **Species**: varchar (50), **BuildingName:** varchar (50))

AnimalInstructions(Species: varchar (50), FeedInstructions: varchar (200))

The following tables do not violate 3nf:

PlotOwner (**SIN**: char (9), Phone#: char (10), **PlotID**: Integer) -- Phone# is a candidate key.

Gardener (**SIN**: char (9), HoursWorked: Integer)

CommunityMember (SIN: char (9), PersonName: varchar (50))

Event (EventID : Integer, EventDescription: varchar (200), DT: Date)

Building (BuildingName: varchar (50), Capacity: Integer, BuildingLocation: varchar (50), DoorCode: Char (4))

Requires (**Task#:** Integer, **PlotID** : Integer, **SupplyID:** varchar (20), **ToolID**: Integer)

PlotTask (Task#: Integer, **PlotID**: Integer, TaskDescription: varchar (200), Deadline: Date, **SIN:** char (9))

Final Tables:

PlotOwner (**SIN**: char (9), Phone#: char (10), **PlotID**: Integer) -- Phone# is a candidate key.

Gardener (**SIN**: char (9), HoursWorked: Integer)

CommunityMember (SIN: char (9), PersonName: varchar (50))

Event (EventID : Integer, EventDescription: varchar (200), DT: Date)

Building (BuildingName: varchar (50), Capacity: Integer, BuildingLocation: varchar (50), DoorCode: Char (4))

Requires (**Task#:** Integer, **PlotID** : Integer, **SupplyID:** varchar (20), **ToolID**: Integer)

Plot (PlotID: Integer, **Width**: Integer, **Height**: Integer, PlantType: varchar (50), **SIN**: char (9))

Areas (Width: Integer, Height: Integer, **Area:** Integer)

PlotPrices (Area: Integer, RentalPrice: Integer)

Supply (SupplyID: Integer, SupplyCount: Integer, SupplyCost: Integer, **SupplyType**: varchar (20), **BuildingName:** varchar (50)) -- Building Name must not be null

SupplyInformation (SupplyType: varchar (20), SupplyCost: Integer, Instructions: varchar (200))

Tool (ToolID: Integer, **Model**: varchar (50), **Make**: varchar (50), LastMaintenance: Date, **BuildingName:** varchar (50)) -- Building Name must not be null

ToolInfo (Model: varchar (50), Make: varchar (50), ToolDescription: varchar (200), ToolCost: Integer)

Animal (AnimalName: varchar (20), **Species**: varchar (50), **BuildingName:** varchar (50))

AnimalInstructions(Species: varchar (50), FeedInstructions: varchar (200))

PlotTask (Task#: Integer, **PlotID**: Integer, TaskDescription: varchar (200), Deadline: Date, **SIN:** char (9))

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.

```
CREATE TABLE PlotOwner (
  SIN CHAR(9),
  PhoneNum CHAR(10),
  PlotID INTEGER
  PRIMARY KEY (SIN),
  FOREIGN KEY (PlotID) REFERENCES Plot (PlotID)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (SIN) REFERENCES CommunityMember(SIN)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
CREATE TABLE Event (
  EventID INTEGER,
  EventDescription VARCHAR (200),
  DT DATE
  PRIMARY KEY(EventID)
);

CREATE TABLE Requires (
  TaskNum INTEGER,
  PlotID INTEGER,
  SupplyID INTEGER,
  ToolID INTEGER
  PRIMARY KEY (TaskNum,PlotID, SupplyID,ToolID),
  FOREIGN KEY (TaskNum) REFERENCES PlotTask(TaskNum)
 ON DELETE CASCADE
 ON UPDATE CASCADE,

  FOREIGN KEY (PlotID) REFERENCES Plot(PlotID)
 ON DELETE CASCADE
 ON UPDATE CASCADE,

  FOREIGN KEY (SupplyID) REFERENCES Supply(SupplyID)
  ON DELETE SET NULL
  ON UPDATE CASCADE,

  FOREIGN KEY (ToolID) REFERENCES Tool(ToolID)
 ON DELETE SET NULL
 ON UPDATE CASCADE
);


CREATE TABLE Gardener (
  SIN CHAR(9),
  HoursWorked INTEGER
  PRIMARY KEY (SIN),
  FOREIGN KEY (SIN) REFERENCES CommunityMember(SIN)
  ON DELETE CASCADE
```

```
    ON UPDATE CASCADE
);

CREATE TABLE CommunityMember (
    SIN char (9),
    PersonName VARCHAR (50)
    PRIMARY KEY (SIN),
);
CREATE TABLE Building (
    BuildingName VARCHAR (50),
    Capacity INTEGER,
    BuildingLocation VARCHAR (50),
    DoorCode CHAR (4)
    PRIMARY KEY (BuildingName)
);



CREATE TABLE PlotTask (
    TaskNum INTEGER,
    PlotID INTEGER,
    TaskDescription VARCHAR(200),
    Deadline DATE,
    SIN CHAR(9)
    PRIMARY KEY (TaskNum,PlotID),
    FOREIGN KEY (PlotID) REFERENCES Plot(PlotID)
   ON DELETE CASCADE,
    FOREIGN KEY (SIN) REFERENCES Gardener(SIN)
    ON DELETE SET NULL
    ON UPDATE CASCADE
);
CREATE TABLE AnimalInstructions (
    Species VARCHAR(50),
    FeedInstructions VARCHAR(200),
    PRIMARY KEY (Species)
);
CREATE TABLE Animal (
    AnimalName VARCHAR (20),
    Species VARCHAR (50),
    BuildingName VARCHAR (200),
    PRIMARY KEY (AnimalName),
    FOREIGN KEY (Species) REFERENCES AnimalInstructions(Species)
    FOREIGN KEY (BuildingName) REFERENCES Building (BuildingName)
    ON DELETE SET NULL
    ON UPDATE CASCADE
);

CREATE TABLE ToolInfo (
    Model VARCHAR(50),
    Make VARCHAR(50),
    ToolDescription VARCHAR(200),
    ToolCost INTEGER
    PRIMARY KEY (Model, Make)
);

CREATE TABLE Plot (
    PlotID INTEGER,
    Width INTEGER,
    Height INTEGER,
    PlantType VARCHAR(50),
    SIN CHAR(9)
    PRIMARY KEY (PlotID),
    FOREIGN KEY (Width, Height) REFERENCES Areas (Width, Height)
    ON UPDATE CASCADE,
    FOREIGN KEY (SIN) REFERENCES PlotOwner(SIN)
   ON UPDATE CASCADE
);

CREATE TABLE Tool (
    ToolID INTEGER,
    Model VARCHAR(50),
    Make VARCHAR(50),
    LastMaintenance DATE,
    BuildingName VARCHAR(50)
    PRIMARY KEY (ToolID),
    FOREIGN KEY (Model,Make) REFERENCES ToolInfo (Model,Make),
    FOREIGN KEY (BuildingName) REFERENCES BuildingName (BuildingName)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
CREATE TABLE SupplyInformation (
    SupplyType VARCHAR(20),
    SupplyCost INTEGER,
    Instructions VARCHAR(200)
    PRIMARY KEY (SupplyType)
);
CREATE TABLE Supply (
    SupplyID INTEGER,
    SupplyType VARCHAR(20),
    BuildingName VARCHAR(50),
    SupplyCount INTEGER
    PRIMARY KEY (SupplyID),
    FOREIGN KEY (SupplyType) REFERENCES SupplyInformation (SupplyType),
    FOREIGN KEY (BuildingName) REFERENCES BuildingName (BuildingName)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

CREATE Table PlotPrices (
    Area INTEGER,
    RentalPrice INTEGER
    PRIMARY KEY (Area)
);

CREATE Table Areas (
    Width INTEGER,
    Height INTEGER,
    Area INTEGER
    PRIMARY KEY(Width, Height)
    FOREIGN KEY (Area) REFERENCES PlotPrices(Area)
    ON DELETE CASCADE
    ON UPDATE CASCADE
)
```

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 on.

```
INSERT INTO PlotOwner(“888789888", 2368634471, 23)
INSERT INTO PlotOwner(“887515887", 2366744768, 40)
INSERT INTO PlotOwner(“895565895", 6045698761, 10)
INSERT INTO PlotOwner(“818786876", 2368634471, 55)
INSERT INTO PlotOwner(“823709808", 2368634471, 83)

INSERT INTO CommunityMember(“823709808", "Kratika”)
INSERT INTO CommunityMember(“895565895", "Madeleine”)
INSERT INTO CommunityMember(“475385473", "John”)
INSERT INTO CommunityMember(“818786876", "Jennifer”)
INSERT INTO CommunityMember(“564556334", "Allan”)
INSERT INTO CommunityMember(“888789888", "Raghav”)
INSERT INTO CommunityMember(“887515887", "Julie”)
INSERT INTO CommunityMember(“455334346", "Emily”)
INSERT INTO CommunityMember(“4165784432", "Will”)

INSERT INTO Gardener (“475385473”, 0)
INSERT INTO Gardener (“564556334”, 13)
INSERT INTO Gardener (“455334346”, 13)
INSERT INTO Gardener (“81886876", 30)
INSERT INTO Gardener (“82370908", 25)

INSERT INTO PlotTask(12,83,” Water all the Tomato Plants”, “20-10-23", “475385473")
INSERT INTO PlotTask(1,10,” Remove all Weeds”, “19-10-23", “823709808")
INSERT INTO PlotTask(55,55,” Harvest Fruits”, “19-10-23", “455334346")
INSERT INTO PlotTask(100,40,” Harvest Fruits”, “25-10-23", “455334346")
INSERT INTO PlotTask(58,55,” Replant Seeds”, “01-11-23", “455334346")

INSERT INTO AnimalInstructions(“Taby Cat”, “Do not feed, exclusively hunts mice”)
INSERT INTO AnimalInstructions(“Black Cat”, “cat food in barn, one scoop”)
INSERT INTO AnimalInstructions(“Squirrel”, “Do not give squirrels bird feed”)
INSERT INTO AnimalInstructions(“Bird”, “bird feed on top right corner of shed”)
INSERT INTO AnimalInstructions(“Dog”, “dogs can have scrap produce from compost bin”)
INSERT INTO AnimalInstructions(“Chicken”, “fill chicken feed tray with chicken feed every morning”)

INSERT INTO Animal (“fatty the fat cat”, “Taby Cat”, “Barn”)
INSERT INTO Animal (“nightmare”, “Black Cat”, null)
INSERT INTO Animal (“Rufus”, “Dog”, “Barn”)
INSERT INTO Animal (“Zoe”, “Dog”, “Barn”)
INSERT INTO Animal (“Princess Peck”, “Chicken”, “Chicken Coop”)

INSERT INTO Supply (1, “Cat Food”, “Barn”, 5)
INSERT INTO Supply (2, “Tomato Seed Pack”, “East Shed”, 15)
INSERT INTO Supply (3, “Magic Bean Sprouts”, “East Shed”, 2)
INSERT INTO Supply (4, Fertilizer bag”, “West Shed”, 10)
INSERT INTO Supply (5, “Watering Can”, “Gazebo”, 2)

INSERT INTO SupplyInformation(“Tomato Seed Pack”, 200, “Plant handful in fertilizer, water once per day”)
INSERT INTO SupplyInformation(“Magic Beans”, 3000, “plant and pray”)
INSERT INTO SupplyInformation(“Cat Food”, 5000, null)
INSERT INTO SupplyInformation(“Fertilizer bag”, 2000, “Add to new plot before planting”)
INSERT INTO SupplyInformation(“Watering Can”, 1000, “Water plants according to watering instructions”)

INSERT INTO Building (“Chicken Coop”, 30, “east side of barn”, null)
INSERT INTO Building (“Gazebo”, 50, “middle of the community garden”, null)
INSERT INTO Building (“Barn”, 45, “northmost end of community garden”, “3759”)
INSERT INTO Building (East Shed”, 60, “east end of the garden”, “2744”)
INSERT INTO Building (“West Shed”, 60, “west end of the garden”, “3664”)

INSERT INTO ToolInfo(“Firebolt 500”,” Wizarding Tools”,” Shovel”,50000)
INSERT INTO ToolInfo(“Nimbus 2000”,” Wizarding Tools”,” Axe”,100000)
INSERT INTO ToolInfo(“Golden Snitch”,” Grindlock’s Farming World”,” Plough”,250000)
INSERT INTO ToolInfo(“Zeus Power”,” Pete’s Farming Tool”,” Rake”,150000)
INSERT INTO ToolInfo(“Poseidon Water Hose”,” Greek Farming Supplies”,” Hose”,5100000)

INSERT INTO Tool (3, “Firebolt 500”,” Wizarding Tools”,”13-12-2002",” East Shed”)
INSERT INTO Tool (10, “Poseidon Water Hose”,” Greek Farming Supplies”,”16-06-2012",” East Shed”)
INSERT INTO Tool (50, “Zeus Power”,” Pete’s Farming Tool”,”13-01-2020",” West Shed”)
INSERT INTO Tool (100, “Golden Snitch”,” Grindlock’s Farming World”,”13-10-2023",” West Shed”)
INSERT INTO Tool (76, Nimbus 2000”,” Wizarding Tools”,”19-05-2018",” West Shed”)

INSERT INTO PlotPrices(2300,98000000)
INSERT INTO PlotPrices(150,100000)
INSERT INTO PlotPrices(2300, 98000000)
INSERT INTO PlotPrices(1500,1000000)
INSERT INTO PlotPrices(1110, 9000000)

INSERT INTO Areas (23,100,2300)
INSERT INTO Areas (15,10,150)
INSERT INTO Areas (111,10,1110)
INSERT INTO Areas (15,100,1500)
INSERT INTO Areas (23,100,2300)

INSERT INTO Requires (12, 83, null, 10)
INSERT INTO Requires (1,10, null, 50)
INSERT INTO Requires (58,55, 2, 100)
INSERT INTO Requires (58,55, 2, 50)
INSERT INTO Requires (58,55, null, 3)

INSERT INTO Plot(83, 15, 10, “Tomatoes”, “823709808")
INSERT INTO Plot (10, 111, 10, “Tomatoes”, “895565895")
INSERT INTO Plot(55, 23, 100, “Strawberries”, “818786876")
INSERT INTO Plot(40, 15, 100, “Tomatoes”, ‘887515887”)
INSERT INTO Plot(23, 23, 100, “Magic Beans”, “888789888”)

INSERT INTO Event (100, “Celebrating the Community Garden’s 25th Anniversary”,”7-09-23")
INSERT INTO Event (1, “Ribbon Cutting at the Community Garden’”,”7-09-98")
INSERT INTO Event (25, “Opening of the Barn Building”,”01-01-2016")
INSERT INTO Event (65, “Tommy’s 7th Birthday Party”,”23-04-2020")
INSERT INTO Event (12, “Farming Workshop for Beginners”,”05-11-2006")
```