University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ____2____

Date: ___2024-10-15_____

Group Number: ____15____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Kenny Niu	37151198	y7r6p	kennyn172@gmail.com
Camille Pureza	72136310	p1y7c	camillepureza@gmail.com
Irene Chang	85567402	d0r9y	irecha@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.)

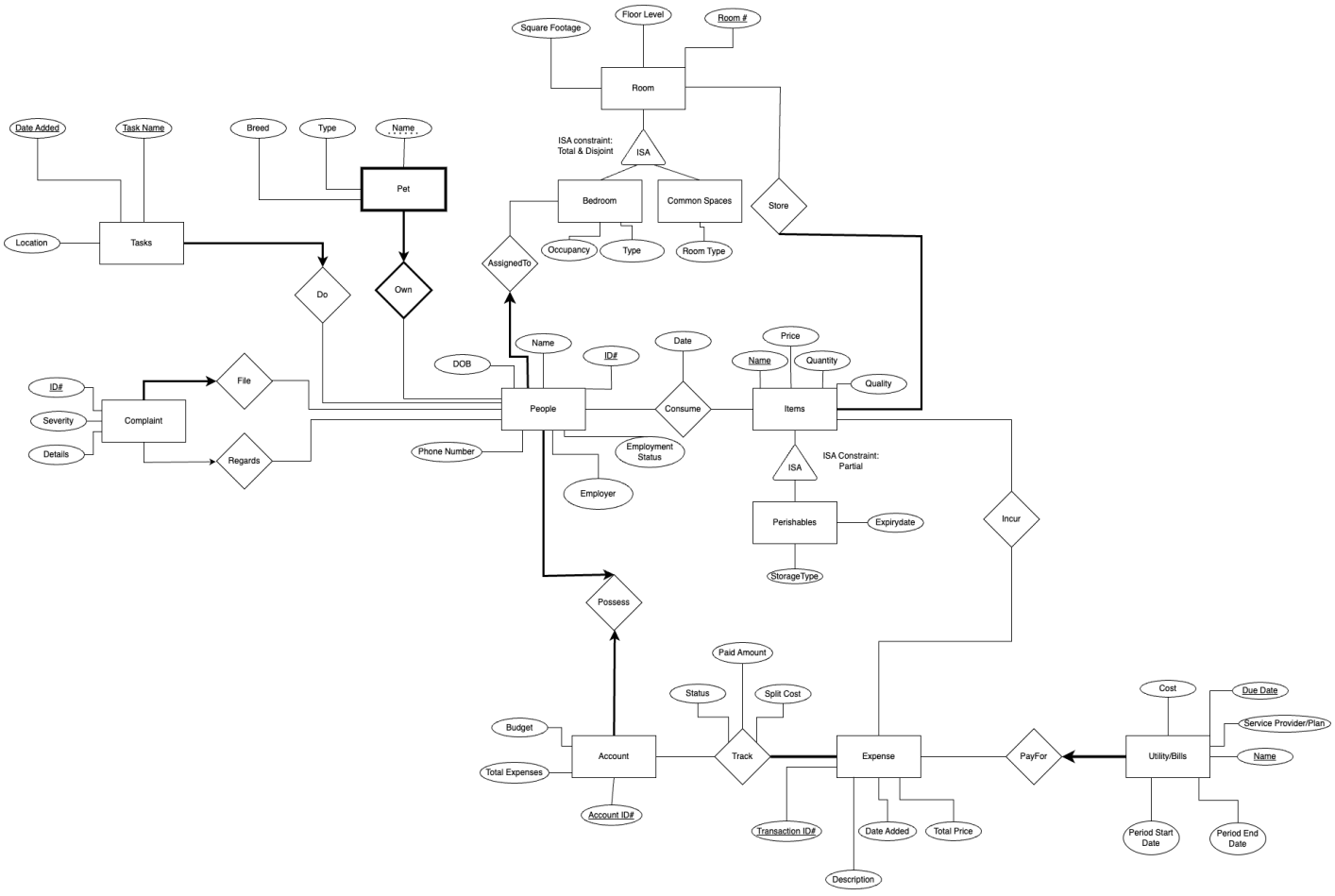
In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Milestone 2: Relational Schema, Normalization, SQL DDL, Tentative Queries

Brief Project Summary

Our project domain is based on a home management system. The application's features specifically help with a household's item storage, member complaints, and finances.

ER Diagram



Changes Made in ER diagram

Tasks/Chores Entity

1. Changed entity's name from "Tasks/Chores" to "Tasks"

We rarely used "chores" when we created the attributes and wrote our relational schemas (used just "Task name" rather than "Task/Chore name"). Also, it keeps our ER diagram concise.

2. Changed entity's primary key from (Task Name) to (Task Name, Date Added)

TA suggestion. A couple tasks will likely have the same name. However, no two tasks should have the same name and date added so this will be our primary key.

File Relationship

1. Changed cardinality constraint to one to many
2. Changed Complaint's participation constraint to total participation

A complaint should only be filed/reported by one person as the complaint will represent how that person was affected. If multiple people have the same complaint, having them file their own complaints will showcase how concerning this issue is. In addition, it facilitates users to only give their side of the story.
Complaint should have total participation as there can't be a complaint that wasn't filed by anyone.

Regards Relationship

1. Changed cardinality constraint to one to many

Similar to the File relationship, we want to have users give specific details on how a single person is involved in the complaint. If there is a complaint that involves two people, then users should give specific details on how EACH person affected them by filing two complaints.

People Entity

1. Removed ISA relationship and removed its subclasses

We thought about the functionality of our application and realized that there wouldn't be any added features for children specifically; only adults would benefit from our application. Therefore, there wasn't a purpose in having children involved in our ER diagrams.

2. Added Phone Number and Employer attribute

We added another non-primary attribute to add meaningful FDs (suggested by the milestone 2 rubric)

AssignedTo Relationship

1. Changed cardinality to one to many
2. Changed People's participation constraint to total participation

A person shouldn't be assigned to more than one bedroom (as that's unfair to other people). We also enforce the idea that everyone should have a bedroom (no one should sleep on the couch).

Room Entity

1. Added new Common Spaces subclass with Room Type attribute

This is to represent rooms that are shared among all household members. This allows us to put more information for the Store relationship and add more meaningful FDs. Incidentally, we still want to allow people to store things in their bedrooms. The room type attribute indicates whether it's the living room, laundry room, basement etc.

2. Changed ISA constraint to Total and Disjoint

It doesn't make sense for a room to be both a bedroom and common space (the living room is unfortunately not a suitable bedroom). In addition, a room can't just be a 'room.' Each room should have a purpose and if not a bedroom, should therefore have a room type.

3. Added new Type attribute to Bedroom subclass

We added another attribute to add meaningful FDs (suggested by the milestone 2 rubric)

Items Entity

1. (TA Suggestion – Not implemented) Kept primary key as (Name)

There shouldn't be items with the same name (e.g. there is no other substitute name for toilet paper). Our item names will be very general to items.

2. Added Quality attribute to the superclass and Storage Type attribute to Perishable subclass

We added another non-primary attribute to add meaningful FDs (suggested by the milestone 2 rubric)

Store Relationship

1. Changed Items' participation constraint to total participation

Every item should be placed somewhere in the house!

PayFor Relationship

1. Changed cardinality to one to many

There's no reason for utility costs or bills to be separated into multiple expenses (logically, makes things very complicated).

Track Relationship

1. Changed Expense's participation to total participation

Every expense should be tied to an account as someone had/has to pay it.

Utility/Bill Entity

1. Changed primary key to (Name, Due Date)

TA Suggestion. Also, it makes sense that there will be utility costs and bills with the same name. However, the composite should be unique and identify the utility/bill.

2. Added Service Provider/Plan, Period Start and End Date attributes

Bills usually have a billing period, which determines the cost of the bill. This was added to add meaningful FDs.

Consume

1. Added Date attribute

We decided it would make sense to have an attribute to log in when you've eaten something. This would help roommates understand who's been eating their snacks ..

Pet Weak Entity

1. Fixed notation for weak entity

TA Suggestion. We bolded the Pet entity box to follow class/textbook notation.

Other Changes

- '#' was added at end of any ID attribute

Relational Schema & Functional Dependencies

DoTask(taskName: VARCHAR, dateAdded: DATE, location: VARCHAR, **personID#**: INTEGER)

- Candidate keys: (taskName, dateAdded)
- Constraints:
  - personID# NOT NULL
- Attribute naming notes:
  - ID# for People changed to personID# for readability

FDs:
taskName, dateAdded → location, personID#

OwnPet(**ownerID#**: INTEGER, name: VARCHAR, breed: VARCHAR, type: VARCHAR)

- Candidate keys: (ownerID#, name)
- Constraints:
  - breed NOT NULL
  - type NOT NULL
- Attribute naming notes:
  - ID# for People changed to ownerID# for readability

FDs:
ownerID#, name → type, breed
Breed → type

FileComplaint(complaintID#: INTEGER, **fromPersonID#**: INTEGER, **toPersonID#:** INTEGER, severity: VARCHAR, details: VARCHAR)

- Candidate keys: complaintID#
- Constraints:
  - fromPersonID# NOT NULL

    - Details NOT NULL
    - Severity NOT NULL
- Attribute naming notes:
    - ID# for People referenced by fromPersonID# and toPersonID# to know which person filed the complaint and who that complaint is about
    - ID# for Complaint changed to complaintID# for readability

complaintID# → severity, details, fromPersonID#, toPersonID#
details → severity

People(ID#: INTEGER, name: VARCHAR, DOB: DATE, employer: VARCHAR, employmentStatus: VARCHAR, **assignedRoom**: INTEGER, phoneNumber: INTEGER)

- Candidate keys: (ID#), (phoneNumber)
- Constraints
    - phoneNumber UNIQUE
    - assignedRoom NOT NULL
    - Employer NOT NULL
    - employmentStatus NOT NULL
    - Name NOT NULL
- Attribute naming notes:
    - room# for Bedroom changed to assignedRoom for readability

ID# → name, DOB, employer, employmentStatus, assignedRoom, phoneNumber
phoneNumber → ID#, name, DOB, employer, employmentStatus, assignedRoom
employer → employmentStatus

Bedroom(room#: INTEGER, floorLevel: INTEGER, squareFootage: INTEGER, occupancy: INTEGER, type: VARCHAR)

- Candidate keys: room#
- Constraints:
    - type NOT NULL
    - occupancy NOT NULL
    - floorLevel NOT NULL
    - squareFootage NOT NULL
- Note:
    - Occupancy refers to max. occupancy in a room

room# → floorLevel, squareFootage, occupancy, type
type → occupancy

CommonSpaces(room#: INTEGER, floorLevel: INTEGER, squareFootage: INTEGER, roomType: VARCHAR)

- Candidate keys: room#
- Constraints

    - floorLevel NOT NULL
    - squareFootage NOT NULL
    - roomType NOT NULL

room# → floorLevel, squareFootage, roomType

Items(<u>name</u>: VARCHAR, price: FLOAT, quantity: INTEGER, expiryDate: DATE, quality: VARCHAR, storageType: VARCHAR)

- Candidate keys: name
- Constraint:
    - Quantity NOT NULL
    - Quality NOT NULL

name → price, quantity, expiryDate, quality, storageType
expiryDate → quality

Store(**<u>room#</u>**: INTEGER, **<u>itemName</u>**: VARCHAR, **storageType**: VARCHAR, **roomType**: VARCHAR)

- Candidate keys: (room#, itemName)
- Attribute naming notes:
    - Name from Items changed to itemName for readability

room#, itemName → storageType, roomType
roomType → storageType

Consume(**<u>personID#</u>**: INTEGER, **<u>itemName</u>**: VARCHAR, <u>date</u>: DATE)

- Candidate keys: (personID#, itemName, date)
- Attribute naming notes:
    - ID# for People changed to personID# for readability
    - Name for Items changed to itemName for readability

All FDs for Consume are trivial
personID#, itemName, date → personID#, itemName, date

Expense(<u>transactionID#:</u> INTEGER, description: VARCHAR, dateAdded: VARCHAR, totalPrice: FLOAT)

- Candidate keys: transactionID#
- Constraint:
    - totalPrice NOT NULL
    - Description NOT NULL
    - dateAdded NOT NULL

transactionID# → description, dateAdded, totalPrice
Description, dateAdded → totalPrice

UtilityBills(name: VARCHAR, cost: FLOAT, dueDate: DATE, **transactionID#:** INTEGER, serviceProviderPlan: VARCHAR, periodStartDate: DATE, periodEndDate: DATE)

- Candidate keys: (name, dueDate)
- Constraints
    - transactionID# NOT NULL
    - periodStartDate NOT NULL
    - periodEndDate NOT NULL
    - cost NOT NULL
    - serviceProviderPlan NOT NULL

name, dueDate → cost, transactionID#, periodStartDate, periodEndDate, serviceProviderPlan
name → serviceProviderPlan
periodStartDate, periodEndDate, serviceProviderPlan → cost

Incur(**itemName**: VARCHAR, **transactionID#**: INTEGER)

- Candidate keys: (itemName, transactionID#)
- Attribute naming notes:
    - Name for Items changed to itemName for readability

All FDs for Incur are trivial
itemName, transactionID# → itemName, transactionID#

PossessAccount(**ownerID#**: INTEGER, accountID#: INTEGER, budget: INTEGER, totalExpenses: FLOAT)

- Candidate keys: (ownerID#), (accountID#)
    - ownerID# NOT NULL
    - budget NOT NULL
    - totalExpenses NOT NULL
- Attribute naming notes:
    - ID# for People changed to ownerID# for readability

OwnerID# → accountID#, budget, totalExpenses
accountID# → ownerID#, budget, totalExpenses

Track(**accountID#**: INTEGER, **transactionID#**: INTEGER, status: VARCHAR, paidAmount: FLOAT, splitCost: FLOAT)

- Candidate keys: (accountID#, transactionID#)
- Constraints
    - Status NOT NULL
    - splitCost NOT NULL
    - paidAmount NOT NULL

accountID#, transactionID# → status, paidAmount, splitCost
paidAmount, splitCost → status

Note: total participation for expense entity cannot be represented in relational schema without ASSERTIONS (pre-Relational Model slide 107). We will implement this when we learn how to.

Normalization

DoTask
Candidate Key: taskName, dateAdded
FDs:

1. taskName, dateAdded → location, personID#

The table is already in BCNF/3NF as the LHS is a superkey.

OwnPet
Candidate Key: (ownerID#, name)

1. ownerID#, name → type, breed
2. Breed → type

The second FD violates BCNF as Breed is not a superkey of OwnPet.
Decomposing FD2:

OwnPet(**ownerID#**, name, breed) & PetTypes(**breed**, type)
Both tables are in BCNF.

FileComplaint
Candidate Key: complaintID#

1. complaintID# → severity, details, fromPersonID#, toPersonID#
2. details → severity

The third FD violates BCNF as details is not a superkey of FileComplaint.
Decompose on the third FD:

FileComplaint(complaintID#, **fromPersonID#, toPersonID#,** details) &
ComplaintSeverity(**details**, severity)
Both tables are in BCNF.

People
Candidate Key: ID#

1. ID# → name, DOB, employer, employmentStatus, phoneNumber, assignedRoom
2. phoneNumber → ID#, name, DOB, employer, employmentStatus, assignedRoom
3. employer → employmentStatus

The third FD violates BCNF because the LHS is not a superkey of People.

Decompose on employer → employmentStatus

People(ID#, name, DOB, employer, phoneNumber, **assignedRoom**) & Employment(**employer**, employmentStatus)

Both tables are in BCNF.

Bedroom
Candidate Key: room#

1. room# → floorLevel, squareFootage, occupancy, type
2. Type → occupancy

Minimal cover:

1. room# → floorLevel
2. room# → squareFootage
3. room# → type
4. Type → occupancy

The fourth FD is not in 3NF.

Decomposing FD2:
Bedroom(room#, floorLevel, squareFootage, type) & BedroomTypes(**type**, occupancy)
Both tables are in 3NF.

CommonSpaces
Candidate Key: room#

1. room# → floorLevel, squareFootage, roomType

The table is already in BCNF as the only FD is the primary key.

Items
Candidate Key: name

1. name → price, quantity, expiryDate, quality, storageType
2. expiryDate → quality

The second FD violates BCNF as expiryDate is not a superkey.
Decompose FD2:

Items(name, price, quantity, expiryDate, storageType) & ItemQuality(**expiryDate**, quality)

Store
Candidate Key: (room#, itemName)

1. room#, itemName → storageType, roomType
2. roomType → storageType

The second FD violates BCNF as the LHS is not a superkey.

Store(**room#, itemName**, **roomType**) & StoragePlaces(**roomType**, **storageType**)

The tables are in BCNF.

Consume
The table is already in BCNF.

Expense
Candidate Key: transactionID#

1. transactionID# → description, dateAdded, totalPrice
2. Description, dateAdded → totalPrice

The second FD is not in 3NF as the LHS is not a superkey and the RHS is not part of the minimal key.

Closures:
transactionID#$^+$={transactionID#, description, dateAdded, totalPrice}
description dateAdded$^+$={description, dateAdded, totalPrice}

We obtain the minimal cover of Expense.
First, we standardize the FDs and reduce.

1. transactionID# → description
2. transactionID# → dateAdded
3. transactionID# → totalPrice
4. Description, dateAdded → totalPrice

We can remove the third FD as we can derive totalPrice without this FD.

Our minimal cover:

1. transactionID# → description
2. transactionID# → dateAdded
3. Description, dateAdded → totalPrice

We decompose by the lossless join method.
The third FD violates 3NF as the LHS is not a superkey and the RHS is not part of the key.

Expense(transactionID#, description, dateAdded) & ExpensePrice(**description**, **dateAdded**, totalPrice)

All FDs are preserved; both tables are in 3NF.

UtilityBills
Candidate Key: (name, dueDate)

1. name, dueDate → cost, transactionID#, periodStartDate, periodEndDate

2. periodStartDate, periodEndDate, serviceProviderPlan → cost
3. name → serviceProviderPlan

The second and third FD violates 3NF since the LHS is not a superkey and the RHS is not part of the key.

We obtain the minimal cover of UtilityBills.

First, we standardize the FDs and reduce.

1. name, dueDate → cost
2. name, dueDate → transactionID#
3. name, dueDate → periodStartDate
4. name, dueDate → periodEndDate
5. name, dueDate → serviceProviderPlan
6. periodStartDate, periodEndDate, serviceProviderPlan → cost
7. name → serviceProviderPlan

We can remove the first and fifth FD as we can still obtain cost and serviceProviderPlan in (name, dueDate)'s closure.

Our minimal cover:

1. name, dueDate → transactionID#
2. name, dueDate → periodStartDate
3. name, dueDate → periodEndDate
4. periodStartDate, periodEndDate, serviceProviderPlan → cost
5. name → serviceProviderPlan

We decompose by the lossless join method.

UtilityBill(<u>name</u>, <u>dueDate</u>, **transactionID#,** periodStartDate, periodEndDate) & UtilityBillProvider(**<u>name</u>**, serviceProviderPlan) & UtilityBillCost(**<u>periodStartDate</u>**, **<u>periodEndDate</u>**, **<u>serviceProviderPlan</u>**, cost)

<u>Incur</u>
The table is already in BCNF as it only has two attributes

<u>PossessAccount</u>
This table is already in 3NF as both FDs' LHS are superkeys.

<u>Track</u>
Candidate Key: (accountID#, transactionID#)

1. accountID#, transactionID# → status, paidAmount, splitCost
2. paidAmount, splitCost → status

The second FD violates BCNF as the LHS is not a superkey of Track.
We decompose on paidAmount, splitCost → status

Track(accountID#, transactionID#, paidAmount, splitCost) & ExpenseStatus(**paidAmount**, **splitCost**, status)

Both tables are in BCNF.

SQL DDL Statements

Note: Oracle doesn't support ON UPDATE CASCADE. However, we add it to our statements for now.

```
CREATE TABLE DoTask (
        taskName VARCHAR,
        dateAdded DATE,
        location VARCHAR,
        personID# INTEGER NOT NULL,
        PRIMARY KEY (taskName, dateAdded),
        FOREIGN KEY (personID#) REFERENCES Person(ID#)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE OwnPet (
        ownerID# VARCHAR,
        name VARCHAR,
        breed VARCHAR NOT NULL,
        PRIMARY KEY (ownerID#, name),
        FOREIGN KEY (ownerID#) REFERENCES Person(ID#)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE PetTypes (
        breed VARCHAR,
        type VARCHAR NOT NULL,
        PRIMARY KEY (breed),
        FOREIGN KEY (breed) REFERENCES OwnPet(breed)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE FileComplaint (
```

```
        complaintID# INT PRIMARY KEY,
        fromPersonID# INT NOT NULL,
        toPersonID# INT,
        details VARCHAR NOT NULL,
        FOREIGN KEY (fromPersonID#) REFERENCES Person(ID#)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (toPersonID#) REFERENCES Person(ID#)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE ComplaintSeverity (
        details VARCHAR,
        severity VARCHAR NOT NULL,
        PRIMARY KEY (details),
        FOREIGN KEY (details) REFERENCES FileComplaint(details)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE People (
        ID# INT PRIMARY KEY,
        name VARCHAR NOT NULL,
        DOB DATE,
        employer VARCHAR NOT NULL,
        phoneNumber INT UNIQUE,
        assignedRoom INT NOT NULL,
        FOREIGN KEY (assignedRoom) REFERENCES Bedroom(room#)
                ON DELETE NO ACTION
                ON UPDATE CASCADE
);

CREATE TABLE Employment (
        employer VARCHAR PRIMARY KEY,
        employmentStatus VARCHAR NOT NULL,
        FOREIGN KEY (employer) REFERENCES People(employer)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Bedroom (
        room# INT PRIMARY KEY,
        floorLevel INT NOT NULL,
```

```
        squareFootage INT NOT NULL,
        type VARCHAR NOT NULL
);

CREATE TABLE BedroomTypes (
        type VARCHAR PRIMARY KEY,
        occupancy INT NOT NULL,
        FOREIGN KEY (type) REFERENCES Bedroom(type)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE CommonSpaces (
        room# INT,
        floorLevel INT NOT NULL,
        squareFootage INT NOT NULL,
        roomType VARCHAR NOT NULL,
        PRIMARY KEY (room#)
);

CREATE TABLE Items (
        name VARCHAR,
        price INT,
        quantity INT NOT NULL,
        expiryDate DATE,
        storageType VARCHAR,
        PRIMARY KEY (name)
);

CREATE TABLE ItemQuality (
        expiryDate DATE,
        quality VARCHAR NOT NULL,
        PRIMARY KEY (expiryDate),
        FOREIGN KEY (expiryDate) REFERENCES Items(expiryDate)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Store (
        room# INT,
        itemName VARCHAR,
        roomType VARCHAR,
        PRIMARY KEY (room#, itemName),
        FOREIGN KEY (room#, roomType) REFERENCES CommonSpaces(room#, roomType)
```

```
                ON DELETE SET NULL
                ON UPDATE CASCADE,
        FOREIGN KEY (itemName) REFERENCES Items(name)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE StoragePlaces(
        roomType VARCHAR,
        storageType VARCHAR,
        PRIMARY KEY (roomType),
        FOREIGN KEY (roomType) REFERENCES CommonSpaces(roomType)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (storageType) REFERENCES Items(storageType)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Consume (
        personID# INT,
        itemName VARCHAR,
        date DATE,
        PRIMARY KEY (personID#, itemName),
        FOREIGN KEY (personID#) REFERENCES People(ID#)
                ON DELETE CASCADE
                ON UPDATE,
        FOREIGN KEY (itemName) REFERENCES Items(name)
                ON DELETE CASCADE
                ON UPDATE
);

CREATE TABLE Expense(
        transactionID# INT PRIMARY KEY,
        description VARCHAR NOT NULL,
        dateAdded DATE NOT NULL
);

CREATE TABLE ExpensePrice(
        description VARCHAR,
        dateAdded DATE,
        totalPrice FLOAT NOT NULL,
        PRIMARY KEY (description, dateAdded),
        FOREIGN KEY (description, dateAdded) REFERENCES
```

```
        Expense(description, dateAdded)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);


CREATE TABLE UtilityBill(
        name VARCHAR,
        dueDate DATE,
        transactionID# INTEGER NOT NULL,
        periodStartDate DATE NOT NULL,
        periodEndDate DATE NOT NULL,
        PRIMARY KEY (name, dueDate),
        FOREIGN KEY (transactionID#) REFERENCES Expense (transactionID#)
                ON DELETE NO ACTION
                ON UPDATE CASCADE
);

CREATE TABLE UtilityBillProvider(
        name VARCHAR PRIMARY KEY,
        serviceProviderPlan VARCHAR NOT NULL,
        FOREIGN KEY (name) REFERENCES UtilityBill(name)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE UtilityBillCost(
        periodStartDate DATE,
        periodEndDate DATE,
        cost FLOAT NOT NULL,
        PRIMARY KEY (periodStartDate, periodEndDate),
        FOREIGN KEY (periodStartDate, periodEndDate) REFERENCES
                UtilityBill(periodStartDate, periodEndDate)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Incur(
        itemName VARCHAR,
        transactionID# INT,
        PRIMARY KEY (itemName, transactionID#),
        FOREIGN KEY (itemName) REFERENCES Items (name)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
```

```
        FOREIGN KEY (transactionID#) REFERENCES Expense (transactionID#)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE PossessAccount(
        accountID# INT PRIMARY KEY,
        ownerID# INT NOT NULL,
        budget FLOAT NOT NULL,
        totalExpenses FLOAT NOT NULL,
        UNIQUE (ownerID#),
        FOREIGN KEY (ownerID#) REFERENCES People (ID#)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Track (
        accountID# INT,
        transactionID# INT,
        paidAmount FLOAT NOT NULL,
        splitCost FLOAT NOT NULL,
        PRIMARY KEY (accountID#, transactionID#),
        FOREIGN KEY (accountID#) REFERENCES PossessAccount
                ON DELETE SET NULL
                ON UPDATE CASCADE,
        FOREIGN KEY (transactionID#) REFERENCES Expense
                ON DELETE CASCADE
                ON UPDATE CASCADE
);


CREATE TABLE ExpenseStatus(
        paidAmount FLOAT,
        splitCost FLOAT,
        status VARCHAR NOT NULL,
        PRIMARY KEY (paidAmount, splitCost),
        FOREIGN KEY (paidAmount, splitCost) REFERENCES Track(paidAmount, splitCost)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

```
INSERT
INTO People(ID#, name, DOB, employer, phoneNumber, assignedRoom)
VALUES (123456789, 'John', '1988-01-01', 'Walmart', '604-111-1111', 1),
        (012345678, 'Mary', '1998-11-22', 'Coffee and Vanilla Cafe', '778-222-2222', 2),
        (001234567, 'Larry', '1988-01-01', 'Coco's', '604-333-3333', 3),
        (000123456, 'Sophia', '1988-01-01', 'SAP', '604-444-4444', 4),
        (000012345, 'Lisa', '1988-01-01', 'Vancity', '604-555-5555', 5),
        (000001234, 'Joe', '1847-01-01', 'None', '604-111-5542', 5)

INSERT
INTO Employment(employer, employmentStatus)
VALUES ('Walmart', 'Employed'),
        ('Coco's', 'Employed'),
        ('Coffee and Vanilla Cafe', 'Employed'),
        ('SAP', 'Employed'),
        ('Vancity', 'Employed'),
        ('None', 'Unemployed')

INSERT
INTO BedroomTypes(type, occupancy)
VALUES ('Twin', 1),
        ('Two Twin', 2),
        ('Double', 2),
        ('Queen', 2),
        ('King', 3)

INSERT
INTO Bedroom(bedroom#, floorLevel, squareFootage, type)
VALUES (1, 1, 49, 'Twin'),
        (2, 1, 86, 'Double),
        (3, 2, 81, '2 Twin'),
        (4, 2, 109, 'King'),
        (5, 2, 96, 'Queen Bed')

INSERT
INTO CommonSpaces(room#, floorLevel, squareFootage, roomType)
VALUES (6, 1, 180, 'Kitchen'),
        (7, 1, 100, 'Laundry'),
        (8, 1, 50, 'Bathroom'),
        (9, 2, 50, 'Bathroom'),
        (10, 1, 210, 'Living room'),
        (11, 1, 160, 'Dining room'),
        (12, 1, 20, 'Pantry room'),
```

```
        (13, 0, 500, ‘Basement’)

INSERT
INTO PetTypes(breed, type)
VALUES (‘Ragdoll, ‘Cat’),
        (‘Siamese’, ‘Cat’),
        (‘Bombay’, ‘Cat’),
        (‘Australian Shepard’, ‘Dog’),
        (‘Betta’, ‘Fish’)

INSERT
INTO OwnPet(ownerID#, name, breed)
VALUES (123456789, ‘Henry’, ‘Australian Shepard’),
        (123456789, ‘Oliver’, ‘Bombay’),
        (000123456, ‘Bluey’, ‘Betta’),
        (000123456, ‘Loki’, ‘Bombay’),
        (000123456, ‘Moon’, ‘Siamese’)

INSERT
INTO Items(name, price, quantity, expiryDate, storageType)
VALUES (‘Dog Food’, 70.99, 2, 2024-11-22, ‘Fridge’),
        (‘Cat Food’, 50.99, 15, 2024-11-28 ‘Fridge’),
        (‘Toilet Paper’, 23.99, 2, NULL, NULL),
        (‘Dish soap’, 7.50, 5, NULL, NULL),
        (‘Apples’, 1.99, 6, 2024-10-10, ‘Open Shelving’),
        (‘Bananas’, 2.50, 4, 2024-09-22, ‘Open Shelving’),
        (‘Oranges’, 12.99, 10, 2024-08-11, ‘Open Shelving’)

INSERT
INTO ItemQuality(expiryDate, quality)
VALUES (2024-11-22, ‘Excellent’),
        (2024-11-28, ‘Excellent’),
        (2024-10-10, ‘Good’),
        (2024-09-22, ‘Bad’),
        (2024-08-11, ‘Bad’)

INSERT
INTO Consume(personID#, itemName, date)
VALUES (123456789, ‘Dog Food’, 2024-05-30),
        (123456789, ‘Cat Food’, 2024-05-31),
        (012345678, ‘Toilet Paper’, 2022-01-30),
        (001234567, ‘Dish soap’, 1984-05-01),
        (000012345, ‘Bananas’, 2023-09-23)
```

```
INSERT
INTO StoragePlaces(roomType, storageType)
VALUES (‘Kitchen’, ‘Fridge’),
        (‘Pantry’, ‘Open Shelving’),
        (‘Living room’, ‘Open Shelving’),
        (‘Dining room’, ‘Open Shelving’),
        (‘Basement’, ‘Fridge’)

INSERT
INTO Store(room#, itemName, roomType)
VALUES (6, ‘Cat Food’, ‘Kitchen’),
        (6, ‘Dog Food’, ‘Kitchen’),
        (11, ‘Apples’, ‘Dining room’),
        (11, ‘Bananas, ‘Dining room’),
        (11, ‘Oranges’, ‘Dining room’)

INSERT
INTO DoTask(taskName, dateAdded, location, personID#)
VALUES (‘Clean Bathroom’, 2024-12-01, ‘Bathroom’, 123456789),
        (‘Wash Dishes’, 2024-01-27, ‘Kitchen’, 012345678),
        (‘Cook Food’, 2024-02-23, ‘Kitchen’, 000012345),
        (‘Walk dog’, 2024-07-12, ‘Outside’, 123456789),
        (‘Mow Lawn’, 2024-02-01, ‘Backyard’, 012345678)

INSERT
INTO UtilityBillProvider(name, serviceProviderPlan)
VALUES (‘Water Bill’, ‘BC Hydro’)
        (‘Electricity Bill’, ‘FortisBC’),
        (‘Gas Bill’, ‘FortisBC’),
        (‘Family Plan Phone Bill’, ‘Telus’),
        (‘Joe Phone Bill’, ‘Rogers’)

INSERT
INTO UtilityBill(name, dueDate, transactionID#, periodStartDate, periodEndDate)
VALUES (‘Water Bill’, 2024-11-22, 4, 2024-09-02, 2024-09-29),
        (‘Electricity Bill’, 2024-11-18, 4, 2024-09-01, 2024-10-28),
        (‘Gas Bill’, 2024-11-18, 4, 2024-09-03, 2024-10-31),
        (‘Family Plan Phone Bill’, 2024-11-18, 6, 2024-09-01, 2024-10-28),
        (‘Joe Phone Bill’, 2024-11-05, 7, 2024-09-01, 2024-10-28)

INSERT
INTO UtilityBillCost(periodStartDate, periodEndDate, serviceProviderPlan cost)
VALUES (2024-09-02, 2024-09-29, ‘BC Hydro’, 120.05),
        (2024-09-01, 2024-10-28, ‘FortisBC’, 180.75),
```

```
        (2024-09-03, 2024-10-31, 'FortisBC', 87.33),
        (2024-09-01, 2024-10-28, 'Telus', 160.05),
        (2024-09-01, 2024-10-28, 'Rogers', 59.61)

INSERT
INTO Expense(transactionID#, description, dateAdded)
VALUES (1, 'Shared Pet Food', 2024-08-10),
        (2, 'Necessities', 2024-09-20),
        (3, 'Shared Snacks', 2024-09-20),
        (4, 'November Utility Bills', 2024-11-13),
        (5, 'Rent Bill', 2024-10-24),
        (6, 'Family Plan Phone Bill', 2024-11-12),
        (7, 'Joe Phone Bill', 2024-11-10)

INSERT
INTO Incur(itemName, transactionID#)
VALUES ('Dog Food', 1),
        ('Cat Food', 1),
        ('Toilet Paper', 2),
        ('Apples', 3),
        ('Bananas', 3)

INSERT
INTO ExpensePrice(description, dateAdded, totalPrice)
VALUES ('Shared Pet Food', 2024-08-10, 103.88),
        ('Necessities', 2024-09-20, 50.60),
        ('Shared Snacks', 2024-09-20, 32.80),
        ('November Utility Bills', 2024-11-13, 388.13),
        ('Rent Bill', 2024-10-24, 960.00),
        ('Family Plan Phone Bill', 2024-11-12, 160.05),
        ('Joe Phone Bill', 2024-11-10, 59.61)

INSERT
INTO PossessAccount(accountID#, ownerID#, budget, totalExpenses)
VALUES (32, 123456789, 100500.52, 5000.11),
        (31, 012345678, 200000.70, 0),
        (36, 001234567, 1400000.21, 24.99),
        (42, 000123456, 4546.59, 5),
        (1, 000012345, 50.11, 6000000),
        (56, 000001234, 100000000, 4444)

INSERT
INTO Track(accountID#, transactionID#, paidAmount, splitCost)
VALUES (32, 1, 12.99, 17.31),
```

```
        (32, 2, 1.11, 8.43),
        (56, 7, 0, 9.94),
        (1, 4, 1.20, 64.69),
        (42, 3, 5.47, 5.47)

INSERT
INTO ExpenseStatus(paidAmount, splitCost, status)
VALUES (12.99, 17.31, 'unpaid'),
        (1.11, 8.43, 'unpaid'),
        (0, 9.94, 'unpaid'),
        (1.20, 64.69, 'unpaid'),
        (5.47, 5.47, 'paid')

INSERT
INTO ComplaintSeverity(details, severity)
VALUES ('Smelly room', 'minor'),
        ('Noise complaint', 'moderate'),
        ('Broke window', 'major'),
        ('Killed my roommate', 'extreme'),
        ('Didn't flush toilet', 'moderate')

INSERT
INTO FileComplaint(complaintID#, fromPersonID#, toPersonID#, details)
VALUES (6123515, 012345678, 123456789, 'Didn't flush toilet'),
        (2, 123456789, 012345678, 'Noise complaint'),
        (424, 012345678, 123456789, 'Smelly room'),
        (1616, 012345678, 123456789, 'Broke window'),
        (5, 123456789, 012345678, 'Killed my roommate')
```