University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: October 19th 2023

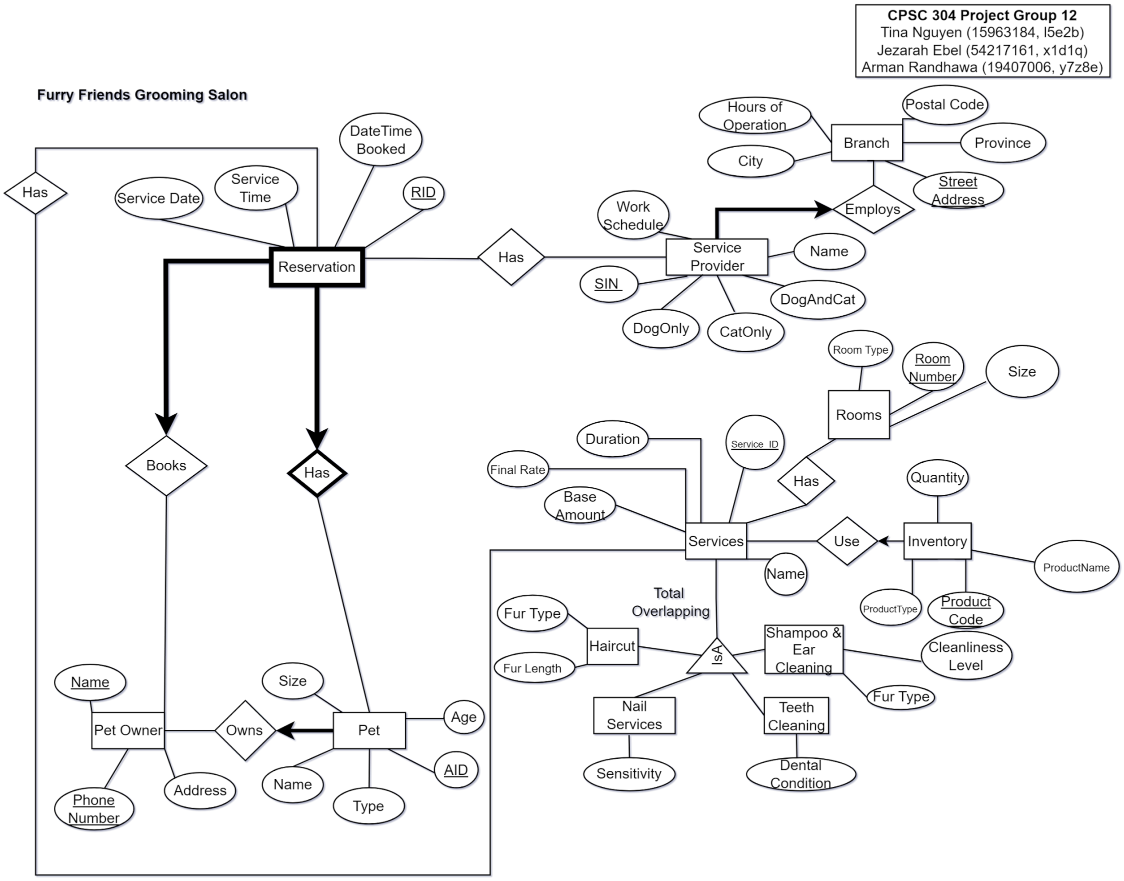
Group Number: 12

Name	Student Number	CS Alias (Userid)	Preferred Email Address
Tina Nguyen	15963184	l5e2b	nguyen.tina25@yahoo.com
Jezarah Ebel	54217161	x1d1q	jebel@student.ubc.ca
Arman Randhawa	19407006	y7z8e	armanr09@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 email 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 brief (~2-3 sentences) summary of your project.
   Our project is on designing a database for a chain pet salon, “Furry Friends Grooming Salon” where pet owners can book reservations for their pets to receive haircuts, nail services, teeth cleaning and shampoo and ear cleaning services. Service providers would be able to view their appointments for the day, the room for the service they will be providing and .
2. The ER diagram you are basing your item #3 (below) on. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why.



Based on the feedback from the project mentor, we agreed that there should be total participation of entity Reservation to Pet Owner because every reservation must have been made by a Pet Owner to exist in our system.

To remove redundancy from our ER we removed Provides, a tertiary relationship between Service, Service Provider and Pet, since this information is already stored in relationships between Reservation and Service, Service Provider and Pet.

In order to go through the normalisation process, some attributes were added to our entity sets. To Branch, we added Province and PostalCode. To Service Provider, we added DogOnly, CatOnly and CatAndDog attributes. We also added attributes ServiceID to Services to make for a more logical primary key for the entity as well as for establishing sensical FDs. Inventory also saw an addition of an attribute, ProductName, and adjusted naming (Product prefixes). Rooms saw the addition of a Size attribute as well as adjusted naming (Room prefix).

3. 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: dom1, attr2: dom2, ...)). 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.
        i. **PetOwner**(POName: VARCHAR(30), POAddress: VARCHAR(50), POPhoneNumber: CHAR(10))
            PK = {POName, POPhoneNumber}
        ii. **OwnsPet**(AID: INT, PName: VARCHAR(30) , PSize: ENUM{'small', 'medium', 'large'}, PAge: INT, PType: ENUM{'cat', 'dog'}, POName: VARCHAR(30), POPhoneNumber: CHAR(10))
            PK = {AID}
            FK = {POName, POPhoneNumber} to PetOwner
        iii. **ReservationHas**(RID: INT, ServiceDate: DATE, ServiceTime: TIME, DateTimeBooked: DATETIME, AID: INT, POPhoneNumber: CHAR(10), POName: VARCHAR(30), ServicesID: INT, SIN: INT)

PK = {RID}
FK = {AID} to OwnsPet, {POName, POPhoneNumber) to PetOwner, {ServicesID} to Services, {SIN} to ServiceProvider

iv. **Branch**(BranchStreetAddress: VARCHAR(50), BranchCity: VARCHAR(30), BranchProvince: VARCHAR(30), BranchPostalCode: CHAR(7), BranchName: VARCHAR(30), HoursOfOperation: VARCHAR(50))

CK = {BranchStreetAddress, BranchCity}, {BranchStreetAddress, BranchPostalCode}
PK = {BranchStreetAddress, BranchCity}

v. **ServiceProvider**(SIN: INT, SPName: VARCHAR(30), WorkSchedule: VARCHAR(50), DogOnly: BOOL, CatOnly: BOOL, DogAndCat: BOOL, BranchStreetAddress: VARCHAR(50), BranchCity: VARCHAR(30))

CK = {SIN}
PK = {SIN}
FK = {BranchStreetAddress, BranchCity} to Branch

vi. **Rooms(**RoomNumber: INT, RoomType: VARCHAR(30), Size: ENUM(‘large’, ‘medium’, ‘small’)**)**

PK={RoomNumber}

vii. **RoomsHasServices(**RoomNumber: INT, ServicesID: INT**)**

PK={RoomNumber, ServicesID}
FKs={RoomNumber} references Room, {ServicesID} references Services

viii. **Services_Inventory(**ProductCode: INT, Quantity: ENUM(‘empty’, ‘low’, ‘half’, ‘full’), ProductName: VARCHAR(30), ProductType: VARCHAR(30), ServicesID: INT**)**

PK={ProductCode}
FK={ServicesID} references Services

ix. **Services**(ServicesID: INT, ServicesName: VARCHAR(30), BaseAmount: FLOAT, Duration: INT, FinalRate: FLOAT**)**

PK={ServicesID}

x. **Haircut**(ServicesID: INT, HaircutFurType: ENUM('soft', 'rugged', 'hairless'), FurLength: FLOAT)

PK={ServicesID}

FK={ServicesID} references Services

xi. **NailServices**(ServicesID: INT, Sensitivity: ENUM('low', 'medium', 'high')

PK={ServicesID}

FK={ServicesID} references Services

xii. **TeethCleaning**(ServicesID: INT, DentalCondition: ENUM('excellent', 'good', 'poor', 'very poor'))

PK={ServicesID}

FK={ServicesID} references Services

xiii. **ShampooAndEarCleaning**(ServicesID: INT, Cleanliness: ENUM('excellent', 'good', 'poor', 'very poor'), SECFurType: ENUM('soft', 'rugged', 'hairless'))

PK={ServicesID}

FK={ServicesID} references Services

4. 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.

      Note: In your list of FDs, there must be some kind of valid FD other than 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 normalisation exercise. Your design must go through a normalisation process.

      For PetOwner:

      POPhoneNumber, POName→POAddress

      For OwnsPet:

AID → PType, PAge, PName, PSize, POName, POPhoneNumber

For ReservationHas:

RID → DateTimeBooked, ServiceTime, ServiceDate, AID, POPhoneNumber, POName, ServiceName, ServiceDuration, ServiceBase, SIN

For Branch: (added attributes BranchProvince and BranchPostalCode)

BranchStreetAddress, BranchCity → BranchName

BranchStreetAddress, BranchCity → BranchProvince

BranchStreetAddress, BranchCity, BranchProvince → BranchPostalCode

BranchPostalCode → BranchCity, BranchProvince

BranchName → HoursOfOperation

For Service Provider: (added attributes DogOnly, CatOnly, and DogAndCat in order to go through normalisation process)

SIN → SPName, WorkSchedule, BranchStreetAddress, BranchCity, DogOnly, CatOnly, DogAndCat

DogOnly, CatOnly → DogAndCat

For Rooms (added attribute Size):

RoomNumber → RoomType, Size

RoomType → Size

For Services_Inventory (added attribute ProductName):

ProductCode → ProductType, Quantity

ProductName → ProductType

For Services (added attribute ServicesID):

ServicesID → BaseAmount, Name, Duration

ServicesID, Duration, BaseAmount → FinalRate

For Haircut:

ServicesID → HaircutFurType, FurLength

For NailServices:

ServicesID → Sensitivity

For TeethCleaning:

ServicesID → DentalCondition

For ShampooAndEarCleaning:

ServicesID ⟶ SECFurType, Cleanliness

5. 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.

Normalization for PetOwner, OwnsPet, and ReservationHas is not needed as they are already in BCNF. The left hand side of all the functional dependencies for these tables are keys.

Normalization for Branch:

Branch(BranchStreetAddress: VARCHAR(50), BranchCity: VARCHAR(30), BranchProvince: VARCHAR(30), BranchPostalCode: CHAR(7), BranchName: VARCHAR(50), HoursOfOperation: VARCHAR)

BranchStreetAddress, BranchCity -> BranchName
BranchStreetAddress, BranchCity -> BranchProvince
BranchStreetAddress, BranchCity, BranchProvince -> BranchPostalCode
BranchPostalCode -> BranchCity, BranchProvince
BranchName -> HoursOfOperation

Let's first find all the minimal keys:

Left	Middle	Right
BranchStreetAddress	BranchPostalCode, BranchCity, BranchProvince, BranchName	HoursOfOperation

BranchStreetAddress+ = {BranchStreetAddress}
BranchStreetAddress, BranchPostalCode+ = {BranchStreetAddress, BranchPostalCode, BranchCity, BranchProvince, BranchName, HoursOfOperation}
BranchStreetAddress, BranchCity+ = {BranchStreetAddress, BranchCity, BranchName, BranchProvince, BranchPostalCode, HoursOfOperation}
BranchStreetAddress, BranchProvince+ = {BranchStreetAddress, BranchProvince}
BranchStreetAddress,BranchName+ = {BranchStreetAddress, BranchName, HoursOfOperation}

The minimal keys are BranchStreetAddress, BranchPostalCode and BranchStreetAddress, BranchCity; this is in line with our Candidate keys.

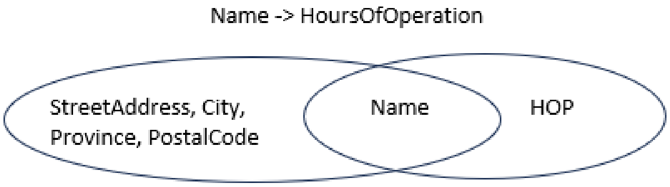
The FD BranchName -> HoursOfOperation is not in 3NF. To decompose to 3NF, find the minimum cover to transform our FDs to be as small as possible.

1. Put FDs in standard form: BranchPostalCode -> BranchCity, BranchProvince
   BranchPostalCode -> City and
   BranchPostalCode -> Province
2. Minimise LHS of each FD: Since BranchStreetAddress, BranchCity -> BranchProvince and BranchStreetAddress, BranchCity, BranchProvince -> BranchPostalCode we can just have BranchStreetAddress, BranchCity -> BranchPostalCode
3. Delete Redundant FDs: Since BranchStreetAddress, BranchCity -> BranchPostalCode and BranchStreetAddress, BranchCity -> BranchProvince; BranchPostalCode -> BranchCity and BranchPostalCode -> BranchProvince are redundant and can be removed.

Our Minimal Cover for Branch':

BranchStreetAddress, BranchCity -> BranchName
BranchStreetAddress, BranchCity -> BranchProvince
BranchStreetAddress, BranchCity -> BranchPostalCode
BranchName -> HoursOfOperation

The FD BranchName -> HoursOfOperation is still not in 3NF so decompose like in class.



B1(BranchName, HoursOfOperation), B2(BranchStreetAddress, BranchCity, BranchProvince, BranchPostalCode, BranchName)

BranchName is a key for B1 since BranchName+={BranchName, HoursOfOperation} so B1 is BCNF & 3NF. BranchName is not a key for B2, so we check another FD BranchStreetAddress, BranchCity -> BranchName
BranchStreetAddress,BranchCity+ = {BranchStreetAddress,BranchCity, BranchName, BranchProvince, BranchPostalCode, HoursOfOperation},

so this FD (and all the rest) is a key for B2 (BranchStreetAddress, BranchCity, BranchProvince, BranchPostalCode, BranchName)

Therefore our tables/relations are

B1(BranchName, HoursOfOperation)
CK: {BranchName}
PK: {BranchName}
B2(BranchStreetAddress, BranchCity, BranchProvince, BranchPostalCode, BranchName)
CK: {BranchStreetAddress, BranchCity}, {BranchStreetAddress, BranchPostalCode}
PK: {BranchStreetAddress, BranchCity}
FK: {BranchName} to B1

ServiceProvider (SIN: integer(9), SPName: VARCHAR(30), WorkSchedule: VARCHAR(50), DogOnly: BOOL, CatOnly: BOOL, DogAndCat: BOOL, BranchStreetAddress: VARCHAR(50), BranchCity: VARCHAR(30))

SIN -> SPName
SIN -> WorkSchedule
SIN -> BranchStreetAddress
SIN -> BranchCity
SIN -> DogOnly
SIN -> CatOnly
SIN -> DogAndCat
DogOnly, CatOnly -> DogAndCat

Let's first find all the minimal keys:

Left	Middle	Right
SIN	DogOnly, CatOnly, DogAndCat	SPName,WorkSchedule, BranchStreetAddress, BranchCity

SIN+ = {SIN, SPName, WorkSchedule, BranchStreetAddress, BranchCity, DogOnly, CatOnly, DogAndCat}

SIN is the minimal key.

DogOnly, CatOnly -> DogAndCat is not in 3NF therefore we can decompose. Let's start by taking the minimal cover:

1. Put FDs in standard form: Already done.
2. Minimize LHS of FDs: None can be minimized.

3. Remove redundant FDs: Since SIN -> CatOnly, SIN -> DogOnly, and SIN -> DogAndCat; DogOnly, CatOnly -> DogAndCat is redundant and can be removed.

Our minimal cover for ServiceProvider is ServiceProvider’:

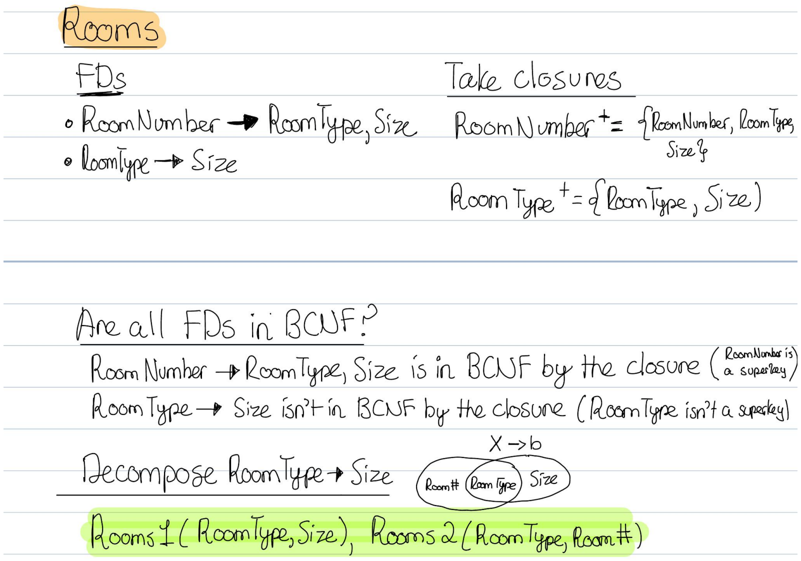
SIN -> SPName
SIN -> WorkSchedule
SIN -> BranchStreetAddress
SIN -> BranchCity
SIN -> DogOnly
SIN -> CatOnly
SIN -> DogAndCat

The minimal cover is now in BCNF and therefore in 3NF, so no need to decompose further. Our table/relation is:

ServiceProvider (<u>SIN</u>, SPName, WorkSchedule, DogOnly, CatOnly, DogAndCat, <u>BranchStreetAddress</u>, <u>BranchCity</u>)

CK: SIN
PK: SIN
FK: {BranchStreetAddress, BranchCity} to Branch

For Rooms:



Rooms

FDs

- RoomNumber → RoomType, Size
- RoomType → Size

Take closures

RoomNumber$^+$ = {RoomNumber, RoomType, Size}

RoomType$^+$ = {RoomType, Size)

Are all FDs in BCNF?

RoomNumber → RoomType, Size is in BCNF by the closure (RoomNumber is a superkey)

RoomType → Size isn't in BCNF by the closure (RoomType isn't a superkey)

Decompose RoomType → Size

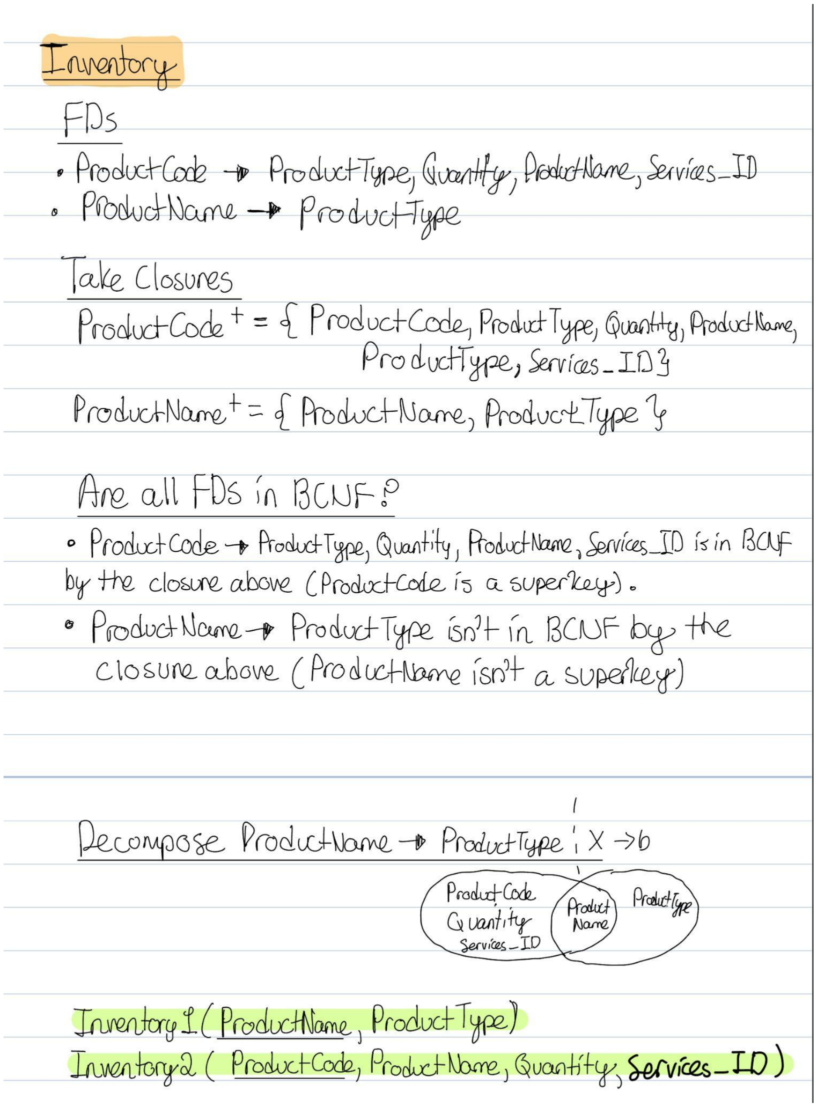
X -> b
Room#
Room Type
Size

Rooms1 (RoomType, Size), Rooms2 (RoomType, Room#)

Final list of relations for Rooms:

- Rooms1(RoomType: VARCHAR(30), Size: ENUM(‘small’, ‘medium’, ‘large’))
  PK={RoomType}
- Rooms2(RoomType: VARCHAR(30), RoomNumber: INT)
  PK={RoomNumber}
  FK={RoomType} references Rooms1

For Inventory:



Inventory

FDs

- ProductCode → ProductType, Quantity, ProductName, Services_ID
- ProductName → ProductType

Take Closures

ProductCode$^+$ = { ProductCode, ProductType, Quantity, ProductName,
ProductType, Services_ID }

ProductName$^+$ = { ProductName, ProductType }

Are all FDs in BCNF?

- ProductCode → ProductType, Quantity, ProductName, Services_ID is in BCNF by the closure above (ProductCode is a superkey).
- ProductName → ProductType isn't in BCNF by the closure above (ProductName isn't a superkey)

Decompose ProductName → ProductType ; X → b

Inventory1(ProductName, ProductType)
Inventory2 (ProductCode, ProductName, Quantity, Services_ID)

Final list of relations for Inventory:

- Services_Inventory1(ProductName: VARCHAR(30), ProductType: VARCHAR(20))
  PK = {ProductName}
- Services_Inventory2(ProductCode: INT, Quantity: ENUM(‘empty’, ‘low’, ‘half’, ‘full’), ProductName: VARCHAR(30), ServicesID: INT)
  PK = {ProductCode}
  FKs = {ServicesID} references Services,
  {ProductName} references Services_Inventory1

For Services:

- There is no need to do normalization since for both of its FDs, the LHS are superkeys since ServicesID, the PK of the relation, is on the LHS of each FDs, making them already in BCNF. Thus, Services will remain the same.

For all relations that are specializations of Services (i.e. Haircut, NailServices, TeethCleaning, ShampooAndEarCleaning)

- There is no need to do normalization since all of the FDs for these relations have the LHS with ServicesID as a superkey. ServicesID is a primary key for each specialization of Services, making them all BCNF. Thus, the relations will remain the same.

Final list of all relations after normalization:

PetOwner(POName: VARCHAR(30), POAddress: VARCHAR(50), POPhoneNumber: CHAR(10))
PK = {POName, POPhoneNumber}

OwnsPet(AID: INT, PName: VARCHAR(30), PSize: ENUM(‘small’, ‘medium’, ‘large’), PAge: INT, PType: ENUM(‘cat’, ‘dog’), POName: VARCHAR(30), POPhoneNumber: CHAR(10))
PK = {AID}
FK = {POName, POPhoneNumber} to PetOwner

ReservationHas(RID: INT, ServiceDate: DATE, ServiceTime: TIME, DateTimeBooked: DATETIME, AID: INT, POPhoneNumber: CHAR(10), POName: VARCHAR(30), ServicesID: INT, SIN: INT)

PK = {RID}

FK = {AID} to OwnsPet, {POName, POPhoneNumber) to PetOwner, {ServicesID} to Services, {SIN} to ServiceProvider

Branch1(BranchName: VARCHAR(50), HoursOfOperation)

CK = {BranchName}

PK = {BranchName}

Branch2(BranchStreetAddress: VARCHAR(50), BranchCity: VARCHAR(30), BranchProvince: VARCHAR(30), BranchPostalCode: CHAR(7), BranchName: VARCHAR(50))

CK = {BranchStreetAddress, BranchCity}, {BranchStreetAddress, BranchPostalCode}

PK = {BranchStreetAddress, BranchCity}

FK = {BranchName} to Branch1

ServiceProvider (SIN: INT, SPName: VARCHAR(30), WorkSchedule: VARCHAR(50), DogOnly: BOOL, CatOnly: BOOL, DogAndCat: BOOL, BranchStreetAddress: VARCHAR(50), BranchCity: VARCHAR(30))

CK = {SIN}

PK = {SIN}

FK = {BranchStreetAddress, BranchCity} to Branch2

Rooms1(RoomType: VARCHAR(30), Size: ENUM(‘small’, ‘medium’, ‘large’)))

PK = {RoomType}

Rooms2(RoomType: VARCHAR(30), RoomNumber: INT)

PK = {RoomNumber}

FK = {RoomType} references Rooms1

Services_Inventory1(ProductName: VARCHAR(30), ProductType: VARCHAR(20))

PK = {ProductName}

Services_Inventory2(ProductCode: INT, Quantity: ENUM(‘empty’, ‘low’, ‘half’, ‘full’), ProductName: VARCHAR(30), ServicesID: INT)

PK = {ProductCode}
FKs = {ServicesID} references Services,
{ProductName} references Services_Inventory1

Services(ServicesID: INT, ServicesName: VARCHAR(30), Duration: INT, BaseAmount: FLOAT, FinalRate: FLOAT)

PK = {ServicesID}

Haircut(ServicesID: INT, HaircutFurType: ENUM(‘soft’, ‘rugged’, ‘hairless’), FurLength: FLOAT)

PK={ServicesID}
FK={ServicesID} references Services

NailServices(ServicesID: INT, Sensitivity: ENUM(‘low’, ‘medium’, ‘high’)

PK={ServicesID}
FK={ServicesID} references Services

TeethCleaning(ServicesID: INT, DentalCondition: ENUM(‘excellent’, ‘good’, ‘poor’, ‘very poor’))

PK={ServicesID}
FK={ServicesID} references Services

ShampooAndEarCleaning(ServicesID: INT, Cleanliness: ENUM(‘excellent’, ‘good’, ‘poor’, ‘very poor’), SECFurType: ENUM(‘soft’, ‘rugged’, ‘hairless’))

PK={ServicesID}
FK={ServicesID} references Services

6. 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 PetOwner (
        POName              VARCHAR(30),
        POAddress           VARCHAR(50),
        POPhoneNumber       CHAR(10),
        PRIMARY KEY (POPhoneNumber, POName))

CREATE TABLE OwnsPet (
        AID                 INT PRIMARY KEY,
        PName               VARCHAR(30),
        PAge                INT,
        PSize               ENUM(‘small’, ‘medium’, ‘large’),
        PType               ENUM(‘cat’, ‘dog’),
        POName              VARCHAR(30),
        POPhoneNumber       CHAR(10),
        FOREIGN KEY (POName, POPhoneNumber) REFERENCES PetOwner(POName,
POPhoneNumber))

CREATE TABLE ReservationHas (
        RID                 INT PRIMARY KEY,
        ServiceDate         DATE,
        ServiceTime         TIME,
        DateTimeBooked      DATETIME,
        AID                 INT,
        POPhoneNumber       CHAR(10),
        POName              VARCHAR(30),
        ServicesID          INT,
        SIN                 INT,
        FOREIGN KEY (AID) REFERENCES OwnsPet(AID)
                ON DELETE CASCADE,
        FOREIGN KEY (POName, POPhoneNumber) REFERENCES PetOwner(POName,
POPhoneNumber)
                ON DELETE CASCADE,
        FOREIGN KEY (ServicesID) REFERENCES Services(ServicesID),
        FOREIGN KEY (SIN) REFERENCES ServiceProvider(SIN))

CREATE TABLE Branch1 (
        BranchName          VARCHAR(50) PRIMARY KEY,
        HoursOfOperation        VARCHAR(50))

CREATE TABLE Branch2 (
        BranchStreetAddress     VARCHAR(50),
        BranchCity              VARCHAR(30),
        BranchProvince          VARCHAR(30),
        BranchPostalCode        CHAR(7),
        BranchName              VARCHAR(50),
        PRIMARY KEY (BranchStreetAddress, BranchCity),
        FOREIGN KEY (BranchName) REFERENCES Branch1(BranchName)
                ON DELETE CASCADE
                ON UPDATE CASCADE)

CREATE TABLE ServiceProvider (
        SIN                     INT PRIMARY KEY,
        SPName                  VARCHAR(30),
        WorkSchedule            VARCHAR(50),
        DogOnly                 BOOL,
        CatOnly                 BOOL,
        DogAndCat               BOOL,
        BranchStreetAddress     VARCHAR(50),
        BranchCity              VARCHAR(30),
        FOREIGN KEY (BranchStreetAddress, BranchCity) REFERENCES
Branch2(BranchStreetAddress, BranchCity))

CREATE TABLE Room1(
        RoomType                VARCHAR(30) PRIMARY KEY,
        Size                    ENUM(‘small’, ‘medium’, ‘large’))

CREATE TABLE Room2(
        RoomNumber              INT PRIMARY KEY,
        RoomType                VARCHAR(30)),
        FOREIGN KEY (RoomType) REFERENCES Room1(RoomType)
                ON DELETE CASCADE
                ON UPDATE CASCADE)

CREATE TABLE RoomsHasServices(
        RoomNumber              INT,
        ServicesID              INT,
        PRIMARY KEY (RoomNumber, ServicesID),
        FOREIGN KEY (RoomNumber) REFERENCES Room2(RoomNumber),
        FOREIGN KEY (ServicesID) REFERENCES Services2(ServicesID))

CREATE TABLE Services_Inventory1(
        ProductName         VARCHAR(30) PRIMARY KEY,
        ProductType         VARCHAR(30))

CREATE TABLE Services_Inventory2(
        ProductCode         INT,
        Quantity            ENUM(‘empty’, ‘low’, ‘half’, ‘full’),
        ProductName         VARCHAR(30),
        ServicesID          INT,
        PRIMARY KEY (ProductCode),
        FOREIGN KEY (ProductName) REFERENCES Services_Inventory1(ProductName)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (ServicesID) REFERENCES Services(ServicesID)
                ON DELETE SET NULL
                ON UPDATE CASCADE)

CREATE TABLE Services(
        ServicesID          INT PRIMARY KEY,
        ServicesName        VARCHAR(30),
        BaseAmount          FLOAT,
        FinalRate           FLOAT,
        Duration            INT)

CREATE TABLE Haircut(
        ServicesID          INT PRIMARY KEY,
        HaircutFurType      ENUM(‘soft’, ‘rugged’, ‘hairless’),
        FurLength           FLOAT,
        FOREIGN KEY (ServicesID) REFERENCES Services(ServicesID)
                ON DELETE CASCADE
                ON UPDATE CASCADE)

CREATE TABLE NailServices(
        ServicesID          INT PRIMARY KEY,
        Sensitivity         ENUM(‘low’, ‘medium’, ‘high’),
        FOREIGN KEY (ServicesID) REFERENCES Services(ServicesID)
                ON DELETE CASCADE
```

```
                ON UPDATE CASCADE)

CREATE TABLE TeethCleaning(
        ServicesID              INT PRIMARY KEY,
        DentalCondition         ENUM(‘excellent’, ‘good’, ‘poor’, ‘very poor’),
        FOREIGN KEY (ServicesID) REFERENCES Services(ServicesID)
                ON DELETE CASCADE
                ON UPDATE CASCADE)

CREATE TABLE ShampooAndEarCleaning(
        ServicesID              INT PRIMARY KEY,
        Cleanliness             ENUM(‘excellent’, ‘good’, ‘poor’, ‘very poor’),
        SECFurType              ENUM(‘soft’, ‘rugged’, ‘hairless’),
        FOREIGN KEY (ServicesID) REFERENCES Services(ServicesID)
                ON DELETE CASCADE
                ON UPDATE CASCADE)
```

7. 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.

PetOwner

```
1: INSERT INTO PetOwner
VALUES ('Justin Bieber', ‘2424 Main Mall Vancouver BC‘, '1111111111');
2: INSERT INTO PetOwner
VALUES ('Taylor Swift', ‘4600 Cambie St Vancouver BC’, '2222222222');
3: INSERT INTO PetOwner
VALUES ('Soulja Boy', ‘9 Avenue Southwest Calgary AB’, '6789998212');
4: INSERT INTO PetOwner
VALUES ('Doja Cat', ‘13450 104 Ave Surrey BC’, '3333333333');
5: INSERT INTO PetOwner
VALUES ('Mariah Carey', ‘6111 River Rd Richmond BC’, '4444444444');
```

OwnsPet

```
1:INSERT INTO OwnsPet
VALUES(1, ‘Buttercup’, 1, ‘small’, ‘dog’, ‘Justin Bieber’, ‘1111111111’)
2:INSERT INTO OwnsPet
```

VALUES(2, ‘Olivia Benson’, 2, ‘small’, ‘cat’, ‘Taylor Swift’, ‘2222222222’)
3:INSERT INTO OwnsPet
VALUES(3, ‘Swag’, 5, ‘large’, ‘dog’, ‘Soulja Boy’, ‘6789998212’)
4:INSERT INTO OwnsPet
VALUES(4, ‘Kitty’, 1, ‘large’, ‘cat’, ‘Doja Cat’, ‘3333333333’)
5:INSERT INTO OwnsPet
VALUES(5, ‘Chantelle’, 3, ‘medium’, ‘dog’, ‘Mariah Carey’, ‘4444444444’)

ReservationHas
1: INSERT INTO ReservationHas
VALUES(1, ‘2023-10-31’, ‘12:00:00’, ‘2023-10-19 18:05:06’’, 1, ‘1111111111’, 1, 1)
2: INSERT INTO ReservationHas
VALUES(2, ‘2023-10-31’, ‘16:30:00’, ‘2023-10-20 09:46:57’’, 2, ‘2222222222’, 2, 2)
3: INSERT INTO ReservationHas
VALUES(3, ‘2023-11-11’, ‘11:00:00’, ‘2023-10-25 6:39:54’’, 3, ‘6789998212’, 3, 3)
4: INSERT INTO ReservationHas
VALUES(4, ‘2023-11-12’, ‘08:15:00’, ‘2023-11-01 12:00:01’’, 4, ‘3333333333’, 4, , 4)
5: INSERT INTO ReservationHas
VALUES(5, ‘2024-01-16’, ‘15:45:00’, ‘2023-12-25 18:05:06’’, 5, ‘4444444444’, 5, 5)

Branch
1:
INSERT INTO Branch1
VALUES (‘Vancouver Kitsilano’, ‘MTWTF 9-6’)
INSERT INTO Branch2
VALUES (‘2000 McDonald St’, ‘Vancouver’, ‘BC’, ‘V6K 3Y2’, ‘Vancouver Kitsilano’)
2:

```
INSERT INTO Branch1
VALUES (‘Richmond Aberdeen’, ‘TWTFSS 8-6’)
INSERT INTO Branch2
VALUES (‘4151 Hazelbridge Way’, ‘Richmond’, ‘BC’, ‘V6X 4J7’, ‘Richmond Aberdeen’)
```

3:

```
INSERT INTO Branch1
VALUES (‘Surrey Central’, ‘MTWTF 9-7’)
INSERT INTO Branch2
VALUES (‘10153 King George Blvd’, ‘Surrey’, ‘BC’, ‘V3T 2W1’, ‘Surrey Central’)
```

4:

```
INSERT INTO Branch1
VALUES (‘Downtown Calgary’, ‘SMTWTF 9-5’)
INSERT INTO Branch2
VALUES (‘328 Centre St S’, ‘Calgary’, ‘AB’, ‘T2G 4X6’, ‘Downtown Calgary’)
```

5:

```
INSERT INTO Branch1
VALUES (‘Country Hills Calgary’, ‘MTWTF 9-5’)
INSERT INTO Branch2
VALUES (‘1510 Country Hills Blvd NE’, ‘Calgary’, ‘AB’, ‘T3K 5Y7’, ‘Country Hills Calgary’)
```

Service Provider

1:

```
INSERT INTO ServiceProvider
VALUES (921343111, ‘Gertrude’, ‘MTTF 9-5’, FALSE, FALSE, TRUE, ‘1510 Country Hills Blvd NE’, ‘Calgary’)
```

2:

```
INSERT INTO ServiceProvider
VALUES (123456789, ‘Jerry’, ‘TT 9-5’, TRUE, FALSE, FALSE, ‘328 Centre St S’, ‘Calgary’)
```

3:

```
INSERT INTO ServiceProvider
```

VALUES (987654321, ‘Tom’, ‘MTWTF 9-5’, FALSE, FALSE, TRUE, ‘2000 McDonald St’, ‘Vancouver’)

4:

INSERT INTO ServiceProvider

VALUES (882671423, ‘Boris’, ‘MWF 9-5’, FALSE, TRUE, FALSE, ‘328 Centre St S’, ‘Calgary’)

5:

INSERT INTO ServiceProvider

VALUES (111222333, ‘Alice’, ‘MTW 9-4’, FALSE, FALSE, TRUE, ‘4151 Hazelbridge Way’, ‘Richmond’)

Room

1:

INSERT INTO Room1 VALUES (‘shampoo room’, ‘small’)

INSERT INTO Room2 VALUES (‘101’, ‘shampoo room’)

2:

INSERT INTO Room1 VALUES (‘shampoo room’, ‘small’)

INSERT INTO Room2 VALUES (‘102’, ‘shampoo room’)

3:

INSERT INTO Room1 VALUES (‘shampoo room’, ‘large’)

INSERT INTO Room2 VALUES (‘103’, ‘shampoo room’)

4:

INSERT INTO Room1 VALUES (‘shampoo room’, ‘large’)

INSERT INTO Room2 VALUES (‘104’, ‘shampoo room’)

5:

INSERT INTO Room1 VALUES (‘shampoo room’, ‘medium’)

INSERT INTO Room2 VALUES (‘105’, ‘shampoo room’)

RoomsHasServices

1: INSERT INTO RoomsHasServices VALUES (101, 1)

2: INSERT INTO RoomsHasServices VALUES (102, 2)

3: INSERT INTO RoomsHasServices VALUES (103, 3)

4: INSERT INTO RoomsHasServices VALUES (104, 4)

5: INSERT INTO RoomsHasServices VALUES (105, 5)

ServicesInventory

```
1:
INSERT INTO Services_Inventory1 VALUES(‘Flea B Gone’, ‘hairspray’)
INSERT INTO Services_Inventory2 VALUES(1, ‘low’, ‘Flea B Gone’, 1)
2:
INSERT INTO Services_Inventory1 VALUES(‘2-in-1 for furry friends’, ‘shampoo’)
INSERT INTO Services_Inventory2 VALUES(2, ‘half’, ‘2-in-1 for furry friends’, 2)
3:
INSERT INTO Services_Inventory1 VALUES(‘3-in-1 for furry friends’, ‘mousse’)
INSERT INTO Services_Inventory2 VALUES(3, ‘full’, ‘3-in-1 for furry friends’, 3)
4:
INSERT INTO Services_Inventory1 VALUES(‘Shiny coat lover’, ‘shampoo’)
INSERT INTO Services_Inventory2 VALUES(4, ‘empty’, ‘Shiny Coat Lover’, 4)
5:
INSERT INTO Services_Inventory1 VALUES(‘Snazzy Shine’, ‘conditioner’)
INSERT INTO Services_Inventory2 VALUES(5, ‘low’, ‘Snazzy Shine’, 5)
```

Services

```
1: INSERT INTO Services VALUES(1, ‘Everything’, 50.00, 60.00, 40)
2: INSERT INTO Services VALUES(2, ‘Everything’, 50.00, 70.00, 80)
3: INSERT INTO Services VALUES(3, ‘Everything’, 50.00, 70.00, 80)
4: INSERT INTO Services VALUES(4, ‘Everything’, 50.00, 75.00, 100)
5: INSERT INTO Services VALUES(5, ‘Everything’, 50.00, 65.00, 60)
```

Haircut

```
1: INSERT INTO Haircut VALUES(1, ‘soft’, 1.2)
2: INSERT INTO Haircut VALUES(2, ‘soft’, 1)
3: INSERT INTO Haircut VALUES(3, ‘rugged’, 3)
4: INSERT INTO Haircut VALUES(4, ‘rugged’, 0.8)
5: INSERT INTO Haircut VALUES(5, ‘hairless’, 0)
```

NailServices

```
1: INSERT INTO NailServices VALUES(1, ‘low’)
2: INSERT INTO NailServices VALUES(2, ‘high’)
3: INSERT INTO NailServices VALUES(3, ‘med’)
4: INSERT INTO NailServices VALUES(4, ‘low’)
5: INSERT INTO NailServices VALUES(5, ‘high’)
```

TeethCleaning

1: INSERT INTO TeethCleaning VALUES(1, ‘good’)
2: INSERT INTO TeethCleaning VALUES(2, ‘excellent’)
3: INSERT INTO TeethCleaning VALUES(3, ‘poor’)
4: INSERT INTO TeethCleaning VALUES(4, ‘very poor’)
5: INSERT INTO TeethCleaning VALUES(5, ‘poor’)

ShampooAndEarCleaning

1: INSERT INTO ShampooAndEarCleaning VALUES(1, ‘excellent’)
2: INSERT INTO ShampooAndEarCleaning VALUES(2, ‘good’)
3: INSERT INTO ShampooAndEarCleaning VALUES(3, ‘good’)
4: INSERT INTO ShampooAndEarCleaning VALUES(4, ‘poor’)
5: INSERT INTO ShampooAndEarCleaning VALUES(5, ‘poor’)