University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: __2______

Date: ___Oct 15_________

Group Number: ____96______

Name	**Student Number**	**CS Alias (Userid)**	**Preferred E-mail Address**
Chloe Van	32383119	r5r4l	chloe.m.van@gmail.com
Kai Groden-Gilchrist	35600148	u9h1b	kaigg@live.ca
Nariman Muldashev	25548158	n1b3b	muldashev11@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

CPSC 304 M2 - Group 96

2. A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.

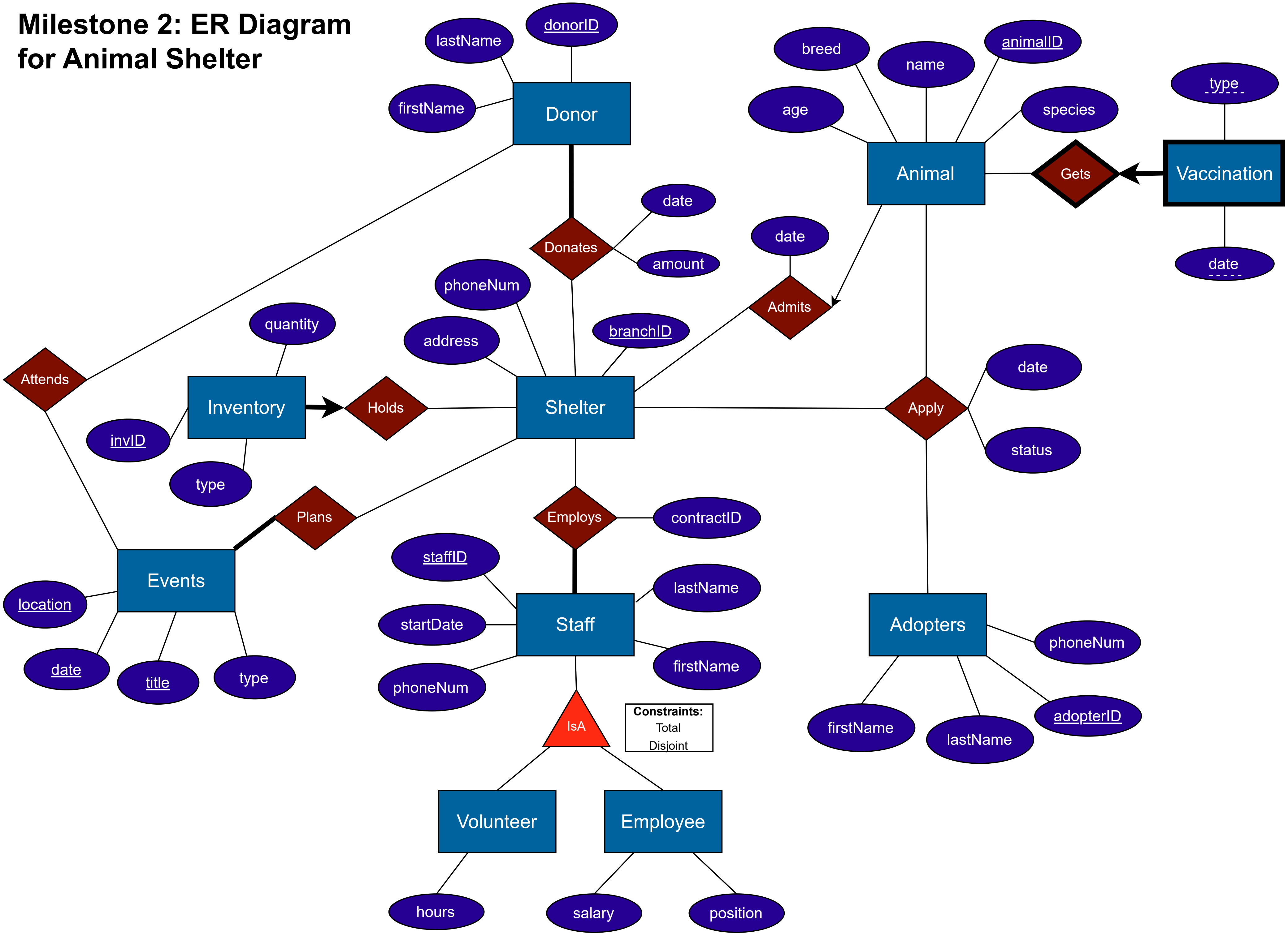
For this project, our team is designing an application for an animal shelter. The domain of the application are daily operations of an animal shelter including adoptions, inventory management, and keeping track of staff info.

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.
If you have decided not to implement the suggestions given by your project mentor, please be sure to leave a note stating why. This is not to say that you must do everything that your project mentor says. In many instances, there are trade-offs between design choices and your decision may be influenced by different factors. That being said, your TAs will often leave suggestions that are meant to help massage your project into a form that will fit with the requirements in future project milestones. If you choose not to take their advice, it would be helpful for them to know why in order to better assist the group moving forward.

ER diagram is displayed on the next page.

Changes since M1 - all TA suggestions incorporated:
- Added constraints defining the ISA relationship
- Added participation constraint from Inventory to Holds
- Added the attends relationship linking donors to events giving us 7 total relationships

Milestone 2: ER Diagram for Animal Shelter



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.

Entities:

- Donor(donorID: varchar[8], lastName: varchar[200], firstName: varchar[200])
- InventoryHolds(invID: varchar[8], **branchID**: varchar[8], quantity: integer, type: varchar[200])
    - Combined the Inventory entity and Holds relationship into one schema given that it is a many-to-one relation
- Events(location: varchar[200], date: date, title: varchar[200], type: varchar[200])
- Shelter(branchID: varchar[8], phoneNum: varchar[10], address: varchar[200])
- Staff(staffID:varchar[8],lastName: varchar[200], firstName: varchar[200], startDate: date, phoneNum: varchar[10])
- Volunteer(**staffID**: varchar[8], hours: integer)
- Employee(**staffID**: varchar[8], salary: integer, position: varchar[200])
- Vaccination(type: varchar[200], date: date, **animalID:** varchar[8])
- AnimalAdmits(animalID: varchar[8], name: varchar[200], species: varchar[200], breed: varchar[200], age: integer, **branchID**: varchar[8], date: date)
    - Combined the Animal entity and Admits relationship into one schema given that it is a many-to-one relation
- Adopters(adopterID: varchar[8], lastName: varchar[200], firstName: varchar[200], phoneNum: varchar[10])

Relationships:

- Attends(**donorID:** varchar[8], **location:** varchar[200], **date**: date, **title:** varchar[200])
- Donates(**donorID:** varchar[8], **branchID**: varchar[8], date: date, amount: float)
- Plans(**location:** varchar[200], **date**: date, **title:** varchar[200], **branchID**: varchar[8])
- Employs(**staffID**: varchar[8], **branchID**: varchar[8], contractID: varchar[8])
- Apply(**branchID**: varchar[8], **adopterID**: varchar[8], **animalID**: varchar[8], status: varchar[200], date: date)

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.

- Donor
    - Primary Key: donorID
    - Candidate Key: NA
    - Foreign Key: NA
    - Not null: NA
    - Unique: NA
    - Other: NA
- InventoryHolds
    - Primary Key: invID
    - Candidate Key: branchID, type
    - Foreign Key: branchID
    - Not null: branchID, type, quantity
    - Unique: (branchID, type)
    - Other: quantity >= 0; use ON DELETE CASCADE on branchID; use ON UPDATE CASCADE on branchID
- Events
    - Primary Key: location, date, title
    - Candidate Key: NA
    - Foreign Key: NA
    - Not null: NA
    - Unique: NA
    - Other: NA
- Shelter
    - Primary Key: branchID
    - Candidate Key: phoneNumber, address
    - Foreign Key: NA
    - Not null: phoneNumber, address
    - Unique: (phoneNumber, address)
    - Other: NA

- Staff
    - Primary Key: staffID
    - Candidate Key: NA
    - Foreign Key: NA
    - Not null: startDate, firstName, lastName, phoneNum
    - Unique: NA
    - Other: NA
- Volunteer
    - Primary Key: staffID
    - Candidate Key: NA
    - Foreign Key: staffID
    - Not null: hours
    - Unique: NA
    - Other: hours >= 0; use ON DELETE CASCADE on staffID; use ON UPDATE CASCADE on staffID
- Employee
    - Primary Key: staffID
    - Candidate Key: NA
    - Foreign Key: staffID
    - Not null: position, salary
    - Unique: NA
    - Other: salary > 0; use ON DELETE CASCADE on staffID; use ON UPDATE CASCADE on staffID
- Vaccination
    - Primary Key: type, date, animalID
    - Candidate Key: NA
    - Foreign Key: animalID
    - Not null: animalID
    - Unique: NA
    - Other: Use ON DELETE CASCADE on animalID; Use ON UPDATE CASCADE on animalID

- AnimalAdmits
  - Primary Key: animalID
  - Candidate Key: NA
  - Foreign Key: branchID
  - Not null: branchID, date
  - Unique: NA
  - Other: Use ON DELETE CASCADE on branchID; Use ON UPDATE CASCADE on branchID;
- Adopters
  - Primary Key: adopterID
  - Candidate Key: NA
  - Foreign Key: NA
  - Not null: firstName, lastName, phoneNum
  - Unique: NA
  - Other: NA
- Attends
  - Primary Key: donorID, location, date, title
  - Candidate Key: NA
  - Foreign Key: donorID, location, date, title
  - Not null: NA
  - Unique: NA
  - Other: Use ON DELETE CASCADE on donorID, location, date, title; Use ON UPDATE CASCADE on donorID, location, date, title;
- Donates
  - Primary Key: donorID, branchID
  - Candidate Key: NA
  - Foreign Key: donorID, branchID
  - Not null: date, amount
  - Unique: NA
  - Other: amount>0; Use ON DELETE CASCADE on donorID, branchID; Use ON UPDATE CASCADE on donorID, branchID

- Plans
    - Primary Key: branchID, location, date, title
    - Candidate Key: NA
    - Foreign Key: branchID, location, date, title
    - Not null: NA
    - Unique: NA
    - Other: Use ON DELETE CASCADE on branchID, location, date, title; Use ON UPDATE CASCADE on branchID, location, date, title
- Employs
    - Primary Key: branchID, staffID
    - Candidate Key: contractID
    - Foreign Key: branchID, staffID
    - Not null: contractID
    - Unique: contractID
    - Other: Use ON DELETE CASCADE on branchID, staffID; Use ON UPDATE CASCADE on branchID, staffID
- Apply
    - Primary Key: branchID, adopterID, animalID
    - Candidate Key: NA
    - Foreign Key: branchID, adopterID, animalID
    - Not null: status, date
    - Unique: NA
    - Other: Use ON DELETE CASCADE on branchID, adopterID, animalID; Use ON UPDATE CASCADE on branchID, adopterID, animalID

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.
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.

- Donor(<u>donorID</u>, lastName, firstName)
  - donorID → lastName, firstName
- InventoryHolds(<u>invID</u>, **branchID**, quantity, type)
  - invID → quantity, type, branchID
  - branchID, type →invID, quantity
- Events(<u>location</u>, <u>date,</u> <u>title</u>, type)
  - location, date, title → type
- Shelter(<u>branchID</u>, phoneNum, address)
  - branchID → phoneNum, address
  - phoneNumber, address → branchID
- Staff(<u>staffID</u>, lastName, firstName, startDate, phoneNum)
  - staffID → lastName, firstName, startDate, phoneNum
- Volunteer(<u>**staffID**</u>, hours)
  - staffID → hours
- Employee(<u>**staffID**</u>, salary, position)
  - staffID → salary, position
  - position → salary
- Vaccination(<u>type</u>, <u>date</u>, <u>**animalID**</u>)
  - NA
- AnimalAdmits(<u>animalID,</u> name, species, breed, age, **branchID**, date)
  - animalID → name, species, breed, age, date, branchID
  - breed -> species
- Adopters(<u>adopterID</u>, lastName, firstName, phoneNum)
  - adopterID → lastName, firstName, phoneNum
- Attends(<u>**donorID**</u>, <u>**location**</u>, <u>**date**</u>, <u>**title**</u>)
  - NA

- Donates (**<u>donorID</u>, <u>branchID</u>**, date, amount)
    - donorID, branchID → date, amount
- Plans(**<u>location</u>, <u>date</u>, <u>title</u>, <u>branchID</u>**)
    - NA
- Employs(**<u>staffID</u>**, **<u>branchID</u>**, contractID)
    - **<u>staffID</u>**, **<u>branchID</u>** → contractID
    - contractID → **<u>staffID</u>**, **<u>branchID</u>**
- Apply(date, **<u>branchID</u>**, **<u>adopterID</u>**, **<u>animalID</u>**, status)
    - **<u>branchID</u>**, **<u>adopterID</u>**, **<u>animalID</u>** → status, date

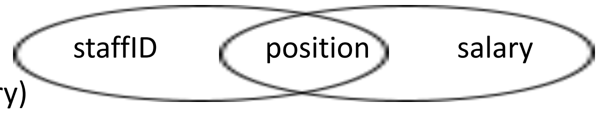
6. Normalization
a. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization. You should show the steps taken for the decomposition. Should there be errors, and no work is shown, no partial credit 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.

- Donor(<u>donorID:</u> varchar[8], lastName: varchar[200], firstName: varchar[200])
    - Candidate Key: NA
    - FDs:
        - donorID → lastName
        - donorID → firstName
    - Closures:
        - donorID+ = {donorID, lastName, firstName}
    - donorID is a superkey so this relationship is in BCNF.
- InventoryHolds(<u>invID:</u> varchar[8], **branchID**: varchar[8], quantity: integer, type: varchar[200])
    - Candidate Key: branchID, type
    - FDs:
        - invID → quantity
        - invID → type
        - invID → branchID
        - branchID, type →invID, quantity

    - Closures:
        - invID+ = {invID, quantity, type, branchID}
        - (branchID, type)+ = {branchID, type, invID, quantity}
    - invID, and (branchID, type) are superkeys so this relationship is in BCNF.
- Events(<u>location:</u> varchar[200], <u>date:</u> date, <u>title:</u> varchar[200], type: varchar[200])
    - Candidate Key: NA
    - FDs:
        - location, date, title → type
    - Closures:
        - (location, date, title)+ = {location, date, title, type}
    - (location, date, title) is a superkey so this relationship is in BCNF.
- Shelter(<u>branchID:</u> varchar[8], phoneNum: varchar[10], address: varchar[200])
    - Candidate Key: phoneNumber, address
    - FDs:
        - branchID → phoneNum
        - branchID → address
        - phoneNumber, address → branchID
    - Closures:
        - branchID+ = {branchID, phoneNumber, address}
        - (phoneNumber, address)+ = {phoneNumber, address, branchID}
    - branchID, and (phoneNumber, address) are superkeys so this relationship is in BCNF.
- Staff(<u>staffID:</u>varchar[8],lastName: varchar[200], firstName: varchar[200], startDate: date, phoneNum: varchar[10])
    - Candidate Key: NA
    - FDs:
        - staffID → lastName
        - staffID → firstName
        - staffID → startDate
        - staffID →phoneNum
    - Closures:
        - staffID+ = {staffID, lastName, firstName, startDate, phoneNum)
    - staffID is a superkey so this relationship is in BCNF.
- Volunteer(<u>**staffID**:</u> varchar[8], hours: integer)
    - Candidate Key: NA
    - FDs:
        - staffID → hours

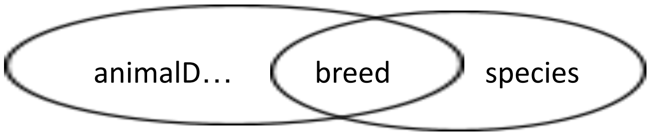
- Closures:
  - staffID+ = {staffID, hours}
- staffID is a superkey so this relationship is in BCNF

- Employee(**<u>staffID</u>**<u>:</u> varchar[8], salary: integer, position: varchar[200])
  - Candidate Key: NA
  - FDs:
    - staffID → salary
    - staffID → position
    - position → salary
  - Closures:
    - staffID+ = {staffID, salary, position}
    - position+ = {position, salary}
  - position is not a superkey and so position → salary violates BCNF. It is enough that just on FD violates BCNF for this relation Employee(**<u>staffID</u>**<u>:</u> varchar[8], salary: integer, position: varchar[200]) to not be in BCNF.
  - Decompose into BCNF:
  - position → salary



    - R1(position,salary)
      - A two attribute relation does not violate BCNF so we can keep R1
      - Rename R1 to SalaryRanges
    - R2(staffID, position)
      - A two attribute relation does not violate BCNF so we can keep R2
      - Rename R2 to Employee
  - Final answer:
    - SalaryRanges(<u>position</u>: varchar[200], salary: integer)
      - Candidate Key: NA
    - Employee(**<u>staffID</u>**<u>:</u> varchar[8], **position**: varchar[200])
      - Candidate Key: NA
- Vaccination(<u>type:</u> varchar[200], <u>date:</u> date, **<u>animalID:</u>** varchar[8])
  - Candidate Key: NA
  - There are no non-trivial FDs, so this relationship is in BCNF.
- AnimalAdmits(<u>animalID:</u> varchar[8], name: varchar[200], species: varchar[200], breed: varchar[200], age: integer, **branchID**: varchar[8], date: date)
  - Candidate Key: NA

- FDs:
  - animalID ⟶ name
  - animalID ⟶ species
  - animalID ⟶ breed
  - animalID ⟶ age
  - animalID ⟶ date
  - animalID ⟶ branchID
  - breed ⟶ species
- Closures:
  - animalID+ = {animalID, name, species, breed, age, date, branchID}
  - breed+ = {breed, species}
- breed is not a superkey and so breed ⟶ species violates BCNF. It is enough that just one FD violates BCNF for this relation AnimalAdmits(<u>animalID:</u> varchar[8], name: varchar[200], species: varchar[200], breed: varchar[200], age: integer, **branchID**: varchar[8], date: date) to not be in BCNF.
- Decompose into BCNF:
- breed ⟶ species



  - R1(breed,species)
    - A two attribute relation does not violate BCNF so we can keep R1
    - Rename R1 to AnimalInfo
  - R2(breed, animalID, name, age, date, branchID)
    - animalID is a superkey, and there are no other non-trivial FDs, so this is in BCNF
    - Rename R2 to AnimalAdmits
- Final answer:
  - AnimalInfo(<u>breed</u>: varchar[200], species: varchar[200])
    - Candidate Key: NA
  - AnimalAdmits(<u>animalID:</u> varchar[8], **breed**: varchar[200], name: varchar[200], age: integer, **branchID**: varchar[8], date: date)
    - Candidate Key: NA

- Adopters(<u>adopterID:</u> varchar[8], lastName: varchar[200], firstName: varchar[200], phoneNum: varchar[10])
  - Candidate Key: NA
  - FDs:
    - adopterID ⟶ lastName
    - adopterID ⟶ firstName
    - adopterID ⟶ phoneNum
  - Closures:

        - adopterID+ = {adopterID, lastName, firstName, phoneNum}
    - adopterID is a superkey so this relationship is in BCNF.
- Attends(**<u>donorID:</u>** varchar[8], **<u>location:</u>** varchar[200], **<u>date</u>**<u>:</u> date, **<u>title:</u>** varchar[200])
    - Candidate Key: NA
    - There are no non-trivial FDs, so this relationship is in BCNF.
- Donates(**<u>donorID:</u>** varchar[8], **<u>branchID</u>**<u>:</u> varchar[8], date: date, amount: float)
    - Candidate Key: NA
    - FDs:
        - donorID, branchID ⟶ date
        - donorID, branchID ⟶ amount
    - Closures:
        - (donorID, branchID)+ = {donorID, branchID, date, amount}
    - (donorID, branchID) is a superkey so this relationship is in BCNF.
- Plans(**<u>location:</u>** varchar[200], **<u>date</u>**<u>:</u> date, **<u>title:</u>** varchar[200], **<u>branchID</u>**<u>:</u> varchar[8])
    - Candidate Key: NA
    - There are no non-trivial FDs, so this relationship is in BCNF.
- Employs(**<u>staffID</u>**<u>:</u> varchar[8], **<u>branchID</u>**<u>:</u> varchar[8], contractID: varchar[8])
    - Candidate Key: contractID
    - FDs:
        - staffID, branchID ⟶ contractID
        - contractID ⟶ staffID
        - contractID ⟶ branchID
    - Closures:
        - (staffID, branchID)+ = {staffID, branchID, contractID}
        - contractID+ = {contractID, staffID, branchID}
    - (staffID, branchID) and contractID are superkeys so this relationship is in BCNF.
- Apply(**<u>branchID</u>**: varchar[8], **<u>adopterID</u>**: varchar[8], **<u>animalID</u>**: varchar[8], status: varchar[200], date: date)
    - Candidate Key: NA
    - FDs:
        - branchID, adopterID, animalID ⟶ status
        - branchID, adopterID, animalID ⟶ date
    - Closures:
        - branchID, adopterID, animalID+ = {date, branchID, adopterID, animalID, status, date}
    - branchID, adopterID, animalID is a superkey so this relationship is in BCNF.

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 varcharacters for a given varcharacter, it is
better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four varcharacters 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 varcharacters in a course name can vary greatly.

-
  ```
  CREATE TABLE Donor(
          donorID             VARCHAR(8),
          lastName            VARCHAR(200),
          firstName           VARCHAR(200),
          PRIMARY KEY         (donorID)
          );
  ```

-
  ```
  CREATE TABLE InventoryHolds(
          invID               VARCHAR(8),
          branchID            VARCHAR(8)           NOT NULL,
          quantity            INTEGER              NOT NULL     CHECK (amount >= 0),
          type                VARCHAR(200)                 NOT NULL,
          PRIMARY KEY (invID),
          FOREIGN KEY  (branchID) REFERENCES Shelter(branchID)
                  ON DELETE CASCADE
                  ON UPDATE CASCADE
          );
  ```

-
  ```
  CREATE TABLE Events(
          location                    VARCHAR(200),
          date                        DATE,
          title                       VARCHAR(200),
          type                        VARCHAR(200),
          PRIMARY KEY (location, date, title)
          );
  ```

-
  ```
  CREATE TABLE Shelter(
          branchID            VARCHAR(8),
          phoneNum            CHAR(10)             NOT NULL,
          address             VARCHAR(200)                 NOT NULL,
          PRIMARY KEY         (branchID),
          UNIQUE              (phoneNum, address)
  ```

```
        );
●   CREATE TABLE Staff(
        staffID             VARCHAR(8),
        lastName            VARCHAR(200)                NOT NULL,
        firstName           VARCHAR(200)                NOT NULL,
        startDate           DATE                NOT NULL,
        phoneNum            CHAR(10)            NOT NULL,
        PRIMARY KEY         (staffID)
        );
●   CREATE TABLE Volunteer(
        staffID             VARCHAR(8),
        hours               INTEGER             CHECK (hours >= 0),
        PRIMARY KEY         (staffID),
        FOREIGN KEY (staffID) REFERENCES Staff(staffID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
        );
●   CREATE TABLE SalaryRanges(
        position            VARCHAR(200)
        salary              INTEGER             NOT NULL        CHECK (salary > 0),
        PRIMARY KEY         (position)
        );
●   CREATE TABLE Employee(
        staffID             VARCHAR(8),
        position            VARCHAR(200)                NOT NULL,
        PRIMARY KEY         (staffID),
        FOREIGN KEY (staffID) REFERENCES Staff(staffID)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREIGN KEY (position) REFERENCES SalaryRanges(position)
                ON DELETE CASCADE
                ON UPDATE CASCADE
        );
●   CREATE TABLE AnimalAdmits (
        animalID                    VARCHAR(8),
        name                        VARCHAR(200),
        breed                       VARCHAR(200)        NOT NULL,
        age                         INTEGER NOT NULL (CHECK (age >=0)),
    date                    DATE          NOT NULL,
    PRIMARY KEY (animalID),
    FOREIGN KEY (branchID) REFERENCES Shelter(branchID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (breed) REFERENCES AnimalInfo(animalID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
    );
```

** If breed is unknown, enter species as breed

-
  ```
  CREATE TABLE AnimalInfo (
      breed                   VARCHAR(8),
      species                 VARCHAR(200) NOT NULL,
      PRIMARY KEY (breed)
      );
  ```

-
  ```
  CREATE TABLE Adopters (
      adopterID                     VARCHAR(8),
      lastName                      VARCHAR(200) NOT NULL,
      firstName                     VARCHAR(200) NOT NULL,
      phoneNum                      VARCHAR(10)  NOT NULL,
      PRIMARY KEY (adopterID)
      );
  ```

-
  ```
  CREATE TABLE Vaccination (
      type                    VARCHAR(200),
      date                    DATE,
      animalID                VARCHAR(8),
      PRIMARY KEY(type,date,animalID),
      FOREIGN KEY (animalID) REFERENCES AnimalAdmits(animalID)
          ON DELETE CASCADE
          ON UPDATE CASCADE
      );
  ```

-
  ```
  CREATE TABLE Attends(
      donorID             VARCHAR(8),
      location            VARCHAR(200),
      date                DATE,
      title,              VARCHAR(200)
      PRIMARY KEY         (donorID, location, date, title),
      FOREIGN KEY         (donorID) REFERENCES
  ```

```
                    Donor(donorID)
                    ON DELETE CASCADE
                    ON UPDATE CASCADE,
        FOREIGN KEY (location, date, title) REFERENCES
                    Events(location,date,title)
                    ON DELETE CASCADE
                    ON UPDATE CASCADE
        );
```

- CREATE TABLE Donates(

```
        donorID         VARCHAR(8),
        branchID        VARCHAR(8),
        date            DATE            NOT NULL,
        amount          FLOAT           NOT NULL     CHECK (amount > 0),
        PRIMARY KEY     (donorID, branchID),
        FOREIGN KEY     (donorID) REFERENCES
                        Donor(donorID)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE,
        FOREIGN KEY     (branchID) REFERENCES
                        Shelter(branchID)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
        );
```

- CREATE TABLE Plans(

```
        location        VARCHAR(200),
        date            DATE,
        title           VARCHAR(200),
        branchID,       VARCHAR(8),
        PRIMARY KEY     (location,date,title,branchID),
        FOREIGN KEY     (location,date,title) REFERENCES
                        Events(location,date,title)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE,
         FOREIGN KEY    (branchID) REFERENCES
                        Shelter(branchID)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
        );
```

-
  ```
  CREATE TABLE Employs(
          staffID          VARCHAR(8),
          branchID         VARCHAR(8),
          contractID       VARCHAR(8)              NOT NULL,
          PRIMARY KEY      (branchID, staffID),
          FOREIGN KEY      (branchID) REFERENCES
                           Shelter(branchID)
                           ON DELETE CASCADE
                           ON UPDATE CASCADE,
          FOREIGN KEY      (staffID) REFERENCES
                           Staff(staffID)
                           ON DELETE CASCADE
                           ON UPDATE CASCADE
          );
  ```

-
  ```
  CREATE TABLE Apply (
          branchID         VARCHAR(8),
          adopterID        VARCHAR(8),
          animalID         VARCHAR(8),
          status           VARCHAR(200)            NOT NULL ,
          date             DATE            NOT NULL,
          PRIMARY KEY      (branchID, adopterID, animalID),
          FOREIGN KEY      (branchID) REFERENCES Shelter(branchID)
                           ON DELETE CASCADE
                           ON UPDATE CASCADE,
          FOREIGN KEY      (adopterID) REFERENCES Adopters(adopterID)
                           ON DELETE CASCADE
                           ON UPDATE CASCADE,
          FOREIGN KEY      (animalID) REFERENCES AnimalAdmits(animalID)
                           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.

- Donor(<u>donorID:</u> varchar[8], lastName: varchar[200], firstName: varchar[200])
    - INSERT INTO Donor(donorID, lastName, firstName) VALUES (“1”, “John”, “Smith”)

    - INSERT INTO Donor(donorID, lastName, firstName) VALUES (“2”, “James”, “Smith”)
    - INSERT INTO Donor(donorID, lastName, firstName) VALUES (“3”, “Anna”, “Lee”)
    - INSERT INTO Donor(donorID, lastName, firstName) VALUES (“4”, “James”, “Monroe”)
    - INSERT INTO Donor(donorID, lastName, firstName) VALUES (“5”, “John”, “Smith”)
- InventoryHolds(<u>invID:</u> varchar[8], **branchID**: varchar[8], quantity: integer, type: varchar[200])
    - INSERT INTO InventoryHolds(invID, branchID, quantity, type) VALUES(“11111111”, “12345678”, 12, “syringe”)
    - INSERT INTO InventoryHolds(invID, branchID, quantity, type) VALUES(“11111112”, “12345678”, 1, “comb”)
    - INSERT INTO InventoryHolds(invID, branchID, quantity, type) VALUES(“11111113”, “12345678”, 22, “collars”)
    - INSERT INTO InventoryHolds(invID, branchID, quantity, type) VALUES(“11111113”, “12345678”, 12, “flea medicine”)
    - INSERT INTO InventoryHolds(invID, branchID, quantity, type) VALUES(“11111113”, “12345678”, 3, “cat treats”)
- Events(<u>location:</u> varchar[200], <u>date:</u> date, <u>title:</u> varchar[200], type: varchar[200])
    - INSERT INTO Events (location,date,title,type) VALUES (“Stanley Park”, “2023-07-15”, “PetPalooza”, “Education”)
    - INSERT INTO Events (location,date,title,type) VALUES (“New Brighton Park”, “2021-03-04”, “Furry Friends Fair”, “Adoption”)
    - INSERT INTO Events (location,date,title,type) VALUES (“Olympic Oval”, “2022-05-10”, “Strut for Strays 5K”, “Fundraising”)
    - INSERT INTO Events (location,date,title,type) VALUES (“Vancouver Art Gallery”, “2029-09-15”, “Paws & Paint Night”, “Fundraising”)
    - INSERT INTO Events (location,date,title,type) VALUES (“Queen Elizabeth Park”, “2022-10-31”, “Barktoberfest”, “Fundraising”)
- Shelter(<u>branchID:</u> varchar[8], phoneNum: varchar[10], address: varchar[200])
    - INSERT INTO Shelter (“12345678”, “7781230033”, “123 west 10th avenue, Vancouver, BC V6E9TS”)
    - INSERT INTO Shelter (“12345679”, “7781230034”, “123 west 11th avenue, Vancouver, BC V6E9TS”)
    - INSERT INTO Shelter (“12345680”, “7781230035”, “123 west 12th avenue, Vancouver, BC V6E9TS”)

    - INSERT INTO Shelter (“12345681”, “7781230036”, “123 west 13th avenue, Vancouver, BC V6E9TS”)
    - INSERT INTO Shelter (“12345682”, “7781230037”, “123 west 14th avenue, Vancouver, BC V6E9TS”)
- Staff(<u>staffID:</u> varchar[8],lastName: varchar[200], firstName: varchar[200], startDate: date, phoneNum: varchar[10])
    - INSERT INTO Staff (staffID, lastName, firstName, startDate, phoneNumber) VALUES (“00000000”, “Johnson”, “Emily”, “2015-10-23”, “7783751826”)
    - INSERT INTO Staff (staffID, lastName, firstName, startDate, phoneNumber) VALUES (“00000001”, “Martinez”, “David”, “1999-04-20”, “2504338590”)
    - INSERT INTO Staff (staffID, lastName, firstName, startDate, phoneNumber) VALUES (“00000002”, “Anderson”, “Sarah”, “2009-06-13”, “6042973456”)
    - INSERT INTO Staff (staffID, lastName, firstName, startDate, phoneNumber) VALUES (“00000003”, “Wilson”, “Michael”, “2003-07-05”, “6041234567”)
    - INSERT INTO Staff (staffID, lastName, firstName, startDate, phoneNumber) VALUES (“00000004”, “Davis”, “Olivia”, “1996-09-01”, “6040001234”)
    - INSERT INTO Staff (staffID, lastName, firstName, startDate, phoneNumber) VALUES (“00000005”, “Taylor”, “Sophia”, “2008-01-19”, “6047893456”)
    - INSERT INTO Staff (staffID, lastName, firstName, startDate, phoneNumber) VALUES (“00000006”, “Brown”, “Ethan”, “2020-08-04”, “7784560192”)
    - INSERT INTO Staff (staffID, lastName, firstName, startDate, phoneNumber) VALUES (“00000007”, “Miller”, “Ava”, “2022-06-18”, “7782950146”)
    - INSERT INTO Staff (staffID, lastName, firstName, startDate, phoneNumber) VALUES (“00000008”, “Clark”, “James”, “2020-09-28”, “2507182934”)
    - INSERT INTO Staff (staffID, lastName, firstName, startDate, phoneNumber) VALUES (“00000009”, “Hernandez”, “Mia”, “2021-12-01”, “6046782048”)
- Volunteer(<u>**staffID**:</u> varchar[8], hours: integer)
    - INSERT INTO Volunteer(staffID, hours)
      VALUES(“00000000”, 10)
    - INSERT INTO Volunteer(staffID, hours)
      VALUES(“00000001”, 5)
    - INSERT INTO Volunteer(staffID, hours)
      VALUES(“00000002”, 32)
    - INSERT INTO Volunteer(staffID, hours)
      VALUES(“00000003”, 400)
    - INSERT INTO Volunteer(staffID, hours)
      VALUES(“00000004”, 111)
- SalaryRanges(<u>position</u>: varchar[200], salary: integer)

- INSERT INTO SalaryRanges(position, salary) VALUES(“Administrative Assistant”, 50000)
- INSERT INTO SalaryRanges(position, salary) VALUES(“Veterinarian”, 90000)
- INSERT INTO SalaryRanges(position, salary) VALUES(“Social Media Administrator”, 60000)
- INSERT INTO SalaryRanges(position, salary) VALUES(“Cleaning Assistant”, 55000)
- INSERT INTO SalaryRanges(position, salary) VALUES(“CEO”, 120000)

- Employee(**<u>staffID</u>**: varchar[8], **position**: varchar[200])
    - INSERT INTO Employee(staffID, position) VALUES(“00000005”, “Administrative Assistant”)
    - INSERT INTO Employee(staffID, position) VALUES(“00000006”, “Veterinarian”)
    - INSERT INTO Employee(staffID, position) VALUES(“00000007”, “Social Media Administrator”)
    - INSERT INTO Employee(staffID, position) VALUES(“00000008”, “Cleaning Assistant”)
    - INSERT INTO Employee(staffID, position) VALUES(“00000009”, “CEO”)
- AnimalAdmits(<u>animalID:</u> varchar[8], **breed**: varchar[200], name: varchar[200], age: integer, **branchID**: varchar[8], date: date)
    - INSERT INTO AnimalAdmits (animalID, breed, name, age, branchID, date) VALUES (“1”, “Dog”, “Mochi”, 2, “1”,”12345678”, “2023-02-01”)
    - INSERT INTO AnimalAdmits (animalID, breed, name, age, branchID, date) VALUES (“2”, “Cat”, “Biscuit”, 5,”12345678”, “2023-02-01”)
    - INSERT INTO AnimalAdmits (animalID, breed, name, age, branchID, date) VALUES (“3”, “German Shepherd”, “Rex”, 0,”12345678”, “2023-03-01”)
    - INSERT INTO AnimalAdmits (animalID, breed, name, age, branchID, date) VALUES (“4”, “Siamese Cat”, “Biscuit”, 2,”12345679”, “2023-04-01”)
    - INSERT INTO AnimalAdmits (animalID, breed, name, age, branchID, date) VALUES (“5”, “French Bulldog”, “Sasha”, 12,”12345680”, “2023-04-01”)
- Vaccination(<u>type:</u> varchar[200], <u>date:</u> date, **<u>animalID:</u>** varchar[8])
    - INSERT INTO Vaccination (type,date,animalID) VALUES (“Rabies”, “2023-01-01”, “1”)
    - INSERT INTO Vaccination (type,date,animalID) VALUES (“Flu”, “2023-01-01”, “2”)

    - INSERT INTO Vaccination (type,date,animalID) VALUES (“Rabies”, “2023-01-02”, “1”)
    - INSERT INTO Vaccination (type,date,animalID) VALUES (“Rabies”, “2023-01-04”, “1”)
    - INSERT INTO Vaccination (type,date,animalID) VALUES (“Rabies”, “2023-01-02”, “2”)
- AnimalInfo(<u>breed</u>: varchar[200], species: varchar[200])
    - INSERT INTO AnimalInfo (breed, species) VALUES (“Dog”, “Dog”)
    - INSERT INTO AnimalInfo (breed, species) VALUES (“Cat”, “Cat”)
    - INSERT INTO AnimalInfo (breed, species) VALUES (“German Shepherd”, “Dog”)
    - INSERT INTO AnimalInfo (breed, species) VALUES (“Siamese Cat”, “Cat”)
    - INSERT INTO AnimalInfo (breed, species) VALUES (“French Bulldog”, “Dog”)
- Adopters(<u>adopterID:</u> varchar[8], lastName: varchar[200], firstName: varchar[200], phoneNum: varchar[10])
    - INSERT INTO Adopters(adopterID, lastName, firstName, phoneNum) VALUES(“100”, “Smith”, “John”, “604-312-1111”)
    - INSERT INTO Adopters(adopterID, lastName, firstName, phoneNum) VALUES(“101”, “Smythe”, “Jon”, “604-312-1112”)
    - INSERT INTO Adopters(adopterID, lastName, firstName, phoneNum) VALUES(“102”, “Adams”, “Gerald”, “604-312-1113”)
    - INSERT INTO Adopters(adopterID, lastName, firstName, phoneNum) VALUES(“103”, “Lillard”, “Leslie”, “604-312-1114”)
    - INSERT INTO Adopters(adopterID, lastName, firstName, phoneNum) VALUES(“104”, “Rapport”, “Rita”, “604-312-1115”)
- Attends(<u>**donorID:**</u> varchar[8], <u>**location:**</u> varchar[200], <u>**date**:</u> date, <u>**title:**</u> varchar[200])
    - INSERT INTO Attends (donorID, location, date, title) VALUES (“1”, “Stanley Park”, “2023-07-15”, “PetPalooza”)
    - INSERT INTO Attends (donorID, location, date, title) VALUES (“2”, “New Brighton Park”, “2021-03-04”, “Furry Friends Fair”)
    - INSERT INTO Attends (donorID, location, date, title) VALUES (“3”, “Olympic Oval”, “2022-05-10”, “Strut for Strays 5K”)
    - INSERT INTO Attends (donorID, location, date, title) VALUES (“4”, “Vancouver Art Gallery”, “2029-09-15”, “Paws & Paint Night”)
    - INSERT INTO Attends (donorID, location, date, title) VALUES (“5”, “Queen Elizabeth Park”, “2022-10-31”, “Barktoberfest”)
- Donates(<u>**donorID:**</u> varchar[8], <u>**branchID**:</u> varchar[8], date: date, amount: float)
    - INSERT INTO Donates (donorID, branchID, date, amount) VALUES (“1” ,”12345678”, “2008-07-12”, “3000.00”)

    - INSERT INTO Donates (donorID, branchID, date, amount) VALUES (“2”, “12345679”, “2003-01-05”, “100.00”)
    - INSERT INTO Donates (donorID, branchID, date, amount) VALUES (“3”, “12345680”, “2000-09-20”, “500.00”)
    - INSERT INTO Donates (donorID, branchID, date, amount) VALUES (“4”, “12345681”, “2020-11-08”, “50.00”)
    - INSERT INTO Donates (donorID, branchID, date, amount) VALUES (“5”, “12345682”, “2022-02-14”, “888.88”)
- Plans(**<u>location:</u>** varchar[200], **<u>date</u>**<u>:</u> date, **<u>title:</u>** varchar[200], **<u>branchID</u>**<u>:</u> varchar[8])
    - INSERT INTO Plans (locations,date,title,branchID)
      VALUES (“Stanley Park”, “2023-07-15”, “PetPalooza”,“12345678”)
    - INSERT INTO Plans (locations,date,title,branchID)
      VALUES (“New Brighton Park”, “2021-03-04”, “Furry Friends Fair”,“12345679”)
    - INSERT INTO Plans (locations,date,title,branchID)
      VALUES (“Olympic Oval”, “2022-05-10”, “Strut for Strays 5K”,“12345678”)
    - INSERT INTO Plans (locations,date,title,branchID)
      VALUES (“Vancouver Art Gallery”, “2029-09-15”, “Paws & Paint Night”,“12345679”)
    - INSERT INTO Plans (locations,date,title,branchID)
      VALUES (“Queen Elizabeth Park”, “2022-10-31”, “Barktoberfest”,“12345680”)
- Employs(**<u>staffID</u>**<u>:</u> varchar[8], **<u>branchID</u>**<u>:</u> varchar[8], contractID: varchar[8])
    - INSERT INTO Employs(staffID,branchID,contractID) VALUES (“00000000”,“12345678”, “1”)
    - INSERT INTO Employs(staffID,branchID,contractID) VALUES (“00000001”,“12345678”, “2”)
    - INSERT INTO Employs(staffID,branchID,contractID) VALUES (“00000002”,“12345678”, “3”)
    - INSERT INTO Employs(staffID,branchID,contractID) VALUES (“00000003”,“12345678”, “4”)
    - INSERT INTO Employs(staffID,branchID,contractID) VALUES (“00000004”,“12345679”, “5”)
    - INSERT INTO Employs(staffID,branchID,contractID) VALUES (“00000005”,“12345679”, “6”)
    - INSERT INTO Employs(staffID,branchID,contractID) VALUES (“00000006”,“12345679”, “7”)
    - INSERT INTO Employs(staffID,branchID,contractID) VALUES (“00000007”,“12345679”, “8”)

    - INSERT INTO Employs(staffID,branchID,contractID) VALUES (“00000008”,“12345679”, “9”)
    - INSERT INTO Employs(staffID,branchID,contractID) VALUES (“00000009”,“12345678”, “10”)
- Apply(<u>**branchID**</u>: varchar[8], <u>**adopterID**</u>: varchar[8], <u>**animalID**</u>: varchar[8], status: varchar[200], date: date)
    - INSERT INTO Apply(branchID, adopterID, animalID, status, date) VALUES(“12345678”, “100”, “1”, “Pending”, “2023-01-14”)
    - INSERT INTO Apply(branchID, adopterID, animalID, status, date) VALUES(“12345678”, “101”, “1”, “Rejected”, “2023-01-05”)
    - INSERT INTO Apply(branchID, adopterID, animalID, status, date) VALUES(“12345678”, “102”, “1”, “Rejected”, “2023-01-05”)
    - INSERT INTO Apply(branchID, adopterID, animalID, status, date) VALUES(“12345678”, “103”, “1”, “Pending”, “2023-01-09”)
    - INSERT INTO Apply(branchID, adopterID, animalID, status, date) VALUES(“12345678”, “104”, “1”, “Pending”, “2023-01-11”)