**The first user group: Vet and Vet Technicians.** These users must be able to insert and read data about care being performed on the patients. This information includes vitals measurements, as well as any treatments or procedures. Vitals, such as age, weight, and blood pressure are all numerical in nature. Certain obvious constraints apply to these categories, such as they cannot be negative, age and bp are integers where weight can be a decimal number (only out to the tenths place). There also must be a few text fields with less constraints. There must be a varchar field, perhaps 30 characters, such that non-numerical treatments can be entered: Name of medicine or procedure, and administered dosage (varchar, so we don’t have to mess with unit conversions). Additionally, a large text field (1-2k varchar) must exist for important notes relating to the procedure. There are little constraints on this information, it is left as very freeform text. This field is most important for more complicated/extensive procedures. All of this treatment information must be stored, along with date information, so as to maintain a record of the patient’s health over time. The patient (animal) itself should be stored in a relation of its own and joinable to the vitals/treatments. The patient will have information such as its name, owner, species and breed. These are all varchar fields.  
  
**The second user group: Front desk, clerical, administrative** **staff.** These users are primarily involved in billing and scheduling. These users need access to the care provided to a patient, and they need to be able to join this to data that contains costs of procedures. The treatment data contains numerical information (out to the hundredths place) for the cost of a standard dosage. This treatment information is used in conjunction with the patient’s care information, allowing the cost of a patient receiving non-standard dosages to be calculated. The cost can be calculated as a fraction either in the SQL itself, or the application that the user is using. Calculating cost on the fly is a difficult task, as dosages are not all measured in the same units (pills, ounces, minutes). Either way, often the amount owed will not typically match the amount in the ‘cost’ table, as the administered dosage likely won’t match the standard dosage we have a cost stored for. This user group must have access to scheduling. Thus, there must be data containing date and time information. Not all vet visits are the same length, so there is the concept of a ‘duration’ as well. This can be implemented as a start and end time (time data type). Constraints can be set up to prevent double-booking, although this could be tricky. The scheduling application software could also be used to protect against double booking. Finally, information about the veterinarians themselves is necessary, as there could be multiple vets with appointments simultaneously. Vet information will be a first name (varchar), last name (varchar), and id (int, unique). In the future, this can be set up with a ‘role’ varchar attribute, in case this database was expanded in the future to include additional functionality, such as managing payroll or employee scheduling. The current implementation does not plan to service those needs, but is easily extensible in the future.

**The last user group #3: Customers**. Customers need to have access to a lot of the same information as the first two users, but in a much more restricted sense. Customers should have access to scheduling information only pertinent to their pets. In addition, they are given access to itemized receipts for the care of their pets, but do not have access to the full care notes, nor do they have access to the marginal cost of the medicines/procedures. For online billing, customers will have access to their account, which is a relation that contains customer first and last name (varchar), a unique customer id (int), and a FK join in all of the pets that they own. Additionally, the customer should have insert/update access to a relation that contains billing information. The customer should be able to store credit card information, including the CC number (varchar), expiration date (date), and the security code (varchar). Finally, there should be several attributes for billing‑street address (varchar), zip code (varchar), town and state (varchar for both). The billing information should be one to many, as one user may have several cards on file.

Combining all of my requirements: First and foremost, I did not see the need to make any compromises in terms of attribute data types, so the attributes as laid out above can stay as listed. In the first design, there will be a relation containing information about care (“chart”), which has attributes described in the first section. This relation is accessible by all three user groups, although the customers will not have access to the detailed notes section. This relation will need to be joinable to a “treatment” relation, which contains medicines and procedures, as well as a standard dosage/time, and a marginal cost. The ‘chart’ relation will need to also be joinable against the “vet” table, and the “patient” table. I will not make it joinable against the scheduling table, however, and thus the chart table will have its own date stored.

The vet table will be joinable against the schedule table as well as the appointment table. The patient table (containing information about pets) needs to contain a natural way to join it again chart, schedule, as well as customer. Customer only needs to be joinable against schedule, patient, and billing.  
 Overall, making this system work for the three user groups seems to be more of granting/restricting access, rather than converting between different domains.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Customer | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| cid | int | 8 | Unique, not null | ON DELETE RESTRICT  Don’t leave orphaned patients in the DB |
| fname | varchar | 50 | Not null |  |
| lname | varchar | 50 | Not null |  |
| phone | varchar | 11 | Not null |  |
| email | varchar | 50 |  |  |

**Primary Key(s):**

cid

**Foreign Key(s):**

**Indices:**

fname, lname

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PATIENT | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| pid | int | 8 | Unique, not null | ON DELETE RESTRICT Patients who have had appointments should not be delete-able, as the vets office will want record of all transactions. |
| name | varchar | 50 | Not null |  |
| species | varchar | 30 | Not null |  |
| breed | varchar | 30 |  |  |
| customer\_id | int | 8 | Not null |  |

**Primary Key(s):**

pid

**Foreign Key(s):**

customer\_id REFERENCES customer(cid)

**Indices:**

customer\_id, species

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TREATMENT | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| tid | int | 8 | Unique, not null | None, no tables reference this one. |
| treatment | varchar | 30 | Not null, upper, no trailing/leading whitespace |  |
| dosage | decimal |  | Not null |  |
| dosage\_unit | varchar | 15 |  |  |
| cost | numeric | 7, 2 | Positive |  |

**Primary Key(s):**

tid

**Foreign Key(s):**

**Indices:**

treatment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VITALS | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| vid | int | 8 | Unique, not null | ON DELETE RESTRICT  This should not be deleted unless the whole appointment is. |
| weight | numeric | 5, 1 | Not null |  |
| diastolic\_bp | smallint |  | Positive |  |
| systolic\_bp | smallint |  | Positive |  |

**Primary Key(s):**

vid

**Foreign Key(s):**

**Indices:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| APPOINTMENT | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| date | date |  | Not null | None, no tables reference this one. |
| patient\_id | int | 8 | Not null |  |
| vet\_id | int | 8 | Not null |  |
| vitals\_id | int | 8 | Not null |  |
| treatment\_id | int | 8 | Not null |  |
| admin\_dosage | decimal |  | Not null |  |
| admin\_dosage\_unit | varchar | 15 |  |  |
| cost | numeric | 7, 2 | Positive |  |

**Primary Key(s):**

date, patient\_id, treatment\_id

**Foreign Key(s):**

patient\_id REFERENCES patient(pid)

treatment\_id REFERENCES treatment(tid)

vet\_id REFERENCES employee(eid)

vitals\_id REFERENCES vitals(vid)

**Indices:**

vet\_id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EMPLOYEE | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| eid | int | 8 | Not null, unique | Yes, make FK null in other tables, rather than delete tuples. |
| fname | varchar | 50 | Not null |  |
| lname | varchar | 50 | Not null |  |

**Primary Key(s):**

eid

**Foreign Key(s):**

**Indices:**

lname

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BILLING | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| bid | int | 8 | Not null, unique | None, no tables references this one. |
| customer\_id | int | 8 | Not null |  |
| card\_no | varchar | 16 | Not null, numerical |  |
| exp\_date | date |  | Not null |  |
| security\_code | varchar | 4 | Not null, numerical |  |
| street\_address | varchar | 30 | Not null |  |
| town | varchar | 30 | Not null |  |
| state | varchar | 2 | Not null, upper case |  |
| zip | char | 5 | Not null |  |

**Primary Key(s):**

bid

**Foreign Key(s):**

customer\_id REFERENCES customer(cid)

**Indices:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SCHEDULE | | | | |
| Attribute Name | Data Type | Size | Constraints/Domain | Cascading Problems |
| customer\_id | int | 8 | Not null | None, no tables reference this one. Appointment cancellations are expected |
| patient\_id | int | 8 | Not null |  |
| date | date |  | Not null |  |
| time\_start | time |  | Not null |  |
| time\_end | time |  | Not null |  |
| vet\_id | int | 8 | Not null |  |

**Primary Key(s):**

patient\_id, date

**Foreign Key(s):**

customer\_id REFERENCES customer(cid)

patient\_id REFERENCES patient(pid)

vet\_id REFERENCES vet(eid)

**Indices:**

vet\_id, time\_start, time\_end

**Normalization:**

First, a new UNF relation is created, containing all of our attributes. Foreign key attributes are omitted, as they are just duplicates of the primary key attributes in the table they reference. There is one derived attribute, */cost*.

**A screenshot of a social media post

Description automatically generated** To normalize the UNF relation, the existing attributes are atomized, and a primary key is selected. In order to atomize the data, tuples are created in order to flatten out any multivalued attributes, and any composite attributes (like name) are broken into their parts. As there are no tuples in this table, there are no multi-valued columns that need to be expanded. Additionally, this data is largely already atomic; specifically, names are already in fname/lname form. The only other attribute that could be decomposed into parts is the street address (number and street), but that level of granularity is not useful for billing information. Another consideration in the UNF is the existence of multiple attributes with the same name that contain slightly different information. In order to differentiate the attributes with duplicated names, they are renamed before moving forward. Renamed attributes are shown in blue in the 1NF relation.

Finally, to achieve 1NF, a primary key is selected that uniquely identifies every tuple in the table (shown highlighted in gold). The “id” attributes compartmentalize most of the data very well, making selecting a PK fairly straightforward. Attribute vid (vitals\_id) is removed, as we noted later on this is an un-needed attribute, as date+pid serve the same purpose. We can also remove bid (billing\_id) for the same reason: cid+card\_no uniquely identify billing data. Both vid and bid are meaningless attributes in and of themselves. Additionally, the second ‘date’ column is removed. Lastly, a new attribute ‘conversion’ is added at this stage, to facilitate unit conversions for medicines/procedures.

While decomposing this relation into several new relations to meet 2NF, I note a previous unspoken assumption. For this design, I am assuming that any given patient will only have a single appointment in a given day. If we wanted to change that, it we would want to add *time\_start* to the *vitals* relation. That is, because we have one appointment per day per patient, *time\_start* is derivable from pid+date\_appointment, and would be removed in the next normalization step. For now, I will continue with the assumption that a patient will only have one appointment per day, maximum.

Additionally, with the attribute removals and relation refactoring, I noted that the vitals relation and the schedule relation now have the same primary keys, and had a natural one to one relation. When converting to 2NF, these previously separate relations are now part of the same ‘appointment’ relation.

**A screenshot of a social media post

Description automatically generated**

Now, we can check to see if we meet 3NF. In order to meet 3NF, we have to make sure that no attribute depends on a non-PK attribute. Looking through our tables, provided the attribute changes we made in 1NF, we can see that our 2NF tables already meet 3NF, without any additional changes.

Table creation, population, and example queries are shown below. The table creation and population are in an equal space font (courier), as it makes the code easier to read. Please note, all example queries based on pet names also include owner names, as it is possible two owners name their pet the same. It is assumed a every pet owned by a single owner has a unique name.

-- I do not know if you wanted the DB creation code, as well. This is the code I used:

CREATE DATABASE vet

WITH

OWNER = postgres

ENCODING = 'UTF8'

CONNECTION LIMIT = -1;

-- table definitions

CREATE TABLE customer(

cid INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

c\_fname VARCHAR(50) NOT NULL,

c\_lname VARCHAR(50) NOT NULL,

phone VARCHAR(11) NOT NULL,

email VARCHAR(50)

);

CREATE TABLE patient(

pid INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

name VARCHAR(50) NOT NULL,

species VARCHAR(30) NOT NULL,

breed VARCHAR(30)

);

CREATE TABLE conversion(

admin\_dosage\_unit VARCHAR(15) NOT NULL,

dosage\_unit VARCHAR(15) NOT NULL,

ratio DECIMAL NOT NULL,

PRIMARY KEY (admin\_dosage\_unit, dosage\_unit)

);

CREATE TABLE treatment(

tid INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

treatment VARCHAR(30) NOT NULL,

dosage DECIMAL NOT NULL,

dosage\_unit VARCHAR(15),

cost NUMERIC(7,2),

CONSTRAINT uppercase CHECK (upper(treatment) = treatment),

CONSTRAINT trimmed CHECK (trim(treatment) = treatment)

);

CREATE TABLE employee(

eid INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

fname VARCHAR(50) NOT NULL,

lname VARCHAR(50) NOT NULL

);

CREATE TABLE billing(

customer\_id INT NOT NULL,

card\_no CHAR(16) NOT NULL,

exp\_date DATE NOT NULL,

security\_code VARCHAR(4) NOT NULL,

street\_address VARCHAR(30) NOT NULL,

town VARCHAR(30) NOT NULL,

state CHAR(2) NOT NULL,

zip CHAR(5) NOT NULL,

CONSTRAINT card\_length CHECK (length(card\_no) = 16),

CONSTRAINT card\_numeric CHECK (card\_no ~\* '^[0-9]+$'),

CONSTRAINT code\_length CHECK (length(security\_code) >= 3),

CONSTRAINT code\_numeric CHECK (security\_code ~\* '^[0-9]+$'),

CONSTRAINT expired CHECK (exp\_date >= CURRENT\_DATE),

PRIMARY KEY (customer\_id, card\_no),

FOREIGN KEY (customer\_id) REFERENCES customer(cid)

);

CREATE TABLE appointment(

date DATE NOT NULL,

pid INT NOT NULL,

eid INT NOT NULL,

time\_start TIME NOT NULL,

time\_end TIME NOT NULL,

weight NUMERIC(5,1) CHECK (weight > 0),

diastolic\_bp SMALLINT CHECK (diastolic\_bp > 0),

systolic\_bp SMALLINT CHECK (systolic\_bp > 0),

CONSTRAINT valid\_time CHECK (time\_start < time\_end),

PRIMARY KEY (date, pid),

FOREIGN KEY (eid) REFERENCES employee,

FOREIGN KEY (pid) REFERENCES patient

);

-- constraint created after, as it depends on appointment existing

CREATE FUNCTION check\_schedule(vet\_id int, appt\_date date, appt\_start time, appt\_end time)

RETURNS boolean AS $$

SELECT NOT EXISTS (

SELECT \*

FROM appointment

WHERE appointment.date = appt\_date

AND appointment.eid = vet\_id

AND appt\_start < appointment.time\_end

AND appt\_end > appointment.time\_start);

$$ LANGUAGE 'sql';

ALTER TABLE appointment ADD CONSTRAINT schedule\_open

CHECK (check\_schedule(eid, date, time\_start, time\_end));

CREATE TABLE administered(

date DATE NOT NULL,

pid INT NOT NULL,

tid INT NOT NULL,

admin\_dosage DECIMAL NOT NULL,

admin\_dosage\_unit VARCHAR(15),

PRIMARY KEY (date, pid, tid),

FOREIGN KEY (date, pid) REFERENCES appointment,

FOREIGN KEY (tid) REFERENCES treatment

);

CREATE TABLE owner(

pid INT NOT NULL,

cid INT NOT NULL,

PRIMARY KEY (pid, cid),

FOREIGN KEY (pid) REFERENCES patient,

FOREIGN KEY (cid) REFERENCES customer

);

-- create appropriate indexes to speed up expected queries

CREATE INDEX ind\_cust\_fname

ON customer (c\_fname);

CREATE INDEX ind\_cust\_lname

ON customer (c\_lname);

CREATE INDEX ind\_species

ON patient (species);

CREATE INDEX ind\_treatment

ON treatment (treatment);

CREATE INDEX ind\_empname

ON employee (lname);

CREATE INDEX ind\_schedule

ON appointment (time\_start, time\_end);

-- inserting data

INSERT INTO customer(c\_fname, c\_lname, phone, email)

VALUES ('Vick', 'Doorman', '14955554938', 'trickyVick@gmail.com'),

('John', 'Smith', '18025551234', 'john.smith@metlife.org'),

('Fred', 'Baker', '18025554893', null),

('Bill', 'Archer', '17525558473', 'archery\_season@hotmail.com');

INSERT INTO patient(name, species, breed)

VALUES ('Abner', 'Cat', 'Tabby'),

('Raz', 'Dog', 'Pit Bull'),

('Creamy', 'Cow', 'Angus'),

('Raoul', 'Cat', null),

('Adun', 'Dog', 'Doberman');

INSERT INTO owner(pid, cid)

VALUES

(1, 1),

(2, 1),

(3, 2),

(4, 3),

(5, 4);

INSERT INTO treatment(treatment, dosage, dosage\_unit, cost)

VALUES ('ADVIL', 200, 'mg', 5.75),

('CPR', 1, 'minute', 150),

('BLOOD PRESSURE', 1, null, 15),

('BLOOD DRAW', 1, null, 25),

('GENERAL EVALUATION', 1, null, 50),

('FLEA MEDICATION, ORAL', 25, 'mg', 65.50),

('BLOOD TEST', 1, null, 45.50),

('PREDNISONE', 75, 'mg', 78.25),

('RABIES VACCINE', 1, null, 94),

('GENERAL ANESTEHTIC', 45, 'mcg', 225.80);

INSERT INTO billing(customer\_id, card\_no, exp\_date, security\_code, street\_address, town, state, zip)

VALUES (1, '1234567890123456', '2020-03-01', '123', '14 Wilson Road', 'Johnson', 'KY', '41219'),

(2, '8367583739593262', '2020-06-01', '4964', '6 Apeture Lane', 'Smallville', 'KY', '41220'),

(3, '9584763839694877', '2023-08-01', '475', '45 Maple Tree Place, Apt 1', 'Smallville', 'KY', '41220'),

(4, '7469567384889299', '2022-07-01', '113', '68 Main Street', 'Burlington', 'KY', '41524'),

(4, '7469843918939299', '2020-02-01', '223', '68 Main Street', 'Burlington', 'KY', '41524');

INSERT INTO employee(fname, lname)

VALUES ('Robert', 'Baratheon'),

('James', 'Kirk');

INSERT INTO appointment(date, eid, time\_start, time\_end, pid, weight, diastolic\_bp, systolic\_bp)

VALUES ('2016-06-24', 2, '14:30', '15:00', 1, 12.3, null, 125),

('2016-12-20', 2, '14:30', '15:00', 1, 12.5, null, 126),

('2017-05-15', 1, '09:00', '09:30', 1, 12.3, null, 125),

('2018-07-30', 2, '12:15', '12:45', 1, 12.3, null, 128),

('2019-04-04', 2, '08:00', '08:30', 1, 12.4, null, 122),

('2016-06-24', 1, '14:30', '15:15', 2, 55, null, 130),

('2017-09-29', 1, '14:30', '15:00', 2, 53, null, 131),

('2019-02-26', 1, '10:45', '11:15', 2, 53, null, 131),

('2019-08-28', 1, '15:15', '15:45', 3, 2215, 17, 24),

('2016-07-30', 2, '11:15', '11:45', 4, 18.1, null, 140),

('2017-10-04', 2, '07:00', '07:30', 4, 16.2, null, 138),

('2017-06-14', 1, '12:30', '12:45', 4, 15.5, null, 135),

('2019-07-19', 2, '12:30', '13:00', 4, 13, null, 130),

('2017-07-02', 2, '09:15', '09:45', 5, 75, null, 133),

('2019-09-01', 2, '09:15', '09:45', 5, 74, null, 135);

INSERT INTO administered(date, pid, tid, admin\_dosage, admin\_dosage\_unit)

VALUES

-- Abner

('2016-06-24', 1, 5, 1, null),

('2016-06-24', 1, 1, 200, 'mg'),

('2016-06-24', 1, 6, 25, 'mg'),

('2016-12-20', 1, 5, 1, null),

('2017-05-15', 1, 5, 1, null),

('2017-05-15', 1, 4, 1, null),

('2018-07-30', 1, 5, 1, null),

('2019-04-04', 1, 5, 1, null),

('2019-04-04', 1, 6, 12.5, 'mg'),

--Raz

('2016-06-24', 2, 5, 1, null),

('2016-06-24', 2, 4, 1, null),

('2016-06-24', 2, 7, 1, null),

('2017-09-29', 2, 5, 1, null),

('2017-09-29', 2, 8, 150, 'mg'),

('2019-02-26', 2, 5, 1, null),

--Creamy

('2019-08-28', 3, 5, 1, null),

('2019-08-28', 3, 10, 175, 'mcg'),

--Raoul

('2016-07-30', 4, 5, 1, null),

('2017-10-04', 4, 5, 1, null),

('2017-10-04', 4, 9, 1, null),

('2017-06-14', 4, 5, 1, null),

('2019-07-19', 4, 5, 1, null),

--Adun

('2017-07-02', 5, 5, 1, null),

('2019-09-01', 5, 5, 1, null);

INSERT INTO conversion

VALUES ('mg', 'mcg', 1000),

('mcg', 'mg', 1/1000),

('g', 'mg', 1000),

('mg', 'g', 1/1000);

1. **User 1: Veterinarian/Vet Technician**
   1. **Query 1**
      1. “When was the last time Vick Doorman’s pet Cat, Abner, received flea medication from our office?
      2. SELECT MAX(date) as most\_recent\_flea\_med

FROM administered

INNER JOIN patient ON administered.pid = patient.pid

INNER JOIN treatment ON administered.treatment\_id = treatment.tid

INNER JOIN owner ON patient.pid = owner.pid

INNER JOIN customer ON owner.cid = customer.cid

WHERE patient.name = 'Abner'

AND customer.c\_fname = 'Vick'

AND customer.c\_lname = 'Doorman'

AND treatment.treatment = 'FLEA MEDICATION, ORAL

* + 1. A screenshot of a cell phone

       Description automatically generated
  1. **Query 2 (nested select)**
     1. “A new vet is taking over patient Raoul, and would like to see a summary of the two most recent vet visits.”
     2. SELECT date, name, treatment, admin\_dosage, admin\_dosage\_unit, employee.lname as vet

FROM administered

NATURAL JOIN patient

NATURAL JOIN owner

NATURAL JOIN customer

NATURAL JOIN appointment

NATURAL JOIN employee

NATURAL JOIN treatment

WHERE patient.name = 'Raoul'

AND customer.c\_fname = 'Fred'

AND customer.c\_lname = 'Baker'

AND date in (SELECT DISTINCT date

FROM appointment

NATURAL JOIN patient

NATURAL JOIN owner

f NATURAL JOIN customer

WHERE patient.name = 'Raoul'

AND customer.c\_fname = 'Fred'

AND customer.c\_lname = 'Baker'

ORDER BY date DESC

FETCH FIRST 2 ROWS ONLY)

* + 1. A screenshot of a cell phone

       Description automatically generated
  1. **Query 3**
     1. How has Fred Baker’s cat Raoul’s weight changed since we first started seeing him in at this office?
     2. SELECT date, weight

FROM administered

NATURAL JOIN patient

NATURAL JOIN owner

NATURAL JOIN customer

NATURAL JOIN appointment

NATURAL JOIN treatment

WHERE patient.name = 'Raoul'

AND customer.c\_fname = 'Fred'

AND customer.c\_lname = 'Baker'

ORDER BY date

* + 1. A screenshot of a cell phone

       Description automatically generated

1. **User 2: Front Desk**
   1. **Query 1**
      1. How many appointments have each of the vets had with dogs?
      2. SELECT lname as vet, COUNT(species) as frequency

FROM appointment

NATURAL JOIN patient

NATURAL JOIN employee

WHERE species = 'Dog'

GROUP BY vet

* + 1. A screenshot of a cell phone

       Description automatically generated
  1. **Query 2** (nested select)
     1. There was a recall on 2017’s rabies vaccine. What are the customer names, pet names, phone numbers, and emails of customers who’s pets received this vaccine in 2017?
     2. SELECT CONCAT(c\_fname, ' ', c\_lname) as name, phone, email, name as pet\_name

FROM patient

NATURAL JOIN owner

NATURAL JOIN customer

WHERE pid in (SELECT DISTINCT pid

FROM administered

NATURAL JOIN treatment

WHERE treatment = 'RABIES VACCINE'

AND EXTRACT (YEAR from date) = 2017)

* + 1. A screenshot of a cell phone

       Description automatically generated
  1. **Query 3**
     1. Are there certain months out of the year we see more general checkups?
     2. SELECT EXTRACT (MONTH FROM date) as month, COUNT(EXTRACT (MONTH FROM date)) as num\_appointments

FROM administered

NATURAL JOIN treatment

WHERE treatment = 'GENERAL EVALUATION'

GROUP BY EXTRACT (MONTH FROM date)

ORDER BY month

* + 1. A screenshot of a social media post

       Description automatically generated

1. **User 3: Customer**
   1. **Query 1 (nested select)**
      1. How much was I billed for each of my cow Creamy’s treatments?
      2. Notes: this is nested, allowing to use the result of our case statement as part of our inline function. Also, we assume any null unit values will cost the marginal amount  
           
         SELECT treatment, admin\_dosage, admin\_dosage\_unit, ROUND ((admin\_dosage / dosage \* cost \* conversion), 2) as total\_cost

FROM (SELECT treatment, admin\_dosage, admin\_dosage\_unit, dosage, cost,

CASE

WHEN admin\_dosage\_unit = dosage\_unit THEN 1

WHEN admin\_dosage\_unit is null THEN 1

WHEN dosage\_unit is null THEN 1

ELSE (SELECT ratio FROM conversion WHERE conversion.admin\_dosage\_unit = admin\_dosage\_unit AND conversion.dosage\_unit = dosage\_unit)

END as conversion

FROM administered

NATURAL JOIN treatment

NATURAL JOIN patient

NATURAL JOIN owner

NATURAL JOIN customer

WHERE name = 'Creamy'

AND c\_fname = 'John'

AND c\_lname = 'Smith') as medications

* + 1. A screenshot of a social media post

       Description automatically generated
  1. Query 2
     1. How long as it been since each of my pets has been to the vet?
     2. SELECT name, AGE(MAX(date)) as since\_last\_appointment

FROM appointment

NATURAL JOIN patient

NATURAL JOIN owner

NATURAL JOIN customer

WHERE c\_fname = 'Vick'

AND c\_lname = 'Doorman'

GROUP BY name

* + 1. A screenshot of a cell phone

       Description automatically generated
  1. **Query 3**
     1. Which, if any, of my payments on file are close to expiring (and what is the expiration date)?
     2. SELECT card\_no, exp\_date

FROM billing

INNER JOIN customer ON customer.cid = billing.customer\_id

WHERE c\_fname = 'Bill'

AND c\_lname = 'Archer'

AND AGE(exp\_date) > - interval '3 mons'

* + 1. A screenshot of a cell phone

       Description automatically generated