Addison Boyer

I. Introduction

Project Description

The Boyer Animal Clinic is a small family owned veterinary practice based out of Missoula Montana. The company contracts with a number of different veterinarians, and provides each veterinary professional with the office space, and supplies necessary to carry out their own individual practices. In exchange, the Boyer Animal Clinic receives a specified percentage of all capital from the sales and services of that individual practice. It is up to the veterinarians to supply their own staff, and the Boyer Animal Clinic gives veterinarians the freedom to pay as they see fit. The Boyer Animal Clinic requires a software system that will manage the client information from within each practice, while keeping it separated from the information of another practice who might also be contracting with the Boyer Animal Clinic. This will mitigate the potential for overlapping clients, and increase the reach of Boyer Animal Clinic in a localized area, and across the country.

Primary Users and Functions

1. Company Administration (Management).
    a. Perform routine audits to gauge the overall success of each clinic and/or veterinarian.
    b. View contracts, and individual clinic sales in order to calculate what is owed to Boyer Animal Clinic.
    c. Create, update, and renew contracts for new and existing veterinarians.
    d. View order requests from individual clinics to process them, and have them shipped to the clinic addresses stored in the database.
2. Veterinarians.
    a. View animal records, past prescriptions, and diagnoses given by themselves and/or other clinics within Boyer Animal Clinics.
    b. Create new examination records for pets brought into their clinic who have received treatment and/or services.
    c. Adjust, and or set Employee pay rates as well as create new employees for their clinics only.
    d. View order request information for accuracy, and completeness before sending them for administration for approval, and subsequent fulfillment.
    e. View clinic finances, in order to make adjustments, make informed decisions about ordering, and schedule employees accordingly.
3. Veterinary Technicians.
    a. View the remaining quantities of medicine, and supplies in order to make informed decisions about when to reorder them.

   b. View pet information required to administer vaccines, and perform less intensive procedures and checkups. For example, allergies, past vaccination dates, etc.
   c. View their own pay information, including rate of pay, number of hours, and bonuses.
   d. Create examination records for less intensive procedures, and checkups.
4. Office Receptionists (Staff).
   a. Book, cancel, and update appointments.
   b. Enter new client contact information into the system, as well as update existing client information.
   c. View client contact information of those who've made appointments at that clinic.
   d. View their own pay information, including rate of pay, number of hours, and bonuses.
5. Pet Owners.
   a. View appointment information, and invoices for their own pets only at all clinics.
   b. View examination history, as well as contact information for all clinics.
   c. Edit their own personal information, including phone numbers, and addresses.

II. Timeline

Semester long project organized in chronological order* Complete steps in the same order as outlined in this report.

III. Conceptual and Logical Model

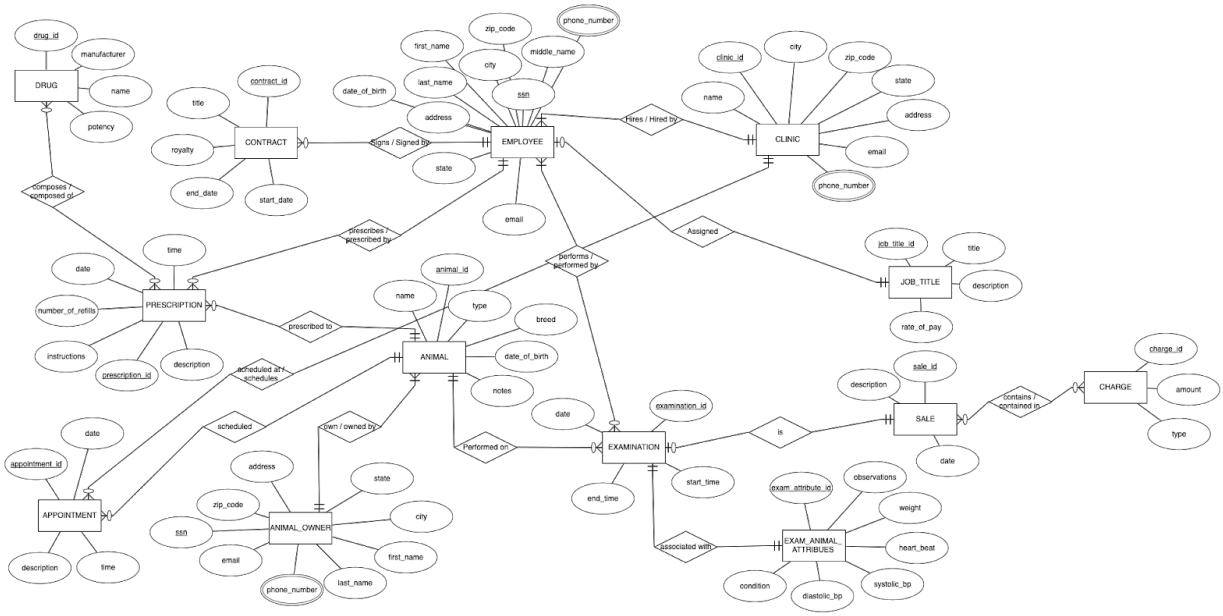
Data Requirements

- Employees are uniquely identified by a social security number. First, middle, and last name must be recorded, as well as a date of birth, email, phone number, and valid mailing address (city, state, and zip code).
- Job titles are uniquely identified by a job title id. Title, description, and rate of pay must be recorded.
- Employees are assigned exactly one job title, and job titles are assigned to only one employee.
- Clinics are uniquely identified by a clinic id. Name, email, phone number, and valid mailing address (city, state, zip code) must be recorded.
- Employees may work at only one clinic, and a single clinic may have multiple employees.

- Contracts are uniquely identified by a contract id. Title, start, and end date must be recorded as well as the royalty percentage between the veterinarian, and the Boyer Animal Clinic.
- Employees may have multiple contracts, but a single contract must be associated with one and only one employee. (Only veterinarians are contracted employees).
- Animal owners are uniquely identified by a social security number. First and last name must be recorded as well as phone number, email address, and valid mailing address (city, state, and zip code).
- Animals are uniquely identified by an animal id. Name, type, breed, date of birth, and notes (such as allergies, diet restrictions, and precautions) must be recorded.
- Animal owners can have multiple animals but an animal can be associated with one and only one owner.
- Appointments are uniquely identified by an appointment id. Date, time, and appointment description (reason for appointment) must be recorded.
- Appointments are scheduled for a single animal, and a single animal can have multiple appointments scheduled.
- Appointments can be scheduled at only one clinic, and a clinic can have multiple appointments scheduled at it.
- Examinations are uniquely identified by an examination id. Start and end time, and date must be recorded.
- Animal Examination Attributes are uniquely identified by an examination attribute id. Weight (lbs), heart beat (bpm), systolic and diastolic blood pressure, animal condition, and observation notes must be recorded.
- Animal examination attributes are associated with a single exam, and an exam has only one related animal examination attribute record.
- An examination is performed on only one animal, and a single animal can have multiple examinations.
- An examination can be performed by one or many employees, and a single employee can perform many examinations.
- A Prescription is uniquely identified by a prescription id. A date, time, description, instructions, and number of refills must be recorded.
- A drug is uniquely identified by a drug id. Manufacturer, name, and potency must be recorded.
- A prescription is prescribed to one and only one animal, and an animal may be prescribed multiple prescriptions.
- A prescription is composed of one or many drugs, and a drug can be prescribed in many prescriptions.
- A prescription can be prescribed by only one employee, and an employee may prescribe many prescriptions.

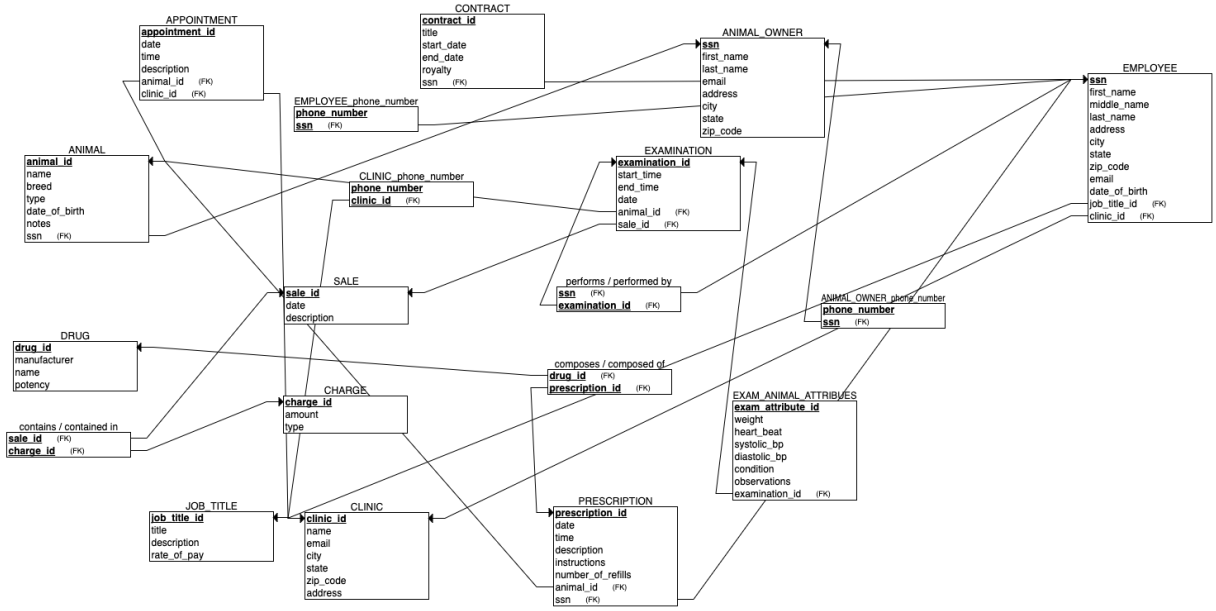
- Sales are uniquely identified by a sale id. Date, and description must be recorded.
- Each sale is associated with a single examination, and an examination corresponds to a single sale.
- A Charge is uniquely identified by a charge id. Name, amount, and type.
- A single charge can be found in many sales, and a sale can contain many charges.

Entity Relationship Diagram



IV. Physical Model

Relational Schema



V. Implementation

Database Tables

CREATE TABLE JOB_TITLE(job_title_id INT PRIMARY KEY NOT NULL, title VARCHAR(20), description VARCHAR(255), rate_of_pay FLOAT(2));

CREATE TABLE CLINIC(clinic_id INT PRIMARY KEY NOT NULL, name VARCHAR(40), email VARCHAR(320), city VARCHAR(20), state CHAR(2), zip_code VARCHAR(10), address VARCHAR(100));

CREATE TABLE CLINIC_PHONE_NUMBER(phone_number VARCHAR(20) NOT NULL, clinic_id INT NOT NULL, FOREIGN KEY (clinic_id) REFERENCES CLINIC(clinic_id), PRIMARY KEY(phone_number, clinic_id));

CREATE TABLE EMPLOYEE(ssn VARCHAR(20) PRIMARY KEY NOT NULL, first_name VARCHAR(20), middle_name VARCHAR(20), last_name VARCHAR(20), address VARCHAR(40), city VARCHAR(20), state CHAR(2), zip_code VARCHAR(10), email VARCHAR(320), date_of_birth DATE, job_title_id INT NOT NULL, clinic_id INT NOT NULL, FOREIGN KEY (job_title_id) REFERENCES JOB_TITLE(job_title_id), FOREIGN KEY (clinic_id) REFERENCES CLINIC(clinic_id));

```
CREATE TABLE EMPLOYEE_PHONE_NUMBER(phone_number VARCHAR(20) NOT NULL, ssn VARCHAR(20) NOT NULL, FOREIGN KEY (ssn) REFERENCES EMPLOYEE (ssn), PRIMARY KEY(phone_number, ssn));

CREATE TABLE DRUG(drug_id INT PRIMARY KEY NOT NULL, manufacturer VARCHAR(40), name VARCHAR(20), potency INT);

CREATE TABLE ANIMAL_OWNER(ssn VARCHAR(20) PRIMARY KEY NOT NULL, first_name VARCHAR(20), last_name VARCHAR(20), email VARCHAR(320), address VARCHAR(40), city VARCHAR(20), state CHAR(2), zip_code VARCHAR(10));

CREATE TABLE ANIMAL_OWNER_PHONE_NUMBER(phone_number VARCHAR(20) NOT NULL, ssn VARCHAR(20) NOT NULL, FOREIGN KEY (ssn) REFERENCES ANIMAL_OWNER (ssn), PRIMARY KEY(phone_number, ssn));

CREATE TABLE ANIMAL(animal_id INT PRIMARY KEY NOT NULL, name VARCHAR(20), breed VARCHAR(20), type VARCHAR(10), date_of_birth DATE, notes VARCHAR(300), ssn VARCHAR(20) NOT NULL, FOREIGN KEY (ssn) REFERENCES ANIMAL_OWNER(ssn));

CREATE TABLE PRESCRIPTION(prescription_id INT PRIMARY KEY NOT NULL, p_date DATE, time DATE, description VARCHAR(300), instructions VARCHAR(300), number_of_refills INT, animal_id INT NOT NULL, ssn VARCHAR(20) NOT NULL, FOREIGN KEY (animal_id) REFERENCES ANIMAL (animal_id), FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn));

CREATE TABLE PRESCRIPTION_DRUG(drug_id INT NOT NULL, prescription_id INT NOT NULL, FOREIGN KEY (prescription_id) REFERENCES PRESCRIPTION(prescription_id), FOREIGN KEY (drug_id) REFERENCES DRUG(drug_id), PRIMARY KEY(drug_id, prescription_id));

CREATE TABLE APPOINTMENT(appointment_id INT PRIMARY KEY NOT NULL, a_date DATE, time DATE, description VARCHAR(300), animal_id INT NOT NULL, clinic_id INT NOT NULL, FOREIGN KEY (animal_id) REFERENCES ANIMAL(animal_id), FOREIGN KEY (clinic_id) REFERENCES CLINIC(clinic_id));

CREATE TABLE SALE(sale_id INT PRIMARY KEY NOT NULL, s_date DATE, description VARCHAR(300));
```

```
CREATE TABLE EXAMINATION(examination_id INT PRIMARY KEY NOT NULL, start_time DATE, end_time DATE, e_date DATE, animal_id INT NOT NULL, sale_id INT NOT NULL, FOREIGN KEY (animal_id) REFERENCES ANIMAL(animal_id), FOREIGN KEY (sale_id) REFERENCES SALE(sale_id));

CREATE TABLE EXAM_ANIMAL_ATTRIBUTES(exam_attribute_id INT PRIMARY KEY NOT NULL, weight FLOAT(2), heart_beat INT, systolic_bp INT, diastolic_bp INT, animal_condition VARCHAR(20), observations VARCHAR(300), examination_id INT NOT NULL, FOREIGN KEY (examination_id) REFERENCES EXAMINATION(examination_id));

CREATE TABLE EMPLOYEE_EXAMINATION(ssn VARCHAR(20) NOT NULL, examination_id INT NOT NULL, FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn), FOREIGN KEY (examination_id) REFERENCES EXAMINATION(examination_id),PRIMARY KEY (ssn, examination_id));

CREATE TABLE CONTRACT(contract_id INT PRIMARY KEY NOT NULL, title VARCHAR(100), start_date DATE, end_date DATE, royalty FLOAT(2), ssn VARCHAR(20) NOT NULL, FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn));

CREATE TABLE CHARGE(charge_id INT PRIMARY KEY NOT NULL, amount FLOAT(2), type VARCHAR(20));

CREATE TABLE SALE_CHARGE(sale_id INT NOT NULL, charge_id INT NOT NULL, FOREIGN KEY (sale_id) REFERENCES SALE(sale_id), FOREIGN KEY (charge_id) REFERENCES CHARGE(charge_id), PRIMARY KEY(sale_id, charge_id));
```

Views

```
# Create dynamic predicate that gets the owners information only
CREATE VIEW OWNER_ANIMAL_INFO AS
SELECT SUBSTR(ANIMAL_OWNER.ssn, LENGTH(ANIMAL_OWNER.SSN)-4, 4) as last_4, first_name, last_name, phone_number, email, name as pet_name, type, notes from ANIMAL_OWNER JOIN ANIMAL ON ANIMAL_OWNER.ssn = ANIMAL.ssn JOIN ANIMAL_OWNER_PHONE_NUMBER ON ANIMAL_OWNER.ssn = ANIMAL_OWNER_PHONE_NUMBER.ssn;
```

```
# Create dynamic predicate that gets only the clinic of the employee
CREATE VIEW DAILY_APPOINTMENTS AS
SELECT SUBSTR(ANIMAL_OWNER.ssn, LENGTH(ANIMAL_OWNER.SSN)-4, 4)
as verify_ssn, first_name, last_name, phone_number, email, name as animal_name,
breed,  type, notes,time, description from ANIMAL_OWNER JOIN ANIMAL ON
ANIMAL.ssn = ANIMAL_OWNER.ssn JOIN ANIMAL_OWNER_PHONE_NUMBER
ON ANIMAL_OWNER_PHONE_NUMBER.ssn = ANIMAL_OWNER.ssn JOIN
APPOINTMENT ON ANIMAL.animal_id = APPOINTMENT.animal_id WHERE
a_date = CURRENT_DATE ORDER BY time;

# Create dynamic predicate that gets only the appointments of that owner
CREATE VIEW FUTURE_APPOINTMENTS AS
SELECT ANIMAL.name as animal_name, type, notes, a_date, time, CLINIC.name,
CLINIC.address, CLINIC.city, CLINIC.state, CLINIC.zip_code,
CLINIC_PHONE_NUMBER.phone_number FROM ANIMAL_OWNER JOIN
ANIMAL ON ANIMAL_OWNER.ssn = ANIMAL.ssn JOIN APPOINTMENT ON
ANIMAL.animal_id = APPOINTMENT.animal_id JOIN CLINIC ON
APPOINTMENT.clinic_id = CLINIC.clinic_id JOIN CLINIC_PHONE_NUMBER ON
CLINIC.clinic_id = CLINIC_PHONE_NUMBER.clinic_id WHERE a_date >=
CURRENT_DATE ORDER BY a_date,time;

# Create dynamic predicate that gets only employees at that clinic
CREATE VIEW CLINIC_EMPLOYEES AS
SELECT first_name, last_name, title, phone_number, email, clinic_id FROM
EMPLOYEE JOIN JOB_TITLE ON EMPLOYEE.job_title_id =
JOB_TITLE.job_title_id JOIN EMPLOYEE_PHONE_NUMBER ON
EMPLOYEE_PHONE_NUMBER.ssn = EMPLOYEE.ssn ORDER BY first_name,
last_name;

# Verify with ANIMAL_OWNER.ssn must be protected
CREATE VIEW PERSCRIPTION_DETAILS AS
SELECT EMPLOYEE.first_name AS vet_first_name, EMPLOYEE.middle_name,
EMPLOYEE.last_name AS vet_last_name, ANIMAL.name as animal_name, p_date,
time, description, instructions, number_of_refills, DRUG.name, manufacturer, potency,
ANIMAL_OWNER.first_name, ANIMAL_OWNER.last_name FROM EMPLOYEE
JOIN PRESCRIPTION ON PRESCRIPTION.ssn = EMPLOYEE.ssn JOIN ANIMAL
ON PRESCRIPTION.animal_id = ANIMAL.animal_id JOIN ANIMAL_OWNER ON
ANIMAL_OWNER.ssn = ANIMAL.ssn JOIN PRESCRIPTION_DRUG ON
PRESCRIPTION_DRUG.prescription_id = PRESCRIPTION.prescription_id JOIN
```

```
DRUG ON PRESCRIPTION_DRUG.drug_id = DRUG.drug_id ORDER BY
DRUG.name;

#Show total revenue by employee
CREATE VIEW CLINIC_TOTALS_BY_EMPLOYEE AS
SELECT SUM(amount) AS total_revenue, CLINIC.name AS clinic_name,
EMPLOYEE.first_name, EMPLOYEE.last_name, EMPLOYEE.ssn FROM CLINIC
JOIN EMPLOYEE ON CLINIC.clinic_id = EMPLOYEE.clinic_id JOIN
EMPLOYEE_EXAMINATION ON EMPLOYEE_EXAMINATION.ssn =
EMPLOYEE.ssn JOIN EXAMINATION ON
EMPLOYEE_EXAMINATION.examination_id = EXAMINATION.examination_id
JOIN SALE SALE ON EXAMINATION.sale_id = SALE.sale_id JOIN
SALE_CHARGE ON SALE.sale_id = SALE_CHARGE.sale_id JOIN CHARGE ON
SALE_CHARGE.charge_id = CHARGE.charge_id GROUP BY
CLINIC.name,EMPLOYEE.first_name,EMPLOYEE.last_name, EMPLOYEE.ssn;

CREATE VIEW CLINIC_TOTALS AS
SELECT SUM(total_revenue) AS clinic_income, clinic_name, royalty,
ROUND(royalty*SUM(total_revenue),2) AS vet_pay,
ROUND((1-royalty)*SUM(total_revenue),2) AS profit FROM
CLINIC_TOTALS_BY_EMPLOYEE JOIN CONTRACT ON CONTRACT.ssn =
CLINIC_TOTALS_BY_EMPLOYEE.ssn GROUP BY clinic_name, royalty;

CREATE VIEW MOST_RECENT_EXAM AS
SELECT ANIMAL.animal_id, name as animal_name, breed, type, weight, heart_beat,
systolic_bp, diastolic_bp, animal_condition, observations, e_date FROM ANIMAL JOIN
EXAMINATION ON ANIMAL.animal_id = EXAMINATION.examination_id JOIN
EXAM_ANIMAL_ATTRIBUTES ON EXAMINATION.examination_id =
EXAM_ANIMAL_ATTRIBUTES.examination_id ORDER BY e_date;

CREATE VIEW PRESCRIPTION_VIEW AS
SELECT EMPLOYEE.first_name, EMPLOYEE.last_name, instructions,  DRUG.name,
CONCAT(potency, 'mg') as potency from EMPLOYEE JOIN prescription ON
EMPLOYEE.ssn = PRESCRIPTION.ssn JOIN PRESCRIPTION_DRUG on
PRESCRIPTION.prescription_id = PRESCRIPTION_DRUG.prescription_id join DRUG
on PRESCRIPTION_DRUG.drug_id = DRUG.drug_id;

# View for Vet appointments
CREATE VIEW VET_DAILY_APPOINTMENTS AS
```

```
SELECT ANIMAL.name, ANIMAL.type, first_name, last_name, time FROM ANIMAL
JOIN APPOINTMENT ON ANIMAL.animal_id = APPOINTMENT.animal_id JOIN
ANIMAL_OWNER ON ANIMAL.ssn = ANIMAL_OWNER.ssn WHERE
APPOINTMENT.a_date = CURRENT_DATE ORDER BY APPOINTMENT.time;

# View for examination records of animal at other clinics

# Make it so they can only see other clinics records.
CREATE VIEW OTHER_CLINICS_VISITED AS
SELECT DISTINCT CLINIC.name, CLINIC_PHONE_NUMBER.phone_number,
CLINIC.email, ANIMAL.animal_id from CLINIC JOIN CLINIC_PHONE_NUMBER
ON CLINIC.clinic_id = CLINIC_PHONE_NUMBER.clinic_id JOIN APPOINTMENT
ON APPOINTMENT.clinic_id = APPOINTMENT.clinic_id JOIN Animal ON
APPOINTMENT.animal_id = ANIMAL.animal_id;
```

Users

```
# Create Users
CREATE USER ADMIN IDENTIFIED BY password;
GRANT CREATE SESSION, DBA TO ADMIN;
CREATE USER ADDISON3745 IDENTIFIED BY password;
CREATE USER CONOR7463 IDENTIFIED BY password;
CREATE USER BRADLEY7348 IDENTIFIED BY password;
CREATE USER ANGELICA3943 IDENTIFIED BY password;
CREATE USER WYATT7483 IDENTIFIED BY password;
CREATE USER JASON6540 IDENTIFIED BY password;
CREATE USER JOHN8434 IDENTIFIED BY password;
```

Roles and Permissions

```
# Create the roles
CREATE ROLE ADMINISTRATOR;
GRANT CREATE SESSION,DBA TO ADMINISTRATOR;
GRANT SELECT ON CLINIC_TOTALS_BY_EMPLOYEE TO ADMINISTRATOR;
GRANT SELECT ON CLINIC_TOTALS_BY_EMPLOYEE TO ADMINISTRATOR;
CREATE ROLE VETERINARIAN;
GRANT CREATE SESSION TO VETERINARIAN;
GRANT SELECT ON CLINIC_TOTALS TO VETERINARIAN;
GRANT SELECT ON OTHER_CLINICS_VISITED TO VETERINARIAN;
GRANT SELECT ON VET_DAILY_APPOINTMENTS TO VETERINARIAN;
```

```
GRANT INSERT,DELETE,UPDATE ON EXAM_ANIMAL_ATTRIBUTES TO
VETERINARIAN;
GRANT INSERT,DELETE,UPDATE ON SALE TO VETERINARIAN;
GRANT INSERT,DELETE,UPDATE ON SALE_CHARGE TO VETERINARIAN;
GRANT SELECT ON CHARGE TO VETERINARIAN;
GRANT SELECT ON CONTRACT TO VETERINARIAN;
GRANT INSERT,UPDATE,DELETE ON PRESCRIPTION TO VETERINARIAN;
GRANT INSERT,UPDATE,DELETE ON PRESCRIPTION_DRUG TO
VETERINARIAN;
GRANT INSERT, UPDATE ON EXAMINATION TO VETERINARIAN;
GRANT SELECT ON MOST_RECENT_EXAM TO VETERINARIAN;

CREATE ROLE VET_TECH;
GRANT CREATE SESSION TO VET_TECH;
GRANT SELECT ON VET_DAILY_APPOINTMENTS TO VET_TECH;
GRANT INSERT, UPDATE ON EXAM_ANIMAL_ATTRIBUTES TO VET_TECH;
GRANT INSERT,UPDATE ON SALE TO VET_TECH;
GRANT INSERT,UPDATE ON SALE_CHARGE TO VET_TECH;
GRANT SELECT ON CHARGE TO VET_TECH;
GRANT INSERT, UPDATE ON EXAMINATION TO VET_TECH;
GRANT SELECT ON MOST_RECENT_EXAM TO VET_TECH;
CREATE ROLE RECEPTIONIST;
GRANT CREATE SESSION TO RECEPTIONIST;
GRANT SELECT ON DAILY_APPOINTMENTS TO RECEPTIONIST;
CREATE ROLE PET_OWNER;
GRANT CREATE SESSION TO PET_OWNER;
GRANT SELECT ON OWNER_ANIMAL_INFO TO PET_OWNER;
```

VPD's

```
# Create the context
CREATE CONTEXT ANIMAL_OWNER_SEC_CTX USING ADMIN.OWNER_SEC;

create or replace package OWNER_SEC is
procedure GET_OWNER_ID;
end OWNER_SEC;
/

create or replace package body OWNER_SEC is
procedure GET_OWNER_ID
```

```
is
OWNER_ID_VAR varchar2(4);
begin
select first_name
into OWNER_ID_VAR FROM OWNER_ANIMAL_INFO
where last_4 =
SYS_CONTEXT('USERENV', 'SESSION_USER');
dbms_session.set_context('ANIMAL_OWNER_SEC_CTX','FIRST_NAME','OWNER_I
D_VAR');
end GET_OWNER_ID;
END OWNER_SEC;
/

create or replace package OWNER_SEC
as
function OWNER_ID_SEC return
varchar2;
END OWNER_SEC;
/

create or replace package body OWNER_SEC AS
function OWNER_ID_SEC return varchar2
is
MY_PREDICATE varchar2(2000);
begin
MY_PREDICATE:='FIRST_NAME=SYS_CONTEXT("ANIMAL_OWNER_SEC_CT
X","FIRST_NAME")';
return MY_PREDICATE;
end OWNER_ID_SEC;
end OWNER_SEC;
/

execute dbms_rls.drop_policy('admin',
'OWNER_ANIMAL_INFO','OWNER_POLICY');
execute dbms_rls.add_policy('admin','OWNER_ANIMAL_INFO', 'OWNER_POLICY',
'admin','OWNER_SEC.OWNER_ID_SEC','SELECT',FALSE,TRUE);
```

Inserting Data

ANIMAL_OWNER Table

```
INSERT INTO ANIMAL_OWNER VALUES('326463746', 'addison', 'boyer',
'addison.boyer@newnetservices.net', '3000 South Higgins Avenue Apt. G26', 'Missoula',
'MT', '59801');

INSERT INTO ANIMAL_OWNER VALUES('432637463', 'conor', 'jones',
'cbjones@gmail.com', '3000 South Higgins Avenue Apt. G26', 'Missoula', 'MT', '59801');

INSERT INTO ANIMAL_OWNER VALUES('746374628', 'nicole', 'padia',
'nicolepadia@gmail.com', '728 Woody Street', 'Missoula', 'MT', '59801');




# ANIMAL_OWNER_PHONE_NUMBER Table
INSERT INTO ANIMAL_OWNER_PHONE_NUMBER VALUES('4062023673',
'326463746');

INSERT INTO ANIMAL_OWNER_PHONE_NUMBER VALUES('6692361085',
'432637463');

INSERT INTO ANIMAL_OWNER_PHONE_NUMBER VALUES('8084696411',
'746374628');

# CLINIC Table
INSERT INTO CLINIC VALUES(1, 'Mountain View Animal Clinic',
'mountainviewac@gmail.com', 'Missoula', 'MT', '59801', '123 Power Street');

INSERT INTO CLINIC VALUES(2, 'River View Animal Clinic',
'riverviewac@gmail.com', 'Missoula', 'MT', '59803', '406 Bridgeway Drive');

# CLINIC_PHONE_NUMBER Table
INSERT INTO CLINIC_PHONE_NUMBER VALUES('4064562673', 1);
INSERT INTO CLINIC_PHONE_NUMBER VALUES ('4065465857',2);

# ANIMAL table

INSERT INTO ANIMAL VALUES(1, 'Lacy', 'domestic short hair', 'cat',
TO_DATE('2018/08/31', 'yyyy/mm/dd'), NULL, '326463746');
```

```
INSERT INTO ANIMAL VALUES(2, 'Gigi', 'rat terrier', 'dog', TO_DATE('2009/03/03',
'yyyy/mm/dd'), NULL, '432637463');

INSERT INTO ANIMAL VALUES(3, 'Spooky', 'domestic short hair', 'cat',
TO_DATE('2003/04/06', 'yyyy/mm/dd'), 'We love spooky', '746374628');

# Drug Table
INSERT INTO DRUG VALUES(1, 'Abbott Laboratories', 'buprenorphine', 2);
INSERT INTO DRUG VALUES(2, 'Laser Pharmaceuticals, LLC', 'carprofen ', 3);
INSERT INTO DRUG VALUES(3, 'Meda U.S.', 'moxifloxacin', 5);

INSERT INTO DRUG VALUES(4, 'Abbott Laboratories', 'neomycin', 3);

INSERT INTO DRUG VALUES(5, 'Meda U.S.', 'phenobarbital', 5);

# JOB_TITLE Table
INSERT INTO JOB_TITLE VALUES(1, 'Veterinarian', NULL, NULL);
INSERT INTO JOB_TITLE VALUES(2, 'Veterinary Tech', NULL, 20.00);
INSERT INTO JOB_TITLE VALUES(3, 'Receptionist', NULL, 13.00);

# EMPLOYEE Table
INSERT INTO EMPLOYEE VALUES('437567348', 'bradley', 'lewis', 'boyer', '800 South
3rd Street', 'Hamilton', 'MT', '59840', 'bboyer@live.com', TO_DATE('1964/04/14',
'yyyy/mm/dd'), 1, 1);

INSERT INTO EMPLOYEE VALUES('746637483', 'wyatt', 'bradley', 'boyer', '756 Oak
Way', 'Missoula', 'MT', '59801', 'wboyer@gmail.com', TO_DATE('1994/02/03',
'yyyy/mm/dd'), 2, 1);

INSERT INTO EMPLOYEE VALUES('453473943', 'angelica', 'nicole', 'boyer', '574
Skyway Drive', 'Missoula', 'MT', '59801', 'angieboyer@hotmail.com',
TO_DATE('1992/08/23', 'yyyy/mm/dd'), 3, 1);

# Veterinarian at another clinic
INSERT INTO EMPLOYEE VALUES('374732382', 'chad', 'matthew', 'beley', '211
Springfield Drive', 'Helena', 'MT', '59602', 'cbeley@gmail.com',
TO_DATE('1995/07/27','yyyy/mm/dd'),1, 2);
```

```
INSERT INTO EMPLOYEE VALUES('657376540', 'jason', 'ray', 'corne', '409 Pine Street', 'Missoula', 'MT', '59801', 'jason.corne@riverviewac.com', TO_DATE('1995/07/6', 'yyyy/mm/dd'),2, 2);

INSERT INTO EMPLOYEE VALUES('543758434', 'john', 'carl', 'rusoff', '301 Mount Avenue', 'Missoula', 'MT', '59801', 'john.rusoff@riverviewac.com', TO_DATE('1995/09/27','yyyy/mm/dd'),3, 2);

# EMPLOYEE_PHONE_NUMBER Table
INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4063632999', '437567348');
INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4062020106', '746637483');
INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4062021190', '453473943');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4064751023', '374732382');
INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4064385533', '657376540');
INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4064382872', '543758434');



# PRESCRIPTION Table
INSERT INTO PRESCRIPTION VALUES(1, TO_DATE('1995/09/27','yyyy/mm/dd'), TO_DATE('1995/09/27','yyyy/mm/dd'), 'neomycin: 20 capsules, buprenorphine: 10 capsules', 'neomycin: take twice daily for two weeks mix with food, buprenorphine: take once daily in the morning with plenty of water', 1, 1, '437567348');

# PRESCRIPTION_DRUG Table
INSERT INTO PRESCRIPTION_DRUG VALUES(4,1);
INSERT INTO PRESCRIPTION_DRUG VALUES(1,1);

# CONTRACT Table
INSERT INTO CONTRACT VALUES(1, 'Mountain View Animal Clinic Vet Contract', TO_DATE('2019/04/21','yyyy/mm/dd'), TO_DATE('2020/04/21','yyyy/mm/dd'), 0.65, '437567348');
```

```
INSERT INTO CONTRACT VALUES(2, 'River View Animal Clinic Vet Contract',
TO_DATE('2019/04/21','yyyy/mm/dd'), TO_DATE('2020/04/21','yyyy/mm/dd'), 0.70,
'374732382');

# CHARGE Table
INSERT INTO CHARGE VALUES (1,100.00,'neuter/spay');

INSERT INTO CHARGE VALUES(2,40.00,'rabies vaccination');

INSERT INTO CHARGE VALUES(3,50.00,'general checkup');

INSERT INTO CHARGE VALUES(4,30.00,'follow up checkup');

INSERT INTO CHARGE VALUES(5, 5.00, 'prescription fee');

INSERT INTO CHARGE VALUES(6, 10.00, 'appointment cancel');

INSERT INTO CHARGE VALUES(7, 60.00, 'microchip fee');

# EMPLOYEE_EXAMINATION Table
INSERT INTO EMPLOYEE_EXAMINATION VALUES('437567348', 1);
INSERT INTO EMPLOYEE_EXAMINATION VALUES('374732382', 1);
INSERT INTO EMPLOYEE_EXAMINATION VALUES('374732382', 2);

INSERT INTO EMPLOYEE_EXAMINATION VALUES('374732382', 3);

#SALE Table
INSERT INTO SALE VALUES(1, TO_DATE('2019/04/25','yyyy/mm/dd'), NULL);
INSERT INTO SALE VALUES(2, TO_DATE('2019/04/25','yyyy/mm/dd'), NULL);

INSERT INTO SALE VALUES(3, TO_DATE('2019/04/25','yyyy/mm/dd'), NULL);

INSERT INTO SALE_CHARGE VALUES(1,3);
INSERT INTO SALE_CHARGE VALUES(1,2);

INSERT INTO SALE_CHARGE VALUES(2,4);

INSERT INTO SALE_CHARGE VALUES(3,3);
```

```
# EXAMINATION Table
INSERT INTO EXAMINATION VALUES(1, TO_DATE('18:18:30', 'hh24:mi:ss'),
TO_DATE('18:40:30', 'hh24:mi:ss'), TO_DATE('2019/04/25','yyyy/mm/dd'), 1, 1);

INSERT INTO EXAMINATION VALUES(2, TO_DATE('20:00:00', 'hh24:mi:ss'),
TO_DATE('20:22:00', 'hh24:mi:ss'), TO_DATE('2019/04/25','yyyy/mm/dd'), 2, 2);


INSERT INTO EXAMINATION VALUES(3, TO_DATE('12:00:00', 'hh24:mi:ss'),
TO_DATE('12:30:00', 'hh24:mi:ss'), TO_DATE('2019/04/25', 'yyyy/mm/dd'), 1, 3);

# EXAM_ANIMAL_ATTRIBUTES Table
INSERT INTO EXAM_ANIMAL_ATTRIBUTES VALUES(1, 9.65, 64, 40, 120,
'healthy', 'everything looks great', 1);

INSERT INTO EXAM_ANIMAL_ATTRIBUTES VALUES(2, 8.50, 80, 36, 130, 'sick',
'recovering slower than expected from treatment', 2);

INSERT INTO EXAM_ANIMAL_ATTRIBUTES VALUES(3, 10.0, 68, 46, 125,
'healthy', 'very healthy young kitten', 1);

# APPOINTMENT TABLE
INSERT INTO APPOINTMENT VALUES(1,TO_DATE('2019/04/26','yyyy/mm/dd'),
TO_DATE('10:30:00', 'hh24/mi/ss'), NULL, 1, 1);

INSERT INTO APPOINTMENT VALUES(2, TO_DATE('2019/04/26', 'yyyy/mm/dd'),
TO_DATE('11:30:00', 'hh24/mi/ss'), NULL, 2, 1);

INSERT INTO APPOINTMENT VALUES(3, TO_DATE('2019/04/26', 'yyyy/mm/dd'),
TO_DATE('12:30:00', 'hh24/mi/ss'), NULL, 1, 2);

INSERT INTO APPOINTMENT VALUES(4, TO_DATE('2019/04/26', 'yyyy/mm/dd'),
TO_DATE('13:30:00', 'hh24/mi/ss'), NULL, 2, 1);

INSERT INTO APPOINTMENT VALUES(5, TO_DATE('2019/04/26', 'yyyy/mm/dd'),
TO_DATE('13:30:00', 'hh24/mi/ss'), NULL, 3,1);
```

VI. Security Plan

Purpose

The purpose of this security plan is to ensure the confidentiality, availability, and integrity of data regarding the clients, employees, and administrative staff of the Boyer Animal Clinic. The plan will formally enumerate the policies, and procedures that ultimately support this objective. The security policies in this document act as a guide for both the management and implementation of information technology solutions within the Boyer Animal Clinic. The detailed security procedures serve to ensure that the security policies are being followed, while minimizing risk for all stakeholders. This plan also ensures the legal obligations of Boyer Animal Clinic in regards to patient information is fulfilled.

Scope

This security plan applies to the entities within all clinics operating under the Boyer Animal Clinic brand name. These include but are not limited to administrative staff, veterinarians, veterinary technicians, assistants, receptionists, interns, volunteers, and any person with access to Boyer Animal Clinics information technology assets. In other words any person with access to sensitive client or employee data.

End-user Security Policies and Procedures

1. Email and Internet
    a. Policy: Email accounts are provided by Boyer Animal Clinic and created by administrators must be used for business related purposes only. Email accounts may not be used by any employee under any circumstances for personal reasons, and especially may not be used to sign up for any service that is not essential to the completion of day to day tasks. Employee emails will be stored in the database, and only veterinarians should be able to contact employees at other clinics under the Boyer Animal Clinic Brand name. All other employees are prohibited from contacting employees in this categorization by phone, or email for any reason. Any information obtained from the database must not be transmitted through any email without the prior written consent of Boyer Animal Clinic. Any violation of this policy may constitute disciplinary action outlined in the procedure below (b).
    b. Procedure: Every employee of the Boyer Animal Clinic will be given a company email for use in their daily job duties. Each email will be formatted as follows first_name.last_name@boyeranimalclinic.com. If two employees share both the same first_name and last_name respectively, a number will be appended to the end of the email in the order that the account was created. Administrators only have access to create accounts, and will also create database user accounts for

each employee as well. Database user accounts should be formatted as follows, first_nameXXXX where (XXXX) corresponds to last 4 digits of that employees social security number. Emails should be used by employees in the following ways. Office receptionists may use email to send appointment reminders to clients who've booked appointments with their clinic only. They may also send emails to veterinary technicians, and veterinarians to inform them that clients have arrived and checked in at the front desk. Owner first name, pet name, pet type, and time of appointment are all acceptable to be included in the body of these types of emails. Office receptionists must also use email to send clients an electronic receipt if requested at the time of service. Veterinary technicians, and veterinarians may use email to inform office receptionists of an estimated completion time of an ongoing examination. Veterinarians only should use email to contact administration, and should never include details of an ongoing contract in the body of an email. If any employee is found to be in violation of following the procedures outlined above, they may be subject to have their email account disabled for a specified amount of time TBD by Boyer Animal Clinic.

2. Passwords
    a. Policy: Both database and user accounts will be given an initial password that is required to be changed before logging into the system. In no circumstance shall any employee share their login information with another. In no circumstance should login credentials be stored on any machine in plain text, or be sent in the body of any email. Such behavior is deemed irresponsible, and is grounds for immediate termination and void any contract made with Boyer Animal Clinic. Passwords are required to be at least 7 characters, and should include a combination of upper/lower case letters, numbers, and symbols. The following procedures below outline the guidelines for behavior regarding passwords.
    b. Procedure: Administrators are the only employees able to modify user database accounts. In the situation that a password reset is necessary, the veterinarian should be instructed to contact an administrator and request a password reset. Once processed the administrator will remotely access a machine at the clinic, and supply his session to allow the user to reset their password. This password reset should be monitored closely by the administrator, in no other circumstances should an administrator give any employee access to their database session. Passwords should be checked for password strength on the client side before being approved.
3. Acceptable use of computer systems and devices
    a. Policy: Computers and devices of Boyer Animal Clinic should be used only for the essential job duties outlined in the employee handbook. Any other use of company information technology property is prohibited. The following procedure

outlines the guidelines for employee use of company information technology equipment.

b. Procedure: When an employee shows up to work, they may only use the computer to connect to the database, and receive information that they need to perform their job duties. They may use email in the ways outlined above, and may also utilize certain IT equipment in the case of emergency. Every other use of these assets is prohibited by Boyer Animal Clinic.

Business and Operations Security Policies and Procedures

1. Client Check-in / Verification
    a. Policy: Only office receptionists should have access to client contact information. In no circumstances should a receptionist share any of the following information with any other client, person, or employee; Pet owner Social Security Number, Phone number, address, date of birth, or email. This information is considered confidential, and should be protected by those with access to it.
    b. Procedure: When a client arrives they will be asked to supply the last 4 digits of their social security number, their pets name, as well as a form of identification in order to verify their identity. If the receptionist is able to verify the client's identity, they should send an email to the veterinarian on duty, and cc the veterinary technician. The email should include the first name of the client, the pets name, and the time of the appointment.
2. Receipt of check-in email / preparation for examination
    a. Policy: Veterinarians and Veterinary technicians should be able to access animal records, past examination records, and previous prescriptions that they have given at their respective animal clinics. In no circumstances shall veterinarians disclose this information to any other person or entity without the prior verbal or written consent of the animal owner. This information is intended to acquaint the veterinarian with the animal before examining them. The veterinarian shall not see the animal until this information has been reviewed.
    b. Procedure: When a Veterinarian receives an email from a receptionist informing them that a client has arrived and been verified the following steps should be taken. 1. The veterinarian should first complete any ongoing examination. If one still persists the veterinarian or veterinary tech should inform the receptionist of the estimated amount of time until completion. 2. If there is no ongoing examination, then the Veterinarian should verify the appointment, and pull the animal records, previous examination records, and prescriptions of the animal referenced in the body of the email. 3. The veterinarian should review this information, and when the review is complete send an email to the receptionist letting them know he or she is ready to begin the examination. In no circumstances shall a veterinarian or veterinary technician begin an examination

without first reviewing previous records. If no records exist, the veterinarian should see if the animal has been seen by any other clinic under the Boyer Animal Clinic brand name. If so they should follow the procedures outlined in the next policy. Otherwise the veterinarian should follow the examination procedure and related policy.

3. Checking for animal records at other clinics
    a. Policy: If during the check-in procedure the veterinary staff finds that the animal to be seen has no past examination records and/or prescriptions at that clinic, they should be able to see if that animal has been seen by any other clinic operating under the Boyer Animal Clinic brand name. If an office receptionist receives a call from a client requesting their animal records to be released, they should be able to verify the client, and animal they are requesting records for.
    b. Procedure: When a veterinarian finds that an animal they are about to examine has been seen at another clinic operating under the Boyer Animal Clinic brand name they should carry out the following steps. Send an email to the receptionist informing them of the clinic(s) that this animal has been seen at. The receptionist will then ask the client if they wish to disclose those examinations to their clinic. If so the client will need to call that clinic, and make a request for the release of that information. If not, the veterinarian should proceed to see the clients animal. Otherwise they should wait for the information to be disclosed by the other animal clinic via email. If an office receptionist receives a call from a client requesting disclosure of records from their clinic, that receptionist should verify the client in the same fashion as check-in then notify the veterinary staff of the needed disclosure of information. The veterinary staff should then send all needed information to the other veterinarian directly by email. Once this information is received and reviewed, the receptionist should notify the client that they are ready to see them.
4. Performing the examination
    a. Policy: veterinarians and veterinary technicians should be able to access examination records, and create new examination records. Receptionists must only view generic animal information. In no circumstance should any veterinarian or veterinary technician share examination information with any other person other than the pet owner, and/or other members of the veterinary staff.
    b. Procedure: Upon starting an examination the veterinarian should record attribute information including heart_beat, blood pressure, condition, and additional observations. A sale record should be created and examination start time should be documented. Depending on the type of appointment a base charge will be added to the sale, and all subsequent sales must be approved by the animal owner.

If the veterinarian feels they should perform a procedure resulting in an extra charge, they should email the receptionist outlining the need for approval of the added charge. Upon approval the charge should be added to the sale.

5. Prescribing medication
    a. Policy: Only a licensed professional should be able to write prescription as outlined by state law. Therefore in compliance with this law, only veterinarians are allowed to create new prescription records, and/or edit and delete them. In no circumstances should prescriptions be written by someone without a license to prescribe medication. Client information is confidential, and prescriptions should not be shared with anyone other than the owner of the animal.
    b. Procedure: If a veterinarian decides that an animal needs to be prescribed medication they should note the reasons for the prescription in the examination observations. Then they should proceed to write the prescription. Prescriptions should be filled out by the veterinarian only. Once a prescription is written it will be handed to the client. The veterinarian will then add the charges associated with the examination. Then the receptionist will print them a receipt outlining an enumeration of charges.
6. Viewing employee, and clinic revenue
    a. Policy: Only administrators should view employee, and clinic revenue. They should also be able to see current royalty contract amounts as well as profits they are making at those clinics. Profit numbers are not to be shared with anyone but administrators of the clinic.
    b. Procedure: Administrators should be able to view this information in the context of a contract with a veterinarian, they should be able to see their pay, as well as the profit being made by the company.

Security Model

The Boyer Animal Clinic software system utilizes a secure state transition security model. After each operation, the state of the system is secure. The system utilized VPD to ensure that users can only query data pertinent to themselves and/or the clinic they are operating under. This ensures the flow of data between clinics is limited, thus minimizing the risk of data integrity loss. Social security numbers are masked by the last 4 digits for verification purposes, and views are utilized to abstract the users from the actual data tables.

VII. Verification of Database Security Implementation

[oracle@localhost oracle]$ sqlplus sys as sysdba

Addison Boyer (Boyer Animal Clinic)

```
SQL*Plus: Release 18.0.0.0.0 - Production on Fri Apr 26 17:43:22 2019
Version 18.3.0.0.0

Copyright (c) 1982, 2018, Oracle.  All rights reserved.

Enter password:

Connected to:
Oracle Database 18c Enterprise Edition Release 18.0.0.0.0 - Production
Version 18.3.0.0.0

SQL> CREATE USER ADMIN IDENTIFIED BY password;

User created.

SQL> GRANT CREATE SESSION, DBA TO ADMIN;

Grant succeeded.

SQL> conn ADMIN;
Enter password:
Connected.
SQL> CREATE TABLE JOB_TITLE(job_title_id INT PRIMARY KEY NOT NULL, title
VARCHAR(20), description VARCHAR(255), rate_of_pay FLOAT(2));

Table created.

SQL> CREATE TABLE CLINIC(clinic_id INT PRIMARY KEY NOT NULL, name
VARCHAR(40), email VARCHAR(320), city VARCHAR(20), state CHAR(2), zip_code
VARCHAR(10), address VARCHAR(100));

Table created.

SQL> CREATE TABLE CLINIC_PHONE_NUMBER(phone_number VARCHAR(20) NOT
NULL, clinic_id INT NOT NULL, FOREIGN KEY (clinic_id) REFERENCES
CLINIC(clinic_id), PRIMARY KEY(phone_number, clinic_id));

Table created.
```

SQL> CREATE TABLE EMPLOYEE(ssn VARCHAR(20) PRIMARY KEY NOT NULL, first_name VARCHAR(20), middle_name VARCHAR(20), last_name VARCHAR(20), address VARCHAR(40), city VARCHAR(20), state CHAR(2), zip_code VARCHAR(10), email VARCHAR(320), date_of_birth DATE, job_title_id INT NOT NULL, clinic_id INT NOT NULL, FOREIGN KEY (job_title_id) REFERENCES JOB_TITLE(job_title_id), FOREIGN KEY (clinic_id) REFERENCES CLINIC(clinic_id));

Table created.

SQL> CREATE TABLE EMPLOYEE_PHONE_NUMBER(phone_number VARCHAR(20) NOT NULL, ssn VARCHAR(20) NOT NULL, FOREIGN KEY (ssn) REFERENCES EMPLOYEE (ssn), PRIMARY KEY(phone_number, ssn));

Table created.

SQL> CREATE TABLE DRUG(drug_id INT PRIMARY KEY NOT NULL, manufacturer VARCHAR(40), name VARCHAR(20), potency INT);

Table created.

SQL> CREATE TABLE ANIMAL_OWNER(ssn VARCHAR(20) PRIMARY KEY NOT NULL, first_name VARCHAR(20), last_name VARCHAR(20), email VARCHAR(320), address VARCHAR(40), city VARCHAR(20), state CHAR(2), zip_code VARCHAR(10));

Table created.

SQL> CREATE TABLE ANIMAL_OWNER_PHONE_NUMBER(phone_number VARCHAR(20) NOT NULL, ssn VARCHAR(20) NOT NULL, FOREIGN KEY (ssn) REFERENCES ANIMAL_OWNER (ssn), PRIMARY KEY(phone_number, ssn));

Table created.

SQL> CREATE TABLE ANIMAL(animal_id INT PRIMARY KEY NOT NULL, name VARCHAR(20), breed VARCHAR(20), type VARCHAR(10), date_of_birth DATE, notes VARCHAR(300), ssn VARCHAR(20) NOT NULL, FOREIGN KEY (ssn) REFERENCES ANIMAL_OWNER(ssn));

Table created.

SQL> CREATE TABLE PRESCRIPTION(prescription_id INT PRIMARY KEY NOT NULL, p_date DATE, time DATE, description VARCHAR(300), instructions VARCHAR(300), number_of_refills INT, animal_id INT NOT NULL, ssn VARCHAR(20) NOT NULL, FOREIGN KEY (animal_id) REFERENCES ANIMAL (animal_id), FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn));

Table created.

SQL> CREATE TABLE PRESCRIPTION_DRUG(drug_id INT NOT NULL, prescription_id INT NOT NULL, FOREIGN KEY (prescription_id) REFERENCES PRESCRIPTION(prescription_id), FOREIGN KEY (drug_id) REFERENCES DRUG(drug_id), PRIMARY KEY(drug_id, prescription_id));

Table created.

SQL> CREATE TABLE APPOINTMENT(appointment_id INT PRIMARY KEY NOT NULL, a_date DATE, time DATE, description VARCHAR(300), animal_id INT NOT NULL, clinic_id INT NOT NULL, FOREIGN KEY (animal_id) REFERENCES ANIMAL(animal_id), FOREIGN KEY (clinic_id) REFERENCES CLINIC(clinic_id));

Table created.

SQL> CREATE TABLE SALE(sale_id INT PRIMARY KEY NOT NULL, s_date DATE, description VARCHAR(300));

Table created.

SQL> CREATE TABLE EXAMINATION(examination_id INT PRIMARY KEY NOT NULL, start_time DATE, end_time DATE, e_date DATE, animal_id INT NOT NULL, sale_id INT NOT NULL, FOREIGN KEY (animal_id) REFERENCES ANIMAL(animal_id), FOREIGN KEY (sale_id) REFERENCES SALE(sale_id));

Table created.

SQL> CREATE TABLE EXAM_ANIMAL_ATTRIBUTES(exam_attribute_id INT PRIMARY KEY NOT NULL, weight FLOAT(2), heart_beat INT, systolic_bp INT, diastolic_bp INT, animal_condition VARCHAR(20), observations VARCHAR(300), examination_id INT NOT NULL, FOREIGN KEY (examination_id) REFERENCES EXAMINATION(examination_id));

Table created.

SQL> CREATE TABLE EMPLOYEE_EXAMINATION(ssn VARCHAR(20) NOT NULL, examination_id INT NOT NULL, FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn), FOREIGN KEY (examination_id) REFERENCES EXAMINATION(examination_id),PRIMARY KEY (ssn, examination_id));

Table created.

SQL> CREATE TABLE CONTRACT(contract_id INT PRIMARY KEY NOT NULL, title VARCHAR(100), start_date DATE, end_date DATE, royalty FLOAT(2), ssn VARCHAR(20) NOT NULL, FOREIGN KEY (ssn) REFERENCES EMPLOYEE(ssn));

Table created.

SQL> CREATE TABLE CHARGE(charge_id INT PRIMARY KEY NOT NULL, amount FLOAT(2), type VARCHAR(20));

Table created.

SQL> CREATE TABLE SALE_CHARGE(sale_id INT NOT NULL, charge_id INT NOT NULL, FOREIGN KEY (sale_id) REFERENCES SALE(sale_id), FOREIGN KEY (charge_id) REFERENCES CHARGE(charge_id), PRIMARY KEY(sale_id, charge_id));

Table created.

SQL> INSERT INTO ANIMAL_OWNER VALUES('326463746', 'addison', 'boyer', 'addison.boyer@newnetservices.net', '3000 South Higgins Avenue Apt. G26', 'Missoula', 'MT', '59801');

1 row created.

SQL> INSERT INTO ANIMAL_OWNER VALUES('432637463', 'conor', 'jones', 'cbjones@gmail.com', '3000 South Higgins Avenue Apt. G26', 'Missoula', 'MT', '59801');

1 row created.

```
SQL> INSERT INTO ANIMAL_OWNER VALUES('746374628', 'nicole', 'padia',
'nicolepadia@gmail.com', '728 Woody Street', 'Missoula', 'MT', '59801');

1 row created.

SQL> INSERT INTO ANIMAL_OWNER_PHONE_NUMBER VALUES('4062023673',
'326463746');

1 row created.

SQL> INSERT INTO ANIMAL_OWNER_PHONE_NUMBER VALUES('6692361085',
'432637463');

1 row created.

SQL> INSERT INTO ANIMAL_OWNER_PHONE_NUMBER VALUES('8084696411',
'746374628');

1 row created.

SQL> INSERT INTO CLINIC VALUES(1, 'Mountain View Animal Clinic',
'mountainviewac@gmail.com', 'Missoula', 'MT', '59801', '123 Power Street');

1 row created.

SQL> INSERT INTO CLINIC VALUES(2, 'River View Animal Clinic',
'riverviewac@gmail.com', 'Missoula', 'MT', '59803', '406 Bridgeway Drive');

1 row created.

SQL> INSERT INTO CLINIC_PHONE_NUMBER VALUES('4064562673', 1);

1 row created.

SQL> INSERT INTO CLINIC_PHONE_NUMBER VALUES ('4065465857',2);

1 row created.
```

```
SQL> INSERT INTO ANIMAL VALUES(1, 'Lacy', 'domestic short hair', 'cat',
TO_DATE('2018/08/31', 'yyyy/mm/dd'), NULL, '326463746');

1 row created.

SQL> INSERT INTO ANIMAL VALUES(2, 'Gigi', 'rat terrier', 'dog', TO_DATE('2009/03/03',
'yyyy/mm/dd'), NULL, '432637463');

1 row created.

SQL> INSERT INTO ANIMAL VALUES(3, 'Spooky', 'domestic short hair', 'cat',
TO_DATE('2003/04/06', 'yyyy/mm/dd'), 'We love spooky', '746374628');

1 row created.

SQL> INSERT INTO DRUG VALUES(1, 'Abbott Laboratories', 'buprenorphine', 2);

1 row created.

SQL> INSERT INTO DRUG VALUES(2, 'Laser Pharmaceuticals, LLC', 'carprofen ', 3);

1 row created.

SQL> INSERT INTO DRUG VALUES(3, 'Meda U.S.', 'moxifloxacin', 5);

1 row created.

SQL> INSERT INTO DRUG VALUES(4, 'Abbott Laboratories', 'neomycin', 3);

1 row created.

SQL> INSERT INTO DRUG VALUES(5, 'Meda U.S.', 'phenobarbital', 5);

1 row created.

SQL> INSERT INTO JOB_TITLE VALUES(1, 'Veterinarian', NULL, NULL);

1 row created.
```

```
SQL> INSERT INTO JOB_TITLE VALUES(2, 'Veterinary Tech', NULL, 20.00);

1 row created.

SQL> INSERT INTO JOB_TITLE VALUES(3, 'Receptionist', NULL, 13.00);

1 row created.

SQL> INSERT INTO EMPLOYEE VALUES('437567348', 'bradley', 'lewis', 'boyer', '800 South
3rd Street', 'Hamilton', 'MT', '59840', 'bboyer@live.com', TO_DATE('1964/04/14',
'yyyy/mm/dd'), 1, 1);

1 row created.

SQL> INSERT INTO EMPLOYEE VALUES('746637483', 'wyatt', 'bradley', 'boyer', '756 Oak
Way', 'Missoula', 'MT', '59801', 'wboyer@gmail.com', TO_DATE('1994/02/03', 'yyyy/mm/dd'),
2, 1);

1 row created.

SQL> INSERT INTO EMPLOYEE VALUES('453473943', 'angelica', 'nicole', 'boyer', '574
Skyway Drive', 'Missoula', 'MT', '59801', 'angieboyer@hotmail.com', TO_DATE('1992/08/23',
'yyyy/mm/dd'), 3, 1);

1 row created.

SQL> INSERT INTO EMPLOYEE VALUES('374732382', 'chad', 'matthew', 'beley', '211
Springfield Drive', 'Helena', 'MT', '59602', 'cbeley@gmail.com',
TO_DATE('1995/07/27','yyyy/mm/dd'),1, 2);

1 row created.

SQL> INSERT INTO EMPLOYEE VALUES('657376540', 'jason', 'ray', 'corne', '409 Pine
Street', 'Missoula', 'MT', '59801', 'jason.corne@riverviewac.com', TO_DATE('1995/07/6',
'yyyy/mm/dd'),2, 2);

1 row created.
```

```
SQL> INSERT INTO EMPLOYEE VALUES('543758434', 'john', 'carl', 'rusoff', '301 Mount
Avenue', 'Missoula', 'MT', '59801', 'john.rusoff@riverviewac.com',
TO_DATE('1995/09/27','yyyy/mm/dd'),3, 2);

1 row created.

SQL> INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4063632999', '437567348');

1 row created.

SQL> INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4062020106', '746637483');

1 row created.

SQL> INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4062021190', '453473943');

1 row created.

SQL> INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4064751023', '374732382');

1 row created.

SQL> INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4064385533', '657376540');

1 row created.

SQL> INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES('4064382872', '543758434');

1 row created.

SQL> INSERT INTO PRESCRIPTION VALUES(1, TO_DATE('1995/09/27','yyyy/mm/dd'),
TO_DATE('1995/09/27','yyyy/mm/dd'), 'neomycin: 20 capsules, buprenorphine: 10 capsules',
'neomycin: take twice daily for two weeks mix with food, buprenorphine: take once daily in the
morning with plenty of water', 1, 1, '437567348');

1 row created.

SQL> INSERT INTO PRESCRIPTION_DRUG VALUES(4,1);
```

1 row created.

SQL> INSERT INTO PRESCRIPTION_DRUG VALUES(1,1);

1 row created.

SQL> INSERT INTO CONTRACT VALUES(1, 'Mountain View Animal Clinic Vet Contract', TO_DATE('2019/04/21','yyyy/mm/dd'), TO_DATE('2020/04/21','yyyy/mm/dd'), 0.65, '437567348');

1 row created.

SQL> INSERT INTO CONTRACT VALUES(2, 'River View Animal Clinic Vet Contract', TO_DATE('2019/04/21','yyyy/mm/dd'), TO_DATE('2020/04/21','yyyy/mm/dd'), 0.70, '374732382');

1 row created.

SQL> INSERT INTO CHARGE VALUES (1,100.00,'neuter/spay');

1 row created.

SQL> INSERT INTO CHARGE VALUES(2,40.00,'rabies vaccination');

1 row created.

SQL> INSERT INTO CHARGE VALUES(3,50.00,'general checkup');

1 row created.

SQL> INSERT INTO CHARGE VALUES(4,30.00,'follow up checkup');

1 row created.

SQL> INSERT INTO CHARGE VALUES(5, 5.00, 'prescription fee');

1 row created.

SQL> INSERT INTO CHARGE VALUES(6, 10.00, 'appointment cancel');

```
1 row created.

SQL> INSERT INTO CHARGE VALUES(7, 60.00, 'microchip fee');

1 row created.

SQL> INSERT INTO SALE VALUES(1, TO_DATE('2019/04/25','yyyy/mm/dd'), NULL);
INSERT INTO SALE VALUES(2, CURRENT_DATE(), NULL);
1 row created.

SQL> INSERT INTO SALE VALUES(2, TO_DATE('2019/04/25','yyyy/mm/dd'), NULL);

1 row created.

SQL> INSERT INTO SALE VALUES(3, TO_DATE('2019/04/25','yyyy/mm/dd'), NULL);

1 row created.

SQL> INSERT INTO SALE_CHARGE VALUES(1,3);

1 row created.

SQL> INSERT INTO SALE_CHARGE VALUES(1,2);

1 row created.

SQL> INSERT INTO SALE_CHARGE VALUES(2,4);

1 row created.

SQL> INSERT INTO SALE_CHARGE VALUES(3,3);

1 row created.

SQL> INSERT INTO EXAMINATION VALUES(1, TO_DATE('18:18:30', 'hh24:mi:ss'),
TO_DATE('18:40:30', 'hh24:mi:ss'), TO_DATE('2019/04/25','yyyy/mm/dd'), 1, 1);

1 row created.
```

```
SQL> INSERT INTO EXAMINATION VALUES(2, TO_DATE('20:00:00', 'hh24:mi:ss'),
TO_DATE('20:22:00', 'hh24:mi:ss'), TO_DATE('2019/04/25','yyyy/mm/dd'), 2, 2);

1 row created.

SQL> INSERT INTO EXAMINATION VALUES(3, TO_DATE('12:00:00', 'hh24:mi:ss'),
TO_DATE('12:30:00', 'hh24:mi:ss'), TO_DATE('2019/04/25', 'yyyy/mm/dd'), 1, 3);

1 row created.

SQL> INSERT INTO EXAM_ANIMAL_ATTRIBUTES VALUES(1, 9.65, 64, 40, 120,
'healthy', 'everything looks great', 1);

1 row created.

SQL> INSERT INTO EXAM_ANIMAL_ATTRIBUTES VALUES(2, 8.50, 80, 36, 130, 'sick',
'recovering slower than expected from treatment', 2);

1 row created.

SQL> INSERT INTO EXAM_ANIMAL_ATTRIBUTES VALUES(3, 10.0, 68, 46, 125,
'healthy', 'very healthy young kitten', 1);

1 row created.

SQL> INSERT INTO EMPLOYEE_EXAMINATION VALUES('437567348', 1);

1 row created.

SQL> INSERT INTO EMPLOYEE_EXAMINATION VALUES('374732382', 1);

1 row created.

SQL> INSERT INTO EMPLOYEE_EXAMINATION VALUES('374732382', 2);

1 row created.

SQL> INSERT INTO EMPLOYEE_EXAMINATION VALUES('374732382', 3);
```

1 row created.

```
SQL> CREATE VIEW OWNER_ANIMAL_INFO AS
SELECT SUBSTR(ANIMAL_OWNER.ssn, LENGTH(ANIMAL_OWNER.SSN)-4, 4) as
last_4, first_name, last_name, phone_number, email, name as pet_name, type, notes from
ANIMAL_OWNER JOIN ANIMAL ON ANIMAL_OWNER.ssn = ANIMAL.ssn JOIN
ANIMAL_OWNER_PHONE_NUMBER ON ANIMAL_OWNER.ssn =
ANIMAL_OWNER_PHONE_NUMBER.ssn;  2
```

View created.

```
SQL> CREATE VIEW DAILY_APPOINTMENTS AS
SELECT SUBSTR(ANIMAL_OWNER.ssn, LENGTH(ANIMAL_OWNER.SSN)-4, 4) as
verify_ssn, first_name, last_name, phone_number, email, name as animal_name, breed,  type,
notes,time, description from ANIMAL_OWNER JOIN ANIMAL ON ANIMAL.ssn =
ANIMAL_OWNER.ssn JOIN ANIMAL_OWNER_PHONE_NUMBER ON
ANIMAL_OWNER_PHONE_NUMBER.ssn = ANIMAL_OWNER.ssn JOIN APPOINTMENT
ON ANIMAL.animal_id = APPOINTMENT.animal_id WHERE a_date = CURRENT_DATE
ORDER BY time;  2
```

View created.

```
SQL> CREATE VIEW FUTURE_APPOINTMENTS AS
SELECT ANIMAL.name as animal_name, type, notes, a_date, time, CLINIC.name,
CLINIC.address, CLINIC.city, CLINIC.state, CLINIC.zip_code,
CLINIC_PHONE_NUMBER.phone_number FROM ANIMAL_OWNER JOIN ANIMAL ON
ANIMAL_OWNER.ssn = ANIMAL.ssn JOIN APPOINTMENT ON ANIMAL.animal_id =
APPOINTMENT.animal_id JOIN CLINIC ON APPOINTMENT.clinic_id = CLINIC.clinic_id
JOIN CLINIC_PHONE_NUMBER ON CLINIC.clinic_id =
CLINIC_PHONE_NUMBER.clinic_id WHERE a_date >= CURRENT_DATE ORDER BY
a_date,time;  2
```

View created.

```
SQL> CREATE VIEW CLINIC_EMPLOYEES AS
SELECT first_name, last_name, title, phone_number, email, clinic_id FROM EMPLOYEE
JOIN JOB_TITLE ON EMPLOYEE.job_title_id = JOB_TITLE.job_title_id JOIN
```

```
EMPLOYEE_PHONE_NUMBER ON EMPLOYEE_PHONE_NUMBER.ssn =
EMPLOYEE.ssn ORDER BY first_name, last_name;  2

View created.


SQL> CREATE VIEW PERSCRIPTION_DETAILS AS
SELECT EMPLOYEE.first_name AS vet_first_name, EMPLOYEE.middle_name,
EMPLOYEE.last_name AS vet_last_name, ANIMAL.name as animal_name, p_date, time,
description, instructions, number_of_refills, DRUG.name, manufacturer, potency,
ANIMAL_OWNER.first_name, ANIMAL_OWNER.last_name FROM EMPLOYEE JOIN
PRESCRIPTION ON PRESCRIPTION.ssn = EMPLOYEE.ssn JOIN ANIMAL ON
PRESCRIPTION.animal_id = ANIMAL.animal_id JOIN ANIMAL_OWNER ON
ANIMAL_OWNER.ssn = ANIMAL.ssn JOIN PRESCRIPTION_DRUG ON
PRESCRIPTION_DRUG.prescription_id = PRESCRIPTION.prescription_id JOIN DRUG ON
PRESCRIPTION_DRUG.drug_id = DRUG.drug_id ORDER BY DRUG.name;  2

View created.

SQL> CREATE VIEW CLINIC_TOTALS_BY_EMPLOYEE AS
SELECT SUM(amount) AS total_revenue, CLINIC.name AS clinic_name,
EMPLOYEE.first_name, EMPLOYEE.last_name, EMPLOYEE.ssn FROM CLINIC JOIN
EMPLOYEE ON CLINIC.clinic_id = EMPLOYEE.clinic_id JOIN
EMPLOYEE_EXAMINATION ON EMPLOYEE_EXAMINATION.ssn = EMPLOYEE.ssn
JOIN EXAMINATION ON EMPLOYEE_EXAMINATION.examination_id =
EXAMINATION.examination_id JOIN SALE SALE ON EXAMINATION.sale_id =
SALE.sale_id JOIN SALE_CHARGE ON SALE.sale_id = SALE_CHARGE.sale_id JOIN
CHARGE ON SALE_CHARGE.charge_id = CHARGE.charge_id GROUP BY
CLINIC.name,EMPLOYEE.first_name,EMPLOYEE.last_name, EMPLOYEE.ssn;  2

View created.

SQL> CREATE VIEW CLINIC_TOTALS AS
SELECT SUM(total_revenue) AS clinic_income, clinic_name, royalty,
ROUND(royalty*SUM(total_revenue),2) AS vet_pay,
ROUND((1-royalty)*SUM(total_revenue),2) AS profit FROM
CLINIC_TOTALS_BY_EMPLOYEE JOIN CONTRACT ON CONTRACT.ssn =
CLINIC_TOTALS_BY_EMPLOYEE.ssn GROUP BY clinic_name, royalty;  2
```

View created.

```
SQL> CREATE VIEW MOST_RECENT_EXAM AS
SELECT ANIMAL.animal_id, name as animal_name, breed, type, weight, heart_beat,
systolic_bp, diastolic_bp, animal_condition, observations, e_date FROM ANIMAL JOIN
EXAMINATION ON ANIMAL.animal_id = EXAMINATION.examination_id JOIN
EXAM_ANIMAL_ATTRIBUTES ON EXAMINATION.examination_id =
EXAM_ANIMAL_ATTRIBUTES.examination_id ORDER BY e_date;  2
```

View created.

```
SQL> CREATE VIEW PRESCRIPTION_VIEW AS
SELECT EMPLOYEE.first_name, EMPLOYEE.last_name, instructions,  DRUG.name,
CONCAT(potency, 'mg') as potency from EMPLOYEE JOIN prescription ON EMPLOYEE.ssn
= PRESCRIPTION.ssn JOIN PRESCRIPTION_DRUG on PRESCRIPTION.prescription_id =
PRESCRIPTION_DRUG.prescription_id join DRUG on PRESCRIPTION_DRUG.drug_id =
DRUG.drug_id;  2
```

View created.

```
SQL> CREATE VIEW VET_DAILY_APPOINTMENTS AS
SELECT ANIMAL.name, ANIMAL.type, first_name, last_name, time FROM ANIMAL JOIN
APPOINTMENT ON ANIMAL.animal_id = APPOINTMENT.animal_id JOIN
ANIMAL_OWNER ON ANIMAL.ssn = ANIMAL_OWNER.ssn WHERE
APPOINTMENT.a_date = CURRENT_DATE ORDER BY APPOINTMENT.time;  2
```

View created.

```
SQL> CREATE VIEW OTHER_CLINICS_VISITED AS
SELECT DISTINCT CLINIC.name, CLINIC_PHONE_NUMBER.phone_number,
CLINIC.email, ANIMAL.animal_id from CLINIC JOIN CLINIC_PHONE_NUMBER ON
CLINIC.clinic_id = CLINIC_PHONE_NUMBER.clinic_id JOIN APPOINTMENT ON
APPOINTMENT.clinic_id = APPOINTMENT.clinic_id JOIN Animal ON
APPOINTMENT.animal_id = ANIMAL.animal_id;  2
```

View created.

```
SQL> CREATE USER ADDISON3745 IDENTIFIED BY password;
```

```
User created.

SQL> CREATE USER CONOR7463 IDENTIFIED BY password;

User created.

SQL> CREATE USER BRADLEY7348 IDENTIFIED BY password;

User created.

SQL> CREATE USER ANGELICA3943 IDENTIFIED BY password;

User created.

SQL> CREATE USER WYATT7483 IDENTIFIED BY password;

User created.

SQL> CREATE USER JASON6540 IDENTIFIED BY password;

User created.

SQL> CREATE USER JOHN8434 IDENTIFIED BY password;

User created.

SQL> CREATE ROLE ADMIN;
CREATE ROLE ADMIN
            *
ERROR at line 1:
ORA-01921: role name 'ADMIN' conflicts with another user or role name


SQL> CREATE ROLE ADMINISTRATOR;

Role created.

SQL> CREATE ROLE VETERINARIAN;
```

Role created.

SQL> CREATE ROLE VET_TECH;

Role created.

SQL> CREATE ROLE RECEPTIONIST;

Role created.

SQL> CREATE ROLE PET_OWNER;

Role created.

SQL> GRANT CREATE SESSION TO ADMINISTRATOR;

Grant succeeded.

SQL> GRANT CREATE SESSION TO VET_TECH;

Grant succeeded.

SQL> CREATE ROLE RECEPTIONIST;
CREATE ROLE RECEPTIONIST
*
ERROR at line 1:
ORA-01921: role name 'RECEPTIONIST' conflicts with another user or role name

SQL> GRANT CREATE SESSION TO RECEPTIONIST;

Grant succeeded.

SQL> GRANT CREATE SESSION TO PET_OWNER;

Grant succeeded.

SQL> GRANT CREATE SESSION,DBA TO ADMINISTRATOR;

Grant succeeded.

SQL> GRANT SELECT ON VET_DAILY_APPOINTMENTS TO VETERINARIAN;

Grant succeeded.

SQL> GRANT INSERT,DELETE,UPDATE ON EXAM_ANIMAL_ATTRIBUTES TO VETERINARIAN;

Grant succeeded.

SQL> GRANT INSERT,DELETE,UPDATE ON SALE TO VETERINARIAN;

Grant succeeded.

SQL> GRANT INSERT,DELETE,UPDATE ON SALE_CHARGE TO VETERINARIAN;

Grant succeeded.

SQL> GRANT SELECT ON CHARGE TO VETERINARIAN;

Grant succeeded.

SQL> GRANT SELECT ON CONTRACT TO VETERINARIAN;

Grant succeeded.

SQL> GRANT INSERT,UPDATE,DELTE ON PRESCRIPTION,PRESCRIPTION_DRUG TO VETERINARIAN;
GRANT INSERT,UPDATE,DELTE ON PRESCRIPTION,PRESCRIPTION_DRUG TO VETERINARIAN
*
ERROR at line 1:
ORA-00990: missing or invalid privilege

SQL> GRANT INSERT,UPDATE,DELETE ON PRESCRIPTION,PRESCRIPTION_DRUG TO VETERINARIAN;

```
GRANT INSERT,UPDATE,DELETE ON PRESCRIPTION,PRESCRIPTION_DRUG TO
VETERINARIAN
                           *
ERROR at line 1:
ORA-00905: missing keyword


SQL> GRANT INSERT,UPDATE,DELETE ON PRESCRIPTION TO VETERINARIAN;

Grant succeeded.

SQL> GRANT INSERT UPDATE ON EXAMINATION TO VETERINARIAN;
GRANT INSERT UPDATE ON EXAMINATION TO VETERINARIAN
     *
ERROR at line 1:
ORA-00990: missing or invalid privilege


SQL> GRANT INSERT, UPDATE ON EXAMINATION TO VETERINARIAN;

Grant succeeded.

SQL> GRANT SELECT ON MOST_RECENT_EXAM TO VETERINARIAN;

Grant succeeded.

SQL> GRANT SELECT ON CLINIC_TOTALS TO VETERINARIAN;

Grant succeeded.

SQL> GRANT SELECT ON CLINIC_TOTALS_BY_EMPLOYEE TO ADMINISTRATOR;

Grant succeeded.

SQL> GRANT INSERT,DELETE,UPDATE ON EXAM_ANIMAL_ATTRIBUTES TO
VETERINARIAN;

Grant succeeded.
```

```
SQL> GRANT INSERT,DELETE,UPDATE ON SALE TO VETERINARIAN;

Grant succeeded.

SQL> GRANT INSERT,DELETE,UPDATE ON SALE_CHARGE TO VETERINARIAN;

Grant succeeded.

SQL> GRANT SELECT ON CHARGE TO VETERINARIAN;

Grant succeeded.

SQL> GRANT SELECT ON CONTRACT TO VETERINARIAN;

Grant succeeded.

SQL> GRANT INSERT,UPDATE,DELETE ON PRESCRIPTION TO VETERINARIAN;

Grant succeeded.

SQL> GRANT INSERT,UPDATE,DELETE ON PRESCRIPTION_DRUG TO
VETERINARIAN;

Grant succeeded.

SQL> GRANT INSERT, UPDATE ON EXAMINATION TO VETERINARIAN;

Grant succeeded.

SQL> GRANT SELECT ON MOST_RECENT_EXAM TO VETERINARIAN;

Grant succeeded.

SQL> GRANT SELECT ON VET_DAILY_APPOINTMENTS TO VET_TECH;

Grant succeeded.

SQL> GRANT INSERT, UPDATE ON EXAM_ANIMAL_ATTRIBUTES TO VET_TECH;
```

```
Grant succeeded.

SQL> GRANT INSERT,UPDATE ON SALE TO VET_TECH;

Grant succeeded.

SQL> GRANT INSERT,UPDATE ON SALE_CHARGE TO VET_TECH;

Grant succeeded.

SQL> GRANT SELECT ON CHARGE TO VET_TECH;

Grant succeeded.

SQL> GRANT INSERT, UPDATE ON EXAMINATION TO VET_TECH;

Grant succeeded.

SQL> GRANT SELECT ON MOST_RECENT_EXAM TO VET_TECH;

Grant succeeded.

SQL> GRANT SELECT ON DAILY_APPOINTMENTS TO RECEPTIONIST;

Grant succeeded.

SQL> GRANT SELECT ON CLINIC_TOTALS_BY_EMPLOYEE TO ADMINISTRATOR;

Grant succeeded.

SQL> GRANT SELECT ON OWNER_ANIMAL_INFO TO PET_OWNER;

Grant succeeded.

SQL> GRANT SELECT ON OTHER_CLINCS_VISITED TO VETERINARIAN;
GRANT SELECT ON OTHER_CLINCS_VISITED TO VETERINARIAN
             *
ERROR at line 1:
ORA-00942: table or view does not exist
```

```
SQL> CREATE VIEW OTHER_CLINICS_VISITED AS
SELECT DISTINCT CLINIC.name, CLINIC_PHONE_NUMBER.phone_number,
CLINIC.email, ANIMAL.animal_id from CLINIC JOIN CLINIC_PHONE_NUMBER ON
CLINIC.clinic_id = CLINIC_PHONE_NUMBER.clinic_id JOIN APPOINTMENT ON
APPOINTMENT.clinic_id = APPOINTMENT.clinic_id JOIN Animal ON
APPOINTMENT.animal_id = ANIMAL.animal_id;  2
CREATE VIEW OTHER_CLINICS_VISITED AS
       *
ERROR at line 1:
ORA-00955: name is already used by an existing object


SQL> GRANT SELECT ON OTHER_CLINICS_VISITED TO VETERINARIAN;

Grant succeeded.

SQL> GRANT PET_OWNER TO ADDISON3745;

Grant succeeded.

SQL> GRANT PET_OWNER TO CONOR7463;

Grant succeeded.

SQL>
SQL> GRANT VETERINARIAN TO BRADLEY7348;

Grant succeeded.

SQL> GRANT RECEPTIONIST TO ANGELICA3943;

Grant succeeded.

SQL> GRANT VET_TECH TO WYATT7483;

Grant succeeded.
```

```
SQL> GRANT VET_TECH TO JASON6540;

Grant succeeded.

SQL> GRANT RECEPTIONIST TO JOHN8434;

Grant succeeded.

SQL>
```