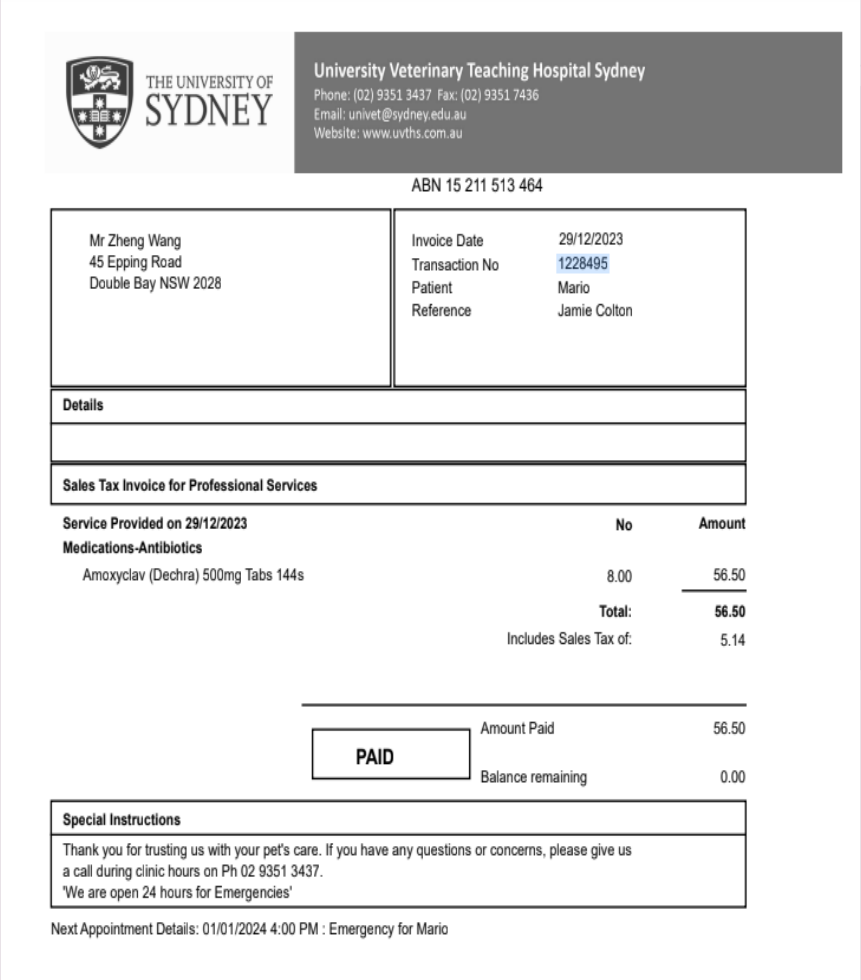
31061 Database Principles
Assessment 5

- Student Name： Zheng Wang
- Student ID： 14403000
- Project name： University Veterinary Teaching Hospital Sydney

- I chose this project because, in December, my dog was bitten by another dog. It was a Sunday, and the University of Sydney Veterinary Hospital was the only one providing emergency services at that time.



THE UNIVERSITY OF SYDNEY

University Veterinary Teaching Hospital Sydney
Phone: (02) 9351 3437 Fax: (02) 9351 7436
Email: univet@sydney.edu.au
Website: www.uvths.com.au

ABN 15 211 513 464

Mr Zheng Wang 45 Epping Road Double Bay NSW 2028	Invoice Date 29/12/2023 Transaction No 1228495 Patient Mario Reference Jamie Colton

Details

Sales Tax Invoice for Professional Services

Service Provided on 29/12/2023	**No**	**Amount**
Medications-Antibiotics		
Amoxyclav (Dechra) 500mg Tabs 144s	8.00	56.50
	Total:	**56.50**
	Includes Sales Tax of:	5.14

PAID

Amount Paid	56.50
Balance remaining	0.00

Special Instructions

Thank you for trusting us with your pet's care. If you have any questions or concerns, please give us a call during clinic hours on Ph 02 9351 3437.
'We are open 24 hours for Emergencies'

Next Appointment Details: 01/01/2024 4:00 PM : Emergency for Mario

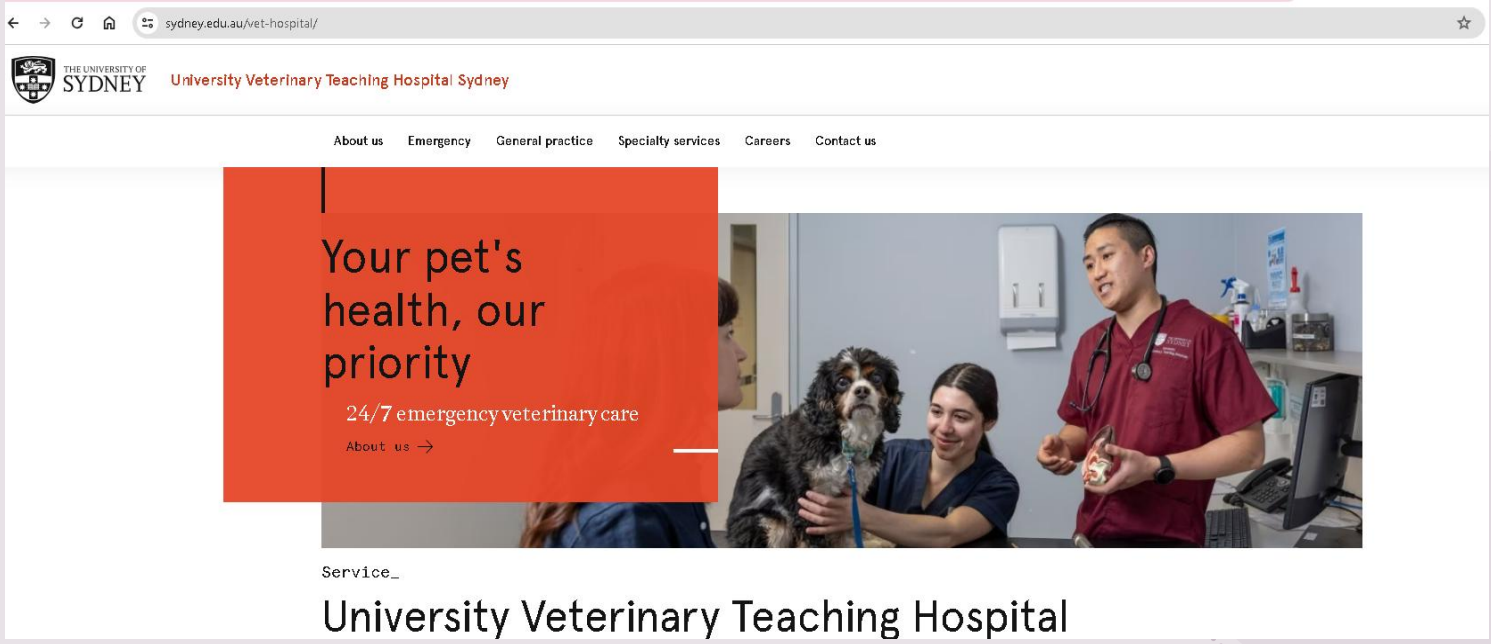
The database model of University Veterinary Teaching Hospital Sydney

The database model has three main modules:

Sick Animal Information: This includes information about the sick animal, its owner, and pet insurance details.

Veterinary Hospital Information: This covers details of the hospital staff, and information on doctors and nurses involved in the treatment.

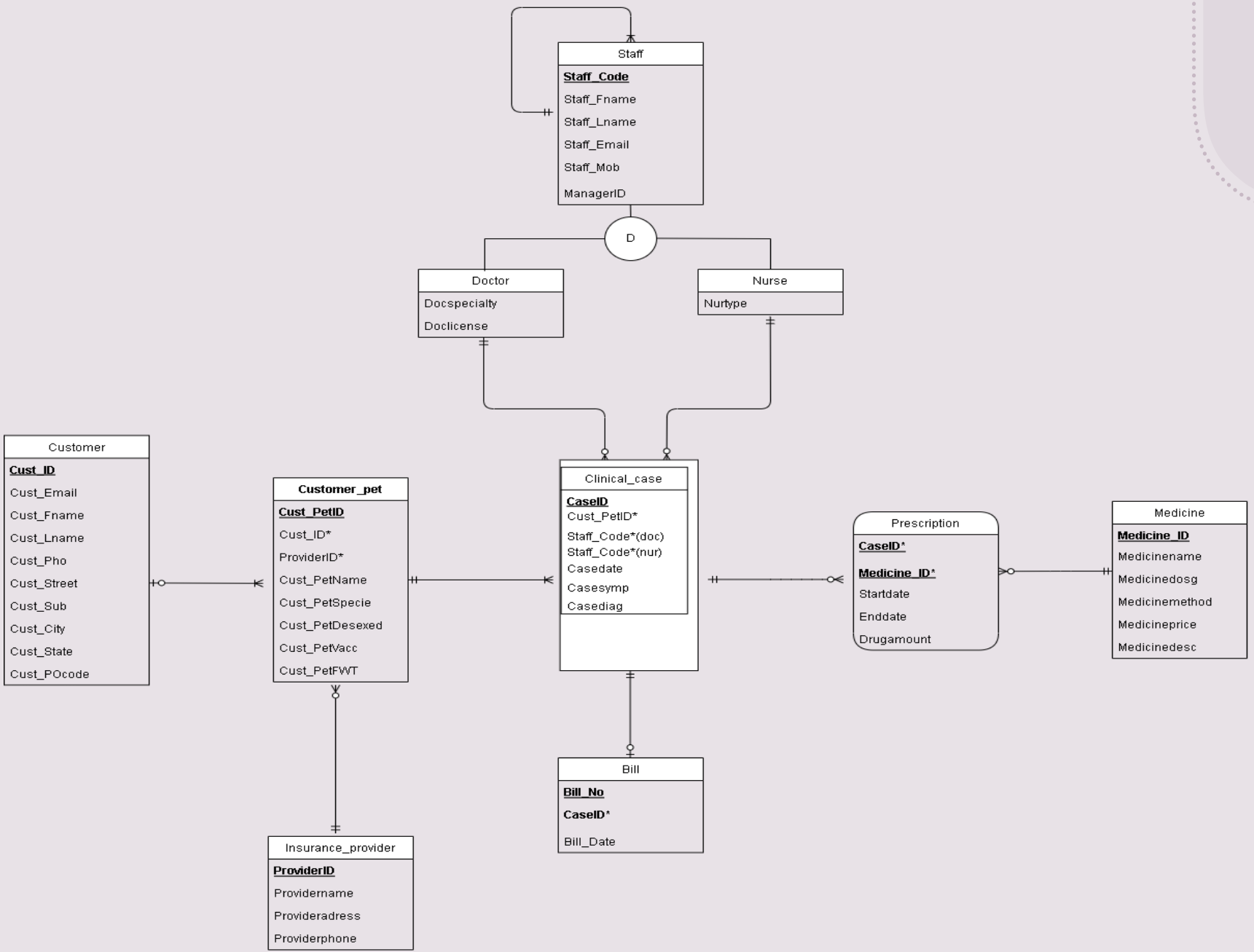
Treatment and Medication Information: This includes details of each case, prescription information, and medication details



https://www.sydney.edu.au/vet-hospital/

Zheng Wang 14403000

31061 Database Principles
Zheng Wang 14403000
assessment 5 Veterinary database



Zheng Wang 14403000

Illustrating a One-to-Many Relationship

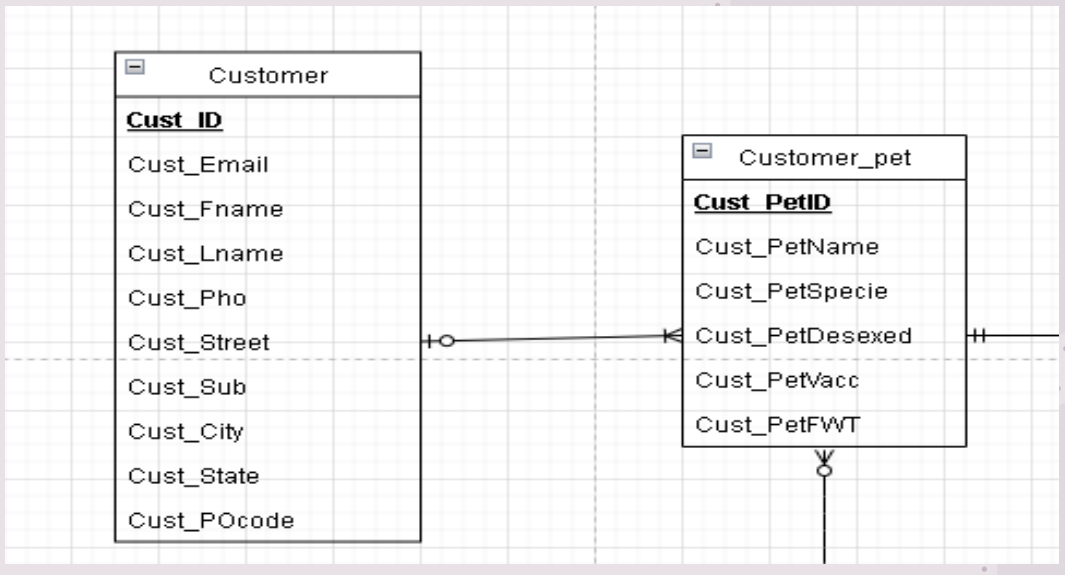
In this context, a Customer can have multiple pets, but each pet is owned by only one customer.

The **Customer** table has a one-to-many relationship with the **Customer_pet** table.

The relationship is represented by the foreign key **Cust_ID** in the **Customer_pet** table that references the primary key **Cust_ID** in the **Customer** table.

Customer: Contains information about each customer, including their ID, email, first name, last name, and phone number.

Customer_pet: Lists all pets owned by customers, with details such as pet name, species, and ownership (linked through **Cust_ID**).



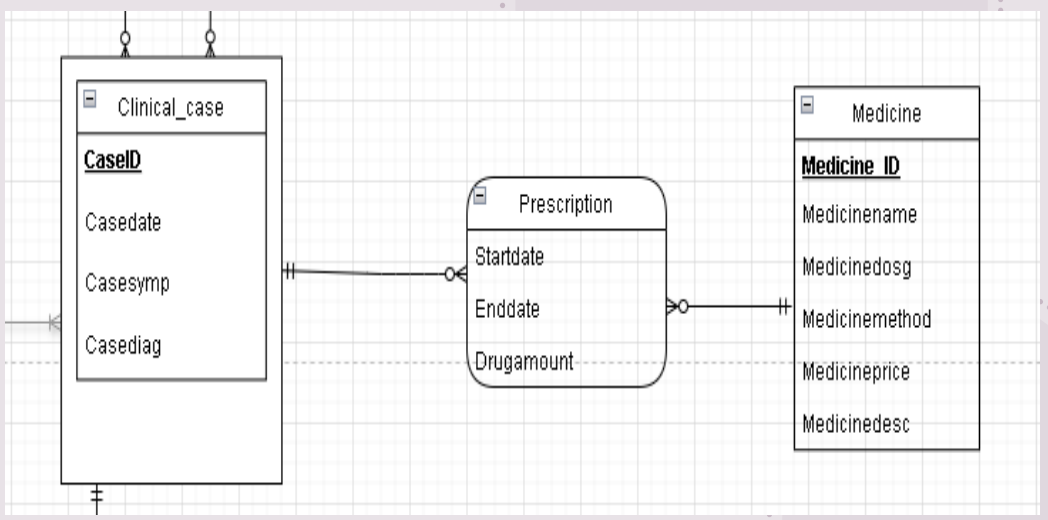

```
postgres=# select* from Customer;
 cust_id |       cust_email        | cust_fname | cust_lname | cust_pho | cust_street | cust_sub | cust_city | cust_state | cust_pocode | cust_divln
---------+-------------------------+------------+------------+----------+-------------+----------+-----------+------------+-------------+------------
       1 | john.doe@example.com    | John       | Doe        | 1234567890 | 123 Elm St | Suburb A | Sydney    | NSW        |        2000 |   12345678
       2 | jane.smith@example.com  | Jane       | Smith      | 2345678901 | 456 Oak St | Suburb B | Melbourne | VIC        |        3000 |   23456789
       3 | alice.brown@example.com | Alice      | Brown      | 3456789012 | 789 Pine St | Suburb C | Brisbane | QLD        |        4000 |   34567890
(3 rows)

postgres=# select * from Customer_pet;
 cust_petid | cust_petname | cust_petspecie | cust_petdesexed | cust_petvacc | cust_petfwt | cust_id
------------+--------------+----------------+-----------------+--------------+-------------+---------
          1 | Buddy        | Dog            | Yes             | Yes          | No          |       1
          2 | Mittens      | Cat            | No              | Yes          | Yes         |       2
          3 | Max          | Rabbit         | Yes             | No           | Yes         |       3
(3 rows)
```

Illustrating a Many-to-Many Relationship

In this context, a clinical case can have multiple medicines prescribed, and a single medicine can be prescribed in multiple cases.

Clinical_case table has a many-to-many relationship with the **Medicine** table.
The **Prescription** handles this relationship, connecting **Clinical_case** and **Medicine.**

Clinical_case: Contains information about each clinical case associated with a pet.
Medicine: Lists all available medicines, including their details.
Prescription: Acts as a bridge between Clinical_case and Medicine, providing information on which medicines are prescribed for each clinical case, as well as start and end dates of prescriptions.




```
postgres=# select * from Clinical_case;
 caseid | cust_petid |  casedate  | casesymp |     casediag      | doctorcode | nursecode
--------+------------+------------+----------+-------------------+------------+-----------
    101 |          1 | 2024-01-15 | Fever    | Infection         | S002       | S004
    102 |          2 | 2024-02-20 | Limping  | Fracture          | S002       | S005
    103 |          3 | 2024-03-25 | Cough    | Respiratory issue | S003       | S006
(3 rows)

postgres=# select * from Medicine;
 medicine_id | medicinename | medicinedosg | medicinemethod | medicineprice |          medicinedesc
-------------+--------------+--------------+----------------+---------------+-----------------------------------
         301 | Amoxicillin  | 500mg        | Oral           |         20.00 | Antibiotic for infection
         302 | Ibuprofen    | 200mg        | Oral           |         10.00 | Pain relief and anti-inflammatory
         303 | Saline       | 10ml         | Injection      |          5.00 | Rehydration solution
(3 rows)

postgres=# select * from Prescription;
 startdate  |  enddate   | drugamount | caseid | cust_petid | medicine_id
------------+------------+------------+--------+------------+-------------
 2024-01-16 | 2024-01-20 | 10mg       |    101 |          1 |         301
 2024-02-21 | 2024-02-26 | 15mg       |    102 |          2 |         302
 2024-03-26 | 2024-03-31 | 5ml        |    103 |          3 |         303
(3 rows)
```

Zheng Wang 14403000

The various queries

a. A Simple Query of a Single Table: List all customer names and their email addresses.
SELECT Cust_Fname, Cust_Lname, Cust_Email FROM Customer;

```
postgres=# SELECT Cust_Fname, Cust_Lname, Cust_Email FROM Customer;
 cust_fname | cust_lname |       cust_email
------------+------------+------------------------
 John       | Doe        | john.doe@example.com
 Jane       | Smith      | jane.smith@example.com
 Alice      | Brown      | alice.brown@example.com
(3 rows)
```

b. A Query Using "Natural Join":Find all prescriptions with the related pet name and medication name.
SELECT Prescription.Startdate, Prescription.Enddate, Customer_pet.Cust_PetName, Medicine.Medicinename
FROM Prescription NATURAL JOIN Customer_pet NATURAL JOIN Medicine;

```
postgres=# SELECT Prescription.Startdate, Prescription.Enddate, Customer_pet.Cust_PetName, Medicine.Med
 startdate  |  enddate   | cust_petname | medicinename
------------+------------+--------------+--------------
 2024-01-16 | 2024-01-20 | Buddy        | Amoxicillin
 2024-02-21 | 2024-02-26 | Mittens      | Ibuprofen
 2024-03-26 | 2024-03-31 | Max          | Saline
(3 rows)
```

c. Cross Product Equivalent to the "Natural Join" Query:Find all prescriptions, including their start and end dates, the associated pet name, and the medication name, based on their IDs.
SELECT Prescription.Startdate, Prescription.Enddate, Customer_pet.Cust_PetName, Medicine.Medicinename FROM Prescription NATURAL JOIN Customer_pet NATURAL JOIN Medicine;

```
postgres=# SELECT Prescription.Startdate, Prescription.Enddate, C
tID AND Prescription.Medicine_ID = Medicine.Medicine_ID;
 startdate  |  enddate   | cust_petname | medicinename
------------+------------+--------------+--------------
 2024-01-16 | 2024-01-20 | Buddy        | Amoxicillin
 2024-02-21 | 2024-02-26 | Mittens      | Ibuprofen
 2024-03-26 | 2024-03-31 | Max          | Saline
(3 rows)
```

Zheng Wang 14403000

The various queries

d. Query Involving "GROUP BY" (with "HAVING"):Count the number of cases treated by each doctor, only showing doctors who have treated more than one case.

SELECT DoctorCode, COUNT(CaseID) AS NumCases FROM Clinical_case GROUP BY DoctorCode HAVING COUNT(CaseID) > 1;

```
postgres=# SELECT DoctorCode, COUNT(CaseID) AS NumCases FROM
 doctorcode | numcases
------------+----------
 S002       |        2
(1 row)
```

e. Query Using a Subquery:Find the pet names of pets that have been prescribed "Amoxicillin".

SELECT Cust_PetName FROM Customer_pet WHERE Cust_PetID IN (SELECT Cust_PetID FROM Prescription WHERE Medicine_ID = (SELECT Medicine_ID FROM Medicine WHERE Medicinename = 'Amoxicillin'));

```
postgres=# SELECT Cust_PetName FROM Customer
 cust_petname
--------------
 Buddy
(1 row)
```

f. Cross Product That Cannot Be Implemented with "Natural Join" (Self-Join):Find pairs of staff members who have the same manager.

SELECT A.Staff_Fname AS Staff1, A.Staff_Lname AS LastName1, B.Staff_Fname AS Staff2, B.Staff_Lname AS LastName2, A.ManagerID FROM Staff A JOIN Staff B ON A.ManagerID = B.ManagerID WHERE A.Staff_code < B.Staff_code;

```
postgres=# SELECT A.Staff_Fname AS Staff1, A.Staff_Lname AS LastName1, B.Staff_Fname AS Staff2, B.Staff_Lname AS LastName2, A.ManagerID
    FROM Staff A JOIN Staff B ON A.ManagerID = B.ManagerID WHERE A.Staff_code < B.Staff_code;
 staff1 | lastname1 | staff2 | lastname2 | managerid
--------+-----------+--------+-----------+-----------
 Jack   | Black     | Olivia | Green     | S001
 Liam   | Jones     | Sophia | Davis     | S002
(2 rows)
```

Zheng Wang 14403000

CHECK statements

1. **Cust_Pho** Length Check in **Customer** Table:
   Ensures that the **Cust_Pho** (customer phone number) is exactly 10 characters long.
2. **Cust_State** Validation in Customer Table:
   Ensures that the **Cust_State** value is one of the valid Australian states.

```sql
CREATE TABLE Customer (
    Cust_ID       NUMERIC(5)     NOT NULL,
    Cust_Email    VARCHAR(30),
    Cust_Fname    VARCHAR(15),
    Cust_Lname    VARCHAR(20),
    Cust_Pho      VARCHAR(10),
    Cust_Street   VARCHAR(30),
    Cust_Sub      VARCHAR(20),
    Cust_City     VARCHAR(20) DEFAULT 'Sydney',
    Cust_State    VARCHAR(4)  DEFAULT 'NSW',
    Cust_Pocode   NUMERIC(4),
    Cust_DivLN    VARCHAR(8),
    CONSTRAINT Customer_PK PRIMARY KEY (Cust_ID),
    CHECK(LENGTH(Cust_Pho) = 10),
    CHECK(Cust_State IN ('NSW', 'VIC', 'QLD', 'SA', 'WA', 'TAS', 'NT', 'ACT')),
    CHECK(Cust_Pocode BETWEEN 200 AND 9999),
    CHECK(LENGTH(Cust_DivLN) = 8)
);
```

```sql
-- Create Customer_pet Table
CREATE TABLE Customer_pet (
    Cust_PetID       NUMERIC(5)     NOT NULL,
    Cust_PetName     VARCHAR(20),
    Cust_PetSpecie   VARCHAR(20),
    Cust_PetDesexed  VARCHAR(3), CHECK (Cust_PetDesexed IN ('Yes', 'No')),
    Cust_PetVacc     VARCHAR(3), CHECK (Cust_PetVacc IN ('Yes', 'No')),
    Cust_PetFWT      VARCHAR(3), CHECK (Cust_PetFWT IN ('Yes', 'No')),
    Cust_ID          NUMERIC(5),
    CONSTRAINT Cust_PetID_PK PRIMARY KEY (Cust_PetID),
    CONSTRAINT Cust_PetID_FK FOREIGN KEY (Cust_ID) REFERENCES Customer (Cust_ID) ON DELETE CASCADE
);
```

3. **Cust_PetDesexed** Validation in **Customer_pet** Table
   Ensures that the **Cust_PetDesexed** column only accepts the values 'Yes' or 'No'
4. **Cust_State** Validation in Customer Table:
   Ensures that the **Cust_PetVacc** only accepts 'Yes' or 'No'..

5. Docspecialty Validation in **Doctor** Table.
   Ensures that the **Docspecialty** can only be either 'General practice' or 'Specialty services'.
6. Nurtype Validation in **Nurse** Table.
   Ensures that the **Nurtype** column in the **Nurse** table can only take on the values 'Surgery', 'Emergency', or 'Care'.

```sql
-- Create Doctor Table
CREATE TABLE Doctor (
    Staff_code          VARCHAR(5),
    Docspecialty        VARCHAR(20),
    CONSTRAINT Doctor_PK PRIMARY KEY (Staff_code),
    CONSTRAINT Doctor_FK FOREIGN KEY (Staff_code) REFERENCES Staff (Staff_code) ON DELETE CASCADE,
    CHECK (Docspecialty IN ('General practice', 'Specialty services'))
);

-- Create Nurse Table
CREATE TABLE Nurse (
    Staff_code     VARCHAR(5),
    Nurtype        VARCHAR(20),
    CONSTRAINT Nurse_PK PRIMARY KEY (Staff_code),
    CONSTRAINT Nurse_FK FOREIGN KEY (Staff_code) REFERENCES Staff (Staff_code) ON DELETE CASCADE,
    CHECK (Nurtype IN ('Surgery', 'Emergency', 'Care'))
);
```

Zheng Wang 14403000

“ON DELETE RESTRICT” and “ON DELETE CASCADE”,

Example of ON DELETE RESTRICT

Scenario: In the **Clinical_case** table has a foreign key that references **Doctor** through the **DoctorCode** column. The **ON DELETE RESTRICT** action prevents a doctor from being deleted if they are still referenced in a clinical case.

If trying to delete a doctor from the **Doctor** table while there are rows in the **Clinical_case** table referencing this doctor, the deletion will be restricted, and an error will be thrown. This helps to maintain referential integrity by ensuring that you don't delete a doctor who is still associated with active clinical cases.

```
-- Create Clinical_case Table
CREATE TABLE Clinical_case (
    CaseID                      NUMERIC(10),
    Cust_PetID                  NUMERIC(5),
    Casedate                    DATE,
    Casesymp                    VARCHAR(500),
    Casediag                    VARCHAR(500),
    DoctorCode                  VARCHAR(5),
    NurseCode                   VARCHAR(5),
    CONSTRAINT Clinical_case_PK PRIMARY KEY (CaseID, Cust_PetID),
    CONSTRAINT Clinical_case_FK1 FOREIGN KEY (Cust_PetID) REFERENCES Customer_pet (Cust_PetID) ON DELETE CASCADE,
    CONSTRAINT Clinical_case_FK2 FOREIGN KEY (DoctorCode) REFERENCES Doctor (Staff_code) ON DELETE RESTRICT,
    CONSTRAINT Clinical_case_FK3 FOREIGN KEY (NurseCode)  REFERENCES Nurse (Staff_code) ON DELETE RESTRICT
);
```

```
-- Create Customer_pet Table
CREATE TABLE Customer_pet (
    Cust_PetID       NUMERIC(5)    NOT NULL,
    Cust_PetName     VARCHAR(20),
    Cust_PetSpecie   VARCHAR(20),
    Cust_PetDesexed  VARCHAR(3), CHECK (Cust_PetDesexed IN ('Yes', 'No')),
    Cust_PetVacc     VARCHAR(3), CHECK (Cust_PetVacc IN ('Yes', 'No')),
    Cust_PetFWT      VARCHAR(3), CHECK (Cust_PetFWT IN ('Yes', 'No')),
    Cust_ID          NUMERIC(5),
    CONSTRAINT Cust_PetID_PK PRIMARY KEY (Cust_PetID),
    CONSTRAINT Cust_PetID_FK FOREIGN KEY (Cust_ID) REFERENCES Customer (Cust_ID) ON DELETE CASCADE
);
```

Example of ON DELETE CASCADE

Scenario: In the **Customer_pet** table has a foreign key that references **Customer**. The ON DELETE CASCADE action will automatically delete all rows in the **Customer_pet** table when a corresponding row in the **Customer** table is deleted.

If a row in the **Customer** table is deleted, any corresponding rows in the **Customer_pet** table that reference this customer through **Cust_ID** will also be automatically deleted. This is useful when the child rows are entirely dependent on the existence of their parent rows.

View

A view is created to displays detailed information about prescriptions, including the pet's name, the prescribed medicine, and the start and end dates of the prescription.

This view joins the **Prescription**, **Customer_pet**, and **Medicine tables** to provide a comprehensive overview of each prescription, including:

1. The start and end date of the prescription.
2. The name of the pet associated with the prescription.
3. The name, dosage, and method of the medicine prescribed.

As a virtual table, use select statement:

SELECT * FROM PrescriptionDetails;

```
--Create a view(a virtual table) that displays detailed information about prescriptions
CREATE VIEW PrescriptionDetails AS
SELECT
    Prescription.Startdate,
    Prescription.Enddate,
    Customer_pet.Cust_PetName,
    Medicine.Medicinename,
    Medicine.Medicinedosg,
    Medicine.Medicinemethod
FROM
    Prescription
JOIN
    Customer_pet ON Prescription.Cust_PetID = Customer_pet.Cust_PetID
JOIN
    Medicine ON Prescription.Medicine_ID = Medicine.Medicine_ID;
```

/home/as5.txt 146:32 Spaces: 4 (Auto)

Terminal

```
postgres=#
postgres=#
postgres=# select * from PrescriptionDetails;
 startdate  |  enddate   | cust_petname | medicinename | medicinedosg | medicinemethod
------------+------------+--------------+--------------+--------------+----------------
 2024-01-16 | 2024-01-20 | Buddy        | Amoxicillin  | 500mg        | Oral
 2024-02-21 | 2024-02-26 | Mittens      | Ibuprofen    | 200mg        | Oral
 2024-03-26 | 2024-03-31 | Max          | Saline       | 10ml         | Injection
(3 rows)
```

Zheng Wang 14403000

Thank you for your time

ZHENG WANG 14403000