# Pet Care Veterinary Clinic Management System

Project Title: Pet Care Veterinary Clinic Management System.

Database: PetCareDB

Git-hub link: https://github.com/Sha-bab/Pet-Care-Veterinary-Clinic-DBM/blob/main/PetCareDB.sql

**Project Overview:**

The PetCareDB project is a relational database system designed to manage operations of a pet care and veterinary clinic. It stores and manages information related to customers, pets, veterinarians, appointments, medical records, orders, products, services, employees, and payments. The goal of this project is to ensure efficient data storage, reduce redundancy, and maintain data integrity.

**Project ER-D *(Indented)*:**

The initial idea about the tables and the attributes of each table and relationship between the tables to build the database.

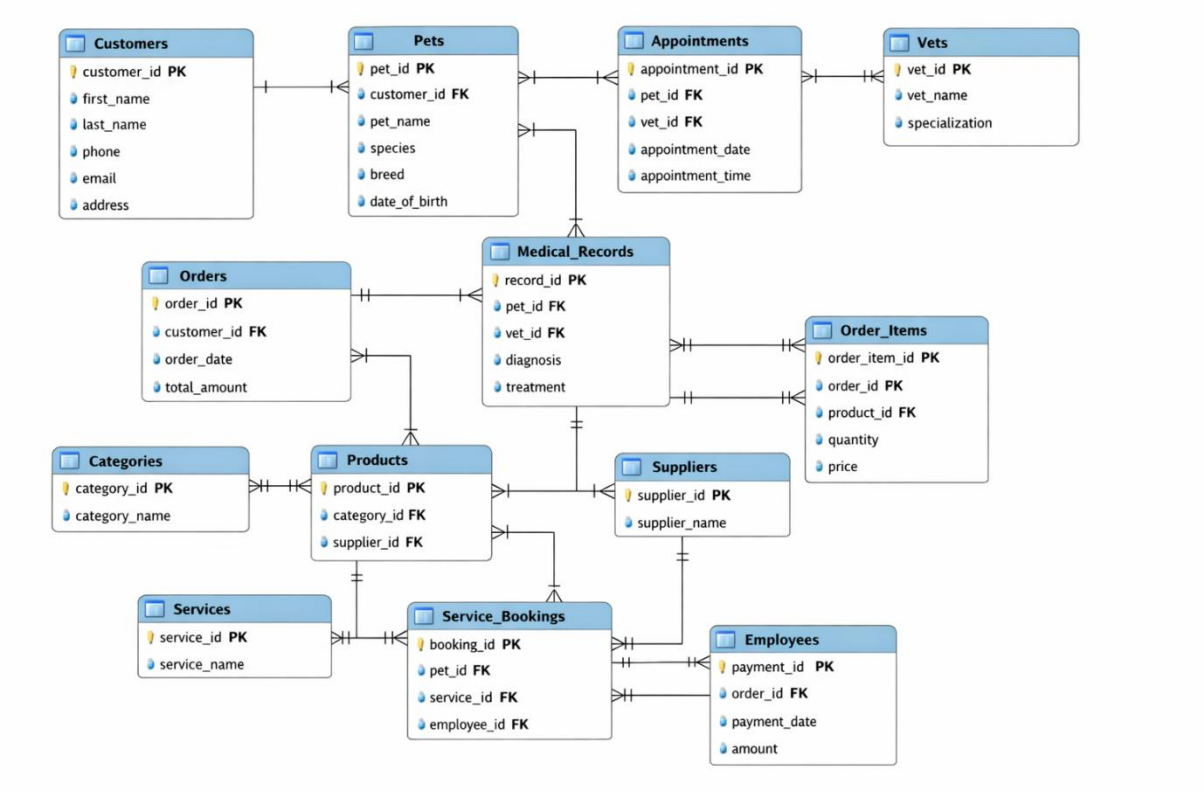



Fig: Draft for tables and their connection

**Table Creation:**

By following the ER D creating all the tables and establishing the connection between all of them.

1️⃣ Database Creation

```
CREATE DATABASE PetCareDB;
USE PetCareDB;
```

- 🔹 What it does
  - Creates a database named PetCareDB
  - Selects it so all tables are created inside it
- 🔹 Why needed
  - Keeps your pet care system separate and organized
  - Avoids mixing tables with other projects

---

2️⃣ Customers Table

```
CREATE TABLE Customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    phone VARCHAR(20),
    email VARCHAR(100),
    address VARCHAR(255)
);
```

- 🔹 What it stores

Information about pet owners

- 🔹 Key points
  - customer_id → Primary Key (PK) (unique customer)
  - AUTO_INCREMENT → MySQL generates ID automatically
- 🔹 Real-life meaning

👉 One customer can own multiple pets and place multiple orders

---

3️⃣ Pets Table

```sql
CREATE TABLE Pets (
    pet_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    pet_name VARCHAR(50),
    species VARCHAR(50),
    breed VARCHAR(50),
    date_of_birth DATE,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

- 🔹 What it stores

Details of each pet

- 🔹 Relationship
  - customer_id → Foreign Key (FK)
  - Links Pets → Customers
- 🔹 Relationship type

👉 One Customer → Many Pets

- 🔹 Example

Customer John owns 2 dogs → both pets reference the same customer_id

---

4️⃣ Vets Table

```sql
CREATE TABLE Vets (
    vet_id INT AUTO_INCREMENT PRIMARY KEY,
    vet_name VARCHAR(100),
    specialization VARCHAR(100)
);
```

- 🔹 What it stores

Information about veterinarians

- 🔹 Usage
  - Used in Appointments
  - Used in Medical_Records
- 🔹 Example

Dr. Smith – Surgery
Dr. Lee – Dermatology

---

5️⃣ Appointments Table

```sql
CREATE TABLE Appointments (
    appointment_id INT AUTO_INCREMENT PRIMARY KEY,
    pet_id INT,
    vet_id INT,
    appointment_date DATE,
    appointment_time TIME,
    FOREIGN KEY (pet_id) REFERENCES Pets(pet_id),
    FOREIGN KEY (vet_id) REFERENCES Vets(vet_id)
);
```

- What it stores

Scheduled vet visits

- Relationships
  - pet_id → which pet
  - vet_id → which vet
- Relationship type
  - One pet → many appointments
  - One vet → many appointments

---

6 Medical_Records Table

```sql
CREATE TABLE Medical_Records (
    record_id INT AUTO_INCREMENT PRIMARY KEY,
    pet_id INT,
    vet_id INT,
    diagnosis TEXT,
    treatment TEXT,
    FOREIGN KEY (pet_id) REFERENCES Pets(pet_id),
    FOREIGN KEY (vet_id) REFERENCES Vets(vet_id)
);
```

- What it stores

Medical history of pets

- Why separate from Appointments?
  - Appointments = schedule
  - Medical records = actual treatment & diagnosis
- Example

Pet: Bruno
Diagnosis: Ear infection
Treatment: Antibiotics

---

## 7️⃣ Orders Table

```sql
CREATE TABLE Orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    total_amount DECIMAL(10,2),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
```

🔹 What it stores

Each purchase made by customers

🔹 Relationship

👉 One customer → many orders

🔹 Example

Customer buys pet food + shampoo

---

## 8️⃣ Categories Table

```sql
CREATE TABLE Categories (
    category_id INT AUTO_INCREMENT PRIMARY KEY,
    category_name VARCHAR(100)
);
```

🔹 What it stores

Groups of products

🔹 Example

- Food
- Medicine
- Accessories

---

## 9️⃣ Suppliers Table

```sql
CREATE TABLE Suppliers (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,
    supplier_name VARCHAR(100)
);
```

🔹 What it stores

Companies that supply products

🔹 Example

Royal Canin

Pedigree

---

🔟 Products Table

```
CREATE TABLE Products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    category_id INT,
    supplier_id INT,
    FOREIGN KEY (category_id) REFERENCES Categories(category_id),
    FOREIGN KEY (supplier_id) REFERENCES Suppliers(supplier_id)
);
```

🔹 What it stores

Actual items sold

🔹 Relationships

- Product → Category
- Product → Supplier

🔹 Relationship type

- One category → many products
- One supplier → many products

---

1️⃣1️⃣ Order_Items Table

```
CREATE TABLE Order_Items (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    price DECIMAL(10,2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

🔹 Why this table?

Orders & Products have a many-to-many relationship

🔹 What it does

👉 Breaks an order into individual items

🔹 Example

Order #5
– Dog food ×2
– Shampoo ×1

---

## 1️⃣2️⃣ Employees Table

```
CREATE TABLE Employees (
    employee_id INT AUTO_INCREMENT PRIMARY KEY,
    employee_name VARCHAR(100)
);
```

- 🔹 What it stores

Staff who perform services

- 🔹 Used in
  - Service bookings

---

## 1️⃣3️⃣ Services Table

```
CREATE TABLE Services (
    service_id INT AUTO_INCREMENT PRIMARY KEY,
    service_name VARCHAR(100)
);
```

- 🔹 What it stores

Non-product services

- 🔹 Example
  - Grooming
  - Vaccination
  - Training

---

## 1️⃣4️⃣ Service_Bookings Table

```
CREATE TABLE Service_Bookings (
    booking_id INT AUTO_INCREMENT PRIMARY KEY,
    pet_id INT,
    service_id INT,
    employee_id INT,
    FOREIGN KEY (pet_id) REFERENCES Pets(pet_id),
    FOREIGN KEY (service_id) REFERENCES Services(service_id),
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id)
);
```

- 🔹 What it stores

Which pet received which service, and who provided it

- 🔹 Relationship

👉 Pet → Service → Employee

---

1️⃣5️⃣ Payments Table

```
CREATE TABLE Payments (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    payment_date DATE,
    amount DECIMAL(10,2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);
```

- 🔹 What it stores

Payment details for orders

- 🔹 Why separate table?
  - Supports multiple payments per order
  - Easier for accounting

**The relationships among tables:**

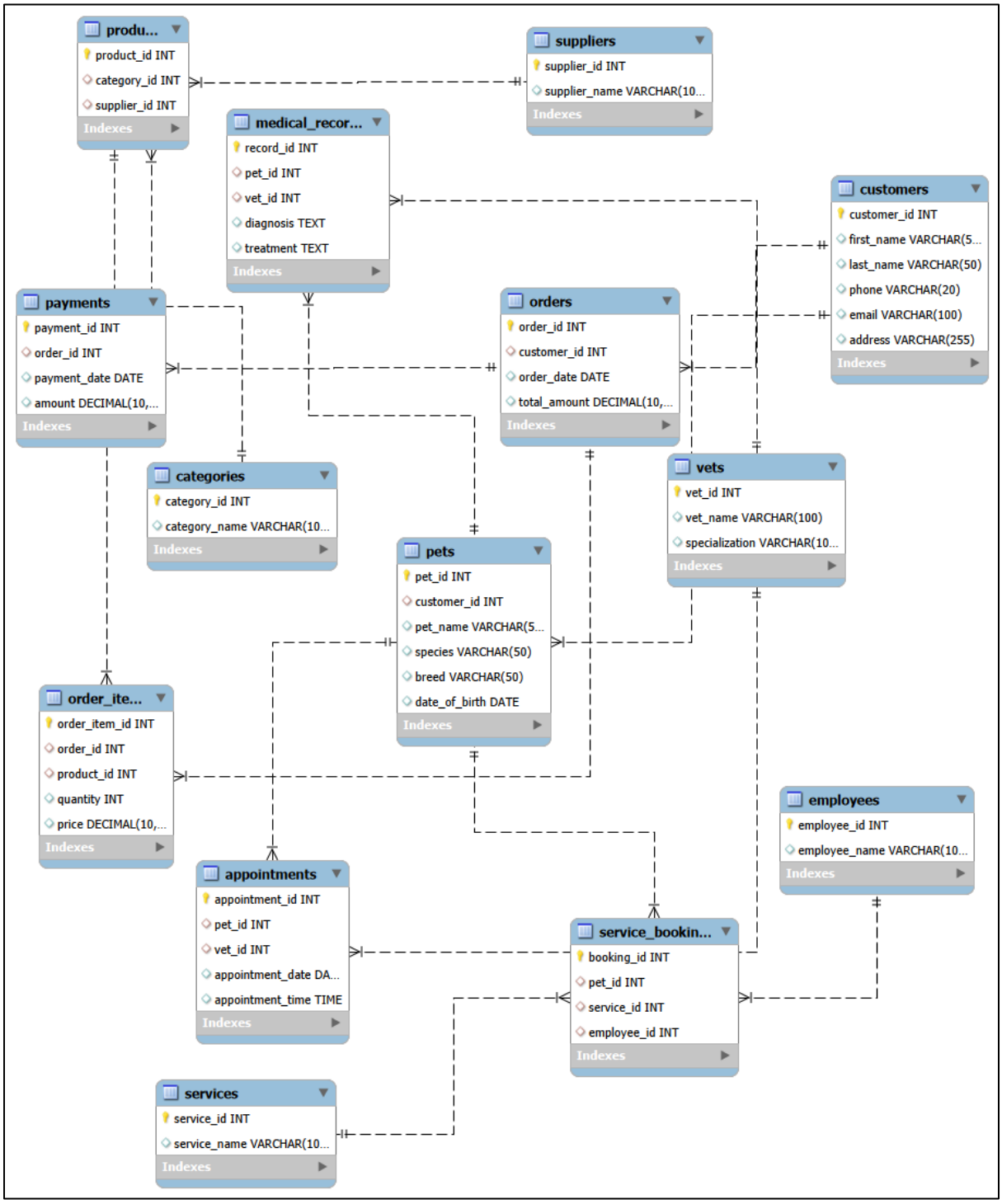
| No. | Parent Table | Child Table | Type | Relationship Description |
|---|---|---|---|---|
| 1 | Customers | Pets | 1:M | One customer can own multiple pets, but each pet belongs to only one customer. |
| 2 | Customers | Orders | 1:M | A customer can place multiple orders, while each order is placed by a single customer. |
| 3 | Pets | Appointments | 1:M | A pet can have many appointments, but each appointment is scheduled for one pet. |
| 4 | Vets | Appointments | 1:M | A veterinarian can attend multiple appointments, but each appointment is handled by one vet. |
| 5 | Pets | Medical_Records | 1:M | A pet can have multiple medical records, each recording diagnosis and treatment details. |
| 6 | Vets | Medical_Records | 1:M | A veterinarian can create many medical records, but each record is created by one vet. |
| 7 | Orders | Order_Items | 1:M | An order can contain multiple order items, but each order item belongs to one order. |
| 8 | Products | Order_Items | 1:M | A product can appear in multiple order items, but each order item refers to one product. |
| 9 | Categories | Products | 1:M | A category can include multiple products, while each product belongs to one category. |
| 10 | Suppliers | Products | 1:M | A supplier can supply multiple products, but each product is supplied by one supplier. |
| 11 | Pets | Service_Bookings | 1:M | A pet can have multiple service bookings, but each booking is for one pet. |
| 12 | Services | Service_Bookings | 1:M | A service can be booked multiple times, but each booking refers to one service. |
| 13 | Employees | Service_Bookings | 1:M | An employee can handle multiple service bookings, but each booking is handled by one employee. |
| 14 | Orders | Payments | 1:M | An order can have one or more payments, while each payment is linked to a single order. |

Fig: Relationships Among Database Tables

**Generate ER Diagram in MySQL Workbench:**

1. Go to the top menu → **Database** → **Reverse Engineer…**
2. **Select your database connection** → Click **Next**
3. Choose **PetCareDB** from the list of schemas → Click **Next**
4. Workbench will retrieve tables, views, and foreign keys → Click **Next**
5. Click **Next** again → Finish

**ER Diagram *(Practical)*:**




**<u>Normalization check:</u>**

**Step 1: First Normal Form (1NF)**

Requirements for 1NF:

1. Every table has a primary key. ✅ Your tables do (e.g., customer_id, pet_id, order_id).

2. All columns are atomic (no repeating groups or arrays). ✅ All columns hold single values, e.g., first_name, quantity, price.
3. No duplicate rows. ✅ Your inserts are unique.

✅ Conclusion: Your database is in 1NF.

---

**Step 2: Second Normal Form (2NF)**

Requirements for 2NF:

1. Must be in 1NF. ✅ Already satisfied.
2. Every non-key column must be fully functionally dependent on the primary key (i.e., no partial dependency on part of a composite key).

Analysis:

- Most tables have single-column primary keys, so full dependency is naturally satisfied.
- Order_Items has order_item_id as primary key (single column). Columns order_id, product_id, quantity, price all depend on order_item_id. ✅ Correct.
- Pets table: customer_id is foreign key, but each column depends on pet_id. ✅ Correct.

✅ Conclusion: Your database is in 2NF.

---

**Step 3: Third Normal Form (3NF)**

Requirements for 3NF:

1. Must be in 2NF. ✅ Already satisfied.
2. No transitive dependencies: non-key columns cannot depend on another non-key column.

Check for transitive dependencies:

- Customers table: phone, email, address depend directly on customer_id, not on first_name or last_name. ✅ OK
- Pets table: pet_name, species, breed, date_of_birth depend on pet_id (not on customer_id). ✅ OK
- Orders table: customer_id is foreign key; order_date and total_amount depend on order_id. ✅ OK
- Order_Items: quantity and price depend on order_item_id. ✅ OK

**Literature Review**

Database Management Systems (DBMS) are essential for efficiently storing, managing, and retrieving structured data. Elmasri and Navathe (2016) explain that relational database design represents real-world entities as tables and uses primary and foreign keys to maintain data integrity. This approach forms the basis of the PetCareDB project, where entities such as customers, pets, veterinarians, and orders are organized in a relational structure.

Normalization is a fundamental concept in database design. According to Date (2004), normalization reduces data redundancy and prevents data anomalies. In this project, normalization up to the Third Normal Form (3NF) is applied to ensure efficient data organization and maintainability.

MySQL is a widely used open-source relational database system that supports SQL, foreign key constraints, and indexing (Oracle, 2023). PetCareDB utilizes MySQL and MySQL Workbench to implement relational constraints and generate ER diagrams for validating database relationships.
Overall, existing literature supports the use of relational databases and ER modeling for service-based systems such as veterinary clinics. The design of PetCareDB aligns with these established principles, ensuring reliability, scalability, and data integrity.

**Testing queries:**

```sql
-- Show all pets with their owners
SELECT Pets.pet_name, Customers.first_name, Customers.last_name
FROM Pets
JOIN Customers ON Pets.customer_id = Customers.customer_id;

-- Show appointments with vets
SELECT Appointments.appointment_date, Appointments.appointment_time, Pets.pet_name, Vets.vet_name
FROM Appointments
JOIN Pets ON Appointments.pet_id = Pets.pet_id
JOIN Vets ON Appointments.vet_id = Vets.vet_id;
```

**Technology required:**

1. **DBMS:** MySQL (Community Edition) – to create and manage the database.
2. **Database Client/IDE:** MySQL Workbench – for writing SQL scripts and visualizing ER diagrams.
3. **Programming Environment (Optional):** Python, Java, C#, or PHP – to connect and interact with the database.
4. **Tools:** Text editor (VS Code, Notepad++) and Excel/Google Sheets for sample data.
5. **System Requirements:** Modern OS (Windows/Linux/macOS), 4 GB RAM, 1 GB free disk space.

**Setup the database on local:**

**Step 1: Clone the Repository**
Open Command Prompt / Terminal and run:
git clone < https://github.com/Sha-bab/Pet-Care-Veterinary-Clinic-DBM/blob/main/PetCareDB.sql>

---

**Step 2: Open MySQL Workbench**

1. Launch MySQL
2. Workbench.
3. Connect to your local MySQL server.

---

**Step 3: Open SQL Script**

1. In Workbench, go to File → Open SQL Script.
2. Select the .sql file you cloned from Git (usually PetCareDB.sql).

---

**Step 4: Run the Script**

1. Click the "Execute" ( ⚡ ) button to run the script.
2. This will:
    - Create the PetCareDB database
    - Create all tables
    - Insert sample data

**References:**

[1] Elmasri, R., & Navathe, S. (2016). *Fundamentals of Database Systems*. Pearson Education.

[2] Date, C. J. (2004). *An Introduction to Database Systems*. Addison-Wesley.

[3] Chen, P. P. (1976). *The Entity-Relationship Model—Toward a Unified View of Data*. ACM.

[4] Oracle Corporation. (2023). *MySQL Documentation*.