DATABASE DESIGN FOR BEIRUTPETS PET STORE

by SQL Squad

Submitted by:

- Jalal Al Arab
- Karim Khalil
- Mamdouh Dabjan
- Kareem Mbayid
- Mohammad Ramadan

1. Introduction:

In Lebanon, where rich culture and warm hospitality define everyday life, pet ownership is becoming increasingly common. However, many pet owners face challenges in accessing quality pet care, veterinary services, and essential pet supplies. The growing need for a well-organized and efficient pet care system has highlighted the importance of structured data management in the industry.

To address this need, BEIRUTPETS PET STORE is developing a comprehensive database system to enhance pet care services and improve customer experiences. This system is designed to efficiently store key aspects of pet ownership, including customer registrations, pet health records, appointments, product orders, and employee operations.

The project follows a structured timeline, beginning with the design of the ER model, followed by a detailed report outlining the database structure, a working demo, and progress updates. Through this initiative, the goal is to create a well-organized, customer-friendly system that transforms pet care management in Lebanon, making it easier for pet owners to provide the best for their beloved companions.

2-ER Diagram Symbols:

An Entity Relationship (ER) Diagram is a visual representation of the relationships between a distinct yet related set of entities that form the core of a database system.

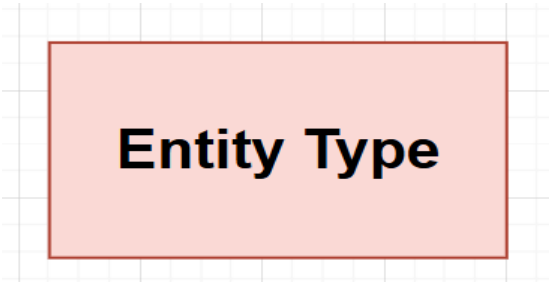
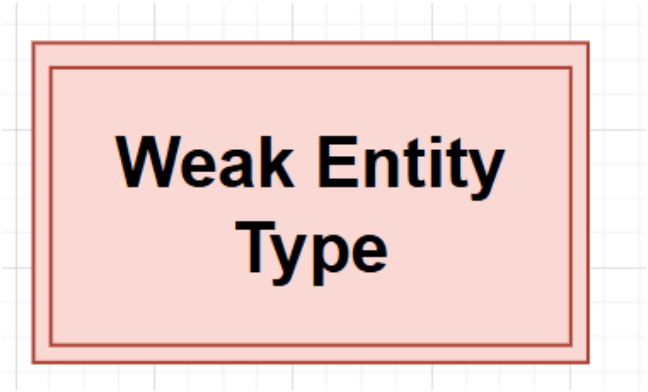
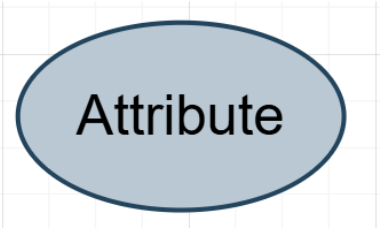
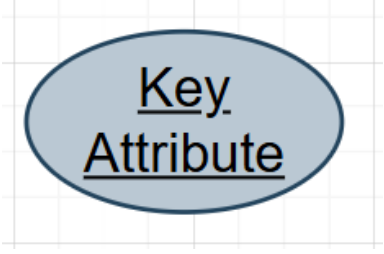
This diagram serves as a valuable tool for explaining the logical structure of databases and is based on three fundamental concepts: entities, attributes, and relationships.

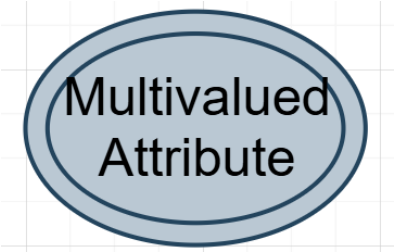
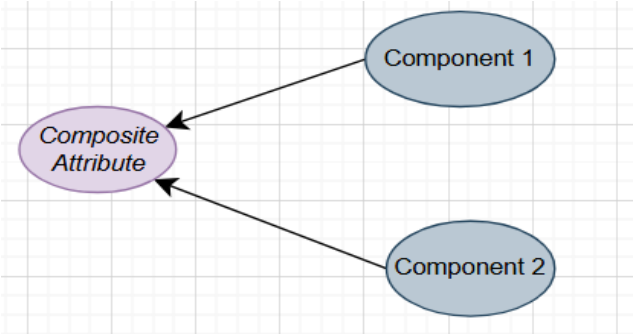
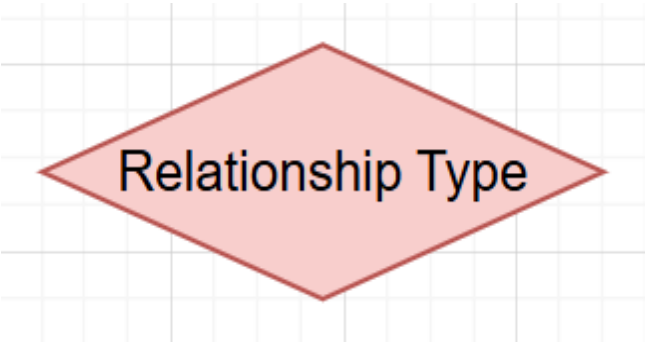
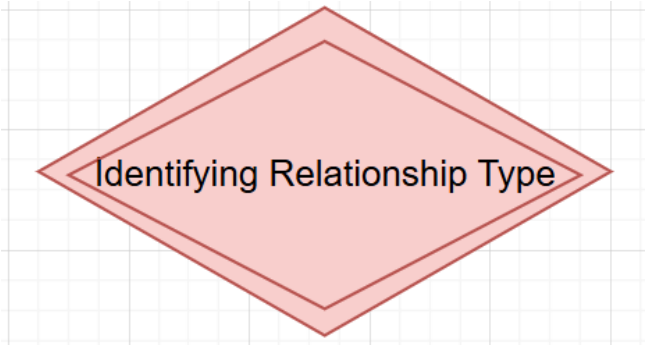
To represent these concepts in an ER diagram, specific symbols are used:

- Rectangles: Entities are represented by rectangles in an ER diagram. Each rectangle corresponds to a distinct entity in the database system.
- Ovals: Attributes, which define the characteristics or properties of entities, are represented by ovals in the diagram. Each oval represents an attribute associated with a particular entity
- Diamond Shapes: Relationships between entities are illustrated using diamond shapes in the ER diagram. These diamonds symbolize the connections or associations between different entities.

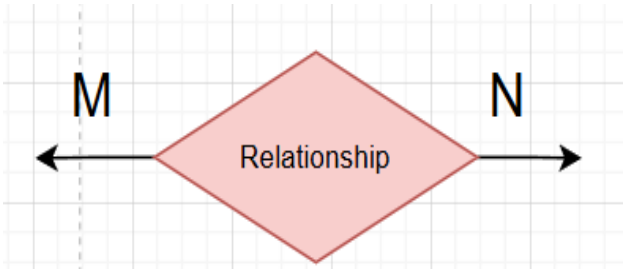
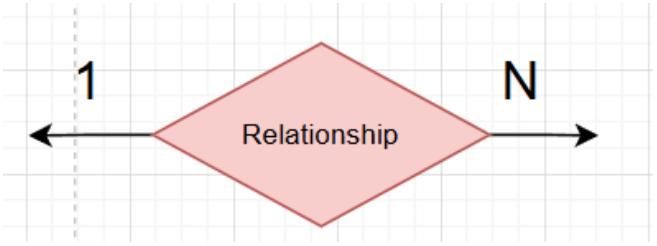
Demonstration:

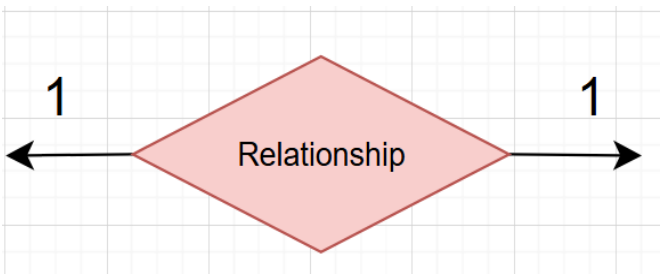
The primary objective of an ER diagram is to provide a visual starting point for the design of a database. To facilitate understanding and comprehensiveness of our ER diagram, we have employed the use of different colors for each symbol based on the following criteria. We are pleased with the outcome and hope that you will find it equally satisfying.

Concept	Symbol
Entity Type	
Weak Entity Type	
Attribute	
key Attribute	

Concept	Symbol
Multivalued Attribute	
Composite attribute	
Relationship Type	
Identifying Relationship Type	

Cardinality and Participation:

Content	Symbol
Total participation	
Partial participation	
Many to Many	
One to Many	

One to One	

2. System Description:

Customers

A customer is a person who owns one or more pets and interacts with the pet care organization. Customers can register in the system, schedule appointments for their pets, place orders for products, and provide feedback on services. Importantly, a customer exists independently of any particular service usage; for example, a customer may be registered in the database even if they have not yet scheduled an appointment.

Pets

A pet is a living entity uniquely identified (e.g., by a Pet ID) and characterized by attributes such as Name, Age, Breed, and Gender. Pets are always associated with their owners (customers) and may have related health records or pet insurance. Once a pet is registered in the database, its health record will be recorded, meaning a pet cannot be registered in the database without its health record. However, pets may or may not be enrolled in health insurance.

Health Records

Health records store detailed medical or wellness information for a pet (for instance, check-ups, vaccinations, and treatments). They include attributes like the record description and the date last updated. A pet has to have a health record even if no medical event has occurred.

Pet Insurance

Pet insurance represents an optional financial service that covers a pet's medical costs. Attributes include the insurance type, coverage dates, and the provider. A pet can exist with or without insurance, and thus the pet insurance entity shows partial participation—it is defined and available in the database independently of whether a pet is currently insured. Furthermore, an insurance's start date cannot be later than its end date chronologically.

Appointments

Appointments are scheduled events during which a pet receives a service (such as a veterinary check-up, grooming, or vaccination) administered by an employee. One key point is that the underlying services (like "grooming" or "check-up") are defined in the system independently of

any appointment. This means that even if a particular service is not booked for an appointment at a given time, it still exists as an offering.

Services

Services represent the various offerings available to customers—such as grooming, medical check-ups, boarding, and vaccinations. A service is part of the database even when no appointment is scheduled for it. This independent existence emphasizes that while an appointment may utilize a service, the service itself participates partially in appointments; its existence is not dependent on an appointment being scheduled.

Orders

An order represents a request from a customer for pet-related products. Orders capture details like order ID and date, and they are processed against the customer's account. Every order has a customer associated with it.

Feedback

Feedback allows customers to evaluate services they have received by providing ratings and comments. Since feedback is optional, not every appointment or service results in customer feedback, but every feedback is provided by a customer. Feedback cannot be uniquely identified by its attributes and has to rely on customer (a Strong entity type) to do so.

Employees

Employees are the people working within the organization to provide pet-related services. They have attributes such as name, contact information, date of joining, and salary. Employees may be involved in scheduling or conducting appointments, processing orders, and managing accounts. Their involvement in any given service or event is optional—for example, not every employee needs to participate in every appointment.

Suppliers

Suppliers are external entities that provide products (like pet food, accessories, or medical supplies) needed by the organization. They participate in the ordering process, yet they are maintained as independent records in the database regardless of current orders, demonstrating another case of partial participation.

CheckInOut

A check-in records the date and time an employee checks in each day. A check-out records the date and time an employee check out each day. Check-in and Check-out events for every employee. A check in record cannot exist without a corrosponding employee, making this entity type a weak entity type

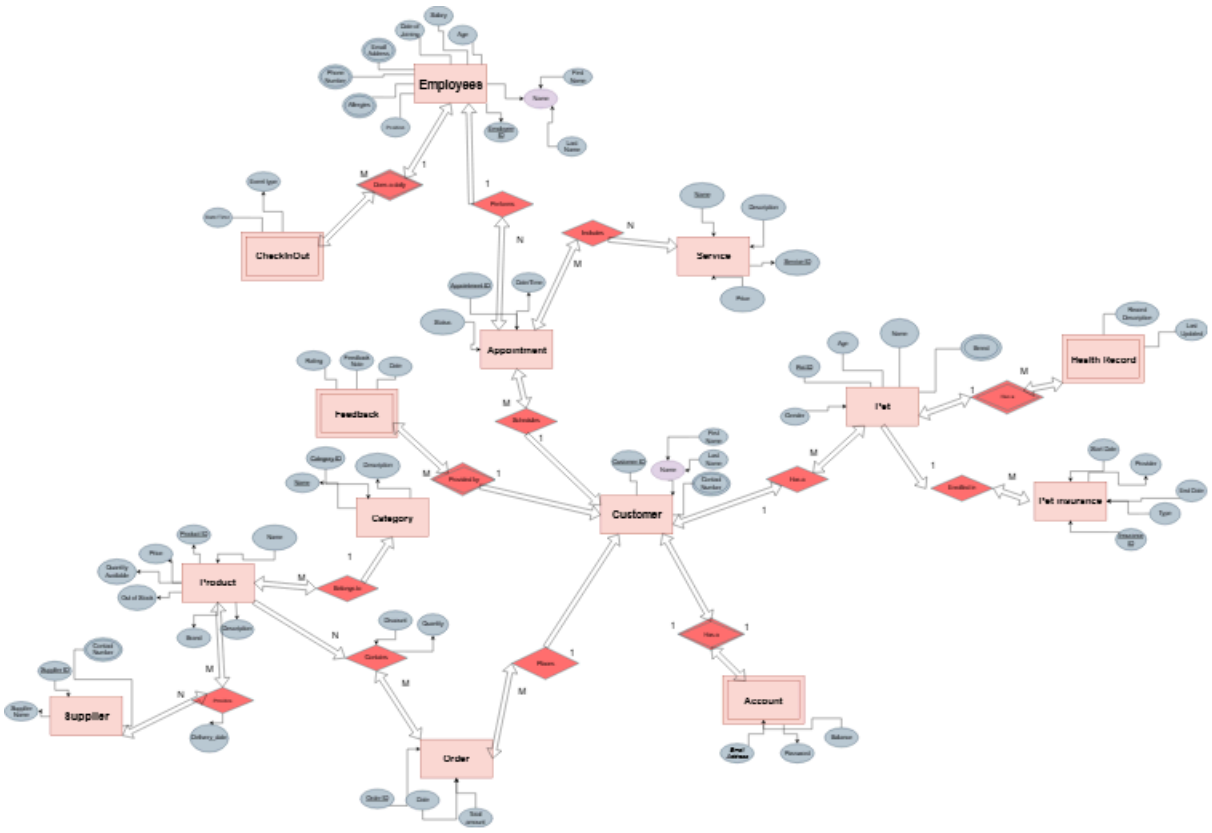
Category

A category is used to organize products into meaningful groups based on shared attributes such as type, purpose, or brand. Each category is defined by a unique identifier and descriptive name, and it helps facilitate search, filtering, and reporting on the product catalog. Categories exist independently of individual products—meaning a category is maintained in the system even if there are no products currently assigned to it.

Product

A product represents an item offered by the organization—such as pet food, accessories, or toys—that is available for purchase. Each product is uniquely identified (e.g., by a Product ID) and characterized by attributes like name, description, price, and inventory levels. Importantly, a product exists in the catalog regardless of whether it is currently part of an order. Additionally, each product must be assigned a category when registering it to the database.

3-Entity Relation Diagram for BeirutPets Pet Store:



If anyone encounters difficulty while zooming in and reading the entities, there's no need to be concerned. We will provide a link for direct access, where you can zoom in without any loss of resolution. Additionally, we will thoroughly go through each entity in the discussion below:

Untitled Diagram (8).drawio

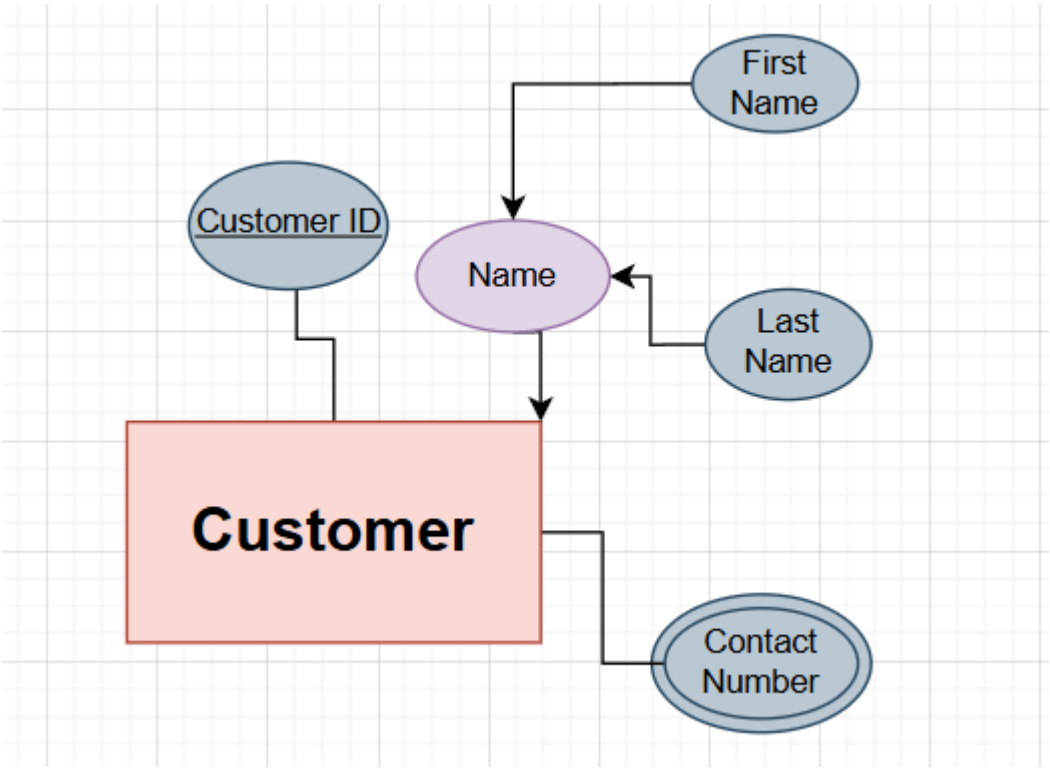
4-Entity types:

- **Customer:**

Customer_ID: A unique identifier for each customer, e.g. a numeric or alphanumeric code (e.g., 001).

Name: A set of characters indicating the customer's given name made up of their first and last name.

Phone Number (optional): A set of digits for direct contact.



- **Pet:**

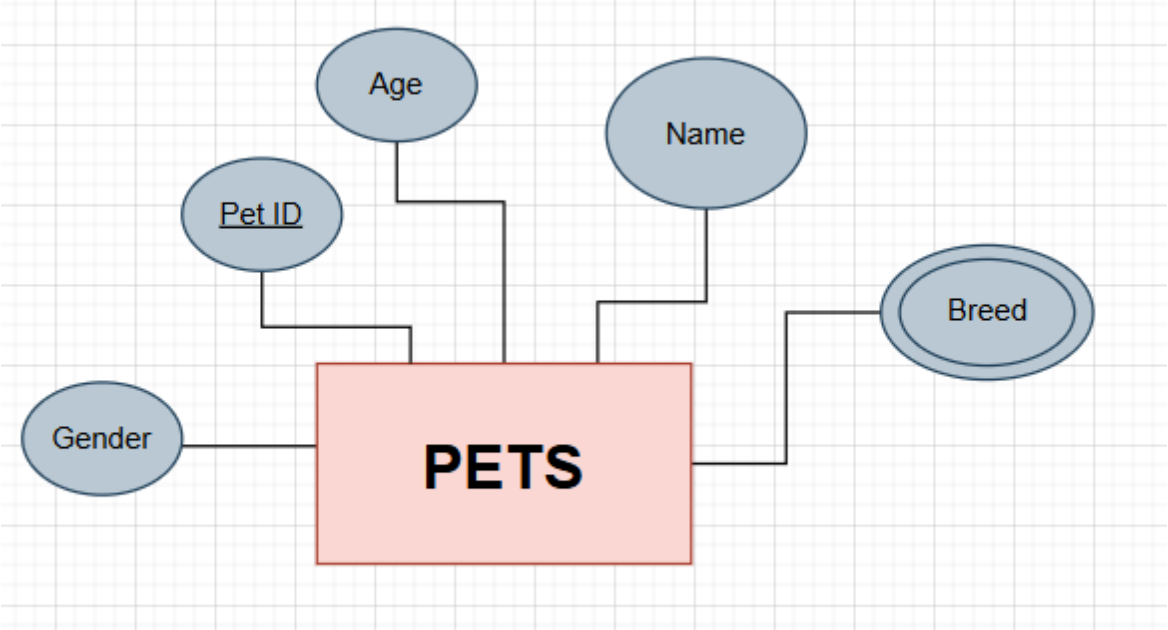
Pet ID: A key attribute used as a unique identifier for each pet (e.g., 100).

Name: A set of characters denoting the pet's name.

Breed: A set of characters describing the pet's breed (e.g., "Golden Retriever"). a pet can have multiple breeds.

Age: An integer indicating the pet's age (e.g., in years).

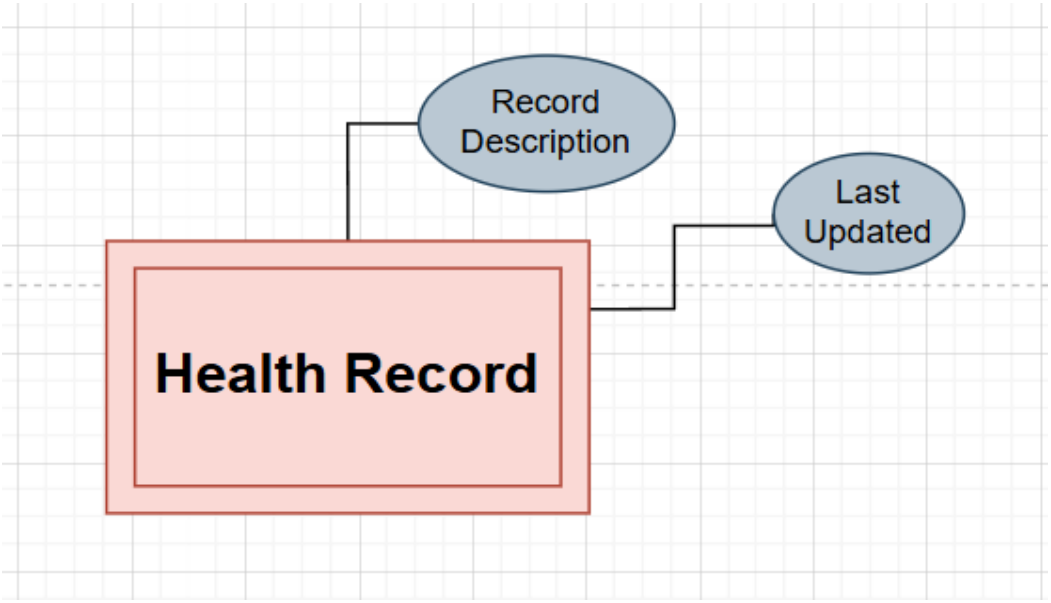
Gender: A set of characters indicating the pet's gender (e.g., "Male," "Female," "Unknown").



- **Health record:**

Record Description: A set of characters summarizing the medical findings or notes.

Last Updated: A date/time field indicating the most recent update (e.g., YYYY-MM-DD HH:MM).



- **Employees:**

Employee ID: A key attribute used as a unique identifier for each employee (e.g., 001).

Name: A set of characters indicating an employee's first and last name.

Date_of_Joining: A date field representing when the employee started (e.g., YYYY-MM-DD).

Salary: A numeric or decimal field for the employee's salary.

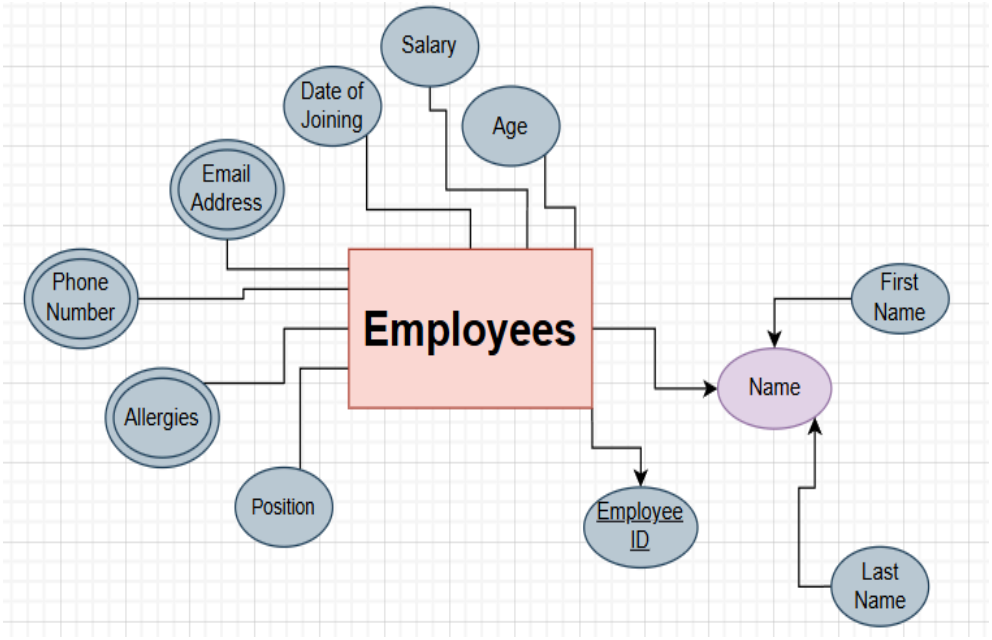
Position: A set of characters describing the job title.

Age: A number indicating the employee's age

Email Address: A set of characters indicating the employee's email address. An employee can have multiple email

Phone number: An Integer indicating an employee's phone number(s)

Allergies: A set of characters indicating if an employee is allergic to certain pets. An employee can be allergic to more than one pet



- **Pet Insurance:**

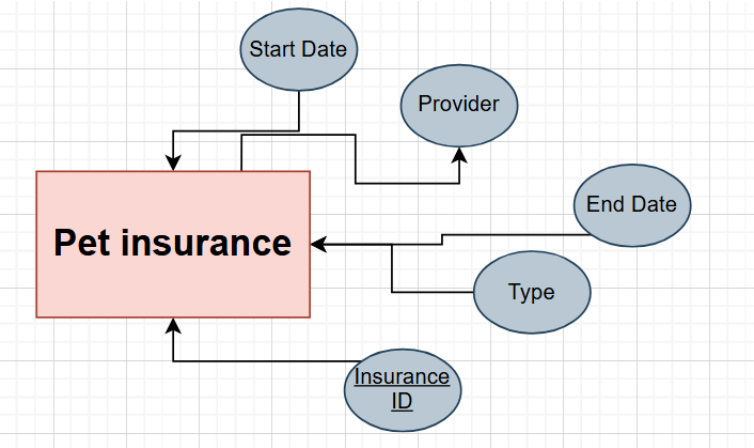
Insurance ID: A key attribute used as a unique identifier for each policy (e.g., INS-100).

Type: A set of characters describing the insurance type (e.g., "Comprehensive," "Accident Only").

Provider: A set of characters indicating the company or entity providing the insurance.

End Date: The policy's end date (e.g., YYYY-MM-DD).

Start Date: The policy's start date

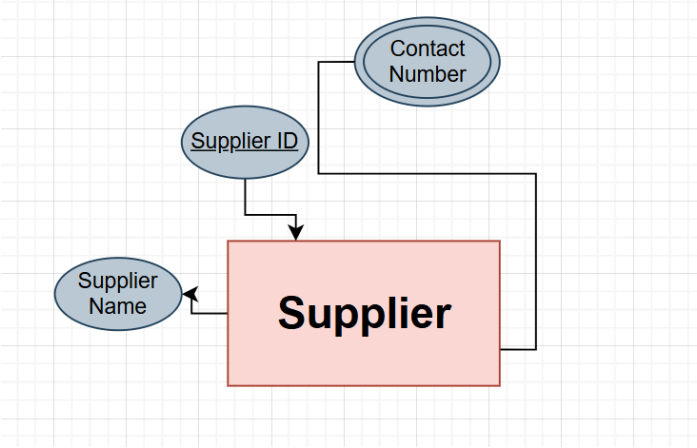


- **Supplier**:

Supplier ID: A key attribute used as a unique identifier

Supplier Name: A set of characters describing the supplier's name.

Contact Number: A set of digits for direct communication. A supplier can have multiple phone numbers

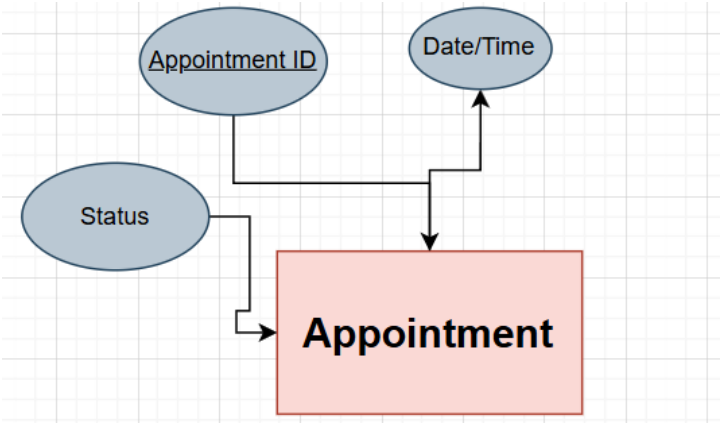


- **Appointment**:

  Appointment ID: A key attribute used as a unique identifier

  Date Time: The scheduled date and time (e.g., YYYY-MM-DD HH:MM).

  Status: A set of characters indicating whether the appointment is “Scheduled,” “Completed,” or “Canceled.”

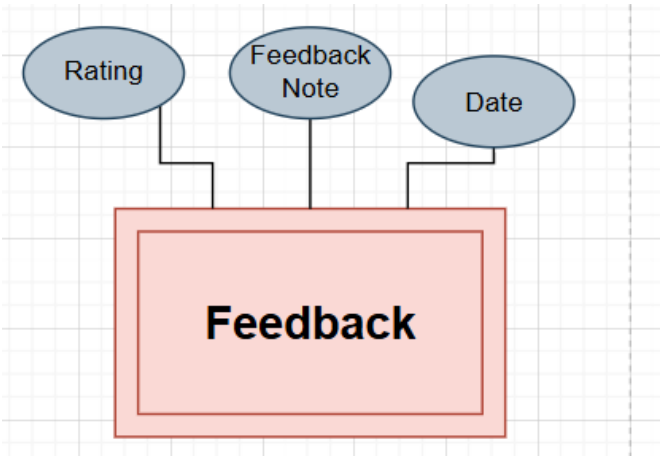


- **Feedback:**

  Date: The date the feedback was provided (e.g., YYYY-MM-DD).

  Rating: A numeric scale indicating satisfaction (e.g., 1–5 stars).

  Feedback_Note (optional): A set of characters describing the customer’s comments.

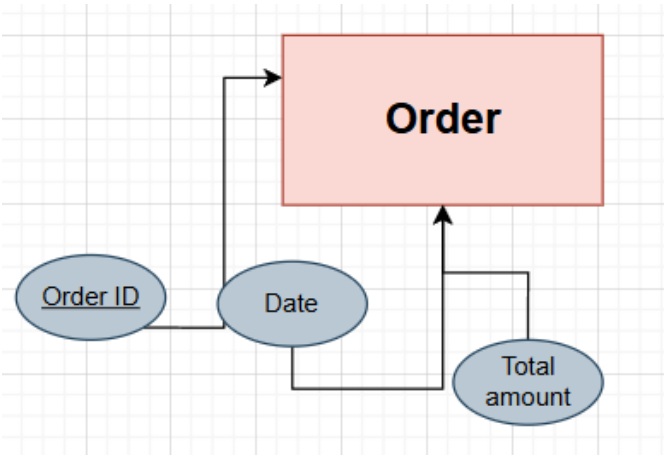


- **Order:**

  Order ID: An key attribute as a unique identifier

  Date: The date the order was placed (e.g., YYYY-MM-DD).

  Total Amount: A numeric or decimal field for the total cost.



- **Product:**

  Product ID: A key attribute used as a unique identifier

  Name: A set of characters describing the product's name (e.g., "Deluxe Dog Food").

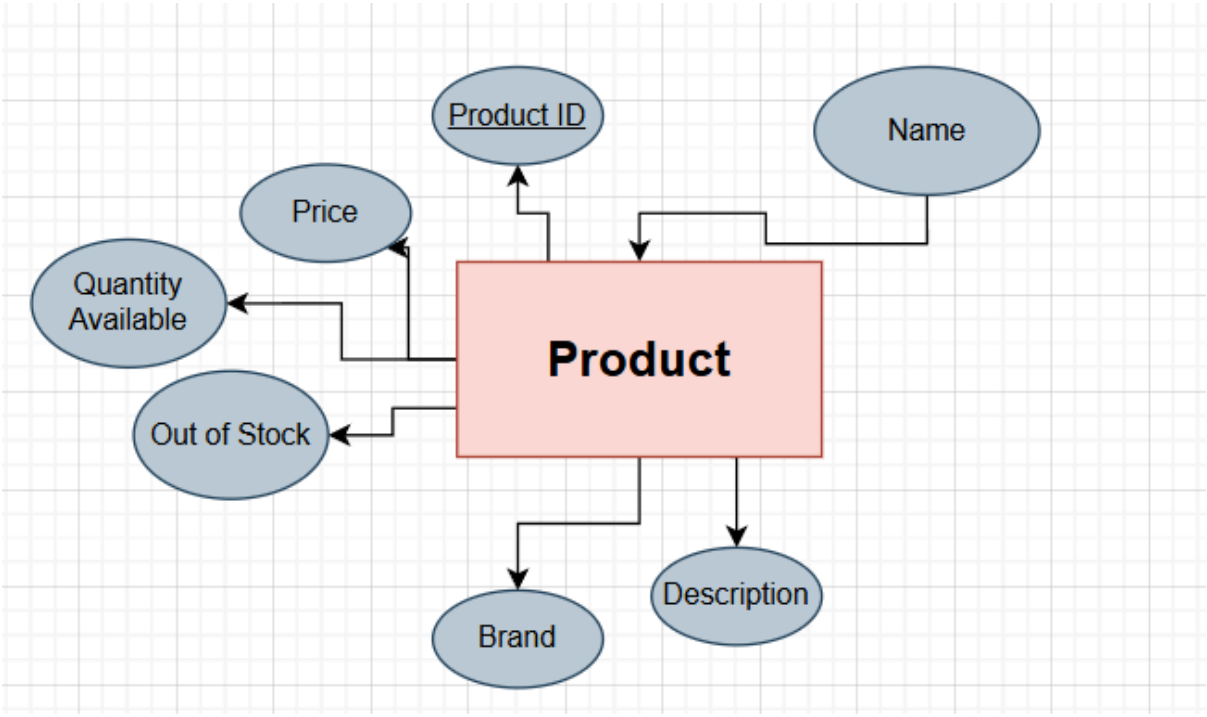
  Description (optional): A set of characters detailing product features or ingredients.

Price: A numeric or decimal field indicating the product's cost.

Quantity Available: An integer indicating how many units are available. The default value is 0

Out of Stock: A boolean indicating if a product is out of stock or not. The default value is 0

Brand: A set of characters describing a product's brand

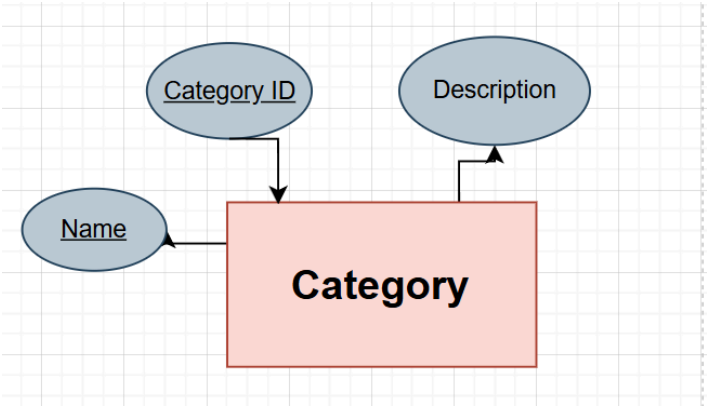


- **Category:**

Category ID: A key attribute

Category Name: A key attribute consisting of a set of characters describing the category (e.g., "Pet Toys").

Description (optional): Additional details about the category's nature or purpose.

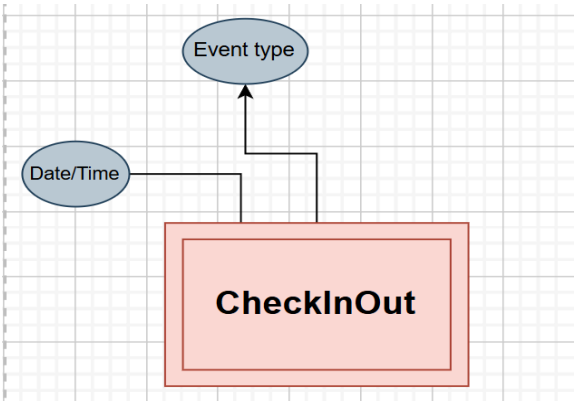


- **CheckInOut:**

  Date Time: The date and time of the check-in (e.g., YYYY-MM-DD HH:MM).

  Status:

  Event type: A set of characters indicating whether the employee checked-in or out



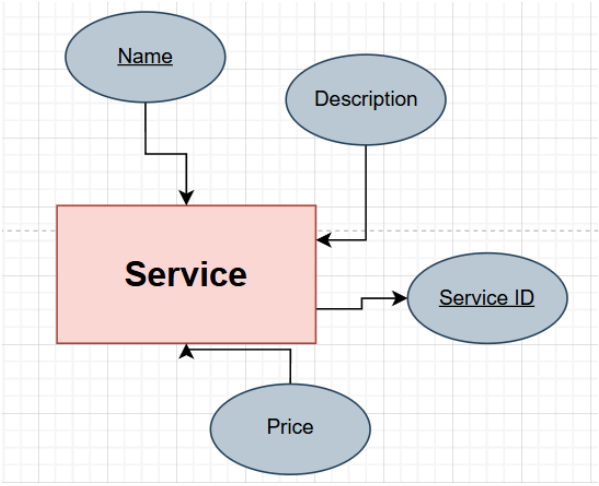
- **Service:**

  Service ID: A key attribute used as a unique identifier

  Name: A key attribute consisting of a set of characters indicating the name (e.g., “Basic Grooming”). Two services cannot have the same name.

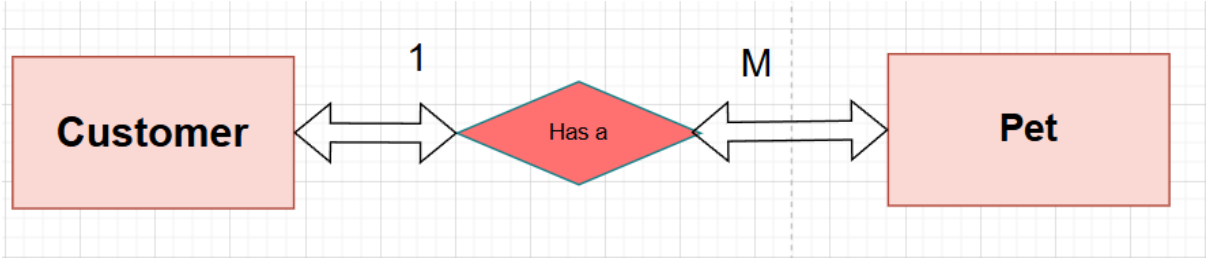
  Description (optional): A set of characters providing details about the service.

Price: A numeric or decimal field representing the standard cost of the service.

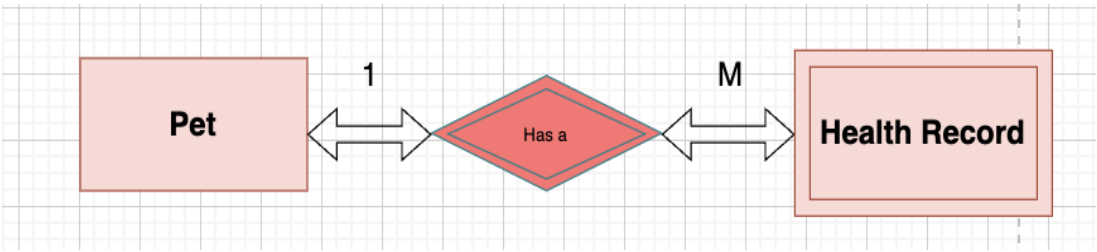


5-Relationships:

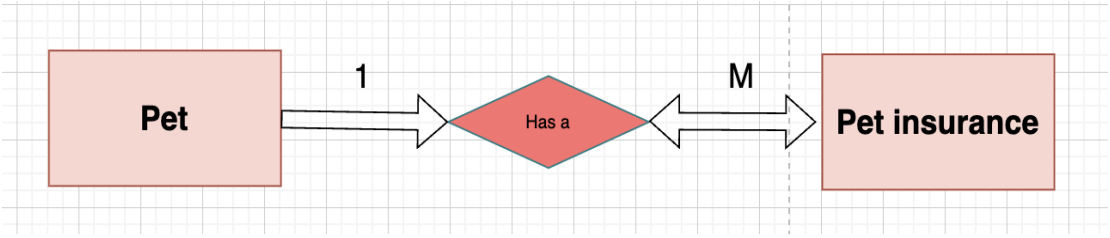
- **Customer-Has-Pet:** The *"Customer-Has-Pet"* relationship associates customers with the pets they own. It represents a one-to-many relationship: one customer can own multiple pets, but each pet is linked to a single customer. This relationship is crucial for tracking ownership details, ensuring personalized pet care, and maintaining clear records of which customer is responsible for each pet.



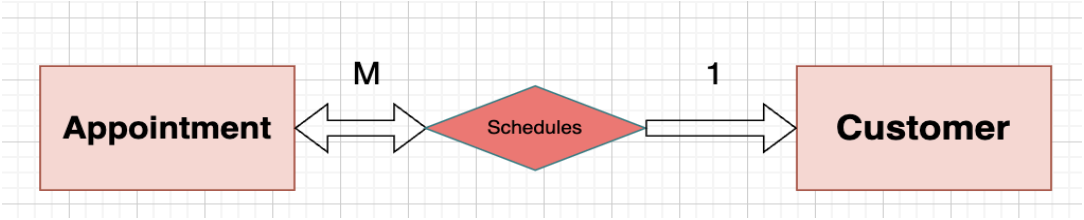
- **Pet-Has-HealthRecord:** The "Pet-Has-HealthRecord" relationship links each pet to one or more health records. This is a one-to-many relationship: a single pet can accumulate multiple records over time (e.g., for vaccinations, check-ups, and treatments). This relationship is fundamental to storing medical histories and supporting proper veterinary care.



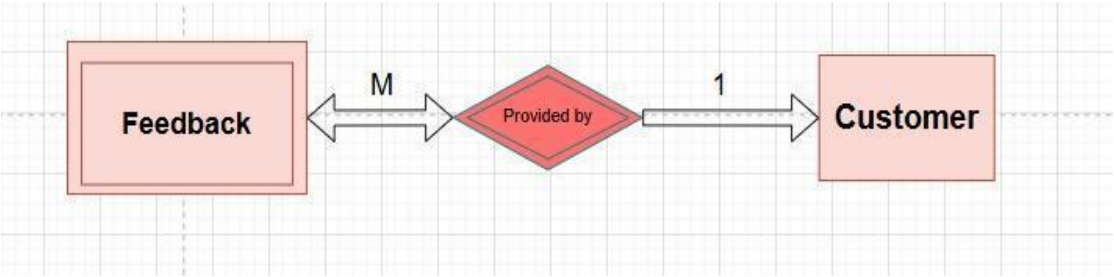
- **Pet-Has-Insurance**: The "Pet-Has-Insurance" relationship connects pets to their insurance policies. This relationship is one-to-many (a pet can hold multiple policies over time). It is vital for handling coverage details, claims, and premium payments.



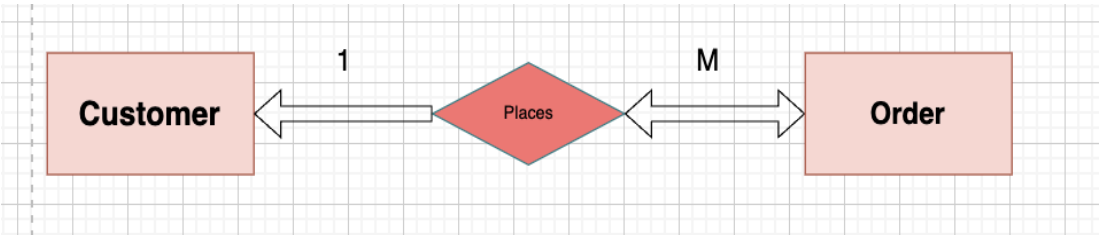
- **Customer-Schedules-Appointment**: The "Customer-Schedules-Appointment" relationship indicates that a customer can book multiple appointments, while each appointment is tied to exactly one customer. This is a one-to-many relationship, ensuring that each appointment record has a clear "owner" for billing, scheduling, and notification purposes.



- **Customer-Provides-Feedback**: The "Customer-Provides-Feedback" relationship associates a customer with any feedback or review they submit. A one-to-many relationship: a single customer can provide multiple feedback entries. This is key for quality improvement and measuring customer satisfaction.



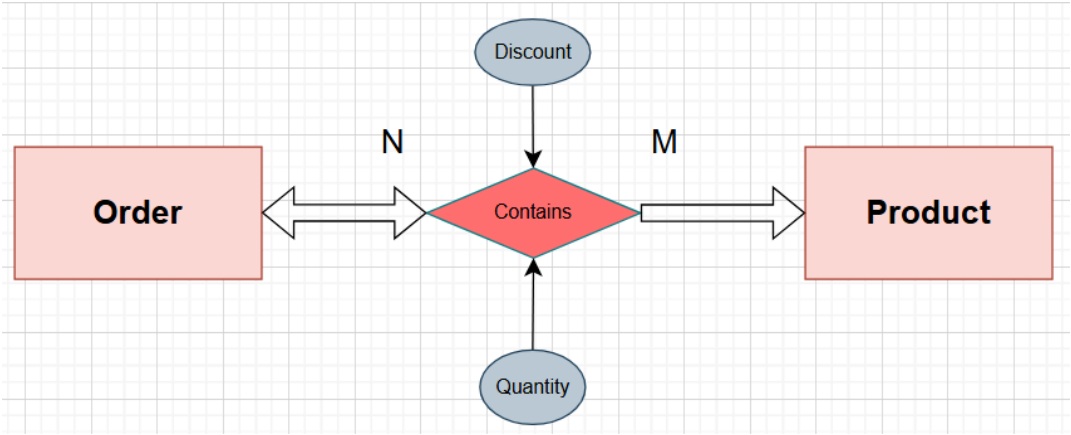
- **Customer-Places-Order**: The "Customer-Places-Order" relationship connects customers to their orders. This is a one-to-many relationship, where a single customer can place numerous orders, and each order belongs to exactly one customer. It underpins the purchasing workflow and enables order tracking.



- **Order-Contains-Product**: The "Order-Contains-Product" relationship captures which products are included in each order. modeled as a many-to-many relationship (one order can have multiple products, and a product can appear in multiple orders), it may require an intersection/bridge entity (e.g., "Order Details") to store quantity and discount information. It is vital for inventory control and sales analytics.

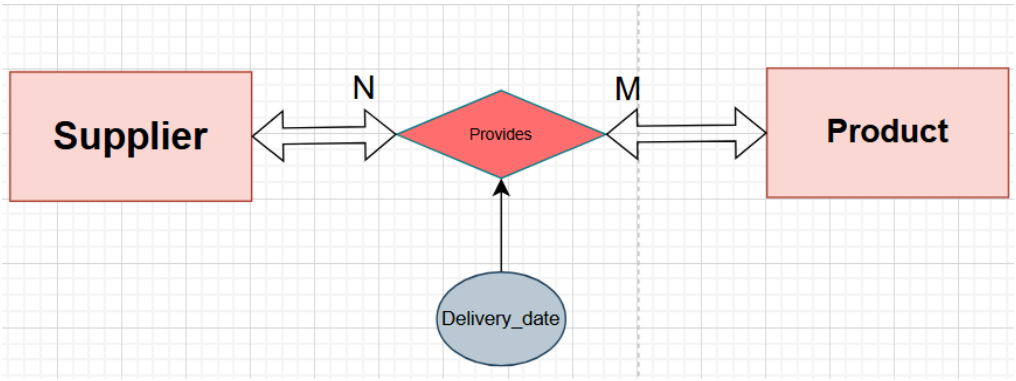
  Discount is an attribute describing if there was a discount on the product

  Quantity is an attribute describing how many items were ordered of that product

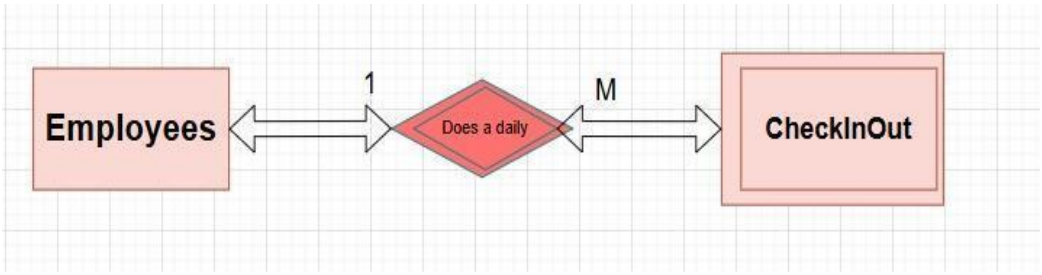


- **Supplier-Provides-Product**: The "Supplier-Provides-Product" relationship links external suppliers to the products they offer. It is a many-to-many (a product may have multiple suppliers). This relationship is critical for managing supply chains, restocking, and vendor relationships.

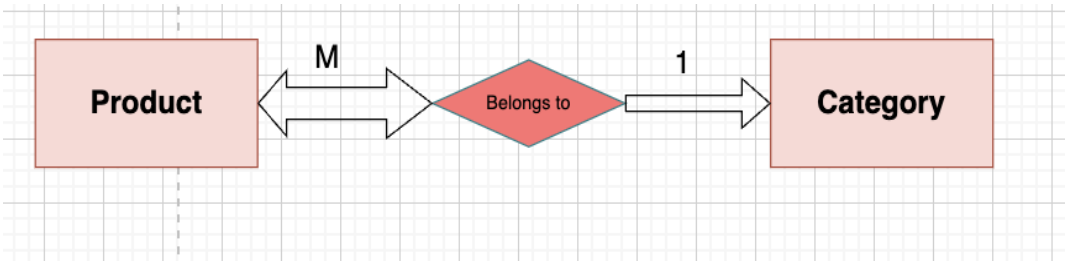
  Delivery date is an attribute describing the date on which the supplier provided the product



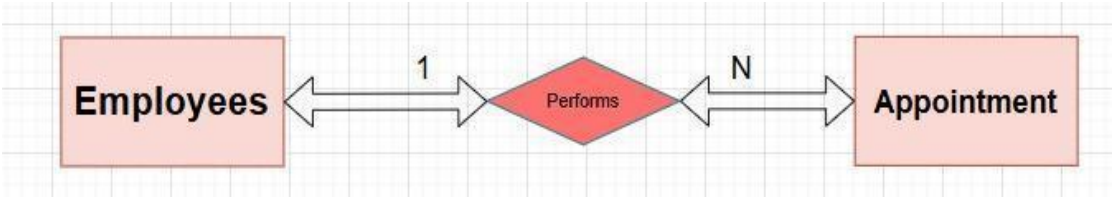
- **Employee-does-a-daily-checkin/out**: This relationship records each check-in/out event a one-to-many relationship: a employee can have many check-in/out entries over time, while each check-in entry pertains to one pet. It is essential for attendance and daily logs.



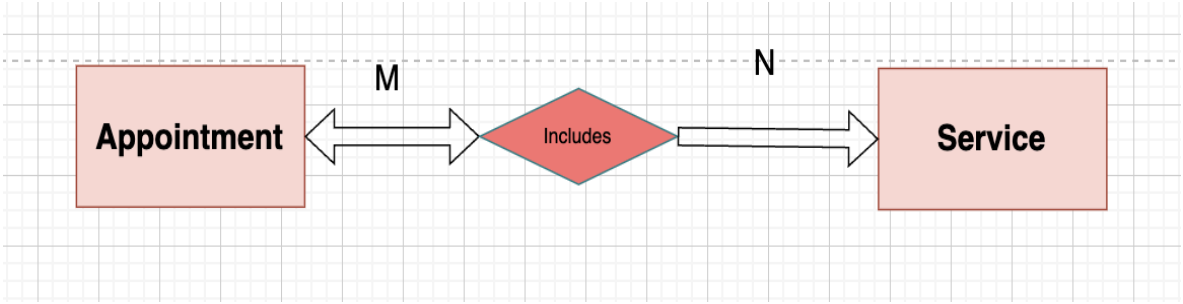
- **Product-belongs-to-Category:** This relationship places products into one category for better organization and searching. This is a one-to-one (each product belongs to exactly one category). It underpins product catalog management and filtering.



- **Employee-Performs-Appointment**: The "Employee-Performs-Appointment" relationship indicates which employee is responsible for carrying out or assisting with an appointment. This can be one-to-many (one employee per appointment. It ensures accurate scheduling, task assignment, and service accountability.



- **Appointment-Includes-Service**: The "Appointment-Includes-Service" relationship establishes which services are provided during a given appointment. It represents a many-to-many relationship, meaning a single appointment can involve multiple services (e.g., grooming, vaccination), and a single service (such as "Grooming") can appear in many different appointments. This relationship is crucial for detailing the scope of each appointment and accurately tracking which services were rendered



6- ER to Relational Mapping Algorithm:

After completing the ER diagram for the database, the entities, attributes, and relationships need to be mapped into a lower-level relational schema following the steps illustrated in class. The steps go as follows:

Step 1: Start by transforming all non-weak entities into relational tables. If any entities have composite attributes, we have to break them down into individual attributes. When there are multiple candidate keys, designate one as the primary key for the table.

Step 2: Proceed to map all weak entities, making sure to include the primary key of the owning entity in the weak entity's table.

Step 3: Next, we have to deal with binary 1:1 relationship types. Convert them into relational tables using the foreign key approach, where the primary key of one entity becomes a foreign key in the other entity's table.

Step 4: Handle binary 1:N relationship types in a similar manner. Create relational tables and apply the foreign key approach to represent these relationships.

Step 5: For binary M:N relationship types, generate a new relational table to represent the relationship. This new table will contain foreign keys that link the primary keys of the participating entities.

Step 6: Address multivalued attributes in the final step. Create new tables for these attributes, including their sub-attributes, and connect them to the primary key of the entity where they are located.

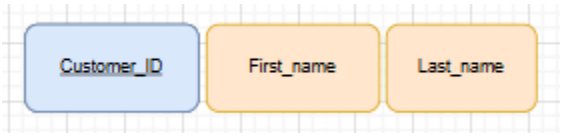
Step 7: If the diagram includes relationships between more than two entities, like ternary or n-ary relationships, you would create dedicated relational tables for these relationships. These tables would typically include foreign keys that link to the primary keys of the participating entities.

Note: In the upcoming steps, Primary Keys will be colored as Blue, while Foreign Keys will be colored as Green

Step01: Mapping the regular entity types:

The first step includes mapping regular entities (those with primary keys) into relations. Each entity is mirrored via a table relation that contains all its simple attributes and a single chosen primary key that is underlined. Most entities in the airport database are regular entities, and they include:

1-CUSTOMER:



Entity Overview

- **Candidate Keys:** Customer_ID is chosen as the primary key.

Resulting Table

- **Table Name**: CUSTOMER
- **Primary Key**: Customer_ID
- **Other Attributes**:
    - First_Name
    - Last_Name

2-PET:

Entity Overview

- **Candidate Keys:** Pet_ID is chosen as the Primary Key

Resulting Table

- **Table Name**: PET
- **Primary Key**: Pet_ID
- **Other Attributes**:
    - Name
    - Age
    - Gender

3-SERVICE :

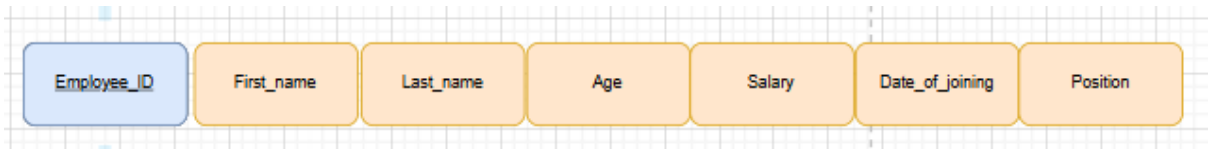
Entity Overview

- **Candidate Keys:** Service_ID is chosen as the Primary Key

Resulting Table

- **Table Name**: SERVICE
- **Primary Key**: Service_ID
- **Other Attributes**:
    - Description (optional)
    - Price
    - Name

4-EMPLOYEE :



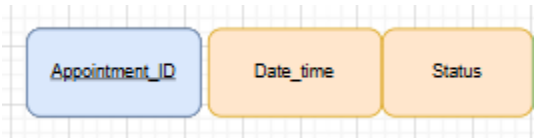
Entity Overview

- **Candidate Key:** Employee_ID is chosen as the Primary Key

Resulting Table

- **Table Name**: EMPLOYEE
- **Primary Key**: Employee_ID
- **Other Attributes**:
    - First_Name
    - Last_Name
    - Age
    - Salary
    - Date_of_Joining
    - Position

5-APPOINTMENT :



Entity Overview.

- **Candidate Key:** Appointment_ID is chosen as the Primary Key

Resulting Table

- **Table Name**: APPOINTMENT
- **Primary Key**: Appointment_ID
- **Other Attributes**:
    - Date_Time
    - Status (e.g., “Scheduled,” “Completed”)

6-ORDER:

Entity Overview

- **Candidate Key:** Order_ID is chosen as the Primary Key

Resulting Table

- **Table Name:** ORDER
- **Primary Key:** Order_ID
- **Other Attributes:**
    - Date
    - Total_Amount

7-PRODUCT:

Entity Overview

- **Candidate Key:** Product_ID is chosen as the Primary Key

Resulting Table

- **Table Name**: PRODUCT
- **Primary Key**: Product_ID
- **Other Attributes**:
    - Name
    - Description *(optional)*
    - Price
    - Quantity_Available
    - Out_of_Stock
    - Brand

8-CATEGORY :

Entity Overview

- **Candidate Key:** Category_ID is chosen as the Primary Key

Resulting Table

- **Table Name**: CATEGORY
- **Primary Key**: Category_ID
- **Other Attributes**:
    - Name
    - Description *(optional)*

9-PET_INSURANCE:

Entity Overview

- **Candidate Key:** Insurance_ID as the primary key.

Resulting Table

- **Table Name**: PET_INSURANCE
- **Primary Key**: Insurance_ID
- **Other Attributes**:
    - Type
    - Provider
    - End_Date
    - Start_Date

10-SUPPLIER:

Entity Overview

- **Candidate Key:** Supplier_ID for the primary key.

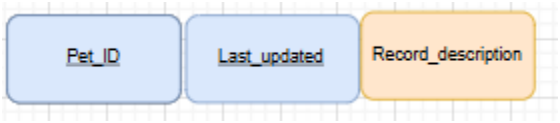
Resulting Table

- **Table Name**: SUPPLIER
- **Primary Key**: Supplier_ID
- **Other Attributes**:
    - Supplier_Name

Step02 : Mapping weak entity types:

For the weak entity types, only simple attributes are used in mapping the relation. Additionally, weak entity relations have a foreign key attribute, which is the primary key of the owning entity. Thus, it can be concluded that the combination of the partial key and the foreign key represents the relation's primary key.

1-HEALTH_RECORD:

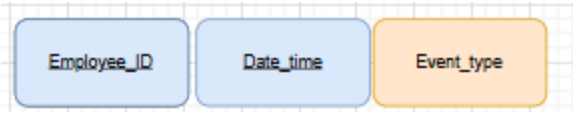


Entity Overview
HEALTH_RECORD is a weak entity because its existence depends on the owning entity, PET. A pet may have multiple health records (for example, for various check-ups or treatments), but a health record does not exist independently of its pet.

Mapping Details

- **Owning Entity:** PET
- **Foreign Key Included:** Pet_ID (from PET)
- **Partial Key:** We assume that the attribute Last_Updated (a date/time stamp) uniquely distinguishes each health record for a given pet.
- **Composite Primary Key**: (Pet_ID, Last_Updated)
- **Other Attributes:**
    - Record_Description

3-CHECK_IN_OUT:



Entity Overview
CHECK_IN_OUT is a weak entity because its records exist only in the context of an owning entity—in this design, it is dependent on **EMPLOYEE.** Every EMPLOYEE has multiple records of checking in and out, but every record is linked to one EMPLOYEE.

Mapping Details

- **Owning Entity**: EMPLOYEE
- **Foreign Key Included**: Employee_ID (from EMPLOYEE)
- **Partial Key**: Date_time.
- **Composite Primary Key**: (Employee_ID, Date_Time)
- **Other Attributes**:
    - Event_type

4-FEEDBACK:

Entity Overview
FEEDBACK is a weak entity because its records exist only in the context of an owning entity—in this design, it is dependent on **CUSTOMER**. Every **CUSTOMER** can provide multiple feedback entries, but each feedback entry is linked to exactly one **CUSTOMER.**

Mapping Details

- **Owning Entity**: CUSTOMER
- **Foreign Key Included**: Customer_ID (from CUSTOMER)
- **Partial Key**: Date
- **Composite Primary Key**: (Customer_ID, Date)
- **Other Attributes**
    - Rating
    - Feedback note

Step03: Mapping Binary 1:1 Relationship Types:

This step includes mapping the binary 1:1 relationship. To do so, we are applying the foreign key approach in which we select the entity with the total participation side on the relation and we call it S. The other participating relation is named T. After that we simply include the primary key of T as a foreign key in S to represent the 1:1 relationship.

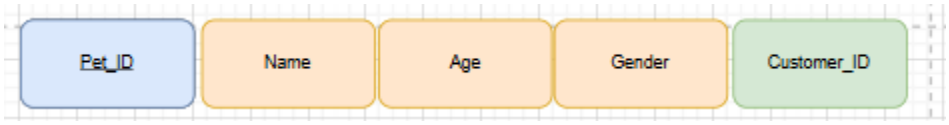
1-CUSTOMER-HAS-ACCOUNT :

Note: This step was already done in the weak entity mapping, so no additional modifications are required

Step04: Mapping Binary 1: N Relationship Types:

In this stage, binary one-to-many relationships are mapped. The entity that has the N on its side is named S while the other participating entity is named T, we then insert the primary key of T as a foreign key in S to represent the 1: N relationship.

1-CUSTOMER-HAS-PET:



Description

The "Customer-Has-Pet" relationship links the entity **CUSTOMER** to the entity **PET**. A single customer (the 1 side) can own many pets (the N side), but each pet belongs to exactly one customer.

Implementation

- Since the N is on the **PET** side, we consider **PET** to be the **S** entity.
- **CUSTOMER** is the **T** entity.
- We insert the primary key of **CUSTOMER** (i.e., **Customer_ID**) as a foreign key into the **PET** table.
- The foreign key is named **Customer_ID** in **PET**.

This ensures that each pet record directly references its owning customer.

2-PET- HAS-PET_INSURANCE:

Description

The “Pet-Has-Insurance” relationship connects the entity **PET** to **PET_INSURANCE**. One pet (the 1 side) can have multiple insurance policies (the N side) over time, but each policy covers exactly one pet.

Implementation

- Since the N is on the **PET_INSURANCE** side, we consider **PET_INSURANCE** to be **S**.
- **PET** is the **T** entity.
- We insert **Pet_ID** (primary key of **PET**) as a foreign key into **PET_INSURANCE**.
- The foreign key is named **Pet_ID** in **PET_INSURANCE**.

This enforces that each insurance policy is tied to a single pet.

3-CUSTOMER-SCHEDULES-APPOINTMENT:

Description

The “Customer-Schedules-Appointment” relationship links the entity **CUSTOMER** to **APPOINTMENT**. A single customer (the 1 side) can schedule multiple appointments (the N side), but each appointment is booked by exactly one customer.

Implementation

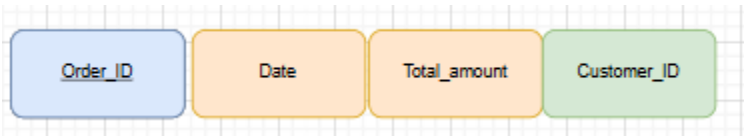
- Since the N is on the **APPOINTMENT** side, we consider **APPOINTMENT** to be **S**.
- **CUSTOMER** is the **T** entity.
- We insert **Customer_ID** (primary key of **CUSTOMER**) as a foreign key into **APPOINTMENT**.
- The foreign key is named **Customer_ID** in **APPOINTMENT**.

This ensures that each appointment record is associated with the correct customer

4-CUSTOMER-PROVIDES-FEEDBACK:

Note: This was already done in the weak entity mapping

5-CUSTOMER-PLACES-ORDER:



Description
The "Customer-Places-Order" relationship connects the entity **CUSTOMER** to **ORDER**. One customer (the 1 side) can place many orders (the N side), but each order is placed by exactly one customer.

Implementation

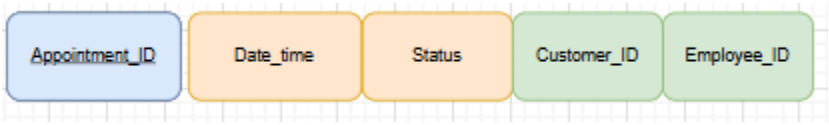
- Since the N is on the **ORDER** side, we consider **ORDER** to be **S**.
- **CUSTOMER** is the **T** entity.
- We insert **Customer_ID** (primary key of **CUSTOMER**) as a foreign key into **ORDER**.
- The foreign key is named **Customer_ID** in **ORDER**.

This ensures each order is associated with the customer who placed it.

6-EMPLOYEE-DOES-A-DAILY-CHECKIN/OUT:

Note: This was already done in the weak entity mapping

7-EMPLOYEE-PERFORMS-APPOINTMENT:



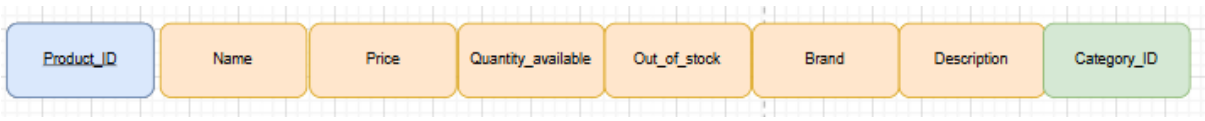
Description
The "Employee-Performs-Appointment" relationship connects the entity **EMPLOYEE** to **APPOINTMENT**. One employee (the 1 side) can handle multiple appointments (the N side), but each appointment is performed by exactly one employee.

Implementation

- Since the N is on the **APPOINTMENT** side, we consider **APPOINTMENT** to be **S**.
- **EMPLOYEE** is the **T** entity.
- We insert **Employee_ID** (primary key of **EMPLOYEE**) as a foreign key into **APPOINTMENT**.
- The foreign key is named **Employee_ID** in **APPOINTMENT**.

This ensures each appointment is tied to the employee responsible for it.

8-PRODUCT-BELONGS-TO-CATEGORY:



Description
The "Product-Belongs-To-Category" relationship places each product in exactly one category for better organization and searching. You noted this as a **one-to-one** relationship: each product belongs to exactly one category, and each category is assigned to exactly one product. This setup underpins product catalog management and filtering.

Implementation

- We treat **PRODUCT** as the entity that will hold the foreign key (S).
- **CATEGORY** is the other entity (T).
- We insert **Category_ID** (the primary key of **CATEGORY**) as a foreign key in the **PRODUCT** table.
- The foreign key is named **Category_ID** in **PRODUCT**.

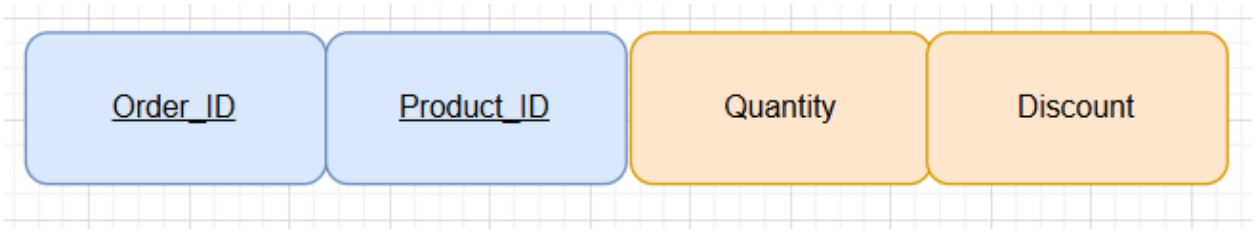
9-PET-HAS-HEALTH_RECORD:

Note: This was already done in the weak entity mapping

Step05: Mapping Binary M: N Relationship Types:

In this step, we are mapping many to many binary relationships. For each N: M relationship we must create a new relation that includes the primary keys of all the participating entities as foreign keys. The totality of all the primary keys will be considered as the primary key for the newly established relation. If the relation has any attributes, those must also be added.

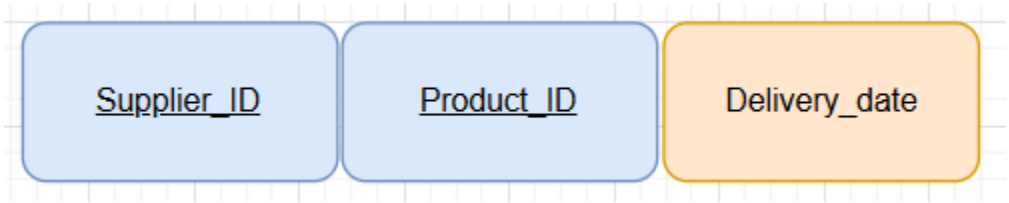
1-OrderProduct:



The **"Order-Contains-Product"** relationship is an N:M relationship that links the **ORDER** entity type to the **PRODUCT** entity type. This allows each order to contain multiple products and each product to be included in multiple orders. The relationship is modeled via the creation of a new relation that includes the primary keys of both **ORDER** and **PRODUCT** as foreign keys.

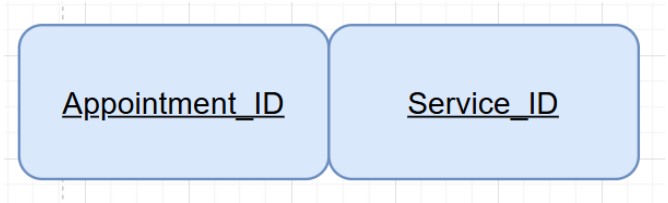
Discount and Quantity are two attributes that add additional information regarding the order

2- SupplierProduct:



The "Supplier-Provides-Product" relationship is an N:M relationship that links the SUPPLIER entity to the PRODUCT entity. This allows each supplier to provide multiple products, and each product may have multiple suppliers. The relationship is modeled via the creation of a new relation that includes the primary keys of both SUPPLIER and PRODUCT as foreign keys.

3-AppointmentService:



The **“Appointment-Includes-Service”** relationship is an N:M relationship that links the **APPOINTMENT** entity to the **SERVICE** entity. This allows each appointment to include multiple services, and each service can be part of multiple appointments. The relationship is modeled via the creation of a new relation that includes the primary keys of both **APPOINTMENT** and **SERVICE** as foreign keys.

Step06: Mapping Multi-Valued Attributes :

At the stage of the mapping, we are dealing with multivalued attributes. Each multivalued attribute is mapped by the creation of a new relation in which we put any associated attributes along with the primary key of the entity on which the multivalued attribute is found. The primary key to the new relation is the combination of all the attributes.

1-Employee Phone Number :

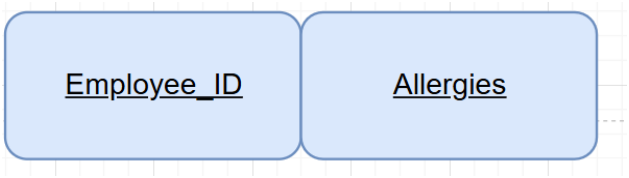
The multivalued attribute **Phone Number** is found in the **EMPLOYEE** entity. It is represented as a relation in which the combination of all the attributes makes up the primary key. This relation holds the primary key **Employee_ID** of the strong entity **EMPLOYEE** in addition to the **Phone_Number** attribute, which together form the primary key. This relation represents the different phone numbers that an employee can have.

EMPLOYEE_PHONE_NUMBER:

2-Employee Allergies :

The multivalued attribute **Allergies** is found in the **EMPLOYEE** entity. It is represented as a relation that contains the primary key **Employee_ID** of the strong entity **EMPLOYEE**, along with the **Allergy** attribute, both forming the primary key. This relation records the different allergies that an employee has, which may be relevant for workplace safety.

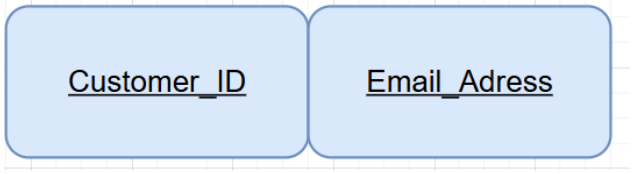
EMPLOYEE_ALLERGIES:



3-Customer Email Address:

The multivalued attribute **Email Address** is found in the **CUSTOMER** entity. It is represented as a relation that contains the **Customer_ID** from the **CUSTOMER** entity and the **Email_Address** attribute, both forming the primary key. This relation stores multiple email addresses that a customer may use for communication.

CUSTOMER_EMAIL_ADDRESS:



4- Supplier Phone Number :

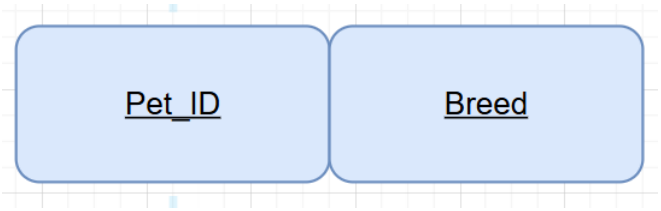
The multivalued attribute **Phone Number** is found in the **SUPPLIER** entity. It is represented as a relation that contains the **Supplier_ID** from the **SUPPLIER** entity and the **Phone_Number** attribute, both forming the primary key. This relation records the different phone numbers through which a supplier can be contacted.

SUPPLIER_PHONE_NUMBER:

5-Pet Breed:

The multivalued attribute **Breed** is found in the **PET** entity. It is represented as a relation that contains the **Pet_ID** from the **PET** entity and the **Breed** attribute, both forming the primary key. This relation accounts for pets that may belong to multiple breeds or mixed breeds.

PET_BREED:



Step07: Mapping of N-ary Relationship Types :

In this final step, we specify the mapping of N-ary relationships connecting more than one entity. We represent our only ternary relationships below and show how they contain foreign keys, the primary keys of each entity it is connecting, and, in addition; the relationship attributes themselves.

Since we don't have any N-ary relationships, we will skip this step.

8- Database Creation:

- CUSTOMER Table

Purpose: Stores information about customers who own pets and use the pet store services.

Description:

- Each customer has a unique ID
- Stores basic customer information (first and last name)
- Customers can own multiple pets (handled in PET table)
- Customers can schedule appointments, place orders, and provide feedback

```
CREATE TABLE CUSTOMER (
    Customer_ID NUMBER PRIMARY KEY,
    First_Name VARCHAR2(50) NOT NULL,
    Last_Name VARCHAR2(50) NOT NULL
);
```

- **PET Table**

Purpose: Stores information about pets owned by customers.

Description:

- Each pet has a unique ID
- Stores pet details (name, age, gender)
- Linked to a single customer (owner)
- Pets can have health records and optional insurance
- Gender is restricted to specific values (Male, Female, Unknown)

```
CREATE TABLE PET (
    Pet_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(50) NOT NULL,
    Age NUMBER,
    Gender VARCHAR2(10) CHECK (Gender IN ('Male', 'Female', 'Unknown')),
    Customer_ID NUMBER NOT NULL,
    CONSTRAINT fk_pet_customer FOREIGN KEY (Customer_ID) REFERENCES
CUSTOMER(Customer_ID)
```

);

- **HEALTH_RECORD Table (Weak Entity)**

Purpose: Stores medical records for pets.

Description:

- Weak entity dependent on PET
- Composite primary key (Pet_ID + Last_Updated)
- Each pet must have at least one health record
- Stores medical descriptions and update timestamps

```
CREATE TABLE HEALTH_RECORD (
   Pet_ID NUMBER NOT NULL,
   Last_Updated TIMESTAMP NOT NULL,
   Record_Description VARCHAR2(1000),
   CONSTRAINT pk_health_record PRIMARY KEY (Pet_ID, Last_Updated),
   CONSTRAINT fk_health_pet FOREIGN KEY (Pet_ID) REFERENCES PET(Pet_ID)
ON DELETE CASCADE
);
```

- **PET_INSURANCE Table**

Purpose: Stores insurance information for pets.

Description:

- Each policy has a unique ID
- Linked to a specific pet
- Stores insurance details (type, provider, coverage dates)
- Includes constraint to ensure start date is before end date

```
CREATE TABLE PET_INSURANCE (
```

```
    Insurance_ID NUMBER PRIMARY KEY,
    Type VARCHAR2(50) NOT NULL,
    Provider VARCHAR2(100) NOT NULL,
    Start_Date DATE NOT NULL,
    End_Date DATE NOT NULL,
    Pet_ID NUMBER NOT NULL,
    CONSTRAINT fk_insurance_pet FOREIGN KEY (Pet_ID) REFERENCES
PET(Pet_ID),
    CONSTRAINT chk_insurance_dates CHECK (Start_Date <= End_Date)
);
```

- **SERVICE Table**

Purpose: Stores services offered by the pet store.

Description:

- Each service has a unique ID
- Service names must be unique
- Stores service details and pricing
- Services can be included in multiple appointments

```
CREATE TABLE SERVICE (
    Service_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(100) NOT NULL UNIQUE,
    Description VARCHAR2(500),
    Price NUMBER(10,2) NOT NULL CHECK (Price >= 0)
);
```

- **EMPLOYEE Table**

Purpose: Stores information about pet store employees.

Description:

- Each employee has a unique ID
- Stores employee details (name, salary, join date)
- Employees can perform appointments and have check-in/out records
- Multivalued attributes handled in separate tables

```
CREATE TABLE EMPLOYEE (
    Employee_ID NUMBER PRIMARY KEY,
    First_Name VARCHAR2(50) NOT NULL,
    Last_Name VARCHAR2(50) NOT NULL,
    Age NUMBER,
    Salary NUMBER(10,2) CHECK (Salary >= 0),
    Date_of_Joining DATE NOT NULL,
    Position VARCHAR2(50) NOT NULL
);
```

- **CHECK_IN_OUT Table (Weak Entity)**

Purpose: Records employee check-in and check-out times.

Description:

- Weak entity dependent on EMPLOYEE
- Composite primary key (Employee_ID + Date_Time)
- Tracks when employees check in and out
- Event type restricted to 'Checkin' or 'Checkout'

```
CREATE TABLE CHECK_IN_OUT (
    Employee_ID NUMBER NOT NULL,
    Date_Time TIMESTAMP NOT NULL,
```

Event_Type VARCHAR2(10) CHECK (Event_Type IN ('Checkin', 'Checkout')),

CONSTRAINT pk_check_in_out PRIMARY KEY (Employee_ID, Date_Time),

CONSTRAINT fk_check_employee FOREIGN KEY (Employee_ID) REFERENCES EMPLOYEE(Employee_ID) ON DELETE CASCADE

);

- **APPOINTMENT Table**

Purpose: Stores scheduled appointments for pet services.

Description:

- Each appointment has a unique ID
- Linked to both a customer and an employee
- Stores appointment details (date/time, status)
- Status restricted to specific values
- Services for appointments handled in junction table

CREATE TABLE APPOINTMENT (

Appointment_ID NUMBER PRIMARY KEY,

Date_Time TIMESTAMP NOT NULL,

Status VARCHAR2(20) CHECK (Status IN ('Scheduled', 'Completed', 'Canceled')),

Customer_ID NUMBER NOT NULL,

Employee_ID NUMBER NOT NULL,

CONSTRAINT fk_appointment_customer FOREIGN KEY (Customer_ID) REFERENCES CUSTOMER(Customer_ID),

CONSTRAINT fk_appointment_employee FOREIGN KEY (Employee_ID) REFERENCES EMPLOYEE(Employee_ID)

);

- **APPOINTMENT_SERVICE Table (Junction Table)**

Purpose: Links appointments to services (M:N relationship).

Description:

- Composite primary key (Appointment_ID + Service_ID)
- Represents which services are included in each appointment
- An appointment can include multiple services
- A service can be part of multiple appointments

```
CREATE TABLE APPOINTMENT_SERVICE (
    Appointment_ID NUMBER NOT NULL,
    Service_ID NUMBER NOT NULL,
    CONSTRAINT pk_appointment_service PRIMARY KEY (Appointment_ID, Service_ID),
    CONSTRAINT fk_as_appointment FOREIGN KEY (Appointment_ID) REFERENCES APPOINTMENT(Appointment_ID) ON DELETE CASCADE,
    CONSTRAINT fk_as_service FOREIGN KEY (Service_ID) REFERENCES SERVICE(Service_ID) ON DELETE CASCADE);
```

- **FEEDBACK Table (Weak Entity)**

Purpose: Stores customer feedback about services.

Description:

- Weak entity dependent on CUSTOMER
- Composite primary key (Customer_ID + Feedback_Date)
- Stores ratings (1–5) and optional comments
- Feedback is optional (not all appointments have feedback)

```
CREATE TABLE FEEDBACK (
    Customer_ID NUMBER NOT NULL,
```

```
    Feedback_Date DATE NOT NULL,
    Rating NUMBER(1) CHECK (Rating BETWEEN 1 AND 5),
    Feedback_Note VARCHAR2(500),
    CONSTRAINT pk_feedback PRIMARY KEY (Customer_ID, Feedback_Date),
    CONSTRAINT fk_feedback_customer FOREIGN KEY (Customer_ID)
REFERENCES CUSTOMER(Customer_ID) ON DELETE CASCADE
);
```

- **CUSTOMER_ORDERS Table**

Purpose: Stores customer orders for products.

Description:

- Each order has a unique ID
- Linked to a customer
- Stores order date and calculated total amount
- Order items handled in junction table
- Total amount defaults to 0 and must be non-negative

```
CREATE TABLE CUSTOMER_ORDERS (
    Order_ID NUMBER PRIMARY KEY,
    Order_Date DATE NOT NULL,
    Total_Amount NUMBER(10,2) DEFAULT 0 CHECK (Total_Amount >= 0),
    Customer_ID NUMBER NOT NULL,
    CONSTRAINT fk_order_customer FOREIGN KEY (Customer_ID) REFERENCES
CUSTOMER(Customer_ID)
);
```

- **CATEGORY Table**

Purpose: Organizes products into categories.

Description:

- Each category has a unique ID
- Stores category name and optional description
- Products must belong to exactly one category
- Categories exist independently of products

```
CREATE TABLE CATEGORY (
    Category_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(50) NOT NULL,
    Description VARCHAR2(500)
);
```

- **PRODUCT Table**

Purpose: Stores products available for purchase.

Description:

- Each product has a unique ID
- Must belong to a category
- Stores product details (name, price, inventory)
- Includes flags for out-of-stock status
- Quantity defaults to 0 and must be non-negative

```
CREATE TABLE PRODUCT (
    Product_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(100) NOT NULL,
    Description VARCHAR2(500),
    Price NUMBER(10,2) NOT NULL CHECK (Price >= 0),
    Quantity_Available NUMBER DEFAULT 0 CHECK (Quantity_Available >= 0),
    Out_of_Stock NUMBER(1) DEFAULT 0 CHECK (Out_of_Stock IN (0, 1)),
```

Brand VARCHAR2(50),

Category_ID NUMBER NOT NULL,

CONSTRAINT fk_product_category FOREIGN KEY (Category_ID) REFERENCES CATEGORY(Category_ID)

);

- **ORDER_PRODUCT Table (Junction Table)**

Purpose: Links orders to products (M:N relationship).

Description:

- Composite primary key (Order_ID + Product_ID)
- Stores quantity and discount for each product in an order
- Quantity must be positive
- Discount must be between 0–100%

CREATE TABLE ORDER_PRODUCT (

Order_ID NUMBER NOT NULL,

Product_ID NUMBER NOT NULL,

Quantity NUMBER NOT NULL CHECK (Quantity > 0),

Discount NUMBER(5,2) DEFAULT 0 CHECK (Discount BETWEEN 0 AND 100),

CONSTRAINT pk_order_product PRIMARY KEY (Order_ID, Product_ID),

CONSTRAINT fk_op_order FOREIGN KEY (Order_ID) REFERENCES CUSTOMER_ORDERS(Order_ID) ON DELETE CASCADE,

CONSTRAINT fk_op_product FOREIGN KEY (Product_ID) REFERENCES PRODUCT(Product_ID) ON DELETE CASCADE

);

- **SUPPLIER Table**

Purpose: Stores information about product suppliers.

Description:

- Each supplier has a unique ID
- Stores supplier name
- Suppliers can provide multiple products
- Products can have multiple suppliers

```
CREATE TABLE SUPPLIER (
   Supplier_ID NUMBER PRIMARY KEY,
   Supplier_Name VARCHAR2(100) NOT NULL
);
```

- **SUPPLIER_PRODUCT Table (Junction Table)**

Purpose: Links suppliers to products (M:N relationship).

Description:

- Composite primary key (Supplier_ID + Product_ID + Delivery_Date)
- Tracks delivery dates for supplier products

```
CREATE TABLE SUPPLIER_PRODUCT (
   Supplier_ID NUMBER NOT NULL,
   Product_ID NUMBER NOT NULL,
   Delivery_Date DATE NOT NULL,
   CONSTRAINT pk_supplier_product PRIMARY KEY (Supplier_ID, Product_ID),
   CONSTRAINT fk_sp_supplier FOREIGN KEY (Supplier_ID) REFERENCES SUPPLIER(Supplier_ID) ON DELETE CASCADE,
   CONSTRAINT fk_sp_product FOREIGN KEY (Product_ID) REFERENCES PRODUCT(Product_ID) ON DELETE CASCADE
);
```

Multivalued Attribute Tables

These tables handle attributes that can have multiple values for a single entity.

- **EMPLOYEE_PHONE_NUMBER**

```
CREATE TABLE EMPLOYEE_PHONE_NUMBER (
   Employee_ID NUMBER NOT NULL,
   Phone_Number VARCHAR2(20) NOT NULL,
   CONSTRAINT pk_employee_phone PRIMARY KEY (Employee_ID, Phone_Number),
   CONSTRAINT fk_epn_employee FOREIGN KEY (Employee_ID) REFERENCES EMPLOYEE(Employee_ID) ON DELETE CASCADE
);
```

- **EMPLOYEE_ALLERGIES**

```
CREATE TABLE EMPLOYEE_ALLERGIES (
   Employee_ID NUMBER NOT NULL,
   Allergy VARCHAR2(100) NOT NULL,
   CONSTRAINT pk_employee_allergies PRIMARY KEY (Employee_ID, Allergy),
   CONSTRAINT fk_ea_employee FOREIGN KEY (Employee_ID) REFERENCES EMPLOYEE(Employee_ID) ON DELETE CASCADE
);
```

- **CUSTOMER_EMAIL_ADDRESS**

```
CREATE TABLE CUSTOMER_EMAIL_ADDRESS (
    Customer_ID NUMBER NOT NULL,
    Email_Address VARCHAR2(100) NOT NULL,
    CONSTRAINT pk_customer_email PRIMARY KEY (Customer_ID, Email_Address),
    CONSTRAINT fk_cea_customer FOREIGN KEY (Customer_ID) REFERENCES
CUSTOMER(Customer_ID) ON DELETE CASCADE
);
```

- **SUPPLIER_PHONE_NUMBER**

```
CREATE TABLE SUPPLIER_PHONE_NUMBER (
    Supplier_ID NUMBER NOT NULL,
    Phone_Number VARCHAR2(20) NOT NULL,
    CONSTRAINT pk_supplier_phone PRIMARY KEY (Supplier_ID, Phone_Number),
    CONSTRAINT fk_spn_supplier FOREIGN KEY (Supplier_ID) REFERENCES
SUPPLIER(Supplier_ID) ON DELETE CASCADE
);
```

- **PET_BREED**

```
CREATE TABLE PET_BREED (
```

```
    Pet_ID NUMBER NOT NULL,
    Breed VARCHAR2(50) NOT NULL,
    CONSTRAINT pk_pet_breed PRIMARY KEY (Pet_ID, Breed),
    CONSTRAINT fk_pb_pet FOREIGN KEY (Pet_ID) REFERENCES PET(Pet_ID) ON
DELETE CASCADE
);
```

9- Data Insertion:

CUSTOMER Table

```
INSERT INTO CUSTOMER VALUES (1, 'John', 'Smith');
INSERT INTO CUSTOMER VALUES (2, 'Emily', 'Johnson');
INSERT INTO CUSTOMER VALUES (3, 'Michael', 'Williams');
INSERT INTO CUSTOMER VALUES (4, 'Sarah', 'Brown');
INSERT INTO CUSTOMER VALUES (5, 'David', 'Jones');
INSERT INTO CUSTOMER VALUES (6, 'Jessica', 'Garcia');
INSERT INTO CUSTOMER VALUES (7, 'Robert', 'Miller');
INSERT INTO CUSTOMER VALUES (8, 'Jennifer', 'Davis');
INSERT INTO CUSTOMER VALUES (9, 'William', 'Rodriguez');
INSERT INTO CUSTOMER VALUES (10, 'Elizabeth', 'Martinez');
INSERT INTO CUSTOMER VALUES (11, 'James', 'Hernandez');
```

PET Table

```
INSERT INTO PET VALUES (1, 'Buddy', 5, 'Male', 1);
INSERT INTO PET VALUES (2, 'Bella', 3, 'Female', 2);
INSERT INTO PET VALUES (3, 'Max', 7, 'Male', 3);
```

```
INSERT INTO PET VALUES (4, 'Lucy', 2, 'Female', 4);

INSERT INTO PET VALUES (5, 'Charlie', 4, 'Male', 5);

INSERT INTO PET VALUES (6, 'Luna', 1, 'Female', 6);

INSERT INTO PET VALUES (7, 'Cooper', 6, 'Male', 7);

INSERT INTO PET VALUES (8, 'Daisy', 5, 'Female', 8);

INSERT INTO PET VALUES (9, 'Rocky', 8, 'Male', 9);

INSERT INTO PET VALUES (10, 'Molly', 4, 'Female', 10);

INSERT INTO PET VALUES (11, 'Bear', 2, 'Male', 1);
```

HEALTH_RECORD Table

```
INSERT INTO HEALTH_RECORD VALUES (1, TO_TIMESTAMP('2023-01-15
10:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'Annual checkup - healthy');

INSERT INTO HEALTH_RECORD VALUES (2, TO_TIMESTAMP('2023-02-20
14:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'Vaccinations updated');

INSERT INTO HEALTH_RECORD VALUES (3, TO_TIMESTAMP('2023-03-10
09:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Dental cleaning performed');

INSERT INTO HEALTH_RECORD VALUES (4, TO_TIMESTAMP('2023-04-05
11:45:00', 'YYYY-MM-DD HH24:MI:SS'), 'Minor skin irritation treated');

INSERT INTO HEALTH_RECORD VALUES (5, TO_TIMESTAMP('2023-05-12
13:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'Annual checkup - needs weight
management');

INSERT INTO HEALTH_RECORD VALUES (6, TO_TIMESTAMP('2023-06-18
15:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Spaying procedure completed');

INSERT INTO HEALTH_RECORD VALUES (7, TO_TIMESTAMP('2023-07-22
10:15:00', 'YYYY-MM-DD HH24:MI:SS'), 'Ear infection treated');

INSERT INTO HEALTH_RECORD VALUES (8, TO_TIMESTAMP('2023-08-30
16:45:00', 'YYYY-MM-DD HH24:MI:SS'), 'Annual vaccinations');

INSERT INTO HEALTH_RECORD VALUES (9, TO_TIMESTAMP('2023-09-14
08:30:00', 'YYYY-MM-DD HH24:MI:SS'), 'Arthritis medication prescribed');
```

```
INSERT INTO HEALTH_RECORD VALUES (10, TO_TIMESTAMP('2023-10-25
12:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Routine checkup - healthy');

INSERT INTO HEALTH_RECORD VALUES (1, TO_TIMESTAMP('2023-11-10
14:00:00', 'YYYY-MM-DD HH24:MI:SS'), 'Follow-up on weight management');
```

PET_INSURANCE Table

```
INSERT INTO PET_INSURANCE VALUES (1, 'Comprehensive', 'PetCare Inc',
TO_DATE('2023-01-01', 'YYYY-MM-DD'), TO_DATE('2024-01-01', 'YYYY-MM-DD'), 1);

INSERT INTO PET_INSURANCE VALUES (2, 'Accident Only', 'SafePets',
TO_DATE('2023-02-15', 'YYYY-MM-DD'), TO_DATE('2024-02-15', 'YYYY-MM-DD'), 2);

INSERT INTO PET_INSURANCE VALUES (3, 'Wellness', 'HealthyPaws',
TO_DATE('2023-03-10', 'YYYY-MM-DD'), TO_DATE('2024-03-10', 'YYYY-MM-DD'), 3);

INSERT INTO PET_INSURANCE VALUES (4, 'Comprehensive', 'PetCare Inc',
TO_DATE('2023-04-05', 'YYYY-MM-DD'), TO_DATE('2024-04-05', 'YYYY-MM-DD'), 4);

INSERT INTO PET_INSURANCE VALUES (5, 'Accident Only', 'SafePets',
TO_DATE('2023-05-20', 'YYYY-MM-DD'), TO_DATE('2024-05-20', 'YYYY-MM-DD'), 5);

INSERT INTO PET_INSURANCE VALUES (6, 'Wellness', 'HealthyPaws',
TO_DATE('2023-06-15', 'YYYY-MM-DD'), TO_DATE('2024-06-15', 'YYYY-MM-DD'), 6);

INSERT INTO PET_INSURANCE VALUES (7, 'Comprehensive', 'PetCare Inc',
TO_DATE('2023-07-01', 'YYYY-MM-DD'), TO_DATE('2024-07-01', 'YYYY-MM-DD'), 7);

INSERT INTO PET_INSURANCE VALUES (8, 'Accident Only', 'SafePets',
TO_DATE('2023-08-10', 'YYYY-MM-DD'), TO_DATE('2024-08-10', 'YYYY-MM-DD'), 8);

INSERT INTO PET_INSURANCE VALUES (9, 'Wellness', 'HealthyPaws',
TO_DATE('2023-09-05', 'YYYY-MM-DD'), TO_DATE('2024-09-05', 'YYYY-MM-DD'), 9);

INSERT INTO PET_INSURANCE VALUES (10, 'Comprehensive', 'PetCare Inc',
TO_DATE('2023-10-15', 'YYYY-MM-DD'), TO_DATE('2024-10-15', 'YYYY-MM-DD'), 10);

INSERT INTO PET_INSURANCE VALUES (11, 'Accident Only', 'SafePets',
TO_DATE('2023-11-01', 'YYYY-MM-DD'), TO_DATE('2024-11-01', 'YYYY-MM-DD'), 1);
```

SERVICE Table

INSERT INTO SERVICE VALUES (1, 'Basic Grooming', 'Bath, brush, nail trim, ear cleaning', 45.00);

INSERT INTO SERVICE VALUES (2, 'Deluxe Grooming', 'Basic grooming plus teeth brushing, paw treatment', 75.00);

INSERT INTO SERVICE VALUES (3, 'Vaccination', 'Core vaccination package', 65.00);

INSERT INTO SERVICE VALUES (4, 'Dental Cleaning', 'Professional teeth cleaning', 120.00);

INSERT INTO SERVICE VALUES (5, 'Health Checkup', 'Comprehensive physical examination', 50.00);

INSERT INTO SERVICE VALUES (6, 'Pet Training', 'One-hour obedience training session', 60.00);

INSERT INTO SERVICE VALUES (7, 'Pet Sitting', 'Daily care while owner is away', 30.00);

INSERT INTO SERVICE VALUES (8, 'Emergency Care', 'Immediate medical attention', 150.00);

INSERT INTO SERVICE VALUES (9, 'Microchipping', 'Permanent ID implantation', 40.00);

INSERT INTO SERVICE VALUES (10, 'Pet Photography', 'Professional photo session', 85.00);

INSERT INTO SERVICE VALUES (11, 'Pet Massage', 'Therapeutic massage session', 55.00);

EMPLOYEE Table

INSERT INTO EMPLOYEE VALUES (1, 'Daniel', 'Wilson', 28, 42000.00, TO_DATE('2021-03-15', 'YYYY-MM-DD'), 'Veterinarian');

INSERT INTO EMPLOYEE VALUES (2, 'Olivia', 'Anderson', 32, 38000.00, TO_DATE('2020-05-20', 'YYYY-MM-DD'), 'Groomer');

INSERT INTO EMPLOYEE VALUES (3, 'Matthew', 'Thomas', 25, 32000.00, TO_DATE('2022-01-10', 'YYYY-MM-DD'), 'Trainer');

INSERT INTO EMPLOYEE VALUES (4, 'Sophia', 'Taylor', 29, 35000.00, TO_DATE('2021-07-05', 'YYYY-MM-DD'), 'Veterinary Technician');

```
INSERT INTO EMPLOYEE VALUES (5, 'Andrew', 'Moore', 35, 45000.00,
TO_DATE('2019-11-12', 'YYYY-MM-DD'), 'Veterinarian');

INSERT INTO EMPLOYEE VALUES (6, 'Isabella', 'Jackson', 27, 33000.00,
TO_DATE('2022-03-25', 'YYYY-MM-DD'), 'Groomer');

INSERT INTO EMPLOYEE VALUES (7, 'Christopher', 'White', 31, 40000.00,
TO_DATE('2020-09-18', 'YYYY-MM-DD'), 'Veterinarian');

INSERT INTO EMPLOYEE VALUES (8, 'Ava', 'Harris', 24, 30000.00,
TO_DATE('2023-02-08', 'YYYY-MM-DD'), 'Receptionist');

INSERT INTO EMPLOYEE VALUES (9, 'Joshua', 'Martin', 30, 37000.00,
TO_DATE('2021-04-30', 'YYYY-MM-DD'), 'Trainer');

INSERT INTO EMPLOYEE VALUES (10, 'Mia', 'Thompson', 26, 34000.00,
TO_DATE('2022-06-15', 'YYYY-MM-DD'), 'Veterinary Technician');

INSERT INTO EMPLOYEE VALUES (11, 'Ethan', 'Garcia', 33, 41000.00,
TO_DATE('2020-12-03', 'YYYY-MM-DD'), 'Veterinarian');
```

CHECK_IN_OUT Table

```
INSERT INTO CHECK_IN_OUT VALUES (1, TO_TIMESTAMP('2023-01-02 08:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkin');

INSERT INTO CHECK_IN_OUT VALUES (1, TO_TIMESTAMP('2023-01-02 17:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkout');

INSERT INTO CHECK_IN_OUT VALUES (2, TO_TIMESTAMP('2023-01-02 08:30:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkin');

INSERT INTO CHECK_IN_OUT VALUES (2, TO_TIMESTAMP('2023-01-02 16:30:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkout');

INSERT INTO CHECK_IN_OUT VALUES (3, TO_TIMESTAMP('2023-01-02 09:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkin');

INSERT INTO CHECK_IN_OUT VALUES (3, TO_TIMESTAMP('2023-01-02 17:30:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkout');

INSERT INTO CHECK_IN_OUT VALUES (4, TO_TIMESTAMP('2023-01-02 08:15:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkin');

INSERT INTO CHECK_IN_OUT VALUES (4, TO_TIMESTAMP('2023-01-02 16:45:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkout');
```

```
INSERT INTO CHECK_IN_OUT VALUES (5, TO_TIMESTAMP('2023-01-02 07:45:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkin');

INSERT INTO CHECK_IN_OUT VALUES (5, TO_TIMESTAMP('2023-01-02 18:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkout');

INSERT INTO CHECK_IN_OUT VALUES (6, TO_TIMESTAMP('2023-01-02 08:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Checkin');

INSERT INTO APPOINTMENT VALUES (1, TO_TIMESTAMP('2023-01-10 10:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Completed', 1, 1);

INSERT INTO APPOINTMENT VALUES (2, TO_TIMESTAMP('2023-01-12 14:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Completed', 2, 2);

INSERT INTO APPOINTMENT VALUES (3, TO_TIMESTAMP('2023-01-15 11:30:00',
'YYYY-MM-DD HH24:MI:SS'), 'Completed', 3, 3);

INSERT INTO APPOINTMENT VALUES (4, TO_TIMESTAMP('2023-01-18 09:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Completed', 4, 4);

INSERT INTO APPOINTMENT VALUES (5, TO_TIMESTAMP('2023-01-20 13:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Completed', 5, 5);

INSERT INTO APPOINTMENT VALUES (6, TO_TIMESTAMP('2023-01-22 15:30:00',
'YYYY-MM-DD HH24:MI:SS'), 'Completed', 6, 6);

INSERT INTO APPOINTMENT VALUES (7, TO_TIMESTAMP('2023-01-25 10:30:00',
'YYYY-MM-DD HH24:MI:SS'), 'Completed', 7, 7);

INSERT INTO APPOINTMENT VALUES (8, TO_TIMESTAMP('2023-01-28 16:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Completed', 8, 8);

INSERT INTO APPOINTMENT VALUES (9, TO_TIMESTAMP('2023-02-01 11:00:00',
'YYYY-MM-DD HH24:MI:SS'), 'Scheduled', 9, 9);

INSERT INTO APPOINTMENT VALUES (10, TO_TIMESTAMP('2023-02-03 14:30:00',
'YYYY-MM-DD HH24:MI:SS'), 'Scheduled', 10, 10);

INSERT INTO APPOINTMENT VALUES (11, TO_TIMESTAMP('2023-02-05 09:30:00',
'YYYY-MM-DD HH24:MI:SS'), 'Canceled', 1, 1);
```

APPOINTMENT_SERVICE Table

```
INSERT INTO APPOINTMENT_SERVICE VALUES (1, 1);
```

```
INSERT INTO APPOINTMENT_SERVICE VALUES (1, 5);
INSERT INTO APPOINTMENT_SERVICE VALUES (2, 2);
INSERT INTO APPOINTMENT_SERVICE VALUES (3, 3);
INSERT INTO APPOINTMENT_SERVICE VALUES (4, 4);
INSERT INTO APPOINTMENT_SERVICE VALUES (5, 5);
INSERT INTO APPOINTMENT_SERVICE VALUES (6, 6);
INSERT INTO APPOINTMENT_SERVICE VALUES (7, 7);
INSERT INTO APPOINTMENT_SERVICE VALUES (8, 8);
INSERT INTO APPOINTMENT_SERVICE VALUES (9, 9);
INSERT INTO APPOINTMENT_SERVICE VALUES (10, 10);
INSERT INTO APPOINTMENT_SERVICE VALUES (1, 3);
INSERT INTO APPOINTMENT_SERVICE VALUES (2, 5);
```

FEEDBACK Table

```
INSERT INTO FEEDBACK VALUES (1, TO_DATE('2023-01-11', 'YYYY-MM-DD'), 5,
'Excellent service, very professional staff');
INSERT INTO FEEDBACK VALUES (2, TO_DATE('2023-01-13', 'YYYY-MM-DD'), 4,
'Good experience, but waiting time was a bit long');
INSERT INTO FEEDBACK VALUES (3, TO_DATE('2023-01-16', 'YYYY-MM-DD'), 5,
'My pet loved the trainer!');
INSERT INTO FEEDBACK VALUES (4, TO_DATE('2023-01-19', 'YYYY-MM-DD'), 3,
'Service was okay, but a bit expensive');
INSERT INTO FEEDBACK VALUES (5, TO_DATE('2023-01-21', 'YYYY-MM-DD'), 5,
'Very thorough health checkup');
INSERT INTO FEEDBACK VALUES (6, TO_DATE('2023-01-23', 'YYYY-MM-DD'), 4,
'Happy with the training session');
INSERT INTO FEEDBACK VALUES (7, TO_DATE('2023-01-26', 'YYYY-MM-DD'), 5,
'Emergency care saved my pet, thank you!');
INSERT INTO FEEDBACK VALUES (8, TO_DATE('2023-01-29', 'YYYY-MM-DD'), 2,
'Not satisfied with the grooming results');
```

INSERT INTO FEEDBACK VALUES (9, TO_DATE('2023-02-02', 'YYYY-MM-DD'), 5, 'Great communication and care');

INSERT INTO FEEDBACK VALUES (10, TO_DATE('2023-02-04', 'YYYY-MM-DD'), 4, 'Good service overall');

CUSTOMER_ORDERS Table

INSERT INTO CUSTOMER_ORDERS VALUES (1, TO_DATE('2023-01-05', 'YYYY-MM-DD'), 125.50, 1);

INSERT INTO CUSTOMER_ORDERS VALUES (2, TO_DATE('2023-01-08', 'YYYY-MM-DD'), 89.75, 2);

INSERT INTO CUSTOMER_ORDERS VALUES (3, TO_DATE('2023-01-10', 'YYYY-MM-DD'), 210.00, 3);

INSERT INTO CUSTOMER_ORDERS VALUES (4, TO_DATE('2023-01-12', 'YYYY-MM-DD'), 45.25, 4);

INSERT INTO CUSTOMER_ORDERS VALUES (5, TO_DATE('2023-01-15', 'YYYY-MM-DD'), 178.90, 5);

INSERT INTO CUSTOMER_ORDERS VALUES (6, TO_DATE('2023-01-18', 'YYYY-MM-DD'), 65.30, 6);

INSERT INTO CUSTOMER_ORDERS VALUES (7, TO_DATE('2023-01-20', 'YYYY-MM-DD'), 320.45, 7);

INSERT INTO CUSTOMER_ORDERS VALUES (8, TO_DATE('2023-01-22', 'YYYY-MM-DD'), 95.60, 8);

INSERT INTO CUSTOMER_ORDERS VALUES (9, TO_DATE('2023-01-25', 'YYYY-MM-DD'), 150.75, 9);

INSERT INTO CUSTOMER_ORDERS VALUES (10, TO_DATE('2023-01-28', 'YYYY-MM-DD'), 72.30, 10);

INSERT INTO CUSTOMER_ORDERS VALUES (11, TO_DATE('2023-01-30', 'YYYY-MM-DD'), 230.00, 1);

CATEGORY Table

INSERT INTO CATEGORY VALUES (1, 'Food', 'Pet food and treats');

INSERT INTO CATEGORY VALUES (2, 'Toys', 'Interactive and chew toys');

INSERT INTO CATEGORY VALUES (3, 'Grooming', 'Brushes, shampoos, and accessories');

INSERT INTO CATEGORY VALUES (4, 'Health', 'Medications and supplements');

INSERT INTO CATEGORY VALUES (5, 'Accessories', 'Collars, leashes, and beds');

INSERT INTO CATEGORY VALUES (6, 'Litter', 'Cat litter and accessories');

INSERT INTO CATEGORY VALUES (7, 'Aquarium', 'Fish tanks and supplies');

INSERT INTO CATEGORY VALUES (8, 'Bird', 'Cages and bird supplies');

INSERT INTO CATEGORY VALUES (9, 'Reptile', 'Terrariums and reptile supplies');

INSERT INTO CATEGORY VALUES (10, 'Small Animal', 'Cages and supplies for rodents');

INSERT INTO CATEGORY VALUES (11, 'Training', 'Training aids and equipment');

PRODUCT Table

INSERT INTO PRODUCT VALUES (1, 'Premium Dog Food', 'High-quality dry dog food', 45.99, 50, 0, 'Royal Canin', 1);

INSERT INTO PRODUCT VALUES (2, 'Chew Toy', 'Durable rubber chew toy', 12.99, 30, 0, 'Kong', 2);

INSERT INTO PRODUCT VALUES (3, 'Grooming Brush', 'Slicker brush for all coat types', 15.50, 25, 0, 'Furminator', 3);

INSERT INTO PRODUCT VALUES (4, 'Flea Treatment', 'Monthly flea prevention', 25.75, 40, 0, 'Frontline', 4);

INSERT INTO PRODUCT VALUES (5, 'Leather Leash', '6-foot leather leash', 29.99, 15, 0, 'Lupine', 5);

INSERT INTO PRODUCT VALUES (6, 'Clumping Cat Litter', 'Scented clumping litter', 18.50, 35, 0, 'Tidy Cats', 6);

INSERT INTO PRODUCT VALUES (7, 'Aquarium Starter Kit', '10-gallon tank with filter', 89.99, 10, 0, 'Aqueon', 7);

```
INSERT INTO PRODUCT VALUES (8, 'Bird Perch', 'Natural wood bird perch', 14.25, 20, 0, 'Prevue', 8);

INSERT INTO PRODUCT VALUES (9, 'Heat Lamp', 'Reptile heat lamp with bulb', 22.99, 12, 0, 'Zoo Med', 9);

INSERT INTO PRODUCT VALUES (10, 'Hamster Wheel', 'Silent spinner exercise wheel', 19.99, 18, 0, 'Kaytee', 10);

INSERT INTO PRODUCT VALUES (11, 'Training Clicker', 'Basic training clicker', 5.99, 50, 0, 'PetSafe', 11);
```

ORDER_PRODUCT Table

```
INSERT INTO ORDER_PRODUCT VALUES (1, 1, 2, 0);

INSERT INTO ORDER_PRODUCT VALUES (1, 2, 1, 10);

INSERT INTO ORDER_PRODUCT VALUES (2, 3, 1, 0);

INSERT INTO ORDER_PRODUCT VALUES (2, 4, 2, 5);

INSERT INTO ORDER_PRODUCT VALUES (3, 5, 1, 15);

INSERT INTO ORDER_PRODUCT VALUES (3, 6, 3, 0);

INSERT INTO ORDER_PRODUCT VALUES (4, 7, 1, 0);

INSERT INTO ORDER_PRODUCT VALUES (5, 8, 2, 10);

INSERT INTO ORDER_PRODUCT VALUES (6, 9, 1, 0);

INSERT INTO ORDER_PRODUCT VALUES (7, 10, 4, 20);

INSERT INTO ORDER_PRODUCT VALUES (8, 11, 1, 0);

INSERT INTO ORDER_PRODUCT VALUES (9, 1, 1, 5);

INSERT INTO ORDER_PRODUCT VALUES (10, 2, 2, 0);
```

SUPPLIER Table

```
INSERT INTO SUPPLIER VALUES (1, 'Pet Supply Distributors');

INSERT INTO SUPPLIER VALUES (2, 'Animal Health Products');
```

```
INSERT INTO SUPPLIER VALUES (3, 'Quality Pet Foods Inc');

INSERT INTO SUPPLIER VALUES (4, 'Grooming Essentials');

INSERT INTO SUPPLIER VALUES (5, 'Toy Manufacturers Ltd');

INSERT INTO SUPPLIER VALUES (6, 'Aquatic Supplies Co');

INSERT INTO SUPPLIER VALUES (7, 'Bird Specialties');

INSERT INTO SUPPLIER VALUES (8, 'Reptile World');

INSERT INTO SUPPLIER VALUES (9, 'Small Animal Products');

INSERT INTO SUPPLIER VALUES (10, 'Training Equipment Inc');

INSERT INTO SUPPLIER VALUES (11, 'Premium Pet Accessories');
```

SUPPLIER_PRODUCT Table

```
INSERT INTO SUPPLIER_PRODUCT VALUES (1, 1, TO_DATE('2023-01-02',
'YYYY-MM-DD'));

INSERT INTO SUPPLIER_PRODUCT VALUES (2, 2, TO_DATE('2023-01-03',
'YYYY-MM-DD'));

INSERT INTO SUPPLIER_PRODUCT VALUES (3, 3, TO_DATE('2023-01-04',
'YYYY-MM-DD'));

INSERT INTO SUPPLIER_PRODUCT VALUES (4, 4, TO_DATE('2023-01-05',
'YYYY-MM-DD'));

INSERT INTO SUPPLIER_PRODUCT VALUES (5, 5, TO_DATE('2023-01-06',
'YYYY-MM-DD'));

INSERT INTO SUPPLIER_PRODUCT VALUES (6, 6, TO_DATE('2023-01-07',
'YYYY-MM-DD'));

INSERT INTO SUPPLIER_PRODUCT VALUES (7, 7, TO_DATE('2023-01-08',
'YYYY-MM-DD'));

INSERT INTO SUPPLIER_PRODUCT VALUES (8, 8, TO_DATE('2023-01-09',
'YYYY-MM-DD'));

INSERT INTO SUPPLIER_PRODUCT VALUES (9, 9, TO_DATE('2023-01-10',
'YYYY-MM-DD'));
```

```
INSERT INTO SUPPLIER_PRODUCT VALUES (10, 10, TO_DATE('2023-01-11',
'YYYY-MM-DD'));

INSERT INTO SUPPLIER_PRODUCT VALUES (11, 11, TO_DATE('2023-01-12',
'YYYY-MM-DD'));
```

EMPLOYEE_PHONE_NUMBER Table

```
INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (1, '555-0101');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (1, '555-0102');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (2, '555-0201');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (3, '555-0301');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (4, '555-0401');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (5, '555-0501');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (6, '555-0601');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (7, '555-0701');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (8, '555-0801');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (9, '555-0901');

INSERT INTO EMPLOYEE_PHONE_NUMBER VALUES (10, '555-1001');
```

EMPLOYEE_ALLERGIES Table

```
INSERT INTO EMPLOYEE_ALLERGIES VALUES (1, 'Pollen');

INSERT INTO EMPLOYEE_ALLERGIES VALUES (2, 'Cat dander');

INSERT INTO EMPLOYEE_ALLERGIES VALUES (3, 'Dust');

INSERT INTO EMPLOYEE_ALLERGIES VALUES (4, 'Peanuts');

INSERT INTO EMPLOYEE_ALLERGIES VALUES (5, 'Bee stings');

INSERT INTO EMPLOYEE_ALLERGIES VALUES (6, 'Latex');

INSERT INTO EMPLOYEE_ALLERGIES VALUES (7, 'Shellfish');

INSERT INTO EMPLOYEE_ALLERGIES VALUES (8, 'Penicillin');
```

```
INSERT INTO EMPLOYEE_ALLERGIES VALUES (9, 'Eggs');

INSERT INTO EMPLOYEE_ALLERGIES VALUES (10, 'Wheat');

INSERT INTO EMPLOYEE_ALLERGIES VALUES (1, 'Mold');
```

CUSTOMER_EMAIL_ADDRESS Table

```
INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (1, 'john.smith@email.com');

INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (2,
'emily.johnson@email.com');

INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (3,
'michael.williams@email.com');

INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (4,
'sarah.brown@email.com');

INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (5, 'david.jones@email.com');

INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (6,
'jessica.garcia@email.com');

INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (7, 'robert.miller@email.com');

INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (8,
'jennifer.davis@email.com');

INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (9,
'william.rodriguez@email.com');

INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (10,
'elizabeth.martinez@email.com');

INSERT INTO CUSTOMER_EMAIL_ADDRESS VALUES (1,
'john.smith.work@email.com');
```

SUPPLIER_PHONE_NUMBER Table

```
INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (1, '555-1001');

INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (2, '555-2001');

INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (3, '555-3001');
```

```
INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (4, '555-4001');
INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (5, '555-5001');
INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (6, '555-6001');
INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (7, '555-7001');
INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (8, '555-8001');
INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (9, '555-9001');
INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (10, '555-1010');
INSERT INTO SUPPLIER_PHONE_NUMBER VALUES (11, '555-1111');
```

PET_BREED Table

```
INSERT INTO PET_BREED VALUES (1, 'Labrador Retriever');
INSERT INTO PET_BREED VALUES (2, 'Golden Retriever');
INSERT INTO PET_BREED VALUES (3, 'German Shepherd');
INSERT INTO PET_BREED VALUES (4, 'Beagle');
INSERT INTO PET_BREED VALUES (5, 'Bulldog');
INSERT INTO PET_BREED VALUES (6, 'Poodle');
INSERT INTO PET_BREED VALUES (7, 'Rottweiler');
INSERT INTO PET_BREED VALUES (8, 'Boxer');
INSERT INTO PET_BREED VALUES (9, 'Dachshund');
INSERT INTO PET_BREED VALUES (10, 'Siberian Husky');
INSERT INTO PET_BREED VALUES (1, 'Mixed Breed');
```

10- Transactions and Queries:

Transaction 1: Pet Owners with Multiple Pets

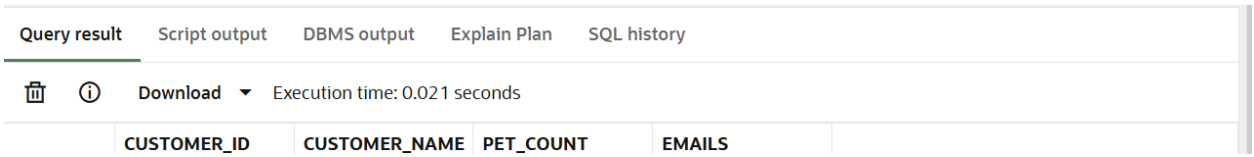
Description:

This query identifies customers who own more than one pet. It counts the number of pets registered under each customer and also aggregates their email addresses into a single string. This is helpful for targeting communications toward highly engaged pet owners or identifying customers who might benefit from multi-pet services or discounts.

Expected Outcome:

You will receive a list of customers who have more than one pet. For each customer, the output will include their full name, the total number of pets they own, and all associated email addresses grouped in a comma-separated format.

```
SELECT c.Customer_ID, c.First_Name || ' ' || c.Last_Name AS Customer_Name,
    COUNT(p.Pet_ID) AS Pet_Count,
    LISTAGG(e.Email_Address, ', ') WITHIN GROUP (ORDER BY e.Email_Address)
AS Emails
FROM CUSTOMER c
JOIN PET p ON c.Customer_ID = p.Customer_ID
JOIN CUSTOMER_EMAIL_ADDRESS e ON c.Customer_ID = e.Customer_ID
GROUP BY c.Customer_ID, c.First_Name, c.Last_Name
HAVING COUNT(p.Pet_ID) > 1;
```



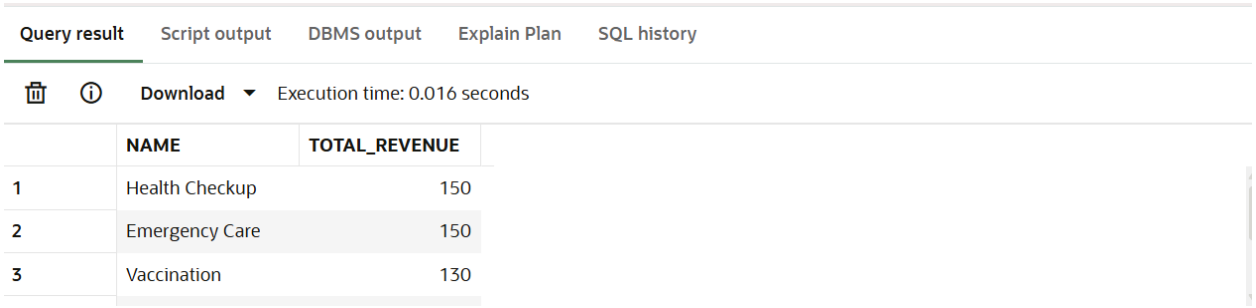
Transaction 2: Top Revenue Generating Services

Description:

This query calculates the total revenue generated by each service based on completed appointments. It filters out services that have earned $50 or less, ensuring only top-performing services are displayed. The results are sorted in descending order to show the highest revenue generators first.

Expected Outcome:
You will get a ranked list of services that have generated over $50 in revenue from completed appointments. Each entry includes the service name and the total revenue it has brought in, making it easier to determine which services contribute most to the business.

```
SELECT s.Name, SUM(s.Price) AS Total_Revenue
FROM SERVICE s
JOIN APPOINTMENT_SERVICE aps ON s.Service_ID = aps.Service_ID
JOIN APPOINTMENT a ON aps.Appointment_ID = a.Appointment_ID
WHERE a.Status = 'Completed'
GROUP BY s.Name
HAVING SUM(s.Price) > 50
ORDER BY Total_Revenue DESC;
```



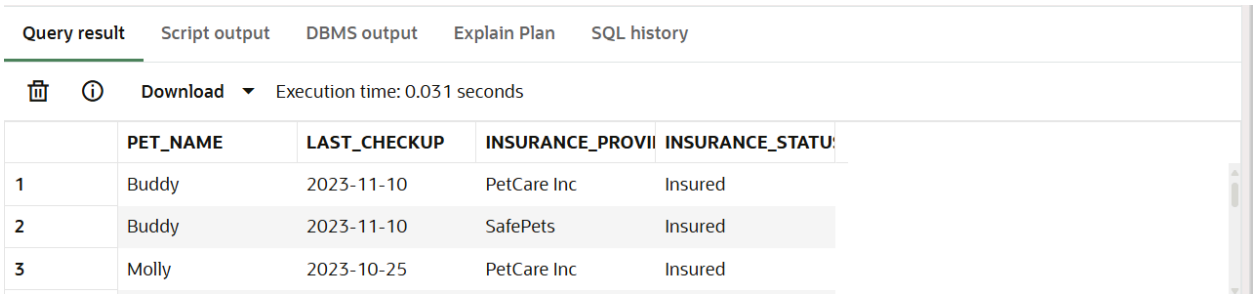
	NAME	TOTAL_REVENUE
1	Health Checkup	150
2	Emergency Care	150
3	Vaccination	130

Transaction 3: Pets with Recent Health Updates

Description:
This query retrieves information on pets whose health records were updated in the year 2023. It includes the name of each pet, the date of their last check-up, and their current insurance status. It also lists the provider if the pet is insured. This information is useful for ensuring timely follow-ups and understanding how many pets are actively monitored and protected.

Expected Outcome:
You will see a list of pets with recent health record updates from 2023, including when their last check-up occurred, whether they have insurance, and if so, the name of the insurance provider. This helps identify actively monitored pets and evaluate insurance coverage across your client base.

```
SELECT p.Name AS Pet_Name,
    TO_CHAR(hr.Last_Updated, 'YYYY-MM-DD') AS Last_Checkup,
    pi.Provider AS Insurance_Provider,
    CASE WHEN pi.Insurance_ID IS NOT NULL THEN 'Insured' ELSE 'Not Insured'
END AS Insurance_Status
FROM PET p
JOIN HEALTH_RECORD hr ON p.Pet_ID = hr.Pet_ID
LEFT JOIN PET_INSURANCE pi ON p.Pet_ID = pi.Pet_ID
WHERE EXTRACT(YEAR FROM hr.Last_Updated) = 2023
ORDER BY hr.Last_Updated DESC;
```



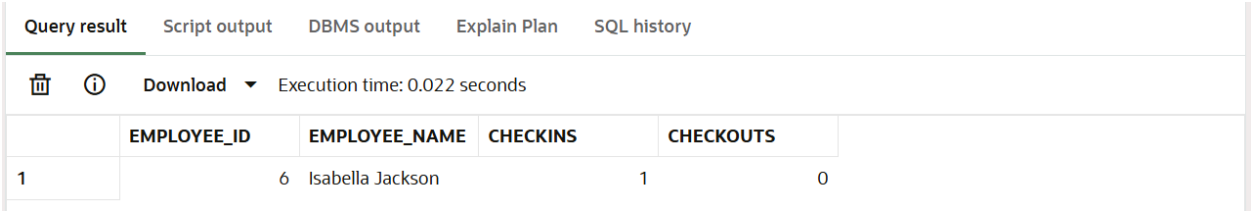
	PET_NAME	LAST_CHECKUP	INSURANCE_PROVI	INSURANCE_STATU
1	Buddy	2023-11-10	PetCare Inc	Insured
2	Buddy	2023-11-10	SafePets	Insured
3	Molly	2023-10-25	PetCare Inc	Insured

Transaction 4: Employee Attendance Analysis

Description:
This query identifies employees with mismatched check-in and check-out records during January 2023. It counts the number of each event type per employee, helping administrators spot potential timesheet or attendance issues that need manual verification or correction.

Expected Outcome:
You will receive a list of employees who have incomplete or inconsistent check-in and check-out logs for January 2023. Each record includes the employee's name and the number of check-ins and check-outs, enabling quick identification of discrepancies.

```
SELECT e.Employee_ID, e.First_Name || ' ' || e.Last_Name AS Employee_Name,
    COUNT(CASE WHEN cio.Event_Type = 'Checkin' THEN 1 END) AS Checkins,
    COUNT(CASE WHEN cio.Event_Type = 'Checkout' THEN 1 END) AS Checkouts
FROM EMPLOYEE e
LEFT JOIN CHECK_IN_OUT cio
    ON e.Employee_ID = cio.Employee_ID
    AND EXTRACT(MONTH FROM cio.Date_Time) = 1
    AND EXTRACT(YEAR FROM cio.Date_Time) = 2023
GROUP BY e.Employee_ID, e.First_Name, e.Last_Name
HAVING COUNT(CASE WHEN cio.Event_Type = 'Checkin' THEN 1 END) <>
    COUNT(CASE WHEN cio.Event_Type = 'Checkout' THEN 1 END);
```



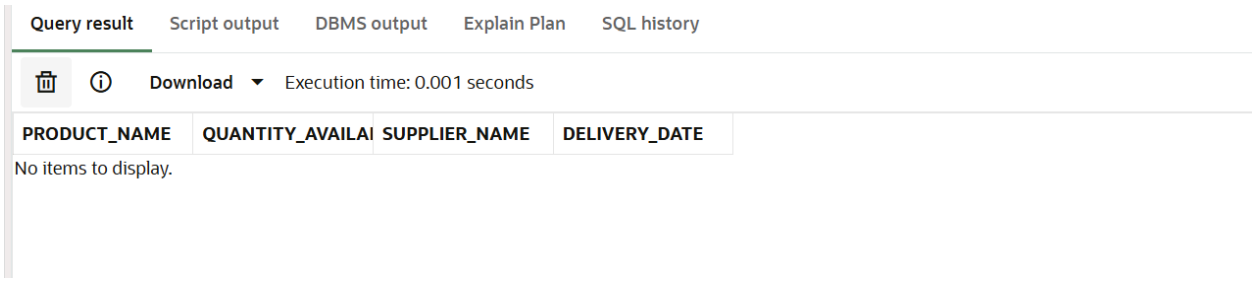
	EMPLOYEE_ID	EMPLOYEE_NAME	CHECKINS	CHECKOUTS
1	6	Isabella Jackson	1	0

Transaction 5: Product Restocking Alert

Description:
This operation flags products with available quantities under 20 units as out of stock and then retrieves their details. It identifies these products along with the supplier responsible and their most recent delivery date. This is useful for inventory management and supplier coordination.

Expected Outcome:
You will obtain a list of products flagged as needing restocking. Each item includes its name, current available quantity, the name of the primary supplier, and the most recent delivery date—helping prioritize restocking efforts.

```
UPDATE PRODUCT
SET Out_of_Stock = 1
WHERE Quantity_Available < 20;
SELECT p.Name AS Product_Name, p.Quantity_Available,
    s.Supplier_Name, sp.Delivery_Date
FROM PRODUCT p
JOIN SUPPLIER_PRODUCT sp ON p.Product_ID = sp.Product_ID
JOIN SUPPLIER s ON sp.Supplier_ID = s.Supplier_ID
WHERE p.Out_of_Stock = 1
ORDER BY p.Quantity_Available ASC;
```



Transaction 6: Customer Lifetime Value

Description:
This query calculates each customer's total spending by summing their expenditures on both services and product purchases. It uses COALESCE to handle customers who may have engaged with only one category, ensuring everyone is included.

Expected Outcome:
You will receive a detailed financial summary for each customer, including their spending on products, services, and a combined total. This report is especially helpful for identifying high-value clients and tailoring loyalty programs or marketing efforts.

```
SELECT c.Customer_ID, c.First_Name || ' ' || c.Last_Name AS Customer_Name,
    COALESCE(SUM(co.Total_Amount), 0) AS Product_Spending,
    COALESCE(SUM(s.Price), 0) AS Service_Spending,
    COALESCE(SUM(co.Total_Amount), 0) + COALESCE(SUM(s.Price), 0) AS Total_Spending
FROM CUSTOMER c
LEFT JOIN CUSTOMER_ORDERS co ON c.Customer_ID = co.Customer_ID
LEFT JOIN APPOINTMENT a ON c.Customer_ID = a.Customer_ID
LEFT JOIN APPOINTMENT_SERVICE aps ON a.Appointment_ID = aps.Appointment_ID
LEFT JOIN SERVICE s ON aps.Service_ID = s.Service_ID
GROUP BY c.Customer_ID, c.First_Name, c.Last_Name
ORDER BY Total_Spending DESC;
```

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.03 seconds

	CUSTOMER_ID	CUSTOMER_NAME	PRODUCT_SPENDIN	SERVICE_SPENDING	TOTAL_SPENDING
1	1	John Smith	1422	320	1742
2	7	Robert Miller	320.45	30	350.45
3	2	Emily Johnson	179.5	125	304.5

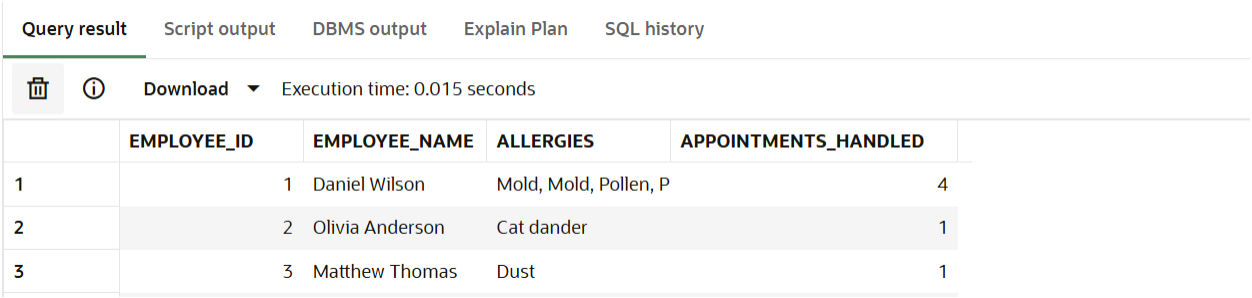
Transaction 7: Employee Allergy Report

Description:
This query lists employees who have recorded allergies and tracks how many appointments they have handled. It also aggregates the specific allergens for each employee, aiding in workplace safety and scheduling.

Expected Outcome:

The result will be a list of employees with at least one allergy, showing their names, the types of allergies they have, and the total number of appointments they have been assigned. This allows for allergy-aware assignment planning and risk management.

```
SELECT e.Employee_ID, e.First_Name || ' ' || e.Last_Name AS Employee_Name,
    LISTAGG(ea.Allergy, ', ') WITHIN GROUP (ORDER BY ea.Allergy) AS Allergies,
    COUNT(a.Appointment_ID) AS Appointments_Handled
FROM EMPLOYEE e
JOIN EMPLOYEE_ALLERGIES ea ON e.Employee_ID = ea.Employee_ID
LEFT JOIN APPOINTMENT a ON e.Employee_ID = a.Employee_ID
GROUP BY e.Employee_ID, e.First_Name, e.Last_Name
HAVING COUNT(ea.Allergy) > 0;
```



Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.015 seconds

	EMPLOYEE_ID	EMPLOYEE_NAME	ALLERGIES	APPOINTMENTS_HANDLED
1	1	Daniel Wilson	Mold, Mold, Pollen, P	4
2	2	Olivia Anderson	Cat dander	1
3	3	Matthew Thomas	Dust	1

Transaction 8: Service Popularity Analysis

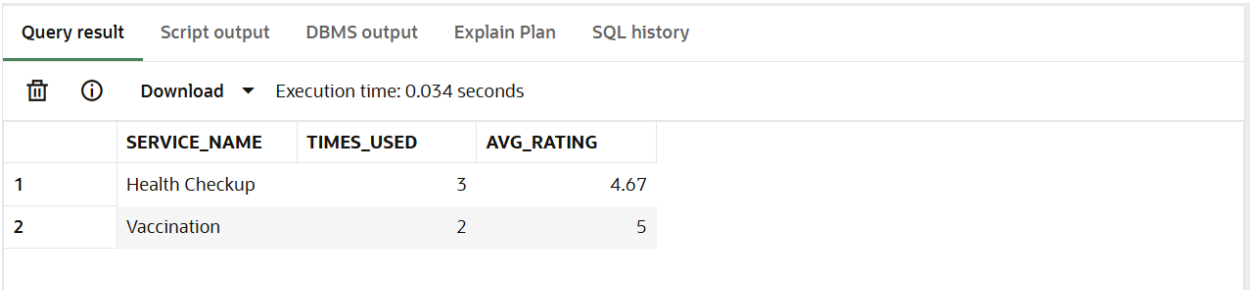
Description:

This query determines which services have been used in more than one appointment

and calculates their average customer feedback rating. This data can inform service development and promotional strategies.

Expected Outcome:
You'll see a list of frequently used services (used in more than one appointment), along with the number of times each service was used and its average customer rating. This helps evaluate both popularity and customer satisfaction.

```
SELECT s.Name AS Service_Name,
    COUNT(aps.Appointment_ID) AS Times_Used,
    ROUND(AVG(f.Rating), 2) AS Avg_Rating
FROM SERVICE s
JOIN APPOINTMENT_SERVICE aps ON s.Service_ID = aps.Service_ID
JOIN APPOINTMENT a ON aps.Appointment_ID = a.Appointment_ID
JOIN FEEDBACK f ON a.Customer_ID = f.Customer_ID
GROUP BY s.Name
HAVING COUNT(aps.Appointment_ID) > 1
ORDER BY Times_Used DESC;
```



	SERVICE_NAME	TIMES_USED	AVG_RATING
1	Health Checkup	3	4.67
2	Vaccination	2	5

Transaction 9: Insurance Expiration Alert

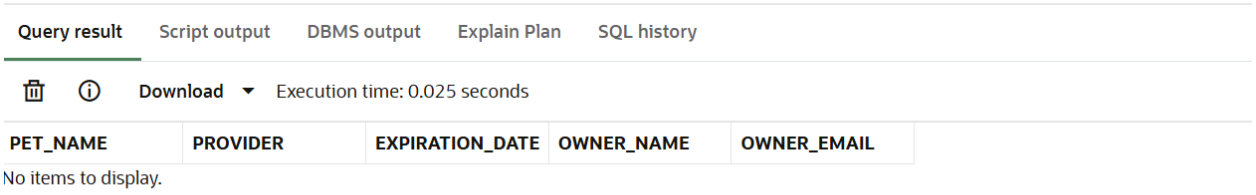
Description:

This query identifies pets whose insurance policies are set to expire within the next 30 days. It includes details about the pet, their owner, the insurance provider, expiration date, and contact email for proactive follow-up.

Expected Outcome:

The output will list pets with upcoming insurance expirations, showing each pet's name, their owner's name and email, the provider's name, and the expiration date. This helps facilitate timely renewals and avoid coverage gaps.

```
SELECT p.Name AS Pet_Name, pi.Provider,
    TO_CHAR(pi.End_Date, 'YYYY-MM-DD') AS Expiration_Date,
    c.First_Name || ' ' || c.Last_Name AS Owner_Name,
    cea.Email_Address AS Owner_Email
FROM PET_INSURANCE pi
JOIN PET p ON pi.Pet_ID = p.Pet_ID
JOIN CUSTOMER c ON p.Customer_ID = c.Customer_ID
JOIN CUSTOMER_EMAIL_ADDRESS cea ON c.Customer_ID = cea.Customer_ID
WHERE pi.End_Date BETWEEN SYSDATE AND SYSDATE + 30;
```



Transaction 10: Complex Customer Activity Report

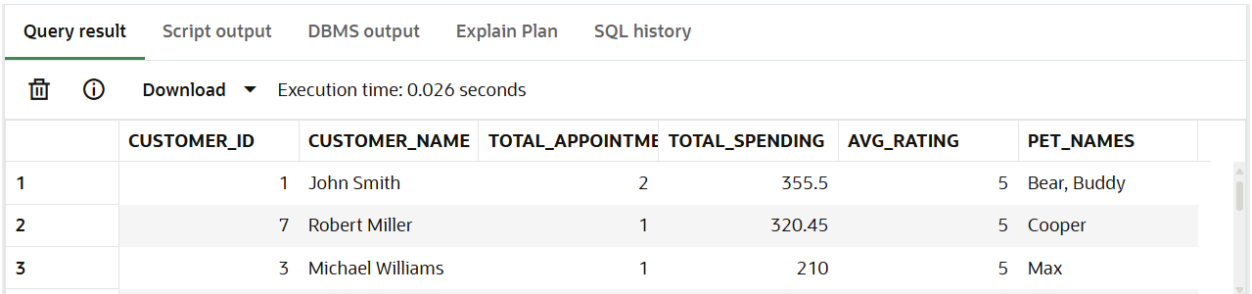
Description:

This comprehensive query provides an overview of each customer's activity with the clinic. It includes total appointments, total amount spent, average feedback rating, and a list of their pets' names. This view is ideal for understanding engagement and loyalty on a per-customer basis.

Expected Outcome:

You will receive a rich profile for each customer, featuring how many appointments they've made, how much they've spent, their average feedback score, and their pets' names. This consolidated view supports strategic planning and customer relationship management.

```
SELECT c.Customer_ID,
    c.First_Name || ' ' || c.Last_Name AS Customer_Name,
    (SELECT COUNT(*) FROM APPOINTMENT a
     WHERE a.Customer_ID = c.Customer_ID) AS Total_Appointments,
    (SELECT SUM(Total_Amount) FROM CUSTOMER_ORDERS co
     WHERE co.Customer_ID = c.Customer_ID) AS Total_Spending,
    (SELECT ROUND(AVG(Rating), 1) FROM FEEDBACK f
     WHERE f.Customer_ID = c.Customer_ID) AS Avg_Rating,
    LISTAGG(p.Name, ', ') WITHIN GROUP (ORDER BY p.Name) AS Pet_Names
FROM CUSTOMER cs
LEFT JOIN PET p ON c.Customer_ID = p.Customer_ID
GROUP BY c.Customer_ID, c.First_Name, c.Last_Name
ORDER BY Total_Spending DESC NULLS LAST;
```



Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.026 seconds

	CUSTOMER_ID	CUSTOMER_NAME	TOTAL_APPOINTME	TOTAL_SPENDING	AVG_RATING	PET_NAMES
1	1	John Smith	2	355.5	5	Bear, Buddy
2	7	Robert Miller	1	320.45	5	Cooper
3	3	Michael Williams	1	210	5	Max

11- Normalization:

Current Relations:

CUSTOMER

Customer_ID, First_Name, Last_Name

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

PET

Pet_ID, Name, Age, Gender, Customer_ID

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

HEALTH_RECORD

Pet_ID, Last_Updated, Record_Description

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

PET_INSURANCE

Insurance_ID, Type, Provider, Start_Date, End_Date, Pet_ID

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

SERVICE

Service_ID, Name, Description, Price

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

EMPLOYEE

Employee_ID, First_Name, Last_Name, Age, Salary, Date_of_Joining, Position

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

CHECK_IN_OUT

Employee_ID, Date_Time, Event_Type

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

APPOINTMENT

Appointment_ID, Date_Time, Status, Customer_ID, Employee_ID

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

APPOINTMENT_SERVICE

Appointment_ID, Service_ID

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

FEEDBACK

Customer_ID, Feedback_Date, Rating, Feedback_Note

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

CUSTOMER_ORDERS

Order_ID, Order_Date, Total_Amount, Customer_ID

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

CATEGORY

Category_ID, Name, Description

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

PRODUCT

Product_ID, Name, Description, Price, Quantity_Available, Out_of_Stock, Brand, Category_ID

- No Partial Dependencies
- Transitive Dependency found (Product_ID -> Quantity_Available, Quantity_Available -> Out_of_Stock)
- Non-key determinants found (Quantity_Available -> Out_of_Stock)
- Not BCNF compliant

ORDER_PRODUCT

Order_ID, Product_ID, Quantity, Discount

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

SUPPLIER

Supplier_ID, Supplier_Name

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

SUPPLIER_PRODUCT

Supplier_ID, Product_ID, Delivery_Date

- No Partial Dependencies
- No Transitive Dependencies
- No Non-key determinants
- BCNF compliant

Relations that violate BCNF:

1) PRODUCT:
    a) Quantity_Available -> Out_of_Stock
        i) Quantity Available is not a supekey (candidate key)
    b) Solution: split the table to two tables
        i) PRODUCT: Product_ID, Name, Description, Price, Brand, Category_ID
        ii) PRODUCT_INVENTORY: Product_ID, Quantity_Available, Out_of_stock

Since all the other tables are properly normalized, this concludes our normalization of the tables to BCNF.