Supporting Small Businesses:

Database Management System for

Local Pet Groomer

EXECUTIVE SUMMARY

**BUSINESS SCENARIO**

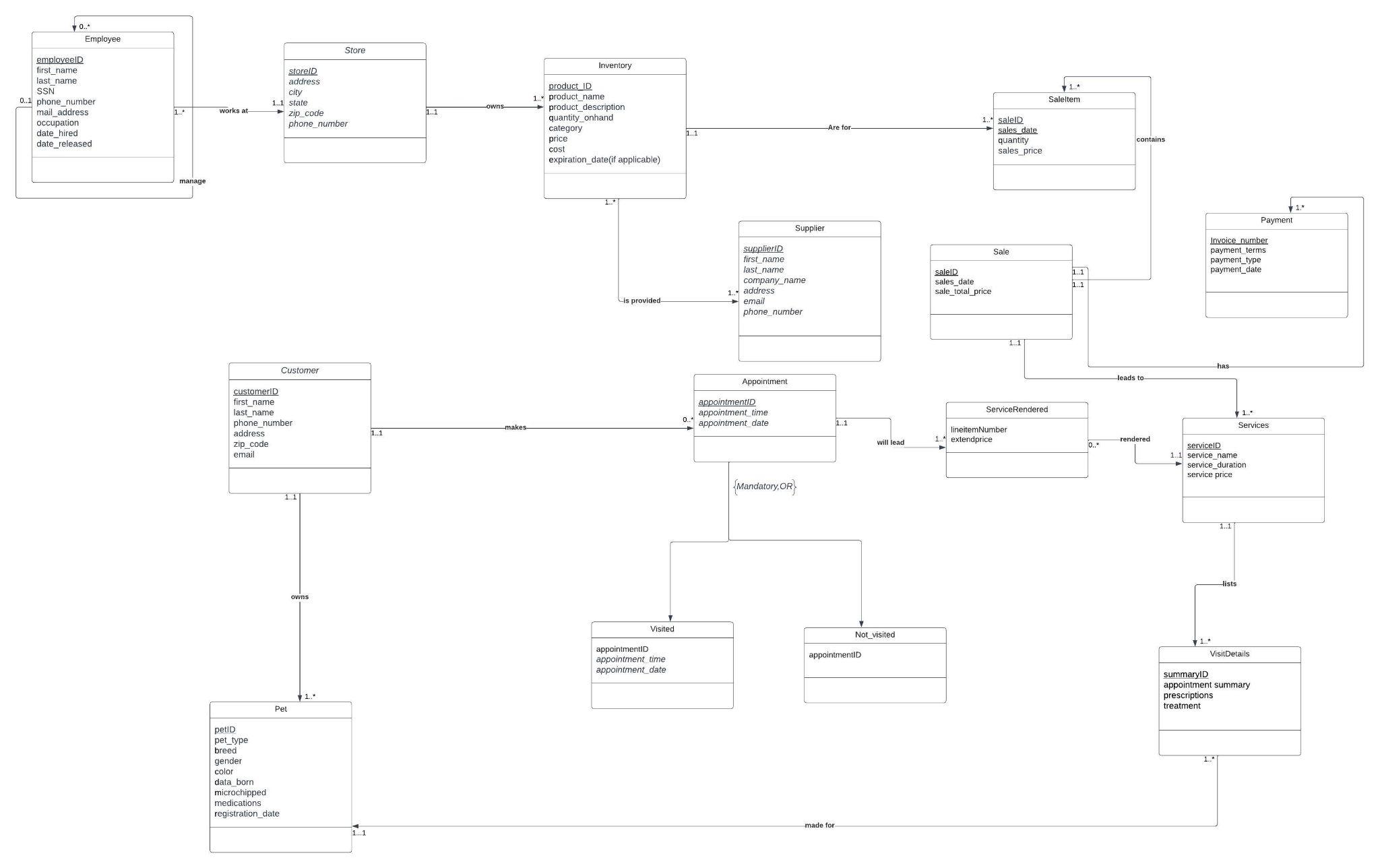
A local campaign to help small businesses and a feature on the evening news has resulted in *Paws ‘n’ All*, a local family-owned pet groomer, being flooded with new customers. As a result, *Paws ‘n’ All* has had difficulty keeping track of appointments, customers, and inventory, resulting in frustration from some customers and additional pressure being placed on the staff.

Our team of professionals operates a small database management consulting firm that provides database solutions for businesses. *Paws ‘n’ All* has reached out to us to create a database management system that allows the owners to effectively and efficiently manage different aspects of their business. Our goal is to create a database management system that allows *Paws ‘n’ All* to store and manage customer, employee, and pet information as well as details such as appointments and inventory for products.

After an initial consultation with *Paws ‘n’ All*, our team has determined that we can create a database system that does the following:

* Stores customer information
* Stores pet information
* Stores employee information
* Stores appointment details
* Stores inventory information
* ​​Stores sales information

The system will house information for all employees working at the store. When a customer enters the store, the employee will be able to record the purpose and details (services rendered) of the customer’s visit in the database system. For example, the employee can determine if the customer is visiting for an appointment and verify if the customer kept the appointment. If the customer is a repeat customer, the employee can access the customer’s information along with the pet’s information. The system will track inventory for any items that are sold as well as the supplier’s information so items with low inventory can be reordered. All payment and invoice information will be stored in the system as well.



**Relationship Sentences\*:**

A **manager** manages zero or many **employees**.

An **employee** may have zero or one **manager**.

A **store** must have one or many **employees**.

An **employee** must work for only one **store**.

Each **store** owns one or many **inventory**.

**Inventory** belongs to only one **store**.

**Inventory** is provided by one or many **suppliers**.

One or many **suppliers** provide one or many **inventories**.

There is **inventory** for one or many **sales items**.

**Sale item** is included in one and only one **inventory**.

**Sale** contains one or many **sale items**.

**Sale item** belong to one and only one **sale**.

**Sale** has one or many **payments**.

**Payment** belongs to one and only one **sale**.

**Sale** leads to one or many **services**.

**Services** have one and only one **sale**.

**Visit detail** summary list one and only one **service**.

**Services** are listed in one or many **visit detail** summaries.

**Visit details** belong to one and only one **pet**.

One **pet** may have one or many **visit details**.

A **customer** must own one or many **pets**.

A **pet** is owned by one and only one **customer**.

A **customer** may make zero or many **appointments**.

An **appointment** is made by one and only one **customer**.

An **appointment** will lead to one or many **services being** **rendered**.

A **service rendered** is from one and only one **appointment**.

A **service rendered** must be one and only one **service**.

Zero or many **services** can be **rendered**.

\*We followed all of your suggestions but we did not link *sale* to *appointment* because an appointment may or may not happen, which means a sale may or may not happen. Also, an appointment does not guarantee a sale either. We removed the relationship between *services* and *payment* and linked *services* to *sales* instead.

**Converting ERD to RDM**

* Employee(employeeID, first\_name, last\_name, SSN, phone\_number, mail\_address, occupation, date\_hired, date\_released, storeID(fk))
* Employee\_manages(managerID, employeeID)
* Store(storeID, address, city, state, zip\_code, phone\_number)
* Inventory(product\_ID, product\_name, product\_description, quantity\_onhand, category, price, cost, expiration\_date, storeID(fk))
* Supplier(supplierID, first\_name, last\_name, company\_name, address, email, phone\_number)
* Inventory\_Supplier(product\_ID*,* supplierID)
* Customer(customerID, first\_name, last\_name, phone\_number, address, zip\_code, email)
* Visited(appointmentID, appointment\_time, appointment\_date, customerID(fk))
* Canceled(appointmentID, appointment\_time, appointment\_date, customerID(fk))
* ServiceRendered(lineitemnumber, extendprice, serviceID(fk), appointmentID(fk))
* Services(serviceID, service\_name, service\_duration, service\_price, saleID(fk))
* VisitDetails(summaryID, appointment\_summary, prescriptions, treatment, petID(fk), serviceID(fk))
* Pet(petID, pet\_type, breed, gender, color, date\_born, microchipped, medications, registration\_date, customerID(fk))
* Payment(Invoice\_number, payment\_terms, payment\_type, payment\_date, saleID(fk))
* Sale(saleID, sales\_date, sale\_total\_price)
* Saleitem(saleID, sales\_date, quantity, sale\_price, productID(fk))

**Normalization**

1. Employee Relation

Step 1: Key exists, 1NF - employeeID

Step 2: Functional dependency

FD1: employeeID -> first\_name, last\_name, SSN, phone\_number, mail\_address, occupation, date\_hired, date\_released, storeID (fk)

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

\*Phone number was identified as a separate key in our first version of normalization. After review, we realized that it wasn’t the best idea to make phone number its own table because phone numbers can change.

1. Store Relation

Step 1: Key - storeID

Step 2: Functional dependency

FD1: storeID -> address, city, state, zip\_code, phone\_number

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

\*Phone number was identified as a separate key in our first version of normalization. After review, we realized that it wasn’t the best idea to make phone number its own table because phone numbers can change. We also removed zip code as a separate key after researching and learning zip codes are not always unique.

1. Inventory Relation

Step 1: Key - product\_ID

Step 2: Functional dependency

FD1: product\_ID -> product\_name, product\_description, quantity\_onhand, category, price, cost, expiration\_date, storeID (fk)

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

1. Supplier Relation

Step 1: Key - supplierID

Step 2: Functional dependency

FD1: supplierID -> first\_name, last\_name, company\_name, address, email, phone\_number

FD2: company\_name -> address, email, phone\_number

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: Yes Transitive Key dependency

~~R1 (supplierID, first\_name, last\_name, company\_name, address, email, phone\_number)~~

R2(company\_name, address, email, phone\_number)

R3(supplierID, first\_name, last\_name, company\_name)

No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

Final Relations include

R2(company\_name,address, email, phone\_number)

R3(supplierID, first\_name, last\_name, company\_name)

1. Customer Relation

Step 1: Key - customerID

Step 2: Functional dependency

FD1: customerID -> first\_name, last\_name, phone\_number, address, zip\_code, email

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

\*Phone number was identified as a separate key in our first version of normalization. After review, we realized that it wasn’t the best idea to make phone number its own table because phone numbers can change. You also advised us not to split the table.

1. Services Relation

Step 1: Key - serviceID

Step 2: Functional dependency

FD1: serviceID -> service\_name, service\_duration, service\_price, saleID (fk)

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

\*Based on your advice, we are not splitting this table.

1. VisitDetails Relation

Step 1: Key - summaryID

Step 2: Functional dependency

FD1: summaryID -> appointment\_summary, prescriptions, treatment, petID (fk), serviceID(fk)

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

1. Pet Relation

Step 1: Key - petID

Step 2: Functional dependency

FD1: petID -> pet\_type, breed, gender, color, date\_born, microchipped, medications, registration\_date, customerID(fk)

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

1. Payment Relation

Step 1: Key - Invoice\_number

Step 2: Functional dependency

FD1: Invoice\_number -> payment\_terms, payment\_type, payment\_date, saleID(fk)

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

1. Sale Relation

Step 1: Key - saleID

Step 2: Functional dependency

FD1: saleID -> sales\_date, sale\_total\_price

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

1. Saleitem Relation

Step 1: Key - saleID, sales\_date

Step 2: Functional dependency

FD1: saleID, sales\_date -> quantity, sale\_price, productID(fk)

Step 3: No Partial Key dependency, checked and satisfies 2NF

Step 4: No Transitive Key dependency, checked and satisfies 3NF

Step 5: All determinants are candidate keys, it’s in BCNF.

**Creating Tables and Inserting Data**

CREATE TABLE Store(

storeID NUMBER NOT NULL,  
address VARCHAR(40),  
city VARCHAR(20),  
state VARCHAR(20),  
zip\_code VARCHAR(20),  
phone\_number VARCHAR(20),

CONSTRAINT pk\_Store PRIMARY KEY (storeID));

INSERT INTO Store VALUES (1001, "2632 Takli Circle", "Brooklyn", "New York", 51202, 1016326532);

INSERT INTO Store VALUES (1002, "4025 Cornell Court", "New York", "New York", 51205, 1016326892);

INSERT INTO Store VALUES (1003, "788 East 145th Avenue", "Queens", "New York", 57805, 1013238956);

INSERT INTO Store VALUES (1004, "1706 Ave V", "Brooklyn", "New York", 11223, 9177458956);

INSERT INTO Store VALUES (1005, "98 Lex Ave", "Bronx", "New York", 11226, 3477458956);

—----------------------------------------------------------------------------------------------------------------------------------------

CREATE TABLE Customer(

customerID NUMBER NOT NULL,  
first\_name VARCHAR(20),  
last\_name VARCHAR(20),  
phone\_number VARCHAR(20),  
address VARCHAR(20),  
zip\_code VARCHAR(20),  
email VARCHAR(20),

CONSTRAINT pk\_Customer PRIMARY KEY (customerID));

INSERT INTO Customer VALUES (1, "Sarah", "Johnson", "123-456-7890", "123 Main St", "12345", "sarah.johnson@example.com");

INSERT INTO Customer VALUES (2, "Jane", "Smith", "555-555-5555", "456 Elm Street", "54321", "jane.smith@email.com");

INSERT INTO Customer VALUES (3, "Mary", "Jones", "987-654-3210", "789 Maple Street", "98765", "mary.jones@email.com");

INSERT INTO Customer VALUES (4, "Peter", "Brown", "098-765-4321", "654 Oak Street", "87654", "peter.brown@email.com");

INSERT INTO Customer VALUES (5, "Susan", "Williams", "321-098-7654", "543 Pine Street", "76543", "susan.williams@email.com");

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE Sale(

saleID NUMBER NOT NULL,  
sales\_date DATE,  
sale\_total\_price NUMBER,

CONSTRAINT pk\_Sale PRIMARY KEY (saleID));

INSERT INTO Sale VALUES(1001,#5/1/2023#,60);

INSERT INTO Sale VALUES(1003,#4/3/2021#,70);

INSERT INTO Sale VALUES(1004,#5/4/2022#,300);

INSERT INTO Sale VALUES(1005,#5/14/2022#,120);

INSERT INTO Sale VALUES(1006,#8/14/2022#,30);

—-----------------------------------------------------------------------------------------------------------------------

CREATE TABLE Services (

serviceID NUMBER NOT NULL,

service\_name VARCHAR(20),

service\_duration NUMBER,

service\_price NUMBER,

saleID NUMBER,

CONSTRAINT pk\_Services PRIMARY KEY (serviceID),

CONSTRAINT fk\_Services FOREIGN KEY (saleID) REFERENCES Sale (saleID));

INSERT INTO Services VALUES(1, 'Dog Grooming', 2, 60,1001);  
INSERT INTO Services VALUES(2, 'Cat Boarding', 1, 25,2004);  
INSERT INTO Services VALUES(3, 'Dog Nail Trimming', 4, 15,2003);  
INSERT INTO Services VALUES(4, 'Dog Bathing', 3,20,1021);  
INSERT INTO Services VALUES(5, 'Skin Exam', 1, 40,1425);

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE Employee(

employeeID NUMBER NOT NULL,

first\_name VARCHAR(20),

last\_name VARCHAR(20),

SSN VARCHAR(20),

phone\_number NUMBER,

mail\_address VARCHAR(20),

occupation VARCHAR(20),

date\_hired DATE,

date\_released DATE,

storeID NUMBER,

CONSTRAINT pk\_Employee PRIMARY KEY (employeeID),

CONSTRAINT fk\_Employee FOREIGN KEY (storeID) REFERENCES Store (storeID));

​​INSERT INTO Employee VALUES(104, 'Emma', 'Davis', '234-56-7890', 6059876543, 'emma.davis@email.com', 'salesperson', #02-MAR-10#, NULL, 1002);

INSERT INTO Employee VALUES(105, 'Michael', 'Wilson', '345-67-8901', 6051234567, 'michael.wilson@email.com', 'technician', #03-MAR-11#, NULL, 1003);

INSERT INTO Employee VALUES(106, 'Sophia', 'Brown', '456-78-9012', 6052345678, 'sophia.brown@email.com', 'administrator', #04-MAR-12#, NULL, 1004);

INSERT INTO Employee VALUES (107, 'William', 'Anderson', '567-89-0123', 6053456789, 'william.anderson@email.com', 'salesperson', #05-MAR-13#, NULL, 1005);

INSERT INTO Employee VALUES (108, 'Olivia', 'Garcia', '678-90-1234', 6054567890, 'olivia.garcia@email.com', 'technician', #06-MAR-14#, NULL, 1001);

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE Inventory(

product\_ID NUMBER NOT NULL,

product\_name VARCHAR(20),

product\_description VARCHAR(20),

quantity\_onhand NUMBER,

category VARCHAR(20),

price NUMBER,

cost NUMBER,

expiration\_date DATE,

storeID NUMBER,

CONSTRAINT pk\_Inventory PRIMARY KEY (product\_ID),

CONSTRAINT fk\_Inventory FOREIGN KEY (storeID) REFERENCES Store (storeID));

INSERT INTO Inventory VALUES(1, 'Dog Food', 'Premium dog food', 50, 'Pet Food', 10.99, 8.5, #2023-12-31#, 1001);

INSERT INTO Inventory VALUES(2, 'Cat Litter', 'Clumping cat litter', 20, 'Pet Supplies', 7.99, 5.75, #2024-06-30#, 1001);

INSERT INTO Inventory VALUES(3, 'Fish Tank', 'Glass fish tank - 10 gallons', 5, 'Aquarium', 29.99, 22.5, NULL, 1002);

INSERT INTO Inventory VALUES(4, 'Bird Cage', 'Large bird cage with accessories', 2, 'Pet Supplies', 79.99, 65.0, NULL, 1002);

INSERT INTO Inventory VALUES(5, 'Hamster Wheel', 'Exercise wheel for hamsters', 15, 'Pet Supplies', 4.99, 3.0, NULL, 1003);

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE Company(

company\_name VARCHAR(50) NOT NULL,

address VARCHAR(20),

email VARCHAR(20),

phone\_number NUMBER,

CONSTRAINT pk\_Company PRIMARY KEY (company\_name));

INSERT INTO Company VALUES ('ABC Corporation', '123 Main Street', 'info@abccorp.com', 5551234567);

INSERT INTO Company VALUES ('DEF Company', '987 Maple Court', 'info@defcompany.com', 5552345678);

INSERT INTO Company VALUES ('LMK Enterprise', '124 Z Avenue', 'contact@lmk.com', 5552462451);  
INSERT INTO Company VALUES ('LP Enterprises', '321 Pine Lane', 'contact@lmnenterprises.com', 5557890123);

INSERT INTO Company VALUES ('POP Ltd.', '478 Elm Street', 'hr@pop.com', 4651232564);  
INSERT INTO Company VALUES ('PQR Ltd.', '789 Oak Road', 'info@pqrltd.com', 5554567890);  
INSERT INTO Company VALUES ('XYZ Industries', '456 Elm Avenue', 'contact@xyzindustries.com', 5559876543);

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE Supplier(

supplierID NUMBER NOT NULL,

first\_name VARCHAR(20),

last\_name VARCHAR(20),

company\_name VARCHAR (20),

CONSTRAINT pk\_Supplier PRIMARY KEY (supplierID),

CONSTRAINT fk\_Supplier FOREIGN KEY (company\_name) REFERENCES Company(company\_name));

INSERT INTO Supplier VALUES (101, 'Jane', 'Faner', 'ABC Corporation');

INSERT INTO Supplier VALUES (102, 'Peter', 'Brown', 'LMK Enterprise');

INSERT INTO Supplier VALUES (103, 'Susie', 'Gil', 'LP Enterprises');

INSERT INTO Supplier VALUES (104, 'William', 'Lesner', 'XYZ Industries');

INSERT INTO Supplier VALUES (105, 'Sam', 'Bernard', 'DEF Company');

INSERT INTO Supplier VALUES (106, 'Matt', 'Brown', 'POP Ltd.');

INSERT INTO Supplier VALUES (107, 'Emma', 'Stone', 'PQR Ltd.');

—----------------------------------------------------------------------------------------------------------------------------------------

CREATE TABLE Saleitem(

saleID NUMBER NOT NULL,

sales\_date DATE NOT NULL,

quantity NUMBER,

sale\_price NUMBER,

product\_ID NUMBER,

CONSTRAINT pk\_Saleitem PRIMARY KEY (saleID, sales\_date),

CONSTRAINT fk\_Saleitem FOREIGN KEY (product\_ID) REFERENCES Inventory (product\_ID)),

CONSTRAINT fk1\_Saleitem FOREIGN KEY (saleID) REFERENCES Sale (saleID);

INSERT INTO Saleitem VALUES (2010, #1/21/2023#,1,7.99,2);

INSERT INTO Saleitem VALUES (2011, #3/18/2023#,1,79.99,4);

INSERT INTO Saleitem VALUES (2013, #1/1/2023#,1,4.99,5);

INSERT INTO Saleitem VALUES (2014, #2/9/2023#,1,4.99,5);

INSERT INTO Saleitem VALUES (2015, #4/9/2023#,2,7.99,2);

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE Payment(

Invoice\_number NUMBER NOT NULL,

payment\_terms VARCHAR (40),

payment\_type VARCHAR (40),

payment\_date DATE,

saleID NUMBER,

CONSTRAINT pk\_Payment PRIMARY KEY (Invoice\_number),

CONSTRAINT fk\_Payment FOREIGN KEY (saleID) REFERENCES Sale (saleID));

INSERT INTO Payment VALUES (801, 'NET 0', 'Cash', #5/1/2023#,1001);

INSERT INTO Payment VALUES (802, 'NET 0', 'Cash', #4/3/2021#,1003);

INSERT INTO Payment VALUES (803, 'NET 0', 'Credit Card', #5/4/2022#,1004);

INSERT INTO Payment VALUES (804, 'NET 0', 'Credit Card', #5/14/2022#,1005);

INSERT INTO Payment VALUES (805, 'NET 0', 'Credit Card', #8/14/2022#,1006);

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE Pet(

petID NUMBER NOT NULL,

pet\_type VARCHAR(20),

breed VARCHAR(20),

gender VARCHAR(20),

color VARCHAR(20),

date\_born DATE,

microchipped VARCHAR(20),

medications VARCHAR(20),

registration\_date DATE,

customerID NUMBER,

CONSTRAINT pk\_Pet PRIMARY KEY (petID),

CONSTRAINT fk\_Pet FOREIGN KEY (customerID) REFERENCES Customer (customerID));

INSERT INTO Pet VALUES(1011, 'Dog', 'Labrador Retriever', 'Male', 'Black', #2019-05-10#, 'Yes', 'None', #2021-01-15#,1);

INSERT INTO Pet VALUES (1012, 'Cat', 'Siamese', 'Female', 'Seal Point', #2020-02-28#, 'No', 'Vaccination', #2021-03-02#,2);

INSERT INTO Pet VALUES (1013, 'Bird', 'Parakeet', 'Male', 'Green', #2020-07-07#, 'Yes', 'None', #2020-08-15#,3);

INSERT INTO Pet VALUES (1014, 'Fish', 'Goldfish', 'Female', 'Orange', #2020-09-12#, 'No', 'Fish Food', #2020-09-15#,4);

INSERT INTO Pet VALUES (1015, 'Rabbit', 'Dwarf Hotot', 'Female', 'White', #2018-11-20#, 'Yes', 'None', #2019-01-05#,5);

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE VisitDetails(

summaryID NUMBER NOT NULL,

appointment\_summary VARCHAR(20),

prescriptions VARCHAR(20),

treatment VARCHAR(20),

petID NUMBER,

serviceID NUMBER,

CONSTRAINT pk\_VisitDetails PRIMARY KEY (summaryID),

CONSTRAINT fk\_VisitDetails FOREIGN KEY (petID) REFERENCES Pet (petID),

CONSTRAINT fk1\_VisitDetails FOREIGN KEY (appointmentID) REFERENCES Visited (appointmentID),

CONSTRAINT fk2\_VisitDetails FOREIGN KEY (serviceID) REFERENCES Services (serviceID));

INSERT INTO VisitDetails VALUES (1, ‘1001’, 'Flea Control', 'Flea spray', '2 times a day', 1011, 16);

INSERT INTO VisitDetails VALUES (2, ‘1002’, 'Microchipping', 'None', 'None', 1014, 18);

INSERT INTO VisitDetails VALUES (3, ‘1004’, 'FeLV Vaccine', 'None', 'None', 1023, 17);

INSERT INTO VisitDetails VALUES (4,’1007’, 'Cat Boarding, 'None', 'None', 1012, 2);

\*\*\* During table creation we decided that we should have appointmentID as an foreign key in VisitDetails Table in order to have a relationship ​​between them

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE Inventory\_Supplier(

product\_ID NUMBER NOT NULL,

supplierID NUMBER NOT NULL,

CONSTRAINT pk\_Inventory\_Supplier PRIMARY KEY (product\_ID, supplierID),

CONSTRAINT fk\_Inventory\_Supplier FOREIGN KEY (product\_ID) REFERENCES Inventory (product\_ID),

CONSTRAINT fk1\_Inventory\_Supplier FOREIGN KEY (supplierID) REFERENCES Supplier (supplierID));

INSERT INTO Inventory\_Supplier VALUES (1, 101);

INSERT INTO Inventory\_Supplier VALUES (2, 103);

INSERT INTO Inventory\_Supplier VALUES (3, 107);

INSERT INTO Inventory\_Supplier VALUES (4, 106);

INSERT INTO Inventory\_Supplier VALUES (5, 106);

\*\*\* We didn’t create this table because in case of necessity we can easily write a query linking inventory and supplier to each other.

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE Visited(

appointmentID NUMBER NOT NULL,

appointment\_time TIME,

appointment\_date DATE,

customerID NUMBER,

CONSTRAINT pk\_Visited PRIMARY KEY (appointmentID),

CONSTRAINT fk\_Visited FOREIGN KEY (customerID) REFERENCES Customer (customerID));

INSERT INTO Visited VALUES (1001, #09:00:00#, #6/30/2022#, 20);

INSERT INTO Visited VALUES (1002, #12:00:00#, #2/19/2023#, 4);

INSERT INTO Visited VALUES (1003, #9:30:00#, #9/14/2021#, 5);

INSERT INTO Visited VALUES (1004, #10:00:00#, #6/15/2022#, 12);

INSERT INTO Visited VALUES (1005, #9:00:00#, #8/14/2022#, 14);

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE Cancelled(

appointmentID NUMBER NOT NULL,

appointment\_time TIME,

appointment\_date DATE,

customerID NUMBER,

CONSTRAINT pk\_Cancelled PRIMARY KEY (appointmentID),

CONSTRAINT fk\_Cancelled FOREIGN KEY (customerID) REFERENCES Customer (customerID));

INSERT INTO Cancelled VALUES (1012, #9:30:00#, #3/24/2023#, 10);

INSERT INTO Cancelled VALUES (1013, #10:30:00#, #2/19/2023#, 18);

INSERT INTO Cancelled VALUES (1014, #12:00:00#, #12/21/2022#, 11);

—------------------------------------------------------------------------------------------------------------------

CREATE TABLE ServiceRendered(

lineitemnumber NUMBER NOT NULL,

serviceID NUMBER,

appointmentID NUMBER,

CONSTRAINT pk\_ServiceRendered PRIMARY KEY (lineitemnumber),

CONSTRAINT fk\_ServiceRendered FOREIGN KEY (serviceID) REFERENCES Services (serviceID),

CONSTRAINT fk1\_ServiceRendered FOREIGN KEY (appointmentID) REFERENCES Visited (appointmentID));

INSERT INTO ServiceRendered VALUES (1, 16,1001);

INSERT INTO ServiceRendered VALUES (2, 18,1002);

INSERT INTO ServiceRendered VALUES (3, 17,1004);

INSERT INTO ServiceRendered VALUES (4, 2,1007);

INSERT INTO ServiceRendered VALUES (5, 6,1008);

—------------------------------------------------------------------------------------------------------------------

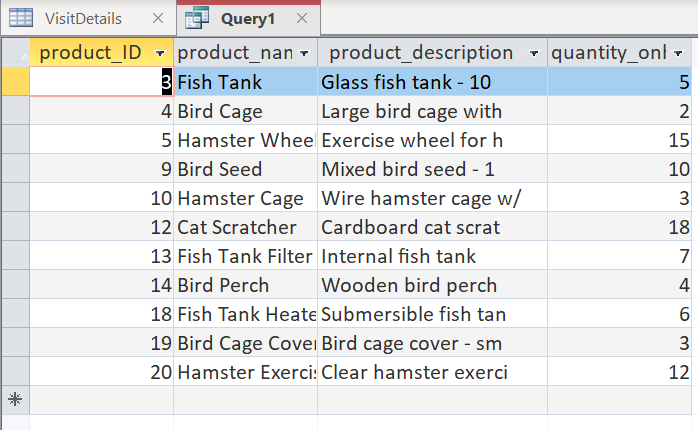
**Scenarios**

1. Write a query to display product\_ID, product\_name, product\_description, quantity\_onhand for an inventory where quantity\_onhand is less than 20

SELECT product\_ID, product\_name, product\_description, quantity\_onhand

FROM Inventory

WHERE quantity\_onhand < 20;



1. Write a query to display customerID, first\_name, last\_name, phone\_number, address,

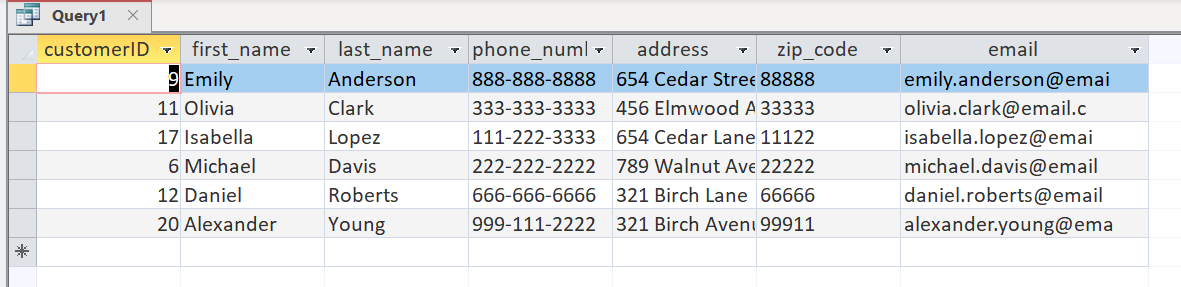
zip\_code, email where first name has letter L in the name and sort them by last name

SELECT customerID, first\_name, last\_name, phone\_number, address, zip\_code, email

FROM Customer

WHERE first\_name Like "\*L\*"

ORDER BY last\_name ASC



1. Write a query to display serviceID,service\_name, service\_duration,service\_price,saleID for the shortest service\_durations.

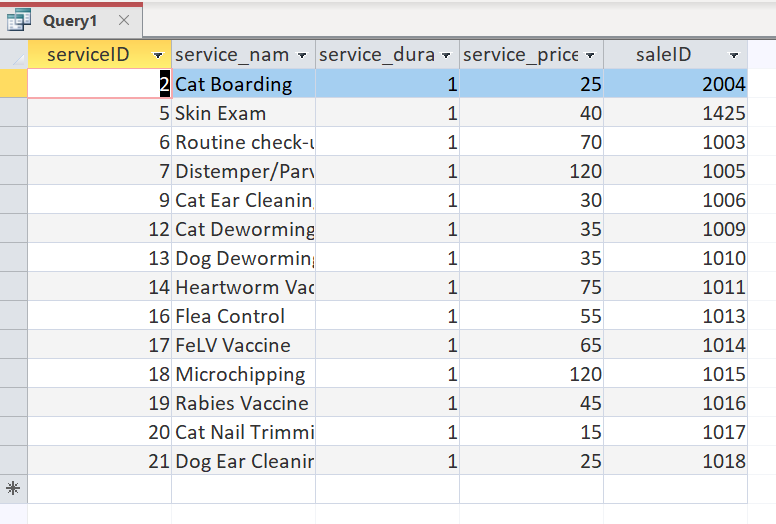
SELECT serviceID,service\_name, service\_duration,service\_price,saleID

FROM Services

WHERE service\_duration =(

SELECT min(service\_duration)

FROM Services);

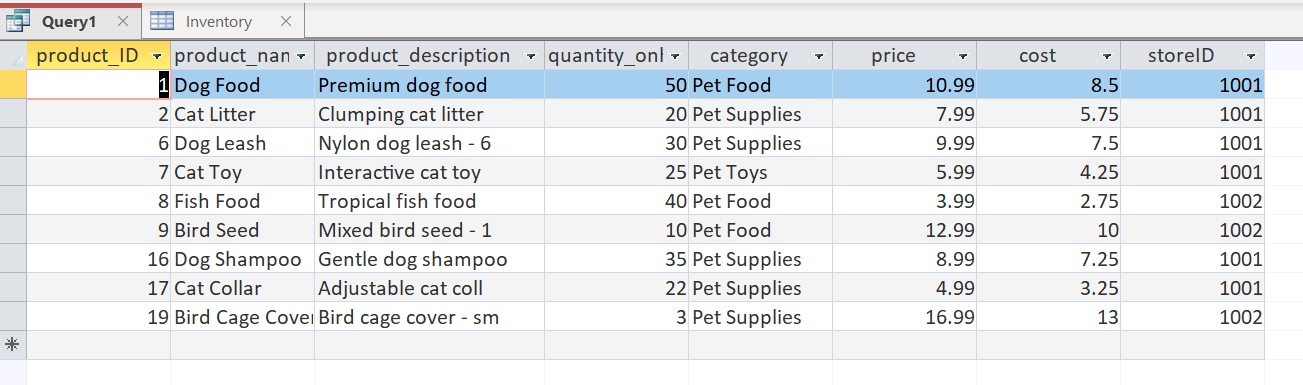


1. Write a query to display product\_ID, product\_name, product\_description, quantity\_onhand, category, price, cost, storeID for an inventory where price <20, product name starts with pet and belongs either 1001 or 1002 storeID

SELECT product\_ID, product\_name,product\_description, quantity\_onhand,category,price,cost,storeID

FROM Inventory

WHERE price <20 AND category Like "Pet\*" AND storeID IN(1001,1002);

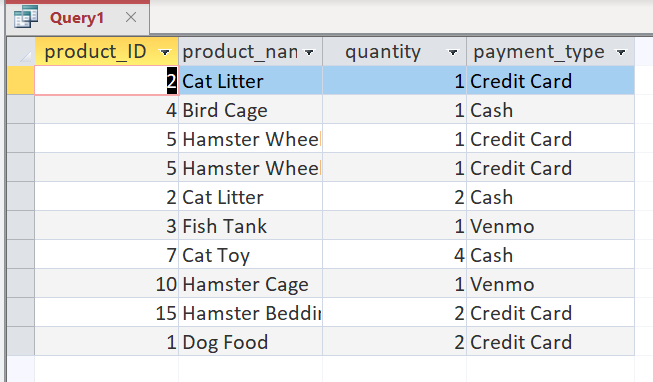


1. Write a query to display what items been sold in what quantity and what was the payment type

SELECT Inventory.product\_ID,Inventory.product\_name,Saleitem.quantity,Payment.payment\_type

FROM Saleitem, Payment, Inventory

WHERE Saleitem.saleID=Payment.saleID AND Saleitem.product\_ID=Inventory.product\_ID;



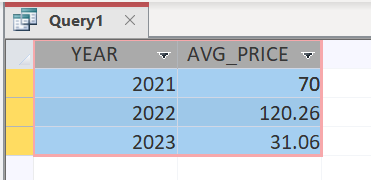
1. Write a query to display average sales per year.

SELECT YEAR (Payment.payment\_date) AS YEAR, ROUND(AVG(sale\_total\_price),2) AS AVG\_PRICE

FROM Sale, Payment

WHERE Sale.saleID=Payment.saleID

GROUP BY YEAR (Payment.payment\_date);



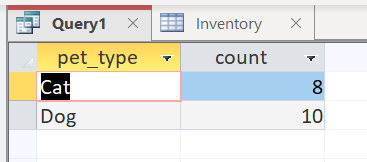
1. Write a query using HAVING to count and group pet types that have more than 3.

SELECT pet\_type, count(pet\_type) AS [count]

FROM Pet

GROUP BY pet\_type

HAVING (count(pet\_type))>3;



**Conclusion**

After weeks of work, we were able to create a database that meets the needs of our client, *Paws ‘n’ All.* We started our project by brainstorming the information that the database would need to store and the tables that we would have to create. Then, we identified the different entities that were needed and their corresponding data type. After doing so, we were ready to create our ERD diagram, which outlined the relationships between the entities. This part was somewhat challenging as we had to keep the needs of our client in mind while also understanding which relationships were the most logical to create.

After we finalized our ERD, we were ready to create our database. As we created the database, we made some minor changes that logically made sense to us and would enhance the user experience while navigating the database. In conclusion, the database we created houses the appropriate information and provides our client with an effective and convenient way to manage the business’s data.