**AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH (AIUB)**

**FACULTY OF SCIENCE & TECHNOLOGY**

A picture containing calendar

Description automatically generated

Course Title

**INTRODUCTION TO DATABASE (CSC2108)**

**Semester: Fall 24-25  
Section: Q**

**TITLE**

**Pet Shop Management System**

**Supervised By**

Saikat Baul

**Submitted By: Group no: 10**

|  |  |
| --- | --- |
| **Name** | **ID** |
| Satadru Barua Jeet | 23-52224-2 |
| Omma Habiba | 23-52177-2 |
| Samiya Siddiqua Sympa | 23-52240-2 |
| Ramisa Anan Hridi | 23-51728-2 |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **TOPICS** | | **Page no.** |
| **Title Page** | | **1** |
| **Table of Content** | | **2** |
| **1.** | **Introduction** | **3** |
| **2.** | **Case Study** | **4** |
| **3.** | **ER Diagram** | **5** |
| **4.** | **Normalization** | **6-14** |
| **5.** | **Finalization** | **15** |
| **6.** | **Table Creation** | **16-22** |
| **7.** | **Data Insertion** | **23-29** |
| **8.** | **Query Test** | **30-34** |
| **9.** | **DB connection** | **35** |
|  |  |  |
|  |  |  |

# **Introduction**

The Pet Shop Management System is designed to make running a pet shop easier by keeping track of customers, staff, inventory, sales, and payment information. The main goal of the system is to automate and simplify the management of pets, supplies, and transactions, making it accurate, clear, and user-friendly for shop owners and staff. Here we used MySQL to create and manage the database. We used DDL to set up the structure (tables, columns, etc.) and DML to work with the data (insert, update, and retrieve data), & Java for connecting the database to the user interface. It helps users interact smoothly with the system. We use SQL queries to do tasks like finding available pets, updating stock, processing sales, and creating reports. The main goal of this project is to create an easy-to-use and efficient system for managing everything in a pet shop. This includes Better Efficiency, reducing manual data entry to lower the chances of errors. Improved Inventory Management, Better Customer Service & Creating reports on sales, inventory, and customer habits to help the pet shop make smart business choices. Beneficiaries: Pet Shop Owners/Managers: They are the main users who will enjoy better efficiency, Pet Shop Staff: They will use the system to handle sales, manage inventory, and access customer details, Customers: They will benefit indirectly from quicker service and more accurate information. The system is aimed at pet shop owners, managers, and staff, and it's designed to be easy to use for people with basic computer skills.

# **Case Study / Scenario**

In a Pet Shop Management System manage the sale of pets and supplies, along with inventory, staff, and customer details. Customers may purchase pets, supplies, each purchase is associated with exactly one Customer, who is identified by a **Customer ID**. The system also stores **Customer Name, Phone, Email, Address.** The shop manages sales which is identified by a Sale ID with details like **Total Amount, Date, Availability**. A Customer can make one or more Sales, and each Sale is linked to exactly one Customer. Each Pet is uniquely identified by a **Pet ID**. Attributes include **Pet Type, Breed, Age, Price,** **Availability**. The system ensures that a pet cannot be sold more than once. Each inventory Supply is uniquely identified by a **Supply ID**. The system stores **Supply Name, Category, Price, Quantity**. A Sale may include multiple Pets and Supplies. A Pet can be included in only one Sale to ensure it is not sold more than once. A Supply may be included in multiple Sales. **Payment** is made by a customer which is identified by a **Payment ID**. Attributes include **Payment Method, Amount.** The system stores **Payment Date**. Each Sale must be associated with one Payment and a Payment is linked to only one Sale. The Sales are handled by Staffs who are identified by **Staff ID**. It also includes **Staff Name, Age, Role, Salary, Staff Phone Number**. A Sale is handled by exactly one Staff Member and a Staff Member can handle multiple Sales. This system ensures efficient operation of the pet shop by managing customers, supplies, sales, and payments efficiently.

# **ER Diagram**

**A diagram of a company

Description automatically generated**

# **Normalization**

1. **Makes relation between Customer & Sales**

UNF:

CusID, CusName, Email, PNo, City, HouseNo, Area, SaleID, SaleDate, tAmount, Availability.

1NF:

CusID, CusName, Email, PNo, City, HouseNo, Area, SaleID, SaleDate, tAmount, Availability.

2NF:

Customer: CusID (pk), CusName, Email, City, HouseNo, Area.

Customer Phone: CusID (fk), PNo(pk)

Sales: SaleID(pk), SaleDate, tAmount, Availability.

3NF:

Customer: CusID (pk), CusName, Email, City, HouseNo, Area.

Customer Phone: CusID (fk), PNo(pk)

Sales: SaleID(pk), SaleDate, tAmount, Availability.

**2. Processes Relationship between Sales & Payment**

UNF:

SaleID, SaleDate, tAmount, Availability, PayID, PayDate, PayMethod, PayAmount

1NF:

SaleID, SaleDate, tAmount, Availability, PayID, PayDate, PayMethod, PayAmount

2NF:

Sales: SaleID(pk), SaleDate, tAmount, Availability

Payment: PayID (pk), PayMethod, PayAmount

Proceeds: SaleID(fk), PayID (fk), PayDate

3NF:

Sales: SaleID(pk), SaleDate, tAmount, Availability

Payment: PayID (pk), PayMethod, PayAmount

Proceeds: SaleID(fk), PayID (fk), PayDate

**3. Handled By Relationship between Sales & Staff**

UNF:

SaleID, SaleDate, tAmount, Availability, StaffID, Staffname, Age, Salary, DOB, Role, StaffPNo

1NF:

SaleID, SaleDate, tAmount, Availability, StaffID, Staffname, Age, Salary, DOB, Role, StaffPNo

2NF:

Sales: SaleID(pk), SaleDate, tAmount, Availability

Staff: StaffID (pk), StaffName, Age, Salary, DOB, Role

Staff Phone Number: StaffID (fk), StaffPNo(pk)

3NF:

Sales: SaleID(pk), SaleDate, tAmount, Availability

Staff: StaffID (pk), StaffName, Age, Salary, DOB, Role

Staff Phone Number: StaffID (fk), StaffPNo(pk)

**4. Includes Relationship between Sales & Pets**

UNF:

SaleID, SaleDate, tAmount, Availability, PetID, Age, Breed, PetType, Price

1NF:

SaleID, SaleDate, tAmount, Availability, PetID, Age, Breed, PetType, Price

2NF:

Sales: SaleID(pk), SaleDate, tAmount, Availability

Pets: PetID(pk), Age, Breed, PetType, Price

3NF:

Sales: SaleID(pk), SaleDate, tAmount, Availability

Pets: PetID(pk), Age, Breed, PetType, Price

**5. Includes Relationship between Sales & Supply**

UNF:

SaleID, SaleDate, tAmount, Availability, SupplyID, SupplyName, Price, Category, Quantity

1NF:

SaleID, SaleDate, tAmount, Availability, SupplyID, SupplyName, Price, Category, Quantity

2NF:

Sales: SaleID(pk), SaleDate, tAmount, Availability

Supply: SupplyID(pk), SupplyName, Price, Category, Quantity

3NF:

Sales: SaleID(pk), SaleDate, tAmount, Availability

Supply: SupplyID(pk), SupplyName, Price, Category, Quantity

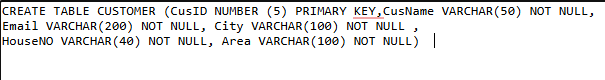
# **Finalization**

1. CusID (pk), CusName, Email, City, HouseNo, Area.
2. CusID (fk), PNo(pk)
3. SaleID(pk), SaleDate, tAmount, Availability.
4. PayID (pk), PayMethod, PayAmount
5. SaleID(fk), PayID (fk), PayDate
6. StaffID (pk), StaffName, Age, Salary, DOB, Role
7. StaffID (fk), StaffPNo(pk)
8. PetID(pk), Age, Breed, PetType, Price
9. SupplyID(pk), SupplyName, Price, Category, Quantity

# **Table Creation (DDL Operations)**

|  |  |  |
| --- | --- | --- |
| StudentID1: 23-52224-2 Name: Satadru Barua Jeet | StudentID3: 23-52240-2 Name: Samiya Siddiqua Sympa | |
| StudentID2: 23-52177-2 Name: Omma Habiba | StudentID4: 23-51728-2 Name: Ramisa Anan Hridi | |
| **CO4**: Creating DML, DDL using Oracle and connection with ODBC/JDBC for existing JAVA application | | |
| **PO-e-2:** Use modern engineering and IT tools for prediction and modeling of complex computer science and engineering problem | | Marks |

**Customer Table:**



A screenshot of a data type

Description automatically generated

**Customer Phone No Table:**

A close up of words

Description automatically generated

A screenshot of a computer

Description automatically generated

**Sales Table:**

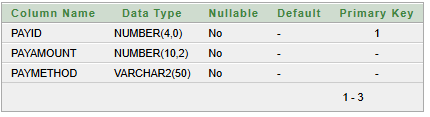


A screenshot of a computer

Description automatically generated

**Payment Table:**





**Proceeds Table:**

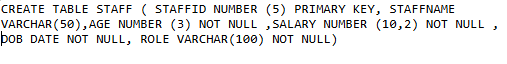
A black and white sign with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

**Staff Table:**



A screenshot of a data

Description automatically generated

**Staff Phone No Table:**



A screenshot of a computer

Description automatically generated

**Pet Table:**

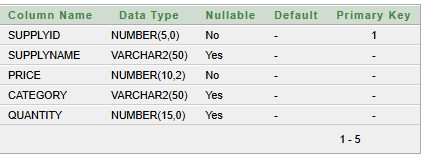


A screenshot of a data table

Description automatically generated

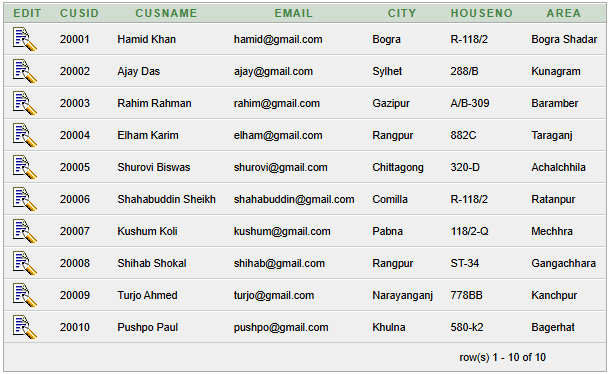
**Supply Table:**





# **Inserted Values in the tables**

**Customer Table:**

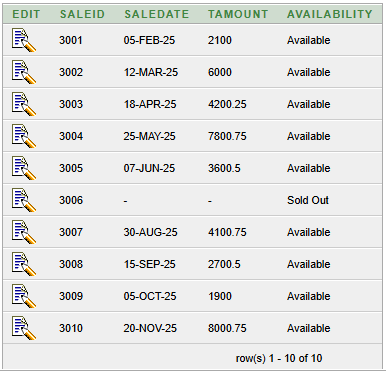


**Customer Phone No Table:**

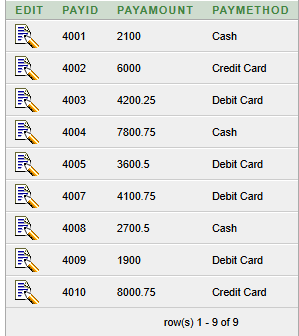
A screenshot of a computer

Description automatically generated

**Sales Table:**



**Payment Table:**



**Proceeds Table:**

A screenshot of a computer

Description automatically generated

**Staff Table:**

A table with numbers and numbers

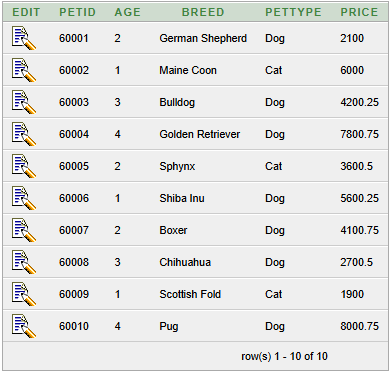
Description automatically generated

**Staff Phone No Table:**

A screenshot of a computer

Description automatically generated

**Pet Table:**



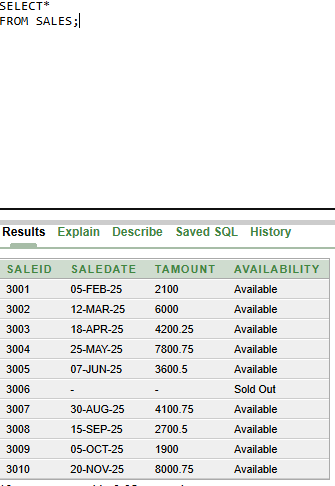
**Supply Table:**

A table with price list

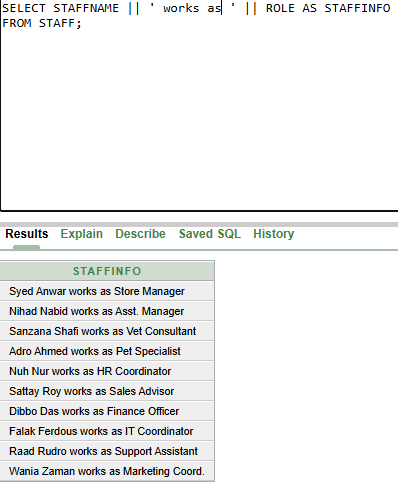
Description automatically generated

# **Query Test in DB**

1. A Simple Query



1. Concatenation

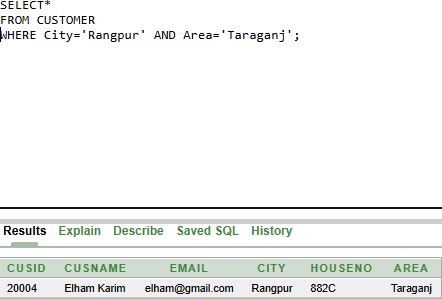


1. WHERE condition

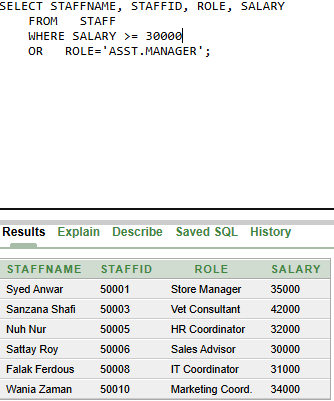
A screenshot of a computer

Description automatically generated

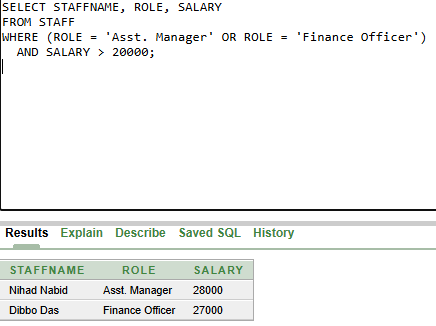
1. WHERE & AND



1. WHERE & OR



1. WHERE & AND & OR MIXED

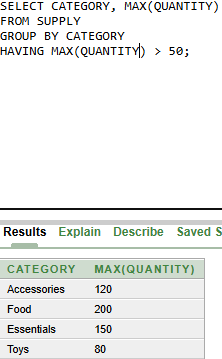


1. GROUP BY

A screenshot of a computer

Description automatically generated

1. GROUP BY and HAVING

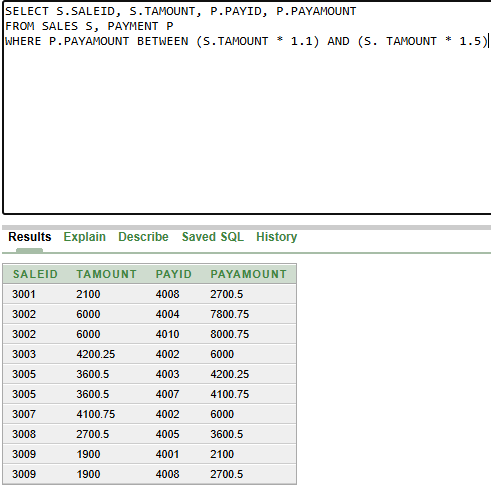


1. Equijoin

A screenshot of a computer

Description automatically generated

1. Non-Equijoin

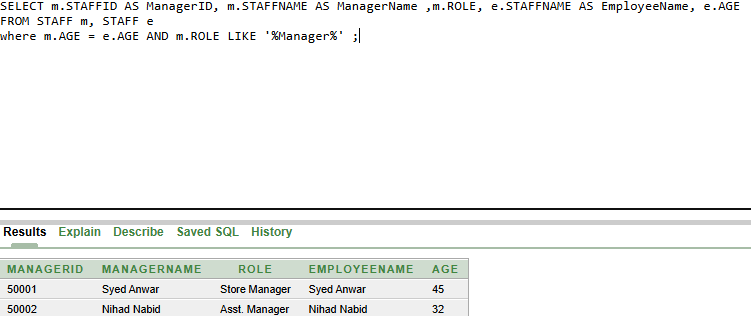


1. Outer Join

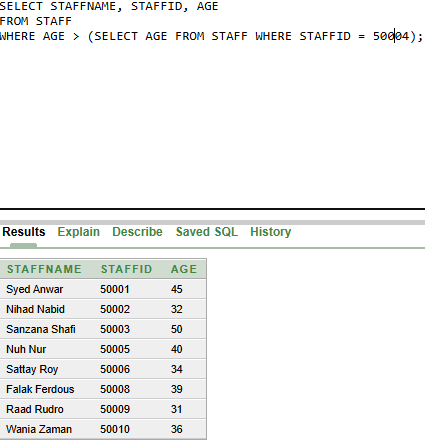
A screenshot of a computer

Description automatically generated

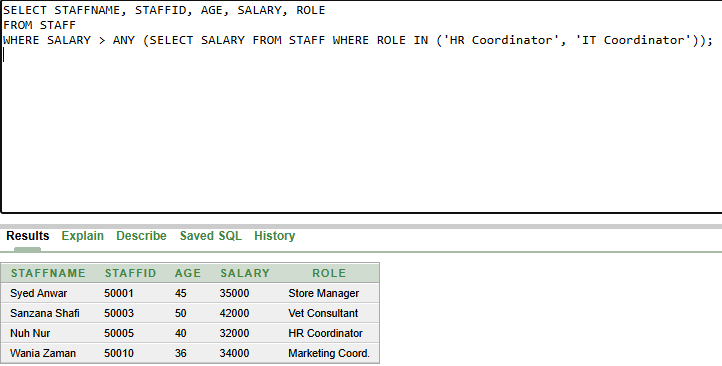
1. Self-Join



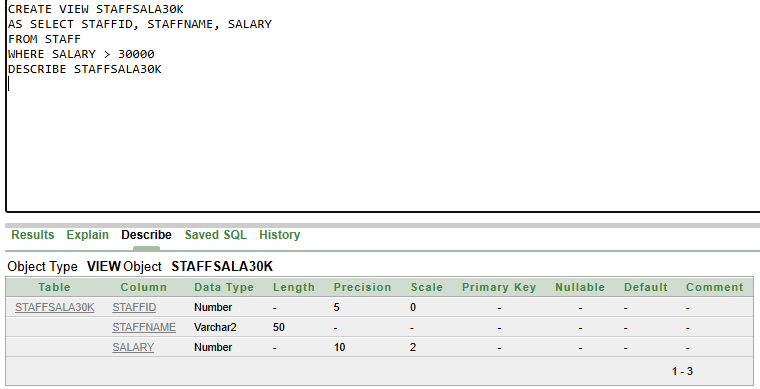
1. Single Subquery



1. Multiple Row Subquery



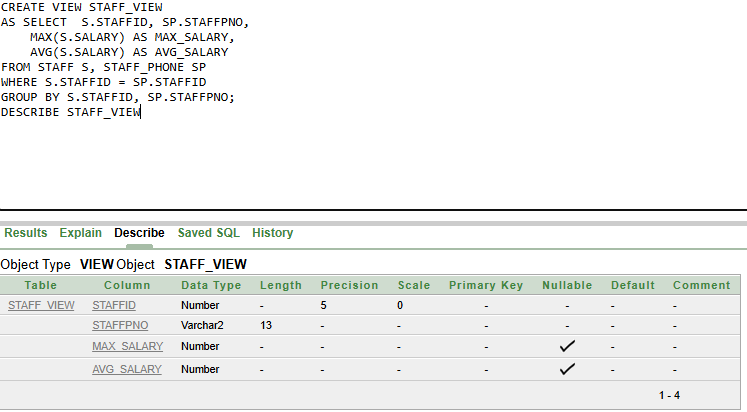
17. Simple View

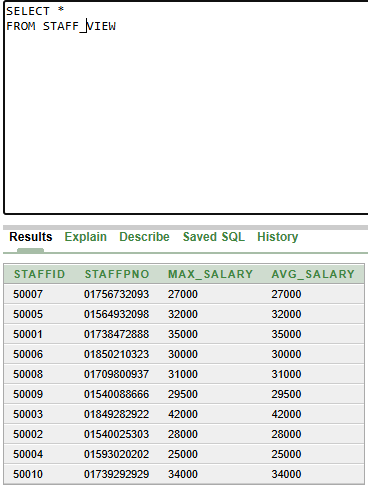


A screenshot of a computer

Description automatically generated

18. Complex View





# **Description of a Successful DB connection**

**Database connection of Customer Table:**

Name: Satadru Barua Jeet

Student ID: 23-52224-2

To integrate MySQL with Java, I began by downloading the MySQL Java Connector (JAR file), which facilitates communication between Java applications and MySQL databases. I searched for the appropriate version using Maven and added the downloaded JAR file to my Eclipse project to enable database connectivity.

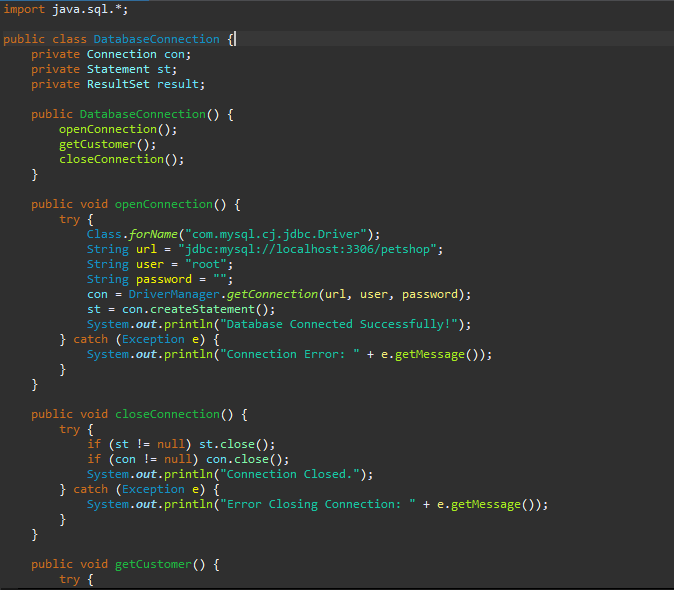
Next, I installed XAMPP, a comprehensive software package that includes Apache and MariaDB (a MySQL-compatible database). After completing the installation, I launched the XAMPP Control Panel and started both Apache and MySQL services. Once they were running, I accessed the MySQL Admin Panel from XAMPP to manage my database.

In the MySQL Admin Panel, I created a database named **petshop** and added a **customer** table within it. The table contained essential columns such as cusid, cusname, email, houseno, city, and area. To test the setup, I inserted sample records into the table, simulating real-world customer data.

To establish a connection between my Java application and the MySQL database, I loaded the MySQL JDBC driver using Class.forName(). Then, using DriverManager, I created a connection to the **petshop** database by providing the required credentials. To verify the connection, I executed a SELECT query to retrieve data from the **customer** table and displayed the results in the console.

By following these steps, I successfully integrated MySQL with Java and performed basic database operations like fetching and displaying records.

.



A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated





A screen shot of a computer program

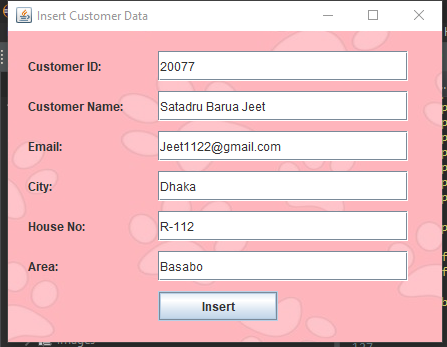
Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen shot of text

Description automatically generated





**Database Connection of Pet Table:**

Name: Omma Habiba

Student ID: 23-52177-2

To establish database connectivity in my Java project, I first downloaded the MySQL Java Connector (JAR file). I searched for the latest compatible version using Maven and imported the JAR file into my Eclipse project, enabling smooth interaction between Java and MySQL.

Next, I installed XAMPP, which provides an easy-to-use environment with MySQL (MariaDB) and Apache. After installation, I opened the XAMPP Control Panel and started both the Apache and MySQL services. With MySQL running, I accessed the MySQL Admin Panel to set up and manage my database.

Within the Admin Panel, I worked with an existing database named **petshop** and created a **pet** table with columns such as **petid, age, breed, pettype, and price**. To validate the setup, I inserted sample records into the table for testing.

On the Java side, I used Class.forName() to load the MySQL JDBC driver and DriverManager.getConnection() to establish a connection to the **petshop** database. Once the connection was successfully established, I executed a SELECT query to fetch data from the **pet** table and displayed the retrieved records in the console.

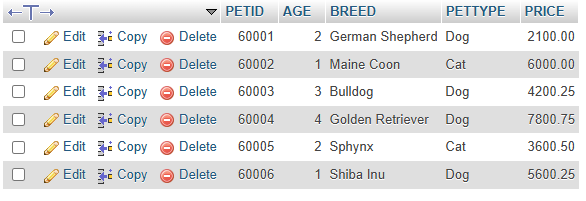
Through this process, I successfully connected my Java application to MySQL and performed basic database operations such as data retrieval and display.



A computer screen shot of a program code

Description automatically generated

A screenshot of a computer

Description automatically generated 

**Database Connection of Supply Table:**

Name: Ramisa Anan Hridi

Student ID: 23-51728-2

To set up database connectivity in my Java project, I began by downloading the MySQL Java Connector (JAR file). I searched for the latest compatible version using Maven and imported the JAR file into my Eclipse project, ensuring seamless communication between Java and MySQL.

Next, I installed XAMPP, a user-friendly package that includes MySQL (MariaDB) and Apache. After completing the installation, I launched the XAMPP Control Panel and started both the Apache and MySQL services. Once MySQL was running, I accessed the MySQL Admin Panel to configure and manage my database.

Within the Admin Panel, I worked with an existing database named **petshop** and created a **supply** table with columns such as **supplyid, supplyname, price, category, and quantity**. To verify the setup, I inserted sample records into the table for testing purposes.

On the Java side, I used Class.forName() to load the MySQL JDBC driver and DriverManager.getConnection() to establish a connection to the **petshop** database. Once the connection was successfully established, I executed a SELECT query to retrieve data from the **supply** table and displayed the fetched records in the console.

By following these steps, I successfully linked my Java application with MySQL, allowing me to perform essential database operations such as retrieving and displaying records.



A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated 

**Database Connection of Staff Phone Number Table:**

Name: Samiya Siddiqua Sympa

Student ID: 23-52240-2

To establish database connectivity in my Java project, I began by downloading the MySQL Java Connector (JAR file). I searched for the latest compatible version using Maven and imported the JAR file into my Eclipse project to facilitate seamless communication between Java and MySQL.

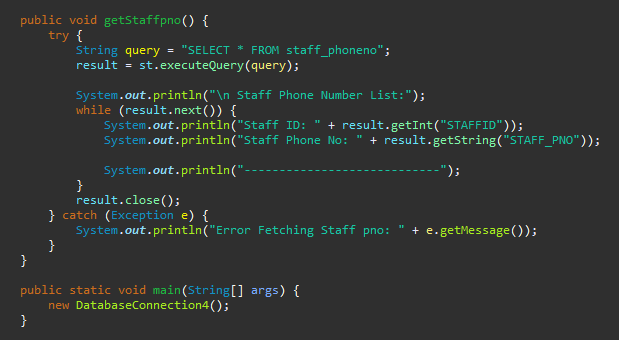
Next, I installed XAMPP, a software suite that includes MySQL (MariaDB) and Apache. After completing the installation, I opened the XAMPP Control Panel and started both the Apache and MySQL services. With MySQL up and running, I accessed the MySQL Admin Panel to configure and manage my database.

Within the Admin Panel, I worked with an existing database named **petshop** and created a table called **staff\_phone\_no** with columns **staffid** and **staffp\_no**. To verify the setup, I inserted sample records into the table for testing.

On the Java side, I used Class.forName() to load the MySQL JDBC driver and DriverManager.getConnection() to establish a connection to the **petshop** database. After successfully connecting, I executed a SELECT query to retrieve data from the **staff\_phone\_no** table and displayed the results in the console.

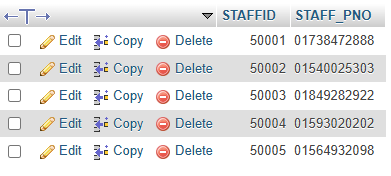
By following these steps, I successfully connected my Java application to MySQL, allowing me to perform fundamental database operations such as retrieving and displaying staff phone numbers.





A screenshot of a computer

Description automatically generated



# **Conclusion**

The **Pet Shop Management System** was developed to streamline pet store operations by integrating a MySQL database with a Java-based backend. The project involved configuring MySQL using XAMPP, designing essential tables, and establishing a connection with the database via the MySQL Java Connector. This setup enables users to efficiently manage pet records, monitor inventory, and process customer transactions while ensuring data security and real-time updates.

Future improvements to the system include the development of an intuitive user interface using JavaFX or a web-based frontend for a more seamless user experience. Additional enhancements such as advanced inventory tracking and secure payment gateway integration will further enhance operational efficiency. Transitioning the database to a cloud-based platform will enhance scalability and performance, while implementing multi-user role management will strengthen security by defining different access levels for administrators, staff, and customers. Furthermore, expanding the system into mobile app development will enhance accessibility and user engagement.

These proposed upgrades will make the system more efficient, scalable, and user-friendly, ensuring smooth and reliable pet shop management experience.