Java JDBC Introduction

JDBC stand for Java Database Connectivity – a powerful API (Application Programming Interface) that allows Java programs to access and manipulate data stored in a wide variety of database. API handle request and response.

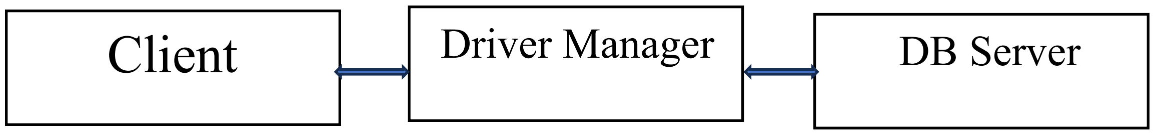
What is JDBC ?

Definition : JDBC is an API that defines how a client can connect to a database, send SQL queries and statements, and retrieve and manipulate the results.

Components : JDBC consists of a set of interfaces and classes written in Java, a JDBC driver that vendors implement to comply with JDBC , and a database server that supports JDBC.

Benefits : JDBC provides platform independence , security , and ease of use for Java developers who need to access databases.

JDBC Architecture :



Client : A JDBC client is any Java Application or applet that connects to a server-based database using JDBC APIs.

Driver Manager : The Driver Manager is a service of the JDBC API that manages the drivers and establishes a connection between a JDBC client and a database server.

Database Server : A DB server is a software program that provides database management functionality within a networked computing environment.

Why JDBC ?

1. JDBC provides a universal API for accessing and interacting with any SQL – compliant Database.

2. Performance

3. Flexibility

JDBC Components :

Drivers , Driver Manager Class , Connection interface , Statement and PreparedStatement interface , ResultSet interface & SQL Exception.

JDBC Drivers :

A) JDBC ODBC Bridge Driver : oldest driver , write in c , performance issue.

B) Native API Driver : Vender Api use in this driver their is issue.

C) Network Protocol Driver : write in java , extra layer use (middle-level)

D) Thin Driver (Only use it..) write in java : This is mostly use this time.

JDBC Driver Manager Class :

The Driver Manager class manages a list of database drivers.

It establishes connections to the database using the appropriate driver and handles the process of loading the driver class.

It provides a standardized method for handling multiple database connection and selecting the appropriate driver.

Connection Interface : is part of driver manager class

Database connection establish [(url) , username , Password].

1.Creation of Statements : The Connection interface is used to create a statement object , which is used to execute SQL queries against the database.

2.Transaction Management : The Connection interface allows transaction to be managed with methods such as commit() and rollback().

3.Retrival of Metadata : The Connection interface provides methods to retrieve metadata from the database, including information about the database structure and the various objects that are defined in it.

Statement Interface : SQL quires run in java code

1.Statement : The Statement interface is used for executing simple SQL queries without parameters.

2. Prepared Statement : Prepared Statement is used for executing precompiled SQL queries with parameters, which can be more efficient and secure than statement object.

3.CallableStatement : It is used for executing database stored procedures. It provides a more efficient way to access them than with SQL statements.

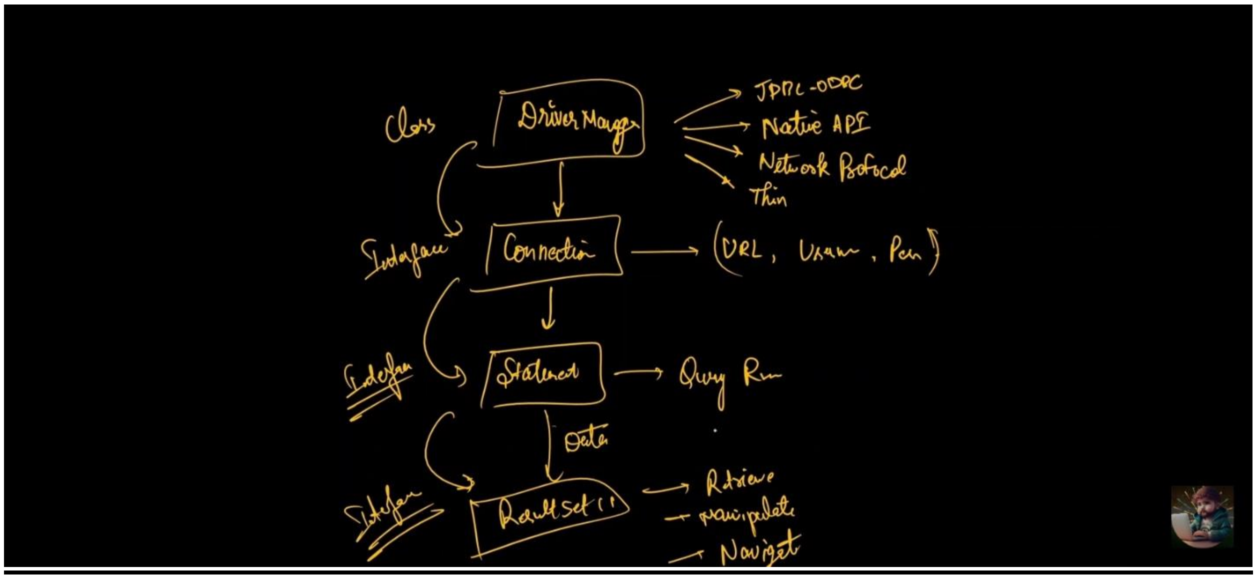
ResultSet : It is Part of Statement

1.Retrieving Data : The ResultSet interface is used to iterate through the rows of the result set and retrieve the data from the columns.

2. Scrolling Through Results : The ResultSet can be scrolled forwards and backwards to navigate through the results of the SQL query.

3. Modification Of Data : The ResultSet allows modifications to the data within the database to be made, including insertion , deletion , and updates.

Show Below Image Component Dependency :



SQL Exception : SQL Exception is an exception class that handles errors and exception related to database interactions. It provides information about the type of error that occurred, and allows for more accurate debugging and error resolution.

Program Flow :

1 : Connect your IDE With Database using necessary connector

2 : Load Necessary Drivers

3 : Create Connection

4 : Create Statement

5 : Execute Query

Requirements :

1.download MySQL Connector /J → Select Platform Independent → Zip download

2.Create the new project in IntelliJ IDE

3.Connect jar file → project structure → library → select java → select jar file & apply it.

4.create JDBC Boiler plat

How to Connect IntelliJ With MySQL:

Requirement : MySQL & IntelliJ IDEA

Work Flow :

1. Set up a MySQL connector – Create a new DB in MySQL

2. Create a new project – Open IntelliJ & Create a new project

3. Add the MySQL connector – Add the MySQL connector to your project's dependencies.

Add jar file in Intellije Project -

[Project Structure – Libraries – Add mysql.connector jar file]

4. Configure the MySQL connection – Set up the connection to your MySQL DB in IntelliJ.

Write Code for Connected the MySQL to IntelliJ

```
import java.sql.Connection;
import  java.sql.DriverManager;
import  java.sql.SQLException;

public class Main {
    public static void main(String[] args){
      String url = "jdbc:mysql://localhost:3306/Students";

      String username ="root";
      String password ="Kiran@123";

      try (Connection connection = DriverManager.getConnection(url, username,
password)){
          System.out.println("Connected to the database.");
          System.out.println(connection);
      }
      catch (SQLException e){
          System.err.println("Connection failed : "+e.getMessage());
      }
    }
}
```

5. Execute SQL Queries – Run SQL queries in IntelliJ to interact with your DB.

<u>Exception Handling java</u> :

Types

Checked : Compile time check

Unchecked : Run time check

Check Exception :-

IOException : Input & Out Time Occurs

SQLException : DB Connection Time

ClassNot FoundException : Missing Class

Uncheck Exception :-

Runtime Exception :

1)ArithmeticException : 10/0

2)NullPointerException

3)NumberFormatException

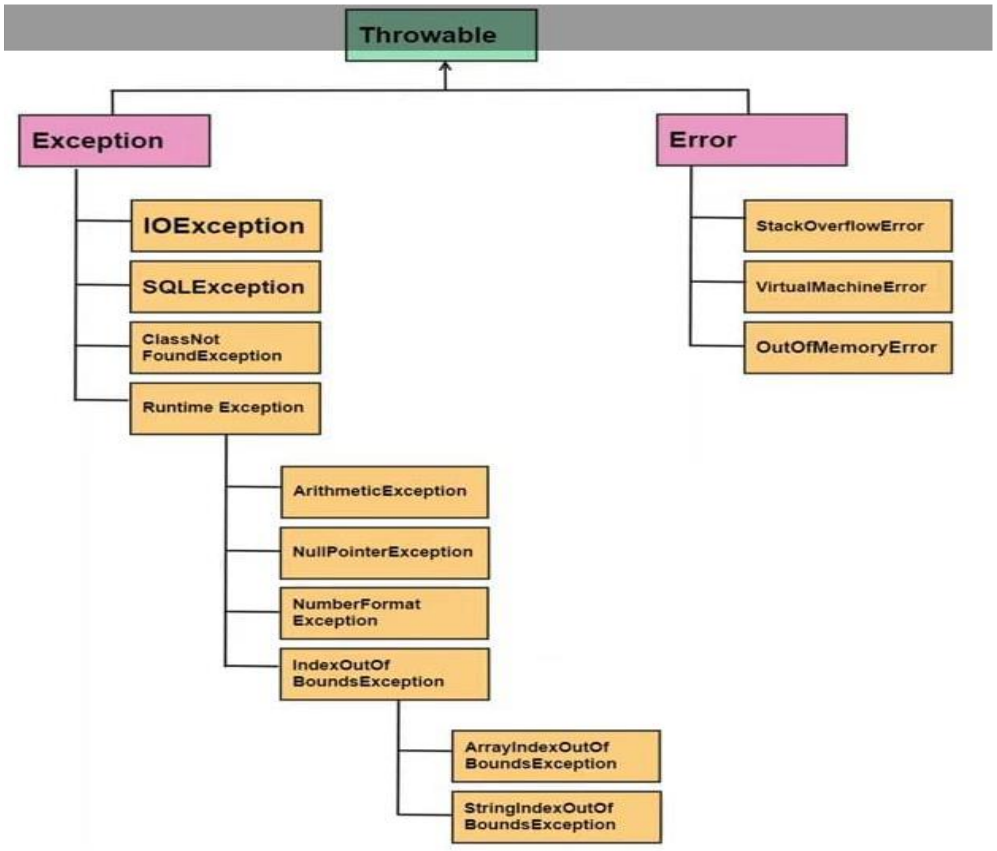
4)IndexOutOf Bounds Exception :

ArrayIndexOutOf BoundsException

StringIndexOutOf BoundsException

5 Keywords :

Try , Catch , Finally , Throw & Throws –



Retrieve Data in Database :

Use This Statement For Only Retrive The Data in database :

ResultSet rs = stmt.executeQuery(query);

Code :

```
import java.sql.*;

public class First1 {
   public static void main(String [] S) throws ClassNotFoundException{

      String url ="jdbc:mysql://localhost:3306/mydatabase";
      String username ="root";
      String password ="Kiran@123";
      String query ="Select * from employees;";

      try {
         Class.forName("com.mysql.jdbc.Driver");
         System.out.println("Drivers loaded Successfully");
      }catch (ClassNotFoundException e){
         System.out.println(e.getMessage());
      }

      try{
         Connection con =
DriverManager.getConnection(url,username,password);
         System.out.println("Coonection Established Successfully !!");
         Statement stmt = con.createStatement();
         ResultSet rs = stmt.executeQuery(query); //this is use for only retrived the
data in databse.
         while(rs.next()){
             int id = rs.getInt("id");
             String name= rs.getString("name");
             String job_title= rs.getString("job_title");
             double salary=rs.getDouble("salary");

             System.out.println("=========================");
             System.out.println("ID : " + id);
             System.out.println("Name : " + name);
             System.out.println("Job_Title: " + job_title);
             System.out.println("Salary : " + salary);
         }
         rs.close();
         stmt.close();
         con.close();
         System.out.println("Connection load Successfully!!!");
      }catch (SQLException e){
         System.out.println(e.getMessage());
      }
   }
}
```

Insert Data in Database :

Use this statement for Insert / Update & Delete Purpose :

```
int rowsaffected  = stmt.executeUpdate(query);
```

Code :

```
import com.mysql.cj.protocol.Resultset;

import java.sql.*;

public class Second {
   public static void main(String [] S) throws ClassNotFoundException{

       String url ="jdbc:mysql://localhost:3306/mydatabase";
       String username ="root";
       String password ="Kiran@123";
       String query ="insert into employees(id,name,job_title,salary)
values(4,'Ganesh','fullstack developer',85000.0);";

       try {
          Class.forName("com.mysql.jdbc.Driver");
          System.out.println("Drivers loaded Successfully");
       }catch (ClassNotFoundException e){
          System.out.println(e.getMessage());
       }

       try{
          Connection con = DriverManager.getConnection(url,username,password);
          System.out.println("Coonection Established Successfully !!");
          Statement stmt = con.createStatement();

          int rowsaffected  = stmt.executeUpdate(query); // this statement use only
for insert / delete / update the data in database

          if(rowsaffected >0){
             System.out.println("Insert Successfull " + rowsaffected + "row(s)
affected");
          }else{
             System.out.println("Insertion failed!!");
          }

          stmt.close();
          con.close();
```

```
            System.out.println("Connection load Successfully!!!");
        }catch (SQLException e){
            System.out.println(e.getMessage());
        }
    }
}
```

Deletion Data From Database :

Code :

```
import java.sql.*;

public class Deletion{
    public static void main(String [] S) throws ClassNotFoundException{

        String url ="jdbc:mysql://localhost:3306/mydatabase";
        String username ="root";
        String password ="Kiran@123";
        String query ="delete from employees where id='3';";

        try {
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("Drivers loaded Successfully");
        }catch (ClassNotFoundException e){
            System.out.println(e.getMessage());
        }

        try{
            Connection con = DriverManager.getConnection(url,username,password);
            System.out.println("Coonection Established Successfully !!");
            Statement stmt = con.createStatement();

            int rowsaffected  = stmt.executeUpdate(query);

            if(rowsaffected >0){
                System.out.println("Deletion Successfull " + rowsaffected + " row(s) 
affected");
```

```
            }else{
                System.out.println("Deletion failed!!");
            }

            stmt.close();
            con.close();
            System.out.println("Connection load Successfully!!!");
        }catch (SQLException e){
            System.out.println(e.getMessage());
        }
    }
}
```

<u>Project : Hotel Reservation System</u>

Create 6 Function :

New Reservations

Check Reservations

Get Room No

Update Reservations

Delete reservations

Exit

Database :

Database name – hotel_db

Table name – reservations

Schema :

Reservation_id – int Auto incr (Primary key)

guest_name – varchar not null

room_number – int not null

contact_number – varchar not null

reservation_date – timestamp default

Create Table :

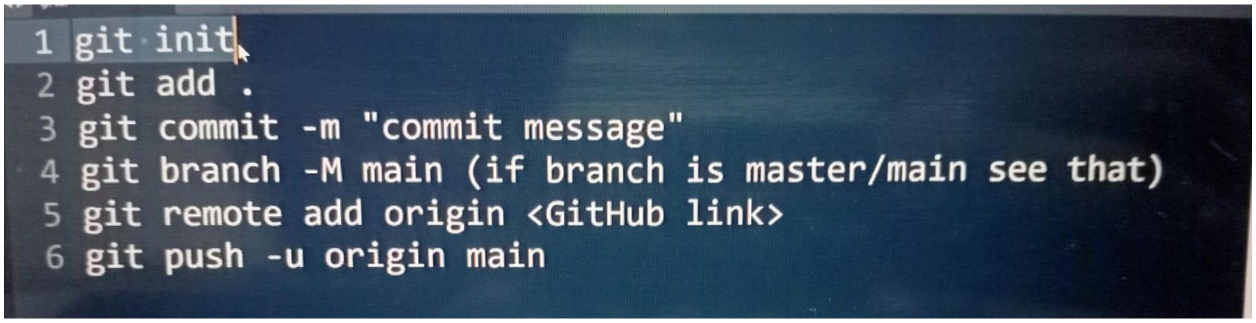
```
mysql> show databases;
mysql> create database hotel_db;
mysql> use hotel_db
mysql> create table reservations(
    -> reservation_id int auto_increment primary key,
    -> guest_name varchar(255) not null,
    -> room_number int not null,contact_number varchar(10) not null,
    -> reservation_date timestamp default current_timestap
    -> );
```

Code :

```
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Connection;
import java.util.Scanner;
import java.sql.Statement;
import java.sql.ResultSet;


public class HotelReservationSystem {
    private static final String url = "jdbc:mysql://localhost:3306/hotel_db";
    private static final String username = "root";
    private static final String password = "Kiran@123";

    public static void main(String[] args) throws ClassNotFoundException,
SQLException {
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
        }catch (ClassNotFoundException e){
            System.out.println(e.getMessage());
        }

        try{
            Connection = DriverManager.getConnection(url, username, password);
            while(true){
                System.out.println();
                System.out.println("HOTEL MANAGEMENT SYSTEM");
                Scanner = new Scanner(System.in);
                System.out.println("1. Reserve a room");
                System.out.println("2. View Reservations");
                System.out.println("3. Get Room Number");
                System.out.println("4. Update Reservations");
                System.out.println("5. Delete Reservations");
                System.out.println("0. Exit");
                System.out.print("Choose an option: ");
                int choice = scanner.nextInt();
                switch (choice) {
                    case 1:
                        reserveRoom(connection, scanner);
                        break;
                    case 2:
                        viewReservations(connection);
                        break;
                    case 3:
                        getRoomNumber(connection, scanner);
                        break;
                    case 4:
                        updateReservation(connection, scanner);
                        break;
                    case 5:
                        deleteReservation(connection, scanner);
                        break;
                    case 0:
                        exit();
                        scanner.close();
                        return;
                    default:
                        System.out.println("Invalid choice. Try again.");
                }
            }

        }catch (SQLException e){
            System.out.println(e.getMessage());
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }


    }

    private static void reserveRoom(Connection connection, Scanner scanner) {
        try {
            System.out.print("Enter guest name: ");
            String guestName = scanner.next();
            scanner.nextLine();
            System.out.print("Enter room number: ");
            int roomNumber = scanner.nextInt();
            System.out.print("Enter contact number: ");
            String contactNumber = scanner.next();

            String sql = "INSERT INTO reservations (guest_name, room_number,
contact_number) " +
                    "VALUES ('" + guestName + "', " + roomNumber + ", '" +
contactNumber + "')";

            try (Statement = connection.createStatement()) {
                int affectedRows = statement.executeUpdate(sql);

                if (affectedRows > 0) {
                    System.out.println("Reservation successful!");
                } else {
                    System.out.println("Reservation failed.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void viewReservations(Connection connection) throws
SQLException {
        String sql = "SELECT reservation_id, guest_name, room_number,
contact_number, reservation_date FROM reservations";

        try (Statement = connection.createStatement();
             ResultSet = statement.executeQuery(sql)) {

            System.out.println("Current Reservations:");
            System.out.println("+----------------+-----------------+---------------+------------
----------+-------------------------+");
            System.out.println("| Reservation ID | Guest           | Room Number   |
Contact Number       | Reservation Date        |");
            System.out.println("+----------------+-----------------+---------------+------------
----------+-------------------------+");

            while (resultSet.next()) {
                int reservationId = resultSet.getInt("reservation_id");
                String guestName = resultSet.getString("guest_name");
                int roomNumber = resultSet.getInt("room_number");
                String contactNumber = resultSet.getString("contact_number");
                String reservationDate =
resultSet.getTimestamp("reservation_date").toString();

                // Format and display the reservation data in a table-like format
                System.out.printf("| %-14d | %-15s | %-13d | %-20s | %-19s   |\n",
                        reservationId, guestName, roomNumber, contactNumber,
reservationDate);
            }

            System.out.println("+----------------+-----------------+---------------+------------
----------+-------------------------+");
        }
    }


    private static void getRoomNumber(Connection connection, Scanner scanner) {
        try {
            System.out.print("Enter reservation ID: ");
            int reservationId = scanner.nextInt();
            System.out.print("Enter guest name: ");
            String guestName = scanner.next();

            String sql = "SELECT room_number FROM reservations " +
                    "WHERE reservation_id = " + reservationId +
                    " AND guest_name = '" + guestName + "'";

            try (Statement = connection.createStatement();
                 ResultSet = statement.executeQuery(sql)) {

                if (resultSet.next()) {
                    int roomNumber = resultSet.getInt("room_number");
                    System.out.println("Room number for Reservation ID " +
reservationId +
                            " and Guest " + guestName + " is: " + roomNumber);
                } else {
                    System.out.println("Reservation not found for the given ID and guest
name.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    private static void updateReservation(Connection connection, Scanner scanner)
{
        try {
            System.out.print("Enter reservation ID to update: ");
            int reservationId = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character

            if (!reservationExists(connection, reservationId)) {
                System.out.println("Reservation not found for the given ID.");
                return;
            }

            System.out.print("Enter new guest name: ");
            String newGuestName = scanner.nextLine();
            System.out.print("Enter new room number: ");
            int newRoomNumber = scanner.nextInt();
            System.out.print("Enter new contact number: ");
            String newContactNumber = scanner.next();

            String sql = "UPDATE reservations SET guest_name = '" +
newGuestName + "', " +
                    "room_number = " + newRoomNumber + ", " +
                    "contact_number = '" + newContactNumber + "' " +
                    "WHERE reservation_id = " + reservationId;

            try (Statement = connection.createStatement()) {
                int affectedRows = statement.executeUpdate(sql);

                if (affectedRows > 0) {
                    System.out.println("Reservation updated successfully!");
                } else {
                    System.out.println("Reservation update failed.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static void deleteReservation(Connection connection, Scanner scanner) {
        try {
            System.out.print("Enter reservation ID to delete: ");
            int reservationId = scanner.nextInt();

            if (!reservationExists(connection, reservationId)) {
                System.out.println("Reservation not found for the given ID.");
                return;
            }

            String sql = "DELETE FROM reservations WHERE reservation_id = " +
reservationId;

            try (Statement = connection.createStatement()) {
                int affectedRows = statement.executeUpdate(sql);

                if (affectedRows > 0) {
                    System.out.println("Reservation deleted successfully!");
                } else {
                    System.out.println("Reservation deletion failed.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    private static boolean reservationExists(Connection connection, int
reservationId) {
        try {
            String sql = "SELECT reservation_id FROM reservations WHERE
reservation_id = " + reservationId;

            try (Statement = connection.createStatement();
                 ResultSet = statement.executeQuery(sql)) {
```

```
                return resultSet.next(); // If there's a result, the reservation exists
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return false; // Handle database errors as needed
        }
    }


    public static void exit() throws InterruptedException {
        System.out.print("Exiting System");
        int i = 5;
        while(i!=0){
            System.out.print(".");
            Thread.sleep(1000);
            i--;
        }
        System.out.println();
        System.out.println("ThankYou For Using Hotel Reservation System!!!");
    }
}
```

Project Upload in Gitub For that purpose use the Below Command :



```
git init
git add .
git commit -m "commit message"
git branch -M main (if branch is master/main see that)
git remote add origin <GitHub link>
git push -u origin main
```

Prepared Statements :

Prepared Statements are a feature in database programming, commonly used in JDBC and access libraries.

They are used to execute SQL Queries with placeholders for parameters.

These placeholders are then filed with specific values when the query is executed.

1. Protection against SQL Injection.
2. Improved Performance.
3. Code Readability and Maintainability.
4. Automatic Data Type Handling.
5. Portability etc….

Retrieved Data from Database using Prepared Statements single & multiple placeholder

Code :

```
 import java.sql.*;
public class Prepared_Statements{
   public static void main(String [] S) throws ClassNotFoundException {

       String url = "jdbc:mysql://localhost:3306/mydatabase";
       String username = "root";
       String password = "Kiran@123";
       String query="Select * from employees where name = ? AND job_title= ?";
       try {
           Class.forName("com.mysql.cj.jdbc.Driver");
           System.out.println("Drivers loaded Successfully");
       } catch (ClassNotFoundException e) {
           System.out.println(e.getMessage());
       }
       try{
           Connection con = DriverManager.getConnection(url,username,password);
           System.out.println("connection Established Successfully !!");
           // Statement statement = con.createStatement();
           PreparedStatement preparedStatements = con.prepareStatement(query);
           preparedStatements.setString(1,"Shra");
```

```
            preparedStatements.setString(2,"Frontend Developer");
            ResultSet = preparedStatements.executeQuery();
            while(resultSet.next()){
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
                String job_title = resultSet.getString("job_title");
                double salary = resultSet.getDouble("salary");
                System.out.println("ID : "+id);
                System.out.println("Name : "+name);
                System.out.println("Job_Title : "+job_title);
                System.out.println("Salary : "+salary);
            }
            resultSet.close();
            preparedStatements.close();
            con.close();
            System.out.println();
            System.out.println("Connection Closed Successfully!!!!");
        }catch (SQLException e){
            System.out.println(e.getMessage());
        }



    }
}
```

Using this Prepared Statement Insert Data in Database :

Code :

```
import java.sql.*;
public class Prepared_Statements{
    public static void main(String [] S) throws ClassNotFoundException {

        String url = "jdbc:mysql://localhost:3306/mydatabase";
        String username = "root";
        String password = "Kiran@123";
        String query="Insert into employees(id,name,job_title,salary) values(?,?,?,?)";
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("Drivers loaded Successfully");
        } catch (ClassNotFoundException e) {
            System.out.println(e.getMessage());
        }
        try{
            Connection con = DriverManager.getConnection(url,username,password);
            System.out.println("connection Established Successfully !!");
            // Statement = con.createStatement();
            PreparedStatement preparedStatements = con.prepareStatement(query);
            preparedStatements.setInt(1,3);
            preparedStatements.setString(2,"Karn");
            preparedStatements.setString(3,"DevOps Engineer");
            preparedStatements.setDouble(4,95000.0);

           int rowsAffected = preparedStatements.executeUpdate();
           if(rowsAffected>0){
               System.out.println("Data Inserted Successfully!!!");
           }else {
               System.out.println("Data Insertion Failed!!!");
           }

            preparedStatements.close();
            con.close();
            System.out.println();
            System.out.println("Connection Closed Successfully!!!!");
        }catch (SQLException e){
            System.out.println(e.getMessage());
        }



    }
}
```

User Input Data :

Code :

```
import java.sql.*;
import java.util.Scanner;

public class Prepared_Statements{
    public static void main(String [] S) throws ClassNotFoundException {

        String url = "jdbc:mysql://localhost:3306/mydatabase";
        String username = "root";
        String password = "Kiran@123";
        String query="Insert into employees(id,name,job_title,salary) values(?,?,?,?)";
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("Drivers loaded Successfully");
        } catch (ClassNotFoundException e) {
            System.out.println(e.getMessage());
        }
        try{
            Connection con = DriverManager.getConnection(url,username,password);
            System.out.println("connection Established Successfully !!");
            // Statement = con.createStatement();
            Scanner sc = new Scanner(System.in);
            int id =sc.nextInt();
            sc.nextLine();
            String name=sc.nextLine();
            String job_title=sc.nextLine();
            Double salary=sc.nextDouble();
            PreparedStatement preparedStatements = con.prepareStatement(query);
            preparedStatements.setInt(1,id);
            preparedStatements.setString(2,name);
            preparedStatements.setString(3,job_title);
            preparedStatements.setDouble(4,salary);

           int rowsAffected = preparedStatements.executeUpdate();
           if(rowsAffected>0){
                System.out.println("Data Inserted Successfully!!!");
            }else {
                System.out.println("Data Insertion Failed!!!");
            }

            preparedStatements.close();
            con.close();
            System.out.println();
            System.out.println("Connection Closed Successfully!!!!");
        }catch (SQLException e){
            System.out.println(e.getMessage());
        }



    }
}
```

Batch Processing :

Multiple data insert in table -

This Code write using PreparedStatement -

Code :

```
import java.sql.*;
import java.util.Scanner;

public class Main{
    private static final String url = "jdbc:mysql://localhost:3306/kiran";
    private static final String username = "root";
    private static final String password = "Kiran@123";

    public static void main(String[] args) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        }catch (ClassNotFoundException e){
            System.out.println(e.getMessage());
        }
        try{
            Connection con = DriverManager.getConnection(url,username,password);
            String query = "Insert into stud(name,marks)values(?,?)";
            PreparedStatement ps = con.prepareStatement(query);
            Scanner sc = new Scanner(System.in);
            while(true){
                System.out.println("Enter name : ");
                String name = sc.next();
                System.out.println("Enter Marks : ");
                int marks = sc.nextInt();
                System.out.print("Enter more data(Y/N) : ");
                String choice = sc.next();
                ps.setString(1,name);
                ps.setInt(2,marks);

                ps.addBatch();
                if(choice.toUpperCase().equals("N")){
                    break;
                }
            }
            int[] arr = ps.executeBatch();
            for(int i = 0; i<arr.length;i++){
                if(arr[i] == 0){
                System.out.println("Query: "+i+" not executed Successfully!!");
            }}

        }catch (SQLException e){
            System.out.println(e.getMessage());
        }
    }
}
```

Transaction :

Two part Commit & Rollback.