OOPS MINI PROJECT

Bike Rental Management System

Aim:

To develop a Bike Rental Management System that connects to a MySQL database and allows users to:

1. Add Bikes
2. Rent a Bike
3. Return a Bike
4. View Available Bikes

Algorithm:

1. Start the program.
2. Connect to the MySQL database.
3. Display a menu with the following options:
    - Add Bike
    - Rent Bike
    - Return Bike
    - View Available Bikes
    - Exit
4. Based on the user's choice:
    - Add Bike: Take inputs (Bike Name, Bike Model, Rental Price), and add them to the bikes table.
    - Rent Bike: Update the bike's availability and record the rental details in the rentals table.

    - Return Bike: Mark the bike as available and update the return date in the rentals table.
    - View Available Bikes: Display all bikes currently available for rent.
    - Exit: Close the database connection and terminate the program.

5. Repeat until the user chooses to exit.

SQL Queries:

```
CREATE DATABASE BikeRentalDB;
USE BikeRentalDB;

CREATE TABLE bikes (
bike_id INT AUTO_INCREMENT PRIMARY KEY,
bike_name VARCHAR(100),
bike_model VARCHAR(50),
rental_price DECIMAL(10, 2),
is_available BOOLEAN DEFAULT TRUE
);

CREATE TABLE rentals (
rental_id INT AUTO_INCREMENT PRIMARY KEY,
bike_id INT,
customer_name VARCHAR(100),
rental_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
return_date TIMESTAMP NULL,
FOREIGN KEY (bike_id) REFERENCES bikes(bike_id)
);
```

JAVA PROGRAM:

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

public class BikeRentalManagement {
private static final String DB_URL =
"jdbc:mysql://localhost:3306/BikeRentalDB";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "";

public static void main(String[] args) {
try (Connection conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
Scanner scanner = new Scanner(System.in)) {
System.out.println("Connected to the database!");

while (true) {
System.out.println("\n--- Bike Rental Management System ---");
System.out.println("1. Add Bike");
System.out.println("2. View Available Bikes");
System.out.println("3. Rent a Bike");
System.out.println("4. Return a Bike");
System.out.println("5. Exit");
System.out.print("Enter your choice: ");
int choice = scanner.nextInt();
scanner.nextLine(); // Consume newline
switch (choice) {
case 1 -> addBike(conn, scanner);
case 2 -> viewAvailableBikes(conn);
case 3 -> rentBike(conn, scanner);
case 4 -> returnBike(conn, scanner);
case 5 -> {
System.out.println("Exiting... Goodbye!");
return;
}
default -> System.out.println("Invalid choice. Please try again.");
}
}
} catch (SQLException e) {
e.printStackTrace();
}
}

private static void addBike(Connection conn, Scanner scanner) throws SQLException {
System.out.print("Enter Bike Name: ");
String name = scanner.nextLine();
System.out.print("Enter Bike Model: ");
String model = scanner.nextLine();
System.out.print("Enter Rental Price: ");
double price = scanner.nextDouble();
```

```
String query = "INSERT INTO bikes (bike_name, bike_model, rental_price)
VALUES (?, ?, ?)";
try (PreparedStatement stmt = conn.prepareStatement(query)) {
stmt.setString(1, name);
stmt.setString(2, model);
stmt.setDouble(3, price);
stmt.executeUpdate();
System.out.println("Bike added successfully!");
}
}

private static void viewAvailableBikes(Connection conn) throws SQLException {
String query = "SELECT * FROM bikes WHERE is_available = TRUE";
try (Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(query)) {
System.out.println("\n--- Available Bikes ---");
while (rs.next()) {
System.out.println("Bike ID: " + rs.getInt("bike_id") +
", Name: " + rs.getString("bike_name") +
", Model: " + rs.getString("bike_model") +
", Price: " + rs.getDouble("rental_price"));
}
}
}

private static void rentBike(Connection conn, Scanner scanner) throws
SQLException {
```

```
System.out.print("Enter your name: ");
String customerName = scanner.nextLine();
System.out.print("Enter Bike ID to rent: ");
int bikeId = scanner.nextInt();

String checkQuery = "SELECT is_available FROM bikes WHERE bike_id = ?";
try (PreparedStatement checkStmt = conn.prepareStatement(checkQuery)) {
checkStmt.setInt(1, bikeId);
try (ResultSet rs = checkStmt.executeQuery()) {
if (rs.next() && rs.getBoolean("is_available")) {
String rentQuery = "INSERT INTO rentals (bike_id, customer_name) VALUES (?,
?)";
try (PreparedStatement rentStmt = conn.prepareStatement(rentQuery)) {
rentStmt.setInt(1, bikeId);
rentStmt.setString(2, customerName);
rentStmt.executeUpdate();

String updateQuery = "UPDATE bikes SET is_available = FALSE WHERE bike_id =
?";
try (PreparedStatement updateStmt = conn.prepareStatement(updateQuery)) {
updateStmt.setInt(1, bikeId);
updateStmt.executeUpdate();
System.out.println("Bike rented successfully!");
}
}
} else {
System.out.println("Bike is not available.");
```

```
}

}

}

}


private static void returnBike(Connection conn, Scanner scanner) throws
SQLException {

System.out.print("Enter Rental ID to return: ");

int rentalId = scanner.nextInt();


String checkQuery = "SELECT bike_id FROM rentals WHERE rental_id = ? AND
return_date IS NULL";

try (PreparedStatement checkStmt = conn.prepareStatement(checkQuery)) {

checkStmt.setInt(1, rentalId);

try (ResultSet rs = checkStmt.executeQuery()) {

if (rs.next()) {

int bikeId = rs.getInt("bike_id");


String updateBikeQuery = "UPDATE bikes SET is_available = TRUE WHERE
bike_id = ?";

try (PreparedStatement updateBikeStmt =
conn.prepareStatement(updateBikeQuery)) {

updateBikeStmt.setInt(1, bikeId);

updateBikeStmt.executeUpdate();

}


String updateRentalQuery = "UPDATE rentals SET return_date =
CURRENT_TIMESTAMP WHERE rental_id = ?";
```

```
try (PreparedStatement updateRentalStmt =
conn.prepareStatement(updateRentalQuery)) {

updateRentalStmt.setInt(1, rentalId);

updateRentalStmt.executeUpdate();

System.out.println("Bike returned successfully!");

}

} else {

System.out.println("Invalid Rental ID or bike already returned.");

}

}

}

}

}
```

INPUT:

Choose an Option:

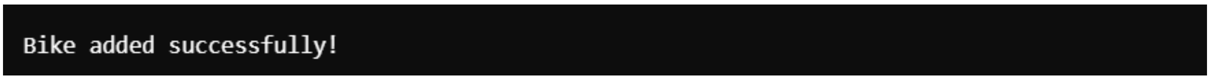
1. Add Bike
2. View Available Bikes
3. Rent a Bike
4. Return a Bike

```
1
Enter Bike Name: Mountain Bike
Enter Bike Model: MTB-X
Enter Rental Price: 500
2
Bike ID: 1, Name: Mountain Bike, Model: MTB-X, Price: 500.00
Bike ID: 2, Name: Road Bike, Model: RB-200, Price: 700.00
Bike ID: 3, Name: Hybrid Bike, Model: HB-100, Price: 600.00
Bike ID: 4, Name: Electric Bike, Model: EB-500, Price: 1200.00
Bike ID: 5, Name: City Bike, Model: CB-100, Price: 400.00
```

```
3
Enter your name: John Doe
Enter Bike ID to rent: 1
4
Enter Rental ID to return: 1
```

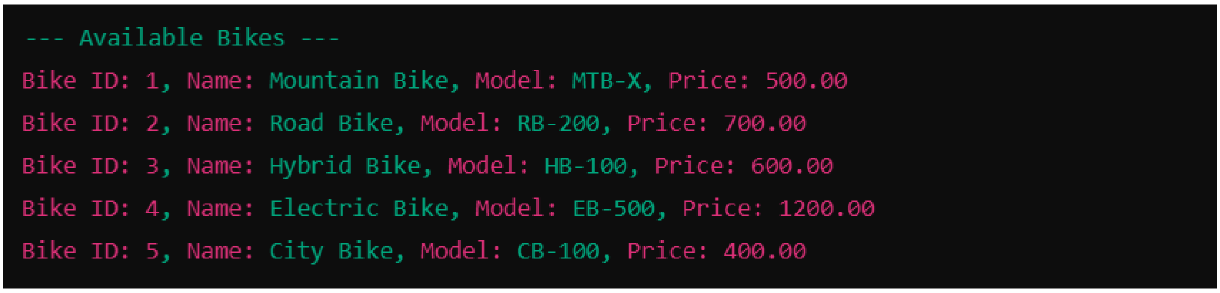
OUTPUT:

1.Add Bike



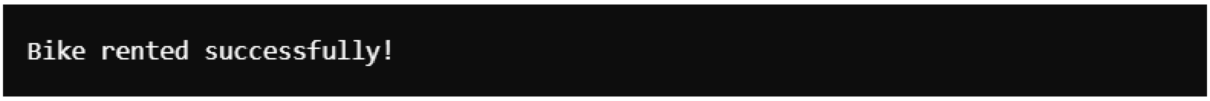
```
Bike added successfully!
```

2.View Available Bikes



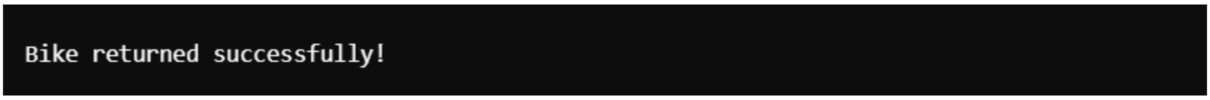
```
--- Available Bikes ---
Bike ID: 1, Name: Mountain Bike, Model: MTB-X, Price: 500.00
Bike ID: 2, Name: Road Bike, Model: RB-200, Price: 700.00
Bike ID: 3, Name: Hybrid Bike, Model: HB-100, Price: 600.00
Bike ID: 4, Name: Electric Bike, Model: EB-500, Price: 1200.00
Bike ID: 5, Name: City Bike, Model: CB-100, Price: 400.00
```

3. Rent a Bike



```
Bike rented successfully!
```

4.Return a Bike



```
Bike returned successfully!
```

RESULT:

The database construction for the bike rental management system has been successfully completed and connected with my sql using java.