King Saud University
College of Computer and Information Sciences
Department of Information Systems

IS230: Introduction to Database Systems
2nd Semester 1445 H

VELOCITY MOTORS

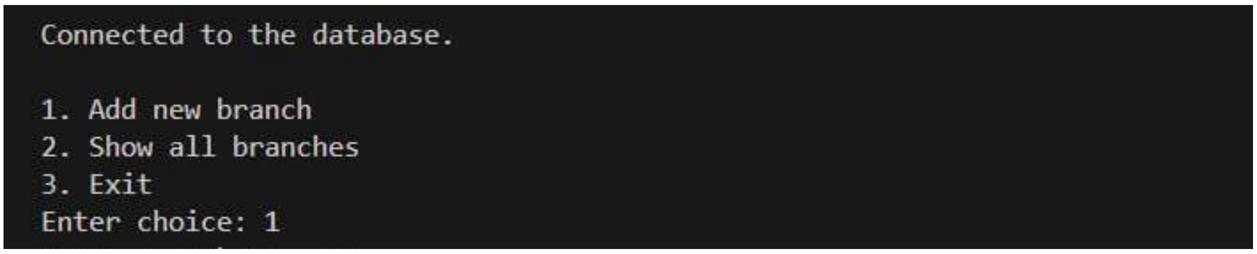
Car Dealership Management Database System | Phase 3

Section #	SN	NAME	ID
Group Number: 2			
67167	25	**Manar Khalaf Alenazi**	443200987
67167	20	**Norah Abdullah Almubarak**	443200845
67167	33	**Fatemah Tawfiq Alelawi**	443204251
67167	14	**Futun Shaya Alhabshan**	443200740
67167	2	**Raghad Alotibi**	442200834
67167	26	**Sarah Alsahli**	443201001

Supervised By: I. Nurah AlQahtani and I. Ghada AlRabeah

Part1: Screenshot of the execution showing how clear and specific messages will be displayed for each.

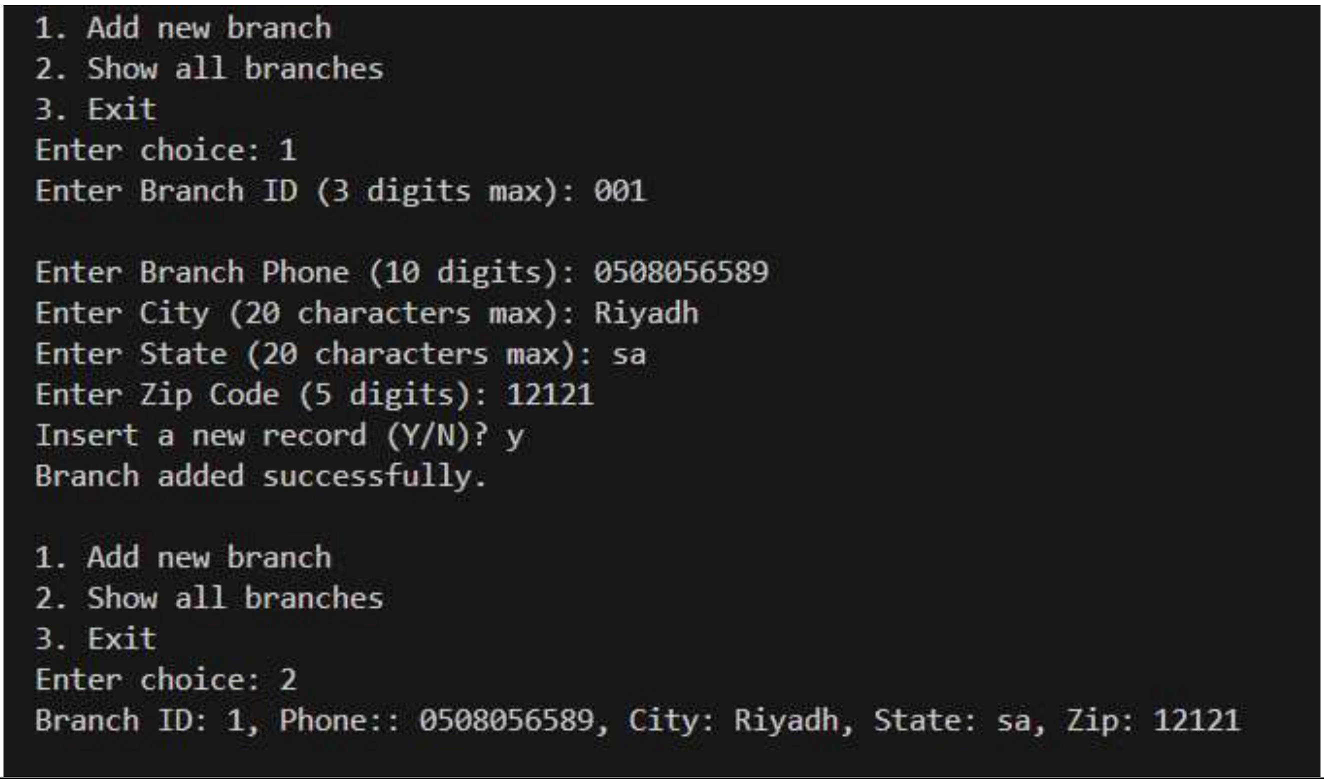
1) **First Screen:**



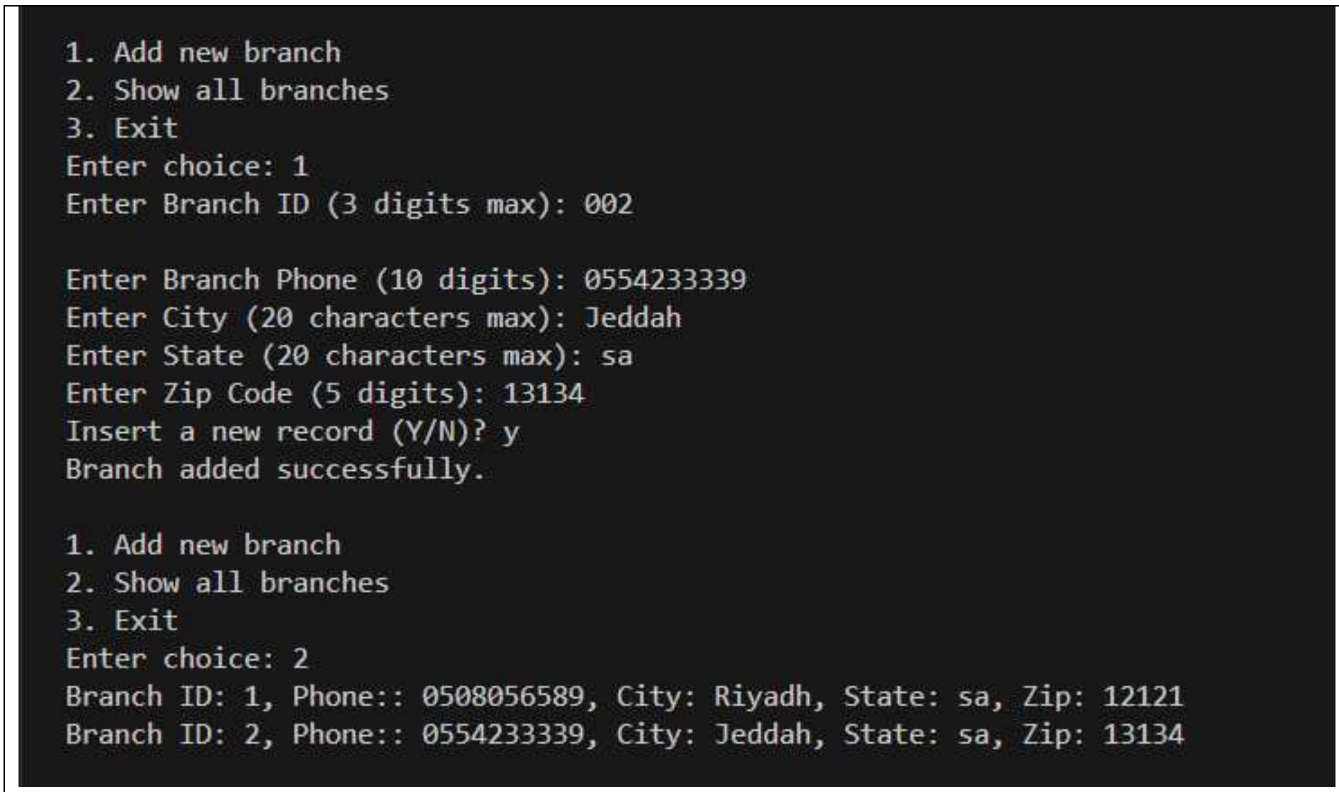
```
Connected to the database.

1. Add new branch
2. Show all branches
3. Exit
Enter choice: 1
```

2) **INSERT Operation (EXECUTION of multiple insertion + dealing with Exception):**

Successful Insertion of two branches:

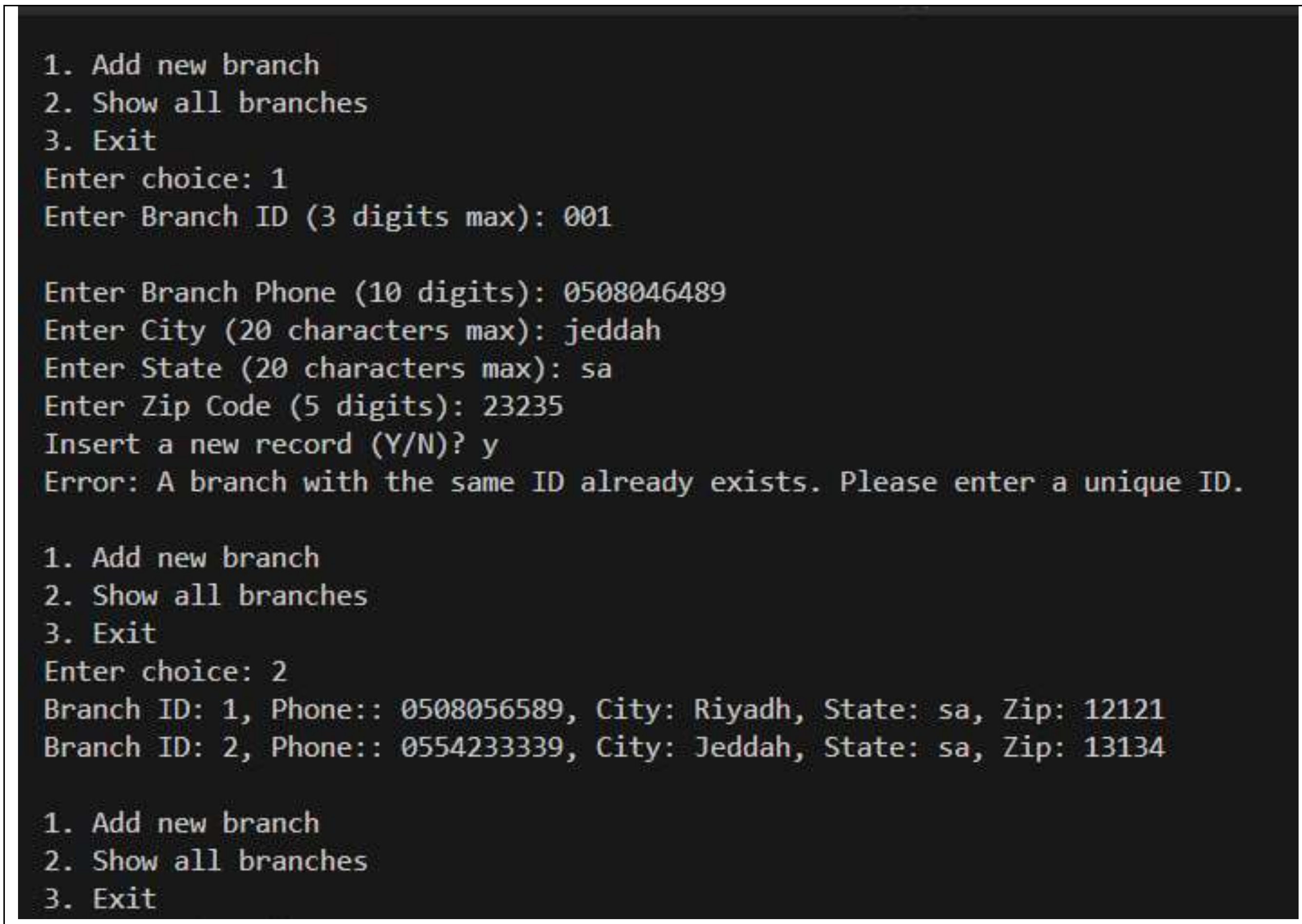


```
1. Add new branch
2. Show all branches
3. Exit
Enter choice: 1
Enter Branch ID (3 digits max): 001

Enter Branch Phone (10 digits): 0508056589
Enter City (20 characters max): Riyadh
Enter State (20 characters max): sa
Enter Zip Code (5 digits): 12121
Insert a new record (Y/N)? y
Branch added successfully.

1. Add new branch
2. Show all branches
3. Exit
Enter choice: 2
Branch ID: 1, Phone:: 0508056589, City: Riyadh, State: sa, Zip: 12121
```



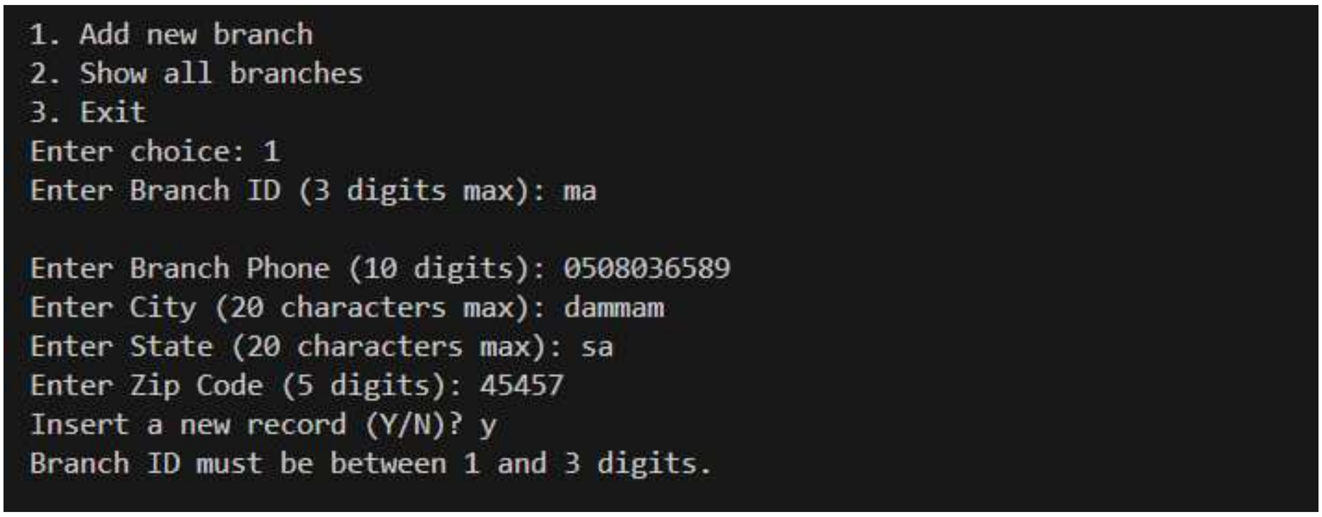
```
1. Add new branch
2. Show all branches
3. Exit
Enter choice: 1
Enter Branch ID (3 digits max): 002

Enter Branch Phone (10 digits): 0554233339
Enter City (20 characters max): Jeddah
Enter State (20 characters max): sa
Enter Zip Code (5 digits): 13134
Insert a new record (Y/N)? y
Branch added successfully.

1. Add new branch
2. Show all branches
3. Exit
Enter choice: 2
Branch ID: 1, Phone:: 0508056589, City: Riyadh, State: sa, Zip: 12121
Branch ID: 2, Phone:: 0554233339, City: Jeddah, State: sa, Zip: 13134
```

Unsuccessful operation, duplicate primary key:



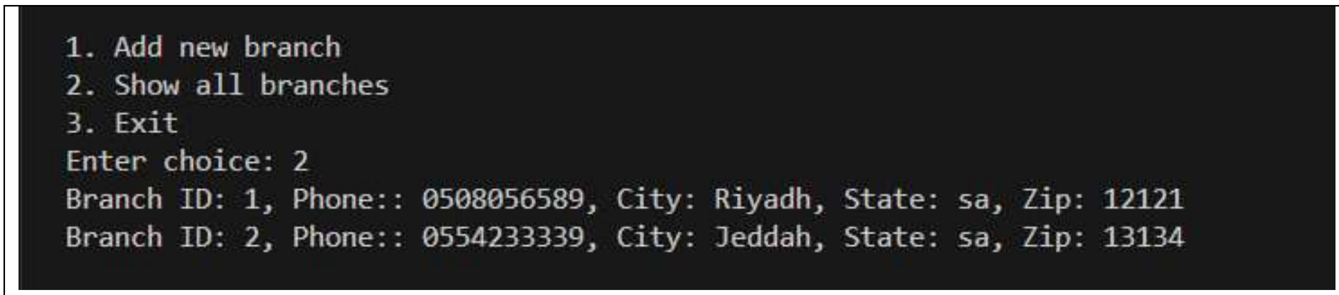
```
1. Add new branch
2. Show all branches
3. Exit
Enter choice: 1
Enter Branch ID (3 digits max): 001

Enter Branch Phone (10 digits): 0508046489
Enter City (20 characters max): jeddah
Enter State (20 characters max): sa
Enter Zip Code (5 digits): 23235
Insert a new record (Y/N)? y
Error: A branch with the same ID already exists. Please enter a unique ID.

1. Add new branch
2. Show all branches
3. Exit
Enter choice: 2
Branch ID: 1, Phone:: 0508056589, City: Riyadh, State: sa, Zip: 12121
Branch ID: 2, Phone:: 0554233339, City: Jeddah, State: sa, Zip: 13134

1. Add new branch
2. Show all branches
3. Exit
```

Unsuccessful operation, Domain constraint violation:



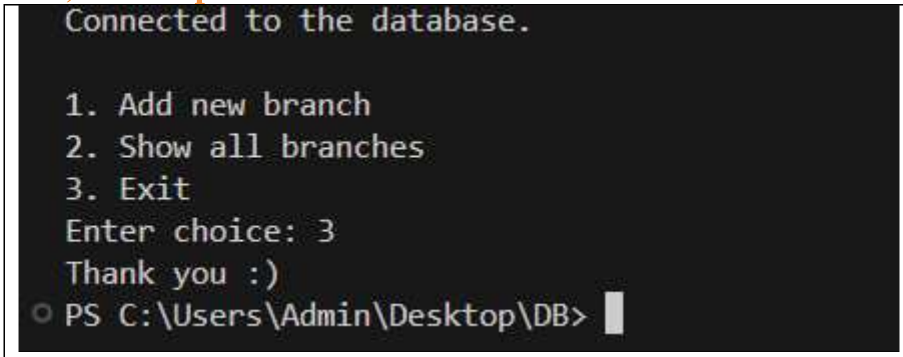
```
1. Add new branch
2. Show all branches
3. Exit
Enter choice: 1
Enter Branch ID (3 digits max): ma

Enter Branch Phone (10 digits): 0508036589
Enter City (20 characters max): dammam
Enter State (20 characters max): sa
Enter Zip Code (5 digits): 45457
Insert a new record (Y/N)? y
Branch ID must be between 1 and 3 digits.
```

3) Display Operation:



```
1. Add new branch
2. Show all branches
3. Exit
Enter choice: 2
Branch ID: 1, Phone:: 0508056589, City: Riyadh, State: sa, Zip: 12121
Branch ID: 2, Phone:: 0554233339, City: Jeddah, State: sa, Zip: 13134
```

4) **Exit operation:**



```
Connected to the database.

1. Add new branch
2. Show all branches
3. Exit
Enter choice: 3
Thank you :)
PS C:\Users\Admin\Desktop\DB>
```

Part 2: Source code

1) INSERTION Code

This is from the manageDatabaseOperations(Connection conn) method called by main responsible for calling addNewBranch(conn, scanner):

```
case 1:
                     addNewBranch(conn, scanner);
                        break;
```

This is the body of addNewBranch(conn, scanner) method

```
private static void addNewBranch(Connection conn, Scanner scanner) {

        System.out.print("Enter Branch ID (3 digits max): ");
        String id= scanner.nextLine();
        scanner.nextLine(); // consume newline

        System.out.print("Enter Branch Phone (10 digits): ");
        String phone = scanner.nextLine();
        System.out.print("Enter City (20 characters max): ");
        String city = scanner.nextLine();
        System.out.print("Enter State (20 characters max): ");
        String state = scanner.nextLine();
        System.out.print("Enter Zip Code (5 digits): ");
        String zip = scanner.nextLine();

        System.out.print("Insert a new record (Y/N)? ");
        String userChoice = scanner.nextLine();

        if (userChoice.equalsIgnoreCase("n")) {
            System.out.println("Operation cancelled by user.");
            return;
        }

        try {

        if (!id.matches("\\d{1,3}")) {
            System.err.println("Branch ID must be between 1 and 3 digits.");
            return;
        }
        int Bid = Integer.parseInt(id);
        String query = "SELECT * FROM branch WHERE BranchID = ?";
        try (PreparedStatement statement = conn.prepareStatement(query)) {
            statement.setInt(1, Bid);
            ResultSet resultSet = statement.executeQuery();

            if (resultSet.next()) {
                System.out.println("Error: A branch with the same ID already exists.
Please enter a unique ID.");
                return; // Return to main menu instead of exiting
            }
        } catch (SQLException e) {
            System.out.println("Error checking for duplicate: " + e.getMessage());
            return; // Return to main menu if there's a SQL error during check
        }


            if (phone.length() > 10 || city.length() > 20 || state.length() > 20 ||
zip.length() > 10) {
                throw new IllegalArgumentException("Input data exceeds size limits.
Please adhere to the maximum lengths.");
            }

            if (phone.isEmpty() || city.isEmpty() || state.isEmpty() ||
zip.isEmpty()) {
                throw new IllegalArgumentException("All fields must be filled.
Please try again.");
            }

            if (!phone.matches("\\d{10}")) {
                throw new IllegalArgumentException("Phone number must be exactly 10
digits.");
            }

            if (!zip.matches("\\d{5}")) {
                throw new IllegalArgumentException("Zip code must be exactly 5
digits.");
            }

            String sql = "INSERT INTO Branch (BranchID, BPhone, City, State, Zip)
VALUES (?, ?, ?, ?, ?)";
            try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

                pstmt.setInt(1, Bid);
                pstmt.setString(2, phone);
                pstmt.setString(3, city);
                pstmt.setString(4, state);
                pstmt.setString(5, zip);

                int affectedRows = pstmt.executeUpdate();
                if (affectedRows == 0) {
                    throw new SQLException("Inserting branch failed, no rows 
affected.");
                }

                System.out.println("Branch added successfully.");
            }
        } catch (SQLException e) {
            System.err.println("SQL Error: " + e.getMessage());
            if (e.getErrorCode() == 1062) {
                System.err.println("Duplicate entry for Branch ID.");
            }
        } catch (IllegalArgumentException e) {
            System.err.println("Validation Error: " + e.getMessage());
        }
    }
```

2) Display Code

This is from the manageDatabaseOperations(Connection conn) method called by main responsible for calling showAllBranches(conn):

```
case 2:
                        showAllBranches(conn);
                        break;
```

```
private static void showAllBranches(Connection conn) {
        String sql = "SELECT * FROM Branch";
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                int id = rs.getInt("BranchID");
                String phone = rs.getString("BPhone");
                String city = rs.getString("City");
                String state = rs.getString("State");
                String zip = rs.getString("Zip");
                System.out.printf("Branch ID: %d, Phone:: %s, City: %s, State: %s, 
Zip: %s%n", id, phone, city, state, zip);
```

```
            }
        } catch (SQLException e) {
            System.err.println("Database error while retrieving branches: " +
e.getMessage());
            e.printStackTrace();
        }
    }
}
```

3) Exit Code

This is part of the manageDatabaseOperations(Connection conn) method called in the main:

```
case 3:

                        System.out.println(" Thank you :)");
                        exit = true;
```

This is part of the main method:

```
finally {

            try {

                if (conn != null) {

                    conn.close();

                }

            } catch (SQLException e) {

                System.err.println("Error closing the connection: " + e.getMessage());

            }

}
```

4) Code dealing with Exceptions

1- Main method: (for establishing & closing the database connection)

```
try {
      conn = DriverManager.getConnection(URL, USER, PASSWORD);
      System.out.println("Connected to the database.");
      manageDatabaseOperations(conn);
    } catch (SQLException e) {
      System.err.println("Error connecting to the database: " +
e.getMessage());
      e.printStackTrace();
    } finally {
      try {
        if (conn != null) {
          conn.close();
        }
      } catch (SQLException e) {
        System.err.println("Error closing the connection: " +
e.getMessage());
      }
    }
        if (affectedRows == 0) {
          throw new SQLException("Inserting branch failed, no rows
affected.");
        }

        System.out.println("Branch added successfully.");
      }
    } catch (SQLException e) {
      System.err.println("SQL Error: " + e.getMessage());
      if (e.getErrorCode() == 1062) {
        System.err.println("Duplicate entry for Branch ID.");
      }
    } catch (IllegalArgumentException e) {
      System.err.println("Validation Error: " + e.getMessage());
    }
  }
```

2- AddNewBranch method:(handling adding branch by validating its attributes)

```
try {
    if (!id.matches("\\d{1,3}")) {
      System.err.println("Branch ID must be between 1 and 3 digits.");
      return;
    }
    int Bid = Integer.parseInt(id);
```

```
        String query = "SELECT * FROM branch WHERE BranchdID = ?";
        try (PreparedStatement statement = conn.prepareStatement(query)) {
            statement.setInt(1, Bid);
            ResultSet resultSet = statement.executeQuery();

            if (resultSet.next()) {
                System.out.println("Error: A branch with the same ID already exists.
Please enter a unique ID.");
                return; // Return to main menu instead of exiting
            }
        } catch (SQLException e) {
            System.out.println("Error checking for duplicate: " + e.getMessage());
            return; // Return to main menu if there's a SQL error during check
        }


            if (phone.length() > 10 || city.length() > 20 || state.length() > 20 ||
zip.length() > 10) {
                throw new IllegalArgumentException("Input data exceeds size limits.
Please adhere to the maximum lengths.");
            }

            if (phone.isEmpty() || city.isEmpty() || state.isEmpty() ||
zip.isEmpty()) {
                throw new IllegalArgumentException("All fields must be filled. Please
try again.");
            }

            if (!phone.matches("\\d{10}")) {
                throw new IllegalArgumentException("Phone number must be exactly 10
digits.");
            }

            if (!zip.matches("\\d{5}")) {
                throw new IllegalArgumentException("Zip code must be exactly 5
digits.");
            }
```

3- showAllBranches() method: (handling data retrieval error)

```
try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                int id = rs.getInt("BranchID");
                String phone = rs.getString("BPhone");
                String city = rs.getString("City");
                String state = rs.getString("State");
                String zip = rs.getString("Zip");
                System.out.printf("Branch ID: %d, Phone:: %s, City: %s, State: %s,
Zip: %s%n", id, phone, city, state, zip);
            }
        } catch (SQLException e) {
            System.err.println("Database error while retrieving branches: " +
e.getMessage());
            e.printStackTrace();
        }
 String sql = "INSERT INTO Branch (BranchID, BPhone, City, State, Zip) VALUES (?, ?,
?, ?, ?)";
            try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                pstmt.setInt(1, Bid);
                pstmt.setString(2, phone);
                pstmt.setString(3, city);
                pstmt.setString(4, state);
                pstmt.setString(5, zip);
                int affectedRows = pstmt.executeUpdate();
```

The Whole Source code:

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

public class VelocityMotorsDB {
    private static final String URL = "jdbc:mariadb://localhost:3306/Velocity_Motors";
    private static final String USER = "admin";
    private static final String PASSWORD = "";

    public static void main(String[] args) {
        Connection conn = null;
        try {
            conn = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Connected to the database.");
            manageDatabaseOperations(conn);
        } catch (SQLException e) {
            System.err.println("Error connecting to the database: " + e.getMessage());
            e.printStackTrace();
        } finally {
            try {
                if (conn != null) {
                    conn.close();
                }
            } catch (SQLException e) {
                System.err.println("Error closing the connection: " + e.getMessage());
            }
        }
    }

    private static void manageDatabaseOperations(Connection conn) {
        Scanner scanner = new Scanner(System.in);
            boolean exit = false;
            while (!exit) {
                System.out.println("\n1. Add new branch");
                System.out.println("2. Show all branches");
                System.out.println("3. Exit");
                System.out.print("Enter choice: ");
                int choice = scanner.nextInt();
                scanner.nextLine(); // consume newline

                switch (choice) {
                    case 1:
                        addNewBranch(conn, scanner);
                        break;
                    case 2:
                        showAllBranches(conn);
                        break;
                    case 3:
                        System.out.println("Thank you :)");
                        exit = true;
                        break;
                    default:
                        System.out.println("Invalid choice. Please choose again.");
                }

        }
    }

    private static void addNewBranch(Connection conn, Scanner scanner) {
        System.out.print("Enter Branch ID (3 digits max): ");
        String id= scanner.nextLine();
        scanner.nextLine(); // consume newline

        System.out.print("Enter Branch Phone (10 digits): ");
        String phone = scanner.nextLine();
        System.out.print("Enter City (20 characters max): ");
        String city = scanner.nextLine();
        System.out.print("Enter State (20 characters max): ");
        String state = scanner.nextLine();
        System.out.print("Enter Zip Code (5 digits): ");
        String zip = scanner.nextLine();

        System.out.print("Insert a new record (Y/N)? ");
        String userChoice = scanner.nextLine();

        if (userChoice.equalsIgnoreCase("n")) {
            System.out.println("Operation cancelled by user.");
            return;
        }

        try {

        if (!id.matches("\\d{1,3}")) {
            System.err.println("Branch ID must be between 1 and 3 digits.");
            return;
        }
        int Bid = Integer.parseInt(id);
        String query = "SELECT * FROM branch WHERE BranchID = ?";
        try (PreparedStatement statement = conn.prepareStatement(query)) {
            statement.setInt(1, Bid);
            ResultSet resultSet = statement.executeQuery();

            if (resultSet.next()) {
                System.out.println("Error: A branch with the same ID already exists.
Please enter a unique ID.");
                return; // Return to main menu instead of exiting
            }
        } catch (SQLException e) {
            System.out.println("Error checking for duplicate: " + e.getMessage());
            return; // Return to main menu if there's a SQL error during check
        }


            if (phone.length() > 10 || city.length() > 20 || state.length() > 20 ||
zip.length() > 10) {
                throw new IllegalArgumentException("Input data exceeds size limits.
Please adhere to the maximum lengths.");
            }

            if (phone.isEmpty() || city.isEmpty() || state.isEmpty() || zip.isEmpty())
{
                throw new IllegalArgumentException("All fields must be filled. Please
try again.");
            }

            if (!phone.matches("\\d{10}")) {
                throw new IllegalArgumentException("Phone number must be exactly 10
digits.");
            }

            if (!zip.matches("\\d{5}")) {
                throw new IllegalArgumentException("Zip code must be exactly 5
digits.");
            }

            String sql = "INSERT INTO Branch (BranchID, BPhone, City, State, Zip)
VALUES (?, ?, ?, ?, ?)";
            try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

                pstmt.setInt(1, Bid);
                pstmt.setString(2, phone);
                pstmt.setString(3, city);
                pstmt.setString(4, state);
                pstmt.setString(5, zip);
                int affectedRows = pstmt.executeUpdate();
                if (affectedRows == 0) {
                    throw new SQLException("Inserting branch failed, no rows
affected.");
                }

                System.out.println("Branch added successfully.");
            }
        } catch (SQLException e) {
            System.err.println("SQL Error: " + e.getMessage());
            if (e.getErrorCode() == 1062) {
                System.err.println("Duplicate entry for Branch ID.");
            }
        } catch (IllegalArgumentException e) {
            System.err.println("Validation Error: " + e.getMessage());
        }
    }

    private static void showAllBranches(Connection conn) {
        String sql = "SELECT * FROM Branch";
        try (Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                int id = rs.getInt("BranchID");
                String phone = rs.getString("BPhone");
                String city = rs.getString("City");
                String state = rs.getString("State");
                String zip = rs.getString("Zip");
                System.out.printf("Branch ID: %d, Phone:: %s, City: %s, State: %s,
Zip: %s%n", id, phone, city, state, zip);
            }
        } catch (SQLException e) {
            System.err.println("Database error while retrieving branches: " +
e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Notes:

We understand that the insertion of a MID (Manager ID) is required in the database but considering that the employee referenced by the MID which serves as a foreign key in the branch, any attempt to insert a MID will result in a referential integrity error since you did not provide any instructions regarding its insertion in the employee table.