Experiment#		**Student ID**	
Date		**Student Name**	THANOS

12. Java Database Connectivity

Aim/Objective: To understand the how a Java application can be connected with any database in the world for data persistence.

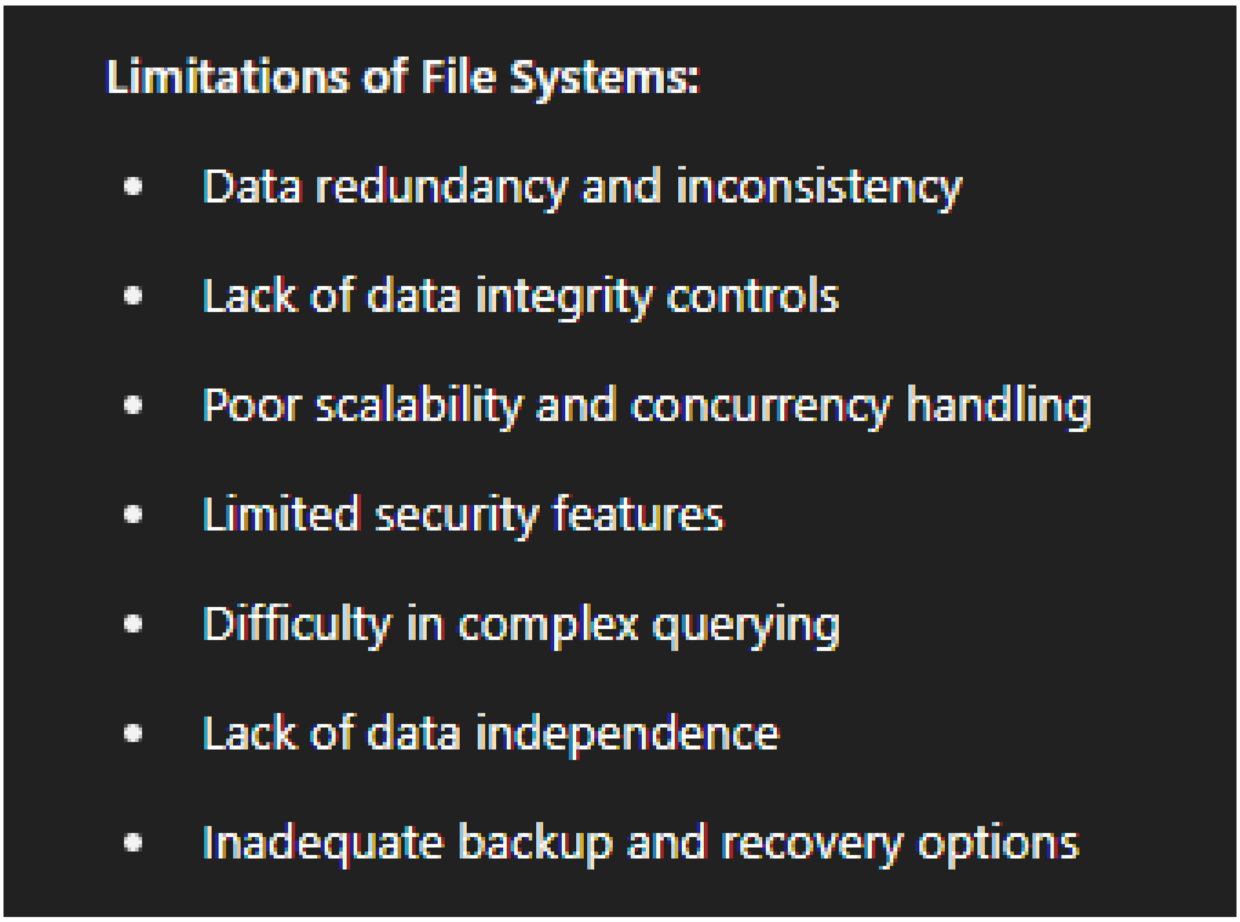
Description: The student will understand the concepts of JDBC.

Pre-Requisites: Classes, Objects and database

Tools: Eclipse IDE for Enterprise Java and Web Developers

Pre-Lab:

1) Explain the limitations of file system making us to use database?

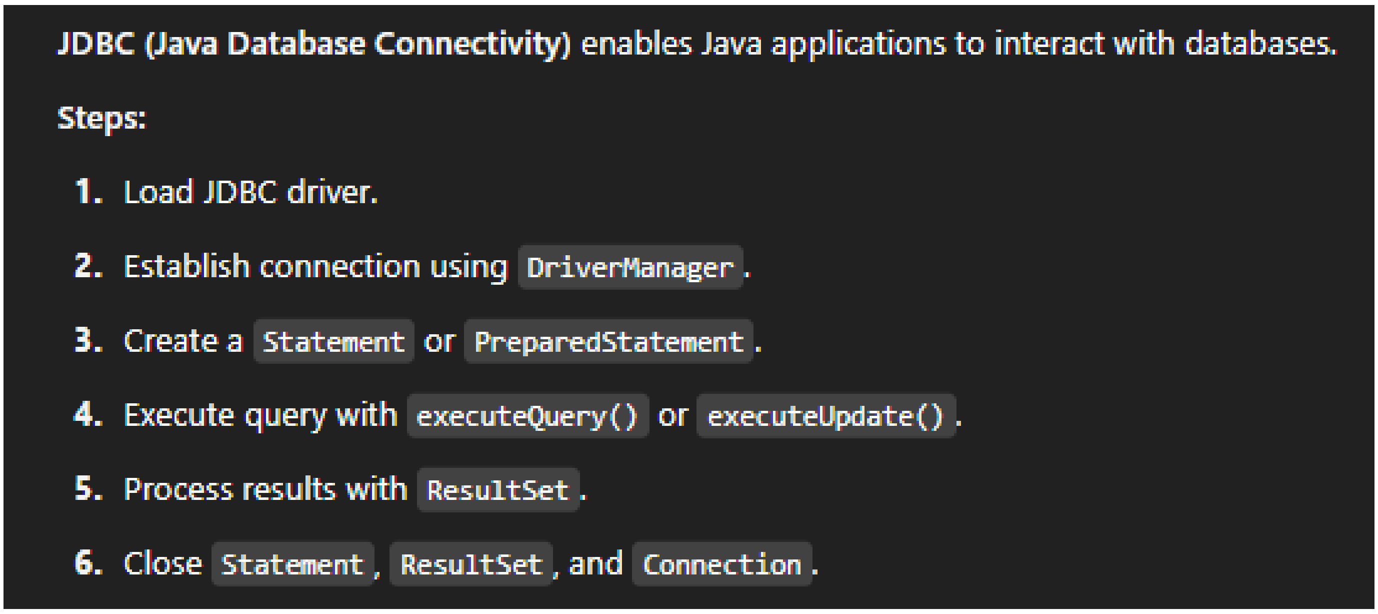


Limitations of File Systems:

- Data redundancy and inconsistency
- Lack of data integrity controls
- Poor scalability and concurrency handling
- Limited security features
- Difficulty in complex querying
- Lack of data independence
- Inadequate backup and recovery options

Experiment#		**Student ID**	
Date		**Student Name**	THANOS

2) Explain the concept of JDBC and its role in Java database connectivity. What are the main steps involved in performing database operations using JDBC?



JDBC (Java Database Connectivity) enables Java applications to interact with databases.

Steps:

1. Load JDBC driver.
2. Establish connection using `DriverManager`.
3. Create a `Statement` or `PreparedStatement`.
4. Execute query with `executeQuery()` or `executeUpdate()`.
5. Process results with `ResultSet`.
6. Close `Statement`, `ResultSet`, and `Connection`.

Experiment#		**Student ID**	
Date		**Student Name**	THANOS

In-Lab:

1) You are developing a Student Information Management System where student details such as student ID, name, address, and program can be managed through a web interface. Implement a servlet called StudentDetailsServlet that accepts student details through a form submission and stores these details in a database using JDBC. Perform the following steps to create and manage the student database:
   Requirements

   A. **Create the Student Database**:
      a. Write a JDBC program to create a Student database.
   B. **Create the Registration Table**:
      a. Write a JDBC program to create a Registration table inside the Student database with fields id, name, address, and program, where id is the primary key.
   C. **Insert Records**:
      a. Write a JDBC program to insert four records into the Registration table.
   D. **Display Records**:
      a. Write a JDBC program to display the records inserted into the Registration table.
   E. **Update Records**:
      a. Write a JDBC program to update the program of students whose id values are 100 and 101.
   F. **Delete Records**:
      a. Write a JDBC program to delete the student record whose id is 101.

   Procedure/Program:

A. Create the Student Database

```
import java.sql.*;

public class CreateDatabase {
    public static void main(String[] args) {
        try {
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/", "root",
"Root@123");
            Statement stmt = conn.createStatement();
            stmt.executeUpdate("CREATE DATABASE Student");
            System.out.println("Database created successfully!");
```

```
  } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

B. Create the Registration Table

```
import java.sql.*;

public class CreateTable {
    public static void main(String[] args) {
        try {
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/Student", "root",
" Root@123");
            Statement stmt = conn.createStatement();
            String sql = "CREATE TABLE Registration (" +
                         "id INT PRIMARY KEY, " +
                         "name VARCHAR(100), " +
                         "address VARCHAR(200), " +
                         "program VARCHAR(100))";
            stmt.executeUpdate(sql);
            System.out.println("Registration table created.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

C. Insert Records

```
import java.sql.*;

public class InsertRecords {
    public static void main(String[] args) {
        try {
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/Student", "root",
 " Root@123");
            PreparedStatement ps = conn.prepareStatement("INSERT INTO
Registration VALUES (?, ?, ?, ?)");

            ps.setInt(1, 100);
            ps.setString(2, "John Doe");
            ps.setString(3, "NY");
            ps.setString(4, "B.Tech");
            ps.executeUpdate();

            ps.setInt(1, 101);
            ps.setString(2, "Jane Smith");
            ps.setString(3, "LA");
            ps.setString(4, "B.Sc");
            ps.executeUpdate();

            ps.setInt(1, 102);
            ps.setString(2, "Mike Ross");
            ps.setString(3, "Chicago");
            ps.setString(4, "MBA");
            ps.executeUpdate();

            ps.setInt(1, 103);
            ps.setString(2, "Rachel Zane");
            ps.setString(3, "Boston");
            ps.setString(4, "BCA");
            ps.executeUpdate();
            System.out.println("Records inserted.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

D. Display Records

```
import java.sql.*;

public class DisplayRecords {
    public static void main(String[] args) {
        try {
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/Student", "root",
" Root@123");
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM Registration");

            while (rs.next()) {
                System.out.println("ID: " + rs.getInt("id"));
                System.out.println("Name: " + rs.getString("name"));
                System.out.println("Address: " + rs.getString("address"));
                System.out.println("Program: " + rs.getString("program"));
                System.out.println("------------------------");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

E. Update Records

```
import java.sql.*;

public class UpdateRecords {
    public static void main(String[] args) {
        try {
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/Student", "root",
 " Root@123");
            PreparedStatement ps = conn.prepareStatement("UPDATE Registration
SET program=? WHERE id=?");

            ps.setString(1, "M.Tech");
            ps.setInt(2, 100);
            ps.executeUpdate();

            ps.setString(1, "M.Sc");
            ps.setInt(2, 101);
            ps.executeUpdate();

            System.out.println("Records updated.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

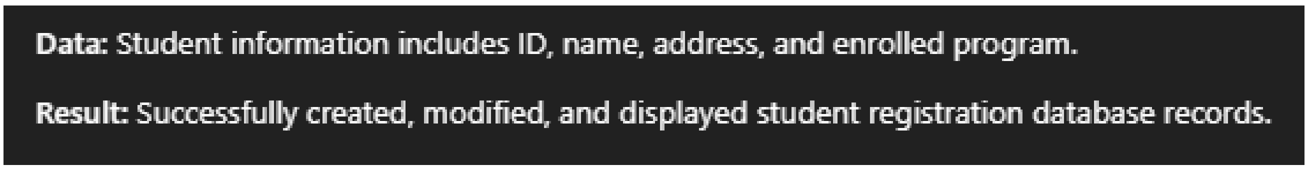
F. Delete Records

```
import java.sql.*;

public class DeleteRecords {
    public static void main(String[] args) {
        try {
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/Student", "root",
 " Root@123");
            PreparedStatement ps = conn.prepareStatement("DELETE FROM
Registration WHERE id=?");
            ps.setInt(1, 101);
            ps.executeUpdate();
            System.out.println("Record deleted.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

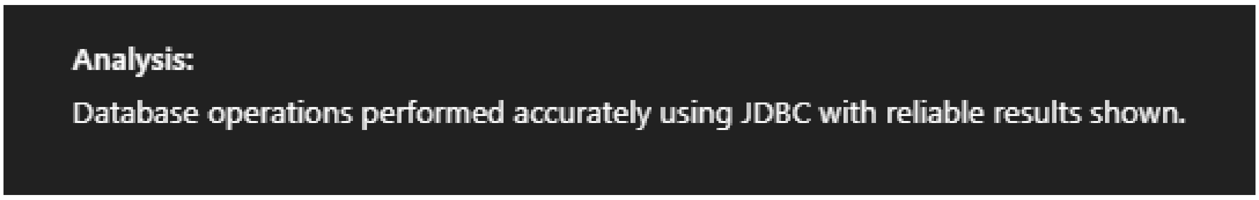
Experiment#		**Student ID**	
Date		**Student Name**	THANOS

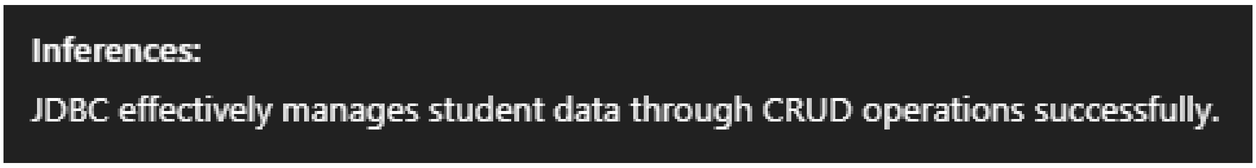
✓ **Data and Results:**


Data: Student information includes ID, name, address, and enrolled program.

Result: Successfully created, modified, and displayed student registration database records.

✓ **Analysis and Inferences:**


Analysis:
Database operations performed accurately using JDBC with reliable results shown.


Inferences:
JDBC effectively manages student data through CRUD operations successfully.

Experiment#		**Student ID**	
Date		**Student Name**	THANOS

VIVA-VOCE Questions (In-Lab):

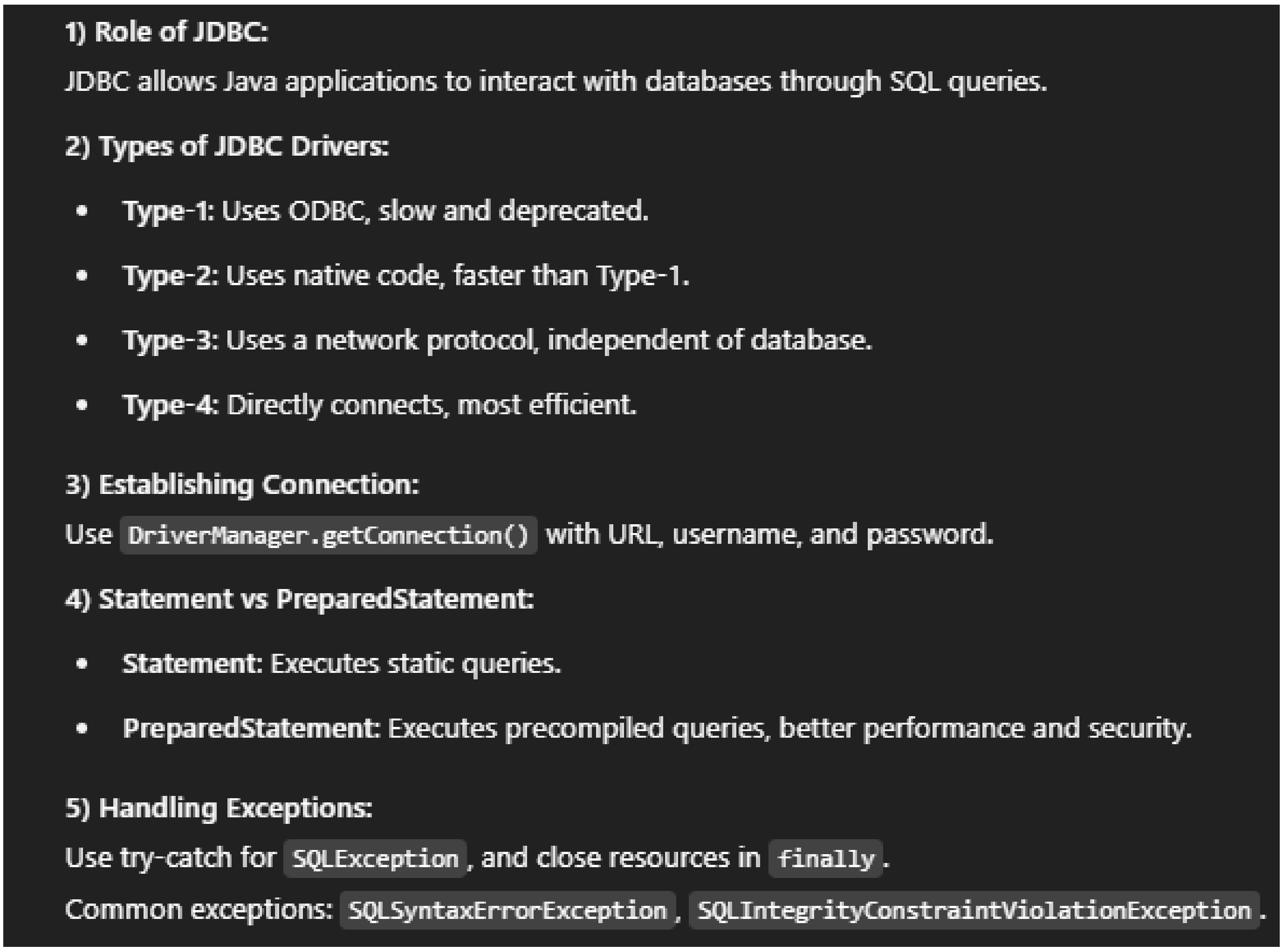
1) What is the role of JDBC in Java database connectivity?

2) What are the different types of JDBC drivers? Explain their differences and advantages.

3) How do you establish a database connection using JDBC?

4) Explain the difference between Statement and PreparedStatement in JDBC.

5) How do you handle exceptions related to database operations in JDBC? What are some common JDBC-related exceptions?



1) Role of JDBC:

JDBC allows Java applications to interact with databases through SQL queries.

2) Types of JDBC Drivers:

- **Type-1:** Uses ODBC, slow and deprecated.
- **Type-2:** Uses native code, faster than Type-1.
- **Type-3:** Uses a network protocol, independent of database.
- **Type-4:** Directly connects, most efficient.

3) Establishing Connection:

Use `DriverManager.getConnection()` with URL, username, and password.

4) Statement vs PreparedStatement:

- **Statement:** Executes static queries.
- **PreparedStatement:** Executes precompiled queries, better performance and security.

5) Handling Exceptions:

Use try-catch for `SQLException`, and close resources in `finally`.

Common exceptions: `SQLSyntaxErrorException`, `SQLIntegrityConstraintViolationException`.

Post-Lab:

1) Create a JDBC program to connect to a PostgreSQL database. Specifically, you need to create a "test" database in the PostgreSQL server and write a JDBC program that checks whether the connection to the database is successful. If the connection is successful, the program should display the message "Connected to the database". If the username or password is incorrect, the program should display "Invalid username or password".
   Procedure/Program:

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

public class PostgreSQLConnection {
    public static void main(String[] args) {
        String url = "jdbc:postgresql://localhost:5432/test";
        String user = "root";
        String password = "Root@123";

        try {
            Connection conn = DriverManager.getConnection(url, user,
    password);
            System.out.println("Connected to the database");
            conn.close();
        } catch (SQLException e) {
            if (e.getSQLState().equals("28P01")) {
```

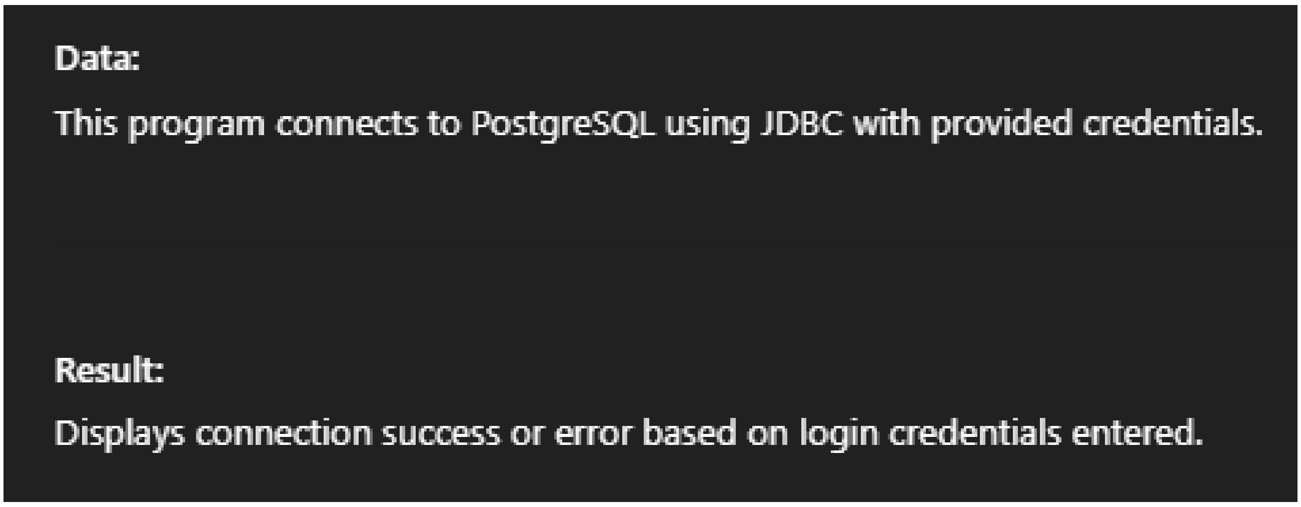
```
System.out.println("Invalid username or password");



      } else {
            System.out.println("Database connection failed: " +
    e.getMessage());
         }
      }
   }
}
```

Experiment#		**Student ID**	
Date		**Student Name**	THANOS

✓ **Data and Results:**



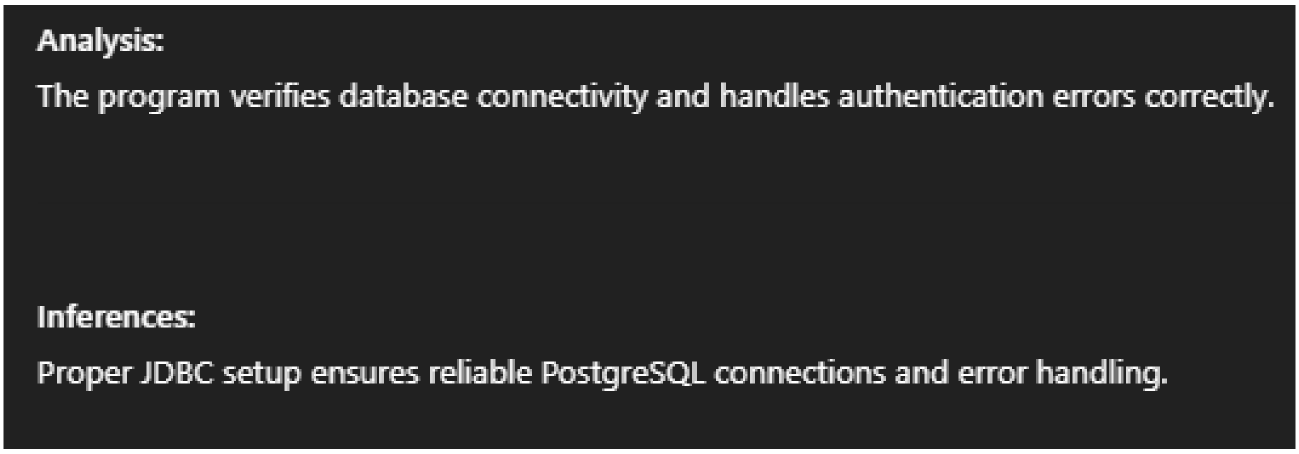
Data:

This program connects to PostgreSQL using JDBC with provided credentials.

Result:

Displays connection success or error based on login credentials entered.

✓ **Analysis and Inferences:**



Analysis:

The program verifies database connectivity and handles authentication errors correctly.

Inferences:

Proper JDBC setup ensures reliable PostgreSQL connections and error handling.

Evaluator Remark (if Any):	**Marks Secured _____ out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.