Problem Statement 1: Establish a Basic Connection

Description:
Write a Java program to establish a connection to a PostgreSQL database named `test_db` using JDBC. Ensure the program prints a success message if the connection is successful and an error message if it fails.

Requirements:

- Use `DriverManager` to establish the connection.
- Handle `SQLException` appropriately.
- Print appropriate messages based on connection status.

Problem Statement 2: Execute a Simple Query

Description:
Create a Java program that connects to a MySQL database named `company_db` and executes a simple `SELECT` query to retrieve all records from a table named `employees`. Print the details of each record retrieved.

Requirements:

- Use `PreparedStatement` to execute the query.
- Handle potential `SQLException`.
- Print the results in a readable format.

Problem Statement 3: Insert Data Using PreparedStatement

Description:
Write a Java program that connects to an Oracle database named `inventory_db` and inserts a new record into a table named `products`. The record should include product ID, name, price, and quantity. Use `PreparedStatement` to insert the data.

Requirements:

- Use placeholders in the `PreparedStatement`.
- Insert sample data into the `products` table.
- Print a confirmation message after a successful insertion.

Problem Statement 4: Handle Transactions

Description:
Develop a Java program that connects to a PostgreSQL database named `bank_db`. Implement a transaction where you transfer money from one account to another within the same table named `accounts`. Ensure that if any step of the transaction fails, the entire transaction is rolled back.

Requirements:

- Use `Connection.setAutoCommit(false)` to manage transactions.
- Handle `SQLException` to roll back transactions if needed.
- Ensure that the balance is updated correctly or the transaction is rolled back in case of an error.

Problem Statement 5: Handle ResultSet and Update Data

Description:
Write a Java program that connects to a SQLite database named `school_db`. Retrieve all records from a table named `students`, update the `grade` of a specific student, and then print the updated records.

Requirements:

- Use `ResultSet` to retrieve and navigate through the records.
- Update the student's grade using `PreparedStatement`.
- Ensure that changes are committed and reflect in the results.