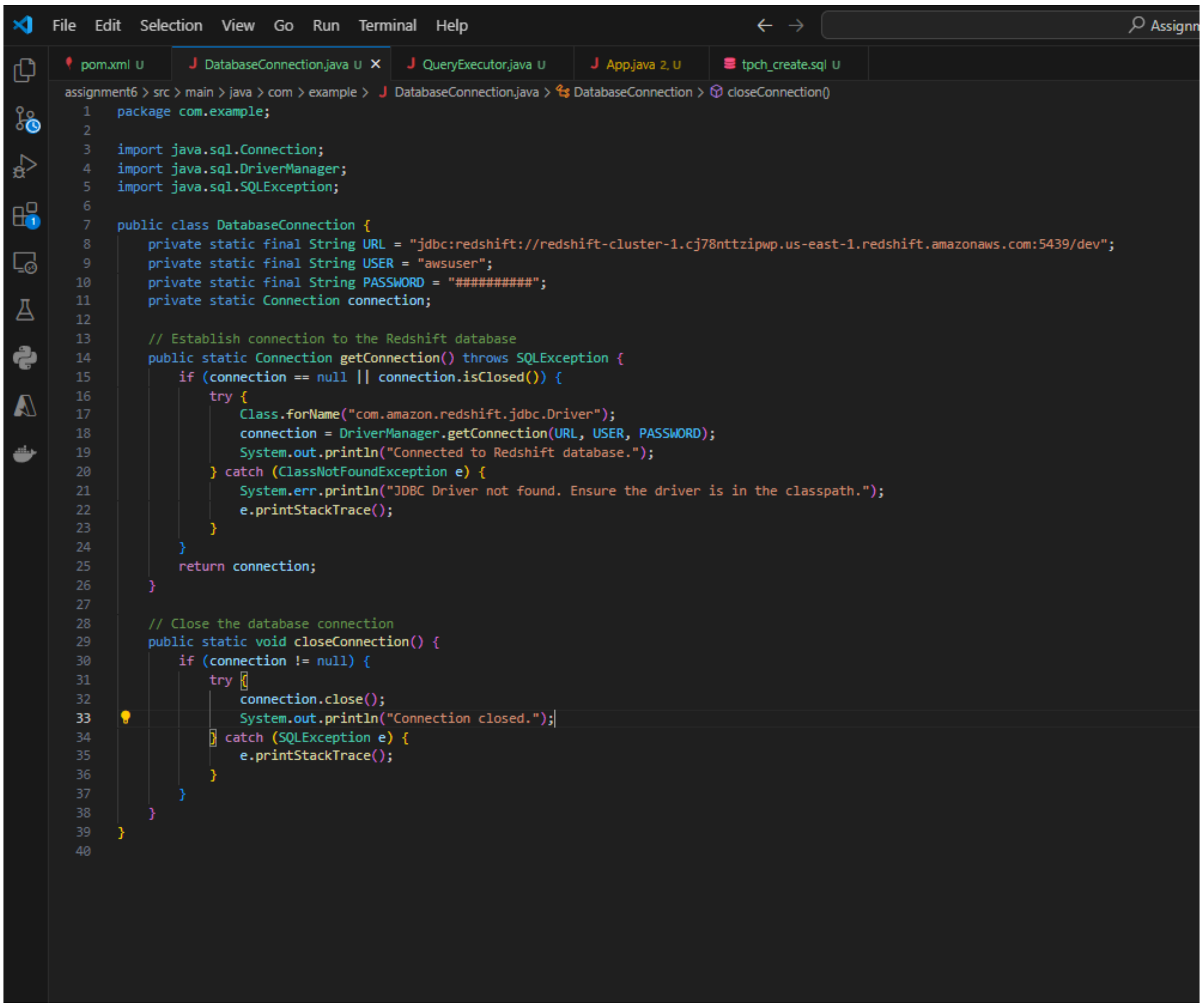
Big data Management

Assignment 6

Jeyadev L

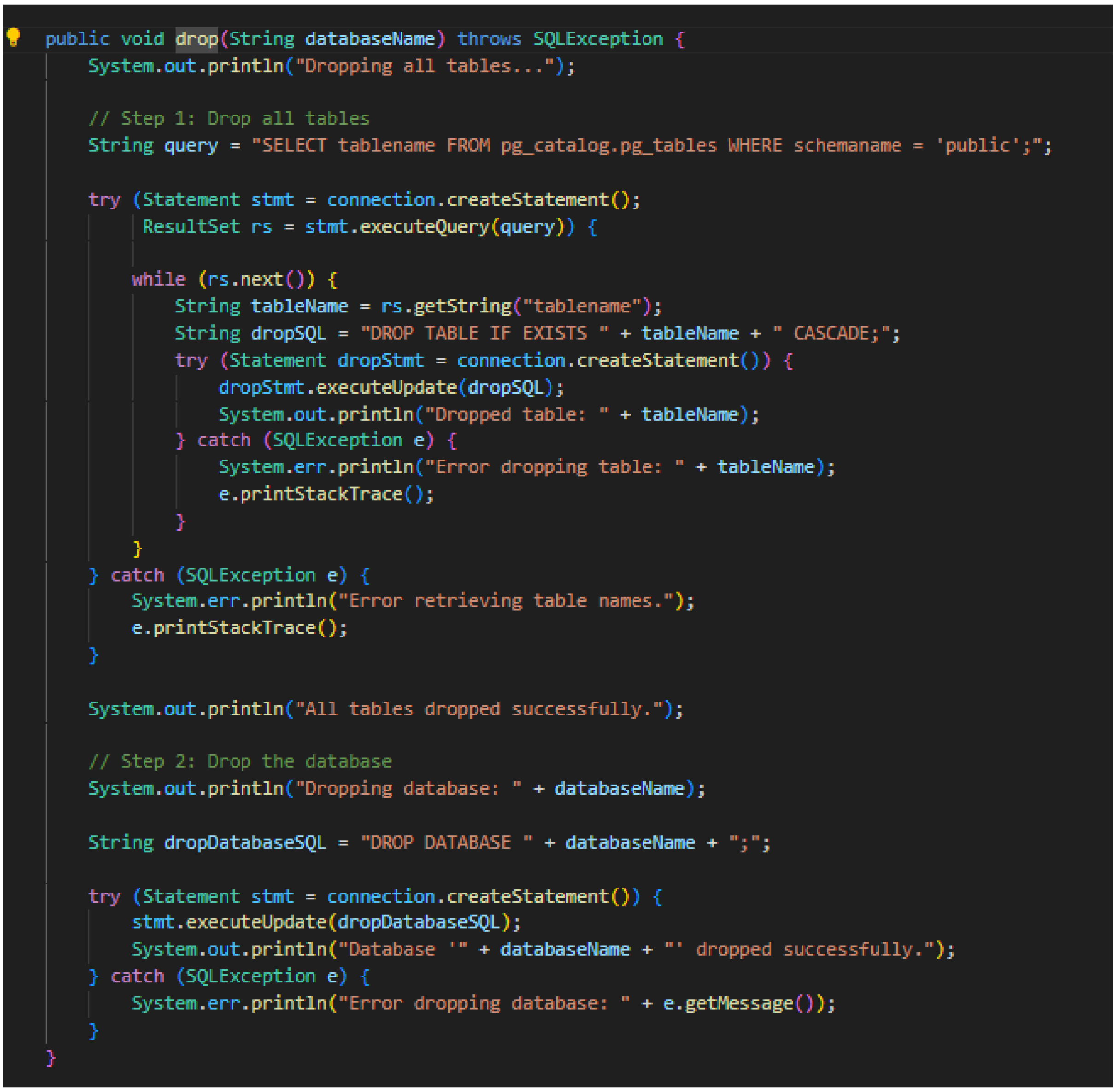
G23AI2071

Helper Class


File Edit Selection View Go Run Terminal Help
Assignm
pom.xml U
DatabaseConnection.java U
QueryExecutor.java U
App.java 2, U
tpch_create.sql U
assignment6 > src > main > java > com > example > DatabaseConnection.java > DatabaseConnection > closeConnection()

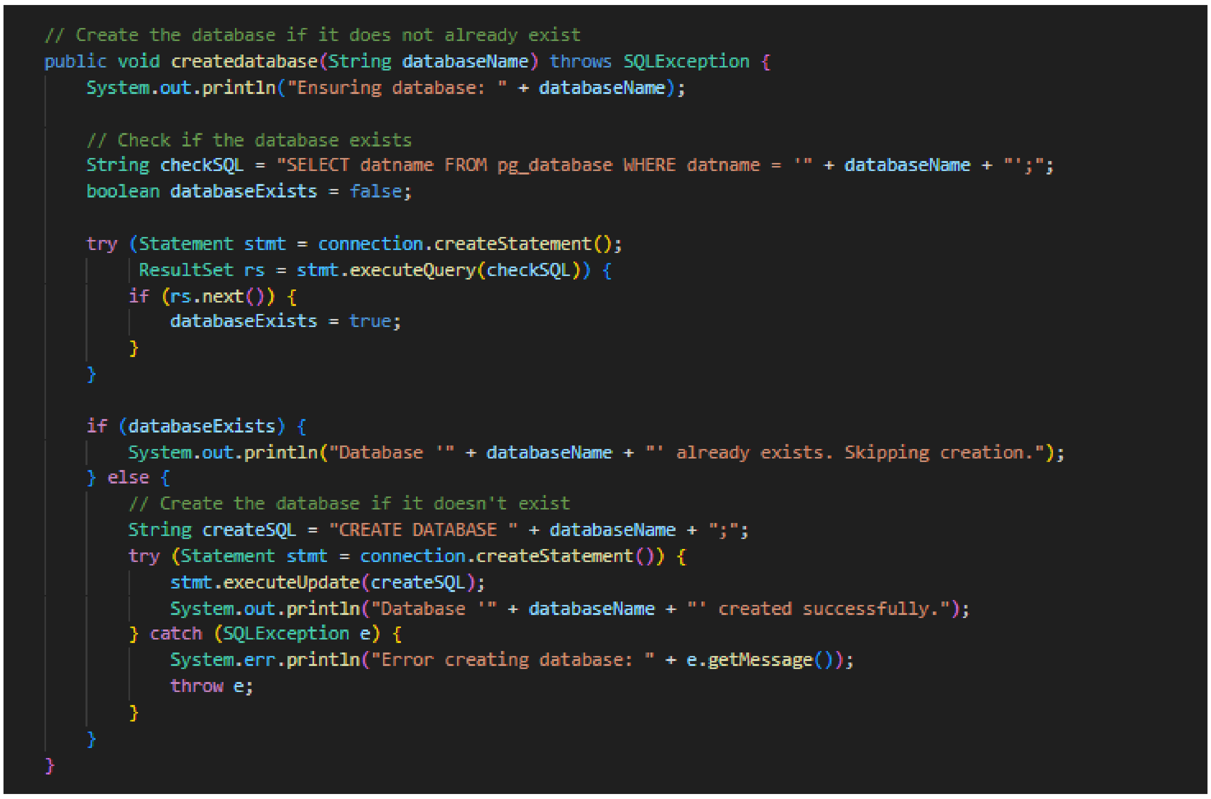
```java
package com.example;

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

public class DatabaseConnection {
    private static final String URL = "jdbc:redshift://redshift-cluster-1.cj78nttzipwp.us-east-1.redshift.amazonaws.com:5439/dev";
    private static final String USER = "awsuser";
    private static final String PASSWORD = "##########";
    private static Connection connection;

    // Establish connection to the Redshift database
    public static Connection getConnection() throws SQLException {
        if (connection == null || connection.isClosed()) {
            try {
                Class.forName("com.amazon.redshift.jdbc.Driver");
                connection = DriverManager.getConnection(URL, USER, PASSWORD);
                System.out.println("Connected to Redshift database.");
            } catch (ClassNotFoundException e) {
                System.err.println("JDBC Driver not found. Ensure the driver is in the classpath.");
                e.printStackTrace();
            }
        }
        return connection;
    }

    // Close the database connection
    public static void closeConnection() {
        if (connection != null) {
            try {
                connection.close();
                System.out.println("Connection closed.");
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Drop Tables and Database

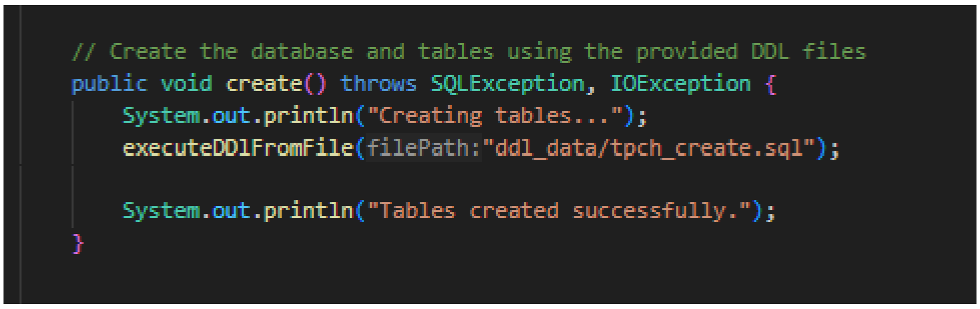


```java
public void drop(String databaseName) throws SQLException {
    System.out.println("Dropping all tables...");

    // Step 1: Drop all tables
    String query = "SELECT tablename FROM pg_catalog.pg_tables WHERE schemaname = 'public';";

    try (Statement stmt = connection.createStatement();
         ResultSet rs = stmt.executeQuery(query)) {

        while (rs.next()) {
            String tableName = rs.getString("tablename");
            String dropSQL = "DROP TABLE IF EXISTS " + tableName + " CASCADE;";
            try (Statement dropStmt = connection.createStatement()) {
                dropStmt.executeUpdate(dropSQL);
                System.out.println("Dropped table: " + tableName);
            } catch (SQLException e) {
                System.err.println("Error dropping table: " + tableName);
                e.printStackTrace();
            }
        }
    } catch (SQLException e) {
        System.err.println("Error retrieving table names.");
        e.printStackTrace();
    }

    System.out.println("All tables dropped successfully.");

    // Step 2: Drop the database
    System.out.println("Dropping database: " + databaseName);

    String dropDatabaseSQL = "DROP DATABASE " + databaseName + ";";

    try (Statement stmt = connection.createStatement()) {
        stmt.executeUpdate(dropDatabaseSQL);
        System.out.println("Database '" + databaseName + "' dropped successfully.");
    } catch (SQLException e) {
        System.err.println("Error dropping database: " + e.getMessage());
    }
}
```

Create Database

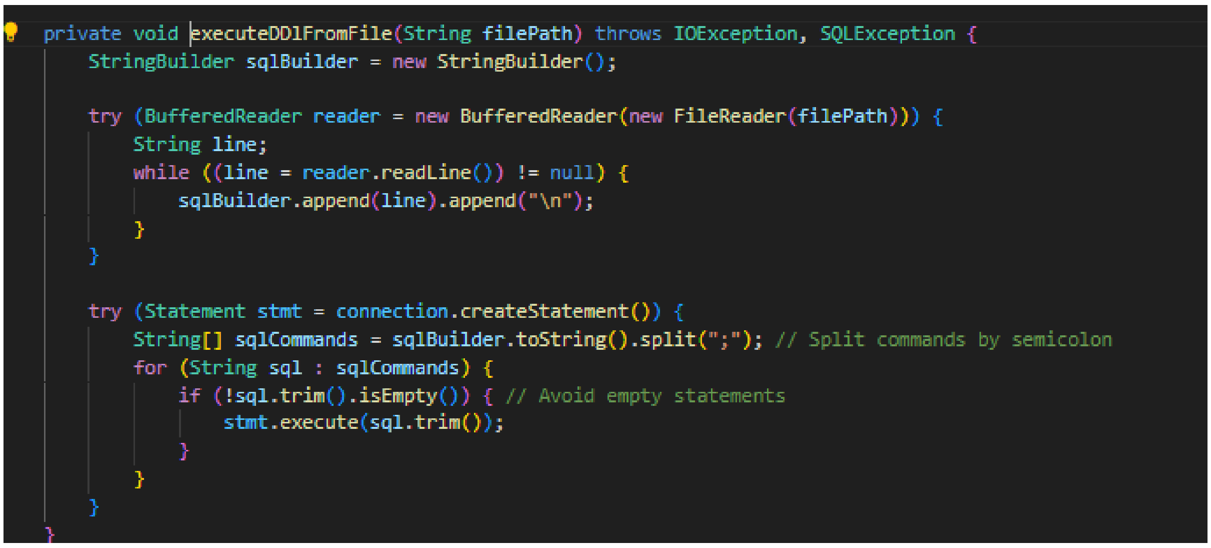


```java
// Create the database if it does not already exist
public void createdatabase(String databaseName) throws SQLException {
    System.out.println("Ensuring database: " + databaseName);

    // Check if the database exists
    String checkSQL = "SELECT datname FROM pg_database WHERE datname = '" + databaseName + "';";
    boolean databaseExists = false;

    try (Statement stmt = connection.createStatement();
         ResultSet rs = stmt.executeQuery(checkSQL)) {
        if (rs.next()) {
            databaseExists = true;
        }
    }

    if (databaseExists) {
        System.out.println("Database '" + databaseName + "' already exists. Skipping creation.");
    } else {
        // Create the database if it doesn't exist
        String createSQL = "CREATE DATABASE " + databaseName + ";";
        try (Statement stmt = connection.createStatement()) {
            stmt.executeUpdate(createSQL);
            System.out.println("Database '" + databaseName + "' created successfully.");
        } catch (SQLException e) {
            System.err.println("Error creating database: " + e.getMessage());
            throw e;
        }
    }
}
```

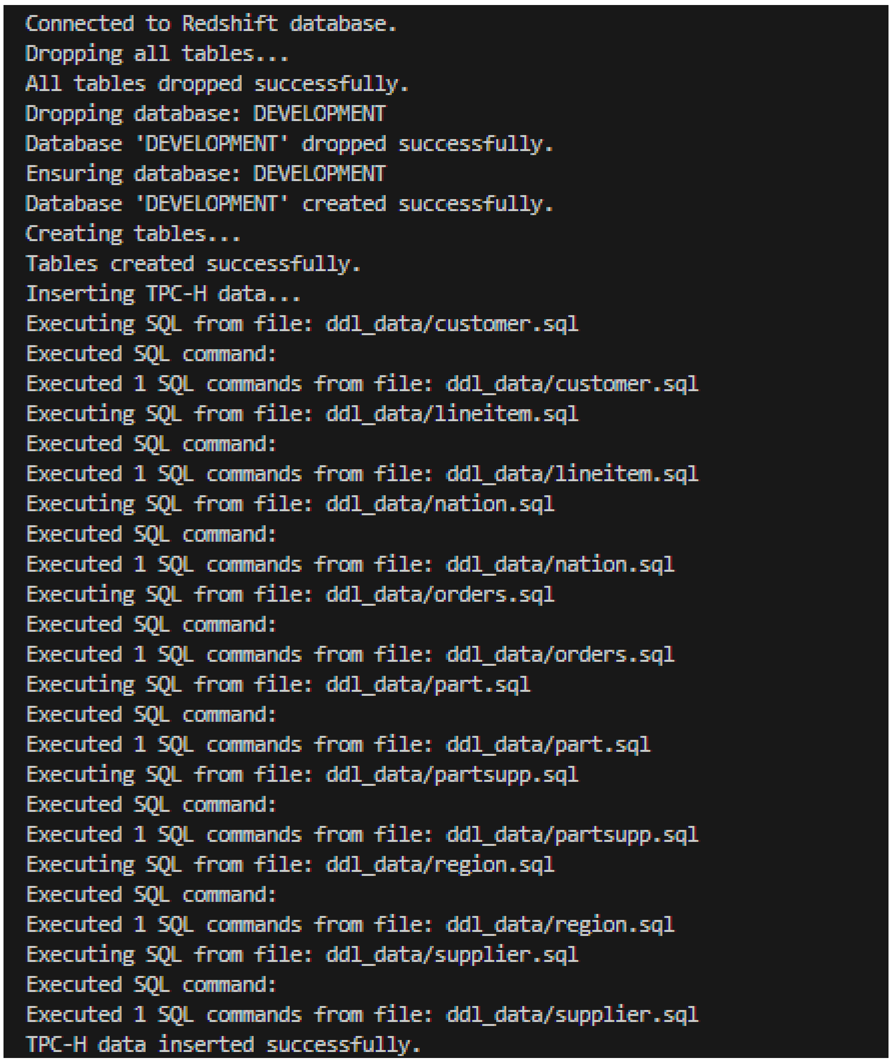


```java
// Create the database and tables using the provided DDL files
public void create() throws SQLException, IOException {
    System.out.println("Creating tables...");
    executeDDlFromFile(filePath:"ddl_data/tpch_create.sql");

    System.out.println("Tables created successfully.");
}
```

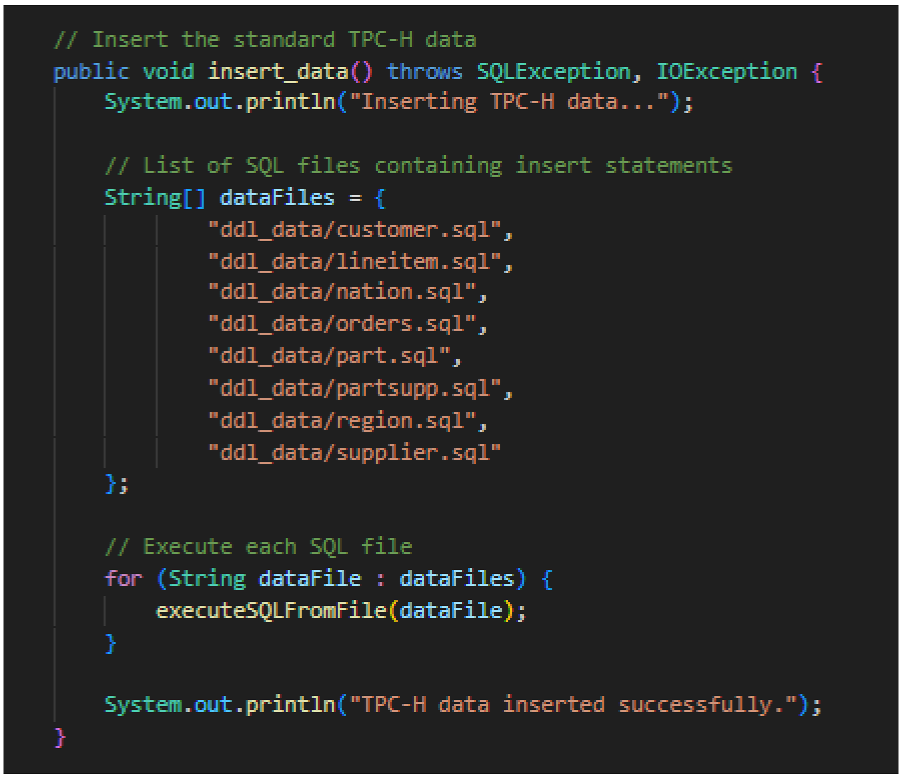
Helper Class



```java
private void executeDDlFromFile(String filePath) throws IOException, SQLException {
    StringBuilder sqlBuilder = new StringBuilder();

    try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
        String line;
        while ((line = reader.readLine()) != null) {
            sqlBuilder.append(line).append("\n");
        }
    }

    try (Statement stmt = connection.createStatement()) {
        String[] sqlCommands = sqlBuilder.toString().split(";"); // Split commands by semicolon
        for (String sql : sqlCommands) {
            if (!sql.trim().isEmpty()) { // Avoid empty statements
                stmt.execute(sql.trim());
            }
        }
    }
}
```

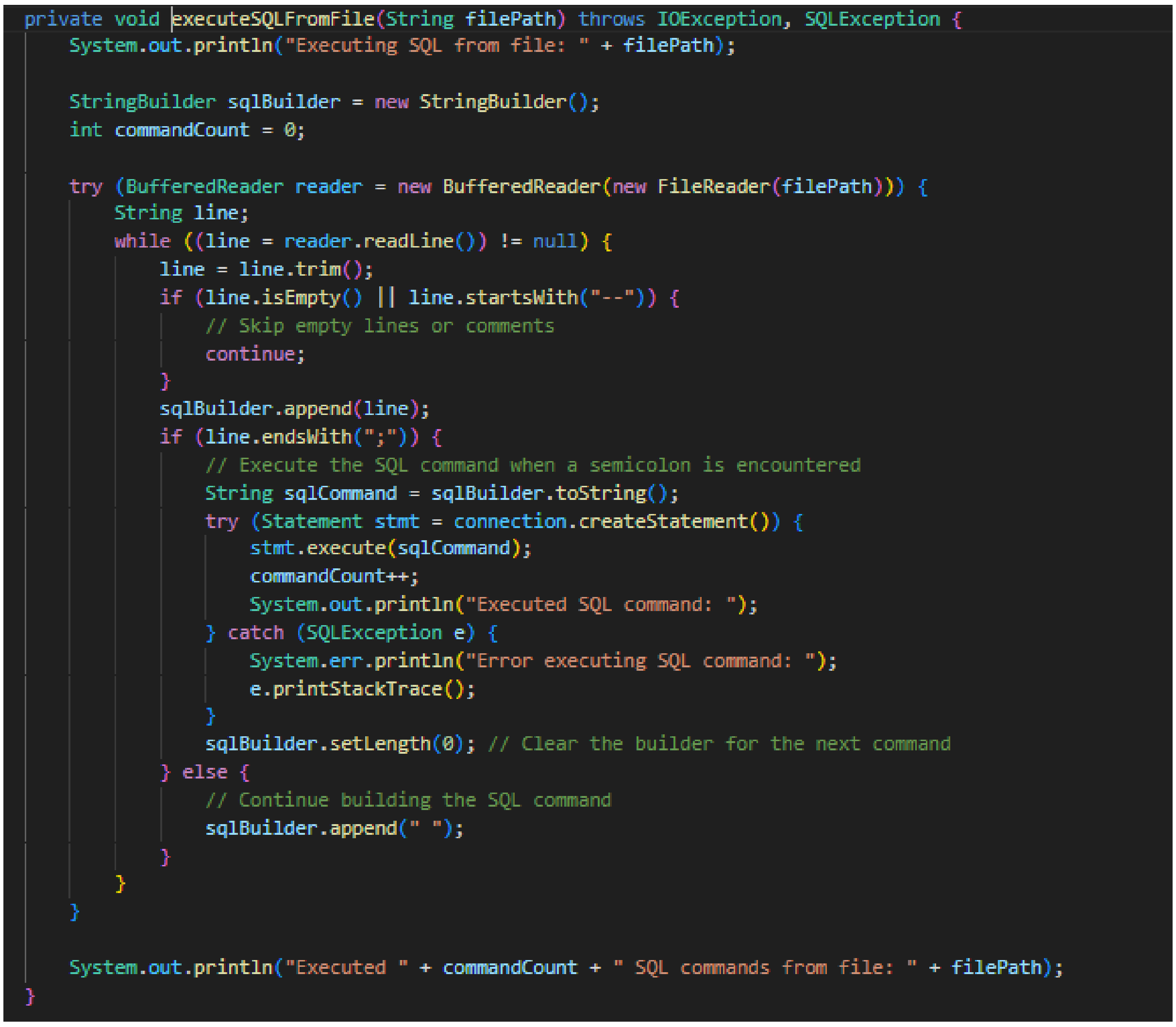


```
Connected to Redshift database.
Dropping all tables...
All tables dropped successfully.
Dropping database: DEVELOPMENT
Database 'DEVELOPMENT' dropped successfully.
Ensuring database: DEVELOPMENT
Database 'DEVELOPMENT' created successfully.
Creating tables...
Tables created successfully.
Inserting TPC-H data...
Executing SQL from file: ddl_data/customer.sql
Executed SQL command:
Executed 1 SQL commands from file: ddl_data/customer.sql
Executing SQL from file: ddl_data/lineitem.sql
Executed SQL command:
Executed 1 SQL commands from file: ddl_data/lineitem.sql
Executing SQL from file: ddl_data/nation.sql
Executed SQL command:
Executed 1 SQL commands from file: ddl_data/nation.sql
Executing SQL from file: ddl_data/orders.sql
Executed SQL command:
Executed 1 SQL commands from file: ddl_data/orders.sql
Executing SQL from file: ddl_data/part.sql
Executed SQL command:
Executed 1 SQL commands from file: ddl_data/part.sql
Executing SQL from file: ddl_data/partsupp.sql
Executed SQL command:
Executed 1 SQL commands from file: ddl_data/partsupp.sql
Executing SQL from file: ddl_data/region.sql
Executed SQL command:
Executed 1 SQL commands from file: ddl_data/region.sql
Executing SQL from file: ddl_data/supplier.sql
Executed SQL command:
Executed 1 SQL commands from file: ddl_data/supplier.sql
TPC-H data inserted successfully.
```



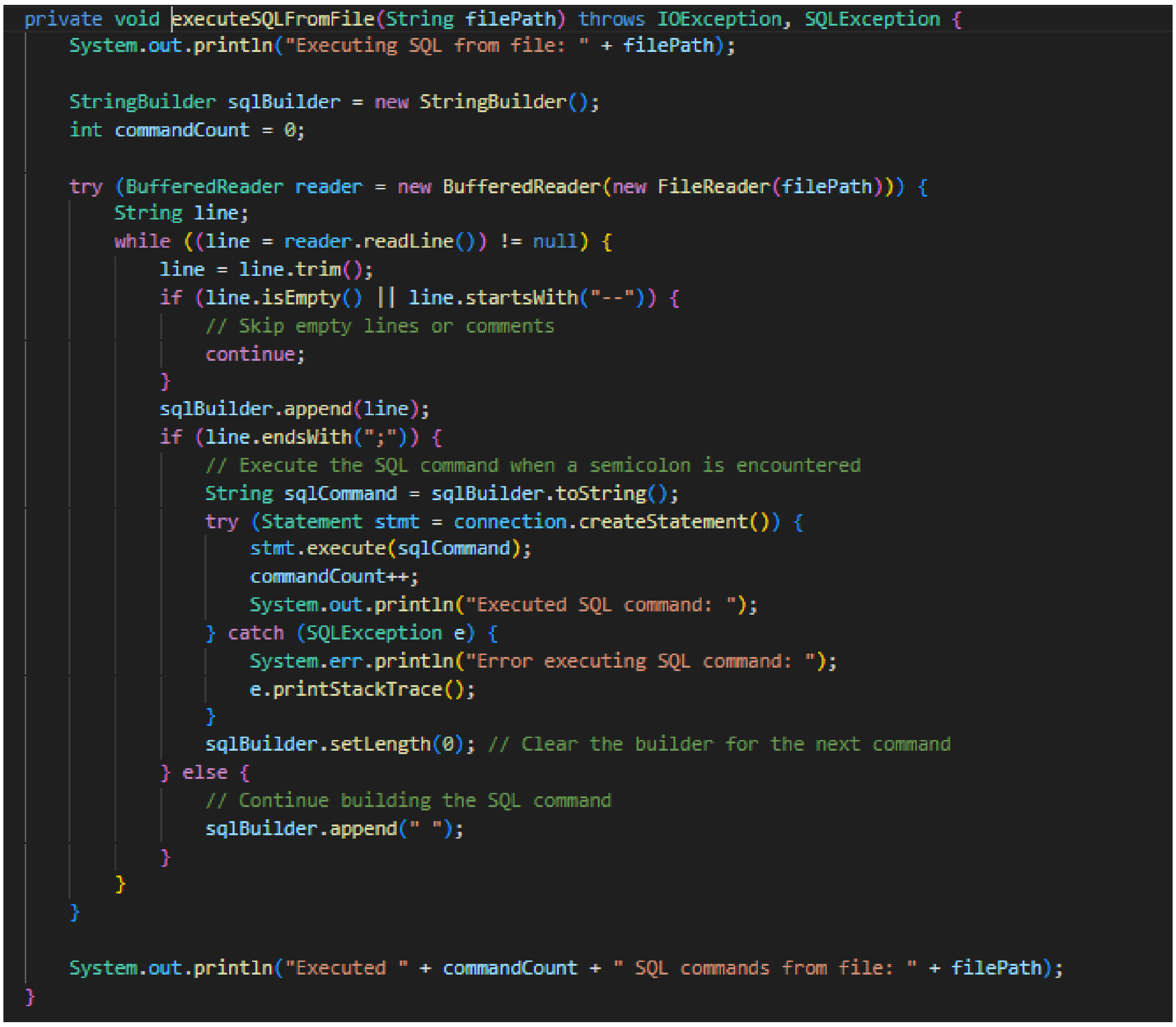
```java
// Insert the standard TPC-H data
public void insert_data() throws SQLException, IOException {
    System.out.println("Inserting TPC-H data...");

    // List of SQL files containing insert statements
    String[] dataFiles = {
            "ddl_data/customer.sql",
            "ddl_data/lineitem.sql",
            "ddl_data/nation.sql",
            "ddl_data/orders.sql",
            "ddl_data/part.sql",
            "ddl_data/partsupp.sql",
            "ddl_data/region.sql",
            "ddl_data/supplier.sql"
    };

    // Execute each SQL file
    for (String dataFile : dataFiles) {
        executeSQLFromFile(dataFile);
    }

    System.out.println("TPC-H data inserted successfully.");
}
```

Helper Class



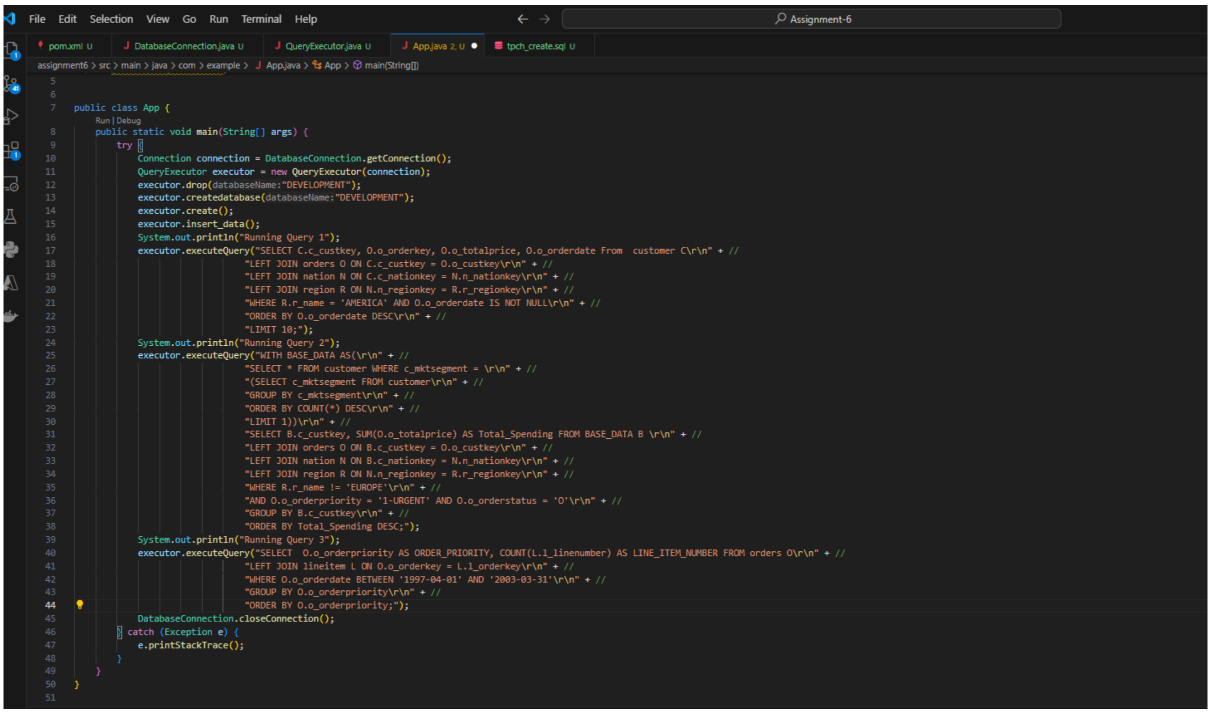
```java
private void executeSQLFromFile(String filePath) throws IOException, SQLException {
    System.out.println("Executing SQL from file: " + filePath);

    StringBuilder sqlBuilder = new StringBuilder();
    int commandCount = 0;

    try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
        String line;
        while ((line = reader.readLine()) != null) {
            line = line.trim();
            if (line.isEmpty() || line.startsWith("--")) {
                // Skip empty lines or comments
                continue;
            }
            sqlBuilder.append(line);
            if (line.endsWith(";")) {
                // Execute the SQL command when a semicolon is encountered
                String sqlCommand = sqlBuilder.toString();
                try (Statement stmt = connection.createStatement()) {
                    stmt.execute(sqlCommand);
                    commandCount++;
                    System.out.println("Executed SQL command: ");
                } catch (SQLException e) {
                    System.err.println("Error executing SQL command: ");
                    e.printStackTrace();
                }
                sqlBuilder.setLength(0); // Clear the builder for the next command
            } else {
                // Continue building the SQL command
                sqlBuilder.append(" ");
            }
        }
    }

    System.out.println("Executed " + commandCount + " SQL commands from file: " + filePath);
}
```

Helper Class



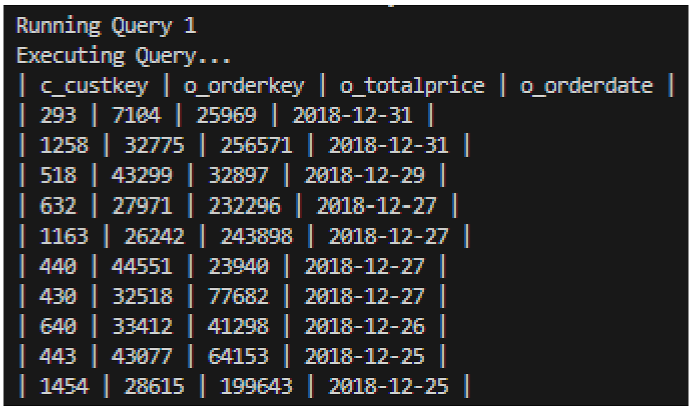
```java
private void executeSQLFromFile(String filePath) throws IOException, SQLException {
    System.out.println("Executing SQL from file: " + filePath);

    StringBuilder sqlBuilder = new StringBuilder();
    int commandCount = 0;

    try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
        String line;
        while ((line = reader.readLine()) != null) {
            line = line.trim();
            if (line.isEmpty() || line.startsWith("--")) {
                // Skip empty lines or comments
                continue;
            }
            sqlBuilder.append(line);
            if (line.endsWith(";")) {
                // Execute the SQL command when a semicolon is encountered
                String sqlCommand = sqlBuilder.toString();
                try (Statement stmt = connection.createStatement()) {
                    stmt.execute(sqlCommand);
                    commandCount++;
                    System.out.println("Executed SQL command: ");
                } catch (SQLException e) {
                    System.err.println("Error executing SQL command: ");
                    e.printStackTrace();
                }
                sqlBuilder.setLength(0); // Clear the builder for the next command
            } else {
                // Continue building the SQL command
                sqlBuilder.append(" ");
            }
        }
    }

    System.out.println("Executed " + commandCount + " SQL commands from file: " + filePath);
}
```

Main Call Function


File Edit Selection View Go Run Terminal Help
Assignment-6
pom.xml U
DatabaseConnection.java U
QueryExecutor.java U
App.java 2, U
tpch_create.sql U
assignment6 > src > main > java > com > example > App.java > App > main(String[])
Run | Debug

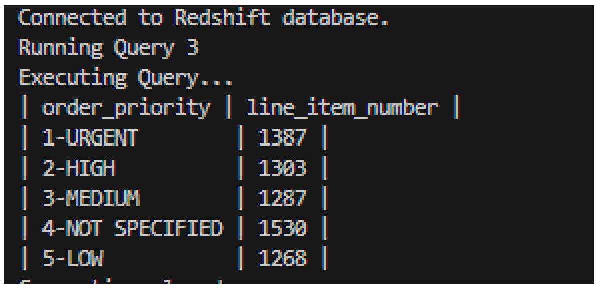
```java


public class App {
    public static void main(String[] args) {
        try {
            Connection connection = DatabaseConnection.getConnection();
            QueryExecutor executor = new QueryExecutor(connection);
            executor.drop(databaseName:"DEVELOPMENT");
            executor.createdatabase(databaseName:"DEVELOPMENT");
            executor.create();
            executor.insert_data();
            System.out.println("Running Query 1");
            executor.executeQuery("SELECT C.c_custkey, O.o_orderkey, O.o_totalprice, O.o_orderdate From  customer C\r\n" + //
                                 "LEFT JOIN orders O ON C.c_custkey = O.o_custkey\r\n" + //
                                 "LEFT JOIN nation N ON C.c_nationkey = N.n_nationkey\r\n" + //
                                 "LEFT JOIN region R ON N.n_regionkey = R.r_regionkey\r\n" + //
                                 "WHERE R.r_name = 'AMERICA' AND O.o_orderdate IS NOT NULL\r\n" + //
                                 "ORDER BY O.o_orderdate DESC\r\n" + //
                                 "LIMIT 10;");
            System.out.println("Running Query 2");
            executor.executeQuery("WITH BASE_DATA AS(\r\n" + //
                                 "SELECT * FROM customer WHERE c_mktsegment = \r\n" + //
                                 "(SELECT c_mktsegment FROM customer\r\n" + //
                                 "GROUP BY c_mktsegment\r\n" + //
                                 "ORDER BY COUNT(*) DESC\r\n" + //
                                 "LIMIT 1))\r\n" + //
                                 "SELECT B.c_custkey, SUM(O.o_totalprice) AS Total_Spending FROM BASE_DATA B \r\n" + //
                                 "LEFT JOIN orders O ON B.c_custkey = O.o_custkey\r\n" + //
                                 "LEFT JOIN nation N ON B.c_nationkey = N.n_nationkey\r\n" + //
                                 "LEFT JOIN region R ON N.n_regionkey = R.r_regionkey\r\n" + //
                                 "WHERE R.r_name != 'EUROPE'\r\n" + //
                                 "AND O.o_orderpriority = '1-URGENT' AND O.o_orderstatus = 'O'\r\n" + //
                                 "GROUP BY B.c_custkey\r\n" + //
                                 "ORDER BY Total_Spending DESC;");
            System.out.println("Running Query 3");
            executor.executeQuery("SELECT  O.o_orderpriority AS ORDER_PRIORITY, COUNT(L.l_linenumber) AS LINE_ITEM_NUMBER FROM orders O\r\n" + //
                                 "LEFT JOIN lineitem L ON O.o_orderkey = L.l_orderkey\r\n" + //
                                 "WHERE O.o_orderdate BETWEEN '1997-04-01' AND '2003-03-31'\r\n" + //
                                 "GROUP BY O.o_orderpriority\r\n" + //
                                 "ORDER BY O.o_orderpriority;");
            DatabaseConnection.closeConnection();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Query 1 Result



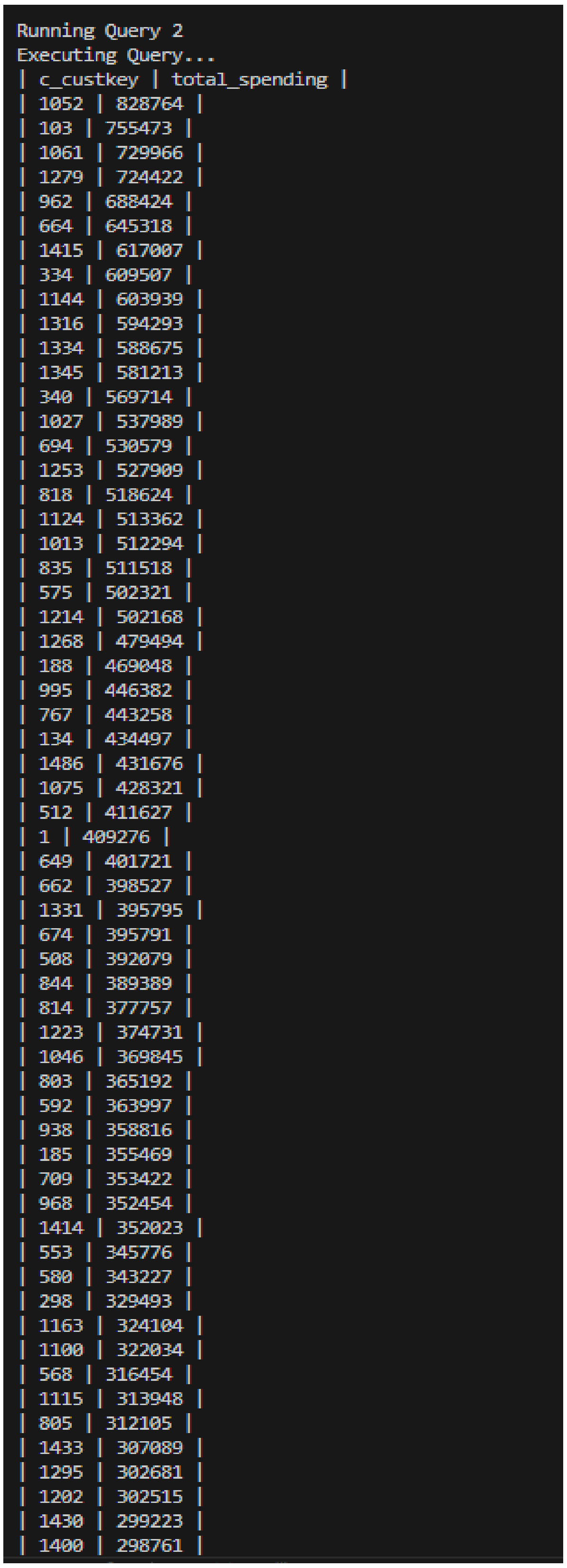
```
Running Query 1
Executing Query...
| c_custkey | o_orderkey | o_totalprice | o_orderdate |
| 293 | 7104 | 25969 | 2018-12-31 |
| 1258 | 32775 | 256571 | 2018-12-31 |
| 518 | 43299 | 32897 | 2018-12-29 |
| 632 | 27971 | 232296 | 2018-12-27 |
| 1163 | 26242 | 243898 | 2018-12-27 |
| 440 | 44551 | 23940 | 2018-12-27 |
| 430 | 32518 | 77682 | 2018-12-27 |
| 640 | 33412 | 41298 | 2018-12-26 |
| 443 | 43077 | 64153 | 2018-12-25 |
| 1454 | 28615 | 199643 | 2018-12-25 |
```

Query 3 Result



```
Connected to Redshift database.
Running Query 3
Executing Query...
| order_priority | line_item_number |
| 1-URGENT        | 1387 |
| 2-HIGH          | 1303 |
| 3-MEDIUM        | 1287 |
| 4-NOT SPECIFIED | 1530 |
| 5-LOW           | 1268 |
```

Query 2 Result



```
Running Query 2
Executing Query...
| c_custkey | total_spending |
| 1052 | 828764 |
| 103 | 755473 |
| 1061 | 729966 |
| 1279 | 724422 |
| 962 | 688424 |
| 664 | 645318 |
| 1415 | 617007 |
| 334 | 609507 |
| 1144 | 603939 |
| 1316 | 594293 |
| 1334 | 588675 |
| 1345 | 581213 |
| 340 | 569714 |
| 1027 | 537989 |
| 694 | 530579 |
| 1253 | 527909 |
| 818 | 518624 |
| 1124 | 513362 |
| 1013 | 512294 |
| 835 | 511518 |
| 575 | 502321 |
| 1214 | 502168 |
| 1268 | 479494 |
| 188 | 469048 |
| 995 | 446382 |
| 767 | 443258 |
| 134 | 434497 |
| 1486 | 431676 |
| 1075 | 428321 |
| 512 | 411627 |
| 1 | 409276 |
| 649 | 401721 |
| 662 | 398527 |
| 1331 | 395795 |
| 674 | 395791 |
| 508 | 392079 |
| 844 | 389389 |
| 814 | 377757 |
| 1223 | 374731 |
| 1046 | 369845 |
| 803 | 365192 |
| 592 | 363997 |
| 938 | 358816 |
| 185 | 355469 |
| 709 | 353422 |
| 968 | 352454 |
| 1414 | 352023 |
| 553 | 345776 |
| 580 | 343227 |
| 298 | 329493 |
| 1163 | 324104 |
| 1100 | 322034 |
| 568 | 316454 |
| 1115 | 313948 |
| 805 | 312105 |
| 1433 | 307089 |
| 1295 | 302681 |
| 1202 | 302515 |
| 1430 | 299223 |
| 1400 | 298761 |
```