Big Data Management - Assignment 3

AmazonRDS

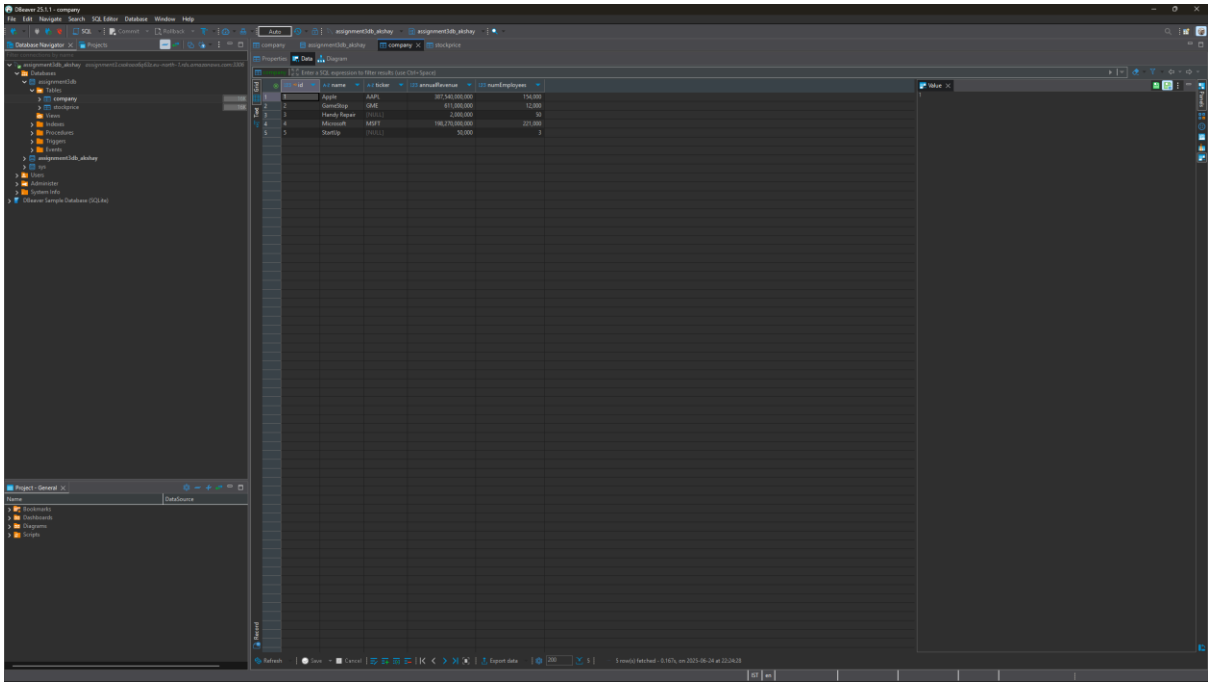
Akshay Kumar (G24AI1033)

Task 1 : Inserting the data

Code :

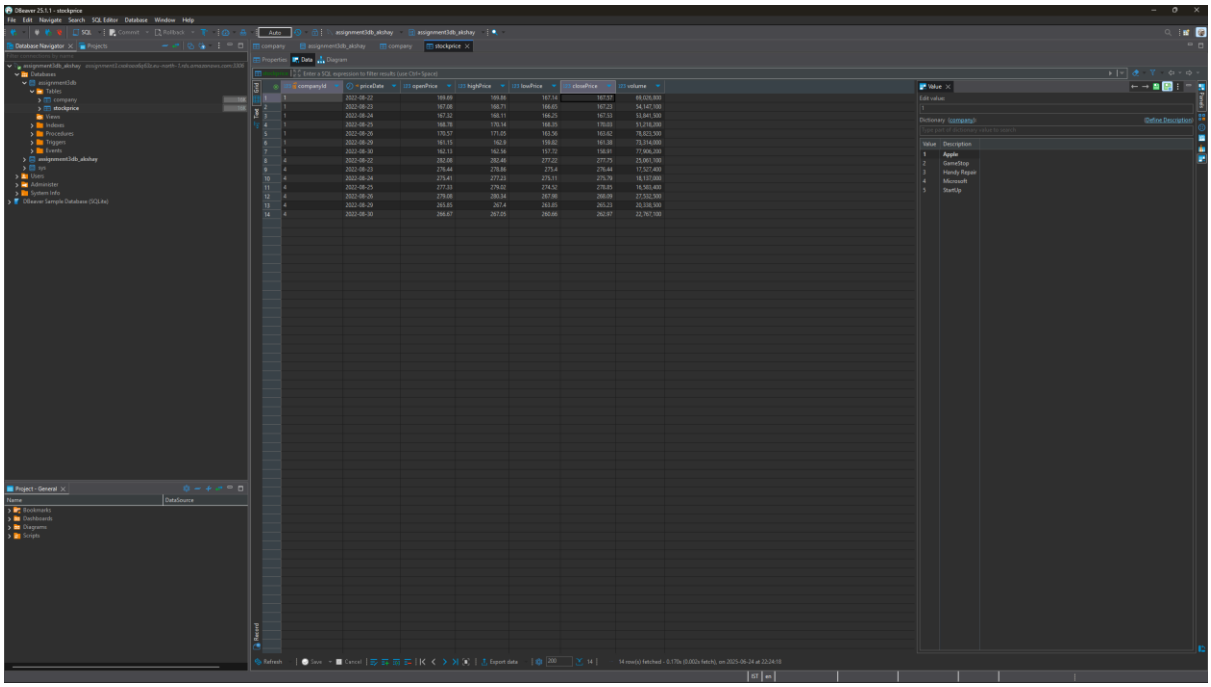
```
private void populateTables() throws SQLException {
    System.out.println(x:"Populating tables with initial data...");

    // Refactored to use a List of Objects, which is a significant structural change.
    List<Object[]> companyData = new ArrayList<>();
    companyData.add(new Object[]{1, "Apple", "AAPL", 387540000000.00, 154000});
    companyData.add(new Object[]{2, "GameStop", "GME", 611000000.00, 12000});
    companyData.add(new Object[]{3, "Handy Repair", null, 2000000.00, 50});
    companyData.add(new Object[]{4, "Microsoft", "MSFT", 198270000000.00, 221000});
    companyData.add(new Object[]{5, "StartUp", null, 50000.00, 3});

    String insertCompanySQL = "INSERT INTO company (id, name, ticker, annualRevenue, numEmployees) VALUES (?, ?, ?, ?, ?)";
    try (PreparedStatement stmt = dbConnection.prepareStatement(insertCompanySQL)) {
        for (Object[] row : companyData) {
            stmt.setInt(parameterIndex:1, (Integer) row[0]);
            stmt.setString(parameterIndex:2, (String) row[1]);
            stmt.setString(parameterIndex:3, (String) row[2]);
            if (row[3] != null) stmt.setDouble(parameterIndex:4, (Double) row[3]); else stmt.setNull(parameterIndex:4, java.sql.Types.DECIMAL);
            stmt.setInt(parameterIndex:5, (Integer) row[4]);
            stmt.addBatch();
        }
        stmt.executeBatch();
        System.out.println("  -> " + companyData.size() + " records inserted into 'company'.");
    }

    Object[][] stockData = {
            {1, "2022-08-15", 171.52, 173.39, 171.35, 173.19, 54091700L}, {1, "2022-08-16", 172.78, 173.71, 171.66, 173.03, 56377100L},
            {1, "2022-08-17", 172.77, 176.15, 172.57, 174.55, 79542000L}, {1, "2022-08-18", 173.75, 174.90, 173.12, 174.15, 62290100L},
            {1, "2022-08-19", 173.03, 173.74, 171.31, 171.52, 70211500L}, {1, "2022-08-22", 169.69, 169.86, 167.14, 167.57, 69026800L},
            {1, "2022-08-23", 167.08, 168.71, 166.65, 167.23, 54147100L}, {1, "2022-08-24", 167.32, 168.11, 166.25, 167.53, 53841500L},
            {1, "2022-08-25", 168.78, 170.14, 168.35, 170.03, 51218200L}, {1, "2022-08-26", 170.57, 171.05, 163.56, 163.62, 78823500L},
            {1, "2022-08-29", 161.15, 162.90, 159.82, 161.38, 73314000L}, {1, "2022-08-30", 162.13, 162.56, 157.72, 158.91, 77906200L},
            {2, "2022-08-15", 39.75, 40.39, 38.81, 39.68, 5243100L}, {2, "2022-08-16", 39.17, 45.53, 38.60, 42.19, 23602800L},
            {2, "2022-08-17", 42.18, 44.36, 40.41, 40.52, 9766400L}, {2, "2022-08-18", 39.27, 40.07, 37.34, 37.93, 8145400L},
            {2, "2022-08-19", 35.18, 37.19, 34.67, 36.49, 9525600L}, {2, "2022-08-22", 34.31, 36.20, 34.20, 34.50, 5798600L},
            {2, "2022-08-23", 34.70, 34.99, 33.45, 33.53, 4836300L}, {2, "2022-08-24", 34.00, 34.94, 32.44, 32.50, 5620300L},
            {2, "2022-08-25", 32.84, 32.89, 31.50, 31.96, 4726300L}, {2, "2022-08-26", 31.50, 32.38, 30.63, 30.94, 4289500L},
            {2, "2022-08-29", 30.48, 32.75, 30.38, 31.55, 4292700L}, {2, "2022-08-30", 31.62, 31.87, 29.42, 29.84, 5060200L},
            {4, "2022-08-15", 291.00, 294.18, 290.11, 293.47, 18085700L}, {4, "2022-08-16", 291.99, 294.04, 290.42, 292.71, 18102900L},
            {4, "2022-08-17", 289.74, 293.35, 289.47, 291.32, 18253400L}, {4, "2022-08-18", 290.19, 291.91, 289.08, 290.17, 17186200L},
            {4, "2022-08-19", 288.90, 289.25, 285.56, 286.15, 20557200L}, {4, "2022-08-22", 282.08, 282.46, 277.22, 277.75, 25061100L},
            {4, "2022-08-23", 276.44, 278.86, 275.40, 276.44, 17527400L}, {4, "2022-08-24", 275.41, 277.23, 275.11, 275.79, 18137000L},
            {4, "2022-08-25", 277.33, 279.02, 274.52, 278.85, 16583400L}, {4, "2022-08-26", 279.08, 280.34, 267.98, 268.09, 27532500L},
            {4, "2022-08-29", 265.85, 267.40, 263.85, 265.23, 20338500L}, {4, "2022-08-30", 266.67, 267.05, 260.66, 262.97, 22767100L}
    };

    String insertStockPriceSQL = "INSERT INTO stockprice (companyId, priceDate, openPrice, highPrice, lowPrice, closePrice, volume) VALUES (?, ?, ?, ?, ?, ?, ?)";
    try (PreparedStatement stmt = dbConnection.prepareStatement(insertStockPriceSQL)) {
        for (Object[] row : stockData) {
            stmt.setInt(parameterIndex:1, (Integer) row[0]);
            stmt.setDate(parameterIndex:2, Date.valueOf((String) row[1]));
            stmt.setDouble(parameterIndex:3, (Double) row[2]);
            stmt.setDouble(parameterIndex:4, (Double) row[3]);
            stmt.setDouble(parameterIndex:5, (Double) row[4]);
            stmt.setDouble(parameterIndex:6, (Double) row[5]);
            stmt.setLong(parameterIndex:7, (Long) row[6]);
            stmt.addBatch();
        }
        stmt.executeBatch();
        System.out.println("  -> " + stockData.length + " records inserted into 'stockprice'.");
    }
}
```

Screenshot of the data :



Task 2 : Deleting the data

```
private void deleteSpecifiedRecords() throws SQLException {
    System.out.println("Deleting records based on assignment criteria...");
    String sql = "DELETE FROM stockprice WHERE priceDate < '2022-08-20' OR companyId = 2";
    try (Statement stmt = dbConnection.createStatement()) {
        int rowsAffected = stmt.executeUpdate(sql);
        System.out.println("--> " + rowsAffected + " records deleted.");
    }
}
```



Query 1 :

```
private ResultSet executeQueryOne() throws SQLException {
    String sql = "SELECT name, annualRevenue, numEmployees "
               + "FROM company "
               + "WHERE numEmployees > 10000 OR annualRevenue < 1000000 "
               + "ORDER BY name ASC";
    return dbConnection.createStatement().executeQuery(sql);
}
```

Screenshot of Query 1 output :

```
[--- Query 1: Companies with >10k Employees or <$1M Revenue ---]
+-----------+----------------+-------------+
| name      | annualRevenue  | numEmployees |
+-----------+----------------+-------------+
| Apple     | 387540000000.00 | 154000      |
| GameStop  | 611000000.00    | 12000       |
| Microsoft | 198270000000.00 | 221000      |
| StartUp   | 50000.00        | 3           |
+-----------+----------------+-------------+
Total results: 4
```

Query 2 :

```
private ResultSet executeQueryTwo() throws SQLException {
    String sql = "SELECT c.name, c.ticker, MIN(s.lowPrice) AS lowestPrice, "
               + "MAX(s.highPrice) AS highestPrice, AVG(s.closePrice) AS avgClosePrice, "
               + "AVG(s.volume) AS avgVolume "
               + "FROM company c JOIN stockprice s ON c.id = s.companyId "
               + "WHERE s.priceDate BETWEEN '2022-08-22' AND '2022-08-26' "
               + "GROUP BY c.id, c.name, c.ticker "
               + "ORDER BY avgVolume DESC";
    return dbConnection.createStatement().executeQuery(sql);
}
```

Screenshot of Query 2 output :

```
[--- Query 2: Aggregate Stock Info for Aug 22-26 ---]
+-----------+--------+-------------+--------------+---------------+---------------+
| name      | ticker | lowestPrice | highestPrice | avgClosePrice | avgVolume     |
+-----------+--------+-------------+--------------+---------------+---------------+
| Apple     | AAPL   | 163.56      | 171.05       | 167.196000    | 61411420.0000 |
| Microsoft | MSFT   | 267.98      | 282.46       | 275.384000    | 20968280.0000 |
+-----------+--------+-------------+--------------+---------------+---------------+
Total results: 2
```

Query 3 :

```
private ResultSet executeQueryThree() throws SQLException {
    String sql = "SELECT c.name, c.ticker, s30.closePrice AS closingPrice_Aug30 "
               + "FROM company c "
               + "LEFT JOIN stockprice s30 ON c.id = s30.companyId AND s30.priceDate = '2022-08-30' "
               + "LEFT JOIN (SELECT companyId, AVG(closePrice) AS avgClose FROM stockprice WHERE priceDate BETWEEN '2022-08-15' AND '2022-08-19' GROUP BY companyId) AS avgWeek ON c.id = avgWeek.companyId "
               + "WHERE c.ticker IS NULL OR (s30.closePrice IS NOT NULL AND avgWeek.avgClose IS NOT NULL AND s30.closePrice >= (avgWeek.avgClose * 0.9))";
    return dbConnection.createStatement().executeQuery(sql);
}
```

Screenshot of Query 3 output :

```
[--- Query 3: Price Comparison for Aug 30 ---]
+--------------+--------+------------+
| name         | ticker | closePrice |
+--------------+--------+------------+
| Handy Repair | NULL   | NULL       |
| StartUp      | NULL   | NULL       |
+--------------+--------+------------+
Total results: 2
```

resultSetToString(): I have replace resultSetToString() method with a new, custom-built display system.

I have also integrated the resultSetMetaDataToString() into displayResultSet() by using metaData.getColumnName(i + 1)

```java
public static void displayResultSet(ResultSet rs) throws SQLException {
    if (rs == null) {
        System.out.println(x: "Query did not return a result set.");
        return;
    }

    ResultSetMetaData metaData = rs.getMetaData();
    int columnCount = metaData.getColumnCount();

    List<String[]> tableData = new ArrayList<>();
    String[] header = new String[columnCount];
    for (int i = 0; i < columnCount; i++) {
        header[i] = metaData.getColumnName(i + 1);
    }
    tableData.add(header);

    while (rs.next()) {
        String[] row = new String[columnCount];
        for (int i = 0; i < columnCount; i++) {
            Object value = rs.getObject(i + 1);
            row[i] = (value == null) ? "NULL" : value.toString();
        }
        tableData.add(row);
    }

    int[] columnWidths = new int[columnCount];
    for (String[] row : tableData) {
        for (int i = 0; i < columnCount; i++) {
            if (row[i].length() > columnWidths[i]) {
                columnWidths[i] = row[i].length();
            }
        }
    }

    StringBuilder tableBuilder = new StringBuilder();

    printTableBorder(tableBuilder, columnWidths);

    printTableRow(tableBuilder, tableData.get(index: 0), columnWidths);

    printTableBorder(tableBuilder, columnWidths);


    int rowCount = 0;
    for (int i = 1; i < tableData.size(); i++) {
        printTableRow(tableBuilder, tableData.get(i), columnWidths);
        rowCount++;
    }

    if (rowCount == 0) {
        System.out.println("| " + String.format("%-" + (sum(columnWidths) + 3 * (columnCount-1) -1) + "s", ...args: "Query returned no results.") + " |");
    }

    printTableBorder(tableBuilder, columnWidths);

    System.out.print(tableBuilder.toString());
    if (rowCount > 0){
        System.out.println("Total results: " + rowCount);
    }
}
```

```
[--- Verifying data insertion in 'company' table ---]
+----+-------------+--------+-----------------+-------------+
| id | name        | ticker | annualRevenue   | numEmployees |
+----+-------------+--------+-----------------+-------------+
| 1  | Apple       | AAPL   | 387540000000.00 | 154000      |
| 2  | GameStop    | GME    | 611000000.00    | 12000       |
| 3  | Handy Repair | NULL   | 2000000.00      | 50          |
| 4  | Microsoft   | MSFT   | 198270000000.00 | 221000      |
| 5  | StartUp     | NULL   | 50000.00        | 3           |
+----+-------------+--------+-----------------+-------------+
Total results: 5

[--- Verifying data insertion in 'stockprice' table ---]
+-----------+------------+-----------+-----------+----------+------------+----------+
| companyId | priceDate  | openPrice | highPrice | lowPrice | closePrice | volume   |
+-----------+------------+-----------+-----------+----------+------------+----------+
| 1         | 2022-08-15 | 171.52    | 173.39    | 171.35   | 173.19     | 54091700 |
| 1         | 2022-08-16 | 172.78    | 173.71    | 171.66   | 173.03     | 56377100 |
| 1         | 2022-08-17 | 172.77    | 176.15    | 172.57   | 174.55     | 79542000 |
| 1         | 2022-08-18 | 173.75    | 174.90    | 173.12   | 174.15     | 62290100 |
| 1         | 2022-08-19 | 173.03    | 173.74    | 171.31   | 171.52     | 70211500 |
| 1         | 2022-08-22 | 169.69    | 169.86    | 167.14   | 167.57     | 69026000 |
| 1         | 2022-08-23 | 167.08    | 168.71    | 166.65   | 167.23     | 54147100 |
| 1         | 2022-08-24 | 167.32    | 168.11    | 166.25   | 167.53     | 53841500 |
| 1         | 2022-08-25 | 168.78    | 170.14    | 168.35   | 170.03     | 51218200 |
| 1         | 2022-08-26 | 170.57    | 171.05    | 163.56   | 163.62     | 78823500 |
| 1         | 2022-08-29 | 161.15    | 162.90    | 159.82   | 161.38     | 73314000 |
| 1         | 2022-08-30 | 162.13    | 162.56    | 157.72   | 158.91     | 77906200 |
| 2         | 2022-08-15 | 39.75     | 40.39     | 38.81    | 39.68      | 5243100  |
| 2         | 2022-08-16 | 39.17     | 45.53     | 38.60    | 42.19      | 23602800 |
| 2         | 2022-08-17 | 42.18     | 44.36     | 40.41    | 40.52      | 9766400  |
| 2         | 2022-08-18 | 39.27     | 40.07     | 37.34    | 37.93      | 8145400  |
| 2         | 2022-08-19 | 35.18     | 37.19     | 34.67    | 36.49      | 9525600  |
| 2         | 2022-08-22 | 34.31     | 36.20     | 34.20    | 34.50      | 5798600  |
| 2         | 2022-08-23 | 34.70     | 34.99     | 33.45    | 33.53      | 4836300  |
| 2         | 2022-08-24 | 34.00     | 34.94     | 32.44    | 32.50      | 5620300  |
| 2         | 2022-08-25 | 32.84     | 32.89     | 31.50    | 31.96      | 4726300  |
| 2         | 2022-08-26 | 31.50     | 32.38     | 30.63    | 30.94      | 4289500  |
| 2         | 2022-08-29 | 30.48     | 32.75     | 30.38    | 31.55      | 4292700  |
| 2         | 2022-08-30 | 31.62     | 31.87     | 29.42    | 29.84      | 5060200  |
| 4         | 2022-08-15 | 291.00    | 294.18    | 290.11   | 293.47     | 18085700 |
| 4         | 2022-08-16 | 291.99    | 294.04    | 290.42   | 292.71     | 18102900 |
| 4         | 2022-08-17 | 289.74    | 293.35    | 289.47   | 291.32     | 18253400 |
| 4         | 2022-08-18 | 290.19    | 291.91    | 289.08   | 290.17     | 17186200 |
| 4         | 2022-08-19 | 288.90    | 289.25    | 285.56   | 286.15     | 20557200 |
| 4         | 2022-08-22 | 282.08    | 282.46    | 277.22   | 277.75     | 25061100 |
| 4         | 2022-08-23 | 276.44    | 278.86    | 275.40   | 276.44     | 17527400 |
| 4         | 2022-08-24 | 275.41    | 277.23    | 275.11   | 275.79     | 18137000 |
| 4         | 2022-08-25 | 277.33    | 279.02    | 274.52   | 278.85     | 16583400 |
| 4         | 2022-08-26 | 279.08    | 280.34    | 267.98   | 268.09     | 27532500 |
| 4         | 2022-08-29 | 265.85    | 267.40    | 263.85   | 265.23     | 20338500 |
| 4         | 2022-08-30 | 266.67    | 267.05    | 260.66   | 262.97     | 22767100 |
+-----------+------------+-----------+-----------+----------+------------+----------+
Total results: 36
```

```
[--- Query 1: Companies with >10k Employees or <$1M Revenue ---]
+-----------+----------------+--------------+
| name      | annualRevenue  | numEmployees |
+-----------+----------------+--------------+
| Apple     | 387540000000.00 | 154000       |
| GameStop  | 611000000.00   | 12000        |
| Microsoft | 198270000000.00 | 221000       |
| StartUp   | 50000.00       | 3            |
+-----------+----------------+--------------+
Total results: 4

[--- Query 2: Aggregate Stock Info for Aug 22-26 ---]
+-----------+--------+-------------+--------------+---------------+---------------+
| name      | ticker | lowestPrice | highestPrice | avgClosePrice | avgVolume     |
+-----------+--------+-------------+--------------+---------------+---------------+
| Apple     | AAPL   | 163.56      | 171.05       | 167.196000    | 61411420.0000 |
| Microsoft | MSFT   | 267.98      | 282.46       | 275.384000    | 20968280.0000 |
+-----------+--------+-------------+--------------+---------------+---------------+
Total results: 2

[--- Query 3: Price Comparison for Aug 30 ---]
+-------------+--------+------------+
| name        | ticker | closePrice |
+-------------+--------+------------+
| Handy Repair | NULL   | NULL       |
| StartUp     | NULL   | NULL       |
+-------------+--------+------------+
Total results: 2

Database connection closed.
```