Assignment-2

40 points

Submission Instructions:

Submit a PDF with code listing, and screenshots showing outputs of insert(), delete(), and the queries. Screenshots should be uniquely distinguishable for each submission. Be careful of plagiarism from online sources/peers.

Extend the Amazon RDS connector code in java to do the following in each of the respective functions.

1. connect() : to connect to the RDS database
2. drop() : to drop the table from the database
3. create() : creates the following table in the database: [10]

   * Creates the table in the database.
   * Table name: company
   * Fields:
   * - id - integer, must be primary key
   * - name - variable character field up to size 50
   * - ticker - character field always of size 10
   * - annualRevenue - must hold up to 999,999,999,999.99 exactly
   * - numEmployees - integer
   *
   * Table name: stockprice
   * Fields:
   * - companyId - integer
   * - priceDate - date of stock price
   * - openPrice - opening price must hold up to 99999999.99
   * - highPrice - high price must hold up to 99999999.99
   * - lowPrice - low price must hold up to 99999999.99
   * - closePrice - closing price must hold up to 99999999.99
   * - volume - number of shares traded, integer
   * - primary key must be companyId and priceDate
   * - add an appropriate foreign key

4. insert() : Inserts test records in the database: [10]

   * Data for company table:
   1, 'Apple', 'AAPL', 387540000000.00 , 154000
   2, 'GameStop', 'GME', 611000000.00, 12000
   3, 'Handy Repair', null, 2000000, 50
   4, 'Microsoft', 'MSFT', '198270000000.00' , 221000
   5, 'StartUp', null, 50000, 3

```
*
* Data for stockprice table:
        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, 69026800
        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
```

5. delete() [5]

   Delete all stock price records where the date is before 2022-08-20 or the company is GameStop

6. queryOne(): //TODO for returning ResultSet [5]

   Query returns company info (name, revenue, employees) that have more than 10000 employees or annual revenue less that 1 million dollars. Order by company name ascending.

7. queryTwo() ://TODO for returning ResultSet [5]

   Query returns the company name and ticker and calculates the lowest price, highest price, average closing price, and average volume in the week of August 22nd to 26th inclusive. Order by average volume descending.

8. queryThree() : //TODO for returning ResultSet [5]

   Query returns a list of all companies that displays their name, ticker, and closing stock price on August 30, 2022 (if exists). Only show companies where their closing stock price on August 30, 2022 is no more than 10% below the closing average for the week of August 15th to 19th inclusive. That is, if closing price is currently 100, the average closing price must be <= 110. Companies without a stock ticker should always be shown in the list. Order by company name ascending.

9. resultSetToString(): converts a ResultSet obtained front he queries to String (Given)
10. resultSetMetaDataToString() : converts resultSetMetaData to String or the String of the metadata (Schema) (Given)

Starter Code:

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;



public class SQLonRDS {

   private Connection con;



   private String url = "iitjdb.cpmqoqquutp0.us-east-1.rds.amazonaws.com:3306";

   private String uid = "admin";

   private String pw = "iitj@12345";



   public static void main(String[] args) {
    SQLonRDS q = new SQLonRDS();
    try {
        q.connect();
        //q.create(); // TODO AS PER THE QUESTION
        //q.drop(); // TODO AS PER THE QUESTION USING ResultSet type as return
        //q.insert(); // TODO AS PER THE QUESTION
          //q.queryOne(); // TODO AS PER THE QUESTION
          //q.queryTwo(); // TODO AS PER THE QUESTION
          //q.queryThree(); // TODO AS PER THE QUESTION



        q.close();
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
    }
}
public void connect() throws SQLException, ClassNotFoundException {
    // Load MySQL JDBC Driver
    Class.forName("com.mysql.cj.jdbc.Driver");
    String jdbcUrl = "jdbc:mysql://" + url + "/mydb?user=" + uid + "&password=" + pw;
    System.out.println("Connecting to database");
    con = DriverManager.getConnection(jdbcUrl);
    System.out.println("Connection Successful");
}
```

```
   public void close() throws SQLException {

       if (con != null) {

           con.close();

       }


public static String resultSetToString(ResultSet rst, int maxrows) throws SQLException
   {
       StringBuffer buf = new StringBuffer(5000);
       int rowCount = 0;
                   if (rst == null)
                           return "ERROR: No ResultSet";
       ResultSetMetaData meta = rst.getMetaData();
       buf.append("Total columns: " + meta.getColumnCount());
       buf.append('\n');
       if (meta.getColumnCount() > 0)
           buf.append(meta.getColumnName(1));
       for (int j = 2; j <= meta.getColumnCount(); j++)
           buf.append(", " + meta.getColumnName(j));
       buf.append('\n');

       while (rst.next())
       {
           if (rowCount < maxrows)
           {
               for (int j = 0; j < meta.getColumnCount(); j++)
               {
                       Object obj = rst.getObject(j + 1);

                       buf.append(obj);
                   if (j != meta.getColumnCount() - 1)
                       buf.append(", ");
               }
               buf.append('\n');
           }
           rowCount++;
       }
       buf.append("Total results: " + rowCount);
       return buf.toString();
   }
  public static String resultSetMetaDataToString(ResultSetMetaData meta) throws SQLException
   {
              StringBuffer buf = new StringBuffer(5000);
              buf.append(meta.getColumnName(1)+" ("+meta.getColumnLabel(1)+",
"+meta.getColumnType(1)+"-"+meta.getColumnTypeName(1)+", "+meta.getColumnDisplaySize(1)+",
"+meta.getPrecision(1)+", "+meta.getScale(1)+")");
```

```
                for (int j = 2; j <= meta.getColumnCount(); j++)
                    buf.append(", "+meta.getColumnName(j)+" ("+meta.getColumnLabel(j)+",
"+meta.getColumnType(j)+"-"+meta.getColumnTypeName(j)+", "+meta.getColumnDisplaySize(j)+",
"+meta.getPrecision(j)+", "+meta.getScale(j)+")");
                return buf.toString();
   }

    }

}
```