

Big data Management

Assignment 3

Jeyadev L

G23AI2071

## RDS Setup

A screenshot of a computer

Description automatically generated

## Connecting to RDS VIA JAVA

A screenshot of a computer

Description automatically generated

## Connecting to RDS VIA MYSQL Workbench

A screenshot of a computer

Description automatically generated

Creating Table and Inserting data

A screenshot of a computer screen

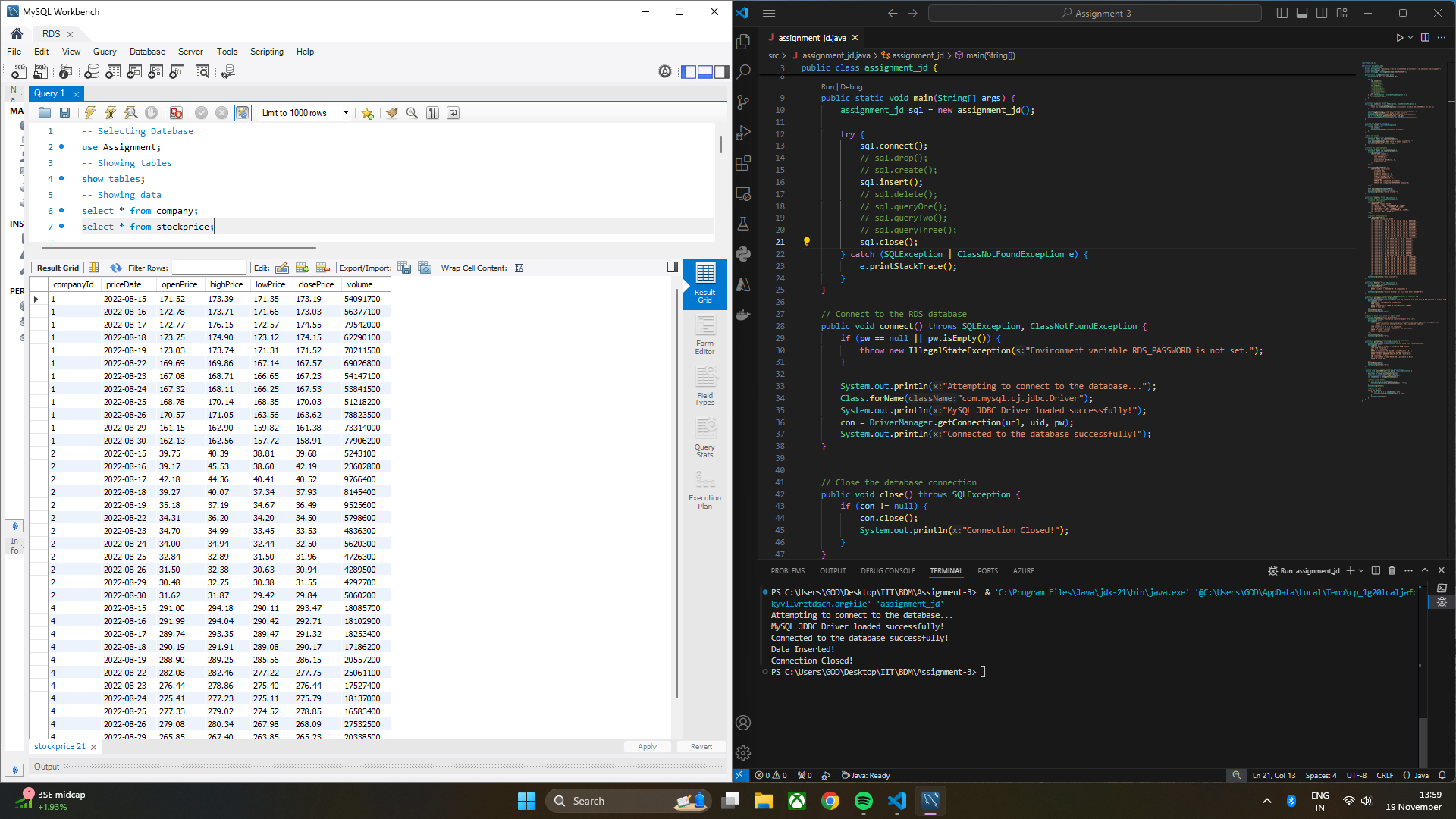
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated



Deleting Specific Data

A screenshot of a computer screen

Description automatically generated

## Query 1

A screenshot of a computer screen

Description automatically generated

## Query 2

A screenshot of a computer screen

Description automatically generated

## Query 3

A screenshot of a computer screen

Description automatically generated

## Helper Class

A screenshot of a computer program

Description automatically generated

## Code :-

*import java.sql.\*;*

*public class assignment\_jd {*

*private Connection con;*

*private String url = System.getenv("RDS\_URL");*

*private String uid = System.getenv("RDS\_USER");*

*private String pw = System.getenv("RDS\_PASSWORD");*

*public static void main(String[] args) {*

*assignment\_jd sql = new assignment\_jd();*

*try {*

*sql.connect();*

*sql.drop();*

*sql.create();*

*sql.insert();*

*sql.delete();*

*sql.queryOne();*

*sql.queryTwo();*

*sql.queryThree();*

*sql.close();*

*} catch (SQLException | ClassNotFoundException e) {*

*e.printStackTrace();*

*} }*

*// Connect to the RDS database*

*public void connect() throws SQLException, ClassNotFoundException {*

*if (pw == null || pw.isEmpty()) {*

*throw new IllegalStateException("Environment variable RDS\_PASSWORD is not set.");}*

*System.out.println("Attempting to connect to the database...");*

*Class.forName("com.mysql.cj.jdbc.Driver");*

*System.out.println("MySQL JDBC Driver loaded successfully!");*

*con = DriverManager.getConnection(url, uid, pw);*

*System.out.println("Connected to the database successfully!");}*

*// Close the database connection*

*public void close() throws SQLException {*

*if (con != null) {*

*con.close();*

*System.out.println("Connection Closed!");}}*

*// Drop the tables*

*public void drop() throws SQLException {*

*Statement stmt = con.createStatement();*

*stmt.executeUpdate("DROP TABLE IF EXISTS stockprice;");*

*stmt.executeUpdate("DROP TABLE IF EXISTS company;");*

*System.out.println("Tables Dropped!"); }*

*// Create the required tables*

*public void create() throws SQLException {*

*Statement stmt = con.createStatement();*

*String companyTable = """*

*CREATE TABLE company (*

*id INT PRIMARY KEY,*

*name VARCHAR(50),*

*ticker CHAR(10),*

*annualRevenue DECIMAL(15, 2),*

*numEmployees INT);""";*

*String stockPriceTable = """*

*CREATE TABLE stockprice (*

*companyId INT,*

*priceDate DATE,*

*openPrice DECIMAL(10, 2),*

*highPrice DECIMAL(10, 2),*

*lowPrice DECIMAL(10, 2),*

*closePrice DECIMAL(10, 2),*

*volume INT,*

*PRIMARY KEY (companyId, priceDate),*

*FOREIGN KEY (companyId) REFERENCES company(id));""";*

*stmt.executeUpdate(companyTable);*

*stmt.executeUpdate(stockPriceTable);*

*System.out.println("Tables Created!"); }*

*// Insert data into tables*

*public void insert() throws SQLException {*

*Statement stmt = con.createStatement();*

*// Insert into company*

*stmt.executeUpdate("""*

*INSERT INTO company VALUES*

*(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);""");*

*// Insert into stockprice*

*stmt.executeUpdate("""*

*INSERT INTO stockprice VALUES*

*(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); """);*

*System.out.println("Data Inserted!");}*

*// Delete specific rows*

*public void delete() throws SQLException {*

*Statement stmt = con.createStatement();*

*stmt.executeUpdate("""*

*DELETE FROM stockprice WHERE priceDate < '2022-08-20' OR companyId = 2; """);*

*System.out.println("Records Deleted! for Price date befor 2022-08-20"); }*

*// Query 1: Companies with more than 10,000 employees or revenue < $1M*

*public void queryOne() throws SQLException {*

*System.out.println("Executing Query 1 to get Companies with more than 10,000 employee or revenue <$1M \n");*

*String query = """*

*SELECT name, ROUND(annualRevenue,2), numEmployees FROM company*

*WHERE numEmployees > 10000 OR annualRevenue < 1000000*

*ORDER BY name ASC; """;*

*executeQuery(query);*

*System.out.println("\n");}*

*// Query 2: Stock price stats for August 22-26*

*public void queryTwo() throws SQLException {*

*System.out.println("Stock price stats for August 22-26 \n");*

*String query = """*

*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.name, c.ticker ORDER BY avgVolume DESC; """;*

*executeQuery(query);*

*System.out.println("\n"); }*

*// Query 3: Companies with closing stock price constraints*

*public void queryThree() throws SQLException {*

*System.out.println("Companies with closing stock price constraints \n");*

*String query = """*

*SELECT c.name, c.ticker, s.closePrice FROM company c LEFT JOIN stockprice s*

*ON c.id = s.companyId WHERE (s.closePrice IS NULL OR s.closePrice >= 0.9 \* (*

*SELECT COALESCE(AVG(closePrice), 0) FROM stockprice WHERE priceDate BETWEEN '2022-08-15' AND '2022-08-19' AND companyId = c.id)) AND (s.priceDate = '2022-08-30' OR s.priceDate IS NULL) ORDER BY c.name ASC; """;*

*executeQuery(query);*

*System.out.println("\n");}*

*// Helper method to execute and print query results*

*private void executeQuery(String query) throws SQLException {*

*Statement stmt = con.createStatement();*

*ResultSet rs = stmt.executeQuery(query);*

*ResultSetMetaData meta = rs.getMetaData();*

*int columnCount = meta.getColumnCount();*

*// Print column headers*

*for (int i = 1; i <= columnCount; i++) {*

*System.out.print(meta.getColumnName(i) + "\t");*

*}*

*System.out.println();*

*// Print row data*

*while (rs.next()) {*

*for (int i = 1; i <= columnCount; i++) {*

*System.out.print(rs.getString(i) + "\t");*

*}*

*System.out.println();*

*}*

*}*

*}*