SQL Solution Section

Data Extraction and Cleaning

1: Create table coffee_sales

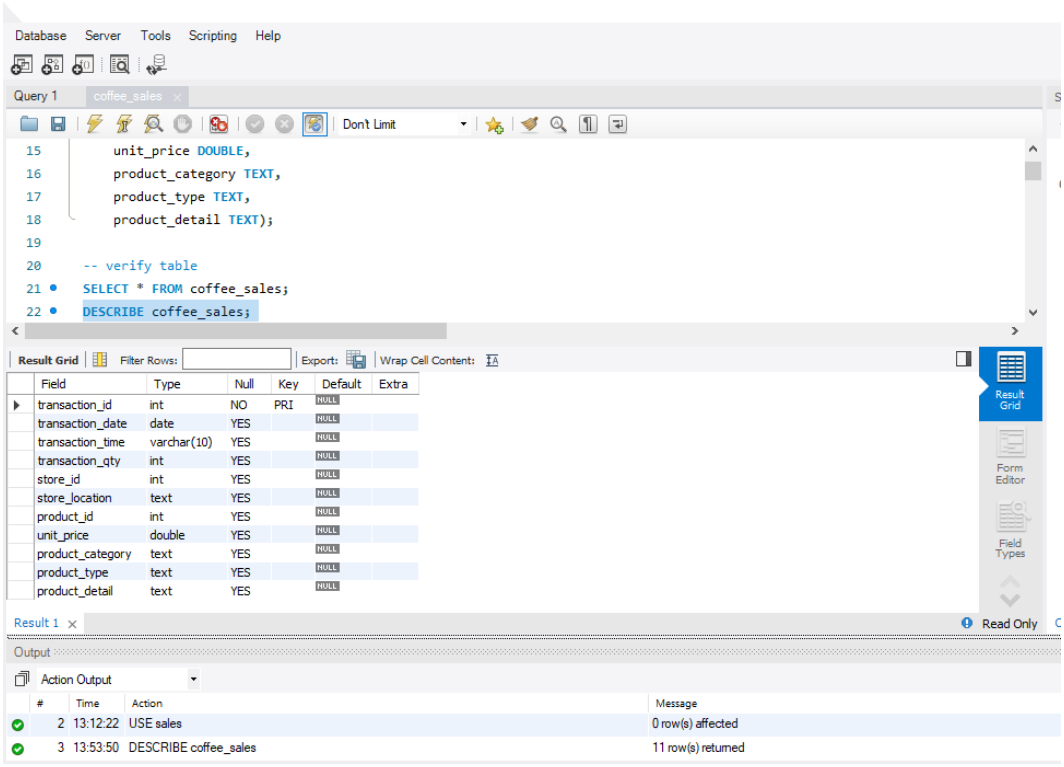
SQL:

```
CREATE TABLE coffee_sales
        (transaction_id INT PRIMARY KEY,
        transaction_date DATE,
        transaction_time VARCHAR(10),
        transaction_qty INT,
        store_id INT,
        store_location TEXT,
        product_id INT,
        unit_price DOUBLE,
        product_category TEXT,
        product_type TEXT,
        product_detail TEXT);
```

- **Verify Table**

SQL:

```
DESCRIBE coffee_sales;
```



2: Import the dataset into coffee_sales table

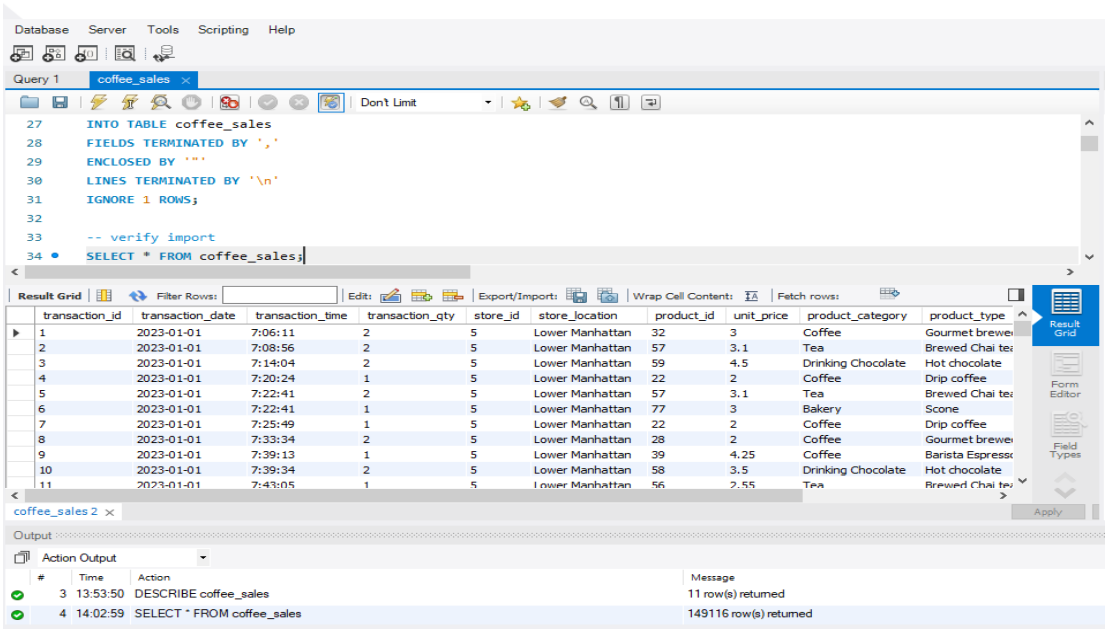
SQL

```
LOAD DATA LOCAL INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\Coffee_shop_sales.csv'

INTO TABLE coffee_sales

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS;
```

- **Verify Import**

SQL

```
SELECT * FROM coffee_sales;
```



3: Create a Worksheet Table (to protect the raw file) and insert dataset

SQL

```
CREATE TABLE coffee_worksheet LIKE coffee_sales;
```

- **verify table**

SQL

```
DESCRIBE coffee_worksheet;
```

4: Insert a copy of dataset into coffee worksheet table

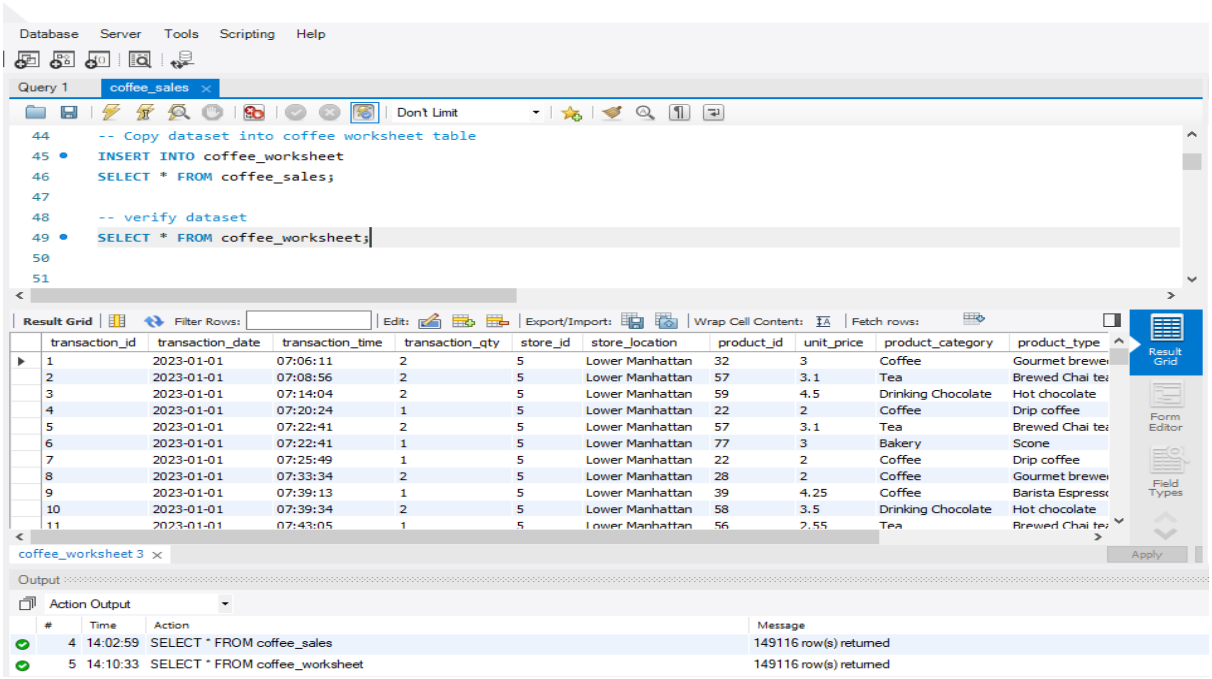
SQL

```
INSERT INTO coffee_worksheet

SELECT * FROM coffee_sales;
```

- **Verify dataset**

SQL

```
SELECT * FROM coffee_worksheet;
```



5: Data Cleaning

Update transaction_time to time standard format and Alter the datatype

- **Step A: Update transaction time**

SQL

```
UPDATE coffee_worksheet
SET
transaction_time = STR_TO_DATE(transaction_time, '%H:%i:%s');
```

- **Step B: Alter Datatype**

SQL

```
ALTER TABLE coffee_worksheet
MODIFY transaction_time TIME;
```

- **verify changes**

SQL

```
DESCRIBE COFFEE_WORKSHEET;

SELECT * FROM COFFEE_WORKSHEET;
```

EXPLORATORY DATA ANALYSIS ON BUSINESS QUESTIONS

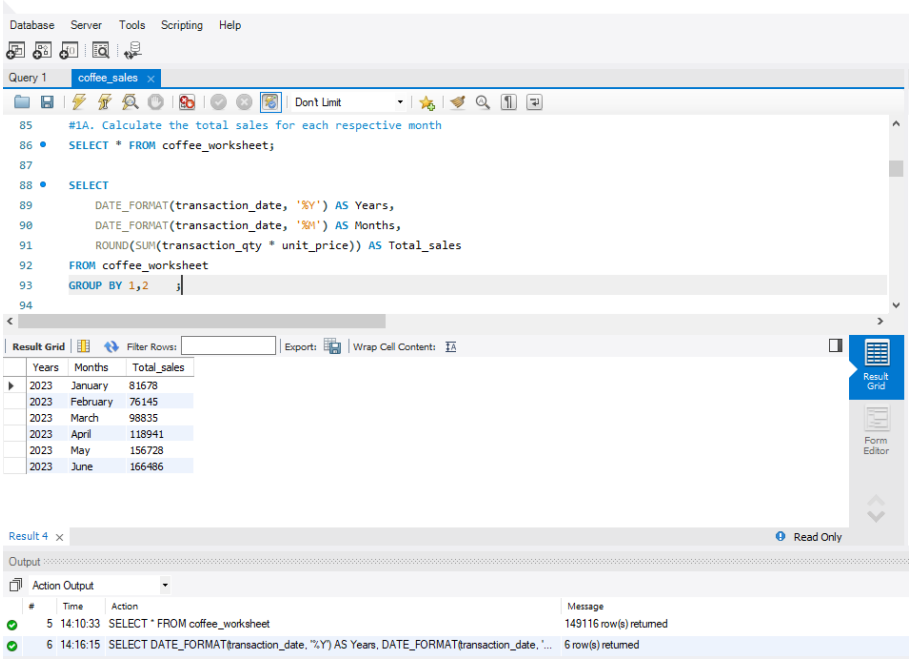
SECTION A: KPI'S REQUIREMENTS

1. Total Sales Analysis

- **1A. total sales for each respective month**

SQL

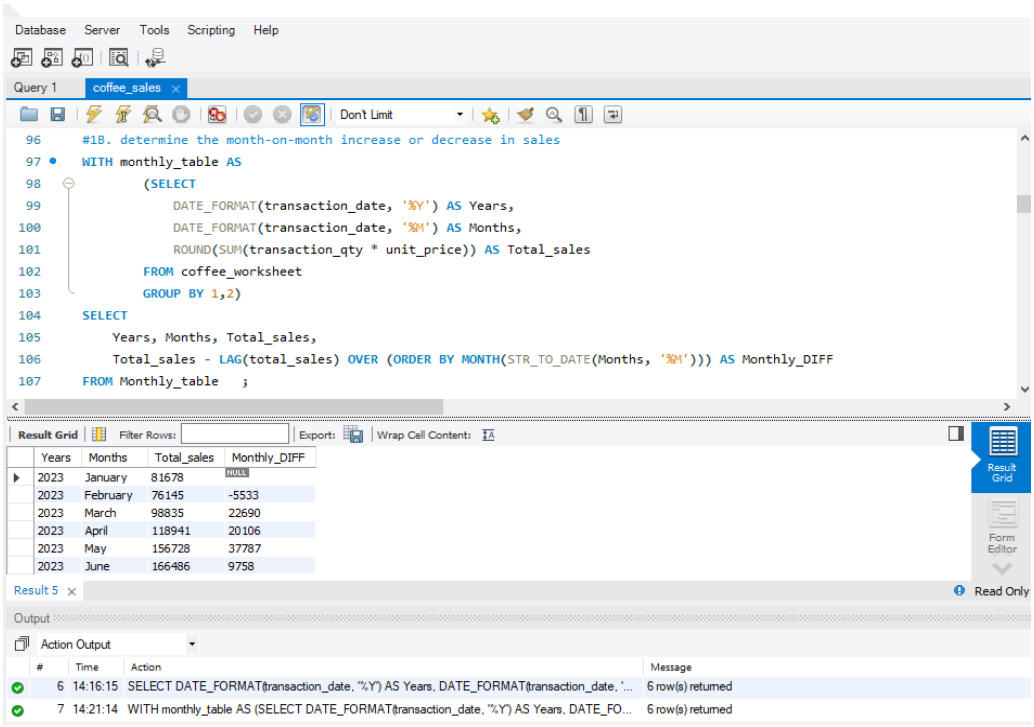
```
SELECT
  DATE_FORMAT(transaction_date, '%Y') AS Years,
  DATE_FORMAT(transaction_date, '%M') AS Months,
  ROUND(SUM(transaction_qty * unit_price)) AS Total_sales
FROM coffee_worksheet
GROUP BY 1,2    ;
```



- **1B. month-on-month increase or decrease in sales**

SQL

```
WITH monthly_table AS
 (SELECT
 DATE_FORMAT(transaction_date, '%Y') AS Years,
 DATE_FORMAT(transaction_date, '%M') AS Months,
 ROUND(SUM(transaction_qty * unit_price)) AS Total_sales
 FROM coffee_worksheet
 GROUP BY 1,2)
SELECT
 Years, Months, Total_sales,
 Total_sales - LAG(total_sales) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M'))) AS Monthly_DIFF
FROM Monthly_table  ;
```



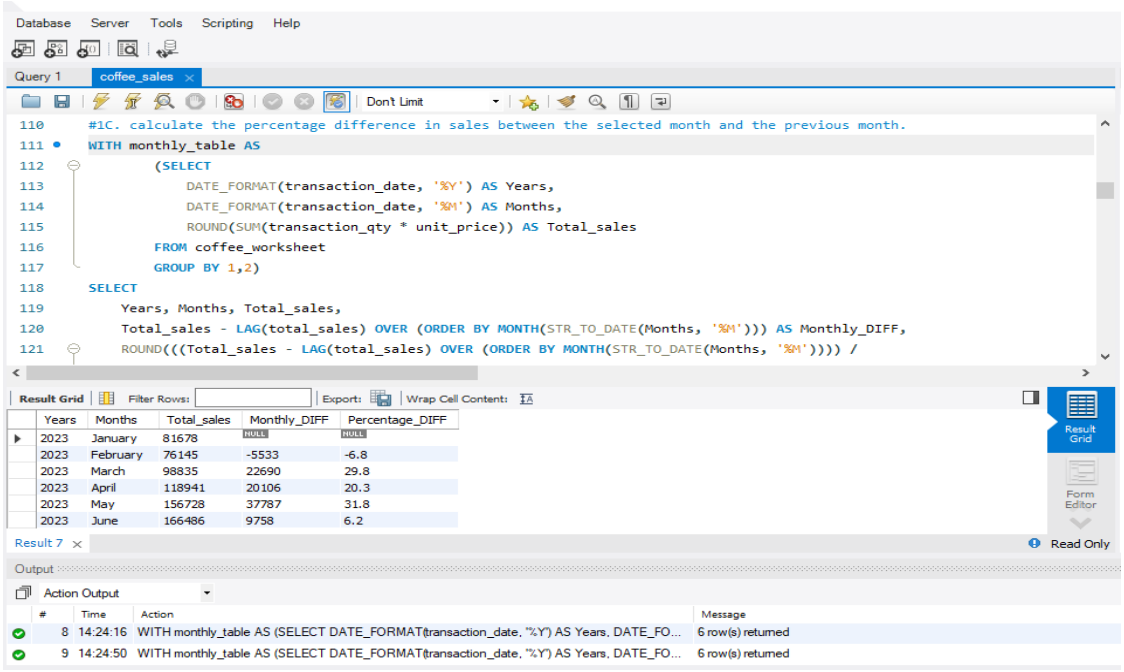
- **1C. percentage difference in sales between the selected month and the previous month**

SQL

```
WITH monthly_table AS
 (SELECT
 DATE_FORMAT(transaction_date, '%Y') AS Years,
 DATE_FORMAT(transaction_date, '%M') AS Months,
 ROUND(SUM(transaction_qty * unit_price)) AS Total_sales
 FROM coffee_worksheet
 GROUP BY 1,2)
SELECT
 Years, Months, Total_sales,
 Total_sales - LAG(total_sales) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M'))) AS Monthly_DIFF,
 ROUND(((Total_sales - LAG(total_sales) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M')))) /
```

LAG(total_sales) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M')))) * 100, 1) AS Percentage_DIFF

FROM Monthly_table ;



2. Total Order Analysis

- **2A. total number of orders for each respective month**

SQL

SELECT

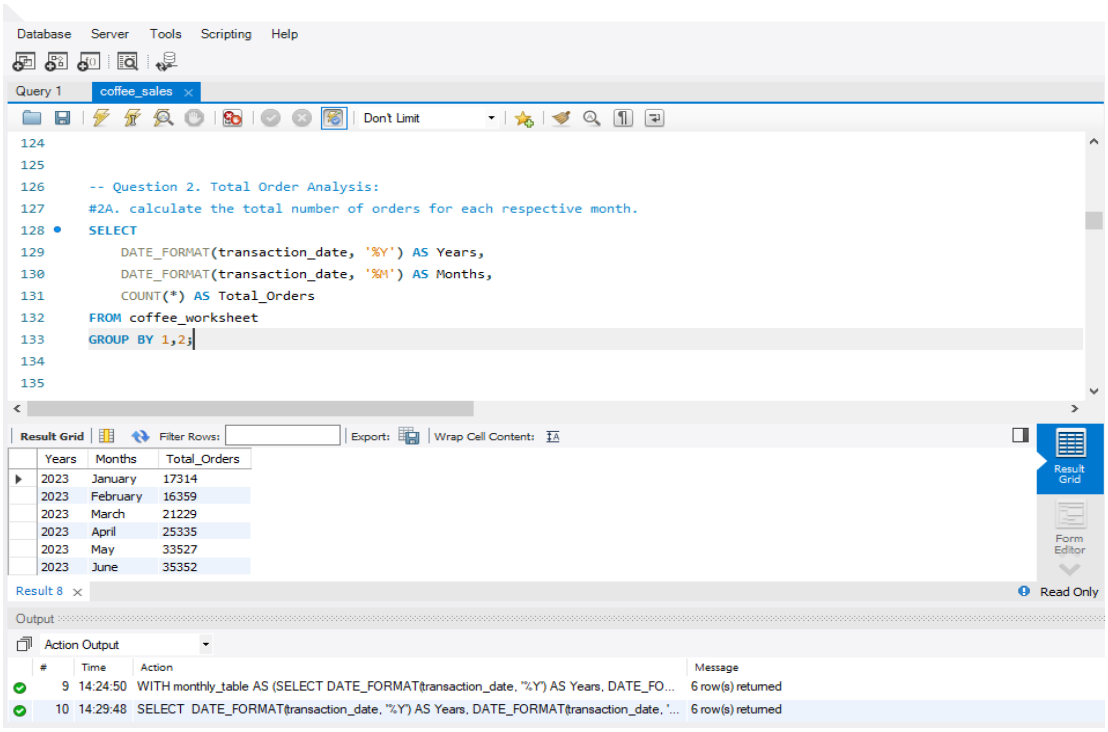
DATE_FORMAT(transaction_date, '%Y') AS Years,

DATE_FORMAT(transaction_date, '%M') AS Months,

COUNT(*) AS Total_Orders

FROM coffee_worksheet

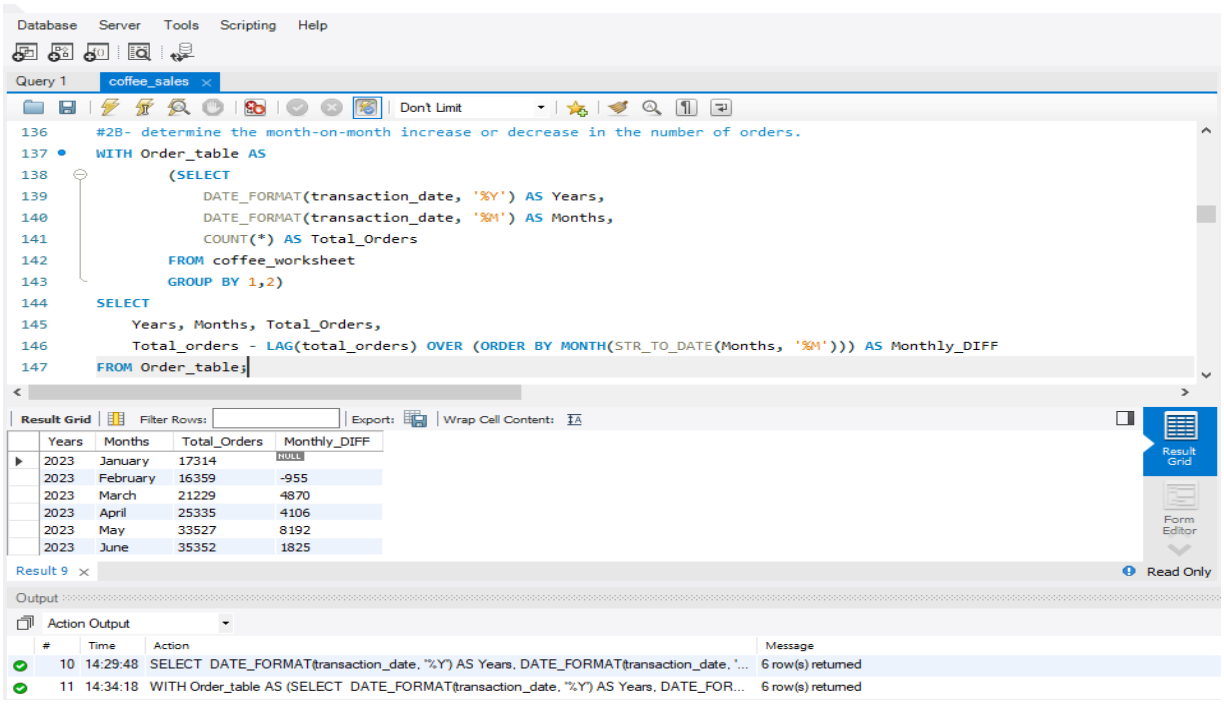
GROUP BY 1,2;



- **2B. month-on-month increase or decrease in the number of orders.**

SQL

```
WITH Order_table AS
(SELECT
DATE_FORMAT(transaction_date, '%Y') AS Years,
DATE_FORMAT(transaction_date, '%M') AS Months,
COUNT(*) AS Total_Orders
FROM coffee_worksheet
GROUP BY 1,2)
SELECT
Years, Months, Total_Orders,
Total_orders - LAG(total_orders) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M'))) AS Monthly_DIFF
FROM Order_table;
```



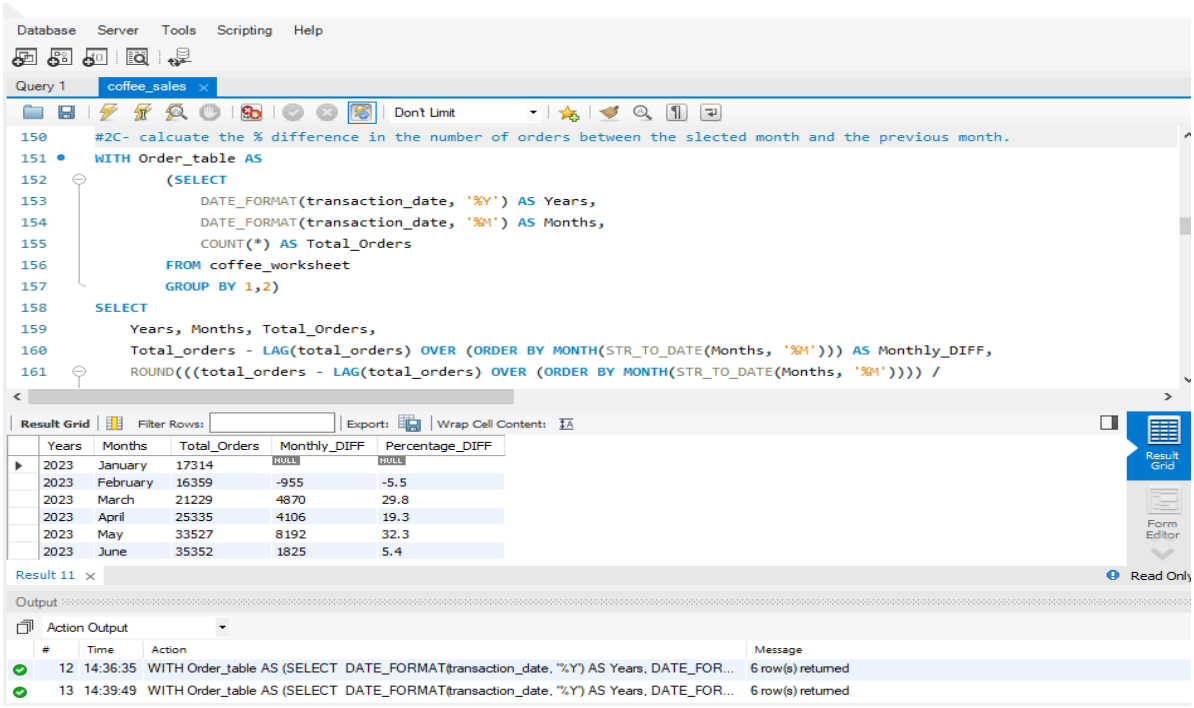
- **2C. percentage difference on the slected month and the previous month.**

SQL

```
WITH Order_table AS
 (SELECT
 DATE_FORMAT(transaction_date, '%Y') AS Years,
 DATE_FORMAT(transaction_date, '%M') AS Months,
 COUNT(*) AS Total_Orders
FROM coffee_worksheet
GROUP BY 1,2)
SELECT
 Years, Months, Total_Orders,
 Total_orders - LAG(total_orders) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M'))) AS Monthly_DIFF,
 ROUND(((total_orders - LAG(total_orders) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M')))) /
```

LAG(total_orders) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M')))) * 100, 1) AS Percentage_DIFF

FROM Order_table;



3: Total Quantity Sold Analysis

- **3A. total quantity sold for each respective month.**

SQL

SELECT

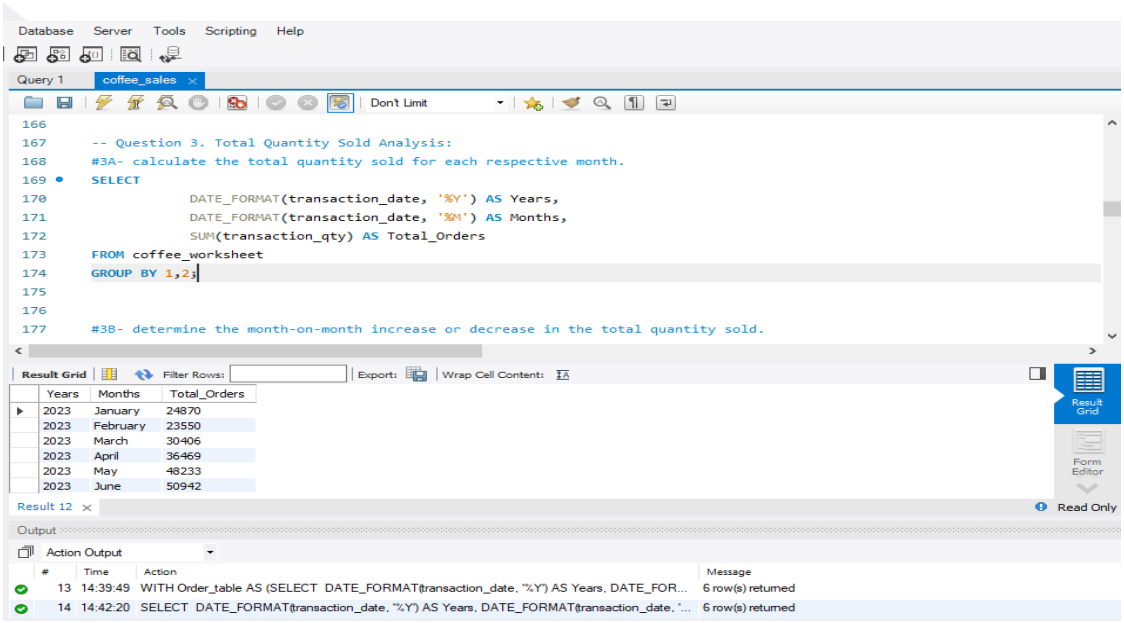
DATE_FORMAT(transaction_date, '%Y') AS Years,

DATE_FORMAT(transaction_date, '%M') AS Months,

SUM(transaction_qty) AS Total_Orders

FROM coffee_worksheet

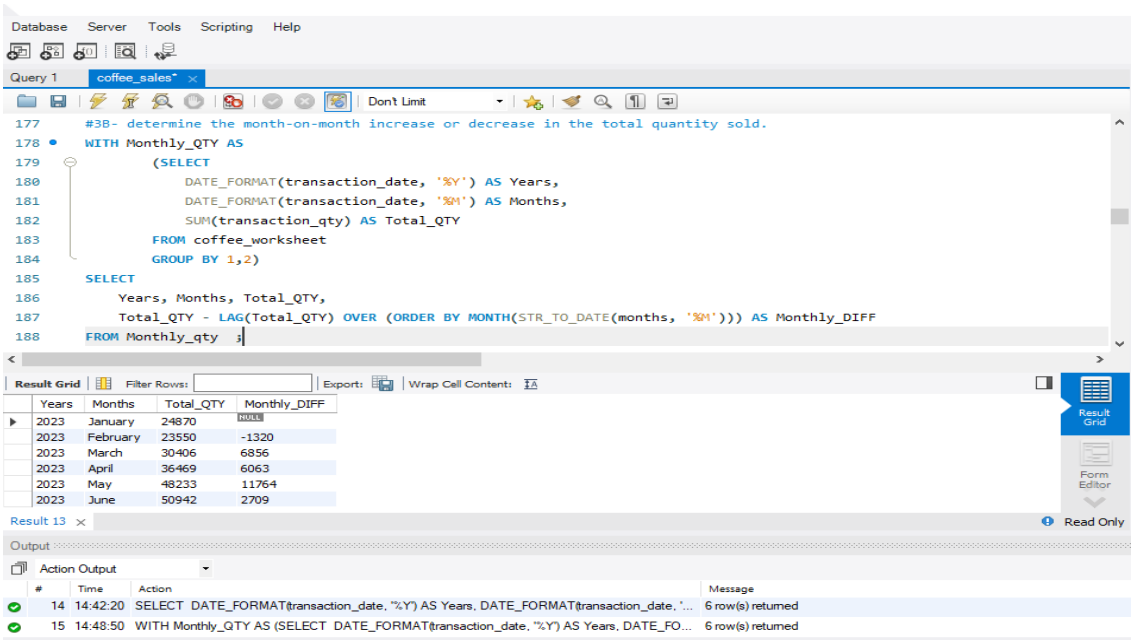
GROUP BY 1,2;



- **3B. month-on-month increase**

SQL

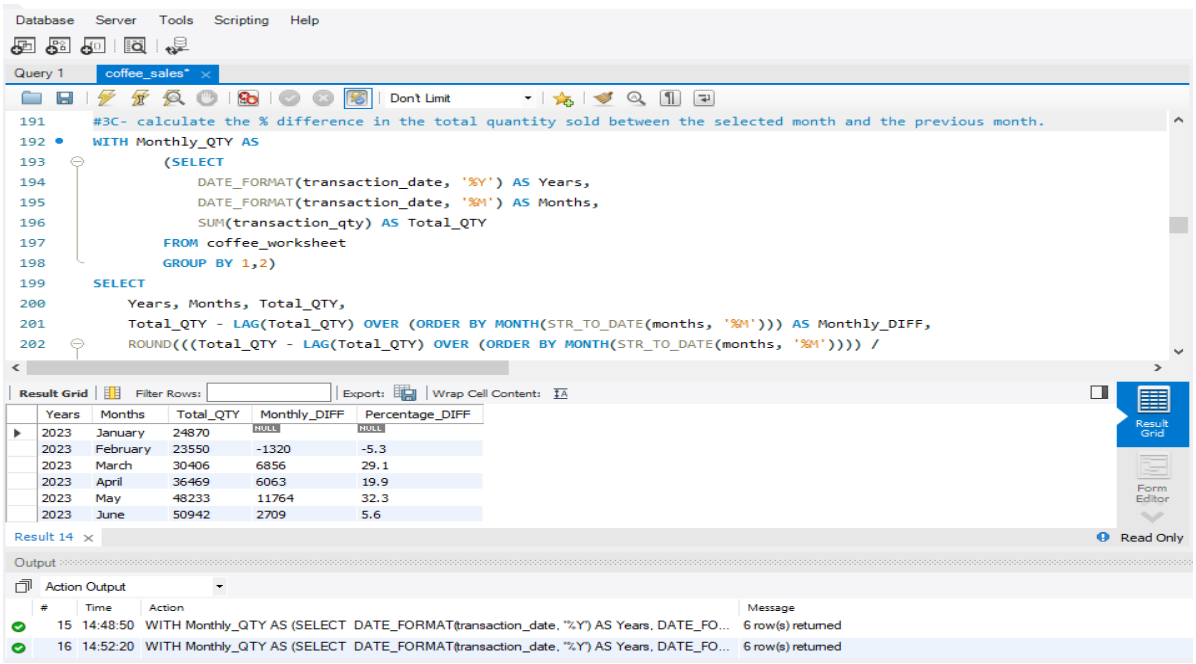
```
WITH Monthly_QTY AS
 (SELECT
 DATE_FORMAT(transaction_date, '%Y') AS Years,
 DATE_FORMAT(transaction_date, '%M') AS Months,
 SUM(transaction_qty) AS Total_QTY
FROM coffee_worksheet
GROUP BY 1,2)
SELECT
 Years, Months, Total_QTY,
 Total_QTY - LAG(Total_QTY) OVER (ORDER BY MONTH(STR_TO_DATE(months, '%M'))) AS Monthly_DIFF
FROM Monthly_qty      ;
```



- **3C. percentage difference btw selected month and the previous month.**

SQL

```
WITH Monthly_QTY AS
 (SELECT
 DATE_FORMAT(transaction_date, '%Y') AS Years,
 DATE_FORMAT(transaction_date, '%M') AS Months,
 SUM(transaction_qty) AS Total_QTY
FROM coffee_worksheet
GROUP BY 1,2)
SELECT
 Years, Months, Total_QTY,
 Total_QTY - LAG(Total_QTY) OVER (ORDER BY MONTH(STR_TO_DATE(months, '%M'))) AS Monthly_DIFF,
 ROUND(((Total_QTY - LAG(Total_QTY) OVER (ORDER BY MONTH(STR_TO_DATE(months, '%M')))) /
 LAG(Total_QTY) OVER (ORDER BY MONTH(STR_TO_DATE(months, '%M')))) * 100, 1) AS Percentage_DIFF
FROM Monthly_qty ;
```

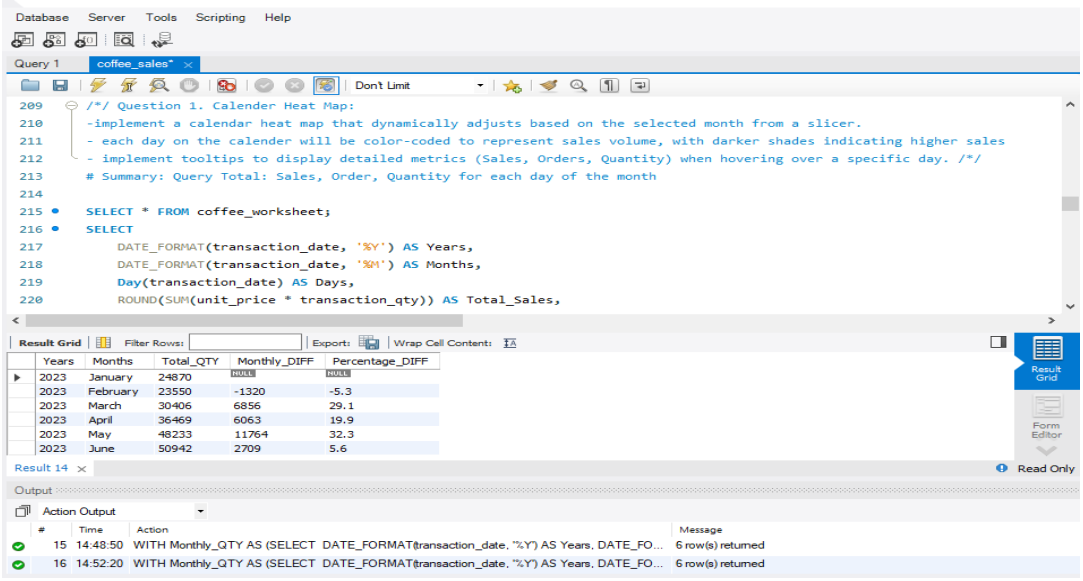


SECTION B: CHARTS REQUIREMENTS

1. The Sales, Order and Quantity for each day of the month

SQL

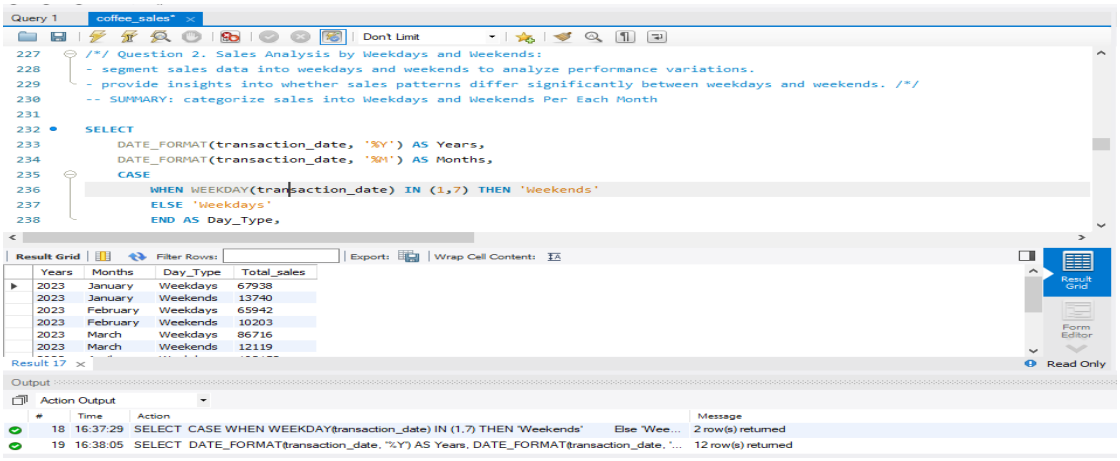
```
SELECT
 DATE_FORMAT(transaction_date, '%Y') AS Years,
 DATE_FORMAT(transaction_date, '%M') AS Months,
 Day(transaction_date) AS Days,
 ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales,
 COUNT(*) AS Total_Orders,
 SUM(transaction_qty) AS Total_QTY
FROM coffee_worksheet
GROUP BY 1,2,3;
```



2. Categorize sales into Weekdays and Weekends Per Each Month

SQL

```
SELECT
 DATE_FORMAT(transaction_date, '%Y') AS Years,
 DATE_FORMAT(transaction_date, '%M') AS Months,
 CASE
 WHEN WEEKDAY(transaction_date) IN (1,7) THEN 'Weekends'
 ELSE 'Weekdays'
 END AS Day_Type,
 ROUND(SUM(transaction_qty * unit_price)) AS Total_sales
FROM coffee_worksheet
GROUP BY 1,2,3;
```

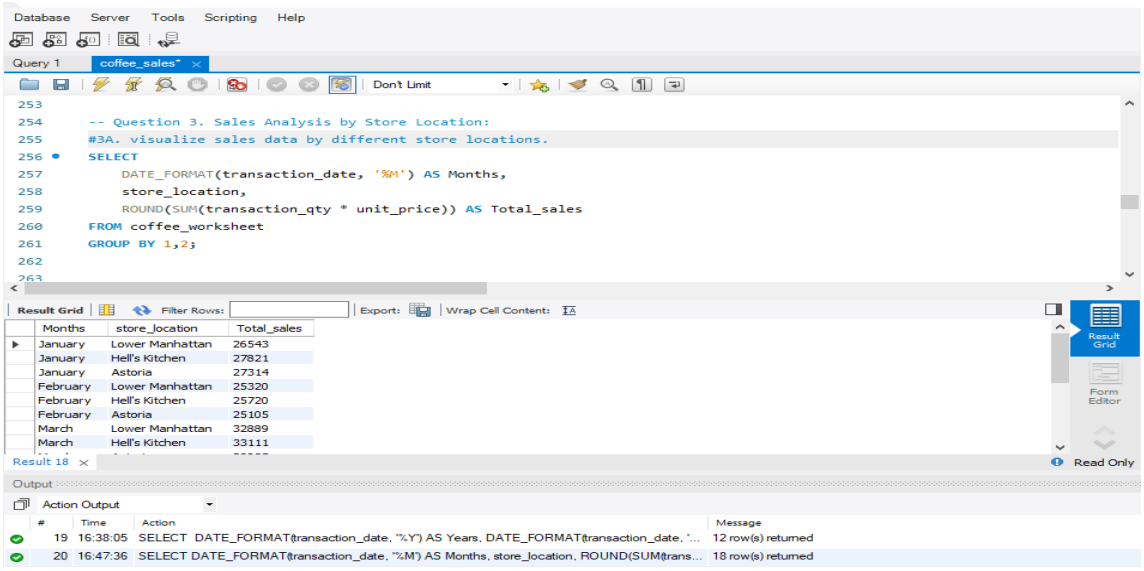


3. Sales Analysis by Store Location:

- **3A. visualize sales data by different store locations.**

SQL

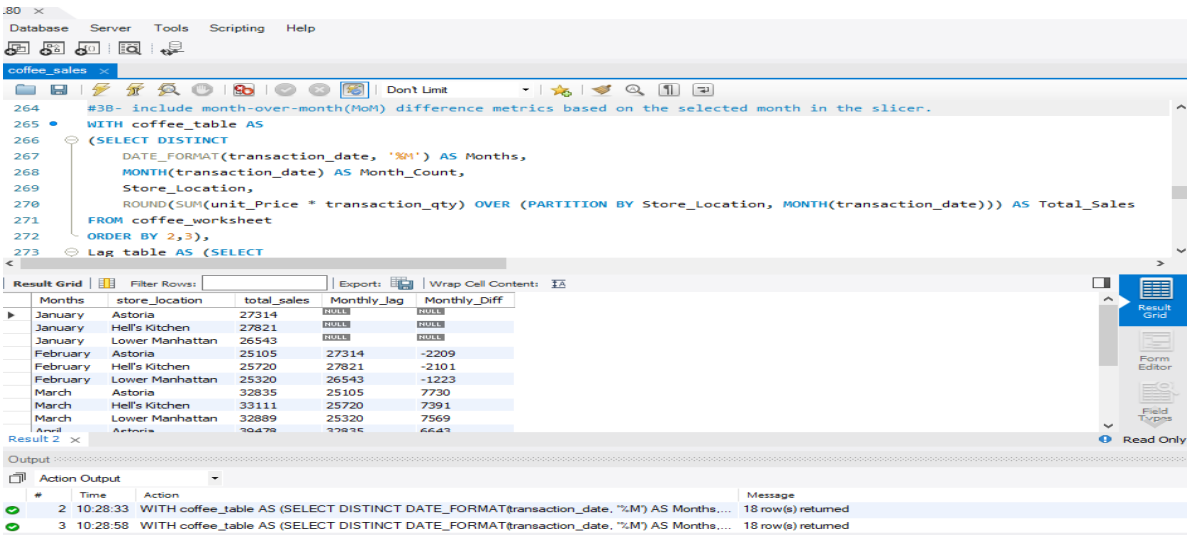
```
SELECT
 DATE_FORMAT(transaction_date, '%M') AS Months,
 store_location,
 ROUND(SUM(transaction_qty * unit_price)) AS Total_sales
FROM coffee_worksheet
GROUP BY 1,2;
```



- **3B. month-over-month(MoM) difference metrics based on the selected month in the slicer.**

SQL

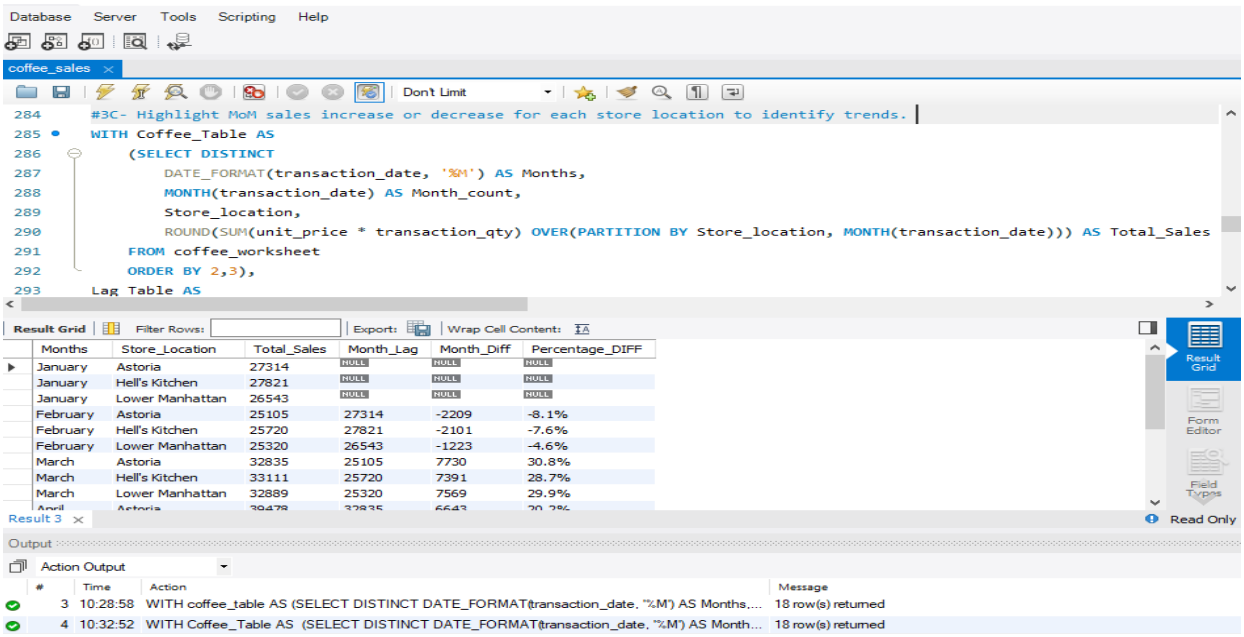
```
WITH coffee_table AS
 (SELECT DISTINCT
 DATE_FORMAT(transaction_date, '%M') AS Months,
 MONTH(transaction_date) AS Month_Count,
 Store_Location,
 ROUND(SUM(unit_Price * transaction_qty) OVER (PARTITION BY Store_Location, MONTH(transaction_date))) AS Total_Sales
FROM coffee_worksheet
ORDER BY 2,3),
Lag_table AS (SELECT
 Months, Month_count, Store_location, Total_Sales,
 LAG(Total_sales) OVER (PARTITION BY store_location ORDER BY Month_count) AS Monthly_Lag
FROM coffee_table)
SELECT
 Months, store_location, total_sales, Monthly_lag,
 (total_sales - Monthly_Lag) AS Monthly_Diff
FROM Lag_Table
ORDER BY Month_count, store_location ;
```



- **3C. MoM sales increase or decrease for each store location to identify trends.**

SQL

```
WITH Coffee_Table AS
 (SELECT DISTINCT
 DATE_FORMAT(transaction_date, '%M') AS Months,
 MONTH(transaction_date) AS Month_count,
 Store_location,
 ROUND(SUM(unit_price * transaction_qty) OVER(PARTITION BY Store_location, MONTH(transaction_date))) AS Total_Sales
FROM coffee_worksheet
ORDER BY 2,3),
Lag_Table AS
 (SELECT
 Months, Month_count, store_location, total_sales,
 LAG(total_sales) OVER (PARTITION BY store_location ORDER BY Month_count) AS Month_Lag
FROM Coffee_table)
SELECT
 Months, Store_Location, Total_Sales, Month_Lag,
 (Total_Sales - Month_Lag) AS Month_Diff,
 CONCAT(ROUND(((Total_Sales - Month_Lag) / Month_Lag) * 100,1), '%') AS Percentage_DIFF
FROM Lag_Table
ORDER BY month_count, 2;
```

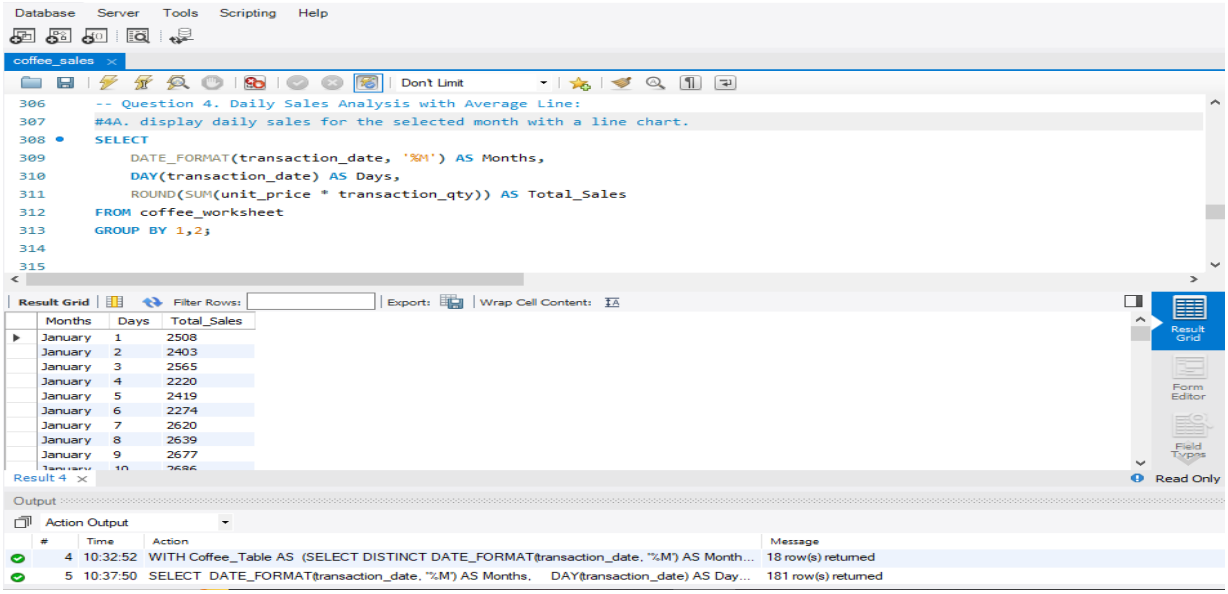


4. Daily Sales Analysis with Average Line:

- **4A. display daily sales for the selected month with a line chart.**

SQL

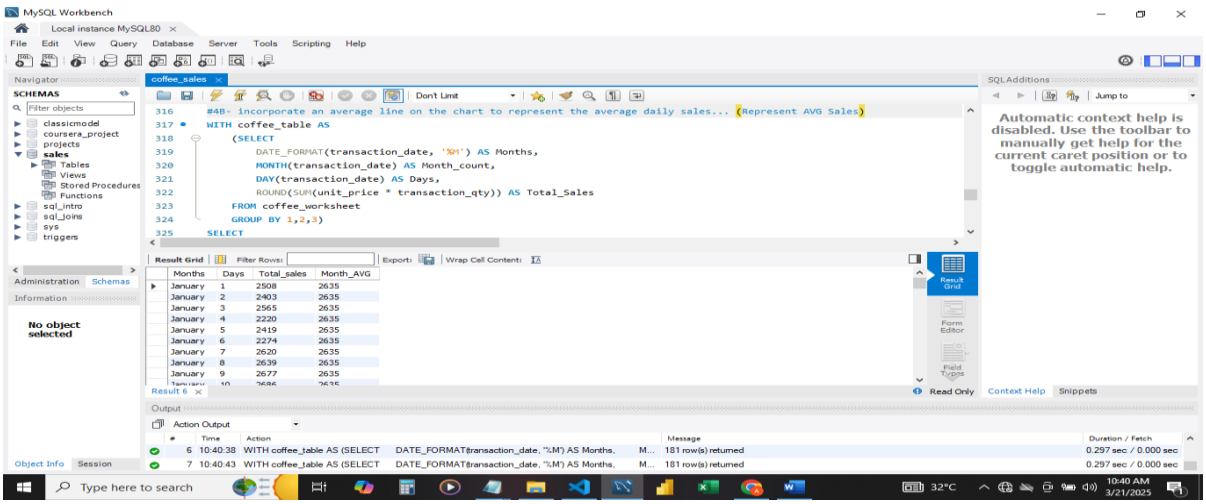
```
SELECT
 DATE_FORMAT(transaction_date, '%M') AS Months,
 DAY(transaction_date) AS Days,
 ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM coffee_worksheet
GROUP BY 1,2;
```



- **4B. incorporate an average line on the chart to represent the average daily sale**

SQL

```
WITH coffee_table AS
 (SELECT
 DATE_FORMAT(transaction_date, '%M') AS Months,
 MONTH(transaction_date) AS Month_count,
 DAY(transaction_date) AS Days,
 ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM coffee_worksheet
GROUP BY 1,2,3)
SELECT
 Months, Days, Total_sales,
 ROUND(AVG(Total_sales) OVER (PARTITION BY Month_count)) AS Month_AVG
FROM coffee_table;
```

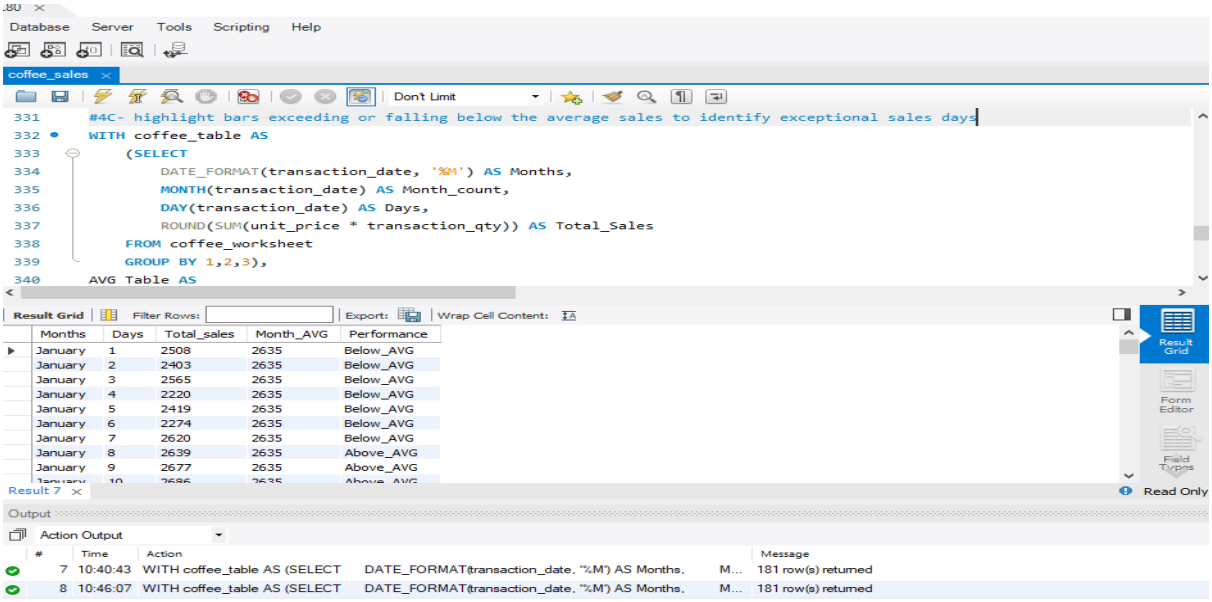


- **4C. show whether it exceeding or falling below the average sales to identify exceptional sales days**

SQL

```
WITH coffee_table AS
        (SELECT
        DATE_FORMAT(transaction_date, '%M') AS Months,
        MONTH(transaction_date) AS Month_count,
        DAY(transaction_date) AS Days,
        ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM coffee_worksheet
GROUP BY 1,2,3),
AVG_Table AS
        (SELECT
        Months, Month_count, Days, Total_sales,
        ROUND(AVG(Total_sales) OVER (PARTITION BY Month_count)) AS Month_AVG
FROM coffee_table)
SELECT
        Months, Days, Total_sales, Month_AVG,
```

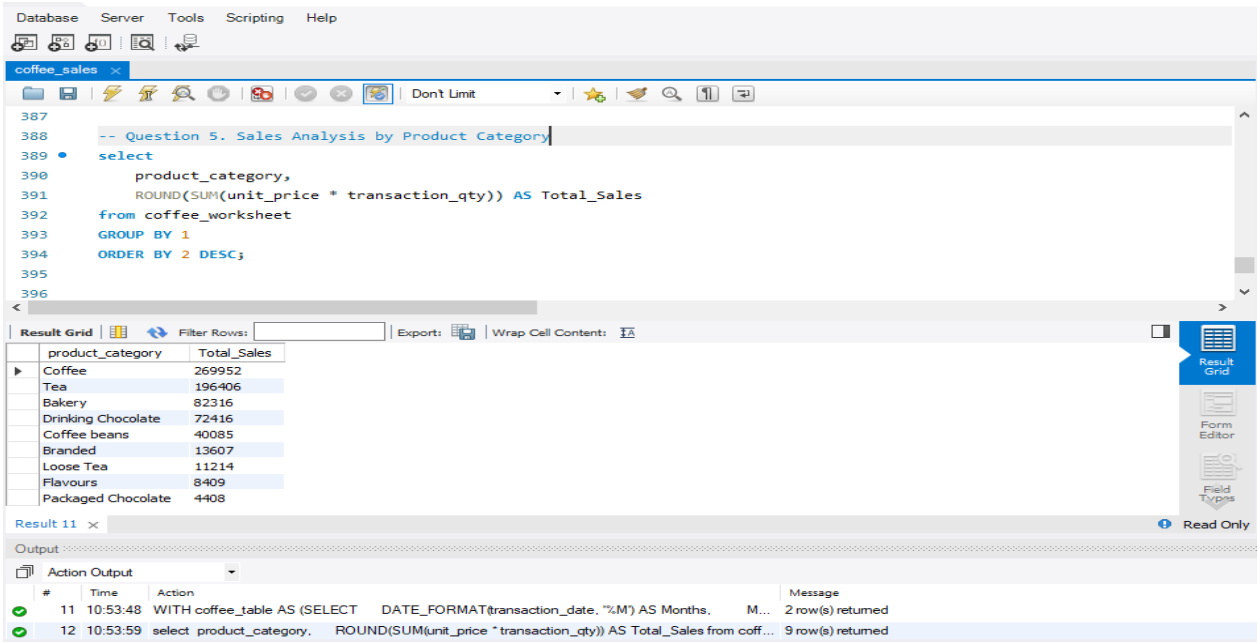
```
        CASE
            WHEN Total_sales > Month_avg Then 'Above_AVG'
            WHEN Total_sales < Month_avg Then 'Below_AVG'
        ELSE 'Average'
            END AS Performance
FROM AVG_Table
ORDER BY month_count, days;
```



5. Sales Analysis by Product Category

SQL

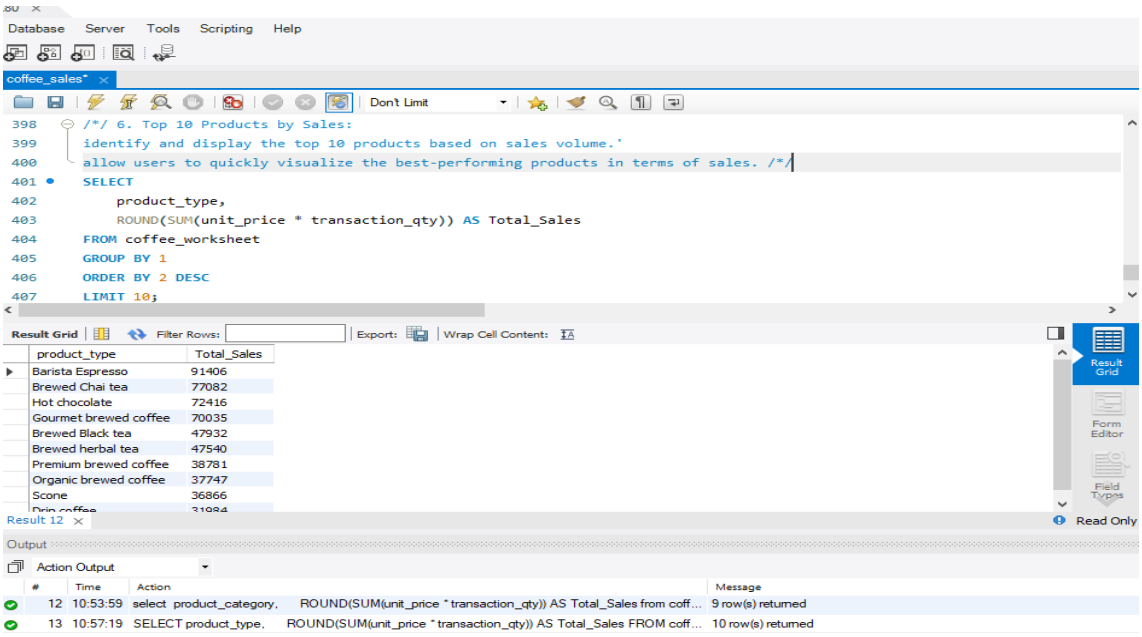
```
SELECT
product_category,
ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM coffee_worksheet
GROUP BY 1
ORDER BY 2 DESC;
```



6. Top 10 Products by Sales

SQL

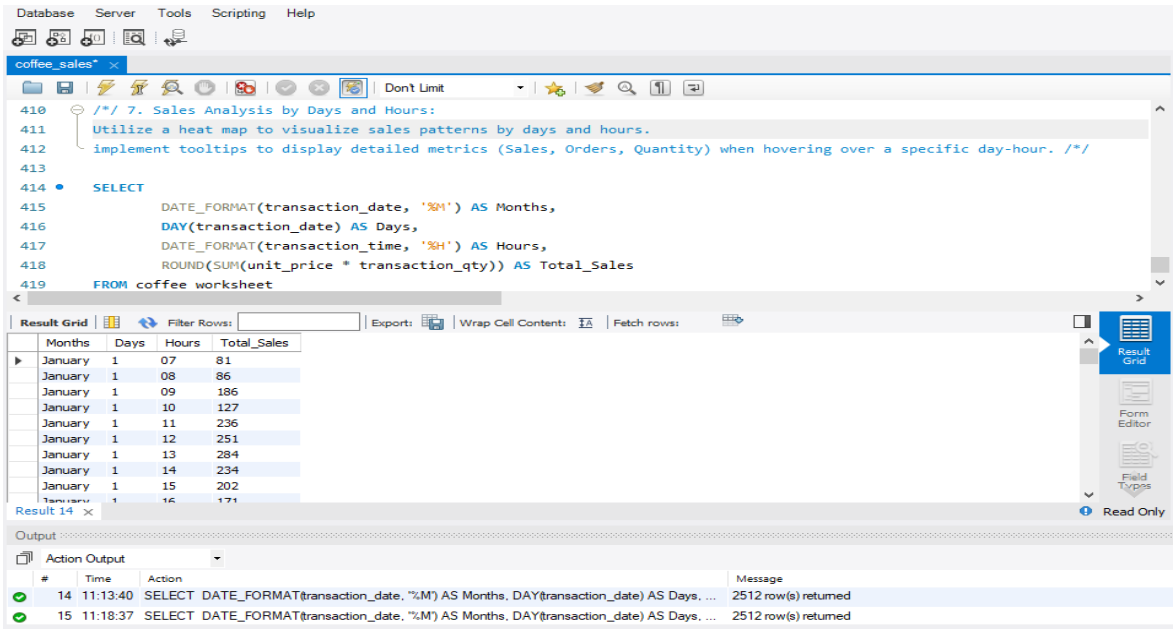
```
SELECT
        product_type,
        ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM coffee_worksheet
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10;
```



7. Sales Analysis by Days and Hours

sql

```
SELECT
    DATE_FORMAT(transaction_date, '%M') AS Months,
    DAY(transaction_date) AS Days,
    DATE_FORMAT(transaction_time, '%H') AS Hours,
    ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM coffee_worksheet
GROUP BY 1,2,3;
```


Database Server Tools Scripting Help
coffee_sales*
Don't Limit
410 /*/ 7. Sales Analysis by Days and Hours:
411 Utilize a heat map to visualize sales patterns by days and hours.
412 implement tooltips to display detailed metrics (Sales, Orders, Quantity) when hovering over a specific day-hour. /*/
413
414 SELECT
415 DATE_FORMAT(transaction_date, '%M') AS Months,
416 DAY(transaction_date) AS Days,
417 DATE_FORMAT(transaction_time, '%H') AS Hours,
418 ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
419 FROM coffee worksheet
Result Grid Filter Rows: Export: Wrap Cell Content: Fetch rows:
Months Days Hours Total_Sales
January 1 07 81
January 1 08 86
January 1 09 186
January 1 10 127
January 1 11 236
January 1 12 251
January 1 13 284
January 1 14 234
January 1 15 202
Result Grid
Form Editor
Field Types
Read Only
Result 14
Output
Action Output
Time Action Message
14 11:13:40 SELECT DATE_FORMAT(transaction_date, '%M') AS Months, DAY(transaction_date) AS Days, ... 2512 row(s) returned
15 11:18:37 SELECT DATE_FORMAT(transaction_date, '%M') AS Months, DAY(transaction_date) AS Days, ... 2512 row(s) returned