SQL Solution Section

Data Extraction and Cleaning

1: Create table coffee_sales

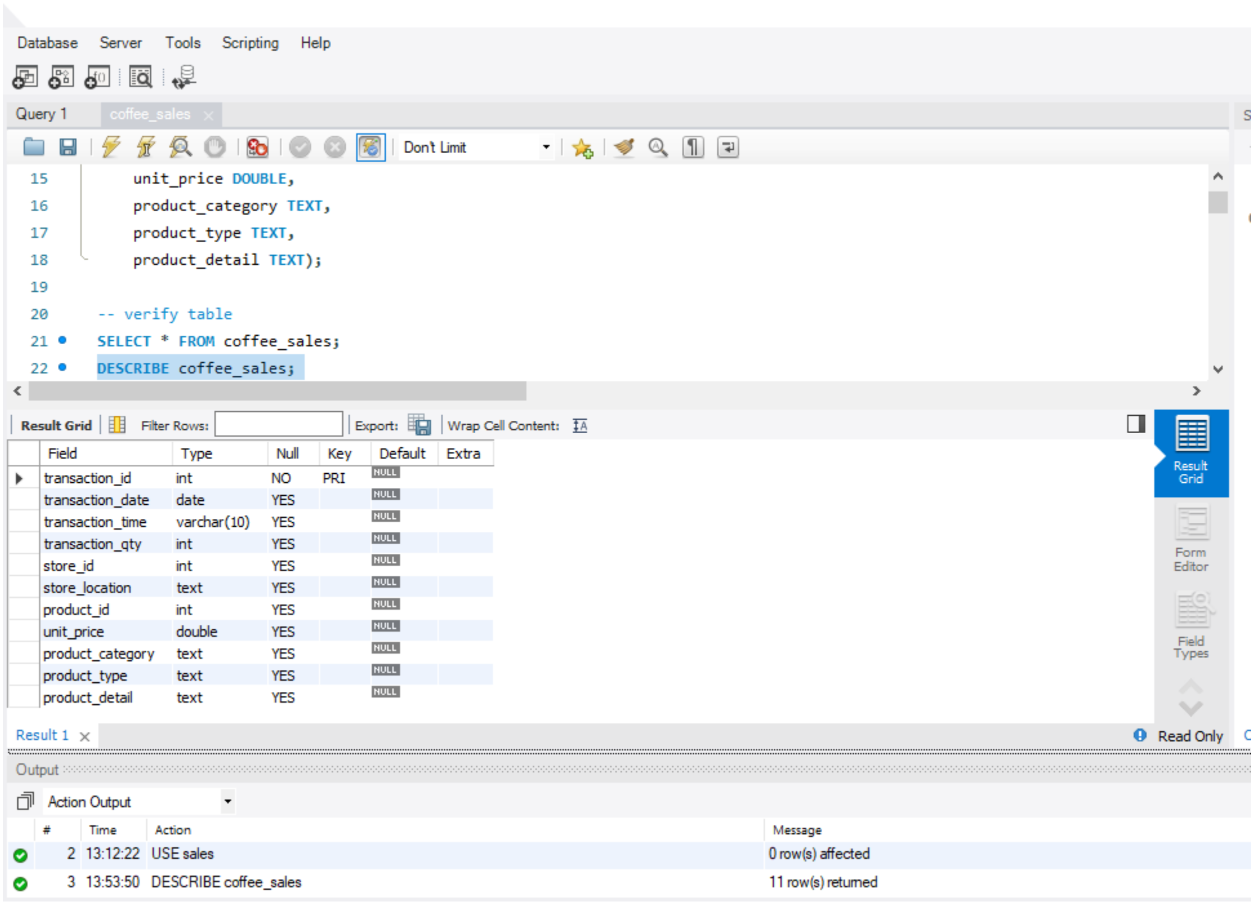
SQL:

```
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;
```



```
    unit_price DOUBLE,
    product_category TEXT,
    product_type TEXT,
    product_detail TEXT);

-- verify table
SELECT * FROM coffee_sales;
DESCRIBE coffee_sales;
```

Field	Type	Null	Key	Default	Extra
transaction_id	int	NO	PRI	NULL	
transaction_date	date	YES		NULL	
transaction_time	varchar(10)	YES		NULL	
transaction_qty	int	YES		NULL	
store_id	int	YES		NULL	
store_location	text	YES		NULL	
product_id	int	YES		NULL	
unit_price	double	YES		NULL	
product_category	text	YES		NULL	
product_type	text	YES		NULL	
product_detail	text	YES		NULL	

```
#  Time      Action                 Message
2  13:12:22  USE sales              0 row(s) affected
3  13:53:50  DESCRIBE coffee_sales  11 row(s) returned
```

2: Import the dataset into coffee_sales table

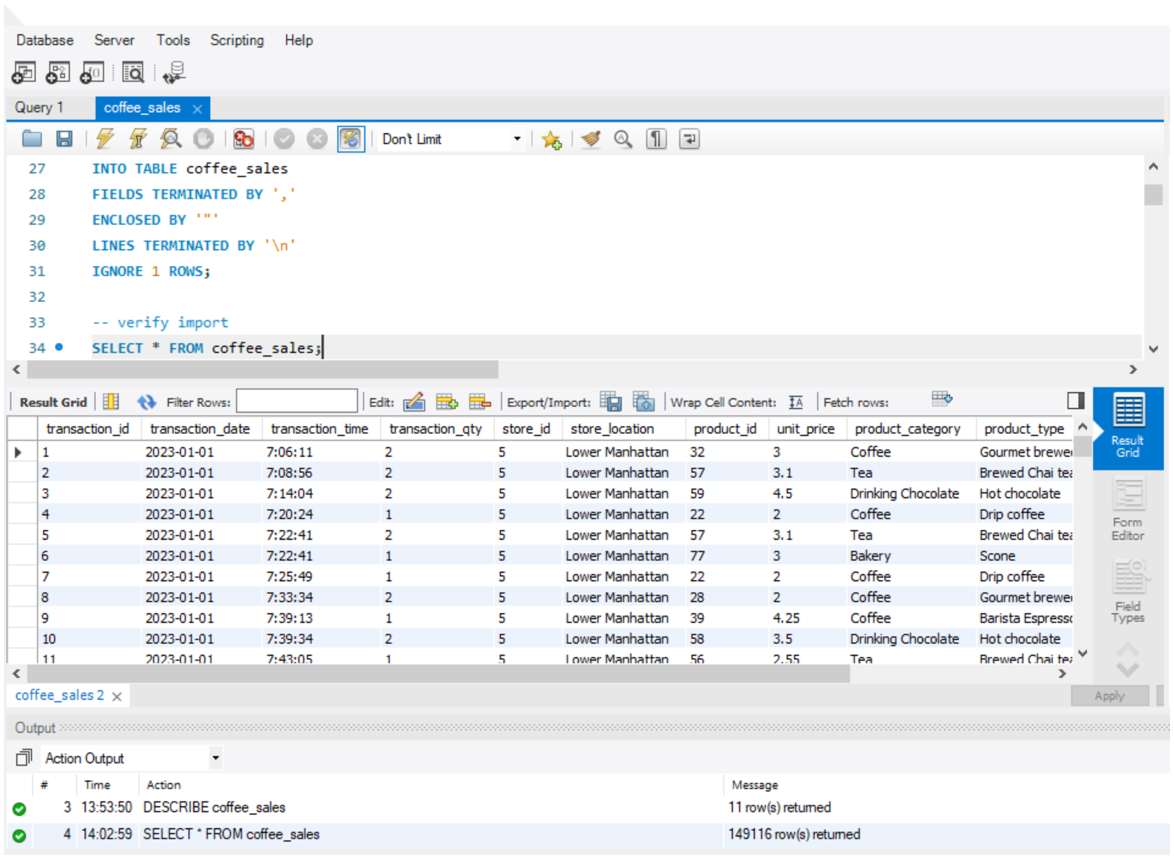
SQL

```
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;



```
INTO TABLE coffee_sales
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

-- verify import
SELECT * FROM coffee_sales;
```

transaction_id	transaction_date	transaction_time	transaction_qty	store_id	store_location	product_id	unit_price	product_category	product_type
1	2023-01-01	7:06:11	2	5	Lower Manhattan	32	3	Coffee	Gourmet brewe
2	2023-01-01	7:08:56	2	5	Lower Manhattan	57	3.1	Tea	Brewed Chai tea
3	2023-01-01	7:14:04	2	5	Lower Manhattan	59	4.5	Drinking Chocolate	Hot chocolate
4	2023-01-01	7:20:24	1	5	Lower Manhattan	22	2	Coffee	Drip coffee
5	2023-01-01	7:22:41	2	5	Lower Manhattan	57	3.1	Tea	Brewed Chai tea
6	2023-01-01	7:22:41	1	5	Lower Manhattan	77	3	Bakery	Scone
7	2023-01-01	7:25:49	1	5	Lower Manhattan	22	2	Coffee	Drip coffee
8	2023-01-01	7:33:34	2	5	Lower Manhattan	28	2	Coffee	Gourmet brewe
9	2023-01-01	7:39:13	1	5	Lower Manhattan	39	4.25	Coffee	Barista Espress
10	2023-01-01	7:39:34	2	5	Lower Manhattan	58	3.5	Drinking Chocolate	Hot chocolate
11	2023-01-01	7:43:05	1	5	Lower Manhattan	56	2.55	Tea	Brewed Chai tea

```
#   Time      Action                          Message
3   13:53:50  DESCRIBE coffee_sales           11 row(s) returned
4   14:02:59  SELECT * FROM coffee_sales      149116 row(s) returned
```

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

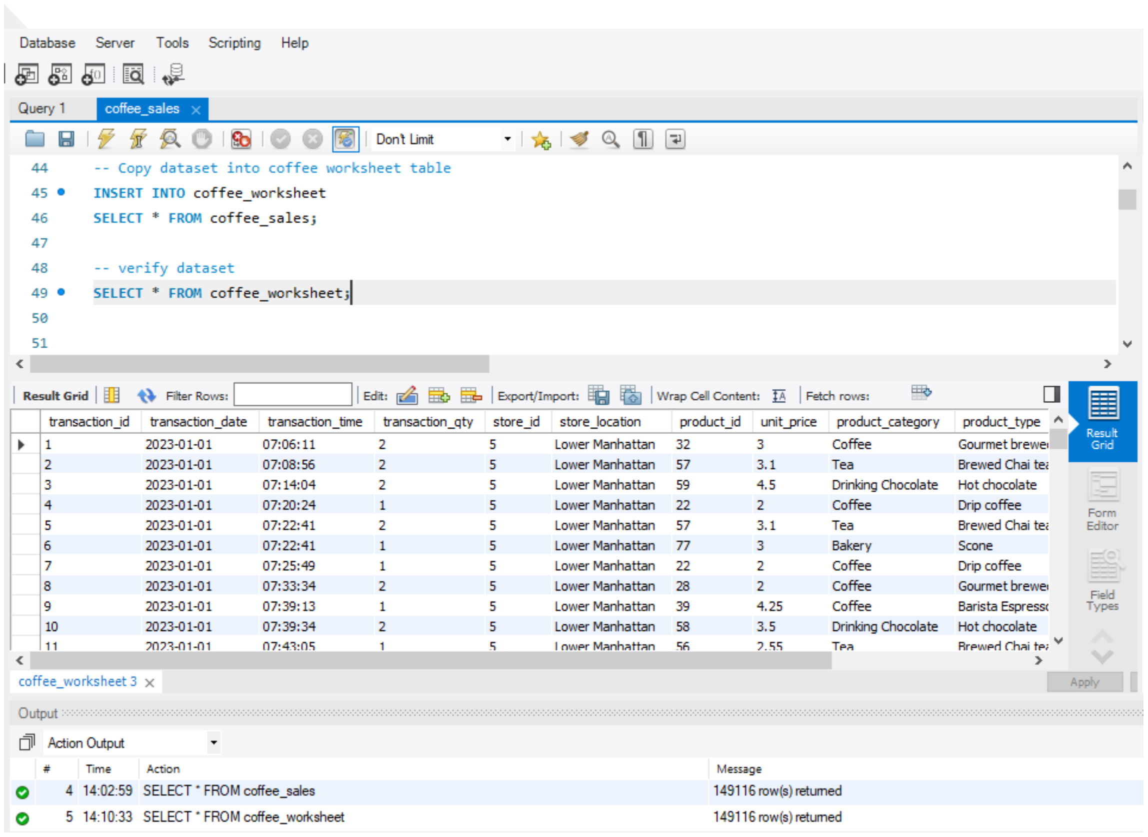
SQL

```
INSERT INTO coffee_worksheet

SELECT * FROM coffee_sales;
```

- **Verify dataset**

SQL

```
SELECT * FROM coffee_worksheet;
```



```
-- Copy dataset into coffee worksheet table
INSERT INTO coffee_worksheet
SELECT * FROM coffee_sales;

-- verify dataset
SELECT * FROM coffee_worksheet;
```

transaction_id	transaction_date	transaction_time	transaction_qty	store_id	store_location	product_id	unit_price	product_category	product_type
1	2023-01-01	07:06:11	2	5	Lower Manhattan	32	3	Coffee	Gourmet brewe
2	2023-01-01	07:08:56	2	5	Lower Manhattan	57	3.1	Tea	Brewed Chai tea
3	2023-01-01	07:14:04	2	5	Lower Manhattan	59	4.5	Drinking Chocolate	Hot chocolate
4	2023-01-01	07:20:24	1	5	Lower Manhattan	22	2	Coffee	Drip coffee
5	2023-01-01	07:22:41	2	5	Lower Manhattan	57	3.1	Tea	Brewed Chai tea
6	2023-01-01	07:22:41	1	5	Lower Manhattan	77	3	Bakery	Scone
7	2023-01-01	07:25:49	1	5	Lower Manhattan	22	2	Coffee	Drip coffee
8	2023-01-01	07:33:34	2	5	Lower Manhattan	28	2	Coffee	Gourmet brewe
9	2023-01-01	07:39:13	1	5	Lower Manhattan	39	4.25	Coffee	Barista Espresso
10	2023-01-01	07:39:34	2	5	Lower Manhattan	58	3.5	Drinking Chocolate	Hot chocolate
11	2023-01-01	07:43:05	1	5	Lower Manhattan	56	2.55	Tea	Brewed Chai tea

#	Time	Action	Message
4	14:02:59	SELECT * FROM coffee_sales	149116 row(s) returned
5	14:10:33	SELECT * FROM coffee_worksheet	149116 row(s) returned

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;
```

<u>EXPLORATORY DATA ANALYSIS ON BUSINESS QUESTIONS</u>

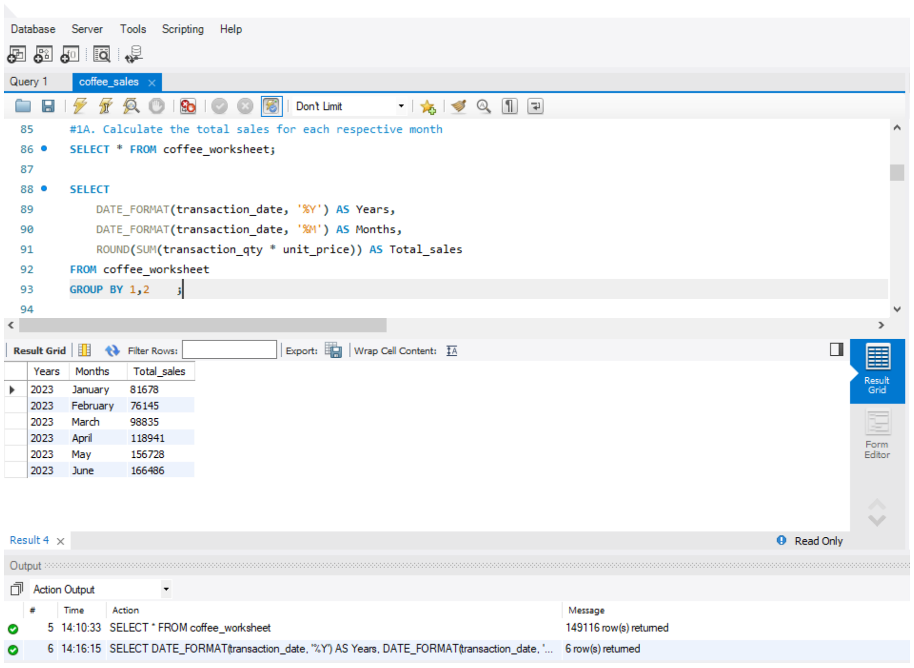
<u>SECTION A: KPI'S REQUIREMENTS</u>

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    ;
```



```
#1A. Calculate the total sales for each respective month
SELECT * FROM coffee_worksheet;

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     ;
```

Years	Months	Total_sales
2023	January	81678
2023	February	76145
2023	March	98835
2023	April	118941
2023	May	156728
2023	June	166486

#	Time	Action	Message
5	14:10:33	SELECT * FROM coffee_worksheet	149116 row(s) returned
6	14:16:15	SELECT DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FORMAT(transaction_date, '...	6 row(s) returned

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

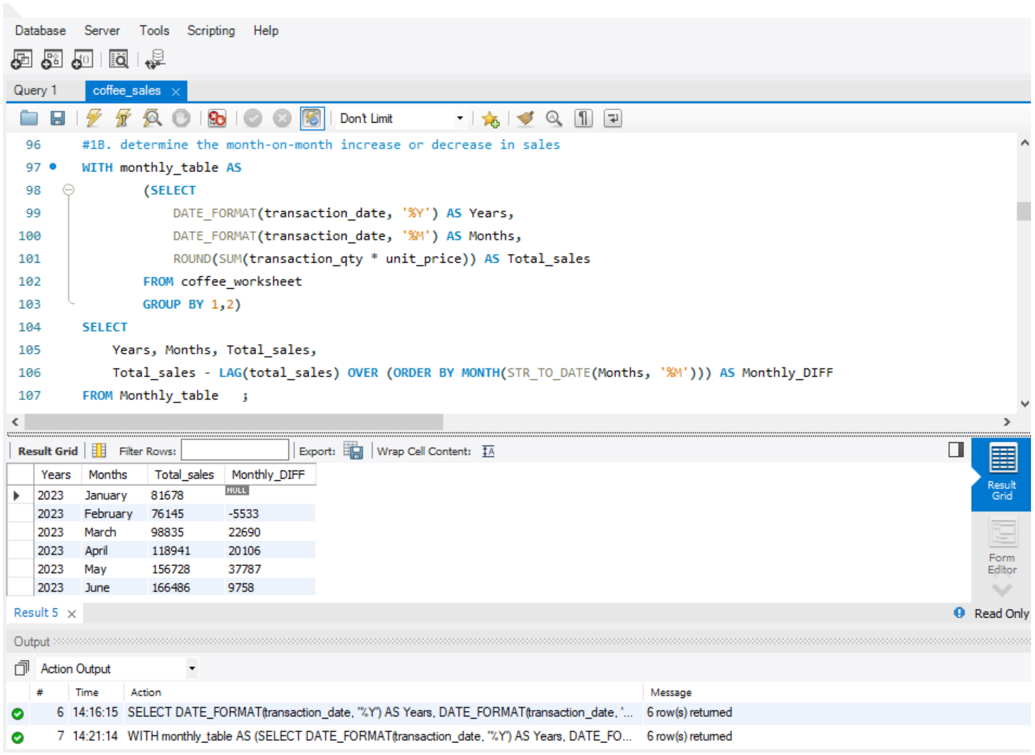
SQL

```
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 ;



```
#1B. determine the month-on-month increase or decrease in sales
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   ;
```

Years	Months	Total_sales	Monthly_DIFF
2023	January	81678	NULL
2023	February	76145	-5533
2023	March	98835	22690
2023	April	118941	20106
2023	May	156728	37787
2023	June	166486	9758

#	Time	Action	Message
6	14:16:15	SELECT DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FORMAT(transaction_date, '...	6 row(s) returned
7	14:21:14	WITH monthly_table AS (SELECT DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FO...	6 row(s) returned

- **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

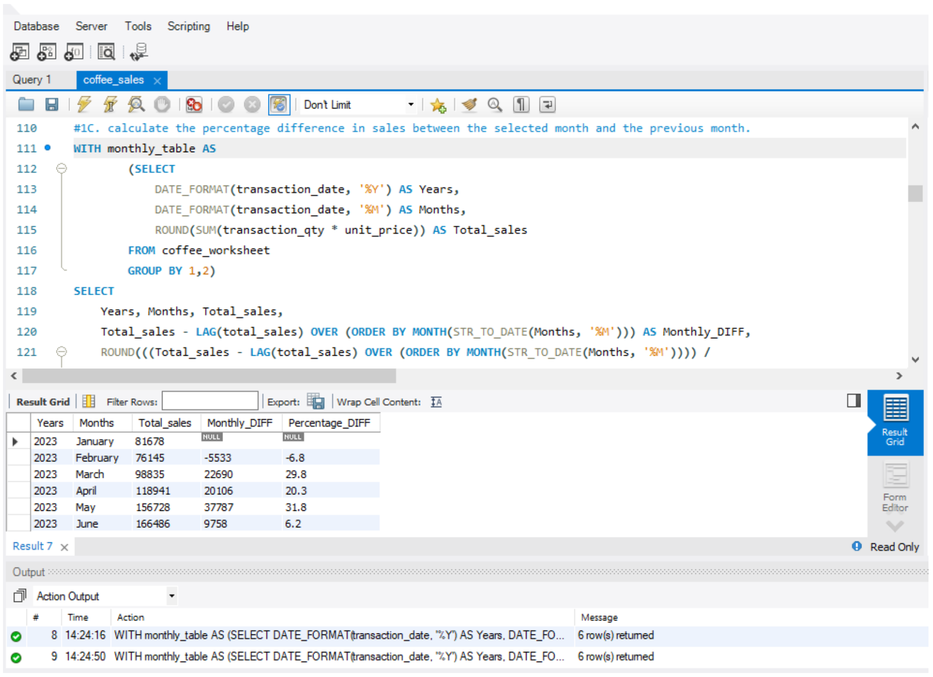
Years, Months, Total_sales,

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 ;



```
#1C. calculate the percentage difference in sales between the selected month and the previous month.
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')))) /
```

Years	Months	Total_sales	Monthly_DIFF	Percentage_DIFF
2023	January	81678	NULL	NULL
2023	February	76145	-5533	-6.8
2023	March	98835	22690	29.8
2023	April	118941	20106	20.3
2023	May	156728	37787	31.8
2023	June	166486	9758	6.2

#	Time	Action	Message
8	14:24:16	WITH monthly_table AS (SELECT DATE_FORMAT(transaction_date, "%Y") AS Years, DATE_FO...	6 row(s) returned
9	14:24:50	WITH monthly_table AS (SELECT DATE_FORMAT(transaction_date, "%Y") AS Years, DATE_FO...	6 row(s) returned

2. Total Order Analysis

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

SQL

SELECT

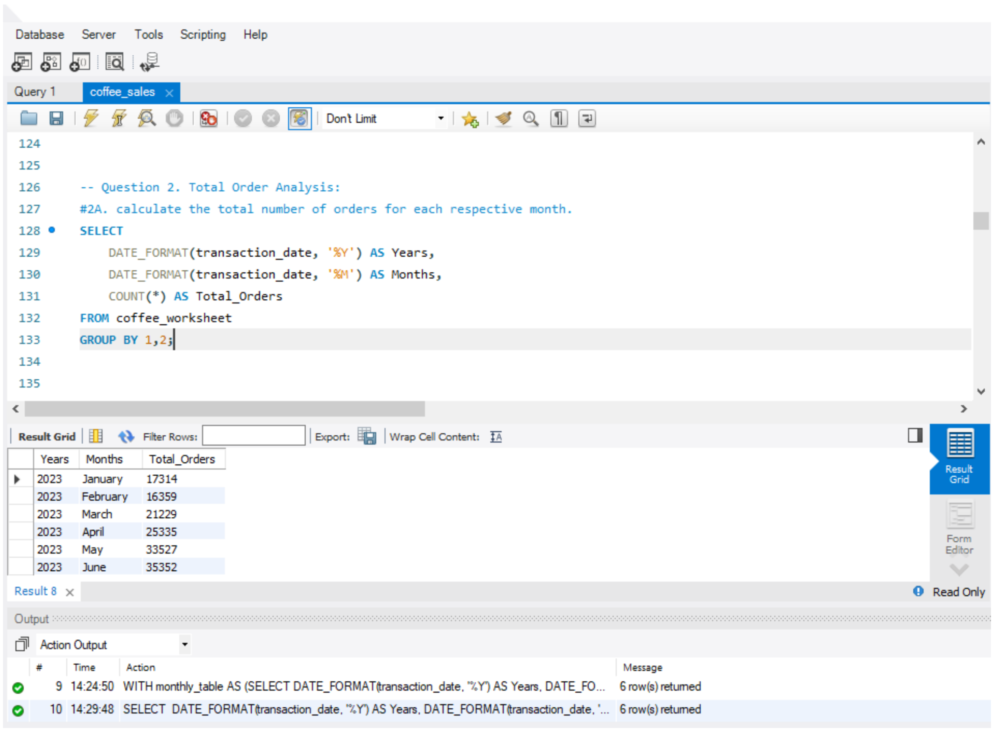
DATE_FORMAT(transaction_date, '%Y') AS Years,

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

COUNT(*) AS Total_Orders

FROM coffee_worksheet

GROUP BY 1,2;



```
-- Question 2. Total Order Analysis:
#2A. calculate the total number of orders for each respective month.
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;
```

Years	Months	Total_Orders
2023	January	17314
2023	February	16359
2023	March	21229
2023	April	25335
2023	May	33527
2023	June	35352

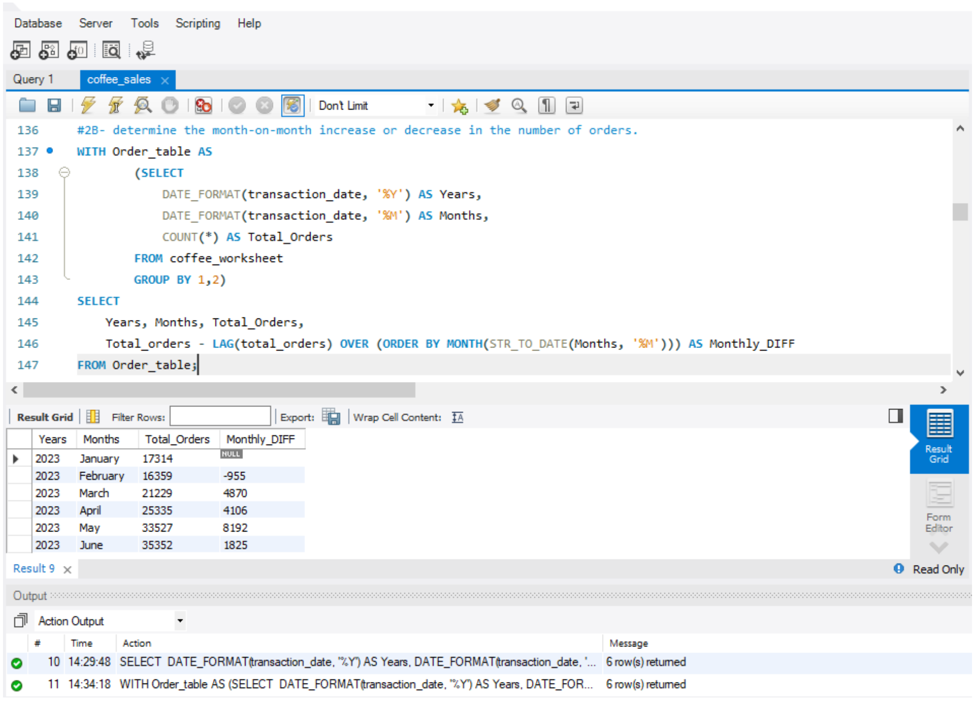
```
#   Time      Action                                                                              Message
9   14:24:50  WITH monthly_table AS (SELECT DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FO...  6 row(s) returned
10  14:29:48  SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FORMAT(transaction_date, '...  6 row(s) returned
```

- **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;



```
#2B- determine the month-on-month increase or decrease in the number of orders.
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;
```

Years	Months	Total_Orders	Monthly_DIFF
2023	January	17314	NULL
2023	February	16359	-955
2023	March	21229	4870
2023	April	25335	4106
2023	May	33527	8192
2023	June	35352	1825

#	Time	Action	Message
10	14:29:48	SELECT DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FORMAT(transaction_date, '...	6 row(s) returned
11	14:34:18	WITH Order_table AS (SELECT DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FOR...	6 row(s) returned

- **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)

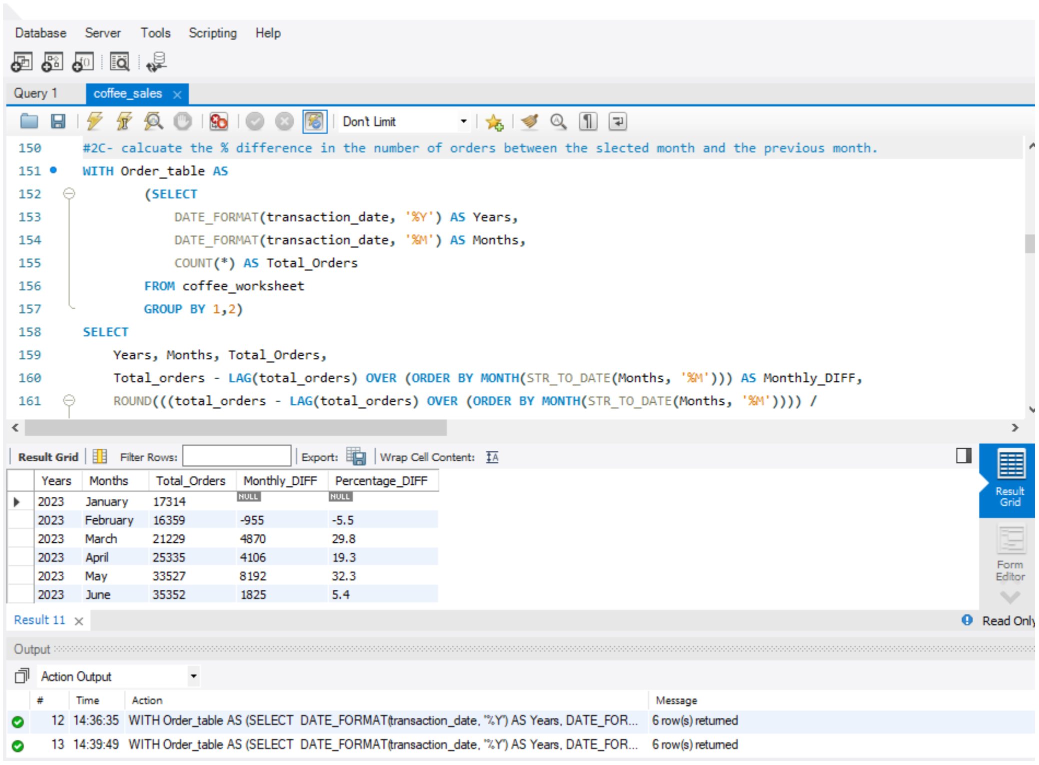
SELECT

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;



```
#2C- calcuate the % difference in the number of orders between the slected month and the previous month.
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')))) /
```

```
  Years  Months    Total_Orders  Monthly_DIFF  Percentage_DIFF
▶ 2023   January   17314         NULL          NULL
  2023   February  16359         -955          -5.5
  2023   March     21229         4870          29.8
  2023   April     25335         4106          19.3
  2023   May       33527         8192          32.3
  2023   June      35352         1825          5.4
```

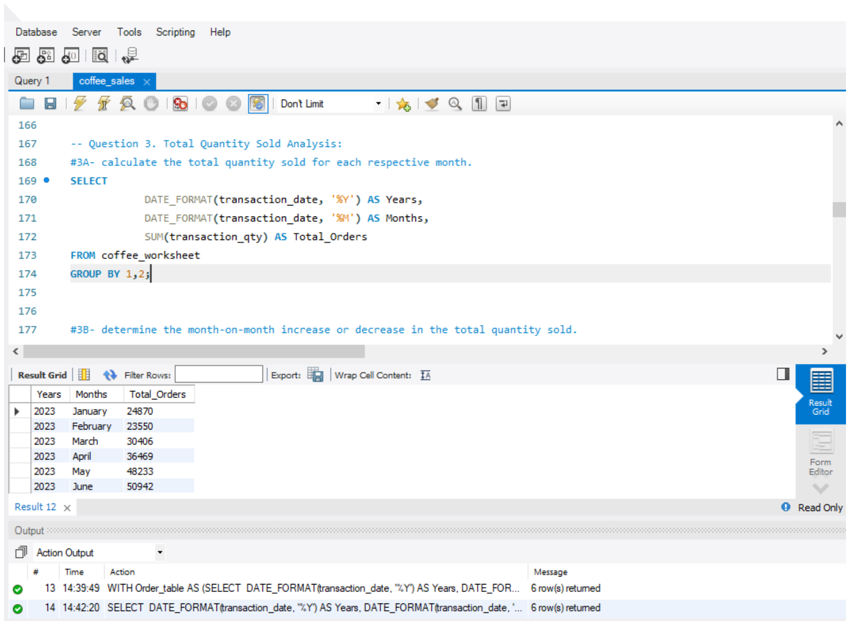
```
#   Time      Action                                                                   Message
12  14:36:35  WITH Order_table AS (SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FOR...  6 row(s) returned
13  14:39:49  WITH Order_table AS (SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FOR...  6 row(s) returned
```

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;
```



```
-- Question 3. Total Quantity Sold Analysis:
#3A- calculate the total quantity sold for each respective month.
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- determine the month-on-month increase or decrease in the total quantity sold.
```

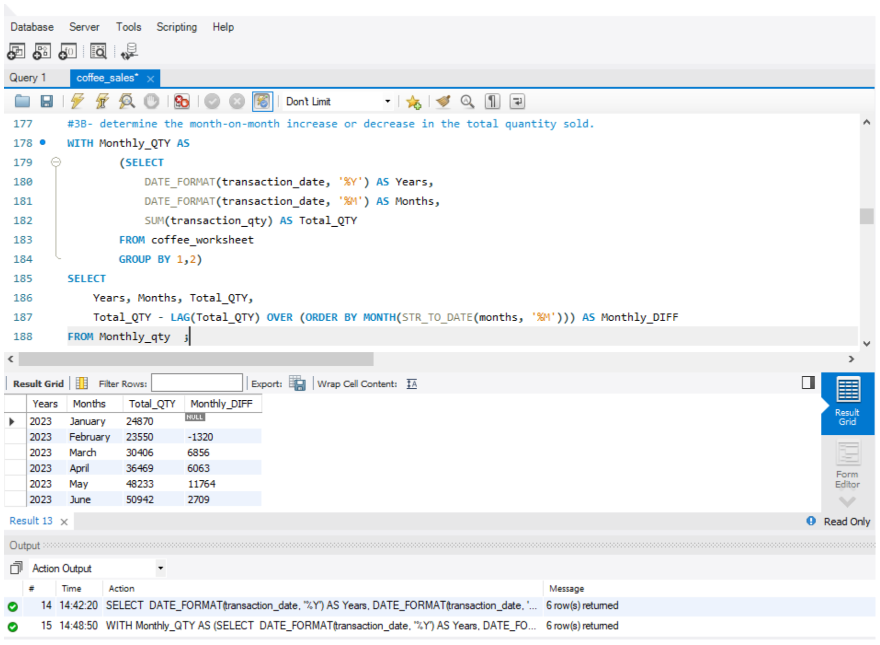
Years	Months	Total_Orders
2023	January	24870
2023	February	23550
2023	March	30406
2023	April	36469
2023	May	48233
2023	June	50942

```
   #   Time      Action                                                                              Message
   13  14:39:49  WITH Order_table AS (SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FOR...   6 row(s) returned
   14  14:42:20  SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FORMAT(transaction_date, '...  6 row(s) returned
```

- **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      ;
```



```
#3B- determine the month-on-month increase or decrease in the total quantity sold.
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   ;
```

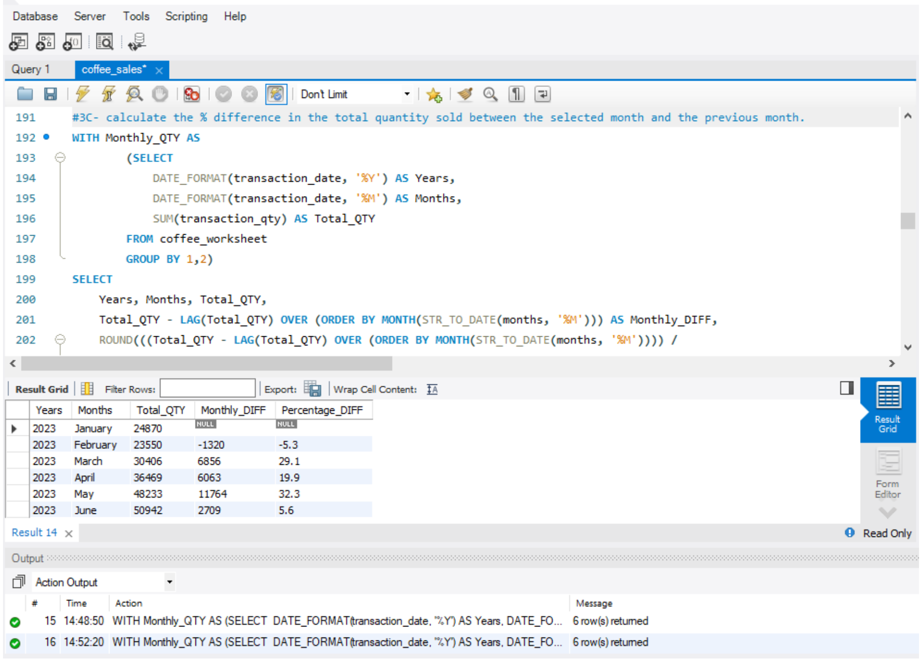
Years	Months	Total_QTY	Monthly_DIFF
2023	January	24870	NULL
2023	February	23550	-1320
2023	March	30406	6856
2023	April	36469	6063
2023	May	48233	11764
2023	June	50942	2709

```
#   Time      Action                                                                         Message
14  14:42:20  SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FORMAT(transaction_date, '...  6 row(s) returned
15  14:48:50  WITH Monthly_QTY AS (SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FO...    6 row(s) returned
```

- **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       ;
```



```
#3C- calculate the % difference in the total quantity sold between the selected month and the previous month.
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')))) /
```

Years	Months	Total_QTY	Monthly_DIFF	Percentage_DIFF
2023	January	24870	NULL	NULL
2023	February	23550	-1320	-5.3
2023	March	30406	6856	29.1
2023	April	36469	6063	19.9
2023	May	48233	11764	32.3
2023	June	50942	2709	5.6

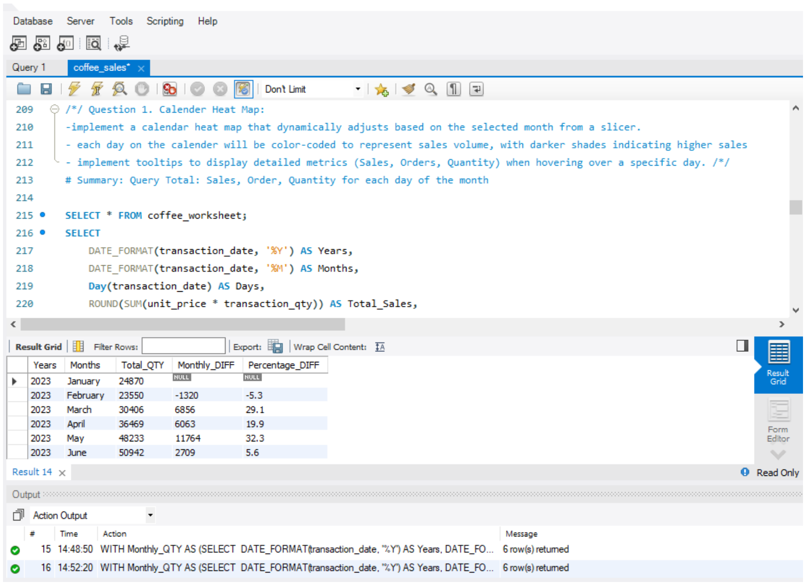
```
#   Time      Action                                                                          Message
15  14:48:50  WITH Monthly_QTY AS (SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FO...  6 row(s) returned
16  14:52:20  WITH Monthly_QTY AS (SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FO...  6 row(s) returned
```

<u>SECTION B: CHARTS REQUIREMENTS</u>

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;
```



```
/*/ Question 1. Calender Heat Map:
-implement a calendar heat map that dynamically adjusts based on the selected month from a slicer.
- each day on the calender will be color-coded to represent sales volume, with darker shades indicating higher sales
- implement tooltips to display detailed metrics (Sales, Orders, Quantity) when hovering over a specific day. /*/
# Summary: Query Total: Sales, Order, Quantity for each day of the month

SELECT * FROM coffee_worksheet;
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,
```

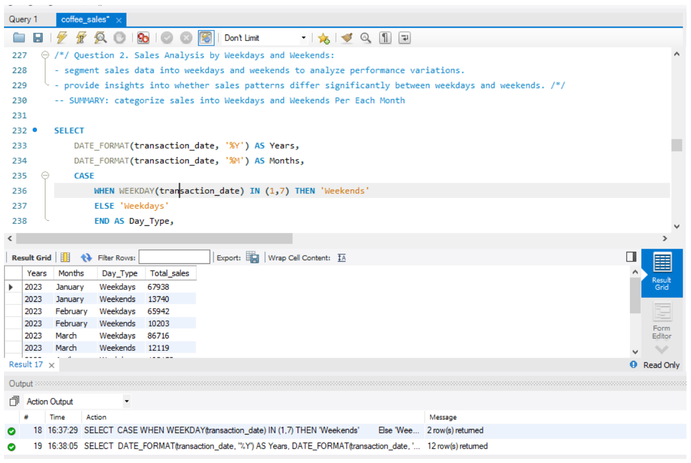
Years	Months	Total_QTY	Monthly_DIFF	Percentage_DIFF
2023	January	24870	NULL	NULL
2023	February	23550	-1320	-5.3
2023	March	30406	6856	29.1
2023	April	36469	6063	19.9
2023	May	48233	11764	32.3
2023	June	50942	2709	5.6

```
#   Time      Action                                                                                  Message
15  14:48:50  WITH Monthly_QTY AS (SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FO...  6 row(s) returned
16  14:52:20  WITH Monthly_QTY AS (SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FO...  6 row(s) returned
```

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;
```



```
/*/ Question 2. Sales Analysis by Weekdays and Weekends:
- segment sales data into weekdays and weekends to analyze performance variations.
- provide insights into whether sales patterns differ significantly between weekdays and weekends. /*/
-- SUMMARY: categorize sales into Weekdays and Weekends Per Each Month

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,
```

Years	Months	Day_Type	Total_sales
2023	January	Weekdays	67938
2023	January	Weekends	13740
2023	February	Weekdays	65942
2023	February	Weekends	10203
2023	March	Weekdays	86716
2023	March	Weekends	12119

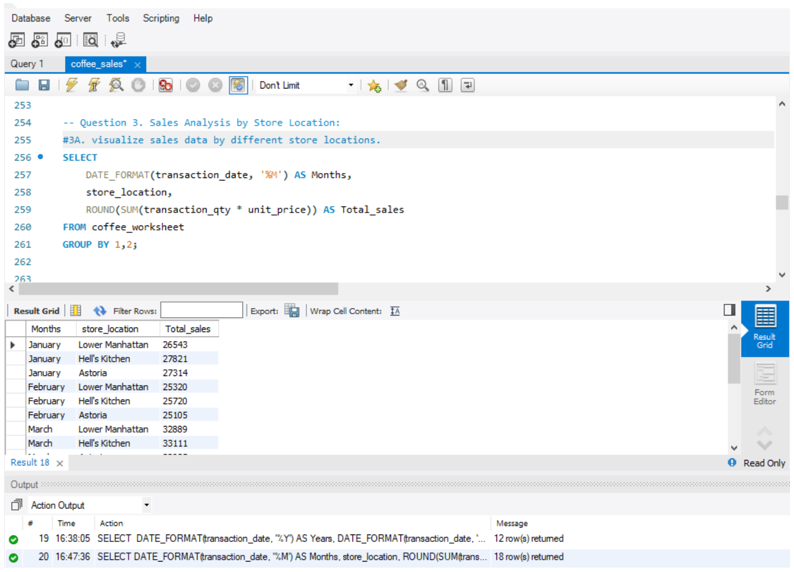
```
#   Time      Action                                                                       Message
18  16:37:29  SELECT  CASE WHEN WEEKDAY(transaction_date) IN (1,7) THEN 'Weekends'   Else 'Wee...  2 row(s) returned
19  16:38:05  SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FORMAT(transaction_date, '...  12 row(s) returned
```

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;
```



```
-- Question 3. Sales Analysis by Store Location:
#3A. visualize sales data by different store locations.
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;
```

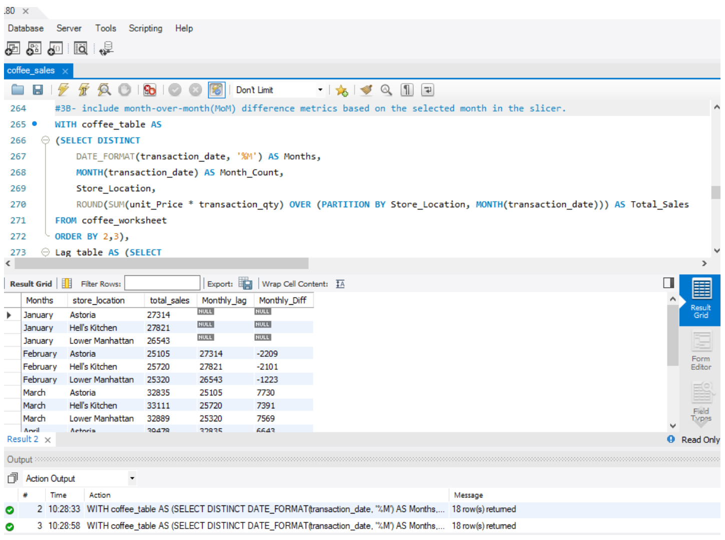
Months	store_location	Total_sales
January	Lower Manhattan	26543
January	Hell's Kitchen	27821
January	Astoria	27314
February	Lower Manhattan	25320
February	Hell's Kitchen	25720
February	Astoria	25105
March	Lower Manhattan	32889
March	Hell's Kitchen	33111

```
#   Time      Action                                                                          Message
19  16:38:05  SELECT  DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FORMAT(transaction_date, '...  12 row(s) returned
20  16:47:36  SELECT DATE_FORMAT(transaction_date, '%M') AS Months, store_location, ROUND(SUM(trans...  18 row(s) returned
```

- **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  ;
```



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

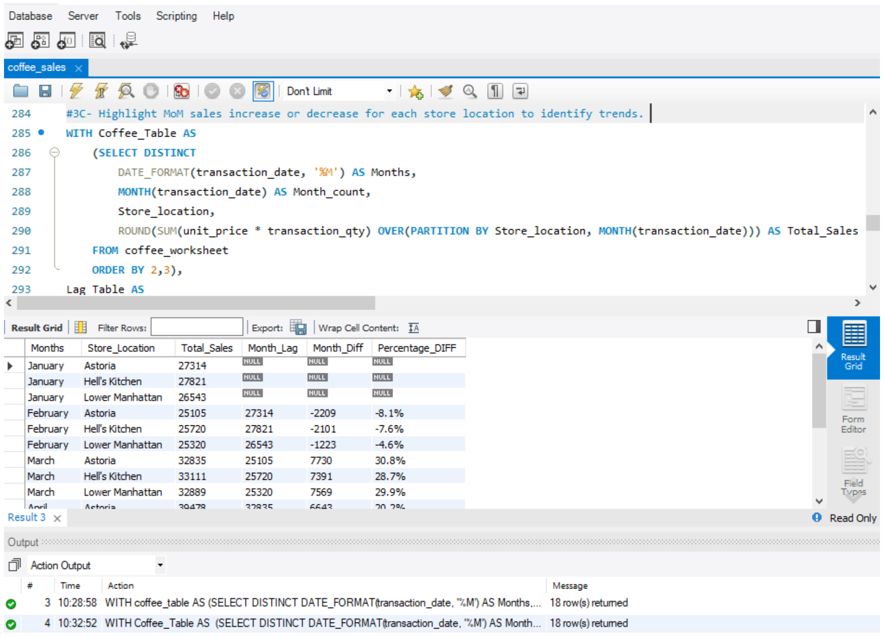
Months	store_location	total_sales	Monthly_lag	Monthly_Diff
January	Astoria	27314	NULL	NULL
January	Hell's Kitchen	27821	NULL	NULL
January	Lower Manhattan	26543	NULL	NULL
February	Astoria	25105	27314	-2209
February	Hell's Kitchen	25720	27821	-2101
February	Lower Manhattan	25320	26543	-1223
March	Astoria	32835	25105	7730
March	Hell's Kitchen	33111	25720	7391
March	Lower Manhattan	32889	25320	7569
April	Astoria	39478	32835	6643

```
#   Time      Action                                                                  Message
2   10:28:33  WITH coffee_table AS (SELECT DISTINCT DATE_FORMAT(transaction_date, '%M') AS Months,...  18 row(s) returned
3   10:28:58  WITH coffee_table AS (SELECT DISTINCT DATE_FORMAT(transaction_date, '%M') AS Months,...  18 row(s) returned
```

- **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;
```



```sql
#3C- Highlight MoM sales increase or decrease for each store location to identify trends.
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
```

Months	Store_Location	Total_Sales	Month_Lag	Month_Diff	Percentage_DIFF
January	Astoria	27314	NULL	NULL	NULL
January	Hell's Kitchen	27821	NULL	NULL	NULL
January	Lower Manhattan	26543	NULL	NULL	NULL
February	Astoria	25105	27314	-2209	-8.1%
February	Hell's Kitchen	25720	27821	-2101	-7.6%
February	Lower Manhattan	25320	26543	-1223	-4.6%
March	Astoria	32835	25105	7730	30.8%
March	Hell's Kitchen	33111	25720	7391	28.7%
March	Lower Manhattan	32889	25320	7569	29.9%

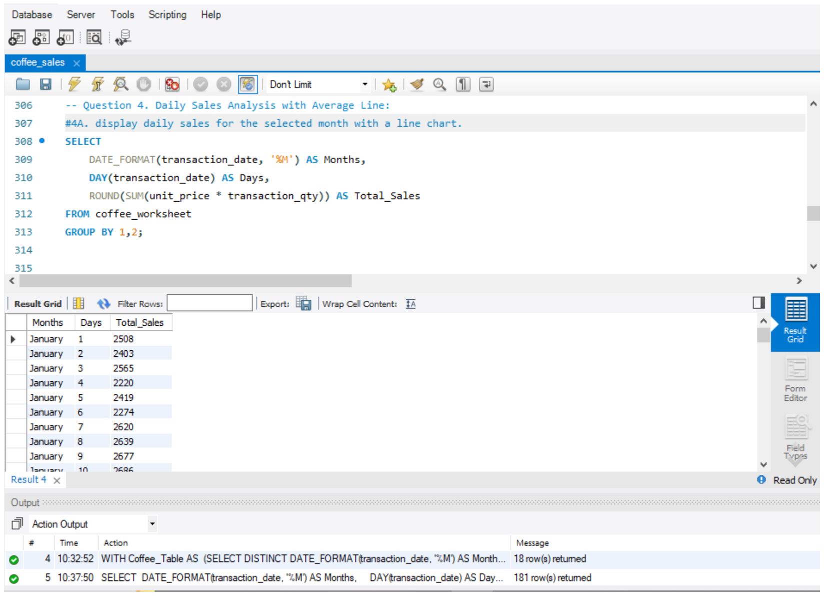
```
#  Time      Action                                                                        Message
3  10:28:58  WITH coffee_table AS (SELECT DISTINCT DATE_FORMAT(transaction_date, '%M') AS Months,...   18 row(s) returned
4  10:32:52  WITH Coffee_Table AS  (SELECT DISTINCT DATE_FORMAT(transaction_date, '%M') AS Month...   18 row(s) returned
```

4. Daily Sales Analysis with Average Line:

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

SQL

```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;
```



```
-- Question 4. Daily Sales Analysis with Average Line:
#4A. display daily sales for the selected month with a line chart.
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;
```

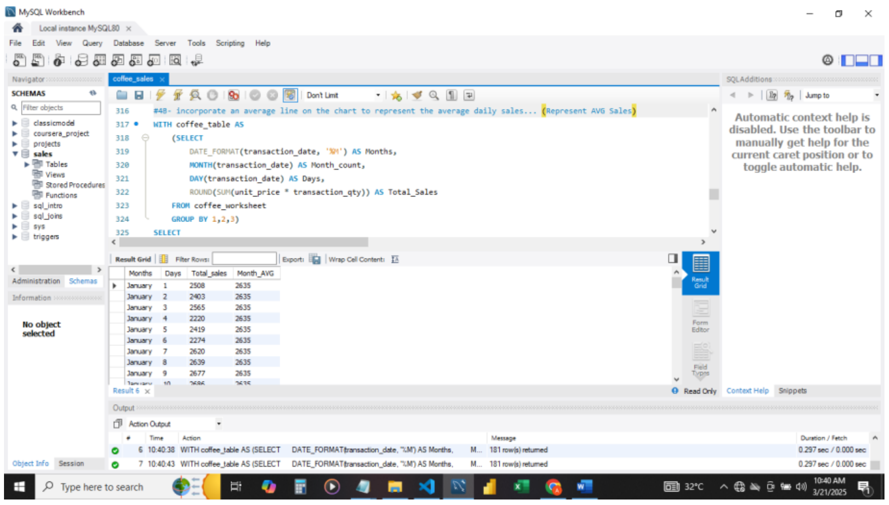
Months	Days	Total_Sales
January	1	2508
January	2	2403
January	3	2565
January	4	2220
January	5	2419
January	6	2274
January	7	2620
January	8	2639
January	9	2677
January	10	2686

```
#   Time      Action                                                                      Message
4   10:32:52  WITH Coffee_Table AS  (SELECT DISTINCT DATE_FORMAT(transaction_date, '%M') AS Month...   18 row(s) returned
5   10:37:50  SELECT  DATE_FORMAT(transaction_date, '%M') AS Months,    DAY(transaction_date) AS Day...   181 row(s) returned
```

- **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;
```



```
#48- incorporate an average line on the chart to represent the average daily sales... (Represent AVG Sales)
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
```

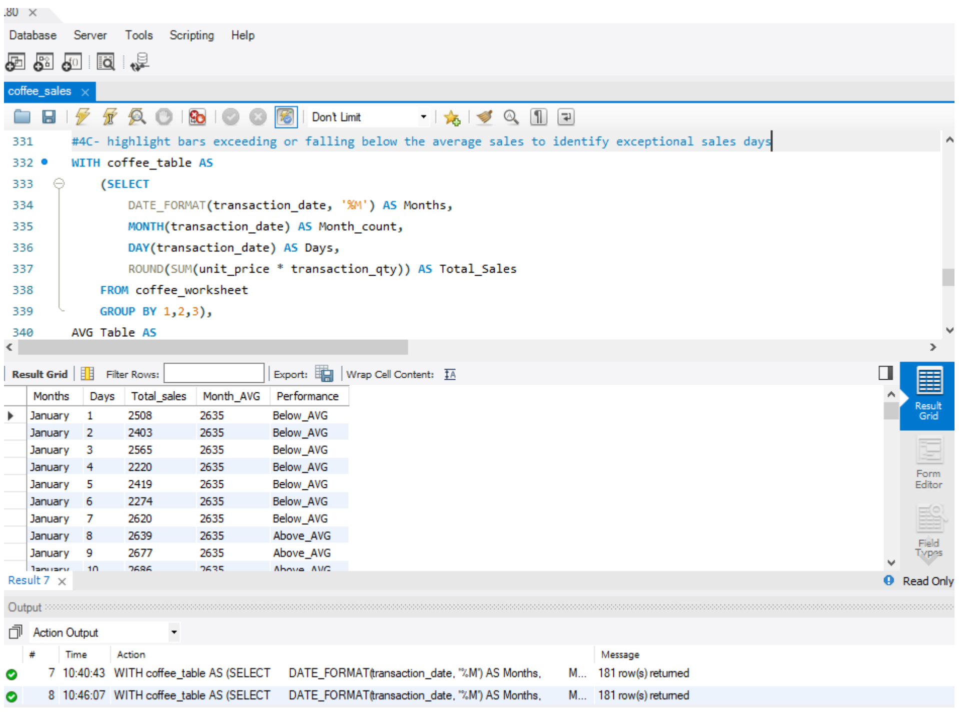
Months	Days	Total_sales	Month_AVG
January	1	2508	2635
January	2	2403	2635
January	3	2565	2635
January	4	2220	2635
January	5	2419	2635
January	6	2274	2635
January	7	2620	2635
January	8	2639	2635
January	9	2677	2635
January	10	2686	2635

```
#  Time      Action                                                                  Message               Duration / Fetch
6  10:40:38  WITH coffee_table AS (SELECT   DATE_FORMAT(transaction_date, '%M') AS Months,   M...  181 row(s) returned   0.297 sec / 0.000 sec
7  10:40:43  WITH coffee_table AS (SELECT   DATE_FORMAT(transaction_date, '%M') AS Months,   M...  181 row(s) returned   0.297 sec / 0.000 sec
```

- **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;
```



```
#4C- highlight bars exceeding or falling below the average sales to identify exceptional sales days
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
```

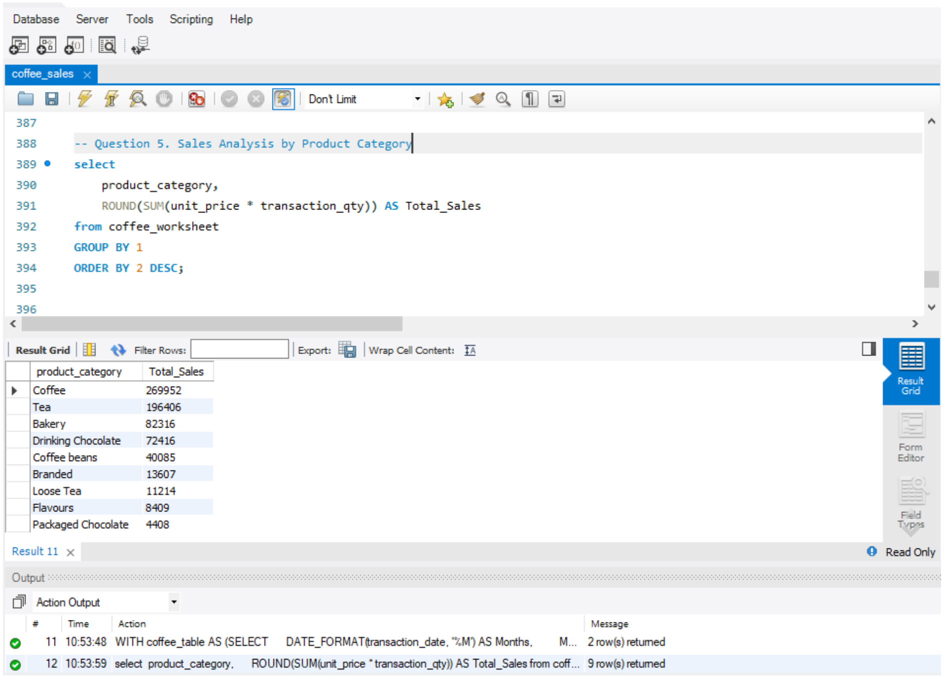
Months	Days	Total_sales	Month_AVG	Performance
January	1	2508	2635	Below_AVG
January	2	2403	2635	Below_AVG
January	3	2565	2635	Below_AVG
January	4	2220	2635	Below_AVG
January	5	2419	2635	Below_AVG
January	6	2274	2635	Below_AVG
January	7	2620	2635	Below_AVG
January	8	2639	2635	Above_AVG
January	9	2677	2635	Above_AVG
January	10	2686	2635	Above_AVG

```
#  Time      Action                                                                           Message
7  10:40:43  WITH coffee_table AS (SELECT   DATE_FORMAT(transaction_date, '%M') AS Months,   M...  181 row(s) returned
8  10:46:07  WITH coffee_table AS (SELECT   DATE_FORMAT(transaction_date, '%M') AS Months,   M...  181 row(s) returned
```

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;
```



```
-- Question 5. Sales Analysis by Product Category
select
    product_category,
    ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
from coffee_worksheet
GROUP BY 1
ORDER BY 2 DESC;
```

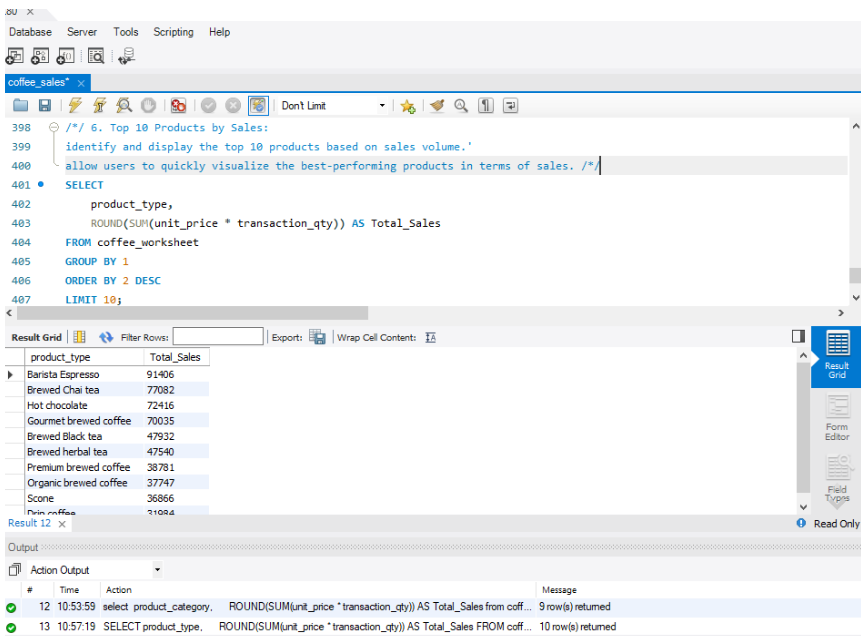
product_category	Total_Sales
Coffee	269952
Tea	196406
Bakery	82316
Drinking Chocolate	72416
Coffee beans	40085
Branded	13607
Loose Tea	11214
Flavours	8409
Packaged Chocolate	4408

```
#   Time      Action                                                                          Message
11  10:53:48  WITH coffee_table AS (SELECT    DATE_FORMAT(transaction_date, '%M') AS Months,    M...  2 row(s) returned
12  10:53:59  select  product_category,    ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales from coff...  9 row(s) returned
```

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;
```



```sql
/*/ 6. Top 10 Products by Sales:
identify and display the top 10 products based on sales volume.'
allow users to quickly visualize the best-performing products in terms of sales. /*/
SELECT
    product_type,
    ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
FROM coffee_worksheet
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10;
```

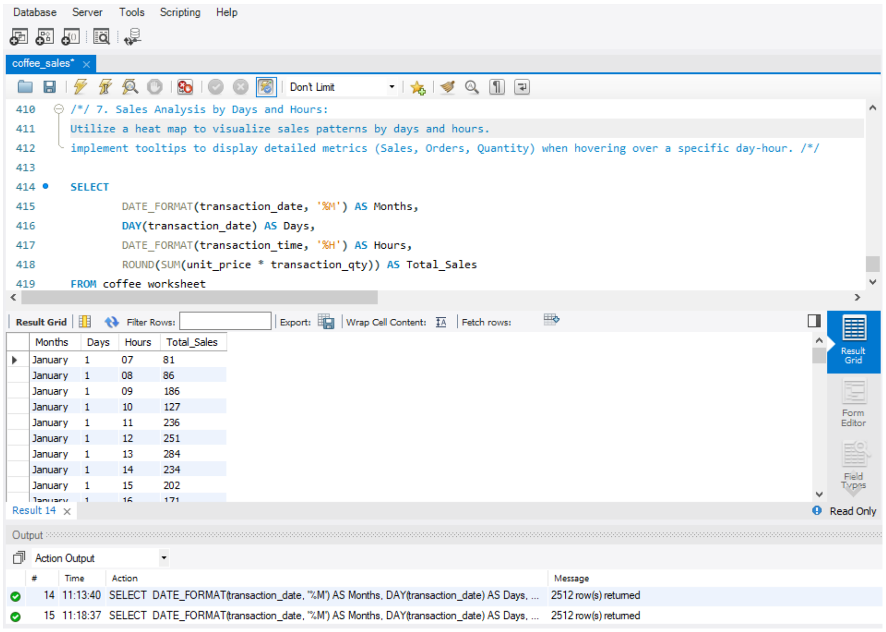
product_type	Total_Sales
Barista Espresso	91406
Brewed Chai tea	77082
Hot chocolate	72416
Gourmet brewed coffee	70035
Brewed Black tea	47932
Brewed herbal tea	47540
Premium brewed coffee	38781
Organic brewed coffee	37747
Scone	36866
Drip coffee	31984

```
#   Time      Action                                                                                   Message
12  10:53:59  select  product_category,   ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales from coff...  9 row(s) returned
13  10:57:19  SELECT product_type,   ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales FROM coff...   10 row(s) returned
```

7. Sales Analysis by Days and Hours

sql

```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;
```



```sql
/*/ 7. Sales Analysis by Days and Hours:
Utilize a heat map to visualize sales patterns by days and hours.
implement tooltips to display detailed metrics (Sales, Orders, Quantity) when hovering over a specific day-hour. /*/

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
```

```
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
January  1     16     171
```

```
#   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
```