

# 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;
```

Database Server Tools Scripting Help

Query 1 coffee\_sales x

15 unit\_price DOUBLE,  
 16 product\_category TEXT,  
 17 product\_type TEXT,  
 18 product\_detail TEXT);  
 19  
 20 -- verify table  
 21 • SELECT \* FROM coffee\_sales;  
 22 • DESCRIBE coffee\_sales;

Result Grid Filter Rows: Export: Wrap Cell Content: [Read Only](#)

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	

Result 1 x

Output

Action Output

#	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;
```

The screenshot shows a database management tool interface. The top menu bar includes 'Database', 'Server', 'Tools', 'Scripting', and 'Help'. Below the menu is a toolbar with various icons. The main window displays a SQL query in a text editor:

```
27 INTO TABLE coffee_sales
28 FIELDS TERMINATED BY ','
29 ENCLOSED BY '"'
30 LINES TERMINATED BY '\n'
31 IGNORE 1 ROWS;
32
33 -- verify import
34 • SELECT * FROM coffee_sales;
```

Below the query editor is a 'Result Grid' showing the results of the query. The grid has columns: transaction\_id, transaction\_date, transaction\_time, transaction\_qty, store\_id, store\_location, product\_id, unit\_price, product\_category, and product\_type. The results are as follows:

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 brewer
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 brewer
9	2023-01-01	7:39:13	1	5	Lower Manhattan	39	4.25	Coffee	Barista Espresso
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

Below the result grid is an 'Output' section showing the execution of the query. It includes a table with columns: #, Time, Action, and Message.

#	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;
```

The screenshot displays a database management interface with a menu bar (Database, Server, Tools, Scripting, Help) and a toolbar. The main window is titled 'Query 1' and contains the following SQL script:

```
-- Copy dataset into coffee worksheet table  
45 • INSERT INTO coffee_worksheet  
46   SELECT * FROM coffee_sales;  
47  
48 -- verify dataset  
49 • SELECT * FROM coffee_worksheet;  
50  
51
```

Below the script, the 'Result Grid' shows the data from the first query. It has columns: transaction\_id, transaction\_date, transaction\_time, transaction\_qty, store\_id, store\_location, product\_id, unit\_price, product\_category, and product\_type. The data is as follows:

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 brewed
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 brewed
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

At the bottom, the 'Output' pane shows the execution results of the queries:

#	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;
```

## **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,
```

```

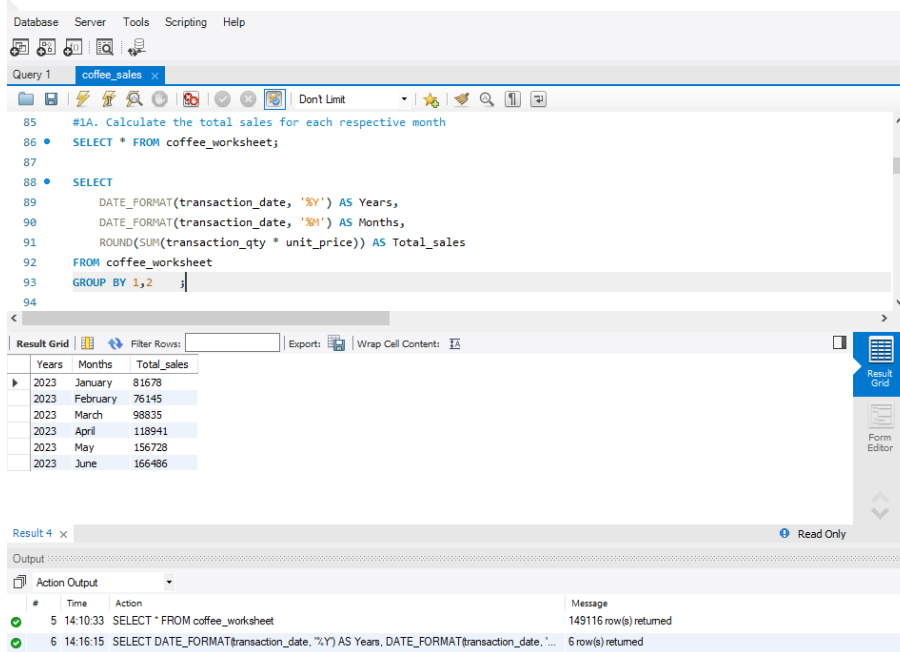
DATE_FORMAT(transaction_date, '%M') AS Months,

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 ;

```

The screenshot shows a database IDE with a query window and a results grid. The query is for determining the month-on-month increase or decrease in sales. The results grid shows data for the years 2023, months January through June, with columns for Years, Months, Total\_sales, and Monthly\_DIFF.

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

- **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,

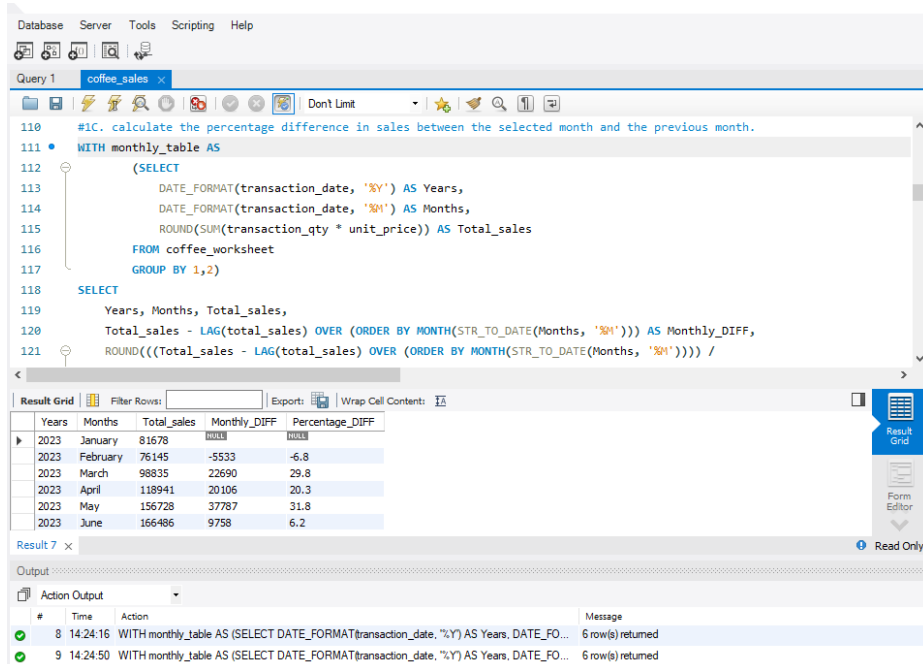
Total_sales - LAG(total_sales) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M')))) AS
Monthly_DIFF,

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 ;

```



The screenshot shows a database query editor with a query named 'coffee\_sales'. The query calculates the percentage difference in sales between the selected month and the previous month. The results are displayed in a table with columns: Years, Months, Total\_sales, Monthly\_DIFF, and Percentage\_DIFF.

Years	Months	Total_sales	Monthly_DIFF	Percentage_DIFF
2023	January	81678	NA	NA
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

## 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;

The screenshot shows a SQL IDE interface with a query editor and a results grid. The query editor contains the following SQL code:

```
124
125
126 -- Question 2. Total Order Analysis:
127 #2A. calculate the total number of orders for each respective month.
128 • SELECT
129     DATE_FORMAT(transaction_date, '%Y') AS Years,
130     DATE_FORMAT(transaction_date, '%M') AS Months,
131     COUNT(*) AS Total_Orders
132 FROM coffee_worksheet
133 GROUP BY 1,2;
134
135
```

The results grid displays the following data:

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

The output pane shows the following messages:

```
9 14:24:50 WITH monthly_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)
10 14:29:48 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; 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;

The screenshot shows a SQL IDE interface with a query editor and a results grid. The query is as follows:

```
136 #2B- determine the month-on-month increase or decrease in the number of orders.
137 WITH Order_table AS
138 (SELECT
139     DATE_FORMAT(transaction_date, '%Y') AS Years,
140     DATE_FORMAT(transaction_date, '%M') AS Months,
141     COUNT(*) AS Total_Orders
142 FROM coffee_worksheet
143 GROUP BY 1,2)
144 SELECT
145     Years, Months, Total_Orders,
146     Total_orders - LAG(total_orders) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M')))) AS Monthly_DIFF
147 FROM Order_table;
```

The results grid shows the following data:

Years	Months	Total_Orders	Monthly_DIFF
2023	January	17314	1000
2023	February	16399	-955
2023	March	21229	4870
2023	April	25335	4106
2023	May	33527	8192
2023	June	35352	1825

The Action Output pane shows the following messages:

#	Time	Action	Message
10	14:29:48	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)	6 row(s) returned
11	14:34:18	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;	6 row(s) returned

- **2C. percentage difference on the selected 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;
```

The screenshot shows a SQL IDE interface with a query editor and a results grid. The query is as follows:

```

150 #2C- calculate the % difference in the number of orders between the selected month and the previous month.
151 WITH Order_table AS
152 (SELECT
153     DATE_FORMAT(transaction_date, '%Y') AS Years,
154     DATE_FORMAT(transaction_date, '%M') AS Months,
155     COUNT(*) AS Total_Orders
156 FROM coffee_worksheet
157 GROUP BY 1,2)
158 SELECT
159     Years, Months, Total_Orders,
160     Total_orders - LAG(total_orders) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M')))) AS Monthly_DIFF,
161     ROUND(((total_orders - LAG(total_orders) OVER (ORDER BY MONTH(STR_TO_DATE(Months, '%M'))))) /

```

The results grid shows the following data:

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

The bottom section of the IDE shows the action output with two messages:

```

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;
```

The screenshot shows a SQL IDE interface with a menu bar (Database, Server, Tools, Scripting, Help) and a toolbar. The main window displays a SQL query in a script editor. The query is as follows:

```

166
167 -- Question 3. Total Quantity Sold Analysis:
168 #3A- calculate the total quantity sold for each respective month.
169 • SELECT
170     DATE_FORMAT(transaction_date, '%Y') AS Years,
171     DATE_FORMAT(transaction_date, '%M') AS Months,
172     SUM(transaction_qty) AS Total_Orders
173 FROM coffee_worksheet
174 GROUP BY 1,2
175
176
177 #3B- determine the month-on-month increase or decrease in the total quantity sold.

```

Below the script editor, the 'Result Grid' is visible, showing the results of the query. The results are as follows:

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

At the bottom, the 'Output' pane shows the execution log with two entries:

```

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, '%M') AS Months, SUM(transaction_qty) AS Total_Orders FROM coffee_worksheet GROUP BY 1,2 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 ;

The screenshot shows a SQL IDE with a query editor and a results grid. The query is as follows:

```

177 #38- determine the month-on-month increase or decrease in the total quantity sold.
178 WITH Monthly_QTY AS
179 (SELECT
180     DATE_FORMAT(transaction_date, '%Y') AS Years,
181     DATE_FORMAT(transaction_date, '%M') AS Months,
182     SUM(transaction_qty) AS Total_QTY
183 FROM coffee_worksheet
184 GROUP BY 1,2)
185 SELECT
186     Years, Months, Total_QTY,
187     Total_QTY - LAG(Total_QTY) OVER (ORDER BY MONTH(STR_TO_DATE(months, '%M')))) AS Monthly_DIFF
188 FROM Monthly_qty ;

```

The results grid shows the following data:

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

The output pane shows the execution of the query, indicating that 6 rows were 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 ;

Query 1: coffee\_sales

```

191 #3C- calculate the % difference in the total quantity sold between the selected month and the previous month.
192 WITH Monthly_QTY AS
193 (SELECT
194     DATE_FORMAT(transaction_date, '%Y') AS Years,
195     DATE_FORMAT(transaction_date, '%M') AS Months,
196     SUM(transaction_qty) AS Total_QTY
197 FROM coffee_worksheet
198 GROUP BY 1,2)
199 SELECT
200     Years, Months, Total_QTY,
201     Total_QTY - LAG(Total_QTY) OVER (ORDER BY MONTH(STR_TO_DATE(months, '%M'))) AS Monthly_DIFF,
202     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

Result 14 x

Output

#	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

## 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,

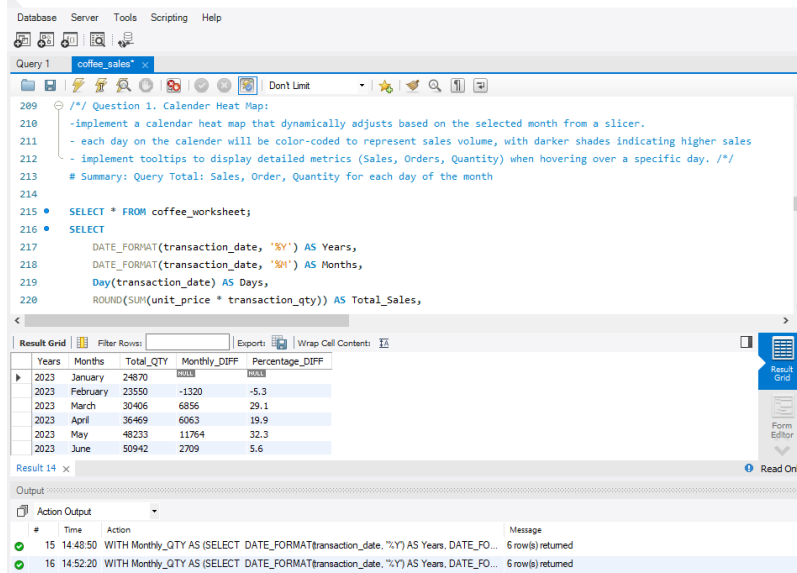
    COUNT(*) AS Total_Orders,

    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;
```

Query 1 coffee\_sales

```

227 -- Question 2. Sales Analysis by Weekdays and Weekends:
228 -- segment sales data into weekdays and weekends to analyze performance variations.
229 -- provide insights into whether sales patterns differ significantly between weekdays and weekends.
230 -- SUPPORT: categorize sales into weekdays and weekends Per Each Month
231
232 SELECT
233     DATE_FORMAT(transaction_date, '%Y') AS Years,
234     DATE_FORMAT(transaction_date, '%M') AS Months,
235     CASE
236         WHEN WEEKDAY(transaction_date) IN (1,7) THEN 'weekends'
237         ELSE 'weekdays'
238     END AS Day_Type,
239     Total_sales
240 FROM coffee_sales
241 GROUP BY Years, Months, Day_Type
242 ORDER BY Years, Months, Day_Type

```

Years	Months	Day_Type	Total_sales
2023	January	Weekdays	67938
2023	January	Weekends	13740
2023	February	Weekdays	65942
2023	February	Weekends	10003
2023	March	Weekdays	86716
2023	March	Weekends	12119

Result 17

Output

#	Time	Action	Message
18	16:37:29	SELECT CASE WHEN WEEKDAY(transaction_date) IN (1,7) THEN 'Weekends' Else 'Weekdays' END AS Day_Type, 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_sales GROUP BY Years, Months, Day_Type ORDER BY Years, Months, Day_Type	2 row(s) returned
19	16:38:05	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_sales GROUP BY Years, Months, Day_Type ORDER BY Years, Months, Day_Type	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;

Database Server Tools Scripting Help

Query 1 coffee\_sales

```

253
254 -- Question 3. Sales Analysis by Store Location:
255 -- #3A. visualize sales data by different store locations.
256 SELECT
257     DATE_FORMAT(transaction_date, '%M') AS Months,
258     store_location,
259     ROUND(SUM(transaction_qty * unit_price)) AS Total_sales
260 FROM coffee_worksheet
261 GROUP BY 1,2;
262
263

```

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

Result 18

Output

#	Time	Action	Message
19	16:38:05	SELECT DATE_FORMAT(transaction_date, '%Y') AS Years, DATE_FORMAT(transaction_date, '%M') AS Months, store_location, ROUND(SUM(transaction_qty * unit_price)) AS Total_sales FROM coffee_worksheet GROUP BY Years, Months, store_location ORDER BY Years, Months, store_location	12 row(s) returned
20	16:47:36	SELECT DATE_FORMAT(transaction_date, '%M') AS Months, store_location, ROUND(SUM(transaction_qty * unit_price)) AS Total_sales FROM coffee_worksheet GROUP BY Months, store_location ORDER BY Months, store_location	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 ;

```

80 X

Database Server Tools Scripting Help

coffee\_sales

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

```

264 WITH coffee_table AS
265 (SELECT DISTINCT
266 DATE_FORMAT(transaction_date, '%M') AS Months,
267 MONTH(transaction_date) AS Month_Count,
268 Store_Location,
269 ROUND(SUM(unit_Price * transaction_qty) OVER (PARTITION BY Store_Location, MONTH(transaction_date))) AS Total_Sales
270 FROM coffee_worksheet
271 ORDER BY 2,3),
272 Lag_table AS (SELECT
273 Months, Month_count, Store_location, Total_Sales,

```

Result Grid

Months	store_location	total_sales	Monthly_Lag	Monthly_Diff
January	Astoria	27314	0000	0000
January	Heff's Kitchen	27821	0000	0000
January	Lower Manhattan	26543	0000	0000
February	Astoria	25105	27314	-2209
February	Heff's Kitchen	25720	27821	-2101
February	Lower Manhattan	25320	26543	-1223
March	Astoria	30835	25105	7730
March	Heff's Kitchen	33111	25720	7391
March	Lower Manhattan	32889	25320	7569
Grand Total	Astoria	104790	7730	6647

Result 2 X

Output

Action Output

#	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;
```

Database Server Tools Scripting Help

coffee\_sales

284 #3C- Highlight Month sales increase or decrease for each store location to identify trends.

285 WITH Coffee\_Table AS

286 (SELECT DISTINCT

287 DATE\_FORMAT(transaction\_date, '%M') AS Months,

288 MONTH(transaction\_date) AS Month\_count,

289 Store\_location,

290 ROUND(SUM(unit\_price \* transaction\_qty) OVER(PARTITION BY Store\_location, MONTH(transaction\_date))) AS Total\_Sales

291 FROM coffee\_worksheet

292 ORDER BY 2,3),

293 Lag Table AS

Result Grid

Months	Store_Location	Total_Sales	Month_Lag	Month_Diff	Percentage_Diff
January	Astoria	27314			
January	Hell's Kitchen	27821			
January	Lower Manhattan	26543			
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%
April	Astoria	30478	32835	-6643	-20.5%

Output

Action Output

#	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

SELECT

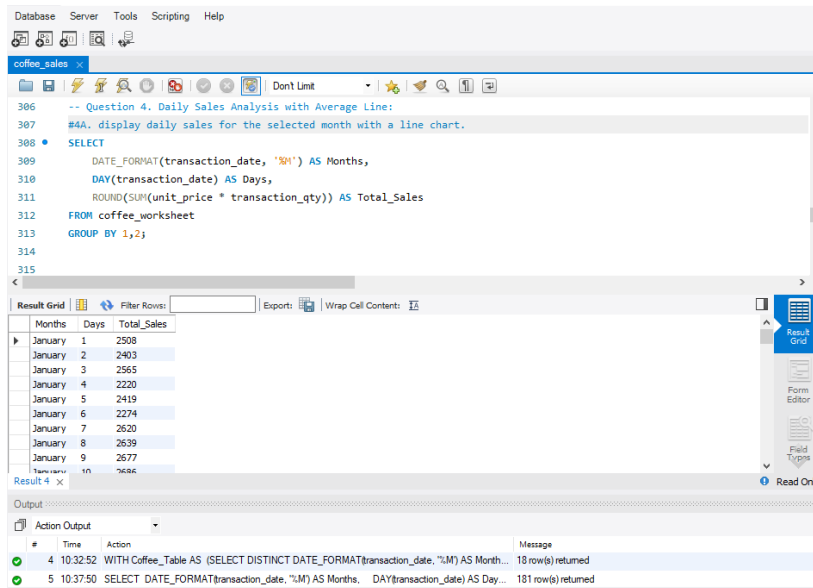
DATE\_FORMAT(transaction\_date, '%M') AS Months,

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

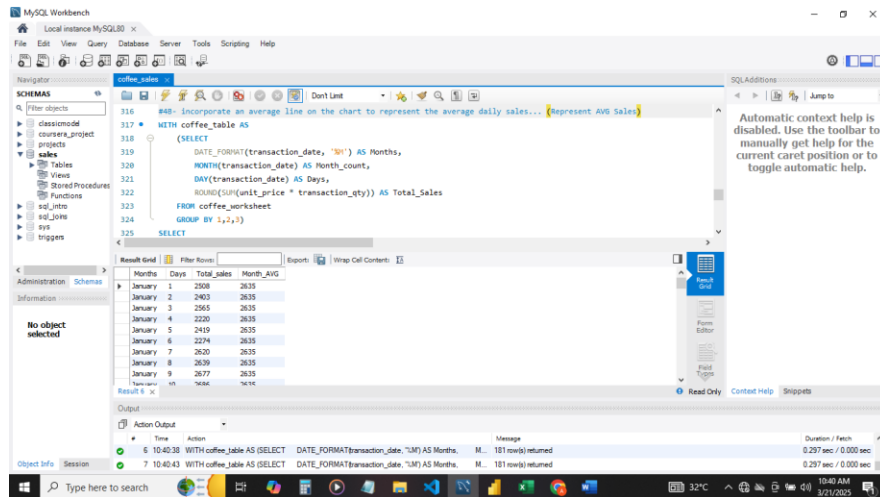
GROUP BY 1,2,3)

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;

The screenshot shows a SQL IDE window with a query editor and a results pane. The query editor contains the following SQL code:

```
331 #4C- highlight bars exceeding or falling below the average sales to identify exceptional sales days
332 WITH coffee_table AS
333 (SELECT
334     DATE_FORMAT(transaction_date, '%M') AS Months,
335     MONTH(transaction_date) AS Month_count,
336     DAY(transaction_date) AS Days,
337     ROUND(SUM(unit_price * transaction_qty)) AS Total_Sales
338 FROM coffee_worksheet
339 GROUP BY 1,2,3),
340 AVG_Table AS
```

The results pane displays a table with the following data:

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

The Action Output pane shows the following messages:

```
# 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;

The screenshot shows a database IDE with a menu bar (Database, Server, Tools, Scripting, Help) and a toolbar. The main window is titled 'coffee\_sales' and contains a SQL query. Below the query editor is a 'Result Grid' showing the results of the query. The results are as follows:

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

Below the result grid is an 'Output' section with an 'Action Output' dropdown. It shows two messages:

- 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;

The screenshot shows a database IDE with a SQL query editor and a results grid. The query is a SQL comment followed by a SELECT statement to find the top 10 products by sales volume.

```

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

```

The results grid displays the following data:

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
Irish coffee	31084

The output pane shows two messages:

```

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

```

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;

```



DatabaseServerToolsScriptingHelp

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 Contents: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
	January	1	16	171

Result 14 ×Read Only

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