**MY SQL QUERIES**

**COFFEE SHOP SALES PROJECT**

**CONVERT DATE (transaction\_date) COLUMN TO PROPER DATE FORMAT**

UPDATE coffee\_shop\_sales

SET transaction\_date = STR\_TO\_DATE(transaction\_date, '%d-%m-%Y');

**ALTER DATE (transaction\_date) COLUMN TO DATE DATA TYPE**

ALTER TABLE coffee\_shop\_sales

MODIFY COLUMN transaction\_date DATE;

**CONVERT TIME (transaction\_time)  COLUMN TO PROPER DATE FORMAT**

UPDATE coffee\_shop\_sales

SET transaction\_time = STR\_TO\_DATE(transaction\_time, '%H:%i:%s');

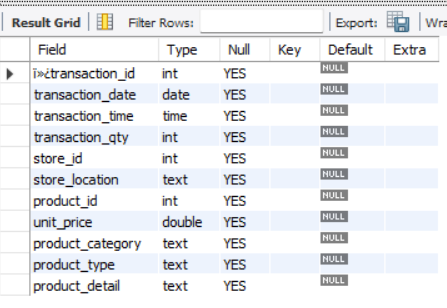
**ALTER TIME (transaction\_time) COLUMN TO DATE DATA TYPE**

ALTER TABLE coffee\_shop\_sales

MODIFY COLUMN transaction\_time TIME;

**DATA TYPES OF DIFFERENT COLUMNS**

DESCRIBE coffee\_shop\_sales;

****

**CHANGE COLUMN NAME `ï»¿transaction\_id` to transaction\_id**

ALTER TABLE coffee\_shop\_sales

CHANGE COLUMN `ï»¿transaction\_id` transaction\_id INT;

**TOTAL SALES**

SELECT ROUND(SUM(unit\_price \* transaction\_qty)) as Total\_Sales

FROM coffee\_shop\_sales

WHERE MONTH(transaction\_date) = 5 -- for month of (CM-May)

A screenshot of a computer

AI-generated content may be incorrect.

**TOTAL SALES KPI - MOM DIFFERENCE AND MOM GROWTH**

SELECT

    MONTH(transaction\_date) AS month,

    ROUND(SUM(unit\_price \* transaction\_qty)) AS total\_sales,

    (SUM(unit\_price \* transaction\_qty) - LAG(SUM(unit\_price \* transaction\_qty), 1)

    OVER (ORDER BY MONTH(transaction\_date))) / LAG(SUM(unit\_price \* transaction\_qty), 1)

    OVER (ORDER BY MONTH(transaction\_date)) \* 100 AS mom\_increase\_percentage

FROM

    coffee\_shop\_sales

WHERE

    MONTH(transaction\_date) IN (4, 5) -- for months of April and May

GROUP BY

    MONTH(transaction\_date)

ORDER BY

    MONTH(transaction\_date);

A screenshot of a computer

AI-generated content may be incorrect.

**Explaination**

**SELECT clause:**

* MONTH(transaction\_date) AS month: Extracts the month from the transaction\_date column and renames it as month.
* ROUND(SUM(unit\_price \* transaction\_qty)) AS total\_sales: Calculates the total sales by multiplying unit\_price and transaction\_qty, then sums the result for each month. The ROUND function rounds the result to the nearest integer.
* (SUM(unit\_price \* transaction\_qty) - LAG(SUM(unit\_price \* transaction\_qty), 1) OVER (ORDER BY MONTH(transaction\_date))) / LAG(SUM(unit\_price \* transaction\_qty), 1) OVER (ORDER BY MONTH(transaction\_date)) \* 100 AS mom\_increase\_percentage with the functions used:
  + SUM(unit\_price \* transaction\_qty): This calculates the total sales for the current month. It multiplies the unit\_price by the transaction\_qty for each transaction and then sums up these values.
  + LAG(SUM(unit\_price \* transaction\_qty), 1) OVER (ORDER BY MONTH(transaction\_date)): This function retrieves the value of the total sales for the previous month. It uses the LAG window function to get the value of the SUM(unit\_price \* transaction\_qty) from the previous row (previous month) ordered by the transaction\_date.
  + (SUM(unit\_price \* transaction\_qty) - LAG(SUM(unit\_price \* transaction\_qty), 1) OVER (ORDER BY MONTH(transaction\_date))): This part calculates the difference between the total sales of the current month and the total sales of the previous month.
  + LAG(SUM(unit\_price \* transaction\_qty), 1) OVER (ORDER BY MONTH(transaction\_date)): This function retrieves the value of the total sales for the previous month again. It's used in the denominator to calculate the percentage increase.
  + (SUM(unit\_price \* transaction\_qty) - LAG(SUM(unit\_price \* transaction\_qty), 1) OVER (ORDER BY MONTH(transaction\_date))) / LAG(SUM(unit\_price \* transaction\_qty), 1) OVER (ORDER BY MONTH(transaction\_date)): This calculates the ratio of the difference in sales between the current and previous months to the total sales of the previous month. It represents the percentage increase or decrease in sales compared to the previous month.
  + 100: This part multiplies the ratio by 100 to convert it to a percentage.
* FROM clause:

coffee\_shop\_sales: Specifies the table from which data is being selected.

* WHERE clause:

MONTH(transaction\_date) IN (4, 5): Filters the data to include only transactions from April and May.

* GROUP BY clause:

MONTH(transaction\_date): Groups the results by month.

* ORDER BY clause:

MONTH(transaction\_date): Orders the results by month.

**TOTAL ORDERS**

SELECT COUNT(transaction\_id) as Total\_Orders

FROM coffee\_shop\_sales

WHERE MONTH (transaction\_date)= 5 -- for month of (CM-May)

A screenshot of a computer

AI-generated content may be incorrect.

**TOTAL ORDERS KPI - MOM DIFFERENCE AND MOM GROWTH**

SELECT

    MONTH(transaction\_date) AS month,

    ROUND(COUNT(transaction\_id)) AS total\_orders,

    (COUNT(transaction\_id) - LAG(COUNT(transaction\_id), 1)

    OVER (ORDER BY MONTH(transaction\_date))) / LAG(COUNT(transaction\_id), 1)

    OVER (ORDER BY MONTH(transaction\_date)) \* 100 AS mom\_increase\_percentage

FROM

    coffee\_shop\_sales

WHERE

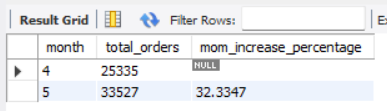
    MONTH(transaction\_date) IN (4, 5) -- for April and May

GROUP BY

    MONTH(transaction\_date)

ORDER BY

    MONTH(transaction\_date);



**TOTAL QUANTITY SOLD**

SELECT SUM(transaction\_qty) as Total\_Quantity\_Sold

FROM coffee\_shop\_sales

WHERE MONTH(transaction\_date) = 5 -- for month of (CM-May)

A screenshot of a computer

AI-generated content may be incorrect.

**TOTAL QUANTITY SOLD KPI - MOM DIFFERENCE AND MOM GROWTH**

SELECT

    MONTH(transaction\_date) AS month,

    ROUND(SUM(transaction\_qty)) AS total\_quantity\_sold,

    (SUM(transaction\_qty) - LAG(SUM(transaction\_qty), 1)

    OVER (ORDER BY MONTH(transaction\_date))) / LAG(SUM(transaction\_qty), 1)

    OVER (ORDER BY MONTH(transaction\_date)) \* 100 AS mom\_increase\_percentage

FROM

    coffee\_shop\_sales

WHERE

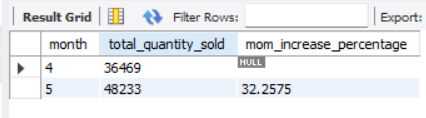
    MONTH(transaction\_date) IN (4, 5)   -- for April and May

GROUP BY

    MONTH(transaction\_date)

ORDER BY

    MONTH(transaction\_date);



**CALENDAR TABLE – DAILY SALES, QUANTITY and TOTAL ORDERS**

SELECT

    SUM(unit\_price \* transaction\_qty) AS total\_sales,

    SUM(transaction\_qty) AS total\_quantity\_sold,

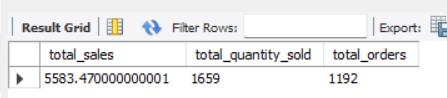
    COUNT(transaction\_id) AS total\_orders

FROM

    coffee\_shop\_sales

WHERE

    transaction\_date = '2023-05-18'; --For 18 May 2023



***If you want to get exact Rounded off values then use below query to get the result:***

SELECT

    CONCAT(ROUND(SUM(unit\_price \* transaction\_qty) / 1000, 1),'K') AS total\_sales,

    CONCAT(ROUND(COUNT(transaction\_id) / 1000, 1),'K') AS total\_orders,

    CONCAT(ROUND(SUM(transaction\_qty) / 1000, 1),'K') AS total\_quantity\_sold

FROM

    coffee\_shop\_sales

WHERE

    transaction\_date = '2023-05-18'; --For 18 May 2023

A screenshot of a computer

AI-generated content may be incorrect.

**SALES TREND OVER PERIOD**

SELECT AVG(total\_sales) AS average\_sales

FROM (

    SELECT

        SUM(unit\_price \* transaction\_qty) AS total\_sales

    FROM

        coffee\_shop\_sales

WHERE

        MONTH(transaction\_date) = 5  -- Filter for May

    GROUP BY

        transaction\_date

) AS internal\_query;

***Query Explanation:***

* This inner subquery calculates the total sales (unit\_price \* transaction\_qty) for each date in May. It filters the data to include only transactions that occurred in May by using the MONTH() function to extract the month from the transaction\_date column and filtering for May (month number 5).
* The GROUP BY clause groups the data by transaction\_date, ensuring that the total sales are aggregated for each individual date in May.
* The outer query calculates the average of the total sales over all dates in May. It references the result of the inner subquery as a derived table named internal\_query.
* The AVG() function calculates the average of the total\_sales column from the derived table, giving us the average sales for May.

A screenshot of a computer

AI-generated content may be incorrect.

**DAILY SALES FOR MONTH SELECTED**

SELECT

    DAY(transaction\_date) AS day\_of\_month,

    ROUND(SUM(unit\_price \* transaction\_qty),1) AS total\_sales

FROM

    coffee\_shop\_sales

WHERE

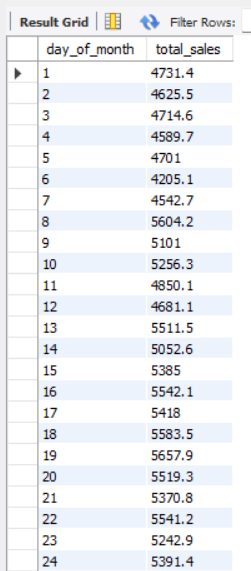
    MONTH(transaction\_date) = 5  -- Filter for May

GROUP BY

    DAY(transaction\_date)

ORDER BY

    DAY(transaction\_date);

                         A screenshot of a data

AI-generated content may be incorrect.

***COMPARING DAILY SALES WITH AVERAGE SALES – IF GREATER THAN “ABOVE AVERAGE” and LESSER THAN “BELOW AVERAGE”***

SELECT

    day\_of\_month,

    CASE

        WHEN total\_sales > avg\_sales THEN 'Above Average'

        WHEN total\_sales < avg\_sales THEN 'Below Average'

        ELSE 'Average'

    END AS sales\_status,

    total\_sales

FROM (

    SELECT

        DAY(transaction\_date) AS day\_of\_month,

        SUM(unit\_price \* transaction\_qty) AS total\_sales,

        AVG(SUM(unit\_price \* transaction\_qty)) OVER () AS avg\_sales

    FROM

        coffee\_shop\_sales

    WHERE

        MONTH(transaction\_date) = 5  -- Filter for May

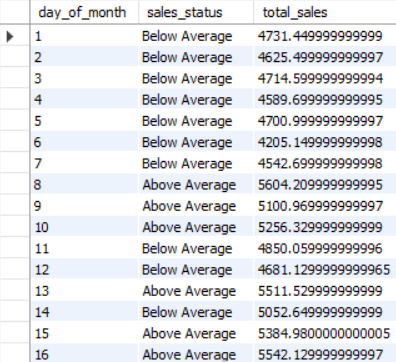
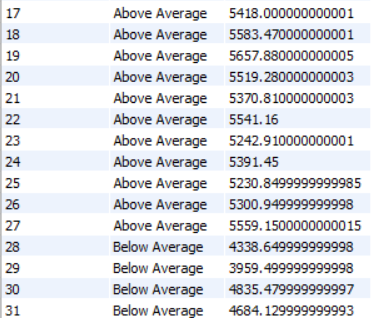
    GROUP BY

        DAY(transaction\_date)

) AS sales\_data

ORDER BY

    day\_of\_month;

**SALES BY WEEKDAY / WEEKEND:**

SELECT

    CASE

        WHEN DAYOFWEEK(transaction\_date) IN (1, 7) THEN 'Weekends'

        ELSE 'Weekdays'

    END AS day\_type,

    ROUND(SUM(unit\_price \* transaction\_qty),2) AS total\_sales

FROM

    coffee\_shop\_sales

WHERE

    MONTH(transaction\_date) = 5  -- Filter for May

GROUP BY

    CASE

        WHEN DAYOFWEEK(transaction\_date) IN (1, 7) THEN 'Weekends'

        ELSE 'Weekdays'

    END;

A screenshot of a computer

AI-generated content may be incorrect.

**SALES BY STORE LOCATION**

SELECT

store\_location,

SUM(unit\_price \* transaction\_qty) as Total\_Sales

FROM coffee\_shop\_sales

WHERE

MONTH(transaction\_date) =5

GROUP BY store\_location

ORDER BY SUM(unit\_price \* transaction\_qty) DESC

A screenshot of a computer

AI-generated content may be incorrect.

**SALES BY PRODUCT CATEGORY**

SELECT

product\_category,

ROUND(SUM(unit\_price \* transaction\_qty),1) as Total\_Sales

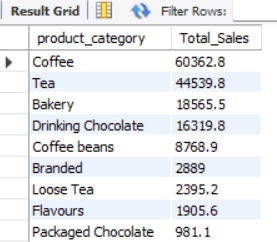
FROM coffee\_shop\_sales

WHERE

MONTH(transaction\_date) = 5

GROUP BY product\_category

ORDER BY SUM(unit\_price \* transaction\_qty) DESC



**SALES BY PRODUCTS (TOP 10)**

SELECT

product\_type,

ROUND(SUM(unit\_price \* transaction\_qty),1) as Total\_Sales

FROM coffee\_shop\_sales

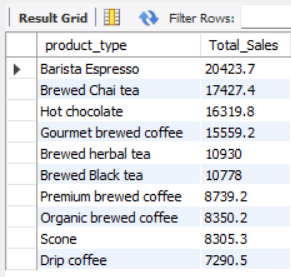
WHERE

MONTH(transaction\_date) = 5

GROUP BY product\_type

ORDER BY SUM(unit\_price \* transaction\_qty) DESC

LIMIT 10



**SALES BY DAY | HOUR**

SELECT

    ROUND(SUM(unit\_price \* transaction\_qty)) AS Total\_Sales,

    SUM(transaction\_qty) AS Total\_Quantity,

    COUNT(\*) AS Total\_Orders

FROM

    coffee\_shop\_sales

WHERE

    DAYOFWEEK(transaction\_date) = 3 -- Filter for Tuesday (1 is Sunday, 2 is Monday, ..., 7 is Saturday)

    AND HOUR(transaction\_time) = 8 -- Filter for hour number 8

    AND MONTH(transaction\_date) = 5; -- Filter for May (month number 5)

A screenshot of a computer

AI-generated content may be incorrect.

***TO GET SALES FROM MONDAY TO SUNDAY FOR MONTH OF MAY***

SELECT

    CASE

        WHEN DAYOFWEEK(transaction\_date) = 2 THEN 'Monday'

        WHEN DAYOFWEEK(transaction\_date) = 3 THEN 'Tuesday'

        WHEN DAYOFWEEK(transaction\_date) = 4 THEN 'Wednesday'

        WHEN DAYOFWEEK(transaction\_date) = 5 THEN 'Thursday'

        WHEN DAYOFWEEK(transaction\_date) = 6 THEN 'Friday'

        WHEN DAYOFWEEK(transaction\_date) = 7 THEN 'Saturday'

        ELSE 'Sunday'

    END AS Day\_of\_Week,

    ROUND(SUM(unit\_price \* transaction\_qty)) AS Total\_Sales

FROM

    coffee\_shop\_sales

WHERE

    MONTH(transaction\_date) = 5 -- Filter for May (month number 5)

GROUP BY

    CASE

        WHEN DAYOFWEEK(transaction\_date) = 2 THEN 'Monday'

        WHEN DAYOFWEEK(transaction\_date) = 3 THEN 'Tuesday'

        WHEN DAYOFWEEK(transaction\_date) = 4 THEN 'Wednesday'

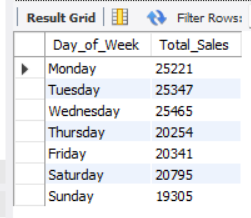
        WHEN DAYOFWEEK(transaction\_date) = 5 THEN 'Thursday'

        WHEN DAYOFWEEK(transaction\_date) = 6 THEN 'Friday'

        WHEN DAYOFWEEK(transaction\_date) = 7 THEN 'Saturday'

        ELSE 'Sunday'

    END;



***TO GET SALES FOR ALL HOURS FOR MONTH OF MAY***

SELECT

    HOUR(transaction\_time) AS Hour\_of\_Day,

    ROUND(SUM(unit\_price \* transaction\_qty)) AS Total\_Sales

FROM

    coffee\_shop\_sales

WHERE

    MONTH(transaction\_date) = 5 -- Filter for May (month number 5)

GROUP BY

    HOUR(transaction\_time)

ORDER BY

    HOUR(transaction\_time);

A screenshot of a computer

AI-generated content may be incorrect.