**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)



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

SELECT

MONTH(transaction\_date) AS month,

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

CASE

WHEN LAG(SUM(unit\_price \* transaction\_qty), 1) OVER (ORDER BY MONTH(transaction\_date)) = 0 THEN 0

ELSE (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

END 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);



1. **MONTH(transaction\_date) AS month**: This extracts the month from the transaction\_date column and assigns it the alias month. This allows us to group the data by month.
2. **ROUND(SUM(unit\_price \* transaction\_qty)) AS total\_sales**: This calculates the total sales for each month by multiplying the unit\_price and transaction\_qty columns, summing the results, and then rounding the final value.
3. **CASE ... WHEN ... THEN ... ELSE ... END AS mom\_increase\_percentage**: This calculates the month-over-month (MoM) increase percentage for each month.
   * LAG(SUM(unit\_price \* transaction\_qty), 1) OVER (ORDER BY MONTH(transaction\_date)) retrieves the total sales for the previous month, using the LAG function to look back one row (i.e., the previous month).
   * The CASE statement checks if the previous month's total sales is zero. If so, it sets the MoM increase percentage to 0 to avoid a division by zero error.
   * If the previous month's total sales is not zero, it calculates the MoM increase percentage as (current month's total sales - previous month's total sales) / previous month's total sales \* 100.
4. **FROM your\_table**: This specifies the table from which the data is being queried.
5. **GROUP BY MONTH(transaction\_date)**: This groups the data by month, allowing the SUM() and other aggregate functions to be calculated for each month.
6. **ORDER BY month**: This sorts the results by the month column in ascending order.

The key aspects of this query are:

* Calculating the total sales for each month
* Calculating the month-over-month increase percentage, handling the case where the previous month's sales is zero
* Grouping the data by month and sorting the results

**TOTAL ORDERS**

SELECT COUNT(transaction\_id) as Total\_Orders

FROM coffee\_shop\_sales

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



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

SELECT

MONTH(transaction\_date) AS month,

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

CASE

WHEN LAG(COUNT(transaction\_id), 1) OVER (ORDER BY MONTH(transaction\_date)) = 0 THEN 0

ELSE (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

END 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)



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

SELECT

MONTH(transaction\_date) AS month,

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

CASE

WHEN LAG(SUM(transaction\_qty), 1) OVER (ORDER BY MONTH(transaction\_date)) = 0 THEN 0

ELSE (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

END 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



**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.



**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);

 

**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;



**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



**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)



**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);

