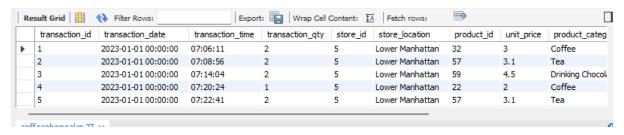
MY SQL QUERIES

COFFEE SHOP SALES PROJECT

DATA given

SELECT * FROM coffeeshopsales;



DATA TYPES OF DIFFERENT COLUMNS

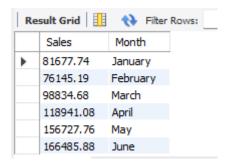
describe coffeeshopsales;



TOTAL SALES KPI - MOM DIFFERENCE AND MOM GROWTH

select round(sum(unit_price* transaction_qty),2) as 'Sales', monthname(transaction_date) as 'Month' from coffeeshopsales

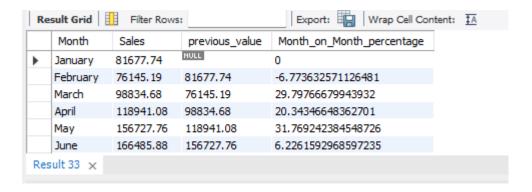
group by monthname(transaction_date);



TOTAL SALES KPI - MOM DIFFERENCE AND MOM GROWTH

with TBL AS (

```
with tbl as (
               select round(sum(unit_price* transaction_qty),2) as 'Sales',
                        monthname(transaction_date) as 'Month'
               from coffeeshopsales
               group by monthname(transaction_date)
               )
       select Month, Sales,
        LAG(Sales) over() AS previous_value,
               CASE
                       WHEN Sales > LAG(Sales) over() THEN 'greater'
                       WHEN Sales = LAG(Sales) over() THEN 'equal'
                       ELSE 'less'
               END AS comparison_result
       from tbl)
SELECT Month,
   Sales,
         previous_value,
   CASE
    WHEN Sales is not null and previous_value is not null THEN ((Sales-
previous_value)/previous_value)*100
    ELSE 0
   END as "Month_on_Month_percentage"
FROM TBL;
```



```
TOTAL ORDERS
with TBL AS (
       with tbl as (
               select monthname(transaction_date) as 'Month',count(transaction_id) as
'Total no Orders' from coffeeshopsales
               group by monthname(transaction_date)
               )
       select Month, Total_no_Orders,
        LAG(Total_no_Orders) over() AS previous_value,
               CASE
                      WHEN Total_no_Orders > LAG(Total_no_Orders) over() THEN 'greater'
                      WHEN Total_no_Orders = LAG(Total_no_Orders) over() THEN 'equal'
                      ELSE 'less'
               END AS comparison_result
       from tbl)
SELECT Month,
   Total_no_Orders,
         previous_value,
   CASE
    WHEN Total_no_Orders is not null and previous_value is not null THEN ((Total_no_Orders-
previous_value)/previous_value)*100
    ELSE 0
   END as "Month_on_Month_percentage"
```

FROM TBL;

Month	Total_no_Orders	previous_value	Month_on_Month_percentage
January	17314	NULL	0
February	16359	17314	-5.5158
March	21229	16359	29.7695
April	25335	21229	19.3415
May	33527	25335	32.3347
June	35352	33527	5.4434

TOTAL QUANTITY SOLD KPI - MOM DIFFERENCE AND MOM GROWTH

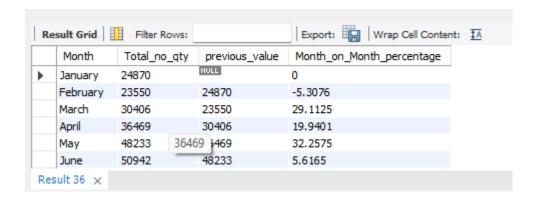
```
with TBL AS (
       with tbl as (
               select monthname(transaction_date) as 'Month',sum(transaction_qty) as
'Total_no_qty' from coffeeshopsales
               group by monthname(transaction_date)
               )
       select Month, Total_no_qty,
        LAG(Total_no_qty) over() AS previous_value,
               CASE
                       WHEN Total_no_qty > LAG(Total_no_qty) over() THEN 'greater'
                       WHEN Total_no_qty = LAG(Total_no_qty) over() THEN 'equal'
                       ELSE 'less'
               END AS comparison_result
       from tbl)
SELECT Month,
   Total_no_qty,
         previous_value,
   CASE
    WHEN Total_no_qty is not null and previous_value is not null THEN ((Total_no_qty-
```

previous_value)/previous_value)*100

ELSE 0

END as "Month_on_Month_percentage"

FROM TBL;



CALENDAR TABLE – DAILY SALES, QUANTITY and TOTAL ORDERS

select transaction_date,concat(round(sum(unit_price * transaction_qty)/100,1), "k") as "Total_sales",

concat(round(sum(transaction_qty)/100,1), "k") as "Total_Qty_sold",

concat(round(count(transaction_id)/100,1), "k") as "Total_Orders"

from coffeeshopsales

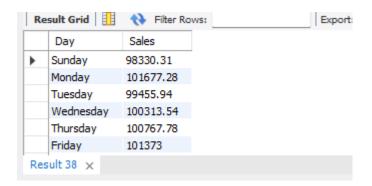
group by transaction_date;



Sales made over the weekdays and weekends

select distinct(dayname(transaction_date)) as 'Day', round(sum((unit_price * transaction_qty)),2) as 'Sales' from coffeeshopsales

group by Day;



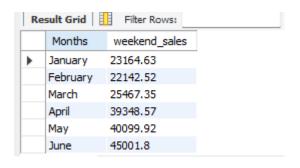
Weekends sales



Weekends sales per month

```
with tbl as ( with tbl as (
```

select distinct(dayname(transaction_date)) as 'Day',monthname(transaction_date)
as "Months", round(sum((unit_price * transaction_qty)),2) as 'Sales' from coffeeshopsales
group by Day, monthname(transaction_date))
select * from tbl
where Day in ('Sunday','Saturday'))
select Months,round(sum(Sales),2) as 'weekend_sales' from tbl



weekdays sales per month

with tbl as (with tbl as (

group by Months;

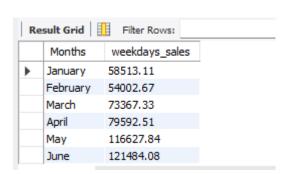
select distinct(dayname(transaction_date)) as 'Day',monthname(transaction_date) as "Months", round(sum((unit_price * transaction_qty)),2) as 'Sales' from coffeeshopsales

group by Day, monthname(transaction_date))

select * from tbl

where Day not in ('Sunday', 'Saturday'))

select Months,round(sum(Sales),2) as 'weekdays_sales' from tbl group by Months;



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

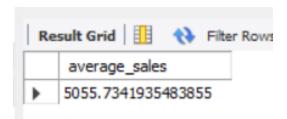
MONTH(transaction_date) = 5 -- Filter for May

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);

Result Grid H				
	day_of_month	total_sales		
•	1	4731.4		
	2	4625.5		
	3	4714.6		
	4	4589.7		
	5	4701		
	6	4205.1		
	7	4542.7		
	8	5604.2		
	9	5101		
	10	5256.3		
	11	4850.1		
	12	4681.1		
	13	5511.5		
	14	5052.6		
	15	5385		
	16	5542.1		

17	5418
18	5583.5
19	5657.9
20	5519.3
21	5370.8
22	5541.2
23	5242.9
24	5391.4
25	5230.8
26	5300.9
27	5559.2
28	4338.6
29	3959.5
30	4835.5
31	4684.1

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_status	total_sales
1	Below Average	4731,449999999999
2	Below Average	4625.499999999997
3	Below Average	4714.599999999994
4	Below Average	4589.699999999995
5	Below Average	4700.999999999997
6	Below Average	4205.149999999998
7	Below Average	4542.699999999998
8	Above Average	5604.209999999995
9	Above Average	5100.969999999997
10	Above Average	5256.329999999999
11	Below Average	4850.059999999996
12	Below Average	4681.1299999999965
13	Above Average	5511.529999999999
14	Below Average	5052.649999999999
15	Above Average	5384.9800000000005
16	Above Average	5542.129999999997

day_of_month;

17	Above Average	5418.000000000001
18	Above Average	5583.470000000001
19	Above Average	5657.880000000005
20	Above Average	5519.280000000003
21	Above Average	5370.810000000003
22	Above Average	5541.16
23	Above Average	5242.910000000001
24	Above Average	5391.45
25	Above Average	5230.8499999999985
26	Above Average	5300.94999999998
27	Above Average	5559.1500000000015
28	Below Average	4338.649999999998
29	Below Average	3959.499999999998
30	Below Average	4835.479999999997
31	Below Average	4684.129999999993

SALES BY WEEKDAY / WEEKEND:

```
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

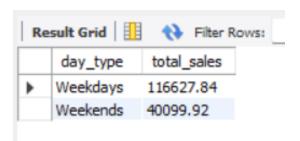
GROUP BY

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

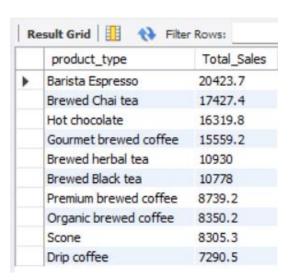
WHERE

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

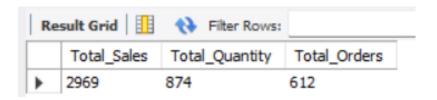
coffee_shop_sales

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'

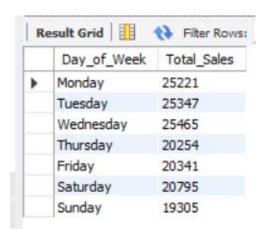
WHEN DAYOFWEEK(transaction_date) = 3 THEN 'Tuesday'

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);

