MY SQL QUERIES

COFFEE SHOP SALES PROJECT

DATA TYPES OF DIFFERENT COLUMNS

DESCRIBE coffee_shop_sales;

Result Grid | Filter Rows: | Export: | Wra

Field	Type	Null	Key	Default	Extra
ï»¿transaction_id	int	YES		NULL	
transaction_date	date	YES		NULL	
transaction_time	time	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	

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)
```

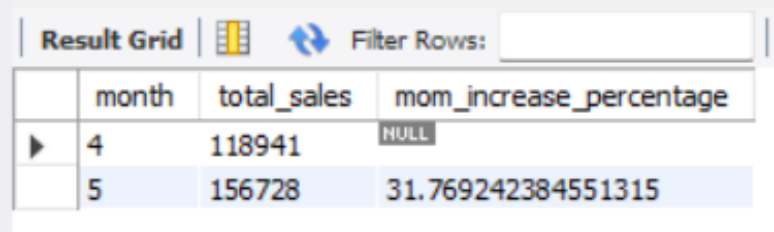
Result Grid | F

Total_Sales
156728

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);
```



month	total_sales	mom_increase_percentage
4	118941	NULL
5	156728	31.769242384551315

Explaination

SELECT clause:

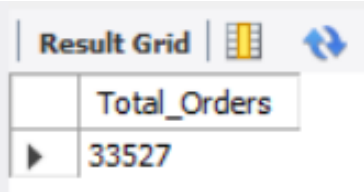
- MONTH(transaction_date) AS month: Extracts the month from the transaction_date column and renames it as month.
- ROUND(SUM(unit_price * transaction_qty)) AS total_sales: Calculates the total sales by multiplying unit_price and transaction_qty, then sums the result for each month. The ROUND function 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

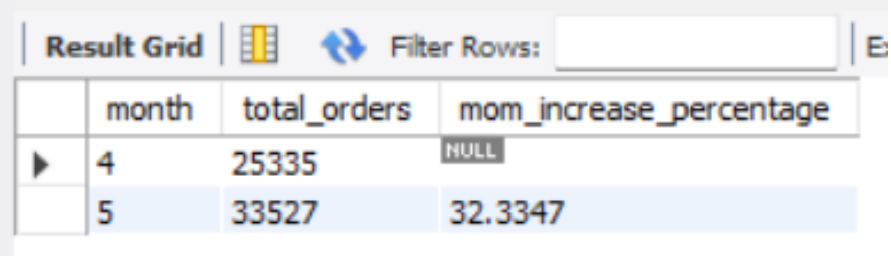
```
SELECT COUNT(transaction_id) as Total_Orders
FROM coffee_shop_sales
WHERE MONTH (transaction_date)= 5 -- for month of (CM-May)
```



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)
```

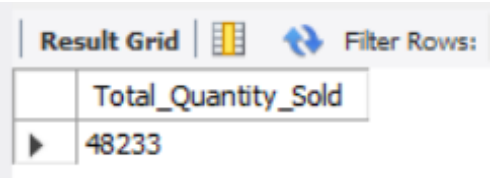
```
    OVER (ORDER BY MONTH(transaction_date))) / LAG(COUNT(transaction_id), 1)
    OVER (ORDER BY MONTH(transaction_date)) * 100 AS mom_increase_percentage
FROM
    coffee_shop_sales
WHERE
    MONTH(transaction_date) IN (4, 5) -- for April and May
GROUP BY
    MONTH(transaction_date)
ORDER BY
    MONTH(transaction_date);
```



month	total_orders	mom_increase_percentage
4	25335	NULL
5	33527	32.3347

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)
```

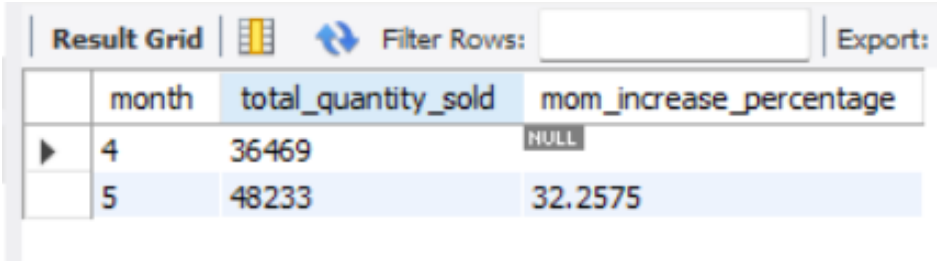


Total_Quantity_Sold
48233

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
```

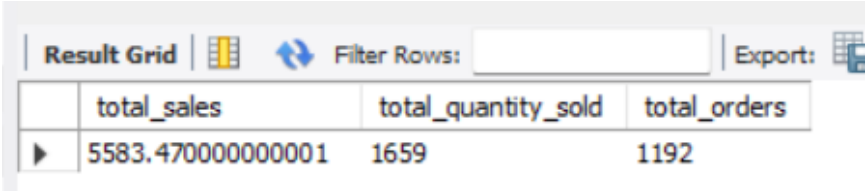
```
 coffee_shop_sales
WHERE
    MONTH(transaction_date) IN (4, 5)   -- for April and May
GROUP BY
    MONTH(transaction_date)
ORDER BY
    MONTH(transaction_date);
```


Result Grid | Filter Rows: | Export:

month	total_quantity_sold	mom_increase_percentage
4	36469	NULL
5	48233	32.2575

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
```

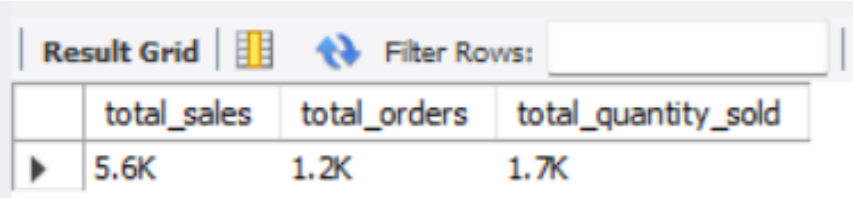

Result Grid | Filter Rows: | Export:

total_sales	total_quantity_sold	total_orders
5583.470000000001	1659	1192

<u>If you want to get exact Rounded off values then use below query to get the result:</u>

```
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
```



Result Grid | Filter Rows:

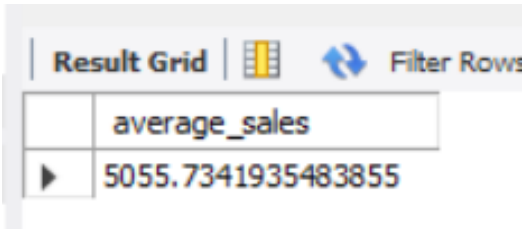
	total_sales	total_orders	total_quantity_sold
▶	5.6K	1.2K	1.7K

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 | Filter Rows:

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

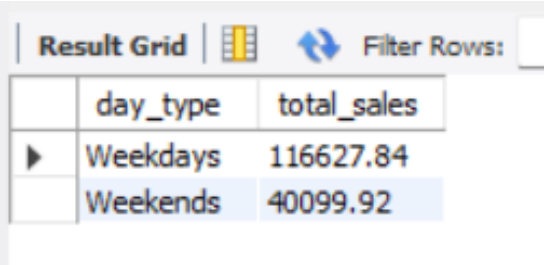
COMPARING DAILY SALES WITH AVERAGE SALES – IF GREATER THAN "ABOVE AVERAGE" and LESSER THAN "BELOW AVERAGE"

```
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;
```

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
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	5300.949999999998
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:

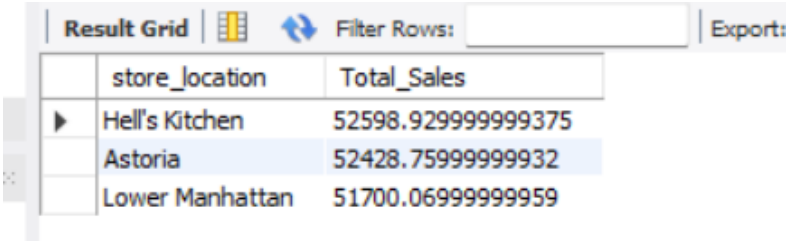
```
SELECT
   CASE
      WHEN DAYOFWEEK(transaction_date) IN (1, 7) THEN 'Weekends'
      ELSE 'Weekdays'
   END AS day_type,
   ROUND(SUM(unit_price * transaction_qty),2) AS total_sales
FROM
   coffee_shop_sales
WHERE
   MONTH(transaction_date) = 5  -- Filter for May
GROUP BY
   CASE
      WHEN DAYOFWEEK(transaction_date) IN (1, 7) THEN 'Weekends'
      ELSE 'Weekdays'
   END;
```



day_type	total_sales
Weekdays	116627.84
Weekends	40099.92

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
```

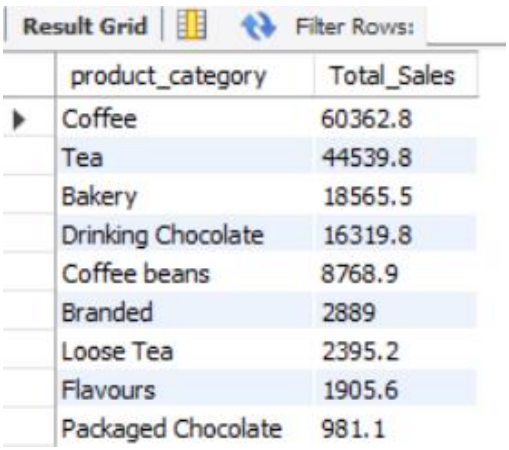


Result Grid | Filter Rows: | Export:

store_location	Total_Sales
Hell's Kitchen	52598.929999999375
Astoria	52428.75999999932
Lower Manhattan	51700.06999999959

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
```



Result Grid | Filter Rows:

product_category	Total_Sales
Coffee	60362.8
Tea	44539.8
Bakery	18565.5
Drinking Chocolate	16319.8
Coffee beans	8768.9
Branded	2889
Loose Tea	2395.2
Flavours	1905.6
Packaged Chocolate	981.1

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
```

Result Grid | Filter Rows:

product_type	Total_Sales
Barista Espresso	20423.7
Brewed Chai tea	17427.4
Hot chocolate	16319.8
Gourmet brewed coffee	15559.2
Brewed herbal tea	10930
Brewed Black tea	10778
Premium brewed coffee	8739.2
Organic brewed coffee	8350.2
Scone	8305.3
Drip coffee	7290.5

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)
```

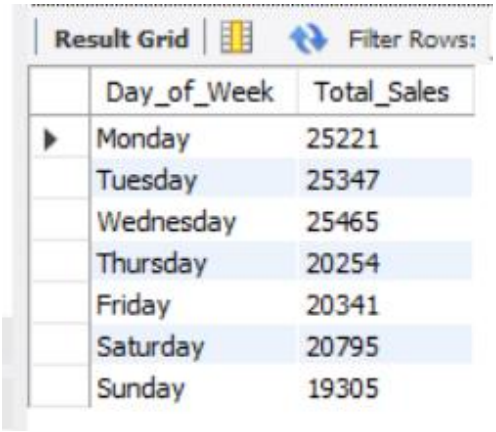
Result Grid | Filter Rows:

Total_Sales	Total_Quantity	Total_Orders
2969	874	612

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;
```


Result Grid | Filter Rows:

Day_of_Week	Total_Sales
Monday	25221
Tuesday	25347
Wednesday	25465
Thursday	20254
Friday	20341
Saturday	20795
Sunday	19305

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);
```

Result Grid | Filter Rows:

Hour_of_Day	Total_Sales
6	4913
7	14351
8	18822
9	19145
10	19639
11	10312
12	8870
13	9379
14	9058
15	9525
16	9154
17	8967
18	7680
19	6256
20	656