

Coffee Shop Sales Analysis

- See all the data imported:

```
SELECT * FROM coffee_shop_sales
```

- **DATA CLEANING:**

Cleaning the coffee_shop_sales field ensures data consistency and accuracy in analysis. We need to change the data type of the transaction_date and transaction_time fields. Additionally, we must rename the field “transaction_id” to “transaction_id”. By standardizing these values, we improve data quality, making it easier to generate insights and maintain uniformity in our datasets.

1) ALTER DATE (transaction_date) COLUMN TO DATE DATA TYPE

```
ALTER TABLE coffee_shop_sales
```

```
MODIFY COLUMN transaction_date DATE;
```

2) ALTER TIME (transaction_time) COLUMN TO DATE DATA TYPE

```
ALTER TABLE coffee_shop_sales
```

```
MODIFY COLUMN transaction_time TIME;
```

3) CHANGE COLUMN NAME `transaction_id` to transaction_id

```
ALTER TABLE coffee_shop_sales
```

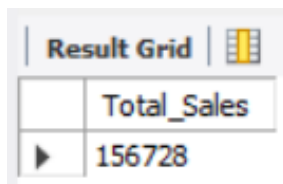
```
CHANGE COLUMN `transaction_id` transaction_id INT;
```

	Field	Type	Null	Key	Default	Extra
►	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	

A. KPI's

1. TOTAL SALES:

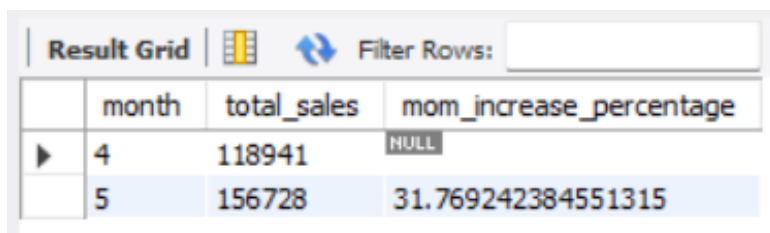
```
SELECT ROUND(SUM(unit_price * transaction_qty)) as Total_Sales
FROM coffee_shop_sales
WHERE MONTH(transaction_date) = 5      --(for month of May)
```



Total_Sales
156728

2. TOTAL SALES - MOM DIFFERENCE AND MOM GROWTH

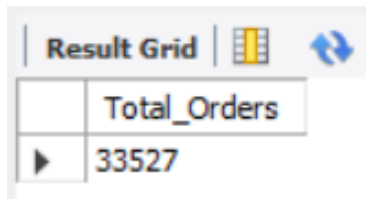
```
SELECT
    MONTH(transaction_date) AS month,
    ROUND(SUM(unit_price * transaction_qty)) AS total_sales,
    (SUM(unit_price * transaction_qty) - LAG(SUM(unit_price * transaction_qty), 1)
    OVER (ORDER BY MONTH(transaction_date))) / LAG(SUM(unit_price * transaction_qty), 1)
    OVER (ORDER BY 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

3. TOTAL ORDERS

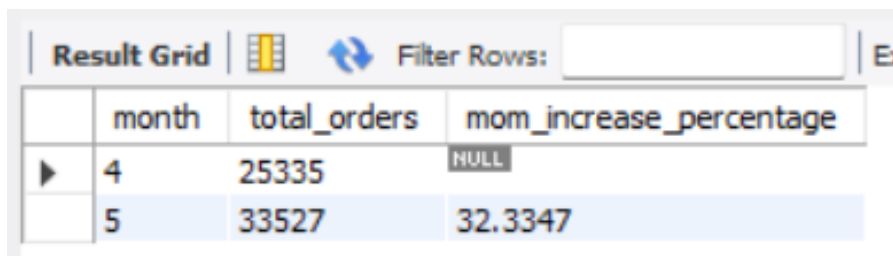
```
SELECT COUNT(transaction_id) as Total_Orders
FROM coffee_shop_sales
WHERE MONTH (transaction_date)= 5      --for month of (May)
```



Total_Orders
33527

4. TOTAL ORDERS - MOM DIFFERENCE AND MOM GROWTH

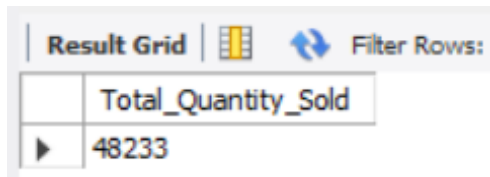
```
SELECT
    MONTH(transaction_date) AS month,
    ROUND(COUNT(transaction_id)) AS total_orders,
    (COUNT(transaction_id) - LAG(COUNT(transaction_id), 1)
    OVER (ORDER BY MONTH(transaction_date))) / LAG(COUNT(transaction_id), 1)
    OVER (ORDER BY MONTH(transaction_date)) * 100 AS mom_increase_percentage
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

5. TOTAL QUANTITY SOLD

```
SELECT SUM(transaction_qty) as Total_Quantity_Sold  
FROM coffee_shop_sales  
WHERE MONTH(transaction_date) = 5      --(for month of May)
```

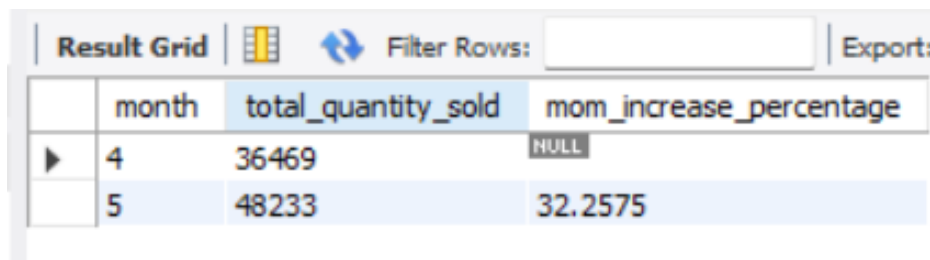


The screenshot shows a 'Result Grid' with a single column labeled 'Total_Quantity_Sold' and a single row with the value '48233'. Above the grid are icons for 'Filter Rows:' and 'Export:'. Below the grid is a 'Result Grid' label and a 'Filter Rows:' input field.

Total_Quantity_Sold
48233

6. 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);
```



The screenshot shows a 'Result Grid' with three columns: 'month', 'total_quantity_sold', and 'mom_increase_percentage'. The first row shows month 4 with a total quantity sold of 36469 and a null mom increase percentage. The second row shows month 5 with a total quantity sold of 48233 and a mom increase percentage of 32.2575. Above the grid are icons for 'Filter Rows:' and 'Export:'. Below the grid is a 'Result Grid' label and a 'Filter Rows:' input field.

month	total_quantity_sold	mom_increase_percentage
4	36469	NULL
5	48233	32.2575

7. CALENDAR TABLE – DAILY SALES, QUANTITY and TOTAL ORDERS

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,

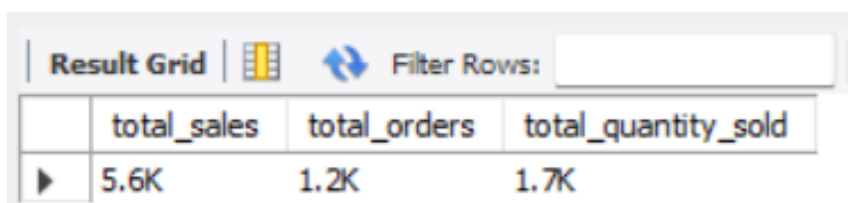
CONCAT(ROUND(SUM(transaction_qty) / 1000, 1), 'K') AS total_quantity_sold

FROM

coffee_shop_sales

WHERE

transaction_date = '2023-05-18'; **(For 18 May 2023)**



	total_sales	total_orders	total_quantity_sold
▶	5.6K	1.2K	1.7K

8. 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

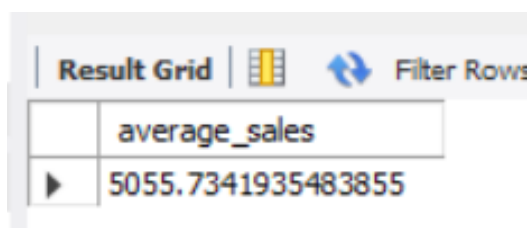
WHERE

MONTH(transaction_date) = 5 **--(Filter for May)**

GROUP BY

transaction_date

) AS internal_query;



	average_sales
▶	5055.7341935483855

9. 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)

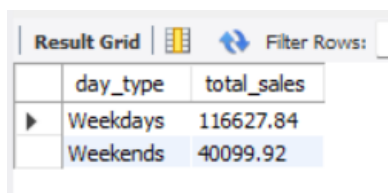
ORDER BY

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

10. 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;
```

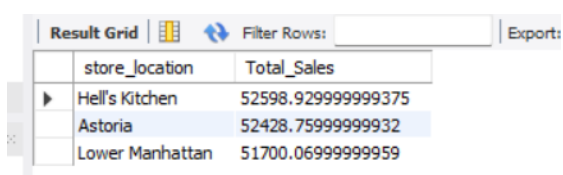


The screenshot shows a 'Result Grid' with two columns: 'day_type' and 'total_sales'. There are two rows: 'Weekdays' with a total sales of 116627.84, and 'Weekends' with a total sales of 40099.92. The 'Weekends' row is highlighted in blue.

day_type	total_sales
Weekdays	116627.84
Weekends	40099.92

11. 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
```



The screenshot shows a 'Result Grid' with two columns: 'store_location' and 'Total_Sales'. There are three rows: 'Hell's Kitchen' with a total sales of 52598.929999999375, 'Astoria' with a total sales of 52428.759999999932, and 'Lower Manhattan' with a total sales of 51700.069999999959. The 'Astoria' row is highlighted in blue.

store_location	Total_Sales
Hell's Kitchen	52598.929999999375
Astoria	52428.759999999932
Lower Manhattan	51700.069999999959

12. SALES BY PRODUCT CATEGORY

```
SELECT

    product_category,

    ROUND(SUM(unit_price * transaction_qty),1) as Total_Sales

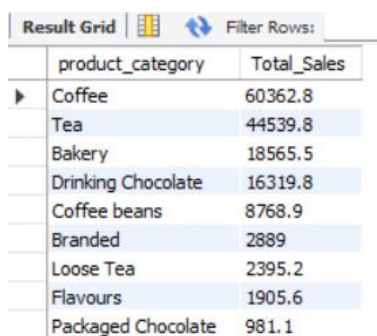
FROM coffee_shop_sales

WHERE

    MONTH(transaction_date) = 5

GROUP BY product_category

ORDER BY SUM(unit_price * transaction_qty) DESC
```



The screenshot shows a 'Result Grid' with two columns: 'product_category' and 'Total_Sales'. The data is sorted in descending order of total sales. The categories and their sales values are: 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), and Packaged Chocolate (981.1).

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

13. SALES BY PRODUCTS (TOP 10)

```
SELECT

    product_type,

    ROUND(SUM(unit_price * transaction_qty),1) as Total_Sales

FROM coffee_shop_sales

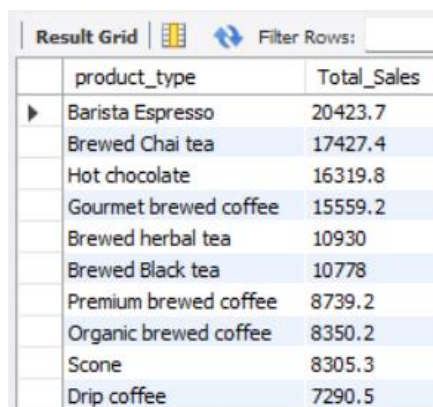
WHERE

    MONTH(transaction_date) = 5

GROUP BY product_type

ORDER BY SUM(unit_price * transaction_qty) DESC

LIMIT 10
```



The screenshot shows a 'Result Grid' with two columns: 'product_type' and 'Total_Sales'. The data is sorted in descending order of total sales, limited to the top 10 products. The product types and their sales values are: 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), and Drip coffee (7290.5).

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

14. 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

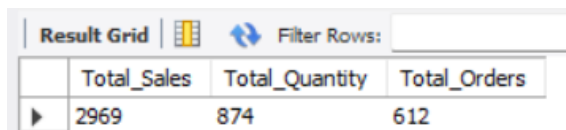
coffee_shop_sales

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))



	Total_Sales	Total_Quantity	Total_Orders
▶	2969	874	612

15. 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	