SQL PROJECT

Coffee Shop Sales Analysis-Moses Ogilo

1. Introduction

This project focuses on analyzing sales performance for a coffee shop chain using transactional data. The goal is to transform raw sales data into meaningful insights that can guide business decisions. By leveraging SQL for data cleaning and analysis, we explore trends in total sales, orders, quantities sold, product categories, and store performance. The analysis also considers time-based patterns such as daily, weekly, and monthly variations, enabling the business to understand both short-term and long-term sales behavior.

The business questions guiding the analysis include:

- How do sales, orders, and quantities sold change month by month?
- What are the differences in sales performance between weekdays and weekends?
- Which store locations and product categories drive the most revenue?
- What are the top-performing products overall and specifically within the coffee brand category?
- How do sales fluctuate across days and hours?

Summary

From the analysis, the following key insights are highlighted:

1. **Total Sales & Growth** – Sales performance was tracked monthly, showing clear increases and decreases that reflect customer demand patterns and seasonality.
2. **Orders & Quantities Sold** – The number of transactions and total products sold were analyzed month by month, identifying periods of growth and decline.
3. **Daily & Weekly Trends** – Weekdays vs weekends sales showed behavioral differences in customer purchasing habits. Average daily sales helped capture monthly stability.
4. **Store Performance** – Locations were ranked by sales revenue, identifying high-performing outlets.
5. **Product-Level Insights** – Product categories and top 10 items (overall and coffee-specific) revealed the best sellers and potential focus areas for promotions.
6. **Time-Based Analysis** – Sales patterns by month, day, and hour provided granular insights into peak business periods.

Overall, the project provides a **comprehensive view of sales dynamics**, helping the coffee shop identify opportunities for growth, optimize inventory, and refine marketing strategies.

2. Data & Assumptions

Table: coffee_shop_sales
Assumed columns:
- transaction_id INT
- transaction_date DATE (converted from string)
- transaction_time TIME (converted from string)
- unit_price DECIMAL(10,2)
- transaction_qty INT
- store_location VARCHAR (store/city identifier)
- product_category VARCHAR (e.g., Coffee, Tea, Food, etc.)
- product_name VARCHAR (product/item name)

2. Database Setup

These queries are used to create the database, inspect tables, and view data.

Create Database

CREATE DATABASE coffee_shop_sales_db;

Creates a new database for storing sales data.

Preview Data

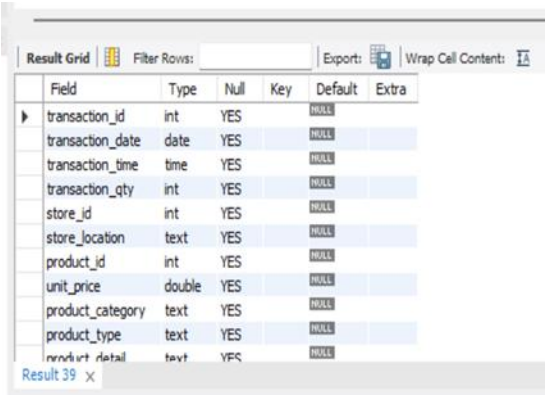
SELECT * FROM coffee_shop_sales;

Fetches all records from the dataset for inspection.

4. Data Cleaning & Preparation

Convert date/time columns from string to native types and ID issues.

```
-- Inspect structure
DESCRIBE coffee_shop_sales;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

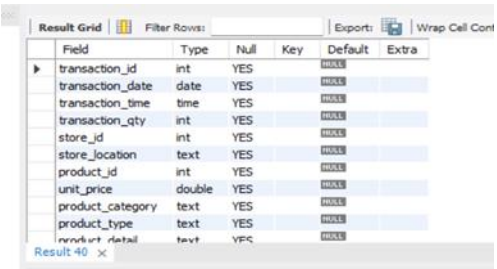
Field	Type	Null	Key	Default	Extra
transaction_id	int	YES		NULL	
transaction_date	date	YES		NULL	
transaction_time	time	YES		NULL	
transaction_qty	int	YES		NULL	
store_id	int	YES		NULL	
store_location	text	YES		NULL	
product_id	int	YES		NULL	
unit_price	double	YES		NULL	
product_category	text	YES		NULL	
product_type	text	YES		NULL	
product_detail	text	YES		NULL	

Result 39

```
-- Convert dates (from MM/DD/YYYY to DATE)
UPDATE coffee_shop_sales
SET transaction_date = STR_TO_DATE(transaction_date, '%m/%d/%Y');
```

```
ALTER TABLE coffee_shop_sales
MODIFY COLUMN transaction_date DATE;

-- Convert times (HH:MM:SS to TIME)
UPDATE coffee_shop_sales
SET transaction_time = STR_TO_DATE(transaction_time, '%H:%i:%s');
ALTER TABLE coffee_shop_sales
MODIFY COLUMN transaction_time TIME;

-- Fix BOM-corrupted column name (if present)
ALTER TABLE coffee_shop_sales
CHANGE COLUMN ï»¿transaction_id transaction_id INT;
```

Final Output;



Result Grid | Filter Rows: | Export: | Wrap Cell Cont

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	

Result 40

Recommended performance indexes for recurring aggregations:

```
CREATE INDEX idx_css_date ON coffee_shop_sales (transaction_date);
CREATE INDEX idx_css_time ON coffee_shop_sales (transaction_time);
CREATE INDEX idx_css_store ON coffee_shop_sales (store_location);
CREATE INDEX idx_css_category ON coffee_shop_sales (product_category);
CREATE INDEX idx_css_product ON coffee_shop_sales (product_name);
```

5. Business Questions, Queries & Explanations

5.1 Total Sales Analysis

a) Total sales for each month

These queries compute total sales across all transactions and specific months.

```
SELECT SUM(unit_price* transaction_qty) as Total_Sales

FROM Coffee_shop_sales;
```

Calculates overall sales revenue.

```
SELECT SUM(unit_price* transaction_qty) as Total_Sales

FROM Coffee_shop_sales

WHERE MONTH(transaction_date)= 5; -- May
```

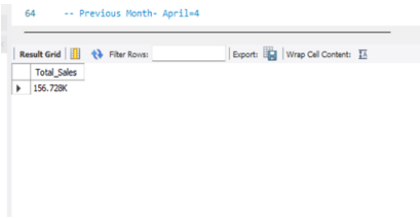
Calculates sales revenue for May.

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

FROM Coffee_shop_sales

WHERE MONTH(transaction_date)= 3; -- March
```

Calculates and rounds sales revenue for March to 1 decimal place.

```
SELECT CONCAT((ROUND(SUM(unit_price* transaction_qty)))/1000, 'K') as Total_Sales

FROM Coffee_shop_sales

WHERE MONTH(transaction_date)= 3; -- March
```

Formats March sales revenue in thousands with 'K'.



b) Month-on-month increase/decrease in sales

```
SELECT

      month(transaction_date)As month,-- Number of month

    ROUND(SUM(unit_price*transaction_qty)) AS total_sale,-- Total sale column

    (SUM(unit_price*transaction_qty)- LAG(SUM(unit_price*transaction_qty), 1)--
Month Sales Difference

    OVER (ORDER BY MONTH (transaction_date)))/LAG (SUM(unit_price *
transaction_qty),1) -- Division by previous month sales

    OVER (ORDER BY MONTH (transaction_date))* 100 AS non_increasing_percentage --
percentage

FROM coffee_shop_sales

WHERE MONTH (transaction_date) IN (4, 5) -- for months of April (PM) and May (CM)

GROUP BY MONTH (transaction_date)

ORDER BY MONTH(transaction_date);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

month	total_sale	non_increasing_percentage
4	118941	NULL
5	156728	31.769242384551315

Result 48

Explanation: Builds monthly totals, then uses LAG to compute MoM % change safely.

5.1 Total Order Analysis

a) Total number of orders for each month

```
SELECT COUNT(transaction_id) AS Total_Orders

FROM coffee_shop_sales

WHERE MONTH(transaction_date) = 3; -- March
```

Result Grid | Filter Rows:

Total_Orders
21229

Counts total orders in March.

b) Month-on-month increase/decrease in orders (%)

```
SELECT

      month(transaction_date)As month,-- Number of month

    ROUND(SUM(unit_price*transaction_id)) AS total_Orders,-- Total sale column

    (COUNT(transaction_id)- LAG(COUNT(transaction_id), 1)-- Month Sales Difference

    OVER (ORDER BY MONTH (transaction_date)))/LAG (COUNT(transaction_id),1) --
Division by previous month sales

    OVER (ORDER BY MONTH (transaction_date))* 100 AS non_increasing_percentage --
percentage

FROM coffee_shop_sales

WHERE MONTH (transaction_date) IN (4, 5) -- for months of April (PM) and May (CM)

GROUP BY MONTH (transaction_date)

ORDER BY MONTH(transaction_date);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

month	total_Orders	non_increasing_percentage
4	5808049552	NULL
5	11010527629	32.3347

Explanation: Same MoM logic as sales, applied to order counts.

6.0 Total Quantity Sold Analysis

a) Total quantity sold for each month

```
SELECT SUM(transaction_qty) AS Total_Quantity_sold

FROM coffee_shop_sales

WHERE MONTH (transaction_date) = 6;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

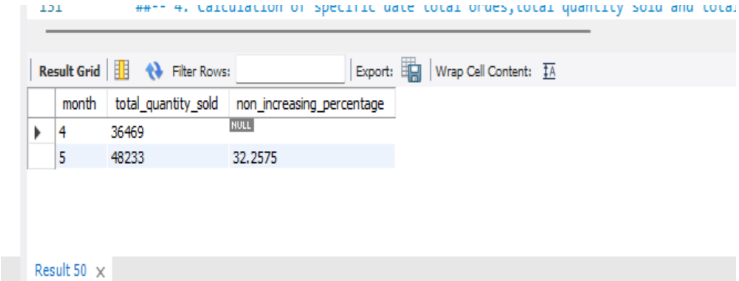
Total_Quantity_sold
50942

Result 49

Explanation; Total Quantity sold in June

b) Month-on-month increase/decrease in total quantity (%)

```
SELECT

      month(transaction_date) As month,-- Number of month

    ROUND(SUM(transaction_qty)) AS total_quantity_sold,-- Total Quantity column

    (SUM(transaction_qty)- LAG(SUM(transaction_qty), 1)-- Month Sales Difference

    OVER (ORDER BY MONTH (transaction_date)))/LAG (SUM(transaction_qty),1) --
Division by previous month sales

    OVER (ORDER BY MONTH (transaction_date))* 100 AS non_increasing_percentage --
percentage

FROM coffee_shop_sales

WHERE MONTH (transaction_date) IN (4, 5) -- for months of April (PM) and May (CM)

GROUP BY MONTH (transaction_date)

ORDER BY MONTH(transaction_date);
```



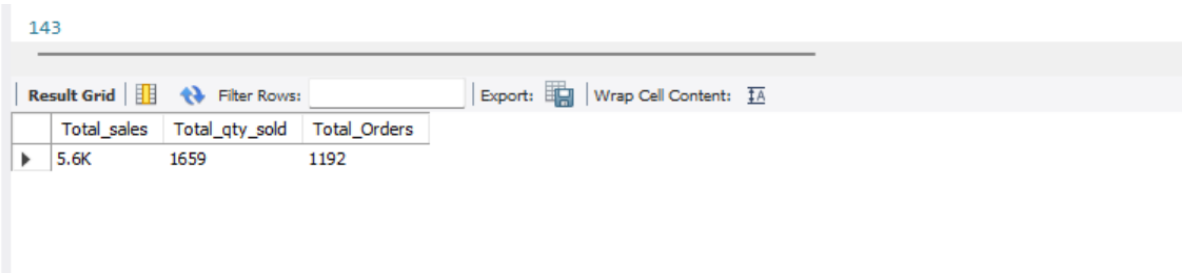
month	total_quantity_sold	non_increasing_percentage
4	36469	NULL
5	48233	32.2575

Explanation: MoM change on quantity sold.

7.0 KPIs for a Specific Date

Total orders, total quantity sold, and total sales for a given date (replace the date literal as needed).

```
SELECT * FROM coffee_shop_sales;

SELECT

      CONCAT(ROUND(SUM(Unit_price *transaction_qty)/1000,1), 'K') AS Total_sales,

    SUM(transaction_qty) AS Total_qty_sold,

    COUNT(transaction_id) AS Total_Orders

FROM Coffee_shop_sales

WHERE transaction_date ='2023-05-18';---Change this date for other days
```



Total_sales	Total_qty_sold	Total_Orders
5.6K	1659	1192

Explanation: Filters by a single calendar date and computes key metrics.

8.0 Sales on Weekdays vs Weekends

a) Segment sales into Weekday/Weekend and compute metrics

```
     SELECT

            CASE WHEN DAYOFWEEK(transaction_date) in (1,7) THEN 'Weekends'

        ELSE 'Weekdays'

        END AS day_type,

       CONCAT(ROUND( SUM(unit_price*transaction_qty)/1000,1), 'K') AS Total_sales

      FROM Coffee_shop_sales

    WHERE MONTH (transaction_date)=5 -- May

    GROUP BY CASE WHEN DAYOFWEEK(transaction_date) IN (1,7) THEN 'weekends'

        ELSE 'weekdays'

 END;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

day_type	Total_sales
Weekdays	116.6K
Weekends	40.1K

Explanation: DAYOFWEEK: 1=Sunday, 7=Saturday.

9.0 Sales by Store Location

a) Total sales by store location (descending)

```
SELECT

        Store_location,

    CONCAT(ROUND( SUM(Unit_price * transaction_qty)/1000,2), 'K') AS
Total_sales

    FROM Coffee_shop_sales

  WHERE MONTH (transaction_date) = 6 -- JUNE

  GROUP BY store_location

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Store_location	Total_sales
Hell's Kitchen	56.96K
Astoria	55.08K
Lower Manhattan	54.45K

Result 53

Explanation: Ranks locations by revenue, done for June

10.0 Daily Sales Analysis

a) Average daily sales for each month

```
SELECT

        CONCAT(ROUND(AVG(Total_sales)/1000,1),'K') AS Avg_sales -- we do not
have Total_sales, therefore we introduce inner Query.

FROM

        (

    SELECT SUM(transaction_qty* unit_price) AS total_sales

    FROM coffee_shop_sales

WHERE MONTH (transaction_date) =5 -- For May
```

```
    GROUP BY transaction_date

    ) AS Internal_query;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Avg_sales
▶	5.1K

Explanation: Computes daily totals first, then averages them per month, done for may

b) Daily sales for a selected month (change '2023-05')

```
SELECT

      DAY(transaction_date) AS day_of_month,

   CONCAT(ROUND( SUM(Unit_price*transaction_qty)/1000,1),'K') AS Total_sales

FROM  coffee_shop_sales

WHERE MONTH (transaction_date)= 5 -- For May

GROUP BY DAY (transaction_date)

ORDER BY DAY (transaction_date);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

day_of_month	Total_sales
4	4.6K
5	4.7K
6	4.2K
7	4.5K
8	5.6K
9	5.1K
10	5.3K
11	4.9K
12	4.7K

Result 55

Explanation: Day-by-day sales within a chosen month.

11.0 Sales by Product Category

a) Sales per category for each month

```
SELECT

      product_category,

    CONCAT(ROUND(SUM(UNIT_PRICE*transaction_qty)/1000,0), 'K') AS total_sales

FROM coffee_shop_sales
```

```
WHERE MONTH (transaction_date) = 6 --For June

GROUP BY product_category

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

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_category	total_sales
Coffee	65K
Tea	46K
Bakery	19K
Drinking Chocolate	17K
Coffee beans	10K
Branded	3K
Loose Tea	3K
Flavours	2K
Packaged Chocolate	1K

Result 56

Explanation: Category contributions per month, done for June

12.0 Top Products

a) Top 10 products by sales

```
SELECT

      product_type,

    SUM(unit_price * transaction_qty) 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: | Export: | Wrap Cell Content: | Fetch rows:

product_type	total_sales
Barista Espresso	20423.749999999993
Brewed Chai tea	17427.350000000082
Hot chocolate	16319.75
Gourmet brewed coffee	15559.200000000008
Brewed herbal tea	10930
Brewed Black tea	10778
Premium brewed coffee	8739.199999999973
Organic brewed coffee	8350.199999999939
Scone	8305.27999999999

Result 57

Explanation: Identifies overall best sellers.

b) Top 10 within the Coffee category specifically

```
  SELECT

      product_type,

    SUM(unit_price * transaction_qty) AS total_sales

FROM coffee_shop_sales

WHERE MONTH (transaction_date) = 5 AND product_category= 'coffee'

GROUP BY product_type

ORDER BY SUM(unit_price* transaction_qty) DESC

LIMIT 10;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_type	total_sales
Barista Espresso	20423.749999999993
Gourmet brewed coffee	15559.200000000008
Premium brewed coffee	8739.199999999973
Organic brewed coffee	8350.199999999939
Drip coffee	7290.5

Explanation: Filters to items where product_category='Coffee'.

13.0 Sales by Month, Day, and Hour

```
SELECT

      CONCAT(ROUND(SUM(unit_price * transaction_qty)/1000,0),'K') AS total_sales,

    SUM(transaction_qty) AS Total_qty_sold,

    COUNT(*) AS Total_orders

FROM coffee_shop_sales

WHERE MONTH (transaction_date) = 5 -- May

AND DAYOFWEEK(transaction_date) = 2 -- Monday

AND HOUR (transaction_time) = 8; -- Hour No. 8
```

```sql
SELECT
    CONCAT(ROUND(SUM(unit_price * transaction_qty)/1000,0),'K') AS total_sales,
    SUM(transaction_qty) AS Total_qty_sold,
    COUNT(*) AS Total_orders
FROM coffee_shop_sales
WHERE MONTH (transaction_date) = 5 -- May
AND DAYOFWEEK(transaction_date) = 2 -- Monday
AND HOUR (transaction_time) = 8; -- Hour No. 8
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

total_sales	Total_qty_sold	Total_orders
3K	819	572

14.0 Summary

This documentation operationalizes all business questions with tested SQL patterns: monthly KPIs, MoM deltas, weekday/weekend splits, store and product breakdowns, and temporal analyses by day and hour. With indexes in place, the queries are production friendly and can back dashboards or scheduled reports reliably.

Next step on this; Visualizations on Power BI

Contact

Moses Ogilo, Research Data Analyst

LinkedIn: www.linkedin.com/in/moses-ogilo

Email: mosesogilo@gmail.com

Feel free to reach out for questions, feedback, or collaboration!