**RETAIL SHOP SALES PROJECT**

In this case study, we were tasked with conducting an end-to-end project (ETL) where we built an interactive dashboard showcasing key performance indicators (KPIs) to address a client or stakeholder business inquiries and provide data-driven insights.

For my project, I employed the ETL method by extracting the 'retail shop sales' dataset from Kaggle. I performed data transformations in MySQL to create specific datasets that address the client's business questions. Subsequently, I loaded these datasets into Power BI to develop a dashboard. This visual tool was designed to explain the KPIs and provide insights to the client, helping them make informed business decisions.

PROBLEM STATEMENT:

KPI’s REQUIREMENTS BY CLIENT/STAKEHOLDER:

1. TOTAL SALES ANALYSIS

* Calculate the total sales for each respective month and the difference in total sales (as percentage) between current and previous month

1. TOTAL ORDER ANALYSIS

* Calculate the total order for each respective month and the difference in total orders (as percentage) between current and previous month

1. TOTAL QUANTITY SOLD ANALYSIS

* Calculate the total quantity sold for each respective month and the difference in total quantity sold (as percentage) between current and previous month

CHARTS REQUIREMENTS:

1. CALENDAR HEAT MAP

* A heat map that adjusts for the chosen month which is colour-coded that shows a darker shade for higher sales, lighter shade for lower sales

1. SALES ANALYSIS BY WEEKDAYS AND WEEKENDS

* Separates sales data into weekdays and weekends to see patterns

1. SALES ANALYSIS BY AGE PER MONTH WITH AVERAGE LINE

* Calculates the average sales by age for the chosen month which is also colour-coded, light orange for sales above average, light blue for sales below average

1. SALES BY GENDER

* Visualises the sales for each gender for that chosen month and shows the difference in sales as percentage between current and previous month

1. SALES BY PRODUCT CATEGORY

* Visualises the sales for each product category for that chosen month and shows the difference in sales as percentage between current and previous month

1. DAILY SALES ANALYSIS WITH AVERAGE LINE

* Calculates the daily average sales for the chosen month which is also colour-coded, light orange for sales above average, light blue for sales below average
* Note: the dashboard should be filtered by month

SQL QUERIES:

**UPDATING DATE (transaction\_date) COLUMN TO PROPER DATE FORMAT AND RENAMING IT TO sale\_date FOR EASY QUERYING**

UPDATE retail\_shop\_sales

SET transaction\_date = STR\_TO\_DATE(transaction\_date, '%d/%m/%Y');

ALTER TABLE retail\_shop\_sales

MODIFY COLUMN transaction\_date DATE;

ALTER TABLE retail\_shop\_sales

CHANGE COLUMN transaction\_date sale\_date DATE;

**ALTERING trans\_id to sale\_id FOR EASY QUERYING**

ALTER TABLE retail\_shop\_sales

CHANGE COLUMN trans\_id sale\_id INT;

**CHECKING IF THE QUERIES ARE SUCCESFUL**

DESCRIBE retail\_shop\_sales;

A screenshot of a computer

Description automatically generated

SELECT \* FROM retail\_shop\_sales;

A screenshot of a table

Description automatically generated

**-- TOTAL SALES FOR MONTH SELECTED**

SELECT ROUND(SUM(total\_amount)) AS total\_sales

FROM retail\_shop\_sales

WHERE MONTH(sale\_date) = 2; -- Februrary

A screenshot of a computer

Description automatically generated

**-- THE DIFFERENCE OF TOTAL SALES FROM CURRENT MONTH TO PREVIOUS MONTH SHOWN AS A PERCENTAGE WITH THE USE OF LAG()**

**-- note that january 2023 will return null as it has no previous month to be comapred to**

SELECT

MONTH(sale\_date) AS month,

ROUND(SUM(total\_amount)) AS total\_sales,

(SUM(total\_amount) - LAG(SUM(total\_amount),1)

OVER (ORDER BY MONTH(sale\_date))) / LAG(SUM(total\_amount),1)

OVER (ORDER BY MONTH(sale\_date)) \* 100 AS mon\_to\_mon\_percent

FROM retail\_shop\_sales

WHERE MONTH(sale\_date) IN (3,4) – March and April

GROUP BY MONTH(sale\_date)

ORDER BY MONTH(sale\_date);

A screenshot of a computer

Description automatically generated

**-- TOTAL ORDERS FOR MONTH SELECTED**

SELECT COUNT(sale\_id) AS total\_orders

FROM retail\_shop\_sales

WHERE MONTH(sale\_date) = 1;

A screenshot of a computer

Description automatically generated

**-- THE DIFFERENCE OF TOTAL ORDERS FROM CURRENT MONTH TO PREVIOUS MONTH SHOWN AS A PERCENTAGE WITH THE USE OF LAG()**

SELECT

MONTH(sale\_date) AS month,

ROUND(COUNT(sale\_id)) as total\_orders,

(COUNT(sale\_id) - lag(COUNT(sale\_id),1)

OVER (ORDER BY MONTH(sale\_date))) / LAG(COUNT(sale\_id),1)

OVER (ORDER BY MONTH(sale\_date)) \* 100 as mon\_to\_mon\_percent

FROM retail\_shop\_sales

WHERE MONTH(sale\_date) IN (3,4)

GROUP BY MONTH(sale\_date)

ORDER BY MONTH(sale\_date);

A screenshot of a computer

Description automatically generated

**-- TOTAL QUANTITY FOR MONTH SELECTED**

SELECT SUM(quantity) as total\_quantity

FROM retail\_shop\_sales

WHERE MONTH(sale\_date) = 1;

A screenshot of a computer

Description automatically generated

**-- THE DIFFERENCE OF TOTAL QUANTITY FROM CURRENT MONTH TO PREVIOUS MONTH SHOWN AS A PERCENTAGE WITH THE USE OF LAG()**

SELECT

MONTH(sale\_date) AS month,

ROUND(SUM(quantity)) AS total\_quantity\_sold,

(SUM(quantity) - LAG(SUM(quantity), 1)

OVER (ORDER BY MONTH(sale\_date))) / LAG(SUM(quantity), 1)

OVER (ORDER BY MONTH(sale\_date)) \* 100 AS mon\_to\_mon\_percent

FROM retail\_shop\_sales

WHERE MONTH(sale\_date) IN (3,4)

GROUP BY MONTH(sale\_date)

ORDER BY MONTH(sale\_date);

A screenshot of a computer

Description automatically generated

**-- TOTAL SALES, TOTAL QUANTITY SOLD and TOTAL ORDERS FOR A SPECIFIC DAY**

SELECT

SUM(total\_amount) AS total\_sales,

SUM(quantity) AS total\_quantity\_sold,

COUNT(sale\_id) AS total\_orders

FROM retail\_shop\_sales

WHERE sale\_date = '2023-01-23';

A screenshot of a computer

Description automatically generated

-- **SALES TRENDLINE FOR MONTH SELECTED**

SELECT AVG(total\_sales) AS average\_sales

FROM

( SELECT SUM(total\_amount) AS total\_sales

FROM retail\_shop\_sales

WHERE MONTH(sale\_date) = 1

GROUP BY sale\_date

) AS internal\_query;

A screenshot of a graph

Description automatically generated

**-- DAILY SALES FOR MONTH SELECTED**

SELECT

DAY(sale\_date) AS day\_of\_month,

ROUND(SUM(total\_amount),1) AS total\_sales

FROM retail\_shop\_sales

WHERE MONTH(sale\_date) = 1

GROUP BY DAY(sale\_date)

ORDER BY DAY(sale\_date);

A screenshot of a data

Description automatically generated

**-- COMPARING DAILY SALES WITH AVERAGE SALES – IF GREATER THAN “ABOVE AVERAGE” and LESSER THAN “BELOW AVERAGE”**

SELECT

day\_of\_month,

CASE

WHEN total\_sales > average\_sales THEN 'Above Average'

WHEN total\_sales < average\_sales THEN 'Below Average'

ELSE 'Average'

END AS sales\_status,

total\_sales

FROM (

SELECT

DAY(sale\_date) AS day\_of\_month,

SUM(total\_amount) AS total\_sales,

AVG(SUM(total\_amount)) OVER () AS average\_sales

FROM

retail\_shop\_sales

WHERE

MONTH(sale\_date) = 1

GROUP BY

DAY(sale\_date)

) AS sales\_data

ORDER BY

day\_of\_month;

A screenshot of a data

Description automatically generated

**-- SALES BY WEEKDAY / WEEKEND FOR MONTH SELECTED**

SELECT

CASE

WHEN DAYOFWEEK(sale\_date) IN (1, 7) THEN 'Weekends'

ELSE 'Weekdays'

END AS day\_type,

ROUND(SUM(total\_amount),2) AS total\_sales

FROM

retail\_shop\_sales

WHERE

MONTH(sale\_date) = 1

GROUP BY

CASE

WHEN DAYOFWEEK(sale\_date) IN (1, 7) THEN 'Weekends'

ELSE 'Weekdays'

END;

A screenshot of a computer

Description automatically generated

**-- SALES BY PRODUCT CATEGORY FOR MONTH SELECTED**

SELECT

product\_category,

SUM(total\_amount) as total\_sales

FROM retail\_shop\_sales

WHERE

MONTH(sale\_date) = 1

GROUP BY product\_category

ORDER BY total\_sales DESC;

A screenshot of a computer

Description automatically generated

**-- SALES BY AGE FOR MONTH SELECTED**

SELECT

age,

SUM(total\_amount) as total\_sales

FROM retail\_shop\_sales

WHERE

MONTH(sale\_date) = 1

GROUP BY age

ORDER BY total\_sales DESC;

A screenshot of a computer

Description automatically generated

**-- SALES BY GENDER FOR MONTH SELECTED**

SELECT

gender,

SUM(total\_amount) as total\_sales

FROM retail\_shop\_sales

WHERE

MONTH(sale\_date) = 1

GROUP BY gender

ORDER BY total\_sales DESC;

A screenshot of a computer

Description automatically generated

**-- SALES BY SPECIFIC DAY OF MONTH**

SELECT

ROUND(SUM(total\_amount)) AS total\_sales,

SUM(quantity) AS total\_quantity,

COUNT(\*) AS total\_orders

FROM

retail\_shop\_sales

WHERE

DAYOFWEEK(sale\_date) = 1 -- Filter for Tuesday (1 is Sunday, 2 is Monday, ..., 7 is Saturday)

AND MONTH(sale\_date) = 1;

A screenshot of a computer

Description automatically generated

**-- TO GET SALES FROM MONDAY TO SUNDAY FOR MONTH SELECTED**

SELECT

CASE

WHEN DAYOFWEEK(sale\_date) = 2 THEN 'Monday'

WHEN DAYOFWEEK(sale\_date) = 3 THEN 'Tuesday'

WHEN DAYOFWEEK(sale\_date) = 4 THEN 'Wednesday'

WHEN DAYOFWEEK(sale\_date) = 5 THEN 'Thursday'

WHEN DAYOFWEEK(sale\_date) = 6 THEN 'Friday'

WHEN DAYOFWEEK(sale\_date) = 7 THEN 'Saturday'

ELSE 'Sunday'

END AS Day\_of\_Week,

ROUND(SUM(total\_amount)) AS total\_sales

FROM

retail\_shop\_sales

WHERE

MONTH(sale\_date) = 1

GROUP BY

CASE

WHEN DAYOFWEEK(sale\_date) = 2 THEN 'Monday'

WHEN DAYOFWEEK(sale\_date) = 3 THEN 'Tuesday'

WHEN DAYOFWEEK(sale\_date) = 4 THEN 'Wednesday'

WHEN DAYOFWEEK(sale\_date) = 5 THEN 'Thursday'

WHEN DAYOFWEEK(sale\_date) = 6 THEN 'Friday'

WHEN DAYOFWEEK(sale\_date) = 7 THEN 'Saturday'

ELSE 'Sunday'

END;

A screenshot of a computer

Description automatically generated