Walmart Sales Data Analysis – Technical Documentation

Created By-Naveen Kumar S

1. Project Overview

- The Walmart Sales Data Analysis project focuses on analyzing retail sales transactions to identify business insights related to product performance, customer behavior, and sales trends.
- The dataset was sourced from the Kaggle Walmart Sales Forecasting competition and contains historical sales records from multiple Walmart branches where I have just changed the city names to CHENNAI,BANGALORE,MUMBAI.
- The objective of this project is to help business stakeholders understand sales performance and optimize strategies using data-driven insights.

2. Tools & Technologies Used

- **Database:** MySQL Workbench
- **Data Analysis:** SQL (MySQL)
- **Data Visualization:** Power BI
- **Dataset Source:** Kaggle Walmart Sales Dataset

3. Dataset Description

The dataset contains **1000 records** and **17 columns** including:

Column	Description	Data Type
invoice_id	Invoice of the sales made	VARCHAR(30)
branch	Branch at which sales were made	VARCHAR(5)
city	The location of the branch	VARCHAR(30)
customer_type	The type of the customer	VARCHAR(30)

gender	Gender of the customer making purchase	VARCHAR(10)
product_line	Product line of the product	VARCHAR(100)
unit_price	The price of each product	DECIMAL(10, 2)
quantity	Quantity of Products sold	INT
VAT	Value added tax	FLOAT(6, 4)
total	The total cost of the purchase	DECIMAL(10, 2)
date	The date on which the purchase was made	DATE
time	The time at which the purchase was made	TIME
payment_method	The total amount paid	VARCHAR(50)
cogs	Cost Of Goods sold	DECIMAL(10, 2)
gross_margin_percentage	Gross margin percentage	FLOAT(11, 9)
gross_income	Gross Income	DECIMAL(10, 2)
rating	Rating	FLOAT(2, 1)
invoice_id	Invoice of the sales made	VARCHAR(30)

Approach Used

4. Data Preparation (Data Wrangling)

Steps performed:

- Created MySQL database and table schema.
- Imported CSV dataset into MySQL Workbench.
- Applied **NOT NULL constraints** to ensure data quality.
- Checked for missing or NULL values.
- Standardized numeric formats and data types.

5. Feature Engineering

This will help use generate some new columns from existing ones.

- Add a new column named `time_of_day` to give insight of sales in the Morning, Afternoon and Evening. This will help answer the question on which part of the day most sales are made.
- Add a new column named `day_name` that contains the extracted days of the week on which the given transaction took place (Mon, Tue, Wed, Thur, Fri). This will help answer the question on which week of the day each branch is busiest.
- Add a new column named `month_name` that contains the extracted months of the year on which the given transaction took place (Jan, Feb, Mar). Help determine which month of the year has the most sales and profit.

6.Exploratory Data Analysis (EDA): Exploratory data analysis is done to answer the listed questions and aims of this project.

The questions includes sections like **Business generic,Product,Sales,Customers**

Pls check the end of the document for all the questions and SQL queries .

7. Power BI Dashboard Development: (Check the dashboard image at Page 5)

The MySQL database was connected directly to Power BI using database connector.

KPIs Created

- Total Revenue
- Total Profit (Gross Income)
- Total Orders
- Total Quantity Sold
- Average Customer Rating

Visualizations

- Revenue by Month
- Revenue by Product Line
- Revenue by City
- Revenue by Customer Type
- Revenue by Gender
- Sales Trend Analysis

Interactive Features

Slicers for:

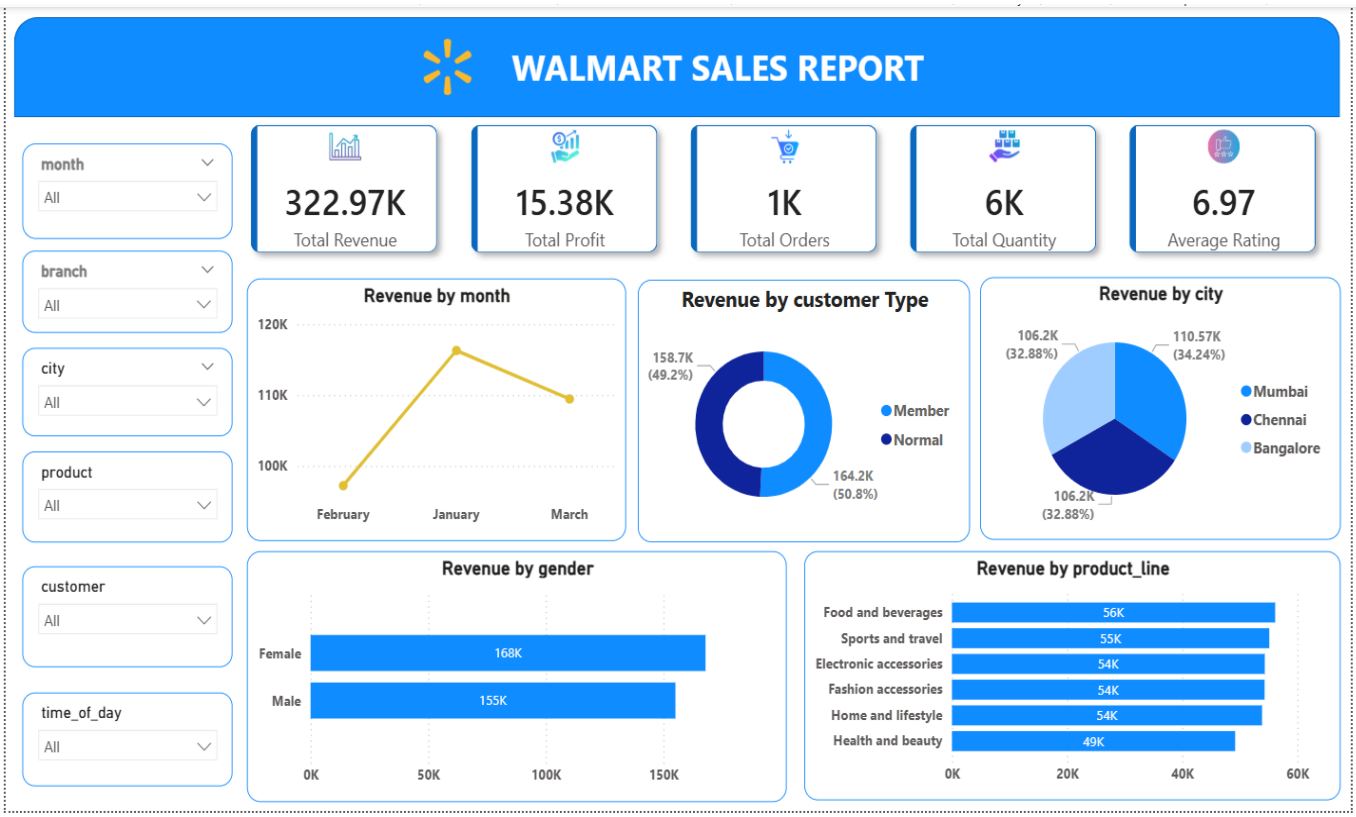
- City
- Product Line
- Customer Type
- Month

8. Key Business Insights

- Identified top performing product categories.
- Observed peak sales periods during specific times of the day.
- Member customers generated higher revenue.
- Certain cities contributed significantly to overall profit.

9. Conclusion

The project successfully demonstrated how SQL and Power BI can be combined to transform raw transactional data into actionable business insights. The dashboard enables decision-makers to monitor performance and improve sales strategies.



SQL Queries

-- Walmart database creation

CREATE DATABASE walmart_sales;

USE walmart_sales;

-- Create table for sales

CREATE TABLE sales(

invoice_id VARCHAR(30) NOT NULL PRIMARY KEY,

branch VARCHAR(5) NOT NULL,

city VARCHAR(30) NOT NULL,

customer_type VARCHAR(30) NOT NULL,

gender VARCHAR(10) NOT NULL,

product_line VARCHAR(100) NOT NULL,

unit_price DECIMAL(10,2) NOT NULL,

quantity INT NOT NULL,

```
VAT FLOAT(6,4),
total DECIMAL(10,2),
date DATE NOT NULL,
time TIME NOT NULL,
payment_method VARCHAR(50) NOT NULL,
cogs DECIMAL(10,2) NOT NULL,
gross_margin_percentage FLOAT(11,9) NOT NULL,
gross_income DECIMAL(10,2) NOT NULL,
rating FLOAT(3,1)
);
```

-- Feature Engineering

```
-- Add a new column named time_of_day
SELECT time,
(CASE
WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
 WHEN `time` BETWEEN "12:00:00" AND "16:00:00" THEN "Afternoon"
 ELSE "Evening"
 END)AS time_of_date
from Sales;
ALTER TABLE sales
ADD  COLUMN time_of_day VARCHAR(20);

UPDATE sales
SET time_of_day=(
CASE
WHEN `time` BETWEEN "00:00:00" AND "12:00:00" THEN "Morning"
 WHEN `time` BETWEEN "12:00:00" AND "16:00:00" THEN "Afternoon"
 ELSE "Evening"
 END
```

);

-- Add a new column named day_name

SELECT date,

DAYNAME(date)

FROM sales;

ALTER TABLE sales

ADD COLUMN day_name VARCHAR(20);

UPDATE sales

SET day_name=DAYNAME(date);

-- Add a new column named month_name

SELECT date,

MONTHNAME(date)

FROM sales;

ALTER TABLE sales

ADD COLUMN month_name VARCHAR(20);

UPDATE sales

SET month_name=MONTHNAME(date);

-- Business Questions To Answer

-- Generic Question

-- How many unique cities does the data have?

SELECT DISTINCT city FROM sales;

SELECT COUNT(DISTINCT city) AS city_count FROM sales;

-- In which city is each branch?

SELECT DISTINCT city,branch FROM sales;

-- Product

-- How many unique product lines does the data have?

```
SELECT DISTINCT product_line FROM SALES;

SELECT COUNT(DISTINCT product_line) AS product_line_count FROM SALES;
```

-- What is the most common payment method?

```
SELECT payment_method,

COUNT( payment_method) AS COUNT FROM sales

GROUP BY payment_method

ORDER BY count DESC;
```

-- What is the most selling product line?

```
SELECT product_line,

COUNT( product_line) AS COUNT FROM sales

GROUP BY product_line

ORDER BY count DESC;
```

-- What is the total revenue by month?

```
SELECT month_name,

SUM(total)AS total_revenue

 FROM sales

 GROUP BY month_name

ORDER BY total_revenue DESC;
```

-- What month had the largest COGS?

```
SELECT

 month_name,

SUM(cogs)AS Cogs

FROM sales

GROUP BY month_name

ORDER BY cogs DESC;
```

-- What product line had the largest revenue?

```
SELECT product_line,
SUM(total)AS largest_revenue
FROM sales
GROUP BY product_line
ORDER BY largest_revenue DESC;
```

-- What is the city with the largest revenue?

```
SELECT city,
SUM(total)AS largest_revenue
FROM sales
GROUP BY city
ORDER BY largest_revenue DESC;
```

-- What product line had the largest VAT?

```
SELECT product_line,
MAX(VAT) AS VAT
FROM sales
GROUP BY product_line
ORDER BY VAT DESC;
```

-- Which branch sold more products than average product sold?

```
SELECT branch,
SUM(quantity) as quantity
FROM sales
GROUP BY branch
HAVING SUM(quantity)>(SELECT AVG(quantity) AS product_sold FROM sales);
```

-- What is the most common product line by gender?

```
SELECT gender,product_line,
COUNT(gender)AS total_count
FROM sales
GROUP BY gender,product_line
ORDER BY total_count DESC;
```

-- What is the average rating of each product line?

```
SELECT product_line,
AVG(rating) AS avg_rating
FROM sales
GROUP BY product_line
ORDER BY avg_rating DESC;
```

-- Fetch each product line and add a column to those product line showing "Good", "Bad". Good if its greater than average sales

```
SELECT
   product_line,
   SUM(total) AS total_sales,
   CASE
      WHEN SUM(total) > (SELECT AVG(total) FROM sales)
      THEN 'Good'
      ELSE 'Bad'
   END AS performance
FROM sales
GROUP BY product_line;
```

-- Sales

-- Number of sales made in each time of the day per weekday

```
SELECT time_of_day,
COUNT(*) AS total_sales
FROM sales
WHERE day_name="Monday"
```

```
GROUP BY time_of_day
ORDER BY total_sales DESC;

-- Which of the customer types brings the most revenue?
SELECT customer_type,
SUM(total) AS most_revenue
FROM sales
GROUP BY customer_type
ORDER BY most_revenue DESC;

-- Which city has the largest tax percent/ VAT (Value Added Tax)?
SELECT city,
MAX(VAT) AS VAT
FROM sales
GROUP BY city
ORDER BY VAT DESC;

-- Which customer type pays the most in VAT?
SELECT customer_type,
SUM(VAT) AS VAT
FROM sales
GROUP BY customer_type
ORDER BY VAT DESC;

-----------------------------------------------------------------------------------------------------------------------------------------------

-- Customers

-- How many unique customer types does the data have?
SELECT  DISTINCT customer_type FROM sales;
SELECT COUNT( DISTINCT customer_type) AS no_of_customers FROM sales;
```

-- How many unique payment methods does the data have?

```
SELECT  DISTINCT payment_method FROM sales;

SELECT COUNT( DISTINCT payment_method )AS no_of_payment_method FROM sales;
```

-- What is the most common customer type?

```
SELECT customer_type,

COUNT(*) AS customer

FROM sales

GROUP BY customer_type

ORDER BY customer DESC;
```

-- What is the gender of most of the customers?

```
SELECT gender,

COUNT(*) AS count_of_gender

FROM sales

GROUP BY gender

ORDER BY count_of_gender DESC;
```

-- What is the gender distribution per branch?

```
SELECT gender,

COUNT(*) AS count_of_gender

FROM sales

WHERE branch="A"

GROUP BY gender

ORDER BY count_of_gender DESC;
```

-- Which time of the day do customers give most ratings?

```
SELECT time_of_day,

AVG(rating) AS no_of_ratings

FROM sales
```

```
GROUP BY time_of_day
ORDER BY no_of_ratings DESC;

-- Which time of the day do customers give most ratings per branch?
SELECT time_of_day,
AVG(rating) AS no_of_ratings
FROM sales
WHERE branch="C"
GROUP BY time_of_day
ORDER BY no_of_ratings DESC;

-- Which day of the week has the best avg ratings?
SELECT day_name,
 AVG(rating) AS rating
 from sales
 GROUP BY day_name
 ORDER BY rating DESC;

-- Which day of the week has the best average ratings per branch?
 SELECT day_name,
 AVG(rating) AS rating
 from sales
 WHERE branch="B"
 GROUP BY day_name
 ORDER BY rating DESC;
```