



Walmart



Sales Performance Analysis of Walmart Stores Using MySQL

By- Kaustav Chakraborty

INTRODUCTION

Walmart, a major retail chain, operates across several cities, offering a wide range of products. The dataset provided contains detailed transaction data, including customer demographics, product lines, sales figures, and payment methods. This project will use advanced SQL techniques to uncover actionable insights into sales performance, customer behavior, and operational efficiencies



BUSINESS PROBLEMS

Walmart wants to optimize its sales strategies by analyzing historical transaction data across branches, customer types, payment methods, and product lines. To achieve this, advanced MySQL queries will be employed to answer challenging business questions related to sales performance, customer segmentation, and product trends

TASK 1: IDENTIFYING THE TOP BRANCH BY SALES GROWTH RATE (6 MARKS)

WALMART WANTS TO IDENTIFY WHICH BRANCH HAS EXHIBITED THE HIGHEST SALES GROWTH OVER TIME. ANALYZE THE TOTAL SALES FOR EACH BRANCH AND COMPARE THE GROWTH RATE ACROSS MONTHS TO FIND THE TOP PERFORMER

```
WITH MonthlyGrowth AS (  
    SELECT  
        Branch,  
        DATE_FORMAT(STR_TO_DATE(Date, '%Y-%m-%d'), '%Y-%m') AS Month,  
        SUM(Total) AS Monthly_Total,  
        LAG(SUM(Total)) OVER (PARTITION BY Branch ORDER BY DATE_FORMAT(STR_TO_DATE(Date, '%Y-%m-%d'), '%Y-%m')) AS Previous_Month_Total  
    FROM walmart  
    WHERE Total IS NOT NULL AND Date IS NOT NULL  
    GROUP BY Branch, Month  
)  
SELECT  
    Branch,  
    AVG((Monthly_Total - Previous_Mnth_Total) / NULLIF(Previous_Month_Total, 0) * 100) AS Avg_Growth_Rate  
FROM MonthlyGrowth  
WHERE Previous_Month_Total IS NOT NULL  
GROUP BY Branch  
ORDER BY Avg_Growth_Rate DESC  
LIMIT 1;
```

TASK 2: FINDING THE MOST PROFITABLE PRODUCT LINE FOR EACH BRANCH

WALMART NEEDS TO DETERMINE WHICH PRODUCT LINE CONTRIBUTES THE HIGHEST PROFIT TO EACH BRANCH. THE PROFIT MARGIN SHOULD BE CALCULATED BASED ON THE DIFFERENCE BETWEEN THE GROSS INCOME AND COST OF GOODS SOLD

```
SELECT
    branch,
    `product line`,
    (`gross income` - `cogs`) AS profit_margin
FROM walmart w1
WHERE (`gross income` - `cogs`) = (
    SELECT MAX(`gross income` - `cogs`)
    FROM walmart w2
    WHERE w1.branch = w2.branch
    AND w1.`product line` = w2.`product line`
)
GROUP BY branch, `product line`, `gross income`, `cogs`;
```

TASK 3: ANALYZING CUSTOMER SEGMENTATION BASED ON SPENDING (6 MARKS)

WALMART WANTS TO SEGMENT CUSTOMERS BASED ON THEIR AVERAGE SPENDING BEHAVIOR. CLASSIFY CUSTOMERS INTO THREE TIERS: HIGH, MEDIUM, AND LOW SPENDERS BASED ON THEIR TOTAL PURCHASE AMOUNTS

```
3 WITH CustomerTotalSpending AS (  
4     SELECT  
5         `Customer ID`,  
6         SUM(`Total`) AS TotalSpending  
7     FROM walmart  
8     GROUP BY `Customer ID`  
9 ),  
10 RankedCustomers AS (  
11     SELECT  
12         `Customer ID`,  
13         TotalSpending,  
14         NTILE(3) OVER (ORDER BY TotalSpending DESC) AS SpendingRank  
15     FROM CustomerTotalSpending  
16 )
```

```
SELECT  
    `Customer ID`,  
    TotalSpending,  
    CASE  
        WHEN SpendingRank = 1 THEN 'High'  
        WHEN SpendingRank = 2 THEN 'Medium'  
        ELSE 'Low'  
    END AS SpendingTier  
FROM RankedCustomers;
```

TASK 4: DETECTING ANOMALIES IN SALES TRANSACTIONS (6 MARKS) WALMART

SUSPECTS THAT SOME TRANSACTIONS HAVE UNUSUALLY HIGH OR LOW SALES COMPARED TO THE AVERAGE FOR THE PRODUCT LINE. IDENTIFY THESE ANOMALIES

```
WITH ProductLineStats AS (  
    -- Calculate the average sales for each product line  
    SELECT  
        `Product line`,  
        AVG(`Total`) AS avg_sales  
    FROM walmart  
    GROUP BY `Product line`  
),  
FlaggedTransactions AS (  
    -- Flag transactions as anomalies if they are more than 30% above or below the average sales  
    SELECT w.`Invoice ID`, w.`Branch`, w.`Product line`, w.`Total`,  
        CASE  
            WHEN w.`Total` > (p.avg_sales * 1.30) OR w.`Total` < (p.avg_sales * 0.70) THEN 'Anomaly'  
            ELSE 'Normal'  
        END AS Status  
    FROM walmart w  
    JOIN ProductLineStats p  
    ON w.`Product line` = p.`Product line`  
)
```

```
    JOIN ProductLineStats p  
    ON w.`Product line` = p.`Product line`  
)  
SELECT  
    `Invoice ID`,  
    `Branch`,  
    `Product line`,  
    `Total`,  
    Status  
FROM FlaggedTransactions  
WHERE Status = 'Anomaly';
```

TASK 5: MOST POPULAR PAYMENT METHOD BY CITY (6 MARKS) WALMART NEEDS TO DETERMINE THE MOST POPULAR PAYMENT METHOD IN EACH CITY TO TAILOR MARKETING STRATEGIES

```
select * from walmart;
```

```
SELECT
    City,
    Payment,
    COUNT(*) AS PaymentCount
FROM walmart
GROUP BY City, Payment
ORDER BY City, PaymentCount DESC;
```

Result Grid				Filter Rows:	Exp
	City	Payment	PaymentCount		
▶	Mandalay	Ewallet	113		
	Mandalay	Cash	110		
	Mandalay	Credit card	109		
	Naypyitaw	Cash	124		
	Naypyitaw	Ewallet	106		
	Naypyitaw	Credit card	98		
	Yangon	Ewallet	126		
	Yangon	Cash	110		
	Yangon	Credit card	104		

TASK 6: MONTHLY SALES DISTRIBUTION BY GENDER WALMART WANTS TO UNDERSTAND THE SALES DISTRIBUTION BETWEEN MALE AND FEMALE CUSTOMERS ON A MONTHLY BASIS

SELECT

```
YEAR(STR_TO_DATE(`date`, '%d-%m-%Y')) AS Year,  
MONTH(STR_TO_DATE(`date`, '%d-%m-%Y')) AS Month,  
`gender`,  
SUM(`total`) AS TotalSales
```

FROM walmart

WHERE `date` IS NOT NULL

```
GROUP BY YEAR(STR_TO_DATE(`date`, '%d-%m-%Y')), MONTH(STR_TO_DATE(`date`, '%d-%m-%Y')),  
ORDER BY Year, Month, `gender`;
```

	Year	Month	gender	TotalSales
►	2019	1	Female	59138.982000000001
	2019	1	Male	57152.885999999999
	2019	2	Female	56335.555499999999
	2019	2	Male	40883.818499999999
	2019	3	Female	52408.3875000000004
	2019	3	Male	57047.119499999997

TASK 7: BEST PRODUCT LINE BY CUSTOMER TYPE (6 MARKS) WALMART WANTS TO KNOW WHICH PRODUCT LINES ARE PREFERRED BY DIFFERENT CUSTOMER TYPES(MEMBER VS. NORMAL)

```
179 • WITH ProductLineCounts AS (  
180     SELECT  
181         `product line`,  
182         `customer type`,  
183         COUNT(*) AS count_product_line  
184     FROM walmart  
185     GROUP BY `customer type`, `product line`  
186 )  
187 SELECT  
188     `customer type`,  
189     `product line`,  
190     count_product_line  
191 FROM ProductLineCounts  
192 WHERE count_product_line = (  
193     SELECT MAX(count_product_line)  
194     FROM ProductLineCounts AS sub  
195     WHERE sub.`customer type` = ProductLineCounts.`customer type`);
```

Result Grid Filter Rows: Export:		
customer type	product line	count_product_line
Normal	Electronic accessories	92
Member	Food and beverages	94
Normal	Fashion accessories	92

TASK 8: IDENTIFYING REPEAT CUSTOMERS WALMART NEEDS TO IDENTIFY CUSTOMERS WHO MADE REPEAT PURCHASES WITHIN A SPECIFIC TIME FRAME (E.G., WITHIN 30 DAYS)

```
SELECT DISTINCT
    w1.`Customer ID`,
    w1.`Invoice ID` AS First_Purchase,
    w2.`Invoice ID` AS Repeat_Purchase,
    DATEDIFF(STR_TO_DATE(w2.`Date`, '%Y-%m-%d'), STR_TO_DATE(w1.`Date`, '%Y-%m-%d')) AS Days_Between_Purchase
FROM walmart w1
JOIN walmart w2
    ON w1.`Customer ID` = w2.`Customer ID`
    AND w1.`Invoice ID` != w2.`Invoice ID` -- Ensure different invoices
    AND DATEDIFF(STR_TO_DATE(w2.`Date`, '%Y-%m-%d'), STR_TO_DATE(w1.`Date`, '%Y-%m-%d')) BETWEEN 1 AND 30 --
ORDER BY w1.`Customer ID`, Days_Between_Purchases;
```

TASK 9: FINDING TOP 5 CUSTOMERS BY SALES VOLUME

WALMART WANTS TO REWARD ITS TOP 5 CUSTOMERS WHO HAVE GENERATED THE MOST SALES REVENUE

SELECT

`Customer ID`,

SUM(`Quantity` * `unit Price`) AS revenue

FROM walmart

GROUP BY `Customer ID`

ORDER BY revenue DESC

LIMIT 5;

RESULT SET 1		Filter Results
	Customer ID	revenue
▶	8	25366.040000000005
	3	22287.870000000006
	2	22278.360000000004
	15	21594.720000000005
	1	21556.710000000001

TASK 10: ANALYZING SALES TRENDS BY DAY OF THE WEEK WALMART WANTS TO ANALYZE THE SALES PATTERNS TO DETERMINE WHICH DAY OF THE WEEK BRINGS THE HIGHEST SALES.

SELECT

```
DAYNAME(STR_TO_DATE(`Date`, '%Y-%m-%d')) AS Day_of_Week,  -- Extract day of the week  
SUM(`Quantity` * `unit Price`) AS Total_Sales  -- Calculate total sales
```

FROM walmart

GROUP BY Day_of_Week

ORDER BY Total_Sales **DESC**;

Day_of_Week	Total_Sales
Wednesday	49137.9200000000006
Friday	48308.0299999999984
Thursday	43102.8200000000001
Sunday	42668.3699999999995
Monday	42447.5799999999994
Saturday	41918.5699999999985
Tuesday	40004.09



THANK YOU