

Retail Sales Analysis Using SQL

Data Analytics Project Report

Prepared by: Jitendrakumar Choudhary

Executive Summary:-

This project analyzes retail sales data using SQL to extract insights that help improve profitability and customer satisfaction. The analysis focuses on product performance, revenue trends, customer behavior, and regional sales patterns. The findings support better inventory management, marketing strategies, and discount planning.

Problem Statement:-

A retail company wants to analyze its sales performance to improve profitability and customer satisfaction. The management is seeking insights into:

- Which products are the best sellers and which are underperforming.
- Revenue trends over time (monthly, quarterly, yearly).
- Customer purchasing behavior (top customers, average order value).
- Impact of discounts on sales and profitability.
- Regional sales performance.

Business Questions:-

1. Total revenue, total orders, and total customers
2. Top 10 best-selling products
3. Monthly revenue trend for the last 12 months
4. Average order value per customer
5. Top 5 customers by revenue
6. Sales contribution by region
7. Products with the highest discount usage
8. Category-wise sales and profit

Dataset Description:-

The project uses four key tables from the retail company's database:

- **Customers:** customer_id, name, region, signup_date
- **Products:** product_id, product_name, category, price
- **Orders:** order_id, customer_id, order_date, region
- **Order_Details:** order_detail_id, order_id, product_id, quantity, discount

1. Total revenue, total orders, and total customers

```
SELECT SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue,  
count(distinct c.customer_id) as total_customers,  
COUNT(DISTINCT o.order_id) AS total_orders  
FROM Order_Details od  
JOIN Orders o ON od.order_id = o.order_id  
join customers c on o.customer_id = c.customer_id  
JOIN Products p ON od.product_id = p.product_id;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
8 count(distinct c.customer_id) as total_customers,  
9 COUNT(DISTINCT o.order_id) AS total_orders  
10 FROM Order_Details od  
11 JOIN Orders o ON od.order_id = o.order_id  
12 join customers c on o.customer_id = c.customer_id  
13 JOIN Products p ON od.product_id = p.product_id;  
14  
15 # Top 10 best-selling products
```

The Results Grid shows the following data:

total_revenue	total_customers	total_orders
13169639.180099977	500	10000

The Output tab shows the execution log for Result 11:

#	Time	Action	Message	Duration / Fetch
11	19:44:35	select c.customer_id, c.name, SUM(od.quantity * p.price * (1 - od.discount)) AS revenue	FROM Orders o JOIN...	Error Code: 1055. Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated colu... 0.000 sec
12	19:45:43	select c.customer_id, c.name, SUM(od.quantity * p.price * (1 - od.discount)) AS revenue	FROM Orders o JOIN...	5 row(s) returned 2.234 sec / 0.000 sec
13	19:47:17	SELECT o.region, SUM(od.quantity * p.price * (1 - od.discount)) AS region_revenue, ROUND(SUM(od.qua...		4 row(s) returned 0.313 sec / 0.000 sec
14	19:55:25	SELECT p.product_name, COUNT(*) AS discount_count, SUM(od.quantity * p.price * od.discount) AS total...		10 row(s) returned 0.047 sec / 0.000 sec
15	19:55:54	SELECT p.category, SUM(od.quantity * p.price * (1 - od.discount)) AS total_sales, SUM(od.quantity * p.pric...		5 row(s) returned 0.078 sec / 0.000 sec
16	19:58:04	SELECT SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue, count(distinct c.customer_id) as total...		1 row(s) returned 1.453 sec / 0.000 sec

2. Top 10 best-selling products

```
SELECT p.product_name, SUM(od.quantity) AS total_quantity
FROM Order_Details od
JOIN Products p ON od.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_quantity DESC
LIMIT 10;
```

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with 'retail_sales_analysis' selected. The main editor window shows the following SQL query:

```
# Top 10 best-selling products
SELECT p.product_name, SUM(od.quantity) AS total_quantity
FROM Order_Details od
JOIN Products p ON od.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_quantity DESC
LIMIT 10;
```

The 'Result Grid' shows the following data:

product_name	total_quantity
Situation	784
Body	708
Blue	631
Paper	624
Want	622
Nor	622
Bag	619
Big	618
Few	604
Speech	589

The 'Output' tab at the bottom shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
12	19:45:43	select c.customer_id, c.name, SUM(od.quantity * p.price * (1 - od.discount)) AS revenue FROM Orders o JOIN Customers c ON o.customer_id = c.customer_id	5 row(s) returned	2.234 sec / 0.000 sec
13	19:47:17	SELECT o.region, SUM(od.quantity * p.price * (1 - od.discount)) AS region_revenue, ROUND(SUM(od.quantity * p.price * (1 - od.discount))) AS total_revenue FROM Orders o JOIN Customers c ON o.customer_id = c.customer_id	4 row(s) returned	0.313 sec / 0.000 sec
14	19:55:25	SELECT p.product_name, COUNT(*) AS discount_count, SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue FROM Products p JOIN Order_Details od ON p.product_id = od.product_id	10 row(s) returned	0.047 sec / 0.000 sec
15	19:55:54	SELECT p.category, SUM(od.quantity * p.price * (1 - od.discount)) AS total_sales, SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue FROM Products p JOIN Order_Details od ON p.product_id = od.product_id	5 row(s) returned	0.078 sec / 0.000 sec
16	19:58:04	SELECT SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue, count(distinct c.customer_id) as total_customers FROM Order_Details od JOIN Customers c ON od.customer_id = c.customer_id	1 row(s) returned	1.453 sec / 0.000 sec
17	19:58:45	SELECT p.product_name, SUM(od.quantity) AS total_quantity FROM Order_Details od JOIN Products p ON od.product_id = p.product_id	10 row(s) returned	0.078 sec / 0.000 sec

3. Monthly revenue trend for the last 12 months

```
SELECT DATE_FORMAT(o.order_date, '%Y-%m') AS month,  
SUM(od.quantity * p.price * (1 - od.discount)) AS revenue  
FROM Orders o  
JOIN Order_Details od ON o.order_id = od.order_id  
JOIN Products p ON od.product_id = p.product_id  
GROUP BY month  
ORDER BY month desc limit 12;
```

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
# Monthly revenue trend for the last 12 months  
SELECT DATE_FORMAT(o.order_date, '%Y-%m') AS month,  
SUM(od.quantity * p.price * (1 - od.discount)) AS revenue  
FROM Orders o  
JOIN Order_Details od ON o.order_id = od.order_id  
JOIN Products p ON od.product_id = p.product_id  
GROUP BY month  
ORDER BY month desc limit 12;
```

The Results Grid shows the following data:

month	revenue
2025-07	395474.13459999999
2025-06	577351.80259999992
2025-05	537093.55530000006
2025-04	960889.9520000002
2025-03	539096.1342000002
2025-02	500655.3510000002
2025-01	569492.12599999992
2024-12	588390.87830000004
2024-11	527391.3407000003
2024-10	516854.18050000013
2024-09	548519.4061000003
2024-08	610749.0015000006

The Output tab shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
13	19:47:17	SELECT o.region, SUM(od.quantity * p.price * (1 - od.discount)) AS region_revenue, ROUND(SUM(od.quantity * p.price * (1 - od.discount)) AS total_sales, 2) AS total_revenue FROM Orders o JOIN Order_Details od ON o.order_id = od.order_id JOIN Products p ON od.product_id = p.product_id GROUP BY o.region	4 row(s) returned	0.313 sec / 0.000 sec
14	19:55:25	SELECT p.product_name, COUNT(*) AS discount_count, SUM(od.quantity * p.price * (1 - od.discount)) AS total_sales, SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue FROM Products p JOIN Order_Details od ON p.product_id = od.product_id GROUP BY p.product_name	10 row(s) returned	0.047 sec / 0.000 sec
15	19:55:54	SELECT p.category, SUM(od.quantity * p.price * (1 - od.discount)) AS total_sales, SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue FROM Products p JOIN Order_Details od ON p.product_id = od.product_id GROUP BY p.category	5 row(s) returned	0.078 sec / 0.000 sec
16	19:58:04	SELECT SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue, count(distinct c.customer_id) as total_customers FROM Order_Details od JOIN Customers c ON od.customer_id = c.customer_id	1 row(s) returned	1.453 sec / 0.000 sec
17	19:58:45	SELECT p.product_name, SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue FROM Order_Details od JOIN Products p ON od.product_id = p.product_id	10 row(s) returned	0.078 sec / 0.000 sec
18	19:59:18	SELECT DATE_FORMAT(o.order_date, '%Y-%m') AS month, SUM(od.quantity * p.price * (1 - od.discount)) AS revenue FROM Orders o JOIN Order_Details od ON o.order_id = od.order_id JOIN Products p ON od.product_id = p.product_id GROUP BY month ORDER BY month desc limit 12	12 row(s) returned	0.234 sec / 0.000 sec

4. Average order value per customer

SELECT

c.customer_id,

c.name,

ROUND(SUM(od.quantity * p.price * (1 - od.discount)) / COUNT(DISTINCT o.order_id), 2) AS
avg_order_value

FROM Customers c

JOIN Orders o ON c.customer_id = o.customer_id

JOIN Order_Details od ON o.order_id = od.order_id

JOIN Products p ON od.product_id = p.product_id

GROUP BY c.customer_id, c.name

ORDER BY avg_order_value DESC;

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
# Average order value per customer
SELECT
  c.customer_id,
  c.name,
  ROUND(SUM(od.quantity * p.price * (1 - od.discount)) / COUNT(DISTINCT o.order_id), 2) AS avg_order_value
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
```

The Results Grid shows the following data:

customer_id	name	avg_order_value
136	Benjamin Serrano	2014.26
261	Cindy Alexander	1942.69
96	Michael Riggs	1928.14
482	Elizabeth Malone	1862.51
242	Mary Scott	1839.65
382	Christopher Yang	1806.62
231	Nicholas Lopez	1782.25
347	Barry Logan	1736.97
282	Paul Lam	1730.1
187	Madeline Vargas	1727.2
99	Megan Austin	1711.89
1900	Tyler Johnson	1699.84

The Output tab shows the execution log with the following messages:

#	Time	Action	Message	Duration / Fetch
14	19:55:25	SELECT p.product_name, COUNT(*) AS discount_count, SUM(od.quantity * p.price * od.discount) AS total...	10 row(s) returned	0.047 sec / 0.000 sec
15	19:55:54	SELECT p.category, SUM(od.quantity * p.price * (1 - od.discount)) AS total_sales, SUM(od.quantity * p.pric...	5 row(s) returned	0.078 sec / 0.000 sec
16	19:58:04	SELECT SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue, count(distinct c.customer_id) as total...	1 row(s) returned	1.453 sec / 0.000 sec
17	19:58:45	SELECT p.product_name, SUM(od.quantity) AS total_quantity FROM Order_Details od JOIN Products p ON ...	10 row(s) returned	0.078 sec / 0.000 sec
18	19:59:18	SELECT DATE_FORMAT(o.order_date, '%Y-%m') AS month, SUM(od.quantity * p.price * (1 - od.discount)) AS ...	12 row(s) returned	0.234 sec / 0.000 sec
19	19:59:57	SELECT c.customer_id, c.name, ROUND(SUM(od.quantity * p.price * (1 - od.discount)) / COUNT(DISTIN...	500 row(s) returned	2.328 sec / 0.000 sec

5. Top 5 customers by revenue

```
select c.customer_id,  
  
c.name,  
  
SUM(od.quantity * p.price * (1 - od.discount)) AS revenue  
  
FROM Orders o  
  
JOIN Order_Details od ON o.order_id = od.order_id  
  
JOIN Products p ON od.product_id = p.product_id  
  
join customers c on o.customer_id = c.customer_id  
  
GROUP BY c.customer_id, c.name  
  
order by revenue desc  
  
limit 5;
```

The screenshot shows the MySQL Workbench interface. The query editor contains the SQL query to find the top 5 customers by revenue. The results are displayed in a table with columns: customer_id, name, and revenue.

customer_id	name	revenue
106	Cynthia Johnson	46394.4422
481	Heather Gonzalez	45562.44300000001
118	Ms. Christine Gutierrez	44713.98599999999
403	Stephen Huynh	43649.59270000001
115	Angela Wiggins	43044.89630000001

The bottom panel shows the 'Action Output' tab with a list of executed queries and their durations.

#	Time	Action	Message	Duration / Fetch
15	19:55:54	SELECT p.category, SUM(od.quantity * p.price * (1 - od.discount)) AS total_sales, SUM(od.quantity * p.pr...	5 row(s) returned	0.078 sec / 0.000 sec
16	19:58:04	SELECT SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue, count(distinct c.customer_id) as total...	1 row(s) returned	1.453 sec / 0.000 sec
17	19:58:45	SELECT p.product_name, SUM(od.quantity) AS total_quantity FROM Order_Details od JOIN Products p ON ...	10 row(s) returned	0.078 sec / 0.000 sec
18	19:59:18	SELECT DATE_FORMAT(o.order_date, '%Y-%m') AS month, SUM(od.quantity * p.price * (1 - od.discount)) AS ...	12 row(s) returned	0.234 sec / 0.000 sec
19	19:59:57	SELECT c.customer_id, c.name, ROUND(SUM(od.quantity * p.price * (1 - od.discount)) / COUNT(DISTIN...	500 row(s) returned	2.328 sec / 0.000 sec
20	20:00:30	select c.customer_id, c.name, SUM(od.quantity * p.price * (1 - od.discount)) AS revenue FROM Orders o JOI...	5 row(s) returned	2.172 sec / 0.000 sec

6. Sales contribution by region

SELECT

o.region,

SUM(od.quantity * p.price * (1 - od.discount)) AS region_revenue,

ROUND(SUM(od.quantity * p.price * (1 - od.discount)) /

(SELECT SUM(od2.quantity * p2.price * (1 - od2.discount))

FROM Order_Details od2

JOIN Products p2 ON od2.product_id = p2.product_id) * 100, 2) AS contribution_percent

FROM Orders o

JOIN Order_Details od ON o.order_id = od.order_id

JOIN Products p ON od.product_id = p.product_id

GROUP BY o.region

ORDER BY region_revenue DESC;

The screenshot shows a database management tool interface with a SQL query editor and a results grid. The query is for calculating sales contribution by region. The results grid shows four rows of data for North, West, East, and South regions.

region	region_revenue	contribution_percent
North	3368708.6158000035	25.58
West	3314453.680200003	25.17
East	3281168.5729999957	24.91
South	3205308.3111000033	24.34

The bottom panel shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
16	19:58:04	SELECT SUM(od.quantity * p.price * (1 - od.discount)) AS total_revenue, count(distinct c.customer_id) as total...	1 row(s) returned	1.453 sec / 0.000 sec
17	19:58:45	SELECT p.product_name, SUM(od.quantity) AS total_quantity FROM Order_Details od JOIN Products p ON ...	10 row(s) returned	0.078 sec / 0.000 sec
18	19:59:18	SELECT DATE_FORMAT(o.order_date, '%Y-%m') AS month, SUM(od.quantity * p.price * (1 - od.discount)) AS ...	12 row(s) returned	0.234 sec / 0.000 sec
19	19:59:57	SELECT c.customer_id, c.name, ROUND(SUM(od.quantity * p.price * (1 - od.discount)) / COUNT(DISTIN...	500 row(s) returned	2.328 sec / 0.000 sec
20	20:00:30	select c.customer_id, c.name, SUM(od.quantity * p.price * (1 - od.discount)) AS revenue FROM Orders o JOI...	5 row(s) returned	2.172 sec / 0.000 sec
21	20:01:11	SELECT o.region, SUM(od.quantity * p.price * (1 - od.discount)) AS region_revenue, ROUND(SUM(od.qua...	4 row(s) returned	0.281 sec / 0.000 sec

7. Products with the highest discount usage

SELECT

p.product_name,

COUNT(*) AS discount_count,

SUM(od.quantity * p.price * od.discount) AS total_discount_value

FROM Order_Details od

JOIN Products p ON od.product_id = p.product_id

WHERE od.discount > 0

GROUP BY p.product_name

ORDER BY discount_count DESC

LIMIT 10;

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
77 # Products with the highest discount usage
78
79 SELECT
80     p.product_name,
81     COUNT(*) AS discount_count,
82     SUM(od.quantity * p.price * od.discount) AS total_discount_value
83 FROM Order_Details od
84 JOIN Products p ON od.product_id = p.product_id
```

The Results window displays the following data:

	product_name	discount_count	total_discount_value
▶ Situation	272	28037.4713000000	16
▶ Body	225	23941.1114	
▶ Else	209	18031.9937000000	2
▶ Want	209	37469.8566000000	14
▶ Bag	207	34671.7458999999	7
▶ Big	206	35210.9579000000	4
▶ Few	203	33352.6312000000	2
▶ Nor	203	41721.3921000000	5
▶ Paper	197	34892.1186	
▶ Response	197	35733.8631999999	85

The Output window shows the following message:

```
1 20:58:56 SELECT p product_name, COUNT(*) AS discount_count, SUM(od.quantity * p.price * od.discount) AS total_... 10 row(s) returned
0.094 sec / 0.000 sec
```


8. Category-wise sales and profit

SELECT

p.category,

SUM(od.quantity * p.price * (1 - od.discount)) AS total_sales,

SUM(od.quantity * p.price * (1 - od.discount)) * 0.25 AS estimated_profit

FROM Order_Details od

JOIN Products p ON od.product_id = p.product_id

GROUP BY p.category

ORDER BY total_sales DESC;

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
88 LIMIT 10;
89
90 # Category-wise sales and profit
91
92 SELECT
93     p.category,
94     SUM(od.quantity * p.price * (1 - od.discount)) AS total_sales,
95     SUM(od.quantity * p.price * (1 - od.discount)) * 0.25 AS estimated_profit
```

The Results tab shows the following data:

category	total_sales	estimated_profit
Beauty	348191.440800018	872047.8602000045
Sports	3034390.565000002	758597.6412500005
Electronics	2595551.438000006	648887.8595000015
Clothing	2445372.1416	611343.0354
Home	1606133.5946999984	401533.3986749996

The Output tab shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
18	19:59:18	SELECT DATE_FORMAT(o.order_date, '%Y-%m') AS month, SUM(od.quantity * p.price * (1 - od.discount)) AS ...	12 row(s) returned	0.234 sec / 0.000 sec
19	19:59:57	SELECT c.customer_id, c.name, ROUND(SUM(od.quantity * p.price * (1 - od.discount)) / COUNT(DISTIN...	500 row(s) returned	2.328 sec / 0.000 sec
20	20:00:30	select c.customer_id, c.name, SUM(od.quantity * p.price * (1 - od.discount)) AS revenue FROM Orders o JOI...	5 row(s) returned	2.172 sec / 0.000 sec
21	20:01:11	SELECT o.region, SUM(od.quantity * p.price * (1 - od.discount)) AS region_revenue, ROUND(SUM(od.qua...	4 row(s) returned	0.281 sec / 0.000 sec
22	20:01:38	SELECT p.product_name, COUNT(*) AS discount_count, SUM(od.quantity * p.price * od.discount) AS total...	10 row(s) returned	0.063 sec / 0.000 sec
23	20:01:38	SELECT p.category, SUM(od.quantity * p.price * (1 - od.discount)) AS total_sales, SUM(od.quantity * p.pric...	5 row(s) returned	0.046 sec / 0.000 sec

Insights and Recommendation:-

The best selling product is Situation with total quantity 784 and under performing product is money with total quantity 170.

The customer name Benjamin Serrano is top customer with average order value 2014.26.

There is not much difference in sales based on region.

The discount can be increase little to make sales more. As it shows more discount the sales is also more for the product.

Conclusion:-

The SQL analysis provided clear insights into product performance, customer behavior, and revenue trends. Implementing recommendations can enhance profitability and operational efficiency.