RetailCo Sales, Customer & Performance Analysis

End-to-End Data Analytics Project Using Python, MySQL & Power BI

1. Introduction

RetailCo is a multi-location retail business operating across different regions, customer segments, and sales channels. As the business scales, management faces challenges in understanding customer behavior, monitoring store and employee performance, controlling discount practices, and tracking revenue trends effectively.

This project was designed to simulate a **real-world retail analytics scenario**, where raw transactional data is transformed into meaningful insights that can support **strategic, operational, and tactical decision-making**.

The analysis combines **Python for data exploration and modeling**, **MySQL for structured business queries**, and **Power BI for interactive dashboards**, delivering a complete end-to-end analytics solution.

2. Business Objectives

Primary Objectives

- Drive business growth by increasing total revenue
- Optimize store and employee performance
- Improve customer retention and reduce churn
- Evaluate the effectiveness of discounts and promotions

Secondary Objectives

- Identify high-value and at-risk customers for targeted strategies
- Analyze store and regional performance to improve operations and inventory planning
- Evaluate employee performance and discount approval behavior
- Understand product performance and cross-selling opportunities

- Track revenue trends and seasonality to guide future planning

Each analysis step and dashboard page in this project is directly aligned with one or more of these objectives.

3. Dataset Overview

The project uses five relational datasets representing different business functions:

1. **Customer Data**
   Contains customer identifiers, customer type (Retail, Online, Wholesale), and demographic attributes.

2. **Transaction Data**
   Core transactional dataset containing invoice details, dates, products, quantities, net amount, and discount amount.

3. **Store Data**
   Store-level information including store ID, region, and location.

4. **Revenue (GL) Data**
   Aggregated revenue records used to validate transactional revenue and support financial trend analysis.

5. **Employee Access & Approval Data**
   Employee-level data indicating sales involvement and discount approval authority.

These datasets together form a realistic retail data model used for customer, store, employee, and revenue analysis.

4. Tools & Technologies Used

- **Python**: Data cleaning, feature engineering, RFM analysis, EDA
- **MySQL**: Business-driven queries, aggregations, ranking, and views
- **Power BI**: Interactive dashboards, KPIs, and visual storytelling
- **GitHub**: Project structure, version control, and documentation

5. Data Cleaning & Preparation

Data preparation was performed using Python and SQL to ensure accuracy and reliability. Key steps included:

- Removing duplicate transaction records
- Handling missing and inconsistent values
- Standardizing date formats for time-based analysis
- Creating derived fields such as revenue per transaction and discount percentages
- Ensuring consistency across customer, store, and employee identifiers

This step was critical to ensure that downstream analysis and dashboards reflected true business performance.

6. Python Analysis – Business Problem Solving

6.1 Customer Analysis

RFM Segmentation

Customers were segmented using **Recency, Frequency, and Monetary value** into groups such as:

- Champions
- Loyal Customers
- At-Risk Customers
- Lost Customers

Business Impact:
This segmentation helps marketing teams identify whom to reward, retain, or re-engage through targeted campaigns.

```
In [100…  max_date = transaction_data["invoice_date"].max()

In [103…  customer_last_purchase = transaction_data.groupby("customer_id")["invoice_date"].max()

In [114…  recency = (max_date - customer_last_purchase).dt.days.sort_values().reset_index()

In [115…  recency
```

Out[115…

	customer_id	invoice_date
0	C010	0

In [85]:
```
freq = transaction_data["customer_id"].value_counts().reset_index()
freq.columns = ["customer_id", "Frequency"]
```

In [87]:
```
freq
```

Out[87]:

	customer_id	Frequency
0	C002	18
1	C070	17
2	C023	15
3	C007	15
4	C043	14
...	...	...
75	C004	5
76	C013	5
77	C053	4
78	C008	4
79	C034	2

80 rows × 2 columns

In [92]:
```
monetary = transaction_data.groupby("customer_id")["net_amount"].sum().sort_values(ascending = False).reset_index
```

In [116…
```
monetary
```

Top Customers Analysis

The top customers by total net revenue were identified, along with:

- Stores contributing most to their purchases
- Products frequently bought by them

Business Impact:
Enables personalized offers and focused retention strategies for high-value customers.

In [ ]:
```
# Top Customers Analysis
# Identify the top 10 customers by total net revenue.
# Analyze which stores and products contribute most to their purchases.
# Goal: Focus marketing and personalized offers on high-value customers.
```

In [173…
```
high_paying_cus = transaction_data.groupby("customer_id")["net_amount"].sum().sort_values(ascending = False).heac
```

In [174…
```
high_paying_cus
```

Out[174…
```
Index(['C002', 'C023', 'C072', 'C070', 'C054', 'C022', 'C037', 'C050', 'C007',
       'C032'],
      dtype='object', name='customer_id')
```

In [178…
```
target_data = transaction_data[transaction_data["customer_id"].isin(high_paying_cus)]
```

In [183…
```
high_paying_cus_stores = target_data["store_id"].value_counts().reset_index()
```

In [189…
```
high_paying_cus_stores = high_paying_cus_stores.merge(store_data[["store_id", "store_name"]], on = "store_id", hc
```

In [190…
```
high_paying_cus_stores
```

Churn Risk Analysis

Customers with no transactions in the last three months were identified as churn-risk customers.

Business Impact:
Supports proactive reactivation campaigns before revenue is permanently lost.

Customer Type Comparison

Retail, Online, and Wholesale customers were compared based on:

- Average revenue per customer
- Average discount received

Business Impact:
Helps tailor pricing, discount, and engagement strategies for each customer type.

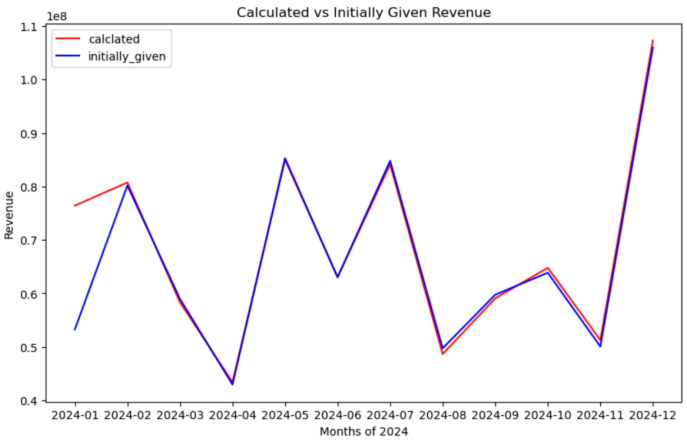
6.2 Store & Revenue Analysis

- Monthly revenue trends were analyzed to identify growth patterns and seasonality
- Store-level revenue and discount behavior were compared
- Regional revenue contribution was evaluated

Business Impact:
Helps management allocate inventory, marketing budgets, and operational focus to high-performing stores and regions.

```
In [256…
plt.figure(figsize = (10,6))
plt.plot(revenue_data["month"], org_rev_per_month, color = "red", label = "calclated")
plt.plot(revenue_data["month"], revenue_data["amount"], color = "blue", label = "initially_given")
plt.xlabel("Months of 2024")
plt.ylabel("Revenue")
plt.legend()
plt.title("Calculated vs Initially Given Revenue")

Out[256…
Text(0.5, 1.0, 'Calculated vs Initially Given Revenue')
```



6.3 Employee & Sales Performance

- Employee contribution to total revenue was analyzed
- Relationship between discounts approved and revenue generated was examined

Business Impact:
Supports performance evaluation, incentive planning, and discount governance.

Employee & Sales Performance

In [279…
```
# Employee Revenue Contribution
# Identify top-performing employees based on total net revenue generated.
# Goal: Reward high performers and optimize staff deployment.
```

In [288…
```
empl_rev = transaction_data.groupby("created_by")["net_amount"].sum().sort_values(ascending = False).reset_index(
```

In [287…
```
employee_data
```

Out[287…

	user_id	user_name	role	can_create_invoice	can_approve_discount	can_modify_master_data
0	u_andika	Andika	Sales Clerk	Y	N	N
1	u_bella	Bella	Sales Clerk	Y	N	N
2	u_citra	Citra	Senior Sales	Y	Y	N
3	u_dwi	Dwi	Store Supervisor	N	Y	N
4	u_eka	Eka	Store Manager	N	Y	Y
5	u_fajar	Fajar	Back Office	N	N	Y
6	u_gita	Gita	Online Channel	Y	Y	N
7	u_hanif	Hanif	Admin	N	N	Y

In [291…
```
empl_rev.columns = ["user_id", "revenue"]
```

In [293…
```
empl_rev = empl_rev.merge(employee_data[["user_id","user_name","role"]], on= "user_id", how = "left")
```

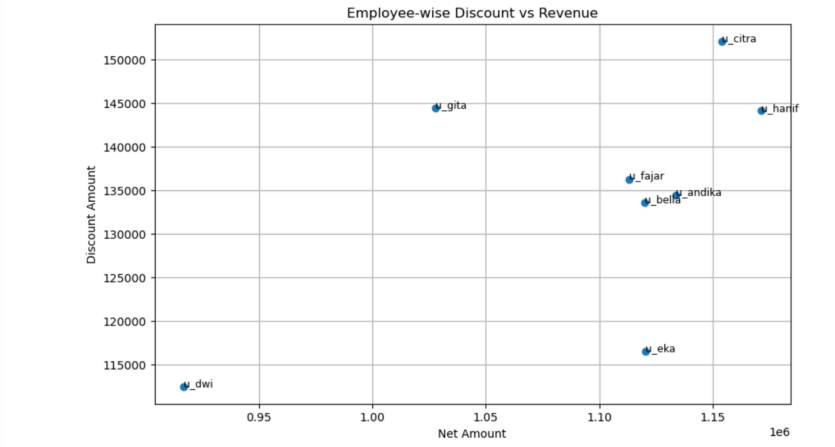
In [295…
```
empl_rev.drop(columns = ["user_name_y", "role_y"])
```

Out[295…

user_id	revenue	user_name_x	role_x

```
for i in range(len(emply_track)):
    x = emply_track["net_amount"][i]
    y = emply_track["discount_amount"][i]
    label = emply_track["created_by"][i]
    plt.text(x, y, label, fontsize=9)

plt.xlabel("Net Amount")
plt.ylabel("Discount Amount")
plt.title("Employee-wise Discount vs Revenue")
plt.grid(True)
```



6.4 Product & Transaction Insights

- Top products by quantity and revenue were identified
- Gross margin was calculated to highlight high-margin products

- Frequently purchased product combinations were analyzed

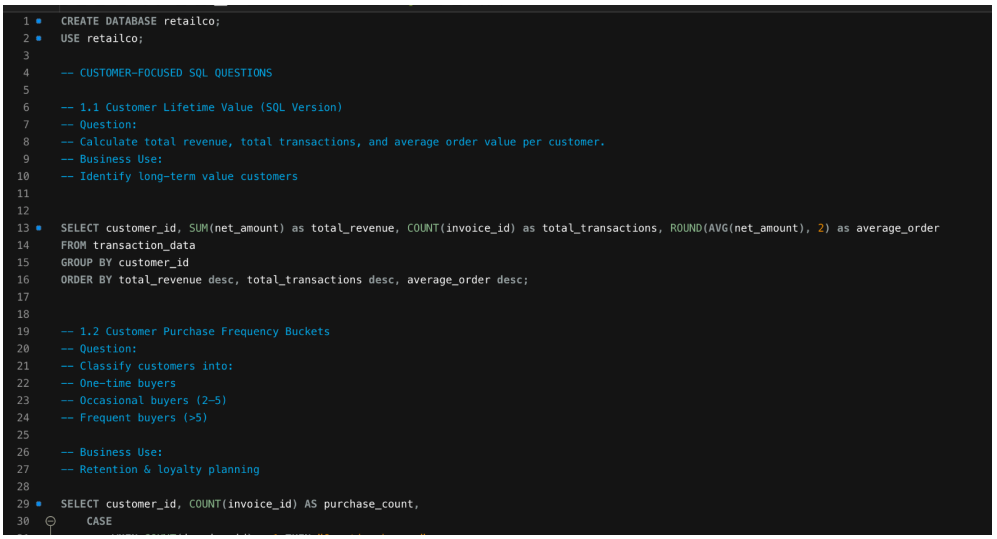
Business Impact:
Improves inventory planning, promotion design, and cross-selling strategies.

7. MySQL Analysis – Strategic & Operational Queries

Advanced SQL queries were designed to answer real business questions:

Customer Intelligence

- Customer lifetime value metrics
- Purchase frequency buckets
- Inactive customer detection
- Discount dependency analysis



```sql
CREATE DATABASE retailco;
USE retailco;

-- CUSTOMER-FOCUSED SQL QUESTIONS

-- 1.1 Customer Lifetime Value (SQL Version)
-- Question:
-- Calculate total revenue, total transactions, and average order value per customer.
-- Business Use:
-- Identify long-term value customers


SELECT customer_id, SUM(net_amount) as total_revenue, COUNT(invoice_id) as total_transactions, ROUND(AVG(net_amount), 2) as average_order
FROM transaction_data
GROUP BY customer_id
ORDER BY total_revenue desc, total_transactions desc, average_order desc;


-- 1.2 Customer Purchase Frequency Buckets
-- Question:
-- Classify customers into:
-- One-time buyers
-- Occasional buyers (2–5)
-- Frequent buyers (>5)

-- Business Use:
-- Retention & loyalty planning

SELECT customer_id, COUNT(invoice_id) AS purchase_count,
    CASE
```

Store & Regional Intelligence

- Store-level revenue vs discount efficiency

- Store ranking within regions

```
60
61  -- STORE & REGIONAL INTELLIGENCE
62
63  -- 2.1 Store Revenue vs Discount Efficiency
64  -- Question:
65  -- For each store:
66  -- Total net revenue
67  -- Average discount
68  -- Revenue per transaction
69  -- Business Use:
70  -- Find stores that sell well without heavy discounts
71
72  SELECT
73      s.store_id,
74      s.store_name,
75      SUM(t.net_amount) AS total_net_revenue,
76      ROUND(AVG(t.discount_amount), 2) AS average_discount,
77      ROUND(AVG(t.net_amount), 2) AS revenue_per_transaction
78  FROM transaction_data t
79  JOIN store_data s ON t.store_id = s.store_id
80  GROUP BY s.store_id, s.store_name;
81
82  -- 2.2 Store Ranking Within Region
83  -- Question:
84  -- Rank stores by revenue within each region.
85  -- Business Use:
86  -- Regional benchmarking
87
88  SELECT
89      s.store_id,
```

Employee Governance

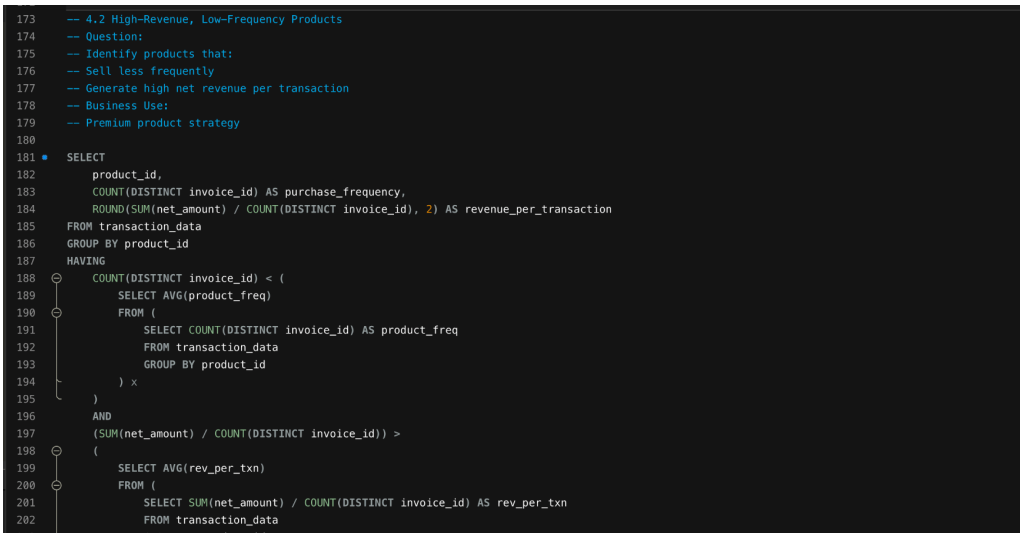
- Impact of discount approval authority
- Detection of unusually high discount behavior
- Revenue per invoice per employee

```
98
99  -- 3.1 Employee Discount Authority Impact
100 -- Question:
101 -- Compare revenue and average discount between:
102 -- Employees who can approve discounts
103 -- Employees who cannot
104
105 -- Business Use:
106 -- Policy evaluation
107
108 SELECT
109     e.user_id,
110     SUM(t.net_amount) AS total_revenue,
111     ROUND(AVG(t.discount_amount), 2) AS average_discount
112 FROM employee_data e
113 JOIN transaction_data t
114     ON e.user_id = t.created_by
115 WHERE e.can_approve_discount = 'Y'
116 GROUP BY e.user_id;
117
118 SELECT
119     e.user_id,
120     SUM(t.net_amount) AS total_revenue,
121     ROUND(AVG(t.discount_amount), 2) AS average_discount
122 FROM employee_data e
123 JOIN transaction_data t
124     ON e.user_id = t.created_by
125 WHERE e.can_approve_discount = 'N'
126 GROUP BY e.user_id;
127
```

Product & Pricing Intelligence

- Product discount elasticity

- High-revenue, low-frequency products



```sql
-- 4.2 High-Revenue, Low-Frequency Products
-- Question:
-- Identify products that:
-- Sell less frequently
-- Generate high net revenue per transaction
-- Business Use:
-- Premium product strategy

SELECT
    product_id,
    COUNT(DISTINCT invoice_id) AS purchase_frequency,
    ROUND(SUM(net_amount) / COUNT(DISTINCT invoice_id), 2) AS revenue_per_transaction
FROM transaction_data
GROUP BY product_id
HAVING
    COUNT(DISTINCT invoice_id) < (
        SELECT AVG(product_freq)
        FROM (
            SELECT COUNT(DISTINCT invoice_id) AS product_freq
            FROM transaction_data
            GROUP BY product_id
        ) x
    )
    AND
    (SUM(net_amount) / COUNT(DISTINCT invoice_id)) >
    (
        SELECT AVG(rev_per_txn)
        FROM (
            SELECT SUM(net_amount) / COUNT(DISTINCT invoice_id) AS rev_per_txn
            FROM transaction_data
```

Time-Based Analysis

- Month-over-month revenue growth
- Store-level seasonality
- Invoice date vs posting date delays

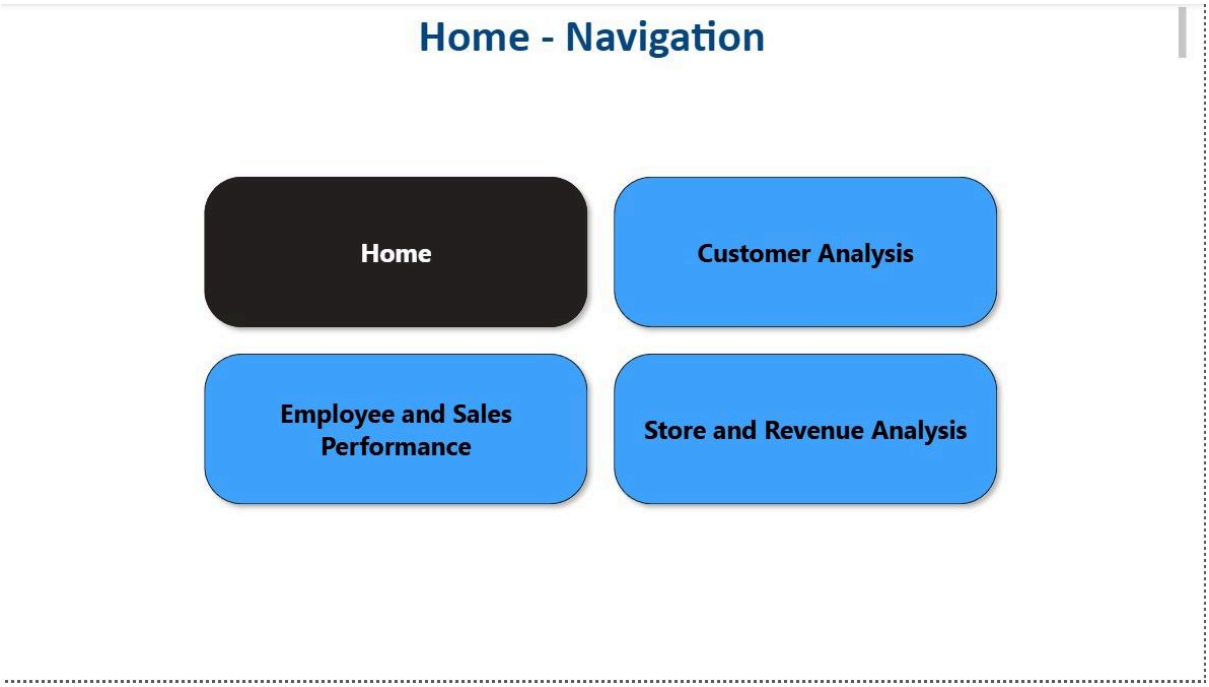
Power BI–Ready Views

SQL views were created for customer and store summaries to ensure smooth integration with Power BI dashboards.

8. Power BI Dashboards – Visual Business Storytelling

8.1 Home Page

The Home page serves as a navigation hub with buttons linking to all analysis pages. This design improves usability and reflects real-world executive dashboards.



8.2 Customer Analysis Dashboard

Key KPIs

- Total number of customers
- Total revenue
- Revenue per customer
- Average transactions per customer

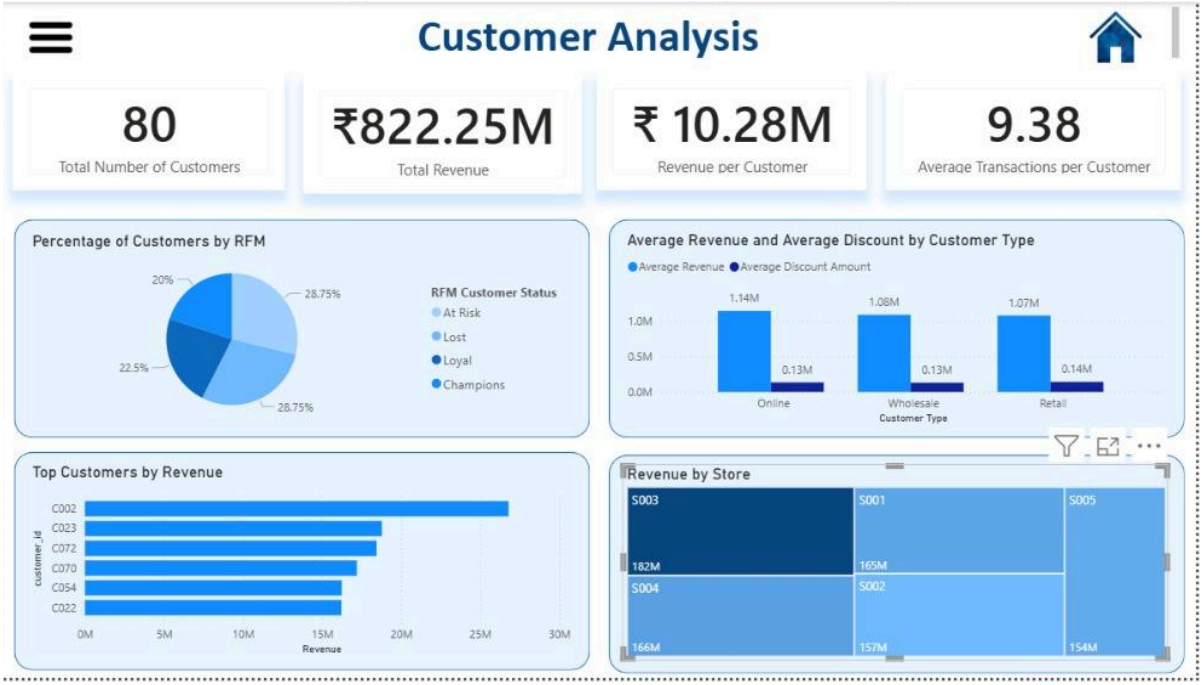
Visuals

- Pie chart showing customer distribution by RFM segment
- Bar chart comparing average revenue and average discount by customer type

Features

- Interactive slicers accessible via a collapsible menu
- Home navigation button for seamless user experience

Business Value
This dashboard helps identify high-value customers, understand customer behavior across segments, and detect discount sensitivity.



8.3 Employee & Sales Performance Dashboard

Key KPIs

- Total number of employees
- Average revenue per employee

Visuals

- Horizontal bar chart of revenue per employee
- Scatter plot showing revenue vs discount per employee
- Monthly revenue trend line

Features

- Consistent slicers and navigation controls

Business Value

Enables performance benchmarking, identifies top performers, and highlights potential

discount misuse.

8.4 Store & Revenue Analysis Dashboard

Key KPIs

- Total number of stores
- Total revenue
- Revenue per store
- Average transactions per store

Visuals

- Monthly revenue trend by store
- Overall monthly revenue trend
- Bar charts of average revenue and discount per store
- Pie chart showing revenue contribution by store

Business Value

Helps management evaluate store performance, detect underperforming locations, and optimize regional strategies.

9. Key Insights

- A small segment of customers contributes a disproportionate share of revenue
- Certain stores consistently outperform others with lower discount dependency
- Higher discounts do not always correlate with higher revenue
- Employee performance varies significantly, indicating scope for optimization
- Revenue shows clear seasonality patterns that can guide planning

10. Business Recommendations

- Invest in retention programs for high-value customers
- Re-engage inactive customers with targeted offers
- Optimize discount policies to protect margins
- Replicate best practices from top-performing stores
- Introduce data-driven employee incentives
- Plan inventory and staffing based on seasonal demand

11. Conclusion

This project demonstrates how **data analytics can convert complex retail data into actionable business insights**. By integrating Python, MySQL, and Power BI, the analysis provides a comprehensive view of customer behavior, employee performance, store efficiency, and revenue trends.

The project closely mirrors real-world analytics workflows and showcases the ability to solve business problems using data-driven approaches.

12. Future Scope

- Advanced customer segmentation using RFM scoring models
- Predictive sales forecasting
- Customer churn prediction using machine learning
- Automated reporting pipelines