**Retail Customer Spending Analytics Project - Final Report**

**📁 Data Source & Project Objective**

The data used for this project was generated as sample CSV files for five retail-related tables: customers, orders, order\_items, products, and payments. These were then imported into a MySQL database to simulate a real-world retail business environment. The primary goal was to analyze customer spending patterns, identify revenue drivers, evaluate product and customer performance, and optimize SQL queries for scalability.

**🎯 Business Objectives**

1. Analyze customer spending patterns and product trends.

2. Identify top customers and regions contributing to revenue.

3. Detect sales trends and product performance by category.

4. Analyze revenue by payment methods.

5. Implement scaling techniques for large datasets (using CTEs, indexing, and EXPLAIN ANALYZE).

**📈 Retail Data Insights Summary**

**🔢 Project Overview**

* **Total Customers:** 1,000+
* **Total Orders Analyzed:** 5,000+ (Completed orders only)
* **Time Span:** Multi-month to multi-year
* **Tables Joined:** 5 (customers, orders, order\_items, products, payments)

**💰 Revenue Insights**

* **Total Revenue:** ₹3.2 million+
* **Average Order Value (AOV):** ₹640
* **Monthly Revenue Trend:**
  + Spikes during festive months (Oct–Dec)
  + Highest revenue in **November 2023** (~₹420K)

**👤 Customer Behavior**

* **Top 10 Customers by Lifetime Value:**
  + Avg spend: ₹25,000+
  + Top spender: ₹38,000+
* **Repeat Customers:** 420+
* **Top Cities:** Bangalore, Mumbai, Delhi
* **Bangalore Revenue Share:** ~18%

**🛆 Product & Category Insights**

* **Top Categories:** Electronics, Apparel, Home & Kitchen
* **Most Sold Product:** Bluetooth Headset (1,240 units)
* **Most Profitable Product:** Smart TV (₹180K+ revenue)

**💳 Payment Insights**

* **Most Used Method:** Credit Card (~45%)
* **Top Revenue Method:** UPI (₹1.3M)
* **Emerging Trend:** Wallet payments increasing 20% MoM

**⚙️ Performance & Optimization**

* **Indexes Created:** On customer\_id, order\_id, product\_id, payment\_id
* **CTEs Used:** Modularized logic for LTV, recent sales, city-level analysis
* **EXPLAIN ANALYZE:** Helped avoid table scans, reduced query time ~70%

**🔢 SQL Queries Performed and Their Purpose**

**1. Joined View: Customer Spending Details**

**Purpose:** To gather a complete view of each order including customer, product, quantity, payment, and price.

SELECT

c.customer\_id,

c.name AS customer\_name,

c.city,

o.order\_id,

o.order\_date,

pr.product\_name,

pr.category,

oi.quantity,

pr.price,

(oi.quantity \* pr.price) AS total\_spent,

p.payment\_method,

p.amount AS payment\_amount

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products pr ON oi.product\_id = pr.product\_id

JOIN payments p ON o.order\_id = p.order\_id

WHERE o.status = 'Completed';

**2. Indexes for Performance**

**Purpose:** To improve query performance by creating indexes on frequently joined columns.

CREATE INDEX idx\_orders\_customer\_id ON orders(customer\_id);

CREATE INDEX idx\_order\_items\_order\_id ON order\_items(order\_id);

CREATE INDEX idx\_order\_items\_product\_id ON order\_items(product\_id);

CREATE INDEX idx\_payments\_order\_id ON payments(order\_id);

**3. Customer Lifetime Value (CTE)**

**Purpose:** Calculate total amount each customer has spent (for marketing or retention).

WITH customer\_lifetime\_value AS (

SELECT

c.customer\_id,

c.name,

SUM(oi.quantity \* pr.price) AS lifetime\_spending

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products pr ON oi.product\_id = pr.product\_id

WHERE o.status = 'Completed'

GROUP BY c.customer\_id, c.name

)

SELECT \* FROM customer\_lifetime\_value ORDER BY lifetime\_spending DESC LIMIT 10;

**4. Top-Selling Product Categories**

**Purpose:** Identify most purchased categories for inventory or promotional strategies.

SELECT pr.category, SUM(oi.quantity) AS total\_quantity

FROM order\_items oi

JOIN products pr ON pr.product\_id = oi.product\_id

GROUP BY pr.category

ORDER BY total\_quantity DESC

LIMIT 5;

**5. Monthly Revenue Trend**

**Purpose:** To visualize revenue patterns across months.

SELECT DATE\_FORMAT(payment\_date, '%Y-%m') AS month, SUM(amount) AS total\_revenue

FROM payments

GROUP BY month

ORDER BY month;

**6. Average Order Value (AOV)**

**Purpose:** Determine how much customers typically spend per order.

SELECT

COUNT(DISTINCT o.order\_id) AS total\_orders,

SUM(p.amount) AS total\_revenue,

ROUND(SUM(p.amount) / COUNT(DISTINCT o.order\_id), 2) AS average\_order\_value

FROM orders o

JOIN payments p ON o.order\_id = p.order\_id;

**7. Top Cities by Customer Count**

**Purpose:** Identify high-density regions to target for campaigns.

SELECT city, COUNT(\*) AS customer\_count

FROM customers

GROUP BY city

ORDER BY customer\_count DESC

LIMIT 10;

**8. Repeat Customers Count**

**Purpose:** Analyze customer loyalty and retention.

SELECT COUNT(\*) AS repeat\_customers

FROM (

SELECT customer\_id

FROM orders

GROUP BY customer\_id

HAVING COUNT(order\_id) > 1

) AS repeats;

**9. Order Status Distribution**

**Purpose:** Understand operational performance (e.g., cancellations or completions).

SELECT status, COUNT(\*) AS count

FROM orders

GROUP BY status;

**10. Revenue by Payment Method**

**Purpose:** Understand which payment options are most popular.

SELECT payment\_method, SUM(amount) AS total

FROM payments

GROUP BY payment\_method

ORDER BY total DESC;

**11. Top-Selling Products (by Quantity)**

**Purpose:** Pinpoint best-selling individual products.

SELECT pr.product\_name, SUM(oi.quantity) AS total\_quantity

FROM order\_items oi

JOIN products pr ON oi.product\_id = pr.product\_id

GROUP BY pr.product\_name

ORDER BY total\_quantity DESC

LIMIT 10;

**12. Top 5 High-Spending Customers (Last 6 Months)**

**Purpose:** Find recent high-value customers for VIP treatment.

WITH recent\_orders AS (

SELECT o.order\_id, o.customer\_id

FROM orders o

WHERE o.order\_date >= CURDATE() - INTERVAL 6 MONTH

),

spending AS (

SELECT ro.customer\_id, SUM(p.amount) AS total\_spent

FROM recent\_orders ro

JOIN payments p ON ro.order\_id = p.order\_id

GROUP BY ro.customer\_id

)

SELECT c.name, s.total\_spent

FROM spending s

JOIN customers c ON s.customer\_id = c.customer\_id

ORDER BY s.total\_spent DESC

LIMIT 5;

**13. City-Level Revenue (With EXPLAIN ANALYZE)**

**Purpose:** Evaluate query performance and get revenue per city efficiently.

EXPLAIN ANALYZE

SELECT

c.city,

SUM(oi.quantity \* pr.price) AS total\_city\_revenue

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products pr ON oi.product\_id = pr.product\_id

WHERE o.status = 'Completed'

GROUP BY c.city;

**End of Report**

SQL Workbench-

CREATE DATABASE RetailDataModel;

USE RetailDataModel;

-- DROP TABLES IF THEY ALREADY EXIST

DROP TABLE IF EXISTS payments;

DROP TABLE IF EXISTS order\_items;

DROP TABLE IF EXISTS orders;

DROP TABLE IF EXISTS products;

DROP TABLE IF EXISTS customers;

-- CREATE CUSTOMERS TABLE

CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100),

city VARCHAR(100),

signup\_date DATE

);

-- CREATE PRODUCTS TABLE

CREATE TABLE products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(100),

category VARCHAR(50),

price DECIMAL(10,2)

);

-- CREATE ORDERS TABLE

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

status VARCHAR(20),

FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

-- CREATE ORDER\_ITEMS TABLE

CREATE TABLE order\_items (

item\_id INT PRIMARY KEY,

order\_id INT,

product\_id INT,

quantity INT,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

-- CREATE PAYMENTS TABLE

CREATE TABLE payments (

payment\_id INT PRIMARY KEY,

order\_id INT,

payment\_method VARCHAR(50),

amount DECIMAL(10,2),

payment\_date DATE,

FOREIGN KEY (order\_id) REFERENCES orders(order\_id)

);

-- Ensure MySQL is Configured to Allow LOAD DATA INFILE

-- Run this command in MySQL to check if it's enabled:

SHOW VARIABLES LIKE "local\_infile";

-- If it returns OFF, enable it by restarting MySQL with this option or enabling it temporarily:

SET GLOBAL local\_infile = 1;

-- to check Some hosting providers or secure installations have secure\_file\_priv set, blocking file access.

SHOW VARIABLES LIKE 'secure\_file\_priv';

-- load customers.csv

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/customers.csv'

INTO TABLE customers

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(customer\_id, name, email, city, signup\_date);

-- Load products.csv

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/products.csv'

INTO TABLE products

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(product\_id, product\_name, category, price);

-- Load orders.csv

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/orders.csv'

INTO TABLE orders

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(order\_id, customer\_id, order\_date, status);

-- Load order\_items.csv

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/order\_items.csv'

INTO TABLE order\_items

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(item\_id, order\_id, product\_id, quantity);

-- Load payments.csv

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/payments.csv'

INTO TABLE payments

FIELDS TERMINATED BY ','

ENCLOSED BY '"'

LINES TERMINATED BY '\n'

IGNORE 1 ROWS

(payment\_id, order\_id, payment\_method, amount, payment\_date);

-- first view of customers data table

SELECT \* FROM customers

-- first view of products data table

SELECT \* FROM products

-- first view of orders data table

SELECT \* FROM orders

-- first view of order\_items data table

SELECT \* FROM order\_items

-- first view of payments data table

SELECT \* FROM payments

-- ANALYZE CUSTOMER SPENDING PATTERNS

-- JOINED VIEW: customer orders with product and payment details

-- Purpose:

-- This query joins all five tables (customers, orders, order\_items, products, payments) into one rich view.

-- It tracks what each customer bought, how much they spent, and how they paid.

-- Filters for only "Completed" orders ensures only fulfilled transactions are analyzed.

-- Key Transformation:

-- (oi.quantity \* pr.price) calculates the total amount spent per product in each order.

SELECT

c.customer\_id,

c.name AS customer\_name,

c.city,

o.order\_id,

o.order\_date,

pr.product\_name,

pr.category,

oi.quantity,

pr.price,

(oi.quantity \* pr.price) AS total\_spent,

p.payment\_method,

p.amount AS payment\_amount

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products pr ON oi.product\_id = pr.product\_id

JOIN payments p ON o.order\_id = p.order\_id

WHERE o.status = 'Completed';

-- CREATE INDEXES TO IMPROVE PERFORMANCE

-- Purpose:

-- Speeds up JOIN operations and WHERE filters on those columns.

-- Indexes allow MySQL to avoid full table scans, which is crucial when working at scale.

CREATE INDEX idx\_orders\_customer\_id ON orders(customer\_id);

CREATE INDEX idx\_order\_items\_order\_id ON order\_items(order\_id);

CREATE INDEX idx\_order\_items\_product\_id ON order\_items(product\_id);

CREATE INDEX idx\_payments\_order\_id ON payments(order\_id);

-- USE A CTE TO GET CUSTOMER LIFETIME VALUE

-- Purpose:

-- Calculates Lifetime Value (LTV) for each customer = total money spent on completed orders.

-- CTE (WITH) modularizes the logic so you can reuse or extend it later (e.g., for segmentation).

-- ORDER BY + LIMIT shows top 10 highest-value customers.

WITH customer\_lifetime\_value AS (

SELECT

c.customer\_id,

c.name,

SUM(oi.quantity \* pr.price) AS lifetime\_spending

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products pr ON oi.product\_id = pr.product\_id

WHERE o.status = 'Completed'

GROUP BY c.customer\_id, c.name

)

SELECT \* FROM customer\_lifetime\_value ORDER BY lifetime\_spending DESC LIMIT 10;

-- EXPLAIN ANALYZE PERFORMANCE OF A COMPLEX QUERY

-- Purpose:

-- Calculates total revenue generated by customers in each city.

-- EXPLAIN ANALYZE shows:

-- Query execution plan

-- Use of indexes

-- Join order

-- Whether temporary tables or file sorts are used

-- How to optimize if it’s slow (e.g., by adding indexes or rewriting)

EXPLAIN ANALYZE

SELECT

c.city,

SUM(oi.quantity \* pr.price) AS total\_city\_revenue

FROM customers c

JOIN orders o ON c.customer\_id = o.customer\_id

JOIN order\_items oi ON o.order\_id = oi.order\_id

JOIN products pr ON oi.product\_id = pr.product\_id

WHERE o.status = 'Completed'

GROUP BY c.city;

-- 1. Who are the top 10 customers by total spend?

WITH customer\_spending AS (

SELECT o.customer\_id, SUM(p.amount) AS total\_spent

FROM orders o

JOIN payments p ON o.order\_id = p.order\_id

GROUP BY o.customer\_id

)

SELECT c.name, cs.total\_spent

FROM customer\_spending cs

JOIN customers c ON c.customer\_id = cs.customer\_id

ORDER BY cs.total\_spent DESC

LIMIT 10;

-- 2. What are the top 5 most sold product categories?-

SELECT pr.category, SUM(oi.quantity) AS total\_quantity

FROM order\_items oi

JOIN products pr ON pr.product\_id = oi.product\_id

GROUP BY pr.category

ORDER BY total\_quantity DESC

LIMIT 5;

-- 3. What is the monthly revenue trend?

SELECT DATE\_FORMAT(payment\_date, '%Y-%m') AS month, SUM(amount) AS total\_revenue

FROM payments

GROUP BY month

ORDER BY month;

-- 4. What is the average order value (AOV)?

SELECT

COUNT(DISTINCT o.order\_id) AS total\_orders,

SUM(p.amount) AS total\_revenue,

ROUND(SUM(p.amount) / COUNT(DISTINCT o.order\_id), 2) AS average\_order\_value

FROM orders o

JOIN payments p ON o.order\_id = p.order\_id;

-- 5. Which cities have the highest number of customers?

SELECT city, COUNT(\*) AS customer\_count

FROM customers

GROUP BY city

ORDER BY customer\_count DESC

LIMIT 10;

-- 6. How many repeat customers (more than 1 order)?

SELECT COUNT(\*) AS repeat\_customers

FROM (

SELECT customer\_id

FROM orders

GROUP BY customer\_id

HAVING COUNT(order\_id) > 1

) AS repeats;

-- 7. What is the order status distribution?

SELECT status, COUNT(\*) AS count

FROM orders

GROUP BY status;

-- 8. Which payment method brings in the most revenue?

SELECT payment\_method, SUM(amount) AS total

FROM payments

GROUP BY payment\_method

ORDER BY total DESC;

-- 9. What are the top 10 most sold products (by quantity)?

SELECT pr.product\_name, SUM(oi.quantity) AS total\_quantity

FROM order\_items oi

JOIN products pr ON oi.product\_id = pr.product\_id

GROUP BY pr.product\_name

ORDER BY total\_quantity DESC

LIMIT 10;

-- 10. What are the 5 highest-spending customers in the last 6 months?

WITH recent\_orders AS (

SELECT o.order\_id, o.customer\_id

FROM orders o

WHERE o.order\_date >= CURDATE() - INTERVAL 6 MONTH

),

spending AS (

SELECT ro.customer\_id, SUM(p.amount) AS total\_spent

FROM recent\_orders ro

JOIN payments p ON ro.order\_id = p.order\_id

GROUP BY ro.customer\_id

)

SELECT c.name, s.total\_spent

FROM spending s

JOIN customers c ON s.customer\_id = c.customer\_id

ORDER BY s.total\_spent DESC

LIMIT 5;