Sales and Analytics System – Salase Industries

Internship Project Report

Prepared by: NANCY (00115)

Role: Data Analyst Intern

Date: June 25, 2025

Executive Summary

This project presents a MySQL-based data analytics system for Salase Industries, an FMCG firm. It includes the design and creation of database tables, realistic sample data insertion, and 30 business-critical SQL queries to extract insights from customer behavior, product performance, and agent efficiency. The goal is to provide a strong backend system that supports decision-making for operations, marketing, and sales strategies.

# ER Diagram

The Entity Relationship (ER) diagram includes the following tables and their relationships:  
- Customers → Sales  
- Products → Sales  
- Sales → Agent\_Sales → SalesAgents  
(Visual ER diagram can be inserted here if needed.)

Database Schema

The following tables were created: Customers, Products, Sales, SalesAgents, and Agent\_Sales.

SQL Table Creation Scripts:

CREATE DATABASE salesindustry;  
USE salesindustry;  
  
CREATE TABLE Customers (  
 CustomerID INT PRIMARY KEY,  
 Name VARCHAR(100),  
 Email VARCHAR(100),  
 Phone VARCHAR(20),  
 City VARCHAR(50),  
 State VARCHAR(50),  
 JoinDate DATE  
);  
  
CREATE TABLE Products (  
 ProductID INT PRIMARY KEY,  
 ProductName VARCHAR(100),  
 Category VARCHAR(50),  
 UnitPrice DECIMAL(10,2),  
 StockQuantity INT  
);  
  
CREATE TABLE Sales (  
 SaleID INT PRIMARY KEY,  
 CustomerID INT,  
 ProductID INT,  
 SaleDate DATE,  
 QuantitySold INT,  
 TotalAmount DECIMAL(10,2),  
 FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),  
 FOREIGN KEY (ProductID) REFERENCES Products(ProductID)  
);  
  
CREATE TABLE SalesAgents (  
 AgentID INT PRIMARY KEY,  
 AgentName VARCHAR(100),  
 Region VARCHAR(50),  
 Phone VARCHAR(20),  
 JoiningDate DATE  
);  
  
CREATE TABLE Agent\_Sales (  
 SaleID INT,  
 AgentID INT,  
 FOREIGN KEY (SaleID) REFERENCES Sales(SaleID),  
 FOREIGN KEY (AgentID) REFERENCES SalesAgents(AgentID)  
);

# Sample Data Inserted

Sample data (10 rows per table) has been inserted for testing and querying. Below is a snippet:

-- Insert into Customers  
INSERT INTO Customers (CustomerID, Name, Email, Phone, City, State, JoinDate) VALUES  
(1, 'Amit Kumar', 'amitk@example.com', '9998877661', 'Delhi', 'Delhi', '2024-01-10'),  
(2, 'Sneha Sharma', 'snehas@example.com', '9898989898', 'Mumbai', 'Maharashtra', '2023-12-05'),  
(3, 'Raj Patel', 'rajp@example.com', '9876543210', 'Ahmedabad', 'Gujarat', '2024-05-12'),  
(4, 'Priya Singh', 'priyas@example.com', '9123456789', 'Bangalore', 'Karnataka', '2023-11-22'),  
(5, 'Karan Verma', 'karanv@example.com', '9345678912', 'Hyderabad', 'Telangana', '2024-03-15'),  
(6, 'Divya Jain', 'divyaj@example.com', '9001122334', 'Kolkata', 'West Bengal', '2024-04-01'),  
(7, 'Manoj Mehta', 'manojm@example.com', '9887766554', 'Chennai', 'Tamil Nadu', '2025-01-05'),  
(8, 'Anjali Kapoor', 'anjalik@example.com', '9988776655', 'Pune', 'Maharashtra', '2024-02-19'),  
(9, 'Ravi Reddy', 'ravir@example.com', '9876123456', 'Secunderabad', 'Telangana', '2023-10-08'),  
(10, 'Pooja Nair', 'poojan@example.com', '9823456781', 'Kochi', 'Kerala', '2024-06-10');  
  
-- Insert into Products  
INSERT INTO Products (ProductID, ProductName, Category, UnitPrice, StockQuantity) VALUES  
(1, 'Toothpaste', 'Personal Care', 45.00, 300),  
(2, 'Shampoo', 'Personal Care', 120.00, 150),  
(3, 'Rice (5kg)', 'Food Grains', 350.00, 200),  
(4, 'Washing Powder', 'Household', 80.00, 500),  
(5, 'Cooking Oil (1L)', 'Groceries', 130.00, 250),  
(6, 'Detergent Bar', 'Household', 30.00, 450),  
(7, 'Biscuits (Pack of 10)', 'Snacks', 90.00, 300),  
(8, 'Hand Sanitizer', 'Health', 60.00, 180),  
(9, 'Milk (1L)', 'Dairy', 50.00, 400),  
(10, 'Tea Powder (500g)', 'Beverages', 180.00, 160);  
  
-- Insert into Sales  
INSERT INTO Sales (SaleID, CustomerID, ProductID, SaleDate, QuantitySold, TotalAmount) VALUES  
(1, 1, 1, '2024-06-01', 2, 90.00),  
(2, 2, 3, '2024-06-02', 1, 350.00),  
(3, 3, 5, '2024-06-03', 3, 390.00),  
(4, 4, 2, '2024-06-04', 1, 120.00),  
(5, 5, 6, '2024-06-05', 4, 120.00),  
(6, 6, 4, '2024-06-06', 2, 160.00),  
(7, 7, 9, '2024-06-07', 5, 250.00),  
(8, 8, 10, '2024-06-08', 1, 180.00),  
(9, 9, 8, '2024-06-09', 2, 120.00),  
(10, 10, 7, '2024-06-10', 3, 270.00);  
  
-- Insert into SalesAgents  
INSERT INTO SalesAgents (AgentID, AgentName, Region, Phone, JoiningDate) VALUES  
(1, 'Vikram Rana', 'North', '9990001111', '2023-01-15'),  
(2, 'Sonal Joshi', 'West', '9887766555', '2022-12-01'),  
(3, 'Arun Pillai', 'South', '9776655443', '2023-03-20'),  
(4, 'Reema Das', 'East', '9665544332', '2023-05-11'),  
(5, 'Faizan Khan', 'Central', '9554433221', '2024-02-14'),  
(6, 'Neha Mehra', 'North-East', '9443322110', '2024-06-01'),  
(7, 'Tarun Ahuja', 'West', '9332211009', '2023-07-22'),  
(8, 'Lavanya Rao', 'South', '9221100998', '2023-08-30'),  
(9, 'Ritesh Gupta', 'North', '9110099887', '2024-01-05'),  
(10, 'Meena Yadav', 'East', '9009988776', '2024-03-18');  
  
-- Insert into Agent\_Sales  
INSERT INTO Agent\_Sales (SaleID, AgentID) VALUES  
(1, 1), (2, 2), (3, 3), (4, 1), (5, 5),  
(6, 4), (7, 6), (8, 3), (9, 2), (10, 7);

# SQL Queries for Business Insights

The following 30 SQL queries are categorized into Aggregation, Filtering, Grouping, Sorting, and Limiting results.

##Aggregation Functions  
#Total sales revenue generated across all products  
  
SELECT SUM(TotalAmount) AS TotalRevenue  
FROM Sales;  
  
SELECT ProductID, SUM(QuantitySold) AS TotalUnitsSold FROM Sales GROUP BY ProductID ORDER BY TotalUnitsSold DESC LIMIT 1;  
  
SELECT AVG(QuantitySold) AS AvgQuantityPerSale FROM Sales;  
  
SELECT COUNT(DISTINCT CustomerID) AS TotalCustomers FROM Sales;  
  
SELECT CustomerID, SUM(TotalAmount) AS TotalSpent FROM Sales GROUP BY CustomerID;  
  
SELECT MAX(TotalAmount) AS MaxSaleAmount FROM Sales;  
  
 SELECT ProductID, AVG(TotalAmount) AS AvgAmountPerProduct FROM Sales GROUP BY ProductID;  
   
 SELECT MIN(QuantitySold) AS MinQuantitySold FROM Sales;  
   
 SELECT P.Category, SUM(S.TotalAmount) AS Revenue FROM Sales S JOIN Products P ON S.ProductID = P.ProductID GROUP BY P.Category;  
   
 SELECT A.AgentID, COUNT(S.SaleID) AS TotalProductsSold FROM Agent\_Sales AG JOIN Sales S ON AG.SaleID = S.SaleID JOIN SalesAgents A ON AG.AgentID = A.AgentID GROUP BY A.AgentID;  
   
 SELECT \* FROM Sales WHERE QuantitySold > 10;  
   
 SELECT \* FROM Customers WHERE State = 'Delhi';  
   
 SELECT \* FROM Sales WHERE SaleDate > '2025-01-01';  
   
 SELECT \* FROM Products WHERE StockQuantity < 200;  
   
 SELECT \* FROM Customers WHERE YEAR(JoinDate) = 2024;  
   
 SELECT CustomerID, SUM(TotalAmount) AS TotalSpent FROM Sales GROUP BY CustomerID HAVING SUM(TotalAmount) > 50000;  
   
 SELECT ProductID, SUM(QuantitySold) AS TotalUnits FROM Sales GROUP BY ProductID HAVING SUM(QuantitySold) > 200;  
   
 SELECT AG.AgentID, SUM(S.TotalAmount) AS TotalHandled FROM Agent\_Sales AG JOIN Sales S ON AG.SaleID = S.SaleID GROUP BY AG.AgentID HAVING SUM(S.TotalAmount) > 100000;  
   
 SELECT P.Category, SUM(S.TotalAmount) AS Revenue FROM Sales S JOIN Products P ON S.ProductID = P.ProductID GROUP BY P.Category HAVING SUM(S.TotalAmount) > 200000;  
   
 SELECT City, COUNT(DISTINCT CustomerID) AS UniqueCustomers FROM Customers GROUP BY City HAVING COUNT(DISTINCT CustomerID) > 5;  
   
 SELECT \* FROM Products ORDER BY UnitPrice DESC;  
   
 SELECT CustomerID, SUM(TotalAmount) AS TotalSpent FROM Sales GROUP BY CustomerID ORDER BY TotalSpent DESC LIMIT 10;  
 SELECT P.ProductID, SUM(S.TotalAmount) AS Revenue FROM Sales S JOIN Products P ON S.ProductID = P.ProductID GROUP BY P.ProductID ORDER BY Revenue DESC LIMIT 5;  
 SELECT \* FROM Sales ORDER BY SaleDate DESC;  
   
SELECT AG.AgentID, SUM(S.TotalAmount) AS TotalSales FROM Agent\_Sales AG JOIN Sales S ON AG.SaleID = S.SaleID GROUP BY AG.AgentID ORDER BY TotalSales DESC;  
SELECT P.Category, SUM(S.TotalAmount) AS TotalRevenue FROM Sales S JOIN Products P ON S.ProductID = P.ProductID GROUP BY P.Category ORDER BY TotalRevenue DESC;  
SELECT \* FROM Sales ORDER BY SaleDate DESC LIMIT 10;  
SELECT CustomerID, COUNT(SaleID) AS Purchases FROM Sales GROUP BY CustomerID ORDER BY Purchases DESC LIMIT 3;  
  
SELECT AgentID, COUNT(SaleID) AS NumSales FROM Agent\_Sales GROUP BY AgentID ORDER BY NumSales DESC LIMIT 5;  
  
SELECT \* FROM Customers ORDER BY JoinDate DESC LIMIT 5;

**Insights and Observations**

Based on the executed queries, the following observations were made:  
- High-performing products and categories were identified.  
- Customer spending patterns helped reveal loyal and high-value clients.  
- Sales agents' efficiency and contribution were measured.  
- Stock levels of popular items can be forecasted.  
- Geographical trends (like state-wise customers) emerged clearly.

# Conclusion

The Sales and Analytics System provides essential data infrastructure and reporting capabilities for Salase Industries. By combining structured relational tables with rich SQL queries, it enables efficient data retrieval, insight generation, and business strategy alignment. This project demonstrates real-world application of SQL in solving core analytics problems for FMCG businesses.