Sales Analytics Database & SQL Reporting System

Objective

The objective of this project is to design and implement a SQL-based sales analytics system that stores customer, product, order, and regional data. Using SQL queries, the project generates business insights such as total revenue, monthly trends, best-selling products, and region-wise performance.

Introduction

A company must analyze sales performance to understand customer behavior, product demand, and revenue trends. A structured SQL database helps store sales data efficiently and enables analytical reporting using queries. This project demonstrates how SQL can be used to build a clean sales database and generate meaningful insights.

Database Design

ER Diagram

Write the relationships:

- One region has many customers
- One customer has many orders
- One order has many order items
- One product appears in many order items

Tables Used

List of tables:

- Regions
- Customers
- Products
- Orders
- Order_Items

SQL Schema

```
mysql> create database sales
    -> ;
Query OK, 1 row affected (0.09 sec)
```

```
mysql> use sales
Database changed
```

```
mysql> CREATE TABLE customers (
    ->     customer_id INT PRIMARY KEY,
    ->     name VARCHAR(100),
    ->     email VARCHAR(100),
    ->     region_id INT,
    ->     FOREIGN KEY (region_id) REFERENCES regions(region_id)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> CREATE TABLE products (
    ->     product_id INT PRIMARY KEY,
    ->     name VARCHAR(100),
    ->     category VARCHAR(50),
    ->     price DECIMAL(10,2)
    -> );
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> CREATE TABLE orders (
    ->     order_id INT PRIMARY KEY,
    ->     customer_id INT,
    ->     order_date DATE,
    ->     FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> 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)
    -> );
Query OK, 0 rows affected (0.06 sec)
```

Sample Data

```
mysql> INSERT INTO regions VALUES
    -> (1, 'North'), (2, 'South'), (3, 'East'), (4, 'West');
Query OK, 4 rows affected (0.02 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO customers VALUES
    -> (101, 'Arjun', 'arjun@gmail.com', 1),
    -> (102, 'Meera', 'meera@gmail.com', 2),
    -> (103, 'Simran', 'simran@gmail.com', 3);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO products VALUES
    -> (1, 'Laptop', 'Electronics', 50000),
    -> (2, 'Phone', 'Electronics', 30000),
    -> (3, 'Shoes', 'Fashion', 2000);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO orders VALUES
    -> (1001, 101, '2025-01-10'),
    -> (1002, 102, '2025-01-11'),
    -> (1003, 103, '2025-01-12');
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO order_items VALUES
    -> (1, 1001, 1, 1),
    -> (2, 1001, 3, 2),
    -> (3, 1002, 2, 1),
    -> (4, 1003, 1, 1);
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

SQL Queries for Analytics

- **Total revenue**

```
mysql> SELECT SUM(p.price * oi.quantity) AS total_revenue
    -> FROM order_items oi
    -> JOIN products p ON oi.product_id = p.product_id;
+---------------+
| total_revenue |
+---------------+
|     134000.00 |
+---------------+
1 row in set (0.02 sec)
```

- **Monthly revenue**

```
mysql> SELECT DATE_FORMAT(order_date, "%Y-%m") AS month,
    ->        SUM(price * quantity) AS revenue
    -> FROM orders o
    -> JOIN order_items oi ON o.order_id = oi.order_id
    -> JOIN products p ON oi.product_id = p.product_id
    -> GROUP BY month;
+---------+-----------+
| month   | revenue   |
+---------+-----------+
| 2025-01 | 134000.00 |
+---------+-----------+
1 row in set (0.01 sec)
```

- **Top 5 Best-Selling Products**

```
mysql> SELECT p.name, SUM(oi.quantity) AS total_sold
    -> FROM order_items oi
    -> JOIN products p ON oi.product_id = p.product_id
    -> GROUP BY p.name
    -> ORDER BY total_sold DESC
    -> LIMIT 5;
+--------+------------+
| name   | total_sold |
+--------+------------+
| Laptop |          2 |
| Shoes  |          2 |
| Phone  |          1 |
+--------+------------+
3 rows in set (0.00 sec)
```

- **Region-wise Revenue**

```
mysql> SELECT r.region_name, SUM(p.price * oi.quantity) AS revenue
    -> FROM orders o
    -> JOIN customers c ON o.customer_id = c.customer_id
    -> JOIN regions r ON c.region_id = r.region_id
    -> JOIN order_items oi ON o.order_id = oi.order_id
    -> JOIN products p ON oi.product_id = p.product_id
    -> GROUP BY r.region_name;
+-------------+----------+
| region_name | revenue  |
+-------------+----------+
| North       | 54000.00 |
| South       | 30000.00 |
| East        | 50000.00 |
+-------------+----------+
3 rows in set (0.00 sec)
```

- **Customer Purchase Summary**

```
mysql> SELECT c.name, COUNT(o.order_id) AS total_orders,
    ->        SUM(p.price * oi.quantity) AS total_spent
    -> 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 p ON oi.product_id = p.product_id
    -> GROUP BY c.name;
+--------+--------------+-------------+
| name   | total_orders | total_spent |
+--------+--------------+-------------+
| Arjun  |            2 |    54000.00 |
| Meera  |            1 |    30000.00 |
| Simran |            1 |    50000.00 |
+--------+--------------+-------------+
3 rows in set (0.00 sec)
```

Conclusion

This project demonstrates how SQL can be used to design a functional sales analytics system. The database structure efficiently stores sales data and supports analytical queries that reveal valuable business insights. Using SQL, companies can monitor performance, identify trends, and make data-driven decisions.