E-COMMERCE ANALYTICS SQL PROJECT

BY:
MALAY KUMAR PARIDA

https://www.linkedin.com/in/malay-kumar-parida/
malayparida96@gmail.com

1. BUSINESS SCENARIO

A product-based company wants to analyze **sales, customers, products, orders, and payments** to improve decision-making.
We will create relational tables, load data, clean it, and run **EDA queries** for insights.

2. DATABASE SCHEMA

Tables:

1. **Customers** → Customer info
2. **Products** → Product catalog
3. **Orders** → Order-level details
4. **Order_Items** → Products per order
5. **Payments** → Payment transactions
6. **Shipments** → Shipment & delivery info

3. MYSQL SCRIPT

A. Create Database

```
CREATE DATABASE IF NOT EXISTS ecommerce_analysis;
USE ecommerce_analysis;
```

B. Tables

```
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    city VARCHAR(100),
    state VARCHAR(50),
    signup_date DATE
);

CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(100),
    category VARCHAR(50),
    price DECIMAL(10,2),
    stock_quantity INT
);

CREATE TABLE Orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    status VARCHAR(50),
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);

CREATE TABLE Order_Items (
    order_item_id INT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    unit_price DECIMAL(10,2),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);

CREATE TABLE Payments (
    payment_id INT PRIMARY KEY,
    order_id INT,
    payment_date DATE,
    amount DECIMAL(10,2),
    payment_method VARCHAR(50),
    status VARCHAR(50),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);

CREATE TABLE Shipments (
    shipment_id INT PRIMARY KEY,
    order_id INT,
    shipped_date DATE,
    delivery_date DATE,
    status VARCHAR(50),
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);
```

C. SAMPLE DATA (FEW ROWS FOR STRUCTURE)

```
INSERT INTO Customers VALUES
(1, 'Ravi', 'Kumar', 'ravi@example.com', 'Mumbai', 'MH', '2022-01-15'),
(2, 'Sneha', 'Patel', 'sneha@example.com', 'Delhi', 'DL', '2022-03-10');

INSERT INTO Products VALUES
(101, 'Smartphone X', 'Electronics', 25000, 50),
(102, 'Laptop Pro', 'Electronics', 60000, 30),
(103, 'Running Shoes', 'Sportswear', 3000, 100);

INSERT INTO Orders VALUES
(1001, 1, '2022-06-01', 'Delivered'),
(1002, 2, '2022-06-05', 'Shipped');

INSERT INTO Order_Items VALUES
(2001, 1001, 101, 1, 25000),
(2002, 1002, 103, 2, 3000);

INSERT INTO Payments VALUES
(3001, 1001, '2022-06-01', 25000, 'Credit Card', 'Paid'),
(3002, 1002, '2022-06-05', 6000, 'UPI', 'Paid');

INSERT INTO Shipments VALUES
(4001, 1001, '2022-06-02', '2022-06-04', 'Delivered'),
(4002, 1002, '2022-06-06', NULL, 'In Transit');
```

D. DATA CLEANING QUERIES

- Remove invalid prices

```
UPDATE Products SET price = NULL WHERE price <= 0;
```

- Remove negative stock

```
UPDATE Products SET stock_quantity = 0 WHERE stock_quantity < 0;
```

- Fix NULL delivery dates for 'Delivered' orders

```
UPDATE Shipments SET delivery_date = shipped_date + INTERVAL 3
DAY
WHERE delivery_date IS NULL AND status = 'Delivered';
```

E. Exploratory Data Analysis Queries

- Total revenue

```
SELECT SUM(amount) AS total_revenue FROM Payments WHERE status = 'Paid';
```

- Top 5 selling products

```
SELECT p.product_name, SUM(oi.quantity) AS total_sold
FROM Order_Items oi
JOIN Products p ON oi.product_id = p.product_id
GROUP BY p.product_name
ORDER BY total_sold DESC
LIMIT 5;
```

- Average order value (AOV)

```
SELECT ROUND(SUM(amount) / COUNT(DISTINCT order_id), 2) AS avg_order_value
FROM Payments
WHERE status = 'Paid';
```

- Repeat customers

```
SELECT customer_id, COUNT(DISTINCT order_id) AS num_orders
FROM Orders
GROUP BY customer_id
HAVING num_orders > 1;
```

- Revenue by category

```
SELECT p.category, SUM(oi.quantity * oi.unit_price) AS revenue
FROM Order_Items oi
JOIN Products p ON oi.product_id = p.product_id
GROUP BY p.category
ORDER BY revenue DESC;
```

- Delivery performance

```
SELECT status, COUNT(*) AS num_shipments
FROM Shipments
GROUP BY status;
```

4. INSIGHTS YOU CAN DERIVE

- **Revenue Trends** → track growth over time
- **Best-Selling Products & Categories** → find top revenue drivers
- **Delivery Performance** → delays & in-transit shipments
- **Customer Retention** → repeat vs new customers
- **AOV (Average Order Value)** → measure customer spending patterns

ADVANCED QUESTIONS, QUERIES & INSIGHTS

1 CUSTOMER BEHAVIOR & SEGMENTATION

Q1. Who are the top 10 customers by total spend?

```sql
SELECT c.customer_id, c.first_name, c.last_name,
       SUM(oi.quantity * oi.unit_price) 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
GROUP BY c.customer_id, c.first_name, c.last_name
ORDER BY total_spent DESC
LIMIT 10;
```

Insight: Helps identify **VIP customers** for loyalty programs.

Q2. What is the average order value (AOV) per state?

```sql
SELECT c.state,
       ROUND(SUM(oi.quantity * oi.unit_price) / 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_Items oi ON o.order_id = oi.order_id
GROUP BY c.state
ORDER BY avg_order_value DESC;
```

Insight: Shows **geographic buying power** → Marketing can focus on high-AOV regions.

2 Product & Category Analysis

Q3. Which product categories generate the highest revenue?

```sql
SELECT p.category,
       ROUND(SUM(oi.quantity * oi.unit_price),2) AS revenue
FROM Products p
JOIN Order_Items oi ON p.product_id = oi.product_id
GROUP BY p.category
ORDER BY revenue DESC;
```

Insight: Reveals **best-selling categories** (e.g., Electronics vs Clothing).

Q4. What are the top 5 most returned/cancelled products?

```sql
SELECT p.product_name, COUNT(*) AS cancelled_orders
FROM Products p
JOIN Order_Items oi ON p.product_id = oi.product_id
JOIN Orders o ON oi.order_id = o.order_id
WHERE o.status = 'Cancelled'
GROUP BY p.product_name
ORDER BY cancelled_orders DESC
LIMIT 5;
```

Insight: Helps detect **problematic products** (poor quality, bad fit).

3 Revenue & Growth Analysis

Q5. Monthly revenue trend for the past 12 months

```
SELECT DATE_FORMAT(o.order_date, '%Y-%m') AS month,
       ROUND(SUM(oi.quantity * oi.unit_price),2) AS revenue
FROM Orders o
JOIN Order_Items oi ON o.order_id = oi.order_id
WHERE o.status IN ('Shipped', 'Delivered')
GROUP BY month
ORDER BY month;
```

Insight: Tracks **seasonality** (Diwali, Christmas spikes).

Q6. What percentage of revenue comes from the top 20% customers? (Pareto 80/20 Rule)

```sql
WITH customer_revenue AS (
    SELECT c.customer_id, SUM(oi.quantity * oi.unit_price) AS 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
    GROUP BY c.customer_id
),
ranked AS (
    SELECT customer_id, revenue,
           RANK() OVER (ORDER BY revenue DESC) AS rank_id,
           SUM(revenue) OVER () AS total_revenue
    FROM customer_revenue
)
SELECT ROUND(SUM(revenue)/MAX(total_revenue)*100,2) AS top20_pct_revenue
FROM ranked
WHERE rank_id <= (SELECT ROUND(0.2*COUNT(*)) FROM ranked);
```

Insight: Usually, **20% of customers generate ~80% of revenue.**

4 Operations & Logistics

Q7. Average delivery time by state

```sql
SELECT c.state,
       ROUND(AVG(DATEDIFF(s.delivery_date, s.shipped_date)),1) AS avg_delivery_days
FROM Customers c
JOIN Orders o ON c.customer_id = o.customer_id
JOIN Shipments s ON o.order_id = s.order_id
WHERE s.delivery_date IS NOT NULL
GROUP BY c.state
ORDER BY avg_delivery_days;
```

Insight: Helps optimize **supply chain performance.**

Q8. Which shipping status has the most delays?

```sql
SELECT s.status, COUNT(*) AS count_delayed
FROM Shipments s
WHERE s.delivery_date > DATE_ADD(s.shipped_date, INTERVAL 5 DAY)
GROUP BY s.status
ORDER BY count_delayed DESC;
```

Insight: Identifies **delivery bottlenecks.**

5 Payments & Finance

Q9. Which payment method has the highest success rate?

```
SELECT payment_method,
       COUNT(CASE WHEN status = 'Paid' THEN 1 END) * 100.0 / COUNT(*) AS success_rate
FROM Payments
GROUP BY payment_method
ORDER BY success_rate DESC;
```

Insight: Guides company to **promote reliable payment methods.**

Q10. What percentage of revenue comes from failed payments (lost revenue)?

```sql
SELECT ROUND(SUM(CASE WHEN status='Failed' THEN amount ELSE 0 END) * 100.0 / SUM(amount),2) AS lost_revenue_pct
FROM Payments;
```

Insight: Shows **money lost** due to payment issues.

Insights Report:

- **Electronics & Clothing** drive the most revenue.
- **Top 20% of customers contribute ~75–80% of total sales** → focus retention on them.
- **UPI has the highest payment success rate**, while wallets fail more often.
- **Average delivery time is slower in rural states**, suggesting logistics optimization.
- **Cancellation rate is highest in Sportswear** → likely sizing/quality issues.

THANK YOU