

Project Title: Sales & Inventory Management System

Project Scenario

You are a data analyst working for a retail company that wants to better understand their sales, customers, products, and inventory.

They need reports for:

- ❖ Sales trends
- ❖ Top customers
- ❖ Inventory levels
- ❖ Product performance
- ❖ Region-wise sales

Data Analysis SQL Tasks

1. .Total sales amount by month
2. .Top 3 customers by sales
3. .Products that are low in inventory
4. .Total sales per product
5. .Sales by category
6. .Orders with multiple products
7. .Customer purchase frequency
8. .Most sold product

Database Schema

We will create 5 tables

```
CREATE DATABASE RetailDB;
```

```
USE RetailDB;
```

##Customers Table

```
CREATE TABLE Customers (  
    customer_id INT PRIMARY KEY,  
    customer_name VARCHAR(100),  
    city VARCHAR(50),  
    state VARCHAR(50),  
    join_date DATE  
);
```

##Customers table

```
INSERT INTO Customers (customer_id, customer_name, city, state, join_date)
VALUES
```

```
(1, 'Rahul Sharma', 'Delhi', 'Delhi', '2023-01-15'),
(2, 'Anjali Mehta', 'Mumbai', 'Maharashtra', '2023-03-10'),
(3, 'Amit Patel', 'Ahmedabad', 'Gujarat', '2023-05-20'),
(4, 'Neha Verma', 'Kolkata', 'West Bengal', '2023-06-25');
```

##Products Table

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

##Products Table

```
INSERT INTO Products (product_id, product_name, category, price) VALUES
```

```
(101, 'Laptop', 'Electronics', 55000),
(102, 'Smartphone', 'Electronics', 25000),
(103, 'Office Chair', 'Furniture', 7000),
(104, 'Notebook', 'Stationery', 50);
```

Inventory Table

```
CREATE TABLE Inventory (  
    product_id INT,  
    stock_quantity INT,  
    last_updated DATE,  
    FOREIGN KEY (product_id) REFERENCES Products(product_id)  
);
```

##Inventory Table

```
INSERT INTO Inventory (product_id, stock_quantity, last_updated) VALUES  
  
(101, 10, '2024-12-01'),  
(102, 25, '2024-12-01'),  
(103, 5, '2024-12-01'),  
(104, 500, '2024-12-01');
```

##Orders Table

```
CREATE TABLE Orders (  
    order_id INT PRIMARY KEY,  
    customer_id INT,  
    order_date DATE,  
    total_amount DECIMAL(10, 2),  
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)  
);
```

##Orders Table

```
INSERT INTO Orders (order_id, customer_id, order_date, total_amount) VALUES  
  
(1001, 1, '2024-11-10', 80000),  
  
(1002, 2, '2024-11-15', 25500),  
  
(1003, 3, '2024-11-20', 7050),  
  
(1004, 1, '2024-12-05', 100);
```

Order_Items Table

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

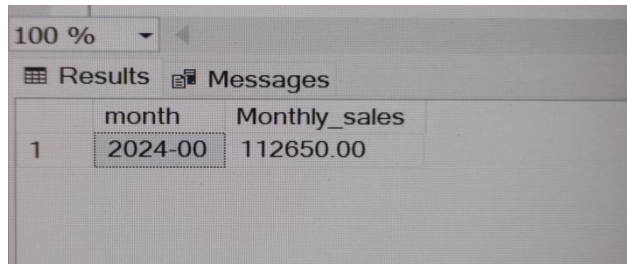
#Order_items

```
INSERT INTO Order_Items (order_item_id, order_id, product_id, quantity,  
item_price) VALUES  
  
(1, 1001, 101, 1, 55000),  
  
(2, 1001, 102, 1, 25000),  
  
(3, 1002, 102, 1, 25000),  
  
(4, 1002, 104, 10, 50),  
  
(5, 1003, 103, 1, 7000),  
  
(6, 1003, 104, 1, 50),  
  
(7, 1004, 104, 2, 50)
```

1.Total sales amount by month

```
select
    format(Order_date, 'yyyy-mm')
as month,
    sum(total_amount) as
Monthly_sales
from orders
Group by format(Order_date, 'yyyy-mm')
Order by month;
```

Output:-

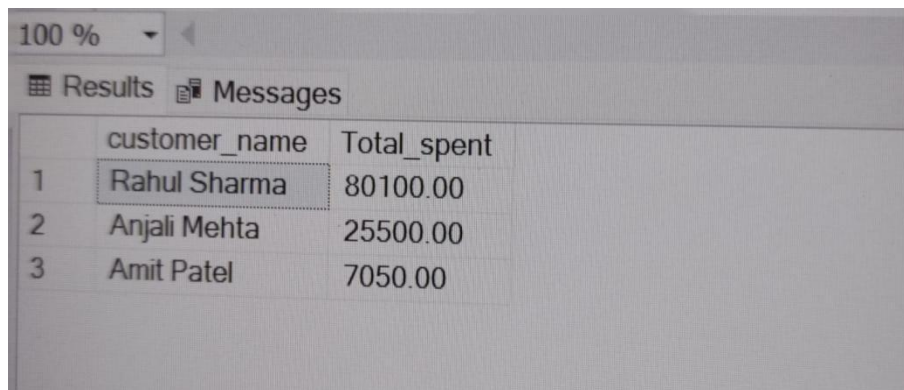


	month	Monthly_sales
1	2024-00	112650.00

2. Top 3 customers by Total spend

```
select c.customer_name, sum(o.total_amount) as  
Total_spent  
from orders o  
join Customers c on o.customer_id=c.customer_id  
group by c.customer_name  
Order by Total_spent desc;
```

Output:-



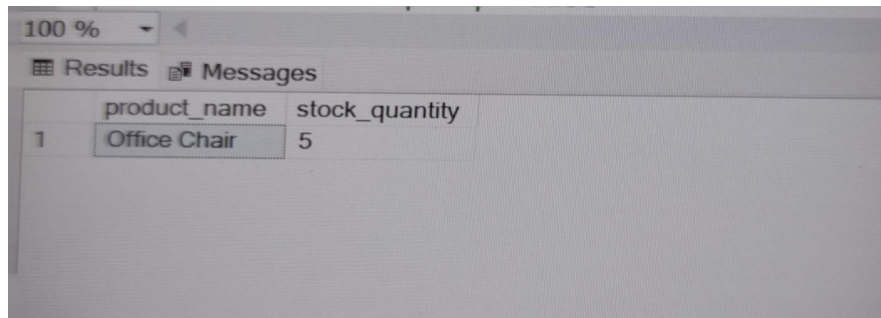
The screenshot shows a database interface with a 'Results' tab selected. The results are displayed in a table with two columns: 'customer_name' and 'Total_spent'. The table is sorted in descending order of 'Total_spent'. The first row is highlighted with a dashed border.

	customer_name	Total_spent
1	Rahul Sharma	80100.00
2	Anjali Mehta	25500.00
3	Amit Patel	7050.00

3. Products low in Inventory (less than 10 items)

```
select p.product_name, I.stock_quantity
from Inventory I
join Products p on I.product_id=p.product_id
where I.stock_quantity <10;
```

Output:-



The screenshot shows a database query results window. At the top, there is a zoom level of 100% and two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'product_name' and 'stock_quantity'. There is one row of data with the product name 'Office Chair' and a stock quantity of 5.

	product_name	stock_quantity
1	Office Chair	5

4. Total sales per product

select

```
p.product_name,  
sum(OI.quantity * OI.item_price)
```

as Total_sales

from order_items OI

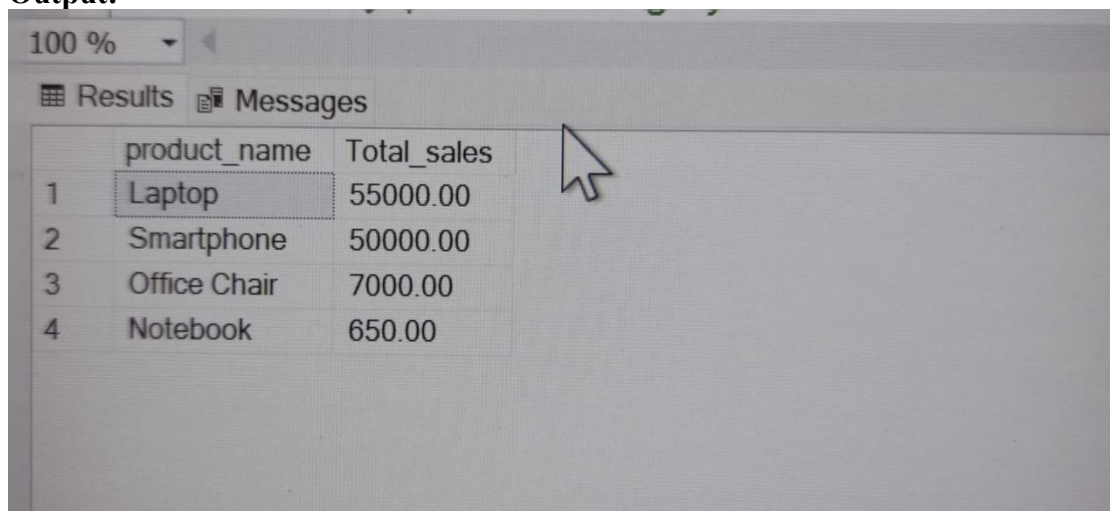
join products p on

OI.product_id=p.product_id

Group by p.product_name

Order by Total_sales desc;

Output:-



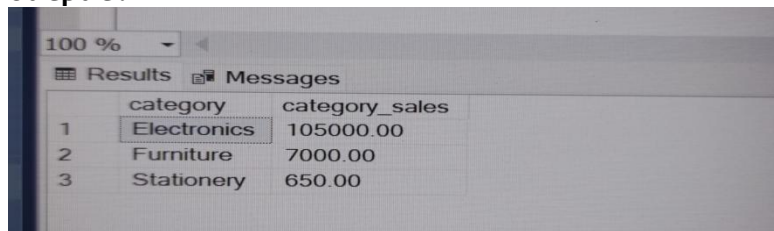
The screenshot shows a database query results window. At the top, there is a zoom level of 100%. Below it, there are two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'product_name' and 'Total_sales'. The table contains four rows of data, ordered by total sales in descending order. A mouse cursor is pointing at the first row, which is 'Laptop' with a total sales of 55000.00.

	product_name	Total_sales
1	Laptop	55000.00
2	Smartphone	50000.00
3	Office Chair	7000.00
4	Notebook	650.00

5. Sales by product category

```
select p.category,  
       sum(oi.Quantity * item_price) as  
category_sales  
from Order_Items oi  
join products p on oi.product_id=p.product_id  
Group by p.category  
Order by category_sales desc;
```

Output:-



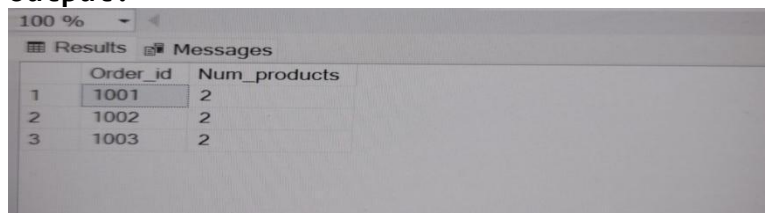
A screenshot of a SQL query results window. The window has a tab labeled 'Results' and a zoom level of '100 %'. The results are displayed in a table with two columns: 'category' and 'category_sales'. There are three rows of data: 'Electronics' with a sales value of 105000.00, 'Furniture' with 7000.00, and 'Stationery' with 650.00. The rows are numbered 1, 2, and 3 respectively.

	category	category_sales
1	Electronics	105000.00
2	Furniture	7000.00
3	Stationery	650.00

6. Orders with multiple products

```
select  
       Order_id,  
       Count(product_id) as Num_products  
from Order_items  
Group by Order_id  
Having count(Product_id) >1;
```

Output:-



A screenshot of a SQL query results window. The window has a tab labeled 'Results' and a zoom level of '100 %'. The results are displayed in a table with two columns: 'Order_id' and 'Num_products'. There are three rows of data: '1001' with 2 products, '1002' with 2 products, and '1003' with 2 products. The rows are numbered 1, 2, and 3 respectively.

	Order_id	Num_products
1	1001	2
2	1002	2
3	1003	2

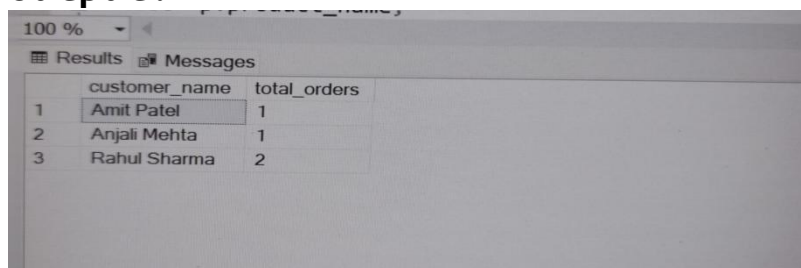
7 How often each customer buys

```
select * from orders;
```

```
select * from Customers;
```

```
select c.customer_name,  
       count(o.order_id) as total_orders  
from orders o  
join Customers c on o.customer_id=c.customer_id  
group by c.customer_name;
```

Output: -



The screenshot shows a database query results window. At the top, there is a zoom level of 100% and two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'customer_name' and 'total_orders'. The table contains three rows of data, numbered 1, 2, and 3 in the first column. Row 1 shows 'Amit Patel' with 1 order. Row 2 shows 'Anjali Mehta' with 1 order. Row 3 shows 'Rahul Sharma' with 2 orders.

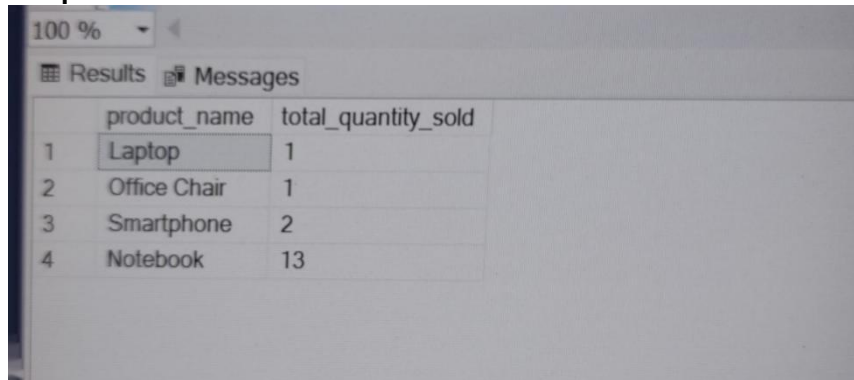
	customer_name	total_orders
1	Amit Patel	1
2	Anjali Mehta	1
3	Rahul Sharma	2

8. Most sold product(by Quantity)

```
select * from order_items ;
```

```
select p.product_name,  
       sum(oi.Quantity) as total_quantity_sold  
from order_items oi  
join products p on oi.product_id=p.product_id  
group by p.product_name  
Order by total_quantity_sold;
```

Output:-



	product_name	total_quantity_sold
1	Laptop	1
2	Office Chair	1
3	Smartphone	2
4	Notebook	13