SQL Data Analysis

Here are different table's name with it's result :

```
select * from Products
```

ProductID	ProductName	Category	Price	Stock
1	Laptop	Electronics	800.00	50
2	Smartphone	Electronics	500.00	100
3	Shoes	Footwear	50.00	200
4	Blender	Home Appliances	100.00	80
5	Watch	Accessories	150.00	120

```
select * from Customers
```

CustomerID	CustomerName	Email	City	State	Country
1	Alice Smith	alice@example.com	New York	NY	USA
2	Bob Johnson	bob@example.com	Los Angeles	CA	USA
3	Charlie Brown	charlie@example.com	Chicago	IL	USA
4	David Wilson	david@example.com	Houston	TX	USA
5	Eve Davis	eve@example.com	Phoenix	AZ	USA

```
select * from Orders
```

OrderID	CustomerID	OrderDate	Totalamount
1	1	2023-01-15	950.00
2	2	2023-02-20	600.00
3	3	2023-03-10	300.00
4	4	2023-04-05	450.00
5	5	2023-05-12	150.00

```
select * from OrderDetails
```

OrderdetailsID	OrderID	ProductID	Quantity	Discount
1	1	1	1	0.00
2	1	3	2	10.00
3	2	2	1	5.00
4	3	4	1	0.00
5	4	5	1	15.00

```
select * from Regions
```

RegionID	RegionName	Country
1	North	USA
2	South	USA
3	East	USA
4	West	USA

Now, Here are the Problem for which I have analyze the data

```
-- Question 1 --> Identify the top 10 products by total sales revenue.

select p.productName, sum(p.Price*od.Quantity) as net_revenue from Products as p inner
join Orderdetails as od
on p.ProductID = od.ProductID group by p.productName order by net_revenue desc
```

	productName	net_revenue
1	Laptop	800.00
2	Smartphone	500.00
3	Watch	150.00
4	Shoes	100.00
5	Blender	100.00

```
--Question 2 --> Find customers who haven't placed an order in the last 6 months.

SELECT c.CustomerName, o.OrderDate FROM
Customers AS c INNER JOIN Orders AS o
ON
c.CustomerID = o.CustomerID WHERE MONTH(o.OrderDate) <= 6;

--or

SELECT c.CustomerName, o.OrderDate FROM
Customers AS c INNER JOIN Orders AS o
ON
c.CustomerID = o.CustomerID WHERE datepart(MM, o.OrderDate) <= 6;
```

	CustomerName	OrderDate
1	Alice Smith	2023-01-15
2	Bob Johnson	2023-02-20
3	Charlie Brown	2023-03-10
4	David Wilson	2023-04-05
5	Eve Davis	2023-05-12

```
--Question 3 --> Calculate the total revenue for each product category

select p.category, sum(od.quantity*p.price)
as total_Revenue from products as p
inner join
orderdetails as od
on p.ProductID = od.ProductID
group by category order by total_Revenue desc;
```

	category	total_Revenue
1	Electronics	1300.00
2	Accessories	150.00
3	Footwear	100.00
4	Home Appliances	100.00

--Qestion 4 --> Compute the average value of an order across all customers.

```
select c.customerName, avg(o.TotalAmount) as avg_order_value
from customers as c
inner join
orders as o
on c.CustomerID = o.CustomerID
group by c.CustomerName
```

	customerName	avg_order_value
1	Alice Smith	950.000000
2	Bob Johnson	600.000000
3	Charlie Brown	300.000000
4	David Wilson	450.000000
5	Eve Davis	150.000000

--Question 5 --> Find all orders where a discount was applied and
--calculate the total revenue after discounts.

```
with cte as (
SELECT
    od.OrderID,
          p.Price,
        od.Quantity,
        od.Discount,
    CASE
        WHEN od.Discount > 0 THEN od.Discount
        ELSE 0
    END AS Discount_Applied
FROM
    OrderDetails AS od
INNER JOIN
    Products AS p
ON
    od.ProductID = p.ProductID
)
select orderID, Discount_Applied,
       sum(quantity*price - (quantity*price*discount)/100) as revenue_After_discount
       from cte
GROUP BY
    OrderID,
Discount_Applied
```

	orderID	Discount_Applied	revenue_After_discount
1	1	0.00	800.00000000
2	3	0.00	100.00000000
3	2	5.00	475.00000000
4	1	10.00	90.00000000
5	4	15.00	127.50000000

-- Question 6 --> Determine which regions contributed the most to overall sales

```
select r.regionName, sum(o.totalamount) as region_sales
from orders as o inner join customers as c
on c.CustomerID = o.CustomerID
inner join regions as r on r.Country = c.Country
group by r.regionName order by region_sales desc;
```

	regionName	region_sales
1	East	2450.00
2	North	2450.00
3	South	2450.00
4	West	2450.00

--Question 7 --> Calculate the profit margin for each product, considering the discount applied.

```
-- net_profit_margin = net_profit/revenue
SELECT
    p.ProductName,
    --SUM(od.Quantity * p.Price) AS Total_Sales,
    --SUM(od.Quantity * p.Price * (1 - od.Discount / 100)) AS Revenue_After_Discount,
    --SUM(od.Quantity * p.Price * (od.Discount / 100)) AS Total_Discount,
    (SUM(od.Quantity * p.Price * (1 - od.Discount / 100)) / SUM(od.Quantity *
p.Price)) * 100 AS Profit_Margin_Percentage
FROM
    Products AS p
INNER JOIN
    OrderDetails AS od
ON
    p.ProductID = od.ProductID
GROUP BY
    p.ProductName
ORDER BY
    Profit_Margin_Percentage DESC;
```

	ProductName	Profit_Margin_Percentage
1	Blender	100.000000
2	Laptop	100.000000
3	Smartphone	95.000000
4	Shoes	90.000000
5	Watch	85.000000

--Question_8 -- > Categorize customers based on their total spending
--(e.g., High-Spending, Medium-Spending, Low-Spending).

```
select c.customerName, sum(p.price*od.quantity) as total_spending,
case
        when sum(p.price*od.quantity) > 800 then 'high_spending'
        when sum(p.price*od.quantity) > 400 and sum(p.price*od.quantity) <= 800 then
'medium_spending'
        else 'low_spending'
end as spending_category
from products as p
```

```
inner join orderdetails as od on p.productID = od.ProductID
inner join orders as o on o.orderID = od.OrderID
inner join customers as c on c.customerID = o.CustomerID group by
c.customerName order by total_spending desc
```

	customerName	total_spending	spending_category
1	Alice Smith	900.00	high_spending
2	Bob Johnson	500.00	medium_spending
3	David Wilson	150.00	low_spending
4	Charlie Brown	100.00	low_spending

--Question 9 --> Analyze the sales trends on a quarterly basis for each product category.

```
select  datepart(year, o.orderdate) as year,
datepart(quarter, o.orderdate) as quarters, p.category,
sum(p.price*od.quantity) as total_sales
from orders as o
inner join orderdetails as od on o.orderID = od.orderID
inner join products as p on p.productID = od.productID
group by category, datepart(year, o.orderdate), datepart(quarter, o.orderdate)
order by quarters;
```

	year	quarters	category	total_sales
1	2023	1	Electronics	1300.00
2	2023	1	Footwear	100.00
3	2023	1	Home Appliances	100.00
4	2023	2	Accessories	150.00

--Question 10 --> Identify products frequently purchased together by analyzing OrderDetails.

```
select od1.productId as P_A,
od2.productID as P_B,
count(*) as frequency
from OrderDetails as od1
inner join OrderDetails as od2
on od1.OrderID = od2.OrderID
and od1.ProductID <> od2.ProductID
group by od1.productId,
od2.productID order by frequency
```

	P_A	P_B	frequency
1	3	1	1
2	1	3	1

```sql
--Question 11 --> Identify customers who have decreased their spending by
--more than 50% compared to the previous year.

with cte as (
select c.customerID, c.customername, year(o.orderdate) as order_year,
sum(o.totalamount) as total_spending
from customers as c inner join orders as o
on c.customerID = o.customerID
group by c.customerID, c.customername, year(o.orderdate)
)
select x.customername, x.order_year as current_year, x.total_spending as
current_spending,
y.order_year as previous_year, y.total_spending as previous_spending
from cte as x join cte as y
on x.customerID = y.customerID
and x.order_year = y.order_year + 1
where x.total_spending < (y.total_spending*0.5)
order by x.customername, x.order_year
```

	customername	current_year	current_spending	previous_year	previous_spending

```sql
--Question 12 --> Predict the next month's sales using historical data (requires
aggregating monthly sales).

with monthly_sale as(
select YEAR(orderdate) as order_year,
month(orderdate) as order_month,
sum(totalamount) as total_sales
from orders
group by YEAR(orderdate),
month(orderdate)
)
, predicted_sales as (
select order_year, order_month, total_sales,
avg(total_sales) over(order by order_year, order_month rows between 5 preceding and
current row)
as predicted_next_month
from monthly_sale
)
select order_year, order_month + 1 as predicted_month,
predicted_next_month as predicted_sales
from predicted_sales order by order_year desc, order_month desc
```

	order_year	predicted_month	predicted_sales
1	2023	6	490.000000
2	2023	5	575.000000
3	2023	4	616.666666
4	2023	3	775.000000
5	2023	2	950.000000