Beginner level

1. Retrieve Customer Details

Question:
Write an SQL query to retrieve the `CustomerID`, `AccountNumber`, and `CustomerType` for all customers who have a `TerritoryID` of 5.

```
SELECT CustomerID, AccountNumber, CustomerType

FROM customer

WHERE TerritoryID = 5;
```

---

2. Count Products in Inventory

Question:
Write a query to count the total number of distinct `ProductID` entries in the `productinventory` table.

```
SELECT COUNT(DISTINCT ProductID) AS TotalDistinctProducts

FROM productinventory;
```

---

3. Calculate Total Sales Amount

Question:
Write an SQL query to calculate the total sales amount (`LineTotal`) from the `salesorderdetail` table.

```
SELECT SUM(LineTotal) AS TotalSalesAmount

FROM salesorderdetail;
```

---

4. Find Expired Credit Cards

Question:
Write an SQL query to list all `CreditCardID` entries from the `creditcard` table where the `ExpYear` is less than the current year.

```
SELECT CreditCardID

FROM creditcard

WHERE ExpYear < YEAR(CURRENT_DATE);
```

---

5. Join Customers and Orders

Question:
"How would you write an SQL query to calculate the total sales (SubTotal) for each customer in the `salesorderheader` table, displaying the `CustomerID` alongside their `TotalSubTotal`?"

```
SELECT CustomerID, SUM(SubTotal) AS TotalSubTotal

FROM salesorderheader

GROUP BY CustomerID;
```

---

6. Filter High-Rated Products

Question:
Write an SQL query to fetch `ProductID`, `Rating`, and `ReviewerName` from the `productreview` table where the `Rating` is 4 or higher.

```
SELECT ProductID, Rating, ReviewerName

FROM productreview
```

WHERE Rating >= 4;

---

7. Analyze Product Categories

Question:
Write an SQL query to list the `Name` and `ProductCategoryID` from the `productcategory` table, ordered alphabetically by `Name`.

SELECT Name, ProductCategoryID

FROM productcategory

ORDER BY Name ASC;

---

8. Identify Orders Pending Shipment

Question:
Write an SQL query to find all `SalesOrderID` entries in the `salesorderheader` table where the `ShipDate` is NULL.

SELECT SalesOrderID

FROM salesorderheader

WHERE ShipDate IS NULL;

---

9. Find Frequent Customers

Question:
Write an SQL query to count the number of orders placed by each `CustomerID` in the

`salesorderheader` table. Display only those customers who have placed more than 10 orders.

SELECT CustomerID, COUNT(SalesOrderID) AS TotalOrders

FROM salesorderheader

GROUP BY CustomerID

HAVING COUNT(SalesOrderID) > 10;

---

10. Average Discount Applied

Question:
Write an SQL query to calculate the average `UnitPriceDiscount` from the `salesorderdetail` table for products that have a `SpecialOfferID` of 1.

SELECT AVG(UnitPriceDiscount) AS AverageDiscount

FROM salesorderdetail

WHERE SpecialOfferID = 1;

---

11. Retrieve customer details who placed orders in the last 6 months.

Use `salesorderheader` to filter orders by `OrderDate`.

SELECT DISTINCT CustomerID
FROM salesorderheader
WHERE OrderDate >= DATEADD(MONTH, -6, GETDATE());

---

12. Identify the top 5 products with the highest total sales revenue.
Aggregate `UnitPrice * OrderQty` from `salesorderdetail`.

```
SELECT TOP 5 ProductID, SUM(UnitPrice * OrderQty) AS TotalRevenue
FROM salesorderdetail
GROUP BY ProductID
ORDER BY TotalRevenue DESC;
```

---

13. Find customers who have purchased products from more than 3 different categories.
Use `productsubcateg` and group purchases by `CustomerID`.

```
SELECT CustomerID
FROM salesorderdetail sd
JOIN product p ON sd.ProductID = p.ProductID
JOIN productsubcategory ps ON p.ProductSubcategoryID = ps.ProductSubcategoryID
GROUP BY CustomerID
HAVING COUNT(DISTINCT ps.ProductSubcategoryID) > 3;
```

---

14. List customers who have used more than one payment method.
Join `salesorderheader` and `creditcard` to identify customers with multiple `CreditCardID` entries.

```
SELECT CustomerID

FROM salesorderheader
```

```
GROUP BY CustomerID

HAVING COUNT(DISTINCT CreditCardID) > 1;
```

---

15. Retrieve the names of products that were not sold in the current year.
Use `product` and exclude `ProductID` present in `salesorderdetail`.

```
SELECT p.Name

FROM product p

WHERE p.ProductID NOT IN
```

---

16. Calculate the average order value for each customer.
Use `salesorderheader` and group by `CustomerID`.

```
SELECT CustomerID, AVG(TotalDue) AS AverageOrderValue
FROM salesorderheader
GROUP BY CustomerID;
```

---

17. Find the most commonly used shipping method for online orders.
Hint: Query `ShipMethodID` from `salesorderheader` where `OnlineOrderFlag = 1`.

```
SELECT ShipMethodID, COUNT(*) AS UsageCount
FROM salesorderheader
WHERE OnlineOrderFlag = 1
GROUP BY ShipMethodID
ORDER BY UsageCount DESC
LIMIT 1;
```

---

18. Identify customers who placed an order but canceled it later.
Use `salesorderheader` and look for specific `Status` codes indicating cancellations.

```
SELECT CustomerID, SalesOrderID, Status
FROM salesorderheader
WHERE Status = 'Canceled'; -- Replace 'Canceled' with the exact code for canceled status.
```

---

19. List employees who processed orders with a total value above $10,000.

Join `salesorderheader` and `employee` and filter by aggregated `TotalDue`.

```
SELECT e.EmployeeID, e.FirstName, e.LastName, soh.SalesOrderID, SUM(soh.TotalDue) AS
TotalOrderValue
FROM salesorderheader soh
JOIN employee e ON soh.SalesPersonID = e.EmployeeID
GROUP BY e.EmployeeID, e.FirstName, e.LastName, soh.SalesOrderID
HAVING SUM(soh.TotalDue) > 10000;
```

---

20. Find the product with the highest reorder point and identify its suppliers.
Hint: Use `product`, `productvendor`, and `VendorID` to determine the relationship.

```
SELECT p.ProductID, p.Name AS ProductName, p.ReorderPoint, v.VendorID, v.Name AS
VendorName
FROM product p
JOIN productvendor pv ON p.ProductID = pv.ProductID
JOIN vendor v ON pv.VendorID = v.VendorID
WHERE p.ReorderPoint = (
   SELECT MAX(ReorderPoint) FROM product
);
```

---

21. Identify Sales Trends Over Time

```
SELECT

   YEAR(OrderDate) AS Year,

   MONTH(OrderDate) AS Month,

   SUM(LineTotal) AS TotalSales

FROM salesorderheader soh

JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID

WHERE YEAR(OrderDate) = 2023

GROUP BY YEAR(OrderDate), MONTH(OrderDate)

ORDER BY Year, Month;
```

---

22. Product Sales Analysis

```
SELECT
    p.ProductID,
    p.Name AS ProductName,
    SUM(sod.UnitPrice * sod.OrderQty) AS TotalSales
FROM product p
JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
GROUP BY p.ProductID, p.Name
ORDER BY TotalSales DESC
LIMIT 5;
```

---

23. Identify Customer Purchase Behavior

```
SELECT
    c.CustomerID,
    COUNT(soh.SalesOrderID) AS PurchaseCount
FROM customer c
JOIN salesorderheader soh ON c.CustomerID = soh.CustomerID
WHERE YEAR(soh.OrderDate) = YEAR(GETDATE()) - 1
GROUP BY c.CustomerID
HAVING PurchaseCount > 10;
```

---

24. Average Order Value by Territory

```
SELECT
    TerritoryID,
    AVG(SubTotal) AS AverageOrderValue
FROM salesorderheader
GROUP BY TerritoryID;
```

---

25. High-Spending Customers

```
SELECT
    soh.CustomerID,
    SUM(sod.LineTotal) AS TotalPurchases
FROM salesorderheader soh
JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID
GROUP BY soh.CustomerID
HAVING TotalPurchases > 10000;
```

---

26. Discount Analysis

```
SELECT
    ProductID,
    SUM(UnitPriceDiscount * OrderQty) AS TotalDiscount
FROM salesorderdetail
GROUP BY ProductID;
```

---

27. Pending Orders

```
SELECT

   SalesOrderID,

   Status,

   OrderDate

FROM salesorderheader

WHERE Status = 'Pending';
```

---

28. Product Category Sales

```
SELECT

   pc.Name AS CategoryName,

   SUM(sod.UnitPrice * sod.OrderQty) AS TotalSales

FROM productcategory pc

JOIN productsubcategory psc ON pc.ProductCategoryID = psc.ProductCategoryID

JOIN product p ON p.ProductSubcategoryID = psc.ProductSubcategoryID

JOIN salesorderdetail sod ON p.ProductID = sod.ProductID

GROUP BY pc.Name;
```

---

29. Top Customers by Sales Order Count

```
SELECT

   c.CustomerID,
    COUNT(soh.SalesOrderID) AS OrderCount

FROM customer c

JOIN salesorderheader soh ON c.CustomerID = soh.CustomerID

GROUP BY c.CustomerID

ORDER BY OrderCount DESC

LIMIT 5;
```

---

30. Sales Performance by Salesperson

```
SELECT

    SalesPersonID,

    SUM(SubTotal) AS TotalSales

FROM salesorderheader

GROUP BY SalesPersonID;
```

---

31. Retrieve the Total Sales Amount for Each Product

```
SELECT ProductID, SUM(LineTotal) AS TotalSalesAmount

FROM salesorderdetail

GROUP BY ProductID;
```

---

32. Display All Customers Who Have Made an Online Order

```
SELECT CustomerID, SalesOrderID

FROM salesorderheader

WHERE OnlineOrderFlag = 1;
```

---

33. Identify the Top 5 Products with the Highest Sales in Terms of Quantity Ordered

```
SELECT ProductID, SUM(OrderQty) AS TotalQuantityOrdered

FROM salesorderdetail

GROUP BY ProductID

ORDER BY TotalQuantityOrdered DESC

LIMIT 5;
```

---

34. Find Details of All Orders Shipped After the Due Date

```
SELECT SalesOrderID, CustomerID, OrderDate, DueDate, ShipDate

FROM salesorderheader

WHERE ShipDate > DueDate;
```

---

35. Retrieve Customers and Their Corresponding Total Spending

```
SELECT soh.CustomerID, SUM(sod.LineTotal) AS TotalSpending

FROM salesorderheader soh

JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID

GROUP BY soh.CustomerID;
```

---

36. Identify All Orders Where the Discount Was Greater Than 10%

```
SELECT SalesOrderID, ProductID, OrderQty, UnitPrice, UnitPriceDiscount

FROM salesorderdetail

WHERE UnitPriceDiscount > 0.1;
```

---

37. Retrieve a List of All Addresses and the Number of Customers Associated with Each Address

```
SELECT a.AddressID, a.AddressLine1, a.City, COUNT(ca.CustomerID) AS
NumberOfCustomers

FROM address a

JOIN customeraddress ca ON a.AddressID = ca.AddressID

GROUP BY a.AddressID, a.AddressLine1, a.City;
```

---

38. Determine the Average Shipping Time (in Days) for All Orders

```
SELECT AVG(DATEDIFF(DAY, OrderDate, ShipDate)) AS AverageShippingTime

FROM salesorderheader

WHERE ShipDate IS NOT NULL;
```

---

39. Find the Top 3 Customers Based on Total Number of Transactions

```
SELECT CustomerID, COUNT(SalesOrderID) AS TotalTransactions

FROM salesorderheader

GROUP BY CustomerID

ORDER BY TotalTransactions DESC

LIMIT 3;
```

---

40. List All Employees Who Have Managed More Than 10 Orders

```
SELECT SalesPersonID, COUNT(SalesOrderID) AS ManagedOrders

FROM salesorderheader

GROUP BY SalesPersonID

HAVING COUNT(SalesOrderID) > 10;
```

---

41. Retrieve Customer Details and Corresponding Addresses

```
SELECT c.CustomerID, c.FirstName, c.LastName, a.AddressLine1, a.City, a.PostalCode

FROM customer c
```

JOIN customeraddress ca ON c.CustomerID = ca.CustomerID

JOIN address a ON ca.AddressID = a.AddressID;

---

42. Find Total Sales Amount for Each Product Category

SELECT pc.ProductCategoryID, pc.Name AS ProductCategoryName, SUM(sod.LineTotal) AS TotalSales

FROM salesorderdetail sod

JOIN product p ON sod.ProductID = p.ProductID

JOIN productcategory pc ON p.ProductCategoryID = pc.ProductCategoryID

GROUP BY pc.ProductCategoryID, pc.Name;

---

43. Display Customers Who Made Purchases in the Last 30 Days

SELECT c.CustomerID, c.FirstName, c.LastName, soh.OrderDate

FROM customer c

JOIN salesorderheader soh ON c.CustomerID = soh.CustomerID

WHERE soh.OrderDate >= DATEADD(DAY, -30, GETDATE());

---

44. Identify the Top 5 Products with the Highest Total Revenue

SELECT p.ProductID, p.Name AS ProductName, SUM(sod.UnitPrice * sod.OrderQty) AS TotalRevenue

FROM salesorderdetail sod

JOIN product p ON sod.ProductID = p.ProductID

```
GROUP BY p.ProductID, p.Name

ORDER BY TotalRevenue DESC

LIMIT 5;
```

---

45. Generate a Summary Report of Orders by Year and Month

```
SELECT YEAR(soh.OrderDate) AS Year, MONTH(soh.OrderDate) AS Month,

     COUNT(soh.SalesOrderID) AS TotalOrders, SUM(sod.LineTotal) AS TotalSalesAmount

FROM salesorderheader soh

JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID

GROUP BY YEAR(soh.OrderDate), MONTH(soh.OrderDate)

ORDER BY Year, Month;
```

---

46. Find Customers Who Have Not Placed Any Orders

```
SELECT c.CustomerID, c.FirstName, c.LastName

FROM customer c

LEFT JOIN salesorderheader soh ON c.CustomerID = soh.CustomerID

WHERE soh.SalesOrderID IS NULL;
```

---

47. Display the Average Discount Applied for Each Product

```
SELECT p.ProductID, p.Name AS ProductName, AVG(sod.UnitPriceDiscount) AS
AverageDiscount
```

```
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
GROUP BY p.ProductID, p.Name;
```

---

48. Total Number of Orders Handled by Each Salesperson

```
SELECT e.SalesPersonID, e.FirstName, e.LastName, COUNT(soh.SalesOrderID) AS
TotalOrders
FROM employee e
JOIN salesorderheader soh ON e.EmployeeID = soh.SalesPersonID
GROUP BY e.SalesPersonID, e.FirstName, e.LastName;
```

---

49. Most Frequently Purchased Product for Each Customer

```
WITH RankedProducts AS (
   SELECT c.CustomerID, c.FirstName, c.LastName, sod.ProductID, p.Name AS ProductName,
         ROW_NUMBER() OVER (PARTITION BY c.CustomerID ORDER BY
SUM(sod.OrderQty) DESC) AS Rank
   FROM customer c
   JOIN salesorderheader soh ON c.CustomerID = soh.CustomerID
   JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID
   JOIN product p ON sod.ProductID = p.ProductID
   GROUP BY c.CustomerID, c.FirstName, c.LastName, sod.ProductID, p.Name
)
SELECT CustomerID, FirstName, LastName, ProductID, ProductName
```

```
FROM RankedProducts

WHERE Rank = 1;
```

---

50. Products with Stock Levels Below Safety Stock Level

```
SELECT p.ProductID, p.Name AS ProductName, pi.Quantity, p.SafetyStockLevel

FROM productinventory pi

JOIN product p ON pi.ProductID = p.ProductID

WHERE pi.Quantity < p.SafetyStockLevel;
```

Intermediate level

---

51. Find the top 5 customers with the highest total sales amount.

Approach: Calculate the total sales amount for each customer using the formula: `TotalSales = (UnitPrice - UnitPriceDiscount) * OrderQty`.

```
 SELECT TOP 5
   CustomerID,
   SUM((UnitPrice - UnitPriceDiscount) * OrderQty) AS TotalSales
FROM salesorderdetail
GROUP BY CustomerID
ORDER BY TotalSales DESC;
```

-

---

52. Calculate the average order value for each customer.

Approach: Use `SUM()` to calculate total order value and divide by the count of orders.

```
 SELECT
    CustomerID,
    AVG((UnitPrice - UnitPriceDiscount) * OrderQty) AS AvgOrderValue
FROM salesorderdetail
GROUP BY CustomerID;
```

-

---

53. Retrieve the total sales amount and the number of orders for each product.

Approach: Combine `SUM()` for sales and `COUNT()` for orders.

```
 SELECT
    ProductID,
    SUM(LineTotal) AS TotalSalesAmount,
    COUNT(SalesOrderID) AS TotalOrders
FROM salesorderdetail
GROUP BY ProductID;
```

-

---

54. Find sales orders where the order was delivered late.

Approach: Compare `ShipDate` with `DueDate` to find delayed orders.

```
 SELECT
    SalesOrderID,
    OrderDate,
    DueDate,
    ShipDate
FROM salesorderheader
WHERE ShipDate > DueDate;
```

-

---

55. Identify the top 3 most frequently purchased products.

Approach: Use `SUM(OrderQty)` grouped by `ProductID` and limit the results to 3.

```
 SELECT TOP 3
    ProductID,
    SUM(OrderQty) AS TotalQuantitySold
FROM salesorderdetail
GROUP BY ProductID
ORDER BY TotalQuantitySold DESC;
```

•

---

56. Total revenue generated by each sales territory.

Approach: Use `GROUP BY TerritoryID` to calculate total revenue from `TotalDue`.

```
 SELECT
    TerritoryID,
    SUM(TotalDue) AS TotalRevenue
FROM salesorderheader
GROUP BY TerritoryID;
```

•

---

57. Customers with monthly orders exceeding $10,000.

Approach: Group orders by customer and month, then filter totals.

```
 SELECT
    CustomerID,
    YEAR(OrderDate) AS Year,
    MONTH(OrderDate) AS Month,
    SUM(TotalDue) AS TotalSales
FROM salesorderheader
GROUP BY CustomerID, YEAR(OrderDate), MONTH(OrderDate)
HAVING SUM(TotalDue) > 10000;
```

•

---

58. Products with no sales activity.

Approach: Use a `LEFT JOIN` to find products in the `product` table without matching records in `salesorderdetail`.

```
 SELECT
```

```
    p.ProductID,
    p.Name
FROM product p
LEFT JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
WHERE sod.ProductID IS NULL;
```

-

---

59. Customers who used a specific credit card type.

Approach: Join `salesorderheader` and `creditcard` tables on `CreditCardID`, filtering by `CardType`.

```
 SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    cc.CardType
FROM customer c
JOIN salesorderheader soh ON c.CustomerID = soh.CustomerID
JOIN creditcard cc ON soh.CreditCardID = cc.CreditCardID
WHERE cc.CardType = 'Visa';
```

-

---

60. Product with the highest sales revenue per unit.

Approach: Calculate `RevenuePerUnit` as `(UnitPrice - UnitPriceDiscount)` and find the maximum.

```
 SELECT TOP 1
    ProductID,
    MAX(UnitPrice - UnitPriceDiscount) AS RevenuePerUnit
FROM salesorderdetail
GROUP BY ProductID
ORDER BY RevenuePerUnit DESC;
```

---

61. Top 5 products with the highest total sales revenue

Approach: Calculate `TotalSales = UnitPrice * OrderQty` and rank the products.

```
SELECT TOP 5

    ProductID,

    SUM(UnitPrice * OrderQty) AS TotalRevenue

FROM salesorderdetail

GROUP BY ProductID

ORDER BY TotalRevenue DESC;
```

---

62. Average time between `OrderDate` and `ShipDate` for each `SalesOrderID`

Approach: Use `DATEDIFF` to calculate days and take the average.

```
SELECT

    SalesOrderID,

    AVG(DATEDIFF(DAY, OrderDate, ShipDate)) AS AvgDaysBetweenOrderAndShip

FROM salesorderheader

GROUP BY SalesOrderID

ORDER BY AvgDaysBetweenOrderAndShip DESC;
```

---

63. Top 3 customers with the highest total revenue

Approach: Sum `TotalDue` from orders grouped by customers.

```
SELECT TOP 3
```

```
    CustomerID,
    SUM(TotalDue) AS TotalRevenue
FROM salesorderheader
GROUP BY CustomerID
ORDER BY TotalRevenue DESC;
```

---

64. Monthly sales trend

Approach: Extract `MONTH` and `YEAR` from `OrderDate` and calculate monthly totals.

```
SELECT
    YEAR(OrderDate) AS Year,
    MONTH(OrderDate) AS Month,
    SUM(TotalDue) AS MonthlyRevenue
FROM salesorderheader
GROUP BY YEAR(OrderDate), MONTH(OrderDate)
ORDER BY Year, Month;
```

---

65. Percentage of online versus offline orders by `TerritoryID`

Approach: Calculate totals for online (`OnlineOrderFlag = 1`) and offline orders, and find percentages.

```
SELECT
    TerritoryID,
    SUM(CASE WHEN OnlineOrderFlag = 1 THEN 1 ELSE 0 END) * 100.0 / COUNT(*) AS OnlineOrderPercentage,
    SUM(CASE WHEN OnlineOrderFlag = 0 THEN 1 ELSE 0 END) * 100.0 / COUNT(*) AS OfflineOrderPercentage

FROM salesorderheader

GROUP BY TerritoryID;
```

---

66. Products sold at a discount and total revenue lost

Approach: Filter by `UnitPriceDiscount > 0` and calculate revenue loss.

```
SELECT

    ProductID,

    SUM(UnitPriceDiscount * OrderQty) AS TotalRevenueLost

FROM salesorderdetail

WHERE UnitPriceDiscount > 0

GROUP BY ProductID;
```

---

67. Average sales revenue per employee

Approach: Join `employee` and `salesorderheader` on employee-related keys, then calculate average revenue.

```
SELECT

    EmployeeID,

    AVG(TotalDue) AS AvgRevenuePerEmployee

FROM employee e

JOIN salesorderheader soh ON e.EmployeeID = soh.SalesPersonID

GROUP BY EmployeeID;
```

---

68. Customers using the same billing and shipping address

Approach: Compare `BillToAddressID` and `ShipToAddressID` and count orders.

```
SELECT
    CustomerID,
    COUNT(*) AS TotalOrders
FROM salesorderheader
WHERE BillToAddressID = ShipToAddressID
GROUP BY CustomerID;
```

---

69. Most frequently purchased product category

Approach: Join `productsubcateg` and `salesorderdetail` to aggregate quantities by category.

```
SELECT
    psc.Name AS CategoryName,
    SUM(sod.OrderQty) AS TotalQuantitySold
FROM productsubcateg psc
JOIN product p ON psc.ProductSubcategoryID = p.ProductSubcategoryID
JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
GROUP BY psc.Name
ORDER BY TotalQuantitySold DESC
LIMIT 1;
```

70. Identify customers with churn risk

Approach: Filter customers with `last_purchase_date` over a year old.

```
SELECT
   CustomerID,
   DATEDIFF(DAY, last_purchase_date, GETDATE()) AS DaysSinceLastPurchase
FROM customer
WHERE DATEDIFF(DAY, last_purchase_date, GETDATE()) > 365;
```

71. Top 5 products with the highest sales value for the current year

Approach: Filter by the current year and calculate total sales for each product.

```
SELECT TOP 5
   ProductID,
   SUM(LineTotal) AS TotalSales
FROM salesorderdetail
WHERE YEAR(OrderDate) = YEAR(GETDATE())
GROUP BY ProductID
ORDER BY TotalSales DESC;
```

72. Average sales per territory with sales over $10,000

Approach: Join the tables and calculate total sales per territory, filtering by those with sales over $10,000.

```
SELECT

   st.TerritoryID,

   AVG(soh.TotalDue) AS AvgSales

FROM salesorderheader soh

JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID

JOIN salesterritory st ON soh.TerritoryID = st.TerritoryID

GROUP BY st.TerritoryID

HAVING SUM(soh.TotalDue) > 10000;
```

---

73. Customers who made purchases during the holiday season

Approach: Filter by `holiday_season` to identify relevant customers.

```
SELECT DISTINCT

   c.CustomerID

FROM customertransaction ct

JOIN customer c ON ct.CustomerID = c.CustomerID

WHERE ct.holiday_season = 1;
```

---

74. Number of orders and average order value per salesperson

Approach: Join the tables to calculate the number of orders and the average order value per salesperson.

```
SELECT
    soh.SalesPersonID,
    COUNT(soh.SalesOrderID) AS NumOrders,
    AVG(soh.TotalDue) AS AvgOrderValue
FROM salesorderheader soh
GROUP BY soh.SalesPersonID;
```

---

75. Total sales per customer (online and in-store purchases)

Approach: Sum the total sales for each customer, considering both online and in-store purchases.

```
SELECT
    c.CustomerID,
    SUM(sod.LineTotal) AS TotalSales
FROM salesorderheader soh
JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID
JOIN customer c ON soh.CustomerID = c.CustomerID
GROUP BY c.CustomerID;
```

---

76. Products purchased but never shipped

Approach: Filter by orders with a `NULL` `ShipDate`.

```
SELECT
    sod.ProductID,
    soh.SalesOrderID,

    soh.OrderDate

FROM salesorderheader soh

JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID

WHERE soh.ShipDate IS NULL;
```

---

77. Customers with recent and past purchases (last 30 days but not in the last 15 days)

Approach: Calculate date differences and filter accordingly.

```
SELECT

    c.CustomerID

FROM salesorderheader soh

JOIN customer c ON soh.CustomerID = c.CustomerID

WHERE DATEDIFF(DAY, soh.OrderDate, GETDATE()) <= 30

AND DATEDIFF(DAY, soh.OrderDate, GETDATE()) > 15

GROUP BY c.CustomerID;
```

---

78. Total sales and average unit price per product category

Approach: Join the `product`, `productcategory`, and `salesorderdetail` tables, then calculate the total sales and average unit price per category.

```
SELECT

    pc.CategoryName,

    SUM(sod.LineTotal) AS TotalSales,
    AVG(sod.UnitPrice) AS AvgUnitPrice
FROM product p
JOIN productcategory pc ON p.ProductCategoryID = pc.ProductCategoryID
JOIN salesorderdetail sod ON p.ProductID = sod.ProductID
GROUP BY pc.CategoryName;
```

---

79. Top 5 products with the highest return rates

Approach: Calculate return rates using `returned items` and `total items sold` data.

```
SELECT TOP 5
    p.ProductID,
    SUM(sod.QuantityReturned) * 1.0 / SUM(sod.OrderQty) AS ReturnRate
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
GROUP BY p.ProductID
ORDER BY ReturnRate DESC;
```

---

80. Sales order details for orders placed in the last 7 days marked as "On Hold"

Approach: Filter by `OrderDate` within the last 7 days and `Status` being "On Hold".

```
SELECT
    soh.OrderDate,
    soh.SalesOrderID,
    sod.ProductID,
    sod.OrderQty,
    sod.UnitPrice
FROM salesorderheader soh
JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID
WHERE soh.Status = 'On Hold'
AND soh.OrderDate >= DATEADD(DAY, -7, GETDATE());
```

---

81. List all products along with their product categories and subcategories

Approach: Join the `product`, `productcategory`, and `productsubcategory` tables.

```
SELECT
    p.ProductName,
    pc.CategoryName,
    psc.SubcategoryName
FROM product p
JOIN productcategory pc ON p.ProductCategoryID = pc.ProductCategoryID
JOIN productsubcategory psc ON p.ProductSubcategoryID = psc.ProductSubcategoryID;
```

---

82. Total sales amount for each product in the last month

Approach: Filter the data by the last month and calculate the total sales.

```
SELECT
    p.ProductName,
    SUM(sod.OrderQty) AS QuantitySold,
    SUM(sod.LineTotal) AS TotalSales
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
WHERE MONTH(sod.OrderDate) = MONTH(DATEADD(MONTH, -1, GETDATE()))
AND YEAR(sod.OrderDate) = YEAR(GETDATE())
GROUP BY p.ProductName;
```

---

83. Average sales per salesperson in the current year

Approach: Filter by the current year and calculate average sales per salesperson.

```
SELECT
    soh.SalesPersonID,
    AVG(soh.TotalDue) AS AvgSales
FROM salesorderheader soh
WHERE YEAR(soh.OrderDate) = YEAR(GETDATE())
GROUP BY soh.SalesPersonID;
```

---

84. Names of all customers who have made purchases in the last 30 days

Approach: Filter by order date to identify customers who made purchases in the last 30 days.

```
SELECT DISTINCT
    c.CustomerName
FROM salesorderheader soh
JOIN customer c ON soh.CustomerID = c.CustomerID
WHERE soh.OrderDate >= DATEADD(DAY, -30, GETDATE());
```

---

85. Most purchased product in the last 6 months

Approach: Filter by the last 6 months and calculate the product with the highest quantity sold.

```
SELECT TOP 1
    p.ProductName,
    SUM(sod.OrderQty) AS QuantitySold
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
WHERE sod.OrderDate >= DATEADD(MONTH, -6, GETDATE())
GROUP BY p.ProductName
ORDER BY QuantitySold DESC;
```

---

86. List all sales transactions completed using a specific payment method

Approach: Filter by a specific payment method (e.g., "Credit Card").

```
SELECT
    soh.SalesOrderID,
    soh.OrderDate,
    soh.PaymentMethod,
    soh.TotalDue
FROM salesorderheader soh
WHERE soh.PaymentMethod = 'Credit Card';
```

---

87. Total sales, tax, and freight for each sales order placed in the last week

Approach: Filter by the last week and calculate the total sales, tax, and freight.

```
SELECT
    soh.SalesOrderID,
    SUM(sod.LineTotal) AS TotalSales,
    SUM(soh.TaxAmt) AS TotalTax,
    SUM(soh.Freight) AS TotalFreight
FROM salesorderheader soh
JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID
WHERE soh.OrderDate >= DATEADD(WEEK, -1, GETDATE())
GROUP BY soh.SalesOrderID;
```

---

88. List of customers eligible for promotional discounts

Approach: Identify customers who have made a certain number of purchases or exceeded a spending threshold, indicating eligibility for discounts.

```
SELECT
    c.CustomerID,
    c.CustomerName
```

```
FROM customer c

JOIN salesorderheader soh ON c.CustomerID = soh.CustomerID

WHERE soh.TotalDue > 500

GROUP BY c.CustomerID, c.CustomerName

HAVING COUNT(soh.SalesOrderID) > 3;
```

---

89. Top 5 products with the highest return rate

Approach: Calculate return rates based on the number of returned items from the `productinventory` and `salesorderdetail` tables.

```
SELECT TOP 5

   p.ProductName,

   SUM(sod.QuantityReturned) * 1.0 / SUM(sod.OrderQty) AS ReturnRate

FROM salesorderdetail sod

JOIN product p ON sod.ProductID = p.ProductID

GROUP BY p.ProductName

ORDER BY ReturnRate DESC;
```

---

90. Total sales value by each salesperson in a particular territory during the current quarter

Approach: Filter by the current quarter and calculate total sales for each salesperson.

```
SELECT

   soh.SalesPersonID,

   SUM(soh.TotalDue) AS TotalSales
```

```
FROM salesorderheader soh

JOIN salesterritory st ON soh.TerritoryID = st.TerritoryID

WHERE YEAR(soh.OrderDate) = YEAR(GETDATE())

AND DATEPART(QUARTER, soh.OrderDate) = DATEPART(QUARTER, GETDATE())

GROUP BY soh.SalesPersonID;
```

---

91. Total sales for each product category in a specific year

Approach: Join the `salesorderdetail` and `product` tables, group by `ProductCategoryID`, and sum the sales for that category.

```
SELECT

   p.ProductCategoryID,

   SUM(sod.LineTotal) AS TotalSales

FROM salesorderdetail sod

JOIN product p ON sod.ProductID = p.ProductID

WHERE YEAR(sod.OrderDate) = 2023  -- Replace with the desired year

GROUP BY p.ProductCategoryID;
```

---

92. Average transaction value for each customer segment based on `CustomerType`

Approach: Group by `CustomerType` and calculate the average `TotalDue` from the `salesorderheader` table.

```
SELECT
    c.CustomerType,
    AVG(soh.TotalDue) AS AvgTransactionValue
FROM salesorderheader soh
JOIN customer c ON soh.CustomerID = c.CustomerID
GROUP BY c.CustomerType;
```

---

93. Customers who have not made any purchases in the last 6 months

Approach: Identify customers with no orders within the last 6 months.

```
SELECT
    c.CustomerID,
    c.CustomerName
FROM customer c
LEFT JOIN salesorderheader soh ON c.CustomerID = soh.CustomerID
WHERE soh.OrderDate < DATEADD(MONTH, -6, GETDATE()) OR soh.OrderDate IS NULL;
```

---

94. Percentage of orders shipped on time for each sales territory

Approach: Join `salesorderheader` and `salesterritory`, calculate the percentage of orders shipped on time.

```
SELECT
    st.TerritoryID,
    COUNT(CASE WHEN soh.ShipDate <= soh.DueDate THEN 1 END) * 100.0 /
COUNT(soh.SalesOrderID) AS OnTimePercentage
```

```
FROM salesorderheader soh

JOIN salesterritory st ON soh.TerritoryID = st.TerritoryID

GROUP BY st.TerritoryID;
```

---

95. Total revenue generated from orders using a special offer

Approach: Join `salesorderdetail` and `specialoffer` tables, then sum the `LineTotal` for orders with special offers.

```
SELECT

   SUM(sod.LineTotal) AS TotalRevenue

FROM salesorderdetail sod

JOIN specialoffer so ON sod.SpecialOfferID = so.SpecialOfferID

WHERE so.SpecialOfferID IS NOT NULL;
```

---

96. Top 5 customers with the highest total spending in the last year

Approach: Sum `TotalDue` for each customer in the last year and order by the highest total spending.

```
SELECT TOP 5

   c.CustomerID,

   c.CustomerName,

   SUM(soh.TotalDue) AS TotalSpending

FROM salesorderheader soh

JOIN customer c ON soh.CustomerID = c.CustomerID

WHERE YEAR(soh.OrderDate) = YEAR(GETDATE()) - 1
```

```
GROUP BY c.CustomerID, c.CustomerName

ORDER BY TotalSpending DESC;
```

---

97. Products that were never sold during a particular month

Approach: Find products with no sales during the given month.

```
SELECT

    p.ProductID,

    p.ProductName

FROM product p

LEFT JOIN salesorderdetail sod ON p.ProductID = sod.ProductID

WHERE MONTH(sod.OrderDate) = 5  -- Replace with the desired month (e.g., May)

AND YEAR(sod.OrderDate) = 2023  -- Replace with the desired year

AND sod.ProductID IS NULL;
```

---

98. Month-over-month sales growth for a specific product category

Approach: Group by month and calculate sales for each month, then compute percentage change.

```
WITH MonthlySales AS (

    SELECT

        MONTH(soh.OrderDate) AS Month,

        YEAR(soh.OrderDate) AS Year,

        p.ProductCategoryID,

        SUM(sod.LineTotal) AS SalesAmount
    FROM salesorderheader soh

    JOIN salesorderdetail sod ON soh.SalesOrderID = sod.SalesOrderID

    JOIN product p ON sod.ProductID = p.ProductID

    WHERE p.ProductCategoryID = 101  -- Replace with the desired ProductCategoryID

    GROUP BY YEAR(soh.OrderDate), MONTH(soh.OrderDate), p.ProductCategoryID

)

SELECT

    Month,

    Year,

    SalesAmount,

    LAG(SalesAmount) OVER (ORDER BY Year, Month) AS PreviousMonthSales,

    (SalesAmount - LAG(SalesAmount) OVER (ORDER BY Year, Month)) * 100.0 /
LAG(SalesAmount) OVER (ORDER BY Year, Month) AS MoMSalesGrowth

FROM MonthlySales

ORDER BY Year, Month;
```

---

99. List customers who used a particular payment method with their total spending

Approach: Join the `customer`, `salesorderheader`, and `paymentmethod` tables to filter by payment method and calculate total spending.

```
SELECT

    c.CustomerID,

    c.CustomerName,

    SUM(soh.TotalDue) AS TotalSpending
```

```
FROM salesorderheader soh
JOIN customer c ON soh.CustomerID = c.CustomerID
WHERE soh.PaymentMethod = 'Credit Card'  -- Replace with the desired payment method
GROUP BY c.CustomerID, c.CustomerName;
```

---

100. Average discount offered per product category

Approach: Join the `salesorderdetail` and `productcategory` tables and calculate the average discount per category.

```
SELECT
    p.ProductCategoryID,
    AVG(sod.UnitPriceDiscount) AS AvgDiscount
FROM salesorderdetail sod
JOIN product p ON sod.ProductID = p.ProductID
GROUP BY p.ProductCategoryID;
```
