**Sample Database Schema**

**Customers Table:**

| **CustomerID** | **CustomerName** |
| --- | --- |
| 1 | John Doe |
| 2 | Jane Smith |
| 3 | Michael Brown |
| 4 | Emily Davis |

**Orders Table:**

| **OrderID** | **CustomerID** | **OrderDate** | **ProductID** |
| --- | --- | --- | --- |
| 101 | 1 | 2023-01-10 | 1 |
| 102 | 3 | 2023-02-15 | 2 |
| 103 | 4 | 2023-03-05 | 3 |
| 104 | 1 | 2023-04-12 | 2 |

**Products Table:**

| **ProductID** | **ProductName** | **Price** |
| --- | --- | --- |
| 1 | Laptop | 1000 |
| 2 | Smartphone | 500 |
| 3 | Tablet | 300 |
| 4 | Headphones | 100 |

**Examples of Aggregate Functions**

1. **COUNT()**: Count the number of orders placed by each customer.

SELECT CustomerID, COUNT(OrderID) AS OrderCount

FROM Orders

GROUP BY CustomerID;

**Result:**

| **CustomerID** | **OrderCount** |
| --- | --- |
| 1 | 2 |
| 3 | 1 |
| 4 | 1 |

1. **SUM()**: Calculate the total price of all products ordered.

SELECT SUM(Products.Price) AS TotalRevenue

FROM Orders

INNER JOIN Products ON Orders.ProductID = Products.ProductID;

**Result:**

| **TotalRevenue** |
| --- |
| 2300 |

1. **AVG()**: Find the average price of products ordered.

SELECT AVG(Products.Price) AS AveragePrice

FROM Orders

INNER JOIN Products ON Orders.ProductID = Products.ProductID;

**Result:**

| **AveragePrice** |
| --- |
| 575 |

1. **MIN()**: Find the lowest priced product.

SELECT MIN(Price) AS LowestPrice

FROM Products;

**Result:**

| **LowestPrice** |
| --- |
| 100 |

1. **MAX()**: Find the highest priced product.

SELECT MAX(Price) AS HighestPrice

FROM Products;

**Result:**

| **HighestPrice** |
| --- |
| 1000 |

**Additional Examples**

1. **COUNT(DISTINCT)**: Count the number of distinct customers who have placed orders.

SELECT COUNT(DISTINCT CustomerID) AS UniqueCustomers

FROM Orders;

**Result:**

| **UniqueCustomers** |
| --- |
| 3 |

1. **SUM() with GROUP BY**: Calculate the total revenue generated by each product.

SELECT Products.ProductName, SUM(Products.Price) AS TotalRevenue

FROM Orders

INNER JOIN Products ON Orders.ProductID = Products.ProductID

GROUP BY Products.ProductName;

**Result:**

| **ProductName** | **TotalRevenue** |
| --- | --- |
| Laptop | 1000 |
| Smartphone | 1000 |
| Tablet | 300 |

1. **AVG() with GROUP BY**: Find the average price of products ordered by each customer.

SELECT Customers.CustomerName, AVG(Products.Price) AS AverageSpent

FROM Orders

INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID

INNER JOIN Products ON Orders.ProductID = Products.ProductID

GROUP BY Customers.CustomerName;

**Result:**

| **CustomerName** | **AverageSpent** |
| --- | --- |
| John Doe | 750 |
| Michael Brown | 500 |
| Emily Davis | 300 |

1. **HAVING**: Use aggregate functions with a HAVING clause to filter groups. Find products ordered more than once.

SELECT ProductID, COUNT(OrderID) AS OrderCount

FROM Orders

GROUP BY ProductID

HAVING COUNT(OrderID) > 1;

**Result:**

| **ProductID** | **OrderCount** |
| --- | --- |
| 2 | 2 |

1. **Complex Query**: Calculate the total revenue and average order value for each customer.

SELECT Customers.CustomerName, SUM(Products.Price) AS TotalSpent, AVG(Products.Price) AS AverageOrderValue

FROM Orders

INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID

INNER JOIN Products ON Orders.ProductID = Products.ProductID

GROUP BY Customers.CustomerName;

**Result:**

| **CustomerName** | **TotalSpent** | **AverageOrderValue** |
| --- | --- | --- |
| John Doe | 1500 | 750 |
| Michael Brown | 500 | 500 |
| Emily Davis | 300 | 300 |

These examples showcase how aggregate functions can be used to perform calculations and summarize data in SQL.