Summarize data using SQL Server extensions

```
CREATE TABLE Restaurants(
RestaurantID INT PRIMARY KEY,
RestaurantName VARCHAR(100) NOT NULL,
Cuisine VARCHAR(50) NOT NULL,
Location VARCHAR(100) NOT NULL,
Rating FLOAT
)

INSERT INTO Restaurants (RestaurantID, RestaurantName, Cuisine, Location, Rating)
VALUES
(1, 'Tasty Bites', 'Italian', 'Downtown', 4.5),
(2, 'Spice Delight', 'Indian', 'Midtown', 4.0),
(3, 'Sushi Haven', 'Japanese', 'Uptown', 4.2),
(4, 'Burger Joint', 'American', 'Downtown', 3.8),
(5, 'Curry House', 'Indian', 'Suburb', 4.1),
(6, 'Pizza Palace', 'Italian', 'Uptown', 4.3);

SELECT * FROM Restaurants

SELECT Cuisine, Location, AVG(Rating) AS AvgRating
FROM Restaurants
GROUP BY Cuisine, Location

SELECT Cuisine, Location, AVG(Rating) AS AvgRating
FROM Restaurants
GROUP BY ROLLUP(Cuisine, Location)

SELECT Cuisine, Location, AVG(Rating) AS AvgRating
FROM Restaurants
GROUP BY CUBE(Cuisine, Location)

SELECT * FROM Restaurants
PIVOT (
AVG(Rating)
FOR Cuisine IN ([Italian], [Japanese], [American], [Indian])
)
AS PivotTable;
```

```
CREATE TABLE Sales (
    SaleDate DATE,
```

```
    ProductCategory VARCHAR(50),
    SalesAmount DECIMAL(10, 2)
);


INSERT INTO Sales (SaleDate, ProductCategory, SalesAmount) VALUES
('2024-01-01', 'Electronics', 1200.00),
('2024-01-01', 'Clothing', 300.00),
('2024-01-02', 'Electronics', 1500.00),
('2024-01-02', 'Clothing', 200.00),
('2024-01-03', 'Electronics', 800.00),
('2024-01-03', 'Clothing', 400.00);


SELECT
    SaleDate,
    ProductCategory,
    SUM(SalesAmount) AS TotalSales
FROM Sales
GROUP BY SaleDate, ProductCategory

SELECT
SaleDate,
ProductCategory,
SUM(SalesAmount) AS TotalSales
FROM SALES
GROUP BY ROLLUP(SaleDate,
ProductCategory)

SELECT
    SaleDate,
    ProductCategory,
    SUM(SalesAmount) AS TotalSales
FROM Sales
GROUP BY
    CUBE(SaleDate, ProductCategory);

SELECT * FROM Sales

SELECT
    SaleDate,
    Electronics,
    Clothing
FROM Sales
PIVOT
    (SUM(SalesAmount) FOR ProductCategory IN ([Electronics], [Clothing])) AS
PivotTable;
```

Views

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName NVARCHAR(100) NOT NULL
)

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderTotal DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
)

INSERT INTO Customers (CustomerID, CustomerName)
VALUES
    (1, 'John Doe'),
    (2, 'Jane Smith'),
    (3, 'Alice Johnson')

INSERT INTO Orders (OrderID, CustomerID, OrderTotal)
VALUES
    (1, 1, 500.00),
    (2, 1, 200.00),
    (3, 2, 150.00),
    (4, 3, 800.00),
    (5, 3, 1200.00)

SELECT * FROM Customers
SELECT * FROM Orders

SELECT
c.CustomerID,
c.CustomerName,
SUM(o.OrderTotal) AS TotalSales
FROM Customers c JOIN Orders o
ON c.CustomerID=o.CustomerID
GROUP BY c.CustomerID,c.CustomerName
HAVING SUM(o.OrderTotal) > 500
```

```
CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName NVARCHAR(100) NOT NULL
)

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderTotal DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
)

INSERT INTO Customers (CustomerID, CustomerName)
VALUES
```

```
    (1, 'John Doe'),
    (2, 'Jane Smith'),
    (3, 'Alice Johnson')

INSERT INTO Orders (OrderID, CustomerID, OrderTotal)
VALUES
    (1, 1, 500.00),
    (2, 1, 200.00),
    (3, 2, 150.00),
    (4, 3, 800.00),
    (5, 3, 1200.00)

SELECT * FROM Customers
SELECT * FROM Orders


CREATE VIEW TOPSALESCUSTOMERS AS
SELECT
c.CustomerID,
c.CustomerName,
SUM(o.OrderTotal) AS TotalSales
FROM Customers c JOIN Orders o
ON c.CustomerID=o.CustomerID
GROUP BY c.CustomerID,c.CustomerName
HAVING SUM(o.OrderTotal) > 500


SELECT * FROM TOPSALESCUSTOMERS
WHERE CustomerID=1

ALTER VIEW TOPSALESCUSTOMERS AS
SELECT
TOP 2
c.CustomerID,
c.CustomerName,
SUM(o.OrderTotal) AS TotalSales
FROM Customers c JOIN Orders o
ON c.CustomerID=o.CustomerID
GROUP BY c.CustomerID,c.CustomerName
HAVING SUM(o.OrderTotal) > 100
```

Indexing

```
CREATE TABLE Restaurants(
RestaurantID INT PRIMARY KEY,
RestaurantName VARCHAR(100) NOT NULL,
Cuisine VARCHAR(50) NOT NULL,
Location VARCHAR(100) NOT NULL,
Rating FLOAT
)

INSERT INTO Restaurants (RestaurantID, RestaurantName, Cuisine, Location, Rating)
VALUES
(1, 'Tasty Bites', 'Italian', 'Downtown', 4.5),
(2, 'Spice Delight', 'Indian', 'Midtown', 4.0),
```

```
(3, 'Sushi Haven', 'Japanese', 'Uptown', 4.2),
(4, 'Burger Joint', 'American', 'Downtown', 3.8),
(5, 'Curry House', 'Indian', 'Suburb', 4.1),
(6, 'Pizza Palace', 'Italian', 'Uptown', 4.3);

CREATE INDEX idx_Cuisine ON Restaurants(Cuisine)

SELECT * FROM Restaurants
WHERE Cuisine = 'Indian'

DROP INDEX idx_Cuisine ON Restaurants
```

Other db objects

Views: Virtual tables created by querying one or more tables, allowing for simplified data access and abstraction.

Stored Procedures: Predefined collections of SQL statements that can be executed as a unit, often used to encapsulate complex operations and improve performance.

Functions: SQL routines that return a single value or table, used to perform calculations or return results based on input parameters.

Triggers: Automatic actions executed in response to specific events (e.g., INSERT, UPDATE, DELETE) on a table, used to enforce business rules or maintain data integrity.

Stored Procedure: Used for performing actions or tasks; does not return a value directly; called using `EXEC`.

Function: Used for calculations or returning values; must return a value or table; called within SQL queries.

Procedures

```
CREATE PROCEDURE GetRestaurantsByCuisine
    @cuisine varchar(50)
AS
BEGIN
SELECT * FROM Restaurants
WHERE Cuisine = @cuisine
END

EXEC GetRestaurantsByCuisine 'Indian'
EXEC GetRestaurantsByCuisine @cuisine='Italian'
```

```
CREATE TABLE Students(
    StudentID INT,
    Subject varchar(50),
    Grade INT
)


CREATE Procedure InsertStudent
@StudentID INT,
@Subject varchar(50),
```

```
@Grade INT
AS
INSERT INTO Students VALUES (@StudentID,@Subject,@Grade)
GO


EXEC InsertStudent 4,'Deepthi',90
EXEC InsertStudent 6,'Divya',99

SELECT * FROM Students
```

Triggers

```
-- Create SumTable
CREATE TABLE SumTable (
    ID INT PRIMARY KEY IDENTITY,
    RestaurantCount INT
);



-- Create Trigger to update SumTable
CREATE TRIGGER UpdateRestaurantCount1
ON Restaurants
AFTER INSERT
AS
BEGIN
    DECLARE @RestaurantCount INT;

    -- Calculate the number of newly inserted restaurants
    SELECT @RestaurantCount = COUNT(*) FROM INSERTED;

    -- Insert the new RestaurantCount into SumTable
    INSERT INTO SumTable (RestaurantCount)
    VALUES (@RestaurantCount);
END;


-- Insert a value into SumTable (valid for only RestaurantCount)
INSERT INTO SumTable (RestaurantCount) VALUES (1);
INSERT INTO SumTable (RestaurantCount) VALUES (10);

-- View the contents of SumTable
SELECT * FROM SumTable;
```

CTE

```
CREATE TABLE Students(
    StudentID int,
    Name varchar(20),
    Age int
)
```

```sql
CREATE TABLE Grades(
StudentID INT,
Marks FLOAT
)

INSERT INTO Students
VALUES (1, 'a', 10),(2,'b',20),(3,'c', 12)

INSERT INTO Grades (StudentID, Marks)
VALUES (1, 85.5);
INSERT INTO Grades (StudentID, Marks)
VALUES (2, 92.0);
INSERT INTO Grades (StudentID, Marks)
VALUES (3, 78.3);
INSERT INTO Grades (StudentID, Marks)
VALUES (4, 64.7);

SELECT * FROM Students

SELECT Name FROM Students
WHERE Age=(SELECT MAX(Age)
FROM Students)

SELECT Name FROM Students
WHERE Age IN (SELECT MAX(Age)
FROM Students)

SELECT s.StudentID, s.name
FROM Students s JOIN  Grades g
ON s.StudentID=g.StudentID
WHERE g.Marks>(SELECT AVG(Marks) FROM GRADES)


WITH STUDENTCTE AS (
SELECT s.name, s.Age, g.Marks
FROM STUDENTS s
INNER JOIN Grades g ON s.StudentID = g.StudentID
)
SELECT name, Age, Marks
FROM STUDENTCTE
WHERE Marks = (SELECT MAX(Marks) FROM STUDENTCTE);
```