sql assignment

1. Types of Views in SQL

A **view** is a virtual table based on the result-set of an SQL query. It can contain rows and columns from one or more tables.

Types of Views:

Type	Description
Simple View	Based on a single table, without group functions or joins.
Complex View	Based on multiple tables, includes group functions and joins.
Materialized View	Stores the result physically and updates periodically. Mostly used in Oracle.
Inline View	A subquery in the `FROM` clause treated as a table.

Examples:

```
-- Simple View
CREATE VIEW vw_EmployeeNames AS
SELECT EmployeeID, FirstName, LastName FROM Employees;

-- Complex View
CREATE VIEW vw_DepartmentEmployees AS
SELECT D.DepartmentName, E.FirstName, E.LastName
FROM Employees E
JOIN Departments D ON E.DepartmentID = D.DepartmentID;

-- Inline View (used in a SELECT statement)
SELECT *
FROM (SELECT FirstName, Salary FROM Employees WHERE Salary > 50000)
AS HighEarners;
```

2. Difference Between Function and Stored Procedure

Feature	Function	Stored Procedure
Returns Value	Must return a value	May or may not return a value
Used in Queries	Can be used in SELECT statements	Cannot be used in SELECT statements directly
Input Parameters	Only input parameters	Can have input, output, or both
DML Operations	Limited (mostly read-only)	Can perform all DML operations

Syntax:

Function

```
CREATE FUNCTION dbo.GetFullName(@FirstName NVARCHAR(50), @LastNa
me NVARCHAR(50))
RETURNS NVARCHAR(100)
AS
BEGIN
    RETURN @FirstName + ' ' + @LastName;
END;
```

Stored Procedure

```
CREATE PROCEDURE dbo.GetEmployeeDetails
    @DepartmentID INT
AS
BEGIN
    SELECT * FROM Employees WHERE DepartmentID = @DepartmentID;
END;
```

---

3. What is an Index in SQL?

An **index** is a database object that improves the speed of data retrieval. Think of it like an index in a book.

Types of Indexes:

Type	Description
Clustered Index	Sorts and stores the data rows in the table based on key values. Only one per table.
Non-Clustered	Creates a separate structure for the index, pointing to the actual data. Multiple allowed.
Unique Index	Does not allow duplicate values in the indexed column.
Composite Index	Index on multiple columns.

Example:

```
-- Clustered Index
CREATE CLUSTERED INDEX idx_EmployeeID ON Employees(EmployeeID);

-- Non-Clustered Index
CREATE NONCLUSTERED INDEX idx_LastName ON Employees(LastName);
```

---

4. Exception Handling in Stored Procedure

```
CREATE PROCEDURE dbo.DivideNumbers
    @Num1 INT,
    @Num2 INT
AS
BEGIN
    BEGIN TRY
        DECLARE @Result FLOAT;
        SET @Result = @Num1 / @Num2;
        PRINT 'Result: ' + CAST(@Result AS VARCHAR);
    END TRY
```

```
    BEGIN CATCH
        PRINT 'Error: Division by zero or other error occurred.';
    END CATCH
END;
```

5. Function to Split String into Rows

```
CREATE FUNCTION dbo.SplitString (
    @Input NVARCHAR(MAX),
    @Delimiter CHAR(1)
)
RETURNS @Output TABLE (Value NVARCHAR(100))
AS
BEGIN
    DECLARE @Start INT = 1, @End INT;
    SET @Input = @Input + @Delimiter;

    WHILE CHARINDEX(@Delimiter, @Input, @Start) > 0
    BEGIN
        SET @End = CHARINDEX(@Delimiter, @Input, @Start);
        INSERT INTO @Output(Value)
        VALUES (LTRIM(RTRIM(SUBSTRING(@Input, @Start, @End - @Start))));
        SET @Start = @End + 1;
    END

    RETURN;
END;

-- Example usage:
SELECT * FROM dbo.SplitString('Stephen;peter;berry;Olivier;caroline;', ';');
```

6. Temporary Table vs Variable Table

Feature	Temporary Table	Variable Table
Scope	Connection/session-wide	Batch or procedure scope
Prefix	Begins with `#` or `##`	Declared using `DECLARE` keyword
Indexing	Can have indexes	Limited indexing support

Syntax:

Temporary Table

```
CREATE TABLE #TempEmployees (
    ID INT,
    Name NVARCHAR(100)
);
```

Table Variable

```
DECLARE @EmpTable TABLE (
    ID INT,
    Name NVARCHAR(100)
);
```
