**SQL Assignment 4**

1)Explain different types of views. Demonstrate with suitable examples.

A) **Simple View**: Simple views are views that are created on a single table. We can perform only basic SQL operations in simple views. That means, we cannot perform analytical and aggregate operations by grouping, sets, etc. in simple views. We can definitely perform insert, update, delete directly from a simple view, but for that, we must have the primary key column in the view.

Ex: CREATE VIEW employee\_dept1\_view AS

SELECT \* FROM employee

WHERE department\_id = 'D01';

2) **Complex View**: Complex views as the name suggest are a bit complicated compared to simple views. Complex views are created on more than one database table. We can perform analytical and aggregate operations in complex views, but unlike simple views, we cannot perform insert, delete, and update directly from a complex view.

Ex:

CREATE VIEW employee\_details AS

SELECT e.employee\_name,

e.salary,

e.highest\_qualification,

d.department\_name,

d.location

FROM

employee as e JOIN departments as d

ON e.department\_id = d.department\_id;

2) What is the difference between function and stored procedure? Write syntax for creating functions and stored procedures.

A)Write a user-defined function when you want to compute and return a value for use in other SQL statements; write a stored procedure when you want instead is to group a possibly.

Syn:

CREATE FUNCTION CalculateTotalPrice(@Quantity INT, @UnitPrice DECIMAL(10, 2))

RETURNS DECIMAL(10, 2)

AS

BEGIN

DECLARE @TotalPrice DECIMAL(10, 2);

SET @TotalPrice = @Quantity \* @UnitPrice;

RETURN @TotalPrice;

END;

Syn:

CREATE PROCEDURE UpdateStockQuantity

@ProductID INT,

@NewQuantity INT

AS

BEGIN

-- Update stock quantity

UPDATE Products

SET StockQuantity = @NewQuantity

WHERE ProductID = @ProductID;

-- Log the transaction

INSERT INTO AuditLog (ProductID, TransactionType, TransactionDate)

VALUES (@ProductID, 'Stock Update', GETDATE());

END;

3) What is an index in SQL? What are the different types of indexes in SQL?

A)  An index is a schema object. It is used by the server to speed up the retrieval of rows by using a pointer.

* The following table lists the types of indexes available in SQL Server and provides links to additional information. With a hash index, data is accessed through an in-memory hash table. Hash indexes consume a fixed amount of memory, which is a function of the bucket count.

4) Showcase an example of exception handling in SQL stored procedure.

A) CREATE PROCEDURE InsertEmployee

@EmployeeID INT,

@EmployeeName NVARCHAR(50)

AS

BEGIN

BEGIN TRY

-- Check if EmployeeID already exists

IF EXISTS (SELECT 1 FROM Employees WHERE EmployeeID = @EmployeeID)

BEGIN

THROW 50001, 'EmployeeID already exists.', 1;

END

-- Insert new employee

INSERT INTO Employees (EmployeeID, EmployeeName)

VALUES (@EmployeeID, @EmployeeName);

PRINT 'Employee inserted successfully.';

END TRY

BEGIN CATCH

-- Handle the exception

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

-- Log the error or take appropriate action

PRINT 'Error occurred: ' + @ErrorMessage;

-- Optionally, re-throw the error

-- THROW; -- Uncomment this line if you want to propagate the error to

5) Create a SQL function to split strings into rows on a given character?

Input String: Stephen;peter;berry;Olivier;caroline;

1. CREATE FUNCTION dbo.SplitStringToRows

(

@InputString NVARCHAR(MAX),

@Delimiter CHAR(1)

)

RETURNS TABLE

AS

RETURN

(

SELECT value AS SplitValue

FROM STRING\_SPLIT(@InputString, @Delimiter)

)

6) What is a temporary and a variable table? Write suitable syntax to create temporary tables and variable tables.

A) Temporary Tables and Table variables in SQL Server, both have their own pros and cons. We need to decide which one to use and when. The table variable (@table) is created in the memory. Whereas, a Temporary table (#temp) is created in the tempdb database.

Syn:

-- Create a local temporary table

CREATE TABLE #TempTable (

ID INT,

Name NVARCHAR(50)

);

-- Insert data into the temporary table

INSERT INTO #TempTable (ID, Name)

VALUES (1, 'John'), (2, 'Jane');

-- Query the temporary table

SELECT \* FROM #TempTable;

-- Drop the temporary table when done

DROP TABLE #TempTable;

A table variable is a variable of table type that exists only in the scope of the batch, function, or stored procedure in which it is declared. Table variables are useful for smaller sets of data, and their scope is limited to the specific block of code in which they are defined.

Syn:

-- Declare a table variable

DECLARE @TableVariable TABLE (

ID INT,

Name NVARCHAR(50)

);

-- Insert data into the table variable

INSERT INTO @TableVariable (ID, Name)

VALUES (1, 'Alice'), (2, 'Bob');

-- Query the table variable

SELECT \* FROM @TableVariable;