**Store procedure**

A stored procedure is a precompiled set of SQL statements that are stored in a database and can be executed repeatedly.

Stored procedures can improve performance and security by reducing network traffic and preventing SQL injection attacks.

1. **Create a stored procedure**

CREATE PROCEDURE GetCustomers

AS

begin

SELECT \* FROM Customers

End

1. **Execute SP**

execute GetCustomers

1. **Drop**

drop proc GetCustomerByID

1. **Alter**

alter PROCEDURE GetCustomerByID

AS

begin

SELECT \* FROM Customers

End

1. **Alter**

sp\_rename 'old\_procedure\_name', 'new\_procedure\_name'

**Store procedure + Variables**

--\_\_\_\_\_\_\_\_ Variable in MSSQL \_\_\_\_\_\_\_\_\_\_\_\_\_\_

declare @id int;

set @id = 5;

select @id as variable;

***In Store Procedures We Do 80% Work with Variables.***

**Example 1.**

create PROCEDURE GetCustomerByID

@CustomerID INT

AS

begin

SELECT \* FROM Customers WHERE id = @CustomerID

End

**Example 2.**

--\_\_\_Return Count\_\_\_\_\_\_

CREATE PROCEDURE GetCustomerCount

AS

BEGIN

DECLARE @Count INT

SELECT @Count = COUNT(\*) FROM Customers

SELECT @Count AS 'CustomerCount'

END

--\_\_\_Execute Procedure\_\_\_\_\_\_

exec GetCustomerCount

**Example 3.**

**Example 4. Try Catch 🡪 prhnaa ka baad**

CREATE PROCEDURE InsertCustomer @CustomerName VARCHAR(50) AS BEGIN TRY INSERT INTO Customers (CustomerName) VALUES (@CustomerName) END TRY BEGIN CATCH SELECT ERROR\_MESSAGE() AS ErrorMessage END CATCH

1. Use dynamic SQL to create and execute SQL statements within a stored procedure. For example:

sqlCopy code

CREATE PROCEDURE GetCustomersByCountry @Country VARCHAR(50) AS BEGIN DECLARE @SQL NVARCHAR(MAX) SET @SQL = 'SELECT \* FROM Customers WHERE Country = ''' + @Country + '''' EXECUTE sp\_executesql @SQL END

These are just a few examples of the many possibilities for creating and using stored procedures in Microsoft SQL Server. With practice and experience, you can become proficient in designing and implementing complex stored procedures to improve the performance and security of your database applications.

**CRUD**

**create table customerPO(**

**id int primary key identity,**

**CustomerName varchar(25),**

**Address varchar(25),**

**City varchar(25),**

**Country varchar(25)**

**)**

INSERT INTO customerPO (CustomerName, Address, City, Country) values

('John Doe', '123 Main St', 'Anytown', 'USA'),

('Jane Smith', '456 Oak St', 'Othertown', 'Canada'),

('Bob Johnson', '789 Elm St', 'Somewhere', 'Mexico'),

('Alice Lee', '321 Pine St', 'Nowhere', 'Australia');

1. **Create Procedure for Insert:**

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1. Insert Customer \_\_\_\_\_\_\_\_\_\_

CREATE PROCEDURE InsertCustomer

@CustomerName VARCHAR(50),

@Address VARCHAR(100),

@City VARCHAR(50),

@Country VARCHAR(50)

AS

BEGIN

INSERT INTO customerPO (CustomerName, Address, City, Country)

VALUES (@CustomerName, @Address, @City, @Country)

END

exec InsertCustomer 'Saqib','Karachi korangi','Karachi','Pakistan'

1. **Create Procedure for Read/Select:**

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Get / Read / Select \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

--\_\_\_\_\_ get by Id \_\_\_\_\_

CREATE PROCEDURE GetCustomerByID

@CustomerID INT

AS

BEGIN

SELECT \* FROM CustomerPO WHERE id = @CustomerID

END

exec GetCustomerByID 1

--\_\_\_\_\_ Get All \_\_\_\_\_\_

CREATE PROCEDURE GetCustomer

AS

BEGIN

SELECT \* FROM CustomerPO

END

exec GetCustomer

1. **Create Procedure for Update:**

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Update \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CREATE PROCEDURE UpdateCustomer

@CustomerID INT,

@CustomerName VARCHAR(50),

@Address VARCHAR(100),

@City VARCHAR(50),

@Country VARCHAR(50)

AS

BEGIN

UPDATE CustomerPO

SET CustomerName = @CustomerName, Address = @Address, City = @City, Country = @Country

WHERE id = @CustomerID;

select \* from CustomerPO

END

exec UpdateCustomer 7,'Rajaa','sfksdkf sf','Karachi','India'

1. **Delete**

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Delete \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CREATE PROCEDURE DeleteCustomer

@CustomerID INT

AS

BEGIN

DELETE FROM CustomerPO WHERE id = @CustomerID ;

select \* from CustomerPO ;

END

exec DeleteCustomer 2

**Complex Example insert , update,delete,select, -- validatin in Single SP**

create database practice2

use practice2

SELECT \* FROM information\_schema.tables

EXEC sp\_MSforeachtable 'DROP TABLE ?'

CREATE TABLE Customers (

CustomerId INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

TotalOrders INT,

TotalOrderAmount MONEY

);

INSERT INTO Customers (CustomerId, FirstName, LastName, Email, TotalOrders, TotalOrderAmount)

VALUES

(1, 'John', 'Doe', 'johndoe@email.com', 0, 0),

(2, 'Jane', 'Doe', 'janedoe@email.com', 0, 0),

(3, 'Bob', 'Smith', 'bobsmith@email.com', 0, 0),

(4, 'Alice', 'Jones', 'alicejones@email.com', 0, 0),

(5, 'Tom', 'Brown', 'tombrown@email.com', 0, 0),

(6, 'Sarah', 'Lee', 'sarahlee@email.com', 0, 0),

(7, 'David', 'Chen', 'davidchen@email.com', 0, 0),

(8, 'Maria', 'Garcia', 'mariagarcia@email.com', 0, 0),

(9, 'Peter', 'Johnson', 'peterjohnson@email.com', 0, 0),

(10, 'Amy', 'Kim', 'amykim@email.com', 0, 0);

CREATE TABLE Orders (

OrderId INT PRIMARY KEY,

CustomerId INT,

OrderDate DATE,

OrderTotal MONEY,

CONSTRAINT FK\_Orders\_Customers FOREIGN KEY (CustomerId) REFERENCES Customers(CustomerId)

);

INSERT INTO Orders (OrderId, CustomerId, OrderDate, OrderTotal)

VALUES

(1, 2, '2022-01-01', 100),

(2, 4, '2022-01-02', 200),

(3, 1, '2022-01-03', 150),

(4, 5, '2022-01-04', 75),

(5, 3, '2022-01-05', 300),

(6, 7, '2022-01-06', 125),

(7, 9, '2022-01-07', 50),

(8, 6, '2022-01-08', 175),

(9, 8, '2022-01-09', 225),

(10, 10, '2022-01-10', 80);

CREATE TABLE Products (

ProductId INT PRIMARY KEY,

ProductName VARCHAR(50),

UnitPrice MONEY,

QuantityInStock INT

);

INSERT INTO Products (ProductId, ProductName, UnitPrice, QuantityInStock)

VALUES

(1, 'Product A', 10, 100),

(2, 'Product B', 20, 200),

(3, 'Product C', 30, 300),

(4, 'Product D', 40, 400),

(5, 'Product E', 50, 500),

(6, 'Product F', 60, 600),

(7, 'Product G', 70, 700),

(8, 'Product H', 80, 800),

(9, 'Product I', 90, 900),

(10, 'Product J', 100, 1000);

CREATE TABLE OrderDetails (

OrderId INT,

ProductId INT,

Quantity INT,

CONSTRAINT PK\_OrderDetails PRIMARY KEY (OrderId, ProductId),

CONSTRAINT FK\_OrderDetails\_Orders FOREIGN KEY (OrderId) REFERENCES Orders(OrderId),

CONSTRAINT FK\_OrderDetails\_Products FOREIGN KEY (ProductId) REFERENCES Products(ProductId)

);

INSERT INTO OrderDetails (OrderId, ProductId, Quantity)

VALUES

(1, 1, 2),

(1, 2, 1),

(2, 3, 3),

(2, 4, 1),

(3, 5, 2),

(3, 6, 2),

(4, 7, 3),

(4, 8, 2),

(5, 9, 4),

(5, 10, 1);

**--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ SP \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

alter PROCEDURE sp\_ProcessOrders

@orderId INT,

@customerId INT,

@orderDate DATE,

@orderTotal MONEY

AS

BEGIN

SET NOCOUNT ON;

BEGIN TRANSACTION; -- Add this line to explicitly start a transaction

-- Check if order already exists

IF EXISTS(SELECT \* FROM Orders WHERE OrderId = @orderId)

BEGIN

RAISERROR('Order already exists', 16, 1);

ROLLBACK; -- Rollback the transaction if there's an error

RETURN;

END

-- Validate customer ID

IF NOT EXISTS(SELECT \* FROM Customers WHERE CustomerId = @customerId)

BEGIN

RAISERROR('Invalid customer ID', 16, 1);

ROLLBACK; -- Rollback the transaction if there's an error

RETURN;

END

-- Insert order into Orders table

INSERT INTO Orders(OrderId, CustomerId, OrderDate, OrderTotal)

VALUES(@orderId, @customerId, @orderDate, @orderTotal);

-- Update customer's total orders and order total in Customers table

UPDATE Customers

SET TotalOrders = TotalOrders + 1,

TotalOrderAmount = TotalOrderAmount + @orderTotal

WHERE CustomerId = @customerId;

-- Generate dynamic SQL to update inventory levels in Products table

DECLARE @sql NVARCHAR(MAX) = N'';

SELECT @sql = @sql + N'UPDATE Products SET QuantityInStock = QuantityInStock - ' + CAST(od.Quantity AS NVARCHAR) + ' WHERE ProductId = ' + CAST(od.ProductId AS NVARCHAR) + ';'

FROM OrderDetails od

WHERE od.OrderId = @orderId;

-- Execute dynamic SQL

EXEC sp\_executesql @sql;

COMMIT;

END

EXEC sp\_ProcessOrders

@orderId = 1001,

@customerId = 1,

@orderDate = '2023-03-01',

@orderTotal = 500.00;

**Create SP** to Delete All View , Tables

--\_\_\_\_\_\_\_\_\_\_\_\_\_ 1. **Without SP** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

select \* from INFORMATION\_SCHEMA.tables

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1. Delete all tables \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DECLARE @sql NVARCHAR(MAX) = ''

SELECT @sql = @sql + 'DROP TABLE ' + QUOTENAME(TABLE\_SCHEMA) + '.' + QUOTENAME(TABLE\_NAME) + ';'

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE = 'BASE TABLE'

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2. Delete all Views \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DECLARE @sql NVARCHAR(MAX) = ''

SELECT @sql = @sql + 'DROP VIEW ' + QUOTENAME(TABLE\_SCHEMA) + '.' + QUOTENAME(TABLE\_NAME) + ';'

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE = 'VIEW'

-- Execute the generated SQL statement

EXEC sp\_executesql @sql

--\_\_\_\_\_\_\_\_\_\_\_\_\_ 2. **With SP** \_\_\_\_\_\_\_\_\_\_\_\_\_

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1. SP \_ Delete all tables \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create procedure sp\_deleteAllTables

as

begin

DECLARE @sql NVARCHAR(MAX) = ''

SELECT @sql = @sql + 'DROP TABLE ' + QUOTENAME(TABLE\_SCHEMA) + '.' + QUOTENAME(TABLE\_NAME) + ';'

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE = 'BASE TABLE'

end

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2. Sp \_ Delete all Views \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create procedure sp\_deleteAllView

as

begin

DECLARE @sql NVARCHAR(MAX) = ''

SELECT @sql = @sql + 'DROP VIEW ' + QUOTENAME(TABLE\_SCHEMA) + '.' + QUOTENAME(TABLE\_NAME) + ';'

FROM INFORMATION\_SCHEMA.TABLES

WHERE TABLE\_TYPE = 'VIEW'

-- Execute the generated SQL statement

EXEC sp\_executesql @sql

end