A **Stored Procedure** in SQL Server is a set of precompiled SQL statements that can be executed as a single unit. Stored procedures improve performance, security, and reusability.

1. **Simple Stored Procedure Example: Get Employee Details**

This stored procedure retrieves employee details based on their **EmployeeID**.

**Table: Employees**

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name NVARCHAR(255),

Position NVARCHAR(255),

Salary DECIMAL(10,2)

);

**Stored Procedure: Get Employee Details by ID**

CREATE PROCEDURE GetEmployeeByID

@EmployeeID INT

AS

BEGIN

SET NOCOUNT ON;

--SET NOCOUNT ON; is used in **Stored Procedures**, **Triggers** to **suppress the "Rows Affected" message** that SQL Server returns after executing a query.

SELECT EmployeeID, Name, Position, Salary

FROM Employees

WHERE EmployeeID = @EmployeeID;

END;

**Executing the Stored Procedure**

EXEC GetEmployeeByID @EmployeeID = 1;

**2.Real-Life Scenario: Job Portal - Posting a Job**

let's create a **stored procedure for a Job Provider to post a job**.

**Scenario**

* A **Job Provider** can post a job if they are a registered user.
* The stored procedure will:
  1. **Check if the Job Provider exists** in the system.
  2. **Insert a new job posting** into the Jobs table.

**SQL Server Stored Procedure**

CREATE PROCEDURE PostJob

@JobProviderID INT,

@JobTitle NVARCHAR(255),

@Company NVARCHAR(255),

@Location NVARCHAR(255),

@Salary DECIMAL(10,2),

@JobDescription NVARCHAR(MAX),

@PostedDate DATE

AS

BEGIN

SET NOCOUNT ON;

-- Check if the Job Provider exists

IF EXISTS (SELECT 1 FROM JobProviders WHERE JobProviderID = @JobProviderID)

BEGIN

-- Insert job posting into the Jobs table

INSERT INTO Jobs (JobProviderID, JobTitle, Company, Location, Salary, JobDescription, PostedDate, Status)

VALUES (@JobProviderID, @JobTitle, @Company, @Location, @Salary, @JobDescription, @PostedDate, 'Open');

PRINT 'Job posted successfully.';

END

ELSE

BEGIN

PRINT 'Job Provider does not exist.';

END

END;

**Tables Used**

**JobProviders Table**

CREATE TABLE JobProviders (

JobProviderID INT PRIMARY KEY,

Name NVARCHAR(255),

Email NVARCHAR(255) UNIQUE

);

**Jobs Table**

CREATE TABLE Jobs (

JobID INT IDENTITY PRIMARY KEY,

JobProviderID INT FOREIGN KEY REFERENCES JobProviders(JobProviderID),

JobTitle NVARCHAR(255),

Company NVARCHAR(255),

Location NVARCHAR(255),

Salary DECIMAL(10,2),

JobDescription NVARCHAR(MAX),

PostedDate DATE,

Status NVARCHAR(50) -- ('Open', 'Closed')

);

**Executing the Stored Procedure**

To post a new job, call the procedure:

EXEC PostJob

@JobProviderID = 1,

@JobTitle = 'Software Engineer',

@Company = 'Tech Corp',

@Location = 'New York',

@Salary = 80000,

@JobDescription = 'Looking for an experienced software engineer with .NET expertise.',

@PostedDate = '2025-03-14';

**3.Real-Life Scenario: Library Management System**

Since you're familiar with ASP.NET Core and library systems, let's consider a scenario where a **librarian** wants to issue a book to a student.

**Scenario**

* A student can borrow a book if it's available.
* The stored procedure will:
  + Check if the book exists.
  + Verify if the book is available.
  + Insert a record into the IssuedBooks table.
  + Update the book count in the Books table.

**SQL Server Stored Procedure**

CREATE PROCEDURE IssueBook

@StudentID INT,

@BookID INT,

@IssuedDate DATE

AS

BEGIN

SET NOCOUNT ON;

-- Check if the book exists and is available

IF EXISTS (SELECT 1 FROM Books WHERE BookID = @BookID AND AvailableCopies > 0)

BEGIN

-- Insert into IssuedBooks table

INSERT INTO IssuedBooks (StudentID, BookID, IssuedDate, ReturnDate, Status)

VALUES (@StudentID, @BookID, @IssuedDate, NULL, 'Issued');

-- Update the Books table to decrease available copies

UPDATE Books

SET AvailableCopies = AvailableCopies - 1

WHERE BookID = @BookID;

PRINT 'Book issued successfully.';

END

ELSE

BEGIN

PRINT 'Book is not available.';

END

END;

**Tables Used**

**Books Table**

CREATE TABLE Books (

BookID INT PRIMARY KEY,

Title NVARCHAR(255),

Author NVARCHAR(255),

AvailableCopies INT

);

**IssuedBooks Table**

CREATE TABLE IssuedBooks (

IssueID INT IDENTITY PRIMARY KEY,

StudentID INT,

BookID INT,

IssuedDate DATE,

ReturnDate DATE NULL,

Status NVARCHAR(50)

);

**Executing the Stored Procedure**

To issue a book, call the stored procedure:

EXEC IssueBook @StudentID = 101, @BookID = 5, @IssuedDate = '2025-03-14';

**Benefits**

1. **Performance** – Precompiled execution improves efficiency.
2. **Security** – Users can execute procedures without direct table access.
3. **Code Reusability** – The same procedure can be used across multiple applications.
4. **Transaction Safety** – Can include error handling and rollback mechanisms.