**1. AFTER INSERT Trigger**

**Task:**

Create an AFTER INSERT trigger on the Employees table that logs inserted employee details into an Employee\_Audit table.

**Solution:**

-- Create Employee\_Audit Table

CREATE TABLE Employee\_Audit (

AuditID INT IDENTITY(1,1) PRIMARY KEY,

EmpID INT,

ActionType VARCHAR(50),

ActionDate DATETIME DEFAULT GETDATE()

);

-- Create Employees Table

CREATE TABLE Employees (

EmpID INT IDENTITY(1,1) PRIMARY KEY,

Name VARCHAR(100),

Position VARCHAR(50),

Salary DECIMAL(10,2)

);

-- Create Trigger

CREATE TRIGGER trg\_AfterInsert

ON Employees

AFTER INSERT

AS

BEGIN

INSERT INTO Employee\_Audit (EmpID, ActionType)

SELECT EmpID, 'INSERT' FROM inserted;

END;

**Test:**

INSERT INTO Employees (Name, Position, Salary)

VALUES ('John Doe', 'Manager', 60000);

SELECT \* FROM Employee\_Audit;

**2. AFTER UPDATE Trigger**

**Task:**

Create an AFTER UPDATE trigger to log salary updates.

**Solution:**

CREATE TRIGGER trg\_AfterUpdate

ON Employees

AFTER UPDATE

AS

BEGIN

INSERT INTO Employee\_Audit (EmpID, ActionType)

SELECT EmpID, 'UPDATE' FROM inserted;

END;

**Test:**

UPDATE Employees SET Salary = 65000 WHERE Name = 'John Doe';

SELECT \* FROM Employee\_Audit;

**3. AFTER DELETE Trigger**

**Task:**

Log deleted employee details.

**Solution:**

CREATE TRIGGER trg\_AfterDelete

ON Employees

AFTER DELETE

AS

BEGIN

INSERT INTO Employee\_Audit (EmpID, ActionType)

SELECT EmpID, 'DELETE' FROM deleted;

END;

**Test:**

DELETE FROM Employees WHERE Name = 'John Doe';

SELECT \* FROM Employee\_Audit;

**4. INSTEAD OF INSERT Trigger**

**Task:**

Prevent direct inserts into the Employees table and insert into PendingEmployees instead.

**Solution:**

-- Create PendingEmployees Table

CREATE TABLE PendingEmployees (

EmpID INT IDENTITY(1,1) PRIMARY KEY,

Name VARCHAR(100),

Position VARCHAR(50),

Salary DECIMAL(10,2)

);

-- Create Trigger

CREATE TRIGGER trg\_InsteadOfInsert

ON Employees

INSTEAD OF INSERT

AS

BEGIN

INSERT INTO PendingEmployees (Name, Position, Salary)

SELECT Name, Position, Salary FROM inserted;

END;

**Test:**

INSERT INTO Employees (Name, Position, Salary)

VALUES ('Jane Doe', 'Developer', 50000);

SELECT \* FROM PendingEmployees; -- Data is here instead

SELECT \* FROM Employees; -- No data inserted

**5. INSTEAD OF UPDATE Trigger**

**Task:**

Prevent salary updates and log an action.

**Solution:**

CREATE TRIGGER trg\_InsteadOfUpdate

ON Employees

INSTEAD OF UPDATE

AS

BEGIN

PRINT 'Salary updates are not allowed!';

END;

**Test:**

UPDATE Employees SET Salary = 70000 WHERE Name = 'Jane Doe';

-- Error message is displayed instead

**6. INSTEAD OF DELETE Trigger**

**Task:**

Prevent deletion and store deleted data in DeletedEmployees.

**Solution:**

-- Create DeletedEmployees Table

CREATE TABLE DeletedEmployees (

EmpID INT PRIMARY KEY,

Name VARCHAR(100),

Position VARCHAR(50),

Salary DECIMAL(10,2),

DeletedOn DATETIME DEFAULT GETDATE()

);

-- Create Trigger

CREATE TRIGGER trg\_InsteadOfDelete

ON Employees

INSTEAD OF DELETE

AS

BEGIN

INSERT INTO DeletedEmployees (EmpID, Name, Position, Salary)

SELECT EmpID, Name, Position, Salary FROM deleted;

END;

**Test:**

DELETE FROM Employees WHERE Name = 'Jane Doe';

SELECT \* FROM DeletedEmployees; -- Data is stored here instead

SELECT \* FROM Employees; -- Employee is still present

**7. DDL Trigger on CREATE TABLE**

**Task:**

Prevent the creation of new tables.

**Solution:**

CREATE TRIGGER trg\_PreventCreateTable

ON DATABASE

FOR CREATE\_TABLE

AS

BEGIN

PRINT 'Table creation is not allowed!';

ROLLBACK;

END;

**Test:**

CREATE TABLE TestTable (ID INT);

-- Error message is displayed instead

**8. DDL Trigger on ALTER TABLE**

**Task:**

Prevent schema modifications on the Employees table.

**Solution:**

CREATE TRIGGER trg\_PreventAlterTable

ON DATABASE

FOR ALTER\_TABLE

AS

BEGIN

PRINT 'Schema modifications are restricted!';

ROLLBACK;

END;

**Test:**

ALTER TABLE Employees ADD Address VARCHAR(255);

-- Error message is displayed instead

**9. DDL Trigger on DROP TABLE**

**Task:**

Prevent the dropping of any table.

**Solution:**

CREATE TRIGGER trg\_PreventDropTable

ON DATABASE

FOR DROP\_TABLE

AS

BEGIN

PRINT 'Table deletion is not allowed!';

ROLLBACK;

END;

**Test:**

DROP TABLE Employees;

-- Error message is displayed instead

**10. LOGON Trigger**

**Task:**

Restrict user login attempts outside office hours (9 AM - 6 PM).

**Solution:**

CREATE TRIGGER trg\_LimitLoginTime

ON ALL SERVER

FOR LOGON

AS

BEGIN

IF DATEPART(HOUR, GETDATE()) NOT BETWEEN 9 AND 18

BEGIN

PRINT 'Logins are only allowed between 9 AM and 6 PM.';

ROLLBACK;

END

END;

**Test:**

Try logging in outside of 9 AM - 6 PM.