

```
-- List down existing DB
```

```
EXEC sp_databases
```

```
SELECT name FROM sys.databases
```

```
-- Creating a DB
```

```
CREATE DATABASE school_db
```

```
CREATE DATABASE demo
```

```
-- Selecting a DB
```

```
USE school_db
```

```
SELECT DB_NAME()
```

```
-- Deleting a DB
```

```
DROP DATABASE demo
```

```
-- Creating a Table
```

```
CREATE TABLE students (
```

```
student_id INT,
```

```
name VARCHAR(100),
```

```
age INT,
```

```
grade INT
```

```
);
```

```
-- Checking existing tables in a DB
```

```
EXEC sp_help 'students'
```

```
-- Inserting Data
```

```
INSERT INTO students(student_id, name, age, grade)
```

```
VALUES (101, 'Raju', 10, 5);
```

```
INSERT INTO students(student_id, name, age, grade)
```

```
VALUES (102, 'Sham', 12, 7), (102, 'Baburao', 14, 9);
```

```
INSERT INTO students
```

```
VALUES (101, 'Paul', 11, 6);
```

```
INSERT INTO students(student_id,age, grade)
```

```
VALUES (105, 12, 7)
```

```
-- Reading data
```

```
SELECT * FROM students;
```

```
SELECT * FROM students WHERE student_id=102
```

```
SELECT name FROM students
```

```
SELECT * FROM students WHERE student_id=101
```

```
-- Updating data
```

```
UPDATE students
```

```
SET grade=12
```

```
WHERE student_id=103
```

```
-- Delete data
```

```
DELETE FROM students
```

WHERE student\_id=105

DELETE FROM students

WHERE 1=1

--- TRUNCATE ----

TRUNCATE table students

-- Task: Creating Employee Table ----

CREATE DATABASE bank\_db

USE bank\_db

CREATE TABLE employees (

emp\_id INT IDENTITY(101,1) PRIMARY KEY,

fname VARCHAR(50) NOT NULL,

lname VARCHAR(50) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

job\_title VARCHAR(50) NOT NULL,

department VARCHAR(50),

salary DECIMAL(10,2) DEFAULT 30000.00,

hire\_date DATE NOT NULL DEFAULT CONVERT(date, GETDATE()),

city VARCHAR(50)

);

EXEC sp\_help 'employees'

**INSERT INTO employees**

**(fname, lname, email, job\_title, department, salary, hire\_date, city)**

**VALUES**

**('Aarav', 'Sharma', 'aarav.sharma@example.com', 'Director', 'Management', 180000, '2019-02-10', 'Mumbai'),**

**('Diya', 'Patel', 'diya.patel@example.com', 'Lead Engineer', 'Tech', 120000, '2020-08-15', 'Bengaluru'),**

**('Rohan', 'Mehra', 'rohan.mehra@example.com', 'Software Engineer', 'Tech', 85000, '2022-05-20', 'Bengaluru'),**

**('Priya', 'Singh', 'priya.singh@example.com', 'HR Manager', 'Human Resources', 95000, '2019-11-05', 'Mumbai'),**

**('Arjun', 'Kumar', 'arjun.kumar@example.com', 'Data Scientist', 'Tech', 110000, '2021-07-12', 'Hyderabad'),**

**('Ananya', 'Gupta', 'ananya.gupta@example.com', 'Marketing Lead', 'Marketing', 90000, '2020-03-01', 'Delhi'),**

**('Vikram', 'Reddy', 'vikram.reddy@example.com', 'Sales Executive', 'Sales', 75000, '2023-01-30', 'Mumbai'),**

**('Sameera', 'Rao', 'sameera.rao@example.com', 'Software Engineer', 'Tech', 88000, '2023-06-25', 'Pune'),**

**('Ishaan', 'Verma', 'ishaan.verma@example.com', 'Recruiter', 'Human Resources', 65000, '2022-09-01', 'Mumbai'),**

**('Kavya', 'Joshi', 'kavya.joshi@example.com', 'Product Designer', 'Design', 92000, '2021-04-18', 'Bengaluru'),**

**('Zain', 'Khan', 'zain.khan@example.com', 'Sales Manager', 'Sales', 115000, '2019-09-14', 'Delhi'),**

**('Nisha', 'Desai', 'nisha.desai@example.com', 'Jr. Data Analyst', 'Tech', 70000, '2024-02-01', 'Hyderabad'),**

**('Aditya', 'Nair', 'aditya.nair@example.com', 'Marketing Analyst', 'Marketing', 68000, '2022-10-10', 'Delhi'),**

**('Fatima', 'Ali', 'fatima.ali@example.com', 'Sales Executive', 'Sales', 78000, '2022-11-22', 'Mumbai'),**

**('Kabir', 'Shah', 'kabir.shah@example.com', 'DevOps Engineer', 'Tech', 105000, '2020-12-01', 'Pune');**

**SELECT \* FROM employees**

```
INSERT INTO employees
```

```
(fname, lname, email, job_title, department, city)
```

```
VALUES
```

```
(null, 'Verma', 'null.verma@example.com', 'Director', 'Management', 'Mumbai')
```

```
select * from employees
```

```
-- WHERE Clause ---
```

```
SELECT * FROM employees WHERE emp_id=111
```

```
SELECT * FROM employees WHERE department != 'Sales'
```

```
SELECT * FROM employees WHERE salary = 100000
```

```
SELECT * FROM employees WHERE hire_date > '2020-12-31'
```

```
-- DISTINCT ----
```

```
SELECT DISTINCT city FROM employees
```

```
-- ORDER BY ---
```

```
SELECT * FROM employees ORDER BY salary DESC
```

```
SELECT * FROM employees ORDER BY hire_date
```

```
SELECT * FROM employees ORDER BY fname DESC
```

```
SELECT department, fname FROM employees ORDER BY department, fname
```

--- LIKE -----

**SELECT \* FROM employees WHERE department LIKE '%MAN%'**

**SELECT \* FROM employees WHERE fname LIKE '[ABCDE]%'**

**SELECT \* FROM employees WHERE fname LIKE '[^A]%'**

**SELECT \* FROM employees WHERE fname LIKE '\_a%'**

**SELECT \* FROM employees WHERE fname LIKE '\_\_\_\_'**

**SELECT \* FROM employees WHERE email LIKE '%gupta%'**

----- TOP -----

**SELECT TOP 3 \* FROM employees ORDER BY salary DESC**

--- Logical Operators ---

**SELECT \* FROM employees WHERE salary=75000 AND department='Sales'**

**SELECT \* FROM employees WHERE salary=75000 OR department='Sales' OR city='Mumbai'**

**SELECT \* FROM employees WHERE department NOT IN ('Tech', 'Sales', 'Management')**

**SELECT \* FROM employees WHERE salary BETWEEN 75000 AND 100000**

--- Aggregate Functions -----

**SELECT COUNT(emp\_id) FROM employees**

**SELECT MIN(salary) FROM employees**

**SELECT MAX(salary) FROM employees**

**SELECT AVG(salary) FROM employees**

**SELECT SUM(salary) FROM employees**

--- GROUP BY -----

```
SELECT department, COUNT(emp_id) as count FROM employees GROUP BY department
```

```
SELECT department, SUM(salary) as count FROM employees GROUP BY department
```

```
SELECT department, AVG(salary) as count FROM employees GROUP BY department
```

```
SELECT city, COUNT(emp_id) FROM employees GROUP BY city
```

```
SELECT department, city, COUNT(emp_id)
```

```
FROM employees GROUP BY department, city
```

```
ORDER BY department
```

--- HAVING Clause ---

```
SELECT department, COUNT(emp_id) as count
```

```
FROM employees
```

```
GROUP BY department HAVING COUNT(emp_id) > 2
```

```
SELECT job_title, AVG(salary) FROM employees GROUP BY job_title
```

```
HAVING AVG(salary) > 90000
```

```
SELECT department, SUM(salary) as total
```

```
FROM employees
```

```
GROUP BY department HAVING SUM(salary) > 200000
```

---- GROUP BY ROLLUP ----

```
SELECT department, COUNT(emp_id) as count FROM employees
```

```
GROUP BY ROLLUP(department)
```

```
SELECT department, SUM(salary) as count FROM employees
```

```
GROUP BY ROLLUP(department)
```

```
SELECT department, COALESCE(city,'Total') as city, COUNT(emp_id)
```

```
FROM employees GROUP BY ROLLUP(department, city)
```

```
ORDER BY department
```

--- SUB QUERIES ---

--- Single Row ----

```
SELECT * FROM employees
```

```
WHERE salary > (SELECT AVG(salary) FROM employees)
```

--- Multi Row ---

```
SELECT * FROM employees
```

```
WHERE department IN (
```

```
SELECT department FROM employees WHERE city='Mumbai'
```

```
)
```

--- Correlated ----

```
SELECT DISTINCT department FROM employees
```

```
SELECT MAX(salary) FROM employees WHERE department = 'Tech'
```

```
SELECT * FROM employees WHERE salary = 120000
```

```
SELECT * FROM employees e1
```

```
WHERE salary = (
    SELECT MAX(salary) FROM employees e2
    WHERE e2.department = e1.department
)
```

```
SELECT * FROM employees WHERE
salary IN (SELECT MAX(salary) FROM employees GROUP BY department)
```

--- INLINE VIEW ---

```
SELECT department, avg
FROM (
    SELECT department, AVG(salary) as avg FROM employees
    GROUP BY department
) AS dept_avg
WHERE avg>90000
```

----- Window Functions -----

```
SELECT fname, salary,
SUM(salary) OVER() as total_sal,
CAST(salary*100 / SUM(salary) OVER() AS DECIMAL(10,2)) AS pct
FROM employees
```

```
SELECT fname, department, salary,
SUM(salary) OVER(PARTITION BY department)
FROM employees
```

---- ROW\_NUMBER() ----

SELECT

```
ROW_NUMBER() OVER(ORDER BY fname) as row_num,  
fname, department, salary  
FROM employees
```

---- RANK() ----

```
SELECT fname, department, salary,  
DENSE_RANK() OVER(PARTITION BY department ORDER BY salary DESC)  
FROM employees
```

UPDATE employees

```
set salary=180000  
where fname='Alex'
```

----- LAG and LEAD -----

```
SELECT fname, department, salary,  
LEAD(salary) OVER(ORDER BY salary DESC)  
FROM employees
```

```
SELECT fname, hire_date, salary,  
LAG(salary) OVER(ORDER BY hire_date),  
salary - LAG(salary) OVER(ORDER BY hire_date) AS differ  
FROM employees
```

--- Running Total ---

```
SELECT fname, department, salary,  
SUM(salary) OVER(PARTITION BY department ORDER BY emp_id, salary DESC)  
FRoM employees
```

```
SELECT fname, department, salary,  
SUM(salary) OVER(  
    ORDER BY salary DESC  
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW  
) as running_total  
FRoM employees
```

---- 3 ROW/EMPLOYEES MOVING AVG -----

```
SELECT fname, hire_date, salary,  
CAST(AVG(salary) OVER(  
    ORDER BY hire_date  
    ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING  
) AS DECIMAL(10,2)) as three_row_avg  
FROM employees
```

---- FIRST\_VALUE, LAST\_VALUE, NTILE -----

```
SELECT fname, department,  
LAST_VALUE(fname) OVER(
```

**PARTITION BY department**

**ORDER by fname**

**ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING**

**)**

**FROM employees**

**SELECT fname, salary,**

**NTILE(5) OVER(ORDER BY salary DESC)**

**FROM employees**

**--- FIND TOP, MIDDLE, BOTTOM Earner in each Department ----**

**SELECT fname, department, salary,**

**NTILE(3) OVER(**

**PARTITION BY DEPARTMENT**

**ORDER BY salary DESC)**

**FROM employees**

**----- CTE -----**

**WITH avgsal AS (**

**SELECT department, AVG(salary) as dept\_avg FROM employees**

**GROUP BY department )**

```
SELECT e.fname, e.department, e.salary, a.dept_avg  
FROM employees e JOIN avgsal a  
ON e.department = a.department  
WHERE e.salary > a.dept_avg  
  
WITH maxsal AS (  
    SELECT department, MAX(salary) as dept_max FROM employees  
    GROUP BY department )  
  
SELECT e.fname, e.department, e.salary, m.dept_max  
FROM employees e JOIN maxsal m  
ON e.department = m.department  
WHERE e.salary = m.dept_max
```

----- STRING FUNCTIONS -----

```
SELECT fname, lname FROM employees  
-- CONCAT --  
SELECT CONCAT(fname, ' ', lname) as full_name FROM employees
```

**--- CONCAT\_WS ---**

-- One:Two:Three:Four

**SELECT CONCAT\_WS(',', emp\_id, lname, department) FROM employees**

**--- SUB STRING ---**

**SELECT SUBSTRING('Hey Buddy', 5, 9)**

**--- REPLACE ---**

**SELECT REPLACE('HEY BUDDY', 'HEY', 'HELLO')**

**SELECT REPLACE(department, 'Human Resources', 'HR') as dept FROM employees**

**--- REVERSE ---**

**SELECT REVERSE('HELLO')**

**--- LENGTH ---**

**SELECT LEN('HELLO WORLD')**

**SELECT LEN(email) as email\_length FROM employees**

**--- UPPER LOWER --**

**SELECT UPPER(fname) FROM employees**

**SELECT LOWER(fname) FROM employees**

**--- LEFT RIGHT -----**

**SELECT LEFT('ABCDPQRS', 3)**

**SELECT RIGHT('ABCDPQRS', 3)**

**--- TRIM -----**

**SELECT LEN(' Alright ')**

```
SELECT LEN(TRIM(' Alright '))
```

--- CHARINDEX ---

```
SELECT CHARINDEX('OM', 'THOMAS')
```

--- STRING FUNCTION EXERCISE ----

```
SELECT CONCAT_WS(':', emp_id, CONCAT(fname, ' ', lname), department)  
FROM employees
```

```
SELECT CONCAT_WS(':', emp_id, fname, UPPER(department))
```

```
FROM employees
```

```
SELECT CONCAT(LEFT(department,1),emp_id), fname FROM employees
```

---- ALTERING Table ----

```
SELECT * FROM employees
```

```
ALTER TABLE employees
```

```
ADD phone VARCHAR(15)
```

```
ALTER TABLE employees
```

```
DROP COLUMN phone
```

```
EXEC sp_help 'employees'
```

```
ALTER TABLE employees
```

```
ALTER COLUMN lname VARCHAR(100) NOT NULL
```

```
--- Changing column name ---
```

```
EXEC sp_rename
```

```
'employees.first_name', 'fname', 'COLUMN'
```

```
--- Changing table name ---
```

```
EXEC sp_rename
```

```
'staff', 'employees'
```

```
SELECT * FROM employees
```

```
--- Adding Constraint ----
```

```
EXEC sp_help 'employees'
```

```
ALTER TABLE employees
```

```
ADD CONSTRAINT default_dept DEFAULT 'Trainee'
```

```
FOR department
```

```
ALTER TABLE employees
```

```
ADD UNIQUE (department)
```

```
INSERT INTO employees
```

```
(fname, lname, email, job_title, city)
```

```
VALUES
```

```
('Paul', 'Philip', 'paul.philip@example.com', 'Fresher', 'Mumbai')
```

```
SELECT * FROM employees
```

--- CASE ----

```
SELECT fname, lname, salary,
```

```
CASE
```

```
WHEN salary > 100000 THEN 'High Earner'
```

```
WHEN salary BETWEEN 80000 AND 100000 THEN 'Medium Earner'
```

```
ELSE 'Standard Earner'
```

```
END as sal_cat
```

```
FROM employees
```

--- Calculate Bonus -----

```
SELECT fname, lname, department, salary,
```

```
CASE
```

```
WHEN department IN ('Sales', 'Marketing') THEN salary*0.10
```

```
WHEN department = 'Tech' THEN salary*0.12
```

```
ELSE salary*0.05
```

```
END as bonus
```

```
FROM employees
```

--- CHECK Constraint -----

```
INSERT INTO employees
```

```
(fname, lname, email, job_title, salary, city)
```

```
VALUES
```

```
('Alex', 'John', 'alex.john@example_com', 'Fresher', -10000, 'Mumbai')
```

```
SELECT * FROM employees
```

```
DELETE FROM employees
```

```
WHERE emp_id=121
```

```
ALTER TABLE employees
```

```
ADD CONSTRAINT chk_emp_positive_sal CHECK (salary>0)
```

```
ALTER TABLE employees
```

```
ADD CONSTRAINT chk_valid_email CHECK (email LIKE '%@%.%')
```

```
ALTER TABLE employees
```

```
DROP CONSTRAINT chk_emp_positive_sal
```

```
SELECT emp_id, fname, department, job_title, city from employees
```

```
----- 1:MANY DATA -----
```

```
CREATE DATABASE store_db
```

```
USE store_db
```

```
CREATE TABLE Customers (
```

```
customer_id INT IDENTITY(100,1) PRIMARY KEY,
```

```
customer_name VARCHAR(100) NOT NULL,
```

```
email VARCHAR(100) UNIQUE
```

```
);
```

```
CREATE TABLE Orders (
```

```
order_id INT IDENTITY(500,1) PRIMARY KEY,
```

```
order_date DATE NOT NULL,  
total_amount DECIMAL(10, 2),  
customer_id INT,  
FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)  
);
```

```
EXEC sp_help 'orders'
```

```
INSERT INTO Customers (customer_name, email)
```

```
VALUES
```

```
('Raju', 'raju@example.com'),  
('Sham', 'sham@example.com'),  
('Baburao', 'baburao@example.com');
```

```
INSERT INTO Orders (order_date, total_amount, customer_id)
```

```
VALUES
```

```
('2025-09-15', 1500.00, 100), -- This links to Raju (customer_id 100)  
('2025-09-28', 800.00, 101), -- This links to Sham (customer_id 101)  
('2025-10-05', 2200.00, 100), -- This links to Raju (customer_id 100)  
('2025-10-12', 500.00, 102), -- This links to Baburao (customer_id 102)  
('2025-10-17', 1200.00, 101); -- New order for Sham (customer_id 101)
```

```
SELECT * FROM Customers
```

```
SELECT * FROM Orders
```

```
INSERT INTO orders (order_date, total_amount)  
VALUES ('2025-10-18', '3500')
```

---- JOINS -----

--- CROSS JOIN ---

**SELECT \* FROM**

**customers CROSS JOIN orders**

--- INNER JOIN ---

**select \* from customers**

**select \* from orders**

**SELECT \* FROM**

**customers INNER JOIN orders**

**ON**

**customers.customer\_id = orders.customer\_id**

**SELECT c.customer\_name, COUNT(o.order\_id), SUM(o.total\_amount) FROM**

**customers c INNER JOIN orders o**

**ON**

**c.customer\_id = o.customer\_id**

**GROUP BY c.customer\_name**

---- LEFT/RIGHT JOIN -----

**SELECT \* FROM**

**customers RIGHT JOIN orders**

**ON**

```
customers.customer_id = orders.customer_id
```

```
SELECT c.customer_name, COUNT(o.order_id), SUM(o.total_amount) FROM  
customers c LEFT JOIN orders o  
ON  
c.customer_id = o.customer_id  
GROUP BY c.customer_name
```

--- OUTER JOIN ---

```
SELECT * FROM  
customers FULL OUTER JOIN orders  
ON  
customers.customer_id = orders.customer_id
```

--- OUTER APPLY -----

```
SELECT TOP 1 * FROM orders WHERE customer_id = 102 ORDER BY order_date DESC
```

```
SELECT * FROM customers
```

```
SELECT
```

```
c.customer_id, c.customer_name,  
o.order_id, o.order_date, o.total_amount  
FROM Customers AS c
```

```
CROSS APPLY (
```

```
SELECT TOP 1 *
FROM Orders AS o
WHERE o.customer_id = c.customer_id
ORDER BY o.order_date DESC
) AS o;
```

---- MANY TO MANY -----

```
CREATE DATABASE institute
```

```
USE institute
```

```
CREATE TABLE courses (
course_id INT IDENTITY(1,1) PRIMARY KEY,
course_name VARCHAR(100) NOT NULL,
course_fee NUMERIC(10, 2) NOT NULL
);
```

```
INSERT INTO courses (course_name, course_fee)
```

```
VALUES
```

```
('Mathematics', 500.00),
('Physics', 600.00),
('Chemistry', 700.00);
```

```
CREATE TABLE students (
student_id INT IDENTITY(1,1) PRIMARY KEY,
```

```
student_name VARCHAR(100) NOT NULL  
);
```

```
INSERT INTO Students (student_name) VALUES  
('Raju'),  
('Sham'),  
('Baburao'),  
('Alex');
```

```
CREATE TABLE enrollment (  
    enrollment_id INT IDENTITY(1,1) PRIMARY KEY,  
    student_id INT NOT NULL,  
    course_id INT NOT NULL,  
    enrollment_date DATE NOT NULL,  
  
    FOREIGN KEY (student_id) REFERENCES students(student_id),  
    FOREIGN KEY (course_id) REFERENCES courses(course_id)  
);
```

```
INSERT INTO enrollment (student_id, course_id, enrollment_date)  
VALUES  
(1, 1, '2025-01-01'), -- Raju enrolled in Mathematics  
(1, 2, '2025-01-15'), -- Raju enrolled in Physics  
(2, 1, '2025-02-01'), -- Sham enrolled in Mathematics  
(2, 3, '2025-02-15'), -- Sham enrolled in Chemistry  
(3, 3, '2025-03-25'); -- Alex enrolled in Chemistry
```

----

**SELECT \* FROM students**

**SELECT \* FROM courses**

**SELECT \* FROM enrollment**

**CREATE VIEW enrollment\_details AS**

**SELECT s.student\_name, c.course\_name, e.enrollment\_date, c.course\_fee**

**FROM enrollment e**

**INNER JOIN students s ON e.student\_id = s.student\_id**

**INNER JOIN courses c ON e.course\_id = c.course\_id**

**SELECT \* FROM enrollment\_details**

**SELECT TABLE\_SCHEMA, TABLE\_NAME**

**FROM INFORMATION\_SCHEMA.VIEWS**

**sp\_helptext 'enrollment\_details'**

**SELECT c.course\_name, COUNT(s.student\_id), SUM(c.course\_fee)**

**FROM enrollment e**

**INNER JOIN students s ON e.student\_id = s.student\_id**

**INNER JOIN courses c ON e.course\_id = c.course\_id**

**GROUP BY c.course\_name**

**----- STORED PROCEDURE -----**

```
CREATE PROCEDURE get_employees_sp  
AS  
BEGIN  
    SELECT emp_id, fname, lname, department, hire_date, city  
    FROM employees  
END  
  
  
EXEC get_employees_sp
```

```
CREATE PROCEDURE get_emp_by_dept_sp  
    @p_department VARCHAR(100)  
AS  
BEGIN  
    SELECT emp_id, fname, lname, department, hire_date, city  
    FROM employees  
    WHERE department = @p_department  
END  
  
  
EXEC get_emp_by_dept_sp 'Sales'
```

----- HOW TO CHECK Existing SP -----

```
SELECT ROUTINE_NAME  
FROM INFORMATION_SCHEMA.ROUTINES  
WHERE ROUTINE_TYPE = 'PROCEDURE'
```

```
sp_helptext 'get_employees_sp'
```

```
ALTER PROCEDURE get_employees_sp
```

```
AS
```

```
BEGIN
```

```
SELECT emp_id, fname, lname, department, job_title, hire_date, city
```

```
FROM employees
```

```
END
```

```
EXEC get_employees_sp
```

```
---
```

```
CREATE PROCEDURE update_emp_salary
```

```
@p_employee_id INT,
```

```
@p_new_salary NUMERIC(10, 2)
```

```
AS
```

```
BEGIN
```

```
UPDATE employees
```

```
SET salary = @p_new_salary
```

```
WHERE emp_id = @p_employee_id;
```

```
END;
```

```
SELECT * FROM employees
```

```
EXEC update_emp_salary 103, 90000
```

```
CREATE PROCEDURE add_employee
    @p_fname VARCHAR(50),
    @p_lname VARCHAR(50),
    @p_email VARCHAR(100),
    @p_job_title VARCHAR(50),
    @p_department VARCHAR(50),
    @p_salary NUMERIC(10, 2),
    @p_city VARCHAR(50)

AS
BEGIN
    INSERT INTO employees (fname, lname, email, job_title, department, salary, city)
    VALUES (@p_fname, @p_lname, @p_email, @p_job_title, @p_department,
            @p_salary, @p_city);

END;

EXEC add_employee 'Sundar', 'Paul', 'sundar.paul@email.com', 'Trainee', 'Tech',
30000, 'Bhopal'
```

----- SP with OUTPUT -----

```
CREATE PROCEDURE get_emp_dept_avg
    @p_dept VARCHAR(100),
    @dept_avg NUMERIC(10,2) OUTPUT
AS
BEGIN
    SELECT
        @dept_avg = AVG(salary) FROM employees
```

```
WHERE department = @p_dept
```

```
END
```

```
-----  
DECLARE @AvgDeptResult NUMERIC(10,2)
```

```
EXEC get_emp_dept_avg 'Sales', @AvgDeptResult OUTPUT
```

```
SELECT @AvgDeptResult
```

```
----- PROCEDURAL LOGIC -----
```

```
ALTER PROCEDURE update_emp_salary_safely_sp
```

```
    @p_employee_id INT,
```

```
    @p_new_salary NUMERIC(10, 2),
```

```
    @p_message VARCHAR(200) OUTPUT
```

```
AS
```

```
BEGIN
```

```
--- Checking if employee exists -----
```

```
IF NOT EXISTS (SELECT 1 FROM employees WHERE emp_id = @p_employee_id)
```

```
BEGIN
```

```
    SET @p_message = 'ERROR: EMP ID does not exists.';
```

```
    RETURN;
```

```
END
```

```
--- GETTING CURRENT Salary -----
```

```
DECLARE @current_sal NUMERIC(10,2);
```

```
SELECT @current_sal=salary from employees
```

```
where emp_id=@p_employee_id;
```

```
--- Comparing Salaries -----
```

```
IF @p_new_salary > @current_sal
BEGIN
    UPDATE employees
    SET salary = @p_new_salary
    WHERE emp_id = @p_employee_id;

    SET @p_message = 'Success: Salary updated!'

END
ELSE
BEGIN
    SET @p_message = 'ERROR: New salary should be greater than current'
END
END;
```

--- User Defined Function -----

--- Triggers ---

```
CREATE TRIGGER trg_AuditSalaryChange
ON Employees
AFTER UPDATE
AS
BEGIN
    IF UPDATE(Salary)
        BEGIN
            INSERT INTO SalaryAudit (EmpID, OldSalary, NewSalary)
            SELECT
                d.EmpID,
```

```
d.Salary AS OldSalary,  
i.Salary AS NewSalary  
FROM deleted d  
INNER JOIN inserted i ON d.EmpID = i.EmpID;  
END  
END;
```

--- Trigger USECASE of BEFORE/INSTEAD OF -----

```
CREATE TRIGGER trg_PreventManagementDeletion  
ON Employees  
INSTEAD OF DELETE  
AS  
BEGIN  
-- Prevent deletion of Management employees
```

```
IF EXISTS (SELECT 1 FROM deleted WHERE Department = 'Management')
```

```
BEGIN
```

```
RAISERROR('Deletion not allowed for Management employees.', 16, 1);
```

```
ROLLBACK TRANSACTION;
```

```
RETURN;
```

```
END;
```

```
-- Allow deletion for others
```

```
DELETE FROM Employees
```

```
WHERE EmpID IN (SELECT EmpID FROM deleted);
```

```
END;
```

--- Generating random data -----

```
CREATE TABLE Employees (
    EmployeeID INT IDENTITY PRIMARY KEY,
    FirstName NVARCHAR(50),
    LastName NVARCHAR(50),
    Department NVARCHAR(50),
    Salary INT
);
```

```
INSERT INTO Employees (FirstName, LastName, Department, Salary)
SELECT TOP (500000)
    LEFT(NEWID(), 8), -- random first name
    LEFT(NEWID(), 8), -- random last name
    CASE ABS(CHECKSUM(NEWID())) % 5
        WHEN 0 THEN 'IT'
        WHEN 1 THEN 'HR'
        WHEN 2 THEN 'Finance'
        WHEN 3 THEN 'Marketing'
        ELSE 'Sales'
    END,
    ABS(CHECKSUM(NEWID())) % 100000 + 30000
FROM sys.objects a
CROSS JOIN sys.objects b;
```