**Employee Department Case Study**

Create Database EmpDeptCaseStudy;

Use EmpDeptCaseStudy;

Create Table Employees(

EmployeeID INT Primary Key,

Firstname VARCHAR(20),

Lastname VARCHAR(20),

DepartmentID INT);

Create Table Departments(

DepartmentID INT Primary Key,

DeptName VARCHAR(20));

Alter Table Employees

Add Constraint FK\_Department Foreign Key(DepartmentID) References Departments(DepartmentID);

Insert into Departments Values(1001, 'Life Science');

Insert into Departments Values(1002, 'Engineering');

Insert into Departments Values(1003, 'Automation');

Insert into Departments Values(1004, 'Production');

Insert into Departments Values(1005, 'IT Support');

Insert into Departments Values(1006, 'Client Support');

Insert Into Employees Values(1, 'Sneha', 'Kasat', 1001);

Insert Into Employees Values(2, 'Rakshit', 'Patel', 1004);

Insert Into Employees Values(3, 'Tammana', 'Gupta', 1003);

Insert Into Employees Values(4, 'Nayanika', 'Ghosh', 1002);

Insert Into Employees Values(5, 'Gaurav', 'Kasat', 1001);

Insert Into Employees Values(6, 'Janhavi', 'Yende', 1005);

Insert Into Employees Values(7, 'Ramesh', 'Desai', 1006);

Alter Table Employees Add Salary Money;

Update Employees Set Salary = 32000 where EmployeeId = 1;

Update Employees Set Salary = 82000 where EmployeeId = 2;

Update Employees Set Salary = 12000 where EmployeeId = 3;

Update Employees Set Salary = 22000 where EmployeeId = 4;

Update Employees Set Salary = 120000 where EmployeeId = 5;

Update Employees Set Salary = 100000 where EmployeeId = 6;

Update Employees Set Salary = 32000 where EmployeeId = 7;

Select \* from Departments;

A screenshot of a computer

Description automatically generated

Select \* from Employees;

A table of names and numbers

Description automatically generated

A screenshot of a computer

Description automatically generated

**--Arithmatic Operators**

Select Firstname, Lastname, Cast(Salary as numeric) + 500 as New\_Salary from Employees;

A table of numbers and letters

Description automatically generated with medium confidence

**--Logical Operators**

Select EmployeeID, Firstname+ ' ' +Lastname as Employee\_Name from Employees

where Salary > 30000 and DepartmentID = 1001;

A screenshot of a computer

Description automatically generated

**--Comparison Operator**

Select EmployeeID, Firstname+ ' ' +Lastname as Employee\_Name from Employees

where Salary > 30000;

A screenshot of a computer

Description automatically generated

**--Date Functions**

Select GETDATE() As CurrentDate;

A date and time on a white background

Description automatically generated

Select DATEADD(Day, 7, GETDATE()) As NextWeek;

A screenshot of a computer

Description automatically generated

Select DATEDIFF(Day, '2024-01-01', GETDATE()) As DaySinceStartOfYear;



**--Math Functions**

Select ABS(-10) As AbsoluteValue;



Select ROUND(123.4567, 2) As RoundedValue;

A screenshot of a computer

Description automatically generated

Select SQRT(9) As SquareRoot;

A screenshot of a computer

Description automatically generated

**--String Functions**

Select UPPER('hello') As UpperCase;

Select LOWER('HEllO') As LowerCase;

Select SUBSTRING('Hello, World!', 2, 5) As SubStringValue;

A screenshot of a computer

Description automatically generated

**--Merge - It allowes you to perform insert, update or delete operations in single statement.**

MERGE INTO Employees AS Target

USING (SELECT EmployeeID, Firstname, Lastname, DepartmentID FROM Employees) AS Source

ON Target.EmployeeID = Source.EmployeeID

WHEN MATCHED THEN

UPDATE SET Firstname = Source.Firstname,

Lastname = Source.Lastname,

DepartmentID = Source.DepartmentID

WHEN NOT MATCHED BY TARGET THEN

INSERT (EmployeeID, Firstname, Lastname, DepartmentID)

VALUES (Source.EmployeeID, Source.Firstname, Source.Lastname, Source.DepartmentID)

WHEN NOT MATCHED BY Source THEN

DELETE;

**--Group by**

Select DepartmentID, Count(\*) As EmployeeCount from Employees

Group By DepartmentID;

A screenshot of a computer

Description automatically generated

**--Grouping Sets- allowes you to define multiple grouping in the same query.**

Select DepartmentID, Count(\*) As EmployeeCount From Employees

Group By GROUPING Sets(

(DepartmentID),());

A screenshot of a computer

Description automatically generated

**--Adding Indexes - Used to speed up retrival of data**

Create Index idx\_LastName

On Employees(Lastname);

**--View - Views are virtual tables that are based on result set of SQL Query**

Create View EmpDeptView as

Select e.DepartmentID, e.Firstname, e.Lastname, d.DeptName

from Employees e

Join Departments d

On e.DepartmentID = d.DepartmentID;

Select \* from EmpDeptView;

A screenshot of a computer

Description automatically generated

**--Transactions**

Begin Transaction;

Begin Try

--Insert new Employee

Insert into Employees Values(8, 'Avinash', 'Dabhade', 1003, 60000);

--Update Existing Employee

Update Employees Set Lastname = 'Bhatiya' where EmployeeID = 3;

Commit Transaction;

End Try

Begin Catch

--Rollback the transaction in case of Error

Rollback Transaction;

--Handle The Error

Declare @ErrorMessage NVarchar(4000) = Error\_Message();

Print @ErrorMessage;

End Catch;

Select \* from Employees;

**--Stored Procedure**

Create Procedure InsertEmp

@EmpId INT,

@Firstname varchar(20),

@Lastname varchar(20),

@DeptID INT,

@Salary Money

AS

Begin

Insert into Employees Values(@EmpId, @Firstname, @Lastname, @DeptID, @Salary);

End

Execute InsertEmp

@EmpId = 9, @Firstname = 'Shubham', @Lastname = 'Bamne', @DeptID = 1004, @Salary = 7000;

**--Triggers**

Create Table EmployeeChanges(

ChangeId INT IDENTITY(1,1) Primary Key,

EmployeeID INT,

ChangeType nVarchar(50),

CahngeDate DateTime Default GetDate()

);

Create Trigger trg\_EmployeeChanges

ON Employees

AFTER Insert, Update, Delete

As

Begin

Declare @EmployeeID INT, @ChangeType nVarchar(50);

If Exists(Select \* from inserted)

Begin

Set @EmployeeID = (Select EmployeeId from inserted);

Set @ChangeType = 'Insert';

Insert Into EmployeeChanges(EmployeeID, ChangeType)

Values(@EmployeeID, @ChangeType);

End

If Exists(Select \* from deleted)

Begin

Set @EmployeeID = (Select EmployeeId from deleted);

Set @ChangeType = 'Delete';

Insert Into EmployeeChanges(EmployeeID, ChangeType)

Values(@EmployeeID, @ChangeType);

End

End;

INSERT INTO Employees (EmployeeID, Firstname, Lastname, DepartmentID, Salary)

VALUES (10, 'John', 'Doe', 1005, 5000);

Select \* from EmployeeChanges;

A screenshot of a computer

Description automatically generated

**--Get Number of employees**

Select Count(\*) As NumberOfEmployees from Employees;

A close-up of a white rectangular object

Description automatically generated

**--Get Number of Employees in different Departments**

Select d.DeptName , Count(\*) as NumberOfEmployees

From Employees e

Inner Join Departments d

On e.DepartmentID = d.DepartmentID

Group By d.DeptName;

A screenshot of a computer

Description automatically generated

**--Retrieve all employees’ first names and their corresponding department names.**

Select e.Firstname As Emp\_Name, d.DeptName As Department

from Employees e

Inner Join Departments d

On e.DepartmentID = d.DepartmentID;

A screenshot of a computer

Description automatically generated

**--Find the total number of employees in each department.**

Select d.DeptName, Count(\*) as No\_of\_Empployees

From Departments d

Join Employees e On d.DepartmentID = e.DepartmentID

Group by d.DeptName;

A screenshot of a computer

Description automatically generated

**--List all employees who have a salary greater than 50,000.**

Select Concat(Firstname, ' ' , Lastname) as Emp\_Name

from Employees

Where Salary>50000;

A screenshot of a computer

Description automatically generated

**--Calculate the average salary of employees in each department.**

Select d.DeptName, Avg(e.Salary) As Avg\_Salary

From Departments d

Join Employees e

On d.DepartmentID = e.DepartmentID

Group by d.DeptName;

A screenshot of a computer

Description automatically generated

**--Retrieve the details of employees who do not belong to the ‘IT Support’ department.**

Select e.\*, d.DeptName from Employees e

join Departments d On e.DepartmentID = d.DepartmentID

where DeptName != 'IT Support';

A table with numbers and letters

Description automatically generated

**--Find the highest salary in the ‘Engineering’ department.**

Select Max(Salary) from Employees e

join Departments d On e.DepartmentID = d.DepartmentID

Where d.DeptName = 'Engineering';



**--Create a stored procedure that takes a department ID as input and returns all employees in that department.**

Create Procedure RetriveEmp

@DeptId INT

AS

Begin

Select \* from Employees Where DepartmentID = @DeptId;

End

Execute RetriveEmp @DeptId = 1004

A screenshot of a computer

Description automatically generated

**--Write a trigger that automatically updates an employee’s salary to 10% more than their current salary whenever a new employee is added to the same department.**

CREATE TRIGGER UpdateSalaryOnNewEmployee

AFTER INSERT ON Employees

FOR EACH ROW

BEGIN

UPDATE Employees

SET Salary = Salary \* 1.10

WHERE DepartmentID = NEW.DepartmentID

AND EmployeeID != NEW.EmployeeID;

END;

**--Create a function that calculates the total salary expenditure for a given department.**

CREATE FUNCTION TotalSalaryExpenditure(@DeptID INT)

RETURNS MONEY

AS

BEGIN

DECLARE @TotalExpenditure MONEY;

SELECT @TotalExpenditure = SUM(Salary)

FROM Employees

WHERE DepartmentID = @DeptID;

RETURN @TotalExpenditure; END;

**--Write a query to find the department with the highest average salary.**

SELECT Top 1 DepartmentID, AVG(Salary) AS AverageSalary

FROM Employees

GROUP BY DepartmentID

ORDER BY AverageSalary DESC;

A screenshot of a computer

Description automatically generated

**--Create a view that shows the employee ID, full name, department name, and salary.**

Create View EmpDetails As

Select e.EmployeeID, CONCAT(e.Firstname, ' ', e.Lastname) As EmpName, d.DeptName, e.Salary

From Employees e

inner Join Departments d

On e.DepartmentID = d.DepartmentID;

Select \* from EmpDetails;

A screenshot of a computer

Description automatically generated

**--Generate a report that lists departments with more than or equal to 2 employees and the total salary expenditure for those departments**

SELECT

d.DepartmentID,

d.DeptName,

COUNT(e.EmployeeID) AS NumberOfEmployees,

SUM(e.Salary) AS TotalSalaryExpenditure

FROM

Employees e

JOIN

Departments d ON e.DepartmentID = d.DepartmentID

GROUP BY

d.DepartmentID, d.DeptName

HAVING

COUNT(e.EmployeeID) >= 2;

A screenshot of a computer

Description automatically generated