Manipulating Data by Using DML Statements Employee Database

Creating database

```
CREATE DATABASE EmployeeDB; USE EmployeeDB;
```

Creating tables

```
CREATE TABLE Employees (
EmployeeID INT PRIMARY KEY IDENTITY(1,1), FirstName VARCHAR(50),
LastName VARCHAR(50), Department VARCHAR(50), DateOfBirth DATE,
Salary DECIMAL(10, 2), JoiningDate DATE
);

CREATE TABLE Departments (
DepartmentID INT PRIMARY KEY IDENTITY(1,1), DepartmentName VARCHAR(50) UNIQUE
);

CREATE TABLE Projects (
ProjectID INT PRIMARY KEY IDENTITY(1,1), ProjectName VARCHAR(100),
Budget DECIMAL(12, 2), DepartmentID INT,
FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID)
);
```

Storing Data in a Table

```
INSERT INTO Employees (FirstName, LastName, Department, DateOfBirth, Salary, JoiningDate) VALUES ('John', 'Doe', 'HR', '1985-05-20', 50000, '2020-01-15'), ('Jane', 'Smith', 'IT', '1990-08-15', 60000, '2019-06-01'), ('Alice', 'Johnson', 'Finance', '1988-12-25', 75000, '2018-07-10'), ('Bob', 'Brown', 'Marketing', '1985-03-05', 55000, '2021-01-20'), ('Eve', 'Davis', 'Sales', '1992-11-11', 52000, '2022-03-17'), ('Charlie', 'Wilson', 'Operations', '1995-04-10', 48000, '2023-05-25'); INSERT INTO Departments (DepartmentName) VALUES ('Marketing'), ('Sales'), ('Operations'), ('Customer Service'), ('Research and Development'), ('IT'), ('Finance'), ('HR'); INSERT INTO Projects (ProjectName, Budget, DepartmentID) VALUES ('Alpha Project', 100000, 1), ('Beta Project', 200000, 2), ('Gamma Project', 150000, 4), ('Delta Project', 250000, 5), ('Epsilon Project', 300000, 6), ('Zeta Project', 180000, 7), ('Theta Project', 500000, 8);
```

Updating Data in a Table

UPDATE Employees SET Salary = Salary + 5000 WHERE EmployeeID = 1;

Deleting Data from a Table

DELETE FROM Employees WHERE EmployeeID = 2;

Retrieving Specific Attributes

SELECT FirstName, LastName FROM Employees;



Retrieving Selected Rows

SELECT * FROM Employees WHERE Department = 'Sales';



Filtering Data:WHERE Clauses

SELECT * FROM Employees WHERE Salary > 60000;



Filtering Data:IN,DISTINCT,AND,OR,BETWEEN,LIKE,Column & table aliases

IN

SELECT * FROM Employees WHERE Department IN ('IT', 'HR');



DATA ENGINEERING MS SQL

DISTINCT

SELECT DISTINCT Department FROM Employees;



AND

SELECT * FROM Employees WHERE Department = 'Finance' AND Salary > 70000;



OR

SELECT * FROM Employees WHERE Department = 'Finance' OR Salary > 70000;



BETWEEN

SELECT * FROM Employees WHERE Salary BETWEEN 50000 AND 100000;



LIKE

SELECT * FROM Employees WHERE FirstName LIKE 'J%';



Implementing Data Integrity

NOT NULL

ALTER TABLE Employees ALTER COLUMN Salary DECIMAL(10, 2) NOT NULL;

UNIQUE

ALTER TABLE Departments ADD CONSTRAINT UQ DepartmentName UNIQUE (DepartmentName);

FOREIGN KEY

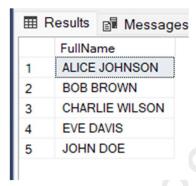
ALTER TABLE Projects ADD CONSTRAINT FK_Department FOREIGN KEY (DepartmentID) REFERENCES Departments(DepartmentID);

Using Functions to Customize the Result Set

Using String Functions

Combine the first and last names of employees and make them uppercase.

SELECT UPPER(FirstName + ' ' + LastName) AS FullName FROM Employees;



Using Date Functions

Calculate age

SELECT FirstName, LastName, YEAR(JoiningDate) AS JoiningYear, DATEDIFF(YEAR, DateOfBirth, GETDATE()) AS Age FROM Employees;



Using Mathematical Functions

Round salary to the nearest thousand and increase it by 10%.

SELECT FirstName, LastName, ROUND(Salary * 1.1, -3) AS AdjustedSalary FROM Employees;



Using System Functions

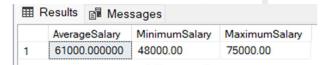
Retrieve the current date and user for each query

SELECT SYSTEM USER AS UserName, GETDATE() AS CurrentDate;



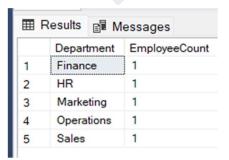
Summarizing Data by Using Aggregate Functions

SELECT AVG(Salary) AS AverageSalary, MIN(Salary) AS MinimumSalary, MAX(Salary) AS MaximumSalary FROM Employees;



Grouping Data

SELECT Department, COUNT(*) AS EmployeeCount FROM Employees GROUP BY Department;



Hands on Exercise: Filtering Data using SQL Queries

Filter Employees with Salary above 50,000 and in the Sales department

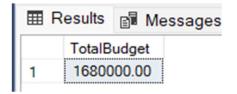
SELECT * FROM Employees WHERE Salary > 50000 AND Department = 'Sales';



Hands on Exercise: Total Aggregations using SQL Queries

Calculate Total Budget for All Projects

SELECT SUM(Budget) AS TotalBudget FROM Projects;



Hands on Exercise: Group By Aggregations using SQL Queries

Calculate average salary of each department

SELECT Department, AVG(Salary) AS AvgSalary FROM Employees GROUP BY Department;



Hands on Exercise: Rules and Restrictions to Group and Filter Data in SQL queries

Rule: Columns in the SELECT clause must either be part of the GROUP BY clause or used with an aggregate function.

SELECT Department, AVG(Salary) AS AvgSalary FROM Employees GROUP BY Department;

[For Result refer previous image]

Hands on Exercise: Filter Data based on Aggregated Results using Group By and Having

Filter Departments with Average Salary Above 60,000

SELECT Department, AVG(Salary) AS AvgSalary FROM Employees GROUP BY Department HAVING AVG(Salary) > 60000;

