

**1. An Enterprise wishes to maintain a database to automate its operations. Enterprise is divided into certain departments and each department consists of employees.**

- a. **Update the employee salary by 15%, whose experience is greater than 10 years**
- a. **Delete the employees, who completed 30 years of service**
- b. **Display the manager who is having maximum number of employees working under him**
- c. **Create a view, which contain employee names and their manager**

```
CREATE TABLE Departments ( DeptID INT PRIMARY KEY, DeptName  
VARCHAR(100));
```

```
CREATE TABLE Employees ( EmpID INT PRIMARY KEY, Name VARCHAR(100),  
Experience INT, Salary DECIMAL(10,2 ), ManagerID INT, DeptID INT, FOREIGN KEY  
(ManagerID) REFERENCES Employees(EmpID), FOREIGN KEY (DeptID)  
REFERENCES Departments(DeptID));
```

#### **Insert into Departments table**

```
INSERT INTO Departments (DeptID, DeptName) VALUES (1, 'HR'), (2, 'Finance'), (3, 'IT');
```

#### **Insert into Employees table**

```
INSERT INTO Employees (EmpID, Name, Experience, Salary, ManagerID, DeptID)
```

```
VALUES (1, 'Alice', 12, 60000.00, NULL, 1), // Alice is the manager of HR
```

```
(2, 'Bob', 8, 50000.00, 1, 1), (3, 'Charlie', 15, 70000.00, 1, 1),
```

```
(4, 'David', 30, 80000.00, NULL, 2), // David is the manager of Finance
```

```
(5, 'Eve', 5, 45000.00, 4, 2), (6, 'Frank', 10, 50000.00, 4, 2),
```

```
(7, 'Grace', 20, 85000.00, NULL, 3), // Grace is the manager of IT
```

```
(8, 'Claret', 11, 75000.00, 7, 3),
```

```
(9, 'Hank', 25, 90000.00, 7, 3), (10, 'Ivy', 2, 40000.00, 7, 3);
```

```
// Bob, Charlie and Claret report to Alice in HR.
```

```
Eve and Frank report to David in Finance.
```

```
Hank and Ivy report to Grace in IT.
```

#### **a) Update the employee salary by 15%, whose experience is greater than 10 years**

```
UPDATE Employees SET Salary = Salary * 1.15 WHERE Experience > 10;
```

**b) Delete the employees, who completed 30 years of service**

```
DELETE FROM Employees WHERE Experience >= 30;
```

**c) Display the manager who has the maximum number of employees working under him.**

- 
- 
- To display Manager ID and maximum number of employees working under him

```
Select ManagerID, count(*) AS NumEmployees from Employees
```

```
where ManagerID IS NOT NULL group by ManagerID
```

```
order by NumEmployees DESC limit 1;
```

- To display Manager name and maximum number of employees working under him

```
SELECT m.Name AS ManagerName, COUNT(e.EmpID) AS NumEmployees FROM Employees e JOIN Employees m ON e.ManagerID = m.EmpID WHERE e.ManagerID IS NOT NULL GROUP BY e.ManagerID, m.Name ORDER BY NumEmployees DESC LIMIT 1;
```

**d) Create a view, which contain employee names and their manager**

```
CREATE VIEW EmpManagerView AS
```

```
SELECT E.Name AS EmployeeName, M.Name AS ManagerName
```

```
FROM Employees E LEFT JOIN Employees M ON E.ManagerID = M.EmpID;
```

```
>SELECT * FROM EmpManagerView;
```

**2. Write an SQL query to find the total revenue, average revenue, minimum revenue, maximum revenue, and number of orders for each product in an e-commerce database.**

The database contains two tables: Orders and Order Details. The Orders table has the columns Order ID, Order Date, and Customer ID, and the Order Details table has the columns Order Detail ID, Order ID, Product ID, Quantity, and Unit Price.

**Create Orders table**

```
CREATE TABLE Orders ( Order_ID INT PRIMARY KEY, Order_Date DATE NOT NULL,  
Customer_ID INT NOT NULL);
```

**Create Order\_Details table**

```
CREATE TABLE Order_Details ( Order_Detail_ID INT PRIMARY KEY, Order_ID INT,  
Product_ID INT NOT NULL, Quantity INT NOT NULL CHECK (Quantity > 0),
```

Unit\_Price DECIMAL(10,2) NOT NULL CHECK (Unit\_Price >= 0), FOREIGN KEY (Order\_ID) REFERENCES Orders(Order\_ID) ON DELETE CASCADE);

**a. Find the total revenue.**

```
select sum(Quantity * Unit_Price) AS Total_Revenue from Order_Details;
```

**b. Find the average revenue.**

```
select avg(Quantity * Unit_Price) AS Average_Revenue from Order_Details;
```

**c. Find the minimum and maximum revenue.**

```
select min(Quantity * Unit_Price) AS Min_Revenue, MAX(Quantity * Unit_Price) AS Max_Revenue from Order_Details;
```

**d. Find the number of orders for each product.**

```
select Product_ID, count(distinct Order_ID) AS Number_of_Orders  
from Order_Details group by Product_ID;
```