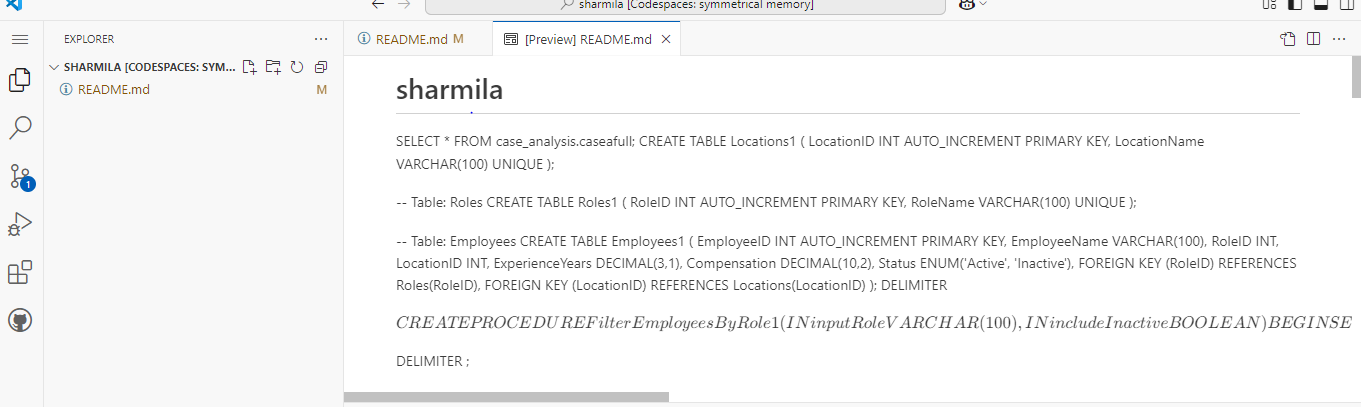
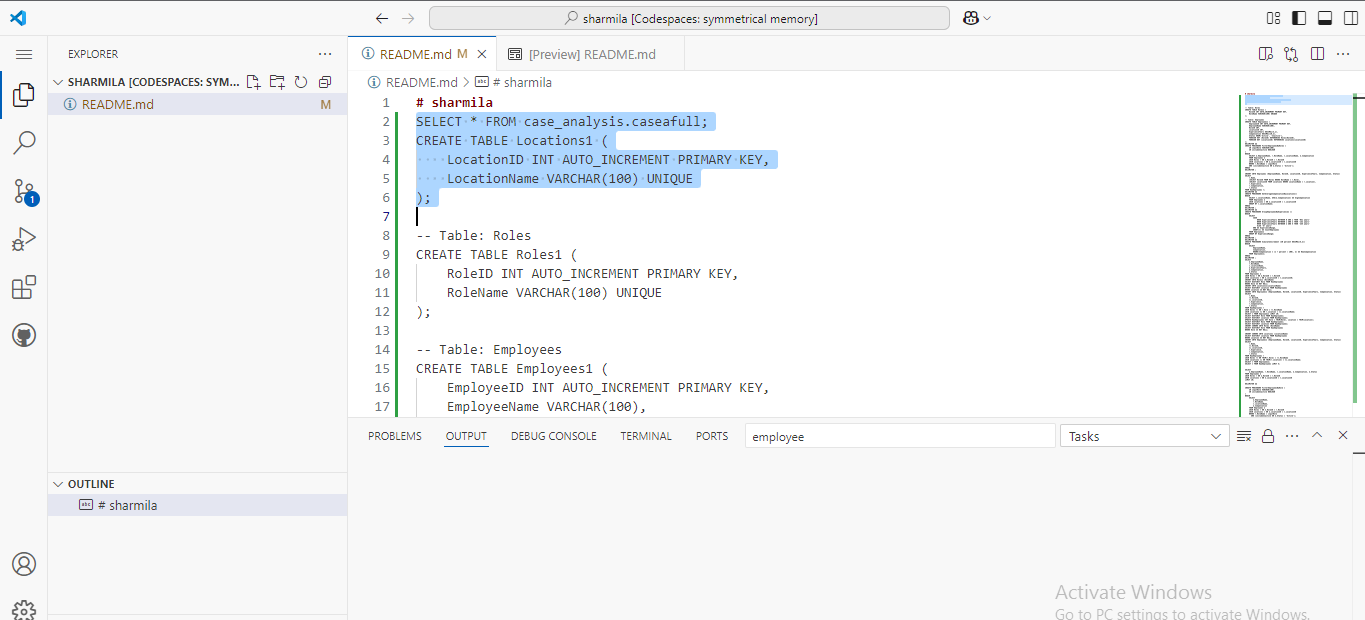
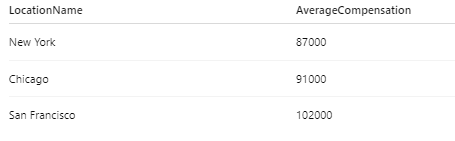
CASE STUDY ASSIGNMENT:

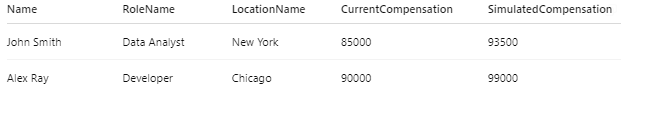














Code used:

**# sharmila**

SELECT \* FROM case\_analysis.caseafull;

CREATE TABLE Locations1 (

    LocationID INT AUTO\_INCREMENT PRIMARY KEY,

    LocationName VARCHAR(100) UNIQUE

);

-- Table: Roles

CREATE TABLE Roles1 (

    RoleID INT AUTO\_INCREMENT PRIMARY KEY,

    RoleName VARCHAR(100) UNIQUE

);

-- Table: Employees

CREATE TABLE Employees1 (

    EmployeeID INT AUTO\_INCREMENT PRIMARY KEY,

    EmployeeName VARCHAR(100),

    RoleID INT,

    LocationID INT,

    ExperienceYears DECIMAL(3,1),

    Compensation DECIMAL(10,2),

    Status ENUM('Active', 'Inactive'),

    FOREIGN KEY (RoleID) REFERENCES Roles(RoleID),

    FOREIGN KEY (LocationID) REFERENCES Locations(LocationID)

);

DELIMITER $$

CREATE PROCEDURE FilterEmployeesByRole1 (

    IN inputRole VARCHAR(100),

    IN includeInactive BOOLEAN

)

BEGIN

    SELECT e.EmployeeName, r.RoleName, l.LocationName, e.Compensation

    FROM Employees e

    JOIN Roles r ON e.RoleID = r.RoleID

    JOIN Locations l ON e.LocationID = l.LocationID

    WHERE r.RoleName = inputRole

    AND (includeInactive OR e.Status = 'Active');

END$$

DELIMITER ;

INSERT INTO Employees (EmployeeName, RoleID, LocationID, ExperienceYears, Compensation, Status)

SELECT

    r.Name,

    (SELECT RoleID FROM Roles WHERE RoleName = r.Role),

    (SELECT LocationID FROM Locations WHERE LocationName = r.Location),

    r.Experience,

    r.Compensation,

    r.Status

FROM RawEmployees r;

DELIMITER $$

CREATE PROCEDURE GetAverageCompensationByLocation1()

BEGIN

    SELECT l.LocationName, AVG(e.Compensation) AS AvgCompensation

    FROM Employees e

    JOIN Locations l ON e.LocationID = l.LocationID

    GROUP BY l.LocationName;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE GroupEmployeesByExperience1 ()

BEGIN

    SELECT

        CASE

            WHEN ExperienceYears BETWEEN 0 AND 1 THEN '0–1 years'

            WHEN ExperienceYears BETWEEN 1 AND 2 THEN '1–2 years'

            WHEN ExperienceYears BETWEEN 2 AND 5 THEN '2–5 years'

            ELSE '5+ years'

        END AS ExperienceRange,

        COUNT(\*) AS CountEmployees

    FROM Employees1

    GROUP BY ExperienceRange;

END$$

DELIMITER ;

DELIMITER $$

CREATE PROCEDURE SimulateIncrement (IN percent DECIMAL(5,2))

BEGIN

    SELECT

        EmployeeName,

        Compensation,

        ROUND(Compensation \* (1 + percent / 100), 2) AS NewCompensation

    FROM Employees1;

END$$

DELIMITER ;

SELECT

    e.EmployeeName,

    r.RoleName,

    l.LocationName,

    e.ExperienceYears,

    e.Compensation,

    e.Status

FROM Employees e

JOIN Roles r ON e.RoleID = r.RoleID

JOIN Locations l ON e.LocationID = l.LocationID;

INSERT INTO Roles1 (RoleName)

SELECT DISTINCT Role FROM RawEmployees

WHERE Role IS NOT NULL;

INSERT INTO Locations1(LocationName)

SELECT DISTINCT Location FROM RawEmployees

WHERE Location IS NOT NULL;

INSERT INTO Employees1 (EmployeeName, RoleID, LocationID, ExperienceYears, Compensation, Status)

SELECT

    r.Name,

    rl.RoleID,

    lc.LocationID,

    r.Experience,

    r.Compensation,

    r.Status

FROM RawEmployees r

JOIN Roles rl ON r.Role = rl.RoleName

JOIN Locations lc ON r.Location = lc.LocationName;

SELECT \* FROM Employees1 LIMIT 10;

SELECT DISTINCT Role FROM RawEmployees;

SELECT DISTINCT Location FROM RawEmployees;

UPDATE RawEmployees SET Role = TRIM(Role), Location = TRIM(Location);

SELECT DISTINCT Role FROM RawEmployees;

SELECT DISTINCT Location FROM RawEmployees;

INSERT IGNORE INTO Roles (RoleName)

SELECT DISTINCT Role FROM RawEmployees

WHERE Role IS NOT NULL;

INSERT IGNORE INTO Locations (LocationName)

SELECT DISTINCT Location FROM RawEmployees

WHERE Location IS NOT NULL;

INSERT INTO Employees1 (EmployeeName, RoleID, LocationID, ExperienceYears, Compensation, Status)

SELECT

    r.Name,

    rl.RoleID,

    lc.LocationID,

    r.Experience,

    r.Compensation,

    r.Status

FROM RawEmployees r

JOIN Roles rl ON TRIM(r.Role) = rl.RoleName

JOIN Locations lc ON TRIM(r.Location) = lc.LocationName;

SELECT \* FROM Employees1;

SELECT \* FROM RawEmployees LIMIT 5;

SELECT

    e.EmployeeName, r.RoleName, l.LocationName, e.Compensation, e.Status

FROM Employees e

JOIN Roles r ON e.RoleID = r.RoleID

JOIN Locations l ON e.LocationID = l.LocationID

LIMIT 10;

DELIMITER $$

CREATE PROCEDURE FilterEmployeesByRole (

    IN inputRole VARCHAR(100),

    IN includeInactive BOOLEAN

)

BEGIN

    SELECT

        e.EmployeeName,

        r.RoleName,

        l.LocationName,

        e.Compensation

    FROM Employees e

    JOIN Roles r ON e.RoleID = r.RoleID

    JOIN Locations l ON e.LocationID = l.LocationID

    WHERE r.RoleName = inputRole

      AND (includeInactive OR e.Status = 'Active');

END$$

DELIMITER ;

CALL FilterEmployeesByRole('Manager', FALSE);

DROP TABLE IF EXISTS Employees1;

DROP TABLE IF EXISTS Roles1;

CREATE TABLE role1 (

    RoleID INT AUTO\_INCREMENT PRIMARY KEY,

    RoleName VARCHAR(100) UNIQUE NOT NULL

);

-- Create Location table

CREATE TABLE location1 (

    LocationID INT AUTO\_INCREMENT PRIMARY KEY,

    LocationName VARCHAR(100) UNIQUE NOT NULL

);

-- Create Employees table

CREATE TABLE employees1 (

    EmployeeID INT AUTO\_INCREMENT PRIMARY KEY,

    EmployeeName VARCHAR(100) NOT NULL,

    RoleID INT,

    LocationID INT,

    ExperienceYears DECIMAL(4,2),

    Compensation DECIMAL(10,2),

    Status VARCHAR(20),

    FOREIGN KEY (RoleID) REFERENCES role1(RoleID),

    FOREIGN KEY (LocationID) REFERENCES location1(LocationID)

);

UPDATE RawEmployees SET Role = TRIM(Role), Location = TRIM(Location);

-- Insert into role1 table

INSERT IGNORE INTO role1 (RoleName)

SELECT DISTINCT Role FROM RawEmployees WHERE Role IS NOT NULL;

-- Insert into location1 table

INSERT IGNORE INTO location1 (LocationName)

SELECT DISTINCT Location FROM RawEmployees WHERE Location IS NOT NULL;

INSERT INTO employees1 (EmployeeName, RoleID, LocationID, ExperienceYears, Compensation, Status)

SELECT

    r.Name,

    rl.RoleID,

    lc.LocationID,

    r.Experience,

    r.Compensation,

    r.Status

FROM RawEmployees r

JOIN role1 rl ON TRIM(r.Role) = rl.RoleName

JOIN location1 lc ON TRIM(r.Location) = lc.LocationName;

SELECT

    e.EmployeeName,

    rl.RoleName,

    lc.LocationName,

    e.ExperienceYears,

    e.Compensation,

    e.Status

FROM employees1 e

JOIN role1 rl ON e.RoleID = rl.RoleID

JOIN location1 lc ON e.LocationID = lc.LocationID

LIMIT 10;

SELECT \* FROM RawEmployees LIMIT 10;

SELECT DISTINCT TRIM(Role) AS Role FROM RawEmployees WHERE Role IS NOT NULL;

SELECT DISTINCT TRIM(Location) AS Location FROM RawEmployees WHERE Location1 IS NOT NULL;

SELECT Role, Location FROM RawEmployees LIMIT 10;

SELECT \* FROM role1;

SELECT \* FROM location1;

SELECT

    r.Name,

    rl.RoleID,

    lc.LocationID,

    r.Experience,

    r.Compensation,

    r.Status

FROM RawEmployees r

JOIN role1 rl ON TRIM(r.Role) = rl.RoleName

JOIN location1 lc ON TRIM(r.Location) = lc.LocationName

LIMIT 10;

SELECT

    r.Name,

    rl.RoleID,

    lc.LocationID,

    r.Experience,

    r.Compensation,

    r.Status

FROM RawEmployees r

JOIN role1 rl ON LOWER(TRIM(r.Role)) = LOWER(rl.RoleName)

JOIN location1 lc ON LOWER(TRIM(r.Location)) = LOWER(lc.LocationName)

LIMIT 10;

INSERT INTO employees1 (EmployeeName, RoleID, LocationID, ExperienceYears, Compensation, Status)

SELECT

    r.Name,

    rl.RoleID,

    lc.LocationID,

    r.Experience,

    r.Compensation,

    r.Status

FROM RawEmployees r

JOIN role1 rl ON LOWER(TRIM(r.Role)) = LOWER(rl.RoleName)

JOIN location1 lc ON LOWER(TRIM(r.Location)) = LOWER(lc.LocationName);

SELECT \* FROM employees1 LIMIT 10;

DELIMITER $$

CREATE PROCEDURE FilterEmployeesByRole3 (

    IN inputRole VARCHAR(100),

    IN includeInactive BOOLEAN

)

BEGIN

    SELECT

        e.EmployeeName,

        r.RoleName,

        l.LocationName,

        e.Compensation

    FROM employees1 e

    JOIN role1 r ON e.RoleID = r.RoleID

    JOIN location1 l ON e.LocationID = l.LocationID

    WHERE r.RoleName = inputRole

      AND (includeInactive OR e.Status = 'Active');

END$$

DELIMITER ;

CALL FilterEmployeesByRole3('Manager', FALSE);

DELIMITER $$

CREATE PROCEDURE GetAverageCompensationByLocation3()

BEGIN

    SELECT

        l.LocationName,

        ROUND(AVG(e.Compensation), 2) AS AvgCompensation

    FROM employees1 e

    JOIN location1 l ON e.LocationID = l.LocationID

    GROUP BY l.LocationName;

END$$

DELIMITER ;

CALL GetAverageCompensationByLocation3();

DELIMITER $$

CREATE PROCEDURE GroupEmployeesByExperience3()

BEGIN

    SELECT

        CASE

            WHEN ExperienceYears BETWEEN 0 AND 1 THEN '0–1 years'

            WHEN ExperienceYears BETWEEN 1 AND 2 THEN '1–2 years'

            WHEN ExperienceYears BETWEEN 2 AND 5 THEN '2–5 years'

            ELSE '5+ years'

        END AS ExperienceRange,

        COUNT(\*) AS CountEmployees

    FROM employees1

    GROUP BY ExperienceRange;

END$$

DELIMITER ;

CALL GroupEmployeesByExperience3();

DELIMITER $$

CREATE PROCEDURE SimulateIncrement3 (

    IN percent DECIMAL(5,2)

)

BEGIN

    SELECT

        EmployeeName,

        Compensation,

        ROUND(Compensation \* (1 + percent / 100), 2) AS NewCompensation

    FROM employees1;

    CALL SimulateIncrement3(10);

END$$

DELIMITER ;

CALL SimulateIncrement3(10);

DELIMITER $$

DELIMITER $$

CREATE PROCEDURE SimulateIncrement5 (

    IN percent DECIMAL(5,2)

)

BEGIN

    SELECT

        EmployeeName,

        Compensation,

        ROUND(Compensation \* (1 + percent / 100), 2) AS NewCompensation

    FROM employees1;

END$$

DELIMITER ;

CALL SimulateIncrement5(10);

SELECT EmployeeName, Compensation FROM employees1;

UPDATE employees1

SET Compensation = 50000

WHERE Compensation IS NULL;

CALL SimulateIncrement5(10);