Name: Gaurav Shivaji Tanpure

PRN: 123B2B329

College: PCCOE

Domain: SQL

WEEK 3 TASK

```
--TASK 1
CREATE TABLE Projects (
   Task_ID INT,
   Start_Date DATE,
   End_Date DATE
);

SELECT * FROM Projects;

INSERT INTO Projects (Task_ID, Start_Date, End_Date)
VALUES
(1, '2015-10-01', '2015-10-02'),
(2, '2015-10-02', '2015-10-03'),
(3, '2015-10-03', '2015-10-04'),
(4, '2015-10-13', '2015-10-14'),
(5, '2015-10-14', '2015-10-15'),
(6, '2015-10-28', '2015-10-29'),
(7, '2015-10-30', '2015-10-31');
```

```
WITH ProjectGroups AS (
   SELECT *,
          DATEADD(DAY, -ROW_NUMBER() OVER (ORDER BY Start_Date), Start_Date) AS
grp
   FROM Projects
),
Grouped AS (
   SELECT
      MIN(Start_Date) AS Project_Start,
      MAX(End_Date) AS Project_End,
      DATEDIFF(DAY, MIN(Start_Date), MAX(End_Date)) + 1 AS Duration
   FROM ProjectGroups
   GROUP BY grp
)
SELECT FORMAT(Project_Start, 'yyyy-MM-dd') + ' ' + FORMAT(Project_End, 'yyyy-MM-dd')
AS Output
FROM Grouped
ORDER BY Duration DESC, Project_Start;


--TASK 2
CREATE TABLE Students (
   ID INT,
   Name VARCHAR(50)
);


CREATE TABLE Friends (
   ID INT,
```

```
    Friend_ID INT
);

CREATE TABLE Packages (
    ID INT,
    Salary FLOAT
);

INSERT INTO Students (ID, Name)
VALUES
(1, 'Ashley'),
(2, 'Samantha'),
(3, 'Julie'),
(4, 'Scarlet');

INSERT INTO Friends (ID, Friend_ID)
VALUES
(1, 2),
(2, 3),
(3, 4),
(4, 1);

INSERT INTO Packages (ID, Salary)
VALUES
(1, 15.20),
(2, 10.05),
```

```
(3, 11.55),
(4, 12.12);

SELECT S.Name
FROM Students S
JOIN Friends F ON S.ID = F.ID
JOIN Packages P1 ON S.ID = P1.ID         -- Student's salary
JOIN Packages P2 ON F.Friend_ID = P2.ID  -- Friend's salary
WHERE P2.Salary > P1.Salary
ORDER BY P2.Salary DESC;

--TASK 3
CREATE TABLE Functions (
   X INT,
   Y INT
);

INSERT INTO Functions (X, Y)
VALUES
(20, 20),
(20, 20),
(20, 21),
(23, 22),
(22, 23),
(21, 20);
```

```
SELECT DISTINCT f1.X, f1.Y
FROM Functions f1
JOIN Functions f2 ON f1.X = f2.Y AND f1.Y = f2.X
WHERE f1.X <= f1.Y
ORDER BY f1.X;


--TASK 4
-- Contests table
CREATE TABLE Contests (
    contest_id INT,
    hacker_id INT,
    name VARCHAR(100)
);

-- Colleges table
CREATE TABLE Colleges (
    college_id INT,
    contest_id INT
);

-- Challenges table
CREATE TABLE Challenges (
    challenge_id INT,
    college_id INT
);
```

```
-- View_Stats table
CREATE TABLE View_Stats (
    challenge_id INT,
    total_views INT,
    total_unique_views INT
);

-- Submission_Stats table
CREATE TABLE Submission_Stats (
    challenge_id INT,
    total_submissions INT,
    total_accepted_submissions INT
);

-- Contests
INSERT INTO Contests VALUES (66406, 17973, 'Rose');
INSERT INTO Contests VALUES (66556, 79153, 'Angela');
INSERT INTO Contests VALUES (94828, 80275, 'Frank');

-- Colleges
INSERT INTO Colleges VALUES (11219, 66406);
INSERT INTO Colleges VALUES (32473, 66556);
INSERT INTO Colleges VALUES (56685, 94828);

-- Challenges
INSERT INTO Challenges VALUES (18765, 11219);
```

```
INSERT INTO Challenges VALUES (47127, 11219);
INSERT INTO Challenges VALUES (60292, 32473);
INSERT INTO Challenges VALUES (72974, 56685);

-- View_Stats
INSERT INTO View_Stats VALUES (47127, 26, 19);
INSERT INTO View_Stats VALUES (47127, 15, 14);
INSERT INTO View_Stats VALUES (18765, 43, 10);
INSERT INTO View_Stats VALUES (18765, 72, 13);
INSERT INTO View_Stats VALUES (75516, 35, 17);
INSERT INTO View_Stats VALUES (60292, 11, 10);
INSERT INTO View_Stats VALUES (72974, 41, 15);
INSERT INTO View_Stats VALUES (75516, 75, 11);


-- Submission_Stats
INSERT INTO Submission_Stats VALUES (75516, 34, 12);
INSERT INTO Submission_Stats VALUES (47127, 27, 10);
INSERT INTO Submission_Stats VALUES (47127, 56, 18);
INSERT INTO Submission_Stats VALUES (75516, 74, 12);
INSERT INTO Submission_Stats VALUES (75516, 83, 8);
INSERT INTO Submission_Stats VALUES (72974, 68, 24);
INSERT INTO Submission_Stats VALUES (72974, 82, 14);
INSERT INTO Submission_Stats VALUES (47127, 28, 11);

SELECT
```

```
    c.contest_id,
    c.hacker_id,
    c.name,
    SUM(COALESCE(s.total_submissions, 0)) AS total_submissions,
    SUM(COALESCE(s.total_accepted_submissions, 0)) AS total_accepted_submissions,
    SUM(COALESCE(v.total_views, 0)) AS total_views,
    SUM(COALESCE(v.total_unique_views, 0)) AS total_unique_views
FROM Contests c
JOIN Colleges col ON c.contest_id = col.contest_id
JOIN Challenges ch ON col.college_id = ch.college_id
LEFT JOIN View_Stats v ON ch.challenge_id = v.challenge_id
LEFT JOIN Submission_Stats s ON ch.challenge_id = s.challenge_id
GROUP BY c.contest_id, c.hacker_id, c.name
HAVING
    SUM(COALESCE(s.total_submissions, 0)) > 0 OR
    SUM(COALESCE(s.total_accepted_submissions, 0)) > 0 OR
    SUM(COALESCE(v.total_views, 0)) > 0 OR
    SUM(COALESCE(v.total_unique_views, 0)) > 0
ORDER BY c.contest_id;

-- TASK 5

CREATE TABLE Hackers (
    hacker_id INT PRIMARY KEY,
    name VARCHAR(50)
);
```

```
CREATE TABLE Submissions (
    submission_date DATE,
    submission_id INT,
    hacker_id INT,
    score INT
);

INSERT INTO Hackers (hacker_id, name) VALUES
(15758, 'Rose'),
(20703, 'Angela'),
(36396, 'Frank'),
(38289, 'Patrick'),
(44065, 'Lisa'),
(53473, 'Kimberly'),
(62529, 'Bonnie'),
(79722, 'Michael');

INSERT INTO Submissions (submission_date, submission_id, hacker_id, score) VALUES
('2016-03-01', 8494, 20703, 0),
('2016-03-01', 22403, 53473, 15),
('2016-03-01', 23965, 79722, 60),
('2016-03-01', 30173, 36396, 70),
('2016-03-02', 34928, 20703, 0),
('2016-03-02', 38740, 15758, 60),
('2016-03-02', 42769, 79722, 25),
```

```
('2016-03-02', 44364, 79722, 60),
('2016-03-03', 45440, 20703, 0),
('2016-03-03', 49050, 36396, 70),
('2016-03-03', 50273, 79722, 5),
('2016-03-04', 50344, 20703, 0),
('2016-03-04', 51360, 44065, 90),
('2016-03-04', 54404, 53473, 65),
('2016-03-04', 61533, 79722, 45),
('2016-03-05', 72852, 20703, 0),
('2016-03-05', 74546, 38289, 0),
('2016-03-05', 76487, 62529, 0),
('2016-03-05', 82439, 36396, 10),
('2016-03-05', 90006, 36396, 40),
('2016-03-06', 90404, 20703, 0);

WITH DailySubmissions AS (
    SELECT submission_date,
        hacker_id,
        COUNT(submission_id) AS submission_count,
        ROW_NUMBER() OVER (PARTITION BY submission_date ORDER BY
COUNT(submission_id) DESC, hacker_id) AS rn
    FROM Submissions
    GROUP BY submission_date, hacker_id
),
DailyUniqueHackers AS (
    SELECT submission_date,
        COUNT(DISTINCT hacker_id) AS unique_hackers
```

```
    FROM Submissions
    GROUP BY submission_date
)
SELECT
    D1.submission_date,
    D2.unique_hackers,
    D1.hacker_id,
    H.name
FROM DailySubmissions D1
JOIN Hackers H ON D1.hacker_id = H.hacker_id
JOIN DailyUniqueHackers D2 ON D1.submission_date = D2.submission_date
WHERE D1.rn = 1
ORDER BY D1.submission_date;

-- TASK 6
CREATE TABLE STATION (
    ID INT,
    CITY VARCHAR(21),
    STATE VARCHAR(2),
    LAT_N FLOAT,
    LONG_W FLOAT
);

INSERT INTO STATION (ID, CITY, STATE, LAT_N, LONG_W) VALUES
(1, 'New York', 'NY', 40.7128, 74.0060),
(2, 'Los Angeles', 'CA', 34.0522, 118.2437),
```

```
(3, 'Chicago', 'IL', 41.8781, 87.6298),
(4, 'Houston', 'TX', 29.7604, 95.3698),
(5, 'Phoenix', 'AZ', 33.4484, 112.0740),
(6, 'Philadelphia', 'PA', 39.9526, 75.1652),
(7, 'San Antonio', 'TX', 29.4241, 98.4936),
(8, 'San Diego', 'CA', 32.7157, 117.1611),
(9, 'Dallas', 'TX', 32.7767, 96.7970),
(10, 'San Jose', 'CA', 37.3382, 121.8863);

SELECT ROUND(ABS(MAX(LAT_N) - MIN(LAT_N)) + ABS(MAX(LONG_W) -
MIN(LONG_W)), 4) AS Manhattan_Distance
FROM STATION;

-- TASK 7
WITH PrimeNumbers AS (
   SELECT 2 AS num
   UNION ALL
   SELECT num + 1
   FROM PrimeNumbers
   WHERE num + 1 <= 1000
),
PrimeFilter AS (
   SELECT num
   FROM PrimeNumbers pn1
   WHERE NOT EXISTS (
      SELECT 1
      FROM PrimeNumbers pn2
```

```
        WHERE pn2.num < pn1.num AND pn1.num % pn2.num = 0
    )
)
-- Combine all primes with '&' separator
SELECT STRING_AGG(CAST(num AS VARCHAR), '&') AS primes
FROM PrimeFilter
OPTION (MAXRECURSION 1000);

-- TASK 8
CREATE TABLE Occupations (
    Name VARCHAR(50),
    Occupation VARCHAR(50)
);

INSERT INTO Occupations (Name, Occupation) VALUES
('Samantha', 'Doctor'),
('Julia', 'Actor'),
('Maria', 'Actor'),
('Meera', 'Singer'),
('Ashley', 'Professor'),
('Ketty', 'Professor'),
('Christeen', 'Professor'),
('Jane', 'Actor'),
('Jenny', 'Doctor'),
('Priya', 'Singer');
```

```
SELECT
   MAX(CASE WHEN Occupation = 'Doctor' THEN Name ELSE NULL END) AS Doctor,
   MAX(CASE WHEN Occupation = 'Professor' THEN Name ELSE NULL END) AS Professor,
   MAX(CASE WHEN Occupation = 'Singer' THEN Name ELSE NULL END) AS Singer,
   MAX(CASE WHEN Occupation = 'Actor' THEN Name ELSE NULL END) AS Actor
FROM (
   SELECT Name, Occupation, ROW_NUMBER() OVER (PARTITION BY Occupation
ORDER BY Name) AS RowNum
   FROM Occupations
) AS Piv
GROUP BY RowNum
ORDER BY RowNum;


-- TASK 9
CREATE TABLE BST (
   N INT,
   P INT
);


INSERT INTO BST (N, P) VALUES
(1, 2),
(3, 2),
(6, 8),
(9, 8),
(2, 5),
(8, 5),
(5, NULL);
```

```
WITH NodeTypes AS (
    SELECT N,
           P,
           CASE
               WHEN P IS NULL THEN 'Root'
               WHEN N NOT IN (SELECT P FROM BST WHERE P IS NOT NULL) THEN 'Leaf'
               ELSE 'Inner'
           END AS NodeType
    FROM BST
)
SELECT N, NodeType
FROM NodeTypes
ORDER BY N;

-- TASK 10
CREATE TABLE Company (
    company_code VARCHAR(10),
    founder VARCHAR(100)
);

INSERT INTO Company VALUES
('C1', 'Monika'),
('C2', 'Samantha');

CREATE TABLE Lead_Manager (
```

```
    lead_manager_code VARCHAR(10),
    company_code VARCHAR(10)
);

INSERT INTO Lead_Manager VALUES
('LM1', 'C1'),
('LM2', 'C2');

CREATE TABLE Senior_Manager (
    senior_manager_code VARCHAR(10),
    lead_manager_code VARCHAR(10),
    company_code VARCHAR(10)
);

INSERT INTO Senior_Manager VALUES
('SM1', 'LM1', 'C1'),
('SM2', 'LM1', 'C1'),
('SM3', 'LM2', 'C2');

CREATE TABLE Manager (
    manager_code VARCHAR(10),
    senior_manager_code VARCHAR(10),
    lead_manager_code VARCHAR(10),
    company_code VARCHAR(10)
);
```

```
INSERT INTO Manager VALUES
('M1', 'SM1', 'LM1', 'C1'),
('M2', 'SM3', 'LM2', 'C2'),
('M3', 'SM3', 'LM2', 'C2');

CREATE TABLE Employee (
    employee_code VARCHAR(10),
    manager_code VARCHAR(10),
    senior_manager_code VARCHAR(10),
    lead_manager_code VARCHAR(10),
    company_code VARCHAR(10)
);

INSERT INTO Employee VALUES
('E1', 'M1', 'SM1', 'LM1', 'C1'),
('E2', 'M1', 'SM1', 'LM1', 'C1'),
('E3', 'M2', 'SM3', 'LM2', 'C2'),
('E4', 'M3', 'SM3', 'LM2', 'C2');

SELECT
    c.company_code,
    c.founder,
    COUNT(DISTINCT lm.lead_manager_code) AS lead_managers,
    COUNT(DISTINCT sm.senior_manager_code) AS senior_managers,
    COUNT(DISTINCT m.manager_code) AS managers,
    COUNT(DISTINCT e.employee_code) AS employees
```

```
FROM Company c
LEFT JOIN Lead_Manager lm ON c.company_code = lm.company_code
LEFT JOIN Senior_Manager sm ON c.company_code = sm.company_code
LEFT JOIN Manager m ON c.company_code = m.company_code
LEFT JOIN Employee e ON c.company_code = e.company_code
GROUP BY c.company_code, c.founder
ORDER BY c.company_code;

-- TASK 11
CREATE TABLE Stud (
   ID INT,
   Name VARCHAR(100)
);

CREATE TABLE Friend (
   ID INT,
   Friend_ID INT
);

CREATE TABLE Package (
   ID INT,
   Salary FLOAT
);

INSERT INTO Stud VALUES
(1, 'Ashley'),
```

```
(2, 'Samantha'),
(3, 'Julia'),
(4, 'Scarlet');

INSERT INTO Friend VALUES
(1, 2),
(2, 3),
(3, 4),
(4, 1);

INSERT INTO Package VALUES
(1, 15.20),
(2, 10.06),
(3, 11.55),
(4, 12.12);

SELECT s.Name
FROM Stud s
JOIN Friend f ON s.ID = f.ID
JOIN Package sp ON s.ID = sp.ID          -- Student's salary
JOIN Package fp ON f.Friend_ID = fp.ID  -- Friend's salary
WHERE fp.Salary > sp.Salary
ORDER BY fp.Salary;

-- TASK 12
```

```
CREATE TABLE JobFamilyCost (
    JobFamily VARCHAR(100),
    Country VARCHAR(50),
    Cost FLOAT
);

-- Inserting sample data into JobFamilyCost
INSERT INTO JobFamilyCost (JobFamily, Country, Cost) VALUES
('Engineering', 'India', 5000),
('Engineering', 'International', 15000),
('Marketing', 'India', 3000),
('Marketing', 'International', 7000),
('Finance', 'India', 4000),
('Finance', 'International', 6000),
('HR', 'India', 2500),
('HR', 'International', 4500);

SELECT
    JobFamily,
    SUM(CASE WHEN Country = 'India' THEN Cost ELSE 0 END) AS India_Cost,
    SUM(CASE WHEN Country = 'International' THEN Cost ELSE 0 END) AS
International_Cost,
    ROUND(
        (SUM(CASE WHEN Country = 'India' THEN Cost ELSE 0 END) * 100.0) /
NULLIF(SUM(Cost), 0),
        2
    ) AS India_Percentage,
```

```
    ROUND(
        (SUM(CASE WHEN Country = 'International' THEN Cost ELSE 0 END) * 100.0) /
NULLIF(SUM(Cost), 0),
        2
    ) AS International_Percentage
FROM JobFamilyCost
GROUP BY JobFamily;


-- TASK 13
CREATE TABLE BusinessUnitFinance (
    BU_Name VARCHAR(100),
    Month DATE,
    Cost FLOAT,
    Revenue FLOAT
);


INSERT INTO BusinessUnitFinance (BU_Name, Month, Cost, Revenue) VALUES
('Tech', '2024-01-01', 10000, 25000),
('Tech', '2024-02-01', 12000, 28000),
('Tech', '2024-03-01', 11000, 30000),
('HR', '2024-01-01', 5000, 10000),
('HR', '2024-02-01', 4500, 11000),
('HR', '2024-03-01', 5200, 12000);


SELECT
    BU_Name,
    FORMAT(Month, 'yyyy-MM') AS Month,
```

```
    Cost,
    Revenue,
    ROUND(CASE
        WHEN Revenue = 0 THEN NULL
        ELSE (Cost / Revenue) * 100
    END, 2) AS Cost_Revenue_Ratio_Percentage
FROM BusinessUnitFinance
ORDER BY BU_Name, Month;

-- TASK 14
CREATE TABLE YourTable (
    EmployeeID INT,
    SubBand VARCHAR(10)
);

INSERT INTO YourTable (EmployeeID, SubBand) VALUES
(1, 'A1'),
(2, 'A1'),
(3, 'A2'),
(4, 'A2'),
(5, 'A2'),
(6, 'A3'),
(7, 'A3'),
(8, 'A3'),
(9, 'A3'),
(10, 'A4');
```

```
SELECT
   SubBand,
   COUNT(*) AS Headcount,
   ROUND(COUNT(*) * 100.0 / SUM(COUNT(*)) OVER (), 2) AS Percentage_Headcount
FROM YourTable
GROUP BY SubBand;

-- TASK 15
CREATE TABLE Employees (
   EmpID INT,
   EmpName VARCHAR(50),
   Salary DECIMAL(10, 2)
);

INSERT INTO Employees (EmpID, EmpName, Salary) VALUES
(1, 'Alice', 7000),
(2, 'Bob', 9000),
(3, 'Charlie', 8000),
(4, 'David', 9500),
(5, 'Eve', 8500),
(6, 'Frank', 7500),
(7, 'Grace', 9200);

SELECT EmpID, EmpName, Salary
FROM (
```

```
    SELECT *,
        DENSE_RANK() OVER (ORDER BY Salary DESC) AS rnk
    FROM Employees
) AS ranked
WHERE rnk <= 5;

-- TASK 16
CREATE TABLE Employee4 (
    EmpID INT,
    Col1 INT,
    Col2 INT
);

INSERT INTO Employee4 (EmpID, Col1, Col2) VALUES
(1, 10, 100),
(2, 20, 200),
(3, 30, 300);

UPDATE E
SET Col1 = V.Col1,
    Col2 = V.Col2
FROM Employee4 E
JOIN (
    SELECT EmpID, Col2 AS Col1, Col1 AS Col2
    FROM Employee4
) V ON E.EmpID = V.EmpID;
```

```
SELECT * FROM Employee4;


-- TASK 17
-- Step 1: Create a new database
CREATE DATABASE SampleDB;
GO

-- Step 2: Use the newly created database
USE SampleDB;
GO

-- Step 3: Create a new SQL Server login
CREATE LOGIN SampleUserLogin WITH PASSWORD = 'Str0ng@Pass123';
GO

-- Step 4: Create a user in SampleDB for the login
CREATE USER SampleUser FOR LOGIN SampleUserLogin;
GO

-- Step 5: Grant db_owner role to the user
EXEC sp_addrolemember 'db_owner', 'SampleUser';
GO

USE SampleDB;
GO
```

```
SELECT
    dp1.name AS RoleName,
    dp2.name AS UserName
FROM
    sys.database_role_members drm
JOIN
    sys.database_principals dp1 ON drm.role_principal_id = dp1.principal_id
JOIN
    sys.database_principals dp2 ON drm.member_principal_id = dp2.principal_id
WHERE
    dp2.name = 'SampleUser';


-- TASK 18
CREATE TABLE EmployeeCosts (
    EmployeeID INT,
    BU VARCHAR(50),
    Salary DECIMAL(10,2),
    WorkHours INT,
    [Month] VARCHAR(7)
);

INSERT INTO EmployeeCosts (EmployeeID, BU, Salary, WorkHours, [Month]) VALUES
(1, 'Finance', 5000.00, 160, '2025-01'),
(2, 'Finance', 6000.00, 170, '2025-01'),
(3, 'Finance', 5500.00, 165, '2025-02'),
```

```
(4, 'IT', 7000.00, 150, '2025-01'),
(5, 'IT', 7200.00, 145, '2025-02');

SELECT
   BU,
   [Month],
   CAST(SUM(Salary * WorkHours) * 1.0 / NULLIF(SUM(WorkHours), 0) AS DECIMAL(10,2)) AS WeightedAvgCost
FROM EmployeeCosts
GROUP BY BU, [Month]
ORDER BY BU, [Month];

-- TASK 19
CREATE TABLE Employees3 (
   EmployeeID INT,
   Salary INT
);

INSERT INTO Employees3 (EmployeeID, Salary) VALUES
(1, 50000),
(2, 60000),
(3, 70000),
(4, 80000),
(5, 90000);

WITH Actual AS (
   SELECT AVG(Salary) AS ActualAvgSalary
```

```
    FROM Employees3
),
Miscalculated AS (
    SELECT AVG(CAST(REPLACE(CAST(Salary AS VARCHAR), '0', '') AS INT)) AS
MiscalculatedAvgSalary
    FROM Employees3
)
SELECT CEILING(Actual.ActualAvgSalary - Miscalculated.MiscalculatedAvgSalary) AS
ErrorAmount
FROM Actual, Miscalculated;


-- TASK 20
CREATE TABLE SourceTable (
    KeyColumn INT PRIMARY KEY,
    Column1 VARCHAR(50),
    Column2 VARCHAR(50)
);

CREATE TABLE TargetTable (
    KeyColumn INT PRIMARY KEY,
    Column1 VARCHAR(50),
    Column2 VARCHAR(50)
);

INSERT INTO SourceTable (KeyColumn, Column1, Column2) VALUES
(1, 'A', 'W'),
(2, 'B', 'X'),
```

```
(3, 'C', 'Y'),
(4, 'D', 'Z');

INSERT INTO TargetTable (KeyColumn, Column1, Column2) VALUES
(1, 'A', 'X');

INSERT INTO TargetTable (KeyColumn, Column1, Column2)
SELECT s.KeyColumn, s.Column1, s.Column2
FROM SourceTable s
WHERE NOT EXISTS (
   SELECT 1
   FROM TargetTable t
   WHERE t.KeyColumn = s.KeyColumn
);
```