ANS 1

WITH ConsecutiveTasks AS (

SELECT

Task\_ID,

Start\_Date,

End\_Date,

LAG(End\_Date) OVER (ORDER BY Start\_Date) AS Prev\_End\_Date

FROM

project

),

Projects AS (

SELECT

Task\_ID,

Start\_Date,

End\_Date,

SUM(CASE

WHEN Start\_Date = DATE\_ADD(Prev\_End\_Date, INTERVAL 1 DAY) THEN 0

ELSE 1

END) OVER (ORDER BY Start\_Date) AS Project\_ID

FROM

ConsecutiveTasks

),

ProjectDates AS (

SELECT

Project\_ID,

MIN(Start\_Date) AS Project\_Start\_Date,

MAX(End\_Date) AS Project\_End\_Date,

DATEDIFF(MAX(End\_Date), MIN(Start\_Date)) + 1 AS Duration

FROM

Projects

GROUP BY

Project\_ID

)

SELECT

Project\_Start\_Date AS Start\_Date,

Project\_End\_Date AS End\_Date,

Duration

FROM

ProjectDates

ORDER BY

Duration ASC,

Project\_Start\_Date ASC;

ANS 2

SELECT

S.Name

FROM

Students S

JOIN

Friends F ON S.ID = F.ID

JOIN

Packages P1 ON S.ID = P1.ID

JOIN

Packages P2 ON F.Friend\_ID = P2.ID

WHERE

P2.Salary > P1.Salary

ORDER BY

P2.Salary;

ANS 3

SELECT

f1.X, f1.Y

FROM

function f1

JOIN

function f2 ON f1.X = f2.Y AND f1.Y = f2.X

WHERE

f1.X < f1.Y

ORDER BY

f1.X, f1.Y;

ANS 6

WITH MinMaxValues AS (

SELECT

MIN(LAT\_N) AS Min\_LAT\_N,

MAX(LAT\_N) AS Max\_LAT\_N,

MIN(LONG\_W) AS Min\_LONG\_W,

MAX(LONG\_W) AS Max\_LONG\_W

FROM

STATION

)

SELECT

ROUND(ABS(Max\_LAT\_N - Min\_LAT\_N) + ABS(Max\_LONG\_W - Min\_LONG\_W), 4) AS Manhattan\_Distance

FROM

MinMaxValues;

ANS 5

WITH ContestDates AS (

SELECT DATE '2016-03-01' AS contest\_date

UNION ALL

SELECT contest\_date + INTERVAL '1' DAY

FROM ContestDates

WHERE contest\_date < DATE '2016-03-15'

),

DailySubmissions AS (

SELECT

submission\_date,

hacker\_id,

COUNT(\*) AS submissions

FROM Submissions

GROUP BY submission\_date, hacker\_id

),

DailyUniqueHackers AS (

SELECT

contest\_date,

COUNT(DISTINCT hacker\_id) AS unique\_hackers

FROM ContestDates cd

JOIN DailySubmissions ds ON cd.contest\_date = ds.submission\_date

GROUP BY contest\_date

),

MaxSubmissionsPerDay AS (

SELECT

submission\_date,

hacker\_id,

submissions,

RANK() OVER (PARTITION BY submission\_date ORDER BY submissions DESC, hacker\_id ASC) AS rank

FROM DailySubmissions

),

TopHackerPerDay AS (

SELECT

msd.submission\_date,

msd.hacker\_id,

h.name,

msd.submissions

FROM MaxSubmissionsPerDay msd

JOIN Hackers h ON msd.hacker\_id = h.hacker\_id

WHERE msd.rank = 1

)

SELECT

cuh.contest\_date,

cuh.unique\_hackers,

thpd.hacker\_id,

thpd.name,

thpd.submissions

FROM

DailyUniqueHackers cuh

JOIN

TopHackerPerDay thpd ON cuh.contest\_date = thpd.submission\_date

ORDER BY

cuh.contest\_date;

ANS 7

WITH RECURSIVE Numbers AS (

SELECT 2 AS num

UNION ALL

SELECT num + 1

FROM Numbers

WHERE num < 1000

),

PrimeCheck AS (

SELECT num,

CASE

WHEN num < 2 THEN 0

WHEN num = 2 THEN 1

ELSE CASE

WHEN NOT EXISTS (

SELECT 1

FROM Numbers AS N

WHERE N.num <= SQRT(P.num) AND P.num % N.num = 0

) THEN 1

ELSE 0

END

END AS is\_prime

FROM Numbers AS P

)

SELECT STRING\_AGG(CAST(num AS VARCHAR), '&') AS primes

FROM PrimeCheck

WHERE is\_prime = 1;

ANS 8

WITH OccupationRanks AS (

SELECT

Name,

Occupation,

ROW\_NUMBER() OVER (PARTITION BY Occupation ORDER BY Name) AS RowNum

FROM

OCCUPATIONS

),

PivotedOccupations AS (

SELECT

RowNum,

MAX(CASE WHEN Occupation = 'Doctor' THEN Name END) AS Doctor,

MAX(CASE WHEN Occupation = 'Professor' THEN Name END) AS Professor,

MAX(CASE WHEN Occupation = 'Singer' THEN Name END) AS Singer,

MAX(CASE WHEN Occupation = 'Actor' THEN Name END) AS Actor

FROM

OccupationRanks

GROUP BY

RowNum

)

SELECT

Doctor, Professor, Singer, Actor

FROM

PivotedOccupations

ORDER BY

RowNum;

ANS 9

WITH NodeTypes AS (

SELECT

N,

CASE

WHEN P IS NULL THEN 'Root'

WHEN N NOT IN (SELECT DISTINCT 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;

ANS 10

SELECT

ec.company\_code,

MAX(c.founder\_name) AS founder\_name,

COUNT(CASE WHEN e.role = 'Lead Manager' THEN 1 END) AS total\_lead\_managers,

COUNT(CASE WHEN e.role = 'Senior Manager' THEN 1 END) AS total\_senior\_managers,

COUNT(CASE WHEN e.role = 'Manager' THEN 1 END) AS total\_managers,

COUNT(e.employee\_id) AS total\_employees

FROM

Employee\_Company ec

JOIN

Employees e ON ec.employee\_id = e.employee\_id

JOIN

Companies c ON ec.company\_code = c.company\_code

GROUP BY

ec.company\_code

ORDER BY

ec.company\_code ASC;

ANS 11

SELECT

s.Name

FROM

Students s

JOIN

Friends f ON s.ID = f.ID

JOIN

Packages ps ON s.ID = ps.ID

JOIN

Packages pf ON f.Friend\_ID = pf.ID

WHERE

pf.Salary > ps.Salary

ORDER BY

pf.Salary;

ANS 12

WITH TotalCosts AS (

SELECT

JobFamilyID,

JobFamilyName,

SUM(CASE WHEN Location = 'India' THEN Cost ELSE 0 END) AS IndiaCost,

SUM(CASE WHEN Location = 'International' THEN Cost ELSE 0 END) AS InternationalCost,

SUM(Cost) AS TotalCost

FROM

JobFamilies

GROUP BY

JobFamilyID, JobFamilyName

),

PercentageCosts AS (

SELECT

JobFamilyID,

JobFamilyName,

IndiaCost,

InternationalCost,

TotalCost,

(IndiaCost \* 100.0 / TotalCost) AS IndiaPercentage,

(InternationalCost \* 100.0 / TotalCost) AS InternationalPercentage

FROM

TotalCosts

)

SELECT

JobFamilyID,

JobFamilyName,

IndiaPercentage,

InternationalPercentage

FROM

PercentageCosts

ORDER BY

JobFamilyID;

ANS 13

SELECT

BU\_ID,

Month,

Cost,

Revenue,

(Cost / Revenue) AS CostRevenueRatio

FROM

BU\_Financials

ORDER BY

BU\_ID,

Month;

ANS 14

SELECT

SubBand,

COUNT(\*) AS Headcount,

(COUNT(\*) \* 100.0 / SUM(COUNT(\*)) OVER ()) AS Percentage

FROM

Employees

GROUP BY

SubBand

ORDER BY

SubBand;

ANS 15

WITH RankedEmployees AS (

SELECT

EmployeeID,

Name,

Salary,

ROW\_NUMBER() OVER (ORDER BY Salary DESC) AS RowNum

FROM

Employees

)

SELECT

EmployeeID,

Name,

Salary

FROM

RankedEmployees

WHERE

RowNum <= 5;

ANS 16

UPDATE Employees

SET

ColumnA = ColumnA + ColumnB,

ColumnB = ColumnA - ColumnB,

ColumnA = ColumnA - ColumnB;

ANS 17

CREATE LOGIN [YourLoginName] WITH PASSWORD = 'YourPassword';

USE [YourDatabaseName];

CREATE USER [YourUserName] FOR LOGIN [YourLoginName];

ALTER ROLE db\_owner ADD MEMBER [YourUserName];

ANS 18

SELECT

BU\_ID,

Month,

SUM(TotalCost) / SUM(EmployeeCount) AS WeightedAverageCostPerEmployee

FROM

EmployeeCosts

GROUP BY

BU\_ID,

Month;

ANS 19

WITH ActualAverage AS (

SELECT AVG(Salary) AS ActualAvgSalary

FROM EMPLOYEES

),

MiscalculatedAverage AS (

SELECT AVG(CAST(REPLACE(CAST(Salary AS VARCHAR), '0', '') AS INT)) AS MiscalculatedAvgSalary

FROM EMPLOYEES

)

SELECT

CEILING(ActualAvgSalary - MiscalculatedAvgSalary) AS SalaryDifference

FROM

ActualAverage, MiscalculatedAverage;

ANS 20

INSERT INTO DestinationTable (ID, Name, Salary, UpdatedAt)

SELECT s.ID, s.Name, s.Salary, s.UpdatedAt

FROM SourceTable s

WHERE NOT EXISTS (

SELECT 1

FROM DestinationTable d

WHERE d.ID = s.ID

);

ANS 4

SELECT

c.contest\_id,

s.hacker\_id,

h.name,

SUM(CASE WHEN s.submission\_id IS NOT NULL THEN 1 ELSE 0 END) AS total\_submissions,

SUM(CASE WHEN s.status = 'accepted' THEN 1 ELSE 0 END) AS total\_accepted\_submissions,

SUM(CASE WHEN cv.view\_type = 'total' THEN 1 ELSE 0 END) AS total\_views,

SUM(CASE WHEN cv.view\_type = 'unique' THEN 1 ELSE 0 END) AS total\_unique\_views

FROM

Contests c

JOIN Submissions s ON c.contest\_id = s.contest\_id

JOIN Hackers h ON s.hacker\_id = h.hacker\_id

LEFT JOIN ContestViews cv ON c.contest\_id = cv.contest\_id AND s.hacker\_id = cv.hacker\_id

GROUP BY

c.contest\_id, s.hacker\_id, h.name

HAVING

SUM(CASE WHEN s.submission\_id IS NOT NULL THEN 1 ELSE 0 END) > 0 OR

SUM(CASE WHEN s.status = 'accepted' THEN 1 ELSE 0 END) > 0 OR

SUM(CASE WHEN cv.view\_type = 'total' THEN 1 ELSE 0 END) > 0 OR

SUM(CASE WHEN cv.view\_type = 'unique' THEN 1 ELSE 0 END) > 0

ORDER BY

c.contest\_id;