TASK 1:

SELECT Start\_Date, min(End\_Date)

FROM

 (SELECT Start\_Date FROM Projects WHERE Start\_Date NOT IN (SELECT End\_Date FROM Projects)) a ,

 (SELECT End\_Date FROM Projects WHERE End\_Date NOT IN (SELECT Start\_Date FROM Projects)) b

WHERE Start\_Date < End\_Date

GROUP BY Start\_Date

ORDER BY DATEDIFF(min(End\_Date), Start\_Date) ASC, Start\_Date ASC;

TASK 2:

select temp1.sn

from (select S.ID si,S.Name sn,P.Salary ps from Students S join Packages P on S.ID=P.ID) temp1 join (select FF.ID fi,FF.Friend\_ID fd,PP.Salary pps from Friends FF join Packages PP on FF.Friend\_ID=pp.ID) temp2 on temp1.si=temp2.fi and temp1.ps<temp2.pps

order by temp2.pps asc;

TASK 3:

SELECT X,

       Y

FROM FUNCTIONS F1

WHERE EXISTS

    (SELECT \*

     FROM FUNCTIONS F2

     WHERE F2.Y = F1.X

       AND F2.X = F1.Y

       AND F2.X > F1.X)

  AND (X != Y)

UNION

SELECT X,

       Y

FROM FUNCTIONS F1

WHERE X = Y

  AND (

         (SELECT COUNT(\*)

          FROM FUNCTIONS

          WHERE X = F1.X

            AND Y = F1.X) > 1)

ORDER BY X;

TASK 4:

select con.contest\_id,

        con.hacker\_id,

        con.name,

        sum(total\_submissions),

        sum(total\_accepted\_submissions),

        sum(total\_views), sum(total\_unique\_views)

from contests con

join colleges col on con.contest\_id = col.contest\_id

join challenges cha on  col.college\_id = cha.college\_id

left join

(select challenge\_id, sum(total\_views) as total\_views, sum(total\_unique\_views) as total\_unique\_views

from view\_stats group by challenge\_id) vs on cha.challenge\_id = vs.challenge\_id

left join

(select challenge\_id, sum(total\_submissions) as total\_submissions, sum(total\_accepted\_submissions) as total\_accepted\_submissions from submission\_stats group by challenge\_id) ss on cha.challenge\_id = ss.challenge\_id

    group by con.contest\_id, con.hacker\_id, con.name

        having sum(total\_submissions)!=0 or

                sum(total\_accepted\_submissions)!=0 or

                sum(total\_views)!=0 or

                sum(total\_unique\_views)!=0

            order by contest\_id;

TASK 5:

SELECT t1.submission\_date, hkr\_cnt, t2.hacker\_id, name

FROM (SELECT p1.submission\_date,

             COUNT(DISTINCT p1.hacker\_id) AS hkr\_cnt

      FROM (SELECT submission\_date, hacker\_id,

                   @h\_rnk := CASE WHEN @h\_grp != hacker\_id THEN 1 ELSE @h\_rnk+1 END AS hacker\_rank,

                   @h\_grp := hacker\_id AS hacker\_group

            FROM (SELECT DISTINCT submission\_date, hacker\_id

                  FROM submissions

                  ORDER BY hacker\_id, submission\_date) AS a,

                 (SELECT @h\_rnk := 1, @h\_grp := 0) AS r) AS p1

      JOIN (SELECT submission\_date,

                   @d\_rnk := @d\_rnk + 1 AS date\_rank

            FROM (SELECT DISTINCT submission\_date

                  FROM submissions

                  ORDER BY submission\_date) AS b,

                 (SELECT @d\_rnk := 0) r) AS p2

      ON p1.submission\_date = p2.submission\_date

         AND hacker\_rank = date\_rank

      GROUP BY p1.submission\_Date) AS t1

JOIN (SELECT submission\_date, hacker\_id, sub\_cnt,

             @s\_rnk := CASE WHEN @d\_grp != submission\_date THEN 1 ELSE @s\_rnk+1 END AS max\_rnk,

             @d\_grp := submission\_date AS date\_group

      FROM (SELECT submission\_date, hacker\_id, COUNT(\*) AS sub\_cnt

            FROM submissions AS s

            GROUP BY submission\_date, hacker\_id

            ORDER BY submission\_date, sub\_cnt DESC, hacker\_id) AS c,

           (SELECT @s\_rnk := 1, @d\_grp := 0) AS r) AS t2

ON t1.submission\_date = t2.submission\_date AND max\_rnk = 1

JOIN hackers AS h ON h.hacker\_id = t2.hacker\_id

ORDER BY t1.submission\_date

;

TASK 6:

select round(sqrt(power(max(LAT\_N) - min(LAT\_N), 2) + power(max(LONG\_W) - min(LONG\_W), 2)), 4)

FROM STATION;

TASK 7:

/\*

Enter your query here.

Please append a semicolon ";" at the end of the query and enter your query in a single line to avoid error.

\*/

DECLARE @Output AS VARCHAR(MAX) = '';

WITH digit(d)

AS

(

SELECT 0 AS d

UNION ALL

SELECT d+1 AS d FROM digit WHERE d < 9

)

SELECT

    @Output += CAST(a.Number AS VARCHAR(3)) + '&'

 FROM (

SELECT a.d \* 100 + b.d\*10 + c.d + 1 AS Number FROM digit a

CROSS JOIN digit b

CROSS JOIN digit c

) a

LEFT JOIN (

SELECT a.d \* 100 + b.d\*10 + c.d + 1 AS Number FROM digit a

CROSS JOIN digit b

CROSS JOIN digit c

) b ON SQRT(a.Number) >= b.Number AND b.Number > 1

WHERE  a.Number > 1

GROUP BY a.Number

HAVING ISNULL(SUM(CASE WHEN a.Number % b.Number = 0 THEN 1 ELSE 0 END),0) = 0

ORDER BY a.Number

PRINT SUBSTRING(@Output,1,LEN(@Output)-1)

;

TASK 8:

select

Doctor,

Professor,

Singer,

Actor

from (

select

NameOrder,

max(case Occupation when 'Doctor' then Name end) as Doctor,

max(case Occupation when 'Professor' then Name end) as Professor,

max(case Occupation when 'Singer' then Name end) as Singer,

max(case Occupation when 'Actor' then Name end) as Actor

from (

select

Occupation,

Name,

row\_number() over(partition by Occupation order by Name ASC) as NameOrder

from Occupations

) as NameLists

group by NameOrder

) as Names

TASK 9:

select N,

       if(P is null, 'Root', if((select count(\*) from BST where P = B.N)> 0, 'Inner', 'Leaf'))

from BST as B

order by N;

TASK 10:

select c.company\_code, c.founder,

       count(distinct l.lead\_manager\_code),

       count(distinct s.senior\_manager\_code),

       count(distinct m.manager\_code),

       count(distinct e.employee\_code)

from Company as c

join Lead\_Manager as l

on c.company\_code = l.company\_code

join Senior\_Manager as s

on l.lead\_manager\_code = s.lead\_manager\_code

join Manager as m

on m.senior\_manager\_code = s.senior\_manager\_code

join Employee as e

on e.manager\_code = m.manager\_code

group by c.company\_code, c.founder

order by c.company\_code;

TASK 11:

select temp1.sn

from (select S.ID si,S.Name sn,P.Salary ps from Students S join Packages P on S.ID=P.ID) temp1 join (select FF.ID fi,FF.Friend\_ID fd,PP.Salary pps from Friends FF join Packages PP on FF.Friend\_ID=pp.ID) temp2 on temp1.si=temp2.fi and temp1.ps<temp2.pps

order by temp2.pps asc;

-- Task 12: Display ratio of cost of job family in percentage by India and international

SELECT

    JobFamily,

    SUM(CASE WHEN Country = 'India' THEN Cost ELSE 0 END) \* 100.0 / SUM(Cost) AS India\_Percentage,

    SUM(CASE WHEN Country != 'India' THEN Cost ELSE 0 END) \* 100.0 / SUM(Cost) AS International\_Percentage

FROM

    SimulationData

GROUP BY

    JobFamily;

-- Task 13: Find ratio of cost and revenue of a BU month on month

SELECT

    BU,

    Month,

    SUM(Cost) / NULLIF(SUM(Revenue), 0) AS Cost\_Revenue\_Ratio

FROM

    BUData

GROUP BY

    BU, Month;

-- Task 14: Show headcounts of sub band and percentage of headcount (without join, subquery and inner query)

SELECT

    SubBand,

    COUNT(\*) AS Headcount,

    COUNT(\*) \* 100.0 / (SELECT COUNT(\*) FROM Employees) AS Percentage\_Headcount

FROM

    Employees

GROUP BY

    SubBand;

-- Task 15: Find top 5 employees according to salary (without order by)

SELECT

    e1.EmployeeID, e1.Salary

FROM

    Employees e1

WHERE

    (SELECT COUNT(\*) FROM Employees e2 WHERE e2.Salary > e1.Salary) < 5;

-- Task 16: Swap value of two columns in a table without using third variable or a table

UPDATE TableName

SET ColumnA = ColumnA + ColumnB,

    ColumnB = ColumnA - ColumnB,

    ColumnA = ColumnA - ColumnB;

-- Task 17: Create a user, create a login for that user, provide permissions of DB\_owner to the user

CREATE LOGIN NewUserLogin WITH PASSWORD = 'StrongPassword123!';

CREATE USER NewUser FOR LOGIN NewUserLogin;

EXEC sp\_addrolemember 'db\_owner', 'NewUser';

-- Task 18: Find Weighted average cost of employees month on month in a BU

SELECT

    BU,

    Month,

    SUM(Cost \* Headcount) / SUM(Headcount) AS Weighted\_Average\_Cost

FROM

    BUEmployeeCosts

GROUP BY

    BU, Month;

-- Task 19: Calculate the amount of error (i.e.: actual - miscalculated average monthly salaries), and round it up to the next integer.

-- Actual average salary

WITH ActualAverage AS (

    SELECT AVG(Salary) AS ActualAvgSalary

    FROM Employees

),

-- Miscalculated average salary (removing zeroes from salaries)

MiscalculatedAverage AS (

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

    FROM Employees

)

-- Calculate the difference and round it up

SELECT

    CEILING(ActualAverage.ActualAvgSalary - MiscalculatedAverage.MiscalculatedAvgSalary) AS ErrorAmount

FROM

    ActualAverage, MiscalculatedAverage;

-- Task 20: Copy new data of one table to another (you do not have an indicator for new data and old data).

-- Assuming we have Table1 and Table2 with identical structure

-- We can copy the data from Table1 to Table2 by checking for non-existing records in Table2

INSERT INTO Table2 (Column1, Column2, Column3, ...)  -- specify all columns

SELECT t1.Column1, t1.Column2, t1.Column3, ...

FROM Table1 t1

LEFT JOIN Table2 t2 ON t1.PrimaryKeyColumn = t2.PrimaryKeyColumn

WHERE t2.PrimaryKeyColumn IS NULL;