```sql
-- Task 1
WITH ProjectGroups AS (
SELECT Task_ID,
        Start_Date,
        End_Date,
        ROW_NUMBER() OVER (ORDER BY Start_Date) - ROW_NUMBER() OVER (PARTITION BY
                Start_Date ORDER BY Task_ID) AS grp
   FROM Projects
)
SELECT MIN(Start_Date) AS Project_Start,
     MAX(End_Date) AS Project_End
FROM ProjectGroups
GROUP BY grp
ORDER BY DATEDIFF(day, MIN(Start_Date), MAX(End_Date)), MIN(Start_Date);

-- Task 2
SELECT S1.Name
FROM Students S1
JOIN Friends F ON S1.ID = F.ID
JOIN Packages P1 ON S1.ID = P1.ID
JOIN Packages P2 ON F.Friend_ID = P2.ID
WHERE P2.Salary > P1.Salary
ORDER BY P2.Salary;

-- Task 3
SELECT DISTINCT LEAST(X, Y) AS X, GREATEST(X, Y) AS Y
FROM Functions F1
JOIN Functions F2 ON F1.X = F2.Y AND F1.Y = F2.X ORDER BY
X, Y;

-- Task 4
WITH ContestStats AS (
   SELECT C.contest_id,
        C.hacker_id,
        C.name,
        COALESCE(SUM(V.total_views), 0) AS total_views,
        COALESCE(SUM(V.total_unique_views), 0) AS total_unique_views,
        COALESCE(SUM(S.total_submissions), 0) AS total_submissions,
        COALESCE(SUM(S.total_accepted_submissions), 0) AS total_accepted_submissions
   FROM Contests C
   LEFT JOIN Challenges H ON C.contest_id = H.contest_id
   LEFT JOIN View_Stats V ON H.challenge_id = V.challenge_id
   LEFT JOIN Submission_Stats S ON H.challenge_id = S.challenge_id
   GROUP BY C.contest_id, C.hacker_id, C.name
)
SELECT contest_id, hacker_id, name, total_views, total_unique_views, total_submissions,
   total_accepted_submissions
FROM ContestStats
WHERE total_views != 0 OR total_unique_views != 0 OR total_submissions != 0 OR
   total_accepted_submissions != 0
```

```sql
ORDER BY contest_id;

-- Task 5
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
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 RECURSIVE PrimeNumbers AS (
   SELECT 2 AS num
   UNION ALL
   SELECT num + 1
   FROM PrimeNumbers
   WHERE num < 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
   )
)
SELECT STRING_AGG(CAST(num AS VARCHAR), '&') AS primes
```

```sql
FROM PrimeFilter
OPTION (MAXRECURSION 0);

-- Task 8
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
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
WITH LeadManagerCount AS (
    SELECT company_code, COUNT(DISTINCT lead_manager_code) AS total_lead_managers
    FROM Lead_Manager
    GROUP BY company_code
),
SeniorManagerCount AS (
    SELECT company_code, COUNT(DISTINCT senior_manager_code) AS total_senior_managers
    FROM Senior_Manager
    GROUP BY company_code
),
ManagerCount AS (
    SELECT company_code, COUNT(DISTINCT manager_code) AS total_managers
    FROM Manager
    GROUP BY company_code
),
EmployeeCount AS (
```

```sql
    SELECT company_code, COUNT(DISTINCT employee_code) AS total_employees     FROM
Employee
    GROUP BY company_code
)
SELECT C.company_code,
       C.founder,
       COALESCE(LM.total_lead_managers, 0) AS total_lead_managers,
       COALESCE(SM.total_senior_managers, 0) AS total_senior_managers,
       COALESCE(M.total_managers, 0) AS total_managers,
       COALESCE(E.total_employees, 0) AS total_employees
FROM Company C
LEFT JOIN LeadManagerCount LM ON C.company_code = LM.company_code
LEFT JOIN SeniorManagerCount SM ON C.company_code = SM.company_code
LEFT JOIN ManagerCount M ON C.company_code = M.company_code
LEFT JOIN EmployeeCount E ON C.company_code = E.company_code ORDER
BY C.company_code;

-- Task 11
SELECT S1.Name
FROM Students S1
JOIN Friends F ON S1.ID = F.ID
JOIN Packages P1 ON S1.ID = P1.ID
JOIN Packages P2 ON F.Friend_ID = P2.ID
WHERE P2.Salary > P1.Salary
ORDER BY P2.Salary;

-- Task 12
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,
    (SUM(CASE WHEN Country = 'India' THEN Cost ELSE 0 END) / NULLIF(SUM(Cost), 0)) * 100 AS
        India_Percentage,
    (SUM(CASE WHEN Country = 'International' THEN Cost ELSE 0 END) / NULLIF(SUM(Cost),
        0)) * 100 AS International_Percentage
FROM YourTable
GROUP BY JobFamily;

-- Task 13
SELECT BU,
       MONTH,
       SUM(Cost) AS Total_Cost,
       SUM(Revenue) AS Total_Revenue,
       SUM(Cost) / NULLIF(SUM(Revenue), 0) AS Cost_Revenue_Ratio
FROM YourTable
GROUP BY BU, MONTH;

-- Task 14
SELECT SubBand,
       COUNT(EmployeeID) AS Headcount,
```

```sql
    (COUNT(EmployeeID) / (SELECT COUNT(*) FROM YourTable)) * 100 AS
Percentage_Headcount FROM YourTable
GROUP BY SubBand;

-- Task 15
SELECT TOP 5 *
FROM Employees
ORDER BY Salary DESC;

-- Task 16
UPDATE TableName
SET ColumnA = ColumnA + ColumnB,
   ColumnB = ColumnA - ColumnB,
   ColumnA = ColumnA - ColumnB;

-- Task 17
CREATE LOGIN new_user WITH PASSWORD = 'password';
CREATE USER new_user FOR LOGIN new_user;
EXEC sp_addrolemember 'db_owner', 'new_user';

-- Task 18
SELECT BU,
     AVG(Cost * Weight) / SUM(Weight) AS WeightedAvgCost
FROM Employees1
GROUP BY BU;

-- Task 19
WITH Actual AS (
   SELECT AVG(Salary) AS ActualAvgSalary
   FROM Employees ),
Miscalculated AS (
   SELECT AVG(CAST(REPLACE(CAST(Salary AS VARCHAR), '0', '') AS INT)) AS
         MiscalculatedAvgSalary
   FROM Employees
)
SELECT CEILING(Actual.ActualAvgSalary - Miscalculated.MiscalculatedAvgSalary) AS ErrorAmount
FROM Actual, Miscalculated;

-- Task 20
INSERT INTO TargetTable (KeyColumn, Column1, Column2)
SELECT KeyColumn, Column1, Column2
FROM SourceTable
WHERE NOT EXISTS (
   SELECT 1
   FROM TargetTable
   WHERE TargetTable.KeyColumn = SourceTable.KeyColumn
);
```