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;
```