Raj Singhaniya
0112cs221101@gmail.com

Level C Task

1. You are given a table, Projects, containing three columns: Task_ID, Start_Date and End_Date. It is guaranteed that the difference between the End_Date and the Start_Date is equal to 1 day for each row in the table.If the End_Date of the tasks are consecutive, then they are part of the same project. Samantha is interested in finding the total number of different projects completed. Write a query to output the start and end dates of projects listed by the number of days it took to complete the project in ascending order If there is more than one project that has the same number of completion days, then order by the start date of the project.

Ans: WITH RankedTasks AS (
SELECT
Task_ID,
Start_Date,
End_Date,
ROW_NUMBER() OVER (ORDER BY Start_Date) AS RowNum
FROM Projects
),
GroupedProjects AS (
SELECT
Task_ID,
Start_Date,
End_Date,
Start_Date - INTERVAL RowNum DAY AS ProjectGroup
FROM RankedTasks
)
SELECT

```
    MIN(Start_Date) AS ProjectStartDate,
    MAX(End_Date) AS ProjectEndDate,
        DATEDIFF(MAX(End_Date), MIN(Start_Date)) + 1 AS
ProjectDuration
FROM GroupedProjects
GROUP BY ProjectGroup
ORDER BY ProjectDuration ASC, ProjectStartDate;
```

2. You are given three tables: Students, Friends and Packages. Students contain two columns: ID and Name. Friends contains two columns: ID and Friend_ID (ID of the ONLY best friend). Packages contain two columns: ID and Salary (offered salary in $ thousands per month). Write a query to output the names of those students whose best friends got offered a higher salary than them. Names must be ordered by the salary amount offered to the best friends. It is guaranteed that no two students got same salary offer.

```
Ans: 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 p1.Salary < p2.Salary
ORDER BY p2.Salary;
```

3. You are given a table, Functions, containing two columns: X and Y. Two pairs (XI, Y1) and (Xz, Y2) are said to be symmetric pairs if Xr = Ya and X2 = Yr. Write a query to output all such symmetric pairs in ascending order by the value of X.

```
Ans: SELECT 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;
```

4. Samantha interviews many candidates from different colleges using coding challenges and contests. Write a query to print the contest_id, hacker_id, name, and the sums of total_submissions, total_accepted_submissions, total_views, and total_unique_views for each contest sorted by contest_id. Exclude the contest from the result if all four sums are.

Note: A specific contest can be used to screen candidates at more than one college, but each college only holds a screening contest. The following tables hold interview data:

Contests: The contest_id is the id of the contest, hacker_id is the id of the hacker who created the contest, and name is the name of the hacker.

Colleges: The college_id is the id of the college, and contest_id is the id of the contest that Samantha used to screen the candidates.

Challenges: The challenge_id is the id of the challenge that belongs to one of the contests whose contest_id Samantha forgot, and college_id is the id of the college where the challenge was given to candidates.

View_Stats: The challenge_id is the id of the challenge, total_views is the number of times the challenge was viewed by candidates, and total_unique_views is the number of times the challenge was viewed by unique candidates.

Submission_Stats: The challenge_id is the id of the challenge, total_submissions is the number of submissions for the challenge, and total_accepted_submission is the number of submissions that achieved full scores.

Ans:
```
SELECT
    c.contest_id,
    c.hacker_id,
    c.name,
COALESCE(SUM(ss.total_submissions), 0) AS total_submissions,
COALESCE(SUM(ss.total_accepted_submissions),0) AS
total_accepted_submissions,
```

```
COALESCE(SUM(vs.total_views), 0) AS total_views,
COALESCE(SUM(vs.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 Submission_Stats ss ON ch.challenge_id =
ss.challenge_id
LEFT JOIN View_Stats vs ON ch.challenge_id = vs.challenge_id
GROUP BY c.contest_id, c.hacker_id, c.name HAVING
SUM(ss.total_submissions) > 0  OR
SUM(ss.total_accepted_submissions) > 0
OR SUM(vs.total_views) > 0
OR SUM(vs.total_unique_views) > 0
ORDER BY c.contest_id;
```

5. Julia conducted a day of learning SQL contests. The start date of the contest was March 01, 2016 and the end date was March 15, 2016. Write a query to print the total number of unique hackers who made at least a submission each day (starting on the first day of the contest), and find the hacker_id and name of the hacker who made the maximum number of submissions each day. If more than one such hacker has a maximum number of submissions, print the lowest hacker_id. The query should print this information for each day of the contest, sorted by the date.
The following tables hold contest data:
Hackers: The hacker_id is the id of the hacker, and name is the name of the hacker.
Submissions: The submission_date is the date of the submission, submission_id is the id of the submission, hacker_id is the id of the hacker who made the submission, and score is the score of the submission.
Ans:
```
WITH dates_hackers AS (
SELECT submission_date, hacker_id, COUNT(*) AS subs FROM
Submissions
```

```
GROUP BY submission_date, hacker_id),
day1_hackers AS (
    SELECT DISTINCT hacker_id
    FROM Submissions
    WHERE submission_date = '2016-03-01'),
min_max AS (
    SELECT hacker_id, MIN(submission_date) AS first_sub,
MAX(submission_date) AS last_sub,
        DATEDIFF(MAX(submission_date), MIN(submission_date))
AS diff
    FROM Submissions
    GROUP BY hacker_id),
cumuCountLag AS (
    SELECT hacker_id, submission_date,
        LAG(submission_date) OVER (PARTITION BY hacker_id
ORDER BY submission_date) AS prevDay,
        COUNT(*) OVER (PARTITION BY hacker_id ORDER BY
submission_date) AS cumuCount FROM dates_hackers),
p1_interm AS (
    SELECT cumuCountLag.hacker_id,
cumuCountLag.submission_date, prevDay, first_sub, cumuCount,
        CASE WHEN (cumuCountLag.hacker_id IN (SELECT *
FROM day1_hackers))
AND (cumuCount = DATEDIFF(cumuCountLag.submission_date,
first_sub) + 1)
            THEN 1 ELSE 0 END AS partic_till FROM
cumuCountLag
LEFT JOIN min_max ON cumuCountLag.hacker_id =
min_max.hacker_id),
p1 AS (
    SELECT submission_date, SUM(partic_till) AS no_hackers_till
    FROM p1_interm
    GROUP BY submission_date),
topHackers AS (
```

```
    SELECT dates_hackers.submission_date,
dates_hackers.hacker_id, h.name, dates_hackers.subs,
           RANK() OVER (PARTITION BY
dates_hackers.submission_date ORDER BY dates_hackers.subs
DESC, dates_hackers.hacker_id) AS sRank
    FROM dates_hackers
    JOIN Hackers h ON dates_hackers.hacker_id = h.hacker_id),
p2 AS (
    SELECT submission_date, hacker_id, name
    FROM topHackers
    WHERE sRank = 1)
SELECT p1.submission_date, p1.no_hackers_till, p2.hacker_id,
p2.name
FROM p1
JOIN p2 ON p1.submission_date = p2.submission_date
ORDER BY p1.submission_date;
```

6. Consider P1(a,b) and P2(c,d) to be two points on a 2D plane.

- **happens to equal the minimum value in Northern Latitude (LAT_N in STATION).**
- **happens to equal the minimum value in Western Longitude (LONG_W in STATION).**
- **happens to equal the maximum value in Northern Latitude (LAT_N in STATION).**
- **happens to equal the maximum value in Western Longitude (LONG_W in STATION).**

Query the Manhattan Distance between points P1 and P2 and round it to a scale of decimal places.

Ans:
```
SELECT ROUND(
    ABS(MAX(LAT_N) - MIN(LAT_N)) + ABS(MAX(LONG_W) -
MIN(LONG_W)),
    4
) AS manhattan_distance
FROM STATION;
```

7. Write a query to print all prime numbers less than or equal to 1000. Print your result on a single line, and use the ampersand (&) character as your separator (instead of a space).
For example, the output for all prime numbers <= 10 would be:
Ans:
```
WITH RECURSIVE Numbers AS (
    SELECT 2 AS num
    UNION ALL
    SELECT num + 1
    FROM Numbers
    WHERE num < 1000
)
SELECT GROUP_CONCAT(num SEPARATOR '&') AS
prime_numbers
FROM Numbers
WHERE num > 1
  AND NOT EXISTS (
      SELECT 1
      FROM Numbers AS n2
      WHERE n2.num > 1
      AND n2.num < Numbers.num
      AND Numbers.num % n2.num = 0
  );
```

8. Task 8. Pivot the Occupation column in OCCUPATIONS so that each Name is sorted alphabetically and displayed underneath its corresponding Occupation. The output column headers should be Doctor, Professor, Singer, and Actor, respectively.
Note: Print NULL when there are no more names corresponding to an occupation.
Ans:
```
WITH RankedOccupations AS (
    SELECT
        Name,
        Occupation,
```

```
ROW_NUMBER() OVER (PARTITION BY Occupation ORDER BY
Name) AS RowNum FROM OCCUPATIONS
)
SELECT
   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 RankedOccupations
GROUP BY RowNum
ORDER BY RowNum;
```

9. You are given a table, BST, containing two columns: Nand P, where N represents the value of a node in Binary Tree, and P is the parent of N. Write a query to find the node type of Binary Tree ordered by the value of the node. Output one of the following for each node:
. Root: If node is root node.
. Leaf: If node is leaf node.
· Inner: If node is neither root nor leaf node.

Ans:
```
SELECT N,
      CASE
         WHEN P IS NULL THEN 'Root'
         WHEN N IN (SELECT DISTINCT P FROM BST) THEN
'Inner'
      ELSE 'Leaf'
      END AS NodeType
FROM BST
ORDER BY N;
```

10. Amber's conglomerate corporation just acquired some new companies. Each of the Founder companies follows this hierarchy:
Given the table schemas below, write a query to print the company_code, founder name, total number of lead managers, total number of senior managers, total number of managers, and total number of employees. Order your output by ascending company_code.
Note:
. The tables may contain duplicate records.
. The company_code is string, so the sorting should not be numeric. For example, if the company_codes are C_1, C_2, and C_10, then the ascending company_codes will be C_1, C_10, and C_2.
The following tables contain company data:
. Company: The company_code is the code of the company and the founder is the founder of the company.
. Lead_Manager: The lead_manager_code is the code of the lead manager, and the company_code is the code of the working company.
. Senior_Manager: The senior_manager_code is the code of the senior manager, the lead_manager_code is the code of its lead manager, and the company_code is the code of the working company.
. Manager: The manager_code is the code of the manager, the senior_manager_code is the code of its senior manager, the lead_manager_code is the code of its lead manager, and the company_code is the code of the working company.
. Employee: The employee_code is the code of the employee, the manager_code is the code of its
manager, the senior_manager_code is the code of its senior manager, the lead_manager_code is the working company.
Ans: SELECT
    c.company_code,
    c.founder,

```
    COUNT(DISTINCT lm.lead_manager_code) AS
lead_manager_count,
    COUNT(DISTINCT sm.senior_manager_code) AS
senior_manager_count,
    COUNT(DISTINCT m.manager_code) AS manager_count,
    COUNT(DISTINCT e.employee_code) AS employee_count
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;
```

11. You are given three tables: Students, Friends and Packages. Students contains two columns: ID and Name. Friends contains two columns: ID and Friend_ID (ID of the ONLY best friend). Packages contains two columns: ID and Salary (offered salary in $ thousands per month). Write a query to output the names of those students whose best friends got offered a higher salary than them. Names must be ordered by

the salary amount offered to the best friends. It is guaranteed that no two students got same salary offer.

Ans:
```
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;
```

12. Display ratio of cost of job family in percentage by India and international (refer simulation data).

Ans:
```
WITH IndiaCosts AS (
    SELECT job_family, SUM(ctc) AS total_ctc
    FROM your_table
    WHERE country = 'India'
    GROUP BY job_family
),
IntlCosts AS (
    SELECT job_family, country, SUM(ctc) AS total_ctc
    FROM your_table
    WHERE country != 'India'
    GROUP BY job_family, country
)
SELECT i.job_family,
       i.country AS india_country,
       i.total_ctc AS india_ctc,
       it.country AS intl_country,
       it.total_ctc AS intl_ctc,
       ROUND(it.total_ctc / i.total_ctc, 2) AS cost_ratio
FROM IndiaCosts i
JOIN IntlCosts it ON i.job_family = it.job_family
ORDER BY i.job_family, cost_ratio;
```

13. Find ratio of cost and revenue of a BU month on month.

Ans: Assuming you have a table named `financial_data` structured like this:

```
finance_data (
    business_unit VARCHAR,
    cost DECIMAL,
    revenue DECIMAL,
    date DATE
)
```

SQL query to calculate the **monthly cost-to-revenue ratio** per BU:

```
SELECT
    business_unit,
    DATE_TRUNC('month', date) AS month,
    SUM(cost) AS total_cost,
    SUM(revenue) AS total_revenue,
    CASE
        WHEN SUM(revenue) = 0 THEN NULL
        ELSE ROUND(SUM(cost)::NUMERIC /
NULLIF(SUM(revenue), 0), 4)
    END AS cost_to_revenue_ratio
FROM
    finance_data
GROUP BY
    business_unit,
    DATE_TRUNC('month', date)
ORDER BY
    business_unit month;
```

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

Ans:

```
SELECT
    sub_band,
    COUNT(*) AS headcount,
    ROUND(100.0 * COUNT(*) / SUM(COUNT(*)) OVER (), 2) AS
percentage
FROM
```

```
    employees
GROUP BY
    sub_band
ORDER BY
    sub_band;
```

15. Find top 5 employees according to salary (without order by).

Ans:
```
SELECT *
FROM (
    SELECT *,
           RANK() OVER (ORDER BY salary DESC) AS rnk
    FROM employees
) t
WHERE rnk <= 5;
```

16. Swap value of two columns in a table without using a third variable or a table.

Ans: For numerical values:
```
UPDATE my_table
SET col1 = col1 + col2,
    col2 = col1 - col2,
    col1 = col1 - col2;
```

For text/String:
```
UPDATE my_table
SET col1 = col1 || col2,
col2 = SUBSTRING(col1 FROM 1 FOR LENGTH(col1) -
LENGTH(col2)),
col1 = SUBSTRING(col1 FROM LENGTH(col2) + 1);
```

17. Create a user,create a login for that user provide permissions of DB_owner to the user.

Ans:**Creating a SQL Server login**
```
CREATE LOGIN Raj WITH PASSWORD = 'Raj@12345';
```

```
GO
```
Using the student database
```
USE student;
GO
```

Creating a database user for the login
```
CREATE USER Raj_user FOR LOGIN Raj_login;
GO
```
Grant db_owner role to the user
```
EXEC sp_addrolemember 'db_owner', 'Raj_user';
GO
```

18. Find Weighted average cost of employees month on month in a BU.

Ans: Assuming:
```
employee_costs (
    employee_id      INT,
    business_unit    VARCHAR,
    cost             DECIMAL,
    month            DATE,     -- or a TIMESTAMP
    weight_factor    DECIMAL   -- e.g., hours worked, FTE %, etc.
)
```

SQL Query to Get Weighted Average Cost Per BU Per Month:
```
SELECT
    business_unit,
    DATE_TRUNC('month', month) AS month,
    ROUND(
    SUM(cost * weight_factor) / NULLIF(SUM(weight_factor), 0),
        2
    ) AS weighted_avg_cost
   FROM
   employee_costs
   GROUP BY
    business_unit,
```

```
    DATE_TRUNC('month', month)
ORDER BY
    business_unit,
    Month;
```

19.Samantha was tasked with calculating the average monthly salaries for all employees in the EMPLOYEES table, but did not realize her keyboard's 0 key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeroes removed), and the actual average salary. Write a query calculating the amount of error (i.e .: actual -miscalculated average monthly salaries), and round it up to the next integer.

Ans:
```
SELECT
    CEIL(
        AVG(salary) - AVG(CAST(REPLACE(CAST(salary AS CHAR), '0', '') AS UNSIGNED))
    ) AS error_amount
FROM
    EMPLOYEES;
```

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

Ans:
```
INSERT INTO destination_table (col1, col2, col3, ...)
SELECT col1, col2, col3, ...
FROM source_table s
WHERE NOT EXISTS (
    SELECT 1 FROM destination_table d
    WHERE d.unique_key = s.unique_key
);
```