SQL- Intermediate (Part I)

Weather Observation Station 20: (1.)

https://www.hackerrank.com/challenges/weather-observation-station-20/problem?isFullScreen=true

```
SELECT ROUND (AVG (LAT_N),4)
FROM (
      SELECT    LAT_N,
          ROW_NUMBER ( )    OVER  (ORDER BY LAT_N) As RowNum
          COUNT (*)   OVER( )   As   Total Rows
     FROM STATION
    ) Subquery
WHERE  RowNum  IN  (Total Rows/2 , Total Rows/2 + 1 , Total Rows + 1/2)
```

round to the 4 digits

Assign sequential number to each row

order by lat-N

$\frac{\text{Total Rows}}{2}$ = ②, $\frac{\text{Total Rows}}{2}+1$ = ③

suppose the no of rows = 4

1
2 } → average of these two i.e $\frac{\text{total rows}}{2}$ and next one: $\frac{\text{total rows}}{2}+1$
3 }
4

$\frac{\text{Total Rows}+1}{2}$ = ③

suppose no of rows = 5

middle row ⇒ 1, 2, ③ → middle row = $\frac{(n+1)}{2}$, 4, 5

Weather Observation Station 5 (2.)

https://www.hackerrank.com/challenges/weather-observation-station-5/problem?isFullScreen=true

```
SELECT   CITY,   LENGTH(CITY)   FROM   STATION
ORDER  BY   LENGTH (CITY) DESC,  CITY  ASC   LIMIT 1;
```

length of city

ascending order city

ordered in descending order (longest city)

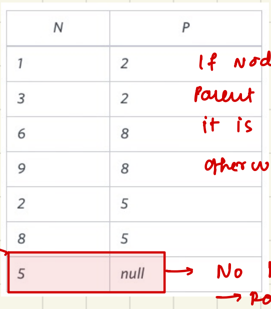
```
SELECT   CITY,   LENGTH (CITY)  FROM   STATION
ORDER   BY   LENGTH (CITY)  ASC,  CITY  ASC   LIMIT 1;
```

ordered in ascending order (shortest city name)

Binary Tree Nodes

https://www.hackerrank.com/challenges/binary-search-tree-1/problem?isFullScreen=true

3.



N	P
1	2
3	2
6	8
9	8
2	5
8	5
5	null

If node is not Parent then it is Leaf otherwise 'Inner'

→ No Parent → Root

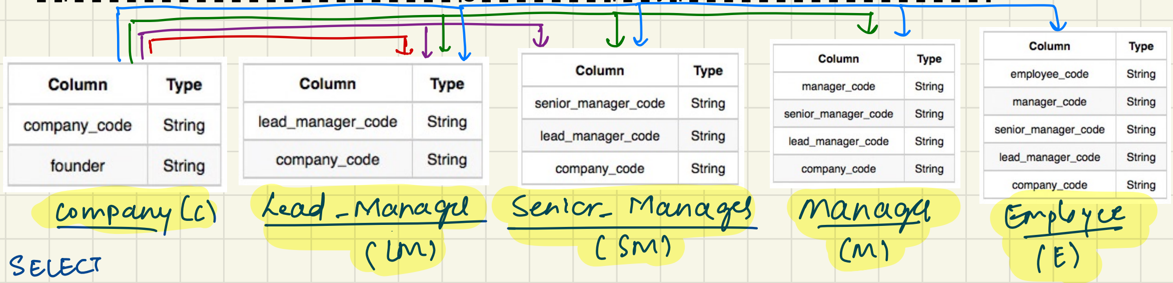
```
SELECT N,
       CASE
           WHEN P IS NULL THEN 'Root'
           WHEN N IN (SELECT DISTINCT P FROM BST) THEN 'Inner'
           ELSE 'Leaf'
       END AS NodeType
FROM BST
ORDER BY N;
```

see if node is parent

NodeType

New Companies

https://www.hackerrank.com/challenges/the-company/problem?isFullScreen=true



Company (C)

Column	Type
company_code	String
founder	String

Lead_Manager (LM)

Column	Type
lead_manager_code	String
company_code	String

Senior_Manager (SM)

Column	Type
senior_manager_code	String
lead_manager_code	String
company_code	String

Manager (M)

Column	Type
manager_code	String
senior_manager_code	String
lead_manager_code	String
company_code	String

Employee (E)

Column	Type
employee_code	String
manager_code	String
senior_manager_code	String
lead_manager_code	String
company_code	String

```
SELECT
    C.company_code, C.founder
    COUNT (DISTINCT LM.lead_manager_code) AS total_lead_managers,
    COUNT (DISTINCT SM.senior_manager_code) AS total_senior_managers,
    COUNT (DISTINCT M.manager_code) AS total_managers,
    COUNT (DISTINCT E.employee_code) AS total_employees,
  FROM company C
LEFT JOIN Lead_Manager LM ON C.company_code = LM.company_code
LEFT JOIN Senior_Manager SM ON
        LM.lead_manager_code = SM.lead_manager_code
        and C.company_code = SM.company_code
LEFT JOIN Manager M ON
      SM.senior_manager_code = M.senior_manager_code
      and LM.lead_manager_code = M.lead_manager_code
      and C.company_code = M.company_code
LEFT JOIN Employee E ON
      M.manager_code = E.manager_code
      SM.senior_manager_code = E.senior_manager_code
      LM.lead_manager_code = E.lead_manager_code
      C.company_code = E.company_code

GROUP BY C.company_code, C.founder
ORDER BY C.company_code;
```

→ grouping by campany code and founder

order in ascending order

Top Competitors

https://www.hackerrank.com/challenges/full-score/problem?isFullScreen=true

```
SELECT   h. hacker-id,   h.name
FROM     Hackers . h
JOIN     (
         SELECT   s. hacker-id  , COUNT (*)  as  full-score-count
         FROM     Submissions   s
         JOIN Challenges   c    on   s. challenge-id = c.challenge-id .
         JOIN Difficulty   d    ON   c. difficulty-level = d.difficulty-level
         WHERE    s.score =     d. score → to ensure highest score for the challenge to consider
         GROUP BY     s. hacker-id  ⤳ group by hackers
         HAVING     COUNT (*) > 1  ⤳ having
) as   fs   ON   h. hacker-id   = fs. hacker-id
ORDER   BY    fs. full-score-count  DESC ,  h. hacker-id  ASC
```

$\underbrace{\text{fs. full-score-count DESC}}_{\text{descending order by total no. of challenges in which hacker got full score}}$, $\underbrace{\text{h. hacker-id ASC}}_{\text{sort them in ascending hacker-id}}$

More concise and clear without using subquery.

```
SELECT    h. hacker-id,    h. name    FROM    Hackers h
JOIN      Submissions  s    ON   h. hacker-id   =  s. hacker-id
JOIN      Challenges   c    ON   s. challenge-id =  c. challenge-id
JOIN      Difficulty   d    ON   c. difficulty-level =  d. difficulty level
WHERE
     s. score  =  d. score  ⤳ filtering out the rows
GROUP  BY
     h. hacker-id, h. name  ⤳ group by people
HAVING
     COUNT  (s. submission-id) >1
ORDER  BY
     COUNT  (s. submission-id)   DESC,  } → same as above
          h. hacker-id   ASC;
```