Nha Do
Data/AI Engineer at AT&T Inc - Chief Data Office

This note is a list of Leetcode problems that I have solved. As a Data/ AI engineer, I am using Python and SQL for my daily work, therefore, they are the main languages I use. Some questions were solved by C++ when I was in college will also be included.

My solutions are not the most optimal ones in term of Big-O. I am just trying to organize my work and present some different approaches and functions that might be helpful. This work is in progress and I will keep it updated over time. Database questions will be my main focus at this moment.

With that being said, there is a lot of room for optimization and code improvement. I am very happy to receive any feedback, comment on how to make it better or any related thing.

Any typo, comment or suggestion please email to nhado401@gmail.com.

Last Updated: 01/2025

Database

Question 1: Rising Temperature - Easy

Problem:

Given table **Weather** that has 3 columns: id (int), recordDate(date), temperature (int).
id is the column with unique values for this table.
There are no different rows with the same recordDate.
This table contains information about the temperature on a certain day.

Column Name	Type
id	int
recordDate	date
temperature	int

The question asks to write a solution to find all dates' id with higher temperatures compared to its previous dates (yesterday). Return the result table in any order.

There are cases when the date is not continuous. If yesterday's data is missing, then we will not consider it. For example, there is 2024-01-10 but not 2024-01-09, in that case, we shouldn't compare temperature on 2024-01-10 with 2024-01-08 (if available).

Approach:

I have 2 approaches on this problem written in PostgreSQL:

1. The better way is to avoid any self join or window functions, therefore running time is improved. I checked the existence of all required conditions in WHERE clause, which means if the temperature of a certain date is higher than the temperature of that date - 1 (implies of yesterday's data), I will select that id.

2. Another solution is to use LAG window function. It does not give a good running time in this question but it's interesting to see how LEAD/LAG function works since it might be helpful for data analysis. I constructed a temporary table with 2 new columns called prevTemp and prevDate so that each row will have data of previous row about date and temperature. I will select id that has temperature $>$ prevTemp and recordDate - prevDate $= 1$ (to avoid situation where yesterday's data is missing)

Solution:

1.
```
Select today.id as Id
From Weather today
WHERE Exists (
    Select 1
    From Weather yesterday
    WHERE today.temperature > yesterday.temperature
    AND today.recordDate - yesterday.recordDate = 1
)
```

2.
```
Select temp.id as Id
From (
    Select id,
           temperature,
           recordDate,
           LAG(temperature, 1) OVER (ORDER BY recordDate) as prevTemp,
           LAG(recordDate) OVER (ORDER BY recordDate) as prevDate
    From Weather ) temp
WHERE temp.temperature > temp.prevTemp
AND temp.recordDate - temp.prevDate = 1
```

Question 2: Second Highest Salary - Medium

Problem:

Given table **Employee** that has 2 columns: id (int), salary (int).
id is the primary key (column with unique values) for this table.
Each row of this table contains information about the salary of an employee.

Column Name	Type
id	int
salary	int

The question asks to write a solution to find the second highest distinct salary from the Employee table. If there is no second highest salary, return null.

Approach:

Ranking question could be easily solved by window functions such as ROW_NUMBER(),

RANK() or DENSE_RANK(). However, since there are cases when many employees have the same salary but we're only interested in distinct value, DENSE_RANK() will be the best candidate.

In my approach, I used DENSE_RANK() to ensure a continuous and unbroken sequence of ranks so that if I set the condition to pick up row = 2, it's always the second highest row.

The last thing to take care of is the NULL case when the query does not return anything. For this, I use COALESCE function in PostgreSQL to return NULL value in the absence of second highest salary.

Solution:

```
Select
COALESCE((Select temp.salary
        FROM (
            Select distinct(salary) as salary,
                   DENSE_RANK() OVER (Order by salary desc) as ro_num
            From Employee ) temp
        WHERE ro_num = 2), NULL) as SecondHighestSalary
```

Question 3: Employee Bonus - Easy

Problem:

Given table **Employee** as below:

Column Name	Type
empID	int
name	varchar
supervisor	int
salary	int

empId is the column with unique values for this table.
Each row of this table indicates the name and the ID of an employee in addition to their salary and the id of their manager.

And table **Bonus** as below:

Column Name	Type
empID	int
bonus	int

empId is the column of unique values for this table.
empId is a foreign key (reference column) to empId from the Employee table.
Each row of this table contains the id of an employee and their respective bonus.

The question asks to write a solution to report the name and bonus amount of each employee with a bonus less than 1000.
Return the result table in any order.
Note that even if the bonus is NULL, we will also consider and return the corresponding name.

Approach:

This question is very straight forward, we can simply do the LEFT JOIN between Employee and Bonus tables with the filter of bonus $<$ 1000 or bonus is NULL.

However, the reason why I still want to discuss this question is because it involves to LEFT JOIN function, which is a very practical function in my day-to-day data analytics. My work requires me to dive deeply into many data sources and tables and perform the comparison to understand them. If we want to spot out the discrepancy between 2 tables, for instance, any records only available in table A but not table B, LEFT JOIN would be very helpful to tackle this task. Using LEFT JOIN combines with WHERE clause in which some columns in table B is NULL will resolve it.

In addition, LEFT JOIN will preserve the case table (root table) when we have to JOIN multiple tables together.

Solution:

```
Select e.name as name, b.bonus as bonus
From Employee e
LEFT JOIN Bonus b
    ON e.empId = b.empId
WHERE b.bonus is NULL
OR b.bonus < 1000
```

Algorithm

Question 1: Missing Number - Easy

Problem:

Given an array **nums** containing n distinct numbers in the range [0, n], return the only number in the range that is missing from the array. All the numbers of nums are unique.
Example:
Input nums = [9,6,4,2,3,5,7,0,1]
Output 8
Explanation: n = 9 since there are 9 numbers, so all numbers are in the range [0,9]. 8 is the missing number in the range since it does not appear in nums.

Approach:

The key point here is all numbers are unique so instead of using multiple for-loop, which will cause the horrible Big-O, we can use a math formula to calculate the sum of n natural integer and subtract it continuously to each array's element. The result of that subtraction is the missing number. The runtime complexity will be only O(n).

$$sum = \frac{n * (n + 1)}{2} \tag{1}$$

Solution:

```
class Solution(object):
    def missingNumber(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        n = len(nums)

        #Set current missing_num equals to the total sum of n numbers
        missing_num = n*(n+1)/2

        for i in range(n):
            #Subtract total sum (missing_num) by each value in given array
            missing_num -= nums[i]

        return int(missing_num)
```

Question 2: Longest Common Prefix - Easy

Problem:

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

Example 1:
Input: strs = ["flower","flow","flight"]
Output: "fl"

Example 2:
Input: strs = ["dog","racecar","car"]
Output: "" (because there is no common prefix among the input strings.)

Note that we only care about PREFIX, if the common words are not prefix, we will not consider and our code should ignore that scenario.

Approach:

My solution uses nested for loop and my strategy is as below:

1. Start with the first letter in the first element of the array. Capture both letter and index.
2. Loop through second element of the array (second word) to the end of the array (last word). Check from each word if the corresponding character at that index equals to letter from step 1. If it does, increase count by 1. Else, return the output immediately because it will no longer have any common word.
3. After we exit from the nested loop, we will check if the count = length of the array - 1 meaning that the letter in step 1 presents in all other words in the array (minus 1 because we already take the first word as reference, so the length of the array should be decreased by 1). If that condition is met, then we find the common character between words in array so we add that into output.
4. Set count = 0 again to start the loop over for the next character.

Note that I take the first word as a reference and force it starts from the first character to the end of the word length and break the loop right away when I encounter the mismatch to make sure I only check for the prefix.

I also use try-except block to avoid all possible error because there could be cases when the length of the first word is longer than other words in the array. Without the try-except block, it will get error of index falls outside of the range.

Solution:

```
class Solution(object):
    def longestCommonPrefix(self, strs):
        """
        :type strs: List[str]
        :rtype: str
        """
        output = ""
        count = 0

        for index, c in enumerate(strs[0]):
            for i in range(1,len(strs)):
```

```
            try:
                if c == strs[i][index]:
                    count+=1
                else:
                    return output
            except:
                pass

        if count == len(strs) - 1:
            output+=c

        count = 0

    return output
```