## **Question 1:**

Will this code work? Provide reasoning.

import numpy as np

a = np.array([[1,2],[1,2],[1,2]])

b = np.array([[1,2],[1,2],[1,2]])

c = np.matmul(a,b)

**Your answer:**

## **Question 2:**

Given a string s, find the first non-repeating character and return its index. If it doesn’t exist, return -1.

**# Example 1:**

AAGB

Answer: 2

**# Example 2:**

AGBA

Answer: 1

**# Example 3:**

ABCCBA

Answer: -1

**Your Answer:**

def first\_unique\_character(s: str) -> int:

co = dict()

for l in s:

if l is co:

co[l] += 1

else:

col[l] = 1

for i range(len(s)):

if co[s[i]] == 1:

return i

return -1

Time O(n)

Space O(n)

## **Question 3:**

| **Column Name** | **Type** |
| --- | --- |
| id | int |
| student\_name | varchar |
| marks | int |

Table Name - **students**

Find the ID of all the students who scored average marks in the exam.

**Your answer:**

avg = df.marks.mean()

df[df[‘marks’] == avg]][‘id’]

## **Question 4:**

| **Column Name** | **Type** |
| --- | --- |
| id | int |
| company\_name | varchar |
| profit | int |
| categoryId | int |
| Table Name - **Company** | |

| **Column Name** | **Type** |
| --- | --- |
| id | int |
| category\_name | varchar |
| Table Name - **Category** | |

The company table contains a list of companies and their profits. The column **categoryId** is a foreign key to the table Category.

The category table is a list of categories a company belongs to. Write an SQL query to show all the companies with the highest profits in every category. if multiple companies have the same profits then show all of them.

| **Column Name** | **Type** |
| --- | --- |
| Category | varchar |
| Company Name | varchar |
| profit | int |
| **Output Format** | |

**Your answer:**

max\_ids = comp\_df.group\_by(‘categoryId’)[max(comp\_df[‘profit’])][‘id’]

SELECT Category.category\_name, MAX(profit) FROM (SELECT \* FROM Company, Category WHERE Company.categoryId = Category.id) GROUP BY Company.categoryId

## **Question 5:**

You are managing a cloud computing platform, and your goal is to allocate resources efficiently based on the computational needs of different tasks. Each task requires a certain amount of computing resources, represented by an integer value. The available computing resources are given in an array.

Write a function or method to determine the minimum number of resource allocations needed to fulfill the computational needs of a given set of tasks. If it's not possible to allocate the necessary resources for all tasks, return -1.

**# Example 1:**

resources = [2, 4, 8,]

total\_needed = 12

print(min\_resource\_allocations(resources, total\_needed))

# Output: 2 (Allocate 8 and 4 to meet the total needed resources)

**# Example 2:**

resources = [5,4]

total\_needed = 7

print(min\_resource\_allocations(resources, total\_needed))

# Output: -1 (Not enough resources to meet the total needed)

**# Example 3:**

resources = [1, 2, 3]

total\_needed = 6

print(min\_resource\_allocations(resources, total\_needed))

# Output: 2 (Allocate 3 and 3 to meet the total needed resources)

resources = [2, 6, 8,]

total\_needed = 12

12

2 6 8 - 1

(10 6 4)

(8 4 2) (4 0 x) (2 x x) - 2

cache = dict()

count = 0

def min\_resource\_allocations(resources, total\_needed, count):

if total\_needed == 0:

return count

results = []

for resource in resources:

result = total\_needed - resource

results.append(result)

result = min(results)

if result in cache:

return cache[result]

cache[result] = min\_resource\_allocations(resources, result])

return -1