**Datum Labs Assessment**

**Python Questions:**

**1.**

def fill\_none\_value(array): #function definition with array as #argument

  if len(array) == 0: # handling a case for length of array=0

    return f"List is empty"

  else:

    for i in range(len(array)):

      if(array[i] == None):

        array[i] = array[i-1] # if current element is None then #fill with the last element

      if (array[0] == None):

        array[i] = array[i+1]

    return array

array1 = [None,2,3,None,None] #function usage

fill\_none\_value(array1)

**Code output:**

**[2, 2, 3, 3, 3]**

**2.**

def split\_and\_return\_list(string1, string2): #function declaration #taking two strings as arguments.

    words1 = string1.split() #list

    words2 = string2.split() #list

    mismatched\_Words = []

    for i in range (len(words2)): #condition checking the case.

      if (words1[i]  != words2[i]):

        mismatched\_Words.append(words1[i])

    return f"{mismatched\_Words}"

# Example usage

input\_string = "This is a sample string" #defining string

input\_string1 = "THIS IS A sample string" #defining string

split\_and\_return\_list(input\_string, input\_string1) #function usage

**Code output:**

**['This', 'is', 'a']**

**3.**

def get\_char\_frequency(str): #function definition

  frequency = {} #deifining of dictionary

  for i in str: #increment each character in the string

    if i in frequency:

      frequency[i] += 1

    else:

      frequency[i] = 1 #if not present then make frequency to one.

  return f"Frequency of each character in list is {frequency}"

def get\_specific\_char\_frequency(str, char): #function for computing #frequency of specific character.

  count = 0

  for i in str:

    if i == char:

      count += 1

  return f"Count of {char} in string {str} is {count}"

my\_string = "This is This is usman usman"

get\_specific\_char\_frequency(my\_string,"i")

**Code ouput:**

**Count of i in string This is This is usman usman is 4**

**SQL Questions:**

**1.**

SELECT

customers\_with\_both\_paid,

total\_unique\_customers.unique\_customers,

ROUND((customers\_with\_both\_paid \* 100.0 / total\_unique\_customers.unique\_customers), 2) AS percentage

FROM (

SELECT

COUNT(DISTINCT cp1.customer\_id) AS customers\_with\_both\_paid

FROM

CustomerPurchases AS cp1

WHERE

cp1.product\_id = 'ProductA'

AND cp1.payment\_status = 'Paid'

AND EXISTS (

SELECT 1

FROM CustomerPurchases AS cp2

WHERE cp1.customer\_id = cp2.customer\_id

AND cp2.product\_id = 'ProductB'

AND cp2.payment\_status = 'Paid'

)

) AS customers\_with\_paid\_number,

(

SELECT

COUNT(DISTINCT customer\_id) AS unique\_customers

FROM

CustomerPurchases

) AS total\_unique\_customers;

**Code output:**

****

**3.**

SELECT

cp.product\_id AS complementary\_product\_id,

COUNT(\*) AS frequency

FROM

CustomerPurchases cp

JOIN

CustomerPurchases cp\_a ON cp.customer\_id = cp\_a.customer\_id

WHERE

cp\_a.product\_id = 'ProductA'

AND cp.product\_id <> 'ProductA'

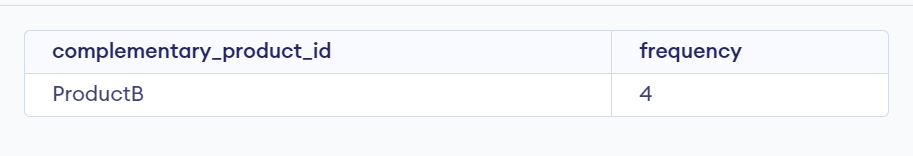
GROUP BY

cp.product\_id

ORDER BY

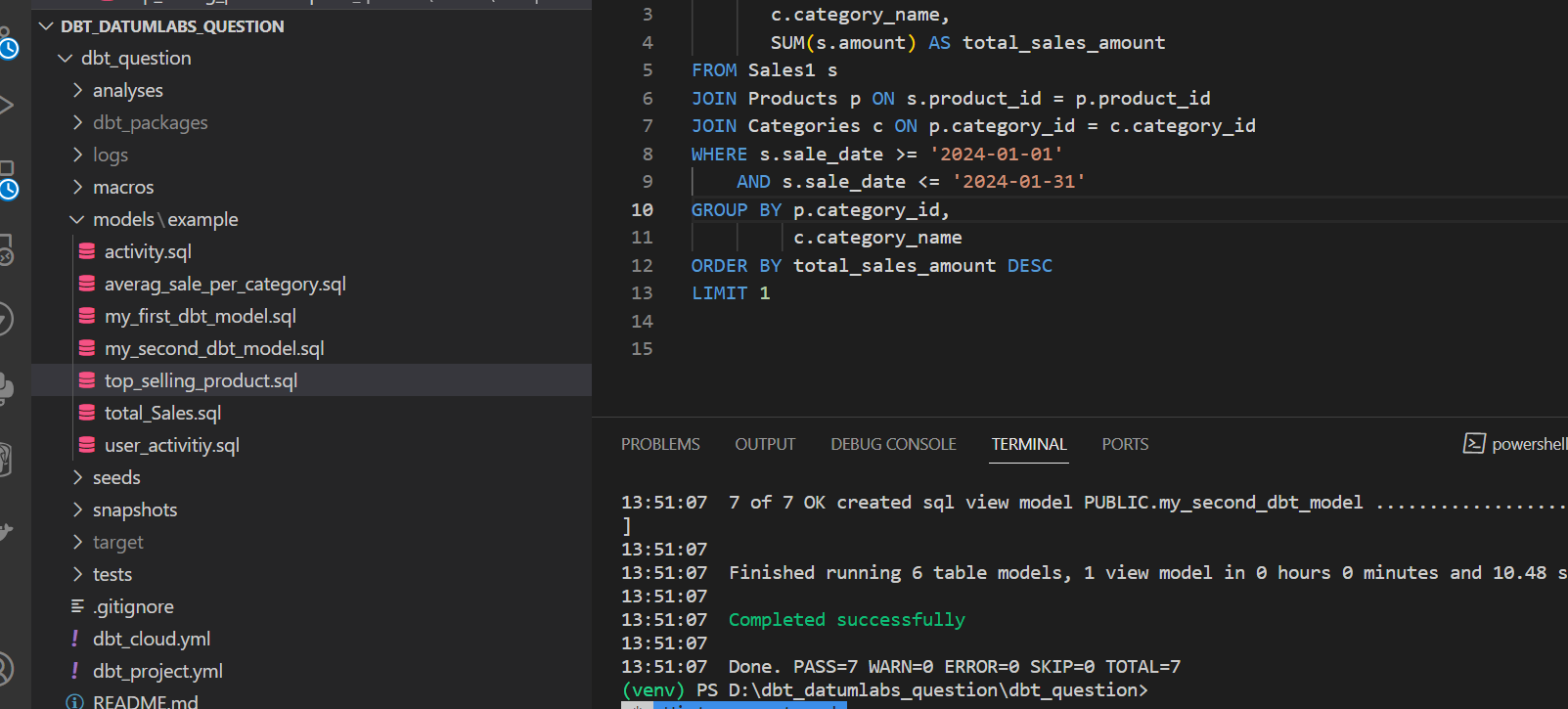
frequency DESC

LIMIT 5;

**Code output  
**

**Dbt Question:**

**The following is the screenshot of the overall environment.**

****

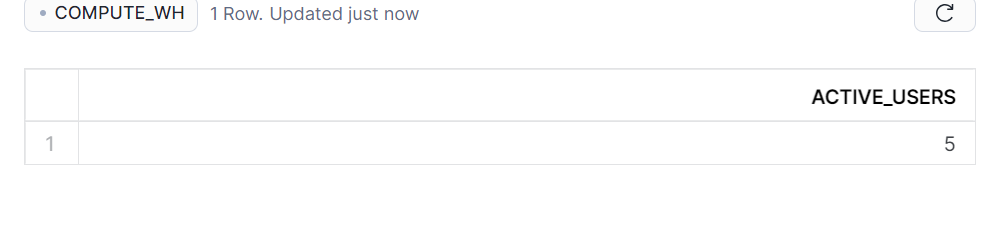
**1.**

{{ config(materialized='table') }}

SELECT COUNT(DISTINCT user\_id) AS active\_users

FROM USERACTIVITY

WHERE activity\_date >= '2024-01-01' and activity\_date <= '2024-01-30'

**Code output in the snowflake database  
**

**2.**

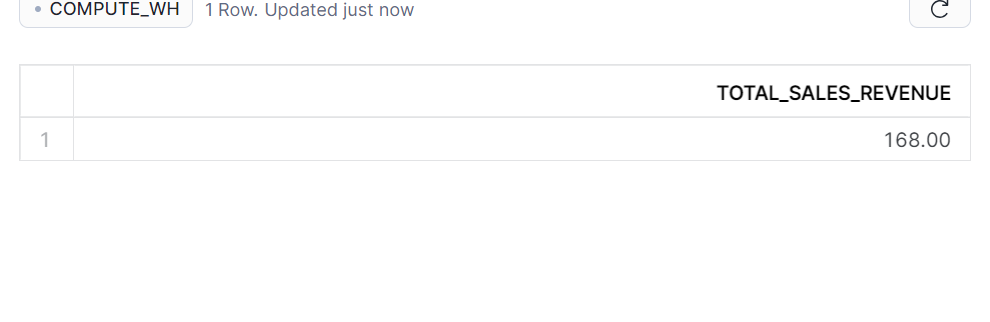
{{ config(materialized='table') }}

SELECT SUM(amount) AS total\_sales\_revenue

FROM Sales

WHERE sale\_date >= '2024-01-01'

    AND sale\_date <= '2024-01-31'

**Code output:  
**

**3.**

{{ config(materialized='table') }}

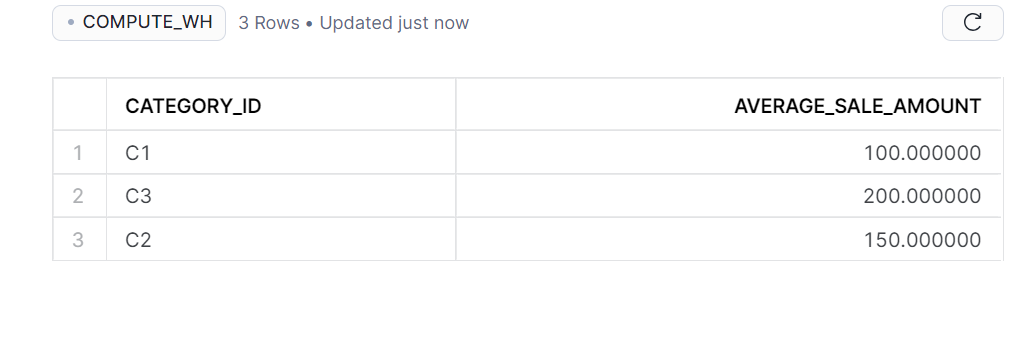
SELECT category\_id,

       AVG(amount) AS average\_sale\_amount

FROM SALES1

GROUP BY category\_id

**Code output:**

****

**4.**

{{ config(materialized='table') }}

SELECT user.user\_id,

       user.user\_name,

       activity.activity\_id,

       activity.activity\_date,

       user.join\_date

FROM USERACTIVITY activity

JOIN USERS user ON user.user\_id = activity.user\_id

WHERE user.join\_date >= '2024-01-01'

    AND user.join\_date <= '2024-01-31'

**Code output:  
No output as there were no users who joined in the January.**

**5.**

{{ config(materialized='table') }}

SELECT p.category\_id,

       c.category\_name,

       SUM(s.amount) AS total\_sales\_amount

FROM Sales1 s

JOIN Products p ON s.product\_id = p.product\_id

JOIN Categories c ON p.category\_id = c.category\_id

WHERE s.sale\_date >= '2024-01-01'

    AND s.sale\_date <= '2024-01-31'

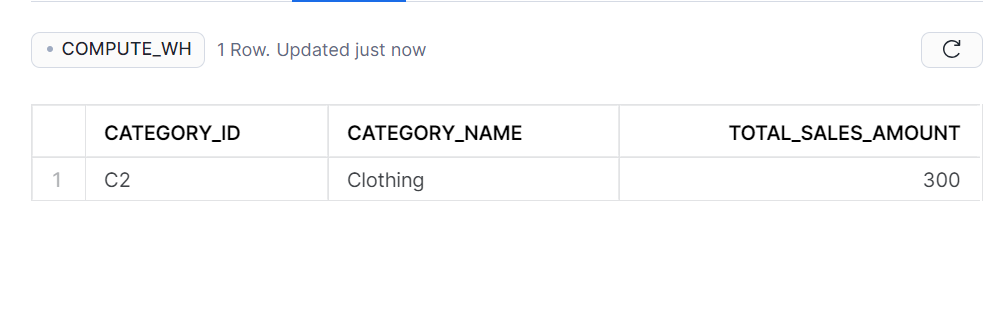
GROUP BY p.category\_id,

         c.category\_name

ORDER BY total\_sales\_amount DESC

LIMIT 1

**Code output:**

****