Q1. Income slab

Problem Statement:

Given a dataframe containing employee income information, calculate and return the percentage of employees in medium income bracket.

Input Format:

A dataframe

Output Format:

A floating point number

Sample Input:

	name	gender	income
0	Bruno	M	low
1	Ariana	F	high
2	Harry	M	medium
3	Selena	F	medium
4	Weeknd	M	medium

Sample Output:

60.0

Use the code below in order to create the given dataframe:

import pandas as pd

data = {

'name': ['Bruno', 'Ariana', 'Harry', 'Selena', 'Weeknd'],

'gender': ['M', 'F', 'M', 'F', 'M'],

'income': ['low', 'high', 'medium', 'medium', 'medium']

}

df = pd.DataFrame(data)

```
import pandas as pd

def calculate(df):
    '''
    INPUT: df -> pandas dataframe

    OUTPUT: result -> float
    '''
    ### STEP 1 : filter employees with medium income

    ### STEP 2: calculate count of medium income employees

    ### STEP 3: calculate percentage of medium income employees

    return result
```

Q2. Population greater than 10 mn

Given dataframe consists of the following information about different countries :

- The population,
- The year it was measured, and
- The continent it belongs to.

Complete the function **population_df()** to return a dataframe consisting of records where the population is **greater than 10 million.**

The dataframe should be **sorted in ascending order**, **first by year and then by population column**.

Input Format:

The data of dataframe is given in a 2d list where the elements of each record in the same order of columns as shown in figure.
This data is used for making the dataframe and passed as an argument.

Output Format:

A dataframe

Sample Input:

	Country	Year	Population	Continent
0	Afghanistan	1952	8425333.0	Asia
1	Australia	1957	9712569.0	Oceania
2	Brazil	1962	76039390.0	Americas
3	China	1957	637408000.0	Asia
4	France	1957	44310863.0	Europe
5	India	1952	372000000.0	Asia
6	United States	1957	171984000.0	Americas

Sample Output:

	Country	Year	Population	Continent
0	India	1952	372000000.0	Asia
1	France	1957	44310863.0	Europe
2	United States	1957	171984000.0	Americas
3	China	1957	637408000.0	Asia
4	Brazil	1962	76039390.0	Americas

Sample Explanation:

From the list of countries, the ones with a population greater than 10 million, sorted according to the year and population are printed, as seen in the sample output.

Use the code below in order to create the given dataframe:

import pandas as pd

data = {

'Country': ['Afghanistan', 'Australia', 'Brazil', 'China', 'France', 'India', 'United States'],

'Year': [1952, 1957, 1962, 1957, 1957, 1952, 1957],

'Population': [8425333.0, 9712569.0, 76039390.0, 637408000.0, 44310863.0, 372000000.0, 171984000.0],

'Continent': ['Asia', 'Oceania', 'Americas', 'Asia', 'Europe', 'Asia', 'Americas']

}

df = pd.DataFrame(data)

```
import pandas as pd
def population_df(df):
    '''df is a dataframe with columns ['Country', 'Year', 'Population', 'Continent']
       Output -> A dataframe is expected to be returned.'''

    # Filter the dataframe for records greater than 10 million


    # Sort the filtered dataframe first by year, then by population


    # Don't modify this line of code
    return df_sorted.reset_index(drop = True)
```

Q3. Concatenation-average

Tests were taken in two sets 1 and 2. Names of the students who participated in the respective tests are given. Marks are recorded according to the subjects, names, and tests.

df1 and **df2** contain **test1** and **test2** respectively.

You are required to return the following:

1. The **average Chemistry score** of students from both the dataframes.
2. The **average score across all the subjects** for students who participated in both sets of exams (here all subjects of both the exams have to be considered together).

Input Format:

Number of testcases, for each testcase there will be six lines of input.

In each new line, scores for each subject are taken for the students participating in set1 then set2.

First three lines are for exams in set1 (i.e. chemistry, physics and hindi).
Last three lines are for exams in set2 (i.e. chemistry, maths and english).

Output Format:

A 2d list with one int element and another element as list i.e. [int, list]

Sample Input:

For dataframes you can consider the following code:

import pandas as pd

df1 = pd.DataFrame({"name": ['tobey', 'peter', 'chris', 'pratt'],'chem_score':[10, 9, 8, 7], 'phy_score':[7, 8, 9, 10], 'Hindi_score':[9, 9, 9, 9]})

df2 = pd.DataFrame({"name": ['chris', 'pratt', 'andrew', 'tom'],'chem_score':[10, 10, 10, 9], 'maths_score':[6, 6, 7, 9], 'eng_score':[9, 10, 10, 9]})

df1

	name	chem_score	phy_score	Hindi_score
0	tobey	10	7	9
1	peter	9	8	9
2	chris	8	9	9
3	pratt	7	10	9

df2

	name	chem_score	maths_score	eng_score
0	chris	10	6	9
1	pratt	10	6	10
2	andrew	10	7	10
3	tom	9	9	9

Sample Output:

[9.125, [8.5, 8.666666666666666]]

Sample's Explanation:

A. For the first requirement, we have to return the average Chemistry score.
   - Here the answer is (10+9+8+7+10+10+10+9)/8 = 9.125

B. For the second requirement, we have to return the average score of both Chris and Pratt in this order.
   - For Chris: (8+9+9+10+6+9)/6 = 8.5
   - Similarly, the average marks for Pratt can be calculated across all the subjects.

Note:

- You are expected to return a 2d list with the rows representing the two requirements.

```
import pandas as pd

def solve(df1,df2):
    # just return the 2d list explained in the description
    # Example of an output [9.125, [8.5, 8.666666666666666]]
    # YOUR CODE GOES HERE

    return ans
```

Q4. House of common

Problem Statement:

Given two dataframes **df1** (containing details of customer) and **df2** (having details of orders),

Merge the dataframes such that resultant dataframe contains records for customer ids which are common in both dataframes.

Input Format:

Line separated dataframes (as dict)

Output Format:

A dataframe

Sample Input:

df1 :

	cust_id	name
0	101	rick
1	102	morty
2	103	pickle
3	104	jerry

df2 :

	order_id	cust_id	amount
0	OR1	102	1200
1	OR3	105	650
2	OR23	101	120
3	OR42	102	989

Sample Output:

	cust_id	name	order_id	amount
0	101	rick	OR23	120
1	102	morty	OR1	1200
2	102	morty	OR42	989

Use the code given below in order to create the dataframes, **df1** and **df2** :

```
import pandas as pd

df1 = pd.DataFrame({
   'cust_id': [101, 102, 103, 104],
   'name': ['rick', 'morty', 'pickle', 'jerry']})


df2 = pd.DataFrame({
   'order_id': ['OR1', 'OR3', 'OR23', 'OR42'],
   'cust_id': [102, 105, 101, 102],
   'amount': [1200, 650, 120, 989]})
```

```
import pandas as pd

def merge(df1, df2):
    '''
    INPUT: df1, df2 -> dataframe

    OUTPUT: result -> dataframe

    '''

    return result
```

Q5. Pandas merge()

The below-given code snippet would print a dataframe.

```
df1 = pd.DataFrame({'name': ['Jack','Ryan','Chris','Sam'],
'rank': [1, 2, 3, 4]})
df2 = pd.DataFrame({'name': ['Ryan', 'Sam', 'Chris', 'Jack'],
'rank': [3, 1, 4, 2]})

print(pd.merge(df1, df2, on="name"))
```

Mark the option that **correctly represents the columns** of the printed dataframe.

A. ['name', 'rank', 'rank']
B. ['name', 'rank_x', 'rank_y']
C. ['name', 'rank']
D. ['name', 'rank_L', 'rank_R']

Q6. Population density

Problem Statement:

Given a dataframe containing **area** and **population** data, calculate the **Population Density** for each state.

The function should return a Series of population densities sorted in **ascending order**.

Note: Population Density is defined as **population per unit area**.

Input Format: A DataFrame

Output Format: A Series

Sample Input:

	area	population
Alaska	1723337	700000
Texas	695662	26448193
California	423967	38332521
New York	783000	19651127

Sample Output:

```
Alaska          0.406189
New York       25.097225
Texas          38.018740
California     90.413926
dtype: float64
```

Use the code below in order to create the given dataframe:

import pandas as pd

data = {

'city': ['Alaska', 'Texas', 'California', 'New York'],

'area': [1723337, 695662, 423967, 783000],

'population': [700000, 26448193, 38332521, 19651127]

}

df = pd.DataFrame(data)

```python
import pandas as pd
def calc_density(df):
    '''
    INPUT: df -> dataframe

    OUTPUT: result -> A series
    '''
    ## STEP 1: calculate population density

    ## STEP 2: sort the values


    return result
```

Q7. Student Filtering

Problem Description:

Given a dataframe containing records of some college students, with columns **"Name", "Age", "Percentage"** and **"Stream"**.

Return the **name** of the students belonging to **either Commerce or Arts Stream** and with a **Percentage greater than or equal to 75.**

Input Format:

First line of the input contains the dataframe, in the form of a dictionary.

Output Format:

The selected student names in a Series.

Sample Input:

	Name	Age	Percentage	Stream
0	Himanshu	15	80	Commerce
1	Robert	14	77	Science
2	Karie	15	83	Arts
3	Rohan	16	45	Commerce
4	John	17	70	Science

Sample Output:

```
0    Himanshu
2       Karie
Name: Name, dtype: object
```

Sample Explanation:

Only Himanshu and Karie have scored more than 75% in either Commerce or Arts Stream.

Use the code below in order to create the given dataframe:

```
import pandas as pd

data = {

   'Name': ['Himanshu', 'Robert', 'Karie', 'Rohan', 'John'],

   'Age': [15, 14, 15, 16, 17],

   'Percentage': [80, 77, 83, 45, 70],

   'Stream': ['Commerce', 'Science', 'Arts', 'Commerce', 'Science']

}

df = pd.DataFrame(data)
```

```python
import pandas as pd
def student_filter(df):
    '''
    input:
    df -> the dataframe provided to the function

    output:
    df_new -> the selected students of the dataframe
    '''
    # Your code starts here

    # Your code ends here
    return df_new
```

Q8. Average column

Given a dataframe consisting of payments data of candidates,

Complete the function new_avg() that

1. Calculates the **mean** payment rounded off to 2 decimals for each candidate and stores it in a column **avg_payment** and

2. Returns the data frame after deleting all the columns other than '**name**', '**mar_payment',** and '**avg_payment**'

Input Format:

A dataframe of payments data

Output Format:

A data frame

Sample Input:

	name	jan_payment	feb_payment	mar_payment
0	a	10	11	100
1	b	12	11	120
2	c	13	11	130
3	d	14	11	130
4	e	15	11	140

Sample Output:

	name	mar_payment	avg_payment
0	a	100	40.33
1	b	120	47.67
2	c	130	51.33
3	d	130	51.67
4	e	140	55.33

Sample Explanation:

- A new column avg_payment is added to the data frame consisting of the average payment of each of the names.

- For example, The avg_payment of 'a' is (10+11+100)/3 = 40.33 and the avg_payment of 'b' is (12+11+120)/3 = 47.67.
- In the end, a dataframe with the columns, the new column (avg_payment*),* and mar_payment is returned because all the other columns are deleted.

Use the code below in order to create the given dataframe:

import pandas as pd

data = {

'name': ['a', 'b', 'c', 'd', 'e'],

'jan_payment': [10, 12, 13, 14, 15],

'feb_payment': [11, 11, 11, 11, 11],

'mar_payment': [100, 120, 130, 130, 140]

}

df = pd.DataFrame(data)

```
import pandas as pd
def new_avg(df):
    '''df is a dataframe with columns ['name', 'jan_payment', 'feb_payment', 'mar_payment']
        A dataframe is expected to be returned'''

    # Create the new column named "avg_payment"

    # drop the columns 'jan_payment' and 'feb_payment'

    # Don't modify this line of code
    return df.reset_index(drop = True)
```

Q9. Concat and drop

df1

	A	B	C
first	1	11	111
second	2	22	222
third	3	33	333
fourth	4	44	444

df2

	D	E	F
fourth	a	aa	aaa
fifth	b	bb	bbb
sixth	c	cc	ccc
seventh	d	dd	ddd

Given the above two data frames **df1** and **df2**, the following operations are performed:

df3=pd.concat([df1,df2])

#Block a
df4=df3.drop(["fourth"])

#Block b
df4 = df3.reset_index().drop_duplicates(subset='index', keep='last')
df4 = df4.set_index('index').sort_index()

#Block c
df3.loc['fourth','A']*2

For the above dataframe **df3**, which option is **True** about the mentioned statements with respect to the code in blocks **a**, **b**, and **c**?

The statements are:

1. Block **a**, drops one of the rows with the label fourth, and the output of block **c** is Nan.

2. Block **a**, drops all the rows with the label fourth, and the output of block **c** is a pandas series.

3. Block **b**, drops all the rows with occurrences of duplicates in the index except the last occurrence of each one.

4. Block **b**, drops all the rows with occurrences of duplicates in the index except the first occurrence of each one.

Note: drop_duplicates() is used to drop duplicate values in columns and the blocks a, b, and c are not run in any combination they all are executed **individually** for the statements.

Options:

A. 1 and 4.
B. 2 and 3.
C. 1 and 3.
D. 2 and 4.

Q10. Merge pandas

Given two dataframes, **df1** and **df2** :

DataFrame: **df1**

	Names	Profession
0	Jack	Accounting
1	Roman	Engineering
2	Steph	Engineering

DataFrame: **df2**

	Profession	skills_required
0	Accounting	math
1	Accounting	spreadsheets
2	Engineering	coding
3	Engineering	linux
4	HR	spreadsheets
5	HR	organization

Which of the following is the **correct** output for the **merge** operation pd.merge(df1, df2)?

a.

```
    Names   Profession skills_required
0    Jack   Accounting            math
1  Roman   Accounting          coding
2  Steph  Engineering           linux
```

b.

```
    Names   Profession skills_required
0    Jack   Accounting            math
1    Jack   Accounting    spreadsheets
2  Roman  Engineering          coding
3  Roman  Engineering           linux
4  Steph  Engineering          coding
5  Steph  Engineering           linux
```

c.

```
    Names   Profession skills_required
0    Jack   Accounting            math
1  Roman   Accounting    spreadsheets
2  Steph  Engineering          coding
3    Jack  Engineering           linux
4  Roman           HR    spreadsheets
```

d.

```
    Names   Profession skills_required
0    Jack   Accounting            math
1  Roman   Accounting    spreadsheets
2  Steph  Engineering          coding
3    Jack  Engineering           linux
4  Roman           HR    spreadsheets
5  Steph           HR    organization
```

Note: When we don't specify the "**on**" parameter while merging, this defaults to merging based on the **intersection** of the columns in both DataFrames.

Q11. Count the amount

Problem Statement:

Given two dataframes customer (containing details of customer) and orders (having details of orders).

Also, name is given which is a customer's name.

Perform the following operation:

1. Merge the dataframe such that the resultant dataframe should contain records of all customer ids present in customer dataframe.
2. Calculate the total order amount for given customer name

Return the merged dataframe and the sum amount.

Input Format:

Line separated dataframes and customer name

Output Format:

A Tuple of merged dataframe and sum amount

Sample Input:

customer:

	cust_id	name
0	101	rick
1	102	morty
2	103	pickle
3	104	jerry

orders:

	order_id	cust_id	amount
0	OR1	102	1200
1	OR3	105	650
2	OR23	101	120
3	OR42	102	989

name: 'morty'

Sample Output

	cust_id	name	order_id	amount
0	101	rick	OR23	120.0
1	102	morty	OR1	1200.0
2	102	morty	OR42	989.0
3	103	pickle	NaN	NaN
4	104	jerry	NaN	NaN

2189.0

Use the code below in order to create the given dataframes:

customer

```
import pandas as pd
data = {
   'cust_id': [101, 102, 103, 104],
   'name': ['rick', 'morty', 'pickle', 'jerry']
}
customer = pd.DataFrame(data)
```

orders

```
import pandas as pd
data = {
   'order_id': ['OR1', 'OR3', 'OR23', 'OR42'],
   'cust_id': [102, 105, 101, 102],
   'amount': [1200, 650, 120, 989]
}
orders = pd.DataFrame(data)
```

```
import pandas as pd
def get_amount(customer, orders, name):
    '''
    INPUT:
        customer, orders -> dataframe
        name -> string

    OUTPUT:
        (merged_df, sum_amount) -> tuple of (dataframe, float)

    '''
    ## STEP 1: merge customer and orders dataframe

    ## STEP 2: calculate sum of amount for given name

    return merged_df, sum_amount
```

Q12. StockCode in UK

Find out the number of unique **StockCode** in the United Kingdom according to the given Sales data?

Dataset: link

A. 7050
B. 900023
C. 1502
D. 5303