Question:

Substring with maximum uppercase characters
Description
Given a string, write a Python program to find the largest substring of uppercase characters and print the length of that substring. Check the sample inputs and outputs for a better understanding.

Input - String
Output - String

Sample Input - I lovE PROgrAMMING
Sample Output - 6

Explanation - AMMING is the largest substring with all characters in uppercase continuously

----
Sample Input - MuMbaI is in MAHArashTRA
Sample Output - 4

Explanation - MAHA is the largest substring with all characters in uppercase continuously.

Sample Input - India WOn the WOrLD CUP
Sample Output - 3

Explanation - CUP is the largest substring with all characters in uppercase continuously.
Execution Time Limit
10 seconds

Submit

Solution:

Way 1:->

```
#read the string
test_str = input()

# when a character in the string is a uppercase, cnt will count the number
of #continuous uppercase characters from that index
cnt=0
# max_run stores the final maximum count which is to be displayed as
answer
max_run=0

for i in range(0,len(test_str)):
  # updating run count on uppercase
  if test_str[i].isupper():
    cnt=cnt+1
  # on encountering lowercase, update the cnt with 0 and start counting
the continuous uppercase run again. max_run  already has the count of
this run stored.
  #updating the value of max_run with cnt, if the current ‘cnt’ is greater
than the previous ‘max_run’
  else:
    if cnt>=max_run:
      max_run=cnt
    cnt=0
#for boundary cases - when there is a potential longest substring at the
end of the string
if cnt>=max_run:
  max_run=cnt

# printing result
print(max_run)
```

Way 2:->

```
#Take input here
test_str = input()

#Write the code here
allCaps = []
n1 = ''
f = 0
for ch in test_str:
  if ch.isupper() :
    f = 1
    n1 += ch
  else:
    if n1 != '':
          f = 0
          allCaps.append(n1)
          n1 = ''
if f == 1:
   f = 0
   allCaps.append(n1)
   n1 = ''
length_lst = [len(ch) for ch in allCaps]
print(max(length_lst))
```

Q .)

Extracting Elements from Array
Description
From a given array, extract all the elements which are greater than 'm' and less than 'n'. Note: 'm' and 'n' are integer values provided as input.

Input format:
A list of integers on line one
Integer 'm' on line two
Integer 'n' on line three

Output format:
1-D array containing integers greater than 'm' and smaller than 'n'.

Sample input:
[1, 5, 9, 12, 15, 7, 12, 9] *(array)*
6 *(m)*
12 *(n)*

Sample output:
[9 7 9]

Sample input:
[1, 5, 9, 12, 15, 7, 12, 9]
12
6

Sol.)

```
#take input here
import ast
input_list=ast.literal_eval(input())
m=int(input())
n=int(input())
```

```
import numpy as np
array_1 = np.array(input_list)#start writing your code from here
final_array = array_1[np.logical_and(array_1 > m , array_1 < n)]#start
writing your code from here

print(final_array)
```

Q.)

Border Rows and Columns

Description
Extract all the border rows and columns from a 2-D array.

Format:
Input: A 2-D Python list
Output: Four NumPy arrays - First column of the input array, first row of the input array, last column of the input array, last row of the input array respectively.

Example:
Input 1:
```
[[11 12 13 14]
 [21 22 23 24]
 [31 32 33 34]]
```
Output 1:
```
[11 21 31]
[11 12 13 14]
[14 24 34]
[31 32 33 34]
```
Execution Time Limit
15 seconds

Sol.)

```
# Read the input list
import ast,sys
input_str = sys.stdin.read()
input_list = ast.literal_eval(input_str)

import numpy as np

# Convert the input list to a NumPy array
array_2d =np.array(input_list)

row,col = array_2d.shape
# Extract the first column, first row, last column and last row respectively using
# appropriate indexing
col_first = array_2d[:,0]
row_first = array_2d[0,:]
col_last =  array_2d[:,col-1]
row_last = array_2d[row-1,:]

print(col_first)
print(row_first)
print(col_last)
print(row_last)
```

Q.)

Create border array

Description

Given a single integer n, create an (n x n) 2D array with 1 on the border and 0 on the inside.

Note: Make sure the array is of type **int.**

Example:

Input 1:

```
4
```

Output 1:

```
[[1 1 1 1]
[1 0 0 1]
[1 0 0 1]
[1 1 1 1]]
```

Input 2:

```
2
```

Output 2:

```
[[1 1]
 [1 1]]
```

Execution Time Limit

15 seconds

Sol.)

```
# Read the variable from STDIN
n = int(input())

import numpy as np

# Create an 'n*n' array of all ones
np_ones = np.ones((n,n),dtype='int8')
```

```
np_ones[1:-1,1:-1]=0

# Fill the array with zeroes from second index (i.e. index 1) to second last index.
# Do this for both row indices and column indices

# Print the array created
print(np_ones)
```

Q.)

Stacking arrays
Description
Merge the three arrays provided to you to form a one 4x4 array.
[Hint: Check the function *np.transpose() in the 'Manipulating Arrays' notebook provided.*]

Input:
Array 1: 3*3
[[7, 13, 14]
[18, 10, 17]
[11, 12, 19]]

Array 2: 1-D array
[16, 6, 1]

Array 3: 1*4 array
[[5, 8, 4, 3]]

Output:
```
[[7 13 14 5]
[18 10 17 8]
[11 12 19 4]
[16 6 1 3]]
```
Execution Time Limit
15 seconds

Sol.)

```
# Read the input
import ast,sys
input_str = sys.stdin.read()
input_list = ast.literal_eval(input_str)
list_1 = input_list[0]
list_2 = input_list[1]
list_3 = input_list[2]
```

```
# Import NumPy
import numpy as np

list_1np = np.array(list_1)
list_2np = np.array(list_2)
list_3np = np.array(list_3)

list_3t = np.transpose(list_3np)

stackF = np.vstack((list_1np,list_2np))

# Write your code here
final_array = np.hstack((stackF,list_3t))
print(final_array)
```

Correct! The np.percentile function helps you calculate the percentile in any array. Here you're calculating the 50th percentile, i.e. the median. Now, since there are 6 elements in the array, there are 2 middle elements, which are 7 and 10. Hence, the median would be 7+10/2 = 8.5.

Q.)

Print Z

Description

Given a single positive integer 'n' greater than 2, create a NumPy array of size (n x n) will all zeros and ones such that the ones make a shape like 'Z'.

Examples:

Input 1:

3

Output 1:

```
[[1 1 1]
 [0 1 0]
 [1 1 1]]
```

Input 2:

5

Output 1:

```
[[1 1 1 1 1]
 [0 0 0 1 0]
 [0 0 1 0 0]
 [0 1 0 0 0]
 [1 1 1 1 1]]
```

Explanation: Notice that the 1s in the array make a shape like 'Z'.

Execution Time Limit
15 seconds

Sol1.) **MY SOLUTION**

```
# Read the input
n = int(input())

# Import the NumPy package
import numpy as np

z = np.zeros((n,n),dtype='int')
row,col = z.shape

for i in range(row):
    for j in range(col):
        if i==0 or i==n-1 or (i+j==n-1):
            #print('{0}th row {1}th col'.format(i,j))
            #print('con1',i==0)
            #print('con2',j==n-1)
            #print('con3',(i+j==n-1))
            z[i][j]=1

print(z)
# Write your code here
```

Sol2.) **PROVIDED SOLUTION**

```
# Read the input
n = int(input())

# Import the NumPy package
import numpy as np

# Create an (n x n) array with all zeros
z = np.zeros((n, n), dtype = int)

# Make the first and last rows all 1s
z[0] = np.ones(1, dtype = int)
z[n-1] = np.ones(1, dtype = int)

# Run a loop from the second row (index 1) till the second last row.
for i in range(1, n):
# Fill the 1s in appropriate indices. Notice that for every row index i, the 1
# will be in the (n-i-1)th row
    z[i][n-i-1] = 1
```

# Print the final value of z
print(z)

Q.)

Sort by Column

Description
Given a 2D NumPy array, sort it by the 1st column. Print the final sorted array as a NumPy array only.

Note: If two values in the 1st column are equal then the column in which the 2nd column value is lesser should come first. If the value in the second column is also the same then go to the third value and so on.

Example:
Input 1:
```
[[9 3 2]
 [4 0 1]
 [5 8 6]]
```
Output 2:
```
[[4 0 1]
 [5 8 6]
 [9 3 2]]
```
Input 2:
```
[[9 3 2]
 [4 0 1]
 [9 8 6]]
```
Output 2:
```
[[4 0 1]
 [9 3 2]
 [9 8 6]]
```

Explanation:
Example 1: Notice that the values in the first column are sorted. Also notice that the whole row should be moved and not just the individual values in the first column. For example, the row with the smallest value, i.e. 4 was moved as a whole.
Example 2: Since this time the first row contains two 9s, the row in which the value in the second column is lesser came first.
Execution Time Limit
15 sec

Sol.)

# Reading the input list

```
import ast,sys
input_str = sys.stdin.read()
input_list = ast.literal_eval(input_str)

# Import the NumPy package
import numpy as np

# Converting the list to a NumPy array
n_array = np.array(input_list)
for i in range(len(n_array[0, :])):
   n_array = n_array[n_array[:,-1-i].argsort()]

print(n_array)
# Write your code here
```

Q.)

Dataframe Pivot Table

Description

Group the data 'df' by 'month' and 'day' and find the mean value for column 'rain' and 'wind' using the pivot table command.

Execution Time Limit

15 seconds

Sol.)

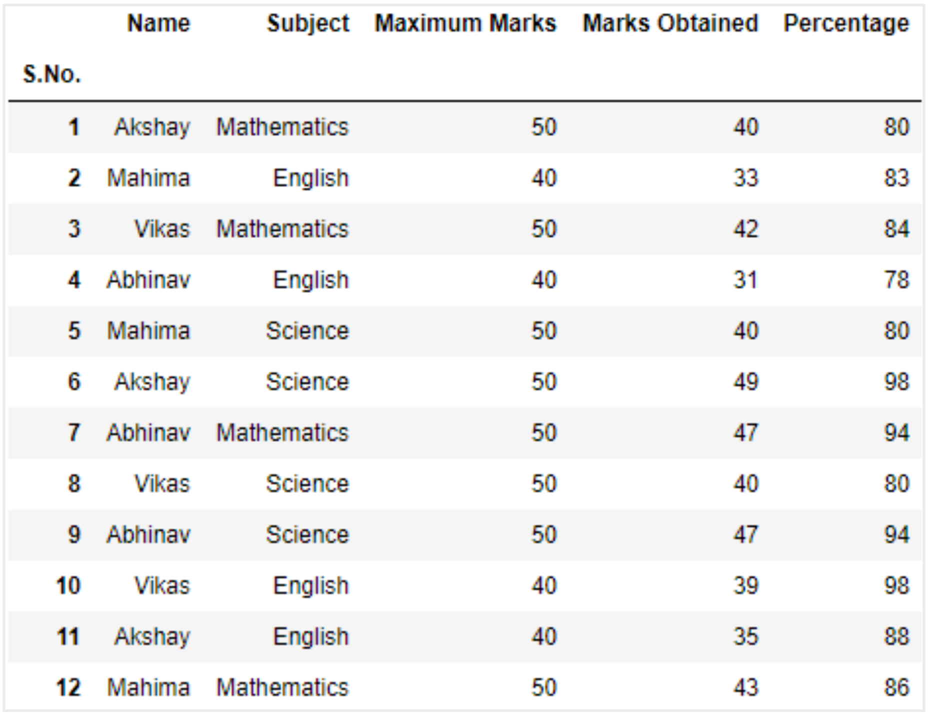
```
import numpy as np
import pandas as pd , ssl
ssl._create_default_https_context = ssl._create_unverified_context
df = pd.read_csv('https://query.data.world/s/
vBDCsoHCytUSLKkLvq851k2b8JOCkF')
df_1
=df.pivot_table(values=["rain","wind"],index=["month","day"],aggfunc={
"rain":"mean","wind":"mean"}) #Type your code here.
print(df_1.head(20))
```

Q.)

Loading a csv with index

Description

Using the file ‘marks.csv’, create a dataframe as shown below.



S.No.	Name	Subject	Maximum Marks	Marks Obtained	Percentage
1	Akshay	Mathematics	50	40	80
2	Mahima	English	40	33	83
3	Vikas	Mathematics	50	42	84
4	Abhinav	English	40	31	78
5	Mahima	Science	50	40	80
6	Akshay	Science	50	49	98
7	Abhinav	Mathematics	50	47	94
8	Vikas	Science	50	40	80
9	Abhinav	Science	50	47	94
10	Vikas	English	40	39	98
11	Akshay	English	40	35	88
12	Mahima	Mathematics	50	43	86

You must be able make the first column of the file as the index and name it 'S.No.'. Also, the columns must be renamed as shown in the image.
Execution Time Limit

Sol.)

```
import numpy as np
import pandas as pd , ssl
ssl._create_default_https_context = ssl._create_unverified_context
# The file is stored at the following path:
# 'https://media-doselect.s3.amazonaws.com/generic/
A08MajL8qN4rq72EpVJbAP1Rw/marks_1.csv'
# Provide your answer below
df = df_q1 = df = pd.read_csv('https://media-
doselect.s3.amazonaws.com/generic/A08MajL8qN4rq72EpVJbAP1Rw/
marks_1.csv',sep='|',header=None,index_col="S.No.",names=["S.No.","Na
me","Subject","Maximum Marks","Marks Obtained","Percentage"]) #
Write your answer here
# Write your answer here

print(df)
```

Q.)

Operations on multiple dataframes
Description
Given three data frames containing the number of gold, silver, and bronze Olympic medals won by some countries, determine the total number of medals

won by each country.
Note: All three data frames don’t have all the same countries. So, ensure you use the ‘fill_value’ argument (set it to zero), to avoid getting NaN values. Also, ensure you sort the final data frame, according to the total medal count in descending order. Make sure that the results are in integers.

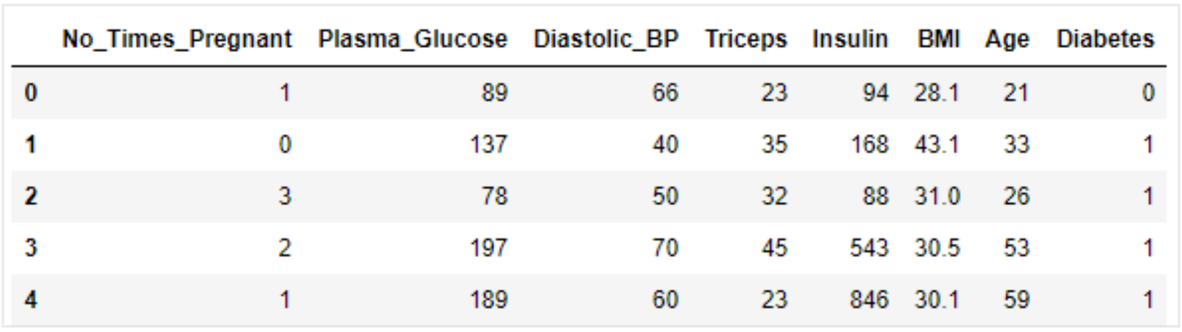
Sol.)

```
import numpy as np
import pandas as pd

# Defining the three dataframes indicating the gold, silver, and bronze
medal counts
# of different countries
gold = pd.DataFrame({'Country': ['USA', 'France', 'Russia'],
                     'Medals': [15, 13, 9]}
                 )
silver = pd.DataFrame({'Country': ['USA', 'Germany', 'Russia'],
                    'Medals': [29, 20, 16]}
                 )
bronze = pd.DataFrame({'Country': ['France', 'USA', 'UK'],
                    'Medals': [40, 28, 27]}
                 )

gold.set_index("Country",inplace=True)
silver.set_index("Country",inplace=True)
bronze.set_index("Country",inplace=True)
tally_1 = gold.add(silver,level="Country",fill_value=0)
final_tally = tally_1.add(bronze,level="Country",fill_value=0)
final_tally=final_tally.astype({"Medals": int})
print(final_tally.sort_values(by="Medals",ascending=False))
```

Q.)

Risk of Diabetes Based on BMI

Description
You've been given the pima_indian_diabetes dataset. Here are its first few rows:



	No_Times_Pregnant	Plasma_Glucose	Diastolic_BP	Triceps	Insulin	BMI	Age	Diabetes
0	1	89	66	23	94	28.1	21	0
1	0	137	40	35	168	43.1	33	1
2	3	78	50	32	88	31.0	26	1
3	2	197	70	45	543	30.5	53	1
4	1	189	60	23	846	30.1	59	1

Notice the BMI and Diabetes column. You need to find the BMI which is most likely to cause Diabetes. First round the BMI to an integer value and return which BMI has the most risk of diabetes based on the 0, 1 diabetes values provided in the dataframe.

Expected Output: Just print a single integer denoting the value of the required BMI. (Please only output an integer value. For example, if the output is 30.0 please convert it to int and output 30.).

SOl. 1)

```
# Import the Pandas package
import pandas as pd

# Reading the input dataframe
pima = pd.read_csv('https://media-doselect.s3.amazonaws.com/generic/
pLZK3n22ezVwAG2XOYW5qEx7V/pima_indian_diabetes.csv')
pima = pima.round({"BMI":0})
pima = pima.astype({"BMI":int})
dia = pima[pima["Diabetes"]==1]
# Write your code here
mostLike = int(dia["BMI"].mode())
print(mostLike)
```

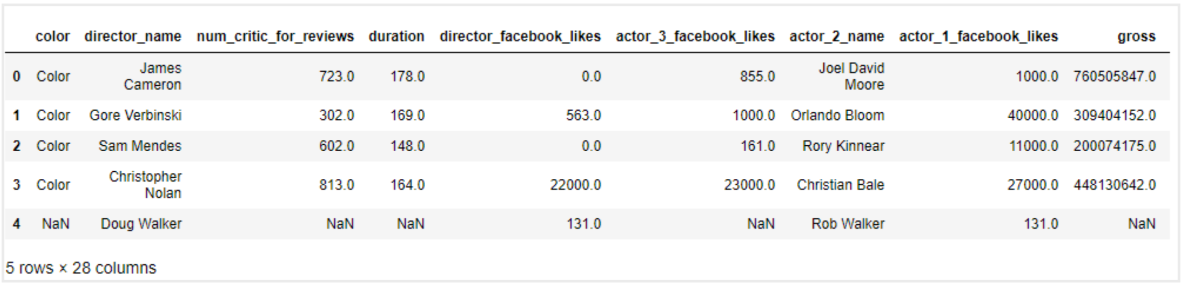
Sol. 2) ALITER FOR ABOVE SOLUTION.

```
# Import the Pandas package
import pandas as pd

# Reading the input dataframe
pima = pd.read_csv('https://media-doselect.s3.amazonaws.com/generic/
pLZK3n22ezVwAG2XOYW5qEx7V/pima_indian_diabetes.csv')
pima = pima.round({"BMI":0})
pima = pima.astype({"BMI":int})
pima_piv = pima.pivot_table(index = 'BMI',values = ['Diabetes'], aggfunc =
'sum')
pima_piv.sort_values(by = 'Diabetes', inplace = True, ascending = False)
# Write your code here
```

mostLike = pima_piv.index[0]
print(mostLike)

Q.)

Impute Missing Values
Description
You're given a movies dataframe which contains quite a few aspects of some movies from 1916-2016. Here are the first few rows of the dataframe.



	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN

5 rows × 28 columns

image.png 26.77 KB

There are a lot of columns that aren't visible. But you might have noticed straight away that there are quite a few missing values in the data frame. Two columns for instance, 'aspect_ratio' and 'facenumber_in_poster' also have a few missing values(NaN). Now, replace the missing values with the 'median' value of the respective columns and print the null value count for both.

Expected Output: First print the number of missing values in both of these columns, then output the median in both the columns and then impute the missing values with the respective medians and print the count of missing values again. Store all of these in a dictionary format like the following:

{'aspect_ratio_mv': 431, 'facenumber_in_poster_mv': 97}
{'aspect_ratio_median: 1.44, 'facenumber_in_poster': 2.0}
{'aspect_ratio_final': 0, 'facenumber_in_poster_final': 0}

The code for the same has been provided in the stub; you just need to complete these dictionaries.

Note: You don't need to use any print statement. The print statements have already been written; you just need to complete the dictionaries provided in the stub.
Execution Time Limit
15 seconds

Sol.)

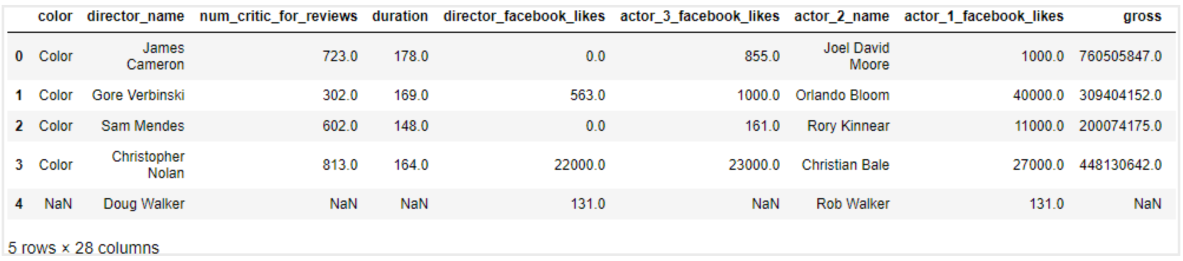
```
# Importing the pandas package
import pandas as pd

# Reading the movies dataframe
movies = pd.read_csv('https://media-doselect.s3.amazonaws.com/
generic/1M2ZzY2M9PEBPJovgoaBgZdbM/movie_data.csv')

# Your aim is to complete the following three print statements after all the
colons
na_c_ar = pd.isna(movies["aspect_ratio"]).sum()
na_c_fp = pd.isna(movies["facenumber_in_poster"]).sum()
subAR = movies[~pd.isna(movies["aspect_ratio"])]
subFP = movies[~pd.isna(movies["facenumber_in_poster"])]

movies["aspect_ratio"].fillna(subAR["aspect_ratio"].median(),inplace=Tr
ue)
movies["facenumber_in_poster"].fillna(subAR["facenumber_in_poster"].
median(),inplace=True)

na_c_ar1 = pd.isna(movies["aspect_ratio"]).sum()
na_c_fp1 = pd.isna(movies["facenumber_in_poster"]).sum()
mv = {'aspect_ratio_mv': na_c_ar, 'facenumber_in_poster_mv': na_c_fp}
median = {'aspect_ratio_median': subAR["aspect_ratio"].median(),
'facenumber_in_poster_median':
subFP["facenumber_in_poster"].median()}
final = {'aspect_ratio_final': na_c_ar1, 'facenumber_in_poster_final':
na_c_fp1}

# Printing the values in the three dictionaries. Please do not edit this part
print(sorted(mv.values()))
print(sorted(median.values()))
print(sorted(final.values()))
```

Q.)

Removing Missing Values

Description

Consider the movie dataset again. Here are the first few rows of the same:



	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN

5 rows × 28 columns

Now, while you imputed missing values in the last question, there are a few columns for which imputing the missing values won't be wise and it's better just to drop them.

There are two columns 'actor_1_facebook_likes' and 'actor_2_facebook_likes' which have quite a few missing values but since you have less data points, you don't want to drop many of them. For a hypothetical analysis that you're conducting, it will be okay if one of them has a missing value but you can't afford to have both missing values. So your aim here is to find the indices of the rows in which both of these columns have missing values simultaneously.

Expected Output:
- First print the indices of the rows where both these columns have missing values. The print statement has been provided in the stub. You just need to fill it.
- After you have printed the above indices, drop these particular rows and print the number of retained rows in the dataframe.

A sample output would look like the following:
[389, 1019, 1178, 3400, 4012]
4847

Here, the list in the first line indicates a sample list which indicates the indices of the rows where both of the columns have missing values. And the second line represents the number of rows remaining in the dataframe after you have dropped the above rows.

Execution Time Limit
15 seconds

Sol.)

# Importing the pandas package
import pandas as pd

# Reading the dataframe
movies = pd.read_csv('https://media-doselect.s3.amazonaws.com/generic/ZY9xWvEzMB7NEoW08r52L8j2O/movie_data%20(1).csv')

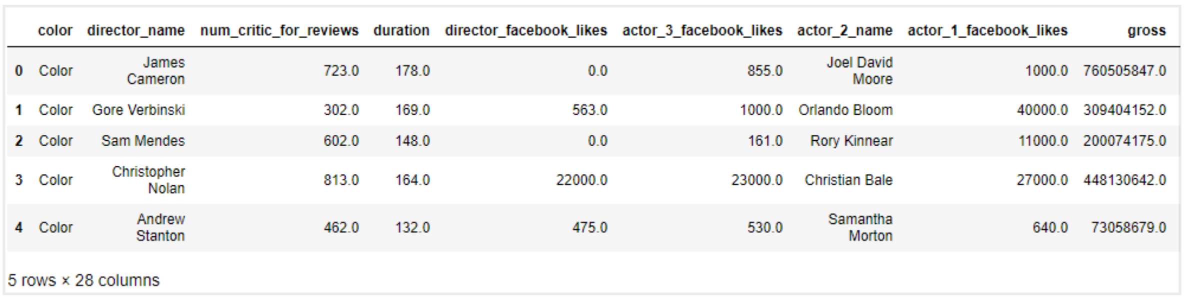
```
bothNa = movies[(pd.isna(movies["actor_1_facebook_likes"])) &
(pd.isna(movies["actor_2_facebook_likes"]))]
# Print out the indices of the rows in which both these columns have
missing values
# as a list
print(list(bothNa.index))
l_dr = list(bothNa.index)
# Write your code for dropping these particular rows here


# Print the number of remaining rows
print(movies.drop(index=l_dr).shape[0])
```

Q.)

Data Analysis

Description

This time you've a fully cleaned movie dataset. The first few rows of the dataset are shown below:



	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0
4	Color	Andrew Stanton	462.0	132.0	475.0	530.0	Samantha Morton	640.0	73058679.0

5 rows × 28 columns

You're a producer looking to make a blockbuster movie. There will primarily be three lead roles in your movie and you wish to cast the most popular actors for it. Now, since you don't want to take a risk, you will cast a trio which has already acted in together in a movie before. The metric that you've chosen to check the popularity is the Facebook likes of each of these actors.

The dataframe has three columns to help you out for the same, viz. 'actor_1_facebook_likes', 'actor_2_facebook_likes', and 'actor_3_facebook_likes'. Your objective is to find the trio which has the most number of Facebook likes combined. But there is a small condition which is that none of the three actors' Facebook likes should be less than half of the other two. For example, the following is a valid combo:
actor_1_facebook_likes: 70000
actor_2_facebook_likes: 40000
actor_3_facebook_likes: 50000

But the below one is not:
actor_1_facebook_likes: 70000
actor_2_facebook_likes: 40000

actor_3_facebook_likes: 30000

since in this case, actor_3_facebook_likes is 30000, which is less than half of actor_1_facebook_likes.

Expected Output: Find out the most popular trio and print them as a list sorted in alphabetical order. For example:
['Brad Pitt', 'Jake Gyllenhaal' 'Meryl Streep']
Execution Time Limit
15 seconds

Submit

Sol.)

```
# Importing the pandas package
import pandas as pd

# Reading the movies file
movies = pd.read_csv('https://media-doselect.s3.amazonaws.com/
generic/pLLQowB8OYx0oBWdMbY4gp4wb/movies_final (2).csv', sep='\t')

# Group the dataframe using the actor names as indices and facebook
likes as values
group = movies.pivot_table(values = ['actor_1_facebook_likes',
'actor_2_facebook_likes', 'actor_3_facebook_likes'],
                    index = ['actor_1_name', 'actor_2_name', 'actor_3_name'],
aggfunc='sum')

# Create a new column 'Total likes' which will contain the sum of likes of all
three actors
group['Total likes'] = group['actor_1_facebook_likes'] +
group['actor_2_facebook_likes'] + group['actor_3_facebook_likes']

# Sort the dataframe using the 'Total likes' column
group.sort_values(by=['Total likes'], inplace=True, ascending = False)

# Reset the index of the grouped dataframe so you can access the indices
as columns easily
group.reset_index(inplace=True)

# Initialise the value of a variable 'j' to 0. This variable will be used to keep
# a track of the rows during the loop iteration
j = 0

# Run a loop through the length of the column 'Total likes'
for i in group['Total likes']:
# Sort the facebook likes of three actors and store them in a variable
'temp'
```

```
   temp = sorted([group.loc[j,'actor_1_facebook_likes'],
group.loc[j,'actor_2_facebook_likes'],
group.loc[j,'actor_3_facebook_likes']])

# Check if the smallest value in temp is greater than half the value of the
other two
# And also check if the middle value is greater than half the value of the
max value
   if temp[0] >= temp[1]/2 and temp[0] >= temp[2]/2 and temp[1] >=
temp[2]/2:
# If the above condition satisfies, print the correspoding actor names as a
sorted list
# and break the loop
      print(sorted([group.loc[j, 'actor_1_name'], group.loc[j,
'actor_2_name'], group.loc[j, 'actor_3_name']]))
      break
# Keep incrementing the value of j with every loop interation
   j += 1
```

Sol.)

```
# Importing the pandas package
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
# Reading the movies file
movies2 = pd.read_csv('https://media-doselect.s3.amazonaws.com/
generic/pLLQowB8OYx0oBWdMbY4gp4wb/movies_final%20(2).csv',
sep='\t')
movies2["MaxFLikes"] =
movies2["actor_3_facebook_likes"].add(movies2["actor_2_facebook_lik
es"]).add(movies2["actor_1_facebook_likes"])
movies2.sort_values(by="MaxFLikes",ascending=False,inplace=True)
subMov2 =
movies2[["actor_3_name","actor_2_name","actor_1_name","actor_3_face
book_likes","actor_2_facebook_likes","actor_1_facebook_likes","MaxFLik
es"]]
subMov2["actor3H"] = subMov2["actor_3_facebook_likes"]/2
subMov2["actor2H"] = subMov2["actor_2_facebook_likes"]/2
subMov2["actor1H"] = subMov2["actor_1_facebook_likes"]/2
subMov2["validTrio"] =
(((subMov2["actor_3_facebook_likes"]>subMov2["actor2H"])&(subMov2
["actor_3_facebook_likes"]>subMov2["actor1H"]))&

((subMov2["actor_2_facebook_likes"]>subMov2["actor3H"])&(subMov2[
"actor_2_facebook_likes"]>subMov2["actor1H"]))&
((subMov2["actor_1_facebook_likes"]>subMov2["actor3H"])&(subMov2[
"actor_1_facebook_likes"]>subMov2["actor2H"]))
                     )
final_df = subMov2[subMov2["validTrio"]==True]
final_df["trios"]= final_df["actor_3_name"]+final_df["actor_2_name"]
+final_df["actor_1_name"]
popp= final_df.trios.value_counts()
final_df[final_df["trios"]==popp.index[0]]
lst_name = list(final_df[final_df["trios"]==popp.index[0]].iloc[0,[0,1,2]])
lst_name.sort()
print(lst_name)
# Write your code here
```