DATA FRAMES

❖ DataFrame Data Structure

- It is **two dimensional (tabular) heterogeneous data labeled array.**
- It has two indices or two axes : **a row index (axis=0)** and a **column index (axis=1)**
- The **row index is known as index** and **the column index is** called the **column name.**
- The indices can be of any data type.
- It is both **value mutable** and **size mutable.**
- We can perform arithmetic operations on rows and columns.

❖ Creating and Displaying a DataFrame

To create a DataFrame object, we can use the syntax:
<dataframe object> = pandas.DataFrame(<a 2D datastructure> , [columns=<column sequence>] , [index=<index sequence>])
where the 2D data structure passed to it, contains the data values.

- **Empty DataFrame**

```
import pandas as pd
df=pd.DataFrame()
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

- **DataFrame from 2D dictionary**

A 2D dictionary is a dictionary having items as (key : value) where value part is a data structure of any type : a list, a series, a dictionary etc. But the value parts of all the keys should have similar structure and equal lengths.

✓ **Creating a DataFrame from 2D dictionary having values as lists:**

```
dict1={'Students':['Neha','Maya','Reena'],
'Marks':[20,40,30],
'Sports':['Cricket', 'Football','Badminton']}
df1=pd.DataFrame(dict1)
print(df1)
```

	Students	Marks	Sports
0	Neha	20	Cricket
1	Maya	40	Football
2	Reena	30	Badminton

- The keys of the dictionary has become columns.
- The columns are placed in sorted order.
- The index is assigned automatically (0 onwards).

We can specify our own index too by using the index argument.

```
df2=pd.DataFrame(dict1,index=['I','II','III'])
print(df2)
```

	Students	Marks	Sports
I	Neha	20	Cricket
II	Maya	40	Football
III	Reena	30	Badminton

- The number of indexes given in the index sequence must match the length of the dictionary's values, otherwise Python will give error.

✓ **Creating a DataFrame from 2D dictionary having values as Series objects.**

- DataFrames are two dimensional representation of series.

```
smarks=pd.Series({'Neha':80,'Maya':90,'Reena':70})
sage=pd.Series({'Neha':25,'Maya':30,'Reena':29})
dict={'Marks':smarks,'Age':sage}
df3=pd.DataFrame(dict)
```

	Marks	Age
Neha	80	25
Maya	90	30
Reena	70	29

```
print(df3)
```

or

```
smarks=pd.Series([80,90,70],index=['Neha','Maya','Reena'])
sage=pd.Series([25,30,29],index=['Neha','Maya','Reena'])
dict={'Marks':smarks,'Age':sage}
df3=pd.DataFrame(dict)
print(df3)
```

```
       Marks  Age
Neha      80   25
Maya      90   30
Reena     70   29
```

- DataFrame object created has **columns assigned** from the **keys of the dictionary object** and its **index assigned** from the **indexes of the Series object** which are the values of the dictionary object.

➢ **Creating a DataFrame from list of dictionaries**

```
student=[{'Neha':50,'Manu':40},{'Neha':60,'Maya':45}]
df4=pd.DataFrame(student,index=['term1','term2'])
print(df4)
```

```
       Neha  Manu  Maya
term1    50  40.0   NaN
term2    60   NaN  45.0
```

- NaN is automatically added in missing places.

❖ Selecting or Accessing Data

```
import pandas as pd
dict={'BS':[80,98,100,65,72],'ACC':[88,67,93,50,90],
   'ECO':[100,75,89,40,96],'IP':[100,98,92,80,86]}
df5=pd.DataFrame(dict,index=['Ammu','Achu','Manu','Anu','Abu'])
print(df5)
```

```
      BS  ACC  ECO   IP
Ammu  80   88  100  100
Achu  98   67   75   98
Manu 100   93   89   92
Anu   65   50   40   80
Abu   72   90   96   86
```

➢ **Selecting / Accessing a column**

Syntax :

<dataframe object>[<column name>] Or <dataframe object>.<column name>

- In the dot notation make sure not to put any quotation marks around the column name.

```
print(df5.BS)
```

or

```
print(df5['BS'])
```

```
Ammu     80
Achu     98
Manu    100
Anu      65
Abu      72
Name: BS, dtype: int64
```

➢ **Selecting / Accessing multiple columns**

Syntax :

<dataframe object>[[<column name>,<column name>,…….]]

- Columns appear in the order of column names given in the list inside square brackets.

```
print(df5[['BS','IP']])
```

```
      BS   IP
Ammu  80  100
Achu  98   98
Manu 100   92
Anu   65   80
Abu   72   86
```

➢ **Selecting / Accessing a subset from a DataFrame using Row/Column names**

<dataframe object>.loc[<start row>:<end row>,<start column>:<end column>]

➢ **To access a row:**

<dataframe object>.loc[<row label>, :]

- Make sure not to miss the colon after comma.

```
print(df5.loc['Ammu', :])
```

```
BS      80
ACC     88
ECO    100
IP     100
Name: Ammu, dtype: int64
```

- **To access multiple rows:**
  <dataframe object>.loc[<start row>:<end row> , :]
  - Python will return all rows falling between start row and end row; along with start row and end row.
    print(df5.loc['Ammu':'Manu', :])
  - Make sure not to miss the colon after comma.

	BS	ACC	ECO	IP
Ammu	80	88	100	100
Achu	98	67	75	98
Manu	100	93	89	92

- **To access selective columns:**
  <dataframe object>.loc[: , <start column> : <end column>]
  - Lists all columns falling between start and end column.

```
print(df5.loc[:,'ACC':'IP'])
```

	ACC	ECO	IP
Ammu	88	100	100
Achu	67	75	98
Manu	93	89	92
Anu	50	40	80
Abu	90	96	86

  - Make sure not to miss the colon before comma.

- **To access range of columns from a range of rows:**
  <dataframe object>.loc[<start row> : <end row>,
  <start column> : <end column>]
  print(df5.loc['Manu':'Abu','ACC':'ECO'])

	ACC	ECO
Manu	93	89
Anu	50	40
Abu	90	96

- **Selecting / Accessing a subset from a DataFrame using Row/Column numeric index/position**
  Sometimes our dataframe object does not contain row or column labels or even we may not remember, then to extract subset from dataframe we can use iloc.
  <dataframe object>.iloc[<start row index> : <end row index>,
  [<start column index> : <end column index>]
  - When we use iloc, then end index is excluded.
    print(df5.iloc[1:3,1:3])

	ACC	ECO
Achu	67	75
Manu	93	89

- **Selecting / Accessing individual value**
  (i) Either give name of row or numeric index in square bracket of column name
  <dataframe object>.<column>[<row name or row numeric index>]
  print(df5.ACC['Achu']) 67
  or
  print(df5.ACC[1])
  (ii) Using at or iat

<dataframe object>.at[<row label>,<column label>]
Or
<dataframeobject>.iat[<numeric row index>,
<numeric column index>]

print(df5.at['Achu','ACC']) 67
or
print(df5.iat[1,1])

❖ Assigning / Modifying Data Values in DataFrame

- **To change or add a column**
  <dataframe object>[<column name>]=<new value>
  - If the given column name does not exist in dataframe then a new column with the name is added.
    df5['ENG']=60
    print(df5)

	BS	ACC	ECO	IP	ENG
Ammu	80	88	100	100	60
Achu	98	67	75	98	60
Manu	100	93	89	92	60
Anu	65	50	40	80	60
Abu	72	90	96	86	60

  - If you want to add a column that has different values for all its rows, then we can assign the data values for each row of the column in the form of a list.
    df5['ENG']=[50,60,40,30,70]
  - There are some other ways for adding a column to a database.
    <dataframe object>.at[: , <column name>]=value
    Or
    <dataframe object>.loc[: ,<column name>]=value
    df5.at[: ,'ENG']=60
    print(df5)
    or
    df5.loc[: ,'ENG']=60
    print(df5)

- **To change or add a row**
  <dataframe object>.at[rowname , :]=value
  or
  <dataframe object>.loc[rowname , :]=value
    df5.at['Sabu', :]=50
    print(df5)
    or
    df5.loc['Sabu', :]=50
    print(df5)

	BS	ACC	ECO	IP	ENG
Ammu	80.0	88.0	100.0	100.0	60.0
Achu	98.0	67.0	75.0	98.0	60.0
Manu	100.0	93.0	89.0	92.0	60.0
Anu	65.0	50.0	40.0	80.0	60.0
Abu	72.0	90.0	96.0	86.0	60.0
Sabu	50.0	50.0	50.0	50.0	50.0

  - If there is no row with such row label, then adds new row with this row label and assigns given values to all its columns.

- **To change or modify a single data value**
  <dataframe object>.<column>[<row label or row index>] = value

```
df5.BS['Ammu']=100
print(df5)
or
df5.BS[0]=100
print(df5)
```

```
        BS    ACC    ECO     IP   ENG
Ammu  100.0  88.0  100.0  100.0  60.0
Achu   98.0  67.0   75.0   98.0  60.0
Manu  100.0  93.0   89.0   92.0  60.0
Anu    65.0  50.0   40.0   80.0  60.0
Abu    72.0  90.0   96.0   86.0  60.0
Sabu   60.0  60.0   60.0   60.0  60.0
```

❖ Deleting columns in DataFrame

- We can use **del** statement, to delete a column
  del <dataframeobject>[<column name>]
  e.g.: del df5[‘ENG’]
- We can use **drop()** also to delete a column. **By default axis=0.**
  <dataframe object> = <dataframeobject>.drop([<columnname or index>],axis=1)
  Or
  <dataframe object> = <dataframeobject>.drop(columns=[<columnnames or indices>])

```
df5=df5.drop([‘ECO’], axis =1)
df5=df5.drop(columns=['ECO','IP'])
```

- We can use **pop()** to delete a column. The deleted column will be returned as Series object.

```
bstud=df5.pop(‘BS’)
print(bstud)
```

❖ Deleting rows in DataFrame

<dataframe object>=<dataframe object>.drop([index or sequence of index], axis=0)

```
df5=df5.drop(['Ammu','Achu'])
or
df5=df5.drop(index=['Ammu','Achu'])
```

Iterating over a DataFrame

- **Using pandas.iterrows() Function**
  - The method <DF>.**iterrows**() views a dataframe in the form of **horizontal** subset ie **row-wise.**
  - Each horizontal subset is in the form of (**row-index, Series**) where **Series contains all column values for that row –index.**
  - We can iterate over a Series object just as we iterate over other sequences.

```
import pandas as pd
dict={'BS':[80,98],'ACC':[88,67]}
df5=pd.DataFrame(dict,index=['Ammu','Achu'])
print(df5,"\n")

for (row,rowseries) in df5.iterrows():
```

```
      BS  ACC
Ammu  80   88
Achu  98   67

Row index: Ammu
containing
At position  0 : 80
At position  1 : 88

Row index: Achu
containing
At position  0 : 98
At position  1 : 67
```

```
print("Row index:",row)
print("containing")
i=0
for val in rowseries:
    print("At position ",i,":",val)
    i=i+1
print()
```

- **Using pandas.iteritems() Function**
  - The method <DF>.**iteritem**() views a dataframe in the form of **vertical** subset ie **column-wise**.
  - Each vertical subset is in the form of (**col-index, Series**) where **Series contains all row values for that column index**.

```
import pandas as pd
dict={'BS':[80,98],'ACC':[88,67]}
df5=pd.DataFrame(dict,index=['Ammu','Achu'])
print(df5,"\n")

for (column,columnseries) in df5.iteritems():
    print("Column index:",column)
    print("containing")
    i=0
    for val in columnseries:
        print("At row ",i,":",val)
        i=i+1
    print()
```

```
      BS  ACC
Ammu  80   88
Achu  98   67

Column index: BS
containing
At row  0 : 80
At row  1 : 98

Column index: ACC
containing
At row  0 : 88
At row  1 : 67
```

❖ Head and Tail Functions

- **head()**

<DF>.head([n=5])

  - To retrieve 5, top rows of a dataframe.
  - We can change the number of rows by specifying value for n.

```
df5.head(5)
df5.head(2)
```

- **tail()**
  - To retrieve 5, bottom rows of a dataframe.
  - We can change the number of rows by specifying value for n.

```
df5.tail(5)
df5.tail(2)
```

❖ Renaming index / column labels

- **rename()** renames the existing index or column labels in a dataframe/series.
- The old and new index/column labels are to be provided **in the form of a dictionary where keys are the old indexes/row labels and the values are the new names for the same.**

```
   p_id     p_name
0   101  Hard disk
1   102  Pen Drive

   Product_ID product_name
0         101    Hard disk
1         102    Pen Drive
```

Syntax:

<DF>.rename(index=None, columns=None, inplace=False)

where index and columns are dictionary like.

inplace, a boolean by default False (which returns a new dataframe with renamed index/labels).

If True then changes are made in the current dataframe.

```
import pandas as pd
dict={'p_id':[101,102],'p_name':['Hard disk','Pen Drive']}
df=pd.DataFrame(dict)
print(df,"\n")
#df.rename(columns={'p_id':'Product_ID','p_name':'product_name'},inplace=True)
#or
df=df.rename(columns={'p_id':'Product_ID','p_name':'product_name'})
print(df)
```

- Columns can also be renamed by using the **columns attribute** of dataframe.

```
   Product_ID product_name
0         101    Hard disk
1         102    Pen Drive
```

```
import pandas as pd
dict={'p_id':[101,102],'p_name':['Hard disk','Pen Drive']}
df=pd.DataFrame(dict)
df.columns=['Product_ID','product_name']
print(df,"\n")
```

❖ Reindexing

- **reindex()** used **to change the order** of the rows or columns in DataFrame/Series and **returns DataFrame/Series after changes.**

```
   product_name  Product_ID
0    Hard disk         101
1    Pen Drive         102
```

Syntax:

<DF>.reindex(index=None, columns=None, fill_value=NaN)

```
df=df.reindex(columns=['product_name','Product_ID'])
print(df)
```

- **If the mentioned indexes/columns do not exist in dataframe, these will be added as per the mentioned order with NaN values.**

```
  product_name  Product_ID  product_category
0    Hard disk         101               NaN
1    Pen Drive         102               NaN
```

```
df=df.reindex(columns=['product_name','Product_ID','product_category'])
print(df)
```

- By using **fill_value**, we can specify which will be filled in the newly added row/column.

```
  product_name  Product_ID product_category
1    Pen Drive         102             Home
0    Hard disk         101             Home
```

```
df=df.reindex(columns=['product_name','Product_ID','product_category'],
        index=[1,0],fill_value='Home')
print(df)
```

❖ Boolean indexing

- Like default indexing (0,1,2…) or labeled indexing , there is one more way to index – Boolean Indexing (Setting row index to True/ False etc.) .
- This helps in displaying the rows of Data Frame, according to True or False as specified in the command.

```
       p_id     p_name
True    101  Hard disk
False   102  Pen Drive
True    103     Camera
```

```
       p_id     p_name
True    101  Hard disk
True    103     Camera
```

```
import pandas as pd
dict={'p_id':[101,102,103],'p_name':['Hard disk','Pen Drive','Camera']}
df=pd.DataFrame(dict)
df.index=[True,False,True]
print(df,"\n")
print(df.loc[True])
```

❖ DataFrame attributes

All information related to a DataFrame object is available through attributes.

<DataFrane object> . <attribute name>

Attribute	Description
index	Returns the index (row labels) of the DataFrame
columns	Returns the column labels of the DataFrame
axes	Returns a list representing both the axes of the Data Frame (axis=0 i.e. index and axis=1 i.e. columns)
values	Returns a Numpy representation of the DataFrame
dtypes	Returns the dtypes of data in the DataFrame
shape	Returns tuple of the shape of the DataFrame
ndim	Returns number of dimensions of the dataframe
size	Returns the number of elements in the dataframe
empty	Returns True if the DataFrame object is empty, otherwise False
T	Transpose index and columns of DataFrame

Case study questions:

1. Consider the following Data Frame df and answer questions

	A	B	C
DEPT	CS	PROD	MEDICAL
EMPNO	101	102	103
ENAME	ABC	PQR	LMN
SALARY	200000	100000	20000

i. Write code to delete column B
ii. Write the output of the below code
print(df.tail(2))
iii. Write code to delete row salary
iv. Change the value of column A to 100
v. Change the value of DEPT of B to MECH
vi. Display DEPT and SALARY of column A and B
vii. Write code to rename column 'A' to 'D' which will not effect original dataframe
viii. Write code to add a column E with values [CS, 104,XYZ, 300000]
ix. Write code to add a row COMM with values [3000,4000,5000]
x. Write code to rename DEPT to DEPARTMENT which will effect the original dataframe
xi. Write code to display DEPT in A
    i. print(df.A['DEPT'])
    ii. print(df['A','DEPT'])
    iii. print(df.iloc[1:2,1:2])
    iv. print(df.iat[3,2])

xii. Write the output of the statement print(len(df))
    i. 3
    ii. 4
    iii. (4,3)
    iv. (3,4)

Answers :=

i. del df['A']
ii.
```
            A      B      C
ENAME      ABC    PQR    LMN
SALARY  200000  100000  20000
```
iii. df=df.drop(['SALARY'],axis=0)
iv. df['A']=100
v. df.B['DEPT']='MECH'
vi. print(df.loc[['DEPT','SALARY'],["A","B"]])
vii. df.rename(columns={"A":"D"},inplace=False)
viii. df['E']=["CS",104,"XYZ",300000]
ix. df.loc['COMM']=[3000,4000,5000]
x. df.rename(index={"DEPT":"DEPARTMENT"},inplace=True)
xi. print(df.A['DEPT'])
xii. 4

2. Consider the following Data Frame df and answer questions

	ACC	BST	ECO	IP
S1	90	91	92	93
S2	94	95	96	97
S3	98	99	100	100
S4	91	92	93	94

i. Create a new column total TOT by adding marks
ii. Find the highest marks scored by student s1
iii. Find the lowest marks scored by student s1
iv. Find the highest marks in ACC
v. Find the lowest marks in IP

Answers:=

i. df['TOT']=df['ACC']+df['BST']+df['ECO']+df['IP']
ii. print(max(df.loc['S1',:]))
iii. print(min(df.loc['S1',:]))
iv. print(max(df['ACC']))
v. print(min(df['IP']))

3. Consider the following Data Frame df and answer questions

	delhi	mumbai	kolkatta	chennai
hospitals	200	300	100	50
population	10	20	30	40
schools	250	350	400	200

i. Display details of city delhi and chennai
ii. Display hospitals in delhi
iii. Display shape of dataframe
iv. Change the population in kolkatta as 50
v. Rename the column population as “pop”

Answers:=

i. print(df[['delhi','chennai']])
ii. print(df.delhi['hospitals'])
iii. print(df.shape)
iv. df.kolkatta['population']=50
v. df.rename(index={"population":"pop"},inplace=True)

4. Consider the following Data Frame df and answer questions

```
          population  schools  hospitals
chennai           40      200        500
delhi             10      250        200
kolkatta          30      400        100
mumbai            20      350        300
>>>
```

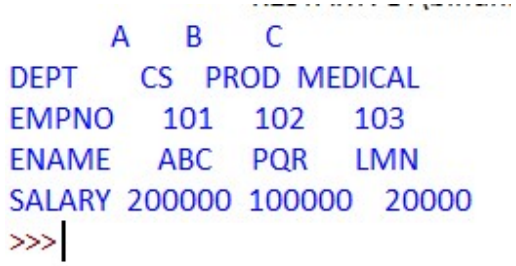
i. Display the name of city whose population >=20 range of 12 to 20
ii. Write command to set all vales of df as 0
iii. Display the df with rows in the reverse order
iv. Display the df with only columns in the reverse order
v. Display the df with rows & columns in the reverse order

Answers:-

i. print(df[df.population>=20])

ii. df[:]=0
iii. print(df.iloc[::-1)
iv. print(df.iloc[:,::-1])
v. print(df.iloc[::-1,::-1])

5. Consider the following Data Frame df and answer questions



Write the ouput of the following

i. print(len(df))
ii. print(df.count())
iii. print(df.count(1))
iv. print(min(df.loc['SALARY']))
v. print(max(df.loc['ENAME']))

Answers

i. 4
ii. A 4
B 4
C 4
dtype: int64
iii. DEPT 3
EMPNO 3
ENAME 3
SALARY 3
dtype: int64
iv. 20000
v. PQR

QUESTIONS ON DATAFRAME

1. What are the purpose of following statements-
1.df.columns
2. df.iloc[: , :-5]
3. df[2:8]
4. df[:]
5. df.iloc[: -4 , :]

Ans:

1. It displays the names of columns of the Dataframe.
2. It will display all columns except the last 5 columns.
3. It displays all columns with row index 2 to 7.
4. It will display entire dataframe with all rows and columns.
5. It will display all rows except the last 4 four rows

2. What will be the output of df.iloc[3:7,3:6]?

Ans:

It will display the rows with index 3 to 6 and columns with index 3 to 5 in a dataframe 'df'.

3. Write a python program to create a data frame with headings (CS and IP) from the list given below-

[[79,92][86,96],[85,91],[80,99]]

Ans:

```
l=[[10,20],[20,30],[30,40]]
df=pd.DataFrame(l,columns=['CS','IP'])
print(df)
```

4. Write python statement to delete the 3rd and 5th rows from dataframe df.

```
df1=df.drop(index=[2,4],axis=0)
        or
 df1=df.drop([2,4])
```

Sl No	MCQ QUESTIONS
1	To display the 3rd, 4th and 5th columns from the 6th to 9th rows of a dataframe you can write (a) DF.loc[6:9, 3:5] (b) DF.loc[6:10, 3:6] (c) DF.iloc[6:10, 3:6] (d) DF.iloc[6:9, 3:5] **ANS: c) DF.iloc[6:10, 3:6]**
2	We can add a new row to a DataFrame using the _____ method (i) rloc[ ] (ii) loc[ ] (iii)iloc[ ] (iv)None of the above **ANS: (ii) loc[ ]**
3	The head() function of dataframe will display how may rows from top if no parameter is passed. (i) 1 (ii) 3 (iii) 5 (iv) None of these **ANS : (iii) 5**
4	To change the 5th column's value at 3rd row as 35 in dataframe DF, you can write (a) DF[4, 6] = 35 (b) DF.iat[4, 6] = 35 (c) DF[3, 5] = 35

	(d) DF.iat[3, 5] = 35 **ANS:- d) DF.iat[3, 5] = 35**
5	Which function is used to find values from a DataFrame D using the index number? a) D.loc b) D.iloc c) D.index d) None of these **ANS: b) D.iloc**
6	In a DataFrame, Axis= 0 represents the elements a.rows b.columns c.both d.None of these. **ANS: a.rows**
7	In DataFrame, by default new column added as the _____ column (i) First (Left Side) (ii) Second (iii)Last (Right Side) (iv) Any where in dataframe **ANS: (iii)Last (Right Side)**
8	Which of the following is correct Features of DataFrame? a. Potentially columns are of different types b. Can Perform Arithmetic operations on rows and columns c. Labeled axes (rows and columns) d. All of the above **ANS: d. All of the above**
9	Write the code to append df2 with df1 a.Df2=Df2.append(Df1) b. Df2=Df2+Df1 c. Df2=Df2.appendwith.Df1 d. Df2=Df1.append(Df1) **ANS: a.Df2=Df2.append(Df1)**
10	When we create DataFrame from List of Dictionaries, then number of columns in DataFrame isequal to the _____ a. maximum number of keys in first dictionary of the list b. maximum number of different keys in all dictionaries of the list c. maximum number of dictionaries in the list

	d. None of the above **ANS: b. maximum number of different keys in all dictionaries of the list**
11	When we create DataFrame from List of Dictionaries, then dictionary keys will become _____ (i) Column labels (ii) Row labels (iii) Both of the above (iv) None of the above **ANS: (i) Column labels**
12	Which method is used to access vertical subset of a dataframe? (i) iterrows() (ii) iteritems() (iii) itercolumns() (iv) itercols() **ANS: (ii) iteritems()**
13	Write statement to transpose dataframe DF. (i) DF.t (ii) DF.transpose (iii)DF.T (iv)DF.T( ) **ANS: (iii)DF.T**
14	In DataFrame, by default new column added as the _____ column a. First (Left Side) b. Second c. Last (Right Side) d. Any where in dataframe **ANS: Last (Right Side)**
15	We can add a new row to a DataFrame using the _____ method (i) rloc[ ] (ii) loc[ ] (iii) iloc[ ] (iv) None of the above **ANS: (ii) loc[ ]**
16	Which of the following function is used to load the data from the CSV file to DataFrame? (i) read.csv( ) (ii) readcsv( ) (iii) read_csv( ) (iv) Read_csv( )

	ANS: (iii) read_csv( )
17	*Which of the following function is not a Boolean reduction function* *(i) Empty* *(ii) Any()* *(iii) All()* *(iv) Fillna()* ***ANS: (iv) Fillna()***
18	Which among the following options can be used to create a DataFrame in Pandas ? (a) A scalar value (b) An ndarray (c) A python dict (d) All of these **ANS:- (d) All of these**
19	Which attribute of a dataframe is used to convert row into columns and columns into rows in a dataframe? a) T b) ndim c) empty d) shape **ANS: a) T**
20	When we create DataFrame from List of Dictionaries, then number of columns in DataFrame is equal to the _____ (i) maximum number of keys in first dictionary of the list (ii) maximum number of different keys in all dictionaries of the list (iii) maximum number of dictionaries in the list (iv) None of the above **ANS: (ii) maximum number of different keys in all dictionaries of the list**
21	Which of the following is/are characteristics of DataFrame? a) Columns are of different types b) Can Perform Arithmetic operations c) Axes are labeled (rows and columns) d) All of the above **ANS: d) All of the above**
22	Write short code to show the information having city=”Delhi” from dataframe SHOP. (a) print(SHOP[City==’Delhi’]) (b) print(SHOP[SHOP.City==’Delhi’]) (c) print(SHOP[SHOP.’City’==’Delhi’]) (d) print(SHOP[SHOP[City]==’Delhi’])

	ANS: (b) print(SHOP[SHOP.City==’Delhi’])
23	Which of the following commands is used to install pandas? (i)pip install python –pandas (ii)pip install pandas (iii)python install python (iv)python install pandas **ANS: (ii) pip install pandas**
24	Which attribute of a dataframe is used to get number of axis? a.T b.Ndim c.Empty d.Shape **ANS: b.Ndim**
25	Display first row of dataframe ‘DF’ (i) print(DF.head(1)) (ii) print(DF[0 : 1]) (iii)print(DF.iloc[0 : 1]) (iv)All of the above **ANS: (iv)All of the above**
26	To delete a column from a DataFrame, you may use statement. (a) remove (b) del (c) drop (d) cancel statement. **ANS:- (b) del**
27	In given code dataframe ‘Df1’ has _____ rows and _____ columns import pandas as pd dict= [{‘a’:10, ‘b’:20}, {‘a’:5, ‘b’:10, ‘c’:20},{‘a’:7, ‘d’:10, ‘e’:20}] Df1 = pd.DataFrame(dict) (i) 3, 3 (ii) 3, 4 (iii)3, 5 (iv)None of the above **ANS: (iii)3, 5**
28	To delete a row from a DataFrame, you may use (a) remove (b) del (c) drop (d) cancel

	ANS:- (c) drop
29	In the following statement, if column 'mark' already exists in the DataFrame 'Df1' then the assignment statement will _____ Df1['mark'] = [95,98,100] #There are only three rows in DataFrame Df1 (i) Return error (ii) Replace the already existing values. (iii)Add new column (iv)None of the above ANS: (**ii) Replace the already existing values.**
30	To skip first 5 rows of CSV file, which argument will you give in read_csv( ) ? (a) skip_rows = 5 (b) skiprows = 5 (c) skip - 5 (d) noread - 5 **ANS:- (a) skip_rows = 5**
31	. Which of the following statement is false: i. DataFrame is size mutable ii. DataFrame is value mutable iii. DataFrame is immutable iv. DataFrame is capable of holding multiple types of data **ANS:- iii. DataFrame is immutable**
32	Which of the following statements is false? (i) Dataframe is size mutable (ii) Dataframe is value mutable (iii) Dataframe is immutable (iv) Dataframe is capable of holding multiple type of data A**NS: (iii) Dataframe is immutable**
33	To delete a row, the parameter axis of function drop( ) is assigned the value _____ (i) 0 (ii) 1 (iii) 2 (iv) 3 **ANS: (i) 0**
34	Which of the following function is used to load the data from the CSV file to DataFrame? (i) read.csv( ) (ii) readcsv( ) (iii)read_csv( )

	(iv)Read_csv( ) **ANS: (iii)read_csv( )**
35	Write code to delete rows those getting 5000 salary. (a) df=df.drop[salary==5000] (b) df=df[df.salary!=5000] (c) df.drop[df.salary==5000,axis=0] (d) df=df.drop[salary!=5000] **ANS: (b) df=df[df.salary!=5000]**
36	DF1.loc[ ] method is used to _____ # DF1 is a DataFrame (i) Add new row in a DataFrame 'DF1' (ii) To change the data values of a row to a particular value (iii)Both of the above (iv)None of the above **ANS: (iii)Both of the above**
37	To iterate over horizontal subsets of dataframe, (a) iterate( ) (b) iterrows( ) function may be used. (c) itercols( ) (d) iteritems( ) **ANS:- (b) iterrows( ) function may be used.**
38	Write code to delete the row whose index value is A1 from dataframe df. (a) df=df.drop('A1') (b) df=df.drop(index='A1') (c) df=df.drop('A1,axis=index') (d) df=df.del('A1') **ANS: (a) df=df.drop('A1')**
39	A two-dimension labeled array that is an ordered collection of columns to store heterogeneous data type is i. Series ii. ii. Numpy array iii. iii. Dataframe iv. iv. Panel **ANS:- iii. Dataframe**
40	To skip 1st, 3rd and 5th rows of CSV file, which argument will you give in read_csv( ) ? (a) skiprows = 11315 (b) skiprows - (1, 3, 5]

	(c) skiprows = [1, 5, 1] (d) Any of these **ANS:- (b) skiprows - (1, 3, 5]**
41	In Pandas _____ is used to store data in multiple columns. (i)Series (ii) DataFrame (iii) Both of the above (iv) None of the above **ANS: (ii) DataFrame**
42	What is dataframe? a. 2 D array with heterogeneous data b. 1 D array with homogeneous data c. 2 D array with homogeneous data d. 1 D array with heterogeneous data **ANS: a. 2 D array with heterogeneous data**
43	In a DataFrame, Axis= 1 represents the_____ elements (a) Row (b) Column (c) True (d) False **ANS: (b) Column**
44	Which of the following is not an attribute of a DataFrame Object ? a. index b. Index c. size d. value **ANS: b. Index**
45	To get top 5 rows of a dataframe, you may use (a) head( ) (b) head(5) (c) top( ) (d) top(5) **ANS:- (a) head( ) , b) head(5)**
46	27. To iterate over horizontal subsets of dataframe, (a) iterate( ) (b) iterrows( ) function may be used. (c) itercols( ) (d) iteritems( )

	ANS:- (b) iterrows( ) function may be used.
47	Write code to delete the row whose index value is A1 from dataframe df. (a) df=df.drop('A1') (b) df=df.drop(index='A1') (c) df=df.drop('A1,axis=index') (d) df=df.del('A1') **ANS: (a) df=df.drop('A1')**
48	A two-dimension labelled array that is an ordered collection of columns to store heterogeneous datatype is v. Series vi. ii. Numpy array vii. iii. Dataframe viii. iv. Panel **ANS:- iii. Dataframe**
49	To skip 1st, 3rd and 5th rows of CSV file, which argument will you give in read_csv( ) ? (a) skiprows = 11315 (b) skiprows - (1, 3, 5] (c) skiprows = [1, 5, 1] (d) Any of these **ANS:- (b) skiprows - (1, 3, 5]**
50	In a DataFrame, Axis= 1 represents the_____ elements (a) Row (b) Column (c) True (d) False **ANS: (b) Column**
51	NaN stands for: a. Not a Number b. None and None c. Null and Null d. None a Number **ANS: a. Not a Number**
52	To get top 5 rows of a dataframe, you may use (a) head( ) (b) head(5) (c) top( ) (d) top(5) **ANS:- (a) head( ) , b) head(5)**
53	The correct statement to read from a CSV file in a dataframeis : (a) .read_csv()

	(b) . read_csv( )() (c) = pandas.read() (d) = pandas.read_csv() **ANS:- (d) = pandas.read_csv()**
54	To delete a column from a dataframe, you may use _____ statement. i. remove() ii. ii. del() iii. iii. drop() iv. iv. cancel() **ANS:- iii. drop()**
55	The following code create a dataframe named 'Df1' with _____ columns. import pandas as pd Df1 = pd.DataFrame([10,20,30]) (i) 1 (ii) 2 (iii) 3 (iv) 4 **ANS: (i) 1**
56	To delete a row from dataframe, you may use _____ statement. i. remove() ii. ii. del() iii. iii. drop() iv. iv. cancel() **ANS:- ii. del()**
57	In a Data-Frame, Axis= 0 represents the elements along the_____ a. Row b. Column c. Row and Column Both d. None of the above **ANS: a. Row**
58	_____ method in Pandas can be used to change the index of rows and columns of a Series or Dataframe (a) rename() (b) reindex() (c) reframe() (d) none of these **ANS: (b) reindex()**
59	Write the single line command to delete the column "marks" from dataframe df using drop function. (a) df=df.drop(col='marks')

	(b) df=df.drop('marks',axis=col) (c) df=df.drop('marks',axis=0) (d) df=df.drop('marks',axis=1) **ANS: (d) df=df.drop('marks',axis=1)**
60	Which of the following is used to give user defined column index in DataFrame? (i) index (ii) column (iii) columns (iv) colindex **ANS: (iii) columns**
61	The following statement will _____ df = df.drop(['Name', 'Class', 'Rollno'], axis = 1) #df is a DataFrame object a. delete three columns having labels 'Name', 'Class' and 'Rollno' b. delete three rows having labels 'Name', 'Class' and 'Rollno' c. delete any three columns d. return error **ANS:- a. delete three columns having labels 'Name', 'Class' and 'Rollno'**
62	Difference between loc() and iloc().: a. Both are Label indexed based functions. b. Both are Integer position-based functions. c. loc() is label based function and iloc() integer position based function. d. loc() is integer position based function and iloc() index position based function. **ANS: c. loc() is label based function and iloc() integer position based function.**
63	Which command will be used to delete 3 and 5 rows of the data frame. Assuming the data frame name as DF. a. DF.drop([2,4],axis=0) b. DF.drop([2,4],axis=1) c. DF.drop([3,5],axis=1) d. DF.drop([3,5]) **ANS: a DF.drop([2,4],axis=0)**
64	Assuming the given structure, which command will give us the given output: **Output Required: (3,4)** (table) a. print(df.shape()) b. print(df.shape)

	EmpCode	Name	Desig
0	1405	VINAY	Clerk
1	1985	MANISH	Works Manager
2	1636	SMINA	Sales Manager
3	1689	RINU	Cleark

c. print(df.size)
d. print(df.size()).

ANS: b. print(df.shape)

65

Write the output of the given command: df1.loc[:0,'Name'] Consider the given dataframe.

	EmpCode	***Name***	***Desig***
0	*1405*	*VINAY*	*Clerk*
1	*1985*	*MANISH*	*Works Manager*
2	*1636*	*SMINA*	*Sales Manager*
3	*1689*	*RINU*	*Clerk*

a. 0 1405 VINAY Clerk
b. VINAY
c. Works Manager
d. Clerk
ANS : VINAY

Index

Importing/Exporting Data between CSV files and Data Frames

Basics of CSV Files

- CSV Files : A CSV file is a delimited value file. All CSV files are simple text files, can contain only numbers and letters and structure the data contained in them in tabular form.
- Files with the CSV extension are usually used to exchange data between different applications. Database programs, analytical software, and other applications that store large amounts of information (for example, contacts and customer data) usually support the CSV format.
- All CSV files have the same general format: each column is separated by a comma and each new row indicates a new row. Some programs that export data to a CSV file may use a different character to separate values, such as a tab, semicolon, or space.
- For example consider the data regarding medals won by India at Commonwealth Games 2022 stored in a csv file medaltally.csv created using a spreadsheet software like Microsoft Excel or Libreoffice Calc.

When opened with spreadsheet like Excel.

medaltally.csv

Sport	Gold medal	Silver medal	Bronze medal
Weightlifting	3	3	4
Judo	0	2	1
Lawn bowls	1	1	0
Table Tennis	3	1	1
Badminton	3	1	2
Squash	0	0	2
Para Powerlifting	1	0	0
Athletics	1	4	3
Wrestling	6	1	5
Boxing	3	1	3
Para Table Tennis	1	0	1
Hockey	0	1	1
Cricket	0	1	0

When opened with text editor like Notepad.

medaltally.csv

```
Sport,Gold medal,Silver medal,Bronze medal
Weightlifting,3,3,4
Judo,0,2,1
Lawn bowls,1,1,0
Table Tennis,3,1,1
Badminton,3,1,2
Squash,0,0,2
Para Powerlifting,1,0,0
Athletics,1,4,3
Wrestling,6,1,5
Boxing,3,1,3
Para Table Tennis,1,0,1
Hockey,0,1,1
Cricket,0,1,0
```

Importing CSV Files to Dataframes

pandas.read_csv is used to read a comma-separated values (csv) file into DataFrame.

pandas.read_csv(*filepath*, *sep=','*, *header='infer'*, *names=NoDefault.no_default*, *index_col=None, skiprows=None, skipfooter=0, nrows=None*)

Parameters

filepath :str, path object or file-like object
Any valid string path is acceptable. The string could be a URL.

sep :str, default ‘,’
Delimiter to use.

header :int, list of int, None, default ‘infer’
Row number(s) to use as the column names, and the start of the data. Default behavior is to infer the column names: if no names are passed the behavior is identical to header=0 and

column names are inferred from the first line of the file, if column names are passed explicitly then the behavior is identical to header=None.

names : array-like, optional

List of column names to use. If the file contains a header row, then you should explicitly pass `header=0` to override the column names. Duplicates in this list are not allowed.

index_col : int, str, sequence of int / str, or False, optional, default None

Column(s) to use as the row labels of the DataFrame, either given as string name or column index.

skiprows : list-like, int

Line numbers to skip (0-indexed) or number of lines to skip (int) at the start of the file.

nrows: int, optional

Number of rows of file to read.

Example 1 : Consider the following file medaltally.csv. Here header argument is 0 and index_col is None. Hence index becomes 0,1,2,3,...,12 and first row as column labels.

medaltally.csv

Sport	Gold medal	Silver medal	Bronze medal
Weightlifting	3	3	4
Judo	0	2	1
Lawn bowls	1	1	0
Table Tennis	3	1	1
Badminton	3	1	2
Squash	0	0	2
Para Powerlifting	1	0	0
Athletics	1	4	3
Wrestling	6	1	5
Boxing	3	1	3
Para Table Tennis	1	0	1
Hockey	0	1	1
Cricket	0	1	0

```
In [4]: import pandas
        medaltally = pandas.read_csv('medaltally.csv')
        print(medaltally)
```

```
                 Sport  Gold medal  Silver medal  Bronze medal
0       Weightlifting           3             3             4
1               Judo            0             2             1
2          Lawn bowls           1             1             0
3        Table Tennis           3             1             1
4           Badminton           3             1             2
5              Squash           0             0             2
6   Para Powerlifting           1             0             0
7           Athletics           1             4             3
8           Wrestling           6             1             5
9              Boxing           3             1             3
10  Para Table Tennis           1             0             1
11            Hockey            0             1             1
12           Cricket            0             1             0
```

Example 2 : Consider the following file medaltally.csv. Here header argument is 0 and index_col is 0. Hence, first column values are taken as index labels and first row as column labels.

medaltally.csv

Sport	Gold medal	Silver medal	Bronze medal
Weightlifting	3	3	4
Judo	0	2	1
Lawn bowls	1	1	0
Table Tennis	3	1	1
Badminton	3	1	2
Squash	0	0	2
Para Powerlifting	1	0	0
Athletics	1	4	3
Wrestling	6	1	5
Boxing	3	1	3
Para Table Tennis	1	0	1
Hockey	0	1	1
Cricket	0	1	0

```
In [4]: import pandas
        medaltally2 = pandas.read_csv('medaltally.csv', index_col=0)
        print(medaltally2)
```

```
                   Gold medal  Silver medal  Bronze medal
Sport
Weightlifting               3             3             4
Judo                        0             2             1
Lawn bowls                  1             1             0
Table Tennis                3             1             1
Badminton                   3             1             2
Squash                      0             0             2
Para Powerlifting           1             0             0
Athletics                   1             4             3
Wrestling                   6             1             5
Boxing                      3             1             3
Para Table Tennis           1             0             1
Hockey                      0             1             1
Cricket                     0             1             0
```

Note: 'Sport' is not an index. It is medaltally2.index.name

Example 3 : Consider the following file medaltally2.csv with the data regarding some events missing. Those missing data are read as NaN.

medaltally2.csv

Sport	Gold medal	Silver medal	Bronze medal
Weightlifting	3	3	4
Judo	0	2	1
Lawn bowls	1	1	0
Table Tennis	3		1
Badminton	3	1	
Squash	0	0	2
Para Powerlifting	1	0	0
Athletics	1	4	
Wrestling	6	1	5
Boxing		1	3
Para Table Tennis	1		1
Hockey	0	1	1
Cricket	0		0

```
In [11]: import pandas
         medaltally2 = pandas.read_csv('medaltally2.csv', index_col=0)
         print(medaltally2)

                           Gold medal  Silver medal  Bronze medal
         Sport
         Weightlifting             3.0           3.0           4.0
         Judo                      0.0           2.0           1.0
         Lawn bowls                1.0           1.0           0.0
         Table Tennis              3.0           NaN           1.0
         Badminton                 3.0           1.0           NaN
         Squash                    0.0           0.0           2.0
         Para Powerlifting         1.0           0.0           0.0
         Athletics                 1.0           4.0           NaN
         Wrestling                 6.0           1.0           5.0
         Boxing                    NaN           1.0           3.0
         Para Table Tennis         1.0           NaN           1.0
         Hockey                    0.0           1.0           1.0
         Cricket                   0.0           NaN           0.0
```

Note: 'Sport' is not an index. It is medaltally2.index.name

Example 4 : Consider the following file medaltally.csv read with only first 5 rows to be read.

medaltally.csv

Sport	Gold medal	Silver medal	Bronze medal
Weightlifting	3	3	4
Judo	0	2	1
Lawn bowls	1	1	0
Table Tennis	3	1	1
Badminton	3	1	2
Squash	0	0	2
Para Powerlifting	1	0	0
Athletics	1	4	3
Wrestling	6	1	5
Boxing	3	1	3
Para Table Tennis	1	0	1
Hockey	0	1	1
Cricket	0	1	0

```
import pandas
medaltally = pandas.read_csv('medaltally.csv', index_col=0, header = 0, nrows = 5)
print(medaltally)

              Gold medal  Silver medal  Bronze medal
Sport
Weightlifting           3             3             4
Judo                    0             2             1
Lawn bowls              1             1             0
Table Tennis            3             1             1
Badminton               3             1             2
```

Example 5 : Consider the following file medaltally.csv to be read with rows 1,3,4,5 to be skipped.

medaltally.csv

Sport	Gold medal	Silver medal	Bronze medal
Weightlifting	3	3	4
Judo	0	2	1
Lawn bowls	1	1	0
Table Tennis	3	1	1
Badminton	3	1	2
Squash	0	0	2
Para Powerlifting	1	0	0
Athletics	1	4	3
Wrestling	6	1	5
Boxing	3	1	3
Para Table Tennis	1	0	1
Hockey	0	1	1
Cricket	0	1	0

```
import pandas
medaltally = pandas.read_csv('medaltally.csv', index_col=0, header = 0, skiprows = [1,3,4,5])
print(medaltally)

                  Gold medal  Silver medal  Bronze medal
Sport
Judo                       0             2             1
Squash                     0             0             2
Para Powerlifting          1             0             0
Athletics                  1             4             3
Wrestling                  6             1             5
Boxing                     3             1             3
Para Table Tennis          1             0             1
Hockey                     0             1             1
Cricket                    0             1             0
```

Here rows numbers are taken as weightlifting 1, Judo 2, and so on.

Example 6 Consider the following file medaltally.csv to be read with user specified column names 'Gold', 'Silver', and 'Bronze'.

medaltally.csv

Sport	Gold medal	Silver medal	Bronze medal
Weightlifting	3	3	4
Judo	0	2	1
Lawn bowls	1	1	0
Table Tennis	3	1	1
Badminton	3	1	2
Squash	0	0	2
Para Powerlifting	1	0	0
Athletics	1	4	3
Wrestling	6	1	5
Boxing	3	1	3
Para Table Tennis	1	0	1
Hockey	0	1	1
Cricket	0	1	0

```
import pandas
medaltally = pandas.read_csv('medaltally.csv', index_col=0, header = 0, names=['Gold', 'Silver', 'Bronze'])
print(medaltally)
```

```
                  Gold  Silver  Bronze
Weightlifting        3       3       4
Judo                 0       2       1
Lawn bowls           1       1       0
Table Tennis         3       1       1
Badminton            3       1       2
Squash               0       0       2
Para Powerlifting    1       0       0
Athletics            1       4       3
Wrestling            6       1       5
Boxing               3       1       3
Para Table Tennis    1       0       1
Hockey               0       1       1
Cricket              0       1       0
```

Note : Here rows numbers are taken as weightlifting 1, Judo 2, and so on.

Example 7 : Consider the following file processed5_medaltally.csv to be read with separator as '#'.

processed5_medaltally.csv

```
Sport#Gold medal#Silver medal#Bronze medal#Total Event wise
Weightlifting#3.0#3.0#4.0#10.0
Judo#0.0#2.0#1.0#3.0
Lawn bowls#1.0#1.0#0.0#2.0
Table Tennis#3.0#1.0#1.0#5.0
Badminton#3.0#1.0#2.0#6.0
Squash#0.0#0.0#2.0#2.0
Para Powerlifting#1.0#0.0#0.0#1.0
Athletics#1.0#4.0#3.0#8.0
Wrestling#6.0#1.0#5.0#12.0
Boxing#3.0#1.0#3.0#7.0
Para Table Tennis#1.0#0.0#1.0#2.0
Hockey#0.0#1.0#1.0#2.0
Cricket#0.0#1.0#0.0#1.0
Total Medal wise#22.0#16.0#23.0#61.0
```

```
import pandas
medaltally = pandas.read_csv('processed5_medaltally.csv', index_col=0, header = 0, sep = '#')
print(medaltally)
```

```
                  Gold medal  Silver medal  Bronze medal  Total Event wise
Sport
Weightlifting            3.0           3.0           4.0              10.0
Judo                     0.0           2.0           1.0               3.0
Lawn bowls               1.0           1.0           0.0               2.0
Table Tennis             3.0           1.0           1.0               5.0
Badminton                3.0           1.0           2.0               6.0
Squash                   0.0           0.0           2.0               2.0
Para Powerlifting        1.0           0.0           0.0               1.0
Athletics                1.0           4.0           3.0               8.0
Wrestling                6.0           1.0           5.0              12.0
Boxing                   3.0           1.0           3.0               7.0
Para Table Tennis        1.0           0.0           1.0               2.0
Hockey                   0.0           1.0           1.0               2.0
Cricket                  0.0           1.0           0.0               1.0
Total Medal wise        22.0          16.0          23.0              61.0
```

Exporting Dataframes to CSV Files

DataFrame.to_csv(*path = None***, *sep=','*, *na_rep=''*, *header=True*, *index=True*)**

DataFrame.to_csv is used to write a pandas object to a comma-separated values (csv) file.Parameters

path : str, default None
String
sep :str, default ‘,’
String of length 1. Field delimiter for the output file.
na_rep : str, default ‘’
Missing data representation.
header : bool or list of str, default True
Write out the column names. If a list of strings is given it is assumed to be aliases for the column names.
index :bool, default True
Write row names (index).

Example 1 : Consider the dataframe medaltally. Here header and index arguments are True; Hence the index and column labels are written to the csv file processed_medaltally.csv.

```
import pandas
print(medaltally)
medaltally.to_csv('processed_medaltally.csv', index = True , header = True)
```

```
                Gold medal  Silver medal  Bronze medal  Total Event wise
Sport
Weightlifting          3.0           3.0           4.0              10.0
Judo                   0.0           2.0           1.0               3.0
Lawn bowls             1.0           1.0           0.0               2.0
Table Tennis           3.0           1.0           1.0               5.0
Badminton              3.0           1.0           2.0               6.0
Squash                 0.0           0.0           2.0               2.0
Para Powerlifting      1.0           0.0           0.0               1.0
Athletics              1.0           4.0           3.0               8.0
Wrestling              6.0           1.0           5.0              12.0
Boxing                 3.0           1.0           3.0               7.0
Para Table Tennis      1.0           0.0           1.0               2.0
Hockey                 0.0           1.0           1.0               2.0
Cricket                0.0           1.0           0.0               1.0
Total Medal wise      22.0          16.0          23.0              61.0
```

processed_medaltally.csv

Sport	Gold medal	Silver medal	Bronze medal	Total Event wise
Weightlifting	3	3	4	10
Judo	0	2	1	3
Lawn bowls	1	1	0	2
Table Tennis	3	1	1	5
Badminton	3	1	2	6
Squash	0	0	2	2
Para Powerlifting	1	0	0	1
Athletics	1	4	3	8
Wrestling	6	1	5	12
Boxing	3	1	3	7
Para Table Tennis	1	0	1	2
Hockey	0	1	1	2
Cricket	0	1	0	1
Total Medal wise	22	16	23	61

Example 2 : Consider the dataframe medaltally. Here header is False and index argument is True; Hence the columns labels are not present in file processed2_medaltally.csv.

```
import pandas
print(medaltally)
medaltally.to_csv('processed2_medaltally.csv', index = True, header = False)
```

```
                Gold medal  Silver medal  Bronze medal  Total Event wise
Sport
Weightlifting          3.0           3.0           4.0              10.0
Judo                   0.0           2.0           1.0               3.0
Lawn bowls             1.0           1.0           0.0               2.0
Table Tennis           3.0           1.0           1.0               5.0
Badminton              3.0           1.0           2.0               6.0
Squash                 0.0           0.0           2.0               2.0
Para Powerlifting      1.0           0.0           0.0               1.0
Athletics              1.0           4.0           3.0               8.0
Wrestling              6.0           1.0           5.0              12.0
Boxing                 3.0           1.0           3.0               7.0
Para Table Tennis      1.0           0.0           1.0               2.0
Hockey                 0.0           1.0           1.0               2.0
Cricket                0.0           1.0           0.0               1.0
Total Medal wise      22.0          16.0          23.0              61.0
```

processed2_medaltally.csv

Weightlifting	3	3	4	10
Judo	0	2	1	3
Lawn bowls	1	1	0	2
Table Tennis	3	1	1	5
Badminton	3	1	2	6
Squash	0	0	2	2
Para Powerlifting	1	0	0	1
Athletics	1	4	3	8
Wrestling	6	1	5	12
Boxing	3	1	3	7
Para Table Tennis	1	0	1	2
Hockey	0	1	1	2
Cricket	0	1	0	1
Total Medal wise	22	16	23	61

Example 3 : Consider the dataframe medaltally. Here header is True and index argument is False; Hence the index labels are not present in the file processed3_medaltally.csv but column labels are present.

```
import pandas
print(medaltally)
medaltally.to_csv('processed3_medaltally.csv', index = False , header = True)
```

```
                Gold medal  Silver medal  Bronze medal  Total Event wise
Sport
Weightlifting          3.0           3.0           4.0              10.0
Judo                   0.0           2.0           1.0               3.0
Lawn bowls             1.0           1.0           0.0               2.0
Table Tennis           3.0           1.0           1.0               5.0
Badminton              3.0           1.0           2.0               6.0
Squash                 0.0           0.0           2.0               2.0
Para Powerlifting      1.0           0.0           0.0               1.0
Athletics              1.0           4.0           3.0               8.0
Wrestling              6.0           1.0           5.0              12.0
Boxing                 3.0           1.0           3.0               7.0
Para Table Tennis      1.0           0.0           1.0               2.0
Hockey                 0.0           1.0           1.0               2.0
Cricket                0.0           1.0           0.0               1.0
Total Medal wise      22.0          16.0          23.0              61.0
```

processed3_medaltally.csv

Gold medal	Silver medal	Bronze medal	Total Event wise
3	3	4	10
0	2	1	3
1	1	0	2
3	1	1	5
3	1	2	6
0	0	2	2
1	0	0	1
1	4	3	8
6	1	5	12
3	1	3	7
1	0	1	2
0	1	1	2
0	1	0	1
22	16	23	61

Example 4 : Consider the dataframe medaltally. Here header is False and index argument is False; Hence the index labels as well as the header are not present in written file processed4_medaltally.csv. Only data is present in the file processed4_medaltally.csv.

```
import pandas
print(medaltally)
medaltally.to_csv('processed4_medaltally.csv', index = False , header = False)
```

Sport	Gold medal	Silver medal	Bronze medal	Total Event wise
Weightlifting	3.0	3.0	4.0	10.0
Judo	0.0	2.0	1.0	3.0
Lawn bowls	1.0	1.0	0.0	2.0
Table Tennis	3.0	1.0	1.0	5.0
Badminton	3.0	1.0	2.0	6.0
Squash	0.0	0.0	2.0	2.0
Para Powerlifting	1.0	0.0	0.0	1.0
Athletics	1.0	4.0	3.0	8.0
Wrestling	6.0	1.0	5.0	12.0
Boxing	3.0	1.0	3.0	7.0
Para Table Tennis	1.0	0.0	1.0	2.0
Hockey	0.0	1.0	1.0	2.0
Cricket	0.0	1.0	0.0	1.0
Total Medal wise	22.0	16.0	23.0	61.0

processed4_medaltally.csv

3	3	4	10
0	2	1	3
1	1	0	2
3	1	1	5
3	1	2	6
0	0	2	2
1	0	0	1
1	4	3	8
6	1	5	12
3	1	3	7
1	0	1	2
0	1	1	2
0	1	0	1
22	16	23	61

Example 5 : Consider the dataframe medaltally. The default delimiter ',' in a csv file can be changed by setting the sep argument in to_csv. Here delimiter is changed to '#'.

```
import pandas
print(medaltally)
medaltally.to_csv('processed5_medaltally.csv', index = True , header = True, sep='#')
```

Sport	Gold medal	Silver medal	Bronze medal	Total Event wise
Weightlifting	3.0	3.0	4.0	10.0
Judo	0.0	2.0	1.0	3.0
Lawn bowls	1.0	1.0	0.0	2.0
Table Tennis	3.0	1.0	1.0	5.0
Badminton	3.0	1.0	2.0	6.0
Squash	0.0	0.0	2.0	2.0
Para Powerlifting	1.0	0.0	0.0	1.0
Athletics	1.0	4.0	3.0	8.0
Wrestling	6.0	1.0	5.0	12.0
Boxing	3.0	1.0	3.0	7.0
Para Table Tennis	1.0	0.0	1.0	2.0
Hockey	0.0	1.0	1.0	2.0
Cricket	0.0	1.0	0.0	1.0
Total Medal wise	22.0	16.0	23.0	61.0

File opened in text editor

processed5_medaltally.csv

Sport#Gold medal#Silver medal#Bronze medal#Total Event wise
Weightlifting#3.0#3.0#4.0#10.0
Judo#0.0#2.0#1.0#3.0
Lawn bowls#1.0#1.0#0.0#2.0
Table Tennis#3.0#1.0#1.0#5.0
Badminton#3.0#1.0#2.0#6.0
Squash#0.0#0.0#2.0#2.0
Para Powerlifting#1.0#0.0#0.0#1.0
Athletics#1.0#4.0#3.0#8.0
Wrestling#6.0#1.0#5.0#12.0
Boxing#3.0#1.0#3.0#7.0
Para Table Tennis#1.0#0.0#1.0#2.0
Hockey#0.0#1.0#1.0#2.0
Cricket#0.0#1.0#0.0#1.0
Total Medal wise#22.0#16.0#23.0#61.0

Exercises

1. Given the following csv file which of the command will correctly read the details into a dataframe from csv file sectors_economy.csv.

sectors_economy.csv

	1953-54	1970-71	1990-91	2014-15
Household	10	12	12	23
Agriculture	1	3	8	18
Industries	40	50	45	44
Transport	44	28	22	2
Service	5	7	13	13

a. pandas.read_csv('sectors_economy.csv', header = 0, index_col = 0)

b. pandas.read_csv('sectors_economy.csv', header = 0, index_col = 1)

c. pandas.read_csv('sectors_economy.csv', header = 1, index_col = 0)

d. pandas.read_csv('sectors_economy.csv', header = 1, index_col = 1)

Consider the following dataframe ***literacy*** for questions 2 and 3.

```
In [15]: literacy
Out[15]:
```

	2001	2011
Andhra Pradesh	60.35	67.65
Karnataka	66.50	75.50
Kerala	90.95	94.00
Tamil Nadu	73.40	80.35

2. Which of the commands will correctly write literacy to literacy.csv with index and column labels.

a. literacy.to_csv("literacy.csv", index = False, header = False)

b. literacy.to_csv("literacy.csv", index = True, header = False)

c. literacy.to_csv("literacy.csv", index = False, header = True)

d. literacy.to_csv("literacy.csv", index = True, header = True)

3. Which of the commands will correctly write literacy to literacy.csv without index and column labels i.e the data alone.

a. literacy.to_csv("literacy.csv", index = False, header = False)

b. literacy.to_csv("literacy.csv", index = True, header = False)

c. literacy.to_csv("literacy.csv", index = False, header = True)

d. literacy.to_csv("literacy.csv", index = True, header = True)

4. Which of the arguments needs to be set so that the index labels will be written into literacy.csv.

a. index b. header c. index_col d. header_row

5. The argument to be set in read_csv for reading user specified number of rows from a csv file is:

a. rows b. row c. nrows d. nrow

Answers

1. a

2. d

3. a

4. a

5. c