New syllabus 2021-22

Chapter 2

Data Handling using Pandas -2

Informatics Practices

Class XII (As per CBSE Board)

Visit : python.mykvs.in for regular updates

Descriptive statistics

Descriptive statistics are used to describe / summarize large data in ways that are meaningful and useful. Means “must knows” with any set of data. It gives us a general idea of trends in our data including:

- The mean, mode, median and range.
- Variance and standard deviation ,quartile
- SumCount, maximum and minimum.

Descriptive statistics is useful because it allows us take decision. For example, let’s say we are having data on the incomes of one million people. No one is going to want to read a million pieces of data; if they did, they wouldn’t be able to get any useful information from it. On the other hand, if we summarize it, it becomes useful: an average wage, or a median income, is much easier to understand than reams of data.

Steps to Get the descriptive statistics

- Step 1: Collect the Data

  Either from data file or from user

- Step 2: Create the DataFrame

  Create dataframe from pandas object

- Step 3: Get the Descriptive Statistics for Pandas DataFrame

  Get the descriptive statistics as per requirement like mean,mode,max,sum etc. from pandas object

Note :- Dataframe object is best for descriptive statistics as it can hold large amount of data and relevant functions.

Descriptive statistics - dataframe

Pandas dataframe object come up with the methods to calculate max, min, count, sum, mean, median, mode, quartile, Standard deviation, variance.

Mean

Mean is an average of all the numbers. The steps required to calculate a mean are:

- sum up all the values of a target variable in the dataset
- divide the sum by the number of values

Descriptive statistics - dataframe

Median- Median is the middle value of a sorted list of numbers. The steps required to get a median from a list of numbers are:

- sort the numbers from smallest to highest
- if the list has an odd number of values, the value in the middle position is the median
- if the list has an even number of values, the average of the two values in the middle will be the median

Mode-To find the mode, or modal value, it is best to put the numbers in order. Then count how many of each number. A number that appears most often is the mode.e.g.{19, 8, 29, 35, 19, 28, 15}. Arrange them in order: {8, 15, 19, 19, 28, 29, 35} .19 appears twice, all the rest appear only once, so 19 is the mode.

Having two modes is called "**bimodal**".Having more than two modes is called "**multimodal**".

Descriptive statistics - dataframe

#e.g. program for data aggregation/descriptive statistics

```
from pandas import DataFrame

Cars = {'Brand': ['Maruti ciaz','Ford ','Tata Indigo','Toyota Corolla','Audi
A9'],
       'Price': [22000,27000,25000,29000,35000],
        'Year': [2014,2015,2016,2017,2018]
       }
```

STEP1

```
df = DataFrame(Cars, columns= ['Brand', 'Price','Year'])
```

STEP2

```
stats_numeric = df['Price'].describe().astype (int)
print (stats_numeric)
```

STEP3

```
#describe method return mean,standard deviationm,min,max,
% values
```

OUTPUT

```
count       5
mean     27600
std     4878
min      22000
25%      25000
50%      27000
75%      29000
max      35000
Name: Price, dtype:
int32
```

Descriptive statistics - dataframe

#e.g. program for data aggregation/descriptive statistics

```
import pandas as pd
import numpy as np
#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
   'Age':pd.Series([26,25,25,24,31]),
   'Score':pd.Series([87,67,89,55,47])}
```

STEP1

```
#Create a DataFrame
df = pd.DataFrame(d)
print("Dataframe contents")
print (df)
```

STEP2

```
print(df.count())
print("count age",df[['Age']].count())
print("sum of score",df[['Score']].sum())
print("minimum age",df[['Age']].min())
print("maximum score",df[['Score']].max())
print("mean age",df[['Age']].mean())
print("mode of age",df[['Age']].mode())
print("median of score",df[['Score']].median())
```

STEP3

OUTPUT

```
Dataframe contents
     Name  Age  Score
0   Sachin  26     87
1    Dhoni  25     67
2    Virat  25     89
3    Rohit  24     55
4  Shikhar  31     47
Name     5
Age      5
Score    5
dtype: int64
count age Age    5
dtype: int64
sum of score Score    345
dtype: int64
minimum age Age    24
dtype: int64
maximum score Score    89
dtype: int64
mean age Age    26.2
dtype: float64
mode of age    Age
0   25
median of score Score    67.0
dtype: float64
```

Descriptive statistics - dataframe

Quantile -

Quantile statistics is a part of a data set. It is used to describe data in a clear and understandable way.The 0,30 quantile is basically saying that 30 % of the observations in our data set is below a given line. On the other hand ,it is also stating that there are 70 % remaining above the line we set.

Common Quantiles

Certain types of quantiles are used commonly enough to have specific names. Below is a list of these:

- The 2 quantile is called the median
- The 3 quantiles are called terciles
- The 4 quantiles are called quartiles
- The 5 quantiles are called quintiles
- The 6 quantiles are called sextiles
- The 7 quantiles are called septiles
- The 8 quantiles are called octiles
- The 10 quantiles are called deciles
- The 12 quantiles are called duodeciles
- The 20 quantiles are called vigintiles
- The 100 quantiles are called percentiles
- The 1000 quantiles are called permilles

Quantiles

The word “quantile” comes from the word quantity. means, a quantile is where a sample is divided into equal-sized or subgroups (that’s why it’s sometimes called a “fractile“). So that’s why ,It can also refer to dividing a probability distribution into areas of equal probability.

The median is a kind of quantile; the median is placed in a probability distribution at center so that exactly half of the data is lower than the median and half of the data is above the median. The median cuts a distribution into two equal parts and so why sometimes it is called 2-quantile.

Quartiles are quantiles; when they divide the distribution into four equal parts. Deciles are quantiles that divide a distribution into 10 equal parts and Percentiles when that divide a distribution into 100 equal parts .

Quantiles

How to Find Quantiles?

Sample question: Find the number in the following set of data where 30 percent of values fall below it, and 70 percent fall above:
2 4 5 7 9 11 12 17 19 21 22 31 35 36 45 44 55 68 79 80 81 88 90 91 92 100 112 113 114 120 121 132 145 148 149 152 157 170 180 190

Step 1: Order the data from smallest to largest. The data in the question is already in ascending order.

Step 2: Count how many observations you have in your data set. this particular data set has 40 items.

Step 3: Convert any percentage to a decimal for “q”. We are looking for the number where 30 percent of the values fall below it, so convert that to .3.

Step 4: Insert your values into the formula:
ith observation = q (n + 1)
ith observation = .3 (40 + 1) = 12.3

Answer: The ith observation is at 12.3, so we round down to 12 (remembering that this formula is an estimate). The 12th number in the set is 31, which is the number where 30 percent of the values fall below it.

Quantiles

How to Find Quartiles in python

In pandas series object->

```
import pandas as pd
import numpy as np
s = pd.Series([1, 2, 4, 5,6,8,10,12,16,20])
r=s.quantile([0.25,0.5,0.75])
print(r)
```

OUTPUT

```
0.25     4.25
0.50     7.00
0.75    11.50
dtype: float64
```

How to Find Quartiles in python

```
#Program in python to find 0.25 quantile of
series[1, 10, 100, 1000]
import pandas as pd
import numpy as np
s = pd.Series([1, 10, 100, 1000])
r=s.quantile(.25)
print(r)
```

OUTPUT 7.75

Solution steps

1. q=0.25 (0.25 quantile)
2. n = 4 (no of elements)

=(n-1)*q+1

=(4-1)*0.25+1

=3*0.25+1

=0.75+1

=1.75

2.Now integer part is a=1 and fraction part is 0.75 and T is term. Now formula for quantile is

=T1+b*(T2-T1)

=1+0.75*(10-1)

=1+0.75*9

=1+6.75 = 7.75 Quantile is 7.75

Note:- That in series [1, 10, 100, 1000] 1 is at 1 position 10 is at 2, 100 is at 3 and so on.Here we are choosing T1 as 1 because at 1 position (integer part of 1.75 is 1) value is 1(T1).here we are choosing value and then next fraction part is between 1 to 10,that is being found by 0.75*(10-1).Its result is 6.75 next to 1.Thats why we are adding 1 with 6.75.

Standard Deviation

standard deviation means measure the amount of variation / dispersion of a set of values.A low standard deviation means the values tend to be close to the mean in a set and a high standard deviation means the values are spread out over a wider range. Standard deviation is the most important concepts as far as finance is concerned. Finance and banking is all about measuring the risk and standard deviation measures risk. Standard deviation is used by all portfolio managers to measure and track risk.

Steps to calculate the standard deviation:

1. Work out the Mean (the simple average of the numbers)
2. Then for each number:subtract the Mean and square the result
3. Then work out the mean of those squared differences.
4. Take the square root of that and we are done!

Standard Deviation

E.g. Std deviation for (9, 2, 12, 4, 5, 7)

Step 1. Work out the mean -(9+2+12+4+5+7) / 6 = 39/6 = 6.5

Step 2. Then for each number: subtract the Mean and square the result - $(9 - 6.5)^2 = (2.5)^2 = 6.25$, $(2 - 6.5)^2 = (-4.5)^2 = 20.25$

Perform same operation for all remaining numbers.

Step 3. Then work out the mean of those squared differences.

Sum = 6.25 + 20.25 + 2.25 + 6.25 + 30.25 + 0.25 = 65.5

Divide by N-1: (1/5) × 65.5 = 13.1(This value is "**Sample Variance**“)

Step 4. Take the square root of that: s = √(13.1) = 3.619...(stddev)

formula for Standard Deviation

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

Above e.g. is for practice purpose otherwise stddev is performed for large amount of data

Standard Deviation

E.g. Std deviation for (9, 2, 12, 4, 5, 7)

```
import pandas as pd
import numpy as np

#Create a DataFrame
info = {
   'Name':['Mohak','Freya','Viraj','Santosh','Mishti','Subrata'],
   'Marks':[9, 2, 12, 4, 5, 7]}
data = pd.DataFrame(info)
# standard deviation of the dataframe
r=data.std()
print(r)
```

OUTPUT

```
Marks    3.619392
dtype: float64
```

Descriptive statistics - dataframe

var() – Variance Function in python pandas is used to calculate variance of a given set of numbers, Variance of a data frame, Variance of column and Variance of rows, let’s see an example of each.

```
#e.g.program
import pandas as pd
import numpy as np
#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
   'Age':pd.Series([26,25,25,24,31]),
   'Score':pd.Series([87,67,89,55,47])}
#Create a DataFrame
df = pd.DataFrame(d)
print("Dataframe contents")
print (df)
print(df.var())
#df.loc[:,“Age"].var() for variance of specific column
#df.var(axis=0) column variance
#df.var(axis=1) row variance
```

Dataframe Operations

Data aggregation – Aggregation is the process of turning the values of a dataset (or a subset of it) into one single value or data aggregation is a multivalued function ,which require multiple values and return a single value as a result.There are number of aggregations possible like count,sum,min,max,median,quartile etc. These(count,sum etc.) are descriptive statistics and other related operations on DataFrame Let us make this clear! If we have a DataFrame like…

	Name	Age	Score
0	Sachin	26	87
1	Dhoni	25	67
2	Virat	25	89
3	Rohit	24	55
4	Shikhar	31	47

…then a simple aggregation method is to calculate the summary of the Score, which is 87+67+89+55+47= 345. Or a different aggregation method would be to count the number of Name, which is 5.

Dataframe operations

Group by

A groupby operation involves some combination of splitting the object, applying a function, and combining the results. This can be used to group large amounts of data and compute operations on these groups.

E.g.

```
import pandas as pd
df = pd.DataFrame({'Animal': ['Tiger', 'Tiger','Parrot', 'Parrot'],
                   'Max Speed': [180., 170., 24., 26.]})
m=df.groupby(['Animal']).mean()
print(m)
```

OUTPUT

```
 Max Speed
Animal
Parrot      25.0
Tiger      175.0
```

Dataframe operations

Sorting

Sorting means arranging the contents in ascending or descending order.There are two kinds of sorting available in pandas(Dataframe).

1. By value(column)
2. By index

1. By value - Sorting over dataframe column/s elements is supported by sort_values() method. We will cover here three aspects of sorting values of dataframe.

- Sort a pandas dataframe in python by Ascending and Descending
- Sort a python pandas dataframe by single column
- Sort a pandas dataframe by multiple columns.

Dataframe operations

Sorting

Sort the python pandas Dataframe by single column – Ascending order

```
import pandas as pd
import numpy as np
#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
  'Age':pd.Series([26,27,25,24,31]),
  'Score':pd.Series([87,89,67,55,47])}
#Create a DataFrame
df = pd.DataFrame(d)
print("Dataframe contents without sorting")
print (df)
df=df.sort_values(by='Score')
print("Dataframe contents after sorting")
print (df)
```

#In above example dictionary object is used to create the dataframe.Elements of dataframe object df is sorted by sort_value() method.As argument we are passing value score for by parameter only.by default it is sorting in ascending manner.

```
OUTPUT
Dataframe contents without sorting
     Name  Age  Score
0  Sachin   26     87
1   Dhoni   27     89
2   Virat   25     67
3   Rohit   24     55
4     Shikhar   31     47

Dataframe contents after sorting
     Name  Age  Score
4  Shikhar   31     47
3   Rohit   24     55
2   Virat   25     67
1   Dhoni   27     87
0  Sachin   26     89
```

Dataframe operations

Sorting

Sort the python pandas Dataframe by single column – Descending order

```
import pandas as pd
import numpy as np
#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
  'Age':pd.Series([26,27,25,24,31]),
  'Score':pd.Series([87,89,67,55,47])}
#Create a DataFrame
df = pd.DataFrame(d)
print("Dataframe contents without sorting")
print (df)
df=df.sort_values(by='Score',ascending=0)
print("Dataframe contents after sorting")
print (df)
```

#In above example dictionary object is used to create the dataframe.Elements of dataframe object df is sorted by sort_value() method.we are passing 0 for Ascending parameter ,which sort the data in descending order of score.

```
OUTPUT
Dataframe contents without sorting
     Name  Age  Score
0   Sachin   26    89
1    Dhoni   27    87
2    Virat   25    67
3    Rohit   24    55
4     Shikhar   31    47

Dataframe contents after sorting
     Name  Age  Score
1    Dhoni   27    89
0   Sachin   26    87
2    Virat   25    67
3    Rohit   24    55
4  Shikhar   31    47
```

Dataframe operations

Sorting

Sort the pandas Dataframe by Multiple Columns

```
import pandas as pd
import numpy as np
#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
  'Age':pd.Series([26,25,25,24,31]),   'Score':pd.Series([87,67,89,55,47])}
#Create a DataFrame
df = pd.DataFrame(d)
print("Dataframe contents without sorting")
print (df)
df=df.sort_values(by=['Age', 'Score'],ascending=[True,False])
print("Dataframe contents after sorting")
print (df)
#In above example dictionary object is used to create
the dataframe.Elements of dataframe object df is s
orted by sort_value() method.we are passing two columns
as by parameter value and in ascending parameter also
with two parameters first true and second false,which
means sort in ascending order of age and descending
order of score
```

```
OUTPUT
Dataframe contents without sorting
     Name  Age  Score
0   Sachin   26    87
1    Dhoni   25    67
2    Virat   25    89
3    Rohit   24    55
4      Shikhar   31    47

Dataframe contents after sorting
     Name  Age  Score
3    Rohit   24    55
2    Virat   25    89
1    Dhoni   25    67
0   Sachin   26    87
4  Shikhar   31    47
```

Dataframe operations

Sorting

2. By index - Sorting over dataframe index sort_index() is supported by sort_values() method. We will cover here three aspects of sorting values of dataframe. We will cover here two aspects of sorting index of dataframe.

- how to sort a pandas dataframe in python by index in Ascending order
- how to sort a pandas dataframe in python by index in Descending order

Dataframe operations

Sorting

sort the dataframe in python pandas by index in ascending order:

```
import pandas as pd
import numpy as np
#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
   'Age':pd.Series([26,25,25,24,31]),
   'Score':pd.Series([87,67,89,55,47])}
#Create a DataFrame
df = pd.DataFrame(d)
df=df.reindex([1,4,3,2,0])
print("Dataframe contents without sorting")
print (df)
df1=df.sort_index()
print("Dataframe contents after sorting")
print (df1)
```

#In above example dictionary object is used to create the dataframe.Elements of dataframe object df is first reindexed by reindex() method,index 1 is positioned at 0,4 at 1 and so on.then sorting by sort_index() method. By default it is sorting in ascending order of index.

```
OUTPUT
Dataframe contents without sorting
      Name  Age  Score
1    Dhoni   25     67
4  Shikhar   31     47
3    Rohit   24     55
2    Virat   25     89
0   Sachin   26     87
Dataframe contents after sorting
      Name  Age  Score
0   Sachin   26     87
1    Dhoni   25     67
2    Virat   25     89
3    Rohit   24     55
4  Shikhar   31     47
```

index

Dataframe operations

Sorting

Sorting pandas dataframe by index in descending order:

```
import pandas as pd
import numpy as np
#Create a Dictionary of series
d = {'Name':pd.Series(['Sachin','Dhoni','Virat','Rohit','Shikhar']),
  'Age':pd.Series([26,25,25,24,31]),
  'Score':pd.Series([87,67,89,55,47])}
#Create a DataFrame
df = pd.DataFrame(d)
df=df.reindex([1,4,3,2,0])
print("Dataframe contents without sorting")
print (df)
df1=df.sort_index(ascending=0)
print("Dataframe contents after sorting")
print (df1)
#In above example dictionary object is used to create
the dataframe.Elements of dataframe object df is first
reindexed by reindex() method,index 1 is positioned at
0,4 at 1 and so on.then sorting by sort_index() method.
Passing ascending=0 as argument for descending order.
```

OUTPUT

```
Dataframe contents without sorting
      Name  Age  Score
1    Dhoni   25     67
4  Shikhar   31     47
3    Rohit   24     55
2    Virat   25     89
0   Sachin   26     87
Dataframe contents after sorting
      Name  Age  Score
4  Shikhar   31     47
3    Rohit   24     55
2    Virat   25     89
1    Dhoni   25     67
0   Sachin   26     87
```

index

Dataframe operations

Indexing

Index is like an address, that's how any data point across the dataframe or series can be accessed. Rows and columns both have indexes, rows indices are called as index and for columns its general column names.

Indexing in pandas used for selecting particular rows and columns of data from a DataFrame. Indexing could mean selecting all the rows and some of the columns, some of the rows and all of the columns, or some of each of the rows and columns. Indexing can also be known as Subset Selection.

Dataframe operations

Indexing e.g.

```
import pandas as pd
students = [ ('Mohak', 34, 'Sydeny') ,('Freya', 30, 'Delhi' ) ,('Rajesh', 16, 'New York') ]
# Create a DataFrame object
dfObj = pd.DataFrame(students, columns = ['Name' , 'Age', 'City'],
index=['a', 'b', 'c'])
#Selecting a Single Row by Index label
rowData = dfObj.loc[ 'b' , : ]
print("Select a Single Row " , rowData , sep='\n')
print("Type : " , type(rowData))
#Selecting multiple Rows by Index labels
rowData = dfObj.loc[ ['c' , 'b'] , : ]
print("Select multiple Rows" , rowData , sep='\n')
#Select both Rows & Columns by Index labels
subset = dfObj.loc[ ['c' , 'b'] ,['Age', 'Name'] ]
print("Select both columns & Rows" , subset , sep='\n')
#Select a single column by Index Position
print(" Select column at index 2 ")
print( dfObj.iloc[ : , 2 ] )
#Select multiple columns by Index range
print(" Select columns in column index range 0 to 2")
print(dfObj.iloc[:, 0:2])
```

OUTPUT

```
Select a Single Row
Name    Freya
Age       30
City    Delhi
Name: b, dtype: object
Type :  <class 'pandas.core.series.Series'>
Select multiple Rows
   Name  Age      City
c  Rajesh   16  New York
b   Freya   30     Delhi
Select both columns & Rows
  Age    Name
c   16  Rajesh
b   30   Freya
 Select column at index 2
a      Sydeny
b       Delhi
c    New York
Name: City, dtype: object
 Select columns in column index range 0 to
2
   Name  Age
a   Mohak   34
b   Freya   30
c  Rajesh   16
```

Dataframe operations

Renaming Indexing e.g.

Index can be renamed using rename method.
e.g.

```
import pandas as pd

df = pd.DataFrame({'A': [11, 21, 31],
                   'B': [12, 22, 32],
                   'C': [13, 23, 33]},
                  index=['ONE', 'TWO', 'THREE'])
df_new = df.rename(columns={'A': 'a'}, index={'ONE': 'one'})
print(df_new)
```

OUTPUT

```
        a   B   C
one    11  12  13
TWO    21  22  23
THREE  31  32  33
```

Pivoting - dataframe

DataFrame -It is a 2-dimensional data structure with columns of different types. It is just similar to a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object.

Pivot –Pivot reshapes data and uses unique values from index/ columns to form axes of the resulting dataframe. Index is column name to use to make new frame’s index.Columns is column name to use to make new frame’s columns.Values is column name to use for populating new frame’s values.

Pivot table - Pivot table is used to summarize and aggregate data inside dataframe.

Pivoting - dataframe

Example of pivot:

	ITEM	COMPANY	RUPEES	USD
	TV	LG	12000	700
	TV	VIDEOCON	10000	650
	AC	LG	15000	800
	AC	SONY	14000	750

DATAFRAME

COMPANY	LG	SONY	VIDEOCON
ITEM			
AC	15000	14000	NaN
TV	12000	NaN	10000

PIVOT

Pivoting - dataframe

There are two functions available in python for pivoting dataframe.

1.Pivot()
2.pivot_table()

1. pivot() - This function is used to create a new derived table(pivot) from existing dataframe. It takes 3 arguments : index, columns, and values. As a value for each of these parameters we need to specify a column name in the original table(dataframe). Then the pivot function will create a new table(pivot), whose row and column indices are the unique values of the respective parameters. The cell values of the new table are taken from column given as the values parameter.

Pivoting - dataframe

```
#pivot() e.g. program
from collections import OrderedDict
from pandas import DataFrame
import pandas as pd
import numpy as np
table = OrderedDict((
   ("ITEM", ['TV', 'TV', 'AC', 'AC']),
   ('COMPANY',['LG', 'VIDEOCON', 'LG', 'SONY']),
   ('RUPEES',  ['12000', '10000', '15000', '14000']),
   ('USD',   ['700', '650', '800', '750'])
))
d = DataFrame(table)
print("DATA OF DATAFRAME")
print(d)
p = d.pivot(index='ITEM', columns='COMPANY', values='RUPEES')
print("\n\nDATA OF PIVOT")
print(p)
print (p[p.index=='TV'].LG.values)
```

```
   ITEM   COMPANY    RUPEES   USD
   TV     LG          12000   700
   TV     VIDEOCON    10000   650
   AC     LG          15000   800
   AC     SONY        14000   750
```

```
COMPANY     LG      SONY     VIDEOCON
 ITEM
  AC       15000   14000       NaN
  TV       12000    NaN       10000
```

#pivot() creates a new table/DataFrame whose columns are the unique values in COMPANY and whose rows are indexed with the unique values of ITEM.Last statement of above program retrun value of TV item LG company i.e. 12000

Pivoting - dataframe

#Pivoting By Multiple Columns

Now in previous example, we want to pivot the values of both RUPEES an USD together, we will have to use pivot function in below manner.

```
p = d.pivot(index='ITEM', columns='COMPANY')
```

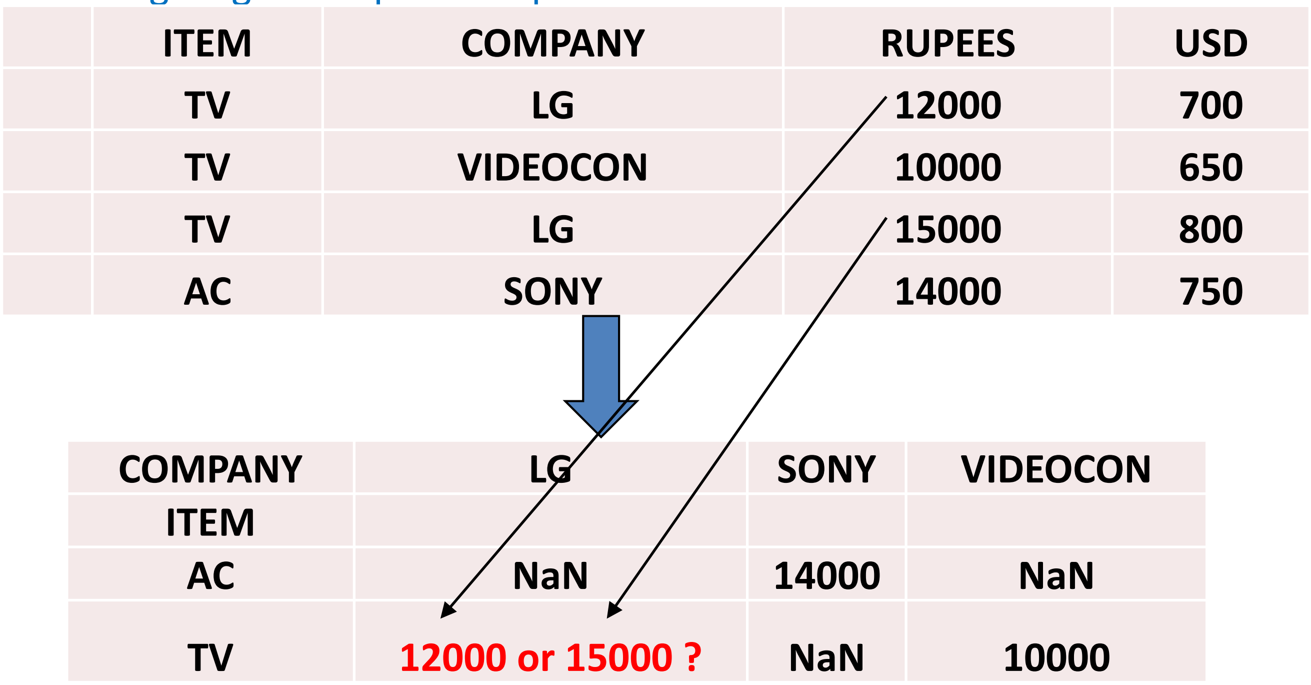
This will return the following pivot.

	RUPEES			USD		
COMPANY	LG	SONY	VIDEOCON	LG	SONY	VIDEOCON
ITEM						
AC	15000	14000	NaN	800	750	NaN
TV	12000	NaN	10000	700	NaN	650

Pivoting - dataframe

#Common Mistake in Pivoting

pivot method takes at least 2 column names as parameters - the index and the columns named parameters. Now the problem is that,What happens if we have multiple rows with the same values for these columns? What will be the value of the corresponding cell in the pivoted table using pivot method? The following diagram depicts the problem:



	ITEM	COMPANY	RUPEES	USD
	TV	LG	12000	700
	TV	VIDEOCON	10000	650
	TV	LG	15000	800
	AC	SONY	14000	750

COMPANY	LG	SONY	VIDEOCON
ITEM			
AC	NaN	14000	NaN
TV	12000 or 15000 ?	NaN	10000

d.pivot(index='ITEM', columns='COMPANY', values='RUPEES')

It throws an exception with the following message:

ValueError: Index contains duplicate entries, cannot reshape

Pivoting - dataframe

#Pivot Table

The pivot_table() method comes to solve this problem. It works like pivot, but it aggregates the values from rows with duplicate entries for the specified columns.

	ITEM	COMPANY	RUPEES	USD
	TV	LG	12000	700
	TV	VIDEOCON	10000	650
	TV	LG	15000	800
	AC	SONY	14000	750

COMPANY	LG	SONY	VIDEOCON
ITEM			
AC	NaN	14000	NaN
TV	13500 = mean(12000,15000)	NaN	10000

```
d.pivot_table(index='ITEM', columns='COMPANY', values='RUPEES‘,aggfunc=np.mean)
```

In essence *pivot_table* is a generalisation of *pivot*, which allows you to aggregate multiple values with the same destination in the pivoted table.

Handling Missing Data

Filling the missing data Eg.

```
import pandas as pd
import numpy as np
raw_data = {'name': ['freya', 'mohak', 'rajesh'],
        'age': [42, np.nan, 36 ] }
df = pd.DataFrame(raw_data, columns = ['name',
'age'])
print(df)
df['age']=df['age'].fillna(0)
print(df)
```

In above e.g. age of mohak is fille

output

```
     name   age
0  freya  42.0
1  mohak   NaN
2  rajesh  36.0
    name   age
1   freya  42.0
2   mohak   0.0
3  rajesh  36.0
```

Note :- The dropna() function is used to remove missing values. df.dropna() will remove the record of mohak

Importing data from a MySQL database into a Pandas data frame

```
import mysql.connector as sql
import pandas as pd
db_connection = sql.connect(host='localhost', database='bank', user='root',
password='root')
db_cursor = db_connection.cursor()
db_cursor.execute('SELECT * FROM bmaster')
table_rows = db_cursor.fetchall()
df = pd.DataFrame(table_rows)
print(df)
```

OUTPUT
Will be as data available in table bmaster

Note :- for mysql.connector library use **pip install mysql_connector** command in command prompt.
Pass proper host name,database name,user name and password in connect method.

Exporting data to a MySQL database from a Pandas data frame

```
import pandas as pd
from sqlalchemy import create_engine
engine = create_engine('mysql+mysqlconnector://root:root@localhost/bank')
lst = ['vishal', 'ram']
lst2 = [11, 22]
# Calling DataFrame constructor after zipping
# both lists, with columns specified
df = pd.DataFrame(list(zip(lst, lst2)),
             columns =['Name', 'val'])
df.to_sql(name='bmaster', con=engine, if_exists = 'replace', index=False)
```

user name password server databasename

Note :- Create dataframe as per the structure of the table.to_sql() method is used to write data from dataframe to mysql table. Standard library sqlalchemy is being used for writing data.