WEEK 3

Explore Pandas module

What is pandas?

- Pandas is an open-source library that is built on top of NumPy library
- It provides various data structures and operations for manipulating numerical data and time series.
- It provides ready to use high-performance data structures and data analysis tools. Pandas module runs on top of NumPy and it is popularly used for data science and data analytics.
- Flexible reshaping and pivoting of data sets
- A DataFrame is a data structure that organizes data into a 2-dimensional table of rows and columns, much like a spreadsheet
- DataFrame is common across many different languages like in R, Scala, and other languages.

Aggregation and Grouping

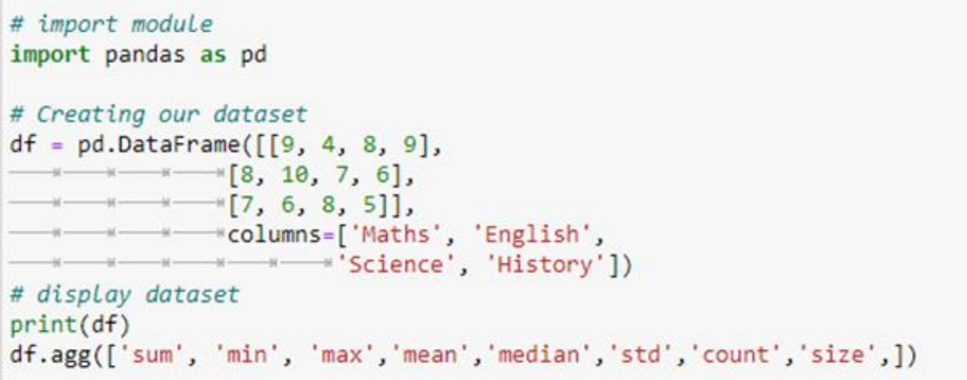
Data aggregation and grouping allows us to create summaries for display or analysis, for example, when calculating average values or creating a table of counts or sums.

Aggregation function

sum()	:Compute sum of column values
min()	:Compute min of column values
max()	:Compute max of column values
mean()	:Compute mean of column
size()	:Compute column sizes
describe()	:Generates descriptive statistics
first()	:Compute first of group values
last()	:Compute last of group values
count()	:Compute count of column values
std()	:Standard deviation of column

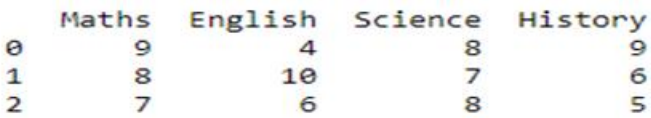
#simple example of using aggregation functions on a Dataframe

Example:

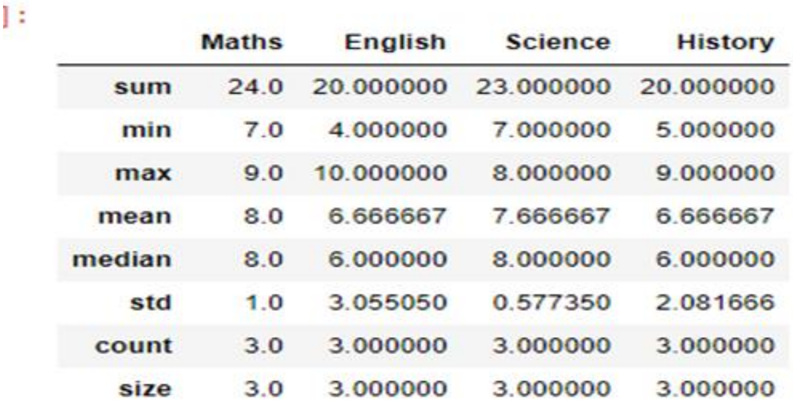


```
# import module
import pandas as pd

# Creating our dataset
df = pd.DataFrame([[9, 4, 8, 9],
                   [8, 10, 7, 6],
                   [7, 6, 8, 5]],
                  columns=['Maths', 'English',
                           'Science', 'History'])
# display dataset
print(df)
df.agg(['sum', 'min', 'max','mean','median','std','count','size',])
```

Output:



```
   Maths  English  Science  History
0      9        4        8        9
1      8       10        7        6
2      7        6        8        5
```



]:

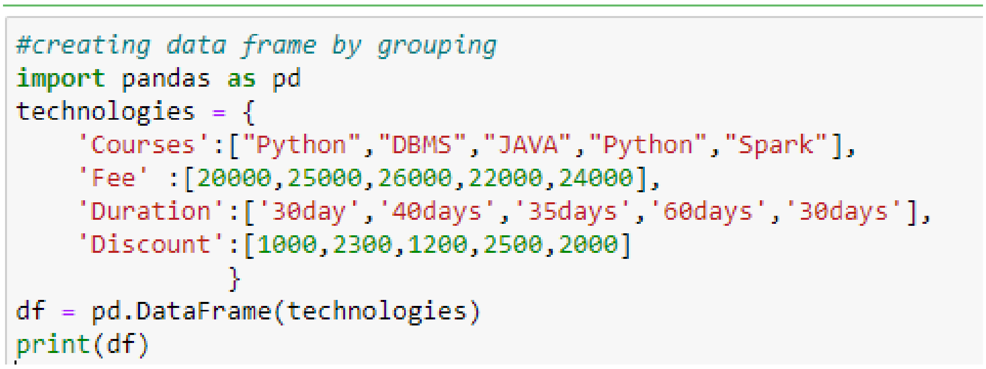
	Maths	English	Science	History
sum	24.0	20.000000	23.000000	20.000000
min	7.0	4.000000	7.000000	5.000000
max	9.0	10.000000	8.000000	9.000000
mean	8.0	6.666667	7.666667	6.666667
median	8.0	6.000000	8.000000	6.000000
std	1.0	3.055050	0.577350	2.081666
count	3.0	3.000000	3.000000	3.000000
size	3.0	3.000000	3.000000	3.000000

Grouping using Pandas

Grouping is used to group data using some criteria from our dataset. It is used as split – apply-combine strategy.

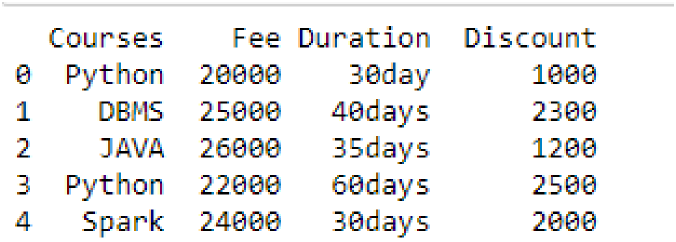
Function Description:

- sum() :Compute sum of column values
- min() :Compute min of column values
- max() :Compute max of column values
- count() :Compute count of column values



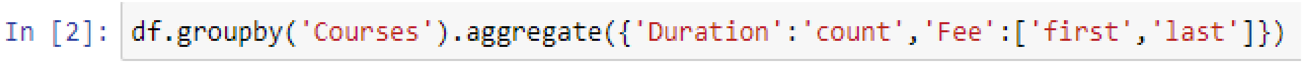
```
#creating data frame by grouping
import pandas as pd
technologies = {
    'Courses':["Python","DBMS","JAVA","Python","Spark"],
    'Fee' :[20000,25000,26000,22000,24000],
    'Duration':['30day','40days','35days','60days','30days'],
    'Discount':[1000,2300,1200,2500,2000]
              }
df = pd.DataFrame(technologies)
print(df)
```

Output:



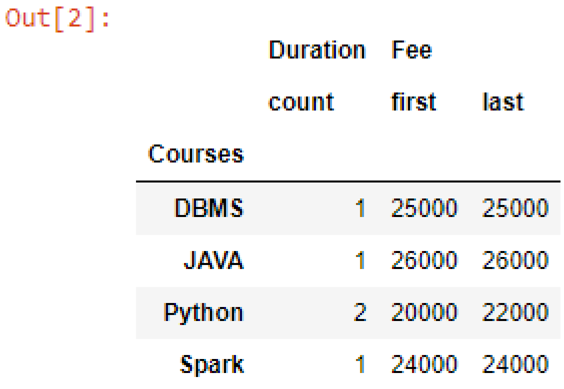
```
  Courses    Fee Duration  Discount
0  Python  20000    30day      1000
1    DBMS  25000   40days      2300
2    JAVA  26000   35days      1200
3  Python  22000   60days      2500
4   Spark  24000   30days      2000
```

Ex1: using groupby functions in dataframe to find first and last element in a group



```
In [2]: df.groupby('Courses').aggregate({'Duration':'count','Fee':['first','last']})
```

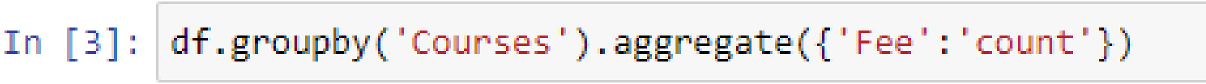
Output:



Out[2]:

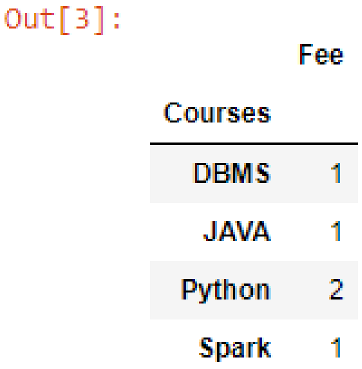
	Duration	Fee	
	count	first	last
Courses			
DBMS	1	25000	25000
JAVA	1	26000	26000
Python	2	20000	22000
Spark	1	24000	24000

Ex2: using groupby functions in dataframe to get count



```
In [3]: df.groupby('Courses').aggregate({'Fee':'count'})
```

Output:



Out[3]:

	Fee
Courses	
DBMS	1
JAVA	1
Python	2
Spark	1

Pivot and Melt function

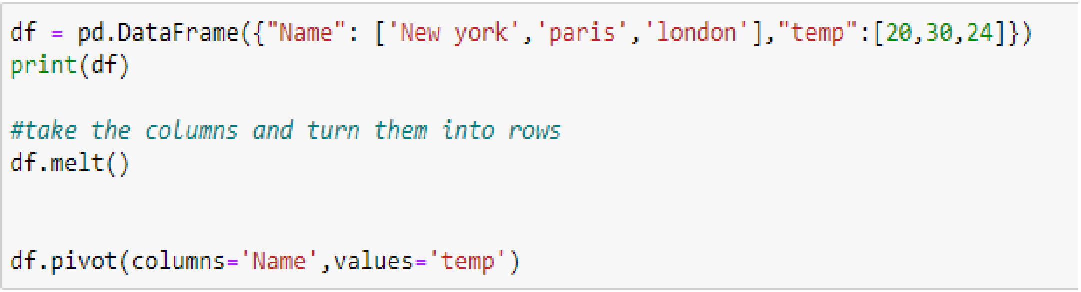
Pivot

Pivot () is used for pivoting the dataframe without applying aggregation. It doesn't contain same values or columns/index. While

Melt

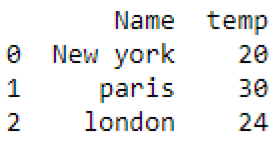
The melt () function enables us to reshape and elongate the data frames in a user-defined manner. It organizes the data values in a long data frame format.

#creating data frame by Pivot and melt functions

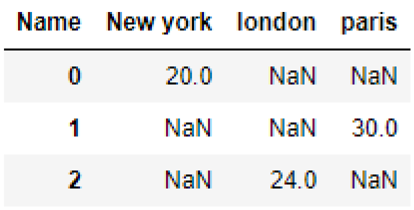


```
df = pd.DataFrame({"Name": ['New york','paris','london'],"temp":[20,30,24]})
print(df)

#take the columns and turn them into rows
df.melt()


df.pivot(columns='Name',values='temp')
```

Output:



```
       Name  temp
0  New york    20
1     paris    30
2    london    24
```



Name	New york	london	paris
0	20.0	NaN	NaN
1	NaN	NaN	30.0
2	NaN	24.0	NaN

Map, Filter and Reduce, Lambda functions

Map

map() function returns a map object (which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.)

Filter

The filter() function returns an iterator were the items are filtered through a function to test if the item is accepted or not

Reduce

The reduce (fun, seq) function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence passed along. This function is defined in “functools” module.

Lambda Function

A lambda function is a small anonymous function. A lambda function can take any number of arguments, but can only have one expression.

Syntax

lambda arguments : expression

Add 1 to argument a, and return the result:

x = lambda a : a + 1
print(x(5))

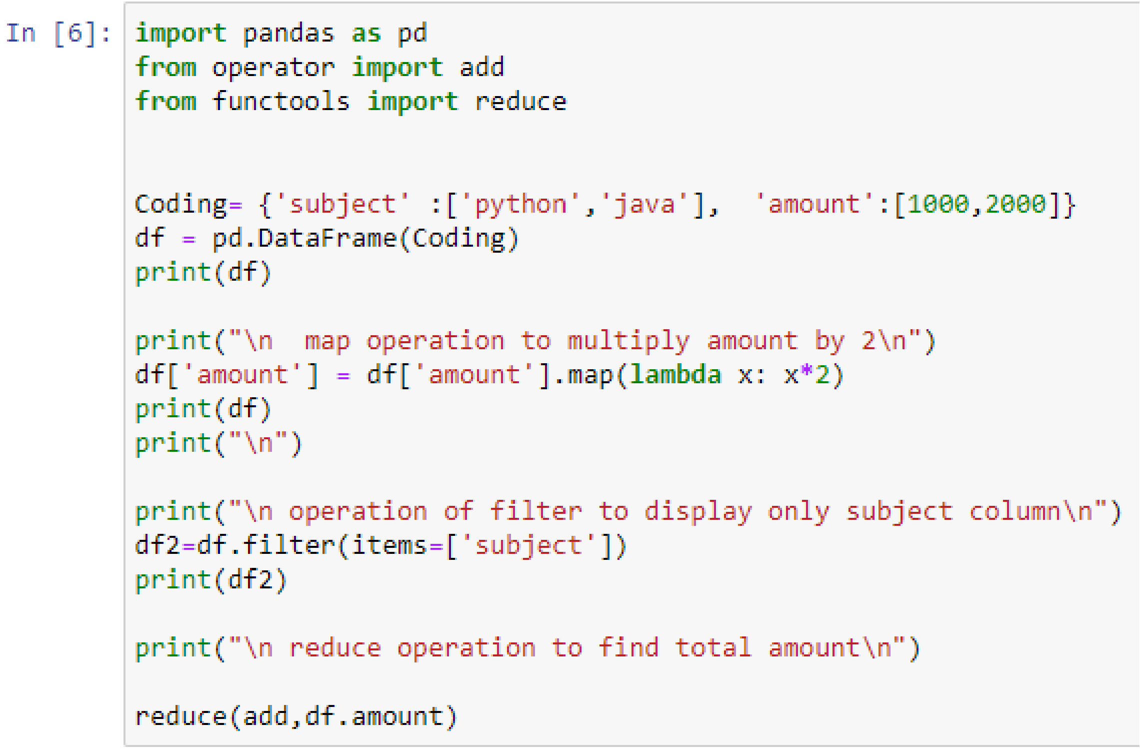
Output 6

Summarize argument a, b, and c and return the result:

x = lambda a, b, c : a + b + c
print(x(5, 6, 2))

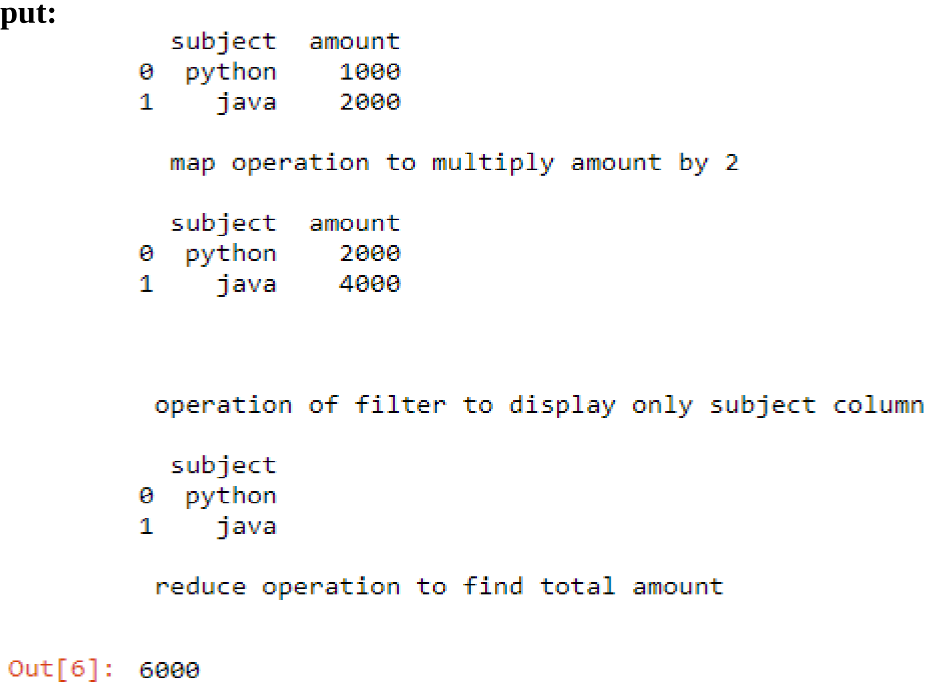
Output 13

#creating data frame by Map, Filter and Reduce functions



```
In [6]: import pandas as pd
        from operator import add
        from functools import reduce


        Coding= {'subject' :['python','java'],  'amount':[1000,2000]}
        df = pd.DataFrame(Coding)
        print(df)

        print("\n  map operation to multiply amount by 2\n")
        df['amount'] = df['amount'].map(lambda x: x*2)
        print(df)
        print("\n")

        print("\n operation of filter to display only subject column\n")
        df2=df.filter(items=['subject'])
        print(df2)

        print("\n reduce operation to find total amount\n")

        reduce(add,df.amount)
```

Output:



```
  subject  amount
0  python    1000
1    java    2000

  map operation to multiply amount by 2

  subject  amount
0  python    2000
1    java    4000



 operation of filter to display only subject column

  subject
0  python
1    java

 reduce operation to find total amount

Out[6]: 6000
```

Time series using Pandas

Time series data consists of data points attached to sequential time stamps. Daily sales, hourly temperature values, and second-level measurements in a chemical process are some examples of time series data.

Syntax of the Pandas date_range

pandas.date_range(start=None, end=None, periods=None, freq=None, tz=None)

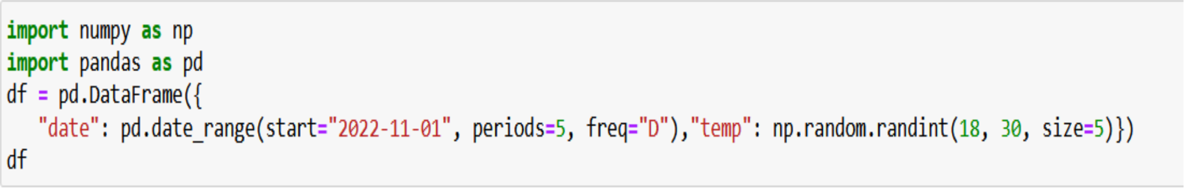
There are many parameters in the methods. But I will explain only the most used parameters.

start: *Starting date. It is the left bound for generating dates.*

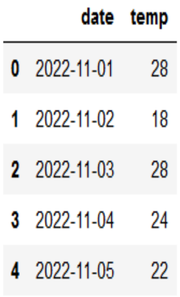
end: *Ending date. It is the upper bound for generating dates.*

periods: *Number of periods to generate.*

freq: *It is used to generate dates on the basis of frequency like "D", "M" e.t.c*



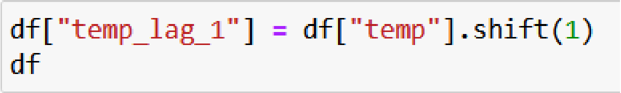
```
import numpy as np
import pandas as pd
df = pd.DataFrame({
   "date": pd.date_range(start="2022-11-01", periods=5, freq="D"),"temp": np.random.randint(18, 30, size=5)})
df
```



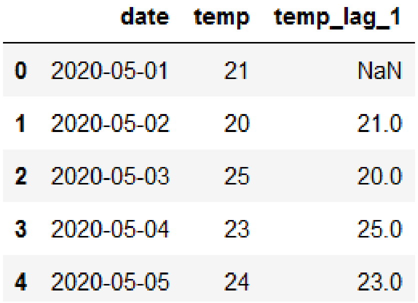
	date	temp
0	2022-11-01	28
1	2022-11-02	18
2	2022-11-03	28
3	2022-11-04	24
4	2022-11-05	22

Shift operation

It is a common operation to shift time series data. We may need to make a comparison between lagged or lead features. In our data frame, we can create a new feature that contains the temperature of the previous day.



```
df["temp_lag_1"] = df["temp"].shift(1)
df
```



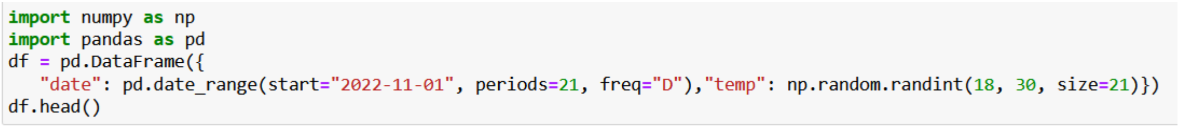
	date	temp	temp_lag_1
0	2020-05-01	21	NaN
1	2020-05-02	20	21.0
2	2020-05-03	25	20.0
3	2020-05-04	23	25.0
4	2020-05-05	24	23.0

Resample

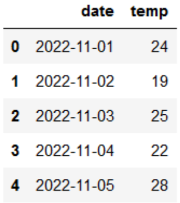
Another common operation performed on time series data is resampling. It involves in changing the frequency of the periods. For instance, we may be interested in the weekly temperature data rather than daily measurements.

The resample function creates groups (or bins) of a specified internal. Then, we can apply aggregation functions to the groups to calculate the value based on resampled frequency.

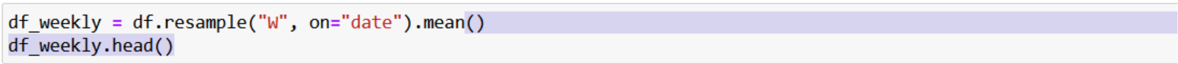
Let's calculate the average weekly temperatures. The first step is to resample the data to week level. Then, we will apply the mean function to calculate the average.



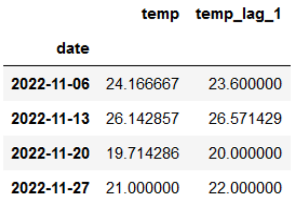
```
import numpy as np
import pandas as pd
df = pd.DataFrame({
   "date": pd.date_range(start="2022-11-01", periods=21, freq="D"),"temp": np.random.randint(18, 30, size=21)})
df.head()
```



	date	temp
0	2022-11-01	24
1	2022-11-02	19
2	2022-11-03	25
3	2022-11-04	22
4	2022-11-05	28



```
df_weekly = df.resample("W", on="date").mean()
df_weekly.head()
```



	temp	temp_lag_1
date		
2022-11-06	24.166667	23.600000
2022-11-13	26.142857	26.571429
2022-11-20	19.714286	20.000000
2022-11-27	21.000000	22.000000