

```
In [1]: empty_dictionary = {}  
print(type(empty_dictionary))  
  
<class 'dict'>
```

```
In [2]: bio_data = {'Name': 'Bob Marley', 'Age':35, 'Height':"5.6 ft", 'Hobby': 'Music'}  
print(bio_data)  
  
{'Name': 'Bob Marley', 'Age': 35, 'Height': '5.6 ft', 'Hobby': 'Music'}
```

```
In [4]: credentials = { 'UserA' : 'wkliopnc' , 'UserB': 98760 , 'UserC' :98760 }
```

```
In [5]: hobby = bio_data['Hobby']  
print(hobby)  
  
Music
```

```
In [6]: age = bio_data['Age']  
print(age)  
  
35
```

```
In [7]: age = bio_data.get('Age')  
print(age)  
  
35
```

```
In [8]: profession = bio_data.get('Profession','NA')  
print(profession)  
  
NA
```

```
In [9]: bio_data['Age'] = 36  
print(bio_data)  
  
{'Name': 'Bob Marley', 'Age': 36, 'Height': '5.6 ft', 'Hobby': 'Music'}
```

```
In [10]: #add a key, val  
bio_data['Profession'] = 'Singer'  
print(bio_data)  
  
{'Name': 'Bob Marley', 'Age': 36, 'Height': '5.6 ft', 'Hobby': 'Music', 'Profes  
sion': 'Singer'}
```

```
In [11]: print('Profession' in bio_data)  
  
True
```

```
In [12]: #get list of keys
print(list(bio_data.keys()))

#get list of values
print(list(bio_data.values()))
```

```
['Name', 'Age', 'Height', 'Hobby', 'Profession']
['Bob Marley', 36, '5.6 ft', 'Music', 'Singer']
```

```
In [13]: new_dictionary = dict(Country='Jamaica', Songs=['One Love', 'Misty Morning'])
```

```
In [14]: bio_data.update(new_dictionary)
print(bio_data)
```

```
{'Name': 'Bob Marley', 'Age': 36, 'Height': '5.6 ft', 'Hobby': 'Music', 'Profession': 'Singer', 'Country': 'Jamaica', 'Songs': ['One Love', 'Misty Morning']}
```

```
In [15]: del bio_data['Songs']
print(bio_data)
```

```
{'Name': 'Bob Marley', 'Age': 36, 'Height': '5.6 ft', 'Hobby': 'Music', 'Profession': 'Singer', 'Country': 'Jamaica'}
```

```
In [16]: students_data = { 1:['Shivam Bansal', 24] , 2:['Udit Bansal',25], 3:['Sonam Gupta']
print(students_data)
```

```
{1: ['Shivam Bansal', 24], 2: ['Udit Bansal', 25], 3: ['Sonam Gupta', 26], 4: ['Saif Ansari', 24], 5: ['Huzefa Calcuttawala', 27]}
```

```
In [17]: print(len(students_data))
```

5

```
In [18]: #see all the details of students.
print(list(students_data.values()))
```

```
[['Shivam Bansal', 24], ['Udit Bansal', 25], ['Sonam Gupta', 26], ['Saif Ansari', 24], ['Huzefa Calcuttawala', 27]]
```

```
In [19]: students_data[6] = ['Manasi Sharma', 22]
print(students_data)
```

```
{1: ['Shivam Bansal', 24], 2: ['Udit Bansal', 25], 3: ['Sonam Gupta', 26], 4: ['Saif Ansari', 24], 5: ['Huzefa Calcuttawala', 27], 6: ['Manasi Sharma', 22]}
```

```
In [20]: del students_data[2]
print(students_data)
```

```
{1: ['Shivam Bansal', 24], 3: ['Sonam Gupta', 26], 4: ['Saif Ansari', 24], 5:
['Huzefa Calcuttawala', 27], 6: ['Manasi Sharma', 22]}
```

```
In [21]: print(list(students_data.keys()))
```

```
[1, 3, 4, 5, 6]
```

```
In [22]: import math
math.sqrt(81)
```

```
Out[22]: 9.0
```

```
In [23]: math.factorial(5)
```

```
Out[23]: 120
```

```
In [24]: n=5
k=3
math.factorial(n)/(math.factorial(k)*math.factorial(n-k))
math.factorial(n)/(math.factorial(k)*math.factorial(n-k))
```

```
Out[24]: 10
```

```
In [25]: from math import factorial as fac
fac(15)
```

```
Out[25]: 1307674368000
```

```
In [26]: h = 50
if h > 50:
    print ('No')
else:
    print('Yes')
```

```
Yes
```

```
In [1]: c = 5
while c != 0:
    print(c)
    c -= 1
```

```
5
4
3
2
1
```

```
In [ ]: while True:
        response = input()
        if int(response) % 7 == 0:
            break
```

```
In [ ]: def hello(who):
        print('hello {}'.format(who))
```

```
In [3]: print("first" "second")
```

firstsecond

```
In [4]: s = 'PARROT'
        s[4]
```

Out[4]: '0'

```
In [5]: help(str)
```

Help on class str in module builtins:

```
class str(object)
|   str(object='') -> str
|   str(bytes_or_buffer[, encoding[, errors]]) -> str
|
|   Create a new string object from the given object. If encoding or
|   errors is specified, then the object must expose a data buffer
|   that will be decoded using the given encoding and error handler.
|   Otherwise, returns the result of object.__str__() (if defined)
|   or repr(object).
|   encoding defaults to sys.getdefaultencoding().
|   errors defaults to 'strict'.
|
|   Methods defined here:
|
|   __add__(self, value, /)
|       Return self+value.
```

```
In [7]: s = 'parrot'
        s.capitalize()
```

Out[7]: 'Parrot'

```
In [8]: list("Characters")
```

Out[8]: ['C', 'h', 'a', 'r', 'a', 'c', 't', 'e', 'r', 's']

```
In [9]: dict("characters")
```

```
-----  
ValueError                                Traceback (most recent call last)  
<ipython-input-9-03b814add21b> in <module>()  
----> 1 dict("characters")  
  
ValueError: dictionary update sequence element #0 has length 1; 2 is required
```

```
In [3]: from urllib.request import urlopen
with urlopen('http://sixty-north.com/c/t.txt') as story:
    story_words = []
    for line in story:
        line_words = line.split()
        for word in line_words:
            story_words.append(word)

print(story_words)
```

```
-----
TimeoutError                                Traceback (most recent call last)
~\AppData\Local\anaconda3\lib\urllib\request.py in do_open(self, http_class, req, **http_conn_args)
    1317         h.request(req.get_method(), req.selector, req.data, headers,
-> 1318                        encode_chunked=req.has_header('Transfer-encoding'))
    1319         except OSError as err: # timeout error

~\AppData\Local\anaconda3\lib\http\client.py in request(self, method, url, body, headers, encode_chunked)
    1238         """Send a complete request to the server."""
-> 1239         self._send_request(method, url, body, headers, encode_chunked)
    1240

~\AppData\Local\anaconda3\lib\http\client.py in _send_request(self, method, url, body, headers, encode_chunked)
    1284         body = _encode(body, 'body')
-> 1285         self.endheaders(body, encode_chunked=encode_chunked)
    1286

~\AppData\Local\anaconda3\lib\http\client.py in endheaders(self, message_body, encode_chunked)
    1233         raise CannotSendHeader()
-> 1234         self._send_output(message_body, encode_chunked=encode_chunked)
    1235

~\AppData\Local\anaconda3\lib\http\client.py in _send_output(self, message_body, encode_chunked)
    1025         del self._buffer[:]
-> 1026         self.send(msg)
    1027

~\AppData\Local\anaconda3\lib\http\client.py in send(self, data)
    963         if self.auto_open:
--> 964             self.connect()
    965         else:

~\AppData\Local\anaconda3\lib\http\client.py in connect(self)
    935         self.sock = self._create_connection(
--> 936             (self.host,self.port), self.timeout, self.source_address)
    937         self.sock.setsockopt(socket.IPPROTO_TCP, socket.TCP_NODELAY, 1)

~\AppData\Local\anaconda3\lib\socket.py in create_connection(address, timeout, source_address)
```

```

723     if err is not None:
--> 724         raise err
725     else:

~\AppData\Local\anaconda3\lib\socket.py in create_connection(address, timeout,
source_address)
712         sock.bind(source_address)
--> 713         sock.connect(sa)
714         # Break explicitly a reference cycle

```

**TimeoutError:** [WinError 10060] A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond

During handling of the above exception, another exception occurred:

```

URLError                                Traceback (most recent call last)
<ipython-input-3-b58b5af38bcc> in <module>()
      1 from urllib.request import urlopen
----> 2 with urlopen('http://sixty-north.com/c/t.txt') as story:
      3     story_words = []
      4     for line in story:
      5         line_words = line.split()

~\AppData\Local\anaconda3\lib\urllib\request.py in urlopen(url, data, timeout,
cafile, capath, cadefault, context)
221     else:
222         opener = _opener
--> 223     return opener.open(url, data, timeout)
224
225 def install_opener(opener):

~\AppData\Local\anaconda3\lib\urllib\request.py in open(self, fullurl, data, timeout)
524         req = meth(req)
525
--> 526     response = self._open(req, data)
527
528     # post-process response

~\AppData\Local\anaconda3\lib\urllib\request.py in _open(self, req, data)
542     protocol = req.type
543     result = self._call_chain(self.handle_open, protocol, protocol
+
--> 544                             '_open', req)
545     if result:
546         return result

~\AppData\Local\anaconda3\lib\urllib\request.py in _call_chain(self, chain, kind, meth_name, *args)
502     for handler in handlers:
503         func = getattr(handler, meth_name)
--> 504     result = func(*args)
505     if result is not None:
506         return result

~\AppData\Local\anaconda3\lib\urllib\request.py in http_open(self, req)

```

```

1344
1345     def http_open(self, req):
-> 1346         return self.do_open(http.client.HTTPConnection, req)
1347
1348     http_request = AbstractHTTPHandler.do_request_

~\AppData\Local\anaconda3\lib\urllib\request.py in do_open(self, http_class, req, **http_conn_args)
1318         encode_chunked=req.has_header('Transfer-encoding'))
1319     except OSError as err: # timeout error
-> 1320         raise URLError(err)
1321     r = h.getresponse()
1322     except:

```

**URLError:** <urlopen error [WinError 10060] A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed to respond>

In [2]: *# creating a 1d arraya using a single list*

```

import numpy as np
array_1d = np.array ([2, 4, 5, 6, 7, 9])
print(array_1d)
print(type(array_1d))

```

```

[2 4 5 6 7 9]
<class 'numpy.ndarray'>

```

In [3]: *# creating a 1d arraya using a single list*

```

import numpy as np
array_2d = np.array ([[2, 4, 5],[ 6, 7, 9]])
print(array_2d)
print(type(array_2d))

```

```

[[2 4 5]
 [6 7 9]]
<class 'numpy.ndarray'>

```

In [6]: *# Create a 3\*3 array using list\_1 = [1,2,3] list\_2 = [4,5,6] list\_3 = [7,8,9]*  
**import** ast,sys

```

import numpy as np
array_1 = np.array ([[1, 2, 3],[4, 5, 6],[7, 8, 9]])

print(array_1)

```

```

[[1 2 3]
 [4 5 6]
 [7 8 9]]

```



```
In [15]: # Create a 3*3 array using list_1 = [1,2,3] list_2 = [4,5,6] list_3 = [7,8,9]
import ast,sys
#input_str = sys.stdin.read()
#input_list = ast.literal_eval(input_str)
#list_1 = input_list[0]
#list_2 = input_list[1]
#list_3 = input_list[2]

list_1 = [1,2,3]
list_2 = [4,5,6]
list_3 = [7,8,9]

import numpy as np
array_1 = np.array([list_1,list_2,list_3])
array_2 = np.array(list_1)
array_3 = np.array(list_2)
array_4 = array_2*array_3
array_5 = array_2**2
array_6 = array_2+array_3
print(array_1)
print(array_2)
print(array_3)
print(array_4)
print(array_5)
print(array_6)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[1 2 3]
[4 5 6]
[ 4 10 18]
[1 4 9]
[5 7 9]
```

```
In [19]: #Perform an element-wise multiplication using list_1 = [2,3,4,5] list_2 = [7,8,9,6]
#Hint: Convert the list to an array and after multiplication convert it back to a list

list_1 = [2,3,4,5]
list_2 = [7,8,9,6]
import numpy as np
array_1 = np.array(list_1)#Type your answer here
array_2 = np.array(list_2)#Type your answer here
array_3 = array_1*array_2#Type your answer here

print(list(array_3))

print(array_3)
```

```
[14, 24, 36, 30]
[14 24 36 30]
```

```
In [38]: import numpy as np
# create 5X3 2 dimensional array of ones with default datatype float64.
array_1 = np.ones((5,3))
# create 5X3 2 dimensional array of ones with default datatype integer.
array_6 = np.ones((5,3),dtype = np.int)
array_2 = np.zeros(3)
array_3 = np.random.random(5)
array_4 = np.arange(10,100,5)
# array of 25 between 15 and 18
array_5 = np.linspace(15,18,25)
print(array_1)
print(array_2)
print(array_3)
print(array_4)
print(array_6)
print(array_5)
#Create an array of first 10 multiples of 5 using the 'arange' function.
array_7 = np.arange(5,55,5)
print(array_7)
```

```
[[1.  1.  1.]
 [1.  1.  1.]
 [1.  1.  1.]
 [1.  1.  1.]
 [1.  1.  1.]]
[0.  0.  0.]
[0.83275204 0.86647743 0.29780628 0.979333    0.70985028]
[10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95]
[[1 1 1]
 [1 1 1]
 [1 1 1]
 [1 1 1]
 [1 1 1]]
[15.    15.125 15.25  15.375 15.5   15.625 15.75  15.875 16.    16.125
 16.25  16.375 16.5   16.625 16.75  16.875 17.    17.125 17.25  17.375
 17.5   17.625 17.75  17.875 18.    ]
[ 5 10 15 20 25 30 35 40 45 50]
```

In [45]: *#Create an array using list list\_1 = [10,11,12,13] and list\_2 = [15,12,13,14]  
#and print the shape and dimension of the array created.*

```
list_1 = [10,11,12,13]
list_2 = [15,12,13,14]
import numpy as np
array_1 = np.array([list_1,list_2])
print(array_1)
#array_1.shape
print(array_1.ndim)
print(array_1.shape)
```

```
[[10 11 12 13]
 [15 12 13 14]]
2
(2, 4)
```

In [54]: *# Create a random large array and process it.*

```
import numpy as np
array_1= np.random.random((1000,300))
```

```
# print first row
print(array_1[1,])
print(array_1.shape)
print(array_1.dtype)
print(array_1.itemsize)
print(array_1.ndim)
```

```
[0.80649677 0.55162442 0.44651401 0.47701209 0.42859912 0.77975405
0.46591274 0.27689779 0.80947984 0.21662684 0.3655462 0.54769068
0.52908255 0.05099134 0.39619353 0.51834591 0.0426613 0.57611871
0.96735961 0.47821094 0.98598753 0.69511954 0.7586764 0.85732279
0.88428747 0.3557488 0.15012623 0.72632942 0.03036386 0.12867723
0.75116441 0.32593737 0.01207448 0.03678673 0.47678584 0.12768942
0.58214378 0.54127876 0.79091173 0.59456168 0.86049421 0.49093159
0.56046316 0.30818558 0.53320614 0.43469991 0.39360042 0.07827278
0.68091167 0.70007023 0.59198383 0.7159765 0.94152995 0.20837323
0.88920517 0.73631415 0.51757972 0.86433867 0.99739276 0.87181265
0.60934817 0.1415654 0.26862255 0.86607782 0.8037377 0.15251813
0.57528889 0.71268161 0.59574215 0.14614738 0.8869052 0.79427184
0.62951339 0.1960688 0.14721482 0.67329766 0.69120368 0.57920432
0.43638952 0.42105551 0.43623329 0.45970772 0.0164101 0.82716031
0.04376496 0.17994072 0.7406025 0.13037203 0.90631653 0.15836997
0.92749593 0.14770067 0.92960967 0.25145331 0.70695548 0.19555101
0.39257045 0.13575597 0.28789077 0.07375115 0.96403943 0.80313771
0.02917173 0.48666191 0.77912268 0.51756297 0.4639424 0.45322914
0.38812763 0.15560133 0.86403352 0.78385157 0.11912181 0.414019
0.15632493 0.60048972 0.9935398 0.06402706 0.99163941 0.02229378
0.10392711 0.01960803 0.6115714 0.9259645 0.69804233 0.10492756
0.42785002 0.72796694 0.24008469 0.74157092 0.71052113 0.62358871
0.91327252 0.0223508 0.08373651 0.91913787 0.26580305 0.5797042
0.36105558 0.43275102 0.98064654 0.85529366 0.23352471 0.92815969
0.38710368 0.17094735 0.84450817 0.21908917 0.66711474 0.13754708
0.49009395 0.40602442 0.62919072 0.83398323 0.17790809 0.04064466
0.31187587 0.03330163 0.79893053 0.99728246 0.01805598 0.98859583
0.80320987 0.13442005 0.46710442 0.45831797 0.13123517 0.29542599
0.49542796 0.51880929 0.37296186 0.76090064 0.7548383 0.48058113
0.83465054 0.52248016 0.21956368 0.46589712 0.75598859 0.42812441
0.73145622 0.16918261 0.02445512 0.00221758 0.36042122 0.8490867
0.33369163 0.67030894 0.38665223 0.16246456 0.45532514 0.74233386
0.96521169 0.36725452 0.29358193 0.46125555 0.53211547 0.26290122
0.65179942 0.25847337 0.81724057 0.99261451 0.21992123 0.89211556
0.29526797 0.17077962 0.54908317 0.91248904 0.97213494 0.63959052
0.14703749 0.60297332 0.66603429 0.49020522 0.87938218 0.62886067
0.44875409 0.52478597 0.38884389 0.02894025 0.90383319 0.21177767
0.43254025 0.93294878 0.27725741 0.34579388 0.53539685 0.79673216
0.94900626 0.77829025 0.35582298 0.23463353 0.00695873 0.52320125
0.57403479 0.35387197 0.39661362 0.515138 0.40797052 0.50839455
0.27935224 0.62498251 0.47124265 0.08278059 0.32522849 0.52843611
0.10311503 0.71629296 0.2100147 0.89140145 0.44107502 0.10752843
0.4129794 0.47232796 0.26068975 0.04262106 0.93320837 0.71532016
0.77673294 0.82747674 0.65518846 0.23316578 0.57558667 0.02203924]
```

```

0.35149204 0.44548199 0.28056153 0.86991441 0.07413055 0.26056994
0.55678097 0.17042038 0.1422051 0.46194247 0.04764662 0.23542336
0.16805661 0.57474717 0.74215953 0.96927519 0.53903366 0.68734613
0.56036842 0.34194987 0.10460768 0.95553562 0.46436383 0.68579989
0.9399565 0.82128614 0.81091975 0.11823248 0.16111312 0.64620289
0.91379759 0.98562967 0.67133502 0.64514114 0.83379166 0.02825724]
(1000, 300)
float64
8
2

```

```

In [57]: #create a 3D array and rearrange it
import numpy as np
array_4 = np.arange(24).reshape(2,3,4)
print(array_4)

```

```

[[[ 0  1  2  3]
   [ 4  5  6  7]
   [ 8  9 10 11]]

  [[12 13 14 15]
   [16 17 18 19]
   [20 21 22 23]]]

```

```

In [66]: import numpy as np
array_1 = np.arange(10)
print(array_1)
print(array_1[5])
print(array_1[[2,5,6]])
print(array_1[2:])
print(array_1[:2])
print(array_1[3:7])
print(array_1[2::3])

```

```

[0 1 2 3 4 5 6 7 8 9]
5
[2 5 6]
[2 3 4 5 6 7 8 9]
[0 1]
[3 4 5 6]
[2 5 8]

```

```

In [68]: import numpy as np
list1 = [1, 2, 3, 5, 4, 6, 7, 8, 5, 3, 2]
array = np.array(list1)
print(array)
print(array[0::2])

```

```

[1 2 3 5 4 6 7 8 5 3 2]
[1 3 4 7 5 2]

```

```
In [73]: # indexing in multidimensional arrays.Create a 2D array
import numpy as np
array = np.array([[2,5,7,5],[4,6,8,10],[10,12,15,19]])
print(array)
# print 3rd row second column
print(array[2,1])
# print 2nd row second column
print(array[1:])
```

```
[[ 2  5  7  5]
 [ 4  6  8 10]
 [10 12 15 19]]
12
[[ 4  6  8 10]
 [10 12 15 19]]
```

```
In [102]: #From a 2D array extract all the rows of the 2 column.
import numpy as np
array = np.array([[5,6,7],[7,6,5],[0,8,7]])
print(array)
print(array[ : ,1])
```

```
[[5 6 7]
 [7 6 5]
 [0 8 7]]
[6 6 8]
```

```
In [5]: import numpy as np

# Reshape a 1-D array to a 3 x 4 array
some_array = np.arange(0, 12).reshape(3, 4)
print(some_array)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

```
In [6]: import numpy as np

# Reshape a 1-D array to a 3 x 4 array
some_array = np.arange(0, 12).reshape(2, 6)
print(some_array)
```

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]
```

```
In [7]: # If you specify -1 as a dimension, the dimensions are automatically calculated  
# -1 means "whatever dimension is needed"  
import numpy as np  
  
# Reshape a 1-D array to a 3 x 4 array  
some_array = np.arange(0, 12).reshape(4, -1)  
print(some_array)
```

```
[[ 0  1  2]  
 [ 3  4  5]  
 [ 6  7  8]  
 [ 9 10 11]]
```

```
In [9]: #transposing an array  
import numpy as np  
  
# Reshape a 1-D array to a 3 x 4 array  
some_array = np.arange(0, 12).reshape(2, 6)  
print(some_array.T)
```

```
[[ 0  6]  
 [ 1  7]  
 [ 2  8]  
 [ 3  9]  
 [ 4 10]  
 [ 5 11]]
```

In [16]: *# stacking. In this you can vstack as number of columns is same (4)*

```
# Creating two arrays
array_1 = np.arange(12).reshape(3, 4)
array_2 = np.arange(20).reshape(5, 4)

array_3 = np.arange(15).reshape(5, 3)
array_4 = np.arange(20).reshape(5, 4)

print(array_1)
print("\n")
print(array_2)
print("\n")
print(np.vstack((array_1,array_2)))
print("\n")
print(np.hstack((array_3,array_4)))
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]]
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]]
```

```
[[ 0  1  2  0  1  2  3]
 [ 3  4  5  4  5  6  7]
 [ 6  7  8  8  9 10 11]
 [ 9 10 11 12 13 14 15]
 [12 13 14 16 17 18 19]]
```



In [18]: *# Basic mathematical operations*

```
a = np.arange(1, 20)
```

```
# sin, cos, exp, log
```

```
print(np.sin(a))
```

```
print(np.cos(a))
```

```
print(np.exp(a))
```

```
print(np.log(a))
```

```
print("\n")
```

```
print(np.sqrt(a))
```

```
[ 0.84147098  0.90929743  0.14112001 -0.7568025  -0.95892427 -0.2794155
 0.6569866  0.98935825  0.41211849 -0.54402111 -0.99999021 -0.53657292
 0.42016704  0.99060736  0.65028784 -0.28790332 -0.96139749 -0.75098725
 0.14987721]
[ 0.54030231 -0.41614684 -0.9899925  -0.65364362  0.28366219  0.96017029
 0.75390225 -0.14550003 -0.91113026 -0.83907153  0.0044257  0.84385396
 0.90744678  0.13673722 -0.75968791 -0.95765948 -0.27516334  0.66031671
 0.98870462]
[2.71828183e+00 7.38905610e+00 2.00855369e+01 5.45981500e+01
1.48413159e+02 4.03428793e+02 1.09663316e+03 2.98095799e+03
8.10308393e+03 2.20264658e+04 5.98741417e+04 1.62754791e+05
4.42413392e+05 1.20260428e+06 3.26901737e+06 8.88611052e+06
2.41549528e+07 6.56599691e+07 1.78482301e+08]
[0.          0.69314718 1.09861229 1.38629436 1.60943791 1.79175947
1.94591015 2.07944154 2.19722458 2.30258509 2.39789527 2.48490665
2.56494936 2.63905733 2.7080502  2.77258872 2.83321334 2.89037176
2.94443898]

[1.          1.41421356 1.73205081 2.          2.23606798 2.44948974
2.64575131 2.82842712 3.          3.16227766 3.31662479 3.46410162
3.60555128 3.74165739 3.87298335 4.          4.12310563 4.24264069
4.35889894]
```

In [24]: *#vectorizing a user function or operation*

```
a = np.arange(1, 20)
```

```
f = np.vectorize(lambda x: x/(x+1))
```

```
print(f(a))
```

```
[0.5          0.66666667 0.75          0.8          0.83333333 0.85714286
0.875          0.88888889 0.9          0.90909091 0.91666667 0.92307692
0.92857143 0.93333333 0.9375          0.94117647 0.94444444 0.94736842
0.95          ]
```

In [27]: *#Given an array, 'array\_3' divide each element with 5.*

```
#Hint: Create a vectorized function, then apply it to the array_3
```

```
a = np.arange(5, 21,5)
```

```
f = np.vectorize(lambda x: x/5)
```

```
print(f(a))
```

```
[1. 2. 3. 4.]
```

```
In [29]: # Linera algerbra functions
# Creating arrays
a = np.arange(1, 10).reshape(3, 3)
b= np.arange(1, 13).reshape(3, 4)
print(a)
print(b)

# Inverse
np.linalg.inv(a)
# Determinant
np.linalg.det(a)
# Eigenvalues and eigenvectors
np.linalg.eig(a)
# Multiply matrices
np.dot(a, b)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

```
Out[29]: array([[ 38,  44,  50,  56],
                [ 83,  98, 113, 128],
                [128, 152, 176, 200]])
```

```
In [30]: # Inverse
np.linalg.inv(a)
```

```
Out[30]: array([[ 3.15251974e+15, -6.30503948e+15,  3.15251974e+15],
                [-6.30503948e+15,  1.26100790e+16, -6.30503948e+15],
                [ 3.15251974e+15, -6.30503948e+15,  3.15251974e+15]])
```

```
In [32]: # Determinant
np.linalg.det(a)
```

```
Out[32]: -9.51619735392994e-16
```

```
In [33]: # Eigenvalues and eigenvectors
np.linalg.eig(a)
```

```
Out[33]: (array([ 1.61168440e+01, -1.11684397e+00, -9.75918483e-16]),
          array([[ -0.23197069, -0.78583024,  0.40824829],
                [-0.52532209, -0.08675134, -0.81649658],
                [-0.8186735 ,  0.61232756,  0.40824829]]))
```

```
In [1]: # import pandas, pd is an alias
import pandas as pd

# Creating a numeric pandas series
s = pd.Series([2, 4, 5, 6, 9])
print(s)
print(type(s))
```

```
0    2
1    4
2    5
3    6
4    9
dtype: int64
<class 'pandas.core.series.Series'>
```

```
In [2]: # creating a series of characters
# notice that the 'dtype' here is 'object'
char_series = pd.Series(['a', 'b', 'af'])
char_series
```

```
Out[2]: 0    a
1    b
2   af
dtype: object
```

```
In [4]: # creating a series of type datetime
date_series = pd.date_range(start = '11-09-2017', end = '12-12-2017')
date_series
type(date_series)
```

```
Out[4]: pandas.core.indexes.datetimes.DatetimeIndex
```

```
In [5]: # Indexing pandas series: Same as indexing 1-d numpy arrays or lists
# accessing the fourth element
s[3]

# accessing elements starting index = 2 till the end
s[2:]
```

```
Out[5]: 2    5
3    6
4    9
dtype: int64
```

```
In [6]: # accessing the fourth element
s[3]
```

```
Out[6]: 6
```

```
In [7]: # accessing the second and the fourth elements
# note that s[1, 3] will not work, you need to pass the indices [1, 3] as a list
s[[1, 3]]
```

```
Out[7]: 1    4
        3    6
        dtype: int64
```

```
In [10]: # using own function with apply statement
import pandas as pd

# Creating a numeric pandas series
s = pd.Series([2, 4, 5, 6, 9])
f = s.apply(lambda x: x**2)
print(f)
```

```
0    4
1   16
2   25
3   36
4   81
dtype: int64
```

```
In [11]: #dataframes
# keys become column names
df = pd.DataFrame({'name': ['Vinay', 'Kushal', 'Aman', 'Saif'],
                   'age': [22, 25, 24, 28],
                   'occupation': ['engineer', 'doctor', 'data analyst', 'teacher']})
df
```

```
Out[11]:
```

	age	name	occupation
0	22	Vinay	engineer
1	25	Kushal	doctor
2	24	Aman	data analyst
3	28	Saif	teacher

```
In [12]: # reading a CSV file as a dataframe
market_df = pd.read_csv("../global_sales_data/market_fact.csv")
```

```
-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-12-b22d4a109963> in <module>()
      1 # reading a CSV file as a dataframe
----> 2 market_df = pd.read_csv("../global_sales_data/market_fact.csv")

~\AppData\Local\anaconda3\lib\site-packages\pandas\io\parsers.py in parser_f(fi
lepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, p
refix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values,
skipinitialspace, skiprows, nrows, na_values, keep_default_na, na_filter, verb
ose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date
parser, dayfirst, iterator, chunksize, compression, thousands, decimal, liner
minator, quotechar, quoting, escapechar, comment, encoding, dialect, tupleize_c
ols, error_bad_lines, warn_bad_lines, skipfooter, skip_footer, doublequote, del
im_whitespace, as_recarray, compact_ints, use_unsigned, low_memory, buffer_line
s, memory_map, float_precision)
    707             skip_blank_lines=skip_blank_lines)
    708
--> 709         return _read(filepath_or_buffer, kwds)
    710
    711     parser_f.__name__ = name

~\AppData\Local\anaconda3\lib\site-packages\pandas\io\parsers.py in _read(filep
ath_or_buffer, kwds)
    447
    448     # Create the parser.
--> 449     parser = TextFileReader(filepath_or_buffer, **kwds)
    450
    451     if chunksize or iterator:

~\AppData\Local\anaconda3\lib\site-packages\pandas\io\parsers.py in __init__(se
lf, f, engine, **kwds)
    816         self.options['has_index_names'] = kwds['has_index_names']
    817
--> 818         self._make_engine(self.engine)
    819
    820     def close(self):

~\AppData\Local\anaconda3\lib\site-packages\pandas\io\parsers.py in _make_engin
e(self, engine)
   1047     def _make_engine(self, engine='c'):
   1048         if engine == 'c':
-> 1049             self._engine = CParserWrapper(self.f, **self.options)
   1050         else:
   1051             if engine == 'python':

~\AppData\Local\anaconda3\lib\site-packages\pandas\io\parsers.py in __init__(se
lf, src, **kwds)
   1693         kwds['allow_leading_cols'] = self.index_col is not False
   1694
-> 1695         self._reader = parsers.TextReader(src, **kwds)
   1696
   1697         # XXX
```

```
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()
```

```
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()
```

```
FileNotFoundError: File b'../global_sales_data/market_fact.csv' does not exist
```

```
In [17]: series = pd.Series([6,7,8,9,2,3,4,5])
series1 = series.apply(lambda x: x**2)
print(series)
print(series1)
```

```
0    6
1    7
2    8
3    9
4    2
5    3
6    4
7    5
dtype: int64
0    36
1    49
2    64
3    81
4     4
5     9
6    16
7    25
dtype: int64
```

```
In [23]: df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKKLvq851k2b8JOckF')
#Sort the dataframe on 'month' and 'day' in ascending order in the dataframe 'df'
df_2 = df.sort_values(by=['month','day'])
print(df_2.head(20))
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
241	4	4	apr	fri	83.0	23.3	85.3	2.3	16.7	20	3.1	0.0	0.00
442	6	5	apr	mon	87.9	24.9	41.6	3.7	10.9	64	3.1	0.0	3.35
19	6	4	apr	sat	86.3	27.4	97.1	5.1	9.3	44	4.5	0.0	0.00
239	7	5	apr	sun	81.9	3.0	7.9	3.5	13.4	75	1.8	0.0	0.00
469	6	3	apr	sun	91.0	14.6	25.6	12.3	13.7	33	9.4	0.0	61.13
470	5	4	apr	sun	91.0	14.6	25.6	12.3	17.6	27	5.8	0.0	0.00
176	6	5	apr	thu	81.5	9.1	55.2	2.7	5.8	54	5.8	0.0	4.61
196	6	5	apr	thu	81.5	9.1	55.2	2.7	5.8	54	5.8	0.0	10.93
240	6	3	apr	wed	88.0	17.2	43.5	3.8	15.2	51	2.7	0.0	0.00
12	6	5	aug	fri	63.5	70.8	665.3	0.8	17.0	72	6.7	0.0	0.00
78	1	2	aug	fri	90.1	108.0	529.8	12.5	14.7	66	2.7	0.0	0.00
142	8	6	aug	fri	90.1	108.0	529.8	12.5	21.2	51	8.9	0.0	0.61
184	8	6	aug	fri	93.9	135.7	586.7	15.1	20.8	34	4.9	0.0	6.96
195	2	5	aug	fri	93.9	135.7	586.7	15.1	23.5	36	5.4	0.0	10.02
261	3	4	aug	fri	91.6	112.4	573.0	8.9	11.2	84	7.6	0.0	3.30
262	2	4	aug	fri	91.6	112.4	573.0	8.9	21.4	42	3.1	0.0	4.25
263	6	3	aug	fri	91.1	141.1	629.1	7.1	19.3	39	3.6	0.0	1.56
264	4	4	aug	fri	94.3	167.6	684.4	13.0	21.8	53	3.1	0.0	6.54
388	6	4	aug	fri	94.8	227.0	706.7	12.0	23.3	34	3.1	0.0	28.74
389	7	4	aug	fri	94.8	227.0	706.7	12.0	23.3	34	3.1	0.0	0.00

```
In [6]: #selecting and indexing data in dataframe
import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKKLvq851k2b8JOckF')
print(df[2:7])
print('\n')
print(df[5::2].head())
type(df)
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0
11	7	5	sep	sat	92.8	73.2	713.0	22.6	19.3	38	4.0	0.0	0.0
13	6	5	sep	mon	90.9	126.5	686.5	7.0	21.3	42	2.2	0.0	0.0

Out[6]: pandas.core.frame.DataFrame

```
In [10]: import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKkLvq851k2b8JOCkF')
sales = df['ISI']
sales.head()
```

```
Out[10]: 0    5.1
1    6.7
2    6.7
3    9.0
4    9.6
Name: ISI, dtype: float64
```

```
In [12]: # Using df.column
import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKkLvq851k2b8JOCkF')
sales = df.ISI
sales.head()
```

```
Out[12]: 0    5.1
1    6.7
2    6.7
3    9.0
4    9.6
Name: ISI, dtype: float64
```

```
In [14]: import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKkLvq851k2b8JOCkF')
sales = df[['ISI', 'day', 'month']]
print(sales.head())
```

```
ISI  day month
0  5.1  fri   mar
1  6.7  tue   oct
2  6.7  sat   oct
3  9.0  fri   mar
4  9.6  sun   mar
```

```
In [17]: import pandas as pd
df = pd.read_csv('C:\Users\Z001MC7\Downloads\Introduction_to_Pandas\global_sales_data\market_fact')
print(df)
```

```
File "<ipython-input-17-e5720112804d>", line 2
df = pd.read_csv('C:\Users\Z001MC7\Downloads\Introduction_to_Pandas\global_sales_data\market_fact')
               ^
```

**SyntaxError:** (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape



```
In [20]: import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLkKlvq851k2b8JOCKF')
print(df.head())

market_df = pd.read_csv("../global_sales_data/market_fact.csv")
market_df.head()
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0

```
-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-20-7bf23c6e9e00> in <module>()
      3 print(df.head())
      4
----> 5 market_df = pd.read_csv("../global_sales_data/market_fact.csv")
      6 market_df.head()
```

```
~\AppData\Local\anaconda3\lib\site-packages\pandas\io\parsers.py in parser_f(fi
lepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, p
refix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values,
skipinitialspace, skiprows, nrows, na_values, keep_default_na, na_filter, verb
ose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_
parser, dayfirst, iterator, chunksize, compression, thousands, decimal, lineter
minator, quotechar, quoting, escapechar, comment, encoding, dialect, tupleize_c
ols, error_bad_lines, warn_bad_lines, skipfooter, skip_footer, doublequote, del
im_whitespace, as_recarray, compact_ints, use_unsigned, low_memory, buffer_line
s, memory_map, float_precision)
    707             skip_blank_lines=skip_blank_lines)
    708
--> 709         return _read(filepath_or_buffer, kwds)
    710
    711     parser_f.__name__ = name
```

```
~\AppData\Local\anaconda3\lib\site-packages\pandas\io\parsers.py in _read(filep
ath_or_buffer, kwds)
    447
    448     # Create the parser.
--> 449     parser = TextFileReader(filepath_or_buffer, **kwds)
    450
    451     if chunksize or iterator:
```

```
~\AppData\Local\anaconda3\lib\site-packages\pandas\io\parsers.py in __init__(se
lf, f, engine, **kwds)
    816         self.options['has_index_names'] = kwds['has_index_names']
    817
--> 818         self._make_engine(self.engine)
    819
    820     def close(self):
```

```
~\AppData\Local\anaconda3\lib\site-packages\pandas\io\parsers.py in _make_engin
e(self, engine)
    1047     def _make_engine(self, engine='c'):
```

```
1048         if engine == 'c':
-> 1049             self._engine = CParserWrapper(self.f, **self.options)
1050         else:
1051             if engine == 'python':

~\AppData\Local\anaconda3\lib\site-packages\pandas\io\parsers.py in __init__(self, src, **kws)
1693         kws['allow_leading_cols'] = self.index_col is not False
1694
-> 1695         self._reader = parsers.TextReader(src, **kws)
1696
1697         # XXX

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader.__cinit__()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._setup_parser_source()

FileNotFoundError: File b'../global_sales_data/market_fact.csv' does not exist
```

```
In [25]: import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKkLvq851k2b8JOckF')
print(df.head(10))
print('\n')

# Selecting a single element
# Note that 2, 4 corresponds to the third row and fifth column
print(df.iloc[2, 4])

print('\n')

# Selecting a single row, and all columns
# Select the 6th row, with label (and index) = 5
print(df.iloc[5])

print('\n')
# Select multiple rows using a list of indices
print(df.iloc[[3, 7, 8]])
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0

90.6

```
X      8
Y      6
month  aug
day    sun
FFMC   92.3
DMC    85.3
DC     488
ISI    14.7
temp   22.2
RH     29
wind   5.4
rain   0
area   0
Name: 5, dtype: object
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0

```
In [29]: import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKKLvq851k2b8JOckF')
print(df.head(10))
print('\n')
# Selecting rows using a range of integer indices
# Notice that 4 is included, 8 is not
print(df.iloc[4:8])
print('\n')
# Selecting a single column
# Notice that the column index starts at 0, and 2 represents the third column (Cu
print((df.iloc[:, 2]).head())
print('\n')
# Selecting multiple columns
print(df.iloc[:, 3:8])
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0

```
0    mar
1    oct
2    oct
3    mar
4    mar
Name: month, dtype: object
```

	day	FFMC	DMC	DC	ISI
0	fri	86.2	26.2	94.3	5.1
1	tue	90.6	35.4	669.1	6.7
2	sat	90.6	43.7	686.9	6.7
3	fri	91.7	33.3	77.5	9.0
4	sun	89.3	51.3	102.2	9.6
5	sun	92.3	85.3	488.0	14.7
6	mon	92.3	88.9	495.6	8.5
7	mon	91.5	145.4	608.2	10.7
8	tue	91.0	129.5	692.6	7.0
9	sat	92.5	88.0	698.6	7.1
10	sat	92.5	88.0	698.6	7.1
11	sat	92.8	73.2	713.0	22.6

12	fri	63.5	70.8	665.3	0.8
13	mon	90.9	126.5	686.5	7.0
14	wed	92.9	133.3	699.6	9.2
15	fri	93.3	141.2	713.9	13.9
16	sat	91.7	35.8	80.8	7.8
17	mon	84.9	32.8	664.2	3.0
18	wed	89.2	27.9	70.8	6.3
19	sat	86.3	27.4	97.1	5.1
20	tue	91.0	129.5	692.6	7.0
21	mon	91.8	78.5	724.3	9.2
22	sun	94.3	96.3	200.0	56.1
23	sat	90.2	110.9	537.4	6.2
24	sat	93.5	139.4	594.2	20.3
25	sun	91.4	142.4	601.4	10.6
26	fri	92.4	117.9	668.0	12.2
27	mon	90.9	126.5	686.5	7.0
28	sat	93.4	145.4	721.4	8.1
29	sun	93.5	149.3	728.6	8.1
..	...	...	...	...	...
487	tue	95.1	141.3	605.8	17.7
488	tue	95.1	141.3	605.8	17.7
489	wed	95.1	141.3	605.8	17.7
490	wed	95.1	141.3	605.8	17.7
491	thu	95.8	152.0	624.1	13.8
492	fri	95.9	158.0	633.6	11.3
493	fri	95.9	158.0	633.6	11.3
494	sat	96.0	164.0	643.0	14.0
495	mon	96.2	175.5	661.8	16.8
496	mon	96.2	175.5	661.8	16.8
497	tue	96.1	181.1	671.2	14.3
498	tue	96.1	181.1	671.2	14.3
499	tue	96.1	181.1	671.2	14.3
500	tue	96.1	181.1	671.2	14.3
501	tue	96.1	181.1	671.2	14.3
502	tue	96.1	181.1	671.2	14.3
503	wed	94.5	139.4	689.1	20.0
504	wed	94.5	139.4	689.1	20.0
505	thu	91.0	163.2	744.4	10.1
506	fri	91.0	166.9	752.6	7.1
507	fri	91.0	166.9	752.6	7.1
508	fri	91.0	166.9	752.6	7.1
509	fri	91.0	166.9	752.6	7.1
510	fri	91.0	166.9	752.6	7.1
511	sun	81.6	56.7	665.6	1.9
512	sun	81.6	56.7	665.6	1.9
513	sun	81.6	56.7	665.6	1.9
514	sun	81.6	56.7	665.6	1.9
515	sat	94.4	146.0	614.7	11.3
516	tue	79.5	3.0	106.7	1.1

[517 rows x 5 columns]

```
In [30]: import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLkLvq851k2b8JOckF')
print(df.head(10))
print('\n')
# Selecting multiple rows and columns
print(df.iloc[3:6, 2:5])
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0

	month	day	FFMC
3	mar	fri	91.7
4	mar	sun	89.3
5	aug	sun	92.3

```
In [37]: #Dataframe iloc
#Description
#Using iloc index the dataframe to print all the rows of the columns at index 3,4,
#Hint: Use 3,4,5 not 2,3,4
import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKKLvq851k2b8JOckF')
print(df.head(10))
print('\n')
print((df.iloc[:, 3:6]).head())

# or
print('\n')
print((df.iloc[:, [3,4,5]]).head())
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0

	day	FFMC	DMC
0	fri	86.2	26.2
1	tue	90.6	35.4
2	sat	90.6	43.7
3	fri	91.7	33.3
4	sun	89.3	51.3

	day	FFMC	DMC
0	fri	86.2	26.2
1	tue	90.6	35.4
2	sat	90.6	43.7
3	fri	91.7	33.3
4	sun	89.3	51.3

```
In [40]: #Label indexing

import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKkLvq851k2b8JOckF')
print(df.head(10))
print('\n')

# Selecting a single element
# Select row label = 3 and column label = 'FFMC'
print(df.loc[3, 'FFMC'])

print('\n')
# Selecting a single row using a single label
# df.loc reads 5 as a label, not index
print((df.loc[5]).head())
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0

91.7

```
X          8
Y          6
month      aug
day        sun
FFMC      92.3
Name: 5, dtype: object
```



```
In [42]: #Using loc function print out all the columns and rows from 2 to 20 of the 'df' d
import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLkKlvq851k2b8JOckF')
print(df.head(25))
print('\n')
print(df.loc[2:20])
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0
10	7	5	sep	sat	92.5	88.0	698.6	7.1	17.8	51	7.2	0.0	0.0
11	7	5	sep	sat	92.8	73.2	713.0	22.6	19.3	38	4.0	0.0	0.0
12	6	5	aug	fri	63.5	70.8	665.3	0.8	17.0	72	6.7	0.0	0.0
13	6	5	sep	mon	90.9	126.5	686.5	7.0	21.3	42	2.2	0.0	0.0
14	6	5	sep	wed	92.9	133.3	699.6	9.2	26.4	21	4.5	0.0	0.0
15	6	5	sep	fri	93.3	141.2	713.9	13.9	22.9	44	5.4	0.0	0.0
16	5	5	mar	sat	91.7	35.8	80.8	7.8	15.1	27	5.4	0.0	0.0
17	8	5	oct	mon	84.9	32.8	664.2	3.0	16.7	47	4.9	0.0	0.0
18	6	4	mar	wed	89.2	27.9	70.8	6.3	15.9	35	4.0	0.0	0.0
19	6	4	apr	sat	86.3	27.4	97.1	5.1	9.3	44	4.5	0.0	0.0
20	6	4	sep	tue	91.0	129.5	692.6	7.0	18.3	40	2.7	0.0	0.0
21	5	4	sep	mon	91.8	78.5	724.3	9.2	19.1	38	2.7	0.0	0.0
22	7	4	jun	sun	94.3	96.3	200.0	56.1	21.0	44	4.5	0.0	0.0
23	7	4	aug	sat	90.2	110.9	537.4	6.2	19.5	43	5.8	0.0	0.0
24	7	4	aug	sat	93.5	139.4	594.2	20.3	23.7	32	5.8	0.0	0.0

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0
10	7	5	sep	sat	92.5	88.0	698.6	7.1	17.8	51	7.2	0.0	0.0
11	7	5	sep	sat	92.8	73.2	713.0	22.6	19.3	38	4.0	0.0	0.0
12	6	5	aug	fri	63.5	70.8	665.3	0.8	17.0	72	6.7	0.0	0.0
13	6	5	sep	mon	90.9	126.5	686.5	7.0	21.3	42	2.2	0.0	0.0
14	6	5	sep	wed	92.9	133.3	699.6	9.2	26.4	21	4.5	0.0	0.0
15	6	5	sep	fri	93.3	141.2	713.9	13.9	22.9	44	5.4	0.0	0.0
16	5	5	mar	sat	91.7	35.8	80.8	7.8	15.1	27	5.4	0.0	0.0
17	8	5	oct	mon	84.9	32.8	664.2	3.0	16.7	47	4.9	0.0	0.0
18	6	4	mar	wed	89.2	27.9	70.8	6.3	15.9	35	4.0	0.0	0.0
19	6	4	apr	sat	86.3	27.4	97.1	5.1	9.3	44	4.5	0.0	0.0
20	6	4	sep	tue	91.0	129.5	692.6	7.0	18.3	40	2.7	0.0	0.0

```
In [52]: import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKKLvq851k2b8JOckF')
print(df.head(25))
print('\n')
# Select all rows where Sales > 3000
# First, we get a boolean array where True corresponds to rows having Sales > 3000
print(df.loc[(df.DMC > 100)])
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0
10	7	5	sep	sat	92.5	88.0	698.6	7.1	17.8	51	7.2	0.0	0.0
11	7	5	sep	sat	92.8	73.2	713.0	22.6	19.3	38	4.0	0.0	0.0
12	6	5	aug	fri	63.5	70.8	665.3	0.8	17.0	72	6.7	0.0	0.0
13	6	5	sep	mon	90.9	126.5	686.5	7.0	21.3	42	2.2	0.0	0.0
14	6	5	sep	wed	92.9	133.3	699.6	9.2	26.4	21	4.5	0.0	0.0
15	6	5	sep	fri	93.3	141.2	713.9	13.9	22.9	44	5.4	0.0	0.0
16	5	5	mar	sat	91.7	35.8	80.8	7.8	15.1	27	5.4	0.0	0.0
17	8	5	oct	mon	84.9	32.8	664.2	3.0	16.7	47	4.9	0.0	0.0
18	6	4	mar	wed	89.2	27.9	70.8	6.3	15.9	35	4.0	0.0	0.0
19	6	4	apr	sat	86.3	27.4	97.1	5.1	9.3	44	4.5	0.0	0.0
20	6	4	sep	tue	91.0	129.5	692.6	7.0	18.3	40	2.7	0.0	0.0
21	5	4	sep	mon	91.8	78.5	724.3	9.2	19.1	38	2.7	0.0	0.0
22	7	4	jun	sun	94.3	96.3	200.0	56.1	21.0	44	4.5	0.0	0.0
23	7	4	aug	sat	90.2	110.9	537.4	6.2	19.5	43	5.8	0.0	0.0
24	7	4	aug	sat	93.5	139.4	594.2	20.3	23.7	32	5.8	0.0	0.0

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.00
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.00
13	6	5	sep	mon	90.9	126.5	686.5	7.0	21.3	42	2.2	0.0	0.00
14	6	5	sep	wed	92.9	133.3	699.6	9.2	26.4	21	4.5	0.0	0.00
15	6	5	sep	fri	93.3	141.2	713.9	13.9	22.9	44	5.4	0.0	0.00
20	6	4	sep	tue	91.0	129.5	692.6	7.0	18.3	40	2.7	0.0	0.00
23	7	4	aug	sat	90.2	110.9	537.4	6.2	19.5	43	5.8	0.0	0.00
24	7	4	aug	sat	93.5	139.4	594.2	20.3	23.7	32	5.8	0.0	0.00
25	7	4	aug	sun	91.4	142.4	601.4	10.6	16.3	60	5.4	0.0	0.00
26	7	4	sep	fri	92.4	117.9	668.0	12.2	19.0	34	5.8	0.0	0.00
27	7	4	sep	mon	90.9	126.5	686.5	7.0	19.4	48	1.3	0.0	0.00
28	6	3	sep	sat	93.4	145.4	721.4	8.1	30.2	24	2.7	0.0	0.00
29	6	3	sep	sun	93.5	149.3	728.6	8.1	22.8	39	3.6	0.0	0.00
42	4	4	aug	tue	94.8	108.3	647.1	17.0	16.6	54	5.4	0.0	0.00
46	5	6	sep	mon	90.9	126.5	686.5	7.0	14.7	70	3.6	0.0	0.00
50	4	4	sep	thu	92.9	137.0	706.4	9.2	20.8	17	1.3	0.0	0.00
52	4	3	aug	wed	92.1	111.2	654.1	9.6	20.4	42	4.9	0.0	0.00
53	4	3	aug	wed	92.1	111.2	654.1	9.6	20.4	42	4.9	0.0	0.00
54	4	3	aug	thu	91.7	114.3	661.3	6.3	17.6	45	3.6	0.0	0.00

55	4	3	sep	thu	92.9	137.0	706.4	9.2	27.7	24	2.2	0.0	0.00
64	2	2	aug	mon	91.1	103.2	638.8	5.8	23.1	31	3.1	0.0	0.00
65	2	2	aug	thu	91.7	114.3	661.3	6.3	18.6	44	4.5	0.0	0.00
66	2	2	sep	fri	92.4	117.9	668.0	12.2	23.0	37	4.5	0.0	0.00
67	2	2	sep	fri	92.4	117.9	668.0	12.2	19.6	33	5.4	0.0	0.00
68	2	2	sep	fri	92.4	117.9	668.0	12.2	19.6	33	6.3	0.0	0.00
73	5	4	aug	tue	88.8	147.3	614.5	9.0	17.3	43	4.5	0.0	0.00
74	5	4	sep	fri	93.3	141.2	713.9	13.9	27.6	30	1.3	0.0	0.00
78	1	2	aug	fri	90.1	108.0	529.8	12.5	14.7	66	2.7	0.0	0.00
79	1	2	aug	tue	91.0	121.2	561.6	7.0	21.6	19	6.7	0.0	0.00
80	1	2	aug	sun	91.4	142.4	601.4	10.6	19.5	39	6.3	0.0	0.00
..	..	..	...	...	...	...	...	...	...	..	...	...	...
482	3	4	aug	sun	94.9	130.3	587.1	14.1	23.4	40	5.8	0.0	1.29
483	8	6	aug	sun	94.9	130.3	587.1	14.1	31.0	27	5.4	0.0	0.00
484	2	5	aug	sun	94.9	130.3	587.1	14.1	33.1	25	4.0	0.0	26.43
485	2	4	aug	mon	95.0	135.5	596.3	21.3	30.6	28	3.6	0.0	2.07
486	5	4	aug	tue	95.1	141.3	605.8	17.7	24.1	43	6.3	0.0	2.00
487	5	4	aug	tue	95.1	141.3	605.8	17.7	26.4	34	3.6	0.0	16.40
488	4	4	aug	tue	95.1	141.3	605.8	17.7	19.4	71	7.6	0.0	46.70
489	4	4	aug	wed	95.1	141.3	605.8	17.7	20.6	58	1.3	0.0	0.00
490	4	4	aug	wed	95.1	141.3	605.8	17.7	28.7	33	4.0	0.0	0.00
491	4	4	aug	thu	95.8	152.0	624.1	13.8	32.4	21	4.5	0.0	0.00
492	1	3	aug	fri	95.9	158.0	633.6	11.3	32.4	27	2.2	0.0	0.00
493	1	3	aug	fri	95.9	158.0	633.6	11.3	27.5	29	4.5	0.0	43.32
494	6	6	aug	sat	96.0	164.0	643.0	14.0	30.8	30	4.9	0.0	8.59
495	6	6	aug	mon	96.2	175.5	661.8	16.8	23.9	42	2.2	0.0	0.00
496	4	5	aug	mon	96.2	175.5	661.8	16.8	32.6	26	3.1	0.0	2.77
497	3	4	aug	tue	96.1	181.1	671.2	14.3	32.3	27	2.2	0.0	14.68
498	6	5	aug	tue	96.1	181.1	671.2	14.3	33.3	26	2.7	0.0	40.54
499	7	5	aug	tue	96.1	181.1	671.2	14.3	27.3	63	4.9	6.4	10.82
500	8	6	aug	tue	96.1	181.1	671.2	14.3	21.6	65	4.9	0.8	0.00
501	7	5	aug	tue	96.1	181.1	671.2	14.3	21.6	65	4.9	0.8	0.00
502	4	4	aug	tue	96.1	181.1	671.2	14.3	20.7	69	4.9	0.4	0.00
503	2	4	aug	wed	94.5	139.4	689.1	20.0	29.2	30	4.9	0.0	1.95
504	4	3	aug	wed	94.5	139.4	689.1	20.0	28.9	29	4.9	0.0	49.59
505	1	2	aug	thu	91.0	163.2	744.4	10.1	26.7	35	1.8	0.0	5.80
506	1	2	aug	fri	91.0	166.9	752.6	7.1	18.5	73	8.5	0.0	0.00
507	2	4	aug	fri	91.0	166.9	752.6	7.1	25.9	41	3.6	0.0	0.00
508	1	2	aug	fri	91.0	166.9	752.6	7.1	25.9	41	3.6	0.0	0.00
509	5	4	aug	fri	91.0	166.9	752.6	7.1	21.1	71	7.6	1.4	2.17
510	6	5	aug	fri	91.0	166.9	752.6	7.1	18.2	62	5.4	0.0	0.43
515	1	4	aug	sat	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00

[289 rows x 13 columns]

```
In [55]: import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKKLvq851k2b8JOckF')
print(df.head())
print('\n')
# Select all rows where Sales > 3000
# First, we get a boolean array where True corresponds to rows having Sales > 3000
print(df.loc[(df.DMC > 100) & (df.DC > 800)])
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
315	3	4	sep	wed	91.2	134.7	817.5	7.2	18.5	30	2.7	0.0	0.00
323	3	5	sep	thu	90.7	136.9	822.8	6.8	12.9	39	2.7	0.0	2.18
342	6	3	sep	mon	91.5	130.1	807.1	7.5	20.6	37	1.8	0.0	0.00
343	8	6	sep	mon	91.5	130.1	807.1	7.5	15.9	51	4.5	0.0	2.18
344	6	3	sep	mon	91.5	130.1	807.1	7.5	12.2	66	4.9	0.0	6.10
345	2	2	sep	mon	91.5	130.1	807.1	7.5	16.8	43	3.1	0.0	5.83
346	1	4	sep	mon	91.5	130.1	807.1	7.5	21.3	35	2.2	0.0	28.19
366	4	5	sep	tue	91.1	132.3	812.1	12.5	15.9	38	5.4	0.0	1.75
367	4	5	sep	tue	91.1	132.3	812.1	12.5	16.4	27	3.6	0.0	0.00
369	4	5	sep	sun	91.0	276.3	825.1	7.1	13.8	77	7.6	0.0	0.00
370	7	4	sep	sun	91.0	276.3	825.1	7.1	13.8	77	7.6	0.0	11.06
374	6	5	sep	fri	90.3	290.0	855.3	7.4	10.3	78	4.0	0.0	18.30
384	8	4	aug	sat	91.6	273.8	819.1	7.7	21.3	44	4.5	0.0	12.18
392	1	3	sep	sun	91.0	276.3	825.1	7.1	21.9	43	4.0	0.0	70.76
406	6	5	sep	sat	87.1	291.3	860.6	4.0	17.0	67	4.9	0.0	3.95
408	4	3	sep	fri	90.3	290.0	855.3	7.4	19.9	44	3.1	0.0	7.80
430	7	4	sep	thu	89.7	287.2	849.3	6.8	19.4	45	3.6	0.0	0.00
434	1	4	aug	fri	90.6	269.8	811.2	5.5	22.2	45	3.6	0.0	0.00
440	5	4	sep	fri	90.3	290.0	855.3	7.4	16.2	58	3.6	0.0	0.00
444	2	5	sep	fri	90.3	290.0	855.3	7.4	16.2	58	3.6	0.0	9.96
448	7	4	sep	wed	89.7	284.9	844.0	10.1	10.5	77	4.0	0.0	0.00
453	4	5	aug	thu	89.4	266.2	803.3	5.6	17.4	54	3.1	0.0	0.00
459	7	4	aug	sat	91.6	273.8	819.1	7.7	15.5	72	8.0	0.0	1.94
462	1	4	sep	sun	91.0	276.3	825.1	7.1	14.5	76	7.6	0.0	3.71

In [57]: *#Print all the columns and the rows where 'area' is greater than 0, 'wind' is greater than 1, and 'temp' is greater than 15*

```
import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLkKlvq851k2b8JOckF')
print(df.loc[(df.area > 0) & (df.wind > 1) & (df.temp > 15)])
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
138	9	9	jul	tue	85.8	48.3	313.4	3.9	18.0	42	2.7	0.0	0.36
139	1	4	sep	tue	91.0	129.5	692.6	7.0	21.7	38	2.2	0.0	0.43
140	2	5	sep	mon	90.9	126.5	686.5	7.0	21.9	39	1.8	0.0	0.47
141	1	2	aug	wed	95.5	99.9	513.3	13.2	23.3	31	4.5	0.0	0.55
142	8	6	aug	fri	90.1	108.0	529.8	12.5	21.2	51	8.9	0.0	0.61
143	1	2	jul	sat	90.0	51.3	296.3	8.7	16.6	53	5.4	0.0	0.71
144	2	5	aug	wed	95.5	99.9	513.3	13.2	23.8	32	5.4	0.0	0.77
145	6	5	aug	thu	95.2	131.7	578.8	10.4	27.4	22	4.0	0.0	0.90
147	8	3	sep	tue	84.4	73.4	671.9	3.2	24.2	28	3.6	0.0	0.96
148	2	2	aug	tue	94.8	108.3	647.1	17.0	17.4	43	6.7	0.0	1.07
149	8	6	sep	thu	93.7	80.9	685.2	17.9	23.7	25	4.5	0.0	1.12
150	6	5	jun	fri	92.5	56.4	433.3	7.1	23.2	39	5.4	0.0	1.19
151	9	9	jul	sun	90.1	68.6	355.2	7.2	24.8	29	2.2	0.0	1.36
152	3	4	jul	sat	90.1	51.2	424.1	6.2	24.6	43	1.8	0.0	1.43
153	5	4	sep	fri	94.3	85.1	692.3	15.9	20.1	47	4.9	0.0	1.46
154	1	5	sep	sat	93.4	145.4	721.4	8.1	29.6	27	2.7	0.0	1.46
155	7	4	aug	sun	94.8	108.3	647.1	17.0	16.4	47	1.3	0.0	1.56
156	2	4	sep	sat	93.4	145.4	721.4	8.1	28.6	27	2.2	0.0	1.61
157	2	2	aug	wed	92.1	111.2	654.1	9.6	18.4	45	3.6	0.0	1.63
158	2	4	aug	wed	92.1	111.2	654.1	9.6	20.5	35	4.0	0.0	1.64
159	7	4	sep	fri	92.4	117.9	668.0	12.2	19.0	34	5.8	0.0	1.69
160	7	4	mar	mon	90.1	39.7	86.6	6.2	16.1	29	3.1	0.0	1.75
161	6	4	aug	thu	95.2	131.7	578.8	10.4	20.3	41	4.0	0.0	1.90
162	6	3	mar	sat	90.6	50.1	100.4	7.8	15.2	31	8.5	0.0	1.94
163	8	6	sep	sat	92.5	121.1	674.4	8.6	17.8	56	1.8	0.0	1.95
164	8	5	sep	sun	89.7	90.0	704.4	4.8	17.8	67	2.2	0.0	2.01
167	6	5	aug	wed	96.0	127.1	570.5	16.5	23.4	33	4.5	0.0	2.51
169	8	6	aug	thu	95.2	131.7	578.8	10.4	20.7	45	2.2	0.0	2.55
170	5	4	sep	wed	92.9	133.3	699.6	9.2	21.9	35	1.8	0.0	2.57
171	8	6	aug	wed	85.6	90.4	609.6	6.6	17.4	50	4.0	0.0	2.69
..	..	..	...	...	...	...	...	...	...	..	...	...	...
459	7	4	aug	sat	91.6	273.8	819.1	7.7	15.5	72	8.0	0.0	1.94
471	4	3	may	fri	89.6	25.4	73.7	5.7	18.0	40	4.0	0.0	38.48
473	9	4	jun	sat	90.5	61.1	252.6	9.4	24.5	50	3.1	0.0	70.32
474	4	3	jun	thu	93.0	103.8	316.7	10.8	26.4	35	2.7	0.0	10.08
475	2	5	jun	thu	93.7	121.7	350.2	18.0	22.7	40	9.4	0.0	3.19
476	4	3	jul	thu	93.5	85.3	395.0	9.9	27.2	28	1.3	0.0	1.76
477	4	3	jul	sun	93.7	101.3	423.4	14.7	26.1	45	4.0	0.0	7.36
478	7	4	jul	sun	93.7	101.3	423.4	14.7	18.2	82	4.5	0.0	2.21
479	7	4	jul	mon	89.2	103.9	431.6	6.4	22.6	57	4.9	0.0	278.53
480	9	9	jul	thu	93.2	114.4	560.0	9.5	30.2	25	4.5	0.0	2.75
482	3	4	aug	sun	94.9	130.3	587.1	14.1	23.4	40	5.8	0.0	1.29
484	2	5	aug	sun	94.9	130.3	587.1	14.1	33.1	25	4.0	0.0	26.43
485	2	4	aug	mon	95.0	135.5	596.3	21.3	30.6	28	3.6	0.0	2.07
486	5	4	aug	tue	95.1	141.3	605.8	17.7	24.1	43	6.3	0.0	2.00
487	5	4	aug	tue	95.1	141.3	605.8	17.7	26.4	34	3.6	0.0	16.40
488	4	4	aug	tue	95.1	141.3	605.8	17.7	19.4	71	7.6	0.0	46.70
493	1	3	aug	fri	95.9	158.0	633.6	11.3	27.5	29	4.5	0.0	43.32
494	6	6	aug	sat	96.0	164.0	643.0	14.0	30.8	30	4.9	0.0	8.59

496	4	5	aug	mon	96.2	175.5	661.8	16.8	32.6	26	3.1	0.0	2.77
497	3	4	aug	tue	96.1	181.1	671.2	14.3	32.3	27	2.2	0.0	14.68
498	6	5	aug	tue	96.1	181.1	671.2	14.3	33.3	26	2.7	0.0	40.54
499	7	5	aug	tue	96.1	181.1	671.2	14.3	27.3	63	4.9	6.4	10.82
503	2	4	aug	wed	94.5	139.4	689.1	20.0	29.2	30	4.9	0.0	1.95
504	4	3	aug	wed	94.5	139.4	689.1	20.0	28.9	29	4.9	0.0	49.59
505	1	2	aug	thu	91.0	163.2	744.4	10.1	26.7	35	1.8	0.0	5.80
509	5	4	aug	fri	91.0	166.9	752.6	7.1	21.1	71	7.6	1.4	2.17
510	6	5	aug	fri	91.0	166.9	752.6	7.1	18.2	62	5.4	0.0	0.43
512	4	3	aug	sun	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44
513	2	4	aug	sun	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29
514	7	4	aug	sun	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16

[212 rows x 13 columns]

In [6]:

#Merging and concatenating dataframes

```
import pandas as pd
market_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/global_
customer_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/glob
product_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/globa
shipping_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/glob
orders_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/global_
print(market_df.head())
print('\n')
print(customer_df.head())
print('\n')
print(product_df.head())
print('\n')
print(shipping_df.head())
print('\n')
print(orders_df.head())
print('\n')
```

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	\
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	

	Profit	Shipping_Cost	Product_Base_Margin
0	-30.51	3.60	0.56
1	4.56	0.93	0.54
2	1148.90	2.50	0.59
3	729.34	14.30	0.37
4	1219.87	26.30	0.38

	Customer_Name	Province	Region	Customer_Segment	Cust_id
0	MUHAMMED MACINTYRE	NUNAVUT	NUNAVUT	SMALL BUSINESS	Cust_1
1	BARRY FRENCH	NUNAVUT	NUNAVUT	CONSUMER	Cust_2
2	CLAY ROZENDAL	NUNAVUT	NUNAVUT	CORPORATE	Cust_3
3	CARLOS SOLTERO	NUNAVUT	NUNAVUT	CONSUMER	Cust_4
4	CARL JACKSON	NUNAVUT	NUNAVUT	CORPORATE	Cust_5

	Product_Category	Product_Sub_Category	Prod_id
0	OFFICE SUPPLIES	STORAGE & ORGANIZATION	Prod_1
1	OFFICE SUPPLIES	APPLIANCES	Prod_2
2	OFFICE SUPPLIES	BINDERS AND BINDER ACCESSORIES	Prod_3
3	TECHNOLOGY	TELEPHONES AND COMMUNICATION	Prod_4
4	FURNITURE	OFFICE FURNISHINGS	Prod_5

Order_ID	Ship_Mode	Ship_Date	Ship_id
----------	-----------	-----------	---------

0	3	REGULAR AIR	20-10-2010	SHP_1
1	293	DELIVERY TRUCK	02-10-2012	SHP_2
2	293	REGULAR AIR	03-10-2012	SHP_3
3	483	REGULAR AIR	12-07-2011	SHP_4
4	515	REGULAR AIR	30-08-2010	SHP_5

	Order_ID	Order_Date	Order_Priority	Ord_id
0	3	13-10-2010	LOW	Ord_1
1	293	01-10-2012	HIGH	Ord_2
2	483	10-07-2011	HIGH	Ord_3
3	515	28-08-2010	NOT SPECIFIED	Ord_4
4	613	17-06-2011	HIGH	Ord_5

```
In [11]: #Merging and concatenating dataframes
import pandas as pd
market_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/global_
customer_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/glob
product_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/globa
shipping_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/glob
orders_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/global

# Merging the dataframes
# Note that Cust_id is the common column/key, which is provided to the 'on' argument
# how = 'inner' makes sure that only the customer ids present in both dfs are included
merge_df = pd.merge(market_df, customer_df, how='inner', on='Cust_id')
print(merge_df.head())
```

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	\
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	
1	Ord_5406	Prod_13	SHP_7549	Cust_1818	42.27	0.01	13	
2	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	
3	Ord_5456	Prod_6	SHP_7625	Cust_1818	2337.89	0.09	43	
4	Ord_5485	Prod_17	SHP_7664	Cust_1818	4233.15	0.08	35	

	Profit	Shipping_Cost	Product_Base_Margin	Customer_Name	Province	Region
0	-30.51	3.60	0.56	AARON BERGMAN	ALBERTA	WEST
1	4.56	0.93	0.54	AARON BERGMAN	ALBERTA	WEST
2	1148.90	2.50	0.59	AARON BERGMAN	ALBERTA	WEST
3	729.34	14.30	0.37	AARON BERGMAN	ALBERTA	WEST
4	1219.87	26.30	0.38	AARON BERGMAN	ALBERTA	WEST

	Customer_Segment
0	CORPORATE
1	CORPORATE
2	CORPORATE
3	CORPORATE
4	CORPORATE



```
In [14]: #Merging and concatenating dataframes
import pandas as pd
market_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/global_
customer_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/glob
product_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/globa
shipping_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/glob
orders_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/global_
df_1 = pd.merge(market_df, customer_df, how='inner', on='Cust_id')
# Example 2: Select all orders from product category = office supplies and from t
# We now need to merge the product_df

df_2 = pd.merge(df_1, product_df, how='inner', on='Prod_id')
df_2.head()
print('\n')
# Select all orders from product category = office supplies and from the corporat
df_2.loc[(df_2['Product_Category']=='OFFICE SUPPLIES') & (df_2['Customer_Segment']
```

```
In [18]: # dataframes having the same columns
df1 = pd.DataFrame({'Name': ['Aman', 'Joy', 'Rashmi', 'Saif'],
                    'Age': ['34', '31', '22', '33'],
                    'Gender': ['M', 'M', 'F', 'M']}
                  )

df2 = pd.DataFrame({'Name': ['Akhil', 'Asha', 'Preeti'],
                    'Age': ['31', '22', '23'],
                    'Gender': ['M', 'F', 'F']}
                  )

df1
print(df1)
print('\n')
print(df2)
print('\n')
# To concatenate them, one on top of the other, you can use pd.concat
# The first argument is a sequence (list) of dataframes
# axis = 0 indicates that we want to concat along the row axis
print(pd.concat([df1, df2], axis = 0))
print('\n')
# A useful and intuitive alternative to concat along the rows is the append() fun
# It concatenates along the rows
df1.append(df2)
```

	Age	Gender	Name
0	34	M	Aman
1	31	M	Joy
2	22	F	Rashmi
3	33	M	Saif

	Age	Gender	Name
0	31	M	Akhil
1	22	F	Asha
2	23	F	Preeti

	Age	Gender	Name
0	34	M	Aman
1	31	M	Joy
2	22	F	Rashmi
3	33	M	Saif
0	31	M	Akhil
1	22	F	Asha
2	23	F	Preeti

Out[18]:

	Age	Gender	Name
0	34	M	Aman
1	31	M	Joy
2	22	F	Rashmi
3	33	M	Saif

	Age	Gender	Name
0	31	M	Akhil
1	22	F	Asha
2	23	F	Preeti

```
In [19]: df1 = pd.DataFrame({'Name': ['Aman', 'Joy', 'Rashmi', 'Saif'],
                             'Age': ['34', '31', '22', '33'],
                             'Gender': ['M', 'M', 'F', 'M']}
                             )
df1
df2 = pd.DataFrame({'School': ['RK Public', 'JSP', 'Carmel Convent', 'St. Paul'],
                    'Graduation Marks': ['84', '89', '76', '91']}
                    )
df2
```

Out[19]:

	Graduation Marks	School
0	84	RK Public
1	89	JSP
2	76	Carmel Convent
3	91	St. Paul

```
In [22]: # Loading libraries and files for Grouping and summarization
import numpy as np

import pandas as pd
market_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/global_
customer_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/glob
product_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/globa
shipping_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/glob
orders_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/global_

#First, we will merge all the dataframes, so we have all the data in one master_d
df_1 = pd.merge(market_df, customer_df, how='inner', on='Cust_id')
df_2 = pd.merge(df_1, product_df, how='inner', on='Prod_id')
df_3 = pd.merge(df_2, shipping_df, how='inner', on='Ship_id')
master_df = pd.merge(df_3, orders_df, how='inner', on='Ord_id')

master_df.head()
```

Out[22]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
2	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.02	0.03	23	-47.64	
3	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.05	0.04	27	23.12	
4	Ord_5484	Prod_16	SHP_7663	Cust_1820	322.82	0.05	35	-17.58	

5 rows × 22 columns



```
In [56]: #Step 1. Grouping using df.groupby()
#Typically, you group the data using a categorical variable, such as customer segment.

#For example, in this case, we will group the data along Customer_Segment.
df_by_segment = master_df.groupby('Customer_Segment')
df_by_segment
print('\n')

#Step 2. Applying a Function
#After grouping, you apply a function to a numeric variable, such as mean(Sales),
# Step 2. Applying a function
# We can choose aggregate functions such as sum, mean, median, etc.
df_by_segment['Profit'].sum()
print('\n')
# Alternatively
df_by_segment.Profit.sum()
# For better readability, you may want to sort the summarised series:
df_by_segment.Profit.sum().sort_values(ascending = False)

#Step 3. Combining the results into a Data Structure
#You can optionally show the results as a dataframe.
# Converting to a df
print(pd.DataFrame(df_by_segment['Profit'].sum().sort_values(ascending = False)))
print('\n')
print(pd.DataFrame(df_by_segment['Profit'].mean().sort_values(ascending = False)))
print('\n')
print(pd.DataFrame(df_by_segment['Profit'].count().sort_values(ascending = False)))
print('\n')
print(pd.DataFrame(df_by_segment['Profit'].describe()))
```

	Profit
Customer_Segment	
CORPORATE	599746.00
HOME OFFICE	318354.03
SMALL BUSINESS	315708.01
CONSUMER	287959.94

	Profit
Customer_Segment	
CORPORATE	194.975943
SMALL BUSINESS	192.270408
CONSUMER	174.627010
HOME OFFICE	156.670290

	Profit
Customer_Segment	
CORPORATE	3076
HOME OFFICE	2032
CONSUMER	1649

SMALL BUSINESS 1642

	count	mean	std	min	25%	50%	\
Customer_Segment							
CONSUMER	1649.0	174.627010	1370.392654	-14140.70	-83.6300	0.260	
CORPORATE	3076.0	194.975943	1224.120222	-11861.46	-83.3075	-2.840	
HOME OFFICE	2032.0	156.670290	1054.269294	-12558.00	-88.7225	-0.430	
SMALL BUSINESS	1642.0	192.270408	1121.383478	-11984.40	-76.8825	-0.725	

	75%	max
Customer_Segment		
CONSUMER	154.6500	27220.69
CORPORATE	165.3350	14440.39
HOME OFFICE	156.6775	10521.33
SMALL BUSINESS	173.9525	13340.26

In [44]:

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0
5	8	6	aug	sun	92.3	85.3	488.0	14.7	22.2	29	5.4	0.0	0.0
6	8	6	aug	mon	92.3	88.9	495.6	8.5	24.1	27	3.1	0.0	0.0
7	8	6	aug	mon	91.5	145.4	608.2	10.7	8.0	86	2.2	0.0	0.0
8	8	6	sep	tue	91.0	129.5	692.6	7.0	13.1	63	5.4	0.0	0.0
9	7	5	sep	sat	92.5	88.0	698.6	7.1	22.8	40	4.0	0.0	0.0

```
In [69]: #Group the data 'df' by 'month' and 'day' and find the mean value for column 'rain'
import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKkLvq851k2b8JOckF')
print(df.head())
print('\n')
df_2 = df.groupby(['month', 'day'])
df_2

df_1 = pd.DataFrame(df_2['rain', 'wind'].mean())

print(df_1.head(20))
```

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0
1	7	4	oct	tue	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.0
2	7	4	oct	sat	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.0
3	8	6	mar	fri	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.0
4	8	6	mar	sun	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.0

			rain	wind
month	day			
apr	fri	0.000000	3.100000	
	mon	0.000000	3.100000	
	sat	0.000000	4.500000	
	sun	0.000000	5.666667	
	thu	0.000000	5.800000	
	wed	0.000000	2.700000	
aug	fri	0.066667	4.766667	
	mon	0.000000	2.873333	
	sat	0.000000	4.310345	
	sun	0.025000	4.417500	
	thu	0.000000	3.503846	
	tue	0.300000	4.567857	
	wed	0.000000	3.520000	
dec	fri	0.000000	4.900000	
	mon	0.000000	8.500000	
	sun	0.000000	8.500000	
	thu	0.000000	4.900000	
	tue	0.000000	8.500000	
	wed	0.000000	8.000000	
feb	fri	0.000000	4.820000	

```

In [75]: # Lamda and pivot
# Loading libraries and files
import numpy as np
import pandas as pd

market_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/global_
customer_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/glob
product_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/globa
shipping_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/glob
orders_df = pd.read_csv("C:/Users/Z001MC7/Downloads/Introduction_to_Pandas/global

# Merging the dataframes to create a master_df
df_1 = pd.merge(market_df, customer_df, how='inner', on='Cust_id')
df_2 = pd.merge(df_1, product_df, how='inner', on='Prod_id')
df_3 = pd.merge(df_2, shipping_df, how='inner', on='Ship_id')
master_df = pd.merge(df_3, orders_df, how='inner', on='Ord_id')

# Create a function to be applied
def is_positive(x):
    return x > 0

# Create a new column
master_df['is_profitable'] = master_df['Profit'].apply(is_positive)
master_df.head()

print('\n')

#or we can use a simpler way

# Create a new column using a lambda function
master_df['is_profitable'] = master_df['Profit'].apply(lambda x: x > 0)
master_df.head()

```

Out[75]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.51	
1	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.90	
2	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.02	0.03	23	-47.64	
3	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.05	0.04	27	23.12	
4	Ord_5484	Prod_16	SHP_7663	Cust_1820	322.82	0.05	35	-17.58	

5 rows × 23 columns



```
In [74]: # Comparing percentage of profitable orders across customer segments using the new
by_segment = master_df.groupby('Customer_Segment')
by_segment.is_profitable.mean()
```

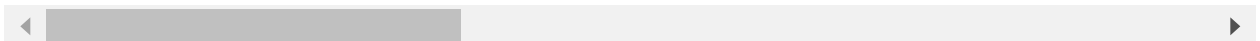
```
Out[74]: Customer_Segment
CONSUMER      0.500910
CORPORATE      0.481469
HOME OFFICE    0.498524
SMALL BUSINESS 0.496346
Name: is_profitable, dtype: float64
```

```
In [76]: # You can also use apply and lambda to alter existing columns
# E.g. you want to see Profit as one decimal place
# apply the round() function
master_df['Profit'] = master_df['Profit'].apply(lambda x: round(x, 1))
master_df.head()
```

```
Out[76]:
```

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_Cost
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.5	10.0
1	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.9	10.0
2	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.02	0.03	23	-47.6	10.0
3	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.05	0.04	27	23.1	10.0
4	Ord_5484	Prod_16	SHP_7663	Cust_1820	322.82	0.05	35	-17.6	10.0

5 rows × 23 columns

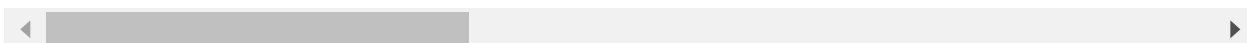


```
In [77]: # Creating a column Profit / Order_Quantity
master_df['profit_per_qty'] = master_df['Profit'] / master_df['Order_Quantity']
master_df.head()
```

Out[77]:

	Ord_id	Prod_id	Ship_id	Cust_id	Sales	Discount	Order_Quantity	Profit	Shipping_!
0	Ord_5446	Prod_16	SHP_7609	Cust_1818	136.81	0.01	23	-30.5	
1	Ord_5446	Prod_4	SHP_7610	Cust_1818	4701.69	0.00	26	1148.9	
2	Ord_5446	Prod_6	SHP_7608	Cust_1818	164.02	0.03	23	-47.6	
3	Ord_2978	Prod_16	SHP_4112	Cust_1088	305.05	0.04	27	23.1	
4	Ord_5484	Prod_16	SHP_7663	Cust_1820	322.82	0.05	35	-17.6	

5 rows × 24 columns



```
In [78]: #pivot table.
# E.g. Compare average Sales across customer segments
master_df.pivot_table(values = 'Sales', index = 'Customer_Segment', aggfunc = 'me
```

Out[78]:

	Sales
Customer_Segment	
CONSUMER	1857.859965
CORPORATE	1787.680389
HOME OFFICE	1754.312931
SMALL BUSINESS	1698.124841

```
In [79]: # E.g. compare total number of profitable orders across regions
# Note that since is_profitable is 1/0, we can directly compute the sum
master_df.pivot_table(values = 'is_profitable', index = 'Region', aggfunc = 'sum')
```

Out[79]:

	is_profitable
Region	
ATLANTIC	544.0
NORTHWEST TERRITORIES	194.0
NUNAVUT	38.0
ONTARIO	916.0
PRARIE	852.0
QUEBEC	360.0
WEST	969.0
YUKON	262.0

```
In [80]: # Grouping by both rows and columns
# Compare the total profit across product categories and customer segments
# Since there are two categorical variables, we use both rows (index) and columns
master_df.pivot_table(values = 'Profit',
                        index = 'Product_Category',
                        columns = 'Customer_Segment',
                        aggfunc = 'sum')
```

Out[80]:

Customer_Segment	CONSUMER	CORPORATE	HOME OFFICE	SMALL BUSINESS
Product_Category				
FURNITURE	42728.5	22008.3	23978.6	28717.5
OFFICE SUPPLIES	88532.4	203038.8	121145.6	105306.8
TECHNOLOGY	156700.1	374701.1	173230.6	181684.1

```
In [83]: #Group the data 'df' by 'month' and 'day' and find the mean value for column 'rain'
import numpy as np
import pandas as pd
df = pd.read_csv('https://query.data.world/s/vBDCsoHCytUSLKkLvq851k2b8JOckF')
df_1 = df.pivot_table(values = ['rain', 'wind'],
                        index = ['month', 'day'],
                        aggfunc = 'mean')
print(df_1.head(20))
```

		rain	wind
month	day		
apr	fri	0.000000	3.100000
	mon	0.000000	3.100000
	sat	0.000000	4.500000
	sun	0.000000	5.666667
	thu	0.000000	5.800000
	wed	0.000000	2.700000
aug	fri	0.066667	4.766667
	mon	0.000000	2.873333
	sat	0.000000	4.310345
	sun	0.025000	4.417500
	thu	0.000000	3.503846
	tue	0.300000	4.567857
	wed	0.000000	3.520000
dec	fri	0.000000	4.900000
	mon	0.000000	8.500000
	sun	0.000000	8.500000
	thu	0.000000	4.900000
	tue	0.000000	8.500000
	wed	0.000000	8.000000
feb	fri	0.000000	4.820000

```
In [ ]:
```