Operations

There are lots of operations with pandas that will be really useful to you, but don't fall into any distinct category. Let's show them here in this lecture:

In [2]:

```
import pandas as pd
df = pd.DataFrame({'col1':[1,2,3,4],'col2':[444,555,666,444],'col3':['abc','def
df
```

Out[2]:

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

In [4]:

```
df['col2'].unique()
```

Out[4]:

```
array([444, 555, 666])
```

In [6]:

```
df['col2'].nunique()
```

Out[6]:

```
3
```

In [7]:

```
df['col2'].value_counts()
```

Out[7]:

```
444    2
555    1
666    1
Name: col2, dtype: int64
```

Info on Unique Values

In [53]:

```
df['col2'].unique()
```

Out[53]:

```
array([444, 555, 666])
```

In [54]:

```
df['col2'].nunique()
```

Out[54]:

```
3
```

In [8]:

```
df['col2'].value_counts()
```

Out[8]:

```
444    2
555    1
666    1
Name: col2, dtype: int64
```

In [10]:

```
df['col1']
```

Out[10]:

```
0    1
1    2
2    3
3    4
Name: col1, dtype: int64
```

In [13]:

```
df[df['col1']>2]
```

Out[13]:

	col1	col2	col3
2	3	666	ghi
3	4	444	xyz

In [15]:

```
df[df['col2'] == 444]
```

Out[15]:

	col1	col2	col3
0	1	444	abc
3	4	444	xyz

Selecting Data

In [16]:

```
#Select from DataFrame using criteria from multiple columns
newdf = df[(df['col1']>2) & (df['col2']==444)]
```

In [18]:

```
newdf
```

Out[18]:

	col1	col2	col3
3	4	444	xyz

In [24]:

```
df
```

Out[24]:

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

In [22]:

```
def multi(x):
    return x * 2
```

In [23]:

```
df['col2'].apply(multi)
```

Out[23]:

```
0     888
1    1110
2    1332
3     888
Name: col2, dtype: int64
```

In [25]:

```
def upper_case(x):
    return x.upper()
```

In [31]:

```
df['col3']= df['col3'].apply(upper_case)
```

In [33]:

```
df
```

Out[33]:

	col1	col2	col3
0	1	444	ABC
1	2	555	DEF
2	3	666	GHI
3	4	444	XYZ

In [34]:

```
df['col3'].apply(len)
```

Out[34]:

```
0    3
1    3
2    3
3    3
Name: col3, dtype: int64
```

Applying Functions

In [58]:

```
def times2(x):
    return x*2
```

In [59]:

```
df['col1'].apply(times2)
```

Out[59]:

```
0    2
1    4
2    6
3    8
Name: col1, dtype: int64
```

In [60]:

```
df['col3'].apply(len)
```

Out[60]:

```
0    3
1    3
2    3
3    3
Name: col3, dtype: int64
```

In [61]:

```
df['col1'].sum()
```

Out[61]:

```
10
```

** Permanently Removing a Column**

In [35]:

```
del df['col1']
```

In [36]:

```
df
```

Out[36]:

	col2	col3
0	444	ABC
1	555	DEF
2	666	GHI
3	444	XYZ

** Get column and index names: **

In [37]:

```
df.columns
```

Out[37]:

```
Index(['col2', 'col3'], dtype='object')
```

In [38]:

```
df.index
```

Out[38]:

```
RangeIndex(start=0, stop=4, step=1)
```

** Sorting and Ordering a DataFrame:**

In [40]:

```
df
```

Out[40]:

	col2	col3
0	444	ABC
1	555	DEF
2	666	GHI
3	444	XYZ

In [41]:

```
df.sort_values(by='col2') #inplace=False by default
```

Out[41]:

	col2	col3
0	444	ABC
3	444	XYZ
1	555	DEF
2	666	GHI

** Find Null Values or Check for Null Values**

In [42]:

```
df.isnull()
```

Out[42]:

	col2	col3
0	False	False
1	False	False
2	False	False
3	False	False

In [43]:

```
# Drop rows with NaN Values
df.dropna()
```

Out[43]:

	col2	col3
0	444	ABC
1	555	DEF
2	666	GHI
3	444	XYZ

** Filling in NaN values with something else: **

In [44]:

```
import numpy as np
```

In [45]:

```
df = pd.DataFrame({'col1':[1,2,3,np.nan],
                   'col2':[np.nan,555,666,444],
                   'col3':['abc','def','ghi','xyz']})
df
```

Out[45]:

	col1	col2	col3
0	1.0	NaN	abc
1	2.0	555.0	def
2	3.0	666.0	ghi
3	NaN	444.0	xyz

In [46]:

```
df.fillna('FILL')
```

Out[46]:

	col1	col2	col3
0	1	FILL	abc
1	2	555	def
2	3	666	ghi
3	FILL	444	xyz

In [47]:

```
data = {'A':['foo','foo','foo','bar','bar','bar'],
     'B':['one','one','two','two','one','one'],
       'C':['x','y','x','y','x','y'],
       'D':[1,3,2,5,4,1]}

df = pd.DataFrame(data)
```

In [48]:

```
df
```

Out[48]:

	A	B	C	D
0	foo	one	x	1
1	foo	one	y	3
2	foo	two	x	2
3	bar	two	y	5
4	bar	one	x	4
5	bar	one	y	1

In [91]:

```
# df.pivot_table(values='D',index=['A', 'B'],columns=['C'])
```

Out[91]:

	C	**x**	**y**
A	**B**		
bar	**one**	4.0	1.0
	two	NaN	5.0
foo	**one**	1.0	3.0
	two	2.0	NaN

Great Job!