Excel vs Python

1) Importing Our Data

Excel:

	A	B	C	D	E	F	G
1	Name	Dept	Start Date	End Date	Sales March	Sales April	Sales May
2	Arun	A	3/1/2020		20000	21000	32000
3	Kumar	A	5/10/2020		45000	48000	49000
4	Suchit	A	3/22/2020	4/25/2020		8000	10000
5	Sudhanshu	B	4/25/2020		28000	29000	25000
6	Krish	B	4/1/2020		32000	26000	31000
7	Vivek	C	5/3/2020		2000	25000	29000
8	Leo	C	3/30/2020	5/15/2020			
9	JK	C	4/25/2020		41000	26000	30000
10	Nirmala	C	5/15/2020		15000	35000	32000
11							
12							
13							

Python:

Import pandas as pd

sales = pd.read_excel('sales.xlsx')

sales

```
In [1]: import pandas as pd
```

```
In [8]: sales = pd.read_excel('D:\Data Science Course details/Sales.xlsx')
        sales
```

Out[8]:

	Name	Dept	Start Date	End Date	Sales March	Sales April	Sales May
0	Arun	A	2020-03-01	NaT	20000.0	21000.0	32000.0
1	Kumar	A	2020-05-10	NaT	45000.0	48000.0	49000.0
2	Suchit	A	2020-03-22	2020-04-25	NaN	8000.0	10000.0
3	Sudhanshu	B	2020-04-25	NaT	28000.0	29000.0	25000.0
4	Krish	B	2020-04-01	NaT	32000.0	26000.0	31000.0
5	Vivek	C	2020-05-03	NaT	2000.0	25000.0	29000.0
6	Leo	C	2020-03-30	2020-05-15	NaN	NaN	NaN
7	JK	C	2020-04-25	NaT	41000.0	26000.0	30000.0
8	Nirmala	C	2020-05-15	NaT	15000.0	35000.0	32000.0

We can notice a few differences between how pandas represent the data vs what we saw in Excel:

- In pandas, **the row numbers start at 0 versus 1 in Excel**.
- The column names in pandas are taken from the data, versus Excel where columns are labelled using letters.
- Where there is a missing value in the original data, pandas has the placeholder **NaN** which indicates that the value is missing, or **null**.
- The sales data has a decimal point added to each value, because pandas stores numeric values that include null (NaN) values as numeric type known as **float** (this doesn't effect anything for us, but we just wanted to explain why this is).

Let's use the type() function to look at the type of our sales variable

Python:

type(sales)

pandas.core.frame.DataFrame

```
In [1]: import pandas as pd
```

```
In [8]: sales = pd.read_excel('D:\Data Science Course details/Sales.xlsx')
        sales
```

Out[8]:

	Name	Dept	Start Date	End Date	Sales March	Sales April	Sales May
0	Arun	A	2020-03-01	NaT	20000.0	21000.0	32000.0
1	Kumar	A	2020-05-10	NaT	45000.0	48000.0	49000.0
2	Suchit	A	2020-03-22	2020-04-25	NaN	8000.0	10000.0
3	Sudhanshu	B	2020-04-25	NaT	28000.0	29000.0	25000.0
4	Krish	B	2020-04-01	NaT	32000.0	26000.0	31000.0
5	Vivek	C	2020-05-03	NaT	2000.0	25000.0	29000.0
6	Leo	C	2020-03-30	2020-05-15	NaN	NaN	NaN
7	JK	C	2020-04-25	NaT	41000.0	26000.0	30000.0
8	Nirmala	C	2020-05-15	NaT	15000.0	35000.0	32000.0

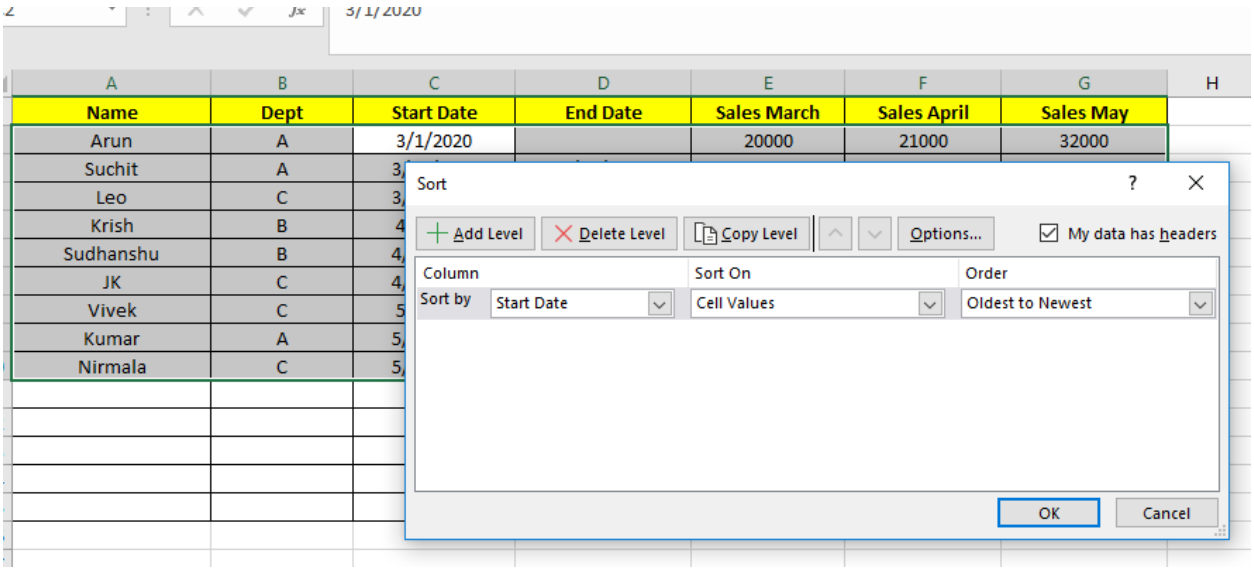
```
In [9]: type(sales)
Out[9]: pandas.core.frame.DataFrame
```

This output tells us that our sales variable is a **DataFrame** object, which is a specific type of object in pandas. Most of the time in pandas when we want to modify a dataframe, we'll use special syntax called a data frame **method**, which allows us to access specific functionality that relates to the dataframe objects. We'll see an example of that in a moment when we complete our first task in pandas

2) Sorting Data

How to sort our data in Excel and Python
In Excel, if we wanted to sort our data by the "Start Date" column, we would:

- Select our data.
- Click the 'Sort' button on the toolbar.
- Select 'Start Date' in the dialog box that opens.



In Pandas, we use the DataFrame.sort_values() method.
sales = sales.sort_values("Start Date")

```
In [10]: sales = sales.sort_values("Start Date")
         sales
```

Out[10]:

	Name	Dept	Start Date	End Date	Sales March	Sales April	Sales May
0	Arun	A	2020-03-01	NaT	20000.0	21000.0	32000.0
2	Suchit	A	2020-03-22	2020-04-25	NaN	8000.0	10000.0
6	Leo	C	2020-03-30	2020-05-15	NaN	NaN	NaN
4	Krish	B	2020-04-01	NaT	32000.0	26000.0	31000.0
3	Sudhanshu	B	2020-04-25	NaT	28000.0	29000.0	25000.0
7	JK	C	2020-04-25	NaT	41000.0	26000.0	30000.0
5	Vivek	C	2020-05-03	NaT	2000.0	25000.0	29000.0
1	Kumar	A	2020-05-10	NaT	45000.0	48000.0	49000.0
8	Nirmala	C	2020-05-15	NaT	15000.0	35000.0	32000.0

Summing the Sales Values

Excel

- Enter a new column name "Sales Q1" in cell H1.
- In cell H2, use the SUM() formula and specify the range of cells using their coordinates.
- Drag the formula down to all rows.

A	B	C	D	E	F	G	H
Name	**Dept**	**Start Date**	**End Date**	**Sales March**	**Sales April**	**Sales May**	**Sales Q1**
Arun	A	3/1/2020		20000	21000	32000	=sum(E2:G2)
Suchit	A	3/22/2020	4/25/2020		8000	10000	
Leo	C	3/30/2020	5/15/2020				
Krish	B	4/1/2020		32000	26000	31000	
Sudhanshu	B	4/25/2020		28000	29000	25000	
JK	C	4/25/2020		41000	26000	30000	
Vivek	C	5/3/2020		2000	25000	29000	
Kumar	A	5/10/2020		45000	48000	49000	
Nirmala	C	5/15/2020		15000	35000	32000	

Python:

q1 = sales[["Sales March","Sales April","Sales May"]]

```
In [11]: q1 = sales[["Sales March","Sales April","Sales May"]]
         q1
```

Out[11]:

	Sales March	Sales April	Sales May
0	20000.0	21000.0	32000.0
2	NaN	8000.0	10000.0
6	NaN	NaN	NaN
4	32000.0	26000.0	31000.0
3	28000.0	29000.0	25000.0
7	41000.0	26000.0	30000.0
5	2000.0	25000.0	29000.0
1	45000.0	48000.0	49000.0
8	15000.0	35000.0	32000.0

We'll use the DataFrame.sum() method and specify axis=1, which tells pandas that we want to sum the rows and not the columns.

sales['Sales Q1'] = q1.sum(axis=1)

```
In [12]: sales['Sales Q1'] =  q1.sum(axis=1)
         sales
```

Out[12]:

	Name	Dept	Start Date	End Date	Sales March	Sales April	Sales May	Sales Q1
0	Arun	A	2020-03-01	NaT	20000.0	21000.0	32000.0	73000.0
2	Suchit	A	2020-03-22	2020-04-25	NaN	8000.0	10000.0	18000.0
6	Leo	C	2020-03-30	2020-05-15	NaN	NaN	NaN	0.0
4	Krish	B	2020-04-01	NaT	32000.0	26000.0	31000.0	89000.0
3	Sudhanshu	B	2020-04-25	NaT	28000.0	29000.0	25000.0	82000.0
7	JK	C	2020-04-25	NaT	41000.0	26000.0	30000.0	97000.0
5	Vivek	C	2020-05-03	NaT	2000.0	25000.0	29000.0	56000.0
1	Kumar	A	2020-05-10	NaT	45000.0	48000.0	49000.0	142000.0
8	Nirmala	C	2020-05-15	NaT	15000.0	35000.0	32000.0	82000.0

3) Joining Manager Data

How to join this data in a "Manager" column in Excel and Python

Excel:

	A	B
1	Dept	Manager
2	A	Richard
3	B	Chris
4	C	Lisa
5		

- start by adding the column name to cell I1.
- use the =**VLOOKUP(B2,Sheet2!A$2:B$4,2)**formula in cell I2, specifying:
    - to lookup the value from cell B2 (the Department)
    - in the selection of manager data, which we specify using coordinates
    - and that we want to select the value from the second column of that data.
- Click and drag the formula down to all cells.

=VLOOKUP(B2,Sheet2!A$2:B$4,2)

Name	Dept	Start Date	End Date	Sales March	Sales April	Sales May	Sales Q1	Manager
Arun	A	3/1/2020		20000	21000	32000	73000	Richard
Suchit	A	3/22/2020	4/25/2020		8000	10000	18000	Richard
Leo	C	3/30/2020	5/15/2020				0	Lisa
Krish	B	4/1/2020		32000	26000	31000	89000	Chris
Sudhanshu	B	4/25/2020		28000	29000	25000	82000	Chris
JK	C	4/25/2020		41000	26000	30000	97000	Lisa
Vivek	C	5/3/2020		2000	25000	29000	56000	Lisa
Kumar	A	5/10/2020		45000	48000	49000	142000	Richard
Nirmala	C	5/15/2020		15000	35000	32000	82000	Lisa

```
In [15]: managers = pd.read_excel("D:\Data Science Course details/managers.xlsx")
         managers
```

Out[15]:

	Dept	Manager
0	A	Richard
1	B	Chris
2	C	Lisa

In order to join mangers data to sales using pandas, we'll use the pandas.merge() function.

Pandas:

sales = pd.merge(sales, managers, how ='left', on='Dept')

```
In [16]: sales = pd.merge(sales, managers, how ='left', on='Dept')
         sales
```

Out[16]:

	Name	Dept	Start Date	End Date	Sales March	Sales April	Sales May	Sales Q1	Manager
0	Arun	A	2020-03-01	NaT	20000.0	21000.0	32000.0	73000.0	Richard
1	Suchit	A	2020-03-22	2020-04-25	NaN	8000.0	10000.0	18000.0	Richard
2	Leo	C	2020-03-30	2020-05-15	NaN	NaN	NaN	0.0	Lisa
3	Krish	B	2020-04-01	NaT	32000.0	26000.0	31000.0	89000.0	Chris
4	Sudhanshu	B	2020-04-25	NaT	28000.0	29000.0	25000.0	82000.0	Chris
5	JK	C	2020-04-25	NaT	41000.0	26000.0	30000.0	97000.0	Lisa
6	Vivek	C	2020-05-03	NaT	2000.0	25000.0	29000.0	56000.0	Lisa
7	Kumar	A	2020-05-10	NaT	45000.0	48000.0	49000.0	142000.0	Richard
8	Nirmala	C	2020-05-15	NaT	15000.0	35000.0	32000.0	82000.0	Lisa

4) Adding a Conditional Column

Excel

- Add a new column name to cell J1.
- Use the IF() formula to check if cell D1 (End Date) is empty, and if so fill J2 with TRUE, otherwise FALSE.

J2 =IF(D2="",TRUE,FALSE)

	A	B	C	D	E	F	G	H	I	J
1	Name	Dept	Start Date	End Date	ales Mar	Sales April	Sales May	Sales Q1	Manager	Current Employee
2	Arun	A	3/1/2020		20000	21000	32000	73000	Richard	TRUE
3	Suchit	A	3/22/2020	4/25/2020		8000	10000	18000	Richard	FALSE
4	Leo	C	3/30/2020	5/15/2020				0	Lisa	FALSE
5	Krish	B	4/1/2020		32000	26000	31000	89000	Chris	TRUE
6	Sudhanshu	B	4/25/2020		28000	29000	25000	82000	Chris	TRUE
7	JK	C	4/25/2020		41000	26000	30000	97000	Lisa	TRUE
8	Vivek	C	5/3/2020		2000	25000	29000	56000	Lisa	TRUE
9	Kumar	A	5/10/2020		45000	48000	49000	142000	Richard	TRUE
10	Nirmala	C	5/15/2020		15000	35000	32000	82000	Lisa	TRUE
11										

Pandas:

sales['Current Employee'] = pd.isnull(sales['End Date'])

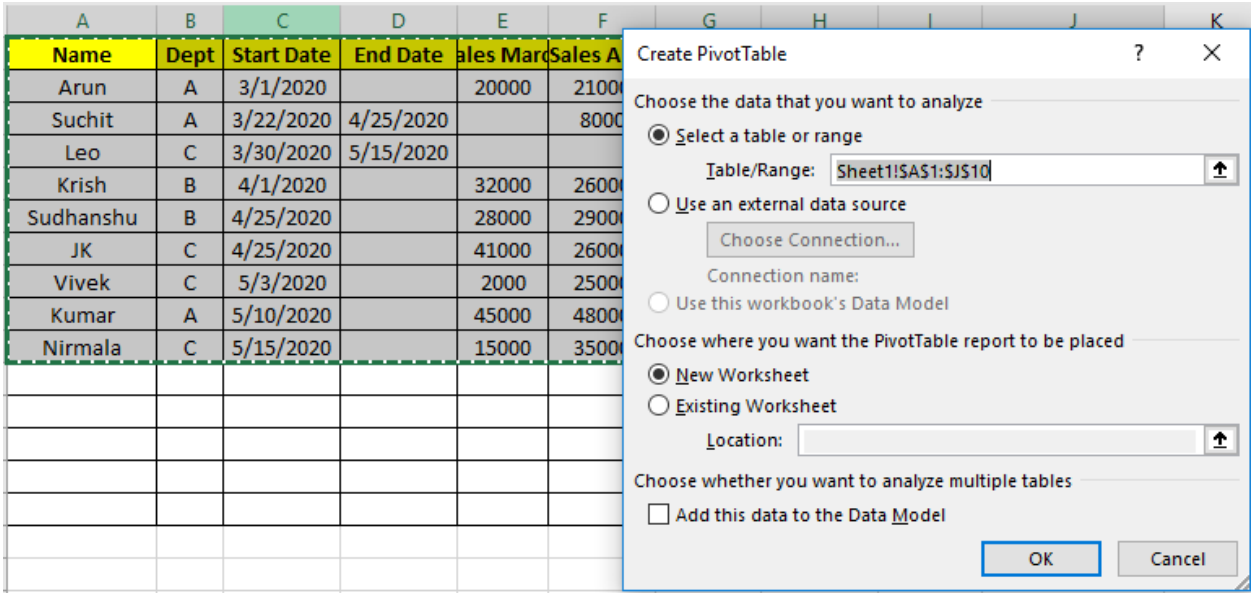
```
In [17]: sales['Current Employee'] = pd.isnull(sales['End Date'])
         sales
```

Out[17]:

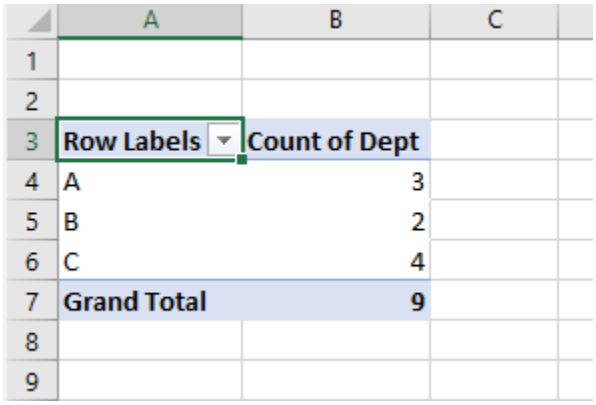
	Name	Dept	Start Date	End Date	Sales March	Sales April	Sales May	Sales Q1	Manager	Current Employee
0	Arun	A	2020-03-01	NaT	20000.0	21000.0	32000.0	73000.0	Richard	True
1	Suchit	A	2020-03-22	2020-04-25	NaN	8000.0	10000.0	18000.0	Richard	False
2	Leo	C	2020-03-30	2020-05-15	NaN	NaN	NaN	0.0	Lisa	False
3	Krish	B	2020-04-01	NaT	32000.0	26000.0	31000.0	89000.0	Chris	True
4	Sudhanshu	B	2020-04-25	NaT	28000.0	29000.0	25000.0	82000.0	Chris	True
5	JK	C	2020-04-25	NaT	41000.0	26000.0	30000.0	97000.0	Lisa	True
6	Vivek	C	2020-05-03	NaT	2000.0	25000.0	29000.0	56000.0	Lisa	True
7	Kumar	A	2020-05-10	NaT	45000.0	48000.0	49000.0	142000.0	Richard	True
8	Nirmala	C	2020-05-15	NaT	15000.0	35000.0	32000.0	82000.0	Lisa	True

5) Pivot Tables

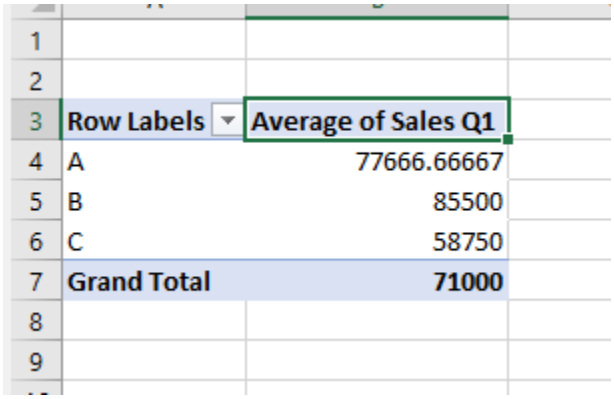
Excel:



Name	Dept	Start Date	End Date	ales Marc	Sales A
Arun	A	3/1/2020		20000	2100
Suchit	A	3/22/2020	4/25/2020		8000
Leo	C	3/30/2020	5/15/2020		
Krish	B	4/1/2020		32000	2600
Sudhanshu	B	4/25/2020		28000	2900
JK	C	4/25/2020		41000	2600
Vivek	C	5/3/2020		2000	2500
Kumar	A	5/10/2020		45000	4800
Nirmala	C	5/15/2020		15000	3500



Row Labels	Count of Dept
A	3
B	2
C	4
Grand Total	9



Row Labels	Average of Sales Q1
A	77666.66667
B	85500
C	58750
Grand Total	71000

Pandas:

sales['Dept'].value_counts()

```
In [18]: sales['Dept'].value_counts()

Out[18]: C    4
         A    3
         B    2
         Name: Dept, dtype: int64
```

to calculate this in pandas, we'll use the DataFrame.pivot_table() method. We need to specify some arguments:

- index: the column to aggregate by.
- values: the column we want to use the values for.
- aggfunc: the aggregation function we want to use, in this case 'mean' average.

sales.pivot_table(index='Dept', values='Sales Q1', aggfunc='mean')

```
In [20]: sales.pivot_table(index='Dept', values='Sales Q1', aggfunc='mean')

Out[20]:
```

Dept	Sales Q1
A	77666.666667
B	85500.000000
C	58750.000000