Python Libraries - Pandas - Describing Data

You will be working with files of varied shape and sizes in the Pandas. One you have loaded the data in the dataframes, it is necessary that you check the created dataframe. However, it would be inefficieent to print the entire dataframe every time. Hence, you should learn how to print limited number of rows in a dataframe.

Project title:Analysis and Visualization of "sales.xlsx"files using python computing library called as pandas

Problem Statement is an organization of sales data related to the E-commerce of Bussiness Analysis.The organization want complete information regarding their sales in market,region,No of Orders,Profit.

As a DataScience Engineer my task is to provide realtime error free solution

Task1:Visualization and analysis of sales depending on mention parameters

Task2:Find out what are the various Region and Market sales is grow

Organization Name:MintrasFashionpvt.lmtd

```
# import the required libraries
import pandas as pd


# Read data from the file 'sales.xlsx'
sales=pd.read_excel("sales.xlsx")

# Check the created dataframe
sales
```

	Market	Region	No_of_Orders	Profit	Sales
0	Africa	Western Africa	251	-12901.51	78476.06
1	Africa	Southern Africa	85	11768.58	51319.50

```
# Read the file with 'Region' as the index column
sales =pd.read_excel("sales.xlsx", index_col=1)

# Check the created dataframe
sales
```

Region	Market	No_of_Orders	Profit	Sales
Western Africa	Africa	251	-12901.51	78476.06
Southern Africa	Africa	85	11768.58	51319.50
North Africa	Africa	182	21643.08	86698.89
Eastern Africa	Africa	110	8013.04	44182.60
Central Africa	Africa	103	15606.30	61689.99
Western Asia	Asia Pacific	382	-16766.90	124312.24
Southern Asia	Asia Pacific	469	67998.76	351806.60
Southeastern Asia	Asia Pacific	533	20948.84	329751.38
Oceania	Asia Pacific	646	54734.02	408002.98
Eastern Asia	Asia Pacific	414	72805.10	315390.77
Central Asia	Asia Pacific	37	-2649.76	8190.74
Western Europe	Europe	964	82091.27	656637.14
Southern Europe	Europe	338	18911.49	215703.93
Northern Europe	Europe	367	43237.44	252969.09
Eastern Europe	Europe	241	25050.69	108258.93
South America	LATAM	496	12377.59	210710.49
Central America	LATAM	930	74679.54	461670.28
Caribbean	LATAM	288	13529.59	116333.05
Western US	USCA	490	44303.65	251991.83
Southern US	USCA	255	19991.83	148771.91
Eastern US	USCA	443	47462.04	264973.98
Central US	USCA	356	33697.43	170416.31
Canada	USCA	49	7246.62	26298.81

```
# Printing first 5 entries from a dataframe
sales =pd.read_excel("sales.xlsx", index_col=1).head()
sales
```

Region	Market	No_of_Orders	Profit	Sales
Western Africa	Africa	251	-12901.51	78476.06
Southern Africa	Africa	85	11768.58	51319.50
North Africa	Africa	182	21643.08	86698.89
Eastern Africa	Africa	110	8013.04	44182.60
Central Africa	Africa	103	15606.30	61689.99

```
# Printing first 8 entries of a dataframe
sales =pd.read_excel("sales.xlsx", index_col=1).head(8)
sales
```

Region	Market	No_of_Orders	Profit	Sales
Western Africa	Africa	251	-12901.51	78476.06
Southern Africa	Africa	85	11768.58	51319.50
North Africa	Africa	182	21643.08	86698.89
Eastern Africa	Africa	110	8013.04	44182.60
Central Africa	Africa	103	15606.30	61689.99

```
# Printing last 5 entries of the dataframe
sales =pd.read_excel("sales.xlsx", index_col=1).tail(5)
sales
```

Region	Market	No_of_Orders	Profit	Sales
Western US	USCA	490	44303.65	251991.83
Southern US	USCA	255	19991.83	148771.91
Eastern US	USCA	443	47462.04	264973.98
Central US	USCA	356	33697.43	170416.31
Canada	USCA	49	7246.62	26298.81

```
# Printing last 3 entries of the dataframe
sales =pd.read_excel("sales.xlsx", index_col=1).tail(3)
sales
```

Region	Market	No_of_Orders	Profit	Sales
Eastern US	USCA	443	47462.04	264973.98
Central US	USCA	356	33697.43	170416.31
Canada	USCA	49	7246.62	26298.81

Summarising the dataframes

A dataframe can have multiple columns and it is very important to understand what each column stores. You must be familiar with the column names, the data it stores, data type of each column, etc. Let's see different commands that will help you to do that.

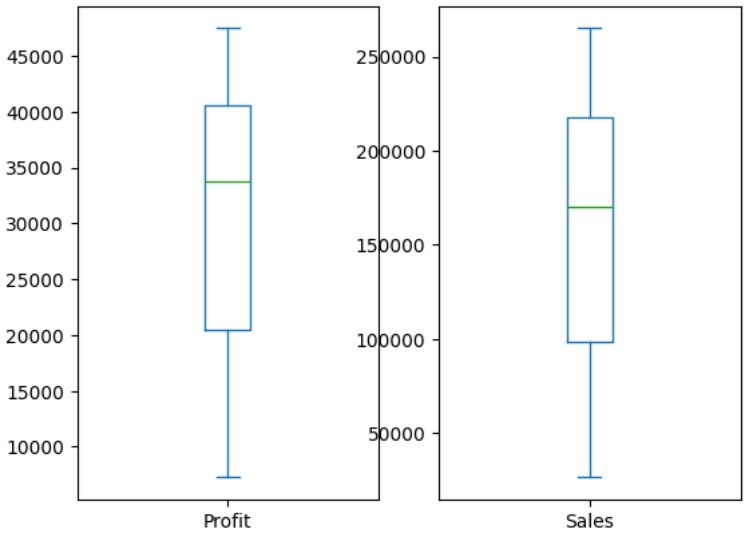
```
# Summarising the dataframe structure
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, Eastern US to Canada
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Market        3 non-null      object
 1   No_of_Orders  3 non-null      int64
 2   Profit        3 non-null      float64
 3   Sales         3 non-null      float64
dtypes: float64(2), int64(1), object(1)
memory usage: 228.0+ bytes
```

```
# Summary of data stored in each column
sales.describe()
```

	No_of_Orders	Profit	Sales
count	3.000000	3.000000	3.000000
mean	282.666667	29468.696667	153896.366667

```
# Graphically summarising the spread of the columns - Profit and Sales
import matplotlib.pyplot as plt
sales[["Profit","Sales"]].plot(kind ="box",subplots=True)
plt.show()
```



Double-click (or enter) to edit

Python Libraries - Pandas - Indexing and Slicing

In this section, you will:

- Select rows from a dataframe
- Select columns from a dataframe
- Select subsets of dataframes

Selecting Rows

Selecting rows in dataframes is similar to the indexing you have seen in numpy arrays. The syntax `df[start_index:end_index]` will subset rows according to the start and end indices.

```
# Read data from the file 'sales.xlsx'
sales =pd.read_excel("sales.xlsx").head()

# Check the created dataframe
# Remember - you should print limited number of entries to check the dataframe


# Selecting first 5 rows of the dataframe
sales[0:5]
```

	Market	Region	No_of_Orders	Profit	Sales
0	Africa	Western Africa	251	-12901.51	78476.06
1	Africa	Southern Africa	85	11768.58	51319.50
2	Africa	North Africa	182	21643.08	86698.89
3	Africa	Eastern Africa	110	8013.04	44182.60
4	Africa	Central Africa	103	15606.30	61689.99

```
# Selecting all the even indices of the dataframe
sales[0::2]
```

	Market	Region	No_of_Orders	Profit	Sales
0	Africa	Western Africa	251	-12901.51	78476.06
2	Africa	North Africa	182	21643.08	86698.89

Selecting Columns

There are two simple ways to select a single column from a dataframe:

- `df['column']` or `df.column` return a series
- `df[['col_x', 'col_y']]` returns a dataframe

```
# Select the column 'Profit' from the dataframe 'Sales'. Output must be in the form of a dataframe.
sales[["Profit"]]
```

	Profit
0	-12901.51
1	11768.58
2	21643.08
3	8013.04
4	15606.30

```
# Check the type of the sliced data
type(sales[["Profit"]])
```

```
pandas.core.frame.DataFrame
```

```
# Select the column 'Profit' from the dataframe 'Sales'. Output must be in the form of a series.
sales["Profit"]
```

```
0   -12901.51
1    11768.58
2    21643.08
3     8013.04
4    15606.30
Name: Profit, dtype: float64
```

```
# Check the type of the sliced data
type(sales["Profit"])
```

```
pandas.core.series.Series
```

Selecting Multiple Columns

You can select multiple columns by passing the list of column names inside the `[]`: `df[['column_1', 'column_2', 'column_n']]`.

```
# Selecting multiple columns from a dataframe
sales[["Profit","Region"]]
```

	Profit	Region
0	-12901.51	Western Africa
1	11768.58	Southern Africa
2	21643.08	North Africa
3	8013.04	Eastern Africa
4	15606.30	Central Africa

Label and Position Based Indexing: `df.loc` and `df.iloc`

You have seen some ways of selecting rows and columns from dataframes. Let's now see some other ways of indexing dataframes, which pandas recommends, since they are more explicit (and less ambiguous).

There are two main ways of indexing dataframes:

1. Label based indexing using `df.loc`
2. Position based indexing using `df.iloc`

Using both the methods, we will do the following indexing operations on a dataframe:

- Selecting single elements/cells
- Selecting single and multiple rows
- Selecting single and multiple columns
- Selecting multiple rows and columns

Label-based Indexing

```
# Select the row with index label as 'Canada'
sales.iloc[0:5,2:]
```

	No_of_Orders	Profit	Sales
0	251	-12901.51	78476.06
1	85	11768.58	51319.50
2	182	21643.08	86698.89
3	110	8013.04	44182.60
4	103	15606.30	61689.99

```
# Select the row with index label as 'Canada' and 'Western Africa'
```

```
# Select the row with index label as 'Canada' and 'Western Africa' along with the columns 'Profit' and 'Sales'
```

Position-based Indexing

```
# Select the top 5 rows and all the columns starting from second column
sales.iloc[0:5,2:]
```

	No_of_Orders	Profit	Sales
0	251	-12901.51	78476.06
1	85	11768.58	51319.50
2	182	21643.08	86698.89
3	110	8013.04	44182.60
4	103	15606.30	61689.99

```
# Select all the entries with positive profit
sales[["Profit"]]>0
```

	Profit
0	False
1	True
2	True
3	True
4	True

```
# Count the number of entries in the dataframe with positive profit
sales[sales[["Profit"]]>0]
```

	Market	Region	No_of_Orders	Profit	Sales
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	11768.58	NaN
2	NaN	NaN	NaN	21643.08	NaN
3	NaN	NaN	NaN	8013.04	NaN
4	NaN	NaN	NaN	15606.30	NaN

```
# Select all the enries in Latin America and European market where Sales>250000
sales[sales[["Profit"]]>0]
```

	Market	Region	No_of_Orders	Profit	Sales
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	11768.58	NaN
2	NaN	NaN	NaN	21643.08	NaN
3	NaN	NaN	NaN	8013.04	NaN
4	NaN	NaN	NaN	15606.30	NaN

Double-click (or enter) to edit

```
sales[sales[["Profit"]]>0].size
```

```
25
```

```
sales[sales[["Profit"]]>0].count()
sales[sales[["Profit"]]>0].size
sales.count()
```

```
Market          5
Region          5
No_of_Orders    5
Profit          5
Sales           5
dtype: int64
```

```
sales=pd.read_excel("sales.xlsx")
sales[(sales["Sales"]>250000)&(sales["Market"].isin(["LATIN","Europe"]))]
```

	Market	Region	No_of_Orders	Profit	Sales
11	Europe	Western Europe	964	82091.27	656637.14
13	Europe	Northern Europe	367	43237.44	252969.09

Conclusion:According to the my conclusion Europe,USCA gives highest sales,these are the two markets which give highest sales.

Python Libraries - Pandas - Describing Data

You will be working with files of varied shape and sizes in the Pandas. One you have loaded the data in the dataframes, it is necessary that you check the created dataframe. However, it would be inefficieent to print the entire dataframe every time. Hence, you should learn how to print limited number of rows in a dataframe.

```
# import the required libraries
import pandas as pd


# Read data from the file 'sales.xlsx'
sales=pd.read_excel("sales.xlsx")

# Check the created dataframe
sales
```

	Market	Region	No_of_Orders	Profit	Sales
0	Africa	Western Africa	251	-12901.51	78476.06
1	Africa	Southern Africa	85	11768.58	51319.50
2	Africa	North Africa	182	21643.08	86698.89
3	Africa	Eastern Africa	110	8013.04	44182.60
4	Africa	Central Africa	103	15606.30	61689.99
5	Asia Pacific	Western Asia	382	-16766.90	124312.24
6	Asia Pacific	Southern Asia	469	67998.76	351806.60
7	Asia Pacific	Southeastern Asia	533	20948.84	329751.38
8	Asia Pacific	Oceania	646	54734.02	408002.98
9	Asia Pacific	Eastern Asia	414	72805.10	315390.77
10	Asia Pacific	Central Asia	37	-2649.76	8190.74
11	Europe	Western Europe	964	82091.27	656637.14
12	Europe	Southern Europe	338	18911.49	215703.93
13	Europe	Northern Europe	367	43237.44	252969.09
14	Europe	Eastern Europe	241	25050.69	108258.93
15	LATAM	South America	496	12377.59	210710.49
16	LATAM	Central America	930	74679.54	461670.28
17	LATAM	Caribbean	288	13529.59	116333.05
18	USCA	Western US	490	44303.65	251991.83
19	USCA	Southern US	255	19991.83	148771.91
20	USCA	Eastern US	443	47462.04	264973.98
21	USCA	Central US	356	33697.43	170416.31
22	USCA	Canada	49	7246.62	26298.81

```
# Read the file with 'Region' as the index column
sales =pd.read_excel("sales.xlsx", index_col=1)

# Check the created dataframe
sales
```

Region	Market	No_of_Orders	Profit	Sales
Western Africa	Africa	251	-12901.51	78476.06
Southern Africa	Africa	85	11768.58	51319.50
North Africa	Africa	182	21643.08	86698.89
Eastern Africa	Africa	110	8013.04	44182.60
Central Africa	Africa	103	15606.30	61689.99
Western Asia	Asia Pacific	382	-16766.90	124312.24
Southern Asia	Asia Pacific	469	67998.76	351806.60
Southeastern Asia	Asia Pacific	533	20948.84	329751.38
Oceania	Asia Pacific	646	54734.02	408002.98
Eastern Asia	Asia Pacific	414	72805.10	315390.77
Central Asia	Asia Pacific	37	-2649.76	8190.74
Western Europe	Europe	964	82091.27	656637.14
Southern Europe	Europe	338	18911.49	215703.93
Northern Europe	Europe	367	43237.44	252969.09
Eastern Europe	Europe	241	25050.69	108258.93

```
# Printing first 5 entries from a dataframe
sales =pd.read_excel("sales.xlsx", index_col=1).head()
sales
```

Region	Market	No_of_Orders	Profit	Sales
Western Africa	Africa	251	-12901.51	78476.06
Southern Africa	Africa	85	11768.58	51319.50
North Africa	Africa	182	21643.08	86698.89
Eastern Africa	Africa	110	8013.04	44182.60
Central Africa	Africa	103	15606.30	61689.99

```
# Printing first 8 entries of a dataframe
sales =pd.read_excel("sales.xlsx", index_col=1).head(8)
sales
```

Region	Market	No_of_Orders	Profit	Sales
Western Africa	Africa	251	-12901.51	78476.06
Southern Africa	Africa	85	11768.58	51319.50
North Africa	Africa	182	21643.08	86698.89
Eastern Africa	Africa	110	8013.04	44182.60
Central Africa	Africa	103	15606.30	61689.99
Western Asia	Asia Pacific	382	-16766.90	124312.24
Southern Asia	Asia Pacific	469	67998.76	351806.60
Southeastern Asia	Asia Pacific	533	20948.84	329751.38

```
# Printing last 5 entries of the dataframe
sales =pd.read_excel("sales.xlsx", index_col=1).tail(5)
sales
```

	Market	No_of_Orders	Profit	Sales
Western US	USCA	490	44303.65	251991.83

```
# Printing last 3 entries of the dataframe
sales =pd.read_excel("sales.xlsx", index_col=1).tail(3)
sales
```

Region	Market	No_of_Orders	Profit	Sales
Eastern US	USCA	443	47462.04	264973.98
Central US	USCA	356	33697.43	170416.31
Canada	USCA	49	7246.62	26298.81

Summarising the dataframes

A dataframe can have multiple columns and it is very important to understand what each column stores. You must be familiar with the column names, the data it stores, data type of each column, etc. Let's see different commands that will help you to do that.

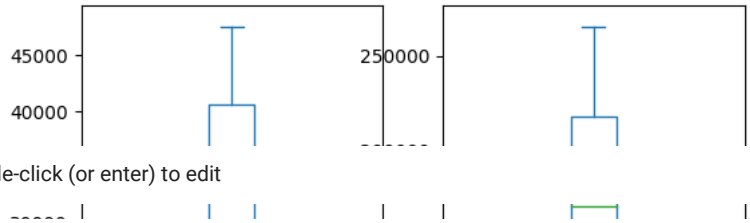
```
# Summarising the dataframe structure
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 3 entries, Eastern US to Canada
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Market        3 non-null      object
 1   No_of_Orders  3 non-null      int64
 2   Profit        3 non-null      float64
 3   Sales         3 non-null      float64
dtypes: float64(2), int64(1), object(1)
memory usage: 228.0+ bytes
```

```
# Summary of data stored in each column
sales.describe()
```

	No_of_Orders	Profit	Sales
count	3.000000	3.000000	3.000000
mean	282.666667	29468.696667	153896.366667
std	206.983896	20438.484304	120192.098698
min	49.000000	7246.620000	26298.810000
25%	202.500000	20472.025000	98357.560000
50%	356.000000	33697.430000	170416.310000
75%	399.500000	40579.735000	217695.145000
max	443.000000	47462.040000	264973.980000

```
# Graphically summarising the spread of the columns - Profit and Sales
import matplotlib.pyplot as plt
sales[["Profit","Sales"]].plot(kind ="box",subplots=True)
plt.show()
```



Double-click (or enter) to edit

Python Libraries - Pandas - Indexing and Slicing

In this section, you will:

- Select rows from a dataframe
- Select columns from a dataframe
- Select subsets of dataframes

Selecting Rows

Selecting rows in dataframes is similar to the indexing you have seen in numpy arrays. The syntax `df[start_index:end_index]` will subset rows according to the start and end indices.

```
# Read data from the file 'sales.xlsx'
sales =pd.read_excel("sales.xlsx").head()

# Check the created dataframe
# Remember - you should print limited number of entries to check the dataframe


# Selecting first 5 rows of the dataframe
sales[0:5]
```

	Market	Region	No_of_Orders	Profit	Sales
0	Africa	Western Africa	251	-12901.51	78476.06
1	Africa	Southern Africa	85	11768.58	51319.50
2	Africa	North Africa	182	21643.08	86698.89
3	Africa	Eastern Africa	110	8013.04	44182.60
4	Africa	Central Africa	103	15606.30	61689.99

```
# Selecting all the even indices of the dataframe
sales[0::2]
```

	Market	Region	No_of_Orders	Profit	Sales
0	Africa	Western Africa	251	-12901.51	78476.06
2	Africa	North Africa	182	21643.08	86698.89
4	Africa	Central Africa	103	15606.30	61689.99

Selecting Columns

There are two simple ways to select a single column from a dataframe:

- `df['column']` or `df.column` return a series
- `df[['col_x', 'col_y']]` returns a dataframe

```
# Select the column 'Profit' from the dataframe 'Sales'. Output must be in the form of a dataframe.
sales[["Profit"]]
```

```
# Check the type of the sliced data
type(sales[["Profit"]])
```

```
pandas.core.frame.DataFrame
```

```
# Select the column 'Profit' from the dataframe 'Sales'. Output must be in the form of a series.
sales["Profit"]
```

```
0   -12901.51
1    11768.58
2    21643.08
3     8013.04
4    15606.30
Name: Profit, dtype: float64
```

```
# Check the type of the sliced data
type(sales["Profit"])
```

```
pandas.core.series.Series
```

Selecting Multiple Columns

You can select multiple columns by passing the list of column names inside the `[]`: `df[['column_1', 'column_2', 'column_n']]`.

```
# Selecting multiple columns from a dataframe
sales[["Profit","Region"]]
```

	Profit	Region
0	-12901.51	Western Africa
1	11768.58	Southern Africa
2	21643.08	North Africa
3	8013.04	Eastern Africa
4	15606.30	Central Africa

Label and Position Based Indexing: `df.loc` and `df.iloc`

You have seen some ways of selecting rows and columns from dataframes. Let's now see some other ways of indexing dataframes, which pandas recommends, since they are more explicit (and less ambiguous).

There are two main ways of indexing dataframes:

1. Label based indexing using `df.loc`
2. Position based indexing using `df.iloc`

Using both the methods, we will do the following indexing operations on a dataframe:

- Selecting single elements/cells
- Selecting single and multiple rows
- Selecting single and multiple columns
- Selecting multiple rows and columns

Label-based Indexing

```
# Select the row with index label as 'Canada'
sales.iloc[0:5,2:]
```

	No_of_Orders	Profit	Sales
0	251	-12901.51	78476.06
1	85	11768.58	51319.50
2	182	21643.08	86698.89
3	110	8013.04	44182.60
4	103	15606.30	61689.99

```
# Select the row with index label as 'Canada' and 'Western Africa'
```

```
# Select the row with index label as 'Canada' and 'Western Africa' along with the columns 'Profit' and 'Sales'
```

Position-based Indexing

```
# Select the top 5 rows and all the columns starting from second column
sales.iloc[0:5,2:]
```

	No_of_Orders	Profit	Sales
0	251	-12901.51	78476.06
1	85	11768.58	51319.50
2	182	21643.08	86698.89
3	110	8013.04	44182.60
4	103	15606.30	61689.99

```
# Select all the entries with positive profit
sales[["Profit"]]>0
```

	Profit
0	False
1	True
2	True
3	True
4	True

```
# Count the number of entries in the dataframe with positive profit
sales[sales[["Profit"]]>0]
```

	Market	Region	No_of_Orders	Profit	Sales
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	11768.58	NaN
2	NaN	NaN	NaN	21643.08	NaN
3	NaN	NaN	NaN	8013.04	NaN
4	NaN	NaN	NaN	15606.30	NaN

```
# Select all the enries in Latin America and European market where Sales>250000
sales[sales[["Profit"]]>0]
```

	Market	Region	No_of_Orders	Profit	Sales
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	11768.58	NaN
2	NaN	NaN	NaN	21643.08	NaN
3	NaN	NaN	NaN	8013.04	NaN
4	NaN	NaN	NaN	15606.30	NaN

Double-click (or enter) to edit

```
sales[sales[["Profit"]]>0].size
```

```
25
```

```
sales[sales[["Profit"]]>0].count()
sales[sales[["Profit"]]>0].size
sales.count()
```

```
Market          5
Region          5
No_of_Orders    5
Profit          5
Sales           5
dtype: int64
```

```
sales=pd.read_excel("sales.xlsx")
sales[(sales["Sales"]>250000)&(sales["Market"].isin(["LATIN","Europe"]))]
```

	Market	Region	No_of_Orders	Profit	Sales
11	Europe	Western Europe	964	82091.27	656637.14
13	Europe	Northern Europe	367	43237.44	252969.09