13-09-2023

Project Title:

Analysis and visualization of Myntra Private Ltd.

Name: T.Jaya Krishna

Branch: CSE-B

Year: B.tech 4th Year

Project Guide

Mr. Lokesh B. Sir

Problem Statement

An organisation ha sales data related to the ecommerce business analysis, the organization wants complete information regarding their sales in Market, Region, No of Order, Profit based on Parameters.

Tools:

1. Jupyter Notebook.
2. Python Programing Language. 3. Numpy (Python Libery_)
4. Google Colab.

Project Title: Analysis and visualization of "sales.excel" using python computing library called as pandas

Problem Statement : an organisation ha sales data related to the ecommers business analysis,the orginasation wants complete information regarding their sales in Market,Region,No of Orders,Pro t based on Parameters.

As a data science engineer my task is provide real time error free solution

Task1:Visualization and Analysis of sales depending on mentions Parameters

Task2:Find out what are the various regions and market sales is grow

Orginasation Name : Myntra Fashion Private Limited

Python Libraries - Pandas - Describing Data

You will be working with les 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 ine cieent 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	Southern Asia	251	-12901.51	78476.06
11	Europe	Northern Europe	37	-2649.76	8190.74
12	Europe	Eastern Europe	85	11768.58	51319.50
13	Europe	Central Asia	964	82091.27	656637.14
14	Europe	Western Europe	182	21643.08	86698.89
15	LATAM	Western Europe	338	18911.49	215703.93
16	LATAM	Southern Europe	110	8013.04	44182.60
17	LATAM	Southern Europe	367	43237.44	252969.09
18	USCA	Northern Europe	103	15606.30	61689.99
19	USCA	Northern Europe	241	25050.69	108258.93
20	USCA	Eastern Europe	382	-16766.90	124312.24
21	USCA	Eastern Europe	496	12377.59	210710.49
22	USCA	Asia Pacific	469	67998.76	351806.60
		South America	930	74679.54	461670.28
		Central America	533	20948.84	329751.38
		Caribbean	288	13529.59	116333.05
		Asia Pacific	646	54734.02	408002.98
		Western US	490	44303.65	251991.83
		Southern US	414	72805.10	315390.77
		Eastern US	255	19991.83	148771.91
		Europe	37	-2649.76	8190.74
		Central US	443	47462.04	264973.98
		Europe	964	82091.27	656637.14
		Canada	356	33697.43	170416.31
		Europe	338	18911.49	215703.93
		Europe	49	7246.62	26298.81
		LATAM	367	43237.44	252969.09
		LATAM	241	25050.69	108258.93
		LATAM	496	12377.59	210710.49
		USCA	930	74679.54	461670.28
		USCA	288	13529.59	116333.05
		USCA	490	44303.65	251991.83
		USCA	255	19991.83	148771.91
		USCA	443	47462.04	264973.98
		USCA	356	33697.43	170416.31
		USCA	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
# 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
```

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()
```

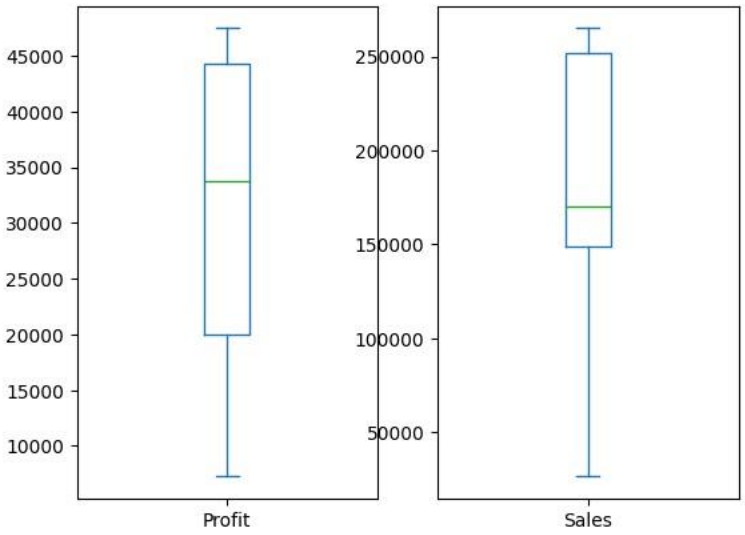
```
<class 'pandas.core.frame.DataFrame'>
Index: 5 entries, Western US to Canada Data
columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
0   Market        5 non-null      object
1   No_of_Orders  5 non-null      int64
2   Profit        5 non-null      float64 3
    Sales         5 non-null      float64
    dtypes: float64(2), int64(1), object(1)
    memory usage: 372.0+ bytes
```

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

	No_of_Orders	Profit	Sales
count	5.000000	5.000000	5.000000
mean	318.600000	30540.314000	172490.568000

std	175.343377	16879.042516	95932.970799
min	49.000000	7246.620000	26298.810000
25%	255.000000	19991.830000	148771.910000
50%	356.000000	33697.430000	170416.310000
75%	443.000000	44303.650000	251991.830000
max	490.000000	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()
sales

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

	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 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"]]
```

	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.loc["Canada"]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-36-ffe22e40163c> in <cell line: 2>()
      1 # Select the row with index label as 'Canada'
----> 2 sales.loc["Canada"]

 ____________________________________ 4 frames _____________________________________
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/range.py in get_loc(self, key, method, tolerance)
393                     raise KeyError(key) from err
394                     self._check_indexing_error(key)
--> 395             raise KeyError(key)
    396         return super().get_loc(key, method=method, tolerance=tolerance)
397

KeyError: 'Canada'

 SEARCH STACK OVERFLOW
```

```
# Select the row with index label as 'Canada' and 'Western Africa'
sales.loc[["Western Africa","Canada"]]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-37-ef3a992e6f55> in <cell line: 2>()
      1 # Select the row with index label as 'Canada' and 'Western Africa' ----
> 2 sales.loc[["Western Africa","Canada"]]

 ____________________________________ 5 frames _____________________________________
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in _raise_if_missing(self, key, indexer,
axis_name)
6128                     if use_interval_msg:
6129                     key = list(key)
# Select the row with index label as 'Canada' and -> 6130                 raise KeyError(f"None of [{key}] are in the
[{axis_name}]"'Western Africa' along with the columns 'Profit' a)                                        nd 'Sales'
        6131
        6132             not_found = list(ensure_index(key)[missing_mask.nonzero()[0]].unique())
```

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

	Market	Region	No_of_Orders	Profit	Sales
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

```
# Count the number of entries in the dataframe with positive profit
sales[sales["Profit"] >0].count()
#sales[sales["Profit"] >0].size
#sales.count()
```

```
Market          4
Region          4
No_of_Orders    4
Profit          4
Sales           4


dtype: int64
```

```
# Select all the enries in Latin America and European market where Sales>250000
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 : Accordinfg to the my conclusion Europe,USCA gives height sales.These are the two markets which give height sales.

0s completed at 12:44 PM