PANDAS DATAFRAME: A Quick Review

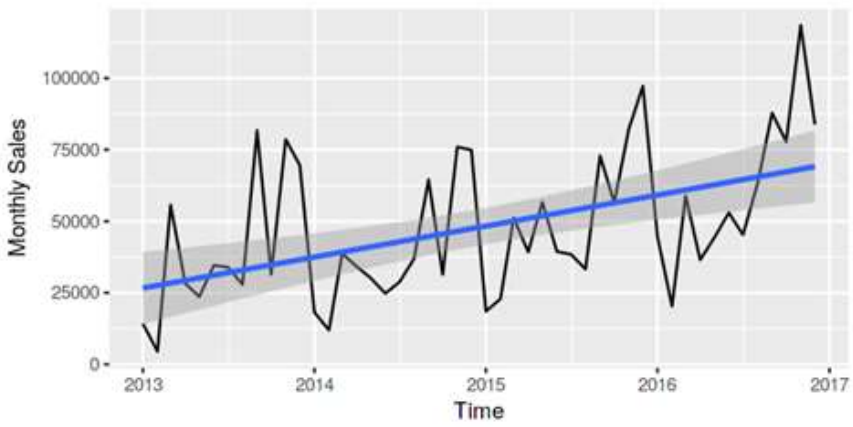
Analysis and Visualization of large datasets represented in a PANDAS DataFrame

A Pandas DataFrame is a type of a 2 dimensional data structure which can be used to store large datasets in a matrix (tabular) format comprising of rows and columns

Figure below shows a Pandas DataFrame comprising of following details of five students arranged in 5 ROWs x 5 COLUMNs

- Every ROW represents a **Data Record**
- Every COLUMN represents an **ATTRIBUTE or PARAMETER** related to STUDENT

Analysis of Dataset having Time-Series Data



TIME DATA SERIES: DataFrame having Date/Time Index Column

- A time-series data is a series of data points or observations recorded at different or regular time intervals.
- A 'time Series' refers to TIME-DEPENDENT data
- It is a sequence of data points taken at equally spaced time intervals.
- The frequency of recorded data points may be hourly, daily, weekly, monthly, quarterly or annually.
- The only variable parameter in a TIME-SERIES data is the TIME

ACTIVITY 1

Date	Product1	Product2	Product3
2023-04-24	100.0	20.0	60.0
2023-04-25	NaN	NaN	55.0
2023-04-26	70.0	78.0	78.0
2023-04-27	82.0	65.0	65.0
2023-04-28	125.0	NaN	NaN
2023-04-29	65.0	100.0	100.0
2023-04-30	NaN	50.0	50.0
2023-05-01	30.0	80.0	80.0
2023-05-02	50.0	55.0	55.0

The above table shows sales record of three products, **Product 1, Product 2 and Product 3**, monitored from 24-04-2023 to 02-05-2023

1. Create a DataFrame as shown above to store detaile of sales of Product 1, Product 2 and Product 3
2. Fill Missing/Invalid Entries as follows:

- In 'Product 1' column: 25
- In 'Product 2' column: 50
- In 'Product 3' column: 30

3. Add a new row containing Total Sales of Product 1, Product 2 and Product 3

4.Compare Total Sales of Product 1, Product 2 and Product 3 through:

- Line plot (scatter plot)
- Bar plot

- Area plot

3. Analyze sales of Product 1 across one week through a Pie Chart

1A. CREATING DATE RANGE using pd.range_range()

and using different date frequency parameters

freq = 'D', Calender Day Frequency

freq = 'W', Weekly Frequency

fred = 'B', Business Day Frequency

Date Range with freq = 'D', Calender Day Frequency

```
# Date Range with freq = 'D', Calender Day Frequency

import pandas as pd

DATE_SERIES= pd.date_range(start='2023/04/24',
                           end='2023/05/02',
                           freq='D')
display(DATE_SERIES)

DATE_SERIES=pd.Series(DATE_SERIES)

display(DATE_SERIES)
```

```
DatetimeIndex(['2023-04-24', '2023-04-25', '2023-04-26', '2023-04-27',
               '2023-04-28', '2023-04-29', '2023-04-30', '2023-05-01',
               '2023-05-02'],
              dtype='datetime64[ns]', freq='D')
0   2023-04-24
1   2023-04-25
2   2023-04-26
3   2023-04-27
4   2023-04-28
5   2023-04-29
6   2023-04-30
7   2023-05-01
8   2023-05-02
dtype: datetime64[ns]
```

Date Range with freq = 'B', Business Day Frequency

```
# Date Range with freq = 'B', Business Day Frequency

import pandas as pd

DATE_SERIES= pd.date_range(start='2023/04/24',
                           end='2023/05/02',
                           freq='B')
display(DATE_SERIES)

DATE_SERIES=pd.Series(DATE_SERIES)

display(DATE_SERIES)
```

```
DatetimeIndex(['2023-04-24', '2023-04-25', '2023-04-26', '2023-04-27',
               '2023-04-28', '2023-05-01', '2023-05-02'],
              dtype='datetime64[ns]', freq='B')
0   2023-04-24
1   2023-04-25
2   2023-04-26
3   2023-04-27
4   2023-04-28
5   2023-05-01
6   2023-05-02
dtype: datetime64[ns]
```

Date Range with freq = 'W', Weekly Frequency

```
# Date Range with freq = 'W', Weekly Frequency

import pandas as pd

DATE_SERIES= pd.date_range(start='2023/04/24',
                           end='2023/05/24',
                           freq='W')
display(DATE_SERIES)

DATE_SERIES=pd.Series(DATE_SERIES)

display(DATE_SERIES)
```

```
DatetimeIndex(['2023-04-30', '2023-05-07', '2023-05-14', '2023-05-21'], dtype='datetime64[ns]', freq='W-SUN')
0   2023-04-30
1   2023-05-07
2   2023-05-14
3   2023-05-21
```

Date Range with freq = 'M', Monthly Frequency

```
# Date Range with freq = 'M', Monthly Frequency

import pandas as pd

DATE_SERIES= pd.date_range(start='2023/04/10',
                           end='2023/07/16',
                           freq='M')
display(DATE_SERIES)

DATE_SERIES=pd.Series(DATE_SERIES)

display(DATE_SERIES)
```

```
DatetimeIndex(['2023-04-30', '2023-05-31', '2023-06-30'], dtype='datetime64[ns]', freq='M')
0   2023-04-30
1   2023-05-31
2   2023-06-30
dtype: datetime64[ns]
```

Some of the values which can be given to **freq** parameter in pd.date_range() is given below:

Alias	Description
B	business day frequency
D	calendar day frequency
W	weekly frequency
H	hourly frequency
T, min	minutely frequency
S	secondly frequency
L, ms	milliseconds
U, us	microseconds
N	nanoseconds

1B. CHANGING DATE FORMAT

using dt.strftime( )

'DATE FORMAT 1: dd-mm-yyyy'

```
# Changing Date Format

import pandas as pd

DATE_SERIES= pd.date_range(start='2023/04/24',
                           end='2023/05/02',
                           freq='D')

DATE_SERIES=pd.Series(DATE_SERIES)

print('FORMAT 1')
DATE_SERIES=DATE_SERIES.dt.strftime('%d-%m-%Y')
display(DATE_SERIES)
```

```
FORMAT 1
0    24-04-2023
1    25-04-2023
2    26-04-2023
3    27-04-2023
4    28-04-2023
5    29-04-2023
6    30-04-2023
7    01-05-2023
8    02-05-2023
dtype: object
```

'DATE FORMAT 2: dd/mm/yyyy'

```
import pandas as pd

DATE_SERIES= pd.date_range(start='2023/04/24',
                           end='2023/05/02',
                           freq='D')

DATE_SERIES=pd.Series(DATE_SERIES)

print('FORMAT 2')
DATE_SERIES=DATE_SERIES.dt.strftime('%d/%m/%Y')
display(DATE_SERIES)
```

```
FORMAT 2
0    24/04/2023
1    25/04/2023
2    26/04/2023
3    27/04/2023
4    28/04/2023
5    29/04/2023
6    30/04/2023
7    01/05/2023
8    02/05/2023
dtype: object
```

'DATE FORMAT 3: April 24, 2023

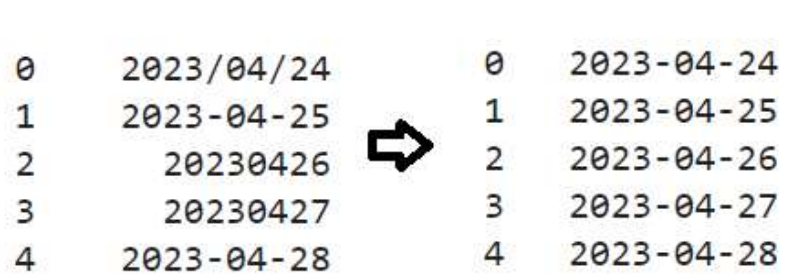
```
import pandas as pd

DATE_SERIES= pd.date_range(start='2023/04/24',
                           end='2023/05/02',
                           freq='D')

DATE_SERIES=pd.Series(DATE_SERIES)

print('FORMAT 3')
DATE_SERIES=DATE_SERIES.dt.strftime('%d %B, %Y')
display(DATE_SERIES)
```

```
FORMAT 3
0    24 April, 2023
1    25 April, 2023
2    26 April, 2023
3    27 April, 2023
4    28 April, 2023
5    29 April, 2023
6    30 April, 2023
7      01 May, 2023
8      02 May, 2023
dtype: object
```

Converting series of DATES into a standard DATE FORMAT (yyyy-mm-dd)

Using pd.to_datetime()

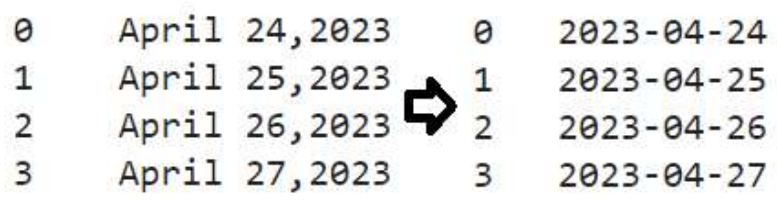
Converting DATE in format yyyy/mm/dd to yyyy-mm-dd




```
import pandas as pd

print('Sequence of Dates in format 2023/04/24')

DATE_SERIES=pd.Series(['2023/04/24','2023-04-25','20230426','20230427','2023-04-28'])

display(DATE_SERIES)

print('Converting DATES in format 2023/04/24 to yyyy-mm-dd format')
DATE_SERIES=pd.to_datetime(DATE_SERIES, format='%Y/%m/%d')
display(DATE_SERIES)
```

```
Sequence of Dates in format 2023/04/24
0    2023/04/24
1    2023-04-25
2      20230426
```

Converting DATE in format April 24, 2023 to yyyy-mm-dd




```
import pandas as pd

print('Sequence of Dates in format April 24, 2023')
DATE_SERIES=pd.Series(['April 24,2023','April 25,2023','April 26,2023','April 27,2023'])
display(DATE_SERIES)

print('Converting DATES in format, April 24, 2023 to yyyy-mm-dd format')
DATE_SERIES=pd.to_datetime(DATE_SERIES, format='%B %d,%Y')
display(DATE_SERIES)
```

```
Sequence of Dates in format April 24, 2023
0    April 24,2023
1    April 25,2023
2    April 26,2023
3    April 27,2023
dtype: object
Converting DATES in format, April 24, 2023 to yyyy-mm-dd format
0   2023-04-24
1   2023-04-25
2   2023-04-26
3   2023-04-27
dtype: datetime64[ns]
```

ACTIVITY 1

Date	Product1	Product2	Product3
2023-04-24	100.0	20.0	60.0
2023-04-25	NaN	NaN	55.0
2023-04-26	70.0	78.0	78.0
2023-04-27	82.0	65.0	65.0
2023-04-28	125.0	NaN	NaN
2023-04-29	65.0	100.0	100.0
2023-04-30	NaN	50.0	50.0
2023-05-01	30.0	80.0	80.0
2023-05-02	50.0	55.0	55.0

ACTIVITY 1

The above table shows sales record of three products, **Product 1, Product 2 and Product 3**, monitored from 24-04-2023 to 02-05-2023

1. Create a DataFrame as shown above to store detaile of sales of Product 1, Product 2 and Product 3
2. Fill Missing/Invalid Entries as follows:

- In 'Product 1' column: 25
- In 'Product 2' column: 50
- In 'Product 3' column: 30

3. Add a new row containing Total Sales of Product 1, Product 2 and Product 3
4. Compare Total Sales of Product 1, Product 2 and Product 3 through:
    - Line plot (scatter plot)
    - Bar plot
    - Area plot
5. Analyze sales of Product 1 across one week through a Pie Chart

I. Create a DataFrame to store detaile of sales of Product 1, Product 2 and Product 3

```
import pandas as pd
import numpy as np

DATE_SERIES= pd.date_range(start='2023/04/24',
                           end='2023/05/02',
                           freq='D')

Sales_Product1=[100, np.nan,70,82,125,65,np.nan,30,50]
Sales_Product2=[20, np.nan,78,65,np.nan,100,50,80,55]
Sales_Product3=[60, 55,78,65,np.nan,100,50,80,55]

SALES ={'Product1':Sales_Product1, 'Product2':Sales_Product2,  'Product3':Sales_Product3}

SALES_RECORD = pd.DataFrame(SALES, index=DATE_SERIES)
SALES_RECORD.index.name='Date'

display(SALES_RECORD)

SALES_RECORD.to_csv('SALES_RECORD.csv')
```

Date	Product1	Product2	Product3
2023-04-24	100.0	20.0	60.0
2023-04-25	NaN	NaN	55.0
2023-04-26	70.0	78.0	78.0
2023-04-27	82.0	65.0	65.0
2023-04-28	125.0	NaN	NaN
2023-04-29	65.0	100.0	100.0
2023-04-30	NaN	50.0	50.0
2023-05-01	30.0	80.0	80.0
2023-05-02	50.0	55.0	55.0

II. Fill Missing/Invalid Entries in multiple columns with different values:

In 'Product 1' column: 25

In 'Product 2' column: 50

In 'Product 3' column: 30

Using fillna() to replace INVALID/MISSING entries in different columns with different values

```
# using isnull() to detect loaction of missing entries
# using fillna() to replace invalid entries

import pandas as pd
import numpy as np

DATE_SERIES= pd.date_range(start='2023/04/24',
                           end='2023/05/02',
                           freq='D')
DATE_SERIES= pd.Series(DATE_SERIES)
DATE_SERIES=DATE_SERIES.dt.strftime('%d %B, %Y')

Sales_Product1=[100, np.nan,70,82,125,65,np.nan,30,50]
Sales_Product2=[20, np.nan,78,65,np.nan,100,50,80,55]
Sales_Product3=[60, 55,78,65,np.nan,100,50,80,55]

SALES ={'Product1':Sales_Product1, 'Product2':Sales_Product2,  'Product3':Sales_Product3}

SALES_RECORD = pd.DataFrame(SALES, index=DATE_SERIES)
SALES_RECORD.index.name='Date'

display(SALES_RECORD)

print('Missing Entries')
print(SALES_RECORD.isnull())

print('\nMETHOD 1')
print('\nModified DataFrame')
SALES_RECORD_M= pd.DataFrame()
SALES_RECORD_M['Product1']=SALES_RECORD['Product1'].fillna(25)
SALES_RECORD_M['Product2']=SALES_RECORD['Product2'].fillna(50)
SALES_RECORD_M['Product3']=SALES_RECORD['Product3'].fillna(30)

display(SALES_RECORD_M)

print('\nMETHOD 2')
print('\nModified DataFrame')
```

```
SALES_RECORD_M=SALES_RECORD.fillna(value={'Product1': 25,'Product2':50,'Product3':30})
display(SALES_RECORD_M)
```

Date	Product1	Product2	Product3
24 April, 2023	100.0	20.0	60.0
25 April, 2023	NaN	NaN	55.0
26 April, 2023	70.0	78.0	78.0
27 April, 2023	82.0	65.0	65.0
28 April, 2023	125.0	NaN	NaN
29 April, 2023	65.0	100.0	100.0
30 April, 2023	NaN	50.0	50.0
01 May, 2023	30.0	80.0	80.0
02 May, 2023	50.0	55.0	55.0

```
Missing Entries
               Product1  Product2  Product3
Date
24 April, 2023     False     False     False
25 April, 2023      True      True     False
26 April, 2023     False     False     False
27 April, 2023     False     False     False
28 April, 2023     False      True      True
29 April, 2023     False     False     False
30 April, 2023      True     False     False
01 May, 2023       False     False     False
02 May, 2023       False     False     False

METHOD 1

Modified DataFrame
```

Date	Product1	Product2	Product3
24 April, 2023	100.0	20.0	60.0
25 April, 2023	25.0	50.0	55.0
26 April, 2023	70.0	78.0	78.0
27 April, 2023	82.0	65.0	65.0
28 April, 2023	125.0	50.0	30.0
29 April, 2023	65.0	100.0	100.0
30 April, 2023	25.0	50.0	50.0
01 May, 2023	30.0	80.0	80.0
02 May, 2023	50.0	55.0	55.0

```
METHOD 2

Modified DataFrame
```

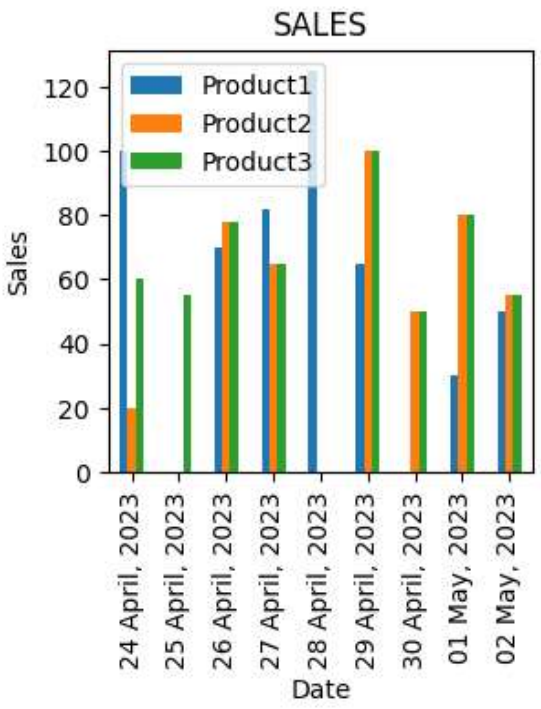
Date	Product1	Product2	Product3
24 April, 2023	100.0	20.0	60.0
25 April, 2023	25.0	50.0	55.0
26 April, 2023	70.0	78.0	78.0
27 April, 2023	82.0	65.0	65.0
28 April, 2023	125.0	50.0	30.0
29 April, 2023	65.0	100.0	100.0
30 April, 2023	25.0	50.0	50.0
01 May, 2023	30.0	80.0	80.0
02 May, 2023	50.0	55.0	55.0

Comparing Sales of Product 1, Product 2 and Product 3 in terms of:

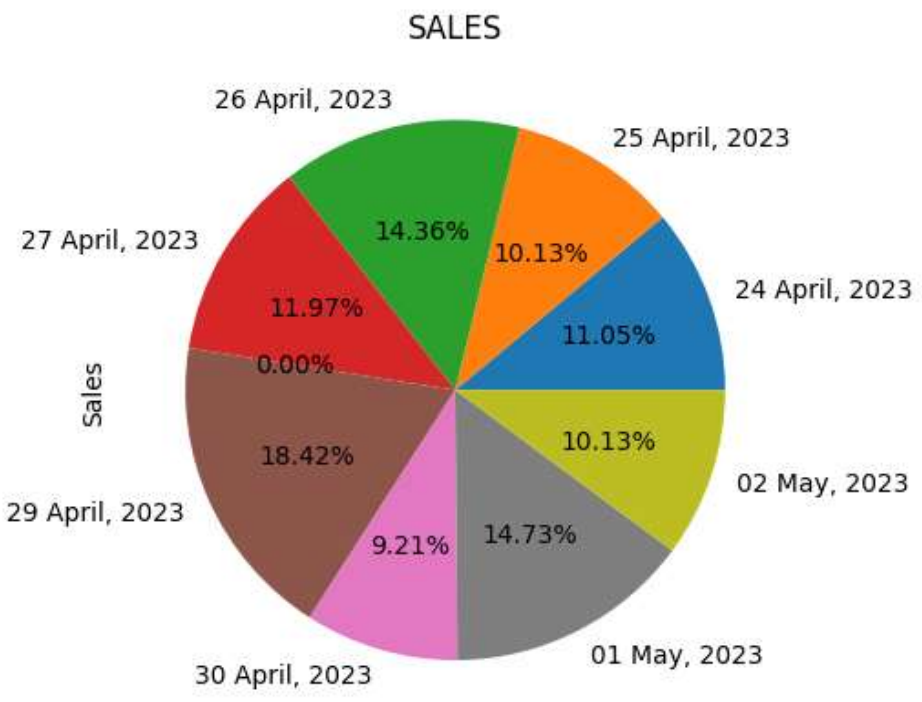
- BAR PLOT
- PIE CHART

```
# BAR PLOT
SALES_RECORD.plot.bar(figsize=(3,3),
                      xlabel='Date',
                      ylabel='Sales',
                      title='SALES')
plt.show()
```

```
# PIE PLOT
SALES_RECORD['Product3'].plot.pie(autopct='%1.2f%%',
                      xlabel='Date', ylabel='Sales',title='SALES')
```




```
<Axes: title={'center': 'SALES'}, ylabel='Sales'>
```



III. Add a new row containing Total Sales of Product 1, Product 2 and Product 3

```
SALES_RECORD_M.loc['TOTAL SALES']=SALES_RECORD_M.sum()

display(SALES_RECORD_M)
```

Date	Product1	Product2	Product3
24 April, 2023	100.0	20.0	60.0
25 April, 2023	25.0	50.0	55.0
26 April, 2023	70.0	78.0	78.0
27 April, 2023	82.0	65.0	65.0
28 April, 2023	125.0	50.0	30.0
29 April, 2023	65.0	100.0	100.0
30 April, 2023	25.0	50.0	50.0
01 May, 2023	30.0	80.0	80.0
02 May, 2023	50.0	55.0	55.0
TOTAL SALES	572.0	548.0	573.0

IV. Compare Total Sales of Product 1, Product 2 and Product 3 through:

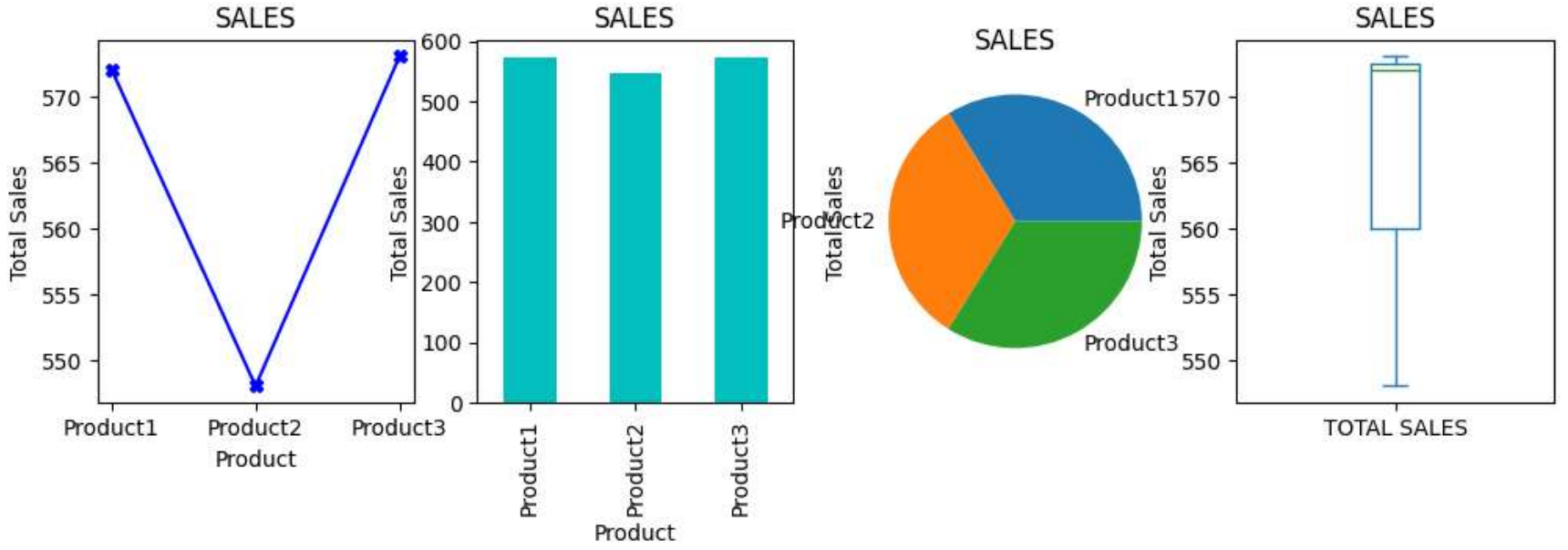
- Line plot (scatter plot)
- Bar plot
- Pie chart
- Box plot

```
import matplotlib.pyplot as plt   # importing matplotlib library

display(SALES_RECORD_M)
```

```
TOTAL_SALES= pd.Series(SALES_RECORD_M.loc['TOTAL SALES'])

print('\nTOTAL SALES OF PRODUCT 1, PRODUCT 2 and PRODUCT 3\n')
display(TOTAL_SALES)

plt.figure(figsize=(12,3)) # defining size of the plot region
                           # 12 inches wide X 3 inches length

'''figsize=(9,3) can accomodate 4 GRAPHS arranged in 1 ROWs and 4 COLUMNS
   with each graph of size 3 inches wide X 3 inches length'''

plt.subplot(1,4,1) # PLOT 1: Line Plot
TOTAL_SALES.plot.line(color='b', marker='X',
                      xlabel = 'Product',
                      ylabel = 'Total Sales',
                      title='SALES')

plt.subplot(1,4,2) # PLOT 2: Bar Plot
TOTAL_SALES.plot.bar(color='c',
                      xlabel = 'Product',
                      ylabel = 'Total Sales',
                      title='SALES')

plt.subplot(1,4,3) # PLOT 3: Area Plot
TOTAL_SALES.plot.pie( xlabel = 'Product',
                      ylabel = 'Total Sales',
                      title='SALES')

plt.subplot(1,4,4) # PLOT 3: Area Plot
TOTAL_SALES.plot.box( ylabel = 'Total Sales',
                      title='SALES')
```

Date	Product1	Product2	Product3
24 April, 2023	100.0	20.0	60.0
25 April, 2023	25.0	50.0	55.0
26 April, 2023	70.0	78.0	78.0
27 April, 2023	82.0	65.0	65.0
28 April, 2023	125.0	50.0	30.0
29 April, 2023	65.0	100.0	100.0
30 April, 2023	25.0	50.0	50.0
01 May, 2023	30.0	80.0	80.0
02 May, 2023	50.0	55.0	55.0
TOTAL SALES	572.0	548.0	573.0

```
TOTAL SALES OF PRODUCT 1, PRODUCT 2 and PRODUCT 3

Product1    572.0
Product2    548.0
Product3    573.0
Name: TOTAL SALES, dtype: float64
<Axes: title={'center': 'SALES'}, ylabel='Total Sales'>
```



Sources of DataSets

There are various online sources from where useful Datasets can be downloaded absolutely free of cost Some of sources are listed below:

1. Google Dataset Search: https://datasetsearch.research.google.com/
2. Kaggle: https://www.kaggle.com/datasets
3. Data.Gov: https://data.gov/

4. UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets.php

5. Global Health Observatory Data Repository: https://apps.who.int/gho/data/node.home

PROGRAM 1

CALORIES DATASET

```
import pandas as pd

CALORIES=pd.read_csv('CaloriesDataSet.csv')

display(CALORIES)
```

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

The above table shows details related to daily workout routine of a person

i. Read the above dataset from an external .csv file and store it in a a dataset

ii. Fill the MISSING entries in different columns with some default value

iii. Detect and Remove Duplicate Entries

iv. Transform all the dates in the ‘Date’ column in the correct format

v. Add a new ROW contain ‘Average values’ of ‘Duration’, ‘Pulse’, ‘Maxpulse’ and ‘Calories’

vi. Analyse the trend of ‘Calories’ burnt each day using:

- LINE Plot
- BAR Plot (columnar plot)
- Area Plot

- Box Plot
- Histogram
- Pie Chart (use sub-plot function to display all these plots together)

PROGRAM 2

1	EMP ID	EMP Name	Age	Joining Date	Salary	EMP Credits
2	1	Satish	21	01-11-2017	50000	3.8
3	2	Reeya	23		75000	4
4	3	Jay	40	22-09-2015	100000	5
5	4	Rahul	35	11102016		4.2
6	5	Roy	26	08-01-2017	45000	3.9
7	6	Jay	28	22-09-2015	100000	4.5
8	7	Vishal	29	05-01-2016		5
9	8	Serah	21	06-02-2018	55000	3.7
10	9	Vishal	29	05-01-2016		5
11	10	Prachi	22	06-02-2018	60000	4.3

The above table shows a database of 10 employees

i. Create a pandas Data Frame to store the above dataset

- use pd.date_range( ) function to create' Joining Date' column
- transform dates in format :'November 1, 2017'

ii. Fill the MISSING entries as follows:

- Missing entry in 'Salary' column to be filled with 50000
- Missing entry in 'Joining Date' column to be filled with 01-01-2018

iii. Detect and Remove Duplicate Entries

iv. Determine the number and names of the employees having maximum credits

v. Compare 'EMP Credits' and 'Salary' of employees using BAR plot and Pie Chart

vi. Analyse the relationship between 'Age', 'Salary' and 'EMP Credits'

- Plot EMP Credits Vs. Age
- Plot EMP Credits Vs. Salary

(use subplot function to show multiple plots in a single plot region)

CANADIAN IMMIGRATION DATA SET

https://open.canada.ca/data/en/dataset/2894b1fa-d71e-4793-959f-48329bd38132

https://www.kaggle.com/datasets/umerkk12/canada-immigration-dataset

```
import pandas as pd

CANADA_IMMIGRATION=pd.read_csv('Canadian Immigration Dataset.csv')
```

```
CANADA_IMMIGRATION.head(5)
```

	Unnamed: 0	Draw Number	Date	Immigration program	Invitations issued	CRS score of lowest-ranked candidate invited	Date (hidden)	Programs covered	Month	Year	month_year	Date Full
0	0	172	1/7/2021	Canadian Experience Class	4750	461	7/1/2021	Canadian Experience Class	1	2021	1/1/2021	7-Jan-21
1	1	171	1/6/2021	Provincial Nominee Program	250	813	6/1/2021	Provincial Nominee Program	1	2021	1/1/2021	6-Jan-21
2	2	170	12/23/2020	No program specified	5000	468	12/23/2020	Canadian Experience Class Federal Skilled Wor...	12	2020	12/1/2020	23-Dec-20
3	3	169	12/9/2020	No program specified	5000	469	9/12/2020	Canadian Experience Class Federal Skilled Wor...	12	2020	12/1/2020	9-Dec-20

Number of ROWS and COLUMNS in the given Dataset

```
print('SHAPE: ROWS X COLUMNS')
```