DATA COLLECTION ¶

In [1]:
```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\\fit1.csv")
sd
```

Out[2]:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	0.06	0.08	0.06	75
1	B	0.04	0.17	0.19	160
2	C	0.10	0.12	0.05	101
3	D	0.03	0.22	0.08	127
4	E	0.25	0.11	0.12	179
5	F	0.08	0.16	0.18	167
6	G	0.19	0.09	0.17	171
7	H	0.26	0.06	0.14	170
8	Grand Total	1.00	1.00	1.00	1150

In [3]:
```
# to display top 10 rows
sd.head(10)
```

Out[3]:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	0.06	0.08	0.06	75
1	B	0.04	0.17	0.19	160
2	C	0.10	0.12	0.05	101
3	D	0.03	0.22	0.08	127
4	E	0.25	0.11	0.12	179
5	F	0.08	0.16	0.18	167
6	G	0.19	0.09	0.17	171
7	H	0.26	0.06	0.14	170
8	Grand Total	1.00	1.00	1.00	1150

DATA CLEANING AND PRE_PROCESSING

In [4]:
```
sd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Row Labels          9 non-null      object 
 1   Sum of Jan          9 non-null      float64
 2   Sum of Feb          9 non-null      float64
 3   Sum of Mar          9 non-null      float64
 4   Sum of Total Sales  9 non-null      int64  
dtypes: float64(3), int64(1), object(1)
memory usage: 488.0+ bytes
```

In [5]:
```
# to display summary of statistics
sd.describe()
```

Out[5]:

	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
count	9.000000	9.000000	9.000000	9.000000
mean	0.223333	0.223333	0.221111	255.555556
std	0.304097	0.295508	0.296625	337.332963
min	0.030000	0.060000	0.050000	75.000000
25%	0.060000	0.090000	0.080000	127.000000
50%	0.100000	0.120000	0.140000	167.000000
75%	0.250000	0.170000	0.180000	171.000000
max	1.000000	1.000000	1.000000	1150.000000

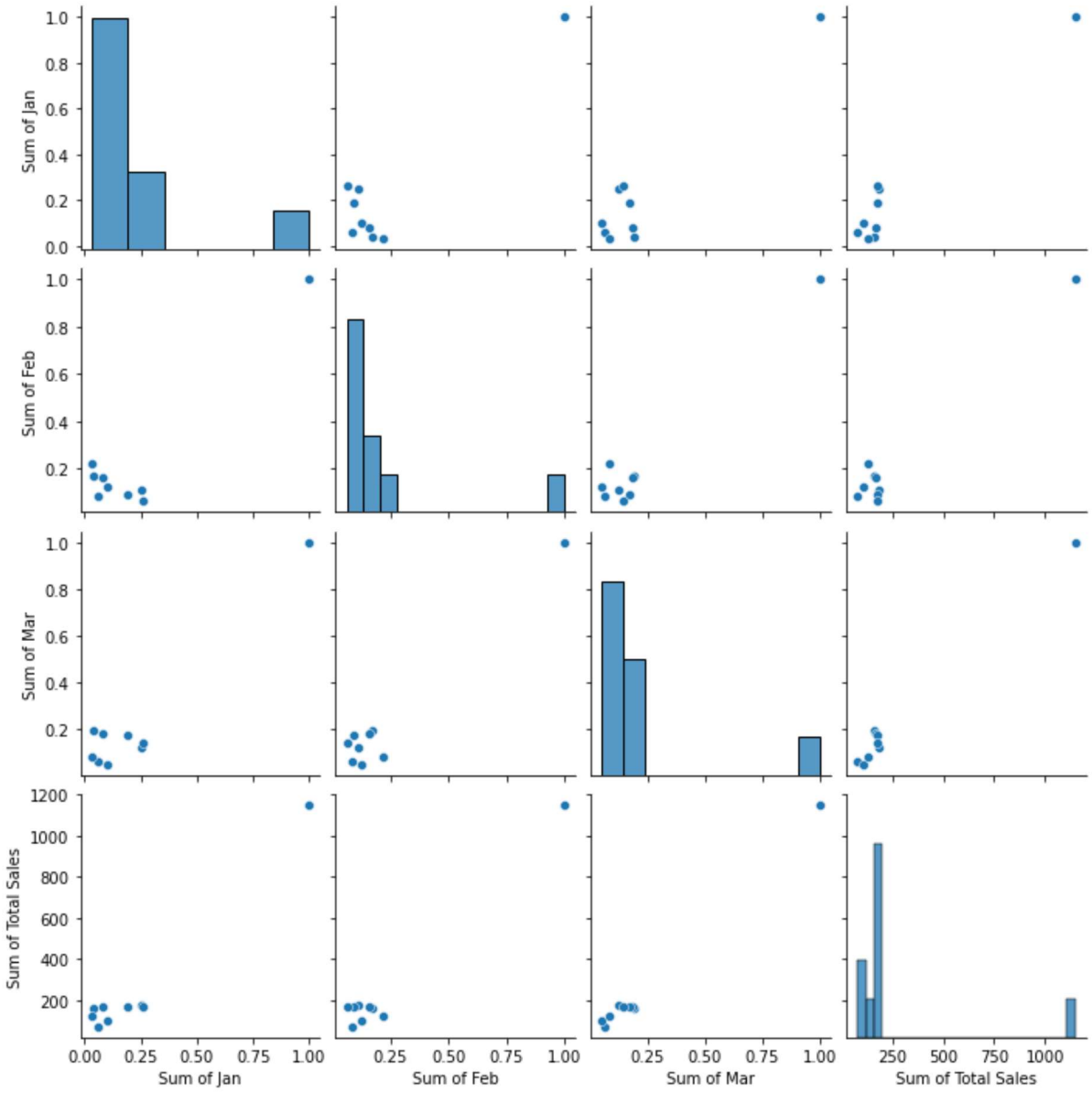
In [6]:
```
#to display colums heading
sd.columns
```

Out[6]:
```
Index(['Row Labels', 'Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales'],
      dtype='object')
```

EDA and visualization

In [7]:
```
sns.pairplot(sd)
```

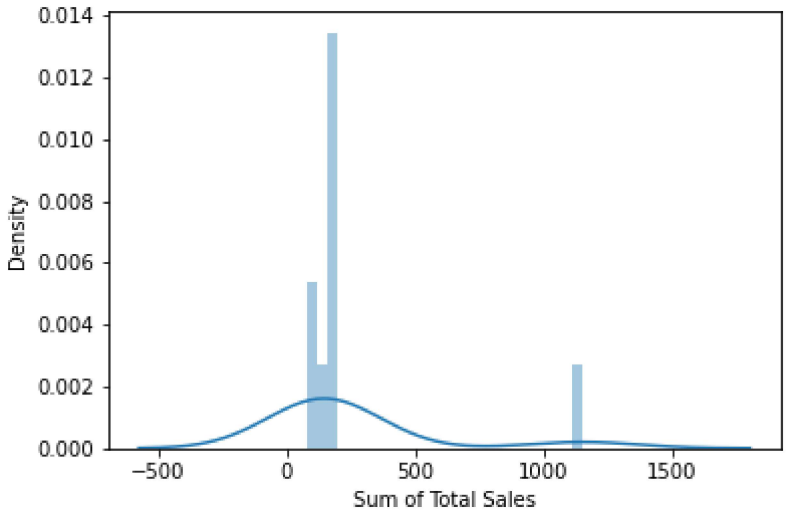
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x18eec7f5af0>
```



In [8]:
```
sns.distplot(sd['Sum of Total Sales'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

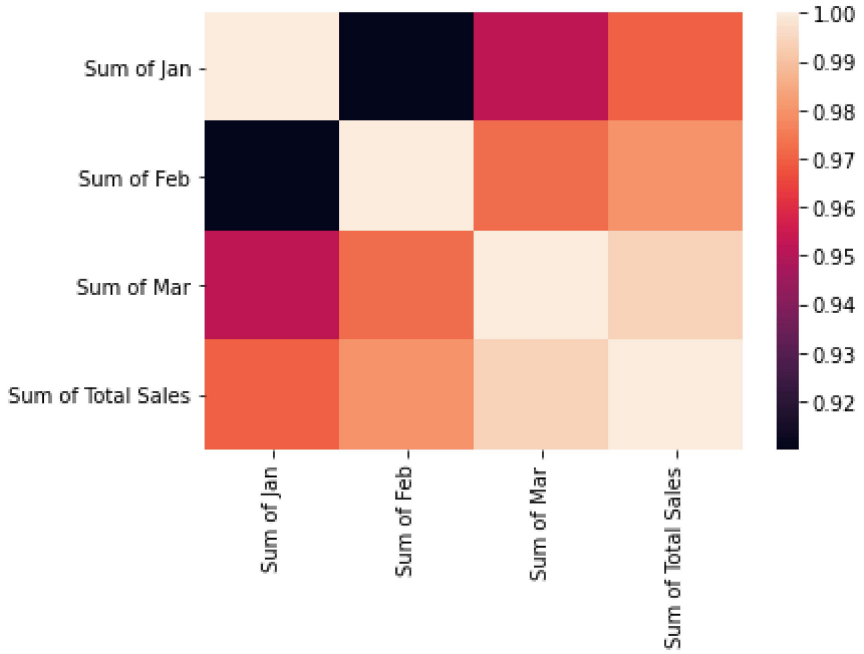
Out[8]: <AxesSubplot:xlabel='Sum of Total Sales', ylabel='Density'>



In [9]:
```
sd1=sd[['Row Labels', 'Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales']]
```

In [10]:
```
sns.heatmap(sd1.corr())
```

Out[10]: <AxesSubplot:>



TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

In [11]:
```
x= sd1[['Sum of Jan', 'Sum of Feb', 'Sum of Mar']]
y=sd1['Sum of Total Sales']
```

In [12]:
```
# To split my dataset  into training data and test data
from sklearn .model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4)
```

In [13]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]:
```
LinearRegression()
```

In [14]:
```
print(lr.intercept_)
```

```
-5.105679303433078
```

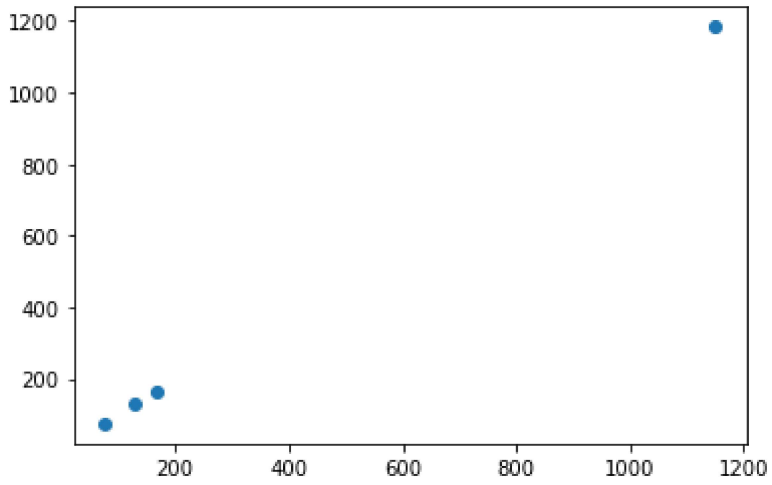
In [15]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
Sum of Jan	350.034788
Sum of Feb	415.947613
Sum of Mar	423.474609

In [16]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x18eef388e80>
```



In [17]:
```
print(lr.score(x_test,y_test))
```

```
0.9984958562974291
```

In [18]:
```
lr.score(x_train,y_train)
```

Out[18]:
```
0.9999857560181115
```

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```

Out[20]:
```
Ridge(alpha=10)
```

In [21]:
```
dr.score(x_test,y_test)
```

Out[21]:
```
-0.24910159066600168
```

In [22]:
```
dr.score(x_train,y_train)
```

Out[22]:
```
0.0036685302763855843
```

In [23]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[23]:
```
Lasso(alpha=10)
```

In [24]:
```
la.score(x_test,y_test)
```

Out[24]:
```
-0.2513488228225036
```

In [25]:
```
la.score(x_train,y_train)
```

Out[25]:
```
0.0
```

ElasticNet

In [26]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[26]:
```
ElasticNet()
```

In [27]:
```
print(en.coef_)
```

```
[ 1.66884606 -0.          1.05429855]
```

In [28]:
```
print(en.intercept_)
```

```
155.77835785712966
```

In [29]:
```
prediction=en.predict(x_test)
```

In [30]:
```
print(en.score(x_test,y_test))
```

```
-0.24553267208354024
```

Evaluation metrics

In [31]:
```
from sklearn import metrics
```

In [33]:
```
print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
mean Absolute Error: 278.0628429786889
```

In [34]:
```
print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean squared Error: 247643.88983072195
```

In [35]:
```
print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root mean Absolytre Error: 497.6383122617489
```

In [ ]: