Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

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

In [2]:
```
df=pd.read_csv("Fitness-1")
```

To display top 10 rows

In [3]:
```
df.head(10)
```

Out[3]:

	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62%	7.73%	6.16%	75
1	B	4.21%	17.27%	19.21%	160
2	C	9.83%	11.60%	5.17%	101
3	D	2.81%	21.91%	7.88%	127
4	E	25.28%	10.57%	11.82%	179
5	F	8.15%	16.24%	18.47%	167
6	G	18.54%	8.76%	17.49%	171
7	H	25.56%	5.93%	13.79%	170
8	Grand Total	100.00%	100.00%	100.00%	1150

Data Cleaning And Pre-Processing

In [4]:
```
1 df.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      object
 2   Sum of Feb          9 non-null      object
 3   Sum of Mar          9 non-null      object
 4   Sum of Total Sales  9 non-null      int64 
dtypes: int64(1), object(4)
memory usage: 488.0+ bytes
```

In [5]:
```
1 # Display the statistical summary
2 df.describe()
```

Out[5]:

	Sum of Total Sales
count	9.000000
mean	255.555556
std	337.332963
min	75.000000
25%	127.000000
50%	167.000000
75%	171.000000
max	1150.000000

In [6]:
```
1 # To display the col headings
2 df.columns
```

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

In [7]:
```
cols=df.dropna(axis=1)
cols
```

Out[7]:

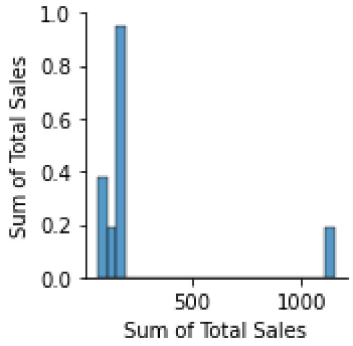
	Row Labels	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	A	5.62%	7.73%	6.16%	75
1	B	4.21%	17.27%	19.21%	160
2	C	9.83%	11.60%	5.17%	101
3	D	2.81%	21.91%	7.88%	127
4	E	25.28%	10.57%	11.82%	179
5	F	8.15%	16.24%	18.47%	167
6	G	18.54%	8.76%	17.49%	171
7	H	25.56%	5.93%	13.79%	170
8	Grand Total	100.00%	100.00%	100.00%	1150

In [ ]:
```
1
```

EDA and Visualization

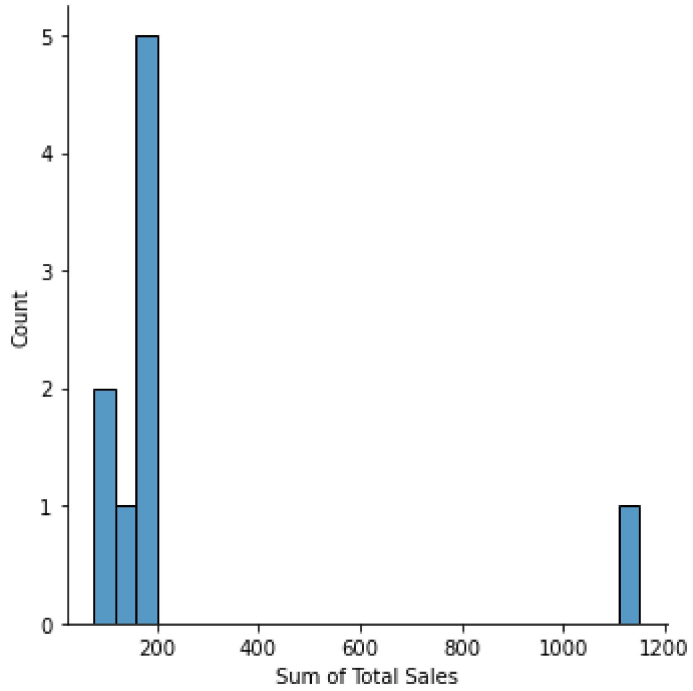
In [8]:
```
sns.pairplot(cols)
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x205ad9a14c0>
```



In [9]:
```
1 sns.displot(df['Sum of Total Sales'])
```

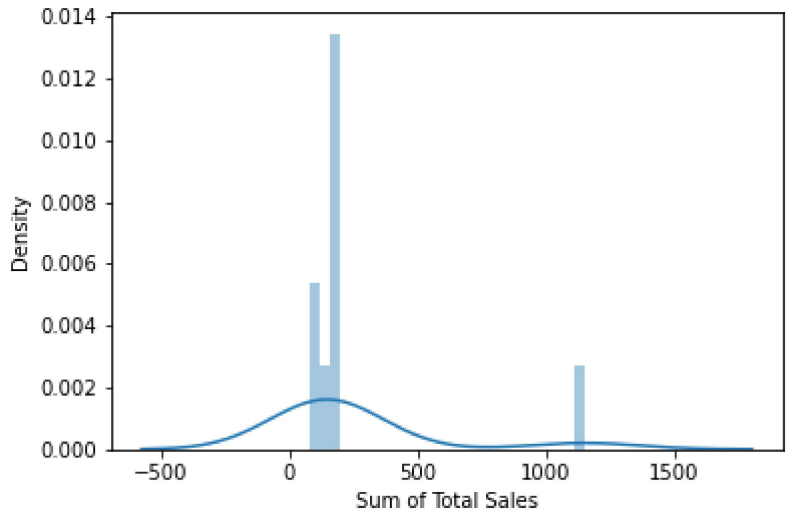
Out[9]: <seaborn.axisgrid.FacetGrid at 0x205ad806b50>



In [10]:
```
# We use displot in older version we get distplot use displot
sns.distplot(df['Sum of Total Sales'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[10]: <AxesSubplot:xlabel='Sum of Total Sales', ylabel='Density'>



In [11]:
```
df1=df[[ 'Sum of Jan', 'Sum of Feb', 'Sum of Mar',
       'Sum of Total Sales']]
df1
```

Out[11]:

	Sum of Jan	Sum of Feb	Sum of Mar	Sum of Total Sales
0	5.62%	7.73%	6.16%	75
1	4.21%	17.27%	19.21%	160
2	9.83%	11.60%	5.17%	101
3	2.81%	21.91%	7.88%	127
4	25.28%	10.57%	11.82%	179
5	8.15%	16.24%	18.47%	167
6	18.54%	8.76%	17.49%	171
7	25.56%	5.93%	13.79%	170
8	100.00%	100.00%	100.00%	1150

In [12]:
```
sns.heatmap(df1.corr())
```

Out[12]: <AxesSubplot:>

To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [13]:
```
x=df1[['Sum of Total Sales']]
y=df1[['Sum of Total Sales']]
```

To split the dataset into test data

In [14]:
```
# importing lib for splitting test data
from sklearn.model_selection import train_test_split
```

In [15]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [16]:
```
from sklearn.linear_model import LinearRegression

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

Out[16]:
```
LinearRegression()
```

In [17]:
```
print(lr.intercept_)
```

```
[-2.84217094e-14]
```

In [18]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

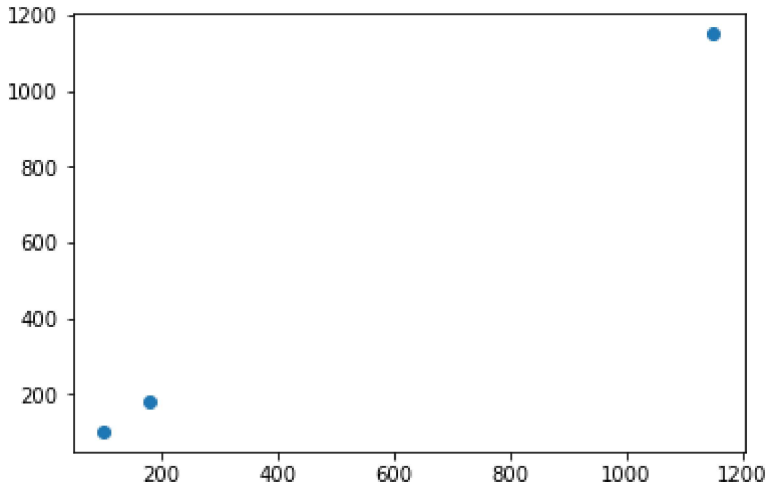
In [19]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[19]:

	0
0	1.0

In [20]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[20]: <matplotlib.collections.PathCollection at 0x205afbe4100>



In [21]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [22]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[22]: Ridge(alpha=10)

In [23]:
```
rr.score(x_test,y_test)
```

Out[23]: 0.9999971737347141

In [24]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[24]:
```
Lasso(alpha=10)
```

In [25]:
```
la.score(x_test,y_test)
```

Out[25]:
```
0.9998979729571049
```

ELASTIC NET

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

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

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

```
[0.99917093]
```

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

```
[0.12021556]
```

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

Out[29]:
```
array([1149.16678182,  178.97181152,  101.0364792 ])
```

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

```
0.9999989805752743
```

EVALUATION METRICS

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

In [32]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.2992952881027833
```

In [33]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.23212595506762534
```

In [34]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 0.48179451539803286
```