In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as py
import seaborn as sns
```

In [2]:

```
d=pd.read_csv(r"C:\Users\user\Desktop\salesman.csv")
d
```

Out[2]:

	SALESMAN	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unn
0	ANU	70.0	80.0	75.0	60.0	72.0	55.0	412.0	NaN	NaN	NaN	
1	BABU	30.0	48.0	35.0	45.0	25.0	37.0	220.0	NaN	NaN	NaN	Ind Sales
2	CHANDRU	65.0	54.0	49.0	54.0	35.0	65.0	322.0	NaN	NaN	NaN	2. Fi p cond
3	DAVID	85.0	71.0	68.0	77.0	88.0	73.0	462.0	NaN	NaN	NaN	3. A using ta c perce
4	EINSTEIN	55.0	25.0	45.0	50.0	53.0	30.0	258.0	NaN	NaN	NaN	4.
5	FAROOK	35.0	45.0	15.0	45.0	45.0	25.0	210.0	NaN	NaN	NaN	5. R retu ra v
6	GOWTHAM	75.0	66.0	59.0	65.0	56.0	30.0	351.0	NaN	NaN	NaN	
7	HARSHITH	29.0	35.0	49.0	48.0	35.0	55.0	247.0	NaN	NaN	NaN	
8	INIYAN	35.0	35.0	50.0	59.0	67.0	73.0	319.0	NaN	NaN	NaN	
9	JOHN	77.0	85.0	77.0	68.0	56.0	25.0	388.0	NaN	NaN	NaN	
10	MONTHLY SALES	556.0	544.0	522.0	571.0	532.0	468.0	NaN	3193.0	NaN	NaN	
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3189.0	NaN	NaN	NaN	

In [3]:

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   SALESMAN     11 non-null     object 
 1   JAN          11 non-null     float64
 2   FEB          11 non-null     float64
 3   MAR          11 non-null     float64
 4   APR          11 non-null     float64
 5   MAY          11 non-null     float64
 6   JUN          11 non-null     float64
 7   TOTAL SALES  11 non-null     float64
 8   Unnamed: 8   1 non-null      float64
 9   Unnamed: 9   0 non-null      float64
 10  Unnamed: 10  0 non-null      float64
 11  Unnamed: 11  6 non-null      object 
dtypes: float64(10), object(2)
memory usage: 1.2+ KB
```

In [4]:

```
d.isna()
```

Out[4]:

	SALESMAN	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnam
0	False	False	False	False	False	False	False	False	True	True	True	T
1	False	False	False	False	False	False	False	False	True	True	True	Fa
2	False	False	False	False	False	False	False	False	True	True	True	Fa
3	False	False	False	False	False	False	False	False	True	True	True	Fa
4	False	False	False	False	False	False	False	False	True	True	True	Fa
5	False	False	False	False	False	False	False	False	True	True	True	Fa
6	False	False	False	False	False	False	False	False	True	True	True	T
7	False	False	False	False	False	False	False	False	True	True	True	Fa
8	False	False	False	False	False	False	False	False	True	True	True	T
9	False	False	False	False	False	False	False	False	True	True	True	T
10	False	False	False	False	False	False	False	True	False	True	True	T
11	True	True	True	True	True	True	True	False	True	True	True	T

In [5]:

```
d.fillna(value=0)
```

Out[5]:

	SALESMAN	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unn
0	ANU	70.0	80.0	75.0	60.0	72.0	55.0	412.0	0.0	0.0	0.0	
1	BABU	30.0	48.0	35.0	45.0	25.0	37.0	220.0	0.0	0.0	0.0	Ind Sale

	SALESMAN	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unn
2	CHANDRU	65.0	54.0	49.0	54.0	35.0	65.0	322.0	0.0	0.0	0.0	2. Fi p cond
3	DAVID	85.0	71.0	68.0	77.0	88.0	73.0	462.0	0.0	0.0	0.0	3. A using ta c perce
4	EINSTEIN	55.0	25.0	45.0	50.0	53.0	30.0	258.0	0.0	0.0	0.0	4.
5	FAROOK	35.0	45.0	15.0	45.0	45.0	25.0	210.0	0.0	0.0	0.0	5. R retur ra v
6	GOWTHAM	75.0	66.0	59.0	65.0	56.0	30.0	351.0	0.0	0.0	0.0	
7	HARSHITH	29.0	35.0	49.0	48.0	35.0	55.0	247.0	0.0	0.0	0.0	
8	INIYAN	35.0	35.0	50.0	59.0	67.0	73.0	319.0	0.0	0.0	0.0	
9	JOHN	77.0	85.0	77.0	68.0	56.0	25.0	388.0	0.0	0.0	0.0	
10	MONTHLY SALES	556.0	544.0	522.0	571.0	532.0	468.0	0.0	3193.0	0.0	0.0	
11	0	0.0	0.0	0.0	0.0	0.0	0.0	3189.0	0.0	0.0	0.0	

In [6]:

```
d.describe()
```

Out[6]:

	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES	Unnamed: 8
count	11.000000	11.000000	11.000000	11.000000	11.000000	11.000000	11.000000	1.0
mean	101.090909	98.909091	94.909091	103.818182	96.727273	85.090909	579.818182	3193.0
std	152.263886	148.884153	142.770763	155.277054	145.500578	128.347540	869.142775	NaN
min	29.000000	25.000000	15.000000	45.000000	25.000000	25.000000	210.000000	3193.0
25%	35.000000	40.000000	47.000000	49.000000	40.000000	30.000000	252.500000	3193.0
50%	65.000000	54.000000	50.000000	59.000000	56.000000	55.000000	322.000000	3193.0
75%	76.000000	75.500000	71.500000	66.500000	69.500000	69.000000	400.000000	3193.0
max	556.000000	544.000000	522.000000	571.000000	532.000000	468.000000	3189.000000	3193.0

In [7]:
```
d.columns
```

Out[7]:
```
Index(['SALESMAN', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'TOTAL SALES',
       'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10', 'Unnamed: 11'],
      dtype='object')
```

In [8]:
```
d.index
```

Out[8]:
```
RangeIndex(start=0, stop=12, step=1)
```

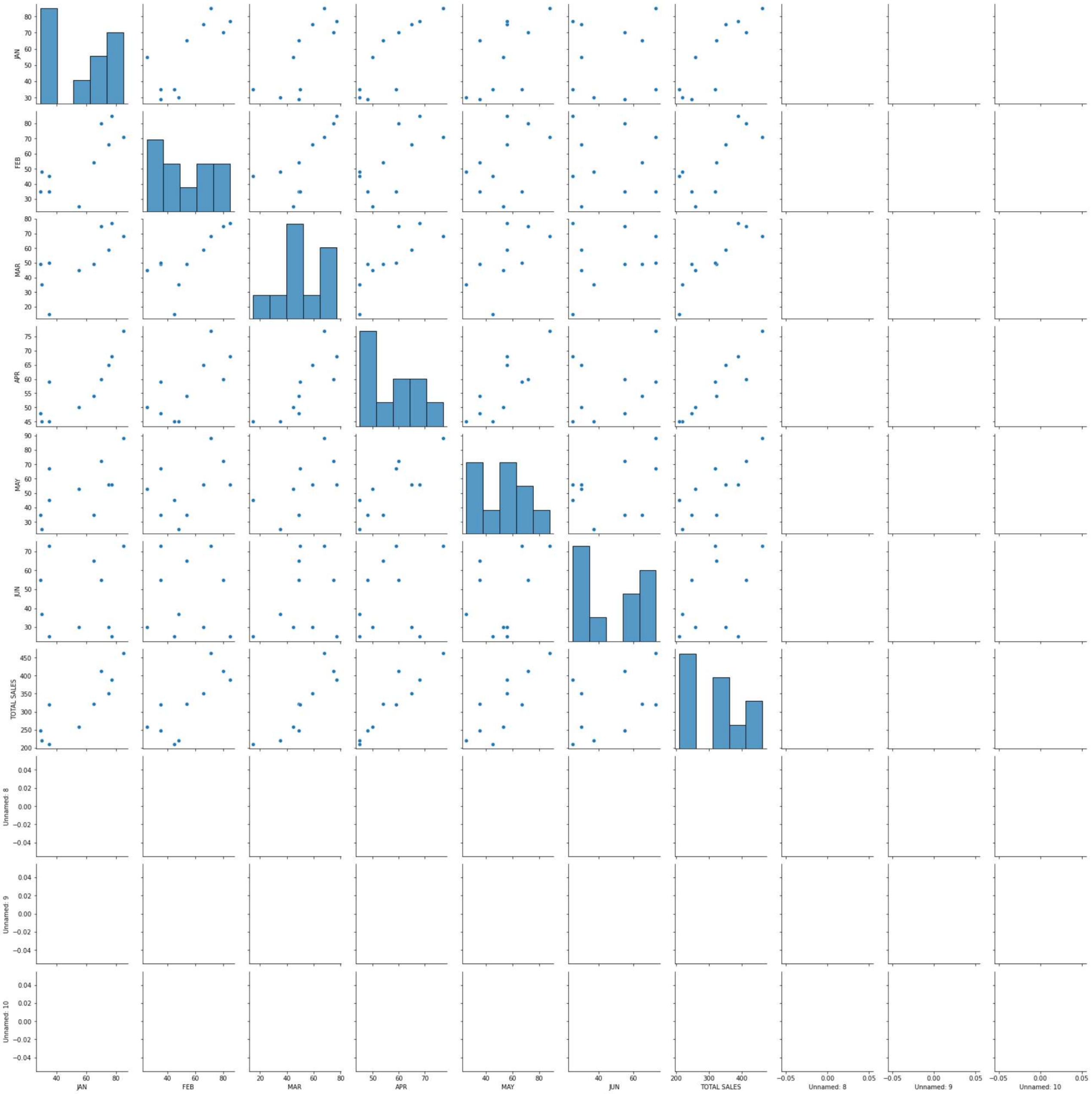
In [9]:
```
d=d.head(10)
d
```

Out[9]:

	SALESMAN	JAN	FEB	MAR	APR	MAY	JUN	TOTAL SALES	Unnamed: 8	Unnamed: 9	Unnamed: 10	Unnamed 11
0	ANU	70.0	80.0	75.0	60.0	72.0	55.0	412.0	NaN	NaN	NaN	NaN
1	BABU	30.0	48.0	35.0	45.0	25.0	37.0	220.0	NaN	NaN	NaN	1 Individua Sales using Sum(
2	CHANDRU	65.0	54.0	49.0	54.0	35.0	65.0	322.0	NaN	NaN	NaN	2. Find the patterr trenc using conditiona fo..
3	DAVID	85.0	71.0	68.0	77.0	88.0	73.0	462.0	NaN	NaN	NaN	3. Analyze using Pivot table as columr percentage
4	EINSTEIN	55.0	25.0	45.0	50.0	53.0	30.0	258.0	NaN	NaN	NaN	4. Insert Pivot charts
5	FAROOK	35.0	45.0	15.0	45.0	45.0	25.0	210.0	NaN	NaN	NaN	5. Rank() - returns the rank of a giver value ..
6	GOWTHAM	75.0	66.0	59.0	65.0	56.0	30.0	351.0	NaN	NaN	NaN	NaN
7	HARSHITH	29.0	35.0	49.0	48.0	35.0	55.0	247.0	NaN	NaN	NaN	33
8	INIYAN	35.0	35.0	50.0	59.0	67.0	73.0	319.0	NaN	NaN	NaN	NaN
9	JOHN	77.0	85.0	77.0	68.0	56.0	25.0	388.0	NaN	NaN	NaN	NaN

In [10]:
```
sns.pairplot(d)
```

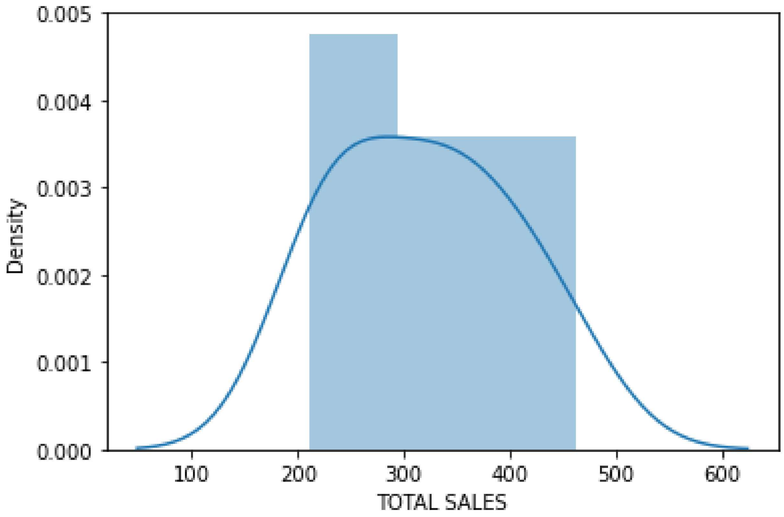
Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x1cf5e25ac70>
```



In [11]:
```python
sns.distplot(d['TOTAL SALES'])
```

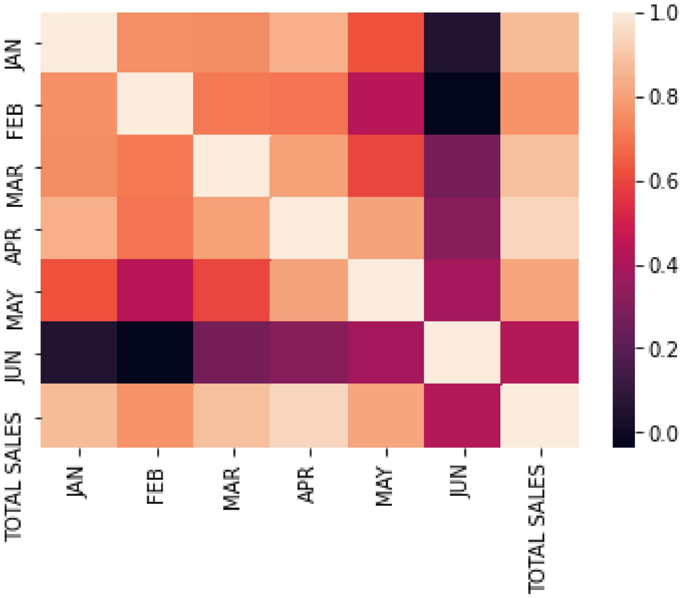
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function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

Out[11]:
```
<AxesSubplot:xlabel='TOTAL SALES', ylabel='Density'>
```



```
In [12]: d1=d[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'TOTAL SALES']]
         sns.heatmap(d1.corr())
```

```
Out[12]: <AxesSubplot:>
```



```
In [13]: x=d1[['JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN']]
         y=d1[ 'TOTAL SALES']
```

```
In [14]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

```
In [15]: from sklearn.linear_model import LinearRegression
```

```
In [16]: lr=LinearRegression()
         lr.fit(x_train,y_train)
```

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

In [17]:

```
print(lr.intercept_)
```

```
-5.613062975537218
```

In [18]:

```
coeff =pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

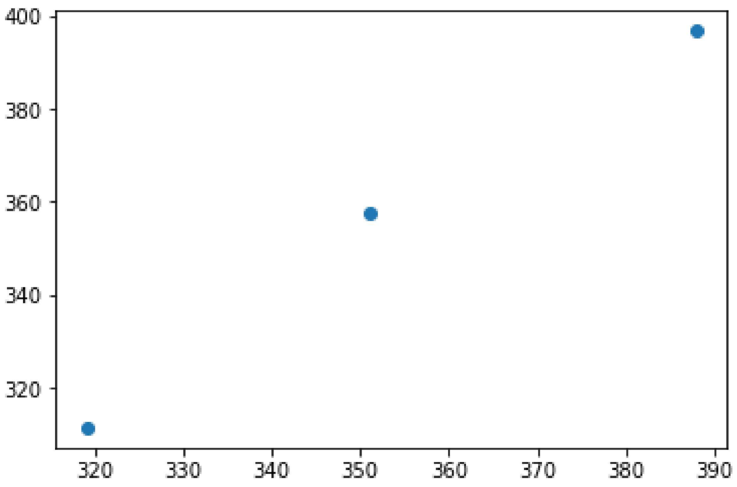
Out[18]:

	Co-efficient
JAN	1.092769
FEB	1.048318
MAR	0.991263
APR	1.200999
MAY	0.887495
JUN	0.853627

In [19]:

```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

```
Out[19]: <matplotlib.collections.PathCollection at 0x1cf643eba30>
```



In [20]:

```
print(lr.score(x_test,y_test))
```

```
0.9278891104256549
```

In [21]:

```
print(lr.score(x_train,y_train))
```

```
1.0
```

Ridge,Lasso

In [22]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [23]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[23]:

```
Ridge(alpha=10)
```

In [24]:

```
rr.score(x_test,y_test)
```

Out[24]:

```
0.9672777756399883
```

In [25]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]:

```
Lasso(alpha=10)
```

In [26]:

```
la.score(x_test,y_test)
```

Out[26]:

```
0.9099345782725198
```

elasticnet regression

In [27]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:

```
ElasticNet()
```

In [28]:

```
print(en.coef_)
```

```
[1.10238852 1.04527613 0.97881937 1.08399162 0.9256584  0.89046154]
```

In [30]:

```
print(en.intercept_)
```

```
-2.745812442670683
```

In [31]:

```
print(en.predict(x_test))
```

```
[312.34172923 355.68206551 394.16550485]
```

In [32]:

```
print(en.score(x_test,y_test))
```

```
0.9562757518364674
```

evaluation metrics

In [34]:

```
from sklearn import metrics
```

Mean Absolute Error

In [41]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 7.513463888824769
```

Mean Squared Error

In [42]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 57.32014489054052
```

Root Mean Squared Error

In [40]:

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

```
Root Mean Squared Error: 7.571006861081327
```

In [ ]: