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("23_Vande Bharat.csv")
df
```

Out[2]:

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City	Termina
0	1	New Delhi - Varanasi Vande Bharat Express	22435/22436	Delhi	New Delhi	Varanasi	Varanas
1	2	New Delhi - Shri Mata Vaishno Devi Katra Vande...	22439/22440	Delhi	New Delhi	Katra	Shri Mat [
2	3	Mumbai Central - Gandhinagar Capital Vande Bha...	20901/20902	Mumbai	Mumbai Central	Gandhinagar	Gandhinag
3	4	New Delhi - Amb Andaura Vande Bharat Express	22447/22448	Delhi	New Delhi	Andaura	Amb
4	5	MGR Chennai Central - Mysuru Vande Bharat Express	20607/20608	Chennai	Chennai Central	Mysuru	Mysor
5	6	Bilaspur - Nagpur Vande Bharat Express	20825/20826	Bilaspur, Chhattisgarh	Bilaspur Junction	Nagpur	Nagpu
6	7	Howrah - New Jalpaiguri Vande Bharat Express	22301/22302	Kolkata	Howrah Junction	Siliguri	New
7	8	Visakhapatnam - Secunderabad Vande Bharat Express	20833/20834	Visakhapatnam	Visakhapatnam Junction	Hyderabad	Secu
8	9	Mumbai CSMT - Solapur Vande Bharat Express	22225/22226	Mumbai	Chhatrapati Shivaji Terminus	Solapur	
9	10	Mumbai CSMT - Sainagar Shirdi Vande Bharat Exp...	22223/22224	Mumbai	Chhatrapati Shivaji Terminus	Shirdi	Saina
10	11	Rani Kamalapati (Habibganj) - Hazrat Nizamuddi...	20171/20172	Bhopal	Habibganj (Rani Kamalapati)	Delhi	Hazrat Ni:

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City	Termina
11	12	Secunderabad - Tirupati Vande Bharat Express	20701/20702	Hyderabad	Secunderabad Junction	Tirupati	
12	13	MGR Chennai Central - Coimbatore Vande Bharat ...	20643/20644	Chennai	Chennai Central	Coimbatore	Coimbatore
13	14	Delhi Cantonment - Ajmer Vande Bharat Express	20977/20978	Delhi	Delhi Cantonment	Ajmer	Ajme
14	15	Kasaragod - Thiruvananthapuram Vande Bharat Ex...	20633/20634	Kasaragod	Kasaragod	Thiruvananthapuram	Thiruvanan
15	16	Howrah - Puri Vande Bharat Express	22895/22896	Kolkata	Howrah Junction	Puri	
16	17	Anand Vihar Terminal - Dehradun Vande Bharat E...	22457/22458	Delhi	Anand Vihar Terminal	Dehradun	Dehradur
17	18	New Jalpaiguri - Guwahati Vande Bharat Express	22227/22228	Siliguri	New Jalpaiguri Junction	Guwahati	
18	19	Mumbai CSMT - Madgaon Vande Bharat Express	22229/22230	Mumbai	Chhatrapati Shivaji Terminus	Madgaon	Madgaor
19	19	Mumbai CSMT - Madgaon Vande Bharat Express	22229/22230	Mumbai	Chhatrapati Shivaji Terminus	Madgaon	Madgaor
20	20	Patna - Ranchi Vande Bharat Express	22349/22350	Patna	Patna Junction	Ranchi	Ranch
21	21	KSR Bengaluru - Dharwad Vande Bharat Express	20661/20662	Bangalore	Bangalore City	Hubbali - Dharwad	
22	22	Rani Kamalapati (Habibganj) - Jabalpur Vande B...	20173/20174	Bhopal	Habibganj (Rani Kamalapati)	Jabalpur	Jabalpu
23	23	Indore - Bhopal Vande Bharat Express	20911/20912	Indore	Indore Junction	Bhopal	Bhopa
24	24	Jodhpur - Sabarmati (Ahmedabad) Vande Bharat E...	12461/12462	Jodhpur	Jodhpur Junction	Ahmedabad	Sabarmat
25	25	Gorakhpur - Lucknow Charbagh	22549/22550	Gorakhpur	Gorakhpur Junction	Charbagh	Lucknow

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City	Termin
		Vande Bharat Express					

In [3]:

```
df.head()
```

Out[3]:

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City	Terminal Station	Operator	No. of Cars
0	1	New Delhi - Varanasi Vande Bharat Express	22435/22436	Delhi	New Delhi	Varanasi	Varanasi Junction	NR	16
1	2	New Delhi - Shri Mata Vaishno Devi Katra Vande...	22439/22440	Delhi	New Delhi	Katra	Shri Mata Vaishno Devi Katra	NR	16
2	3	Mumbai Central - Gandhinagar Capital Vande Bha...	20901/20902	Mumbai	Mumbai Central	Gandhinagar	Gandhinagar Capital	WR	16
3	4	New Delhi - Amb Andaura Vande Bharat Express	22447/22448	Delhi	New Delhi	Andaura	Amb Andaura	NR	16
4	5	MGR Chennai Central - Mysuru Vande Bharat Express	20607/20608	Chennai	Chennai Central	Mysuru	Mysore Junction	SR	16

Data Cleaning and Data Preprocessing

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
```

```
 0   Sr. No.              26 non-null     int64 
 1   Train Name           26 non-null     object
 2   Train Number         26 non-null     object
 3   Originating City     26 non-null     object
 4   Originating Station  26 non-null     object
 5   Terminal City        26 non-null     object
 6   Terminal Station     26 non-null     object
 7   Operator             26 non-null     object
 8   No. of Cars          26 non-null     int64 
 9   Frequency            26 non-null     object
 10  Distance             26 non-null     object
 11  Travel Time          26 non-null     object
 12  Speed                26 non-null     object
 13  Average Speed        26 non-null     object
 14  Inauguration         26 non-null     object
 15  Average occupancy    26 non-null     object
dtypes: int64(2), object(14)
memory usage: 3.4+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	Sr. No.	No. of Cars
count	26.000000	26.000000
mean	13.230769	12.923077
std	7.306478	3.969112
min	1.000000	8.000000
25%	7.250000	8.000000
50%	13.500000	16.000000
75%	19.000000	16.000000
max	25.000000	16.000000

In [6]:

```
df.columns
```

Out[6]:

```
Index(['Sr. No.', 'Train Name', 'Train Number', 'Originating City',
       'Originating Station', 'Terminal City', 'Terminal Station', 'Operator',
       'No. of Cars', 'Frequency', 'Distance', 'Travel Time', 'Speed',
       'Average Speed', 'Inauguration', 'Average occupancy'],
      dtype='object')
```

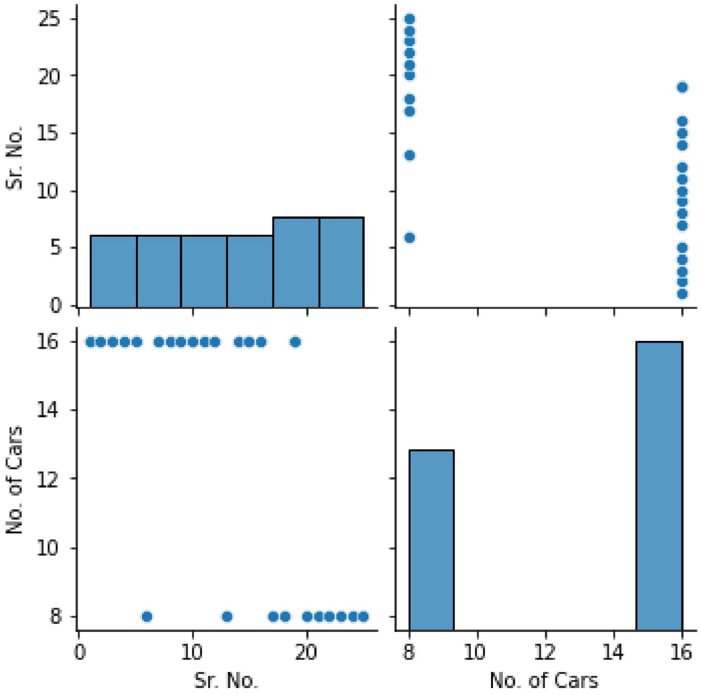
EDA and Visualization

In [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x21263ae24f0>
```



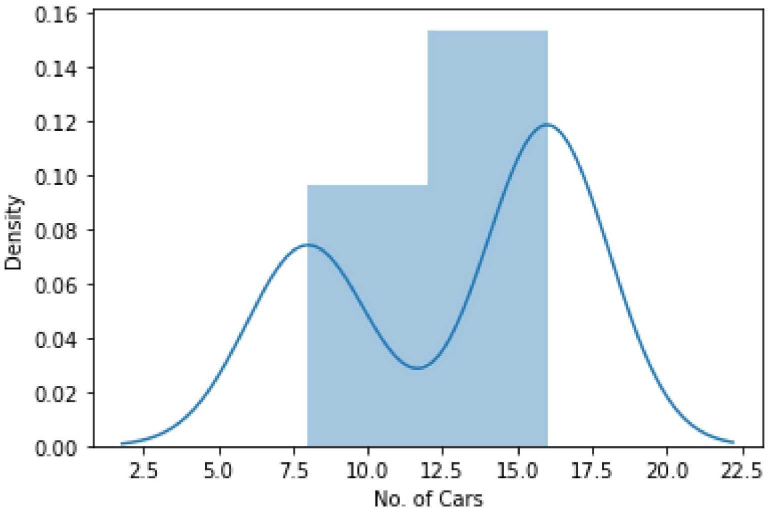
In [8]:

```
sns.distplot(df['No. of Cars'])
```

```
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[8]:

```
<AxesSubplot:xlabel='No. of Cars', ylabel='Density'>
```



In [9]:

```
df1=df[['Sr. No.','No. of Cars']]
df1
```

Out[9]:

	Sr. No.	No. of Cars
0	1	16
1	2	16

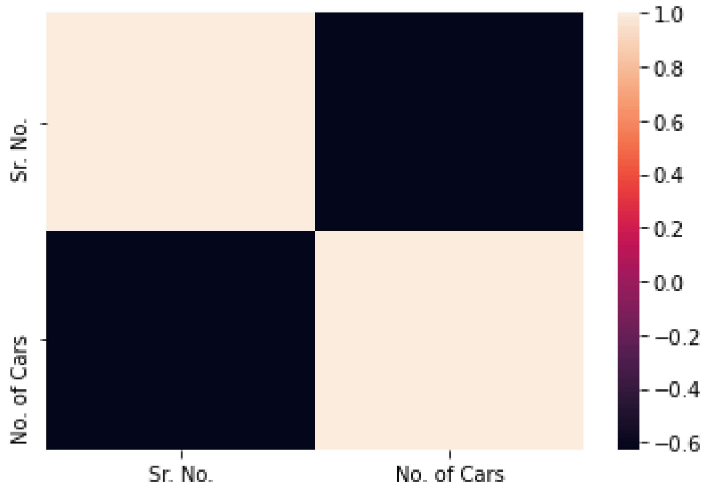
	Sr. No.	No. of Cars
2	3	16
3	4	16
4	5	16
5	6	8
6	7	16
7	8	16
8	9	16
9	10	16
10	11	16
11	12	16
12	13	8
13	14	16
14	15	16
15	16	16
16	17	8
17	18	8
18	19	16
19	19	16
20	20	8
21	21	8
22	22	8
23	23	8
24	24	8
25	25	8

In [10]:

```
sns.heatmap(df1.corr())
```

Out[10]:

```
<AxesSubplot:>
```



To Train the Model -Model Building

We are going to train Linear Regression model;We need to spilt out data into two variables x and y where x is independent variable (input) and y is dependent variable on x(output) we could ignore address column as it is not requiired for our model

In [11]:
```
x=df1[['Sr. No.','Sr. No.']]
y=df1['No. of Cars']
```

In [12]:
```
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 [13]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

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

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

```
19.24557218246223
```

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

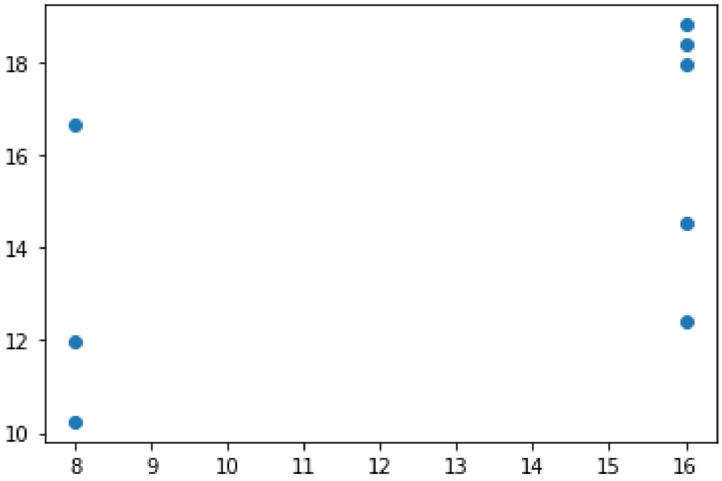
Out[15]:

	Co-efficient
Sr. No.	-0.21427
Sr. No.	-0.21427

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

Out[16]:

```
<matplotlib.collections.PathCollection at 0x21265d7b730>
```



In [17]:

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

Out[17]:

```
-0.07210084906277947
```

In [18]:

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

Out[18]:

```
0.5155007646874922
```

In [19]:

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

In [20]:

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

Out[20]:

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

In [21]:

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

Out[21]:

```
-0.0654196319050504
```

In [22]:

```
rr.score(x_train,y_train)
```

Out[22]:

```
0.5154792245393192
```

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]: 0.2068989365102809
```

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

```
Out[25]: 0.3615070680783108
```

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

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

```
In [27]: en.coef_
```

```
Out[27]: array([-0.20808219, -0.20633077])
```

```
In [28]: en.intercept_
```

```
Out[28]: 19.03601439808896
```

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

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

```
Out[30]: -0.03887339232066078
```

Evaluation Metrics

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

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

```
Mean Absolute Error: 3.326569951474803
```

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

```
Mean Squared Error: 15.583100884809912
```

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

```
Root Mean Squared Error: 3.947543652046157
```

Model Saving

In [35]:

```
import pickle
```

In [36]:

```
filename="prediction"
pickle.dump(lr,open(filename,'wb'))
```

In [37]:

```
import pandas as pd
import pickle
```

In [38]:

```
filename="prediction"
model=pickle.load(open(filename,'rb'))
```

In [39]:

```
real=[[2,2],[5,5]]
result=model.predict(real)
```

In [40]:

```
result
```

Out[40]:

```
array([18.38849129, 17.10286995])
```