In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df=pd.read_csv("insurance (1).csv")
df
```

Out[2]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520
...	...	...	...	...	...	...	...
1333	50	male	30.970	3	no	northwest	10600.54830
1334	18	female	31.920	0	no	northeast	2205.98080
1335	18	female	36.850	0	no	southeast	1629.83350
1336	21	female	25.800	0	no	southwest	2007.94500
1337	61	female	29.070	0	yes	northwest	29141.36030

1338 rows × 7 columns

In [3]:
```
df.shape
```

Out[3]:
```
(1338, 7)
```

In [4]:
```
df.describe()
```

Out[4]:

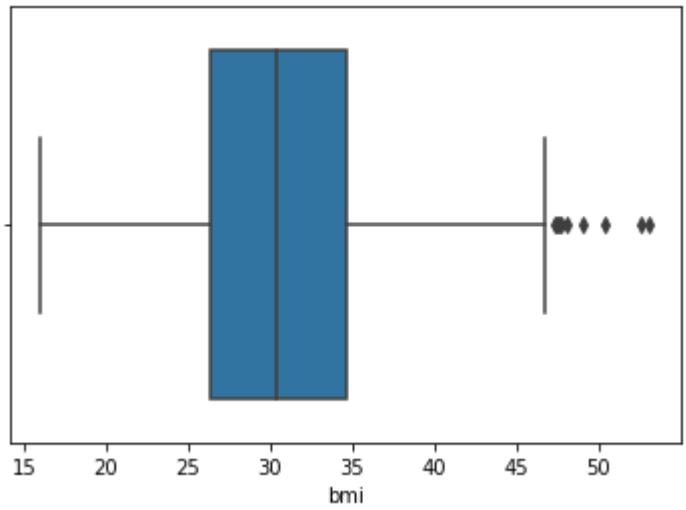
	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
age         0
sex         0
bmi         0
children    0
smoker      0
region      0
charges     0
dtype: int64
```

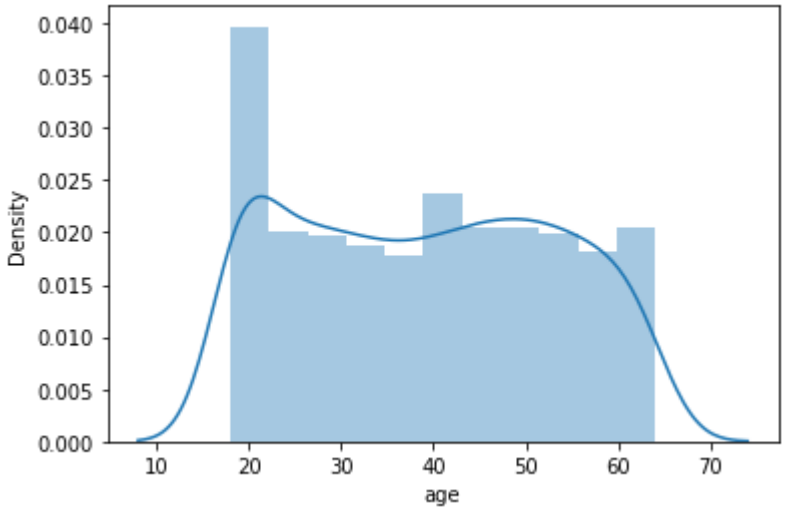
In [6]:
```
sns.boxplot(df.bmi)
```

Out[6]:
```
<AxesSubplot:xlabel='bmi'>
```



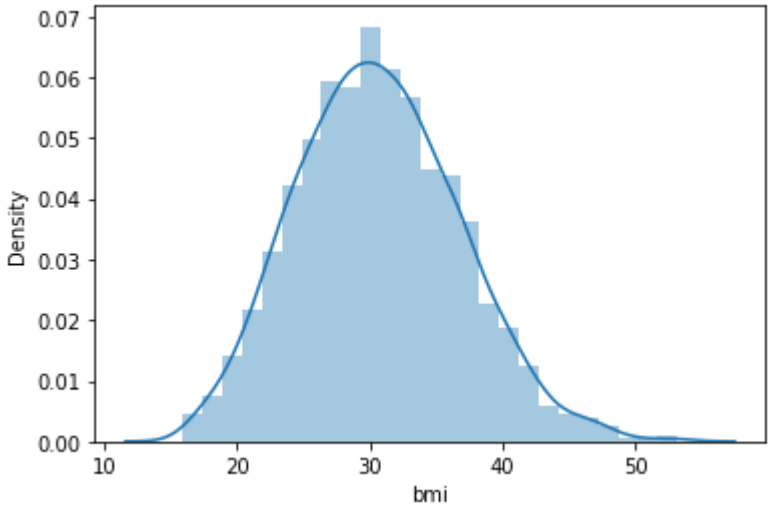
In [11]:
```
sns.distplot(df.age)
print("skewness",df.age.skew())
```

```
skewness 0.05567251565299186
```



In [18]:
```
sns.distplot(df.bmi)
df.bmi.mean()
```

Out[18]:
```
30.66339686098655
```

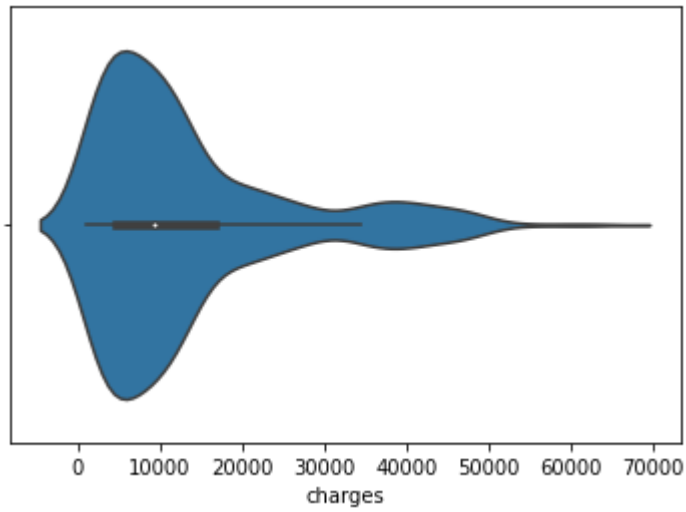


In [16]:
```
df.bmi.median()
```

Out[16]:
```
30.4
```

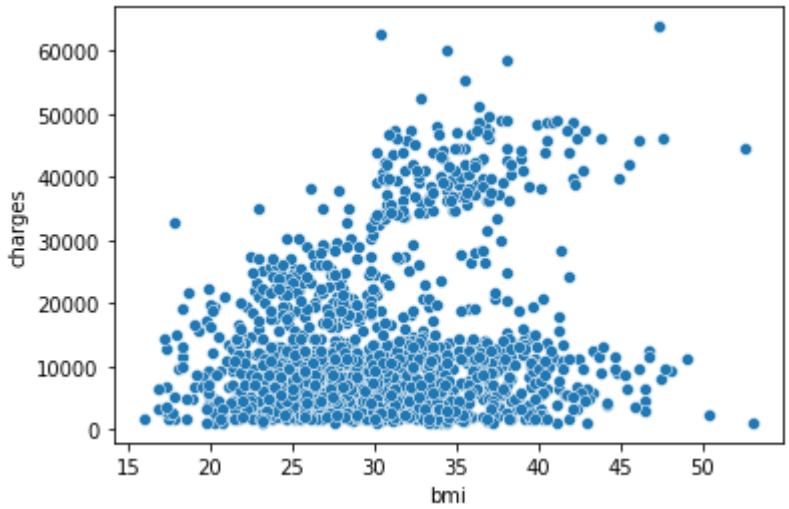
In [19]:
```
sns.violinplot(df.charges)
```

Out[19]:
```
<AxesSubplot:xlabel='charges'>
```



In [20]:
```
sns.scatterplot(x='bmi',y='charges',data=df)
```

Out[20]:
```
<AxesSubplot:xlabel='bmi', ylabel='charges'>
```



BMI for males and females

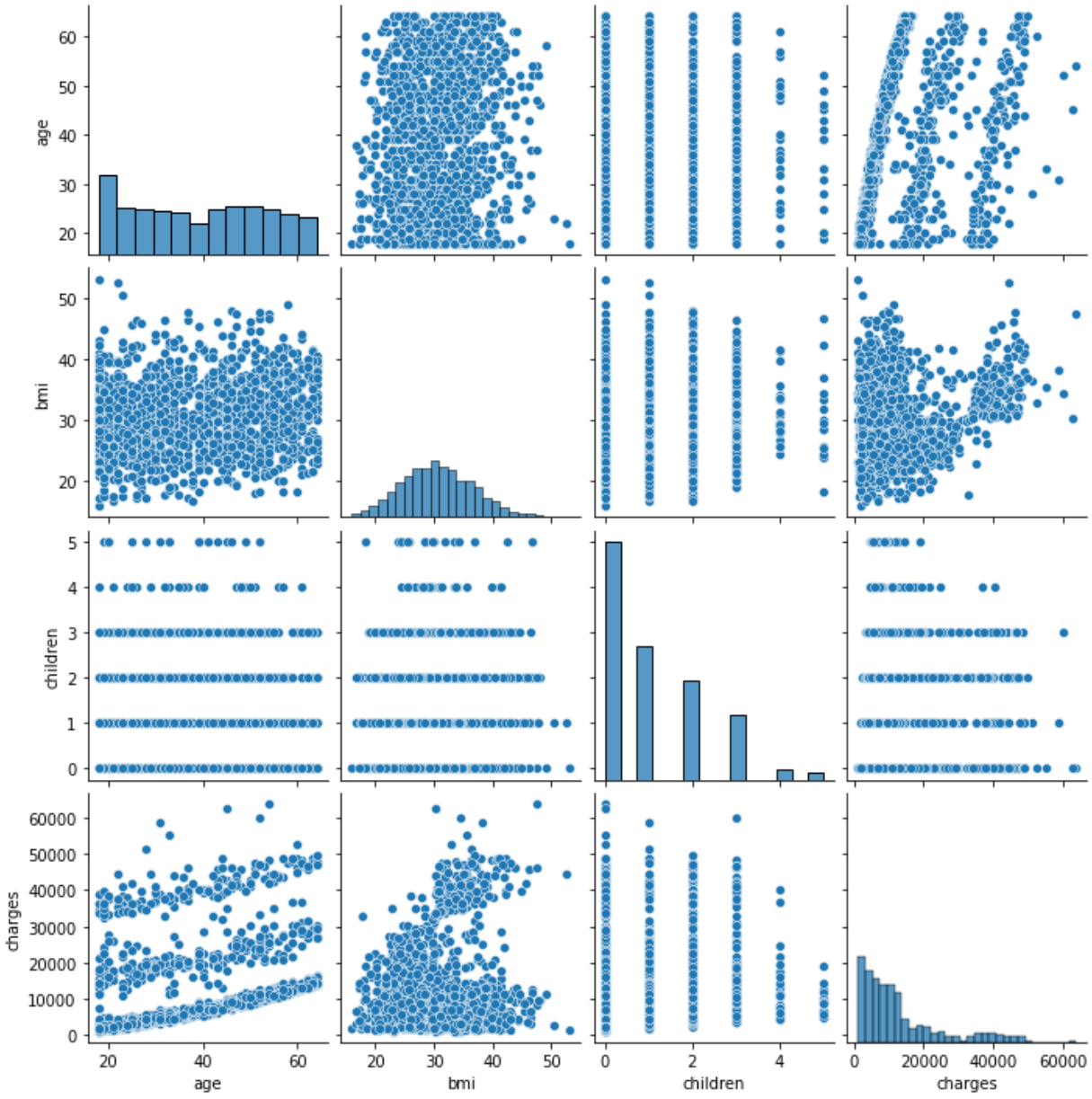
In [23]:
```
df.groupby(['sex'])['bmi'].sum()
```

Out[23]:
```
sex
female    20110.070
male      20917.555
Name: bmi, dtype: float64
```

In [26]:
```
num_col=df.select_dtypes(include=['int','float'])
```

In [27]:
```
sns.pairplot(data=num_col)
```

Out[27]:
```
<seaborn.axisgrid.PairGrid at 0x1df04d51b70>
```



In [28]:
```
corr=num_col.corr()
```

In [29]:
```
sns.heatmap(corr,annot=True)
```

Out[29]: <AxesSubplot:>

`Handling Categorical data

In [30]:
```
cat_col=df.select_dtypes(include=['object'])
```

In [31]:
```
cat_col
```

Out[31]:

	sex	smoker	region
0	female	yes	southwest
1	male	no	southeast
2	male	no	southeast
3	male	no	northwest
4	male	no	northwest
...	...	...	...
1333	male	no	northwest
1334	female	no	northeast
1335	female	no	southeast
1336	female	no	southwest
1337	female	yes	northwest

1338 rows × 3 columns

In [32]:
```
from sklearn.preprocessing import LabelEncoder
```

In [33]:
```
le=LabelEncoder()
```

In [34]:
```
for col in cat_col:
    le=LabelEncoder()
    df[col]=le.fit_transform(df[col])
```

In [35]:
```
df.head()
```

Out[35]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520

In [37]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [38]:
```
min_max=MinMaxScaler()
```

In [39]:
```
min_max.fit_transform(df[['bmi']])
```

Out[39]:
```
array([[0.3212268 ],
       [0.47914985],
       [0.45843422],
       ...,
       [0.56201238],
       [0.26472962],
       [0.35270379]])
```

In [40]:
```
from sklearn.preprocessing import StandardScaler
```

In [41]:
```
ss=StandardScaler()
```

In [42]:
```
ss.fit_transform(df[['bmi']])
```

Out[42]:
```
array([[-0.45332   ],
       [ 0.5096211 ],
       [ 0.38330685],
       ...,
       [ 1.0148781 ],
       [-0.79781341],
       [-0.26138796]])
```

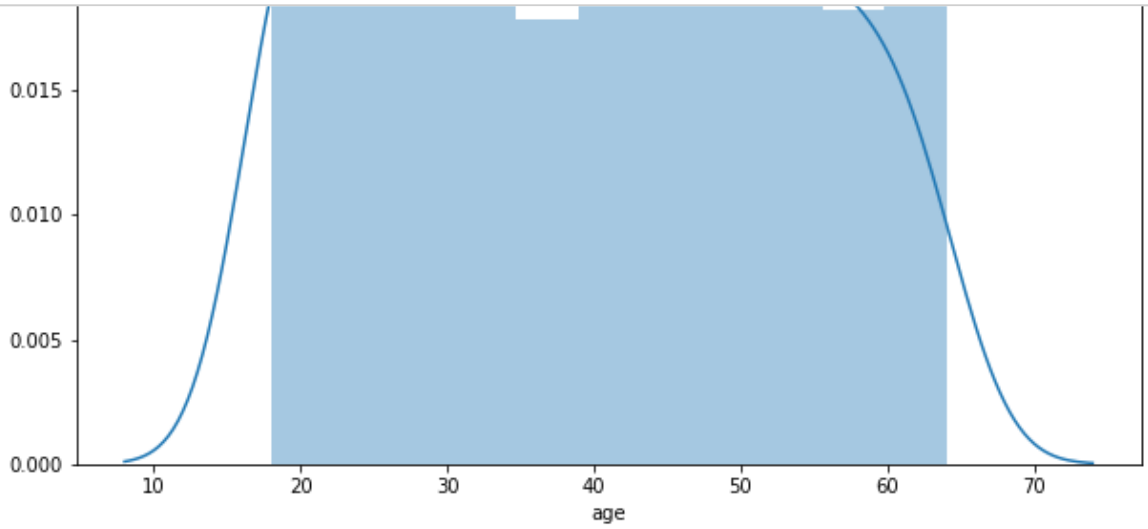
In [43]:
```
df
```

Out[43]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	0	1	10600.54830
1334	18	0	31.920	0	0	0	2205.98080
1335	18	0	36.850	0	0	2	1629.83350
1336	21	0	25.800	0	0	3	2007.94500
1337	61	0	29.070	0	1	1	29141.36030

1338 rows × 7 columns

In [44]:
```
for col in num_col:
    print(col)
    print('skewness',df[col].skew())
    print('kurtosis',df[col].kurt())
    plt.figure(figsize=(10,10))
    sns.distplot(df[col])
    plt.show()
```




```
bmi
skewness 0.2840471105987448
kurtosis -0.05073153135467834
```

In [45]:
```
x=df.iloc[:,:-1]#Training data
y=df.iloc[:,-1]#Testing data
```

In [46]:
```
x
```

Out[46]:

	age	sex	bmi	children	smoker	region
0	19	0	27.900	0	1	3
1	18	1	33.770	1	0	2
2	28	1	33.000	3	0	2
3	33	1	22.705	0	0	1
4	32	1	28.880	0	0	1
...	...	...	...	...	...	...
1333	50	1	30.970	3	0	1
1334	18	0	31.920	0	0	0
1335	18	0	36.850	0	0	2
1336	21	0	25.800	0	0	3
1337	61	0	29.070	0	1	1

1338 rows × 6 columns

In [47]:
```
y
```

Out[47]:
```
0       16884.92400
1        1725.55230
2        4449.46200
3       21984.47061
4        3866.85520
           ...
1333    10600.54830
1334     2205.98080
1335     1629.83350
1336     2007.94500
1337    29141.36030
Name: charges, Length: 1338, dtype: float64
```

splitting data

In [48]:
```
from sklearn.model_selection import train_test_split
```

In [87]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=123
```

In [88]:
```
x_train.shape
```

Out[88]: (936, 6)

In [89]:
```
y_train.shape
```

Out[89]: (936,)

In [90]:
```
y_test.shape
```

Out[90]: (402,)

In [91]:
```
x_test.shape
```

Out[91]: (402, 6)

Model Building

In [92]:
```
from sklearn.linear_model import LinearRegression
```

In [93]:
```
reg=LinearRegression()
```

In [94]:
```
reg.fit(x_train,y_train)
```

Out[94]:
```
▾ LinearRegression

LinearRegression()
```

In [95]:
```
y_train_pred=reg.predict(x_train)
y_test_pred=reg.predict(x_test)
```

In [96]:
```
y_train_pred
```

Out[96]:
```
array([ 8.14259855e+03,  1.44168619e+04,  7.19982210e+03,  3.28854161e+04,
        2.94610576e+04,  1.04383624e+04,  3.17952432e+04,  7.86558258e+03,
        3.88561982e+04,  1.02715278e+04,  4.25579616e+03,  7.35158232e+03,
        7.02951770e+03,  1.00499786e+04,  5.57487564e+03,  3.78743167e+04,
        3.70483135e+04,  1.32132015e+04,  1.49220421e+04,  2.61804889e+04,
        1.04762009e+04,  1.11006090e+04,  1.06094913e+04,  8.64186275e+03,
        3.75294316e+04,  1.76231376e+04,  1.63425435e+04,  3.91902128e+04,
        5.02438054e+03,  5.79430522e+03,  5.12243288e+03,  1.21390583e+04,
        2.29349955e+04,  1.77119121e+03,  6.85939923e+03,  3.01274749e+04,
        1.20099602e+04,  1.04651895e+04,  9.75434388e+03,  2.49476358e+04,
        9.91246384e+03,  1.38667638e+04,  8.63380105e+02,  1.70674866e+03,
        1.37601982e+04,  1.07485625e+04,  1.15694035e+04,  1.06739405e+04,
        6.08206166e+03,  1.02482210e+04,  8.16692564e+03,  9.21591756e+03,
        6.91477038e+03,  7.87067780e+03,  5.85496101e+03,  9.99147365e+03,
        9.20926898e+03,  1.13498902e+04,  3.88131556e+03,  3.45246685e+04,
        7.29280792e+03,  4.18254175e+03,  9.17976421e+03,  1.50257276e+04,
        3.93093866e+04,  1.65143394e+04,  2.98790702e+04,  1.50850951e+04,
        7.46120896e+03,  3.47287170e+04,  3.72705966e+04,  3.35430651e+04,
        1.16192690e+04,  8.95128026e+03,  6.38458070e+02,  1.28701172e+04,
```

In [97]:
```
y_test_pred
```

Out[97]:
```
array([ 1.55319667e+04,  9.51619844e+03,  2.83341303e+04,  5.36165725e+03,
        1.14977191e+04,  1.12770325e+04,  2.84818292e+03,  2.22248819e+03,
        4.45419112e+03,  8.81101145e+03,  8.71714119e+03,  1.35995111e+04,
        1.33444996e+04,  2.99293763e+04,  1.37720291e+04,  3.10510278e+04,
        1.50832570e+04,  1.72526684e+03,  3.36403616e+04,  3.67784399e+04,
        3.34411743e+04,  4.11934013e+04,  3.17252957e+03,  1.01712572e+04,
       -5.35325945e+01,  9.59185567e+03,  1.64327524e+04,  6.57095072e+03,
        1.47513939e+04, -5.19679604e+02,  9.28219974e+03,  1.64996937e+04,
        5.29474627e+03,  3.76431146e+03,  6.93573633e+03,  1.14603044e+04,
        1.77348674e+04,  7.02628562e+03,  9.83898680e+03,  1.16517139e+04,
        7.12601669e+03,  7.95848427e+03,  5.41639869e+03,  3.18049796e+04,
        7.37291174e+03,  2.38865923e+03,  1.31400244e+04,  1.39119932e+04,
        1.46154279e+04,  9.43537877e+03,  1.28432468e+04,  1.11711223e+04,
        2.73020882e+04,  1.15288421e+04,  1.42796256e+04,  6.31439797e+03,
        3.19993033e+04,  2.82742869e+04,  3.85495351e+04,  6.55096839e+03,
        1.64136464e+04,  9.57496936e+03,  3.84240192e+04,  2.84782849e+04,
        1.75739994e+03,  4.26485403e+03,  1.78157402e+04,  5.71331001e+03,
        1.55985407e+04,  1.48959207e+04,  1.08323710e+04,  1.16182336e+04,
        1.63181998e+04,  1.68925411e+04,  1.15318732e+04,  3.19025444e+04,
        4.12919789e+03,  1.45816675e+04,  5.67099084e+03,  3.11500552e+04,
        2.01618635e+03,  3.11300110e+04,  2.80499493e+04,  1.53960828e+04,
        2.55145446e+03,  1.64468305e+04,  2.91236269e+04,  1.32726759e+04,
        1.04970262e+04,  3.91850034e+04,  9.76819203e+03,  9.30734124e+03,
        8.73865648e+03,  6.27868550e+03, -7.55691365e+01,  2.70515287e+03,
        6.32895650e+03,  8.20785574e+03,  4.14838122e+03,  3.90290499e+03,
        1.13979293e+04,  8.84075362e+03,  1.33281031e+04,  6.52670837e+03,
        7.53622608e+03,  3.43810181e+03,  2.76388992e+04,  8.35126293e+03,
        1.35949601e+04,  8.61353611e+03,  4.54037386e+03,  3.73443835e+04,
        3.16514992e+04,  1.38079406e+04,  3.11290203e+04,  1.35403313e+04,
        7.95451984e+03,  9.47231094e+03,  1.30148719e+04,  7.36564336e+03,
        3.83616842e+04,  2.67319761e+04,  5.35488794e+03,  1.15002556e+04,
        5.60979051e+03,  4.12484655e+03,  3.30418557e+04,  7.58684918e+03,
        3.74427474e+04,  3.67687976e+04,  4.50612920e+03,  6.30232908e+03,
       -2.87671664e+02,  1.45191650e+04,  9.38679336e+03,  6.90663700e+03,
        6.22918881e+03,  5.86783454e+03,  4.06554105e+03,  1.37937875e+04,
        4.73126922e+03,  2.38285347e+04,  2.78627153e+04,  8.72213751e+03,
        2.58279899e+04,  9.59611791e+03,  3.03316405e+04,  7.17066802e+03,
        8.43305395e+03,  2.09817518e+03,  3.27086839e+03,  1.26957328e+04,
        2.61781984e+04,  1.17891682e+04,  3.47809859e+03,  5.49560114e+03,
        3.75241902e+04,  1.11596733e+04,  1.35438013e+04,  5.87260471e+03,
        5.81770770e+03,  1.05487716e+04,  8.93170315e+03,  3.63710003e+04,
        3.04315825e+03,  3.85215143e+03,  1.18514795e+04,  2.86387039e+04,
        2.70018427e+04,  1.07802734e+04,  5.70642357e+03,  3.00812469e+04,
        1.21957550e+04,  1.45541785e+04,  1.17352598e+04,  5.64701752e+03,
        1.84153532e+03,  9.94003315e+03,  1.29296072e+04,  7.39543611e+03,
        7.65280796e+03,  1.17432700e+04,  9.94115748e+03,  4.10483896e+03,
        1.19659554e+04,  7.30769482e+03,  1.40970913e+04,  4.13228080e+03,
        8.03043840e+03,  2.88674077e+04,  3.17002102e+04,  1.35369887e+04,
        3.39015307e+04,  1.06213223e+03,  1.03526887e+04,  8.09615765e+03,
        2.32273879e+04,  9.25822986e+03,  2.97393237e+04,  1.27214707e+04,
        3.90722165e+04,  1.36230439e+04, -2.99644103e+01,  7.47724058e+03,
        4.28091561e+03,  2.73581031e+04,  3.48158698e+04,  3.89389329e+04,
        3.09052244e+03,  2.93929193e+03,  9.98698836e+03,  1.15692876e+03,
        1.26047380e+04,  1.00300914e+04,  1.24958198e+04,  3.86518251e+04,
```

```
       1.23996241e+04,  6.88255656e+03,  2.71786300e+04,  1.29194602e+04,
       2.70165538e+04,  7.10766927e+03,  7.33134850e+03,  1.44750428e+04,
       2.77779582e+04,  3.76285395e+03,  7.31137442e+03,  5.65445014e+02,
       1.74769841e+04,  1.70913965e+04,  9.52736731e+03,  3.40518588e+03,
       6.18511704e+03,  2.79176393e+04,  9.63166610e+03,  1.13832831e+04,
       5.37366557e+03,  1.56342431e+04,  3.42469370e+04,  3.35410094e+04,
       8.99690168e+03,  1.00333103e+04,  1.32924073e+04,  8.24691066e+03,
       1.12495221e+04,  1.03973552e+04,  1.48425833e+04,  3.25539489e+03,
       3.58806200e+03,  3.22130727e+04,  1.16420007e+04,  9.63386122e+03,
       8.50238635e+03,  1.70087844e+04,  1.19765818e+04,  4.77634443e+03,
       3.12379537e+04,  1.58230023e+04,  1.28070722e+04,  1.18455412e+04,
       3.31769339e+04,  5.54078523e+03,  3.98884514e+04,  8.70147643e+03,
       5.79285749e+03,  1.02641939e+04,  5.21058590e+03,  2.50289711e+04,
       7.40295581e+03,  3.37059691e+04,  8.82486459e+03,  9.77548060e+03,
      -1.25183968e+03,  1.14666295e+04,  9.18905376e+03,  9.16535474e+03,
       2.68829516e+04,  1.23609424e+04,  3.24888257e+04,  9.78281450e+03,
       1.12301545e+04,  3.76706271e+03,  3.17313066e+04,  9.96038934e+03,
       8.63370438e+03,  1.18157339e+04,  3.29555778e+04,  1.23677871e+04,
       1.53433236e+04,  5.66856492e+03,  3.19460949e+04,  3.53655239e+04,
       1.08653539e+04,  9.61011205e+03,  1.61224852e+04,  3.68227046e+04,
       2.55904946e+04,  1.00784976e+04,  1.59919112e+04,  7.55163584e+03,
       2.60059837e+04, -1.92441073e+03,  1.13605277e+04,  3.03596335e+04,
       1.12279593e+04,  1.00198491e+04,  1.69164038e+04,  3.40160568e+04,
       3.29197564e+04,  6.29369987e+03,  2.74216779e+04,  2.61558021e+03,
       8.30719989e+03,  2.93228025e+04,  1.25421749e+04,  2.78595452e+04,
       8.57046993e+03,  2.31952511e+04,  9.96794609e+03,  1.04089920e+04,
       1.42784508e+04,  7.89470163e+03,  9.21546664e+03,  1.32020577e+04,
       5.48826723e+03,  1.26750736e+04,  1.64830856e+04,  1.00153872e+04,
       1.55638486e+04,  1.57537691e+04,  1.31452956e+04,  1.45713142e+04,
       1.38469288e+04,  1.01636814e+03,  7.18344205e+03,  2.85892454e+04,
       5.36476201e+03,  3.22051578e+04,  4.16181223e+03,  1.86333759e+04,
       3.63884713e+04,  1.18096382e+04,  5.33764195e+03,  6.22772750e+03,
       1.48084929e+04,  6.18215813e+02,  9.91435021e+03,  1.19284672e+04,
       4.57998857e+03,  1.27685313e+04,  7.73746093e+03,  2.98450936e+04,
       1.61146659e+04,  7.08855680e+03,  6.78557149e+03,  1.05191398e+04,
       3.86806571e+03,  1.91742185e+03,  4.66380723e+03,  2.83632844e+04,
       3.21561348e+04,  7.03715628e+03,  7.67263787e+03,  1.31062757e+04,
       3.03178154e+03,  6.64696133e+03,  7.35869825e+03,  1.10985059e+04,
       9.39670439e+03,  1.09662730e+04,  7.98811625e+03,  4.50535011e+03,
       1.01465310e+04,  3.38432888e+03,  2.66717036e+04,  3.70822566e+04,
       2.90196249e+04,  7.00951038e+03,  2.72013736e+04,  6.40787940e+03,
       9.55985936e+03,  2.60389015e+03,  4.48312769e+03,  1.28364091e+04,
       1.57051610e+03,  1.81553709e+04,  6.50138072e+03,  1.32231656e+04,
       4.10497994e+04,  1.28352915e+04,  2.07569283e+03,  1.67172249e+04,
       9.30750559e+03,  1.61305249e+04,  1.07594164e+04,  3.50833178e+03,
       3.58727811e+03,  7.03140088e+03,  5.69505670e+03,  2.80961794e+04,
       7.25487388e+03,  4.92622786e+03])
```

In [98]:
```
reg.intercept_
```

Out[98]:
```
-11966.536839327007
```

In [99]:
```
reg.coef_
```

Out[99]:
```
array([  258.01052077,   -89.42500591,   343.81965818,   479.31668732,
       23703.74833645,  -388.43417176])
```

Find R2 Score

In [100]:
```
from sklearn.metrics import r2_score,mean_squared_error
```

In [101]:
```
train=r2_score(y_train,y_train_pred)
```

In [102]:
```
test=r2_score(y_test,y_test_pred)
```

In [103]:
```
train
```

Out[103]:
```
0.7454618032357492
```

In [104]:
```
test
```

Out[104]:
```
0.7626072475418817
```

In [105]:
```
def metric(y_actual,y_pred):
    r2=r2_score(y_actual,y_pred)
    RMSE=np.sqrt(mean_squared_error(y_actual,y_pred))
    print("r2 score: {} | RMSE: {} ".format(round(r2,2),round(RMSE,2)))
```

In [106]:
```
print("Training performance")
metric(y_train,y_train_pred)
print("Testing performance")
metric(y_test,y_test_pred)
```

```
Training performance
r2 score: 0.75 | RMSE: 6140.17
Testing performance
r2 score: 0.76 | RMSE: 5823.57
```

In [ ]:

In [ ]: