# A real estate agent want help to predict the house price for regions in Usa.he gave us the dataset to work on to use linear Regression model.Create a model that helps him to estimate

# Data Collection

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

In [116]:
```
#import the dataset
data=pd.read_csv(r"C:\Users\user\Desktop\Vicky\13_placement.csv")
```

In [117]:
```
#to display top 10 rows
data.head()
```

Out[117]:

| | cgpa | placement_exam_marks | placed |
|---|---|---|---|
| **0** | 7.19 | 26.0 | 1 |
| **1** | 7.46 | 38.0 | 1 |
| **2** | 7.54 | 40.0 | 1 |
| **3** | 6.42 | 8.0 | 1 |
| **4** | 7.23 | 17.0 | 0 |

In [118]:
```
#to display null values
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   cgpa                  1000 non-null   float64
 1   placement_exam_marks  1000 non-null   float64
 2   placed                1000 non-null   int64  
dtypes: float64(2), int64(1)
memory usage: 23.6 KB
```

In [119]:
```
data.shape
```

Out[119]: (1000, 3)

In [120]:
```
#to display summary of statistics
data.describe()
```

Out[120]:

| | cgpa | placement_exam_marks | placed |
|---|---|---|---|
| **count** | 1000.000000 | 1000.000000 | 1000.000000 |
| **mean** | 6.961240 | 32.225000 | 0.489000 |
| **std** | 0.615898 | 19.130822 | 0.500129 |
| **min** | 4.890000 | 0.000000 | 0.000000 |
| **25%** | 6.550000 | 17.000000 | 0.000000 |
| **50%** | 6.960000 | 28.000000 | 0.000000 |
| **75%** | 7.370000 | 44.000000 | 1.000000 |
| **max** | 9.120000 | 100.000000 | 1.000000 |

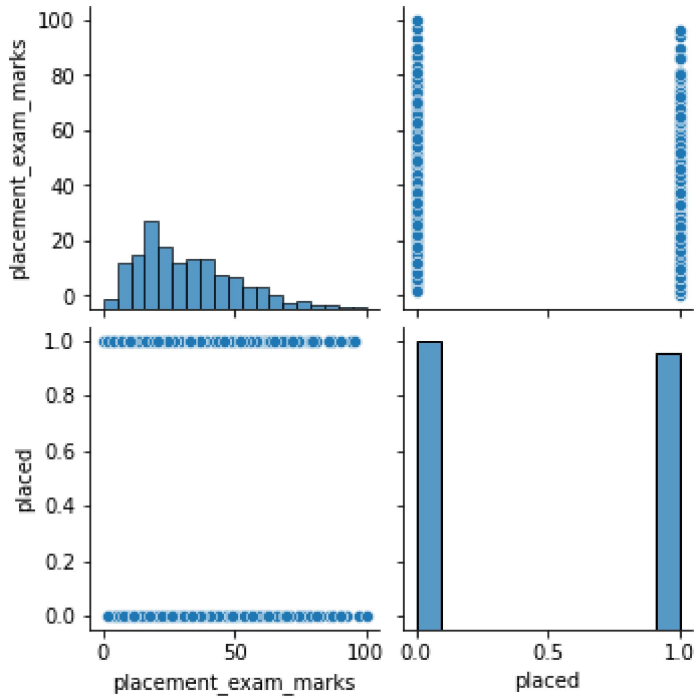
In [121]:
```
#to display columns name
data.columns
```

Out[121]:
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

In [122]:
```
data1=data[['placement_exam_marks', 'placed']]
```

In [132]:
```
sns.pairplot(data1)
```

Out[132]:
```
<seaborn.axisgrid.PairGrid at 0x250f82acdf0>
```

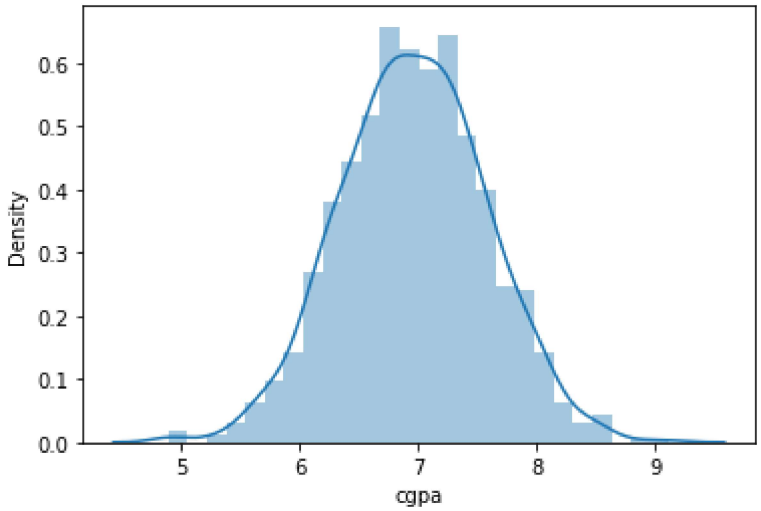



# EDA and Visualization

In [124]:
```
sns.distplot(data['cgpa'])
```

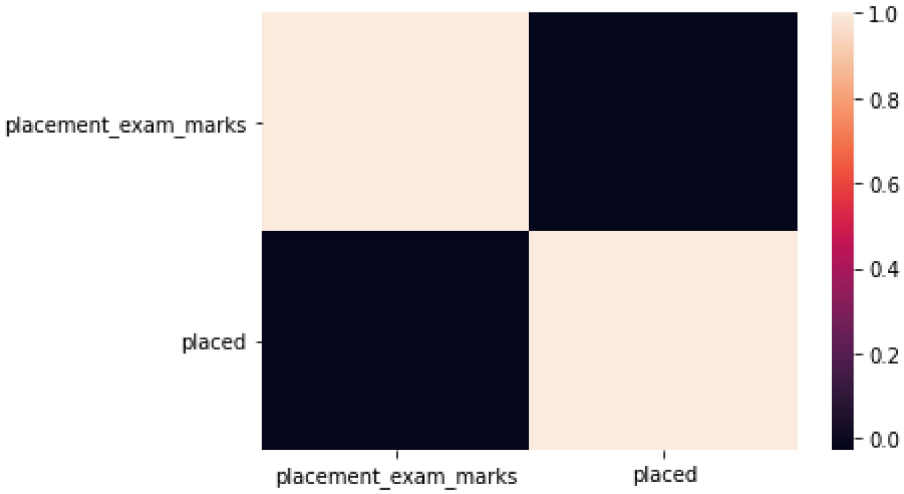
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function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

Out[124]: <AxesSubplot:xlabel='cgpa', ylabel='Density'>




In [125]:
```
sns.heatmap(data1.corr())
```

Out[125]: <AxesSubplot:>




# To train the model

we are going to train the linear regression model ;We need to split the two variable x and y where x in independent variable (input) and y is dependent of x(output) so we could ignore address columns as it is not requires for our model

In [133]:
```
x=data1[[ 'placement_exam_marks', 'placed']]
y=data1['placement_exam_marks']
```

In [134]:
```
#To split test and train data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.6)
```

In [135]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[135]:
```
LinearRegression()
```

In [86]:
```
lr.intercept_
```

Out[86]:
```
0.0
```

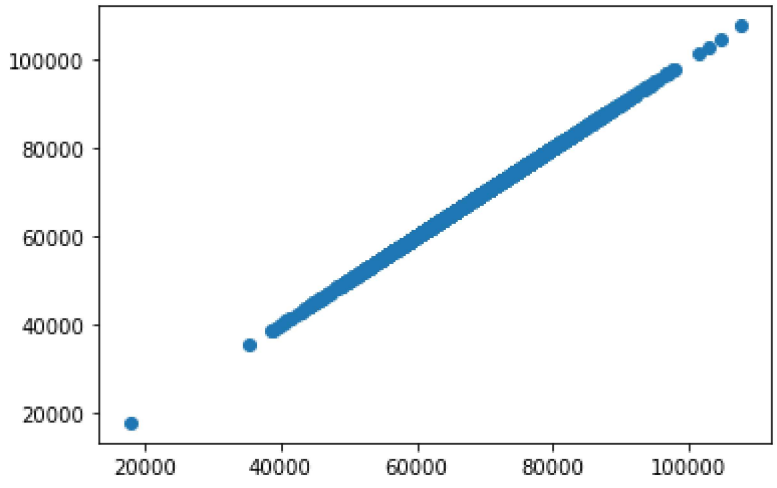
In [87]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[87]:

| | Co-efficient |
|---|---|
| **Price** | -1.125639e-17 |
| **Avg. Area Income** | 1.000000e+00 |

In [88]:
```
prediction = lr.predict(x_train)
plt.scatter(y_train,prediction)
```

Out[88]:
```
<matplotlib.collections.PathCollection at 0x250f6c3b8e0>
```




In [136]:
```
lr.score(x_test,y_test)
```

Out[136]:
```
1.0
```

In [137]:
```
lr.score(x_train,y_train)
```

Out[137]:
```
1.0
```

In [138]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [139]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[139]:
```
0.9999999957614583
```

In [140]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_test,y_test)
```

Out[140]:
```
0.9993267893556692
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: