Problem Statement:

A real estate agent want to help to predict the house price for regions in USA.He gave us the dataset to work on to use Linear Regression modelCreate a Model that helps him to 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("13_placement.csv")
df
```

Out[2]:

	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
...	...	...	...
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
998	8.62	46.0	1
999	4.90	10.0	1

1000 rows × 3 columns

In [3]:
```
df.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 [4]:
```
df.head()
```

Out[4]:

	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

Data cleaning and Pre-Processing

In [5]:
```
df.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 [6]:
```
df.describe()
```

Out[6]:

	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 [7]:
```
df.dropna(axis='columns')
```

Out[7]:

	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

	cgpa	placement_exam_marks	placed
4	7.23	17.0	0
...	...	...	...
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
998	8.62	46.0	1
999	4.90	10.0	1

1000 rows × 3 columns

In [8]:
```
a = df.dropna(axis='columns')
a.columns
```

Out[8]:
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

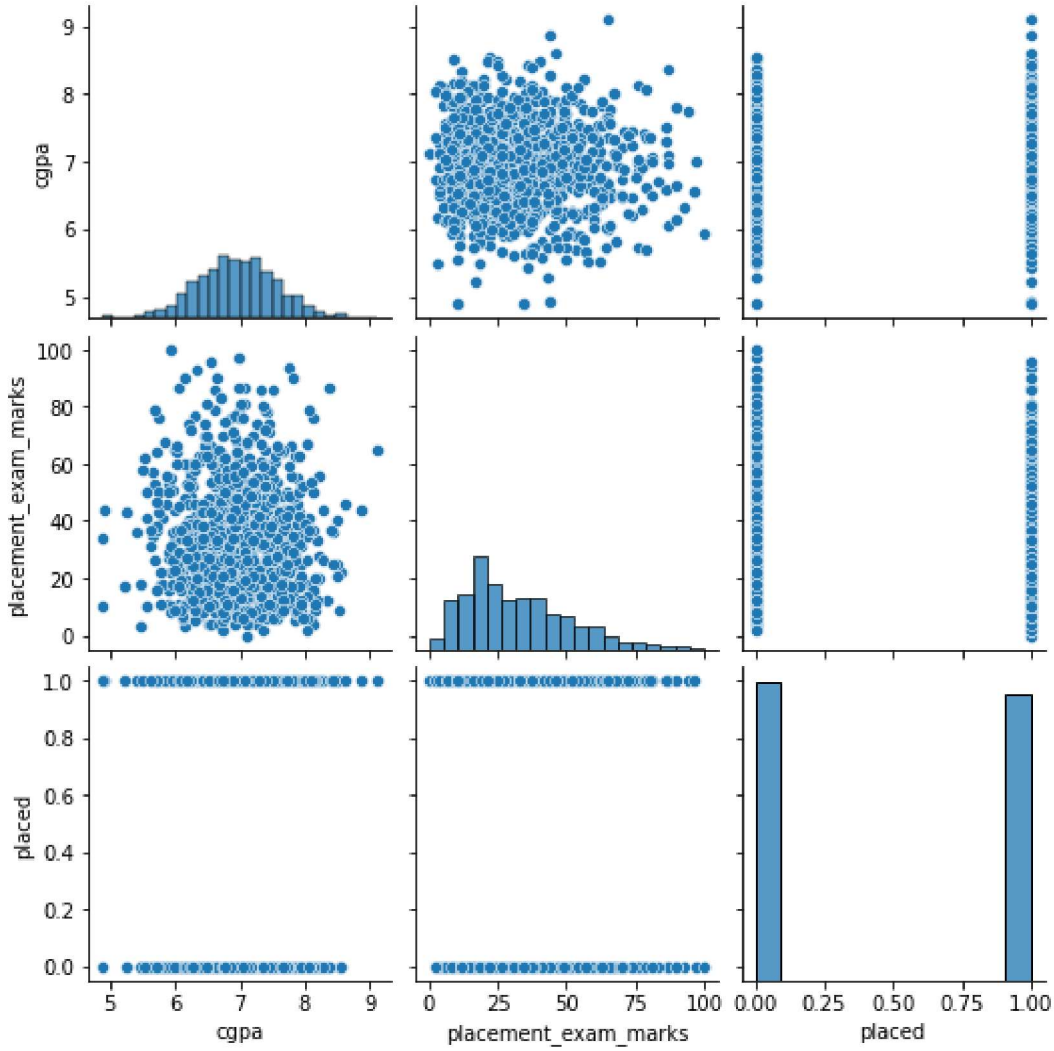
In [9]:
```
df.columns
```

Out[9]:
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

EDA and VISUALIZATION

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

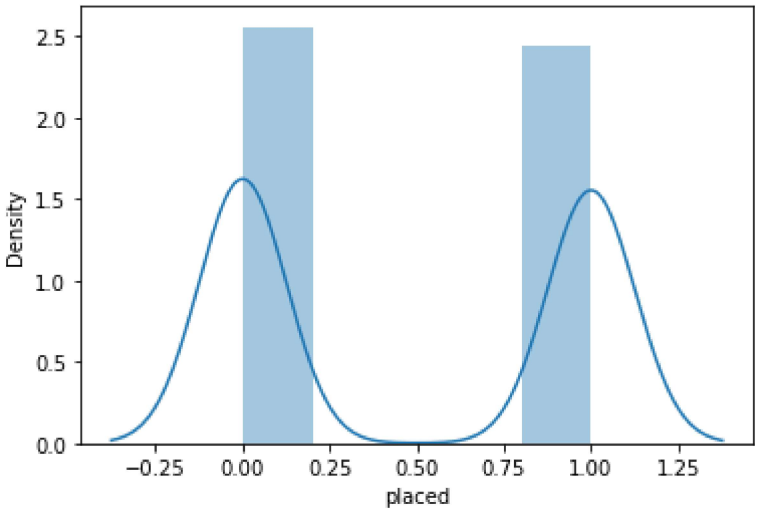
Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x1944e35f970>
```



In [11]:
```
sns.distplot(df['placed'])
```

```
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='placed', ylabel='Density'>

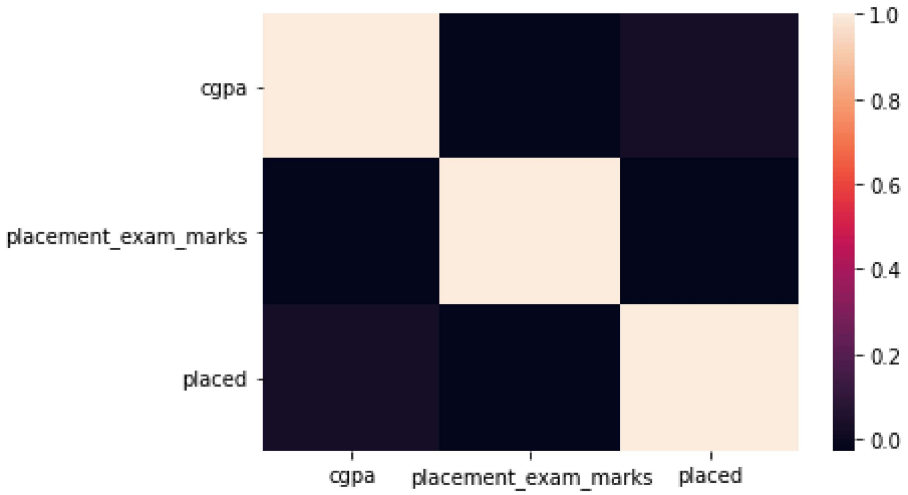


In [12]:
```
df1=df[[ 'cgpa', 'placement_exam_marks', 'placed']]
```

Plot Using Heat Map

In [13]:
```
sns.heatmap(df1.corr())
```

Out[13]: <AxesSubplot:>



To Train The Model-Model Building

we are going to train Linera Regression Model;We need to split out data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

In [14]:
```
x=df1[['cgpa', 'placement_exam_marks' ]]
y=df1[  'placed']
```

To Split my dataset into training and test data

In [15]:
```
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 [16]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[16]: LinearRegression()

In [17]:
```
lr.intercept_
```

Out[17]: 0.5343540417029523

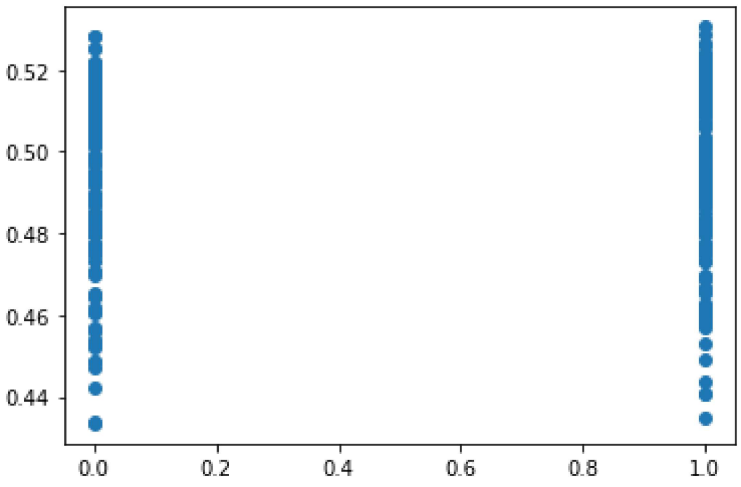
In [18]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[18]:

	Co-efficient
cgpa	-0.000532
placement_exam_marks	-0.001109

In [19]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[19]: <matplotlib.collections.PathCollection at 0x19450ae6a90>



In [20]:
```
lr.score(x_test,y_test)
```

Out[20]: -0.004375709330400879

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

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

Out[22]: 0.0017730191796051509

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

Out[23]: -0.004371279044101728

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]:
```
-0.0012444334555343772
```