DATA COLLECTION

In [1]:

```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
# To Import Dataset
sd=pd.read_csv(r"c:\Users\user\Downloads\\placement.csv")
sd
```

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]:

```
# to display top 10 rows
sd.head(10)
```

Out[3]:

	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
5	7.30	23.0	1
6	6.69	11.0	0
7	7.12	39.0	1
8	6.45	38.0	0
9	7.75	94.0	1

DATA CLEANING AND PRE_PROCESSING

In [4]:

```
sd.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 [5]:

```
# to display summary of statistics
sd.describe()
```

Out[5]:

	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 [6]:

```
#to display colums heading
sd.columns
```

Out[6]:

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

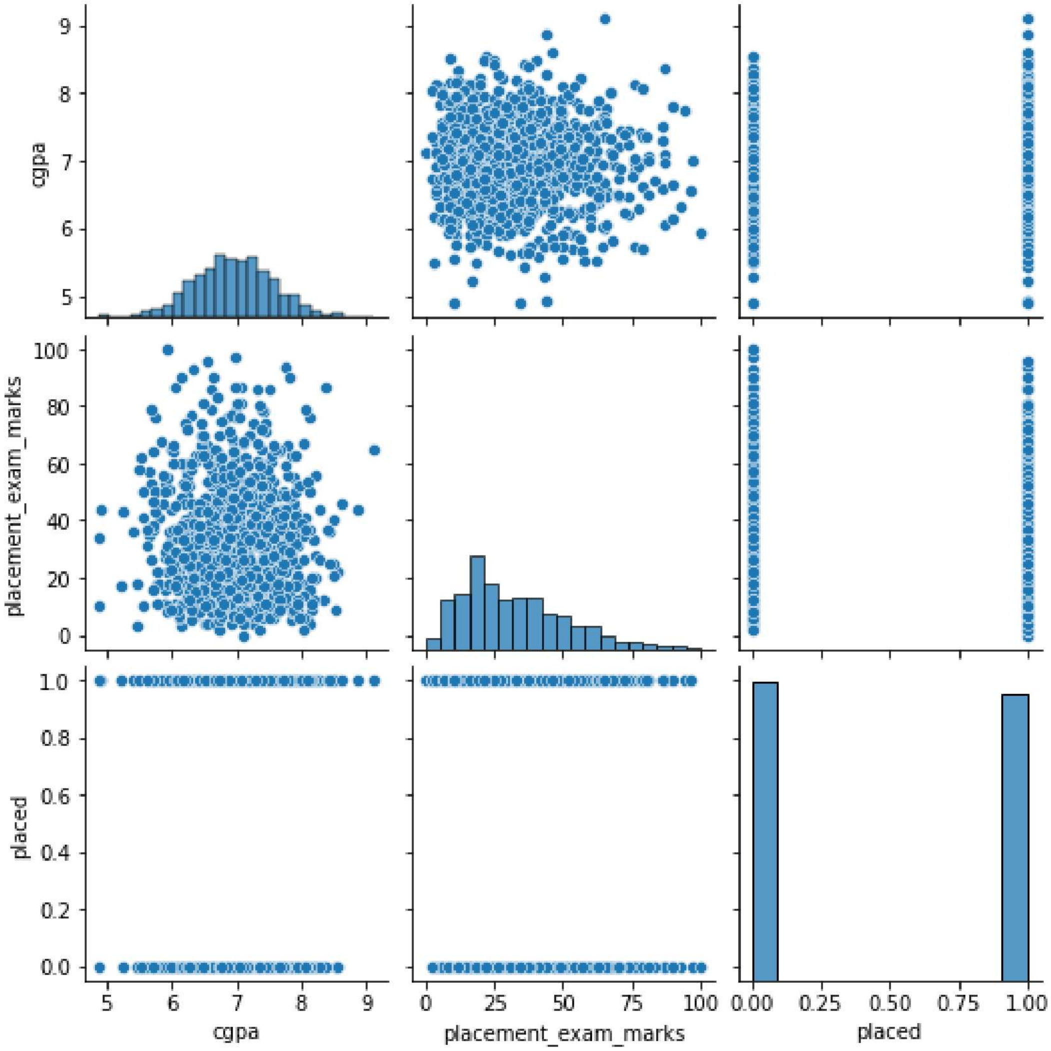
EDA and visualization

In [7]:

```
sns.pairplot(sd)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x220767c6fd0>
```



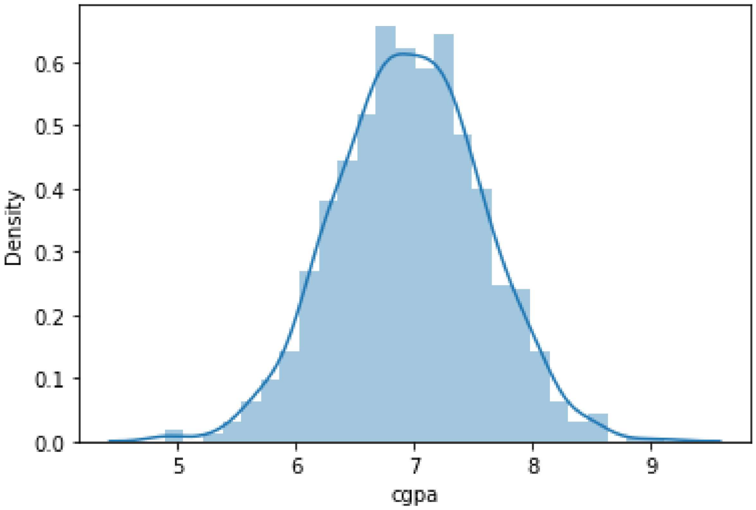
In [8]:

```
sns.distplot(sd['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[8]:

```
<AxesSubplot:xlabel='cgpa', ylabel='Density'>
```



In [9]:

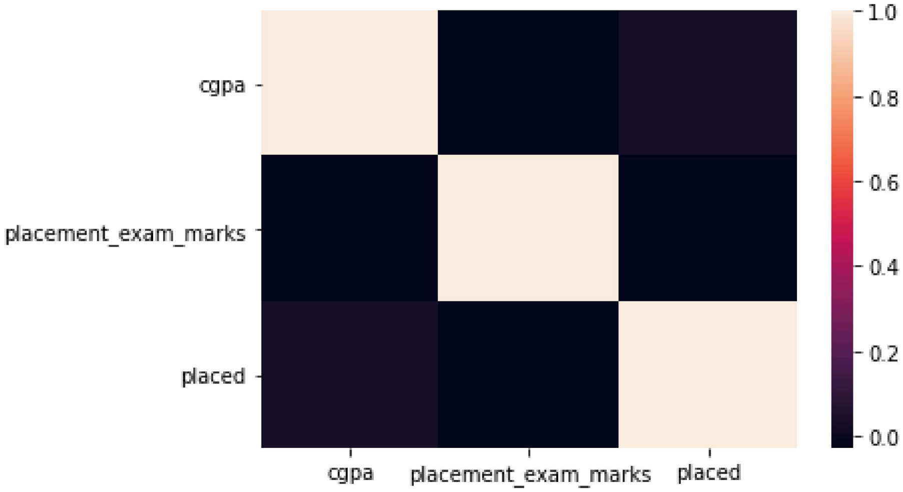
```
sd1=sd[['cgpa', 'placement_exam_marks', 'placed']]
```

In [10]:

```
sns.heatmap(sd1.corr())
```

Out[10]:

```
<AxesSubplot:>
```



TO TRAIN THE MODEL _MODEL BUILDING

we are goint train Liner Regression model; we need to split out the data into two varibles x and y where x is independent on x (output) and y is dependent on x(output) adress coloumn as it is not required our model

In [11]:
```
x= sd1[['cgpa', 'placement_exam_marks']]
y=sd1['placed']
```

In [12]:
```
# To split my dataset  into training data and test data
from sklearn .model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4)
```

In [13]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

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

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

```
0.5099041525907622
```

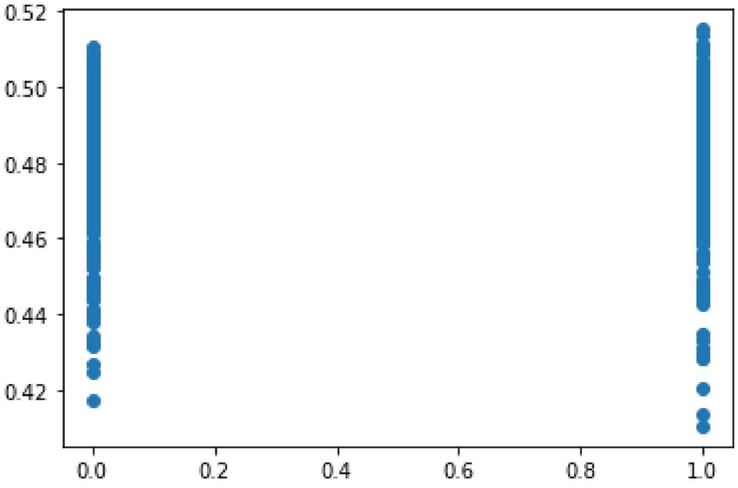
In [15]:
```
coeff= pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
cgpa	0.000756
placement_exam_marks	-0.001088

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

Out[16]:
```
<matplotlib.collections.PathCollection at 0x22078f80520>
```



In [17]:
```
print(lr.score(x_test,y_test))
```

```
-0.003601326842912256
```

In [18]:
```
lr.score(x_train,y_train)
```
Out[18]:
```
0.0017995666628813911
```

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
dr=Ridge(alpha=10)
dr.fit(x_train,y_train)
```
Out[20]:
```
Ridge(alpha=10)
```

In [21]:
```
dr.score(x_test,y_test)
```
Out[21]:
```
-0.003605905461431025
```

In [22]:
```
dr.score(x_train,y_train)
```
Out[22]:
```
0.0017995651466249374
```

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]:
```
-0.0020250506262655676
```

In [25]:
```
la.score(x_train,y_train)
```
Out[25]:
```
0.0
```

ElasticNet

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

In [27]:
```
print(en.coef_)
```
```
[ 0. -0.]
```

In [28]:
```
print(en.intercept_)
```
```
0.48
```

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

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

```
-0.0020250506262655676
```

Evaluation metrics

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

```
In [33]: print("mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
mean Absolute Error: 0.5001000000000001
```

```
In [34]: print("mean squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
mean squared Error: 0.2505
```

```
In [35]: print("Root mean Absolytre Error:",np.sqrt(metrics.mean_squared_error(y_test,pr
```

```
Root mean Absolytre Error: 0.500499750249688
```

```
In [ ]:
```