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(r'C:\Users\user\Downloads\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.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

In [4]:

```
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 [5]:

```
df.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]:

```
df.columns
```

Out[6]:

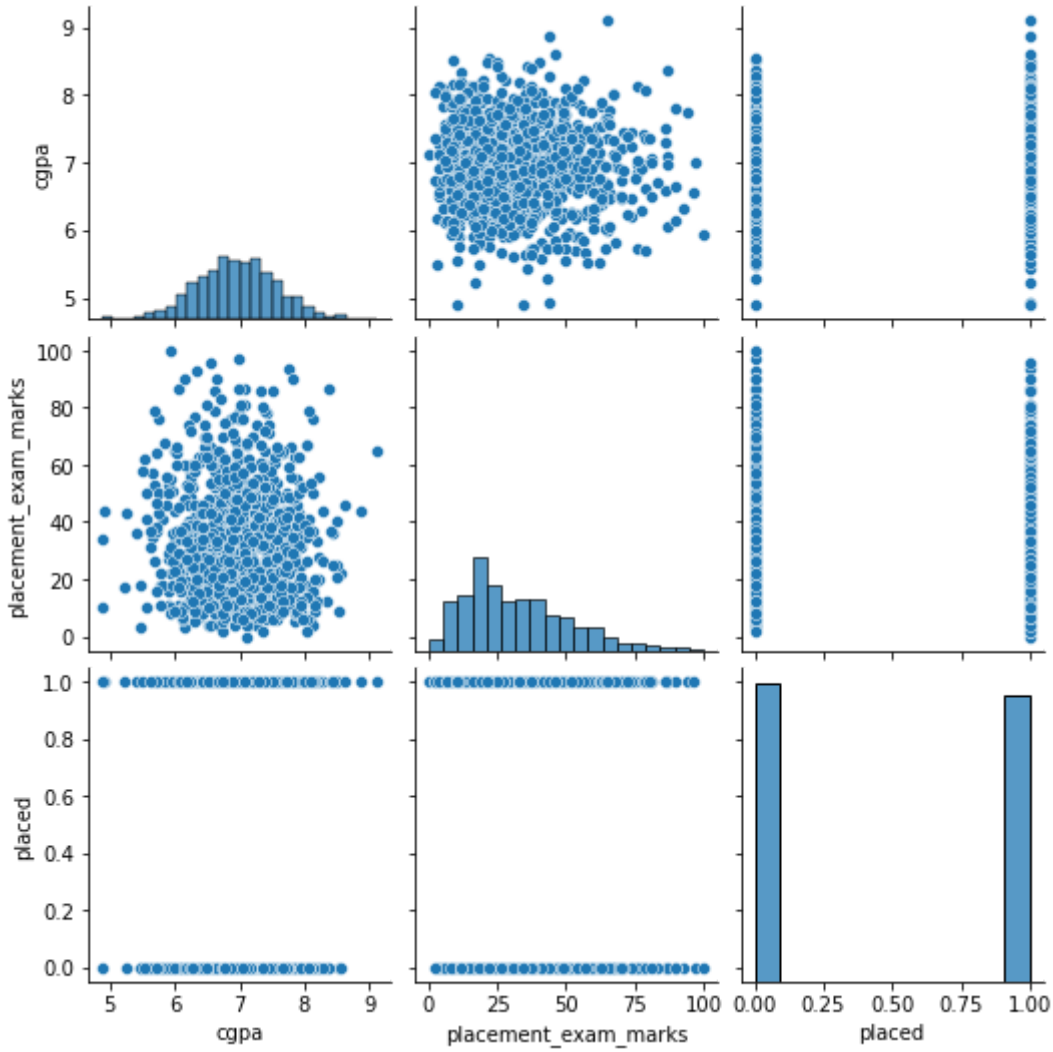
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

In [7]:

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

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x15b9258d190>
```



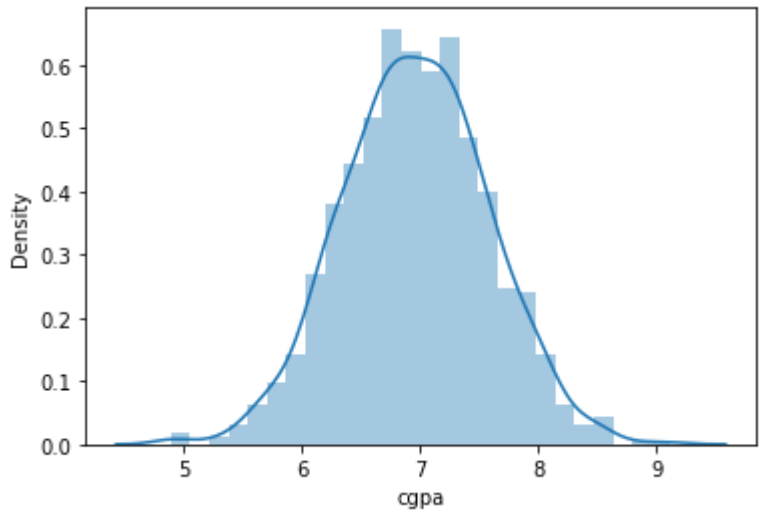
In [8]:

```
sns.distplot(df['cgpa'])
```

```
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 adapt your code to use either `displot` (a figure
-level function with similar flexibility) or `histplot` (an axes-level fun
ction for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]:

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

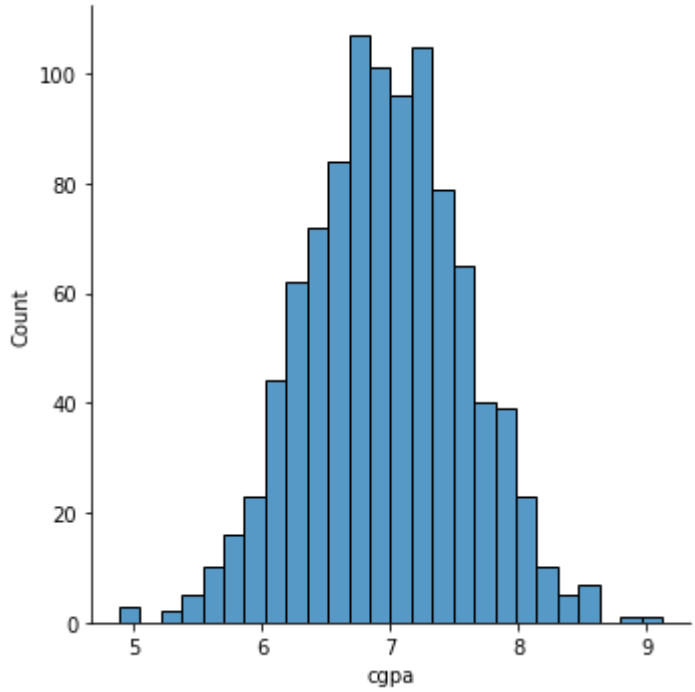


In [9]:

```
sns.displot(df["cgpa"])
```

Out[9]:

```
<seaborn.axisgrid.FacetGrid at 0x15b944565b0>
```



In [10]:

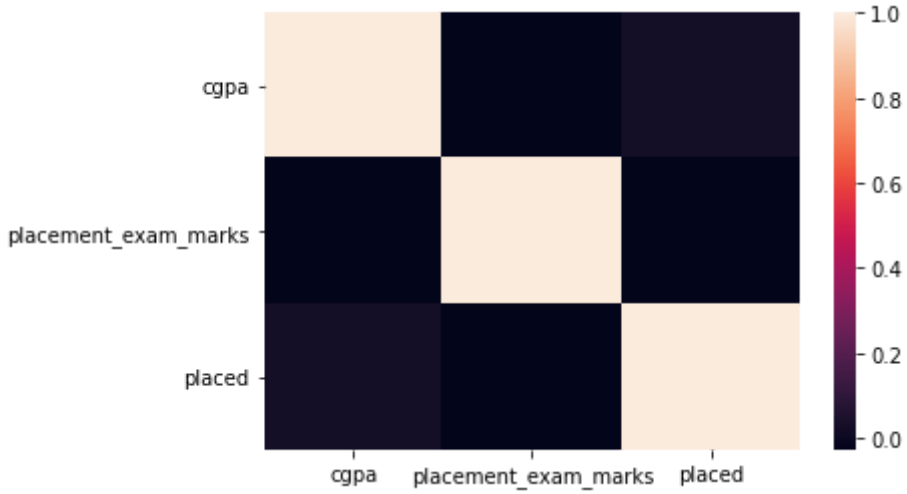
```
df1=df[['cgpa', 'placement_exam_marks', 'placed']]
```

In [11]:

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

Out[11]:

<AxesSubplot:>



In [12]:

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

In [13]:

```
from sklearn.model_selection import train_test_split
```

In [14]:

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [15]:

```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)#ValueError: Input contains NaN, infinity or a value too large for
```

Out[15]:

```
LinearRegression()
```

In [16]:

```
print(lr.intercept_)
```

```
[6.99757515]
```

In [17]:

```
coef= pd.DataFrame(lr.coef_)
coef
```

Out[17]:

	0	1
0	-0.000935	-0.028717

In [18]:

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

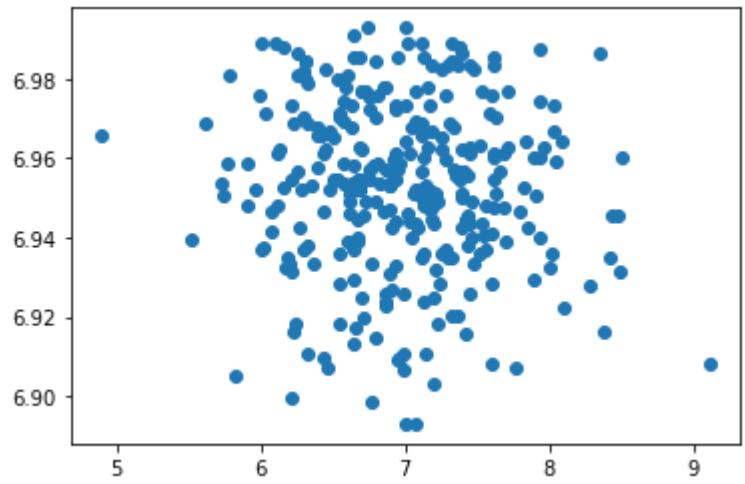
```
-0.008568551000891489
```

In [19]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[19]:

```
<matplotlib.collections.PathCollection at 0x15b94dcf8e0>
```



In [20]:

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

Out[20]:

```
-0.008568551000891489
```

In [21]:

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

Out[21]:

```
0.0013907902362197966
```

In [22]:

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

In [23]:

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

Out[23]:

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

In [24]:

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

Out[24]:

```
-0.008139019121807811
```

In [25]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[25]:

```
Lasso(alpha=10)
```

In [26]:

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

Out[26]:

```
-0.0017084266310283258
```

Elastic Net

In [27]:

```
from sklearn.linear_model import ElasticNet
en = ElasticNet()
en.fit(x_train,y_train)
```

Out[27]:

```
ElasticNet()
```

In [28]:

```
print(en.coef_)
```

```
[-0. -0.]
```

In [29]:

```
print(en.intercept_)
```

```
[6.95367143]
```

In [30]:

```
prediction=en.predict(x_test)
print(prediction)
```

```
[6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143
 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143 6.95367143]
```

In [31]:

```
print(en.score(x_test,y_test))
```

```
-0.0017084266310283258
```

Evaluation Metrics

In [32]:

```
from sklearn import metrics
```

In [33]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.4850333333333333
```

In [34]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.3731902708163265
```

In [35]:

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

```
Root Mean Squared Error: 0.6108930109408083
```