In [1]:

```
import pandas as pd
import numpy as np
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
```

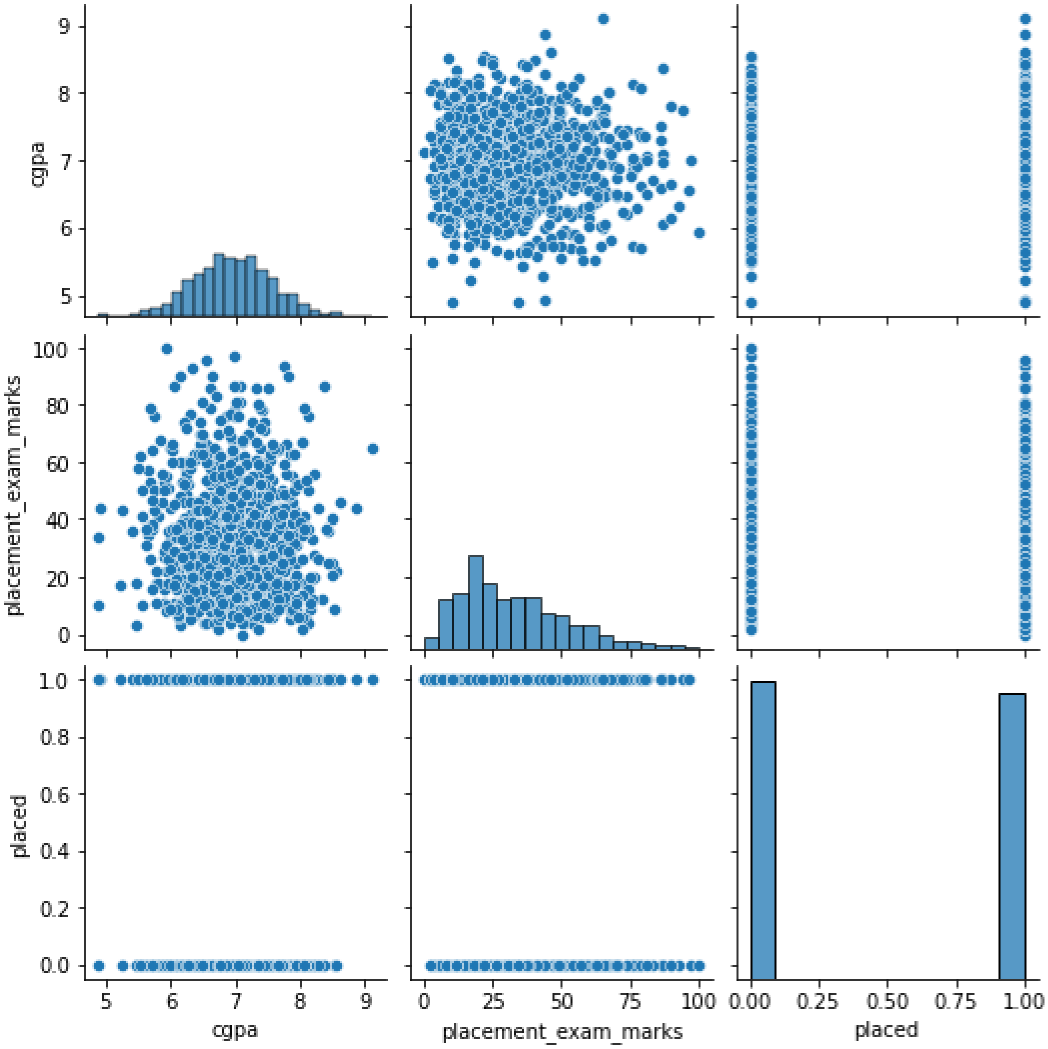
```
In [4]: df.describe()
```

Out[4]:

	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

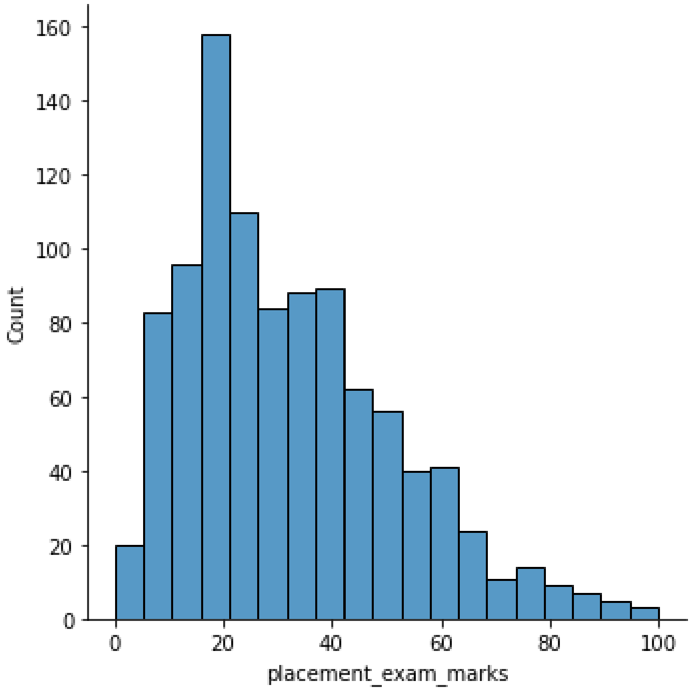
```
In [5]: sns.pairplot(df)
```

```
Out[5]: <seaborn.axisgrid.PairGrid at 0x2ac2478fc70>
```



In [6]:
```
sns.displot(df['placement_exam_marks'])
```

Out[6]:
```
<seaborn.axisgrid.FacetGrid at 0x2ac26166ee0>
```



In [7]:
```
df1=df.drop(['placed'],axis=1)
df1
```

Out[7]:

	cgpa	placement_exam_marks
0	7.19	26.0
1	7.46	38.0
2	7.54	40.0
3	6.42	8.0
4	7.23	17.0
...	...	...
995	8.87	44.0
996	9.12	65.0
997	4.89	34.0
998	8.62	46.0
999	4.90	10.0

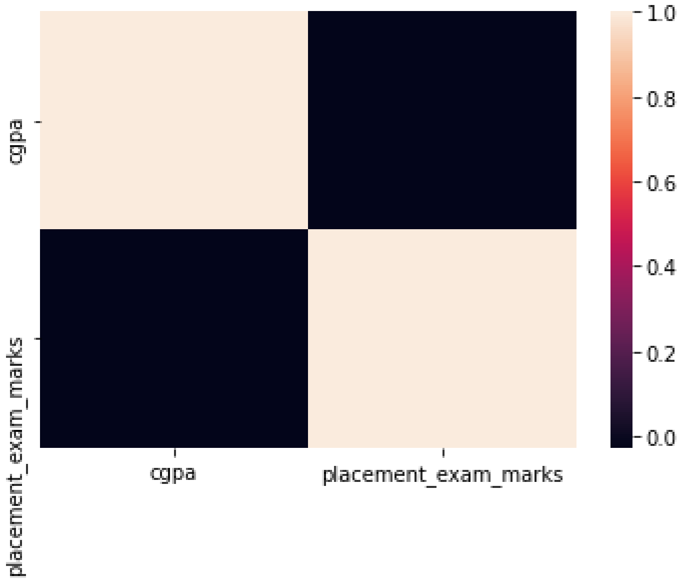
1000 rows × 2 columns

In [8]:

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

Out[8]:

```
<AxesSubplot:>
```



In [9]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [10]:

```
y=df['placement_exam_marks']
x=df1.drop(['placement_exam_marks'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
```

```
     cgpa
562  6.06
299  6.75
935  8.12
320  7.18
603  6.55
..    ...
604  6.89
801  6.31
661  5.72
99   7.46
891  7.42

[700 rows x 1 columns]
```

In [11]:

```
model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

Out[11]:

```
34.04938202440506
```

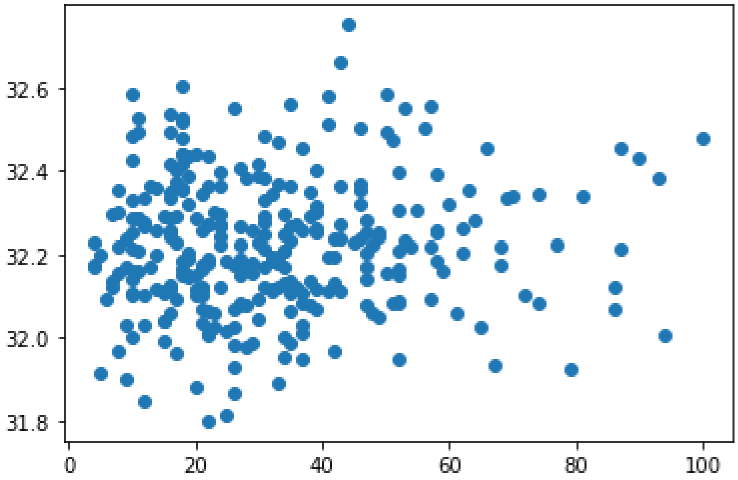
In [12]:
```
coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

Out[12]:

	Coefficient
cgpa	-0.263793

In [13]:
```
prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[13]:
```
<matplotlib.collections.PathCollection at 0x2ac269874f0>
```



In [14]:
```
model.score(x_test,y_test)
```

Out[14]:
```
0.0011252355314238516
```

In [15]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [16]:
```
rr = Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[16]:
```
Ridge(alpha=10)
```

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

Out[17]:
```
0.0010865758505722578
```

In [18]:
```
la = Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[18]:
```
Lasso(alpha=10)
```

In [19]:
```
la.score(x_test,y_test)
```

Out[19]:
```
-6.778268905938134e-06
```

In [20]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
print(en.coef_)
print(en.intercept_)
print(en.predict(x_test))
print(en.score(x_test,y_test))
from sklearn import metrics
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prec
```

```
[-0.]
32.21
[32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21
 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21 32.21]
-6.778268905938134e-06
Mean Absolute Error: 15.087896283234647
Mean Squared Error: 368.4107175132832
Root Mean Squared Error: 19.19402817319187
```