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

In [2]:
```
d=pd.read_csv(r"C:\Users\user\Downloads\13_placement - 13_placement.csv")
d
```

Out[2]:

	cgpa	placement_exam_marks	placed
0	7.19	26	1
1	7.46	38	1
2	7.54	40	1
3	6.42	8	1
4	7.23	17	0
...	...	...	...
995	8.87	44	1
996	9.12	65	1
997	4.89	34	0
998	8.62	46	1
999	4.90	10	1

1000 rows × 3 columns

In [3]:
```
d.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   int64  
 2   placed               1000 non-null   int64  
dtypes: float64(1), int64(2)
memory usage: 23.6 KB
```

In [4]:
```
d.head()
```

Out[4]:

	cgpa	placement_exam_marks	placed
0	7.19	26	1
1	7.46	38	1
2	7.54	40	1
3	6.42	8	1

	cgpa	placement_exam_marks	placed
4	7.23	17	0

In [5]:
```
d.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]:
```
d.columns
```

Out[6]:
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

In [7]:
```
d.index
```

Out[7]:
```
RangeIndex(start=0, stop=1000, step=1)
```

In [8]:
```
d=d.head(100)
d
```

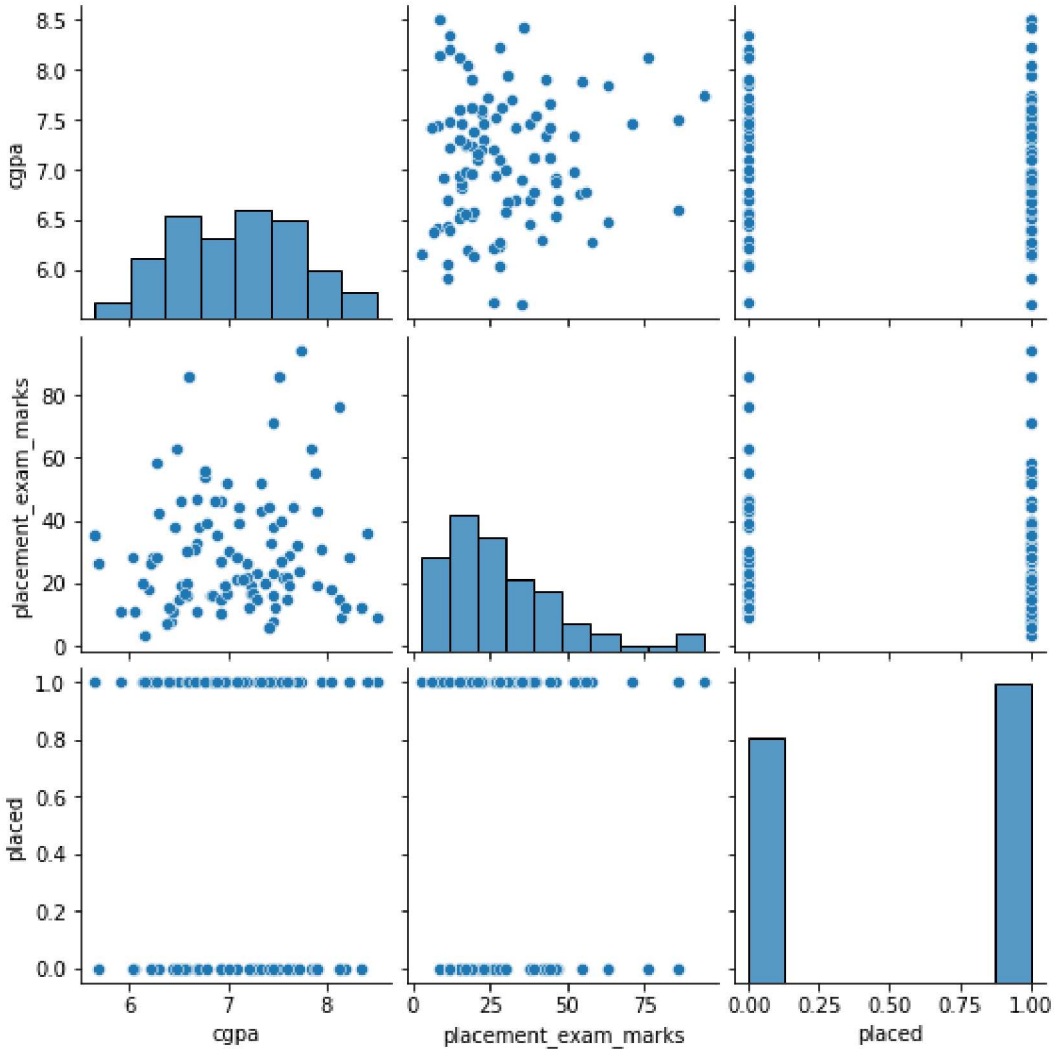
Out[8]:

	cgpa	placement_exam_marks	placed
0	7.19	26	1
1	7.46	38	1
2	7.54	40	1
3	6.42	8	1
4	7.23	17	0
...	...	...	...
95	6.89	35	1
96	7.09	28	1
97	7.33	52	1
98	8.12	76	0
99	7.46	23	0

100 rows × 3 columns

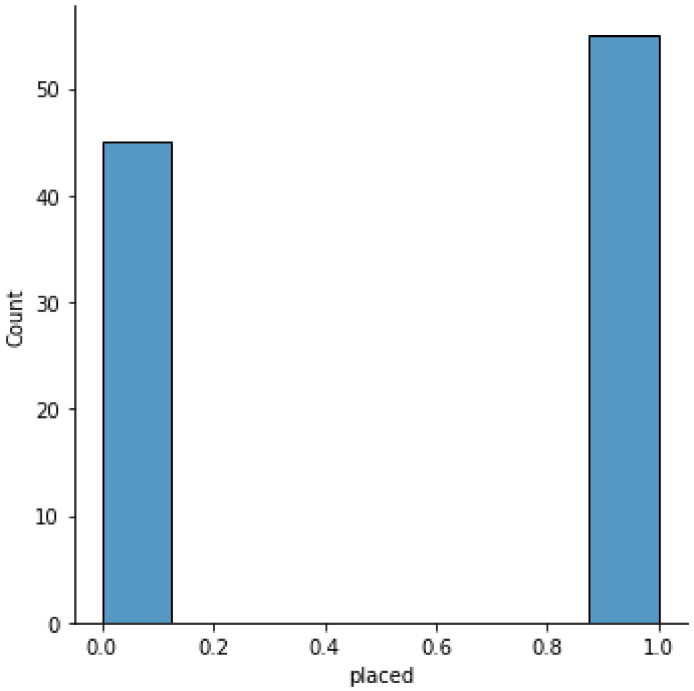
In [9]:
```
sns.pairplot(d)
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x1f33c8b8940>



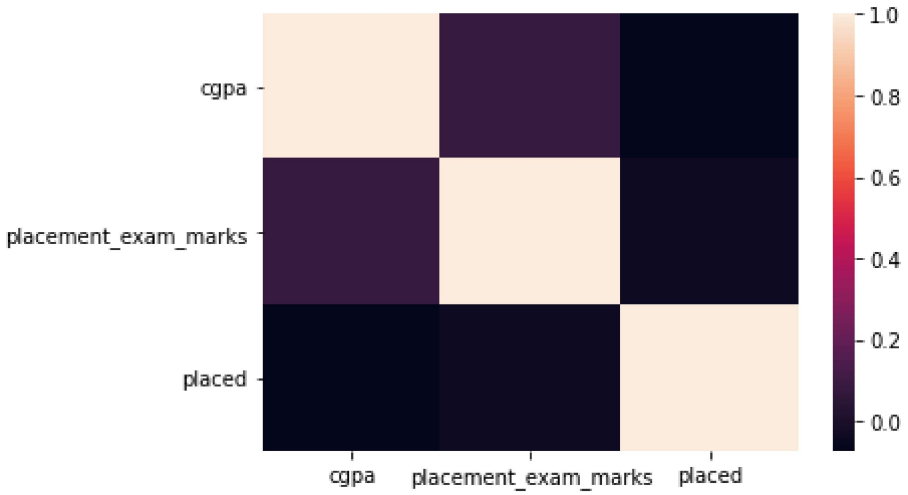
In [10]:
```
sns.displot(d['placed'])
```

Out[10]: <seaborn.axisgrid.FacetGrid at 0x1f33ddc29d0>



In [11]:
```
d1=d[['cgpa', 'placement_exam_marks', 'placed']]
sns.heatmap(d1.corr())
```

Out[11]: <AxesSubplot:>



In [12]:
```
x=d1[['cgpa', 'placement_exam_marks']]
y=d1['placed']
```

In [13]:
```
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 [14]:
```
from sklearn.linear_model import LinearRegression
```

In [15]:
```
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
print(lr.intercept_)
```

```
0.7417699275843355
```

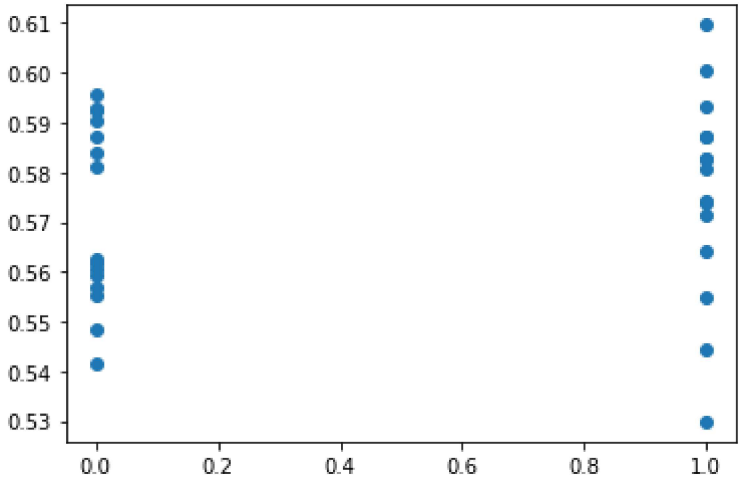
In [17]:
```
coeff =pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[17]:

	Co-efficient
cgpa	-0.025214
placement_exam_marks	0.000296

In [18]:
```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[18]:
```
<matplotlib.collections.PathCollection at 0x1f33ef91ee0>
```



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

```
-0.013954196528763507
```

In [20]:
```
print(lr.score(x_train,y_train))
```

```
0.0009731436587950837
```

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

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

Out[22]:
```
Ridge(alpha=10)
```

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

Out[23]:
```
-0.016311900920582545
```

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]:
```
-0.020408163265306367
```

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_)
```

```
0.5714285714285714
```

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

```
[0.57142857 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857
 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857
 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857
 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857
 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857]
```

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

```
-0.020408163265306367
```

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

In [32]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.4977147782588006
```

In [33]:
```
print(en.coef_)
```

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

In [34]:
```
print(en.intercept_)
```

```
0.5714285714285714
```

In [35]:
```
print(en.predict(x_test))
```

```
[0.57142857 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857
 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857
 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857
 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857
 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857 0.57142857]
```

In [36]:
```
print(en.score(x_test,y_test))
```

```
-0.020408163265306367
```

In [37]:
```
from sklearn import metrics
```

In [38]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.4977147782588006
```

In [39]:
```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.2534885491321909
```

In [40]:
```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 0.5034764633348722
```

In [ ]:
```

```