In [ ]:

```
```

In [165]:

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

In [166]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\13_placement.csv")
a
```

Out[166]:

	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 [167]:

```
a=a.head(10)
a
```

Out[167]:

	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 [168]:

```
# to find
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   cgpa                  10 non-null     float64
 1   placement_exam_marks  10 non-null     float64
 2   placed                10 non-null     int64  
dtypes: float64(2), int64(1)
memory usage: 368.0 bytes
```

In [169]:

```
# to display summary of statastic
a.describe()
```

Out[169]:

	cgpa	placement_exam_marks	placed
count	10.000000	10.000000	10.000000
mean	7.115000	33.400000	0.700000
std	0.454832	24.423122	0.483046
min	6.420000	8.000000	0.000000
25%	6.797500	18.500000	0.250000
50%	7.210000	32.000000	1.000000
75%	7.420000	38.750000	1.000000
max	7.750000	94.000000	1.000000

In [170]:

```
# to display colum heading
a.columns
```

Out[170]:

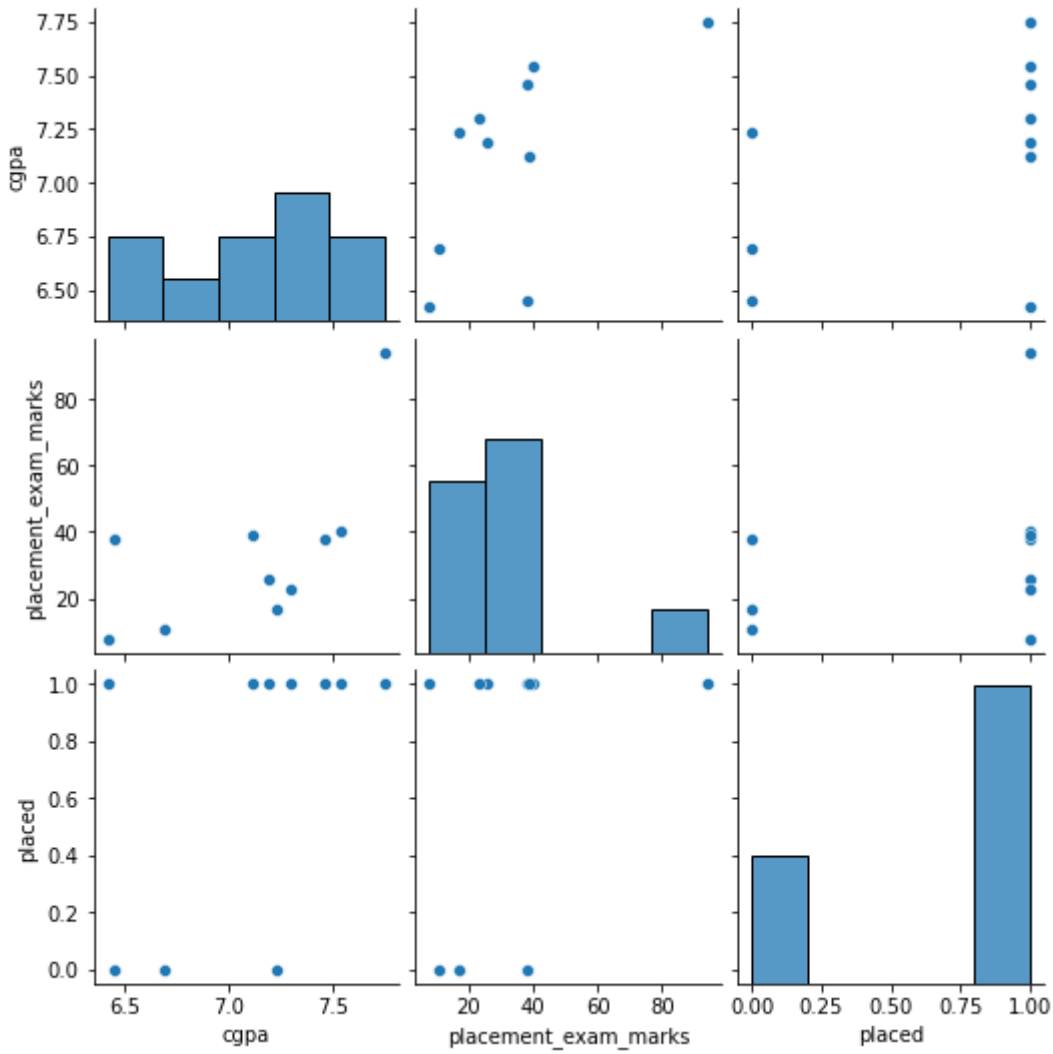
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

In [171]:

```
sns.pairplot(a)
```

Out[171]:

```
<seaborn.axisgrid.PairGrid at 0x198ce7e0550>
```

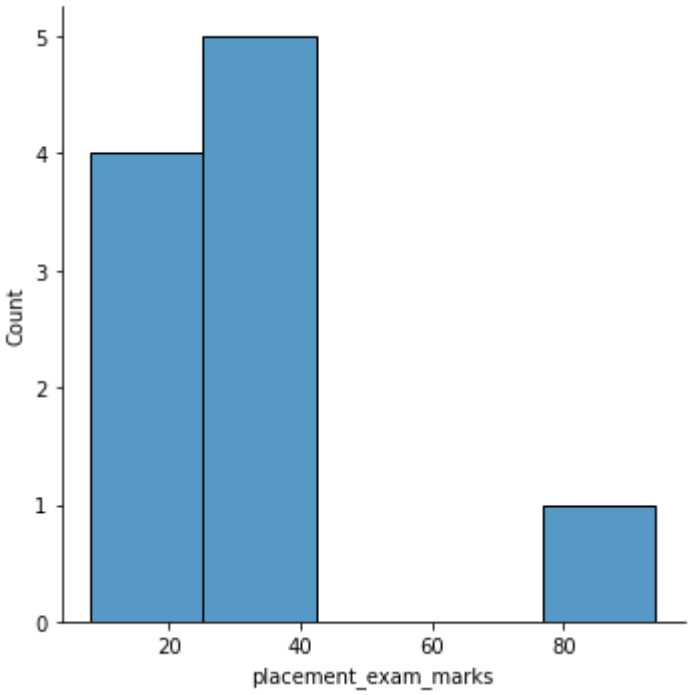


In [172]:

```
sns.displot(a["placement_exam_marks"])
```

Out[172]:

```
<seaborn.axisgrid.FacetGrid at 0x198cf088ee0>
```



In [173]:

```
b=a[['cgpa', 'placement_exam_marks', 'placed']]
b
```

Out[173]:

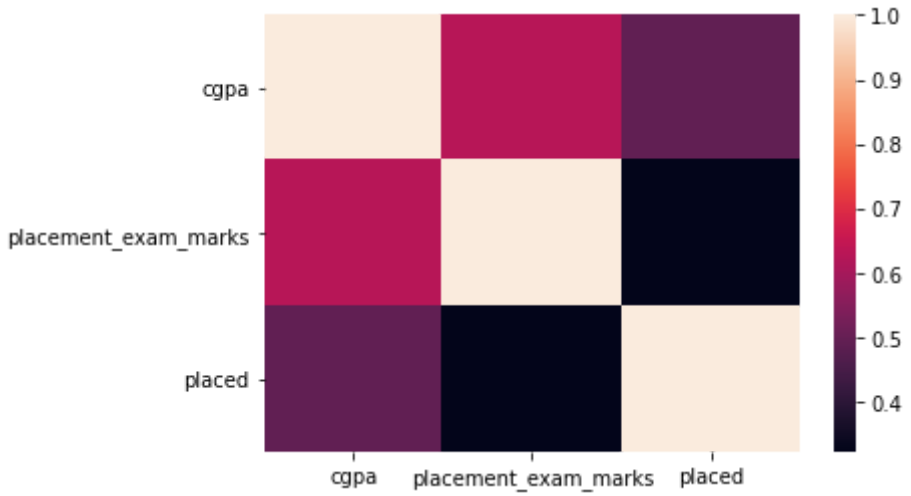
	cgpa	placement_exam_marks	placed
0	7.19	26.0	1
1	7.46	38.0	1
2	7.54	40.0	1
3	6.42	8.0	1
4	7.23	17.0	0
5	7.30	23.0	1
6	6.69	11.0	0
7	7.12	39.0	1
8	6.45	38.0	0
9	7.75	94.0	1

In [174]:

```
sns.heatmap(b.corr())
```

Out[174]:

<AxesSubplot:>



In [176]:

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

In [177]:

```
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 [178]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[178]:

```
LinearRegression()
```

In [179]:

```
lr.intercept_
```

Out[179]:

```
-1.4210854715202004e-14
```

In [180]:

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

Out[180]:

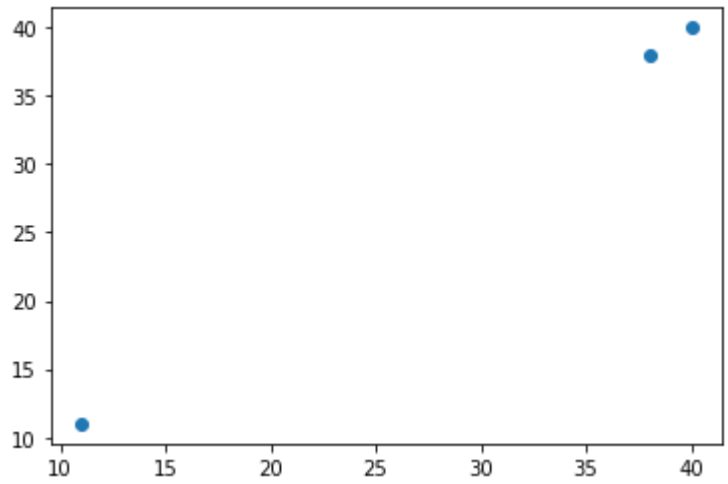
	Co-efficient
cgpa	1.102898e-15
placement_exam_marks	1.000000e+00
placed	1.516385e-15

In [181]:

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

Out[181]:

```
<matplotlib.collections.PathCollection at 0x198cf307520>
```



In [182]:

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

Out[182]:

```
1.0
```

In [183]:

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

Out[183]:

```
1.0
```

In [184]:

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

In [185]:

```
rr=Ridge(alpha=10)
rr.fit(x_test,y_test)
```

Out[185]:

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

In [186]:

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

Out[186]:

```
0.9996497286992322
```

In [187]:

```
la=Lasso(alpha=10)
la.fit(x_test,y_test)
```

Out[187]:

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

In [188]:

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

Out[188]:

```
0.9967305435047605
```

In [189]:

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

Out[189]:

```
ElasticNet()
```

In [190]:

```
en.coef_
```

Out[190]:

```
array([0.        , 0.99853786, 0.        ])
```

In [191]:

```
en.intercept_
```

Out[191]:

```
0.05117493472585011
```

In [192]:

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

Out[192]:

```
array([37.99561358, 39.9926893 , 11.03509138])
```

In [193]:

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

Out[193]:

```
0.9999975144364713
```

EVALUATION METRICS

In [194]:

```
from sklearn import metrics
```

In [195]:

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

```
Mean Absolute Error: 0.015596170583116612
```

In [196]:

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

```
Mean Squared Error 0.000434697443798353
```

In [197]:

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

```
Root Mean Squared Error 0.020849399123196645
```

In [ ]:

```

```