Problem statement

Data collection

Importing libraries

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

Importing dataset

In [2]:
```
data=pd.read_csv(r"C:\Users\user\Downloads\PLACEMENT.csv")
data
```

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

head

In [3]:
```
# to display first 8 dataset values
da=data.head(8)
da
```

Out[3]:

	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
5	7.30	23	1
6	6.69	11	0
7	7.12	39	1

info

In [4]:
```
# to identify missing values
data.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
```

describe

In [5]:
```
# to display summary of the dataset
data.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

columns

In [6]:
```
# to display headings of the dataset
data.columns
```

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

In [7]:
```
a=data.dropna(axis=1)
a
b=a.head(8)
b
```

Out[7]:

	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
5	7.30	23	1
6	6.69	11	0
7	7.12	39	1

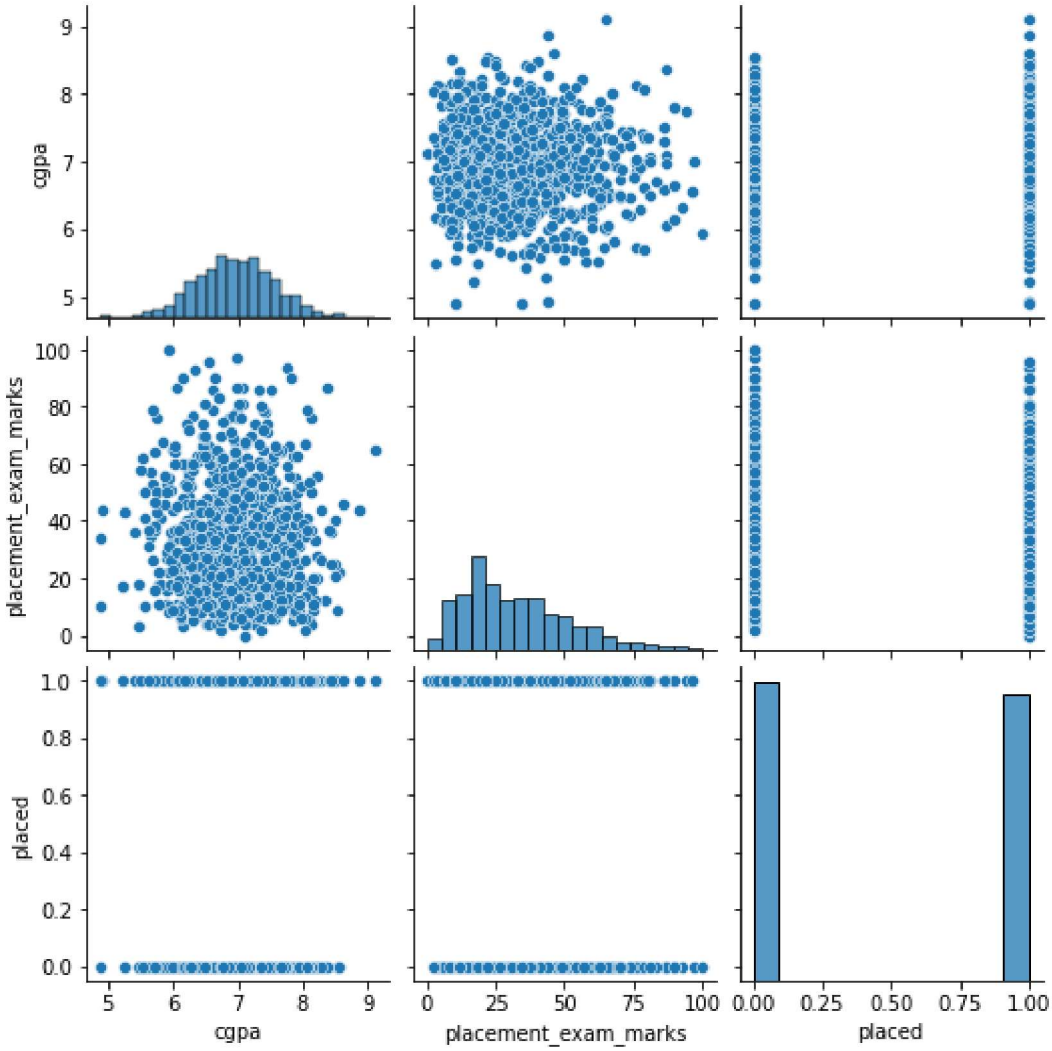
In [8]:
```
a.columns
```

Out[8]:
```
Index(['cgpa', 'placement_exam_marks', 'placed'], dtype='object')
```

EDA and Visualization

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

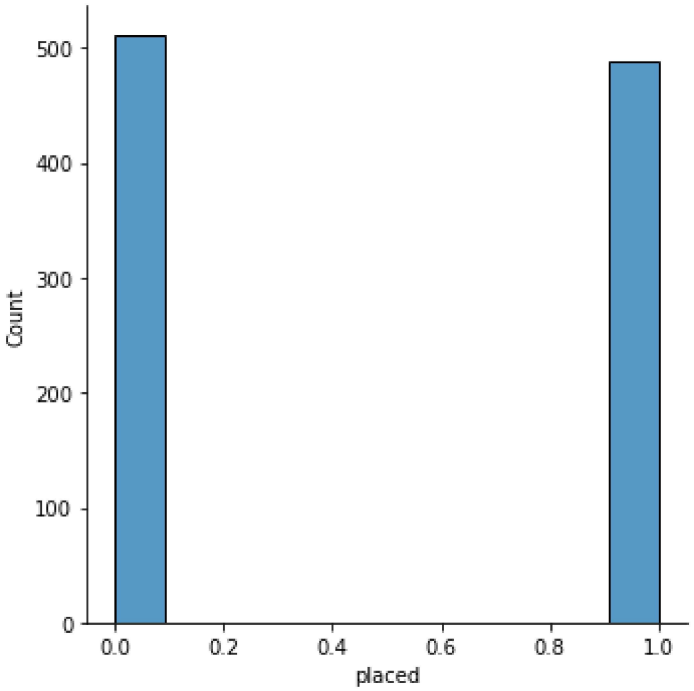
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x13c55f41550>
```



distribution plot

In [10]:
```
sns.displot(a["placed"])
```

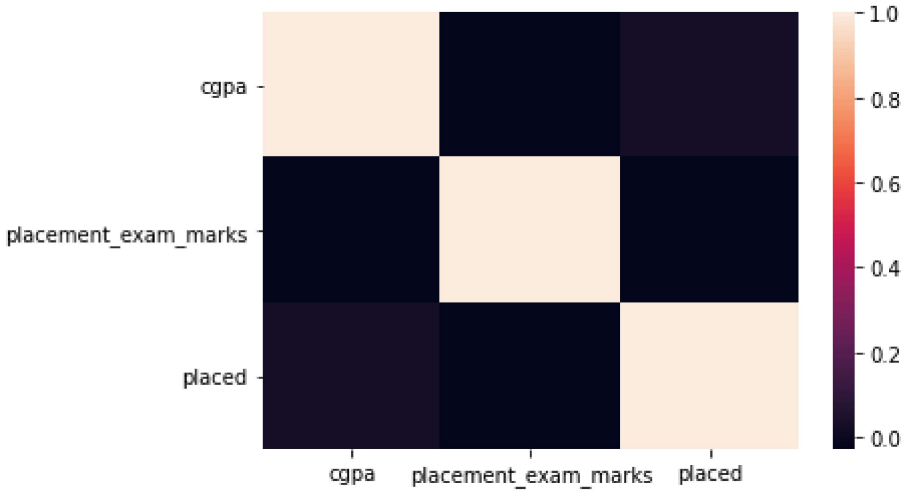
Out[10]: <seaborn.axisgrid.FacetGrid at 0x13c57bcdb20>



correlation

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

Out[11]: <AxesSubplot:>



To train the model-Model Building

In [12]:
```
x=a[['placed']]
y=a['placement_exam_marks']
```

In [13]:
```
# to split my dataset into training 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=0.3)
```

In [14]:
```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]:
```
LinearRegression()
```

In [15]:
```
print(lr.intercept_)
```

```
33.328611898017
```

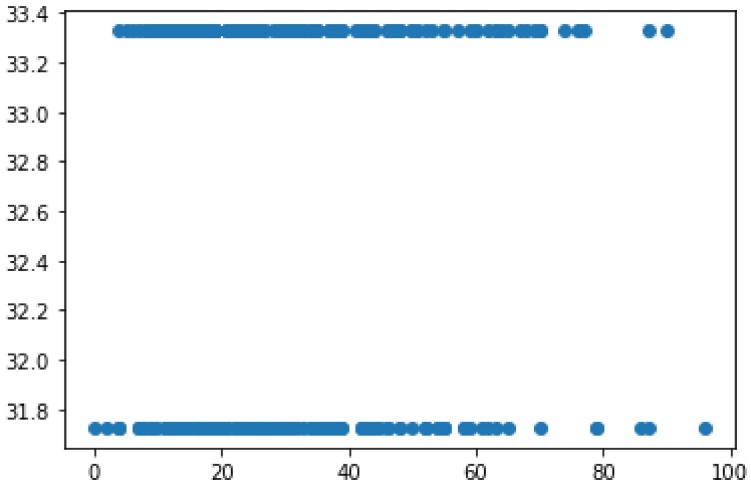
In [16]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
placed	-1.602387

In [17]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]:
```
<matplotlib.collections.PathCollection at 0x13c58676370>
```



In [18]:
```
print(lr.score(x_test,y_test))
```

```
-0.0059288704050051155
```

In [19]:
```
lr.score(x_train,y_train)
```

Out[19]:
```
0.0017389154837557097
```

Ridge regression

In [20]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[21]: -0.005677495470711946

In [22]:
```
rr.score(x_train,y_train)
```

Out[22]: 0.0017338339433277117

Lasso regression

In [23]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_train,y_train)
```

Out[23]: 0.0

In [24]:
```
la.score(x_test,y_test)
```

Out[24]: -0.0029795159037266927

In [25]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[25]: ElasticNet()

In [26]:
```
print(en.coef_)
```

```
[-0.]
```

In [27]:
```
print(en.intercept_)
```

```
32.534285714285716
```

In [34]:
```
predict=en.predict(x_test)
```

In [31]:
```
print(en.score(x_test,y_test))
```

```
-0.0029795159037266927
```

In [32]:
```
from sklearn import metrics
```

In [35]:
```
print("Mean Absolute error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute error: 15.382609523809524
```

In [36]:
```
print("Mean Squared error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Squared error: 357.78618503401356
```

In [38]:
```
print("Root squared error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root squared error: 18.915236848477832
```