Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

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("placement")
```

To display top 10 rows

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

Data Cleaning And Pre-Processing

In [4]:
```
1 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]:
```
1 # Display the statistical summary
2 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]:
```
1 # To display the col headings
2 df.columns
```

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

In [7]:
```
1 cols=df.dropna(axis=1)
```

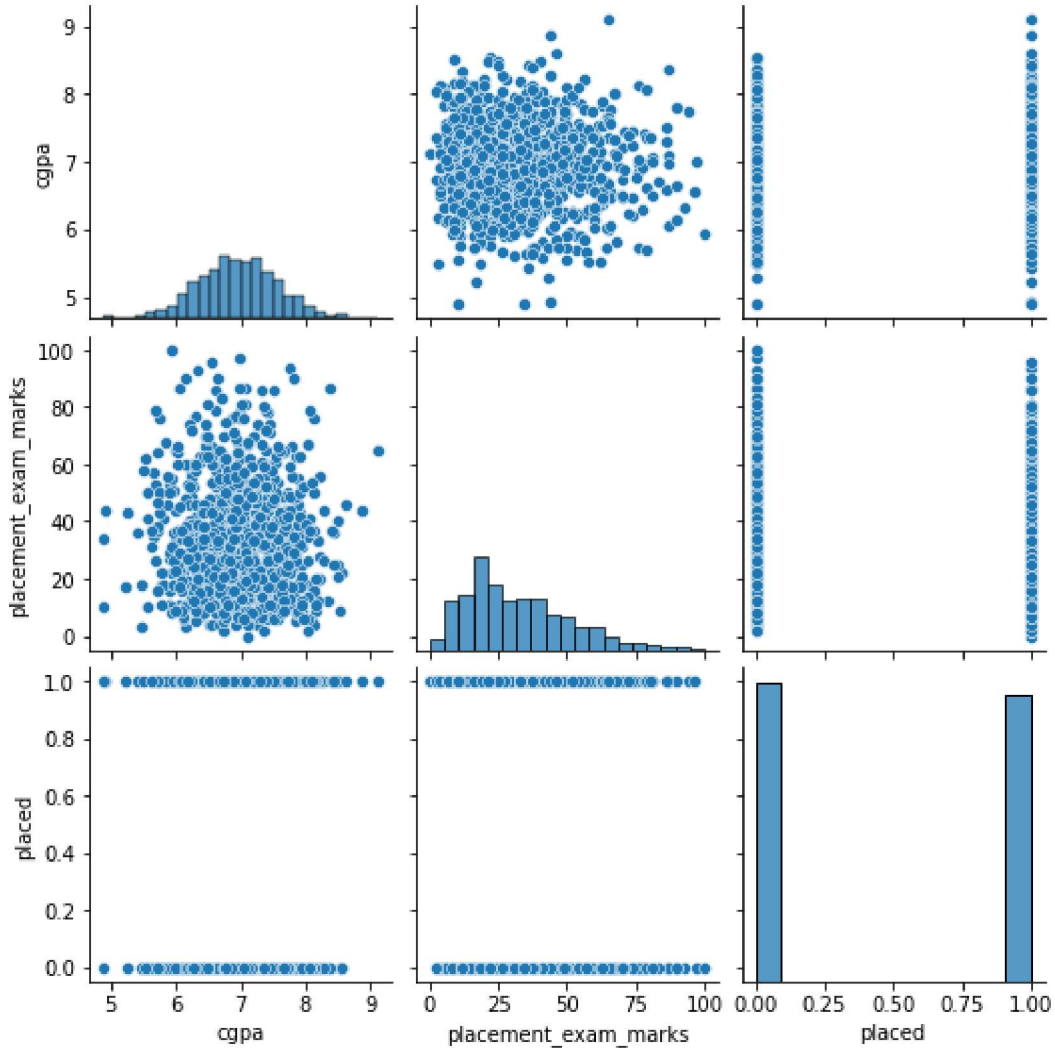
In [8]:
```
1 cols.columns
```

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

EDA and Visualization

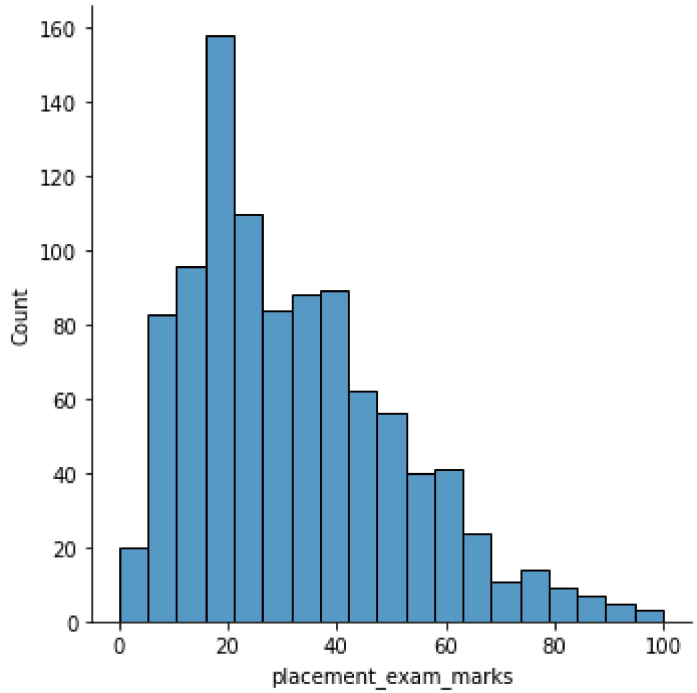
In [9]:
```
1 sns.pairplot(cols)
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x239544a6880>



In [10]:
```
1 sns.displot(df['placement_exam_marks'])
```

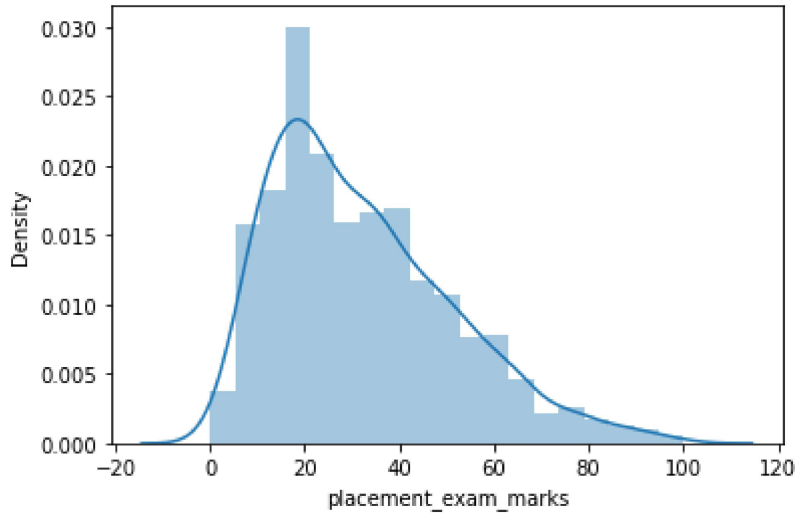
Out[10]: <seaborn.axisgrid.FacetGrid at 0x23956525700>



In [11]:
```
# We use displot in older version we get distplot use displot
sns.distplot(df['placement_exam_marks'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated 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 function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[11]: <AxesSubplot:xlabel='placement_exam_marks', ylabel='Density'>



In [12]:
```
df1=cols[['cgpa', 'placement_exam_marks', 'placed']]
df1
```

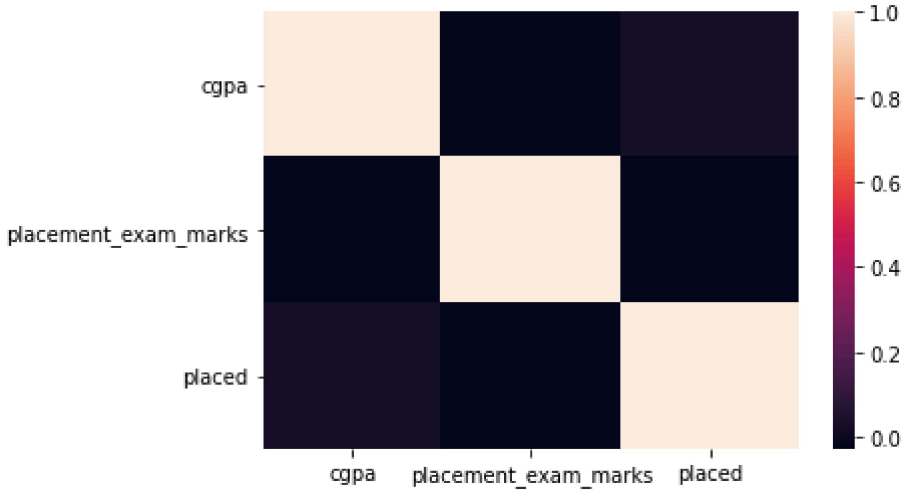
Out[12]:

	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 [13]:
```
1 sns.heatmap(df1.corr())
```

Out[13]: <AxesSubplot:>



To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [14]:
```
1 x=df1[['cgpa', 'placement_exam_marks', 'placed']]
2 y=df1[['placed']]
```

To split the dataset into test data

In [15]:
```
1 # importing lib for splitting test data
2 from sklearn.model_selection import train_test_split
```

In [16]:
```
1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [17]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[17]:
```
LinearRegression()
```

In [18]:
```
print(lr.intercept_)
```

```
[-1.11022302e-16]
```

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

```
1.0
```

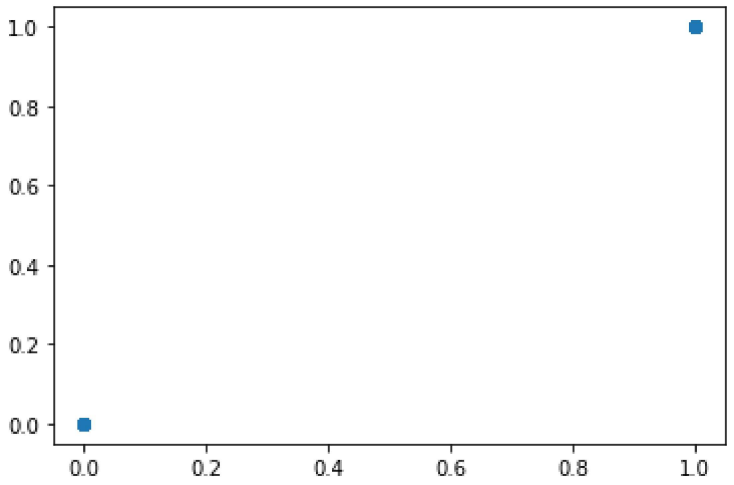
In [20]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[20]:

	0	1	2
0	1.508114e-18	-6.017229e-19	1.0

In [21]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x23956dfeee0>
```



In [22]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [23]:
```
rr=Ridge(alpha=20)
rr.fit(x_train,y_train)
```

Out[23]:
```
Ridge(alpha=20)
```

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

Out[24]:
```
0.9893931081876479
```

In [25]:
```
la=Lasso(alpha=20)
la.fit(x_train,y_train)
```

Out[25]:
```
Lasso(alpha=20)
```

In [26]:
```
la.score(x_test,y_test)
```

Out[26]:
```
-0.0036014405762305746
```

In [ ]:
```

```