Analysis and inferences of placement dataset of students:

1) **Replace the NaN values with the correct value. And justify why you have chosen the same.**

Step 1: Import pandas library "import pandas as pd"
Step 2: Read csv file, assign the file to the variable and call to the action.

```
import pandas as pd
```

```
dataset=pd.read_csv("Placement.csv")
```

```
dataset
```

Step 3: First, check the nan values present in which column and how many are present in each column.

```
dataset.isna().sum()
```

```
sl_no              0
gender             0
ssc_p              0
ssc_b              0
hsc_p              0
hsc_b              0
hsc_s              0
degree_p           0
degree_t           0
workex             0
etest_p            0
specialisation     0
mba_p              0
status             0
salary            67
dtype: int64
```

Step 4: separated the dataset into numerical and categorical data. Because only numerical data find the descriptive values.

```python
def Quanqual(dataset):
        qual=[]
        quan=[]
        for columnName in dataset.columns:
            if (dataset[columnName].dtypes=="O"):
                qual.append(columnName)
            else:
                quan.append(columnName)
        return quan,qual
```

```python
quan,qual=Quanqual(dataset)
```

```python
dataset[quan]
```

Step 5: Then find the descriptive statistics values of the dataset like mean, median, mode and IQR etc. To find the outliers are presented or not. If it is present, replace it by the lower whistler and upper whistler.

```python
def Univariate(dataset,quan):
    descriptive=pd.DataFrame(index=["mean","median","mode","Q1:25th","Q2:50th",
                                    "Q3:75th","99th","Q4:100th","IQR","min","max",
                                    "lower whister","upper whister","skewness","kurtosis","var","std"],columns=quan)
    for columnName in quan:
        descriptive[columnName]["mean"]=dataset[columnName].mean()
        descriptive[columnName]["median"]=dataset[columnName].median()
        descriptive[columnName]["mode"]=dataset[columnName].mode()[0]
        descriptive[columnName]["Q1:25th"]=dataset.describe()[columnName]["25%"]
        descriptive[columnName]["Q2:50th"]=dataset.describe()[columnName]["50%"]
        descriptive[columnName]["Q3:75th"]=dataset.describe()[columnName]["75%"]
        descriptive[columnName]["99th"]=np.percentile(dataset[columnName],99)
        descriptive[columnName]["Q4:100th"]=dataset.describe()[columnName]["max"]
        descriptive[columnName]["IQR"]=descriptive[columnName]["Q3:75th"]-descriptive[columnName]["Q1:25th"]
        descriptive[columnName]["min"]=dataset.describe()[columnName]["min"]
        descriptive[columnName]["max"]=dataset.describe()[columnName]["max"]
        descriptive[columnName]["lower whister"]=descriptive[columnName]["Q1:25th"]-(1.5*descriptive[columnName]["IQR"])
        descriptive[columnName]["upper whister"]=descriptive[columnName]["Q3:75th"]+(1.5*descriptive[columnName]["IQR"])
        descriptive[columnName]["skewness"]=dataset[columnName].skew()
        descriptive[columnName]["kurtosis"]=dataset[columnName].kurtosis()
        descriptive[columnName]["var"]=dataset[columnName].var()
        descriptive[columnName]["std"]=dataset[columnName].std()
    return descriptive
```

```python
Univariate(dataset,quan)
```

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
mean	108.0	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
median	108.0	67.0	65.0	66.0	71.0	62.0	265000.0
mode	1	62.0	63.0	65.0	60.0	56.7	300000.0
Q1:25th	54.5	60.6	60.9	61.0	60.0	57.945	240000.0
Q2:50th	108.0	67.0	65.0	66.0	71.0	62.0	265000.0
Q3:75th	161.5	75.7	73.0	72.0	83.5	66.255	300000.0
99th	212.86	87.0	91.86	83.86	97.0	76.1142	NaN
Q4:100th	215.0	89.4	97.7	91.0	98.0	77.89	940000.0
IQR	107.0	15.1	12.1	11.0	23.5	8.31	60000.0
min	1.0	40.89	37.0	50.0	50.0	51.21	200000.0
max	215.0	89.4	97.7	91.0	98.0	77.89	940000.0
lower whister	-106.0	37.95	42.75	44.5	24.75	45.48	150000.0
upper whister	322.0	98.35	91.15	88.5	118.75	78.72	390000.0
skewness	0.0	-0.132649	0.163639	0.244917	0.282308	0.313576	3.569747
kurtosis	-1.2	-0.60751	0.450765	0.052143	-1.08858	-0.470723	18.544273
var	3870.0	117.228377	118.755706	54.151103	176.251018	34.028376	8734295412.759695
std	62.209324	10.827205	10.897509	7.358743	13.275956	5.833385	93457.45242

```
def Outliers():
    L_outliers=[]
    G_outliers=[]
    for columnName in quan:
        if(descriptive[columnName]["min"]<descriptive[columnName]["lower whister"]):
            L_outliers.append(columnName)
        if(descriptive[columnName]["max"]>descriptive[columnName]["upper whister"]):
            G_outliers.append(columnName)
    return L_outliers,G_outliers
```

```
L_outliers,G_outliers= Outliers()
```

```
 L_outliers
```

```
['hsc_p']
```

```
G_outliers
```

```
['hsc_p', 'degree_p', 'salary']
```

```
for columnName in L_outliers:
    dataset[columnName][dataset[columnName]<descriptive[columnName]["lower whister"]]=descriptive[columnName]["lower whister
for columnName in G_outliers:
    dataset[columnName][dataset[columnName]>descriptive[columnName]["upper whister"]]=descriptive[columnName]["upper whister
```

Step 5: Because of outliers present in the quality dataset, replace the nan values with median.

```
from sklearn.impute import SimpleImputer
imp=SimpleImputer(missing_values=np.nan,strategy="median")
imp.fit(dataset[quan])
df=imp.transform(dataset[quan])
df=pd.DataFrame(df,columns=quan)
```

```
df
```

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
0	1.0	67.00	91.00	58.00	55.0	58.80	270000.0
1	2.0	79.33	78.33	77.48	86.5	66.28	200000.0
2	3.0	65.00	68.00	64.00	75.0	57.80	250000.0
3	4.0	56.00	52.00	52.00	66.0	59.43	265000.0
4	5.0	85.80	73.60	73.30	96.8	55.50	390000.0

```
preprocessed=pd.concat(dataset,axis=1)
```

```
preprocessed
```

```
preprocessed.to_csv("preplacement.csv")
```

Step 6: After preprocessing the dataset, save it in a csv file.

2) How many of them are not placed?

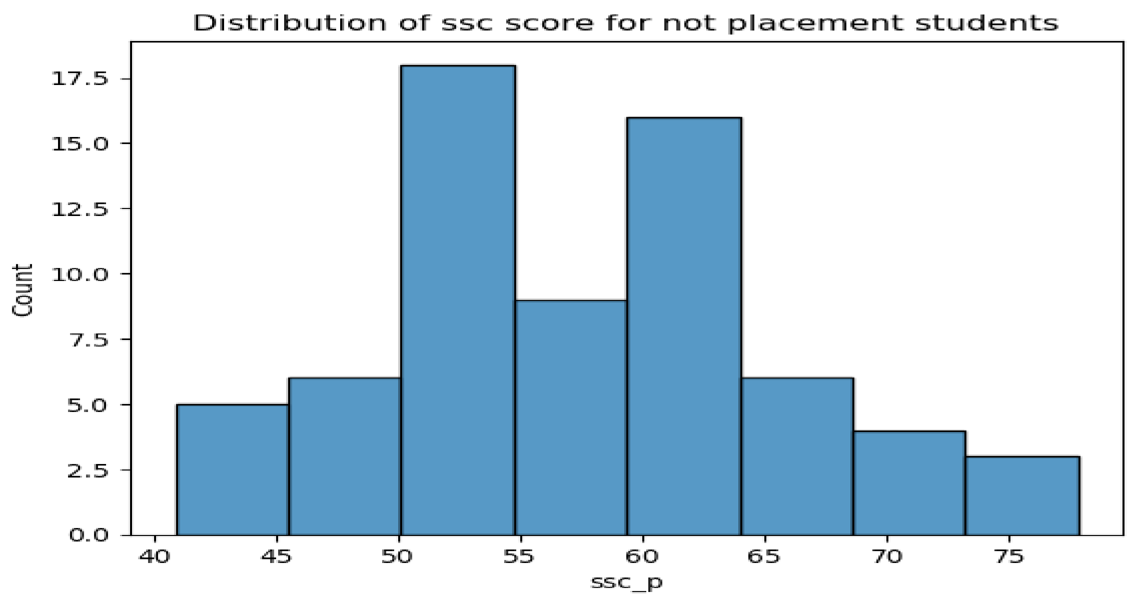
```
dataset["status"].value_counts()
```

```
Placed        148
Not Placed     67
Name: status, dtype: int64
```

Sixty seven of them are not placed.

3) Find the reason for non placement from the dataset?

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.histplot(data=df,x="ssc_p")
plt.title("Distribution of ssc score for not placement students")
plt.show()
```



Using histograms the most of the non placed students scored below the average score. It can also be one of the reasons for non placement of the students.

4) What kind of relation between salary and mba_p

```
dataset.corr()
```

	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
ssc_p	1.000000	0.513478	0.538686	0.261993	0.388478	0.174282
hsc_p	0.513478	1.000000	0.434606	0.240775	0.348452	0.127095
degree_p	0.538686	0.434606	1.000000	0.227147	0.402376	0.091780
etest_p	0.261993	0.240775	0.227147	1.000000	0.218055	0.231600
mba_p	0.388478	0.348452	0.402376	0.218055	1.000000	0.194934
salary	0.174282	0.127095	0.091780	0.231600	0.194934	1.000000

The MBA pass mark and Salary of the student are only 19% correlated. Because MBA park marks only 100 and the salary is in Lakhs.

5) Which specialization is getting a minimum salary?

```
average_salary_by_specialisation=dataset.groupby("specialisation")["salary"].mean()
```

```
average_salary_by_specialisation
```

```
specialisation
Mkt&Fin     278891.666667
Mkt&HR      267157.894737
Name: salary, dtype: float64
```

```
min_salary_specialization=average_salary_by_specialisation.idxmin()
```

```
min_salary_specialization
```

```
'Mkt&HR'
```

```
print(f"The specialization with the minimum average salary is:{min_salary_specialization}")
```

```
The specialization with the minimum average salary is:Mkt&HR
```

Mkt & HR specialization is getting a minimum salary.

6) How many of them are getting above 500,000 salaries?

```
4]:  df=dataset['salary']
```

```
5]:  df
```

```
5]: 0      270000.0
    1      200000.0
    2      250000.0
    3      265000.0
    4      390000.0
             ...
    210    390000.0
    211    275000.0
    212    295000.0
    213    204000.0
    214    265000.0
    Name: salary, Length: 215, dtype: float64
```

```
7]:  df.max()
```

```
7]: 390000.0
```

```
8]:  df[df>500000].count()
```

```
8]: 0
```

No, one is getting the above 5,00,000 salaries. Because I have removed the outliers.

```
In [43]: dfs=dataset['salary']
```

```
In [44]: dfs[dfs>500000].count()
Out[44]: 3
```

Three students are getting the above 500,000 salaries. It is from the not preprocessed dataset.

7) Test the Analysis of Variance between etest_p and mba_p at significance level 5%.(Make decisions using Hypothesis Testing).

```
import scipy.stats as stats
stats.f_oneway(dataset['etest_p'],dataset['mba_p'])
```

```
F_onewayResult(statistic=98.64487057324706, pvalue=4.672547689133573e-21)
```

null hypothesis H0

There is no differences between pass mark of etest and mba

alternative hypothesis H1

There is differences between pass mark of etest and mba

The calculated p-value is less than 0.05, we reject the null hypothesis, So that we conclude there are differences between pass marks of E-test and MBA.

8) **Test the similarity between the degree_t(Sci &Tech) and specialization(Mkt & HR) with respect to salary at a significance level of 5%.(Make decisions using Hypothesis Testing).**

Null hypothesis (H_0):

There is no significance between the degree_t(Sci &Tech) and specialization(Mkt & HR) with respect to salary.

Alternative hypothesis (H_a):

There is significance between the degree_t(Sci &Tech) and specialization(Mkt & HR) with respect to salary.

Level of significance: alpha= 0.05.

Test Statistic :

```
from scipy.stats import ttest_ind
degree_tST = dataset[dataset['degree_t']=="Sci&Tech"]["salary"]
specialisation= dataset[dataset['specialisation']=="Mkt&HR"]["salary"]
ttest_ind(degree_tST,specialisation)
```

```
Ttest_indResult(statistic=2.7397802052503586, pvalue=0.006884085818060l395)
```

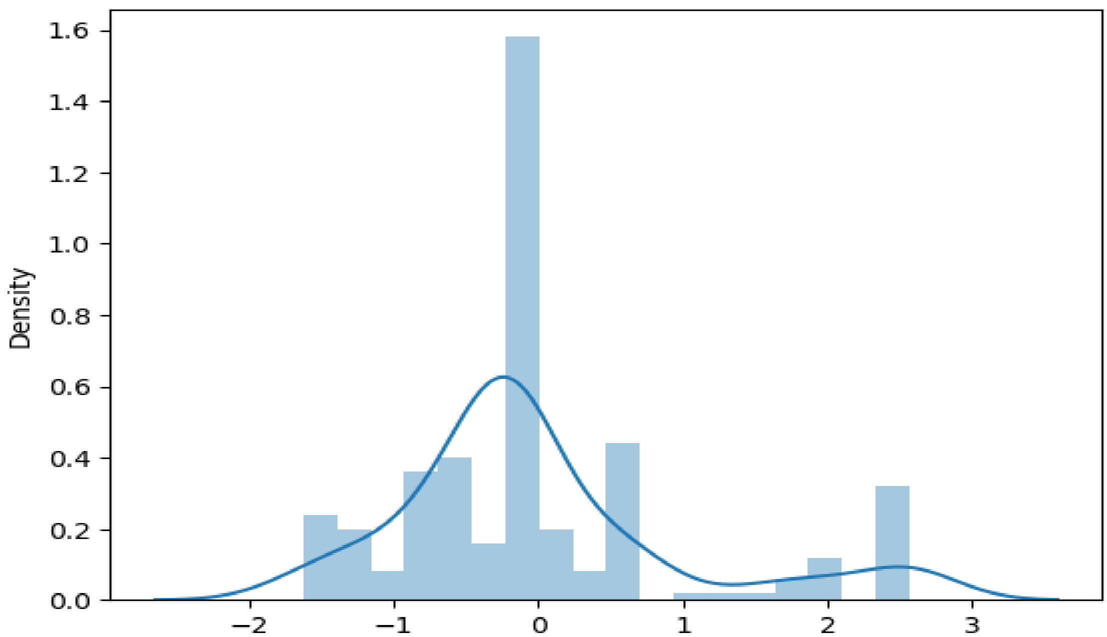
Inferences:

From the calculation p-value is less than 0.05, so that we reject the null hypothesis. There is significance between the degree_t(Sci &Tech) and specialization(Mkt & HR) with respect to salary.

9) Convert the normal distribution to standard normal distribution for the salary column.

```
def stdNBgraph(dataset):
    import seaborn as sns
    mean=dataset.mean()
    std=dataset.std()
    values=[i for i in dataset]
    z_score=[((j-mean)/std) for j in values]
    sns.distplot(z_score,kde=True)
    sum(z_score)/len(z_score)
```

```
stdNBgraph(dataset["salary"])
```



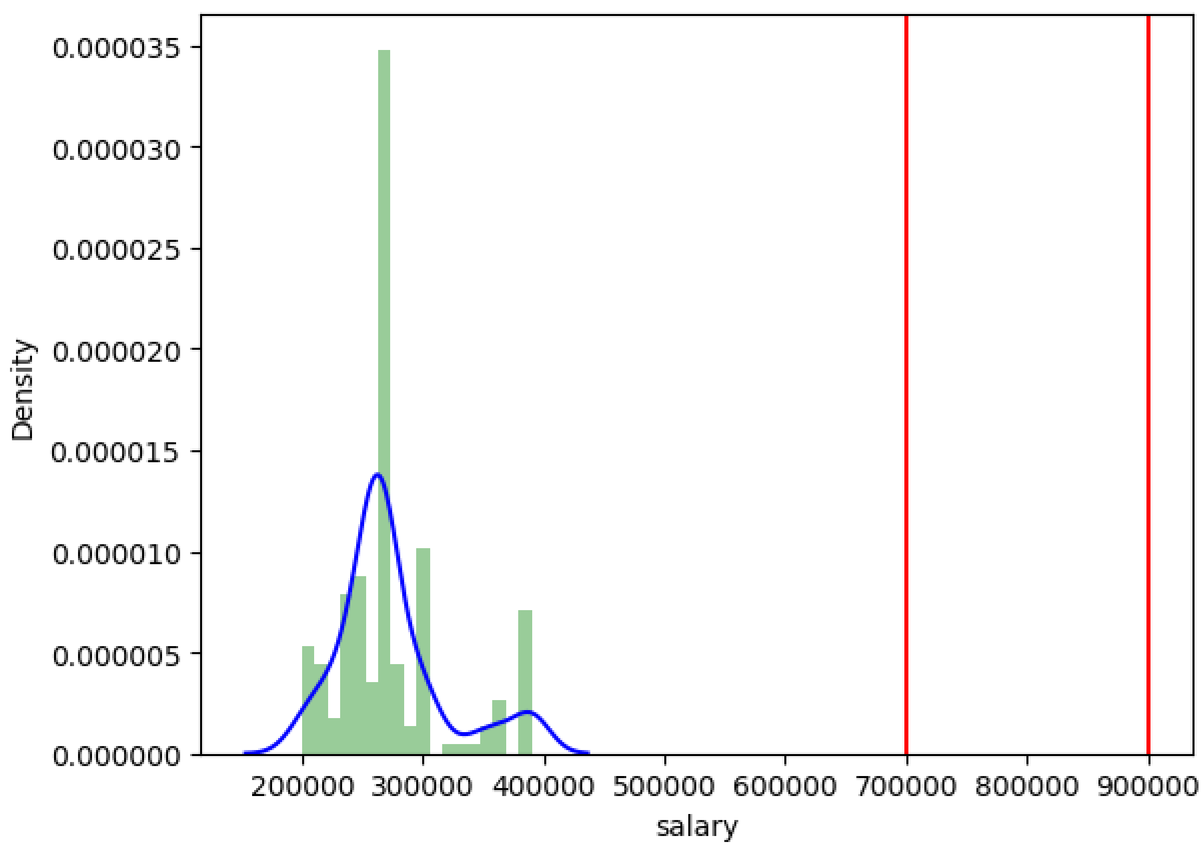
Normal distribution convert into standard normal distribution $N(\mu, \sigma^2) \rightarrow N(0, 1)$.

10) What is the probability Density Function of the salary range from 700000 to 900000?

```
def get_pdf_probability(dataset,startrange,endrange): # Create defining for probability density function
    from matplotlib import pyplot # import pyplot for diagram
    from scipy.stats import norm  # import norm for normal distribution
    import seaborn as sns # Seaborn is a well-known Python library for data visualization that offers a user-friendly interf
    ax=sns.distplot(dataset,kde=True,kde_kws={"color":"blue"},color="Green") # Drawn histogram and normal curve
    pyplot.axvline(startrange,color='Red') # vertical line axis drawn in startrange
    pyplot.axvline(endrange,color ='Red') # vertical line axis drawn in endrange
    sample=dataset # assige dataset sample
    sample_mean=sample.mean() # mean
    sample_std=sample.std() # S.D
    print("Mean=%.3f,Standard Deviation=%.3f"%(sample_mean,sample_std))
    dist=norm(sample_mean,sample_std)
    values=[value for value in range(startrange,endrange)] # one line for loop for create list
    probabilities=[dist.pdf(value) for value in values] # one line for loop for creat list
    prob=sum(probabilities) # adding
    print("The area between range({},{}):{}".format(startrange,endrange,sum(probabilities)))
    return prob
```

```
get_pdf_probability(dataset["salary"],700000,900000)
```

```
Mean=273706.977,Standard Deviation=45356.044
The area between range(700000,900000):2.7593660167492822e-21
```



11) **Test the similarity between the degree_t (Sci&Tech) with respect to etest_p and mba_p at significance level of 5%.(Make decisions using Hypothesis Testing).**

Null hypothesis (H0):

There is no significant difference between the degree_t (Sci&Tech) with respect to etest_p and mba_p.

Alternative hypothesis (Ha):

There is a significant difference between the degree_t (Sci&Tech) with respect to etest_p and mba_p.

Test statistics:

```
from scipy.stats import ttest_rel
degree_tet=dataset[dataset['degree_t']=="Sci&Tech"]["etest_p"]
degree_tmt=dataset[dataset['degree_t']=="Sci&Tech"]["mba_p"]
ttest_rel(degree_tet,degree_tmt)
```

```
Ttest_relResult(statistic=5.0049844583693615, pvalue=5.517920600505392e-06)
```

Inference:

From the calculation p-value is less than 0.05, so that we reject the null hypothesis. There is a significant difference between the degree_t (Sci&Tech) with respect to etest_p and mba_p.

12) Which parameter is highly correlated with salary?

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

def calc_vif(X):
    vif=pd.DataFrame()
    vif["variables"]=X.columns
    vif["VIF"]=[variance_inflation_factor(X.values,i)for i in range(X.shape[1])]
    return(vif)
```

```
calc_vif(dataset[['mba_p','salary']])
```

	variables	VIF
0	mba_p	34.233984
1	salary	34.233984

```
calc_vif(dataset[['ssc_p','salary']])
```

	variables	VIF
0	ssc_p	23.667333
1	salary	23.667333

```
calc_vif(dataset[['hsc_p','salary']])
```

	variables	VIF
0	hsc_p	22.569761
1	salary	22.569761

```
calc_vif(dataset[['degree_p','salary']])
```

	variables	VIF
0	degree_p	28.536543
1	salary	28.536543

```
calc_vif(dataset[['etest_p','salary']])
```

	variables	VIF
0	etest_p	22.178609
1	salary	22.178609

All parameters are highly correlated with salary. But the variable mba_p and salary have a high correlation.

13) plot any useful graph and explain it.

```
import matplotlib.pyplot as plt
import seaborn as sns
correlation_matrix = dataset.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```