In [62]:
```python
import pandas as pd
import seaborn as sns
import numpy as np
import scipy.stats as st
```

In [63]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [64]:
```python
df = pd.read_excel("D:\BPC\Python\Credit card project\default_of_credit_card_clients_0.xlsx")
df = pd.DataFrame(df)
df
```

Out[64]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2
0	1	20000	2	2	1	24	2	2	0	0	...	0	0	0	0	689
1	2	120000	2	2	2	26	0	2	0	0	...	3272	3455	3261	0	1000
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019
4	5	50000	1	2	1	57	0	0	0	0	...	20940	19146	19131	2000	36681
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	29996	220000	1	3	1	39	0	0	0	0	...	88004	31237	15980	8500	20000
29996	29997	150000	1	3	2	43	0	0	0	0	...	8979	5190	0	1837	3526
29997	29998	30000	1	2	2	37	4	3	2	0	...	20878	20582	19357	0	0
29998	29999	80000	1	3	1	41	1	0	0	0	...	52774	11855	48944	85900	3409
29999	30000	50000	1	2	1	46	0	0	0	0	...	36535	32428	15313	2078	1800

30000 rows × 25 columns

In [65]:
```python
df.describe()
```

Out[65]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	...	3
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.485500	0.356767	0.320033	0.304067	0.258767	...	4
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	9.217904	0.760594	0.801727	0.790589	0.761113	...	6
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000	0.000000	0.000000	0.000000	0.000000	...	-17
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000	0.000000	0.000000	0.000000	0.000000	...	
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.000000	0.000000	0.000000	...	1
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000	0.000000	0.000000	0.000000	0.000000	...	5
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000	8.000000	8.000000	8.000000	...	89

8 rows × 25 columns

In [66]:
```python
#Dropping the rows with 0, 5, 6 values in EDUCATION and 0 in MARRIAGE columns
df = df[df['MARRIAGE'] != 0]
df = df.query('EDUCATION != 0 and EDUCATION != 5 and EDUCATION != 6')

# Print modified DataFrame
print("\nModified DataFrame:")
df
```

```
Modified DataFrame:
```

Out[66]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2
0	1	20000	2	2	1	24	2	2	0	0	...	0	0	0	0	689
1	2	120000	2	2	2	26	0	2	0	0	...	3272	3455	3261	0	1000
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019
4	5	50000	1	2	1	57	0	0	0	0	...	20940	19146	19131	2000	36681
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	29996	220000	1	3	1	39	0	0	0	0	...	88004	31237	15980	8500	20000
29996	29997	150000	1	3	2	43	0	0	0	0	...	8979	5190	0	1837	3526
29997	29998	30000	1	2	2	37	4	3	2	0	...	20878	20582	19357	0	0
29998	29999	80000	1	3	1	41	1	0	0	0	...	52774	11855	48944	85900	3409
29999	30000	50000	1	2	1	46	0	0	0	0	...	36535	32428	15313	2078	1800

29601 rows × 25 columns

In [67]:
```python
#dropping unnecessary columns
df = df.drop(columns=['LIMIT_BAL', 'EDUCATION', 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6',
                      'default payment next month'])
df
```

Out[67]:

	ID	SEX	MARRIAGE	AGE	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT
0	1	2	1	24	3913	3102	689	0	0	0	0	689	0	
1	2	2	2	26	2682	1725	2682	3272	3455	3261	0	1000	1000	100
2	3	2	2	34	29239	14027	13559	14331	14948	15549	1518	1500	1000	100
3	4	2	1	37	46990	48233	49291	28314	28959	29547	2000	2019	1200	110
4	5	1	1	57	8617	5670	35835	20940	19146	19131	2000	36681	10000	900
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
29995	29996	1	1	39	188948	192815	208365	88004	31237	15980	8500	20000	5003	304
29996	29997	1	2	43	1683	1828	3502	8979	5190	0	1837	3526	8998	12
29997	29998	1	2	37	3565	3356	2758	20878	20582	19357	0	0	22000	420
29998	29999	1	1	41	-1645	78379	76304	52774	11855	48944	85900	3409	1178	192
29999	30000	1	1	46	47929	48905	49764	36535	32428	15313	2078	1800	1430	100

29601 rows × 16 columns

In [68]:
```python
#Calculating the total bill amount, total pay amount, total outstanding amount after 6 months for each row
df['TOT_BILL_AMT'] = df[['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6']].sum(axis=1)
df['TOT_PAY_AMT'] = df[['PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6']].sum(axis=1)
df['OUTS_AMT'] = df['TOT_BILL_AMT'] - df['TOT_PAY_AMT']
df
```

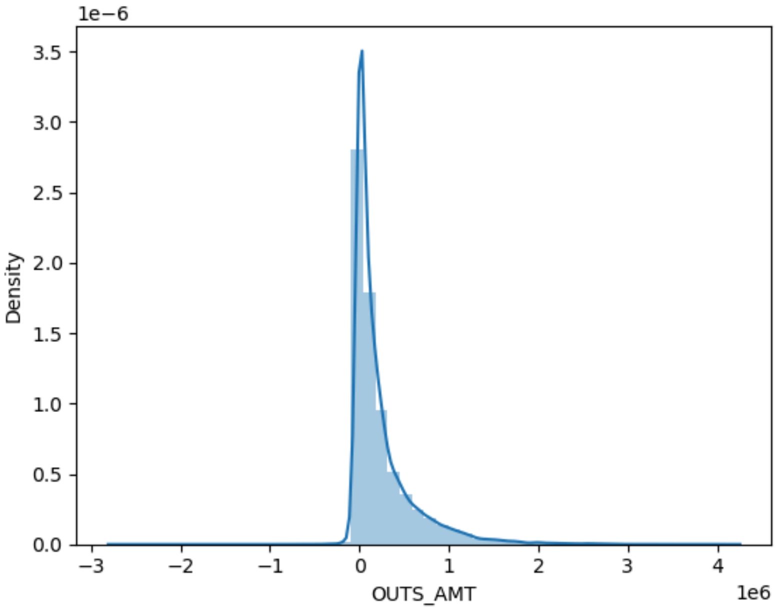
Out[68]:

	ID	SEX	MARRIAGE	AGE	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT
0	1	2	1	24	3913	3102	689	0	0	0	0	689	0	
1	2	2	2	26	2682	1725	2682	3272	3455	3261	0	1000	1000	100
2	3	2	2	34	29239	14027	13559	14331	14948	15549	1518	1500	1000	100
3	4	2	1	37	46990	48233	49291	28314	28959	29547	2000	2019	1200	110
4	5	1	1	57	8617	5670	35835	20940	19146	19131	2000	36681	10000	900
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
29995	29996	1	1	39	188948	192815	208365	88004	31237	15980	8500	20000	5003	304
29996	29997	1	2	43	1683	1828	3502	8979	5190	0	1837	3526	8998	12
29997	29998	1	2	37	3565	3356	2758	20878	20582	19357	0	0	22000	420
29998	29999	1	1	41	-1645	78379	76304	52774	11855	48944	85900	3409	1178	192
29999	30000	1	1	46	47929	48905	49764	36535	32428	15313	2078	1800	1430	100

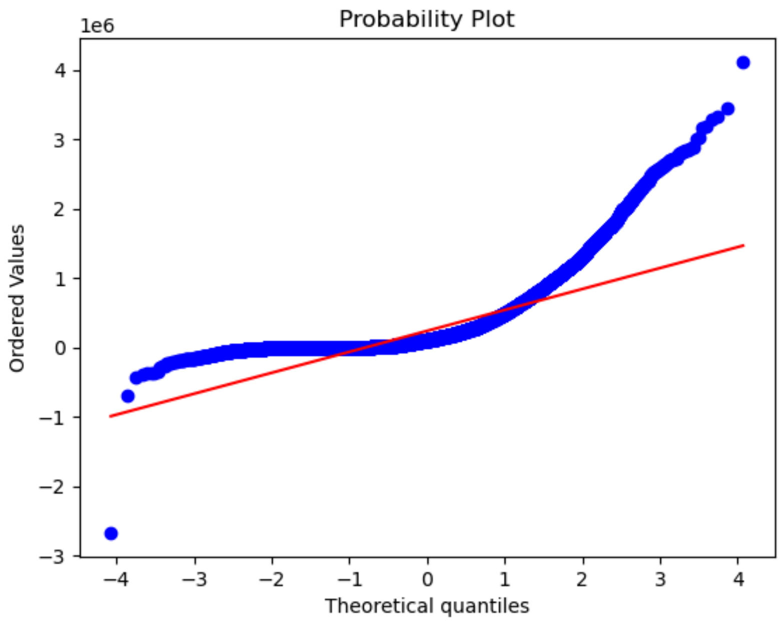
29601 rows × 19 columns

In [69]:
```python
sns.distplot(df['OUTS_AMT'])
df['OUTS_AMT'].mean()
```

Out[69]: 237371.5910611128



In [83]:
```
#Checking normal distribution with Q-Q plot

import matplotlib.pyplot as plt
st.probplot(df['OUTS_AMT'], dist = "norm", plot = plt)
plt.show()
```



In [84]:
```
median_value = np.median(df['OUTS_AMT'])
mode_value = st.mode(df['OUTS_AMT'])
print('Median: ', median_value)
print('Mode: ', mode_value)
```

```
Median:  101445.0
Mode:  ModeResult(mode=0, count=1382)
```

The dataset is related to the finance sector and it is very common to find outliers in the financial data. Removing those data points from the dataset, sometimes, erases important insights. Therefore, all 30,000 data points, in the dataset given, have been taken into account for the analysis. That has been considered to be the population size. From the above histogram, it is evident that the dataset does not follow the normal distribution. Sampling distribution of the mean with the sample size of 30 is to be considered, as per the central limit theorem, for further analyses.

In [85]:
```
#Selecting the required columns:
df1 = df.drop(columns=['BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1', 'PAY_AMT2',
                       'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6', 'TOT_BILL_AMT', 'TOT_PAY_AMT'])
df1
```

Out[85]:

	ID	SEX	MARRIAGE	AGE	OUTS_AMT
0	1	2	1	24	7015
1	2	2	2	26	12077
2	3	2	2	34	90635
3	4	2	1	37	222946
4	5	1	1	57	50290
...	...	...	...	...	...
29995	29996	1	1	39	682799
29996	29997	1	2	43	6692
29997	29998	1	2	37	39196
29998	29999	1	1	41	119430
29999	30000	1	1	46	222566

29601 rows × 5 columns

In [86]:
```python
# Create a list to store the sampled DataFrames
sampled_dfs = []

# Loop to create 30 random samples of 1000 data points each with different random states
for i in range(30):
    sample = df1.sample(1000, replace = False, random_state = i)
    sampled_dfs.append(sample)

# Access individual DataFrames
sample_1 = sampled_dfs[0]
sample_2 = sampled_dfs[1]
sample_3 = sampled_dfs[2]
sample_4 = sampled_dfs[3]
sample_5 = sampled_dfs[4]
sample_6 = sampled_dfs[5]
sample_7 = sampled_dfs[6]
sample_8 = sampled_dfs[7]
sample_9 = sampled_dfs[8]
sample_10 = sampled_dfs[9]
sample_11 = sampled_dfs[10]
sample_12 = sampled_dfs[11]
sample_13 = sampled_dfs[12]
sample_14 = sampled_dfs[13]
sample_15 = sampled_dfs[14]
sample_16 = sampled_dfs[15]
sample_17 = sampled_dfs[16]
sample_18 = sampled_dfs[17]
sample_19 = sampled_dfs[18]
sample_20 = sampled_dfs[19]
sample_21 = sampled_dfs[20]
sample_22 = sampled_dfs[21]
sample_23 = sampled_dfs[22]
sample_24 = sampled_dfs[23]
sample_25 = sampled_dfs[24]
sample_26 = sampled_dfs[25]
sample_27 = sampled_dfs[26]
sample_28 = sampled_dfs[27]
sample_29 = sampled_dfs[28]
sample_30 = sampled_dfs[29]
```

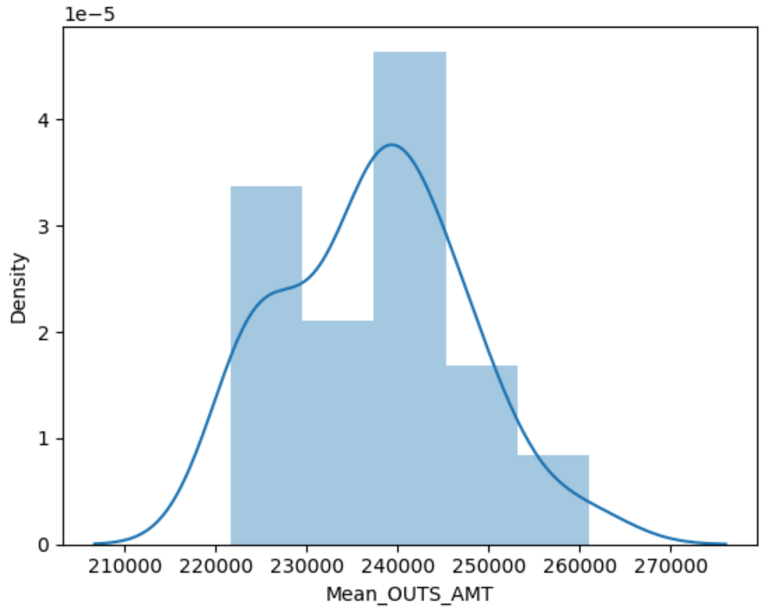
In [87]:
```python
# Calculate the mean of the 'OUTS_AMT' column for each DataFrame in the list
means_OUTS_AMT = [sample_df['OUTS_AMT'].mean() for sample_df in sampled_dfs]

# Convert the list of means into a DataFrame
means_df = pd.DataFrame(means_OUTS_AMT, columns=['Mean_OUTS_AMT'])
means_df
```

Out[87]:

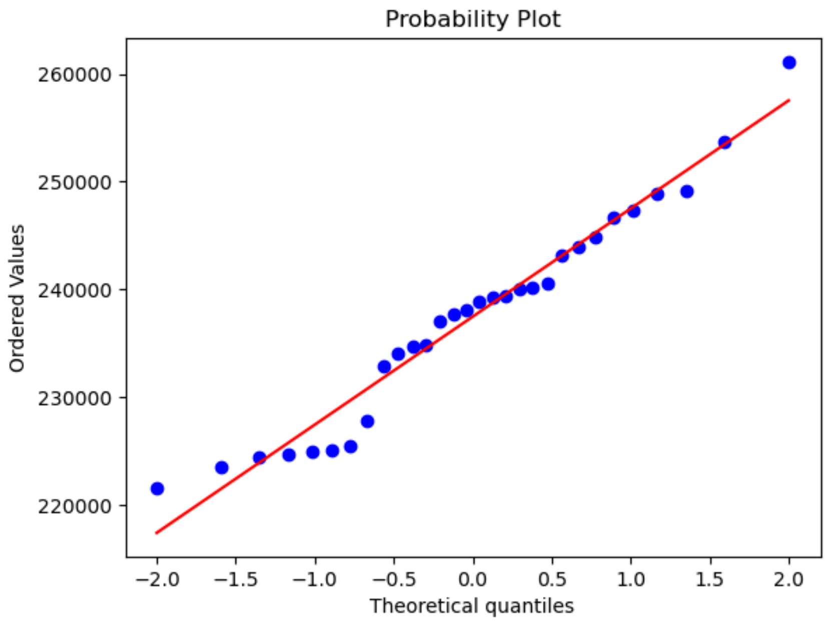
	Mean_OUTS_AMT
0	253688.460
1	240514.577
2	227829.955
3	224694.957
4	224479.294
5	223568.011
6	261116.397
7	238857.360
8	232892.483
9	238127.193
10	224889.958
11	243942.185
12	239269.395
13	237741.867
14	247268.326
15	240079.546
16	246726.758
17	234025.231
18	234817.365
19	221618.543
20	239334.356
21	244823.585
22	234688.100
23	225124.022
24	225502.796
25	243177.934
26	236977.487
27	249185.335
28	248928.618
29	240125.772

In [88]:
```python
sns.distplot(means_df['Mean_OUTS_AMT'])
```

Out[88]: <Axes: xlabel='Mean_OUTS_AMT', ylabel='Density'>



In [89]:
```
#Proving normal distribution with Q-Q plot

import matplotlib.pyplot as plt
st.probplot(means_df['Mean_OUTS_AMT'], dist = "norm", plot = plt)
plt.show()
```



Therefore, the data got a normal distribution and it has been used for further analyses using z-statistic. The mean of sampling distribution infers that the customers have positive outstanding amount i.e. they have more credits rather than debits.

CI of the Outstanding Amount

In [76]:
```
CI1 = st.norm.interval(confidence=0.95,
                 loc=np.mean(means_df['Mean_OUTS_AMT']))
print('Confidence Interval: ')
print(CI1)
```

```
Confidence Interval:
(237465.2355693488, 237469.15549731784)
```

It can be said from CI1, with 95% confidence level, that the outstanding amount of the debtors has a narrow margin of USD 233345.41 and USD 233349.33. Therefore, the chance to default in paying the credit back will increase if the outstanding amount of a debtor enters this interval.

Since the sampling distribution of the mean is normal z-statistic is used for analyses.

CI of the outstanding amount of male and female customers

In [77]:
```
#Taking 10% random samples each for male and female customers
male_samples = df1[df1['SEX'] == 1]
female_samples = df1[df1['SEX'] == 2]
rand_male = male_samples.sample(frac = 0.1, replace = False, random_state = i)
rand_female = female_samples.sample(frac = 0.1, replace = False, random_state = i)

print('10% random sampling of male population:')
print(rand_male)
print('*'*100)
print('10% random sampling of female population:')
print(rand_female)
print('*'*100)

#Confidence Interval
CI_male = st.norm.interval(confidence=0.95,
                 loc=np.mean(rand_male['OUTS_AMT']))
print('Confidence Interval for male sample: ')
print(CI_male)

CI_female = st.norm.interval(confidence=0.95,
                 loc=np.mean(rand_female['OUTS_AMT']))
print('Confidence Interval for female sample: ')
print(CI_female)
```

```
10% random sampling of male population:
          ID  SEX  MARRIAGE  AGE  OUTS_AMT
21029  21030    1         1   35    251195
13692  13693    1         1   47    301546
13803  13804    1         2   39    109566
23856  23857    1         2   31    285668
11688  11689    1         2   42    234467
...      ...  ...       ...  ...       ...
16907  16908    1         1   35      -830
23929  23930    1         2   23     63011
8253    8254    1         1   45    786329
29597  29598    1         2   44    653400
27196  27197    1         1   48      4376

[1175 rows x 5 columns]
****************************************************************************************************
10% random sampling of female population:
          ID  SEX  MARRIAGE  AGE  OUTS_AMT
25747  25748    2         1   28    504263
2142    2143    2         3   49    363482
19034  19035    2         1   42    149838
10041  10042    2         1   39     -2719
28752  28753    2         3   41    563952
...      ...  ...       ...  ...       ...
14578  14579    2         2   22     18356
9882    9883    2         1   47    132139
21951  21952    2         2   27       106
14102  14103    2         2   39       464
10090  10091    2         2   26    459703

[1786 rows x 5 columns]
****************************************************************************************************
Confidence Interval for male sample: 
(252704.42982324952, 252708.34975121857)
Confidence Interval for female sample: 
(221627.9666877512, 221631.88661572026)
```

CI of the outstanding amount of married and single customers

In [78]:
```
#Taking 10% random samples each for married and single customers
mar_samples = df1[df1['MARRIAGE'] == 1]
sing_samples = df1[df1['MARRIAGE'] == 2]
rand_mar = mar_samples.sample(frac = 0.1, replace = False, random_state = i)
rand_sing = sing_samples.sample(frac = 0.1, replace = False, random_state = i)

print('10% random sampling of married population:')
print(rand_mar)
print('*'*100)
print('10% random sampling of sing population:')
print(rand_sing)
print('*'*100)

#Confidence Interval
CI_mar = st.norm.interval(confidence=0.95,
                 loc=np.mean(rand_mar['OUTS_AMT']))
print('Confidence Interval for married sample: ')
print(CI_mar)

CI_sing = st.norm.interval(confidence=0.95,
                 loc=np.mean(rand_sing['OUTS_AMT']))
print('Confidence Interval for single sample: ')
print(CI_sing)
```

```
10% random sampling of married population:
          ID  SEX  MARRIAGE  AGE  OUTS_AMT
5097    5098    2         1   31    466257
17656  17657    2         1   44     91863
29541  29542    1         1   47     22410
7981    7982    2         1   40    303020
1262    1263    1         1   44    195261
...      ...  ...       ...  ...       ...
5810    5811    2         1   33      -896
21731  21732    2         1   26      1928
196      197    2         1   34         0
1775    1776    2         1   37     96904
8069    8070    1         1   47      5390

[1348 rows x 5 columns]
****************************************************************************************************
10% random sampling of sing population:
          ID  SEX  MARRIAGE  AGE  OUTS_AMT
7069    7070    1         2   38    298799
7702    7703    1         2   38      4756
2189    2190    2         2   22    109925
13001  13002    2         2   26     -2160
9293    9294    1         2   40     -6154
...      ...  ...       ...  ...       ...
16446  16447    2         2   54     96530
14478  14479    2         2   23    163979
21934  21935    2         2   27      -350
21810  21811    2         2   26    823243
19208  19209    2         2   30    596326

[1581 rows x 5 columns]
****************************************************************************************************
Confidence Interval for married sample: 
(234010.18766212824, 234014.1075900973)
Confidence Interval for single sample: 
(227217.61056099966, 227221.5304889687)
```

CI of the outstanding amount based on age

In [79]:
```
#2 age gropus are considered based on the age range between 21 and 79:
#Age interval: (79 - 21)/2 = 29
#2 age classes are: 21 - 50 and >50.
```

In [80]:
```
#Taking 10% random samples each for 21 - 50 and >50 yr old customers
young_samples = df1[df1['AGE'] <= 50]
old_samples = df1[df1['AGE'] >50]
rand_young = young_samples.sample(frac = 0.1, replace = False, random_state = i)
rand_old = old_samples.sample(frac = 0.1, replace = False, random_state = i)

print('10% random sampling of population aged between 21 and 50 yr:')
print(rand_young)
print('*'*100)
print('10% random sampling of population aged above 50 yr:')
print(rand_old)
print('*'*100)

#Confidence Interval
CI_young = st.norm.interval(confidence=0.95,
                 loc=np.mean(rand_young['OUTS_AMT']))
print('Confidence Interval for young sample: ')
print(CI_young)

CI_old = st.norm.interval(confidence=0.95,
                 loc=np.mean(rand_old['OUTS_AMT']))
print('Confidence Interval for old sample: ')
print(CI_old)
```

```
10% random sampling of population aged between 21 and 50 yr:
          ID  SEX  MARRIAGE  AGE  OUTS_AMT
11326  11327    2         2   29    374133
27453  27454    1         1   49    105505
29027  29028    2         2   41      -406
16982  16983    2         2   28     25855
1765    1766    2         2   27    143765
...      ...  ...       ...  ...       ...
17365  17366    2         2   33   1432948
11786  11787    1         2   27     32396
2186    2187    2         1   39     29871
4618    4619    1         2   29    331656
13698  13699    2         2   31     21830

[2737 rows x 5 columns]
****************************************************************************************************
10% random sampling of population aged above 50 yr:
          ID  SEX  MARRIAGE  AGE  OUTS_AMT
7445    7446    2         2   51    117030
25144  25145    1         1   55    125848
20227  20228    2         1   51    196036
11063  11064    2         1   52         0
18238  18239    1         1   52    143676
...      ...  ...       ...  ...       ...
14096  14097    1         1   61     79315
7932    7933    1         1   70         0
15128  15129    1         2   52     85792
8563    8564    1         1   52     20474
26564  26565    2         1   55    947168

[223 rows x 5 columns]
****************************************************************************************************
Confidence Interval for young sample:
(221602.84602797747, 221606.76595594652)
Confidence Interval for old sample:
(269463.20147099305, 269467.12139896216)
```