In [2]:

```
import pandas as pd
import numpy as np
```

You need to think about cleaning the data first. Common data problems include duplicates, missing, and errors in the data. Mark rows with data problems as “Missing” in the FICO column.

In [54]:

```
df = pd.read_csv('fico.csv')
df
```

Out[54]:

	acct_id	FICO
0	1	768
1	2	850
2	3	677
3	4	843
4	5	796
...	...	...
100008	99996	NaN
100009	99997	NaN
100010	99998	NaN
100011	99999	SSS
100012	100000	NaN

100013 rows × 2 columns

In [55]:

```
df.info(verbose=True, null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100013 entries, 0 to 100012
Data columns (total 2 columns):
 #   Column   Non-Null Count   Dtype 
---  ------   --------------   ----- 
 0   acct_id  100013 non-null  int64 
 1   FICO     70398 non-null   object
dtypes: int64(1), object(1)
memory usage: 1.5+ MB

<ipython-input-55-6ac79f7ef903>:1: FutureWarning: null_counts is deprecated.
Use show_counts instead
  df.info(verbose=True, null_counts=True)
```

In [56]:

```
df.describe()
```

Out[56]:

	acct_id
count	100013.000000
mean	49996.449842
std	28868.457180
min	1.000000
25%	24999.000000
50%	49994.000000
75%	74997.000000
max	100000.000000

In [57]:

```
# Finding the percentage of missing values in all columns",
round(df.isnull().mean()*100,2).sort_values(ascending = False)
```

Out[57]:

```
FICO       29.61
acct_id     0.00
dtype: float64
```

In [58]:

```
df = df.dropna()
```

In [59]:

```
df = df[df.FICO != 'SSS']
```

In [98]:

```
df = df[df.FICO != 'AA']
df.shape
```

Out[98]:

```
(70376, 2)
```

In [62]:

```
df = df.astype('str').astype('int')
```

In [99]:

```
# sorting by id
df.sort_values("acct_id", inplace = True)

# dropping ALL duplicate values
df.drop_duplicates(subset ="acct_id",
                     keep = False, inplace = True)
df
```

Out[99]:

	acct_id	FICO
0	1	768
1	2	850
2	3	677
3	4	843
4	5	796
...	...	...
99998	99986	836
99999	99987	850
100001	99989	850
100002	99990	830
100004	99992	850

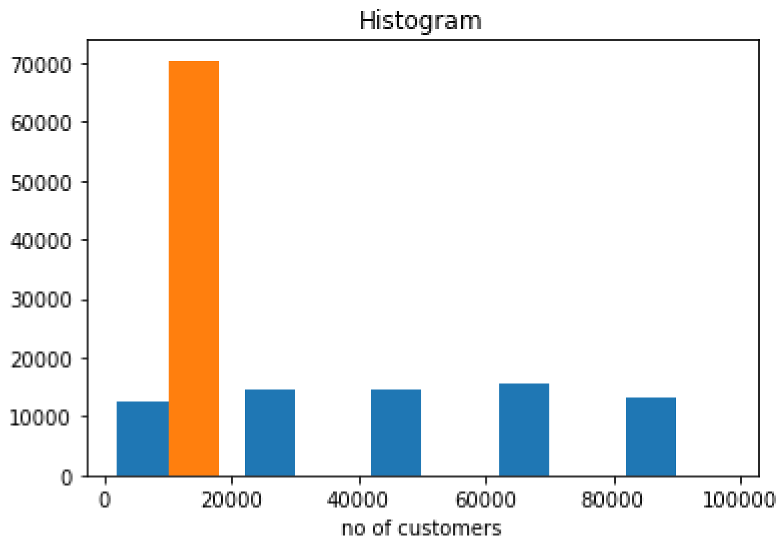
70376 rows × 2 columns

In [69]:

```
import matplotlib.pyplot as plt
plt.xlabel('no of customers')
plt.title('Histogram')
plt.hist(df,5)
plt.show
```

Out[69]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [70]:

```
round(df.isnull().mean()*100,2).sort_values(ascending = False)
```

Out[70]:

```
acct_id    0.0
FICO       0.0
dtype: float64
```

Now the whole FICO data is clean by removing all the Null and unwanted values.

Lets clean the region data

In [71]:

```
df1 = pd.read_csv('region.csv')
df1
```

Out[71]:

	acct_id	region
0	1	New York
1	2	Dallas
2	3	Los Angeles
3	4	Chicago
4	5	Philadelphia
...	...	...
100168	99996	Chicago
100169	99997	New York
100170	99998	San Diego
100171	99999	Chicago
100172	100000	Dallas

100173 rows × 2 columns

In [72]:

```
# Finding the percentage of missing values in all columns",
round(df1.isnull().mean()*100,2).sort_values(ascending = False)
```

Out[72]:

```
acct_id    0.0
region     0.0
dtype: float64
```

As we can see their is no NUll values present in region dataset.

In [73]:

```
# sorting by id
df1.sort_values("acct_id", inplace = True)

# dropping ALL duplicate values
df1.drop_duplicates(subset ="acct_id",
                     keep = False, inplace = True)
```

In [85]:

```
df1
```

Out[85]:

```
        acct_id       region
     0        1     New York
     1        2       Dallas
     2        3  Los Angeles
     3        4      Chicago
     4        5 Philadelphia
   ...      ...          ...
100168    99996      Chicago
100169    99997     New York
100170    99998    San Diego
100171    99999      Chicago
100172   100000       Dallas
```

99876 rows × 2 columns

Now both the data are cleaned

Create a temp table to store the information of FICO score and region for each customer.

In [101]:

```
frames = [df,df1]
result = pd.concat(frames, axis=1)
```

In [102]:

```
result = result.drop('acct_id', axis=1)
```

In [122]:

```
result.sort_values("FICO", inplace = True, ascending = False)
```

In [119]:

```
result.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100119 entries, 100004 to 100172
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   FICO    70376 non-null  float64
 1   region  99876 non-null  object 
dtypes: float64(1), object(1)
memory usage: 2.3+ MB
```

In [123]:

```
result = result[result.FICO != 850.0]
```

As we know the highest score is 850. after excluding 850 we get the second highest score is 849

In [146]:

```
result.dropna()
```

Out[146]:

	FICO	region
7606	420.0	New York
85879	433.0	Chicago
44588	439.0	Dallas
19168	440.0	Charlotte
70941	447.0	San Diego
...	...	...
43391	849.0	Los Angeles
89142	849.0	Dallas
19745	849.0	Houston
8739	849.0	Chicago
68543	849.0	Chicago

58125 rows × 2 columns

Avg score of all the regions

In [147]:

```
round(result.FICO.mean(),2)
```

Out[147]:

```
772.46
```