In [1]:
```
import numpy as np
import pandas as pd
```

In [19]:
```
df_region = pd.read_csv("/Users/charvigupta/Desktop/FDS_intern_written_test
df_region
```

Out[19]:

	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 [2]:
```
df_fico = pd.read_csv("/Users/charvigupta/Desktop/FDS_intern_written_test_d
df_fico
```

Out[2]:

	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 [3]:
```
df_fico.info()
```

```
<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
```

In [4]:
```
df_fico['FICO'].value_counts().index.values.tolist()
```

Out[4]:
```
['850',
 '836',
 '845',
 '828',
 '841',
 '820',
 '839',
 '842',
 '838',
 '834',
 '837',
 '830',
 '827',
 '835',
 '844',
 '840',
 '831',
 '811',
 '849',
```

In [5]:
```
## Finding Duplicates
duplicate_indices = df_fico[df_fico['acct_id'].duplicated()].index.values.t
```

In [6]:
```
duplicate_indices
```

Out[6]:
```
[140,
 190,
 192,
 194,
 196,
 30506,
 30508,
 30510,
 30512,
 30514,
 30516,
 30518,
 30520]
```

all duplicates have the same values, so we can drop them

In [7]:
```
# df_fico_deduplicated = df_fico.drop_duplicates(subset=['acct_id'])
# df_fico_deduplicated.shape
# sorted(df_fico_deduplicated['FICO'].value_counts().index.values)
```

In [8]:
```
### Missing values
missing_indices = df_fico[df_fico['FICO'].isna()==True].index.values.tolist
missing_indices
```

Out[8]:
```
[10,
 17,
 40,
 42,
 43,
 45,
 46,
 47,
 56,
 58,
 63,
 64,
 68,
 84,
 91,
 103,
 106,
 113,
 119,
```

In [9]:
```
# Find error values
error_indices = df_fico[(df_fico['FICO']=='AA')|(df_fico['FICO']=='SSS')].i
```

In [10]:
```
# Total error indices =

all_error_indices = sorted(list(set(error_indices).union(set(duplicate_indi
```

In [11]:
```
len(all_error_indices)
```

Out[11]:
```
29627
```

In [ ]:
```
def apply_missing(df):
    for index, row in df.iterrows():
        if index in all_error_indices:
            df.at[index, 'FICO'] = "Missing"
    return df
```

In [ ]:
```
df_result = apply_missing(df_fico.copy())
```

In [ ]:
```
df_result.to_csv("q1_1.csv", index=False)
```

Part 1.2

In [129]:
```
df_fico.shape
```

Out[129]:
```
(100013, 2)
```

In [130]:
```
# removing duplicate entries

df_fico[df_fico['acct_id'].isin(df_fico['acct_id'][df_fico['acct_id'].dupli
```

Out[130]:

	acct_id	FICO
139	140	752
140	140	752
189	189	745
190	189	745
191	190	828
192	190	828
193	191	752
194	191	752
195	192	850
196	192	850
30505	30501	773
30506	30501	773
30508	30502	798
30507	30502	798
30509	30503	820
30510	30503	820
30511	30504	NaN
30512	30504	NaN
30513	30505	820
30514	30505	820
30515	30506	NaN
30516	30506	NaN
30517	30507	NaN
30518	30507	NaN
30519	30508	850
30520	30508	850

In [131]:
```
df_fico_deduplicated = df_fico.drop_duplicates(subset=['acct_id'])
df_fico_deduplicated = df_fico_deduplicated.reset_index(drop=True)
```

In [132]:
```
df_fico_deduplicated['acct_id']
```

Out[132]:
```
0              1
1              2
2              3
3              4
4              5
            ...
99995      99996
99996      99997
99997      99998
99998      99999
99999     100000
Name: acct_id, Length: 100000, dtype: int64
```

In [133]:
```
df_fico_deduplicated
```

Out[133]:

	acct_id	FICO
0	1	768
1	2	850
2	3	677
3	4	843
4	5	796
...	...	...
99995	99996	NaN
99996	99997	NaN
99997	99998	NaN
99998	99999	SSS
99999	100000	NaN

100000 rows × 2 columns

In [134]:
```
df_fico_deduplicated['acct_id'].isna().sum()
```

Out[134]:
```
0
```

In [135]:
```
df_region_deduplicated = df_region.drop_duplicates(subset=['acct_id'])
df_region_deduplicated = df_region_deduplicated.reset_index(drop=True)
```

In [136]:
```
# df_region[df_region['acct_id'].isin(df_region['acct_id'][df_region['acct_
```

In [137]:
```
df_region_deduplicated
```

Out[137]:

	acct_id	region
0	1	New York
1	2	Dallas
2	3	Los Angeles
3	4	Chicago
4	5	Philadelphia
...	...	...
99995	99996	Chicago
99996	99997	New York
99997	99998	San Diego
99998	99999	Chicago
99999	100000	Dallas

100000 rows × 2 columns

In [138]:
```
df_region_deduplicated.shape, df_fico_deduplicated.shape
```

Out[138]:
```
((100000, 2), (100000, 2))
```

In [ ]:
```

```

In [139]:
```
df_region_deduplicated.dtypes, df_fico_deduplicated.dtypes
```

Out[139]:
```
(acct_id     int64
 region     object
 dtype: object,
 acct_id     int64
 FICO       object
 dtype: object)
```

In [140]:
```
df_region_deduplicated['acct_id'].isin(df_fico_deduplicated['acct_id'])
```

Out[140]:
```
0         True
1         True
2         True
3         True
4         True
          ...
99995     True
99996     True
99997     True
99998     True
99999     True
Name: acct_id, Length: 100000, dtype: bool
```

In [141]:
```
result = pd.merge(df_fico_deduplicated, df_region_deduplicated, how="left")
```

In [142]:
```
result
```

Out[142]:

	acct_id	FICO	region
0	1	768	New York
1	2	850	Dallas
2	3	677	Los Angeles
3	4	843	Chicago
4	5	796	Philadelphia
...	...	...	...
99995	99996	NaN	Chicago
99996	99997	NaN	New York
99997	99998	NaN	San Diego
99998	99999	SSS	Chicago
99999	100000	NaN	Dallas

100000 rows × 3 columns

In [72]:
```
## temp Drop rows with missing values in FICO
```

In [143]:
```
res_temp = result.dropna(subset='FICO')
print(result.shape, res_temp.shape)
```

```
(100000, 3) (70388, 3)
```

In [144]:
```
res_temp2 = res_temp[(res_temp['FICO']!='AA')]
res_temp3 = res_temp2[(res_temp2['FICO']!='SSS')]

print(res_temp2.shape, res_temp.shape)
```

```
(70387, 3) (70388, 3)
```

In [145]:
```
res_temp3['FICO'] = res_temp3['FICO'].astype('int64')
```

```
/var/folders/qb/wmdrw3xx75vd1jpzfxx7n_mh0000gn/T/ipykernel_64677/12649617
30.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  res_temp3['FICO'] = res_temp3['FICO'].astype('int64')
```

In [146]:
```
comp = (res_temp3.groupby(['region'], as_index=False)['FICO'].mean())
```

In [148]:
```
comp['FICO'] = comp['FICO'].astype(int)
```

In [149]:
```
comp
```

Out[149]:

	region	FICO
0	Charlotte	785
1	Chicago	785
2	Dallas	783
3	Houston	786
4	Los Angeles	789
5	New York	785
6	Philadelphia	785
7	Phoenix	788
8	San Antonio	784
9	San Diego	784

In [ ]:
```

```

In [150]:
```
def fill_avg_region_fico(df, comp):
    for index, row in df.iterrows():
        try:
            if df.at[index, 'FICO'].isnumeric:
                df.at[index, 'FICO'] = int(df.at[index, 'FICO'])
            else:
                reg = df.at[index, 'region']
                df.at[index, 'FICO'] = comp[comp['region']==reg]['FICO'].va
        except:
            reg = df.at[index, 'region']
            df.at[index, 'FICO'] = comp[comp['region']==reg]['FICO'].values
    return df
```

In [151]:
```
result
```

Out[151]:

	acct_id	FICO	region
0	1	768	New York
1	2	850	Dallas
2	3	677	Los Angeles
3	4	843	Chicago
4	5	796	Philadelphia
...	...	...	...
99995	99996	NaN	Chicago
99996	99997	NaN	New York
99997	99998	NaN	San Diego
99998	99999	SSS	Chicago
99999	100000	NaN	Dallas

100000 rows × 3 columns

In [152]:
```
final1 = fill_avg_region_fico(result.copy(), comp)
final1
```

Out[152]:

	acct_id	FICO	region
0	1	768	New York
1	2	850	Dallas
2	3	677	Los Angeles
3	4	843	Chicago
4	5	796	Philadelphia
...	...	...	...
99995	99996	785	Chicago
99996	99997	785	New York
99997	99998	784	San Diego
99998	99999	785	Chicago
99999	100000	783	Dallas

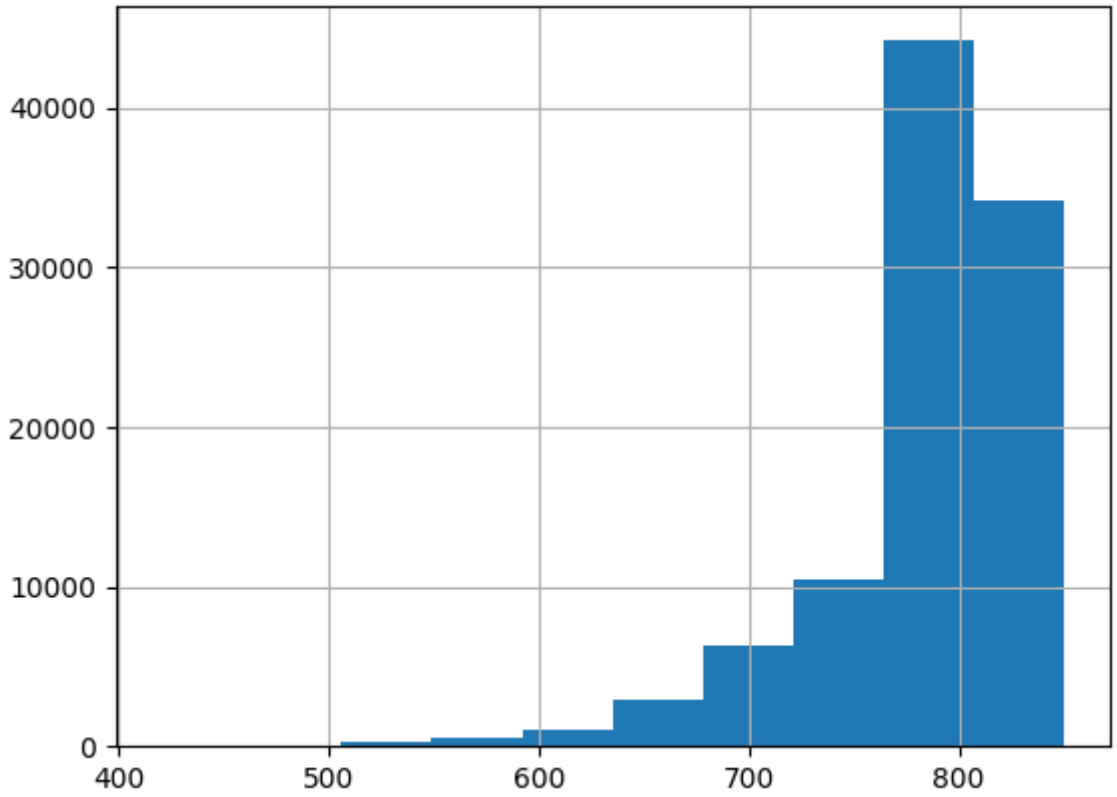
100000 rows × 3 columns

In [159]:
```
final1['FICO'] = final1['FICO'].astype(int)
```

In [160]:
```
final1.to_csv("q1_3.csv", index=False)
```

In [161]:
```
final1['FICO'].hist()
```

Out[161]: <AxesSubplot: >



In [162]:
```
final1['FICO'].dtype
```

Out[162]: dtype('int64')

In [168]:
```
pd.qcut(final1['FICO'], q=5, labels=['A', 'B', 'C', 'D', 'E'])
```

Out[168]:
```
0        B
1        E
2        A
3        E
4        D
        ..
99995    B
99996    B
99997    B
99998    B
99999    B
Name: FICO, Length: 100000, dtype: category
Categories (5, object): ['A' < 'B' < 'C' < 'D' < 'E']
```

In [171]:
```
pd.qcut(final1['FICO'], q=5)
```

Out[171]:
```
0          (758.0, 785.0]
1          (836.0, 850.0]
2        (419.999, 758.0]
3          (836.0, 850.0]
4          (791.0, 836.0]
                 ...
99995      (758.0, 785.0]
99996      (758.0, 785.0]
99997      (758.0, 785.0]
99998      (758.0, 785.0]
99999      (758.0, 785.0]
Name: FICO, Length: 100000, dtype: category
Categories (5, interval[float64, right]): [(419.999, 758.0] < (758.0, 78
5.0] < (785.0, 791.0] < (791.0, 836.0] < (836.0, 850.0]]
```

In [172]:
```
final1['cut'] = pd.qcut(final1['FICO'], q=5)
final1['cut_bucket'] = pd.qcut(final1['FICO'], q=5, labels=['A', 'B', 'C',

```

In [173]:
```
final1
```

Out[173]:

	acct_id	FICO	region	cut	cut_bucket
0	1	768	New York	(758.0, 785.0]	B
1	2	850	Dallas	(836.0, 850.0]	E
2	3	677	Los Angeles	(419.999, 758.0]	A
3	4	843	Chicago	(836.0, 850.0]	E
4	5	796	Philadelphia	(791.0, 836.0]	D
...	...	...	...	...	...
99995	99996	785	Chicago	(758.0, 785.0]	B
99996	99997	785	New York	(758.0, 785.0]	B
99997	99998	784	San Diego	(758.0, 785.0]	B
99998	99999	785	Chicago	(758.0, 785.0]	B
99999	100000	783	Dallas	(758.0, 785.0]	B

100000 rows × 5 columns

In [212]:
```
ans = {}
keys = sorted(final1['cut'].value_counts().index.tolist())
values = sorted(final1['cut'].value_counts().values.tolist())

final1['cut'].value_counts()
```

Out[212]:
```
(758.0, 785.0]      29137
(791.0, 836.0]      20646
(419.999, 758.0]    20233
(836.0, 850.0]      19226
(785.0, 791.0]      10758
Name: cut, dtype: int64
```

In [213]:
```
ans = {str(k):int(v) for k,v in zip(keys, values)}
```

In [214]:
```
sorted(final1['cut'].value_counts().index)
```

Out[214]:
```
[Interval(419.999, 758.0, closed='right'),
 Interval(758.0, 785.0, closed='right'),
 Interval(785.0, 791.0, closed='right'),
 Interval(791.0, 836.0, closed='right'),
 Interval(836.0, 850.0, closed='right')]
```

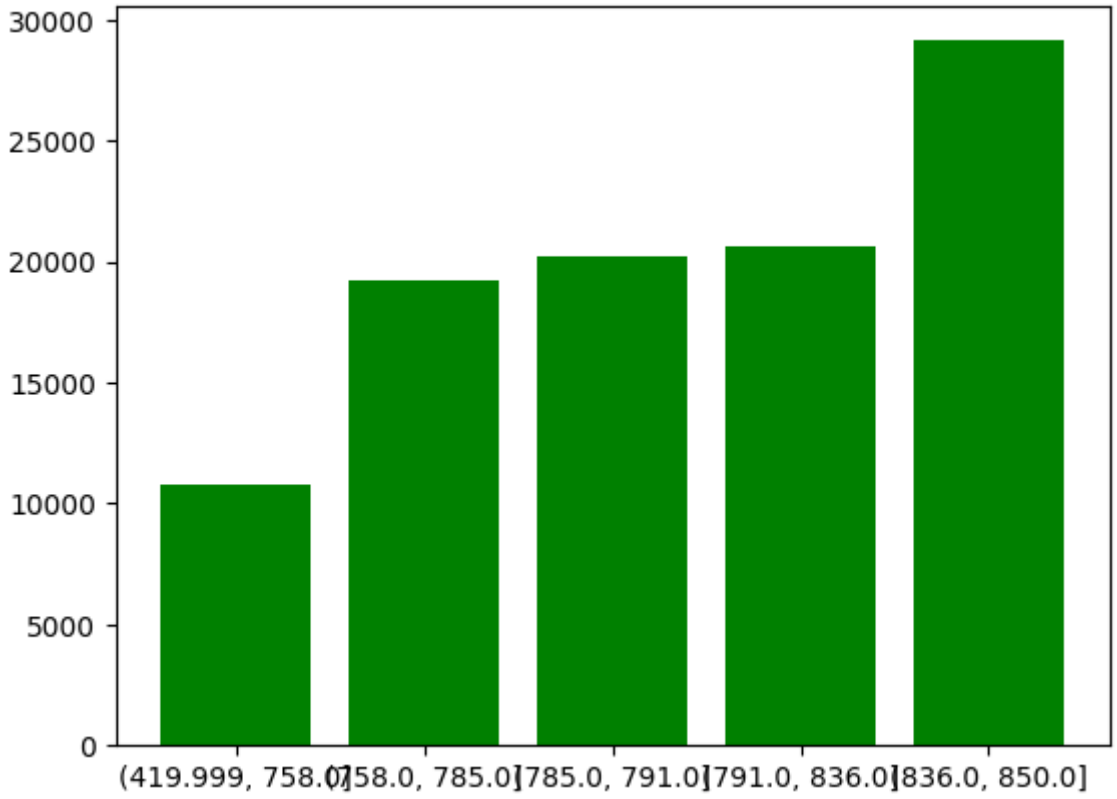
In [215]:
```
ans
```

Out[215]:
```
{'(419.999, 758.0]': 10758,
 '(758.0, 785.0]': 19226,
 '(785.0, 791.0]': 20233,
 '(791.0, 836.0]': 20646,
 '(836.0, 850.0]': 29137}
```

In [198]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [216]:
```
plt.bar(list(ans.keys()), ans.values(), color='g')
```

Out[216]:
```
<BarContainer object of 5 artists>
```



In [ ]:

In [ ]: