数据探索性分析与数据预处理

1. 问题描述

自行选择2个数据集进行探索性分析与预处理。

2. 数据集

所选数据集：Chicago Building Violations

- building-violations.csv

In [1]:
```python
#导入库
import matplotlib
import numpy as np
import pandas as pd
```

In [3]:
```python
#读取数据
path = "../data/building_violations/building-violations.csv"
data = pd.read_csv(path)
```

In [4]:
```python
data.head()
```

Out[4]:

	ID	VIOLATION LAST MODIFIED DATE	VIOLATION DATE	VIOLATION CODE	VIOLATION STATUS	VIOLATION STATUS DATE	VIOLATION DESCRIPTION	VIOLATION LOCATION	VIOLATION INSPECTOR COMMENTS	VIOL. ORDIN
0	6392482	2019-12-04T12:40:09.000	2019-12-04T00:00:00.000	CN196019	OPEN	NaN	NO POSTED ADDRESS	OTHER : :OTHER	BUILDING ADRESSES - INCOMPLETE WITH MISSING #'S.	Post a of buil consp p
1	6392480	2019-12-04T12:40:09.000	2019-12-04T00:00:00.000	CN061014	OPEN	NaN	REPAIR EXTERIOR WALL	OTHER : :OTHER	WEST AND SOUTH ELEVATIONS / EXTERIOR WALLS - M...	Fa maint exterio of
2	6392335	2019-12-04T14:00:12.000	2019-12-04T00:00:00.000	CN138106	OPEN	NaN	STOP/REMOVE NUISANCE	OTHER : :OTHER	YARD AREA; ABANDON VEHICLE.	Remo nuisan 28
3	6391883	2019-12-04T08:32:01.000	2019-12-04T00:00:00.000	CN197039	OPEN	NaN	RELOCATE SMOKE DETECTOR	INTERIOR:003 :	HALLWAY - SMOKE DETECTOR - 4FT BELOW CEILING.	Re impr in detec
4	6392369	2019-12-04T14:14:24.000	2019-12-04T00:00:00.000	CN065034	OPEN	NaN	REPAIR WINDOW SILLS	OTHER : :OTHER	EAST WINDOW SILLS; OPEN JOINTS.	Fa m windov good

5 rows × 32 columns

In [5]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1677788 entries, 0 to 1677787
Data columns (total 32 columns):
 #   Column                        Non-Null Count    Dtype  
---  ------                        --------------    -----  
 0   ID                            1677788 non-null  int64  
 1   VIOLATION LAST MODIFIED DATE  1677788 non-null  object 
 2   VIOLATION DATE                1677788 non-null  object 
 3   VIOLATION CODE                1677788 non-null  object 
 4   VIOLATION STATUS              1677788 non-null  object 
 5   VIOLATION STATUS DATE         641589 non-null   object 
 6   VIOLATION DESCRIPTION         1667020 non-null  object 
 7   VIOLATION LOCATION            780506 non-null   object 
 8   VIOLATION INSPECTOR COMMENTS  1502325 non-null  object 
 9   VIOLATION ORDINANCE           1630207 non-null  object 
 10  INSPECTOR ID                  1677788 non-null  object 
 11  INSPECTION NUMBER             1677788 non-null  int64  
 12  INSPECTION STATUS             1677772 non-null  object 
 13  INSPECTION WAIVED             1677788 non-null  object 
 14  INSPECTION CATEGORY           1677788 non-null  object 
 15  DEPARTMENT BUREAU             1677788 non-null  object 
 16  ADDRESS                       1677788 non-null  object 
 17  STREET NUMBER                 1677788 non-null  int64  
 18  STREET DIRECTION              1677788 non-null  object 
 19  STREET NAME                   1677788 non-null  object 
 20  STREET TYPE                   1664247 non-null  object 
 21  PROPERTY GROUP                1677788 non-null  int64  
 22  SSA                           321521 non-null   float64
 23  LATITUDE                      1676278 non-null  float64
 24  LONGITUDE                     1676278 non-null  float64
 25  LOCATION                      1676278 non-null  object 
 26  Community Areas               1675509 non-null  float64
 27  Zip Codes                     1676278 non-null  float64
 28  Boundaries - ZIP Codes        1675509 non-null  float64
 29  Census Tracts                 1676243 non-null  float64
 30  Wards                         1675509 non-null  float64
 31  Historical Wards 2003-2015    1675509 non-null  float64
dtypes: float64(9), int64(4), object(19)
memory usage: 409.6+ MB
```

去除显然为独特信息，无法依靠统计填补缺失的ID、VIOLATION DESCRIPTION、VIOLATION INSPECTOR COMMENTS列数据。

In [6]:
```python
data = data.drop(columns=['ID', 'VIOLATION DESCRIPTION', 'VIOLATION INSPECTOR COMMENTS'])

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1677788 entries, 0 to 1677787
Data columns (total 29 columns):
 #   Column                        Non-Null Count    Dtype  
---  ------                        --------------    -----  
 0   VIOLATION LAST MODIFIED DATE  1677788 non-null  object 
 1   VIOLATION DATE                1677788 non-null  object 
 2   VIOLATION CODE                1677788 non-null  object 
 3   VIOLATION STATUS              1677788 non-null  object 
 4   VIOLATION STATUS DATE         641589 non-null   object 
 5   VIOLATION LOCATION            780506 non-null   object 
 6   VIOLATION ORDINANCE           1630207 non-null  object 
 7   INSPECTOR ID                  1677788 non-null  object 
 8   INSPECTION NUMBER             1677788 non-null  int64  
 9   INSPECTION STATUS             1677772 non-null  object 
 10  INSPECTION WAIVED             1677788 non-null  object 
 11  INSPECTION CATEGORY           1677788 non-null  object 
 12  DEPARTMENT BUREAU             1677788 non-null  object 
 13  ADDRESS                       1677788 non-null  object 
 14  STREET NUMBER                 1677788 non-null  int64  
 15  STREET DIRECTION              1677788 non-null  object 
 16  STREET NAME                   1677788 non-null  object 
 17  STREET TYPE                   1664247 non-null  object 
 18  PROPERTY GROUP                1677788 non-null  int64  
 19  SSA                           321521 non-null   float64
 20  LATITUDE                      1676278 non-null  float64
 21  LONGITUDE                     1676278 non-null  float64
 22  LOCATION                      1676278 non-null  object 
 23  Community Areas               1675509 non-null  float64
 24  Zip Codes                     1676278 non-null  float64
 25  Boundaries - ZIP Codes        1675509 non-null  float64
 26  Census Tracts                 1676243 non-null  float64
 27  Wards                         1675509 non-null  float64
 28  Historical Wards 2003-2015    1675509 non-null  float64
dtypes: float64(9), int64(3), object(17)
memory usage: 371.2+ MB
```

3. 数据分析要求

3.1. 数据可视化与摘要

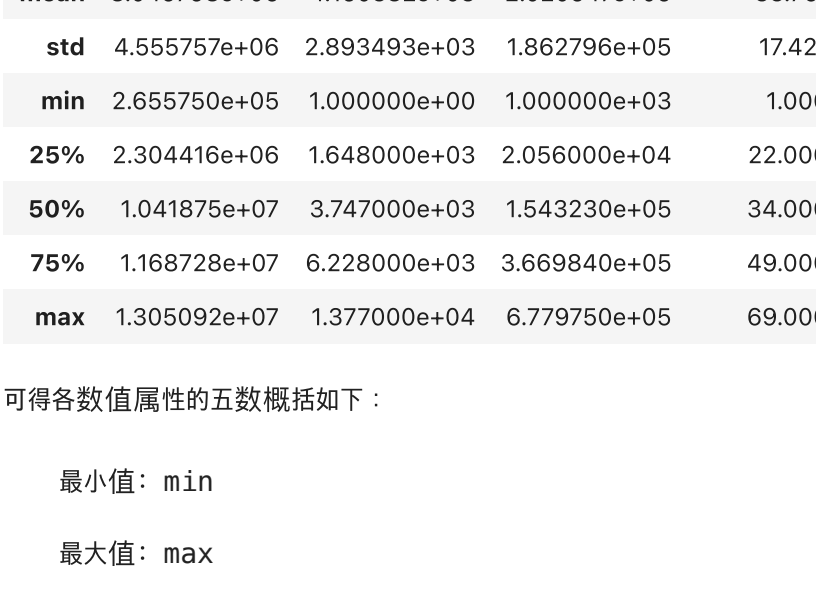
3.1.1. 数据摘要

标称属性，给出每个可能取值的频数。

In [7]:
```python
#由于功能重复，仅显示一个标称属性'country'的频数统计和可视化，可通过更改i的值显示其他属性
i=3
print(data.columns[i])
print(getattr(data, data.columns[i]).value_counts())
data[data.columns[i]].value_counts().head(10).plot.barh()
```

```
VIOLATION STATUS
OPEN         1030958
COMPLIED      641247
NO ENTRY        5583
Name: VIOLATION STATUS, dtype: int64
<AxesSubplot:>
```

Out[7]:



数值属性，给出5数概括及缺失值的个数。

In [8]:
```python
data.describe()
```

Out[8]:

	INSPECTION NUMBER	STREET NUMBER	PROPERTY GROUP	SSA	LATITUDE	LONGITUDE	Community Areas	Zip Codes	Boun ZII
count	1.677788e+06	1.677788e+06	1.677788e+06	321521.000000	1.676278e+06	1.676278e+06	1.675509e+06	1.676278e+06	1.6755
mean	8.049798e+06	4.150382e+03	2.020547e+05	33.769197	4.184566e+01	-8.767266e+01	3.873350e+01	1.933197e+04	3.118
std	4.555757e+06	2.893493e+03	1.862796e+05	17.428210	8.742421e-02	5.760184e-02	2.008963e+01	5.606228e+03	1.9439
min	2.655750e+05	1.000000e+00	1.000000e+03	1.000000	4.164467e+01	-8.791444e+01	1.000000e+00	4.299000e+03	1.0000
25%	2.304416e+06	1.648000e+03	2.056000e+04	22.000000	4.177090e+01	-8.771392e+01	2.400000e+01	2.119000e+04	1.5000
50%	1.041875e+07	3.747000e+03	1.543230e+05	34.000000	4.185400e+01	-8.766985e+01	3.600000e+01	2.156900e+04	2.8000
75%	1.168728e+07	6.228000e+03	3.669840e+05	49.000000	4.191350e+01	-8.763288e+01	5.800000e+01	2.224800e+04	5.2000
max	1.305092e+07	1.377000e+04	6.779750e+05	69.000000	4.202269e+01	-8.752468e+01	7.700000e+01	2.262000e+04	6.1000

可得各数值属性的五数概括如下：

最小值：min

最大值：max

四分位数（Q1）：25%

中位数：50%

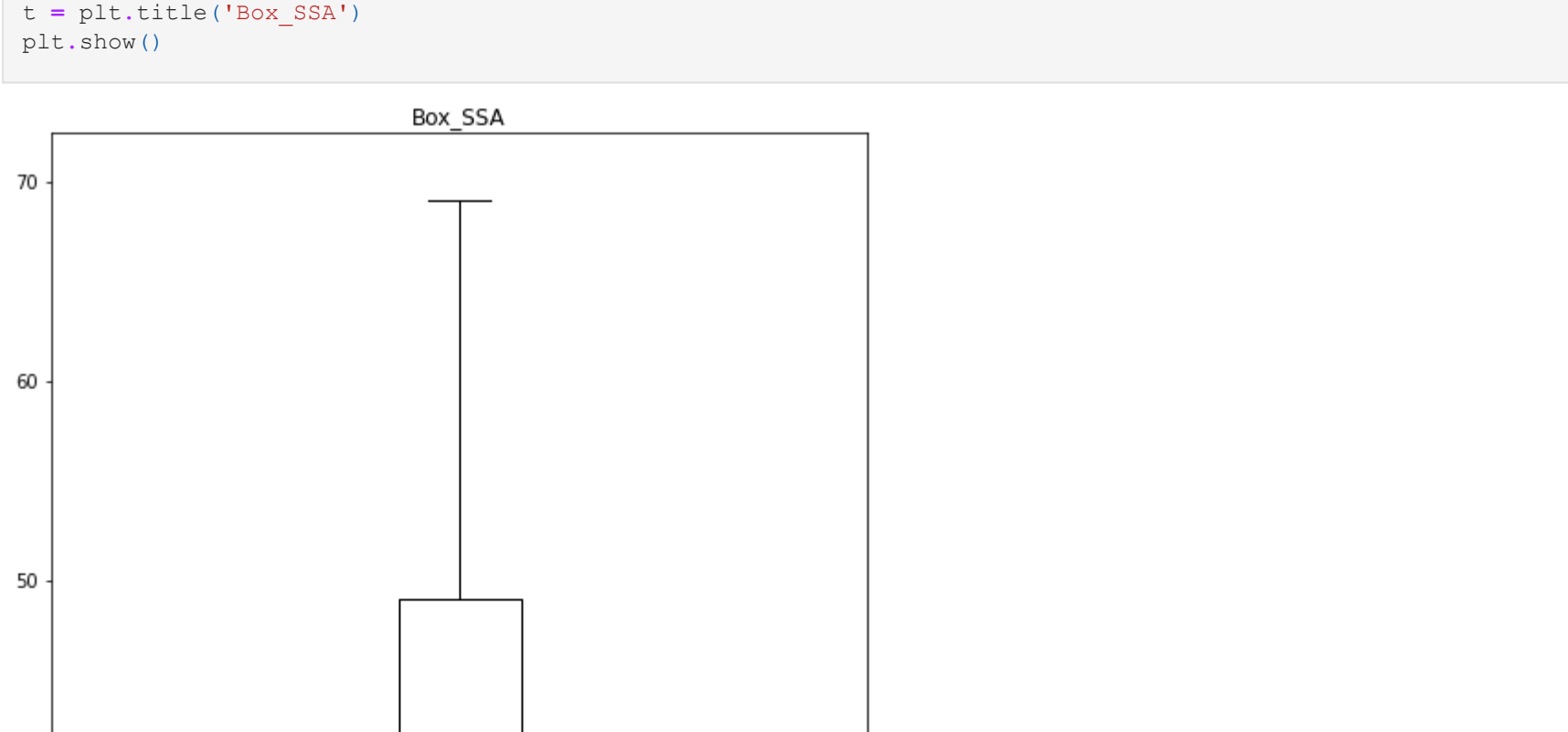
四分位数（Q3）：75%

In [9]:
```python
#缺失值个数统计(由于数据过多，仅以LONGITUDE为例)
print('LONGITUDE:',data['LONGITUDE'].isnull().sum())
```

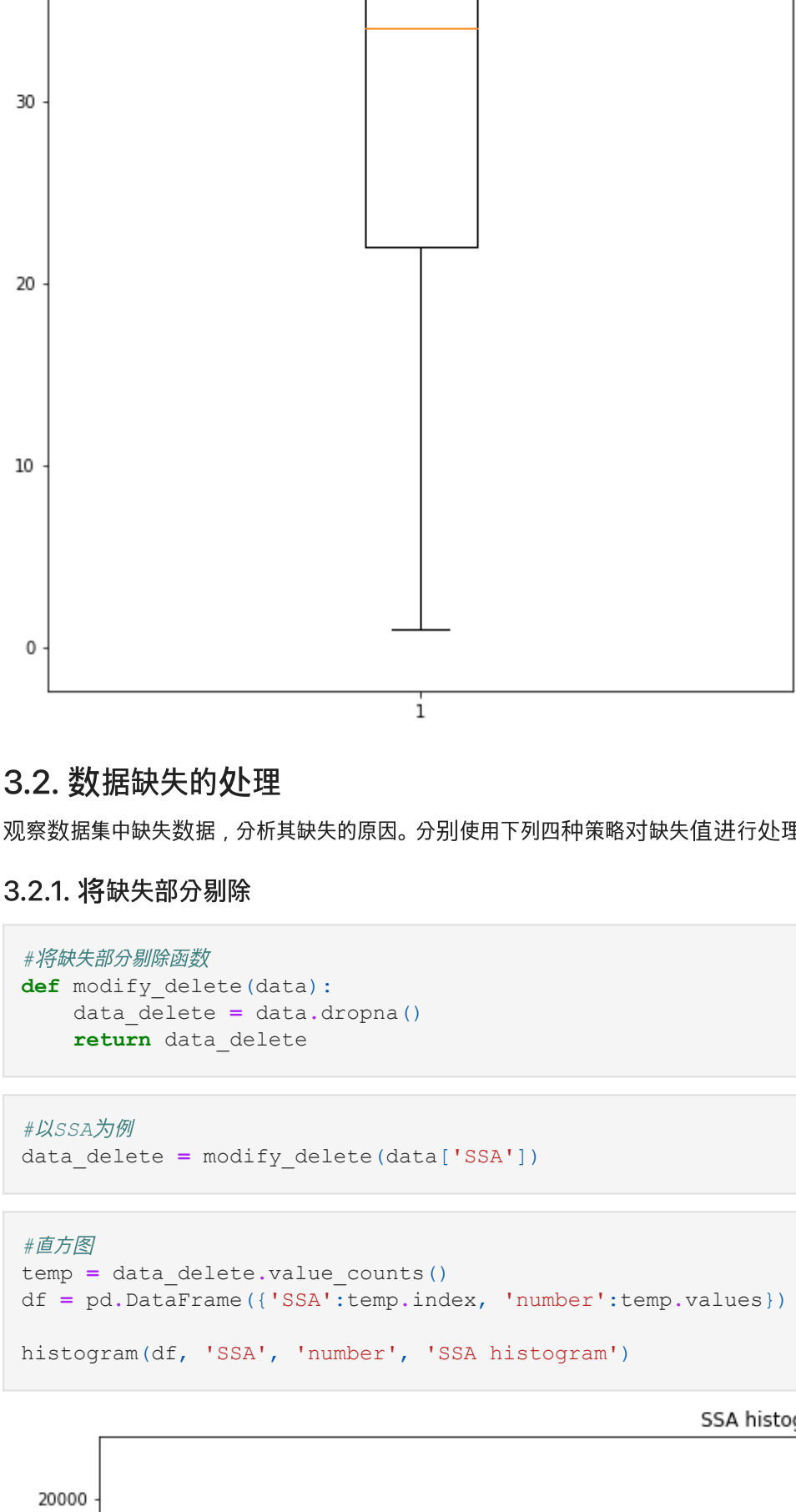
```
LONGITUDE: 1510
```

3.1.2. 数据可视化

In [10]:
```python
#直方图(由于数据过多，仅以SSA为例)
import seaborn as sns
import matplotlib.pyplot as plt

def histogram(data, x, y ,title):
    plt.figure(figsize = (15,6))
    plt.title(title)
    sns.set_color_codes("pastel")
    sns.barplot(x=x, y=y, data=df)
    locs, labels = plt.xticks()
    plt.show()

#SSA
temp = data['SSA'].value_counts()
df = pd.DataFrame({'SSA':temp.index, 'number':temp.values})

histogram(df, 'SSA', 'number', 'SSA histogram')
```



In [11]:
```python
fig = plt.figure(figsize=(8, 15))
plt.boxplot(data['SSA'].loc[data['SSA']<300], notch=False, sym='o', vert=True)
t = plt.title('Box_SSA')
plt.show()
```



3.2. 数据缺失的处理

观察数据集中缺失数据，分析其缺失的原因。分别使用下列四种策略对缺失值进行处理。

3.2.1. 将缺失部分剔除

In [12]:
```python
#将缺失部分剔除函数
def modify_delete(data):
    data_delete = data.dropna()
    return data_delete
```

In [13]:
```python
#以SSA为例
data_delete = modify_delete(data['SSA'])
```

In [14]:
```python
#直方图
temp = data_delete.value_counts()
df = pd.DataFrame({'SSA':temp.index, 'number':temp.values})

histogram(df, 'SSA', 'number', 'SSA histogram')
```

3.2.2. 用最高频率值来填补缺失值

In [15]:
```python
#以最高频率值填补缺失值函数
def modify_most(data):
    temp = data.mode()[0]     #求众数
    data_most = data.fillna(temp)
    return data_most
```

In [16]:
```python
#以points为例
data_most = modify_most(data['SSA'])
```

In [17]:
```python
#直方图
temp = data_most.value_counts()
df = pd.DataFrame({'SSA':temp.index, 'number':temp.values})

histogram(df, 'SSA', 'number', 'SSA histogram')
```

3.2.3. 通过属性的相关关系来填补缺失值

使用纬度LATITUDE填补经度LONGITUDE

In [18]:
```python
data_fill = pd.DataFrame(data, columns=['LATITUDE', 'LONGITUDE'])
```

In [19]:
```python
data_fill.head(10)
```

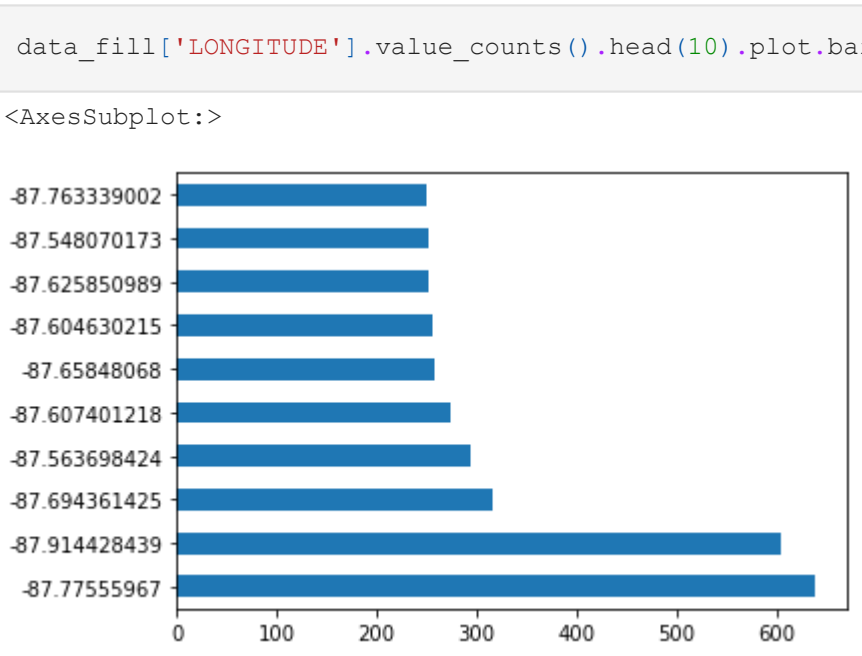
Out[19]:

	LATITUDE	LONGITUDE
0	41.749169	-87.602551
1	41.749169	-87.602551
2	41.711751	-87.537842
3	41.844521	-87.712416
4	41.753908	-87.562784
5	41.806815	-87.611539
6	41.753908	-87.562784
7	41.749169	-87.602551
8	41.711751	-87.537842
9	41.748732	-87.659904

In [20]:
```python
data_fill['LONGITUDE'].value_counts().head(10).plot.barh()
```

Out[20]:

```
<AxesSubplot:>
```

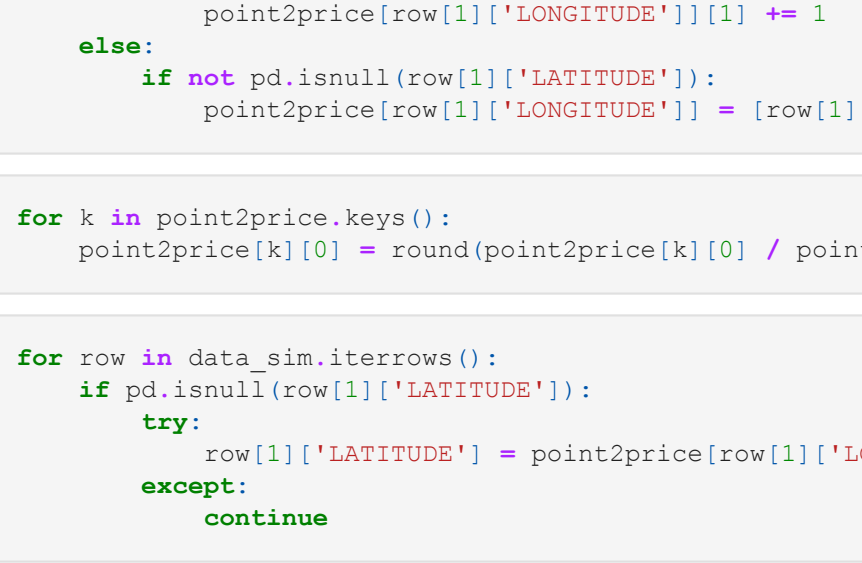


In [22]:
```python
dict = {}
for row in data_fill.iterrows():
    dict[row[1]['LATITUDE']] = row[1]['LONGITUDE']

for row in data_fill.iterrows():
    try:
        region = dict[row[1]['LATITUDE']]
    except:
        continue
    row[1]['LONGITUDE'] = region
```

In [23]:
```python
data_fill['LONGITUDE'].value_counts().head(10).plot.barh()
```

Out[23]:

```
<AxesSubplot:>
```



3.2.4. 通过数据对象之间的相似性来填补缺失值

In [24]:
```python
data_sim = data[['LATITUDE','LONGITUDE']]
```

In [25]:
```python
point2price = {}
for row in data_sim.iterrows():
    if point2price.get(row[1]['LONGITUDE'], None):
        if not pd.isnull(row[1]['LATITUDE']):
            point2price[row[1]['LONGITUDE']][0] += row[1]['LATITUDE']
            point2price[row[1]['LONGITUDE']][1] += 1
    else:
        if not pd.isnull(row[1]['LATITUDE']):
            point2price[row[1]['LONGITUDE']] = [row[1]['LATITUDE'], 1]
```

In [26]:
```python
for k in point2price.keys():
    point2price[k][0] = round(point2price[k][0] / point2price[k][1], 4)
```

In [27]:
```python
for row in data_sim.iterrows():
    if pd.isnull(row[1]['LATITUDE']):
        try:
            row[1]['LATITUDE'] = point2price[row[1]['LONGITUDE']][0]
        except:
            continue
```

In [30]:
```python
#对被填充后的LATITUDE画直方图
temp = data_sim['LATITUDE'].value_counts()
df = pd.DataFrame({'LATITUDE':temp.index, 'number':temp.values})

histogram(df, 'LATITUDE', 'number', 'LATITUDE histogram')
```