1.Consumer & Visitor Insights For Neighborhoods

[https://github.com/AdamFocus/kaggleDataAnalysis (https://github.com/AdamFocus/kaggleDataAnalysis)](https://github.com/AdamFocus/kaggleDataAnalysis)

读取数据集

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
dtype = {'census_block_group': 'object', 'date_range_start': 'int', 'date_range_end':'int',
       'raw_visit_count':'float', 'raw_visitor_count':'float', 'visitor_home_cbgs':'object',
       'visitor_work_cbgs':'object', 'distance_from_home':'float', 'related_same_day_brand':'obj
ect',
       'related_same_month_brand':'object', 'top_brands':'object', 'popularity_by_hour':'object'
,
       'popularity_by_day':'object'}
data = pd.read_csv('cbg_patterns.csv', dtype=dtype)
```

数据集的形状

In [3]:

```
data.shape
```

Out[3]:

```
(220735, 13)
```

数据可视化和摘要

标称属性

对于该数据集，标称属性为census_block_group、related_same_day_brand、related_same_month_brand、top_brands、visitor_home_cbgs、visitor_work_cbgs、popularity_by_hour、popularity_by_day

1. census_block_group：记录了Census Block Group所对应的的特有的12位FIPS码

In [4]:

```
cbgVC=data['census_block_group'].value_counts()
cbgVC
```

Out[4]:

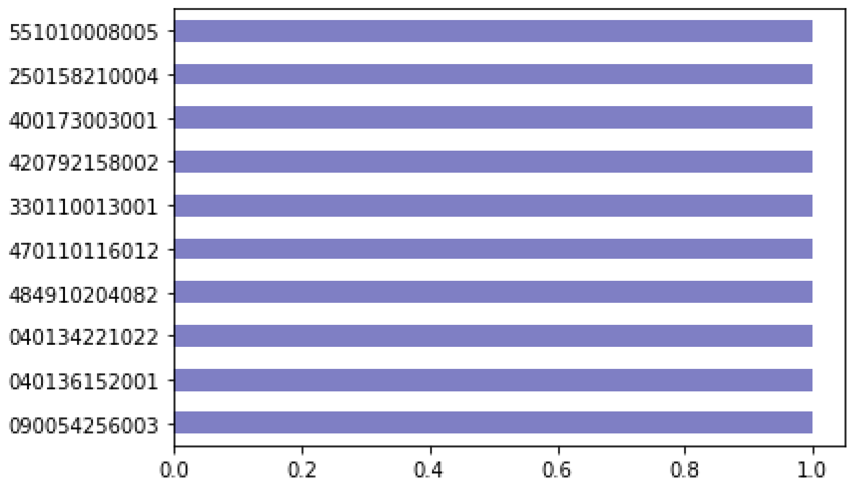
```
090054256003    1
040136152001    1
040134221022    1
484910204082    1
470110116012    1
               ..
060372324001    1
390650003002    1
400173014071    1
131210076021    1
515500209063    1
Name: census_block_group, Length: 220734, dtype: int64
```

In [5]:

```
cbgvcTop10=cbgVC.head(10)
cbgvcTop10.plot(kind='barh', color='darkblue', alpha=0.5)
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x243c2b80c48>
```



2. related_same_day_brand

In [6]:

```
dayVC=data['related_same_day_brand'].value_counts()
dayVC
```

Out[6]:

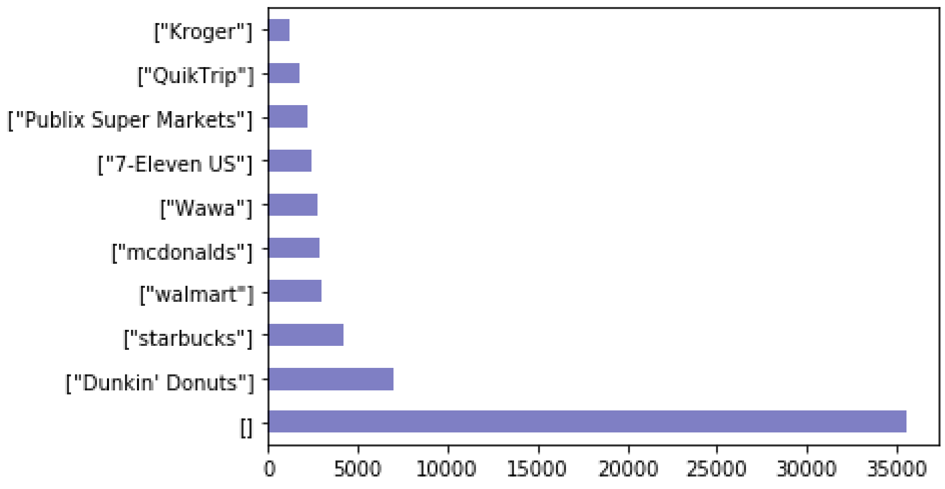
```
[]
35542
["Dunkin' Donuts"]
7010
["starbucks"]
4178
["walmart"]
3054
["mcdonalds"]
2911

...
["Applebee's","walmart","MainStay Suites","United States Postal Service (USPS)","S
UBWAY","Family Dollar Stores","Dunn Bros Coffee","Pita Pit"]        1
["Shell Oil","Pilot Travel Centers","BP","mcdonalds","Exxon Mobil"]
1
["walmart","Sinclair Oil","United States Postal Service (USPS)"]
1
["Kroger","Gatti's Pizza","mcdonalds","Speedway","Texas Roadhouse","Dollar Genera
l"]                                                                    1
["mcdonalds","Pizza King","Speedway","CountryMark","Kroger"]
1
Name: related_same_day_brand, Length: 73198, dtype: int64
```

In [7]:

```
dayVCTop10=dayVC.head(10)
dayVCTop10.plot(kind='barh', color='darkblue', alpha=0.5)
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x243c2969c48>
```



3. related_same_month_brand

In [8]:

```
monthVC=data['related_same_month_brand'].value_counts()
monthVC
```

Out[8]:

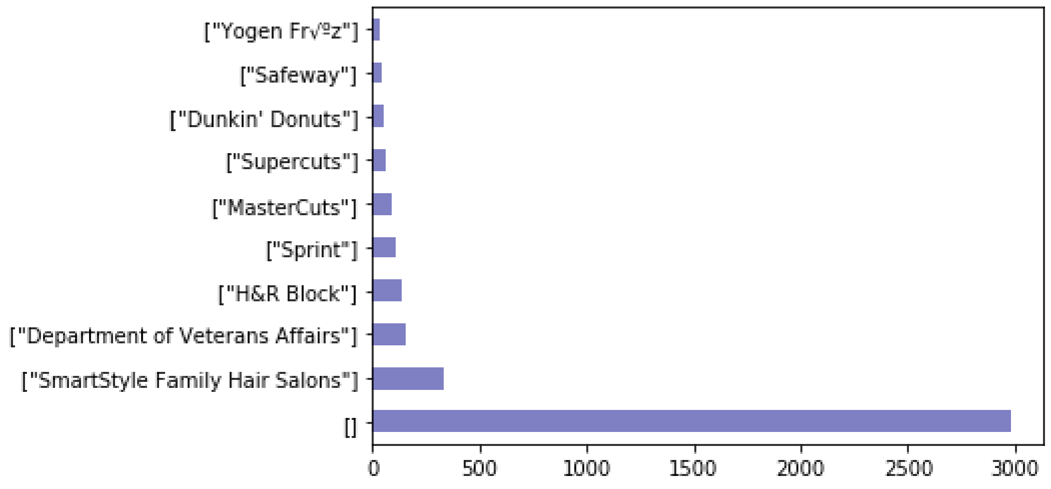
```
[]
2983
["SmartStyle Family Hair Salons"]
335
["Department of Veterans Affairs"]
156
["H&R Block"]
134
["Sprint"]
112

...
["Pilot Travel Centers","SUBWAY","ConocoPhillips","walmart","Love's Travel Stops a
nd Country Stores","mcdonalds","Sinclair Oil","TravelCenters of America","Exxon Mo
bil","Shell Oil"]       1
["Dunkin' Donuts","CVS","Cumberland Farms","mcdonalds","walmart","Target","Stop &
Shop","Dollar Tree","Shell Oil","Price Chopper"]
1
["ConocoPhillips","Kum & Go","Safeway","King Soopers","Loaf 'N Jug","7-Eleven U
S","Kaiser Permanente","starbucks","walmart","Goodwill Industries"]
1
["Casey's General Stores","walmart","Dillons Supermarkets","Sonic","Kwik Shop","mc
donalds","Dollar General","Pizza Hut","SUBWAY","Applebee's"]
1
["walmart","Phillips 66","mcdonalds","Casey's General Stores","Dollar General","Ta
co Bell","Walgreens","SUBWAY","Hy-Vee","BP"]
1
Name: related_same_month_brand, Length: 185558, dtype: int64
```

In [9]:

```
monthVCTop10=monthVC.head(10)
monthVCTop10.plot(kind='barh', color='darkblue', alpha=0.5)
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x243c2b0a908>
```



4. top_brands

In [10]:

```
topbrandVC=data['top_brands'].value_counts()
topbrandVC
```

Out[10]:

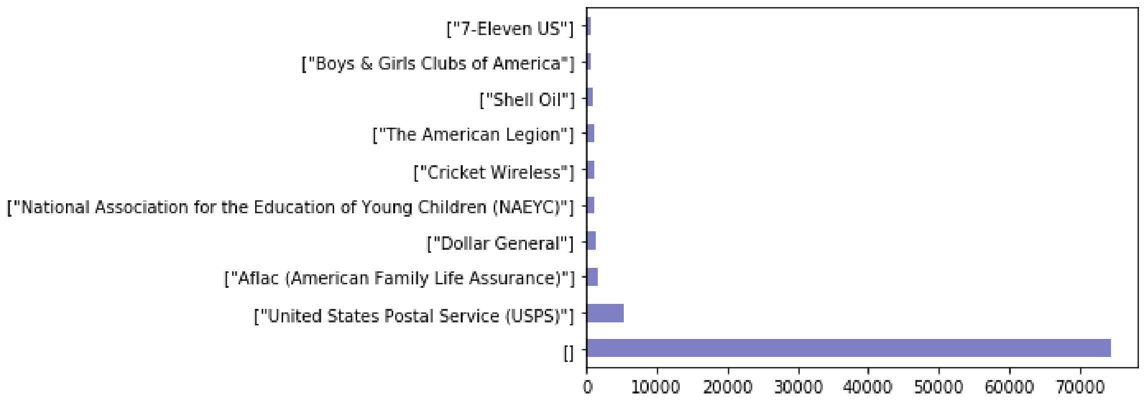
```
[]
74540
["United States Postal Service (USPS)"]
5352
["Aflac (American Family Life Assurance)"]
1517
["Dollar General"]
1308
["National Association for the Education of Young Children (NAEYC)"]
1071

...
["mcdonalds","Walgreens","Micro Center"]
1
["Culver's","Mattress Firm","Mobil","Gordon Food Service (GFS)","Art Van Furnitur
e","Fifth Third Bank","Sprint","Chase","CPR Cell Phone Repair","ATI Physical Thera
py"]        1
["Shell Oil","The Salvation Army","Cricket Wireless","Gravely"]
1
["Hallmark Cards","Shaw's","CVS","Dunkin' Donuts","SUBWAY","Nissan North Americ
a","BMW","Subaru","Group 1 Automotive","O'Reilly Auto Parts"]
1
["Whataburger","Chicken Express","Taco Bell","7-Eleven US","Shell Oil","Childtime
Learning Centers"]
1
Name: top_brands, Length: 98086, dtype: int64
```

In [11]:

```
topbrandVCTop10=topbrandVC.head(10)
topbrandVCTop10.plot(kind='barh', color='darkblue', alpha=0.5)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x243c2bf4d88>
```



5. visitor_home_cbgs

In [12]:

```
homeVC=data['visitor_home_cbgs'].dropna().value_counts()
homeVC
```

Out[12]:

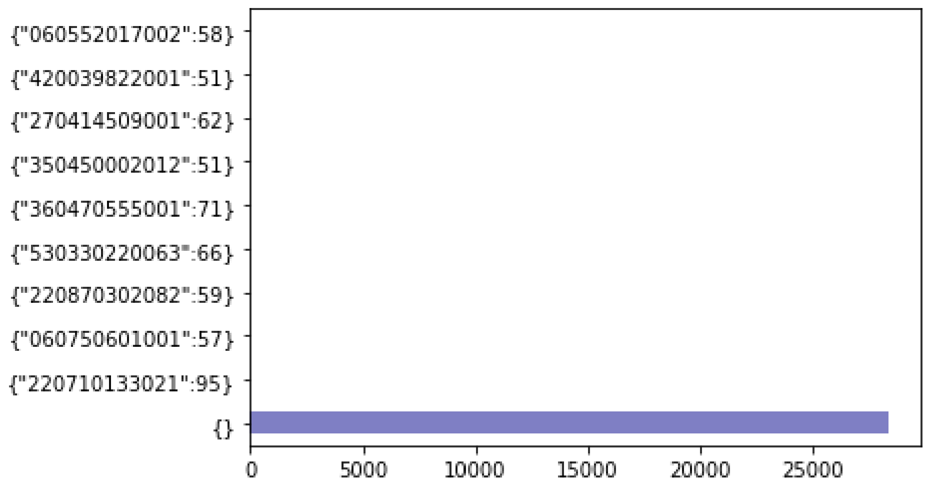
```
{}
28412
{"220710133021":95}
4
{"060750601001":57}
4
{"220870302082":59}
4
{"530330220063":66}
3

...
{"160010101001":149,"160010102012":117,"160010102211":115,"160010003042":76,"16001
0103311":75,"160010102241":74,"160010103321":66,"160010007021":66,"160010102231":6
3,"160010102232":62,"160010102251":58,"160010102013":58,"160010103351":56}
1
{"120570138041":143,"120570139161":95,"120570138071":93,"120570135012":83,"1205701
20021":83,"120570137031":82,"120570133151":74,"120570137023":74,"120570139151":7
3,"120570139082":70,"120570037001":69,"120570141221":68,"120570140071":62,"1205701
35031":59,"120570135011":58,"120570141211":58,"120570120023":58,"120570121031":5
6,"120570036002":55,"120570036003":53,"120570137041":52,"120570036004":52,"1205701
41091":51}         1
{"511539011004":63}
1
{"171279701004":71}
1
{"060750610001":65}
1
Name: visitor_home_cbgs, Length: 191832, dtype: int64
```

In [13]:

```
homeVCTop10=homeVC.head(10)
homeVCTop10.plot(kind='barh', color='darkblue', alpha=0.5)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x243c49853c8>
```



6. visitor_work_cbgs

In [14]:

```
workVC=data['visitor_work_cbgs'].value_counts()
workVC
```

Out[14]:

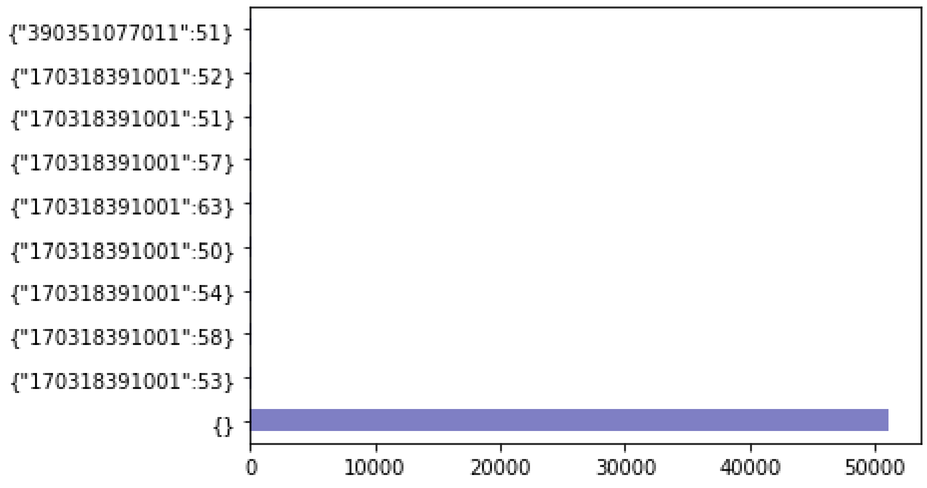
```
{}
51152
{"170318391001":53}
15
{"170318391001":58}
14
{"170318391001":54}
14
{"170318391001":50}
13

...
{"360595185012":119,"360595177051":108,"360599811001":65}
1
{"120990050002":132,"120990070021":75,"120990045003":71,"120990047044":67,"1209900
37002":59,"120990043003":54}
1
{"320310004005":97}
1
{"483290101141":251,"481350004001":236,"481350024005":221,"483290101143":212,"4832
90102002":179,"481350031001":173,"483290002004":150,"481350013002":122,"4813500130
03":116,"481350023003":116,"483290101142":114,"483290005001":104,"481350030004":9
1,"481350030003":91,"483290003021":87,"483290101091":87,"481350016001":87,"4832901
01124":75,"480039502003":74,"481350015003":72,"483290101121":67,"483290101134":6
6,"483290101122":66,"483290101131":61,"483290003033":59,"483290014002":58,"4813500
06006":56,"481350008004":55,"481350025011":54,"481350020003":51,"481350018001":5
0,"481350020001":50,"483899501001":50}         1
{"180390008025":121,"180390003021":62}
1
Name: visitor_work_cbgs, Length: 166013, dtype: int64
```

In [15]:

```
workVCTop10=workVC.head(10)
workVCTop10.plot(kind='barh', color='darkblue', alpha=0.5)
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x243c4ca8e08>
```



7. popularity_by_hour

In [16]:

```
hourpopVC=data['popularity_by_hour'].value_counts()
hourpopVC
```

Out[16]:

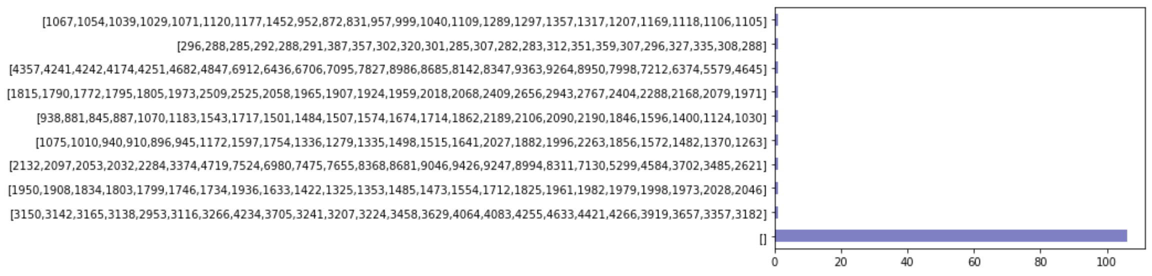
```
[]
106
[3150,3142,3165,3138,2953,3116,3266,4234,3705,3241,3207,3224,3458,3629,4064,4083,4
255,4633,4421,4266,3919,3657,3357,3182]          1
[1950,1908,1834,1803,1799,1746,1734,1936,1633,1422,1325,1353,1485,1473,1554,1712,1
825,1961,1982,1979,1998,1973,2028,2046]          1
[2132,2097,2053,2032,2284,3374,4719,7524,6980,7475,7655,8368,8681,9046,9426,9247,8
994,8311,7130,5299,4584,3702,3485,2621]          1
[1075,1010,940,910,896,945,1172,1597,1754,1336,1279,1335,1498,1515,1641,2027,1882,
1996,2263,1856,1572,1482,1370,1263]              1

...
[1228,1175,1140,1228,1168,1162,1373,1679,1861,1382,1493,1570,1588,1557,1590,2176,2
027,1810,1688,1515,1507,1492,1431,1353]          1
[2833,2693,2624,2594,2616,2792,3480,10960,9453,7545,7529,7643,7972,7873,9173,8696,
7508,7682,7350,6312,5202,4300,3543,3126]         1
[2176,2019,1992,2003,1997,2253,3329,6815,5724,4892,4771,5259,5517,5502,6380,7069,6
799,6257,4959,4361,3938,3401,2761,2382]          1
[925,894,897,888,915,989,1564,3377,2622,2390,2284,2492,2336,2695,2740,2393,2382,22
78,2214,1781,1507,1188,1158,1007]                1
[1231,1190,1177,1176,1196,1166,1378,1805,1312,1089,1078,1100,1198,1217,1274,1696,1
762,2134,1957,1661,1602,1496,1321,1261]          1
Name: popularity_by_hour, Length: 220630, dtype: int64
```

In [17]:

```
hourpopVCTop10=hourpopVC.head(10)
hourpopVCTop10.plot(kind='barh', color='darkblue', alpha=0.5)
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x243c4fc0c08>
```



8. popularity_by_day

In [18]:

```
daypopVC=data['popularity_by_day'].value_counts()
daypopVC
```

Out[18]:

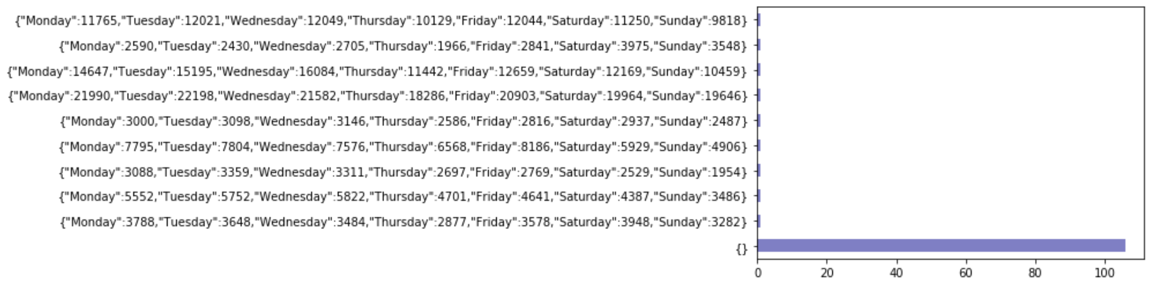
```
{}
106
{"Monday":3788,"Tuesday":3648,"Wednesday":3484,"Thursday":2877,"Friday":3578,"Satu
rday":3948,"Sunday":3282}          1
{"Monday":5552,"Tuesday":5752,"Wednesday":5822,"Thursday":4701,"Friday":4641,"Satu
rday":4387,"Sunday":3486}          1
{"Monday":3088,"Tuesday":3359,"Wednesday":3311,"Thursday":2697,"Friday":2769,"Satu
rday":2529,"Sunday":1954}          1
{"Monday":7795,"Tuesday":7804,"Wednesday":7576,"Thursday":6568,"Friday":8186,"Satu
rday":5929,"Sunday":4906}          1

...
{"Monday":4779,"Tuesday":4674,"Wednesday":4819,"Thursday":3964,"Friday":4079,"Satu
rday":3754,"Sunday":3594}          1
{"Monday":550,"Tuesday":612,"Wednesday":506,"Thursday":479,"Friday":486,"Saturda
y":510,"Sunday":529}                 1
{"Monday":5956,"Tuesday":5983,"Wednesday":5856,"Thursday":4841,"Friday":5617,"Satu
rday":5255,"Sunday":4616}          1
{"Monday":11600,"Tuesday":11897,"Wednesday":12205,"Thursday":9498,"Friday":9534,"S
aturday":6939,"Sunday":5814}       1
{"Monday":9803,"Tuesday":10034,"Wednesday":9814,"Thursday":8469,"Friday":8975,"Sat
urday":8000,"Sunday":6198}         1
Name: popularity_by_day, Length: 220630, dtype: int64
```

In [19]:

```
daypopVCTop10=daypopVC.head(10)
daypopVCTop10.plot(kind='barh', color='darkblue', alpha=0.5)
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x243c5041908>
```



数值属性

其余属性为数值属性，分别是data_range_start、data_range_end、raw_visit_count、raw_visitor_count、distance_from_home。 其中data_range_start、data_range_end为固定值

In [20]:

```
def fiveNumber(name):
    #五数概括 Minimum（最小值）、Q1、Median（中位数、）、Q3、Maximum（最大值）
    Minimum=data[name].min()
    Maximum=data[name].max()
    Q1 = data[name].describe()['25%']
    Q3 = data[name].describe()['75%']
    Median=data[name].describe()['50%']
    IQR = Q3-Q1
    lower_limit=Q1-1.5*IQR #下限值
    upper_limit=Q3+1.5*IQR #上限值
    return [Minimum,Q1,Median,Q3,Maximum,lower_limit,upper_limit]
```

In [21]:

```
print('raw_visit_count五数概括:',fiveNumber('raw_visit_count'))
print('raw_visitor_count五数概括:',fiveNumber('raw_visitor_count'))
print('distance_from_home五数概括:',fiveNumber('distance_from_home'))
```

```
raw_visit_count五数概括: [60.0, 17042.0, 30640.0, 56678.0, 7179900.0, -42412.0, 11
6132.0]
raw_visitor_count五数概括: [50.0, 3430.0, 6541.0, 13099.0, 6113949.0, -11073.5, 27
602.5]
distance_from_home五数概括: [706.0, 8584.0, 14614.0, 31397.75, 6297845.0, -25636.6
25, 65618.375]
```

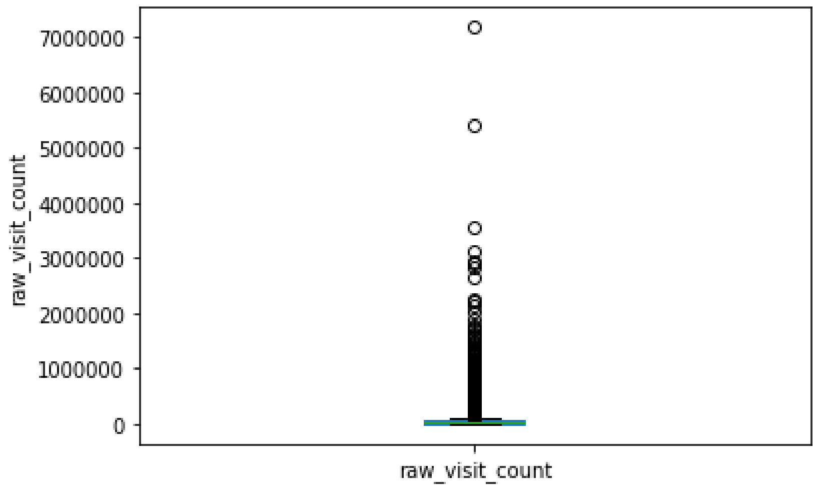
绘制盒图

In [22]:

```
def boxplot(name):
    fig,axes=plt.subplots()
    data[name].plot(kind='box',ax=axes)
    axes.set_ylabel(name)
```

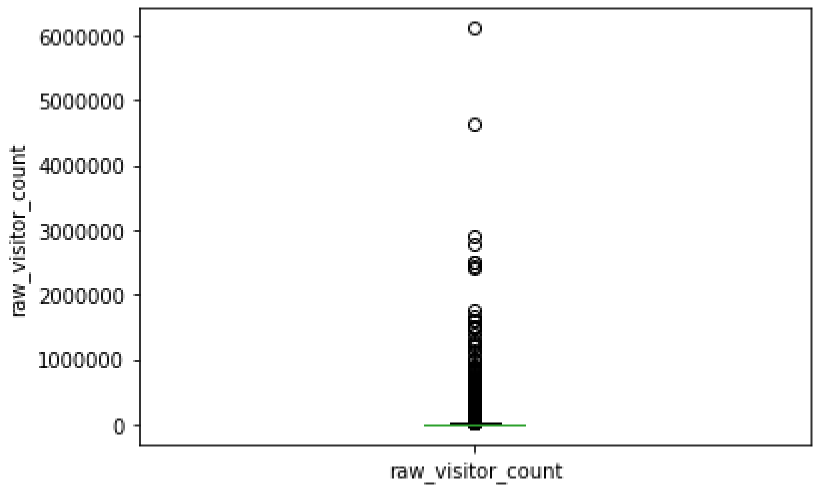
In [23]:

```
boxplot('raw_visit_count')
```



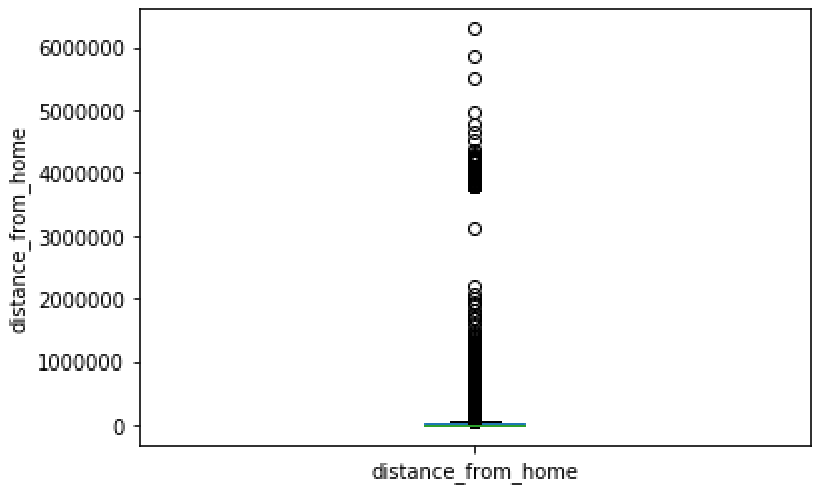
In [24]:

```
boxplot('raw_visitor_count')
```



In [25]:

```
boxplot('distance_from_home')
```



绘制去除离群点前和去除后的直方图

In [26]:

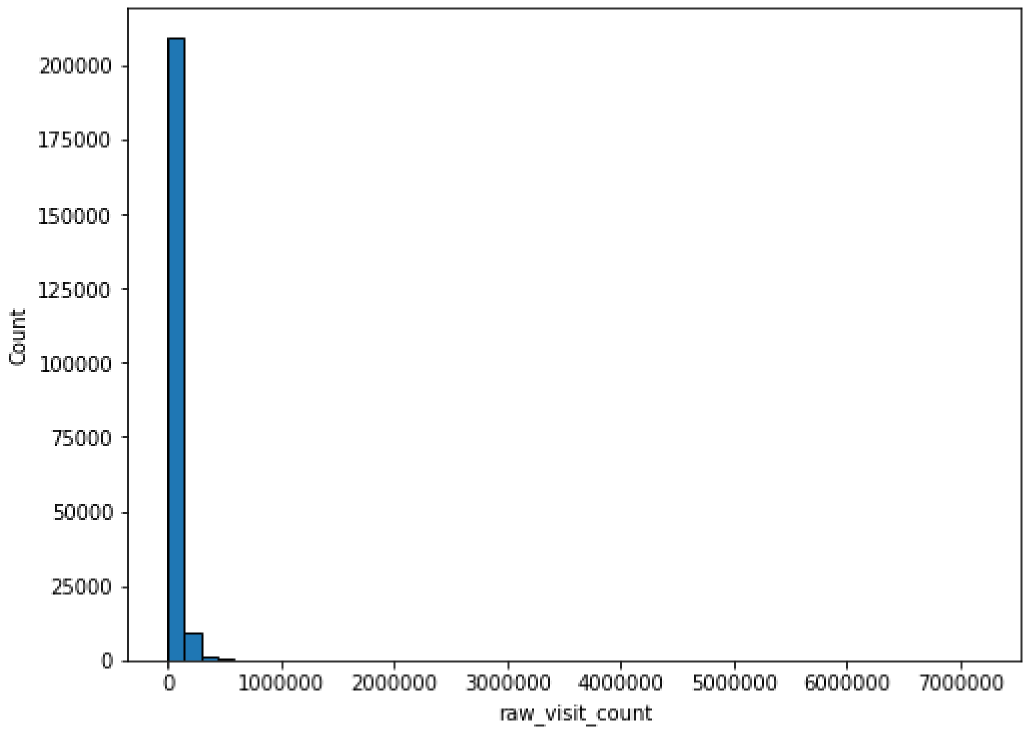
```
def Beforehist(name):
    plt.figure(figsize = (8, 6))
    plt.hist(data[name].dropna(), bins = 50, edgecolor = 'black')
    plt.xlabel(name)
    plt.ylabel('Count')
```

In [27]:

```
def Afterhist(name):
    x=data.copy(deep=True)
    Q1 = x[name].describe()['25%']
    Q3 = x[name].describe()['75%']
    iqr = Q3-Q1
    x = x[(x[name] > (Q1 - 3 * iqr)) & (x[name] < (Q3 + 3 * iqr))]
    plt.figure(figsize = (8, 6))
    plt.hist(x[name].dropna(), bins = 50, edgecolor = 'black')
    plt.xlabel(name)
    plt.ylabel('Count')
```

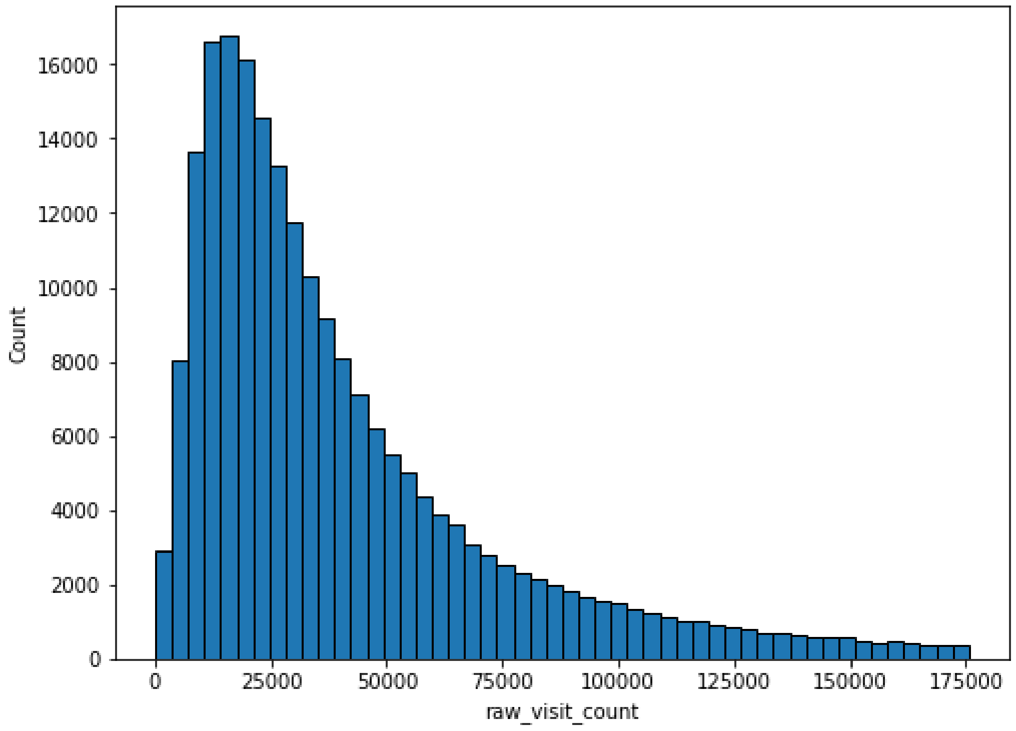
In [28]:

```
Beforehist('raw_visit_count')
```



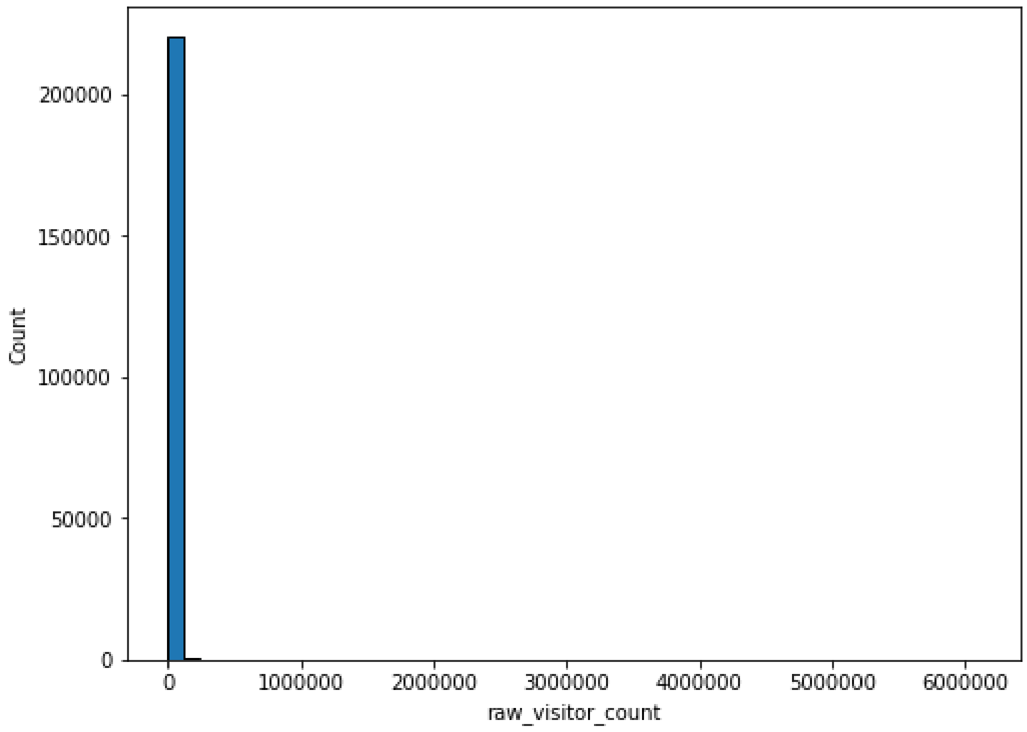
In [29]:

```
Afterhist('raw_visit_count')
```



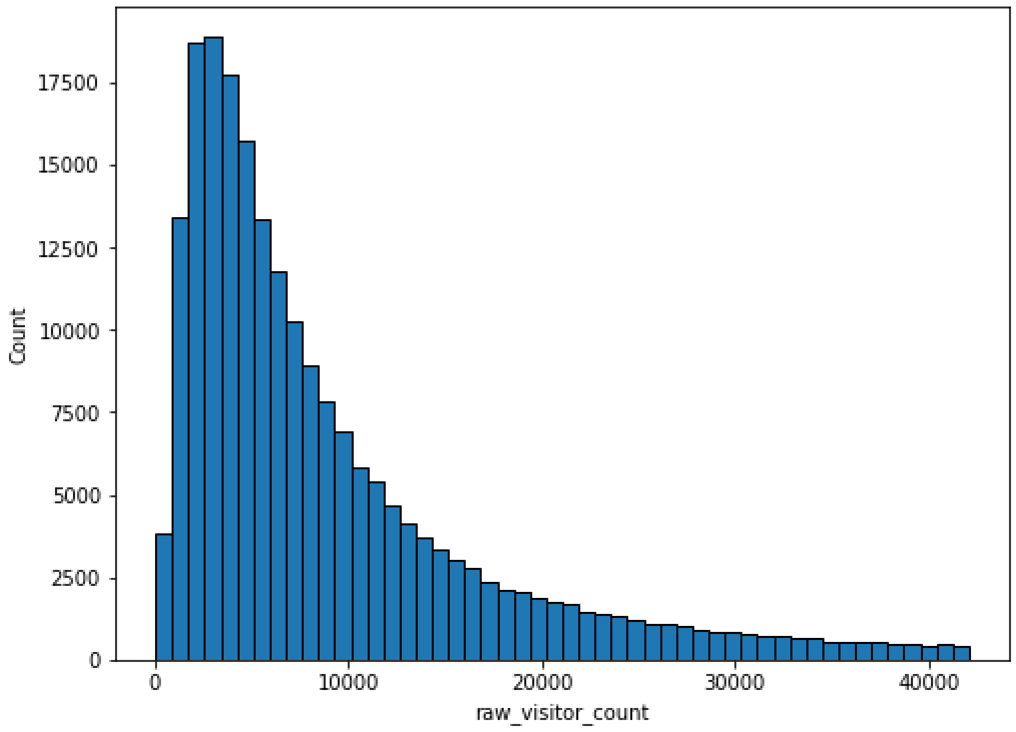
In [30]:

```
Beforehist('raw_visitor_count')
```



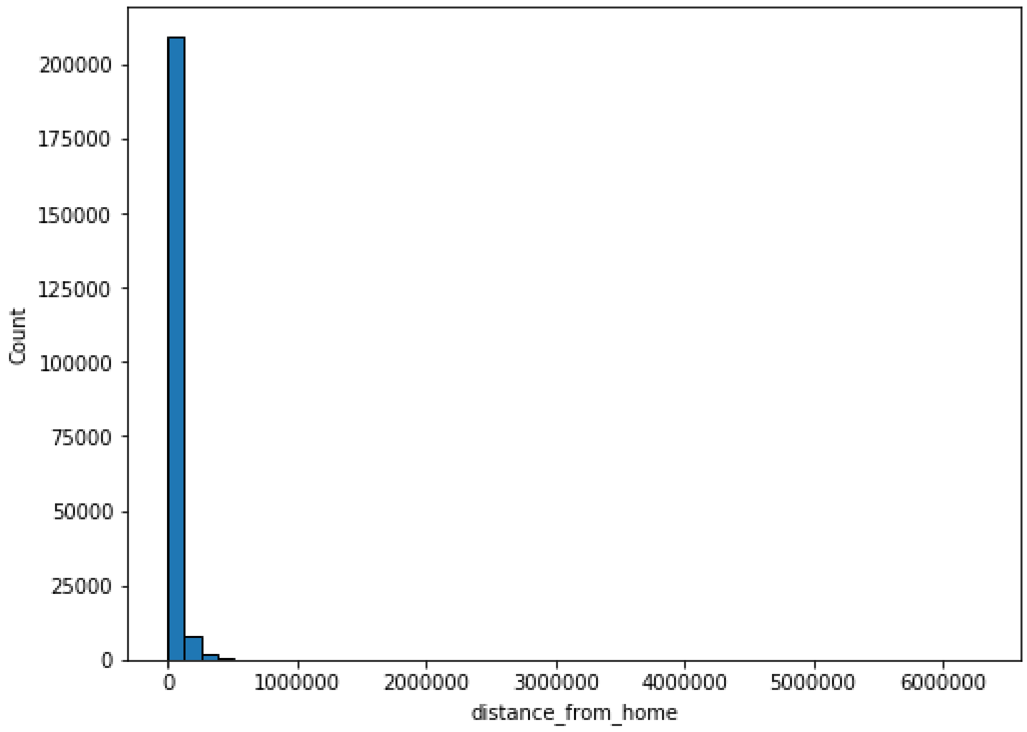
In [31]:

```
Afterhist('raw_visitor_count')
```



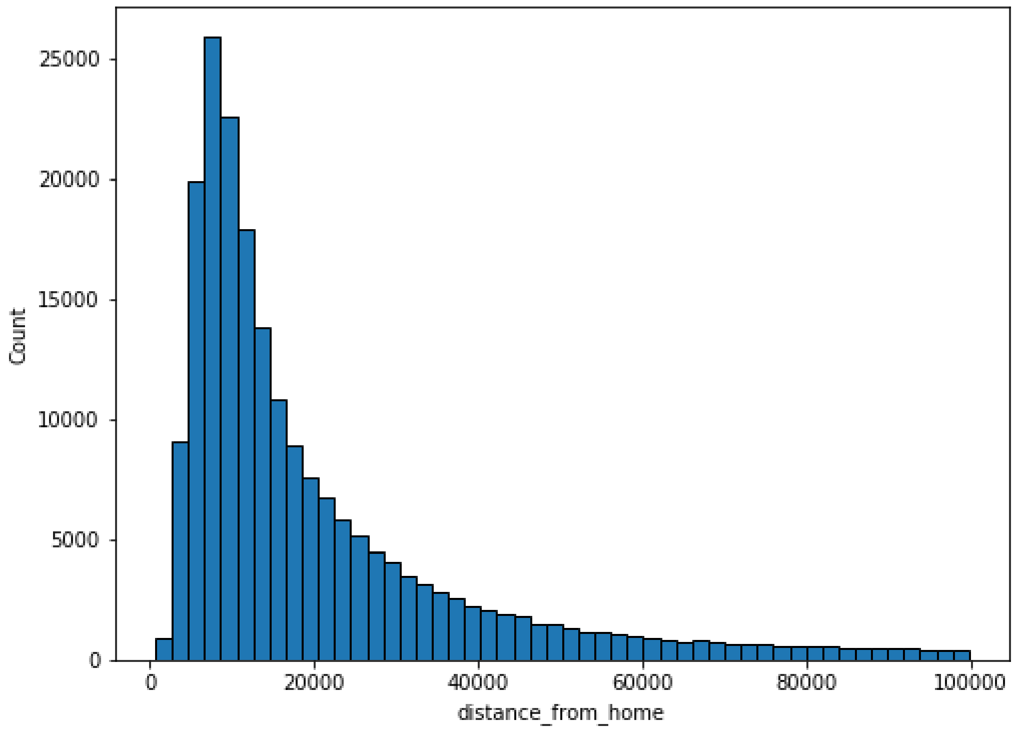
In [32]:

```
Beforehist('distance_from_home')
```



In [33]:

```
Afterhist('distance_from_home')
```



数据缺失的处理

各个标称属性的缺失情况：

- census_block_group：仅缺失一项
- related_same_day_brand：缺失35542项
- related_same_month_brand：缺失2569项
- top_brands：缺失74540项
- visitor_home_cbgs：缺失28303项
- visitor_work_cbgs：缺失51152项
- popularity_by_hour：缺失106项
- popularity_by_day：缺失106项

其他属性的缺失情况

In [34]:

```
VisitMissnum = data['raw_visit_count'].isna().sum()
VisitorMissnum=data['raw_visitor_count'].isna().sum()
Time1=data['date_range_start'].isna().sum()
Time2=data['date_range_end'].isna().sum()
Distancenum=data['distance_from_home'].isna().sum()
```

In [35]:

```
print("raw_visit_count缺失：",VisitMissnum,"项")
print("raw_visitor_count缺失：",VisitorMissnum,"项")
print("date_range_start缺失：",Time1,"项")
print("date_range_end缺失：",Time2,"项")
print("distance_from_home缺失：",Distancenum,"项")
```

```
raw_visit_count缺失： 106 项
raw_visitor_count缺失： 106 项
date_range_start缺失： 0 项
date_range_end缺失： 0 项
distance_from_home缺失： 217 项
```

1.将缺失部分剔除

通过观察数据集，popularity_by_hour、popularity_by_day、raw_visit_count、raw_visitor_count缺失的106项皆为数据集中的空数据，故可以将数据集中最后106行删除

In [36]:

```
newData=data.copy(deep=True)
```

In [37]:

```
newData.drop(index=(newData.loc[(newData['raw_visit_count'].isna())].index),inplace=True)
newData.shape
```

Out[37]:

```
(220629, 13)
```

2.用最高频率值来填补缺失值

查看此时最受欢迎的品牌

In [38]:

```
topBrandsList=[]
i=0
for row in newData['top_brands'].values:
    k=eval(row)
    topBrandsList.extend(k)
#     if i<10:
#         print(row)
#     i=i+1
topBrandsSe=pd.Series(topBrandsList)
topBrandsSeVC=topBrandsSe.value_counts()
topBrandsSeVC
```

Out[38]:

```
United States Postal Service (USPS)    22119
SUBWAY                                 16600
Dollar General                         13515
mcdonalds                              12670
Shell Oil                              10940
                                       ...
Pacific Theatres                           1
Don Pablo's                                1
Gift-ology                                 1
Morgan Jewelers                            1
Site for Sore Eyes                         1
Length: 3126, dtype: int64
```

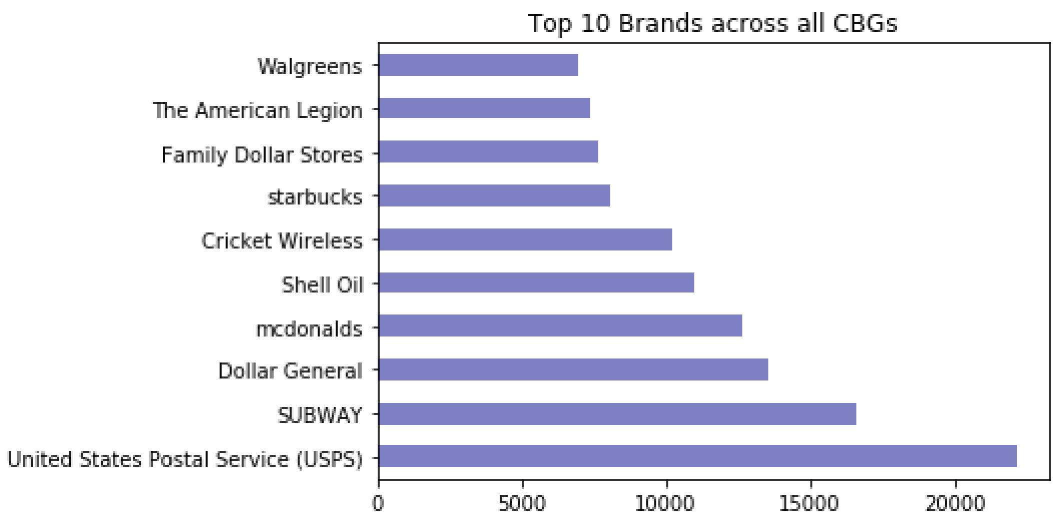
In [39]:

```
most=str([topBrandsSeVC.index[0]])
```

绘图查看此时排名前十的topBrands

In [40]:

```
top10=topBrandsSeVC.head(10)
top10.plot(kind='barh', color='darkblue', alpha=0.5)
plt.title('Top 10 Brands across all CBGs')
plt.show()
```



将所有空的top_brands填充为United States Postal Service (USPS)

In [41]:

```
for index,row in newData.iterrows():
    j=row['top_brands']
    if isinstance(j,str):
        k=eval(j)
        if k==[]:
            newData.at[index,'top_brands']=most
    else:
        print(j)
```

绘图查看替换后排名前十的topBrands

In [42]:

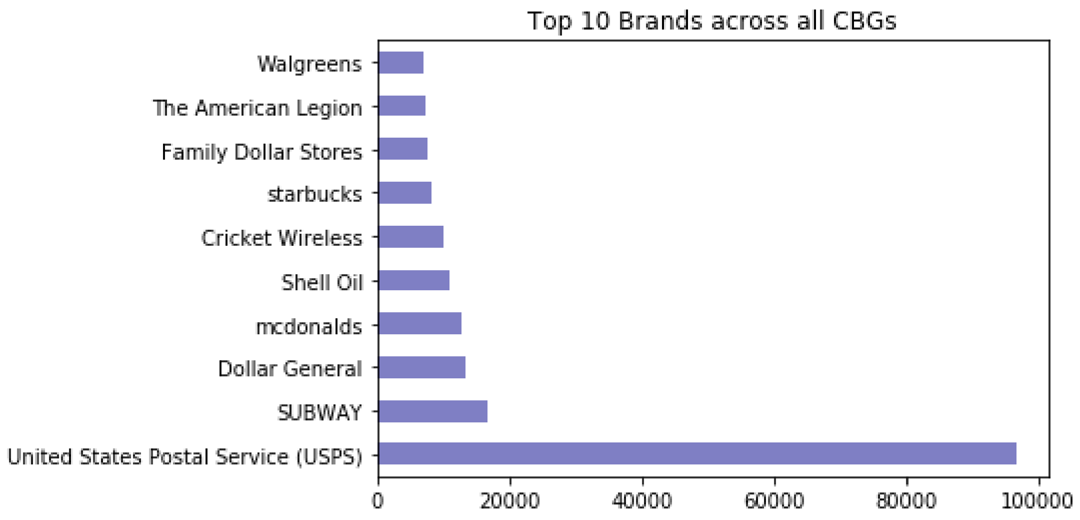
```
newtopBrandsList=[]
i=0
for row in newData['top_brands'].values:
    k=eval(row)
    newtopBrandsList.extend(k)
#     if i<10:
#         print(row)
#     i=i+1
newtopBrandsSe=pd.Series(newtopBrandsList)
newtopBrandsSeVC=newtopBrandsSe.value_counts()
newtopBrandsSeVC
```

Out[42]:

```
United States Postal Service (USPS)    96553
SUBWAY                                 16600
Dollar General                         13515
mcdonalds                              12670
Shell Oil                              10940
                                       ...
GANT                                       1
Maryland Live! Casino                      1
Montblanc                                  1
Tile For Less                              1
Wilson Fuel                                1
Length: 3126, dtype: int64
```

In [43]:

```
top10=newtopBrandsSeVC.head(10)
top10.plot(kind='barh', color='darkblue', alpha=0.5)
plt.title('Top 10 Brands across all CBGs')
plt.show()
```



In [44]:

```
newData['top_brands'].value_counts()
```

Out[44]:

```
['United States Postal Service (USPS)']
74434
["United States Postal Service (USPS)"]
5352
["Aflac (American Family Life Assurance)"]
1517
["Dollar General"]
1308
["National Association for the Education of Young Children (NAEYC)"]
1071

...
["Circle K Stores","O'Reilly Auto Parts","Extra Space Storage"]
1
["mcdonalds","Walgreens","Micro Center"]
1
["Culver's","Mattress Firm","Mobil","Gordon Food Service (GFS)","Art Van Furnitur
e","Fifth Third Bank","Sprint","Chase","CPR Cell Phone Repair","ATI Physical Thera
py"]        1
["Shell Oil","The Salvation Army","Cricket Wireless","Gravely"]
1
["Whataburger","Chicken Express","Taco Bell","7-Eleven US","Shell Oil","Childtime
Learning Centers"]
1
Name: top_brands, Length: 98086, dtype: int64
```

3.通过属性的相关关系来填补缺失值

可以借助raw_visit_count和raw_visitor_count来推断distance_from_home

In [45]:

```
Distancenum=newData['distance_from_home'].isna().sum()
Distancenum
```

Out[45]:

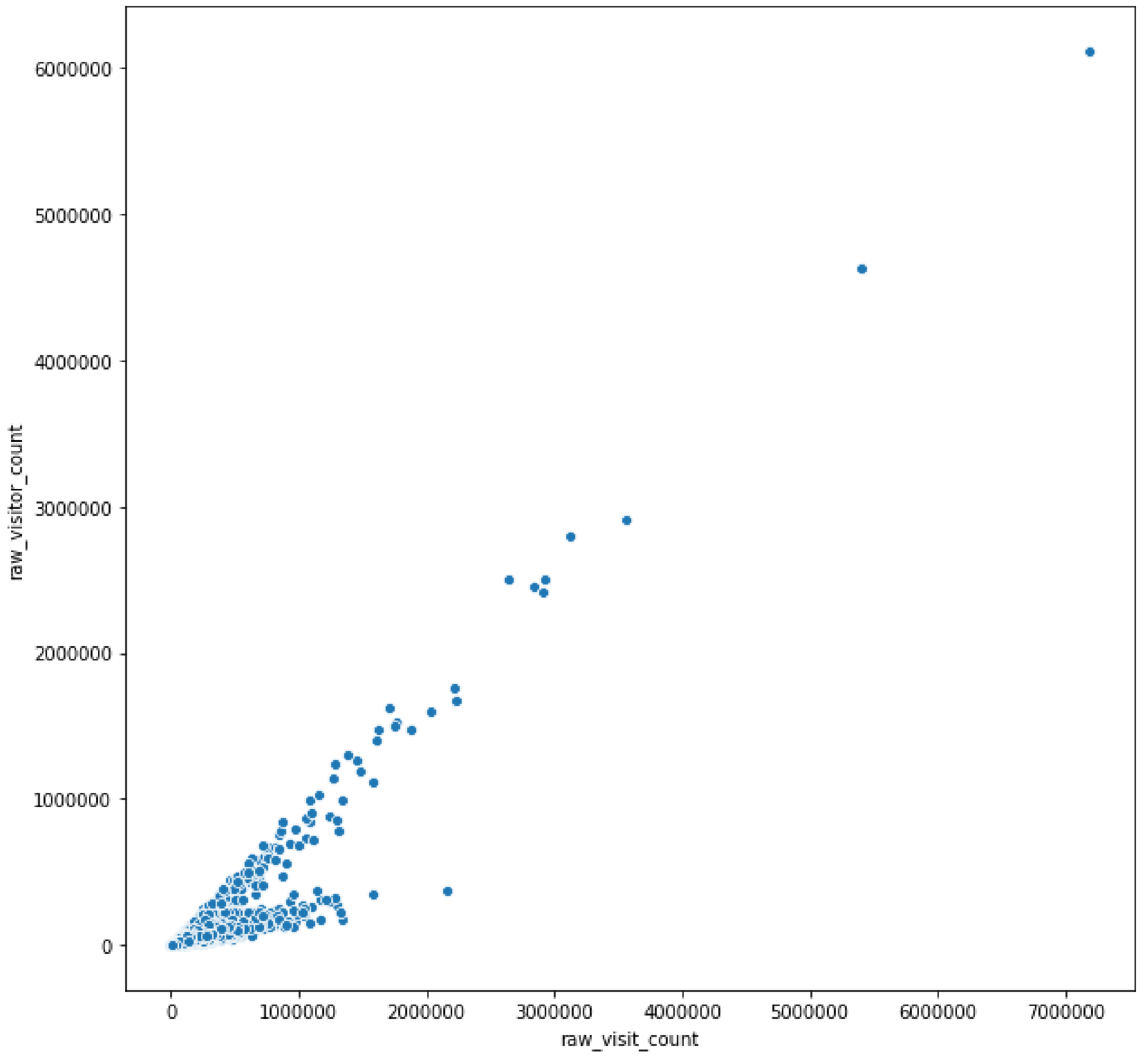
```
111
```

distance_from_home目前缺少111项

绘图查看raw_visit_count和raw_visitor_count的关系，可以借助一个参数percent_unique_visitor近似表示二者之间的线性关系

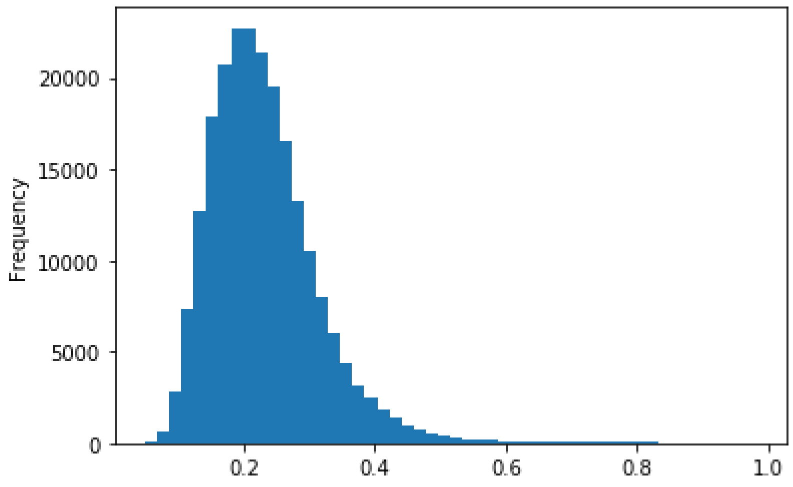
In [46]:

```
plt.figure(figsize=(10,10))
sns.scatterplot(x='raw_visit_count',y='raw_visitor_count', data=newData)
plt.show()
```



In [47]:

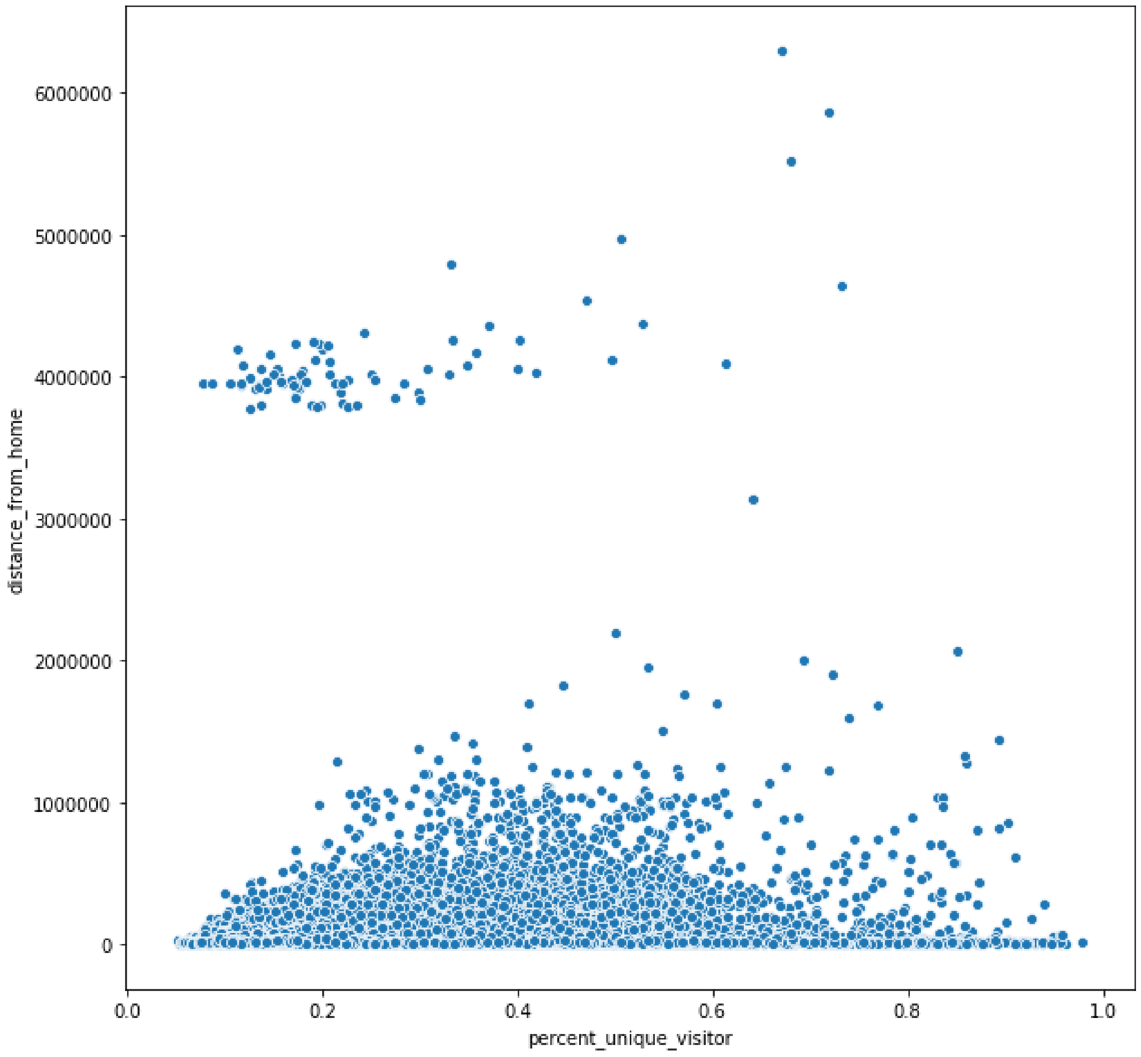
```
newData['percent_unique_visitor'] = newData['raw_visitor_count']/newData['raw_visit_count']
newData['percent_unique_visitor'].plot(kind='hist',bins=50)
plt.show()
```



对于distance缺失的项，我们可以借助raw_visit_count和raw_visitor_count计算percent_unique_visitor，寻找对应的distance进行填充

In [48]:

```
plt.figure(figsize=(10,10))
sns.scatterplot(y='distance_from_home',x='percent_unique_visitor', data=newData)
plt.show()
```



构造出distance与percent_unique_visitor的dataframe，计算出每一个percent_unique_visitor对应的平均distance，并进行填充

In [49]:

```
for index,row in newData.iterrows():
    m=row['distance_from_home']
#     if np.isnan(m):
```

4.通过数据对象之间的相似性来填补缺失值

related_same_day_brand和related_same_month_brand存在一定的相似关系，故可以二者可以互相填充

当related_same_day_brand缺失时，从related_same_month_brand中提取补充，如果related_same_month_brand同时缺失，则填充上面获取的最大可能的brand值。

如果related_same_day_brand没有缺失，查看related_same_month_brand是否缺失，如果缺失则用related_same_day_brand填充

填充前的related_same_month_brand中出现的品牌，对其绘图

In [50]:

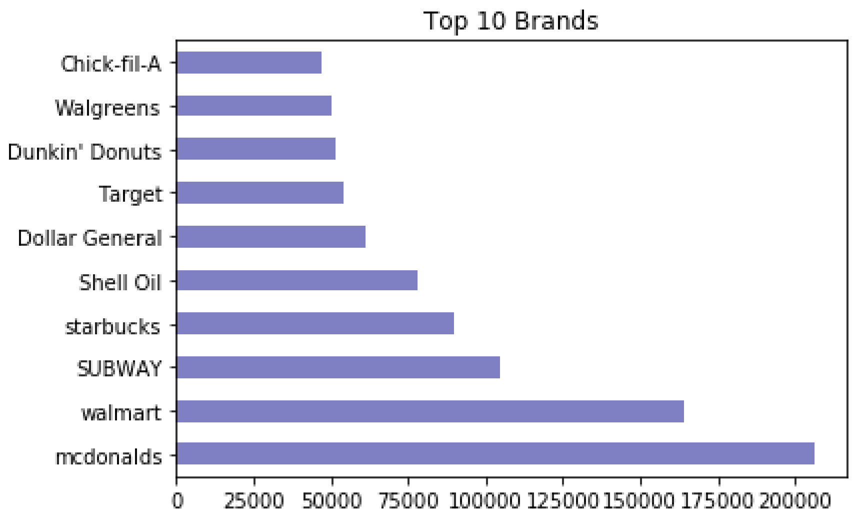
```
monthBrandsList=[]
i=0
for row in newData['related_same_month_brand'].values:
    k=eval(row)
    monthBrandsList.extend(k)
#     if i<10:
#         print(row)
#     i=i+1
monthBrandsSe=pd.Series(monthBrandsList)
monthBrandsSeVC=monthBrandsSe.value_counts()
monthBrandsSeVC
```

Out[50]:

```
mcdonalds                 206438
walmart                   164347
SUBWAY                    104876
starbucks                  89666
Shell Oil                  77980
                           ...
New Mexico DMV                 1
Motel 6                        1
NrGize Lifestyle Cafe          1
Metro Mattress                 1
The Luxury Collection          1
Length: 1017, dtype: int64
```

In [51]:

```
monthtop10=monthBrandsSeVC.head(10)
monthtop10.plot(kind='barh', color='darkblue', alpha=0.5)
plt.title('Top 10 Brands')
plt.show()
```



进行填充

In [52]:

```
for index,row in newData.iterrows():
    j=row['related_same_day_brand']
#     if isinstance(j,str):
    k=eval(j)
    if k==[]:
        m=row['related_same_month_brand']
        n=eval(m)
        if n==[]:
            newData.at[index,'related_same_month_brand']=most
            newData.at[index,'related_same_day_brand']=most
        else:
            newData.at[index,'related_same_day_brand']=m
    else:
        m=row['related_same_month_brand']
        n=eval(m)
        if n==[]:
            newData.at[index,'related_same_month_brand']=j
```

查看填充后的数据

In [53]:

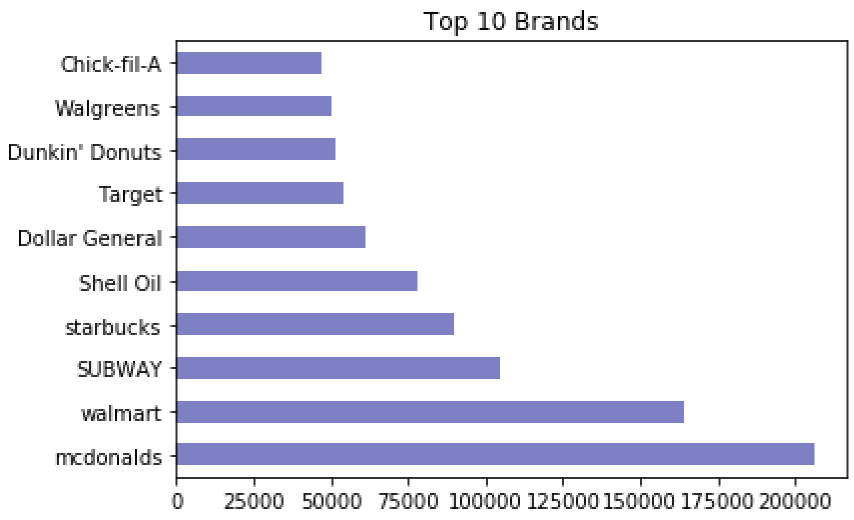
```
newmonthBrandsList=[]
i=0
for row in newData['related_same_month_brand'].values:
    k=eval(row)
    newmonthBrandsList.extend(k)
#     if i<10:
#         print(row)
#     i=i+1
newmonthBrandsSe=pd.Series(newmonthBrandsList)
newmonthBrandsSeVC=newmonthBrandsSe.value_counts()
newmonthBrandsSeVC
```

Out[53]:

```
mcdonalds                            206512
walmart                              164459
SUBWAY                               104963
starbucks                             89723
Shell Oil                             78010
                                      ...
Tijuana Flats                             1
World Gym                                 1
RC Willey Home Furnishings                1
Cinnabon                                  1
American Family Insurance (AmFam)         1
Length: 1111, dtype: int64
```

In [54]:

```
monthtop10=newmonthBrandsSeVC.head(10)
monthtop10.plot(kind='barh', color='darkblue', alpha=0.5)
plt.title('Top 10 Brands')
plt.show()
```



In [55]:

```
newData['related_same_day_brand'].value_counts()
```

Out[55]:

```
["Dunkin' Donuts"]
7044
["starbucks"]
4179
["walmart"]
3054
["mcdonalds"]
2911
["Wawa"]
2834

...
["Pilot Travel Centers","Love's Travel Stops and Country Stores","mcdonalds","SUBW
AY","TravelCenters of America","Phillips 66","Shell Oil","Freightliner Trucks","De
nny's"]        1
["Dunkin' Donuts","ShopRite","starbucks","CVS","walmart","Exxon Mobil","Acme Marke
ts","Costco Wholesale Corp.","T.J. Maxx","mcdonalds"]
1
["starbucks","Safeway","Chevron","Target","Costco Wholesale Corp.","In-N-Out Burge
r","SUBWAY","Shell Oil","ARCO","Jack in the Box"]
1
["Dunkin' Donuts","mcdonalds","BP","starbucks","Baskin Robbins","Burger King U
S","Food Bazaar Supermarket","Popeyes Louisiana Kitchen","SUBWAY","Rite Aid"]
1
["mcdonalds","Dollar General","Food City"]
1
Name: related_same_day_brand, Length: 100972, dtype: int64
```

In [56]:

```
newData['related_same_month_brand'].value_counts()
```

Out[56]:

```
['United States Postal Service (USPS)']
2096
["SmartStyle Family Hair Salons"]
338
["Department of Veterans Affairs"]
159
["H&R Block"]
135
["Sprint"]
113

...
["walmart","Circle K Stores","mcdonalds","Dollar General","Kroger","Chick-fil-
A","BP","Shell Oil","Waffle House","Dollar Tree"]
1
["walmart","Sonic","Dollar General","mcdonalds","Exxon Mobil","SUBWAY","Valero Ene
rgy","Taco Bell","Brookshire's Grocery Company","Shell Oil"]
1
["Pilot Travel Centers","SUBWAY","ConocoPhillips","walmart","Love's Travel Stops a
nd Country Stores","mcdonalds","Sinclair Oil","TravelCenters of America","Exxon Mo
bil","Shell Oil"]       1
["Dunkin' Donuts","CVS","Cumberland Farms","mcdonalds","walmart","Target","Stop &
Shop","Dollar Tree","Shell Oil","Price Chopper"]
1
["walmart","Phillips 66","mcdonalds","Casey's General Stores","Dollar General","Ta
co Bell","Walgreens","SUBWAY","Hy-Vee","BP"]
1
Name: related_same_month_brand, Length: 185994, dtype: int64
```