HH_Census_datasets_DataArts

June 25, 2020

In this notebook, we document and give a high level description of the Organizational, Household and Census data we have collected in our databases. Accessing this data require an userid and a password. The databases are hosted on a SQL server. Connecting to the server through an API using for example, python, would require necessary odbc driver.

1 Database: OrgDB

Import the general libraries first and connect to the SQL server

[1]:
```
import pyodbc
import numpy as np
import pandas as pd
import os,sys
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale=1.5)
%matplotlib inline
import datetime
```

[2]:
```
#- Reading my userid and password from the environment variables
hhuid=os.environ['HHUID']
hhpwd=os.environ['HHPWD']
#- Now we establish a connection so that we will be able to perform a query␣
 ↪using pandas read_sql module
#- in python one can use pyodbc or create an engine using sqlalchemy
driver='/usr/local/lib/libmsodbcsql.17.dylib' #- local odbc drive
server='129.119.63.219'
dbname='OrgDB'
port=1433
cnxnOrg=pyodbc.connect(driver=driver,server=server,database=dbname,uid=hhuid,\
                       pwd=hhpwd,port=port)
```

Check the tables in the database

[3]:
```
cursor = cnxnOrg.cursor()
for row in cursor.tables(tableType='TABLE'):
    if row[1]=='dbo': #- avoiding system tables
        print(row[2])
```

```
HHOrgData
HHOrgStatic
```

These tables have already been cleaned out from raw form and integrated for the static and organization level variables. We will explore each of these tables below.

[4]:
```python
def load_data(cnxn,sqlquery):
    """
    cnxn: pyodbc.Connection object
    sqlquery: sql query string
    returns pandas dataframe from the sqlquery.
    Use only for small databases if running from stanalone node-- to make␣
 ↪efficient
    need distributed architecture for larger databases
    """
    cursor=cnxn.cursor()
    data=pd.read_sql(sqlquery,cursor.connection)
    return data
```

For data description we will limit our queries to a few rows. If one expects to extract the full table, it may be slow with the above function. One may increase the data loading efficiency by some form of parallel processing.

1.0.1 HHOrgStatic

Static information for the Organizations that have reported household transactions

[5]:
```python
#- look at the schema
for row in cursor.columns(table='HHOrgStatic'):
    print(row[3],row[5])
```

```
NCARID float
OrgID bigint
ORGName varchar
ADDRESS varchar
CITY varchar
STATE varchar
ZIP float
ZIP9 varchar
STATENO float
County float
FTRACT float
CensusBlock float
CNTYNM varchar
CBSA float
LATITUDE float
LONGITUDE float
Active bigint
InactiveDate float
```

sec_no float

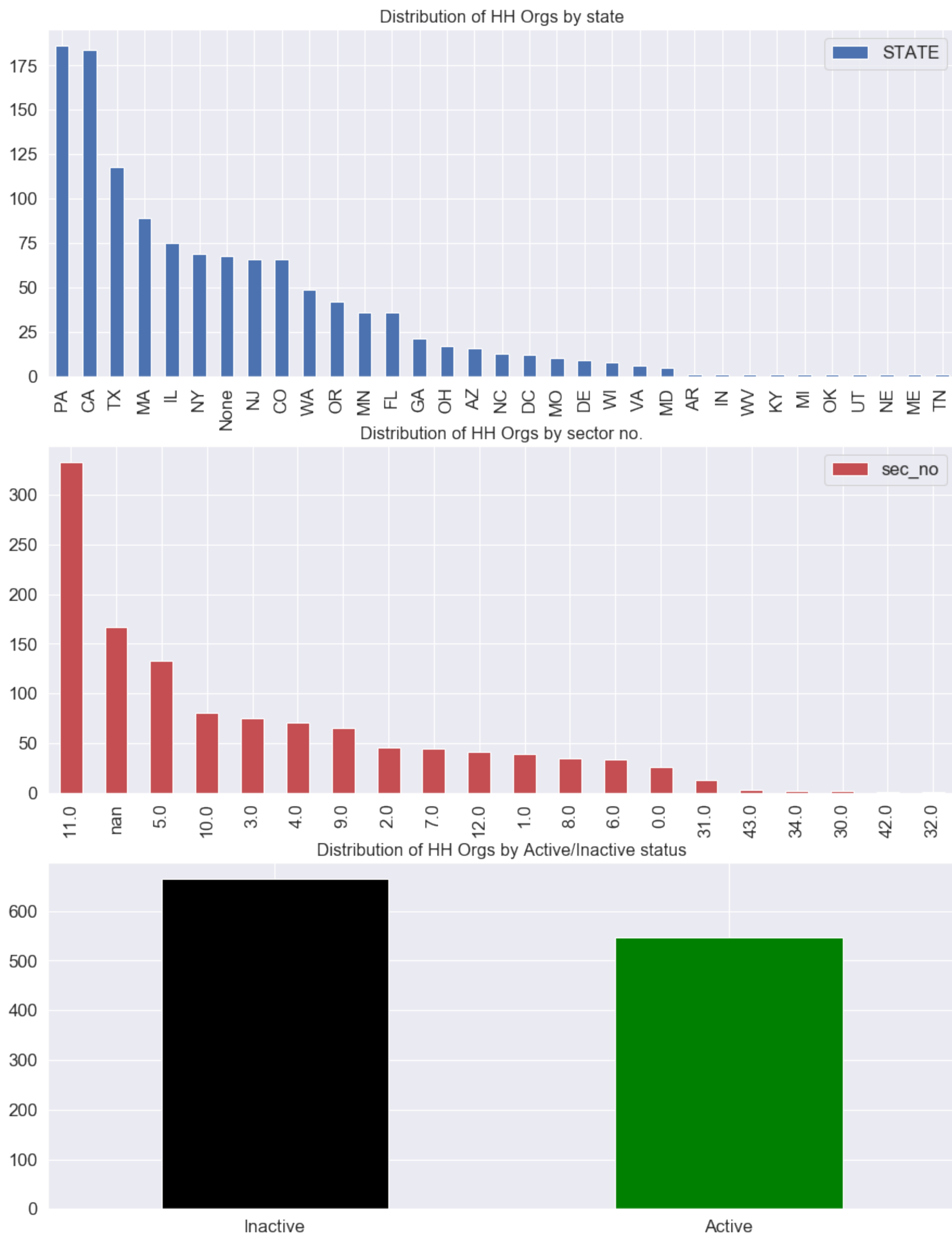
[6]:
```
sqlquery='select * from HHOrgStatic'
hhIntDF=load_data(cnxnOrg,sqlquery)
hhIntDF.head()
```

[6]:
```
     NCARID  OrgID                     ORGName                  ADDRESS  \
0  154202.0   1516              Barter Theatre               PO Box 867
1  150159.0    186          WaterTower Theatre         15650 Addison Rd
2  162722.0    851     Front Porch Theatricals  112 Sewickley Ridge Cir
3  146464.0   1083          Baum School of Art           510 W Linden St
4  146462.0   1084  Lehigh Valley Arts Council          840 Hamilton St

          CITY STATE      ZIP        ZIP9  STATENO   County        FTRACT  \
0     ABINGDON    VA  24212.0  24212-0867     51.0  51191.0  5.119101e+10
1      ADDISON    TX  75001.0  75001-3285     48.0  48113.0  4.811301e+10
2  ALEPPO TWP    PA  15143.0  15143-8978     42.0  42003.0  4.200356e+10
3    ALLENTOWN    PA  18101.0  18101-1416     42.0  42077.0  4.207701e+10
4    ALLENTOWN    PA  18101.0  18101-2455     42.0  42077.0  4.207701e+10

   CensusBlock      CNTYNM     CBSA   LATITUDE  LONGITUDE  Active  \
0       3011.0  WASHINGTON  28700.0  36.706928 -81.974386       0
1       2012.0      DALLAS  19124.0  32.962209 -96.829781       1
2       2028.0   ALLEGHENY  38300.0  40.540856 -80.147076       0
3       1001.0      LEHIGH  10900.0  40.604335 -75.469017       0
4       1035.0      LEHIGH  10900.0  40.601318 -75.474660       1

   InactiveDate  sec_no
0      201807.0    11.0
1           NaN    11.0
2      201711.0     8.0
3      201709.0     1.0
4           NaN     3.0
```

[7]:
```
#- Categorical Distributions
fig=plt.figure(figsize=(15,20))
ax1=plt.subplot(311)
hhIntDF['STATE'].astype(str).value_counts().plot(kind='bar')
ax1.legend()
plt.title('Distribution of HH Orgs by state',fontsize=16)
ax2=plt.subplot(312)
hhIntDF['sec_no'].astype(str).value_counts().plot(kind='bar',color='r')
ax2.legend()
plt.title('Distribution of HH Orgs by sector no.',fontsize=16)
ax3=plt.subplot(313)
hhIntDF['Active'].astype(str).value_counts().plot.bar(color=['Black','Green'])
```

```
#hhIntDF['Active'].astype(str).value_counts().
 ↪plot(kind='bar',color=['Black','Green'],label='Inactive')
ax3.set_xticklabels(['Inactive','Active'],rotation=0)
plt.title('Distribution of HH Orgs by Active/Inactive status',fontsize=16)
```

[7]: Text(0.5, 1.0, 'Distribution of HH Orgs by Active/Inactive status')



1.0.2 HHOrgData

[8]:
```
#- look at the schema
n=0
for row in cursor.columns(table='HHOrgData'):
    if n<=20:  #- only looking at the first 20 fields. Total 411
        if row[1]=='dbo':
            print(row[3],row[5])
    n+=1
```

```
OrgID bigint
year bigint
CNTART float
MKTADV float
ARTSATCD float
FRATNDTO float
PDATND float
ALLATTTO float
BOARDCD float
TRUSTNCD float
ENDTOTCD float
FTEMPS float
FTSEAS float
FTVOLS float
DEVSATCD float
GASAT float
HITIX float
LOTIX float
DMAILN float
MKTTOT float
MKTSAT float
```

HHOrgData Table consists of 411 variables with OrgID, year and the the remaining 409 numeric variables for the organizations (that have reported HH transactions) spanning from 2008 through 2019. Let's see some description below.

[9]:
```
#Load the HH ORg data
sqlquery='select * from HHOrgData'
HHcompDF=load_data(cnxnOrg,sqlquery)
HHcompDF.head()
```

[9]:
```
   OrgID  year   CNTART  MKTADV  ARTSATCD  FRATNDTO    PDATND  ALLATTTO  \
0   1012  2008  36598.0  7026.0       NaN     100.0    5003.0    5103.0
1   1012  2009  57887.0  8755.0       NaN     300.0    3925.0    4225.0
2   1012  2010  37799.0  2219.0       NaN       0.0  105260.0  105260.0
```

```
3   1012  2011      0.0   400.0      NaN      0.0  120545.0  120545.0
4   1012  2012  33819.0   713.0  76718.0    491.0       0.0    4100.0

   BOARDCD  TRUSTNCD  …  GABENCD  PRGBENCD  UWEBVIS  ArtsActivity  \
0     10.0       8.0  …  10268.0   21800.0      0.0      0.233247
1     10.0       9.0  …   4325.0   25390.0      0.0      0.524995
2     10.0       7.0  …   1662.0   22111.0      0.0      0.509109
3      9.0       9.0  …   2088.0   26273.0      0.0      0.518762
4      8.0       8.0  …   2983.0   24379.0      0.0      0.497554

   ArtsProviders  GrantActivity  Hospitality  Substitute  SocioEcon  \
0       0.310376      -0.027368     0.187422   -0.147217  -0.093418
1       0.191257      -0.017818     0.327811   -0.200109  -0.096941
2       0.221737       0.444290     0.205557   -0.152727   0.383591
3       0.341954      -0.444695     0.204866   -0.135336   0.431744
4       0.184335      -0.444695     0.177981   -0.245605   0.528201

         TOTPOP
0  11406.837973
1  11406.837973
2  11485.493201
3  11553.009420
4  11590.029118

[5 rows x 410 columns]
```

[10]:

```python
#- For display, let's take a subset and look at some correlation
selected_fields=['ArtsActivity', 'ArtsProviders',
       'GrantActivity', 'Hospitality', 'Substitute', 'SocioEcon',
       'TOTPOP']
HHcomp_subset=HHcompDF[selected_fields]
HHcomp_subset.describe()
```

[10]:

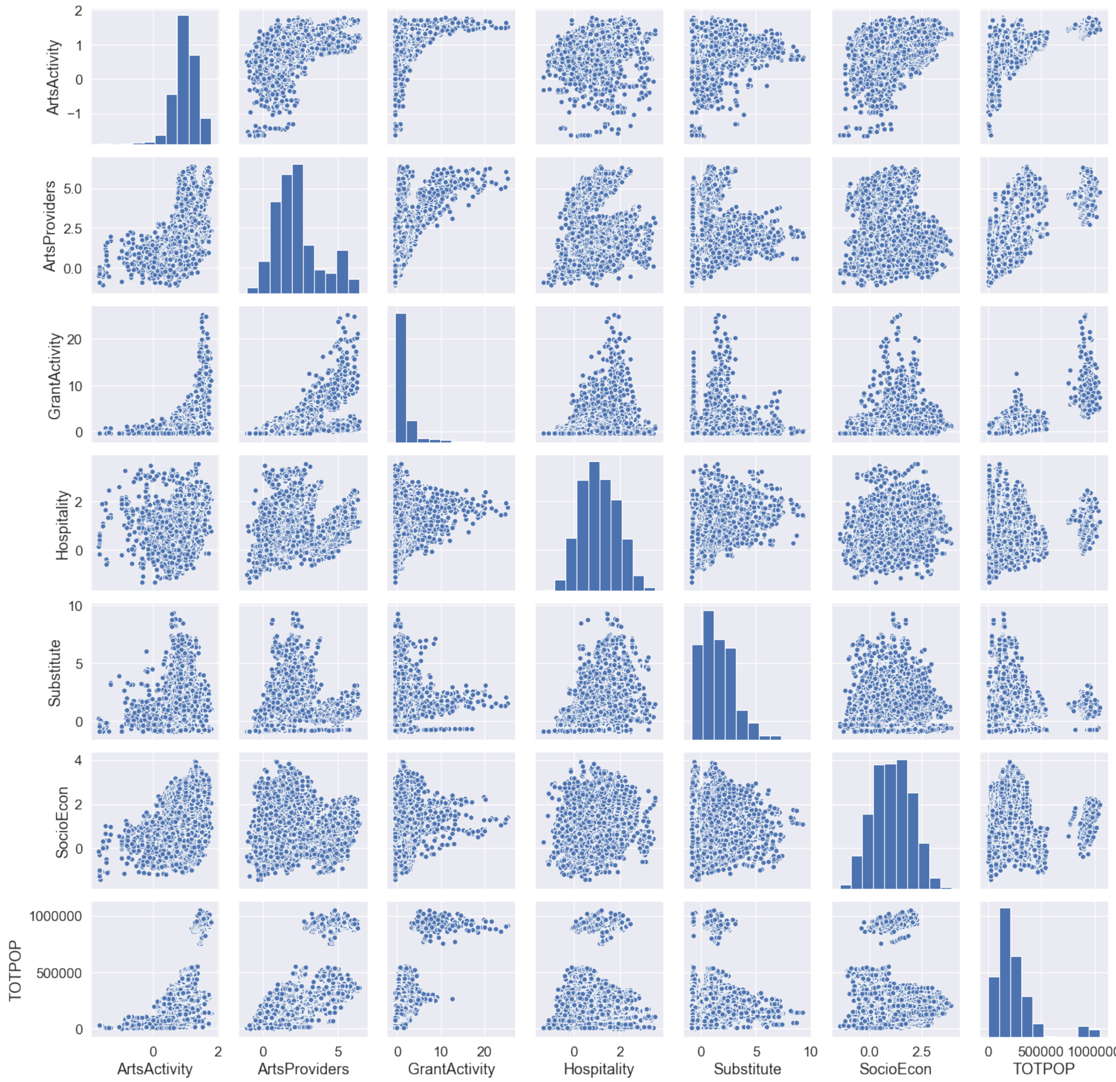
```
       ArtsActivity  ArtsProviders  GrantActivity   Hospitality    Substitute  \
count  13246.000000   13246.000000   13246.000000  13246.000000  13246.000000
mean       0.974061       2.271096       1.723004      1.094885      1.448059
std        0.381326       1.519767       2.821947      0.798728      1.603717
min       -1.655881      -1.041982      -0.444695     -1.328030     -0.879976
25%        0.791436       1.200293       0.351602      0.504058      0.376510
50%        0.994557       2.009976       1.047828      1.047564      1.275327
75%        1.207564       2.867896       1.939133      1.691370      2.495204
max        1.786953       6.376923      25.179070      3.550326      9.360109

          SocioEcon        TOTPOP
count  13246.000000  1.324600e+04
mean       1.096317  2.470085e+05
std        0.900768  2.000732e+05
```

```
min      -1.422394  1.538331e+03
25%       0.420591  1.273388e+05
50%       1.082471  1.902962e+05
75%       1.760716  3.074388e+05
max       3.935416  1.051378e+06
```

One can look at the correlations in a pair plot

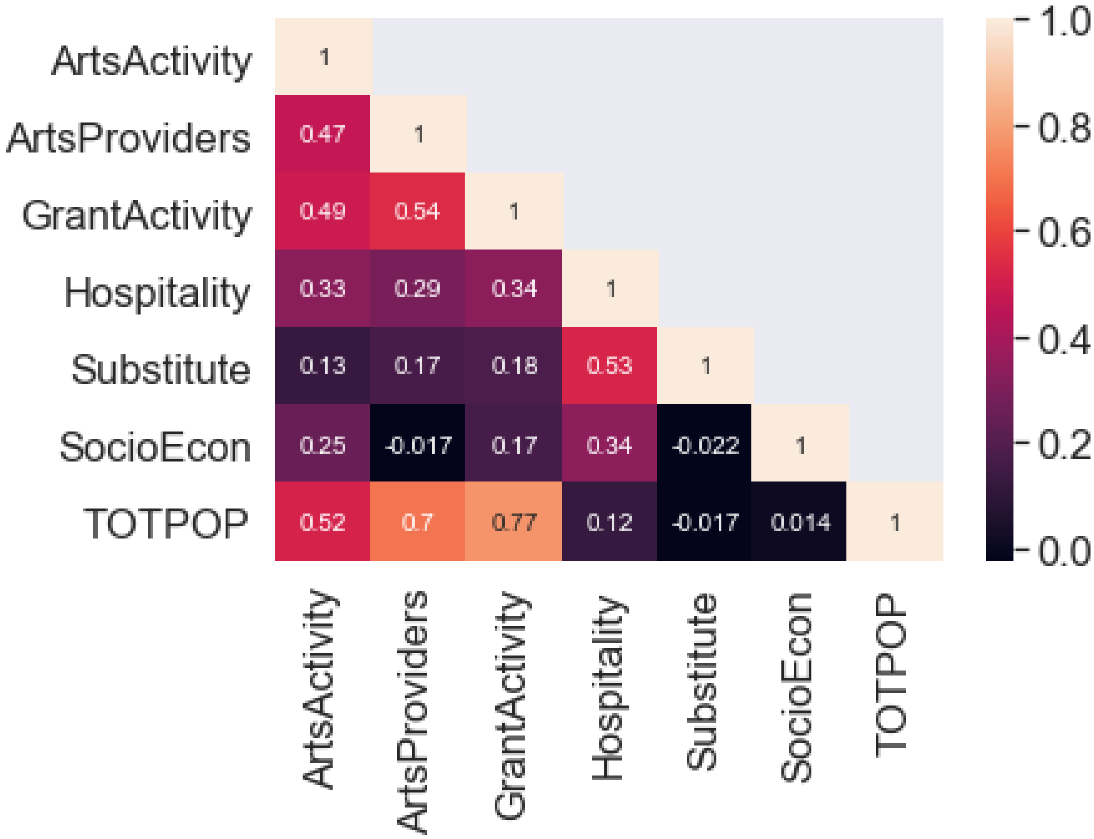
[11]:
```
sns.pairplot(HHcomp_subset)
```

[11]: <seaborn.axisgrid.PairGrid at 0x1a236557d0>



Or one can also create a correlation matrix/see the overall correlation coefficients across the variables

[12]:
```python
corrMatrix=HHcomp_subset.corr()
corrMatrix=corrMatrix.where(np.tril(np.ones(corrMatrix.shape)).astype(np.bool))
#mask = np.triu(np.ones_like(corrMatrix, dtype=np.bool))
sns.heatmap(corrMatrix,annot=True)
plt.show()
```



The tables above can be joined by the ORGID/householdID. In this framework the join can be performed in the SQL query itself, or at the dataframe level. For larger tables, it is more efficient to perform the join operations in the SQL query itself

[13]:
```python
HHcomp_subset.head()
```

[13]:
```
   ArtsActivity  ArtsProviders  GrantActivity  Hospitality  Substitute  \
0      0.233247       0.310376      -0.027368     0.187422   -0.147217
1      0.524995       0.191257      -0.017818     0.327811   -0.200109
2      0.509109       0.221737       0.444290     0.205557   -0.152727
3      0.518762       0.341954      -0.444695     0.204866   -0.135336
4      0.497554       0.184335      -0.444695     0.177981   -0.245605

   SocioEcon      TOTPOP
```

0	-0.093418	11406.837973
1	-0.096941	11406.837973
2	0.383591	11485.493201
3	0.431744	11553.009420
4	0.528201	11590.029118

2 Database: HHDB

[14]:
```
dbname='HHDB'
cnxnHH=pyodbc.connect(driver=driver,server=server,database=dbname,uid=hhuid,\
                       pwd=hhpwd,port=port)
```

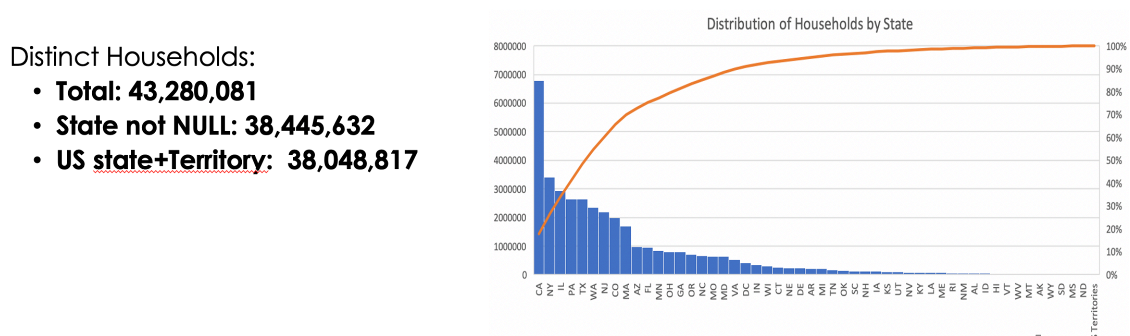
Checking the tables in this DB

[15]:
```
cursor = cnxnHH.cursor()
for row in cursor.tables(tableType='TABLE'):
    if row[1]=='dbo': #- avoiding system tables
        print(row[2])
```

```
HHActivity
HHStatic
```

2.0.1 Household Static

Static data showing geo coded Households



[16]:
```
sqlquery='select top 100 * from HHStatic'
hshldDF=load_data(cnxnHH,sqlquery)
hshldDF.head()
```

[16]:

	HouseholdID	CountyCode	FTract	BlockGroup	City	State	PostalCode	\
0	-40653585	None	None	None	None	None	None	
1	-23727456	None	None	None	None	None	None	
2	-139036295	None	None	None	None	None	None	
3	-133529841	None	None	None	Staten Island	NY	10305	
4	-124867765	None	None	None	None	None	None	

```
   Fipsstatecode
0            NaN
1            NaN
2            NaN
3           36.0
4            NaN
```

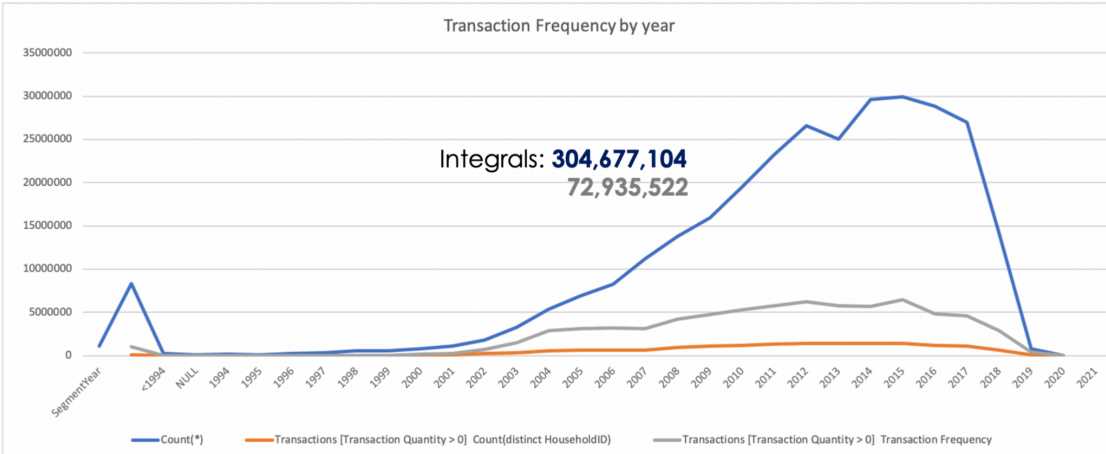
2.0.2 Household Activity

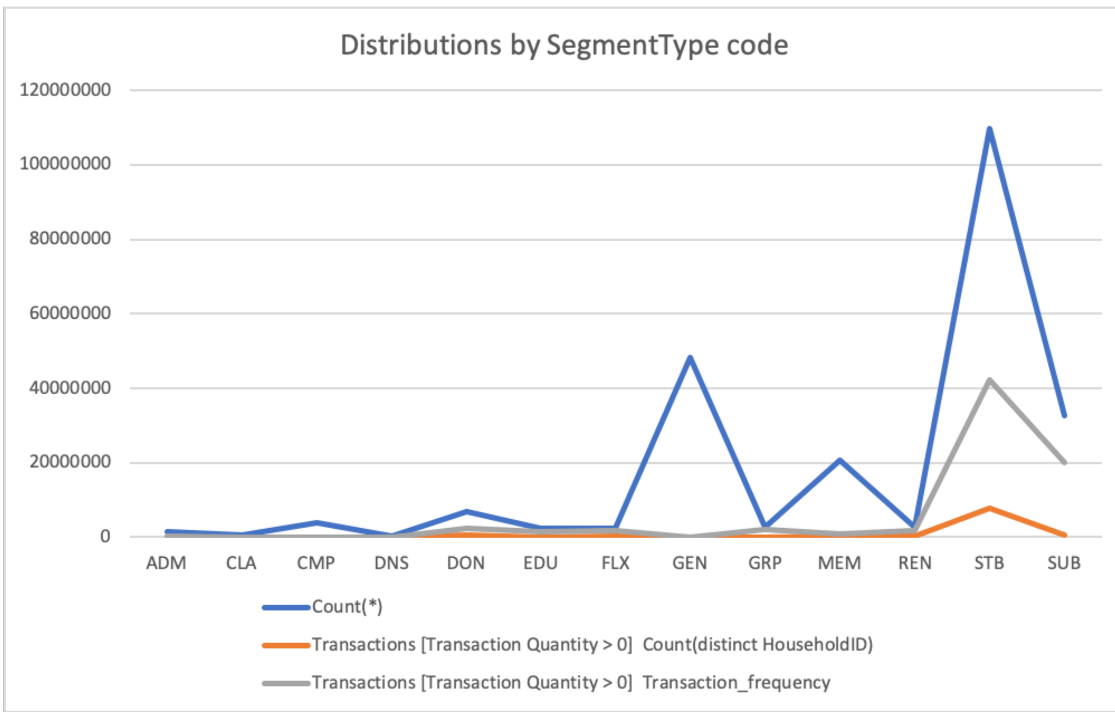
[17]:
```
#- look at the schema
for row in cursor.columns(table='HHActivity'):
    if row[1]=='dbo':
        print(row[3],row[5])
```

```
OrgID int
HouseholdID int
SegmentYear smallint
SegmentTypeCode varchar
SegmentDesc varchar
TransactionAmount money
TransactionQty int
OrderDate datetime
EventDate datetime
```

[18]:
```
sqlquery='select top 100 * from HHActivity'
ActDF=load_data(cnxnHH,sqlquery)
ActDF.head()
```

[18]:
```
   OrgID  HouseholdID  SegmentYear SegmentTypeCode SegmentDesc  \
0     95      2480252         2014             GEN     Dabbler
1     95      4166657         2014             GEN     Dabbler
2     95      4290532         2014             GEN     Dabbler
3     95      2571066         2014             GEN     Dabbler
4     95      5990076         2014             GEN     Dabbler

  TransactionAmount TransactionQty OrderDate EventDate
0              None           None      None      None
1              None           None      None      None
2              None           None      None      None
3              None           None      None      None
4              None           None      None      None
```





The tables above can be joined by the ORGID/householdID. In this framework the join can be performed in the SQL query itself, or at the dataframe level. For larger tables, it is more efficient to perform the join operations in the SQL query itself

3 Database: CensusDB

We also have cleaned and integrated Census TRACT and Block Group level data that can be merged with the HH data for TRACT and Block group level analyses. For this, the database is CensusDB

[19]:
```
#- We create a new connection instance
censusdb='CensusDB'
```

```
cnxnCNS=pyodbc.
↪connect(driver=driver,server=server,database=censusdb,uid=hhuid,pwd=hhpwd,port=port)
```

[20]:
```
#- checking the tables
cursor = cnxnCNS.cursor()
for row in cursor.tables(tableType='TABLE'):
    if row[1]=='dbo': #- avoiding system tables
        print(row[2])
```

```
BlkGrpcommute
BlkGrpecon
BlkGrpeduc
BlkGrplatin
BlkGrpLvl
BlkGrpmedhhinc
BlkGrppoverty
BlkGrprace
Tractdemo
Tractecon
Tracteduc
Tracthshld
TractLvl
```

The table names indicate the kinds of data in each table. The BlkGrp data span 2013-2019 and tract level data span from 2008-2019. The integrated tables are BlkGrpLvl and TractLvl and all the others are intermediate. Therefore we will only explore the final integrated tables at the Census Block Group and Census Tract level

3.0.1 BlkGrpLvl

[21]:
```
sqlquery='select top 100 * from BlkGrpLvl'
BlkGrpDF=load_data(cnxnCNS,sqlquery)
BlkGrpDF.head()
```

[21]:

	YEAR	STATE	BlkGrp	CommuteN	AvgCommute	TotHse	LT50P	\
0	2013	Alabama	10010201001	268	14.082090	205	34.146341	
1	2013	Alabama	10010201002	570	33.156140	411	49.635036	
2	2013	Alabama	10010202001	535	27.691589	439	50.569476	
3	2013	Alabama	10010202002	398	26.097990	394	62.436548	
4	2013	Alabama	10010204001	501	21.055888	416	29.807692	

	GT100P	GT125P	GT150P	…	GradPlusP	MedHInc	WHITP	\
0	39.512195	10.243902	0.000000	…	11.616162	72375	86.656201	
1	16.058394	9.245742	6.812652	…	10.574413	52788	87.446627	
2	18.451025	3.189066	0.000000	…	3.363519	46979	30.296457	
3	14.974619	4.314721	3.045685	…	11.500701	43438	36.728395	
4	29.326923	19.230769	11.057692	…	9.948980	69375	97.794118	

```
       BLCKP    AMINDP     ASIAP  HAWAP  TOTPOP  NotLat  Latin
0  13.343799  0.000000  0.000000    0.0     637     637      0
1   5.380017  0.853971  0.000000    0.0    1171    1171      0
2  62.039046  0.000000  6.290672    0.0    1383    1334     49
3  62.448560  0.000000  0.000000    0.0     972     970      2
4   0.000000  2.022059  0.000000    0.0    1088    1072     16

[5 rows x 24 columns]
```

So that shows the Block Group level economic demographic, commute time etc by year for each Block Group.

3.0.2 TractLvl

[22]:
```
sqlquery='select top 20 * from TractLvl'
TractDF=load_data(cnxnCNS,sqlquery)
TractDF.head()
```

[22]:
```
   YEAR       STATE       TRACT  POP16  LT50P  GT100P  GT150P  GT200P  \
0  2011  California  6037575401   3598   21.5     1.6     0.0     0.0
1  2011  California  6037575402   2334   19.3     0.9     0.0     0.0
2  2011  California  6037575500     37    NaN     NaN     NaN     NaN
3  2011  California  6037575801   1783   11.7     4.4     0.0     0.0
4  2011  California  6037575802   3510   22.2     0.9     2.2     1.4

   MedHInc  POVPERC  …  MarSze  MalHseSze  FemHseSze  NonFamSze  TotFam  \
0  35270.0     36.0  …    4.63       6.14       4.38       1.56     854
1  30900.0     26.4  …    3.80       3.88       4.12       1.57     751
2      NaN     45.9  …    0.00       0.00       0.00       0.00       0
3  32344.0     38.2  …    4.96       3.50       4.54       1.63     522
4  32109.0     37.0  …    4.59       3.37       4.41       1.43     982

   AvFamSze  MARKID18  MALKID18  FEMKID18  SameSex
0      4.39       295        95       242      0.0
1      3.63       273       111       172      2.4
2      0.00         0         0         0      0.0
3      4.31       231         7       147      0.1
4      3.99       279        42       264      1.5

[5 rows x 37 columns]
```

This shows Tract level data for education, demographics, economy etc.

4 ASIDE – Combining aka merging aka joining data sets

We show two ways to merge the data sets and pick Tract level census data to do so as an example

[23]:
```
#- Tract level Census data. we pick three tables
Tracttables=['Tractdemo','Tractecon','Tracteduc']
for tab in Tracttables:
    print("Table schema for : ", tab)
    for row in cursor.columns(table=tab):
        print(row[3],row[5])
```

```
Table schema for :  Tractdemo
TRACT bigint
TOTPOP bigint
WHIT bigint
BLCK bigint
AMIND bigint
ASIA bigint
HAWA bigint
LATIN bigint
YEAR bigint
STATE varchar
Table schema for :  Tractecon
TRACT bigint
POP16 bigint
LT50P float
GT100P float
GT150P float
GT200P float
MEDHINC float
PovPerc float
YEAR bigint
STATE varchar
Table schema for :  Tracteduc
TRACT bigint
POP25 bigint
BACHP float
GRADP float
BachPlusP float
YEAR bigint
STATE varchar
```

As we see, we have TRACT, YEAR, STATE in all Tract tables, so we will use these to join the tables.

Using SQL join query – fast

[24]:
```
sqlquery='select a.TRACT,a.YEAR,a.STATE,TOTPOP,WHIT,BLCK,AMIND,ASIA,HAWA,LATIN,\
          POP16,LT50P,GT100P,GT150P,GT200P,PovPerc,\
          POP25,BACHP,GRADP,BachPlusP from Tractecon a \
          full outer join Tracteduc b \
          on a.TRACT=b.TRACT and a.YEAR=b.YEAR and a.STATE=b.STATE \
```

```
        full outer join Tractdemo c \
        on a.TRACT=c.TRACT and a.YEAR=c.YEAR and a.STATE=c.STATE'
```

[25]:
```
#%%timeit
#TRACT_dataDF=load_data(cnxnCNS,sqlquery)
#==> 2min 16s ± 12.5 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[26]:
```
t1 = datetime.datetime.now()
TRACT_dataDF=load_data(cnxnCNS,sqlquery)
t2 = datetime.datetime.now()
print("Time taken to execute the query and load to DF  [Seconds] ", (t2-t1).
 ↪seconds)
```

```
Time taken to execute the query and load to DF  [Seconds]  149
```

[27]:
```
print(TRACT_dataDF.shape)
TRACT_dataDF.head()
```

```
(814013, 20)
```

[27]:
```
        TRACT  YEAR    STATE  TOTPOP    WHIT   BLCK  AMIND  ASIA  HAWA  LATIN  \
0  1001020100  2008  Alabama  1852.0  1552.0  291.0   67.0   0.0   0.0   15.0
1  1001020100  2010  Alabama  1809.0  1516.0  330.0   77.0   0.0   0.0   15.0
2  1001020100  2011  Alabama  1768.0  1560.0  223.0  107.0   4.0   0.0    0.0
3  1001020100  2013  Alabama  1808.0  1650.0  170.0   57.0  14.0   0.0    0.0
4  1001020100  2016  Alabama  2010.0  1737.0  298.0    6.0  17.0  21.0   53.0

   POP16  LT50P     GT100P   GT150P    GT200P    PovPerc   POP25      BACHP  \
0   1396   14.7  18.021468  1.88031  5.956905   9.091817  1234.0  11.050633
1   1392   14.7  21.500000  2.00000  7.000000  10.500000  1242.0  13.700000
2   1398   17.2  21.300000  4.90000  5.800000  10.200000  1284.0  10.800000
3   1404   13.1  24.200000  4.90000  1.300000  10.500000  1162.0  15.700000
4   1580    7.9  18.700000  8.10000  0.700000   9.900000  1298.0  16.600000

       GRADP  BachPlusP
0   9.729163  20.750948
1  11.800000  25.400000
2   9.100000  19.900000
3  10.900000  26.700000
4  14.700000  31.400000
```

Using individual dataframe – slow

[28]:
```
#%%timeit
#squery='select * from Tractecon'
#testDF=load_data(cnxnCNS,squery)
```

[29]:
```python
t3 = datetime.datetime.now()
squery1='select * from Tractecon'
squery2='select * from Tracteduc'
squery3='select * from Tractdemo'

print("Reading Tract economy data")
econDF=load_data(cnxnCNS,squery1)
print("Reading Tract education data")
educDF=load_data(cnxnCNS,squery2)
print("Reading Tract demographics data")
demoDF=load_data(cnxnCNS,squery3)

tract_mergeDF1=econDF.merge(educDF,on=['TRACT','YEAR','STATE'],how='outer')
tract_mergeDF2=tract_mergeDF1.
 ↪merge(demoDF,on=['TRACT','YEAR','STATE'],how='outer')

t4 = datetime.datetime.now()
print("Time taken on full data queries and DF merge [Seconds] ", (t4-t3).
 ↪seconds)
```

```
Reading Tract economy data
Reading Tract education data
Reading Tract demographics data
Time taken on full data queries and DF merge [Seconds]  152
```

[30]:
```python
print(tract_mergeDF2.shape)
tract_mergeDF2.head()
```

```
(814013, 21)
```

[30]:

	TRACT	POP16	LT50P	GT100P	GT150P	GT200P	MEDHINC	\
0	1001020100	1396	14.7	18.021468	1.880310	5.956905	60255.0	
1	1001020200	1516	17.3	13.474851	0.298741	1.568570	34570.0	
2	1001020300	2549	21.8	11.497938	3.663303	0.458445	37101.0	
3	1001020400	3638	15.6	13.656101	3.482013	1.328458	48153.0	
4	1001020500	6948	12.5	18.500227	5.361798	0.970599	58256.0	

	PovPerc	YEAR	STATE	…	BACHP	GRADP	BachPlusP	TOTPOP	\
0	9.091817	2008	Alabama	…	11.050633	9.729163	20.750948	1852.0	
1	12.967858	2008	Alabama	…	14.157831	7.590959	21.930150	2045.0	
2	6.914586	2008	Alabama	…	11.327994	1.362692	12.643148	3443.0	
3	5.438941	2008	Alabama	…	13.756875	6.813766	20.713601	4639.0	
4	5.378651	2008	Alabama	…	21.315216	9.980556	31.834601	9339.0	

	WHIT	BLCK	AMIND	ASIA	HAWA	LATIN
0	1552.0	291.0	67.0	0.0	0.0	15.0
1	855.0	1128.0	0.0	22.0	0.0	6.0

```
2  2891.0   539.0    0.0   31.0   0.0   39.0
3  4486.0    85.0   22.0   14.0   0.0  128.0
4  8067.0  1131.0   88.0  146.0   0.0  471.0

[5 rows x 21 columns]
```