HH_Census_datasets_DataArts

June 11, 2020

1 HHDB Database

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

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='HHDB'
port=1433
cnxnHH=pyodbc.connect(driver=driver,server=server,database=dbname,uid=hhuid,\
                      pwd=hhpwd,port=port)
```

Check the tables in the database

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

```
HHActivity
HHOrgCompany
HHOrgCompanyStats
HHOrgStatic
HHStatic
```

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

[4]:
```
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

[5]:
```
#- 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
```

```
NetworkCode varchar
NetworkName varchar
Active bigint
InactiveDate float
AnnualRevenue float
AnnualRevenueYear float
PostalCode varchar
TRG_Genre varchar
sec_no float
```

[6]:
```
sqlquery='select * from HHOrgStatic'
hhIntDF=load_data(cnxnHH,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	…	LONGITUDE	\
0	ABINGDON	VA	24212.0	24212-0867	51.0	51191.0	…	-81.974386	
1	ADDISON	TX	75001.0	75001-3285	48.0	48113.0	…	-96.829781	
2	ALEPPO TWP	PA	15143.0	15143-8978	42.0	42003.0	…	-80.147076	
3	ALLENTOWN	PA	18101.0	18101-1416	42.0	42077.0	…	-75.469017	
4	ALLENTOWN	PA	18101.0	18101-2455	42.0	42077.0	…	-75.474660	

	NetworkCode	NetworkName	Active	InactiveDate	\
0	None	None	0	201807.0	
1	CNNT	TRG Community: North Texas	1	NaN	
2	GPAC	Greater Pittsburgh Arts Counc	0	201711.0	
3	CNPH	TRG Community: Philadelphia	0	201709.0	
4	CNPH	TRG Community: Philadelphia	1	NaN	

	AnnualRevenue	AnnualRevenueYear	PostalCode	TRG_Genre	\
0	8392321.0	2016.0	24210-3202	Education - Performing Arts	
1	1418207.0	2018.0	75001-3285	Theater	
2	145000.0	2016.0	15143-8978	Theater	
3	0.0	0.0	18101-1416	Museum - Visual Art/Gallery	
4	253347.0	2017.0	18101-2456	Community/Cultural Center	

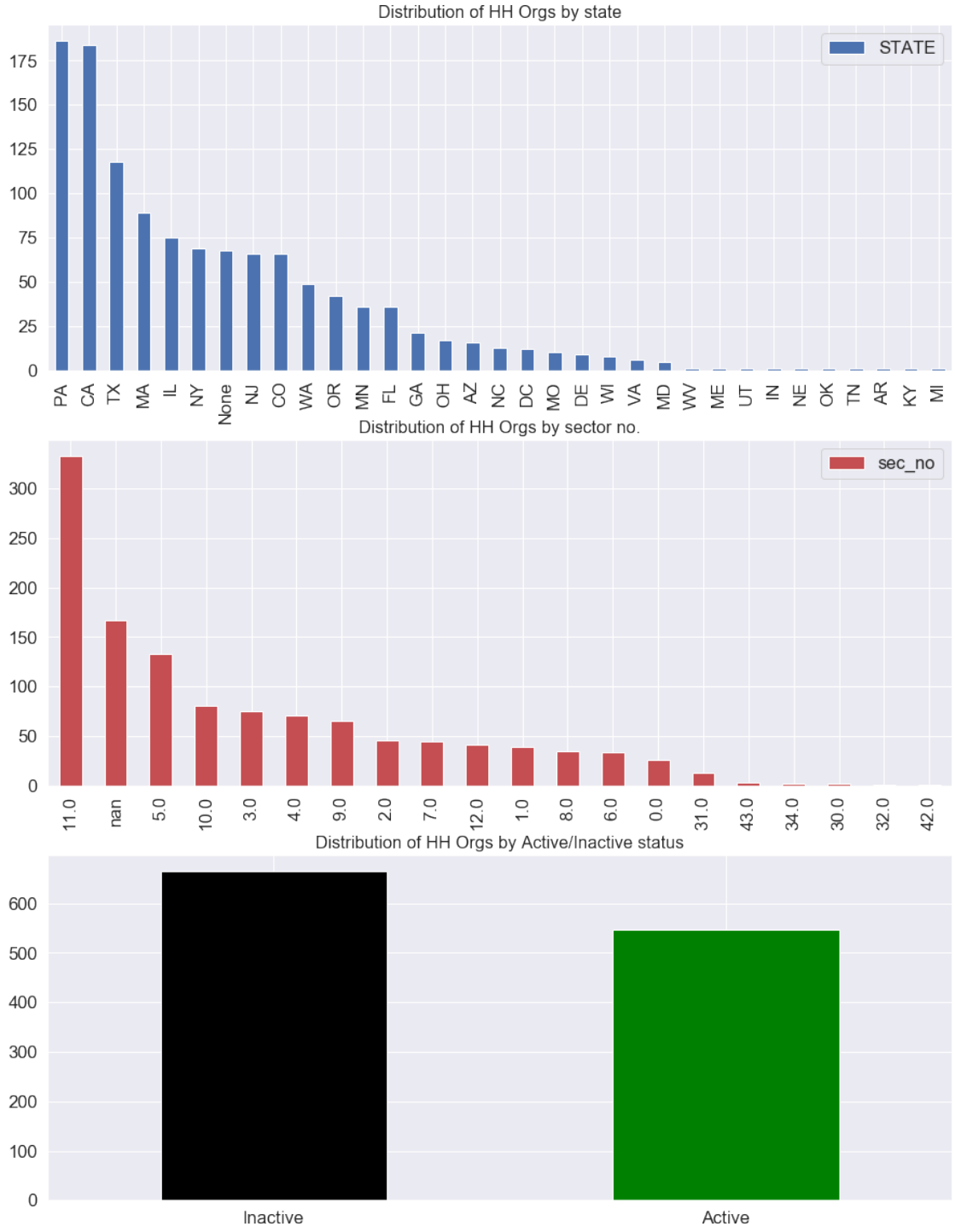
	sec_no
0	11.0
1	11.0
2	8.0
3	1.0

```
4     3.0

[5 rows x 25 columns]
```

[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')
```


Distribution of HH Orgs by state
STATE
175
150
125
100
75
50
25
0
PA
CA
TX
MA
IL
NY
None
NJ
CO
WA
OR
MN
FL
GA
OH
AZ
NC
DC
MO
DE
WI
VA
MD
WV
ME
UT
IN
NE
OK
TN
AR
KY
MI
Distribution of HH Orgs by sector no.
sec_no
300
250
200
150
100
50
0
11.0
nan
5.0
10.0
3.0
4.0
9.0
2.0
7.0
12.0
1.0
8.0
6.0
0.0
31.0
43.0
34.0
30.0
32.0
42.0
Distribution of HH Orgs by Active/Inactive status
600
500
400
300
200
100
0
Inactive
Active

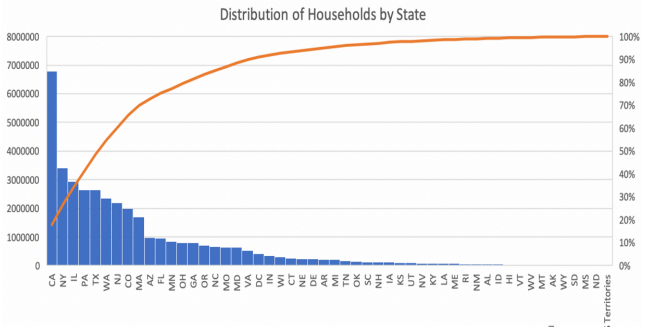
1.0.2 Household data

[8]:
```
#- look at the schema
for row in cursor.columns(table='HHStatic'):
    if row[1]=='dbo':
        print(row[3],row[5])
```

```
HouseholdID bigint
CountyCode varchar
FTract varchar
BlockGroup varchar
City varchar
State varchar
PostalCode varchar
Fipsstatecode float
```

Distinct Households:

- **Total: 43,280,081**
- **State not NULL: 38,445,632**
- **US state+Territory: 38,048,817**



[9]:
```
sqlquery='select top 100 * from HHStatic'
hshldDF=load_data(cnxnHH,sqlquery)
hshldDF.head()
```

[9]:

	HouseholdID	CountyCode	FTract	BlockGroup	City	State	PostalCode	Fipsstatecode
0	-40653585	None	None	None	None	None	None	NaN
1	-23727456	None	None	None	None	None	None	NaN
2	-139036295	None	None	None	None	None	None	NaN
3	-133529841	None	None	None	Staten Island	NY	10305	36.0
4	-124867765	None	None	None	None	None	None	NaN

1.0.3 Activity

[10]:
```
#- 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
```

[11]:
```
sqlquery='select top 100 * from HHActivity'
ActDF=load_data(cnxnHH,sqlquery)
ActDF.head()
```

[11]:

	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

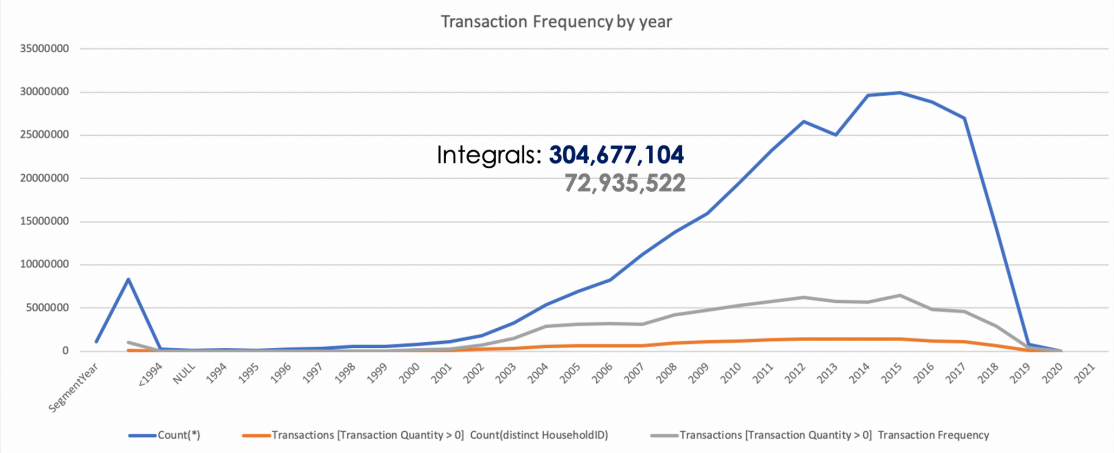
[12]:
```
#Checking where Transaction information is available
sqlquery='select top 100 * from HHActivity where TransactionQty>0'
ActDF=load_data(cnxnHH,sqlquery)
ActDF.head()
```

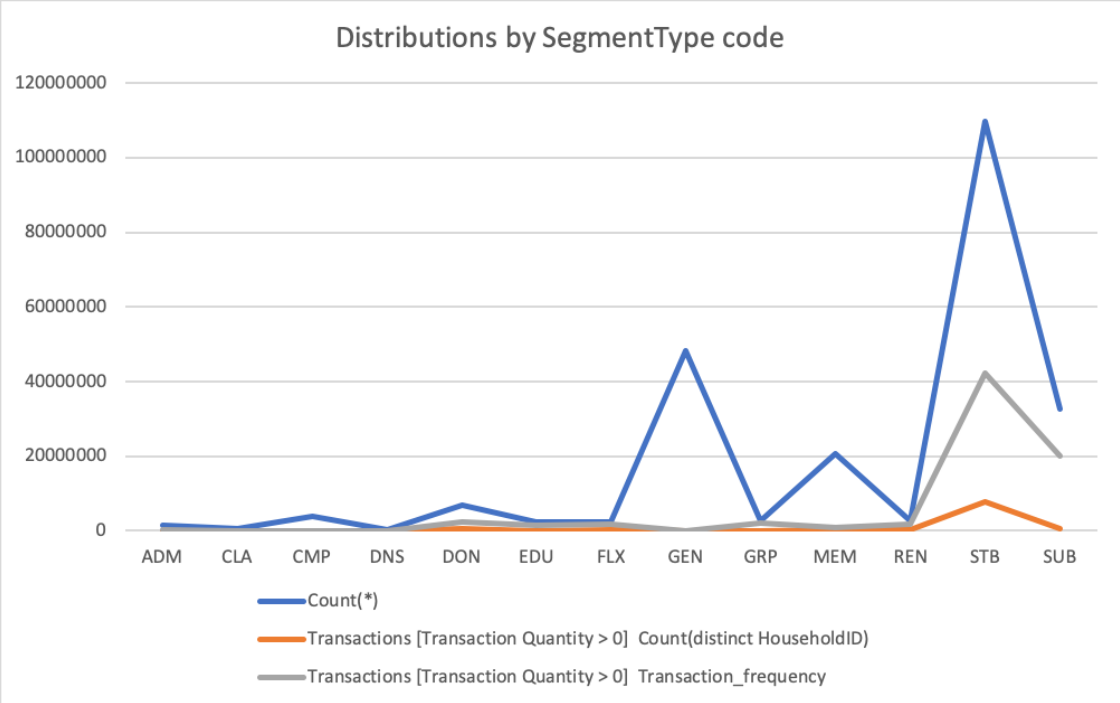
[12]:

	OrgID	HouseholdID	SegmentYear	SegmentTypeCode	SegmentDesc	\
0	280	591215	2013	STB	Bethany	
1	280	7688452	2014	STB	Row After Row	
2	280	14598194	2013	STB	Bethany	
3	280	694624	2013	STB	Jackie	
4	280	7614365	2013	STB	Collapse	

	TransactionAmount	TransactionQty	OrderDate	EventDate

0	75.0	1	2013-01-30 11:51:00	2013-02-17 14:30:00
1	60.0	3	2014-01-21 09:13:00	2014-02-09 14:30:00
2	45.0	2	2013-01-27 20:19:00	2013-02-02 14:30:00
3	45.0	2	2013-02-19 21:25:00	2013-03-17 14:30:00
4	40.0	2	2013-01-31 15:11:00	2013-04-27 19:30:00





1.0.4 HHOrgCompany

[13]:
```
#- look at the schema
n=0
for row in cursor.columns(table='HHOrgCompany'):
    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
```

HHOrgCompany Table consists of 411 variables with OrgID, year and the the remaining 409 numeric variables for the HH correlated organizations spanning from 2008 through 2019. The description of the numeric fields are given in the HHOrgCompanyStats table. But let's see some description below as well.

[14]:
```
#Load the HH ORg company data
sqlquery='select * from HHOrgCompany'
HHcompDF=load_data(cnxnHH,sqlquery)
HHcompDF.head()
```

[14]:

	OrgID	year	CNTART	MKTADV	ARTSATCD	FRATNDTO	PDATND	ALLATTTO	\
0	988	2011	26207.0	192405.0	NaN	21705.0	106971.0	128676.0	
1	988	2012	31336.0	168825.0	0.0	18459.0	0.0	122142.0	
2	988	2013	23730.0	192526.0	0.0	17226.0	0.0	121901.0	
3	988	2014	16711.0	0.0	0.0	0.0	0.0	120905.0	
4	988	2015	22312.0	0.0	0.0	0.0	0.0	127928.0	

	BOARDCD	TRUSTNCD	…	GABENCD	PRGBENCD	UWEBVIS	ArtsActivity	\
0	35.0	34.0	…	217996.0	250668.0	200000.0	-0.618403	
1	35.0	34.0	…	199854.0	194426.0	0.0	-0.219790	
2	35.0	34.0	…	278410.0	163688.0	0.0	-0.165559	
3	32.0	32.0	…	249765.0	148773.0	409693.0	-0.352757	

```
4     30.0      23.0  …  273109.0  154471.0  460773.0     -1.309778

   ArtsProviders  GrantActivity  Hospitality  Substitute  SocioEcon  \
0       1.853958      -0.444695     1.466882    3.122040   0.703857
1       1.751128      -0.195203     1.552820    3.084513   0.775479
2       1.977572      -0.444695     1.331060    3.061206   0.738692
3       2.034108      -0.444695     1.360293    3.247793   0.894977
4       1.969259      -0.444695     1.355147    3.130012   0.996784

         TOTPOP
0  6758.877079
1  6761.001879
2  6802.624666
3  6795.824603
4  6818.913158

[5 rows x 410 columns]
```

[15]:
```
#- 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()
```

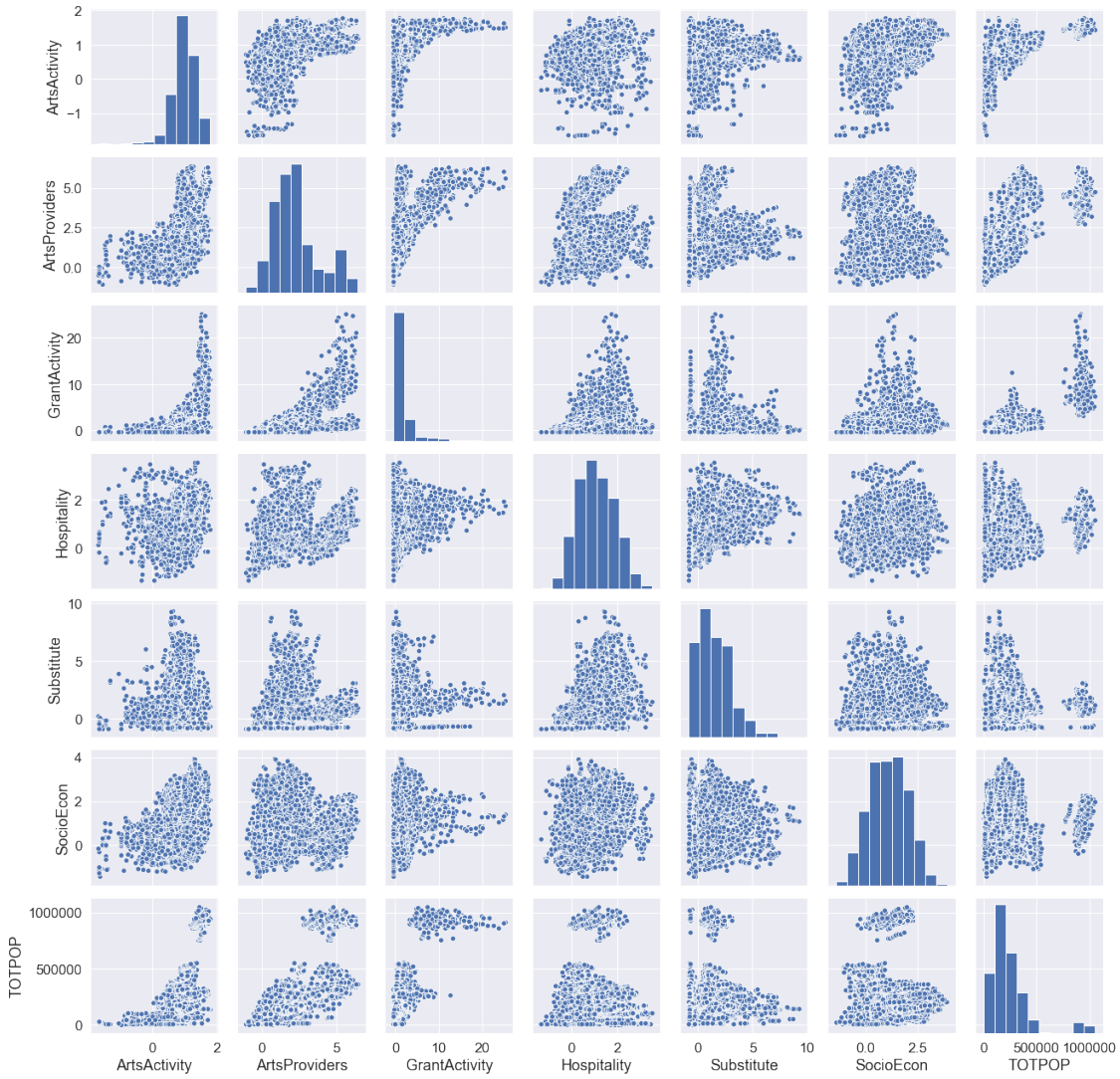
[15]:
```
       ArtsActivity  ArtsProviders  GrantActivity   Hospitality    Substitute  \
count  13246.000000   13246.000000   13246.000000  13246.000000  13246.000000
mean       0.974061       2.271096       1.723004      1.094885      1.448059
std        0.381326       1.519767       2.821947      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

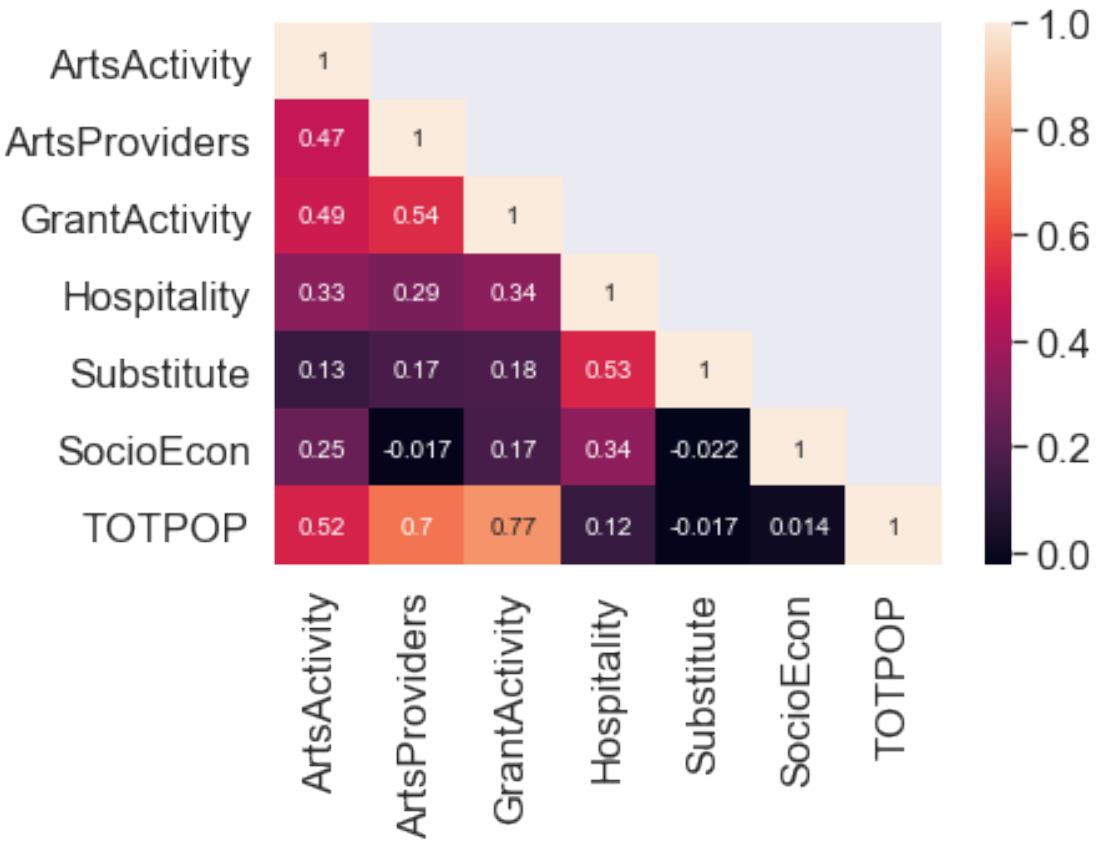
[16]:
```
sns.pairplot(HHcomp_subset)
```

[16]: <seaborn.axisgrid.PairGrid at 0x1a23c0bd50>



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

[17]:
```
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

[18]:
```
HHcomp_subset
```

[18]:

	ArtsActivity	ArtsProviders	GrantActivity	Hospitality	Substitute	\
0	-0.618403	1.853958	-0.444695	1.466882	3.122040	
1	-0.219790	1.751128	-0.195203	1.552820	3.084513	
2	-0.165559	1.977572	-0.444695	1.331060	3.061206	
3	-0.352757	2.034108	-0.444695	1.360293	3.247793	
4	-1.309778	1.969259	-0.444695	1.355147	3.130012	
…	…	…	…	…	…	
13241	0.487631	-0.085477	-0.444695	-0.256045	0.176473	
13242	0.521980	-0.112383	-0.444695	-0.257550	0.224276	
13243	0.378935	-0.198211	-0.444695	-0.220531	0.326361	
13244	0.559540	-0.506832	-0.443646	-0.855271	-0.818810	
13245	0.491091	-0.506832	-0.443646	-0.855271	-0.818810	

	SocioEcon	TOTPOP

```
0        0.703857   6758.877079
1        0.775479   6761.001879
2        0.738692   6802.624666
3        0.894977   6795.824603
4        0.996784   6818.913158
…           …           …
13241    0.371721  71287.656183
13242    0.370371  71744.895173
13243    0.399870  72133.488822
13244    0.510956  73720.210891
13245    0.514528  73720.210891

[13246 rows x 7 columns]
```

2 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
BlkGrpmedhhinc
BlkGrppoverty
BlkGrprace
Tractdemo
Tractecon
Tracteduc
Tracthshld
```

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. Lets see some of the data

2.0.1 BlkGrpcommute

[21]:
```
sqlquery='select * from BlkGrpcommute'
BlkComDF=load_data(cnxnCNS,sqlquery)
BlkComDF.head()
```

[21]:

	BlkGrp	YEAR	CommuteN	AvgCommute	STATE
0	270332701002	2018	401	13.264339	Minnesota
1	270332701003	2018	503	16.127237	Minnesota
2	270332702002	2018	307	22.446254	Minnesota
3	270332702003	2018	208	22.817308	Minnesota
4	270332703002	2018	412	16.371359	Minnesota

So that shows the average commute time by year for each BlkGrp.

2.0.2 BlkGrpecon

[22]:
```
sqlquery='select * from BlkGrpecon'
BlkeconDF=load_data(cnxnCNS,sqlquery)
BlkeconDF.head()
```

[22]:

	BlkGrp	YEAR	TotHse	LT50p	GT100p	GT125p	GT150p	GT200p	STATE
0	270332701002	2018	435	0.450575	0.055172	0.022989	0.018391	0.002299	Minnesota
1	270332701003	2018	539	0.551020	0.072356	0.040816	0.040816	0.024119	Minnesota
2	270332702002	2018	288	0.423611	0.197917	0.093750	0.052083	0.038194	Minnesota
3	270332702003	2018	218	0.399083	0.142202	0.055046	0.032110	0.027523	Minnesota
4	270332703002	2018	322	0.285714	0.298137	0.121118	0.077640	0.012422	Minnesota

This shows the economy data for each BlkGrp by year. And so on is the data for education, ethnicity, race and poverty. The tract level data also include the same information for the tract levels.

3 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 varchar
GT100P varchar
GT150P varchar
GT200P varchar
MEDHINC varchar
PovPerc varchar
YEAR bigint
STATE varchar
Table schema for :  Tracteduc
TRACT bigint
POP25 bigint
BACHP varchar
GRADP varchar
BachPlusP varchar
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]  117
```

[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	\
0	1396	14.699999999999999	18.021468290000001	1.8803097219999998	
1	1392	14.699999999999999	21.5	2.0	
2	1398	17.2	21.3	4.9	
3	1404	13.1	24.2	4.9	
4	1580	7.9	18.7	8.1	

	GT200P	PovPerc	POP25	BACHP	\
0	5.9569049620000003	9.0918167759999999	1234.0	11.050633270000001	
1	7.0	10.5	1242.0	13.7	
2	5.8	10.2	1284.0	10.8	
3	1.3	10.5	1162.0	15.7	
4	0.7	9.9	1298.0	16.6	

	GRADP	BachPlusP
0	9.7291630170000012	20.750948409999999
1	11.8	25.4
2	9.1	19.9
3	10.9	26.7
4	14.7	31.4

Using individual dataframe – slow

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

[29]:
```
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]  143
```

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

```
(814013, 21)
```

[30]:

	TRACT	POP16	LT50P	GT100P	GT150P	GT200P	MEDHINC	PovPerc	YEAR	STATE	…	BACHP	GRADP	BachPlusP
0	1001020100	1396	14.699999999999999	18.021468290000001	1.8803097219999998	5.9569049620000003	60255	9.0918167759999999	2008	Alabama	…	11.050633270000001	9.7291630170000012	20.750948409999999
1	1001020200	1516	17.300000000000001	13.474851320000001	0.29874091199999997	1.5685701780000001	34570	12.96785751	2008	Alabama	…	14.157830929999999	7.590958616	21.930149579999998
2	1001020300	2549	21.800000000000001	11.4979377	3.6633025469999998	0.45844473399999996	37101	6.914585679	2008					
3	1001020400	3638	15.6	13.65610053	3.4820126600000001	1.3284576959999999	48153	5.4389412429999995	2008					
4	1001020500	6948	12.5	18.500227150000001	5.3617976160000005	0.97059919500000003	58256	5.3786507590000001	2008					

```
2  Alabama  …          11.32799376          1.362691852  12.643148160000001
3  Alabama  …  13.756875389999999  6.8137658339999998  20.713600530000001
4  Alabama  …          21.31521575  9.9805559810000002  31.834601150000001

   TOTPOP    WHIT    BLCK  AMIND   ASIA  HAWA  LATIN
0  1852.0  1552.0   291.0   67.0    0.0   0.0   15.0
1  2045.0   855.0  1128.0    0.0   22.0   0.0    6.0
2  3443.0  2891.0   539.0    0.0   31.0   0.0   39.0
3  4639.0  4486.0    85.0   22.0   14.0   0.0  128.0
4  9339.0  8067.0  1131.0   88.0  146.0   0.0  471.0

[5 rows x 21 columns]
```