Week #4 - Python Practice

Where we left off last session was with having read from and written to a simple SQLite3 RDB. This DB, 'xyz.db,' started out with having one table of XYZ customer purchase transaction records in it, 'xyztrans.' You wrote a second table to this DB with customer information called 'xyzcust.'

It turns out that XYZ has purchased some “enhancement data” on its customers from Experian. Enhancement data are often used by retailers, direct marketers, and others to better understand their customers and for marketing campaign targeting. In what follows we're going to read XYZ's enhancement data and merge it with the XYZ customer data you've previously worked with.

So, let's launch a Canopy IPython session, and import into it pandas (as pd), numpy (as np), and SQLAlchemy. Import from pandas DataFrame and Series. Finally, read into your session the customer information data that's in the xyzcust table in the SQLite DB xyz.db. (You did save it, right?) Put it into a DataFrame called xyzcust.

XYZ's Experian data are in a json (javascript object notation) file called zdata.json. This file is available on Canvas. Put it it your default working directory to make your life easier.

Here's what the first customer record looks like in this zdata.json file:

'{"0":
{"ACCTNO":"PHSWPGWPQ","ZCREDIT":"Y","ZCRAFTS":"U","ZGOURMET":"U","ZCOMPU
TR":"Y","ZHITECH":"Y","ZONLINE":"Y","ZSPENDER":"U","ZGOLFERS":"U","ZGOLF
ERP":5.0,"ZDONORS":"U","ZDONORSP":2.0,"ZPETS":"U","ZPETSP":8.0,"ZARTS":"
U","ZARTSP":6.0,"ZMOB":"Y","ZMOBP":0.0,"ZFITNESS":"U","ZFITNESP":5.0,"ZO
UTDOOR":"U","ZOUTDOOP":5.0,"ZTRAVANY":"U","ZTRAVANP":7.0,"ZINVESTR":"U",
"ZINVESTP":5.0,"ZAUTOOWN":"U","ZAUTOOWP":6.0,"ZGARDEN":"U","ZGARDENP":3.
0,"ZCOLLECT":"U","ZCOLLECP":7.0,"ZCRUISE":"U","ZCRUISEP":7.0,"ZSPORTS":"
Y","ZSPORTSP":0.0,"ZSWEEPS":"U","ZSWEEPSP":5.0,"ZPOLITIC":"U","ZPOLITIP"
:2.0,"ZMUSIC":"U","ZMUSICP":8.0,"ZREAD":"U","ZREADP":4.0,"ZCHLDPRD":"Y",
"ZCHLDPRP":0.0,"ZDIY":"U","ZDIYP":5.0,"ZSELFIMP":"U","ZSELFIPP":8.0,"ZRE
LIGON":"U","ZRELIGOP":4.0,"ZGRANDPR":"U","ZGRANDPP":4.0,"ZCLOTHNG":"Y","
ZCLOTHNP":0.0,"ZDONENVR":"U","ZDONENVP":6.0,"ZMUTUAL":"U","ZMUTUALP":6.0
,"ZWGHTCON":"U","ZWGHTCOP":7.0,"ZPRCHPHN":"U","ZPRCHPHP":8.0,"ZPRCHTV":"
U","ZPRCHTVP":8.0,"ZMOBMULT":"Y","ZMOBMULP":0.0,"ZCREDPLP":6.0,"ZCREDPLT
":"U","ZDOGS":"U","ZDOGSP":7.0,"ZCATS":"U","ZCATSP":7.0,"ZHEALTH":"U","Z
HEALTHP":3.0,"ZAUTOINT":"U","ZAUTOINP":8.0,"ZSKIING":"U","ZSKIINGP":8.0,
"ZASTRLGY":"U","ZASTRLGP":9.0,"ZBOATS":"U","ZBOATSP":7.0,"ZCELL":"U","ZC
ELLP":8.0,"ZCOMMCON":"U","ZCOMMCOP":8.0,"ZHMDECOR":"Y","ZHMDECOP":0.0,"Z

ELEP":0.0,"ZCOHACON":"U","ZCOHACOP":0.0,"ZHADLCOR":"Y","ZHADLCOP":0.0,"Z
HOMEENT":"U","ZHOMEENP":8.0,"ZKITCHEN":"U","ZKITCHEP":1.0,"ZMOBAV":"U","
ZMOBAVP":9.0,"ZMOBBOOK":"Y","ZMOBBOOP":0.0,"ZMOBCLTH":"Y","ZMOBCLTP":0.0
,"ZMOBFIN":"U","ZMOBFINP":5.0,"ZMOBGIFT":"U","ZMOBGIFP":6.0,"ZMOBGRDN":"
U","ZMOBGRDP":2.0,"ZMOBJWL":"U","ZMOBJWLP":8.0,"ZMUSCLAS":"U","ZMUSCLAP"
:6.0,"ZMUSCNTR":"U","ZMUSCNTP":5.0,"ZMUSCRST":"U","ZMUSCRSP":6.0,"ZMUSOL
DI":"U","ZMUSOLDP":7.0,"ZMUSROCK":"U","ZMUSROCP":9.0,"ZPBCARE":"U","ZPBC
AREP":5.0,"ZPHOTO":"U","ZPHOTOP":8.0,"ZPRCHONL":"Y","ZPRCHONP":0.0,"ZTEN
NIS":"U","ZTENNISP":9.0,"ZTRAVDOM":"U","ZTRAVDOP":6.0,"ZTRAVFOR":"U","ZT
RAVFOP":6.0,"ZVOLUNTR":"U","ZVOLUNTP":6.0}}','ZMUSCLAS":"U","ZMUSCLAP":6
.0,"ZMUSCNTR":"U","ZMUSCNTP":5.0,"ZMUSCRST":"U","ZMUSCRSP":6.0,"ZMUSOLDI
":"U","ZMUSOLDP":7.0,"ZMUSROCK":"U","ZMUSROCP":9.0,"ZPBCARE":"U","ZPBCAR
EP":5.0,"ZPHOTO":"U","ZPHOTOP":8.0,"ZPRCHONL":"Y","ZPRCHONP":0.0,"ZTENNI
S":"U","ZTENNISP":9.0,"ZTRAVDOM":"U","ZTRAVDOP":6.0,"ZTRAVFOR":"U","ZTRA
VFOP":6.0,"ZVOLUNTR":"U","ZVOLUNTP":6.0}}'

You can see (you might need to look closely) that this record consists of key value pairs enclosed with a pair of curly brackets, {}, enclosing what's essentially like a Python dict. That first “0” (zero) enclosed by the outer curlies is a record key or index value. The next record in the file has a “1,” the one after that “2,” and so on. Inside the inner pair of curly brackets are key value pairs. Each key and its value are seperated by a colon, “:”. The pairs are comma-separated. The keys in this file are Experian variable names (with the exception of the record key; they all start with “Z”), and the values are the data values on those variables. You'll note that some variables are character type, and others are numeric. (You can tell here by what's enclosed in double quotes. No quotes, numeric.)

On Canvas you'll find a couple of items that provide some information about this data. There's a spreadsheet named Experian-Z-variables.key.xls that defines what each variable is. The file Experian-Data-Dictionary.pdf defines the codes used for the variables, but you'll have to dig around in a bit to find the Z variables. (Oh, those XYZ data people!)

First, let's retrieve the table xyzcust from xyz.db, the SQLite3 RDB that you saved it into. If you don't remember how to do this, go back and review how to connect to this RDB in the Session 3 Practice by using SQLAlchemy's “create_engine” method. Then, once you've connected to this DB, you should be able to read the customer account records into a Pandas DataFrame by:

In [1]:
```
import os

import cPickle as pickle

import pandas as pd  # panda's nickname is pd

import numpy as np  # numpy as np

from pandas import DataFrame, Series     # for convenience
```

In [2]:
```
import sqlalchemy
from sqlalchemy import create_engine
from sqlalchemy import schema
```

In [3]:
```
engine=create_engine('sqlite:///xyz.db')
```

In [4]:
```
xyzcust=pd.read_sql_table('xyzcust',engine)
```

where “engine” is the connection you made to the RDB as in Session 3. It has type() sqlalchemy.engine.base.Engine.

You may recall that one of the columns in this table is ACCTNO. ACCTNO is an unique customer identifier that is assigned at the time that a customer becomes a customer of XYZ.

Now, let's get the Experian data. It's in the file zdata.json, which is available to you on Canvas. Put it in your working directory for easy reference, and then do:

In [5]:
```
zdata=pd.read_json('zdata.json',orient='index')
```

The setting of the “orient” parameter indicates that the json file has a “dict-like” organization: it's indexed row by data row where the rows in this case correspond to customers.

If you look at the columns of zdata you'll see that one of the columns is ACCTNO, the same unique identifier that's in xyzcust. Do you think that there are the same number of customers in these to files? Compare len(xyzcust) and len(zdata). The answer is “no!” There are more records in zdata than in xyzcust.

Are there any duplicated records in zdata? You checked for them in xyzcust in the last Session Practice. If there are duplicated records,

are they whole records, or just different records with the same ACCTNO? If you don't know how to look into these questions, review what was covered in Session 3.

You are now faced with an existential data science question: what about the extra customers in zdata that aren't in xyzcust? Are they real customers whose records in xyzcust are missing for some reason? We don't know at this point, but in a bit we'll set these “ghost customers” aside so they can be looked into them sometime.

Next we're going to join the customer records in xyzcust with their corresponding records in zdata to get a larger single data set that you'll then store for future use. It's possible that not every customer in xyzcust has a record in zdata. One way to find out is to merge the two DataFrames into a single Dataframe by doing an “inner join” kind of merge using ACCTNO. (You probably had already observed that ACCTNO has no duplicates in either xyzcust or zdata, right?)

Another way to determine whether there are any ACCTNO values that aren't in both DataFrames, you could of course look for values that don't exist in both Frames, for example:

In [6]:
```
zNotC=set(zdata.ACCTNO)-set(xyzcust.ACCTNO)
```

or

In [7]:
```
zNotC=set(zdata.ACCTNO).difference(set(xyzcust.ACCTNO))
```

will result in zNotC being a Python set of ACCTNO values that are in zdata but not in xyzcust. The complementary operation should return a null set, “set(),” if all the ACCTNO's in xyzdata are in zdata. Try this and see what you get.

Save this zNotC set to use later in selecting records from zdata.

So let's get on with some merging. You can do an inner, one-to-one join using Pandas as follows:

In [8]:
```
custZData=pd.merge(xyzcust,zdata,on='ACCTNO')
```

custZData should have the number of rows (customer records) in it that xyzcust has.

With Pandas you can do other kinds of joins, like outer joins many-to-one joins, and many-to-many joins. You can also join DataFrames on an

index by using a DataFrames's .join and .merge methods.

Let's go back and split out those "ghost customer" records in zdata so we can set them aside for consideration in the future. Someone may be talked into investigating them, as customers are usually too valuable to lose or misplace. We need to find them in zdata, of course. But we know what their ACCTNO's are. They're in the Python set we called zNotC. Let's extract them from zdata:

In [9]:
```
zGhostCust=zdata[zdata.ACCTNO.isin(list(zNotC))]    # Boolean select rows
zGhostCust
```

Out[9]:

	ACCTNO	ZARTS	ZARTSP	ZASTRLGP	ZASTRLGY	ZAUTOINP	ZAUTOINT
100	WWQYGALPH	U	5.0	8.0	U	7.0	U
10097	WYGADALDL	U	7.0	9.0	U	8.0	U
10141	PHHDLWWHA	U	4.0	8.0	U	6.0	U
10214	SYHQHGYPH	U	3.0	8.0	U	6.0	U
10308	GYQLPLQGY	U	4.0	8.0	U	7.0	U
10400	SWWSSPAWS	U	3.0	8.0	U	7.0	U
10475	PHLAGGAGY	Y	0.0	8.0	U	8.0	U
10476	GADQQSSWQ	U	3.0	8.0	U	7.0	U
10500	SALADQADP	U	1.0	9.0	U	8.0	U
10517	WAWLQDQWA	U	5.0	7.0	U	6.0	U
10538	WGSHPDPP	U	4.0	8.0	U	7.0	U
10596	WSHHAGGPA	U	5.0	9.0	U	8.0	U
10621	WPDHYWAPH	U	4.0	9.0	U	7.0	U
10632	PWDWGSQQW	U	1.0	9.0	U	8.0	U
10668	LLSGHPAQW	U	6.0	8.0	U	6.0	U
10678	AHGASWHHG	U	2.0	8.0	U	7.0	U
10739	WSAHAGAPW	U	2.0	9.0	U	7.0	U
10769	SPSWPPGPY	U	4.0	6.0	U	6.0	U
10802	ALLWASQAY	U	4.0	8.0	U	6.0	U
10816	SYWQDAHLY	U	4.0	4.0	U	7.0	U
10824	WGWQDHHHD	Y	0.0	8.0	U	0.0	Y

10863	WPSLDPDWD	U	3.0	9.0	U	9.0	U
10921	GDSWYQQP	U	1.0	8.0	U	8.0	U
10938	SGGLWWDHQ	U	1.0	8.0	U	7.0	U
10989	LDSSWDQAS	U	2.0	8.0	U	6.0	U
11020	AQHWWGYWL	U	1.0	9.0	U	8.0	U
11071	GHDDAHQWA	U	2.0	9.0	U	8.0	U
11089	ADWYQAQPD	U	3.0	3.0	U	0.0	Y
11112	GGSHAHPWQ	U	2.0	9.0	U	8.0	U
11158	GQGLHAGD	U	5.0	8.0	U	7.0	U
...	...	...	...	...	...	...	...
8649	SLGAAQLGG	U	4.0	8.0	U	7.0	U
8705	GQSHPSHY	U	3.0	8.0	U	7.0	U
8729	LHGAYWGAQ	Y	0.0	8.0	U	7.0	U
8742	AHPAGLLQ	U	1.0	9.0	U	0.0	Y
8858	SSDSGPQPP	U	5.0	6.0	U	8.0	U
8865	WLWHSSWSG	U	2.0	9.0	U	9.0	U
8884	AQPAPPYLD	U	2.0	8.0	U	8.0	U
8997	WWAPPSDQY	Y	0.0	9.0	U	8.0	U
9086	LALSSHPYW	U	4.0	8.0	U	6.0	U
9109	AGWSWDYWY	U	5.0	9.0	U	7.0	U
9161	LWDYPHWWL	U	6.0	6.0	U	4.0	U
9182	SHASSHHPW	U	4.0	9.0	U	8.0	U
9219	GYSLQGALH	Y	0.0	9.0	U	8.0	U
9350	WSQSPWAGG	U	5.0	8.0	U	8.0	U
9507	LYDPWSDAW	U	6.0	8.0	U	6.0	U
951	AQYQHYPWQ	U	2.0	9.0	U	8.0	U
9512	SWYHDQSP	U	3.0	8.0	U	8.0	U
9539	SGDPGGWGP	U	4.0	7.0	U	6.0	U
9541	WAPGPQQDL	U	3.0	8.0	U	7.0	U

9549	SGWYASHSL	U	4.0	8.0	U	8.0	U
9570	GAYSQPSQL	U	4.0	6.0	U	4.0	U
9632	GHHHSHDDW	U	4.0	8.0	U	6.0	U
9671	SSSPPLWHD	Y	0.0	7.0	U	9.0	U
9766	SASAWSLGW	Y	0.0	8.0	U	0.0	Y
9886	PLWAYQPPQ	U	5.0	8.0	U	6.0	U
989	SLLWSHAPW	U	2.0	8.0	U	8.0	U
9931	PWDWGQHAG	None	NaN	NaN	None	NaN	None
9946	SHSGWGPLH	U	3.0	8.0	U	7.0	U
9970	WLYALSDWP	U	1.0	9.0	U	0.0	Y
9971	SGDAHHLDL	U	2.0	7.0	U	8.0	U

601 rows × 132 columns

```
zGhostCust should be a DataFrame with as many rows as the set zNotC has
items.  Save custZData and zGhostCust in the format(s) of your choice
for possible use in a later Session.  You might want to poke custZData
into your RDB xyz.db.  Note that it can be useful to have made copies of
your RDB and other data files from time to time as backups.  SQLite3
DB's are usually small enough for this to be convenient.  Pickles can be
little or big.

Here's a last question.  Suppose we wanted to extract from zdata all the
records that do have a match on ACCTNO in xyzcust.  We already have
them, of course, but is there a complement to the Boolean row selection
above that would get them directly out of zdata?  (Hint: Remember what
the twiddle, “~”, can be used for in Pandas methods calls?)
```

Deliverable :

What is the total number of the Ghost Customers in the data? Write and execute Python code snippet to get the total number of the Ghost Customers

In [10]:
```
ghostCustomers = len(zGhostCust.index)
print ghostCustomers
```

```
601
```

In [ ]:
```

```