Siddikov_Exercise_1

July 26, 2019

1 Deliverables:

- Submit a single zip-compressed file that has the name: YourLastName_Exercise_3 that has the following files:

1. Your **PDF document** that has your Source code and output
2. Your **ipynb script** that has your Source code and output

2 Objectives:

In this exercise, you will:

- Perform data analysis tasks on data read from a CSV file and loaded into a DataFrame object
- Use sqlalchemy to load data stored in a DatFrame object into sqlite database engine
- Use sqlalchemy to connect to sqlite database engine to execute SQL queries

Formatting Python Code When programming in Python, refer to Kenneth Reitz' PEP 8: The Style Guide for Python Code: http://pep8.org/ (Links to an external site.)Links to an external site. There is the Google style guide for Python at https://google.github.io/styleguide/pyguide.html (Links to an external site.)Links to an external site. Comment often and in detail.

[1]:
```
import os

import  pickle

import pandas as pd  # panda's nickname is pd

import numpy as np  # numpy as np

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

[2]:
```
### https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/
### Execute the code line by line in jupyter-notebook
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

[3]:
```
# read in the file
xyzcust10=pd.read_csv('xyzcust10.csv')
```

[4]:
```
# what are the data types
(xyzcust10).dtypes
```

[4]:
```
ACCTNO                      object
ZIP                          int64
ZIP4                         int64
LTD_SALES                  float64
LTD_TRANSACTIONS             int64
YTD_SALES_2009             float64
YTD_TRANSACTIONS_2009        int64
CHANNEL_ACQUISITION         object
BUYER_STATUS                object
ZIP9_Supercode               int64
ZIP9_SUPERCODE               int64
dtype: object
```

[5]:
```
type(xyzcust10)
```

[5]:
```
pandas.core.frame.DataFrame
```

[6]:
```
# writing out the file as a pickle file
pickle.dump(xyzcust10,open('xyzcust10.p','wb'))
```

[7]:
```
# Lecture Video:  read back in pickle file and make 2 copies
xyzcust10=pickle.load(open('xyzcust10.p','rb'))

xyzcust10red = xyzcust10.copy() # by default makes a deep copy

xyzcust10rev1=xyzcust10.copy() # by default makes a deep copy
```

The above assumes that xyzcust10.p is in your default directory. Otherwise, you'll need to include a path specification, of course.

xyzcust10 should be a pandas DataFrame:

[8]:
```
type(xyzcust10)
```

[8]:
```
pandas.core.frame.DataFrame
```

[9]:
```
xyzcust10.head()
```

[9]:
```
      ACCTNO    ZIP  ZIP4  LTD_SALES  LTD_TRANSACTIONS  YTD_SALES_2009  \
0  WDQQLLDQL  60084  5016       90.0                 1             0.0
1  WQWAYHYLA  60091  1750     4227.0                 9          1263.0
2  GSHAPLHAW  60067   900      420.0                 3           129.0
3  PGGYDYWAD  60068  3838     6552.0                 6             0.0
4  LWPSGPLLS  60090  3932      189.0                 3            72.0

   YTD_TRANSACTIONS_2009 CHANNEL_ACQUISITION BUYER_STATUS  ZIP9_Supercode  \
0                      0                  IB     INACTIVE       600845016
1                      3                  RT       ACTIVE       600911750
2                      1                  RT       ACTIVE       600670900
3                      0                  RT     INACTIVE       600683838
```

```
4                       1                  RT       ACTIVE       600903932

   ZIP9_SUPERCODE
0        600845016
1        600911750
2        600670900
3        600683838
4        600903932
```

xyzcust10 appears to have two nine-digit ZIP supercode columns with slightly different column labels or names. To see them, try entering xyzcust10.columns or xyzcust10.dtypes at the command prompt. Are the values in these two columns the same? **Yes, they are the same.** If so, we can get rid of one of them. There are different ways we can figure out whether they are the same, but a simple way is to test each pair of values to see if they are equal or not, and then to total up the results, the number of equal pairs or not equal pairs:

[10]:

```
# Lecture Video: Look at columns
xyzcust10.columns

# Lecture Video: Look at file attribution
xyzcust10.info()
```

[10]:

```
Index(['ACCTNO', 'ZIP', 'ZIP4', 'LTD_SALES', 'LTD_TRANSACTIONS',
       'YTD_SALES_2009', 'YTD_TRANSACTIONS_2009', 'CHANNEL_ACQUISITION',
       'BUYER_STATUS', 'ZIP9_Supercode', 'ZIP9_SUPERCODE'],
      dtype='object')


<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30471 entries, 0 to 30470
Data columns (total 11 columns):
ACCTNO                    30471 non-null object
ZIP                       30471 non-null int64
ZIP4                      30471 non-null int64
LTD_SALES                 30471 non-null float64
LTD_TRANSACTIONS          30471 non-null int64
YTD_SALES_2009            30471 non-null float64
YTD_TRANSACTIONS_2009     30471 non-null int64
CHANNEL_ACQUISITION       30471 non-null object
BUYER_STATUS              30471 non-null object
ZIP9_Supercode            30471 non-null int64
ZIP9_SUPERCODE            30471 non-null int64
dtypes: float64(2), int64(6), object(3)
memory usage: 2.6+ MB
```

[11]:

```
# are the two zip code columns exactly the same
# Lecture Video: summing up the number of records that do not equal
# so they are exactly the same, they are duplicates
(xyzcust10.ZIP9_Supercode!=xyzcust10.ZIP9_SUPERCODE).sum()
```

[11]:

```
0
```

which will return zero if the values in the two columns are the same. What result do you get? **I got zero result. So, the two Series are identical.**

Note that what's going on here is that what's in the parentheses is a logical test of inequality between the two columns of the DataFrame (which are also pandas Series objects), which results in a Series of true or false Boolean values. The post-pended .sum() function adds up over the Series by treating the Trues as 1's, and the Falses as 0's. So if the result is zero, the two Series are identical, except for their names, of course.

We could have also expressed the logical comparison in the parens as ((xyzcust10.ZIP9$_Supercode == xyzcust10.ZIP9_SUPERCODE)$)

to get the same result, since the the twidddle, the , works in some pandas contexts as not. What kind of result do you think you'd get with the following variation:

(xyzcust10.ZIP9$_Supercode == xyzcust10.ZIP9_SUPERCODE).sum()$

Why might it be different?

Note that we could have referred to the columns differently, for example:

xyzcust10['ZIP9$_Supercode'$]

Columns in DataFrames can be referred to in different ways. We'll see more of them going forward.

[12]:

```
# Lecture Video: will show values of field
xyzcust10['ZIP9_Supercode']
```

[12]:

```
0         600845016
1         600911750
2         600670900
3         600683838
4         600903932
5         600858670
6         600913447
7         600911613
8         600683668
9         600911759
10        600818325
11        600562960
12        600912813
13        600673528
14        600603209
15        600891326
16        600692129
17        600911453
18        600682219
19        600624628
20        600912346
21        600614527
22        600612123
23        600894622
24        600626077
25        600818248
```

```
26       600932706
27       600623210
28       600933840
29       600905705
           ...
30441    600987410
30442    600987615
30443    600988020
30444    600988426
30445    600988550
30446    600987893
30447    600987977
30448    600987805
30449    600988014
30450    600988671
30451    600988128
30452    600988760
30453    600988093
30454    600987108
30455    600987552
30456        60098
30457    600989172
30458    600988958
30459    600989029
30460    600987869
30461    600982556
30462    600980142
30463    600982857
30464    600983342
30465    600987858
30466    600983951
30467    600989681
30468    600983858
30469    600987927
30470    600984160
Name: ZIP9_Supercode, Length: 30471, dtype: int64
```

So, Oops! Someone included the same column in the data twice, but with slightly different names. Why waste the space? Why risk confusion? Let's get rid of one of them:

We could do:

[13]:
```python
# one way to delete a column
del xyzcust10['ZIP9_Supercode']
del xyzcust10rev1['ZIP9_Supercode']
```

or

[14]:
```python
# Lecture Video: another way to delete column
# axis = 1 specifies a column
# inplace = True means that data specified is changed
# inplace = False means that a copy of the object is returned
xyzcust10red.drop('ZIP9_Supercode',axis=1,inplace=True)
```

[15]:
```
# Lecture Video: see if the column is gone
xyzcust10.columns
```

[15]:
```
Index(['ACCTNO', 'ZIP', 'ZIP4', 'LTD_SALES', 'LTD_TRANSACTIONS',
       'YTD_SALES_2009', 'YTD_TRANSACTIONS_2009', 'CHANNEL_ACQUISITION',
       'BUYER_STATUS', 'ZIP9_SUPERCODE'],
      dtype='object')
```

Next we're going to shift gears and gobble up some transaction data for XYZ's customers. They are in a table in a SQLite3 relational database (RDB) file that's called xyz.db. This file is available to you on Canvas. At this point you might want to pickle xyzcust10rev1 in case you need to end your session and start again later. Remember that things in a Python session are not permanent.

To make things simple you'll want to put the xyz.db file in a place where you can find it easily from in Canopy. Your default directory would be a good bet. Remember what it is? See what os.getcwd() tells you.

[16]:
```
os.getcwd()
```

[16]:
```
'C:\\Users\\asidd\\Desktop\\MSDS\\420 Database Systems\\Lecture 5\\Exercise 3
Files Version b\\Exercise 3 Files Version b'
```

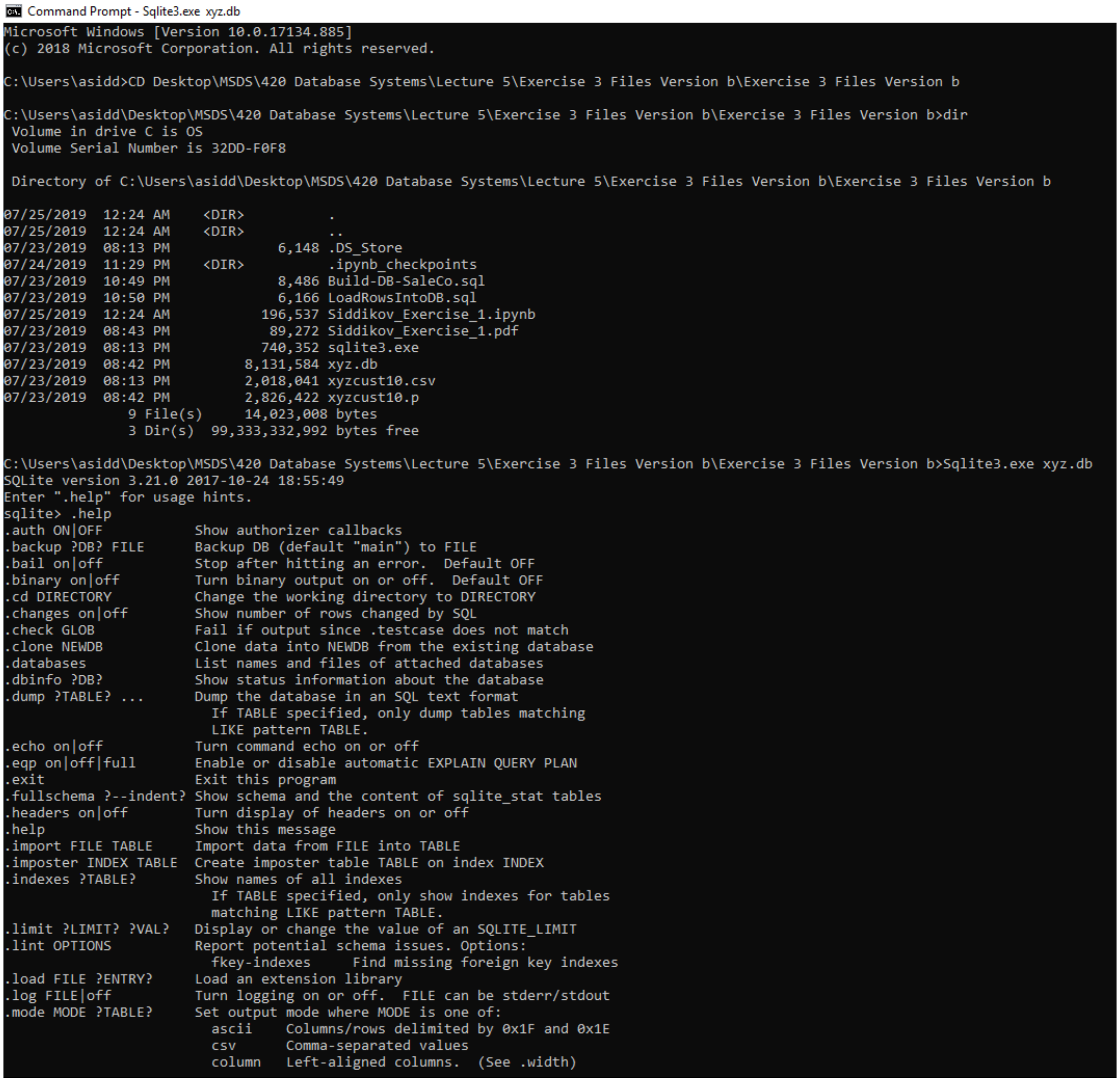
If you installed the sqlite3 client, you can take a look at this database (DB) using it and without using Python. sqlSQLite3 is a very simple and easy to use RDB, and it doesn't require a server. Assuming that you've installed it and that you're in the directory were you put xyztrans.db, using the command from your OS command prompt:

c:$v203 > sqlite3xyz.dbSQLiteversion3.8.8.32015 - 02 - 2513 : 29 :$
$11Enter''.help''forusagehints.sqlite >$

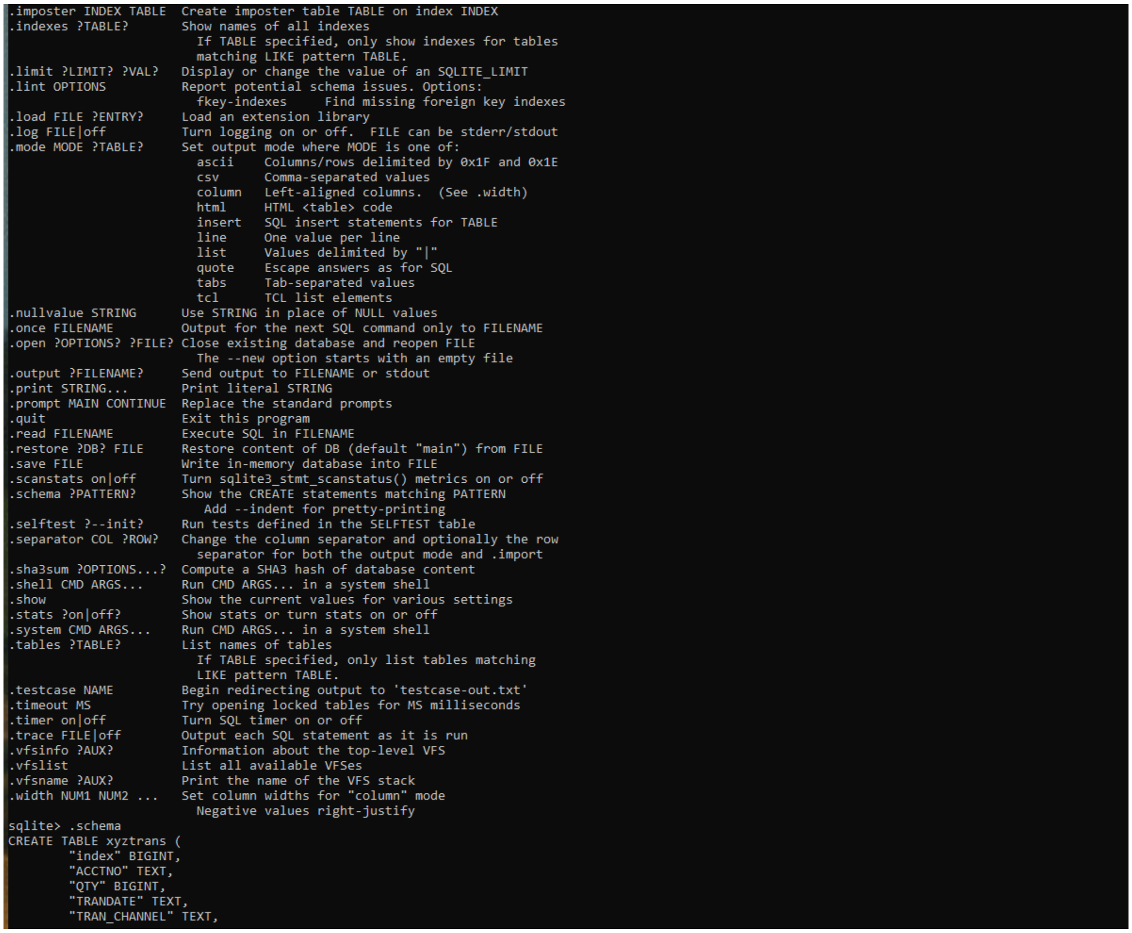
will start sqlite3 and open the db file. You can see the tables in this db with the sqlite3 command .tables . (That's a period, . before tables. Help in sqlSQLite3 is .help .)

sqlite> .tables xyztrans sqlite>

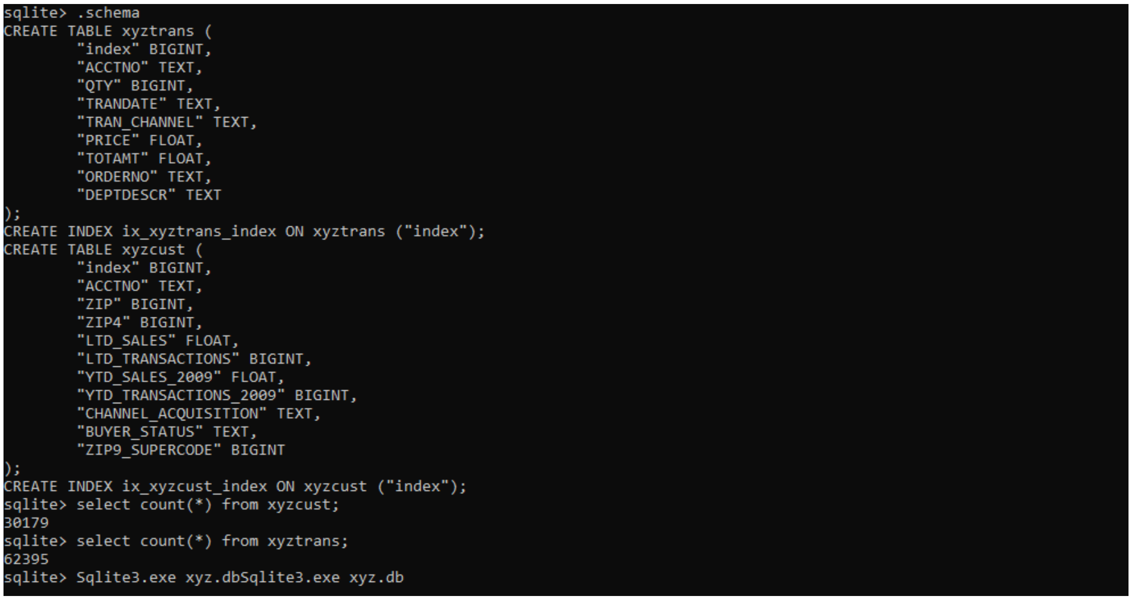
[17]:
```
### Optional ###
from IPython.display import Image
from IPython.display import display
x = Image(filename=r'C:\Users\asidd\Downloads\msds_420_cmd1.PNG')
y = Image(filename=r'C:\Users\asidd\Downloads\msds_420_cmd2.PNG')
z = Image(filename=r'C:\Users\asidd\Downloads\msds_420_cmd3.PNG')
display(x, y, z)
```



```
Microsoft Windows [Version 10.0.17134.885]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\asidd>CD Desktop\MSDS\420 Database Systems\Lecture 5\Exercise 3 Files Version b\Exercise 3 Files Version b

C:\Users\asidd\Desktop\MSDS\420 Database Systems\Lecture 5\Exercise 3 Files Version b\Exercise 3 Files Version b>dir
 Volume in drive C is OS
 Volume Serial Number is 32DD-F0F8

 Directory of C:\Users\asidd\Desktop\MSDS\420 Database Systems\Lecture 5\Exercise 3 Files Version b\Exercise 3 Files Version b

07/25/2019  12:24 AM    <DIR>          .
07/25/2019  12:24 AM    <DIR>          ..
07/23/2019  08:13 PM             6,148 .DS_Store
07/24/2019  11:29 PM    <DIR>          .ipynb_checkpoints
07/23/2019  10:49 PM             8,486 Build-DB-SaleCo.sql
07/23/2019  10:50 PM             6,166 LoadRowsIntoDB.sql
07/25/2019  12:24 AM           196,537 Siddikov_Exercise_1.ipynb
07/23/2019  08:43 PM            89,272 Siddikov_Exercise_1.pdf
07/23/2019  08:13 PM           740,352 sqlite3.exe
07/23/2019  08:42 PM         8,131,584 xyz.db
07/23/2019  08:13 PM         2,018,041 xyzcust10.csv
07/23/2019  08:42 PM         2,826,422 xyzcust10.p
               9 File(s)     14,023,008 bytes
               3 Dir(s)  99,333,332,992 bytes free

C:\Users\asidd\Desktop\MSDS\420 Database Systems\Lecture 5\Exercise 3 Files Version b\Exercise 3 Files Version b>Sqlite3.exe xyz.db
SQLite version 3.21.0 2017-10-24 18:55:49
Enter ".help" for usage hints.
sqlite> .help
.auth ON|OFF           Show authorizer callbacks
.backup ?DB? FILE      Backup DB (default "main") to FILE
.bail on|off           Stop after hitting an error.  Default OFF
.binary on|off         Turn binary output on or off.  Default OFF
.cd DIRECTORY          Change the working directory to DIRECTORY
.changes on|off        Show number of rows changed by SQL
.check GLOB            Fail if output since .testcase does not match
.clone NEWDB           Clone data into NEWDB from the existing database
.databases             List names and files of attached databases
.dbinfo ?DB?           Show status information about the database
.dump ?TABLE? ...      Dump the database in an SQL text format
                         If TABLE specified, only dump tables matching
                         LIKE pattern TABLE.
.echo on|off           Turn command echo on or off
.eqp on|off|full       Enable or disable automatic EXPLAIN QUERY PLAN
.exit                  Exit this program
.fullschema ?--indent? Show schema and the content of sqlite_stat tables
.headers on|off        Turn display of headers on or off
.help                  Show this message
.import FILE TABLE     Import data from FILE into TABLE
.imposter INDEX TABLE  Create imposter table TABLE on index INDEX
.indexes ?TABLE?       Show names of all indexes
                         If TABLE specified, only show indexes for tables
                         matching LIKE pattern TABLE.
.limit ?LIMIT? ?VAL?   Display or change the value of an SQLITE_LIMIT
.lint OPTIONS          Report potential schema issues. Options:
                         fkey-indexes     Find missing foreign key indexes
.load FILE ?ENTRY?     Load an extension library
.log FILE|off          Turn logging on or off.  FILE can be stderr/stdout
.mode MODE ?TABLE?     Set output mode where MODE is one of:
                         ascii    Columns/rows delimited by 0x1F and 0x1E
                         csv      Comma-separated values
                         column   Left-aligned columns.  (See .width)
```



```
.imposter INDEX TABLE  Create imposter table TABLE on index INDEX
.indexes ?TABLE?       Show names of all indexes
                         If TABLE specified, only show indexes for tables
                         matching LIKE pattern TABLE.
.limit ?LIMIT? ?VAL?   Display or change the value of an SQLITE_LIMIT
.lint OPTIONS          Report potential schema issues. Options:
                         fkey-indexes     Find missing foreign key indexes
.load FILE ?ENTRY?     Load an extension library
.log FILE|off          Turn logging on or off.  FILE can be stderr/stdout
.mode MODE ?TABLE?     Set output mode where MODE is one of:
                         ascii    Columns/rows delimited by 0x1F and 0x1E
                         csv      Comma-separated values
                         column   Left-aligned columns.  (See .width)
                         html     HTML <table> code
                         insert   SQL insert statements for TABLE
                         line     One value per line
                         list     Values delimited by "|"
                         quote    Escape answers as for SQL
                         tabs     Tab-separated values
                         tcl      TCL list elements
.nullvalue STRING      Use STRING in place of NULL values
.once FILENAME         Output for the next SQL command only to FILENAME
.open ?OPTIONS? ?FILE? Close existing database and reopen FILE
                         The --new option starts with an empty file
.output ?FILENAME?     Send output to FILENAME or stdout
.print STRING...       Print literal STRING
.prompt MAIN CONTINUE  Replace the standard prompts
.quit                  Exit this program
.read FILENAME         Execute SQL in FILENAME
.restore ?DB? FILE     Restore content of DB (default "main") from FILE
.save FILE             Write in-memory database into FILE
.scanstats on|off      Turn sqlite3_stmt_scanstatus() metrics on or off
.schema ?PATTERN?      Show the CREATE statements matching PATTERN
                          Add --indent for pretty-printing
.selftest ?--init?     Run tests defined in the SELFTEST table
.separator COL ?ROW?   Change the column separator and optionally the row
                         separator for both the output mode and .import
.sha3sum ?OPTIONS...?  Compute a SHA3 hash of database content
.shell CMD ARGS...     Run CMD ARGS... in a system shell
.show                  Show the current values for various settings
.stats ?on|off?        Show stats or turn stats on or off
.system CMD ARGS...    Run CMD ARGS... in a system shell
.tables ?TABLE?        List names of tables
                         If TABLE specified, only list tables matching
                         LIKE pattern TABLE.
.testcase NAME         Begin redirecting output to 'testcase-out.txt'
.timeout MS            Try opening locked tables for MS milliseconds
.timer on|off          Turn SQL timer on or off
.trace FILE|off        Output each SQL statement as it is run
.vfsinfo ?AUX?         Information about the top-level VFS
.vfslist               List all available VFSes
.vfsname ?AUX?         Print the name of the VFS stack
.width NUM1 NUM2 ...   Set column widths for "column" mode
                         Negative values right-justify
sqlite> .schema
CREATE TABLE xyztrans (
        "index" BIGINT,
        "ACCTNO" TEXT,
        "QTY" BIGINT,
        "TRANDATE" TEXT,
        "TRAN_CHANNEL" TEXT,
```



```
sqlite> .schema
CREATE TABLE xyztrans (
        "index" BIGINT,
        "ACCTNO" TEXT,
        "QTY" BIGINT,
        "TRANDATE" TEXT,
        "TRAN_CHANNEL" TEXT,
        "PRICE" FLOAT,
        "TOTAMT" FLOAT,
        "ORDERNO" TEXT,
        "DEPTDESCR" TEXT
);
CREATE INDEX ix_xyztrans_index ON xyztrans ("index");
CREATE TABLE xyzcust (
        "index" BIGINT,
        "ACCTNO" TEXT,
        "ZIP" BIGINT,
        "ZIP4" BIGINT,
        "LTD_SALES" FLOAT,
        "LTD_TRANSACTIONS" BIGINT,
        "YTD_SALES_2009" FLOAT,
        "YTD_TRANSACTIONS_2009" BIGINT,
        "CHANNEL_ACQUISITION" TEXT,
        "BUYER_STATUS" TEXT,
        "ZIP9_SUPERCODE" BIGINT
);
CREATE INDEX ix_xyzcust_index ON xyzcust ("index");
sqlite> select count(*) from xyzcust;
30179
sqlite> select count(*) from xyztrans;
62395
sqlite> Sqlite3.exe xyz.dbSqlite3.exe xyz.db
```

There are a couple of different ways to read and write data to RDBs using Python, but the most flexible and easiest may be by using what's in pandas. pandas will make use of the SQLAlchemy package, which is available for installation within Canopy. (Did you install it in Session 1?) SQLAlchemy provides a consistent interface with different RDBs, SQLite being one of them.

Let's get SQLAlchemy into our IPython session:

[18]:
```python
import sqlalchemy
```

Now if you do the sqlalchemy.<tab> trick from the command prompt, you'll be able to see SQLAlchemy's various (and many) attributes and functions.

To simplify things, let's get a function out of SQLAlchemy that we'll use to define the SQLite3 db we'll be working with:

[19]:
```python
from sqlalchemy import create_engine
```

Now let's specify the xyz db as the SQLite3 RDB we want to work with:

[20]:
```python
# specify  the database we will work with
engine=create_engine('sqlite:///xyz.db')
```

This assumes that you have xyz.db in your current working directory. There are different valid syntaxes, e.g.

sqlite:///:memory: (or, sqlite://) sqlite:///relative/path/to/file.db sqlite:////absolute/path/to/file.db

We used the second syntax, above. Be sure to use the correct number of slashes for the version you want to use. You need the enclosing single quotes, too. There's only one table in this RDB. It's called xyztrans. Let's read it into a DataFrame:

[21]:
```python
# read in one table
xyztrans=pd.read_sql('xyztrans', engine)
```

xyztrans is a DataFrame. This defaults to reading all records from the db. What columns have been read from the table xyztrans? Try:

[22]:
```python
xyztrans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62395 entries, 0 to 62394
Data columns (total 9 columns):
index           62395 non-null int64
ACCTNO          62395 non-null object
QTY             62395 non-null int64
TRANDATE        62395 non-null object
TRAN_CHANNEL    62395 non-null object
PRICE           62395 non-null float64
TOTAMT          62395 non-null float64
ORDERNO         62395 non-null object
DEPTDESCR       62395 non-null object
dtypes: float64(2), int64(2), object(5)
memory usage: 4.3+ MB
```

or

[23]:
```python
xyztrans.columns
```

[23]:
```
Index(['index', 'ACCTNO', 'QTY', 'TRANDATE', 'TRAN_CHANNEL', 'PRICE', 'TOTAMT',
       'ORDERNO', 'DEPTDESCR'],
      dtype='object')
```

This db has only one table in it. What if it had more than one, and you didn't know their names? How would you know? Well, one way is to read some metadata from it:

[24]:
```
# look at the database schema
from sqlalchemy import schema
```

[25]:
```
# retrieving schema for all of engine (xyz database)
xyzMetaData=schema.MetaData(bind=engine)
xyzMetaData.reflect()
```

[26]:
```
xyzMetaData.tables
```

[26]:
```
immutabledict({'xyzcust': Table('xyzcust',
MetaData(bind=Engine(sqlite:///xyz.db)), Column('index', BIGINT(),
table=<xyzcust>), Column('ACCTNO', TEXT(), table=<xyzcust>), Column('ZIP',
BIGINT(), table=<xyzcust>), Column('ZIP4', BIGINT(), table=<xyzcust>),
Column('LTD_SALES', FLOAT(), table=<xyzcust>), Column('LTD_TRANSACTIONS',
BIGINT(), table=<xyzcust>), Column('YTD_SALES_2009', FLOAT(), table=<xyzcust>),
Column('YTD_TRANSACTIONS_2009', BIGINT(), table=<xyzcust>),
Column('CHANNEL_ACQUISITION', TEXT(), table=<xyzcust>), Column('BUYER_STATUS',
TEXT(), table=<xyzcust>), Column('ZIP9_SUPERCODE', BIGINT(), table=<xyzcust>),
schema=None), 'xyztrans': Table('xyztrans',
MetaData(bind=Engine(sqlite:///xyz.db)), Column('index', BIGINT(),
table=<xyztrans>), Column('ACCTNO', TEXT(), table=<xyztrans>), Column('QTY',
BIGINT(), table=<xyztrans>), Column('TRANDATE', TEXT(), table=<xyztrans>),
Column('TRAN_CHANNEL', TEXT(), table=<xyztrans>), Column('PRICE', FLOAT(),
table=<xyztrans>), Column('TOTAMT', FLOAT(), table=<xyztrans>),
Column('ORDERNO', TEXT(), table=<xyztrans>), Column('DEPTDESCR', TEXT(),
table=<xyztrans>), schema=None)})
```

xyzMetaData.tables will be a dict that contains information about the db. Tables will be keys in this dict:

[27]:
```
#␣
 ↪----------------------------------------------------------------------------------
# if you have run this code before, you will have xyzcust table in your␣
 ↪database
# to remove the table, you can run the code below
#␣
 ↪----------------------------------------------------------------------------------
sql = ('DROP TABLE IF EXISTS xyzcust;')
result = engine.execute(sql)

# you should only be able to see the yxztrans table
xyzMetaData=schema.MetaData(bind=engine)
xyzMetaData.reflect()
xyzMetaData.tables
```

[27]:

```
immutabledict({'xyztrans': Table('xyztrans',
MetaData(bind=Engine(sqlite:///xyz.db)), Column('index', BIGINT(),
table=<xyztrans>), Column('ACCTNO', TEXT(), table=<xyztrans>), Column('QTY',
BIGINT(), table=<xyztrans>), Column('TRANDATE', TEXT(), table=<xyztrans>),
Column('TRAN_CHANNEL', TEXT(), table=<xyztrans>), Column('PRICE', FLOAT(),
table=<xyztrans>), Column('TOTAMT', FLOAT(), table=<xyztrans>),
Column('ORDERNO', TEXT(), table=<xyztrans>), Column('DEPTDESCR', TEXT(),
table=<xyztrans>), schema=None)})
```

[28]:

```
xyzMetaData.tables.keys()
```

[28]:

```
dict_keys(['xyztrans'])
```

At this point there’s only one table name, ’xyztrans, in xyz.db. You’ll see another method for inspecting DB’s below.

We’re going to write the xyz customer records into a new table in the sqlite3 RDB, but before we do that let’s make sure that the records are unique, that is, that no customer has more than one record. We can do this with some pandas DataFrame methods. Using the customer DataFrame xyzcust10rev1

[29]:

```
# lets check for row duplicates
xyzcust10rev1.duplicated().sum()
```

[29]:

```
292
```

will return a zero if all records are unique, or the number of rows in xyzcust10rev1 that are duplicates. The reason is that the duplicated() method for the DataFrame returns a Series of Trues and Falses, a Boolean Series. Summing over the Series forces the values to be cast as numeric.

Oops. There are some duplicates. How many duplicates do you find in xyzcust10rev1? To rid a DataFrame of unduplicated rows,

[30]:

```
# drop duplicates and then make sure they are gone
xyzcustUnDup=xyzcust10rev1.drop_duplicates()

xyzcustUnDup.duplicated().sum()
```

[30]:

```
0
```

How many unique customer records do you now have? By the way, note that you could have limited your examination to just one or more columns, for example just ACCTNO, customer account number, by providing ACCTNO as an argument or by using it to define a Series:

[31]:

```
xyzcust10rev1.duplicated('ACCTNO').sum()
```

[31]:

```
292
```

[32]:

```
xyzcust10rev1.ACCTNO.duplicated().sum()
```

[32]:

```
292
```

When there are duplicates of a record, which of them do you think .drop$_{d}uplicates()retains$?

Now that we’ve checked for, and have removed, duplicate customer records, from the customer records, let’s write them into a new table in xyztrans.db.

[33]:

```
# add the data from the csv file into a new table in xyz database
xyzcustUnDup.to_sql('xyzcust', engine, chunksize = 1)
```

Did it create the table in xyz.db? Check:

[34]:
```
pd.read_sql_table('xyzcust', engine).columns
```

[34]:
```
Index(['index', 'ACCTNO', 'ZIP', 'ZIP4', 'LTD_SALES', 'LTD_TRANSACTIONS',
       'YTD_SALES_2009', 'YTD_TRANSACTIONS_2009', 'CHANNEL_ACQUISITION',
       'BUYER_STATUS', 'ZIP9_SUPERCODE'],
      dtype='object')
```

[35]:
```
xyztest = pd.read_sql('xyzcust', engine)

# does xyzcust have the correct number of records after deleting duplicates?
# 30,471 original records - 292 duplicates = 30179
xyztest.count()
```

[35]:
```
index                    30179
ACCTNO                   30179
ZIP                      30179
ZIP4                     30179
LTD_SALES                30179
LTD_TRANSACTIONS         30179
YTD_SALES_2009           30179
YTD_TRANSACTIONS_2009    30179
CHANNEL_ACQUISITION      30179
BUYER_STATUS             30179
ZIP9_SUPERCODE           30179
dtype: int64
```

should produce the columns of the DataFrame you wrote to the db. Remember that engine refers to the SQLite3 DB by way of defining the connection using SQLAlchemy's create$_{e}nginemethod$.

How many tables are there now in xyz.db? And, what are their names?

[36]:
```
xyzMetaData=schema.MetaData(bind=engine)
xyzMetaData.reflect()
xyzMetaData.tables
```

[36]:
```
immutabledict({'xyzcust': Table('xyzcust',
MetaData(bind=Engine(sqlite:///xyz.db)), Column('index', BIGINT(),
table=<xyzcust>), Column('ACCTNO', TEXT(), table=<xyzcust>), Column('ZIP',
BIGINT(), table=<xyzcust>), Column('ZIP4', BIGINT(), table=<xyzcust>),
Column('LTD_SALES', FLOAT(), table=<xyzcust>), Column('LTD_TRANSACTIONS',
BIGINT(), table=<xyzcust>), Column('YTD_SALES_2009', FLOAT(), table=<xyzcust>),
Column('YTD_TRANSACTIONS_2009', BIGINT(), table=<xyzcust>),
Column('CHANNEL_ACQUISITION', TEXT(), table=<xyzcust>), Column('BUYER_STATUS',
TEXT(), table=<xyzcust>), Column('ZIP9_SUPERCODE', BIGINT(), table=<xyzcust>),
schema=None), 'xyztrans': Table('xyztrans',
MetaData(bind=Engine(sqlite:///xyz.db)), Column('index', BIGINT(),
table=<xyztrans>), Column('ACCTNO', TEXT(), table=<xyztrans>), Column('QTY',
BIGINT(), table=<xyztrans>), Column('TRANDATE', TEXT(), table=<xyztrans>),
Column('TRAN_CHANNEL', TEXT(), table=<xyztrans>), Column('PRICE', FLOAT(),
table=<xyztrans>), Column('TOTAMT', FLOAT(), table=<xyztrans>),
Column('ORDERNO', TEXT(), table=<xyztrans>), Column('DEPTDESCR', TEXT(),
table=<xyztrans>), schema=None)})
```

```
[37]: xyzMetaData.tables.keys()
```

```
[37]: dict_keys(['xyzcust', 'xyztrans'])
```

Another way to look at the metadata of an RDB using SQLAlchemy is by using the inspect method:

```
[38]: xyzMetaData
```

```
[38]: MetaData(bind=Engine(sqlite:///xyz.db))
```

```
[39]: from sqlalchemy import inspect
```

```
[40]: insp=inspect(engine)
```

```
[41]:  # we have two tables in our database
         insp.get_table_names()
```

```
[41]: ['xyzcust', 'xyztrans']
```

Do you think there are any duplicates in the order transaction data? If so, what would you make of them? You can use SQLAlchemy to query a DB so as to import selected records from an RDB. You can also append records to existing tables in an RDB, create various kinds of DB indexes, and pretty much do everything you would do using standard SQL while interacting with an RDB using a client for it. As a query example, suppose we wanted to select from the xyz tranaction data in the xyztrans.db all transactions made in XYZ's retail stores. These are coded as RT in the table's TRAN$_CHANNEL.Wecoulddo$:

```
[42]: rttrans=pd.read_sql_query("SELECT * \
                                  FROM xyztrans \
                                  WHERE TRAN_CHANNEL='RT'", engine)
```

A last point about SQLAlchemy: it has its own declarative language that provides means of interacting with DB's that is more object oriented than traditional SQL is. You can find lots of documentation about SQLAlchemy at http://www.sqlalchemy.org.

```
[43]: # look at RT data
      rttrans
```

```
[43]:        index    ACCTNO  QTY   TRANDATE TRAN_CHANNEL    PRICE   TOTAMT  \
      0          0    WGDQLA    1  09JUN2009           RT   599.85   599.85
      1          1    WGDQLA    1  09JUN2009           RT    39.00    39.00
      2          2    WGDQLA    1  28NOV2009           RT    15.00    15.00
      3          3    WGDQLA    1  28NOV2009           RT    69.00    69.00
      4          4    WGDQLA    1  28NOV2009           RT    84.00    84.00
      5          5    WGDQLA    1  28NOV2009           RT    69.00    69.00
      6          6    WGDQLA    1  28NOV2009           RT    89.85    89.85
      7          7    WGDQLA    1  28NOV2009           RT   119.85   119.85
      8          8    APSYYW    1  07JUN2009           RT    22.50    22.50
      9          9    APSYYW    1  07JUN2009           RT    44.85    44.85
      10        10    APSYYW    1  07JUN2009           RT    30.00    30.00
      11        11    APSYYW    1  07JUN2009           RT    30.00    30.00
      12        13    GGDWGY    1  14SEP2009           RT   239.85   239.85
```

```
13        14     GGDWGY    1  18DEC2009           RT   234.00   234.00
14        15     HHSSAL    1  13SEP2009           RT    66.00    66.00
15        16     HHSSAL    1  13SEP2009           RT    66.00    66.00
16        17     HHSSAL    1  13SEP2009           RT    38.25    38.25
17        18     HHSSAL    1  13SEP2009           RT    28.50    28.50
18        19     HHSSAL    1  13SEP2009           RT    43.50    43.50
19        20     HHSSAL    1  13SEP2009           RT    24.00    24.00
20        21     HHSSAL    1  13SEP2009           RT    42.00    42.00
21        22     HHSSAL    1  13SEP2009           RT    38.85    38.85
22        23     HHSSAL    1  13SEP2009           RT   105.00   105.00
23        24     HHSSAL    1  13SEP2009           RT    30.00    30.00
24        25     HHSSAL    1  13SEP2009           RT    32.85    32.85
25        26     HHSSAL    1  13SEP2009           RT    84.00    84.00
26        27     HHSSAL    1  18DEC2009           RT    28.50    28.50
27        28     HHSSAL    1  18DEC2009           RT    43.50    43.50
28        29     HHSSAL    1  18DEC2009           RT    27.00    27.00
29        30     HHSSAL    1  18DEC2009           RT    31.50    31.50
...      ...        ...  ...        ...          ...      ...      ...
53781  62350  GYLAPPYPQ    1  11OCT2009           RT    59.85    59.85
53782  62351  GYLAPPYPQ    1  11OCT2009           RT   126.00   126.00
53783  62352  GYLAPPYPQ    1  11OCT2009           RT    81.00    81.00
53784  62353  GYLAPPYPQ    1  11OCT2009           RT    36.00    36.00
53785  62354  GYLAPPYYW    1  10OCT2009           RT    31.50    31.50
53786  62355  GYLPADYQL    1  14OCT2009           RT    59.85    59.85
53787  62356  GYLPADYQL    1  14OCT2009           RT    36.00    36.00
53788  62357  GYLPADYQL    1  14OCT2009           RT    72.00    72.00
53789  62358  GYLPADYQL    1  14OCT2009           RT    72.00    72.00
53790  62359  GYLPADYQL    1  14OCT2009           RT    27.00    27.00
53791  62360  GYLPADYQL    1  14OCT2009           RT    48.00    48.00
53792  62361  GYLPADYQL    1  14OCT2009           RT    66.00    66.00
53793  62362  GYLPADYQL    1  14OCT2009           RT    57.00    57.00
53794  62364  GYLHWWQGW    1  21NOV2009           RT    36.00    36.00
53795  62365  GYLHWWQGW    1  21NOV2009           RT    30.00    30.00
53796  62366  GYLHWWQGW    1  21NOV2009           RT    28.50    28.50
53797  62367  GYLHWWQGW    1  21NOV2009           RT    54.00    54.00
53798  62368  GYLHWWQGW    1  21NOV2009           RT    28.50    28.50
53799  62369  GYLYSQQSG    1  27NOV2009           RT    27.00    27.00
53800  62370  GYLYSQQSG    1  27NOV2009           RT    45.00    45.00
53801  62371  GYLYSQQSG    1  27NOV2009           RT    74.85    74.85
53802  62372  GYLYSQQSG    1  21NOV2009           RT    62.64    62.64
53803  62373  GYLYSQQSG    1  21NOV2009           RT   299.85   299.85
53804  62374  GYLYSQQSG    1  29OCT2009           RT   299.85   299.85
53805  62375  GYLYSQQSG    1  14NOV2009           RT    32.85    32.85
53806  62376  GYLYSQQSG    1  14NOV2009           RT    45.00    45.00
53807  62377  GYLYSQQSG    1  14NOV2009           RT    15.00    15.00
53808  62378  GYLYSQQSG    1  29NOV2009           RT    42.00    42.00
53809  62379  GYLYSQQSG    1  29NOV2009           RT    74.85    74.85
```

```
53810  62381  GYGWWHQWW    1  24OCT2009          RT  1199.90  1199.85

            ORDERNO                     DEPTDESCR
0      CCXXNNXXXXUX                    Home Audio
1      CCXXNNXXXXUX              Small Appliances
2      CCXNXXKXXXRI              Small Appliances
3      CCXNXXKXXXRI              Small Appliances
4      CCXNXXKXXXRI              Small Appliances
5      CCXNXXKXXXRI              Small Appliances
6      CCXNXXKXXXRI              Small Appliances
7      CCXNXXKXXXRI                    Home Audio
8      CCXNKNNXXXNC  Mobile Electronic Accessories
9      CCXNKNNXXXNC  Mobile Electronic Accessories
10     CCXNKNNXXXNC  Mobile Electronic Accessories
11     CCXNKNNXXXNC  Mobile Electronic Accessories
12     CCXZZKRXXXKI                    Home Audio
13     CCXCUKRXXXVI          Portable Electronics
14     CCXZVKRXXXNI              Small Appliances
15     CCXZVKRXXXNI              Small Appliances
16     CCXZVKRXXXNI  Mobile Electronic Accessories
17     CCXZVKRXXXNI  Mobile Electronic Accessories
18     CCXZVKRXXXNI  Mobile Electronic Accessories
19     CCXZVKRXXXNI              Small Appliances
20     CCXZVKRXXXNI  Mobile Electronic Accessories
21     CCXZVKRXXXNI  Mobile Electronic Accessories
22     CCXZVKRXXXNI              Small Appliances
23     CCXZVKRXXXNI  Mobile Electronic Accessories
24     CCXZVKRXXXNI  Mobile Electronic Accessories
25     CCXZVKVXXXNI  Mobile Electronic Accessories
26     CCXCURUXXXVI  Mobile Electronic Accessories
27     CCXCURUXXXVI  Mobile Electronic Accessories
28     CCXCURUXXXVI              Small Appliances
29     CCXCURUXXXVI  Mobile Electronic Accessories
...             ...                           ...
53781  CCXINNVXXXKC              Small Appliances
53782  CCXINNVXXXKC              Small Appliances
53783  CCXINNVXXXKC              Small Appliances
53784  CCXINNVXXXKC              Small Appliances
53785  CCXKEUIXXXNC  Mobile Electronic Accessories
53786  CCXXUKZXXXNI              Small Appliances
53787  CCXXUKZXXXNI  Mobile Electronic Accessories
53788  CCXXUKZXXXNI              Small Appliances
53789  CCXXUKZXXXNI              Small Appliances
53790  CCXXUKZXXXNI  Mobile Electronic Accessories
53791  CCXXUKZXXXNI              Small Appliances
53792  CCXXUKZXXXNI              Small Appliances
53793  CCXXUKZXXXNI              Small Appliances
```

```
53794  CCXNCKZXXXVC                     Home Audio
53795  CCXNCKZXXXVC  Mobile Electronic Accessories
53796  CCXNCKZXXXVC  Mobile Electronic Accessories
53797  CCXNCKZXXXVC  Mobile Electronic Accessories
53798  CCXNCKZXXXVC  Mobile Electronic Accessories
53799  CCXXNXZXXXNI               Small Appliances
53800  CCXXNXZXXXNI               Small Appliances
53801  CCXXNXZXXXNI               Small Appliances
53802  CCXUVZUXXXKI           Portable Electronics
53803  CCXUVZUXXXKI                     Home Audio
53804  CCXIVCCXXXNI                     Home Audio
53805  CCXCXIKXXXNI  Mobile Electronic Accessories
53806  CCXCXIKXXXNI  Mobile Electronic Accessories
53807  CCXCXIKXXXNI             Mobile Electronics
53808  CCXCRZEXXXNI  Mobile Electronic Accessories
53809  CCXCRZIXXXNI               Small Appliances
53810  CCXKXKRXXXRI                     Home Audio

[53811 rows x 9 columns]
```

[44]:
```
# read in all customer data
custtrans=pd.read_sql_query("SELECT * FROM xyzcust", engine)
```

[45]:
```
# look at first 5 rows
custtrans.head()
```

[45]:
```
   index     ACCTNO    ZIP  ZIP4  LTD_SALES  LTD_TRANSACTIONS  YTD_SALES_2009  \
0      0  WDQQLLDQL  60084  5016       90.0                 1             0.0
1      1  WQWAYHYLA  60091  1750     4227.0                 9          1263.0
2      2  GSHAPLHAW  60067   900      420.0                 3           129.0
3      3  PGGYDYWAD  60068  3838     6552.0                 6             0.0
4      4  LWPSGPLLS  60090  3932      189.0                 3            72.0

   YTD_TRANSACTIONS_2009 CHANNEL_ACQUISITION BUYER_STATUS  ZIP9_SUPERCODE
0                      0                  IB     INACTIVE       600845016
1                      3                  RT       ACTIVE       600911750
2                      1                  RT       ACTIVE       600670900
3                      0                  RT     INACTIVE       600683838
4                      1                  RT       ACTIVE       600903932
```

[46]:
```
# read in all transactional data
allrttrans=pd.read_sql_query("SELECT * FROM xyztrans", engine)
```

[47]:
```
# look at first five rows
allrttrans.head()
```

[47]:
```
   index  ACCTNO  QTY   TRANDATE TRAN_CHANNEL   PRICE  TOTAMT       ORDERNO  \
0      0  WGDQLA    1  09JUN2009           RT  599.85  599.85  CCXXNNXXXXUX
1      1  WGDQLA    1  09JUN2009           RT   39.00   39.00  CCXXNNXXXXUX
2      2  WGDQLA    1  28NOV2009           RT   15.00   15.00  CCXNXXKXXXRI
```

```
3       3  WGDQLA     1  28NOV2009           RT    69.00    69.00  CCXNXXKXXXRI
4       4  WGDQLA     1  28NOV2009           RT    84.00    84.00  CCXNXXKXXXRI

           DEPTDESCR
0         Home Audio
1   Small Appliances
2   Small Appliances
3   Small Appliances
4   Small Appliances
```

3 Requirements :

1. Get a list of all records in xyzcust table where YTD_SALES_2009 > 1000
2. Get a list of all records in xyzcust table where YTD_SALES_2009 > 1000 and CHANNEL_ACQUISITION = 'RT'
3. What is the total number of records in in xyzcust table where YTD_SALES_2009 > 1000, CHANNEL_ACQUISITION = 'RT', and ZIP = 60056

[48]:
```
# Write your python code that meets the above requirements in this cell
```

[49]:
```
pd.read_sql_query("SELECT * \
                   FROM xyzcust \
                   WHERE YTD_SALES_2009 > 1000", engine)
```

[49]:
```
       index     ACCTNO    ZIP  ZIP4  LTD_SALES  LTD_TRANSACTIONS  \
0          1  WQWAYHYLA  60091  1750     4227.0                 9
1         12  WLDAYHQLW  60091  2813     3240.0                 7
2         24   ASDHAYAW  60062  6077     3411.0                19
3         31    HDWAWLH  60069  3402    25476.0                93
4         40  GSHLHGHWW  60070  2352     3576.0                10
5         77  LGDGQPGDH  60061  4540     2364.0                17
6         78   GQHYPQYD  60093  2902    12828.0                51
7        116  WYDPLSHGP  60091  1707     7671.0                25
8        126  WYPYWWPQP  60091  1620     4812.0                23
9        139  SGAHSWLHA  60093  3748    14448.0                 5
10       231  GHYYWDLAL  60093  1004    36495.0                97
11       307  WHAHHQAAP  60056  2948     4860.0                 6
12       313  GGDALSQLG  60091  2553     3300.0                 1
13       326  LPSLDDGYA  60062  5154     3435.0                16
14       364  GHPGDAWDD  60098  2424     1272.0                 6
15       388  PWLYYQADS  60069  3211     3201.0                 5
16       397  WDDASHSAA  60062  6028     2148.0                 9
17       461    SLYLSYH  60084  9767     6978.0                17
18       479   WGHGGADH  60067  6775     8943.0                40
19       487  PGLQALDPY  60067  4242     7665.0                20
20       493   PLDDDQHL  60076  2132     1170.0                 8
21       545  ALQSDWDWD  60091  3024     1362.0                 3
```

```
22      563  GGHAHHYDW  60091  1524     4341.0                17
23      564  GGAWQLAQP  60074  3875     5529.0                46
24      584  LDAGWDWGH  60091  1636     4527.0                23
25      605  SSWQPHAAL  60068  2865     2844.0                10
26      628  SGHWGWYYA  60089  6822     6762.0                15
27      655  PYSWDYHPS  60091  1512     5484.0                14
28      701  PPLQYQSLW  60093  1501     3195.0                 6
29      777   YYAPWDWP  60062  1027     1020.0                 2
...     ...        ...    ...   ...        ...               ...
1603  30029  SLADGALPA  60067  4638     2331.0                15
1604  30041  WPWQSAYYY  60062  4938     2844.0                13
1605  30061   HAYLQDGD  60062  5159     2466.0                 5
1606  30098  PGAGWYPHW  60067  4858     2919.0                10
1607  30118   GSSDDHAD  60093  3828    10062.0                10
1608  30120  SPSYSLDAA  60093  1638     4329.0                14
1609  30148  WQDSWAQGG  60074  7042     3897.0                 7
1610  30151  ADAWGPSAP  60060  1021     2712.0                18
1611  30160  AHWYWYAPH  60091  1134     6444.0                13
1612  30172  SQHGQPYWD  60098     0     8238.0                14
1613  30181  GPDQDAYYD  60098  3215     4557.0                 4
1614  30208  SLLWSDPQS  60098  8871     4071.0                13
1615  30221  WPPPHLQPS  60098  8146     6768.0                14
1616  30225  WDYHSAPDH  60098  8993     6090.0                11
1617  30228   PGDAAPPD  60098  9446     1437.0                 3
1618  30233   LDDLASSS  60098  7855    12981.0                36
1619  30260  LWYGPLGPS  60098  9011     4191.0                 8
1620  30283  WSAYGYYQS  60098  3362     4203.0                 7
1621  30286  ASSAWWQHH  60098  7903     1380.0                 4
1622  30300  PWDWPPDAY  60098  7881     1929.0                 6
1623  30304  AQPLGQSHD  60098  4206     4608.0                17
1624  30310  GGSDQLHGY  60098  2271     1068.0                 2
1625  30329  WSAGSPDPQ  60098  8877     3669.0                21
1626  30330  WDGYGAQQH  60098  8048     2685.0                 6
1627  30336  PLHHGGQYH  60098  8075     6681.0                16
1628  30358  LWWAWAPQD  60098  8091    21030.0                20
1629  30379  AYQWWQLHY  60098  7943     4092.0                 9
1630  30406   WWQYYPSA  60098  3133     2100.0                 3
1631  30408  WLLWDLLYD  60098  7807     1827.0                 2
1632  30454  LLQLHHQYP  60098  7108     2184.0                 3

      YTD_SALES_2009  YTD_TRANSACTIONS_2009 CHANNEL_ACQUISITION BUYER_STATUS  \
0             1263.0                      3                  RT       ACTIVE
1             2064.0                      3                  RT       ACTIVE
2             1875.0                      5                  RT       ACTIVE
3             1623.0                      4                  RT       ACTIVE
4             1398.0                      3                  IB       ACTIVE
5             1359.0                      7                  RT       ACTIVE
```

```
6             1815.0                      7                   RT       ACTIVE
7             1152.0                      6                   IB       ACTIVE
8             1116.0                      5                   RT       ACTIVE
9            14448.0                      5                   RT       ACTIVE
10            5586.0                     13                   RT       ACTIVE
11            2349.0                      2                   CB       ACTIVE
12            3300.0                      1                   IB       ACTIVE
13            1410.0                      8                   RT       ACTIVE
14            1272.0                      6                   RT       ACTIVE
15            1029.0                      3                   RT       ACTIVE
16            1026.0                      3                   RT       ACTIVE
17            1152.0                      3                   CB       ACTIVE
18            1845.0                      8                   RT       ACTIVE
19            3702.0                      7                   RT       ACTIVE
20            1170.0                      8                   RT       ACTIVE
21            1245.0                      2                   RT       ACTIVE
22            1083.0                      2                   RT       ACTIVE
23            1122.0                      9                   IB       ACTIVE
24            1023.0                      3                   IB       ACTIVE
25            1818.0                      5                   RT       ACTIVE
26            1023.0                      2                   RT       ACTIVE
27            1218.0                      3                   RT       ACTIVE
28            1074.0                      1                   CB       ACTIVE
29            1020.0                      2                   IB       ACTIVE
...              ...                    ...                  ...          ...
1603          1134.0                      3                   RT       ACTIVE
1604          1038.0                      3                   IB       ACTIVE
1605          2298.0                      3                   RT       ACTIVE
1606          1827.0                      2                   RT       ACTIVE
1607          1728.0                      1                   RT       ACTIVE
1608          1416.0                      1                   RT       ACTIVE
1609          1008.0                      2                   RT       ACTIVE
1610          1377.0                      6                   RT       ACTIVE
1611          4173.0                      9                   RT       ACTIVE
1612          1404.0                      3                   RT       ACTIVE
1613          1920.0                      2                   RT       ACTIVE
1614          1293.0                      4                   RT       ACTIVE
1615          2655.0                      2                   RT       ACTIVE
1616          1449.0                      3                   RT       ACTIVE
1617          1437.0                      3                   IB       ACTIVE
1618          1308.0                      8                   CB       ACTIVE
1619          1158.0                      2                   RT       ACTIVE
1620          1989.0                      1                   RT       ACTIVE
1621          1296.0                      3                   RT       ACTIVE
1622          1491.0                      2                   RT       ACTIVE
1623          1740.0                      5                   RT       ACTIVE
1624          1068.0                      2                   RT       ACTIVE
```

```
1625          1263.0                      4                  RT       ACTIVE
1626          1296.0                      1                  RT       ACTIVE
1627          2985.0                      7                  RT       ACTIVE
1628          5322.0                      5                  RT       ACTIVE
1629          2625.0                      3                  RT       ACTIVE
1630          1800.0                      2                  IB       ACTIVE
1631          1827.0                      2                  RT       ACTIVE
1632          1248.0                      2                  RT       ACTIVE

      ZIP9_SUPERCODE
0          600911750
1          600912813
2          600626077
3          600693402
4          600702352
5          600614540
6          600932902
7          600911707
8          600911620
9          600933748
10         600931004
11         600562948
12         600912553
13         600625154
14         600982424
15         600693211
16         600626028
17         600849767
18         600676775
19         600674242
20         600762132
21         600913024
22         600911524
23         600743875
24         600911636
25         600682865
26         600896822
27         600911512
28         600931501
29         600621027
...              ...
1603       600674638
1604       600624938
1605       600625159
1606       600674858
1607       600933828
1608       600931638
```

```
1609       600747042
1610       600601021
1611       600911134
1612       600988087
1613       600983215
1614       600988871
1615       600988146
1616       600988993
1617       600989446
1618       600987855
1619       600989011
1620       600983362
1621       600987903
1622       600987881
1623       600984206
1624       600982271
1625       600988877
1626       600988048
1627       600988075
1628       600988091
1629       600987943
1630       600983133
1631       600987807
1632       600987108

[1633 rows x 11 columns]
```

[50]:

```
pd.read_sql_query("SELECT * FROM xyzcust \
                    WHERE YTD_SALES_2009 > 1000 \
                    AND CHANNEL_ACQUISITION = 'RT'", engine)
```

[50]:

```
      index     ACCTNO    ZIP  ZIP4  LTD_SALES  LTD_TRANSACTIONS  \
0         1  WQWAYHYLA  60091  1750     4227.0                 9
1        12  WLDAYHQLW  60091  2813     3240.0                 7
2        24   ASDHAYAW  60062  6077     3411.0                19
3        31    HDWAWLH  60069  3402    25476.0                93
4        77  LGDGQPGDH  60061  4540     2364.0                17
5        78   GQHYPQYD  60093  2902    12828.0                51
6       126  WYPYWWPQP  60091  1620     4812.0                23
7       139  SGAHSWLHA  60093  3748    14448.0                 5
8       231  GHYYWDLAL  60093  1004    36495.0                97
9       326  LPSLDDGYA  60062  5154     3435.0                16
10      364  GHPGDAWDD  60098  2424     1272.0                 6
11      388  PWLYYQADS  60069  3211     3201.0                 5
12      397  WDDASHSAA  60062  6028     2148.0                 9
13      479   WGHGGADH  60067  6775     8943.0                40
14      487  PGLQALDPY  60067  4242     7665.0                20
15      493   PLDDDQHL  60076  2132     1170.0                 8
```

```
16      545  ALQSDWDWD  60091  3024     1362.0                 3
17      563  GGHAHHYDW  60091  1524     4341.0                17
18      605  SSWQPHAAL  60068  2865     2844.0                10
19      628  SGHWGWYYA  60089  6822     6762.0                15
20      655  PYSWDYHPS  60091  1512     5484.0                14
21      808  GHGLQQYYH  60081  8744     1320.0                 1
22      823  APLLSGSDG  60067  7900     6492.0                16
23      844  WYAYGPPLP  60093  2436     7176.0                27
24      879   DLHSWSDP  60091  1526    37998.0                53
25      890  WPGWAWSQG  60089  3341     4464.0                14
26      910  GHQASLYSH  60093  2521     4323.0                 9
27     1012  AGDDLWSWL  60056  2137     1806.0                 5
28     1021    YLSAYGS  60076  2844     1074.0                 1
29     1030  AAGSALPSD  60091  2158     3687.0                 7
...     ...        ...    ...   ...        ...               ...
1177  29978   ASHDGGYLY  60076  2854     2796.0                 2
1178  29992   GPYSDDGPQ  60093  4251     2454.0                 7
1179  29996   GHPWWLHHD  60069  3062     1089.0                 4
1180  30024    SADWAGWQ  60091  1603     1425.0                 3
1181  30029   SLADGALPA  60067  4638     2331.0                15
1182  30061    HAYLQDGD  60062  5159     2466.0                 5
1183  30098   PGAGWYPHW  60067  4858     2919.0                10
1184  30118    GSSDDHAD  60093  3828    10062.0                10
1185  30120   SPSYSLDAA  60093  1638     4329.0                14
1186  30148   WQDSWAQGG  60074  7042     3897.0                 7
1187  30151   ADAWGPSAP  60060  1021     2712.0                18
1188  30160   AHWYWYAPH  60091  1134     6444.0                13
1189  30172   SQHGQPYWD  60098     0     8238.0                14
1190  30181   GPDQDAYYD  60098  3215     4557.0                 4
1191  30208   SLLWSDPQS  60098  8871     4071.0                13
1192  30221   WPPPHLQPS  60098  8146     6768.0                14
1193  30225   WDYHSAPDH  60098  8993     6090.0                11
1194  30260   LWYGPLGPS  60098  9011     4191.0                 8
1195  30283   WSAYGYYQS  60098  3362     4203.0                 7
1196  30286   ASSAWWQHH  60098  7903     1380.0                 4
1197  30300   PWDWPPDAY  60098  7881     1929.0                 6
1198  30304   AQPLGQSHD  60098  4206     4608.0                17
1199  30310   GGSDQLHGY  60098  2271     1068.0                 2
1200  30329   WSAGSPDPQ  60098  8877     3669.0                21
1201  30330   WDGYGAQQH  60098  8048     2685.0                 6
1202  30336   PLHHGGQYH  60098  8075     6681.0                16
1203  30358   LWWAWAPQD  60098  8091    21030.0                20
1204  30379   AYQWWQLHY  60098  7943     4092.0                 9
1205  30408   WLLWDLLYD  60098  7807     1827.0                 2
1206  30454   LLQLHHQYP  60098  7108     2184.0                 3

      YTD_SALES_2009  YTD_TRANSACTIONS_2009 CHANNEL_ACQUISITION BUYER_STATUS  \
0             1263.0                           3                  RT       ACTIVE
1             2064.0                           3                  RT       ACTIVE
2             1875.0                           5                  RT       ACTIVE
3             1623.0                           4                  RT       ACTIVE
4             1359.0                           7                  RT       ACTIVE
5             1815.0                           7                  RT       ACTIVE
6             1116.0                           5                  RT       ACTIVE
7            14448.0                           5                  RT       ACTIVE
8             5586.0                          13                  RT       ACTIVE
9             1410.0                           8                  RT       ACTIVE
10            1272.0                           6                  RT       ACTIVE
11            1029.0                           3                  RT       ACTIVE
12            1026.0                           3                  RT       ACTIVE
13            1845.0                           8                  RT       ACTIVE
14            3702.0                           7                  RT       ACTIVE
15            1170.0                           8                  RT       ACTIVE
16            1245.0                           2                  RT       ACTIVE
17            1083.0                           2                  RT       ACTIVE
18            1818.0                           5                  RT       ACTIVE
19            1023.0                           2                  RT       ACTIVE
20            1218.0                           3                  RT       ACTIVE
21            1320.0                           1                  RT       ACTIVE
22            1209.0                           2                  RT       ACTIVE
23            1461.0                           4                  RT       ACTIVE
24            5133.0                          11                  RT       ACTIVE
25            2007.0                           1                  RT       ACTIVE
26            1395.0                           3                  RT       ACTIVE
27            1806.0                           5                  RT       ACTIVE
28            1074.0                           1                  RT       ACTIVE
29            2097.0                           3                  RT       ACTIVE
...              ...                         ...                 ...          ...
1177          2730.0                           1                  RT       ACTIVE
1178          1704.0                           5                  RT       ACTIVE
1179          1089.0                           4                  RT       ACTIVE
1180          1104.0                           2                  RT       ACTIVE
1181          1134.0                           3                  RT       ACTIVE
1182          2298.0                           3                  RT       ACTIVE
1183          1827.0                           2                  RT       ACTIVE
1184          1728.0                           1                  RT       ACTIVE
1185          1416.0                           1                  RT       ACTIVE
1186          1008.0                           2                  RT       ACTIVE
1187          1377.0                           6                  RT       ACTIVE
1188          4173.0                           9                  RT       ACTIVE
1189          1404.0                           3                  RT       ACTIVE
1190          1920.0                           2                  RT       ACTIVE
1191          1293.0                           4                  RT       ACTIVE
1192          2655.0                           2                  RT       ACTIVE
```

```
1193          1449.0                      3                   RT       ACTIVE
1194          1158.0                      2                   RT       ACTIVE
1195          1989.0                      1                   RT       ACTIVE
1196          1296.0                      3                   RT       ACTIVE
1197          1491.0                      2                   RT       ACTIVE
1198          1740.0                      5                   RT       ACTIVE
1199          1068.0                      2                   RT       ACTIVE
1200          1263.0                      4                   RT       ACTIVE
1201          1296.0                      1                   RT       ACTIVE
1202          2985.0                      7                   RT       ACTIVE
1203          5322.0                      5                   RT       ACTIVE
1204          2625.0                      3                   RT       ACTIVE
1205          1827.0                      2                   RT       ACTIVE
1206          1248.0                      2                   RT       ACTIVE

      ZIP9_SUPERCODE
0          600911750
1          600912813
2          600626077
3          600693402
4          600614540
5          600932902
6          600911620
7          600933748
8          600931004
9          600625154
10         600982424
11         600693211
12         600626028
13         600676775
14         600674242
15         600762132
16         600913024
17         600911524
18         600682865
19         600896822
20         600911512
21         600818744
22         600677900
23         600932436
24         600911526
25         600893341
26         600932521
27         600562137
28         600762844
29         600912158
...              ...
```

```
1177        600762854
1178        600934251
1179        600693062
1180        600911603
1181        600674638
1182        600625159
1183        600674858
1184        600933828
1185        600931638
1186        600747042
1187        600601021
1188        600911134
1189        600988087
1190        600983215
1191        600988871
1192        600988146
1193        600988993
1194        600989011
1195        600983362
1196        600987903
1197        600987881
1198        600984206
1199        600982271
1200        600988877
1201        600988048
1202        600988075
1203        600988091
1204        600987943
1205        600987807
1206        600987108

[1207 rows x 11 columns]
```

[55]:
```
len(pd.read_sql_query("SELECT * FROM xyzcust \
                    WHERE YTD_SALES_2009 > 1000 \
                    AND CHANNEL_ACQUISITION = 'RT'\
                    AND ZIP = 60056", engine))
```

[55]:
```
49
```