Siddikov _ Exercise 4 version b

August 16, 2019

1 Deliverables:

- Submit two files that has the name: YourLastName_Exercise_4:

1. Your **PDF document** that has your Source code and output
2. Your **ipynb script** that has your Source code and output
3. You may zip these 2 files and submit

2 Objectives:

In this exercise, you will:

- Analyze the dataset in the given CSV file
- Clean the given dataset
- Load the dataset into sqlite database engine
- Execute different 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.

2.0.1 Data Preparation

As a data scientist for BestDeal retailer, you have been tasked with improving their revenue and the effectiveness of the marketing campaign of their electronic products. The given dataset has 10,000 records for the purchases of their customers and is used to predict customers shopping patterns and to provide answers for ad-hoc queries. The dataset DirtyData4BestDeal10000.csv is drawn from its database of customers.

[1]:
```
import pandas as pd  # panda's nickname is pd

import numpy as np  # numpy as np

from pandas import DataFrame, Series     # for convenience

import sqlalchemy
```

```
from sqlalchemy import create_engine

from sqlalchemy import inspect

%matplotlib inline
# ignore all future warnings
from warnings import simplefilter
simplefilter(action='ignore', category=FutureWarning)
import warnings
warnings.filterwarnings("ignore")
```

2.0.2 Lets ead the dirtydata4bestdeal CSV and load into a dataframe object

[2]:
```
dirtydata4bestdeal=pd.read_csv('DirtyData4BestDeal10000.csv')
```

[3]:
```
# Do you see NaN values below?

dirtydata4bestdeal.head()
```

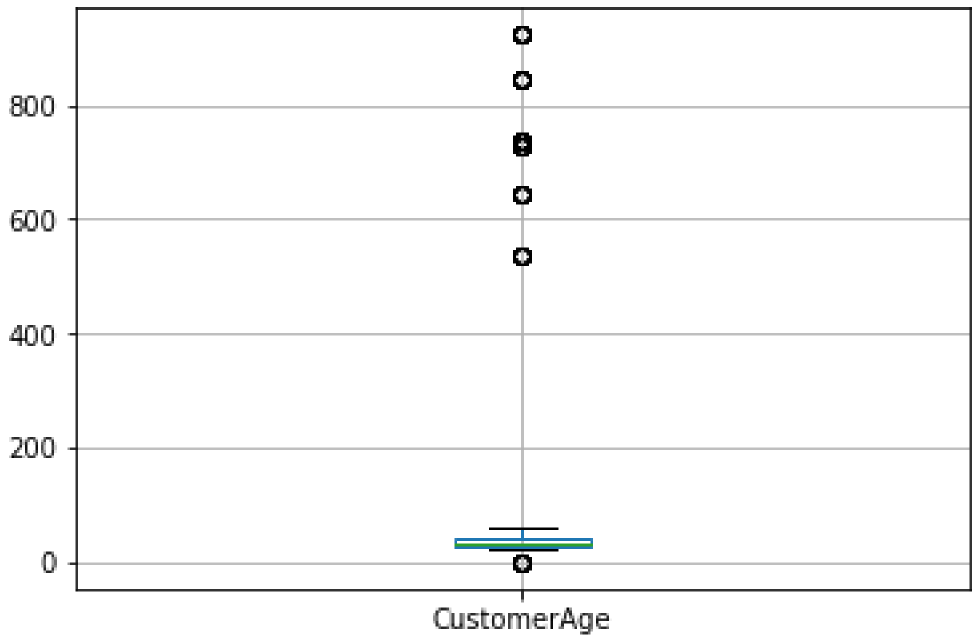
[3]:
```
   ZipCode  CustomerAge  SamsungTV46LED SonyTV42LED XBOX360 DellLaptop  \
0  30134.0         35.0               1           1       1          0
1  62791.0         43.0               0           1       0          0
2  60611.0         23.0               1         NaN       0          1
3  60616.0         56.0               0           1       1          1
4  30303.0         25.0               1         NaN       0        NaN

  BoseSoundSystem  BoseHeadSet  SonyHeadSet  iPod          ...          \
0               0          1.0          1.0   0.0          ...
1               1          0.0          1.0   0.0          ...
2               0          NaN          1.0   1.0          ...
3               0          0.0          1.0   1.0          ...
4               1          1.0          1.0   0.0          ...

   GalaxyTablet  SurfaceTablet  HPLaptop  HDMICable  SpeakerCable  \
0             1            0.0       1.0        1.0           1.0
1             1            0.0       1.0        0.0           1.0
2             0            0.0       1.0        0.0           1.0
3             0            0.0       1.0        0.0           1.0
4             1            0.0       1.0        1.0           1.0

   CallOfDutyGame  GrandTheftAutoGame  ASUSLaptop  LenevoLaptop  \
0             1.0                 0.0         1.0           1.0
1             1.0                 0.0         1.0           1.0
2             1.0                 0.0         NaN           1.0
3             0.0                 0.0         1.0           0.0
4             1.0                 0.0         1.0          10.0
```

```
   TVStandWallMount
0                 1
1                 1
2                 1
3                 0
4                 0

[5 rows x 34 columns]
```

2.0.3 Lets use boxplot to visualize the data and get an idea if there are dirty/messy/invalid data

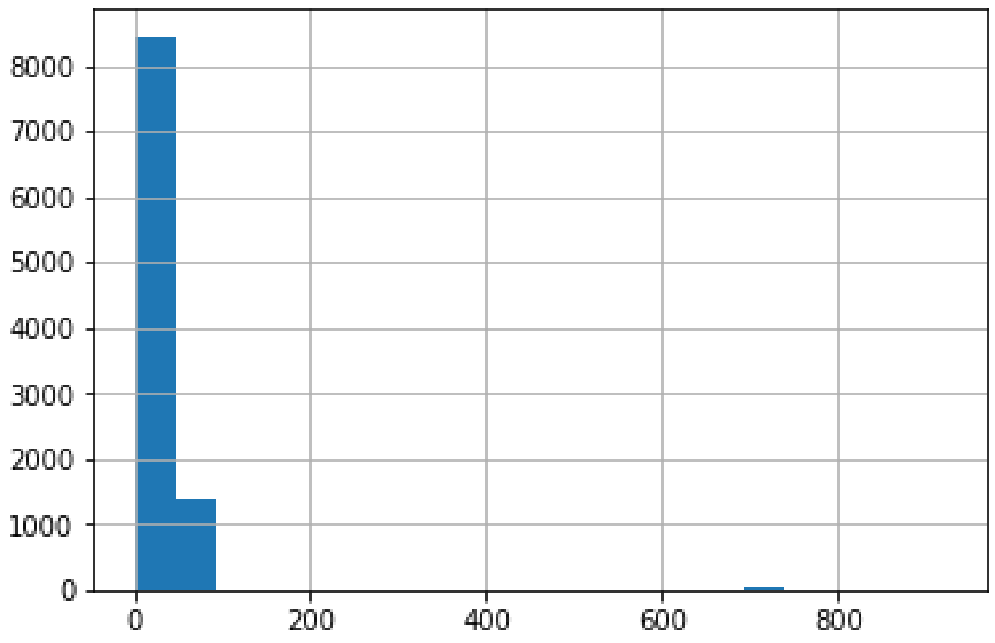
[4]:
```
# check out customer age
dirtydata4bestdeal.boxplot(column='CustomerAge')
```

[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25111d66a20>
```



[5]:
```
# check out customer age with a histogram
dirtydata4bestdeal['CustomerAge'].hist(bins=20)
```

[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25111e31320>
```

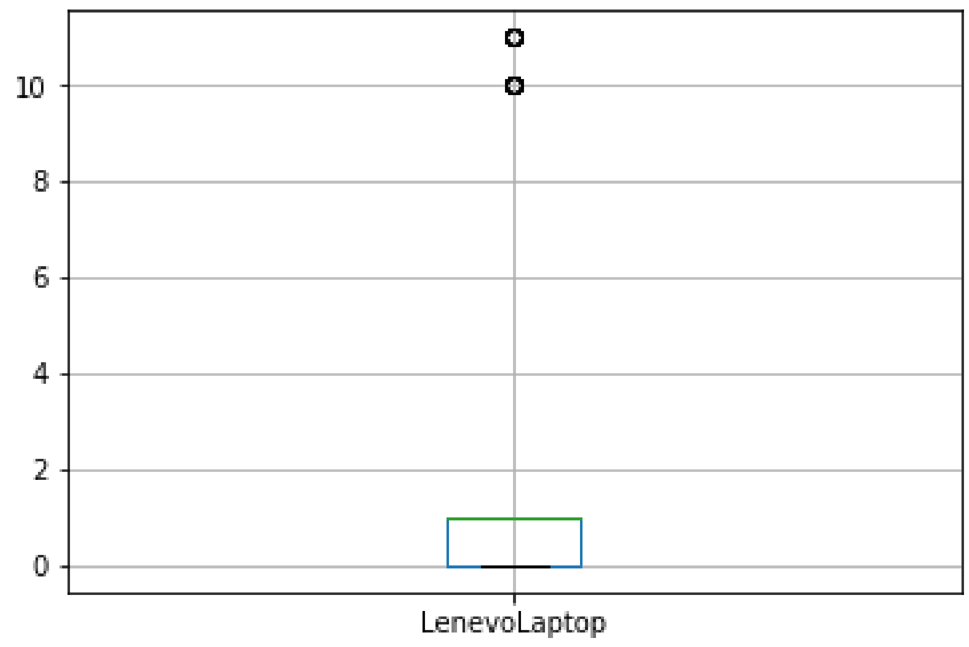


[6]:
```
# look at details of LenenovaLaptop
dirtydata4bestdeal.LenevoLaptop.describe()
```

[6]:
```
count    9976.000000
mean        0.629711
std         0.627375
min         0.000000
25%         0.000000
50%         1.000000
75%         1.000000
max        11.000000
Name: LenevoLaptop, dtype: float64
```

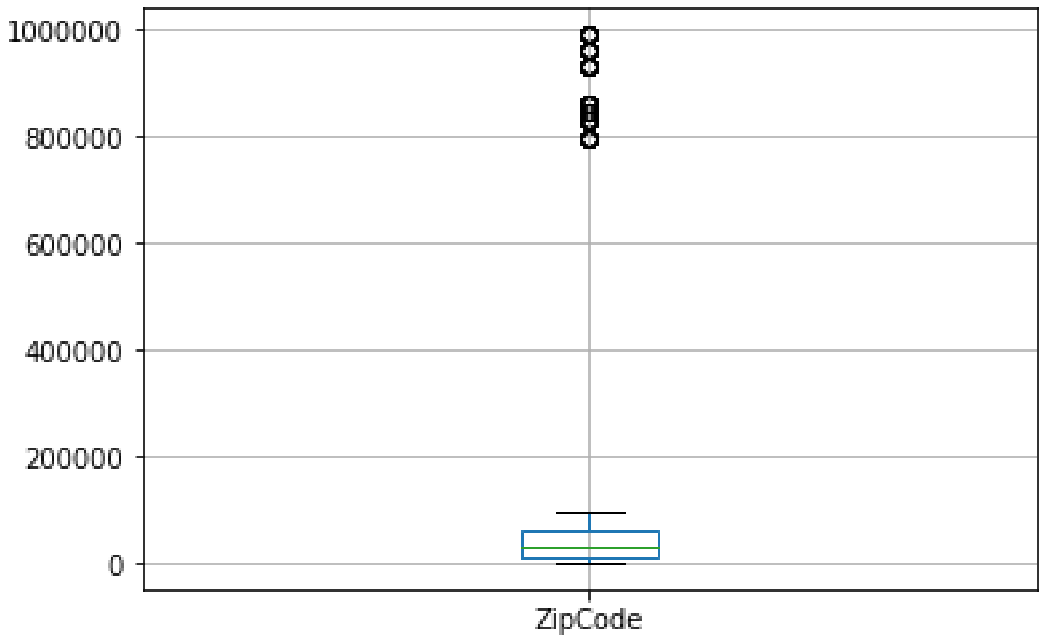
[7]:
```
dirtydata4bestdeal.boxplot(column='LenevoLaptop')
```

[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x251140dcac8>
```



[8]:
```
# look at zip codes
dirtydata4bestdeal.boxplot(column='ZipCode')
```

[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25111e264a8>
```



2.0.4 Lets clean the dirty/messy data in the dirtydata4bestdeal dataframe object

You need to write your python code such that: 1. rows/records/tuples/transactions in the data frame that have missing values for fields/columns will be removed 2. rows/records/tuples/transactions in the data frame that have invalid/abnormal values for fields/columns will be removed

Examples of invalid/dirty/messy data: 1. NaN values in the dataframe (Blank/Empty cells in the CSV file)

2. Every product has a value 1 which means bought or 0 which means NOT bought; values like 11, 10, 9 are examples of invalid data

3. CustomerAge value range could be from 18 to 150; values like 723, 634 are examples of invalid data

4. Zipcode should have at least 5 digits

[9]:
```
dirtydata4bestdeal.head()
```

[9]:
```
   ZipCode  CustomerAge  SamsungTV46LED SonyTV42LED XBOX360 DellLaptop  \
0  30134.0         35.0               1           1       1          0
1  62791.0         43.0               0           1       0          0
2  60611.0         23.0               1         NaN       0          1
3  60616.0         56.0               0           1       1          1
4  30303.0         25.0               1         NaN       0        NaN

  BoseSoundSystem  BoseHeadSet  SonyHeadSet  iPod        ...           \
0               0          1.0          1.0   0.0        ...
1               1          0.0          1.0   0.0        ...
2               0          NaN          1.0   1.0        ...
3               0          0.0          1.0   1.0        ...
4               1          1.0          1.0   0.0        ...

   GalaxyTablet  SurfaceTablet  HPLaptop  HDMICable  SpeakerCable  \
0             1            0.0       1.0        1.0           1.0
1             1            0.0       1.0        0.0           1.0
2             0            0.0       1.0        0.0           1.0
3             0            0.0       1.0        0.0           1.0
4             1            0.0       1.0        1.0           1.0

   CallOfDutyGame  GrandTheftAutoGame  ASUSLaptop  LenevoLaptop  \
0             1.0                 0.0         1.0           1.0
1             1.0                 0.0         1.0           1.0
2             1.0                 0.0         NaN           1.0
3             0.0                 0.0         1.0           0.0
4             1.0                 0.0         1.0          10.0

   TVStandWallMount
0                 1
1                 1
2                 1
3                 0
```

```
4                 0

[5 rows x 34 columns]
```

[10]:
```
#␣
↪-----------------------------------------------------------------------------------

# Add the rest of your code here to clean the data

# steps you must take
# - eliminate NA's
# - product values should only be either a 0 or a 1
# - customer's age needs to be valid
# - zipcodes should have at least 5 digits

# Optional steps
# - if there are other things you want to clean, clearly document them
#   and run them in this section before you create a database


#␣
↪-----------------------------------------------------------------------------------
```

[11]:
```
# Drop the NaN values

cleandata4bestdeal=dirtydata4bestdeal.dropna()
cleandata4bestdeal.head()

# Do you see NaN values dropped below?
```

[11]:
```
   ZipCode  CustomerAge  SamsungTV46LED SonyTV42LED XBOX360 DellLaptop  \
0  30134.0         35.0               1           1       1          0
1  62791.0         43.0               0           1       0          0
3  60616.0         56.0               0           1       1          1
5   2108.0         55.0               1           1       1          1
6  90033.0         44.0               1           1       1          1

  BoseSoundSystem  BoseHeadSet  SonyHeadSet  iPod         ...          \
0               0          1.0          1.0   0.0         ...
1               1          0.0          1.0   0.0         ...
3               0          0.0          1.0   1.0         ...
5              10          0.0          0.0   0.0         ...
6               0          0.0          0.0   0.0         ...

   GalaxyTablet  SurfaceTablet  HPLaptop  HDMICable  SpeakerCable  \
0             1            0.0       1.0        1.0           1.0
1             1            0.0       1.0        0.0           1.0
3             0            0.0       1.0        0.0           1.0
```

```
5             1            1.0       1.0        1.0           1.0
6             1            1.0       1.0        1.0           0.0

   CallOfDutyGame  GrandTheftAutoGame  ASUSLaptop  LenevoLaptop  \
0             1.0                 0.0         1.0           1.0
1             1.0                 0.0         1.0           1.0
3             0.0                 0.0         1.0           0.0
5             1.0                 0.0         1.0           0.0
6             1.0                 1.0         0.0           0.0

   TVStandWallMount
0                 1
1                 1
3                 0
5                 0
6                 1

[5 rows x 34 columns]
```

[12]:
```
# convert objects and floats into integers
cleandata4bestdeal['SonyTV42LED'] = pd.
 ↪to_numeric(cleandata4bestdeal['SonyTV42LED'], errors='coerce').fillna(0).
 ↪astype(int)
cleandata4bestdeal['XBOX360'] = pd.to_numeric(cleandata4bestdeal['XBOX360'],␣
 ↪errors='coerce').fillna(0).astype(int)
cleandata4bestdeal['DellLaptop'] = pd.
 ↪to_numeric(cleandata4bestdeal['DellLaptop'], errors='coerce').fillna(0).
 ↪astype(int)
cleandata4bestdeal['BoseSoundSystem'] = pd.
 ↪to_numeric(cleandata4bestdeal['BoseSoundSystem'], errors='coerce').fillna(0).
 ↪astype(int)

cleandata4bestdeal = cleandata4bestdeal.astype('int32')
```

[13]:
```
# product values should only be either a 0 or a 1
cleandata4bestdeal.loc[:,'SamsungTV46LED':'TVStandWallMount'] = \
cleandata4bestdeal.loc[:,'SamsungTV46LED':'TVStandWallMount']\
[cleandata4bestdeal.loc[:,'SamsungTV46LED':'TVStandWallMount'].isin([0, 1])].
 ↪fillna(0).astype(int)
```

[14]:
```
cleandata4bestdeal.loc[:,'SamsungTV46LED':'TVStandWallMount'].apply(pd.
 ↪value_counts)
```

[14]:
```
   SamsungTV46LED  SonyTV42LED  XBOX360  DellLaptop  BoseSoundSystem  \
0            3123         1834     1814        4519             4907
1            6309         7598     7618        4913             4525

   BoseHeadSet  SonyHeadSet  iPod  iPhone  Panasonic50LED        ...         \
0         4595         1536  7806    6030            7215        ...
```

```
1         4837          7896  1626    3402            2217        ...

   GalaxyTablet  SurfaceTablet  HPLaptop  HDMICable  SpeakerCable  \
0          2884           8773       NaN       4812          3073
1          6548            659    9432.0       4620          6359

   CallOfDutyGame  GrandTheftAutoGame  ASUSLaptop  LenevoLaptop  \
0            2569                6204        3927          3638
1            6863                3228        5505          5794

   TVStandWallMount
0              2745
1              6687

[2 rows x 32 columns]
```

[15]:
```
#There are zero and over 500 ages; which are invalid
cleandata4bestdeal.loc[:,'CustomerAge'].value_counts().sort_index()
```

[15]:
```
0        8
21     209
22     281
23     775
24     184
25     497
26     352
27     513
28     495
29     472
30      16
31     224
32     184
33     184
34     488
35     373
36     200
37     512
38     457
39     104
41      96
42     184
43     472
44     536
45     128
46     151
47     104
49     192
51     104
```

```
53      24
54     297
55      72
56     192
57     144
59     120
61      32
536      8
643      8
727      8
737     16
843      8
923      8
Name: CustomerAge, dtype: int64
```

[16]:
```
#Zip codes should be less than 5 digits

cleandata4bestdeal.loc[:,'ZipCode'].astype('int').value_counts().sort_index()
```

[16]:
```
2108       632
2109       955
2110       224
10065      788
30134     1173
30303     1001
33129      554
33130      280
44114      526
60532      243
60585      248
60603      240
60611       62
60616      960
62791        3
90024      144
90033      665
94102      166
94158      512
794158       8
830134       8
844114       8
860616       8
930134       8
960616       8
990033       8
Name: ZipCode, dtype: int64
```

[17]:
```
#customer's age needs to be valid

cleandata4bestdeal_1 = cleandata4bestdeal[
    cleandata4bestdeal['CustomerAge'].between(20,70)]
```

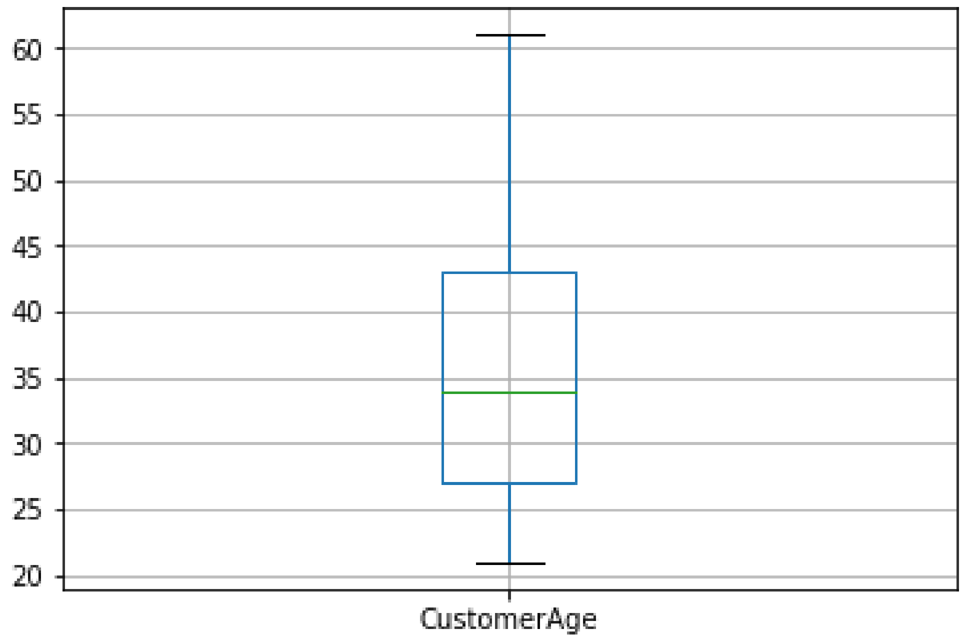
[18]:
```
#zipcodes should have at least 5 digits

cleandata4bestdeal_2 = cleandata4bestdeal_1[
    cleandata4bestdeal_1['ZipCode'] < 100000]
```

[19]:
```
# check the df shape after cleaning the data

print(cleandata4bestdeal.shape)
print(cleandata4bestdeal_2.shape)
```

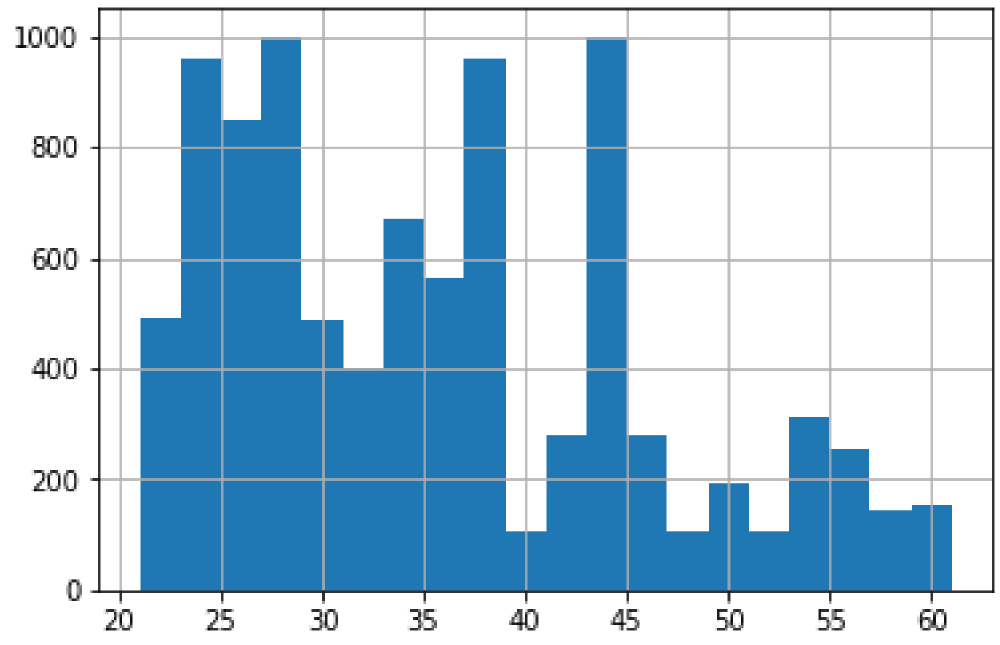
```
(9432, 34)
(9312, 34)
```

[ ]:
```

```

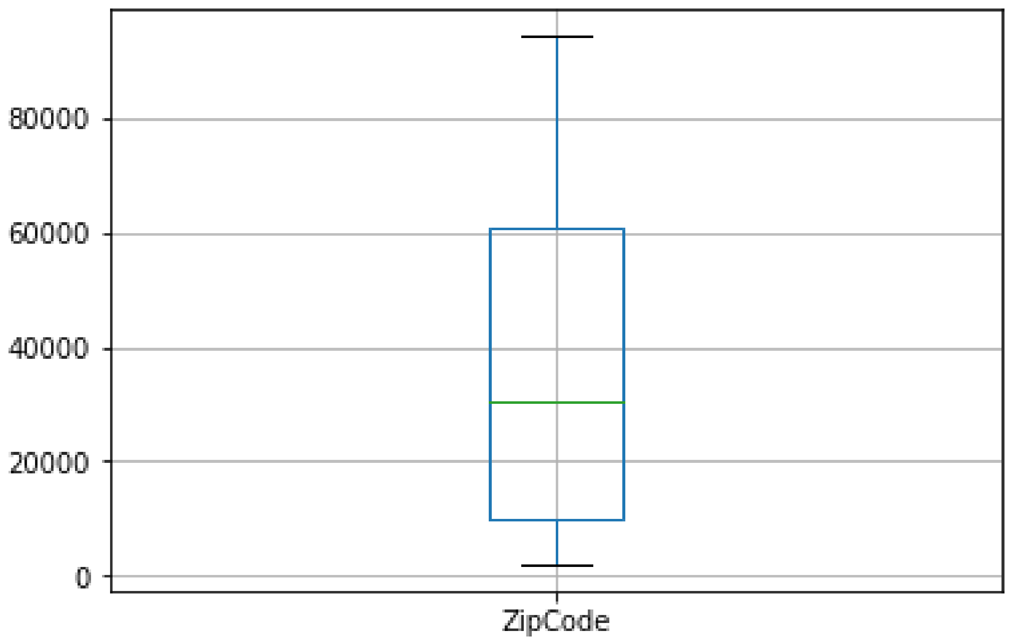
[20]:
```
# after cleaning the customer age - does the boxplot still show outliers?
# how does the histogram look?
# if this does not look better - you are not ready to proceed
cleandata4bestdeal_2.boxplot(column='CustomerAge');
```



[21]:
```
cleandata4bestdeal_2['CustomerAge'].hist(bins=20);
```



[22]:
```
# boxplot after cleaning the zip code

cleandata4bestdeal_2.boxplot(column='ZipCode');
```



2.0.5 Lets store the cleaned data into the Database

[23]:
```
# how many records did you end up with after the data cleaning?
cleandata4bestdeal = cleandata4bestdeal_2
cleandata4bestdeal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9312 entries, 0 to 9999
Data columns (total 34 columns):
ZipCode               9312 non-null int32
CustomerAge           9312 non-null int32
SamsungTV46LED        9312 non-null int32
SonyTV42LED           9312 non-null int32
XBOX360               9312 non-null int32
DellLaptop            9312 non-null int32
BoseSoundSystem       9312 non-null int32
BoseHeadSet           9312 non-null int32
SonyHeadSet           9312 non-null int32
iPod                  9312 non-null int32
iPhone                9312 non-null int32
Panasonic50LED        9312 non-null int32
SonyPS4               9312 non-null int32
WiiU                  9312 non-null int32
WDexternalHD          9312 non-null int32
SamsungTV55LED        9312 non-null int32
SonyTV60LED           9312 non-null int32
SandiskMemoryCard     9312 non-null int32
SonySoundSystem       9312 non-null int32
SonyCamera            9312 non-null int32
PanasonicCamera       9312 non-null int32
HPPrinter             9312 non-null int32
SonyDVDplayer         9312 non-null int32
ToshibaDVDplayer      9312 non-null int32
GalaxyTablet          9312 non-null int32
SurfaceTablet         9312 non-null int32
HPLaptop              9312 non-null int32
HDMICable             9312 non-null int32
SpeakerCable          9312 non-null int32
CallOfDutyGame        9312 non-null int32
GrandTheftAutoGame    9312 non-null int32
ASUSLaptop            9312 non-null int32
LenevoLaptop          9312 non-null int32
TVStandWallMount      9312 non-null int32
dtypes: int32(34)
memory usage: 1.3 MB
```

[24]:
```
# now that your data has been cleaned, lets store it in a database

# NOTE - if you run this code more than once, the database will exist and this␣
 ↪section will fail
# NOTE - to run this more than once, you need to delete the database first
#      OR - change the database name to create a new database

engine = create_engine('sqlite:///bestdeal1.db')
```

[25]:
```
cleandata4bestdeal.to_sql('trans4cust', engine)
```

** Sanity Test: Did it create the table in bestdeal.db? Check!!**

[26]:
```
insp=inspect(engine)
```

[27]:
```
 insp.get_table_names()
```

[27]:
```
['trans4cust']
```

[28]:
```
pd.read_sql_table('trans4cust', engine).columns
```

[28]:
```
Index(['index', 'ZipCode', 'CustomerAge', 'SamsungTV46LED', 'SonyTV42LED',
       'XBOX360', 'DellLaptop', 'BoseSoundSystem', 'BoseHeadSet',
       'SonyHeadSet', 'iPod', 'iPhone', 'Panasonic50LED', 'SonyPS4', 'WiiU',
       'WDexternalHD', 'SamsungTV55LED', 'SonyTV60LED', 'SandiskMemoryCard',
       'SonySoundSystem', 'SonyCamera', 'PanasonicCamera', 'HPPrinter',
       'SonyDVDplayer', 'ToshibaDVDplayer', 'GalaxyTablet', 'SurfaceTablet',
       'HPLaptop', 'HDMICable', 'SpeakerCable', 'CallOfDutyGame',
       'GrandTheftAutoGame', 'ASUSLaptop', 'LenevoLaptop', 'TVStandWallMount'],
      dtype='object')
```

should produce the columns of the DataFrame you wrote to the db.

2.0.6 Now we are ready to query the Database

Query example #1: get the transactions for the customers in zipCode 60616

[29]:
```
# ======================================================================
# **********************************************************************
#
# WARNING - this pre-run notebook is using dirty data
# WARNING - after cleaning the data, your output should look different
#
# ======================================================================
# **********************************************************************
```

[30]:
```
resultsForBestDealCustTrans=pd.read_sql_query("SELECT * FROM trans4cust WHERE␣
 ↪ZipCode='60616'", engine)
```

[31]:
```
resultsForBestDealCustTrans.head()
```

[31]:
```
   index  ZipCode  CustomerAge  SamsungTV46LED  SonyTV42LED  XBOX360  \
0      3    60616           56               0            1        1
1     16    60616           43               0            1        1
```

```
2    18  60616           54              1           0         0
3    23  60616           43              1           1         1
4    34  60616           31              0           1         1

   DellLaptop  BoseSoundSystem  BoseHeadSet  SonyHeadSet       ...         \
0           1                0            0            1       ...
1           0                1            0            1       ...
2           1                0            1            1       ...
3           0                1            1            1       ...
4           1                0            0            1       ...

   GalaxyTablet  SurfaceTablet  HPLaptop  HDMICable  SpeakerCable  \
0             0              0         1          0             1
1             1              0         1          1             1
2             0              1         1          0             1
3             1              1         1          1             0
4             1              0         1          1             1

   CallOfDutyGame  GrandTheftAutoGame  ASUSLaptop  LenevoLaptop  \
0               0                   0           1             0
1               1                   0           1             1
2               1                   0           1             1
3               1                   0           1             1
4               1                   1           0             0

   TVStandWallMount
0                 0
1                 1
2                 1
3                 1
4                 1

[5 rows x 35 columns]
```

Query example #2: get the transactions for ALL customers

[32]:
```
resultsForBestDealCustTrans=pd.read_sql_query("SELECT * \
    FROM trans4cust", engine)
```

[33]:
```
resultsForBestDealCustTrans.head()
```

[33]:
```
   index  ZipCode  CustomerAge  SamsungTV46LED  SonyTV42LED  XBOX360  \
0      0    30134           35               1            1        1
1      1    62791           43               0            1        0
2      3    60616           56               0            1        1
3      5     2108           55               1            1        1
4      6    90033           44               1            1        1
```

```
   DellLaptop  BoseSoundSystem  BoseHeadSet  SonyHeadSet        ...         \
0           0                0            1            1        ...
1           0                1            0            1        ...
2           1                0            0            1        ...
3           1                0            0            0        ...
4           1                0            0            0        ...

   GalaxyTablet  SurfaceTablet  HPLaptop  HDMICable  SpeakerCable  \
0             1              0         1          1             1
1             1              0         1          0             1
2             0              0         1          0             1
3             1              1         1          1             1
4             1              1         1          1             0

   CallOfDutyGame  GrandTheftAutoGame  ASUSLaptop  LenevoLaptop  \
0               1                   0           1             1
1               1                   0           1             1
2               0                   0           1             0
3               1                   0           1             0
4               1                   1           0             0

   TVStandWallMount
0                 1
1                 1
2                 0
3                 0
4                 1

[5 rows x 35 columns]
```

Query example #3: get the number of customers in every ZipCode sorted by ZipCode

[34]:
```
resultsForBestDealCustTrans=pd.read_sql_query("SELECT ZipCode , COUNT(*) as␣
 ↪'num_customers' \
            FROM trans4cust \
            GROUP BY ZipCode \
            ORDER BY ZipCode", engine)
```

[35]:
```
resultsForBestDealCustTrans
```

[35]:
```
   ZipCode  num_customers
0     2108            632
1     2109            943
2     2110            224
3    10065            776
4    30134           1165
5    30303           1001
6    33129            546
```

```
7       33130              280
8       44114              526
9       60532              243
10      60585              248
11      60603              240
12      60611               62
13      60616              952
14      62791                3
15      90024              144
16      90033              657
17      94102              166
18      94158              504
```

Query example #4: get the number of customers for every Age Group in ZipCode 60616 sorted by CustomerAge

[36]:
```
resultsForBestDealCustTrans=pd.read_sql_query(
"SELECT CustomerAge , COUNT(*) as 'num_customers' \
    FROM trans4cust \
    WHERE ZipCode=60616 \
    GROUP BY CustomerAge  \
    ORDER BY CustomerAge", engine)
```

[37]:
```
resultsForBestDealCustTrans
```

[37]:
```
    CustomerAge  num_customers
0            21             56
1            22             32
2            23             40
3            25             88
4            26             48
5            27             32
6            28             32
7            29             56
8            31             16
9            32             16
10           34             96
11           35             72
12           37             64
13           38             24
14           39              8
15           43             48
16           44             88
17           45             24
18           46             24
19           51              8
20           54             48
21           56             32
```

Query example #5: Plot in a stacked-bar figure the number of customers who bought SonyTV60LED and/or BoseSoundSystem in every zipcode that has more than 400 customers who bought these two products(either bought one of these products or the two products)

[38]:
```
SonyTV60LEDCustTrans=pd.read_sql_query(
"SELECT ZipCode , COUNT(*) as 'num_customers' FROM trans4cust \
    WHERE SonyTV60LED=1  GROUP BY ZipCode HAVING COUNT(*) > 400", engine)

BoseSoundSystemCustTrans=pd.read_sql_query(
"SELECT ZipCode , COUNT(*) as 'num_customers' FROM trans4cust \
    WHERE BoseSoundSystem=1 GROUP BY ZipCode HAVING COUNT(*) > 400", engine)
```

[39]:
```
SonyTV60LEDCustTrans
```

[39]:
```
   ZipCode  num_customers
0     2108            416
1     2109            599
2    10065            455
3    30134            773
4    30303            524
5    60616            689
```

[40]:
```
BoseSoundSystemCustTrans
```

[40]:
```
   ZipCode  num_customers
0     2109            428
1    30134            824
2    30303            472
3    60616            466
4    90033            405
```

[41]:
```
SonyTV60LEDCustTrans.ZipCode
```

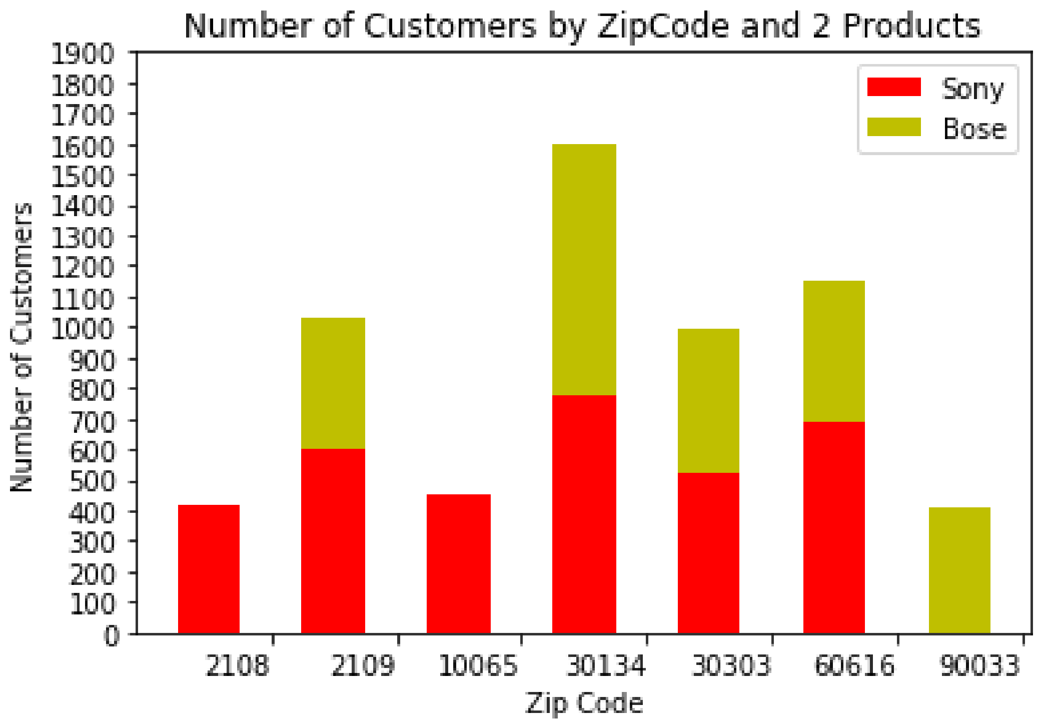
[41]:
```
0     2108
1     2109
2    10065
3    30134
4    30303
5    60616
Name: ZipCode, dtype: int64
```

[42]:
```
import numpy

#   There are zipcodes that Sony got bought but not Bose
#   but there are also zipcodes that Bose got bought but not Sony
#
#   AND we need to use stacked-bar graph and we have a potentially asymmetrical␣
 ↪set  of zipcode values
#   So, we need to do somework to create the symmteric set of zipcode values␣
 ↪for Sony and Bose
```

```
sonyZipCodeTuples=tuple(SonyTV60LEDCustTrans.ZipCode.astype(numpy.int))
sony_num_customersTuples=tuple(SonyTV60LEDCustTrans.num_customers.astype(numpy.
 ↪int))

boseZipCodeTuples=tuple(BoseSoundSystemCustTrans.ZipCode.astype(numpy.int))
bose_num_customersTuples=tuple(BoseSoundSystemCustTrans.num_customers.
 ↪astype(numpy.int))

sony_dict = dict(zip(sonyZipCodeTuples, sony_num_customersTuples))
bose_dict = dict(zip(boseZipCodeTuples, bose_num_customersTuples))

for key in bose_dict.keys():
    if ((key in sony_dict.keys()) == False): sony_dict[key]=0

for key in sony_dict.keys():
    if ((key in bose_dict.keys()) == False): bose_dict[key]=0

bose_zip= sorted(bose_dict.keys())

sony_zip= sorted(sony_dict.keys())

bose_zip_tuple=tuple(bose_zip)

sony_zip_tuple=tuple(sony_zip)

bose_customer_list=[]

for bose in bose_zip_tuple:
    bose_customer_list.append(bose_dict[bose])

sony_customer_list=[]

for sony in sony_zip_tuple:
    sony_customer_list.append(sony_dict[sony])

bose_customer_tuple=tuple(bose_customer_list)
sony_customer_tuple=tuple(sony_customer_list)
```

[43]:
```
# See docs for bar_stack at the URL
# http://matplotlib.org/examples/pylab_examples/bar_stacked.html

import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

```
ind = np.arange(len(sony_customer_tuple))

# the width of the bars: can also be len(x) sequence
width = .5

p1 = plt.bar(ind, sony_customer_tuple, width,  color='r')
p2 = plt.bar(ind, bose_customer_tuple, width, color='y',␣
 ↪bottom=sony_customer_tuple)

plt.ylabel('Number of Customers')
plt.xlabel('Zip Code')

plt.title('Number of Customers by ZipCode and 2 Products')

plt.xticks(ind + width, sony_zip_tuple, horizontalalignment='right')

plt.yticks(np.arange(0, 2000, 100))
plt.legend((p1[0], p2[0]), ('Sony', 'Bose'))

plt.show()
```



3 Requirements :

1. (Use SQL/SQlite): show the top 3 zip codes with the most customers
2. (Use SQL/SQlite): selecting the customers from the top 3 zip codes (results from question 1), what are ages of the customers? Sort output by most customers. You can show all 3 zip codes combined or show ages by zip codes.
3. (Use SQL/SQlite): get the number of customers who bought DellLaptop and HPPrinter for every Age group sorted by CustomerAge.
4. (Use SQL/SQlite): Get the list of ZipCodes where no customer bought XBOX360 (this query means NOT even a single csutomer in that zip code bought XBOX360).
5. (Use SQL/SQlite/Matplotlib): Plot in a stacked-bar figure the number of customers who bought HPLaptop and/or HPPrinter but did NOT buy WDexternalHD for every CustomerAge group that has more than 100 customers who bought these two products(either bought one of these products or the two products but didn't buy WDexternalHD).

[44]:
```
# Write your python code that meets the above requirements in this cell
# Question 1
# (Use SQL/SQlite): show the top 3 zip codes with the most customers

Ans_1 = pd.read_sql_query(
    "SELECT ZipCode, COUNT(*) as 'num_customers' \
    FROM trans4cust \
    GROUP BY ZipCode \
    ORDER BY num_customers DESC\
    Limit 3;", engine)
Ans_1
```

[44]:
```
   ZipCode  num_customers
0    30134           1165
1    30303           1001
2    60616            952
```

[45]:
```
# Question 2
# (Use SQL/SQlite): selecting the customers from the top 3 zip codes
# (results from question 1), what are ages of the customers?
# Sort output by most customers. You can show all 3 zip codes combined
# or show ages by zip codes.

Ans_2 = pd.read_sql_query(
    "SELECT ZipCode, CustomerAge, COUNT(*) as 'num_customers' \
    FROM trans4cust \
    WHERE ZipCode in (30134, 30303, 60616)\
    GROUP BY ZipCode, CustomerAge \
    ORDER BY num_customers DESC", engine)
Ans_2
```

[45]:
```
     ZipCode  CustomerAge  num_customers
0      30134           25            155
1      60616           34             96
```

2	60616	25	88
3	60616	44	88
4	30134	29	84
5	30303	26	83
6	30303	27	81
7	30303	44	77
8	30134	43	75
9	30303	29	75
10	30134	34	74
11	30303	23	73
12	60616	35	72
13	30303	43	68
14	30134	28	67
15	60616	37	64
16	30303	34	61
17	30134	32	58
18	30134	44	58
19	30134	31	56
20	30134	37	56
21	60616	21	56
22	60616	29	56
23	30303	41	49
24	30134	22	48
25	60616	26	48
26	60616	43	48
27	60616	54	48
28	30134	38	45
29	30303	25	42
..	...	...	...
47	30303	37	28
48	30303	54	28
49	30303	24	27
50	30303	31	27
51	30303	49	27
52	30303	61	27
53	30303	56	26
54	30134	51	24
55	60616	38	24
56	60616	45	24
57	60616	46	24
58	30303	33	21
59	30303	59	21
60	30134	45	19
61	60616	31	16
62	60616	32	16
63	30303	45	14
64	30303	55	14

```
65    30134          42           10
66    30134          54            8
67    60616          39            8
68    60616          51            8
69    30303          30            7
70    30303          35            7
71    30303          51            7
72    30134          26            2
73    30134          33            2
74    30134          39            2
75    30134          59            2
76    30134          46            1

[77 rows x 3 columns]
```

[46]:
```
# Question 3
# (Use SQL/SQlite): get the number of customers who bought DellLaptop
# and HPPrinter for every Age group sorted by CustomerAge.

Ans_3 = pd.read_sql_query(
"SELECT CustomerAge, COUNT(*) as 'num_customers' \
    FROM trans4cust \
    WHERE DellLaptop = 1\
    AND HPPrinter = 1\
    GROUP BY CustomerAge \
    ORDER BY CustomerAge", engine)
Ans_3
```

[46]:
```
    CustomerAge  num_customers
0            21            201
1            22            217
2            23            320
3            25             65
4            26            192
5            27            280
6            28             56
7            29            151
8            31            208
9            32            184
10           34            128
11           35            136
12           36            200
13           38             16
14           39             88
15           42             72
16           44            192
17           45             32
18           46             63
```

```
19          47          32
20          51          24
21          53          24
22          54         128
23          56         184
24          57          64
25          59          80
26          61          32
```

[47]:
```
# Question 4
# (Use SQL/SQlite): Get the list of ZipCodes where no customer bought XBOX360
# (this query means NOT even a single csutomer in that zip code bought XBOX360).

Ans_4 = pd.read_sql_query(
"SELECT ZipCode, COUNT(*) as 'num_customers' \
    FROM trans4cust \
    WHERE XBOX360 = 0\
    GROUP BY ZipCode", engine)
Ans_4
```

[47]:
```
    ZipCode  num_customers
0      2108             56
1      2109            216
2      2110             96
3     10065            168
4     30134            248
5     30303            220
6     33129             73
7     33130             40
8     44114             97
9     60532             32
10    60585             96
11    60603             88
12    60611              8
13    60616             81
14    62791              3
15    90024             16
16    90033            104
17    94102             36
18    94158            128
```

[48]:
```
# Question 5
# (Use SQL/SQlite/Matplotlib): Plot in a stacked-bar figure the number of␣
 ↪customers
# who bought HPLaptop and/or HPPrinter but did NOT buy WDexternalHD for every␣
 ↪CustomerAge
# group that has more than 100 customers who bought these two products
# (either bought one of these products or the two products but didn't buy␣
↪WDexternalHD).

Ans_5 = pd.read_sql_query(
"SELECT CustomerAge , COUNT(*) as 'num_customers' \
    FROM trans4cust \
    WHERE HPLaptop = 1 \
    AND HPPrinter = 1 \
    AND WDexternalHD = 0 \
    GROUP BY CustomerAge HAVING COUNT(*) > 100", engine)

Ans_5_HPLaptop = pd.read_sql_query(
"SELECT CustomerAge , COUNT(*) as 'num_customers' \
    FROM trans4cust \
    WHERE HPLaptop = 1 \
    AND WDexternalHD = 0 \
    GROUP BY CustomerAge HAVING COUNT(*) > 100", engine)

Ans_5_HPPrinter = pd.read_sql_query(
"SELECT CustomerAge , COUNT(*) as 'num_customers' \
    FROM trans4cust \
    WHERE HPPrinter = 1 \
    AND WDexternalHD = 0 \
    GROUP BY CustomerAge HAVING COUNT(*) > 100", engine)
```

[49]:
```
Ans_5_HPLaptop
```

[49]:
```
    CustomerAge  num_customers
0            21            200
1            22            216
2            23            462
3            25            151
4            26            214
5            27            314
6            28            126
7            29            299
8            31            210
9            32            176
10           34            186
11           35            348
12           36            191
13           42            178
14           43            104
15           44            326
16           54            149
17           56            169
```

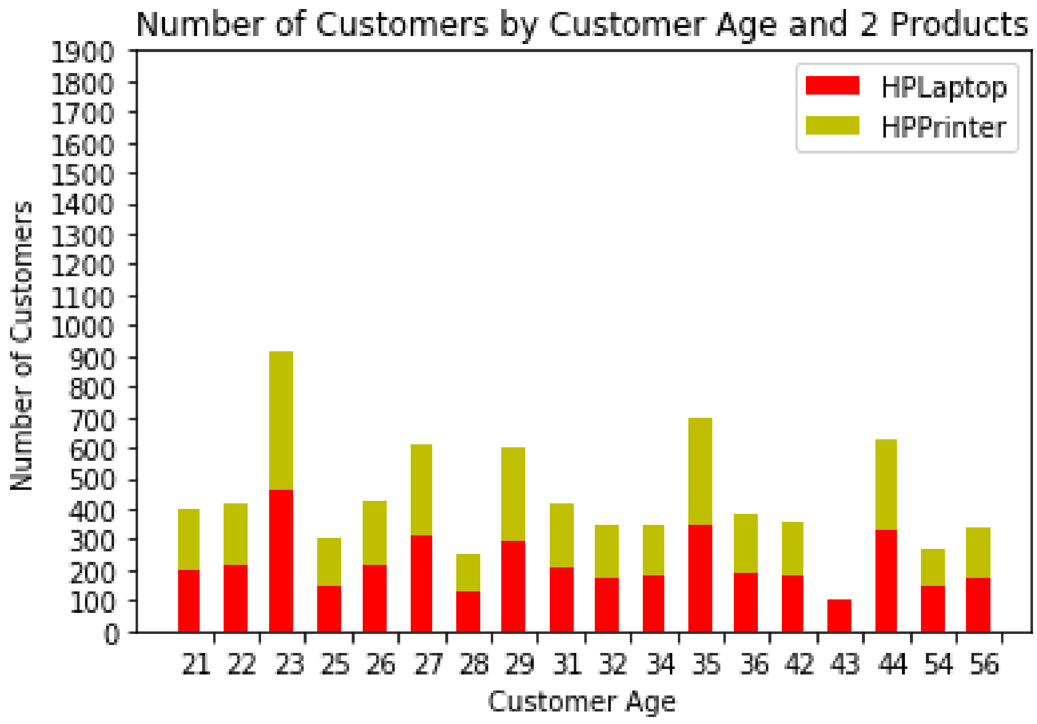
[50]:
```
Ans_5_HPPrinter
```

[50]:
```
    CustomerAge  num_customers
0            21            200
1            22            206
2            23            454
3            25            151
4            26            214
5            27            295
6            28            126
7            29            298
8            31            210
9            32            176
10           34            164
11           35            348
12           36            191
13           42            178
14           44            299
15           54            122
16           56            169
```

[51]:
```
HPLaptop_cusage = tuple(Ans_5_HPLaptop.CustomerAge.astype(numpy.int))
HPLaptop_cus = tuple(Ans_5_HPLaptop.num_customers.astype(numpy.int))

HPPrinter_cusage = tuple(Ans_5_HPPrinter.CustomerAge.astype(numpy.int))
HPPrinter_cus = tuple(Ans_5_HPPrinter.num_customers.astype(numpy.int))

HPLaptop_dict = dict(zip(HPLaptop_cusage, HPLaptop_cus))
HPPrinter_dict = dict(zip(HPPrinter_cusage, HPPrinter_cus))

for key in HPPrinter_dict.keys():
    if ((key in HPLaptop_dict.keys()) == False): HPLaptop_dict[key]=0

for key in HPLaptop_dict.keys():
    if ((key in HPPrinter_dict.keys()) == False): HPPrinter_dict[key]=0

HPPrinter_cusage= sorted(HPPrinter_dict.keys())

HPLaptop_cusage= sorted(HPLaptop_dict.keys())

HPPrinter_cusage_tuple=tuple(HPPrinter_cusage)

HPLaptop_cusage_tuple=tuple(HPLaptop_cusage)

HPPrinter_customer_list=[]

for HPPrinter in HPPrinter_cusage_tuple:
    HPPrinter_customer_list.append(HPPrinter_dict[HPPrinter])
```

```
HPLaptop_customer_list=[]

for HPLaptop in HPLaptop_cusage_tuple:
    HPLaptop_customer_list.append(HPLaptop_dict[HPLaptop])

HPPrinter_customer_tuple=tuple(HPPrinter_customer_list)
HPLaptop_customer_tuple=tuple(HPLaptop_customer_list)
```

[53]:
```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

ind = np.arange(len(HPLaptop_customer_tuple))

# the width of the bars: can also be len(x) sequence
width = .5

p1 = plt.bar(ind, HPLaptop_customer_tuple, width,  color='r')
p2 = plt.bar(ind, HPPrinter_customer_tuple, width, color='y', bottom =␣
 ↪HPLaptop_customer_tuple)

plt.ylabel('Number of Customers')
plt.xlabel('Customer Age')

plt.title('Number of Customers by Customer Age and 2 Products')

plt.xticks(ind + width, HPLaptop_cusage_tuple, horizontalalignment='right')

plt.yticks(np.arange(0, 2000, 100))
plt.legend((p1[0], p2[0]), ('HPLaptop', 'HPPrinter'))

plt.show();
```


Number of Customers by Customer Age and 2 Products
HPLaptop
HPPrinter
Number of Customers
Customer Age
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
21
22
23
25
26
27
28
29
31
32
34
35
36
42
43
44
54
56