In [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
```

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

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

dirtydata4bestdeal.head()
```

Out[3]:

	ZipCode	CustomerAge	SamsungTV46LED	SonyTV42LED	XBOX360	DellLaptop	BoseSoundSyst
0	30134.0	35.0	1	1	1	0	0
1	62791.0	43.0	0	1	0	0	1
2	60611.0	23.0	1	NaN	0	1	0
3	60616.0	56.0	0	1	1	1	0
4	30303.0	25.0	1	NaN	0	NaN	1

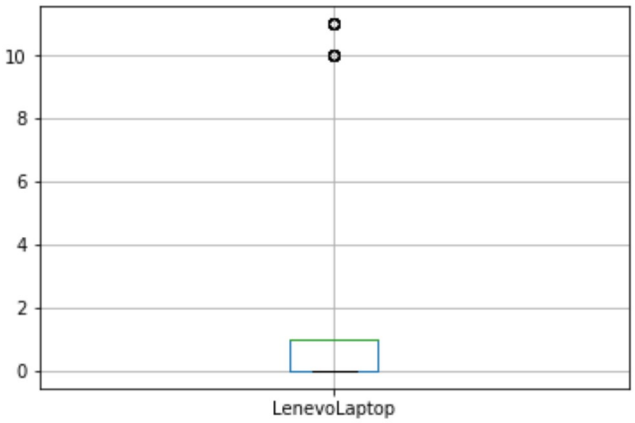
5 rows × 34 columns

In [4]:
```
dirtydata4bestdeal.boxplot(column='CustomerAge')
```

Out[4]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d9669b8828>
```

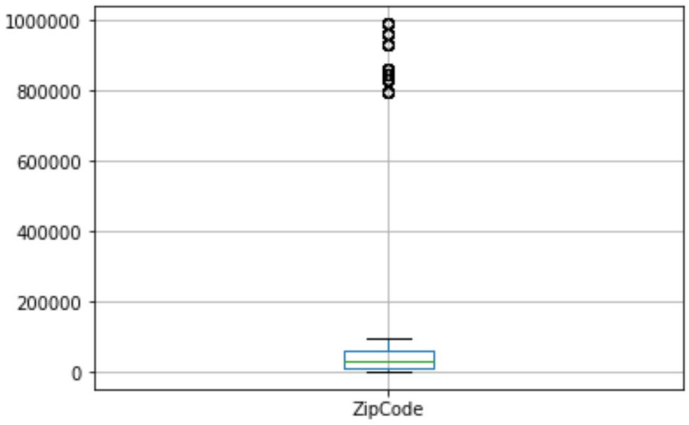
In [5]:
```
dirtydata4bestdeal.boxplot(column='LenevoLaptop')
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d966ed57b8>
```



In [6]:
```
dirtydata4bestdeal.boxplot(column='ZipCode')
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d966f24128>
```



In [7]:
```
# Drop the NaN values

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

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

Out[7]:

	ZipCode	CustomerAge	SamsungTV46LED	SonyTV42LED	XBOX360	DellLaptop	BoseSoundSyst
0	30134.0	35.0	1	1	1	0	0
1	62791.0	43.0	0	1	0	0	1
3	60616.0	56.0	0	1	1	1	0
5	2108.0	55.0	1	1	1	1	10
6	90033.0	44.0	1	1	1	1	0

5 rows × 34 columns

In [40]:
```
#cleandata4bestdeal.replace({2108:dropna(), 2109:dropna(), 2110:dropna(), 11:dropna
(), 111:dropna(), 10:dropna(), 9:dropna(), 8:dropna()})
#cleandata4bestdeal.replace({2108: drop.na, 2109: drop.na, 2110: drop.na, 11: drop.
na, 111: drop.na, 10: drop.na, 9: drop.na, 8: drop.na})
cleandata4bestdeal2 = cleandata4bestdeal.replace([2108, 2109, 2110, 11, 111, 10, 9,
8, 923, 843, 737, 727, 643, 536], np.nan)
cleandata4bestdeal2

cleandata4bestdeal3 = cleandata4bestdeal2.dropna()
cleandata4bestdeal3
```

Out[40]:

	ZipCode	CustomerAge	SamsungTV46LED	SonyTV42LED	XBOX360	DellLaptop	BoseSoundS
0	30134.0	35.0	1	1	1	0	0
1	62791.0	43.0	0	1	0	0	1
3	60616.0	56.0	0	1	1	1	0
6	90033.0	44.0	1	1	1	1	0
13	62791.0	27.0	1	1	0	1	0
16	60616.0	43.0	0	1	1	0	1
18	60616.0	54.0	1	0	0	1	0
19	60603.0	59.0	0	1	1	1	0
20	30134.0	28.0	1	1	1	0	1
21	33130.0	27.0	1	1	1	1	0
23	60616.0	43.0	1	1	1	0	1
24	30303.0	43.0	0	1	1	0	1
26	90033.0	56.0	0	1	1	1	0
28	90024.0	51.0	1	0	1	1	0
30	90024.0	37.0	0	1	1	0	1
33	30134.0	27.0	1	0	0	1	0
34	60616.0	31.0	0	1	1	1	0
36	60616.0	25.0	1	1	1	0	0
37	60603.0	37.0	0	1	1	0	1
38	30134.0	42.0	1	0	0	1	0
39	33130.0	31.0	0	1	1	1	0
40	44114.0	34.0	1	1	1	1	0
42	30303.0	29.0	1	1	1	0	0
44	90033.0	27.0	1	0	0	1	0
46	33129.0	22.0	1	0	0	1	0
49	94158.0	35.0	1	1	1	0	0
50	30303.0	34.0	0	1	1	0	1
51	30134.0	44.0	1	0	0	1	1
52	44114.0	54.0	0	1	1	1	0
53	62791.0	44.0	1	1	0	0	1
...	...	...	...	...	...	...	...
9955	44114.0	36.0	0	1	1	1	0
9956	60616.0	45.0	1	1	1	0	1
9957	30303.0	41.0	1	1	1	1	0
9959	90033.0	29.0	1	1	1	0	1
9961	33129.0	26.0	0	1	1	0	1
9964	94158.0	33.0	1	0	0	1	0

In [41]:
```
engine=create_engine('sqlite:///bestdeal.db')
```

In [42]:
```
cleandata4bestdeal3.to_sql('trans4cust', engine)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-42-dec9c7478111> in <module>()
----> 1 cleandata4bestdeal3.to_sql('trans4cust', engine)

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\generic.py in
to_sql(self, name, con, schema, if_exists, index, index_label, chunksize, dtype)
   2125         sql.to_sql(self, name, con, schema=schema, if_exists=if_exists,
   2126                    index=index, index_label=index_label, chunksize=chunk
size,
-> 2127                    dtype=dtype)
   2128
   2129     def to_pickle(self, path, compression='infer',

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\io\sql.py in to_sql
(frame, name, con, schema, if_exists, index, index_label, chunksize, dtype)
    448     pandas_sql.to_sql(frame, name, if_exists=if_exists, index=index,
    449                       index_label=index_label, schema=schema,
--> 450                       chunksize=chunksize, dtype=dtype)
    451
    452

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\io\sql.py in to_sql
(self, frame, name, if_exists, index, index_label, schema, chunksize, dtype)
   1146                           if_exists=if_exists, index_label=index_label,
   1147                           schema=schema, dtype=dtype)
-> 1148         table.create()
   1149         table.insert(chunksize)
   1150         if (not name.isdigit() and not name.islower()):

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\io\sql.py in create
(self)
    561         if self.exists():
    562             if self.if_exists == 'fail':
--> 563                 raise ValueError("Table '%s' already exists." % self.nam
e)
    564             elif self.if_exists == 'replace':
    565                 self.pd_sql.drop_table(self.name, self.schema)

ValueError: Table 'trans4cust' already exists.
```

In [43]:
```
cleandata4bestdeal3.to_sql('trans4cust', engine, if_exists='replace')
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
~\AppData\Local\Continuum\anaconda3\lib\site-packages\sqlalchemy\engine\base.py
in _execute_context(self, dialect, constructor, statement, parameters, *args)
   1192                         parameters,
-> 1193                         context)
   1194         except BaseException as e:

~\AppData\Local\Continuum\anaconda3\lib\site-packages\sqlalchemy\engine\default.
py in do_execute(self, cursor, statement, parameters, context)
    506     def do_execute(self, cursor, statement, parameters, context=None):
--> 507         cursor.execute(statement, parameters)
    508

OperationalError: too many SQL variables

The above exception was the direct cause of the following exception:

OperationalError                          Traceback (most recent call last)
<ipython-input-43-767af133abe5> in <module>()
----> 1 cleandata4bestdeal3.to_sql('trans4cust', engine, if_exists='replace')

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\generic.py in
to_sql(self, name, con, schema, if_exists, index, index_label, chunksize, dtype)
   2125         sql.to_sql(self, name, con, schema=schema, if_exists=if_exists,
   2126                    index=index, index_label=index_label, chunksize=chunk
size,
-> 2127                    dtype=dtype)
   2128
   2129     def to_pickle(self, path, compression='infer',

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\io\sql.py in to_sql
(frame, name, con, schema, if_exists, index, index_label, chunksize, dtype)
    448     pandas_sql.to_sql(frame, name, if_exists=if_exists, index=index,
    449                       index_label=index_label, schema=schema,
--> 450                       chunksize=chunksize, dtype=dtype)
    451
    452

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\io\sql.py in to_sql
(self, frame, name, if_exists, index, index_label, schema, chunksize, dtype)
   1147                           schema=schema, dtype=dtype)
   1148         table.create()
-> 1149         table.insert(chunksize)
   1150         if (not name.isdigit() and not name.islower()):
   1151             # check for potentially case sensitivity issues (GH7815)

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\io\sql.py in insert
(self, chunksize)
    661
    662                 chunk_iter = zip(*[arr[start_i:end_i] for arr in data_li
st])
--> 663                 self._execute_insert(conn, keys, chunk_iter)
    664
    665     def _query_iterator(self, result, chunksize, columns, coerce_float=T
rue,

~\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\io\sql.py in _execu
te_insert(self, conn, keys, data_iter)
    636         """Insert data into this table with database connection"""
    637         data = [{k: v for k, v in zip(keys, row)} for row in data_iter]
--> 638         conn.execute(*self.insert_statement(data, conn))
    639
    640     def insert(self, chunksize=None):
```

In [44]:
```
insp=inspect(engine)
```

In [45]:
```
 insp.get_table_names()
```

Out[45]:
```
['trans4cust']
```

In [46]:
```
pd.read_sql_table('trans4cust', engine).columns
```

Out[46]:
```
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')
```

In [47]:
```
resultsForBestDealCustTrans=pd.read_sql_query("SELECT * FROM trans4cust WHERE ZipCo
de='60616'", engine)
```

In [48]:
```
resultsForBestDealCustTrans.head()
```

Out[48]:

	index	ZipCode	CustomerAge	SamsungTV46LED	SonyTV42LED	XBOX360	DellLaptop	BoseSound

0 rows × 35 columns

In [49]:
```
resultsForBestDealCustTrans=pd.read_sql_query("SELECT * FROM trans4cust", engine)
```

In [50]:
```
resultsForBestDealCustTrans.head()
```

Out[50]:

	index	ZipCode	CustomerAge	SamsungTV46LED	SonyTV42LED	XBOX360	DellLaptop	BoseSound

0 rows × 35 columns

In [51]:
```
resultsForBestDealCustTrans=pd.read_sql_query("SELECT ZipCode , COUNT(*) as 'num_cu
stomers' FROM trans4cust GROUP BY ZipCode  ORDER BY ZipCode", engine)
```

In [52]:
```
resultsForBestDealCustTrans
```

Out[52]:

	ZipCode	num_customers

In [53]:
```
resultsForBestDealCustTrans=pd.read_sql_query(
"SELECT CustomerAge , COUNT(*) as 'num_customers' FROM trans4cust WHERE ZipCode=606
16 GROUP BY CustomerAge  ORDER BY CustomerAge", engine)
```

In [54]:
```
resultsForBestDealCustTrans
```

Out[54]:

	CustomerAge	num_customers

In [55]:
```
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)
```

In [56]:
```
SonyTV60LEDCustTrans
```

Out[56]:

	ZipCode	num_customers

In [57]:
```
BoseSoundSystemCustTrans
```

Out[57]:

	ZipCode	num_customers

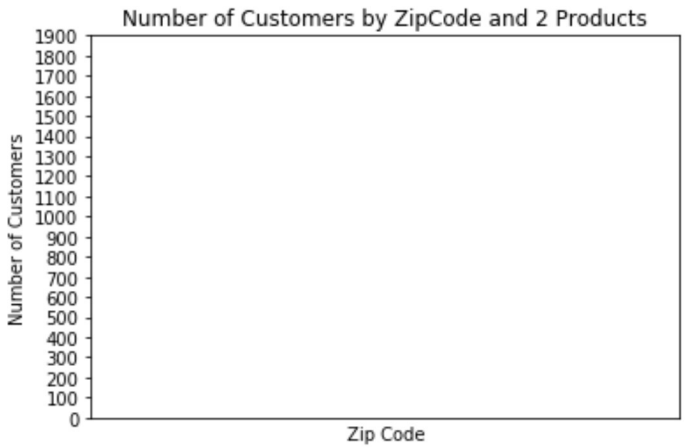
In [58]:
```
SonyTV60LEDCustTrans.ZipCode
```

Out[58]:
```
Series([], Name: ZipCode, dtype: object)
```

In [59]:
```
import numpy

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)
```

In [60]:
```
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()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-60-35172ddcc867> in <module>()
     24
     25 plt.yticks(np.arange(0, 2000, 100))
---> 26 plt.legend((p1[0], p2[0]), ('Sony', 'Bose'))
     27
     28 plt.show()

IndexError: tuple index out of range
```



In [61]:
```
#(Use SQL/SQlite): get the number of customers who bought DellLaptop and HPPrinter
for every Age group sorted
#by CustomerAge

query6=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)
query6
```

Out[61]:

	CustomerAge	num_customers

In [62]:
```
#(Use SQL/SQlite): Get the list of ZipCodes where no customer bought XBOX360 (this
query means NOT even a
#single customer in that zip code bought XBOX360)
query7=pd.read_sql_query("SELECT Zipcode FROM trans4cust WHERE XBOX360=0", engine)
query7

#List should be:
#ZipCode
#62791
#60611
#30303
#60616
#30134
#90033
#33129
#33130
#94158
#44114
#60603
#60585
#90024
#94102
#60532
#10065
```

Out[62]:

	ZipCode

In [63]:
```
#(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 produ
cts but didn't buy
#WDexternalHD)

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)
hplaptop

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)
hpprinter
```

Out[63]:

	CustomerAge	num_customers

In [64]:
```
#Last requirement continued...

hplaptopTuples=tuple(hplaptop.CustomerAge.astype(numpy.int))
hplaptop_num_customersTuples=tuple(hplaptop.num_customers.astype(numpy.int))

hpprinterTuples=tuple(hpprinter.CustomerAge.astype(numpy.int))
hpprinter_num_customersTuples=tuple(hpprinter.num_customers.astype(numpy.int))

hplaptop_dict = dict(zip(hplaptopTuples, hplaptop_num_customersTuples))
hpprinter_dict = dict(zip(hpprinterTuples, hpprinter_num_customersTuples))

for key in hplaptop_dict.keys():
    if ((key in hpprinter_dict.keys()) == False): hpprinter_dict[key]=0


for key in hpprinter_dict.keys():
    if ((key in hplaptop_dict.keys()) == False): hplaptop_dict[key]=0

hpprinter_age= sorted(hpprinter_dict.keys())

hplaptop_age= sorted(hplaptop_dict.keys())

hpprinter_age_tuple=tuple(hpprinter_age)

hplaptop_age_tuple=tuple(hplaptop_age)

hpprinter_customer_list=[]

for hpprinter in hpprinter_age_tuple:
    hpprinter_customer_list.append(hpprinter_dict[hpprinter])

hplaptop_customer_list=[]

for hplaptop in hplaptop_age_tuple:
    hplaptop_customer_list.append(hplaptop_dict[sony])

hpprinter_customer_tuple=tuple(hpprinter_customer_list)
hplaptop_customer_tuple=tuple(hplaptop_customer_list)
```

In [65]:
```
#Last requirement continued...

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_custo
mer_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, sony_zip_tuple, horizontalalignment='right')

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

plt.show()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-65-7f18609e39f6> in <module>()
     25
     26 plt.yticks(np.arange(0, 2000, 100))
---> 27 plt.legend((p1[0], p2[0]), ('HPLaptop', 'HPPrinter'))
     28
     29 plt.show()

IndexError: tuple index out of range
```