SCORE: 40

In [1]:
```
import pandas as pd  # panda's nickname is pd
import numpy as np  # numpy as np
from pandas import DataFrame, Series, Categorical
from sqlalchemy import create_engine
```

In [2]:
```
engine=create_engine('sqlite:///xyz.db')           # the db is in my current working directory
```

In [3]:
```
xyzcustnew=pd.read_sql_table('xyzcust',engine)
```

In [4]:
```
# Refer to exercise #7 how we calculated this value for xyz db
heavyCut= 423  #heavyCut is a constant
```

In [5]:
```
heavyCat=Categorical(np.where(xyzcustnew.YTD_SALES_2009>heavyCut,1,0))
heavyCat.describe()
```

Out[5]:

	counts	freqs
categories		
0	25795	0.854733
1	4384	0.145267

In [6]:
```
heavyCat.rename_categories(['regular','heavy'],inplace=True)
```

In [7]:
```
heavyCat.describe()
```

Out[7]:

	counts	freqs
categories		
regular	25795	0.854733
heavy	4384	0.145267

In [8]:
```
heavyCat[:10]
```

Out[8]:
```
[regular, heavy, regular, regular, regular, regular, heavy, regular, regular, regular]
Categories (2, object): [regular, heavy]
```

In [9]:
```
xyzcustnew['heavyCat']=heavyCat
```

In [10]:
```
buyerType=pd.get_dummies(heavyCat)
```

In [11]:
```
buyerType[:3]
```

Out[11]:

	regular	heavy
0	1	0
1	0	1
2	1	0

In [12]:
```
xyzcustnew['typeReg']=buyerType['regular']
xyzcustnew['typeHeavy']=buyerType['heavy']
```

In [13]:
```
xyzcustnew.columns
```

Out[13]:
```
Index(['index', 'ACCTNO', 'ZIP', 'ZIP4', 'LTD_SALES', 'LTD_TRANSACTIONS',
       'YTD_SALES_2009', 'YTD_TRANSACTIONS_2009', 'CHANNEL_ACQUISITION',
       'BUYER_STATUS', 'ZIP9_SUPERCODE', 'heavyCat', 'typeReg', 'typeHeavy'],
      dtype='object')
```

In [14]:
```
# for this exercises we need to create trCountsChrono object similar to what we did
in exercises #8

xyztrans=pd.read_sql('xyztrans', engine)

trandate=xyztrans.TRANDATE      # should be a Series

daystr=trandate.str[0:2]                # two digit date numbers slice

mostr=trandate.str[2:5]         # the three letter month abbreviations

yearstr=trandate.str[5:]                # four digit years
```

In [15]:
```
#create a dictionary for the months
monums={'JAN':'1', 'FEB':'2', 'MAR':'3', 'APR':'4', 'MAY':'5', 'JUN':'6', 'JUL':'7'
, 'AUG':'8', 'SEP':'9', 'OCT':'10', 'NOV':'11','DEC':'12'}
#month
monos=mostr.map(monums) # do a dict lookup for each value of mostr

transtr=yearstr+'-'+monos+'-'+daystr
```

In [16]:
```
trDateTime=pd.to_datetime(transtr)
```

In [17]:
```
trCounts=trDateTime.value_counts()
```

In [18]:
```
newIndex=pd.date_range(trCounts.index.min(),trCounts.index.max())

trCountsChrono=trCounts.reindex(index=newIndex)
```

In [19]:
```
print(trCountsChrono.head())
```

```
2009-01-01    176
2009-01-02    305
2009-01-03    365
2009-01-04    231
2009-01-05    144
Freq: D, Name: TRANDATE, dtype: int64
```

In [20]:
```
trDF=DataFrame()
```

In [21]:
```
trDF['date'] = trCountsChrono.index
trDF['transactions'] = trCountsChrono.values
```

In [22]:
```
trDF.columns
```

Out[22]:
```
Index(['date', 'transactions'], dtype='object')
```

In [23]:
```
trDF.head()
```

Out[23]:

	date	transactions
0	2009-01-01	176
1	2009-01-02	305
2	2009-01-03	365
3	2009-01-04	231
4	2009-01-05	144

In [24]:
```
trDF.dtypes
```

Out[24]:
```
date            datetime64[ns]
transactions             int64
dtype: object
```

In [25]:
```
trMed=trDF.transactions.median()                # here's the median
```

In [26]:
```
heavyLight=lambda x  : x >= trMed and 'heavy' or 'light'  # an example anon functio
n
```

In [27]:
```
trDF['vol']=trDF.transactions.map(heavyLight)   # 'vol' is the heavy/light column
```

In [28]:
```
trDF['monum']=trDF.date.dt.month        # .dt is the datetime accessor
```

In [29]:
```
trDFnd=trDF.drop('date',axis=1) # axis=1 means here a column is selected to drop
```

In [30]:
```
trDFgrouped=trDFnd.groupby(['monum','vol']).sum()
```

In [31]:
```
trDFgrouped.loc[11,'heavy']
```

Out[31]:
```
transactions    8402
Name: (11, heavy), dtype: int64
```

In [32]:

```
trDFgrouped.loc[list(range(1,7))]
```

Out[32]:

		transactions
monum	**vol**	
1	**heavy**	5255
	light	572
2	**heavy**	761
	light	1625
3	**heavy**	1130
	light	1664
4	**heavy**	2327
	light	1727
5	**heavy**	2172
	light	2076
6	**heavy**	2878
	light	1495

In [33]:

```
trDFgrouped.iloc[0:6]           # .iloc here, but .loc above.
```

Out[33]:

		transactions
monum	**vol**	
1	**heavy**	5255
	light	572
2	**heavy**	761
	light	1625
3	**heavy**	1130
	light	1664

In [34]:
```
trDFgrouped[(3,'light'):(7,'heavy')]
```

Out[34]:

		transactions
monum	**vol**	
3	**light**	1664
4	**heavy**	2327
	light	1727
5	**heavy**	2172
	light	2076
6	**heavy**	2878
	light	1495
7	**heavy**	4440

In [35]:
```
trDFgrouped[(3,'light'):6]
```

Out[35]:

		transactions
monum	**vol**	
3	**light**	1664
4	**heavy**	2327
	light	1727
5	**heavy**	2172
	light	2076
6	**heavy**	2878
	light	1495

```
In [36]: trDFgrouped.xs('light',level='vol')
```

Out[36]:

	transactions
monum	
1	572
2	1625
3	1664
4	1727
5	2076
6	1495
7	564
8	1938
9	1942
10	2241
11	49
12	257

```
In [37]: trDFgrouped.xs('light',level='vol').T           # the transpose of the above
```

Out[37]:

monum	1	2	3	4	5	6	7	8	9	10	11	12
transactions	572	1625	1664	1727	2076	1495	564	1938	1942	2241	49	257

```
In [38]: mo=trDFgrouped.index.get_level_values(0)        # the month numbers
```

```
In [39]: volType=trDFgrouped.index.get_level_values(1)   # vol
```

```
In [40]: trDFpiv=DataFrame({'month':mo,'vol': volType, 'transactions':trDFgrouped.transactio
         ns})            # data as a dict
```

```
In [41]: trDFpived=trDFpiv.pivot(index='month',columns='vol',values='transactions')
```

```
In [42]: trDFpiv['randy']=np.random.randn(len(trDFpiv))
```

```
In [43]: trDFpived2=trDFpiv.pivot(index='month',columns='vol')
```

```
In [44]: xyzdata=xyzcustnew[['BUYER_STATUS','heavyCat','CHANNEL_ACQUISITION']]
```

```
In [45]: xyzgrouped=xyzdata.groupby(['BUYER_STATUS','heavyCat','CHANNEL_ACQUISITION'])
```

```
In [46]: xyzCountData = xyzgrouped.size()         # a MultiIndexed Series of counts
```

In [47]:
```
print(xyzCountData.unstack())
```

```
CHANNEL_ACQUISITION     CB    IB    RT
BUYER_STATUS heavyCat
ACTIVE       regular   443  1112  7393
             heavy     356   703  3325
INACTIVE     regular   691  1249  7056
LAPSED       regular   372  1111  6368
```

In [48]:
```
unStackxyz=xyzCountData.unstack()             # what we had just above
```

In [49]:
```
unStackxyz.T.stack()           # .T is the transpose
```

Out[49]:

	BUYER_STATUS	ACTIVE	INACTIVE	LAPSED
CHANNEL_ACQUISITION	heavyCat			
CB	regular	443	691.0	372.0
	heavy	356	NaN	NaN
IB	regular	1112	1249.0	1111.0
	heavy	703	NaN	NaN
RT	regular	7393	7056.0	6368.0
	heavy	3325	NaN	NaN

In [50]:
```
unStackxyz.T.stack(0).unstack(1)
```

Out[50]:

heavyCat	regular			heavy		
BUYER_STATUS	ACTIVE	INACTIVE	LAPSED	ACTIVE	INACTIVE	LAPSED
CHANNEL_ACQUISITION						
CB	443	691	372	356.0	NaN	NaN
IB	1112	1249	1111	703.0	NaN	NaN
RT	7393	7056	6368	3325.0	NaN	NaN

In [51]:
```
unStackxyz.T.stack(level=['heavyCat','BUYER_STATUS'])
```

Out[51]:
```
CHANNEL_ACQUISITION  heavyCat  BUYER_STATUS
CB                   regular   ACTIVE           443.0
                               INACTIVE         691.0
                               LAPSED           372.0
                     heavy     ACTIVE           356.0
IB                   regular   ACTIVE          1112.0
                               INACTIVE        1249.0
                               LAPSED          1111.0
                     heavy     ACTIVE           703.0
RT                   regular   ACTIVE          7393.0
                               INACTIVE        7056.0
                               LAPSED          6368.0
                     heavy     ACTIVE          3325.0
dtype: float64
```

```
In [52]: unStackxyz.T.stack(level=['BUYER_STATUS','heavyCat'])
```

```
Out[52]: CHANNEL_ACQUISITION  BUYER_STATUS  heavyCat
         CB                   ACTIVE        regular      443.0
                                            heavy        356.0
                              INACTIVE      regular      691.0
                              LAPSED        regular      372.0
         IB                   ACTIVE        regular     1112.0
                                            heavy        703.0
                              INACTIVE      regular     1249.0
                              LAPSED        regular     1111.0
         RT                   ACTIVE        regular     7393.0
                                            heavy       3325.0
                              INACTIVE      regular     7056.0
                              LAPSED        regular     6368.0
         dtype: float64
```

```
In [53]: xyzcust=xyzcustnew[['BUYER_STATUS','heavyCat','LTD_SALES']].copy()
```

```
In [54]: xyzcustm=pd.melt(xyzcust,id_vars=['BUYER_STATUS','heavyCat'],var_name="LTD_SALES")
```

In [55]:

```
print(xyzcustm)
```

```
      BUYER_STATUS heavyCat  LTD_SALES   value
0         INACTIVE  regular  LTD_SALES    90.0
1           ACTIVE    heavy  LTD_SALES  4227.0
2           ACTIVE  regular  LTD_SALES   420.0
3         INACTIVE  regular  LTD_SALES  6552.0
4           ACTIVE  regular  LTD_SALES   189.0
5           ACTIVE  regular  LTD_SALES  4278.0
6           ACTIVE    heavy  LTD_SALES  1869.0
7           ACTIVE  regular  LTD_SALES    33.0
8         INACTIVE  regular  LTD_SALES   735.0
9         INACTIVE  regular  LTD_SALES   468.0
10          ACTIVE  regular  LTD_SALES   804.0
11          LAPSED  regular  LTD_SALES   219.0
12          ACTIVE    heavy  LTD_SALES  3240.0
13        INACTIVE  regular  LTD_SALES   180.0
14          ACTIVE  regular  LTD_SALES   423.0
15        INACTIVE  regular  LTD_SALES   306.0
16          LAPSED  regular  LTD_SALES  1002.0
17          ACTIVE  regular  LTD_SALES  1155.0
18          ACTIVE  regular  LTD_SALES   612.0
19          ACTIVE  regular  LTD_SALES   633.0
20        INACTIVE  regular  LTD_SALES   114.0
21          ACTIVE  regular  LTD_SALES   294.0
22        INACTIVE  regular  LTD_SALES   849.0
23        INACTIVE  regular  LTD_SALES    72.0
24          ACTIVE    heavy  LTD_SALES  3411.0
25          ACTIVE    heavy  LTD_SALES  1023.0
26          LAPSED  regular  LTD_SALES   873.0
27          ACTIVE    heavy  LTD_SALES  2778.0
28          ACTIVE    heavy  LTD_SALES  2676.0
29          LAPSED  regular  LTD_SALES   528.0
...            ...      ...        ...     ...
30149       ACTIVE  regular  LTD_SALES   861.0
30150       ACTIVE  regular  LTD_SALES   837.0
30151       ACTIVE  regular  LTD_SALES  2478.0
30152       ACTIVE  regular  LTD_SALES    84.0
30153       ACTIVE    heavy  LTD_SALES  2877.0
30154     INACTIVE  regular  LTD_SALES  1611.0
30155       LAPSED  regular  LTD_SALES  1860.0
30156       LAPSED  regular  LTD_SALES    48.0
30157       ACTIVE  regular  LTD_SALES   195.0
30158       LAPSED  regular  LTD_SALES    60.0
30159     INACTIVE  regular  LTD_SALES   252.0
30160       LAPSED  regular  LTD_SALES   594.0
30161       LAPSED  regular  LTD_SALES  1272.0
30162       ACTIVE    heavy  LTD_SALES  2184.0
30163       ACTIVE  regular  LTD_SALES   759.0
30164     INACTIVE  regular  LTD_SALES   756.0
30165       ACTIVE  regular  LTD_SALES  1365.0
30166       ACTIVE    heavy  LTD_SALES  2490.0
30167       ACTIVE    heavy  LTD_SALES   438.0
30168     INACTIVE  regular  LTD_SALES   549.0
30169       ACTIVE  regular  LTD_SALES   150.0
30170       ACTIVE  regular  LTD_SALES    93.0
30171     INACTIVE  regular  LTD_SALES   834.0
30172     INACTIVE  regular  LTD_SALES   147.0
30173       LAPSED  regular  LTD_SALES   816.0
30174       ACTIVE  regular  LTD_SALES  2736.0
30175       ACTIVE  regular  LTD_SALES  2412.0
30176     INACTIVE  regular  LTD_SALES   429.0
30177     INACTIVE  regular  LTD_SALES   651.0
30178       ACTIVE    heavy  LTD_SALES  4527.0

[30179 rows x 4 columns]
```

In [56]:
```
pd.pivot_table(xyzcustnew,values='YTD_SALES_2009',index=['BUYER_STATUS','heavyCat']
,columns=['CHANNEL_ACQUISITION'])
```

Out[56]:

	CHANNEL_ACQUISITION	CB	IB	RT
BUYER_STATUS	heavyCat			
ACTIVE	regular	205.334086	191.047662	167.993913
	heavy	2397.606742	1251.559033	1158.506165
INACTIVE	regular	0.000000	0.000000	0.000000
LAPSED	regular	0.000000	0.000000	0.000000

In [57]:
```
pd.pivot_table(xyzcustnew,values='YTD_SALES_2009',index=['BUYER_STATUS'],columns=['
heavyCat','CHANNEL_ACQUISITION'])
```

Out[57]:

heavyCat	regular			heavy		
CHANNEL_ACQUISITION	CB	IB	RT	CB	IB	RT
BUYER_STATUS						
ACTIVE	205.334086	191.047662	167.993913	2397.606742	1251.559033	1158.50616
INACTIVE	0.000000	0.000000	0.000000	NaN	NaN	NaN
LAPSED	0.000000	0.000000	0.000000	NaN	NaN	NaN

In [58]:
```
pd.pivot_table(xyzcustnew,values='YTD_SALES_2009',index=['BUYER_STATUS'],columns=['
heavyCat','CHANNEL_ACQUISITION'],aggfunc=np.sum)
```

Out[58]:

heavyCat	regular			heavy		
CHANNEL_ACQUISITION	CB	IB	RT	CB	IB	RT
BUYER_STATUS						
ACTIVE	90963.0	212445.0	1241979.0	853548.0	879846.0	3852033.0
INACTIVE	0.0	0.0	0.0	NaN	NaN	NaN
LAPSED	0.0	0.0	0.0	NaN	NaN	NaN

In [59]:
```
pd.pivot_table(xyzcustnew,values='YTD_SALES_2009',index=['BUYER_STATUS'],columns=['
heavyCat','CHANNEL_ACQUISITION'],aggfunc=np.sum,margins=True)
```

Out[59]:

heavyCat	regular			heavy			All
CHANNEL_ACQUISITION	CB	IB	RT	CB	IB	RT	
BUYER_STATUS							
ACTIVE	90963.0	212445.0	1241979.0	853548.0	879846.0	3852033.0	7130814.0
INACTIVE	0.0	0.0	0.0	NaN	NaN	NaN	0.0
LAPSED	0.0	0.0	0.0	NaN	NaN	NaN	0.0
All	90963.0	212445.0	1241979.0	853548.0	879846.0	3852033.0	7130814.0

In [60]:
```
xyzGrouper=xyzcustnew.groupby(['BUYER_STATUS','heavyCat'])
```

In [61]:

```
xyzGrouper.agg({'YTD_SALES_2009': [np.mean, np.std],'LTD_SALES':[np.mean,np.std]})
```

Out[61]:

		YTD_SALES_2009		LTD_SALES	
		mean	std	mean	std
BUYER_STATUS	heavyCat				
ACTIVE	regular	172.707532	107.584023	1001.845105	1466.075631
	heavy	1274.048130	5434.616517	4096.179745	34210.646330
INACTIVE	regular	0.000000	0.000000	568.014784	850.966479
LAPSED	regular	0.000000	0.000000	841.467329	1374.447756

In [62]:

```
def coefV(x):                      # a baby CV function that accepts a sequence
    return np.std(x)/np.mean(x)
```

In [63]:

```
buyerStats=xyzcustnew[['BUYER_STATUS','LTD_SALES','LTD_TRANSACTIONS']]
buyerGrouper=buyerStats.groupby(['BUYER_STATUS'])
buyerGrouper.agg(coefV)
```

Out[63]:

	LTD_SALES	LTD_TRANSACTIONS
BUYER_STATUS		
ACTIVE	9.758480	1.153501
INACTIVE	1.498058	0.784441
LAPSED	1.633290	0.987139

In [64]:

```
def ptiles(x):
    p5=np.percentile(x,5)
    p95=np.percentile(x,95)
    return p5, p95
```

In [65]:

```
buyerGrouper.agg([np.mean, ptiles])
```

Out[65]:

	LTD_SALES		LTD_TRANSACTIONS	
	mean	ptiles	mean	ptiles
BUYER_STATUS				
ACTIVE	2019.364086	(81.0, 6544.349999999997)	6.935794	(1.0, 20.0)
INACTIVE	568.014784	(60.0, 1776.0)	2.263895	(1.0, 6.0)
LAPSED	841.467329	(63.0, 2904.0)	3.498280	(1.0, 9.0)

In [66]:

```
buyerGrouper.agg([np.mean,ptiles]).loc['ACTIVE','LTD_SALES']
```

Out[66]:

```
mean                        2019.36
ptiles    (81.0, 6544.349999999997)
Name: ACTIVE, dtype: object
```

In [67]:

```
#Get the trDFgrouped data starting from the May heavy day counts to the August heavy counts
trDFgrouped[(5,'heavy'):(8,'heavy')]
```

Out[67]:

		transactions
monum	**vol**	
5	**heavy**	2172
	light	2076
6	**heavy**	2878
	light	1495
7	**heavy**	4440
	light	564
8	**heavy**	1682

✓

In [69]:

```
#Group xyz customers using BUYER_STATUS, heavyCat, and ZIP, and apply np.sum function on the aggregated
#data for YTD_SALES_2009 and LTD_SALES columns
xyzGrouper2=xyzcustnew.groupby(['BUYER_STATUS','heavyCat','ZIP'])
```

In [70]:

```
##Group xyz customers using BUYER_STATUS, heavyCat, and ZIP, and apply np.sum function on the aggregated
#data for YTD_SALES_2009 and LTD_SALES columns -- continued
xyzGrouper2.agg({'YTD_SALES_2009': [np.sum],'LTD_SALES':[np.sum]})
```

Serves the purpose.

But first column ZIP is better.

Out[70]:

			YTD_SALES_2009	LTD_SALES
			sum	**sum**
BUYER_STATUS	**heavyCat**	**ZIP**		
ACTIVE	**regular**	**60056**	68913.0	332196.0
		60060	68520.0	339567.0
		60061	68328.0	400569.0
		60062	141237.0	762387.0
		60064	2169.0	9129.0
		60065	1002.0	2784.0
		60067	156429.0	922680.0
		60068	140133.0	802815.0
		60069	43623.0	280686.0
		60070	24051.0	134265.0
		60071	4311.0	20112.0
		60072	2037.0	14583.0
		60073	29877.0	143901.0
		60074	72999.0	349026.0
		60076	53040.0	252438.0
		60077	39546.0	183588.0
		60078	1878.0	7410.0
		60081	16446.0	76662.0
		60083	14445.0	81954.0
		60084	39834.0	243837.0
		60085	18714.0	88857.0
		60087	13749.0	59997.0
		60088	1053.0	2538.0
		60089	100038.0	481086.0
		60090	32934.0	153108.0
		60091	178533.0	1127982.0
		60093	169671.0	1449606.0
		60094	357.0	543.0
		60096	5544.0	34929.0
		60097	5805.0	29565.0
...	**...**	**...**	...	...
LAPSED	**regular**	**60064**	0.0	3537.0
		60065	0.0	7359.0
		60067	0.0	682167.0
		60068	0.0	571056.0