GNB3

September 27, 2022

1 How much are top merchants spending per order? Are we retaining these top buyers?

To accomplish in this notebook- - Define "Top buyer" - They come every month? They have spent the most money? They spend a certain amount every month? - Recommend definition of "top buyers"

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime, timedelta
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.metrics import silhouette_score
baskets = pd.read_csv('new_baskets_full.csv')
baskets['datetime'] = baskets['placed_at'].apply(lambda x: datetime.
 ↪fromisoformat(x))

#pandas.Series.dt is an interface on a pandas series that gives you convenient␣
 ↪access to operations on data stored as a pandas datetime.
baskets['date'] = baskets['datetime'].dt.date
baskets['year'] = baskets['datetime'].dt.year
baskets['month'] = baskets['datetime'].dt.month
baskets['day'] = baskets['datetime'].dt.day
baskets['hour'] = baskets['datetime'].dt.hour
baskets['weekday'] = baskets['datetime'].dt.weekday
```

behaviors and reasons to incentivise top buyers, look at future behavior, look into loyalty index. What are the hallmarks of top purchaers? How can we encourage more? How can we encourage the ones we have?

Start by finding top buyers in terms of total amount spent.

[3]:
```
baskets['spent'] = baskets['price'] * baskets['qty']
```

[4]:
```
baskets.describe()
```

[4]:
```
                  id       order_id   merchant_id         sku_id  \
count  336472.000000  336472.000000  336472.000000  336472.000000
mean   168236.500000   29079.405656     798.592706     525.308685
std     97131.244225   18909.738357     550.271799     304.262943
min         1.000000       1.000000       1.000000       1.000000
25%     84118.750000   11485.000000     352.000000     322.000000
50%    168236.500000   28436.000000     664.000000     438.000000
75%    252354.250000   46193.250000    1217.000000     589.000000
max    336472.000000   62048.000000    2138.000000    1617.000000

          top_cat_id     sub_cat_id           qty          price  \
count  336461.000000  336461.000000  3.364720e+05  3.364720e+05
mean       10.319098      45.395065  3.789684e+01  1.378956e+05
std         7.906257      27.767388  1.035873e+04  1.744689e+05
min         1.000000       1.000000  1.000000e+00  4.375000e-02
25%         4.000000      27.000000  1.000000e+00  4.600000e+04
50%         8.000000      43.000000  2.000000e+00  1.070000e+05
75%        14.000000      69.000000  5.000000e+00  1.845000e+05
max        33.000000      96.000000  4.800000e+06  5.875000e+07

                year          month            day           hour  \
count  336472.000000  336472.000000  336472.000000  336472.000000
mean     2021.539941       6.659588      15.970758      12.702486
std         0.498403       3.932984       8.796420       4.228485
min      2021.000000       1.000000       1.000000       0.000000
25%      2021.000000       3.000000       8.000000      10.000000
50%      2022.000000       7.000000      16.000000      12.000000
75%      2022.000000      11.000000      24.000000      15.000000
max      2022.000000      12.000000      31.000000      23.000000

             weekday         spent
count  336472.000000  3.364720e+05
mean        2.620928  5.346555e+05
std         1.831302  2.859301e+06
min         0.000000  5.000000e+00
25%         1.000000  1.165000e+05
50%         3.000000  2.080000e+05
75%         4.000000  4.350000e+05
max         6.000000  3.831222e+08
```

Remove first 5 months of data from dataframe.

[5]:
```
baskets.date.min()
```

[5]:
```
datetime.date(2021, 4, 9)
```

[6]:
```
baskets = baskets[baskets['date'] > pd.to_datetime('2021-7-31').date()]
```

[7]:
```
baskets.date.min()
```

[7]:
```
datetime.date(2021, 8, 1)
```

Find total amount spent by each merchant and sort highest to lowest.

[8]:
```
baskets.groupby('merchant_id').spent.sum().sort_values(ascending = False)
```

[8]:
```
merchant_id
664     3.947079e+09
441     1.729729e+09
366     1.724671e+09
122     1.491349e+09
430     1.441119e+09
            …
1158    3.575000e+05
1931    3.450000e+05
2018    3.290000e+05
411     3.190000e+05
1157    2.459000e+05
Name: spent, Length: 2128, dtype: float64
```

Notice that the top 5 most spending merchants are 664,441,366,122,430. Lets plot their spending over time to see if they are consistent spenders.

[9]:
```
baskets['month_order'] = ((baskets['year']-2021)*12) + baskets['month'] -7
```

Created month_order to reindex data to start at end of july

[10]:
```
baskets.sample(10)
```

[10]:
```
            id  order_id               placed_at  merchant_id  sku_id  \
284062  284215     51538 2022-05-08 06:04:45.344          122     184
236327  238141     43527 2022-03-05 11:19:00.555         1774     970
232437  232486     42817 2022-03-01 07:29:26.233         1842     390
297651  297766     54403 2022-05-31 14:21:23.503          866     859
201207  201439     35943 2022-02-07 13:45:29.681          644     355
85307    85488     11890 2021-11-10 13:18:58.575          490     532
65311    65423      8471 2021-10-28 21:13:57.355          606     390
330874  330269     60788 2022-08-03 18:29:29.923         1503     438
315183  315318     57365 2022-07-03 15:16:06.260         1295     227
116247  116489     17998 2021-12-01 11:16:56.683         1096     202

        top_cat_id  sub_cat_id  qty    price                datetime  \
284062         1.0        12.0   10  48500.0 2022-05-08 06:04:45.344
236327         4.0        28.0    3  93000.0 2022-03-05 11:19:00.555
```

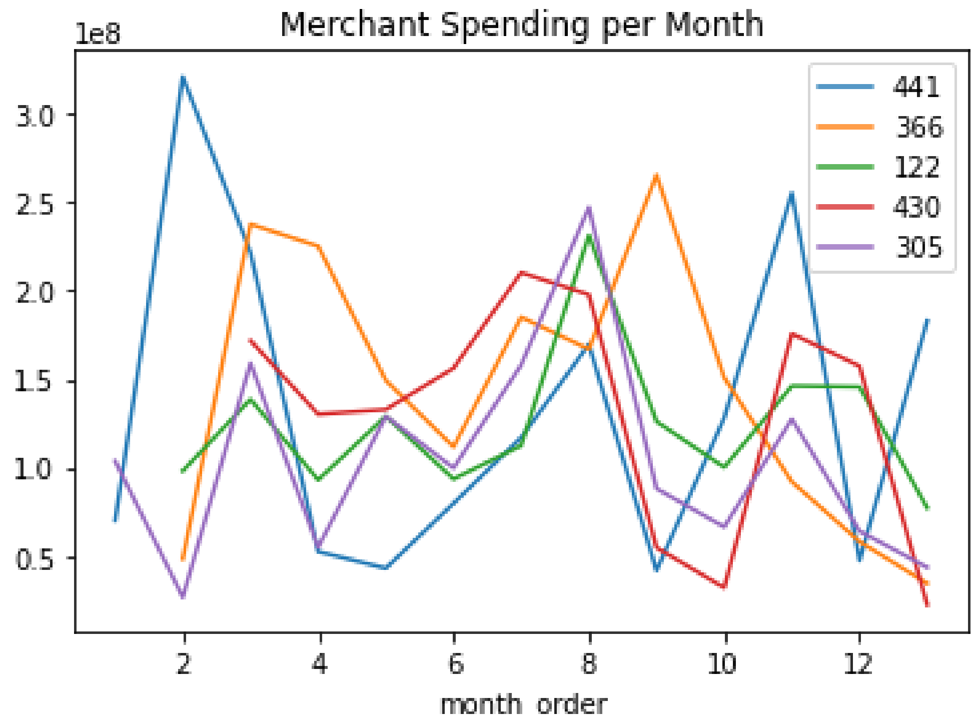
```
232437          4.0         51.0    96    13250.0 2022-03-01 07:29:26.233
297651         14.0         80.0     3    25000.0 2022-05-31 14:21:23.503
201207         29.0          5.0     3    35500.0 2022-02-07 13:45:29.681
85307           8.0         43.0     2    46000.0 2021-11-10 13:18:58.575
65311           4.0         51.0    16    17500.0 2021-10-28 21:13:57.355
330874          3.0         70.0     2   238000.0 2022-08-03 18:29:29.923
315183         28.0         16.0     1    58000.0 2022-07-03 15:16:06.260
116247          3.0          9.0     5    13000.0 2021-12-01 11:16:56.683

              date  year  month  day  hour  weekday      spent  month_order
284062  2022-05-08  2022      5    8     6        6   485000.0           10
236327  2022-03-05  2022      3    5    11        5   279000.0            8
232437  2022-03-01  2022      3    1     7        1  1272000.0            8
297651  2022-05-31  2022      5   31    14        1    75000.0           10
201207  2022-02-07  2022      2    7    13        0   106500.0            7
85307   2021-11-10  2021     11   10    13        2    92000.0            4
65311   2021-10-28  2021     10   28    21        3   280000.0            3
330874  2022-08-03  2022      8    3    18        2   476000.0           13
315183  2022-07-03  2022      7    3    15        6    58000.0           12
116247  2021-12-01  2021     12    1    11        2    65000.0            5
```

Created options to parse through top 5 spenders

[11]:
```
options = [441,366,122,430, 305]
```

[12]:
```
for i in range(5):
    baskets[baskets['merchant_id']==options[i]].groupby('month_order')['spent'].
 ↪sum().plot(title = 'Merchant Spending per Month').legend([441,366,122,430,␣
 ↪305],fancybox = True)
```



1.1 Let's compare spending in the last 6 months. Will the same buyers be in the list?

[13]:
```
sixbaskets = baskets[baskets['month_order'] > 6]
```

[14]:
```
sixbaskets.head()
```

[14]:

	id	order_id	placed_at	merchant_id	sku_id	\
191588	191782	33178	2022-02-01 04:59:15.772	1425	378	
191589	191783	33178	2022-02-01 04:59:15.772	1425	380	
191590	191784	33178	2022-02-01 04:59:15.772	1425	383	
191591	191749	33923	2022-02-01 06:53:29.594	1702	511	
191592	191750	33923	2022-02-01 06:53:29.594	1702	522	

	top_cat_id	sub_cat_id	qty	price	datetime	\
191588	8.0	43.0	1	100500.0	2022-02-01 04:59:15.772	
191589	8.0	43.0	1	100500.0	2022-02-01 04:59:15.772	
191590	8.0	43.0	1	100500.0	2022-02-01 04:59:15.772	
191591	3.0	91.0	1	104000.0	2022-02-01 06:53:29.594	
191592	3.0	94.0	10	176000.0	2022-02-01 06:53:29.594	

	date	year	month	day	hour	weekday	spent	month_order
191588	2022-02-01	2022	2	1	4	1	100500.0	7

```
191589  2022-02-01  2022      2    1     4        1   100500.0            7
191590  2022-02-01  2022      2    1     4        1   100500.0            7
191591  2022-02-01  2022      2    1     6        1   104000.0            7
191592  2022-02-01  2022      2    1     6        1  1760000.0            7
```

[15]:
```python
sixbaskets.groupby('merchant_id').spent.sum().sort_values(ascending = False)
```

[15]:
```
merchant_id
497     959077900.0
366     952641100.0
441     941911050.0
122     938894750.0
2084    932772739.0
           …
484        137500.0
195        123000.0
643         99500.0
706         62000.0
1435        53500.0
Name: spent, Length: 1729, dtype: float64
```

Top spenders are merchant_id's 497,366,441,122,2048.

441,366,122,430,305 were the top spenders over the year, it seems that 497 and 2048 have spent a lot more in the second half, while 430 and 305 have not sustained their high spending as much in the second half.

Found that 2084 is an outlier, don't know where code went but need to drop.

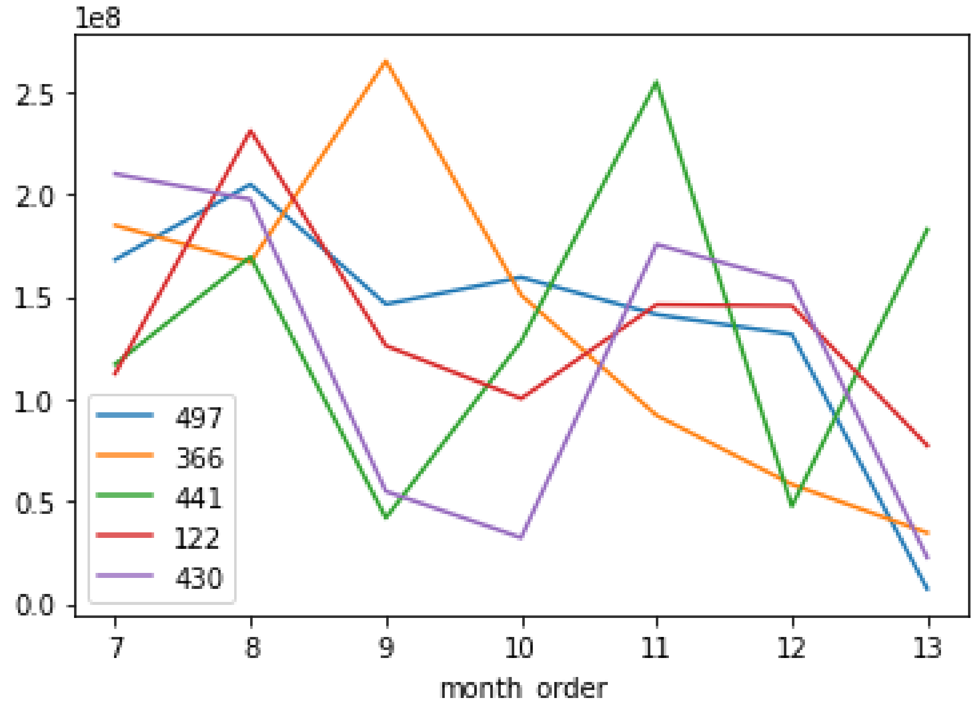
[16]:
```python
sixbaskets = sixbaskets.drop((sixbaskets[sixbaskets['merchant_id']==2084]).
 ↪index)
```

[17]:
```python
sixbaskets.groupby('merchant_id').spent.sum().sort_values(ascending = False)
```

[17]:
```
merchant_id
497     959077900.0
366     952641100.0
441     941911050.0
122     938894750.0
430     850189675.0
           …
484        137500.0
195        123000.0
643         99500.0
706         62000.0
1435        53500.0
Name: spent, Length: 1728, dtype: float64
```

[18]:
```
options = [497,366,441,122,430]
```

[19]:
```
for i in range(5):
    sixbaskets[sixbaskets['merchant_id']==options[i]].
 ↪groupby('month_order')['spent'].sum().plot().
 ↪legend([497,366,441,122,430],fancybox = True)
```



Looks like all sales are declining in July of 2022 besides 441. Why is this? 441 seems to have a regular pattern of large orders every two months. They are consistent, which is valuable. There are not any outliers, so let's look at the sales from 430 and 305 in the second half to see if they are still top buyers.

Looking at this new graph, I see no outliers, everyone is still purchasing now semi-consitently. For the monetary score, I am going to use the data from the whole year. This is because changes in the last 6 months may be more relevant to the frequency score. Top 5 are: 1. 441 (5) 2. 366 (4) 3. 122 (3) 4. 430 (2) 5. 305 (1)

RFM- recency, frequency, monetary. Find matrix to rank buyers by all three. R+F+M = Score. Recency is not as relevant

1.2 Next, I am going to work on frequency: The angle I will take to rank frequency is whoever has the highest number of orders.

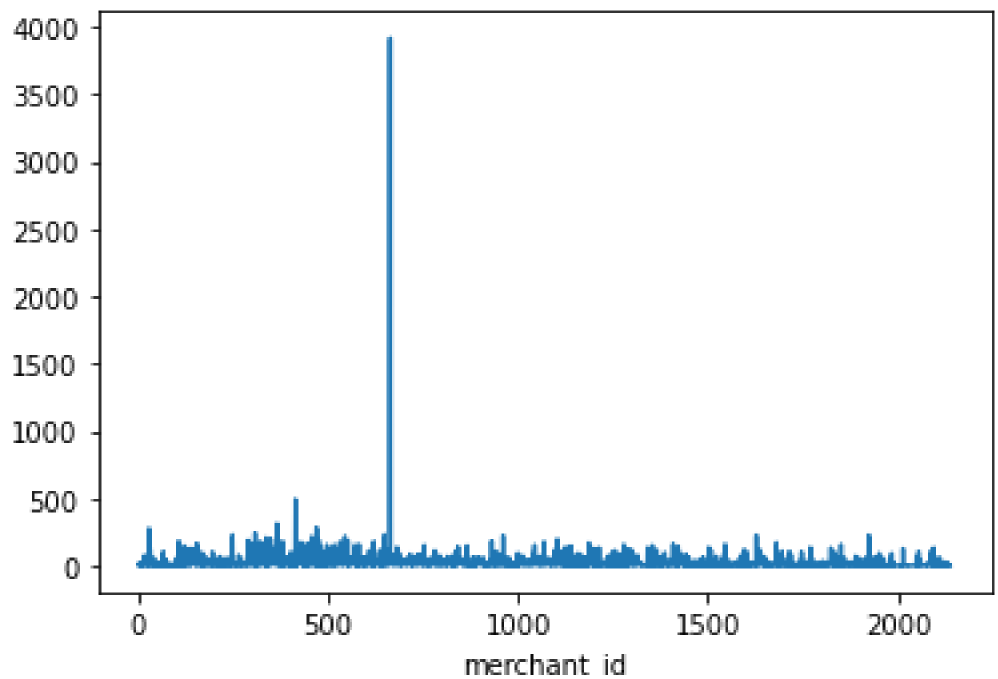
for each merchant, total their orders per month.

[20]:
```
baskets.groupby(['merchant_id']).order_id.nunique().sort_values(ascending =␣
↪False)
```

[20]:
```
merchant_id
664    3918
414     497
366     315
470     284
29      279
       …
47        1
114       1
42        1
95        1
37        1
Name: order_id, Length: 2128, dtype: int64
```

[21]:
```
baskets.groupby(['merchant_id']).order_id.nunique().plot()
```

[21]: <AxesSubplot:xlabel='merchant_id'>



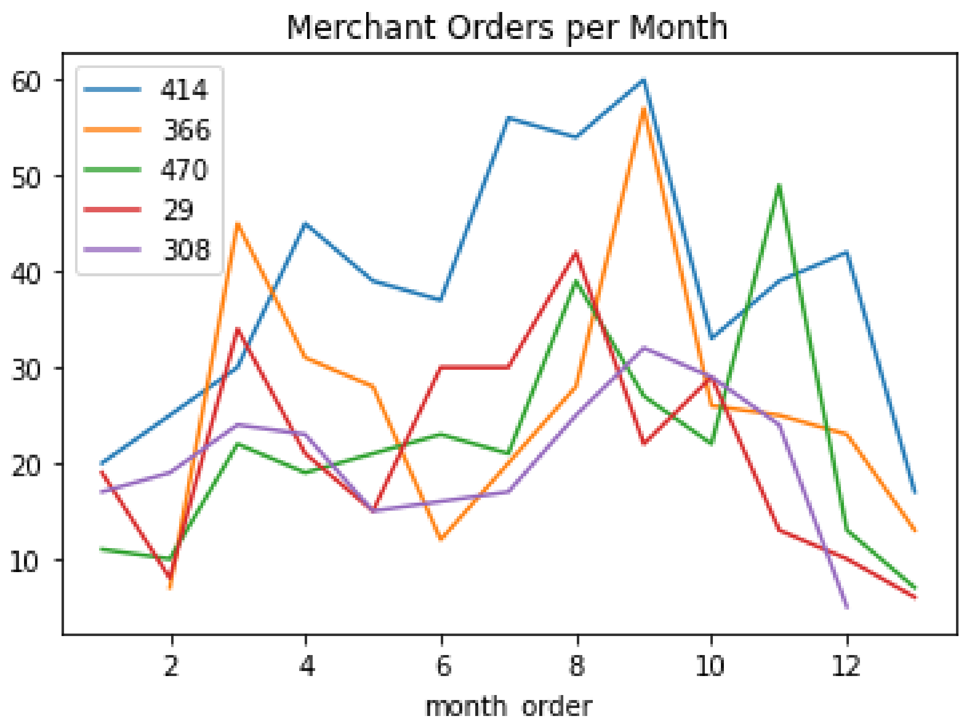
Plot to show how many purchases each merchant has made over the year. Notice how much 414 stands out. They are far and away the most frequent purchaser.

Top orderers are 1. 414 (5) 2. 366 (4) 3. 470 (3) 4. 29 (2) 5. 308 (1)

Let's look at their order habits over time.

[22]:
```
options = [414,366,470,29,308]
```

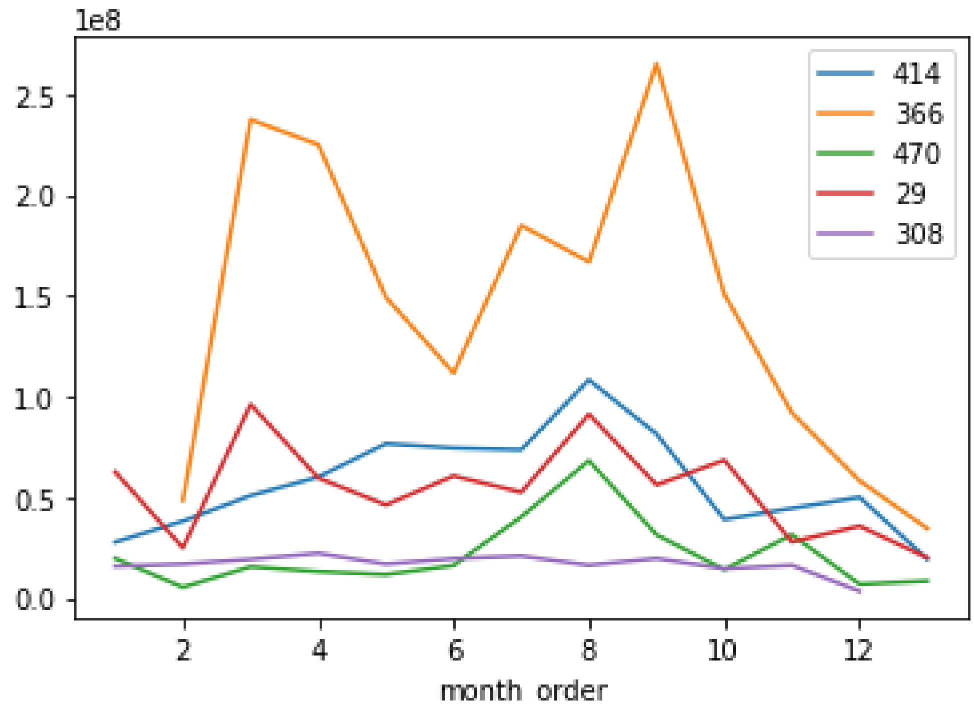
[23]:
```
for i in range(5):
    baskets[baskets['merchant_id']==options[i]].
↪groupby('month_order')['order_id'].nunique().plot(title = 'Merchant Orders␣
↪per Month').legend([414,366,470,29,308],fancybox = True)
```



It looks like all merchants have consistently made orders over the year, following a generally consistent pattern. I notice that the orders drop a bit at the end- is this because of a holiday? or covid rising again over the summer?

Lets plot what their spending habits are.

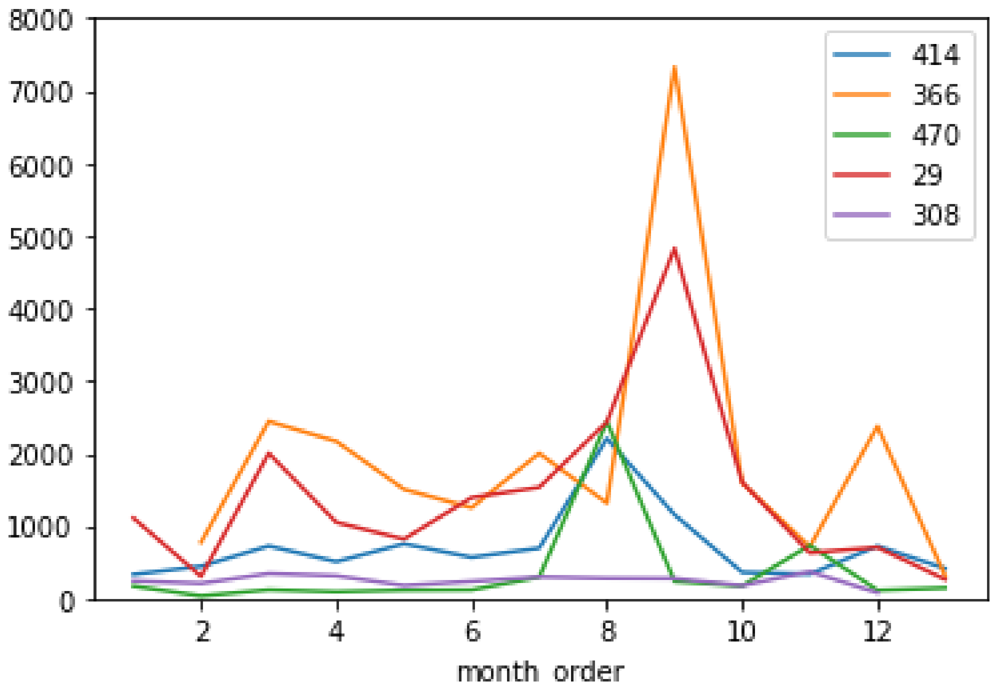
[24]:
```
for i in range(5):
    baskets[baskets['merchant_id']==options[i]].groupby('month_order')['spent'].
↪sum().plot().legend([414,366,470,29,308],fancybox = True)
```



366 is spending quit a bit over the year. This makes sense because they were also on the list for top spenders.

I wonder how many items they are buying? Are their orders large or small?

[25]:
```
for i in range(5):
    baskets[baskets['merchant_id']==options[i]].groupby('month_order')['qty'].
 ↪sum().plot(ylim = (0, 8000)).legend([414,366,470,29,308],fancybox = True)
```



All of the merchants spiked in their items purchased in March through april. Their spending also spiked during this time. I predicted that this would not happen because of the Ramadon holiday. Maybe they are buying more because people want to not go far from home to shop so corner stores have large stock?

1.2.1 I've been noticing that a lot of the significant statistics are with merchants with low merchant id's compared to the median. Could this be becuase they assign merchant id's in the order in which they start buying? This would make sense because higher merchant id's wouldnt have the same amount of time to biuld up profiles that would spend a lot of money. In another year, some of the bigger merchants may be in the 1-2 thousands. Let's test just the last month for frequency and overall spending to see if there are higher merchant id's in the running, which would support my hypothesis.

[26]:
```
onemonth = baskets[baskets['month_order'] > 12]
```

[27]:
```
onemonth.groupby('merchant_id').spent.sum().sort_values(ascending = False)
```

[27]:
```
merchant_id
441     182885500.0
122      77248250.0
1780     70122500.0
1770     55285500.0
2095     49160500.0
```

```
        …
1450       417500.0
864        360000.0
460        189400.0
1711       172000.0
964         40000.0
Name: spent, Length: 271, dtype: float64
```

[28]:
```
onemonth.groupby(['merchant_id']).order_id.nunique().sort_values(ascending =␣
↪False)
```

[28]:
```
merchant_id
2095    29
1924    26
1681    20
1889    19
249     19
        ..
1377     1
1291     1
1284     1
1262     1
1357     1
Name: order_id, Length: 271, dtype: int64
```

As I predicted, there has been a sharp rise of higher merchant id order numbers in the last month. Many merchants with ids in the thousands have broken into the top 5 both for spending and for frequency. This doesn't affect our overall findings, but note this caveat in the poster. Our top spenders now will probably not continue forever with the new merchants spending a lot.

Now, returning to the RFM ranking, my results are: 1. 366 (8) 2. 414 (5) 3. 441 (5) 4. 470 (3) 5. 122 (3)

I gave prefferential treatment to top spenders rather than frequent purchasers, since to a for-profit business, money spent is the ultimate measure.

I think a better way to estimate the value of a customer to the company would be some statistics for a specific merchant over thier time as a customer. I am going to make a merchant table to try to do this.

[29]:
```
merchants = baskets.groupby(['merchant_id']).agg({'spent': 'sum', 'order_id':␣
↪'nunique', 'date': 'nunique', 'sku_id':'nunique', 'top_cat_id':'nunique',␣
↪'sub_cat_id':'nunique'}).reset_index()
```

[30]:
```
merchants.sample(10)
```

[30]:

	merchant_id	spent	order_id	date	sku_id	top_cat_id	sub_cat_id
1038	1049	106184046.0	43	37	182	23	55
1256	1267	314206000.0	56	41	198	21	46

```
1130          1141     5509500.0        8      8      32            12            18
192            201    75558300.0       39     26      87            17            35
33              35   133400800.0       36     31     254            25            58
1470          1481     5416500.0        5      4      11             4             8
1301          1312     2642500.0        4      3      14             4             6
1026          1037    66488500.0       15     12      69            12            31
944            955   180998150.0       64     42      70            12            25
365            375   696597250.0      142    131     166            23            44
```

How can we use this to find the top merchants over time? I want to find their total spending and divide it over the time they've been a customer. lets make a new dataframe to better prepare for this.

[31]:
```python
dfmerch = baskets.groupby(['merchant_id']).agg({'spent': 'sum', 'month_order':
 ↪'min', 'order_id':'nunique'}).reset_index()
```

[32]:
```python
dfmerch.sample(10)
```

[32]:

	merchant_id	spent	month_order	order_id
137	146	326808000.0	6	28
1541	1552	8846500.0	6	4
1696	1707	16302400.0	7	17
1193	1204	11492000.0	4	5
160	169	20229250.0	3	6
408	419	134125000.0	3	75
209	218	7107000.0	3	10
548	559	227319400.0	1	67
2092	2103	31621050.0	11	51
972	983	98611060.0	3	24

Now lets use this to make a row that shows the months that they've been a customer.

[33]:
```python
dfmerch['loyalty'] = 14 - dfmerch['month_order']
```

[34]:
```python
dfmerch.sample(10)
```

[34]:

	merchant_id	spent	month_order	order_id	loyalty
356	366	1.724671e+09	2	315	12
1775	1786	8.577000e+06	7	2	7
1949	1960	7.912500e+06	8	9	6
1434	1445	1.621000e+07	4	19	10
1886	1897	1.071900e+07	7	8	7
1107	1118	1.417850e+07	3	3	11
1037	1048	2.826602e+08	3	53	11
876	887	6.622500e+06	3	5	11
1261	1272	1.642430e+07	3	12	11
1565	1576	2.392478e+07	6	16	8

Now we can use this data to rank merchants in terms of money spent over the months they've been a customer.

[35]:
```
dfmerch['avgspent'] = dfmerch['spent'] / dfmerch['loyalty']
```

[36]:
```
dfmerch.sample(10)
```

[36]:

	merchant_id	spent	month_order	order_id	loyalty	avgspent
790	801	48159150.0	2	29	12	4.013262e+06
1825	1836	477399500.0	7	15	7	6.819993e+07
92	100	6705500.0	3	7	11	6.095909e+05
1433	1444	28539500.0	4	28	10	2.853950e+06
342	352	29835650.0	3	32	11	2.712332e+06
2061	2072	1525000.0	9	3	5	3.050000e+05
1409	1420	168962397.0	4	100	10	1.689624e+07
1832	1843	58377000.0	9	61	5	1.167540e+07
2091	2102	9077650.0	12	15	2	4.538825e+06
949	960	21492000.0	4	7	10	2.149200e+06

[37]:
```
dfmerch.sort_values(by = 'avgspent', ascending = False).head(10)
```

[37]:

	merchant_id	spent	month_order	order_id	loyalty	avgspent
653	664	3.947079e+09	2	3918	12	3.289233e+08
2073	2084	9.327727e+08	9	3	5	1.865545e+08
10	12	4.872730e+08	11	6	3	1.624243e+08
356	366	1.724671e+09	2	315	12	1.437226e+08
430	441	1.729729e+09	1	157	13	1.330561e+08
419	430	1.441119e+09	3	159	11	1.310108e+08
113	122	1.491349e+09	2	149	12	1.242791e+08
295	305	1.368091e+09	1	126	13	1.052377e+08
30	32	1.213518e+09	2	125	12	1.011265e+08
638	649	1.078626e+09	3	231	11	9.805689e+07

This is interesting- we have new contenders at the top. Looking back, the top 3 are outliers that have been identified in the past. let's drop all three.

[38]:
```
dfmerch = dfmerch.drop((dfmerch[dfmerch['merchant_id']==664]).index)
dfmerch = dfmerch.drop((dfmerch[dfmerch['merchant_id']==2084]).index)
dfmerch = dfmerch.drop((dfmerch[dfmerch['merchant_id']==12]).index)
```

[39]:
```
dfmerch.sort_values(by = 'avgspent', ascending = False).head(10)
```

[39]:

	merchant_id	spent	month_order	order_id	loyalty	avgspent
356	366	1.724671e+09	2	315	12	1.437226e+08
430	441	1.729729e+09	1	157	13	1.330561e+08
419	430	1.441119e+09	3	159	11	1.310108e+08
113	122	1.491349e+09	2	149	12	1.242791e+08
295	305	1.368091e+09	1	126	13	1.052377e+08

30	32	1.213518e+09	2	125	12	1.011265e+08
638	649	1.078626e+09	3	231	11	9.805689e+07
312	322	1.266263e+09	1	179	13	9.740484e+07
486	497	1.239717e+09	1	157	13	9.536284e+07
2084	2095	3.539850e+08	10	136	4	8.849625e+07

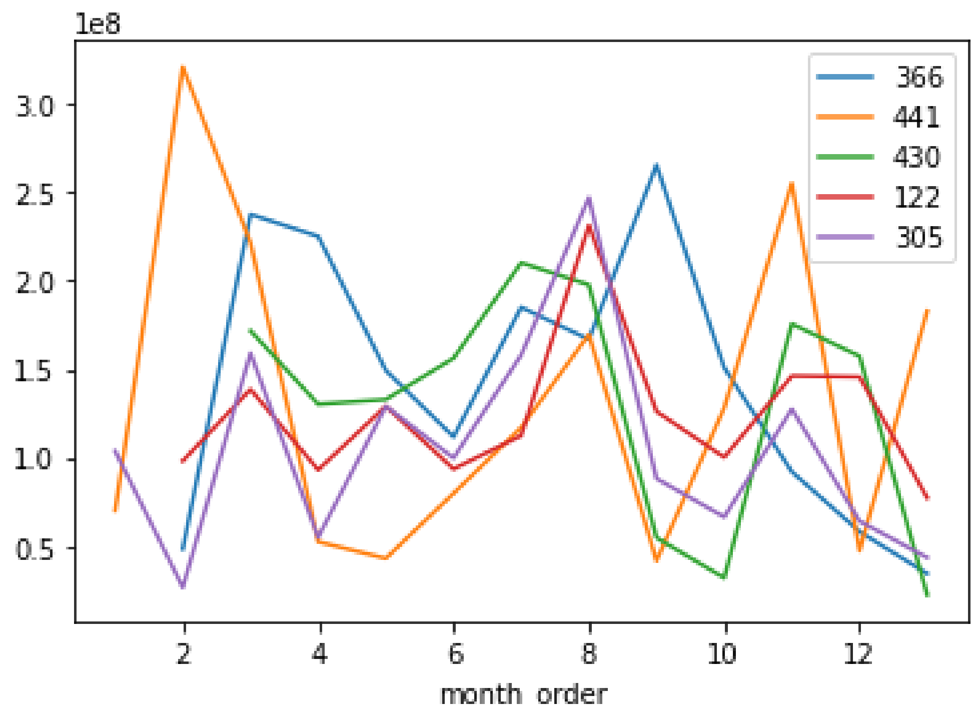
Now we can see that if we are not including how long a merchant has been a customer that we have a new top 5 spenders.

They are: 1. 366 2. 441 3. 430 4. 122 5. 305

Re-run top 5.

[40]:
```
options = [366, 441, 430, 122, 305]
```

[41]:
```
for i in range(5):
    baskets[baskets['merchant_id']==options[i]].groupby('month_order')['spent'].
↪sum().plot().legend([366, 441, 430, 122, 305],fancybox = True)
```

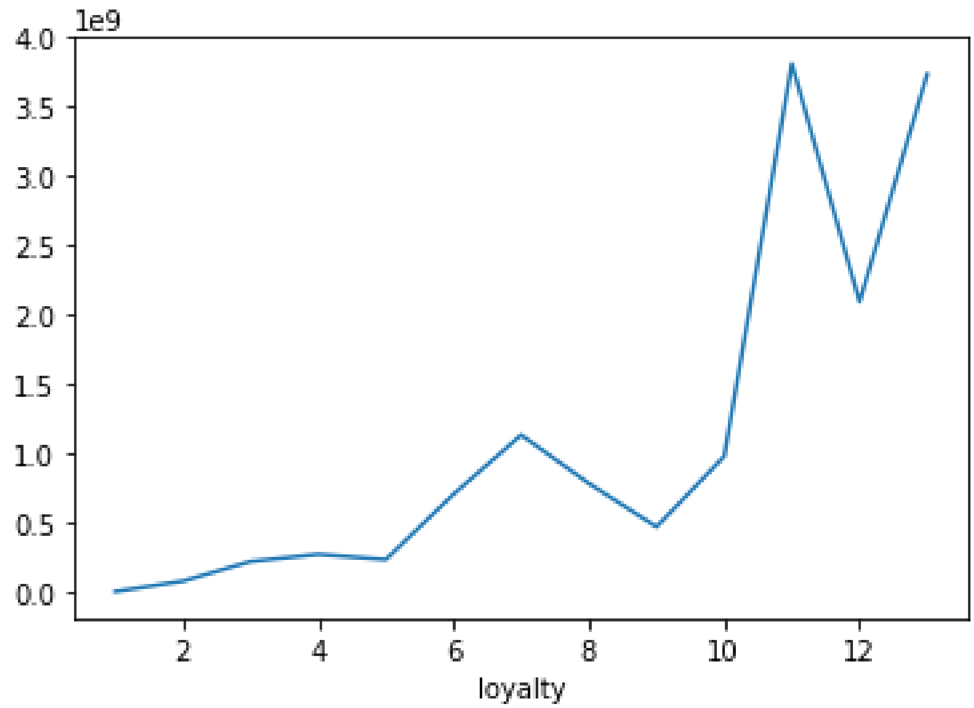


Each merchant in this plot looks like they are not outliers. The top five for this method are: 1. 366 2. 441 3. 430 4. 122 5. 305

These two models for finding the top 5 reconcile pretty well with each other. 3/5 of the top 5 match, and 366 is the top customer in both. I feel that the second model represents a more fair analysis because all of the customers seem to have been loyal for a long time, so we are not ignoring new customers. These top 5 are the biggest spenders over all time, but also over each month they've purchased from us on average.

```
[42]: dfmerch.groupby(by ='loyalty')['avgspent'].sum().plot()
```

```
[42]: <AxesSubplot:xlabel='loyalty'>
```



Wow- from how this graph looks, the customers that have been with us longer clearly spend more per month. This could be because there are more of them- let's try to filter this variable out.

How can I remove factor of merchants per loyalty month? Take average spending of all merchants over each month and plot.

```
[43]: dfmerch.groupby(by ='loyalty')['merchant_id'].nunique()
```

```
[43]: loyalty
1        3
2       18
3       22
4       25
5       44
6      167
7      225
8      157
9      120
```

```
10    243
11    651
12    232
13    218
Name: merchant_id, dtype: int64
```

[44]:
```
df1 = dfmerch.groupby(by ='loyalty').sum().avgspent
```

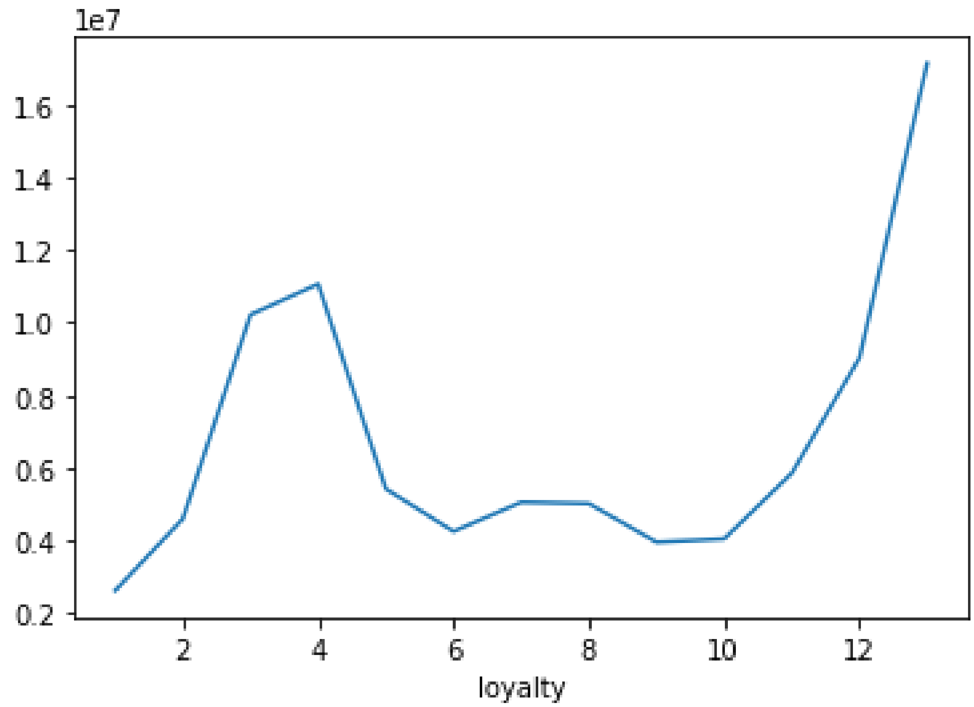
[45]:
```
df2 = dfmerch.groupby(by ='loyalty').nunique().merchant_id
```

[46]:
```
df1/df2
```

[46]:
```
loyalty
1     2.618000e+06
2     4.595447e+06
3     1.021592e+07
4     1.106171e+07
5     5.422363e+06
6     4.248987e+06
7     5.054014e+06
8     5.017685e+06
9     3.953830e+06
10    4.043968e+06
11    5.856158e+06
12    9.033538e+06
13    1.714170e+07
dtype: float64
```

[47]:
```
(df1/df2).plot()
```

[47]:
```
<AxesSubplot:xlabel='loyalty'>
```

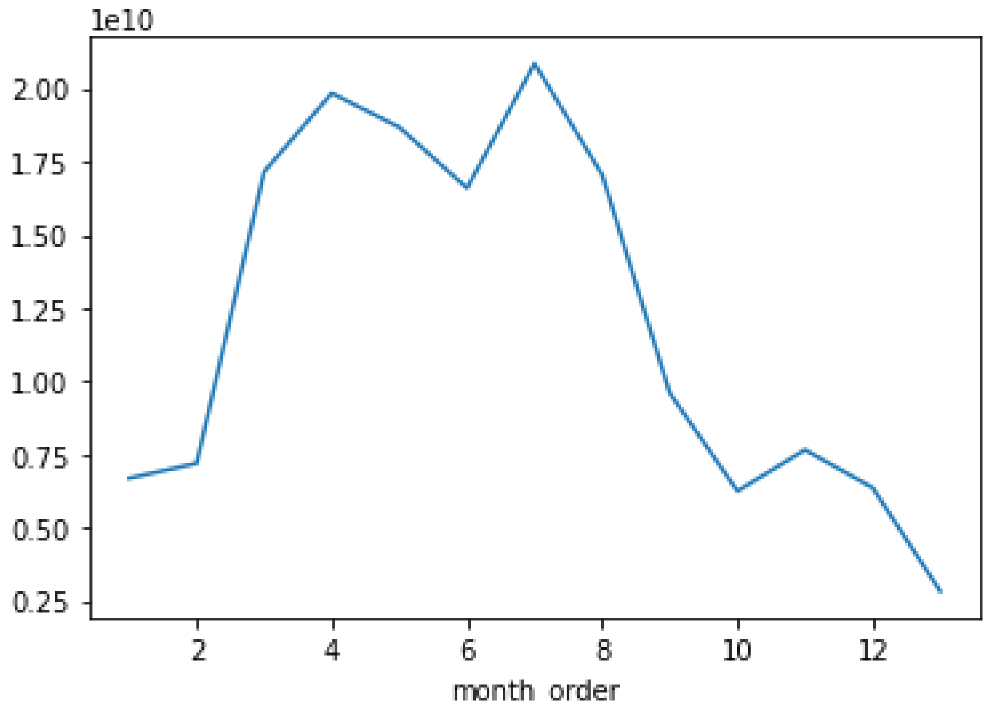


This plot is extrememly interesting- it seems to be saying that customers that have been with us from 2-6 months, and 10-12 months are spending a majority of the money. What is going on with these in-between spenders? The drop off for 1 month makes sense, since in the most recent month, we only have 15 days of data.

Further questions: Look into spending habits of our 6-10 month customers. Which loyalty categories do our top 5 fall into? Most likely on the higher end since they are bigger overall spenders.

[48]:
```
baskets.groupby('month_order').spent.sum().plot()
```

[48]: <AxesSubplot:xlabel='month_order'>



336, 441, 122. Top three merchants over both models.

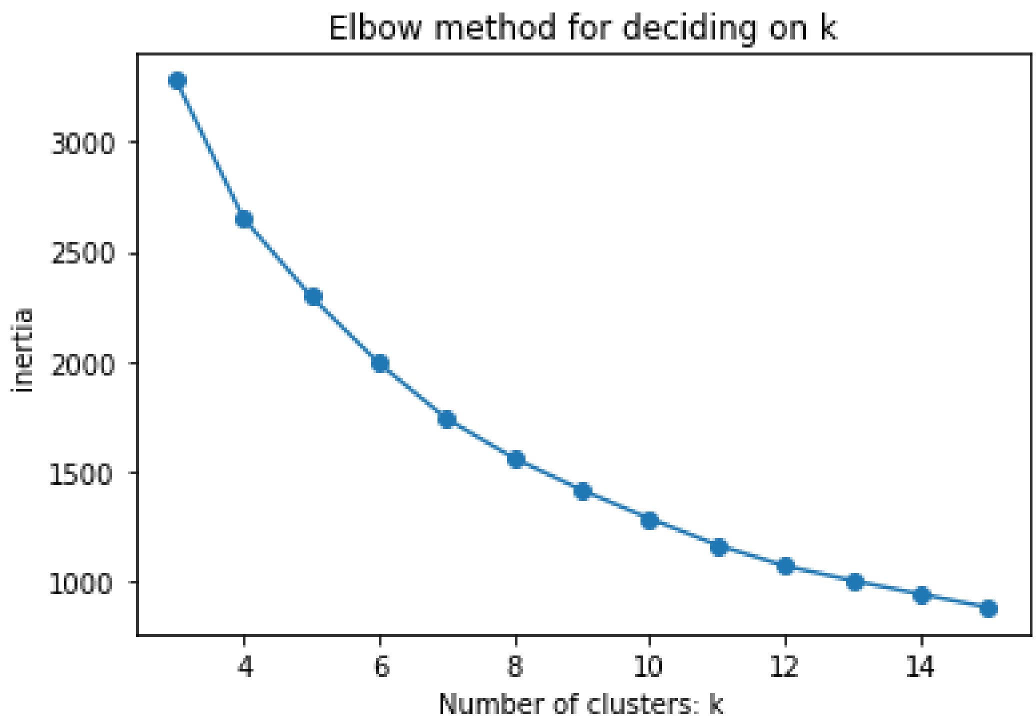
Clustering tests on merchants dataframe: dfmerch

[49]:
```
def find_elbow(df, colnames, clusters_range):
    df_for_cluster = df.loc[:,colnames]
    stscaler = StandardScaler().fit(df_for_cluster)
    normalized_df = stscaler.transform(df_for_cluster)

    inertias = [] # wcss: Within Cluster Sum of Squares
    for k in clusters_range:
        kmeans = KMeans(init='k-means++',n_clusters=k,n_init=100, max_iter=300,␣
↪random_state=0).fit(normalized_df)
        inertias.append(kmeans.inertia_)
    plt.figure()
    plt.plot(clusters_range,inertias, marker='o')
    plt.title('Elbow method for deciding on k')
    plt.xlabel('Number of clusters: k')
    plt.ylabel('inertia')
    plt.show()
    return
```

[50]:
```
colnames = dfmerch.columns[2:]
clusters_range = [3,4,5,6,7,8,9,10,11,12,13,14,15]
```

```
find_elbow(dfmerch, colnames,clusters_range)
```



[51]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, normalize
from sklearn.metrics import silhouette_score

def run_kmeans(df, colnames, k):
    df_for_cluster = df.loc[:,colnames]
    stscaler = StandardScaler().fit(df_for_cluster)
    normalized_df = stscaler.transform(df_for_cluster)

    kmeans = KMeans(init='k-means++',n_clusters=k,n_init=100, max_iter=300,␣
 ↪random_state=0).fit(normalized_df)
    df['cluster'] = kmeans.labels_
    return df

colnames = dfmerch.columns[2:]
k = 6
merchants_kmeans = run_kmeans(dfmerch, colnames,k)
merchants_kmeans.groupby("cluster").size()
```

[51]:
```
cluster
0    483
```

```
1     910
2      35
3      88
4     273
5     336
dtype: int64
```

[63]:

```
merchants_kmeans[merchants_kmeans.cluster == 2]
```

[63]:

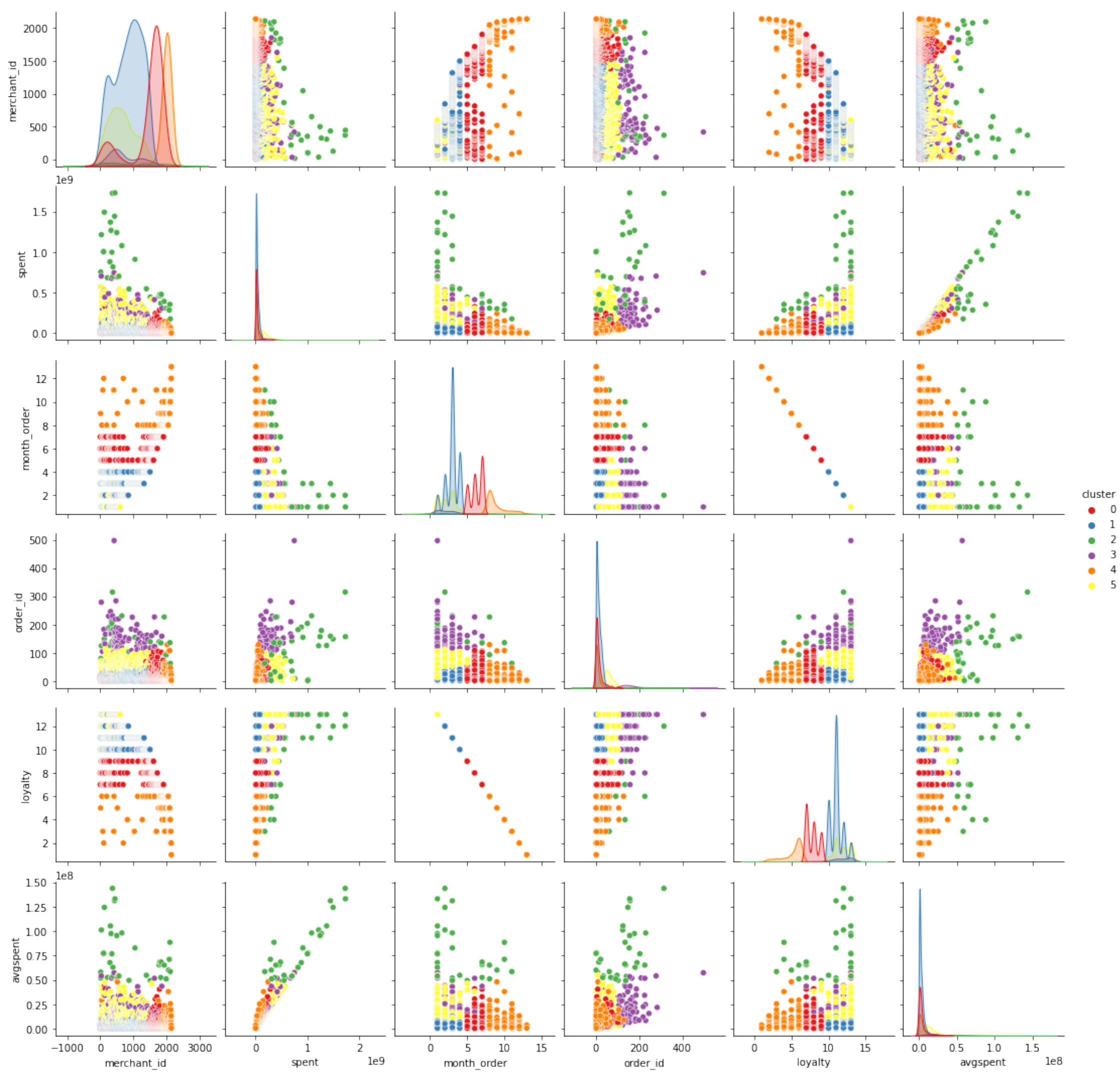
```
      merchant_id         spent  month_order  order_id  loyalty      avgspent  \
11             13  7.534382e+08            1         9       13  5.795678e+07
15             17  5.705613e+08            3        80       11  5.186921e+07
30             32  1.213518e+09            2       125       12  1.011265e+08
86             93  1.000304e+09            1         2       13  7.694650e+07
90             98  1.010569e+09            1         5       13  7.773608e+07
91             99  1.003596e+09            1         2       13  7.719969e+07
113           122  1.491349e+09            2       149       12  1.242791e+08
206           215  6.825753e+08            1        70       13  5.250579e+07
280           290  8.796906e+08            1       191       13  6.766851e+07
295           305  1.368091e+09            1       126       13  1.052377e+08
312           322  1.266263e+09            1       179       13  9.740484e+07
339           349  9.970763e+08            1       204       13  7.669818e+07
356           366  1.724671e+09            2       315       12  1.437226e+08
365           375  6.965972e+08            1       142       13  5.358440e+07
370           380  8.170310e+08            1       182       13  6.284854e+07
419           430  1.441119e+09            3       159       11  1.310108e+08
430           441  1.729729e+09            1       157       13  1.330561e+08
486           497  1.239717e+09            1       157       13  9.536284e+07
550           561  7.010735e+08            1        36       13  5.392873e+07
638           649  1.078626e+09            3       231       11  9.805689e+07
853           864  5.438185e+08            3       104       11  4.943805e+07
1033         1044  9.096947e+08            3       155       11  8.269952e+07
1376         1387  5.466440e+08            4        99       10  5.466440e+07
1492         1503  4.435634e+08            5       139        9  4.928483e+07
1759         1770  4.561210e+08            7       139        7  6.516014e+07
1781         1792  4.770495e+08            7        18        7  6.814993e+07
1810         1821  3.667897e+08            7        33        7  5.239853e+07
1813         1824  4.632534e+08            7       135        7  6.617906e+07
1825         1836  4.773995e+08            7        15        7  6.819993e+07
1913         1924  3.904352e+08            8       227        6  6.507254e+07
1973         1984  3.060403e+08            8        93        6  5.100672e+07
2042         2053  2.999915e+08            9         2        5  5.999830e+07
2065         2076  2.841870e+08           10        23        4  7.104675e+07
2084         2095  3.539850e+08           10       136        4  8.849625e+07
2098         2109  1.754556e+08           11        63        3  5.848522e+07

      cluster
```

```
11      2
15      2
30      2
86      2
90      2
91      2
113     2
206     2
280     2
295     2
312     2
339     2
356     2
365     2
370     2
419     2
430     2
486     2
550     2
638     2
853     2
1033    2
1376    2
1492    2
1759    2
1781    2
1810    2
1813    2
1825    2
1913    2
1973    2
2042    2
2065    2
2084    2
2098    2
```

[53]:
```
sns.pairplot(data=merchants_kmeans, hue="cluster", palette="Set1")
```

[53]: <seaborn.axisgrid.PairGrid at 0x7fa178d59c70>



The RFM is a commonly used industry model, while clustering is a striactly data science practice. The support of my top five from both models shows a strong case for those top 5. A way to find future top merchants is to run both models and find consistencies between clustering and RFM model. Talk about importance of monetary- which parts predicted the best? Use data to make assesments rather than intuition- RFM has lots of assumptions- dies it have science behind it. Quote clustering reference. Business can use clustering in the future, knowing that RFM supports it, look at other merchants in this cluster as valuable according to this notebook.

Regression model?

[y, x1, x2, x3] -> average spent, spent in first month, spent in last month, total spent.

[54]:
```
x = dfmerch.loc[:,['spent','order_id']]
y = dfmerch.loc[:,['avgspent']]
```

[55]:
```
xtrain = x[500:]
xtest = x[:500]
ytrain = y[500:]
ytest = y[:500]
```

[56]:
```
lin_r = linear_model.LinearRegression()
```

[57]:
```
lin_r.fit(xtrain, ytrain)
```

[57]:
```
LinearRegression()
```

[58]:
```
avg_pred = lin_r.predict(xtest)
```

[59]:
```
print("Coefficients: \n", lin_r.coef_)
```

```
Coefficients:
 [[9.60157141e-02 2.75397760e+03]]
```

[60]:
```
print("Mean squared error %.2f" % mean_squared_error(ytest, avg_pred))
```

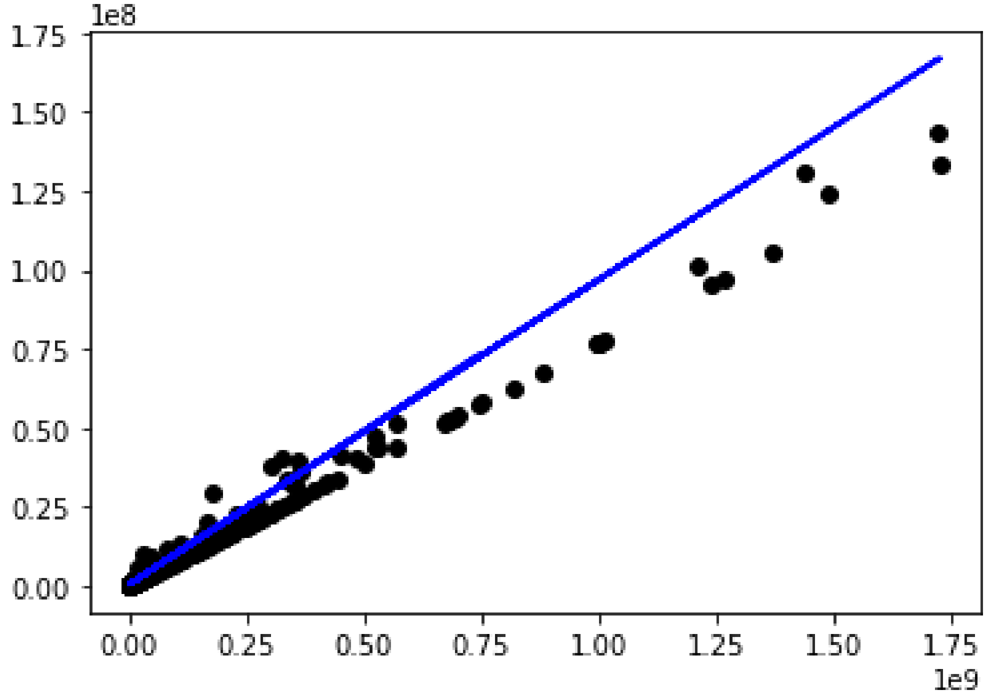
```
Mean squared error 22674540625835.45
```

[61]:
```
print("Coefficient of determination: %.2f" % r2_score(ytest, avg_pred))
```

```
Coefficient of determination: 0.94
```

[62]:
```
plt.scatter(xtest['spent'], ytest, color = "black")
plt.plot(xtest['spent'], avg_pred, color = "blue")
```

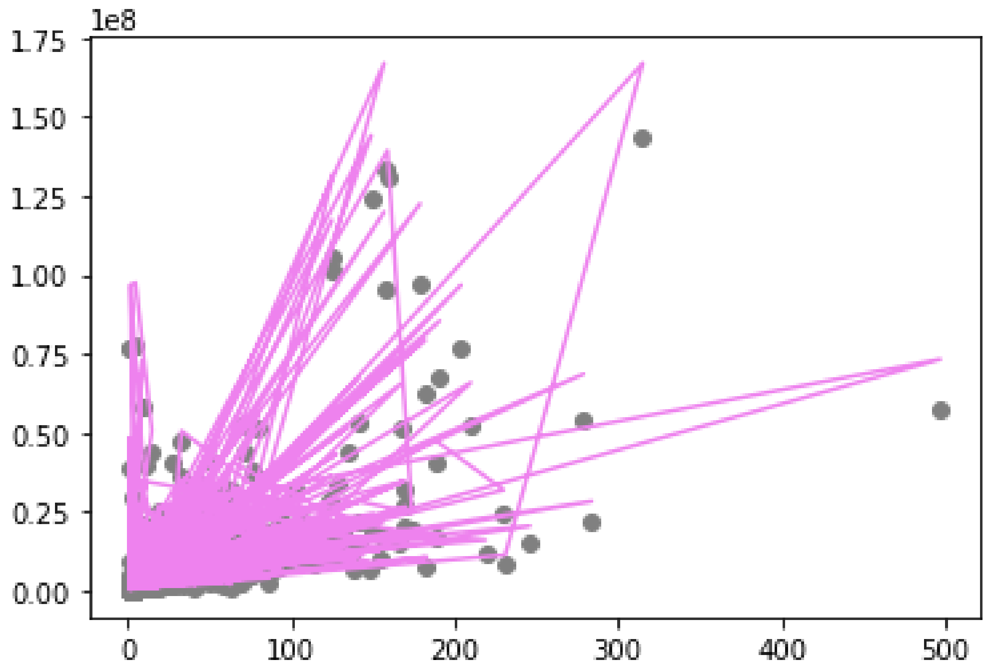
[62]:
```
[<matplotlib.lines.Line2D at 0x7fa17dcf40a0>]
```



This regression seems to be a generally good fit for the data. How to include in poster? How to tie into overall question? That taking top merchants we can predict average spending based on total spending?

[64]:
```
plt.scatter(xtest['order_id'], ytest, color= "gray")
plt.plot(xtest['order_id'], avg_pred, color = "violet")
```

[64]:
```
[<matplotlib.lines.Line2D at 0x7fa17a586ca0>]
```


1e8
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
100
200
300
400
500

[ ]: