Small Business Merchants

Transactional Analysis and Customer Segmentation

By: Mohit Bansal
+91-9958541140

Contents (1/2)…

Objectives and Scope of Study

Business Objectives

- Analyse and study the data for **small business merchants.**
- Develop **Univariate/Bivariate analysis** for the transactions as per business requirement
- **Clustering / Customer Segmentation** - Identify key Clusters based on Transaction type and description.

Scope of Study

- Process and develop **summary of the transactions**
- **Identify patterns** between variables using distribution plots/charts.
- **Time series analysis** based on daily transactions.

Data Period and data fields

- **Data Period : Nov-2015 to Mar-2017**
- A **data sample of individual bank statement transactions from 20 small business merchants** (indexed by Lead ID) from various industries.
- Each merchant could have multiple accounts (bank account id) over multiple months.
- Transactions may be **debits (withdrawals) or credits (deposits)** along with transaction description.

Approach/ Strategy?

Understand the requirement → **Gather data, Cleanse and analyse** → **Data analysis and Customer Segmentation**

Overview of the Analysis

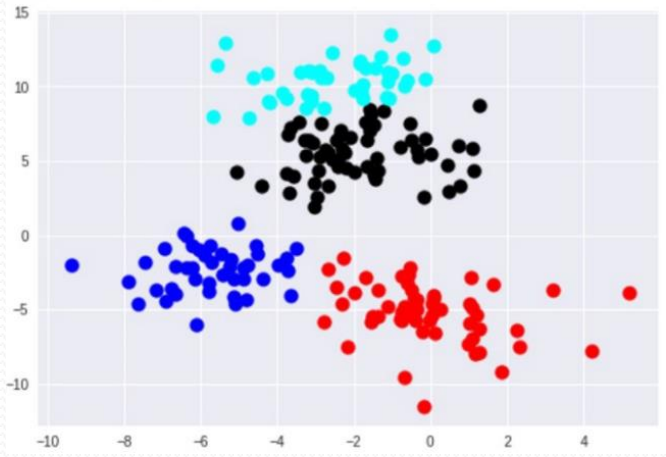
Process flow

- Understand the business requirements
- Data Cleansing (Outlier/Missing Value Analysis)
- Descriptive Analytics on the data
- A combination of NLP model with clustering

Cluster Output of transactions

Four Types of transaction descriptions

- Card Swipe transactions
- WorldPay
- Online Transfer
- Cheque/Check Payments



Data File Format: MS Excel file

Software platform used for analysis: Google Colab (Python) & Google Drive, MS Excel & CSV Files, MS PowerPoint, Adobe PDF

Data Import and Cleansing of data

- The data information provided for bank transactions as per below snapshot:

1	Lead ID	bankid	bank_account_id	account_number	Industry	post_date	description	transaction_type	amount	running_balance	trans_order	month name
9981	318465	9262	13425	xxxx6643	Health Care and Social Assistan	22-Jul-16	POS Withdrawal - USPS xxxxxxxxx x583	debit	9.8	805.99	3	7
9982	318465	9262	13425	xxxx6643	Health Care and Social Assistan	23-Jul-16	POS Withdrawal - FUEL COFFEE 1705 N	debit	10.7	733.46	3	7

- **Data Imported in Google Colab** which uses Python as the Language
- Benefits of using Google Colab –
  - User can use Google Cloud Server's to code and process data
  - Use Google's Software and latest updated versions, no need to install latest version of python or packages separately, it installs and stores on cloud environment.
  - Utilize Graphical Processing unit (GPU) and Tensor processing unit (TPU) from cloud server.

- The files imported seems to be have empty rows when checked for tail.
- 'Nan' appears in last rows showing the rows with full missing data, initially processes 144836 rows.

bank_2 = bank_1.dropna(how='all')

- Resulting values comes to have 29029 records.

```
[ ] #**Connect Google Colab to My google drive**
    from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive

[ ] %matplotlib inline
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns

    bank_1 = pd.read_csv('/content/drive/MyDrive/Bank_Data/Bank_Data_v1.csv')
    bank_1['month_year'] = pd.to_datetime(bank_1['post_date']).dt.to_period('M')
    print(bank_1)

        Lead ID  bankid  ...                Primary_key month_year
    0    308148    8535  ...  308148_8535_12460_42439_1    2016-03
    1    308148    8535  ...  308148_8535_12460_42450_1    2016-03
    2    308148    8535  ...  308148_8535_12460_42450_3    2016-03
    3    308148    8535  ...  308148_8535_12460_42450_2    2016-03
    4    308148    8535  ...  308148_8535_12460_42451_1    2016-03
```

Descriptive statistics (1/3)…

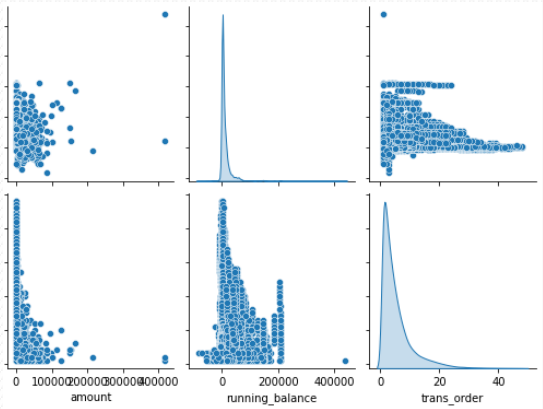
- **Analyse and identify the patterns between variables.**
- **Partial Missing Values** Reported: **'Month Name',** Deleted column to create one from date.
- Summary of numerical values with vital stats like count, mean, min, max and percentiles.

```
[ ]  del bank_3['month name']
     bank_3.describe()
```

	Lead ID	bankid	bank_account_id	amount	running_balance	trans_order
count	29029.000000	29029.000000	29029.000000	29029.000000	29029.000000	29029.000000
mean	320714.718282	8823.888491	14052.030556	1927.453475	10247.851975	5.397223
std	5051.317064	683.963276	703.164708	7450.638452	20055.683105	5.445505
min	308148.000000	6192.000000	12460.000000	0.000000	-84727.980000	1.000000
25%	316728.000000	8534.000000	13234.000000	29.250000	798.850000	2.000000
50%	321380.000000	8535.000000	14049.000000	160.500000	3820.940000	4.000000
75%	323253.000000	8545.000000	14636.000000	788.460000	11898.880000	7.000000
max	330698.000000	10656.000000	15148.000000	419000.000000	437942.290000	48.000000

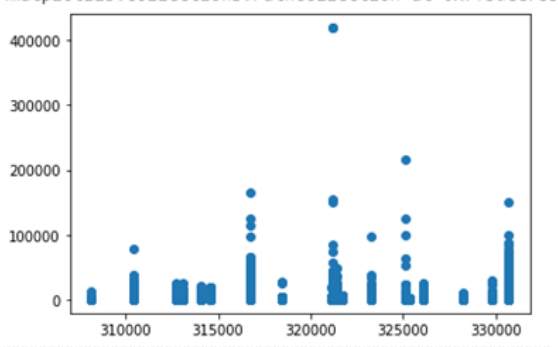
- **Analyse the outliers and distribution between variables which might produce bias in the data.**

```
[ ]  sns.pairplot(bank_3, diag_kind='kde')
```



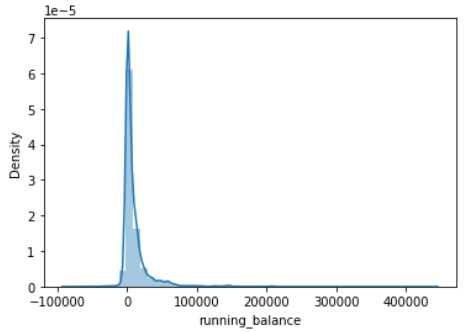

```
[ ]  plt.scatter( bank_3['Lead ID'],bank_3['amount'])
```

```
<matplotlib.collections.PathCollection at 0x7fbdcc783
```




```
[ ]  sns.distplot(bank_3['running_balance'])
```

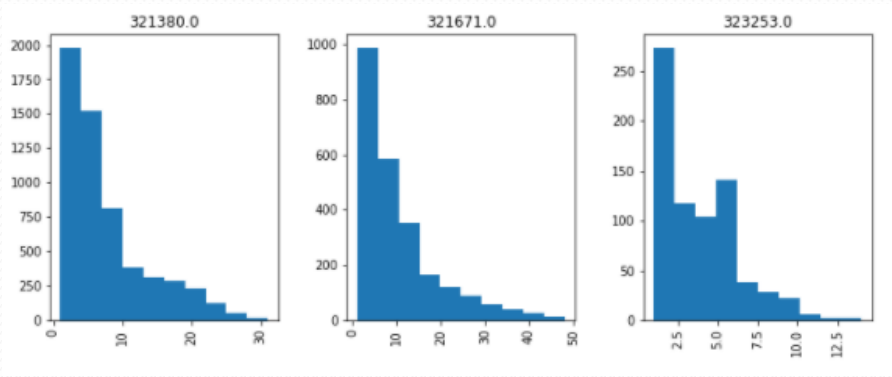
```
/usr/local/lib/python3.6/dist-packages/seaborn/distribut
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f3c2493b4e0
```



Descriptive statistics (2/3)…

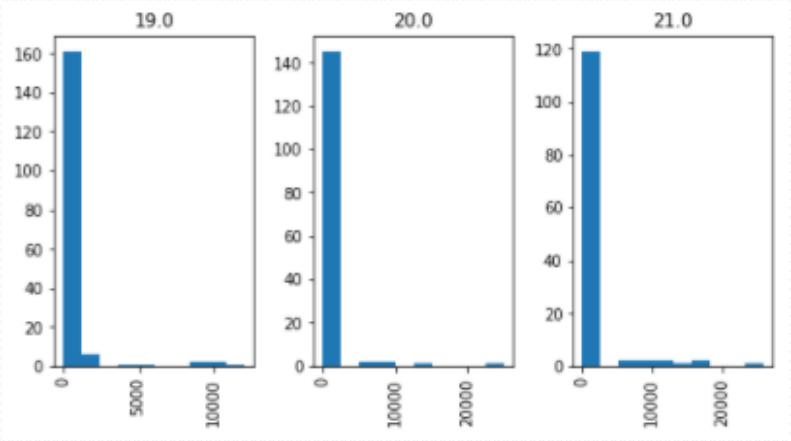
- **Histogram to understand the transactional behaviour and derive insights:**
  - Merchant ID Vs Trans_order - Out of 21 merchant IDs, 4 of them have maximum of >30 daily transactions on their accounts.

```
[ ]  bank_3.hist(by='Lead ID',column='trans_order',figsize=(20,30))
```



  - Trans_order Vs Amount - Majority of transaction amount (**>20K**) is mainly till **21 transactions.**

```
[ ]  bank_3.hist(by='trans_order',column='amount',figsize=(20,40))
```



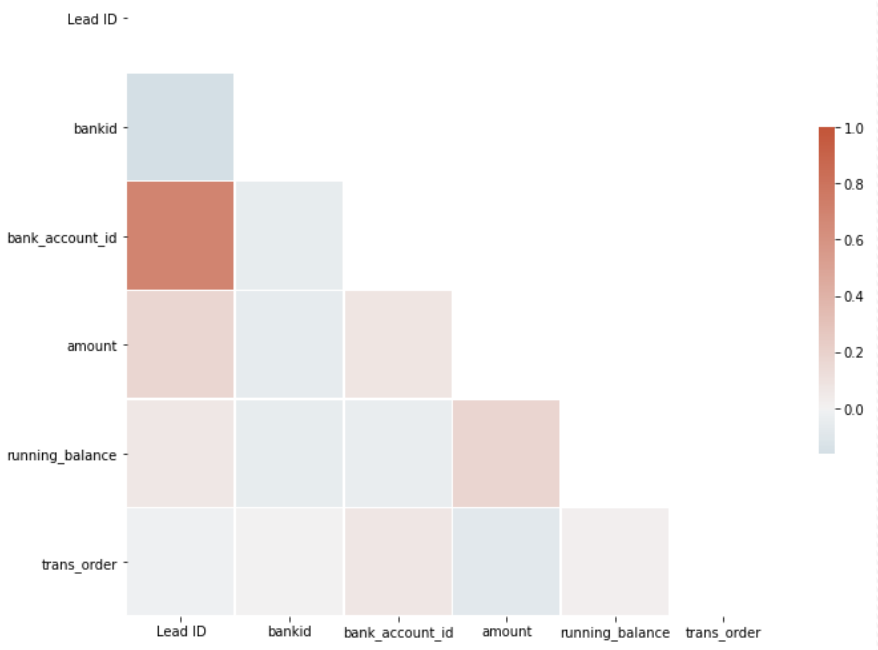
Descriptive statistics (3/3)

- **Considering the Correlation Analysis to study the relation between the variables.**
- Post Analysing the results, there seems to be no high correlation between values (ignoring Lead ID Vs Bank account ID as these are ID variables)**.**

```
[ ]  # Compute the correlation matrix
     corr = bank_3.corr()
     print(corr)
```

```
                  Lead ID    bankid  ...  running_balance  trans_order
Lead ID          1.000000 -0.158936  ...         0.063881    -0.008326
bankid          -0.158936  1.000000  ...        -0.053088     0.001317
bank_account_id  0.700699 -0.050178  ...        -0.041772     0.076003
amount           0.179994 -0.057082  ...         0.181640    -0.078309
running_balance  0.063881 -0.053088  ...         1.000000     0.022697
trans_order     -0.008326  0.001317  ...         0.022697     1.000000

[6 rows x 6 columns]
```

```
[ ]  # Generate a mask for the upper triangle
     mask = np.triu(np.ones_like(corr, dtype=bool))

     # Set up the matplotlib figure
     f, ax = plt.subplots(figsize=(11, 9))

     # Generate a custom diverging colormap
     cmap = sns.diverging_palette(230, 20, as_cmap=True)
     sns.heatmap(corr, mask=mask, cmap=cmap, vmax=1, center=0,
                 square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

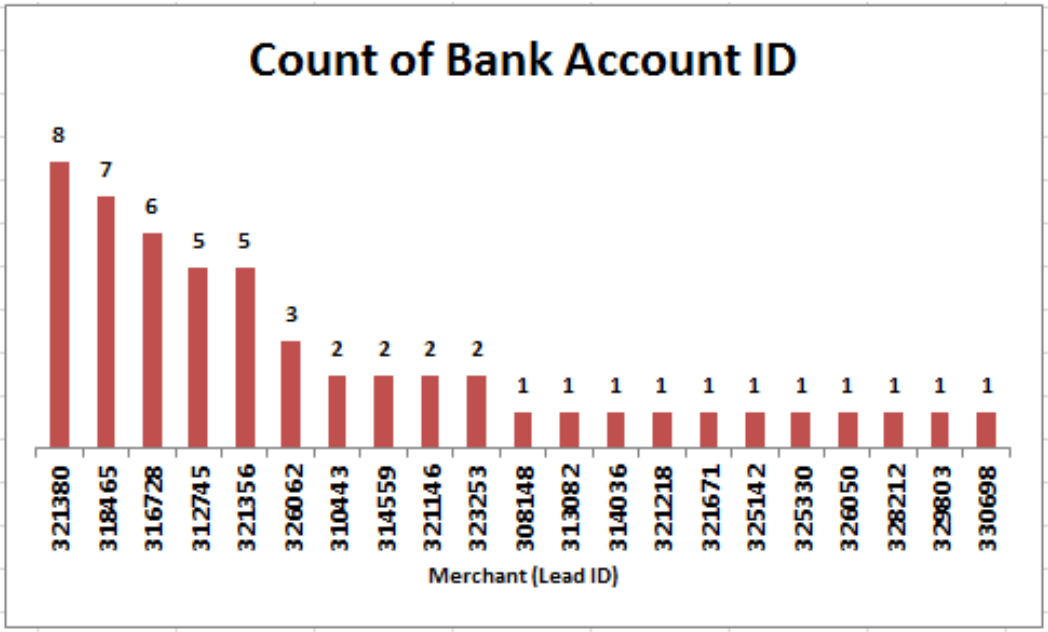


Analytics Insights as per requirement (1/5)…

1(i). the number of bank accounts for each merchant

```
[ ]  # 1. (i) the number of bank accounts for each merchant
     Data_1 = bank_3.groupby('Lead ID')['bank_account_id'].nunique()
     print(Data_1)
     Data_1.to_csv('/content/drive/MyDrive/Bank_Data/Data_1.csv', index=True)
```

```
Lead ID
308148    1
310443    2
312745    5
```

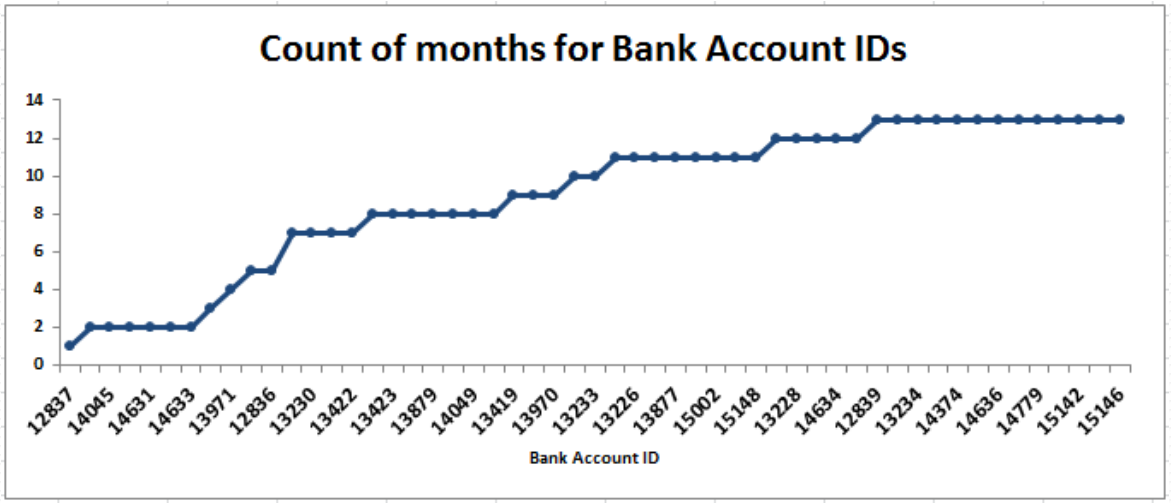


Analytics Insights as per requirement (2/5)…

1(ii). the number of months of each bank account for which data is available

```
[ ]  # 1. (ii) the number of months of each bank account for which data is available
     import datetime

     #bank_3['month_year'] = pd.to_datetime(bank_3['post_date']).dt.to_period('M')
     Data_2 = bank_3.groupby('bank_account_id')['month_year'].nunique()
     print(Data_2)
     Data_2.to_csv('/content/drive/MyDrive/Bank_Data/Data_2.csv', index=True)
```

```
bank_account_id
12460    10
12654     7
12655    12
12835     5
12836     5
12837     1
12838    11
```



Analytics Insights as per requirement (3/5)…

1(iii). the total number of credits (deposits), debits (withdrawals) and their averages per month for each bank account and each merchant

```
[9]  # 1. (iii) the total number of credits (deposits), debits (withdrawals) and their averages per month for each bank account and each merchant
     # Mean refers to the averages
     Data_3 = pd.pivot_table(bank_3,index=['Lead ID','bank_account_id','month_year'],columns=['transaction_type'],values=['amount'],aggfunc=[len,np.mean])
     print(Data_3.round(decimals=0))
     Data_3.to_csv('/content/drive/MyDrive/Bank_Data/Data_3.csv', index=True)
```

```
                                        len           mean
                                     amount         amount
transaction_type                     credit  debit   credit    debit
Lead ID bank_account_id month_year
308148  12460           2016-03         5.0   15.0   1220.0    333.0
                        2016-04         1.0   12.0   1500.0    216.0
                        2016-05         4.0    8.0    194.0     99.0
                        2016-06         4.0   13.0    592.0    156.0
                        2016-07        20.0   30.0    405.0    280.0
...                                     ...    ...      ...      ...
330698  14374           2016-10        27.0   77.0  33553.0  11766.0
                        2016-11        26.0   85.0  32040.0   9800.0
                        2016-12        33.0   93.0  27355.0   9495.0
                        2017-01        32.0  196.0  28218.0   4656.0
                        2017-02         6.0   38.0  32059.0   5011.0

[478 rows x 4 columns]
```

Sample Output for one Merchant ID: 308148
No. of Records shown in Sample snap = 10
Total records in output table = 478

			Count		Average	
			amount	amount	amount	amount
transaction_type			credit	debit	credit	debit
Lead ID	bank_account_id	month_year				
308148	12460	2016-03	5	15	1,220	333
308148	12460	2016-04	1	12	1,500	216
308148	12460	2016-05	4	8	194	99
308148	12460	2016-06	4	13	592	156
308148	12460	2016-07	20	30	405	280
308148	12460	2016-08	38	41	363	185
308148	12460	2016-09	32	50	233	213
308148	12460	2016-10	32	57	229	156
308148	12460	2016-11	33	89	569	173
308148	12460	2016-12	3	5	131	103

Analytics Insights as per requirement (4/5)…

1(iv). The total dollar value of credits, debits and their averages per month for each bank account and each merchant

```
[ ]  #1. (iv) the total dollar value of credits, debits and their averages per month for each bank account and each merchant
     # Mean refers to the averages
     Data_4 = pd.pivot_table(bank_3,index=['Lead ID','bank_account_id','month_year'],columns=['transaction_type'],values=['amount'],aggfunc=[np.sum,np.mean])
     print(Data_4.round(decimals=2))
     Data_4.to_csv('/content/drive/MyDrive/Bank_Data/Data_4.csv', index=True)
```

```
                                        sum              mean
                                     amount            amount
transaction_type                     credit      debit  credit     debit
Lead ID bank_account_id month_year
308148  12460           2016-03      6098.62    4995.32 1219.72    333.02
                        2016-04      1500.00    2595.75 1500.00    216.31
                        2016-05       774.95     791.74  193.74     98.97
                        2016-06      2369.38    2028.87  592.34    156.07
                        2016-07      8105.04    8396.89  405.25    279.90
...                                       ...        ...     ...       ...
330698  14374           2016-10    905930.37  905998.55 33552.98  11766.21
                        2016-11    833031.28  833031.28 32039.66   9800.37
                        2016-12    902727.28  883050.45 27355.37   9495.17
                        2017-01    902978.48  912591.12 28218.08   4656.08
                        2017-02    192354.78  190403.36 32059.13   5010.61

[478 rows x 4 columns]
```

Sample Output for one Merchant ID: 308148
No. of Records shown in Sample snap = 10
Total records in output table = 478

			Sum		Average	
			amount	amount	amount	amount
transaction_type			credit	debit	credit	debit
Lead ID	bank_ac	month_year				
308148	12460	2016-03	6,099	4,995	1,220	333
308148	12460	2016-04	1,500	2,596	1,500	216
308148	12460	2016-05	775	792	194	99
308148	12460	2016-06	2,369	2,029	592	156
308148	12460	2016-07	8,105	8,397	405	280
308148	12460	2016-08	13,782	7,569	363	185
308148	12460	2016-09	7,471	10,647	233	213
308148	12460	2016-10	7,339	8,864	229	156
308148	12460	2016-11	18,779	15,398	569	173
308148	12460	2016-12	393	514	131	103

Analytics Insights as per requirement (5/5)…

1(v). Aggregate the answers to (iii) and (iv) at the merchant level, industry level and bank id level

```
[ ]  #1. (v) Aggregate the answers to (iii) and (iv) at the merchant level, industry level and bankid level
     Data_5 = pd.pivot_table(bank_3,index=['Lead ID','Industry','bankid'],columns=['transaction_type'],values=['amount'],aggfunc=[len, np.sum, np.mean])
     print(Data_5.round(decimals=0))
     Data_5.to_csv('/content/drive/MyDrive/Bank_Data/Data_5.csv', index=True)
```

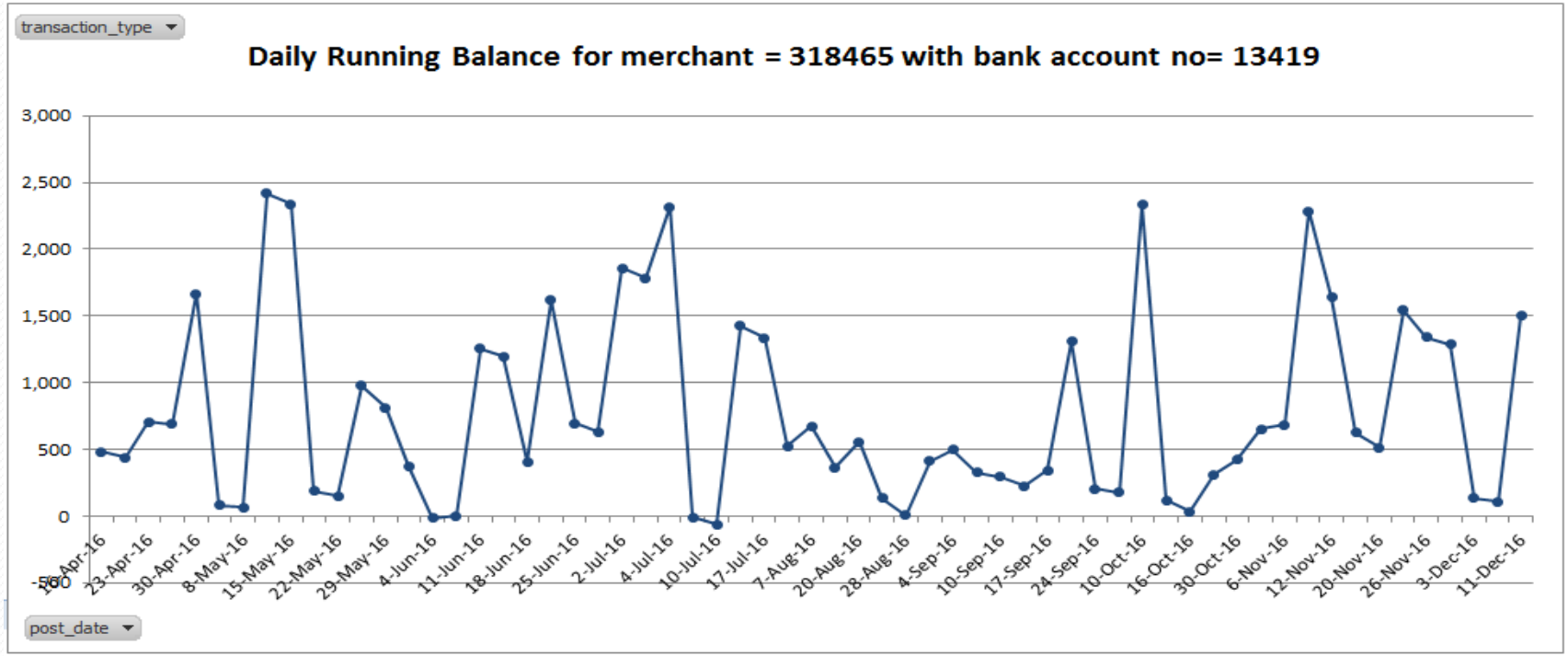
Total records in output table = 23

			Count		Sum		Average	
			amount	amount	amount	amount	amount	amount
transaction_type			credit	debit	credit	debit	credit	debit
Lead ID	Industry	bankid						
308148	Accommodation and Food Services	8535	172	320	66,611	61,800	387	193
310443	Construction	6192	35	246	3,25,133	3,46,662	9,290	1,409
312745	Professional, Scientific, and Technical Services	8544	160	600	7,19,438	7,13,251	4,496	1,189
313082	Professional, Scientific, and Technical Services	8535	112	1,026	5,47,025	5,58,877	4,884	545
314036	Retail Trade	9966	276	1,985	4,45,920	4,32,975	1,616	218
314559	Information Technology	8534	143	284	5,27,618	5,19,814	3,690	1,830
316728	Construction	8534	30	93	1,31,837	1,25,563	4,395	1,350
316728	Construction	8544	268	2,221	23,31,183	22,90,281	8,698	1,031
318465	Health Care and Social Assistance	9262	369	1,694	2,31,849	2,29,561	628	136
321146	Retail Trade	10479	121	454	1,05,932	1,00,592	875	222
321218	Agriculture, Forestry, Fishing and Hunting	10656	57	849	14,44,858	14,66,503	25,348	1,727
321356	Other Services (except Public Administration)	8544	697	788	41,00,524	41,42,485	5,883	5,257
321380	Finance and Insurance	8534	4,436	687	25,12,410	25,15,316	566	3,661
321380	Finance and Insurance	10591	503	86	2,09,684	1,61,625	417	1,879
321671	Retail Trade	8534	642	1,788	3,12,137	3,07,190	486	172
323253	Retail Trade	8545	308	425	9,76,417	9,90,455	3,170	2,330
325142	Other Services (except Public Administration)	8535	115	876	4,72,618	4,63,603	4,110	529
325330	Retail Trade	8544	312	257	82,662	80,230	265	312
326050	Accommodation and Food Services	8534	190	459	2,88,066	2,71,316	1,516	591
326062	Accommodation and Food Services	8535	852	1,443	8,41,637	8,47,078	988	587
328212	Educational Services	8535	282	345	1,77,403	1,74,828	629	507
329803	Construction	8535	116	655	2,19,079	2,12,388	1,889	324
330698	Health Care and Social Assistance	8545	346	906	109,40,565	109,29,049	31,620	12,063

Time Series: Merchant ID: 318465 & Bank Account: 13419 (1/2)

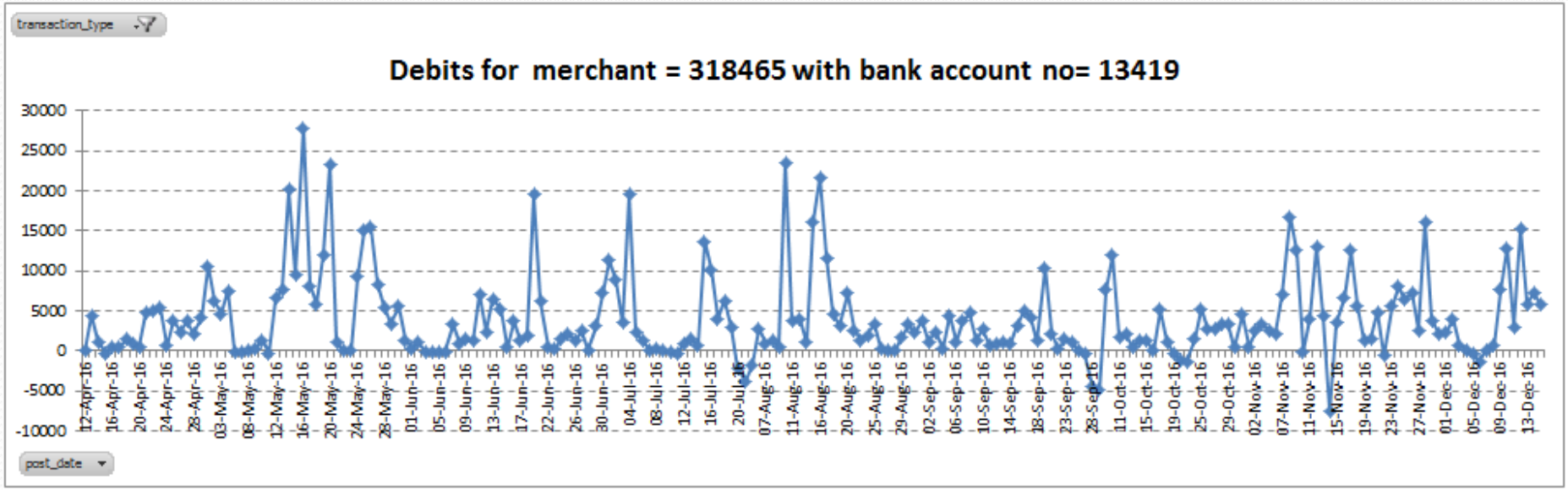
2. Consider Lead ID: 318465, bank_account_id: 13419 - plot the withdrawals, deposits and daily balance as a daily time series; do the same for Lead ID: 326062, bank_account_id : 14046

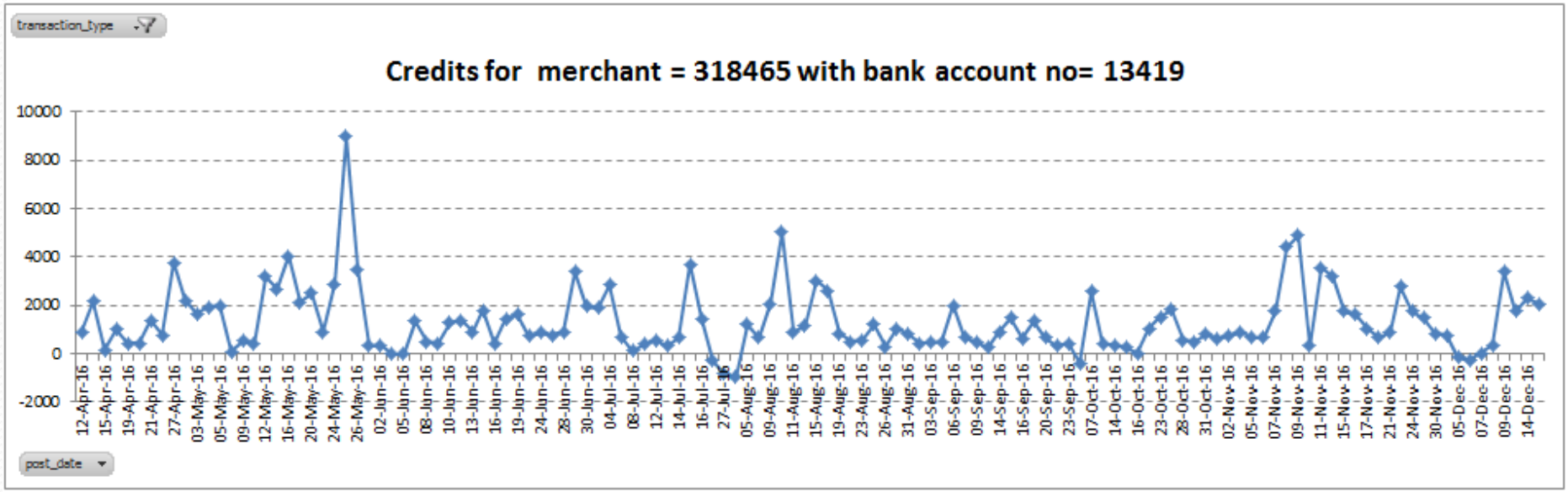
Data was filtered for the transactions to obtain the last transaction with running balance for each day as the final daily balance.



The Same has been done for Lead ID: 326062, bank_account_id : 14046 in the Python Code Submitted.

Time Series: Merchant ID: 318465 & Bank Account: 13419 (2/2)

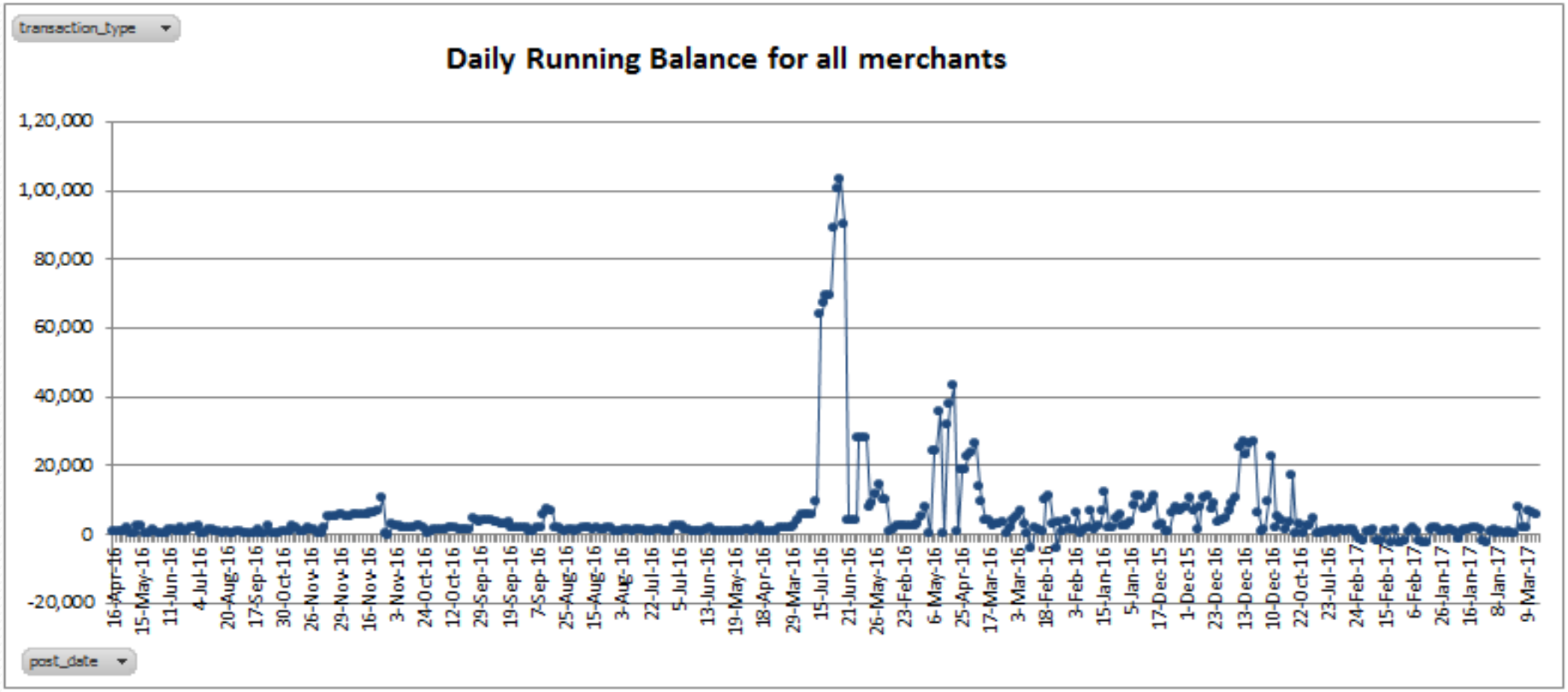




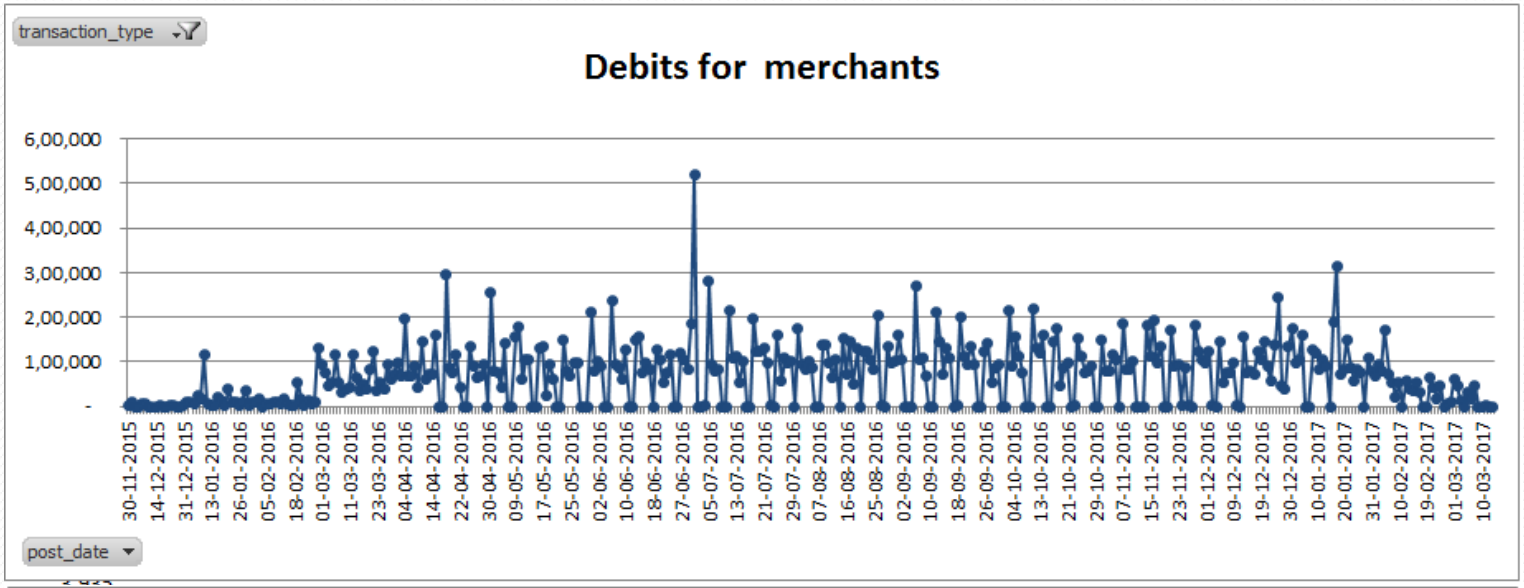
Overall Time Series for the data including all merchants (1/2)…

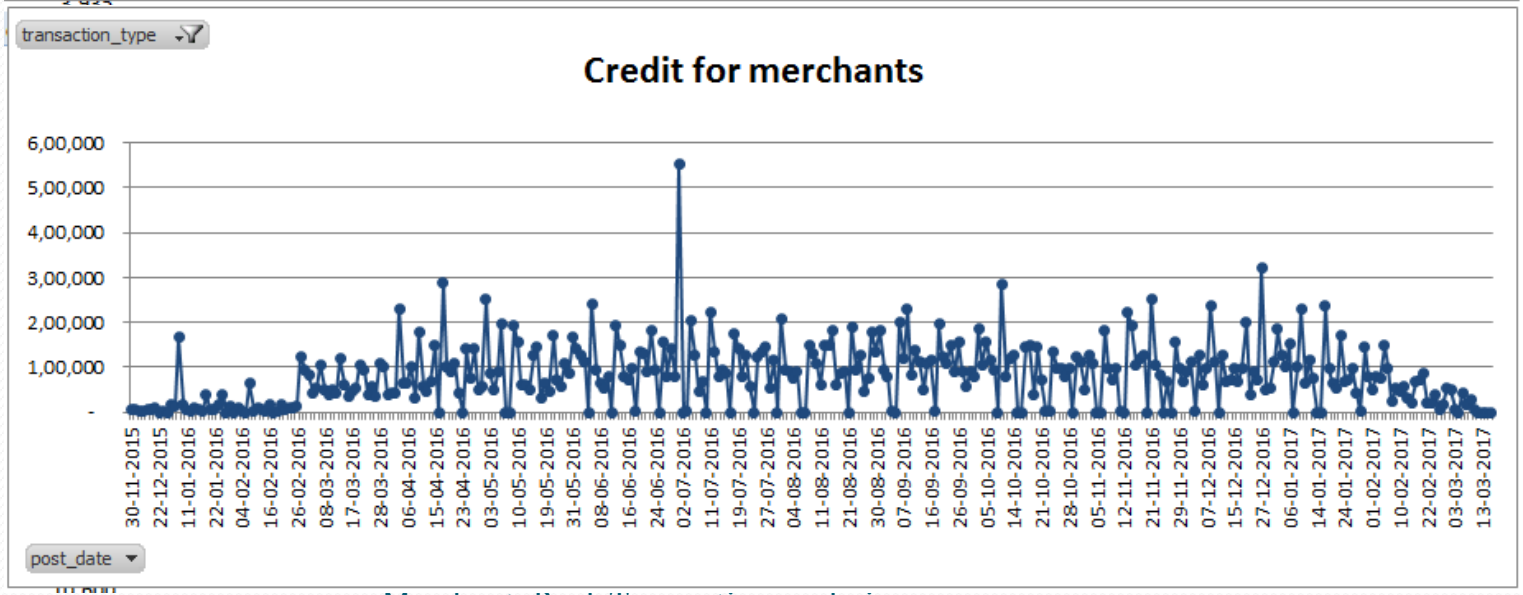
3. 3. For the merchants with the above Lead IDs, plot the withdrawals, deposits and daily balance as a daily time series (aggregate over all their bank accounts)

Data was filtered for the transactions to obtain the last transaction with running balance for each day as the final daily balance.



Overall Time Series for the data including all merchants (2/2)





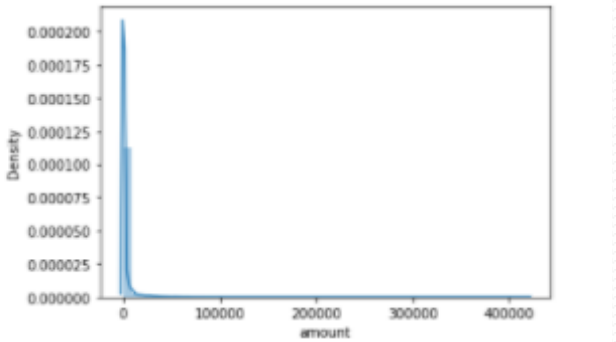
Cash Flow of the business : Additional Analysis

- **Top 5 and Bottom 5 Analysis (by Merchant and Bank ID) and Distributions for amounts**

```
[ ]  #Top 5 Merchant IDs by amount
     #del bank_3['month name']

     #bank_3.to_csv('/content/drive/MyDrive/Bank_Data/Bank_Data_v3.csv', index=False)

     bank_4 = bank_3.sort_values('amount',ascending = False).groupby('Lead ID').head(2)
     print(bank_4.head(5))

     #Bottom 5 Merchant IDs by amount
     bank_4 = bank_3.sort_values('amount',ascending = True).groupby('Lead ID').head(2)
     print(bank_4.head(5))


     #Top 5 bank IDs by amount
     bank_4 = bank_3.sort_values('amount',ascending = False).groupby('bankid').head(2)
     print(bank_4.head(5))

     #Bottom 5 bank IDs by amount
     bank_4 = bank_3.sort_values('amount',ascending = False).groupby('bankid').head(2)
     print(bank_4.head(5))
```

```
[ ]  sns.distplot(bank_3['amount'])
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distribution
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f3c24992400>
```




```
[ ]  sns.distplot(bank_3['running_balance'])
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distribution
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f3c2493b4e0>
```

Output

A	B	C	D	E	F	G	H	I	J	K	L	M
Lead ID	bankid	bank_account_id	account_number	Industry	post_date	description	transaction_type	amount	running_balance	trans_order	Primary_key	month_year
321218	10656	14779	x0441	Agriculture,	01-Jul-16	DDA Deposi	credit	4,19,000	4,37,942	1	321218_10	2016-07
321218	10656	14779	x0441	Agriculture,	01-Jul-16	Wire Transf	debit	4,19,000	18,942	2	321218_10	2016-07
325142	8535	14049	xxxx3690	Other Servic	18-Jan-17	WITHDRAW.	debit	2,15,000	-11,476	2	325142_85	2017-01
316728	8544	13232	8118	Constructio	06-Jan-16	eDeposit in	credit	1,64,457	1,86,945	6	316728_85	2016-01
330698	8545	14374	1693	Health Care	18-Apr-16	ACH deposit	credit	1,50,000	65,272	4	330698_85	2016-04

Additional Analysis Opportunities

Potential Factors which could be used further to customize/personalize the services

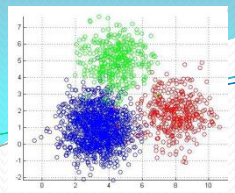
- Percentiles within Merchant and bank ID
- Outliers to remove possible bias in results

- Check for data discrepancy in running balance vs Amounts

Outlier Analysis | Data Errors | New Business Opportunity | Industry Analysis

- No. of transactions for merchants help analyse their volume of trade, Opportunity for New Business

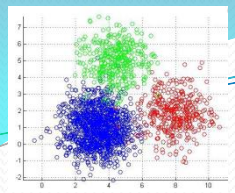
- Distribution of amounts and Industry level analysis which could help understand the GAP between different merchants.



Application of NLP: A Cluster Model to segment the banking transaction descriptions (1/3)…

- **Cluster Model done using below process:**

Declare Primary Key	Pre-Processing Data	k-means clustering with ELBOW Method	k-means clustering with ELBOW Method
1. Include a new field primary key for reference. 2. It is combination of Lead ID, Bank ID, Bank Account ID and transaction order	**1. Initially running a k-means clustering on the** Numerical fields in data to understand the cluster behaviour**.** **2. Keep only the fields required in the data. We keep** Primary Key, Transaction type and transaction description.	1. We consider a combination of **transaction description and transaction type** for Clustering. 2. For Clusters, install the required **packages NLTK, k-means and sklearn, tfidfvectorizer, train test split**	1. We consider a combination of **transaction description and transaction type** for Clustering. 2. Split the train test sample and use the whole data for modelling clusters. 3. **Apply K-means clustering on** string for 10 clusters 4. **Use ELBOW Method to decide on the inflexion point** to find optimal solution



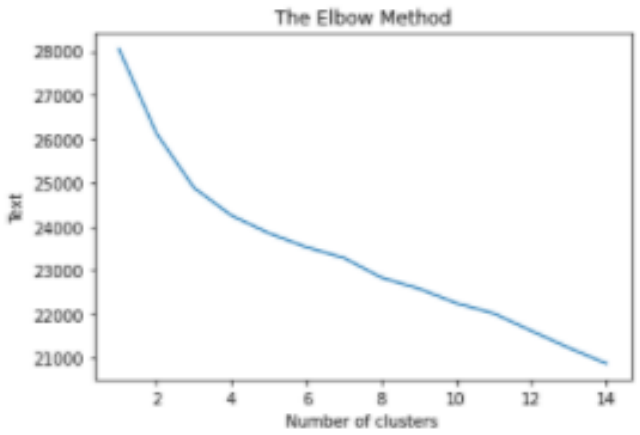
Application of NLP: A Cluster Model to segment the banking transaction descriptions (2/3)…

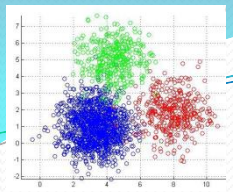
- **Fitting for 15 clusters to test for minimum no. of clusters.**
- **From the below chart, 3 or 4 looks a valid no of clusters to test the model.**
- **We fit the model for both the clusters and view the 3/4 clusters by transaction descriptions.**

```
print("Top terms per cluster:")
order_centroids = kmeans.cluster_centers_.argsort()[:, ::-1]
terms = tfidf_vectorizer.get_feature_names()
for i in range(5):
    print("Cluster %d:" % i),
    for ind in order_centroids[i, :15]:
        print(' %s' % terms[ind]),
    print
```

```
Top terms per cluster:
Cluster 0:
 desdep
 idlkxx
 trf
 worldpay
 idcredit
 xx
 co
 indncitgo
 indncrystal
 famil
 indnconcorde
 party
 station
 sho
 lake
Cluster 1:
 check
 debit
 xxdebit
 xxxxxxxdebit
 cashed
```

```
# Using the elbow method to find the optimal number of clusters
sse = []
for i in range(1, 15):
    kmeans_2 = KMeans(n_clusters = i, init = 'k-means++', random_state = 5)
    kmeans_2.fit(tfidf)
    sse.append(kmeans_2.inertia_)
plt.plot(range(1, 15), sse)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Text')
plt.show()
```





Application of NLP: A Cluster Model to segment the banking transaction descriptions (3/3)

- **Output to show 4 clusters:**

Lead ID	bankid	bank_account_id	account_number	Industry	post_date	description	transaction_type	amount	running_balance	trans_order	Primary_key	month_year	trans_type	year	month	Cluster_ID
316728	8544	13234	8126	Constructio	15-Jan-16	CHECK # 110	debit	751.77	4331.47	5	316728_854	2016-01	2	2016	1	2
316728	8544	13234	8126	Constructio	15-Jan-16	CASHED CHE	debit	1006.08	-941	2	316728_854	2016-01	2	2016	1	2
316728	8544	13234	8126	Constructio	15-Jan-16	ONLINE TRA	credit	6500	5559	3	316728_854	2016-01	1	2016	1	3
312745	8544	12839	3582	Professiona	19-Jan-16	ATM CASH D	credit	800	7487.31	3	312745_854	2016-01	1	2016	1	0
312745	8544	12839	3582	Professiona	19-Jan-16	NON-WELLS	debit	2.5	6711.67	1	312745_854	2016-01	2	2016	1	0

- **Below are the 4 groups with Transaction description and dates for all merchants (Can be filtered for one merchant):**

post_date	description	transaction_type	amount	Cluster_
19-Jan-16	ATM CASH DEPOSIT ON 01/19 1341 STELTON RD. PISCA	credit	800	0
19-Jan-16	NON-WELLS FARGO ATM TRANSACTION FEE	debit	3	0
19-Jan-16	RECURRING PAYMENT AUTHORIZED ON 01/16 SXM*SIR	debit	24	0
19-Jan-16	ATM WITHDRAWAL AUTHORIZED ON 01/19 1341 STELT	debit	40	0
19-Jan-16	NON-WF ATM WITHDRAWAL AUTHORIZED ON 01/16 B	debit	306	0
19-Jan-16	BILL PAY Terreno 60 Ethel ON-LINE No Account Numbe	debit	4,446	0
19-Jan-16	PURCHASE AUTHORIZED ON 01/15 SHELL OIL xxxx2460	debit	12	0
19-Jan-16	PURCHASE AUTHORIZED ON 01/15 JAKES FOOD &	debit	22	0
19-Jan-16	PURCHASE AUTHORIZED ON 01/14 SNARFS BROADWAY	debit	26	0
19-Jan-16	PURCHASE AUTHORIZED ON 01/16 STAPLSxxxxxxxx8300	debit	43	0

post_date	description	transaction_type	amount	Cluster_
29-Feb-16	WorldPay DES:DEP TRF ID:LKxx1140 xx2716 INDN:CASH	credit	2	1
29-Feb-16	WorldPay DES:DEP TRF ID:LKxx3402 xx2716 INDN:VISTA	credit	5	1
29-Feb-16	WorldPay DES:DEP TRF ID:LKxx5876 xx2816 INDN:RSVK	credit	5	1
29-Feb-16	WorldPay DES:DEP TRF ID:LKxx5876 xx2716 INDN:RSVK	credit	7	1
29-Feb-16	WorldPay DES:DEP TRF ID:LKxx7818 xx2616 INDN:CONC	credit	18	1
29-Feb-16	WorldPay DES:DEP TRF ID:LKxx5331 xx2716 INDN:CRYST	credit	18	1
29-Feb-16	WorldPay DES:DEP TRF ID:LKxx5876 xx2616 INDN:RSVK	credit	18	1
29-Feb-16	WorldPay DES:DEP TRF ID:LKxx0222 xx2716 INDN:ZAKIA	credit	20	1

post_date	description	transaction_type	amount	Cluster_
15-Jan-16	CHECK # 1109	debit	751.77	2
15-Jan-16	CASHED CHECK # 1108	debit	1006.08	2
19-Jan-16	CASHED CHECK # 556	debit	200	2
19-Jan-16	CHECK # 1103	debit	200	2
19-Jan-16	CHECK # 1113	debit	200	2
19-Jan-16	CASHED CHECK # 1116	debit	250	2
19-Jan-16	CHECK # 1114	debit	250	2
19-Jan-16	CHECK # 1115	debit	394.79	2
19-Jan-16	CHECK # 1112	debit	760	2
19-Jan-16	CHECK # 1113	debit	962.57	2

post_date	description	transaction_type	amount	Cluster_
15-Jan-16	ONLINE TRANSFER FROM METRO CONCRETE CREATIO R	credit	6500	3
22-Jan-16	ONLINE TRANSFER TO METRO CONCRETE CREATIONS RI	debit	6000	3
22-Jan-16	ONLINE TRANSFER FROM METRO CONCRETE CREATIO R	credit	6000	3
25-Jan-16	ONLINE TRANSFER REF #IBEXWTF58N TO PLATINUM CA	debit	254.97	3
01-Feb-16	ONLINE TRANSFER TO METRO CONCRETE CREATIONS RI	debit	5000	3
01-Feb-16	ONLINE TRANSFER FROM METRO CONCRETE CREATIO R	credit	5000	3
03-Feb-16	ONLINE TRANSFER FROM RANGOLI LLC REF #IBETZVJGT	credit	1020	3

Thank you!

Questions?