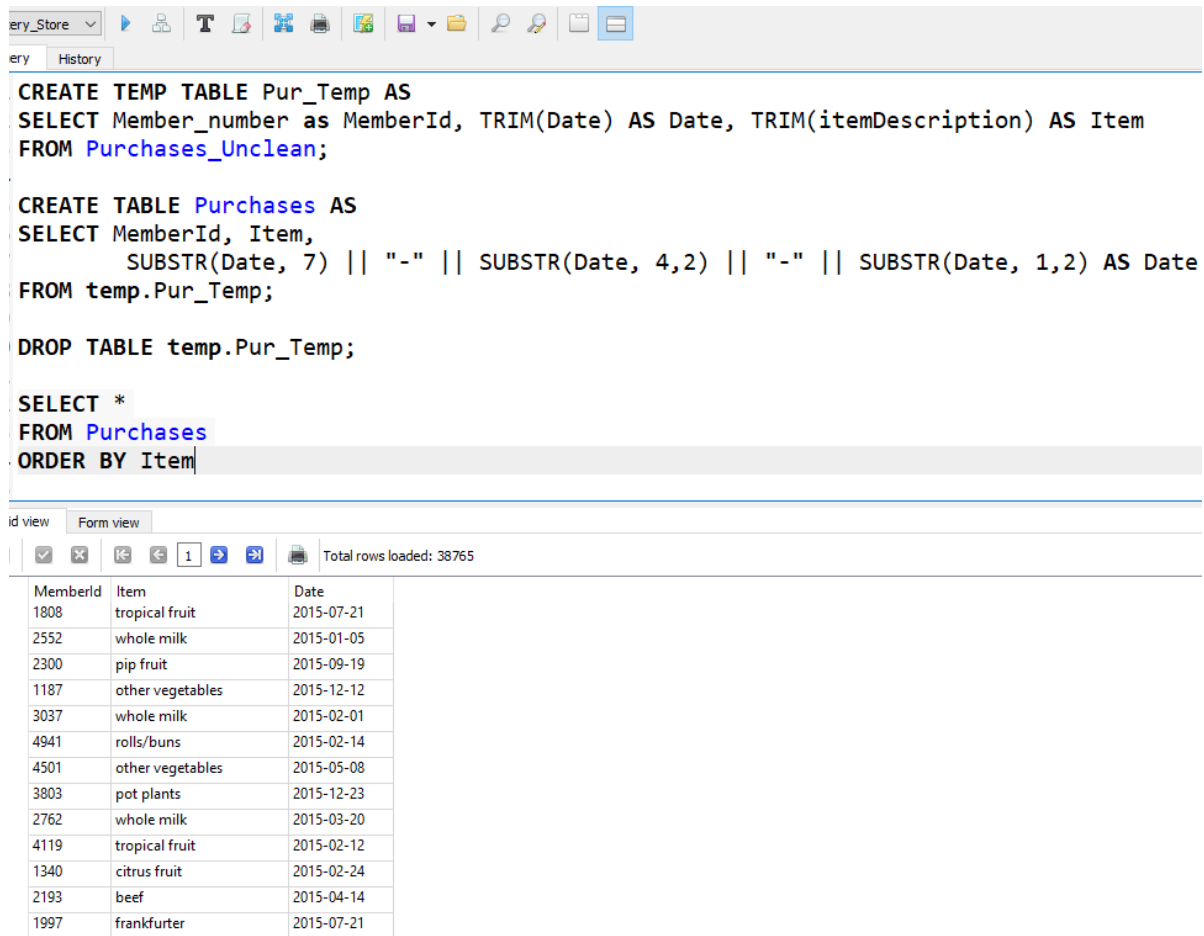


Question 1

I. Cleaning and creating the table for analysis

Going through the data, I realized there was no missing data nor any format inconsistency. Member Ids were all 4 digits and correct. No typo in Item Description was spotted either. Columns Date and Item description were trimmed and the date was changed into yyyy-mm-dd format. Additionally, Column names were standardized.



```
CREATE TEMP TABLE Pur_Temp AS
SELECT Member_number as MemberId, TRIM(Date) AS Date, TRIM(itemDescription) AS Item
FROM Purchases_Unclean;

CREATE TABLE Purchases AS
SELECT MemberId, Item,
       SUBSTR(Date, 7) || "-" || SUBSTR(Date, 4,2) || "-" || SUBSTR(Date, 1,2) AS Date
FROM temp.Pur_Temp;

DROP TABLE temp.Pur_Temp;

SELECT *
FROM Purchases
ORDER BY Item
```

id view Form view

Total rows loaded: 38765

MemberId	Item	Date
1808	tropical fruit	2015-07-21
2552	whole milk	2015-01-05
2300	pip fruit	2015-09-19
1187	other vegetables	2015-12-12
3037	whole milk	2015-02-01
4941	rolls/buns	2015-02-14
4501	other vegetables	2015-05-08
3803	pot plants	2015-12-23
2762	whole milk	2015-03-20
4119	tropical fruit	2015-02-12
1340	citrus fruit	2015-02-24
2193	beef	2015-04-14
1997	frankfurter	2015-07-21

Question 2

For a RFM analysis Recency, Frequency and Monetary Value should be calculated for each customer.

Monetary Value: Since the data does not include any information on the values of items and the money spent by a customer, Monetary Value can not be calculated as it is normally. I decided to use the number of items bought in total by a customer to calculate Monetary Value.

Recency: It is calculated as $1/(\text{last working date} + 1 \text{ day} - \text{Last purchase date})$

Frequency: This variable equals the number of days at which a customer has made a transaction.

For calculating these variables, First using the following query, Last day of purchase, Frequency and Monetary Value were calculated.

Grocery_Store

Query History

```
1 SELECT MemberId, MAX(Date) AS Last_Purchase_Date, COUNT(DISTINCT Date) AS Frequency, COUNT() AS MonetaryValue
2 FROM Purchases
3 GROUP BY MemberId
```

Grid view Form view

Total rows loaded: 3898

	MemberId	Last_Purchase_Date	Frequency	MonetaryValue
1	1000	2015-11-25	5	13
2	1001	2015-05-02	5	12
3	1002	2015-08-30	4	8
4	1003	2015-02-10	4	8
5	1004	2015-12-02	8	21
6	1005	2014-01-23	2	4
7	1006	2015-06-14	4	15
8	1008	2015-10-03	2	12
9	1009	2015-10-05	4	9
10	1010	2015-07-31	5	12
11	1011	2015-12-09	3	13
12	1012	2015-11-19	4	11
13	1013	2015-10-02	8	19
14	1014	2015-10-03	4	10
15	1015	2015-05-04	3	7
16	1016	2015-10-05	4	11
17	1017	2015-09-30	5	11
18	1018	2015-05-23	4	8
19	1019	2014-12-10	1	2
20	1020	2015-10-13	3	10
21	1021	2015-11-07	3	8
22	1022	2014-06-07	3	6
23	1023	2015-06-28	6	17
24	1024	2015-08-10	1	4
25	1025	2015-08-19	3	6
26	1026	2015-05-28	5	17
27	1027	2015-05-17	4	9

Using the results, a data frame is created to hold the final RFM values.

```
In [11]: 1 Customers_RFM_2 = Customers_RFM_Unscored.copy()
2 Customers_RFM_2.columns = ['Recency', 'Frequency', 'Monetary']
3 Customers_RFM_2['Recency'] = 1/(last_day - Customers_RFM_2['Recency']).dt.days
4 Customers_RFM_2.head(10)
```

Out[11]:

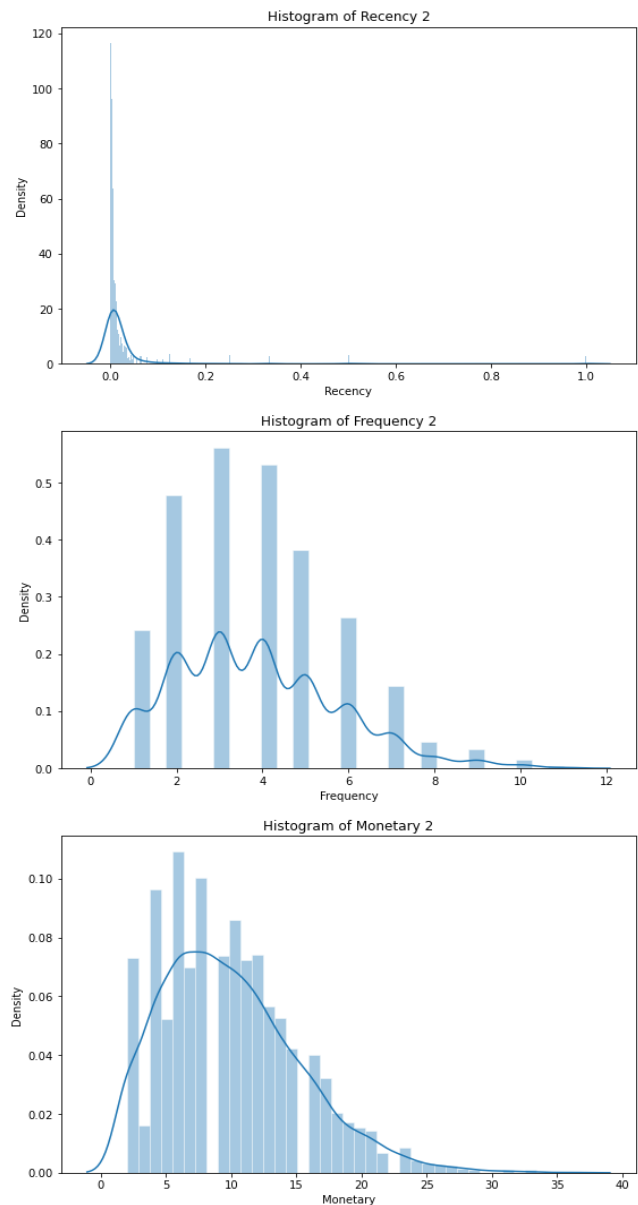
	Recency	Frequency	Monetary
MemberId			
1000	0.027778	5	13
1001	0.004115	5	12
1002	0.008130	4	8
1003	0.003086	4	8
1004	0.034483	8	21
1005	0.001414	2	4
1006	0.005000	4	15
1008	0.011236	2	12
1009	0.011494	4	9
1010	0.006536	5	12

Histogram plot is plotted for the RFM values. It is illustrated on the right side of the page.

Question 3

Customer segmentation with k-means clustering algorithm is done at this stage of the analysis. Looking at the data we see that RFM scores are not distributed normally and are unsymmetrical. Considering the fact that a k-means clustering requires the data to be normally distributed we are to handle this problem.

There are multiple mathematical transformations that can be used to make the data more symmetrical. Considering the way our data is distributed we notice that it resembles the power-law distribution and one approach to make the data symmetrical is using the Box-Cox Transformation. The code below is used to apply the transformation on the data and the next picture illustrates the data after the transformation.



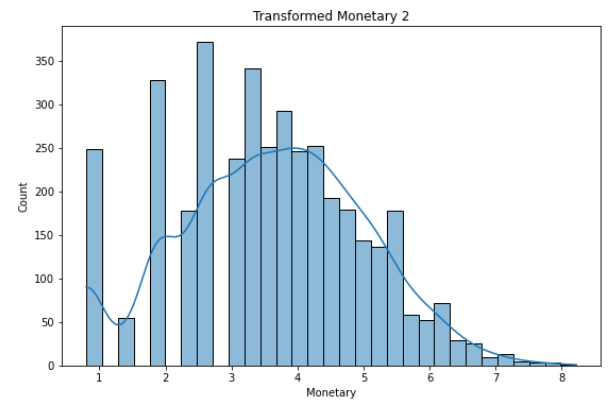
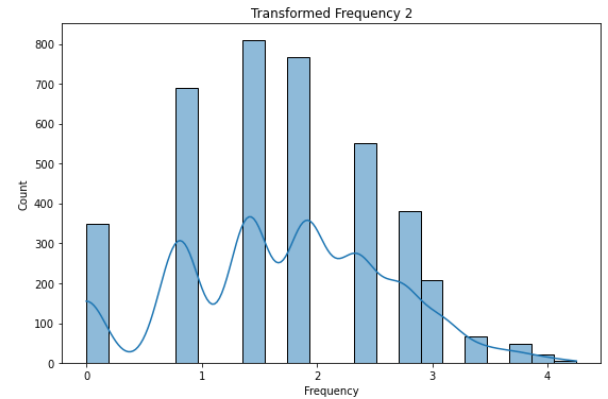
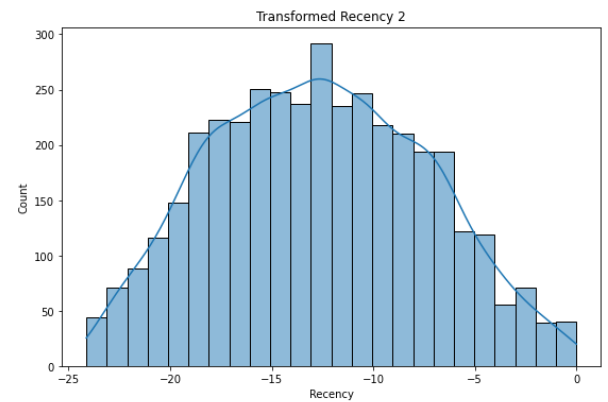
```
In [23]: 1 customers_RFM_Transformed_2 = pd.DataFrame(index= Customers_RFM_2.index)
          2 customers_RFM_Transformed_2["Recency"] = stats.boxcox(Customers_RFM_2['Recency'])[0]
          3 customers_RFM_Transformed_2["Frequency"] = stats.boxcox(Customers_RFM_2['Frequency'])[0]
          4 customers_RFM_Transformed_2["Monetary"] = stats.boxcox(Customers_RFM_2['Monetary'])[0]
          5 customers_RFM_Transformed_2.tail()
```

Out[23]:

	Recency	Frequency	Monetary
MemberId			
4996	-7.016235	1.412097	3.847542
4997	-1.763905	0.810273	2.654617
4998	-9.852175	0.000000	0.802660
4999	-2.132343	2.725071	5.191431
5000	-17.690835	1.412097	2.988455

As we can see the data has been altered so that they are more symmetrical. However, the data are not ready yet for the k-means clustering. The standard deviation and mean for RFM are not the same and this can lead to one variable having more weight on the clustering results which is not favorable in our case.

In order to normalize the data, the following code is run. On the next page we can see how it affects the data.



```
In [36]: 1 scaler = StandardScaler()
2
3 scaler.fit(customers_RFM_Transformed_2)
4 customers_RFM_Normalised_2 = df(scaler.transform(customers_RFM_Transformed_2), index=customers_RFM_Transformed_2.index, columns=customers_RFM_Transformed_2.columns)
5 customers_RFM_Normalised_2.head(10)
```

```
Out[36]:
```

	Recency	Frequency	Monetary
MemberId			
1000	1.083711	0.700267	0.679759
1001	-0.588682	0.700267	0.518780
1002	0.132977	0.224643	-0.219648
1003	-0.946600	0.224643	-0.219648
1004	1.214202	1.868441	1.764258
1005	-2.110171	-0.987600	-1.227829
1006	-0.365227	0.224643	0.981175
1008	0.422372	-0.987600	0.518780
1009	0.441547	0.224643	-0.017743
1010	-0.080704	0.700267	0.518780

As we can see, the data are normalized

The mean of RFM is:

Recency 0.0

Frequency -0.0

Monetary 0.0

dtype: float64

The std of RFM is:

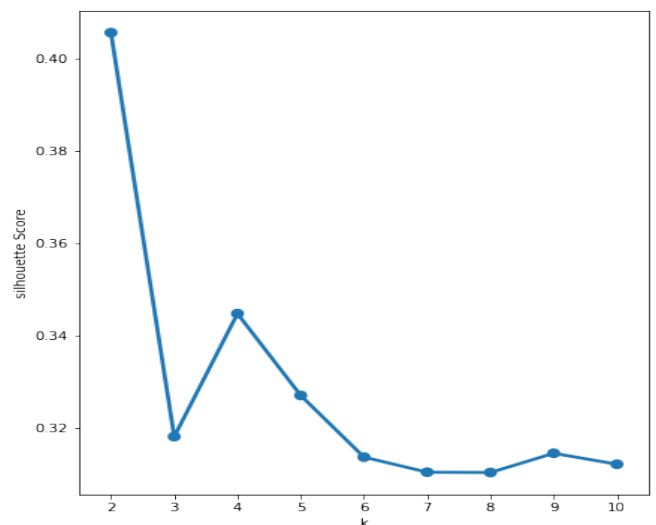
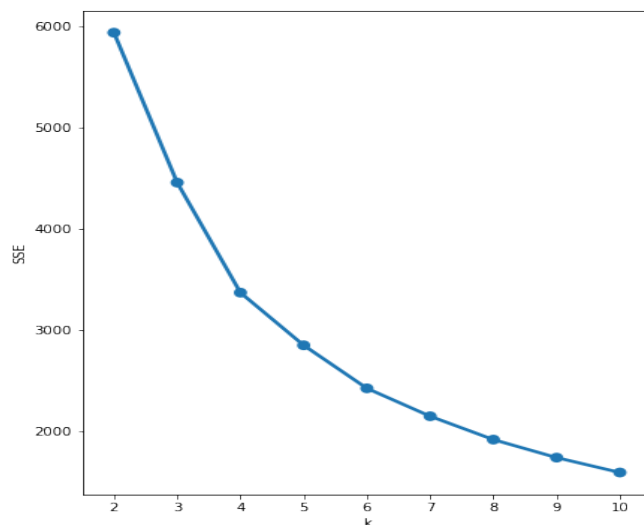
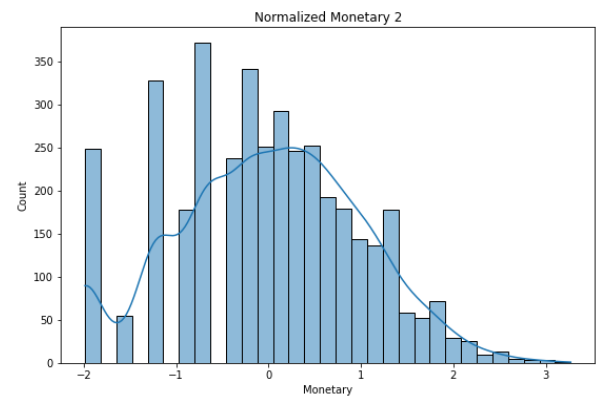
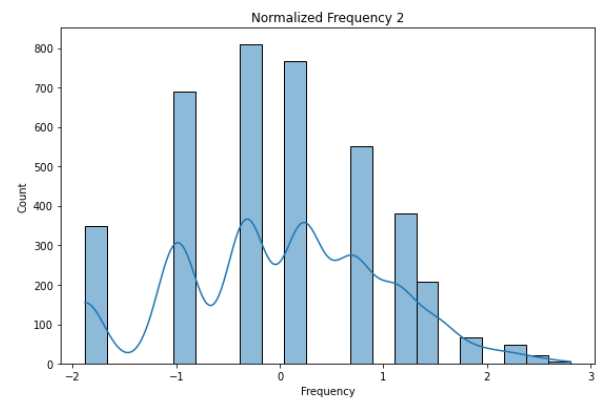
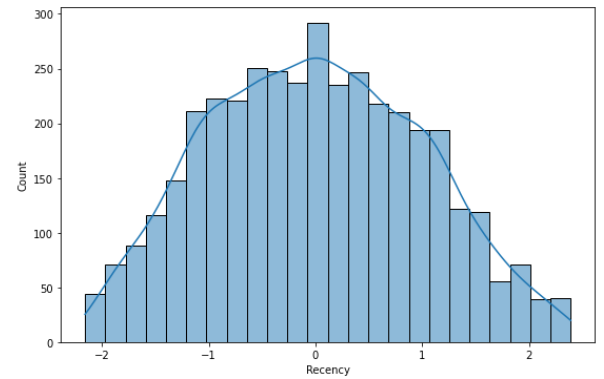
Recency 1.0

Frequency 1.0

Monetary 1.0

dtype: float64

Finally, the data are ready to be clustered using k-means. First, we must find the optimal value for k. For this analysis I have calculated WCSS and Silhouette Score for the k values 1 to 10. The values are illustrated below. Using the elbow method, we can see WCSS line is broken in k = 4 and also k = 6. For deciding which k value to choose I used silhouette score. As we can notice this measurement favors k = 4. Therefore, for our analysis k value will be set as 4.

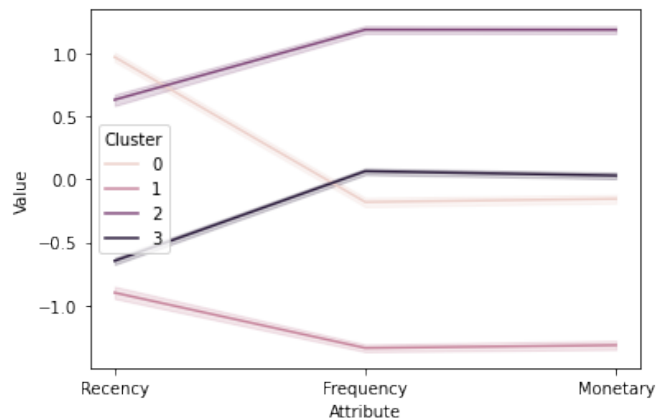


```
In [72]: 1 model = KMeans(n_clusters=4, random_state=300)
2 model.fit(customers_RFM_Normalised_2)
3 Customers_RFM_Clustered_2 = Customers_RFM_2.copy()
4 Customers_RFM_Clustered_2["Cluster"] = model.labels_
5 Customers_RFM_Clustered_2.groupby('Cluster').agg({
6 'Recency': 'mean',
7 'Frequency': 'mean',
8 'Monetary': ['mean', 'count']})
9
```

B:\Software\Anaconda\lib\site-packages\sklearn\cluster_kme from 10 to 'auto' in 1.4. Set the value of 'n_init' explici warnings.warn(

Out[72]:

	Recency	Frequency	Monetary	
	mean	mean	mean	count
Cluster				
0	0.061044	3.325397	8.563492	882
1	0.004143	1.607101	3.895858	845
2	0.038413	6.231898	16.707436	1022
3	0.004376	3.744996	9.438642	1149



The image on the left showcases the clustering code and its resulting clusters, while the image on the right displays a snake plot, a valuable tool for result interpretation. From the analysis, we can observe the presence of four distinct clusters. Two clusters exhibit low recency, as well as average to low frequency and monetary values. In contrast, the remaining cluster demonstrates high recency, along with either average or high frequency and monetary values.

Question 4

According to the clusters and their RFM values we can interpret them in the following way.

Cluster 0: These customers have a high recency value which indicates they have made a purchase recently. The average values for frequency and monetary value. This cluster contains customers that have mild loyalty to the store and they have recently made purchases therefore they are potential highly valued customers that can be incentivized into most loyal customers using promotions and maybe loyalty privileges.

Cluster 1: Low value for all 3 variables. These customers are the least worthy ones for the business and are expendable. Allocating resources for them would be least efficient.

Cluster 2: These customers are the most valuable and loyal customers for the have purchased goods from the store many times and also their last transaction was not a long ago. Budget should be allocated for these people so that they stay loyal and keep coming back. Informing them about the goods can be a good idea.

Cluster 3: The customers in this cluster are the ones at risk. Although they have average frequency and monetary value which indicates their profitability, their low recency means they have not purchased anything recently. This group should both be surveyed on why they have not made any purchase recently and also incentivized to come back to the store.

Question 5

The diagram illustrates a possible attribute tree for our data.

Transaction holds every transaction any customer has made with store, its date, value and items.

Cluster describes the results of the RFM analysis and each customer type.

Considering the nature of transaction entity which is dynamic, it is the best fact choice for neither of the others are so.

Dimension will be Date, Item, Customer and Cluster because they are almost static and do not change very much and can be used to filter and query transactions according to business needs.

Measures for the data warehouse are number of Items bought, the amount paid and total amount of a transaction. These values are measurements for every record of transaction.

Fact: Transaction

Dimensions: Date, Item, Customer, Cluster

Measures: Total price, Paid value, number of each purchased item.

