Assignment-5

Problem Statement 1: Write Load the "Country-data.csv" dataset into a DataFrame and perform the following tasks:

1. Create a seperate DataFrame with only numeric data by remove the "country" column
2. Scale the data using the Standard Scaler to create a scaled DataFrame
3. Plotting dendograms with the complete linkage method
4. Creating cluster labels using cut tree
5. Perform the 4-Component PCA on DataFrame
6. Now, from final the DataFrame, analyze how low GDP rate corresponds to the child mortality rate around the world

Example:

Dataset:

	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200

Output:

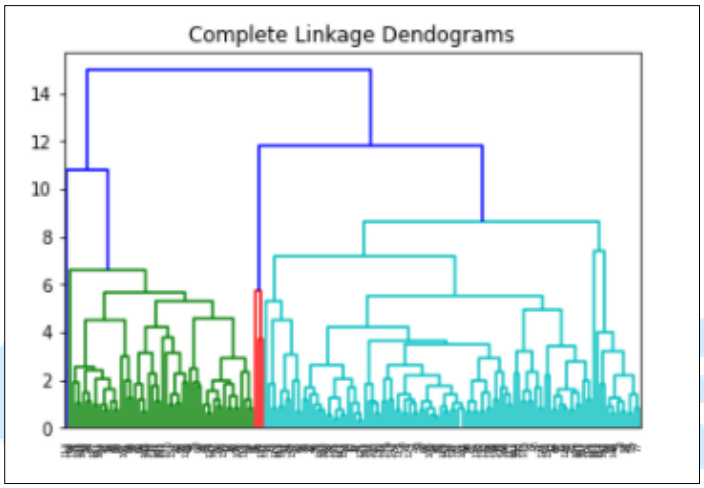
1. Create a seperate DataFrame with only numeric data by remove the "country" column

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
4	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200

2. Scale the data using the Standard Scaler to create a scaled DataFrame

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	1.291532	-1.138280	0.279088	-0.082455	-0.808245	0.157336	-1.619092	1.902882	-0.679180
1	-0.538949	-0.479658	-0.097016	0.070837	-0.375369	-0.312347	0.647866	-0.859973	-0.485623
2	-0.272833	-0.099122	-0.966073	-0.641762	-0.220844	0.789274	0.670423	-0.038404	-0.465376
3	2.007808	0.775381	-1.448071	-0.165315	-0.585043	1.387054	-1.179234	2.128151	-0.516268
4	-0.695634	0.160668	-0.286894	0.497568	0.101732	-0.601749	0.704258	-0.541946	-0.041817
...	...	...	...	...	...	...	...	...	...
162	-0.225578	0.200917	-0.571711	0.240700	-0.738527	-0.489784	-0.852161	0.365754	-0.546913
163	-0.526514	-0.461363	-0.695862	-1.213499	-0.033542	3.616865	0.546361	-0.316678	0.029323
164	-0.372315	1.130305	0.008877	1.380030	-0.658404	0.409732	0.286958	-0.661206	-0.637754
165	0.448417	-0.406478	-0.597272	-0.517472	-0.658924	1.500916	-0.344633	1.140944	-0.637754
166	1.114951	-0.150348	-0.338015	-0.662477	-0.721358	0.590015	-2.092785	1.624609	-0.629546

3. Plotting dendograms with the complete linkage method



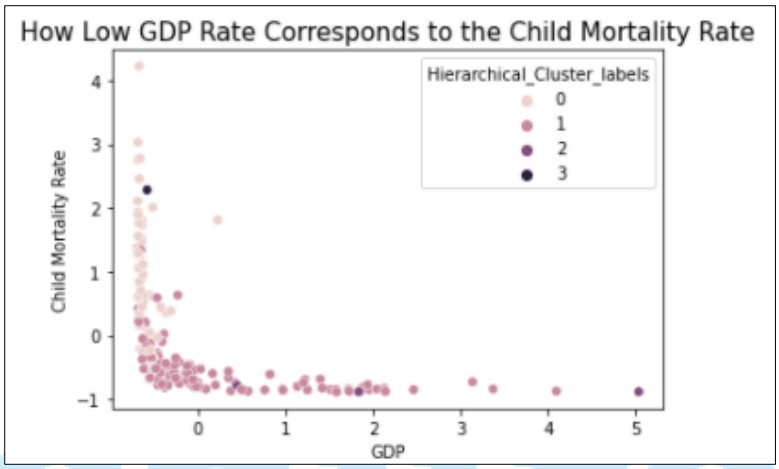
4. Creating cluster labels using cut tree

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp	Hierarchical_Cluster_labels
0	1.291532	-1.138280	0.279088	-0.082455	-0.808245	0.157336	-1.619092	1.902882	-0.679180	0
1	-0.538949	-0.479658	-0.097016	0.070837	-0.375369	-0.312347	0.647866	-0.859973	-0.485623	1
2	-0.272833	-0.099122	-0.966073	-0.641762	-0.220844	0.789274	0.670423	-0.038404	-0.465376	1
3	2.007808	0.775381	-1.448071	-0.165315	-0.585043	1.387054	-1.179234	2.128151	-0.516268	0
4	-0.695634	0.160668	-0.286894	0.497568	0.101732	-0.601749	0.704258	-0.541946	-0.041817	1
...	...	...	...	...	...	...	...	...	...	...
162	-0.225578	0.200917	-0.571711	0.240700	-0.738527	-0.489784	-0.852161	0.365754	-0.546913	0
163	-0.526514	-0.461363	-0.695862	-1.213499	-0.033542	3.616865	0.546361	-0.316678	0.029323	1
164	-0.372315	1.130305	0.008877	1.380030	-0.658404	0.409732	0.286958	-0.661206	-0.637754	1
165	0.448417	-0.406478	-0.597272	-0.517472	-0.658924	1.500916	-0.344633	1.140944	-0.637754	0
166	1.114951	-0.150348	-0.338015	-0.662477	-0.721358	0.590015	-2.092785	1.624609	-0.629546	0

5. Perform the 4-Component PCA on DataFrame

	PC1	PC2	PC3	PC4	Hierarchical_Cluster_Labels
0	-2.913787	0.088354	0.721003	0.996699	0
1	0.429358	-0.587859	0.321052	-1.171193	1
2	-0.282988	-0.446657	-1.225135	-0.850127	1
3	-2.930969	1.699437	-1.521734	0.875966	0
4	1.031988	0.130488	0.192922	-0.844808	1

6. Now, from final the DataFrame, analyze how low GDP rate corresponds to the child mortality rate around the world



Problem Statement 2: Write a Python program that reads the *"Credit Card Customer Data.csv"* (provided on LMS) The following are the tasks that need to be taken into consideration while constructing the solution to Segregate customers based on the data provided with the help of k-means clustering.

Tasks to be performed:

1. Load the Given CSV file into a DataFrame
2. Find missing values and drop the unnecessary columns
3. Univariate and bivariate analysis
4. Standardize the whole dataset
5. Find the within-cluster sum of square
6. Find silhouette score
7. Use a line plot using matplotlib to find scores for different sizes of K and choose the best size for the cluster and build the final model
8. Observe Cluster behavior with different columns
9. Print Co-ordinates of all centroids and silhouette scores for the final model

Example:

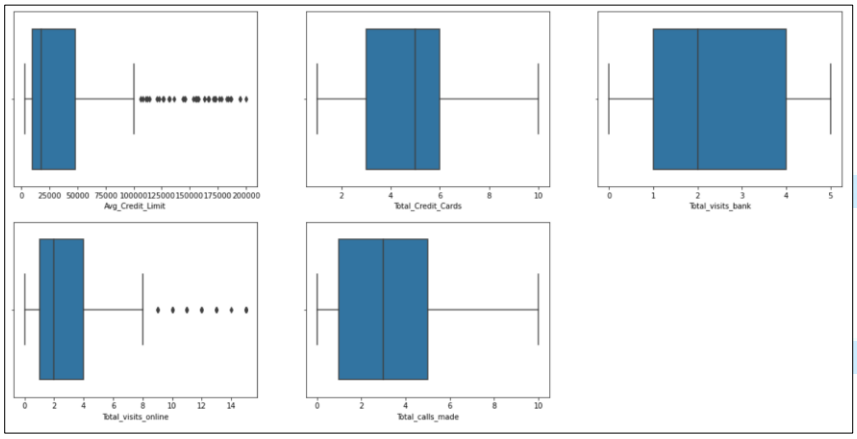
Dataset:

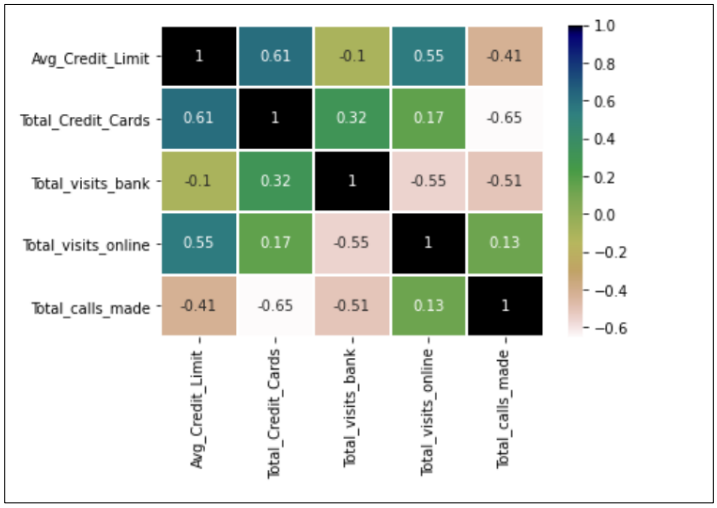
	Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
0	1	87073	100000	2	1	1	0
1	2	38414	50000	3	0	10	9
2	3	17341	50000	7	1	3	4
3	4	40496	30000	5	1	1	4
4	5	47437	100000	6	0	12	3
...	...	...	...	...	...	...	...
655	656	51108	99000	10	1	10	0
656	657	60732	84000	10	1	13	2
657	658	53834	145000	8	1	9	1
658	659	80655	172000	10	1	15	0
659	660	80150	167000	9	0	12	2

660 rows × 7 columns

Output:

3. Univariate and bivariate analysis



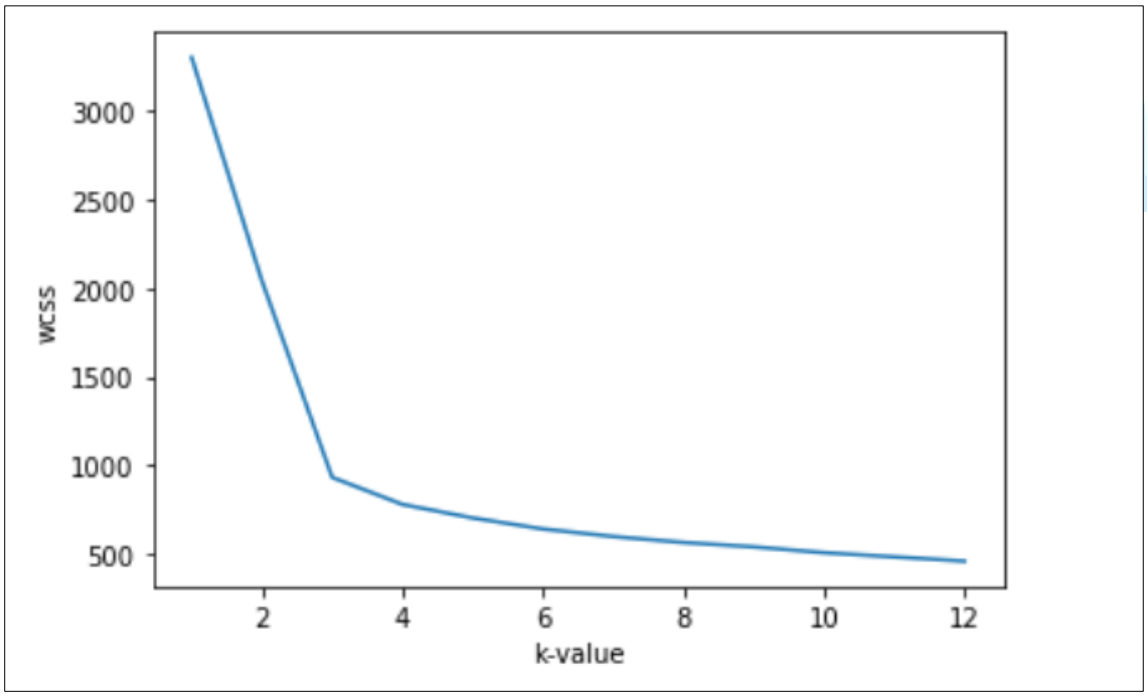


4. Standardize the whole dataset

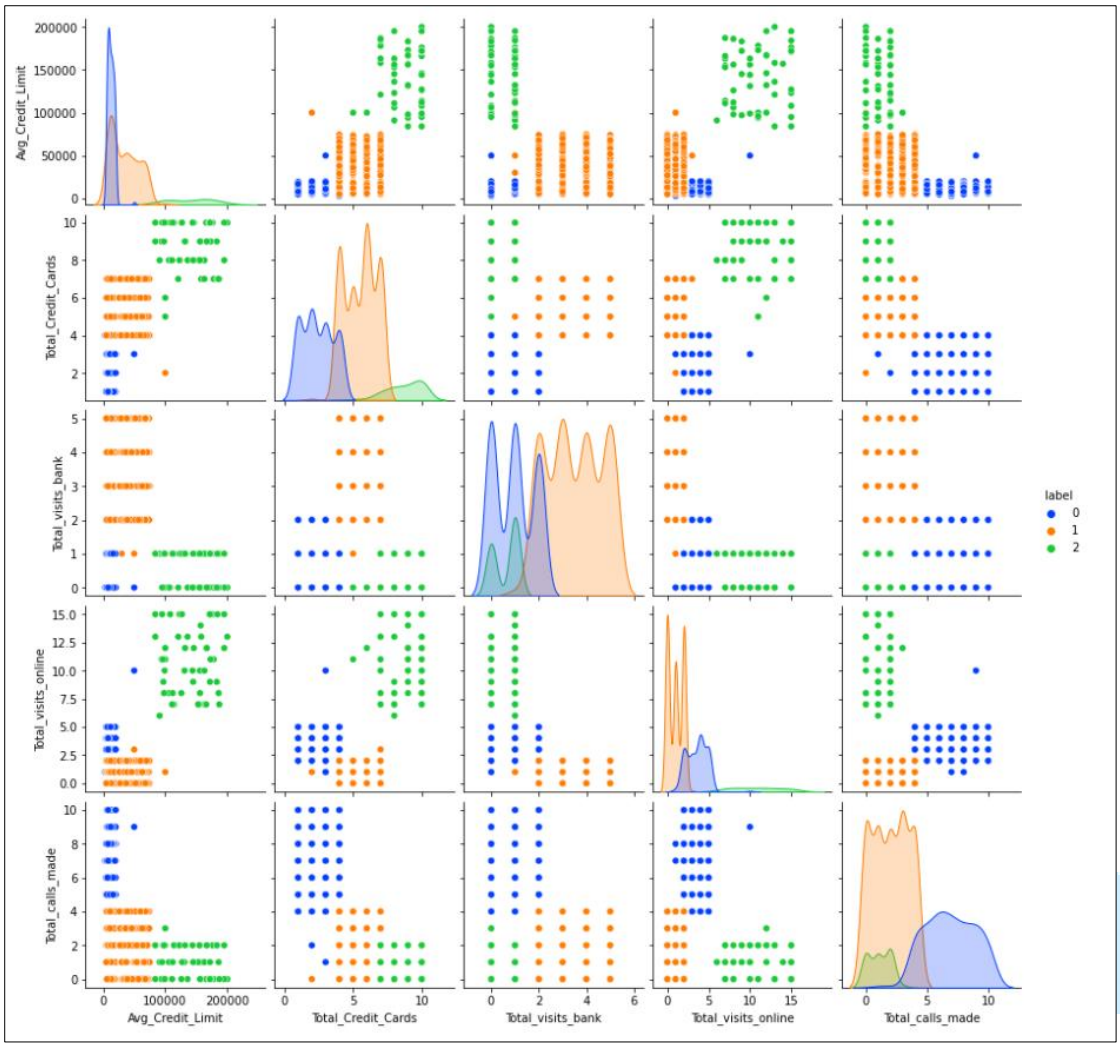
	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
0	1.740187	-1.249225	-0.860451	-0.547490	-1.251537
1	0.410293	-0.787585	-1.473731	2.520519	1.891859
2	0.410293	1.058973	-0.860451	0.134290	0.145528
3	-0.121665	0.135694	-0.860451	-0.547490	0.145528
4	1.740187	0.597334	-1.473731	3.202298	-0.203739
...	...	...	...	...	...
655	1.713589	2.443892	-0.860451	2.520519	-1.251537
656	1.314621	2.443892	-0.860451	3.543188	-0.553005
657	2.937092	1.520613	-0.860451	2.179629	-0.902271
658	3.655235	2.443892	-0.860451	4.224968	-1.251537
659	3.522245	1.982253	-1.473731	3.202298	-0.553005

660 rows × 5 columns

7. Plot the score for different sizes of K and choose the best size for the cluster and build the final model



8. Observe Cluster behavior with different columns



9. Print Co-ordinates of all centroids and silhouette scores for the final model

Co ordinates

```
array([[-0.59579625, -1.05962278, -0.9015185 ,  0.32299678,  1.14810882],
       [-0.02106178,  0.37368962,  0.6663945 , -0.55367163, -0.55300488],
       [ 2.83176409,  1.86222621, -1.10576269,  2.82731942, -0.87432983]])
```

Final scores

```
0.5157182558881063
```

Problem Statement 3: DBSCAN Clustering

Load the "Mall_Customers.csv" dataset into a DataFrame to perform the following tasks:

1. Find the correlation among the all the columns and drop the column/s with the least correlation
2. Encode the "Gender" column using get_dummies() function
3. Perform Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering with eps=12.5 and min_samples=4
4. Print the size of each cluster and also the size of outliers' cluster
5. Using a scatter plot shows how annual income corresponds to the spending rates of customers

Example:

Dataset:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Output:

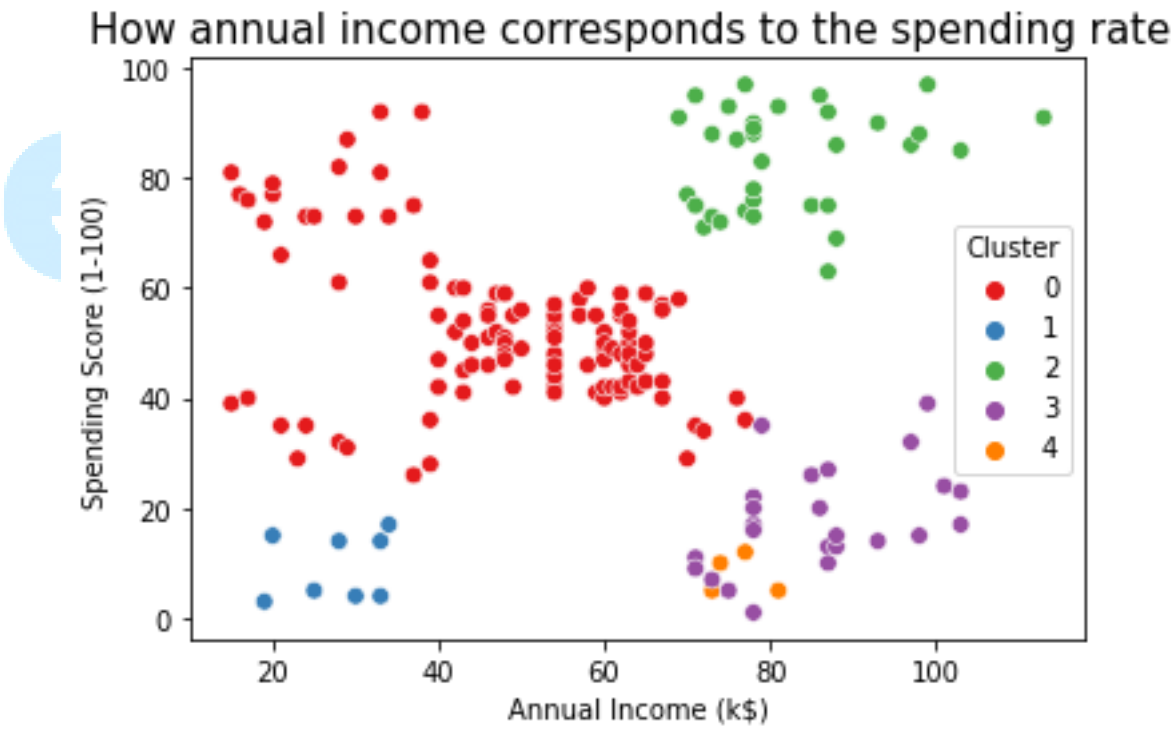
1. Find the correlation among the all the columns and drop the column/s with the least correlation

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
CustomerID	1.000000	-0.026763	0.977548	0.013835
Age	-0.026763	1.000000	-0.012398	-0.327227
Annual Income (k$)	0.977548	-0.012398	1.000000	0.009903
Spending Score (1-100)	0.013835	-0.327227	0.009903	1.000000

2. Encode the "Gender" column using get_dummies() function

	Age	Annual Income (k$)	Spending Score (1-100)	Gender_Female	Gender_Male
0	19	15	39	0	1
1	21	15	81	0	1
2	20	16	6	1	0
3	23	16	77	1	0
4	31	17	40	1	0

5.Using a scatter plot shows how annual income corresponds to the spending rates of customers



Problem Statement 4: Write a Python program that reads the Groceries data.csv (provided on LMS) file into a DataFrame. The following are the tasks that need to be taken into consideration while constructing the solution to using the apriori algorithm and list out items that are sold most frequently with other items. Dataset file contains tabular data, where it has items, date, member number, day of the month, day of the week, etc.

Tasks to be performed:

1. Install mlxtend library for further process
2. Load the Groceries data.csv data into a Data frame
3. Print customer data where the member number is 1001
4. Create a new column as “item count”, and give the count as 1(because all customers bought 1 item on each day only)
5. Drop unnecessary columns like "month","day","year","day_of_week"
6. Create a new data frame where all data is grouped by member id and items they bought and set their value as item count.
7. Use the Apriori algorithm and generate frequent itemsets that have the support of at least 7%
8. Generating the rules with their corresponding support, confidence, and lift
9. Filtering out the values with lift > = 1 and confidence > = 0.5

Example:

Dataset:

	Member_number	Date	itemDescription	year	month	day	day_of_week
0	1808	2015-07-21	tropical fruit	2015	7	21	1
1	2552	2015-05-01	whole milk	2015	5	1	4
2	2300	2015-09-19	pip fruit	2015	9	19	5
3	1187	2015-12-12	other vegetables	2015	12	12	5
4	3037	2015-01-02	whole milk	2015	1	2	4
...	...	...	...	...	...	...	...
38760	4471	2014-08-10	sliced cheese	2014	8	10	6
38761	2022	2014-02-23	candy	2014	2	23	6
38762	1097	2014-04-16	cake bar	2014	4	16	2
38763	1510	2014-03-12	fruit/vegetable juice	2014	3	12	2
38764	1521	2014-12-26	cat food	2014	12	26	4

38765 rows × 7 columns

Output:

3. Print customer data where the member number is 1001

	Member_number	Date	itemDescription	year	month	day	day_of_week
364	1001	2015-01-20	frankfurter	2015	1	20	1
5695	1001	2015-02-05	frankfurter	2015	2	5	3
6612	1001	2015-04-14	beef	2015	4	14	1
9391	1001	2014-07-02	sausage	2014	7	2	2
11046	1001	2014-12-12	whole milk	2014	12	12	4
16513	1001	2015-01-20	soda	2015	1	20	1
21844	1001	2015-02-05	curd	2015	2	5	3
22761	1001	2015-04-14	white bread	2015	4	14	1
25540	1001	2014-07-02	whole milk	2014	7	2	2
27195	1001	2014-12-12	soda	2014	12	12	4
32575	1001	2015-01-20	whipped/sour cream	2015	1	20	1
32727	1001	2014-07-02	rolls/buns	2014	7	2	2

5. Drop unnecessary columns like "month","day","year","day_of_week"

	Member_number	Date	itemDescription	item_count
0	1808	2015-07-21	tropical fruit	1
1	2552	2015-05-01	whole milk	1
2	2300	2015-09-19	pip fruit	1
3	1187	2015-12-12	other vegetables	1
4	3037	2015-01-02	whole milk	1
...	...	...	...	...
38760	4471	2014-08-10	sliced cheese	1
38761	2022	2014-02-23	candy	1
38762	1097	2014-04-16	cake bar	1
38763	1510	2014-03-12	fruit/vegetable juice	1
38764	1521	2014-12-26	cat food	1

38765 rows × 4 columns

6. Create a new data frame where all data is a group by member id and items they bought and set their value as item count

itemDescription	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	bags	baking powder	bathroom cleaner	beef	berries	...	turkey	vinegar	waffles	whipped/sour cream	whisky	whisky	white bread	white wine	whole milk	yogurt
Member_number																					
1000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.0
1001	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	2.0	0.0
1002	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1003	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1004	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0
4998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
5000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

3898 rows × 167 columns

9. Filtering out the values with lift > = 1 and confidence > = 0.5

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
1	(bottled beer)	(whole milk)	0.158799	0.458184	0.085428	0.537964	1.174124	0.012669	1.172672
9	(bottled water)	(whole milk)	0.213699	0.458184	0.112365	0.525810	1.147597	0.014452	1.142615
11	(canned beer)	(whole milk)	0.165213	0.458184	0.087224	0.527950	1.152268	0.011526	1.147795
19	(domestic eggs)	(whole milk)	0.133145	0.458184	0.070292	0.527938	1.152242	0.009287	1.147766
21	(newspapers)	(whole milk)	0.139815	0.458184	0.072345	0.517431	1.129310	0.008284	1.122775