Project - 2

Name: Naga Hemachand Chinta
Email id: chint068@umn.edu
Experiment-1:

1. The number of records in the dataset are 7501

Importing the dataset

In [3]:
```
# reading the dataset

data = pd.read_csv('store_transaction.csv', header = None)

# Let's check the shape of the dataset
data.shape
```
Out[3]:
```
(7501, 20)
```

2. In 1111 transaction, a customer has bought 19 items being the maximum number of items bought by a customer.

In [9]:
```
data.count(axis=1).nlargest(2)
```
Out[9]:
```
0       20
1111    19
dtype: int64
```

3.

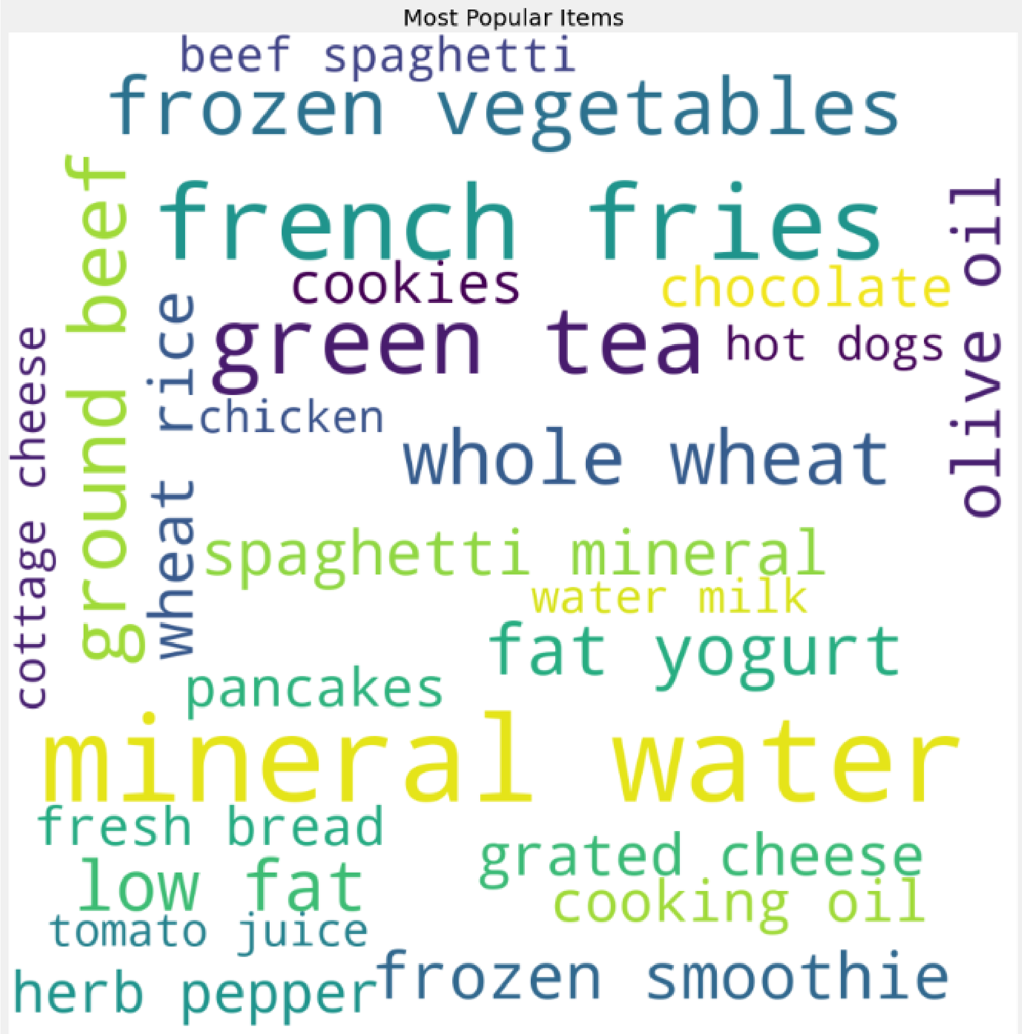
Transaction	Items
Transaction-1	burgers, meatballs, eggs
Transaction-2	Chutney
Transaction-3	turkey, avocado
Transaction-4	low fat yogurt
Transaction-5	eggs, pet food

These are random five transactions of the customers. Since the data contains the information about 7500 transactions of different customers buying different items from the store, it is not possible from the given data to tell different transactions of one particular customer but any five transactions of random customers can be seen with the given data.

Project - 2

4. When the max words is set to 25

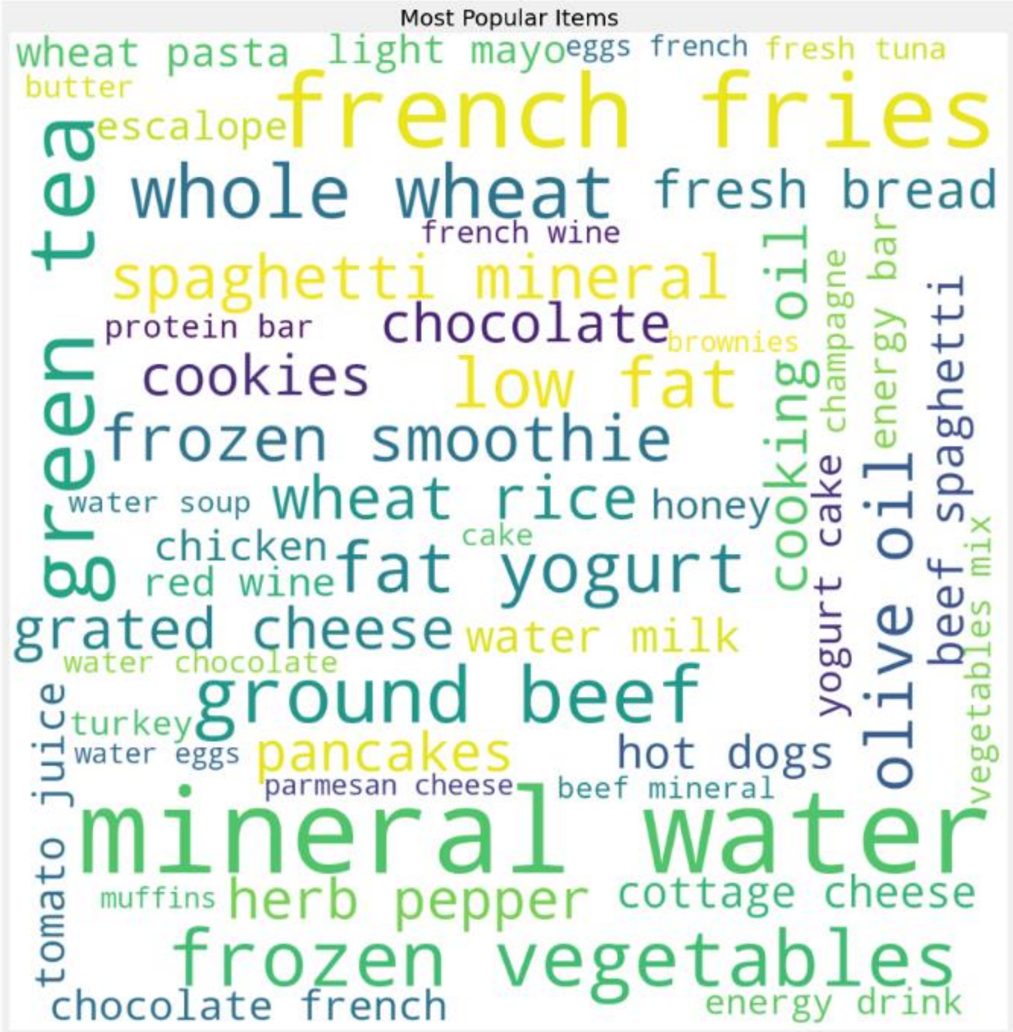
```
plt.rcParams['figure.figsize'] = (15, 15)
wordcloud = WordCloud(background_color = 'white', width = 1200,  height = 1200, max_words = 25, regexp = r"\w[\w]+").generate(st
plt.imshow(wordcloud)
plt.axis('off')
plt.title('Most Popular Items',fontsize = 20)
plt.show()
```



Project - 2

When the max words is set to 50

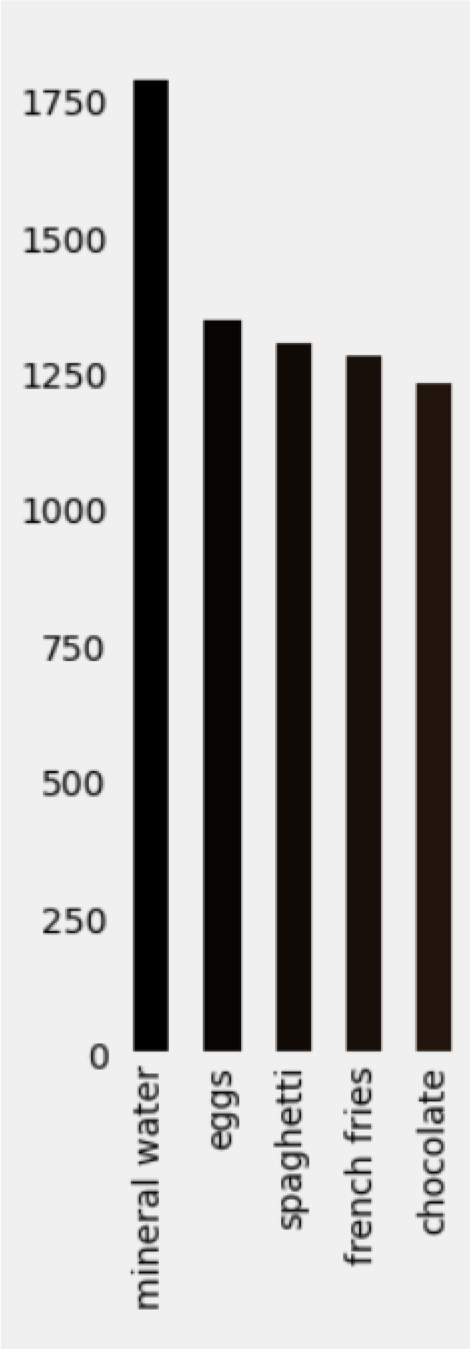
```
plt.rcParams['figure.figsize'] = (15, 15)
wordcloud = WordCloud(background_color = 'white', width = 1200,  height = 1200, max_words = 50, regexp = r"\w[\w]+").generate(st
plt.imshow(wordcloud)
plt.axis('off')
plt.title('Most Popular Items',fontsize = 20)
plt.show()
```



The word cloud is a very popular way to visualize the words with maximum count but as the number of max words limit increase more word clouds will be formed in the image increasing the clutter and making it difficult to analyse and when the limit is too small the information can be sometimes misleading too.

Project - 2

5. Mineral water, eggs, Spaghetti, French fries, chocolate



Project - 2

6.

```
-*-*-*-*-*-*-*-*-*-*
   Apple  Bananas   Beer  Chicken   Milk   Rice    nan
0   True    False   True     True  False   True  False
1   True    False   True    False  False   True   True
2   True    False   True    False  False  False   True
3   True     True  False    False  False  False   True
4  False    False   True     True   True   True  False
5  False    False   True    False   True   True   True
6  False    False   True    False   True  False   True
7   True     True  False    False  False  False   True
```

7. After deleting ‘nan’, there are 120 unique items in the input data.

```
------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7501 entries, 0 to 7500
Columns: 120 entries,  asparagus to zucchini
dtypes: bool(120)
memory usage: 879.1 KB
```

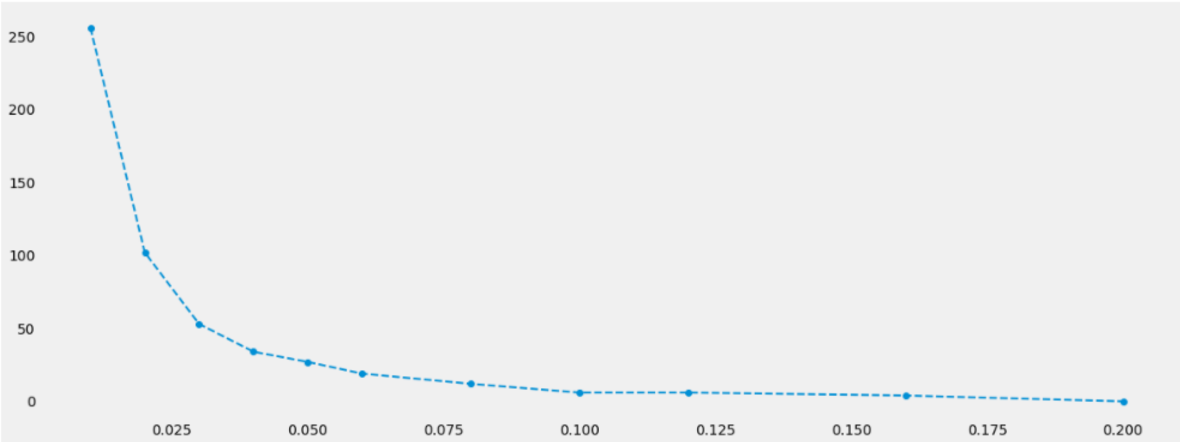
Project - 2

8.

```
minsup_arry = [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.08, 0.1, 0.12, 0.16, 0.2]
num = []
for i in minsup_arry:
    frequent_itemsets = apriori(data, min_support = i, use_colnames=True)
    frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x: len(x))
   # print(frequent_itemsets)
    #print(frequent_itemsets.shape)
    num1 = len(frequent_itemsets.axes[0])
    num.append(num1-1)
    #print("----")
print(minsup_arry)
print(num)

#plot the number of items and the corresponding support threshold
plt.plot(minsup_arry, num, linestyle = "dashed", linewidth =2, marker = 'o', markersize=6)
plt.grid()
plt.show()
```

```
[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.08, 0.1, 0.12, 0.16, 0.2]
[256, 102, 53, 34, 27, 19, 12, 6, 6, 4, 0]
```



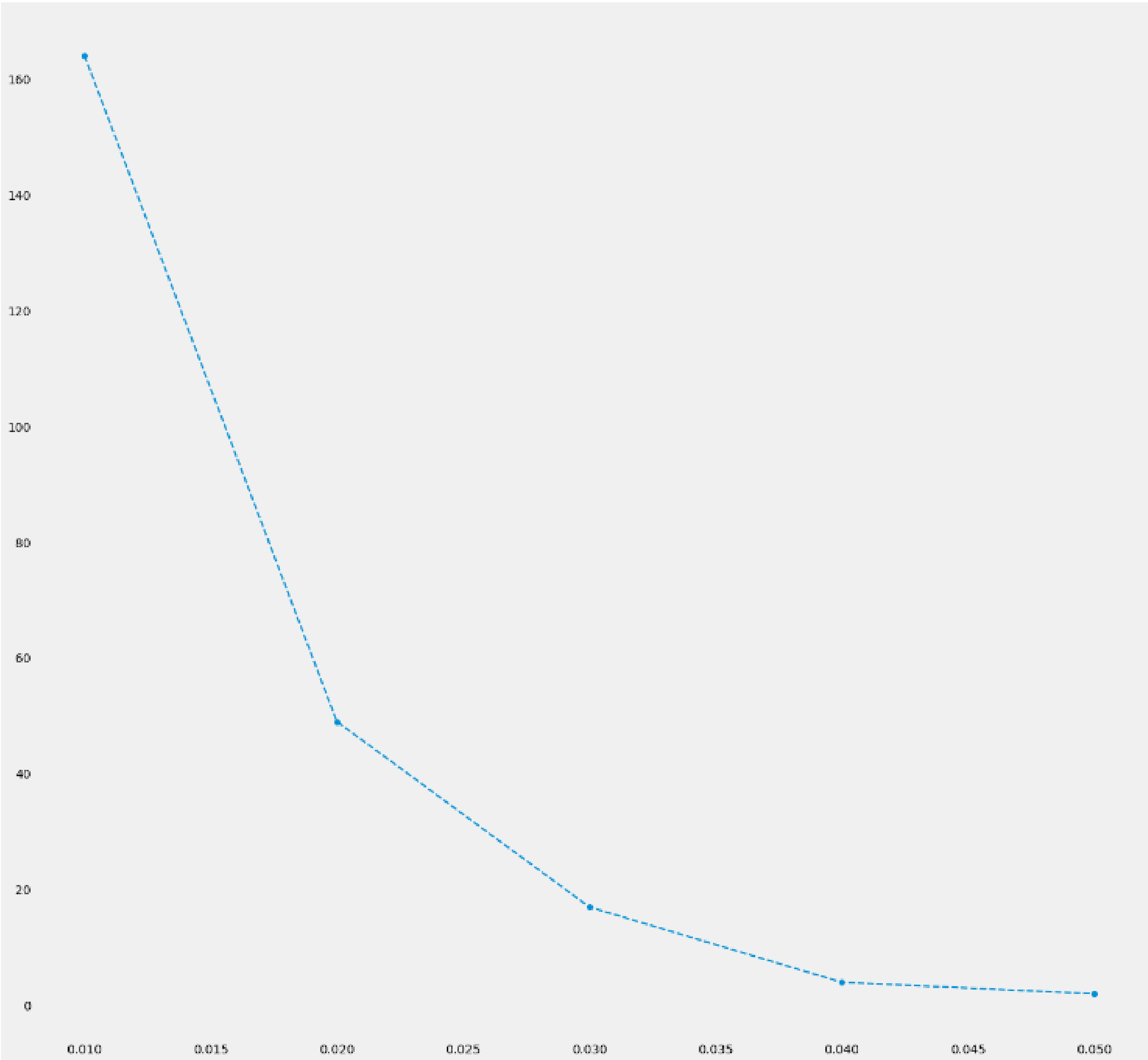
From the above graph, as the support threshold increases the number frequent itemsets are decreasing because the chances for the items to associate together decreases in general as the minimum frequency for the items to exist together is high.

9. For lesser threshold there is more chance for more itemsets to exist together. The subsets of a frequent itemsets should also be frequent, so, one of the three itemsets is failing to exist together when support threshold is set 0.2

Project - 2

10.

```
[0.01, 0.02, 0.03, 0.04, 0.05]
[164, 49, 17, 4, 2]
```



Along with the threshold as the length threshold increase the number items decrease even more because the itemsets beneath the length threshold are not considered.

Project - 2

11.

```
      support        itemsets
46   0.238368  (mineral water)
-----------------------------------
      support    itemsets
13   0.163845  (chocolate)
-----------------------------------
      support itemsets
19   0.179709   (eggs)
-----------------------------------
       support                itemsets
144   0.050927  (mineral water, eggs)
-----------------------------------
      support                     itemsets
118   0.05266  (mineral water, chocolate)
```

Experiment-2:

1. Minimum Threshold = 3
   Apple – 4
   Egg – 3
   Carrot – 3
   Milk – 2
   As per Apriori algorithm, milk is eliminated because it's less than the threshold 3.
   {Apple, Egg} – 3
   {Apple, Carrot} – 2
   {Egg, Carrot} – 1

   As per the Apriori algorithm, only {Apple, Egg} is the one pair which has a minimum threshold of 3.

2.

C. Display summary statistics for order data

```
print('dimensions: {0};   size: {1};   unique_orders: {2};   unique_items: {3}'
      .format(orders.shape, size(orders), len(orders.index.unique()), len(orders.value_counts())))
```

```
dimensions: (32434489,);   size: 518.95 MB;   unique_orders: 3214874;   unique_items: 49677
```

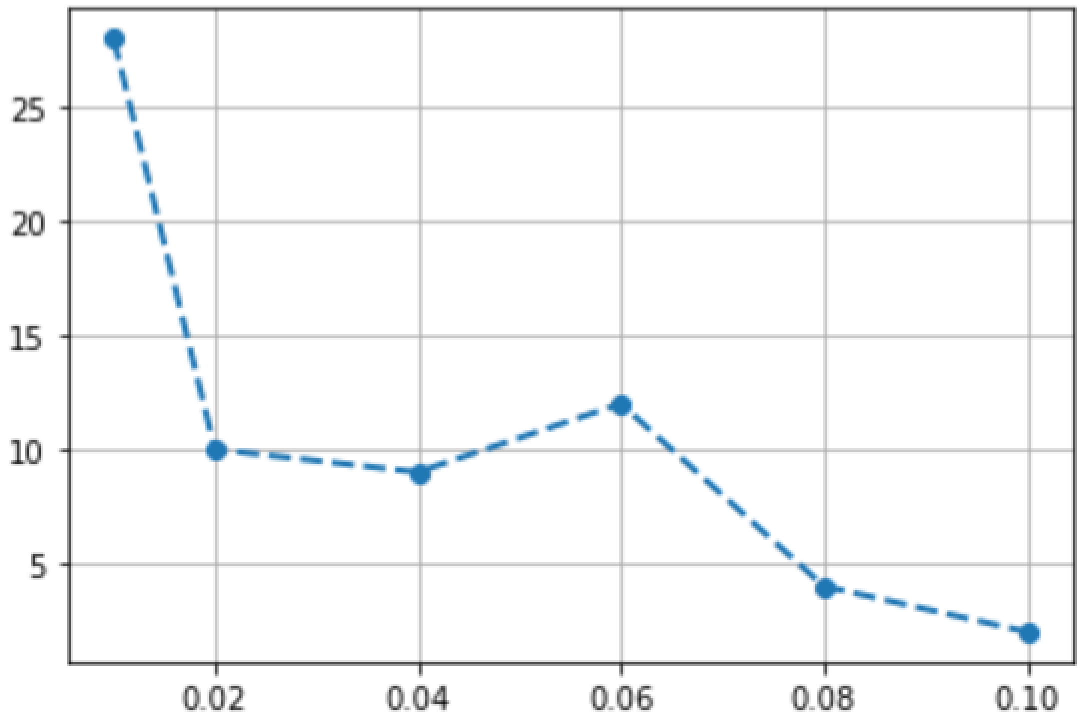
The unique orders are the records without any duplicates.
The average is 2.557634e+04.

Project - 2

```
count    3.243449e+07
mean     2.557634e+04
std      1.409669e+04
min      1.000000e+00
25%      1.353000e+04
50%      2.525600e+04
75%      3.793500e+04
max      4.968800e+04
Name: item_id, dtype: float64
```

3.

```
[0.01, 0.02, 0.04, 0.06, 0.08, 0.1]
[28. 10.  9. 12.  4.  2.]
```



When the threshold is less then the number of associations will be more and consequently the algorithm runtime will be more.

Project - 2

4.

```
%%time
import matplotlib.pyplot as plt
min_sup = [0.01, 0.02, 0.04, 0.06, 0.08, 0.1]
num = []
for i in min_sup:
    rules = association_rules(orders, i)
    print(rules)
    num1 = len(rules)
    num = np.append(num,num1)
    #print(num)
    print("---------")
    print("---------")
    #num=append.(num1)
print(min_sup)
print(num)
#plot the number of items and the corresponding support threshold
plt.plot(min_sup, num, linestyle = "dashed", linewidth =2, marker = 'o', markersize=6)
plt.grid()
plt.show()
```

For min_sup = 0.01

	item_A	item_B	freqAB	supportAB	freqA	supportA	freqB	supportB	\
14	24852	28204	23876	0.014341	401800	0.241341	82313	0.049441	
13	24852	45066	20196	0.012131	401800	0.241341	74704	0.044871	

	confidenceAtoB	confidenceBtoA	lift
14	0.059423	0.290064	1.201882
13	0.050264	0.270347	1.120186

For min_sup = 0.02

	item_A	item_B	freqAB	supportAB	freqA	supportA	freqB	supportB	\
6	24852	16797	29182	0.024398	341573	0.285576	103795	0.086779	
0	13176	27966	25620	0.021420	284048	0.237481	119283	0.099728	

	confidenceAtoB	confidenceBtoA	lift
6	0.085434	0.281150	0.984504
0	0.090196	0.214783	0.904422

For min_sup = 0.04

	item_A	item_B	freqAB	supportAB	freqA	supportA	freqB	supportB	\
6	24852	16797	29182	0.046428	234940	0.373789	83098	0.132209	
1	13176	47209	37628	0.059866	206519	0.328571	148170	0.235738	

	confidenceAtoB	confidenceBtoA	lift
6	0.124210	0.351176	0.939503
1	0.182201	0.253952	0.772897

For min_sup = 0.06

	item_A	item_B	freqAB	supportAB	freqA	supportA	freqB	supportB	\
2	24852	21903	35590	0.116937	118625	0.389764	132748	0.436167	
1	24852	21137	38564	0.126709	118625	0.389764	150887	0.495766	

Project - 2

	confidenceAtoB	confidenceBtoA	lift
2	0.300021	0.268102	0.687858
1	0.325092	0.255582	0.655736

For min_sup = 0.08

Item pairs with support >= 0.08:   4

	item_A	item_B	freqAB	supportAB	freqA	supportA	freqB	supportB
0	24852	21137	38564	0.325308	57125	0.481880	117577	0.991826
2	13176	21137	40029	0.337666	62597	0.528040	117577	0.991826
3	21137	13176	21599	0.182199	117577	0.991826	62597	0.528040
1	21137	24852	17592	0.148398	117577	0.991826	57125	0.481880

	confidenceAtoB	confidenceBtoA	lift
0	0.675081	0.327989	0.680645
2	0.639472	0.340449	0.644742
3	0.183701	0.345048	0.347892
1	0.149621	0.307956	0.310494

For min_sup = 0.1

Item pairs with support >= 0.1:   2

	item_A	item_B	freqAB	supportAB	freqA	supportA	freqB	supportB
0	24852	13176	654	0.556122	1176	1.0	1176	1.0
1	13176	24852	522	0.443878	1176	1.0	1176	1.0

	confidenceAtoB	confidenceBtoA	lift
0	0.556122	0.556122	0.556122
1	0.443878	0.443878	0.443878

The lift value for most of the associations are less than zero. Hence, I would suggest that company should be able to sell more products with association greater than one. Hence, I would select min_sup of 0.01. One more advantage here is since there are very few associations with lift >1, inventory stock of those few items can be increased and without the risk of losing for deadstock the company can make more sales with these increased stock items with lift>1.

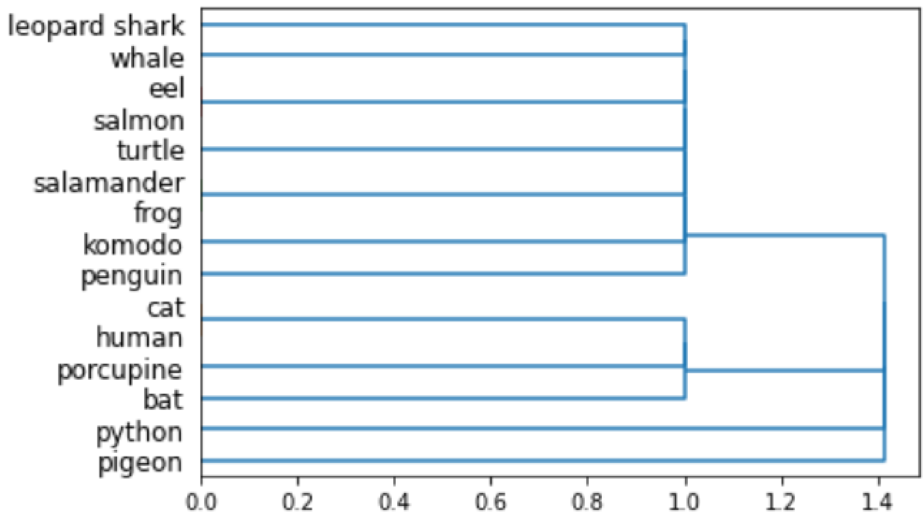
Experiment-3:

1.

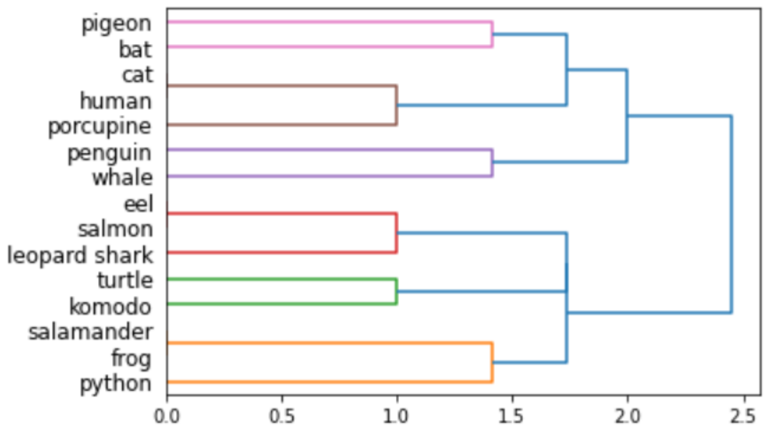
	user	Jaws	Star Wars	Exorcist	Omen	Cluster ID
0	Paul	4	5	2	4	1
1	Adel	1	2	3	4	0
2	Kevin	2	3	5	5	0
3	Jessi	1	1	3	2	0

Project - 2

2. K value at the elbow in the plot is chosen in k-means and as per the elbow method this k value is optimal. In the given toy dataset at k=2, the elbow can be observed. Hence k=2 is the optimal value of k and has been chosen in the experiment.
3.
    1.

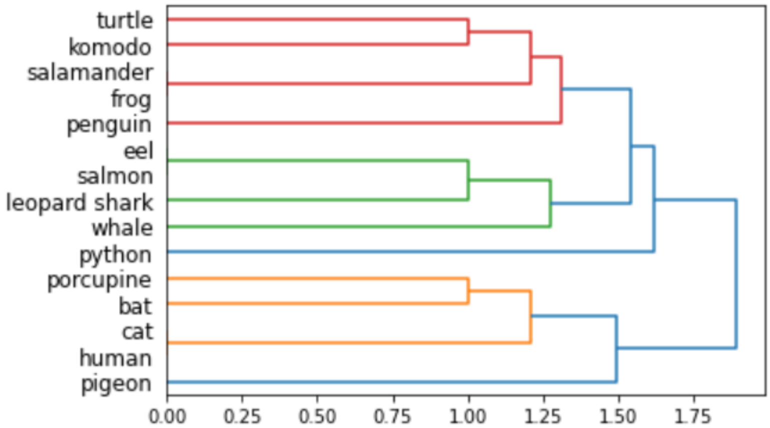


    2.



    3.

Project - 2



Group average makes more sense because the clustering is more true in this case.

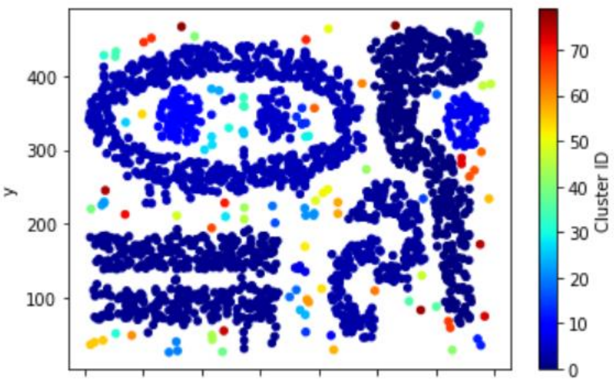
4.

When min_points = 1

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=15.5, min_samples=1).fit(data)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = pd.DataFrame(db.labels_,columns=['Cluster ID'])
result = pd.concat((data,labels), axis=1)
result.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')
```

```
<AxesSubplot:xlabel='x', ylabel='y'>
```



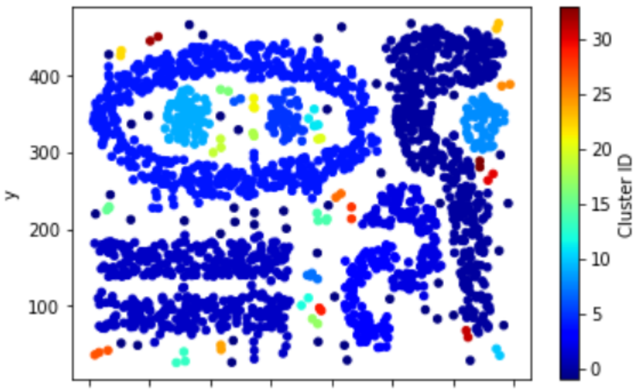
When min_points = 2

Project - 2

```python
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=15.5, min_samples=2).fit(data)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = pd.DataFrame(db.labels_,columns=['Cluster ID'])
result = pd.concat((data,labels), axis=1)
result.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')
```

```
<AxesSubplot:xlabel='x', ylabel='y'>
```

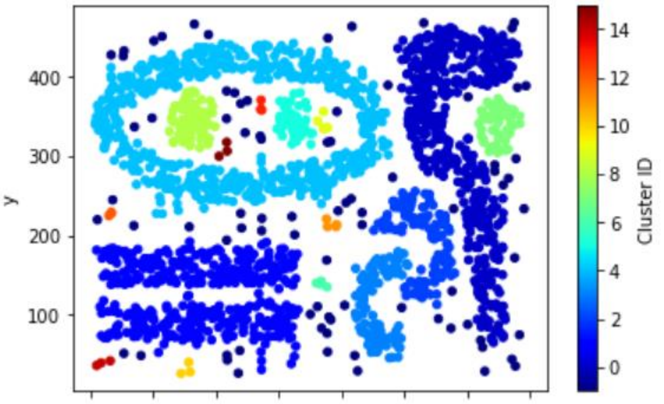


When min_points = 3

```python
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=15.5, min_samples=3).fit(data)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = pd.DataFrame(db.labels_,columns=['Cluster ID'])
result = pd.concat((data,labels), axis=1)
result.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')
```

```
<AxesSubplot:xlabel='x', ylabel='y'>
```



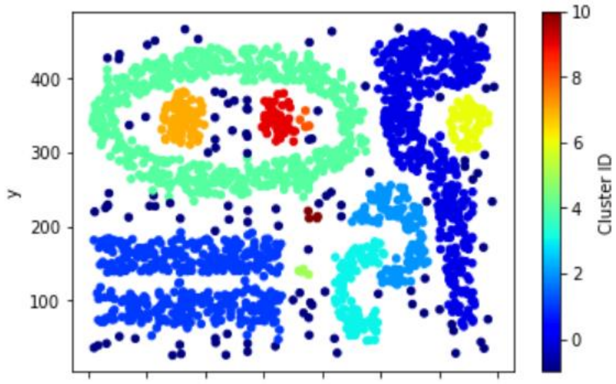
When min_points = 4

Project - 2

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=15.5, min_samples=4).fit(data)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = pd.DataFrame(db.labels_,columns=['Cluster ID'])
result = pd.concat((data,labels), axis=1)
result.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')
```

```
<AxesSubplot:xlabel='x', ylabel='y'>
```

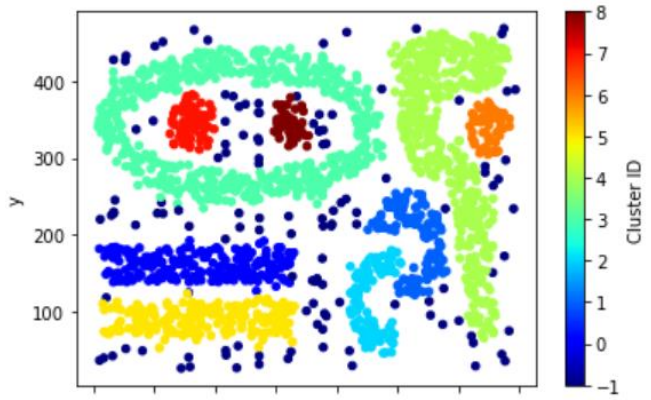


When min_points = 5

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps=15.5, min_samples=5).fit(data)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = pd.DataFrame(db.labels_,columns=['Cluster ID'])
result = pd.concat((data,labels), axis=1)
result.plot.scatter(x='x',y='y',c='Cluster ID', colormap='jet')
```

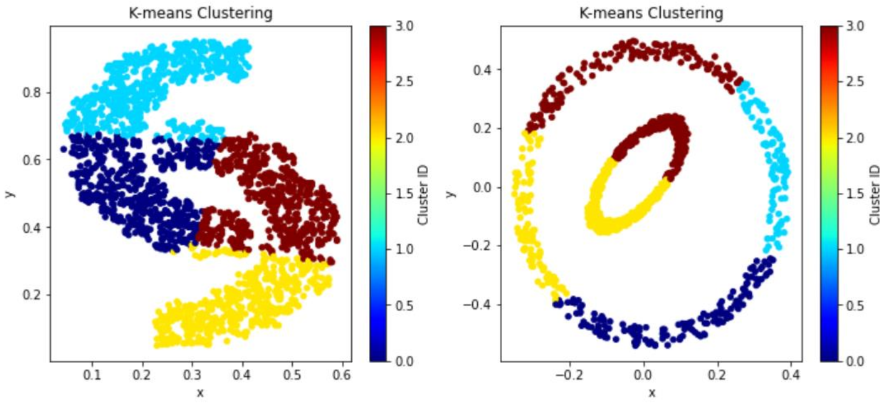
```
<AxesSubplot:xlabel='x', ylabel='y'>
```



Project - 2

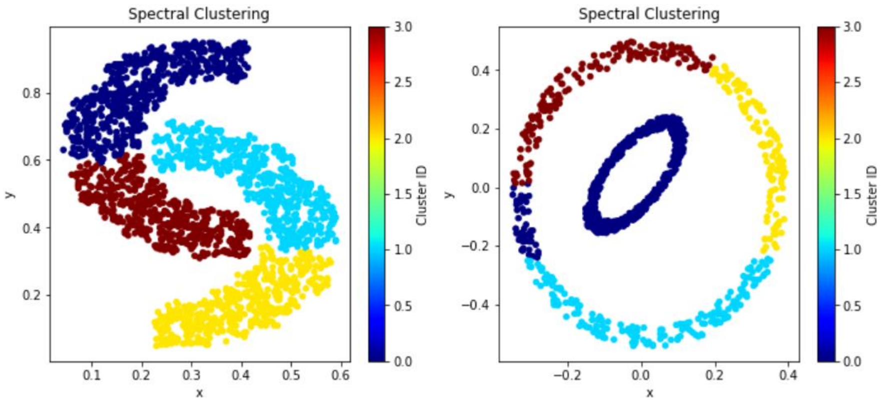
5. When k=4,
   kmeans:

```
Text(0.5, 1.0, 'K-means Clustering')
```



   Spectral kmeans

```
Text(0.5, 1.0, 'Spectral Clustering')
```



   Spectral kmeans clusters the arbitrary data points more meaningfully than the regular kmeans method.

Project - 2

Experiment-4:

1.

	abstract_word_count	body_word_count
count	24584.000000	24584.000000
mean	216.446673	4435.475106
std	137.065117	3657.421423
min	1.000000	23.000000
25%	147.000000	2711.000000
50%	200.000000	3809.500000
75%	255.000000	5431.000000
max	3694.000000	232431.000000

2. The author might have submitted their articles to multiple journals thereby creating duplicate records. In the data pre-processing after creating the unique values the duplicates are removed and this data set after removing duplicates is used for further analysis.
3.

Now that we have the text cleaned up, we can create our features vector which can be fed into a clustering or dimensionality reduction algorithm. For our first try, we will focus on the text on the body of the articles. Let's grab that:

```
text = df_covid.drop(["paper_id", "abstract", "abstract_word_count", "body_word_count", "authors", "title", "journal", "abstract_
```

```
text.head(5)
```

	body_text
1625	introduction human beings are constantly expos...
1626	pathogens and vectors can now be transported r...
1627	a1111111111 a1111111111 a1111111111 a111111111...
1628	in addition to preventative care and nutrition...
1629	ubiquitination is a widely used posttranslatio...

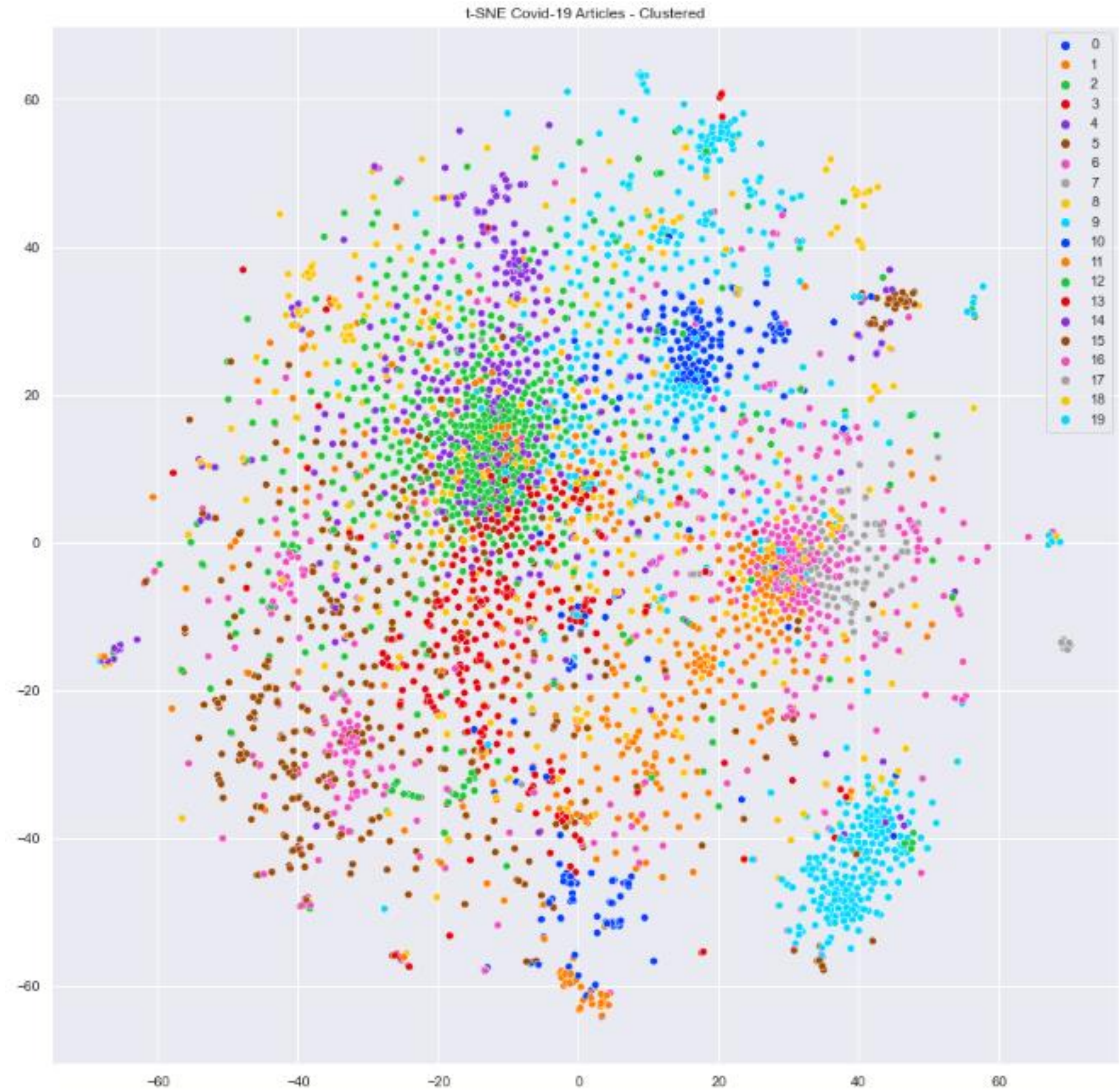
Let's transform 1D DataFrame into 1D list where each index is an article (instance), so that we can work with words from each instance:

```
text_arr = text.stack().tolist()
print(text_arr)
len(text_arr)
```

4. N-grams are continuous sequences of words or symbols or tokens in a document. 2-gram is also called as bigram and 2-gram list for the given list is [“the 2019”, “2019 novel”, “novel coronavirus”, “coronavirus sarscov2”, “sarscov2 identified”, “identified as”, “as the”, “the cause”, “cause of”].

Project - 2

5. Hashing vectorizer is a vectorizer which uses the hashing trick to find the token string name to feature integer index mapping. Conversion of text documents into matrix is done by this vectorizer where it turns the collection of documents into a sparse matrix which are holding the token occurrence counts. The feature size to 2**12(4096) is used in the analysis in the experiment.
6. When k = 20 and feature size = 2**14 = 16384



Project - 2

7. When k = 20 and feature size = 2**14 = 16384

8.

C-8: One health, global health and policies, Food policies and roles in changing environment, Global health training.
C-4: Virus mediated autoimmunity, vaccination supresses the spread, Investigation of antibodies, phase for viral production
C-6: Virome diversity, infections, predicting zoonoses, cross-species evolution
C-16: viral journeys, mechanisms expoited by pathogen, structural basis, illuminating pathogen
C-10: detecting respiratory viruses, sites of early viral replication, mutation, immune mediated disorders.

Project - 2

Social and economic cluster is C-18.