Amazon Fine Food Review - K Means, Agglomerative, DBSCAN

1. Objective

To Cluster the same type of Data points

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
import warnings
import sqlite3
warnings.filterwarnings("ignore")
```

2. Data Cleaning

In [2]:
```
#connecting database

con=sqlite3.connect("database.sqlite")

# Read data from database

raw_data=pd.read_sql_query("""SELECT * FROM Reviews WHERE Score !=3""",c

# Removal of Duplicates

pre_data=raw_data.drop_duplicates(['UserId','ProfileName','Time','Text']

# Removal of Unconditioning data (denominator>numerator)

pre_data=pre_data[pre_data.HelpfulnessNumerator<=pre_data.HelpfulnessDen


# Finding NaN values in dataframe

# Reference
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.isnu

# Findind NaN values

if pre_data.isnull().values.any() == False:
    print("There is No NaN values in the DataFrame")
else:
    print(" There is NaN values present in the DataFrame")
```

```
There is No NaN values in the DataFrame
```

In [3]:
```
# sort data based on Time

filter_data=pre_data.sort_values(by=["Time"],axis=0)

# Class Label changing
# positive class label = 1
# negative class label = 0
a=[]
for i in filter_data["Score"]:
    if i > 3:
        a.append(1)
    else:
        a.append(0)
filter_data["Score"]=a
```

In [4]:
```
filter_data.shape
```

Out[4]:
```
(364171, 10)
```

In [5]:
```
filter_data["Score"].value_counts()
```

Out[5]:
```
1    307061
0     57110
Name: Score, dtype: int64
```

3. Text Preprocessing

- We took the Text column for the further review idendification task, because text is the most important feature compared to other features.

In [7]:
```
# References
# https://medium.com/@jorlugaqui/how-to-strip-html-tags-from-a-string-in
# https://stackoverflow.com/a/40823105/4084039
# https://stackoverflow.com/questions/19790188/expanding-english-languag
# https://stackoverflow.com/questions/18082130/python-regex-to-remove-al
# https://stackoverflow.com/questions/5843518/remove-all-special-charact
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://gist.github.com/sebleier/554280
# stemming tutorial: https://www.geeksforgeeks.org/python-stemming-words
# Lemmatisation tutorial: https://www.geeksforgeeks.org/python-lemmatiza
# NLTK Stemming package list: https://www.nltk.org/api/nltk.stem.html

from nltk.stem.snowball import EnglishStemmer
import re
from tqdm import tqdm
stemmer=EnglishStemmer()
```

In [8]:
```
raw_text_data=filter_data["Text"].values
```

In [9]:
```
# Stopwords

stopwords= set(['since','br', 'the', 'i', 'me', 'my', 'myself', 'we', 'ou
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves'
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'hav
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'in
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'o
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so'
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "s
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'di
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
            'won', "won't", 'wouldn', "wouldn't"])

# expanding contractions

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [10]:
```
preprocessed_text_data=[]
for i in tqdm(raw_text_data):
# removing of HTML tags
    a=re.sub("<.*?>"," ",i)
# removing url
    b=re.sub(r"http\S+"," ",a)
# expanding contractions
    c=decontracted(b)
# removing alpha_numeric
    d=re.sub("\S*\d\S*", " ",c)
# removing Special characters
    e=re.sub('[^A-Za-z0-9]+', ' ',d)
# removing stopwords
    k=[]
    for w in e.split():
        if w.lower() not in stopwords:
            s=(stemmer.stem(w.lower())).encode('utf8')
            k.append(s)
    preprocessed_text_data.append(b' '.join(k).decode())
```

100%|██████████| 364171/364171 [06:59<00:00, 868.30it/s]

In [11]:
```
filter_data["Text"]=preprocessed_text_data
```

In [12]:
```
filter_data.shape
```

Out[12]: (364171, 10)

4. K-Means Clustering

4.1 Data

In [14]:
```
# we took the sample data size as 50k

final_data=filter_data[:50000]
final_data.shape
```

Out[14]: (50000, 10)

In [15]:
```
X=final_data.Text
```

4.2 Featurization

4.2.1 Bag of Words (BoW)

In [16]:
```
# Reference
# https://scikit-learn.org/stable/modules/generated/sklearn.feature_extr

from sklearn.feature_extraction.text import CountVectorizer
```

In [17]:
```
bow_model=CountVectorizer(ngram_range=(1,2),min_df=5,max_features=500)

# BOW on data

bow_train_vec1=bow_model.fit_transform(X)
```

In [20]:
```
# the number of words in BOW or Vector size

print("The size of BOW vectorizer")
print(bow_train_vec1.get_shape()[1])
```

```
The size of BOW vectorizer
500
```

4.2.2 TFIDF

In [21]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.feature_extr

from sklearn.feature_extraction.text import TfidfVectorizer
```

In [22]:
```
tfidf_model=TfidfVectorizer(ngram_range=(1,2),min_df=5,max_features=500)

# TFIDF on data

tfidf_train_vec1=tfidf_model.fit_transform(X)
```

In [23]:
```
# the number of words in TFDIF or Vector size

print("The size of TFIDF vectorizer")
print(tfidf_train_vec1.get_shape()[1])
```

```
The size of TFIDF vectorizer
500
```

4.2.3 W2V

In [24]:
```
# References
# https://radimrehurek.com/gensim/models/word2vec.html
# https://machinelearningmastery.com/develop-word-embeddings-python-gens
# http://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/#.W17S

from gensim.models import Word2Vec
```

In [25]:
```
list_sentences_train=[]
for i in tqdm(list(X)):
    list_sentences_train.append(i.split())
```

```
100%|██████████| 50000/50000 [00:00<00:00, 90830.71it/s]
```

In [26]:
```
word2vec_model=Word2Vec(list_sentences_train,min_count=5,size=50,workers=
```

In [27]:
```
word2vec_words_train=list(word2vec_model.wv.vocab)
print(" Number of words")
print("_____________________")
print(" ")
print(len(word2vec_words_train))
print("="*125)
print(" sample words")
print("_____________________")
print(" ")
print(word2vec_words_train[100:150])
```

```
 Number of words
_____________________

9639
================================================================================
================================================================
 sample words
_____________________

['agre', 'good', 'time', 'watch', 'collect', 'fill', 'comedi', 'actio
n', 'whatev', 'els', 'want', 'call', 'enjoy', 'entertain', 'hesit', 'pi
ck', 'edit', 'guess', 'market', 'plan', 'famili', 'elimin', 'strong',
'element', 'usual', 'version', 'warn', 'uncut', 'avoid', 'apart', 'infe
st', 'fruit', 'fli', 'hour', 'trap', 'quot', 'attract', 'mani', 'withi
n', 'practic', 'gone', 'may', 'long', 'term', 'solut', 'crazi', 'consi
d', 'buy', 'caution', 'surfac']
```

4.2.4 Avg W2V

In [28]:
```
# Reference
# formula of Avg word2vec = sum of all (wi)[i=0 to n]/n

# avg word2vec on training data

avg_word2vec_train=[]
for i in tqdm(list_sentences_train):
    vector=np.zeros(50)
    no_of_words=0
    for k in i:
        try:
            w2v_data=word2vec_model.wv[k]
            vector=vector+w2v_data
            no_of_words=no_of_words+1
        except:
            pass
    if no_of_words != 0:
        vector=vector/no_of_words
    avg_word2vec_train.append(vector)
avg_w2v_train=np.asmatrix(avg_word2vec_train)
print("shape of Avg Word2vec train")
print(avg_w2v_train.shape)
```

```
100%|██████████| 50000/50000 [00:11<00:00, 4417.53it/s]

shape of Avg Word2vec train
(50000, 50)
```

4.2.5 TFIDF W2V

In [29]:
```
# References
# https://stackoverflow.com/questions/21553327
# https://github.com/devBOX03


# tfidf word2vec on training data

model=TfidfVectorizer()
tfidf_w2v_model=model.fit_transform(X)
tfidf_w2v=model.get_feature_names()
tfidf_word2vec_train=[]
row=0
for i in tqdm(list_sentences_train):
    vec=np.zeros(50)
    weight_sum=0
    for w in i:
        try:
            w2v_freq=word2vec_model.wv[w]
            tfidf_freq=tfidf_w2v_model[row,tfidf_w2v.index(w)]
            vec=vec+(w2v_freq*tfidf_freq)
            weight_sum=weight_sum+tfidf_freq
        except:
            pass
    vec=vec/weight_sum
    tfidf_word2vec_train.append(vec)
    row=row+1
tfidf_w2v_train=np.asmatrix(tfidf_word2vec_train)
print("Shape of TFIDF word2vec train")
print(tfidf_w2v_train.shape)
```

```
100%|██████████| 50000/50000 [16:47<00:00, 49.65it/s]

Shape of TFIDF word2vec train
(50000, 50)
```

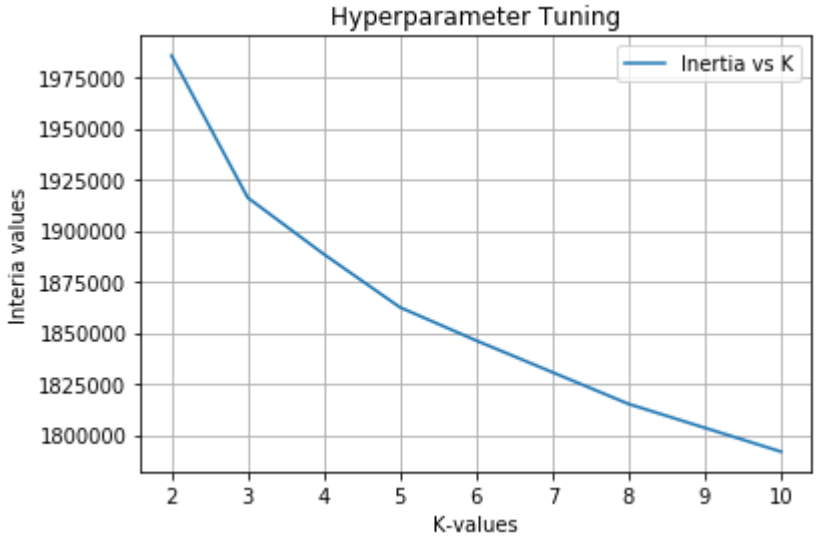
4.3 K-Means using BoW

In [30]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMea
# https://imaddabbura.github.io/post/kmeans_clustering/
# https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html
# https://github.com/PushpendraSinghChauhan/Amazon-Fine-Food-Reviews/blo

from sklearn.cluster import KMeans
```

In [31]:
```
# Hyperparameter tuning

k = [2,3,4,5,6,8,10]

inertias=[]

for i in tqdm(k):

    model = KMeans(n_clusters=i,n_jobs=-1)
    model.fit(bow_train_vec1)
    sum_sq_values = model.inertia_
    inertias.append(sum_sq_values)
```

```
100%|██████████| 7/7 [1:24:38<00:00, 811.21s/it]
```

In [63]:
```
# plotting the k vs inertia

plt.close()
plt.plot(k,inertias,label="Inertia vs K")
plt.grid()
plt.title("Hyperparameter Tuning")
plt.xlabel("K-values")
plt.ylabel("Interia values")
plt.legend()
plt.show()
```



Observation:

- By using the elbow method the best k (number of clusters) is 6

In [35]:
```
# Applying Best Hyperparameter

model= KMeans(n_clusters=6,n_jobs=-1)
model.fit(bow_train_vec1)
labels=model.labels_
```

Number Datapoints in Each Cluster

In [42]:
```
# Data points seperation as per the clusters

number_points = labels.shape[0]
print("Number of Datapoints")
print(number_points)
```

```
Number of Datapoints
50000
```

In [58]:
```
# Datapoints divided by clusters as per the label name

cluster_1=[]
cluster_2=[]
cluster_3=[]
cluster_4=[]
cluster_5=[]
cluster_6=[]

for i in range(0,number_points):

    if labels[i] == 0:
        cluster_1.append(i)
    if labels[i] == 1:
        cluster_2.append(i)
    if labels[i] == 2:
        cluster_3.append(i)
    if labels[i] == 3:
        cluster_4.append(i)
    if labels[i] == 4:
        cluster_5.append(i)
    if labels[i] == 5:
        cluster_6.append(i)
```

In [65]:
```
# References
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable
```

In [66]:
```
# The number of datapoints in each cluster

a=PrettyTable()

a.field_names = ["Cluster", "Number of Data Points"]

print(" The number of datapoints in each cluster")
print("="*120)

a.add_row([1,str(len(cluster_1))])
a.add_row([2,str(len(cluster_2))])
a.add_row([3,str(len(cluster_3))])
a.add_row([4,str(len(cluster_4))])
a.add_row([5,str(len(cluster_5))])
a.add_row([6,str(len(cluster_6))])
print(a)
```

```
 The number of datapoints in each cluster
========================================================================
================================================
+---------+-----------------------+
| Cluster | Number of Data Points |
+---------+-----------------------+
|    1    |         11297         |
|    2    |          1216         |
|    3    |          2153         |
|    4    |          256          |
|    5    |         33657         |
|    6    |          1421         |
+---------+-----------------------+
```

Wordcloud for each cluster:

Cluster 1

- Getting the sample reviews in Cluster 1

In [161]:
```
# References
# https://www.geeksforgeeks.org/generating-word-cloud-python/

from wordcloud import WordCloud
```

In [168]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(11297,size=3)
```

In [169]:
```
rand_num = list(rand_num)
```

In [170]:
```
rand_num
```

Out[170]:
```
[1061, 225, 6566]
```

In [171]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_1[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [172]:
```
string_1 = " ".join(word_cloud)
```

In [173]:
```
string_1
```

Out[173]:
```
'love granola cereal general say best granola ever peopl bear nake phil
osophi love back everi bag believ food hand made wholesom natur ingredi
actual pronounc although eat process food appreci appeal tradit prepar
food granola top list nutrit tast like granola high calori fat take not
e fat come larg nut contain also contain expel press canola oil no tran
fat satur fat per serv cereal softer granola though not soft enough con
sid chewi larg chunk nut also contain cranberri flax seed optim nutrit
digest health six pack sold amazon make price per packag reason ingredi
quit expens valu proposit think not great product great buy delici nutr
iti great combin recommend high got best hot sauc ever flavor concern b
uy one blair hotter sauc add bit increas sauc heat like heat level taba
sco time flavor never met anyon not love sauc tri hunt decent treat gsd
tplo surgeri recov nice search healthi treat came across happi hip ingr
edi natur absolut love idea also benefiti joint add special bonus produ
ct no sugar garbag key protein diet digest problem not problem chang no
t go wrong'
```

In [174]:
```
wordcloud_1 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 2

- Getting the sample reviews in Cluster 2

In [177]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(1216,size=3)
rand_num = list(rand_num)
```

In [178]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_2[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [179]:
```
string_2 = " ".join(word_cloud)
```

In [180]:
```
string_2
```

Out[180]:
```
'coffe great husband total coffe snob happi roast fresh coffe quick del
iveri torani wonder italian sound although made south san francisco pre
tti ubiquit bay area found mani groceri store especi coffe bar cafe lea
st good sweeten best actual impart promis flavor peppermint deliv would
rank hazelnut far musti old tast almond torani peppermint add peppermin
t flavor get nice non alcohol addit hot chocol coffe dessert steam milk
ingredi nutrit inform one ounc syrup serv ml bottl contain calori no fa
t gram carbohydr gram sugar no wonder sweet presenc actua peppermint sl
ight suspect list ingredi pure cane sugar water natur flavor natur flav
or sodium benzoat potassium sorbat preserv citric acid one hint close t
ight ant love stuff keurig coffe maker offic coupl year enjoy immens pr
obabl tri everi offer avail offic includ regular caffein coffe far flav
or emeril jazz decaf tend favor decaf one stand among one favorit offic
timothi rainforest espresso good right howev ran emeril forc brew cup r
ainforest left disappoint tast depart review may call tast burnt palat
robust flavor thorough enjoy coffe'
```

In [181]:
```
wordcloud_2 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 3

- Getting the sample reviews in Cluster 3

In [182]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(2153,size=3)
rand_num = list(rand_num)
```

In [184]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_3[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [185]:
```
string_3 = " ".join(word_cloud)
```

In [186]:
```
string_3
```

Out[186]:
```
'great peopl asthma like tast best result abl breath freeli releiv cong
est feel better inhal although let tea steep recommend minut yogi tea b
reath deep tea bag reorder blow first oz tin week use flavor black tea
mix anoth herbal tea thing leeri becom addict adagio great tea servic r
eceiv tea birthday gift general not like tea odd flavor not like stuff
like cinnamon lemon rose hip orang rind tea love smell flavor light ple
asant tea bag nice qualiti recommend tea'
```

In [187]:
```
wordcloud_3 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 4

- Getting the sample reviews in Cluster 4

In [196]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(256,size=3)
rand_num = list(rand_num)
```

In [197]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_4[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [198]:
```
string_4 = " ".join(word_cloud)
```

In [199]:
```
string_4
```

Out[199]:
```
'tri tea first time tri origin chai indian spice tea yesterday fell lov
e chai tea cannot tast differ tea origin indian spice tea cup back back
sure tast exact know origin vanilla no honey not add sugar anyth els te
a cannot tast honey tea excel sure cannot go wrong celesti season chai
tea look differ flavor origin tri chocol enhanc tea good sweet slight m
ilder one term indian spice black pepper flavor still wonder tea health
benefit white tea black tea folk unit kingdom drink pg tip tea brand pe
rhap tri choic way folk ireland drink tea per person countri pg tip tea
pg tip special blend tea home standard tea specialti retail usa offer p
g tip special blend tea hope amazon soon pre digest era histori tea int
roduc name pre gest tee suggest tea could consum food eaten grocer gave
abbrevi pg compani ad tip compani use best part tea plant two top leav
bud tea plant make tea mani black tea sold usa grade basic medium grade
orang peko general tea leav top two bud one reason pg tip get higher re
view tea pg tip also use pyramid tea bag allow ampl room tea expand inf
us addit opinion pg tip blend two blend rather blend wide assort thus b
ecom realli good amazon pg tip price nickel tea bag good valu import pr
oduct order count box set two even less expens higher volum pg tip make
wonder ice tea good news american drink tea ice tea hot tea worldwid ho
t tea consum ice tea worldwid tea consum beverag next water usa sixth c
onsum beverag next water good question ponder drink adequ amount tea da
ili consid good one health much enjoy drink realli good cuppa cup tea e
speci nomin cost nickel cup compar one gourmet coffe place charg upward
four five dollar larg cup coffe guest home alway offer pg tip cuppa alw
ay say never tast tea good thus rate star qualiti nomin price morn thun
der excel name drink feel might put peopl tea good without whacki crazi
tast expect real potent punch upon read yerba mate drank one favorit ch
ai mix sure would palat pleas right tea simpl ingredi list short roast
mate black tea no filler no funni stuff no funki herb spice upon search
internet found yerba mate herb tradit prepar energ tea consum nativ sou
th american countri contain high concentr antioxid flavonoid possess po
tent free radic quench activ also includ sever b vitamin vitamin c vita
min e beta caroten calcium mix black tea mate tast like dark chocol hin
t mix tea subtl yet eye open experi upon first sip tri take good care h
ealth drink lot tea water no idea high antioxid tea per serv compar bro
ccoli tomato juic orang juic tea much easier prepar eat bowl full brocc
oli drink tomato juic quit refresh also tast nice milk sugar not add le
mon sugar let soak minut make slight smoki chocolati exot brew yet stil
l tea drank breakfast tea time love cover art simpl good celesti season
reach transform mani coffe addict tea lover kasia'
```

In [200]:
```
wordcloud_4 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 5

- Getting the sample reviews in Cluster 5

In [201]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(33657,size=3)
rand_num = list(rand_num)
```

In [202]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_5[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [203]:
```
string_5 = " ".join(word_cloud)
```

In [204]:
```
string_5
```

Out[204]:
```
'buy year son daughter love quick snack high salt least nutrit no longe
r interest mom homecook main eat junk noth tri say chang least tri find
healthiest junk food dog listen snack good given young children sort tr
ick beg snack come folk kid perfect dog qualiti bean fantas smooth not
bitter medium bold tast depend quaniti bean use good would recomend bra
nd flavor bean buy'
```

In [205]:
```
wordcloud_5 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 6

- Getting the sample reviews in Cluster 6

In [206]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(1421,size=3)
rand_num = list(rand_num)
```

In [207]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_6[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [208]:
```
string_6 = " ".join(word_cloud)
```

In [209]:
```
string_6
```

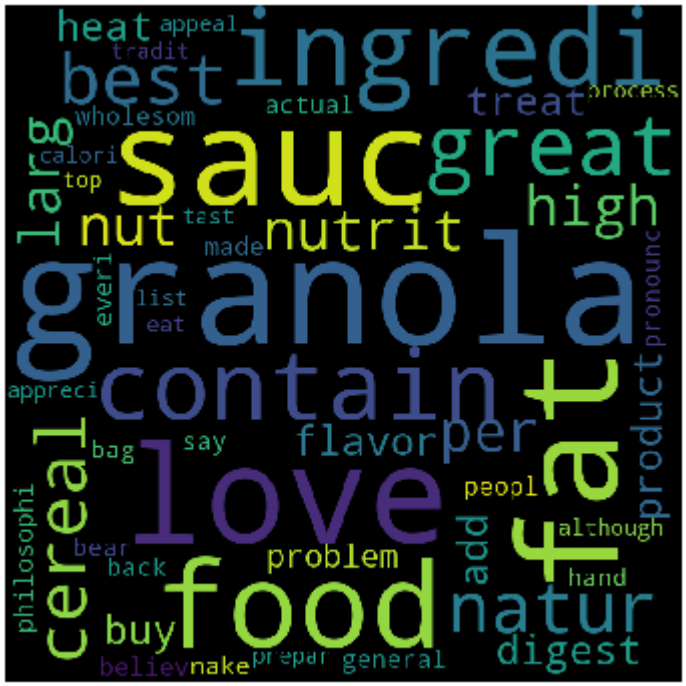
Out[209]:
```
'kraft light balsam vinaigrett extra virgin oliv oil reduc fat dress wh
ew long name help peopl like diet tast inde bold think tast come sodium
salad dress hand inde made extra virgin oliv oil wonder gram fat tables
poon serv gram gram carb per serv no artifici preserv bonus probabl eve
n better health amazon nutrit fact would indic not think amazon got qui
t right quot calor content nutrit fact tabl bottl product yes match pic
tur exact product nutrit fact label bottl indic calori total per serv s
erv consist tablespoon addit nutrit fact label bottl indic amount calor
i come fat total calori per serv not calori amazon note would like thin
k kraft nutrit label correct perhap amazon accident made misprint two u
nfortun amount sodium indic webpag accur overal great dress tast great
add punch salad simpli not find salad dress fewer calori although might
watch sodium issu salt free diet check nutritionist doctor first make s
ure dress someth enjoy salad use salad mixtur romain iceberg lettuc car
rot thrown sure would go spinach salad well enjoy fan natur anyth espec
i tasti healthi altern someth enjoy whole life know bad regular soda on
e bad thing load artifici color sweeten corn syrup yuck hard seem worth
risk also get empti calori averag switch come along true claim noth art
ifici juic sparkl water natur ingredi great chose particular flavor ora
ng tangerin want strong citrus tast love drink howev fail meet promis y
es notast orangi citrus much tast appl grape juic use well moreov not b
old pop carbon beverag miner water bite wors part good drink clock calo
ri regular coke sugar compar coke side less half sodium mani regular co
la deliv vit c vit want kind health benefit would rather take direct so
urc orang overal natur health pros switch orang tangerin simpli not jus
tifi high calori carb count switch beverag co may done better sell spar
kl water twist orang peel want provid truli healthi altern tradit soda
thought cooki bit pricey subscrib save price compar organ sweet market
luckili tast not like organ sweet market good thing organ cooki dri cru
mbl singl bite late juli cooki creami fill not greasi like oreo outer c
ooki good crunch textur hold well not crumbl no thing healthi cooki get
occasion sweet tooth cooki great way satiat tooth without complet blow
diet organ boot no tast smell look green tea cooki would never know gre
en tea not label also good thing know get great benefit ecgc without st
rong tea tast nice ad bonus treat cooki offici sweet treat splurg moder
not blow food routin'
```

In [210]:
```
wordcloud_6 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Plotting The Wordcloud

- **Cluster 1**

In [211]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_1)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- This cluster says about how the product tastes and quality

- **Cluster 2**

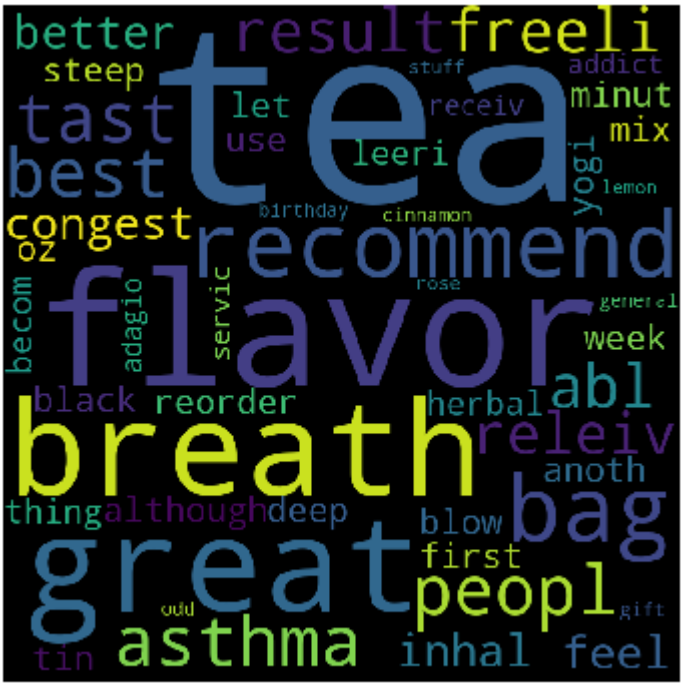
In [212]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_2)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Observation:

- This cluster says about coffee and dairy products.

- **Cluster 3**

In [213]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_3)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

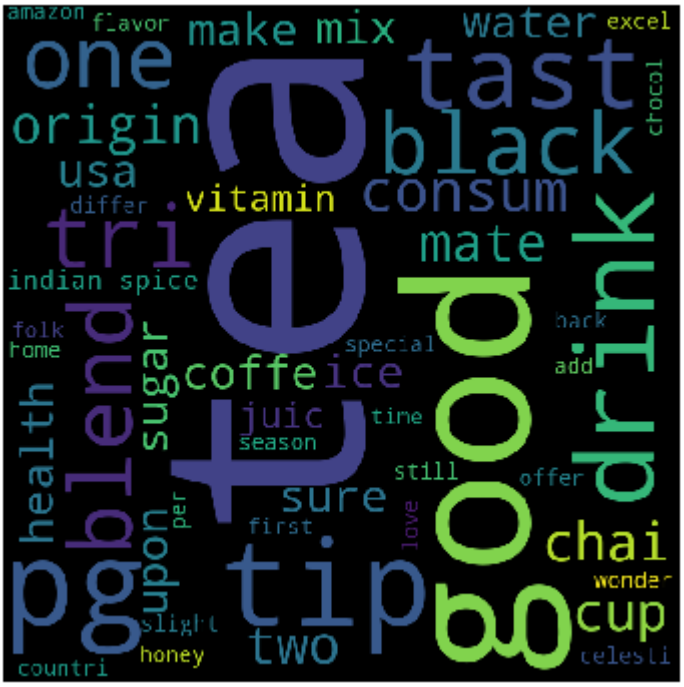


Observation:

- This cluster says about tea products.

- **Cluster 4**

In [214]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_4)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

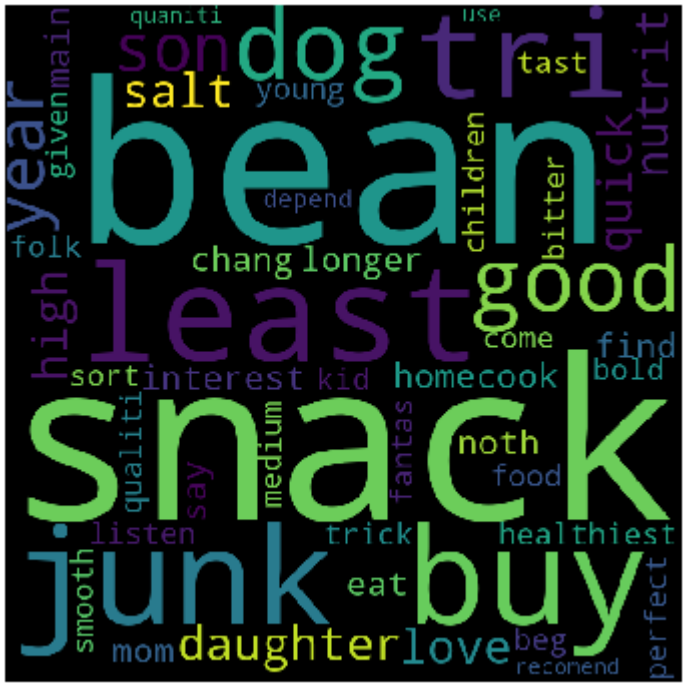


Observation:

- This cluster says about tea products.

- **Cluster 5**

In [215]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_5)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

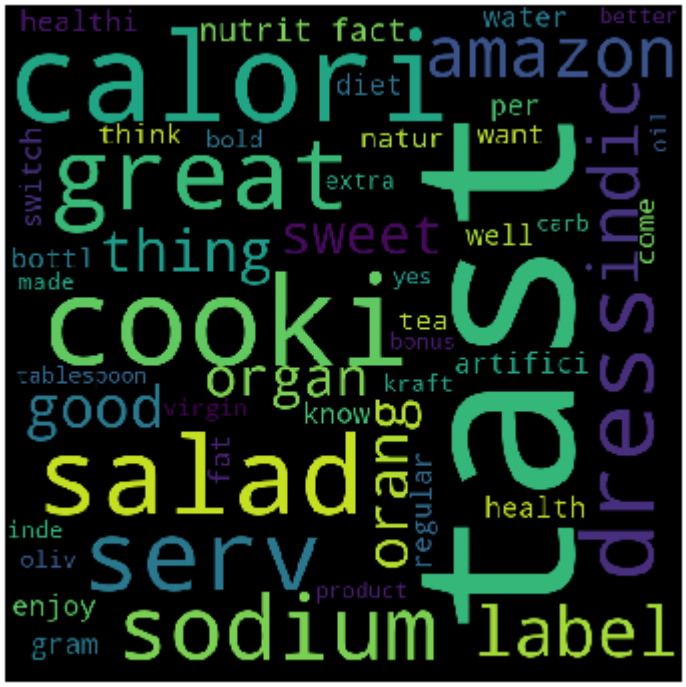


Observation:

- This cluster says about snack products.

- **Cluster 6**

In [216]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_6)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- This cluster says about cookies and product quality as well as tast.

Performance Metric of K means using BoW

In [233]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_me

from sklearn.metrics import silhouette_score
```

In [234]:
```
score = silhouette_score(bow_train_vec1,labels)
```

In [235]:
```
score
```

Out[235]:
```
0.1602613063543015
```

Observation:

- As per the silhouette score document if the score is nearest to the Zero. The Clusters are Overlapped. So here The Silhouette Score is 0.16. So here the chance of clusters

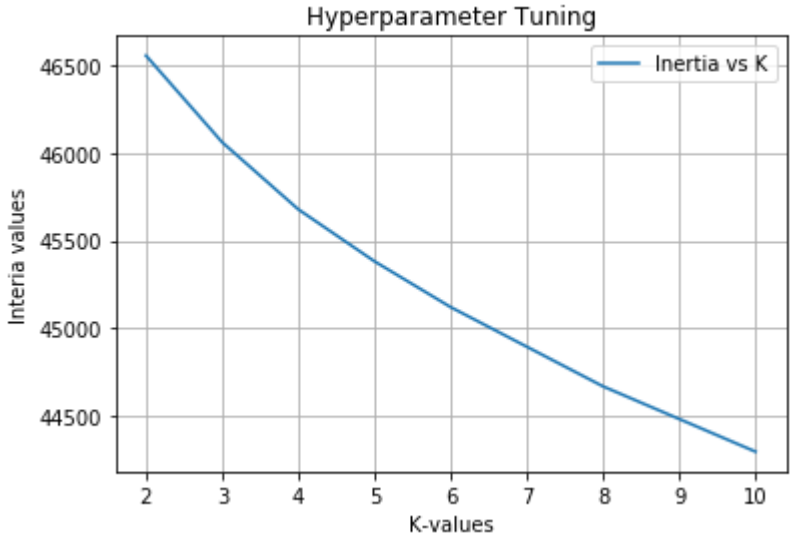
overlapping is high.

4.4 K-Means using TFIDF

In [236]:
```
# Hyperparameter tuning

k = [2,3,4,5,6,8,10]

inertias=[]

for i in tqdm(k):

    model = KMeans(n_clusters=i,n_jobs=-1)
    model.fit(tfidf_train_vec1)
    sum_sq_values = model.inertia_
    inertias.append(sum_sq_values)
```

```
  0%|          | 0/7 [00:00<?, ?it/s]
 14%|▉         | 1/7 [06:00<36:03, 360.66s/it]
 29%|██▊       | 2/7 [16:20<36:32, 438.40s/it]
 43%|████▎     | 3/7 [27:56<34:22, 515.72s/it]
 57%|█████▋    | 4/7 [42:16<30:56, 618.99s/it]
 71%|███████▏  | 5/7 [1:00:20<25:17, 758.55s/it]
 86%|████████▌ | 6/7 [1:22:17<15:26, 926.05s/it]
100%|██████████| 7/7 [1:42:00<00:00, 1003.11s/it]
```

In [237]:
```
# plotting the k vs inertia

plt.close()
plt.plot(k,inertias,label="Inertia vs K")
plt.grid()
plt.title("Hyperparameter Tuning")
plt.xlabel("K-values")
plt.ylabel("Interia values")
plt.legend()
plt.show()
```



Observation:

- By using the elbow method the best k (number of clusters) is 6

In [238]:
```
# Applying Best Hyperparameter

model= KMeans(n_clusters=6,n_jobs=-1)
model.fit(tfidf_train_vec1)
labels=model.labels_
```

Number Datapoints in Each Cluster

In [239]:
```
# Data points seperation as per the clusters

number_points = labels.shape[0]
print("Number of Datapoints")
print(number_points)
```

```
Number of Datapoints
50000
```

In [240]:
```
# Datapoints divided by clusters as per the label name

cluster_1=[]
cluster_2=[]
cluster_3=[]
cluster_4=[]
cluster_5=[]
cluster_6=[]

for i in range(0,number_points):

    if labels[i] == 0:
        cluster_1.append(i)
    if labels[i] == 1:
        cluster_2.append(i)
    if labels[i] == 2:
        cluster_3.append(i)
    if labels[i] == 3:
        cluster_4.append(i)
    if labels[i] == 4:
        cluster_5.append(i)
    if labels[i] == 5:
        cluster_6.append(i)
```

In [242]:
```
# The number of datapoints in each cluster

b=PrettyTable()

b.field_names = ["Cluster", "Number of Data Points"]

print(" The number of datapoints in each cluster")
print("="*120)

b.add_row([1,str(len(cluster_1))])
b.add_row([2,str(len(cluster_2))])
b.add_row([3,str(len(cluster_3))])
b.add_row([4,str(len(cluster_4))])
b.add_row([5,str(len(cluster_5))])
b.add_row([6,str(len(cluster_6))])
print(b)
```

```
 The number of datapoints in each cluster
========================================================================
================================================
+---------+-----------------------+
| Cluster | Number of Data Points |
+---------+-----------------------+
|    1    |         25081         |
|    2    |          3266         |
|    3    |          4254         |
|    4    |         10341         |
|    5    |          3041         |
|    6    |          4017         |
+---------+-----------------------+
```

Wordcloud for each cluster:

Cluster 1

- Getting the sample reviews in Cluster 1

In [249]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(25081,size=3)
```

In [250]:
```
rand_num = list(rand_num)
```

In [251]:
```
rand_num
```

Out[251]:
```
[19568, 10550, 14639]
```

In [252]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_1[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [253]:
```
string_1 = " ".join(word_cloud)
```

In [254]:
```
string_1
```

Out[254]:
```
'use decor ginger bread kid actual pick bread eat curious tri tast good
liven ginger bread decor tast product high recommend better altern regu
lar cook oil healthi good bodi use yeast bread machin bake best yeast m
arket'
```

In [255]:
```
wordcloud_1 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 2

- Getting the sample reviews in Cluster 2

In [256]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(3266,size=3)
rand_num = list(rand_num)
```

In [257]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_2[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [258]:
```
string_2 = " ".join(word_cloud)
```

In [259]:
```
string_2
```

Out[259]:
```
'four older cat one seem ill two horribl skin chang food trip vet withi
n week skin healthi look gorgeous worth iam scienc diet cat would eat w
ild one cheapest peopl planet worth hard earn way cheaper trip vet trul
i believ food save cat life treat buy dog one crohn diseas give servic
excel receiv quick not first time bought shipper not last best friend a
ge shetland sheepdog name jake absolut love treat use get local pet sto
re recent move not find anywher nearbi cours turn amazon happili half p
rice ship use pay jake hip dysplasia surgeri young adult dog walk notic
limp slowli becom wors shun glucosamin supplement littl help found wond
er treat gait not seem degrad happi still get love littl dog piec anyth
make life littl better littl longer great thing enjoy eat make godsend'
```

In [260]:
```
wordcloud_2 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 3

- Getting the sample reviews in Cluster 3

In [261]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(4254,size=3)
rand_num = list(rand_num)
```

In [262]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_3[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [263]:
```
string_3 = " ".join(word_cloud)
```

In [264]:
```
string_3
```

Out[264]:
```
'tea lover not touch coffe not even mocha coffe icecream year gotten kn
ow various differ tea tea brand adagio one best brand tri price better
averag tea qualiti yunnan gold adagio premium grade yunnan tea lower gr
ade yunnan jig quit good inexpens yunnan gold fantast assert tea abl st
and nice milk sugar not slightest bit bitter black still tasti prefer b
lack tea light sweet even accident steep steep minut smooth mellow natu
r sweet tea complex flavor spici yet subtl fyi consid black tea black t
ea name base upon long allow ferment green tea least ferment oolong fer
ment black tea allow reach complet ferment general speak tea ferment ca
ffien contain perfect day tea pleasur first thing morn yet delight day
night caffien insomnia avoid drink caffien beverag pm like bold tea lik
e hearti assam cylon tea might delic although still suggest give tri mi
ght seduc mani virtu ad adult children easter basket truli enjoy tast l
ight crisp total delici plan use rest tea parti hostess gift blend litt
l appl cider mix tazo passion tea see happen creat magic appl cider exp
eri hint cranberri hibiscus flower gather ancient land snuggl spice cid
er good choic spoon exact amount desir packet cider work equal well typ
e tea would expect enjoy ski distant forest also munch homemad browni e
ggnog cooki appl cider also work well gypsi tea especi lemon jasmin dee
p purpl red color hibiscus fragranc tazo tea enough make anyon passion
tea yet realli tea look like tea realli herbal infus made steep hibiscu
s flower orang peel licoric cinnamon rose hip lemongrass red poppi swir
l hot water origin flavor also slight reminisc hot appl cider hot cranb
erri juic extra delici cider mix tazo tea mysteri experi lose sens aban
don far tri flavor count alway impress rebecca review'
```

In [265]:
```
wordcloud_3 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 4

- Getting the sample reviews in Cluster 4

In [266]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(10341,size=3)
rand_num = list(rand_num)
```

In [267]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_4[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [268]:
```
string_4 = " ".join(word_cloud)
```

In [269]:
```
string_4
```

Out[269]:
```
'favorit starbuck roast dont go local starbuck purchas price go dont pa
y ship handel stuff plus grind flour wet anoth product bought open spil
l flour salvag flour could not contact amazon problem tri follow proced
ur not sucess product not cheap not sure anoth order amazon add kid mil
k ad protein calcium clump togeth mix blender tast great worth price av
oid hormon non organ milk hard find around glad amazon carri offer chea
per special order health food store add homemad ice ice cream'
```

In [270]:
```
wordcloud_4 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 5

- Getting the sample reviews in Cluster 5

In [271]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(3041,size=3)
rand_num = list(rand_num)
```

In [272]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_5[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [273]:
```
string_5 = " ".join(word_cloud)
```

In [274]:
```
string_5
```

Out[274]:
```
'anoth great coffe lavazza usual use cafe espresso blend one even bette
r finer cafe espresso crema refin tast brew daili gaggia revolut deligh
t decaf gevalia coffe cappucino delici come perfect everi time tassimo
unfortun varieti decaf coffe avail current excel come close coffe hous
qualiti purchas coffe regular donat urban soup kitchen fall winter no b
rand found come close price especi discount whatev purchas not speak ta
st not coffe drinker homeless client full prais ad coffe day menu homel
ess indig plenti complain live yet complain coffe coffe vacuum pack rea
lli help origin fresh open flavor get pack'
```

In [275]:
```
wordcloud_5 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 6

- Getting the sample reviews in Cluster 6

In [276]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(4017,size=3)
rand_num = list(rand_num)
```

In [277]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_6[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [278]:
```
string_6 = " ".join(word_cloud)
```

In [279]:
```
string_6
```

Out[279]:
```
'would high recommend chocol marzipan hous friend famili honest best wh
ite chocol ever like white chocol tri one not good snack bitter tast no
t surpris though chocol bitter tast without sugar raw defin get great s
hake recepi flavor realli come'
```

In [280]:
```
wordcloud_6 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Plotting The Wordcloud

- **Cluster 1**

In [281]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_1)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Observation:

- This cluster says about baked goods.

- **Cluster 2**

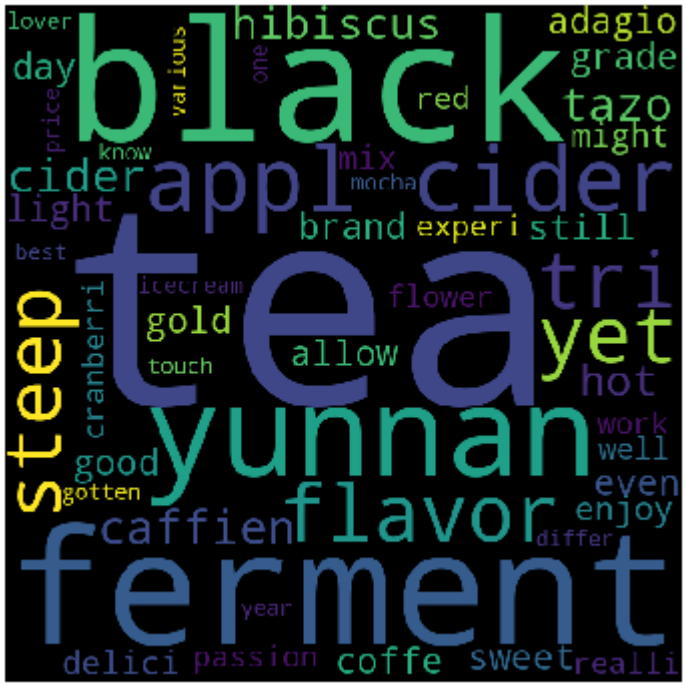
In [282]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_2)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Observation:

- This cluster says about pet food products.

- **Cluster 3**

In [283]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_3)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

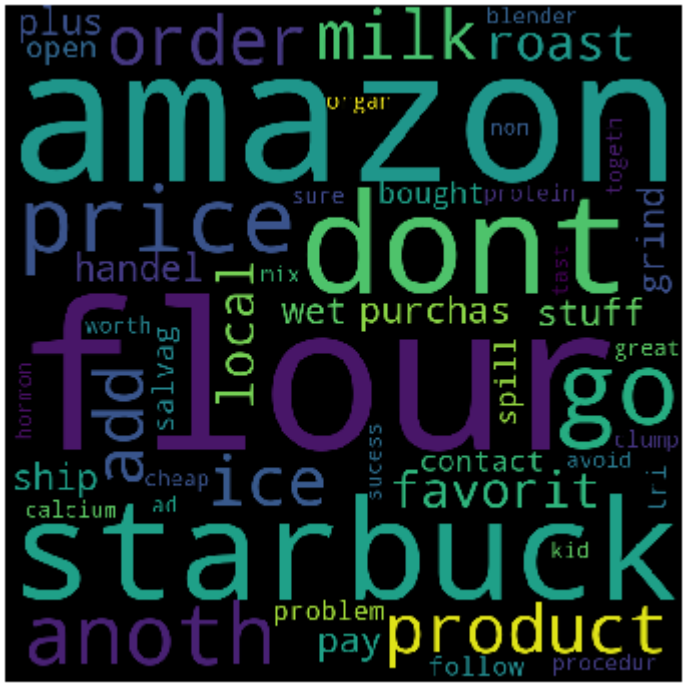


Observation:

- This cluster says about tea products.

- **Cluster 4**

In [284]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_4)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

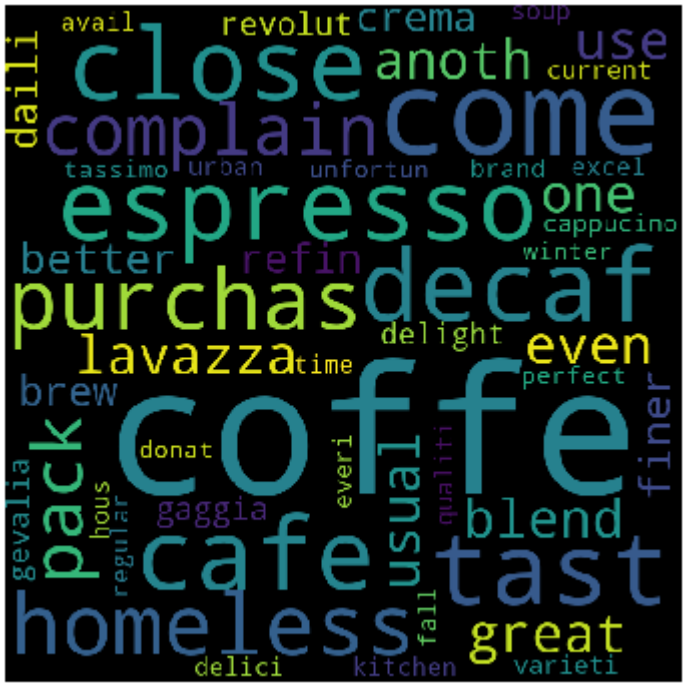


Observation:

- This cluster says about dairy products.

- **Cluster 5**

In [285]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_5)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- This cluster says about coffee products.
- **Cluster 6**

In [286]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_6)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Observation:

- This cluster says about choco products,product quality and taste.

Performance Metric of K means using TFIDF

In [287]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_me

from sklearn.metrics import silhouette_score
```

In [288]:
```
score = silhouette_score(tfidf_train_vec1,labels)
```

In [289]:
```
score
```

Out[289]:
```
0.02185273520661483
```

Observation:

- As per the silhouette score document if the score is nearest to the Zero. The Clusters are Overlapped. So here The Silhouette Score is 0.02. So here the chance of clusters

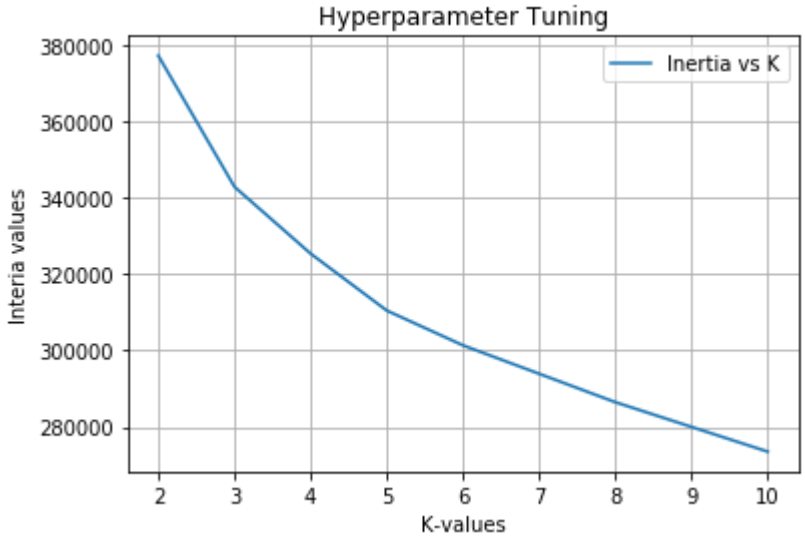
overlapping is high.

4.5 K-Means using Avg W2V

In [290]:
```
# Hyperparameter tuning

k = [2,3,4,5,6,8,10]

inertias=[]

for i in tqdm(k):

    model = KMeans(n_clusters=i,n_jobs=-1)
    model.fit(avg_w2v_train)
    sum_sq_values = model.inertia_
    inertias.append(sum_sq_values)
```

```
  0%|          | 0/7 [00:00<?, ?it/s]
 14%|▉         | 1/7 [00:03<00:22,  3.74s/it]
 29%|██▊       | 2/7 [00:05<00:14,  2.99s/it]
 43%|████▎     | 3/7 [00:06<00:10,  2.68s/it]
 57%|█████▋    | 4/7 [00:09<00:07,  2.55s/it]
 71%|███████▏  | 5/7 [00:12<00:05,  2.72s/it]
 86%|████████▌ | 6/7 [00:16<00:03,  3.21s/it]
100%|██████████| 7/7 [00:21<00:00,  3.74s/it]
```

In [291]:
```
# plotting the k vs inertia

plt.close()
plt.plot(k,inertias,label="Inertia vs K")
plt.grid()
plt.title("Hyperparameter Tuning")
plt.xlabel("K-values")
plt.ylabel("Interia values")
plt.legend()
plt.show()
```



Observation:

- By using the elbow method the best k (number of clusters) is 6

In [292]:
```
# Applying Best Hyperparameter

model= KMeans(n_clusters=6,n_jobs=-1)
model.fit(avg_w2v_train)
labels=model.labels_
```

Number Datapoints in Each Cluster

In [293]:
```
# Data points seperation as per the clusters

number_points = labels.shape[0]
print("Number of Datapoints")
print(number_points)
```

```
Number of Datapoints
50000
```

In [294]:
```
# Datapoints divided by clusters as per the label name

cluster_1=[]
cluster_2=[]
cluster_3=[]
cluster_4=[]
cluster_5=[]
cluster_6=[]

for i in range(0,number_points):

    if labels[i] == 0:
        cluster_1.append(i)
    if labels[i] == 1:
        cluster_2.append(i)
    if labels[i] == 2:
        cluster_3.append(i)
    if labels[i] == 3:
        cluster_4.append(i)
    if labels[i] == 4:
        cluster_5.append(i)
    if labels[i] == 5:
        cluster_6.append(i)
```

In [295]:
```
# The number of datapoints in each cluster

c=PrettyTable()

c.field_names = ["Cluster", "Number of Data Points"]

print(" The number of datapoints in each cluster")
print("="*120)

c.add_row([1,str(len(cluster_1))])
c.add_row([2,str(len(cluster_2))])
c.add_row([3,str(len(cluster_3))])
c.add_row([4,str(len(cluster_4))])
c.add_row([5,str(len(cluster_5))])
c.add_row([6,str(len(cluster_6))])
print(c)
```

```
 The number of datapoints in each cluster
========================================================================
================================================
+---------+-----------------------+
| Cluster | Number of Data Points |
+---------+-----------------------+
|    1    |          3979         |
|    2    |         11841         |
|    3    |          7379         |
|    4    |          6349         |
|    5    |         11653         |
|    6    |          8799         |
+---------+-----------------------+
```

Wordcloud for each cluster:

Cluster 1

- Getting the sample reviews in Cluster 1

In [296]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(3979,size=3)
```

In [297]:
```
rand_num = list(rand_num)
```

In [298]:
```
rand_num
```

Out[298]:
```
[993, 3511, 3145]
```

In [299]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_1[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [300]:
```
string_1 = " ".join(word_cloud)
```

In [301]:
```
string_1
```

Out[301]:
```
'system litter box clean much better scoop hand especi multipl cat howe
v thing not perfect clump get stuck sifter especi urin fresh end wipe a
nyway outsid buy electron self clean litter box not read great thing sy
stem job pretti painless dog love even goe counter stare contain time b
one cat see bag come run love death one girl know age year old'
```

In [302]:
```
wordcloud_1 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 2

- Getting the sample reviews in Cluster 2

In [303]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(11841,size=3)
rand_num = list(rand_num)
```

In [304]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_2[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [305]:
```
string_2 = " ".join(word_cloud)
```

In [306]:
```
string_2
```

Out[306]:
```
'search local store month find brook rich tangi ketchup final heard no
longer made went internet see could find bottl anywher discov made cana
da also found could find websit chanc bought case share bottl famili fr
iend rais ketchup ever put tabl make barbecu sauc give extra tang want
not ketchup person want kind want eat spent first year life germani rem
emb first time got care packag germani gummi bear thrill noth like eat
origin gummi candi second round herb kit batch purchas three kit mix po
d make fall winter mix cilantro basil pars thyme savori sage oregano ba
sil thyme savori oregano new italian herb kit everyth plant either spro
ut day earli right time savori appear delic bunch grown tallest three w
eek start basil first sprout pars seem slow steadi herb good least six
month proper care sure could extend life easi care great alway fresh he
rb dispos recom product anyon love fresh herb anyon problem grow herb p
ast'
```

In [307]:
```
wordcloud_2 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 3

- Getting the sample reviews in Cluster 3

In [308]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(7379,size=3)
rand_num = list(rand_num)
```

In [309]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_3[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [310]:
```
string_3 = " ".join(word_cloud)
```

In [311]:
```
string_3
```

Out[311]:
```
'coffe good organ better would tell tri coffe nantucket blend one favor
it k cup coffe made medium blend coffe tast great flavor creamer regula
r creamer milk definit buy one love keurig coffe maker discov buy coffe
thru amazon littl cheaper prime ship flavor like rich strong coffe hint
espresso flavor'
```

In [312]:
```
wordcloud_3 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 4

- Getting the sample reviews in Cluster 4

In [313]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(6349,size=3)
rand_num = list(rand_num)
```

In [314]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_4[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [315]:
```
string_4 = " ".join(word_cloud)
```

In [316]:
```
string_4
```

Out[316]:
```
'brussel bonsai best ship say ship product alway top notch custom servi
c sale fantast find excel qualiti almond reson price realli go fast go
order soon big cheez favorit nowher found area reli mail order free shi
p better order direct jolli time plus time order amazon special offer d
iscount spend bought two case keep popcorn til next deal come around'
```

In [317]:
```
wordcloud_4 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 5

- Getting the sample reviews in Cluster 5

In [318]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(11653,size=3)
rand_num = list(rand_num)
```

In [319]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_5[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [320]:
```
string_5 = " ".join(word_cloud)
```

In [321]:
```
string_5
```

Out[321]:
```
'one best maca powder ever tri tast excel energet effect notic high rec
ommend nativa natur product across board nut tast fresh hint sweet dark
chocol almond tast dark chocol great thing chocol flavor compliment alm
ond well great crunch no tast almond uniform size great snack anytim da
y canist generous size great valu money high recommend give almond emer
ald tri emerald sea salt pepper cashew great flavor not over oili like
nut tri not howev peopl low toler black pepper plenti cling right nut n
otic aftertast pepper stay sea salt not overpow tast better ordinari sa
lt game qualiti product seek emerald nut tri futur'
```

In [322]:
```
wordcloud_5 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 6

- Getting the sample reviews in Cluster 6

In [323]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(8799,size=3)
rand_num = list(rand_num)
```

In [324]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_6[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [325]:
```
string_6 = " ".join(word_cloud)
```

In [326]:
```
string_6
```

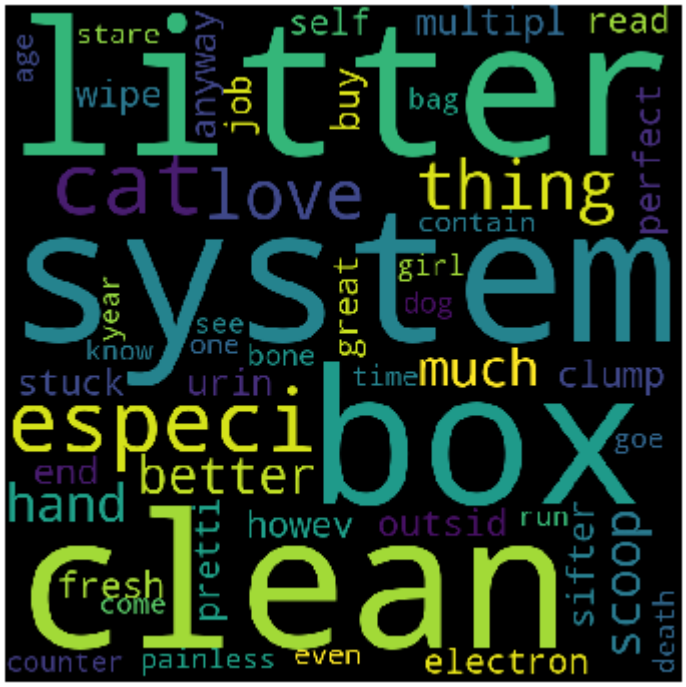
Out[326]:
```
'product recommend us head chef resort jamaica wife use ever stuff grea
t chicken leg like spici go would not dare argu nutrit valu salt diet e
njoy flavor salt food found salt much better salt us particular enjoy s
alt light sprinkl authent butter french baguett c est tres bon great mu
stard get low sodium diet great way jazz low sodium sauc found combin z
ero sodium bbq sauc shot tabasco creat sauc even enjoy no salt bread sw
iss chees homemad salt free mayonnais popular brand lower sodium turkey
get decent sandwich sodium also work great dip unsalt pretzel thing kee
p four star better experi make scratch'
```

In [327]:
```
wordcloud_6 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Plotting The Wordcloud

- **Cluster 1**

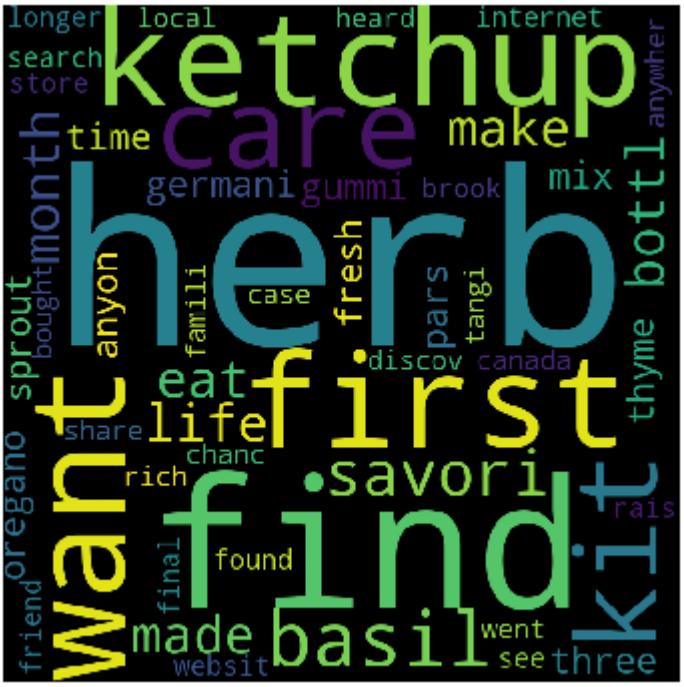
In [328]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_1)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- This cluster says about how the product tastes and quality
- **Cluster 2**

In [329]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_2)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

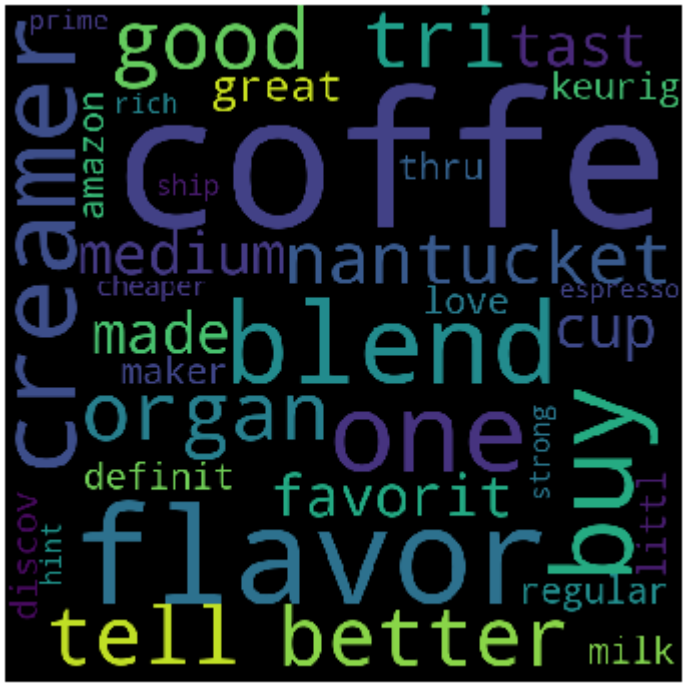


Observation:

- This cluster says about how the product tastes and quality.

- **Cluster 3**

In [330]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_3)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

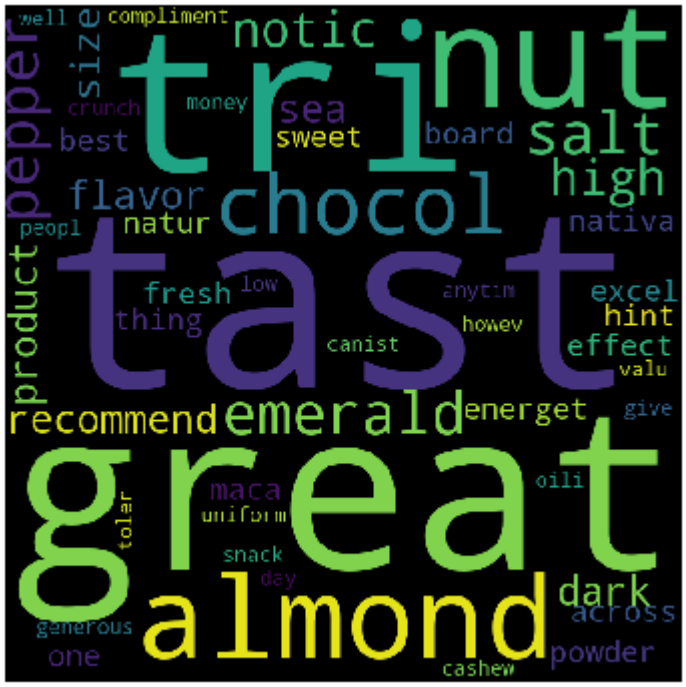
- This cluster says about coffee products.

- **Cluster 4**

In [331]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_4)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

- **Cluster 5**

In [332]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_5)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

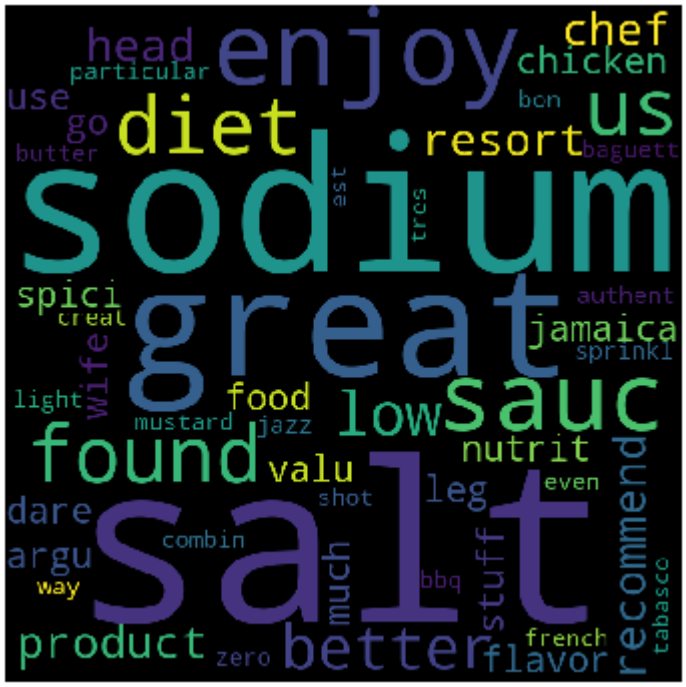


Observation:

- This cluster says about choco products.

- **Cluster 6**

In [333]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_6)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- This cluster says about cooking powder products.

Performance Metric of K means using Avg W2V

In [334]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_me

from sklearn.metrics import silhouette_score
```

In [335]:
```
score = silhouette_score(avg_w2v_train,labels)
```

In [336]:
```
score
```

Out[336]: 0.08058237853777399

Observation:

- As per the silhouette score document if the score is nearest to the Zero. The Clusters are Overlapped. So here The Silhouette Score is 0.08. So here the chance of clusters

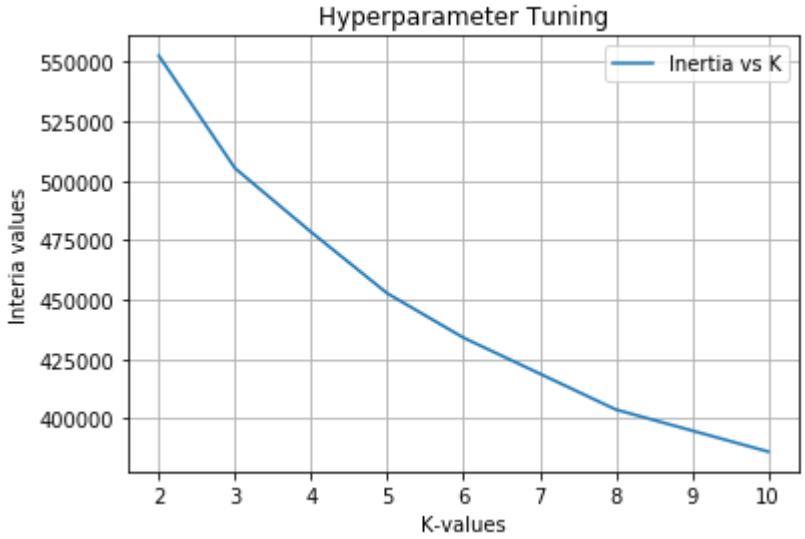
overlapping is high.

4.6 K-Means using TFIDF W2V

In [337]:
```
# Hyperparameter tuning

k = [2,3,4,5,6,8,10]

inertias=[]

for i in tqdm(k):

    model = KMeans(n_clusters=i,n_jobs=-1)
    model.fit(tfidf_w2v_train)
    sum_sq_values = model.inertia_
    inertias.append(sum_sq_values)
```

```
  0%|          | 0/7 [00:00<?, ?it/s]
 14%|▉         | 1/7 [00:03<00:20,  3.44s/it]
 29%|██▊       | 2/7 [00:04<00:14,  2.82s/it]
 43%|████▎     | 3/7 [00:06<00:10,  2.51s/it]
 57%|█████▋    | 4/7 [00:09<00:07,  2.52s/it]
 71%|███████▏  | 5/7 [00:11<00:05,  2.53s/it]
 86%|████████▌ | 6/7 [00:15<00:02,  2.79s/it]
100%|██████████| 7/7 [00:20<00:00,  3.51s/it]
```

In [338]:
```
# plotting the k vs inertia

plt.close()
plt.plot(k,inertias,label="Inertia vs K")
plt.grid()
plt.title("Hyperparameter Tuning")
plt.xlabel("K-values")
plt.ylabel("Interia values")
plt.legend()
plt.show()
```



Observation:

- By using the elbow method the best k (number of clusters) is 6

In [339]:
```
# Applying Best Hyperparameter

model= KMeans(n_clusters=6,n_jobs=-1)
model.fit(tfidf_w2v_train)
labels=model.labels_
```

Number Datapoints in Each Cluster

In [340]:
```
# Data points seperation as per the clusters

number_points = labels.shape[0]
print("Number of Datapoints")
print(number_points)
```

```
Number of Datapoints
50000
```

In [341]:
```
# Datapoints divided by clusters as per the label name

cluster_1=[]
cluster_2=[]
cluster_3=[]
cluster_4=[]
cluster_5=[]
cluster_6=[]

for i in range(0,number_points):

    if labels[i] == 0:
        cluster_1.append(i)
    if labels[i] == 1:
        cluster_2.append(i)
    if labels[i] == 2:
        cluster_3.append(i)
    if labels[i] == 3:
        cluster_4.append(i)
    if labels[i] == 4:
        cluster_5.append(i)
    if labels[i] == 5:
        cluster_6.append(i)
```

In [342]:
```
# The number of datapoints in each cluster

b=PrettyTable()

b.field_names = ["Cluster", "Number of Data Points"]

print(" The number of datapoints in each cluster")
print("="*120)

b.add_row([1,str(len(cluster_1))])
b.add_row([2,str(len(cluster_2))])
b.add_row([3,str(len(cluster_3))])
b.add_row([4,str(len(cluster_4))])
b.add_row([5,str(len(cluster_5))])
b.add_row([6,str(len(cluster_6))])
print(b)
```

```
 The number of datapoints in each cluster
========================================================================
================================================
+---------+-----------------------+
| Cluster | Number of Data Points |
+---------+-----------------------+
|    1    |         14585         |
|    2    |         14848         |
|    3    |          3533         |
|    4    |          3894         |
|    5    |          3131         |
|    6    |         10009         |
+---------+-----------------------+
```

Wordcloud for each cluster:

Cluster 1

- Getting the sample reviews in Cluster 1

In [343]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(14585,size=3)
```

In [344]:
```
rand_num = list(rand_num)
```

In [345]:
```
rand_num
```

Out[345]:
```
[5869, 6581, 13523]
```

In [346]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_1[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [347]:
```
string_1 = " ".join(word_cloud)
```

In [348]:
```
string_1
```

Out[348]:
```
'drank can bottl green tea discov lipton varieti not spend money over s
weeten product uniqu pyramid bag get much flavor intens tea without ad
sweeten must add sugar honey tast like orang flavor without doctor also
like amazon great price zevia twist tast refresh flavor natur oil lemon
lime realli come water drinker soda never quench thirst stuff healthi n
one artifici sweeten color flavor chemic regular diet soda tri believ n
ice light refresh carbon fruit juic someth like orangina far natur tast
make sens unlik orangina juic no ad sweeten complaint ounc rather small
general prefer larger drink definit not bargain drink two can time'
```

In [349]:
```
wordcloud_1 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 2

- Getting the sample reviews in Cluster 2

In [350]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(14848,size=3)
rand_num = list(rand_num)
```

In [351]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_2[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [352]:
```
string_2 = " ".join(word_cloud)
```

In [353]:
```
string_2
```

Out[353]:
```
'love aero garden second crop grow seven differ kind basil seed germin
within three day expect begin harvest anoth week two request addit aero
garden mother day present use grow salad green take fenugreek brewer ye
ast capsul well drink tea day pump everi hrs hour except work hrs day p
ump oz everi singl time first oz time everi hrs talk major increas love
pleasant tastey stevia natur sweeten also high recommend start notic li
ttl bug fli thru home almost immedi receiv tree not make connect right
away window near tree start crawl bug forc throw tree no longer tree ho
us hundr dead bug deal one worst purchas ever'
```

In [354]:
```
wordcloud_2 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 3

- Getting the sample reviews in Cluster 3

In [355]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(3533,size=3)
rand_num = list(rand_num)
```

In [356]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_3[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [357]:
```
string_3 = " ".join(word_cloud)
```

In [358]:
```
string_3
```

Out[358]:
```
'favorit tea use make ice tea everi day great deal mani tea bag not run
tea bag love tea add littl honey sit back relax slight citrus flavor fi
nd refressh want drink tradit tea one grab strong smooth no bitter meta
l tast use drink twine irish breakfast tea tea bag box pack one day gro
cer pick instead wow lucki often drink bigelow vanilla chai tea box pac
k black tea delici vanilla spice tradit black tea found none like bette
r english breakfast tea'
```

In [359]:
```
wordcloud_3 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 4

- Getting the sample reviews in Cluster 4

In [360]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(3894,size=3)
rand_num = list(rand_num)
```

In [361]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_4[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [362]:
```
string_4 = " ".join(word_cloud)
```

In [363]:
```
string_4
```

Out[363]:
```
'dog love edibl bone floppier brown rubber bone favorit long time flopp
i brown bone survi puppi teeth almost no damag white bone realli stiff
hard almost plastic feel no interest age not like either ignor complet
not even bother buri backyard favorit thing treat toy mayb dog like bet
ter bone great black lab diamond dog food puppi sever varieti dog start
puppi food mainten lab mainten formula year old love addit make coat sh
ini fantast product bull terrier capabl spit pill no matter insert pean
ut butter steak chicken hot dog name tri patient wait get pill readi gu
lp absolut amaz never without product'
```

In [364]:
```
wordcloud_4 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 5

- Getting the sample reviews in Cluster 5

In [365]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(3131,size=3)
rand_num = list(rand_num)
```

In [366]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_5[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [367]:
```
string_5 = " ".join(word_cloud)
```

In [368]:
```
string_5
```

Out[368]:
```
'drank one commut cup morn would probabl need back caffein notch two es
presso made stove top espresso maker delici almost worth get bed great
robust charg morn add water prefer less strong brew sprinkl cinnamon me
llow hook part daili ritual certain get well blend coffe wonder full bo
di flavor top five tri dozen blend yes dozen green mountain distinguish
leader k cup pack not much flavor coffe though appreci aroma green moun
tain coffe especi bold extra bold robust plain tast great'
```

In [369]:
```
wordcloud_5 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 6

- Getting the sample reviews in Cluster 6

In [370]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(10009,size=3)
rand_num = list(rand_num)
```

In [371]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_6[i])

for i in index:

    word_cloud.append(X.values[i])
```

In [372]:
```
string_6 = " ".join(word_cloud)
```

In [373]:
```
string_6
```

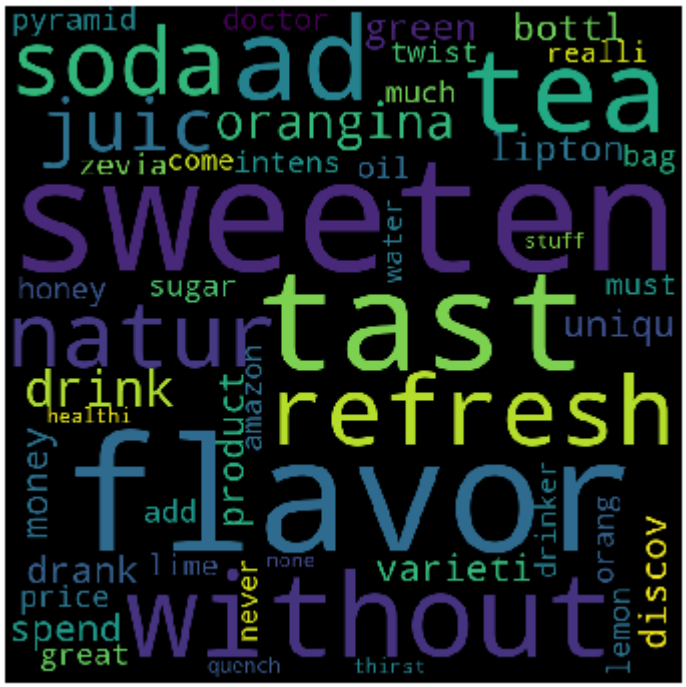
Out[373]:
```
'everybodi say uncl timmi chili best slave hour simmer right friend fam
ili love best alway request baloney pull fantast chili trick year no ki
d chili around long good easi anyth els pound hamburg kidney bean dice
tomato packet chili throw dice onion done shoot make night not want coo
k easi truth ever discontinu chili realli creek best secret whole kitch
en well not rate probabl not haha use product year no bitter tast chemi
c use extract alo plant one drink gargl swish mouth use sinus irrig fem
inin use would use colon irrit oral prevent part colon cancer therapi a
sk health practition close theatr top could good tast good flavor book
say popcorn'
```

In [374]:
```
wordcloud_6 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Plotting The Wordcloud

- **Cluster 1**

In [375]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_1)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

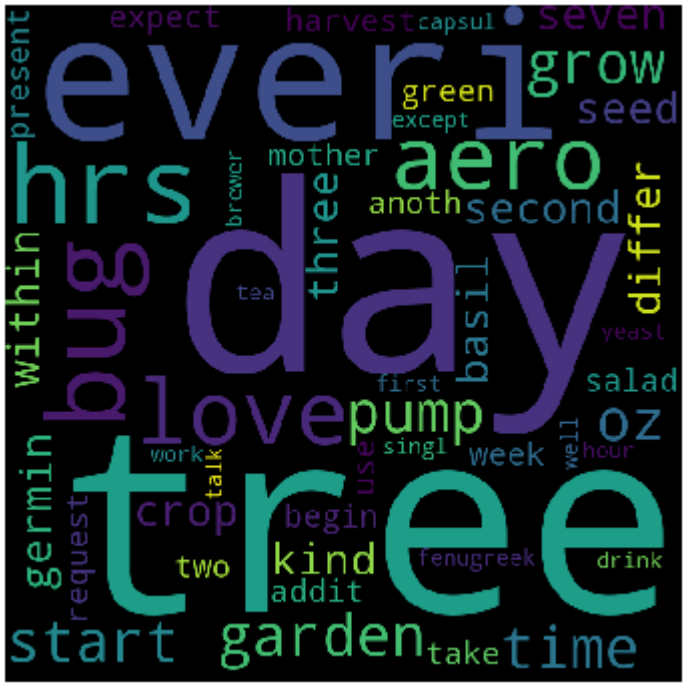


Observation:

- This cluster says about Green Tea like products.

- **Cluster 2**

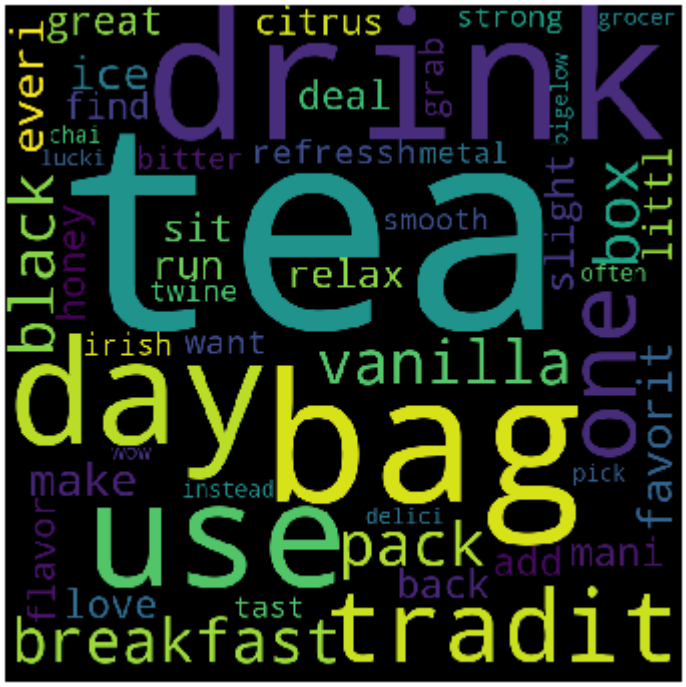
In [376]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_2)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- This cluster says about how the product tastes and quality
- **Cluster 3**

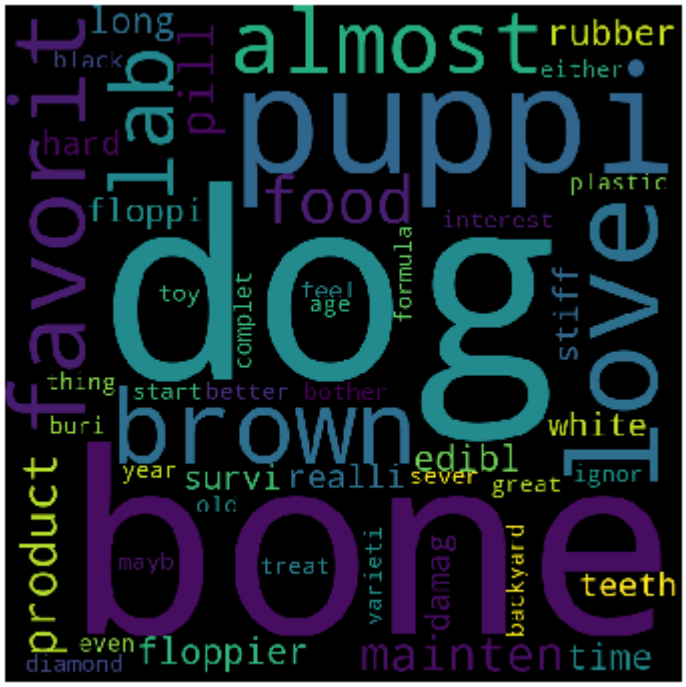
In [377]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_3)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- This cluster says about tea products
- **Cluster 4**

In [378]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_4)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

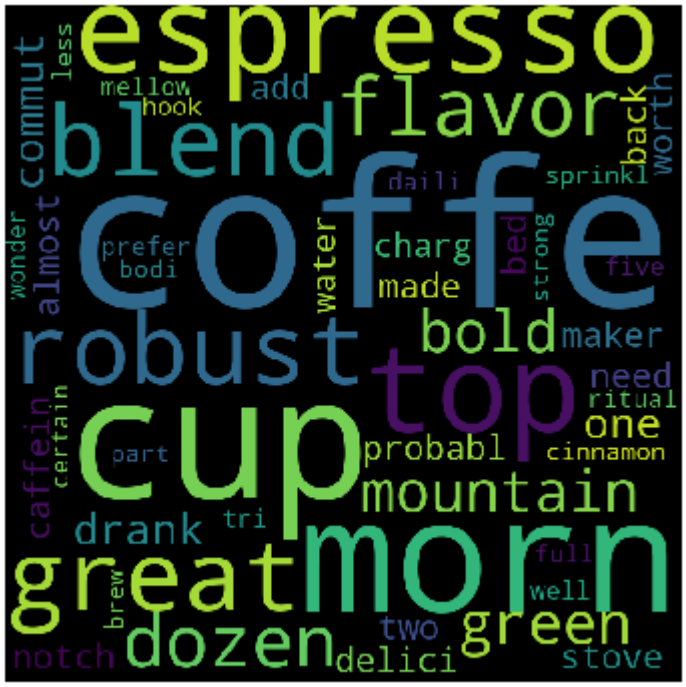


Observation:

- This cluster says about pet products.

- **Cluster 5**

In [379]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_5)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

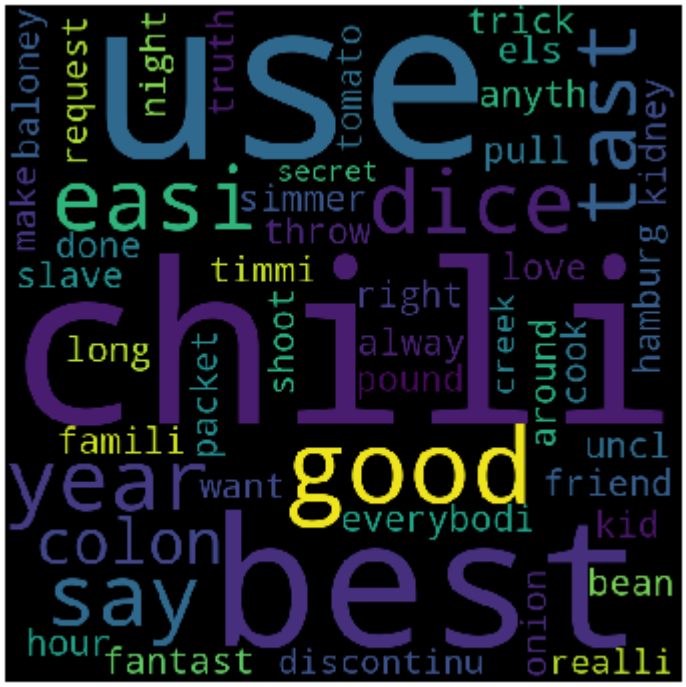


Observation:

- This cluster says about coffee products.

- **Cluster 6**

In [380]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_6)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- This cluster says about cooking powder products.

Performance Metric of K means using TFIDF W2V

In [381]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_me

from sklearn.metrics import silhouette_score
```

In [382]:
```
score = silhouette_score(tfidf_w2v_train,labels)
```

In [383]:
```
score
```

Out[383]:
```
0.11610920768204656
```

Observation:

- As per the silhouette score document if the score is nearest to the Zero. The Clusters are Overlapped. So here The Silhouette Score is 0.12. So here the chance of clusters

overlapping is high.

5. Agglomerative Clustering

5.1 Data

In [384]:
```
# we took the sample data size as 5k

final_data=filter_data[:5000]
final_data.shape
```

Out[384]: (5000, 10)

In [385]:
```
Y=final_data.Text
```

5.2 Featurization

5.2.1 W2V

In [387]:
```
# References
# https://radimrehurek.com/gensim/models/word2vec.html
# https://machinelearningmastery.com/develop-word-embeddings-python-gens
# http://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/#.W17S

from gensim.models import Word2Vec
```

In [388]:
```
list_sentences_train=[]
for i in tqdm(list(Y)):
    list_sentences_train.append(i.split())
```

```
  0%|          | 0/5000 [00:00<?, ?it/s]
100%|██████████| 5000/5000 [00:00<00:00, 150800.47it/s]
```

In [389]:
```
word2vec_model=Word2Vec(list_sentences_train,min_count=5,size=50,workers=
```

In [390]:
```
word2vec_words_train=list(word2vec_model.wv.vocab)
print(" Number of words")
print("_____________________")
print(" ")
print(len(word2vec_words_train))
print("="*125)
print(" sample words")
print("_____________________")
print(" ")
print(word2vec_words_train[100:150])
```

```
 Number of words
_____________________

3966
================================================================================
================================================================
 sample words
_____________________

['fill', 'comedi', 'action', 'whatev', 'els', 'want', 'call', 'enjoy',
'entertain', 'hesit', 'pick', 'edit', 'guess', 'market', 'plan', 'famil
i', 'elimin', 'strong', 'element', 'usual', 'version', 'warn', 'uncut',
'avoid', 'apart', 'fruit', 'fli', 'hour', 'trap', 'quot', 'attract', 'm
ani', 'within', 'practic', 'gone', 'may', 'long', 'term', 'solut', 'cra
zi', 'consid', 'buy', 'caution', 'surfac', 'sticki', 'tri', 'touch', 'h
appen', 'say', 'name']
```

5.2.2 Avg W2V

In [391]:
```
# Reference
# formula of Avg word2vec = sum of all (wi)[i=0 to n]/n

# avg word2vec on training data

avg_word2vec_train=[]
for i in list_sentences_train:
    vector=np.zeros(50)
    no_of_words=0
    for k in i:
        try:
            w2v_data=word2vec_model.wv[k]
            vector=vector+w2v_data
            no_of_words=no_of_words+1
        except:
            pass
    if no_of_words != 0:
        vector=vector/no_of_words
    avg_word2vec_train.append(vector)
avg_w2v_train1=np.asmatrix(avg_word2vec_train)
print("shape of Avg Word2vec train")
print(avg_w2v_train1.shape)
```

```
shape of Avg Word2vec train
(5000, 50)
```

5.2.3 TFIDF W2V

In [393]:
```
# References
# https://stackoverflow.com/questions/21553327
# https://github.com/devBOX03


# tfidf word2vec on training data

model=TfidfVectorizer()
tfidf_w2v_model=model.fit_transform(Y)
tfidf_w2v=model.get_feature_names()
tfidf_word2vec_train=[]
row=0
for i in list_sentences_train:
    vec=np.zeros(50)
    weight_sum=0
    for w in i:
        try:
            w2v_freq=word2vec_model.wv[w]
            tfidf_freq=tfidf_w2v_model[row,tfidf_w2v.index(w)]
            vec=vec+(w2v_freq*tfidf_freq)
            weight_sum=weight_sum+tfidf_freq
        except:
            pass
    vec=vec/weight_sum
    tfidf_word2vec_train.append(vec)
    row=row+1
tfidf_w2v_train1=np.asmatrix(tfidf_word2vec_train)
print("Shape of TFIDF word2vec train")
print(tfidf_w2v_train1.shape)
```

```
Shape of TFIDF word2vec train
(5000, 50)
```

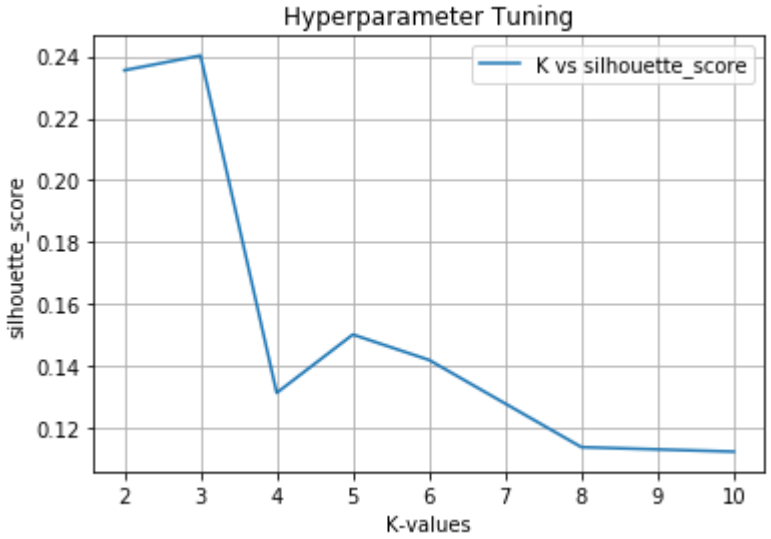
5.3Agglomerative using Avg W2V

In [394]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Aggl

from sklearn.cluster import AgglomerativeClustering
```

In [396]:
```
# Hyperparameter tuning

k = [2,3,4,5,6,8,10]

silhouette_score_value=[]

for i in tqdm(k):

    model = AgglomerativeClustering(n_clusters=i)
    model.fit(avg_w2v_train1)
    labels = model.labels_
    score = silhouette_score(avg_w2v_train1,labels)
    silhouette_score_value.append(score)
```

```
  0%|          | 0/7 [00:00<?, ?it/s]
 14%|▉         | 1/7 [00:02<00:12,  2.07s/it]
 29%|██▊       | 2/7 [00:04<00:10,  2.05s/it]
 43%|████▎     | 3/7 [00:06<00:08,  2.03s/it]
 57%|█████▋    | 4/7 [00:08<00:06,  2.02s/it]
 71%|███████   | 5/7 [00:10<00:04,  2.01s/it]
 86%|████████▌ | 6/7 [00:12<00:01,  2.00s/it]
100%|██████████| 7/7 [00:14<00:00,  2.00s/it]
```

In [397]:
```
# plotting the k vs inertia

plt.close()
plt.plot(k,silhouette_score_value,label="K vs silhouette_score")
plt.grid()
plt.title("Hyperparameter Tuning")
plt.xlabel("K-values")
plt.ylabel("silhouette_score")
plt.legend()
plt.show()
```



Observation:

- By using the silhouette_score value the best k (number of clusters) is 3. (k=3, silhouette_score =0.24)

In [398]:
```
# Applying Best Hyperparameter

model= AgglomerativeClustering(n_clusters=3)
model.fit(avg_w2v_train1)
labels=model.labels_
```

Number Datapoints in Each Cluster

In [399]:
```
# Data points seperation as per the clusters

number_points = labels.shape[0]
print("Number of Datapoints")
print(number_points)
```

```
Number of Datapoints
5000
```

In [400]:
```
# Datapoints divided by clusters as per the label name

cluster_1=[]
cluster_2=[]
cluster_3=[]

for i in range(0,number_points):

    if labels[i] == 0:
        cluster_1.append(i)
    if labels[i] == 1:
        cluster_2.append(i)
    if labels[i] == 2:
        cluster_3.append(i)
```

In [401]:
```
# The number of datapoints in each cluster

d=PrettyTable()

d.field_names = ["Cluster", "Number of Data Points"]

print(" The number of datapoints in each cluster")
print("="*120)

d.add_row([1,str(len(cluster_1))])
d.add_row([2,str(len(cluster_2))])
d.add_row([3,str(len(cluster_3))])
print(d)
```

```
 The number of datapoints in each cluster
========================================================================
================================================
+---------+-----------------------+
| Cluster | Number of Data Points |
+---------+-----------------------+
|    1    |          2559         |
|    2    |          1617         |
|    3    |          824          |
+---------+-----------------------+
```

Wordcloud for each cluster:

Cluster 1

- Getting the sample reviews in Cluster 1

In [402]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(2559,size=3)
```

In [403]:
```
rand_num = list(rand_num)
```

In [404]:
```
rand_num
```

Out[404]:
```
[167, 1816, 1146]
```

In [405]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_1[i])

for i in index:

    word_cloud.append(Y.values[i])
```

In [406]:
```
string_1 = " ".join(word_cloud)
```

In [407]:
```
string_1
```

Out[407]:
```
'worst dri mango product tast threw away whole bag ingredi not read use
sugar preserv thought natur no sugar saw think use bunch sugar also use
preserv well miss ingredi still ok tast good product also use sugar pre
serv howev worst dri mango ever neither pleas nor health alreadi lost m
oney spend not want bother mouth nor health worst threw away whole bag
decis no attach recommend philipin import dire mango product mango fine
refer bought organ natur dri mango quit differ philipin product real dr
i somewhat hard rather soft chewi like philipin product use no sugar no
preserv love flavor unlik sport drink half way canist not tire flavor d
ay day drink stuff day week mix well water provid adequa nutrit endur w
orkout slight odd aftertast bit chalki not tri powder drink not sure no
rmal product type specif product either way like stuff would purchas gi
nger chew amaz thing ginger help nausea sort includ caus chemo motion s
ick heartburn heard even menstrual cramp ailment fresh ginger strong sl
ight calmer bit sweet portabl come individu wrap great travel love ging
er chew ginger peopl'
```

In [408]:
```
wordcloud_1 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 2

- Getting the sample reviews in Cluster 2

In [409]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(1617,size=3)
rand_num = list(rand_num)
```

In [410]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_2[i])

for i in index:

    word_cloud.append(Y.values[i])
```

In [411]:
```
string_2 = " ".join(word_cloud)
```

In [412]:
```
string_2
```

Out[412]:
```
'would high recommend chill freezer first great cold treat not melt not
eat fast enough big fan find cheap long eat within day moist chewi norm
al buy bagel bagel store short fund want yummi snack aorund hous feed f
riend good call also good cream chees cours develop sugarfre recip find
egg invalu ad extra liquid exist recip date not disappoint bake result
yet would purchas'
```

In [413]:
```
wordcloud_2 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 3

- Getting the sample reviews in Cluster 3

In [414]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(824,size=3)
rand_num = list(rand_num)
```

In [415]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_3[i])

for i in index:

    word_cloud.append(Y.values[i])
```

In [416]:
```
string_3 = " ".join(word_cloud)
```

In [417]:
```
string_3
```

Out[417]:
```
'tea not tast like expect long time use loos leaf white tea jasmin enjo
yingtea com realli happi want portabl tea bag offer enjoyingtea not pro
duct offer purchas republ tea asian jasmin white tea cost plus world ma
rket open canist not smell like jasmin rancid candi smell jasmin usual
sweet floral scent tast marri well green white tea tea not smell tast l
ike almost smell like mix orang flavor white tea offer use sparklett bo
ttl water not fan tap water tast tea even better water no light tea tas
t like usual white tea tea turn golden yellow like tang tast not tast l
ike jasmin tap water one could imagin tast would much wors least live r
ead elsewher tea use tea bag usual lower grade usual tast way resort bu
y box tea filter bag make tea bag loos leaf tea alreadi bottom line fam
ilair tast white tea tea bag dissappoint tri loos leaf white tea jasmin
instead tea provid luscious refresh break daili grind help relax well g
ot great lemon tast not overpow let steep three minut bound enjoy much
bag contain natur ingredi dri whole lemon hibiscus rosehip lemon grass
no artifici ingredi stringless tea bag easili remov cup whenev want sim
pli use spoon get also recommend use spoon stir tea bag around bit cup
drink tea also gluten free plus us not want gluten diet decaffein somet
h alway prefer like lemon flavor tea high recommend enjoy smile favorit
coffe medium roast coffe pod senseo douw egbert discov smooth full bodi
coffe purchas senseo coffe maker last fall enjoy full tast expresso lik
e coffe smooth foami textur get senseo coffe maker like gourmet coffe s
hop home everi morn also discov mani differ favor coffe senseo douw egb
ert make enjoy tri differ occas'
```

In [418]:
```
wordcloud_3 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Plotting The Wordcloud

- **Cluster 1**

In [419]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_1)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Observation:

- This cluster says about Negative reviews of the product.

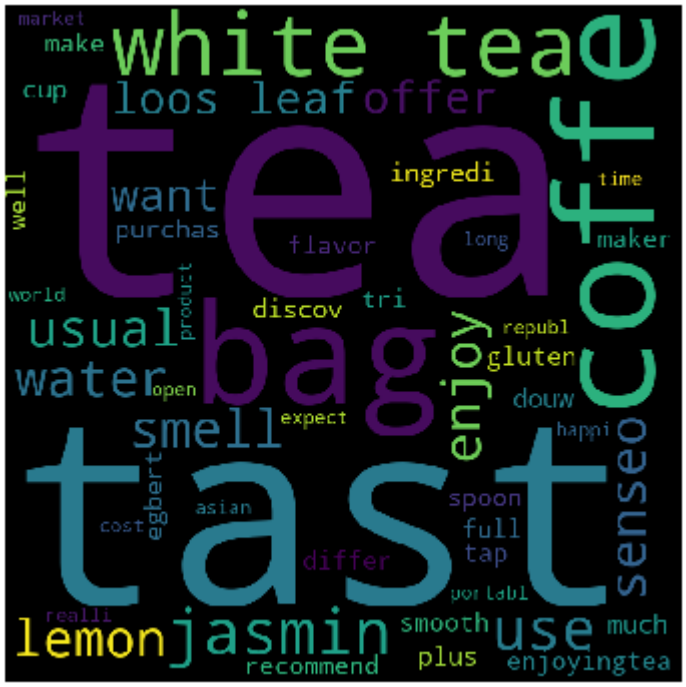
- **Cluster 2**

In [420]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_2)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Observation:

- This cluster says about Positive reviews of the products.
- **Cluster 3**

In [421]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_3)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- This cluster says about tea products.

Performance Metric of Agglomerative using Avg W2V

In [422]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_me

from sklearn.metrics import silhouette_score
```

In [423]:
```
score = silhouette_score(avg_w2v_train1,labels)
```

In [424]:
```
score
```

Out[424]:
```
0.24021730070249372
```

Observation:

- As per the silhouette score document if the score is nearest to the Zero. The Clusters are Overlapped. So here The Silhouette Score is 0.24. So here the chance of clusters

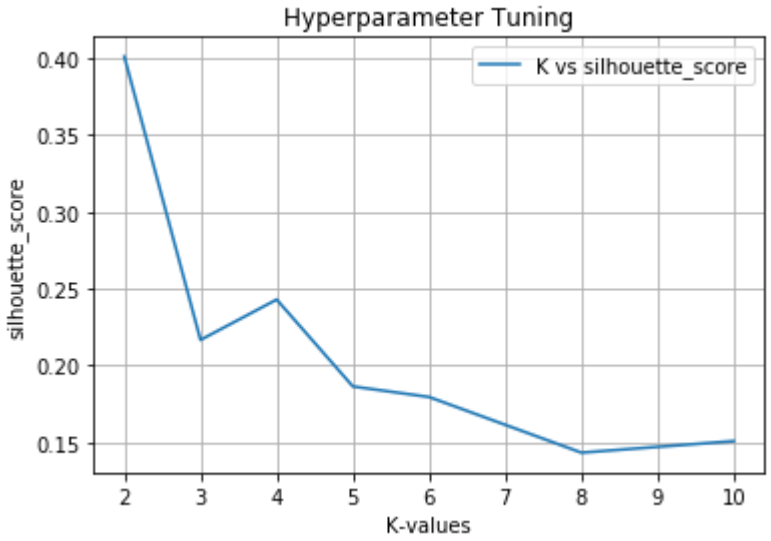
overlapping is high.

5.4 Agglomerative using TFIDF W2V

In [425]:
```
# Hyperparameter tuning

k = [2,3,4,5,6,8,10]

silhouette_score_value=[]

for i in tqdm(k):

    model = AgglomerativeClustering(n_clusters=i)
    model.fit(tfidf_w2v_train1)
    labels = model.labels_
    score = silhouette_score(tfidf_w2v_train1,labels)
    silhouette_score_value.append(score)
```

```
  0%|          | 0/7 [00:00<?, ?it/s]
 14%|▉         | 1/7 [00:02<00:12,  2.13s/it]
 29%|██▊       | 2/7 [00:04<00:10,  2.10s/it]
 43%|████▎     | 3/7 [00:06<00:08,  2.09s/it]
 57%|█████▋    | 4/7 [00:08<00:06,  2.10s/it]
 71%|███████   | 5/7 [00:10<00:04,  2.10s/it]
 86%|████████▌ | 6/7 [00:12<00:02,  2.07s/it]
100%|██████████| 7/7 [00:14<00:00,  2.06s/it]
```

In [426]:
```
# plotting the k vs inertia

plt.close()
plt.plot(k,silhouette_score_value,label="K vs silhouette_score")
plt.grid()
plt.title("Hyperparameter Tuning")
plt.xlabel("K-values")
plt.ylabel("silhouette_score")
plt.legend()
plt.show()
```



Observation:

- By using the silhouette_score value the best k (number of clusters) is 2. (k=2, silhouette_score =0.40)

In [427]:
```
# Applying Best Hyperparameter

model= AgglomerativeClustering(n_clusters=2)
model.fit(tfidf_w2v_train1)
labels=model.labels_
```

Number Datapoints in Each Cluster

In [428]:
```
# Data points seperation as per the clusters

number_points = labels.shape[0]
print("Number of Datapoints")
print(number_points)
```

```
Number of Datapoints
5000
```

In [429]:
```
# Datapoints divided by clusters as per the label name

cluster_1=[]
cluster_2=[]

for i in range(0,number_points):

    if labels[i] == 0:
        cluster_1.append(i)
    if labels[i] == 1:
        cluster_2.append(i)
```

In [431]:
```
# The number of datapoints in each cluster

e=PrettyTable()

e.field_names = ["Cluster", "Number of Data Points"]

print(" The number of datapoints in each cluster")
print("="*120)

e.add_row([1,str(len(cluster_1))])
e.add_row([2,str(len(cluster_2))])
print(e)
```

```
 The number of datapoints in each cluster
========================================================================
================================================
+---------+----------------------+
| Cluster | Number of Data Points |
+---------+----------------------+
|    1    |          4184        |
|    2    |          816         |
+---------+----------------------+
```

Wordcloud for each cluster:

Cluster 1

- Getting the sample reviews in Cluster 1

In [432]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(4184,size=3)
```

In [433]:
```
rand_num = list(rand_num)
```

In [434]:
```
rand_num
```

Out[434]:
```
[720, 324, 2822]
```

In [435]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_1[i])

for i in index:

    word_cloud.append(Y.values[i])
```

In [436]:
```
string_1 = " ".join(word_cloud)
```

In [437]:
```
string_1
```

Out[437]:
```
'love tast caramel crisp crunch thin cooki not fill no better altern ea
t day bite touch cranberri sooth flavor aroma appl reminisc cranberri c
ocktail serv cold great flavor ice tea hot flake cinnamon drift la la l
and consum near zero calori appl pie stuff good product excel dog chew
stick dog absolut love not swell throat like rawhid sometim price amazo
n much better dog show pet shop discount catalogu still littl expens co
stco came within day order'
```

In [438]:
```
wordcloud_1 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 2

- Getting the sample reviews in Cluster 2

In [439]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(816,size=3)
rand_num = list(rand_num)
```

In [440]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_2[i])

for i in index:

    word_cloud.append(Y.values[i])
```

In [441]:
```
string_2 = " ".join(word_cloud)
```

In [442]:
```
string_2
```

Out[442]:
```
'receiv jar instant espresso gift mother law quit insult first never us
e instant coffe kind could not figur gave past weekend coffe bean must
coffe morn made wow awesom fact drink mug right right review love howev
make latt fashion also ad hot chocol mocha latt definit look neighborho
od store not find get amazon definit great buy kcup strong enough tast
largest set say want someth coffe water reach dark magic cheer senseo m
achin frank coffe lover not live starbuck need one pod use rich flavour
design work machin'
```

In [443]:
```
wordcloud_2 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Plotting The Wordcloud

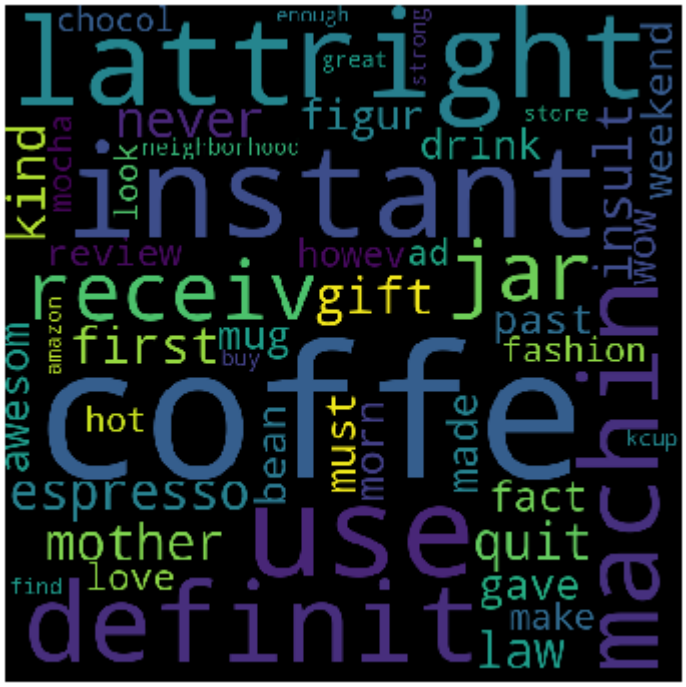
- **Cluster 1**

In [444]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_1)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Observation:

- This cluster says about how the product tastes and quality
- **Cluster 2**

In [445]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_2)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- This cluster says about how the product tastes and quality

Performance Metric of Agglomerative using TFIDF W2V

In [446]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_me

from sklearn.metrics import silhouette_score
```

In [447]:
```
score = silhouette_score(tfidf_w2v_train1,labels)
```

In [448]:
```
score
```

Out[448]:
```
0.401189873208517
```

Observation:

- As per the silhouette score document if the score is nearest to the Zero. The Clusters are Overlapped. So here The Silhouette Score is 0.40. So here the chance of clusters

overlapping is high.

6. DBSCAN Clustering

6.1 DBSCAN using Avg W2V

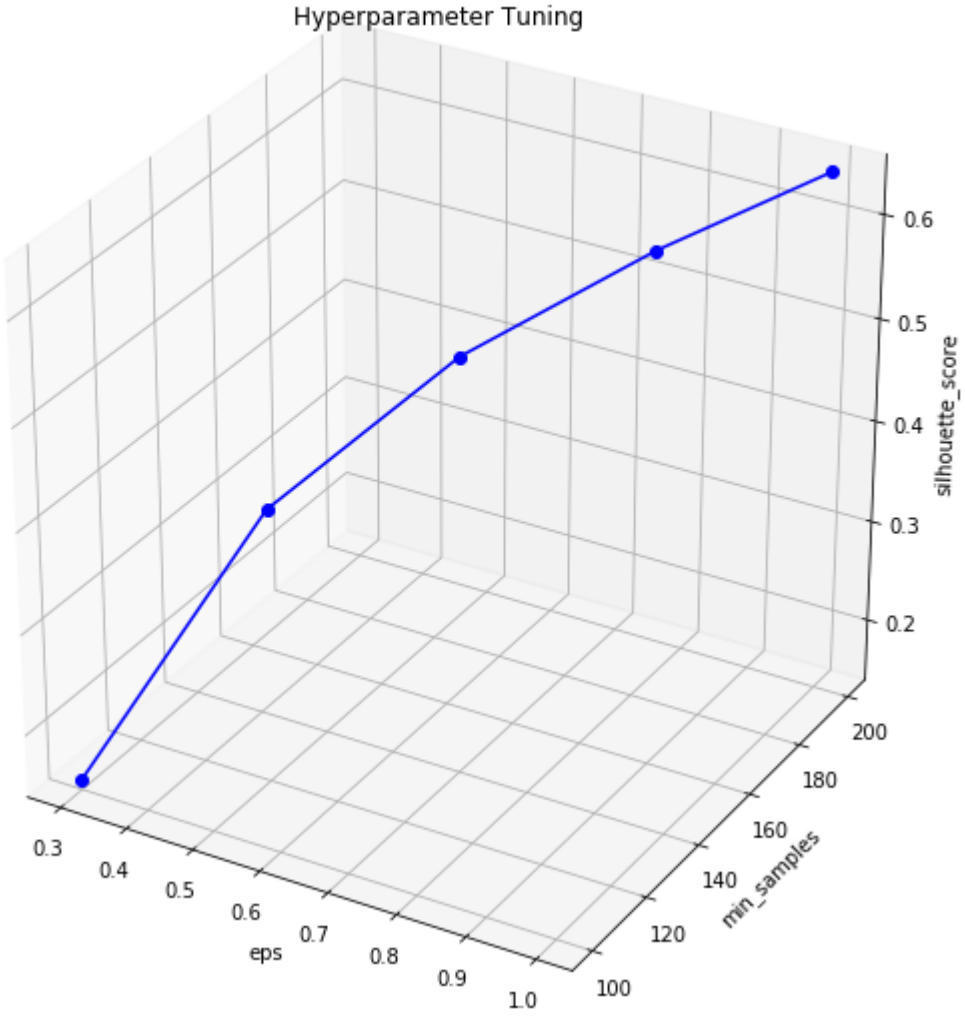
In [450]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSC

from sklearn.cluster import DBSCAN
```

In [473]:
```
# https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html

# Hyperparameter tuning

e=[0.3,0.5,0.7,0.9,1.0]
min_sample=[100,120,140,160,200]
silhouette_score_value=[]

for i,j in tqdm(zip(e,min_sample)):

    model= DBSCAN(eps=i,min_samples=j)
    model.fit(avg_w2v_train1)
    labels=model.labels_
    score=silhouette_score(avg_w2v_train1,labels)
    silhouette_score_value.append(score)
```

```
0it [00:00, ?it/s]
1it [00:02,  2.89s/it]
2it [00:06,  3.16s/it]
3it [00:11,  3.60s/it]
4it [00:16,  4.04s/it]
5it [00:22,  4.66s/it]
```

In [475]:
```
# References
# https://pythonprogramming.net/matplotlib-3d-scatterplot-tutorial/

from mpl_toolkits.mplot3d import Axes3D
```

In [477]:
```
# Hyperparameter Tuning

plt.close()
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
ax.plot(e,min_sample,silhouette_score_value,c='b', marker='o')
ax.set_xlabel('eps')
ax.set_ylabel('min_samples')
ax.set_zlabel('silhouette_score')
plt.title("Hyperparameter Tuning")
plt.show()
```



Observation:

- By using the silhouette_score value the best eps=1.0 and best min_samples=200

In [478]:
```
# Applying Best Hyperparameter

model= DBSCAN(eps=1.0,min_samples=200)
model.fit(avg_w2v_train1)
labels=model.labels_
```

In [479]:
```
set(labels)
```

Out[479]:
```
{-1, 0}
```

Observation:

- There is only two sets one is one cluster and another one is noise points.

Number Datapoints in Each Cluster

In [480]:
```
# Data points seperation as per the clusters

number_points = labels.shape[0]
print("Number of Datapoints")
print(number_points)
```

```
Number of Datapoints
5000
```

In [481]:
```
# Datapoints divided by clusters as per the label name

cluster_1=[]
cluster_2=[]

for i in range(0,number_points):

    if labels[i] == 0:
        cluster_1.append(i)
    if labels[i] == -1:
        cluster_2.append(i)
```

In [482]:
```
# The number of datapoints in each cluster

d=PrettyTable()

d.field_names = ["Cluster", "Number of Data Points"]

print(" The number of datapoints in each cluster")
print("="*120)

d.add_row([1,str(len(cluster_1))])
d.add_row([-1,str(len(cluster_2))])
print(d)
```

```
 The number of datapoints in each cluster
========================================================================
================================================
+---------+----------------------+
| Cluster | Number of Data Points |
+---------+----------------------+
|    1    |          4990        |
|    -1   |           10         |
+---------+----------------------+
```

Wordcloud for each cluster:

Cluster 1

- Getting the sample reviews in Cluster 1

In [483]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand

# randomly generated index values

rand_num = np.random.randint(4990,size=3)
```

In [484]:
```
rand_num = list(rand_num)
```

In [485]:
```
rand_num
```

Out[485]: [224, 135, 750]

In [486]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_1[i])

for i in index:

    word_cloud.append(Y.values[i])
```

In [487]:
```
string_1 = " ".join(word_cloud)
```

In [488]:
```
string_1
```

Out[488]:
```
'sportea answer healthi daili drink thirst quench great tast drink tea
drink day not get advers effect get soda regular caffin load ice tea ze
ro calori littl caffin zero calori lot electrolyt vitamin c zero calori
sweet refresh hint citrus flavor zero calori drink gallon day big fan w
ish would bottl sportea junki tip take larg cup like big gulp cup put s
portea ice tea bag bottom cover ice add water top let sportea steep min
readi guilt free sip day yum tapatio salsa picant one favorit hot sauc
flavor smoki complex tabasco popular hot sauc although goe perfect almo
st kind mexican southwestern food also great compliment varieti dish in
clud spaghetti soup addit sodium content relat low per teaspoon mani co
mpar brand make excel low sodium way add flavor food high recommend smo
ke bomb spring load spike trap gave mole invas call extermin use trap n
ot miss caught six two day found trap line order two month later saw an
oth mole tunnel next day care set trap power way extermin get mole'
```

In [489]:
```
wordcloud_1 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 2

- Getting the sample reviews in Cluster 2

In [490]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(10,size=3)
rand_num = list(rand_num)
```

In [491]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_2[i])

for i in index:

    word_cloud.append(Y.values[i])
```

In [492]:
```
string_2 = " ".join(word_cloud)
```

In [493]:
```
string_2
```

Out[493]:
```
'stuff hot great flavor hot sauc collector differ hot sauc absolut favo
rit stuff hot great flavor hot sauc collector differ hot sauc absolut f
avorit buy local store great product'
```

In [494]:
```
wordcloud_2 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Plotting The Wordcloud

- **Cluster 1**

In [495]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_1)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Observation:

- This cluster says about how the product tastes and quality
- **Cluster 2**

In [496]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_2)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

Observation:

- These are all noisy points of the cluster.

Performance Metric of DBSCAN using Avg W2V

In [497]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_me

from sklearn.metrics import silhouette_score
```

In [498]:
```
score = silhouette_score(avg_w2v_train1,labels)
```

In [499]:
```
score
```

Out[499]:
```
0.6463535708832251
```

Observation:

- As per the silhouette score document if the score is nearest to One. The Clusters are less Overlapped. So here The Silhouette Score is 0.64.

6.2 DBSCAN using TFIDF W2V

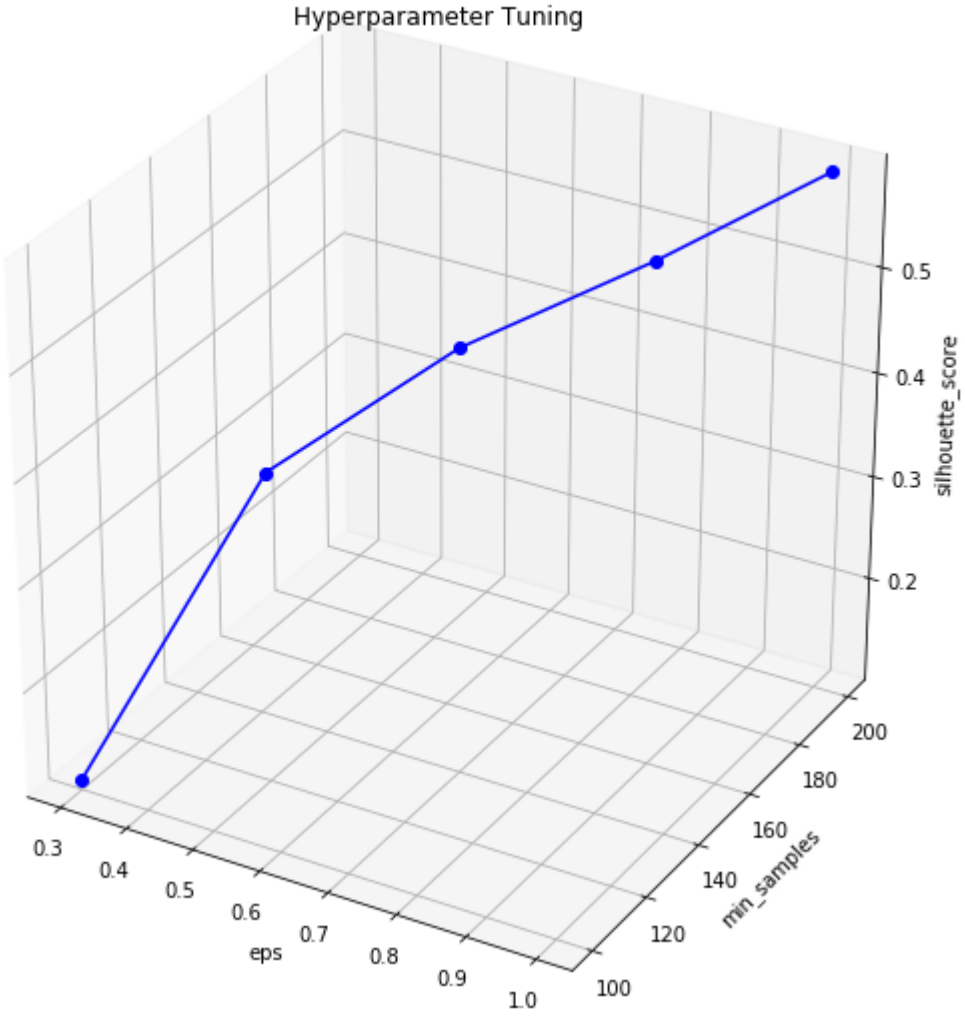
In [500]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSC

from sklearn.cluster import DBSCAN
```

In [503]:
```
# https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html

# Hyperparameter tuning

e=[0.3,0.5,0.7,0.9,1.0]
min_sample=[100,120,140,160,200]
silhouette_score_value=[]

for i,j in tqdm(zip(e,min_sample)):

    model= DBSCAN(eps=i,min_samples=j)
    model.fit(tfidf_w2v_train1)
    labels=model.labels_
    score=silhouette_score(tfidf_w2v_train1,labels)
    silhouette_score_value.append(score)
```

```
0it [00:00, ?it/s]
1it [00:02,  2.52s/it]
2it [00:05,  2.77s/it]
3it [00:10,  3.19s/it]
4it [00:14,  3.59s/it]
5it [00:19,  3.93s/it]
```

In [504]:
```
# References
# https://pythonprogramming.net/matplotlib-3d-scatterplot-tutorial/

from mpl_toolkits.mplot3d import Axes3D
```

In [505]:
```
# Hyperparameter Tuning

plt.close()
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
ax.plot(e,min_sample,silhouette_score_value,c='b', marker='o')
ax.set_xlabel('eps')
ax.set_ylabel('min_samples')
ax.set_zlabel('silhouette_score')
plt.title("Hyperparameter Tuning")
plt.show()
```



Observation:

- By using the silhouette_score value the best eps=1.0 and best min_samples=200

In [506]:
```
# Applying Best Hyperparameter

model= DBSCAN(eps=1.0,min_samples=200)
model.fit(tfidf_w2v_train1)
labels=model.labels_
```

In [507]:
```
set(labels)
```

Out[507]:
```
{-1, 0}
```

Observation:

- There is only two sets one is one cluster and another one is noise points.

Number Datapoints in Each Cluster

In [508]:
```
# Data points seperation as per the clusters

number_points = labels.shape[0]
print("Number of Datapoints")
print(number_points)
```

```
Number of Datapoints
5000
```

In [509]:
```
# Datapoints divided by clusters as per the label name

cluster_1=[]
cluster_2=[]

for i in range(0,number_points):

    if labels[i] == 0:
        cluster_1.append(i)
    if labels[i] == -1:
        cluster_2.append(i)
```

In [510]:
```
# The number of datapoints in each cluster

d=PrettyTable()

d.field_names = ["Cluster", "Number of Data Points"]

print(" The number of datapoints in each cluster")
print("="*120)

d.add_row([1,str(len(cluster_1))])
d.add_row([-1,str(len(cluster_2))])
print(d)
```

```
 The number of datapoints in each cluster
========================================================================
================================================
+---------+----------------------+
| Cluster | Number of Data Points |
+---------+----------------------+
|    1    |          4934        |
|    -1   |           66         |
+---------+----------------------+
```

Wordcloud for each cluster:

Cluster 1

- Getting the sample reviews in Cluster 1

In [511]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand

# randomly generated index values

rand_num = np.random.randint(4934,size=3)
```

In [512]:
```
rand_num = list(rand_num)
```

In [513]:
```
rand_num
```

Out[513]:
```
[17, 2574, 1965]
```

In [514]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_1[i])

for i in index:

    word_cloud.append(Y.values[i])
```

In [515]:
```
string_1 = " ".join(word_cloud)
```

In [516]:
```
string_1
```

Out[516]:
```
'beetlejuic awe inspir wonder amus comed romp explor incred possibl lif
e boundari absurd tell tale recent dead marri coupl sudden get led chao
tic world supernatur adam barbara maitland alec baldwin geena davi disc
ov mani conflict rather human imperfect haunt live live also plagu afte
rlif well unlik film project seem blind assign dispassion filmmak comme
rci reason beetlejuic plot bizarr subject matter remark complement burt
on unusu macabr artist sensibl extraordinarili well creat unbeliev bril
liant guidanc imagin film director tim burton pee wee big adventur batm
an ed wood sleepi hollow film uniqu creativ landscap culmin essenti abu
nd ironi outlandish yet human behavior grace bodi burton work augment d
evious energet perform glenn shadix jeffrey jone winona ryder catherin
hara geena davi alec baldwin film bustl uninhibit brilliant hilar persi
st push film level almost affabl euphoria pair ingeni screenplay tour d
e forc perform michael keaton beetlejuic film transform exuber jovial e
xercis extrem satisfi philosoph percept mani level though comedi usual
consid unabl undeserv deep critic analysi beetlejuic undeni inspir conc
ept flawless transfer film one outstand comedi film dvd packag dvd incl
ud theatric trailer isol danni elfman music track choic watch film anam
orph widescreen pan scan hope beetlejuic eventu grace special edit alwa
y hope devic water plant turn light base set add nutrient plan fastest
grow plant come seed kit could not tell one includ probabl salad fastes
t grow week gormet herb popular spec take standard plug instead prong p
lug anywher instead kitchen bowl hold cup water one pint shi gallon lig
ht goe hour hour depend plant water goe root spong light time longer th
irsti plant hand polin rapid second wave plant tomato pepper everi day
flower pepper intern basil requir wash bowl everi week otherwis littl m
ainten light come water low anoth light come week plant need nutrient p
retti much forget light also work help pot plant grow near plant like h
uman comfort temperatur f pepper tomato produc fruit safe nutrient diff
er plant breakdown dinubabear site come set kit start sprout normal fru
it give nutrient tablet everi week aerogrow guarente kit sprout describ
call replac kit new kit avail septemb french italian japanes herb good
lot cheaper chocol not lot better'
```

In [517]:
```
wordcloud_1 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Cluster 2

- Getting the sample reviews in Cluster 2

In [518]:
```
# References
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.rand


# randomly generated index values

rand_num = np.random.randint(66,size=3)
rand_num = list(rand_num)
```

In [519]:
```
# Reviews in the cluster 1

index=[]
word_cloud=[]

for i in rand_num:

    index.append(cluster_2[i])

for i in index:

    word_cloud.append(Y.values[i])
```

In [520]:
```
string_2 = " ".join(word_cloud)
```

In [521]:
```
string_2
```

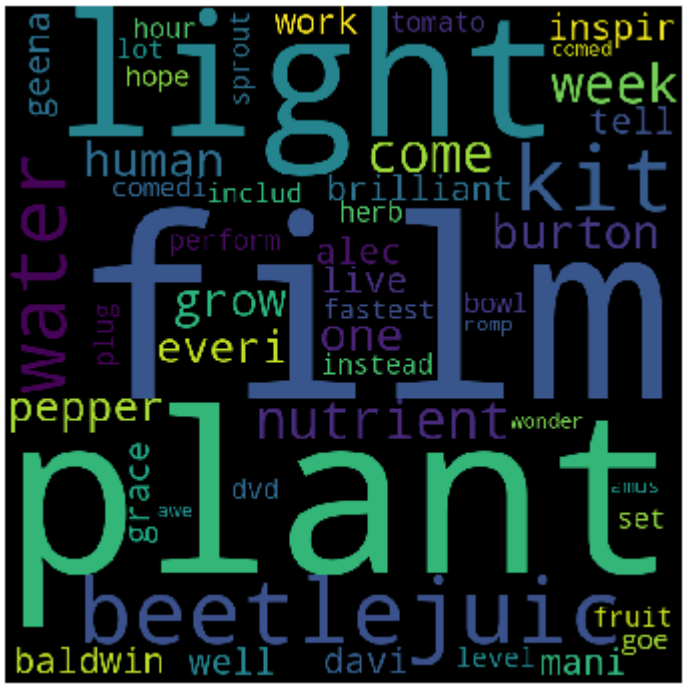
Out[521]:
```
'dog akita hate take pill not pill pocket dog pill pocket cat click no
button review help read comment valu dog take small pill andi take pill
hypothyroid past year pill tini wrap meat chees never work simpli ate a
round pill spit year way could get andi take pill liter shove throat ha
nd small enough mouth big enough thank trust dog one day vet notic pack
ag someth call pill pocket two kind one dog one cat came flavor andi wo
uld like dog pill pocket came differ size small larg cat pill pocket ca
me one size small look packag care notic packag cat pill pocket packag
small dog pill pocket yet packag small dog cost packag cat huh amazon c
om price quantiti differ look andi tini thyroid pill look pill pocket p
ictur packag decid might well save buck buy kitti pill pocket hey andi
not know eat someth intend cat happi decis ever bought salmon chicken f
lavor pill pocket cat andi like pocket perfect size big enough thyroid
pill date gulp everi one beg anoth like fact get packag would bought pi
ll pocket dog pill pocket great product cat dog wish discov sooner neve
r use pill pocket dog confid excel give product star andi pill bigger w
ould get pill pocket dog know would like much time though stick smaller
pill pocket cat last day longer review pass along idea perhap dog owner
not consid dog take tini pill like andi go econom choic get pill pocket
cat dog never know differ kona pod highest qualiti pod found yet love k
ona coffe extrem pleas find pod certain cost littl cheap pod worth espe
ci consid pay cup coffe retail best coffe world home less dark roast be
st coffe pod avail senseo coffe machin full bodi coffe coffe made sense
o coffe machin never burnt fantast'
```

In [522]:
```
wordcloud_2 = WordCloud(width=720, height=720, max_words=50).generate(st
```

Plotting The Wordcloud

- **Cluster 1**

In [523]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_1)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

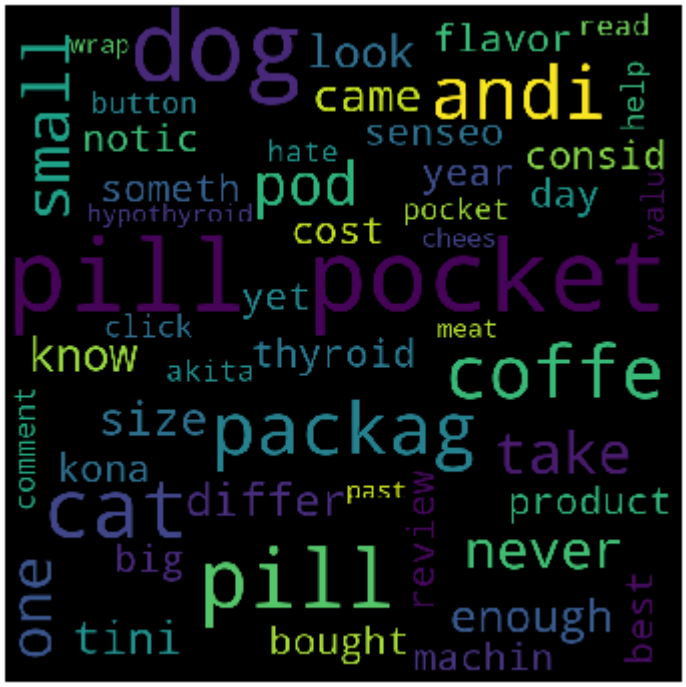


Observation:

- This cluster says about how the product tastes and quality

- **Cluster 2**

In [524]:
```
plt.close()
plt.figure(figsize = (5,5))
plt.imshow(wordcloud_2)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Observation:

- These are all noisy points of the cluster.

Performance Metric of DBSCAN using TFIDF W2V

In [525]:
```
# References
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_me

from sklearn.metrics import silhouette_score
```

In [526]:
```
score = silhouette_score(tfidf_w2v_train1,labels)
```

In [527]:
```
score
```

Out[527]:
```
0.5946827234675885
```

Observation:

- As per the silhouette score document if the score is nearest to One. The Clusters are less Overlapped. So here The Silhouette Score is 0.59.

7. Conclusion:

Performance of the model.

- Here we choose silhouette_score as the performance metric. As per the silhouette_score, Best value is 1. Worst value is -1. if the value is near to the zero the clusters are overlapped.

In [531]:
```
a=PrettyTable()

a.field_names = ["Model", "Vectorizer", "Silhouette_score"]

a.add_row(["K- Means","BoW",0.16])
a.add_row(["K- Means","TFIDF",0.02])
a.add_row(["K- Means","Avg W2V",0.08])
a.add_row(["K- Means","Tfidf W2V",0.12])
a.add_row(["Agglomerative","Avg W2V",0.24])
a.add_row(["Agglomerative","Tfidf W2V",0.40])
a.add_row(["DBSCAN","Avg W2V",0.64])
a.add_row(["DBSCAN","Tfidf W2V",0.59])

print(a)
```

```
+---------------+------------+------------------+
|     Model     | Vectorizer | Silhouette_score |
+---------------+------------+------------------+
|    K- Means   |    BoW     |       0.16       |
|    K- Means   |   TFIDF    |       0.02       |
|    K- Means   |  Avg W2V   |       0.08       |
|    K- Means   | Tfidf W2V  |       0.12       |
| Agglomerative |  Avg W2V   |       0.24       |
| Agglomerative | Tfidf W2V  |       0.4        |
|     DBSCAN    |  Avg W2V   |       0.64       |
|     DBSCAN    | Tfidf W2V  |       0.59       |
+---------------+------------+------------------+
```

1. K-Means Clustering:

Data Cleaning ,Preprocessing and splitting:

- In the Data Cleaning process, we clean the duplicate datapoints and unconditioning data points. After the data cleaning process we get 364171 data points and sort based on timestamp.
- Then select the Review Text Feature as a important feature, then do data preprocessing on all the data points.
- Then select top 50k sample data points for further process.

Featurization:

- Then apply the data points on BOW,TFIDF,Avg W2V and TFIDF W2V for converting text to vector.

K-means model:

- Then apply these featurization vector on K - means model.Best number of clusters are find out by using elbow method.

Wordcloud:

- Then plot the wordcloud for each cluster of the model.

2. Agglomerative Clustering:

Data Cleaning ,Preprocessing and splitting:

- In the Data Cleaning process, we clean the duplicate datapoints and unconditioning data points. After the data cleaning process we get 364171 data points and sort based on timestamp.
- Then select the Review Text Feature as a important feature, then do data preprocessing on all the data points.
- Then select top 5k sample data points for further process.

Featurization:

- Then apply the data points on Avg W2V and TFIDF W2V for converting text to vector.

Agglomerative model:

- Then apply these featurization vector on Agglomerative model.Best number of clusters are find out by using elbow method.

Wordcloud:

- Then plot the wordcloud for each cluster of the model.

3. DBSCAN Clustering:

Data Cleaning ,Preprocessing and splitting:

- In the Data Cleaning process, we clean the duplicate datapoints and unconditioning data points. After the data cleaning process we get 364171 data points and sort based on timestamp.
- Then select the Review Text Feature as a important feature, then do data preprocessing on all the data points.
- Then select top 5k sample data points for further process.

Featurization:

- Then apply the data points on Avg W2V and TFIDF W2V for converting text to vector.

Agglomerative model:

- Then apply these featurization vector on DBSCAN model.Best Hyperparameters (eps, min_samples) are find out by using elbow method.

Wordcloud:

- Then plot the wordcloud for each cluster of the model.