In [ ]:
```
import pandas as pd
import plotly.express as px
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from matplotlib import colors
import seaborn as sns
```

In [ ]:
```
data=pd.read_csv('mcdonalds.csv')
data.head()
```

Out[ ]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgustir
0	No	Yes	No	Yes	No	Yes	Yes	No	Yes	No	N
1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	N
2	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	N
3	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No	Y
4	No	Yes	No	Yes	Yes	Yes	Yes	No	No	Yes	N

In [ ]:
```
data.columns
```

Out[ ]:
```
Index(['yummy', 'convenient', 'spicy', 'fattening', 'greasy', 'fast', 'cheap',
       'tasty', 'expensive', 'healthy', 'disgusting', 'Like', 'Age',
       'VisitFrequency', 'Gender'],
      dtype='object')
```

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1453 entries, 0 to 1452
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   yummy           1453 non-null   object
 1   convenient      1453 non-null   object
 2   spicy           1453 non-null   object
 3   fattening       1453 non-null   object
 4   greasy          1453 non-null   object
 5   fast            1453 non-null   object
 6   cheap           1453 non-null   object
 7   tasty           1453 non-null   object
 8   expensive       1453 non-null   object
 9   healthy         1453 non-null   object
 10  disgusting      1453 non-null   object
 11  Like            1453 non-null   object
 12  Age             1453 non-null   int64 
 13  VisitFrequency  1453 non-null   object
 14  Gender          1453 non-null   object
dtypes: int64(1), object(14)
memory usage: 170.4+ KB
```

In [ ]:
```
df=data.drop(data.columns[[11,12,13,14]],axis=1)
df.head()
```

Out[ ]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgustir
0	No	Yes	No	Yes	No	Yes	Yes	No	Yes	No	N
1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	N
2	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	N
3	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No	Y
4	No	Yes	No	Yes	Yes	Yes	Yes	No	No	Yes	N

In [ ]:
```
data=data[data.columns].replace({'Yes':1, 'No':0,'I hate it!-5':'-5','I love it!+5
data['Like']=pd.to_numeric(data['Like'])
```

In [ ]:
```
df=df[df.columns].replace({'Yes':1, 'No':0})
print(df.head())
round(df.mean(axis=0),2)
```

```
   yummy  convenient  spicy  fattening  greasy  fast  cheap  tasty  expensive  \
0      0           1      0          1       0     1      1      0          1   
1      1           1      0          1       1     1      1      1          1   
2      0           1      1          1       1     1      0      1          1   
3      1           1      0          1       1     1      1      1          0   
4      0           1      0          1       1     1      1      0          0   

   healthy  disgusting
0        0           0
1        0           0
2        1           0
3        0           1
4        1           0
```

Out[ ]:
```
yummy          0.55
convenient     0.91
spicy          0.09
fattening      0.87
greasy         0.53
fast           0.90
cheap          0.60
tasty          0.64
expensive      0.36
healthy        0.20
disgusting     0.24
dtype: float64
```

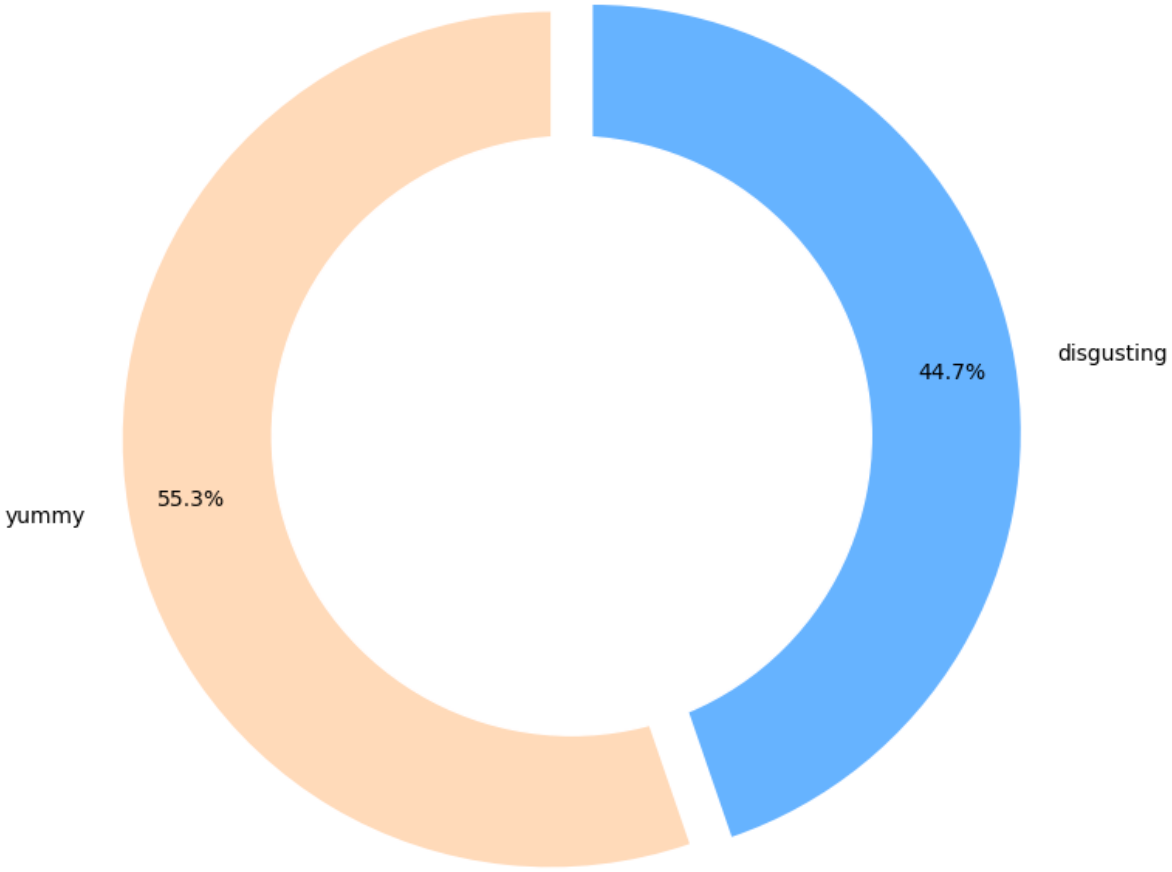
In [ ]:
```
pca_df=PCA(n_components=11)
principalComp=pca_df.fit_transform(df)
pcadf=pd.DataFrame(principalComp,columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7
pcadf.head()
```

Out[ ]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
0	0.425367	-0.219079	0.663255	-0.401300	0.201705	-0.389767	-0.211982	0.163235	0.181007
1	-0.218638	0.388190	-0.730827	-0.094724	0.044669	-0.086596	-0.095877	-0.034756	0.111476
2	0.375415	0.730435	-0.122040	0.692262	0.839643	-0.687406	0.583112	0.364379	-0.322288
3	-0.172926	-0.352752	-0.843795	0.206998	-0.681415	-0.036133	-0.054284	-0.231477	-0.028003
4	0.187057	-0.807610	0.028537	0.548332	0.854074	-0.097305	-0.457043	0.171758	-0.074409

In [ ]:
```
features = df.columns.values.tolist()
X = df[features]

pca = PCA(n_components=11)
components = pca.fit_transform(X)

loadings = pca.components_.T * np.sqrt(pca.explained_variance_)

fig = px.scatter(components, x=0, y=1)

for i, feature in enumerate(features):
    fig.add_annotation(
        ax=0, ay=0,
        axref="x", ayref="y",
        x=loadings[i, 0],
        y=loadings[i, 1],
        showarrow=True,
        arrowsize=2,
        arrowhead=2,
        xanchor="right",
        yanchor="top"
    )
    fig.add_annotation(
        x=loadings[i, 0],
        y=loadings[i, 1],
        ax=0, ay=0,
        xanchor="center",
        yanchor="bottom",
        text=feature,
        yshift=5,
    )
fig.show()
```

In [ ]:
```
labels = ['yummy','disgusting']
sizes = [df.query('yummy == 1').yummy.count(),df.query('yummy == 0').yummy.count()
#colors
colors = ['#ffdaB9','#66b3ff']
#explsion
explode = (0.05,0.05)
plt.figure(figsize=(8,8))
my_circle=plt.Circle( (0,0), 0.7, color='white')
plt.pie(sizes, colors = colors, labels=labels, autopct='%1.1f%%', startangle=90, p
p=plt.gcf()
plt.axis('equal')
p.gca().add_artist(my_circle)
plt.show()
```

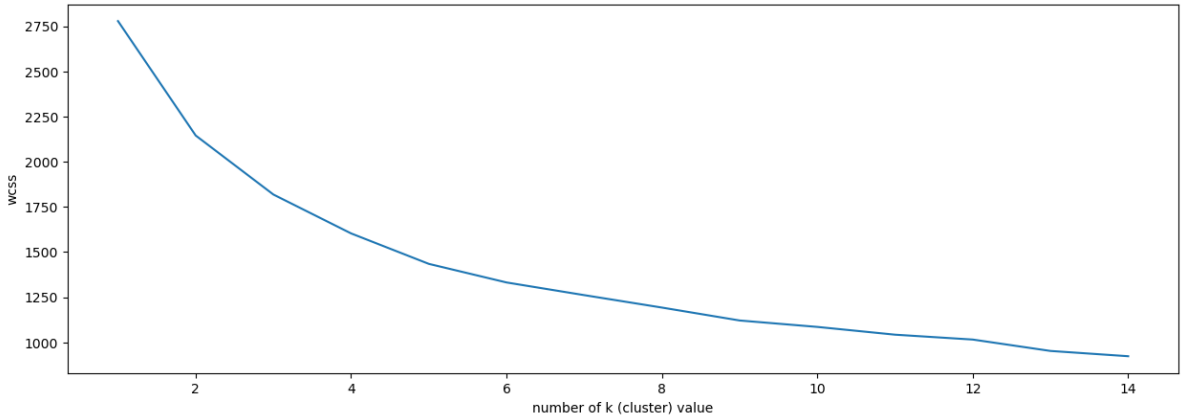


K-Means Clustering.

In [ ]:
```
wcss = []
for k in range(1,15):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(df)
    wcss.append(kmeans.inertia_)

#create model
kmeans = KMeans(n_clusters=5)
data_predict = kmeans.fit_predict(df)

# the best value is elbow value. It's 5.
plt.figure(figsize=(15,5))
plt.plot(range(1,15),wcss)
plt.xlabel("number of k (cluster) value")
```

```
plt.ylabel("wcss")
plt.show()
```

```
d:\CODING\anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chu
nks than available threads. You can avoid it by setting the environment variable O
MP_NUM_THREADS=6.
```

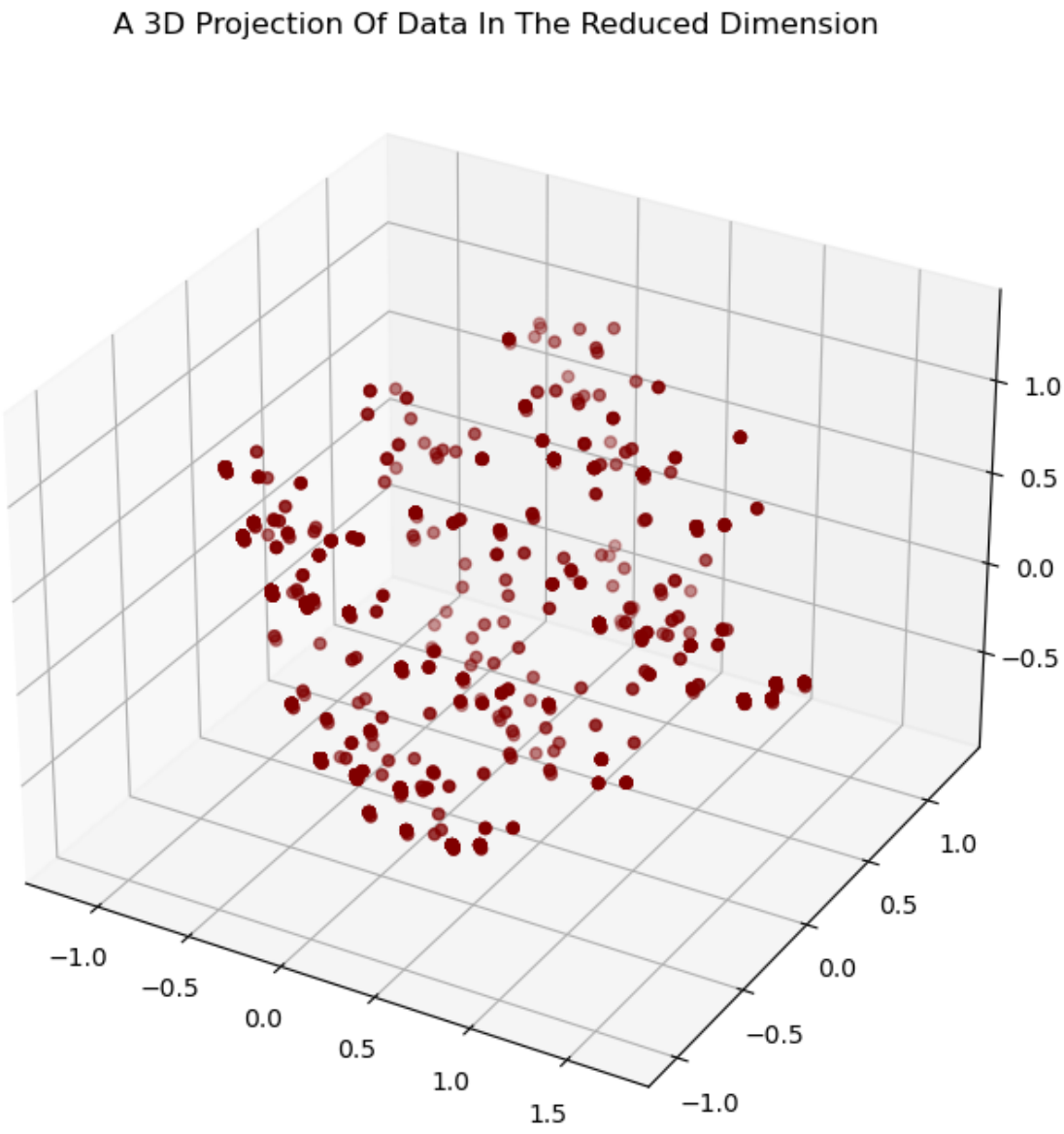


In [ ]:
```
pca = PCA(n_components=3)
pca.fit(df)
PCA_ds = pd.DataFrame(pca.transform(df), columns=(["col1","col2", "col3"]))
PCA_ds.describe().T
```

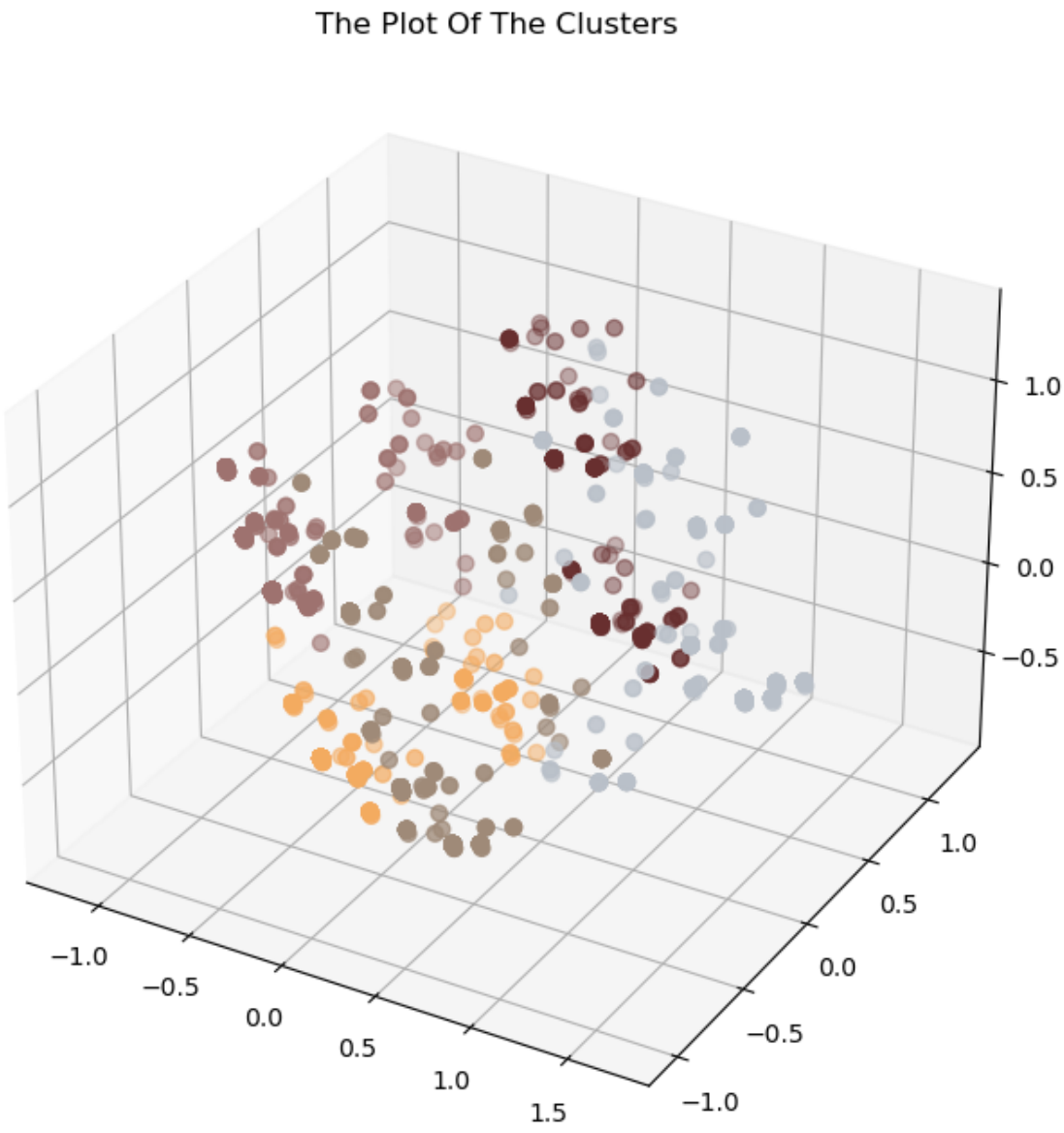
Out[ ]:

	count	mean	std	min	25%	50%	75%	max
col1	1453.0	-3.973269e-18	0.757050	-1.188421	-0.547679	-0.070746	0.493301	1.705573
col2	1453.0	-8.248353e-17	0.607456	-1.040274	-0.356848	-0.115163	0.391554	1.279130
col3	1453.0	-4.375491e-17	0.504619	-0.880813	-0.450779	0.026877	0.351594	1.326976

In [ ]:
```
#A 3D Projection Of Data In The Reduced Dimension
x =PCA_ds["col1"]
y =PCA_ds["col2"]
z =PCA_ds["col3"]
#To plot
fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111, projection="3d")
ax.scatter(x,y,z, c="maroon", marker="o" )
ax.set_title("A 3D Projection Of Data In The Reduced Dimension")
plt.show()
```

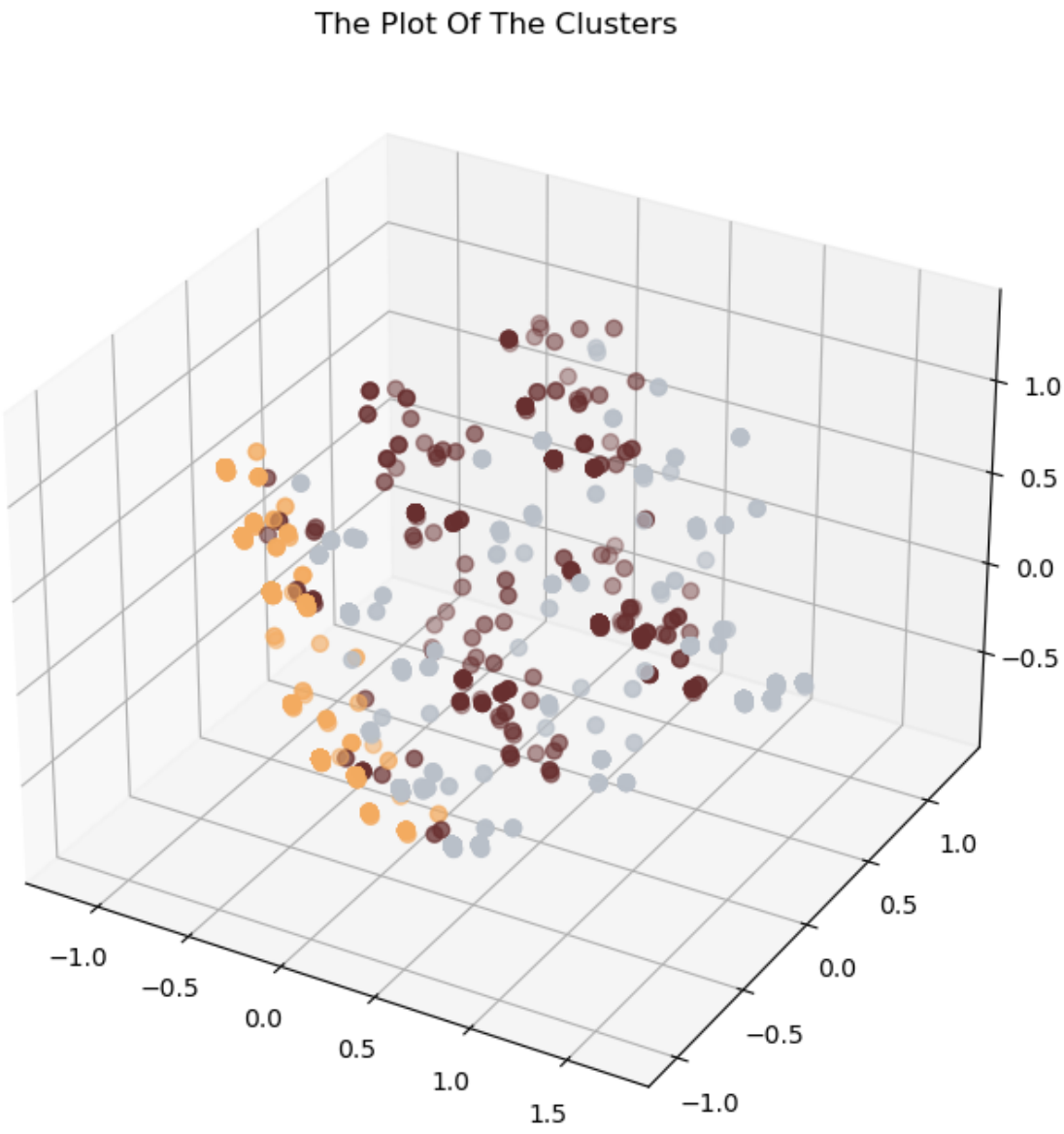


In [ ]:
```
#Plotting the clusters
PCA_ds['Clusters']=data_predict
df['Clusters']=data_predict
cmap = colors.ListedColormap(["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["Clusters"], marker='o', cmap = cmap )
ax.set_title("The Plot Of The Clusters")
plt.show()
```

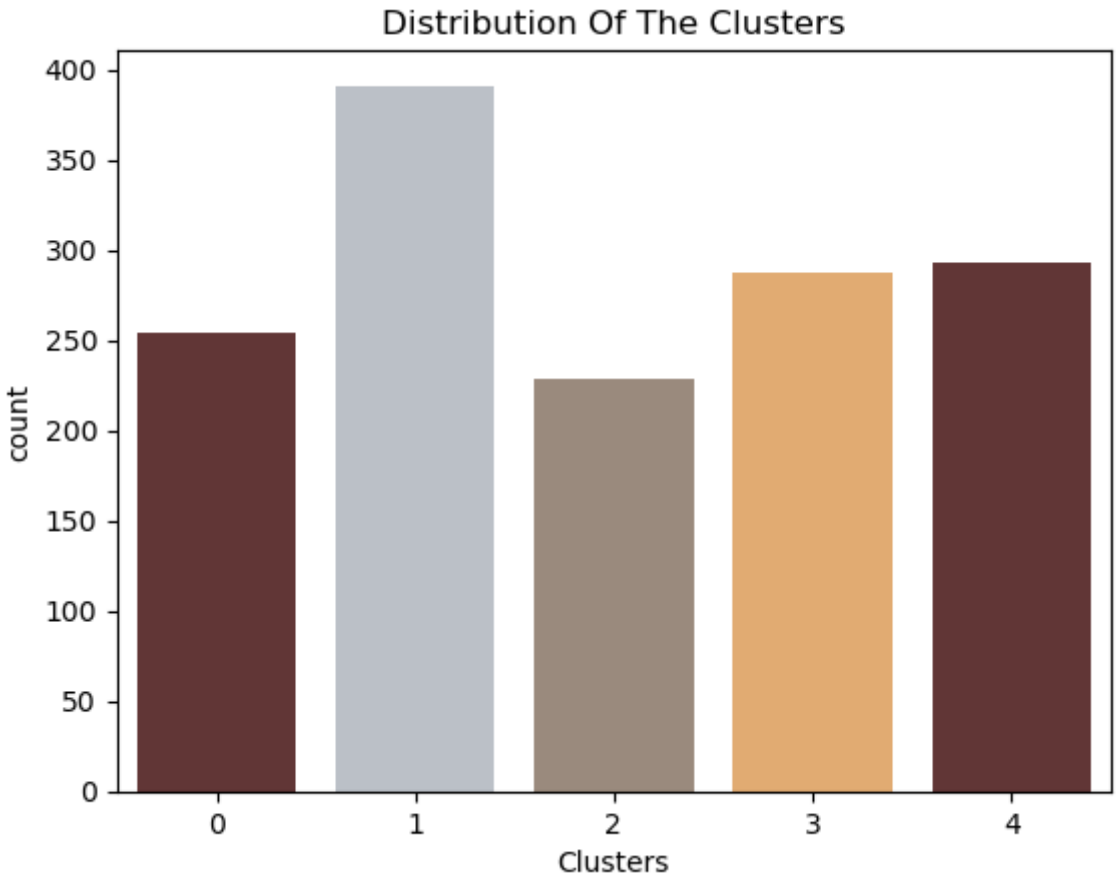


Using Gaussian Mixture Model.

In [ ]:
```
X=np.array(X)
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3).fit(X)
labels = gmm.predict(X)
PCA_ds['Clusters']=labels
cmap = colors.ListedColormap(["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["Clusters"], marker='o', cmap = cmap )
ax.set_title("The Plot Of The Clusters")
plt.show()
probs = gmm.predict_proba(X)
```

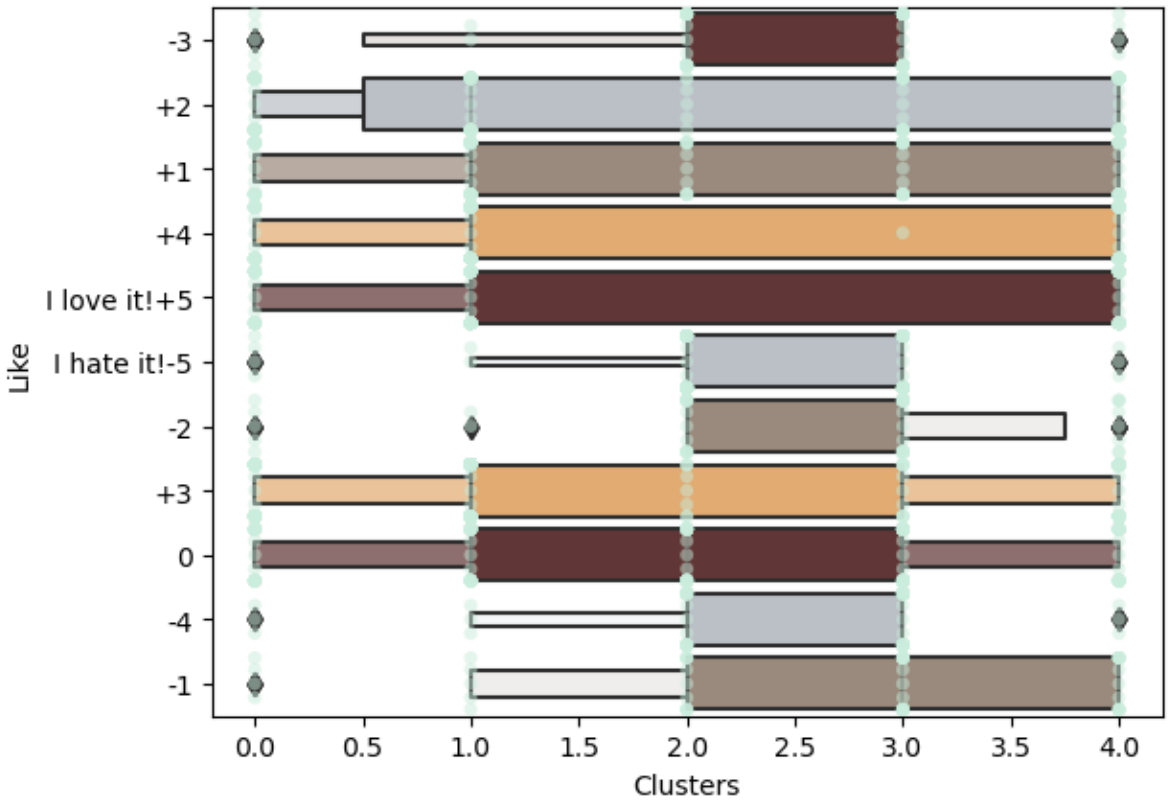


In [ ]:
```
#Plotting countplot of clusters
pal = ["#682F2F","#B9C0C9", "#9F8A78","#F3AB60"]
pl = sns.countplot(x=df["Clusters"], palette= pal)
pl.set_title("Distribution Of The Clusters")
plt.show()
```



Describing segments

In [ ]:
```
plt.figure()
pl=sns.swarmplot(x=df["Clusters"], y=data["Like"], color= "#CBEDDD", alpha=0.5 )
pl=sns.boxenplot(x=df["Clusters"], y=data["Like"], palette=pal)
plt.show()
```



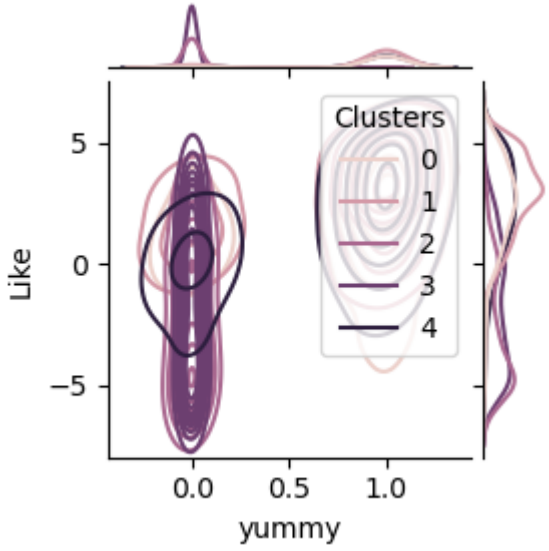
Profiling

In [ ]:
```
print(data['Like'])
data['Like'].info()
```

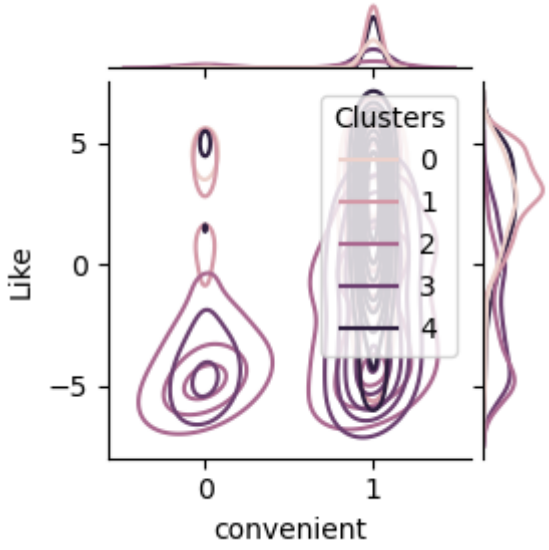
```
0      -3
1       2
2       1
3       4
4       2
       ..
1448   -5
1449    2
1450    3
1451    4
1452   -3
Name: Like, Length: 1453, dtype: int64
<class 'pandas.core.series.Series'>
RangeIndex: 1453 entries, 0 to 1452
Series name: Like
Non-Null Count  Dtype
--------------  -----
1453 non-null   int64
dtypes: int64(1)
memory usage: 11.5 KB
```

In [ ]:
```
Personal = df.columns.drop('Clusters').tolist()

for i in Personal:
    plt.figure(figsize=(5,5))
    sns.jointplot(x=df[i], y=data["Like"], hue =df["Clusters"], kind="kde",height=
    plt.show()
```

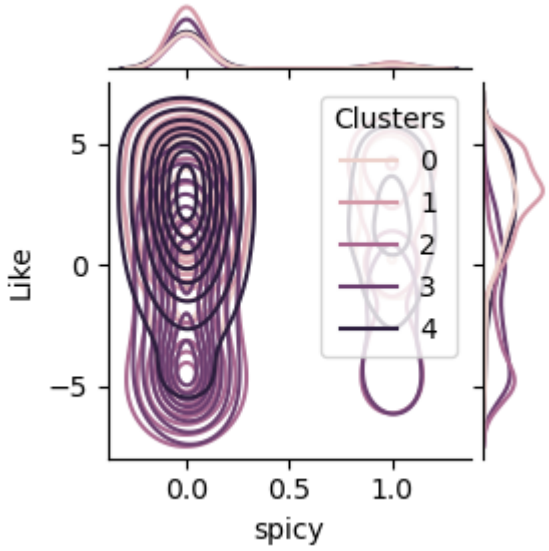
```
<Figure size 500x500 with 0 Axes>
```



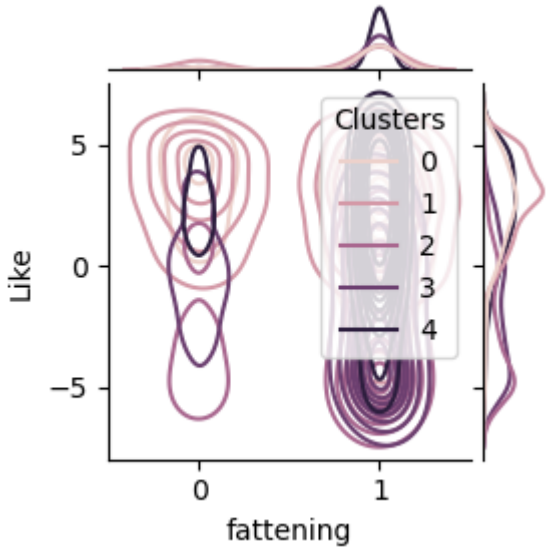

```
<Figure size 500x500 with 0 Axes>
```




```
<Figure size 500x500 with 0 Axes>
```

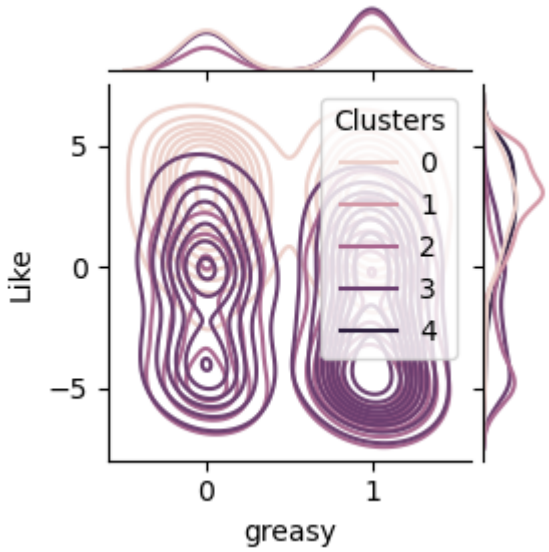



```
<Figure size 500x500 with 0 Axes>
```

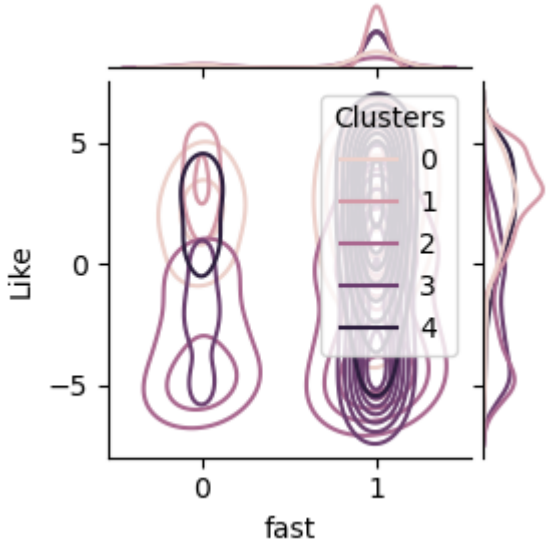



```
d:\CODING\anaconda\lib\site-packages\seaborn\distributions.py:316: UserWarning:

Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to d
isable this warning.

d:\CODING\anaconda\lib\site-packages\seaborn\distributions.py:316: UserWarning:

Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to d
isable this warning.
```

```
<Figure size 500x500 with 0 Axes>
```

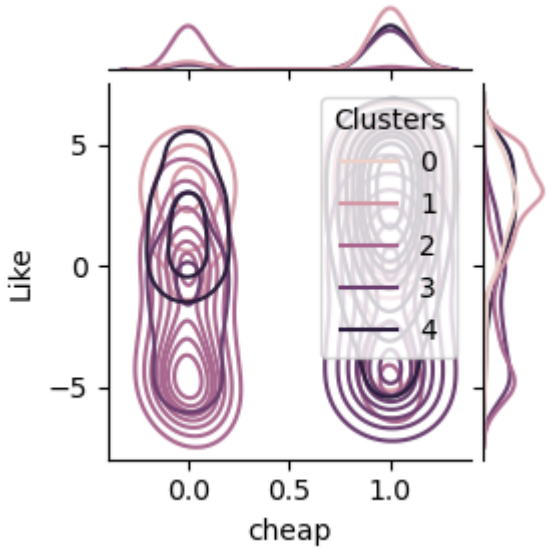



```
<Figure size 500x500 with 0 Axes>
```

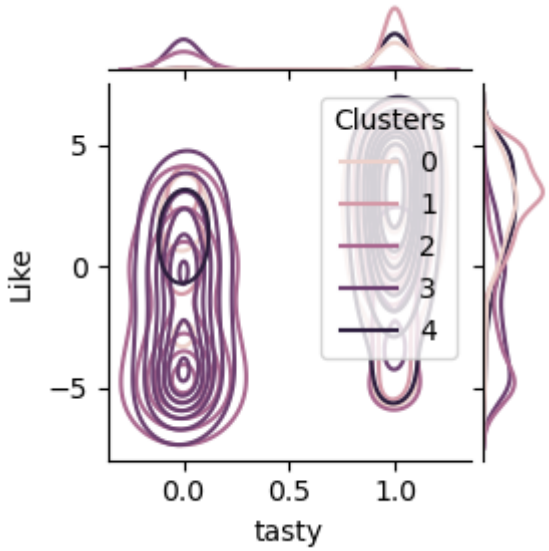



```
d:\CODING\anaconda\lib\site-packages\seaborn\distributions.py:316: UserWarning:

Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to d
isable this warning.
```

```
<Figure size 500x500 with 0 Axes>
```

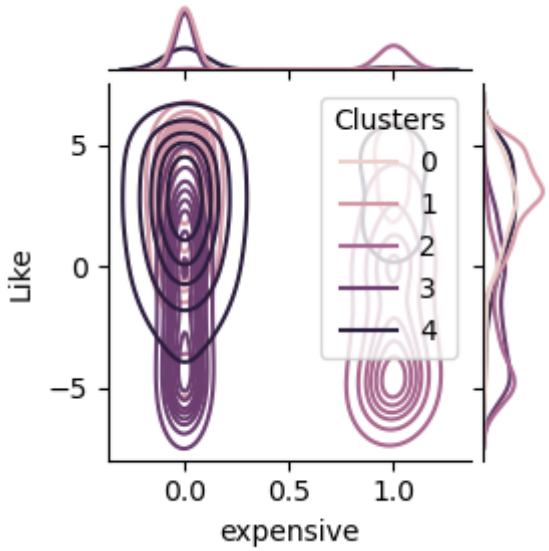



```
<Figure size 500x500 with 0 Axes>
```

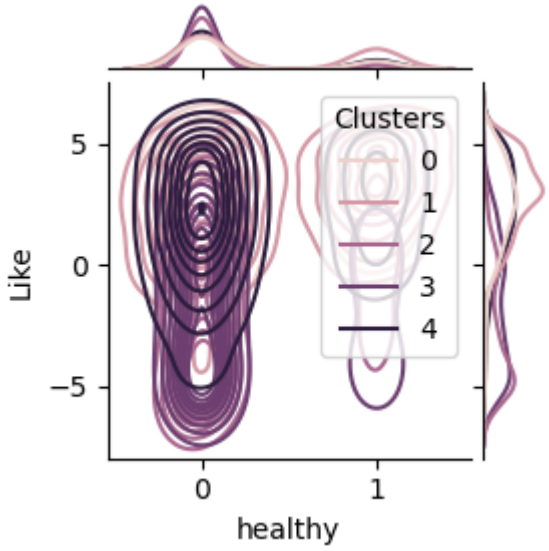


```
d:\CODING\anaconda\lib\site-packages\seaborn\distributions.py:316: UserWarning:

Dataset has 0 variance; skipping density estimate. Pass `warn_singular=False` to d
isable this warning.
```

```
<Figure size 500x500 with 0 Axes>
```




```
<Figure size 500x500 with 0 Axes>
```




```
<Figure size 500x500 with 0 Axes>
```