```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
mydata = pd.read_csv('google_review_ratings.csv')
```

```
mydata
```

	User	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10	Categor 1
0	User 1	0.00	0.00	3.63	3.65	5.00	2.92	5.00	2.35	2.33	2.64	1.
1	User 2	0.00	0.00	3.63	3.65	5.00	2.92	5.00	2.64	2.33	2.65	1.

```
mydata.shape #Check the shape of the data
```

```
(5456, 26)
```

```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5456 entries, 0 to 5455
Data columns (total 26 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   User         5456 non-null   object 
 1   Category 1   5456 non-null   float64
 2   Category 2   5456 non-null   float64
 3   Category 3   5456 non-null   float64
 4   Category 4   5456 non-null   float64
 5   Category 5   5456 non-null   float64
 6   Category 6   5456 non-null   float64
 7   Category 7   5456 non-null   float64
 8   Category 8   5456 non-null   float64
 9   Category 9   5456 non-null   float64
 10  Category 10  5456 non-null   float64
 11  Category 11  5456 non-null   object 
 12  Category 12  5455 non-null   float64
 13  Category 13  5456 non-null   float64
 14  Category 14  5456 non-null   float64
 15  Category 15  5456 non-null   float64
 16  Category 16  5456 non-null   float64
 17  Category 17  5456 non-null   float64
 18  Category 18  5456 non-null   float64
 19  Category 19  5456 non-null   float64
 20  Category 20  5456 non-null   float64
```

```
 21  Category 21  5456 non-null   float64
 22  Category 22  5456 non-null   float64
 23  Category 23  5456 non-null   float64
 24  Category 24  5455 non-null   float64
 25  Unnamed: 25  2 non-null      float64
dtypes: float64(24), object(2)
memory usage: 1.1+ MB
```

```
mydata.isnull().sum()
```

```
User             0
Category 1       0
Category 2       0
Category 3       0
Category 4       0
Category 5       0
Category 6       0
Category 7       0
Category 8       0
Category 9       0
Category 10      0
Category 11      0
Category 12      1
Category 13      0
Category 14      0
Category 15      0
Category 16      0
Category 17      0
Category 18      0
Category 19      0
Category 20      0
Category 21      0
Category 22      0
Category 23      0
Category 24      1
Unnamed: 25    5454
dtype: int64
```

```
mydata.drop('Unnamed: 25',axis=1,inplace=True) #dropping the unwanted column
```

```
mydata.drop('User',axis=1,inplace=True) #dropping the unwanted column
```

```
mydata.drop('User',axis=1,inplace=True) #dropping the unwanted column
```

```
mydata.columns
```

```
Index(['Category 1', 'Category 2', 'Category 3', 'Category 4', 'Category 5',
       'Category 6', 'Category 7', 'Category 8', 'Category 9', 'Category 10',
       'Category 11', 'Category 12', 'Category 13', 'Category 14',
       'Category 15', 'Category 16', 'Category 17', 'Category 18',
       'Category 19', 'Category 20', 'Category 21', 'Category 22',
       'Category 23', 'Category 24'],
      dtype='object')
```

Convert the datatype having object to float

```
mydata['Category 11'] = pd.to_numeric(mydata['Category 11'],errors = 'coerce')
```

```
mydata['Category 11']
```

```
0       1.70
1       1.70
2       1.70
3       1.73
4       1.70
        ...
5451    1.02
5452    1.01
5453    0.99
5454    0.97
5455    0.95
Name: Category 11, Length: 5456, dtype: float64
```

```
#Dropping the null values from the data table
```

```
mydata.dropna(axis=0,inplace=True)
```

```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5454 entries, 0 to 5455
Data columns (total 24 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Category 1   5454 non-null   float64
 1   Category 2   5454 non-null   float64
 2   Category 3   5454 non-null   float64
 3   Category 4   5454 non-null   float64
 4   Category 5   5454 non-null   float64
 5   Category 6   5454 non-null   float64
 6   Category 7   5454 non-null   float64
 7   Category 8   5454 non-null   float64
 8   Category 9   5454 non-null   float64
 9   Category 10  5454 non-null   float64
 10  Category 11  5454 non-null   float64
 11  Category 12  5454 non-null   float64
 12  Category 13  5454 non-null   float64
 13  Category 14  5454 non-null   float64
 14  Category 15  5454 non-null   float64
 15  Category 16  5454 non-null   float64
 16  Category 17  5454 non-null   float64
 17  Category 18  5454 non-null   float64
 18  Category 19  5454 non-null   float64
 19  Category 20  5454 non-null   float64
 20  Category 21  5454 non-null   float64
 21  Category 22  5454 non-null   float64
 22  Category 23  5454 non-null   float64
 23  Category 24  5454 non-null   float64
dtypes: float64(24)
memory usage: 1.0 MB
```

```
mydata.describe()
```

	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category
count	5454.000000	5454.000000	5454.000000	5454.000000	5454.000000	5454.000000	5454.00000
mean	1.455746	2.320048	2.489059	2.797103	2.958904	2.893423	3.35147
std	0.827732	1.421576	1.247503	1.309188	1.338785	1.282101	1.41329
min	0.000000	0.000000	0.000000	0.830000	1.120000	1.110000	1.12000
25%	0.920000	1.360000	1.540000	1.730000	1.770000	1.790000	1.93000
50%	1.340000	1.910000	2.060000	2.460000	2.670000	2.680000	3.23000

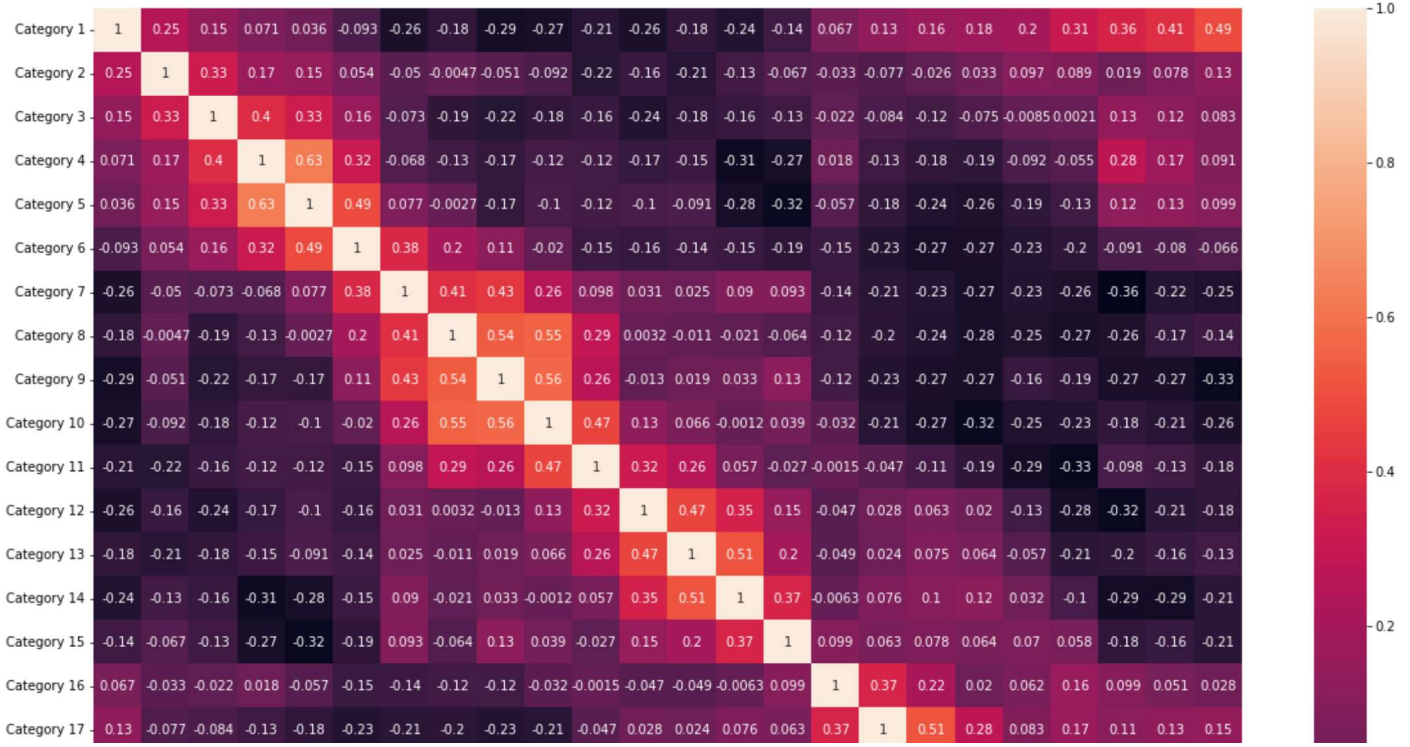
```
mydata_corr=mydata.corr()
```

```
mydata_corr
```

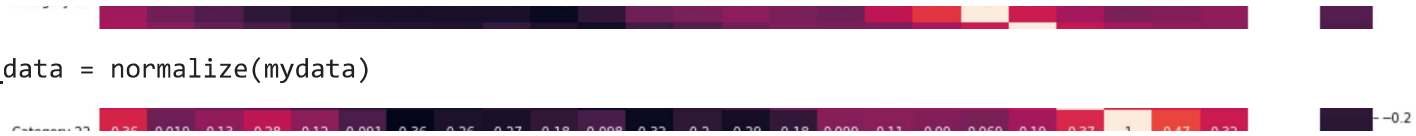
	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8
Category 1	1.000000	0.248970	0.149097	0.070810	0.035677	-0.092964	-0.264541	-0.181291
Category 2	0.248970	1.000000	0.325429	0.167332	0.153719	0.054090	-0.050117	-0.004703
Category 3	0.149097	0.325429	1.000000	0.396694	0.329425	0.160567	-0.073423	-0.186372
Category 4	0.070810	0.167332	0.396694	1.000000	0.626787	0.315107	-0.068372	-0.128352
Category 5	0.035677	0.153719	0.329425	0.626787	1.000000	0.489528	0.077164	-0.002728
Category 6	-0.092964	0.054090	0.160567	0.315107	0.489528	1.000000	0.382338	0.200532
Category 7	-0.264541	-0.050117	-0.073423	-0.068372	0.077164	0.382338	1.000000	0.406966
Category 8	-0.181291	-0.004703	-0.186372	-0.128352	-0.002728	0.200532	0.406966	1.000000
Category 9	-0.290303	-0.050811	-0.219795	-0.169703	-0.170545	0.114800	0.432183	0.536705
Category 10	-0.274588	-0.092441	-0.180092	-0.116054	-0.101121	-0.019901	0.255999	0.551613
Category 11	-0.214648	-0.221363	-0.158311	-0.117328	-0.124882	-0.148941	0.097737	0.292756
Category 12	-0.261221	-0.157124	-0.235697	-0.166125	-0.104913	-0.163905	0.030940	0.003219
Category 13	-0.178659	-0.213903	-0.182451	-0.145323	-0.090563	-0.137150	0.025466	-0.010656
Category 14	-0.237170	-0.126659	-0.162349	-0.307871	-0.280430	-0.153007	0.089702	-0.021491

Category 15	-0.135066	-0.066598	-0.133955	-0.271844	-0.323627	-0.187154	0.093076	-0.064381
Category 16	0.067343	-0.033044	-0.022477	0.018217	-0.056550	-0.149552	-0.143057	-0.124401
Category	0.130020	-0.077300	-0.084304	-0.132399	-0.184121	-0.229388	-0.207111	-0.203083

```
plt.figure(figsize=(20,15))
sns.heatmap(mydata_corr,annot=True)
plt.show()
```




```
from sklearn.preprocessing import normalize
```




```
norm_data = normalize(mydata)
```

```
norm_data
```

```
array([[0.        , 0.        , 0.33357969, ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.33148681, ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.3317615 , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.0749856 , 0.39885955, 0.3214808 , ..., 0.39885955, 0.39885955,
        0.08854682],
       [0.07238949, 0.30860782, 0.30860782, ..., 0.38099731, 0.38099731,
        0.0853434 ],
```

```
            [0.07508624, 0.32168526, 0.39519074, ..., 0.39519074, 0.39519074,
             0.09247463]])
```

```
data_norm_df = pd.DataFrame(norm_data, columns = mydata.columns)
```

```
data_norm_df
```

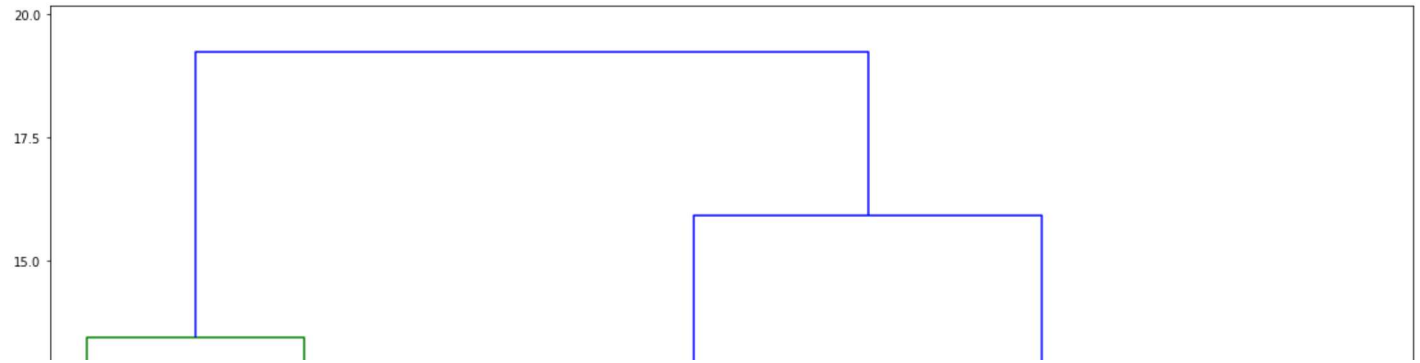
	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Catego
0	0.000000	0.000000	0.333580	0.335418	0.459476	0.268334	0.459476	0.215954	0.2141
1	0.000000	0.000000	0.331487	0.333313	0.456593	0.266651	0.456593	0.241081	0.2127
2	0.000000	0.000000	0.331762	0.331762	0.456972	0.266872	0.456972	0.241281	0.2129
3	0.000000	0.045907	0.333288	0.333288	0.459075	0.268100	0.459075	0.215765	0.2139
4	0.000000	0.000000	0.331762	0.331762	0.456972	0.266872	0.456972	0.241281	0.2129
...	...	...	...	...	...	...	...	...	
5449	0.066725	0.366622	0.293298	0.204575	0.203109	0.188444	0.178178	0.079924	0.1297
5450	0.079340	0.426561	0.342955	0.238021	0.237168	0.219253	0.151003	0.091284	0.1501
5451	0.074986	0.398860	0.321481	0.223361	0.221766	0.205014	0.139601	0.083761	0.1396
5452	0.072389	0.308608	0.308608	0.214120	0.212596	0.185927	0.134111	0.078485	0.1325
5453	0.075086	0.321685	0.395191	0.222888	0.221307	0.203128	0.191272	0.080619	0.1375

5454 rows × 24 columns

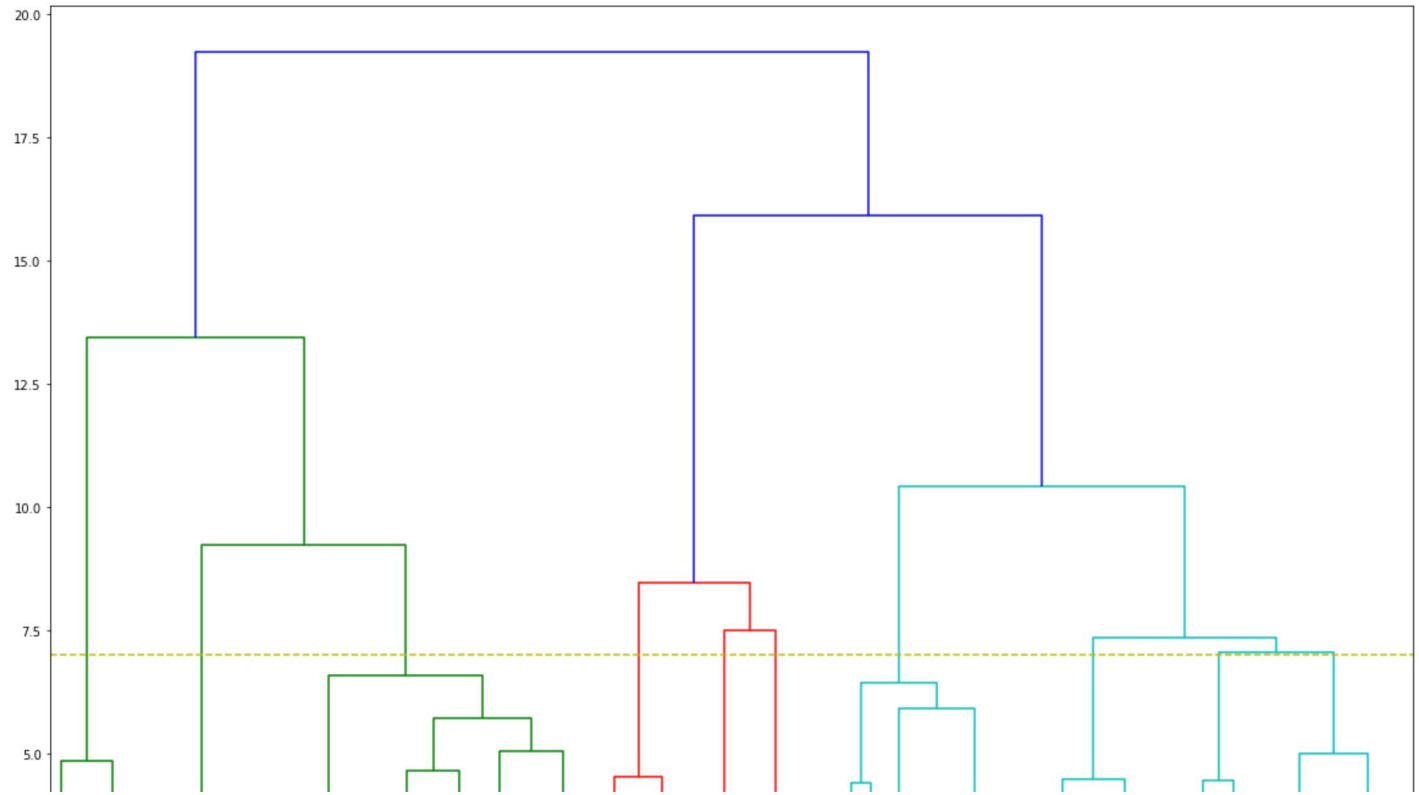
Performing Hierarchical Clustering

```
import scipy.cluster.hierarchy as HCluster
```

```
plt.figure(figsize = (20,15))
dendrogram = HCluster.dendrogram(HCluster.linkage(data_norm_df, method = 'ward'))
```




```
plt.figure(figsize = (20,15))
dendrogram = HCluster.dendrogram(HCluster.linkage(data_norm_df, method = 'ward'))
plt.axhline(y=7, color = 'y', linestyle = '--');
```




```
from sklearn.cluster import AgglomerativeClustering as AG
```


```
cluster = AG(n_clusters=10)
```

```
cluster.fit_predict(data_norm_df)
```

```
array([8, 8, 8, ..., 6, 6, 6])
```