IMPORTING LIBRARIES

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

IMPORTING DATASET

```
df=pd.read_csv("/content/User Profile Matching in Social Networks (Responses) - Form Respo
df.head()
```

	Timestamp	Email Address	Name	Age	Zodiac	Gender	I'm an	pı
0	9/8/2020 13:34:28	nandithankumar@gmail.com	Nanditha Menon	21	Leo	Female	Ambivert	anc Inc
1	9/8/2020 14:43:00	titusaishu@gmail.com	Aishwarya Titus	21	Aries	Female	Ambivert	Out
2	9/8/2020 14:44:08	meghanab2000@gmail.com	B Meghana	21	Pisces	Female	Introvert	anc Inc
3	9/8/2020 14:45:07	sayoojsanthosh21@gmail.com	Sayooj	20	NaN	Male	Ambivert	Out

```
df.columns=["timestamp","email","name","age","sunSign","gender","character","preferance","
```

CLEANING THE DATA

```
df.nunique()
```

```
timestamp     133
email         126
name          127
age            24
sunSign        30
gender          4
character       3
preferance      2
TV              3
genres        126
music          93
food            3
pets            9
dtype: int64
```

```
df.drop_duplicates(subset=["email"],inplace=True)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 126 entries, 0 to 132
Data columns (total 13 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   timestamp   126 non-null    object
 1   email       126 non-null    object
 2   name        126 non-null    object
 3   age         126 non-null    int64 
 4   sunSign     109 non-null    object
 5   gender      126 non-null    object
 6   character   126 non-null    object
 7   preferance  126 non-null    object
 8   TV          126 non-null    object
 9   genres      126 non-null    object
 10  music       126 non-null    object
 11  food        126 non-null    object
 12  pets        126 non-null    object
dtypes: int64(1), object(12)
memory usage: 13.8+ KB
```

```
df.index=range(len(df.index))
```

```
df=df.drop(['timestamp'],axis=1)
```

```
df.head()
```

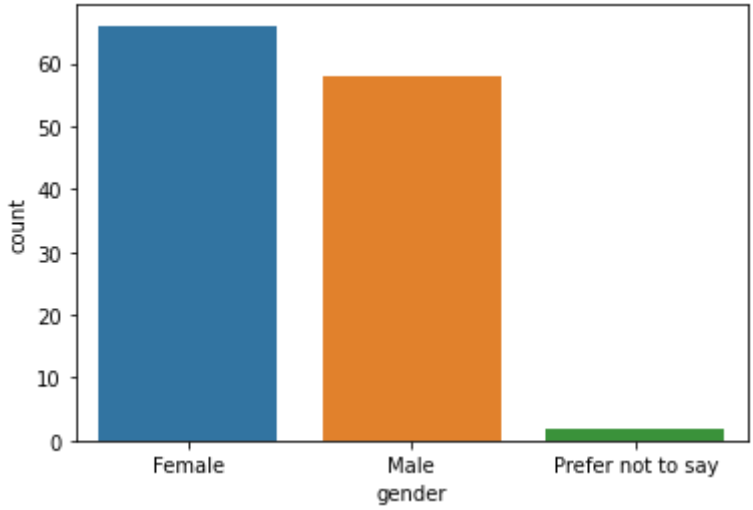
	email	name	age	sunSign	gender	character	preferance
0	nandithankumar@gmail.com	Nanditha Menon	21	Leo	Female	Ambivert	Warm and cozy Indoors.
1	titusaishu@gmail.com	Aishwarya	21	Aries	Female	Ambivert	The great
2	meghanab2000@gmail.com	Meghana	21	Pisces	Female	Introvert	cozy

EXPLORATORAY DATA ANALYSIS

Gender Count Plot

```
sns.countplot(df.gender)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd5726379b0>
```



Handling the Zodiac Sign column data

```
df.sunSign.unique()
```

```
array(['Leo', 'Aries', 'Pisces ', nan, 'Taurus', 'Gemini', 'Taurus ',
       'SAGITTARIUS', 'Cancer', '-', 'Libra', 'Capricorn ', 'Pisces',
       'Sagittarius', 'Capricon', 'Virgo',
       'Aquarius (according to tamil zodiac system)', 'Sagittarius ',
       'Capricorn', 'CANCER', 'pisces', 'Aquarius', 'No idea 😅',
       'Aquarius ', 'Gemini ', 'Scorpio', 'Scorpio ', 'Lion', 'No idea',
       'Who Cares'], dtype=object)
```

```
for i in df.index:
  try:
    df["sunSign"][i]=df["sunSign"][i].lower()
  except:
    df['sunSign'][i]="NaN"
  temp=df["sunSign"][i].split(" ")
```

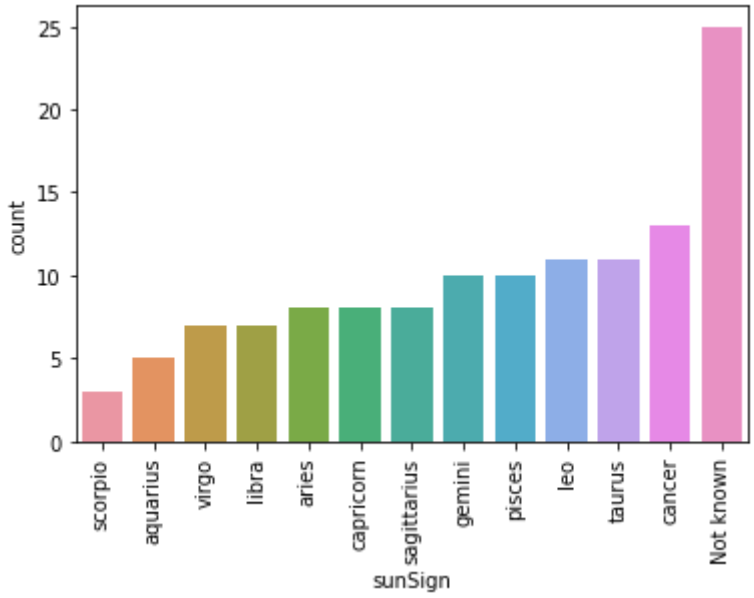
```
    df["sunSign"][i]=temp[0]
    if df["sunSign"][i] not in ("aries","taurus","leo","cancer","pisces","scorpio","libra","
      df['sunSign'][i]="Not known"
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  """
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  if __name__ == '__main__':
```

Zodiac Count Plot

```
order = df['sunSign'].value_counts(ascending=True).index
sns.countplot(x='sunSign', data=df, order=order)
plt.xticks(rotation=90)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12]),
 <a list of 13 Text major ticklabel objects>)
```



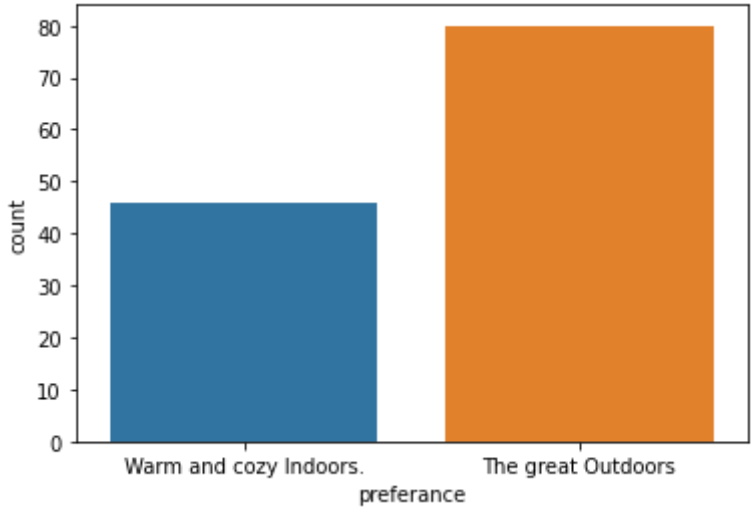
Observation - Not many people seem to focus on their zodiac sign

Since not many people care about their zodiac sign, the weightage assigned to zodiac sign can be less compared to the other attributes.

Personality Count Plot

```
order = df['preferance'].value_counts(ascending=True).index
sns.countplot(x='preferance', data=df, order=order)
```

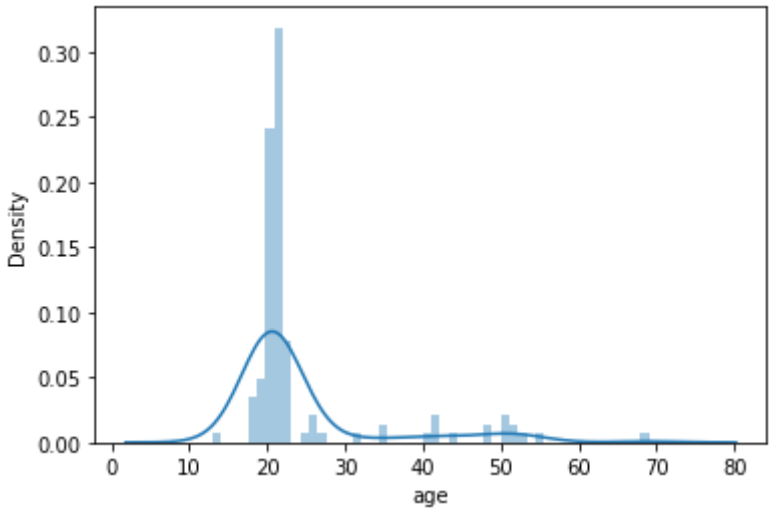
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd57250f470>
```



Age Visual Analysis

```
sns.distplot(df.age)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fd571fcc080>
```

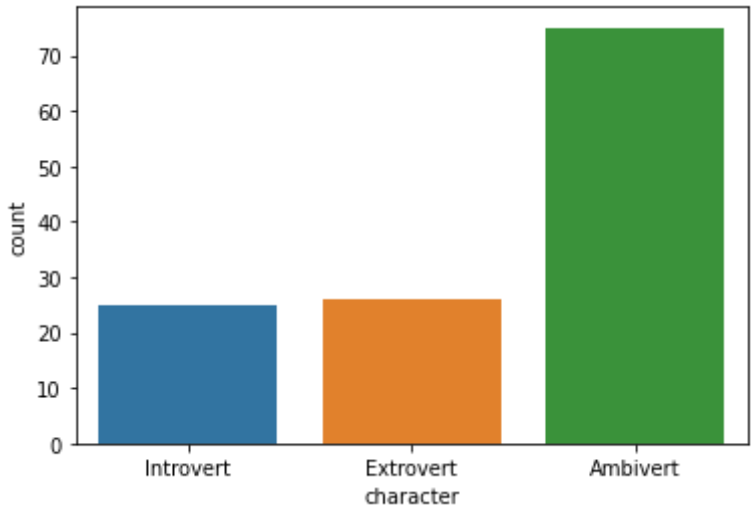


The age is concentrate in the 20 and 21 but I think it is becuase the data is mostly collected from our collegemates.

Social Preferences

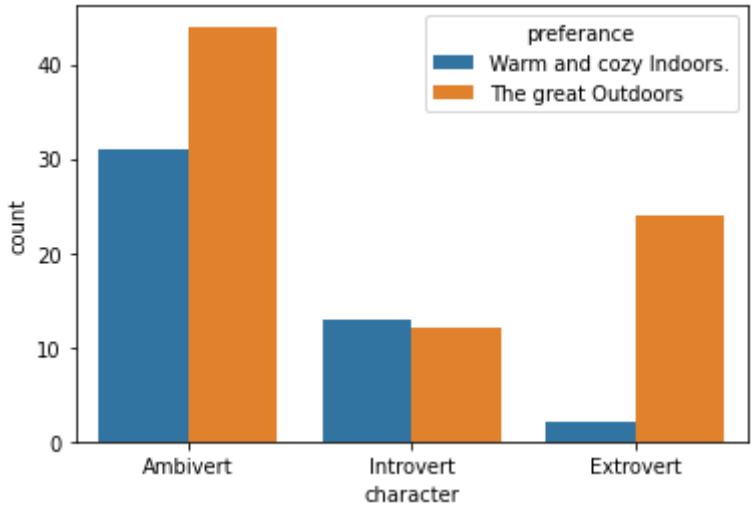
```
order = df['character'].value_counts(ascending=True).index
sns.countplot(x='character', data=df, order=order)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd571c620b8>
```




```
sns.countplot(df.character, hue=df.preferance)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd571ebf438>
```

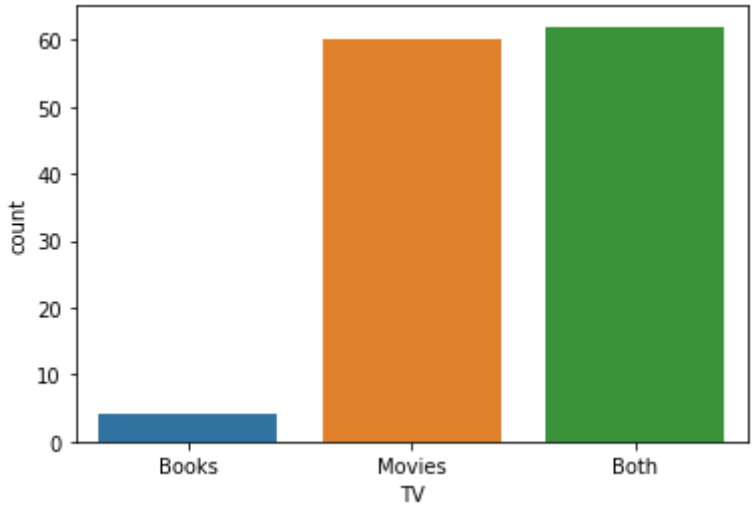


Introverts seem to not mind outdoor or indoor while leaning to the indoor. Ambiverts and extroverts have an outdoor prefernace.

Entertainment Preferences

```
order = df['TV'].value_counts(ascending=True).index
sns.countplot(x='TV', data=df, order=order)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd570c190f0>
```

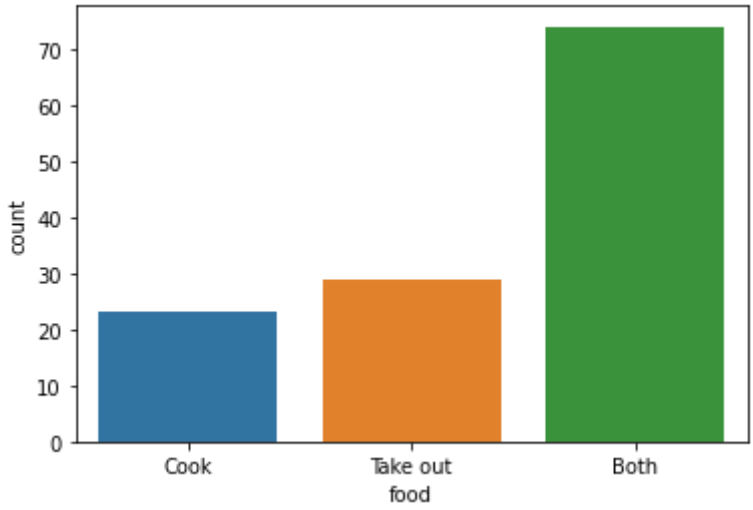


There is a general liking towards TV shows compared to books. People who like only books seem to be very low.

Food Preferences

```
order = df['food'].value_counts(ascending=True).index
sns.countplot(x='food', data=df, order=order)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd570afc9b0>
```




```
sns.countplot(df.gender, hue=df.character)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd570a6fac8>
```

Handling Pets column data

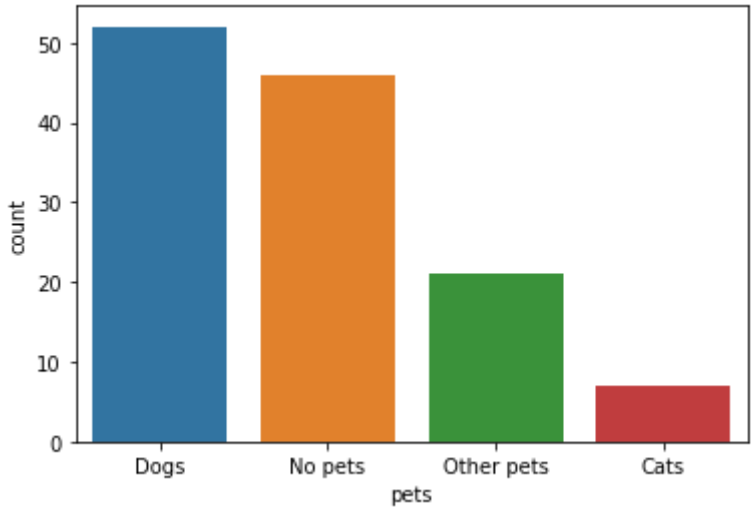
```
for i in df.index:
  if df["pets"][i] not in ("Dogs","Cats","No pets"):
    df["pets"][i]="Other pets"
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarnir
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  This is separate from the ipykernel package so we can avoid doing imports until
```

Pet Preferences

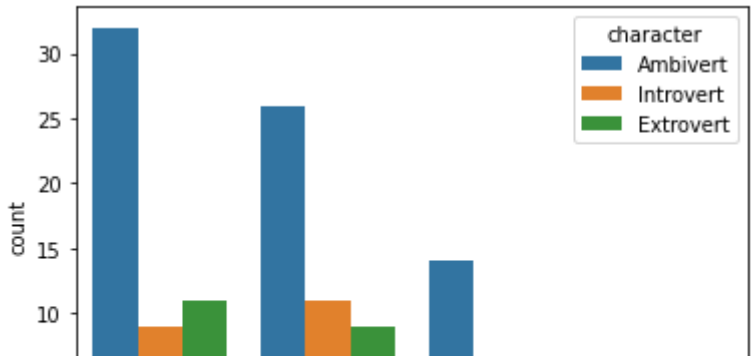
```
sns.countplot(df.pets)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd5709ea400>
```




```
sns.countplot(df.pets,hue=df.character)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fd5709c0ba8>
```



Handling TV Show Genre column data

```
listGenres=set(["Action Genre","Animation","Comedy Genre","Crime","Drama","Experimental","
for i in df.index:
  try:
    temp=df["genres"][i]
    temp=temp.split(",")
    temp=[j.strip() for j in temp]
    for k in range(len(temp)):
      if temp[k] not in listGenres:
        temp[k]="other TVShow Genre"
    df["genres"][i]=temp
  except:
    pass
listGenres.add("otherTVShowGenre")
```

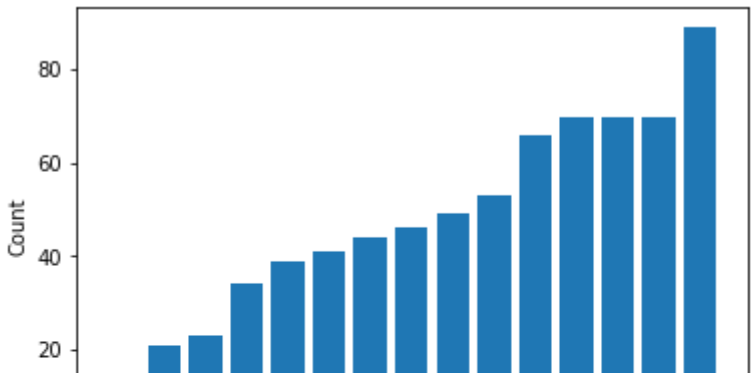
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  # Remove the CWD from sys.path while we load stuff.
```

```
for i in listGenres:
  temp_list=[]
  for j in df.index:
    try:
      if i in df["genres"][j]:
        temp_list.append(1)
      else:
        temp_list.append(0)
    except:
      pass
  df[i]=temp_list


df.drop(labels="genres",axis=1,inplace=True)
df.head()
```

	email	name	age	sunSign	gender	character	preferance
0	nandithankumar@gmail.com	Nanditha Menon	21	leo	Female	Ambivert	Warm and cozy Indoors.
1	titusaishu@gmail.com	Aishwarya Titus	21	aries	Female	Ambivert	The great Outdoors
2	meghanab2000@gmail.com	B Meghana	21	pisces	Female	Introvert	Warm and cozy Indoors.
3	sayoojsanthosh21@gmail.com	Sayooj	20	Not known	Male	Ambivert	The great Outdoors
4	vishnu21200@gmail.com	Vishnu Mahesh Pothuvath	20	pisces	Male	Ambivert	Warm and cozy Indoors.

TV Genre Preferences

```
# Let's start by visualizing the distribution of gender in the dataset.
fig, ax = plt.subplots()
# Counting the Genres of TV Shows and Movies for each person and summing them up
genreCount=[]
for i in listGenres:
  genreCount.append(df[i].sum())
def sort_list(list1, list2):
    zipped_pairs = zip(list2, list1)
    l1 = [y for x, y in sorted(zipped_pairs)]
    return l1
listGenres=sort_list(listGenres,genreCount)
genreCount.sort()
ax.bar(listGenres, genreCount)
ax.set_xlabel('TV Show Genre')
ax.set_ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```



Handling Music Genre column data

```
listMusicGenres=set(["Blues","Jazz","Rock Music","Country","Pop","Hip Hop","EDM","K-POP","
for i in df.index:
  try:
    temp=df["music"][i]
    temp=temp.split(",")
    temp=[j.strip() for j in temp]
    for k in range(len(temp)):
      if temp[k] not in listMusicGenres:
        temp[k]="otherMusic"
    df["music"][i]=temp
  except:
    pass
listMusicGenres.add("otherMusic")
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  # Remove the CWD from sys.path while we load stuff.
```

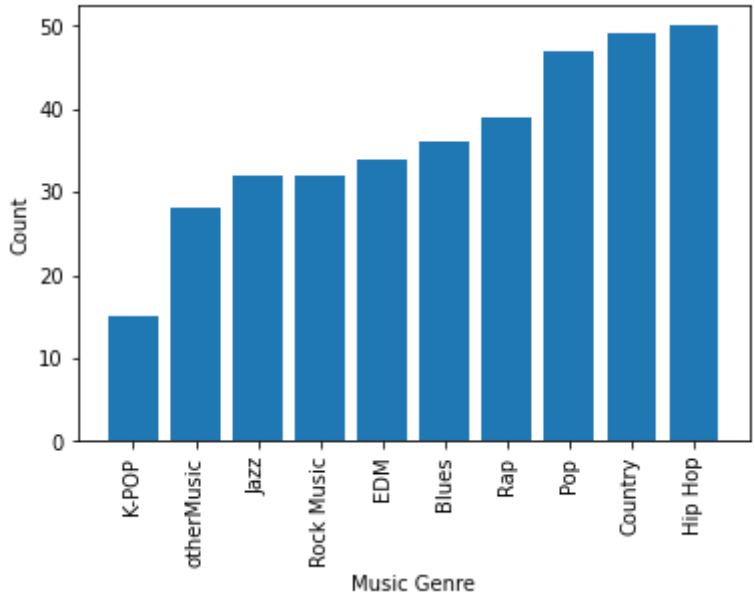
```
for i in listMusicGenres:
  temp_list=[]
  for j in df.index:
    try:
      if i in df["music"][j]:
        temp_list.append(1)
      else:
        temp_list.append(0)
    except:
      pass
  df[i]=temp_list
```

```
df.drop(labels="music",axis=1,inplace=True)
df.head()
```

	email	name	age	sunSign	gender	character	preferance
0	nandithankumar@gmail.com	Nanditha Menon	21	leo	Female	Ambivert	Warm and cozy Indoors.
1	titusaishu@gmail.com	Aishwarya Titus	21	aries	Female	Ambivert	The great Outdoors
2	meghanab2000@gmail.com	B Meghana	21	pisces	Female	Introvert	Warm and cozy Indoors.
3	sayoojsanthosh21@gmail.com	Sayooj	20	Not known	Male	Ambivert	The great Outdoors
		Vishnu					Warm and

Music Preferences

```
# Let's start by visualizing the distribution of gender in the dataset.
fig, ax = plt.subplots()
# Counting Music genres in the same way as TV Show Genres
musicGenreCount=[]
for i in listMusicGenres:
  musicGenreCount.append(df[i].sum())
# Plotting the bar graph
def sort_list(list1, list2):
    zipped_pairs = zip(list2, list1)
    l1 = [y for x, y in sorted(zipped_pairs)]
    return l1
listMusicGenres=sort_list(listMusicGenres,musicGenreCount)
musicGenreCount.sort()
ax.bar(listMusicGenres, musicGenreCount)
ax.set_xlabel('Music Genre')
ax.set_ylabel('Count')
plt.xticks(rotation=90)
plt.show()
```



ENCODING OTHER CATEGORICAL DATA FOR CORRELATION

```
dummies=pd.get_dummies(df.sunSign)
df=df.drop(['sunSign'],axis=1)
df = pd.concat([df, dummies], axis=1, sort=False)


dummies=pd.get_dummies(df.food)
df=df.drop(['food'],axis=1)
df = pd.concat([df, dummies], axis=1, sort=False)


df.rename(columns={'Both':'Both_Food'},inplace=True)


dummies=pd.get_dummies(df.TV)
df=df.drop(['TV'],axis=1)
df = pd.concat([df, dummies], axis=1, sort=False)


dummies=pd.get_dummies(df.pets)
df=df.drop(['pets'],axis=1)
df = pd.concat([df, dummies], axis=1, sort=False)


dummies=pd.get_dummies(df.character)
df=df.drop(['character'],axis=1)
df = pd.concat([df, dummies], axis=1, sort=False)


df.head()
```

	email	name	age	gender	preferance	Action Genre	Drama	Hist
0	nandithankumar@gmail.com	Nanditha Menon	21	Female	Warm and cozy Indoors.	0	1	
1	titusaishu@gmail.com	Aishwarya Titus	21	Female	The great Outdoors	1	0	
2	meghanab2000@gmail.com	B Meghana	21	Female	Warm and cozy Indoors.	0	0	
3	sayoojsanthosh21@gmail.com	Sayooj	20	Male	The great Outdoors	1	0	
4	vishnu21200@gmail.com	Vishnu Mahesh Pothuvath	20	Male	Warm and cozy Indoors.	0	0	

```
# Import label encoder
from sklearn import preprocessing
```

```
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()
df['gender']= label_encoder.fit_transform(df['gender'])
df['preferance']= label_encoder.fit_transform(df['preferance'])


details=df.iloc[:,:2]
details.head()
```

	email	name
0	nandithankumar@gmail.com	Nanditha Menon
1	titusaishu@gmail.com	Aishwarya Titus
2	meghanab2000@gmail.com	B Meghana
3	sayoojsanthosh21@gmail.com	Sayooj
4	vishnu21200@gmail.com	Vishnu Mahesh Pothuvath

```
df=df.iloc[:,2:]


df.head()
```

	age	gender	preferance	Action Genre	Drama	Historical	Romance	Experimental	Crime
0	21	0	1	0	1	0	0	0	1
1	21	0	0	1	0	0	0	0	1
2	21	0	1	0	0	1	1	0	1
3	20	1	0	1	0	1	0	1	1
4	20	1	1	0	0	1	1	0	0

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(df.iloc[:,:1])
```

```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

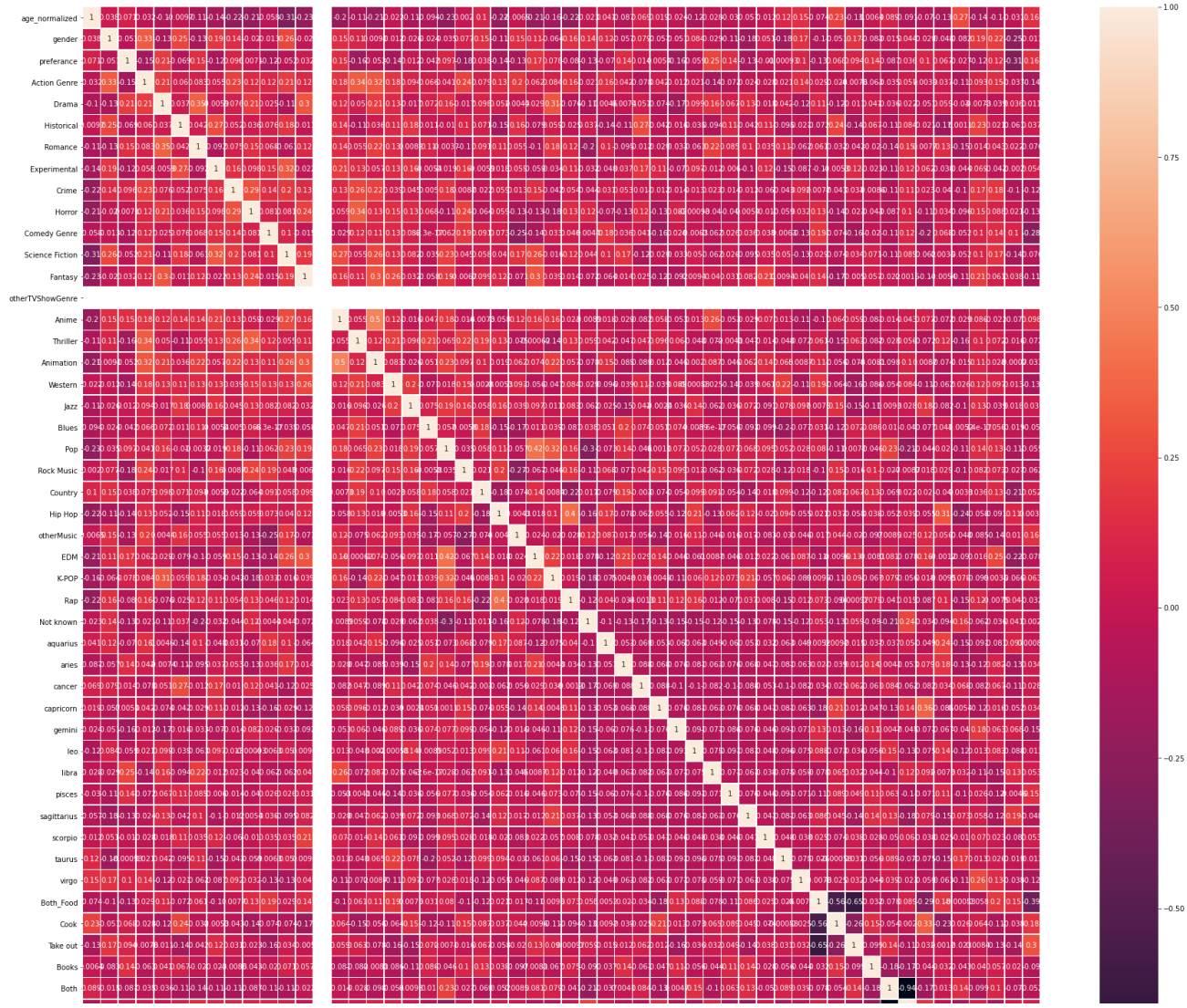
```
temp=scaler.transform(df.iloc[:,:1])
temp=pd.DataFrame(temp,columns=['age_normalized'])
df=pd.concat([temp,df],axis=1,sort=False)


df=df.drop(['age'],axis=1)


df.head()
```

	age_normalized	gender	preferance	Action Genre	Drama	Historical	Romance	Experiment
0	0.142857	0	1	0	1	0	0	
1	0.142857	0	0	1	0	0	0	
2	0.142857	0	1	0	0	1	1	
3	0.125000	1	0	1	0	1	0	
4	0.125000	1	1	0	0	1	1	

CORRELATION HEATMAP

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(30,30))
sns.heatmap(df.corr(), annot=True, linewidths=.5, ax=ax)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd570822e80>
```



When variables used in clustering have strong correlation, some variables get a higher weight than others. If two variables are perfectly correlated, they effectively represent the same concept. But that concept is now represented twice in the data and hence gets twice the weight of all the other variables. The final solution is likely to be skewed in the direction of that concept, which could be a problem if it's not anticipated

STRONG CORRELATION FEATURES LIST -

```
corr = df.corr()
c1 = corr.abs()
upper = c1.where(np.triu(np.ones(c1.shape), k=1).astype(np.bool))
# Find index of feature columns with correlation greater than 0.5
to_drop = [column for column in upper.columns if any(upper[column] >= 0.5)]
print(to_drop)
```

```
['Cook', 'Take out', 'Movies', 'No pets', 'Extrovert', 'Introvert']
```

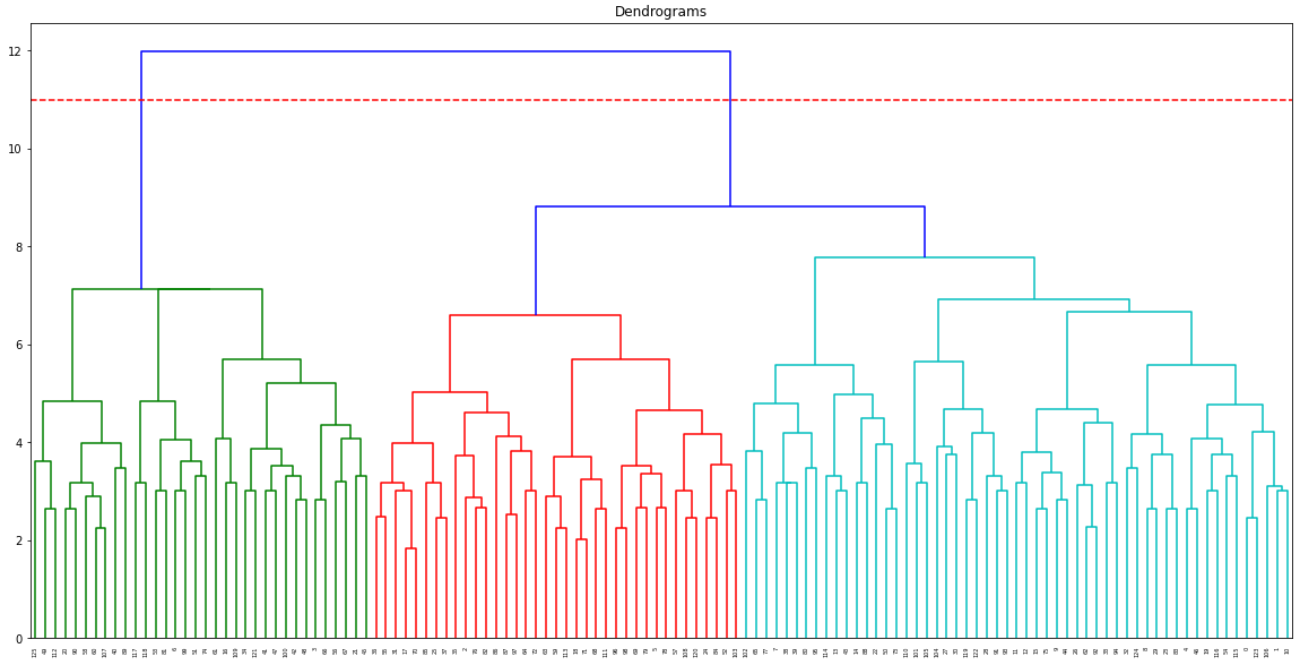
```
df.drop(columns=to_drop,axis=1,inplace=True)
```

CLUSTERING ALGORITHMS

HIERARCHICAL CLUSTERING

```
import scipy.cluster.hierarchy as hc
plt.figure(figsize=(20, 10))
plt.title("Dendrograms")
linkage=hc.linkage(df, method='ward')
dend = hc.dendrogram(linkage)
plt.axhline(y=11, color='r', linestyle='--')
```

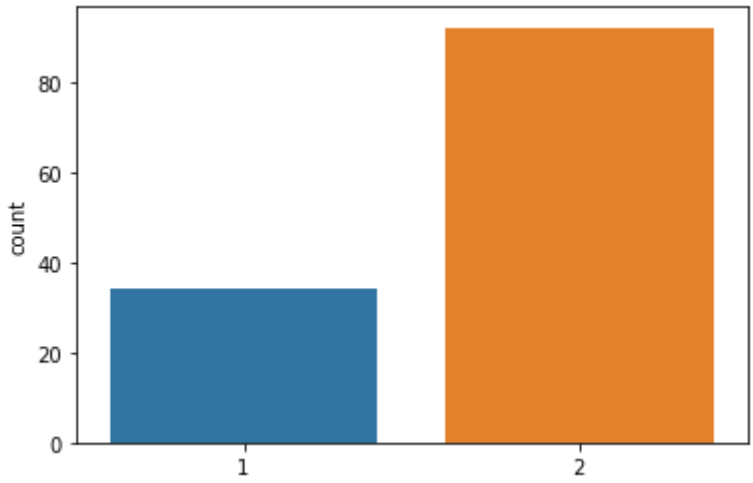
```
<matplotlib.lines.Line2D at 0x7fd5613c3c18>
```




```
from scipy.cluster.hierarchy import fcluster
fl = fcluster(linkage,2,criterion='maxclust')
```

```
sns.countplot(fl)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid pc

<matplotlib.axes._subplots.AxesSubplot at 0x7fd56279ff28>
```




```
import plotly.express as px
from sklearn.decomposition import PCA

pca = PCA(n_components=3)
components = pca.fit_transform(df)

fig = px.scatter_3d(
    components, x=0, y=1, z=2, color=fl,
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'}
)
fig.show()
```

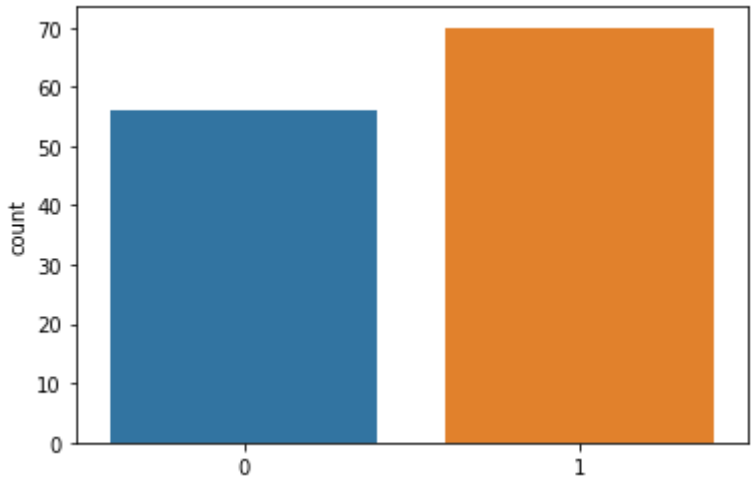
KMEANS

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=3).fit(df)
```

```
y_kmeans=kmeans.labels_
```

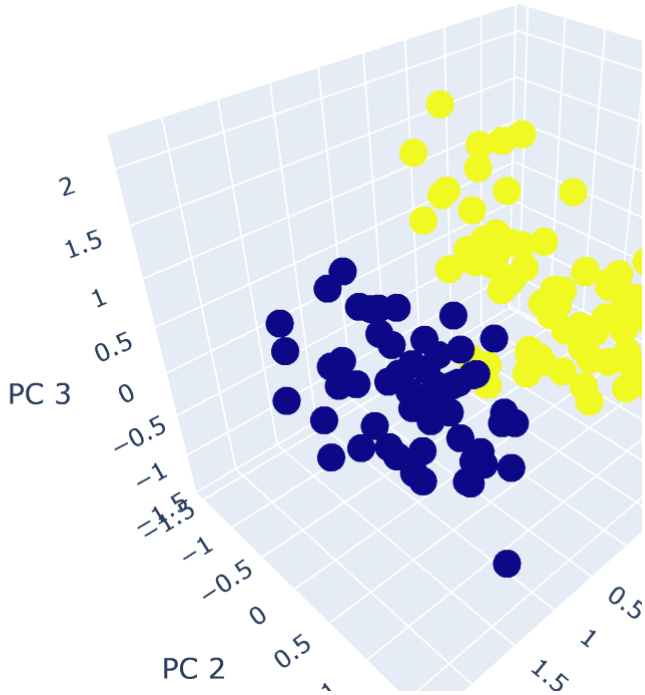
```
sns.countplot(y_kmeans)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid pc

<matplotlib.axes._subplots.AxesSubplot at 0x7fd561cb4f60>
```




```
import plotly.express as px
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=3)
components = pca.fit_transform(df)

fig = px.scatter_3d(
    components, x=0, y=1, z=2, color=y_kmeans,
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'}
)
fig.show()
```

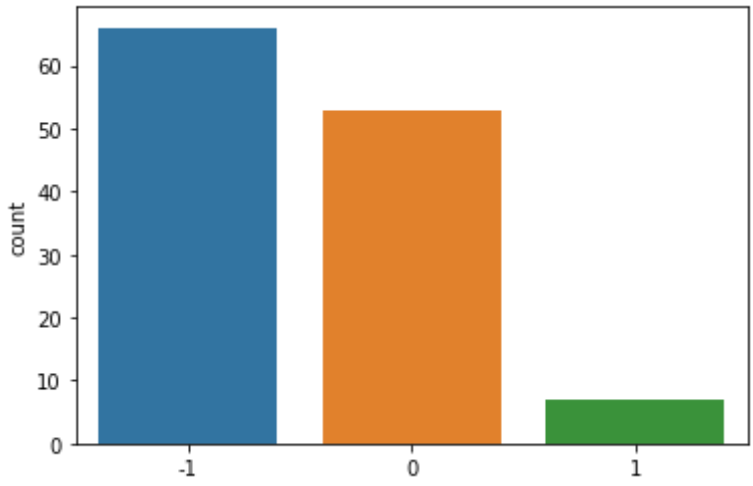


DBSCAN

```
from sklearn.cluster import DBSCAN
clustering = DBSCAN(eps=3, min_samples=5).fit(df)
db_lables=clustering.labels_
```

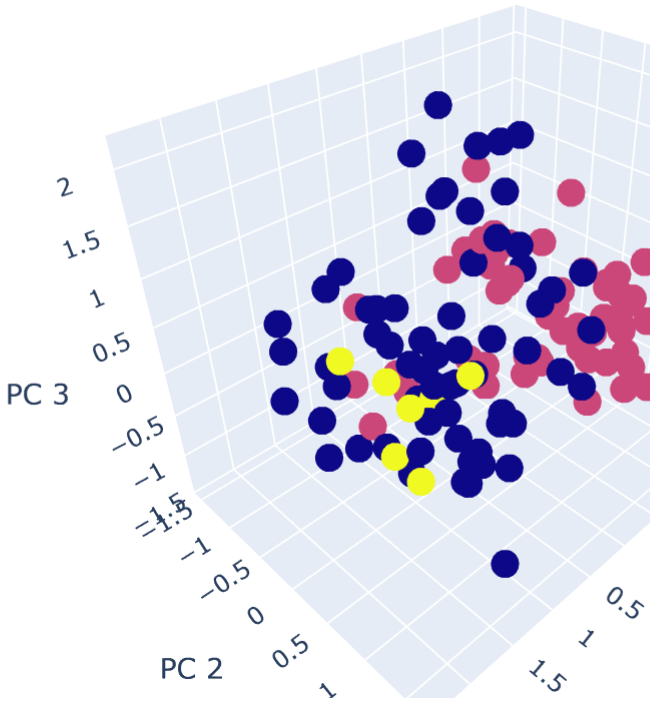
```
sns.countplot(db_lables)
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid pc

<matplotlib.axes._subplots.AxesSubplot at 0x7fd5628a83c8>
```



-1 -> Noisy samples DBSCAN is classifying more than 40 users as noisy sample - not suitable for clustering this way. Inaccurate

```
import plotly.express as px
from sklearn.decomposition import PCA


pca = PCA(n_components=3)
components = pca.fit_transform(df)

fig = px.scatter_3d(
    components, x=0, y=1, z=2, color=db_lables,
    labels={'0': 'PC 1', '1': 'PC 2', '2': 'PC 3'}
)
fig.show()
```



PERFORMANCE EVALUATION METRICS

```
from sklearn import metrics

cluster_metrics=pd.DataFrame(columns=['silhoutte score','db score','ch score'],index=['com
cluster_metrics['silhoutte score']['complete hierarchical']=metrics.silhouette_score(df, f
cluster_metrics['silhoutte score']['kmeans']=metrics.silhouette_score(df, y_kmeans, metric
```

```
cluster_metrics['silhoutte score']['kmeans']=metrics.silhouette_score(df, y_kmeans, metric
cluster_metrics['silhoutte score']['dbscan']=metrics.silhouette_score(df, db_lables, metri

cluster_metrics['db score']['complete hierarchical']=metrics.davies_bouldin_score(df, fl)
cluster_metrics['db score']['kmeans']=metrics.davies_bouldin_score(df, y_kmeans)
cluster_metrics['db score']['dbscan']=metrics.davies_bouldin_score(df, db_lables)

cluster_metrics['ch score']['complete hierarchical']=metrics.calinski_harabasz_score(df, f
cluster_metrics['ch score']['kmeans']=metrics.calinski_harabasz_score(df, y_kmeans)
cluster_metrics['ch score']['dbscan']=metrics.calinski_harabasz_score(df, db_lables)


cluster_metrics
```

	silhoutte score	db score	ch score
complete hierarchical	0.0794176	3.20255	9.47503
kmeans	0.0912907	3.03123	13.3231
dbscan	0.0407356	3.32315	6.34775

Other evaluation metrics are not usable since this is an unsupervised clustering.

Evaluation metrics k-means is a little better than "complete" hierarchical clustering

Nearest Neighbour Search

```
df['cluster_ward']=fl


df['cluster_kmeans']=y_kmeans


cluster1=df[df['cluster_kmeans']==1]


cluster2=df[df['cluster_kmeans']==0]
```

KNN *WITHOUT TAKING THE CLUSTERS INTO CONSIDERATION*

```
import time

start=time.time()

from sklearn.neighbors import NearestNeighbors
import numpy as np
nbrs = NearestNeighbors(n_neighbors=5, algorithm='kd_tree').fit(df)
distances, indices = nbrs.kneighbors(df)
```

```
end = time.time()

# total time taken
print(f"Runtime of the program is {end - start}")
```

```
Runtime of the program is 0.01264333724975586
```

```
nearest={}
for i in indices:
  cur=details['name'][i[0]]
  nearest[cur]=[]
  for j in i:
    nearest[cur].append(details['name'][j])
```

```
nearest['Nanditha Menon']
```

```
['Nanditha Menon', 'Sneha', 'Vismaya R Mohan', 'Yukta Srivastava', 'Sai sudha']
```

```
nearest['Srihitha']
```

```
['Srihitha', 'Seema', 'Neshva Salim', 'Jaswanth', 'Subhadra Narayanan Kutty']
```

```
nearest['Rishi']
```

```
['Rishi', 'Muralidhar', 'Krishnasagar', 'Vikram Chandrasekaran', 'Mithun']
```

```
nearest['Aakarsh']
```

```
['Aakarsh', 'Viswesh K S', 'V.N.V SRI RAM ', 'Sanjay', 'Harsha']
```

KNN *TAKING THE CLUSTERS INTO CONSIDERATION*

```
import time

start=time.time()

from sklearn.neighbors import NearestNeighbors
import numpy as np
nbrs = NearestNeighbors(n_neighbors=5, algorithm='kd_tree').fit(cluster2)
distances2, indices2 = nbrs.kneighbors(cluster2)
```

```
end = time.time()

# total time taken
print(f"Runtime of the program is {end - start}")
```

```
Runtime of the program is 0.005358695983886719
```

```
map_index2={}
```

```
 k=0
 for i in cluster2.index:
   map_index2[k]=i
   k+=1


nearest_2={}
k=0
for i in cluster2.index:
  cur=details['name'][i]
  nearest_2[cur]=[]
  for j in indices2[k]:
    nearest_2[cur].append(details['name'][map_index2[j]])
  k+=1


nearest_2['Aakarsh']
```

```
['Aakarsh', 'Viswesh K S', 'V.N.V SRI RAM ', 'Sanjay', 'Harsha']
```

```
nearest_2['Rishi']
```

```
['Rishi', 'Muralidhar', 'Krishnasagar', 'Mithun', 'Midhun Babu']
```

```
from sklearn.neighbors import NearestNeighbors
import numpy as np
nbrs = NearestNeighbors(n_neighbors=5, algorithm='kd_tree').fit(cluster1)
distances1, indices1 = nbrs.kneighbors(cluster1)


 map_index1={}
 k=0
 for i in cluster1.index:
   map_index1[k]=i
   k+=1


nearest_1={}
k=0
for i in cluster1.index:
  cur=details['name'][i]
  nearest_1[cur]=[]
  for j in indices1[k]:
    nearest_1[cur].append(details['name'][map_index1[j]])
  k+=1


nearest_1['Srihitha']
```

```
['Srihitha', 'Seema', 'Neshva Salim', 'Jaswanth', 'Subhadra Narayanan Kutty']
```

```
nearest_1['Nanditha Menon']
```

```
['Nanditha Menon',
 'Sneha',
```

```
 'Vismaya R Mohan',
 'Yukta Srivastava',
 'Nivitha Varghese ']
```

ADVANTAGES

1. Clustering the users reduces the size of the dataset passed into the knn algorithm. This in turn leads to reduced execution time while finding k nearest neighbours of the current node.
2. Clustering can also provide deeper insights into the users through clusters and related cluster analysis.

DISADVANTAGES

1. The nodes near the edge might suffer from variation in predictions if the clusters are not far enough.

FUTURE DIRECTION

1. Add columns with descriptive sentences so users are able to express themselves more and then try to find the nearest neighbours or clusters.