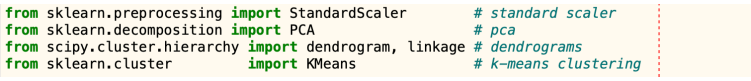
1. Import libraries

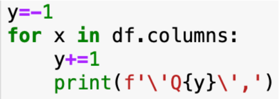


```
from sklearn.preprocessing import StandardScaler        # standard scaler
from sklearn.decomposition import PCA                   # pca
from scipy.cluster.hierarchy import dendrogram, linkage # dendrograms
from sklearn.cluster         import KMeans              # k-means clustering
```

```
from sklearn.model_selection import train_test_split   # train-test-split
```

2. Load data as “df”

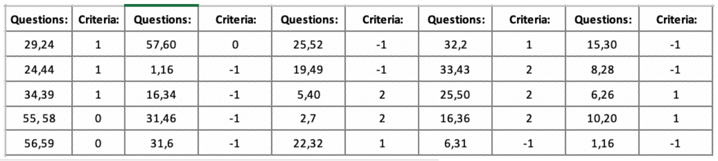
3. Rename columns as Q1, Q2…



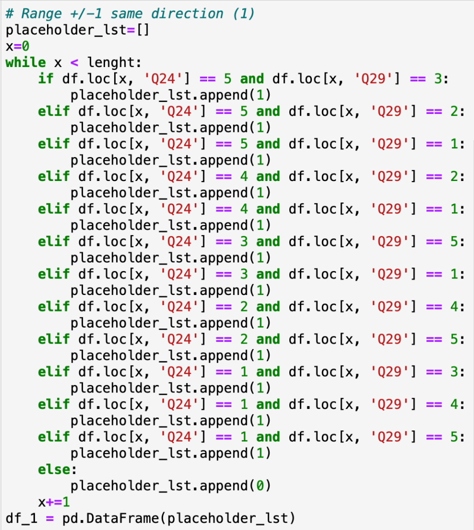
```
y=-1
for x in df.columns:
    y+=1
    print(f'\'Q{y}\',')
```

4. Explore data and heat maps (S 8b)

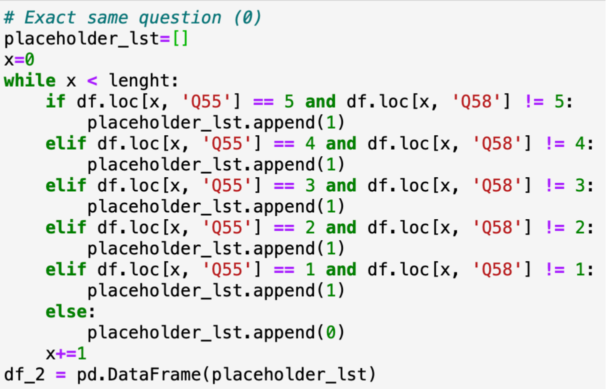
5. Data quality analysis



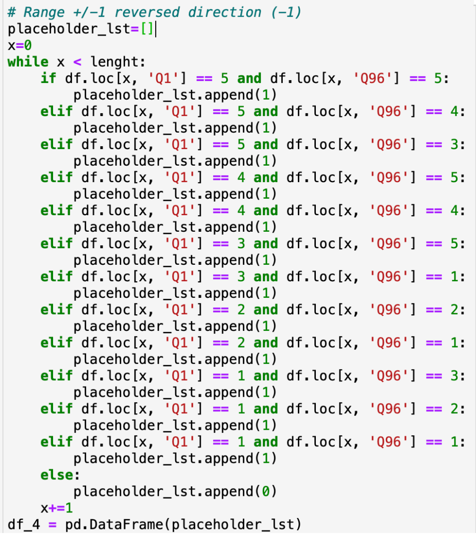
Questions:	Criteria:	Questions:	Criteria:	Questions:	Criteria:	Questions:	Criteria:	Questions:	Criteria:
29,24	1	57,60	0	25,52	-1	32,2	1	15,30	-1
24,44	1	1,16	-1	19,49	-1	33,43	2	8,28	-1
34,39	1	16,34	-1	5,40	2	25,50	2	6,26	1
55, 58	0	31,46	-1	2,7	2	16,36	2	10,20	1
56,59	0	31,6	-1	22,32	1	6,31	-1	1,16	-1



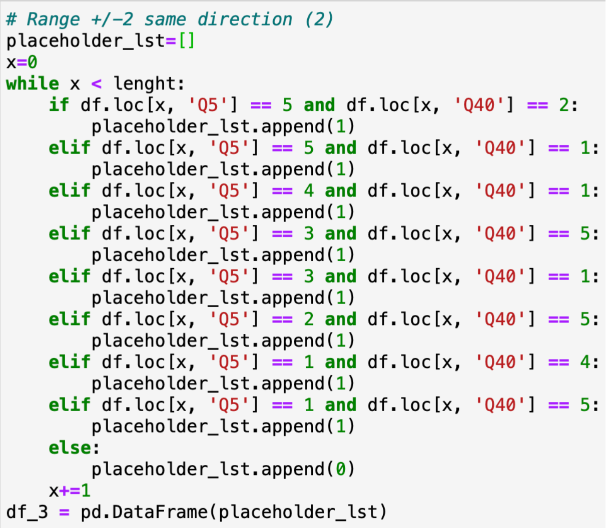
```
# Range +/-1 same direction (1)
placeholder_lst=[]
x=0
while x < lenght:
    if df.loc[x, 'Q24'] == 5 and df.loc[x, 'Q29'] == 3:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 5 and df.loc[x, 'Q29'] == 2:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 5 and df.loc[x, 'Q29'] == 1:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 4 and df.loc[x, 'Q29'] == 2:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 4 and df.loc[x, 'Q29'] == 1:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 3 and df.loc[x, 'Q29'] == 5:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 3 and df.loc[x, 'Q29'] == 1:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 2 and df.loc[x, 'Q29'] == 4:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 2 and df.loc[x, 'Q29'] == 5:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 1 and df.loc[x, 'Q29'] == 3:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 1 and df.loc[x, 'Q29'] == 4:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q24'] == 1 and df.loc[x, 'Q29'] == 5:
        placeholder_lst.append(1)
    else:
        placeholder_lst.append(0)
    x+=1
df_1 = pd.DataFrame(placeholder_lst)
```



```
# Exact same question (0)
placeholder_lst=[]
x=0
while x < lenght:
    if df.loc[x, 'Q55'] == 5 and df.loc[x, 'Q58'] != 5:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q55'] == 4 and df.loc[x, 'Q58'] != 4:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q55'] == 3 and df.loc[x, 'Q58'] != 3:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q55'] == 2 and df.loc[x, 'Q58'] != 2:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q55'] == 1 and df.loc[x, 'Q58'] != 1:
        placeholder_lst.append(1)
    else:
        placeholder_lst.append(0)
    x+=1
df_2 = pd.DataFrame(placeholder_lst)
```

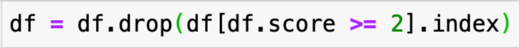


```
# Range +/-1 reversed direction (-1)
placeholder_lst=[]
x=0
while x < lenght:
    if df.loc[x, 'Q1'] == 5 and df.loc[x, 'Q96'] == 5:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 5 and df.loc[x, 'Q96'] == 4:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 5 and df.loc[x, 'Q96'] == 3:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 4 and df.loc[x, 'Q96'] == 5:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 4 and df.loc[x, 'Q96'] == 4:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 3 and df.loc[x, 'Q96'] == 5:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 3 and df.loc[x, 'Q96'] == 1:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 2 and df.loc[x, 'Q96'] == 2:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 2 and df.loc[x, 'Q96'] == 1:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 1 and df.loc[x, 'Q96'] == 3:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 1 and df.loc[x, 'Q96'] == 2:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q1'] == 1 and df.loc[x, 'Q96'] == 1:
        placeholder_lst.append(1)
    else:
        placeholder_lst.append(0)
    x+=1
df_4 = pd.DataFrame(placeholder_lst)
```



```
# Range +/-2 same direction (2)
placeholder_lst=[]
x=0
while x < lenght:
    if df.loc[x, 'Q5'] == 5 and df.loc[x, 'Q40'] == 2:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q5'] == 5 and df.loc[x, 'Q40'] == 1:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q5'] == 4 and df.loc[x, 'Q40'] == 1:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q5'] == 3 and df.loc[x, 'Q40'] == 5:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q5'] == 3 and df.loc[x, 'Q40'] == 1:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q5'] == 2 and df.loc[x, 'Q40'] == 5:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q5'] == 1 and df.loc[x, 'Q40'] == 4:
        placeholder_lst.append(1)
    elif df.loc[x, 'Q5'] == 1 and df.loc[x, 'Q40'] == 5:
        placeholder_lst.append(1)
    else:
        placeholder_lst.append(0)
    x+=1
df_3 = pd.DataFrame(placeholder_lst)
```

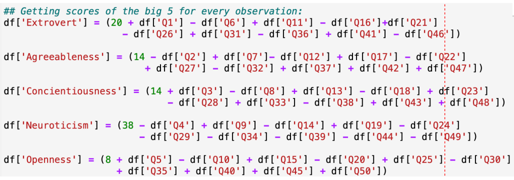
6. Drop the rows based on data quality



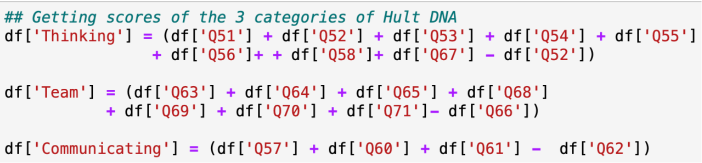
```
df = df.drop(df[df.score >= 2].index)
```

**Depending on results

7. Feature Engineering

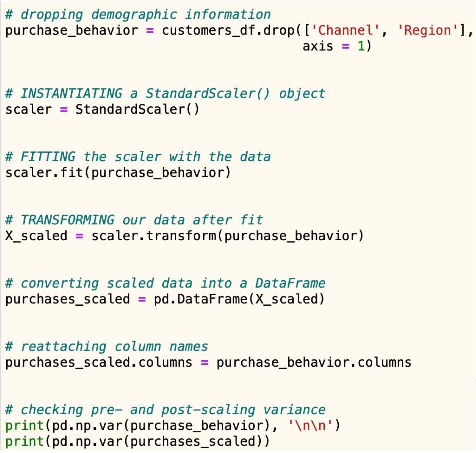


```
## Getting scores of the big 5 for every observation:
df['Extrovert'] = (20 + df['Q1'] - df['Q6'] + df['Q11'] - df['Q16']+df['Q21']
                    - df['Q26'] + df['Q31'] - df['Q36'] + df['Q41'] - df['Q46'])

df['Agreeableness'] = (14 - df['Q2'] + df['Q7']- df['Q12'] + df['Q17'] - df['Q22']
                        + df['Q27'] - df['Q32'] + df['Q37'] + df['Q42'] + df['Q47'])

df['Concientiousness'] = (14 + df['Q3'] - df['Q8'] + df['Q13'] - df['Q18'] + df['Q23']
                           - df['Q28'] + df['Q33'] - df['Q38'] + df['Q43'] + df['Q48'])

df['Neuroticism'] = (38 - df['Q4'] + df['Q9'] - df['Q14'] + df['Q19'] - df['Q24']
                      - df['Q29'] - df['Q34'] - df['Q39'] - df['Q44'] - df['Q49'])

df['Openness'] = (8 + df['Q5'] - df['Q10'] + df['Q15'] - df['Q20'] + df['Q25'] - df['Q30']
                   + df['Q35'] + df['Q40'] + df['Q45'] + df['Q50'])
```

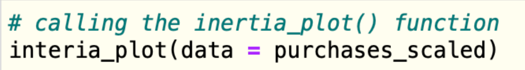


```
## Getting scores of the 3 categories of Hult DNA
df['Thinking'] = (df['Q51'] + df['Q52'] + df['Q53'] + df['Q54'] + df['Q55']
                  + df['Q56']+ + df['Q58']+ df['Q67'] - df['Q52'])

df['Team'] = (df['Q63'] + df['Q64'] + df['Q65'] + df['Q68']
              + df['Q69'] + df['Q70'] + df['Q71']- df['Q66'])

df['Communicating'] = (df['Q57'] + df['Q60'] + df['Q61'] -  df['Q62'])
```

8. Dropping demographics & Script 9 blanks

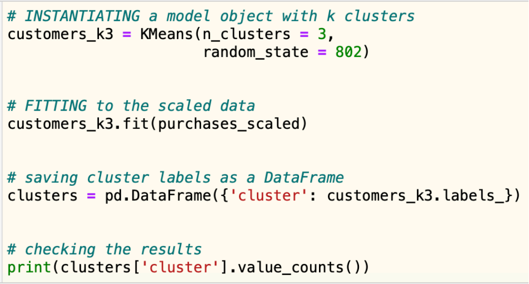


```
# dropping demographic information
purchase_behavior = customers_df.drop(['Channel', 'Region'],
                                      axis = 1)


# INSTANTIATING a StandardScaler() object
scaler = StandardScaler()


# FITTING the scaler with the data
scaler.fit(purchase_behavior)


# TRANSFORMING our data after fit
X_scaled = scaler.transform(purchase_behavior)


# converting scaled data into a DataFrame
purchases_scaled = pd.DataFrame(X_scaled)


# reattaching column names
purchases_scaled.columns = purchase_behavior.columns


# checking pre- and post-scaling variance
print(pd.np.var(purchase_behavior), '\n\n')
print(pd.np.var(purchases_scaled))
```



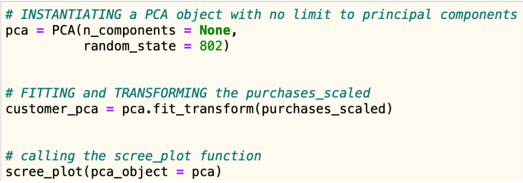
```
# calling the inertia_plot() function
interia_plot(data = purchases_scaled)
```

k-Means Clustering: If we know how many clusters we would like to build, we can take advantage of k-means clustering. This is a more robust way to create clusters and is also a technique that can be used to predict on new data.

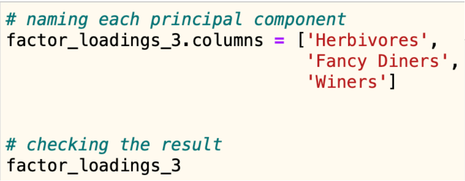


```
# INSTANTIATING a model object with k clusters
customers_k3 = KMeans(n_clusters = 3,
                      random_state = 802)


# FITTING to the scaled data
customers_k3.fit(purchases_scaled)


# saving cluster labels as a DataFrame
clusters = pd.DataFrame({'cluster': customers_k3.labels_})


# checking the results
print(clusters['cluster'].value_counts())
```

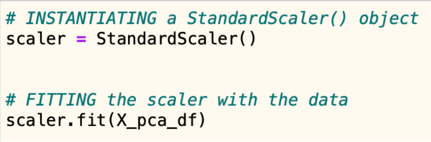
9. Script 10



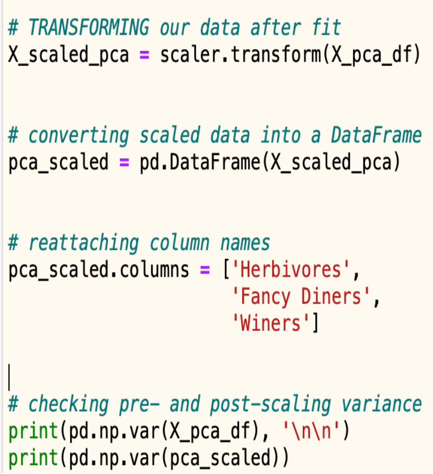
```
# INSTANTIATING a PCA object with no limit to principal components
pca = PCA(n_components = None,
          random_state = 802)


# FITTING and TRANSFORMING the purchases_scaled
customer_pca = pca.fit_transform(purchases_scaled)


# calling the scree_plot function
scree_plot(pca_object = pca)
```



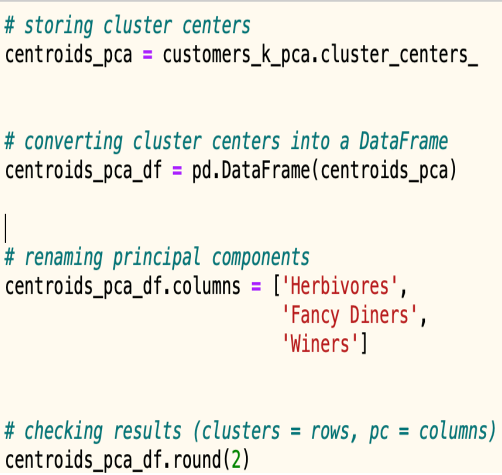
```
# naming each principal component
factor_loadings_3.columns = ['Herbivores',
                             'Fancy Diners',
                             'Winers']


# checking the result
factor_loadings_3
```



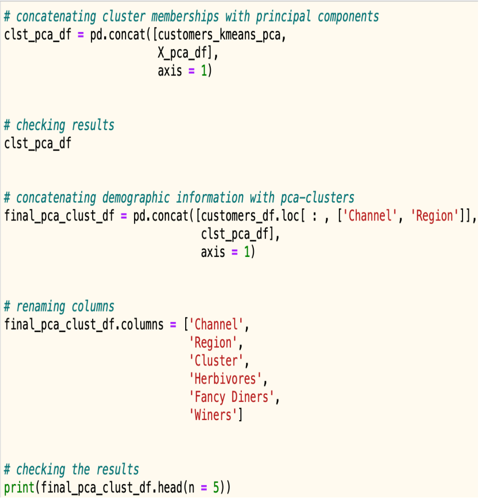
```
# INSTANTIATING a StandardScaler() object
scaler = StandardScaler()


# FITTING the scaler with the data
scaler.fit(X_pca_df)
```



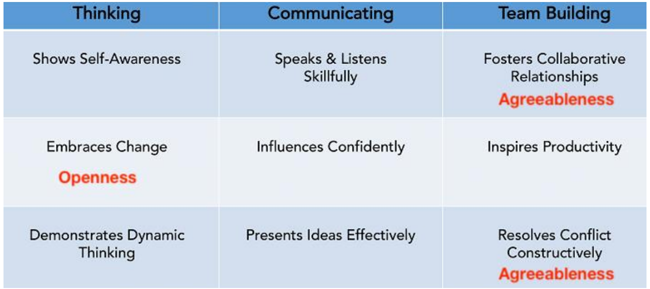
```
# TRANSFORMING our data after fit
X_scaled_pca = scaler.transform(X_pca_df)


# converting scaled data into a DataFrame
pca_scaled = pd.DataFrame(X_scaled_pca)


# reattaching column names
pca_scaled.columns = ['Herbivores',
                      'Fancy Diners',
                      'Winers']


# checking pre- and post-scaling variance
print(pd.np.var(X_pca_df), '\n\n')
print(pd.np.var(pca_scaled))
```



```
# storing cluster centers
centroids_pca = customers_k_pca.cluster_centers_


# converting cluster centers into a DataFrame
centroids_pca_df = pd.DataFrame(centroids_pca)


# renaming principal components
centroids_pca_df.columns = ['Herbivores',
                            'Fancy Diners',
                            'Winers']


# checking results (clusters = rows, pc = columns)
centroids_pca_df.round(2)
```



```
# concatenating cluster memberships with principal components
clst_pca_df = pd.concat([customers_kmeans_pca,
                         X_pca_df],
                         axis = 1)


# checking results
clst_pca_df


# concatenating demographic information with pca-clusters
final_pca_clust_df = pd.concat([customers_df.loc[ : , ['Channel', 'Region']],
                                clst_pca_df],
                                axis = 1)


# renaming columns
final_pca_clust_df.columns = ['Channel',
                              'Region',
                              'Cluster',
                              'Herbivores',
                              'Fancy Diners',
                              'Winers']


# checking the results
print(final_pca_clust_df.head(n = 5))
```



Thinking	Communicating	Team Building
Shows Self-Awareness	Speaks & Listens Skillfully	Fosters Collaborative Relationships Agreeableness
Embraces Change Openness	Influences Confidently	Inspires Productivity
Demonstrates Dynamic Thinking	Presents Ideas Effectively	Resolves Conflict Constructively Agreeableness