Import modules

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib
%matplotlib inline
warnings.filterwarnings('ignore')
pd.options.display.max_columns = 99
```

Loading the dataset

In [2]:
```
df = pd.read_csv("turkiye-student-evaluation_generic.csv")
df.head()
```

Out[2]:

	instr	class	nb.repeat	attendance	difficulty	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27	Q28
0	1	2	1	0	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
1	1	2	1	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
2	1	2	1	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	1	2	1	1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	1	2	1	0	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

In [3]:
```
# statistical info
df.describe()
```

Out[3]:

	instr	class	nb.repeat	attendance	difficulty	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27	Q28
count	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000	5820.000000
mean	2.465567	7.276288	1.214089	1.675601	2.783505	3.929897	3.073883	3.178064	3.063474	3.105642	3.037388	3.066321	3.041924	3.100979	3.090722	3.165648	3.035
std	1.118419	3.680175	0.632376	1.474875	1.348897	1.341077	1.282531	1.252567	1.284594	1.279989	1.292807	1.289350	1.275309	1.295443	1.305
min	1.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000
25%	1.000000	4.000000	1.000000	0.000000	1.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000
50%	3.000000	7.000000	1.000000	1.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000
75%	3.000000	10.000000	1.000000	3.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000
max	3.000000	13.000000	3.000000	4.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000

In [4]:
```
# datatype info
df.info()
```

Out[4]:
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5820 entries, 0 to 5819
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instr       5820 non-null   int64
 1   class       5820 non-null   int64
 2   nb.repeat   5820 non-null   int64
 3   attendance  5820 non-null   int64
 4   difficulty  5820 non-null   int64
 5   Q1          5820 non-null   int64
 6   Q2          5820 non-null   int64
 7   Q3          5820 non-null   int64
 8   Q4          5820 non-null   int64
 9   Q5          5820 non-null   int64
 10  Q6          5820 non-null   int64
 11  Q7          5820 non-null   int64
 12  Q8          5820 non-null   int64
 13  Q9          5820 non-null   int64
 14  Q10         5820 non-null   int64
 15  Q11         5820 non-null   int64
 16  Q12         5820 non-null   int64
 17  Q13         5820 non-null   int64
 18  Q14         5820 non-null   int64
 19  Q15         5820 non-null   int64
 20  Q16         5820 non-null   int64
 21  Q17         5820 non-null   int64
 22  Q18         5820 non-null   int64
 23  Q19         5820 non-null   int64
 24  Q20         5820 non-null   int64
 25  Q21         5820 non-null   int64
 26  Q22         5820 non-null   int64
 27  Q23         5820 non-null   int64
 28  Q24         5820 non-null   int64
 29  Q25         5820 non-null   int64
 30  Q26         5820 non-null   int64
 31  Q27         5820 non-null   int64
 32  Q28         5820 non-null   int64
dtypes: int64(33)
memory usage: 1.5 MB
```

Preprocessing the dataset

In [5]:
```
# check for null values
df.isnull().sum()
```

Out[5]:
```
instr         0
class         0
nb.repeat     0
attendance    0
difficulty    0
Q1            0
Q2            0
Q3            0
Q4            0
Q5            0
Q6            0
Q7            0
Q8            0
Q9            0
Q10           0
Q11           0
Q12           0
Q13           0
Q14           0
Q15           0
Q16           0
Q17           0
Q18           0
Q19           0
Q20           0
Q21           0
Q22           0
Q23           0
Q24           0
Q25           0
Q26           0
Q27           0
Q28           0
dtype: int64
```

Exploratory Data Analysis

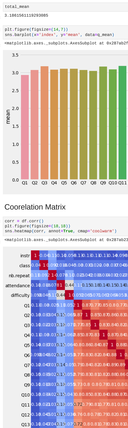
In [6]:
```
# set new style for the graph
plt.style.use("fivethirtyeight")
```

In [7]:
```
sns.countplot(df["instr"])
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x287ab1b2d90>
```

In [8]:
```
sns.countplot(df["class"])
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x287ab22e780>
```



In [9]:
```
# find mean of questions
x_questions = df.iloc[:, 5:33]
q_mean = x_questions.mean(axis=0)
total_mean = q_mean.mean()
```

In [10]:
```
q_mean = q_mean.to_frame('mean')
q_mean.reset_index(level=0, inplace=True)
q_mean.head()
```

Out[10]:

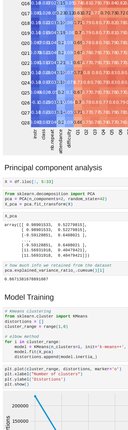
	index	mean
0	Q1	3.929897
1	Q2	3.073883
2	Q3	3.178064
3	Q4	3.063474
4	Q5	3.105642

In [11]:
```
total_mean
```

Out[11]:
```
3.1865611192919985
```

In [12]:
```
plt.figure(figsize=(14,7))
sns.barplot(x='index', y='mean', data=q_mean)
```

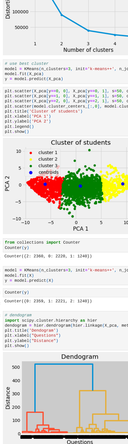
Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x287ab25e6d0>
```



Coorelation Matrix

In [13]:
```
corr = df.corr()
plt.figure(figsize=(18,18))
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x287ab23ed40>
```



Principal component analysis

In [14]:
```
X = df.iloc[:, 5:33]
```

In [15]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2, random_state=42)
X_pca = pca.fit_transform(X)
```

In [16]:
```
X_pca
```

Out[16]:
```
array([[ 0.90001533,  0.52278915],
       [ 0.90001533,  0.52278915],
       [-0.59128951,  0.6480021 ],
       ...,
       [-0.59128951,  0.6480021 ],
       [11.58031918,  0.48479421],
       [11.58031918,  0.48479421]])
```

In [17]:
```
# how much info we retained from the dataset
pca.explained_variance_ratio_.cumsum()[1]
```

Out[17]:
```
0.8671703478991087
```

Model Training

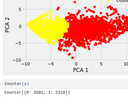
In [18]:
```
# KMeans clustering
from sklearn.cluster import KMeans
distortions = []
cluster_range = range(1,6)

# elbow method
for i in cluster_range:
    model = KMeans(n_clusters=i, init='k-means++', n_jobs=-1, random_state=42)
    model.fit(X_pca)
    distortions.append(model.inertia_)
```

In [19]:
```
plt.plot(cluster_range, distortions, marker='o')
plt.xlabel("Number of clusters")
plt.ylabel("Distortions")
plt.show()
```


In [20]:
```
# use best cluster
model = KMeans(n_clusters=3, init='k-means++', n_jobs=-1, random_state=42)
model.fit(X_pca)
y = model.predict(X_pca)
```

In [21]:
```
plt.scatter(X_pca[y==0, 0], X_pca[y==0, 1], s=50, c='red', label='cluster 1')
plt.scatter(X_pca[y==1, 0], X_pca[y==1, 1], s=50, c='blue', label='cluster 2')
plt.scatter(X_pca[y==2, 0], X_pca[y==2, 1], s=50, c='green', label='cluster 3')
plt.scatter(model.cluster_centers_[:, 0], model.cluster_centers_[:, 1], s=200, c='blue', label='centroids')
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.legend()
plt.show()
```



In [22]:
```
from collections import Counter
Counter(y)
```

Out[22]:
```
Counter({2: 2366, 0: 2226, 1: 1228})
```

In [23]:
```
model = KMeans(n_clusters=3, init='k-means++', n_jobs=-1, random_state=42)
model.fit(X_pca)
y = model.predict(X_pca)
```

In [24]:
```
Counter(y)
```

Out[24]:
```
Counter({0: 2359, 1: 2221, 2: 1240})
```

In [25]:
```
# dendogram
import scipy.cluster.hierarchy as hier
dendogram = hier.dendrogram(hier.linkage(X_pca, method='ward'))
plt.title("Dendogram")
plt.xlabel("Questions")
plt.ylabel("Distance")
plt.show()
```


In [26]:
```
from sklearn.cluster import AgglomerativeClustering
model = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
y = model.fit_predict(X_pca)
```

In [27]:
```
plt.scatter(X_pca[y==0, 0], X_pca[y==0, 1], s=50, c='red', label='cluster 1')
plt.scatter(X_pca[y==1, 0], X_pca[y==1, 1], s=50, c='yellow', label='cluster 2')
plt.title("Cluster of students")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.legend()
plt.show()
```


In [28]:
```
Counter(y)
```

Out[28]:
```
Counter({0: 3562, 1: 2258})
```