In [21]:

```
import pandas as pd
import numpy as np
data=pd.read_csv(r"C:\Users\Iyas Hussain\Downloads\AI-Data.csv")
data.head(60)
```

Out[21]:

| | gender | NationalITy | PlaceofBirth | StageID | GradeID | SectionID | Topic | Semester | R |
|---|---|---|---|---|---|---|---|---|---|
| **0** | M | KW | KuwaIT | lowerlevel | G-04 | A | IT | F | |
| **1** | M | KW | KuwaIT | lowerlevel | G-04 | A | IT | F | |
| **2** | M | KW | KuwaIT | lowerlevel | G-04 | A | IT | F | |
| **3** | M | KW | KuwaIT | lowerlevel | G-04 | A | IT | F | |
| **4** | M | KW | KuwaIT | lowerlevel | G-04 | A | IT | F | |
| **5** | F | KW | KuwaIT | lowerlevel | G-04 | A | IT | F | |
| **6** | M | KW | KuwaIT | MiddleSchool | G-07 | A | Math | F | |
| **7** | M | KW | KuwaIT | MiddleSchool | G-07 | A | Math | F | |
| **8** | F | KW | KuwaIT | MiddleSchool | G-07 | A | Math | F | |
| **9** | F | KW | KuwaIT | MiddleSchool | G-07 | B | IT | F | |
| **10** | M | KW | KuwaIT | MiddleSchool | G-07 | A | Math | F | |
| **11** | M | KW | KuwaIT | MiddleSchool | G-07 | B | Math | F | |
| **12** | M | KW | KuwaIT | lowerlevel | G-04 | A | IT | F | |
| **13** | M | lebanon | lebanon | MiddleSchool | G-08 | A | Math | F | |
| **14** | F | KW | KuwaIT | MiddleSchool | G-08 | A | Math | F | |
| **15** | F | KW | KuwaIT | MiddleSchool | G-06 | A | IT | F | |
| **16** | M | KW | KuwaIT | MiddleSchool | G-07 | B | IT | F | |
| **17** | M | KW | KuwaIT | MiddleSchool | G-07 | A | Math | F | |
| **18** | F | KW | KuwaIT | MiddleSchool | G-07 | A | IT | F | |
| **19** | M | KW | KuwaIT | MiddleSchool | G-07 | B | IT | F | |
| **20** | F | KW | KuwaIT | MiddleSchool | G-07 | A | IT | F | |
| **21** | F | KW | KuwaIT | MiddleSchool | G-07 | B | IT | F | |
| **22** | M | KW | KuwaIT | MiddleSchool | G-07 | A | IT | F | |
| **23** | M | KW | KuwaIT | MiddleSchool | G-07 | A | IT | F | |
| **24** | M | KW | KuwaIT | MiddleSchool | G-07 | B | IT | F | |
| **25** | M | KW | KuwaIT | MiddleSchool | G-07 | A | IT | F | |
| **26** | M | KW | KuwaIT | MiddleSchool | G-07 | B | IT | F | |
| **27** | M | KW | KuwaIT | MiddleSchool | G-08 | A | Arabic | F | |
| **28** | M | KW | KuwaIT | MiddleSchool | G-08 | A | Science | F | |
| **29** | F | KW | KuwaIT | MiddleSchool | G-08 | A | Arabic | F | |
| **30** | F | KW | KuwaIT | MiddleSchool | G-08 | A | Arabic | F | |
| **31** | M | KW | KuwaIT | MiddleSchool | G-07 | A | IT | F | |
| **32** | F | KW | KuwaIT | lowerlevel | G-07 | A | IT | F | |
| **33** | M | KW | KuwaIT | lowerlevel | G-05 | A | English | F | |
| **34** | M | KW | KuwaIT | MiddleSchool | G-07 | B | Science | F | |
| **35** | M | KW | KuwaIT | MiddleSchool | G-07 | A | English | F | |
| **36** | M | KW | KuwaIT | MiddleSchool | G-07 | B | Science | F | |

| | gender | NationalITy | PlaceofBirth | StageID | GradeID | SectionID | Topic | Semester | R |
|---|---|---|---|---|---|---|---|---|---|
| **37** | F | Egypt | Egypt | MiddleSchool | G-07 | A | IT | F | |
| **38** | M | KW | KuwaIT | MiddleSchool | G-06 | A | IT | F | |
| **39** | F | SaudiArabia | SaudiArabia | MiddleSchool | G-07 | B | Science | F | |
| **40** | F | KW | KuwaIT | MiddleSchool | G-07 | A | IT | F | |
| **41** | M | KW | KuwaIT | MiddleSchool | G-07 | A | IT | F | |
| **42** | M | KW | KuwaIT | HighSchool | G-09 | A | IT | F | |
| **43** | F | KW | KuwaIT | HighSchool | G-09 | A | IT | F | |
| **44** | F | KW | KuwaIT | HighSchool | G-09 | A | IT | F | |
| **45** | M | KW | KuwaIT | MiddleSchool | G-07 | A | Quran | F | |
| **46** | M | KW | KuwaIT | lowerlevel | G-05 | A | English | F | |
| **47** | F | KW | KuwaIT | HighSchool | G-12 | A | English | F | |
| **48** | F | KW | KuwaIT | HighSchool | G-12 | A | English | F | |
| **49** | F | KW | KuwaIT | HighSchool | G-12 | A | English | F | |
| **50** | F | KW | KuwaIT | HighSchool | G-12 | A | English | F | |
| **51** | M | KW | KuwaIT | HighSchool | G-12 | A | English | F | |
| **52** | F | KW | KuwaIT | HighSchool | G-12 | A | English | F | |
| **53** | F | KW | KuwaIT | HighSchool | G-11 | A | Science | F | |
| **54** | M | KW | KuwaIT | HighSchool | G-12 | A | English | F | |
| **55** | M | KW | KuwaIT | MiddleSchool | G-07 | A | Math | F | |
| **56** | M | USA | USA | MiddleSchool | G-08 | B | Math | F | |
| **57** | M | KW | KuwaIT | MiddleSchool | G-07 | A | Math | F | |
| **58** | M | KW | KuwaIT | MiddleSchool | G-07 | A | Math | F | |
| **59** | F | USA | USA | MiddleSchool | G-07 | A | Math | F | |

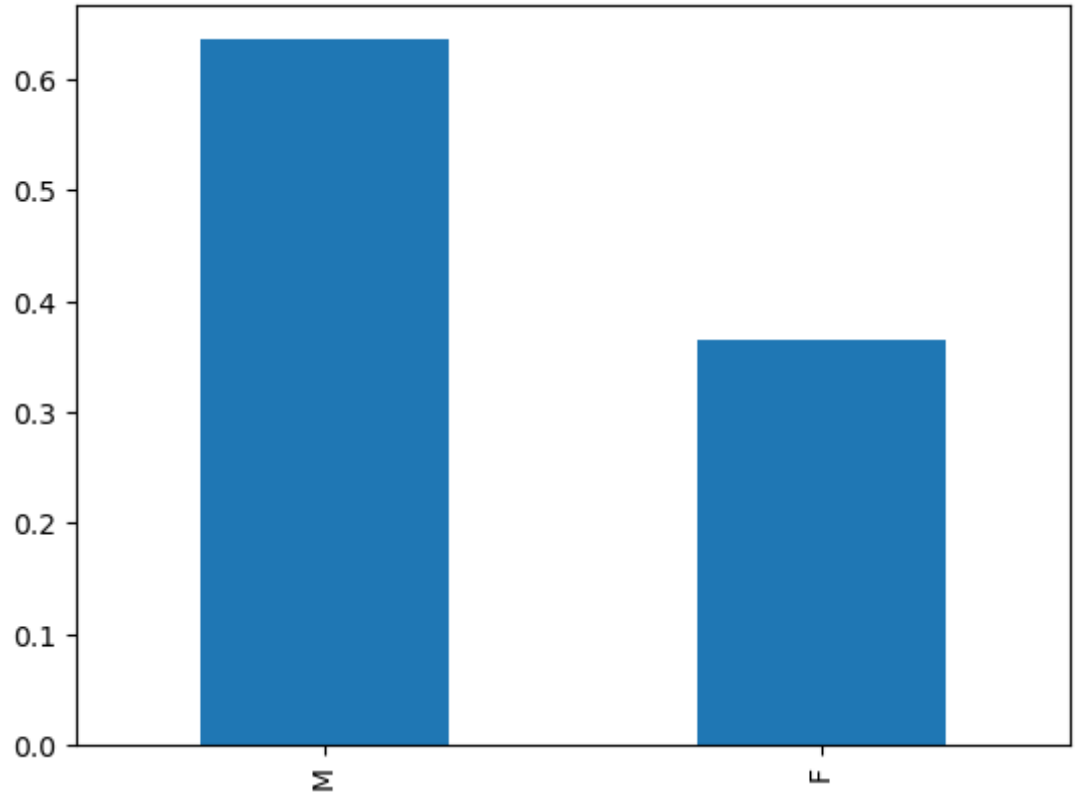
In [28]:

```
##Gender
print('percentage',data.gender.value_counts (normalize=True))
data.gender.value_counts(normalize=True).plot(kind='bar')
```

```
percentage M    0.635417
F    0.364583
Name: gender, dtype: float64
```

Out[28]:

```
<Axes: >
```




In [63]:

```
print()
```

```
(480, 17)
```

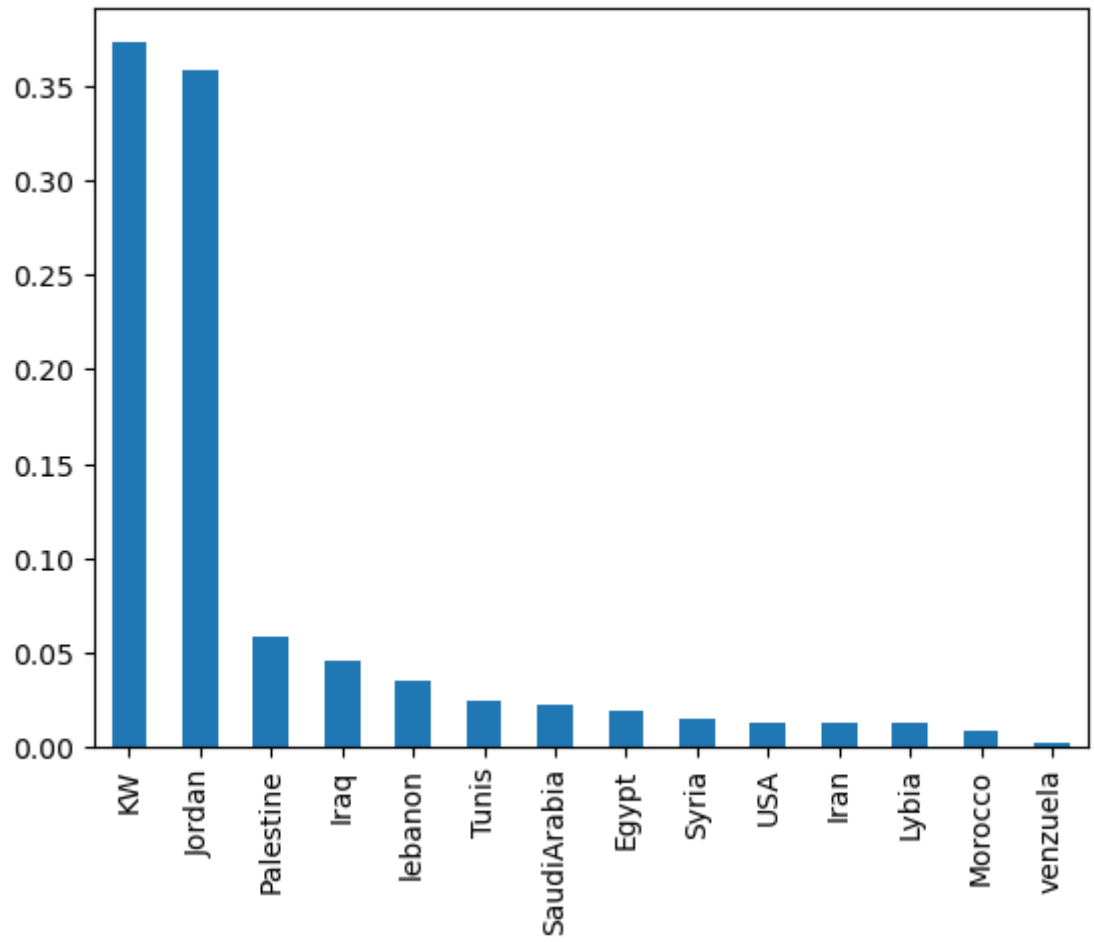
In [68]:

```
##Nationality
print('percentage',data.NationalITy.value_counts (normalize=True))
data.NationalITy.value_counts(normalize=True).plot(kind='bar')
```

```
percentage KW             0.372917
Jordan         0.358333
Palestine      0.058333
Iraq           0.045833
lebanon        0.035417
Tunis          0.025000
SaudiArabia    0.022917
Egypt          0.018750
Syria          0.014583
USA            0.012500
Iran           0.012500
Lybia          0.012500
Morocco        0.008333
venzuela       0.002083
Name: NationalITy, dtype: float64
```

Out[68]:

```
<Axes: >
```




print('percentage',data.GradeID.value_counts (normalize=True))
data.GradeID.value_counts(normalize=True).plot(kind='bar')

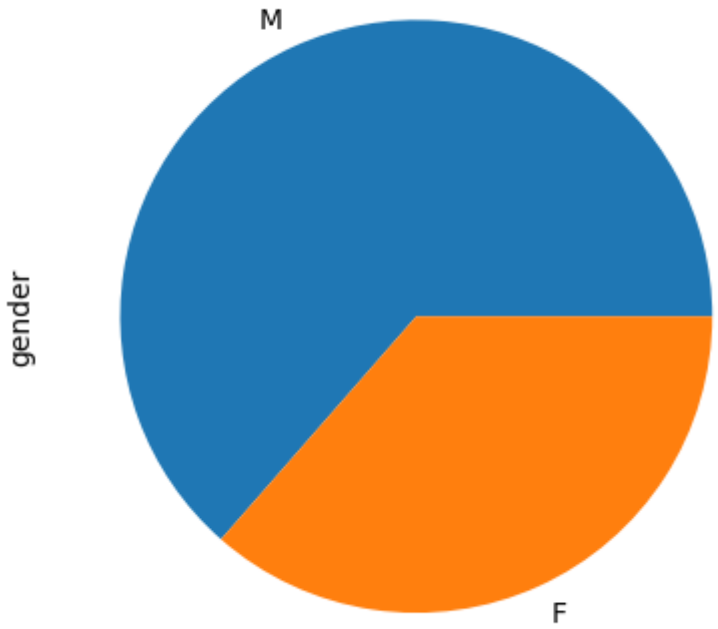
In [35]:

```
print('percentage',data.GradeID.value_counts (normalize=True))
data.gender.value_counts(normalize=True).plot(kind='pie')
```

```
percentage G-02    0.306250
G-08    0.241667
G-07    0.210417
G-04    0.100000
G-06    0.066667
G-11    0.027083
G-12    0.022917
G-09    0.010417
G-10    0.008333
G-05    0.006250
Name: GradeID, dtype: float64
```

Out[35]:

```
<Axes: ylabel='gender'>
```



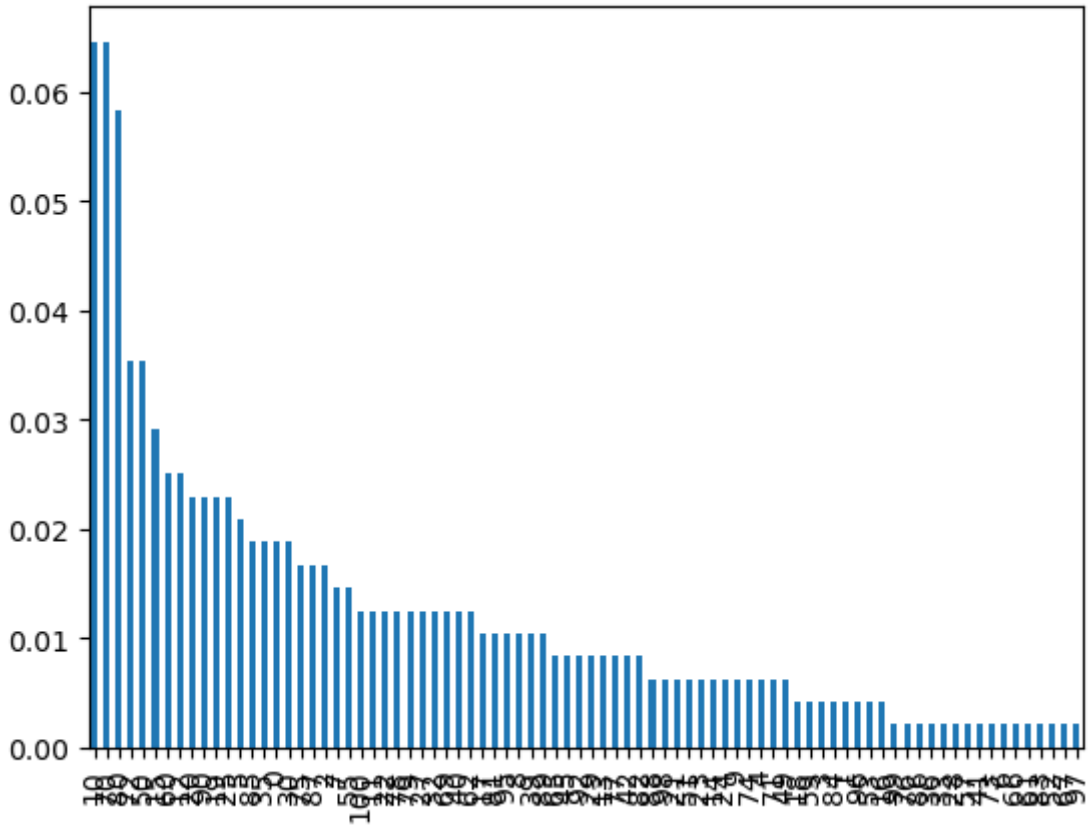
In [69]:

```
print('percentage',data.raisedhands.value_counts (normalize=True))
data.raisedhands.value_counts(normalize=True).plot(kind='bar')
```

```
percentage 10    0.064583
70    0.064583
80    0.058333
72    0.035417
50    0.035417
        ...
61    0.002083
83    0.002083
52    0.002083
67    0.002083
97    0.002083
Name: raisedhands, Length: 82, dtype: float64
```

Out[69]:

<Axes: >



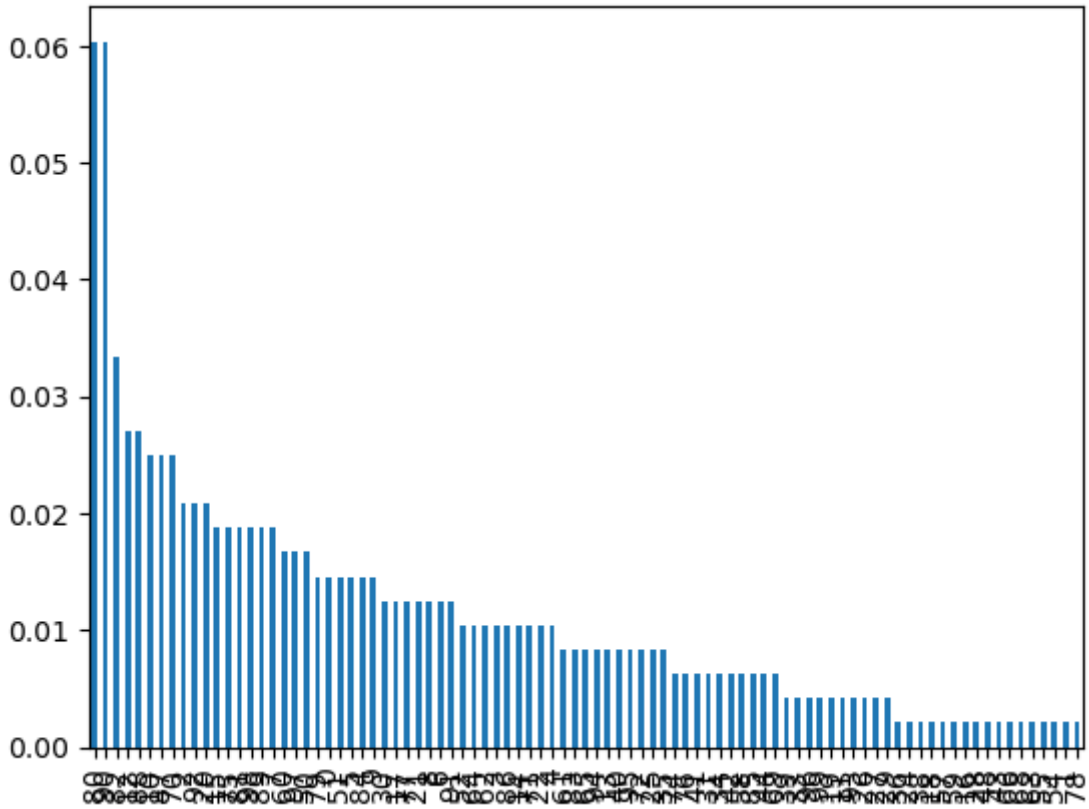
In [70]:

```
print('percentage',data.VisITedResources.value_counts (normalize=True))
data.VisITedResources.value_counts(normalize=True).plot(kind='bar')
```

```
percentage 80    0.060417
90    0.060417
82    0.033333
12    0.027083
88    0.027083
        ...
63    0.002083
55    0.002083
54    0.002083
1     0.002083
78    0.002083
Name: VisITedResources, Length: 89, dtype: float64
```

Out[70]:

<Axes: >



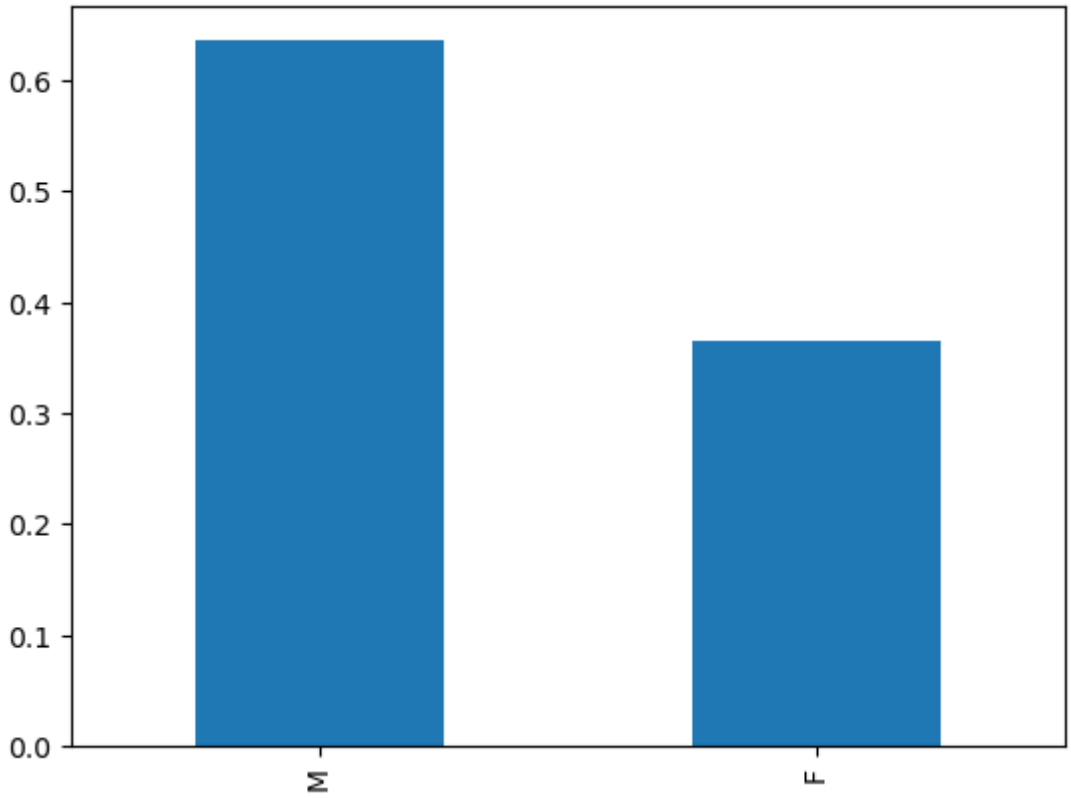
In [41]:

```
print('percentage',data.AnnouncementsView.value_counts (normalize=True))
data.gender.value_counts(normalize=True).plot(kind='bar')
```

```
percentage 12    0.043750
42    0.033333
50    0.033333
40    0.033333
2     0.029167
        ...
93    0.002083
17    0.002083
24    0.002083
91    0.002083
78    0.002083
Name: AnnouncementsView, Length: 88, dtype: float64
```

Out[41]:

```
<Axes: >
```



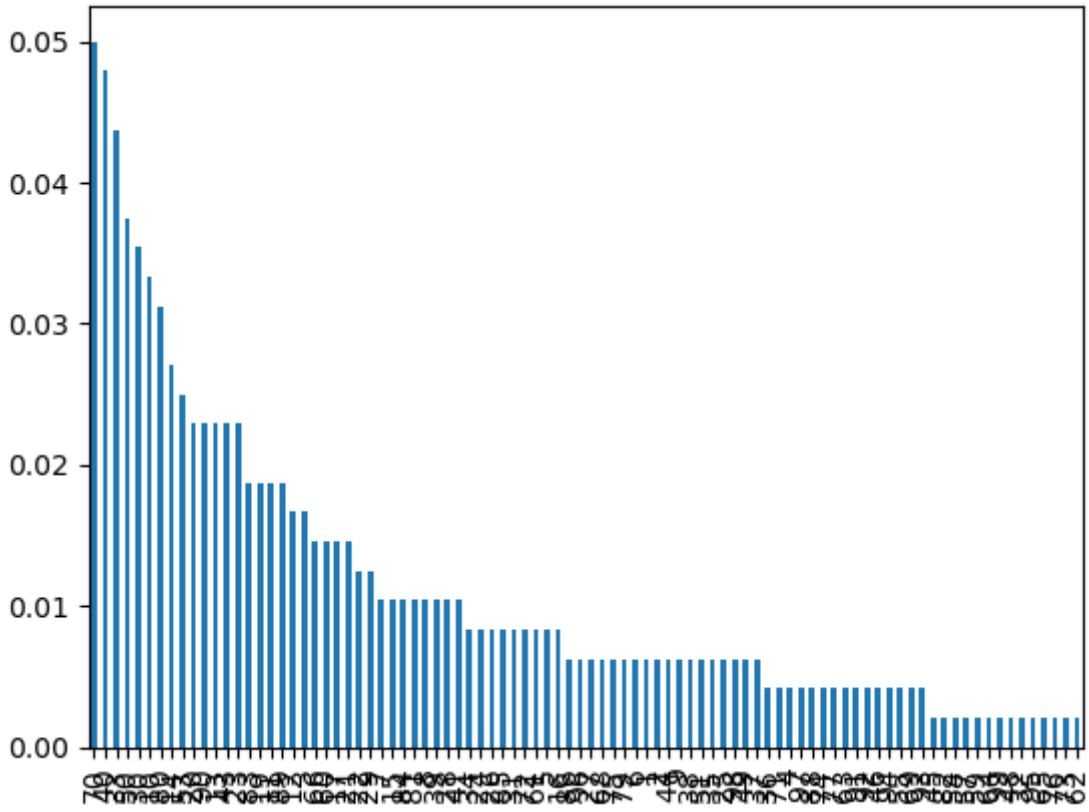
In [71]:

```
print('percentage',data.Discussion.value_counts (normalize=True))
data.Discussion.value_counts(normalize=True).plot(kind='bar')
```

```
percentage 70    0.050000
40    0.047917
33    0.043750
50    0.037500
30    0.035417
        ...
95    0.002083
65    0.002083
76    0.002083
73    0.002083
62    0.002083
Name: Discussion, Length: 90, dtype: float64
```

Out[71]:

<Axes: >



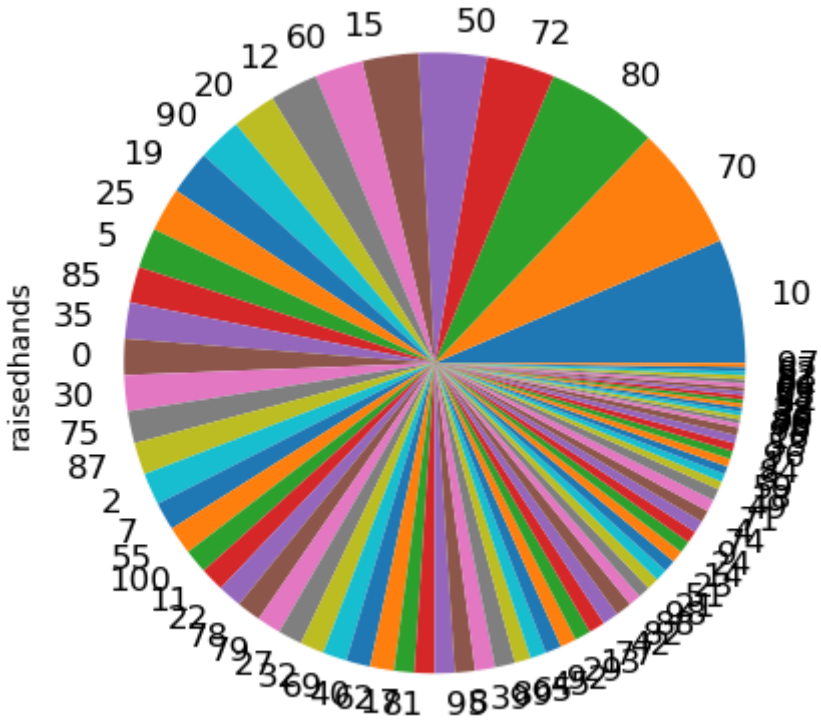
In [79]:

```
print('percentage',data.raisedhands.value_counts (normalize=True))
data.raisedhands.value_counts(normalize=True).plot(kind='pie',figsize=(5,10),fontsize =12
```

```
percentage 10    0.064583
70    0.064583
80    0.058333
72    0.035417
50    0.035417
        ...
61    0.002083
83    0.002083
52    0.002083
67    0.002083
97    0.002083
Name: raisedhands, Length: 82, dtype: float64
```

Out[79]:

```
<Axes: ylabel='raisedhands'>
```



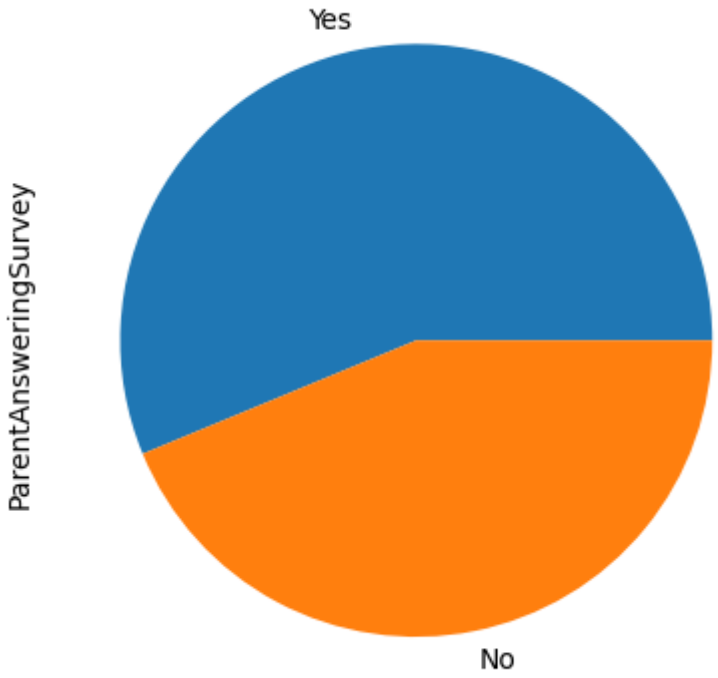
In [72]:

```
print('percentage',data.ParentAnsweringSurvey.value_counts (normalize=True))
data.ParentAnsweringSurvey.value_counts(normalize=True).plot(kind='pie')
```

```
percentage Yes    0.5625
No     0.4375
Name: ParentAnsweringSurvey, dtype: float64
```

Out[72]:

```
<Axes: ylabel='ParentAnsweringSurvey'>
```



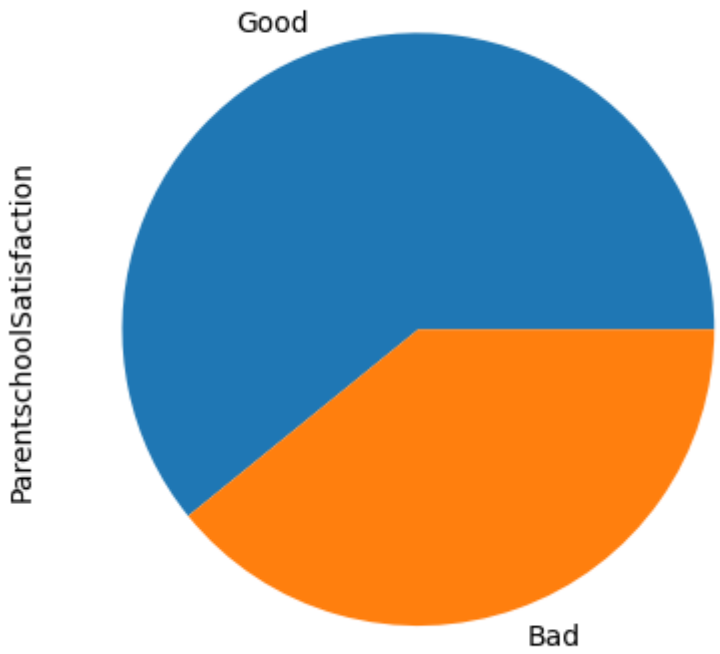
In [81]:

```
+95]poiuytrQ
```

```
percentage Good    0.608333
Bad     0.391667
Name: ParentschoolSatisfaction, dtype: float64
```

Out[81]:

```
<Axes: ylabel='ParentschoolSatisfaction'>
```

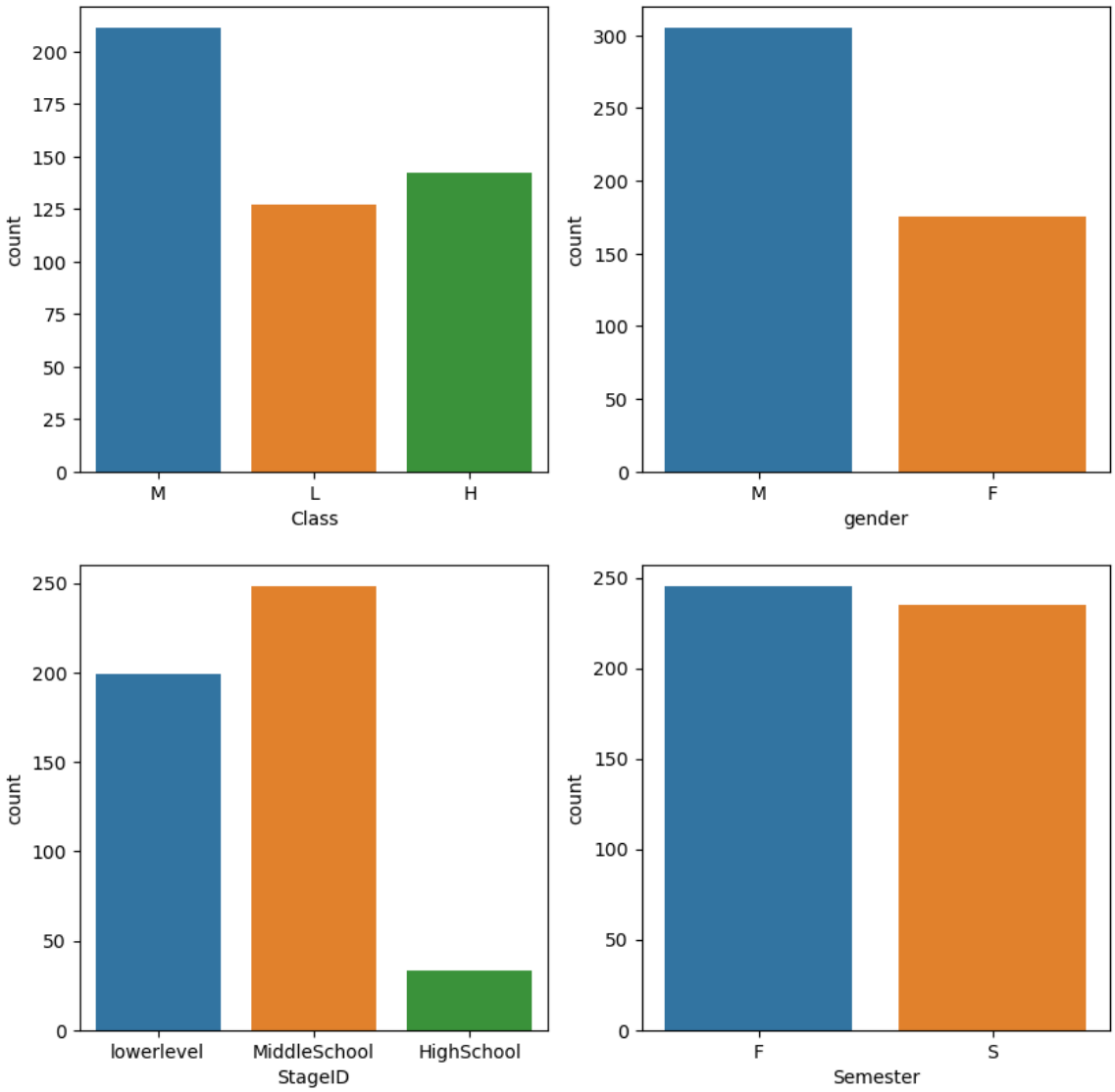


In [87]:

```
import seaborn as sns
fig, axarr =plt.subplots(2,2,figsize=(10,10))
sns.countplot(x='Class', data=data, ax=axarr[0,0])
sns.countplot(x='gender', data=data, ax=axarr[0,1])
sns.countplot(x='StageID', data=data, ax=axarr[1,0])
sns.countplot(x='Semester', data=data, ax=axarr[1,1])
```

Out[87]:

```
<Axes: xlabel='Semester', ylabel='count'>
```

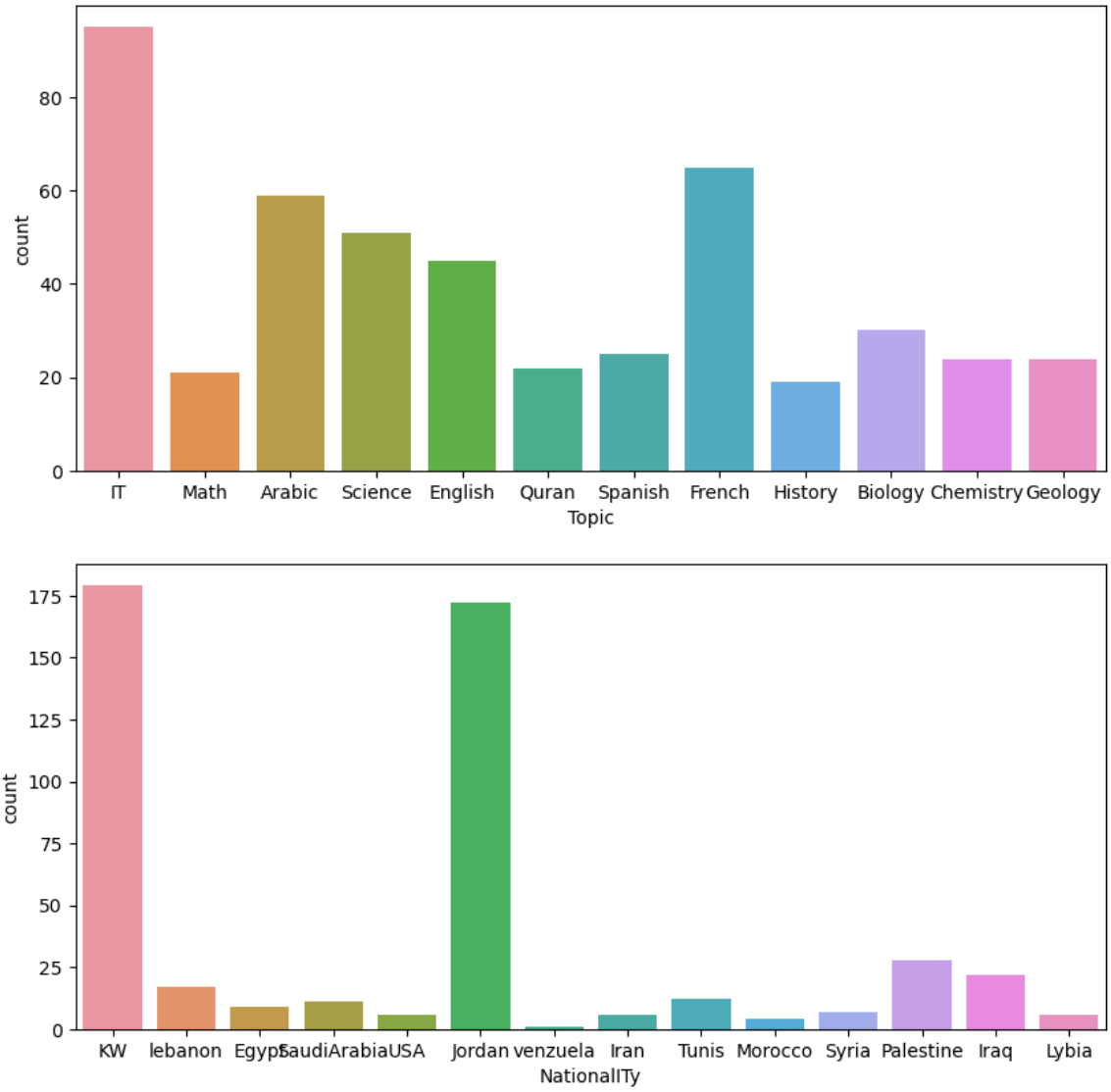


In [88]:

```
fig, axarr =plt.subplots(2,1,figsize=(10,10))
sns.countplot(x='Topic', data=data, ax=axarr[0])
sns.countplot(x='NationalITy', data=data, ax=axarr[1])
```

Out[88]:

```
<Axes: xlabel='NationalITy', ylabel='count'>
```

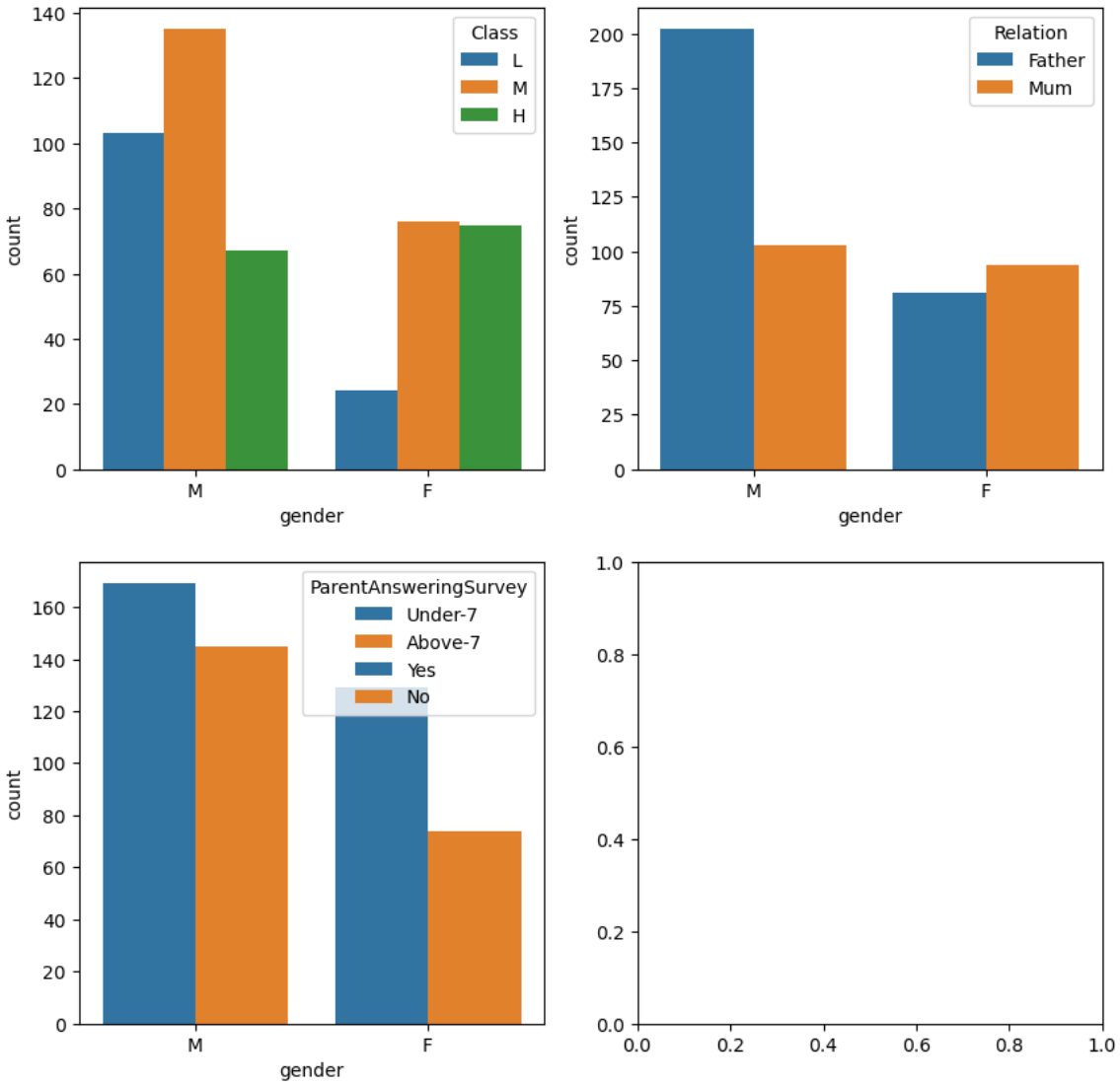


In [94]:

```
sns.countplot(x='gender', hue="Class", data=data, ax=axarr[0,0], order=['M',"F"], hue_orc
sns.countplot(x='gender', hue="Relation", data=data, ax=axarr[0,1], order=['M',"F"])
sns.countplot(x='gender', hue="StudentAbsenceDays", data=data, ax=axarr[1,0], order=['M',
sns.countplot(x='gender', hue="ParentAnsweringSurvey", data=data, ax=axarr[1,0],order= ['
```

Out[94]:

```
<Axes: xlabel='gender', ylabel='count'>
```

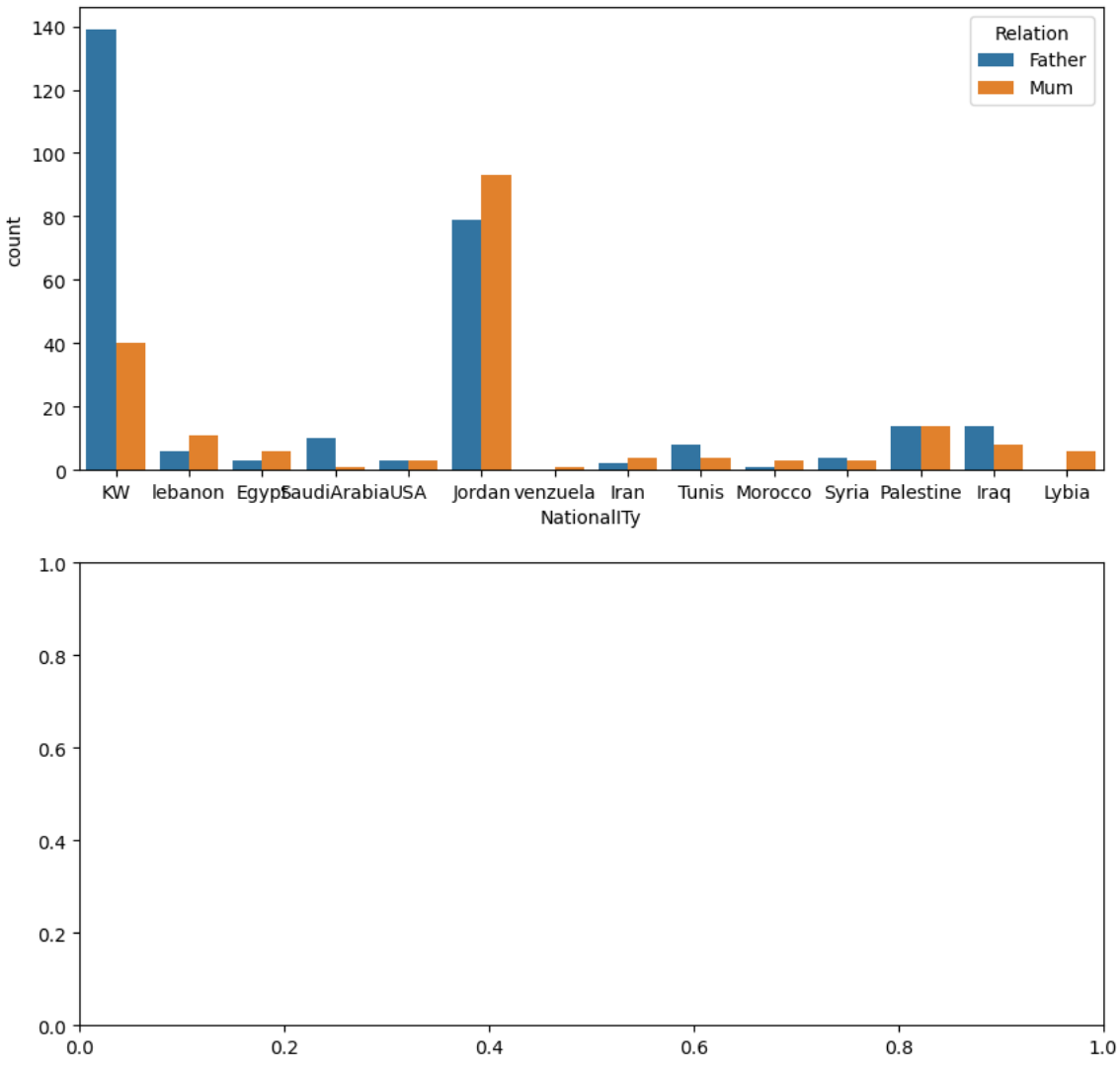


In [110]:

```
fig, axarr =plt.subplots(2,1,figsize=(10,10))
sns.countplot(x='NationalITy', hue="Relation", data=data, ax=axarr[0])
```

Out[110]:

```
<Axes: xlabel='NationalITy', ylabel='count'>
```



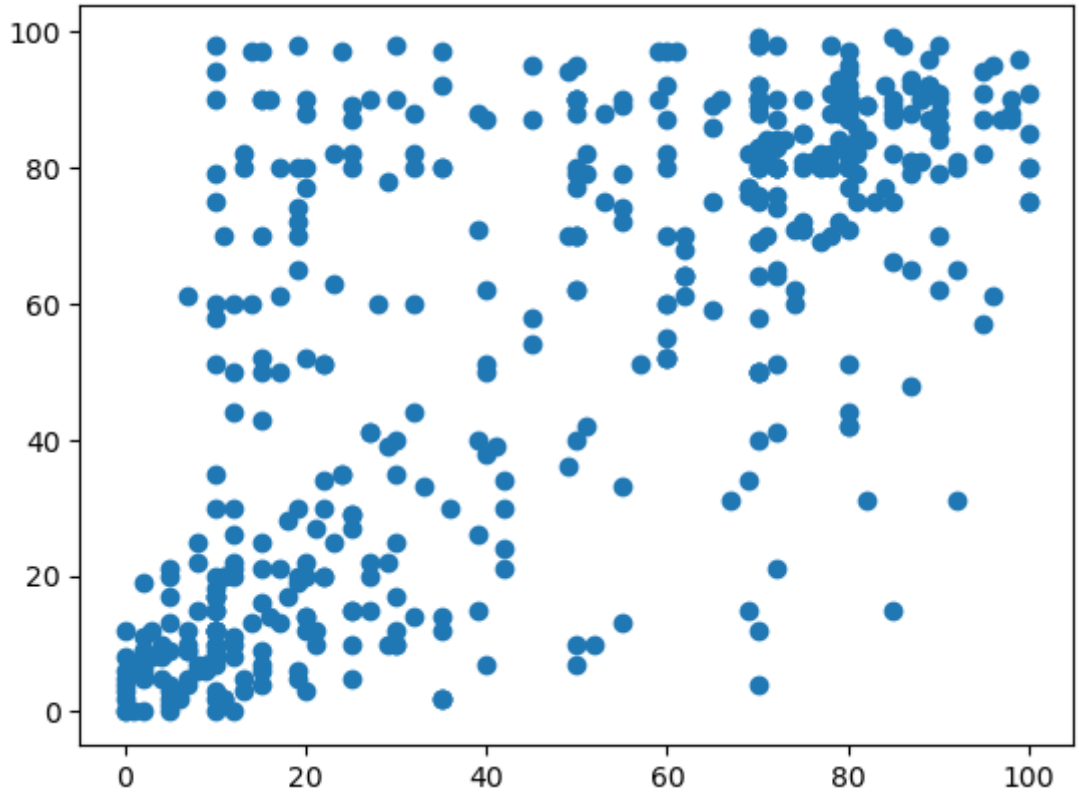

In [ ]:

In [55]:

```
import matplotlib.pyplot as plt
plt.scatter(df['raisedhands'],df['VisITedResources'])
```

Out[55]:

```
<matplotlib.collections.PathCollection at 0x24c49c342b0>
```




In [ ]:

```
from sklearn.cluster import KMeans
```

In [61]:

```
wcss = []

for i in the range{0,10}
    km=KMeans(n_clusters=i)
    km.fit_predict(df)
    wcss.append(km.inertia_)
```

```
  Cell In[61], line 3
    for i in the range{0,10}
                      ^
SyntaxError: invalid syntax
```

In [57]:

```
wcss
```

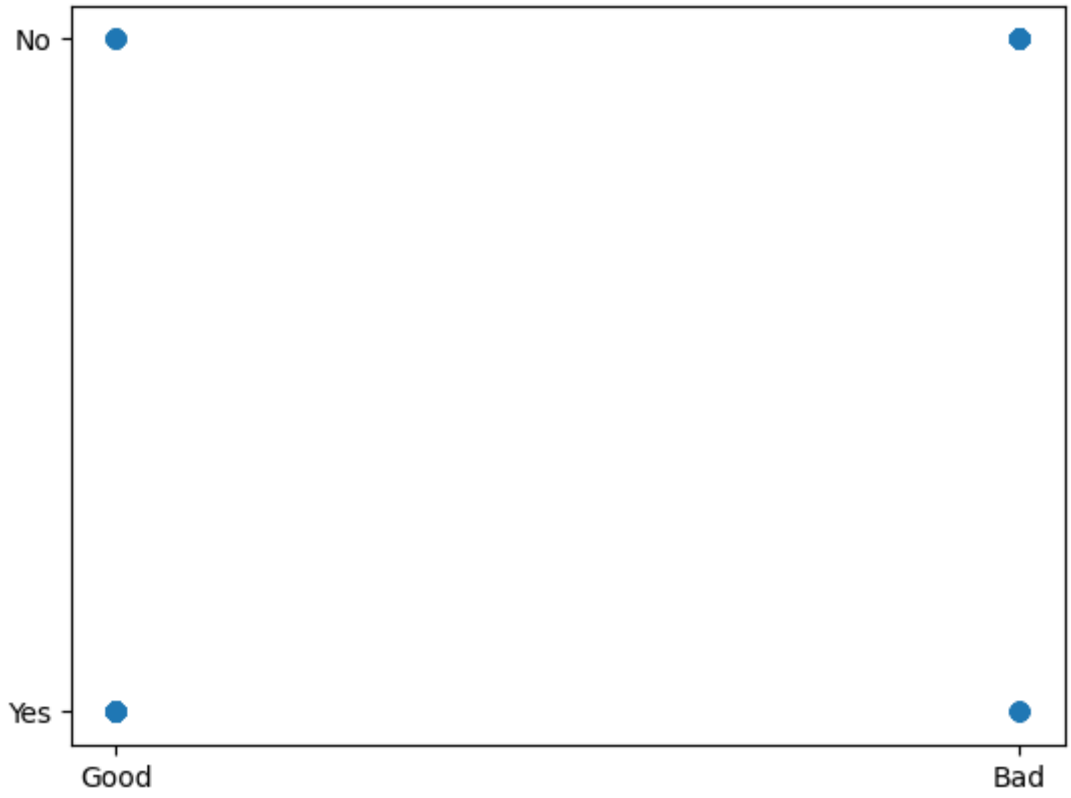
```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
Cell In[57], line 1
----> 1 wcss

NameError: name 'wcss' is not defined
```

In [62]:

```
import matplotlib.pyplot as plt
plt.scatter(df['ParentschoolSatisfaction'],df['ParentAnsweringSurvey'])
```

Out[62]:

```
<matplotlib.collections.PathCollection at 0x24c49a89300>
```



In [*]:

```
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import time as t
import sklearn.utils as u
import sklearn.preprocessing as pp
import sklearn.tree as tr
import sklearn.ensemble as es
import sklearn.metrics as m
import sklearn.linear_model as lm
import sklearn.neural_network as nn
import numpy as np
#import random as rnd
import warnings as w
w.filterwarnings('ignore')
data = pd.read_csv(r"C:\Users\Iyas Hussain\Downloads\AI-Data.csv")
ch = 0
while(ch != 10):
    print("1.Marks Class Count Graph\t2.Marks Class Semester-wise Graph\n3.Marks Class Ge
    ch = int(input("Enter Choice: "))
    if (ch == 1):
        print("Loading Graph....\n")
        t.sleep(1)
        print("\tMarks Class Count Graph")
        axes = sb.countplot(x='Class', data=data, order=['L', 'M', 'H'])
        plt.show()
    elif (ch == 2):
        print("Loading Graph....\n")
        t.sleep(1)
        print("\tMarks Class Semester-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='Semester', hue='Class', data=data, hue_order=['L', 'M', 'H'], axe
        plt.show()
    elif (ch == 3):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Gender-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='gender', hue='Class', data=data, order=['M', 'F'], hue_order=['L'
        plt.show()
    elif (ch == 4):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Nationality-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='NationalITy', hue='Class', data=data, hue_order=['L', 'M', 'H'],
        plt.show()
    elif (ch == 5):
        print("Loading Graph: \n")
        t.sleep(1)
        print("\tMarks Class Grade-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='GradeID', hue='Class', data=data, order=['G-02', 'G-04', 'G-05',
        plt.show()
    elif (ch ==6):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Section-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
```

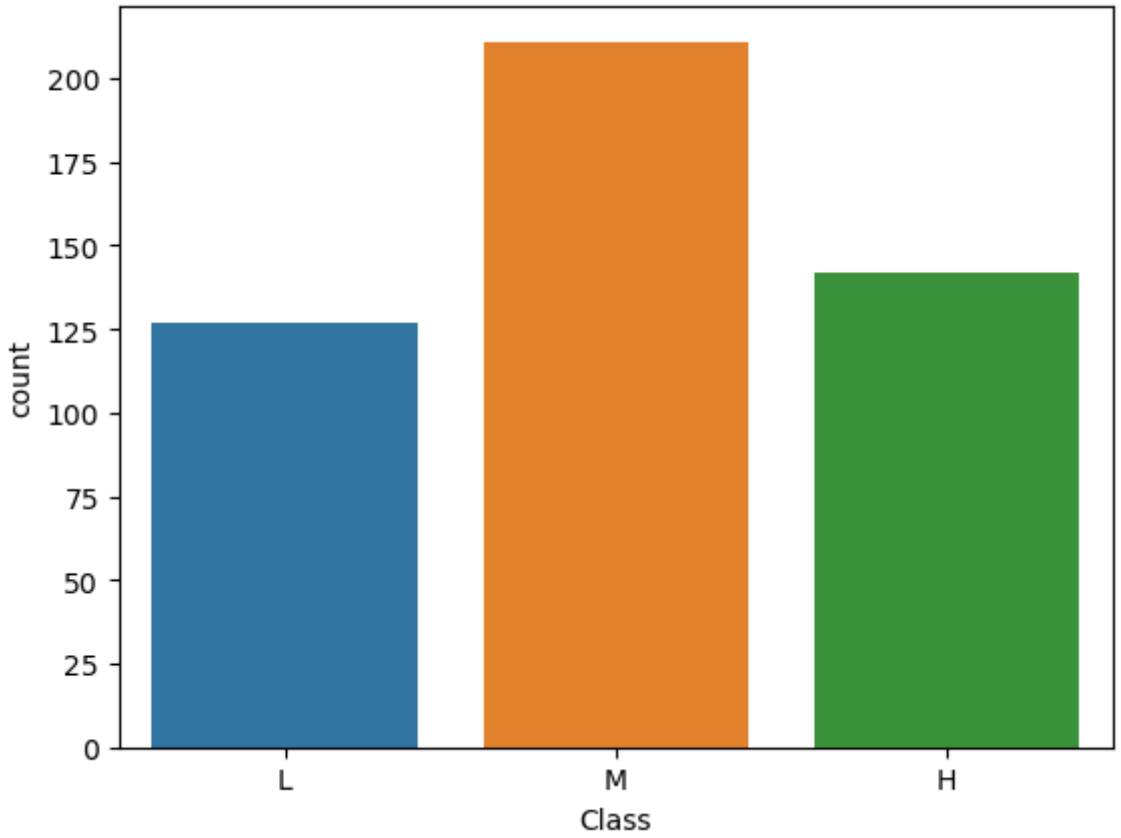
```
        sb.countplot(x='SectionID', hue='Class', data=data, hue_order = ['L', 'M', 'H'],
        plt.show()
    elif (ch == 7):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Topic-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='Topic', hue='Class', data=data, hue_order = ['L', 'M', 'H'], axes
        plt.show()
    elif (ch == 8):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Stage-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='StageID', hue='Class', data=data, hue_order = ['L', 'M', 'H'], ax
        plt.show()
    elif (ch == 9):
        print("Loading Graph..\n")
        t.sleep(1)
        print("\tMarks Class Absent Days-wise Graph")
        fig, axesarr = plt.subplots(1, figsize=(10, 6))
        sb.countplot(x='StudentAbsenceDays', hue='Class', data=data, hue_order = ['L', 'M
        plt.show()
if(ch == 10):
    print("Exiting..\n")
    t.sleep(1)
#cor = data.corr()
#print(cor)
data = data.drop("gender", axis=1)
data = data.drop("StageID", axis=1)
data = data.drop("GradeID", axis=1)
data = data.drop("NationalITy", axis=1)
data = data.drop("PlaceofBirth", axis=1)
data = data.drop("SectionID", axis=1)
data = data.drop("Topic", axis=1)
data = data.drop("Semester", axis=1)
data = data.drop("Relation", axis=1)
data = data.drop("ParentschoolSatisfaction", axis=1)
data = data.drop("ParentAnsweringSurvey", axis=1)
#data = data.drop("VisITedResources", axis=1)
data = data.drop("AnnouncementsView", axis=1)
u.shuffle(data)
countD = 0
countP = 0
countL = 0
countR = 0
countN = 0
gradeID_dict = {"G-01" : 1,
                "G-02" : 2,
                "G-03" : 3,
                "G-04" : 4,
                "G-05" : 5,
                "G-06" : 6,
                "G-07" : 7,
                "G-08" : 8,
                "G-09" : 9,
                "G-10" : 10,
                "G-11" : 11,
                "G-12" : 12}
data = data.replace({"GradeID" : gradeID_dict})
#sig = []
```

```
for column in data.columns:
    if data[column].dtype == type(object):
        le = pp.LabelEncoder()
        data[column] = le.fit_transform(data[column])
ind = int(len(data) * 0.70)
feats = data.values[:, 0:4]
lbls = data.values[:,4]
feats_Train = feats[0:ind]
feats_Test = feats[(ind+1):len(feats)]
lbls_Train = lbls[0:ind]
lbls_Test = lbls[(ind+1):len(lbls)]
modelD = tr.DecisionTreeClassifier()
modelD.fit(feats_Train, lbls_Train)
lbls_predD = modelD.predict(feats_Test)
for a,b in zip(lbls_Test, lbls_predD):
    if(a==b):
        countD += 1
accD = (countD/len(lbls_Test))
print("\nAccuracy measures using Decision Tree:")
print(m.classification_report(lbls_Test, lbls_predD),"\n")
print("\nAccuracy using Decision Tree: ", str(round(accD, 3)))
t.sleep(1)
modelR = es.RandomForestClassifier()
modelR.fit(feats_Train, lbls_Train)
lbls_predR = modelR.predict(feats_Test)
for a,b in zip(lbls_Test, lbls_predR):
    if(a==b):
        countR += 1
print("\nAccuracy Measures for Random Forest Classifier: \n")
#print("\nConfusion Matrix: \n", m.confusion_matrix(lbls_Test, lbls_predR))
print("\n", m.classification_report(lbls_Test,lbls_predR))
accR = countR/len(lbls_Test)
print("\nAccuracy using Random Forest: ", str(round(accR, 3)))
t.sleep(1)
modelP = lm.Perceptron()
modelP.fit(feats_Train, lbls_Train)
lbls_predP = modelP.predict(feats_Test)
for a,b in zip(lbls_Test, lbls_predP):
    if a == b:
        countP += 1
accP = countP/len(lbls_Test)
print("\nAccuracy measures using Linear Model Perceptron:")
print(m.classification_report(lbls_Test, lbls_predP),"\n")
print("\nAccuracy using Linear Model Perceptron: ", str(round(accP, 3)), "\n")
t.sleep(1)
modelL = lm.LogisticRegression()
modelL.fit(feats_Train, lbls_Train)
lbls_predL = modelL.predict(feats_Test)
for a,b in zip(lbls_Test, lbls_predL):
    if a == b:
        countL += 1
accL = countL/len(lbls_Test)
print("\nAccuracy measures using Linear Model Logistic Regression:")
print(m.classification_report(lbls_Test, lbls_predL),"\n")
print("\nAccuracy using Linear Model Logistic Regression: ", str(round(accP, 3)), "\n")
t.sleep(1)
modelN = nn.MLPClassifier(activation="logistic")
modelN.fit(feats_Train, lbls_Train)
lbls_predN = modelN.predict(feats_Test)
for a,b in zip(lbls_Test, lbls_predN):
    #sig.append(1/(1+ np.exp(-b)))
```

```
    if a==b:
        countN += 1
#print("\nAverage value of Sigmoid Function: ", str(round(np.average(sig), 3)))
print("\nAccuracy measures using MLP Classifier:")
print(m.classification_report(lbls_Test, lbls_predN),"\n")
accN = countN/len(lbls_Test)
print("\nAccuracy using Neural Network MLP Classifier: ", str(round(accN, 3)), "\n")
choice = input("Do you want to test specific input (y or n): ")
if(choice.lower()=="y"):
    gen = input("Enter Gender (M or F): ")
    if (gen.upper() == "M"):
        gen = 1
    elif (gen.upper() == "F"):
          gen = 0
    nat = input("Enter Nationality: ")
    pob = input("Place of Birth: ")
    gra = input("Grade ID as (G-<grade>): ")
    if(gra == "G-02"):
        gra = 2
    elif (gra == "G-04"):
        gra = 4
    elif (gra == "G-05"):
        gra = 5
    elif (gra == "G-06"):
        gra = 6
    elif (gra == "G-07"):
        gra = 7
    elif (gra == "G-08"):
        gra = 8
    elif (gra == "G-09"):
        gra = 9
    elif (gra == "G-10"):
        gra = 10
    elif (gra == "G-11"):
        gra = 11
    elif (gra == "G-12"):
        gra = 12
    sec = input("Enter Section: ")
    top = input("Enter Topic: ")
    sem = input("Enter Semester (F or S): ")
    if (sem.upper() == "F"):
        sem = 0
    elif (sem.upper() == "S"):
        sem = 1
    rel = input("Enter Relation (Father or Mum): ")
    if (rel == "Father"):
        rel = 0
    elif (rel == "Mum"):
        rel = 1
    rai = int(input("Enter raised hands: "))
    res = int(input("Enter Visited Resources: "))
    ann = int(input("Enter announcements viewed: "))
    dis = int(input("Enter no. of Discussions: "))
    sur = input("Enter Parent Answered Survey (Y or N): ")
    if (sur.upper() == "Y"):
        sur = 1
    elif (sur.upper() == "N"):
        sur = 0
    sat = input("Enter Parent School Satisfaction (Good or Bad): ")
    if (sat == "Good"):
        sat = 1
```

```
    elif (sat == "Bad"):
        sat = 0
    absc = input("Enter No. of Abscenes(Under-7 or Above-7): ")
    if (absc == "Under-7"):
        absc = 1
    elif (absc == "Above-7"):
        absc = 0
    arr = np.array([rai, res, dis, absc])
    #arr = np.array([gen, rnd.randint(0, 30), rnd.randint(0, 30), sta, gra, rnd.randint(0
    predD = modelD.predict(arr.reshape(1, -1))
    predR = modelR.predict(arr.reshape(1, -1))
    predP = modelP.predict(arr.reshape(1, -1))
    predL = modelL.predict(arr.reshape(1, -1))
    predN = modelN.predict(arr.reshape(1, -1))
    if (predD == 0):
        predD = "H"
    elif (predD == 1):
        predD = "M"
    elif (predD == 2):
        predD = "L"
    if (predR == 0):
        predR = "H"
    elif (predR == 1):
        predR = "M"
    elif (predR == 2):
        predR = "L"
    if (predP == 0):
        predP = "H"
    elif (predP == 1):
        predP = "M"
    elif (predP == 2):
        predP = "L"
    if (predL == 0):
        predL = "H"
    elif (predL == 1):
        predL = "M"
    elif (predL == 2):
        predL = "L"
    if (predN == 0):
        predN = "H"
    elif (predN == 1):
        predN = "M"
    elif (predN == 2):
        predN = "L"
    t.sleep(1)
    print("\nUsing Decision Tree Classifier: ", predD)
    t.sleep(1)
    print("Using Random Forest Classifier: ", predR)
    t.sleep(1)
    print("Using Linear Model Perceptron: ", predP)
    t.sleep(1)
    print("Using Linear Model Logisitic Regression: ", predL)
    t.sleep(1)
    print("Using Neural Network MLP Classifier: ", predN)
    print("\nExiting...")
    t.sleep(1)
else:
    print("Exiting..")
    t.sleep(1)
```

```
1.Marks Class Count Graph       2.Marks Class Semester-wise Graph
3.Marks Class Gender-wise Graph 4.Marks Class Nationality-wise Graph
5.Marks Class Grade-wise Graph  6.Marks Class Section-wise Graph
7.Marks Class Topic-wise Graph  8.Marks Class Stage-wise Graph
9.Marks Class Absent Days-wise  10.No Graph

Enter Choice: 1
Loading Graph....

        Marks Class Count Graph
```




```
1.Marks Class Count Graph       2.Marks Class Semester-wise Graph
3.Marks Class Gender-wise Graph 4.Marks Class Nationality-wise Graph
5.Marks Class Grade-wise Graph  6.Marks Class Section-wise Graph
7.Marks Class Topic-wise Graph  8.Marks Class Stage-wise Graph
9.Marks Class Absent Days-wise  10.No Graph

Enter Choice:
```

In [116]:

```
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
File C:\ProgramData\anaconda3\lib\site-packages\seaborn\palettes.py:235, i
n color_palette(palette, n_colors, desat, as_cmap)
    233 try:
    234     # Perhaps a named matplotlib colormap?
--> 235     palette = mpl_palette(palette, n_colors, as_cmap=as_cmap)
    236 except (ValueError, KeyError):  # Error class changed in mpl36

File C:\ProgramData\anaconda3\lib\site-packages\seaborn\palettes.py:406, i
n mpl_palette(name, n_colors, as_cmap)
    405 else:
--> 406     cmap = get_colormap(name)
    408 if name in MPL_QUAL_PALS:

File C:\ProgramData\anaconda3\lib\site-packages\seaborn\_compat.py:133, in
get_colormap(name)
    132 try:
--> 133     return mpl.colormaps[name]
    134 except AttributeError:

File C:\ProgramData\anaconda3\lib\site-packages\matplotlib\cm.py:82, in Co
lormapRegistry.__getitem__(self, item)
     81 except KeyError:
---> 82     raise KeyError(f"{item!r} is not a known colormap name") from
None

KeyError: "'set1' is not a known colormap name"

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call las
t)
Cell In[116], line 1
----> 1 plot = sns.countplot(x="Class", hue="Relation", data=data, order=
["L", "M", "H"], palette="set1")
      2 plot.set(xlabel='Class', ylabel='Count', title='Gender compariso
n')
      3 plt.show()

File C:\ProgramData\anaconda3\lib\site-packages\seaborn\categorical.py:294
3, in countplot(data, x, y, hue, order, hue_order, orient, color, palette,
saturation, width, dodge, ax, **kwargs)
   2940 elif x is not None and y is not None:
   2941     raise ValueError("Cannot pass values for both `x` and `y`")
-> 2943 plotter = _CountPlotter(
   2944     x, y, hue, data, order, hue_order,
   2945     estimator, errorbar, n_boot, units, seed,
   2946     orient, color, palette, saturation,
   2947     width, errcolor, errwidth, capsize, dodge
   2948 )
   2950 plotter.value_label = "count"
   2952 if ax is None:

File C:\ProgramData\anaconda3\lib\site-packages\seaborn\categorical.py:153
2, in _BarPlotter.__init__(self, x, y, hue, data, order, hue_order, estima
tor, errorbar, n_boot, units, seed, orient, color, palette, saturation, wi
dth, errcolor, errwidth, capsize, dodge)
   1529 """Initialize the plotter."""
```

```
   1530 self.establish_variables(x, y, hue, data, orient,
   1531                          order, hue_order, units)
-> 1532 self.establish_colors(color, palette, saturation)
   1533 self.estimate_statistic(estimator, errorbar, n_boot, seed)
   1535 self.dodge = dodge

File C:\ProgramData\anaconda3\lib\site-packages\seaborn\categorical.py:69
6, in _CategoricalPlotter.establish_colors(self, color, palette, saturatio
n)
    693             levels = self.hue_names
    694         palette = [palette[l] for l in levels]
--> 696     colors = color_palette(palette, n_colors)
    698 # Desaturate a bit because these are patches
    699 if saturation < 1:

File C:\ProgramData\anaconda3\lib\site-packages\seaborn\palettes.py:237, i
n color_palette(palette, n_colors, desat, as_cmap)
    235             palette = mpl_palette(palette, n_colors, as_cmap=as_cm
ap)
    236         except (ValueError, KeyError):  # Error class changed in m
pl36
--> 237             raise ValueError(f"{palette!r} is not a valid palette
name")
    239 if desat is not None:
    240     palette = [desaturate(c, desat) for c in palette]

ValueError: 'set1' is not a valid palette name


In [ ]:
```