AIT580
Survey Exploration
Group Member:

1. **IWAN (G01236399)**
2. **Ting-Yeh Yang (G01233543)**
3. **Jing-Ting Huang (G01226692)**
4. **Dohyung Lee (G01229136)**

Part 1
Identify and clean up any data items that need to be made uniform or transformed. Explain what you did

This Survey Exploration is done by using python. The detail step of each process in preparing the data will be explained in the following process:

a. **Data Extraction**

This process obtains the data from the csv file. However, in this process we only obtain the questionnaire column; since the things that we will summarize and visualize in the further step are the questionnaire data. The code for the data extraction is such the following:

```
data3 = pd.read_csv('Responses-20200407204228.csv', sep = ',')
data3.info()
data3['Unnamed: 0']
data4 = data3.drop(columns=['Unnamed: 0',
                            'Unnamed: 1',
                            'Unnamed: 2',
                            'Unnamed: 3',
                            'Unnamed: 4',
                            'Unnamed: 5',
                            'Unnamed: 6',
                            'Unnamed: 7',
                            'Unnamed: 8',
                            'Unnamed: 9',
                            'Unnamed: 10',
                            'Unnamed: 11'])
data4.info()
```

The information of each column after the extraction can be shown at following:

```
In [11]: data4.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67 entries, 0 to 66
Data columns (total 25 columns):
Q1        20 non-null float64
Q1.1      25 non-null float64
Q1.2      24 non-null object
Q2        43 non-null object
Q2.1      25 non-null object
Q2.2      1 non-null object
Q2.3      1 non-null object
Q3        65 non-null object
Q4        65 non-null object
Q5        65 non-null object
Q6        65 non-null object
Q7        65 non-null object
Q8        48 non-null object
Q8.1      21 non-null object
Q8.2      1 non-null object
Q8.3      0 non-null float64
Q9        65 non-null object
Q10       20 non-null object
Q10.1     16 non-null object
Q10.2     31 non-null object
Q11       13 non-null object
Q11.1     20 non-null object
Q11.2     24 non-null object
Q11.3     9 non-null object
Q11.4     2 non-null object
dtypes: float64(3), object(22)
memory usage: 13.2+ KB
```

b. cleansing AIT Section Column (Question 1)(JT)

In this process, we identify the AIT Section Column into 001, 004, DL1, and Not Answered to clean the data. And the results show that there are 66 rows and none of them are “Not Answered”.

```
###############################################################################
#cleansing AIT Section Column (Question 1)
###############################################################################
AIT_Section = []
i = 1
while (i < len(data4['Q1'])):

    if (not(math.isnan(pd.to_numeric(data4['Q1'][i], errors='coerce')))):
        AIT_Section.append('001')

    if (not(math.isnan(pd.to_numeric(data4['Q1.1'][i], errors='coerce')))):
        AIT_Section.append('004')

    if (not(math.isnan(pd.to_numeric(data4['Q1.2'][i], errors='coerce')))):
        AIT_Section.append('DL1')

    if (math.isnan(pd.to_numeric(data4['Q1'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q1.1'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q1.2'][i], errors='coerce'))):
        AIT_Section.append('Not Answered')

    i = i + 1
###############################################################################
len(AIT_Section)
###############################################################################
```

```
In [27]: len(AIT_Section)
Out[27]: 66
```

c. Cleansing Gender Column (Question 2)(Tim)

This process deals with the multiple-choice question 2. It merges the four fields into one field. The source code of this process is such the following:

```
Gender = []
i = 1
while (i < len(data4['Q2'])):

    if (not(math.isnan(pd.to_numeric(data4['Q2'][i], errors='coerce')))):
        Gender.append('Male')

    if (not(math.isnan(pd.to_numeric(data4['Q2.1'][i], errors='coerce')))):
        Gender.append('Female')

    if (not(math.isnan(pd.to_numeric(data4['Q2.2'][i], errors='coerce')))):
        Gender.append('NonBinary')

    if (not(math.isnan(pd.to_numeric(data4['Q2.3'][i], errors='coerce')))):
        Gender.append('Prefer not to answer')

    if (math.isnan(pd.to_numeric(data4['Q2'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q2.1'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q2.2'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q2.2'][i], errors='coerce'))):
        Gender.append('Not Answered')

    i = i + 1
```

Other than that, we also make sure that the total row of the data after this process should be the same as the total of the original row which is 66 rows.

```
In [2]: len(Gender)
Out[2]: 66
```

d. cleansing Age Column (Question 3)(JT)

In this process, we clean the age data and show the not identified data

```
################################################################################
Age = []
i = 1
while (i < len(data4['Q3'])):
    if (not(math.isnan(pd.to_numeric(data4['Q3'][i], errors='coerce')))):
        Age.append(data4['Q3'][i])
    else:
        Age.append("Not Identified")
    i = i + 1
################################################################################
len(Age)
################################################################################
```

```
In [30]: len(Age)
Out[30]: 66
```

e. cleansing Height Column (Question 4)(JT)

In this process, we clean the height data and show the not identified data.

```
################################################################################
#cleansing Height Column (Question 4)
################################################################################
Height = []
i = 1
while (i < len(data4['Q4'])):
    if (not(math.isnan(pd.to_numeric(data4['Q4'][i], errors='coerce')))):
        Height.append(data4['Q4'][i])
    else:
        Height.append("Not Identified")
    i = i + 1
################################################################################
len(Height)
################################################################################
```

```
In [31]: len(Height)
Out[31]: 66
```

f. cleansing Country of Citizenship Column (Question 5)(Tim)

This process identifies and unifies the country's name by Country_converter because there are several capitalization issues. To be specific, two citizenships come from the same country but the inputs are different because of capital, lowercase and initial.

In addition, since there is a view data which is not completed, so the blank data is substituted by "not found". The source code of this process is such the following:

```
import country_converter as coco
cc = coco.CountryConverter()

Citizenship1 = []
i = 1
while (i < len(data4['Q5'])):
    Citizenship1.append(data4['Q5'][i])
    i = i + 1

Citizenship = cc.convert(names = Citizenship1, to = 'name_short')
```

Other than that, we also make sure that the total row of the data after this process should be the same as the total of the original row which is 66 rows. The new data of this process will be such the following:

```
In [3]: len(Citizenship)
Out[3]: 66
```

g. cleansing the undergrad degree Column(Question 6)(Lee)

This process cleans and separates the undergraduate degree into 4 classifications such as the following:

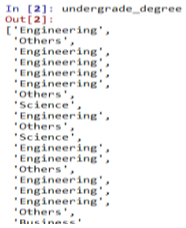
- Science
- Engineering
- Business
- Others

The code of this process is such the following:

```
undergrade_degree = []
i = 1
while (i < len(data4['Q6'])):
    if (not(pd.isnull(data4['Q6'][i]))):
        if ("math" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Science")
        elif ("stat" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Science")
        elif ("computer" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Engineering")
        elif ("engineer" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Engineering")
        elif ("electronic" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Engineering")
        elif ("cs" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Engineering")
        elif ("tech" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Engineering")
        elif ("cyber" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Engineering")
        elif ("stat" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Science")
        elif ("math" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Science")
        elif ("mathematics" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Science")
        elif ("physics" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Science")
        elif ("biology" in data4['Q6'][i].casefold()):
            undergrade_degree.append("Science")
        elif ("business" in data4['Q6'][i].lower()):
            undergrade_degree.append("Business")
```

```
        else:
            undergrade_degree.append("Others")
    else:
        undergrade_degree.append("Others")
    i = i + 1
```

After generating the code above, the result of the data classification will be such the following:



The total row of the result also confirmed to be 66 rows as it will be used to do the further summary and visualization.

h. cleansing Expected Graduate Column(Question 7)(Lee)

In this process, we classified the expected graduation date into three type such as the following:

- Spring
- Fall
- Not Sure

The range is classified through the month. If the expected graduate month is before June, so we classify it into Spring. However, the expected graduate month is after June, so it will be classified into fall. The code of this process is such the follow:

```
Expected_Graduation = []
i = 1
while (i < len(data4['Q7'])):
    if (not(pd.isnull(data4['Q7'][i]))):
        if ((pd.to_numeric(data4['Q7'][i][5:][:2])) < 6):
            Expected_Graduation.append("spring " + data4['Q7'][i][:4])
        else:
            Expected_Graduation.append("fall " + data4['Q7'][i][:4])
    else:
        Expected_Graduation.append("Not Sure")
    i = i + 1
```

The result of the classification is such the follow:

The total line of the result is also confirmed to be 66 rows since it will be used for the further summary and the visualization process.

i. cleansing Laptop Used Column(Question 8)(Tim)

This process merges the three columns and classifies the answer to the following type:

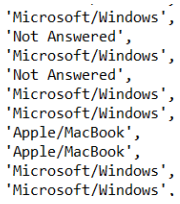
- Microsoft/Windows, Apple/MacBook, and Other
- Microsoft/Windows and Apple/MacBook
- Apple/MacBook and Other
- Microsoft/Windows and Other
- Microsoft/Windows
- Apple/MacBook
- Other
- Not Answered

As we can see that there are many classifications since the answer of this questionnaires' question allows the participant to answer more than one type. The code of the classification is such the follow:

```
laptop_used = []
i = 1
while (i < len(data4['Q8'])):
    if (((pd.to_numeric(data4['Q8'][i], errors='coerce')) == 1) and
        ((pd.to_numeric(data4['Q8.1'][i], errors='coerce')) == 1) and
        ((pd.to_numeric(data4['Q8.2'][i], errors='coerce')) == 1)):
        laptop_used.append('Microsoft/Windows, Apple/MacBook, and Other')
    elif (((pd.to_numeric(data4['Q8'][i], errors='coerce')) == 1) and
          ((pd.to_numeric(data4['Q8.1'][i], errors='coerce')) == 1)):
        laptop_used.append('Microsoft/Windows and Apple/MacBook')
    elif (((pd.to_numeric(data4['Q8.1'][i], errors='coerce')) == 1) and
          ((pd.to_numeric(data4['Q8.2'][i], errors='coerce')) == 1)):
        laptop_used.append('Apple/MacBook and Other')
    elif (((pd.to_numeric(data4['Q8'][i], errors='coerce')) == 1) and
          ((pd.to_numeric(data4['Q8.2'][i], errors='coerce')) == 1)):
        laptop_used.append('Microsoft/Windows and Other')
    else:
        if (not(math.isnan(pd.to_numeric(data4['Q8'][i], errors='coerce')))):
```

```
            laptop_used.append('Microsoft/Windows')
        if (not(math.isnan(pd.to_numeric(data4['Q8.1'][i], errors='coerce')))):
            laptop_used.append('Apple/MacBook')
        if (not(math.isnan(pd.to_numeric(data4['Q8.2'][i], errors='coerce')))):
            laptop_used.append('Other')
        if (math.isnan(pd.to_numeric(data4['Q8'][i], errors='coerce')) and
            math.isnan(pd.to_numeric(data4['Q8.1'][i], errors='coerce')) and
            math.isnan(pd.to_numeric(data4['Q8.2'][i], errors='coerce'))):
            laptop_used.append('Not Answered')
    i = i + 1
```

The result of the classification is such the following:

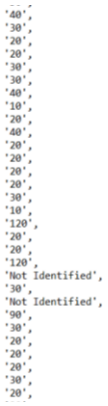


j. cleansing Commuting Time Column(Question 9)(Iwan)

This process cleans and merges the three columns into one column. Since there is a view data which is not completed, so the blank data is substituted by "*Not Identified*". The source code of this process is such the following:

```
commuting_time = []
i = 1
while (i < len(data4['Q9'])):
    if (not(math.isnan(pd.to_numeric(data4['Q9'][i], errors='coerce')))):
        commuting_time.append(data4['Q9'][i])
    else:
        commuting_time.append("Not Identified")
     i = i + 1
```

Other than that, we also make sure that the total row of the data after this process should be the same as the total of the original row which is 66 rows. The new data of this process will be such the following:

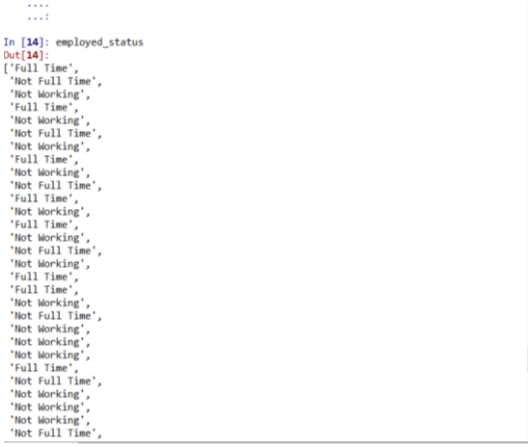


k. cleansing employed status Column(Question 10)(Iwan)

In this process, we also merge the three column answers of the questionnaire. We classified the answer into 4 types such as Full Time, Not Full Time, Not Working, and Do not know the employee status. We considered adding the new classification, which is Do not know the employee status, because we found several rows that had no answer. The code for this code is such the following:

```
employed_status = []
i = 1
while (i < len(data4['Q10'])):
    if (not(math.isnan(pd.to_numeric(data4['Q10'][i], errors='coerce')))):
        employed_status.append('Full Time')
    if (not(math.isnan(pd.to_numeric(data4['Q10.1'][i], errors='coerce')))):
        employed_status.append('Not Full Time')
    if (not(math.isnan(pd.to_numeric(data4['Q10.2'][i], errors='coerce')))):
        employed_status.append('Not Working')
    if (math.isnan(pd.to_numeric(data4['Q10'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q10.1'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q10.2'][i], errors='coerce'))):
        employed_status.append('Do not know the employee status')
    i = i + 1
```

Also, after this process we make sure that the total row of the new data is 66. The new data of the students' employment status is such the following:



```
In [14]: employed_status
Out[14]:
['Full Time',
 'Not Full Time',
 'Not Working',
 'Full Time',
 'Not Working',
 'Not Full Time',
 'Not Working',
 'Full Time',
 'Not Working',
 'Not Full Time',
 'Full Time',
 'Not Working',
 'Full Time',
 'Not Working',
 'Not Full Time',
 'Not Working',
 'Full Time',
 'Full Time',
 'Not Working',
 'Not Full Time',
 'Not Working',
 'Not Working',
 'Not Working',
 'Full Time',
 'Not Full Time',
 'Not Working',
 'Not Working',
 'Not Working',
 'Not Full Time',
```

l. cleansing level of programming skill in Python Column(Question 11)(Iwan)

In this process the data is classified into 6 types such as Little/none, Some familiarity, Average user, Frequent use for projects, Fluent/expert, and do not know the level. We add the last classification, which does not know the level, because we see there is some blank data and we assume that the questionnaire participants who leave this question blank since they did not check any option in the answer. Therefore, we conclude that such participants do not know well his level in programming. The code if this process is such the following:

```
level_programming = []
```

```
i = 1
while (i < len(data4['Q11'])):
    if (not(math.isnan(pd.to_numeric(data4['Q11'][i], errors='coerce')))):
        level_programming.append('Little/none')
    if (not(math.isnan(pd.to_numeric(data4['Q11.1'][i], errors='coerce')))):
        level_programming.append('Some familiarity')
    if (not(math.isnan(pd.to_numeric(data4['Q11.2'][i], errors='coerce')))):
        level_programming.append('Average user')
    if (not(math.isnan(pd.to_numeric(data4['Q11.3'][i], errors='coerce')))):
        level_programming.append('Frequent use for projects')
    if (not(math.isnan(pd.to_numeric(data4['Q11.4'][i], errors='coerce')))):
        level_programming.append('Fluent/expert')
    if (math.isnan(pd.to_numeric(data4['Q11'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q11.1'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q11.2'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q11.3'][i], errors='coerce')) and
        math.isnan(pd.to_numeric(data4['Q11.4'][i], errors='coerce'))):
        level_programming.append('Do not know the level')
        i = i + 1
```

After the cleansing, we also ensure that the total row of the new number is the same as the original row, which is 66 rows. The new data after this cleansing and classification is such the following:

```
Out[17]:
['Frequent use for projects',
 'Some familiarity',
 'Average user',
 'Frequent use for projects',
 'Little/none',
 'Average user',
 'Average user',
 'Some familiarity',
 'Average user',
 'Average user',
 'Average user',
 'Little/none',
 'Frequent use for projects',
 'Average user',
 'Average user',
 'Average user',
 'Some familiarity',
 'Little/none',
 'Some familiarity',
 'Little/none',
 'Little/none',
 'Some familiarity',
 'Average user',
 'Do not know the level',
 'Average user',
 'Some familiarity',
 'Some familiarity',
 'Some familiarity',
 'Little/none',
 'Some familiarity',
 'Average user',
 'Little/none',
```

m. Merging the questionnaire data as one new data frame

After all the data through the cleansing and the classification process, the data will be merged into a one new data frame. This new data frame will then be used for the further summarization and visualization. The code of this process can be seen in the following:

```
questionare_data = pd.DataFrame({'AIT_Section' : AIT_Section,
                                 'Gender' : Gender,
                                 'Age' : Age,
                                 'Height' : Height,
                                 'Citizenship' : Citizenship,
                                 'undergrade_degree' : undergrade_degree,
                                 'Expected_Graduation' : Expected_Graduation,
                                 'laptop_used' : laptop_used,
                                 'commuting_time' : commuting_time,
```

```
                                    'employed_status' : employed_status,
                                    'level_programming' : level_programming})
questionare_data.info()
questionare_data
```

Since the total row of all data components are the same, which is 66 rows, so this new data frame is successfully formed. The data frame is shown by the following:

Out[18]:

	AIT_Section	Gender	...	employed_status	level_programming
0	001	Male	...	Full Time	Frequent use for projects
1	004	Male	...	Not Full Time	Some familiarity
2	DL1	Female	...	Not Working	Average user
3	004	Male	...	Full Time	Frequent use for projects
4	DL1	Male	...	Not Working	Little/none
..	...	...	...	...	...
61	DL1	Male	...	Not Working	Average user
62	004	Male	...	Full Time	Some familiarity
63	DL1	Male	...	Not Working	Average user
64	004	Male	...	Not Working	Little/none
65	DL1	Male	...	Full Time	Some familiarity

[66 rows x 11 columns]

Part 2

Create summary statistics and visualizations for each of the 11 questions, categorized by section (001,004, DL1). For each, explain any interesting differences that you observe(or indicate no real difference)

a. Summarize and visualize the participants based on each section(JT)

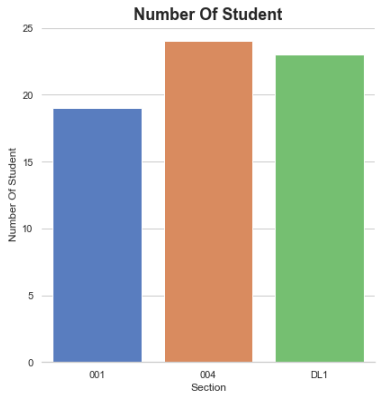
This is code to summarize the section. We can tell from the this graph that there 19 people in section 001, 24 people in section 004, and 23 people in section DL1

```
#summary of Section
################################################################################
questionare_data.groupby(['AIT_Section']).size().reset_index(name='counts')
################################################################################
```

	AIT_Section	counts
0	001	19
1	004	24
2	DL1	23

After visualizing this data, we can easily compare the number of people in each section

```
#########################################################################
AITSection = questionare_data.groupby(['AIT_Section']).size().reset_index(name='counts')

sns.set(style="whitegrid")

g = sns.catplot(x="AIT_Section",
                y="counts",
                data=AITSection,
                height=6,
                kind="bar",
                palette="muted")
g.despine(left=True)
g.set_ylabels("Number Of Student")
g.set_xlabels("Section")
ax = plt.gca()
ax.set_title("Number Of Student", weight='bold').set_fontsize('18')
#########################################################################
```



b. Summarize and visualize the gender based on each section(Tim)

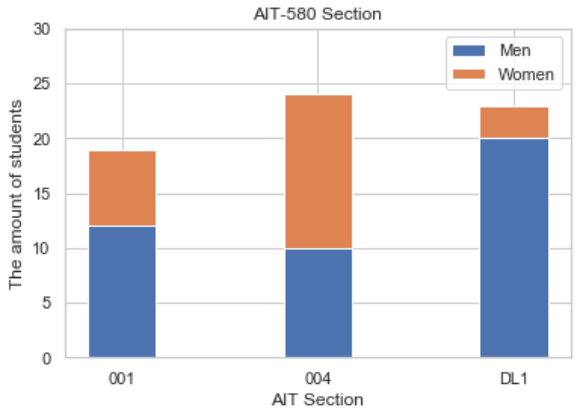
The code to perform such summary is such the following:

```
In [16]: AITSection_gender=questionare_data.groupby(['AIT_Section',
'Gender']).size().reset_index(name='counts')

In [17]: AITSection_gender
Out[17]:
  AIT_Section  Gender  counts
0         001  Female       7
1         001    Male      12
2         004  Female      14
3         004    Male      10
4         DL1  Female       3
5         DL1    Male      20
```

Furthermore, we do the visualization of the data. The code for the data visualization is shown by the following:

```
AITSection_gender=questionare_data.groupby(['AIT_Section', 'Gender']).size().reset_index(name='counts')
N = 3
section_male = (AITSection_gender.at[1,'counts'],
             AITSection_gender.at[3,'counts'],
             AITSection_gender.at[5,'counts'])
section_female = (AITSection_gender.at[0,'counts'],
             AITSection_gender.at[2,'counts'],
             AITSection_gender.at[4,'counts'])

ind = np.arange(N)    # the x locations for the groups
width = 0.35       # the width of the bars: can also be len(x) sequence

p1 = plt.bar(ind, section_male, width)
p2 = plt.bar(ind, section_female, width,
             bottom=section_male)

plt.ylabel('The amount of students')
plt.title('AIT-580 Section')
plt.xticks(ind, ('001', '004', 'DL1'))
plt.yticks(np.arange(0, 31, 5))
plt.xlabel('AIT Section')
plt.legend((p1[0], p2[0]), ('Men', 'Women'))

plt.show()
```

The result of the visualization is such the following:



According to the summary and the visualization, we can see the proportion of these three sections about gender clearly. In the 004 section, the amount of females is more than male. As for 001 and DL1 sections, the amount of male is more than females.

c. Summarize and visualize the age based on each section(JT)

In this code, we want to summarize the age by section and we can observe clearly the number of people in each age in the AIT 580 course and the number of people in each section.

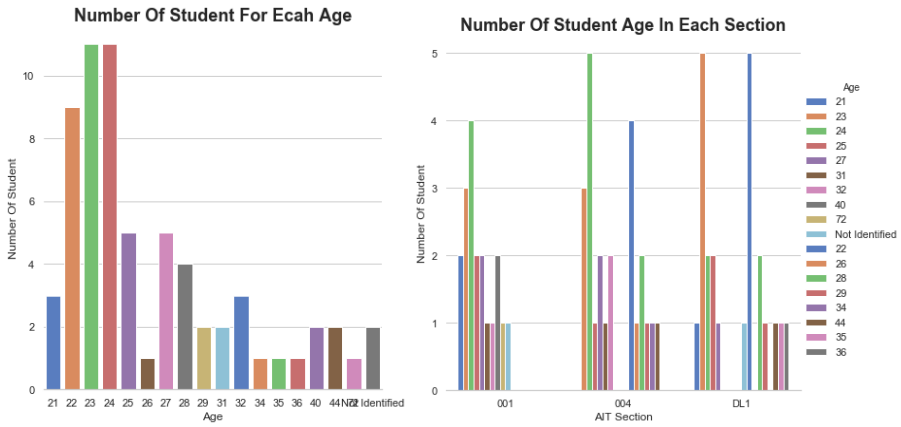
```
questionare_data.groupby(['Age']).size().reset_index(name='counts')
questionare_data.groupby(['AIT_Section', 'Age']).size().reset_index(name='counts')
```

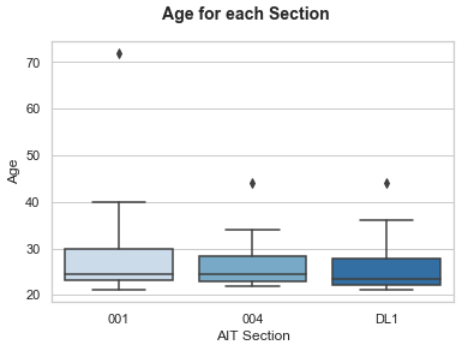
Out[37]:

	Age	counts
0	21	3
1	22	9
2	23	11
3	24	11
4	25	5
5	26	1
6	27	5
7	28	4
8	29	2
9	31	2
10	32	3
11	34	1
12	35	1
13	36	1
14	40	2
15	44	2
16	72	1
17	Not Identified	2

	AIT_Section	Age	counts
0	001	21	2
1	001	23	3
2	001	24	4
3	001	25	2
4	001	27	2
5	001	31	1
6	001	32	1
7	001	40	2
8	001	72	1
9	001	Not Identified	1
10	004	22	4
11	004	23	3
12	004	24	5
13	004	25	1
14	004	26	1
15	004	27	2
16	004	28	2
17	004	29	1
18	004	31	1
19	004	32	2
20	004	34	1
21	004	44	1
22	DL1	21	1
23	DL1	22	5
24	DL1	23	5
25	DL1	24	2
26	DL1	25	2
27	DL1	27	1
28	DL1	28	2
29	DL1	29	1
30	DL1	35	1
31	DL1	36	1
32	DL1	44	1
33	DL1	Not Identified	1

To visualize the age based on each section, we try to use the bar chart and box chart to make it easier to tell the age information. From the bar chart, we can tell the distribution from each section. and to make the chart tell more valuable information, we use boxchart to know the statistical data. We can observe from the box chart that the age of the students in section 001 is relatively high.





d. Summarize and visualize the students' height based on each section(JT)

In this code, we want to summarize the height by section and we can observe clearly the number of people in each height in the AIT 580 course and the number of people in each section.

```
617 ###########################################################################
618 questionare_data.groupby(['Height']).size().reset_index(name='counts')
619 questionare_data.groupby(['AIT_Section', 'Height']).size().reset_index(name='counts')
```

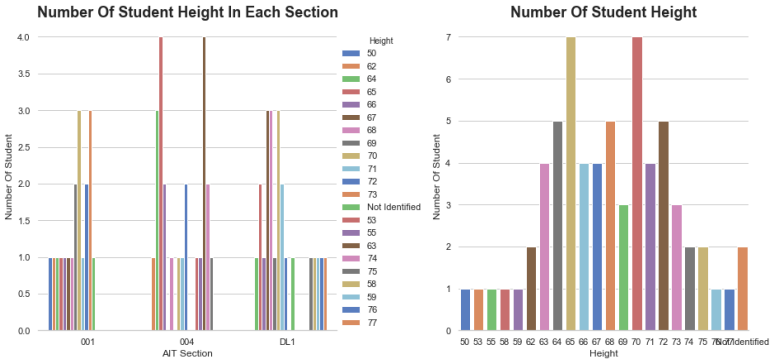
Out[43]:

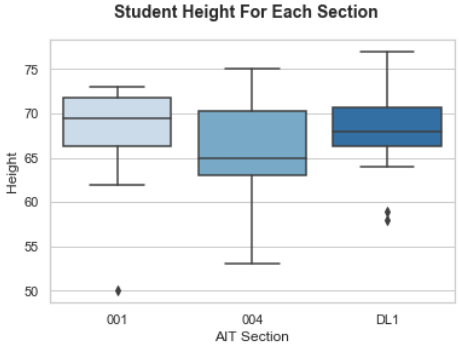
	Height	counts
0	50	1
1	53	1
2	55	1
3	58	1
4	59	1
5	62	2
6	63	4
7	64	5
8	65	7
9	66	4
10	67	4
11	68	5
12	69	3
13	70	7
14	71	4
15	72	5
16	73	3
17	74	2
18	75	2
19	76	1
20	77	1
21	Not Identified	2

	AIT_Section	Height	counts
0	001	50	1
1	001	62	1
2	001	64	1
3	001	65	1
4	001	66	1
5	001	67	1
6	001	68	1
7	001	69	2
8	001	70	3
9	001	71	1
10	001	72	2
11	001	73	3
12	001	Not Identified	1
13	004	53	1
14	004	55	1
15	004	62	1
16	004	63	4
17	004	64	3
18	004	65	4
19	004	66	2
20	004	68	1
21	004	70	1
22	004	71	1
23	004	72	2
24	004	74	2
25	004	75	1
26	DL1	58	1
27	DL1	59	1
28	DL1	64	1
29	DL1	65	2
30	DL1	66	1
31	DL1	67	3
32	DL1	68	3
33	DL1	69	1

From the bar chart, we can tell the distribution from each section. And to make the chart tell more valuable information, we use boxchart to know the statistical data. We can observe from the box chart that the height of the students in section 001 is relatively high.

However, students in section DL4 have the smaller interquartile range, that means the difference in height of students in this class is small. Also, we tell from the box chart that students in section DL4 have the larger interquartile range means the height difference of students in this class is relatively large.





e. Summarize and visualize the students' citizenship based on each section(Tim)

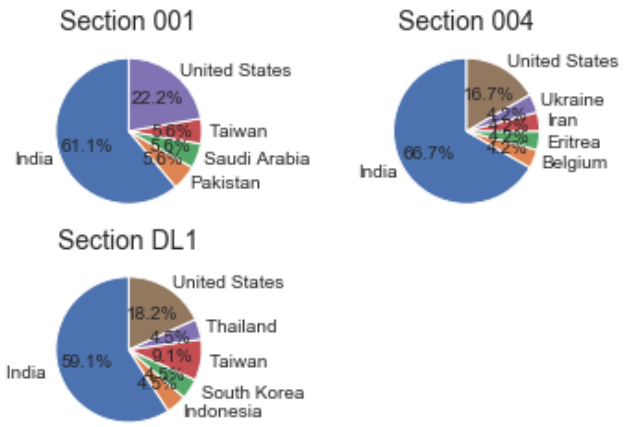
The code to perform such summary is such the following:

```
In [19]: AITSection_Citizenship1=questionare_data.groupby(['AIT_Section',
'Citizenship']).size().reset_index(name='counts')

In [20]: AITSection_Citizenship1
Out[20]:
   AIT_Section    Citizenship  counts
0          001          India      11
1          001       Pakistan       1
2          001   Saudi Arabia       1
3          001         Taiwan       1
4          001  United States       4
5          001     not found       1
6          004        Belgium       1
7          004        Eritrea       1
8          004          India      16
9          004           Iran       1
10         004        Ukraine       1
11         004  United States       4
12         DL1          India      13
13         DL1      Indonesia       1
14         DL1    South Korea       1
15         DL1         Taiwan       2
16         DL1       Thailand       1
17         DL1  United States       4
18         DL1     not found       1
```

Furthermore, we do the visualization of the data. The code for the data visualization is shown by the following:

```
plt.figure()

plt.subplot(2,2,1)
labels1 =  'India' , 'Pakistan','Saudi Arabia','Taiwan','United States'
sizes1= [AITSection_Citizenship1.at[0,'counts'],AITSection_Citizenship1.at[1,'counts'],
        AITSection_Citizenship1.at[2,'counts'],AITSection_Citizenship1.at[3,'counts'],
        AITSection_Citizenship1.at[4,'counts']]
explode = (0.01, 0.01, 0.01)
plt.pie(sizes1, labels=labels1, startangle=90, autopct='%.1f%%',textprops = {"fontsize" : 10})
plt.title('Section 001',{"fontsize" : 14})

plt.subplot(2,2,2)
labels2 =  'India','Belgium' , 'Eritrea','Iran','Ukraine ','United States'
sizes2= [AITSection_Citizenship1.at[8,'counts'],AITSection_Citizenship1.at[6,'counts'],
        AITSection_Citizenship1.at[7,'counts'],AITSection_Citizenship1.at[9,'counts'],
        AITSection_Citizenship1.at[10,'counts'],AITSection_Citizenship1.at[11,'counts']]
plt.pie(sizes2, labels=labels2, startangle=90, autopct='%.1f%%',textprops = {"fontsize" : 10})
plt.title('Section 004',{"fontsize" : 14})

plt.subplot(2,2,3)
labels3 =  'India ' , 'Indonesia',' South Korea','Taiwan','Thailand ','United States'
sizes3= [AITSection_Citizenship1.at[12,'counts'],AITSection_Citizenship1.at[13,'counts'],
        AITSection_Citizenship1.at[14,'counts'],AITSection_Citizenship1.at[15,'counts'],
        AITSection_Citizenship1.at[16,'counts'],AITSection_Citizenship1.at[17,'counts']]
plt.pie(sizes3, labels=labels3, startangle=90, autopct='%.1f%%',textprops = {"fontsize" : 10})
plt.title('Section DL1',{"fontsize" : 14})

plt.show()
```

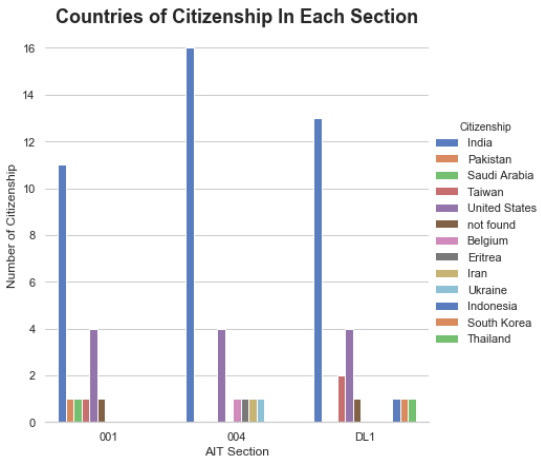
The result of the visualization is such the following:



In addition, we do another visualization of the data. The code for the data visualization is shown by the following:

```
AITSection_Citizenship2 = questionare_data.groupby(['AIT_Section', 'Citizenship']).size().reset_index(name='counts')

import seaborn as sns
sns.set(style="whitegrid")

# Draw a nested barplot to show the total of expert for class and sex
g = sns.catplot(x="AIT_Section",
                y="counts",
                hue="Citizenship",
                data=AITSection_Citizenship2,
                height=6,
                kind="bar",
                palette="muted")
g.despine(left=True)
g.set_ylabels("Number of Citizenship")
g.set_xlabels("AIT Section")
ax = plt.gca()
ax.set_title("Countries of Citizenship In Each Section", weight='bold').set_fontsize('18')
```

The result of the visualization is such the following:



According to the summary and the visualization, we can see the proportion of countries in each section. These pie and bar charts about the country of citizenship show that result clearly. In these three sections, the amount of students from India is more than other countries. Then there is the United States. As for other countries, it depends on different sections.

f. Summarize and visualize the undergrad degree based on each section(Lee)

This data is considered as the nominal data. Therefore, the data summary of the students' undergrad degree is such the following:

	AIT_Section	undergrade_degree	counts
0	001	Business	1
1	001	Engineering	13
2	001	Others	5
3	004	Business	2
4	004	Engineering	16
5	004	Others	3
6	004	Science	3
7	DL1	Engineering	16
8	DL1	Others	4
9	DL1	Science	3

The code to perform such summary is such the following:

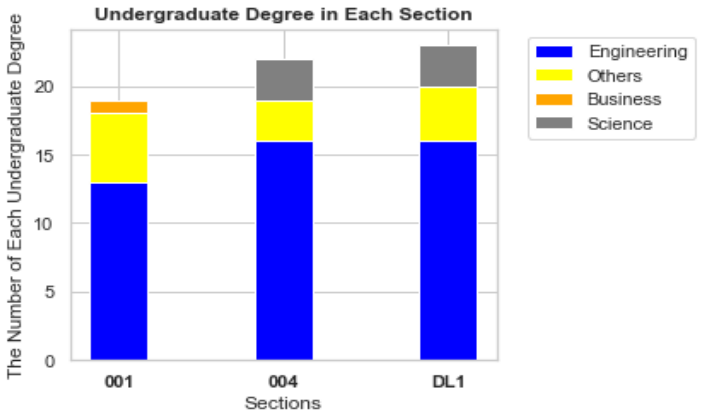
```
AITSection_undergrade_degree = questionare_data.groupby(['AIT_Section',
'undergrade_degree'])['AIT_Section'].count().reset_index(name='counts')
```

Furthermore, we do the visualization of the data. The code for the data visualization is shown by the following:

```
Business = [AITSection_undergrade_degree.at[0, 'counts'],
            AITSection_undergrade_degree.at[3, 'counts'],
            0]
Engineering = [AITSection_undergrade_degree.at[1, 'counts'],
               AITSection_undergrade_degree.at[4, 'counts'],
               AITSection_undergrade_degree.at[7, 'counts']]
```

```
Science = [0,
           AITSection_undergrade_degree.at[6, 'counts'],
           AITSection_undergrade_degree.at[9, 'counts']]
Others = [AITSection_undergrade_degree.at[2, 'counts'],
          AITSection_undergrade_degree.at[5, 'counts'],
          AITSection_undergrade_degree.at[8, 'counts']]

bars1 = Engineering
bars2 = Others
bars3 = Business
barsTS = Science

# Heights of bars1 + bars2
bars = np.add(bars1, bars2).tolist()

# The position of the bars on the x-axis
r = [1, 2, 3]

names = ['001', '004', 'DL1']
barWidth = 0.35       # the width of the bars: can also be len(x) sequence

# Create category_1
p1 = plt.bar(r, bars1, color='blue', edgecolor='white', width=barWidth)
# Create category_2
p2 = plt.bar(r, bars2, bottom=bars1, color='yellow', edgecolor='white', width=barWidth)
# Create category_3
p3 = plt.bar(r, bars3, bottom=bars, color='orange', edgecolor='white', width=barWidth)
# Create barsTS
p4 = plt.bar(r, barsTS, bottom=bars, color='Grey', edgecolor='white', width=barWidth)

# Custom X axis
fig.add_subplot(111)
plt.xticks(r, names, fontweight='bold')
plt.xlabel("Sections")
plt.ylabel("The Number of Each Undergraduate Degree")
plt.title("Undergraduate Degree in Each Section", fontweight='bold')
plt.legend((p1[0], p2[0], p3[0], p4[0]), ('Engineering', 'Others', 'Business',
'Science'), bbox_to_anchor=(1.05, 1.0),loc = 'upper left')
plt.tight_layout()
plt.show()
```

The result of the visualization is such the following:



According to the results, the number of students who have Engineering degrees for their undergraduate study dominate the section. However, in section 001, the undergraduate degree seems to be more diverse than the other section. Because we can observe that Business and much larger Others exist in comparison with the other sections.

g. Summarize and visualize the Expected Graduation based on each section(Lee)

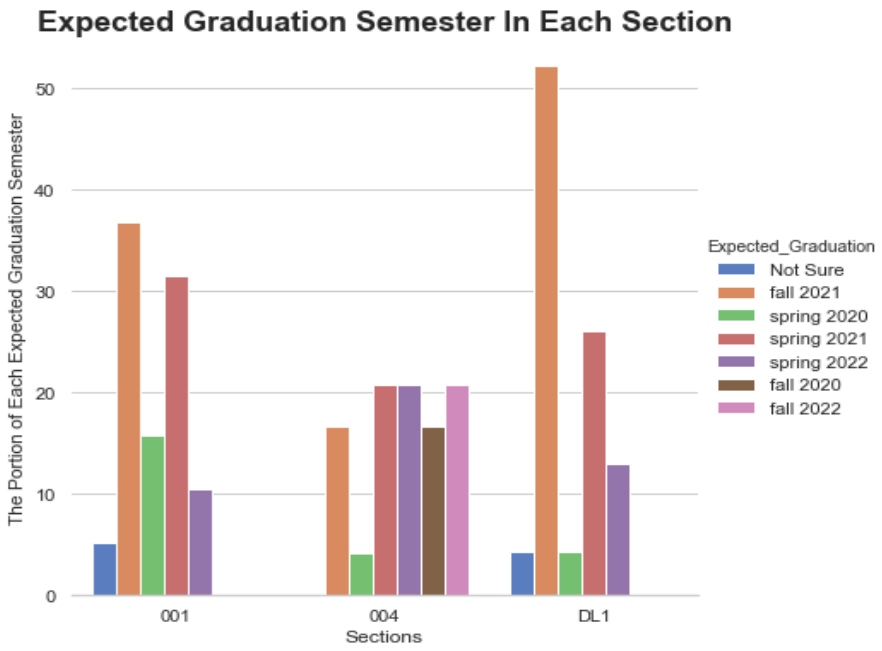
This data is considered as the nominal data. Therefore, the data summary of the students' Expected Graduate is such the following:

	AIT_Section	Expected_Graduation	percentage
0	001	Not Sure	5.263158
1	001	fall 2021	36.842105
2	001	spring 2020	15.789474
3	001	spring 2021	31.578947
4	001	spring 2022	10.526316
5	004	fall 2020	16.666667
6	004	fall 2021	16.666667
7	004	fall 2022	20.833333
8	004	spring 2020	4.166667
9	004	spring 2021	20.833333
10	004	spring 2022	20.833333
11	DL1	Not Sure	4.347826
12	DL1	fall 2021	52.173913
13	DL1	spring 2020	4.347826
14	DL1	spring 2021	26.086957
15	DL1	spring 2022	13.043478

The code to perform such summary is such the following:

```
AITSection_Expected_Graduation_sum =
AITSection_Expected_Graduation.groupby(['AIT_Section',
'Expected_Graduation']).agg({'counts':'sum'})
AITSection_Expected_Graduation_sum
AITSection_Expected_Graduation_pct =
AITSection_Expected_Graduation_sum.groupby('AIT_Section').apply(lambda x: 100 * x /
float(x.sum())).reset_index()
AITSection_Expected_Graduation_pct= AITSection_Expected_Graduation_pct.rename(columns =
{'counts':'percentage'})
```

Furthermore, we do the visualization of the data. The code for the data visualization is shown by the following:



According to the summary and the visualization result, the number of students who expect to graduate in Fall 2021 dominates the section 001 and DL1. In addition, we can observe that students who are in section 004 are distributed evenly, in comparison with the other sections.

h. Summarize and visualize the students' Laptop Used based on each section(Tim)

This data is considered as the nominal data. Therefore, the data summary of the Laptop used is such the following:

	AIT_Section	laptop_used	counts
0	001	Apple/MacBook	5
1	001	Microsoft/Windows	11
2	001	Microsoft/Windows and Apple/MacBook	2
3	001	Not Answered	1
4	004	Apple/MacBook	7
5	004	Microsoft/Windows	16
6	004	Microsoft/Windows and Apple/MacBook	1
7	DL1	Apple/MacBook	5
8	DL1	Microsoft/Windows	17
9	DL1	Not Answered	1

The code to perform such summary is such the following:

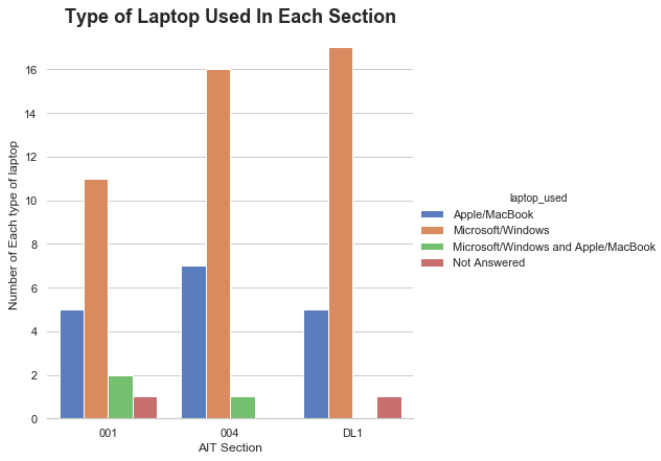
```
questionare_data.groupby(['AIT_Section',
'laptop_used']).size().reset_index(name='counts')
```

Furthermore, we do the visualization of the data. The code for the data visualization is shown by the following:

```
AITSection_laptop_used = questionare_data.groupby(['AIT_Section',
```

```
'laptop_used']).size().reset_index(name='counts')

import seaborn as sns
sns.set(style="whitegrid")

# Draw a nested barplot to show the total of expert for class and sex
g = sns.catplot(x="AIT_Section",
                y="counts",
                hue="laptop_used",
                data=AITSection_laptop_used,
                height=6,
                kind="bar",
                palette="muted")
g.despine(left=True)
g.set_ylabels("Number of Each type of laptop")
g.set_xlabels("AIT Section")
ax = plt.gca()
ax.set_title("Type of Laptop Used In Each Section", weight='bold').set_fontsize('18')
```

The result of the visualization is such the following:



According to the summary and the visualization result, the number of Windows User in AIT580 course is dominating. Another interesting thing is, section 001 and 004 contain students that use more than one laptop. Also, it can be concluded as well that all students in section 004 have laptops since all students answer the laptop used question.

i. Summarize and visualize the Commuting Time based on each section(Iwan)

This data is considered as the ratio data. Therefore, the data summary of the commuting time is such the following:

```
  AIT_Section count unique top freq
0         001    19      8  20   10
1         004    24      7  30   10
```

```
2         DL1    23      5  30    9
```

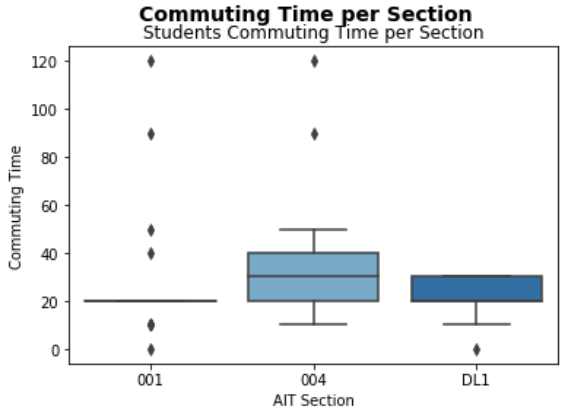
The code to perform such summary is such the following:

```
questionare_data.groupby("AIT_Section")['commuting_time'].describe().reset_index()
```

Furthermore, we do the visualization of the data. The code for the data visualization is shown by the following:

```
fig = plt.figure()
fig.suptitle('Commuting Time per Section', fontsize=14, fontweight='bold')
ax = fig.add_subplot(111)
sns.boxplot(x = questionare_data["AIT_Section"],
            y=pd.to_numeric(questionare_data["commuting_time"],errors='coerce'),
            palette="Blues")
ax.set_title('Students Commuting Time per Section')
ax.set_xlabel('AIT Section')
ax.set_ylabel('Commuting Time')
plt.show()
```

The result of the visualization is such the following:



According to the summary and the visualization, the biggest number of students who spend the time longer to reach the campus is in the 004 section. We also can see that even though the mean is the same as the online section, the number of students which spend time longer to reach campus is students in section 004.

j. Summarize and visualize the employed status based on each section(Iwan)

This data is considered as the nominal data. Therefore, the data summary of the students' employment status is such the following:

```
   AIT_Section                  employed_status  counts
0          001  Do not know the employee status       1
1          001                       Full Time       4
2          001                   Not Full Time       3
```

```
3          001                       Not Working      11
4          004                         Full Time       9
5          004                     Not Full Time       8
6          004                       Not Working       7
7          DL1   Do not know the employee status       1
8          DL1                         Full Time       6
9          DL1                     Not Full Time       4
10         DL1                       Not Working      12
```

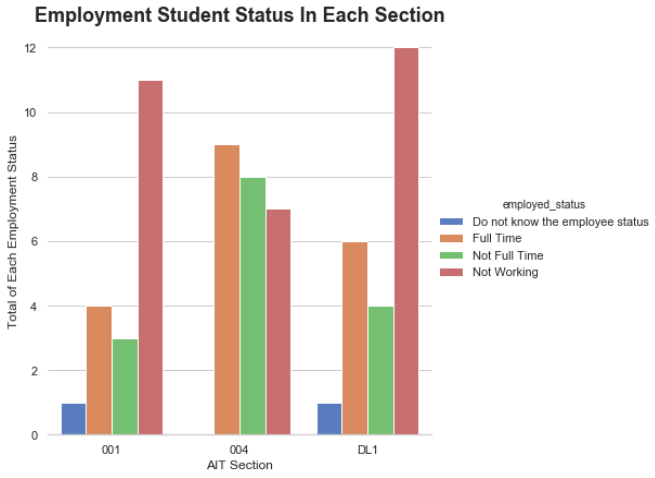
The code to perform such summary is such the following:

```
AITSection_employedStatus =
questionare_data.groupby(['AIT_Section','employed_status']).size().reset_index(name='counts')
```

Furthermore, we do the visualization of the data. The code for the data visualization is shown by the following:

```
sns.set(style="whitegrid")
g = sns.catplot(x="AIT_Section",
                y="counts",
                hue="employed_status",
                data=AITSection_employedStatus,
                height=6,
                kind="bar",
                palette="muted")
g.despine(left=True)
g.set_ylabels("Total of Each Employment Status")
g.set_xlabels("AIT Section")
ax = plt.gca()
ax.set_title("Employment Student Status In Each Section",
weight='bold').set_fontsize('18')
```

The result of the visualization is such the following:



According to the data summary and the data visualization, the number of students who do not work in the DL1 section has the biggest number compared to the two other sections. Other than that, the students who work full time and not full time have the most number in

section 004. We conclude that the number of students who work in section 004 is greater than the other sections.

k. Summarize and visualize the level of programming skill in Python based on each section(Iwan)

This data is considered as the nominal data. Therefore, the data summary of the students' level of the programming skill in Python is such the following:

	AIT_Section	level_programming	counts
0	001	Average user	7
1	001	Do not know the level	1
2	001	Frequent use for projects	3
3	001	Little/none	4
4	001	Some familiarity	4
5	004	Average user	6
6	004	Do not know the level	1
7	004	Fluent/expert	1
8	004	Frequent use for projects	2
9	004	Little/none	6
10	004	Some familiarity	8
11	DL1	Average user	10
12	DL1	Do not know the level	1
13	DL1	Frequent use for projects	3
14	DL1	Little/none	2
15	DL1	Some familiarity	7

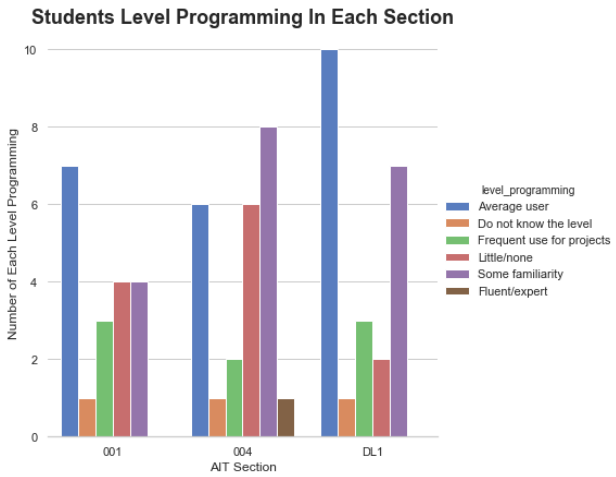
The code to perform such summary is such the following:

```
questionare_data.groupby(['AIT_Section', 'level_programming']).size().reset_index(name='counts')
```

Furthermore, we do the visualization of the data. The code for the data visualization is shown by the following:

```
AITSection_levelProgramming = questionare_data.groupby(['AIT_Section',
'level_programming']).size().reset_index(name='counts')

sns.set(style="whitegrid")

# Draw a nested barplot to show the total of expert for class and sex
g = sns.catplot(x="AIT_Section",
                y="counts",
                hue="level_programming",
                data=AITSection_levelProgramming,
                height=6,
                kind="bar",
                palette="muted")
g.despine(left=True)
g.set_ylabels("Number of Each Level Programming")
g.set_xlabels("AIT Section")
ax = plt.gca()
ax.set_title("Students Level Programming In Each Section", weight='bold').set_fontsize('18')
```

The result of the visualization is such the following:



According to the summary and the visualization, the number of students with the level of python programming in average users has the most number in the DL1 section. Besides that, the number of students with the level in little/non has the most number in section 004. At the same time, section 004 is also the only section which has a student in fluent/expert level of python programming.

Part 3

Discuss some suggestions for how to improve the online learning experience for this experiment

(LEE)

From the perspective of the department of enrollment workers, they might come up with the big data analysis of the entire students of GMU in order to improve the efficiency of online learning lectures, unless violating the privacy of Act. In particular, as of this survey, the commuting time, employed status, and laptop environment can be considered how many classes will be opened as an online section. For example, we might add more information from some influential predictors such as whether having a seperate space for studying and the range of the preference time in each weekday.

(IWAN)

The online learning is great so far. In my perspective, I think the way the material is presented is as good as in the class session. Moreover, the example of how the material is always

provided after the material is presented. The other important thing for me is the recording because through the recording, I can review the material easier. Some articles that are usually given can also be a good way to improve the online learning course as the students are informed how to implement the chapter.

Other than that, the online survey probably can be a good way to improve the future online course. For instance, since the students who are supposed to be in the online course are students who work full time. However, the number of students who work full time is dominating in section 004. Besides that, the commuting time can be a good way to decide which student can attend the online course since as shown through the survey, it seems the number of students who spend more time to arrive on campus in session 004 is greater than the online section. Those can be a good consideration as well to improve the online course in the future if such information can be retrieved before the course starts.

(Ting-Yeh Yang)

From my perspective, I prefer to take the online courses on account of the following reasons.

First of all, I can repeat the record of every course if I didn't listen to the content of course clearly. To be specific, as an international student, my English ability is not good enough to understand the content 100%. However, I can loop the course by recording function to make sure that I can follow the content of course.

In addition, the class time is flexible. I can take the courses any time I want. Take myself as an example, the class time of the DL1 section is 19:30. However, I have the other three courses on the same day including 10:30 to 12:20, 12:30 to 14:20, and 16:30 to 19:20 respectively. That is a big burden for me. I cannot pay attention to all courses so much.

The online learning is nice so far. However, I would like to provide a suggestion. Give us more team assignments. In my opinion, data analytics is a subjective major. Even though we can use statistics skills and program language to present, we still need to explain them by myself. It means that we can explain them in several aspects. Team assignments can help me listen to the ideas from other members. I think that is helpful for me.

(JING-TING HUANG)

In my perspective, I think there are several benefits from online classes. However, I think there is something we can do to improve online classes. for instance, I think we can have more interaction in the online class. The online class is lack of space that students can face with the professor, and sometimes, it makes students reduce efficiency. To solve this problem, I think both

students and professors should work together to create an interactive environment. Professor can ask students questions or hold class competitions to arouses students' desire to compete