In [423]:
```
#LIBRARY TO USE

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.style as style
style.use('seaborn-bright')
import seaborn as sns
import math
from matplotlib.patches import ConnectionPatch
```

In [313]:
```
master = pd.read_csv("master.csv")
```

In [314]:
```
extra = pd.read_csv("extra_questions_withID.csv")
```

In [315]:
```
#extra
```

In [316]:
```
extra = extra.iloc[:, [0] + list(range(42, 56))]
```

In [317]:
```
datF = pd.merge(master, extra, on="RespondentId")
```

In [318]:
```
datF
```

Out[318]:

	Unnamed: 0	X	RespondentId	Enroll	Employ	Working	Hrs	Age	Ethnicity	Income	...	DiffLvlFA	ChallFA	EmergFood	DiffConcentrate
0	26832	NaN	95772100	Full-time	Part-time	Off campus	More than 19 hours	NaN	1	Less than $10,000	...	A little difficult	1 4	NaN	Almost every day
1	26833	NaN	95772104	Full-time	No	NaN	NaN	2.0	1	30,000 to 39,999	...	Moderately difficult	1	4	About once a week
2	26834	NaN	95772110	Full-time	Full-time	Off campus	More than 19 hours	NaN	1	90,000 to 99,999	...	A little difficult	1	NaN	About once a week
3	26835	NaN	95772115	Full-time	Full-time	Off campus	More than 19 hours	NaN	1	20,000 to 29,999	...	Moderately difficult	1 2	NaN	About once a week
4	26836	NaN	95772121	Full-time	Part-time	Both	More than 19 hours	NaN	1	30,000 to 39,999	...	A little difficult	1 2 3	3 4	Almost every day
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
1738	28570	NaN	95997981	Full-time	Part-time	On campus	19 hours or less	NaN	1	70,000 to 79,999	...	I am not sure	1 2 5	NaN	Never
1739	28571	NaN	96000664	Full-time	Part-time	Off campus	19 hours or less	NaN	1	$100,000 or more	...	I am not sure	1	NaN	Never
1740	28572	NaN	96001101	Part-time	Part-time	Both	More than 19 hours	NaN	1	10,000 to 19,999	...	Not difficult at all	1 3	NaN	Never
1741	28573	NaN	96001710	Full-time	Full-time	Off campus	More than 19 hours	NaN	11	$100,000 or more	...	Not difficult at all	1	NaN	Never
1742	28574	NaN	96005041	Part-time	Part-time	Off campus	19 hours or less	NaN	1, 11	30,000 to 39,999	...	Very difficult	1 5	NaN	About once a week

1743 rows × 73 columns

In [405]:
```
# Grouping Food Security
def categorize_index(index):
    if index in range(0,1):
        return "Marginal/High Food Security"
    elif index in range(2, 4):
        return "Low Food Security"
    elif index in range(5, 6):
        return "Very Low Food Security"
    elif isinstance(index, float) and math.isnan(index):
        return "No Response"
    else:
        return index


datF['USDAcat'] = datF['index'].apply(categorize_index)
```

In [384]:
```
Food_Ins = datF['USDAcat']
Food_Ins
```

Out[384]:
```
0               6.0
1       No Response
2       No Response
3       No Response
4               6.0
           ...
1738    No Response
1739    No Response
1740    No Response
1741    No Response
1742            6.0
Name: USDAcat, Length: 1743, dtype: object
```

In [387]:
```
# Grouping Gender
datF['Gender'] = datF['Gender'].apply(lambda x: 'Female' if x == '1' else ('Male' if x == '2' else 'Others'))

Gender = datF['Gender']
```

In [388]:
```
Gender
```

Out[388]:
```
0       Female
1       Female
2       Female
3       Female
4         Male
         ...
1738    Female
1739    Female
1740    Female
1741      Male
1742      Male
Name: Gender, Length: 1743, dtype: object
```

In [390]:
```
#Grouping The Loans Into Two

datF['FedAid'] = datF['FedAid'].apply(lambda x: 'Loans' if x == 'Emergency Loan' else x)

FA = datF['FedAid']
```

In [393]:
```
FA
```

Out[393]:
```
0             Loans
1        Work-study
2             Loans
3        Work-study
4             Loans
           ...
1738    Scholarship
1739     Work-study
1740     Work-study
1741     Work-study
1742     Work-study
Name: FedAid, Length: 1743, dtype: object
```

In [400]:
```
# Grouping Income For Analysis
# Assuming 'datC' is a DataFrame containing the 'Income' column
datF['newgroup'] = datF['Income'].apply(lambda x: 'Less than $20,000' if x in ['$10,000 to $19,999', 'Less tha

Incom_ G = datF['newgroup']
```

In [402]:
```
# Regroup Of Income Level

ReG_Income = pd.Categorical(datF['newgroup'], categories=["Less than $20,000", "$20,000 to $49,999", "$50,000
```

In [403]:
```
datF["College"] = np.where(["College"] == "1", "Business",
                           np.where(datF["College"] == "2", "Education",
                                    np.where(datF["College"] == "3", "Engineering",
                                             np.where(datF["College"] == "4", "Liberal Arts",
                                                      np.where(datF["College"] == "5", "Health Science",
                                                               np.where(datF["College"] == "6", "Nursing",
                                                                        np.where(datF["College"] == "7", "Scien
                                                                                 np.where(datF["College"] == "8
                                                                                          np.where(datF["Colleg
```

In [404]:
```
Col = datF["College"]

Col
```

Out[404]:
```
0                  Nursing
1       Multiple  Colleges
2                  Pharmacy
3              Liberal Arts
4                  Pharmacy
                  ...
1738                 Other
1739        Health Science
1740        Health Science
1741               Pharmacy
1742           Engineering
Name: College, Length: 1743, dtype: object
```

In [408]:
```
# Educational level
datF['Classification'] = pd.Categorical(datF['Classification'],
                                        categories=["Freshman", "Sophomore", "Junior", "Senior",
                                                    "Special Course",
                                                    "Masters", "Doctoral"], ordered=True)

datF['USDAcat'] = pd.Categorical(datF['USDAcat'], categories=["Very Low Food Security", "Low Food Security", "
```

In [410]:
```
Col = datF['Classification']

Col
```

Out[410]:
```
0       Doctoral
1         Senior
2         Junior
3         Senior
4         Senior
           ...
1738      Senior
1739         NaN
1740    Freshman
1741         NaN
1742      Junior
Name: Classification, Length: 1743, dtype: category
Categories (7, object): ['Freshman' < 'Sophomore' < 'Junior' < 'Senior' < 'Special Course' < 'Masters' < 'Do
ctoral']
```

How is use of government federal aid/assistance associated with food insecurity as measured by the USDA index or categories?

INTRODUCTION;

We want to investigate how food insecurity could affect the level or success of students in various colleges on campus. We are using a case study on UTEP campus as a sample. This studies will also help us to know the which ethnicity benifits mostly from the government aid and why that. We

In [414]:
```
#FOOD SECURITY

Food_Security = pd.DataFrame(datF['USDAcat'].value_counts(dropna=False)).reset_index()

Food_Security.columns = ['USDAcat', 'Frequency']

Food_Security
```

Out[414]:

	USDAcat	Frequency
0	NaN	1499
1	Very Low Food Security	155
2	Low Food Security	89
3	Maginal/High Food Security	0

In [ ]:
```

```

In [418]:
```
# Seting No Response To Zero.
Food_Security.loc[len(Food_Security)] = ['No response', 0]
```

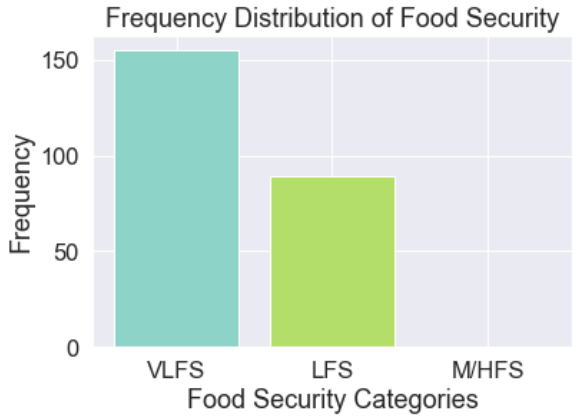
In [439]:
```
Food_Security['USDAcat'] = Food_Security['USDAcat'].fillna('No response')
```

In [440]:
```
# The Percentage In the ratio of Food Security

Food_Security['percent'] = round(Food_Security['Frequency'] / Food_Security['Frequency'].sum() * 100, 0)
```

In [441]:
```
Food_Security.describe
```

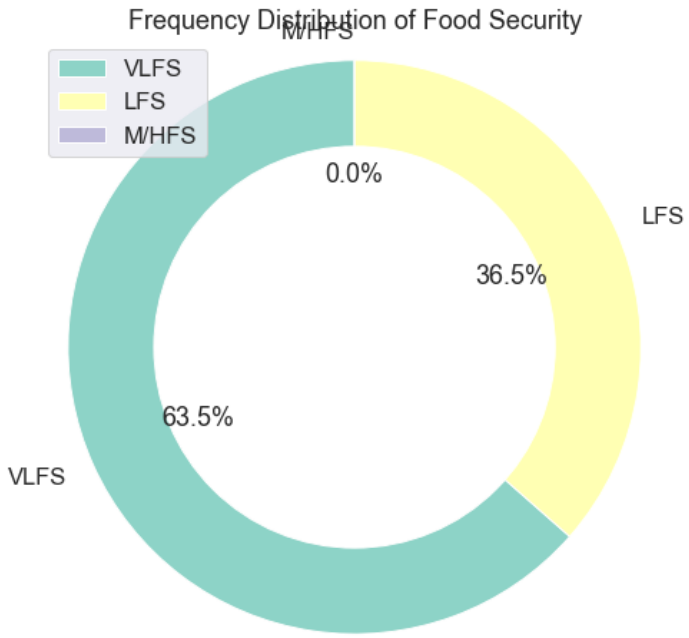
Out[441]:
```
<bound method NDFrame.describe of                      USDAcat  Frequency  percent
0                 No response       1499     86.0
1      Very Low Food Security        155      9.0
2           Low Food Security         89      5.0
3  Maginal/High Food Security          0      0.0
4                 No response          0      0.0>
```

In [447]:
```
import numpy as np
import matplotlib.pyplot as plt

# Data
categories = ['No response', 'VLFS', 'LFS', 'M/HFS']
frequencies = [1499, 155, 89, 0]

# Remove "No response" category
categories = categories[1:]
frequencies = frequencies[1:]

# Generate a color for each bar
colors = plt.cm.Set3(np.linspace(0, 1, len(categories)))

# Create bar plot with different colors
plt.bar(categories, frequencies, color=colors)

# Add labels and title
plt.xlabel('Food Security Categories')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of Food Security')

# Display the plot
plt.show()
```



In [460]:
```
import matplotlib.pyplot as plt

# Data
categories = ['No response', 'VLFS', 'LFS', 'M/HFS']
frequencies = [1499, 155, 89, 0]

# Remove "No response" category
categories = categories[1:]
frequencies = frequencies[1:]

# Generate a color for each category
colors = plt.cm.Set3(range(len(categories)))

# Set figure size
plt.figure(figsize=(8, 8))

# Create donut plot
plt.pie(frequencies, labels=categories, colors=colors, autopct='%1.1f%%', startangle=90, wedgeprops={'edgecolo

# Add a circle at the center to create a donut shape
center_circle = plt.Circle((0, 0), 0.7, color='white', edgecolor='white')
plt.gca().add_artist(center_circle)

# Add labels and title
plt.title('Frequency Distribution of Food Security')

# Display the plot
plt.axis('equal')
plt.legend()
plt.show()
```

```
/var/folders/bt/sgfthxs95jlfbrcxn96skxyh0000gn/T/ipykernel_15588/1538101994.py:21: UserWarning: Setting the
'color' property will override the edgecolor or facecolor properties.
```



In [455]:
```
df_grouped = sort_FA_FS_count.pivot(index='USDAcat', columns='FedAid').reindex().fillna(0)
df_grouped.columns = df_grouped.columns.get_level_values(1)

pivot = df_grouped

pivot
```

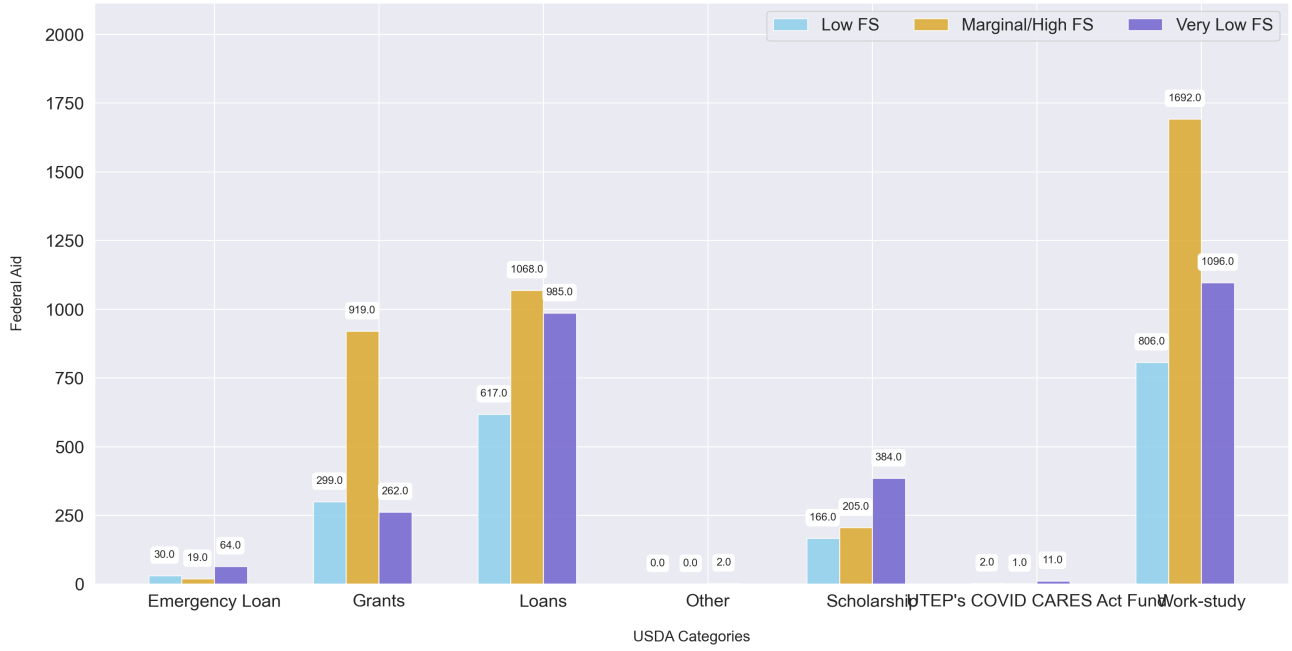
Out[455]:

FedAid USDAcat	Emergency Loan	Grants	Loans	Other	Scholarship	UTEP's COVID CARES Act Fund	Work-study
Low FS	30.0	299.0	617.0	0.0	166.0	2.0	806.0
Marginal/High FS	19.0	919.0	1068.0	0.0	205.0	1.0	1692.0
Very Low FS	64.0	262.0	985.0	2.0	384.0	11.0	1096.0

In [456]:

```
df = pivot.fillna(0).T.copy()
fig, ax = plt.subplots(figsize = (20,10), dpi = 200)

# select the colors you would like to use for each category
colors = ['skyblue','goldenrod','slateblue','seagreen', 'green', 'orange', 'purple']

# used to set the title, y, and x labels
# ax.set_title('\nF\n', fontsize = 14)
ax.set_xlabel('\nUSDA Categories\n', fontsize = 14)
ax.set_ylabel('\nFederal Aid\n', fontsize = 14)

# create an offsetting x axis to iterate over within each group
x_axis = np.arange(len(df))+1

# center each group of columns
offset = -0.3

# iterate through each set of values and the colors associated with each
# category
for index, col_name, color in zip(x_axis, df.columns, colors):
    print(col_name)

    x = x_axis+offset
    height = df[col_name].values

    ax.bar(
        x,
        height,
        width = 0.2,
        color = color,
        alpha = 0.8,
        label = col_name
    )

    offset += 0.2

    # set the annotations
    props = dict(boxstyle='round', facecolor='white', alpha=1)

    for horizontal, vertical in zip(x, height):

        ax.text(
            horizontal-len(str(vertical))/60,
            vertical+65,
            str(vertical),
            fontsize=10,
            bbox=props)


# set the y limits so the legend appears above the bars
ax.set_ylim(0, df.to_numpy().max()*1.25)

# relabel the x axis
ax.set_xticks(x_axis)                    # offset values
ax.set_xticklabels(df.index.to_list())  # set the labels for each group

# the legend can be set to multiple values. 'Best' has Matplotlib automatically set the location.
# setting ncol to the length of the dataframe columns sets the legend horizontally by the length
# of the columns

plt.legend(loc = 'best', ncol=len(df.columns), fontsize = 16)
plt.show()
```

```
Low FS
Marginal/High FS
Very Low FS
```



####From the plot we can see the above visualization that there is a very high food insecurity in the UTEP campus, from the visualization of the about 63.5 percentage from those who answered the questions have very low food security and about 36.5 percent has very low food security. We can infer from this visualization that a lot of students have very high food insecurity. When we grouped the government aid into into categories we also notice from the visualization we can see that the students who work and study have very high security and those who take emergency loan were have low food security accross the three.

INTRODUCTION

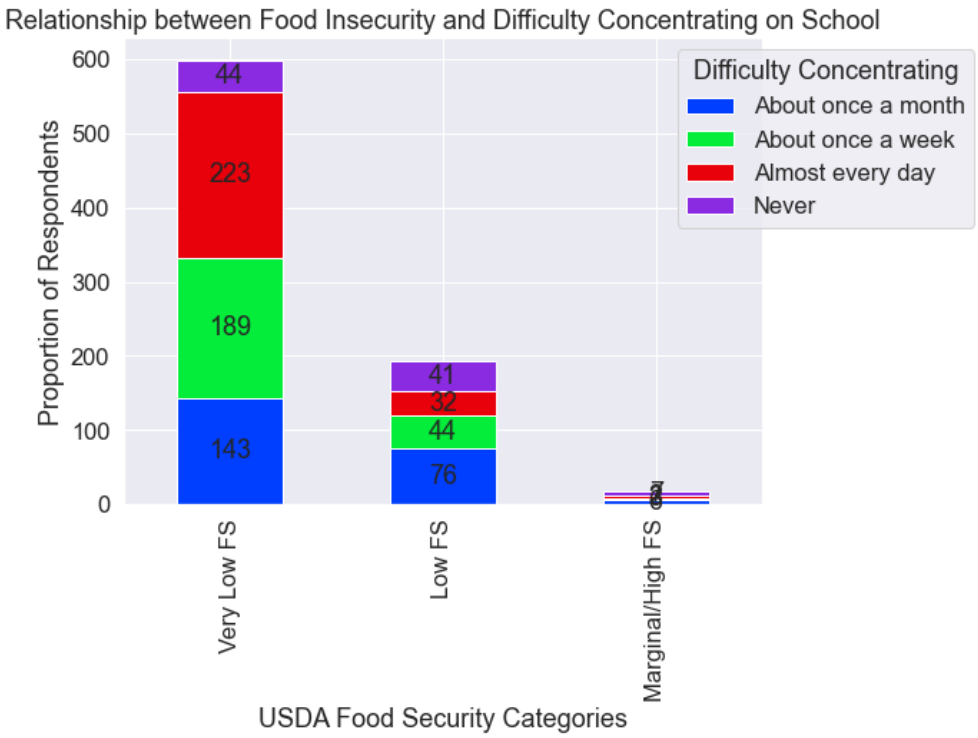
We wish to use visualization to show our audience if food insecurities have a relationship with the programe or the degree completion.

Does food insecurity (as measured by USDA index or categories) have a relationship with the items pertaining to concentration on school and degree progress/completion?

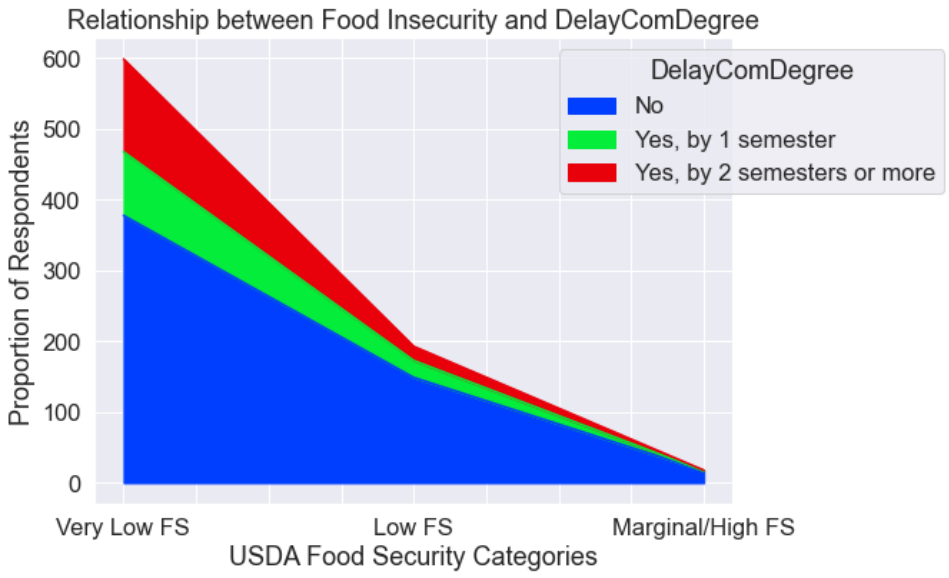
In [489]:
```
# Load data1
extra = pd.read_csv('extra_questions_withID.csv')

# Load data2
master = pd.read_csv('master.csv')
```

In [494]:
```
# Select the relevant columns from each dataset
dat1 = extra.loc[:, ['RespondentId', 'DiffConcentrate','DelayComplDegree', 'TimeDelayComplDegree']]
df2 = data2.loc[:, ['RespondentId', 'index']]

# Merge the two dataframes based on the common 'RespondentId' values
df = pd.merge(df1, df2, on='RespondentId', how='inner')
df.dropna(subset=['index'], inplace=True)
```

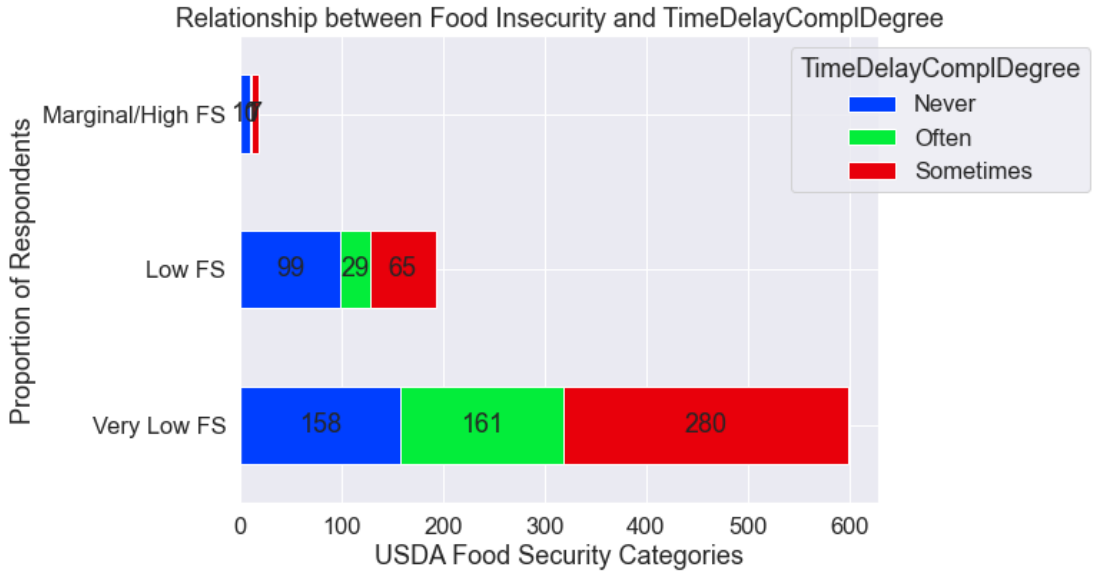
In [495]:
```
# Define the conditions and corresponding replacements
conditions = [
    (df['index'] >= 0) & (df['index'] <= 1),
    (df['index'] >= 2) & (df['index'] <= 4),
    (df['index'] >= 5) & (df['index'] <= 6)
]
replacements = [
    'Marginal/High FS',
    'Low FS',
    'Very Low FS'
]

# Apply the replacements using numpy.select()
df['index'] = np.select(conditions, replacements, default='na')
# Print the updated DataFrame


# filter relevant variables
food_security = df[['index']]
concentration = df[['DiffConcentrate']]

# merge data
merged_data = pd.concat([food_security, concentration], axis=1)

# group data by food security status and difficulty concentrating
grouped_data = merged_data.groupby(['index', 'DiffConcentrate']).size().reset_index(name='count')

# pivot the data
pivoted_data = grouped_data.pivot(index='index', columns='DiffConcentrate', values='count')


# sort the columns alphabetically
pivoted_data = pivoted_data.reindex(sorted(pivoted_data.columns), axis=1)

# define a custom sorting function
def custom_sort(label):
    if 'V' in label:
        return 0
    elif 'L' in label:
        return 1
    elif 'M' in label:
        return 2
    else:
        return 3

# sort the index using the custom function
pivoted_data = pivoted_data.iloc[pivoted_data.index.map(custom_sort).argsort()]

# plot the stacked bar chart
ax = pivoted_data.plot(kind='bar', stacked=True, figsize=(8,6))

# add data labels
for p in ax.containers:
    ax.bar_label(p, label_type='center')

# set plot properties
plt.title('Relationship between Food Insecurity and Difficulty Concentrating on School')
plt.xlabel('USDA Food Security Categories')
plt.ylabel('Proportion of Respondents')
plt.legend(title='Difficulty Concentrating', bbox_to_anchor=(1.35,1))
plt.show()
```


Relationship between Food Insecurity and Difficulty Concentrating on School
Difficulty Concentrating
About once a month
About once a week
Almost every day
Never
Proportion of Respondents
600
500
400
300
200
100
0
44
223
189
143
41
32
44
76
Very Low FS
Low FS
Marginal/High FS
USDA Food Security Categories

In [492]:
```
# filter relevant variables
food_security = df[['index']]
concentration = df[['DelayComplDegree']]

# merge data
merged_data = pd.concat([food_security, concentration], axis=1)

# group data by food security status and difficulty concentrating
grouped_data = merged_data.groupby(['index', 'DelayComplDegree']).size().reset_index(name='count')

# pivot the data
pivoted_data = grouped_data.pivot(index='index', columns='DelayComplDegree', values='count')


# sort the columns alphabetically
pivoted_data = pivoted_data.reindex(sorted(pivoted_data.columns), axis=1)

# define a custom sorting function
def custom_sort(label):
    if 'V' in label:
        return 0
    elif 'L' in label:
        return 1
    elif 'M' in label:
        return 2
    else:
        return 3

# sort the index using the custom function
pivoted_data = pivoted_data.iloc[pivoted_data.index.map(custom_sort).argsort()]

# plot the stacked bar chart
ax = pivoted_data.plot(kind='area', stacked=True, figsize=(8,6))

# add data labels
for p in ax.containers:
    ax.bar_label(p, label_type='center')

# set plot properties
plt.title('Relationship between Food Insecurity and DelayComDegree')
plt.xlabel('USDA Food Security Categories')
plt.ylabel('Proportion of Respondents')
plt.legend(title='DelayComDegree', bbox_to_anchor=(1.35,1))
plt.show()
```



In [493]:
```
# filter relevant variables
food_security = df[['index']]
concentration = df[['TimeDelayComplDegree']]

# merge data
merged_data = pd.concat([food_security, concentration], axis=1)

# group data by food security status and difficulty concentrating
grouped_data = merged_data.groupby(['index', 'TimeDelayComplDegree']).size().reset_index(name='count')

# pivot the data
pivoted_data = grouped_data.pivot(index='index', columns='TimeDelayComplDegree', values='count')


# sort the columns alphabetically
pivoted_data = pivoted_data.reindex(sorted(pivoted_data.columns), axis=1)

# define a custom sorting function
def custom_sort(label):
    if 'V' in label:
        return 0
    elif 'L' in label:
        return 1
    elif 'M' in label:
        return 2
    else:
        return 3

# sort the index using the custom function
pivoted_data = pivoted_data.iloc[pivoted_data.index.map(custom_sort).argsort()]

# plot the stacked bar chart
ax = pivoted_data.plot(kind='barh', stacked=True, figsize=(8,6))


# add data labels
for p in ax.containers:
    ax.bar_label(p, label_type='center')

# set plot properties
plt.title('Relationship between Food Insecurity and TimeDelayComplDegree')
plt.xlabel('USDA Food Security Categories')
plt.ylabel('Proportion of Respondents')
plt.legend(title='TimeDelayComplDegree', bbox_to_anchor=(1.35,1))
plt.show()
```



In [ ]:

INTRODUCTION

We are trying isualization to check if gender or ethnicity differences in he items perterning to concentration on school and degree affects progress or completion.

Are there gender or ethnicity differences in the items pertaining to concentration on school and degree progress/completion?

In [470]:
```
GenderF = pd.DataFrame(datF['Gender'].value_counts()).reset_index()
GenderF.columns = ['Gender', 'Freq']
GenderF = GenderF[~GenderF['Gender'].isna()]
GenderF['percent'] = round(GenderF['Freq'] / sum(GenderF['Freq']) * 100, 0)
```

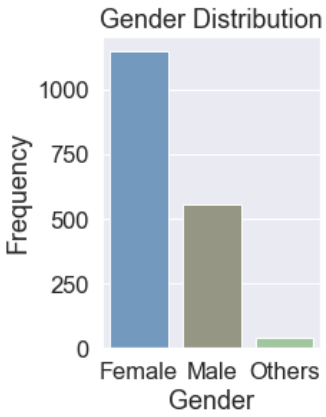
In [471]:
```
custom_palette = ["#6699CC", "#999981", "#99CC99"]
```

In [472]:
```
fig = plt.figure(figsize=(8, 4))
gs = gridspec.GridSpec(1, 2, width_ratios=[1, 1])
```

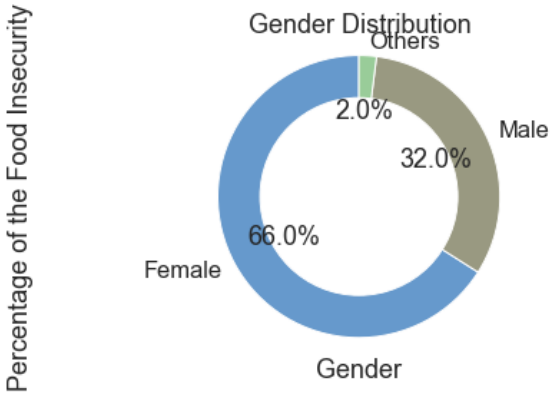
```
<Figure size 576x288 with 0 Axes>
```

In [473]:
```
ax0 = plt.subplot(gs[0])
sns.barplot(data=GenderF, x='Gender', y='Freq', palette=custom_palette)
ax0.set_xlabel('Gender')
ax0.set_ylabel('Frequency')
ax0.set_title('Gender Distribution')
```

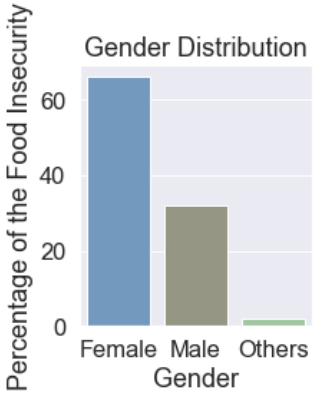
Out[473]:
```
Text(0.5, 1.0, 'Gender Distribution')
```



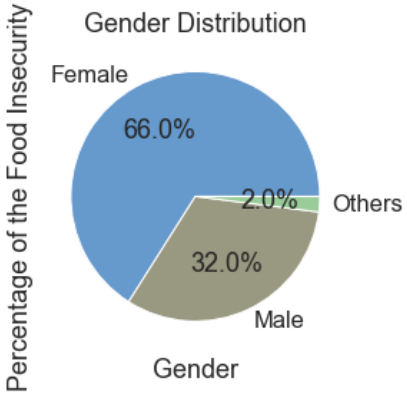
In [482]:
```
# Create a donut plot for gender distribution
plt.figure(figsize=(8, 4))

# Create outer pie chart
outer_colors = custom_palette
plt.pie(GenderF['percent'], labels=GenderF['Gender'], colors=outer_colors, autopct='%1.1f%%', startangle=90)

# Create inner circle for donut shape
center_circle = plt.Circle((0, 0), 0.7, color='white')
fig = plt.gcf()
fig.gca().add_artist(center_circle)

# Set aspect ratio to equal to make it a donut shape
plt.axis('equal')

# Add labels and title
plt.xlabel('Gender')
plt.ylabel('Percentage of the Food Insecurity')
plt.title('Gender Distribution')

plt.show()
```



In [474]:
```
ax1 = plt.subplot(gs[1])
sns.barplot(data=GenderF, x='Gender', y='percent', palette=custom_palette)
ax1.set_xlabel('Gender')
ax1.set_ylabel('Percentage of the Food Insecurity')
ax1.set_title('Gender Distribution')

plt.tight_layout()
plt.show()
```



In [481]:
```
# Create a pie chart for gender distribution
plt.figure(figsize=(8, 4))
plt.pie(GenderF['percent'], labels=GenderF['Gender'], colors=custom_palette, autopct='%1.1f%%')
plt.xlabel('Gender')
plt.ylabel('Percentage of the Food Insecurity')
plt.title('Gender Distribution')
plt.show()
```



In [475]:
```
gender_ConInSch = datF.groupby(['Gender', 'DiffConcentrate']).size().reset_index(name='Freq')
gender_ConInSch = gender_ConInSch[~((gender_ConInSch['Gender'] == "NA") | (gender_ConInSch['DiffConcentrate'] =
```

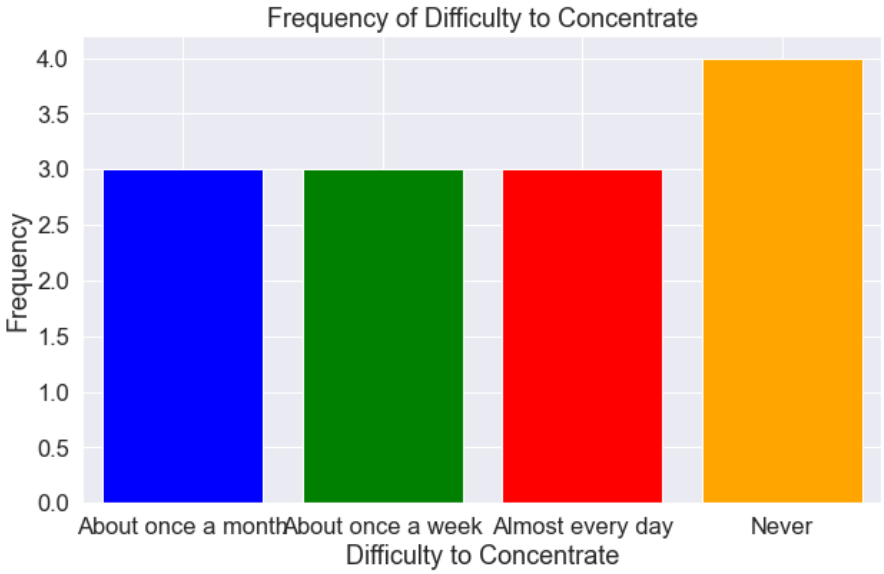
In [477]:
```
total_freq = gender_ConInSch.groupby('Gender')['Freq'].sum()

total_freq
```

Out[477]:
```
Gender
Female    1133
Male       543
Others      39
Name: Freq, dtype: int64
```

In [480]:
```
gender_ConInSch['DiffConcentrate'] = pd.Categorical(gender_ConInSch['DiffConcentrate'],
                                                    categories=["Almost every day", "About once a week", "Abou
                                                    ordered=True)

gender_DIff=gender_ConInSch['DiffConcentrate']

gender_DIff
```

Out[480]:
```
0     About once a month
1      About once a week
2       Almost every day
3                  Never
4     About once a month
5      About once a week
6       Almost every day
7                  Never
8     About once a month
9      About once a week
10      Almost every day
11                 Never
Name: DiffConcentrate, dtype: category
Categories (4, object): ['Almost every day' < 'About once a week' < 'About once a month' < 'Never']
```

In [488]:
```
import matplotlib.pyplot as plt

# Data
difficulties = ['About once a month', 'About once a week', 'Almost every day', 'Never']
frequencies = [3, 3, 3, 4]

# Create a larger figure
plt.figure(figsize=(10, 6))  # Adjust the width and height as desired

# Define custom colors for each bar
colors = ['blue', 'green', 'red', 'orange']

# Create a bar chart with custom colors
plt.bar(difficulties, frequencies, color=colors)

# Add labels and title
plt.xlabel('Difficulty to Concentrate')
plt.ylabel('Frequency')
plt.title('Frequency of Difficulty to Concentrate')

# Extend the x-axis limits
plt.xlim(-0.5, len(difficulties) - 0.5)

# Display the plot
plt.show()
```



It evident that some students experience some difficulties in concentration, the female normally experience more difficulties than the male student. On futher research I also released that higher hispanic student experience high food insecurity than any other ethnicity which make sense because there is a higher hispanic population in the UTEP campus compare to other ethnicity so the ratio make some sense.