In [1]:
```
import pandas as pd
from datetime import datetime,timedelta, date
import copy
import statistics
```

In [2]:
```
def Last_Office_hrs(date,class_name):
    pd = Office_Hours[Office_Hours['Class ']== class_name]
    pd = pd[pd['Available Office Hrs']<date]
    pd = pd[pd['Available Office Hrs']== max(pd['Available Office Hrs'])]
    return pd.iloc[0]['Available Office Hrs']
```

In [3]:
```
# Loading the Datasets
Working_on_Assignments = pd.read_excel("Courses.xlsx", sheet_name="Working on Assignments")
Completed_Assignments = pd.read_excel("Courses.xlsx", sheet_name="Completed Assignments")
Office_Hours = pd.read_excel("Courses.xlsx", sheet_name="Office Hours")
Grade_Requirements = pd.read_excel("Courses.xlsx", sheet_name="Grade Requirements")
Grading_Rubic = pd.read_excel("Courses.xlsx", sheet_name="Grading Rubic")
Available_Time = pd.read_excel("Courses.xlsx", sheet_name="Available Time")
```

In [4]:
```
# Removing Completed Assignments
Working_on_Assignments = Working_on_Assignments[Working_on_Assignments['Total Time (mins)']!='NaN']

# Filtering Assignments due in the next few weeks
today = datetime.today()
one_week = today+ timedelta(days=7)
One_Week_Assignments = Working_on_Assignments[Working_on_Assignments['Due Date']<=one_week]
Missed_Assignments = One_Week_Assignments[One_Week_Assignments['Due Date']<=today]
One_Week_Assignments = One_Week_Assignments[One_Week_Assignments['Due Date']>today]


# Filtering for selected columns
#Class no space
One_Week_Assignments = One_Week_Assignments[['Class','Type' ,'Due Date', 'Estimated (mins)']]
One_Week_Assignments = One_Week_Assignments.sort_values(by=['Due Date'])
One_Week_Assignments
```

Out[4]:

	Class	Type	Due Date	Estimated (mins)
2	Econometrics	Content	2023-09-25	120
6	Business Analytics	Content Part 1	2023-09-25	120
7	Business Analytics	Content Part 2	2023-09-25	120
10	Applied Integrative Projects	Completion of Weekly Checklist	2023-09-25	60
11	Applied Integrative Projects	HW	2023-09-25	60

In [5]:
```
# Creating Frame and adding Date
weekday_num = today.weekday()
Weekday_dict = {0 : 'Monday',
                1 : 'Tuesday',
                2 : 'Wednesday',
```

```
                3 : 'Thursday',
                4 : 'Friday',
                5 : 'Saturday',
                6 : 'Sunday'}

Planner_Dict = {}
# Looping through days of the week
for num in range(7):
    next_day = today+ timedelta(days=num)
    next_day_num = next_day.weekday()
    next_day_weekday = Weekday_dict[next_day_num]
    Planner_Dict[next_day_weekday] = {'Date':next_day}

# Planner_Dict
```

In [6]:
```
# Adding Time Limits
Available_Time_Dict = Available_Time.to_dict('records')
Available_Time_Dict
for Days in Available_Time_Dict:
    weekday = Days['Days ']
#     Planner_Dict[weekday]= {}
    Preferred = Days['Preferred Time']
    Available = Days['Available Time']
    if Preferred >= 3:
        factor = (Preferred/3) * 1 #every 3 hrs rewards a 1 hr break
        Preferred = Preferred-factor
    if Available >= 3:
        factor = (Available/3) * 1 #every 3 hrs rewards a 1 hr break
        Available = Available-factor

    Planner_Dict[weekday]['Preferred']= round(Preferred,2)
    Planner_Dict[weekday]['Available']= round(Available,2)
    Planner_Dict[weekday]['Before_OH_Points'] = 0
    Planner_Dict[weekday]['Before_OH_Max'] = 0
    Planner_Dict[weekday]['Time Estimated'] = 0
    Planner_Dict[weekday]['Assignments'] = []

    Planner_Dict['Equallity_Score'] = 0
    Planner_Dict['Overall_Score'] = 0

# Planner_Dict
```

In [7]:
```
Office_Hours = Office_Hours[Office_Hours['Available Office Hrs']<=one_week]
Office_Hours = Office_Hours[Office_Hours['Available Office Hrs']>=today]
Office_Hours_Dict = Office_Hours.to_dict('records')
# Office_Hours_Dict
```

In [8]:
```
One_Week_Assignments_Dict = One_Week_Assignments.to_dict('records')
Weekly_MAX_TIME = 0
for each in One_Week_Assignments_Dict:
    Weekly_MAX_TIME = each['Estimated (mins)']+Weekly_MAX_TIME

Weekly_MAX_TIME = Weekly_MAX_TIME/60
```

```
# Weekly_MAX_TIME
One_Week_Assignments
```

Out[8]:

	Class	Type	Due Date	Estimated (mins)
2	Econometrics	Content	2023-09-25	120
6	Business Analytics	Content Part 1	2023-09-25	120
7	Business Analytics	Content Part 2	2023-09-25	120
10	Applied Integrative Projects	Completion of Weekly Checklist	2023-09-25	60
11	Applied Integrative Projects	HW	2023-09-25	60

In [9]:

```
Master_List = [Planner_Dict]
One_Week_Assignments_Dict = copy.deepcopy(One_Week_Assignments.to_dict('records'))
Dummy_List = []
First = True


for num in range(len(One_Week_Assignments_Dict)):
    Assignment = One_Week_Assignments_Dict[num]
    Dummy_List2 = []
    if Dummy_List == [] and First == True:
        First = False
        Dummy_List = copy.deepcopy(Master_List)
    for Planner in Dummy_List:
        for day in Planner:
            Dummy_Dict = copy.deepcopy(Planner)
            try:

                Time_for_assignment = (Assignment['Estimated (mins)']/60)* 1.20 # applying a 20% plus buffer
                Total_time = Dummy_Dict[day]['Time Estimated'] + Time_for_assignment

                # Hard-Cuts offs
                if Total_time<=Dummy_Dict[day]['Preferred'] and Dummy_Dict[day]['Date'] <= (Assignment['Due Date']):

                    Dummy_Dict[day]['Assignments'].append(Assignment)
                    Dummy_Dict[day]['Time Estimated'] = Total_time
                    try:
                        Office_Hrs = Last_Office_hrs(Assignment['Due Date'], Assignment['Class'])
                        if Dummy_Dict[day]['Date'] < Office_Hrs:
                            Dummy_Dict[day]['Before_OH_Points'] = Dummy_Dict[day]['Before_OH_Points'] + 1

                    except ValueError as err:
                        x = 1
                    Dummy_Dict[day]['Before_OH_Max'] = Dummy_Dict[day]['Before_OH_Max'] + 1

                    score = []
                    Worst_Case_list = []
                    Current_Case_list = []

                    for day in Dummy_Dict:
                        try:
```

```
                            if Dummy_Dict[day]['Before_OH_Max'] != 0:
                                score.append(Dummy_Dict[day]['Before_OH_Points']/Dummy_Dict[day]['Before_OH_Max'])

                            if Dummy_Dict[day]['Preferred'] != 0:
                                Current_Case_list.append(Dummy_Dict[day]['Time Estimated'])
                                if Worst_Case_list ==[]:
                                    Worst_Case_list.append(Weekly_MAX_TIME)
                                else:
                                    Worst_Case_list.append(0)

                        except TypeError as err:
                            x=1
                    OH_score = statistics.mean(score)

                    WC_STD = statistics.pstdev(Worst_Case_list)
                    C_STD = statistics.pstdev(Current_Case_list)
                    Range_of_STD = (WC_STD-C_STD)/WC_STD

                    Dummy_Dict['Overall_Score'] = OH_score*1 +Range_of_STD*6


                    Dummy_List2.append(Dummy_Dict)
                    Check_Assignment = Assignment
            except TypeError as err:
                x =1
    Dummy_List2 = sorted(Dummy_List2, key=lambda d: d['Overall_Score'], reverse = True)
#     Dummy_List = copy.deepcopy(Dummy_List2[1:round(len(Dummy_List2)/1.20,)]) # filters out the bottom 20%
    Dummy_List = copy.deepcopy(Dummy_List2)
# Making sure all Assignment has been assigned
if Check_Assignment != One_Week_Assignments_Dict[-1]:
    print("error not enought time alloted for homework to be done in time")
    x = 1/0

Dummy_List[0]
```

Out[9]:
```
{'Tuesday': {'Date': datetime.datetime(2023, 9, 19, 12, 45, 33, 76279),
  'Preferred': 5.33,
  'Available': 6.67,
  'Before_OH_Points': 1,
  'Before_OH_Max': 1,
  'Time Estimated': 2.4,
  'Assignments': [{'Class': 'Business Analytics',
    'Type': 'Content Part 1 ',
    'Due Date': Timestamp('2023-09-25 00:00:00'),
    'Estimated (mins)': 120}]},
 'Wednesday': {'Date': datetime.datetime(2023, 9, 20, 12, 45, 33, 76279),
  'Preferred': 5.33,
  'Available': 8.67,
  'Before_OH_Points': 0,
  'Before_OH_Max': 1,
  'Time Estimated': 2.4,
  'Assignments': [{'Class': 'Econometrics',
    'Type': 'Content',
    'Due Date': Timestamp('2023-09-25 00:00:00'),
    'Estimated (mins)': 120}]},
 'Thursday': {'Date': datetime.datetime(2023, 9, 21, 12, 45, 33, 76279),
  'Preferred': 5.33,
```

```
 'Available': 6.67,
 'Before_OH_Points': 0,
 'Before_OH_Max': 1,
 'Time Estimated': 2.4,
 'Assignments': [{'Class': 'Business Analytics',
   'Type': 'Content Part 2',
   'Due Date': Timestamp('2023-09-25 00:00:00'),
   'Estimated (mins)': 120}]},
'Friday': {'Date': datetime.datetime(2023, 9, 22, 12, 45, 33, 76279),
 'Preferred': 5.33,
 'Available': 8.67,
 'Before_OH_Points': 0,
 'Before_OH_Max': 2,
 'Time Estimated': 2.4,
 'Assignments': [{'Class': 'Applied Integrative Projects ',
   'Type': 'Completion of Weekly Checklist',
   'Due Date': Timestamp('2023-09-25 00:00:00'),
   'Estimated (mins)': 60},
  {'Class': 'Applied Integrative Projects ',
   'Type': 'HW',
   'Due Date': Timestamp('2023-09-25 00:00:00'),
   'Estimated (mins)': 60}]},
'Saturday': {'Date': datetime.datetime(2023, 9, 23, 12, 45, 33, 76279),
 'Preferred': 0,
 'Available': 8.67,
 'Before_OH_Points': 0,
 'Before_OH_Max': 0,
 'Time Estimated': 0,
 'Assignments': []},
'Sunday': {'Date': datetime.datetime(2023, 9, 24, 12, 45, 33, 76279),
 'Preferred': 0,
 'Available': 8.67,
 'Before_OH_Points': 0,
 'Before_OH_Max': 0,
 'Time Estimated': 0,
 'Assignments': []},
'Monday': {'Date': datetime.datetime(2023, 9, 25, 12, 45, 33, 76279),
 'Preferred': 5.33,
 'Available': 6.67,
 'Before_OH_Points': 0,
 'Before_OH_Max': 0,
 'Time Estimated': 0,
 'Assignments': []},
'Equallity_Score': 0,
'Overall_Score': 4.45}
```

In [10]:
```
num = 0
```

In [11]:
```
string = ''

for days in Dummy_List[num]:
    if days != 'Equallity_Score' and days != 'Overall_Score':
        try:
            if Dummy_List[num][days]['Preferred'] != 0:
                string = ('{} {} {}\n'.format(string, days, Dummy_List[num][days]['Date']))
                string = ('{}     The excepting time spent studying in hours: {}\n'.format(string,round(Dummy_List[num][days]['Time Estimated'],2)))
                for items in Dummy_List[num][days]['Assignments']:
#                     print(items)
```

```
                    string = ('{}     Class: {}\n'.format(string, items['Class']))
                    string = ('{}            Type: {}\n '.format(string, items['Type']))
                    string = ('{}          Due Date: {}\n '.format(string,items['Due Date']))
                    string = ('{}          Estimated (mins): {}\n'.format(string,items['Estimated (mins)']))

        except TypeError as err:
            x=1
#             print(days)
print(string)
```

```
Tuesday 2023-09-19 12:45:33.076279
    The excepting time spent studying in hours: 2.4
    Class: Business Analytics
           Type: Content Part 1
           Due Date: 2023-09-25 00:00:00
           Estimated (mins): 120
Wednesday 2023-09-20 12:45:33.076279
    The excepting time spent studying in hours: 2.4
    Class: Econometrics
           Type: Content
           Due Date: 2023-09-25 00:00:00
           Estimated (mins): 120
Thursday 2023-09-21 12:45:33.076279
    The excepting time spent studying in hours: 2.4
    Class: Business Analytics
           Type: Content Part 2
           Due Date: 2023-09-25 00:00:00
           Estimated (mins): 120
Friday 2023-09-22 12:45:33.076279
    The excepting time spent studying in hours: 2.4
    Class: Applied Integrative Projects
           Type: Completion of Weekly Checklist
           Due Date: 2023-09-25 00:00:00
           Estimated (mins): 60
    Class: Applied Integrative Projects
           Type: HW
           Due Date: 2023-09-25 00:00:00
           Estimated (mins): 60
Monday 2023-09-25 12:45:33.076279
    The excepting time spent studying in hours: 0
```

In [12]:
```
Missed_Assignments
```

Out[12]:

	Class	Type	Due Date	Time	Estimated (mins)	Total Time (mins)	Grade
0	Business Analytics	Content Part 1	2023-09-18	18:00:00	120	NaN	NaN
1	Business Analytics	Content Part 2	2023-09-18	18:00:00	120	NaN	NaN
3	Business Analytics	HW	2023-09-17	12:59:00	90	NaN	NaN
4	Applied Integrative Projects	Completion of Weekly Checklist	2023-09-18	18:00:00	60	NaN	NaN
5	Applied Integrative Projects	HW	2023-09-18	18:00:00	60	NaN	NaN