CS 649 Big Data: Tools and Methods
Spring Semester, 2022
Doc 17 Assignment 1
Mar 8, 2022

```
dfff = df[["Building","Room","Subject"]].groupby(["Building","Room"]).count()
dfff = dfff.rename(columns={"Subject":"num_of_classes"})
#df.groupby(["Building","Room"]).count().reset_index()
dfff
```

```
import pandas as pd
import numpy as np

schedule = pd.read_table("search.tsv", sep='\t', skiprows=[0])
sch_df = pd.DataFrame(schedule)
sch_df
```

```
#for name,group in df1d:
    #print(name)
    #print(group)

df1a = df[["Start Time", "Course #", "Suffix", "Days"]].copy()
df1b = df1a.loc[(df1a["Days"] != "ARR") & (df1a["Days"] != "NaN")]
#df1d
df1b = df1b.groupby(["Start Time", "Days", "Course #"])
#df1d
for name,group in df1b:
    print(name)
    print(group)
#filter1 = df1a["Start Time"] == 800
#filter2 = df1a["Days"] == "T"
#filter3 = df1a["Course #"] < 200
#df1e = df1a.where(filter1)
#display(df1e.dropna())
```

What is df1b
Why overwrite it?

```
df1a = df[["Start Time", "Course #", "Suffix", "Days"]].copy()
df1b = df1a.loc[(df1a["Days"] != "ARR") & (df1a["Days"] != "NaN")]

df1b = df1b.groupby(["Start Time", "Days", "Course #"])

for name,group in df1b:
    print(name)
    print(group)
```

```
start_col = "Start Time"
days_col = "Days"
course_col = "Course #"
suffix_col = "Suffix"
q1_added_col = "Full Course Number"

# Get rid of on-line listings (same schedule # but no meeting time)
q1_data = df.loc[df[start_col].notna()]

# Drop duplicate rows from dataframe and add new col for full course name
cols_to_keep = [course_col, suffix_col, start_col, days_col]

q1_df = q1_data.sort_values(start_col).drop_duplicates()[cols_to_keep]
q1_df[q1_added_col] = q1_df[course_col].astype(str) + q1_df[suffix_col]

# Convert "Start Time" column to ints (as opposed to floats)
q1_df[start_col] = q1_df[start_col].astype(int)
```

```
start_col = "Start Time"
days_col = "Days"
course_col = "Course #"
suffix_col = "Suffix"
full_course_col = "Full Course Number"
cols_to_keep = [course_col, suffix_col, start_col, days_col]

def remove_online(schedule_df):
    #online course have no meeting time
    return schedule_df.loc[df[start_col].notna()
inperson_classes = remove_online(schedule_df)

# Drop duplicate rows from dataframe and add new col for full course name



q1_df = q1_data.sort_values(start_col).drop_duplicates()[cols_to_keep]
q1_df[q1_added_col] = q1_df[course_col].astype(str) + q1_df[suffix_col]

# Convert "Start Time" column to ints (as opposed to floats)
q1_df[start_col] = q1_df[start_col].astype(int)
```

What is going on here?

```
course_info.columns = course_info.loc[0,:]
course_info = course_info.loc[1:,:]
course_info = course_info.loc[(course_info.Days != 'ARR') & (~course_info.Days.isnull()),:]
course_info.head()
```

```
dff = df.groupby(["Title"]).total_enrolled.sum().reset_index()
dff
```

What is:

```
df
dff
```

```
course_enrollment = class_schedule.groupby(["Title"]).total_enrolled.sum().reset_index()
course_enrollment
```

```
search_sheet = search_sheet[['Course #', 'Suffix', 'Start Time', 'Days']].copy()


 search_sheet['Intro MW'] = [str(search_sheet.loc[row, 'Course #']) + 'L'
                             if int(search_sheet.loc[row, 'Course #']) < 200 and
                                str(search_sheet.loc[row, 'Days']) == 'MW' and
                                str(search_sheet.loc[row, 'Suffix']) == 'L'
                             else str(search_sheet.loc[row, 'Course #'])
                             if int(search_sheet.loc[row, 'Course #']) < 200 and
                               str(search_sheet.loc[row, 'Days']) == 'MW'
                             else ''
                             for row in range(len(search_sheet['Course #']))]
```

```
search_sheet['Intro TTH'] = [str(search_sheet.loc[row, 'Course #']) + 'L'
                             if int(search_sheet.loc[row, 'Course #']) < 200 and
                                str(search_sheet.loc[row, 'Days']) == 'TTH' and
                                str(search_sheet.loc[row, 'Suffix']) == 'L'
                             else str(search_sheet.loc[row, 'Course #'])
                             if int(search_sheet.loc[row, 'Course #']) < 200 and
                                str(search_sheet.loc[row, 'Days']) == 'TTH'
                             else ''
                             for row in range(len(search_sheet['Course #']))]
```

```
search_sheet['Lower MW'] = [str(search_sheet.loc[row, 'Course #']) + 'L'
                            if 200 <= int(search_sheet.loc[row, 'Course #']) < 500 and
                              str(search_sheet.loc[row, 'Days']) == 'MW' and
                              str(search_sheet.loc[row, 'Suffix']) == 'L'
                            else str(search_sheet.loc[row, 'Course #'])
                            if 200 <= int(search_sheet.loc[row, 'Course #']) < 500 and
                               str(search_sheet.loc[row, 'Days']) == 'MW'
                            else ''
                            for row in range(len(search_sheet['Course #']))]
```

```
def select_course_time(schedule, start_level = 0, end_level = 800, days):
        str(schedule.loc[row, 'Course #']) + 'L'
        if start_level <= int(schedule.loc[row, 'Course #']) < end_level and \
           str(schedule.loc[row, 'Days']) == days and \
           str(schedule.loc[row, 'Suffix']) == 'L'\
         else str(schedule.loc[row, 'Course #'])
         if 200 <= int(schedule.loc[row, 'Course #']) < 500 and \
            str(schedule.loc[row, 'Days']) == days\
          else ''
          for row in range(len(schedule['Course #']))
```

```
search_sheet['Intro MW'] = [select_course(search_sheet, end_level = 200, days = 'MW')]
search_sheet['Intro TTH'] = [select_course(search_sheet, end_level = 200, days = 'TH')]
search_sheet['Lower MW'] = [select_course(search_sheet, 200, 500, 'MW')]
search_sheet['Lower TTH'] = [select_course(search_sheet, 200, 500,  'TH')]
```

I I

```
search_sheet = search_sheet_raw.copy()

search_sheet['Suffix'] = search_sheet['Suffix'].fillna('AA').astype(str)

# Sorts classes by number and suffix so classes appear in sorted order and labs appear after normal classes

search_sheet = search_sheet.sort_values(by=['Course #', 'Suffix'], ascending=[True, True]).reset_index()

# Converts single day classes to either MW or TTH respectively

search_sheet['Days'] = ['MW' if 'M' in str(row) or 'W' in str(row) else

                        'TTH' if 'T' in str(row) or 'TH' in str(row) else

                        '' for row in search_sheet['Days']]

# Removes classes with multiple sections

search_sheet = search_sheet.drop_duplicates(subset=['Course #', 'Suffix', 'Start Time', 'Days']).reset_index()


search_sheet = search_sheet[['Course #', 'Suffix', 'Start Time', 'Days']].copy()


# Sorts classes by intro, lower, and upper and by day into different columns

search_sheet['Intro MW'] = [str(search_sheet.loc[row, 'Course #']) + 'L'

                            if int(search_sheet.loc[row, 'Course #']) < 200 and str(search_sheet.loc[row, 'Days']) == 'MW' and str(search_sheet.loc[row, 'Suffix']) == 'L'

                            else str(search_sheet.loc[row, 'Course #'])

                            if int(search_sheet.loc[row, 'Course #']) < 200 and str(search_sheet.loc[row, 'Days']) == 'MW'

                            else ''

                            for row in range(len(search_sheet['Course #']))]


search_sheet['Intro TTH'] = [str(search_sheet.loc[row, 'Course #']) + 'L'

                             if int(search_sheet.loc[row, 'Course #']) < 200 and str(search_sheet.loc[row, 'Days']) == 'TTH' and str(search_sheet.loc[row, 'Suffix']) == 'L'

                             else str(search_sheet.loc[row, 'Course #'])

                             if int(search_sheet.loc[row, 'Course #']) < 200 and str(search_sheet.loc[row, 'Days']) == 'TTH'

                             else ''

                             for row in range(len(search_sheet['Course #']))]


search_sheet['Lower MW'] = [str(search_sheet.loc[row, 'Course #']) + 'L'

                            if 200 <= int(search_sheet.loc[row, 'Course #']) < 500 and str(search_sheet.loc[row, 'Days']) == 'MW' and str(search_sheet.loc[row, 'Suffix']) == 'L'

                            else str(search_sheet.loc[row, 'Course #'])

                            if 200 <= int(search_sheet.loc[row, 'Course #']) < 500 and str(search_sheet.loc[row, 'Days']) == 'MW'

                            else ''

                            for row in range(len(search_sheet['Course #']))]


search_sheet['Lower TTH'] = [str(search_sheet.loc[row, 'Course #']) + 'L'

                             if 200 <= int(search_sheet.loc[row, 'Course #']) < 500 and str(search_sheet.loc[row, 'Days']) == 'TTH' and str(search_sheet.loc[row, 'Suffix']) == 'L'

                             else str(search_sheet.loc[row, 'Course #'])

                             if 200 <= int(search_sheet.loc[row, 'Course #']) < 500 and str(search_sheet.loc[row, 'Days']) == 'TTH'

                             else ''

                             for row in range(len(search_sheet['Course #']))]


search_sheet['Upper MW'] = [str(search_sheet.loc[row, 'Course #']) + 'L'

                            if int(search_sheet.loc[row, 'Course #']) >= 500 and str(search_sheet.loc[row, 'Days']) == 'MW' and str(search_sheet.loc[row, 'Suffix']) == 'L'

                            else str(search_sheet.loc[row, 'Course #'])

                            if int(search_sheet.loc[row, 'Course #']) >= 500 and str(search_sheet.loc[row, 'Days']) == 'MW'

                            else ''

                            for row in range(len(search_sheet['Course #']))]


search_sheet['Upper TTH'] = [str(search_sheet.loc[row, 'Course #']) + 'L'

                             if int(search_sheet.loc[row, 'Course #']) >= 500 and str(search_sheet.loc[row, 'Days']) == 'TTH' and str(search_sheet.loc[row, 'Suffix']) == 'L'

                             else str(search_sheet.loc[row, 'Course #'])

                             if int(search_sheet.loc[row, 'Course #']) >= 500 and str(search_sheet.loc[row, 'Days']) == 'TTH'

                             else ''

                             for row in range(len(search_sheet['Course #']))]



search_sheet = search_sheet[['Start Time', 'Intro MW', 'Intro TTH', 'Lower MW',

                             'Lower TTH', 'Upper MW', 'Upper TTH']

                            ].copy()

search_sheet.columns = ['Time', 'Intro MW', 'Intro TTH', 'Lower MW',

                        'Lower TTH', 'Upper MW', 'Upper TTH']

search_sheet['Time'] = search_sheet['Time'].fillna(0.0).astype(int)


# Groups separated class columns by time

search_sheet_intro_mw = search_sheet.groupby(['Time'])['Intro MW'].apply(','.join).drop([0]).reset_index().set_index('Time', drop=True)

search_sheet_intro_tth = search_sheet.groupby(['Time'])['Intro TTH'].apply(','.join).drop([0]).reset_index().set_index('Time', drop=True)

search_sheet_lower_mw = search_sheet.groupby(['Time'])['Lower MW'].apply(','.join).drop([0]).reset_index().set_index('Time', drop=True)

search_sheet_lower_tth = search_sheet.groupby(['Time'])['Lower TTH'].apply(','.join).drop([0]).reset_index().set_index('Time', drop=True)

search_sheet_upper_mw = search_sheet.groupby(['Time'])['Upper MW'].apply(','.join).drop([0]).reset_index().set_index('Time', drop=True)

search_sheet_upper_tth = search_sheet.groupby(['Time'])['Upper TTH'].apply(','.join).drop([0]).reset_index().set_index('Time', drop=True)


# Formats cells to remove extra commas

search_sheet_intro_mw['Intro MW'] = [re.sub(r',+', ', ', str(row)) for row in search_sheet_intro_mw['Intro MW']]

search_sheet_intro_tth['Intro TTH'] = [re.sub(r',+', ', ', str(row)) for row in search_sheet_intro_tth['Intro TTH']]

search_sheet_lower_mw['Lower MW'] = [re.sub(r',+', ', ', str(row)) for row in search_sheet_lower_mw['Lower MW']]

search_sheet_lower_tth['Lower TTH'] = [re.sub(r',+', ', ', str(row)) for row in search_sheet_lower_tth['Lower TTH']]

search_sheet_upper_mw['Upper MW'] = [re.sub(r',+', ', ', str(row)) for row in search_sheet_upper_mw['Upper MW']]

search_sheet_upper_tth['Upper TTH'] = [re.sub(r',+', ', ', str(row)) for row in search_sheet_upper_tth['Upper TTH']]


search_sheet_intro_mw = search_sheet_intro_mw['Intro MW'].apply(lambda row: row.strip(', ')).reset_index().set_index('Time', drop=True)

search_sheet_intro_tth = search_sheet_intro_tth['Intro TTH'].apply(lambda row: row.strip(', ')).reset_index().set_index('Time', drop=True)

search_sheet_lower_mw = search_sheet_lower_mw['Lower MW'].apply(lambda row: row.strip(', ')).reset_index().set_index('Time', drop=True)

search_sheet_lower_tth = search_sheet_lower_tth['Lower TTH'].apply(lambda row: row.strip(', ')).reset_index().set_index('Time', drop=True)

search_sheet_upper_mw = search_sheet_upper_mw['Upper MW'].apply(lambda row: row.strip(', ')).reset_index().set_index('Time', drop=True)

search_sheet_upper_tth = search_sheet_upper_tth['Upper TTH'].apply(lambda row: row.strip(', ')).reset_index().set_index('Time', drop=True)


# Merges grouped dataframes into a single dataframe

schedule_table = search_sheet_intro_mw.merge(search_sheet_intro_tth, on='Time')

schedule_table = schedule_table.merge(search_sheet_lower_mw, on='Time')

schedule_table = schedule_table.merge(search_sheet_lower_tth, on='Time')

schedule_table = schedule_table.merge(search_sheet_upper_mw, on='Time')

schedule_table = schedule_table.merge(search_sheet_upper_tth, on='Time')


schedule_table
```

```
# For task 2
def task_2(data):
    data_frame = pd.DataFrame(data)
    new_frame = {'Course' : data_frame['Course #'].astype(str) + " " \
                + data_frame['Title'],'No of Students': data_frame['Total Enrolled']}
output = pd.DataFrame(new_frame).groupby('Course').sum().astype(int)
return output
```

Course	Students
100COMPTR SCIENCE PRINCIPLES	38
150INTRO COMPUTER PROGRAMMNG	175
150INTRO COMPUTER PROGRM LAB	288
160INT COMPUTER PROGRAMMING	221
160INT COMPUTR PROGAMMNG LAB	430
200INTRO DATA SCIENCE PYTHON	22
210DATA STRUCTURES	64
240COMPUTER ORGANIZATION	65
250INTRO TO SOFTWARE SYSTEMS	57
299SPECIAL STUDY	0
320PROGRAMMING LANGUAGES	98

	Courses	Students
0	583 3D GAME PROGRAMMING	77.0
1	596 ADV 3D GAME PRGRMNG	27.0
2	460 ALGORITHMS	115.0
3	649 BIG DATA TOOLS & METHODS	60.0
4	799 COMPREHENSIVE EXAM	1.0
5	100 COMPTR SCIENCE PRINCIPLES	38.0
6	581 COMPUTATIONAL LINGUISTICS	15.0
7	370 COMPUTER ARCHITECTURE	127.0
8	240 COMPUTER ORGANIZATION	65.0
9	574 COMPUTER SECURITY	148.0

Course	
100 COMPTR SCIENCE PRINCIPLES	38
150 INTRO COMPUTER PROGRAMMNG	175
150 INTRO COMPUTER PROGRM LAB	288
160 INT COMPUTER PROGRAMMING	221
160 INT COMPUTR PROGAMMNG LAB	430
200 INTRO DATA SCIENCE PYTHON	22
210 DATA STRUCTURES	64
240 COMPUTER ORGANIZATION	65
250 INTRO TO SOFTWARE SYSTEMS	57
299 SPECIAL STUDY	0
320 PROGRAMMING LANGUAGES	98

Room	Course #
101	1
105	2
119	2
123	2
130	3
144	4
410	3
425	3
439	2
445	2
LINE	28

Building	Room	No. of classes
AH	2116	2
AL	101	1
	204	1
COM	105	1
	207	1
ENS	291	1
GMCS	305	1
	306	1
	308	2
	314	1
	324	1
	425	3

Level	Time	Course
0	930.0	200, 200
	1400.0	150, 160, 160
Intro MW	1600.0	160
	1730.0	150, 150, 160, 160, 160
	800.0	150, 150, 160, 160
	930.0	150
	1100.0	100, 160
Intro TH	1400.0	160
	1530.0	160, 160
	1600.0	150
	1730.0	150, 150
	1400.0	370
Lower MW	1600.0	210, 370
	1900.0	240
	800.0	210
Lower TTH	1400.0	240, 250, 320

Start Time	Intro MW	Intro TTH	Lower MW	Lower TTH	UpperMW	Upper TTH
800.0		150L,150L,160L	490	210		578
930.0		150	200	200		
1100.0		100 ,160L		480 ,496		549 ,581
1230.0				440		
1400.0	150 ,160 ,160L		160	370 ,490	240 ,250 ,320 ,460	578 ,582
1530.0			160L		320	
1600.0	160		150	210 ,370	480	532 ,574
1730.0	150L,150L,160 ,160L,160L		150L	460 ,490	440 ,480	530 ,696
1900.0				240 ,490 ,490		514 ,574 ,662

new_day_name Start Time	Intro MW	Intro TTH	Lower MW	Lower TTH
800.0	NaN	150L,150L,160L,160L	490	210,496
930.0	NaN	150	200	200
1100.0	NaN	100,160L	NaN	480,496
1230.0	NaN	NaN	NaN	440
1400.0	150,160L,160	160	370,490	240,250,320,460,460
1530.0	NaN	160L,160L	NaN	320

Problem 1

```
df = pd.read_csv('/Users/rwhitney/data/search.xls',sep = '\t', header = 1)
df = df.replace(np.nan,'',regex=True)
df = df.rename(columns={'Course #':'course_num','Total Enrolled':'total_enrolled','Start Time':'start_time'})
df["start_time"] = df.start_time.replace('', 0, regex=True)


 df['course'] = df['course_num'].astype(str) + df['Suffix'] + ' '
 mwlist = ['M','W','MW']
 tthlist = ['T', 'TH', 'TTH']


intro_mw_df = df[(df.course_num <200) & (df.Days.isin(mwlist))]
intro_tth_df = df[(df.course_num <200) & (df.Days.isin(tthlist))]
lower_mw_df = df[(df.course_num <500) & (df.course_num >=200) & (df.Days.isin(mwlist))]
lower_tth_df = df[(df.course_num <500) & (df.course_num >=200) & (df.Days.isin(tthlist))]
upper_mw_df = df[(df.course_num >=500) & (df.Days.isin(mwlist))]
upper_tth_df = df[(df.course_num >=500) & (df.Days.isin(tthlist))]
```

```
sched_df = pd.DataFrame(columns = [
    "Intro_MW",
    "Intro_TTH",
    "Lower_MW",
    "Lower_TTH",
    "Upper_MW",
    "Upper_TTH"
    ],
   index = df.start_time.sort_values().unique())

sched_df = sched_df.iloc[1: , :]
sched_df["Intro_MW"] = intro_mw_df.groupby("start_time").course.sum()
sched_df["Intro_TTH"] = intro_tth_df.groupby("start_time").course.sum()
sched_df["Lower_MW"] = lower_mw_df.groupby("start_time").course.sum()
sched_df["Lower_TTH"] = lower_tth_df.groupby("start_time").course.sum()
sched_df["Upper_MW"] = upper_mw_df.groupby("start_time").course.sum()
sched_df["Upper_TTH"] = upper_tth_df.groupby("start_time").course.sum()
sched_df = sched_df.fillna('')

sched_df
```

```
sched_df = pd.DataFrame(columns = [], index = df.start_time.sort_values().unique())

sched_df = sched_df.iloc[1: , :]
sched_df["Intro_MW"] = intro_mw_df.groupby("start_time").course.sum()
sched_df["Intro_TTH"] = intro_tth_df.groupby("start_time").course.sum()
sched_df["Lower_MW"] = lower_mw_df.groupby("start_time").course.sum()
sched_df["Lower_TTH"] = lower_tth_df.groupby("start_time").course.sum()
sched_df["Upper_MW"] = upper_mw_df.groupby("start_time").course.sum()
sched_df["Upper_TTH"] = upper_tth_df.groupby("start_time").course.sum()
sched_df = sched_df.fillna('')

sched_df
```

Problem 1 Pivot Table solution

Pivot Table

Reshaping & Grouping

values: column to group
columns: Values in this column
become columns in pivot table
Index: Values in this column become index

Pivot

df

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t

```
df.pivot(index='foo',
         columns='bar',
         values='baz')
```

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

df

	A	B	C	D	E
0	foo	one	small	1	2
1	foo	one	large	2	4
2	foo	one	large	2	5
3	foo	two	small	3	5
4	foo	two	small	3	6
5	bar	one	large	4	6
6	bar	one	small	5	8
7	bar	two	small	6	9
8	bar	two	large	7	9

```
table = pd.pivot_table(df, values='D',
                       index=['A', 'B'],
                       columns=['C'],
                       aggfunc=np.sum)
```

C		large	small
A	B		
bar	one	4.0	5.0
	two	7.0	6.0
foo	one	4.0	1.0
	two	NaN	6.0

df

	A	B	C	D	E
0	foo	one	small	1	2
1	foo	one	large	2	4
2	foo	one	large	2	5
3	foo	two	small	3	5
4	foo	two	small	3	6
5	bar	one	large	4	6
6	bar	one	small	5	8
7	bar	two	small	6	9
8	bar	two	large	7	9

```
table = pd.pivot_table(df, values='D',
                       index=['A', 'B'],
                       columns=['C'],
                       aggfunc=np.sum
                       fill_value = 0)
```

C		large	small
A	B		
bar	one	4.0	5.0
	two	7.0	6.0
foo	one	4.0	1.0
	two	0	6.0

```
table = pd.pivot_table(df,
                       values=['D', 'E'],
                       index=['A', 'C'],
                       aggfunc={'D': np.mean,'E': np.mean})
```

df

	A	B	C	D	E
0	foo	one	small	1	2
1	foo	one	large	2	4
2	foo	one	large	2	5
3	foo	two	small	3	5
4	foo	two	small	3	6
5	bar	one	large	4	6
6	bar	one	small	5	8
7	bar	two	small	6	9
8	bar	two	large	7	9

		D	E
A	C		
bar	large	5.500000	7.500000
	small	5.500000	8.500000
foo	large	2.000000	4.500000
	small	2.333333	4.333333

df

	A	B	C	D	E
0	foo	one	small	1	2
1	foo	one	large	2	4
2	foo	one	large	2	5
3	foo	two	small	3	5
4	foo	two	small	3	6
5	bar	one	large	4	6
6	bar	one	small	5	8
7	bar	two	small	6	9
8	bar	two	large	7	9

```
table = pd.pivot_table(df,
                       values=['D', 'E'],
                       index=['A', 'C'],
                       aggfunc={'D': np.mean,
                       'E': [min, max, np.mean]})
```

Table 1-1

		D	E		
		mean	max	mean	min
A	C				
bar	large	5.500000	9	7.500000	6
	small	5.500000	9	8.500000	8
foo	large	2.000000	5	4.500000	4
	small	2.333333	6	4.333333	2

```
        main_data = pd.read_table("search.xls", header=1)
        main_data.head()

         main_data["Course"] = main_data["Course #"].astype(str) + data["Suffix"].fillna('')

def class_division(row):
    if (row['Course #'] < 200 ) and (row['Days'] == "TTH" or row['Days'] == "T" or row['Days'] =="TH"):
        val = "Intro TTH"
    elif (row['Course #'] < 200) and (row['Days'] == "M" or row['Days'] == "MW" or row['Days'] == "W"):
        val = "Intro MW"
    elif (row['Course #'] >= 200 and row["Course #"] < 500) and (row['Days'] == "TTH" or row['Days'] ==
"T" or row['Days'] =="TH"):
        val = "Lower TTH"
    elif (row['Course #'] >= 200 and row["Course #"] < 500)and (row['Days'] == "M" or row['Days'] ==
"MW" or row['Days'] == "W"):
        val = "Lower MW"
    elif row['Course #'] >= 500 and (row['Days'] == "TTH" or row['Days'] == "T" or row['Days'] =="TH"):
        val = "Upper TTH"
    elif row['Course #'] >= 500 and (row['Days'] == "M" or row['Days'] == "MW" or row['Days'] == "W"):
        val = "Upper MW"
    else:
        val = "other"
    return val
```

```
def days(row):
    if row['Days'] == "TTH" or row['Days'] == "T" or row['Days'] =="TH"
        return 'TTH'
    else:
        return 'MW'


def class_level(row):
    if row['Course #'] < 200:
        return 'Intro'
    if  row['Course #'] >= 500
        return 'Upper'
   return 'Lower'



def class_division(row):
    return class_level(row) + ' ' + days(row)
```

```
main_data["class_type"] = main_data.apply(class_division, axis = 1)
res_data = pd.pivot_table(main_data, index="Start Time", columns="class_type",
                          values = "Course",
                          aggfunc =lambda x: ','.join((x.astype(str))))
res_data.index = res_data.index.astype(int)
res_data.fillna('')
```

Question 2

```
main_data["Courses"] = main_data["Course #"].astype(str) + ' ' + main_data["Title"]

data_second = main_data[['Courses', 'Total Enrolled']].groupby(by='Courses').sum()
data_second = data_second
data_second.rename(columns={'Total Enrolled':'Students'}, inplace=True)
data_second["Students"] = data_second["Students"].astype(int)
data_second
```

Courses	Students
100 COMPTR SCIENCE PRINCIPLES	38
150 INTRO COMPUTER PROGRAMMNG	175
150 INTRO COMPUTER PROGRM LAB	288
160 INT COMPUTER PROGRAMMING	221
160 INT COMPUTR PROGAMMNG LAB	430
200 INTRO DATA SCIENCE PYTHON	22

```
main_data[main_data['Room']!= 'LINE']['Room'].fillna('')
data_third = main_data[main_data['Room'].notna()]
data_third[['Building','Room','Course #']].groupby(['Building','Room']).count()
```

		Course #
Building	**Room**	
AH	**2116**	2
AL	**101**	1
	204	1
COM	**105**	1
	207	1
ENS	**291**	1
GMCS	**305**	1
	306	1
	308	2
	314	1
	324	1

```
import os
import numpy as np
import pandas as pd
file = os.path.join("/Users/rwhitney/data/","search.xls")
df = pd.read_csv(file,sep='\t', skiprows=[0])

df['Suffix'] = df['Suffix'].fillna('')
df['Course'] = df.apply(lambda row : str(row['Course #']) + row.Suffix,axis=1)

MWFilter = (df.Days == 'M') | (df.Days == 'W') | (df.Days == 'MW')
TTHFilter = (df.Days == 'T') | (df.Days == 'TH') | (df.Days == 'TTH')

IntroFilter = (df['Course #'] < 200)
LowerFilter = (200 <= df['Course #']) & (df['Course #']< 500)
UpperFilter = (500 <= df['Course #'])

group = df.loc[(MWFilter & IntroFilter)].groupby('Start Time')
IntroMW = group['Course'].apply(list)

group = df.loc[(TTHFilter & IntroFilter)].groupby('Start Time')
IntroTTH = group['Course'].apply(list)
```

```
finalDf = pd.DataFrame({
    'Intro MW' : IntroMW,
    'Intro TTH': IntroTTH,
    'Lower MW': LowerMW,
    'Lower TTH': LowerTTH,
    'Upper MW': UpperMW,
    'Upper TTH': UpperTTH
})


finalDf= finalDf.fillna('')
finalDf = finalDf.applymap(lambda col : ','.join(col) if type(col) is list else '')
finalDf
```

Problem 3

```
ans3 = pd.DataFrame(columns=["ClassRoom","No. of Classes"])
ans3['ClassRoom'] = filtered_df['Building'].dropna().map(str) + ' ' + filtered_df['Room'].dropna().map(str)
ans3["No. of Classes"] = filtered_df['Title']
groupAns3 = ans3.groupby('ClassRoom',as_index =False).count()
groupAns3.style.set_caption("No.of classes are held in each classroom").set_table_styles([{
    'selector': 'caption',
    'props': [
        ('color', 'black'),
        ('font-size', '16px'),
        ('text-align', 'center')
    ]
}])
```