Midterm Exam

Brandan Owens and Loan Pham

Q.1

In [860…

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import string
```

In [861…

```
# (a) Set random seed to be 50.

np.random.seed(50)
np.random.randn(50)
```

Out[861…

```
array([-1.56035211, -0.0309776 , -0.62092842, -1.46458049,  1.41194612,
       -0.47673214, -0.78046921,  1.07026774, -1.2822926 , -1.3274789 ,
        0.12633764,  0.86219372,  0.69673696, -0.33456518, -0.99752606,
        1.59890829,  3.31407535,  0.98777046,  0.12386626,  0.74278539,
       -0.39395585,  0.14811582, -0.41223445, -0.16071506,  0.13953147,
        0.28546937, -0.28126199,  1.71090732, -0.14976664,  0.69030672,
        1.09520951,  1.3384087 , -1.36898167,  0.48642763,  0.75352168,
        0.36346459, -0.31471048,  1.37328117, -0.62441716,  0.375754  ,
       -0.20041632,  0.74303806,  0.85736196, -1.50618929, -1.66635217,
       -0.2189948 , -0.35885843,  0.37852769,  0.68421537, -1.16785607])
```

In [862…

```
# (b) Create a dataframe with four columns of data:-Each column has 26 random integers
#     -The name of the columns = ['i','ii','iii','iv']-Create an index column and the v
n = 26
ix = list(string.ascii_uppercase)
df = pd.DataFrame(dict(
    i =np.random.randint(-5, 28, size=n),
    ii =np.random.randint(-5, 28, size=n),
    iii = np.random.randint(-5, 28, size=n),
    iv = np.random.randint(-5, 28, size=n), index = ix))
df
```

Out[862…

	i	ii	iii	iv	index
0	22	25	-1	6	A
1	23	9	6	21	B
2	26	21	14	27	C
3	10	17	9	18	D
4	0	1	-2	-2	E
5	-1	25	23	23	F
6	-1	26	-1	0	G
7	13	4	22	23	H

	i	ii	iii	iv	index
8	12	-5	2	-4	I
9	21	11	8	-2	J
10	3	21	16	-5	K
11	6	-1	23	7	L
12	22	-4	16	7	M
13	22	8	9	27	N
14	2	9	8	16	O
15	-4	-3	1	9	P
16	3	17	7	14	Q
17	-2	-5	-5	11	R
18	16	9	-5	14	S
19	-2	25	3	27	T
20	1	21	27	-1	U
21	10	-1	9	10	V
22	26	11	-1	13	W
23	17	24	27	26	X
24	-5	9	2	17	Y
25	6	17	20	-3	Z

In [863…

```
# (c) access the values in rows from “G” to “Q” and columns from “ii” to “iv”.

df.iloc[6:17, 2:5]
```

Out[863…

	iii	iv	index
6	-1	0	G
7	22	23	H
8	2	-4	I
9	8	-2	J
10	16	-5	K
11	23	7	L
12	16	7	M
13	9	27	N
14	8	16	O
15	1	9	P
16	7	14	Q

In [864…

```
# (d) Replace the negative values of “iii” with 0.

num = df["iii"]
print(np.where(num < 0, 0, num))
```

```
[ 0  6 14  9  0 23  0 22  2  8 16 23 16  9  8  1  7  0  0  3 27  9  0 27
  2 20]
```

In [865…

```
# (e) If “iv” < 0, replace the values of “i” with 20.
df.loc[df['iv'] <= 0,'i'] = 20
display(df)
```

	i	ii	iii	iv	index
0	22	25	-1	6	A
1	23	9	6	21	B
2	26	21	14	27	C
3	10	17	9	18	D
4	20	1	-2	-2	E
5	-1	25	23	23	F
6	20	26	-1	0	G
7	13	4	22	23	H
8	20	-5	2	-4	I
9	20	11	8	-2	J
10	20	21	16	-5	K
11	6	-1	23	7	L
12	22	-4	16	7	M
13	22	8	9	27	N
14	2	9	8	16	O
15	-4	-3	1	9	P
16	3	17	7	14	Q
17	-2	-5	-5	11	R
18	16	9	-5	14	S
19	-2	25	3	27	T
20	20	21	27	-1	U
21	10	-1	9	10	V
22	26	11	-1	13	W
23	17	24	27	26	X
24	-5	9	2	17	Y

	i	ii	iii	iv	index
25	20	17	20	-3	Z

In [866…

```
# (f)  Create a list of unique values of “iv”

list(df['iv'].unique())
```

Out[866… [6, 21, 27, 18, -2, 23, 0, -4, -5, 7, 16, 9, 14, 11, -1, 10, 13, 26, 17, -3]

In [867…

```
# (g) For all values greater than 0 in column “i”, find the mean.

col_i = df[df["i"]>0].mean()
col_i.loc["i"]
```

Out[867… 17.047619047619047

In [868…

```
# (h) Drop all the rows with “ii” > 25.
df.dropna()
df.drop(df[df['ii'] > 25].index, inplace=True)
display(df)
```

	i	ii	iii	iv	index
0	22	25	-1	6	A
1	23	9	6	21	B
2	26	21	14	27	C
3	10	17	9	18	D
4	20	1	-2	-2	E
5	-1	25	23	23	F
7	13	4	22	23	H
8	20	-5	2	-4	I
9	20	11	8	-2	J
10	20	21	16	-5	K
11	6	-1	23	7	L
12	22	-4	16	7	M
13	22	8	9	27	N
14	2	9	8	16	O
15	-4	-3	1	9	P
16	3	17	7	14	Q
17	-2	-5	-5	11	R
18	16	9	-5	14	S

	i	ii	iii	iv	index
19	-2	25	3	27	T
20	20	21	27	-1	U
21	10	-1	9	10	V
22	26	11	-1	13	W
23	17	24	27	26	X
24	-5	9	2	17	Y
25	20	17	20	-3	Z

Q.2

In [869…

```
# (a) Read in “Education.csv”.
edu = income_data = pd.read_csv("../dataFiles/Education.csv")
edu.rename(columns={"Region/Country/Area": "Location"}, inplace=True)
```

In [870…

```
# (b) The data of the column “Enrollments (Thousands)” are not numbers, convert them in
edu["Enrollments (Thousands)"].replace(',','',regex=True,inplace=True)
display(edu)
```

	Unnamed: 0	Location	Year	Data	Enrollments (Thousands)
0	0	Total, all countries or areas	2005	Students enrolled in primary education (thousa...	678990
1	1	Total, all countries or areas	2005	Gross enrollement ratio - Primary (male)	104.8
2	2	Total, all countries or areas	2005	Gross enrollment ratio - Primary (female)	99.8
3	3	Total, all countries or areas	2005	Students enrolled in secondary education (thou...	509100
4	4	Total, all countries or areas	2005	Gross enrollment ratio - Secondary (male)	65.7
...	...	...	...	...	...
8157	8157	Zimbabwe	2013	Gross enrollment ratio - Tertiary (male)	6.5
8158	8158	Zimbabwe	2013	Gross enrollment ratio - Tertiary (female)	5.5
8159	8159	Zimbabwe	2015	Students enrolled in tertiary education (thous...	136
8160	8160	Zimbabwe	2015	Gross enrollment ratio - Tertiary (male)	8.9
8161	8161	Zimbabwe	2015	Gross enrollment ratio - Tertiary (female)	8.0

8162 rows × 5 columns

In [871…
```
# (c) Find the unique values in the column “Data”, and then create a list.
list(edu["Data"].unique())
```

Out[871…
```
['Students enrolled in primary education (thousands)',
 'Gross enrollement ratio - Primary (male)',
 'Gross enrollment ratio - Primary (female)',
 'Students enrolled in secondary education (thousands)',
 'Gross enrollment ratio - Secondary (male)',
 'Gross enrollment ratio - Secondary (female)',
 'Students enrolled in tertiary education (thousands)',
 'Gross enrollment ratio - Tertiary (male)',
 'Gross enrollment ratio - Tertiary (female)']
```

In [872…
```
# (d) Create a new dataframe, df2, of primary students’ enrollment in India by filterin

df2 = edu.loc[(edu['Location'] == 'India') & (edu['Data'] == 'Students enrolled in prim
display(df2)
```

	Unnamed: 0	Location	Year	Data	Enrollments (Thousands)
3729	3729	India	2003	Students enrolled in primary education (thousa...	125569
3744	3744	India	2010	Students enrolled in primary education (thousa...	138414
3753	3753	India	2014	Students enrolled in primary education (thousa...	137809
3762	3762	India	2015	Students enrolled in primary education (thousa...	138518
3771	3771	India	2016	Students enrolled in primary education (thousa...	145803

In [873…
```
# (e) From df2, create a series with index = “Year”, and values = “Enrollments (Thousan
yearly_enrollments = pd.Series(df2['Enrollments (Thousands)'].values.astype(float), ind
print(yearly_enrollments.mean())
display(yearly_enrollments)
```

```
137222.6
Year
2003    125569.0
2010    138414.0
2014    137809.0
2015    138518.0
2016    145803.0
dtype: float64
```

In [874…
```
# (f)  Between “Year” = 2003 and “Year” = 2016, insert the missing years to the series.
avg = yearly_enrollments.mean()
print(avg)
idx = range(2003,2013)
s3 = pd.Series([avg ,avg ,avg ,avg ,avg ,avg ,avg ,avg ,avg ], index=[2004,2005,2006,20
yearly_enrollments.append(s3).sort_index()
```

```
137222.6
```

Out[874…

```
2003    125569.0
2004    137222.6
2005    137222.6
2006    137222.6
2007    137222.6
2008    137222.6
2009    137222.6
2010    138414.0
2011    137222.6
2012    137222.6
2013    137222.6
2014    137809.0
2015    138518.0
2016    145803.0
dtype: float64
```

Q.3

In [875…

```
#(a)  Use the file "university_towns.txt" to generate a dataframe. The dataframe contai
#-read each line of the file, if the last six characters of the line are [edit], the li
import re
university_towns = []
with open('../dataFiles/university_towns.txt') as file:
    for line in file:
        if '[edit]' in line:
            state = line
        else: university_towns.append((state, line))

df_town = pd.DataFrame(university_towns, columns=['State', 'RegionName'])
df_town['State'] = df_town['State'].apply(lambda x: re.split('\(',x)[0]).str.strip()
df_town['State'] = df_town['State'].apply(lambda x: re.split('\[',x)[0]).str.strip()
df_town['RegionName'] = df_town['RegionName'].apply(lambda x: re.split('\(',x)[0]).str.
df_town['RegionName'] = df_town['RegionName'].apply(lambda x: re.split('\[',x)[0]).str.
df_town
```

Out[875…

	State	RegionName
0	Alabama	Auburn
1	Alabama	Florence
2	Alabama	Jacksonville
3	Alabama	Livingston
4	Alabama	Montevallo
…	…	…
512	Wisconsin	River Falls
513	Wisconsin	Stevens Point
514	Wisconsin	Waukesha
515	Wisconsin	Whitewater
516	Wyoming	Laramie

517 rows × 2 columns

In [876…

```
# (b)  Read in the file "City_Zhvi_AllHomes.csv". The dataframe shows the mean housing
city = pd.read_csv("../dataFiles/City_Zhvi_AllHomes.csv")
city
```

Out[876…

	RegionID	RegionName	State	Metro	CountyName	SizeRank	1996-04	1996-05	1996-06
0	6181	New York	NY	New York	Queens	1	NaN	NaN	NaN
1	12447	Los Angeles	CA	Los Angeles-Long Beach-Anaheim	Los Angeles	2	155000.0	154600.0	154400.0
2	17426	Chicago	IL	Chicago	Cook	3	109700.0	109400.0	109300.0
3	13271	Philadelphia	PA	Philadelphia	Philadelphia	4	50000.0	49900.0	49600.0
4	40326	Phoenix	AZ	Phoenix	Maricopa	5	87200.0	87700.0	88200.0
…	…	…	…	…	…	…	…	…	…
13043	398343	Urbana	NY	Corning	Steuben	13044	66900.0	65800.0	65500.0
13044	398496	New Denmark	WI	Green Bay	Brown	13045	NaN	NaN	NaN
13045	398839	Angels	CA	NaN	Calaveras	13046	115600.0	116400.0	118200.0
13046	399114	Holland	WI	Sheboygan	Sheboygan	13047	129900.0	130200.0	130300.0
13047	737788	Lebanon Borough	NJ	New York	Hunterdon	13048	143500.0	143200.0	141700.0

13048 rows × 268 columns

In [877…

```
# (c)  Group by "State" and find the mean housing values in 2018-01
city.groupby('State')["2018-01"].mean()
```

Out[877…

```
State
AK    246541.666667
AL    138077.348066
AR    131735.772358
AZ    231847.826087
CA    657378.363384
CO    360702.247191
CT    292281.967213
DC    548300.000000
DE    190376.470588
FL    259333.454545
GA    169254.379562
HI    569347.727273
IA    165052.564103
ID    206910.810811
IL    181724.911661
IN    120469.756098
KS    120517.222222
KY    171249.350649
```

```
LA    151873.154362
MA    401147.005988
MD    316600.314465
ME    224160.655738
MI    159130.490018
MN    209692.592593
MO    160944.827586
MS    112942.748092
MT    213105.000000
NC    177653.944020
ND    189627.272727
NE    160684.090909
NH    249761.375661
NJ    360580.829016
NM    204458.536585
NV    353037.037037
NY    272502.441614
OH    144978.227061
OK    106475.776398
OR    284471.508380
PA    177385.641026
RI    328392.682927
SC    167120.786517
SD    192005.882353
TN    144433.050847
TX    201511.250000
UT    279161.224490
VA    249843.750000
VT    230966.216216
WA    357790.068493
WI    191169.892473
WV    127950.000000
WY    218228.571429
Name: 2018-01, dtype: float64
```

In [878…

```
# (d)  Generate a series which shows the percentage change in housing price between 200
city['change_over_time'] = ((city['2018-01']-city['2000-01']) / city['2000-01'] * 100).
ser1 = pd.Series(city['change_over_time'].values, index = city[['State','RegionName']])
ser1.sort_values(ascending = False).dropna()
```

		0	1	2	3	4	5	6	7	8
City	**State**									
New York	**NY**	NaN	459100.0	84200.0	93500.0	114700.0	110100.0	389000.0	92200.0	672500.0	7
Los Angeles	**CA**	NaN	459100.0	84200.0	93500.0	114700.0	110100.0	389000.0	92200.0	672500.0	7
Chicago	**IL**	NaN	459100.0	84200.0	93500.0	114700.0	110100.0	389000.0	92200.0	672500.0	7
Philadelphia	**PA**	NaN	459100.0	84200.0	93500.0	114700.0	110100.0	389000.0	92200.0	672500.0	7
Phoenix	**AZ**	NaN	459100.0	84200.0	93500.0	114700.0	110100.0	389000.0	92200.0	672500.0	7
...	**...**	...	...	...	...	...	...	...	...	...	
Urbana	**NY**	NaN	459100.0	84200.0	93500.0	114700.0	110100.0	389000.0	92200.0	672500.0	7
New Denmark	**WI**	NaN	459100.0	84200.0	93500.0	114700.0	110100.0	389000.0	92200.0	672500.0	7
Angels	**CA**	NaN	459100.0	84200.0	93500.0	114700.0	110100.0	389000.0	92200.0	672500.0	7
Holland	**WI**	NaN	459100.0	84200.0	93500.0	114700.0	110100.0	389000.0	92200.0	672500.0	7

City	State	0	1	2	3	4	5	6	7	8
Lebanon Borough	**NJ**	NaN	459100.0	84200.0	93500.0	114700.0	110100.0	389000.0	92200.0	672500.0	7

13048 rows × 13048 columns

In [879…
```
# (e)  The current dataframe shows the housing values in each month. Convert it to show
```

Q.4

In [880…
```
# (a)  Read in the file “gdplev.xlsx”  (It is an excel file, not csv file. You need to
gdplev = pd.read_excel("../dataFiles/gdplev.xlsx", skiprows=5)

display(gdplev)
```

	Year/Quartile	GDP
0	1947Q1	243.1
1	1947Q2	246.3
2	1947Q3	250.1
3	1947Q4	260.3
4	1948Q1	266.2
...	...	...
279	2016Q4	18905.5
280	2017Q1	19057.7
281	2017Q2	19250.0
282	2017Q3	19500.6
283	2017Q4	19738.9

284 rows × 2 columns

In [881…
```
# (b)  Look at the column “GDP in billions of current dollars”. Find the year and the q
gdp = gdplev['GDP']
rec = None
for i in range(0, len(gdp)-2):
    if (gdp[i] > gdp [i+1]) and (gdp[i+1] > gdp[i+2]):
        rec = i
        break
recession_start = gdplev.iloc[rec,0]
recession_start
```

Out[881…
```
'1948Q4'
```

In [882…

```
# (c)  For the recession in part (c), find the year and the quarter that it ends. (Look
end = None
for i in range(0, len(gdp)-2):
    if (gdp[i] > gdp [i+1]) and (gdp[i+1] > gdp[i+2]):
        rec = i
        break
for i in range(rec,len(gdp)-2):
        if (gdp[i+2] > gdp[i+1]) & (gdp[i+1] > gdp[i]):
            end = i+2
            break
recession_end = gdplev.iloc[end,0]
recession_end
```

Out[882…

```
'1950Q2'
```