```
# TASK 1
'''
1. Consider the two tables shown below called population and countyseats.
population:
 state      county          year    population
1 California    Orange  2000    2846289
2 California    Orange  2010    3010232
3 California    Los Angeles 2000    3694820
4 California    Los Angeles 2010    3792621

countyseats:
 statename      countyname  countyseat
1 California    Orange          Santa Ana
2 California    Los Angeles     Los Angeles
3 California    San Diego       San Diego
4 Oregon        Wasco           The Dalles

You should be able to calculate the output by hand though you may use R to check
your answer.
'''
```

```
'\n1. Consider the two tables shown below called population and
countyseats.\npopulation:\n state\t\tcounty       \tyear \tpopulation\n1
California\tOrange\t2000 \t2846289\n2 California\tOrange\t2010 \t3010232\n3
California\tLos Angeles\t2000 \t3694820\n4 California\tLos Angeles\t2010
\t3792621\n\ncountyseats:\n statename    \tcountyname\tcountyseat\n1 California
\tOrange      \tSanta Ana\n2 California \tLos Angeles \tLos Angeles\n3 California
\tSan Diego   \tSan Diego\n4 Oregon     \tWasco       \tThe Dalles\n\nYou should
be able to calculate the output by hand though you may use R to check your
answer.\n'
```

```
!pip install pandas
import pandas as pd
population = pd.DataFrame(
    {
        "state": ["California", "California", "California", "California"],
        "county": ["Orange", "Orange", "Los Angeles", "Los Angeles"],
        "year": [2000, 2010, 2000, 2010],
        "population": [2846289, 3010232, 3694820, 3792621]
    }
)

countyseats = pd.DataFrame(
    {
        "statename": ["California", "California", "California", "Oregon"],
        "countyname": ["Orange", "Los Angeles", "San Diego", "Wasco"],
        "countyseat": ["Santa Ana", "Los Angeles", "San Diego", "The Dalles"]
```

```
    }
)
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(2.2.2)
Requirement already satisfied: numpy>=1.22.4 in
/usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas)
(1.16.0)
```

```
# Draw the output table from the following operations (you should be able to
calculate the output by hand though you may use R to check your answers).
# a) pd.merge(population, countyseats) <- ERROR
```

```
# b) pd.merge(population, countyseats, left_on="state", right_on="statename")
pd.merge(population, countyseats, left_on="state", right_on="statename")
```

	state	county	year	population	statename	countyname	countyseat
0	California	Orange	2000	2846289	California	Orange	Santa Ana
1	California	Orange	2000	2846289	California	Los Angeles	Los Angeles
2	California	Orange	2000	2846289	California	San Diego	San Diego
3	California	Orange	2010	3010232	California	Orange	Santa Ana
4	California	Orange	2010	3010232	California	Los Angeles	Los Angeles
5	California	Orange	2010	3010232	California	San Diego	San Diego
6	California	Los Angeles	2000	3694820	California	Orange	Santa Ana
7	California	Los Angeles	2000	3694820	California	Los Angeles	Los Angeles
8	California	Los Angeles	2000	3694820	California	San Diego	San Diego
9	California	Los Angeles	2010	3792621	California	Orange	Santa Ana
10	California	Los Angeles	2010	3792621	California	Los Angeles	Los Angeles
11	California	Los Angeles	2010	3792621	California	San Diego	San Diego

```
# c) pd.merge(population, countyseats, left_on=["state", "county"],
right_on=["statename", "countyname"])
pd.merge(population, countyseats, left_on=["state", "county"],
right_on=["statename", "countyname"])
```

	state	county	year	population	statename	countyname	countyseat
0	California	Orange	2000	2846289	California	Orange	Santa Ana
1	California	Orange	2010	3010232	California	Orange	Santa Ana
2	California	Los Angeles	2000	3694820	California	Los Angeles	Los Angeles
3	California	Los Angeles	2010	3792621	California	Los Angeles	Los Angeles

```
# d) pd.merge(population, countyseats, left_on=["state", "county", "year"],
right_on=["statename", "countyname", "countyseat"]) <- ERROR
```

```
# TASK 2
# Consider the billboard dataset that is supplied with the tidyr package which
shows the Billboard top 100 song rankings in the year 2000. Apply the Pandas data
wrangling methods to answer these questions. For each question, give only the
code
import statsmodels.api as sm

billboard_dataset = sm.datasets.get_rdataset("billboard", "tidyr")
billboard = billboard_dataset.data

billboard_melt = billboard.melt(
    id_vars=['artist', 'track', 'date.entered'],
    value_vars=[f'wk{i}' for i in range(1, 77)],
    var_name='week',
    value_name='rank'
).dropna()
billboard_melt
```

	artist	track	date.entered	week	rank
0	2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk1	87.0
1	2Ge+her	The Hardest Part Of ...	2000-09-02	wk1	91.0
2	3 Doors Down	Kryptonite	2000-04-08	wk1	81.0
3	3 Doors Down	Loser	2000-10-21	wk1	76.0
4	504 Boyz	Wobble Wobble	2000-04-15	wk1	57.0
...	...	...	...	...	...
19716	Creed	Higher	1999-09-11	wk63	50.0
19833	Lonestar	Amazed	1999-06-05	wk63	45.0
20033	Creed	Higher	1999-09-11	wk64	50.0
20150	Lonestar	Amazed	1999-06-05	wk64	50.0
20350	Creed	Higher	1999-09-11	wk65	49.0

```
# a) Show for each track, how many weeks it spent on the chart
track_weeks = billboard_melt.groupby('track').size()
track_weeks
```

track	0
(Hot S**t) Country G...	34
3 Little Words	9
911	19
A Country Boy Can Su...	3
A Little Gasoline	6
...	...
You Won't Be Lonely ...	13
You'll Always Be Lov...	19
You're A God	21
Your Everything	16
www.memory	15

```
# b) List tracks in decreasing order of number of weeks spent on the chart
track_weeks.sort_values(ascending=False)
```

track	0
Higher	57
Amazed	55
Kryptonite	53
Breathe	53
With Arms Wide Open	47
...	...
Deck The Halls	2
Souljas	1
Toca's Miracle	1
No Me Dejes De Quere...	1
Cherchez LaGhost	1

```
# c) Show for each track, its top rank
track_top_rank = billboard_melt.groupby('track')['rank'].max()
track_top_rank
```

track	rank
(Hot S**t) Country G...	100.0
3 Little Words	99.0
911	96.0
A Country Boy Can Su...	93.0
A Little Gasoline	99.0

track	rank
...	...
You Won't Be Lonely ...	97.0
You'll Always Be Lov...	100.0
You're A God	64.0
Your Everything	100.0
www.memory	99.0

```
# d) List tracks in increasing order of its top rank
track_top_rank.sort_values()
```

track	rank
Hot Boyz	38.0
Bye Bye Bye	42.0
Music	44.0
There U Go	46.0
Maria, Maria	48.0
...	...
Stay Or Let It Go	100.0
U Don't Love Me	100.0
U Understand	100.0
I Need A Hot Girl	100.0
(Hot S**t) Country G...	100.0

```
# e) Show for each artist, their top rank
artist_top_rank = billboard_melt.groupby('artist')['rank'].max()
artist_top_rank
```

artist	rank
2 Pac	99.0
2Ge+her	92.0
3 Doors Down	81.0
504 Boyz	96.0
98^0	94.0
...	...
Yankee Grey	95.0
Yearwood, Trisha	91.0
Ying Yang Twins	99.0
Zombie Nation	99.0

artist	rank
matchbox twenty	60.0

```
# f) List artists in increasing order of their top rank
artist_top_rank.sort_values()
```

artist	rank
Elliott, Missy "Misdemeanor"	38.0
Santana	48.0
Janet	59.0
matchbox twenty	60.0
Kandi	66.0
...	...
Limp Bizkit	100.0
Lil' Mo	100.0
Lil Bow Wow	100.0
Sister Hazel	100.0
McEntire, Reba	100.0

```
# g) List tracks that spent more than 35 weeks in the charts
track_weeks[track_weeks > 35]
```

track	0
Amazed	55
Bent	39
Breathe	53
Everything You Want	41
He Wasn't Man Enough	37
Higher	57
I Wanna Know	44
Kryptonite	53
With Arms Wide Open	47

```
# h) List tracks that spent more than 35 weeks in the charts along with their
artists
billboard_melt.groupby(['track', 'artist']).size()[lambda x: x > 35]
```

track	artist	0
Amazed	Lonestar	55
Bent	matchbox twenty	39
Breathe	Hill, Faith	53
Everything You Want	Vertical Horizon	41
He Wasn't Man Enough	Braxton, Toni	37
Higher	Creed	57
I Wanna Know	Joe	44
Kryptonite	3 Doors Down	53
With Arms Wide Open	Creed	47

```
# TASK 3
# The demographics.csv  file (available in the Data Wrangling module on Canvas)
gives the proportion of a country's population in different age groups and some
other demographic data such as mortality rates and expected lifetime. You can
read a CSV file into a DataFrame using Pandas read_csv(), like so: demo =
pd.read_csv("demographics.csv")
demo = pd.read_csv("demographics.csv")
demo
```

	Country Name	Country Code	Series Name	Series Code
0	Afghanistan	AFG	Life expectancy at birth, total (years)	SP.DYN.LE00.IN
1	Afghanistan	AFG	Urban population	SP.URB.TOTL
2	Afghanistan	AFG	Population, total	SP.POP.TOTL
3	Afghanistan	AFG	Population ages 80 and above, female	SP.POP.80UP.FE
4	Afghanistan	AFG	Population ages 80 and above, male	SP.POP.80UP.MA
...	...	...	...	...
3880	Zimbabwe	ZWE	Mortality rate, adult, male (per 1,000 male ad...	SP.DYN.AMRT.MA
3881	Zimbabwe	ZWE	Population, female	SP.POP.TOTL.FE.IN
3882	Zimbabwe	ZWE	Population, male	SP.POP.TOTL.MA.I
3883	Zimbabwe	ZWE	Population ages 65 and above, female	SP.POP.65UP.FE.IN
3884	Zimbabwe	ZWE	Population ages 65 and above, male	SP.POP.65UP.MA.IN

```
# a) The data is not "tidy". In 2-3 sentences, explain why.
# It uses the "wide" format, where multiple columns (such as those for different
demographic indicators) are spread across multiple rows for the same country.
Instead of having one observation per row, multiple variables (like age groups or
sex-related differences) are represented across different rows. Also, there are
repeated rows for the same country based on different demographic series (like
life expectancy or population), which makes it difficult to analyze.
```

```
# b) Transform the table to tidy data with one country per row.
demo_tidy = demo.pivot(index=['Country Name', 'Country Code'], columns='Series
Name', values='YR2015').reset_index()
demo_tidy
```

Series Name	Country Name	Country Code	Life expectancy at birth, total (years)	Mortality rate, ad
0	Afghanistan	AFG	63.377	206.746
1	Albania	ALB	78.025	51.195
2	Algeria	DZA	76.09	84.44
3	American Samoa	ASM	NA	NA
4	Andorra	AND	NA	NA
...	...	...	...	...
254	West Bank and Gaza	PSE	73.442	96.217
255	World	WLD	71.9478983340412	116.083524344293
256	Yemen, Rep.	YEM	66.085	198.071
257	Zambia	ZMB	61.737	255.312
258	Zimbabwe	ZWE	59.534	339.493

```
# c) Add the male/female population numbers together
demo_tidy['Life expectancy at birth, total (years)'] =
pd.to_numeric(demo_tidy['Life expectancy at birth, total (years)'],
errors='coerce')
demo_tidy['Urban population'] = pd.to_numeric(demo_tidy['Urban population'],
errors='coerce')
demo_tidy['Population, total'] = pd.to_numeric(demo_tidy['Population, total'],
errors='coerce')
demo_tidy['Population ages 80 and above, female'] =
pd.to_numeric(demo_tidy['Population ages 80 and above, female'], errors='coerce')
demo_tidy['Population ages 80 and above, male'] =
pd.to_numeric(demo_tidy['Population ages 80 and above, male'], errors='coerce')
demo_tidy['Population ages 15-64, female'] = pd.to_numeric(demo_tidy['Population
ages 15-64, female'], errors='coerce')
demo_tidy['Population ages 15-64, male'] = pd.to_numeric(demo_tidy['Population
ages 15-64, male'], errors='coerce')
demo_tidy['Population ages 0-14, female'] = pd.to_numeric(demo_tidy['Population
ages 0-14, female'], errors='coerce')
demo_tidy['Population ages 0-14, male'] = pd.to_numeric(demo_tidy['Population
ages 0-14, male'], errors='coerce')
demo_tidy['Mortality rate, adult, female (per 1,000 female adults)'] =
pd.to_numeric(demo_tidy['Mortality rate, adult, female (per 1,000 female
adults)'], errors='coerce')
demo_tidy['Mortality rate, adult, male (per 1,000 male adults)'] =
pd.to_numeric(demo_tidy['Mortality rate, adult, male (per 1,000 male adults)'],
errors='coerce')
demo_tidy['Population ages 65 and above, female'] =
pd.to_numeric(demo_tidy['Population ages 65 and above, female'], errors='coerce')
```

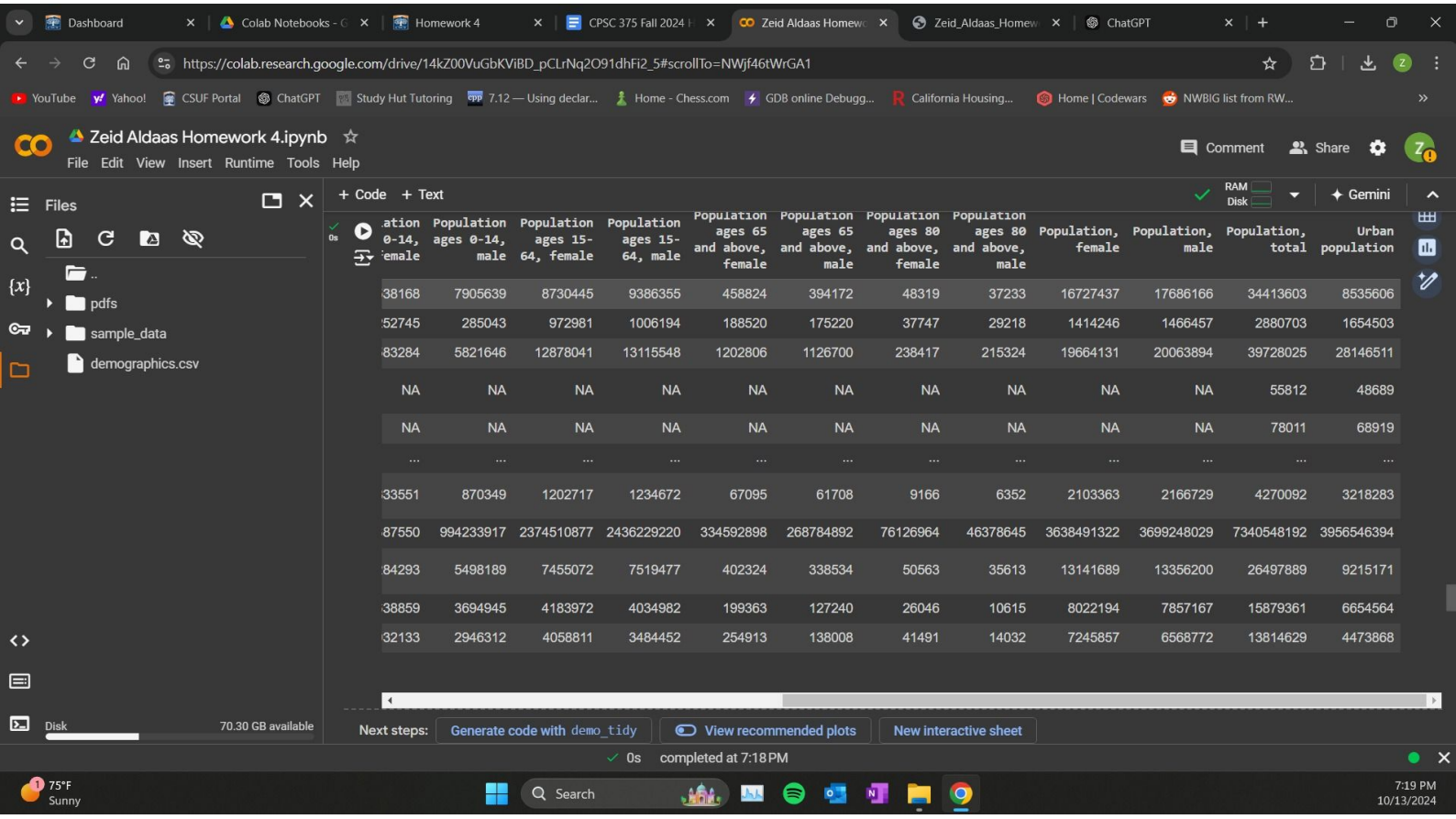
```
demo_tidy['Population ages 65 and above, male'] =
pd.to_numeric(demo_tidy['Population ages 65 and above, male'], errors='coerce')

demo_tidy['SP.DYN.LE00.IN'] = demo_tidy['Life expectancy at birth, total
(years)']
demo_tidy['SP.URB.TOTL'] = demo_tidy['Urban population']
demo_tidy['SP.POP.TOTL'] = demo_tidy['Population, total']
demo_tidy['SP.POP.80UP.IN'] = demo_tidy['Population ages 80 and above, female']
+ demo_tidy['Population ages 80 and above, male']
demo_tidy['SP.POP.1564.IN'] = demo_tidy['Population ages 15-64, female'] +
demo_tidy['Population ages 15-64, male']
demo_tidy['SP.POP.0014.IN'] = demo_tidy['Population ages 0-14, female'] +
demo_tidy['Population ages 0-14, male']
demo_tidy['SP.DYN.AMRT'] = demo_tidy['Mortality rate, adult, female (per 1,000
female adults)'] + demo_tidy['Mortality rate, adult, male (per 1,000 male
adults)']
demo_tidy['SP.POP.TOTL.IN'] = demo_tidy['Population, total']
demo_tidy['SP.POP.65UP.IN'] = demo_tidy['Population ages 65 and above, female']
+ demo_tidy['Population ages 65 and above, male']

demo_final = demo_tidy.drop(columns=[
    ''
    'Life expectancy at birth, total (years)', 'Urban population',
    'Population, total',
    'Mortality rate, adult, female (per 1,000 female adults)', 'Mortality rate,
    adult, male (per 1,000 male adults)',
    'Population ages 0-14, female', 'Population ages 0-14, male',
    'Population ages 15-64, female', 'Population ages 15-64, male',
    'Population ages 65 and above, female', 'Population ages 65 and above, male',
    'Population ages 80 and above, female', 'Population ages 80 and above, male',
    'Population, female', 'Population, male',
    'Country Name',
])

demo_final
```

Series Name	Country Code	SP.DYN.LE00.IN	SP.URB.TOTL	SP.POP.TOTL	SP.POP.80UP.IN	SP.PO
0	AFG	63.377000	8.535606e+06	3.441360e+07	85552.0	1.8116
1	ALB	78.025000	1.654503e+06	2.880703e+06	66965.0	1.9791
2	DZA	76.090000	2.814651e+07	3.972802e+07	453741.0	2.5993
3	ASM	NaN	4.868900e+04	5.581200e+04	NaN	NaN
4	AND	NaN	6.891900e+04	7.801100e+04	NaN	NaN
...	...	...	...	...	...	...
254	PSE	73.442000	3.218283e+06	4.270092e+06	15518.0	2.4373
255	WLD	71.947898	3.956546e+09	7.340548e+09	122505609.0	4.8107
256	YEM	66.085000	9.215171e+06	2.649789e+07	86176.0	1.4974

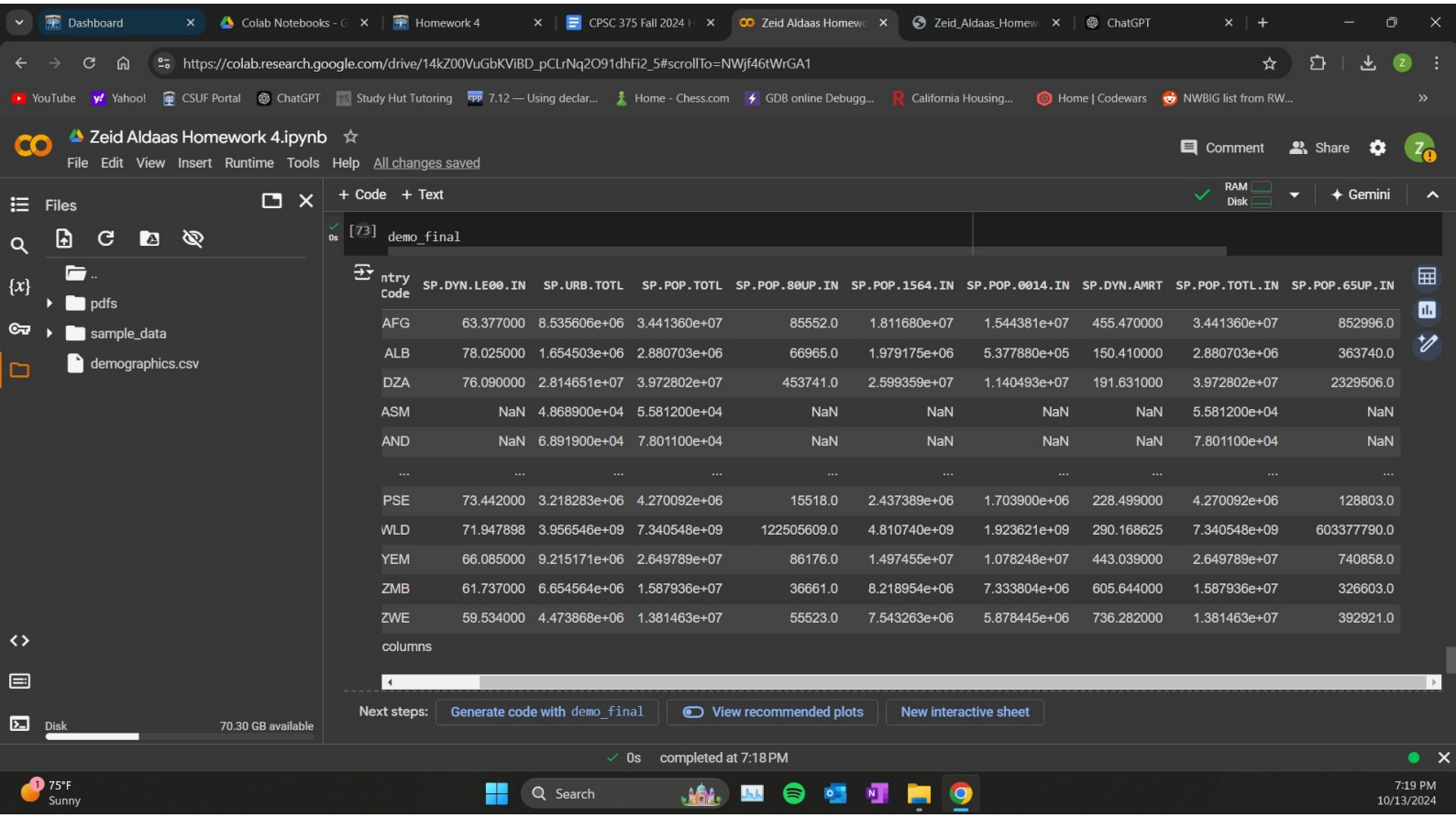
Series Name	Country Code	SP.DYN.LE00.IN	SP.URB.TOTL	SP.POP.TOTL	SP.POP.80UP.IN	SP.PO
257	ZMB	61.737000	6.654564e+06	1.587936e+07	36661.0	8.2189
258	ZWE	59.534000	4.473868e+06	1.381463e+07	55523.0	7.5432

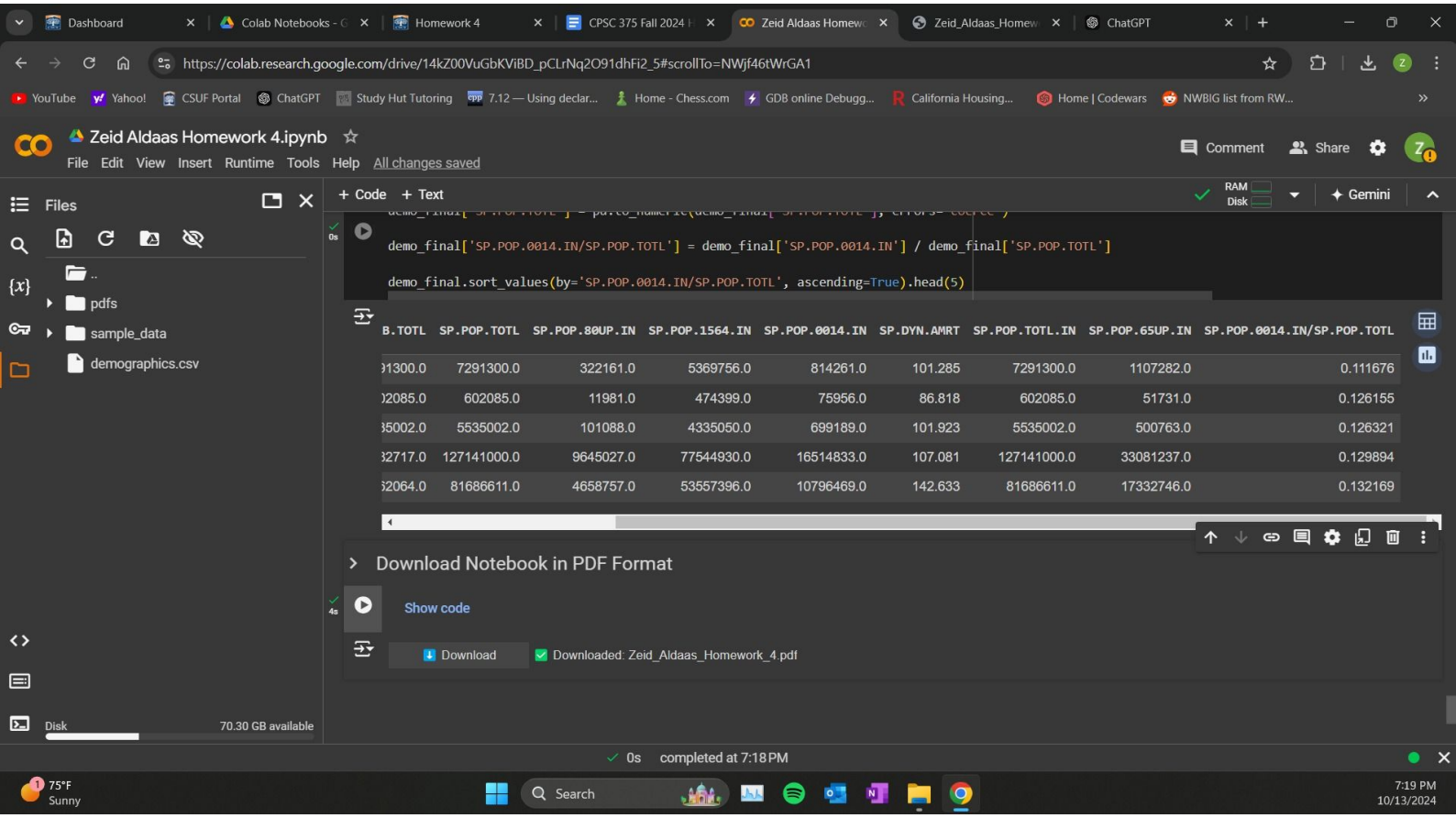
```
# d) Write code to show the top 5 countries with the lowest proportion of the
population below 14 years old (i.e., SP.POP.0014.IN/SP.POP.TOTL) [Code, and list
of 5 countries]
demo_final['SP.POP.0014.IN'] = pd.to_numeric(demo_final['SP.POP.0014.IN'],
errors='coerce')
demo_final['SP.POP.TOTL'] = pd.to_numeric(demo_final['SP.POP.TOTL'],
errors='coerce')

demo_final['SP.POP.0014.IN/SP.POP.TOTL'] = demo_final['SP.POP.0014.IN'] /
demo_final['SP.POP.TOTL']

demo_final.sort_values(by='SP.POP.0014.IN/SP.POP.TOTL', ascending=True).head(5)
```

Series Name	Country Code	SP.DYN.LE00.IN	SP.URB.TOTL	SP.POP.TOTL	SP.POP.80UP.IN	SP.PO
101	HKG	84.278049	7291300.0	7291300.0	322161.0	536975
140	MAC	83.707000	602085.0	602085.0	11981.0	474399
205	SGP	82.743902	5535002.0	5535002.0	101088.0	433505
113	JPN	83.793902	116182717.0	127141000.0	9645027.0	775449
86	DEU	80.641463	63062064.0	81686611.0	4658757.0	535573



ation 0-14, emale	Population ages 0-14, male	Population ages 15-64, female	Population ages 15-64, male	Population ages 65 and above, female	Population ages 65 and above, male	Population ages 80 and above, female	Population ages 80 and above, male	Population, female	Population, male	Population, total	Urban population
38168	7905639	8730445	9386355	458824	394172	48319	37233	16727437	17686166	34413603	8535606
52745	285043	972981	1006194	188520	175220	37747	29218	1414246	1466457	2880703	1654503
83284	5821646	12878041	13115548	1202806	1126700	238417	215324	19664131	20063894	39728025	28146511
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	55812	48689
NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	78011	68919
...	...	...	...	...	...	...	...	...	...	...	...
33551	870349	1202717	1234672	67095	61708	9166	6352	2103363	2166729	4270092	3218283
87550	994233917	2374510877	2436229220	334592898	268784892	76126964	46378645	3638491322	3699248029	7340548192	3956546394
84293	5498189	7455072	7519477	402324	338534	50563	35613	13141689	13356200	26497889	9215171
38859	3694945	4183972	4034982	199363	127240	26046	10615	8022194	7857167	15879361	6654564
32133	2946312	4058811	3484452	254913	138008	41491	14032	7245857	6568772	13814629	4473868


Dashboard
Colab Notebooks - G
Homework 4
CPSC 375 Fall 2024 H
Zeid Aldaas Homew
Zeid_Aldaas_Homew
ChatGPT
https://colab.research.google.com/drive/14kZ00VuGbKViBD_pCLrNq2O91dhFi2_5#scrollTo=NWjf46tWrGA1
YouTube
Yahoo!
CSUF Portal
ChatGPT
Study Hut Tutoring
7.12 — Using declar...
Home - Chess.com
GDB online Debugg...
California Housing...
Home | Codewars
NWBIG list from RW...
Zeid Aldaas Homework 4.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
RAM
Disk
Gemini
[73] demo_final
ntry Code | SP.DYN.LE00.IN | SP.URB.TOTL | SP.POP.TOTL | SP.POP.80UP.IN | SP.POP.1564.IN | SP.POP.0014.IN | SP.DYN.AMRT | SP.POP.TOTL.IN | SP.POP.65UP.IN
AFG | 63.377000 | 8.535606e+06 | 3.441360e+07 | 85552.0 | 1.811680e+07 | 1.544381e+07 | 455.470000 | 3.441360e+07 | 852996.0
ALB | 78.025000 | 1.654503e+06 | 2.880703e+06 | 66965.0 | 1.979175e+06 | 5.377880e+05 | 150.410000 | 2.880703e+06 | 363740.0
DZA | 76.090000 | 2.814651e+07 | 3.972802e+07 | 453741.0 | 2.599359e+07 | 1.140493e+07 | 191.631000 | 3.972802e+07 | 2329506.0
ASM | NaN | 4.868900e+04 | 5.581200e+04 | NaN | NaN | NaN | NaN | 5.581200e+04 | NaN
AND | NaN | 6.891900e+04 | 7.801100e+04 | NaN | NaN | NaN | NaN | 7.801100e+04 | NaN
... | ... | ... | ... | ... | ... | ... | ... | ... | ...
PSE | 73.442000 | 3.218283e+06 | 4.270092e+06 | 15518.0 | 2.437389e+06 | 1.703900e+06 | 228.499000 | 4.270092e+06 | 128803.0
WLD | 71.947898 | 3.956546e+09 | 7.340548e+09 | 122505609.0 | 4.810740e+09 | 1.923621e+09 | 290.168625 | 7.340548e+09 | 603377790.0
YEM | 66.085000 | 9.215171e+06 | 2.649789e+07 | 86176.0 | 1.497455e+07 | 1.078248e+07 | 443.039000 | 2.649789e+07 | 740858.0
ZMB | 61.737000 | 6.654564e+06 | 1.587936e+07 | 36661.0 | 8.218954e+06 | 7.333804e+06 | 605.644000 | 1.587936e+07 | 326603.0
ZWE | 59.534000 | 4.473868e+06 | 1.381463e+07 | 55523.0 | 7.543263e+06 | 5.878445e+06 | 736.282000 | 1.381463e+07 | 392921.0
columns
Next steps: Generate code with demo_final
View recommended plots
New interactive sheet
Disk 70.30 GB available
0s completed at 7:18 PM
75°F Sunny
Search
7:19 PM 10/13/2024


Dashboard
Colab Notebooks - G
Homework 4
CPSC 375 Fall 2024 H
Zeid Aldaas Homew
Zeid_Aldaas_Homew
ChatGPT
https://colab.research.google.com/drive/14kZ00VuGbKViBD_pCLrNq2O91dhFi2_5#scrollTo=NWjf46tWrGA1
YouTube
Yahoo!
CSUF Portal
ChatGPT
Study Hut Tutoring
7.12 — Using declar...
Home - Chess.com
GDB online Debugg...
California Housing...
Home | Codewars
NWBIG list from RW...
Zeid Aldaas Homework 4.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code
+ Text
RAM
Disk
Gemini
demo_final['SP.POP.0014.IN/SP.POP.TOTL'] = demo_final['SP.POP.0014.IN'] / demo_final['SP.POP.TOTL']
demo_final.sort_values(by='SP.POP.0014.IN/SP.POP.TOTL', ascending=True).head(5)
B.TOTL SP.POP.TOTL SP.POP.80UP.IN SP.POP.1564.IN SP.POP.0014.IN SP.DYN.AMRT SP.POP.TOTL.IN SP.POP.65UP.IN SP.POP.0014.IN/SP.POP.TOTL
91300.0 7291300.0 322161.0 5369756.0 814261.0 101.285 7291300.0 1107282.0 0.111676
02085.0 602085.0 11981.0 474399.0 75956.0 86.818 602085.0 51731.0 0.126155
35002.0 5535002.0 101088.0 4335050.0 699189.0 101.923 5535002.0 500763.0 0.126321
32717.0 127141000.0 9645027.0 77544930.0 16514833.0 107.081 127141000.0 33081237.0 0.129894
52064.0 81686611.0 4658757.0 53557396.0 10796469.0 142.633 81686611.0 17332746.0 0.132169
Download Notebook in PDF Format
Show code
Download
Downloaded: Zeid_Aldaas_Homework_4.pdf
pdfs
sample_data
demographics.csv
Disk 70.30 GB available
0s completed at 7:18 PM
75°F Sunny
Search
7:19 PM 10/13/2024