BEA_gdp_cpi

January 31, 2023

This notebook is for connecting to the Bureau of Economic Analysis(BEA) api and requesting gdp and consumer price index(cpi) for all counties in the US. Once data is requested, the data is cleaned and some visualizations are created.

[1]:
```
#load libraries for this project
import pandas as pd
import numpy as np
import bea_api_key
import requests, json
import re
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.ticker import ScalarFormatter
```

[2]:
```
#personal api key
key = bea_api_key.bea_key
```

[3]:
```
datasets = requests.get('https://apps.bea.gov/api/data?
 ↪&UserID={}&method=GETDATASETLIST&'.format(key))
```

[4]:
```
#lists all the datasets available through the BEA api -- I believe the dataset␣
 ↪we want is the "regional" set
# datasets.json()
```

Get ParameterList – retrieves a list of the parameters(required and optional) for a particular dataset. Required Parameters:UserID, Method, DatasetName

[5]:
```
regional_params = requests.get('https://apps.bea.gov/api/data?
 ↪&UserID={}&method=getparameterlist&datasetname=Regional'.format(key))
```

[6]:
```
# regional_params.json()
```

parameters within the regional dataset:

GeoFips (str) comma delimited list of 5 character geographic codes – COUNTY for all counties STATE for all states MSA for all MSAs MIC for all micropolitan areas PORT for all state metro/nonmetro portions DIV for all metropolitan devisions CSA for all combined statistical areas State abbreviations ex(NY) for all counties in one state LineCode (int) line code for a stat or

industry TableName (str) reginal income or product table to retrieve Year (str) comma delimited list of years – ALL for all years

GetParameterValues – retrieves a list of the valid values for a particular parameter. Required Parameters:UserID, Method, DatasetName, ParameterName

[7]:
```
#get values used in the regional parameter
regional_paramvals_tablename = requests.get('https://apps.bea.gov/api/data?
↪&UserID={}&method=GetParameterValues&datasetname=Regional&ParameterName=TableName'.
↪format(key))
```

[8]:
```
#possible data to gather -- show to team and decide what values we want
# regional_paramvals_tablename.json()
```

[9]:
```
 {'Key': 'CAGDP9',
     'Desc': 'Real GDP by county and metropolitan area (NAICS)'},
```

[9]:
```
({'Key': 'CAGDP9',
  'Desc': 'Real GDP by county and metropolitan area (NAICS)'},)
```

[10]:
```
#geofips are locations we can hone in on
regional_paramvals_geofips = requests.get('https://apps.bea.gov/api/data?
↪&UserID={}&method=GetParameterValues&datasetname=Regional&ParameterName=GeoFips'.
↪format(key))
```

[11]:
```
#we want data for every county in the US so we will use key '00000'
#if we want to get specific by state/county we can request specifics using the␣
↪given keys
# regional_paramvals_geofips.json()
```

[12]:
```
regional_paramvals_linecode = requests.get('https://apps.bea.gov/api/data?
↪&UserID={}&method=GetParameterValues&datasetname=Regional&ParameterName=LineCode'.
↪format(key))
```

[13]:
```
# regional_paramvals_linecode.json()
```

Each Dataset contains different dimensions. There are a few pre-defined dimensions that are common to most Datasets, including: • CL_UNIT – a descriptor of the units reported for the data value (e. g. USD for U. S. dollars, and PC for percent) • UNIT_MULT – a descriptor of the multiplierthat appliesto the data value. This value is the base10 exponentthatshould be applied to the data value (e. g. amounts reported in millions would have a UNIT_MULT of 6; amounts reported in billions would have a UNIT_MULT of 9).

The specific meaning of each dimension is described in the Appendix for each dataset.

NOTES ON REGIONAL DATASET

METHOD = getdata PARAMETERS: tablename (required) – we will choose from the list of tables offered / loop through multiple tables making requests

linecode (required) – LineCode corresponds to the statistic in a table. It can either be one value(ie.1,10,11), or 'ALL' to retrieve all the statistics for one GeoFips.

GeoFips parameter – (required, multiple value) GeoFips specifies geography. It can be all states (STATE), all counties (COUNTY), all Metropolitan Statistical Areas (MSA), all Micropolitan Statistical Areas (MIC), all Metropolitan Divisions (DIV), all Combined Statistical Areas (CSA), all metropolitan/nonmetropolitan portions (PORT), or state post office abbreviation for all counties in one state (e.g. NY). It can also be a list of ANSI state-county codes or metropolitan area codes. For example, the counties in Connecticut and Delaware– 09001,09003,09005,09007,09009,09011,09013,09015,10001,10003,10005. **Only one GeoFips is allowed when LineCode parameter value is 'ALL'.

Year parameter – (optional, multiple value) Year is either a list of comma delimited years, LAST5, LAST10, or ALL. Year will default to LAST5 years if the parameter is not specified.

[14]:
```
#test request
regional_gdp_test = requests.get('https://apps.bea.gov/api/data?
↪&UserID={}&method=GetData&datasetname=Regional&TableName=CAGDP2&GeoFips=COUNTY&LineCode=1&Y
↪format(key))
```

[15]:
```
#this grabbed gdp all industry total for each county and each year
# regional_gdp_test.json()
```

[16]:
```
regional_gdp_test_json = regional_gdp_test.json()
```

[17]:
```
# regional_gdp_test_json
```

[18]:
```
regional_gdp_test_json_data =␣
↪regional_gdp_test_json['BEAAPI']['Results']['Data']
```

[19]:
```
# regional_gdp_test_json_data
```

[20]:
```
#initiate empty lists to move data from dict to df
code_gdp = []
geofips_gdp = []
location_gdp = []
year_gdp = []
unit_gdp = []
unit_mult_gdp = []
value_gdp = []
```

```
for i in regional_gdp_test_json_data:
#     print(i)
    code_gdp.append(i['Code'])
    geofips_gdp.append(i['GeoFips'])
    location_gdp.append(i['GeoName'])
    year_gdp.append(i['TimePeriod'])
    unit_gdp.append(i['CL_UNIT'])
    unit_mult_gdp.append(i['UNIT_MULT'])
    value_gdp.append(i['DataValue'])
```

[21]:
```
len(code_gdp)
```

[21]:
```
65478
```

[22]:
```
regional_gdp_df = pd.DataFrame(code_gdp, columns = ['code'])
regional_gdp_df['geofips'] = geofips_gdp
regional_gdp_df['location'] = location_gdp
regional_gdp_df['year'] = year_gdp
regional_gdp_df['unit'] = unit_gdp
regional_gdp_df['unit_mult'] = unit_mult_gdp
regional_gdp_df['value'] = value_gdp
```

[23]:
```
regional_gdp_df.head()
```

[23]:

	code	geofips	location	year	unit	unit_mult	value
0	CAGDP2-1	01001	Autauga, AL	2008	Thousands of dollars	3	1,096,667
1	CAGDP2-1	01001	Autauga, AL	2010	Thousands of dollars	3	1,265,180
2	CAGDP2-1	01001	Autauga, AL	2018	Thousands of dollars	3	1,808,759
3	CAGDP2-1	01001	Autauga, AL	2020	Thousands of dollars	3	1,781,726
4	CAGDP2-1	01001	Autauga, AL	2014	Thousands of dollars	3	1,571,737

[24]:
```
regional_gdp_df.tail()
```

[24]:

	code	geofips	location	year	unit	unit_mult
65473	CAGDP2-1	56045	Weston, WY	2004	Thousands of dollars	3
65474	CAGDP2-1	56045	Weston, WY	2003	Thousands of dollars	3
65475	CAGDP2-1	56045	Weston, WY	2014	Thousands of dollars	3
65476	CAGDP2-1	56045	Weston, WY	2006	Thousands of dollars	3
65477	CAGDP2-1	56045	Weston, WY	2017	Thousands of dollars	3

```
          value
65473  201,252
65474  185,337
65475  278,892
65476  300,488
65477  285,691
```

[25]:
```python
#find the rows with 2 commas in the location
def count_commas(column):
    count = 0
    for item in column:
        count += item.count(',')
    return count

loc_commas = []

for i, row in enumerate(regional_gdp_df['location']):
    if count_commas(row) == 2:
        loc_commas.append(i)
```

[26]:
```python
def split_string(string):
    split_list = string.split(',')
    if len(split_list) >= 2:
        # split based on the first comma
        first_part = split_list[0]
        second_part = split_list[1]
        print(second_part)
    else:
        # split based on the second comma
        first_part = split_list[0]
        second_part = None
    return first_part, second_part

def split_column_first(series):
    return series.str.split(',').apply(lambda x: split_string(x[0]))

def split_column_second(series):
    return series.str.split(',').apply(lambda x: split_string(x[1]))

def add_split_columns(df, column_name):
    split_result_first = split_column_first(df[column_name])
    split_result_second = split_column_second(df[column_name])
    df['first_part_county'] = split_result_first.apply(lambda x: x[0])
    df['second_part_county'] = split_result_second.apply(lambda x: x[0])
    return df
```

[27]:
```
add_split_columns(regional_gdp_df, 'location')
```

[27]:

	code	geofips	location	year	unit	unit_mult	value	first_part_county	second_part_county
0	CAGDP2-1	01001	Autauga, AL	2008	Thousands of dollars	3	1,096,667	Autauga	AL
1	CAGDP2-1	01001	Autauga, AL	2010	Thousands of dollars	3	1,265,180	Autauga	AL
2	CAGDP2-1	01001	Autauga, AL	2018	Thousands of dollars	3	1,808,759	Autauga	AL
3	CAGDP2-1	01001	Autauga, AL	2020	Thousands of dollars	3	1,781,726	Autauga	AL
4	CAGDP2-1	01001	Autauga, AL	2014	Thousands of dollars	3	1,571,737	Autauga	AL
…	…	…	…	…	…	…	…	…	…
65473	CAGDP2-1	56045	Weston, WY	2004	Thousands of dollars	3	201,252	Weston	WY
65474	CAGDP2-1	56045	Weston, WY	2003	Thousands of dollars	3	185,337	Weston	WY
65475	CAGDP2-1	56045	Weston, WY	2014	Thousands of dollars	3	278,892	Weston	WY
65476	CAGDP2-1	56045	Weston, WY	2006	Thousands of dollars	3	300,488	Weston	WY
65477	CAGDP2-1	56045	Weston, WY	2017	Thousands of dollars	3	285,691	Weston	WY

[65478 rows x 9 columns]

[28]:
```
#split location into county and state columns using the comma delimiter --␣
 ↪issues with virgina having commas in county name
regional_gdp_df['state'] = regional_gdp_df['location'].str.split(',',␣
 ↪expand=True)[2]

regional_gdp_df['state'] = regional_gdp_df['state'].
 ↪fillna(regional_gdp_df['location'].str.split(',', expand=True)[1])

#some states still have a * next to their name
regional_gdp_df['state'].unique()

regional_gdp_df['state'] = regional_gdp_df['state'].replace('\\*','',␣
 ↪regex=True)
```

[29]:
```
# find matching values between second_part_county and state
mask = regional_gdp_df['second_part_county'] == regional_gdp_df['state']
```

```
# change values in col2 where mask is True to NaN
regional_gdp_df.loc[mask, 'second_part_county'] = 'NA'

regional_gdp_df = regional_gdp_df.replace('NA', float('NaN'))

regional_gdp_df['county'] = regional_gdp_df.apply(lambda x:␣
↪str(x['first_part_county']) + str(x['second_part_county']) if pd.
↪notna(x['second_part_county']) and pd.notna(x['first_part_county']) else␣
↪str(x['first_part_county']) if pd.notna(x['first_part_county']) else␣
↪str(x['second_part_county']), axis=1)
```

[30]:
```
regional_gdp_df
```

[30]:
```
           code geofips     location  year                 unit unit_mult  \
0      CAGDP2-1   01001  Autauga, AL  2008  Thousands of dollars         3
1      CAGDP2-1   01001  Autauga, AL  2010  Thousands of dollars         3
2      CAGDP2-1   01001  Autauga, AL  2018  Thousands of dollars         3
3      CAGDP2-1   01001  Autauga, AL  2020  Thousands of dollars         3
4      CAGDP2-1   01001  Autauga, AL  2014  Thousands of dollars         3
…         …     …          …   …                    …       …
65473  CAGDP2-1   56045   Weston, WY  2004  Thousands of dollars         3
65474  CAGDP2-1   56045   Weston, WY  2003  Thousands of dollars         3
65475  CAGDP2-1   56045   Weston, WY  2014  Thousands of dollars         3
65476  CAGDP2-1   56045   Weston, WY  2006  Thousands of dollars         3
65477  CAGDP2-1   56045   Weston, WY  2017  Thousands of dollars         3

           value first_part_county second_part_county state   county
0      1,096,667          Autauga                 NaN    AL  Autauga
1      1,265,180          Autauga                 NaN    AL  Autauga
2      1,808,759          Autauga                 NaN    AL  Autauga
3      1,781,726          Autauga                 NaN    AL  Autauga
4      1,571,737          Autauga                 NaN    AL  Autauga
…          …                …                   …   …       …
65473    201,252           Weston                 NaN    WY   Weston
65474    185,337           Weston                 NaN    WY   Weston
65475    278,892           Weston                 NaN    WY   Weston
65476    300,488           Weston                 NaN    WY   Weston
65477    285,691           Weston                 NaN    WY   Weston

[65478 rows x 11 columns]
```

[31]:
```
test = regional_gdp_df.iloc[loc_commas]
```

[32]:
```
test
```

[32]:
```
           code geofips                                  location  year  \
61068  CAGDP2-1   51907       Augusta, Staunton + Waynesboro, VA*  2008
```

```
61069  CAGDP2-1   51907       Augusta, Staunton + Waynesboro, VA*  2002
61070  CAGDP2-1   51907       Augusta, Staunton + Waynesboro, VA*  2016
61071  CAGDP2-1   51907       Augusta, Staunton + Waynesboro, VA*  2011
61072  CAGDP2-1   51907       Augusta, Staunton + Waynesboro, VA*  2013
…         …     …                                          …   …
61378  CAGDP2-1   51945  Rockbridge, Buena Vista + Lexington, VA*  2003
61379  CAGDP2-1   51945  Rockbridge, Buena Vista + Lexington, VA*  2014
61380  CAGDP2-1   51945  Rockbridge, Buena Vista + Lexington, VA*  2021
61381  CAGDP2-1   51945  Rockbridge, Buena Vista + Lexington, VA*  2006
61382  CAGDP2-1   51945  Rockbridge, Buena Vista + Lexington, VA*  2017

                       unit unit_mult      value first_part_county  \
61068  Thousands of dollars         3  4,353,973           Augusta
61069  Thousands of dollars         3  3,429,356           Augusta
61070  Thousands of dollars         3  4,997,833           Augusta
61071  Thousands of dollars         3  4,589,040           Augusta
61072  Thousands of dollars         3  4,774,561           Augusta
…                        …       …         …                 …
61378  Thousands of dollars         3    901,029        Rockbridge
61379  Thousands of dollars         3  1,161,424        Rockbridge
61380  Thousands of dollars         3  1,473,567        Rockbridge
61381  Thousands of dollars         3  1,027,229        Rockbridge
61382  Thousands of dollars         3  1,280,170        Rockbridge

            second_part_county state                              county
61068     Staunton + Waynesboro    VA       Augusta Staunton + Waynesboro
61069     Staunton + Waynesboro    VA       Augusta Staunton + Waynesboro
61070     Staunton + Waynesboro    VA       Augusta Staunton + Waynesboro
61071     Staunton + Waynesboro    VA       Augusta Staunton + Waynesboro
61072     Staunton + Waynesboro    VA       Augusta Staunton + Waynesboro
…                           …   …                                    …
61378   Buena Vista + Lexington    VA  Rockbridge Buena Vista + Lexington
61379   Buena Vista + Lexington    VA  Rockbridge Buena Vista + Lexington
61380   Buena Vista + Lexington    VA  Rockbridge Buena Vista + Lexington
61381   Buena Vista + Lexington    VA  Rockbridge Buena Vista + Lexington
61382   Buena Vista + Lexington    VA  Rockbridge Buena Vista + Lexington

[105 rows x 11 columns]
```

[33]:
```
regional_gdp_df = regional_gdp_df.
↪drop(['first_part_county','second_part_county'], axis = 1)
```

[34]:
```
regional_gdp_df.head(10)
```

[34]:
```
       code geofips     location  year                 unit unit_mult  \
0  CAGDP2-1   01001  Autauga, AL  2008  Thousands of dollars         3
1  CAGDP2-1   01001  Autauga, AL  2010  Thousands of dollars         3
```

```
2  CAGDP2-1   01001  Autauga, AL  2018  Thousands of dollars         3
3  CAGDP2-1   01001  Autauga, AL  2020  Thousands of dollars         3
4  CAGDP2-1   01001  Autauga, AL  2014  Thousands of dollars         3
5  CAGDP2-1   01001  Autauga, AL  2019  Thousands of dollars         3
6  CAGDP2-1   01001  Autauga, AL  2007  Thousands of dollars         3
7  CAGDP2-1   01001  Autauga, AL  2003  Thousands of dollars         3
8  CAGDP2-1   01001  Autauga, AL  2017  Thousands of dollars         3
9  CAGDP2-1   01001  Autauga, AL  2016  Thousands of dollars         3

       value state   county
0  1,096,667    AL  Autauga
1  1,265,180    AL  Autauga
2  1,808,759    AL  Autauga
3  1,781,726    AL  Autauga
4  1,571,737    AL  Autauga
5  1,784,796    AL  Autauga
6  1,192,292    AL  Autauga
7    824,096    AL  Autauga
8  1,754,321    AL  Autauga
9  1,793,087    AL  Autauga
```

[35]:
```
#119 rows with (NA) value fill to 0 and then use avg
regional_gdp_df[regional_gdp_df['value'] == '(NA)']
```

[35]:
```
          code geofips                                location  year  \
1512  CAGDP2-1   02063                  Chugach Census Area, AK*  2008
1513  CAGDP2-1   02063                  Chugach Census Area, AK*  2018
1514  CAGDP2-1   02063                  Chugach Census Area, AK*  2010
1515  CAGDP2-1   02063                  Chugach Census Area, AK*  2017
1516  CAGDP2-1   02063                  Chugach Census Area, AK*  2014
…        …      …                                        …    …
2070  CAGDP2-1   02280  Wrangell-Petersburg Census Area, AK*  2011
2074  CAGDP2-1   02280  Wrangell-Petersburg Census Area, AK*  2015
2076  CAGDP2-1   02280  Wrangell-Petersburg Census Area, AK*  2012
2077  CAGDP2-1   02280  Wrangell-Petersburg Census Area, AK*  2016
5391  CAGDP2-1   08014                      Broomfield, CO*  2001

                      unit unit_mult value state  \
1512  Thousands of dollars         3  (NA)    AK
1513  Thousands of dollars         3  (NA)    AK
1514  Thousands of dollars         3  (NA)    AK
1515  Thousands of dollars         3  (NA)    AK
1516  Thousands of dollars         3  (NA)    AK
…                      …       …    …    …
2070  Thousands of dollars         3  (NA)    AK
2074  Thousands of dollars         3  (NA)    AK
2076  Thousands of dollars         3  (NA)    AK
```

```
2077  Thousands of dollars         3  (NA)    AK
5391  Thousands of dollars         3  (NA)    CO

                                    county
1512              Chugach Census Area AK*
1513              Chugach Census Area AK*
1514              Chugach Census Area AK*
1515              Chugach Census Area AK*
1516              Chugach Census Area AK*
…                                    …
2070  Wrangell-Petersburg Census Area AK*
2074  Wrangell-Petersburg Census Area AK*
2076  Wrangell-Petersburg Census Area AK*
2077  Wrangell-Petersburg Census Area AK*
5391                       Broomfield CO*

[119 rows x 9 columns]
```

[36]:
```
#remove comma and fill na values with state avg gdp and convert to numeric

regional_gdp_df['value'] = regional_gdp_df['value'].replace(',','', regex=True)

regional_gdp_df['value'] = regional_gdp_df['value'].replace('(NA)',np.nan,␣
↪regex=True)

regional_gdp_df['value'] = pd.to_numeric(regional_gdp_df['value'])

regional_gdp_df['value'] = regional_gdp_df['value'].fillna(regional_gdp_df.
↪groupby('geofips')['value'].transform('mean'))

regional_gdp_df['state'] = regional_gdp_df['state'].str.replace(' ', '')
```

[37]:
```
#group by state and see the avg gdp by state/year
grouped_df = regional_gdp_df.groupby(['state','year']).
↪agg(mean_gdp_per_state_year = ('value', 'mean'))
```

[38]:
```
#remove scientific notation
def to_full_value(x):
    return format(x, 'f')

grouped_df['mean_gdp_per_state_year'] = grouped_df['mean_gdp_per_state_year'].
↪apply(to_full_value)
```

[39]:
```
grouped_df = grouped_df.reset_index()
```

[40]:
```
grouped_df['mean_gdp_per_state_year'] = pd.
↪to_numeric(grouped_df['mean_gdp_per_state_year'])
```

[41]:
```
#lets just do 1 state to not crowd the dataframe
cali = grouped_df[grouped_df['state'] == 'CA']
```

[42]:
```
grouped_df[grouped_df['state'] == 'CA']
```

[42]:

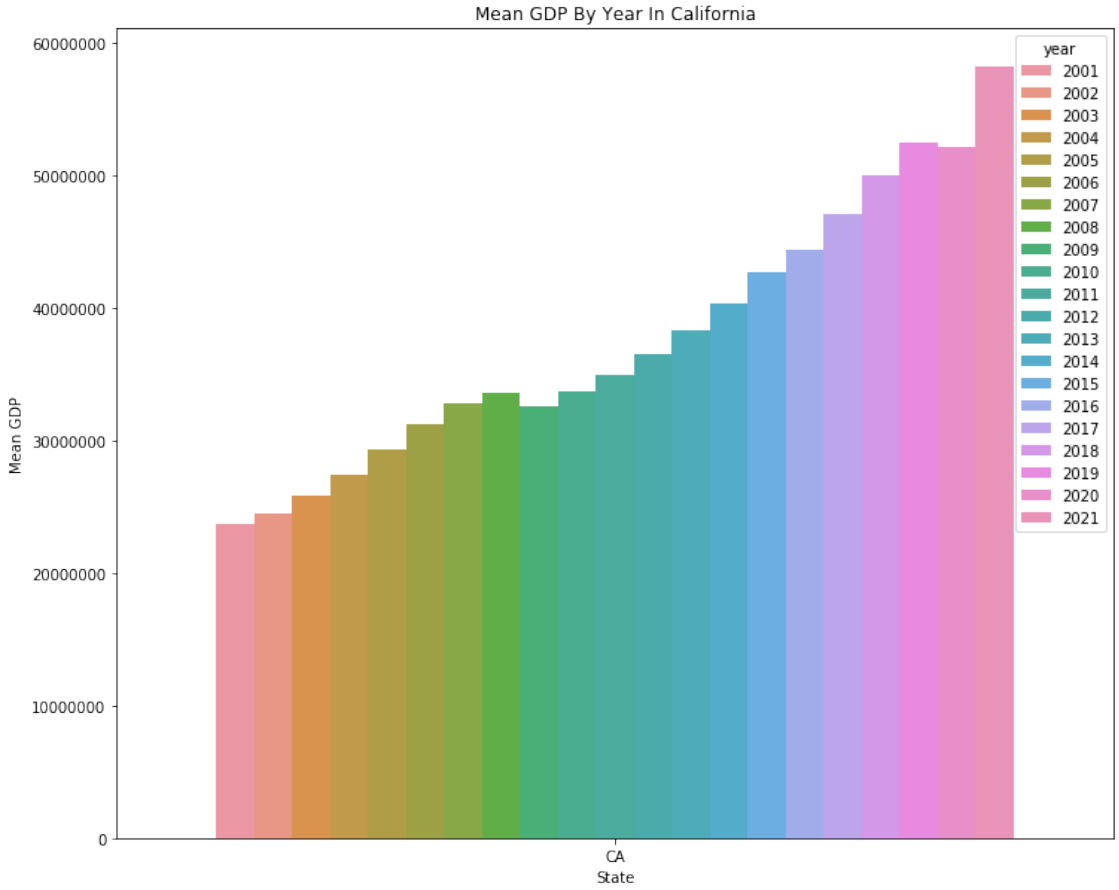
	state	year	mean_gdp_per_state_year
84	CA	2001	2.372002e+07
85	CA	2002	2.445568e+07
86	CA	2003	2.582618e+07
87	CA	2004	2.738237e+07
88	CA	2005	2.928552e+07
89	CA	2006	3.124500e+07
90	CA	2007	3.273969e+07
91	CA	2008	3.352923e+07
92	CA	2009	3.258907e+07
93	CA	2010	3.369125e+07
94	CA	2011	3.488793e+07
95	CA	2012	3.643270e+07
96	CA	2013	3.828258e+07
97	CA	2014	4.026356e+07
98	CA	2015	4.264751e+07
99	CA	2016	4.430403e+07
100	CA	2017	4.704729e+07
101	CA	2018	4.995174e+07
102	CA	2019	5.246024e+07
103	CA	2020	5.207195e+07
104	CA	2021	5.815932e+07

In this barplot you can see the mean GDP for the state of california is increasing pretty rapidly. Aside from the dip around 2009, the GDP is only getting larger.

[43]:
```
# plot barplot
plt.figure(figsize=(12, 10))

cali_gdp = sns.barplot(x = 'state',
            y = 'mean_gdp_per_state_year',
            hue = 'year',
            data = cali)
cali_gdp.yaxis.get_major_formatter().set_scientific(False)
cali_gdp.set_ylabel('Mean GDP')
cali_gdp.set_xlabel('State')
cali_gdp.set_title('Mean GDP By Year In California')
```

[43]:
```
Text(0.5, 1.0, 'Mean GDP By Year In California')
```



[44]:
```
grouped_df
```

[44]:

	state	year	mean_gdp_per_state_year
0	AK	2001	9.175888e+05
1	AK	2002	9.547297e+05
2	AK	2003	1.021820e+06
3	AK	2004	1.117835e+06
4	AK	2005	1.266487e+06
…	…	…	…
1066	WY	2017	1.607854e+06
1067	WY	2018	1.703115e+06
1068	WY	2019	1.714263e+06
1069	WY	2020	1.579582e+06
1070	WY	2021	1.804792e+06

[1071 rows x 3 columns]

[45]:
```
regional_gdp_df_minus_dc = grouped_df[grouped_df['state'] != 'DC']
```

[46]:
```
grouped_df = regional_gdp_df_minus_dc.groupby(['state']).
 ↪agg(mean_gdp_per_state_year = ('mean_gdp_per_state_year', 'mean'))
top_5 = grouped_df['mean_gdp_per_state_year'].nlargest(5)
top_5 = top_5.reset_index()
```

This graph shows the top 5 highest AVG GDP from 2021. California takes the highest, but the rest are east coast states.

[47]:
```
# plot barplot
plt.figure(figsize=(8, 6))

top_gdp = sns.barplot(x = 'state',
            y = 'mean_gdp_per_state_year',
            hue = 'state',
            data = top_5)
top_gdp.yaxis.get_major_formatter().set_scientific(False)
top_gdp.set_ylabel('Mean GDP')
top_gdp.set_xlabel('State')
top_gdp.set_title('Top 5 States With Highest Avg GDP From 2021')
```

[47]:
```
Text(0.5, 1.0, 'Top 5 States With Highest Avg GDP From 2021')
```