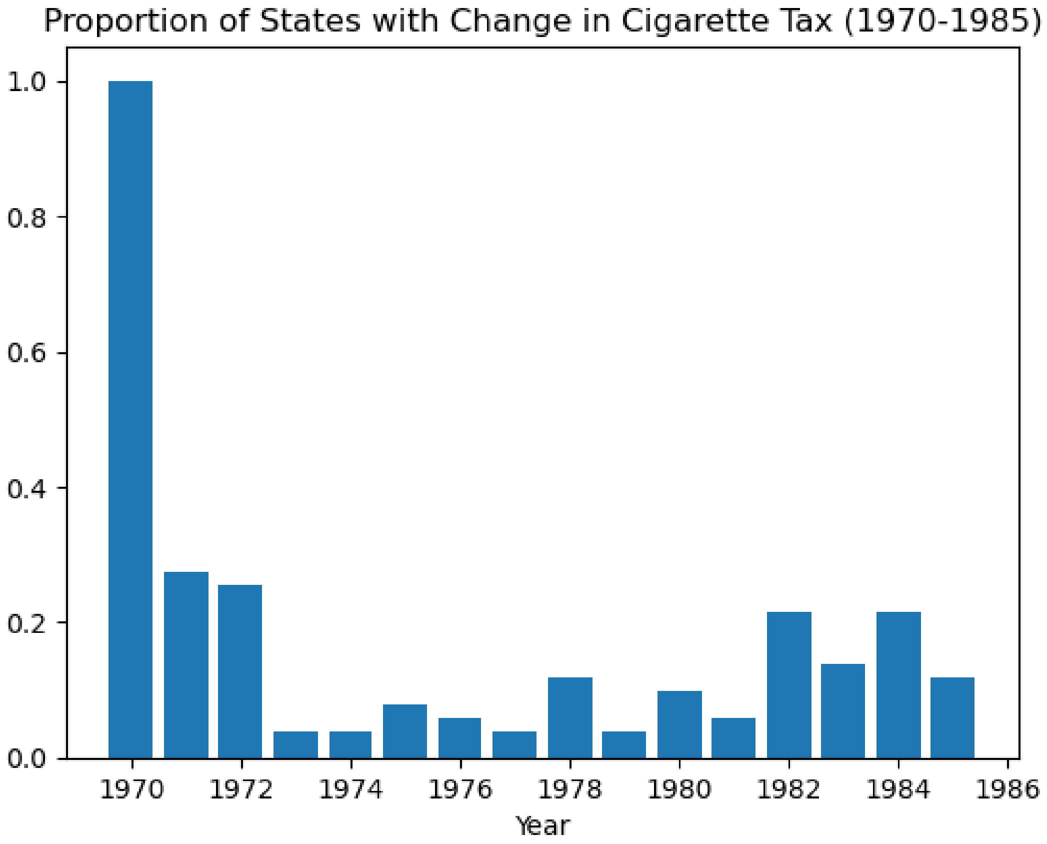
```
#calling packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import pyfixest as pf
#importting data
tax_data = pd.read_csv('/Users/avanthpakanati/Desktop/ECON:HLTH Research Seminar /Homework3/d
```

```
ModuleNotFoundError: No module named 'pyfixest'
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Input In [6], in <cell line: 7>()
      5 import seaborn as sns
      6 import statsmodels.api as sm
----> 7 import pyfixest as pf
      8 #importting data
      9 tax_data = pd.read_csv('/Users/avanthpakanati/Desktop/ECON:HLTH Research Seminar /Hom
ModuleNotFoundError: No module named 'pyfixest'
```

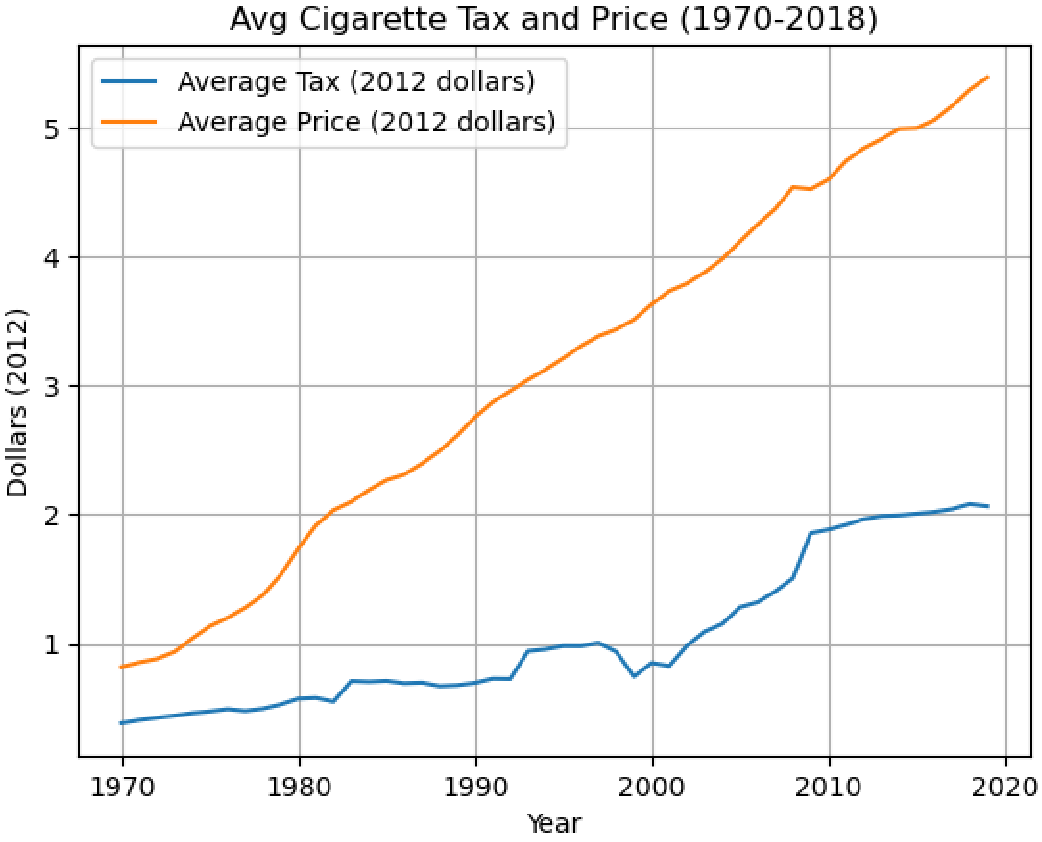
```
#summarize the data
#question 1

#finding prop of states that had change in tax from 1970-1986
tax_data = tax_data.sort_values(by=['state', 'Year'])
tax_data_1 = tax_data[(tax_data['Year']>=1970) & (tax_data['Year']<=1985)]
tax_data_1['tax_change'] = tax_data_1.groupby('state')['tax_state'].diff().ne(0).astype(int)
tax_change = tax_data_1.groupby('Year')['tax_change'].mean()


plt.bar(tax_change.index, tax_change.values)
plt.title('Proportion of States with Change in Cigarette Tax (1970-1985)')
plt.xlabel('Year')
plt.show()
```

```
/var/folders/2q/wzjp_2kd355b8clhzqwmytb40000gn/T/ipykernel_2702/86190851.py:7: SettingWithCop
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
```

```
tax_data_1['tax_change'] = tax_data_1.groupby('state')['tax_state'].diff().ne(0).astype(in
```



```
#question 2
#in 2012 $$$
cpi_2012 = tax_data.loc[tax_data['Year'] == 2012, 'price_cpi'].iloc[0]
tax_data['tax_dollar_2012'] = tax_data['tax_dollar'] * (cpi_2012 / tax_data['price_cpi'])
tax_data['price_per_pack_2012'] = tax_data['cost_per_pack'] * (cpi_2012 / tax_data['price_cp

avg_values = tax_data.groupby('Year')[['tax_dollar_2012', 'price_per_pack_2012']].mean()

#plot graph
plt.plot(avg_values.index, avg_values['tax_dollar_2012'], label='Average Tax (2012 dollars)'
plt.plot(avg_values.index, avg_values['price_per_pack_2012'], label='Average Price (2012 dol
plt.legend()
plt.title('Avg Cigarette Tax and Price (1970-2018)')
plt.xlabel('Year')
plt.ylabel('Dollars (2012)')
plt.grid(True)
plt.show()
```

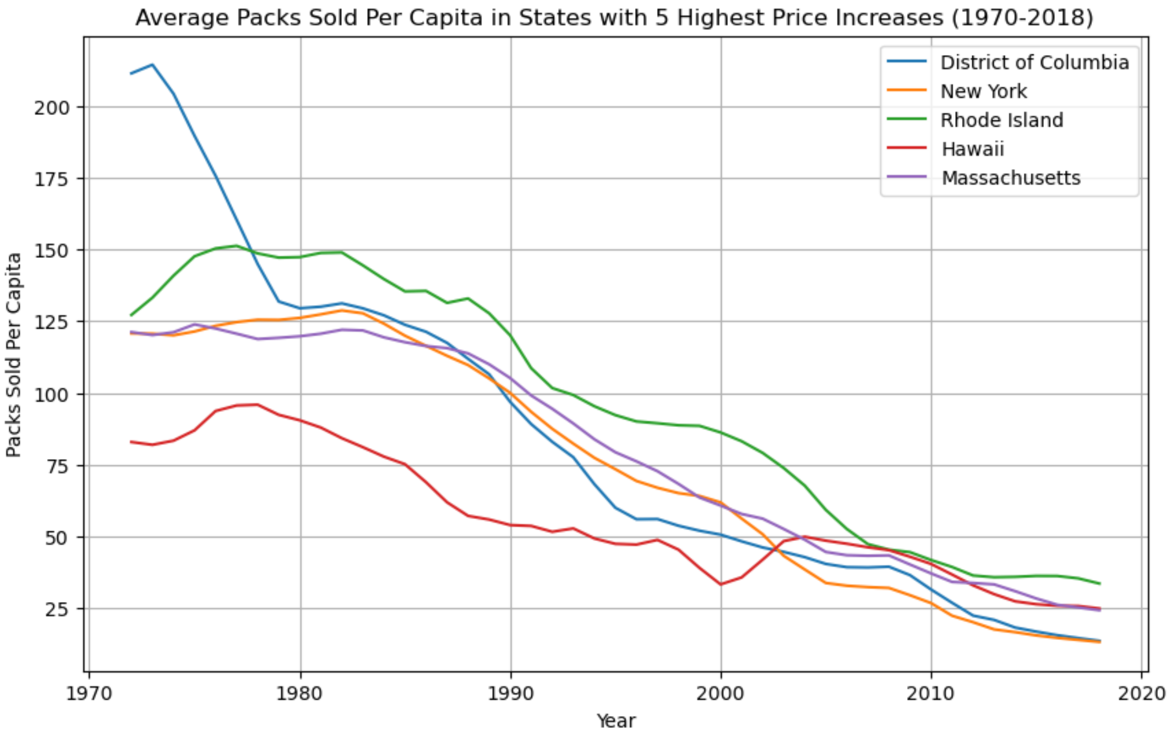


```
avg_values = tax_data.groupby('Year')[['tax_2012', 'price_cpi']].mean()


#line graph
plt.plot(avg_values.index, avg_values['tax_2012'], label='Average Tax (2012 dollars)')
plt.plot(avg_values.index, avg_values['price_cpi'], label='Average Price (2012 dollars)')
plt.legend()
plt.title('Cigarette Tax and Price in 2012 Dollars (1970-2018)')
plt.xlabel('Year')
plt.ylabel('Dollars (2012 value)')
plt.grid(True)
plt.show()
```

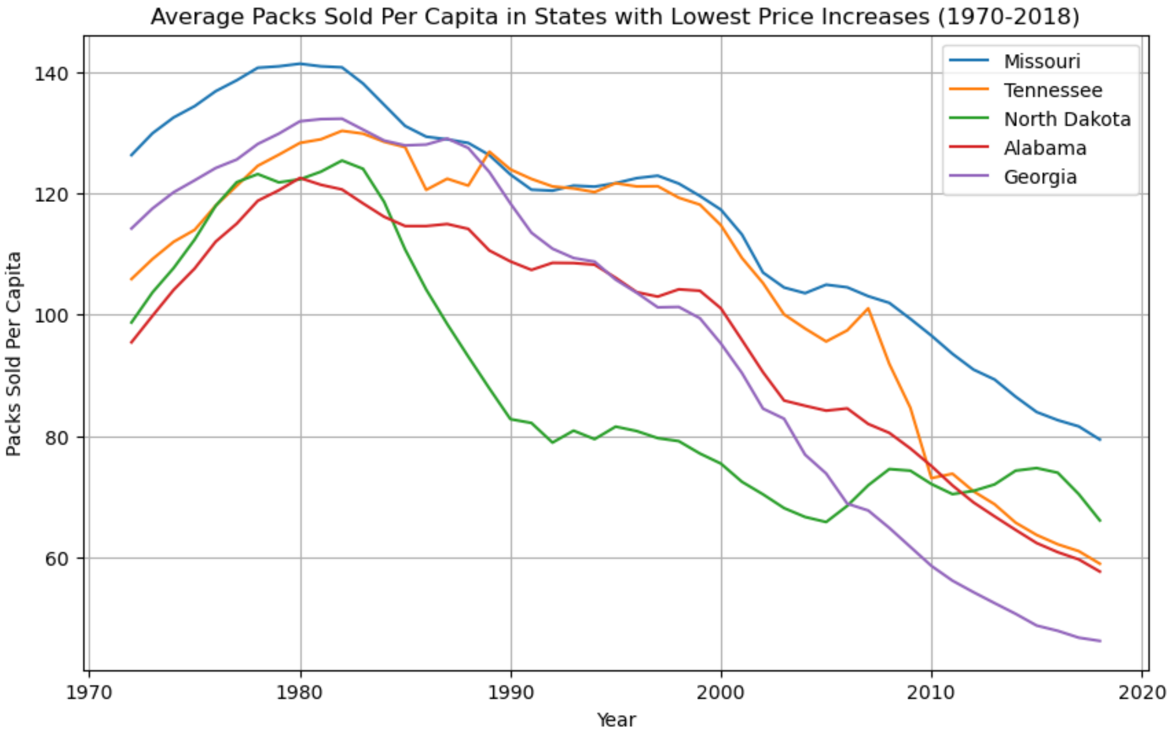
```
KeyError: "Columns not found: 'tax_2012'"
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [5], in <cell line: 1>()
----> 1 avg_values = tax_data.groupby('Year')[['tax_2012', 'price_cpi']].mean()
      4 #line graph
      5 plt.plot(avg_values.index, avg_values['tax_2012'], label='Average Tax (2012 dollars)
File ~/anaconda/lib/python3.9/site-packages/pandas/core/groupby/generic.py:1338, in DataFrame
   1329 if isinstance(key, tuple) and len(key) > 1:
   1330     # if len == 1, then it becomes a SeriesGroupBy and this is actually
   1331     # valid syntax, so don't raise warning
   1332     warnings.warn(
   1333         "Indexing with multiple keys (implicitly converted to a tuple "
   1334         "of keys) will be deprecated, use a list instead.",
   1335         FutureWarning,
   1336         stacklevel=find_stack_level(),
   1337     )
-> 1338 return super().__getitem__(key)
File ~/anaconda/lib/python3.9/site-packages/pandas/core/base.py:240, in SelectionMixin.__geti
    238     if len(self.obj.columns.intersection(key)) != len(set(key)):
    239         bad_keys = list(set(key).difference(self.obj.columns))
--> 240         raise KeyError(f"Columns not found: {str(bad_keys)[1:-1]}")
    241     return self._gotitem(list(key), ndim=2)
    243 elif not getattr(self, "as_index", False):
KeyError: "Columns not found: 'tax_2012'"
```

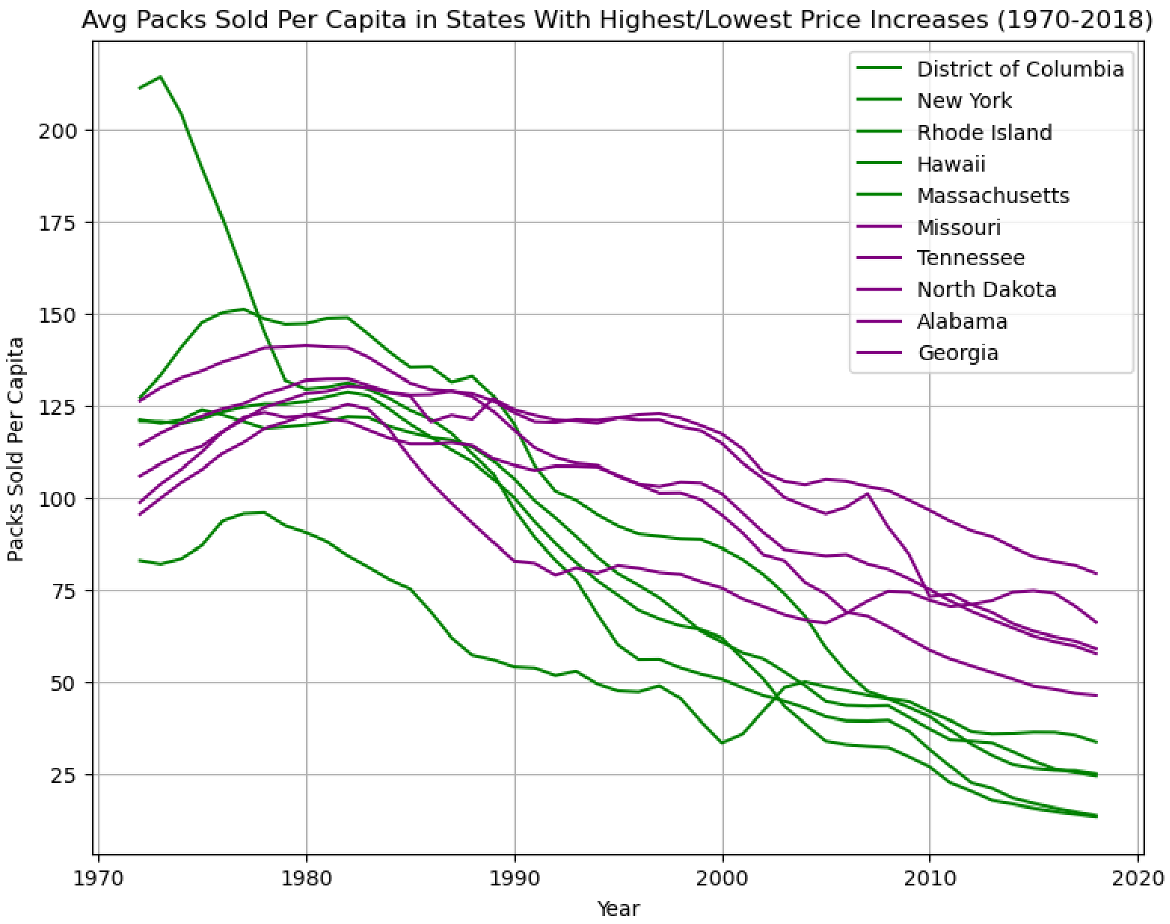
```
#question 3
#years 1970-2018
tax_data_3 = tax_data[(tax_data['Year'] >= 1970) & (tax_data['Year'] <= 2018)]

change_in_price = tax_data_3.groupby('state')['price_cpi'].agg(['first', 'last'])
change_in_price['price_increase'] = change_in_price['last'] - change_in_price['first']

#finding 5 states with highest price increase
top_5_states = change_in_price.nlargest(5, 'price_increase').index.tolist()
top_5 = tax_data_3[tax_data_3['state'].isin(top_5_states)]



#plotting the graph

plt.figure(figsize=(10, 6))
for state in top_5_states:
    state_data = top_5[top_5['state'] == state]
    plt.plot(state_data['Year'], state_data['sales_per_capita'].rolling(window=3).mean(), lab
plt.title('Average Packs Sold Per Capita in States with 5 Highest Price Increases (1970-2018
plt.xlabel('Year')
plt.ylabel('Packs Sold Per Capita')
plt.legend()
plt.grid(True)
plt.show()
```



```
#question 4
#Lowest increase in cig prices


# 5 states with the lowest price increase
bottom_5_states = change_in_price.nsmallest(5, 'price_increase').index.tolist()
bottom_5 = tax_data_3[tax_data_3['state'].isin(bottom_5_states)]

plt.figure(figsize=(10, 6))
for state in bottom_5_states:
    state_data = bottom_5[bottom_5['state'] == state]
    plt.plot(state_data['Year'], state_data['sales_per_capita'].rolling(window=3).mean(), la
plt.title('Average Packs Sold Per Capita in States with 5 Lowest Price Increases (1970-2018)
plt.xlabel('Year')
plt.ylabel('Packs Sold Per Capita')
plt.legend()
plt.grid(True)
plt.show()
```



```
#question 5 comparing states with lowest and highest price increase
#putting it all in one graph
plt.figure(figsize=(9, 7))
for state in top_5_states:
    state_data = top_5[top_5['state'] == state]
    plt.plot(state_data['Year'], state_data['sales_per_capita'].rolling(window=3).mean(), la
for state in bottom_5_states:
    state_data = bottom_5[bottom_5['state'] == state]
    plt.plot(state_data['Year'], state_data['sales_per_capita'].rolling(window=3).mean(), la



plt.title('Avg Packs Sold Per Capita in States With Highest/Lowest Price Increases (1970-201
plt.xlabel('Year')
plt.ylabel('Packs Sold Per Capita')
plt.legend()
plt.grid(True)
plt.show()
```



```
#Estimating ATE

#QUESTION 6

cig_data = tax_data[(tax_data['Year'] >= 1970) & (tax_data['Year'] <= 1990)]
cig_data['ln_sales'] = np.log(cig_data['sales_per_capita'])
cig_data['ln_total_tax'] = np.log(cig_data['tax_dollar'])
cig_data['ln_price'] = np.log(cig_data['price_cpi'])
```

```python
#  OLS
X = sm.add_constant(cig_data['ln_price'])  # Add constant for intercept
Y = cig_data['ln_sales']

model = sm.OLS(Y, X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               ln_sales   R-squared:                       0.294
Model:                            OLS   Adj. R-squared:                  0.293
Method:                 Least Squares   F-statistic:                     445.1
Date:                Tue, 18 Mar 2025   Prob (F-statistic):           6.98e-83
Time:                        13:29:17   Log-Likelihood:                 263.40
No. Observations:                1071   AIC:                            -522.8
Df Residuals:                    1069   BIC:                            -512.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.3854      0.028    193.692      0.000       5.331       5.440
ln_price      -0.8094      0.038    -21.098      0.000      -0.885      -0.734
==============================================================================
Omnibus:                       89.160   Durbin-Watson:                   0.183
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              466.536
Skew:                           0.128   Prob(JB):                    4.93e-102
Kurtosis:                       6.223   Cond. No.                         10.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

/var/folders/2q/wzjp_2kd355b8clhzqwmytb40000gn/T/ipykernel_2702/707357993.py:2: SettingWithCo
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  cig_data['ln_sales'] = np.log(cig_data['sales_per_capita'])
```

```
/var/folders/2q/wzjp_2kd355b8clhzqwmytb40000gn/T/ipykernel_2702/707357993.py:3: SettingWithCo
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  cig_data['ln_total_tax'] = np.log(cig_data['tax_dollar'])
/var/folders/2q/wzjp_2kd355b8clhzqwmytb40000gn/T/ipykernel_2702/707357993.py:4: SettingWithCo
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  cig_data['ln_price'] = np.log(cig_data['price_cpi'])
```

```python
#Question 7 and 8, log sales on log prices
#First stage
first_stage = sm.OLS(cig_data['ln_price'], sm.add_constant(cig_data['ln_total_tax'])).fit()
print("First stage Regression (ln_price ~ ln_total_tax):\n")
print(first_stage.summary())
```

```
First stage Regression (ln_price ~ ln_total_tax):

                            OLS Regression Results
==============================================================================
Dep. Variable:               ln_price   R-squared:                       0.683
Model:                            OLS   Adj. R-squared:                  0.683
Method:                 Least Squares   F-statistic:                     2301.
Date:                Mon, 17 Mar 2025   Prob (F-statistic):          8.21e-269
Time:                        09:08:01   Log-Likelihood:                -86.164
No. Observations:                1071   AIC:                             176.3
Df Residuals:                    1069   BIC:                             186.3
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            1.1786      0.033     35.712      0.000       1.114       1.243
ln_total_tax     1.0803      0.023     47.973      0.000       1.036       1.125
==============================================================================
Omnibus:                       30.760   Durbin-Watson:                   0.408
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               32.668
Skew:                           0.421   Prob(JB):                     8.06e-08
Kurtosis:                       3.156   Cond. No.                         8.72
```

```
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
# Log prices from the first stage
price_hat = first_stage.predict(sm.add_constant(cig_data['ln_total_tax']))

# Second-stage regression (IV Regression)
second_stage = sm.OLS(cig_data['ln_sales'], sm.add_constant(price_hat)).fit()
print("\nSecond stage Regression (ln_sales ~ pricehat):\n")
print(second_stage.summary())
```

```
Second stage Regression (ln_sales ~ pricehat):

                            OLS Regression Results
==============================================================================
Dep. Variable:               ln_sales   R-squared:                       0.236
Model:                            OLS   Adj. R-squared:                  0.235
Method:                 Least Squares   F-statistic:                     330.3
Date:                Mon, 17 Mar 2025   Prob (F-statistic):           1.56e-64
Time:                        09:08:01   Log-Likelihood:                 221.17
No. Observations:                1071   AIC:                            -438.3
Df Residuals:                    1069   BIC:                            -428.4
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.7101      0.008    573.443      0.000       4.694       4.726
0             -0.2843      0.016    -18.175      0.000      -0.315      -0.254
==============================================================================
Omnibus:                       83.338   Durbin-Watson:                   0.157
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              430.014
Skew:                           0.023   Prob(JB):                     4.20e-94
Kurtosis:                       6.104   Cond. No.                         2.98
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```python
#Question 9 - repeat for 1991-2015
cig_data2 = tax_data[(tax_data['Year'] >= 1991) & (tax_data['Year'] <= 2015)]

cig_data2['ln_sales'] = np.log(cig_data2['sales_per_capita'])
cig_data2['ln_price'] = np.log(cig_data2['cost_per_pack'])
cig_data2['ln_total_tax'] = np.log(cig_data2['tax_dollar'])

#running OLS regression
X2 = sm.add_constant(cig_data2['ln_price'])
Y2 = cig_data2['ln_sales']


reg2 = sm.OLS(Y2, X2).fit()
print(reg2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               ln_sales   R-squared:                       0.533
Model:                            OLS   Adj. R-squared:                  0.532
Method:                 Least Squares   F-statistic:                     1451.
Date:                Mon, 17 Mar 2025   Prob (F-statistic):         1.52e-212
Time:                        09:08:01   Log-Likelihood:                -296.47
No. Observations:                1275   AIC:                             596.9
Df Residuals:                    1273   BIC:                             607.2
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.0395      0.023    219.934      0.000       4.995       5.084
ln_price      -0.6656      0.017    -38.094      0.000      -0.700      -0.631
==============================================================================
Omnibus:                       19.351   Durbin-Watson:                   0.158
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               33.046
Skew:                           0.064   Prob(JB):                     6.67e-08
Kurtosis:                       3.778   Cond. No.                         5.37
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

/var/folders/2q/wzjp_2kd355b8clhzqwmytb40000gn/T/ipykernel_90894/1083457502.py:4: SettingWit
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  cig_data2['ln_sales'] = np.log(cig_data2['sales_per_capita'])
/var/folders/2q/wzjp_2kd355b8clhzqwmytb40000gn/T/ipykernel_90894/1083457502.py:5: SettingWith
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  cig_data2['ln_price'] = np.log(cig_data2['cost_per_pack'])
/var/folders/2q/wzjp_2kd355b8clhzqwmytb40000gn/T/ipykernel_90894/1083457502.py:6: SettingWith
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide
  cig_data2['ln_total_tax'] = np.log(cig_data2['tax_dollar'])
```

```
#QUESTION 10
# Creating a table to summarize results
summary_table = pd.DataFrame({
    'Years': ['1970-1990', '1970-1990', '1991-2015', '1991-2015'],
```

```
ValueError: All arrays must be of the same length
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [29], in <cell line: 3>()
      1 #QUESTION 10
      2 # Creating a table to summarize results
----> 3 results_table = pd.DataFrame(      4     'Time Period': ['1970-1990', '1970-1990', '
     13 # Print the table without the index and with lines between each column/row
     14 print(results_table.to_string(index=False, line_width=80))
File ~/anaconda/lib/python3.9/site-packages/pandas/core/frame.py:636, in DataFrame.__init__(s
    630     mgr = self._init_mgr(
    631         data, axes="index": index, "columns": columns, dtype=dtype, copy=copy
    632     )
    634 elif isinstance(data, dict):
    635     # GH#38939 de facto copy defaults to False only in non-dict cases
--> 636     mgr = dict_to_mgr(data, index, columns, dtype=dtype, copy=copy, typ=manager)
    637 elif isinstance(data, ma.MaskedArray):
    638     import numpy.ma.mrecords as mrecords
File ~/anaconda/lib/python3.9/site-packages/pandas/core/internals/construction.py:502, in di
```

```
    494     arrays = [
    495         x
    496         if not hasattr(x, "dtype") or not isinstance(x.dtype, ExtensionDtype)
    497         else x.copy()
    498         for x in arrays
    499     ]
    500     # TODO: can we get rid of the dtype special case above?
--> 502 return arrays_to_mgr(arrays, columns, index, dtype=dtype, typ=typ, consolidate=copy)
File ~/anaconda/lib/python3.9/site-packages/pandas/core/internals/construction.py:120, in ar
    117 if verify_integrity:
    118     # figure out the index, if necessary
    119     if index is None:
--> 120         index = _extract_index(arrays)
    121     else:
    122         index = ensure_index(index)
File ~/anaconda/lib/python3.9/site-packages/pandas/core/internals/construction.py:674, in _e
    672 lengths = list(set(raw_lengths))
    673 if len(lengths) > 1:
--> 674     raise ValueError("All arrays must be of the same length")
    676 if have_dicts:
    677     raise ValueError(
    678         "Mixing dicts with non-Series may lead to ambiguous ordering."
    679     )
ValueError: All arrays must be of the same length
```