Assessment 8 ETL Report

Authors: Shervorn Mathews,
Janeel Abrahams,
Jin Hee Lee

ETL Process Date: 4/19/2023

Table Of Contents

Introduction

The following document provides a step by step guide on how to extract, transform and load data obtained from the Annual Business Survey (ABS) APIs for 2019 hosted on www.census.gov/. The data obtained from this API can be used to answer questions regarding the ethnicity, race, veteran status of employers as well as the revenue, employees size, years of business and annual payroll of the business that participated in the survey.

Extraction

The data within this report was extracted from the Annual Business Survey (ABS) APIs for 2019 hosted on www.census.gov/.

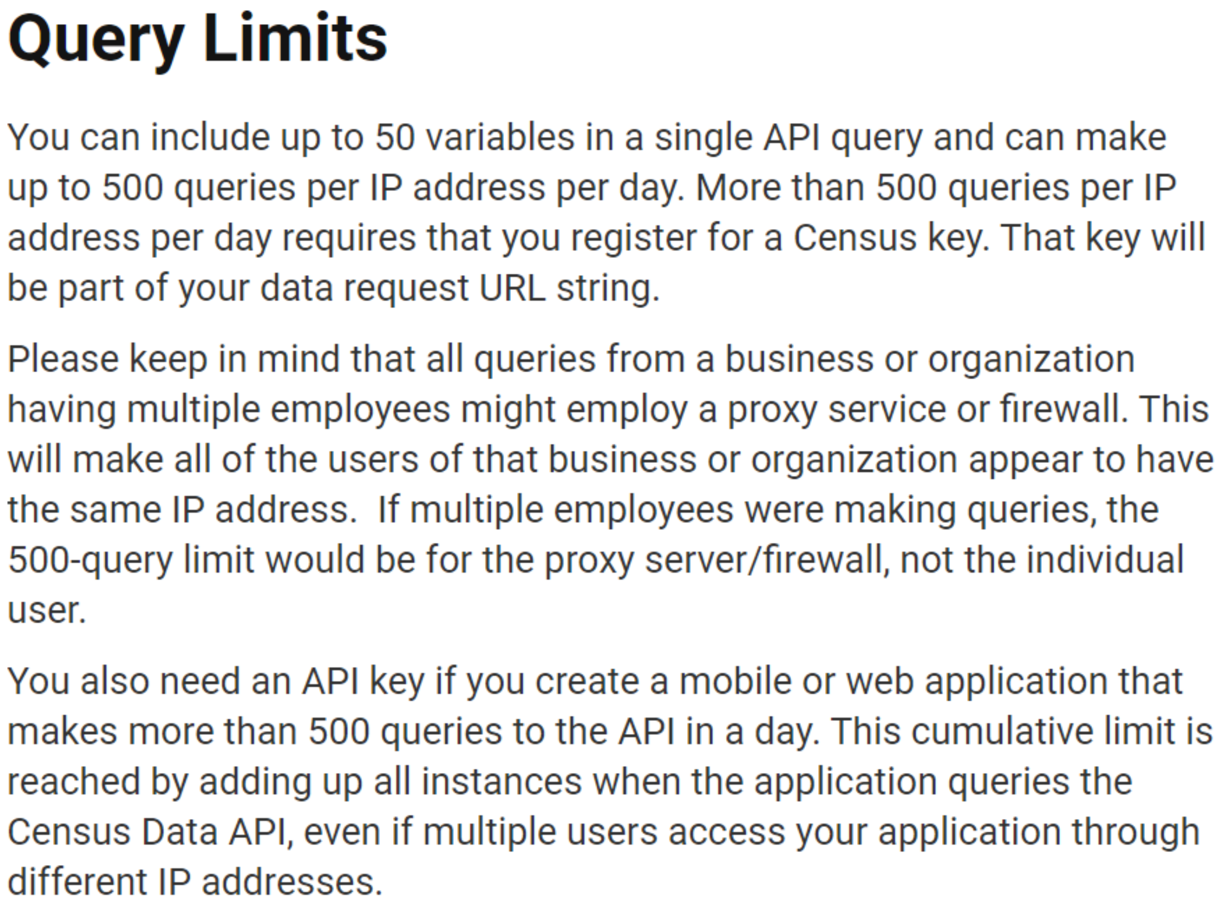
An API guide can be found in the following link:

https://www.census.gov/data/developers/guidance/api-user-guide.Help_&_Contact_Us.html

An API can be obtained from the following link:

https://api.census.gov/data/key_signup.html

Note: The API doesn't require a key to be called although it states it does at its end. Calling the API without obtaining a key invokes a query limit found in the API guide stated below:



Query Limits

You can include up to 50 variables in a single API query and can make up to 500 queries per IP address per day. More than 500 queries per IP address per day requires that you register for a Census key. That key will be part of your data request URL string.

Please keep in mind that all queries from a business or organization having multiple employees might employ a proxy service or firewall. This will make all of the users of that business or organization appear to have the same IP address. If multiple employees were making queries, the 500-query limit would be for the proxy server/firewall, not the individual user.

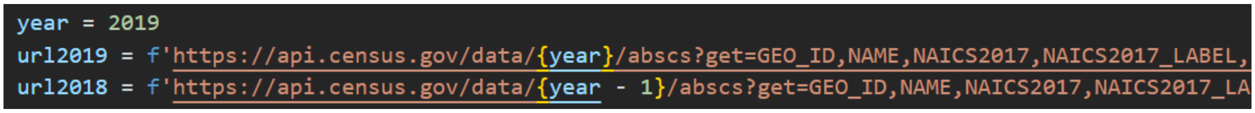
You also need an API key if you create a mobile or web application that makes more than 500 queries to the API in a day. This cumulative limit is reached by adding up all instances when the application queries the Census Data API, even if multiple users access your application through different IP addresses.

An example of an API call being used on the year 2018 can be found below:

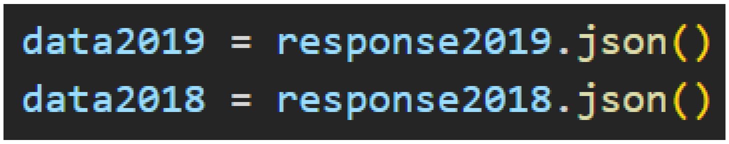
api.census.gov/data/2018/abscs?get=GEO_ID,NAME,NAICS2017,NAICS2017_LABEL,SEX,SEX_LABEL,ETH_GROUP,ETH_GROUP_LABEL,RACE_GROUP,RACE_GROUP_LABEL,VET_GROUP,VET_GROUP_LABEL,EMPSZFI,EMPSZFI_LABEL,YEAR,FIRMPDEMP,FIRMPDEMP_F,RCPPDEMP,RCPPDEMP_F,EMP,EMP_F,PAYANN,PAYANN_F,FIRMPDEMP_S,FIRMPDEMP_S_F,RCPPDEMP_S,RCPPDEMP_S_F,EMP_S,EMP_S_F,PAYANN_S,PAYANN_S_F&for=us:*&key=YOUR_KEY_GOES_HERE

Snippets of the code calling the API in this report can be found below.

Note: This snippet of code focuses on the year 2019 per the requirements of the assigned project as well as the year 2018:



```
year = 2019
url2019 = f'https://api.census.gov/data/{year}/abscs?get=GEO_ID,NAME,NAICS2017,NAICS2017_LABEL,
url2018 = f'https://api.census.gov/data/{year - 1}/abscs?get=GEO_ID,NAME,NAICS2017,NAICS2017_LA
```



```
data2019 = response2019.json()
data2018 = response2018.json()
```

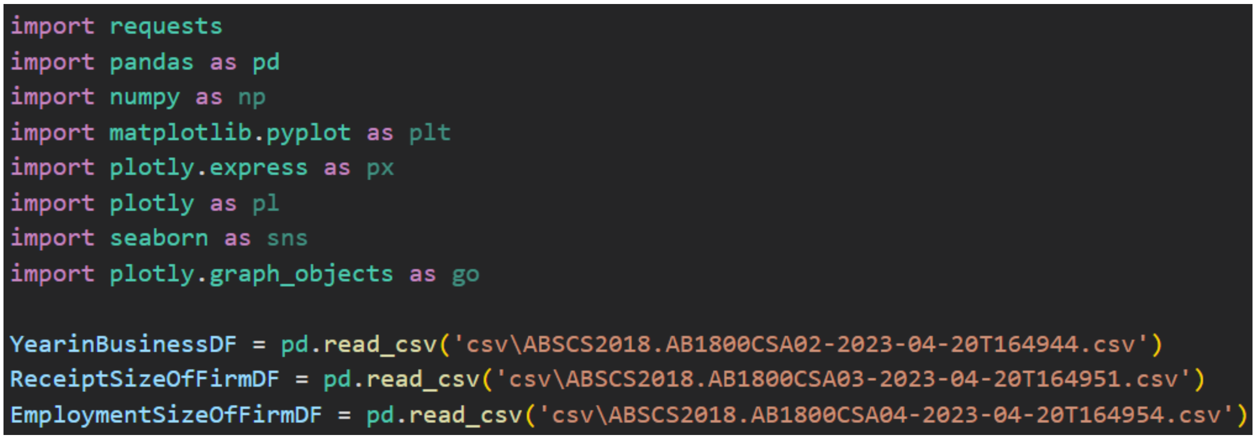
Supplementary datasets were also obtained from the following links:

https://www.census.gov/data/tables/2019/econ/abs/2019-abs-company-summary.html

https://data.census.gov/table?tid=ABSCS2018.AB1800CSA02&hidePreview=true

https://data.census.gov/table?tid=ABSCS2018.AB1800CSA03&hidePreview=true

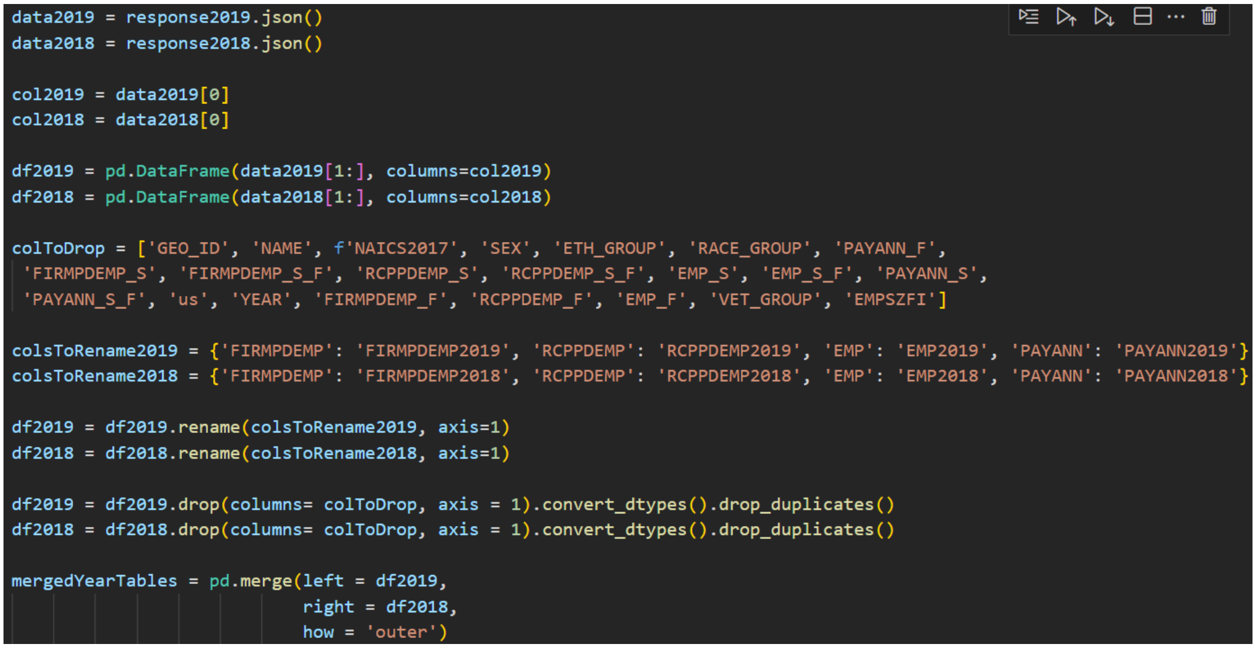
https://data.census.gov/table?tid=ABSCS2018.AB1800CSA04&hidePreview=true



```
import requests
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly as pl
import seaborn as sns
import plotly.graph_objects as go

YearinBusinessDF = pd.read_csv('csv\ABSCS2018.AB1800CSA02-2023-04-20T164944.csv')
ReceiptSizeOfFirmDF = pd.read_csv('csv\ABSCS2018.AB1800CSA03-2023-04-20T164951.csv')
EmploymentSizeOfFirmDF = pd.read_csv('csv\ABSCS2018.AB1800CSA04-2023-04-20T164954.csv')
```

Transform

The following snippet of code explains how the API data was cleaned and merged.

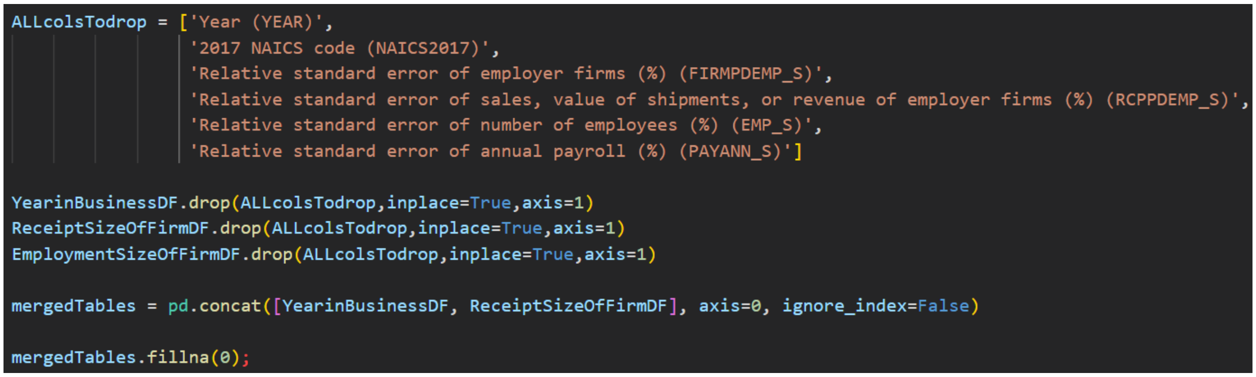


```python
data2019 = response2019.json()
data2018 = response2018.json()

col2019 = data2019[0]
col2018 = data2018[0]

df2019 = pd.DataFrame(data2019[1:], columns=col2019)
df2018 = pd.DataFrame(data2018[1:], columns=col2018)

colToDrop = ['GEO_ID', 'NAME', f'NAICS2017', 'SEX', 'ETH_GROUP', 'RACE_GROUP', 'PAYANN_F',
 'FIRMPDEMP_S', 'FIRMPDEMP_S_F', 'RCPPDEMP_S', 'RCPPDEMP_S_F', 'EMP_S', 'EMP_S_F', 'PAYANN_S',
 'PAYANN_S_F', 'us', 'YEAR', 'FIRMPDEMP_F', 'RCPPDEMP_F', 'EMP_F', 'VET_GROUP', 'EMPSZFI']

colsToRename2019 = {'FIRMPDEMP': 'FIRMPDEMP2019', 'RCPPDEMP': 'RCPPDEMP2019', 'EMP': 'EMP2019', 'PAYANN': 'PAYANN2019'}
colsToRename2018 = {'FIRMPDEMP': 'FIRMPDEMP2018', 'RCPPDEMP': 'RCPPDEMP2018', 'EMP': 'EMP2018', 'PAYANN': 'PAYANN2018'}

df2019 = df2019.rename(colsToRename2019, axis=1)
df2018 = df2018.rename(colsToRename2018, axis=1)

df2019 = df2019.drop(columns= colToDrop, axis = 1).convert_dtypes().drop_duplicates()
df2018 = df2018.drop(columns= colToDrop, axis = 1).convert_dtypes().drop_duplicates()

mergedYearTables = pd.merge(left = df2019,
                            right = df2018,
                            how = 'outer')
```

It can summarized as follows:

1. Receive the API data as json files
2. Take the first row of data as a variable
3. Create dataframes with pandas and use the variables stored previously as the headers
4. Create a list filled with the names of redundant columns called in with the API
5. Create a dictionary filled with the names of columns to replace as needed
6. Rename columns as needed
7. Drop columns as needed
8. Merge final table

The following snippet of code explains how the supplementary data was cleaned and merged.



```python
ALLcolsTodrop = ['Year (YEAR)',
                 '2017 NAICS code (NAICS2017)',
                 'Relative standard error of employer firms (%) (FIRMPDEMP_S)',
                 'Relative standard error of sales, value of shipments, or revenue of employer firms (%) (RCPPDEMP_S)',
                 'Relative standard error of number of employees (%) (EMP_S)',
                 'Relative standard error of annual payroll (%) (PAYANN_S)']

YearinBusinessDF.drop(ALLcolsTodrop,inplace=True,axis=1)
ReceiptSizeOfFirmDF.drop(ALLcolsTodrop,inplace=True,axis=1)
EmploymentSizeOfFirmDF.drop(ALLcolsTodrop,inplace=True,axis=1)

mergedTables = pd.concat([YearinBusinessDF, ReceiptSizeOfFirmDF], axis=0, ignore_index=False)

mergedTables.fillna(0);
```

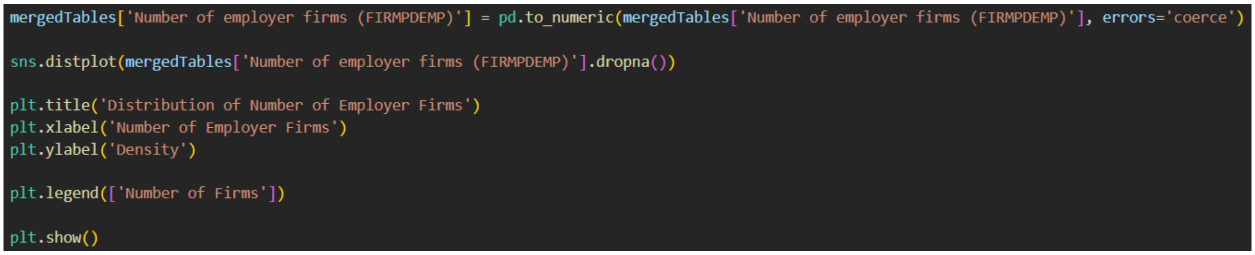
It can summarized as follows:

1. Create a list filled with the names of redundant columns

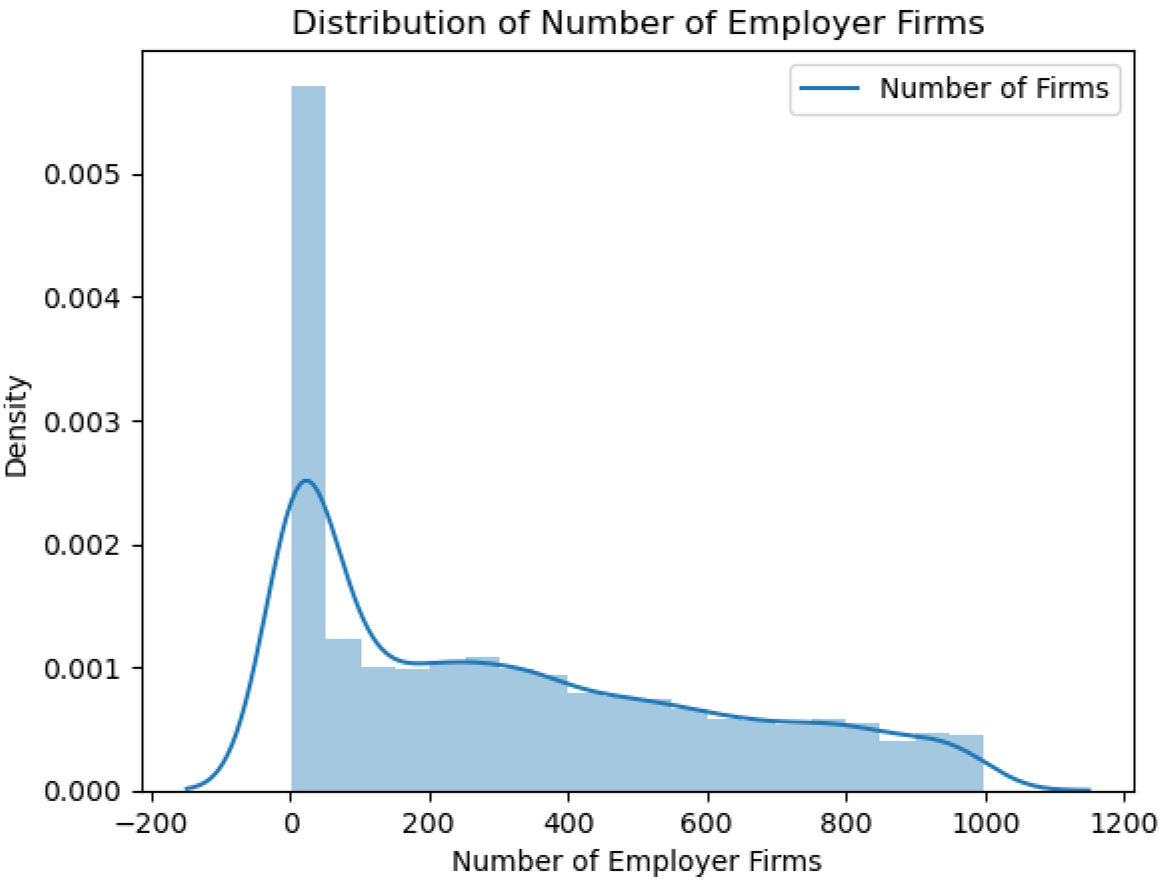
2. Drop columns as needed
3. Merge final table
4. Replace all NaN values with 0

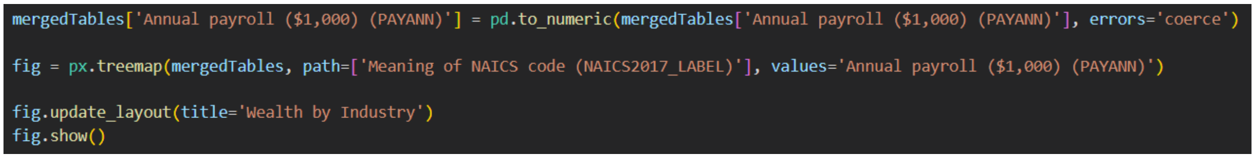
Load

The following section features examples of how the data collected from the API and the supplementary csv can be used to create visuals where the snippet of code is followed by the visual it creates.

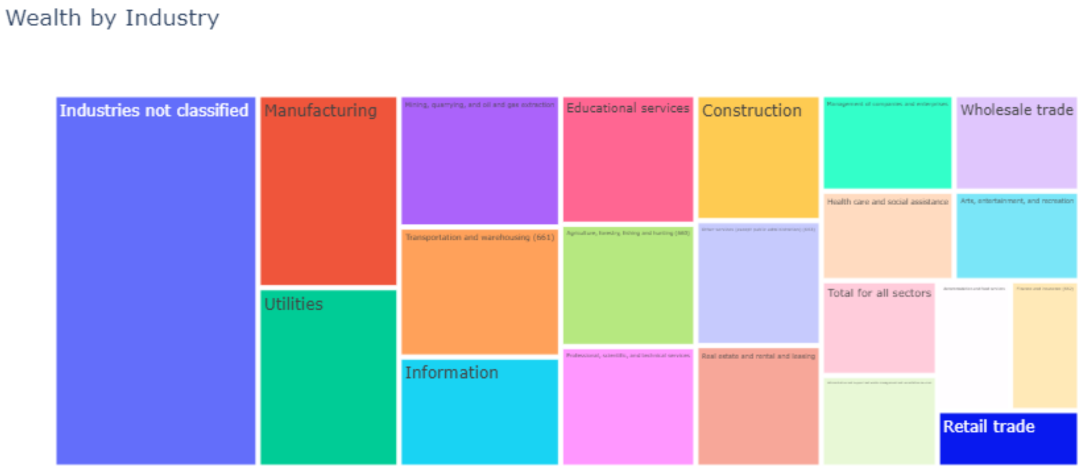


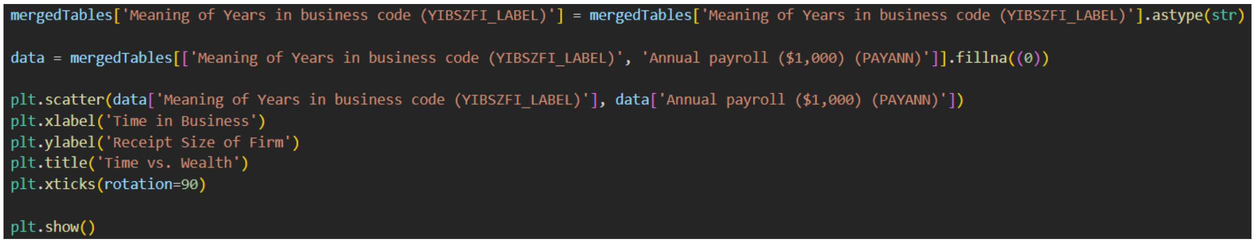
```
mergedTables['Number of employer firms (FIRMPDEMP)'] = pd.to_numeric(mergedTables['Number of employer firms (FIRMPDEMP)'], errors='coerce')

sns.distplot(mergedTables['Number of employer firms (FIRMPDEMP)'].dropna())

plt.title('Distribution of Number of Employer Firms')
plt.xlabel('Number of Employer Firms')
plt.ylabel('Density')

plt.legend(['Number of Firms'])

plt.show()
```



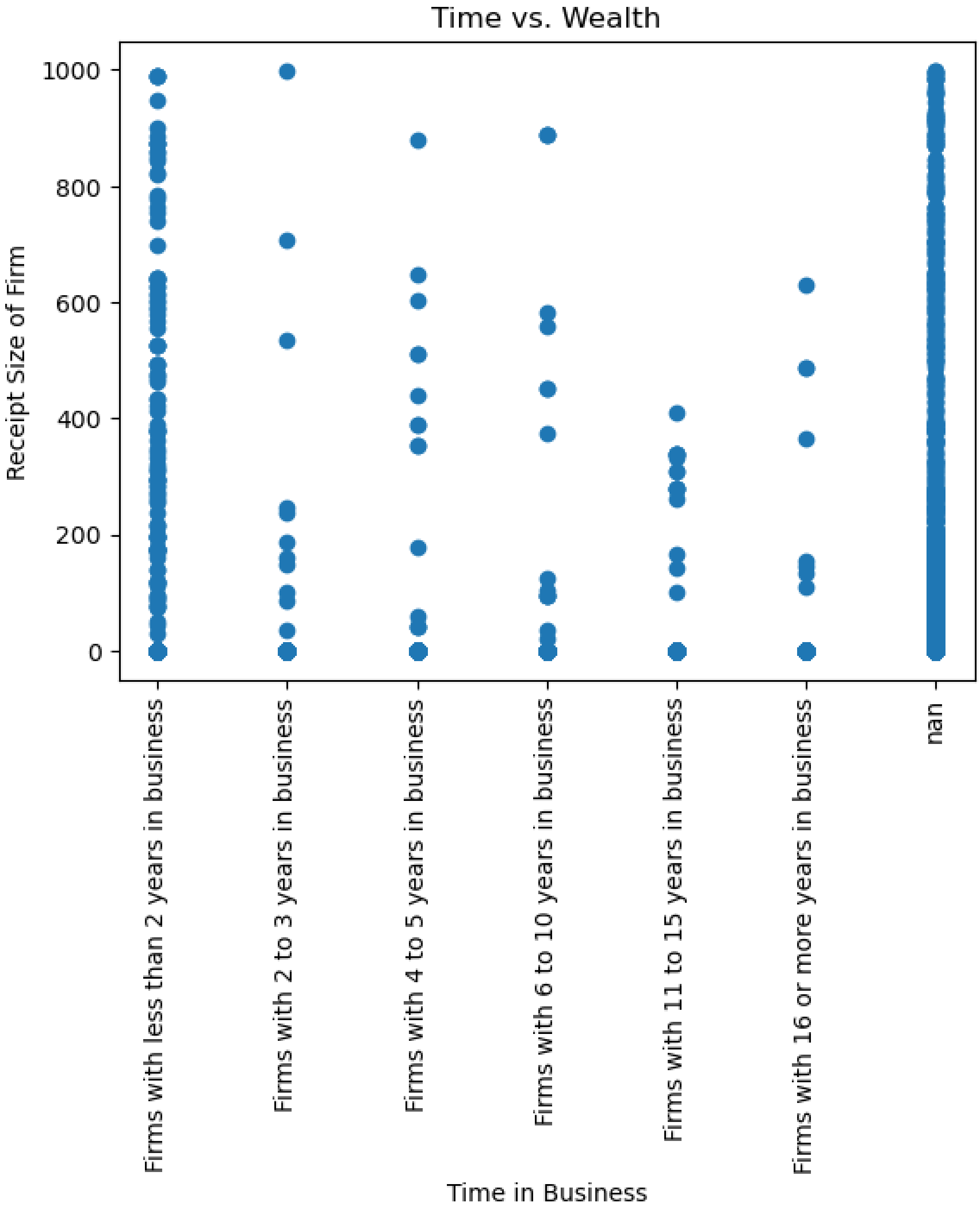


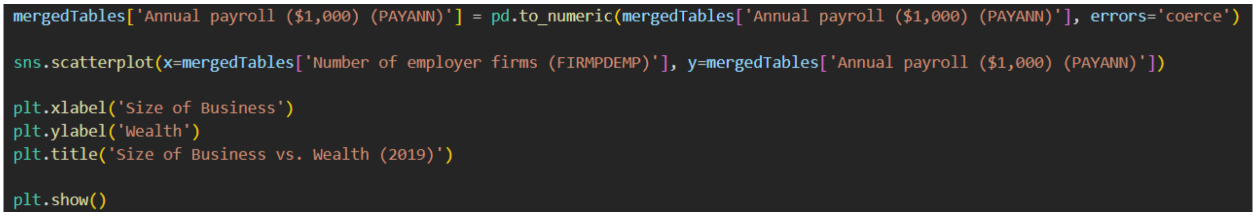
```python
mergedTables['Annual payroll ($1,000) (PAYANN)'] = pd.to_numeric(mergedTables['Annual payroll ($1,000) (PAYANN)'], errors='coerce')

fig = px.treemap(mergedTables, path=['Meaning of NAICS code (NAICS2017_LABEL)'], values='Annual payroll ($1,000) (PAYANN)')

fig.update_layout(title='Wealth by Industry')
fig.show()
```



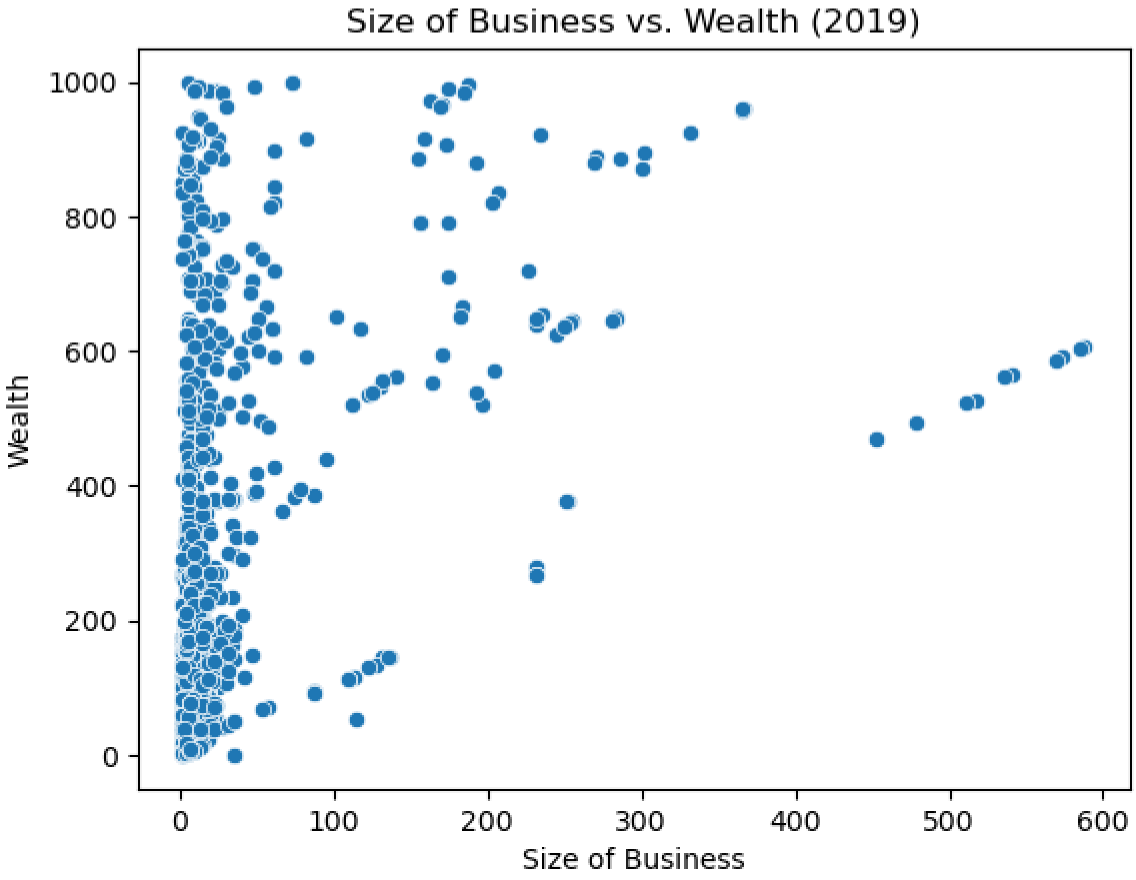


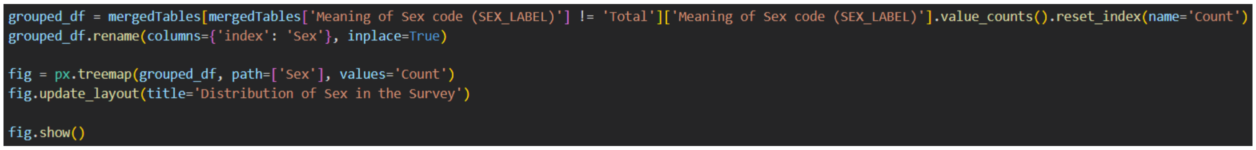
```python
mergedTables['Meaning of Years in business code (YIBSZFI_LABEL)'] = mergedTables['Meaning of Years in business code (YIBSZFI_LABEL)'].astype(str)

data = mergedTables[['Meaning of Years in business code (YIBSZFI_LABEL)', 'Annual payroll ($1,000) (PAYANN)']].fillna((0))

plt.scatter(data['Meaning of Years in business code (YIBSZFI_LABEL)'], data['Annual payroll ($1,000) (PAYANN)'])
plt.xlabel('Time in Business')
plt.ylabel('Receipt Size of Firm')
plt.title('Time vs. Wealth')
plt.xticks(rotation=90)

plt.show()
```


Time vs. Wealth
Receipt Size of Firm
1000
800
600
400
200
0
Firms with less than 2 years in business
Firms with 2 to 3 years in business
Firms with 4 to 5 years in business
Firms with 6 to 10 years in business
Firms with 11 to 15 years in business
Firms with 16 or more years in business
nan
Time in Business

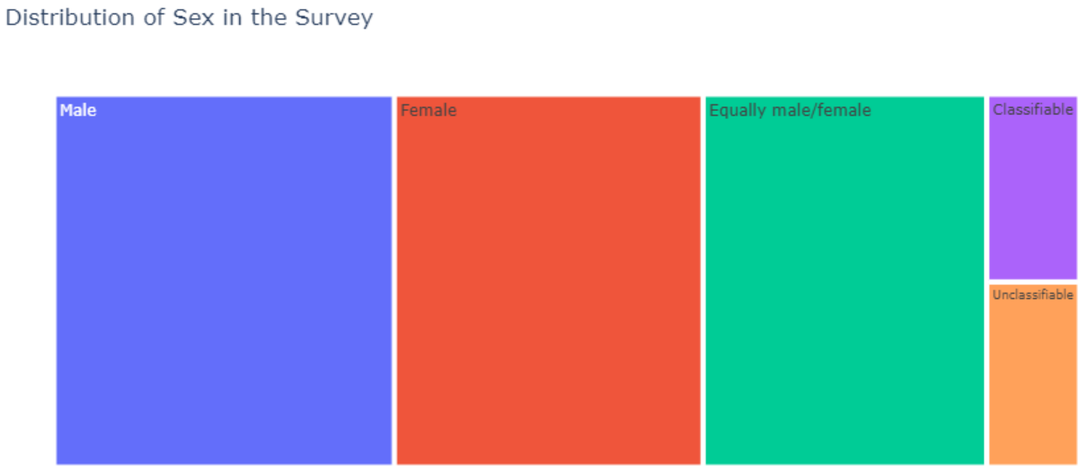


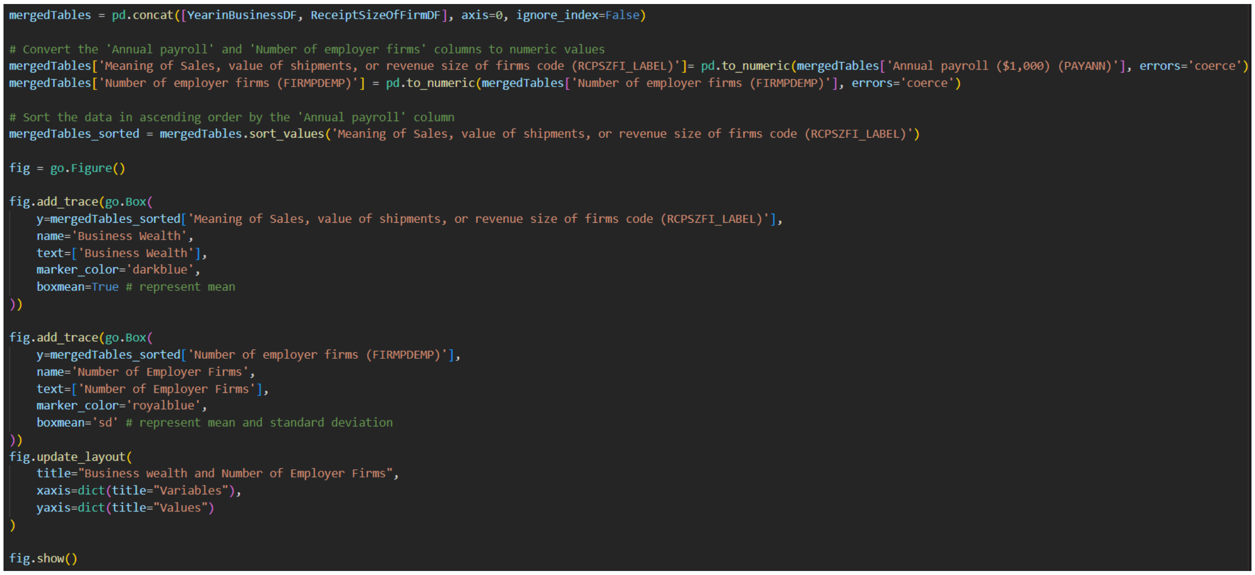
```
mergedTables['Annual payroll ($1,000) (PAYANN)'] = pd.to_numeric(mergedTables['Annual payroll ($1,000) (PAYANN)'], errors='coerce')

sns.scatterplot(x=mergedTables['Number of employer firms (FIRMPDEMP)'], y=mergedTables['Annual payroll ($1,000) (PAYANN)'])

plt.xlabel('Size of Business')
plt.ylabel('Wealth')
plt.title('Size of Business vs. Wealth (2019)')

plt.show()
```



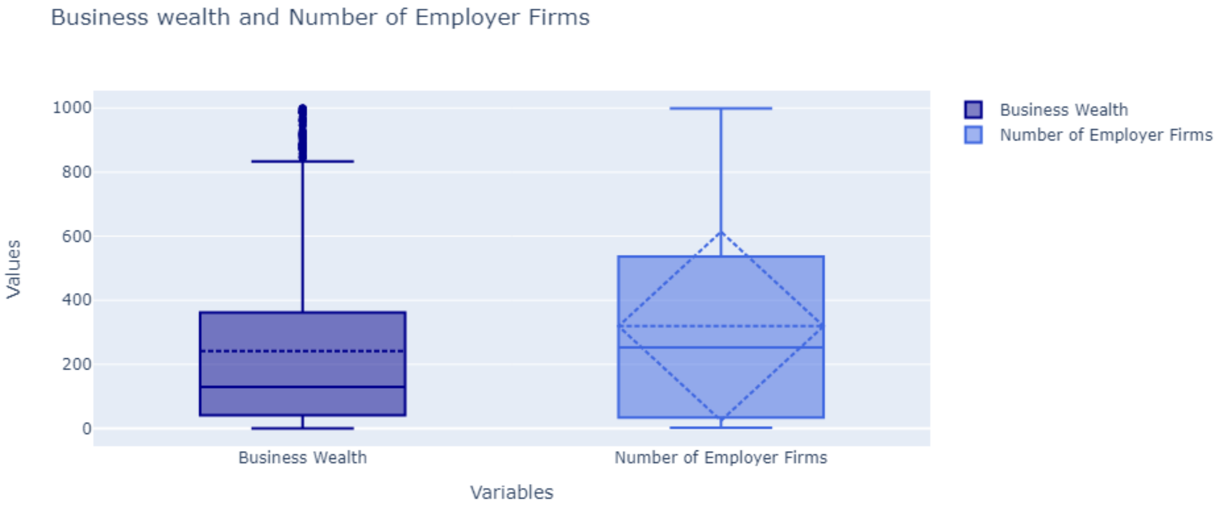


```
grouped_df = mergedTables[mergedTables['Meaning of Sex code (SEX_LABEL)'] != 'Total']['Meaning of Sex code (SEX_LABEL)'].value_counts().reset_index(name='Count')
grouped_df.rename(columns={'index': 'Sex'}, inplace=True)

fig = px.treemap(grouped_df, path=['Sex'], values='Count')
fig.update_layout(title='Distribution of Sex in the Survey')

fig.show()
```

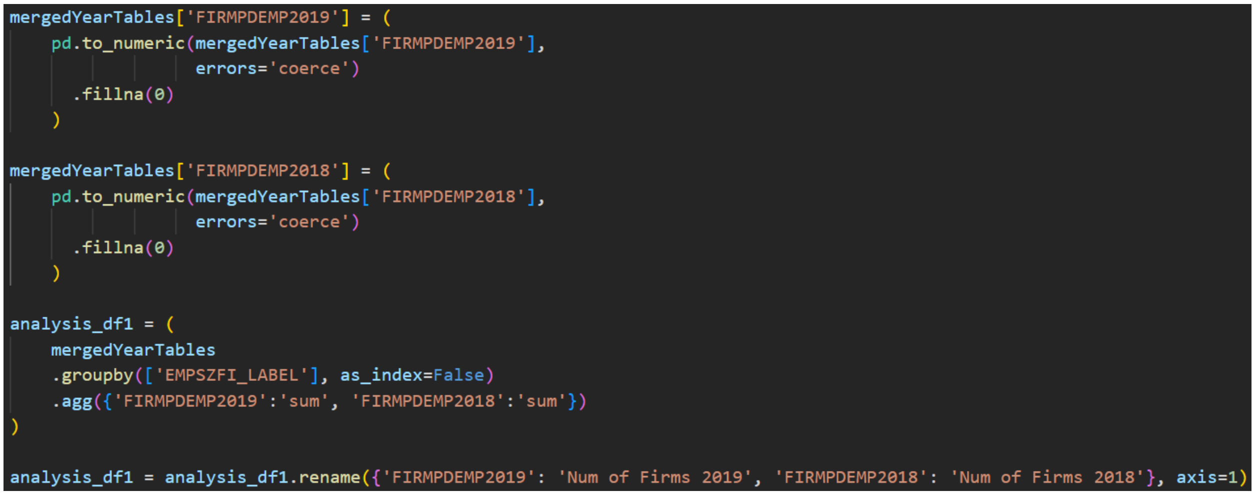




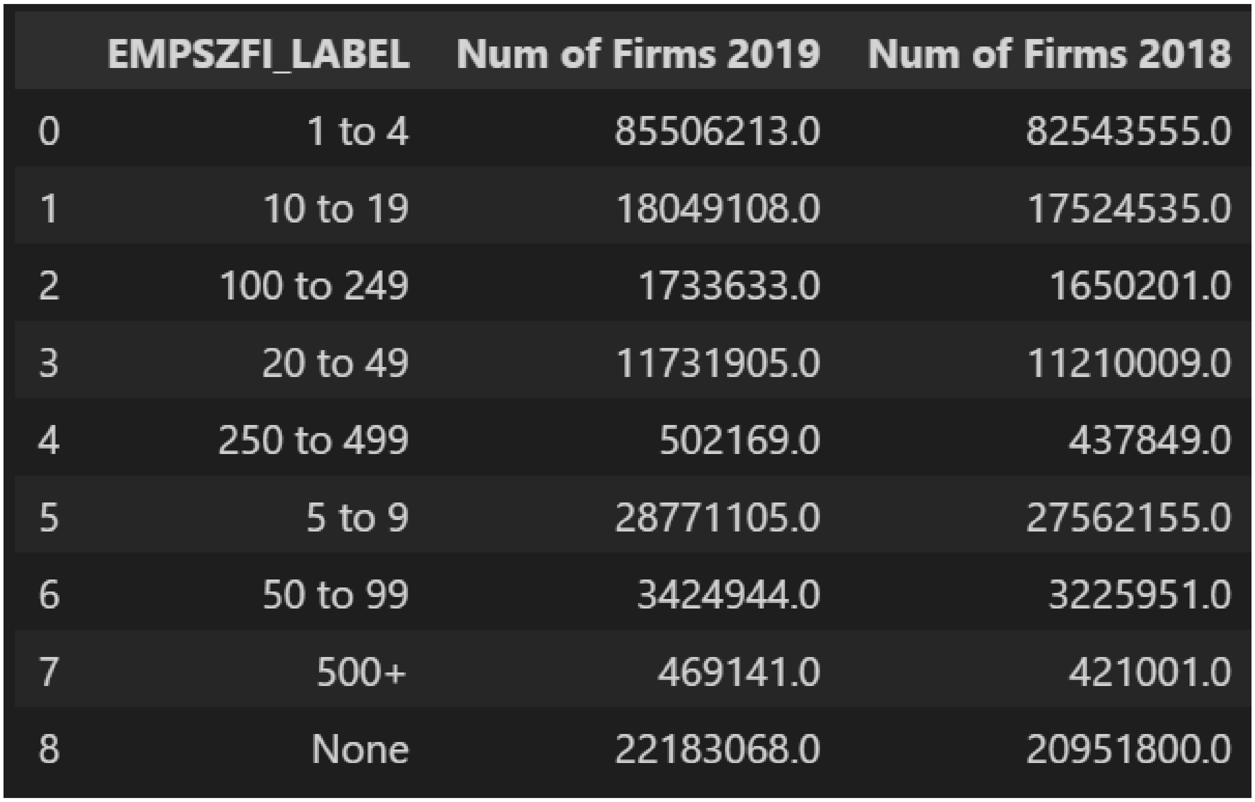
```
mergedTables = pd.concat([YearinBusinessDF, ReceiptSizeOfFirmDF], axis=0, ignore_index=False)

# Convert the 'Annual payroll' and 'Number of employer firms' columns to numeric values
mergedTables['Meaning of Sales, value of shipments, or revenue size of firms code (RCPSZFI_LABEL)']= pd.to_numeric(mergedTables['Annual payroll ($1,000) (PAYANN)'], errors='coerce')
mergedTables['Number of employer firms (FIRMPDEMP)'] = pd.to_numeric(mergedTables['Number of employer firms (FIRMPDEMP)'], errors='coerce')

# Sort the data in ascending order by the 'Annual payroll' column
mergedTables_sorted = mergedTables.sort_values('Meaning of Sales, value of shipments, or revenue size of firms code (RCPSZFI_LABEL)')

fig = go.Figure()

fig.add_trace(go.Box(
    y=mergedTables_sorted['Meaning of Sales, value of shipments, or revenue size of firms code (RCPSZFI_LABEL)'],
    name='Business Wealth',
    text=['Business Wealth'],
    marker_color='darkblue',
    boxmean=True # represent mean
))

fig.add_trace(go.Box(
    y=mergedTables_sorted['Number of employer firms (FIRMPDEMP)'],
    name='Number of Employer Firms',
    text=['Number of Employer Firms'],
    marker_color='royalblue',
    boxmean='sd' # represent mean and standard deviation
))
fig.update_layout(
    title="Business wealth and Number of Employer Firms",
    xaxis=dict(title="Variables"),
    yaxis=dict(title="Values")
)

fig.show()
```



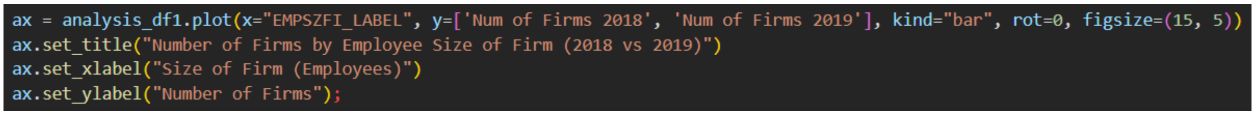
The following snippets of code explains how the API data was to create a visual:



```
mergedYearTables['FIRMPDEMP2019'] = (
    pd.to_numeric(mergedYearTables['FIRMPDEMP2019'],
                  errors='coerce')
      .fillna(0)
    )

mergedYearTables['FIRMPDEMP2018'] = (
    pd.to_numeric(mergedYearTables['FIRMPDEMP2018'],
                  errors='coerce')
      .fillna(0)
    )

analysis_df1 = (
    mergedYearTables
    .groupby(['EMPSZFI_LABEL'], as_index=False)
    .agg({'FIRMPDEMP2019':'sum', 'FIRMPDEMP2018':'sum'})
)

analysis_df1 = analysis_df1.rename({'FIRMPDEMP2019': 'Num of Firms 2019', 'FIRMPDEMP2018': 'Num of Firms 2018'}, axis=1)
```



	EMPSZFI_LABEL	Num of Firms 2019	Num of Firms 2018
0	1 to 4	85506213.0	82543555.0
1	10 to 19	18049108.0	17524535.0
2	100 to 249	1733633.0	1650201.0
3	20 to 49	11731905.0	11210009.0
4	250 to 499	502169.0	437849.0
5	5 to 9	28771105.0	27562155.0
6	50 to 99	3424944.0	3225951.0
7	500+	469141.0	421001.0
8	None	22183068.0	20951800.0



```
ax = analysis_df1.plot(x="EMPSZFI_LABEL", y=['Num of Firms 2018', 'Num of Firms 2019'], kind="bar", rot=0, figsize=(15, 5))
ax.set_title("Number of Firms by Employee Size of Firm (2018 vs 2019)")
ax.set_xlabel("Size of Firm (Employees)")
ax.set_ylabel("Number of Firms");
```

It can be summarized as follows:

1. Convert all values contained within desired columns from string to numeric via the pandas.to_numeric method
2. Create table to analyze where values are grouped by one (or multiple) desired columns and values of columns are aggregated
3. Rename columns as desired
4. Plot barchart

Data Sources

The datasets used for this project were accessed from April, 19th, 2023 to April, 23rd, 2023 and obtained from the following locations:

Bureau, U.S. Census. *Explore Census Data*, https://www.census.gov/data/tables/2019/econ/abs/2019-abs-company-summary.html

Bureau, U.S. Census. *Explore Census Data*, https://data.census.gov/table?tid=ABSCS2018.AB1800CSA02&hidePreview=true.

Bureau, U.S. Census. *Explore Census Data*, https://data.census.gov/table?tid=ABSCS2018.AB1800CSA03&hidePreview=true

Bureau, U.S. Census. *Explore Census Data*, https://data.census.gov/table?tid=ABSCS2018.AB1800CSA04&hidePreview=true