ETL Report

Increasing Audio Streaming Revenue

Land’o’datalakes: Vanessa Gleason, Alistair Marsden, Olivier Rochaix, Eduard Stalmakov

ETL Process 09/22/22

Introduction

What the project is about The United States Census Bureau’s Annual Business Survey (ABS) of 2019 (2019 ABS) “asked questions on selected economic and demographic characteristics for businesses and business owners by sex, ethnicity, race, and veteran status. Data were also collected on R&D (for businesses with 1-9 employees), innovation, technology, and financing. The 2019 ABS covers data for reference year 2018.” (United States Census Bureau, 2021)

We specifically investigated the datasets ‘Characteristics of Business Owners’ and ‘Technology Characteristics of Businesses‘ within the 2019 ABS.

The purpose of the final report generated using the data processed in this ETL report will be to analyze the typical characteristics of a modern day business owner in the United States. It intends to reveal how much of an emphasis the modern day businessman puts on work life balance. It will analyze business owner ages and whether that affects technologies sold by firms. Additionally, it will assess what is the sex of the modern day business owner. Lastly, we will describe the motivations of business owners for entering and staying in their positions, as well as the levels of usage and impact of technology on employment numbers in various industries.

In this ETL Report, the data sources will be identified in the Extraction section. Then, the required transformations will be identified in the Transforming section.

Data Sources

In APA format United States Census Bureau. (2021, OCTOBER 28). *2019 Annual Business Survey (ABS) APIs: Characteristics of Business Owners*. Developers. Retrieved September 2, 2022, from https://api.census.gov/data/2018/abscbo.html

United States Census Bureau. (2021, OCTOBER 28). *2019 Annual Business Survey (ABS) APIs: Technology Characteristics of Businesses*. Retrieved SEPTEMBER 02, 2022, from https://api.census.gov/data/2018/abstcb.html

Extraction

All data sets were accessed September 2, 2022. (United States Census Bureau, 2021)

1. **Main File**
    a. Group members will already have their Census Data API Key saved to a config.py file in the same folder as the new file.

    b. Open a new Blank Jupyter Notebook file in Visual Studio Code

    c. Input the following into the first cell:

```
from config import api_key
import pandas as pd
import requests
import matplotlib.pyplot as plt
import seaborn as sns
```

    d. To Access the Business Owner data (United States Census Bureau, 2021):

        i. In a new cell, enter:

```
url =
f'https://api.census.gov/data/2018/abscbo?get=GEO_ID,NAME,NAICS2017,NAICS2017_LABEL,
OWNER_SEX,OWNER_SEX_LABEL,OWNER_ETH,OWNER_ETH_LABEL,OWNER_RACE,OWNER_RACE_LABEL,OWNE
```

```
R_VET,OWNER_VET_LABEL,QDESC,QDESC_LABEL,OWNCHAR,OWNCHAR_LABEL,OWNPDEMP,OWNPDEMP_PCT&
for=us:*&key={api_key}'
res = requests.get(url).json()
bo_df = pd.DataFrame(res)
bo_df.columns = bo_df.iloc[0]
bo_df.drop(bo_df.index[0], inplace=True)
bo_df
```

ii. The ‘url’ line returns the following columns:

1. GEO_ID
2. NAME
3. NAICS2017
4. NAICS2017_LABEL
5. OWNER_SEX
6. OWNER_SEX_LABEL
7. OWNER_ETH
8. OWNER_ETH_LABEL
9. OWNER_RACE
10. OWNER_RACE_LABEL
11. OWNER_VET
12. OWNER_VET_LABEL
13. QDESC
14. QDESC_LABEL
15. OWNCHAR
16. OWNCHAR_LABEL
17. OWNPDEMP
18. OWNPDEMP_PCT

19. us

iii. The ‘res’ line sends the request to retrieve the desired data

iv. The ‘bo_df’ line creates the dataframe with the desired data

v. The ‘bo_df.columns’ line and the following line, beginning ‘bo_df.drop’ line work together to move the column labels from the first row of data to the headers

vi. Lastly, the ‘bo_df’ line allows us to view the data as a table

vii. This returns a table with 104622 rows and 19 columns

e. To Access the “Technology Characteristics of Business” data (United States Census Bureau, 2021):

i. In a new cell, enter:

```
url =
f'https://api.census.gov/data/2018/abscbo?get=GEO_ID,NAME,NAICS2017,NAICS2017_LABEL,
OWNER_SEX,OWNER_SEX_LABEL,OWNER_ETH,OWNER_ETH_LABEL,OWNER_RACE,OWNER_RACE_LABEL,OWNE
R_VET,OWNER_VET_LABEL,QDESC,QDESC_LABEL,OWNCHAR,OWNCHAR_LABEL,OWNPDEMP,OWNPDEMP_PCT&
for=us:*&key={api_key}'
res = requests.get(url).json()
bo_df = pd.DataFrame(res)
bo_df.columns = bo_df.iloc[0]
bo_df.drop(bo_df.index[0], inplace=True)
bo_df
```

ii. The ‘url’ line returned the following columns:

1. GEO_ID
2. NAME
3. NAICS2017
4. NAICS2017_LABEL
5. NSFSZFI
6. NSFSZFI_LABEL
7. FACTORS_P

8. FACTORS_P_LABEL
9. FACTORS_U
10. FACTORS_U_LABEL
11. IMPACTWF_P,
12. IMPACTWF_P_LABEL
13. IMPACTWF_U
14. IMPACTWK_U_LABEL
15. MOTPRODTECH
16. MOTPRODTECH_LABEL
17. MOTUSETECH
18. MOTUSETECH_LABEL
19. TECHSELL
20. TECHSELL_LABEL
21. TECHUSE
22. TECHUSE_LABEL
23. us

iii. The ‘res’ line sends the request to retrieve the desired data

iv. The ‘tech_df’ line creates the dataframe with the desired data

v. The ‘tech_df.columns’ and the following line, beginning ‘tech_df.drop’ work together to move the column labels from the first row of data to the headers

vi. Lastly, ‘tech_df’ allows us to view the data as a table

vii. This returns a table with 12480 rows and 27 columns

f. This concludes the importation of the data. Next, each member of the team transforms this data to answer their specific question(s) concerning our overarching question, “What does the modern day business owner look like?

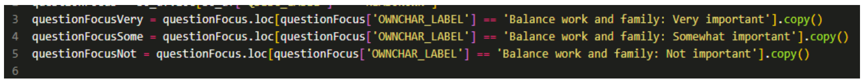
2. Transformations

a. In Robert's Data:

i. To create visualization 1

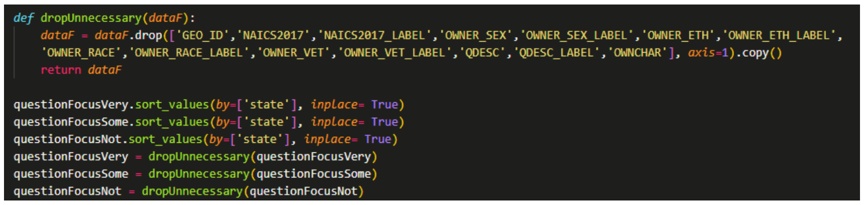
1. The first key step was to pull out the specific question from the larger api call and made sure that I had my api call drilled down to the state level.. This is done through the relatively traditional .loc[] call on the larger dataframe looking for the qdesc_label, reasonown. This focuses the dataframe down to just the questions and responses asked to business owners.

2. Then I split this new table into three separate tables. One table focused on responses with Very Important as their response, one as Somewhat Important, and the last one as Not Important.



```python
questionFocusVery = questionFocus.loc[questionFocus['OWNCHAR_LABEL'] == 'Balance work and family: Very important'].copy()
questionFocusSome = questionFocus.loc[questionFocus['OWNCHAR_LABEL'] == 'Balance work and family: Somewhat important'].copy()
questionFocusNot = questionFocus.loc[questionFocus['OWNCHAR_LABEL'] == 'Balance work and family: Not important'].copy()
```

3. Then I sorted each table by the state id as it is a common integer value that only appears once per table. Before dropping what I deemed to be all unnecessary columns, either because they don't have any change in values, or they weren't something that we were going to be inspecting.



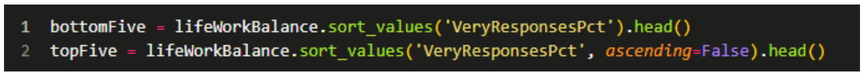
```python
def dropUnnecessary(dataF):
    dataF = dataF.drop(['GEO_ID','NAICS2017','NAICS2017_LABEL','OWNER_SEX','OWNER_SEX_LABEL','OWNER_ETH','OWNER_ETH_LABEL',
    'OWNER_RACE','OWNER_RACE_LABEL','OWNER_VET','OWNER_VET_LABEL','QDESC','QDESC_LABEL','OWNCHAR'], axis=1).copy()
    return dataF

questionFocusVery.sort_values(by=['state'], inplace= True)
questionFocusSome.sort_values(by=['state'], inplace= True)
questionFocusNot.sort_values(by=['state'], inplace= True)
questionFocusVery = dropUnnecessary(questionFocusVery)
questionFocusSome = dropUnnecessary(questionFocusSome)
questionFocusNot = dropUnnecessary(questionFocusNot)
```

4. Next, I renamed the columns OWNPDEMP and OWNPDEMP_PCT to more accurately represent what is in each column. Having the level of response and either responses or responsespct depending on the corresponding column.

5. Finally I merged the three tables together on the state value and made sure to remove any extra name or ownchar_label columns, and lastly I was sure to cast the VeryResponsePct as a float so that it could be put into the map.

ii. To create visualization 2

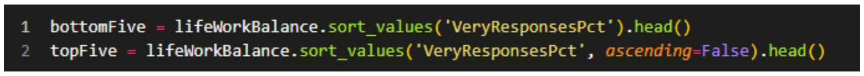
1. I sorted the values by VeryResponsesPct and took the top five results using the .head() function.



```python
bottomFive = lifeWorkBalance.sort_values('VeryResponsesPct').head()
topFive = lifeWorkBalance.sort_values('VeryResponsesPct', ascending=False).head()
```

iii. To create visualization 3

1. I sorted the values by VeryResponsesPct with ascending equal to False, and took the top five results using the .head() function.



```python
bottomFive = lifeWorkBalance.sort_values('VeryResponsesPct').head()
topFive = lifeWorkBalance.sort_values('VeryResponsesPct', ascending=False).head()
```

b. In Technology vs Owner Age (Olivier’s) Data:

1. Once the group made the broad query from the API calls, we separated and did research on filtered versions of the DataFrames. I had the initial question of "What kind of impact did the Age of Business Owners have on Technologies Sold?" For my first visualization, I wanted to see the distribution of owners in the US by their age:

i. Because we took such a broad grab of the dataset originally, I had to filter down the data by relevant columns in order to get realistic numbers. I filtered the `bo_df` DataFrame's ['QDESC'] column by the code 'O09' in order to only get rows that were related to the question "What was the age of Owner 1 (2, 3, or 4) as of December 31, 2018?". I then used `.copy()` to copy the resulting DataFrame and stored it in the variable `age`.

`age = bo_df[bo_df['QDESC'] == 'O09'].copy()`

ii. I filtered the new `age` DataFrame's ['NAICS2017_LABEL'] column by 'Total for all sectors'. I also filtered the ['OWNER_SEX_LABEL'], ['OWNER_ETH_LABEL'], ['OWNER_RACE_LABEL'], and ['OWNER_VET_LABEL'] columns by 'All owners of respondent firms'. This filtering allowed for following aggregations to produce realistic numbers.

```
age = age[age['NAICS2017_LABEL'] == 'Total for all sectors']

age = age[age['OWNER_SEX_LABEL'] == 'All owners of respondent firms']

age = age[age['OWNER_ETH_LABEL'] == 'All owners of respondent firms']

age = age[age['OWNER_RACE_LABEL'] == 'All owners of respondent firms']
age = age[age['OWNER_VET_LABEL'] == 'All owners of respondent firms']
```

iii. I cast the `age` DataFrame's ['OWNPDEMP'] column as an integer so that I would be able to aggregate on that column.

`age['OWNPDEMP'] = age['OWNPDEMP'].astype(int)`

iv. I used the `.groupby()` method and passed in 'OWNCHAR_LABEL' in order to get a sum of the ['OWNPDEMP'] column's values for each answer to the question asked.

`age = age.groupby('OWNCHAR_LABEL')['OWNPDEMP'].sum()`

v. I dropped the 'Total reporting' and 'No response' answers, and renamed the 'Under 25' answer to '24 or under' in order to facilitate ordering the values.

```
age.drop([age.index[5],age.index[6]], inplace=True)

age.rename({'Under 25':'24 or under'}, inplace=True)

age.sort_index(ascending=True, inplace=True)
```

vi. This resulted in a Series that had 6 age groups and a summation of the number of respondents who answered the survey with those age groups. I then used this Series to build a histogram.

2. After I made my visualization for the distribution of owners by age groups, I looked at the distribution of firms by the technologies that they sold:

i. I filtered the `tech_df` DataFrame's ['NAICS2017_LABEL'] column by the value 'Total for all sectors'. I then copied the DataFrame using `.copy()` and saved it to the variable `tech_sell`.

`tech_sell = tech_df[tech_df['NAICS2017_LABEL'] == 'Total for all sectors'].copy()`

ii. I filtered the new `tech_sell` DataFrame's ['NSFSZFI_LABEL'], ['FACTORS_P_LABEL'], ['FACTORS_U_LABEL'], ['IMPACTWF_P_LABEL'], ['IMPACTWF_U_LABEL'], ['IMPACTWK_P_LABEL'], ['IMPACTWK_U_LABEL'], ['MOTPRODTECH_LABEL'], ['MOTUSETECH_LABEL'], and ['TECHUSE_LABEL'] columns by the value 'All firms' in order to eliminate any value bloat in the following aggregation.

```
tech_sell = tech_sell[tech_sell['NSFSZFI_LABEL'] == 'All firms']
tech_sell = tech_sell[tech_sell['FACTORS_P_LABEL'] == 'All firms']

tech_sell = tech_sell[tech_sell['FACTORS_U_LABEL'] == 'All firms']

tech_sell = tech_sell[tech_sell['IMPACTWF_P_LABEL'] == 'All firms']

tech_sell = tech_sell[tech_sell['IMPACTWF_U_LABEL'] == 'All firms']

tech_sell = tech_sell[tech_sell['IMPACTWK_P_LABEL'] == 'All firms']

tech_sell = tech_sell[tech_sell['IMPACTWK_U_LABEL'] == 'All firms']

tech_sell = tech_sell[tech_sell['MOTPRODTECH_LABEL'] == 'All firms']

tech_sell = tech_sell[tech_sell['MOTUSETECH_LABEL'] == 'All firms']

tech_sell = tech_sell[tech_sell['TECHUSE_LABEL'] == 'All firms']
```

iii. I cast the `tech_sell` DataFrame's ['FIRMPDEMP'] column as an integer.

`tech_sell['FIRMPDEMP'] = tech_sell['FIRMPDEMP'].astype(int)`

iv. I used `.groupby()`, passing in 'TECHSELL_LABEL' in order to get a sum of the ['FIRMPDEMP'] column's values for each answer the question asked. I was interested in the ['TECH_SELL'] column because the answers to the question "During the three years 2016 to 2018, did this business sell the following technologies or goods or services that included the following technologies?" were logged in it.

`tech_sell = tech_sell.groupby('TECHSELL_LABEL')['FIRMPDEMP'].sum()`

v. I was only interested in answers that indicated that firms sold tech, so I dropped any value that wasn't tagged with 'Yes'.

```
tech_sell.drop([tech_sell.index[0], tech_sell.index[1], tech_sell.index[2], tech_sell.index[4], tech_sell.index[5], tech_sell.index[6]], inplace=True)

tech_sell.drop([tech_sell.index[2], tech_sell.index[3], tech_sell.index[4], tech_sell.index[6], tech_sell.index[7], tech_sell.index[8]], inplace=True)

tech_sell.drop([tech_sell.index[4], tech_sell.index[5], tech_sell.index[6]], inplace=True)
```

vi. I sorted the aggregated values in ascending order.

`tech_sell.sort_values(ascending=True, inplace=True)`

vii. This resulted in a Series with five 'Yes' answers and a summation of the number of firms that chose those answers in the ABS survey. I used this Series to make a histogram.

3. After looking at the two datasets individually and making histograms, I worked to merge the two tables. As there were no immediately obvious columns to merge the datasets on, I had to transform the datasets to accomplish a proper merge:

a. I started by making a DataFrame containing unique instances of the `bo_df` DataFrame's ['NAICS2017_LABEL'] column's values and the summation of owners that answered 'Under 25' to the age question;

i. I filtered the `bo_df` DataFrame's ['OWNCHAR_LABEL'] column by the value 'Under 25'. I then copied it and stored it in the variable `aged_24_under`.

`aged_24_under = bo_df[bo_df['OWNCHAR_LABEL'] == 'Under 25'].copy()`

ii. I filtered the ['OWNER_SEX_LABEL'], ['OWNER_ETH_LABEL'], ['OWNER_RACE_LABEL'], and ['OWNER_VET_LABEL'] columns by 'All owners of respondent firms'. This filtering allowed for the following aggregation to produce realistic numbers.

```
aged_24_under = aged_24_under[aged_24_under['OWNER_SEX_LABEL'] == 'All owners of respondent firms']

aged_24_under = aged_24_under[aged_24_under['OWNER_ETH_LABEL'] == 'All owners of respondent firms']

aged_24_under = aged_24_under[aged_24_under['OWNER_RACE_LABEL'] == 'All owners of respondent firms']

aged_24_under = aged_24_under[aged_24_under['OWNER_VET_LABEL'] == 'All owners of respondent firms']
```

iii. I cast ['OWNPDEMP'] as an integer.

`aged_24_under['OWNPDEMP'] = aged_24_under['OWNPDEMP'].astype(int)`

iv. I grouped on 'NAICS2017_LABEL' to get a summation of the ['OWNPDEMP'] values.

`aged_24_under_by_industry = aged_24_under.groupby('NAICS2017_LABEL')['OWNPDEMP'].sum()`

v. I reset the resulting Series' index to make turn it back into a DataFrame.

`aged_24_under_by_industry = aged_24_under_by_industry.reset_index()`

vi. I renamed the new ['OWNPDEMP'] column to 'Owners Aged 24 or Under'.

`aged_24_under_by_industry.rename(columns={'OWNPDEMP':'Owners Aged 24 or Under'}, inplace=True)`

vii. This resulted in a DataFrame with a ['NAICS2017_LABEL'] and ['Owners Aged 24 or Under'] column and 21 rows.

b. I used the steps above to make dataframes for each of the answer categories I had previously looked at. Specifically, I made one for '25 to 34', '35 to 44', '45 to 54', '55 to 64', and '65 or over'.

```

aged_QUESTION_ANSWER_HERE = bo_df[bo_df['OWNCHAR_LABEL'] ==
'QUESTION_ANSWER_HERE'].copy()

aged_QUESTION_ANSWER_HERE =
aged_QUESTION_ANSWER_HERE[aged_QUESTION_ANSWER_HERE['OWNER_SEX_LABEL'] == 'All
owners of respondent firms']

aged_QUESTION_ANSWER_HERE =
aged_QUESTION_ANSWER_HERE[aged_QUESTION_ANSWER_HERE['OWNER_ETH_LABEL'] ==
'All owners of respondent firms']

aged_QUESTION_ANSWER_HERE =
aged_QUESTION_ANSWER_HERE[aged_QUESTION_ANSWER_HERE['OWNER_RACE_LABEL'] ==
'All owners of respondent firms']

aged_QUESTION_ANSWER_HERE =
aged_QUESTION_ANSWER_HERE[aged_QUESTION_ANSWER_HERE['OWNER_VET_LABEL'] == 'All
owners of respondent firms']

aged_QUESTION_ANSWER_HERE['OWNPDEMP'] =
aged_QUESTION_ANSWER_HERE['OWNPDEMP'].astype(int)

aged_QUESTION_ANSWER_HERE_by_industry =
aged_QUESTION_ANSWER_HERE.groupby('NAICS2017_LABEL')['OWNPDEMP'].sum()

aged_QUESTION_ANSWER_HERE_by_industry =
aged_QUESTION_ANSWER_HERE_by_industry.reset_index()
```

aged_QUESTION_ANSWER_HERE_by_industry.rename(columns={'OWNPDEMP':'Owners Aged QUESTION_ANSWER_HERE'}, inplace=True)

```

c. I then merged the DataFrames I created in steps 4a and 4b:

i. I merged the `aged_24_under_by_industry` DataFrame on the `aged_25_34_by _industry` DataFrame. I used an 'inner' join on the 'NAICS2017_LABEL' column. I stored this new DataFrame in the variable `age_group_by_industry`.

`age_group_by_industry = aged_24_under_by_industry.merge(aged_25_34_by_industry, how='inner', on='NAICS2017_LABEL')`

ii. I merged the rest of the DataFrames onto the new `age_group_by_industry` DataFrame.

```
age_group_by_industry = age_group_by_industry.merge(aged_35_44_by_industry, how='inner', on='NAICS2017_LABEL')

age_group_by_industry = age_group_by_industry.merge(aged_45_54_by_industry, how='inner', on='NAICS2017_LABEL')

age_group_by_industry = age_group_by_industry.merge(aged_55_64_by_industry, how='inner', on='NAICS2017_LABEL')

age_group_by_industry = age_group_by_industry.merge(aged_65_over_by_industry, how='inner', on='NAICS2017_LABEL')
```

iii. I dropped the 'Total for all sectors' and 'No specific industry' answers from the merged dataset. These values did not provide any important information to me.

`age_group_by_industry.drop([age_group_by_industry.index[8], age_group_by_industry.index[17]], inplace=True)`

d. Once the grouping and merging was finished for the "Characteristics of Business Owners" dataset, I started on the "Technology Characteristics of Businesses" dataset. I started by making a DataFrame containing unique instances of the `tech_df` DataFrame's ['NAICS2017_LABEL'] column's values and the summation of firms that answered 'Robotics: Yes' to the tech sold question;
```

i. I filtered the `tech_df` DataFrame's ['TECHSELL_LABEL'] column by the value 'Robotics: Yes'. I then copied the new DataFrame and stored it in a variable called `tech_sell_robots`.

`tech_sell_robots = tech_df[tech_df['TECHSELL_LABEL'] == 'Robotics: Yes'].copy()`

ii. I filtered the ['NSFSZFI_LABEL'], ['FACTORS_P_LABEL'], ['FACTORS_U_LABEL'], ['IMPACTWF_P_LABEL'], ['IMPACTWF_U_LABEL'], ['IMPACTWK_P_LABEL'], ['IMPACTWK_U_LABEL'], ['MOTPRODTECH_LABEL'], ['MOTUSETECH_LABEL'], and ['TECHUSE_LABEL'] columns by the value 'All firms'.

```

tech_sell_robots = tech_sell_robots[tech_sell_robots['NSFSZFI_LABEL'] == 'All firms']

tech_sell_robots = tech_sell_robots[tech_sell_robots['FACTORS_P_LABEL'] == 'All firms']

tech_sell_robots = tech_sell_robots[tech_sell_robots['FACTORS_U_LABEL'] == 'All firms']

tech_sell_robots = tech_sell_robots[tech_sell_robots['IMPACTWF_P_LABEL'] == 'All firms']

tech_sell_robots = tech_sell_robots[tech_sell_robots['IMPACTWF_U_LABEL'] == 'All firms']

tech_sell_robots = tech_sell_robots[tech_sell_robots['IMPACTWK_P_LABEL'] == 'All firms']

tech_sell_robots = tech_sell_robots[tech_sell_robots['IMPACTWK_U_LABEL'] == 'All firms']

tech_sell_robots = tech_sell_robots[tech_sell_robots['MOTPRODTECH_LABEL'] == 'All firms']

tech_sell_robots = tech_sell_robots[tech_sell_robots['MOTUSETECH_LABEL'] == 'All firms']

tech_sell_robots = tech_sell_robots[tech_sell_robots['TECHUSE_LABEL'] == 'All firms']

```

iii. I cast the ['FIRMPDEMP'] as an integer.

`tech_sell_robots['FIRMPDEMP'] = tech_sell_robots['FIRMPDEMP'].astype(int)`

iv. I grouped on 'NAICS2017_LABEL' to get a summation of the ['FIRMPDEMP'] values per industry.

`tech_sell_robots = tech_sell_robots.groupby('NAICS2017_LABEL')['FIRMPDEMP'].sum()`

v. I reset the index to turn the resulting Series into a DataFrame.

`tech_sell_robots = tech_sell_robots.reset_index()`

vi. I renamed the ['FIRMPDEMP'] column in the new DataFrame to ['Firms Selling Robotics Tech'].

`tech_sell_robots.rename(columns={'FIRMPDEMP':'Firms Selling Robotics Tech'}, inplace=True)`

e. I used the steps above to make dataframes for each of the answer categories I had previously looked at. Specifically, I made one for 'Artificial Intelligence: Yes', 'Specialized Equipment: Yes', 'Cloud-Based: Yes', and 'Specialized Software: Yes'.

```
tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_df[tech_df['TECHSELL_LABEL'] == 'QUESTION_ANSWER_HERE'].copy()

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE[tech_sell_QUESTION_ANSWER_ABBRV_HERE['NSFSZFI_LABEL'] == 'All firms']

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE[tech_sell_QUESTION_ANSWER_ABBRV_HERE['FACTORS_P_LABEL'] == 'All firms']

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE[tech_sell_QUESTION_ANSWER_ABBRV_HERE['FACTORS_U_LABEL'] == 'All firms']

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE[tech_sell_QUESTION_ANSWER_ABBRV_HERE['IMPACTWF_P_LABEL'] == 'All firms']

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE[tech_sell_QUESTION_ANSWER_ABBRV_HERE['IMPACTWF_U_LABEL'] == 'All firms']

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE[tech_sell_QUESTION_ANSWER_ABBRV_HERE['IMPACTWK_P_LABEL'] == 'All firms']
tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE[tech_sell_QUESTION_ANSWER_ABBRV_HERE['IMPACTWK_U_LABEL'] == 'All firms']

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE[tech_sell_QUESTION_ANSWER_ABBRV_HERE['MOTPRODTECH_LABEL'] == 'All firms']

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE[tech_sell_QUESTION_ANSWER_ABBRV_HERE['MOTUSETECH_LABEL'] == 'All firms']

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE[tech_sell_QUESTION_ANSWER_ABBRV_HERE['TECHUSE_LABEL'] == 'All firms']

tech_sell_QUESTION_ANSWER_ABBRV_HERE['FIRMPDEMP'] = tech_sell_QUESTION_ANSWER_ABBRV_HERE['FIRMPDEMP'].astype(int)

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE.groupby('NAICS2017_LABEL')['FIRMPDEMP'].sum()

tech_sell_QUESTION_ANSWER_ABBRV_HERE = tech_sell_QUESTION_ANSWER_ABBRV_HERE.reset_index()

tech_sell_QUESTION_ANSWER_ABBRV_HERE.rename(columns={'FIRMPDEMP':'Firms Selling Specialized Software'}, inplace=True)
```

f. I then merged the DataFrames I created in steps 4d and 4e:

i. I merged the `tech_sell_robots` DataFrame onto the `tech_sell_ai` DataFrame. I used an 'inner' join on the 'NAICS2017_LABEL' column. I stored this new DataFrame in the variable `tech_sell_by_industry`.

`tech_sell_by_industry = tech_sell_robots.merge(tech_sell_ai, how='inner', on='NAICS2017_LABEL')`

ii. I merged the rest of the DataFrames to `tech_sell_by_industry`.

```
tech_sell_by_industry = tech_sell_by_industry.merge(tech_sell_speceq, how='inner', on='NAICS2017_LABEL')
```

tech_sell_by_industry = tech_sell_by_industry.merge(tech_sell_cloud, how='inner', on='NAICS2017_LABEL')

tech_sell_by_industry = tech_sell_by_industry.merge(tech_sell_specsw, how='inner', on='NAICS2017_LABEL')

```

iii. I dropped the 'Total for all sectors' and 'No specific industry' answers from the merged dataset. These values did not provide any important information to me.

`tech_sell_by_industry.drop([tech_sell_by_industry.index[8], tech_sell_by_industry.index[17]], inplace=True)`

g. Once I was finished grouping and merging the data that I wanted for the "Technology Characteristics of Businesses" dataset, I merged the `age_group_by_industry` and `tech_sell_by_industry` DataFrames:

i. I merged the `age_group_by_industry` DataFrame on the `tech_sell_by_industry` DataFrame. I used an 'inner' join on the 'NAICS2017_LABEL' column. I stored the new DataFrame in a variable called `merged_cbo_tcb`.

`merged_cbo_tcb = age_group_by_industry.merge(tech_sell_by_industry, how='inner', on='NAICS2017_LABEL')`

ii. I renamed the 'NAICS2017_LABEL' column to 'Industry'.

`merged_cbo_tcb.rename(columns={'NAICS2017_LABEL':'Industry'}, inplace=True)`

iii. This resulted in a DataFrame that contained valid data from both datasets. I used this merged DataFrame to create a correlation matrix, then I used findings gleaned from the correlation matrix to make two scatter plots.

## 4. I then dropped some values to make zoomed in versions of the scatter plots that I had:

a. For my first scatter plot I dropped values that weren't clumped in the lower left hand corner.

```

zoom_merged_cbo_tcb = merged_cbo_tcb.drop([merged_cbo_tcb.index[0], merged_cbo_tcb.index[1], merged_cbo_tcb.index[4], merged_cbo_tcb.index[7], merged_cbo_tcb.index[12], merged_cbo_tcb.index[13], merged_cbo_tcb.index[15]])

```

b. I did the same for the second one, with slightly different values.
```

```

zoom_merged_cbo_tcb = merged_cbo_tcb.drop([merged_cbo_tcb.index[0],
merged_cbo_tcb.index[1], merged_cbo_tcb.index[4], merged_cbo_tcb.index[7],
merged_cbo_tcb.index[10], merged_cbo_tcb.index[12], merged_cbo_tcb.index[13],
merged_cbo_tcb.index[15], merged_cbo_tcb.index[18]]

```

c. To create Industry By Sex Data:

i. First, to eliminate duplicate data being reported when assessing the amount of business owners of each sex, I needed to elminate sex being reported in other columns that would add to my totals. To accomplish this I made a copy of the data as 'bo_dfREV' and then reduced columns that may represent sex data to only represent "All owners of respondent firms" or "Total reporting", see the following code:

```
bo_dfREV = bo_df.copy()
bo_dfREV = bo_dfREV[bo_dfREV['OWNER_ETH_LABEL'] == 'All owners of respondent firms']
bo_dfREV = bo_dfREV[bo_dfREV['OWNER_RACE_LABEL'] == 'All owners of respondent 
firms']
bo_dfREV = bo_dfREV[bo_dfREV['OWNER_VET_LABEL'] == 'All owners of respondent firms']
bo_dfREV = bo_dfREV[bo_dfREV['QDESC_LABEL'] == 'YRACQBUS']
bo_dfREV = bo_dfREV[bo_dfREV['OWNCHAR_LABEL'] == 'Total reporting']
```

ii. From our 'Characteristics of Business Owners (CBO) Dataset' I first defined the columns to include in my research. I created a new dataset called NAICS_bo.

iii. The attributes I included are:

1. NAICS2017 which is the code for North American Industry Classification System code
2. NACIS2017_LABEL which defines the label for the NACIS code in NACIS2017
3. OWNER_SEX which codes a business owner sex
4. OWNER_SEX_LABEL defines the label for the code given in OWNER_SEX

iv. Here is the code:

```
NAICS_bo = bo_dfREV[['NAICS2017','NAICS2017_LABEL', 'OWNER_SEX','OWNER_SEX_LABEL',
'OWNPDEMP']].copy()

NAICS_bo = pd.DataFrame(NAICS_bo)
```

v. Then, I reset the index. Here is the code:

```
NAICS_bo.reset_index()
```

vi. Some of the NACIS2017_LABELs were very long. To make them appear better on a visual, I shortened the longer ones. Here is the code:

```
NAICS_bo.loc[NAICS_bo['NAICS2017_LABEL'] == 'Administrative and support and waste management and remediation services', 'NAICS2017_LABEL'] = 'Admin/Support/Waste mgmt./Remediation svcs.'
NAICS_bo.loc[NAICS_bo['NAICS2017_LABEL'] == 'Other services (except public administration)', 'NAICS2017_LABEL'] = 'Other svcs (not public admin)'
NAICS_bo.loc[NAICS_bo['NAICS2017_LABEL'] == 'Professional, scientific, and technical services', 'NAICS2017_LABEL'] = 'Pro, sci, and tech svcs'
NAICS_bo.loc[NAICS_bo['NAICS2017_LABEL'] == 'Mining, quarrying, and oil and gas extraction', 'NAICS2017_LABEL'] = 'Mining/Quarrying/Oil and gas extraction'
NAICS_bo.loc[NAICS_bo['NAICS2017_LABEL'] == 'Management of companies and enterprises', 'NAICS2017_LABEL'] = 'Mgmt. of companies and enterprises'
```

vii. Next I removed the NACIS2017 code 00 because that sums all the industries and would not be meaningful in visuals concerned about top industries. Here is the code:

```
NAICS_bo = NAICS_bo.loc[NAICS_bo["NAICS2017"] != '00']
```

viii. At this point the data is workable and I can investigate the specific questions I have within it.

ix. To transform 'Top 5 Industries of Female Business Owners' data

1. First make a specialized copy of my dataset ‘NACIS_bo’ that only shows the rows where the data concerns females, where OWNER_SEX = 002. Here is the code:

```
female = NAICS_bo[NAICS_bo['OWNER_SEX'] == '002'].copy()
```

2. Then, convert the ‘OWNPDEMP’ (Number of owners of respondent employer firms) column to an integer. Here is the code:

```
female['OWNPDEMP'] = female['OWNPDEMP'].astype(int)
```

3. #only show the rows NAICS label, owner sex label, and # of responding firms

4. Then, only show the columns 'NAICS2017_LABEL', 'OWNER_SEX', 'OWNPDEMP'

```
female = female[['NAICS2017_LABEL', 'OWNER_SEX', 'OWNPDEMP']]
```

5. Next, I renamed the columns 'NAICS2017_LABEL' and 'OWNPDEMP' ‘Industry’ and 'Female Business Owners', respectively to make them easier to interpret. Here is the code:

```
rename OWNPDEMP to Female Business Owners
female.rename(columns = {'OWNPDEMP':'Female Business Owners',
'NAICS2017_LABEL':'Industry'}, inplace = True)
#group that by NAICS code
```

6. Then to get the count of how many female business owners are in each industry, I grouped by industry. Here is the code:

```
female = female.groupby('Industry')['Female Business Owners'].sum()
```

7. Then, I reset the index and made my ‘female’ data into a dataframe - ‘female_df’

```
female.reset_index()
female_df = pd.DataFrame(female)
```

8. Next, because I want to make my visualization on only the top five industries with female business owners I performed nlargest and saved that data to a new data frame

   a. If you wanted to see a different number of top industries in the visualization, you could change ‘5’ here to your desired number

```
top_female=female.nlargest(5)
top_female_df = pd.DataFrame(top_female)
```

x. To transform 'Top 5 Industries of Male Business Owners' data

1. First make a specialized copy of my dataset ‘NACIS_bo’ that only shows the rows where the data concerns males, where OWNER_SEX = 003. Here is the code:

```
male = NAICS_bo[NAICS_bo['OWNER_SEX'] == '003'].copy()
```

2. Then, convert the ‘OWNPDEMP’ (Number of owners of respondent employer firms) column to an integer. Here is the code:

```
male['OWNPDEMP'] = male['OWNPDEMP'].astype(int)
```

3. #only show the rows NAICS label, owner sex label, and # of responding firms

4. Then, only show the columns 'NAICS2017_LABEL', 'OWNER_SEX', 'OWNPDEMP'

```
male = male[['NAICS2017_LABEL', 'OWNER_SEX', 'OWNPDEMP']]
```

5. Next, I renamed the columns 'NAICS2017_LABEL' and 'OWNPDEMP' ‘Industry’ and 'Male Business Owners', respectively to make them easier to interpret. Here is the code:

```
rename OWNPDEMP to Male Business Owners
male.rename(columns = {'OWNPDEMP':'Male Business Owners',
'NAICS2017_LABEL':'Industry'}, inplace = True)
#group that by NAICS code
```

6. Then to get the count of how many male business owners are in each industry, I grouped by industry. Here is the code:

```
male = male.groupby('Industry')['Male Business Owners'].sum()
```

7. Then, I reset the index and made my ‘male’ data into a dataframe - ‘male_df’

```
male.reset_index()
male_df = pd.DataFrame(male)
```

8. Next, because I want to make my visualization on only the top five industries with male business owners I performed nlargest and saved that data to a new data frame

a. If you wanted to see a different number of top industries in the visualization, you could change ‘5’ here to your desired number

```
top_male=male.nlargest(5)
top_male_df = pd.DataFrame(top_male)
```

xi. To transform ‘Percent Business Ownership by Sex’ data

1. Merge the previous male_df and female_df as follows:

```
merge_sex = pd.concat([male_df, female_df], axis=1, join="inner")
```

2. Then, add a row to the bottom of the table with the sum the total business owners by sex as follows:

```
merge_sex.loc["Total Business Owners"] = merge_sex.sum()
```

3. Next, make a dataframe of just the totals. Here is the code:

```
merge_sex_df = merge_sex.tail(1)
merge_sex_df = pd.DataFrame(merge_sex_df)
```

4. Then, to make a pie chart, we will need to pivot the table as follows:

```
sex_pivot =pd.pivot_table(
    data=merge_sex_df,
    columns=['Industry']
)
```

xii. To transform ‘Top 5 Industries with Least Sex Disparity in Business Ownership’ data

1. To compare the sums of total males and females in specific industries, I merged the ‘Industries for Male Business Owners' with ‘Industries for Female Business Owners'. Here is the code:

```
result = pd.concat([male_df, female_df], axis=1, join="inner")
```

2. Then, create a new column subtracting the male business owners from female business owners, per industry. Here is the code:

```
result['Sex Difference'] = result['Female Business Owners']-result['Male Business
Owners']
```

3. Then sort the rows so the greatest numbers on ‘Sex Difference’ is at the top. Here is the code:

```
result=result.sort_values(by='Sex Difference', ascending=False)
```

4. To get the top five industries, use this code:

    a. If you would like the visual to present a different number of industries, change ‘5’ to the desired number below.

```
results = result.head(5)
```

d. In Peter’s Data:

i. To create visualization 1

1. Isolate the five columns shown from ‘bo_df’ as a separate dataframe called ‘ownchar’.

```
ownchar = bo_df[['NAICS2017', 'NAICS2017_LABEL', 'OWNCHAR', 'OWNCHAR_LABEL', 'OWNPDEMP']]
```

2. Create three more dataframes based on the relevance of a few reasons for owning business (start business, carry on family business, help community).

```
reason_start = ownchar.loc[ownchar['OWNCHAR'].isin(['O01N', 'O01S', 'O01V'])]
reason_family = ownchar.loc[ownchar['OWNCHAR'].isin(['OR1N', 'OR1S', 'OR1V'])]
reason_help = ownchar.loc[ownchar['OWNCHAR'].isin(['OS1N', 'OS1S', 'OS1V'])]
```

3. Concatenate these dataframes, ignoring index and axis set to 0.

```
reasons = pd.concat([reason_start, reason_family, reason_help], ignore_index=True, axis=0)
```

4. Cast ‘OWNPEMP’ as an integer.

```
reasons['OWNPDEMP'] = reasons['OWNPDEMP'].astype(int)
```

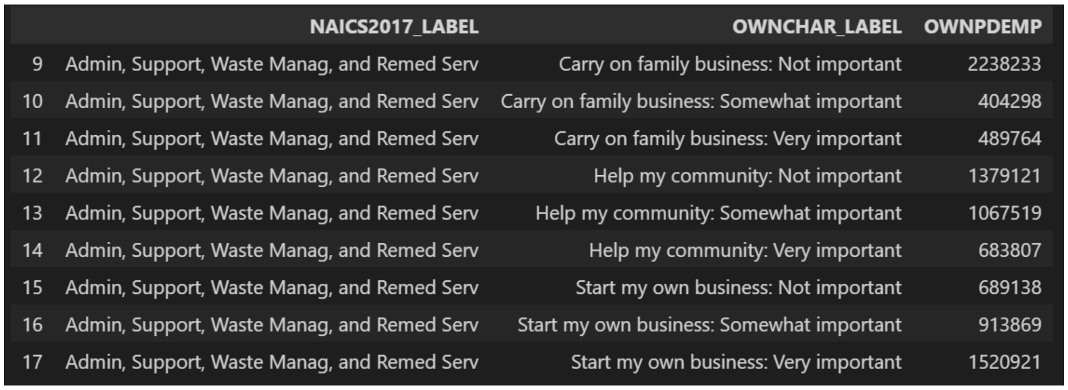
5. Group the ‘reasons’ dataframe doubly by [‘NAICS2017_LABEL’, ‘OWNCHAR_LABEL’] and average ‘OWNPDEMP’.

```
rn = reasons.groupby(['NAICS2017_LABEL', 'OWNCHAR_LABEL'],
as_index=False)['OWNPDEMP'].average()
```

6. Drop five underrepresented or impertinent sets of NAICS2017 sector rows from the new dataframe as below.

```
rn.drop(rn[rn['NAICS2017_LABEL'] == 'Total for all sectors'].index, inplace=True)
rn.drop(rn[rn['NAICS2017_LABEL'] == 'Industries not classified'].index, inplace=True)
rn.drop(rn[rn['NAICS2017_LABEL'] == 'Agriculture, forestry, fishing and hunting'].index, inplace=True)
rn.drop(rn[rn['NAICS2017_LABEL'] == 'Utilities'].index, inplace=True)
rn.drop(rn[rn['NAICS2017_LABEL'] == 'Mining, quarrying, and oil and gas extraction'].index, inplace=True)
```

7. Abbreviate 'Administrative and support and waste management and remediation services' rows to 'Admin, Support, Waste Manag, and Remed Serv' by a `str.replace` method. A segment of the result is shown.



	NAICS2017_LABEL	OWNCHAR_LABEL	OWNPDEMP
9	Admin, Support, Waste Manag, and Remed Serv	Carry on family business: Not important	2238233
10	Admin, Support, Waste Manag, and Remed Serv	Carry on family business: Somewhat important	404298
11	Admin, Support, Waste Manag, and Remed Serv	Carry on family business: Very important	489764
12	Admin, Support, Waste Manag, and Remed Serv	Help my community: Not important	1379121
13	Admin, Support, Waste Manag, and Remed Serv	Help my community: Somewhat important	1067519
14	Admin, Support, Waste Manag, and Remed Serv	Help my community: Very important	683807
15	Admin, Support, Waste Manag, and Remed Serv	Start my own business: Not important	689138
16	Admin, Support, Waste Manag, and Remed Serv	Start my own business: Somewhat important	913869
17	Admin, Support, Waste Manag, and Remed Serv	Start my own business: Very important	1520921

ii. To create visualization 2

1. Extract columns for 'IMPACTWK_U_LABEL' in the dataframe 'tech_df' regarding 'Artificial Intelligence', 'Specialized Software', and 'Robotics'. Name this table 'tech_impact.'

```
tech_impact = tech_df[tech_df["IMPACTWK_U_LABEL"].str.contains("Equipment")==False]
tech_impact = tech_impact[tech_impact["IMPACTWK_U_LABEL"].str.contains("Cloud")==False]
```

2. Delete rows of 'IMPACTWK_U_LABEL' containing 'Not applicable', 'Total reporting', 'Did not change', and 'All firms'.

```
tech_impact = tech_impact[tech_impact["IMPACTWK_U_LABEL"].str.contains("Not applicable")==False]
tech_impact = tech_impact[tech_impact["IMPACTWK_U_LABEL"].str.contains("Total Reporting")==False]
tech_impact = tech_impact[tech_impact["IMPACTWK_U_LABEL"].str.contains("Did not change")==False]
tech_impact = tech_impact[tech_impact["IMPACTWK_U_LABEL"] != 'All firms']
```

3. Delete rows of 'NAICS2017_LABEL' containing 'Total for all sectors'.

```
tech_impact = tech_impact[tech_impact["NAICS2017_LABEL"].str.contains("Total for all sectors")==False]
```

4. Cast 'FIRMPDEMP' and 'FIRMPDEMP_PCT' as integer and float, respectively.

```
tech_impact['FIRMPDEMP'] = tech_impact['FIRMPDEMP'].astype(int)
tech_impact['FIRMPDEMP_PCT'] = tech_impact['FIRMPDEMP_PCT'].astype(float)
```

5. Isolate the eight most popular sectors by number of respondents via .loc and .isin methods.

```
tech_impact = tech_impact.loc[tech_impact["NAICS2017_LABEL"].isin(['Accommodation and food services',
         'Administrative and support and waste management and remediation services',
         'Health care and social assistance',
         'Professional, scientific, and technical services',
         'Retail trade',
         'Manufacturing',
         'Wholesale trade',
         'Construction'])]
```

6. Drop columns 0-2, 5-18, and 20-28, leaving columns 3, 4, and 19. A snippet is shown below.

```
tech_impact.drop([tech_impact.columns[0], tech_impact.columns[1], tech_impact.columns[2],...)
```

7. Reset the index of 'tech_impact'.

```
tech_impact = tech_impact.reset_index()
```

8. Drop the resulting 'index' column.

```
tech_impact.drop('index', axis=1, inplace=True)
```

9. Rename 'IMPACTWK_U_LABEL' to simply 'Impact' for ease of reference.

```
tech_impact.rename(columns={'IMPACTWK_U_LABEL':'Impact'}, inplace=True)
```

10. Abbreviate the 'Administrative and support…' rows as in Peter's visualization 1 step 7.

```
tech_impact['NAICS2017_LABEL'] = tech_impact['NAICS2017_LABEL'].str.replace('Administrative and support and waste management and remediation services', 'Admin, Support, Waste Manag, and Remed Serv')
```

11. Shorten the strings of the impact columns for each of the three technology types specified in step 1. An example is shown. Let Prod. = production workers, Nonprod. = nonproduction workers, Superv. = supervisory workers, and Nonsuperv. = nonsupervisory workers. One of the three technologies of interest for each worker type is shown below.

```
tech_impact['Impact'] = tech_impact['Impact'].str.replace('Artificial Intelligence: Increased number of production workers employed by this business', 'AI: INCREASED Prod.')
tech_impact['Impact'] = tech_impact['Impact'].str.replace('Artificial Intelligence: Decreased number of production workers employed by this business', 'AI: DECREASED Prod.')
tech_impact['Impact'] = tech_impact['Impact'].str.replace('Artificial Intelligence: Increased number of nonproduction workers employed by this business', 'AI: INCREASED Nonprod.')
tech_impact['Impact'] = tech_impact['Impact'].str.replace('Artificial Intelligence: Decreased number of nonproduction workers employed by this business', 'AI: DECREASED Nonprod.')
tech_impact['Impact'] = tech_impact['Impact'].str.replace('Artificial Intelligence: Increased number of supervisory workers employed by this business', 'AI: INCREASED Superv.')
tech_impact['Impact'] = tech_impact['Impact'].str.replace('Artificial Intelligence: Decreased number of supervisory workers employed by this business', 'AI: DECREASED Superv.')
tech_impact['Impact'] = tech_impact['Impact'].str.replace('Artificial Intelligence: Increased number of nonsupervisory workers employed by this business', 'AI: INCREASED Nonsuperv.')
tech_impact['Impact'] = tech_impact['Impact'].str.replace('Artificial Intelligence: Decreased number of nonsupervisory workers employed by this business', 'AI: DECREASED Nonsuperv.')
```

12. Rename the ‘Impact’ column to ‘Impact of Technology (Artificial Intelligence, Specialized Software, Robotics)\nOn Types of Workers Employed’.

```
tech_impact.rename(columns={'Impact':'Impact of Technology (Artificial Intelligence, Specialized Software, Robotics)\nOn Types of Workers Employed'}, inplace=True)
```

13. Pivot the ‘tech_impact’ dataframe on values ‘FIRMPDEMP’, index ‘Impact of Technology…’, and columns ‘NAICS2017_LABEL’, with ‘aggfunc’ equal to ‘np.sum’. Call this pivot table ‘piv’.

```
piv = tech_impact.pivot_table(values='FIRMPDEMP', index=['Impact of Technology (Artificial Intelligence, Specialized Software, Robotics)\nOn Types of Workers Employed'], columns=['NAICS2017_LABEL'], aggfunc=np.sum)
```

14. Convert ‘piv’ back to a pd.DataFrame by parameterizing piv.to_records().

```
piv = pd.DataFrame(piv.to_records())
```

15. Rename ‘NAICS2017_LABEL’ to ‘Industry Sector’.

```
piv.rename(columns={'NAICS2017_LABEL':'Industry Sector'}, inplace=True)
```

16. In a new cell, produce a list of columns in the desired order for the subsequent graph.

```
list_piv = ['Impact of Technology (Artificial Intelligence, Specialized Software, Robotics)\nOn Types of
Workers Employed',
 'Professional, scientific, and technical services',
 'Manufacturing',
 'Retail trade',
 'Health care and social assistance',
 'Wholesale trade',
 'Construction',
 'Accommodation and food services',
 'Admin, Support, Waste Manag, and Remed Serv']
```

17. Reindex the dataframe to this list.

```
piv = piv.reindex(columns=list_piv)
```

18. Change all 0 values in the table to 1 so that the values appear on the log scale graph. There are four columns to modify.

```
condition  = (piv['Health care and social assistance'] == 0)
piv.loc[condition, 'Health care and social assistance'] = 1
condition  = (piv['Construction'] == 0)
piv.loc[condition, 'Construction'] = 1
condition  = (piv['Accommodation and food services'] == 0)
piv.loc[condition, 'Accommodation and food services'] = 1
condition  = (piv['Admin, Support, Waste Manag, and Remed Serv'] == 0)
piv.loc[condition, 'Admin, Support, Waste Manag, and Remed Serv'] = 1
```

iii. To create visualization 3

1. Create a ‘techuse’ dataframe from the ‘tech_df’ columns: ‘NAICS2017_LABEL’, ‘TECHUSE_LABEL’, ‘FIRMPDEMP’.

```
techuse = tech_df[['NAICS2017_LABEL','TECHUSE_LABEL', 'FIRMPDEMP']]
```

2. Cast ‘FIRMPDEMP’ as an integer.

```
techuse['FIRMPDEMP'] = techuse['FIRMPDEMP'].astype(int)
```

3. Eliminate the duplicates in ‘techuse’.

```
techuse = techuse.drop_duplicates()
```

4. Remove the ‘TECHUSE_LABEL’ rows containing ‘All firms’, ‘Don’t know’, ‘but did not know’, ‘Total Reporting’, and ‘Total Use’, forming a new dataframe ‘tu’ in so doing.

```
tu = techuse[techuse["TECHUSE_LABEL"].str.contains("All firms")==False]
tu = tu[tu["TECHUSE_LABEL"].str.contains("Don't know")==False]
tu = tu[tu["TECHUSE_LABEL"].str.contains("but did not use")==False]
tu = tu[tu["TECHUSE_LABEL"].str.contains("Total Reporting")==False]
tu = tu[tu["TECHUSE_LABEL"].str.contains("Total use")==False]
```

5. Group the ‘tu’ dataframe by ‘TECHUSE_LABEL’, taking the median of ‘FIRMPDEMP’.

```
tu = tu.groupby('TECHUSE_LABEL', as_index=False)['FIRMPDEMP'].median()
```

6. Split the ‘TECHUSE_LABEL’ into two columns ‘TECHUSE_LABEL’ and ‘AMOUNT’ according to the colon+space delimiter (‘: ’), expanding them.

```
tu[['TECHUSE_LABEL', 'AMOUNT']] = tu['TECHUSE_LABEL'].str.split(': ', expand=True)
```

7. Shorten and reduce redundancy in the strings ‘Did not use’, ‘High use’, ‘Moderate use’, and ‘Low use’ within the ‘AMOUNT’ column to ‘None’, ‘High’, ‘Moderate’, and ‘Low’, respectively.

```
tu['AMOUNT'] = tu['AMOUNT'].str.replace('Did not use', 'None')
tu['AMOUNT'] = tu['AMOUNT'].str.replace('High use', 'High')
tu['AMOUNT'] = tu['AMOUNT'].str.replace('Moderate use', 'Moderate')
tu['AMOUNT'] = tu['AMOUNT'].str.replace('Low use', 'Low')
```

8. Rename the ‘Cloud-Based’ rows in the ‘TECHUSE_LABEL’ column to ‘Cloud-Based Systems’.

```
tu['TECHUSE_LABEL'] = tu['TECHUSE_LABEL'].str.replace('Cloud-Based', 'Cloud-Based Systems')
```

Load

Creating the Visualizations:

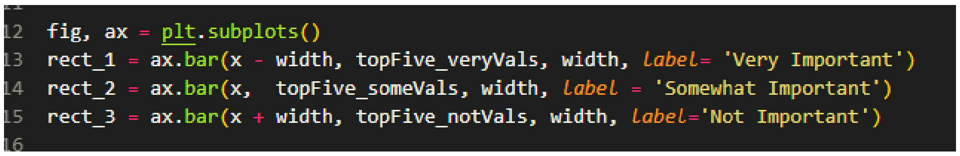
a. In Life Work Balance
   i. Visual 1:
      1. We use the plotly.graph_objects library, to call go.Figure(go.Choroplethmapbox()) inside this we pulled a geojson from github

that allows us to draw the states onto the map. This will be stored as the passed in argument 'geojson'. We also pass in, locations, z, colorbar_title, colorscale, zmin, zmax, marker_opacity, and marker_line_width. Locations define what states correspond to what values from the geojson file retrieved from github. Z gives the function the actual numeric data upon which to create the coloring. Colorbar_title gives a label to the legend and zmin and zmax define the min and max values on the colors defined by the map.

2. We then tackle an update of the layout to include the title, geo, mapbox_style, mapbox_zoom, mapbox_center, and margin. The most important one here is geo, which requires a dictionary in which we can define things like scope, which focuses the map, projection which defines what kind of version of the map we want, and showlakes, which allows us to request that the lakes appear on the map. Finally, the mapbox options allow for a bit more restriction over how the final image comes out and where the center of the map is when we first look at the image. Note that normally the map is interactive and allows us to see anywhere on the globe.

ii. Visual 2 and 3:

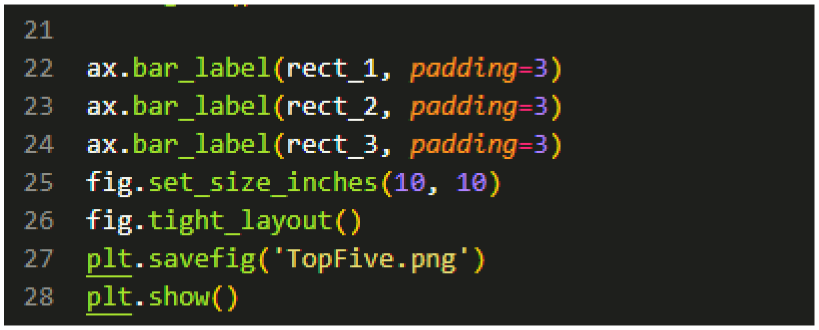
1. Define the plt.subplots() function and then create three bar objects. Inside these bar objects, do one with x -width, x, and x + width. This way we can get three bars right next to each other and they don't overlap. Be aware that you may have to adjust the initial width value so that one set of three bars doesn't overlap with another set of three bars.



```
fig, ax = plt.subplots()
rect_1 = ax.bar(x - width, topFive_veryVals, width, label= 'Very Important')
rect_2 = ax.bar(x,  topFive_someVals, width, label = 'Somewhat Important')
rect_3 = ax.bar(x + width, topFive_notVals, width, label='Not Important')
```

2. Next, we set the ylabel to percentage and the title of the graph corresponding to the appropriate data being shown. We also use .set_xticks() to allow for non-numeric values on the x axis of the graph.

3. Finally we add a legend and bar labels to get values onto the graph to allow for some visual clarity before applying a size, tight_layout, and saving the figure as a png.



```

ax.bar_label(rect_1, padding=3)
ax.bar_label(rect_2, padding=3)
ax.bar_label(rect_3, padding=3)
fig.set_size_inches(10, 10)
fig.tight_layout()
plt.savefig('TopFive.png')
plt.show()
```

a. In Technology vs Owner Age Data:

1. Visuals 1:

a. Set the figure size and use matplotlib's `.plot()` feature to initialize a horizontal bar chart. Set the fontsize to 12 and list the colors for the visualization.

```
plt.figure(figsize=(15,10))
ax = age.plot(kind='barh', fontsize=12, color=['#8b0000','#8b0000','#8b0000','#D3D3D3','#00008b','#D3D3D3'])
```

b. Set the title, with a fontsize of 30 and a pad size of 30. Set the y label and x label and give them font sizes of 15

```
plt.title('Distribution of Business Owner Ages in the US (2018)', fontsize=30, pad=30)
plt.ylabel('Age Groups', fontsize=15)
plt.xlabel('Number of Owners (in Millions)', fontsize=15)
```

c. Use `.text()` to add value labels to the bars.

```
plt.text(age.iloc[0], 0, '   15.6K', fontsize=12, color='#8b0000')
plt.text(age.iloc[1], 1, '   220.8K', fontsize=12, color='#8b0000')
plt.text(age.iloc[2], 2, '   692.7K', fontsize=12, color='#8b0000')
plt.text(age.iloc[4], 4, '   1.24M', fontsize=12, color='#00008b')
```

d. Remove the top and right spines, and set the x axis limit to (0,1400000)

```
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.xlim(0,1400000)
```

e. Repeat for second histogram:

```
plt.figure(figsize=(15,10))
ax = tech_sell.plot(kind='barh', fontsize=12, color=['#8b0000', '#8b0000', '#8b0000', '#00008b', '#D3D3D3'])
plt.title('Distribution of Types of Tech Sold by Firms in the US (2018)', fontsize=30, pad=30)
plt.ylabel('Type of Technology', fontsize=15)
plt.xlabel('Number of Firms', fontsize=15)

plt.text(tech_sell.iloc[0], 0, '   15.0K', fontsize=12, color='#8b0000')
plt.text(tech_sell.iloc[1], 1, '   19.7K', fontsize=12, color='#8b0000')
plt.text(tech_sell.iloc[2], 2, '   108.6K', fontsize=12, color='#8b0000')
plt.text(tech_sell.iloc[3], 3, '   152.3K', fontsize=12, color='#00008b')
plt.text(tech_sell.iloc[4], 4, '   185.3K', fontsize=12)
```

```
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
plt.xlim(0,200000)
```

2. **Visual 2:**
    a. Turn the `merged_cbo_tcb` into a correlation matrix.

```
corr_matrix = merged_cbo_tcb.corr()
```

    b. Set the figure size and turn it into a heatmap with seaborn. Set annot to True, vmin to -1, vmax to 1 and the color mapping.

```
plt.figure(figsize=(15,10))
sns.heatmap(corr_matrix, annot=True, vmin=-1, vmax=1, cmap='flare')
```

    c. Set the title, the fontsize to 30, its loc to 30, and give it pad of 30.

```
plt.title('      Correlation: Age Groups vs. Tech Sold (2018)', fontsize=30, loc='right', pad=30)
```

3. **Visuals 3:**
    a. Set the figure size, set up a scatter plot, give it a color, and change the points sizes.

```
plt.figure(figsize=(15,10))
plt.scatter(merged_cbo_tcb['Owners Aged 35-44'], merged_cbo_tcb['Firms Selling Specialized Equipment'], alpha=1, c='orange', s=100)
```

    b. Set the title, x label, y label, and an x axis limit.

```
plt.title('Number of Owners Aged 35-44 vs. Number of Firms Selling Specialized Equipment -- by Industry (2018)', fontsize=25, pad=30)
plt.xlabel('Owners Aged 35-44', fontsize=20)
plt.ylabel('Firms Selling Specialized Equipment', fontsize=20)
plt.xlim(0,150000)
```

    c. Create a function to add labels to the points and use it.

```
def label_point(x, y, val, ax):
    a = pd.concat({'x':x, 'y':y, 'val':val}, axis=1)
    for i, point in a.iterrows():
        plt.text(point['x']+0.5,point['y']+200,str(point['val']), fontsize=8)
```

```
label_point(merged_cbo_tcb['Owners Aged 35-44'],merged_cbo_tcb['Firms Selling Specialized Equipment'],merged_cbo_tcb['Industry'],
plt.gca())
```

d. Repeat for other scatter plot:

```
plt.figure(figsize=(15,10))
plt.scatter(merged_cbo_tcb['Owners Aged 55-64'], merged_cbo_tcb['Firms Selling Specialized Equipment'], alpha=1, c='blue', s=100)
plt.title('Number of Owners Aged 55-64 vs. Number of Firms Selling Specialized Equipment -- by Industry (2018)', fontsize=25, pad=30)
plt.xlabel('Owners Aged 55-64', fontsize=20)
plt.ylabel('Firms Selling Specialized Equipment', fontsize=20)
plt.xlim(0,260000)

label_point(merged_cbo_tcb['Owners Aged 55-64'],merged_cbo_tcb['Firms Selling Specialized Equipment'],merged_cbo_tcb['Industry'],
plt.gca())
```

c. To create Industry By Sex Visuals:

i. To load the 'Top industries of Female Business Owners' visual

1. To create the visual, I first set the framework to change my theme as desired. Here is the code:

```
fig=plt.figure(figsize=(15,10))
sns.set_theme(style="white", palette='pastel')
```

2. Next, I call on my dataframe and plot() then pass in the title and kind values.

```
ax = top_female_df.plot(kind='bar', width= 1, ax=fig.gca())
```

3. Because I want the female bar chart to have the same axis as the male bar chart, I will adjust the y axis. Here is the code:

```
plt.ylim(0.0,500000)
```

4. Then I make the table look nice by adding the table title, and axis titles. Here is the code:

```
ax.set_title('Top 5 Industries of Female Business Owners', fontsize=24)
ax.set_xlabel('Industries', fontsize=22)
ax.set_ylabel('Number of Owners', fontsize=22)
```

5. Then, to clarify what the bars are indicating, I annotated the number of business owners above each bar. Here is the code:

```
for p in ax.patches:
     ax.annotate((p.get_height()),(p.get_x()+p.get_width()/2.,p.get_height()),
ha='center',va='center', xytext=(0, 10), fontsize=16, textcoords='offset points')
```

6. To make the table more readable, I removed the right and top spines and added a y grid. Here is the code:

```
ax.grid(axis='y')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```

7. Lets make the legend a readable size:

```
plt.legend(fontsize=18)
```

8. Because my industry names are long, let's angle them and make all the ticks larger to read. Here is the code:

```
plt.xticks(rotation=45, ha='right', fontsize=18) #angle and centered under column!
plt.yticks(fontsize=18)
```

9. This code will show the table:

```
plt.show()
```

ii. To load the 'Top industries of Female Business Owners' visual

1. To create the visual, I first set the figure size and framework to change my theme as desired. Here is the code:

```
fig=plt.figure(figsize=(15,10))
sns.set_theme(style="white", palette='pastel')
```

2. Next, I call on my dataframe and plot() then pass in the kind value and call on the figure size.

```
ax = top_male_df.plot(kind='bar', width=1,ax=fig.gca())
```

3. Because I want the male bar chart to have the same axis as the female bar chart I will adjust the y axis. Here is the code:

```
plt.ylim(0.0,500000)
```

4. Then I make the table look nice by adding the table title, and axis titles. Here is the code:

```
ax.set_title('Top Industries of Male Business Owners', fontsize=24)#correct
ax.set_xlabel('Industries', fontsize=22)
ax.set_ylabel('Number of Owners', fontsize=22)
```

5. Then, to clarify what the bars are indicating, I annotated the number of business owners above each bar. Here is the code:

```
for p in ax.patches:
    ax.annotate((p.get_height()),(p.get_x()+p.get_width()/2.,
p.get_height()),ha='center',va='center', xytext=(0, 10), fontsize=16,
textcoords='offset points')
```

6. To make the table more readable, I removed the right and top spines and added a y grid. Here is the code:

```
ax.grid(axis='y')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```

7. Lets make the legend a readable size:

```
plt.legend(fontsize=18)
```

8. Because my industry names are long, let's angle them and make all the ticks larger to read. Here is the code:

```
plt.xticks(rotation=45, ha='right', fontsize=18) #angle and centered under column!
plt.yticks(fontsize=18)
```

9. This code will show the table:

```
plt.show()
```

iii. To load the 'Business Ownership by Sex' visual

1. To create the visual, I first set the figure size and framework to change my theme as desired. Here is the code:

```
fig=plt.figure(figsize=(15,10))
sns.set_theme(style="white", palette='pastel')
```

2. Next, I call on my dataframe and plot() then pass in the kind of chart as pie, what I want represented, the figure size, and some visual adjustments like presenting the percentages, adjusting the font size, and not presenting the y label. Here is the code:

```
ax = sex_pivot.plot(kind='pie', y='Total Business Owners', ax=fig.gca(),
autopct='%1.0f%%', shadow=True, textprops={'fontsize': 22}, labeldistance=None,
ylabel='')
```

3. Then, I added the title as seen here:

```
ax.set_title('Business Ownership by Sex\n\n', fontsize=24)
```

4. Lastly, I adjusted the font size in the legend and plt.show presents the table. The code is:

```
plt.legend(fontsize=18, bbox_to_anchor=(.225, 1.07), loc='upper left')
plt.show()
```

iv. Load to create the 'Top 5 Industries with Least Sex Disparity in Business Ownership'

1. To create the visual, I first set the figure size and framework to change my theme as desired. Here is the code:

```
fig=plt.figure(figsize=(15,10))
sns.set_theme(style="white", palette='pastel')
```

2. Next, I call on my dataframe and plot() then pass in the kind of chart as bar, the desired bar width, and the figure size. Here is the code:

```
ax = results.plot(kind='bar', width=1, ax=fig.gca())
```

3. Then, I added the title and axis labels as well as their desired font size as seen here:

```
ax.set_title('Top 5 Industries with Least Sex Disparity in Business Ownership',
fontsize=24)#correct
ax.set_xlabel('Industries', fontsize=22)
ax.set_ylabel('Number of Owners', fontsize=22)
```

4. Then, to clarify what the bars are indicating, I annotated the number of business owners above each bar. Here is the code:

```
for p in ax.patches:
    ax.annotate((p.get_height()),(p.get_x()+p.get_width()/2.,
p.get_height()),ha='center',va='center', xytext=(0, 10), fontsize=12,
textcoords='offset points')
```

5. To make the table more readable, I removed the right and top spines and added a y grid. Here is the code:

```
ax.grid(axis='y')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
```

6. Let's make the legend a more readable size:

```
plt.legend(fontsize=18)
```

7. Because my industry names are long, let's angle them and make all the ticks larger to read. Here is the code:

```
plt.xticks(rotation=45, ha='right', fontsize=18) #angle and centered under column!
plt.yticks(fontsize=18)
```

8. This code will show the table:

```
plt.show()
```

b. Peter's data

i. Visual 1

1. In a new cell, import the following libraries for the visual.

```
import numpy as np
from matplotlib.colors import LogNorm, Normalize
from matplotlib.ticker import MaxNLocator
```

2. Change the figure size via (15,10).

```
plt.figure(figsize=(15,10))
```

3. Pivot ‘rn’ on values ‘OWNPDEMP’, index ‘OWNCHAR_LABEL’, and columns ‘NAICS2017_LABEL’.

```
df_heatmap = rn.pivot_table(values='OWNPDEMP', index='OWNCHAR_LABEL', columns='NAICS2017_LABEL')
```

4. Sort the columns in ‘df_heatmap’ by values inside them.

```
df_heatmap = df_heatmap.sort_values(df_heatmap.last_valid_index(), axis=1)
```

5. Further extract the top 9 industry sectors by number of respondents to create a square of rows and columns.

```
df_heatmap = df_heatmap.iloc[:,7:16]
```

6. Generate a heatmap adding a log scale, color gradient ‘coolwarm’, and tick marks with ‘cbar_kws’.

```
hm = sns.heatmap(df_heatmap,cmap='coolwarm',annot=False,norm=LogNorm(),square=True,
cbar_kws={'ticks':MaxNLocator(8), 'format':'%.i'})
```

7. Format the x-axis label, y-axis label, and title as below.

```
hm.set_xlabel('NAICS2017 Sector')
hm.set_ylabel('Owner Characteristics')
hm.set_title('Variability in Reasons for Owning Business by Top 9 Industry Sectors
(Logarithmic Scale)', fontsize=15)
```

8. Rotate the x-axis tick labels to conserve space.

```
plt.xticks(rotation=45,horizontalalignment='right')
```

9. Perform plt.show().

```
plt.show()
```

ii. Visual 2

1. In a new cell, import the libraries ‘seaborn’ and ‘matplotlib.ticker’.

```
import seaborn as sns
import numpy as np
import matplotlib.ticker as mticker
```

2. Set the figure size to (15,10).

```
plt.figure(figsize=(15,10))
```

3. Set the theme to ‘whitegrid’.

```
sns.set_theme(style="whitegrid")
```

4. Set the font_scale to 1 and style to ‘ticks’.

```
sns.set(font_scale=1, style='ticks')
```

5. Make a PairGrid from ‘piv.columns[1:9]’ as x_vars and ‘Impact of Technology…” as y_vars, sorting the values of the ‘Professional, scientific, and technical services’ in descending order, setting height to 12, and aspect to 0.2.

```
g = sns.PairGrid(piv.sort_values('Professional, scientific, and technical services', ascending=False),
               x_vars=piv.columns[1:9], y_vars="Impact of Technology (Artificial Intelligence,
Specialized Software, Robotics)\nOn Types of Workers Employed",
               height=12, aspect=0.2)
```

6. Apply a map method to the pairgrid by passing in the stripplot function to generate a dot plot. Set size to 10, orient ‘h’, jitter ‘False’, palette ‘flare_r’, linewidth 1, and edgecolor ‘w’.

```
g.map(sns.stripplot, size=10, orient="h", jitter=False,
      palette="flare_r", linewidth=1, edgecolor="w")
```

7. Rename the xlabel to ‘Number of firms reporting this impact’.

```
g.set(xlabel="Number of firms\nreporting this impact")
```

8. Set the xscale to ‘log’.

```
g.set(xscale='log')
```

9. Insert log ticks in a for loop iterating over the pairgrid .axes.flat . The specifications came with this seaborn example.

```
for ax in g.axes.flat:
    locmin = mticker.LogLocator(base=10, subs=np.arange(0.1, 1, 0.1), numticks=10)
    ax.xaxis.set_minor_locator(locmin)
```

10. Write the correct industry sector names for each of pairgrid plot based on the ordered list from step 16.

```python
titles = ['Professional, scientific,\nand technical services',
        'Manufacturing',
        'Retail trade',
        'Health care and\nsocial assistance',
        'Wholesale trade',
        'Construction',
        'Accommodation and\nfood services',
        'Admin, Support, Waste\nManag, and Remed Serv']
```

11. Add the titles to the pairgrid plots in a for loop over a zip function parameterizing g.axes.flat and titles, with ax and title as iterators.

```python
for ax, title in zip(g.axes.flat, titles):
    ax.set(title=title)
    ax.xaxis.grid(False)
    ax.yaxis.grid(True)
```

12. Insert the sns.despine method with left 'False' and bottom 'True'.

```python
sns.despine(left=False, bottom=True)
```

13. Perform plt.show()

```python
plt.show()
```

iii. Visual 3

1. In a new cell, turn the figure size to (15,10).

```python
plt.figure(figsize=(15,10))
```

2. Set the sns theme to style 'darkgrid'.

```python
sns.set_theme(style="darkgrid")
```

3. Let 'g' equal an sns.catplot based on 'tu', with the specifications as below. In particular, x represents the categorical 'TECHUSE_LABEL', y the numerical 'FIRMPDEMP', the slicer on 'AMOUNT', size of confidence interval 'ci' equal to the standard deviation 'sd', palette 'bright', alpha '.6', height '6', and edgecolor 'black'.

```python
g = sns.catplot(
    data=tu, kind="bar",
    x="TECHUSE_LABEL", y="FIRMPDEMP", hue="AMOUNT",
    ci="sd", palette="bright", alpha=.6, height=6,
    edgecolor="black")
```

4. Despine the left of ‘g’ with left = True.

```
g.despine(left=True)
```

5. Set the axis labels to ‘Type of Technology’, ‘Number of Respondent Firms (log scale)’, and font weight equal to ‘bold’.

```
g.set_axis_labels("\nType of Technology", "Number of Respondent Firms (log scale)",
fontweight='bold')
```

6. Provide the legend title ‘Level’.

```
g.legend.set_title("Level")
```

7. Set the y-axis scale to ‘log’.

```
g.set(yscale='log')
```

8. Reconfigure the figure size by the ‘.set_size_inches’ method as (14,6).

```
g.figure.set_size_inches(14,6)
```

9. Put the number of respondents to each field on top of the bars through a for loop iterating over g.face_axis(0,0).patches. Fill in all of the required fields for ax.text. Join each index via dot method on get_x()+0.02, get_height(), and formatting this on 0:.0f precision, and setting color to ‘gray’, rotation to ‘45’, and size to ‘10’.

```
ax = g.facet_axis(0, 0)
for p in ax.patches:
    ax.text(p.get_x()+0.02,
            p.get_height(),
            '{0:.0f}'.format(p.get_height()),
            color='gray', rotation=45, size=10)
```

10. Add a plot title ‘Types of technology use by level in firms from 2016 to 2018’, with font size equal to 20.

```
plt.title("Types of Technology Use by Level in Firms from 2016 to 2018\n", fontsize=20)
```

11. Perform plt.show().

```
plt.show()
```

Conclusion

In this ETL Report, the data sources were identified in the Extraction section. Then, the required transformations were identified in the Transforming section. Lastly, the steps and documents required to Load the data to visualizations were presented in the Loading section.