ETL Report

Group #6:

Joel Garcia, Nate Williamson, Logan Sell, & Beth Vander Hoek

09/03/2022

Introduction:

For this project, we explored the Census 2019 Annual Business Survey's four datasets on companies, characteristics of businesses, characteristics of business owners, and technology characteristics of businesses for 2018 at both the nation and state levels. While looking at the data, we decided to investigate the following four questions:

1. How does worker pay and employment type vary by owner race and gender?
2. How does veteran business ownership vary by gender, race, and service-disability status?
3. How is technology used across businesses?
4. How does owner education attainment relate to annual pay within different industries?

After extracting, cleaning, and transforming the data, we then were able to use it to answer these questions using visualizations to display our findings.

Data Sources:

U.S. Census Bureau.(2019). *Annual Business Survey (ABS): Characteristics of Businesses* [Data set]. https://api.census.gov/data/2018/abscb.html Accessed on 09/03/2022

U.S. Census Bureau.(2019). *Annual Business Survey (ABS): Characteristics of Business Owners* [Data set]. https://api.census.gov/data/2018/abscbo.html Accessed on 09/03/2022

U.S. Census Bureau.(2019). *Annual Business Survey (ABS): Company Summary* [Data set]. https://api.census.gov/data/2018/abscs.html Accessed on 09/03/2022

U.S. Census Bureau.(2019). *Annual Business Survey (ABS): Technology Characteristics of Businesses* [Data set]. https://api.census.gov/data/2018/abstcb.html Accessed on 09/03/2022

Extraction:

To begin, we loaded this data using a requested and activated API key from the Census Business Survey 2019 API website. We used a separate API url call for each of the four datasets (listed below). To call the API for its data, we took the following steps:

1. Import requests, json, and pandas as pd for Characteristics of Business Owners
2. Set URL variable equal to the corresponding url for the dataset wished to be called
    a. Company Summary: https://api.census.gov/data/2018/abscs.html
    b. Characteristics of Businesses: https://api.census.gov/data/2018/abscb.html

c. Characteristics of Business Owners: https://api.census.gov/data/2018/abscbo.html

d. Technology Characteristics of Businesses: https://api.census.gov/data/2018/abstcb.html

3. Set PARAMS dictionary equal to needed values for each dataset. Set ‘get’ equal to all columns. (NOTE: pull in all columns in this step, even if not planning to use them. This ensures all data comes through). Set ‘for’ equal to either state or us, depending on the scope for the dataset. Set ‘key’ equal to your API key.

   a. Company Summary:

```
PARAMS = {
    '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': 'state',
    'key' : '14289fd2fcc314628b968bb3c535f234569e6f9e'
}
```

   b. Characteristics of Businesses:

```
PARAMS = {
    '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,QDESC,QDESC_LABEL,BUSCHAR,BUSCHAR_LABEL,YEAR,FIRMPDEMP,FIRMPDEMP_F,FIRMPDEMP_PCT,FIRMPDEMP_PCT_F,RCPPDEMP,RCPPDEMP_F,RCPPDEMP_PCT,RCPPDEMP_PCT_F,EMP,EMP_F,EMP_PCT,EMP_PCT_F,PAYANN,PAYANN_F,PAYANN_PCT,PAYANN_PCT_F,FIRMPDEMP_S,FIRMPDEMP_S_F,FIRMPDEMP_PCT_S,FIRMPDEMP_PCT_S_F,RCPPDEMP_S,RCPPDEMP_S_F,RCPPDEMP_PCT_S,RCPPDEMP_PCT_S_F,EMP_S,EMP_S_F,EMP_PCT_S,EMP_PCT_S_F,PAYANN_S,PAYANN_S_F,PAYANN_PCT_S,PAYANN_S_F',
    'for': 'us',
    'key' : '14289fd2fcc314628b968bb3c535f234569e6f9e'
}
```

   c. Characteristics of Business Owners:

```
PARAMS = {
    'get':
'GEO_ID,NAME,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,OWNCHAR_LABEL,YEAR,OWNPDEMP,OWNPDEMP_F,OWNPDEMP_PCT,OWNPDEMP_PCT_F,OWNPDEMP_S,OWNPDEMP_S_F,OWNPDEMP_PCT_S,OWNPDEMP_PCT_S_F',
    'for': 'us',
    'key' : '14289fd2fcc314628b968bb3c535f234569e6f9e'
}
```

d. Technology Characteristics of Businesses:

```
a. PARAMS = {
b.     'get':
   'GEO_ID,NAME,NAICS2017,NAICS2017_LABEL,SEX,SEX_LABEL,ETH_GROUP,ETH_GROUP_LABE
   L,RACE_GROUP,RACE_GROUP_LABEL,VET_GROUP,VET_GROUP_LABEL,NSFSZFI,NSFSZFI_LABEL
   ,FACTORS_P,FACTORS_P_LABEL,YEAR,FIRMPDEMP,FIRMPDEMP_F,FIRMPDEMP_PCT,FIRMPDEMP
   _PCT_F,RCPPDEMP,RCPPDEMP_F,RCPPDEMP_PCT,RCPPDEMP_PCT_F,EMP,EMP_F,EMP_PCT,EMP_
   PCT_F,PAYANN,PAYANN_F,PAYANN_PCT,PAYANN_PCT_F,FIRMPDEMP_S,FIRMPDEMP_S_F,FIRMP
   DEMP_PCT_S,FIRMPDEMP_PCT_S_F,RCPPDEMP_S,RCPPDEMP_S_F,RCPPDEMP_PCT_S,RCPPDEMP_
   PCT_S_F,EMP_S,EMP_S_F,EMP_PCT_S,EMP_PCT_S_F,PAYANN_S,PAYANN_S_F,PAYANN_PCT_S,
   PAYANN_PCT_S_F',
c.     'for': 'us',
d.     'key' : '14289fd2fcc314628b968bb3c535f234569e6f9e'
e. }
```

4. Use requests to pull in data using URL variable and PARAMS dictionary.

```
a. r = requests.get(url = URL, params = PARAMS)
```

5. Use JSON to read data into json format

```
a. data = r.json()
```

6. Use pandas to change json into dataframe, making first row column names and rest of the row's column data

7. Set dataset equal to df_company, df_business, df_owner, df_technology respectively.

```
a. df_owner = pd.DataFrame(data[1:], columns=data[0])
```

8. Be sure to repeat this process until you have all four datasets in as data frames!

Transformation:

For the transformation process, we mainly focused on making the datasets more user friendly. We did not need to replace any NA values as there were not any in the columns we focused on. We dropped unneeded columns, renamed columns, and changed column types.

1. Drop the columns not needed to investigate our questions.
   a. Company Summary:

```
df_company = df_company.drop(['NAICS2017', 'SEX', 'ETH_GROUP', 'RACE_GROUP',
'VET_GROUP', 'EMPSZFI', 'YEAR', 'FIRMPDEMP_F','FIRMPDEMP_S', 'FIRMPDEMP_S_F',
'RCPPDEMP_S', 'RCPPDEMP_F','RCPPDEMP_S_F', 'EMP_F','EMP_S', 'EMP_S_F',
'PAYANN_F','PAYANN_S', 'PAYANN_S_F', 'state'],axis=1)
```

   b. Characteristics of Businesses:

```
df_business = df_business.drop(['NAME', 'NAICS2017', 'SEX', 'ETH_GROUP',
'RACE_GROUP', 'VET_GROUP_LABEL', 'QDESC', 'BUSCHAR', 'YEAR', 'FIRMPDEMP_F',
'FIRMPDEMP_PCT_F', 'RCPPDEMP_F', 'RCPPDEMP_PCT_F', 'EMP_F', 'EMP_PCT_F',
'PAYANN_F', 'PAYANN_PCT_F', 'FIRMPDEMP_S', 'FIRMPDEMP_S_F', 'FIRMPDEMP_PCT_S',
'FIRMPDEMP_PCT_S_F', 'RCPPDEMP_S', 'RCPPDEMP_S_F', 'RCPPDEMP_PCT_S',
'RCPPDEMP_PCT_S_F', 'EMP_S', 'EMP_S_F', 'EMP_PCT_S', 'EMP_PCT_S_F', 'PAYANN_S',
'PAYANN_S_F', 'PAYANN_PCT_S', 'PAYANN_S_F', 'PAYANN_PCT_S', 'PAYANN_S_F', 'us'],
axis=1)
```

c. Characteristics of Business Owners:

```
df_owner = df_owner.drop(['NAME', 'NAICS2017', 'OWNER_SEX', 'OWNER_ETH',
'OWNER_RACE', 'OWNER_VET', 'QDESC', 'OWNCHAR', 'YEAR', 'us', 'OWNPDEMP_F',
'OWNPDEMP_PCT_F', 'OWNPDEMP_S', 'OWNPDEMP_S_F', 'OWNPDEMP_PCT_S',
'OWNPDEMP_PCT_S_F'], axis=1)
```

d. Technology Characteristics of Businesses:

```
df_technology =
df_technology.drop(["NAME","NAICS2017","SEX","ETH_GROUP","RACE_GROUP","VET_GROUP"
,"NSFSZFI","FACTORS_P","YEAR","FIRMPDEMP_F","FIRMPDEMP_PCT_F","RCPPDEMP_F","RCPPD
EMP_PCT_F","EMP_F","EMP_PCT_F","PAYANN_F","PAYANN_PCT_F","FIRMPDEMP_S","FIRMPDEMP
_S_F","FIRMPDEMP_PCT_S","FIRMPDEMP_PCT_S_F","RCPPDEMP_S","RCPPDEMP_S_F","RCPPDEMP
_PCT_S","RCPPDEMP_PCT_S_F","EMP_S","EMP_S_F","EMP_PCT_S","EMP_PCT_S_F","PAYANN_S"
,"PAYANN_S_F","PAYANN_PCT_S","PAYANN_PCT_S_F","us"], axis=1)
```

2. Rename the columns to user-friendly names that are not random acronyms.
   a. Company Summary:

```
df_company = df_company.rename(columns={'NAICS2017_LABEL': 'Industry', 'NAME':
'State', 'SEX_LABEL': 'Gender', 'ETH_GROUP_LABEL': 'Ethnicity',
'RACE_GROUP_LABEL': 'Race', 'VET_GROUP_LABEL': 'Veteran', 'EMPSZFI_LABEL':
'FirmSize', 'FIRMPDEMP': 'NumberFirms', 'RCPPDEMP': 'FirmSales',
'EMP':'NumberEmployees', 'PAYANN': 'AnnualPayroll'})
```

b. Characteristics of Businesses:

```
df_business = df_business.rename(columns={'NAICS2017_LABEL’:
‘Industru’,’SEX_LABEL': 'Sex', 'ETH_GROUP_LABEL': 'Ethnicity',
'RACE_GROUP_LABEL': 'Race', 'VET_GROUP': 'VetStatus', 'QDESC_LABEL': 'Question',
'BUSCHAR_LABEL': 'BusinessCharacteristic', 'FIRMPDEMP': 'NumberFirms',
'FIRMPDEMP_PCT': 'PercentFirms', 'RCPPDEMP': 'FirmSales', 'RCPPDEMP_PCT':
'PercentSales', 'EMP': 'NumberEmployees', 'EMP_PCT': 'PercentEmployees',
'PAYANN': 'AnnualPayroll', 'PAYANN_PCT': 'PercentPayroll'})
```

c. Characteristics of Business Owners:

```
df_owner = df_owner.rename(columns={'NAICS2017_LABEL': 'Industry',
'OWNER_SEX_LABEL': 'OwnerGender', 'OWNER_ETH_LABEL':'OwnerEthnicity',
'OWNER_RACE_LABEL':'OwnerRace', 'OWNER_VET_LABEL': 'OwnerVet',
'QDESC_LABEL':'OwnerQuestion', 'OWNCHAR_LABEL':'OwnerCharacteristic',
'OWNPDEMP':'NumberOwners', 'OWNPDEMP_PCT': 'PercentOwners'})
```

   d. Technology Characteristics of Businesses:

```
df_technology = df_technology.rename(columns={'NAICS2017_LABEL': 'Industry',
'SEX_LABEL': 'Gender', 'ETH_GROUP_LABEL': 'Ethnicity', 'RACE_GROUP_LABEL':
'Race', 'VET_GROUP_LABEL': 'VeteranStatus', 'NSFSZFI_LABEL': 'EmploymentSize',
'FACTORS_P_LABEL': 'FactorsAntiTech', 'FIRMPDEMP': 'NumberFirms',
'FIRMPDEMP_PCT': 'PercentFirms', 'RCPPDEMP': 'FirmSales', 'RCPPDEMP_PCT':
'SalesPercent', 'EMP': 'NumberEmployees', 'EMP_PCT': 'PercentEmployees',
'PAYANN': 'AnnualPayroll', 'PAYANN_PCT': 'PercentPayroll'})
```

3. Change the data types of some columns to integer or float columns.
   a. Company Summary:

```
df_company = df_company.astype({'NumberFirms': int, 'FirmSales': int,
'NumberEmployees': int})
```

   b. Characteristics of Businesses:

```
df_business = df_business.astype({'NumberFirms': int, 'PercentFirms': float,
'FirmSales': float, 'PercentSales': float, 'NumberEmployees': int,
'PercentEmployees': float, 'AnnualPayroll': int, 'PercentPayroll': float})
```

   c. Characteristics of Business Owners:

```
df_owner = df_owner.astype({'NumberOwners': int, 'PercentOwners': float})
```

   d. Technology Characteristics of Businesses:

```
df_technology = df_technology.astype({'NumberFirms': int, 'PercentFirms': float,
'FirmSales': float, 'SalesPercent': float, 'NumberEmployees': int,
'PercentEmployees': float, 'AnnualPayroll': float, 'PercentPayroll': float})
```

4. Merge two datasets df_owner and df_business for visualizations:
   a. First further transform df_owner and df_business so they can be inner merged on the NCAIS2017_LABEL (now known as Industry) column.
   b. df_owner:
      i. Set OwnerQuestion only equal to question on highest education level. Set new filtered dataset to new name with copy() at end

```
df_owner_grouped = df_owner[df_owner.OwnerQuestion == 'EDUC'].copy()
```

      ii. Group by Industry and OwnerCharacteristic columns and sum NumberOwners column

```
df_owner_grouped = df_owner_grouped.groupby(['Industry',
'OwnerCharacteristic'])['NumberOwners'].sum()
```

```
df_owner_grouped = df_owner_grouped.reset_index()
```

iii. Set OwnerCharacteristic equal to all values except not ‘Total reporting’ and ‘Item not reported as these do not show any education level

```
df_owner_grouped = df_owner_grouped [df_owner_grouped.OwnerCharacteristic !=
'Total reporting']
df_owner_grouped = df_owner_grouped[df_owner_grouped.OwnerCharacteristic != 'Item
not reported']
```

iv. Create and add a new column called PercentOwners that divides NumberOwner column by the sum of owners for each industry and multiplies by 100. This gives the percent of owners at each education level for each industry. Use this new df_owner dataset for visualizations.

```
df_owner_grouped ['PercentOwners'] = (df_owner_grouped ['NumberOwners'] /
df_owner_grouped.groupby('Industry')['NumberOwners'].transform('sum')) * 100
```

v. Create a ba_above named list of OwnerCharacteristic values that represent bachelor’s degree attainment and above

```
ba_above = ['Bachelor\'s degree', 'Doctorate degree', 'Master\'s degree',
'Professional degree beyond a bachelor\'s degree']
```

vi. Filter df_owner so that it only contains OwnerCharacteristic values within the ba_above list. Name filtered dataset df_owner_ba_above and add copy() at end to make this a new separate dataset

```
df_owner_ba_above = df_owner_grouped [df_owner_grouped
['OwnerCharacteristic'].isin(ba_above)].copy()
```

vii. Create and add a new column called SumPercentOwners that groups by Industry and PercentOwners and adds the PercentOwners values of each industry together for a total percentage of owners with a bachelor’s degree or higher

```
df_owner_ba_above['SumPercentOwners'] =
df_owner_ba_above.groupby('Industry')['PercentOwners'].transform('sum')
```

viii. Reset value of df_owner_ba_above to only be equal to values where OwnerCharacteristic is equal to Bachelor’s degree. Since we are only concerned with SumPercentOwners (which is the same for every education level at each industry), it does not particularly matter which OwnerCharacteristic you choose to filter by.

```
df_owner_ba_above = df_owner_ba_above[df_owner_ba_above['OwnerCharacteristic'] ==
'Bachelor\'s degree']
```

ix. Now the dataset should only have one row per Industry with five columns: Industry, OwnerCharacteristic, NumberOwners, PercentOwners, SumPercentOwners. Set this dataset equal to only the following two columns needed: Industry, SumPercentOwners

```
df_owner_ba_above = df_owner_ba_above[['Industry', 'SumPercentOwners']]
```

x. Also drop unneeded industry rows that do not focus on specific industries from the dataset

```
df_owner_ba_above = df_owner_ba_above[df_owner_ba_above['Industry'] != 'Total for all sectors']
df_owner_ba_above = df_owner_ba_above[df_owner_ba_above['Industry'] != 'Industries not classified']
```

c. Whew! That is it for df_owner. Now time for df_business:

i. Group df_business by Industry and sum AnnualPayroll column. Set this equal to df_business_merge and add .copy() at the end to create separate dataset. (Ideally we would choose to sum by FirmSales, but this value is annoyingly not broken down by industry).

```
df_business_merge = df_business.groupby(['Industry'], as_index=False)['AnnualPayroll'].sum().copy()
```

ii. Remove unneeded industry rows that do not focus on a specific industry

```
df_business_merge = df_business_merge[df_business_merge['Industry'] != 'Total for all sectors']
df_business_merge = df_business_merge[df_business_merge['Industry'] != 'Industries not classified']
```

iii. All done with df_business! Both are ready for merge.

d. Now it is time to merge our two further transformed datasets: df_owner_ba_above and df_business_merge

e. Merge df_owner_ba_above with df_business_merge as an inner join on the column Industry. Name this new dataset df_owner_business_merge

```
df_owner_business_merge = df_owner_ba_above.merge(df_business_merge_1, how='inner', on='Industry')
```

f. Merge complete! This new dataset can now be used to create visualizations that relate owner education level to Annual Payrolls of industries

g.

Load: Because we created visualizations using pandas, we did not load these cleaned data sets to any other platform or program.

<u>Conclusion:</u>

For our ETL process, we focused most of our energy on dropping unnecessary columns and changing the names of columns to understandable titles rather than vague acronyms. We also took many steps to transform the Owner and Business datasets in order to merge them. By following these steps exactly as outlined, other groups can replicate our ETL process to produce similar results. Now that the data is successfully transformed, we can use it to create visualizations that answer the following questions: How does worker pay and employment type vary by owner race and gender? How does veteran business ownership vary by gender, race, and service-disability status? What are the trends in technology use across businesses? How does owner education attainment relate to annual pay within different industries?