Gamilo ETL:

Extract:

We want to drill into the demographic information of employees and owners, and potentially find relationships between the two.

1. To answer this question, we needed to make a few calls to the API.
    a. We got the RACE_GROUP, RACE_GROUP_LABEL, NAICS2017, NAICS2017_LABEL, and EMP variables from the ABS Company Summary data.
    b. We got the OWNER_RACE, OWNER_RACE_LABEL, OWNPDEMP, NAICS2017, and NAICS2017_LABEL variables from the ABS Business Owner data.
    c. We got the RACE_GROUP, RACE_GROUP_LABEL, and EMP variables, but this time aggregated by state, from the ABS Company Summary data.
    d. The states are represented by codes. Fortunately, we can get the data dictionary for these codes from the census website itself: https://www.census.gov/library/reference/code-lists/ansi/ansi-codes-for-states.html
2. We applied the .json() method to the response and stored it in a variable. This will coerce the data into an array that pd.DataFrame class can interpret.

Transform/Load

Since we needed to do our work in pandas, we loaded the dataframes into pandas first and then applied the transformations. As this was census data taken directly from a government source, we did not need to do much cleaning.

1. We loaded in each of the API queries mentioned above as pandas dataframes, using the pd.Dataframe() class. The column headers are represented by the first index of the array, so we stored the first index, loaded the rest of the array, and then assigned the columns to that first index.
2. We then filtered out aggregate data (Total Races, Total of Industry, etc.) by filtering the DataFrames where the race codes are not equal to 00, 90, 91, 92, as well as filtering out NAICS Codes equal to 00.
3. The numeric values (EMP, OWNPDEMP) were represented as string types in the DataFrames. To accommodate for that, we recast them as integers using the .astype(int) method. We also cast the state codes as integers to avoid having to deal with padded zeros.
4. In regards to merging, we merged the first two queries (Race of Owners, Race of Employees) by using an inner join with left on as [RACE_GROUP_LABEL, NAICS2017] and right on as [OWNER_RACE_LABEL, NAICS2017]. For the state data, we merged the 3rd query by using an inner join with left on as [state] and right on as [STATE].
5. To make plotting easier, each table was converted to a PivotTable with values as the numeric columns, the index as the desired rows (NAICS_2017_LABEL for the first and second, and STATE_NAME for the third) and the RACE_GROUP_LABEL as the desired columns. Aggregation was on np.sum.
6. For the first two queries (Race of Owners, Race of Employees) now represented as PivotTables, we used a seaborn barplot and a for loop to plot the subplots.

7. For the merge of the first two queries, we used two plots: a seaborn jointplot with x = OWNPDEMP and y = EMP and kind = 'reg' as well as a seaborn relplot with the same x and y, but with OWNER_RACE_LABEL as the hue to distinguish between the elements of the scatterplot
8. For the third query (Race of Employees by State) we also used a seaborn barplot and a for loop to plot the subplots.

Atkinson ETL:

General Cleaning:

1. Import the libraries you will need for querying the API, data analysis, and visualization: Requests, JSON, Pandas, Numpy, MatPlotLib, and Seaborn
2. Import your API key from your config.py file
3. Set up for API request using the Census URL containing the information you will need for your analysis. For my purposes, I used the URL: https://api.census.gov/data/2018/abscs?get=EMP,SEX,SEX_LABEL,RACE_GROUP,RACE_GROUP_LABEL,ETH_GROUP,ETH_GROUP_LABEL,YIBSZFI,YIBSZFI_LABEL,FIRMPDEMP&for=us:*&key={api_key}
4. Run the request, and then cast your result as a Pandas data frame.
5. The column names are located in row 0. Indicate that these are the column names, and then drop row 0.
6. Cast the 'EMP', 'YIBSZFI', and 'FIRMPDEMP columns as integers
7. Sort the 'YIBSZFI' column by value; this will help in creating the chart.
8. Create a new dataframe with the columns 'EMP', 'SEX_LABEL', 'RACE_GROUP_LABEL', 'ETH_GROUP_LABEL', 'YIBSZFI', 'YIBSZFI_LABEL', and 'FIRMPDEMP'.

Pie Chart:

1. Create a data frame with the columns 'YIBSZFI_LABEL', and 'FIRMPDEMP'.
2. Filter out anything with the value 'All firms' from the 'YIBSZFI_LABEL' column
3. Create a pivot table with the index as 'YIBSZFI_LABEL' and the values as 'FIRMPDEMP'.
4. Use this pivot table data frame to create a pie chart showing the number of firms represented by years in business.
5. Repeat this process with the columns 'YIBSZFI_LABEL', and 'EMP'.

Stacked Bar Chart by Race:

1. Create a data frame with the columns 'EMP', 'RACE_GROUP_LABEL', 'YIBSZFI', and 'YIBSZFI_LABEL'.
2. Filter out anything with the value 'All firms' from the 'YIBSZFI_LABEL' column
3. Filter out anything with the value 'Total', 'Classifiable', 'Unclassifiable', 'Minority', and 'Nonminority' from the 'RACE_GROUP_LABEL' column.
4. Create a pivot table with 'YIBSZFI' as the index, 'RACE_GROUP_LABEL' as the columns, and 'EMP' as the values.
5. Sort the 'YIBSZFI' column in descending order.
6. Create a column in the pivot table that totals across rows.
7. Create a new column for each race category that calculates the percent of that race group represented based on the total number of employees in that sector.

8. Drop the non-percent rows from the pivot table.
9. Create a horizontal stacked bar chart and then replace y tick labels with the correlating label to the value indicated.
10. Drop the 'White %' column from the pivot table.
11. Create a horizontal stacked bar chart and then replace y tick labels with the correlating label to the value indicated.

Stacked Bar Chart by Sex:

1. Create a data frame with the columns 'EMP', 'SEX_LABEL', 'YIBSZFI', and 'YIBSZFI_LABEL'.
2. Filter out anything with the value 'All firms' from the 'YIBSZFI_LABEL' column
3. Filter out anything with the value 'Total' from the 'SEX_LABEL' column.
4. Create a pivot table with 'YIBSZFI' as the index, 'SEX_LABEL' as the columns, and 'EMP' as the values.
5. Sort the 'YIBSZFI' column in descending order.
6. Create a column in the pivot table that totals across rows.
7. Create a new column for each race category that calculates the percent of that sex represented based on the total number of employees in that sector.
8. Drop the non-percent rows from the pivot table.
9. Create a horizontal stacked bar chart and then replace y tick labels with the correlating label to the value indicated.

Thomas ETL:

Extract:

1. From the US Census website, request an API Key using the following URL: https://api.census.gov/data/key_signup.html
2. Store the API key as a variable in a config.py file and use the activated API key in any API queries to ensure that data is being accessed legally.
3. Make a request to the API to pull any relevant data. For the purposes of these visualizations, I used: https://api.census.gov/data/2018/abstcb?get=GEO_ID,NAME,EMP,RACE_GROUP,RACE_GROUP_LABEL,FIRMPDEMP,IMPACTWF_U,IMPACTWF_U_LABEL,TECHUSE,TECHUSE_LABEL&for=us:*&NAICS2017=00&key={api_key}
4. Store the response of the API request in a variable, and then open that variable in a .json()
5. Create a DataFrame out of the json variable created in step 4.

Transform:

1. In the created DataFrame, we replace the columns with the contents of the first row using columns = .loc[0] and then drop the [0] row to remove duplicate data.
2. Cast the Employee column as an integer type so that we will be able to plot the data later.
3. Replace the IMPACTWF_U_LABEL column to only contain values related to Specialized Software by storing a list in a variable.
4. Replace the RACE_GROUP_LABEL to remove the aggregation categories. Race should exclude Total, Minority, and Nonminority.

5. Replace TECHUSE_LABEL column to only contain values related to Specialized Software: Moderate Use and Specialized Software: High Use.

Load:

1. For Horizontal Bar Chart, set x as 'RACE_GROUP_LABEL' and set y as 'EMP'. Set ticks starting from 0 and increase by 10,000,000 each tick until 60,000,000
2. For Vertical Bar Chart, filter the 'IMPACTWF_U_LABEL' column further to only show 'Specialized Software: Increased skill level of workers employed by this business'. Set x as 'RACE_GROUP_LABEL' and set y as 'EMP'. Set ticks starting from 0 and increase by 1,000,000 each tick until 8,000,000.
3. For Donut Charts, first create two individual pie charts of the RACE_GROUP_LABEL column category with the value as 'EMP'. The first pie chart is TECH_USE_LABEL 'Specialized Software: Moderate Use', and the second pie chart is 'Specialized Software: High Use'. Finally, using the race groups and percentages generated by the pie charts, create a plotly donut chart with corresponding subplots.

<u>Carlson ETL process:</u>

1. Created a variable that contains a list of the columns that I wanted access to & a function that formats it properly within a url
2. Created a variable that contains the url, api key, and the function/variable referenced in step 1
3. Made a request to the API to pull the data & saved it in a variable as a dataframe
4. Set the headers equal to the first row and subsequently dropped the first row, which was an exact copy of the headers
5. Cast the values in column OWNPDEMP which contains a numeric value as integer values
6. Renamed the field "Before 1980" as "1980 & Earlier" in the column "OWNCHAR_LABEL" for easy sorting
7. Dropped the rows with fields equal to "Don't know" and "Item not reported" in the column "OWNCHAR_LABEL"
8. Created a dataframe containing only specific race fields in the column "OWNER_RACE_LABEL" by dropping rows with values of "All owners of respondent firms," "Minority," and "Nonminority," and excluding total values by excluding rows with the value "CO" in the column "OWNCHAR." (Stored in a variable)
9. Excluded rows containing the value "Total reporting" in the field "OWNCHAR_LABEL" in the main dataframe
10. Created a dataframe containing total aggregated data for specific year range fields by only keeping rows with the value "00" in the column "OWNER_RACE"
11. Used the dataframe created in step 10 to create a plot showing the distribution of business owners based on the year they acquired their business
12. Used the dataframe created in step 8 to create a plot showing the distribution of business owners based on their race

13. Created two separate dataframes using the main dataframe; one containing only data for the field “Minority” in OWNCHAR_LABEL, and one containing only data for the field “Nonminority” in the same column
    a. Dropped irrelevant columns; kept only columns related to year of acquisition & number of business owners
    b. Renamed the OWNPDEMP column to reflect which demographic was reflected in the numbers, for each new dataframe (Minority and Nonminority business owners)
    c. Merged the two dataframes on the year of acquisition
    d. Created a grouped bar plot showing the number of Minority business owners relative to the number of Nonminority business owners
14. Followed the same steps in 13 to create the same type of visualization, but showing all five main race categories by year