ETL Report

Census API Data - Annual Business Survey 2019

M08: Pandas and Visualizations

Dev10 - Group 4

Sargis Abrahamyan, Justin Bartell, Ben Hines, Lindsey Oh

17 January 2022

Introduction

We wanted to show the greatest predictors of business success in the United States. Our data exclusively comes from the United States Census Bureau’s 2017-2019 Annual Business Survey. This API has an extensive querying ability, making extraction more complex while leaving transformation much simpler.

Data Sources

The 2017-2019 Annual Business Survey has four different datasets: Company Summary (ABSCS), Characteristics of Businesses (ABSCB), Characteristics of Business Owners (ABSCBO), Technology Characteristics of Businesses (ABSTCB).

APA Citation:

U.S. Government (2021, October 14). *Annual Business Survey (ABS) APIs*. United States Census Bureau. Retrieved January 14, 2022, from https://www.census.gov/data/developers/data-sets/abs.html

Extraction

abscs (Average Annual Pay Per Employee by State)

1. Import pandas, requests, json, plotly.graph_objs
2. Read in “state_abbrv.csv” as a dataframe, to map state names with state abbreviations.
3. Create variables for each part of the URL which changes the API request: dataset name (abscs), parameters (NAME,PAYANN,EMP), location (state:*), key.
4. Create an empty list to hold data from 3 different years (2017-2019)
5. Create a for loop and iterate over three years 2017-2019
6. For each year
   a. Create a variable to store the f-string for the URL.
   b. Create a variable to store the requested data from the URL as a string. The requested data is in JSON format.
   c. Pop the first element from the API data.
   d. Create a dataframe with data from step 6b and headers from step 6c
   e. Append the dataframe to the list created in step 4

abscs (Industry vs Annual Pay Per Employee)

1. Import pandas, json, requests, and matplotlib
2. Create variables for each part of the URL which changes the API request: dataset name (abscs), year (2017), parameters (NAICS2017,NAICS2017_LABEL,PAYANN,EMP), location (us:*), key.
3. Create variable for the f-string for URL.
4. Create a variable to store the requested data from the URL as a string. The data requested data is in JSON format. Cast the JSON data as a list.

5. Pop out header and store as variable.
6. Store as a dataframe the request response from the URL with columns set as header variable created in step 5.
7. Repeat steps 1 – 6 for years 2018 and 2019.

abscb (Owner Demographics vs Employee Pay)

1. Import pandas, requests, json.
2. Create variables for each part of the URL which changes the API request: year (2019), dataset name (abscb), parameters (SEX, SEX_LABEL, PAYANN, PAYANN_S, EMP, EMP_S), location (us:*), key.
3. Create a variable to store the f-string for the URL.
4. Create a variable to store the requested data from the URL as a string. The data requested data is in JSON format. Cast the JSON data as a list.
5. Create a variable to store the popped header list.
6. Create a variable to store the data frame created from the list in step 4. Set columns equal to the list created in step 5.
7. Repeat steps 1-6 for years 2018 and 2019.
8. Repeat steps 1-7 for Ethnicity (ETH_GROUP, ETH_GROUP_LABEL), Race (RACE_GROUP, RACE_GROUP_LABEL), Veteran Status (VET_GROUP, VET_GROUP_LABEL), and the other business ownership types (QDESC,QDESC_LABEL,BUSCHAR,BUSCHAR_LABEL).

abscbo (Owner Demographic Data: Sex, Ethnicity, Race, Veteran Status)

1. Import pandas, requests, json.
2. Create variables for each part of the URL which changes the API request: year (2017), dataset name (abscbo), parameters (OWNER_SEX, OWNER_SEX_LABEL, OWNPDEMP), location (us:*), key.
3. Create a variable to store the f-string for the URL.
4. Create a variable to store the requested data from the URL as a string. The data requested data is in JSON format. Cast the JSON data as a list.
5. Create a variable to store the popped header list.
6. Create a variable to store the data frame created from the list in step 4. Set columns equal to the list created in step 5.
7. Repeat steps 1-6 for years 2018 and 2019.
8. Repeat steps 1-7 for Ethnicity, Race, Veteran Status.

Transformation

abscb (Owner Demographics vs Employee Pay: Sex, Ethnicity, Veteran Status)

1. Convert the ‘PAYANN’ and the ‘EMP’ columns to numeric.
2. Create a new column named ‘Average Annual Employee Pay’ that divides the ‘PAYANN’ column by the ‘EMP’ column
3. Drop the ‘PAYANN’, ‘EMP’, and ‘us’ columns
4. Remove any columns that have ‘Total’ in a row
5. Set the index to the ‘SEX_LABEL’, ‘ETH_GROUP_LABEL’, or ‘VET_GROUP_LABEL’--respectively

abscb (Owner Demographics vs Employee Pay: Race)

1. Only include the rows where RACE_GROUP is equal to ‘90’, ‘91’, or ‘92’ (minority, equally minority/nonminority, or nonminority)
2. Convert the ‘PAYANN’ and the ‘EMP’ columns to numeric.
3. Create a new column named ‘Average Annual Employee Pay’ that divides the ‘PAYANN’ column by the ‘EMP’ column
4. Drop the ‘PAYANN’, ‘EMP’, and ‘us’ columns
5. Remove any columns that have ‘Total’ in a row
6. Set the index to the ‘RACE_GROUP_LABEL’

abscb (Owner Demographics vs Employee Pay: Owner Number, Family-Owned, Spouses)

1. Remove any rows with BUSCHAR in ['A1','BP','NB','BO','BZ','BY','ME','MF','MD','BX','BQ04']
2. Create three DataFrames where the QDESC is equal to either ‘B01’, ‘B02’, ‘B03’. Do steps 3-7 for each dataframe.
3. Convert the ‘PAYANN’ and the ‘EMP’ columns to numeric.
4. Create a new column named ‘Average Annual Employee Pay’ that divides the ‘PAYANN’ column by the ‘EMP’ column
5. Drop the ‘PAYANN’, ‘EMP’, and ‘us’ columns
6. Remove any columns that have ‘Total’ in a row
7. Set the index to ‘BUSCHAR_LABEL’

abscs (Average Annual Pay Per Employee by State)

1. Create an empty list
2. Create a variable ‘year’, set it to 2017
3. Create a for loop to iterate over the list created in step 4(extraction)
    a. Inner merge each dataframe with state_abbrv dataframe on ‘State’ on the right and ‘NAME’ on the left
    b. Drop columns ‘NAME’, ‘state’
    c. Add a new column named f‘Annual Pay Per Emoloyee ({year})’
        i. Set it to be PAYANN / EMP * 1000 (convert to numeric)
    d. Drop columns ‘PAYANN’, ‘EMP’, ‘State’
    e. Append the dataframe to the list created in step 1
    f. Increment year variable
4. Inner merge the datasets in the list from step 1 on ‘State_abbr’

5. Create a new column and set it to be the average of the following three columns: Annual Pay Per Employee (2017)’ Annual Pay Per Employee (2018)’ Annual Pay Per Employee (2019)’
6. Create a new column set to be percentage difference between 'Annual Pay Per Employee (2017)’ and 'Annual Pay Per Employee (2019)’

abscbo (Owner Demographic Data: Sex, Ethnicity, Veteran Status)

1. Convert the OWNPDEMP column to numeric.
2. Create a variable for to store each of the following values: All owners of respondent firms, Female, Male.
3. Create a variable for each of the following values: percent Female, percent Male
4. Initialize a new column “2017 Percent of All Owners” and assign each row in this column the total (100) and respective percent.
5. Drop all columns except OWNER_SEX_LABEL, 2017 Percent of All Owners.
6. Repeat steps 1-5 for years 2018 and 2019.
7. Merge each of the three tables created in step 5 as an inner join on OWNER_SEX_LABEL.
8. Create a new data frame with columns “Year”, “Female”, “Male”. Manually fill in the rows for “Year”. Fill in the remaining values from the respective cells of the table created in step 7.
9. Repeat steps 1-8 for Ethnicity, Veteran Status.

abscbo (Owner Demographic Data: Race)

1. Convert the OWNPDEMP column to numeric.
2. Create a variable to store the list of races: “All owners of respondent firms”, “Minority”, “Nonminority”.
3. Keep the rows of the data frame whose value in the column “OWNER_RACE_LABEL” is in the list of races
4. Sort the data frame rows by alphabetical order according to the value in the column “OWNER_RACE_LABEL”.
5. Create a variable to store each of the following values: All owners of respondent firms, Minority, Nonminority.
6. Create a variable for each of the following values: percent Minority, percent Nonminority
7. Initialize a new column “2017 Percent of All Owners” and assign each row in this column the total (100) and respective percent.
8. Drop all columns except OWNER_RACE_LABEL, 2017 Percent of All Owners.
9. Repeat steps 1-5 for years 2018 and 2019.
10. Merge each of the three tables created in step 5 as an inner join on OWNER_RACE_LABEL.
11. Create a new data frame with columns “Year”, “Minority”, “Nonminority”. Manually fill in the rows for “Year”. Fill in the remaining values from the respective cells of the table created in step 10.

abscs (Industry vs Employee Pay)

1. Convert column ‘EMP’ to numeric.
2. Convert column ‘PAYANN’ to numeric.
3. Drop duplicates from column ‘NAICS2017_LABEL’
4. Create column ‘Average Pay ($1000)’ and assign it to the division of values in the ‘PAYANN’ column by the ‘EMP’ column.
5. Drop columns ‘us’, ‘EMP’, ‘PAYANN’, ‘NAICS2017’
6. Set index to ‘NAICS2017_LABEL’
7. Drop rows ‘Industries not classified’ and ‘Total for all sectors’.
8. Rename index ‘Securities, commodity contracts, and other financial investments and related activities’ with ‘Securities and other investments’.
9. Rename index ‘Securities and commodity contracts intermediation and brokerage’ with ‘Securities and contract intermediation''.
10. Rename index ‘Research and development in biotechnology (except nanobiotechnology)’ with ‘Biotech R&D’
11. Rename index ‘Sporting goods, hobby, musical instrument, and book stores’ with ‘Book and hobby stores’
12. Sort dataframe on the column ‘Average Pay ($1000)’
13. Get the 5 lowest paying industries by calling nsmallest() on the column ‘Average Pay ($1000)’
14. Get the 5 highest paying industries by calling nlargest() on the column ‘Average Pay ($1000)’

Load

If we were to load each of these DataFrames into a relational database:

1. Export each data as .csv.
2. Create a database and set it as active.
3. Import data via the import wizard from the .csv file. Set relevant data types and primary keys, and whether or not to allow nulls for the given columns.
4. Create the seven tables within the schema shown below with the relevant columns using the CREATE TABLE function.
5. Using the INSERT function, populate the tables with data from the denormalized table created from the .csv file.
6. Drop the denormalized table from the database.

Conclusion

The data was successfully extracted and transformed for this project’s purposes.