DSC-540 Project Milestone-4

Chitramoy Mukherjee

Apply 5 transformations to https://api.census.gov/data/2017/ecnclcust

In [127…
```
# Import modules
import matplotlib.pyplot as plt
import pandas as pd
import os
```

In [135…
```
 # Construct the full API link using the API_KEY received through key request

def generate_api_link(api_key, dataset, variables, geography):
    base_url = "https://api.census.gov/data"
    endpoint = f"{dataset}"
    params = {
        "get": f"{variables}",
        "for": f"{geography}",
        "key": api_key
    }

    api_link = f"{base_url}/{endpoint}?{'&'.join([f'{k}={v}' for k, v in params.items(

    return api_link

if __name__ == "__main__":
    census_api_key = "6f54e9fd4f7eef82cba525eb1d738c0da4048c66"
    census_dataset = "2017/ecnclcust"
    census_variables = "NAICS2017_LABEL,NAME,GEO_ID,TYPOP,TYPOP_LABEL,TAXSTAT_LABEL,TA
    census_geography = "state:*&NAICS2017"

    api_link = generate_api_link(census_api_key, census_dataset, census_variables, cer

    census_data = fetch_census_data(census_api_url)

    if census_data:

        for record in census_data[1:6]:  # Printing first 5 rows , skipping the header
            print(record)
    else:
        print("No data retrieved.")
```

```
['Legal services', 'Alaska', '0400000US02', '00', 'All establishments', 'Establishmen
ts subject to federal income tax', 'T', '270019', '5411', '02']
['Legal services', 'Alabama', '0400000US01', '00', 'All establishments', 'Establishme
nts subject to federal income tax', 'T', '2759330', '5411', '01']
['Legal services', 'Arkansas', '0400000US05', '00', 'All establishments', 'Establishm
ents subject to federal income tax', 'T', '0', '5411', '05']
['Legal services', 'Arizona', '0400000US04', '00', 'All establishments', 'Establishme
nts subject to federal income tax', 'T', '3485673', '5411', '04']
['Legal services', 'California', '0400000US06', '00', 'All establishments', 'Establis
hments subject to federal income tax', 'T', '44033660', '5411', '06']
```

Census Data API: Variables in /data/2017/ecnbasic/variables

NAICS2017_LABEL : Type of Business/Service

NAME : State Name

GEO_ID : Geographic identifier code

TYPOP : Type of operation code

TYPOP_LABEL : Wholesale Trade

TAXSTAT_LABEL : Wholesale Trad labels

TAXSTAT : Tax status code

RCPTOT : Sales, value of shipments, or revenue

NAICS2017 : 2017 NAICS code

state : State code

Ethical implecations on census API data

Census data is often used to allocate resources, determine political representation, and make policy decisions. There is an ethical obligation to ensure that the data collection and analysis processes are fair, unbiased, and do not disproportionately disadvantage specific communities or demographics.

Openness and transparency in the data collection and analysis processes are critical. The public should have access to information about how the census is conducted, what data is collected, and how it is used.

There is an ethical responsibility to collect accurate and reliable data. Inaccurate data can lead to incorrect policy decisions, misallocation of resources, and unfair treatment of certain groups or regions.

One of the primary ethical considerations is the protection of individual privacy. Census data often includes personal information, and there is a responsibility to ensure that the data is collected, stored, and used in a way that respects individuals' privacy rights.

In [136…

```
# Find duplicates based on 'NAICS2017_LABEL', 'NAME', 'GEO_ID','CLASSCUST_LABEL','CLAS

def fetch_census_data(api_url):
    response = requests.get(api_url)

    if response.status_code == 200:
        data = response.json()
        return data
    else:
        print(f"Error: {response.status_code}")
        return None
```

```
if __name__ == "__main__":
    census_api_url = "https://api.census.gov/data/2017/ecnclcust?get=NAICS2017_LABEL,N

    census_data = fetch_census_data(census_api_url)

    if census_data:
        df = pd.DataFrame(census_data[1:], columns=census_data[0])
# Select columns for finding duplicates
        selected_columns = ['NAICS2017_LABEL', 'NAME', 'GEO_ID', 'state']


        duplicates = df[df.duplicated(subset=selected_columns, keep=False)] # Find and

        if not duplicates.empty:
            print("Duplicate Records:")
            print(duplicates)
        else:
            print("No duplicates found.")
    else:
        print("No data retrieved.")
```

```
No duplicates found.
```

In [137…

```
# Check for Null value in key fields NAME and state column
# Make the API request
response = requests.get(url, params=params)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Parse the JSON response
    data = response.json()

    # Create a DataFrame from the API response
    df = pd.DataFrame(data[1:], columns=data[0])

    # Check for null values in 'NAME' and 'state' columns
    null_values_name = df['NAME'].isnull().sum()
    null_values_state = df['state'].isnull().sum()

    print(f"Null values in 'NAME': {null_values_name}")
    print(f"Null values in 'state': {null_values_state}")

else:
    print(f"Error: {response.status_code}")
    print(response.text)
```

```
Null values in 'NAME': 0
Null values in 'state': 0
```

In [138…

```
# Rename the NAME column as STATE_NAME and state AS STATE_CD and display data
# Make the API request
response = requests.get(url, params=params)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Parse the JSON response
    data = response.json()

    # Create a DataFrame from the API response
    df = pd.DataFrame(data[1:], columns=data[0])
```

```
    # Handle missing values by dropping rows with missing values
    df = df.dropna()

    # Rename columns
    df = df.rename(columns={'NAME': 'STATE_NAME', 'state': 'STATE_CD'})

    # Display the first 100 rows of the transformed DataFrame
    print(df.head())

else:
    print(f"Error: {response.status_code}")
    print(response.text)
```

```
  NAICS2017_LABEL STATE_NAME       GEO_ID TYPOP         TYPOP_LABEL  \
0  Legal services     Alaska  0400000US02    00  All establishments
1  Legal services     Alaska  0400000US02    00  All establishments
2  Legal services     Alaska  0400000US02    00  All establishments
3  Legal services     Alaska  0400000US02    00  All establishments
4  Legal services     Alaska  0400000US02    00  All establishments

                                TAXSTAT_LABEL TAXSTAT CLASSCUST  \
0  Establishments subject to federal income tax       T       001
1  Establishments subject to federal income tax       T      5045
2  Establishments subject to federal income tax       T       514
3  Establishments subject to federal income tax       T       515
4  Establishments subject to federal income tax       T       516

                     CLASSCUST_LABEL  RCPTOT NAICS2017 STATE_CD
0              All classes of customer  270019      5411       02
1  Household consumers and individuals       0      5411       02
2             Business firms and farms       0      5411       02
3                  Federal government       0      5411       02
4          State and local governments       0      5411       02
```

In [139…

```
# Convert numeric columns to appropriate types

# Make the API request
response = requests.get(url, params=params)

# Create a DataFrame from the API response
df = pd.DataFrame(data[1:], columns=data[0])

numeric_columns = ['TYPOP', 'TAXSTAT', 'RCPTOT']
df[numeric_columns] = df[numeric_columns].apply(pd.to_numeric, errors='coerce')


print("Summary Statistics:")# Display summary statistics to identify outliers
print(df.describe())
```

```
Summary Statistics:
       TYPOP  TAXSTAT        RCPTOT
count  305.0      0.0  3.050000e+02
mean     0.0      NaN  9.526400e+05
std      0.0      NaN  4.410963e+06
min      0.0      NaN  0.000000e+00
25%      0.0      NaN  0.000000e+00
50%      0.0      NaN  0.000000e+00
75%      0.0      NaN  0.000000e+00
max      0.0      NaN  4.656130e+07
```

In [134…

```python
# Make the API request
response = requests.get(url, params=params)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Parse the JSON response
    data = response.json()

# Create a DataFrame from the API response
    df = pd.DataFrame(data[1:], columns=data[0])

# Convert string columns to lowercase for consistency
    string_columns = ['NAICS2017_LABEL', 'NAME', 'TYPOP_LABEL', 'TAXSTAT_LABEL']
    df[string_columns] = df[string_columns].apply(lambda x: x.str.lower())

# Display the first few records of the DataFrame
    print(df.head())

else:
    print(f"Error: {response.status_code}")
    print(response.text)
```

```
  NAICS2017_LABEL    NAME       GEO_ID TYPOP         TYPOP_LABEL  \
0  legal services  alaska  0400000US02    00  all establishments
1  legal services  alaska  0400000US02    00  all establishments
2  legal services  alaska  0400000US02    00  all establishments
3  legal services  alaska  0400000US02    00  all establishments
4  legal services  alaska  0400000US02    00  all establishments

                                TAXSTAT_LABEL TAXSTAT CLASSCUST  \
0  establishments subject to federal income tax       T       001
1  establishments subject to federal income tax       T      5045
2  establishments subject to federal income tax       T       514
3  establishments subject to federal income tax       T       515
4  establishments subject to federal income tax       T       516

                     CLASSCUST_LABEL  RCPTOT NAICS2017 state
0              All classes of customer  270019      5411    02
1  Household consumers and individuals       0      5411    02
2             Business firms and farms       0      5411    02
3                  Federal government       0      5411    02
4          State and local governments       0      5411    02
```