ETL Report

Elsa Carlson, Kylee LaPierre, Tarick Mehanna, and Logan Sell
September 29, 2022

Introduction

Health insurance is a type of insurance that covers a portion of the cost of medical expenses. There is a portion of the population that is uninsured in the United States; understanding the factors behind lacking health insurance or sufficient access to healthcare could illuminate the way to improve health insurance coverage in the country. For the purpose of this work, those with *sufficient access to healthcare* include people who do not delay or avoid seeking healthcare services due to cost. *Uninsured* includes those who do not have health insurance.

We are interested in understanding the factors that lead to a lack of health insurance and lacking health insurance or sufficient access to healthcare nationwide and in the counties in Minnesota in 2019. We are looking into race and ethnicity, gender, age, income, immigration status, educational attainment, and location to see how these play a role in someone's insurance status.

Data Sources

1) Bureau, U. S. C. (2019). *CB1900CBP: All Sectors: County Business Patterns by Legal Form of Organization and Employment Size Class for U.S., States, and Selected Geographies: 2019*. Explore census data. Retrieved September 21, 2022, from https://data.census.gov/cedsci/table?q=hospital+cb&g=0100000US%240400000_0400000US27&tid=CBP2019.CB1900CBP

2) Bureau, U. S. C. (2019). *DP05: ACS Demographic and Housing Estimates*. Explore census data. Retrieved September 21, 2022, from https://data.census.gov/cedsci/table?q=DP05&g=0100000US%240400000&tid=ACSDP1Y2019.DP05

3) Bureau, U. S. C. (2019). *S2701: Selected Characteristics of Health Insurance Coverage in the United States*. Explore census data. Retrieved September 21, 2022, from https://data.census.gov/cedsci/table?q=health+insurance&g=0100000US%240500000_0400000US27&tid=ACSST1Y2019.S2701&moe=false

4) Bureau, U. S. C. (2019). *S2703: Private Health Insurance Coverage by Type and Selected Characteristics*. Explore census data. Retrieved September 21, 2022, from https://data.census.gov/cedsci/table?q=health+insurance&g=0400000US27%2C27%240500000&tid=ACSST1Y2019.S2703

5) Bureau, U. S. C. (2019). *S2704: Public Health Insurance Coverage by Type and Selected Characteristics*. Explore census data. Retrieved September 21, 2022, from https://data.census.gov/cedsci/table?q=health+insurance&g=0400000US27%2C27%240500000&tid=ACSST1Y2019.S2704

6) Bureau, U. S. C. (2022, August 10). *2008 - 2020 small area health insurance estimates (SAHIE) using the American Community Survey (ACS)*. Census.gov. Retrieved September 21, 2022, from https://www.census.gov/data/datasets/time-series/demo/sahie/estimates-acs.html *Only used the year 2019 from this website.*

7) CDC. (2021, April 5). *NHIS - 2019 NHIS*. Centers for Disease Control and Prevention. Retrieved September 21, 2022, from https://www.cdc.gov/nchs/nhis/2019nhis.htm

8) CDC. (2021, December 16). *Behavioral risk factors: Selected metropolitan area risk trends (SMART) MMSA prevalence data (2011 to present)*. Centers for Disease Control and Prevention. Retrieved September 21, 2022, from https://chronicdata.cdc.gov/Behavioral-Risk-Factors/Behavioral-Risk-Factors-Selected-Metropolitan-Area/j32a-sa6u?category=Behavioral-Risk-Factors&view_name=Behavioral-Risk-Factors-Selected-Metropolitan-Area

9) CDC. (2021, September 14). *BRFSS: Table of health care access/coverage*. Centers for Disease Control and Prevention. Retrieved September 21, 2022, from https://chronicdata.cdc.gov/Behavioral-Risk-Factors/BRFSS-Table-of-Health-Care-Access-Coverage/f7a2-7inb

10) Prescott, R. (2016, June 22). *USA-cities-and-states/US_CITIES_STATES_COUNTIES.CSV at master · Grammakov/USA-Cities-and-States*. GitHub. Retrieved September 26, 2022, from https://github.com/grammakov/USA-cities-and-states/blob/master/us_cities_states_counties.csv

- This source was utilized for the SAHIE dataset for a list of all counties in the United States.

Extraction

All datasets except for the BRFSS Coverage survey results were formatted as CSVs, downloaded from the sources listed above. All CSVs were uploaded into the cohort's data lake & read into a databrick as Spark dataframes.

While exporting the BRFSS SMART dataset (Data Source #8), we filtered to include only data for the state of Minnesota & the year 2019.

We accessed the BRFSS Coverage data (Data Source #9) using an API, using the following steps:

1. We made an account with the CDC to use an API.
2. Then we created an app token, as required for the use of this API
3. We used the app token and the account credentials to call the API with Socrata, specifying a maximum number of rows of 158,720 to ensure all rows were included.
4. We read the BRFSS dataset into a databrick as a Pandas dataframe.

Details on the cleaning & transformation for each dataset are explained in the following section.

<u>Transformation</u>

We used two databricks in the transformation process: one for general cleaning of the data & one for transforming the data into a normalized format so that it could be loaded into our SQL database.

In our two databricks, we imported the following:

```python
import requests
import pyspark.sql.functions
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
from pyspark.sql.functions import *
from pyspark.sql.types import StringType,DecimalType
from pyspark.sql.functions import input_file_name, substring
from pyspark.sql.functions import isnan, when, count, col
from sodapy import Socrata
```

Initial cleaning:
The following steps were completed in the “Dataset cleaning” databrick, where we first read in each initial CSV file as a Spark dataframe & then changed each one to a Pandas dataframe for cleaning.

For reading in the CSVs, we used the mount point ‘**/mnt/healthcare/dataIn’.**

State

1. Created a State dataframe using the values in the column ‘locationdesc’ in the BRFSS Coverage dataset (#9 in the list of data sources)
2. Changed “District of Columbia” to “DC”
3. Added New Jersey
4. Replaced spaces in state names with underscores
5. Sorted by StateName
6. Used a for loop to create list of IDs, corresponding to the primary key field in the SQL table
7. Added the list of IDs as a column called StateID

County (Data Source #10)

1. Created a County dataframe using the counties from data source #10, which lists all counties within the United States including Puerto Rico and DC.
2. Dropped all values except for the column with county names
3. Sorted by county.
4. Used a for loop to create a list of IDs, corresponding to the primary key field in the SQL table
5. Added the IDs as a column called CountyID

Hospital Count (Data Source #1)

For the Hospital Count by State, the following steps were performed for transforming and cleaning the data:

1. After reading the dataset in as a Spark dataframe, we changed it into a Pandas dataframe and changed the column names.
    a. Geographic Area Name (NAME) to States
    b. 2017 NAICS code (NAICS2017) to StateCode
    c. Meaning of NAICS code (NAICS2017_LABEL) to Label
    d. Meaning of Legal form of organization code (LFO_LABEL) to Different_Establishments
    e. Meaning of Employment size of establishments code (EMPSZES_LABEL) to Establishment_Size
    f. Year (YEAR) to Year
    g. Number of establishments (ESTAB) to Total_Establishments
    h. Annual payroll ($1,000) (PAYANN) to Annual_Payroll
    i. First-quarter payroll ($1,000) (PAYQTR1) to Q1Payroll
    j. Number of employees (EMP) to Employee_Size
2. We then selected the States, Different_Establishments, Establishment_Size, Total_Establishments, and Employee_Size columns and created a new dataframe from these.

3. Next, we changed the type of the Employee_Size column to be an integer.
4. We returned the dataframe to a Spark dataframe.

Demographics (Data Source #2):

1. After importing, we dropped all NA values within the dataset which removes 5 rows: SEX AND AGE, RACE, Race alone or in combination with one or more other races, HISPANIC OR LATINO AND RACE, and CITIZEN< VOTING AGE POPULATION.
2. We then removed the suffix ('!!Estimate') and replaced the spaces with an underscore for each column name then saved this as a variable for later use. Each column name in the original dataset is a state name.
3. We then dropped the following row indices and categories from the dataset as we were not interested in these categories or they were categories that repeated:
    a. 4, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 65, 66, 67, 68, 69, 70, 71, 73, 75, 76, 77, 78, 79, 86, 87, 88, 89, 91, 92, and 93.
    b. Respective category names: Sex ratio (males per 100 females), Median age (years), Under 18 years, 16 years and over, 18 years and over, 21 years and over, 62 years and over, 65 years and over, 18 years and over, Male, Female, Sex ratio (males per 100 females), 65 years and over, Male, Female, Sex ratio (males per 100 females), Total population, One race, Two or more Races, One race, White, Black or African American, American Indian and Alaska Native, Cherokee tribal grouping, Chippewa tribal grouping, Navajo tribal grouping, Sioux tribal grouping, Asian, Asian Indian, Chinese, Filipino, Japanese, Korean, Vietnamese, Other Asian, Native Hawaiian and Other Pacific Islander, Native Hawaiian, Guamanian or Chamorro, Samoan, Other Pacific Islander, Some other race, Two or more races, White and Black or African American, White and American Indian and Alaska Native, White and Asian, Black or African American and American Indian and Alaska Native, Total population, White, Black or African American, American Indian and Alaska Native, Asian, Native Hawaiian and Other Pacific Islander, Some other race, Total Population, Mexican, Puerto Rican, Cuban, Other Hispanic or Latino, Not Hispanic or Latino, Two or more races, Two races including Some other race, (Two races excluding Some other race, and Three or more Races), Total housing units, (Citizen, 18 and over population), Male, and Female.
4. Then we transposed the dataset such that the categories we are interested in became the columns & the states became their own column.
5. We then renamed the columns to the respective categories and dropped the first column which had the previous columns.
6. We then reset the index and renamed the columns as stated below.
    a. Since transposing the dataset sets the column names equal to their index for the row value, each column is renamed as their category as stated below:
        i. 1 to Total_Population
        ii. 2 to Male

        iii. 3 to Female
        iv. 5 to Under_5Y
        v. 6 to _5_to_9Y
        vi. 7 to_10_to_14Y
        vii. 8 to _15_to_19Y
        viii. 9 to _20_to_24Y
        ix. 10 to _25_to_34Y
        x. 11 to _35_to_44Y
        xi. 12 to _45_to_54Y
        xii. 13 to _55_to_59Y
        xiii. 14 to _60_to_64Y
        xiv. 15 to _65_to_74Y
        xv. 16 to _75_to_84Y
        xvi. 17 to _85_and_Older
        xvii. 74 to Hispanic
        xviii. 80 to White
        xix. 81 to African_American
        xx. 82 to American_Indian
        xxi. 83 to Asian
        xxii. 84 to Pacific_Islander
        xxiii. 85 to Some_Other_Race
        xxiv. 94 to State
7. Then, we removed the comma from all values in each column except State and changed the type of each column besides State to integer.
8. Lastly, we changed District_of_Columbia to DC in the State column.

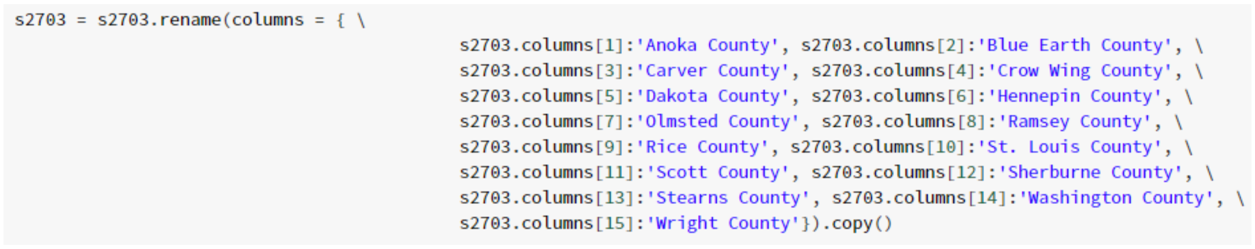
Health Insurance Characteristics (Data Source #3):

1. We first dropped the rows that were all NA values, which were AGE, SEX, RACE AND HISPANIC OR LATINO ORIGIN, LIVING ARRANGEMENTS, NATIVITY AND U.S. CITIZENSHIP STATUS, DISABILITY STATUS, EDUCATIONAL ATTAINMENT, EMPLOYMENT STATUS, WORK EXPERIENCE, HOUSEHOLD INCOME (IN 2019 INFLATION-ADJUSTED DOLLARS), AND RATIO OF INCOME TO POVERTY LEVEL IN THE PAST 12 MONTHS.
2. We then selected the first column (Label (Grouping)) which contained the categories for each row and stripped the white space from the beginning of each then set this equal to the first column.
3. Next, we transposed the table and went through the index column to split the state, county, and insurance category from each then put them in their own columns as it was listed like: Baldwin County, Alabama!!Total!!Estimate.
4. Next, we renamed the columns to the categories and reset the index.
5. We then replaced each ‘N’ string in all columns with 0 to make it easier to impute.
6. Next, we dropped the following columns as we were not interested in them:
    a. Total household population, In non-family households and other living arrangements, In family households, In married couple families, In other families,

(Male reference person, no spouse present), (Female reference person, no spouse present), With a disability, No disability, Civilian noninstitutionalized population 19 to 64 years, In labor force, Employed, Unemployed, Not in labor force, (Worked full-time, year round in the past 12 months), (Worked less than full-time, year round in the past 12 months), Did not work, Civilian noninstitutionalized population for whom poverty status is determined, Civilian noninstitutionalized population 26 years and over, Under 19 years, 19 to 64 years, 65 years and older, Two or more races, White alone, not Hispanic or Latino, Below 138 percent of the poverty threshold, 138 to 399 percent of the poverty threshold, At or above 400 percent of the poverty threshold, and Below 100 percent of the poverty threshold.

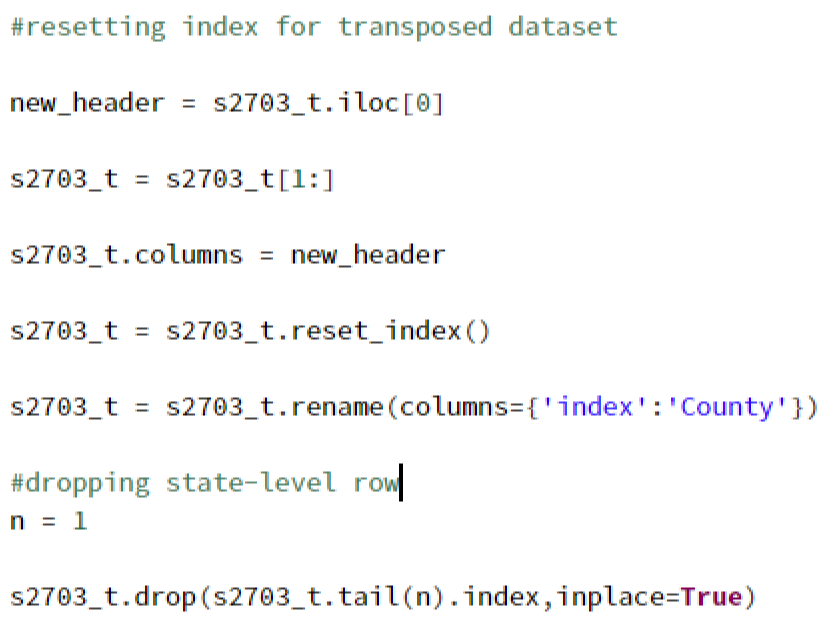
7. We then went through each column except State, County, and Insurance Category to remove the commas and turn the type into an integer.
8. Finally, we reorganized the columns, renamed them as stated below, stripped the white space in the State column, replaced each space with an underscore, and replaced District_of_Columbia with DC.

Private Coverage in Minnesota (Data Source #4):

1. Create a variable and set it equal to *spark.read.options(header = 'True').csv('/mnt/healthcare/dataIn/ACSST1Y2019.S2703-2022-09-21T120206.csv')*. We called this variable s2703. Run this line of code.
2. Set the variable s2703 from Step 4 equal to itself, followed by the code *.toPandas()* without a space. Run this line of code.
3. Now we will remove columns with total population numbers and population percentages:
   a. s2703 = s2703[s2703.columns.drop(list(s2703.filter(regex='Total')))]
   b. s2703 = s2703[s2703.columns.drop(list(s2703.filter(regex='Percent')))]
4. Next, we will remove 3 rows that are almost entirely blank. Run the following three lines of code:
   a. s2703 = s2703.drop(index = s2703.index[1])
   b. s2703 = s2703.drop(index = s2703.index[13])
   c. s2703 = s2703.drop(index = s2703.index[25])
5. We will now shorten the column names. Run the code pictured below:
6. 

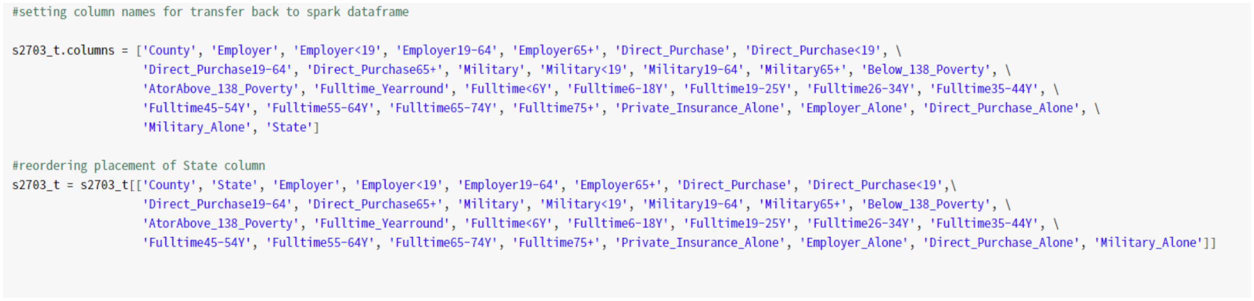
```
s2703 = s2703.rename(columns = { \
                                        s2703.columns[1]:'Anoka County', s2703.columns[2]:'Blue Earth County', \
                                        s2703.columns[3]:'Carver County', s2703.columns[4]:'Crow Wing County', \
                                        s2703.columns[5]:'Dakota County', s2703.columns[6]:'Hennepin County', \
                                        s2703.columns[7]:'Olmsted County', s2703.columns[8]:'Ramsey County', \
                                        s2703.columns[9]:'Rice County', s2703.columns[10]:'St. Louis County', \
                                        s2703.columns[11]:'Scott County', s2703.columns[12]:'Sherburne County', \
                                        s2703.columns[13]:'Stearns County', s2703.columns[14]:'Washington County', \
                                        s2703.columns[15]:'Wright County'}).copy()
```

7. Transpose the dataframe with the following code:
   a. s2703_t = s2703.T
8. After transposing the dataframe, we need to reset the index column and remove the final row, which we will not use in this case. Run the code pictured below:



```
#resetting index for transposed dataset

new_header = s2703_t.iloc[0]

s2703_t = s2703_t[1:]

s2703_t.columns = new_header

s2703_t = s2703_t.reset_index()

s2703_t = s2703_t.rename(columns={'index':'County'})

#dropping state-level row
n = 1

s2703_t.drop(s2703_t.tail(n).index,inplace=True)
```

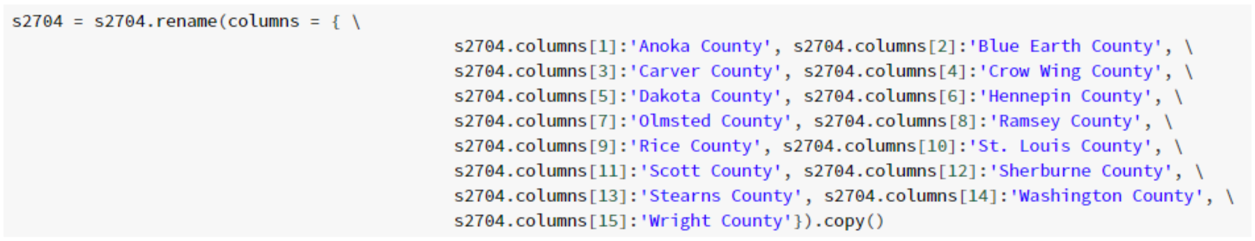
9.
10. Convert column data types by setting s2703_t equal to *s2703_t.convert_dtypes()*. Run this line of code.
11. Drop the civilian noninstitutionalized population column by setting s2703_t equal to *s2703_t.drop(['Civilian noninstitutionalized population'], axis = 1)* and running the code.
12. Remove commas in columns with numbers by setting s2703_t equal to *s2703_t.replace(",","", regex=True)* and running the code.
13. Create a new column that identifies the county in each row as being part of Minnesota with the following code:
    a. s2703_t['State'] = 'Minnesota'
14. Modify column names and placement with the code depicted below:



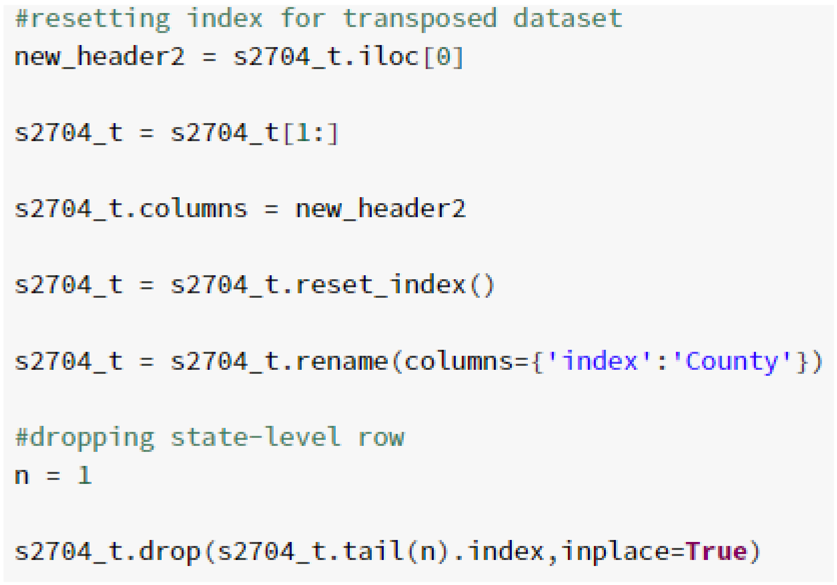
```
#setting column names for transfer back to spark dataframe

s2703_t.columns = ['County', 'Employer', 'Employer<19', 'Employer19-64', 'Employer65+', 'Direct_Purchase', 'Direct_Purchase<19', \
                   'Direct_Purchase19-64', 'Direct_Purchase65+', 'Military', 'Military<19', 'Military19-64', 'Military65+', 'Below_138_Poverty', \
                   'AtorAbove_138_Poverty', 'Fulltime_Yearround', 'Fulltime<6Y', 'Fulltime6-18Y', 'Fulltime19-25Y', 'Fulltime26-34Y', 'Fulltime35-44Y', \
                   'Fulltime45-54Y', 'Fulltime55-64Y', 'Fulltime65-74Y', 'Fulltime75+', 'Private_Insurance_Alone', 'Employer_Alone', 'Direct_Purchase_Alone', \
                   'Military_Alone', 'State']

#reordering placement of State column
s2703_t = s2703_t[['County', 'State', 'Employer', 'Employer<19', 'Employer19-64', 'Employer65+', 'Direct_Purchase', 'Direct_Purchase<19',\
                   'Direct_Purchase19-64', 'Direct_Purchase65+', 'Military', 'Military<19', 'Military19-64', 'Military65+', 'Below_138_Poverty', \
                   'AtorAbove_138_Poverty', 'Fulltime_Yearround', 'Fulltime<6Y', 'Fulltime6-18Y', 'Fulltime19-25Y', 'Fulltime26-34Y', 'Fulltime35-44Y', \
                   'Fulltime45-54Y', 'Fulltime55-64Y', 'Fulltime65-74Y', 'Fulltime75+', 'Private_Insurance_Alone', 'Employer_Alone', 'Direct_Purchase_Alone', 'Military_Alone']]
```

Public Coverage in Minnesota (Data Source #5):

1. Set the variable s2704 from Step 4 equal to itself, followed by the code *.toPandas()* without a space. Run this line of code.
2. Now we will remove columns with total population numbers and population percentages:
    a. s2704 = s2704[s2704.columns.drop(list(s2704.filter(regex='Total')))]
    b. s2704 = s2704[s2704.columns.drop(list(s2704.filter(regex='Percent')))]
3. Next, we will remove 3 rows that are almost entirely blank. Run the following three lines of code:
    a. s2704 = s2704.drop(index = s2704.index[1])

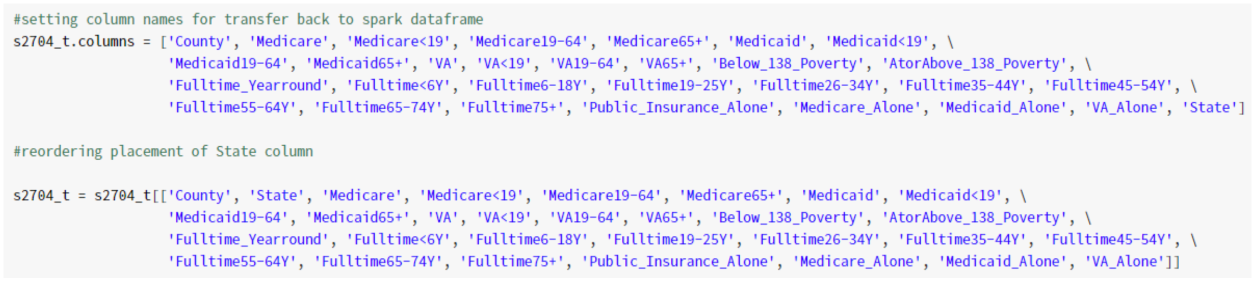
   b. s2704 = s2704.drop(index = s2704.index[13])
   c. s2704 = s2704.drop(index = s2704.index[25])
4. We will now shorten the column names. Run the code pictured below:
5. 

```
s2704 = s2704.rename(columns = { \
                                    s2704.columns[1]:'Anoka County', s2704.columns[2]:'Blue Earth County', \
                                    s2704.columns[3]:'Carver County', s2704.columns[4]:'Crow Wing County', \
                                    s2704.columns[5]:'Dakota County', s2704.columns[6]:'Hennepin County', \
                                    s2704.columns[7]:'Olmsted County', s2704.columns[8]:'Ramsey County', \
                                    s2704.columns[9]:'Rice County', s2704.columns[10]:'St. Louis County', \
                                    s2704.columns[11]:'Scott County', s2704.columns[12]:'Sherburne County', \
                                    s2704.columns[13]:'Stearns County', s2704.columns[14]:'Washington County', \
                                    s2704.columns[15]:'Wright County'}).copy()
```

6. Transpose the dataframe with the following code:
   a. s2704_t = s2704.T
7. After transposing the dataframe, we need to reset the index column and remove the final row, which we will not use in this case. Run the code pictured below:
8. 

```
#resetting index for transposed dataset
new_header2 = s2704_t.iloc[0]

s2704_t = s2704_t[1:]

s2704_t.columns = new_header2

s2704_t = s2704_t.reset_index()

s2704_t = s2704_t.rename(columns={'index':'County'})

#dropping state-level row
n = 1

s2704_t.drop(s2704_t.tail(n).index,inplace=True)
```

9. Convert column data types by setting s2704_t equal to *s2704_t.convert_dtypes()*. Run this line of code.
10. Drop the civilian noninstitutionalized population column by setting s2704_t equal to *s2704_t.drop(['Civilian noninstitutionalized population'], axis* = *1)* and running the code.
11. Remove commas in columns with numbers by setting s2704_t equal to *s2704_t.replace(",","", regex=True)* and running the code.
12. Create a new column that identifies the county in each row as being part of Minnesota with the following code:
    a. s2704_t['State'] = 'Minnesota'
13. Modify column names and placement with the code depicted below:

14. 

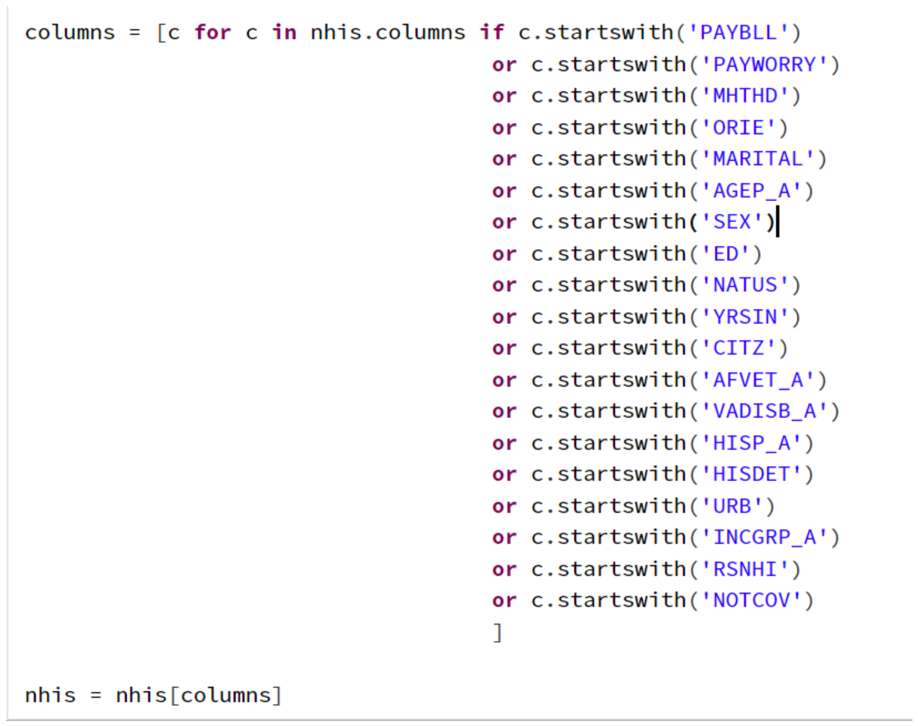
```
#setting column names for transfer back to spark dataframe
s2704_t.columns = ['County', 'Medicare', 'Medicare<19', 'Medicare19-64', 'Medicare65+', 'Medicaid', 'Medicaid<19', \
                   'Medicaid19-64', 'Medicaid65+', 'VA', 'VA<19', 'VA19-64', 'VA65+', 'Below_138_Poverty', 'AtorAbove_138_Poverty', \
                   'Fulltime_Yearround', 'Fulltime<6Y', 'Fulltime6-18Y', 'Fulltime19-25Y', 'Fulltime26-34Y', 'Fulltime35-44Y', 'Fulltime45-54Y', \
                   'Fulltime55-64Y', 'Fulltime65-74Y', 'Fulltime75+', 'Public_Insurance_Alone', 'Medicare_Alone', 'Medicaid_Alone', 'VA_Alone', 'State']

#reordering placement of State column

s2704_t = s2704_t[['County', 'State', 'Medicare', 'Medicare<19', 'Medicare19-64', 'Medicare65+', 'Medicaid', 'Medicaid<19', \
                   'Medicaid19-64', 'Medicaid65+', 'VA', 'VA<19', 'VA19-64', 'VA65+', 'Below_138_Poverty', 'AtorAbove_138_Poverty', \
                   'Fulltime_Yearround', 'Fulltime<6Y', 'Fulltime6-18Y', 'Fulltime19-25Y', 'Fulltime26-34Y', 'Fulltime35-44Y', 'Fulltime45-54Y', \
                   'Fulltime55-64Y', 'Fulltime65-74Y', 'Fulltime75+', 'Public_Insurance_Alone', 'Medicare_Alone', 'Medicaid_Alone', 'VA_Alone']]
```

SAHIE (Data Source #6):

1. Skip the first 78 lines (which show the data dictionary) while reading the dataset in as a Spark dataframe, then transform to Pandas.
2. Rename the headers to: 'year', 'version', 'StateID', 'countyfips', 'geocat', 'agecat', 'racecat', 'sexcat', 'incomecat', 'numdemo', 'numdemo_moe', 'NUI', 'nui_moe', 'NI', 'ni_moe', 'PCTUIdemo', 'PCTUIdemo_moe', 'PCTIdemo', 'PCTIdemo_moe', 'PCTUI', 'pctui_moe', 'PCTI', 'pcti_moe', 'state_name', 'county_name'
3. Drop the following columns: 'version', 'StateID', 'countyfips', 'geocat'
4. Replace the following character sequences with np.NaN
    a. '   .' (7 spaces followed by a period)
    b. '   . ' (3 spaces followed by a period followed by another space)
5. In the ‘county_name’ column, replace the apostrophe marks with an empty string
6. Import the SimpleImputer function (code: *from sklearn.impute import SimpleImputer*)
7. Impute the null values (NaNs that were replaced in step 4) with the mean for the following columns: 'numdemo', 'numdemo_moe', 'NUI', 'nui_moe', 'NI', 'ni_moe', 'PCTUIdemo', 'PCTUIdemo_moe', 'PCTIdemo', 'PCTIdemo_moe', 'PCTUI', 'pctui_moe', 'PCTI', 'pcti_moe'

NHIS (Data Source #7):

1. We created a list of columns to keep by selecting columns that began with or contained certain words or letters & selected those columns using the following code. (see the NHIS Columns document for details on the titles of these columns & explanations)



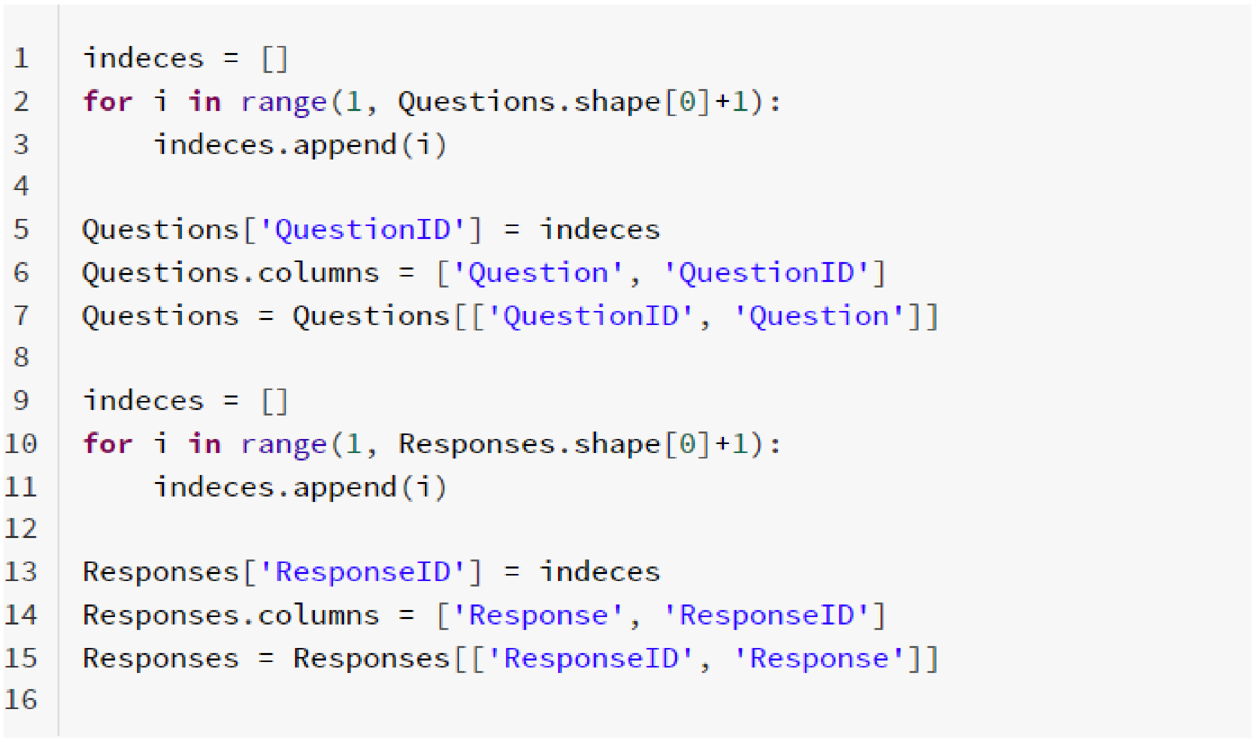
```
columns = [c for c in nhis.columns if c.startswith('PAYBLL')
                                     or c.startswith('PAYWORRY')
                                     or c.startswith('MHTHD')
                                     or c.startswith('ORIE')
                                     or c.startswith('MARITAL')
                                     or c.startswith('AGEP_A')
                                     or c.startswith('SEX')
                                     or c.startswith('ED')
                                     or c.startswith('NATUS')
                                     or c.startswith('YRSIN')
                                     or c.startswith('CITZ')
                                     or c.startswith('AFVET_A')
                                     or c.startswith('VADISB_A')
                                     or c.startswith('HISP_A')
                                     or c.startswith('HISDET')
                                     or c.startswith('URB')
                                     or c.startswith('INCGRP_A')
                                     or c.startswith('RSNHI')
                                     or c.startswith('NOTCOV')
                                     ]

nhis = nhis[columns]
```

BRFSS SMART (Data Source #8):

Transforming this dataset included creating tables for Question & Response, which were also used with data source #9. The steps for this are shown immediately before the transformations for this dataset.

Question & Response Tables

1. In a databrick notebook, run the code *df.select('column').distinct().collect()* to identify unique column values. Here, df is the dataframe representing either the Metro dataset or the Coverage dataset, while column represents either the Question or Response column. To acquire the unique values for both columns in both dataframes, run the code 4 times total–once for each dataframe and column combination.
2. Once verifying the distinct values, set two variables, Q and R, equal to df.select('column').distinct().collect(), with df representing either the Metro or Coverage dataframe and column equaling 'Question' for the Q variable and 'Response' for the R variable.
3. If for any reason one dataframe produces a greater number of distinct values for Question or Response, default to using that dataframe to be inclusive of all values. Both Coverage and Metro tables will need to reference the Question and Response tables in SQL.
4. Make a variable, Questions, equal to Q.toPandas()
5. Make a variable, Responses, equal to R.toPandas()
6. To populate both Questions and Responses with an ID column, run the code pictured below in Step 7:

7. 

```python
indeces = []
for i in range(1, Questions.shape[0]+1):
    indeces.append(i)

Questions['QuestionID'] = indeces
Questions.columns = ['Question', 'QuestionID']
Questions = Questions[['QuestionID', 'Question']]

indeces = []
for i in range(1, Responses.shape[0]+1):
    indeces.append(i)

Responses['ResponseID'] = indeces
Responses.columns = ['Response', 'ResponseID']
Responses = Responses[['ResponseID', 'Response']]

```

1. The BRFSS SMART dataframe, read into the databrick as a Spark dataframe & subsequently changed to a Pandas dataframe, is referred to as smart2019.
2. Set the variable smart2019 equal to itself, followed by the code *.loc[(smart2019['Class'] == 'Health Care Access/Coverage')]* without a space. Run this line of code.
3. Set smart2019 equal to smart2019[['Locationdesc', 'Question', 'Response', 'Sample_Size', 'Data_value']]. Run this line of code.
4. So far, we have specified the columns and rows we want for the Metro table. Now, we will modify the data types of the columns for placement in SQL. Run the following three lines of code:
    a. smart2019 = smart2019.convert_dtypes()
    b. smart2019 = smart2019.replace(",","", regex=True)
    c. smart2019['Sample_Size'] = smart2019['Sample_Size'].astype(int)
5. Now we will replace the Question and Response strings with the QuestionID and Response ID numbers from the Question and Response tables. Consult the ETL Report section on those tables for more information on those tables. Run the following lines of code:
    a. smart2019= smart2019.replace(['About how long has it been since you last visited a doctor for a routine checkup?', 'Adults aged 18-64 who have any kind of health care coverage (variable calculated from one or more BRFSS questions)', 'Do you have any kind of health care coverage?', 'Do you have one person you think of as your personal doctor or health care provider?', 'Was there a time in the

past 12 months when you needed to see a doctor but could not because of cost?'], [1,2,3,4,5])
    b. smart2019=smart2019.replace(['5 or more years ago', 'More than one', 'Never', 'No', 'Within the past 2 years', 'Within the past 5 years', 'Within the past year', 'Yes', 'Yes only one'], [1,2,3,4,5,6,7,8,9])
6. Now we will rename and re-order the columns. Run the following lines of code:
    a. smart2019.columns = ['Location_Desc', 'QuestionID', 'ResponseID', 'Sample_Size', 'Data_Value']
    b. smart2019 = smart2019[['QuestionID', 'ResponseID', 'Location_Desc', 'Sample_Size', 'Data_Value']]

BRFSS Coverage (Data Source #9):

1. Drop unnecessary columns: keep data value, sample size, break out, break out category, question, response, and location.
2. Drop null values, which represent data that was excluded
3. Cast the data value column as the float datatype.
4. Cast the sample size column as the int datatype.

Following these steps, each dataset was returned to a Spark dataframe and read to a CSV in an Azure blob called “Cleaned Data.” To do this, we employed a second mount point **‘/mnt/healthcare/cleandataOut/CleanedData’.**

Transformation for normalization:

The following steps were completed in the “SQL Populate Tables” databrick. Each dataset listed below, with the exception of Category & BreakOut, was first read in as a Spark dataframe from the Azure blob “Cleaned Data,” then changed to a Pandas dataframe.

We imported the following libraries:

```
import pyspark.sql.functions
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pyspark.sql.functions import *
from pyspark.sql.types import StringType,DecimalType
from pyspark.sql.functions import input_file_name, substring
from pyspark.sql.functions import isnan, when, count, col
```

State:

1. Read in the State dataframe as a Spark dataframe & transform to Pandas before transforming the following tables.

County:

1. Read in the County dataframe as a Spark dataframe & transform to Pandas before transforming the following tables.

Hospital Counts (Data Source #1)

1. In the States column, we replaced the value 'District of Columbia' with 'DC' and all spaces with an underscore
2. We replaced the values in the States column with the corresponding ID value in the State dataframe.
3. The final table schema from start to finish is as follows:

Hospital table mapping:

Source Table Columns	***Destination Table Columns***
Geographic Area Name (NAME)	StateID
Meaning of Legal form of organization code (LFO_LABEL)	Different_Establishments
Meaning of Employment size of establishments code (EMPSZES_LABEL)	Establishment_Size
Number of establishments (ESTAB)	Total_Establishments
Number of employees (EMP)	Employee_Size

Demographics (Data Source #2):

1. Replace the State name column with the IDs from the State dataframe.

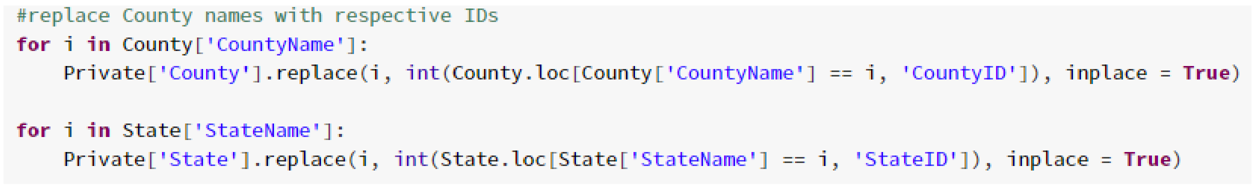
2. The final table schema from start to finish is as follows:

Demographics Transposed Table mapping:

Source Table Columns	***Destination Table Columns***
Label (Grouping)	StateID
Civilian non-institutionalized population	Total_Population
Under 5 years	Under5Y
5 to 9 years	_5_to_9Y
10 to 14 years	_10_to_14Y
15 to 19 years	_15_to_19Y
20 to 24 years	_15_to_19Y
25 to 34 years	_25_to_34Y
35 to 44 years	_35_to_44Y
45 to 54 years	_45_to_54Y
55 to 64 years	_55_to_64Y
65 to 74 years	_65_to_74Y
75 to 84 years	_75_to_84Y
85 years and older	_85_and_Older
Hispanic or Latino (of any race)	Hispanic
White alone	White
Black or African American alone	African_American
American Indian and Alaska Native alone	American_Indian

Asian alone	Asian
Native Hawaiian and Other Pacific Islander alone	Pacific_Islander
Some other race alone	Some_Other_Race

Private Coverage in Minnesota (Data Source #4):

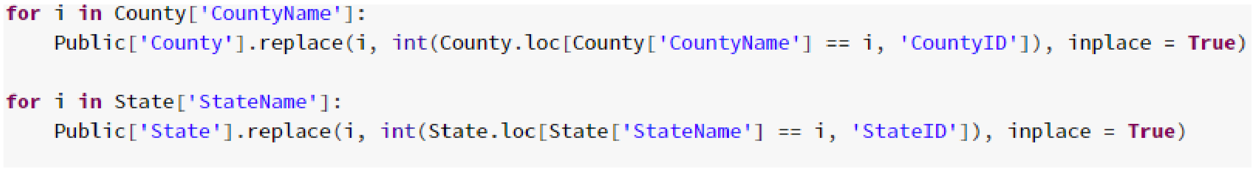
1. Refer to this dataframe, imported as referenced above, as Private.
2. Use the County and State dataframes to replace county and state names in Private with their respective IDs. Consult the sections on the County and State tables in this report for more information on them. Run the code depicted below:
3. 

```
#replace County names with respective IDs
for i in County['CountyName']:
    Private['County'].replace(i, int(County.loc[County['CountyName'] == i, 'CountyID']), inplace = True)

for i in State['StateName']:
    Private['State'].replace(i, int(State.loc[State['StateName'] == i, 'StateID']), inplace = True)
```

4. Rename the County and State columns with the following code:
    a. Private = Private.rename(columns={'County':'CountyID', 'State':'StateID'})
5. Make Private a Spark dataframe by setting it equal to spark.createDataFrame(Private). Run this line of code.
6. Specify variables corresponding to your SQL server, database, table, username and password.

The columns retained in the private coverage dataset directly correspond to columns in the original data source, with State and County being changed to StateID and CountyID.

Public Coverage in Minnesota (Data Source #5)

1. This dataframe, after importing & setting as a Pandas dataframe as referenced above, is referred to as Public.
2. Now use the County and State dataframes to replace county and state names in Public with their respective IDs. Consult the sections on the County and State tables in this report for more information on them. Run the code depicted below:
3. 

```
for i in County['CountyName']:
    Public['County'].replace(i, int(County.loc[County['CountyName'] == i, 'CountyID']), inplace = True)

for i in State['StateName']:
    Public['State'].replace(i, int(State.loc[State['StateName'] == i, 'StateID']), inplace = True)
```

4. Rename the County and State columns with the following code:
    a. Public = Public.rename(columns={'County':'CountyID', 'State':'StateID'})

The columns retained in the public coverage dataset directly correspond to columns in the original data source, with State and County being changed to StateID and CountyID.

SAHIE (Data Source #6)

1. Replace all values of 'District of Columbia' in the 'state_name' column with 'DC'
2. Replace all spaces in the 'state_name' column with an underscore
3. In order to connect to the state table, replace all values in the 'state_name' column with the corresponding StateID values (see state table mapping for more information)
4. In order to connect to the county table, replace all values in the 'county_name' column with the corresponding CountyID values (see county table mapping for more information)
5. The SQL schema for this table matches the following order: 'agecat', 'racecat', 'sexcat', 'incomecat', 'numdemo', 'numdemo_moe', 'NUI', 'nui_moe', 'NI', 'ni_moe', 'PCTUIdemo', 'PCTUIdemo_moe', 'PCTIdemo', 'PCTIdemo_moe', 'PCTUI', 'pctui_moe', 'PCTI', 'pcti_moe', 'StateID', 'CountyID'

NHIS (Data Source #7):

1. Reorder the columns as follows:

'URBRRL', 'INCGRP_A', 'YRSINUS_A', 'CITZNSTP_A', 'NOTCOV_A', 'RSNHIMISS_A', 'RSNHIJOB_A', 'EDUC_A', 'HISDETP_A', 'HISP_A', 'SEX_A', 'AGEP_A', 'NATUSBORN_A', 'VADISB_A', 'AFVET_A', 'MARITAL_A', 'ORIENT_A', 'MHTHDLY_A', 'PAYWORRY_A', 'PAYBLL12M_A', 'RSNHIOTH_A', 'RSNHIWAIT_A', 'RSNHIMEET_A', 'RSNHICONF_A', 'RSNHIELIG_A', 'RSNHIWANT_A', 'RSNHICOST_A'

The SQL schema for this table matches the order listed here. The columns retained also correspond directly to columns in the original dataset,

BRFSS SMART (Data Source #8):

For this dataset as well as the Question & Response tables, no extra transformations before the steps in the Load section were needed to prepare this dataset for the SQL database.

BRFSS Coverage (Data Source #9):

1. Create a dataframe containing break out category values (Category), sorted alphabetically by category name
   a. Drop break out category column in the coverage dataframe
2. Use a for loop to create IDs for each value, beginning at 1
3. Create a dataframe containing break out values & categories (BreakOut), sorted alphabetically by category name
4. Use a for loop to create IDs for each break out value, beginning at 1

5. Use a for loop to replace category values in the BreakOut table with the corresponding IDs

For this dataset, it was necessary to create a BreakOut table and a Coverage table using the values in the corresponding columns in this dataset.

1. Use the previously-created Question & Response tables to replace the values in the Question and Response columns with the corresponding IDs, using a for loop for each table
2. Use BreakOut & Category tables to replace the values in those columns in the BRFSS Coverage table
    a. This replaces BreakOut & Category string values with the corresponding IDs.

For the final steps, skip directly to the Load section of this document.

Streamed Dataset

We ran the Health Insurance Characteristics dataset (Data Source #3) through a producer and consumer to simulate streamed data. We used the producer & consumer code provided to us, which can be seen in the Healthcare-Producer & Healthcare-Consumer notebooks. We automated the producer & consumer using a data factory pipeline. We then completed the following transformations in the Healthcare-Consumer notebook.

1. We read the cleaned Health Insurance Characteristics dataset into the producer databrick from the Azure blob as a Spark dataframe.
2. We turned the Spark dataframe into a dictionary & ran the producer.
3. After running the producer databrick, the consumer databrick returned a dictionary, which we used to create a dataframe.
    a. We created a Spark dataframe out of the dictionary which we then turned into a Pandas dataframe.
4. We reordered the columns in the dataframe.
5. We created a list of the names of columns to be cast as an integer type.
6. We then cast those columns as the integer type.
7. We created IDs for this dataframe to include when writing to SQL & added them in a new column.
    a. Because this is a streamed dataset, we are *overwriting* the SQL tables; we included our own ID column to keep IDs consistent each time.
8. We turned the final dataframe including the IDs into a Spark dataframe.
9. The process for loading it into SQL was the same as the others.

10. The table schema from start to finish is as follows:

Health Insurance Characteristics Table mapping:

Source Table Columns	***Destination Table Columns***
Label (Grouping) *(contained state, county, and insurance category in the original dataset; we split these into 3 distinct columns for the destination table)*	StateID
	CountyID
	Insurance_Category
Under 6 years	Under_6Y
6 to 18 years	_6_to_18Y
19 to 25 years	_19_to_25Y
26 to 34 years	_26_to_34Y
35 to 44 years	_35_to_44Y
45 to 54 years	_45_to_54Y
55 to 64 years	_55_to_64Y
65 to 74 years	_65_to_74Y
75 years and older	_75_an_Older
White alone	White
Black or African American alone	African_American
American Indian and Alaska Native alone	American_Indian
Asian alone	Asian

Native Hawaiian and Other Pacific Islander alone	Pacific_Islander
Some other race alone	Some_Other_Race
Hispanic or Latino (of any race)	Hispanic
Native born	Native_Born
Foreign born	Foreign_Born
Naturalized	Naturalized
Not a citizen	Not_A_Citizen
Less than high school graduate	Less_Than_High_School
High school graduate (includes equivalency)	High_School_or_Equivalent
Some college or associate's degree	Some_College
Bachelor's degree or higher	Bachelors_or_Higher
Under $25,000	Under_25000S
$25,000 to $49,999	_25000_to_49999S
$50,000 to $74,999	_50000_to_79999S
$75,000 to $99,999	_75000_to_99999S
$100,000 and over	Over_100000S

Load

For each dataset, including Health Insurance Characteristics, the loading process included the following:

1. We ensured each dataframe was in a Spark dataframe format after the cleaning & transformation process.
2. We initialized the SQL schema, as seen in our DDL file, in Azure Data Studio.
3. We read the SQL table into our “SQL Populating Tables” databrick notebook.
4. Using an if statement to check the number of rows in the imported SQL table, we verified that the SQL tables were unpopulated.

Extra steps for State, County, Question, Response, & Category while loading to SQL:

1. Select only the main column, not the ID column, to load to SQL.
2. Sort the selected column to ensure that SQL populated the ID fields in a manner consistent with the IDs created in the databrick.

Extra step for BreakOut:

1. Follow step 1 as completed for State, County, Question, Response, & Category
2. Sort BreakOut by CategoryID

Order to load datasets into SQL database:

1. Load Question & Response before loading Metro and Coverage
2. Load Category before BreakOut
3. Load BreakOut before Coverage

Conclusion

After extracting, transforming, and loading the data from our nine sources into a SQL database, we connected the SQL database to a dataflow in PowerBI. This enabled us to refresh the streamed dataset on a schedule. By connecting it to a dataflow, we were also able to access all the data in order to create visualizations & a dashboard in PowerBI. Following the ETL process, we moved forward with creating a machine learning model based on the Health Insurance Characteristics dataset & creating visuals to answer the questions we posed in the introduction.