ETL Report: CMS and US Census Data

Alysia Halverson, Charlie Rehder, Jakob Thunen

5/23/2022

Introduction

This project requires the use of several datasets provided by the Center for Medicare and Medicaid (CMS) and the US Census Bureau. To elaborate, data on several different value-based purchasing metrics and comparisons of procedure insurance estimates were retrieved from the CMS, and a table of public insurance (Medicare/Medicaid/VA) demographics by state was retrieved from the Census Bureau. This data was fortunately relatively clean and close to the desired format, but it needed to be transformed nonetheless to use it for visualizations and machine learning.

Data Sources

The Centers for Medicare and Medicaid Services

To better understand health insurance quality of care, specifically related to Medicare and Medicaid payments and quality measures, we will be utilizing the following five datasets. A complete data dictionary can be found at https://data.cms.gov/provider-data/sites/default/files/data_dictionaries/hospital/HospitalCompare-DataDictionary.pdf. All the following datasets have a related variable, Facility ID, that we will use to join the data.

- *Hospital Value-Based Purchasing (HVBP) - Efficiency Scores*. (2022, April 27). [Dataset]. https://data.cms.gov/provider-data/dataset/su9h-3pvj
    - The Efficiency and Cost Reduction dataset provides details on the Medicare Spending per Beneficiary (MSPB) measure. The MSPB measure is the hospital's MSPB measure ratio calculated as MSPB amount (hospital's Medicare spending per beneficiary dollar amount) divided by the median MSPB amount (national median MSPB dollar amount).

- *Hospital Value-Based Purchasing (HVBP) - Person and Community Engagement Domain Scores (HCAHPS)*. (2022, April 27). [Dataset]. https://data.cms.gov/provider-data/dataset/avtz-f2ge
    - The Person and Community Engagement Domain Scores dataset provides details on eight dimensions related to patient experience of care collected from Hospital Consumer Assessment of Healthcare Providers and Systems (HCAHPS) surveys. CMS calculates these scores using the "top-box raw score", which is the unrounded percentage of a hospital's patients who chose the most positive

response to the HCAHPS survey items. More information about these scores can be found in the HVBP Step-by-Step Guide (linked). Higher percentages correspond to higher quality.

- *Hospital Value-Based Purchasing (HVBP) - Safety*. (2022, April 27). [Dataset]. https://data.cms.gov/provider-data/dataset/dgmq-aat3
    - The Safety Measures dataset provides details on healthcare-associated infections (HAI). These measures are standardized infection ratios (SIRs) using the number of observed infections (numerator) divided by the number of predicted infections, calculated by the CDC, (denominator). Lower values correspond to higher quality.

- *Hospital Value-Based Purchasing (HVBP) - Clinical Outcomes Domain Scores*. (2022, April 27). [Dataset]. https://data.cms.gov/provider-data/dataset/pudb-wetr
    - The Clinical Outcomes dataset provides details on mortality on Acute Myocardial Infarction (AMI), Heart Failure (HF), Pneumonia (PN), and Total Hip/Knee Arthroplasty (COMP-HIP-KNEE) in a survival ratio. Higher values indicate better outcomes.

- *Payment and value of care - Hospital*. (Jan 26, 2022). [Dataset]. https://data.cms.gov/provider-data/dataset/c7us-v4mf
    - The Payment and Value of Care dataset provides details on payments and cost estimates. The dataset also includes the payment category (Is the payment greater, no different, or less than the average payment?) and value of care category (Is the mortality better, average, or worse? Are the complications average or worse? Was the payment high, low, or average?). Our ML model will attempt to predict payment price using these features.

United States Census Bureau

- U.S. Census Bureau, *2015-2019 5-Year American Community* Survey, Table S2704: Public Health Insurance Coverage by Type and Selected Characteristics (All States within United States). (2020) [Data set]. https://data.census.gov/cedsci/table?t=Health%20Insurance&g=0100000US%2404000000&tid=ACSST5Y2020.S2704&tp=true
    - The Public Health Insurance Coverage by Type and Selected Characteristics (All States within United States) will provide population counts for the state, the

number of residents on public health insurance alone, and the breakdown populations for each type of public health insurance: Medicare, Medicaid, and VA-only. We will merge on State/State ID to join this dataset with our other data.

Extraction

Data from the above sources can be extracted in two ways: downloaded directly from the respective webpages as CSVs or retrieved from the organizations' APIs as JSON objects. The raw CSVs were used to determine the steps to take in cleaning and transforming the data, and the APIs were used to retrieve the data automatically in the data factory.

Within the data factory, the Kafka producer acquires data from the APIs and streams it to the Kafka consumers, which in turn sends the data to the Azure Data Lake. Then, the data is pulled by databricks to be transformed in the next step.

CMS Data

Producer

1. Kafka producer (team-rocket-cms-producer) is created in Databricks
2. Producer calls CMS API through a query string and calls on six datasets
3. Topics are created with names respective to its dataset name
4. Producer iterates through each dataset and sends json response to confluent

Consumer

5. Kafka consumer (team-rocket-cms-consumer) is created in Databricks
6. Six consumers are created by subscribing to the six topics created by the producer
7. Kafka messages are consumed, decoded, formatted, and saved in Azure Data Lake to csv files for ETL

US Census

Producer

1. Kafka producer (team-rocket-census-producer) is created in Databricks
2. Producer calls Census API with key and timeout of 20 within a try/except loop
3. API is then saved into spark data frame
4. Transforming steps are taken to properly name columns and dropping original header
5. Topic is created (rcensus)
6. Producer iterates through dataframe and sends messages for each row as a dictionary

Consumer

7. Kafka consumer (team-rocket-census-consumer) is created in Databricks
8. Consumer subscribes to rcensus topic
9. Kafka messages are consumed, decoded, and formatted into dictionaries
10. Spark data frame is created for brief cleaning (detailed below)

Transformation

Below are the steps taken to transform each dataset. Transformation was done via PySpark on Azure Databricks.

CMS Data

To determine the necessary transformations and test them, the CSVs from the CMS website were used, but the data factory will use the API for automation.

hvbp_clinical_outcomes.csv

1. Read in csv from container, `header=true`
2. Create dataframe (`df`)
3. Select only the necessary columns as noted in Exercise-Initial-Capstone-Data-Sources
4. Rename columns according to SQL schema
5. Cast appropriate columns from string to int or double data types
6. Deal with nulls by imputing the median value of the respective column and create new dataframe (`df_imputed`)

Payment_and_Value_of_Care-Hospital.csv

1. Read in csv from container, `header=true`
2. Create dataframe (`df`)
3. Drop the unnecessary columns as noted in Exercise-Initial-Capstone-Data-Sources
4. Filter out payment footnote and value of care footnote values to only values with a space ‘ ‘
5. Remove dollar sign from values and cast from string to double with `f.regexp_replace`
6. Drop payment footnote and value of care footnote columns
7. Rename columns according to SQL schema
8. Cast remaining columns from string to int
9. Map out payment category values to numeric based on Exercise-Initial-Capstone-Data-Sources
10. Map out value of care category values to numeric based on Exercise-Initial-Capstone-Data-Sources
11. Check for `null`s (should be none)
12. Create new dataframe (`facilityDF`) and group by facility ID, create columns with average payment, lower estimate, higher estimate, payment category mapping, and value of care category mapping
13. Rename `facilityDF` columns according to SQL schema

hvbp_safety.csv

1. Read in csv from container, `header=true`
2. Create dataframe (`df`)
3. Select only the necessary columns as noted in Exercise-Initial-Capstone-Data-Sources
4. Rename columns according to SQL schema
5. Cast appropriate columns from string to int or double data types
6. Deal with nulls by imputing the median value of the respective column and create new dataframe (`df_imputed`)

hvbp_person_and_community_engagement.csv

1. Read in csv from container, `header=true`
2. Create dataframe (`df`)
3. Drop the unnecessary columns as noted in Exercise-Initial-Capstone-Data-Sources
4. Rename columns according to SQL schema
5. Remove percent sign from values and cast from string to double with `f.regexp_replace`
6. Cast remaining columns from string to int or double data types
7. Deal with nulls by imputing the median value of the respective column and create new dataframe (`df_imputed`)

hvbp_efficiency_and_cost_reduction.csv

1. Read in csv from container, `header=true`
2. Create dataframe (`df`)
3. Select only the necessary columns as noted in Exercise-Initial-Capstone-Data-Sources
4. Rename columns according to SQL schema
5. Cast appropriate columns from string to int or double data types
6. Deal with nulls by imputing the median value of the respective column and create new dataframe (`df_imputed`)

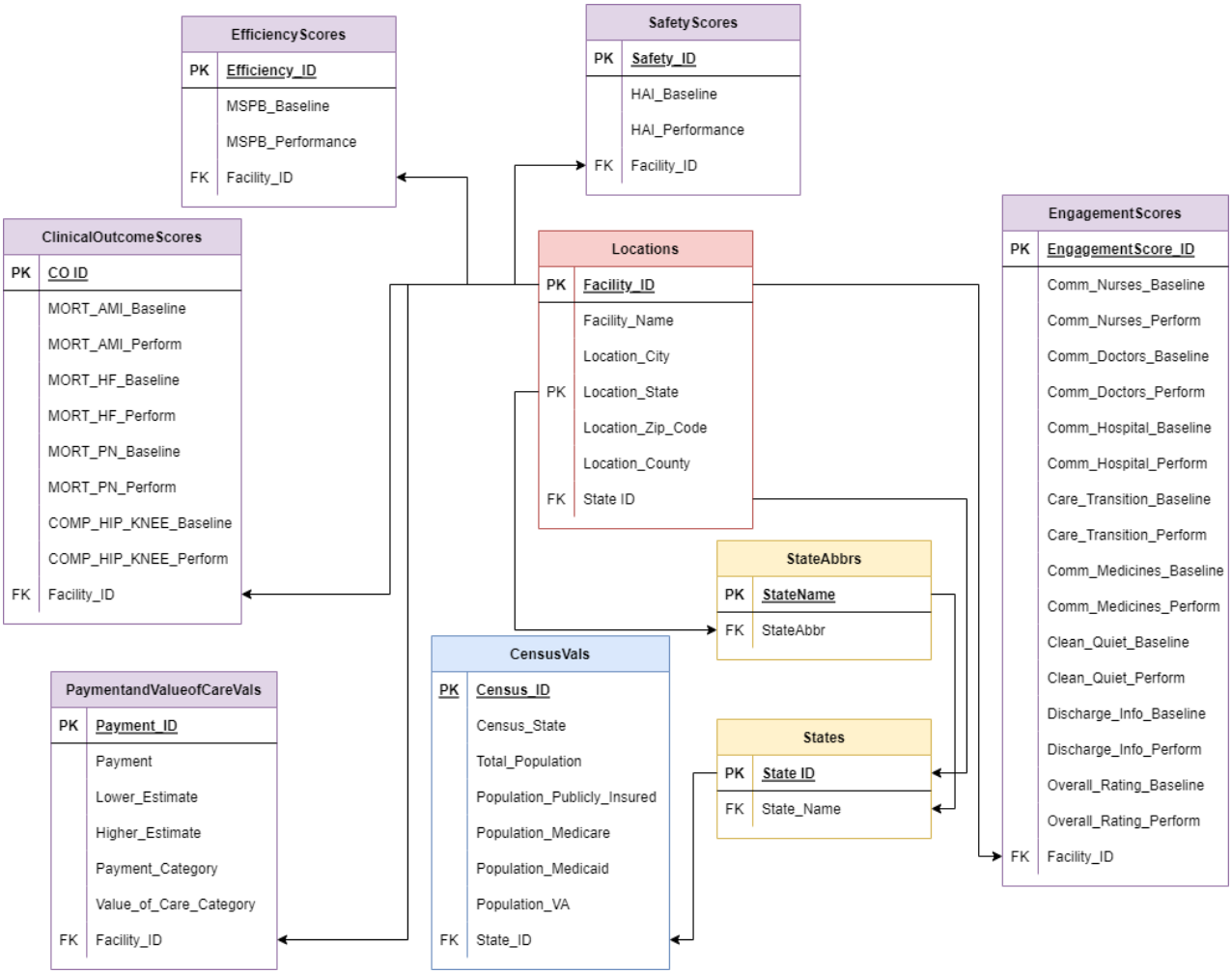
US Census

Unlike the CMS datasets, we were able to acquire a dataset from the Census API that was trimmed down to just the data we needed, so that was used instead of downloading a CSV. Due to the minimal amount of cleaning needed, its transformation was processed in the Kafka consumer.

1. Consume all messages, placing messages into “kafkaListDictionaries”
2. Create dataframe (`censusListdf`) from “kafkaListDictionaries”
3. Ensure no NAs or duplicates remain in dataframe (in case messages were duplicated)
4. Cast appropriate columns from string to int or float data types
5. Rename column names according to SQL schema
6. Establish connection to database
7. Write `censusListdf` to Census table in SQL database

Loading

After transforming the data, it is then loaded into an SQL database whose schema is displayed below. As with the extraction and transformation steps, the data was loaded manually from their respective dataframes while developing and testing the systems, and automatically via Azure Databricks once the testing was complete.



Conclusion

The extraction, transformation, and loading of the data for this project was relatively simple, provided the necessary technical knowledge. In addition to this report, the SQL queries used to create the database and populate the Location and States tables will be included in the Git repository for this project, as well as iPython notebooks of the databricks used to transform the data and populate the other tables. Following the ETL process described in this document, the next step for the project is to export the data to be run through a Machine Learning model to predict the actual price of medical procedures based on conventional estimates, as well as send the raw data to be visualized in a dashboard.