Diabetes ETL Report

Hayden Muscha, Jack Lynn, Temesgen Fekadu, Sam Wainright

8/9/2022

Introduction

Like most non-communicable diseases, diabetes differentially affects people based on genetic, community, and lifestyle factors that potentiate the development of diabetes. To determine what areas of the US are in most need for diabetes outreach, we analyzed and manipulated data from the CDC, US Census Bureau, Rui-Ci Health Care, US Department of Agriculture, and JAEB Center for Health Research. A three-pronged approach is used to create a holistic narrative around diabetes care in the US: 1.) national health datasets describe the demographics of diabetes; 2.) Rui-Ci Health Care's measurable health factors dataset builds into machine learning models to predict diabetes diagnoses; and 3.) JAEB Center data track diabetic patients' blood glucose levels in real time, with deep learning predicting hazardous spikes in blood glucose. From these found commonalities, we are examining the US on a state level basis to see which states or regions should be more active in addressing the factors that can lead to improved diabetic outcomes.

Data Sources

Centers for Disease Control and Prevention. (2017, January 26). National Health and Nutrition Examination Survey. Retrieved August 5, 2022, from www.kaggle.com website: https://www.kaggle.com/datasets/cdc/national-health-and-nutrition-examination-survey

Centers for Disease Control and Prevention. (2021, June 3). National Center for Chronic Disease Prevention and Health Promotion, Division of Nutrition, Physical Activity, and Obesity. Data, Trend and Maps [online]. Retrieved August 5, 2022, from Cdc.gov website: https://www.cdc.gov/nccdphp/dnpao/data-trends-maps/index.html

Centers for Disease Control and Prevention. (2022, March 24). U.S. Chronic Disease Indicators: Diabetes | Chronic Disease and Health Promotion Data & Indicators. Retrieved August 5, 2022, from Socrata website: https://chronicdata.cdc.gov/Chronic-Disease-Indicators/U-S-Chronic-Disease-Indicators-Diabetes/f8ti-h92k/data

Chen, Y., Zhang, X.-P., Yuan, J., Cai, B., Wang, X.-L., Wu, X.-L., … Li, X.-Y. (2018, August 21). Data from: Association of body mass index and age with incident diabetes in Chinese adults: a population-based cohort study. Retrieved August 5, 2022, from datadryad.org website: https://doi.org/10.5061/dryad.ft8750v

JAEB Center for Health Research. (2019). A Randomized Clinical Trial to Assess the Efficacy and Safety of Continuous Glucose Monitoring in Youth < 8 with Type 1 Diabetes. Retrieved August 5, 2022, from public.jaeb.org website: https://public.jaeb.org/dataset/563

U.S. Department of Agriculture. (2021, September 8). Food Security in the U.S. USDA (2019). Retrieved from www.ers.usda.gov website: https://www.ers.usda.gov/topics/food-nutrition-assistance/food-security-in-the-u-s/interactive-charts-and-highlights/

United States Census Bureau. (2022a, March 17). American Community Survey: B06009: PLACE OF BIRTH BY EDUCATIONAL ATTAINMENT IN THE UNITED STATES. Retrieved August 5, 2022, from data.census.gov website: https://data.census.gov/cedsci/table?q=education%20by%20state&tid=ACSDT5Y2020.B06009

United States Census Bureau. (2022b, March 17). American Community Survey: S1901: INCOME IN THE PAST 12 MONTHS (IN 2020 INFLATION-ADJUSTED DOLLARS) U.S. Census Bureau (2020). Retrieved from data.census.gov website: https://data.census.gov/cedsci/table?q=S1901&tid=ACSST5Y2020.S1901

Extraction

1) Extracting the NHANES data from Kaggle:
    a. Travel to the Kaggle page hosting the CDC NHANES data: https://www.kaggle.com/datasets/cdc/national-health-and-nutrition-examination-survey
    b. Download the following files: “demographics.csv”, “examination.csv”, and “questionnaire.csv”.
    c. These files will be downloaded in the ZIP format and must be unzipped and placed in the relevant folder for later transformation.

2) Extracting the CDC data on exercise:
    a. Travel to the CDC website: https://nccd.cdc.gov/dnpao_dtm/rdPage.aspx?rdReport=DNPAO_DTM.ExploreByTopic&islClass=PA&islTopic=PA1&go=GO
    b. Below the blue “Category & Topic” bar, click on the indicator drop down menu and select “Adults aerobically active 150 minutes”.
    c. After the page finishes loading, click the “Year” drop down menu and select “2013”.
    d. In the upper right-hand corner of the window, click the gear wheel below the “About the Data” link.
    e. Click on “Export CSV File (.csv)”
    f. Place the CSV file in the relevant folder for later transformation.

3) Extracting the *U.S. Chronic Disease Indicators: Diabetes* from the CDC:
    a. Travel to the CDC website: https://chronicdata.cdc.gov/Chronic-Disease-Indicators/U-S-Chronic-Disease-Indicators-Diabetes/f8ti-h92k/data
    b. Click the tab in the top right corner above the table labeled “Export”.
    c. From the drop-down menu, select “CSV for Excel”.
    d. Place the CSV file in the relevant folder for later transformation.

4) Extracting the Data from *Association of body mass index and age with incident diabetes in Chinese adults: a population-based cohort study* from Datadryad:
    a. Travel to the Datadryad website: https://doi.org/10.5061/dryad.ft8750v
    b. Click on “Download” dataset in the top-right corner.
    c. The file will download as a ZIP file and needs to be unzipped and placed in the appropriate folder for cleaning.

5) Extracting the JAEB CGM and Patient Data:
    a. Travel to the JAEB: https://public.jaeb.org/dataset/563
    b. Enter your full name, email, company/institution, and planned use of data.
    c. Click the “I agree…” check box and download the dataset.
    d. Extract the ZIP file, open the *RT-CGM Randomized Clinical Trial* folder.
    e. In the *DataTables* folder we will be using the following CSV’s:
        i. tblAAddlCEData
        ii. tblADataRTCGM_Blind_Baseline
        iii. tblAPTSummary
    f. Place the CSV file in the relevant folder for later transformation.

6) Extracting the *Food Security in the U.S. USDA (2019)* from the USDA:
    a. Travel to the USDA website: https://www.ers.usda.gov/topics/food-nutrition-assistance/food-security-in-the-u-s/interactive-charts-and-highlights/
    b. Before the “Graphs”, there is a link labeled “Food Security Data File”, next to it is an icon that appears as such:

    c. Click that image to download “food_security_csv_datafiles.zip”.

   d. Unzip the package and enter the *food_security_csv_datafiles* folder.
   e. In the folder there is a CSV file named “Food security by state_2020.csv”.
   f. Extract that CSV file to the appropriate folder for later transformation.

7) Extracting *B06009: PLACE OF BIRTH BY EDUCATIONAL ATTAINMENT IN THE UNITED STATES* from the Census Bureau American Community Survey:
   a. Travel to the Census Bureau Site: https://data.census.gov/cedsci/table?q=education%20by%20state&tid=ACSDT5Y2020.B06009
   b. Above the table there is an icon labeled “CSV”, click it.
   c. The CSV file is now downloaded and should be placed in the appropriate folder for later transformation.
8) Extracting *S1901: INCOME IN THE PAST 12 MONTHS (IN 2020 INFLATION-ADJUSTED DOLLARS)* from the Census Bureau American Community Survey:
   a. Travel to the Census Bureau Site: https://data.census.gov/cedsci/table?q=S1901&tid=ACSST5Y2020.S1901
   b. Above the table there is an icon labeled “CSV”, click it.
   c. The CSV file is now downloaded and should be placed in the appropriate folder for later transformation.

Transformation

1) Transforming the NHANES data from Kaggle:
   a. Create a Python file and import the *pandas* library
   b. The three tables to be utilized from this set are the *Demographics* dataset, the *Questionnaire* dataset, and the *Examination* dataset.
   c. Read the relevant CSV files for each dataset into their own *DataFrame* using *pandas*.
   d. Select the relevant columns for each *DataFrame*.

      i. From the Questionnaire we took: 'SEQN', 'ALQ101', 'ALQ120Q', 'BPQ020', 'BPQ030', 'DIQ010', 'DIQ160', 'DIQ170', 'DBD895', 'INDFMMPI', 'SMQ020', 'SMQ040', 'WHD010', 'WHD020'
      ii. From the Examination we took: 'SEQN', 'BPXSY1', 'BPXDI1'
      iii. From the Demographic we took: 'SEQN', 'RIAGENDR', 'RIDAGEYR', 'RIDRETH3', 'DMDEDUC2', 'INDHHIN2'
   e. We then inner merged the *DataFrame’s* on SEQN.
   f. Then rename the columns for clarity:
      i. 'SEQN':'SurveyID', 'BPXSY1':'SystolicBP', 'BPXDI1':'DiastolicBP', 'ALQ101':'12drinksInaYear', 'ALQ120Q':'DrinksInLastYear', 'BPQ020':'BeenDiagnostedHypertenisve', 'BPQ030':'MultipleHypertensionDiagnosis', 'DIQ010':'DiagnosedDiabetic', 'DIQ160':'DiagnosedPrediabetic', 'DIQ170':'DiagnosedAtRiskDiabetes', 'DBD895':'NumMealsNotAtHomePerMonth', 'INDFMMPI':'FamilyMonthlyPovertLevel', 'SMQ020':'Smoked100cigs', 'SMQ040':'CurrentSmoker', 'WHD010':'Height', 'WHD020':'Weight', 'RIAGENDR':'Gender', 'RIDAGEYR':'Age(yrs)', 'RIDRETH3':'Ethnicity', 'DMDEDUC2':'AdultEducationLevel', 'INDHHIN2':'AnnualHouseholdIncome'
   g. This data set lacked BMI which is one of the indicators that we are tracking and comparing. We created a BMI column from height and weight using the formula BMI = weight (lb) ÷ height^2 (inches) * 703, as our weight was measured in pounds and our height was provided in inches.
      i. The code to add the columns is as follows:
         1. df['BMI'] = df['Weight'].mul(703) / df['Height'].pow(2)
   h. Replace the categorical numbers for the questionnaire response with the appropriate Value descriptions. This is based on the tables provided by the CDC. Links for the tables are:

      i. *Demographic*: https://wwwn.cdc.gov/nchs/nhanes/search/datapage.aspx?Component=Demographics&CycleBeginYear=2013
      ii. *Examination*: https://wwwn.cdc.gov/nchs/nhanes/search/datapage.aspx?Component=Examination&CycleBeginYear=2013

iii. *Questionnaire*: https://wwwn.cdc.gov/nchs/nhanes/search/datapage.aspx?Component=Questionnaire&CycleBeginYear=2013

i. For ease of repetition each table conversion is showed below:

i. First *12drinksInaYear*:

Code or Value	Value Description	Count	Cumulative
1	Yes	3790	3790
2	No	1623	5413
7	Refused	0	5413
9	Don't know	8	5421
.	Missing	503	5924

1. Convert the column to *String* type.
2. Replace '1.0' with 'True'.
3. Replace '2.0' with 'False'.
4. Replace '9.0' with an empty value.
5. Replace 'nan' with an empty value.
6. If there were any responses of ‘Refused’, they would be categorized as an empty value.

ii. For *DrinksInLastYear*:

Code or Value	Value Description	Count	Cumulative
0 to 365	Range of Values	4475	4475
777	Refused	0	4475
999	Don't know	4	4479
.	Missing	1445	5924

1. Convert the column to *String* type.
2. Replace '999.0' with an empty value.
3. Replace 'nan' with an empty value.
4. If there were any responses of ‘Refused’ they would be categorized as an empty value.

iii. For *BeenDiagnostedHypertensive*:

Code or Value	Value Description	Count	Cumulative
1	Yes	2174	2174
2	No	4285	6459
7	Refused	0	6459
9	Don't know	5	6464
.	Missing	0	6464

1. Convert the column to *String* type.
2. Replace '1.0' with 'True'.
3. Replace '2.0' with 'False'.
4. Replace '9.0' with an empty value.
5. Replace 'nan' with an empty value.
6. If there were any responses of ‘Refused’, they would be categorized as an empty value.

iv. For *MultipleHypertensionDiagnosis*:

Code or Value	Value Description	Count	Cumulative
1	Yes	1738	1738
2	No	428	2166
7	Refused	0	2166
9	Don't know	8	2174
.	Missing	4290	6464

1. Convert the column to *String* type.
2. Replace '1.0' with 'True'.
3. Replace '2.0' with 'False'.

4. Replace '9.0' with an empty value.
5. Replace 'nan' with an empty value.
6. If there were any responses of ‘Refused’, they would be categorized as an empty value.

v. For *DiagnosedDiabetic*:

Code or Value	Value Description	Count	Cumulative
1	Yes	737	737
2	No	8841	9578
3	Borderline	185	9763
7	Refused	1	9764
9	Don't know	5	9769
.	Missing	1	9770

1. Convert the column to *String* type.
2. Replace '1.0' with 'True'.
3. Replace '2.0' with 'False'.
4. Replace '3.0' with an empty value.
5. Replace '7.0' with an empty value.
6. Replace '9.0' with an empty value.
7. Replace 'nan' with an empty value.

vi. For *DiagnosedPrediabetic*:

Code or Value	Value Description	Count	Cumulative
1	Yes	278	278
2	No	6001	6279
7	Refused	0	6279
9	Don't know	8	6287
.	Missing	3483	9770

1. Convert the column to *String* type.
2. Replace '1.0' with 'True'.
3. Replace '2.0' with 'False'.
4. Replace '9.0' with an empty value.
5. Replace 'nan' with an empty value.
6. If there were any responses of ‘Refused’, they would be categorized as an empty value.

vii. For *DiagnosedAtRiskDiabetes*:

Code or Value	Value Description	Count	Cumulative
1	Yes	873	873
2	No	5577	6450
7	Refused	0	6450
9	Don't know	19	6469
.	Missing	3301	9770

1. Convert the column to *String* type.
2. Replace '1.0' with 'True'.
3. Replace '2.0' with 'False'.
4. Replace '9.0' with an empty value,
5. Replace 'nan', with an empty value.
6. If there were any responses of ‘Refused’, they would be categorized as an empty value.

viii. For *NumMealsNotAtHomePerMonth*:

Code or Value	Value Description	Count	Cumulative
1 to 21	Range of Values	7320	7320

0	None	2362	9682
5555	More than 21 meals per week	5	9687
7777	Refused	1	9688
9999	Don’t Know	15	9703
.	Missing	472	10175

1. Convert the column to *int* type.
2. Replace 5555 with 21 the upper limit.
3. Replace 7777 with an empty value.
4. Replace 9999 with an empty value.
5. Fill the ‘na’ values with the median values.

ix. For *FamilyMonthlyPovertLevel*:

Code or Value	Value Description	Count	Cumulative
0 to 4.97	Range of Values	8002	8002
5	Value greater than or equal to 5.00	1107	9109
.	Missing	1066	10175

1. Convert missing values to the median.

x. For *Smoked100cigs*:

Code or Value	Value Description	Count	Cumulative
1	Yes	2579	2579
2	No	3532	6111
7	Refused	0	6111
9	Don't know	2	6113
.	Missing	1055	7168

1. Convert the column to *String* type.
2. Replace '1.0' with 'True'.
3. Replace '2.0' with 'False'.
4. Replace '9.0' with an empty value.
5. Replace 'nan' with an empty value.
6. If there were any responses of ‘Refused’, they would be categorized as an empty value.

xi. For *CurrentSmoker*:

Code or Value	Value Description	Count	Cumulative
1	Every day	992	992
2	Some days	240	1232
3	Not at all	1347	2579
7	Refused	0	2579
9	Don't know	0	2579
.	Missing	4589	7168

1. Convert the column to *String* type.
2. Replace '1.0' with 'True'.
3. Replace '2.0' with 'True'.
4. Replace '3.0' with 'False'.
5. Replace 'nan' with an empty value.
6. If there were any responses of ‘Refused’ or ‘Don’t Know’, they would be categorized as an empty value

xii. For *Gender*:

Code or Value	Value Description	Count	Cumulative
1	Male	5003	5003
2	Female	5172	10175
.	Missing	0	10175

1. Convert the column to *String* type.
2. Replace '1.0' with 'Male'.

3. Replace '2.0' with 'Female'.

xiii. For *Ethnicity*:

Code or Value	Value Description	Count	Cumulative
1	Mexican American	1730	1730
2	Other Hispanic	960	2690
3	Non-Hispanic White	3674	6364
4	Non-Hispanic Black	2267	8631
6	Non-Hispanic Asian	1074	9705
7	Other Race - Including Multi-Racial	470	10175
.	Missing	0	10175

1. Convert the column to *String* type.
2. Replace '1', with 'Mexican American'.
3. Replace '2', with 'Other Hispanic'.
4. Replace '3', with 'Non-Hispanic White'.
5. Replace '4', with 'Non-Hispanic Black '.
6. Replace '6', with 'Non-Hispanic Asian'.
7. Replace '7', with ' Other Race - Including Multi-Racial'.
8. If there was any missing data, remove the entry.

xiv. For *AdultEducationLevel*:

Code or Value	Value Description	Count	Cumulative
1	Less than 9th grade	455	455
2	9-11th grade (Includes 12th grade with no diploma)	791	1246
3	High school graduate/GED or equivalent	1303	2549
4	Some college or AA degree	1770	4319
5	College graduate or above	1443	5762
7	Refused	2	5764
9	Don't Know	5	5769
.	Missing	4406	10175

1. Convert the column to *String* type.
2. Replace '1.0' with 'No High School Diploma or GED'.
3. Replace '2.0' with 'No High School Diploma or GED'.
4. Replace '3.0' with 'High School Diploma or GED/equivalent'.
5. Replace '4.0' with 'Some College or Associates degree'.
6. Replace '5.0' with 'Bachelors or Higher'.
7. Replace '7.0' with 'Unknown'.
8. Replace '9.0' with 'Unknown'.
9. Replace 'nan' with 'Unknown'.

xv. For *AnnualHouseholdIncome*:

Code or Value	Value Description	Count	Cumulative
1	$ 0 to $ 4,999	273	273
2	$ 5,000 to $ 9,999	407	680
3	$10,000 to $14,999	639	1319
4	$15,000 to $19,999	658	1977
5	$20,000 to $24,999	880	2857
6	$25,000 to $34,999	1185	4042
7	$35,000 to $44,999	913	4955
8	$45,000 to $54,999	764	5719
9	$55,000 to $64,999	521	6240
10	$65,000 to $74,999	378	6618
12	$20,000 and Over	323	6941

13	Under $20,000	133	7074
14	$75,000 to $99,999	860	7934
15	$100,000 and Over	1781	9715
77	Refused	252	9967
99	Don't know	75	10042
.	Missing	133	10175

1. Convert the column to *String* type.
2. Replace '1.0' with 'less than 5,000'.
3. Replace '2.0' with '5,000 to 9,999'.
4. Replace '3.0' with '10,000 to 14,999'.
5. Replace '4.0' with '15,000 to 19,999'.
6. Replace '5.0' with '20,000 to 24,999'.
7. Replace '6.0' with '25,000 to 34,999'.
8. Replace '7.0' with '35,000 to 44,999'.
9. Replace '8.0' with '45,000 to 54,999'.
10. Replace '9.0' with '55,000 to 64,999'.
11. Replace '10.0' with '65,000 to 74,999'.
12. Replace '12.0' with '20,000 and Over'.
13. Replace '13.0' with 'Under 20,000'.
14. Replace '14.0' with '75,000 to 99,999'.
15. Replace '15.0' with '100,000 and Over'.
16. Replace '77.0' with 'Unknown'.
17. Replace '99.0' with 'Unknown'.
18. Replace 'nan.0' with 'Unknown'.

j. All columns and values are now renamed. Next all null values are dropped from the dataset.

k. Save and export the *DataFrame* to a CSV file named ‘U.S. NHANES Survey Data.csv’ in the relevant data container.

2) Transforming the CDC data on exercise:

a. Create a Python file and import the *pandas* library.

b. Read the CSV file into a *pandas DataFrame*.

c. Copy only the columns *YearStart*, *LocationDesc*, *Data_Value*, and *Sample_Size* into a new *DataFrame.*

d. Use the *dropna()* function to remove all null rows .

e. In the *Sample_Size* column, change the replace the commas with an empty string, then change the data types for *YearStart* and *Sample_Size* to *int*.

f. Export the *DataFrame* to a CSV file named “ExerciseData_2013_150min.csv” to the data container.

3) Transforming *U.S. Chronic Disease Indicators: Diabetes* from the CDC:

a. Read the relevant CSV file into its own *DataFrame* using *pandas.*

b. Filter the *DataFrame* by the *Question* column where the *Question* is 'Prevalence of diagnosed diabetes among adults aged >= 18 years'.

c. Keep only *YearEnd*, *LocationAbbr*, *LocationDesc*, *DataValue*, and *Stratification1* columns in the *DataFrame*.

d. Remove the locations you do not want from the *LocationDesc c*olumns.

i. We dropped “Guam”, “Virgin Islands”, “Puerto Rico”, and “District of Columbia”.

e. Fill all empty spots in the *DataValue* column with 0.

f. Export the *DataFrame* to a CSV file named “Diabetes Prevalence in the US by State and Demographic.csv” to the new container.

4) Transforming *Association of body mass index and age with incident diabetes in Chinese adults: a population-based cohort study* data:

a. Read the relevant CSV file into its own *DataFrame* using *pandas*.

b. Import *matplotlib.pylot and StandardScaler* from *sklearn.preprocessing*.

c. Drop the columns: *id*, *site*, *year of followup*, *Diabetes diagnosed during followup （1,Yes）*.

d. Rename the columns to be more understandable:

i. 'Age (y)': 'age', 'Gender(1, male; 2, female)': 'sex', 'height(cm)': 'height', 'weight(kg)': 'weight', 'BMI(kg/m2)': 'bmi', 'SBP(mmHg)': 'sbp', 'DBP(mmHg)': 'dbp', 'FPG (mmol/L)': 'fpg', 'Cholesterol(mmol/L)': 'chol', 'Triglyceride(mmol/L)': 'tg', 'HDL-c(mmol/L)': 'hdlc', 'LDL(mmol/L)': 'ldl', 'ALT(U/L)': 'alt', 'AST(U/L)': 'ast', 'BUN(mmol/L)': 'bun', 'CCR(umol/L)': 'ccr', 'FPG of final visit(mmol/L)': 'fpg_final', 'censor of diabetes at followup(1, Yes; 0, No)': 'diabetes', 'smoking status(1,current smoker;2, ever smoker;3,never smoker)': 'smoker', 'drinking status(1,current drinker;2, ever drinker;3,never drinker)': 'drinker', 'family histroy of diabetes(1,Yes;0,No)': 'fam_hist'

e. Cleaning the columns:

i. For *sex*, replace:
1. 1 with 'male',
2. 2 with 'female'

ii. For *height, sbp*, and *dbp*, remove all rows with null values.

iii. For *sbp*, *dbp*, *hdlc*, and *ldl*, remove rows more than 3 standard deviations from the mean.

iv. For *fpg*, *chol*, *tg*, *hdlc*, *ldl*, *alt*, *ast*, *bun*, *ccr*, and *fpg_final,* replace null values with the mean.

v. For *smoker*, replace:
1. Null values and 0 with 'no info'.
2. 1 with 'current smoker'.
3. 2 with 'former smoker'.
4. 3 with 'never smoker'.

vi. For *drinker*, replace:
1. Null values and 0 with 'no info'.
2. 1 with 'current drinker'.
3. 2 with 'former drinker'.
4. 3 with 'never drinker'.

vii. For *fam_hist*, remove all rows with null values.

viii. For *fam_*hist, replace:
1. 0 with 'no family history of diabetes'.
2. 1 with 'family history of diabetes'.

ix. For *diabetes*, remove all rows with null values.

x. For *diabetes* replace:
1. 0 with 'no'.
2. 1 with 'yes'.

f. Export the *DataFrame* to a CSV file named "data_clean.csv".

g. Load the file into the appropriate data container.

5) Transform the CGM data from the JAEB dataset:

a. Create a Python file and import the *pandas* library.

b. Read the "tblADataRTCGM_Blind_Baseline" CSV file from into a *DataFrame*.

c. Use the *drop_duplicates()* and *drop_na()* functions to remove duplicates and null rows.

d. Change the data type of the *DeviceDtTm* column to *datetime64*.

e. Export the *DataFrame* to a CSV file named "CGM_Data.csv".

f. Load the file into the appropriate data container.

6) Transform the Patient data from the JAEB data set:

a. Create a Python file and import the *pandas* library.

b. Read the "tblAptSummary" and "tblAAddlCEData" CSV's into their own *DataFrame*'s.

c. In the "Summary" *DataFrame*, sort values by *PtID*, and select only rows where the value in the *EduCareGvrP* is "Subject".

d. In the *Summary DataFrame*, select only the following rows: *PtID*, *Gender*, *Race*, *Ethnicity*, *EducareGvrPEdu*, *Height*, *Weight*, *AgeAsofRandDt*, *DurDiabetes*.

e. In the *CEData DataFrame*, sort values by *PtID*, and select only the columns *PtID* and *HxSmoke*.

f. Use *pandas.merge()* to merge the *Summary DataFrame* with the *CEData DataFrame* on the *PtID* column, with “inner” for the how parameter.

g. In the merged *DataFrame*, rename the following columns: *EduCareGvrPEdu* to *Education*, *AgeAsOfRandDt* to *Age*, *DurDiabetes* to *diabetesLen*, *HxSmoke* to *Smoke*.

h. Export the *DataFrame* to a CSV file named “Patient_Data.csv”.

i. Load the file into the appropriate data container.

7) Transforming *Food Security in the U.S. USDA (2019)* data from the USDA:

   a. Read the relevant CSV file into its own *DataFrame* using *pandas*.

   b. The *Year* column is currently a string that is spanning years. Rename it to the upper bound of the listed years:

      i. '2001-2003': 2003, '2004-2006': 2006, '2007-2009': 2009, '2010-2012': 2012, '2013-2015': 2015, '2016-2018': 2018

   c. Drop the irrelevant columns *Number of households (average)*, *Number of households interviewed*, *Food insecurity-Margin of error*, and *Very low food security-Margin of error* from the *DataFrame*.

   d. Convert the *Year* column to *numeric*.

   e. Export the *DataFrame* to a CSV file named “Food Insecurity.csv”.

   f. Load the file into the appropriate data cotainer.

8) Transforming *B06009: PLACE OF BIRTH BY EDUCATIONAL ATTAINMENT IN THE UNITED STATES* data from the Census Bureau American Community Survey:

   a. Read the relevant csv into its own *DataFrame* using *pandas*.

   b. Drop the irrelevant columns--*Margin of error* columns--and any rows referring to "District of Columbia” and “Puerto Rico”:

      i. df = df[df.columns.drop(list(df.filter(regex='Margin')))]

      ii. df = df[df.columns.drop(list(df.filter(regex='District')))]

      iii. df = df[df.columns.drop(list(df.filter(regex='Puerto')))]

   c. All rows after row index 4 are irrelevant and should be dropped.

   d. Rename the columns to be more usable:

      i. 'Label (Grouping)':'Label'

      ii. 'Alabama!!Estimate':'Alabama,

      iii. The rest of the columns have the same naming structure as shown in ii. and are renamed in the same manner for all 50 states.

   e. Pivot the table using the transpose function.

   f. Rename the new columns:

      i. 0:'Total Population', 1:'Less than high school graduate', 2:'High school graduate (includes equivalency)', 3:'Some college or associate degree', 4:'Bachelors Degree’

   g. Drop the row with *Label*.

   h. Export to an intermediary CSV so that the *States* will no longer be treated like an index.

   i. Import the intermediary CSV as a *DataFrame*.

   j. Rename the column 'Unnamed: 0':'State'.

   k. Export the *DataFrame* to a CSV file named “Education by state.csv”.

   l. Load the file into the appropriate data container.

9) Transforming *S1901: INCOME IN THE PAST 12 MONTHS (IN 2020 INFLATION-ADJUSTED DOLLARS)* from the Census Bureau American Community Survey:

   a. Read the relevant CSV into its own *DataFrame* using *pandas*.

   b. Drop the irrelevant columns using regex (*“Households!!Margin of Error, Families!!Estimate, Families!!Margin of Error, Married-couple families!!Estimate, Married-couple families!!Margin of Error, !!Nonfamily households!!Estimate, Nonfamily households!!Margin of Error”*), and anything to do with “Puerto Rico” and “District of Columbia”.

   c. Rename the columns to be more usable:

      i. 'Label (Grouping)':'Label'

      ii. 'Alabama!!Households!!Estimate':'Alabama'

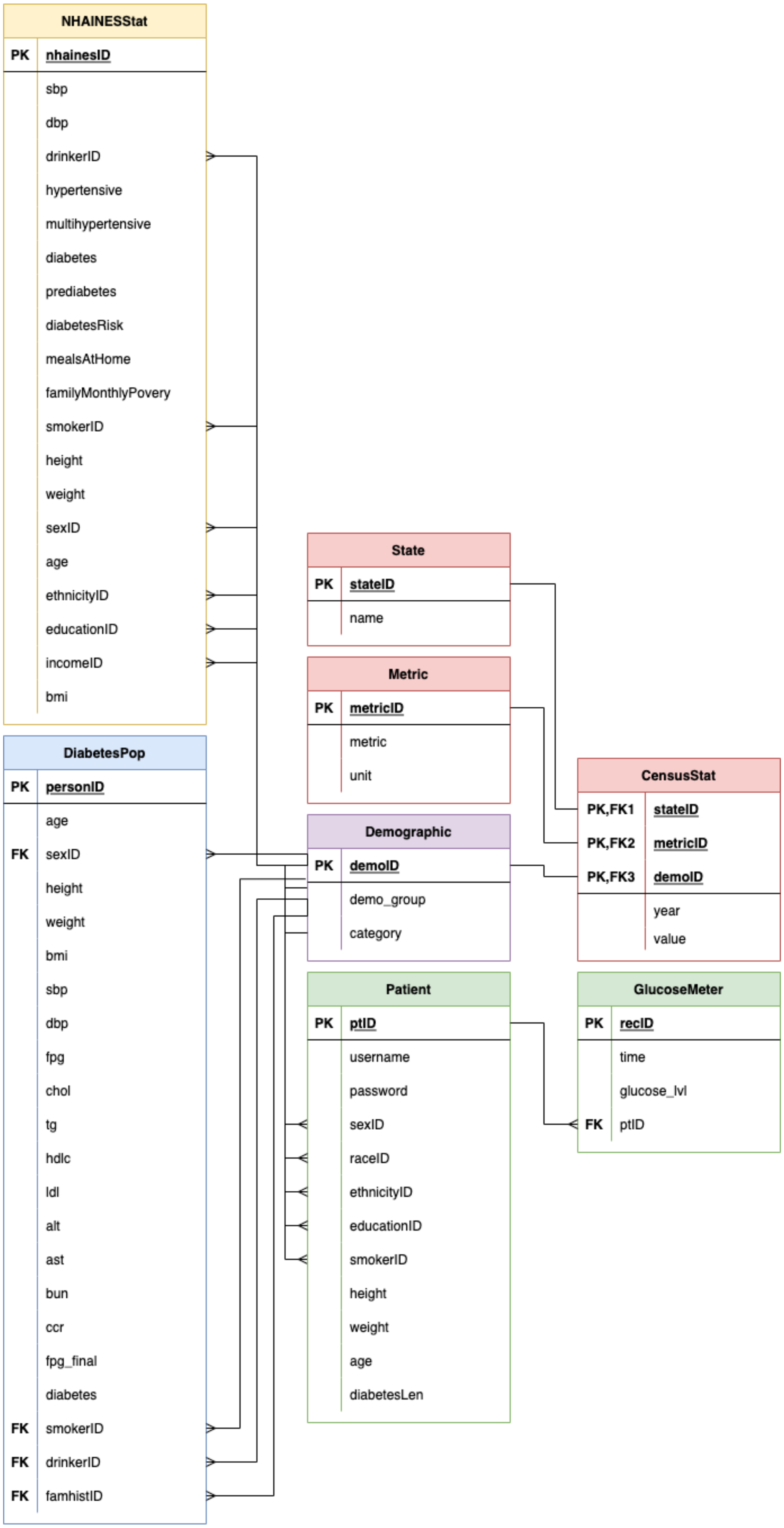
        iii. The rest of the columns have the same naming structure as shown in ii. and are renamed in the same manner for all 50 states.

d. All rows after row index 11 are irrelevant and should be dropped.
e. Pivot the table using the transpose function.
f. Rename the new columns:
    i. 0:'Total', 1:'Less than 10,000', 2:'10,000 to 14,999', 3:'15,000 to 24,999', 4:'25,000 to 34,999', 5:'35,000 to 49,999', 6:'50,000 to 74,999', 7:'75,000 to 99,999', 8:'100,000 to 149,999', 9:'150,000 to 199,999', 10:'200,000 or more', 11:'Median income (dollars)'
g. Drop the row with Label.
h. Export to an intermediary CSV so that the *States* will no longer be treated like an index.
i. Import the intermediary CSV as a *DataFrame*.
j. Rename the column 'Unnamed: 0':'State'.
k. Export the *DataFrame* to a CSV file named “Income Brackets by State.csv”.
l. Load the file into the appropriate data container.

Since your end goal will be to load your data in SQL Server, include table mappings that identify the source data and its destination.

Load

Prior to the loading process, a layout for the SQL database and its tables is pertinent for understanding the next steps. Below is our ER Diagram and the layout of the tables:



1) Creating the SQL database:
   a. Create a SQL database and add the users and the appropriate roles.
      i. For this project all members were assigned the role *db_owner*.
   b. For the exact SQL query to create the database visit:

        i. https://github.com/jackrlynn3/capstone-diabetes/blob/main/datasets/sql-queries/database-formation.sql

    c. There is a separate table for the data that is being streamed live while the other information is static.
    d. For the exact SQL query for creating that table visit:
        i. https://github.com/jackrlynn3/capstone-diabetes/blob/main/datasets/sql-queries/cgm%20creation.sql

2) Loading the NHANES data from Kaggle:
    a. The loading in its entirety can be seen at: https://github.com/jackrlynn3/capstone-diabetes/blob/main/datasets/kafka/NHAINESStat-Populator.ipynb
    b. Import the following into your Jupyter Notebook
        i. import uuid
        ii. from confluent_kafka.admin import AdminClient, NewTopic
        iii. from pyspark.sql.functions import col
        iv. from pyspark.sql.functions import lit
        v. from pyspark.sql.types import FloatType
        vi. from pyspark.sql.types import StringType
        vii. import pandas as pd
    c. Create a mount point the data container where the U.S. NHANES Survey Data.csv is stored.
    d. Define some helper functions to assist in the process. These functions are:
        i. *readInTable* - which reads in a table
        ii. *readInFile* – which reads in a file
        iii. *saveToTable* – which writes to a table
        iv. *formDictConverter* – which converts a table into a dictionary
        v. *addDemographics* – a series of helper functions that add the Demographic to the table, make sure there are no duplicates in the demographics table, and creates categories for the demographics.
    e. Populating the Demographic table from the NHANES Data
        i. Get the starting demoID from the Demographic table
        ii. Utilizing the addDemographics function, create all the categories for the 'AdultEducationLevel', 'AnnualHouseholdIncome', and 'Ethnicity' in the U.S. NHANES Survey Data.csv.
        iii. Change the demographics for female and male to make them start with an uppercase letter.
    f. Populating the NHAINESStat from the U.S. NHANES Survey Data.csv:
        i. Define some more helper functions to aid in the population of the NHAINESStat table
            1. *renameCols* – renames columns in a table
            2. *dropCols* removes columns we have found to be unnecessary since the cleaning process
            3. *catToID* converts demographics into *demoID*
    g. Read the file “U.S. NHANES Survey Data.csv”
    h. Rename the following columns:
        i. 'SurveyID', 'SystolicBP', 'DiastolicBP', '12drinksInaYear', 'BeenDiagnostedHypertenisve', 'MultipleHypertensionDiagnosis', 'DiagnosedDiabetic', 'DiagnosedPrediabetic', 'DiagnosedAtRiskDiabetes', 'NumMealsNotAtHomePerMonth', 'FamilyMonthlyPovertLevel', 'Smoked100cigs', 'Height', 'Weight', 'Gender', 'Age(yrs)', 'Ethnicity', 'AdultEducationLevel', 'AnnualHouseholdIncome', 'BMI' to 'nhainesID', 'sbp', 'dbp', 'drinkerID', 'hypertensive', 'multihypertensive', 'diabetes', 'prediabetes', 'diabetesRisk', 'mealsAtHome', 'familyMonthlyPoverty', 'smokerID', 'height', 'weight', 'sexID', 'age', 'ethnicityID', 'educationID', 'incomeID', 'bmi' respectively.
    i. Drop the columns:
        i. '_c0', 'DrinksInLastYear', 'CurrentSmoker'
    j. Add the drinker ID from where True = '12+ alc. drinks/yr' and False = '<12 alc. drinks/yr'.

k. Use the *catToID* to populate the category *drinkerID.*

l. Add the *smokerID* from where True = ' Smoked 100+ cigs.' and False = ' Smoked <100 cigs'.

m. Use the *catToID* to populate the category *smokerID*.

n. Using *catToID*, populate *sexID*, *IncomeID*, *educationID*, and *ethnicityID*.

o. Use *saveToTable* to save the changes to the SQL database.

3) Loading the *CDC Data on Exercise*, *Diabetes Prevalence in the U.S. By State and Demographic*, *Education by State*, I*ncome Brackets by State*, and *Food Insecurity* are all done in the same notebook and will be collectively addressed here.

   a. The loading in its entirety can be seen at: https://github.com/jackrlynn3/capstone-diabetes/blob/main/datasets/kafka/CensusStat-Populator.ipynb

   b. Import the following into your Jupyter Notebook

      i. *import uuid*

      ii. *from confluent_kafka.admin import AdminClient, NewTopic*

      iii. *from pyspark.sql.functions import col*

      iv. *from pyspark.sql.functions import lit*

      v. *from pyspark.sql.types import FloatType*

      vi. *from pyspark.sql.types import StringType*

      vii. *import pandas as pd*

   c. Mount to the container that is storing the previously stored CSV's.

   d. Here we write a function to populate the metrics table as mentioned in the ER Diagram.

      i. Connect to the table.

      ii. Get the starting value.

      iii. Iterate through each metric.

         1. Get the metric and measurements for each iteration.

         2. Check to see if the metric is already in the database.

         3. Convert the metric to dictionary entry.

      iv. Convert the metrics to a *DataFrame*.

      v. Overwrite and Save to the SQL.

      vi. Append the metrics to the SQL.

      vii. Return the dictionary with the metric conversions.

   e. Add all the metrics to the database:

      i. Metrics = 'diabetes', 'highest education', 'exercise (150+ min/wk)', 'food security', 'very low food security', 'income bracket'

      ii. Measurements = 'percentage', 'percentage', 'percentage', 'percentage', 'percentage', 'percentage'

   f. Here we write a function to populate the State Table in the SQL database.

      i. Read in the Diabetes Prevalence in the US by State and Demographic.csv.

      ii. Extract the states data.

      iii. Add "Washington DC".

      iv. Convert to a *DataFrame*.

      v. Save the state data to the State SQL table.

      vi. Return a dictionary with the state conversions.

   g. Add the states that are present to the database.

   h. Add a value for "National" data and "Washington DC".

   i. Populate the demographic database with data from the "Diabetes Prevalence in the US by State and Demographic.csv", "Education by state.csv", and the "Income Brackets by State.csv" using the *addDemogrpahics* function that was utilized in the NHANES Data Loading Procedure.

      i. From the "Diabetes Prevalence in the US by State and Demographic.csv", get categories and for *Stratification1*:

        1. Iterate through each category and remove any duplicates.
        2. Remove *Overall* to handle it later.
        3. Add all demographics under the race/ethnicity category.
        4. Save to the table.
    ii. From the “Education by state.csv”, remove the *columns _c0*, *State*, and *Total Population*.
        1. Iterate through each category that remains and remove any duplicates.
        2. Add all new demographics under the education level category.
        3. Save to the table.
    iii. Add a 'General' category.
    iv. From “Income Brackets by State.csv”, remove the columns *_c0*, *State*, *Total*.
        1. Iterate through each category that remains and remove any duplicates.
        2. Add all new demographics under the income bracket category.
        3. Save to the table.
    v. Add in uppercase version of “Male” and “Female”.
    vi. Add "Overall” to track general data.

j. We will be using the same helper functions in Populating the *NHAINESStat* to populate the data tables in this notebook:
    i. *renameCols* – renames columns
    ii. *dropCols* - drops columns
    iii. *addCols* – adds columns
    iv. *catToID* - converts demographics into *demoID*

k. For “Diabetes Prevalence in the US by State and Demographic”:
    i. Read in the “Diabetes Prevalence in the US by State and Demographic.csv”.
    ii. Convert all previously found categories to ID using *catToID* and additionally *Stratification1*, then create the diabetes *DataFrame*.
    iii. Rename the columns ['LocationAbbr', 'Stratification1', 'YearEnd', 'DataValue'], to ['stateID', 'demoID', 'year', 'value'].
    iv. Drop the columns *LocationDesc*, *personID*, and *_c0*.
    v. Add in the appropriate metric ID for the value in the dataset.
    vi. Save the newly generated table to the SQL database as *CensusStat*.

l. For Education by state:
    i. Read in the “Education by state.csv” file.
    ii. Using *catToID*, get the columns to iterate through
    iii. For each category individually:
        1. Get the percentage of the population.
        2. Convert the state values and create the education *DataFrame*.
        3. Add in the other columns *metricID*, *demoID*, and *year*.
        4. Populate these columns with the highest education metric id, the demographic ids in the Education by State set, and 2017 respectively.
        5. Save the new table to SQL.

m. For Exercise Data:
    i. Read in the “ExerciseData_2013_150min.csv” file.
    ii. Drop unnecessary columns *Low_Confidence_Limit*, *High_Confidence_Limit,* and *Sample_Size*.
    iii. Rename columns to correct names: ['YearStart', 'Data_value', 'LocationDesc'] to ['year', 'value', 'stateID'].
    iv. Using *catToID*, convert state values and create the exercise *DataFrame*.
    v. Add in the columns *metricID* and *demoID*.
    vi. Populate these columns with the exercise metric ids, and the general demo IDs.
    vii. Drop the column *_c0.*

viii. Save the table to the *CensusStat* table.

n. For the Food Security data:

i. Read in the “Food Insecurity.csv” file.

ii. Convert the state values and create the food *DataFrame*.

iii. Using a for loop, split the *DataFrame* into two parts 'Food insecurity prevalence', and 'Very low food security prevalence'.

1. Get a *DataFrame* with columns *year*, *col*, and *StateID*.
2. Rename the column *col* with *value*.
3. Add the appropriate metrics to each column.
4. Create a metric id for both 'Food insecurity prevalence', and 'Very low food security prevalence'.

iv. Save the table to the *CensusStat* table.

o. For the Income Data:

i. Read in the “Income Brackets by State.csv” file.

ii. Using *catToID*, convert state values and create the income *DataFrame*.

iii. Get the categories that will be iterated through.

1. Remove *_c0*, *Total*, *State*, and *Median income (dollars)*

iv. Iterate through each category:

1. Create a *DataFrame* of each category and its corresponding state for each iteration.
2. Add in the columns *metricID*, *demoID*, and *year*, filled respectively with the created metric id for each income bracket, the generated demographic id for each category, and 2017.
3. Remove the percentage mark from the string in values and create a new *DataFrame*.
4. Join the *DataFrame* that was created to the other created *DataFrame’s* in the iteration.
5. Update the table in the SQL database.

4) Loading the CGM data:

a. The loading in its entirety can be seen at: https://github.com/jackrlynn3/capstone-diabetes/blob/main/datasets/kafka/GlucoseMeter-Populator.ipynb

b. Import the following into your Jupyter Notebook:

i. import uuid

ii. from confluent_kafka.admin import AdminClient, NewTopic

iii. from pyspark.sql.functions import col

c. Mount to the data container.

d. Load in the file “CGM_Data.csv”.

e. Rename the columns to fit the database:

i. 'RecID' to 'recID'

ii. 'PtID' to 'ptID'

iii. 'DeviceDtTm' to 'time'

iv. 'Glucose' to 'glucose_lvl'

f. Drop *_c0* as it is unnecessary.

g. Save the file to the SQL database.

5) Loading the data from *Association of body mass index and age with incident diabetes in Chinese adults: a population-based cohort study*:

a. The loading in its entirety can be seen at: https://github.com/jackrlynn3/capstone-diabetes/blob/main/datasets/kafka/DiabetesPop-Populator.ipynb

b. Import the following into your Jupyter Notebook:

i. import uuid

ii. from confluent_kafka.admin import AdminClient, NewTopic

iii. from pyspark.sql.functions import col

c. Mount to the data container.

d. Load in the file “chinese-diabetes-clean.csv”.

e. Create a function to format the category data.

f. Create new categories for the *demographic* table.

i. From ['sex', 'smoker', 'drinker', 'fam_hist'] to ['sex', 'smoker', 'drinker', 'family history']

g. Save the demographic data to the *Demographic* SQL table.

h. Populate the *DiabetesPop* Table:

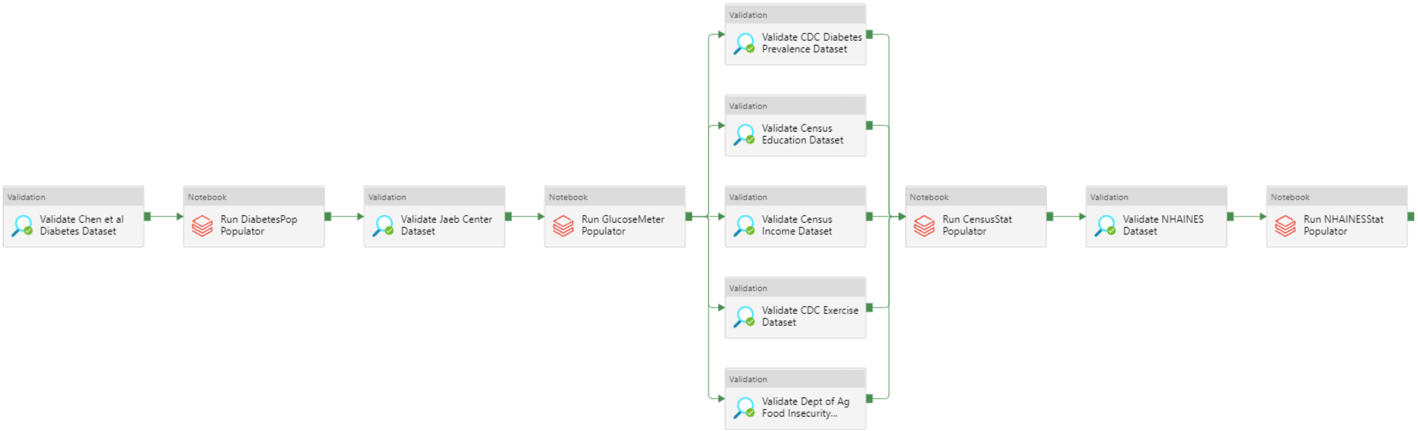
i. Create a dictionary to convert demographic groups to ids.

ii. Format the *DataFrame* to reference the other tables.

iii. Save the *DataFrame* to the SQL database.

6) Set up the data factory to populate the database through Azure.

a. The data factory for the transformed static data looks as below:


Validation
Validate Chen et al Diabetes Dataset
Notebook
Run DiabetesPop Populator
Validation
Validate Jaeb Center Dataset
Notebook
Run GlucoseMeter Populator
Validation
Validate CDC Diabetes Prevalence Dataset
Validation
Validate Census Education Dataset
Validation
Validate Census Income Dataset
Validation
Validate CDC Exercise Dataset
Validation
Validate Dept of Ag Food Insecurity...
Notebook
Run CensusStat Populator
Validation
Validate NHAINES Dataset
Notebook
Run NHAINESStat Populator

7) CGM Producer

a. Import the following into your Jupyter Notebook:

i. from confluent_kafka import Consumer

ii. from time import sleep

iii. import uuid

iv. from confluent_kafka import Producer, Consumer, KafkaError, KafkaException

v. import json

vi. from confluent_kafka.admin import AdminClient, NewTopic

vii. from pyspark.sql.functions import when

viii. from pyspark.sql.functions import col

ix. import random as r

b. Mount to the data container where the CGM data is stored.

c. Read in the data.

d. Select patients that with patient IDs 1, 14, and 33 since they have all their data points.

e. Produce the first 24 hours of data so that there is data populated.

f. Start producing the next 24 hours of data, sending a line every 10 seconds.

8) CGM Consumer

a. Import the following into your Jupyter Notebook:

i. from confluent_kafka import Consumer

ii. from time import sleep

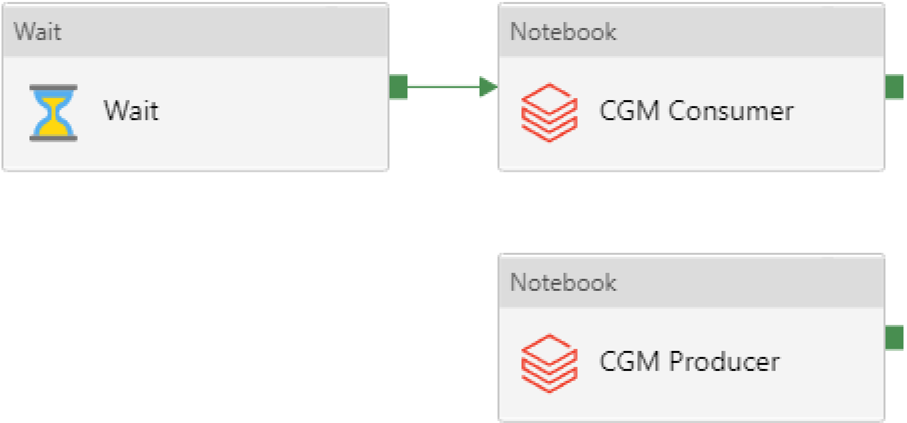
iii. import uuid

iv. from confluent_kafka import Producer, Consumer, KafkaError, KafkaException

v. import json

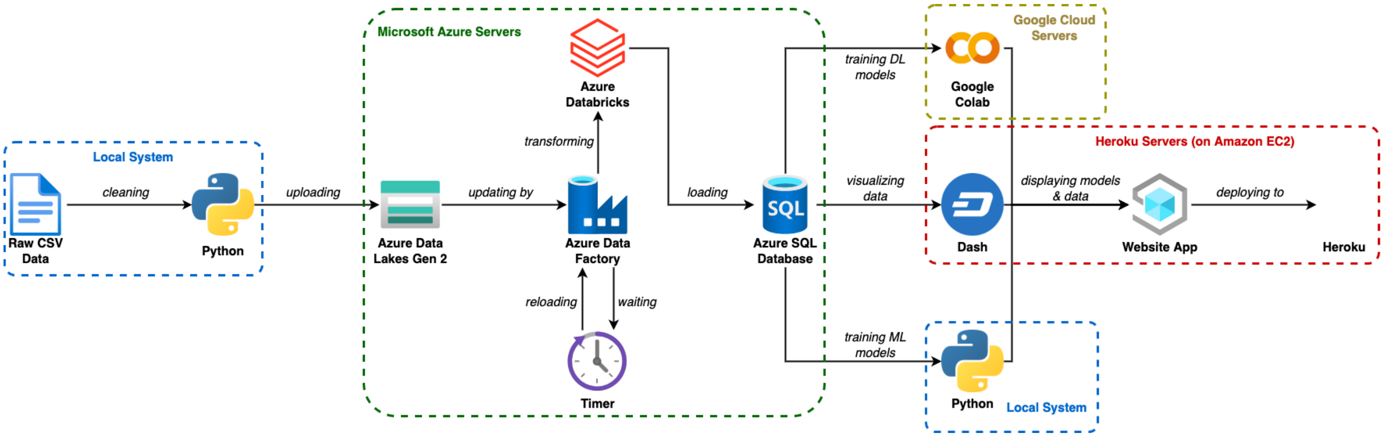
b. Create a Kafka consumer that consumes all the messages created from the previous Kafka Topic and loads them to the database *CGM_Stream* table while updating them.

9) Streaming data factory:



Conclusion

After completion of these processes this is the structure of the network diagram for performing our analysis and the creation of our presentation.


Local System
Raw CSV Data
cleaning
Python
uploading
Microsoft Azure Servers
Azure Data Lakes Gen 2
updating by
Azure Data Factory
transforming
Azure Databricks
reloading
waiting
Timer
loading
Azure SQL Database
training DL models
Google Cloud Servers
Google Colab
visualizing data
Heroku Servers (on Amazon EC2)
Dash
displaying models & data
Website App
deploying to
Heroku
training ML models
Python
Local System