**INFO 5770 - GROUP PROJECT - PHASE 2:** **DATA PREPROCESSING**

**TITLE: "A COMPARATIVE ANALYSIS OF RESPRATORY DICEASE CONCENTRATING ON ASTHMA AND INFLUENZA"**

**GROUP MEMBERS:**

ARJUN PALANISWAMY, EUID: ap1839

AAVULA USHA, EUID: ua0095

KAUSHIKAN DINAKARAN, EUID: kd0596

ROHITH SINGH THAKUR, EIUD: rt0674

**COLUMNS DESCRIPTION:**

**1. DUPERSID:** A unique identifier for each individual in the survey.

**2. SEX:** The respondent’s gender, typically coded as 1 for male and 2 for female.

**3. DOBMM and DOBYY:** The month and year of birth for each individual. DOBMM represents the month (01-12), and DOBYY the year (e.g., 1990).

**4. AGEDIAG:** Age at which a specific condition (e.g., asthma) was diagnosed. Missing values may be coded as negative or null.

**5. MARRY31X:** Marital status of the respondent, where common values include:

- 1 for married,

- 2 for widowed,

- 3 for divorced,

- 4 for separated,

- 5 for never married, and

- 6 for living with a partner.

**6. RACEV2X:** The racial/ethnic background of the respondent. This may include codes like:

- 1 for White,

- 2 for Black,

- 3 for Asian,

- 4 for other, and

- 5 for multiple races.

**7. REGION42:** Region of residence in the U.S., often coded as:

- 1 for Northeast,

- 2 for Midwest,

- 3 for South, and

- 4 for West.

**8. TOTEXP19:** Total annual healthcare expenditure for the respondent in the survey year 2019. This includes costs covered by insurance, out-of-pocket payments, etc.

**9. ICD10CDX:** The ICD-10 diagnosis code, representing specific health conditions or diagnoses. For example, J45 represents asthma.

**10. ADAPPT42:** Number of medical appointments attended in the last year.

**11. ADRISK42:** A risk assessment score for conditions related to asthma or other health concerns, indicating the perceived risk level based on the respondent’s health status.

**12. ASATAK31:** Indicator of whether the respondent had an asthma attack in the last 12 months:

- 1 for "Yes,"

- 0 for "No,"

- Negative or null values might represent missing data.

**13. ASDALY31:** Indicates whether the respondent takes daily preventive asthma medication:

- 1 for "Yes,"

- 2 for "No."

**14. ASSTIL31:** Represents if the respondent currently has asthma:

- 1 for "Yes,"

- 0 for "No."

**15. ASTHEP31:** The time since the last asthma episode, typically coded in days, weeks, or months since the last occurrence.

**16. ASACUT31:** Indicates the use of a quick-relief inhaler within the last three months:

- 1 for "Yes,"

- 0 for "No."

**17. ASPREV31:** Shows if preventive asthma medication has ever been used, often a binary indicator:

- 1 for "Yes,"

- 0 for "No."

**18. ASPKFL31:** Indicates possession of a peak flow meter (a device for asthma monitoring):

- 1 for "Yes,"

- 0 for "No."

**19. MCDOC19:** A variable related to Medicaid or doctor visits in 2019, tracking access or frequency of visits.

**20. ADHECR42:** A measure of healthcare adherence, which might reflect compliance with medical recommendations for health conditions.

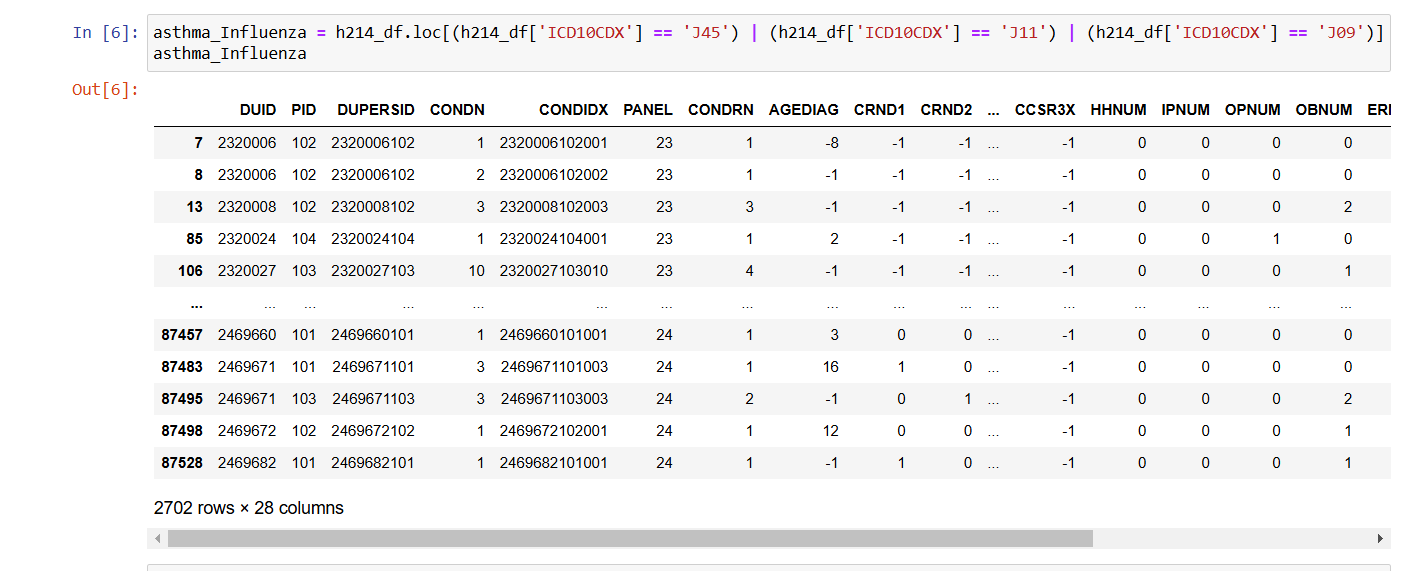
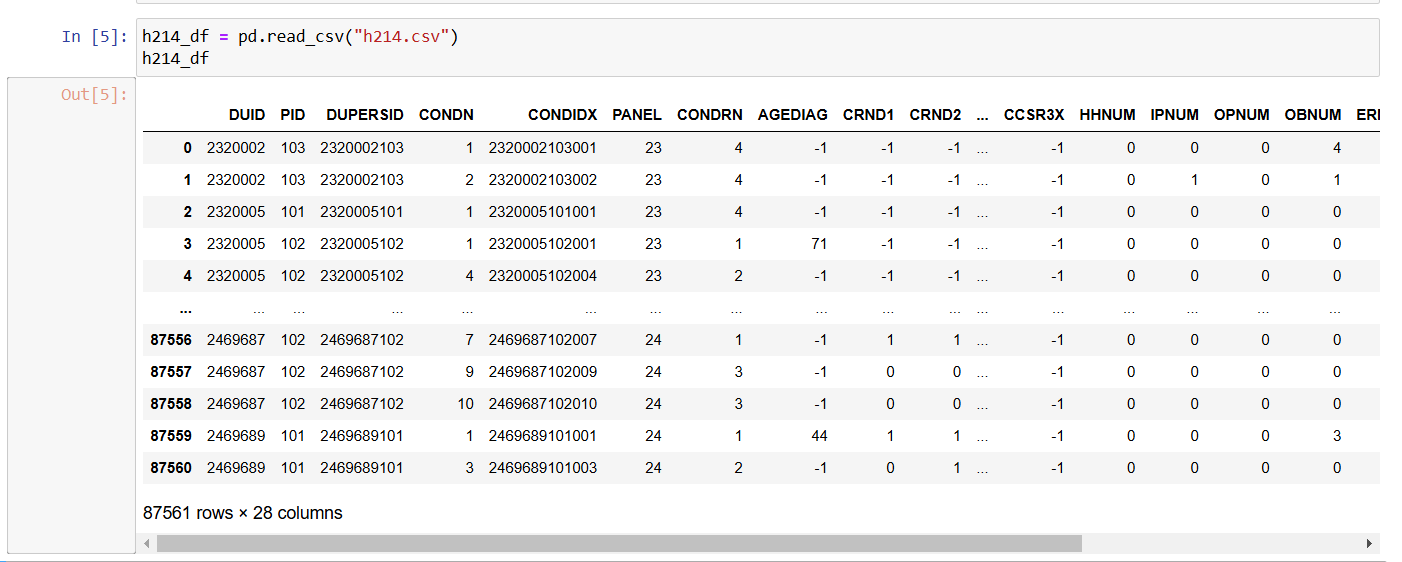
**21. Current\_Age:** The current age of the respondent.

**22. Flu shot taken:** Indicates whether the respondent received a flu shot in the past years:

- 1 for "Yes,"

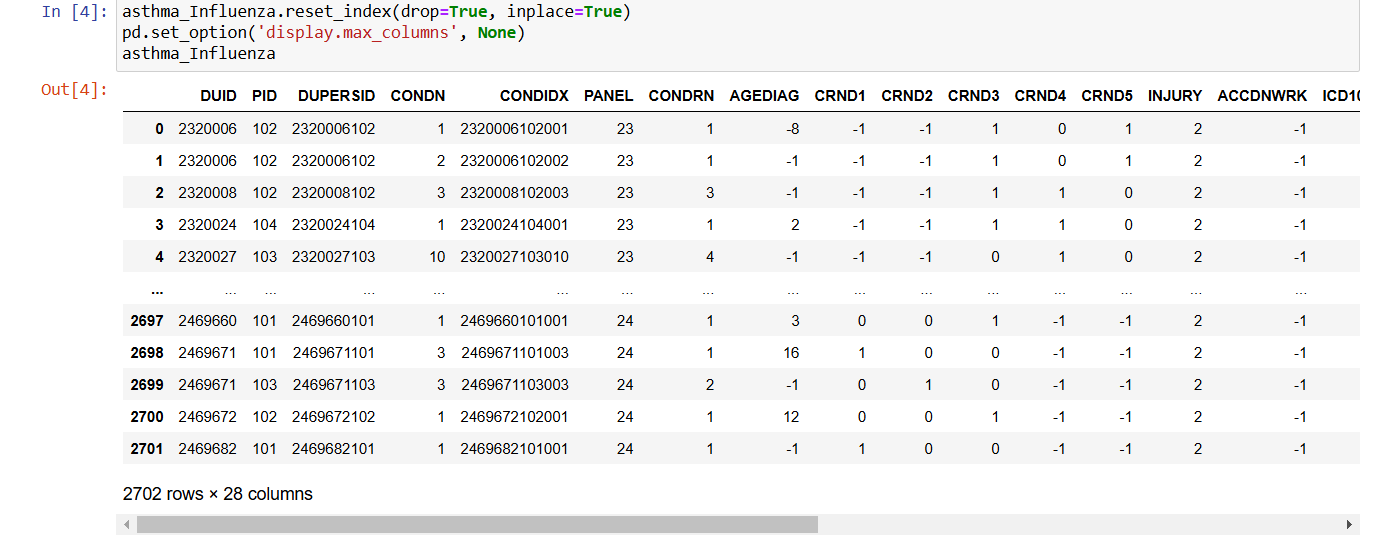
- 0 for "No."

**DATA LOADING AND FILTERING:**

****

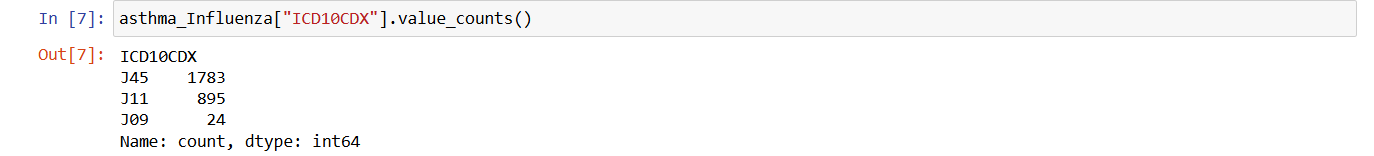
The dataset gets loaded to a pandas DataFrame. The script filters out records of asthma - ICD10CDX code J45 and influenza - ICD10CDX codes J11 and J09. These are ICD-10 codes for classifications of diseases and are used in this medical data to represent asthma-J45 and influenza-J11/J09. It has totally 2702 records and 28 columns.

**INDEX RESETTING AND COLUMN DISPLAY:**

****

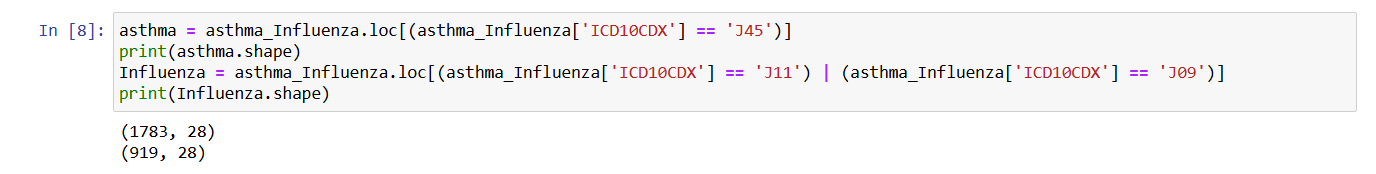
The data is filtered and the index reset to ensure records are clean and in order. Additional configuration is made to pandas, allowing it to 2702 rows and 28 columns.

**COUNTING OCCURRENCES OF CONDITIONS:**

****

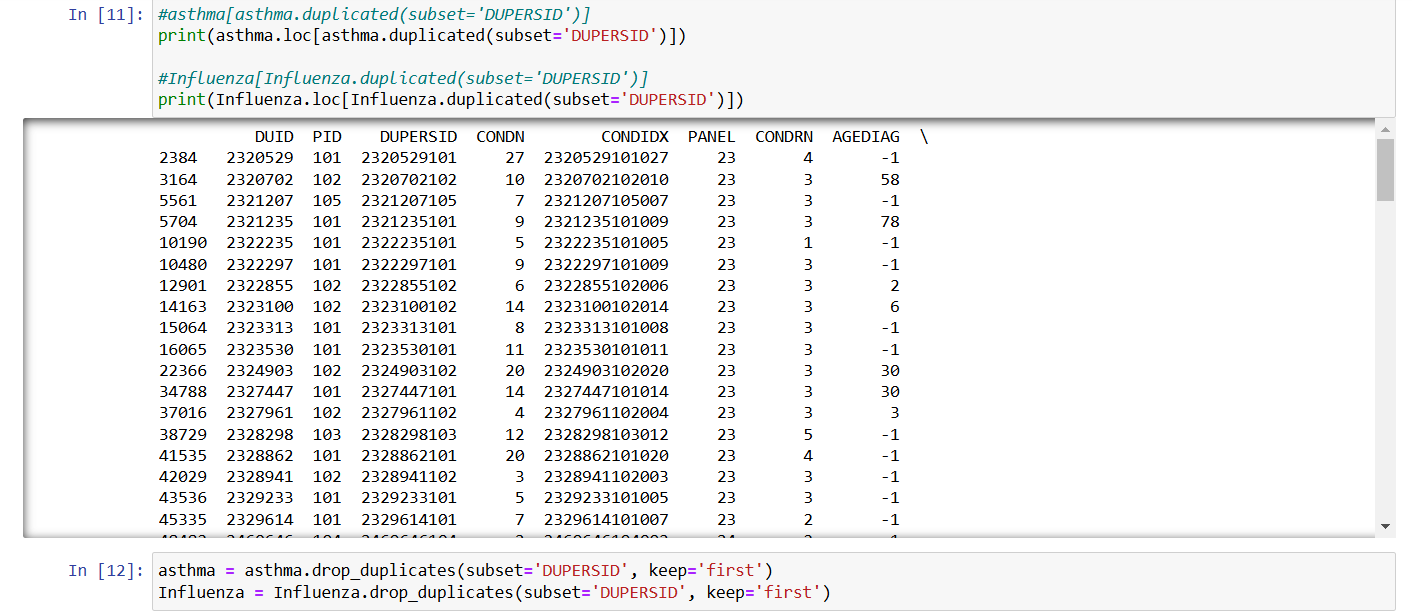
The step follows through and counts how many times each condition occurs in the dataset. This gives an idea of the frequency of asthma versus the influenza cases as a simple insight into what this dataset is made up of. In summary, it has 1783 record of asthma and 919 record of influenza.

**SEGMENTATION OF ASTHMA AND INFLUENZA DATA**

****

With this, after cleaning and filtering, the data is divided into two separate DataFrames comprising asthma cases only and influenza cases only. The shape for each is printed to show the number of patient records identified for each condition.

**CHECK AND REMOVE DUPLICATES:**

****

It checks for duplicate records using the DUPERSID column, which most likely forms a unique patient identifier. It then removes any such duplicate records, retaining only the first occurrence of each DUPERSID. The step ensures that each patient appears once; otherwise, if there are duplicates, the result would be wrong.

**MERGING CLEANED DATA:**

****

After removing duplicates, the cleaned DataFrames are combined into one: one for asthma and one for influenza. The consolidated dataset will contain all unique patient records from these two conditions into one, thus making further analysis or a comparison between the two groups quite easy. It has total of 2667 records and 28 columns.

**JOINING H216.CSV**

**A screenshot of a computer

Description automatically generated**

* The code first reads, into a DataFrame called h216\_df, a new data set, h216.csv.
* It performs an inner merge next, between new\_df, which contained combined asthma and influenza data, and h216\_df on the DUPERSID column as the key.
* An inner merge will only include rows where DUPERSID exists in both datasets meaning only patients that exist in both datasets will be retained in the merged DataFrame: selected\_df.
* This step incorporates more patient attributes from h216.csv in order to extend the already existing dataset to one that is more informative.

**SELECTING SPECIFIC COLUMNS**

**A screenshot of a computer

Description automatically generated**

Here, the code narrows down the merged DataFrame to a set of relevant columns. These columns are selected to focus on important patient characteristics and health-related factors. The columns chosen include:

* **DUPERSID**: Patient identifier.
* **SEX**: Gender of the patient.
* **DOBMM**, **DOBYY**: Month and year of birth.
* **AGEDIAG**: Age at the time of diagnosis.
* **MARRY31X**: Marital status.
* **RACEV2X**: Racial/ethnic category of the patient.
* **REGION42**: Geographic region where the patient resides.
* **TOTEXP19**: Total healthcare expenditure in 2019.
* **ICD10CDX**: The ICD-10 diagnosis code (asthma or influenza).
* **ADAPPT42**, **ADRISK42**: Indicators of appointment adherence and health risk.
* **ASATAK31** to **ASPKFL31**: Various asthma-specific attributes, possibly indicating attack frequency, daily activity limitations, severity of illness, prevention measures, and peak flow usage.
* **MCDOC19**: Number of doctor visits in 2019.
* **DSFL1853** to **DSFL2053**: Dietary or health survey responses related to specific health factors.
* **ADHECR42**: Indicator for adherence to emergency care recommendations.

Selecting these specific columns ensures that the dataset is manageable and focused on key variables needed for analysis.

**DESCRIPTIVE STATISTICS**

**A screenshot of a computer

Description automatically generated**

This line generates a summary of the descriptive statistics for the selected columns. The describe() function, with include='all', provides:

* Count: Number of non-null entries in each column.
* Unique: Number of unique values.
* Top: Most frequent value.
* Frequency: Frequency of the top value.
* Mean, Standard Deviation (std), Min, 25th percentile (25%), 50th percentile (median), 75th percentile (75%), and Max: Summary statistics for numeric columns.

**PURPOSE AND ANALYSIS:**

* This program, below, cleanses and joins data on asthma and influenza patients. It does so by combining additional information from another data set known as h216.csv.
* In focusing on key demographic and socioeconomic variables, as well as some health-related variables, the resulting DataFrame called selected\_df is quite substantial to study the patient characteristics, health care utilization, and selected specific health outcomes associated with these two respiratory conditions.
* Descriptive statistics provide an overview of the dataset with regards to demographic and health-related factors on central tendencies, variability, and distribution.

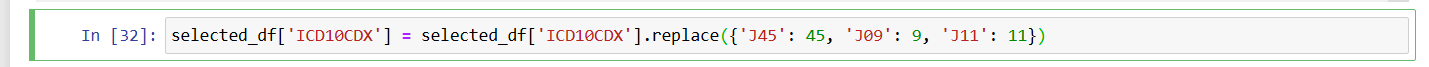
**DISPLAYING THE ENTIRE DATAFRAME**

**A screenshot of a computer

Description automatically generated**

As explained earlier, this line enables the printing of all rows without truncation when selected\_df is printed; it aids in visual scanning of the full dataset.

**CONVERTING ICD-10 CODES TO NUMERIC FORM**

****

The replace function below is utilized to transform the numeric equivalent values for each ICD-10 diagnosis code.

* 'J45' (Asthma) - Maps to 45
* 'J09' (Influenza, avian or other animal-origin influenza) - Maps to 9
* 'J11' (Influenza, other and unspecified influenza) - Maps to 11.

The reason is that this numerical conversion will ease the job of performing statistical analysis, particularly when some numerical computation or modeling is done.

**HANDLING NEGATIVE VALUES AS MISSING DATA AND CALCULATING NON-MISSING AND MISSING VALUE COUNTS**

**A screenshot of a computer

Description automatically generatedA white background with black and white clouds

Description automatically generated**

* The mask function will replace all values in asthma\_selected\_missing that are less than 0 with, that is, it marks these values as missing.
* Generally, in many health datasets, negative values can indicate missing, unknown, or inapplicable data, rather than actual numerical values. This would ensure such entries get appropriately marked as missing.
* Output: The Nun values are in the DataFrame where there were negative values originally.
* The isnull().sum() method counts the number of missing values in any column.
* row\_count and column\_count get the total number of rows and columns in new\_df (the combined dataset of asthma and influenza).

The code prints:

* Missing values count: A list of columns sorted by the number of missing values, from most to least missing entries.
* Available values count : For each column, total rows, i.e., the row\_count minus the count of missing values. This describes how many non-missing values are there for each attribute.
* The column DSFL1853 has 250 missing values; hence, 750 entries are available for this column, that is, 1000 - 250 entries.
* SEX and DUPERSID columns have no missing values, so all 1,000 entries are available.

**CATEGORIZATION OF COLUMNS**

**A screenshot of a computer

Description automatically generated**

**Objective:** To identify if columns are categorical or numerical to enable the correct type of data analysis.

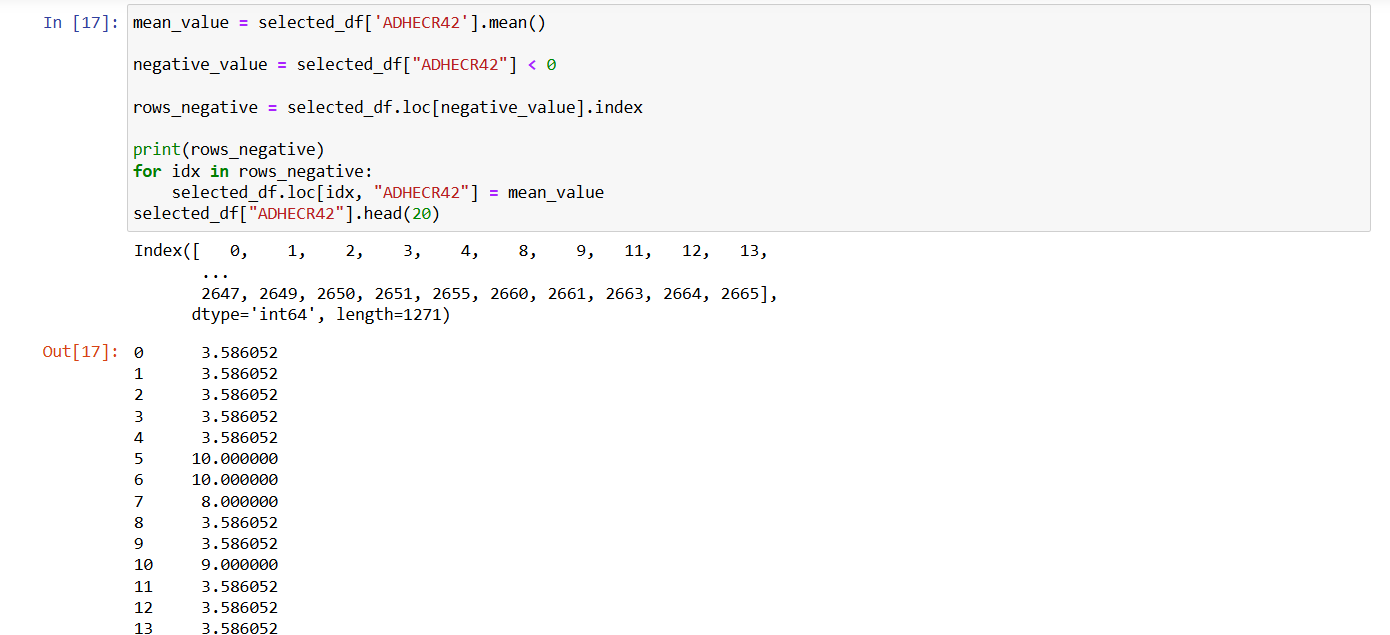
**Method:**

* Calculate the number of unique values for each column using nunique().
* Any column that has a value count of 12 or less is considered categorical, including nominal and ordinal data types.
* Above 12 unique values categorized into numerical, typically represents continuous data.

**Output:**

The categorization lets the analyst understand the distribution of data and further assists in the selection of proper statistical or machine learning models for the analysis of data.

**2. HANDLING NEGATIVE VALUES IN ADHECR42**

****

**Objective:** This section is dedicated to data quality by processing negative values occurring in the ADHECR42 column. Most of these values might have been anomalies or placeholder values for missing data.

**Method:**

* Calculate mean of variable ADHECR42, excluding those negative or anomaly values.
* Identify the rows that contain negative values within the ADHECR42 column and replace those rows with the mean calculated. Rationale: Replacing negative values with the mean keeps the data from skewing it and maintains a consistent range for analysis. Result: Clean ADHECR42 column with outliers managed appropriately.

**3. AGE CALCULATION FROM BIRTH DATE OBJECTIVE:**

**A computer screen shot of a computer code

Description automatically generated**

An accurate age column, Current\_Age, given the birth year, DOBYY, and birth month, DOBMM.

**Approach:**

* First, DOBYY and DOBMM are combined into Birth\_Date. The day is set to 1st. Then, datetime.now() is used to calculate current date. Under the lambda function, it calculates age and computes the difference between current year and birth year, subject to whether the birth month and day have passed this year.

**Rationale:**

* By directly providing Current\_Age, raw birth date information is not needed in analyses that require only age information.
* This cleaned the dataset by adding a new column entitled Current\_Age and then removing the Birth\_Date column as it was now an interim column that served little further purpose for simplification in this dataset.

**4. CREATION OF FLU SHOT INDICATOR, FLU\_SHOT\_TAKEN:**

**A screenshot of a computer screen

Description automatically generated**

**Objective:** It would create one column that is indicative of a participant receiving the flu shot based on three different columns, namely DSFL1853, DSFL1953, and DSFL2053.

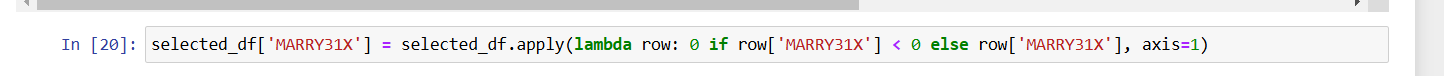
**Method:**

* The apply() function sets the Flu\_shot\_taken value to 1 if any of the DSFL1853, DSFL1953, or DSFL2053 columns have a value of 1, meaning the flu shot was taken. If none have 1, it assigns 2.
* To reduce noise in the dataset, original columns related to flu shot status are dropped.

**Rationale:** This simplifies all further analyses by consolidating flu shot data into one indicator column.

**Result:** Added a new column Flu\_shot\_taken which indicates whether or not the participant took a flu shot to track flu status easily.

**NORMALIZATION OF MARRY31X COLUMN**

****

This will ensure that all values in the column MARRY31X will be non-negative.

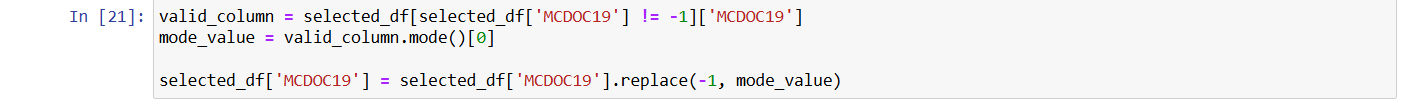
**Approach:**

The apply() function will be used along with a lambda to replace any negative value in the column MARRY31X with 0. This is an assumption that the records are negative as a placeholder for missing data or undefined states.

**Justification:**

* Normalizing the values of the column MARRY31X would ensure, at analysis time, the interpretation is consistent. Values indicating unknown or invalid data would be standardized as 0.
* A cleaned MARRY31X column with all negative values replaced by 0.

**IMPUTATION OF MISSING VALUES IN MCDOC19**

****

**Objective:** Replace placeholder values (-1) of MCDOC19 column with most frequent (mode) valid value.

**Approach:**

* A slice of selected\_df, comprising of rows where MCDOC19 is not -1, is obtained.
* The mode of MCDOC19 in the slice is calculated.
* The replace() function is used on MCDOC19 to replace -1 values with the calculated mode.

**Rationale:** Imputing with the mode provides assurance that the replacement is representative of the most common occurring valid value, therefore remaining true to the preservation of data integrity and minimizing bias.

**Result:** There are no longer any -1 values in the MCDOC19 column, further enhancing the reliability of the data.

**HANDLING MISSING VALUES IN REGION42 BY RACEV2X**

**A screenshot of a computer

Description automatically generated**

**Objective:** Fill in negative or placeholder values for REGION42 using the mode of REGION42 for each group in RACEV2X.

**Method:**

* First, groupby() and agg() with a lambda function compute the mode of REGION42 for each group defined by RACEV2X.
* The computed mode prints out for reference.
* An apply() function updates REGION42, replacing negative values with the respective group mode.

**Rationale:**

This targeted imputation preserves the demographic correlations between race (RACEV2X) and region (REGION42), making the filled data more representative.

**Result:**

It updates the REGION42 column to have all valid values, and placeholder values replaced with a common regional code for each race group.

**IMPUTATION OF AGEDIAG BASED ON CURRENT\_AGE**

**A computer code with text

Description automatically generated with medium confidence**

**Objective:** Fill negative values in the column AGEDIAG by the mode of AGEDIAG within each Current\_Age age group.

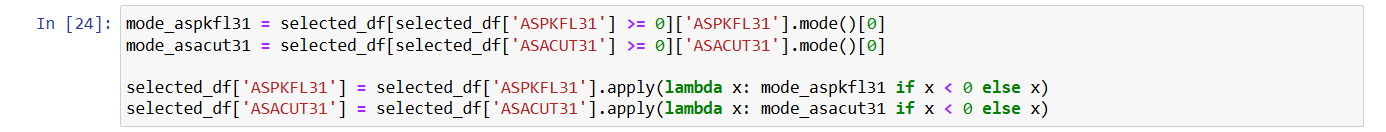
**Approach:**

* For every age group Current\_Age, the mode of AGEDIAG is computed.
* A function replace\_with\_mode is defined that replaces negative values in AGEDIAG with the appropriate mode given Current\_Age. If the age group is not found in the dictionary mode\_agediag\_by\_age, the original value is maintained.
* This will preserve the relationship between AGEDIAG and Current\_Age such that imputed values are representative typical cases for each age group.

**Output:**

The AGEDIAG column is cleaned; negative values are replaced with the mode applicable for the participant's age.

**GLOBAL MODE IMPUTATION FOR ASPKFL31 AND ASACUT31**

****

**Objective:** Replace negative values in the ASPKFL31 and ASACUT31 columns with their respective global modes.

**Approach:** Mode for each column is taken from valid entries - that is, entries that are not negative. apply() function makes sure that the negative entries in these two columns get replaced by the global mode.

**Rationale:** The imputation of missing data using a global mode in which no strong subgroup patterns are to be expected, more appropriately, this approach ensures consistency for the dataset without introducing new biases.

**Result:** Missing or invalid values in ASPKFL31 and ASACUT31 have been replaced with their respective global modes to make the data complete.

**IMPUTATION FOR ASATAK31 ON BASIS OF CONDITIONAL ASDALY31**

**A screenshot of a computer code

Description automatically generated**

**Objective:** Use conditional imputation to fill in the negative values of ASATAK31 based on the values in the ASDALY31 column.

**Approach:**

* The mode of ASATAK31 is estimated separately for records corresponding to ASDALY31=1 ("yes") and ASDALY31=2 ("no").
* Use the replace\_asatak() function to replace negative values found in ASATAK31 with its mode value based on ASDALY31 value.

**Rationale:** Conditional imputation maintains logical consistency, ensuring the imputed values of ASATAK31 align with participants' ASDALY31 status.

**Output:** The ASATAK31 column has been cleaned, with negative values replaced with contextually appropriate modes.

**ASDALY31 CONDITIONAL IMPUTATION USING ASATAK31**

**Objective:** To impute the negative values of ASDALY31 using conditional logic based on the ASATAK31 column.

**Approach:**

* We determine the mode of ASDALY31 for which ASATAK31 is 1 indicating yes and 2 indicating no.
* These modes are applied to the negative values in ASDALY31 using the function replace\_asdaly().

**Rationale:** This will align ASDALY31 with the respective values of ASATAK31, hence logical consistency in relations of data.

**Result:** The ASDALY31 column is populated with negative values replaced by contextually relevant modes.

**THE RESULTED DATAFRAME:**

**A screenshot of a computer

Description automatically generated**

**IDENTIFYING TARGET COLUMNS**

**A white rectangular object with a black border

Description automatically generated**

**Objective:** To select choice columns on which to conduct Outlier analysis.

**Approach:** The locate list is generated using the range() function to include columns whose indices are 2, 3, and 4. The additional indices 8, 20, and 21 are added manually.

* This returns a list of column indices that is searched for outliers.

**COMPUTING IQR AND QUARTILES**

**Objective:** The first quartile - Q1, the third quartile - Q3 and the interquartile range, IQR, for the columns below.

A screenshot of a computer

Description automatically generated

**Methodology:**

* Q1 and Q3 will be computed for each column in the locate list using the appropriate quantile() function.
* The IQR will be calculated as Q3 minus Q1.

**Explanation:** IQR gives us the range in which most of the data points lie. Anything less than Q1 - 1.5 \* IQR or greater than Q3 + 1.5 \* IQR are outliers.

**Output:** Q1, Q3 and IQR calculated and printed out.

**OUTLIER DETECTION**

**Objective:** Find which rows have outlier values in the specified columns**.**

**A screenshot of a computer

Description automatically generatedA group of small colored squares

Description automatically generated with medium confidence**

**Approach:** It first uses np.where() to select the row indices where values are either below Q1 - 1.5 \* IQR or above Q3 + 1.5 \* IQR. Then np.unique() returns the unique row indices along with the count of each. These counts are filtered to return the indices of those rows containing outliers in at least three of the input columns.

**Rationale:** Multiple columns outlier detection would mean that only significant outliers remain - meaning rows with deviations in more than one column. A single-column outlier might just be data variability.

**Output:** Prints various unique indices where no outliers happened and their counts.

The rows that correspond to the condition of having outliers in at least three columns.

**REMOVING OUTLIER ROWS**

**Objective:** Eliminate the outlier rows from the DataFrame.

A screenshot of a computer code

Description automatically generated

**Approach:**

* The indices of the outlier-conditioned rows are collected within the counts>=3 in rows\_to\_drop.
* That then filters selected\_df to get only those rows not in selected\_df.index isin(rows\_to\_drop).
* Removing rows with large numbers of outliers can help keep consistency in a set of data, reducing skew on any analysis or modeling.

**Output:** The shape of the now changed DataFrame is output as confirmation how many rows and columns after removing outliers.

**Result:** This dataset has significant rows removed that contained outliers, hence it is a cleaner and more representative data structure for further analysis.

**Checking redundancy of attributes by using correlations tests and remove redundant attributes**

**A screenshot of a computer program

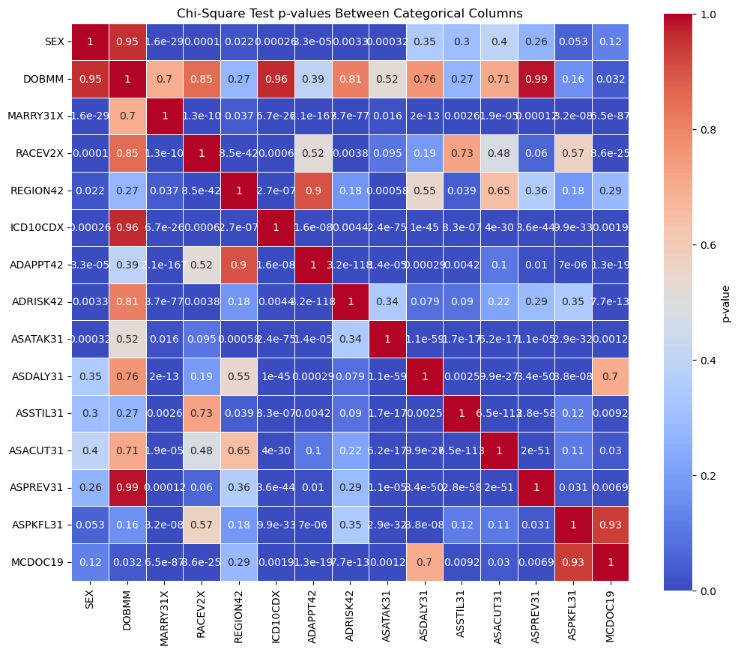
Description automatically generated**

**A close up of a number

Description automatically generated**

A screenshot of a computer

Description automatically generated



We check categorical attribute redundancy by conducting a Chi-Square test of independence between each pair of categorical variables. The columns selected for the analysis are: 'SEX', 'DOBMM', 'MARRY31X', 'RACEV2X', 'REGION42', 'ICD10CDX', 'ADAPPT42', 'ADRISK42', 'ASATAK31', 'ASDALY31', 'ASSTIL31', 'ASACUT31', 'ASPREV31', 'ASPKFL31', and 'MCDOC19'. A Chi-Square test will be performed for every combination of two variables using the chi2\_contingency function returning a p-value. The p-value determines the relationship between the two variables. If the p-value is greater than 0.05, the variables are independent, and therefore there is no redundancy between the two variables. If it is less than 0.05, then the variables are dependent, and there is redundancy. For example, using a p-value of 0.85, this means the two variables tested are independent, and therefore they must not be taken as redundant variables. After that, a heatmap is created to show the p-values that make it easy to identify redundant variables through visualization.

A computer screen shot of a computer code

Description automatically generated

In particular, we are testing the redundancy of categorical variables in a dataset by Chi-Square tests of independence for variable pairs. We hope to identify non-redundant variable pairs whose p-value from the Chi-Square test is greater than 0.85. A value above this number indicates that there is no strong association between the two tested variables in question and, therefore, they are independent with no redundancy.

The code first applies the Chi-Square test results to the matrix of p-values, and then a threshold is applied to filter out only the pairs that have low association, thereby not being redundant, as chosen by a threshold of 0.85. The resulting pairs are stored in independent\_columns, and this is printed. This will contain the pairs of categorical variables that are non-redundant, i.e., those with a low association as filtered out by the threshold value of the p-value. The pairs are:

* ('SEX', 'DOBMM')
* ('DOBMM', 'RACEV2X')
* ('DOBMM', 'ICD10CDX')
* ('DOBMM', 'ASPREV31')
* ('REGION42', 'ADAPPT42')
* ('ASPKFL31', 'MCDOC19')



It would remove the DOBMM column from the DataFrame called selected\_df . The reason this is carried out is based on our previous analysis where a few pairs of categorical variables showed low association, in other words, non-redundant. Specifically, the pair ('SEX', 'DOBMM') , ('DOBMM', 'RACEV2X'), ('DOBMM', 'ICD10CDX'), ('DOBMM', 'ASPREV31') had low association as we got a p-value greater than 0.85.  
The following variables using the Chi-Square test are independent and hence have a low association DOBMM.  
Based on the analysis, this column is dropped from the DataFrame since it may not be needed for further analysis, either because of redundancy or because it lacks predictive power when used in conjunction with other variables.



This will be the last step in our data preprocessing workflow, where we will assign the cleaned and preprocessed DataFrame, selected\_df, to a new variable called normalized\_asthma\_influenza\_selected, which now will represent the final version after all necessary adjustments: removal of redundant attributes, drop of the DOBMM column due to low correlation with other categorical variables. First, after the preparation of the dataset for further analysis, it is exported to a CSV file entitled "asthma\_influenza\_data\_processed.csv". The data is normalized and cleaned and kept in structured format and saved into the file with an index column off via an index=False argument. In this way, the saved file is ready for further analysis, modeling, or even sharing with stakeholders; thus, the stage of data pre-processing is complete.

**CONTRIBUTION OF GROUP MEMBERS:**

KAUSHIKAN DINAKARAN: Data Cleaning and helping in Data Processing.

ARJUN PALANISWAMY: Data Processing.

ROHITH SINGH THAKUR: Data Understanding and Final Report formatting.

USHA AAVULA: Verification of duplicate removal and Report.

**REFERENCE:**

1. Veerapandian, R., Snyder, J. D., & Samarasinghe, A. E. (2018). Influenza in asthmatics: For better or for worse? Frontiers in Immunology, 9. <https://doi.org/10.3389/fimmu.2018.01843>.
2. Li, K. H., Leong, P.-Y., Tseng, C.-F., Wang, Y. H., & Wei, J. C.-C. (2021). Influenza vaccination is associated with lower incidental asthma risk in patients with atopic dermatitis: A nationwide cohort study. Frontiers in Immunology, 12. <https://doi.org/10.3389/fimmu.2021.729501>.
3. Jha, A., Dunning, J., Tunstall, T., Thwaites, R. S., Hoang, L. T., Kon, O. M., Zambon, M. C., Hansel, T. T., & Openshaw, P. J. (2019). Patterns of systemic and local inflammation in patients with asthma hospitalised with influenza. European Respiratory Journal, 54(4), 1900949. <https://doi.org/10.1183/13993003.00949-2019>.
4. Wang, Z., Li, Y., Gao, Y., Fu, Y., Lin, J., Lei, X., Zheng, J., & Jiang, M. (2023). Global, regional, and national burden of asthma and its attributable risk factors from 1990 to 2019: A systematic analysis for the global burden of disease study 2019. Respiratory Research, 24(1). <https://doi.org/10.1186/s12931-023-02475-6>.