Group-6-ETL-Project

Project 2: ETL Challenge

This project was completed by Group 6 which consists of Sam, Serena, Vignesh and Nicklas.

The Jupyter notebook will create a database with information about volcanic eruptions since the 1800s.

Process:

Four csv files from two different sources have been used to create the final database in PostgreSQL.

The following steps were taken to load the data and take it through the clean-up process:

1. The data was then extracted from the websites in the form of csv files.

2.Each csv file was then added to a Jupyter notebook to start the cleaning process with pandas.

3.Once the cleaning process was finished a schema was created using quickdatabasediagrams.com.

4.The schema was then exported as an .sql file.

5.Once the database was finished the engineered tables were loaded into the final Postgres database.

EXTRACT

The following data sources were used for the process:

1. eruptions.csv - source https://data.world/jessymorgan25/volcanoeruptions

2. volcanos.csv - source: https://data.world/jessymorgan25/volcanoeruptions

3. events.csv - source: https://data.world/jessymorgan25/volcanoeruptions

4. damage.csv - source: https://data.cerdi.uca.fr/erup-vol/

These sources can be found in the resources folder of the repository.

TRANSFORM

1. eruptions.csv:

    - Firstly, the null values in the year columns were dropped.

- The null values then remaining in the day and month columns where then replaced with 1 to allow for datetime conversion.

- Data of volcanoes from before the year 1800 were dropped.

- Using lambda and astype functions were used to convert the date columns from floats to string and then concatenated to form a valid datetime format which was appended to new columns in the data frame.

- The newly created date columns were then transformed to datetime using pandas to_datetime function.

- Unnecessary columns were then dropped to finalise the data frame.

2. volcano.csv:

- Due to broad data-range, we have decided to filter for data of last eruption happened on and after 1800, however the columns are filled with some unknowns and not in integer format.

- To achieve this, replacing “Unknown” with value zero, then change the column into integer. using loc to filter year that is greater than 1800

- setting index and remove irrelevant columns.

3. events.csv:

- All the columns were loaded onto the data frame.

- Drop duplicates to make sure the primary key has unique values and overall data is not repeated.

- Reset index and drop null values across the data frame.

- Check for yearly occurrence of volcanic eruptions using value count’s function.

- Filtering by years
  Check for the values of the data frame by using describe especially focusing on the oldest of years so we can filter the years after 1800. This is achieved by using the. loc function calling for years greater than 1800.

- Check with describe function again if it worked.

- Filled missing date values with 1 so the dates can have start of year or month if the day or month of an event is missing by replacing the null value with 1.

- Convert the dates to string and concatenate and convert back to datetime format by using pd.to_datetime function.

- Drop unnecessary column and check for the finally cleaned data frame.

4. damages.csv:

- The data columns were checked for their type and number of rows. The values were displayed using the value_counts function for several columns of interest. It was clear that the data need a lot of cleaning and reformatting, mainly because of the mixing of numeric data with comments.

- A defined function was created to cover the most common clean-up requirements including use of the split function and replace function. Any conversion of a column to numeric was done subsequently.

- Unnecessary columns were then dropped to finalise the data frame. Several columns were renamed to keep them consistent with other columns originating from the other data sources.

LOAD

Once the transformation has been completed the tables were created from the schema created in quickdatabasediagrams.com.

A database was created within PostgreSQL to which the schema was uploaded.

Once the schema was uploaded to the database the final importing of data can occur. Take note that the data should only be uploaded once to avoid any errors with the primary keys.