Team: Kat, Brant, Collin

ETL-Project

10.16.2019

Purpose: transform call data to allow analysis by unique call, date, user

Extract: We began our project by collecting the call records from Terradatum's phone system which was exported as multiple CSV files. We also used a csv file of employee data.

Transform: There were several variables that we wanted to redefine in our call record csv files. We used Pandas in a Jupyter Notebook to perform all data transformations.

We took the caller ID field (clid), removed the pre id characters, quotes, greater than and less than, and separated the name and number into 2 separate fields (clidname, clidnum). The data "DID: OLATHE KS" <+19135382895> was transformed into OLATHE and +19135382895.

In order to create reports by date that were more detailed than could be created from a single field containing the data and time. We transformed the calldate field and split the date and time into 2 separate fields. Then we split the date into Year, Month and Day. In addition, we added a field to reflect the quarter (qtr) 1, 2, 3, 4. The data 2019-06-29 14:35:39 was transformed into 2019-06-29, 14:35:39, 2019, 06, 29, 2.

The phone system recorded call duration in seconds. We wanted to report on the call duration in minutes. We added a field, converting the seconds into minutes. A call in the duration field of 100 seconds was displayed in the durationmin field as 1.67.

We filtered the data to only reflect a call disposition of ANSWERED. We did this because every uniqueid had duplicate lines with a disposition of "NO ANSWER" (adding an extension to the end of the uniqueid, that we named 'subid'). We did not find the information obtained within "NO ANSWER" of value. We split the uniqueid into 2 fields - uniquecall and subid. This enabled us to filter the data further to ensure we were looking at unique call records.

There were additional fields that we did not intend to use in our reports. Those tables were dropped. They included accountcode, amaflags, userfield, dcontext, and clid.

The resulting DataFrame was saved as a csv file.

The employee file was much simpler. We split the Name field into FName LName and dropped the Name field.

The resulting DataFrame was saved as a csv file.

Load: We imported the resulting User and UsageData csv files into a Postgres databases allowing for reports to be drawn on the two databases to gather user information associated with the calls made. In addition, the transformed data from the final Dataframe was saved as a JSON file.