Designing and Implementing a Data Warehouse with US Immigration Data – Data Engineering Capstone

Project Write-Up

Udacity Data Engineering Nanodegree Capstone Project

Immigration Data Warehouse

Aron Tharp

January 2021

https://github.com/arontha/Udacity-Data-Engineering-Capstone

Project Overview

Purpose

The purpose of our data warehouse is to support analysis on immigration trends to the US, be it for business, pleasure, or travel. By incorporating diverse datasets, we can answer a wide range of questions such as who is immigrating where and when, is climate change driving immigration, does the racial diversity of the US population affect immigration levels, and do immigrants seek a US destination that is like their home climate? Our data warehouse will allow for analytics tables to be generated by any user with SQL knowledge to answer these questions and many more that they may have.

Since we are leveraging AWS, our data warehouse can be accessible to others regardless of their physical location if we provide them with the Redshift cluster details. AWS allows for flexibility in the uptime of the database with cost being in the form of hourly rates and hosting on AWS servers.

Final Data Model

Our data model employs the star schema for our OLAP data warehouse. This is preferred to other models such as 3NF to ensure that SQL queries are and logic for joins are not overly complicated. 3NF generally requires more tables and are better when data integrity is a priority, but the star schema's simplicity is better for us since our purpose is analytics.

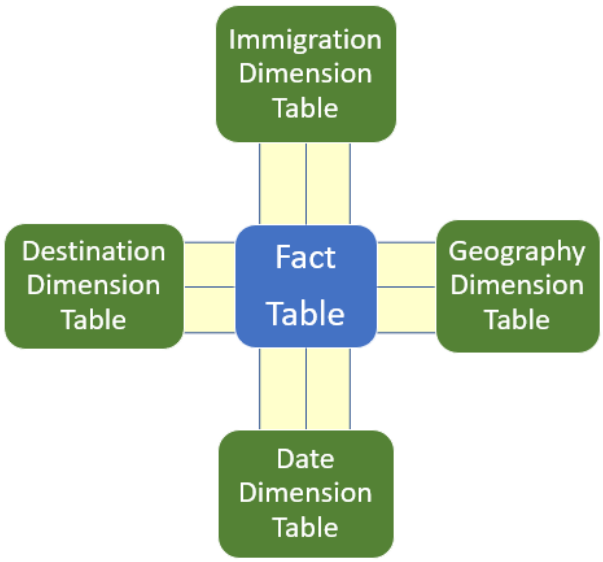

Figure 1: Star schema chosen as final data model

The tables in the star schema represent the main categories that may be relevant to an immigration observer and are not 1-to-1 with our initial datasets. Clean, manipulation, and restructuring our data is an important part of this project that are required to transform our input data into these tables.

Tools and Technologies

Python and Pandas and SQL

Python is an easy choice for the data analysis, cleaning and manipulation needed on our data. Its expansive set of libraries, such as Pandas, is very conducive to this type of work. The power, speed, and flexibility of Pandas makes it an obvious choice for the heavy lifting of the data up until the point at which it is loaded into the data warehouse. SQL is the standard for relational databases and is such the tool of choice for our data warehouse. SQL was chosen over NoSQL to allow for joins by analysts looking to run reports from our data warehouse.

Jupyter

Jupyter notebooks provide both a user-friendly environment for the data engineer to develop the data warehouse and an easy-to-understand format for others to review the work of the data engineer.

AWS and Redshift

AWS is a mainstream choice for big data and hosting in the cloud. AWS has many services including Redshift which supports data warehouses. Redshift is a scalable resource with a reasonably low cost that is easily managed by the data engineer. It allows for a distributed userbase and is compatible with other AWS services. Its speed for the cost and flexibility of the number and speed of nodes is ideal for our purposes.

Spark and Airflow

If the scope of this data warehouse were to change, it would benefit from the incorporation of PySpark and Apache Airflow. The former allows Pandas-like operations to be performed across multiple nodes to avoid memory issues and greatly increase speed on huge datasets. The latter allows for scheduling of runs if the warehouse were to need regular updates and has a user-friendly dashboard for the data engineer to oversee the pipeline.

Datasets

Immigration (i94)

The largest dataset for this project is a month of i94 data from the US National Tourism and Trade Office exceeding 3 million rows and 29 columns, making the total number of cells around 90 million. The dataset has many fields including visa type, visa reason, destination state, origin country, age, mode of transport, etc. These different fields will provide many ways for a user to aggregate the data to understand the trends.

Temperature (Climate Change)

The second largest dataset for this project comes from Kaggle and provides information related to temperature and climate change across centuries for many cities in the world. This dataset contains more rows than the i94 dataset, almost 9 million, but only contains 7 columns making the total number of cells around 60 million.

US Cities

This dataset from OpenSoft gives demographics for the US cities including median age, population by race, and foreign-born population. This will support analysis by showing whether trends in immigration exist due to the racial makeup of the destination. For example, one could be interested in seeing whether Mexican tourists visit primarily cities with large Hispanic populations.

International Airports

This public dataset, available under the Open Data Commons (ODC) Public Domain Dedication and License (PDDL), is an extensive list of international airports of all sizes. Since there are cities in different states and countries with the same name, the data will be limited to large airports to allow for accurate joins with other datasets.

Project Steps (Summary of Jupyter Notebooks)

Scoping and Data Gathering (Jupyter Notebook Step 1…)

The first notebook imports each dataset and reviews the columns, length and values of each dataset. The different formats of the datasets are apparent, such as the cities dataset having a semi-colon as a delimiter. Another anomaly of the cities dataset is its format. For each city, there are multiple rows that correspond to each racial population. This is not ideal for our purposes; we will need to reformat this data in step 2 so that each racial population is a separate column, and each city is only one row. We review a small sample of the i94 dataset, only 1000 rows, just to look at it. However, any cleaning for the immigration dataset will only be important on the full dataset so it will need to wait until the 4th step. An important observation at this stage is that the i94 dataset only provides the state destination and not the city and the origin is only provided in the form of a country. This will dictate our data model and the keys of our tables, as well as require some manipulation of the corresponding data. Definitions for each field in each of the datasets can be found in the Jupyter Notebook titled Step 1. Some very minor alterations to the data take place in this step and the changes are saved in the folder titled data_first_cleaning.

Explore, Assess and Clean Data (Jupyter Notebook Step 2…)

The bulk of the cleaning takes place in step 2, with the main exception being the i94 dataset. We exclude the i94 dataset at this step due to its size. It is not efficient for us to load it in, clean it, and then save it to be loaded in again at a later step.

Starting with the airports dataset, some exploration shows us that we have more information than is relevant for our purposes. The types of airports are comprehensive and not indicative of the ways in which people would generally immigrate. For example, the dataset contains heliports, balloon ports and

seaplane bases. These will be removed from our dataset and we will limit ourselves only to airports. The next step in cleaning the airports dataset is to add additional information to the continents column. We correct the North American countries which were populated with nan instead of NA. There are 3 columns in this dataset which seem to have a lot of overlap but are not identical: ident, gps_code and local_code. While we will not delete any of these columns, our final data model will only be populated with local_code.

Next, we clean the cities dataset. As discovered in step 1, our biggest task is to restructure this dataset so that each city is only one row using the groupby function. After transformation, a simple check of the population figures shows that this data is either old or incomplete. While a year is not provided for this population, a simple Google search confirms that the values of total population are lower than present. We assume that the cities are still proportionally populated and adjust the data accordingly into percentages to ensure the proportions are maintained. The total population values are transformed into a percentage of the total for the entire dataset and the segmented population columns are transformed into percentages of the total population or that row. Lastly, as this dataset is limited to US cities, it will only be relevant for the destination which is in the form of states instead of cities. For this reason, we will condense our data into one row per state and eliminate the cities field.

State	Median Age	Male Population	Female Population	Total Population	Number of Veterans	Foreign-born	Average Household Size	American Indian and Alaska Native	Asian	Black or African-American	Hispanic or Latino	White
Alabama	36.23	47.4	52.6	0.9	6.8	5	2.43	0.8	2.7	49.6	3.7	47.5
Alaska	32.2	51.2	48.8	0.3	9.2	11.1	2.77	12.2	12.3	7.7	9.1	71.2

Figure 2: States dataset sample (previously cities dataset)

Lastly, we approach the temperatures dataset. Since we want to do not want to overburden our data warehouse with a large amount of potentially useless data, some assumptions about the users use cases are necessary. We will assume that immigration is dictates by recent climate changes, therefore we do not need to maintain centuries worth of climate data in our data warehouse. We choose to condense our data down to the current year and maintain overall trends for the past 10 and 20 years, respectively. Our temperature dataset only has cities and countries but not states. Since our destinations are according to state, we will want to be able to see the climate for a given state. By joining the cities dataset to this one, we can provide that extra column and allow for a better use of the temperature data in our data warehouse.

Define Data Model (Jupyter Notebook Step 3…)

Now that we have a better understanding of our datasets, we can start to best imagine what an end user of our data warehouse would need. As discussed earlier in this writeup, the star schema is ideal for our purposes. It is easier to understand and manipulate with SQL by the end user looking to perform analytics on our OLAP data warehouse. We also need to imagine the data warehouse in the logical structure that an analyst would want, which is not the same as the input datasets. Since we have a lot of data on the individuals, we will maintain them in an immigration table. The destination has extra information related to population that we can keep in a destination table, but the origin country does not have the corresponding information therefore an origin table is not necessary. Our temperature and airport datasets are both at an international city level, therefore we will combine them into a geography table which will provide some geographical information to the analyst looking to better understand the

origin or destination. Lastly, to allow for growth in our data warehouse, we will keep a date table which includes month and year fields even though the current data is only for June 2016.

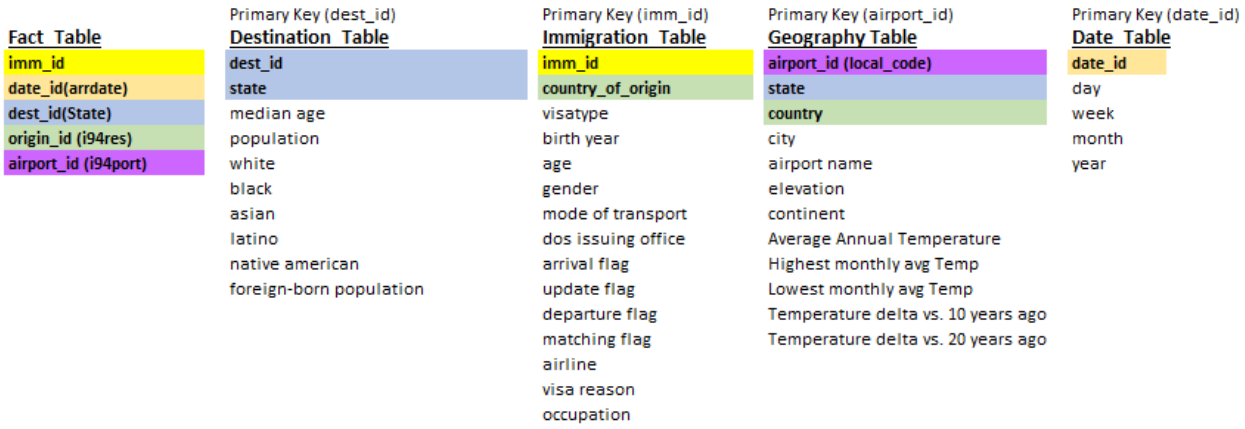

Figure 3: Tables in Data Model

Run ETL Pipeline (Jupyter Notebook Step 4…)

Running our ETL pipeline consists of loading our datasets into the workspace, restructuring them into the format needed for our tables, and then loading them into Redshift. Since the immigration dataset could not be cleaned earlier due to its size, it will also be cleaned in this step. The immigration data contains many fields, several of which are encoded, and has extra quotes which need cleaned. The visa reason, mode of transport, and country of origin fields all need decoded so that the end user can understand these data. The fields all need renamed into something more understandable and quite a few of the columns are removed due to the lack of interesting information.

To load our tables into Redshift, we must join the datasets accordingly. The airports and temperature datasets are joined to create the geography table. The information for the date table will be extracted from the arrival date in the i94 data. The states dataset is already in a format for the destination table. The facts and immigration tables will come directly from the immigration dataset. A data dictionary for these tables is also specified (see Appendix).

The Redshift cluster must be launched before running this notebook and its information stored in the dwh.cfg file, along with the key and secret information of the AWS account performing the ETL operation. Since insert one line at a time is an inefficient process in Redshift, but the insert operation has an upper limit on the size, the tables are updated 10000 rows at a time or less. For cleanliness, the SQL queries are saved in separate files which are found in the sql_queries folder of the workspace.

Simple data quality checks are then performed on the data by ensuring that all tables have rows in them. Additionally, a sample query is run to show the top destinations.

Update Requirements of Data

Infrequent

Overtime the needed to update our data is clear. Our temperature data becomes obsolete due to climate change, but our horizon of temperature data is annual therefore an update schedule of an annual basis is reasonable. The population information from our cities information also could be

updated on an annual basis, or even less since censuses are only performed once per decade. The airports dataset is arguably the most static and would need to be updated with the least frequency – only minor changes would occur over a course of years.

Frequent

One month of i94 data has millions of rows. The behavior of this dataset is likely to change with seasonality and other trends and differences would be visible from one month to the next. It is likely that business travel, vacationing, and studying would also see evolution in its trends even across months of the same year. For this reason, it is recommended that a monthly update be performed with addition i94 data if feasible to support analysis on relevant data.

Exploring Different Scenarios

Data Increased 100x

Any project must take into consideration how it would need to evolve if the data requirements changed. In our database, we are only using i94 immigration data for one month. If continue loading data into this database for a decade, the database will need to be redesigned.

The most immediate issue would be the limitations of the Pandas library. Since it does not support parallel computing, we already have the issue where our code may take several minutes to run with our current amount of data. If the data increased one hundred times, we would run out of memory. The PySpark library can serve as a replacement for Pandas to address this issue. PySpark's support for running operations on multiple nodes allows for huge datasets to be processed very quickly. The data engineer must keep in mind that programming of PySpark and Pandas are not identical. PySpark employs lazy evaluation and has some quirks in debugging and error tracking that are quite different than Pandas.

Another consideration for the size of data is in the tables themselves. In our current data model, we have already accounted for the scenario of the i94 immigration data being loaded for a much longer time period. The date table has fields for month and year which seem to be redundant given our current data since it is only populated with June 2016. However, this was an intentional design to allow for growth.

Lastly, the technology choice of Redshift allows for very large databases. The choice of node allows the user to employ whatever need is appropriate for their use. Through trial and error, the data engineer can determine what is appropriate for the size of the data. It is anticipated that faster and more nodes are required if the data grows by 100x. S3 is also a very attractive storage site for huge data and would also need to be evaluated.

Lastly, such large data sizes can drive a change from SQL to NoSQL. If this change were necessary then the data model itself would need to be reconsidered since NoSQL does not permit joins and the current data model anticipates users joining the various tables in the data model.

Dashboard of Data Updated Daily @7am

Regular updates to this database would be best facilitated by a change in technology. Apache Airflow would allow for the scheduling and modulation of the database load. Airflow supports use with Redshift and makes scheduling of tasks very convenient – an advantage since we are already using Redshift. Airflow can also be programmed in Python, and the ETL code would need to be updated accordingly. Airflow provides a very user-friendly dashboard for the data engineer to quickly troubleshoot any issues which may arise. Since Airflow is only a workflow management platform, the users of the database would not need to know about it.

Number of Database Users Exceeds 100

Our choice of technology already supports this. Redshift delivers a very cost-efficient solution that is accessible to many users. Since Redshift is part of AWS, if any of our diverse userbase wished to use other AWS services in conjunction with our data warehouse then it would be very straightforward.

Appendix

Data Dictionary for the respective tables as outlined in step 3

facts_table
- imm_id int - unique identifier for each instance of immigration
- date_id date - based on arrival date - from i94 dataset
- dest_id varchar - 2 letter US State abbreviation - from i94 dataset
- origin_id varchar - country of origin
- airport_id varchar - 3 character airport code (limited to large airports) - from i94 dataset

destination_table
- dest_id varchar - 2 letter US State abbreviation - from cities dataset
- state varchar - 2 letter US State abbreviation - from cities dataset
- median_age float - median age of the state's population
- population float - population as a percent of the US total
- white float - racial population as a percent of the state's total population
- black float - racial population as a percent of the state's total population
- asian float - racial population as a percent of the state's total population
- latino float - racial population as a percent of the state's total population
- native_american float - racial population as a percent of the state's total population
- foreign_born float - foreign-born population as a percent of the state's total population

geography_table
- airport_id varchar - 3 character airport code (limited to large airports) - from airports dataset
- state varchar - 2 letter US State abbreviation - from cities dataset
- country varchar - country name - from airports dataset
- city varchar - city name - from airports dataset
- airport_name varchar - name of airport
- elevation int - elevation in feet of airport - from airports dataset
- continent varchar - name of continent - from airports dataset
- avg_yrly_temp float - average temperature of 2013
- mnthly_high_temp float - highest monthly temperature of 2013
- mnthly_low_temp float - lowest monthly temperature of 2013
- temp_delta_10_yrs float - increase/decrease in yearly average temperature from 2003 to 2013
- temp_delta_20_yrs float - increase/decrease in yearly average temperature from 1993 to 2013

date_table
- date_id date - based on arrival date - from i94 dataset
- day int - day of arrival date
- week int - week of arrival date
- month int - month of arrival date
- year int - year of arrival date

immigration_table
- imm_id int - unique identifier for each instance of immigration
- country_of_origin varchar - same as origin_id in facts_table
- visatype varchar - class of stay
- birth_year int - birth year of immigrant
- age int - age of immigrant
- gender varchar - gender of immigrant
- mode_transport varchar - air, sea or land
- dos_issuing_office varchar - department of state office issuing visa

- arrival_flag varchar - flag for arrival to US
- update_flag varchar - flag for update of status in US
- departure_flag varchar - flag for departure from US
- matching_flag varchar - flag for matching of arrival and departure
- airline varchar - airline flown to arrive in US
- visa_reason varchar - business, pleasure or student
- occupation varchar - occupation to be performed in US