**Capstone Project: Reporting US demographics integrated with Immigration data (City-Wise)**

**Introduction**:

I have chosen the US immigration data (~ 3mn records) for my project which has the the information of the immigrants such as the mode of arrival, gender and their birth year, etc for each month and year.

Along with that, I chose the US cities demographics data which has the information such as the male and female population and median age for each city within State.

**What's the goal? What queries will you want to run?**

My use case is to create a data model to ingest the City wise demographics data and create a data pipeline which will report the US demographics data along with the Immigration data. The data will ultimately help in reporting-

1. The change in the distribution of the population (City-wise) considering the Immigrants (assuming US cities demographics data to be static).

Reporting crucial data like the busiest mode of arrival of the immigrants & the busiest state to receive the migrants flow.

Queries for above analysis and reporting could be made on demand.

**How would Spark or Airflow be incorporated?**

Spark and Airflow would be crucial to my use case as the data of immigration is huge (~500MB for one month), so I have utilized Spark’s power for assessment and data cleaning purpose. In future, even if the size of the data increase, with Spark optimization techniques like partitioning the data and increasing the parallelism, we can retain/increase the performance.

Airflow would be handy and useful for scheduled run of this pipeline with a clear picture of the data integrity checks to the end users as well.

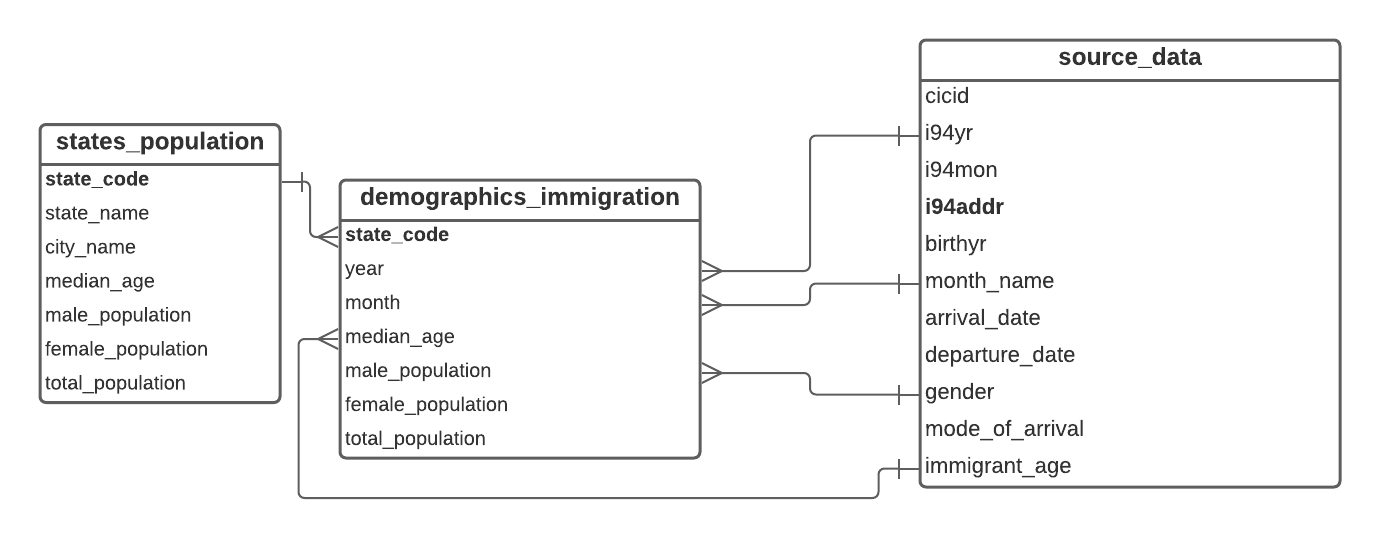
It would be easier to add any new check, if required, in the existing pipeline with only minor changes.

**Data Model**:

Using the two data sets, US cities demographics and the US immigration data, I have created -

one fact table - states\_population which will have the static information from the US cities demographics data, and

one dimension table which is being loaded from the cleaned immigration data after transformation through the etl\_pipeline and it will report the state wise demographics for the immigration data.



**Why did you choose the model you chose?**

The main reason behind creating such a data model is the nature of the data that I have selected and the way it will change with time. The states population data is a static data while the immigration data has lots of information but I have decided to use only few of them as per my use case which would be in relation with the states demographic data as well.

The only matching column between the states demographics data and the immigration data is ‘state code’ (referred as i94addr in immigration data), due to which the kind of analysis and reporting that could have been made using both these was restricted.

I chose to create a data model that will use the matching column (state code) to fetch the integrated demographics data from both the data - immigration and cities demographics data where above fits well.

**Clearly state the rationale for the choice of tools and technologies for the project.**

**Tools & Technologies used**:

Spark (on EMR) - The capability of spark has been used to assess the immigration data since the data was huge (~500MB for just one month) and the same has been utilized to clean the data using the Pyspark wrangling methods.

Amazon Redshift - Has been used for creating the Postgres tables and load data into them.

**Steps of the process-**

Data Gathering & Wrangling:

* Immigration data and the US cities data were gathered (both available in Project resources).
* Immigration data (parquet files) is being read from S3 into pyspark dataframe and assessment is done for identifying quality issues like missing or duplicates.
* Immigration data is cleaned and written into csv file on S3 location.

Data modelling with Postgres:

* Redshift cluster is created and the configuration is provided in a config file to the workspace.
* SQL queries and create tables statements written for creating tables as per the data model.
* Tables created and verified.

Creating and running data pipeline:

* Create a data pipeline to load the cities demographics data into states\_population table and load the data into immigration\_demographics table after transforming as per the requirement based on the ‘state\_code’.
* Data verified and sample queries ran.

**Propose how often the data should be updated and why?**

* Immigration data - This is a frequently changing table on daily basis, but the data has information on the monthly basis for each year, so I am assuming that the data pipeline can be ran on the start of every month to get updates from this.
* Cities demographics data - Although, this can be also be a regularly changing data, but I am assuming that such data are based on census which happens yearly or every 10 years, so this can be ran on demand whenever the new data is available.

**Include a description of how you would approach the problem differently under the following scenarios:**

* **If the data was increased by 100x.**
* **If the pipelines were run on a daily basis by 7am.**
* **If the database needed to be accessed by 100+ people.**

1. **If the data was increased by 100x** - There are multiple options that should be used together if the data was increased by 100x-

* Partition the data to increase the parallelism while processing the data with spark jobs, in my case, this can be achieved by partitioning the immigration\_demographics data on (year, month) columns together.
* Increase the EMR cluster capacity with more number of core and task nodes to enable more executors to be working at once.

1. **If the pipelines were run on a daily basis by 7am** - Since the scripts are already properly segregated, we can simply create an airflow DAG in a manner that will be scheduled to run at 7am daily in a manner to perform all the tasks in order.

Steps as follows:

* Create DAG with all the processes in order
* Place your code build on an S3 location
* Create a bootstrap script that will sync the code from S3 to EMR everyday before 7am on start of EMR
* Point your spark-submit jobs to the above EMR location to run in the Airflow DAG script.

1. **If the database needed to be accessed by 100+ people** -

This can be done on the infrastructure side to manage such situation-

different IAM roles can be created and permissions can be given to such roles on the basis of the use cases for different users. A user can then take access to the suitable role and perform only such tasks that the role has access to. For example- Role A can just allow to view the table but Role B can allow to update the schema or drop the table along with view access.