Introduction:

As a part of the Udacity DEND project, we are considering the **I94 Immigration Data** which originates from the trade.gov website. We are looking to help analytics team derive insights, by properly modelling the datasets

In order to provide the ability for analytics teams to dive deep, we have additional datasets
World Temperature Data from Kaggle
US City Demographic Data from OpenSoft
Airport Code from Datahub

I94 Data has fields such as DataOfArrival, Unique Field cicid, Visatype, Arrival Location, Airport Details, Place of Origin, Date Of Validity, Visa Type amongst other I94 related information

Temperature Data consist of Average Temperature and it is variance every day from 1820 to 2019 across various cities in various countries

USCity dataset consists of population statistics such as Total Population, Median Age, Gender Classification, Household sizes across various cities in United States

Airport dataset consists of details such as Name of Airport, Location, coordinates and ident values that helps in uniquely identifying the dataset.

Technical Justification :

As a part of building a process for Analytics Users across the 4 datasets it has been to store the data in a data lake in S3, instead of a traditional data warehouses and following are the reasons as to why:
1) Sizes of the datasets are 3 Mn records (I94Data), 680K records (Temperature), 55k records and 3k records (US cities). These are current sizes and there is a chance that they might grow down the line as well and for such a case it makes sense to plan ahead by moving into a Data lake
2) Flexibility is another reason as to why a data lake is preferred to a data warehouse. In this scenario, a platform is being built for analytics teams to derive insights, but that said none of the insights have been identified yet and unless there is a clearly defined set of business objective(s), it is not feasible to build a rigorous DWH model
3) Future proofing for additional needs. There might be requirements from the business down the line as well to gather additional datasets example Airplane details such as FltType, CompanyName, SeatNumber,NumberOfLayover, ClassOfTravel..etc or some other dataset from a location that we might need be aware of and this leads to a possibility of processing log files, images, audio and video files.etc and since we need to make provision for the future, it again makes sense to build a data lake Solution so that we don't restrict to structured or semi-structured data with a data warehouse

In order to build a process for big sized datasets which can be handled through batch based processing instead of real time processing, we leverage Spark (specifically PySpark) to populate the data in the Data Lake (AWS s3 buckets), Spark has preferred over Hadoop another distributed processing framework, due to the in-memory processing of data in Spark which helps in making Spark process around 100* faster in comparison hadoop reads and writes intermediate data into disk (https://www.ibm.com/cloud/blog/hadoop-vs-spark),and AWS S3 is chosen as our data lake infrastructure, due to ease at which we can set up it and scale up and scale down depending on demand, without having any maintenance overhead associated with it.

Steps Performed:

Extract Data:

We extract the I94Data from SAS Datasets into Pyspark Dataframes and likewise for the Airport, Temperature and USCities dataset we load into Data Frames from Flat files.
In addition to that we have the I94ADDR, I94CityAndResCode, I94Port flat files which are originally a part of the I94Data's metadata. These 3 fields contain the actual values behind the coded values stored in the I 94 Data and it makes sense to include them, so that users are able to infer the description behind the actual coded value

Cleaning:

In an approach to Build a modelled solution, effort has been made to clean the data from the disparate datasources. For each of the datasources, the number of null data for each attribute in the dataset has been computed and the one's with a high number of Nulls have been removed. Redundancy across the dataset has also been addressed, example let's say one of datasets has FirstName, LastName and FullName, in such a scenario the first name and last name field have been dropped.Also any duplicated record across the datasets have been identified and dropped

Upload to DataLake:

After the data modelling is complete, Transformations are performed and partitions are identified using data is written into Datalake, in S3 buckets in parquet file format.
Parquet files store data in columnar format making it suitable to read data in columnar databases such as Redshift. In addition to that it makes it suitable to access Metadata and not just the data. Also, the fact the schema changes (Addition of columns) is easy to handle in parquet

Validation:

In our validate that pipeline is working as expected we perform the following operations:

1) Read the data from all the s3 buckets into DataFrame and confirm that counts of dataframe is greater than 0
2) Count the number of records in each DataFrame and compare with the count distinct of number of unique records using the unique field(s) in the same Dataframe
3) Check to see that number of occurrences of Null in the Unique field(s) in the same Dataframe to be equal to zero.

Implementation and Scenarios:

Data available in Data Lake can be loaded into Redshift (AWS columnar database solution, optimized for analytics) and by having the framework of loading the data into data lake and providing the facility for Analytics teams to access the data in the Data Lake, ensures that the scalability conundrum is solved ahead of time. We use Xlarge AWS Emr cluster to process and load the data into S3 buckets and in case of large data growth 50-100*, we have the ability to tune up our clusters to 16Xlarge cluster (if the general cluster type is opted) and likewise the analytics teams leveraging Redshift can scale up to 16XLarge cluster (for RA3 node). That said, we can even address for a faster growth of data with a data lake, especially when the data starts to become more real time in nature (Example tweets on twitter) by using Kinesis Firehouse write directly to a Data Lake in raw format and depending on the nature of the data, leveraging tools like AWS Lambda, AWS EMR for processing and finally leveraging AWS Redshift or Athena for querying on top for reporting purposes

EMR Instance Types:
https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-supported-instance-types.html
Redshift Cluster Types:
https://docs.aws.amazon.com/redshift/latest/mgmt/working-with-clusters.html
Kinesis Firehose:
https://aws.amazon.com/blogs/big-data/build-a-data-lake-using-amazon-kinesis-data-streams-for-amazon-dynamodb-and-apache-hudi/

The pipeline has been built in such a way that it can be run on schedule using Airflow (assuming we want the data available by 7 AM), we can have a DAG, that runs at 5 AM, that has several operators such as Loading the Source Data into Spark Dataframes, Apply Transformations, WriteDataToDataLake in Parquet Format, Perform Validations. The DAG can be set to retry(number of times, can be customized) in case of failure. In addition to that alerts to email and/or slack is also an option that is available using Airflow

Finally once the data is available on Data Lake, it can be copied over to Redshift from S3 based on permissions. We can manage several users(even in 1000s) by creating groups and providing specific roles to each of these groups. Example QA/ BA people might need just read only access on Redshift over the concerned schemas, whereas a Developer or Business User might need a Select/Insert/Update/Delete access

https://docs.aws.amazon.com/redshift/latest/dg/r_Database_objects.html