Project Title

Data Engineering Capstone Project

Project Summary

--describe your project at a high level--

The project follows the follow steps:

- Step 1: Scope the Project and Gather Data
- Step 2: Explore and Assess the Data
- Step 3: Define the Data Model
- Step 4: Run ETL to Model the Data
- Step 5: Complete Project Write Up

In [7]:

```
# Do all imports and installs here
import pandas as pd
import os
from pyspark.sql.functions import monotonically_increasing_id, udf, from_unixtim
e
from pyspark.sql import types as types_
from datetime import datetime, timedelta
```

Step 1: Scope the Project and Gather Data

Scope

Explain what you plan to do in the project in more detail. What data do you use? What is your end solution look like? What tools did you use? etc>

Describe and Gather Data

Describe the data sets you're using. Where did it come from? What type of information is included?

In [2]:

```
# files path
airport_code_csv = 'airport-codes_csv.csv'
us_cities_demographics = 'us-cities-demographics.csv'
immigration_data_sample = 'immigration_data_sample.csv'
```

In [4]:

```
# Read in the data here
airport_code_csv_df = pd.read_csv(airport_code_csv, header=0)
us_cities_demographics_df = pd.read_csv(us_cities_demographics, header=0, sep
=';')
immigration_data_sample_df = pd.read_csv(immigration_data_sample, header=0)
```

In [5]:

```
airport_code_csv_df.head()
```

Out[5]:

	ident	type	name	elevation_ft	continent	iso_country	iso_region	municipality
0	00A	heliport	Total Rf Heliport	11.0	NaN	US	US-PA	Bensalem
1	00AA	small_airport	Aero B Ranch Airport	3435.0	NaN	US	US-KS	Leoti
2	00AK	small_airport	Lowell Field	450.0	NaN	US	US-AK	Anchor Point
3	00AL	small_airport	Epps Airpark	820.0	NaN	US	US-AL	Harvest
4	00AR	closed	Newport Hospital & Clinic Heliport	237.0	NaN	US	US-AR	Newport

In [6]:

```
us_cities_demographics_df.head()
```

Out[6]:

	City	State	Median Age	Male Population	Female Population	Total Population	Number of Veterans	Foreign-born
0	Silver Spring	Maryland	33.8	40601.0	41862.0	82463	1562.0	30908.0
1	Quincy	Massachusetts	41.0	44129.0	49500.0	93629	4147.0	32935.0
2	Hoover	Alabama	38.5	38040.0	46799.0	84839	4819.0	8229.0
3	Rancho Cucamonga	California	34.5	88127.0	87105.0	175232	5821.0	33878.0
4	Newark	New Jersey	34.6	138040.0	143873.0	281913	5829.0	86253.0

In [24]:

```
type(us_cities_demographics_df.iloc[2]['Median Age'])
```

Out[24]:

```
numpy.float64
```

In [7]:

```
immigration_data_sample_df.head()
```

Out[7]:

	Unnamed: 0	cicid	i94yr	i94mon	i94cit	i94res	i94port	arrdate	i94mode	i94addr
0	2027561	4084316.0	2016.0	4.0	209.0	209.0	HHW	20566.0	1.0	HI
1	2171295	4422636.0	2016.0	4.0	582.0	582.0	MCA	20567.0	1.0	TX
2	589494	1195600.0	2016.0	4.0	148.0	112.0	OGG	20551.0	1.0	FL
3	2631158	5291768.0	2016.0	4.0	297.0	297.0	LOS	20572.0	1.0	CA
4	3032257	985523.0	2016.0	4.0	111.0	111.0	CHM	20550.0	3.0	NY

5 rows × 29 columns

In [2]:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.\
config("spark.jars.repositories", "https://repos.spark-packages.org/").\
config("spark.jars.packages", "saurfang:spark-sas7bdat:2.0.0-s_2.11").\
enableHiveSupport().getOrCreate()

#df_spark = spark.read.format('com.github.saurfang.sas.spark').load('../../data/18-83510-I94-Data-2016/i94_apr16_sub.sas7bdat')
```

In [14]:

```
airport_code_csv_df_spark = spark.read.csv(airport_code_csv, header=True)
us_cities_demographics_df_spark = spark.read.option("delimiter", ";").csv(us_cities_demographics, header=True)
immigration_data_sample_df_spark = spark.read.csv(immigration_data_sample, header=True)
```

In [11]:

```
airport_code_csv_df_spark.printSchema()
airport_code_csv_df_spark.show(5,truncate=False)
```

```
root
 |-- ident: string (nullable = true)
 |-- type: string (nullable = true)
 |-- name: string (nullable = true)
 |-- elevation_ft: string (nullable = true)
 |-- continent: string (nullable = true)
 |-- iso_country: string (nullable = true)
 |-- iso_region: string (nullable = true)
 |-- municipality: string (nullable = true)
 |-- gps_code: string (nullable = true)
 |-- iata_code: string (nullable = true)
 |-- local_code: string (nullable = true)
 |-- coordinates: string (nullable = true)

+-----+-------------+---------------------------------+------------
+---------+-----------+----------+------------+--------+---------+--
--------+-----------------------------------+
|ident|type         |name                             |elevation_ft
|continent|iso_country|iso_region|municipality|gps_code|iata_code|lo
cal_code|coordinates                        |
+-----+-------------+---------------------------------+------------
+---------+-----------+----------+------------+--------+---------+--
--------+-----------------------------------+
|00A  |heliport     |Total Rf Heliport                |11
|NA       |US         |US-PA     |Bensalem    |00A     |null     |00
A       |-74.93360137939453, 40.07080078125   |
|00AA |small_airport|Aero B Ranch Airport             |3435
|NA       |US         |US-KS     |Leoti       |00AA    |null     |00
AA      |-101.473911, 38.704022               |
|00AK |small_airport|Lowell Field                     |450
|NA       |US         |US-AK     |Anchor Point|00AK    |null     |00
AK      |-151.695999146, 59.94919968          |
|00AL |small_airport|Epps Airpark                     |820
|NA       |US         |US-AL     |Harvest     |00AL    |null     |00
AL      |-86.77030181884766, 34.86479949951172|
|00AR |closed       |Newport Hospital & Clinic Heliport|237
|NA       |US         |US-AR     |Newport     |null    |null     |nu
ll      |-91.254898, 35.6087                  |
+-----+-------------+---------------------------------+------------
+---------+-----------+----------+------------+--------+---------+--
--------+-----------------------------------+
only showing top 5 rows
```

In [15]:

```
us_cities_demographics_df_spark.printSchema()
us_cities_demographics_df_spark.show(5,truncate=False)
```

```
root
 |-- City: string (nullable = true)
 |-- State: string (nullable = true)
 |-- Median Age: string (nullable = true)
 |-- Male Population: string (nullable = true)
 |-- Female Population: string (nullable = true)
 |-- Total Population: string (nullable = true)
 |-- Number of Veterans: string (nullable = true)
 |-- Foreign-born: string (nullable = true)
 |-- Average Household Size: string (nullable = true)
 |-- State Code: string (nullable = true)
 |-- Race: string (nullable = true)
 |-- Count: string (nullable = true)

+----------------+-------------+----------+---------------+---------
--------+----------------+------------------+------------+----------
------------+----------+-------------------------+-----+
|City            |State        |Median Age|Male Population|Female Po
pulation|Total Population|Number of Veterans|Foreign-born|Average Ho
usehold Size|State Code|Race                     |Count|
+----------------+-------------+----------+---------------+---------
--------+----------------+------------------+------------+----------
------------+----------+-------------------------+-----+
|Silver Spring   |Maryland     |33.8      |40601          |41862
|82463           |1562              |30908       |2.6
|MD        |Hispanic or Latino       |25924|
|Quincy          |Massachusetts|41.0      |44129          |49500
|93629           |4147              |32935       |2.39
|MA        |White                    |58723|
|Hoover          |Alabama      |38.5      |38040          |46799
|84839           |4819              |8229        |2.58
|AL        |Asian                    |4759 |
|Rancho Cucamonga|California   |34.5      |88127          |87105
|175232          |5821              |33878       |3.18
|CA        |Black or African-American|24437|
|Newark          |New Jersey   |34.6      |138040         |143873
|281913          |5829              |86253       |2.73
|NJ        |White                    |76402|
+----------------+-------------+----------+---------------+---------
--------+----------------+------------------+------------+----------
------------+----------+-------------------------+-----+
only showing top 5 rows
```

In [13]:

```
immigration_data_sample_df_spark.printSchema()
immigration_data_sample_df_spark.show(5,truncate=False)
```

```
root
 |-- _c0: string (nullable = true)
 |-- cicid: string (nullable = true)
 |-- i94yr: string (nullable = true)
 |-- i94mon: string (nullable = true)
 |-- i94cit: string (nullable = true)
 |-- i94res: string (nullable = true)
 |-- i94port: string (nullable = true)
 |-- arrdate: string (nullable = true)
 |-- i94mode: string (nullable = true)
 |-- i94addr: string (nullable = true)
 |-- depdate: string (nullable = true)
 |-- i94bir: string (nullable = true)
 |-- i94visa: string (nullable = true)
 |-- count: string (nullable = true)
 |-- dtadfile: string (nullable = true)
 |-- visapost: string (nullable = true)
 |-- occup: string (nullable = true)
 |-- entdepa: string (nullable = true)
 |-- entdepd: string (nullable = true)
 |-- entdepu: string (nullable = true)
 |-- matflag: string (nullable = true)
 |-- biryear: string (nullable = true)
 |-- dtaddto: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- insnum: string (nullable = true)
 |-- airline: string (nullable = true)
 |-- admnum: string (nullable = true)
 |-- fltno: string (nullable = true)
 |-- visatype: string (nullable = true)

+-------+---------+------+------+------+------+-------+-------+-----
--+-------+-------+------+-------+-----+--------+--------+-----+----
---+-------+-------+-------+-------+--------+------+------+-------+-
------------+-----+--------+
|_c0    |cicid    |i94yr |i94mon|i94cit|i94res|i94port|arrdate|i94mo
de|i94addr|depdate|i94bir|i94visa|count|dtadfile|visapost|occup|entd
epa|entdepd|entdepu|matflag|biryear|dtaddto |gender|insnum|airline|a
dmnum       |fltno|visatype|
+-------+---------+------+------+------+------+-------+-------+-----
--+-------+-------+------+-------+-----+--------+--------+-----+----
---+-------+-------+-------+-------+--------+------+------+-------+-
------------+-----+--------+
|2027561|4084316.0|2016.0|4.0   |209.0 |209.0 |HHW    |20566.0|1.0
|HI     |20573.0|61.0  |2.0    |1.0  |20160422|null    |null |G
|O      |null   |M      |1955.0 |07202016|F     |null  |JL     |5658
2674633.0|00782|WT      |
|2171295|4422636.0|2016.0|4.0   |582.0 |582.0 |MCA    |20567.0|1.0
|TX     |20568.0|26.0  |2.0    |1.0  |20160423|MTR     |null |G
|R      |null   |M      |1990.0 |10222016|M     |null  |*GA    |9436
1995930.0|XBLNG|B2      |
|589494 |1195600.0|2016.0|4.0   |148.0 |112.0 |OGG    |20551.0|1.0
|FL     |20571.0|76.0  |2.0    |1.0  |20160407|null    |null |G
|O      |null   |M      |1940.0 |07052016|M     |null  |LH     |5578
0468433.0|00464|WT      |
|2631158|5291768.0|2016.0|4.0   |297.0 |297.0 |LOS    |20572.0|1.0
|CA     |20581.0|25.0  |2.0    |1.0  |20160428|DOH     |null |G
|O      |null   |M      |1991.0 |10272016|M     |null  |QR     |9478
9696030.0|00739|B2      |
|3032257|985523.0 |2016.0|4.0   |111.0 |111.0 |CHM    |20550.0|3.0
|NY     |20553.0|19.0  |2.0    |1.0  |20160406|null    |null |Z
```

```
|K      |null   |M      |1997.0 |07042016|F     |null  |null   |4232
2572633.0|LAND |WT      |
+-------+---------+------+------+------+------+-------+-------+-----
--+-------+-------+------+-------+-----+--------+--------+-----+----
---+-------+-------+-------+-------+--------+------+------+-------+-
------------+-----+--------+
only showing top 5 rows
```

In [5]:

```
import os
```

In [ ]:

```
import os
#write to parquet
# df_spark.write.parquet("sas_data")
# df_spark=spark.read.parquet("sas_data")
output_path = 'parquet_output/'
airport_code_csv_df_spark.write.mode('overwrite').parquet(os.path.join(output_pa
th,'airport_code_csv_df_spark'))
```

In [29]:

```
us_cities_demographics_df_spark.columns
```

Out[29]:

```
['City',
 'State',
 'Median Age',
 'Male Population',
 'Female Population',
 'Total Population',
 'Number of Veterans',
 'Foreign-born',
 'Average Household Size',
 'State Code',
 'Race',
 'Count']
```

In [30]:

```
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("City", "city")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("State", "state")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("Median Age", "median_age")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("Male Population", "male_population")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("Female Population", "female_population")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("Total Population", "total_population")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("Number of Veterans", "number_of_veterans")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("Foreign-born", "foreign_born")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("Average Household Size", "average_household_size")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("State Code", "state_code")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("Race", "race")
us_cities_demographics_df_spark = us_cities_demographics_df_spark.withColumnRena
med("Count", "count")

us_cities_demographics_df_spark.write.parquet(os.path.join(output_path,'us_citie
s_demographics_df_spark'))
```

In [25]:

```
immigration_data_sample_df_spark.write.parquet(os.path.join(output_path,'immigra
tion_data_sample_df_spark'))
```

Step 2: Explore and Assess the Data

Explore the Data

Identify data quality issues, like missing values, duplicate data, etc.

Cleaning Steps

Document steps necessary to clean the data

In [3]:

```
import os
#read parquest data
output_path = 'parquet_output/'
airport_code_csv_df_spark = spark.read.parquet(os.path.join(output_path,'airport
_code_csv_df_spark'))
us_cities_demographics_df_spark = spark.read.parquet(os.path.join(output_path,'u
s_cities_demographics_df_spark'))
immigration_data_sample_df_spark = spark.read.parquet(os.path.join(output_pat
h,'immigration_data_sample_df_spark'))
```

In [34]:

```
airport_code_csv_df_spark.printSchema()
```

```
root
 |-- ident: string (nullable = true)
 |-- type: string (nullable = true)
 |-- name: string (nullable = true)
 |-- elevation_ft: string (nullable = true)
 |-- continent: string (nullable = true)
 |-- iso_country: string (nullable = true)
 |-- iso_region: string (nullable = true)
 |-- municipality: string (nullable = true)
 |-- gps_code: string (nullable = true)
 |-- iata_code: string (nullable = true)
 |-- local_code: string (nullable = true)
 |-- coordinates: string (nullable = true)
```

In [35]:

```
us_cities_demographics_df_spark.printSchema()
```

```
root
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- median_age: string (nullable = true)
 |-- male_population: string (nullable = true)
 |-- female_population: string (nullable = true)
 |-- total_population: string (nullable = true)
 |-- number_of_veterans: string (nullable = true)
 |-- foreign_born: string (nullable = true)
 |-- average_household_size: string (nullable = true)
 |-- state_code: string (nullable = true)
 |-- race: string (nullable = true)
 |-- count: string (nullable = true)
```

In [32]:

```
immigration_data_sample_df_spark.printSchema()
```

```
root
 |-- _c0: string (nullable = true)
 |-- cicid: string (nullable = true)
 |-- i94yr: string (nullable = true)
 |-- i94mon: string (nullable = true)
 |-- i94cit: string (nullable = true)
 |-- i94res: string (nullable = true)
 |-- i94port: string (nullable = true)
 |-- arrdate: string (nullable = true)
 |-- i94mode: string (nullable = true)
 |-- i94addr: string (nullable = true)
 |-- depdate: string (nullable = true)
 |-- i94bir: string (nullable = true)
 |-- i94visa: string (nullable = true)
 |-- count: string (nullable = true)
 |-- dtadfile: string (nullable = true)
 |-- visapost: string (nullable = true)
 |-- occup: string (nullable = true)
 |-- entdepa: string (nullable = true)
 |-- entdepd: string (nullable = true)
 |-- entdepu: string (nullable = true)
 |-- matflag: string (nullable = true)
 |-- biryear: string (nullable = true)
 |-- dtaddto: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- insnum: string (nullable = true)
 |-- airline: string (nullable = true)
 |-- admnum: string (nullable = true)
 |-- fltno: string (nullable = true)
 |-- visatype: string (nullable = true)
```

In [4]:

```
# Performing cleaning tasks here
# clean immigration_data_sample_df_spark
immigration_data_sample_df_spark_clean = immigration_data_sample_df_spark.na.fil
l({'_c0': 0.0, 'i94port': 0.0,'i94mode': 0.0, 'i94addr': 'NULL_VALUE','depdate':
0.0, 'i94bir': 'NULL_VALUE', \
                          'i94visa': 0.0, 'count': 0.0, 'dtadfile': 'NULL_VALUE',
'visapost': 'NULL_VALUE', \
                          'occup': 'NULL_VALUE', 'entdepa': 'NULL_VALUE', 'entdep
d': 'NULL_VALUE', 'entdepu': 'NULL_VALUE', \
                          'matflag': 'NULL_VALUE', 'biryear': 0.0, 'dtaddto': 'NUL
L_VALUE', 'gender': 'NULL_VALUE', \
                          'insnum': 'NULL_VALUE', 'airline': 'NULL_VALUE', 'admnu
m': 0.0, 'fltno': 'NULL_VALUE', 'visatype': 'NULL_VALUE'})
```

In [ ]:

```
immigration_data_sample_df_spark_clean.printSchema()
```

In [6]:

```
immigration_data_sample_df_spark_clean.show(2,truncate=False)
```

```
+-------+---------+------+------+------+------+-------+-------+-----
--+-------+-------+------+-------+-----+--------+----------+--------
--+-------+-------+----------+-------+-------+--------+------+------
----+-------+-------------+-----+--------+
|_c0    |cicid    |i94yr |i94mon|i94cit|i94res|i94port|arrdate|i94mo
de|i94addr|depdate|i94bir|i94visa|count|dtadfile|visapost  |occup
|entdepa|entdepd|entdepu   |matflag|biryear|dtaddto |gender|insnum
|airline|admnum       |fltno|visatype|
+-------+---------+------+------+------+------+-------+-------+-----
--+-------+-------+------+-------+-----+--------+----------+--------
--+-------+-------+----------+-------+-------+--------+------+------
----+-------+-------------+-----+--------+
|2027561|4084316.0|2016.0|4.0   |209.0 |209.0 |HHW    |20566.0|1.0
|HI     |20573.0|61.0  |2.0    |1.0  |20160422|NULL_VALUE|NULL_VALUE
|G      |O      |NULL_VALUE|M      |1955.0 |07202016|F     |NULL_VAL
UE|JL     |56582674633.0|00782|WT      |
|2171295|4422636.0|2016.0|4.0   |582.0 |582.0 |MCA    |20567.0|1.0
|TX     |20568.0|26.0  |2.0    |1.0  |20160423|MTR       |NULL_VALUE
|G      |R      |NULL_VALUE|M      |1990.0 |10222016|M     |NULL_VAL
UE|*GA    |94361995930.0|XBLNG|B2      |
+-------+---------+------+------+------+------+-------+-------+-----
--+-------+-------+------+-------+-----+--------+----------+--------
--+-------+-------+----------+-------+-------+--------+------+------
----+-------+-------------+-----+--------+
only showing top 2 rows
```

In [ ]:

```

```

Step 3: Define the Data Model

3.1 Conceptual Data Model

Map out the conceptual data model and explain why you chose that model

3.2 Mapping Out Data Pipelines

List the steps necessary to pipeline the data into the chosen data model

Step 4: Run Pipelines to Model the Data

4.1 Create the data model

Build the data pipelines to create the data model.

4.1.1 create admissions table

In [8]:

```
# Write code here
# Create admissions table
immigration_data_sample_df_spark_clean.createOrReplaceTempView("admissions")
admissions_table = spark.sql("""
    SELECT  DISTINCT admnum   AS admission_id,
                     i94res   AS country_code,
                     i94bir   AS age,
                     gender   AS gender
    FROM admissions
    ORDER BY age ASC
""")
admissions_table.printSchema()
admissions_table.show(5)
```

```
root
 |-- admission_id: string (nullable = false)
 |-- country_code: string (nullable = true)
 |-- age: string (nullable = false)
 |-- gender: string (nullable = false)

+-------------+------------+---+------+
| admission_id|country_code|age|gender|
+-------------+------------+---+------+
|  721095085.0|       692.0|1.0|     F|
|55972037033.0|       108.0|1.0|     M|
|93323727930.0|       687.0|1.0|     M|
|24292172827.0|       276.0|1.0|     M|
|92539917330.0|       692.0|1.0|     M|
+-------------+------------+---+------+
only showing top 5 rows
```

In [9]:

```
# write to parquet
admissions_table.write.parquet(os.path.join(output_path,'admissions_table_df_spa
rk'))
admissions_table_df_spark = spark.read.parquet(os.path.join(output_path,'admissi
ons_table_df_spark'))
```

In [10]:

```
admissions_table_df_spark.printSchema()
admissions_table_df_spark.show(2)
```

```
root
 |-- admission_id: string (nullable = true)
 |-- country_code: string (nullable = true)
 |-- age: string (nullable = true)
 |-- gender: string (nullable = true)

+-------------+------------+----+------+
| admission_id|country_code| age|gender|
+-------------+------------+----+------+
|59501691433.0|       103.0|34.0|     M|
|56357999633.0|       438.0|34.0|     F|
+-------------+------------+----+------+
only showing top 2 rows
```

4.1.2 create us_airports table

In [16]:

```
# Create airports table only US airports
airport_code_csv_df_spark.createOrReplaceTempView("us_airports")
us_airports_table = spark.sql("""
    SELECT  DISTINCT name   AS airport_name,
                     SUBSTRING(iso_region,4)   AS state_code,
                     iso_country   AS iso_country
    FROM us_airports
    WHERE iso_region LIKE 'US%'
    ORDER BY airport_name ASC
""")
us_airports_table.printSchema()
us_airports_table.show(3)
```

```
root
 |-- airport_name: string (nullable = true)
 |-- state_code: string (nullable = true)
 |-- iso_country: string (nullable = true)

+--------------------+----------+-----------+
|        airport_name|state_code|iso_country|
+--------------------+----------+-----------+
|"Fly ""N"" K Airp...|        AR|         US|
|"Holict ""Private...|        TX|         US|
|"NJSP - Troop ""A...|        NJ|         US|
+--------------------+----------+-----------+
only showing top 3 rows
```

In [11]:

```
# write to parquet
us_airports_table.write.mode("overwrite").parquet(os.path.join(output_path,'us_airports_table_df_spark'))
us_airports_table_df_spark = spark.read.parquet(os.path.join(output_path,'us_airports_table_df_spark'))
```

In [12]:

```
us_airports_table_df_spark.printSchema()
us_airports_table_df_spark.show(2)
```

```
root
 |-- airport_name: string (nullable = true)
 |-- state_code: string (nullable = true)
 |-- iso_country: string (nullable = true)

+--------------------+----------+-----------+
|        airport_name|state_code|iso_country|
+--------------------+----------+-----------+
|United Ca Bank Da...|        CA|         US|
|United Coal Heliport|        VA|         US|
+--------------------+----------+-----------+
only showing top 2 rows
```

4.1.3 create us_city_demographics table

In [15]:

```
# Create admissions table
us_cities_demographics_df_spark.createOrReplaceTempView("us_city_demographics")
us_city_demographics_table = spark.sql("""
    SELECT  DISTINCT city   AS city,
                     state   AS state,
                     state_code   AS state_code,
                     median_age   AS median_age,
                     male_population   AS male_population,
                     female_population   AS female_population,
                     total_population   AS total_population
    FROM us_city_demographics
    ORDER BY city ASC
""")
us_city_demographics_table.printSchema()
us_city_demographics_table.show(5)
```

```
root
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- state_code: string (nullable = true)
 |-- median_age: string (nullable = true)
 |-- male_population: string (nullable = true)
 |-- female_population: string (nullable = true)
 |-- total_population: string (nullable = true)

+-------+----------+----------+----------+---------------+----------
-------+----------------+
|   city|     state|state_code|median_age|male_population|female_pop
ulation|total_population|
+-------+----------+----------+----------+---------------+----------
-------+----------------+
|Abilene|     Texas|        TX|      31.3|          65212|
60664|          125876|
|  Akron|      Ohio|        OH|      38.1|          96886|
100667|          197553|
|Alafaya|   Florida|        FL|      33.5|          39504|
45760|           85264|
|Alameda|California|        CA|      41.4|          37747|
40867|           78614|
| Albany|  New York|        NY|      32.8|          47627|
50825|           98452|
+-------+----------+----------+----------+---------------+----------
-------+----------------+
only showing top 5 rows
```

In [7]:

```
# write to parquet
us_city_demographics_table.write.parquet(os.path.join(output_path,'us_city_demog
raphics_table_df_spark'))
us_city_demographics_table_df_spark = spark.read.parquet(os.path.join(output_pat
h,'us_city_demographics_table_df_spark'))
```

In [8]:

```
us_city_demographics_table_df_spark.printSchema()
us_city_demographics_table_df_spark.show(2)
```

```
root
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- state_code: string (nullable = true)
 |-- median_age: string (nullable = true)
 |-- male_population: string (nullable = true)
 |-- female_population: string (nullable = true)
 |-- total_population: string (nullable = true)

+-------+----------+----------+----------+---------------+----------
-------+----------------+
|   city|     state|state_code|median_age|male_population|female_pop
ulation|total_population|
+-------+----------+----------+----------+---------------+----------
-------+----------------+
|Atlanta|   Georgia|        GA|      33.8|         223960|
239915|          463875|
| Auburn|Washington|        WA|      37.1|          36837|
39743|          76580|
+-------+----------+----------+----------+---------------+----------
-------+----------------+
only showing top 2 rows
```

4.1.4 create arrival_time table

In [22]:

```
@udf(types_.TimestampType())
def convert_to_timestamp(days):
    start_ = datetime(1960,1,1)
    print(days)
    duration = timedelta(days=int(float(days)))
    return (start_+duration)
```

In [13]:

```
temp = convert_to_timestamp(immigration_data_sample_df_spark.arrdate)
```

In [15]:

```
type(temp)
```

Out[15]:

```
pyspark.sql.column.Column
```

In [23]:

```
# Create arrival time table
immigration_data_sample_df_spark_clean = immigration_data_sample_df_spark_clean.
withColumn("arrival_time", convert_to_timestamp(immigration_data_sample_df_spar
k.arrdate))
```

In [24]:

```
immigration_data_sample_df_spark_clean.select("arrival_time").show()
```

```
+-------------------+
|       arrival_time|
+-------------------+
|2016-04-22 00:00:00|
|2016-04-23 00:00:00|
|2016-04-07 00:00:00|
|2016-04-28 00:00:00|
|2016-04-06 00:00:00|
|2016-04-08 00:00:00|
|2016-04-12 00:00:00|
|2016-04-02 00:00:00|
|2016-04-28 00:00:00|
|2016-04-01 00:00:00|
|2016-04-07 00:00:00|
|2016-04-27 00:00:00|
|2016-04-15 00:00:00|
|2016-04-26 00:00:00|
|2016-04-08 00:00:00|
|2016-04-01 00:00:00|
|2016-04-06 00:00:00|
|2016-04-13 00:00:00|
|2016-04-24 00:00:00|
|2016-04-14 00:00:00|
+-------------------+
only showing top 20 rows
```

In [25]:

```
immigration_data_sample_df_spark_clean.createOrReplaceTempView("arrival_time")
arrival_time_table = spark.sql("""
    SELECT DISTINCT  arrival_time           AS arrival_time,
                     hour(arrival_time)       AS arrival_hour,
                     day(arrival_time)        AS arrival_day,
                     month(arrival_time)      AS arrival_month,
                     year(arrival_time)       AS arrival_year
    FROM arrival_time
""")
arrival_time_table.printSchema()
arrival_time_table.show(5, truncate=False)
```

```
root
 |-- arrival_time: timestamp (nullable = true)
 |-- arrival_hour: integer (nullable = true)
 |-- arrival_day: integer (nullable = true)
 |-- arrival_month: integer (nullable = true)
 |-- arrival_year: integer (nullable = true)

+-------------------+------------+-----------+-------------+--------
----+
|arrival_time       |arrival_hour|arrival_day|arrival_month|arrival_
year|
+-------------------+------------+-----------+-------------+--------
----+
|2016-04-01 00:00:00|0           |1          |4            |2016
|
|2016-04-26 00:00:00|0           |26         |4            |2016
|
|2016-04-02 00:00:00|0           |2          |4            |2016
|
|2016-04-05 00:00:00|0           |5          |4            |2016
|
|2016-04-07 00:00:00|0           |7          |4            |2016
|
+-------------------+------------+-----------+-------------+--------
----+
only showing top 5 rows
```

In [26]:

```
arrival_time_table.count()
```

Out[26]:

```
30
```

In [27]:

```
# write to parquet
arrival_time_table.write.parquet(os.path.join(output_path,'arrival_time_table_df
_spark'))
arrival_time_table_df_spark = spark.read.parquet(os.path.join(output_path,'arriv
al_time_table_df_spark'))
```

In [ ]:

```
```

In [8]:

```
# # Join us city demographics with airports
# us_city_df_spark_joined = us_city_demographics_table_df_spark.join(us_airports
_table_df_spark, ['state_code'])
```

In [9]:

```
# us_city_df_spark_joined.count()
```

Out[9]:

```
485916
```

In [20]:

```
# us_city_df_spark_joined.printSchema()
# us_city_df_spark_joined.show(2)
```

```
root
 |-- state_code: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- median_age: string (nullable = true)
 |-- male_population: string (nullable = true)
 |-- female_population: string (nullable = true)
 |-- total_population: string (nullable = true)
 |-- airport_name: string (nullable = true)
 |-- iso_country: string (nullable = true)

+----------+---------+----------+----------+---------------+--------
---------+----------------+--------------------+-----------+
|state_code|     city|     state|median_age|male_population|female_p
opulation|total_population|        airport_name|iso_country|
+----------+---------+----------+----------+---------------+--------
---------+----------------+--------------------+-----------+
|        CA|Yuba City|California|      34.5|          33654|
33290|           66944|United Ca Bank Da...|         US|
|        CA| Whittier|California|      36.1|          44397|
43039|           87436|United Ca Bank Da...|         US|
+----------+---------+----------+----------+---------------+--------
---------+----------------+--------------------+-----------+
only showing top 2 rows
```

In [24]:

```
# # join immigrations with us city above
# us_immigrations_df_spark_joined = immigration_data_sample_df_spark_clean.join
(us_city_df_spark_joined,
#                                                                          i
mmigration_data_sample_df_spark_clean.i94addr == us_city_df_spark_joined.state_c
ode)
```

In [25]:

```
# us_immigrations_df_spark_joined.printSchema()
# us_immigrations_df_spark_joined.show(2)
```

```
root
 |-- _c0: string (nullable = false)
 |-- cicid: string (nullable = true)
 |-- i94yr: string (nullable = true)
 |-- i94mon: string (nullable = true)
 |-- i94cit: string (nullable = true)
 |-- i94res: string (nullable = true)
 |-- i94port: string (nullable = false)
 |-- arrdate: string (nullable = true)
 |-- i94mode: string (nullable = false)
 |-- i94addr: string (nullable = false)
 |-- depdate: string (nullable = false)
 |-- i94bir: string (nullable = false)
 |-- i94visa: string (nullable = false)
 |-- count: string (nullable = false)
 |-- dtadfile: string (nullable = false)
 |-- visapost: string (nullable = false)
 |-- occup: string (nullable = false)
 |-- entdepa: string (nullable = false)
 |-- entdepd: string (nullable = false)
 |-- entdepu: string (nullable = false)
 |-- matflag: string (nullable = false)
 |-- biryear: string (nullable = false)
 |-- dtaddto: string (nullable = false)
 |-- gender: string (nullable = false)
 |-- insnum: string (nullable = false)
 |-- airline: string (nullable = false)
 |-- admnum: string (nullable = false)
 |-- fltno: string (nullable = false)
 |-- visatype: string (nullable = false)
 |-- state_code: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- median_age: string (nullable = true)
 |-- male_population: string (nullable = true)
 |-- female_population: string (nullable = true)
 |-- total_population: string (nullable = true)
 |-- airport_name: string (nullable = true)
 |-- iso_country: string (nullable = true)

+-------+---------+------+------+------+------+-------+-------+-----
--+-------+-------+------+-------+-----+--------+----------+--------
--+-------+-------+----------+-------+-------+--------+----------+--
--------+-------+-------------+-----+--------+----------+---------+-
---------+----------+---------------+-----------------+-------------
---+--------------------+-----------+
|    _c0|    cicid| i94yr|i94mon|i94cit|i94res|i94port|arrdate|i94mo
de|i94addr|depdate|i94bir|i94visa|count|dtadfile|  visapost|     occ
up|entdepa|entdepd|   entdepu|matflag|biryear| dtaddto|    gender|
insnum|airline|       admnum|fltno|visatype|state_code|     city|
state|median_age|male_population|female_population|total_population|
airport_name|iso_country|
+-------+---------+------+------+------+------+-------+-------+-----
--+-------+-------+------+-------+-----+--------+----------+--------
--+-------+-------+----------+-------+-------+--------+----------+--
--------+-------+-------------+-----+--------+----------+---------+-
---------+----------+---------------+-----------------+-------------
---+--------------------+-----------+
|1920712|3874218.0|2016.0|   4.0| 148.0| 112.0|    SFR|20565.0|
1.0|     CA|20582.0|  49.0|    2.0|  1.0|20160421|NULL_VALUE|NULL_VA
LUE|      O|      O|NULL_VALUE|      M| 1967.0|07192016|NULL_VALUE|N
```

```
ULL_VALUE|     LH|56534271933.0|00454|      WT|        CA|Yuba City|
California|      34.5|          33654|            33290|           6
6944|United Ca Bank Da...|         US|
| 697642|1407986.0|2016.0|   4.0| 245.0| 245.0|    SFR|20552.0|
1.0|     CA|20566.0|  60.0|    2.0|  1.0|20160408|       BEJ|NULL_VA
LUE|      G|      O|NULL_VALUE|      M| 1956.0|10072016|         M|N
ULL_VALUE|     MU|93032766930.0|00589|      B2|        CA|Yuba City|
California|      34.5|          33654|            33290|           6
6944|United Ca Bank Da...|         US|
+-------+---------+------+------+------+------+-------+-------+-----
--+-------+-------+------+-------+-----+--------+----------+--------
--+-------+-------+----------+-------+-------+--------+----------+--
--------+-------+-------------+-----+--------+----------+---------+-
---------+----------+---------------+-----------------+------------
---+--------------------+-----------+
only showing top 2 rows
```

In [28]:

```
# us_immigrations_df_spark_joined=us_immigrations_df_spark_joined.withColumn("im
migration_id", monotonically_increasing_id())
```

In [ ]:

```
# us_immigrations_df_spark_joined.printSchema()
# us_immigrations_df_spark_joined.show(5)
```

In [28]:

```
# join immigrations with us city above
us_immigrations_df_spark_joined = immigration_data_sample_df_spark_clean.join(ar
rival_time_table_df_spark, ['arrival_time'])
```

In [29]:

```
us_immigrations_df_spark_joined.show(1)
```

```
+------------------+-------+---------+------+------+------+------+-
------+-------+-------+-------+-------+------+-------+-----+--------
+----------+----------+-------+-------+----------+-------+-------+--
------+------+----------+-------+-------------+-----+--------+------
------+-----------+-------------+------------+
|       arrival_time|    _c0|    cicid| i94yr|i94mon|i94cit|i94res|i
94port|arrdate|i94mode|i94addr|depdate|i94bir|i94visa|count|dtadfile
|  visapost|     occup|entdepa|entdepd|   entdepu|matflag|biryear| d
taddto|gender|    insnum|airline|       admnum|fltno|visatype|arriva
l_hour|arrival_day|arrival_month|arrival_year|
+------------------+-------+---------+------+------+------+------+-
------+-------+-------+-------+-------+------+-------+-----+--------
+----------+----------+-------+-------+----------+-------+-------+--
------+------+----------+-------+-------------+-----+--------+------
------+-----------+-------------+------------+
|2016-04-22 00:00:00|2027561|4084316.0|2016.0|   4.0| 209.0| 209.0|
HHW|20566.0|    1.0|     HI|20573.0|  61.0|    2.0|  1.0|20160422|NU
LL_VALUE|NULL_VALUE|      G|      O|NULL_VALUE|      M| 1955.0|07202
016|     F|NULL_VALUE|     JL|56582674633.0|00782|      WT|
0|         22|            4|        2016|
+------------------+-------+---------+------+------+------+------+-
------+-------+-------+-------+-------+------+-------+-----+--------
+----------+----------+-------+-------+----------+-------+-------+--
------+------+----------+-------+-------------+-----+--------+------
------+-----------+-------------+------------+
only showing top 1 row
```

In [30]:

```
us_immigrations_df_spark_joined.count()
```

Out[30]:

```
1000
```

4.1.5 create us_immigrations table

In [31]:

```
us_immigrations_df_spark_joined=us_immigrations_df_spark_joined.withColumn("immi
gration_id", monotonically_increasing_id())
```

In [34]:

```
# Create admissions table
us_immigrations_df_spark_joined.createOrReplaceTempView("us_immigrations")
us_immigrations_table = spark.sql("""
    SELECT  DISTINCT immigration_id   AS immigration_id,
                     i94addr   AS state_code,
                     admnum   AS admission_id,
                     airline   AS air_line,
                     fltno   AS flight_number,
                     arrival_time   AS arrival_time,
                     depdate   AS departure_date,
                     biryear   AS birth_year
    FROM us_immigrations
    ORDER BY arrival_time
""")
us_immigrations_table.printSchema()
us_immigrations_table.show(2)
```

```
root
 |-- immigration_id: long (nullable = false)
 |-- state_code: string (nullable = false)
 |-- admission_id: string (nullable = false)
 |-- air_line: string (nullable = false)
 |-- flight_number: string (nullable = false)
 |-- arrival_time: timestamp (nullable = true)
 |-- departure_date: string (nullable = false)
 |-- birth_year: string (nullable = false)

+--------------+----------+-------------+--------+-------------+----
---------------+--------------+----------+
|immigration_id|state_code| admission_id|air_line|flight_number|
arrival_time|departure_date|birth_year|
+--------------+----------+-------------+--------+-------------+----
---------------+--------------+----------+
|           777|        NV|55463158833.0|      BA|        00275|2016
-04-01 00:00:00|       20553.0|    1981.0|
|           791|NULL_VALUE|44162582033.0|      MU|        00763|2016
-04-01 00:00:00|       20550.0|    1986.0|
+--------------+----------+-------------+--------+-------------+----
---------------+--------------+----------+
only showing top 2 rows
```

In [35]:

```
us_immigrations_table.count()
```

Out[35]:

```
1000
```

In [37]:

```
# write to parquet
us_immigrations_table.write.mode("overwrite").parquet(os.path.join(output_pat
h,'us_immigrations_table_df_spark'))
us_immigrations_table_df_spark = spark.read.parquet(os.path.join(output_path,'us
_immigrations_table_df_spark'))
```

In [38]:

```
us_immigrations_table_df_spark.printSchema()
us_immigrations_table_df_spark.count()
```

```
root
 |-- immigration_id: long (nullable = true)
 |-- state_code: string (nullable = true)
 |-- admission_id: string (nullable = true)
 |-- air_line: string (nullable = true)
 |-- flight_number: string (nullable = true)
 |-- arrival_time: timestamp (nullable = true)
 |-- departure_date: string (nullable = true)
 |-- birth_year: string (nullable = true)
```

Out[38]:

```
1000
```

In [ ]:

```

```

4.2 Data Quality Checks

Explain the data quality checks you'll perform to ensure the pipeline ran as expected. These could include:

- Integrity constraints on the relational database (e.g., unique key, data type, etc.)
- Unit tests for the scripts to ensure they are doing the right thing
- Source/Count checks to ensure completeness

Run Quality Checks

In [41]:

```
admissions_table_df_spark = spark.read.parquet(os.path.join(output_path,'admissions_table_df_spark'))
us_airports_table_df_spark = spark.read.parquet(os.path.join(output_path,'us_airports_table_df_spark'))
us_city_demographics_table_df_spark = spark.read.parquet(os.path.join(output_path,'us_city_demographics_table_df_spark'))
arrival_time_table_df_spark = spark.read.parquet(os.path.join(output_path,'arrival_time_table_df_spark'))
us_immigrations_table_df_spark = spark.read.parquet(os.path.join(output_path,'us_immigrations_table_df_spark'))
```

In [53]:

```
# Perform quality checks here
quality_results = {
    'admissions':{'count_record':0, 'count_null':0, 'quality_check':"Not checked
yet"},
    'airports':{'count_record':0, 'count_null':0, 'quality_check':"Not checked y
et"},
    'us_city_demographics':{'count_record':0, 'count_null':0, 'quality_check':"N
ot checked yet"},
    'arrival_time':{'count_record':0, 'count_null':0, 'quality_check':"Not check
ed yet"},
    'immigrations':{'count_record':0, 'count_null':0, 'quality_check':"Not check
ed yet"}
}
```

4.2.1 check admissions table

In [54]:

```
admissions_table_df_spark.createOrReplaceTempView("admissions_table")
admissions_check = spark.sql("""
    SELECT COUNT(1)
    FROM admissions_table
    WHERE admission_id IS NULL OR admission_id == ""
""")
result1 = admissions_check.collect()[0][0]
```

In [55]:

```
admissions_check.show()
```

```
+--------+
|count(1)|
+--------+
|       0|
+--------+
```

In [56]:

```
admissions_check = spark.sql("""
    SELECT COUNT(1)
    FROM admissions_table
""")
result2 = admissions_check.collect()[0][0]
admissions_check.show()
```

```
+--------+
|count(1)|
+--------+
|    1000|
+--------+
```

In [57]:

```
def check_quality(table_name,quality_results, result1, result2):
    print('count_null: ', result1)
    print('count_record: ', result2)
    quality_results[table_name]['count_null'] = result1
    quality_results[table_name]['count_record'] = result2
    if result1 == 0 and result2 > 0:
        quality_results[table_name]['quality_check'] = 'GOOD'
    else:
        quality_results[table_name]['quality_check'] = 'NOT GOOD'

    return quality_results
```

In [58]:

```
quality_results = check_quality('admissions', quality_results, result1, result2)
```

```
count_null:  0
count_record:  1000
```

4.2.2 check us_airports table

In [59]:

```
us_airports_table_df_spark.createOrReplaceTempView("us_airports_table")
us_airports_check = spark.sql("""
    SELECT COUNT(1)
    FROM us_airports_table
    WHERE airport_name IS NULL OR airport_name == ""
""")
result1 = us_airports_check.collect()[0][0]
us_airports_check = spark.sql("""
    SELECT COUNT(1)
    FROM us_airports_table
""")
result2 = us_airports_check.collect()[0][0]
quality_results = check_quality('airports', quality_results, result1, result2)
```

```
count_null:  0
count_record:  22608
```

4.2.3 check us_city_demographics table

In [60]:

```
us_city_demographics_table_df_spark.createOrReplaceTempView("us_city_demographic
s_table")
us_city_demographics_check = spark.sql("""
    SELECT COUNT(1)
    FROM us_city_demographics_table
    WHERE city IS NULL OR city == ""
""")
result1 = us_city_demographics_check.collect()[0][0]
us_city_demographics_check = spark.sql("""
    SELECT COUNT(1)
    FROM us_city_demographics_table
""")
result2 = us_city_demographics_check.collect()[0][0]
quality_results = check_quality('us_city_demographics', quality_results, result
1, result2)
```

```
count_null:  0
count_record:  596
```

4.2.4 check arrival_time table

In [61]:

```
arrival_time_table_df_spark.createOrReplaceTempView("arrival_time_table")
arrival_time_check = spark.sql("""
    SELECT COUNT(1)
    FROM arrival_time_table
    WHERE arrival_time IS NULL OR arrival_time == ""
""")
result1 = arrival_time_check.collect()[0][0]
arrival_time_check = spark.sql("""
    SELECT COUNT(1)
    FROM arrival_time_table
""")
result2 = arrival_time_check.collect()[0][0]
quality_results = check_quality('arrival_time', quality_results, result1, result
2)
```

```
count_null:  0
count_record:  30
```

4.2.5 check us_immigrations table

In [64]:

```
us_immigrations_table_df_spark.createOrReplaceTempView("us_immigrations_table")
us_immigrations_check = spark.sql("""
    SELECT COUNT(1)
    FROM us_immigrations_table
    WHERE immigration_id IS NULL OR immigration_id == "" OR
          admission_id IS NULL OR admission_id == "" OR
          state_code IS NULL OR state_code == "" OR
          arrival_time IS NULL OR arrival_time == ""
""")
result1 = us_immigrations_check.collect()[0][0]
us_immigrations_check = spark.sql("""
    SELECT COUNT(1)
    FROM us_immigrations_table
""")
result2 = us_immigrations_check.collect()[0][0]
quality_results = check_quality('immigrations', quality_results, result1, result
2)
```

```
count_null:  0
count_record:  1000
```

results of checking quality

In [65]:

```
quality_results
```

Out[65]:

```
{'admissions': {'count_record': 1000,
  'count_null': 0,
  'quality_check': 'GOOD'},
 'airports': {'count_record': 22608, 'count_null': 0, 'quality_chec
k': 'GOOD'},
 'us_city_demographics': {'count_record': 596,
  'count_null': 0,
  'quality_check': 'GOOD'},
 'arrival_time': {'count_record': 30,
  'count_null': 0,
  'quality_check': 'GOOD'},
 'immigrations': {'count_record': 1000,
  'count_null': 0,
  'quality_check': 'GOOD'}}
```

4.3 Data dictionary

Create a data dictionary for your data model. For each field, provide a brief description of what the data is and where it came from. You can include the data dictionary in the notebook or in a separate file.

Data dictionary for this project is described in described_dictionary.json file

Step 5: Complete Project Write Up

- Clearly state the rationale for the choice of tools and technologies for the project.
- Propose how often the data should be updated and why.
- Write a description of how you would approach the problem differently under the following scenarios:
  - The data was increased by 100x.
  - The data populates a dashboard that must be updated on a daily basis by 7am every day.
  - The database needed to be accessed by 100+ people.

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

- I use Pandas and Spark in Python, because it have many libraries to read, write, clean, process data
- These libraries are easy to use, have many special documents on Internet

5.2 Propose how often the data should be updated and why.

- In this project, dataset was limited. So, i use local storage to store input and outpu data
- When dataset is greater than, we can use AWS S3 instead to avoid extra costs

5.3 Write a description of how you would approach the problem differently under the following scenarios:

- The data was increased by 100x
  - Input and output data should be stored in AWS S3, because it can scale according to need
  - Spark Cluster should be used to process parallel
  - Database should use AWS RDS
- The data populates a dashboard that must be updated on a daily basis by 7am every day
  - ETL should be only process changed information (input, output) to optimize processing
  - Output data should be stored AWS RDS and have backup plans to make it available all time
- The database needed to be accessed by 100+ people
  - Output data should be stored AWS RDS and have backup plans to make it available all time
  - Create many replicas and use loadbalancer to balance query processing
  - Store results of complex queries which take time for faster response

In [ ]:

In [ ]:

In [ ]: