Capstone Project-backup

June 7, 2021

1 Project Title

1.0.1 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 [40]: # Do all imports and installs here
         from collections import defaultdict
         from datetime import datetime, timedelta
         from pyspark.sql.functions import udf
         import psycopg2
         import re
         import pandas as pd
```

1.0.2 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?

1.0.3 Dataset:

- I94 Immigration Data: This data comes from the US National Tourism and Trade Office. A data dictionary is included in the workspace. This is where the data comes from. There's a sample file so you can take a look at the data in csv format before reading it all in. You do not have to use the entire dataset, just use what you need to accomplish the goal you set at the beginning of the project.
- World Temperature Data: This dataset came from Kaggle.
- U.S. City Demographic Data: This data comes from OpenSoft.
- Airport Code Table: This is a simple table of airport codes and corresponding cities

Immigration Data

```
In [8]: fname = '../../data/18-83510-I94-Data-2016/i94_apr16_sub.sas7bdat'
```

```
In [9]: def split_large_data_sas(path,file):
            path_name=path+'chunk'
            chunk_size=100000
            batch_no=1
            for chunk in pd.read_sas(file, encoding="ISO-8859-1",chunksize=chunk_size):
                chunk.to_csv(path_name+str(batch_no)+'.csv',index=False)
                batch_no+=1
```

it take sometime to split large dataset into small ones, be patient, if you want to checkout where the smaller datasets are, go to ./data folder.

```
In [10]: split_large_data_sas("./data/",fname)
```

```
In [11]: df = pd.read_csv('./data/chunk1.csv')
         df.head()
```

```
Out[11]:    cicid   i94yr  i94mon  i94cit  i94res i94port  arrdate  i94mode i94addr  \
         0    6.0  2016.0     4.0   692.0   692.0     XXX  20573.0      NaN     NaN   
         1    7.0  2016.0     4.0   254.0   276.0     ATL  20551.0      1.0      AL   
         2   15.0  2016.0     4.0   101.0   101.0     WAS  20545.0      1.0      MI   
         3   16.0  2016.0     4.0   101.0   101.0     NYC  20545.0      1.0      MA   
         4   17.0  2016.0     4.0   101.0   101.0     NYC  20545.0      1.0      MA   

            depdate   ...     entdepu  matflag      biryear   dtaddto gender insnum  \
         0      NaN   ...           U      NaN  1979.000101  10282016    NaN    NaN   
         1      NaN   ...           Y      NaN  1991.000185       D/S      M    NaN   
         2  20691.0   ...         NaN        M  1961.000910  09302016      M    NaN   
         3  20567.0   ...         NaN        M  1988.000422  09302016    NaN    NaN   
         4  20567.0   ...         NaN        M  2012.000681  09302016    NaN    NaN   

           airline        admnum  fltno visatype  
         0     NaN  1.897628e+09    NaN       B2  
         1     NaN  3.736796e+09  00296       F1  
         2      OS  6.666432e+08     93       B2  
         3      AA  9.246846e+10  00199       B2  
         4      AA  9.246846e+10  00199       B2  

         [5 rows x 28 columns]
```

Temperature Data

```
In [12]: temperature_data = '../../data2/GlobalLandTemperaturesByCity.csv'
In [13]: def split_large_data_csv(path,file):
             path_name=path+'chunk'
             chunk_size=100000
             batch_no=1
             for chunk in pd.read_csv(file,chunksize=chunk_size):
                 chunk.to_csv(path_name+str(batch_no)+'.csv',index=False)
                 batch_no+=1
```

it take sometime to split large dataset into small ones, be patient, if you want to checkout where the smaller datasets are, go to ./data_temperature folder

```
In [14]: split_large_data_csv("./data_temperature/",temperature_data)
```

```
In [15]: df2 = pd.read_csv('./data_temperature/chunk1.csv')
```

```
In [16]: df2.head()
```

```
Out[16]:            dt  AverageTemperature  AverageTemperatureUncertainty   City  \
         0  1743-11-01               6.068                          1.737  Århus
         1  1743-12-01                 NaN                            NaN  Århus
         2  1744-01-01                 NaN                            NaN  Århus
         3  1744-02-01                 NaN                            NaN  Århus
         4  1744-03-01                 NaN                            NaN  Århus

            Country Latitude Longitude
         0  Denmark   57.05N    10.33E
         1  Denmark   57.05N    10.33E
         2  Denmark   57.05N    10.33E
         3  Denmark   57.05N    10.33E
         4  Denmark   57.05N    10.33E
```

Load only one chunk of the Immergation Data

```
In [26]: # Configure the necessary Spark environment
         import os
         import sys

         pyspark_submit_args = os.environ.get("PYSPARK_SUBMIT_ARGS", "")
         if not "pyspark-shell" in pyspark_submit_args: pyspark_submit_args += " pyspark-shell"
         os.environ["PYSPARK_SUBMIT_ARGS"] = pyspark_submit_args

         spark_home = os.environ.get('SPARK_HOME', None)
         sys.path.insert(0, spark_home + "/python")

         # Add the py4j to the path.
         # You may need to change the version number to match your install
         sys.path.insert(0, os.path.join(spark_home, "python/lib/py4j-0.8.2.1-src.zip"))

         # Initialize PySpark
         exec(open(os.path.join(spark_home, "python/pyspark/shell.py")).read())
```

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.3
      /_/

Using Python version 3.6.3 (default, Dec  9 2017 04:28:46)
SparkSession available as 'spark'.
```

```
In [27]: from pyspark.sql import SparkSession
```

```
In [31]: spark_i94 = SparkSession.builder.appName('Udacity').getOrCreate()
         df_spark_i94=spark_i94.read.csv('./data/chunk1.csv',inferSchema=True,header=True)
         df_spark_i94.printSchema()
```

```
root
 |-- cicid: double (nullable = true)
 |-- i94yr: double (nullable = true)
 |-- i94mon: double (nullable = true)
 |-- i94cit: double (nullable = true)
 |-- i94res: double (nullable = true)
 |-- i94port: string (nullable = true)
 |-- arrdate: double (nullable = true)
 |-- i94mode: double (nullable = true)
 |-- i94addr: string (nullable = true)
 |-- depdate: double (nullable = true)
 |-- i94bir: double (nullable = true)
 |-- i94visa: double (nullable = true)
 |-- count: double (nullable = true)
 |-- dtadfile: integer (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: double (nullable = true)
 |-- dtaddto: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- insnum: string (nullable = true)
 |-- airline: string (nullable = true)
 |-- admnum: double (nullable = true)
 |-- fltno: string (nullable = true)
 |-- visatype: string (nullable = true)
```

Load only one chunk of the Temperature Data

```
In [33]: spark_temperature = SparkSession.builder.appName('Udacity2').getOrCreate()
         df_spark_temperature=spark_temperature.read.csv('./data_temperature/chunk1.csv',inferSc
         df_spark_temperature.printSchema()
```

```
root
 |-- dt: timestamp (nullable = true)
 |-- AverageTemperature: double (nullable = true)
 |-- AverageTemperatureUncertainty: double (nullable = true)
 |-- City: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- Latitude: string (nullable = true)
 |-- Longitude: string (nullable = true)
```

1.0.4 Step 2: Explore and Assess the Data

Explore the Data Identify data quality issues, like missing values, duplicate data, etc.

```
In [67]: re_obj = re.compile(r'\'(.*)\'.*\'(.*)\'')
         i94portvalid = {}
         with open('valid.txt') as f:
              for data in f:
                  match = re_obj.search(data)
                  i94portvalid[match[1]]=[match[2]]
```

Cleaning Steps

Clean immigration data

```
In [82]: def clean_immigration_data1(file):
             spark1 = SparkSession.builder.appName('Udacity').getOrCreate()
             df=spark1.read.csv(file,inferSchema=True,header=True)
             df = df.filter(df.i94port.isin(list(i94portvalid.keys())))
             return df
```

```
In [83]: df_clean = clean_immigration_data1('./data/chunk1.csv')
```

```
In [85]: df_clean.show(5)
```

```
+-----+------+------+------+------+-------+-------+-------+-------+-------+------+-------+-----+
|cicid| i94yr|i94mon|i94cit|i94res|i94port|arrdate|i94mode|i94addr|depdate|i94bir|i94visa|count|
+-----+------+------+------+------+-------+-------+-------+-------+-------+------+-------+-----+
|  7.0|2016.0|   4.0| 254.0| 276.0|    ATL|20551.0|    1.0|     AL|   null|  25.0|    3.0|  1.0|
| 15.0|2016.0|   4.0| 101.0| 101.0|    WAS|20545.0|    1.0|     MI|20691.0|  55.0|    2.0|  1.0|
| 16.0|2016.0|   4.0| 101.0| 101.0|    NYC|20545.0|    1.0|     MA|20567.0|  28.0|    2.0|  1.0|
| 17.0|2016.0|   4.0| 101.0| 101.0|    NYC|20545.0|    1.0|     MA|20567.0|   4.0|    2.0|  1.0|
| 18.0|2016.0|   4.0| 101.0| 101.0|    NYC|20545.0|    1.0|     MI|20555.0|  57.0|    1.0|  1.0|
+-----+------+------+------+------+-------+-------+-------+-------+-------+------+-------+-----+
only showing top 5 rows
```

```
In [86]: df_clean.schema
```

```
Out[86]: StructType(List(StructField(cicid,DoubleType,true),StructField(i94yr,DoubleType,true),S
```

Clean temperature data

```
In [69]: df2 = spark.read.format("csv").option("header", "true").load(temperature_data)
```

```
In [70]: df2 = df2.filter(df2.AverageTemperature != 'NaN')
```

```
In [71]: @udf()
         def get_i94port(city):
             for key in i94portvalid:
                 if city.lower() in i94portvalid[key][0].lower():
                     return key
```

```
In [72]: df2 = df2.withColumn("i94port", get_i94port(df2.City))
```

```
In [73]: df2.first()
```

```
Out[73]: Row(dt='1743-11-01', AverageTemperature='6.068', AverageTemperatureUncertainty='1.73699
```

```
In [74]: df2.show(5)
```

```
+----------+------------------+-----------------------------+-----+-------+--------+---------+--
|        dt|AverageTemperature|AverageTemperatureUncertainty| City|Country|Latitude|Longitude|i9
+----------+------------------+-----------------------------+-----+-------+--------+---------+--
|1743-11-01|             6.068|           1.7369999999999999|Århus|Denmark|  57.05N|   10.33E|
|1744-04-01|5.7879999999999985|           3.6239999999999997|Århus|Denmark|  57.05N|   10.33E|
|1744-05-01|            10.644|           1.2830000000000001|Århus|Denmark|  57.05N|   10.33E|
|1744-06-01|14.050999999999998|                        1.347|Århus|Denmark|  57.05N|   10.33E|
|1744-07-01|            16.082|                        1.396|Århus|Denmark|  57.05N|   10.33E|
+----------+------------------+-----------------------------+-----+-------+--------+---------+--
only showing top 5 rows
```

filter the value not null

```
In [76]: df2 = df2.filter(df2.i94port != 'null')
```

```
In [77]: df2.show(5)
```

```
+----------+------------------+-----------------------------+--------+--------------+--------+--
|        dt|AverageTemperature|AverageTemperatureUncertainty|    City|       Country|Latitude|Lo
+----------+------------------+-----------------------------+--------+--------------+--------+--
|1743-11-01|             8.758|                        1.886|Aberdeen|United Kingdom|  57.05N|
|1744-04-01|6.0699999999999985|           2.9339999999999997|Aberdeen|United Kingdom|  57.05N|
|1744-05-01|             7.751|                        1.494|Aberdeen|United Kingdom|  57.05N|
|1744-06-01|             10.62|                        1.574|Aberdeen|United Kingdom|  57.05N|
|1744-07-01|             12.35|                        1.591|Aberdeen|United Kingdom|  57.05N|
+----------+------------------+-----------------------------+--------+--------------+--------+--
only showing top 5 rows
```

```
In [78]: def clean_temperature_data1(file):
             df2 = spark.read.format("csv").option("header", "true").load(file)
             df2 = df2.filter(df2.AverageTemperature != 'NaN')
             df2 = df2.withColumn("i94port", get_i94port(df2.City))
             df2 = df2.filter(df2.i94port != 'null')
             return df2
```

```
In [79]: df2_clean=clean_temperature_data1('./data_temperature/chunk1.csv')
```

```
In [80]: df2_clean.show()
```

```
+----------+------------------+-----------------------------+--------+--------------+--------+--
|        dt|AverageTemperature|AverageTemperatureUncertainty|    City|       Country|Latitude|Lo
+----------+------------------+-----------------------------+--------+--------------+--------+--
|1743-11-01|             8.758|                        1.886|Aberdeen|United Kingdom|  57.05N|
|1744-04-01| 6.069999999999999|                        2.934|Aberdeen|United Kingdom|  57.05N|
|1744-05-01|             7.751|                        1.494|Aberdeen|United Kingdom|  57.05N|
|1744-06-01|             10.62|                        1.574|Aberdeen|United Kingdom|  57.05N|
|1744-07-01|             12.35|                        1.591|Aberdeen|United Kingdom|  57.05N|
|1744-09-01|            11.224|                        1.606|Aberdeen|United Kingdom|  57.05N|
|1744-10-01|             8.945|                        1.731|Aberdeen|United Kingdom|  57.05N|
|1744-11-01|             7.836|                        1.585|Aberdeen|United Kingdom|  57.05N|
|1744-12-01|             5.263|                        1.871|Aberdeen|United Kingdom|  57.05N|
|1745-01-01|             4.136|                        1.825|Aberdeen|United Kingdom|  57.05N|
|1745-02-01|             2.436|           1.6769999999999998|Aberdeen|United Kingdom|  57.05N|
|1745-03-01|              3.24|                        1.456|Aberdeen|United Kingdom|  57.05N|
|1745-04-01|             4.819|                        1.429|Aberdeen|United Kingdom|  57.05N|
|1750-01-01|             5.441|                        1.463|Aberdeen|United Kingdom|  57.05N|
|1750-02-01|              7.31|                        2.474|Aberdeen|United Kingdom|  57.05N|
|1750-03-01|             7.335|                        3.024|Aberdeen|United Kingdom|  57.05N|
|1750-04-01|             6.604|                        1.308|Aberdeen|United Kingdom|  57.05N|
|1750-05-01| 8.328000000000001|                        1.509|Aberdeen|United Kingdom|  57.05N|
|1750-06-01|10.802999999999999|                        1.393|Aberdeen|United Kingdom|  57.05N|
|1750-07-01|            14.367|                          1.4|Aberdeen|United Kingdom|  57.05N|
+----------+------------------+-----------------------------+--------+--------------+--------+--
only showing top 20 rows
```

```
In [81]: df2_clean.schema

Out[81]: StructType(List(StructField(dt,StringType,true),StructField(AverageTemperature,StringTy
```

Clean multiple data chunks under data folder for immigration data

```
In [89]: import functools
         def unionAll(dfs):
             return functools.reduce(lambda df1,df2: df1.union(df2.select(df1.columns)), dfs)

In [90]: def clean_large_data_sas(path):
             path_name=path+'chunk'
             csv_chunks=fileCount("./data/","csv")
             chunk_size=2#csv_chunks
             batch_no=1
             schemas=clean_immigration_data1(path_name+str(batch_no)+'.csv').schema
             df1 = spark.createDataFrame([], schemas)

             for chunk in range(chunk_size):
                 file=path_name+str(batch_no)+'.csv'
                 df_immigration_chunk= clean_immigration_data1(file)
                 df1 = unionAll([df1,df_immigration_chunk])
                 batch_no+=1
             return df1

In [91]: df_immigration_all=clean_large_data_sas("./data/")

In [92]: df_immigration_all.show(5)

+-----+------+------+------+------+-------+-------+-------+-------+-------+------+-------+-----+
|cicid| i94yr|i94mon|i94cit|i94res|i94port|arrdate|i94mode|i94addr|depdate|i94bir|i94visa|count|
+-----+------+------+------+------+-------+-------+-------+-------+-------+------+-------+-----+
|  7.0|2016.0|   4.0| 254.0| 276.0|    ATL|20551.0|    1.0|     AL|   null|  25.0|    3.0|  1.0|
| 15.0|2016.0|   4.0| 101.0| 101.0|    WAS|20545.0|    1.0|     MI|20691.0|  55.0|    2.0|  1.0|
| 16.0|2016.0|   4.0| 101.0| 101.0|    NYC|20545.0|    1.0|     MA|20567.0|  28.0|    2.0|  1.0|
| 17.0|2016.0|   4.0| 101.0| 101.0|    NYC|20545.0|    1.0|     MA|20567.0|   4.0|    2.0|  1.0|
| 18.0|2016.0|   4.0| 101.0| 101.0|    NYC|20545.0|    1.0|     MI|20555.0|  57.0|    1.0|  1.0|
+-----+------+------+------+------+-------+-------+-------+-------+-------+------+-------+-----+
only showing top 5 rows
```

Clean multiple data chunks under data_temperature folder for temperature data

```
In [93]: def clean_large_data_csv(path):
             path_name=path+'chunk'
```

```
    csv_chunks=fileCount("./data_temperature/","csv")
    chunk_size=2
    batch_no=1
    schemas=clean_temperature_data1(path_name+str(batch_no)+'.csv').schema
    df1 = spark.createDataFrame([], schemas)
    for chunk in range(chunk_size):
        file=path_name+str(batch_no)+'.csv'
        df_temperature_data1=clean_temperature_data1(file)
        df1 = unionAll([df1,df_temperature_data1])
        batch_no+=1
    return df1
```

```
In [94]: df_temperature_all=clean_large_data_csv("./data_temperature/")
```

```
In [95]: df_temperature_all.show(5)
```

```
+----------+------------------+-----------------------------+--------+--------------+--------+--
|        dt|AverageTemperature|AverageTemperatureUncertainty|    City|       Country|Latitude|Lo
+----------+------------------+-----------------------------+--------+--------------+--------+--
|1743-11-01|             8.758|                        1.886|Aberdeen|United Kingdom|  57.05N|
|1744-04-01| 6.069999999999999|                        2.934|Aberdeen|United Kingdom|  57.05N|
|1744-05-01|             7.751|                        1.494|Aberdeen|United Kingdom|  57.05N|
|1744-06-01|             10.62|                        1.574|Aberdeen|United Kingdom|  57.05N|
|1744-07-01|             12.35|                        1.591|Aberdeen|United Kingdom|  57.05N|
+----------+------------------+-----------------------------+--------+--------------+--------+--
only showing top 5 rows
```

1.0.5 Step 3: Define the Data Model

3.1 Conceptual Data Model joined table: immigration data joined with city temperature data

3.2 Mapping Out Data Pipelines

1. get the full data from the path '../../data/18-83510-I94-Data-2016/i94_jun16_sub.sas7bdat' and '../../data2/GlobalLandTemperaturesByCity.csv'
2. created folder data and data_temperature, use the folder path to save the chunked data from above.
3. data cleansing, clean the invalid data, data with nulls and data match to che i94port
4. extract the columns which are useful for analysis
5. create result table by merging immigration and temperature tables

1.0.6 Step 4: Run Pipelines to Model the Data

4.1 Create the data model Build the data pipelines to create the data model.

```
In [98]: fname = '../../data/18-83510-I94-Data-2016/i94_apr16_sub.sas7bdat'
```

```
In [99]: split_large_data_sas("./data/",fname)
In [100]: df_immigration_all=clean_large_data_sas("./data/")
```

```
In [103]: # Extract columns
          immigration_table = df_immigration_all.select(["i94yr", "i94mon", "i94cit", "i94port",
```

```
In [104]: temperature_data = '../../data2/GlobalLandTemperaturesByCity.csv'
          split_large_data_csv("./data_temperature/",temperature_data)
```

```
In [105]: df_temperature_data1=clean_large_data_csv("./data_temperature/")
```

```
In [106]: temp_table = df_temperature_data1.select(["AverageTemperature", "City", "Country", "La
```

```
In [108]: #  create table of results of the immigration and temperature data
          df_immigration_all.createOrReplaceTempView("immigration_view")
          df_temperature_data1.createOrReplaceTempView("temperature_view")
```

```
In [109]: # 9 merge two tables
          result_table = spark.sql('''
          select immigration_view.i94yr as year,
                 immigration_view.i94mon as month,
                 immigration_view.i94cit as city,
                 immigration_view.i94port as i94port,
                 immigration_view.arrdate as arrival_date,
                 immigration_view.depdate as departure_date,
                 immigration_view.i94visa as reason,
                 temperature_view.AverageTemperature as temperature,
                 temperature_view.Latitude as latitude,
                 temperature_view.Longitude as longitude
          from immigration_view
          JOIN temperature_view ON (immigration_view.i94port = temperature_view.i94port)
          ''')
```

```
In [110]: n = 2
          path="./results/result.parquet"
          spark_df  = result_table.repartition(n)
          spark_df.write.mode("overwrite").partitionBy("i94port").parquet(path)
```

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 [111]: def quality_check(df, description):
              result = df.count()
              if result == 0:
                  print("Data quality check failed for {} with zero records".format(description)
              else:
                  print("Data quality check passed for {} with {} records".format(description, r
              return 0
```

```
In [113]: quality_check(df_immigration_all, "immigration table")
          quality_check(df_temperature_data1, "temperature table")
```

```
Data quality check passed for immigration table with 199796 records
Data quality check passed for temperature table with 20341 records
```

```
Out[113]: 0
```

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.

result_table - Columns: - i94yr : 4 digit year, - i94mon : numeric month, - i94cit : 3 digit code of origin city, - i94port : 3 character code of destination USA city, - arrdate : arrival date in the USA, - i94mode : 1 digit travel code, - depdate : departure date from the USA, - i94visa : reason for immigration, - AverageTemperature : average temperature of destination city,

Step 5: Complete Project Write Up

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

- We should split the large scale datasets in chunks, and we should process with apache spark for better performance

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

- The data was increased by 100x.: We can use cloud services for example AWS, with the redshift, it is an analystial database can optimized for heavy workloads.

- The data populates a dashboard that must be updated on a daily basis by 7am every day.: Use Ariflow, which can create daily quality check and send email if the operation failed and freeze the dashboards

- The database needed to be accessed by 100+ people.: use Amazon Redshift, it can add nodes to data warehouse and enable performance on data warehouse

```
In [ ]:
```