**Big Data Capstone Project : Health care Data Analysis**

**Project Flow:**

1. **City\_report\_capstone\_project:**

**Data Ingestion:**

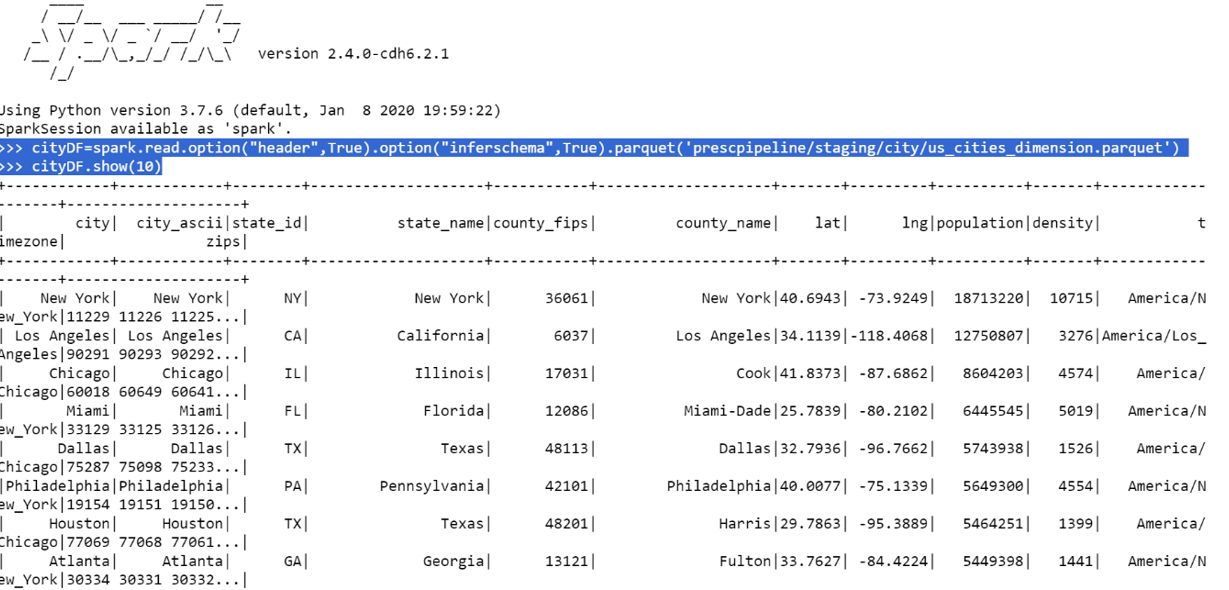
**Step 1:** Sample data files are provided and they have to be ingested in to HDFS in to the project’s location

**hdfs dfs -mkdir -p prescpipeline/staging/city**

**hdfs dfs -put us\_cities\_dimension.parquet prescpipeline/staging/city**

**cityDF=spark.read.option("header",True).option("inferschema",True).parquet('prescpipeline/staging/city/us\_cities\_dimension.parquet')**

**cityDF.show(10)**



**cityDF.printSchema()**

**cityDF.count()**

A white background with black and white clouds

Description automatically generated

**Data Cleansing/Preprocessing:**

**step2:**  cleanse the data and use only the data that is required for analysis/reports using pyspark

**Selecting the required columns**

**cityDF1=cityDF.select('city','state\_id','state\_name','county\_name','population','zips')**

**cityDF1.show(4)**

**cityDF1.printSchema()**

A screenshot of a computer

Description automatically generated

**Converting the required fields to upper case**

import pyspark.sql.functions as F

cityDF2=cityDF1.select(F.upper(cityDF1.city).alias("city"),'state\_id',F.upper(cityDF1.state\_name).alias("state\_name"),F.upper(cityDF1.county\_name).alias("county\_name"),'population','zips')

cityDF2.show(3)

A screenshot of a computer

Description automatically generated

**Step 3 :Transform Logic: City Report**

**Calculate the Number of zips in each city**

**from pyspark.sql.functions import col, lit**

**from pyspark.sql.functions import split, explode , size**

**split\_col=split(cityDF1['ZIPS'],' ')**

**df\_split = cityDF1.withColumn('Zip\_Split\_cnt',size(split\_col))**

**df\_split.createOrReplaceTempView("City\_split\_df")**

A screenshot of a computer

Description automatically generated

**Calculate the number of distinct Prescribers assigned for each City.**

**prescriber1\_df = spark.sql("SELECT count(distinct npi) as Prescriber\_Counts,np\_city from prescriber group by np\_city").show()**

A screen shot of a computer

Description automatically generated

**Calculate total total\_claim\_count prescribed for each city.**

**prescriber1\_df = spark.sql("SELECT sum(tot\_claim\_cnt) as total\_claim\_counts, np\_state from prescriber group by np\_state") .show()**



**Do not report a city in the final report if no prescriber is assigned to it.**

prescriber1\_df.createOrReplaceTempView("prescriber1")

df\_split.createOrReplaceTempView("City\_split\_df")

spark.sql( "SELECT CITY as City\_Name,STATE\_NAME as State\_Name,COUNTY\_NAME as County\_Name ,POPULATION as City\_Population,Zip\_Split\_cnt as Number\_of\_Zips,p.Prescriber\_Counts,p.total\_claim\_counts FROM City\_split\_df c inner join prescriber1 p on upper(c.CITY) = p.np\_city and c.STATE\_ID = p.np\_state").show()

A screenshot of a computer

Description automatically generated

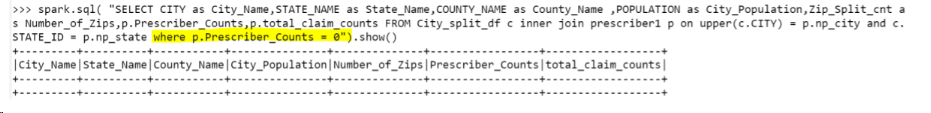
spark.sql( "SELECT CITY as City\_Name,STATE\_NAME as State\_Name,COUNTY\_NAME as County\_Name ,POPULATION as City\_Population,Zip\_Split\_cnt as Number\_of\_Zips,p.Prescriber\_Counts,p.total\_claim\_counts FROM City\_split\_df c inner join prescriber1 p on upper(c.CITY) = p.np\_city and c.STATE\_ID = p.np\_state where p.Prescriber\_Counts != 0").show(),

A screenshot of a computer

Description automatically generated

**Checking if any city having no prescriber :**

spark.sql( "SELECT CITY as City\_Name,STATE\_NAME as State\_Name,COUNTY\_NAME as County\_Name ,POPULATION as City\_Population,Zip\_Split\_cnt as Number\_of\_Zips,p.Prescriber\_Counts,p.total\_claim\_counts FROM City\_split\_df c inner join prescriber1 p on upper(c.CITY) = p.np\_city and c.STATE\_ID = p.np\_state where p.Prescriber\_Counts = 0").show()



**Writing the City Report outputs in to Hive/HDFS:**

from pyspark.sql import SparkSession

spark=SparkSession.builder.appName("SparkHive").enableHiveSupport().config("spark.sql.warehouse.dir","/user/hive/warehouse").getOrCreate()

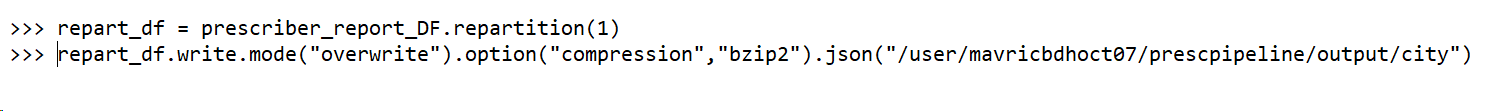
spark.sparkContext.setLogLevel("ERROR")

City\_report\_df = spark.sql( "SELECT CITY as City\_Name,STATE\_NAME as State\_Name,COUNTY\_NAME as County\_Name ,POPULATION as City\_Population,Zip\_Split\_cnt as Number\_of\_Zips,p.Prescriber\_Counts,p.total\_claim\_counts FROM City\_split\_df c inner join prescriber1 p on upper(c.CITY) = p.np\_city and c.STATE\_ID = p.np\_state where p.Prescriber\_Counts != 0")

**No of splits: 1**

repart\_df = prescriber\_report\_DF.repartition(1)

repart\_df.write.mode("overwrite").option("compression","bzip2").json("/user/mavricbdhoct07/prescpipeline/output/city")



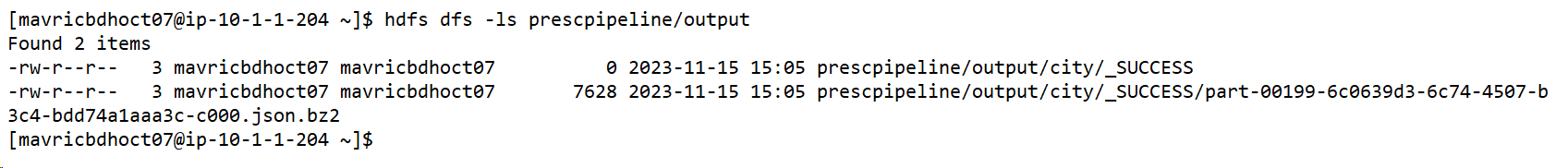
**City output In HDFS – 1 Split showing**

**Output City Report Layout**

**No of splits: 1**

**Output format: JSON**

**Compression : Bzip2**



[mavricbdhoct07@ip-10-1-1-204 ~]$ hdfs dfs -ls prescpipeline/output Found 2 items

-rw-r--r-- 3 mavricbdhoct07 mavricbdhoct07 0 2023-11-15 15:05 prescpipeline/output/city/\_SUCCESS

-rw-r--r-- 3 mavricbdhoct07 mavricbdhoct07 7628 2023-11-15 15:05 prescpipeline/output/city/\_SUCCESS/part-00199-6c0639d3-6c74-4507-b3c4-bdd74a1aaa3c-c000.json.bz2

**City output In hive :**

City\_report\_df.write.mode("overwrite").saveAsTable("City\_Final\_Report")

**desc Final\_City\_Report;**

A screenshot of a computer

Description automatically generated

**Select \* from City\_Final\_Report limit 10;**

A screenshot of a computer

Description automatically generated

**select count(\*) from City\_Final\_Report;**

