**CAPSTONE PROJECT**

**Project: Health care Data Analysis**

**CITI Report:**

**Data Ingestion:**

**Step1:**

hdfs dfs -mkdir -p prescpipeline/staging/city

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

A yellow and black text

Description automatically generated

**Step2:**

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

cityDF.printSchema()

A close-up of a computer screen

Description automatically generated

cityDF.show(10)

A screenshot of a computer

Description automatically generated

**cityDF.count()**

A screenshot of a computer

Description automatically generated

**Data Cleansing/Preprocessing:**

**Step1: 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 code

Description automatically generated

**Step2: 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 yellow line with black text

Description automatically generated

**Transform Logic: City Report**

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

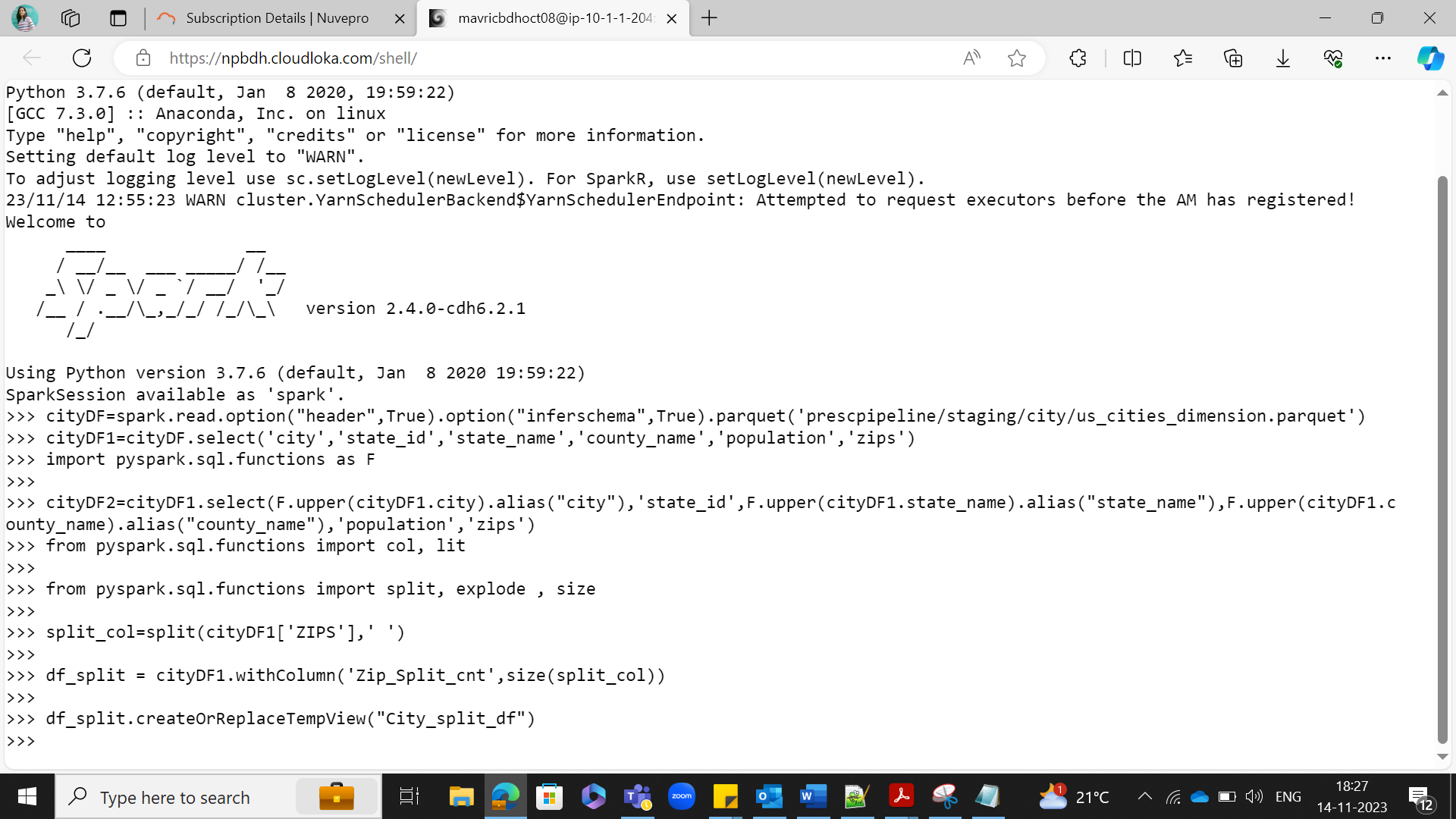
from pyspark.sql.functions import col, lit

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")



**2.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 screenshot of a computer

Description automatically generated

**3.Calculate 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()

A screenshot of a computer

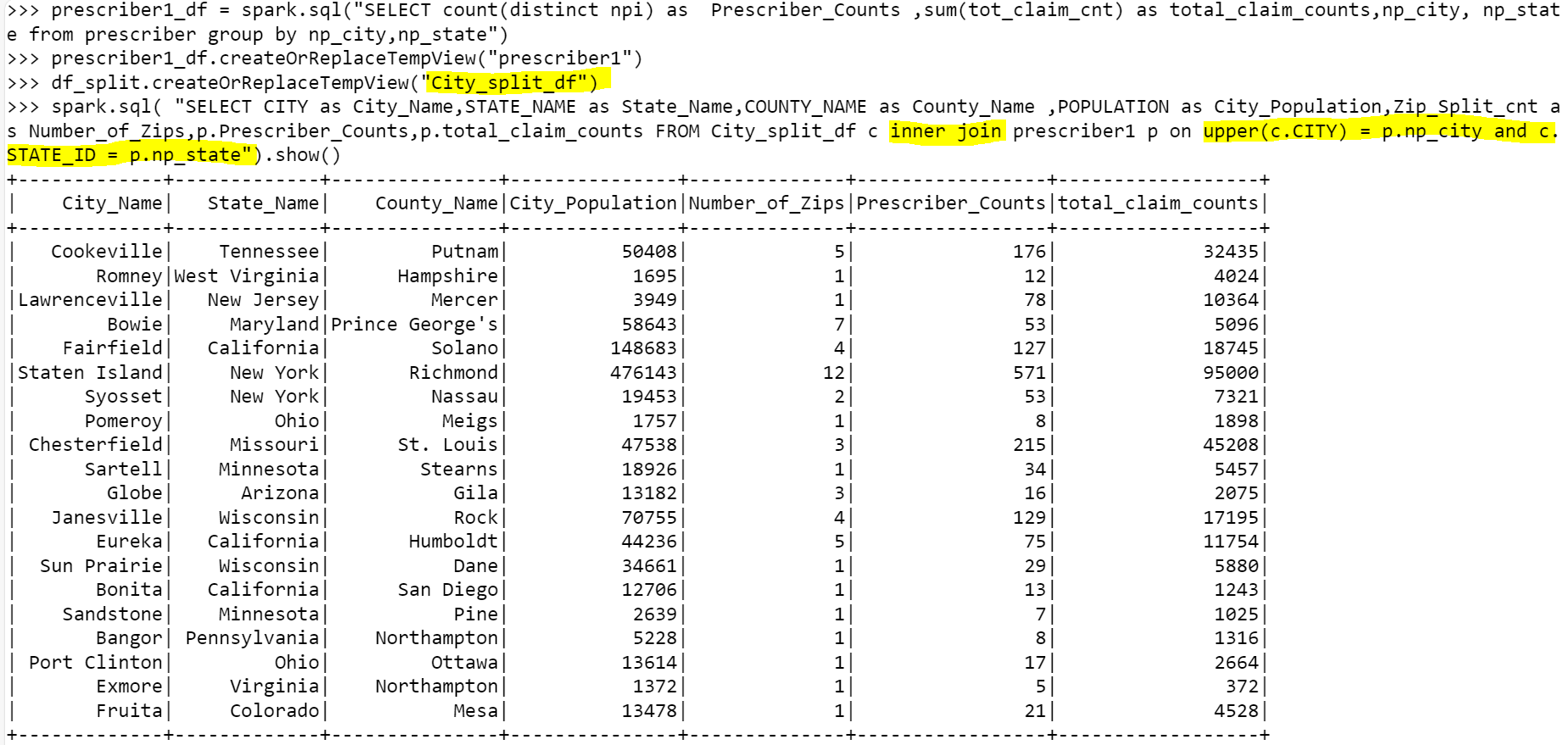
Description automatically generated

**4.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()**



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

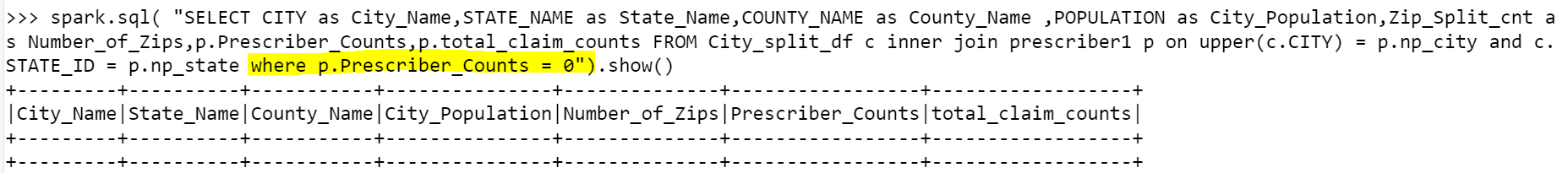
Description automatically generated

**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 a

s 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")

City\_report\_df.write.mode("overwrite").option("compression","bzip2").json("/user/mavricbdhoct08/prescpipeline/output/city")

A close-up of a text

Description automatically generated

**City output In HDFS**

hdfs dfs -ls /user/mavricbdhoct08/prescpipeline/output/city

[mavricbdhoct08@ip-10-1-1-204 ~]$ hdfs dfs -ls /user/mavricbdhoct08/prescpipeline/output/city

Found 201 items

-rw-r--r-- 3 mavricbdhoct08 mavricbdhoct08 0 2023-11-14 13:22 /user/mavricbdhoct08/prescpipeline/output/city/\_SUCCESS

-rw-r--r-- 3 mavricbdhoct08 mavricbdhoct08 1231 2023-11-14 13:22 /user/mavricbdhoct08/prescpipeline/output/city/part-00000-f8a0de71

-6359-41dc-a76b-167998807997-c000.json.bz2

A yellow mark on a white background

Description automatically generated

**City output In hive :**

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

**desc Final\_City\_Report;**

A close up of a text

Description automatically generated

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

A screenshot of a computer

Description automatically generated**select count(\*) from City\_Final\_Report;**

A close-up of a number

Description automatically generated