**CAPSTONE PROJECT**

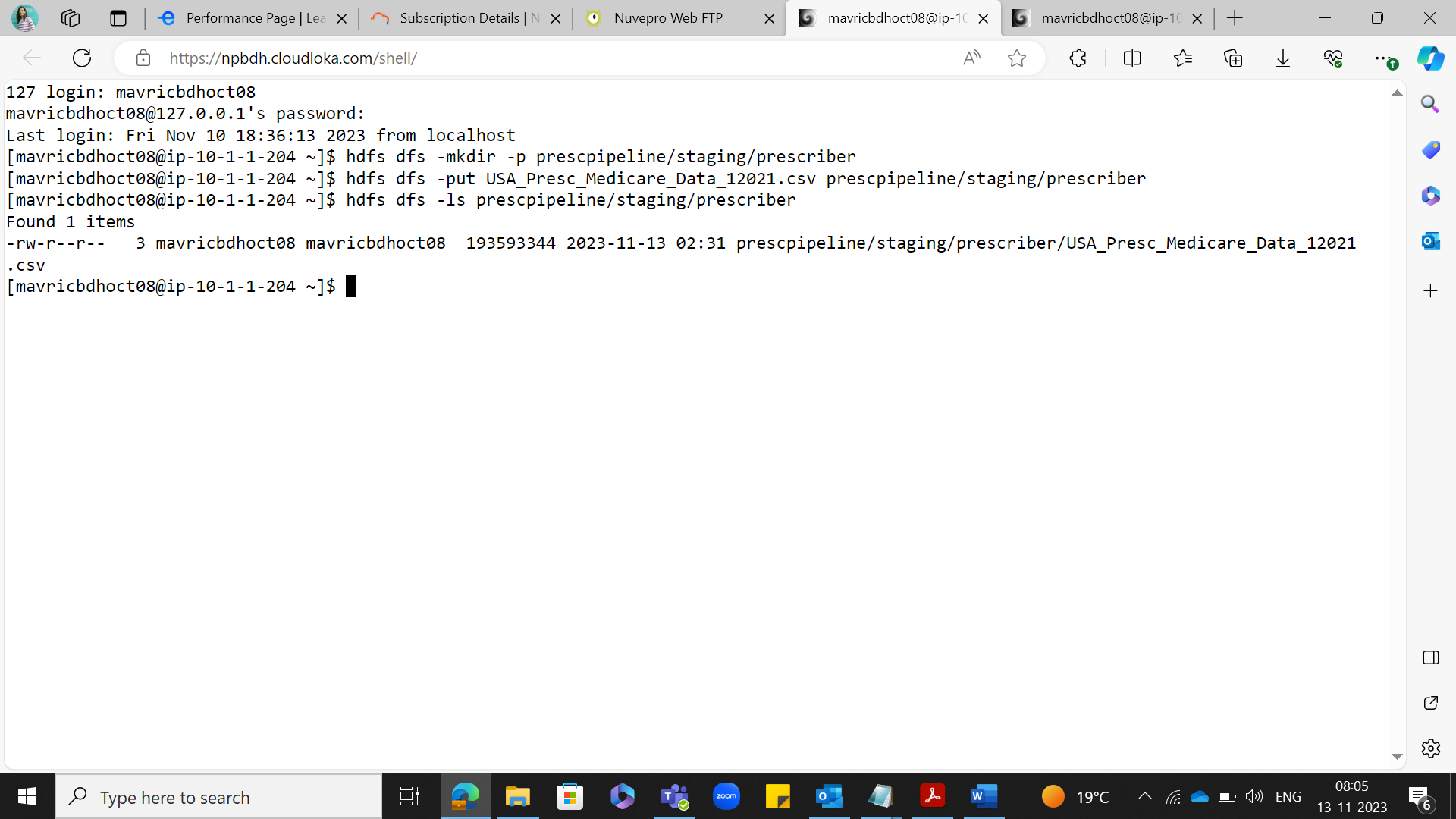
**Project: Health care Data Analysis**

**Data Ingestion: Prescriber Report**

**step1:**

hdfs dfs -mkdir -p prescpipeline/staging/prescriber

hdfs dfs -put USA\_Presc\_Medicare\_Data\_12021.csv prescpipeline/staging/prescriber



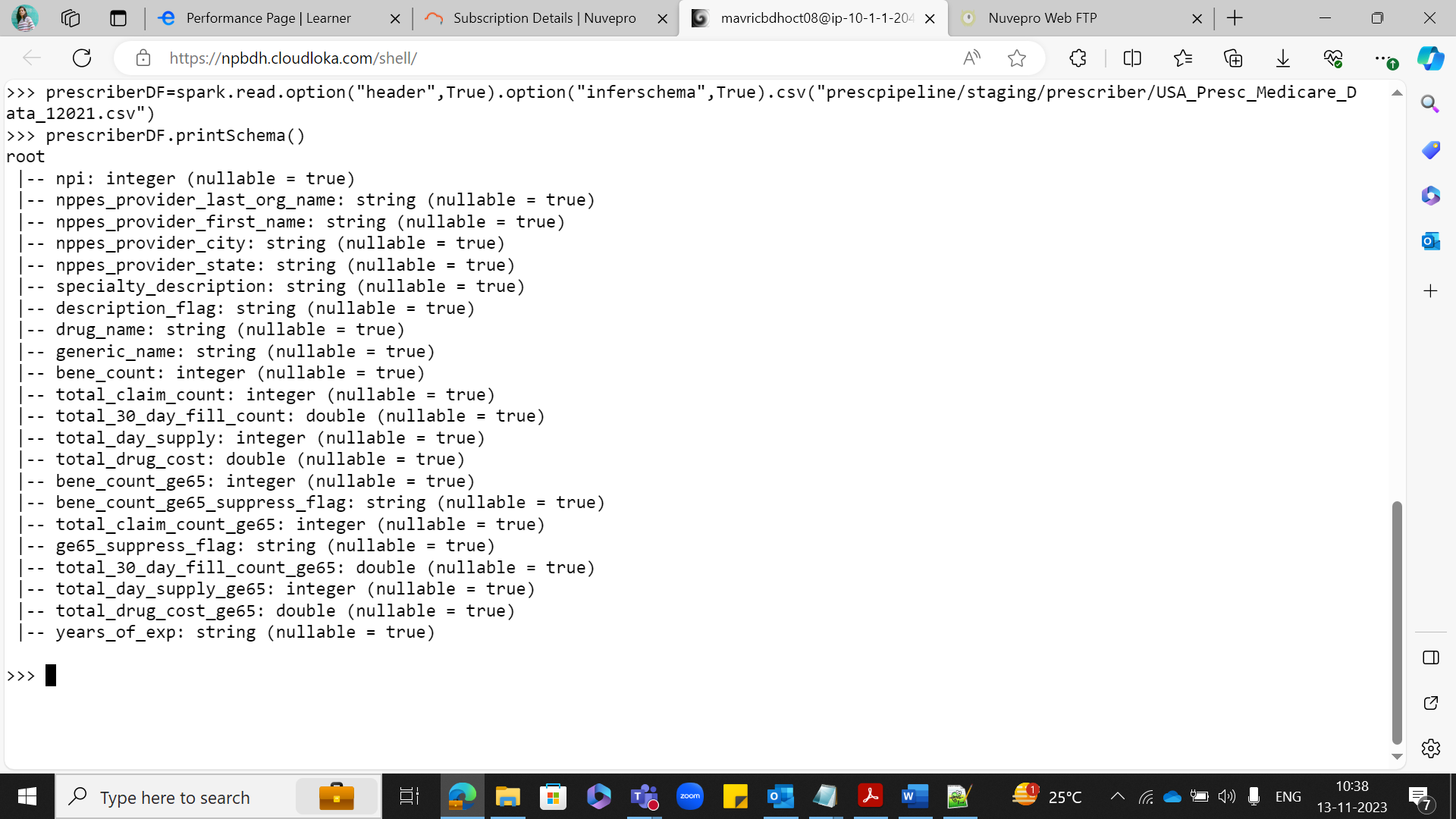
Hdfs dfs -cat prescpipeline/staging/prescriber/USA\_Presc\_Medicare\_Data\_12021.csv | wc -l

A screenshot of a computer

Description automatically generated

**step2:** prescriberDF=spark.read.option("header",True).option("inferschema",True).csv("prescpipeline/staging/prescriber/USA\_Presc\_Medicare\_Data\_12021.csv")

prescriberDF.printSchema()



prescriberDF.show(10)

A screenshot of a computer

Description automatically generated

prescriberDF.count()

A screenshot of a computer

Description automatically generated

**Data Cleansing/Preprocessing:**

**Step1:** Selecting the required columns:

pres\_step1\_df=prescriberDF.select('npi','nppes\_provider\_last\_org\_name','nppes\_provider\_first\_name','nppes\_provider\_city','nppes\_provider\_state','specialty\_description','drug\_name','total\_claim\_count','total\_day\_supply','total\_drug\_cost','years\_of\_exp')

A screenshot of a computer

Description automatically generated

**Step2:** **Rename the above fields to shorter names:**

prescriberDF1=prescriberDF.select('npi',(prescriberDF.nppes\_provider\_last\_org\_name).alias ('np\_Lname'),(prescriberDF.nppes\_provider\_first\_name).alias ('np\_Fname'),(prescriberDF.nppes\_provider\_city).alias ('np\_city'),(prescriberDF.nppes\_provider\_state).alias ('np\_state'),(prescriberDF.specialty\_description).alias ('spcl\_desc'),(prescriberDF.drug\_name).alias ('drug\_nm'),(prescriberDF.total\_claim\_count).alias ('tot\_claim\_cnt'),(prescriberDF.total\_day\_supply).alias ('tot\_day\_suply'),(prescriberDF.total\_drug\_cost) .alias ('tot\_drug\_cst'), (prescriberDF.years\_of\_exp) .alias ('yrs\_of\_exp'))

prescriberDF1.printSchema()

A screenshot of a computer

Description automatically generated

**step3:** **Add a Country Field 'USA' to the above data**

from pyspark.sql.functions import col, lit

prescriberDF1=prescriberDF1.withColumn("Country", lit("USA"))

prescriberDF1.printSchema()

A screenshot of a computer

Description automatically generated

prescriberDF1.show(3)

A screenshot of a computer

Description automatically generated

**Step 4: Clean the “years\_of\_exp” to extract only the numbers.**

import pyspark.sql.functions as f

pres\_step1\_df=pres\_step1\_df.withColumn("yrs\_of\_exp",f.regexp\_extract("yrs\_of\_exp", r'([0-9]+)', 1))

A screenshot of a computer

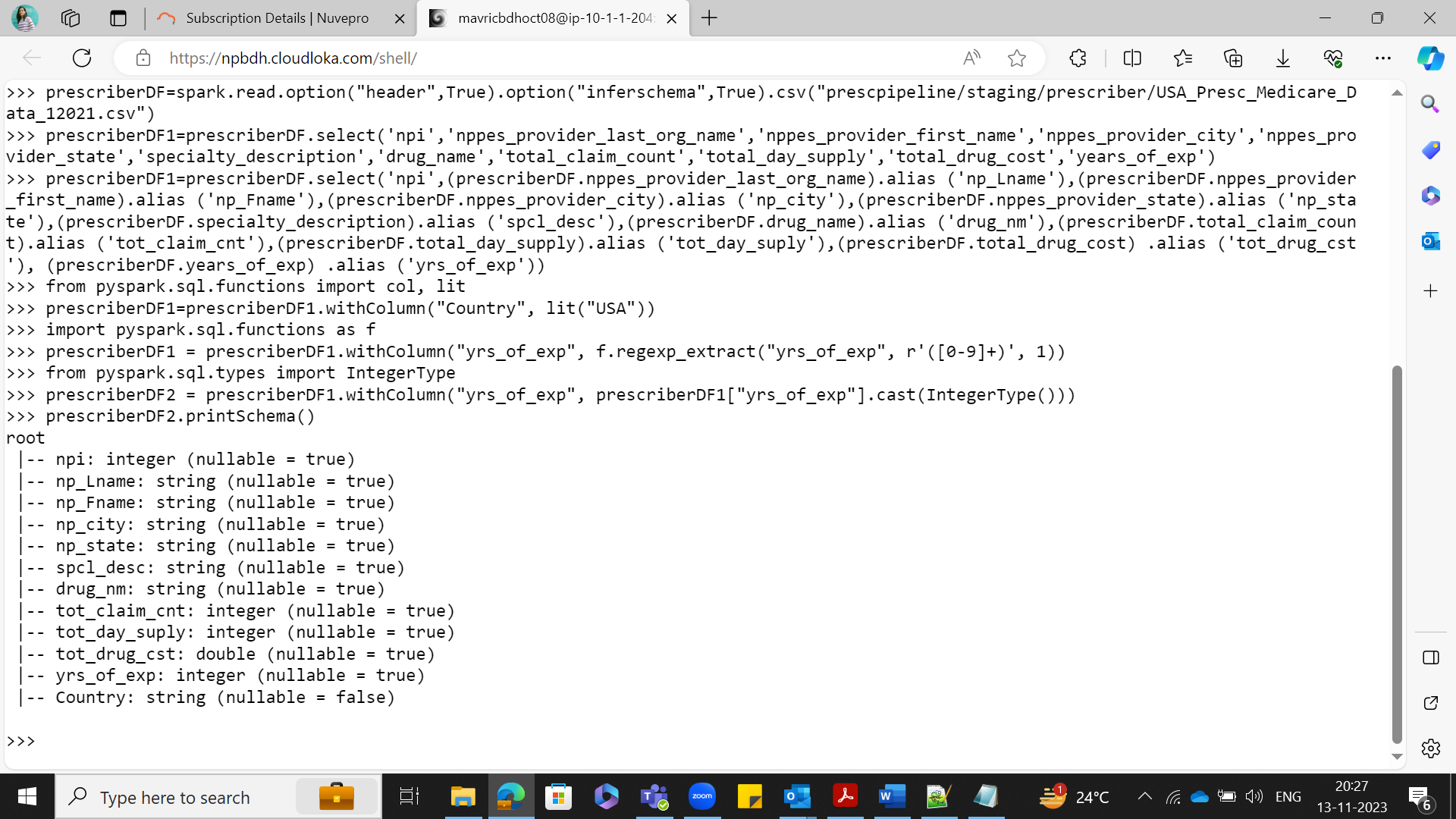
Description automatically generated

**Step5:Convert the years\_of\_exp field to integer**

from pyspark.sql.types import IntegerType

prescriberDF2 = prescriberDF1.withColumn("yrs\_of\_exp", prescriberDF1["yrs\_of\_exp"].cast(IntegerType()))

prescriberDF2.printSchema()



**step6:Combine First Name and Last Name in to a single field and remove the individual columns**

from pyspark.sql.functions import concat, concat\_ws,col, lit

prescriberDF2 = prescriberDF2.withColumn("presc\_fullname",concat\_ws(" ", "np\_Fname", "np\_Lname"))

prescriberDF2.show(5)

A screenshot of a computer

Description automatically generated

**Removing individual columns**

prescriberDF2 = prescriberDF2.drop("np\_Fname", "np\_Lname")

prescriberDF2.show(3)

A screenshot of a computer

Description automatically generated

**Step7:Count the number of null values for each column**

from pyspark.sql.functions import col,isnan, when, count

prescriberDF2.select([count(when(isnan(c) | col(c).isNull(),c)).alias(c) for c in prescriberDF2.columns]).show()

A screenshot of a computer

Description automatically generated

**Step8: Clean all the Null/Nan Values**

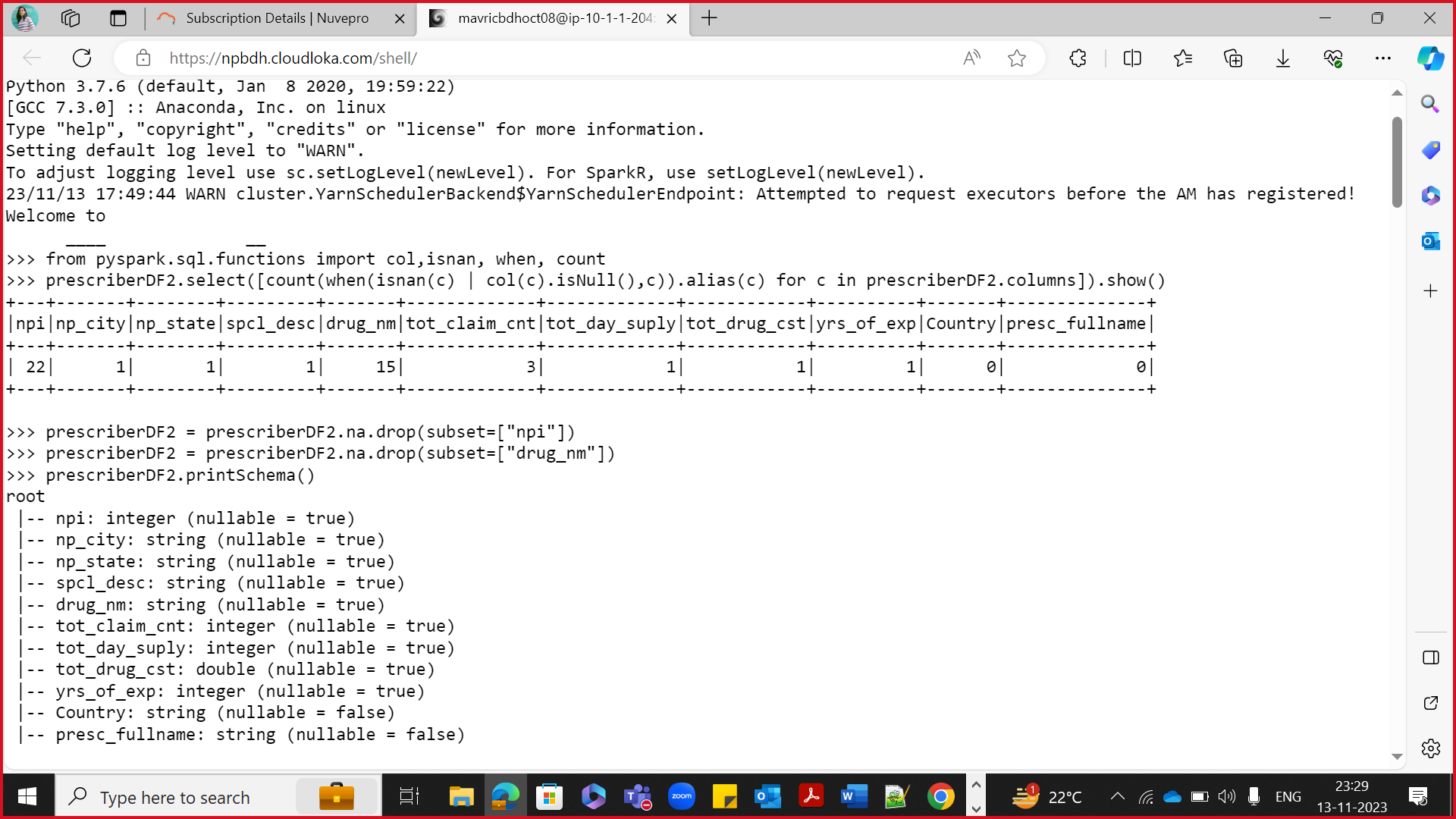
Delete the records where the PRESC\_ID and DRUG\_NAME is fields are NULL. Use dropna() of dataframe

prescriberDF2 = prescriberDF2.na.drop(subset=["npi"])

prescriberDF2 = prescriberDF2.na.drop(subset=["drug\_nm"])

prescriberDF2.printSchema()

prescriberDF2.select([count(f.when(isnan(c) | col(c).isNull(),c)).alias(c) for c in prescriberDF1.columns]).show()



A close-up of a computer code

Description automatically generated

**Transformations:**

**Step 1 :Transformation**

**Transform logic: Prescriber Report:**

Top 5 Prescribers with highest total\_claim\_count per each state.

Consider the prescribers only from 20 to 50 years of experience.

from pyspark.sql import functions as F

prescriberDF2 = prescriberDF2.filter(F.col("yrs\_of\_exp").between(20,50))

prescriberDF2.createOrReplaceTempView("prescriber")

spark.sql("SELECT min(yrs\_of\_exp) , max(yrs\_of\_exp) from prescriber").show()

A white background with black text

Description automatically generated

from pyspark.sql import Window

from pyspark.sql.functions import dense\_rank

from pyspark.sql.functions import concat, concat\_ws,col, lit

window = Window.partitionBy("np\_state").orderBy(col("tot\_claim\_cnt").desc())

prescriberDF2 = prescriberDF2.withColumn("tot\_claim\_cnt\_rank", dense\_rank().over(window))

prescriberDF2.createOrReplaceTempView("prescriber")

A screenshot of a computer code

Description automatically generated

spark.sql("SELECT npi as Prescriber\_ID , presc\_fullname as Prescriber\_Full\_Name, np\_state as Prescriber\_State,Country as Prescriber\_Country, yrs\_of\_exp as Prescriber\_Years\_of\_Experience,tot\_claim\_cnt as Total\_claim\_count,tot\_day\_suply as Total\_Days\_Supply,tot\_drug\_cst as Total\_Drug\_Cost from prescriber where tot\_claim\_cnt\_rank <= 5").show()

A black and white text

Description automatically generated with medium confidence

\*\*\*\*Run in HDFS\*\*\*

hdfs dfs -mkdir -p prescpipeline/output/prescriber

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

**Writing the output into HDFS in ORC format and snappy compression :**

from pyspark.sql import SparkSession

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

spark.sparkContext.setLogLevel("ERROR")

prescriber\_report\_DF = spark.sql("SELECT npi as Prescriber\_ID , presc\_fullname as Prescriber\_Full\_Name, np\_state as Prescriber\_State,Country as Prescriber\_Country, yrs\_of\_exp as Prescriber\_Years\_of\_Experience,tot\_claim\_cnt as Total\_claim\_count,tot\_day\_suply as Total\_Days\_Supply,tot\_drug\_cst as Total\_Drug\_Cost from prescriber where tot\_claim\_cnt\_rank <= 5").show()

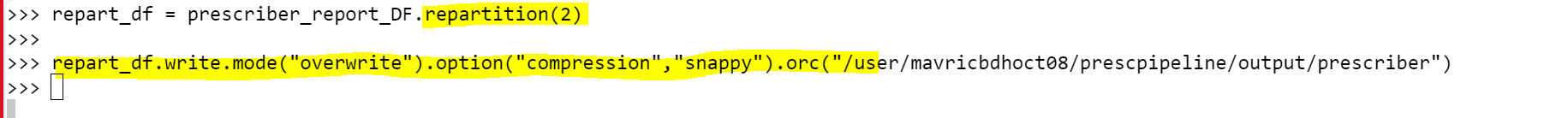
A screenshot of a computer

Description automatically generated

No of splits: 2

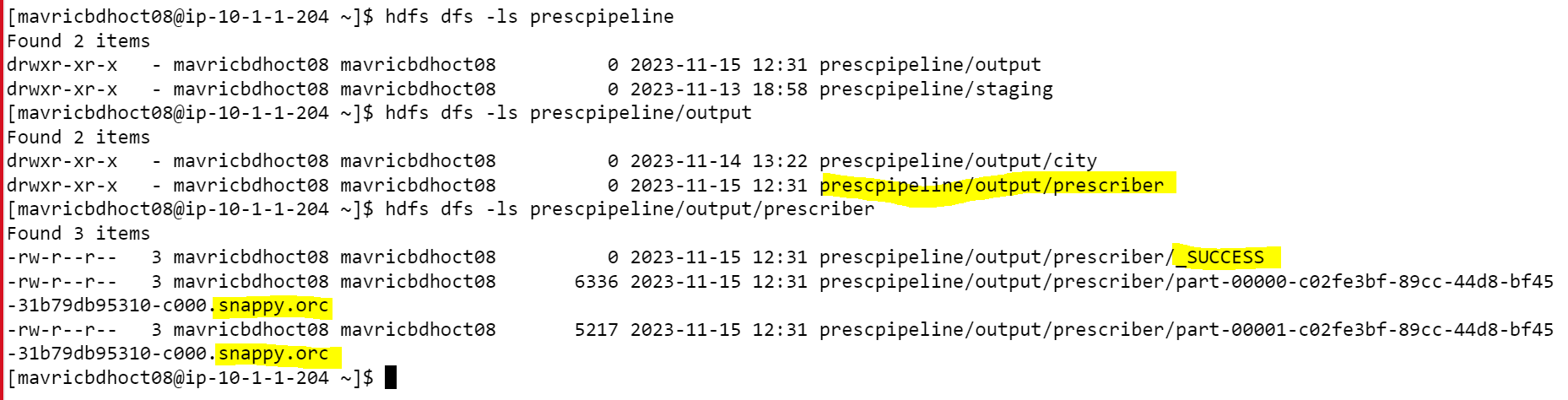
repart\_df = prescriber\_report\_DF.repartition(2)

repart\_df.write.mode("overwrite").option("compression","snappy").orc("/user/mavricbdhoct08/prescpipeline/output/prescriber")



**checking the output report file generated in hdfs:**

hdfs dfs -ls /user/mavericbdhoct08/prescpipeline/output/prescriber



**OUTPUT: HDFS**

**Output Prescriber report layout**

No of splits: 2

Output format: orc

Compression : snappy

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

Found 3 items

-rw-r--r-- 3 mavricbdhoct08 mavricbdhoct08 0 2023-11-15 12:31 prescpipeline/output/prescriber/\_SUCCESS

-rw-r--r-- 3 mavricbdhoct08 mavricbdhoct08 6336 2023-11-15 12:31 prescpipeline/output/prescriber/part-00000-c02fe3bf-89cc-44d8-bf45

-31b79db95310-c000.snappy.orc

-rw-r--r-- 3 mavricbdhoct08 mavricbdhoct08 5217 2023-11-15 12:31 prescpipeline/output/prescriber/part-00001-c02fe3bf-89cc-44d8-bf45

-31b79db95310-c000.snappy.orc

[mavricbdhoct08@ip-10-1-1-204 ~]$

**OUTPUT: HIVE Beeline**

prescriber\_report\_DF.write.partitionBy("Total\_claim\_count").mode("overwrite").saveAsTable("indhudb.prescriber\_Final\_report")

use indhudb;

show tables;

A close up of a text

Description automatically generated

select \* from prescriber\_Final\_report order by Total\_claim\_count desc ;

A screenshot of a computer

Description automatically generated

