**TASK REPORT**

**PRACTICE CASE 8**

**BIG DATA TOOLS: INTEGRATING THE DATA TO TACKLE THE BIG DATA PHENOMENON IN WONDERMART**



By:

Muhammad Ridho Aulia

Student number

Data Engineer Student

Data Fellowship

2020

**Chapter 1**

**Introduction**



You’ve landed a great job with McJager Consulting (MJC) as a data engineer. MJC is proposing a digitalization data project with WonderMart, the world’s largest retail store, to help address their problem of integrating the data to tackle the Big Data phenomenon. Your team of engineers have to create a data pipeline using Sqoop and HiveQL/Impala to store the data.

Your tasks are :

1. Install Cloudera Quickstart VM on your local computer

2. Store the data into HDFS using Cloudera Quickstart VM from your local machine using WinSCP or Tortoise.

3. Using Sqoop to import the data from your VM into HDFS

4. Create a table on HiveQL/Impala to represent the data. All the types of data must be customized based on HiveQL metadata.

5. Using Sqoop incremental method is a plus to inserting the data

6. Describe each step by drawing a flowchart via draw.io

**Chapter 2**

**Progress Report**

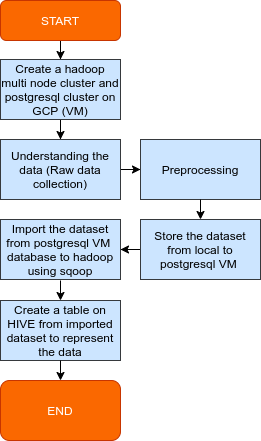
|  |  |  |  |
| --- | --- | --- | --- |
| Day/Date | Task | Level | Comments |
| 12/05/2020 | Create a multinode cluster VM in GCP | Easy | 1 for NameNode, 2 for DataNode |
| 14/05/2020 | Create PostgreSQL Database VM in GCP | Easy | Choose the version you need in marketplace |
| 14/05/2020 | Understanding the data (customer churn dataset) | Easy | Checking the file and structure (There is a duplicate in Id) |
| 18/05/2020 | Import the dataset to  PostgreSQL database | Medium | Using python script to build the engine and connect it to postgresql |
| 18/05/2020 | Import dataset from  PostgreSQL to HDFS | Medium | Using Sqoop Incremental |
| 19/05/2020 | Create table in HIVE  from the data in HDFS Hard | Medium | Adjust the SQL query with HIVE query |
| 20/05/2020 | Create a machine learning model | Hard | Using random forest and xgboost |
| 20/05/2020 | Create the flowchart | Medium | Via draw.io |
| 20/05/2020 | Write the report | Medium | Using WPS |

**Chapter 3**

**Task Report**

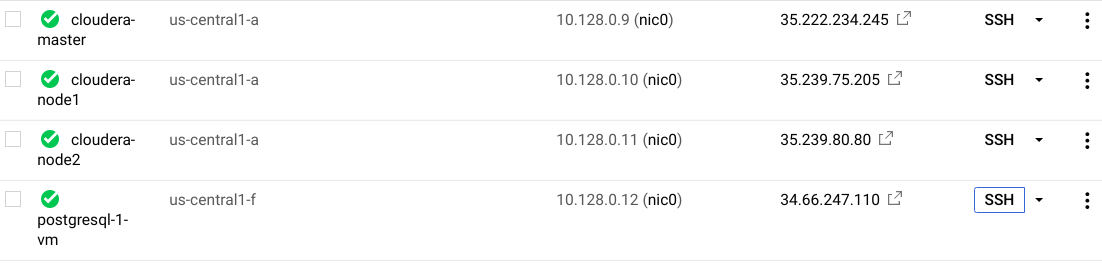
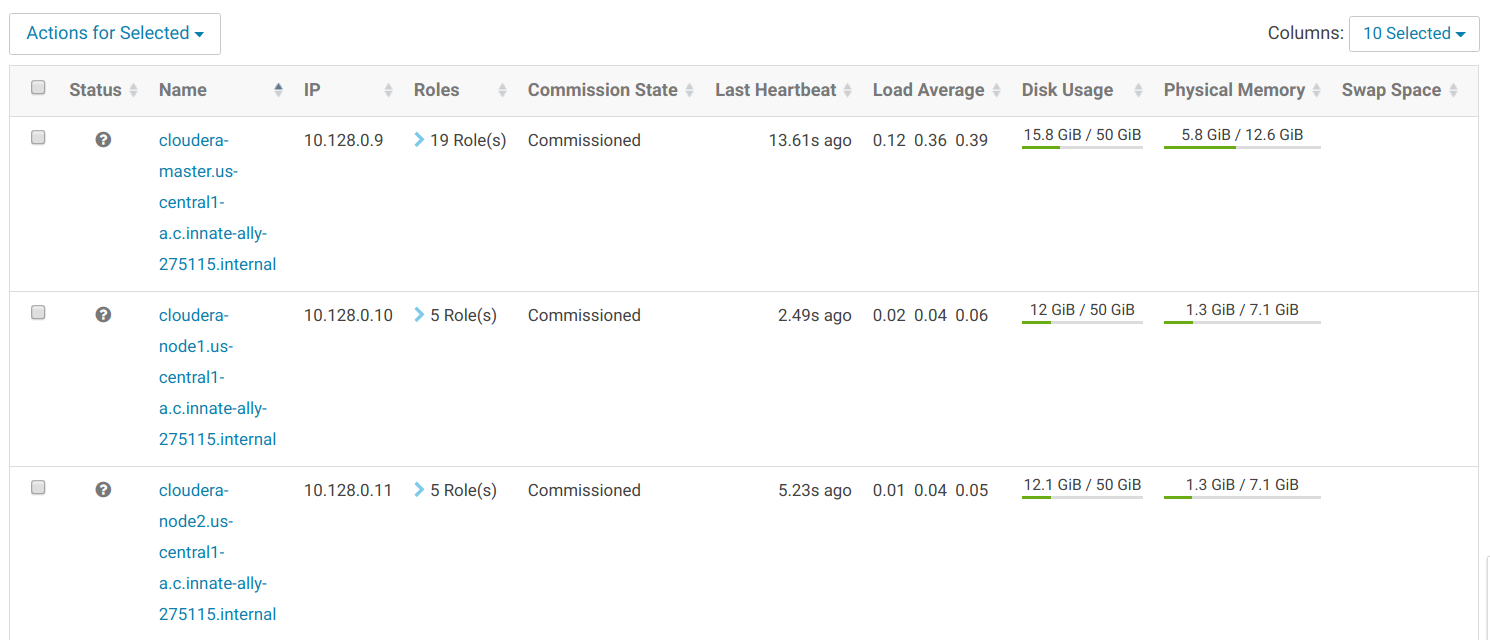
In this project, first, we must understanding the data, we must view the field and column and the purpose of each. Then if the data needs to be cleaned, we did**. In this case because the Id column is set into primary key we must ensure that the id column does not contain duplicate contents. For null contents we will handle it with SQOOP.**

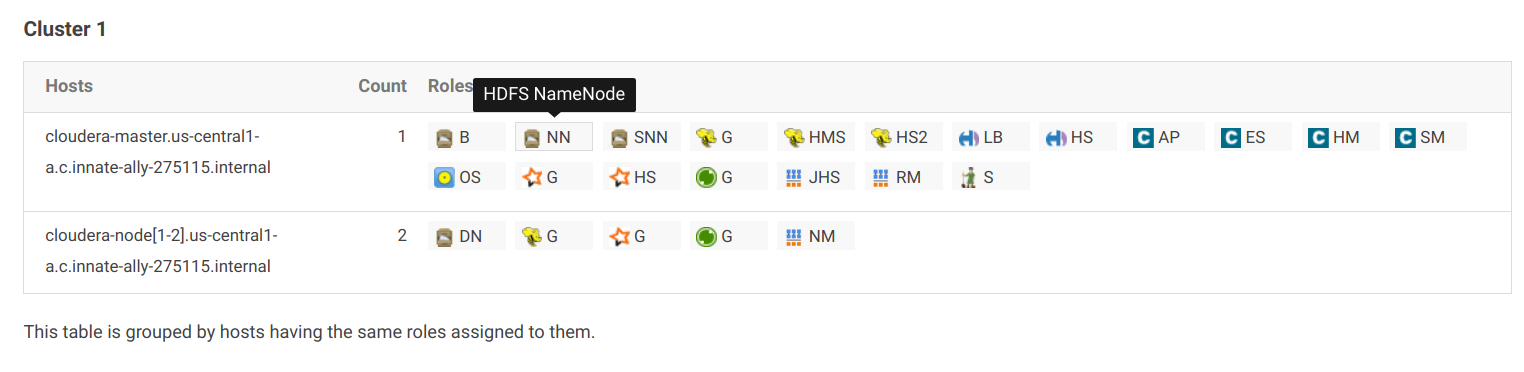
Next, we create a hadoop multinode cluster and postgresql virtual machine on Google Cloud Platform. Then we store the dataset from local to postgresql with some python queries. Next we import the dataset from PostgreSQL VM database to hadoop using sqoop incremental and last we create the table on HIVE from imported dataset to represent the data. All step described in the following flowchart:



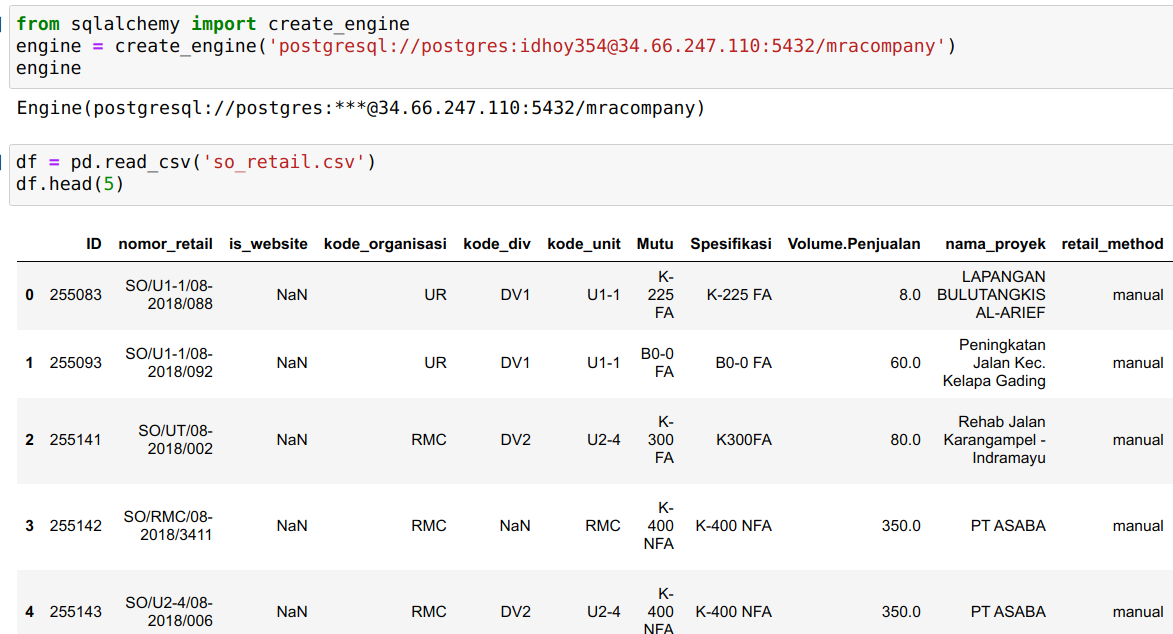
**There is the explanation from the workflow diagram above:**

1. **Create multinode cluster and postgresql VM on Google Cloud Platform**

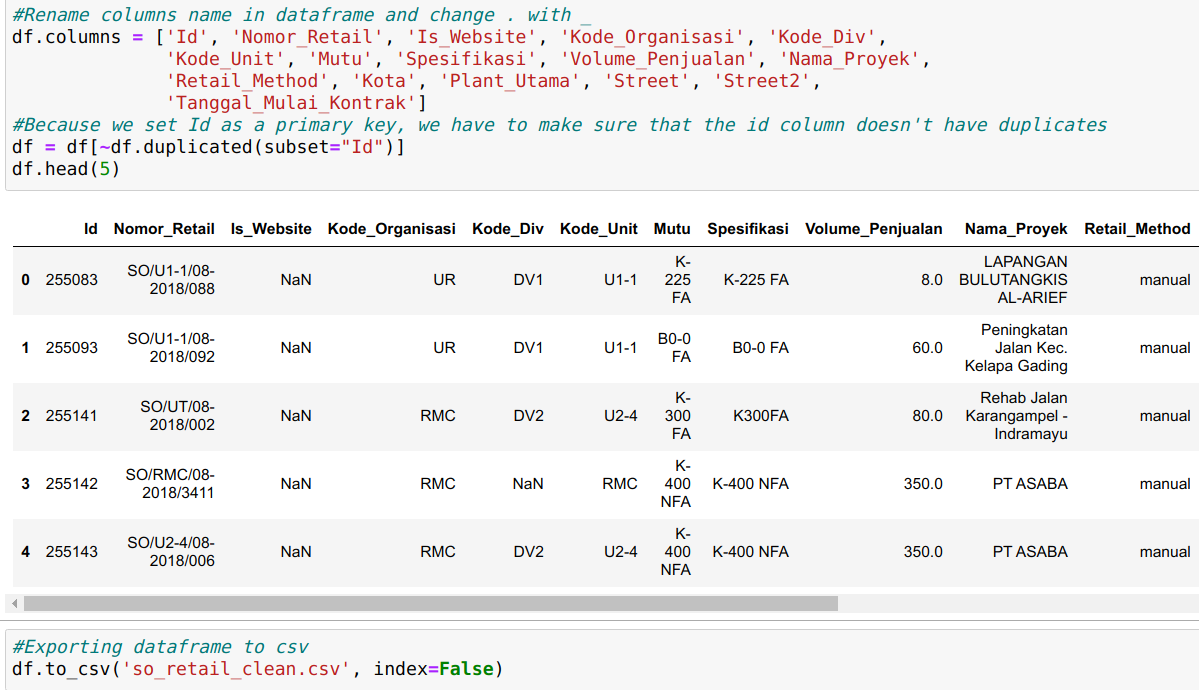
****In Google Cloud Platform, we built 4 Virtual Machines (VM), one is for MasterNode/NameNode, two is for DataNode and last one is for PostgreSQL Database cloud VM. We make 3 nodes because we run a parallel processing for Hadoop on Cloudera Manager.

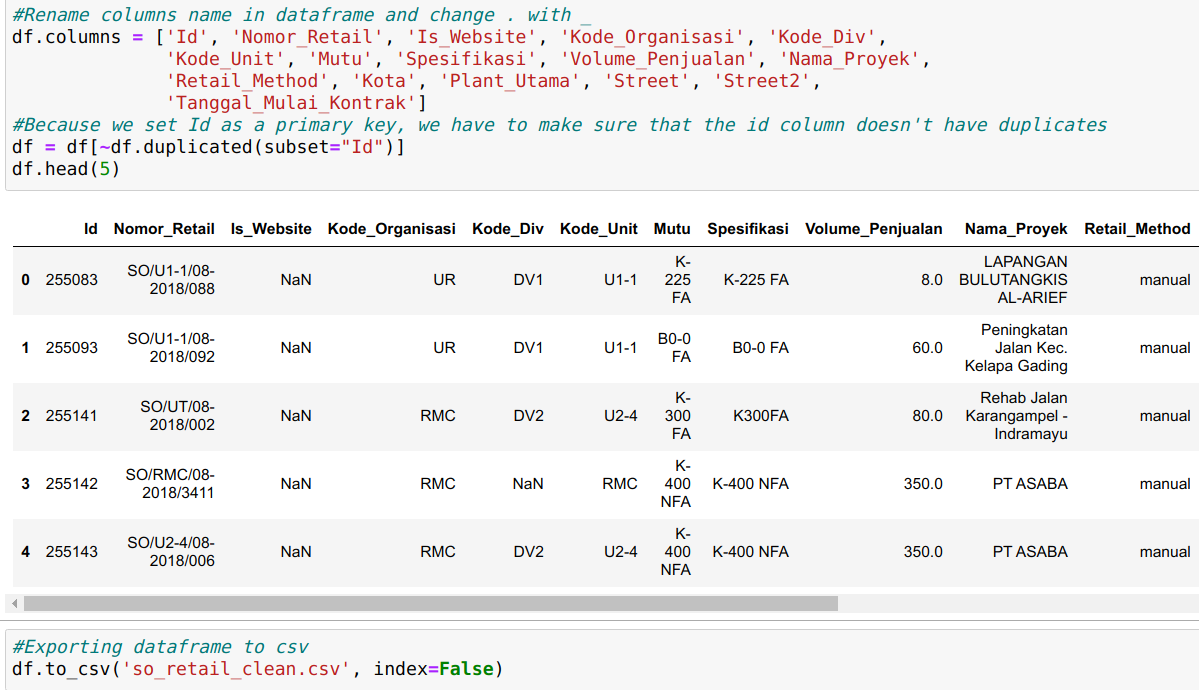
Next we install Cloudera Manager on a NameNode/MasterNode (you can check it via Cloudera Manager in Roles tabs).

1. **Understanding the data and data preprocessing**

In this project the dataset is build in csv format. First we import the dataset (csv) into a local PostgreSQL table using ORM SQLalchemy in python. Then we create the engine for the connection to PostgreSQL and read the dataset using pandas.

**In this case because the Id column is set into primary key we must ensure that the Id column does not contain duplicate contents.** We will dropped the rows containing duplicates**.** We recognize that the columns name on dataset contained non-standard naming ex: Volume.Penjualan and Plant.Utama. We change the column names into capitalize each word and replace the period with underscore. And we export the dataframe to csv file back.



****

1. **Import the dataset from local PostgreSQL database to HDFS using SCOOP**

We build a database using PostgreSQL in this project. Based on the Sqoop book that we read, we decided to use Sqoop Incremental to import the data in PostgreSQL table into HDFS. As we mentioned earlier, we handled the null problems with scoop arguments, we put the antislash and N to transform the null data to make it easier when we do a hive query in the future to see the number of nulls, because if not changed hdfs will represent null as **'null' (string) .**

sqoop import \

--connect jdbc:postgresql://34.66.247.110/mracompany \

--username postgres \

--password idhoy354 \

--table retail \

--incremental append \

--check-column Id \

--last-value 1 \

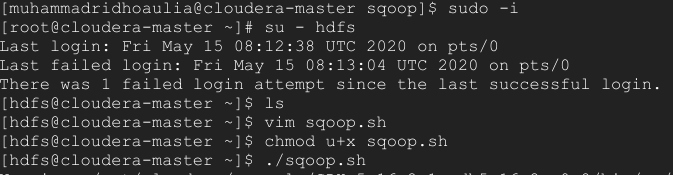
**--null-string '\\N' \**

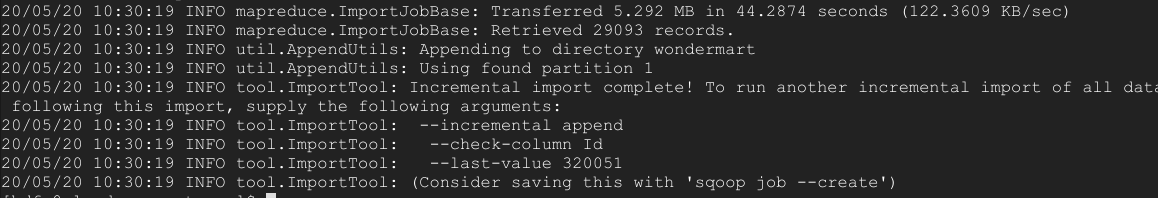
**--null-non-string '\\N' \**

--target-dir /user/hdfs/wondermart \

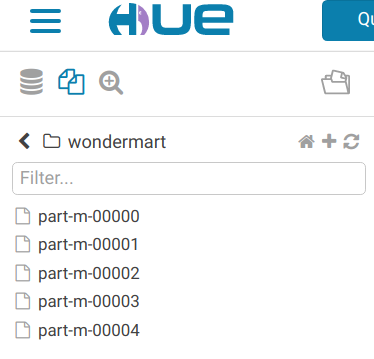
-- --schema public

We put the .sh code to MasterNode in GCP VM, change the file permission and run the .sh file:

If the process is success map reduce will working and parted yhe file into 4 pieces and this the result when we implement the incremental sqoop, they will checking primary key and last value and mark it to facilitate the next import based on the last value, so it will be faster in importing data in the future

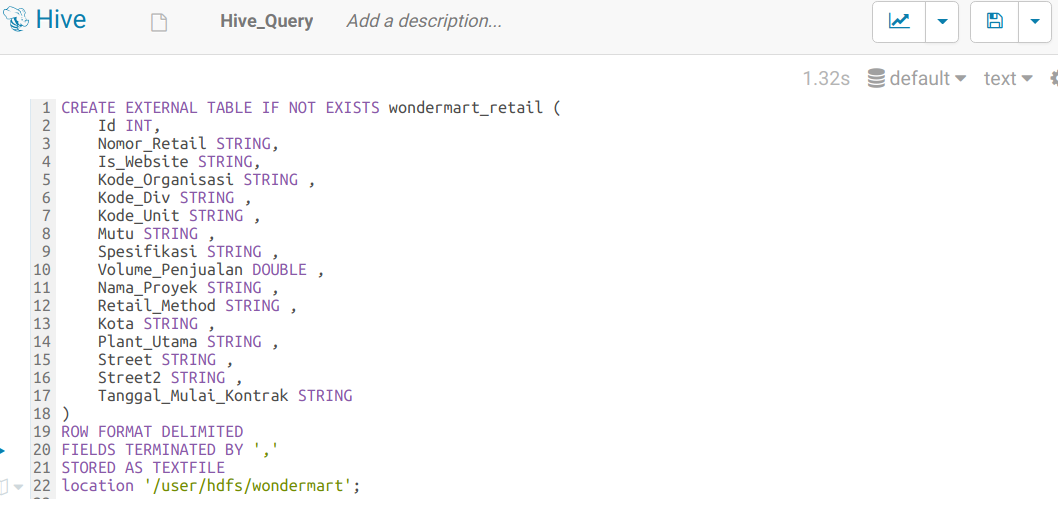
****

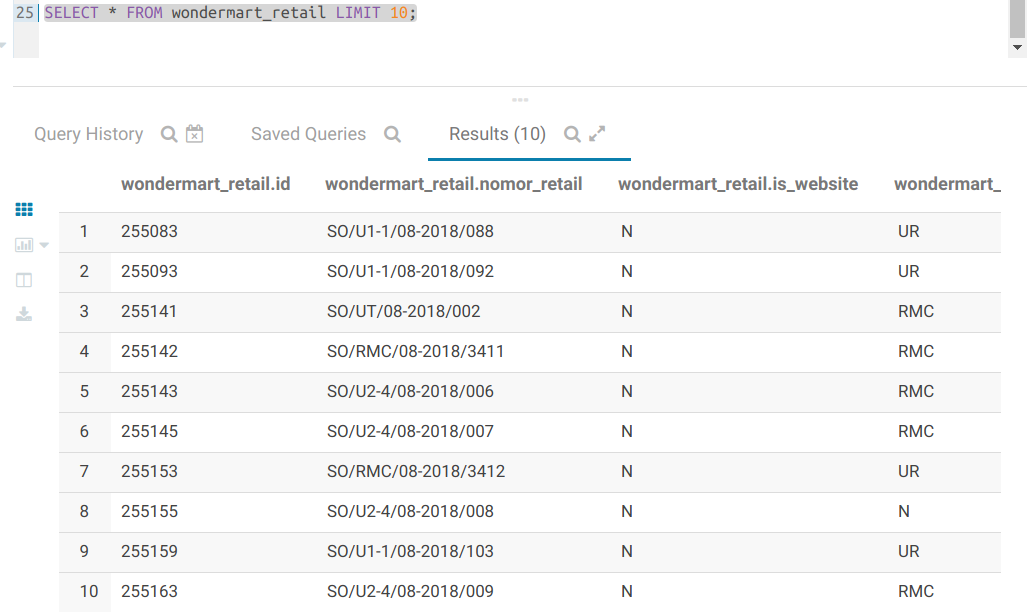
You can see in HUE page at (user/hdfs/wondermart), which there is the dataset file exist.



‘

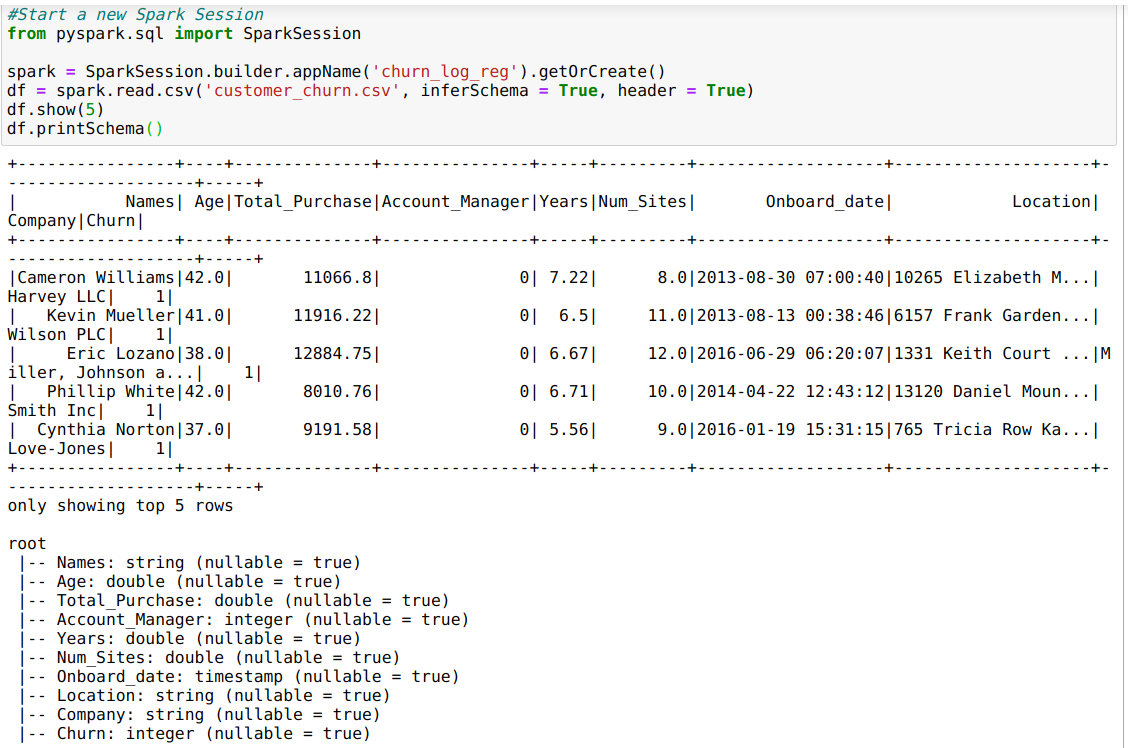
1. **Create a table on HIVE to represent the data**

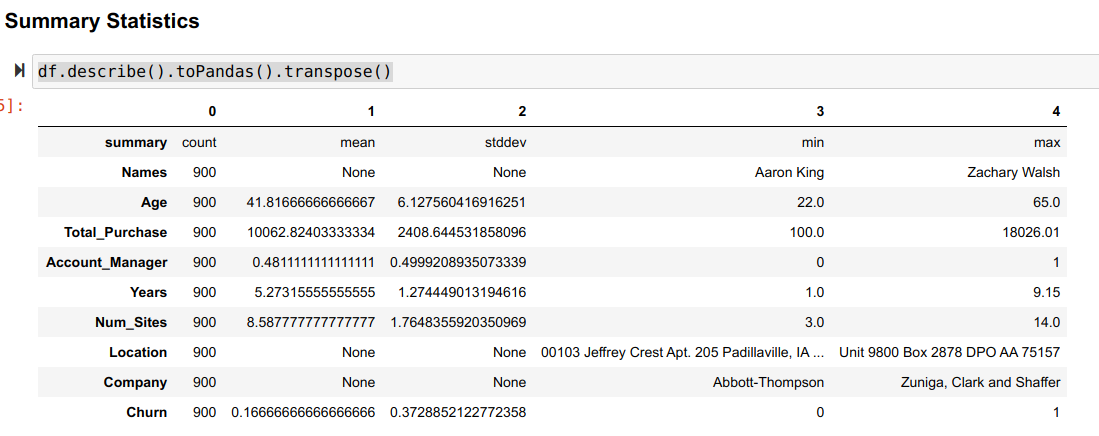
The last steps, is we create a table for data that has been successfully imported into HADOOP. We made the table with HIVE on the HUE page in Cloudera Manager. The HIVE syntax is quite different from other RDBMS like PostgreSQL or MySQL. I f you generate DDL from Dbeaver there is some syntax and data type that you must be customized to HIVE syntax. Below is the syntax:

Next, we try to select the table with HIVE query, there is the result, **you can see the null value handled with N (not empty).**

1. **Create a machine learning model with PySpark/SparkSQL.**

We try to make a machine learning model using Logistic Regression in PySpark. To predict the customer churn or not. First, we install Java Development Kit (JDK 8 or High) and PySpark library. Next we start a new spark session, load the dataset, showing it and print the schema of the dataset.

We can see the summary statistics using describe function:



Next we build the pipeline and use vector assembler to combine the raw features and features generated from various transforms into a single feature vector. In this project we use Logistic Regression to predict customr churn.

from pyspark.ml.feature import VectorAssembler

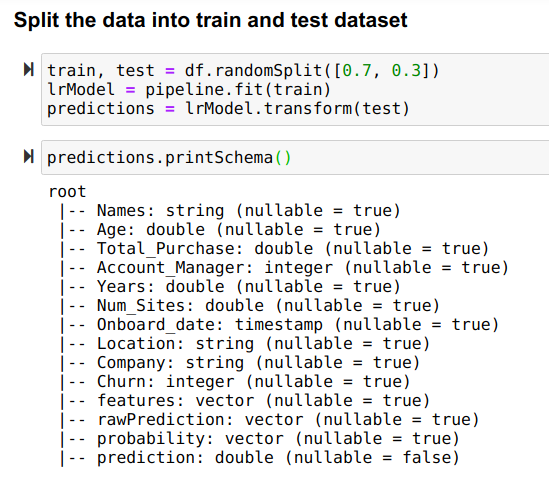
from pyspark.ml import Pipeline

from pyspark.ml.classification import LogisticRegression

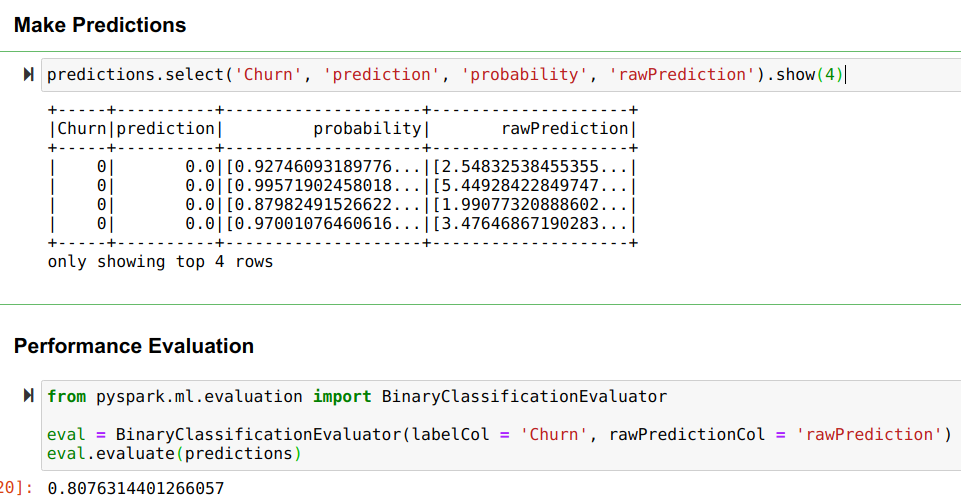
assembler = VectorAssembler(inputCols = ['Age', 'Total\_Purchase', 'Account\_Manager', 'Years', 'Num\_Sites',], outputCol = 'features')

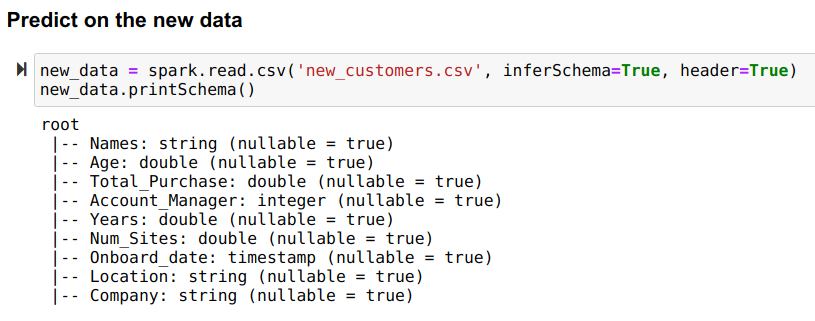
log\_reg = LogisticRegression(featuresCol = 'features', labelCol = 'Churn', maxIter=10)

pipeline = Pipeline(stages = [assembler, log\_reg])

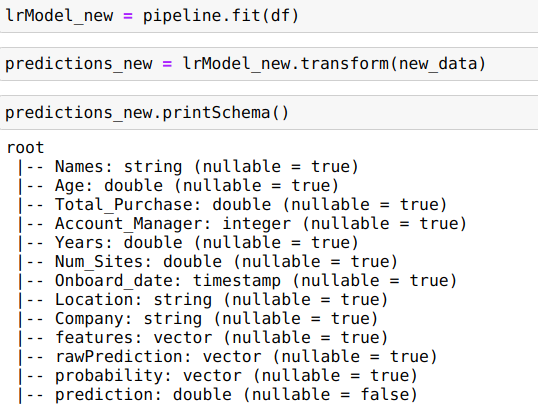
And then we split the dataset into train and test and make the predictions using pipeline.

Next, we make a prediction and evaluate it, we reach 0.8, it’s good:

We try to predict the new data:



Use the pipeline that we build earlier and print the predictions schema:



And try to predict several rows, the result are shown below:

