Scala

**Var a -mutable**

**Val b -immutable**

Scala vs python

**Statically typed dynamically typed**

It does not allow other data types for that variable

var a=1

a=”hi” --not supported

Python: it overwrites the old declaration by the new

a=1

a=’hi’ ===> it overwrites int data type to string

**Type inferred :**  it takes data type from the data

var a=1 → int type is inferred from 1

val salesdf=spark.read.option(“header”,”true”).option(“inferSchema”,”true”).csv(“src/superstore.csv”)

Day 9

#spark streaming vs spark structured streaming using wordcount

import org.apache.spark.\_

import org.apache.spark.streaming.\_

import org.apache.spark.streaming.StreamingContext.\_

val conf = new SparkConf().setMaster("local[2]").setAppName("NetworkWordCount")

val ssc = new StreamingContext(conf, Seconds(10))

val lines = ssc.socketTextStream("localhost", 9998)

val words = lines.flatMap(\_.split(" "))

val pairs = words.map(word => (word, 1))

val wordCounts = pairs.reduceByKey(\_ + \_)

wordCounts.print()

###spark structured streaming

val lines = spark.readStream

.format("socket")

.option("host", "localhost")

.option("port", 9998)

.load()

val words = lines.as[String].flatMap(\_.split(" "))

val wordCounts = words.groupBy("value").count()

val query = wordCounts.writeStream

.outputMode("complete")

.format("console")

.start()

=======================================================

#3to get scala spark

spark-shell

val salesdf=spark.read.option("header","true").option("inferSche

================================================================

Day 7

joining two dataframes

/;

empDF.join(deptDF,empDF.emp\_dept\_id == deptDF.dept\_id,"inner").show()

empDF.join(deptDF,"dept\_id","inner").show()

How to load json file into hive table and handle the query

explore : lateral view , explode commands

how to query array data type in hive table

performing join in dataframe and rdd

pyspark - joins , agg , filter , sort, converting dataframe column to list

converting rdd to dataframe

creating dataframe from schema and data

concating to columns in dataframe and rdd

full\_names=enroll.select("memberId",F.concat\_ws(" ",F.col("firstName"),F.col("lastName")).alias("full\_name")).show()

def most\_common\_tags(self, tags: DataFrame, n: int) -> List[str]:

tag\_counts = tags.select(tags["tags"].alias("tag")) \

.groupBy("tag") \

.count() \

.orderBy("count", ascending=False) \

.limit(n)

most\_common\_tags\_list = [row["tag"] for row in tag\_counts.collect()]

return most\_common\_tags\_list

videos.csv videoId , videoname , views , likes tags ==> videoId , tag

## to run a python code

spark-submit /home/labuser/Downloads/spark-3.5.0-bin-hadoop3/examples/src/main/python/wordcount.py dataengg/demo.txt

##create new file video\_analysis.py in home

from pyspark.sql import DataFrame

from main.base import PySparkJobInterface

import pyspark.sql.functions as F

from typing import List

from pyspark.sql import SparkSession

def top\_n\_videos\_by\_likes\_to\_dislikes\_ratio(self, videos: DataFrame, n: int) -> DataFrame:

# Write your code here

def most\_common\_tags(self, tags: DataFrame, n: int) -> List[str]:

case 2

def filter\_membership(self, enrollment: DataFrame, memberships: DataFrame) -> DataFrame:

# Write your code here

def generate\_full\_name(self, enrollment: DataFrame, membership: DataFrame) -> DataFrame:

# Write your code here

full\_names = enrollments.select(

"member\_id",

F.concat\_ws(" ", F.col("first\_name"), F.col("last\_name")).alias("full\_name")

)

return full\_names

================================================================================

https://github.com/akgeoinsys/akdatasets.git

spark-submit /home/labuser/Downloads/spark-3.5.0-bin-hadoop3/examples/src/main/python/streaming/network\_wordcount.py localhost 9998

hive for mapreduce / spark-sql for spark sql ==> beeline for both ,just we need to set the execution engine

##in new terminal

beeline

beeline> !connect jdbc:hive2://cdhserver:10000

===============================================================

>>> usersdf.cache()

DataFrame[name: string, favorite\_color: string, favorite\_numbers: array<int>]

>>> usersdf.show()

+------+--------------+----------------+

| name|favorite\_color|favorite\_numbers|

+------+--------------+----------------+

|Alyssa| null| [3, 9, 15, 20]|

| Ben| red| []|

+------+--------------+----------------+

## reading superstore,csv as salesdf in pyspark

spark.read.csv("/user/labuser/src/superstore.csv")

hdfs dfs -put /home/labuser/datasets/emp.csv hdfs://cdhserver:8020/user/labuser/src