pyspark.sql.SparkSession class

from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

SparkSession.stop() method

spark.stop()

pyspark.sql.DataFrame

pyspark.sql.Row

df = spark.read.load(path, options…)

The creation of a DataFrame from JSON files :

load(path) method of the pyspark.sql.DataFrameReader class

df = spark.read.load(path, format="json", …) or df = spark.read.json(path, …)

multiLine = True

spark.createDataFrame(data, schema) method

**Create a DataFrame from a csv file**

Create a Spark Session object

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons.csv

df = spark.read.load("persons.csv", format="csv", header=True, inferSchema=True)

**Create a DataFrame from a JSON files**

**Example1:**

Create a DataFrame from a JSON Lines text formatted file containing the profiles of a set of persons ♣ Each line of the file contains a JSON object containing name and age of a person ▪ Age can assume the null value

# Create a Spark Session object

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons. json

df = spark.read.load( "persons.json", format="json")

**Example2:**

Create a DataFrame from a folder containing a set of “standard” multiline JSON files \* Each input JSON file contains the profile of one person \* Name and Age ▪ Age can assume the null value

# Create a Spark Session object

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons.csv

df = spark.read.load( "folder\_JSONFiles/", format="json", multiLine=True)

**Example3:**

Create a DataFrame from a Python list containing the following data ♣ (19, "Justin") ♣ (30, "Andy") ♣ (None, "Michael") \* The column names must be set to “age” and “name”

# Create a Spark Session object

spark = SparkSession.builder.getOrCreate()

# Create a Python list of tuples

profilesList = [(19, "Justin"), (30, "Andy"), (None, "Michael")]

# Create a DataFrame from the profilesList

df = spark.createDataFrame(profilesList,["age","name"])

**Example4:**

Create a DataFrame from a csv file containing the profiles of a set of persons ♣ Each line of the file contains name and age of a person ♣ The first line contains the header, i.e., the name of the attributes/columns \* Transform the input DataFrame into an RDD, select only the name field/column and store the result in the output folder

# Create a Spark Session object

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons.csv

df = spark.read.load( "persons.csv", format="csv", header=True, inferSchema=True)

#Define an RDD based on the content of the DataFrame

rddRows = df.rdd

# Use the map transformation to extract the name field/column

rddNames = rddRows.map(lambda row: row.name)

Store the result

rddNames.saveAsTextFile(outputPath)

**Example4:**

Create a DataFrame from a csv file containing the profiles of a set of persons ♣ The content of persons.csv is name,age Andy,30 Michael, Justin,19 Michael,20 ϒ Print the content of the first 2 persons (i.e., the first 2 rows of the DataFrame)

Create a new DataFrame without duplicates (about the name)

#Create a Spark Session object

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons.csv

df = spark.read.load( "persons.csv", format="csv", header=True, inferSchema=True)

df.show(2)

df\_distinct = df.distinct()

**Example5:**

Create a DataFrame from the persons2.csv file that contains the profiles of a set of persons ♣ The first line contains the header ♣ The others lines contain the users’ profiles ▪ One line per person ▪ Each line contains name, age, and gender of a person ♣

Example name,age,gender Paul,40,male John,40,male ..

\*Create a new DataFrame containing only name and age of the persons

# Create a Spark Session object

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons2.csv

df = spark.read.load( "persons2.csv", format="csv", header=True, inferSchema=True)

dfNamesAges = df.select("name", "age")

**Example6:**

Create a DataFrame from the persons2.csv file that contains the profiles of a set of persons ♣ The first line contains the header ♣ The others lines contain the users’ profiles ▪ Each line contains name, age, and gender of a person ϒ Create a new DataFrame containing the same columns of the initial dataset plus an additional column called newAge containing the value of age incremented by one

# Create a Spark Session object

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons.csv

df = spark.read.load("persons2.csv", format="csv", header=True, inferSchema=True)

# Create a new DataFrame with four columns: name, age, gender, newAge = age +1

dfNewAge = df.selectExpr("name", "age", "gender", "age+1 as newAge")

**Example6:**

Create a DataFrame from the persons.csv file that contains the profiles of a set of persons ♣ The first line contains the header ♣ The others lines contain the users’ profiles ▪ Each line contains name and age of a person ϒ Create a new DataFrame containing only the persons with age between 20 and 31

spark = SparkSession.builder.getOrCreate()

df = spark.read.load( "persons.csv", format="csv", header=True, inferSchema=True)

df\_filtered = df.filter("age>=20 and age<=31")

**Example6:**

Create two DataFrames ♣ One based on the persons\_id.csv file that contains the profiles of a set of persons ▪ Schema: uid,name,age ♣ One based on the liked\_sports.csv file that contains the liked sports for each person ▪ Schema: uid,sportname ϒ Join the content of the two DataFrames (uid is the join column) and show it on the standard output

spark = SparkSession.builder.getOrCreate()

# Read persons\_id.csv and store it in a DataFrame

dfPersons = spark.read.load("persons\_id.csv", format="csv", header=True, inferSchema=True)

# Read liked\_sports.csv and store it in a DataFrame

dfUidSports = spark.read.load("liked\_sports.csv", format="csv", header=True, inferSchema=True)

# Join the two input DataFrames

dfPersonLikes = dfPersons.join(dfUidSports, dfPersons.uid == dfUidSports.uid)

# Print the result on the standard output

dfPersonLikes.show()

**Example7:**

Create two DataFrames ♣ One based on the persons\_id.csv file that contains the profiles of a set of persons ▪ Schema: uid,name,age ♣ One based on the banned.csv file that contains the banned users ▪ Schema: uid,bannedmotivation ϒ Select the profiles of the non-banned users and show them on the standard output

spark = SparkSession.builder.getOrCreate()

# Read persons\_id.csv and store it in a DataFrame

dfPersons = spark.read.load("persons\_id.csv", format="csv", header=True, inferSchema=True)

# Read banned.csv and store it in a DataFrame

dfBannedUsers = spark.read.load("banned.csv", format="csv", header=True, inferSchema=True)

# Apply the Left Anti Join on the two input DataFrames

dfSelectedProfiles = dfPersons.join(dfBannedUsers, dfPersons.uid == dfBannedUsers.uid, "left\_anti“)

# Print the result on the standard output

dfSelectedProfiles.show()

**Example8:**

Create a DataFrame from the persons.csv file that contains the profiles of a set of persons ♣ The first line contains the header ♣ The others lines contain the users’ profiles ▪ Each line contains name and age of a person ϒ Create a Dataset containing the average value of age

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons.csv

df = spark.read.load("persons.csv", format="csv", header=True, inferSchema=True)

# Compute the average of age

averageAge = df.agg({"age": "avg"})

**Example8:**

Create a DataFrame from the persons.csv file that contains the profiles of a set of persons ♣ The first line contains the header ♣ The others lines contain the users’ profiles ▪ Each line contains name and age of a person ϒ Create a DataFrame containing the for each name the average value of age

spark = SparkSession.builder.getOrCreate()

df = spark.read.load( "persons.csv", format="csv", header=True, inferSchema=True)

grouped = df.groupBy("name").avg("age")

**Example9:**

Create a DataFrame from the persons.csv file that contains the profiles of a set of persons ♣ The first line contains the header ♣ The others lines contain the users’ profiles ▪ Each line contains name and age of a person ϒ Create a DataFrame containing the for each name the average value of age and the number of person with that name

spark = SparkSession.builder.getOrCreate()

df = spark.read.load( "persons.csv", format="csv", header=True, inferSchema=True)

grouped = df.groupBy("name") .agg({"age": "avg", "name": "count"})

**Example10:**

Create a DataFrame from a JSON file containing the profiles of a set of persons ♣ Each line of the file contains a JSON object containing name, age, and gender of a person ϒ Create a new DataFrame containing only the persons with age between 20 and 31 and print them on the standard output ♣ Use the SQL language to perform this operation

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons. json

df = spark.read.load( "persons.json", format="json")

# Assign the “table name” people to the df DataFrame

df.createOrReplaceTempView("people");

# Select the persons with age between 20 and 31 by querying the people table

selectedPersons = spark.sql("SELECT \* FROM people WHERE age>=20 and age<=31")

# Print the result on the standard output

selectedPersons.show()

**Example11:**

Create two DataFrames ♣ One based on the persons\_id.csv file that contains the profiles of a set of persons ▪ Schema: uid,name,age ♣ One based on the liked\_sports.csv file that contains the liked sports for each person ▪ Schema: uid,sportname ϒ Join the content of the two DataFrames and show it on the standard output

spark = SparkSession.builder.getOrCreate()

# Read persons\_id.csv and store it in a DataFrame

dfPersons = spark.read.load("persons\_id.csv", format="csv", header=True, inferSchema=True)

# Assign the “table name” people to the dfPerson

dfPersons.createOrReplaceTempView("people")

# Read liked\_sports.csv and store it in a DataFrame

dfUidSports = spark.read.load("liked\_sports.csv", format="csv", header=True, inferSchema=True)

# Assign the “table name” liked to dfUidSports

dfUidSports.createOrReplaceTempView("liked")

# Join the two input tables by using the SQL-like syntax

dfPersonLikes = spark.sql("SELECT \* from people, liked where people.uid=liked.uid")

# Print the result on the standard output

dfPersonLikes.show()

**Example12:**

Create a DataFrame from the persons.csv file that contains the profiles of a set of persons ♣ The first line contains the header ♣ The others lines contain the users’ profiles ▪ Each line contains name and age of a person ϒ Create a DataFrame containing for each name the average value of age and the number of person with that name ♣ Print its content on the standard output

Input file name,age Andy,30 Michael,15 Justin,19 Andy,40 ϒ

Expected output name,avg(age),count(name) Andy,35,2 Michael,15,1 Justin,19,1

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons.csv

df = spark.read.load( "persons.json", format="json")

# Assign the “table name” people to the df DataFrame

df.createOrReplaceTempView("people")

# Define groups based on the value of name and compute average and number of records for each group

nameAvgAgeCount = spark.sql("SELECT name, avg(age), count(name) FROM people GROUP BY name")

# Print the result on the standard output

nameAvgAgeCount.show()

**Example13:**

Create a DataFrame from the persons.csv file that contains the profiles of a set of persons ♣ The first line contains the header ♣ The others lines contain the users’ profiles ▪ Each line contains name, age, and gender of a person ϒ Store the DataFrame in the output folder by using the saveAsTextFile(..) method

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons.csv

df = spark.read.load( "persons.csv", format="csv", header=True, inferSchema=True)

# Save it df.rdd.saveAsTextFile(outputPath)

**Example14:**

Create a DataFrame from the persons.csv file that contains the profiles of a set of persons ♣ The first line contains the header ♣ The others lines contain the users’ profiles ▪ Each line contains name, age, and gender of a person ϒ Store the DataFrame in the output folder by using the write() method ♣ Store the result by using the CSV format

spark = SparkSession.builder.getOrCreate()

# Create a DataFrame from persons.csv

df = spark.read.load( "persons.csv", format="csv", header=True, inferSchema=True)

# Save it df.write.csv(outputPath, header=True)

**Example14:**

Define a UDFs that, given a string, returns the length of the string

# Create a Spark Session object

spark = SparkSession.builder.getOrCreate()

# Define the UDF # name: length # output: integer value

spark.udf.register("length", lambda x: len(x))

**Example15:**

\*Use of the defined UDF in a selectExpr transformation

result = inputDF.selectExpr("length(name) as size")

\*Use of the defined UDF in a SQL query

result = spark.sql("SELECT length(name) FROM profiles")

**Example16:**

Create a DataFrame from the purchases.csv file ♣ The first line contains the header ♣ The others lines contain the quantities of purchased products by users ▪ Each line contains userid,productid,quantity ϒ Create a first DataFrame containing the result of the cube method. Define one group for each pair userid, productid and compute the sum of quantity in each group ϒ Create a second DataFrame containing the result of the rollup method. Define one group for each pair userid, productid and compute the sum of quantity in each group

spark = SparkSession.builder.getOrCreate()

dfPurchases = spark.read.load("purchases.csv", format="csv", header=True, inferSchema=True)

dfCube=dfPurchases.cube("userid","productid “).agg({"quantity": "sum"})

dfRollup=dfPurchases.rollup("userid","productid“) .agg({"quantity": "sum"})

**Example17:**

We can suggest/force it by creating a broadcast version of a DataFrame

dfPersonLikesBroadcast = dfUidSports.join(broadcast(dfPersons), dfPersons.uid == dfUidSports.uid)