Data_processing_using_PySpark

November 14, 2025

1 Data Processing using Pyspark

[ ]:
```
#configuración en google colab de spark y pyspark
from google.colab import drive
drive.mount('/content/gdrive')
```

[ ]:
```
#instalar java y spark
!apt-get install openjdk-17-jdk-headless -qq > /dev/null
!wget -q https://downloads.apache.org/spark/spark-4.0.1/spark-4.0.1-bin-hadoop3.
 ↪tgz
!tar xf spark-4.0.1-bin-hadoop3.tgz
!pip install -q findspark
```

[ ]:
```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-17-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-4.0.1-bin-hadoop3"
```

[ ]:
```
import findspark
findspark.init()
```

[ ]:
```
from pyspark.sql import SparkSession
# en el cluster EMR no hay necesidad de crear el objeto spark ni sc, ya viene␣
 ↪con AWS EMR / Notebooks

# spark local con archivos locales:
spark = SparkSession.builder.master("local[*]").getOrCreate()


sc = spark.sparkContext
```

[ ]:
```
# Load csv Dataset
# desde gdrvie
df=spark.read.csv('gdrive/MyDrive/st0263-252/bigdata/datasets/sample_data.
 ↪csv',inferSchema=True,header=True)

# desde local
#df=spark.read.csv('../datasets/sample_data.csv',inferSchema=True,header=True)
```

```
# desde S3
#df=spark.read.csv('s3a://username_datalake/datasets/sample_data.
 ↪csv',inferSchema=True,header=True)
```

[ ]:
```
#columns of dataframe
df.columns
```

[ ]:
```
#check number of columns
len(df.columns)
```

[ ]:
```
#number of records in dataframe
df.count()
```

[ ]:
```
#shape of dataset
print((df.count(),len(df.columns)))
```

[ ]:
```
#printSchema
df.printSchema()
```

[ ]:
```
#fisrt few rows of dataframe
df.show(5)
```

[ ]:
```
#select only 2 columns
df.select('age','mobile').show(5)
```

[ ]:
```
#info about dataframe
df.describe().show()
```

[ ]:
```
from pyspark.sql.types import StringType,DoubleType,IntegerType
```

[ ]:
```
#add column
df.withColumn("age_after_10_yrs",(df["age"]+10)).show(10,False)
```

[ ]:
```
#add column
df.withColumn('age_double',df['age'].cast(DoubleType())).show(10,False)
```

[ ]:
```
#add column
df.withColumn("age_after_10_yrs",(df["age"]+10)).show(10,False)
```

[ ]:
```
#filter the records
df.filter(df['mobile']=='Vivo').show()
```

[ ]:
```
#filter the records
df.filter(df['mobile']=='Vivo').select('age','ratings','mobile').show()
```

[ ]:
```
#filter the multiple conditions
df.filter(df['mobile']=='Vivo').filter(df['experience'] >10).show()
```

```
[ ]: #filter the multiple conditions
df.filter((df['mobile']=='Vivo')&(df['experience'] >10)).show()
```

```
[ ]: #Distinct Values in a column
df.select('mobile').distinct().show()
```

```
[ ]: #distinct value count
df.select('mobile').distinct().count()
```

```
[ ]: df.groupBy('mobile').count().show(5,False)
```

```
[ ]: # Value counts
df.groupBy('mobile').count().orderBy('count',ascending=False).show(5,False)
```

```
[ ]: # Value counts
df.groupBy('mobile').mean().show(5,False)
```

```
[ ]: df.groupBy('mobile').sum().show(5,False)
```

```
[ ]: # Value counts
df.groupBy('mobile').max().show(5,False)
```

```
[ ]: # Value counts
df.groupBy('mobile').min().show(5,False)
```

```
[ ]: #Aggregation
df.groupBy('mobile').agg({'experience':'sum'}).show(5,False)
```

```
[ ]: # UDF
from pyspark.sql.functions import udf
```

```
[ ]: #normal function
def price_range(brand):
    if brand in ['Samsung','Apple']:
        return 'High Price'
    elif brand =='MI':
        return 'Mid Price'
    else:
        return 'Low Price'
```

```
[ ]: #create udf using python function
brand_udf=udf(price_range,StringType())
#apply udf on dataframe
df.withColumn('price_range',brand_udf(df['mobile'])).show(10,False)
```

```
[ ]: #using lambda function
age_udf = udf(lambda age: "young" if age <= 30 else "senior", StringType())
#apply udf on dataframe
```

```
df.withColumn("age_group", age_udf(df.age)).show(10,False)
```

[ ]:
```
#pandas udf
from pyspark.sql.functions import pandas_udf, PandasUDFType
```

[ ]:
```
#create python function
def remaining_yrs(age):
    yrs_left=100-age

    return yrs_left
```

[ ]:
```
# library requires by pandas_udf()
!pip install pyarrow
```

[ ]:
```
#create udf using python function
length_udf = pandas_udf(remaining_yrs, IntegerType())
#apply pandas udf on dataframe
df.withColumn("yrs_left", length_udf(df['age'])).show(10,False)
```

[ ]:
```
#udf using two columns
def prod(rating,exp):
    x=rating*exp
    return x
```

[ ]:
```
#create udf using python function
prod_udf = pandas_udf(prod, DoubleType())
#apply pandas udf on multiple columns of dataframe
df.withColumn("product", prod_udf(df['ratings'],df['experience'])).
 ↪show(10,False)
```

[ ]:
```
#duplicate values
df.count()
```

[ ]:
```
#drop duplicate values
df=df.dropDuplicates()
```

[ ]:
```
#validate new count
df.count()
```

[ ]:
```
#drop column of dataframe
df_new=df.drop('mobile')
```

[ ]:
```
df_new.show(10)
```

[ ]:
```
# saving file to csv
```

[ ]:
```
#current working directory
!pwd
```

[ ]:
```
#target directory
#pathcsv_out='../out/df_csv'
pathcsv_out='gdrive/MyDrive/st0263-252/out/df_csv'
# hacia S3
# write_uri='s3://bucket_name/df_csv'
```

[ ]:
```
#save the dataframe as single csv
df.coalesce(1).write.format("csv").option("header","true").save(pathcsv_out)
```

[ ]:
```
# parquet
```

[ ]:
```
#target location
#pathparquet_out='../out/df_parquet'
pathparquet_out='gdrive/MyDrive/st0263-252/out/df_parquet'

# hacia S3
# write_uri='s3://bucket_name/df_parquet'
```

[ ]:
```
#save the data into parquet format
df.write.format('parquet').save(pathparquet_out)
```