ex47

August 14, 2022

[ ]:
```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

conf = SparkConf().setAppName("ex47")
sc = SparkContext(conf=conf)
sparkSQL = SparkSession.builder.getOrCreate()
```

[2]:
```
inputPath = "data/Ex47/data/"
outputPath = "out47/"
```

[ ]:
```
#loading data
df = sparkSQL.read.load(inputPath,
                        format="csv",
                        header=True,
                        inferSchema=True)
df.show()
```

[6]:
```
#using dataframes
df_filtered = df.filter("gender='male'")
df_newAge = df_filtered.selectExpr("name", "age+1 AS newAge")
df_sorted = df_newAge.sort(df_newAge.newAge.desc(), df_newAge.name)
df_sorted.write.csv(outputPath, header=False)
```

[ ]:
```
#using SQL
df.createOrReplaceTempView("people")
df = sparkSQL.sql("""
SELECT name, age+1 AS newAge
FROM people
WHERE gender='male
SORT BY age desc, name
""")
```

ex48

August 14, 2022

[ ]:
```
from pyspark import SparkContext, SparkConf
from pyspark.sql import SparkSession

conf = SparkConf().setAppName("ex48")
sc = SparkContext(conf=conf)
ssql = SparkSession.builder.getOrCreate()
```

[2]:
```
inputPath = "data/Ex48/data/"
outputPath = "out48/"
```

[ ]:
```
df = ssql.read.load(
    inputPath,
    format="csv",
    header=True,
    inferSchema=True
)
df.show()
```

[5]:
```
dfNameCountedAgeAvareged = df.groupBy("name").agg({"name":"count", "age":"avg"})
```

[7]:
```
df_filtered = dfNameCountedAgeAvareged.filter("count(name)>=2")
```

[8]:
```
final_df = df_filtered.select("name", "avg(age)")
```

[9]:
```
final_df.write.csv(outputPath, header=False)
```

[ ]:
```
#using SQL
df.createOrReplaceTempView("people")
df_sql = ssql.sql("""
SELECT name, avg(age) as ageavg
FROM people
GROUP BY name
HAVING count(*) >=2
""")
```

ex49

August 14, 2022

[ ]:
```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

conf = SparkConf().setAppName("ex49")
sc = SparkContext(conf=conf)
ssql = SparkSession.builder.getOrCreate()
```

[3]:
```
inputPath = "data/Ex49/data/"
outputPath = "out49/"
```

[4]:
```
df = ssql.read.load(
    inputPath,
    format="csv",
    header=True,
    inferSchema=True
)
```

[ ]:
```
#definisco una UDF per implementare il mapping richiesto di age
ssql.udf.register("newAge", lambda age: "["+str((age//10)*10)+"-"+str((age//
 ↪10)*10+9)+"]")
```

[6]:
```
final_df = df.selectExpr("name", "surname", "newAge(age) as AgeCategory").write.
 ↪csv(outputPath, header=True)
```

[ ]:
```
#posso fare la stessa cosa in SQL dopo aver definito la nuova UDF
ssql.createOrReplaceTempView("people")
df_sql = ssql.sql("""
SELECT name, surname, newAge(age) as AgeCategory
FROM people
""")
```

ex50

August 14, 2022

[ ]:
```
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

conf = SparkConf().setAppName("ex50")
sc = SparkContext(conf=conf)
ssql = SparkSession.builder.getOrCreate()
```

[2]:
```
inputPath = "data/Ex50/data/"
outputPath = "out50/"
```

[3]:
```
df = ssql.read.load(
    inputPath,
    format="csv",
    header=True,
    inferSchema=True
)
```

[ ]:
```
ssql.udf.register("Concatenation", lambda name, surname : name+" "+surname)
```

[5]:
```
final_df = df.selectExpr("Concatenation(name, surname) as name_surname")
```

[6]:
```
final_df.write.csv(outputPath, header=False)
```

[ ]:
```
#per usare SQL, esattamnte come prima, una volta aver definito la UDF:
df.createOrReplaceTempView("people")
final_df = ssql.sql("""
SELECT Concatenation(name, surname) as name_surname
FROM people
""")
```