ORDERBY & SORT IN PYSPARK

orderBy & sort in PySpark

Cmd 2

```
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType

spark = SparkSession.builder.appName("ordering").getOrCreate()

data = [("Tom", 25), ("Akshay", 55)]
schema = StructType([
    StructField("name", StringType(), True),
    StructField("age", IntegerType(), True)
])
df1 = spark.createDataFrame(data, schema)
df1.orderBy("name").show()
df1.sort(df1.age.asc()).show()
df1.sort("name").show()
```

▸ (3) Spark Jobs

▸ df1: pyspark.sql.dataframe.DataFrame = [name: string, age: integer]

```
+------+---+
|  name|age|
+------+---+
|Akshay| 55|
|   Tom| 25|
+------+---+

+------+---+
|  name|age|
+------+---+
|   Tom| 25|
|Akshay| 55|
+------+---+

+------+---+
|  name|age|
+------+---+
|Akshay| 55|
|   Tom| 25|
+------+---+
```

UNION & UNIONALL IN PYSPARK

Union & UnionAll() in PySpark

In newer versions of Spark, using union will not perform deduplication, and both union and unionAll will behave the same way, i.e., they will include duplicate rows.

Cmd 4

Python

```
data1 = [("Tom", 25), ("Akshay", 55)]
schema1 =['name','age']
data2= [("Tom", 25), ("Akshay", 55)]
schema2 =['name','age']
data3= [("Thomas", 25), ("steve", 55)]
schema3 =['name','age']

df1 = spark.createDataFrame(data1, schema1)
df2=spark.createDataFrame(data2, schema2)
df3=spark.createDataFrame(data3, schema3)

df1.union(df2).show()
df1.unionAll(df2).show()
```

```
df1.union(df2).show()
df1.unionAll(df2).show()
```

▸ (6) Spark Jobs

▸ df1: pyspark.sql.dataframe.DataFrame = [name: string, age: long]

▸ df2: pyspark.sql.dataframe.DataFrame = [name: string, age: long]

▸ df3: pyspark.sql.dataframe.DataFrame = [name: string, age: long]

```
+------+---+
|  name|age|
+------+---+
|   Tom| 25|
|Akshay| 55|
|   Tom| 25|
|Akshay| 55|
+------+---+

+------+---+
|  name|age|
+------+---+
|   Tom| 25|
|Akshay| 55|
|   Tom| 25|
|Akshay| 55|
+------+---+
```

GROUPBY & AGG FUNCTION

groupBy agg() function in PySpark

Cmd 6

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import sum, desc
spark = SparkSession.builder.appName("HighestPurchaseAmount").getOrCreate()

data = [
    (1, 100, "2023-01-15"),
    (2, 150, "2023-02-20"),
    (1, 200, "2023-03-10"),
    (3, 50, "2023-04-05"),
    (2, 120, "2023-05-15"),
    (1, 300, "2023-06-25")
]

columns = ["customer_id", "purchase_amount", "purchase_date"]
df = spark.createDataFrame(data, columns)

total_purchase_df = df.groupBy("customer_id").agg(sum("purchase_amount").alias("total_purchase_amount"))
# Order by total purchase amount in descending order and select the top row
highest_purchase_customer = total_purchase_df.orderBy(desc("total_purchase_amount")).first()
# Print the result
print("Customer with the highest total purchase amount:")
print("Customer ID:", highest_purchase_customer["customer_id"])
print("Total Purchase Amount:", highest_purchase_customer["total_purchase_amount"])
```

▸ (2) Spark Jobs

▸ df: pyspark.sql.dataframe.DataFrame = [customer_id: long, purchase_amount: long ... 1 more field]

▸ total_purchase_df: pyspark.sql.dataframe.DataFrame = [customer_id: long, total_purchase_amount: long]

```
Customer with the highest total purchase amount:
Customer ID: 1
Total Purchase Amount: 600
```

UNIONBYNAME() IN PYSPARK

unionByName()

Cmd 8

```
# Create two DataFrames
data1=[(1,'tom','male')]
schema1 = ["id", "name", "gender"]

data2=[(1,'thomas',2000)]
schema2=["id","name","salary"]

df1 = spark.createDataFrame(data1, schema1)
df2 = spark.createDataFrame(data2, schema2)

# Using union (order of columns matters)
union_df = df1.union(df2)
union_df.show()

# Using unionByName (order of columns doesn't matter  & columns will be ordered based on their names.)
union_by_name_df = df1.unionByName(allowMissingColumns=True,other=df2)
union_by_name_df.show()
```

▸ (6) Spark Jobs

▸ (6) Spark Jobs

▸ df1: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 1 more field]

▸ df2: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 1 more field]

▸ union_df: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 1 more field]

▸ union_by_name_df: pyspark.sql.dataframe.DataFrame = [id: long, name: string ... 2 more fields]

```
+---+------+------+
| id|  name|gender|
+---+------+------+
|  1|   tom|  male|
|  1|thomas|  2000|
+---+------+------+


+---+------+------+------+
| id|  name|gender|salary|
+---+------+------+------+
|  1|   tom|  male|  null|
|  1|thomas|  null|  2000|
+---+------+------+------+
```