pyspark groupBy examples

ddas.tech/pyspark-groupby-examples/

November 22, 2022

Created By: Debasis Das (21-Nov-2022)

groupBy groups the DataFrame using the specified columns, so we can run aggregation on them.

Following are a set of methods available for aggregation on a DataFrame

- **agg(*exprs)**: Computes aggregates and returns the result as a DataFrame
- **avg(* cols)**: Computes average values for each numeric columns for each group.
- **count()** – Counts the number of records for each group.
- **max(*cols)** – Computes the max value for each numeric columns for each group.
- **mean(* cols)** – Computes average values for each numeric columns for each group.
- **min(* cols)** – Computes the min value for each numeric column for each group.
- **sum(* cols)** – Compute the sum for each numeric columns for each group.

```
import findspark
findspark.init()

import pyspark
from pyspark.sql import SparkSession
from pyspark import SparkConf
from pyspark.sql.types import StructType, StructField, DateType, StringType,
IntegerType
from pyspark.sql.functions import expr

conf = SparkConf().setMaster("local[3]").setAppName("SparkDataFrameGroupBy")
spark = SparkSession.builder.config(conf=conf).getOrCreate()
spark.sparkContext.setLogLevel("WARN")
print(spark)
```

Lets start by creating some sample data

```
import pandas as pd
import numpy as np

countries = ["USA","Mexico","Brazil","Canada"]
cars = ["BMW X5","BMW X7","Ford Explorer","Ford Expedition","Jeep Wrangler","Jeep
Cherokee"]
weeks = []
for i in range(1,5):
    weeks.append(f"Week_{i}")

num_records = 96
df = pd.DataFrame({"country":np.random.choice(countries,num_records),
                   "car":np.random.choice(cars,num_records),
                   "week":np.random.choice(weeks,num_records),
                   "units_sales":np.random.randint(20,size = num_records),
                   "used_new":np.random.choice(["Used","New"],num_records),
                   "price_per_unit":np.random.randint(low = 20000, high = 55000,size
= num_records)
                  })
print(df.head(10))
print(df.shape)
```

	country	car	week	units_sales	used_new	price_per_unit
0	Canada	Jeep Cherokee	Week_2	5	New	37177
1	USA	BMW X7	Week_3	1	New	27322
2	Canada	Ford Explorer	Week_1	1	Used	35826
3	USA	Jeep Cherokee	Week_4	12	New	24610
4	Canada	Ford Explorer	Week_1	0	New	35688
5	Mexico	Ford Expedition	Week_1	17	New	51116
6	USA	Jeep Cherokee	Week_4	6	New	44375
7	Canada	BMW X7	Week_2	5	Used	23811
8	USA	Ford Explorer	Week_1	5	Used	27027
9	Canada	BMW X7	Week_4	15	Used	51215

```
(96, 6)
```

Now lets create the spark dataframe from the pandas dataframe

```
sparkDF=spark.createDataFrame(df)
sparkDF.printSchema()
sparkDF.show(5)
```

```
root
 |-- country: string (nullable = true)
 |-- car: string (nullable = true)
 |-- week: string (nullable = true)
 |-- units_sales: long (nullable = true)
 |-- used_new: string (nullable = true)
 |-- price_per_unit: long (nullable = true)

+-------+---------------+------+-----------+--------+--------------+
|country|            car|  week|units_sales|used_new|price_per_unit|
+-------+---------------+------+-----------+--------+--------------+
|    USA|  Jeep Cherokee|Week_1|         14|     New|         54291|
| Brazil|  Ford Explorer|Week_2|          2|     New|         51667|
| Brazil|         BMW X5|Week_1|          0|     New|         42302|
| Canada|Ford Expedition|Week_4|         11|     New|         38956|
| Canada|Ford Expedition|Week_2|         14|    Used|         51705|
+-------+---------------+------+-----------+--------+--------------+
only showing top 5 rows
```

Now lets add a new column called as revenue which is product of unit sales and price per unit

```
df1 = sparkDF.withColumn('revenue', (sparkDF.units_sales * sparkDF.price_per_unit))
df1.show(5)

+-------+---------------+------+-----------+--------+--------------+-------+
|country|            car|  week|units_sales|used_new|price_per_unit|revenue|
+-------+---------------+------+-----------+--------+--------------+-------+
|    USA|  Jeep Cherokee|Week_1|         14|     New|         54291| 760074|
| Brazil|  Ford Explorer|Week_2|          2|     New|         51667| 103334|
| Brazil|         BMW X5|Week_1|          0|     New|         42302|      0|
| Canada|Ford Expedition|Week_4|         11|     New|         38956| 428516|
| Canada|Ford Expedition|Week_2|         14|    Used|         51705| 723870|
+-------+---------------+------+-----------+--------+--------------+-------+
only showing top 5 rows
```

Lets start with a simple example of doing a group by using sum function

```
revenueByCountry = df1.groupBy("country").sum("revenue")
revenueByCountry.show()

+-------+------------+
|country|sum(revenue)|
+-------+------------+
|    USA|     6249630|
| Mexico|     9239064|
| Canada|    11064790|
| Brazil|     7088142|
+-------+------------+

revenueByCountry.sort("sum(revenue)").show()
```

```
+-------+------------+
|country|sum(revenue)|
+-------+------------+
|    USA|     6249630|
| Brazil|     7088142|
| Mexico|     9239064|
| Canada|    11064790|
+-------+------------+

revenueByCountry.sort("sum(revenue)",ascending=False).show()

+-------+------------+
|country|sum(revenue)|
+-------+------------+
| Canada|    11064790|
| Mexico|     9239064|
| Brazil|     7088142|
|    USA|     6249630|
+-------+------------+
```

Multi Column groupBy

```
revenueByCountryCar =
df1.groupBy("country","car").sum("revenue").orderBy("country","car")
revenueByCountryCar.show(10)

+-------+---------------+------------+
|country|            car|sum(revenue)|
+-------+---------------+------------+
| Brazil|         BMW X5|      391329|
| Brazil|         BMW X7|     2639767|
| Brazil|Ford Expedition|     2337074|
| Brazil|  Ford Explorer|     1796776|
| Brazil|  Jeep Cherokee|      373944|
| Canada|         BMW X5|     1214830|
| Canada|         BMW X7|     1205818|
| Canada|Ford Expedition|     1984226|
| Canada|  Ford Explorer|      461532|
| Canada|  Jeep Cherokee|     1517203|
+-------+---------------+------------+
only showing top 10 rows

countByCountry = df1.groupBy("country").count()
countByCountry.show()

+-------+-----+
|country|count|
+-------+-----+
|    USA|   21|
| Mexico|   26|
| Canada|   28|
| Brazil|   21|
+-------+-----+
```

```
from pyspark.sql.functions import *
salesAggregate = df1.groupBy("country").agg(
    sum("units_sales").alias("tot_unit"),
    min("units_sales").alias("min_unit_sl"),
    max("units_sales").alias("max_unit_sl"),
    avg("units_sales").alias("avg_unit_sl"),
    avg("price_per_unit").alias("avg_price"),
    sum("revenue").alias("total_rev")
)
salesAggregate.show()
```

```
+-------+--------+-----------+-----------+-----------------+-----------------+---------+
|country|tot_unit|min_unit_sl|max_unit_sl|      avg_unit_sl|        avg_price|total_rev|
+-------+--------+-----------+-----------+-----------------+-----------------+---------+
|    USA|     171|          0|         17|8.142857142857142|36700.71428571428|  6249630|
| Mexico|     255|          0|         19|9.807692307692308|          33909.0|  9239064|
| Canada|     272|          0|         19|9.714285714285714|39565.96428571428| 11064790|
| Brazil|     185|          0|         19| 8.80952380952381|39308.80952380953|  7088142|
+-------+--------+-----------+-----------+-----------------+-----------------+---------+
```

```
salesdf1Agg = df1.groupBy("country","car").agg(
    {"revenue":"sum",
     "units_sales":"sum"}
).orderBy("country")
salesdf1Agg.show(10)
```

```
+-------+--------------+------------+----------------+
|country|           car|sum(revenue)|sum(units_sales)|
+-------+--------------+------------+----------------+
| Brazil|  Jeep Cherokee|      373944|               8|
| Brazil|  Ford Explorer|     1796776|              51|
| Brazil|         BMW X7|     2639767|              60|
| Brazil|         BMW X5|      391329|              11|
| Brazil|Ford Expedition|     2337074|              59|
| Canada|         BMW X5|     1214830|              39|
| Canada|  Jeep Wrangler|     2925280|              76|
| Canada|Ford Expedition|     1984226|              49|
| Canada|  Ford Explorer|      461532|              12|
| Canada|         BMW X7|     1205818|              37|
+-------+--------------+------------+----------------+
only showing top 10 rows
```

```
salesdf2Agg = df1.groupBy("country","car").agg(
    min('price_per_unit'),
    max('price_per_unit'),
    avg('price_per_unit')
).orderBy("country","car")
salesdf2Agg.show(5)
```

```
+-------+--------------+------------------+------------------+------------------+
|country|           car|min(price_per_unit)|max(price_per_unit)|avg(price_per_unit)|
+-------+--------------+------------------+------------------+------------------+
| Brazil|         BMW X5|             25137|             42302|            35071.5|
| Brazil|         BMW X7|             20006|             54435|            45905.0|
| Brazil|Ford Expedition|             20503|             54010|           40201.25|
| Brazil|  Ford Explorer|             24182|             51667|  38966.57142857143|
| Brazil|  Jeep Cherokee|             46743|             46743|            46743.0|
+-------+--------------+------------------+------------------+------------------+
only showing top 5 rows
```

```
saleByCountryWeek =
df1.groupBy("country","week").sum("units_sales","revenue").orderBy("country","week")
saleByCountryWeek.show(10)
```

```
+-------+------+----------------+------------+
|country|  week|sum(units_sales)|sum(revenue)|
+-------+------+----------------+------------+
| Brazil|Week_1|              61|     2231673|
| Brazil|Week_2|              49|     2067129|
| Brazil|Week_3|              22|     1058548|
| Brazil|Week_4|              57|     2181540|
| Canada|Week_1|              47|     1875436|
| Canada|Week_2|              69|     3275314|
| Canada|Week_3|              68|     2447161|
| Canada|Week_4|              61|     1710978|
| Mexico|Week_1|              41|     1787169|
| Mexico|Week_2|              54|     1908588|
+-------+------+----------------+------------+
only showing top 10 rows
```