M Hendra Herviawan

kaggle linkedin github

Pyspark: GroupBy and Aggregate Functions

Sun 18 June 2017 Data Science M Hendra Herviawan #Data Wrangling, #Pyspark, #Apache Spark

GroupBy allows you to group rows together based off some column value, for example, you could group together sales data by the day the sale occured, or group repeat customer data based off the name of the customer.

Once you've performed the GroupBy operation you can use an aggregate function off that data. An aggregate function aggregates multiple rows of data into a single output, such as taking the sum of inputs, or counting the number of inputs.

```
from pyspark.sql import SparkSession

# May take a little while on a local computer
spark = SparkSession.builder.appName("groupbyagg").getOrCreate()
spark
```

```
        <div>
            <p><b>SparkSession - in-memory</b></p>

    <div>
        <p><b>SparkContext</b></p>

        <p><a href="http://10.142.0.3:4040">Spark UI</a></p>

        <dl>
          <dt>Version</dt>
            <dd><code>v2.2.0</code></dd>
          <dt>Master</dt>
            <dd><code>local[*]</code></dd>
          <dt>AppName</dt>
            <dd><code>groupbyagg</code></dd>
        </dl>
    </div>

        </div>
```

```
df = spark.read.csv('sales_info.csv', inferSchema=True, header=True)

df.printSchema()
```

```
root
 |-- Company: string (nullable = true)
 |-- Person: string (nullable = true)
 |-- Sales: double (nullable = true)
```

```
#Showing the data
df.show()
```

```
+-------+-------+-----+
|Company| Person|Sales|
+-------+-------+-----+
|   GOOG|    Sam|200.0|
|   GOOG|Charlie|120.0|
|   GOOG|  Frank|340.0|
|   MSFT|   Tina|600.0|
|   MSFT|    Amy|124.0|
|   MSFT|Vanessa|243.0|
|     FB|   Carl|870.0|
|     FB|  Sarah|350.0|
|   APPL|   John|250.0|
|   APPL|  Linda|130.0|
|   APPL|   Mike|750.0|
|   APPL|  Chris|350.0|
+-------+-------+-----+
```

```
df.describe().show()
```

```
+-------+-------+-------+------------------+
|summary|Company| Person|             Sales|
+-------+-------+-------+------------------+
|  count|     12|     12|                12|
|   mean|   null|   null| 360.5833333333333|
| stddev|   null|   null|250.08742410799007|
|    min|   APPL|  Chris|             120.0|
|    max|   MSFT|Vanessa|             870.0|
+-------+-------+-------+------------------+
```

Order by

```
# OrderBy
# Ascending
df.orderBy("Sales").show()

# this produces the same result
# df.orderBy(df["Sales"]).show()
```

```
+-------+-------+-----+
|Company| Person|Sales|
+-------+-------+-----+
|   GOOG|Charlie|120.0|
|   MSFT|    Amy|124.0|
|   APPL|  Linda|130.0|
|   GOOG|    Sam|200.0|
|   MSFT|Vanessa|243.0|
|   APPL|   John|250.0|
|   GOOG|  Frank|340.0|
|     FB|  Sarah|350.0|
|   APPL|  Chris|350.0|
|   MSFT|   Tina|600.0|
|   APPL|   Mike|750.0|
|     FB|   Carl|870.0|
+-------+-------+-----+
```

```
# Descending call off the column itself.
df.orderBy(df["Sales"].desc()).show()
```

```
+-------+-------+-----+
|Company| Person|Sales|
+-------+-------+-----+
|     FB|   Carl|870.0|
|   APPL|   Mike|750.0|
|   MSFT|   Tina|600.0|
|     FB|  Sarah|350.0|
|   APPL|  Chris|350.0|
|   GOOG|  Frank|340.0|
|   APPL|   John|250.0|
|   MSFT|Vanessa|243.0|
|   GOOG|    Sam|200.0|
|   APPL|  Linda|130.0|
|   MSFT|    Amy|124.0|
|   GOOG|Charlie|120.0|
+-------+-------+-----+
```

Dataframe Aggregation

A set of methods for aggregations on a DataFrame:

- agg
- avg
- count
- max
- mean
- min
- pivot
- sum

```
df.groupBy('Company')
```

```
<pyspark.sql.group.GroupedData at 0x7f532c65eba8>
```

This returns a GroupedData object, off of which you can all various methods

```
# Max
df.groupBy('Company').max().show()
```

```
+-------+----------+
|Company|max(Sales)|
+-------+----------+
|   APPL|     750.0|
|   GOOG|     340.0|
|     FB|     870.0|
|   MSFT|     600.0|
+-------+----------+
```

```
# Sum
df.groupBy('Company').sum().show()
```

```
+-------+----------+
|Company|sum(Sales)|
+-------+----------+
|   APPL|    1480.0|
|   GOOG|     660.0|
|     FB|    1220.0|
|   MSFT|     967.0|
+-------+----------+
# Sum by Agg
group_data = df.groupBy("Company")
group_data.agg({'Sales':'sum'}).show()
```

```
+-------+----------+
|Company|sum(Sales)|
+-------+----------+
|   APPL|    1480.0|
|   GOOG|     660.0|
|     FB|    1220.0|
|   MSFT|     967.0|
+-------+----------+
```

```
# Max by agg
group_data.agg({'Sales':'max'}).show()
```

```
+-------+----------+
|Company|max(Sales)|
+-------+----------+
|   APPL|     750.0|
|   GOOG|     340.0|
|     FB|     870.0|
|   MSFT|     600.0|
+-------+----------+
```

Not all methods need a groupby call, instead you can just call the generalized .agg() method, that will call the aggregate across all rows in the dataframe column specified. It can take in arguments as a single column, or create multiple aggregate calls all at once using dictionary notation.

```
# Sum
df.agg({'Sales':'sum'}).show()
```

```
+----------+
|sum(Sales)|
+----------+
|    4327.0|
+----------+
```

Group Function

Seperti layaknya SQL, Spark memiliki group function

```
from pyspark.sql.functions import countDistinct, avg, stddev
```

```
df.select(avg('Sales')).show()
```

```
+-----------------+
|       avg(Sales)|
+-----------------+
|360.5833333333333|
+-----------------+
```

```
# Change name of columns with alias
df.select(countDistinct("Sales").alias("Distinct Sales")).show()
```

```
+--------------+
|Distinct Sales|
+--------------+
|            11|
+--------------+
```

```
# That is a lot of precision for digits! Let's use the format_number to fix that!
from pyspark.sql.functions import format_number
sales_std = df.select(stddev("Sales").alias('std'))

# format_number("col_name",decimal places)
sales_std.select(format_number('std',2).alias('std_2digits')).show()
```

```
+-----------+
|std_2digits|
+-----------+
|     250.09|
+-----------+
```

```
# or with this one liner
df.select(stddev("Sales").alias('std')).select(format_number('std',2).alias('std_2digits')).show()
```

```
+-----------+
|std_2digits|
+-----------+
|     250.09|
+-----------+
```