PySpark Window Functions

NNK (https://sparkbyexamples.com/author/admin/) -
PySpark (https://sparkbyexamples.com/category/pyspark/)

PySpark Window functions are used to calculate results such as the rank, row number e.t.c over a range of input rows. In this article, I've explained the concept of window functions, syntax, and finally how to use them with PySpark SQL and PySpark DataFrame API. These come in handy when we need to make aggregate operations in a specific window frame on DataFrame columns.

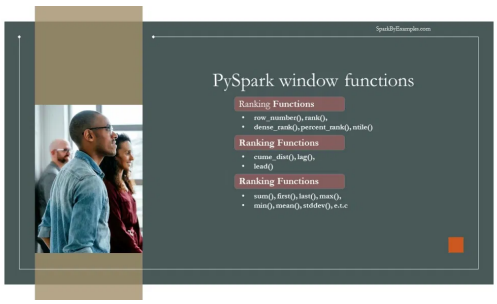
When possible try to leverage standard library as they are little bit more compile-time safety, handles null and perform better when compared to

UDF's. If your application is critical on performance try to avoid using custom UDF at all costs as these are not guarantee on performance.

1. Window Functions

PySpark Window functions operate on a group of rows (like frame, partition) and return a single value for every input row. PySpark SQL supports three kinds of window functions:

- ranking functions
- analytic functions
- aggregate functions



PySpark Window Functions

The below table defines Ranking and Analytic functions and for aggregate functions, we can use any existing aggregate functions (https://sparkbyexamples.com/pyspark/pyspark-aggregate-functions/) as a window function.

To perform an operation on a group first, we need to partition the data using `Window.partitionBy()` , and for row number and rank function we need to additionally order by on partition data using `orderBy` clause.

PySpark – withColumnRenamed() (https://sparkbyexamples.com/pyspark/pyspark-rename-dataframe-column/)

PySpark – where() & filter() (https://sparkbyexamples.com/pyspark/pyspark-where-filter/)

PySpark – drop() & dropDuplicates() (https://sparkbyexamples.com/pyspark/pyspark-distinct-to-drop-duplicates/)

PySpark – orderBy() and sort() (https://sparkbyexamples.com/pyspark/pyspark-orderby-and-sort-explained/)

PySpark – groupBy() (https://sparkbyexamples.com/pyspark/pyspark-groupby-explained-with-example/)

PySpark – join() (https://sparkbyexamples.com/pyspark/pyspark-join-explained-with-examples/)

PySpark – union() & unionAll() (https://sparkbyexamples.com/pyspark/pyspark-union-and-unionall/)

PySpark – unionByName() (https://sparkbyexamples.com/spark/spark-merge-two-dataframes-with-different-columns/)

PySpark – UDF (User Defined Function) (https://sparkbyexamples.com/pyspark/pyspark-udf-user-defined-function/)

PySpark – map() (https://sparkbyexamples.com/pyspark/pyspark-map-transformation/)

PySpark – flatMap() (https://sparkbyexamples.com/pyspark/pyspark-flatmap-transformation/)

pyspark – foreach() (https://sparkbyexamples.com/pyspark/pyspark-loop-iterate-through-rows-in-dataframe/#use-foreach-loop-through-dataframe)

PySpark – sample() vs sampleBy()

Click on each link to know more about these functions along with the Scala examples.

WINDOW FUNCTIONS USAGE & SYNTAX	PYSPARK WINDOW FUNCTIONS DESCRIPTION
row_number(): Column	Returns a sequential number starting from 1 within a window partition
rank(): Column	Returns the rank of rows within a window partition, with gaps.
percent_rank(): Column	Returns the percentile rank of rows within a window partition.
dense_rank(): Column	Returns the rank of rows within a window partition without any gaps. Where as Rank() returns rank with gaps.
ntile(n: Int): Column	Returns the ntile id in a window partition
cume_dist(): Column	Returns the cumulative distribution of values within a window partition
lag(e: Column, offset: Int): Column lag(columnName: String, offset: Int): Column lag(columnName: String, offset: Int, defaultValue: Any): Column	returns the value that is `offset` rows before the current row, and `null` if there is less than `offset` rows before the current row.

WINDOW FUNCTIONS USAGE & SYNTAX	PYSPARK WINDOW FUNCTIONS DESCRIPTION
lead(columnName: String, offset: Int): Column lead(columnName: String, offset: Int): Column lead(columnName: String, offset: Int, defaultValue: Any): Column	returns the value that is `offset` rows after the current row, and `null` if there is less than `offset` rows after the current row.

Before we start with an example, first let's create a PySpark DataFrame (https://sparkbyexamples.com/pyspark/different-ways-to-create-dataframe-in-pyspark/) to work with.

```
spark = SparkSession.builder.ap

simpleData = (("James", "Sales"
    ("Michael", "Sales", 4600),
    ("Robert", "Sales", 4100),
    ("Maria", "Finance", 3000),
    ("James", "Sales", 3000),
    ("Scott", "Finance", 3300),
    ("Jen", "Finance", 3900),
    ("Jeff", "Marketing", 3000)
    ("Kumar", "Marketing", 2000
    ("Saif", "Sales", 4100) \
  )

columns= ["employee_name", "dep
df = spark.createDataFrame(data
df.printSchema()
df.show(truncate=False)
```

Yields below output

pyspark/pyspark-read-and-write-parquet-file/)

PySpark – Read & Write JSON file (https://sparkbyexamples.com/pyspark/pyspark-read-json-file-into-dataframe/)


```
root
 |-- employee_name: string (nul
 |-- department: string (nullab
 |-- salary: long (nullable = t

+-------------+----------+-----
|employee_name|department|salar
+-------------+----------+-----
|James        |Sales     |3000
|Michael      |Sales     |4600
|Robert       |Sales     |4100
|Maria        |Finance   |3000
|James        |Sales     |3000
|Scott        |Finance   |3300
|Jen          |Finance   |3900
|Jeff         |Marketing |3000
|Kumar        |Marketing |2000
|Saif         |Sales     |4100
+-------------+----------+-----
```


PySpark Window

nking functions

row_number Window

nction

_number() window function is used
ive the sequential row number
ting from 1 to the result of each
dow partition.

PySpark Built-In Functions

PySpark – when() (https://sparkbyexamples.com/pyspark/pyspark-when-otherwise/)

PySpark – expr() (https://sparkbyexamples.com/pyspark/pyspark-sql-expr-expression-function/)

```
from pyspark.sql.window import 
from pyspark.sql.functions impo
windowSpec  = Window.partitionB

df.withColumn("row_number",row_
    .show(truncate=False)
```

Yields below output.

```
+-------------+----------+-----
|employee_name|department|salar
+-------------+----------+-----
|James        |Sales     |3000
|James        |Sales     |3000
|Robert       |Sales     |4100
|Saif         |Sales     |4100
|Michael      |Sales     |4600
|Maria        |Finance   |3000
|Scott        |Finance   |3300
|Jen          |Finance   |3900
|Kumar        |Marketing |2000
|Jeff         |Marketing |3000
+-------------+----------+-----
```

2.2 rank Window Function

`rank()` window function is used to provide a rank to the result within a window partition. This function leaves gaps in rank when there are ties.

```
"""rank"""
from pyspark.sql.functions impo
df.withColumn("rank",rank().ove
    .show()
```

Yields below output.

```
+-------------+----------+-----
|employee_name|department|salar
+-------------+----------+-----
|        James|     Sales|  300
|        James|     Sales|  300
|       Robert|     Sales|  410
|         Saif|     Sales|  410
|      Michael|     Sales|  460
|        Maria|   Finance|  300
|        Scott|   Finance|  330
|          Jen|   Finance|  390
|        Kumar| Marketing|  200
|         Jeff| Marketing|  300
+-------------+----------+-----
```

This is the same as the `RANK` function in SQL.

2.3 dense_rank Window Function

`dense_rank()` window function is used to get the result with rank of rows within a window partition without any gaps. This is similar to `rank()` function difference being rank function leaves gaps in rank when there are ties.

```
"""dens_rank"""
from pyspark.sql.functions impo
df.withColumn("dense_rank",dens
    .show()
```

Yields below output.

PySpark – dense_rank() (https://sparkbyexamples.com/pyspark/pyspark-window-functions/#dense_rank)

PySpark – percent_rank() (https://sparkbyexamples.com/pyspark/pyspark-window-functions/#percent_rank)

PySpark – typedLit() (https://sparkbyexamples.com/pyspark/pyspark-lit-add-literal-constant/#typedlit)

PySpark – from_json() (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/#from_json)

PySpark – to_json() (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/#to_json)

PySpark – json_tuple() (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/#json_tuple)

PySpark – get_json_object() (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/#get_json_object)

PySpark – schema_of_json() (https://sparkbyexamples.com/pyspark/pyspark-json-functions-with-examples/#schema_of_json)

```
+-------------+----------+-----
|employee_name|department|salar
+-------------+----------+-----
|        James|     Sales|  300
|        James|     Sales|  300
|       Robert|     Sales|  410
|         Saif|     Sales|  410
|      Michael|     Sales|  460
|        Maria|   Finance|  300
|        Scott|   Finance|  330
|          Jen|   Finance|  390
|        Kumar| Marketing|  200
|         Jeff| Marketing|  300
+-------------+----------+-----
```

This is the same as the `DENSE_RANK` function in SQL.

2.4 percent_rank Window Function

```
""" percent_rank """
from pyspark.sql.functions impo
df.withColumn("percent_rank",pe
    .show()
```

Yields below output.

```
+-------------+----------+-----
|employee_name|department|salar
+-------------+----------+-----
|        James|     Sales|  300
|        James|     Sales|  300
|       Robert|     Sales|  410
|         Saif|     Sales|  410
|      Michael|     Sales|  460
|        Maria|   Finance|  300
|        Scott|   Finance|  330
|          Jen|   Finance|  390
|        Kumar| Marketing|  200
|         Jeff| Marketing|  300
+-------------+----------+-----
```

This is the same as the `PERCENT_RANK` function in SQL.

2.5 ntile Window Function

`ntile()` window function returns the relative rank of result rows within a window partition. In below example we have used 2 as an argument to ntile hence it returns ranking between 2 values (1 and 2)

```
"""ntile"""
from pyspark.sql.functions impo
df.withColumn("ntile",ntile(2).
    .show()
```

Yields below output.

```
+-------------+----------+-----
|employee_name|department|salar
+-------------+----------+-----
|        James|     Sales|  300
|        James|     Sales|  300
|       Robert|     Sales|  410
|         Saif|     Sales|  410
|      Michael|     Sales|  460
|        Maria|   Finance|  300
|        Scott|   Finance|  330
|          Jen|   Finance|  390
|        Kumar| Marketing|  200
|         Jeff| Marketing|  300
+-------------+----------+-----
```

This is the same as the `NTILE` function in SQL.

3. PySpark Window Analytic functions

3.1 cume_dist Window Function

`cume_dist()` window function is used to get the cumulative distribution of values within a window partition.

This is the same as the `DENSE_RANK` function in SQL.

```
""" cume_dist """
from pyspark.sql.functions impo
df.withColumn("cume_dist",cume_
    .show()
```

```
+-------------+----------+-----
|employee_name|department|salar
+-------------+----------+-----
|        James|     Sales|  300
|        James|     Sales|  300
|       Robert|     Sales|  410
|         Saif|     Sales|  410
|      Michael|     Sales|  460
|        Maria|   Finance|  300
|        Scott|   Finance|  330
|          Jen|   Finance|  390
|        Kumar| Marketing|  200
|         Jeff| Marketing|  300
+-------------+----------+-----
```

3.2 lag Window Function

This is the same as the `LAG` function in SQL.

```
"""lag"""
from pyspark.sql.functions impo
df.withColumn("lag",lag("salary
      .show()
```

```
+-------------+----------+-----
|employee_name|department|salar
+-------------+----------+-----
|        James|     Sales|  300
|        James|     Sales|  300
|       Robert|     Sales|  410
|         Saif|     Sales|  410
|      Michael|     Sales|  460
|        Maria|   Finance|  300
|        Scott|   Finance|  330
|          Jen|   Finance|  390
|        Kumar| Marketing|  200
|         Jeff| Marketing|  300
+-------------+----------+-----
```

3.3 lead Window Function

This is the same as the `LEAD` function in SQL.

```
 """lead"""
from pyspark.sql.functions impo
df.withColumn("lead",lead("sala
    .show()
```

```
+-------------+----------+-----
|employee_name|department|salar
+-------------+----------+-----
|        James|     Sales|  300
|        James|     Sales|  300
|       Robert|     Sales|  410
|         Saif|     Sales|  410
|      Michael|     Sales|  460
|        Maria|   Finance|  300
|        Scott|   Finance|  330
|          Jen|   Finance|  390
|        Kumar| Marketing|  200
|         Jeff| Marketing|  300
+-------------+----------+-----
```

4. PySpark Window Aggregate Functions

In this section, I will explain how to calculate sum, min, max for each department using PySpark SQL Aggregate window functions and `WindowSpec`. When working with Aggregate functions, we don't need to use order by clause.

```
windowSpecAgg  = Window.partiti
from pyspark.sql.functions impo
df.withColumn("row",row_number(
  .withColumn("avg", avg(col("s
  .withColumn("sum", sum(col("s
  .withColumn("min", min(col("s
  .withColumn("max", max(col("s
  .where(col("row")==1).select(
  .show()
```

This yields below output

```
+----------+------+-----+----+-
|department|   avg|  sum| min| 
+----------+------+-----+----+-
|     Sales|3760.0|18800|3000|4
|   Finance|3400.0|10200|3000|3
| Marketing|2500.0| 5000|2000|3
+----------+------+-----+----+-
```

Please refer for more Aggregate Functions (https://sparkbyexamples.com/spark/spark-sql-aggregate-functions/)

Source Code of Window Functions Example

```
import pyspark
from pyspark.sql import SparkSe

spark = SparkSession.builder.ap

simpleData = (("James", "Sales"
    ("Michael", "Sales", 4600),
    ("Robert", "Sales", 4100),
    ("Maria", "Finance", 3000),
    ("James", "Sales", 3000),
    ("Scott", "Finance", 3300),
    ("Jen", "Finance", 3900),
    ("Jeff", "Marketing", 3000)
    ("Kumar", "Marketing", 2000
    ("Saif", "Sales", 4100) \
  )

columns= ["employee_name", "dep

df = spark.createDataFrame(data

df.printSchema()
df.show(truncate=False)

from pyspark.sql.window import
from pyspark.sql.functions impo
windowSpec  = Window.partitionB

df.withColumn("row_number",row_
    .show(truncate=False)

from pyspark.sql.functions impo
df.withColumn("rank",rank().ove
    .show()

from pyspark.sql.functions impo
df.withColumn("dense_rank",dens
    .show()

from pyspark.sql.functions impo
df.withColumn("percent_rank",pe
    .show()

from pyspark.sql.functions impo
df.withColumn("ntile",ntile(2).
    .show()

from pyspark.sql.functions impo
df.withColumn("cume_dist",cume_
   .show()

from pyspark.sql.functions impo
df.withColumn("lag",lag("salary
      .show()
```

```
from pyspark.sql.functions impo
df.withColumn("lead",lead("sala
    .show()

windowSpecAgg  = Window.partiti
from pyspark.sql.functions impo
df.withColumn("row",row_number(
  .withColumn("avg", avg(col("s
  .withColumn("sum", sum(col("s
  .withColumn("min", min(col("s
  .withColumn("max", max(col("s
  .where(col("row")==1).select(
  .show()
```

The complete source code is available at PySpark Examples GitHub (https://github.com/spark-examples/pyspark-examples/blob/master/pyspark-window-functions.py) for reference.

Conclusion

In this tutorial, you have learned what are PySpark SQL Window functions their syntax and how to use them with aggregate function along with several examples in Scala.

References

I would recommend reading Window Functions Introduction (https://databricks.com/blog/2015/07/15/introducing-window-functions-in-spark-sql.html) and SQL Window Functions API (https://github.com/apache/spark/blob/master/sql/core/src/main/scala/org/apache/spark/sql/functions.scala) blogs for a further understanding of Windows functions. Also, refer to SQL Window functions (http://www.sqlservertutorial.net/sql-server-window-functions/) to know window functions from native SQL.

Happy Learning !!

(https://sparkbyexamples.com/pyspark/pyspark-window-functions/?share=facebook&nb=1)

(https://sparkbyexamples.com/pyspark/pyspark-window-functions/?share=reddit&nb=1)

(https://sparkbyexamples.com/pyspark/pyspark-window-functions/?share=pinterest&nb=1)

4

(https://sparkbyexamples.com/pyspark/pyspark-window-functions/?share=tumblr&nb=1)

(https://sparkbyexamples.com/pyspark/pyspark-window-functions/?share=pocket&nb=1)

(https://sparkbyexamples.com/pyspark/pyspark-window-functions/?share=linkedin&nb=1)

(https://sparkbyexamples.com/pyspark/pyspark-window-functions/?share=twitter&nb=1)

TAGS: AGGREGATE FUNCTIONS (HTTPS://SPARKBYEXAMPLES.COM/TAG/AGGREGATE-FUNCTIONS/), ANALYTIC FUNCTIONS (HTTPS://SPARKBYEXAMPLES.COM/TAG/ANALYTIC-FUNCTIONS/), OVER (HTTPS://SPARKBYEXAMPLES.COM/TAG/OVER/), RANK (HTTPS://SPARKBYEXAMPLES.COM/TAG/RANK/), RANKING FUNCTIONS (HTTPS://SPARKBYEXAMPLES.COM/TAG/RANKING-FUNCTIONS/), ROW (HTTPS://SPARKBYEXAMPLES.COM/TAG/ROW/)

(https://sparkbyexamples.com/author/admin/)

NNK (Https://Sparkbyexamples.Com/Author/Admin/)

SparkByExamples.com is a Big Data and Spark examples community page, all examples are simple and easy to understand and well tested in our development environment Read more .. (https://sparkbyexamples.com/about-sparkbyexamples/)

THIS POST HAS 9 COMMENTS

Raymond

25 APR 2021 REPLY

Great post, keep it up

Anonymous

22 FEB 2021 REPLY

This is great, would appreciate, we add more examples for order by (rowsBetween and rangeBetween)

NNK 23 FEB 2021 REPLY

Sure. will do.

BobCG 10 DEC 2020 REPLY

Great job!!!
The same result for Window Aggregate Functions:
df.groupBy('dep').agg(
avg('salary').alias('avg'),
sum('salary').alias('sum'),
min('salary').alias('min'),
max('salary').alias('max')
).select('dep', 'avg', 'sum', 'min', 'max').show()

David 4 FEB 2021 REPLY

The difference would be that with the Window Functions you can append these new columns to the existing DataFrame. If you just group by department you would have the department plus the aggregate values but not the employee name or salary for each one.

Anonymous

8 DEC 2020 REPLY

Awesome explanations.

NNK 10 DEC 2020 REPLY

Thanks.

Ayan 19 OCT 2020 REPLY

Great job.Super easy to comprehend

NNK 19 OCT 2020 REPLY

Thanks for your comment and liking Pyspark window functions.

Anonymous

16 OCT 2020 REPLY

thank you very much

Leave a Reply

Categories

Apache Hadoop (https://sparkbyexamples.com/category/hadoop/)

Apache Spark (https://sparkbyexamples.com/category/spark/)

Apache Spark Streaming (https://sparkbyexamples.com/category/spark/apache-spark-streaming/)

Apache Kafka (https://sparkbyexamples.com/category/kafka/)

Apache HBase (https://sparkbyexamples.com/category/hbase/)

Apache Cassandra (https://sparkbyexamples.com/category/cassandra/)

Snowflake Database (https://sparkbyexamples.com/category/snowflake/)

H2O Sparkling Water (https://sparkbyexamples.com/category/h2o-sparkling-water/)

PySpark (https://sparkbyexamples.com/category/pyspark/)

Recent Posts

Spark regexp_replace() – Replace String Value (https://sparkbyexamples.com/spark/spark-regexp_replace-replace-string-value/)

How to Run a PySpark Script from Python? (https://sparkbyexamples.com/pyspark/run-pyspark-script-from-python-subprocess/)

Spark SQL like() Using Wildcard Example (https://sparkbyexamples.com/spark/spark-sql-like-using-wildcard-example/)

Spark isin() & IS NOT IN Operator Example (https://sparkbyexamples.com/spark/spark-isin-is-not-in-operator-example/)

Spark – Get Size/Length of Array & Map Column (https://sparkbyexamples.com/spark/spark-get-size-length-of-array-map-column/)

Spark Using Length/Size Of a DataFrame Column (https://sparkbyexamples.com/spark/spark-using-length-size-of-a-dataframe-column/)

Spark rlike() Working with Regex Matching Examples (https://sparkbyexamples.com/spark/spark-rlike-regex-matching-examples/)

Spark Check String Column Has Numeric Values (https://sparkbyexamples.com/spark/spark-check-string-column-has-numeric-values/)

Spark Check Column Data Type is Integer or String (https://sparkbyexamples.com/spark/spark-check-column-data-type-is-integer-or-string/)

About SparkByExamples.Com

SparkByExamples.com is a Big Data and Spark examples community page, all examples are simple and easy to understand, and well tested in our development environment Read more .. (https://sparkbyexamples.com/about-sparkbyexamples/)

Follow Us

(https://twitter.com/sparkbyexamples) (https://www.facebook.com/sparkbyexamples/) (https://www.linkedin.com/in/n-nk-b860a8193/) (https://github.com/spark-examples/)