

# PySpark 3.0 Quick Reference Guide

## What is Apache Spark?

- Open Source cluster computing framework
- Fully scalable and fault-tolerant
- Simple API's for **Python**, SQL, Scala, and R
- Seamless streaming and batch applications
- Built-in libraries for data access, streaming, data integration, graph processing, and advanced analytics / machine learning

## Spark Terminology

- **Driver:** the local process that manages the spark session and returned results
- **Workers:** computer nodes that perform parallel computation
- **Executors:** processes on worker nodes that do the parallel computation
- **Action:** is either an instruction to return something to the driver or to output data to a file system or database
- **Transformation:** is anything that isn't an action and are performed in a lazy fashion
- **Map:** indicates operations that can run in a row independent fashion
- **Reduce:** indicates operations that have intra-row dependencies
- **Shuffle:** is the movement of data from executors to run a Reduce operation
- **RDD:** Redundant Distributed Dataset is the legacy in-memory data format
- **DataFrame:** a flexible object oriented data structure that has a row/column schema
- **Dataset:** a DataFrame like data structure that doesn't have a row/column schema

## Spark Libraries

- **ML:** is the machine learning library with tools for statistics, featureization, evaluation, classification, clustering, frequent item mining, regression, and recommendation
- **GraphFrames / GraphX:** is the graph analytics library
- **Structured Streaming:** is the library that handles real-time streaming via micro-batches and unbounded DataFrames

## Spark Data Types

- **Strings**
  - StringType
- **Dates / Times**
  - DateType
  - TimestampType
- **Numeric**
  - DecimalType
  - DoubleType
  - FloatType
  - ByteType
  - IntegerType
  - LongType
  - ShortType
- **Complex Types**
  - ArrayType
  - MapType
  - StructType
  - StructField
- **Other**
  - BooleanType
  - BinaryType
  - NullType (None)

## PySpark Session (spark)

- spark.createDataFrame()
- spark.range()
- spark.streams
- spark.sql()
- spark.table()
- spark.udf()
- spark.version()
- spark.stop()

## PySpark Catalog (spark.catalog)

- cacheTable()
- clearCache()
- createTable()
- createExternalTable()
- currentDatabase
- dropTempView()
- listDatabases()
- listTables()
- listFunctions()
- listColumns()
- isCached()
- recoverPartitions()
- refreshTable()
- refreshByPath()
- registerFunction()
- setCurrentDatabase()
- uncacheTable()

## PySpark Data Sources API

- **Input Reader / Streaming Source (spark.read, spark.readStream)**
  - load()
  - schema()
  - table()
- **Output Writer / Streaming Sink (df.write, df.writeStream)**
  - bucketBy()
  - insertInto()
  - mode()
  - outputMode() # streaming
  - partitionBy()
  - save()
  - saveAsTable()
  - sortBy()
  - start() # streaming
  - trigger() # streaming
- **Common Input / Output**
  - csv()
  - format()
  - jdbc()
  - json()
  - parquet()
  - option(), options()
  - orc()
  - text()

## Structured Streaming

- **StreamingQuery**
  - awaitTermination()
  - exception()
  - explain()
  - foreach()
  - foreachBatch()
  - id
  - isActive
  - lastProgress
  - name
  - processAllAvailable()
  - recentProgress
  - runId
  - status
  - stop()
- **StreamingQueryManager (spark.streams)**
  - active
  - awaitAnyTermination()
  - get()
  - resetTerminated()

## PySpark DataFrame Actions

- **Local (driver) Output**
  - collect()
  - show()
  - toJSON()
  - toLocalIterator()
  - toPandas()
  - take()
  - tail()
- **Status Actions**
  - columns()
  - explain()
  - isLocal()
  - isStreaming()
  - printSchema()
  - dtypes
- **Partition Control**
  - repartition()
  - repartitionByRange()
  - coalesce()

- **Distributed Function**
  - foreach()
  - foreachPartition()

## PySpark DataFrame Transformations

- **Grouped Data**
  - cube()
  - groupBy()
  - pivot()
  - cogroup()
- **Stats**
  - approxQuantile()
  - corr()
  - count()
  - cov()
  - crosstab()
  - describe()
  - freqItems()
  - summary()
- **Column / cell control**
  - drop() # drops columns
  - fillna() #alias to na.fill/replace()
  - select(), selectExpr()
  - withColumn()
  - withColumnRenamed()
  - colRegex()
- **Row control**
  - asc()
  - asc\_nulls\_first()
  - asc\_nulls\_last()
  - desc()
  - desc\_nulls\_first()
  - desc\_nulls\_last()
  - distinct()
  - dropDuplicates()
  - dropna() #alias to na.drop
  - filter()
  - limit()
- **Sorting**
  - asc()
  - asc\_nulls\_first()
  - asc\_nulls\_last()
  - desc()
  - desc\_nulls\_first()
  - desc\_nulls\_last()
  - sort()
  - orderBy()
  - sortWithinPartitions()
- **Sampling**
  - sample()
  - sampleBy()
  - randomSplit()
- **NA (Null/Missing) Transformations**
  - na.drop()
  - na.fill()
  - na.replaced()
- **Caching / Checkpointing / Pipelining**
  - checkpoint()
  - localCheckpoint()
  - persist(), unpersist()
  - withWatermark() # streaming
  - toDF()
  - transform()
- **Joining**
  - broadcast()
  - join()
  - crossJoin()
  - exceptAll()
  - hint()
  - intersect(), intersectAll()
  - subtract()
  - union()
  - unionByName()
- **Python Pandas**
  - apply()
  - pandas\_udf()
  - mapInPandas()
  - applyInPandas()
- **SQL**
  - createGlobalTempView()
  - createOrReplaceGlobalTempView()
  - createOrReplaceTempView()
  - createTempView()
  - registerJavaFunction()
  - registerJavaUDAF()

# PySpark 3.0 Quick Reference Guide

## PySpark DataFrame Functions

### • Aggregations (df.groupBy())

- agg()
- approx\_count\_distinct()
- count()
- countDistinct()
- mean()
- min(), max()
- first(), last()
- grouping()
- grouping\_id()
- kurtosis()
- skewness()
- stddev()
- stddev\_pop()
- stddev\_samp()
- sum()
- sumDistinct()
- var\_pop()
- var\_samp()
- variance()

### • Column Operators

- alias()
- between()
- contains()
- eqNullSafe()
- isNull(), isNotNull()
- isin()
- isnan()
- like()
- rlike()
- getItem()
- getField()
- startsWith(), endsWith()

### • Basic Math

- abs()
- exp(), expm1()
- factorial()
- floor(), ceil()
- greatest(), least()
- pow()
- round(), bround()
- rand()
- random()
- sqrt(), cbrt()
- log(), log2(), log10(), log1p()
- signum()

### • Trigonometry

- cos(), cosh(), acos()
- degrees()
- hypot()
- radians()
- sin(), sinh(), asin()
- tan(), tanh(), atan(), atan2()

### • Multivariate Statistics

- corr()
- covar\_pop()
- covar\_samp()

### • Conditional Logic

- coalesce()
- nanvl()
- otherwise()
- when()

### • Formatting

- format\_string()
- format\_number()

### • Row Creation

- explode(), explode\_outer()
- posexplode(), posexplode\_outer()

### • Schema Inference

- schema\_of\_csv()
- schema\_of\_json()

### • Date & Time

- add\_months()
- current\_date()
- current\_timestamp()
- date\_add(), date\_sub()
- date\_format()
- date\_trunc()
- datediff()
- dayofweek()
- dayofmonth()
- dayofyear()
- from\_unixtime()
- from\_utc\_timestamp()
- hour()
- last\_day(), next\_day()
- minute()
- month()
- months\_between()
- quarter()
- second()
- to\_date()
- to\_timestamp()
- to\_utc\_timestamp()
- trunc()
- unix\_timestamp()
- weekofyear()
- window()
- year()

### • String

- concat()
- concat\_ws()
- format\_string()
- initcap()
- instr()
- length()
- levenshtein()
- locate()
- lower(), upper()
- lpad(), rpad()
- ltrim(), rtrim()
- overlay()
- regexp\_extract()
- regexp\_replace()
- repeat()
- reverse()
- soundex()
- split()
- substring()
- substring\_index()
- translate()
- trim()

### • Hashes

- crc32()
- hash()
- md5()
- sha1(), sha2()
- xxhash64()

### • Special

- col()
- expr()
- input\_file\_name()
- lit()
- monotonically\_increasing\_id()
- spark\_partition\_id()

### • Collections (Arrays & Maps)

- array()
- array\_contains()
- array\_distinct()
- array\_except()
- array\_intersect()
- array\_join()
- array\_max(), array\_min()
- array\_position()
- array\_remove()
- array\_repeat()
- array\_sort()
- array\_union()
- arrays\_overlap()
- arrays\_zip()
- create\_map()
- element\_at()
- flatten()
- map\_concat()
- map\_entries()
- map\_from\_arrays()
- map\_from\_entries()
- map\_keys()
- map\_values()
- sequence()
- shuffle()
- size()
- slice()
- sort\_array()

### • Conversion

- base64(), unbase64()
- bin()
- cast()
- conv()
- encode(), decode()
- from\_avro(), to\_avro()
- from\_csv(), to\_csv()
- from\_json(), to\_json()
- get\_json\_object()
- hex(), unhex()

## PySpark Windowed Aggregates

### • Window Operators

- over()

### • Window Specification

- orderBy()
- partitionBy()
- rangeBetween()
- rowsBetween()

### • Ranking Functions

- ntile()
- percentRank()
- rank(), denseRank()
- row\_number()

### • Analytical Functions

- cume\_dist()
- lag(), lead()

### • Aggregate Functions

- All of the listed aggregate functions

### • Window Specification Example

```
from pyspark.sql.window import Window
windowSpec = \
Window \
.partitionBy(...) \
.orderBy(...) \
.rowsBetween(start, end) # ROW Window Spec
# or
.rangeBetween(start, end) # RANGE Window Spec

# example usage in a DataFrame transformation
df.withColumn('rank',rank(...).over(windowSpec))
```

©WiseWithData 2020-Version 3.0-0622