

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 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, featurization, 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.fillreplace()
 - 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.replace()
- **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()
 - randn()
 - 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