#_ important PySpark Operations [+100]

RDD (Resilient Distributed Dataset) **Operations:**

- parallelize(): Create an RDD.
- map(): Transform each element of the RDD.
- filter(): Return a new RDD with only the elements that satisfy a condition.
- reduce(): Aggregate RDD elements using a function.
- collect(): Return all the elements of the RDD.
- count(): Count the RDD's elements.
- first(): Return the first element of the RDD.
- take(): Return the first 'n' elements of the RDD.
- foreach(): Apply a function to each element of the RDD.
- groupByKey(): Group values with the same key.
- reduceByKey(): Reduce values with the same key using a function.
- sortBy(): Sort the RDD.
- join(): Join two RDDs.
- union(): Return a new RDD that contains the union of the elements in the source RDD and another RDD.

DataFrame **Operations:**

- createDataFrame(): Create a DataFrame from an RDD or list.
- select(): Select specific columns from a DataFrame.
- filter() or where(): Filter rows in a DataFrame.
- groupBy(): Group by a column or columns.
- orderBy() or sort(): Sort by one or more columns.
- drop(): Drop a column.
- withColumn(): Add or replace a column.
- withColumnRenamed(): Rename a column.
- join(): Join two DataFrames.
- describe(): Compute summary statistics.
- dropna(): Drop rows with null values.
- fillna(): Fill null values.
- agg(): Aggregate data after grouping.

By: Waleed Mousa

- distinct(): Return distinct rows.
- limit(): Limit the number of rows.

SparkSQL **Operations**:

- spark.sql(): Execute SQL queries.
- createOrReplaceTempView(): Create a temporary view.
- createGlobalTempView(): Create a global temporary view.

Data Sources and Writing Data:

- read.csv(): Read data from a CSV file.
- write.csv(): Write data to a CSV file.
- read.json(): Read data from a JSON file.
- write.json(): Write data to a JSON file.
- read.parquet(): Read data from a Parquet file.
- write.parquet(): Write data to a Parquet file.
- read.jdbc(): Read data from a JDBC source.
- write.jdbc(): Write data to a JDBC source.

MLlib - Machine Learning **Library**:

- VectorAssembler(): Assemble feature vectors.
- StringIndexer(): Convert string columns to numeric.
- OneHotEncoder(): One-hot encode categorical features.
- StandardScaler(): Scale features.
- LinearRegression(): Linear regression model.
- DecisionTreeClassifier(): Decision tree classification model.
- KMeans(): K-means clustering.
- CrossValidator(): Cross-validation for model selection.
- TrainValidationSplit(): Train-validation for hyperparameter tuning.

GraphX **Operations**:

- Graph(): Create a graph.
- vertices: Access vertices of a graph.
- edges: Access edges of a graph.

By: Waleed Mousa

- `triplets`: Access triplets of a graph.
- `inDegrees`: Compute the in-degree of each vertex.
- `outDegrees`: Compute the out-degree of each vertex.
- `subgraph()`: Generate a subgraph.
- `mapVertices()`: Transform the vertices of a graph.
- `mapEdges()`: Transform the edges of a graph.

Streaming:

- `StreamingContext()`: Create a streaming context.
- `updateStateByKey()`: Maintain stateful information.
- `window()`: Return a new DStream computed based on windowed batches.
- `reduceByKeyAndWindow()`: Reduce by key over a window.

Performance and Optimization:

- `cache() or persist()`: Cache an RDD or DataFrame.
- `unpersist()`: Remove data from memory.
- `broadcast()`: Broadcast a read-only variable.
- `repartition()`: Repartition the data.
- `coalesce()`: Decrease the number of partitions.

Utility Functions:

- `udf()`: Create a user-defined function.
- `lit()`: Create a column of literal value.
- `when()`: Evaluate a condition.

Statistics and Linear Algebra (MLlib):

- `Statistics.colStats()`: Column statistics.
- `Statistics.corr()`: Correlation between two series.
- `DenseVector()`: Create a dense vector.
- `SparseVector()`: Create a sparse vector.
- `RowMatrix()`: Create a row matrix.

Advanced Features:

- windowSpec(): Define a window specification.
- over(): Apply a window specification.
- lead() and lag(): Lead and lag functions in window operations.
- pivot(): Pivot data.
- explode(): Transform array or map column into multiple rows.

Other Functions and Methods:

- functions.concat(): Concatenate two or more columns.
- functions.substring(): Extract a substring.
- functions.year() and functions.month(): Extract year and month.
- functions.dayofyear() and functions.dayofmonth(): Extract day.
- functions.round(): Round numbers.
- functions.length(): Compute the length of a string.
- functions.size(): Compute the size of a list or map.
- functions.isnan(): Check for NaN values.
- functions.isnull(): Check for NULL values.
- functions.rand(): Generate random values.
- functions.split(): Split a string.
- functions.array(): Create an array.
- functions.array_contains(): Check if an array contains a value.
- functions.map(): Create a map.
- functions.map_keys() and functions.map_values(): Access keys and values of a map.
- functions.struct(): Create a struct.
- functions.from_json() and functions.to_json(): Work with JSON.
- functions.current_date() and functions.current_timestamp(): Get current date and time.
- functions.date_add() and functions.date_sub(): Add or subtract days from a date.
- functions.date_diff(): Compute difference between two dates.
- SparkContext.addFile() and SparkFiles.get(): Distributing auxiliary files (e.g., Python files, data files) required by tasks.