Spark DataFrame Operations

Explanations & Documentation Links

Let's not get lost in the Spark DataFrame jungle! Here's a handy list of the most essential operations—including the ones we've covered in the tutorial and a few extra gems to level up your skills.

Basic Operations

Operation	Explanation	How to use	Documentation link
select	Selects specific columns from a DataFrame	`df.select('column_name')` `df.select(['col1', 'col2'])`	.select()
drop	Removes one or more columns	`df.drop('col_name')` `df.drop('col1', 'col2')`	.drop()
col	Refers to a column when used in expressions.		.col()
withColumn	Adds a new column to the DataFrame	`df.withColumn('new_col', some_expression)`	.withColumn()
withColumnRenamed	Renames a column	`df.withColumnRenamed('new_name', 'old_name')`	.withColumnRenamed()
filter	Filters rows based on a condition.	`df.filter(col('col_name') == 'some_value')` `df.filter(col('col_name') > 10)`	.filter()
alias	Assigns an alias (temporary name) to a column.	`df.select(col('col_name').alias('new_name'))`	.alias()
orderBy	Sorts rows based on a column (ascending by default)	`df.orderBy(col('col_name').desc())`	.orderBy()
sort	Sorts rows based on a column (ascending by default)	`df.sort(col('col_name').desc())`	.sort()
limit	Return only a specific number of rows.	`df.limit(10)`	.limit()

dropDuplicates	Removes duplicate rows from the DataFrame (can be based on columns)	`df.dropDuplicates()`
		`#drops all duplicate rows`
		`df.dropDuplicates(['col_name'])`
		`#drops duplicates based on a single column`
			.dropDuplicates()

fillna	Replaces null values with a specified value.	`df.fillna(value)`
		
		`df.fillna({'col1': 'some value'})`
			.fillna()

distinct	Returns a new DataFrame containing distinct rows	`df.distinct()`	.distinct()

Aggregations & Grouping

Operation	Explanation	How to use	Documentation link
groupBy	Groups the DataFrame by one or more columns	`df.groupBy('col_name')`	.groupBy()
agg	Performs aggregation on grouped data		.agg()

count	Counts the number of rows	`df.count()`
		`#counts all rows`
		`df.groupBy('col_name').count()`
		`#counts the num of rows per group`
			.count()

sum	Computes the sum of a column.	`df.groupBy('col_name').sum('value')`	.sum()
avg	Computes the average of a column.	`df.groupBy('col_name').avg('value')`	.avg()
min	Finds the minimum value of a column.	`df.groupBy('col_name').min('value')`	.min()
max	Finds the maximum value of a column.	`df.groupBy('col_name').max('value')`	.max()

Notes:

- **.groupBy()** must be followed by an **aggregation function** (e.g., .count(), .sum(), .avg()).
- **.agg()** allows multiple aggregations at once

Joins & Set Operations

Operation	Explanation	How to use	Documentation link
join	Performs inner, left, right, or full joins between DataFrames.	`df1.join(df2, 'col_name', how = 'join type')`	.join()
union	Combines DataFrames (must have the same schema).	`df1.union(df2)`	.union()
intersect	Gets common rows between two DataFrames.	`df1.intersect(df2)`	.intersect()
except	Get rows present in one DataFrame but not the other.	`df1.except(df2)`	.except()

Date & Time Functions

Operation	Explanation	How to use	Documentation link
to_date	Converts a string column to a date.	`df.withColumn('date_col', to_date(col('string_col'), 'yyyy-MM-dd'))`	.to_date()

year / month / dayofweek	Extract year, month, or day of the week from a date.	`df.withColumn('year', year(col('date_col')))`
		
		`df.withColumn('month', month(col('date_col')))`
		
		`df.withColumn('day', dayofweek(col('date_col')))`
			.year()
			
			.month()
			
			.dayofweek()

datediff	Calculate the difference between two dates.	`df.withColumn('date_diff', datediff(col('end_date'), col('start_date')))`	.datediff()