Module 3 Cheat Sheet: Apache Spark

Package/Method	Description	Code Example

appName()	A name for your job to display on the cluster web UI.	`from pyspark.sql import SparkSession`
		`spark = SparkSession.builder.appName("MyApp").getOrCreate()`
		Copied!

cache()	An Apache Spark transformation often used on a DataFrame, data set, or RDD when you want to perform multiple actions. cache() caches the specified DataFrame, data set, or RDD in the memory of your cluster's workers. Since cache() is a transformation, the caching operation takes place only when a Spark action (for example, count(), show(), take(), or write()) is also used on the same DataFrame, data set, or RDD in a single action.	`df = spark.read.csv("customer.csv")`
		`df.cache()`
		Copied!

count()	Returns the number of elements with the specified value.	`count = df.count()`
		`print(count)`
		Copied!

createTempView()	Creates a temporary view that can	`df.createOrReplaceTempView("cust_tbl")`

	later be used to query the data. The only required parameter is the name of the view.	Copied!
filter()	Returns an iterator where the items are filtered through a function to test if the item is accepted or not.	`filtered_df = df.filter(df['age'] > 30)` Copied!
getOrCreate()	Get or instantiate a SparkContext and register it as a singleton object.	`spark = SparkSession.builder.getOrCreate()` Copied!
import	Used to make code from one module accessible in another. Python imports are crucial for a successful code structure. You may reuse code and keep your projects manageable by using imports effectively, which can increase your productivity.	`from pyspark.sql import SparkSession` Copied!

len()	Returns the number of items in an object. When the object is a string, the len() function returns the number of characters in the string.	`row_count = len(df.collect())`
		`print(row_count)`
		Copied!

map()	Returns a map object (an iterator) of the results after applying the given function to each item of a given	`rdd = df.rdd.map(lambda row: (row['name'],`
		`row['age']))`
		Copied!

	iterable (list, tuple, etc.)	
pip	To ensure that requests will function, the pip program searches for the package in the Python Package Index (PyPI), resolves any dependencies, and installs everything in your current Python environment.	`pip list` Copied!
pip install	The pip install <package> command looks for the latest version of the package and installs it.	`pip install pyspark` Copied!
print()	Prints the specified message to the screen or other standard output device. The message can be a string or any other object; the object will be converted into a string before being written to the screen.	`print("Hello, PySpark!")` Copied!
printSchema()	Used to print or display the schema of the DataFrame or data set in tree format along with the column name and data type. If you have a DataFrame or data set with a nested structure, it displays the schema in a nested tree format.	`df.printSchema()` Copied!
sc.parallelize()	Creates a parallelized	

	collection. Distributes a local Python collection to form an RDD. Using range is recommended if the input represents a range for performance.	`rdd = sc.parallelize([1, 2, 3, 4, 5])` Copied!
select()	Used to select one or multiple columns, nested columns, column by index, all columns from the list, by regular expression from a DataFrame. select() is a transformation function in Spark and returns a new DataFrame with the selected columns.	`selected_df = df.select('name', 'age')` Copied!
show()	Spark DataFrame show() is used to display the contents of the DataFrame in a table row and column format . By default, it shows only twenty rows, and the column values are truncated at twenty characters.	`df.show()` Copied!
spark.read.json	Spark SQL can automatically infer the schema of a JSON data set and load it as a DataFrame. The read.json() function loads data from a directory of	`json_df = spark.read.json("customer.json")` Copied!

	JSON files where each line of the files is a JSON object. Note that the file offered as a JSON file is not a typical JSON file.	

spark.sql()	To issue any SQL query, use the sql() method on the SparkSession instance . All spark.sql queries executed in this manner return a DataFrame on which you may perform further Spark operations if required.	`result = spark.sql("SELECT name, age FROM cust_tbl WHERE age > 30")`
		`result.show()`
		Copied!

time()	Returns the current time in the number of seconds since the Unix Epoch.	`from pyspark.sql.functions import current_timestamp`
		`current_time = df.select(current_timestamp().alias("current_time"))`
		`current_time.show()`
		Copied!

Changelog

Date	Version	Changed by	Change Description
2023-09-06	1.0	Sameeksha Saxena	Initial version created

IBM Corporation 2023. All rights reserved.