

Module 4 Cheat Sheet: DataFrames and Spark SQL

Package/Method	Description	Code Example
appName()	A name for your job to display on the cluster web UI.	<div>1. 1</div> <div>2. 2</div> <div>1. from pyspark.sql import SparkSession</div> <div>2. spark = SparkSession.builder.appName("MyApp").getOrCreate()</div> <div>Copied!</div> <div>1. 1</div> <div>2. 2</div> <div>3. 3</div> <div>4. 4</div> <div>1. from pyspark.sql import SparkSession</div> <div>2. spark = SparkSession.builder.appName("MyApp").getOrCreate()</div> <div>3. data = [("Jhon", 30), ("Peter", 25), ("Bob", 35)]</div> <div>4. columns = ["name", "age"]</div>
createDataFrame()	Used to load the data into a Spark DataFrame.	<div>Copied!</div> <div>Creating a DataFrame</div> <div>1. 1</div> <div>1. df = spark.createDataFrame(data, columns)</div> <div>Copied!</div>
createTempView()	Create a temporary view that can later be used to query the data. The only required parameter is the name of the view.	<div>1. 1</div> <div>1. df.createOrReplaceTempView("cust_tbl")</div> <div>Copied!</div> <div>Replace NULL/None values in a DataFrame</div>
fillna()	Used to replace NULL/None values on all or selected multiple DataFrame columns with either zero (0), empty string, space, or any constant literal values.	<div>1. 1</div> <div>1. filled_df = df.fillna(0)</div> <div>Copied!</div> <div>Replace with zero</div>
filter()	Returns an iterator where the items are filtered through a function to test if the item is accepted or not.	<div>1. 1</div> <div>1. filtered_df = df.filter(df['age'] > 30)</div> <div>Copied!</div>
getOrCreate()	Get or instantiate a SparkContext and register it as a singleton object.	<div>1. 1</div> <div>1. spark = SparkSession.builder.getOrCreate()</div> <div>Copied!</div> <div>Grouping data and performing aggregation</div>
groupby()	Used to collect the identical data into groups on DataFrame and perform count, sum, avg, min, max functions on the grouped data.	<div>1. 1</div> <div>1. grouped_df = df.groupBy("age").agg({"age": "count"})</div> <div>Copied!</div> <div>Returning the first 5 rows</div>
head()	Returns the first <i>n</i> rows for the object based on position.	<div>1. 1</div> <div>1. first_5_rows = df.head(5)</div> <div>Copied!</div>
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.	<div>1. 1</div> <div>1. from pyspark.sql import SparkSession</div> <div>Copied!</div> <div>1. 1</div> <div>1. import pandas as pd</div> <div>Copied!</div>
pd.read_csv()	Required to access data from the CSV file from Pandas that retrieves data in the form of the data frame.	<div>Reading data from a CSV file into a DataFrame</div> <div>1. 1</div> <div>1. df_from_csv = pd.read_csv("data.csv")</div> <div>Copied!</div>
pip	To ensure that requests will function, the pip program searches for the package in the Python Package Index (PyPI), resolves any dependencies,	<div>1. 1</div> <div>1. pip list</div>

Package/Method	Description	Code Example
	and installs everything in your current Python environment.	<div>Copied!</div> <pre>1. 1</pre>
pip install	The pip install <package> command looks for the latest version of the package and installs it.	<div>Copied!</div> <pre>1. pip install pyspark</pre>
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.	<div>Copied!</div> <pre>1. 1 1. df.printSchema()</pre>
		<div>Copied!</div> <pre>1. 1 1. import pandas as pd</pre>
		Create a sample DataFrame
		<div>Copied!</div> <pre>1. 1 2. 2 1. data = {'A': [1, 2, 3], 'B': [4, 5, 6]} 2. df = pd.DataFrame(data)</pre>
rename()	Used to change the row indexes and the column labels.	<div>Copied!</div> <div>Rename columns</div> <pre>1. 1 1. df = df.rename(columns={'A': 'X', 'B': 'Y'})</pre> <div>Copied!</div> <div>The columns 'A' and 'B' are now renamed to 'X' and 'Y'</div> <pre>1. 1 1. print(df)</pre> <div>Copied!</div>
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.	<div>Copied!</div> <pre>1. 1 1. selected_df = df.select('name', 'age')</pre>
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.	<div>Copied!</div> <pre>1. 1 1. df.show()</pre>
sort()	Used to sort DataFrame by ascending or descending order based on single or multiple columns.	<div>Copied!</div> <div>Sorting DataFrame by a column in ascending order</div> <pre>1. 1 1. sorted_df = df.sort("age")</pre> <div>Copied!</div> <div>Sorting DataFrame by multiple columns in descending order</div> <pre>1. 1 1. sorted_df_desc = df.sort(["age", "name"], ascending=[False, True])</pre> <div>Copied!</div> <pre>1. 1 1. from pyspark import SparkContext</pre>
SparkContext()	It is an entry point to Spark and is defined in org.apache.spark package since version 1.x and used to programmatically create Spark RDD, accumulators, and broadcast variables on the cluster.	<div>Copied!</div> <div>Creating a SparkContext</div> <pre>1. 1 1. sc = SparkContext("local", "MyApp")</pre>
SparkSession	It is an entry point to Spark, and creating a SparkSession instance would be the first statement you would write to the program with RDD, DataFrame, and dataset	<div>Copied!</div> <pre>1. 1 1. from pyspark.sql import SparkSession</pre> <div>Copied!</div>

Package/Method	Description	Code Example
		Creating a SparkSession
		<pre>1. 1 1. spark = SparkSession.builder.appName("MyApp").getOrCreate()</pre> <div>Copied!</div>
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 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.	<pre>1. 1 1. json_df = spark.read.json("customer.json")</pre> <div>Copied!</div>
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.	<pre>1. 1 2. 2 1. result = spark.sql("SELECT name, age FROM cust_tbl WHERE age > 30") 2. result.show()</pre> <div>Copied!</div>
		Registering a UDF (User-defined Function)
spark.udf.register()	In PySpark DataFrame, it is used to register a user-defined function (UDF) with Spark, making it accessible for use in Spark SQL queries. This allows you to apply custom logic or operations to DataFrame columns using SQL expressions.	<pre>1. 1 2. 2 3. 3 4. 4 5. 5 1. from pyspark.sql.functions import udf 2. from pyspark.sql.types import StringType 3. def my_udf(value): 4. return value.upper() 5. spark.udf.register("my_udf", my_udf, StringType())</pre> <div>Copied!</div>
		Filtering rows based on a condition
where()	Used to filter the rows from DataFrame based on the given condition. Both filter() and where() functions are used for the same purpose.	<pre>1. 1 1. filtered_df = df.where(df['age'] > 30)</pre> <div>Copied!</div>
		Adding a new column and performing transformations
withColumn()	Transformation function of DataFrame used to change the value, convert the data type of an existing column, create a new column, and many more.	<pre>1. 1 2. 2 1. from pyspark.sql.functions import col 2. new_df = df.withColumn("age_squared", col("age") ** 2)</pre> <div>Copied!</div>
		Renaming an existing column
withColumnRenamed()	Returns a new DataFrame by renaming an existing column.	<pre>1. 1 1. renamed_df = df.withColumnRenamed("age", "years_old")</pre> <div>Copied!</div>



Skills Network