Module 6 Cheat Sheet: Monitoring and Tuning

Package/Method	Description	Code Example
agg()	Used to get the aggregate values like count, sum, avg, min, and max for each group.	`agg_df = df.groupBy("column_name").agg({"column_to_aggregate": "sum"})`
cache()	Apache Spark transformation that is often used on a DataFrame, data set, or RDD when you want to perform more than one action. 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, Dataset, or RDD in a single action.	`df = spark.read.csv("customer.csv")` `df.cache()`

cd	Used to move efficiently from the existing working directory to different directories on your system.	Basic syntax of the cd command:
		`cd [options]… [directory]`
		Example 1: Change directory location to folder1.
		`cd /usr/local/folder1`
		Example 2: Get back to the previous working directory.
		`cd -`
		Example 3: Move up one level from the present working directory tree.
		`cd ..`

def	Used to define a function. It is placed before a function name that is provided by the user to create a user-defined function.	`def greet(name):`
		This function takes a name as a parameter and prints a greeting.
		`print(f"Hello, {name}!")`
		Calling the function:
		`greet("John")`

docker exec	Runs a new command in a running container. Only runs while the container's primary process is running, and it is not restarted	`docker exec -it container_name command_to_run` `docker exec -it my_container /bin/bash`

Package/Method	Description	Code Example
	if the container is restarted.	

docker rm	Used to remove one or more containers.	To remove a single container by name or ID:
		`docker rm container_name_or_id`
		To remove multiple containers by specifying their names or IDs:
		`docker rm container1_name_or_id container2_name_or_id`
		To remove all stopped containers:
		`docker rm $(docker ps -aq)`

docker run	It runs a command in a new container, getting the image and starting the container if needed.	`docker run [OPTIONS] IMAGE [COMMAND] [ARG...]`

for	The *for* loop operates on lists of items. It repeats a set of commands for every item in a list.	`fruits = ["apple", "banana", "cherry"]`
		Iterating through the list using a *for* loop for fruit in fruits:
		`print(f"I like {fruit}s")`

groupby()	Used to collect the identical data into groups on DataFrame and perform count, sum, avg, min, max functions on the grouped data.	`import pandas as pd`
		Sample DataFrame:
		`data = {'Category': ['A', 'B', 'A', 'B', 'A', 'B'],`
		`'Value': [10, 20, 15, 25, 30, 35]}`
		`df = pd.DataFrame(data)`
		Grouping by "Category" and performing aggregation operations:
		`grouped = df.groupby('Category').agg({'Value': ['count', 'sum', 'mean', 'min', 'max']})`
		`print(grouped)`

repartition()	Used to increase or decrease the RDD or DataFrame partitions by number of partitions or by a single column name or multiple column names.	Create a sample DataFrame:
		`data = [("John", 25), ("Peter", 30), ("Julie", 35), ("David", 40), ("Eva", 45)]`
		`columns = ["Name", "Age"]`
		`df = spark.createDataFrame(data, columns)`
		Show the current number of partitions.
		`print("Number of partitions before repartitioning: ", df.rdd.getNumPartitions())`
		Repartition the DataFrame to 2 partitions.
		`df_repartitioned = df.repartition(2)`
		Show the number of partitions after repartitioning.
		`print("Number of partitions after repartitioning: ", df_repartitioned.rdd.getNumPartitions())`
		Stop the SparkSession.
		`spark.stop()`

return	Used to end the execution of the function call and returns the result (value of the expression following the return keyword) to the caller.	`def add_numbers(a, b):`
		`result = a + b`
		`return result`
		Calling the function and capturing the returned value:
		`sum_result = add_numbers(5, 6)`
		Printing the result.
		`print("The sum is:", sum_result)`
		Output.
		`The sum is: 11`

show()	Spark DataFrame show() is used to display the contents of the DataFrame in a	`df.show()`

Package/Method	Description	Code Example
	table row and column format. By default, it shows only 20 rows, and the column values are truncated at 20 characters.	

spark.read.csv("path")	Using this, you can read a CSV file with fields delimited by pipe, comma, tab (and many more) into a Spark DataFrame.	`from pyspark.sql import SparkSession`
		Create a SparkSession.
		`spark = SparkSession.builder.appName("CSVReadExample").getOrCreate()`
		Read a CSV file into a Spark DataFrame.
		`df = spark.read.csv("path_to_csv_file.csv", header=True, inferSchema=True)`
		Show the first few rows of the DataFrame.
		`df.show()`
		Stop the SparkSession.
		`spark.stop()`

wget	Stands for web get. The wget is a free noninteractive file downloader command. Noninteractive means that it can work in the background when the user is not logged in.	Basic syntax of the wget command; commonly used options are `[-V], [-h], [-b], [-e], [-o], [-a], [-q]`
		`wget [options]… [URL]…`
		Example 1: Specifies to download file.txt over HTTP website URL into the working directory.
		`wget http://example.com/file.txt`
		Example 2: Specifies to download the archive.zip over HTTP website URL in the background and returns you to the comn the interim.
		`wget -b http://www.example.org/files/archive.zip`

withColumn()	Transformation function of DataFrame which is used to change the value, convert the datatype of an existing column, create a new column, and many more.	Sample DataFrame:
		`data = [("John", 25), ("Peter", 30), ("David", 35)]`
		`columns = ["Name", "Age"]`
		`df = spark.createDataFrame(data, columns)`
		Using withColumn to create a new column and change values
		`updated_df = df \`
		`.withColumn("DoubleAge", col("Age") * 2) # Create a new column "DoubleAge" by doubling the`
		`updated_df = updated_df \`
		`.withColumn("AgeGroup", when(col("Age") <= 30, "Young")`
		`.when((col("Age") > 30) & (col("Age") <= 40), "Middle-aged")`
		`.otherwise("Old")) # Create a new column "AgeGroup" based on conditions`
		`updated_df.show()`
		Stop the SparkSession.
		`spark.stop()`

Skills Network