**SPARK CHEAT SHEET**

**Basic**

Command: appName()

Description: A name for your job to display on the cluster web UI.

Example:

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("MyApp").getOrCreate()

Command: cache()

Description: Caches the specified DataFrame, data set, or RDD in the memory of your cluster's workers. The caching operation takes place only when a Spark action (e.g., `count()`, `show()`) is also used on the same DataFrame, data set, or RDD.

Example:

df = spark.read.csv("customer.csv")

df.cache()

Command: count()

Description: Returns the number of elements with the specified value.

Example:

count = df.count()

print(count)

Command: createTempView()

Description: Creates a temporary view that can later be used to query the data.

Example:

df.createOrReplaceTempView("cust\_tbl")

Command: filter()

Description: Returns an iterator where the items are filtered through a function to test if the item is accepted or not.

Example:

filtered\_df = df.filter(df['age'] > 30)

Command: getOrCreate()

Description: Get or instantiate a SparkContext and register it as a singleton object.

Example:

spark = SparkSession.builder.getOrCreate()

Command: len()

Description: Returns the number of items in an object.

Example:

row\_count = len(df.collect())

print(row\_count)

Command: map()

Description: Returns a map object after applying the given function to each item of a given iterable.

Example:

rdd = df.rdd.map(lambda row: (row['name'], row['age']))

Command: printSchema()

Description: Prints the schema of the DataFrame in a tree format, including column names and data types.

Example:

df.printSchema()

Command: sc.parallelize()

Description: Creates a parallelized collection and distributes it to form an RDD.

Example:

rdd = sc.parallelize([1, 2, 3, 4, 5])

Command: select()

Description: Selects one or multiple columns from a DataFrame.

Example:

selected\_df = df.select('name', 'age')

Command: show()

Description: Displays the contents of the DataFrame in a table format.

Example:

df.show()

Command: spark.read.json

Description: Loads data from a directory of JSON files into a DataFrame.

Example:

json\_df = spark.read.json("customer.json")

Command: spark.sql()

Description: Issues an SQL query using the SparkSession instance and returns a DataFrame.

Example:

result = spark.sql("SELECT name, age FROM cust\_tbl WHERE age > 30")

result.show()

Command: time()

Description: Returns the current time in seconds since the Unix Epoch.

Example:

from pyspark.sql.functions import current\_timestamp

current\_time = df.select(current\_timestamp().alias("current\_time"))

current\_time.show()

**Spark SQL**

Command: createDataFrame()

Description: Used to load the data into a Spark DataFrame.

Example:

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("MyApp").getOrCreate()

data = [("Jhon", 30), ("Peter", 25), ("Bob", 35)]

columns = ["name", "age"]

df = spark.createDataFrame(data, columns)

Command: fillna()

Description: 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.

Example:

filled\_df = df.fillna(0)

Command: groupby()

Description: Used to collect identical data into groups on DataFrame and perform count, sum, avg, min, max functions on the grouped data.

Example:

grouped\_df = df.groupBy("age").agg({"age": "count"})

Command: head()

Description: Returns the first n rows for the object based on position.

Example:

first\_5\_rows = df.head(5)

Command: rename()

Description: Used to change the row indexes and the column labels.

Example:

import pandas as pd

data = {'A': [1, 2, 3], 'B': [4, 5, 6]}

df = pd.DataFrame(data)

df = df.rename(columns={'A': 'X', 'B': 'Y'})

print(df)

Command: sort()

Description: Used to sort DataFrame by ascending or descending order based on single or multiple columns.

Example:

sorted\_df = df.sort("age")

sorted\_df\_desc = df.sort(["age", "name"], ascending=[False, True])

Command: SparkContext()

Description: Entry point to Spark to create Spark RDD, accumulators, and broadcast variables on the cluster.

Example:

from pyspark import SparkContext

sc = SparkContext("local", "MyApp")

Command: spark.sql()

Description: Use the sql() method on the SparkSession instance to issue SQL queries.

Example:

result = spark.sql("SELECT name, age FROM cust\_tbl WHERE age > 30")

result.show()

Command: spark.udf.register()

Description: Used to register a user-defined function (UDF) with Spark, allowing SQL expressions in DataFrame columns.

Example:

from pyspark.sql.functions import udf

from pyspark.sql.types import StringType

def my\_udf(value):

return value.upper()

spark.udf.register("my\_udf", my\_udf, StringType())

Command: withColumn()

Description: Transformation function to change the value, convert the data type, or create a new column.

Example:

from pyspark.sql.functions import col

new\_df = df.withColumn("age\_squared", col("age") \*\* 2)

Command: withColumnRenamed()

Description: Returns a new DataFrame by renaming an existing column.

Example:

renamed\_df = df.withColumnRenamed("age", "years\_old")

**Development and Runtime Environment Options**

Command: sc.setloglevel()

Description: Using this method, you can change the log level to the desired level. Valid log levels include ALL, DEBUG, ERROR, FATAL, INFO, OFF, TRACE, and WARN.

Example:

from pyspark import SparkContext

sc = SparkContext("local", "LogLevelExample")

sc.setLogLevel("INFO")

Command: setMaster()

Description: Denotes where to run your Spark application, local or cluster. When you run on a cluster, you need to specify the address of the Spark master or Driver URL for a distributed cluster. We usually assign a local[\*] value to setMaster() in Spark while doing internal testing.

Example:

from pyspark import SparkContext

sc = SparkContext("local[\*]", "MyApp")

**Monitoring and Tuning**

Command: agg()

Description: Used to get the aggregate values like count, sum, avg, min, and max for each group.

Example:

agg\_df = df.groupBy("column\_name").agg({"column\_to\_aggregate": "sum"})

Command: repartition()

Description: Used to increase or decrease the RDD or DataFrame partitions by number of partitions or by a single column name or multiple column names.

Example:

data = [("John", 25), ("Peter", 30), ("Julie", 35), ("David", 40), ("Eva", 45)]

columns = ["Name", "Age"]

df = spark.createDataFrame(data, columns)

print("Number of partitions before repartitioning: ", df.rdd.getNumPartitions())

Repartition the DataFrame to 2 partitions.

df\_repartitioned = df.repartition(2)

Command: withColumn()

Description: 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.

Example:

updated\_df = df \

.withColumn("DoubleAge", col("Age") \* 2)

updated\_df = updated\_df \

.withColumn("AgeGroup", when(col("Age") <= 30, "Young")

.when((col("Age") > 30) & (col("Age") <= 40), "Middle-aged")

.otherwise("Old"))

updated\_df.show()