Hoja de trucos del Módulo 4: DataFrames y Spark SQL

Paquete/Método	Descripción	Ejemplo de Código
appName()	Un nombre para tu trabajo que se mostrará en la interfaz web del clúster.	`from pyspark.sql import SparkSession` `spark = SparkSession.builder.appName("MyApp").getOrCreate()`

createDataFrame()	Se utiliza para cargar los datos en un DataFrame de Spark.	`from pyspark.sql import SparkSession`
		`spark = SparkSession.builder.appName("MyApp").getOrCreate()`
		`data = [("Jhon", 30), ("Peter", 25), ("Bob", 35)]`
		`columns = ["name", "age"]`
		Creando un DataFrame
		`df = spark.createDataFrame(data, columns)`

createTempView()	Crea una vista temporal que se puede usar más tarde para consultar los datos. El único parámetro requerido es el nombre de la vista.	`df.createOrReplaceTempView("cust_tbl")`
fillna()	Se utiliza para reemplazar valores NULL/None en todas o en múltiples columnas seleccionadas del DataFrame con cero (0), cadena vacía, espacio, o cualquier valor literal constante.	Reemplazar valores NULL/None en un DataFrame `filled_df = df.fillna(0)` Reemplazar con cero
filter()	Devuelve un iterador donde los elementos se filtran a través de una función para probar si el elemento es aceptado o no.	`filtered_df = df.filter(df['age'] > 30)`

Paquete/Método	Descripción	Ejemplo de Código
getOrCreate()	Obtiene o instancia un SparkContext y lo registra como un objeto singleton.	`spark = SparkSession.builder.getOrCreate()`
groupby()	Se utiliza para agrupar los datos idénticos en grupos en el DataFrame y realizar funciones de conteo, suma, promedio, mínimo, máximo sobre los datos agrupados.	Agrupando datos y realizando agregaciones `grouped_df = df.groupBy("age").agg({"age": "count"})`
head()	Devuelve las primeras *n* filas del objeto según la posición.	Devolviendo las primeras 5 filas `first_5_rows = df.head(5)`
import	Se utiliza para hacer que el código de un módulo sea accesible en otro. Las importaciones en Python son cruciales para una estructura de código exitosa. Puedes reutilizar código y mantener tus proyectos manejables utilizando importaciones de manera efectiva, lo que puede aumentar tu productividad.	`from pyspark.sql import SparkSession`

pd.read_csv()	Requerido para acceder a datos desde un archivo CSV de Pandas que recupera datos en forma de DataFrame.	`import pandas as pd`
		Leyendo datos de un archivo CSV en un DataFrame
		`df_from_csv = pd.read_csv("data.csv")`

pip	Para asegurar que las solicitudes funcionen, el programa pip busca el paquete en el Índice de Paquetes de Python (PyPI), resuelve cualquier dependencia e instala todo en tu entorno actual de Python.	`pip list`

Paquete/Método	Descripción	Ejemplo de Código
pip install	El comando pip install <package> busca la versión más reciente del paquete e instala.	`pip install pyspark`
printSchema()	Se utiliza para imprimir o mostrar el esquema del DataFrame o conjunto de datos en formato de árbol junto con el nombre de la columna y el tipo de dato. Si tienes un DataFrame o conjunto de datos con una estructura anidada, muestra el esquema en un formato de árbol anidado.	`df.printSchema()`

rename()	Se utiliza para cambiar los índices de fila y las etiquetas de columna.	`import pandas as pd`
		Crear un DataFrame de ejemplo
		`data = {'A': [1, 2, 3], 'B': [4, 5, 6]}`
		`df = pd.DataFrame(data)`
		Renombrar columnas
		`df = df.rename(columns={'A': 'X', 'B': 'Y'})`
		Las columnas 'A' y 'B' ahora se han renombrado a 'X' y 'Y'
		`print(df)`

select()	Se utiliza para seleccionar una o múltiples columnas, columnas anidadas, columna por índice, todas las columnas de la lista, por expresión regular de un DataFrame. select() es una función de transformación en Spark y	`selected_df = df.select('name', 'age')`

Paquete/Método	Descripción	Ejemplo de Código
	devuelve un nuevo DataFrame con las columnas seleccionadas.	
show()	El método show() de Spark DataFrame se utiliza para mostrar el contenido del DataFrame en formato de tabla con filas y columnas. Por defecto, muestra solo veinte filas, y los valores de las columnas se truncarán en veinte caracteres.	`df.show()`

sort()	Se utiliza para ordenar el DataFrame en orden ascendente o descendente según una o múltiples columnas.	Ordenando el DataFrame por una columna en orden ascendente
		`sorted_df = df.sort("age")`
		Ordenando el DataFrame por múltiples columnas en orden descendente
		`sorted_df_desc = df.sort(["age", "name"], ascending=[False, True])`

SparkContext()	Es un punto de entrada a Spark y se define en el paquete org.apache.spark desde la versión 1.x y se utiliza para crear programáticamente RDD de Spark, acumuladores y variables de difusión en el clúster.	`from pyspark import SparkContext`
		Creamos un SparkContext
		`sc = SparkContext("local", "MyApp")`

SparkSession	Es un punto de entrada a Spark, y crear una instancia de SparkSession sería la primera instrucción que escribirías en el programa con RDD, DataFrame y conjunto de datos.	`from pyspark.sql import SparkSession`
		Creamos una SparkSession
		`spark = SparkSession.builder.appName("MyApp").getOrCreate()`

Paquete/Método	Descripción	Ejemplo de Código
spark.read.json()	Spark SQL puede inferir automáticamente el esquema de un conjunto de datos JSON y cargarlo como un DataFrame. La función read.json() carga datos de un directorio de archivos JSON donde cada línea de los archivos es un objeto JSON. Ten en cuenta que el archivo ofrecido como archivo JSON no es un archivo JSON típico.	`json_df = spark.read.json("customer.json")`
spark.sql()	Para emitir cualquier consulta SQL, utiliza el método sql() en la instancia de SparkSession. Todas las consultas spark.sql ejecutadas de esta manera devuelven un DataFrame sobre el cual puedes realizar más operaciones de Spark si es necesario.	`result = spark.sql("SELECT name, age FROM cust_tbl WHERE age > 30")` `result.show()`

spark.udf.register()	En el DataFrame de PySpark, se utiliza para registrar una función definida por el usuario (UDF) con Spark, haciéndola accesible para su uso en consultas SQL de Spark. Esto te permite aplicar lógica o operaciones personalizadas a las columnas del DataFrame utilizando expresiones SQL.	Registrando una UDF (Función definida por el usuario)
		`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())`

where()	Se utiliza para filtrar las filas del DataFrame en función de la condición dada. Tanto las funciones filter() como where() se utilizan para el mismo propósito.	Filtrando filas basadas en una condición `filtered_df = df.where(df['age'] > 30)`

withColumn()	Función de transformación del DataFrame utilizada para cambiar el valor, convertir el tipo de dato de una columna existente, crear una nueva columna, y mucho más.	Agregando una nueva columna y realizando transformaciones
		`from pyspark.sql.functions import col`
		`new_df = df.withColumn("age_squared", col("age") ** 2)`

withColumnRenamed()	Devuelve un nuevo DataFrame renombrando una columna existente.	Renombrando una columna existente `renamed_df = df.withColumnRenamed("age", "years_old")`

Paquete/Método	Descripción	Ejemplo de Código