

Referencia rápida para operaciones dentro de Microsoft Fabric en diferentes lenguajes.

Charla Go Go Power Wranglers! Tu aliado para la transformación de datos en Fabric.

[Power BI Days España](#), Santiago de Compostela 26/10/2024

⚠ INDICACIONES ⚠

- Lenguajes:
 - Para poder hacer uso de pandas y determinadas funciones de PySpark, deben incluirse las referencias e importaciones necesarias dentro del código.
 - Para Spark SQL, puede bien definirse como lenguaje por defecto del cuaderno o emplearlo gracias al magic command `%%sql`
- Tabla de referencia:
 - Los *textos en cursiva* representan nombres y valores que pueden cambiar en función de lo especificado en el código.
 - Los nombres de las funciones, métodos y sentencias se muestran en **negrita**.

❖ Acción ❖	Ⓜ PowerQuery Ⓜ	🐼 Pandas 🐼	✳ PySpark ✳	✳️ Spark SQL ✳
Importación de la librería / uso	X	import pandas as pd	from pyspark.sql import functions as F	%%sql
Leer un CSV	Csv.Document(<source>, <columns>, optional <delimiter>, optional <extraValues>, optional <encoding>)	pd.read_csv(file_path)	spark.read.csv(file_path)	X
Visualizar las N primeras filas	Table.FirstN(#"Previous Step", N)	df.head(N)	df.show(N)	SELECT * FROM Table1 LIMIT N
Obtener total de filas	Table.RowCount(#"Previous Step")	df.shape[0]	df.count()	SELECT COUNT(*) FROM Table1
Visualizar esquema	Table.Schema("Table1")	df.dtypes	df.printSchema()	DESCRIBE Table1
Seleccionar columnas concretas	Table.SelectColumns(#"Previous Step",{"col1", "col2"})	df[["col1", "col2"]]	df.select("col1","col2")	SELECT col1, col2 FROM Table1
Filtrar datos	Table.SelectRows(#"Previous Step", each [col1] > 5)	df[df.col1 > 5]	df.filter(df["col1"] > 5)	SELECT * FROM Table1 WHERE col1 > 5
Ordenar por columnas	Table.Sort(#"Previous Step",{{"col1", Order.Ascending}})	df.sort_values("col1")	df.orderBy("col1")	SELECT * FROM Table1 ORDER BY col1

❖ Acción ❖	Ⓜ PowerQuery Ⓜ	🐼 Pandas 🐾	✿ PySpark ✿	✳️ Spark SQL ✳️
Rellenar valores vacíos (con un cero)	Table.ReplaceValue(#"Table1", null, "0", Replacer.ReplaceValue,{"col1"})	df[“col1”].fillna(0)	df.na.fill(value=0, subset=[“col1”])	UPDATE Table1 SET col1=0 WHERE column IS NULL
Unir DataFrames / tablas	Table.NestedJoin(#"Table1", {"col1", "col2"}, #"Table2", {"col1", "col2"}, "Table2", JoinKind.Inner)	pd.merge(df1, df2, on="col1", how="inner")	df1.join(df2, on="col1", how="inner")	SELECT * FROM Table1 AS A INNER JOIN Table2 AS B ON A.col1 = B.col2
Concatenar DataFrames / tablas	Table.Combine({#"Table1", #“Table2”})	pd.concat((df1, df2))	df1.union(df2)	SELECT * FROM Table1 UNION SELECT * FROM Table2
Agrupar por columnas	Table.Group(#"Table1", {"col1"}, {{"Average", each List.Average([col2]), type nullable number}})	df.groupby("col1")	df.groupBy("col1")	SELECT col1 FROM Table1 GROUP BY col1
Obtener valores únicos	Table.Distinct(#"Table1", {"col1"})	df[“col1”].unique()	df.select("col1").distinct()	SELECT DISTINCT col1 FROM Table1
Renombrar columnas	Table.RenameColumns(#"Table1", {"nombre_actual", "nombre_nuevo"})	df.rename(columns = {"nombre_anterior" : "nombre_nuevo"})	df.withColumnsRenamed(“nombre_viejo” : “nombre_nuevo”))	ALTER TABLE Table1 RENAME COLUMN nombre_actual TO nombre_nuevo
Eliminar columnas	Table.RemoveColumns(#"Previous Step", {"col1", "col2"})	df.drop(columns = [“col1”, “col2”])	df.drop("col1", "col2")	ALTER TABLE Table1 DROP COLUMNS col1, col2
Eliminar espacios en blanco	Table.TransformColumns(#"Previous Step", {"col1", Text.Trim, type text})	df[“col1”].str.strip()	df.withColumn("col1", trim("col1"))	SELECT TRIM(col1) FROM Table1
Longitud de un campo	Text.Length("col1")	df[“col1”].str.len()	df.withColumn("length", df.length("col1"))	SELECT LEN(col1) AS length FROM Table1
Extraer el año	Date.Year("datecol")	df[“datecol”].year	df.withColumn("year", year("datecol"))	SELECT YEAR(datecol) AS year FROM Table1
Extraer el mes	Date.Month("datecol")	df[“datecol”].month	df.withColumn("month", month("datecol"))	SELECT MONTH(datecol) AS month FROM Table1
Extraer el día	Date.Day("datecol")	df[“datecol”].day	df.withColumn("day", day("datecol"))	SELECT DAY(datecol) AS day FROM Table1
Fecha actual	DateTime.LocalNow() as datetime	pd.Timestamp.today()	df.withColumn("current_timestamp", current_timestamp())	SELECT CURRENT_TIMESTAMP() AS current_timestamp