kggfrnl8e

March 3, 2025

1 1. Introducción a DataFrames:

1.1 Conceptos básicos:

Los DataFrames son una abstracción de datos estructurados, organizados en filas y columnas, con un esquema definido. Esta estructura facilita la manipulación y el análisis de datos utilizando las APIs de Spark SQL.

1.2 Creación de DataFrames:

- **Desde RDDs**: Los DataFrames pueden crearse a partir de RDDs (colecciones distribuidas de datos). La creación de un DataFrame desde un RDD permite trabajar con datos no estructurados transformándolos en un formato tabular.
- **Desde archivos**: Spark SQL permite la creación de DataFrames desde varios formatos de archivos, como CSV, JSON y Parquet. Puedes cargar estos archivos directamente en un DataFrame utilizando la API de Spark SQL. También se pueden usar otros formatos de archivo.
- **Desde tablas Hive**: Puedes crear DataFrames a partir de tablas existentes en Hive, aprovechando el metastore de Hive.
- **Otras fuentes**: Spark puede leer datos de diversas fuentes incluyendo bases de datos relacionales mediante JDBC, NoSQL, ORC, y otros sistemas de almacenamiento.

1.2.1 1. Cargar datos desde un RDD:

Para convertir un RDD en un DataFrame, se utiliza la función `toDF()` o `createDataFrame()`.

- **toDF()**: Infiere el esquema del DataFrame a partir del RDD, normalmente usado con una tupla o lista en Python. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.toDF.h (similar, la original no está documentada en https://spark.apache.org/docs/latest/api/python/reference/api/
- **createDataFrame()**: Permite especificar explícitamente el esquema (`StructType`) del DataFrame. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.Spark

[0]:
```
# Inicializamos sesión

from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("RDDtoDF").getOrCreate()
```

[0]:
```
import sys
print("Python version: ", sys.version)
```

```
Python version:  3.9.21 (main, Dec  4 2024, 08:53:34)
[GCC 9.4.0]
```

[0]:
```python
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
print("Spark version: ", sc.version)
```

```
Spark version:  3.3.2
```

[0]:
```python
## Ejemplo con toDF():
rdd = spark.sparkContext.parallelize([("Alice", 34), ("Bob", 23)])
df = rdd.toDF(["name", "age"])
df.show()
```

```
+-----+---+
| name|age|
+-----+---+
|Alice| 34|
|  Bob| 23|
+-----+---+
```

[0]:
```python
## Ejemplo con createDataFrame() (Python):
from pyspark.sql import Row
from pyspark.sql.types import StructType, StructField, StringType, IntegerType

schema = StructType([
    StructField("name", StringType(), True),
    StructField("age", IntegerType(), True)
])

rdd = spark.sparkContext.parallelize([("Alice", 34), ("Bob", 23)])
df = spark.createDataFrame(rdd, schema)
df.show()
```

```
+-----+---+
| name|age|
+-----+---+
|Alice| 34|
|  Bob| 23|
+-----+---+
```

1.2.2 2. Cargar datos desde ficheros CSV:

Sintaxis: Se utiliza `spark.read.csv()`. Se pueden especificar opciones como `header` para indicar si el archivo tiene encabezado e `inferSchema` para que Spark infiera los tipos de datos.

- `header` indica si la primera línea del archivo CSV contiene los nombres de las columnas.

- `inferSchema` permite a Spark determinar automáticamente los tipos de datos de cada columna.

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameReader.csv.ht

Consideraciones

- **Rutas de archivos**: Asegúrate de proporcionar las rutas correctas a tus archivos CSV y Parquet.
- **Esquema**: Si no se utiliza `inferSchema` al leer archivos CSV, el esquema del DataFrame debe especificarse explícitamente.
- **DataFrames**: Los DataFrames proporcionan una forma de procesar y analizar datos estructurados. A diferencia de los RDDs, los DataFrames están basados en un esquema, es decir, conocen los nombres y tipos de las columnas de un conjunto de datos.

[0]:
```
# Desde un fichero CSV
#df=spark.read.option("header", "true").option("inferSchema", "true").csv("c:/
 ↪BDASpark/olive.csv")
df = spark.read.option("header", "true").option("inferSchema", "true").
 ↪csv("dbfs:/FileStore/olive.csv")

df.show()
```

```
+-------+----+---------------+------------------+------------+-------+------
--------------+------------------+-------+--------------------+----------+---
--------------+------------+
|Country|Year|Beginning Stocks|Domestic Consumption|Ending Stocks|Exports|Feed
Waste Dom. Cons.|Food Use Dom. Cons.|Imports|Industrial Dom.
Cons.|Production|Total Distribution|Total Supply|
+-------+----+---------------+------------------+------------+-------+------
--------------+------------------+-------+--------------------+----------+---
--------------+------------+
|Algeria|1964|              0|                15|           0|      3|
0|                15|      0|                 0|        18|
18|          18|
|Algeria|1965|              0|                12|           0|      5|
0|                12|      0|                 0|        17|
17|          17|
|Algeria|1966|              0|                16|           0|      0|
0|                16|      0|                 0|        16|
16|          16|
|Algeria|1967|              0|                15|           0|      7|
0|                15|      0|                 0|        22|
22|          22|
|Algeria|1968|              0|                11|           0|      7|
0|                11|      0|                 0|        18|
18|          18|
|Algeria|1969|              0|                19|           0|      3|
0|                19|      0|                 0|        22|
```

```
22|          22|
|Algeria|1970|              0|                  12|            0|      1|
0|                  12|      0|                   0|         13|
13|          13|
|Algeria|1971|              0|                  20|            2|      1|
0|                  20|      0|                   0|         23|
23|          23|
|Algeria|1972|              2|                  13|            2|      2|
0|                  13|      0|                   0|         15|
17|          17|
|Algeria|1973|              2|                  11|            3|      4|
0|                  11|      0|                   0|         16|
18|          18|
|Algeria|1974|              3|                  10|            1|      0|
0|                  10|      0|                   0|          8|
11|          11|
|Algeria|1975|              1|                  16|            3|      0|
0|                  16|      0|                   0|         18|
19|          19|
|Algeria|1976|              3|                  15|            3|      0|
0|                  15|      0|                   0|         15|
18|          18|
|Algeria|1977|              3|                   8|            0|      0|
0|                   8|      0|                   0|          5|
8|           8|
|Algeria|1978|              0|                  13|            1|      0|
0|                  13|      0|                   0|         14|
14|          14|
|Algeria|1979|              1|                  11|            0|      0|
0|                  11|      0|                   0|         10|
11|          11|
|Algeria|1980|              0|                  18|            0|      0|
0|                  18|      0|                   0|         18|
18|          18|
|Algeria|1981|              0|                  15|            0|      0|
0|                  15|      0|                   0|         15|
15|          15|
|Algeria|1982|              0|                  16|            0|      0|
0|                  16|      0|                   0|         16|
16|          16|
|Algeria|1983|              0|                  13|            0|      0|
0|                  13|      0|                   0|         13|
13|          13|
+-------+----+---------------+--------------------+-------------+-------+------
---------------+-------------------+-------+--------------------+----------+---
---------------+------------+
only showing top 20 rows
```

1.2.3 3. Cargar datos desde ficheros Parquet:

Se utiliza `spark.read.parquet()`.
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameReader.parqu

Consideraciones

- **Esquema**: El esquema se almacena en el mismo archivo.

[0]:
```
# Desde un único fichero parquet
# df=spark.read.parquet("c:/BDASpark/palm.parquet")
df=spark.read.parquet("dbfs:/FileStore/palm.parquet")
df.show()
```

```
+-----------+----+--------------+---------------+--------------------+---------
----+-------+--------------------+------------------+-------+----------------
----+----------+------------------+------------+-----+
|    Country|Year|Area Harvested|Beginning Stocks|Domestic Consumption|Ending
Stocks|Exports|Feed Waste Dom. Cons.|Food Use Dom. Cons.|Imports|Industrial Dom.
Cons.|Production|Total Distribution|Total Supply|Yield|
+-----------+----+--------------+---------------+--------------------+---------
----+-------+--------------------+------------------+-------+----------------
----+----------+------------------+------------+-----+
|Afghanistan|1964|           0.0|            0.0|                 0.0|
0.0|    0.0|                 0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1965|           0.0|            0.0|                 0.0|
0.0|    0.0|                 0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1966|           0.0|            0.0|                 0.0|
0.0|    0.0|                 0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1967|           0.0|            0.0|                 0.0|
0.0|    0.0|                 0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1968|           0.0|            0.0|                 0.0|
0.0|    0.0|                 0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1969|           0.0|            0.0|                 0.0|
0.0|    0.0|                 0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1970|           0.0|            0.0|                 0.0|
0.0|    0.0|                 0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1971|           0.0|            0.0|                 0.0|
0.0|    0.0|                 0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1972|           0.0|            0.0|                 0.0|
0.0|    0.0|                 0.0|               0.0|    0.0|
```

```
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1973|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1974|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1975|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1976|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1977|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1978|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1979|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1980|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1981|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1982|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
|Afghanistan|1983|          0.0|              0.0|                  0.0|
0.0|    0.0|                  0.0|               0.0|    0.0|
0.0|       0.0|               0.0|         0.0|  0.0|
+-----------+----+--------------+---------------+--------------------+---------
----+-------+--------------------+-------------------+-------+----------------
----+----------+------------------+-----------+-----+
only showing top 20 rows
```

[0]:
```
# Desde un directorio con múltiples archivos
# df = spark.read.parquet("c:/BDASpark/olive.parquet")
df = spark.read.parquet("dbfs:/FileStore/palm.parquet")
df.show()
```

```
+-----------+----+--------------+---------------+--------------------+---------
----+-------+--------------------+-------------------+-------+----------------
----+----------+------------------+-----------+-----+
```

Country	Year	Area Harvested	Beginning Stocks	Domestic Consumption	Ending Stocks	Exports	Feed Waste Dom. Cons.	Food Use Dom. Cons.	Imports	Industrial Dom. Cons.	Production	Total Distribution	Total Supply	Yield
Afghanistan	1964	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1965	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1966	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1967	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1968	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1969	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1970	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1971	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1972	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1973	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1974	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1975	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1976	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Afghanistan	1977	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
|Afghanistan|1978|          0.0|             0.0|                 0.0|
0.0|    0.0|                 0.0|                 0.0|    0.0|
0.0|       0.0|              0.0|          0.0|  0.0|
|Afghanistan|1979|          0.0|             0.0|                 0.0|
0.0|    0.0|                 0.0|                 0.0|    0.0|
0.0|       0.0|              0.0|          0.0|  0.0|
|Afghanistan|1980|          0.0|             0.0|                 0.0|
0.0|    0.0|                 0.0|                 0.0|    0.0|
0.0|       0.0|              0.0|          0.0|  0.0|
|Afghanistan|1981|          0.0|             0.0|                 0.0|
0.0|    0.0|                 0.0|                 0.0|    0.0|
0.0|       0.0|              0.0|          0.0|  0.0|
|Afghanistan|1982|          0.0|             0.0|                 0.0|
0.0|    0.0|                 0.0|                 0.0|    0.0|
0.0|       0.0|              0.0|          0.0|  0.0|
|Afghanistan|1983|          0.0|             0.0|                 0.0|
0.0|    0.0|                 0.0|                 0.0|    0.0|
0.0|       0.0|              0.0|          0.0|  0.0|
+-----------+----+--------------+----------------+--------------------+---------
----+-------+---------------------+-------------------+-------+----------------
----+----------+------------------+------------+-----+
only showing top 20 rows
```

2 2. API de DataFrames:

Para realizar transformaciones en un DataFrame en Spark con Python, se utilizan diversas funciones que permiten modificar, seleccionar, o agregar datos. Aquí te presento las sintaxis y ejemplos de uso de algunas de las transformaciones más comunes:

2.1 Transformaciones:

- **select**: Permite seleccionar columnas específicas de un DataFrame.
- **filter**: Permite filtrar filas basadas en una condición.
- **withColumn**: Permite añadir nuevas columnas o modificar las existentes.
- **Otras transformaciones**: El API incluye otras transformaciones para manipular los datos como groupBy, sort y join. También permite crear funciones definidas por el usuario para manipulación personalizada de datos.

2.2 Acciones:

- **show**: Muestra las primeras filas de un DataFrame.
- **count**: Cuenta el número de filas en un DataFrame.
- **collect**: Retorna todos los elementos de un DataFrame al driver (cuidado con el uso en grandes datasets).
- **Otras acciones**: Incluyen take, takeSample y describe para obtener información y estadísticas sobre los DataFrames.

2.3 Consideraciones:

- **Inmutabilidad**: Los DataFrames son inmutables; cada transformación crea un nuevo DataFrame.
- **show()**: La función `show()` se utiliza para mostrar una muestra de los datos resultantes tras una transformación.
- **Importaciones**: Algunas funciones requieren importaciones adicionales desde `pyspark.sql.functions`, como `col`, `lit`, `expr`, `avg`, `count`, etc.
- **Expresiones SQL**: Puedes usar expresiones SQL con `expr()` y `selectExpr()` para transformaciones más complejas.
- **Columnas**: Las columnas se pueden referenciar usando su nombre como string, usando la notación de corchetes sobre el DataFrame o con la función `col()`.

2.3.1 1. select():

Se utiliza para seleccionar un subconjunto de columnas de un DataFrame. También se puede usar `selectExpr()` para seleccionar columnas con expresiones SQL. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.select.html

[0]:
```
df = spark.createDataFrame([(2, "Alice"), (5, "Bob")], schema=["age", "name"])

df.select("name", "age").show()
df.select('*').show()
df.select(df.name, (df.age + 10).alias('age')).show()

# Variante con expresiones SQL
df.selectExpr("name", "age", "age * 2 as double_age").show()
```

```
+-----+---+
| name|age|
+-----+---+
|Alice|  2|
|  Bob|  5|
+-----+---+

+---+-----+
|age| name|
+---+-----+
|  2|Alice|
|  5|  Bob|
+---+-----+

+-----+---+
| name|age|
+-----+---+
|Alice| 12|
|  Bob| 15|
+-----+---+
```

```
+-----+---+----------+
| name|age|double_age|
+-----+---+----------+
|Alice|  2|         4|
|  Bob|  5|        10|
+-----+---+----------+
```

2.3.2 2. filter() o where():

Se utiliza para filtrar filas basadas en una condición. `filter()` y `where()` son sinónimos. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.filter.html

[0]:
```
df.filter(df["age"] > 30).show()
df.where(df["age"] > 30).show()
```

```
+---+----+
|age|name|
+---+----+
+---+----+

+---+----+
|age|name|
+---+----+
+---+----+
```

2.3.3 3. withColumn():

Se utiliza para añadir una nueva columna o reemplazar una existente. La función `lit()` crea una columna con un valor literal y `expr()` permite usar expresiones SQL. https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.withColumn.

[0]:
```
from pyspark.sql.functions import lit, expr

df.withColumn("age_plus_10", df["age"] + 10).show()
df.withColumn("is_adult", lit(True)).show()
df.withColumn("age_times_2", expr("age * 2")).show()
```

```
+---+-----+-----------+
|age| name|age_plus_10|
+---+-----+-----------+
|  2|Alice|         12|
|  5|  Bob|         15|
+---+-----+-----------+

+---+-----+--------+
|age| name|is_adult|
+---+-----+--------+
```

```
|  2|Alice|    true|
|  5|  Bob|    true|
+---+-----+--------+

+---+-----+-----------+
|age| name|age_times_2|
+---+-----+-----------+
|  2|Alice|          4|
|  5|  Bob|         10|
+---+-----+-----------+
```

2.3.4 4. withColumnRenamed():

Se utiliza para renombrar una columna existente.
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.withColumnl

[0]:
```
df.withColumnRenamed("age", "years").show()
```

```
+-----+-----+
|years| name|
+-----+-----+
|    2|Alice|
|    5|  Bob|
+-----+-----+
```

2.3.5 5. groupBy():

Se utiliza para agrupar filas con valores iguales en una columna y realizar operaciones de agregación. Se combina con funciones de agregación como `count()`, `sum()`, `avg()`, `min()`, `max()`.
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.groupBy.htm

[0]:
```
# df.groupBy("occupation").agg({"age": "avg", "*": "count"}).show()
df.groupBy("name").agg({"age": "avg", "*": "count"}).show()
```

```
+-----+--------+--------+
| name|count(1)|avg(age)|
+-----+--------+--------+
|Alice|       1|     2.0|
|  Bob|       1|     5.0|
+-----+--------+--------+
```

2.3.6 6. sort() o orderBy():

Se utiliza para ordenar las filas del DataFrame. `sort()` y `orderBy()` son equivalentes y pueden usar el orden ascendente (`asc`) o descendente (`desc`).

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.sort.html
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.orderBy.html

[0]:
```
df.sort("age").show()
df.sort(df["age"].desc()).show()
df.orderBy("name").show()
df.orderBy(df["name"].desc()).show()
```

```
+---+-----+
|age| name|
+---+-----+
|  2|Alice|
|  5|  Bob|
+---+-----+

+---+-----+
|age| name|
+---+-----+
|  5|  Bob|
|  2|Alice|
+---+-----+

+---+-----+
|age| name|
+---+-----+
|  2|Alice|
|  5|  Bob|
+---+-----+

+---+-----+
|age| name|
+---+-----+
|  5|  Bob|
|  2|Alice|
+---+-----+
```

2.3.7 7. drop():

Se utiliza para eliminar una o varias columnas del DataFrame.
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.drop.html

[0]:
```
df.drop("occupation").show()
```

```
+---+-----+
|age| name|
+---+-----+
|  2|Alice|
```

```
|  5|  Bob|
+---+-----+
```

2.3.8 8. distinct():

Se utiliza para eliminar las filas duplicadas del DataFrame.
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.distinct.html

[0]:
```
df.distinct().show()
```

```
+---+-----+
|age| name|
+---+-----+
|  2|Alice|
|  5|  Bob|
+---+-----+
```

2.3.9 9. join():

Se utiliza para combinar dos DataFrames basados en una o varias columnas en común. Se puede especificar el tipo de join: 'inner', 'outer', 'left_outer', 'right_outer', o 'leftsemi'.
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.join.html

[0]:
```
# Creamos dos DataFrames de ejemplo con una columna 'id' en común
data1 = [(1, "Alice"), (2, "Bob"), (3, "Charlie")]
data2 = [(1, "New York"), (2, "London"), (4, "Tokyo")]

df1 = spark.createDataFrame(data1, schema=["id", "name"])
df2 = spark.createDataFrame(data2, schema=["id", "city"])

df1.join(df2, df1["id"] == df2["id"], "inner").show()
df1.join(df2, df1["id"] == df2["id"], "left_outer").show()
```

```
+---+-----+---+--------+
| id| name| id|    city|
+---+-----+---+--------+
|  1|Alice|  1|New York|
|  2|  Bob|  2|  London|
+---+-----+---+--------+

+---+-------+----+--------+
| id|   name|  id|    city|
+---+-------+----+--------+
|  1|  Alice|   1|New York|
|  2|    Bob|   2|  London|
|  3|Charlie|null|    null|
```

```
+---+-------+----+--------+
```

2.3.10 10. union() o unionAll():

Se utiliza para combinar dos DataFrames con las mismas columnas. `union()` elimina duplicados, `unionAll()` no.
https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.union.html
y https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.unionAll.ht

[0]:
```
df1.union(df2).show()
df1.unionAll(df2).show()
```

```
+---+--------+
| id|    name|
+---+--------+
|  1|   Alice|
|  2|     Bob|
|  3| Charlie|
|  1|New York|
|  2|  London|
|  4|   Tokyo|
+---+--------+

+---+--------+
| id|    name|
+---+--------+
|  1|   Alice|
|  2|     Bob|
|  3| Charlie|
|  1|New York|
|  2|  London|
|  4|   Tokyo|
+---+--------+
```

2.3.11 11. map():

Se utiliza para aplicar una función a cada fila del DataFrame, convirtiéndolo a RDD.
La función map() se aplica a RDDs (Resilient Distributed Datasets), no directamente a DataFrames.
Para usar map en un DataFrame, primero debes convertirlo a un RDD usando df.rdd.
https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.map.html

[0]:
```
df = spark.createDataFrame([(2, "Alice"), (5, "Bob")], schema=["age", "name"])
df.rdd.map(lambda row: (row.name, row.age * 2)).toDF(["name", "double_age"]).
  ↪show()
```

```
+-----+----------+
| name|double_age|
```

```
+-----+----------+
|Alice|         4|
|  Bob|        10|
+-----+----------+
```