**Explode() Function**

In PySpark, the explode() function is used to transform a column of arrays or maps into multiple rows, effectively "exploding" the elements of the array or map into separate rows. This is particularly useful when you have a nested structure in your DataFrame and want to unnest it.

1. Exploding Arrays in Nested Data:

from pyspark.sql import SparkSession  
from pyspark.sql.functions import explode  
  
*# Create a SparkSession*spark = SparkSession.builder.appName("nested\_data\_example").getOrCreate()  
  
*# Sample data with a column containing arrays*data = [(1, [10, 20, 30]),

(2, [40, 50]),

(3, [60, 70, 80, 90])]

df = spark.createDataFrame(data, ["id", "values"])  
  
*# Explode the 'values' array column*df1 = df.select("id",explode("values").alias("exploded\_value"))  
  
df1.show()

**Output:**

+---+--------------+

| id|exploded\_value|

+---+--------------+

| 1| 10|

| 1| 20|

| 1| 30|

| 2| 40|

| 2| 50|

| 3| 60|

| 3| 70|

| 3| 80|

| 3| 90|

+---+--------------+

1. Exploding Maps in Nested Data:

from pyspark.sql import SparkSession  
from pyspark.sql.functions import explode, col  
  
*# Create SparkSession*spark = SparkSession.builder.appName("nested\_map\_example").getOrCreate()  
  
*# Sample data with a column containing maps*data = [(1, {"A": 10, "B": 20}),

(2, {"C": 30, "D": 40}),

(3, {"E": 50})]  
  
*# Create DataFrame*df = spark.createDataFrame(data, ["id", "map\_values"])  
  
*# Explode the 'map\_values' map column*df1 = df.select('id',explode('map\_values').alias('key','value'))  
  
df1.show()

**Output:**

+---+---+-----+

| id|key|value|

+---+---+-----+

| 1| A| 10|

| 1| B| 20|

| 2| D| 40|

| 2| C| 30|

| 3| E| 50|

+---+---+-----+

1. Exploding Multiple Array Columns:

from pyspark.sql import \*  
from pyspark.sql.functions import \*  
  
*# Create SparkSession*spark = SparkSession.builder.appName("multiple\_array\_columns").getOrCreate()  
  
*# Sample data with multiple array columns*data = [(1, [10, 20], ["A", "B"]),  
 (2, [30], ["C"]),  
 (3, [40, 50], ["D", "E"])]  
df = spark.createDataFrame(data, ["id", "values1", "values2"])  
  
*# Explode 'values1' array column*df\_exploded\_1 =

df.withColumn('exploded\_values1', explode(col('values1')))  
  
*# Explode 'values2' array column*df\_exploded\_2 = df\_exploded\_1.withColumn('exploded\_values2',explode(col('values2')))  
  
df\_exploded\_2.show()

**Output:**

+---+--------+-------+----------------+----------------+

| id| values1|values2|exploded\_values1|exploded\_values2|

+---+--------+-------+----------------+----------------+

| 1|[10, 20]| [A, B]| 10| A|

| 1|[10, 20]| [A, B]| 10| B|

| 1|[10, 20]| [A, B]| 20| A|

| 1|[10, 20]| [A, B]| 20| B|

| 2| [30]| [C]| 30| C|

| 3|[40, 50]| [D, E]| 40| D|

| 3|[40, 50]| [D, E]| 40| E|

| 3|[40, 50]| [D, E]| 50| D|

| 3|[40, 50]| [D, E]| 50| E|

+---+--------+-------+----------------+----------------+

**split() Function**

In PySpark, the split() function is used to split a string into an array of substrings based on a specified delimiter. This function is similar to the standard Python split() method for strings but operates on distributed data in a Spark DataFrame or RDD (Resilient Distributed Dataset).

1. Splitting a sentence into words

from pyspark.sql import \*  
from pyspark.sql.functions import \*  
  
*# Create a SparkSession*spark = SparkSession.builder.appName("SplitFunctionScenario1").getOrCreate()  
  
*# Sample data*data = [(1,'Hello, How are you?'),(2,'PySpark is awesome')]  
  
*# Create a DataFrame*df = spark.createDataFrame(data, ["id","sentences"])  
  
*# Use split() to split the "sentences" column by whitespace*df = df.withColumn("words", split(col("sentences"), " "))  
  
*# Show the result*df.show(truncate=False)

**output:**

+---+-------------------+------------------------+

|id |sentences |words |

+---+-------------------+------------------------+

|1 |Hello, How are you?|[Hello,, How, are, you?]|

|2 |PySpark is awesome |[PySpark, is, awesome] |

+---+-------------------+------------------------+

1. Splitting a comma-separated list into an array of elements

from pyspark.sql import \*  
from pyspark.sql.functions import \*  
  
*# Create a SparkSession*spark = SparkSession.builder.appName("SplitFunctionScenario2").getOrCreate()  
  
*# Sample data*data = [('A',"1,2,3",), ('B',"4,5,6,7",)]  
  
*# Create a DataFrame*df = spark.createDataFrame(data, ["Alphabets","numbers"])  
  
*# Use split() to split the "numbers" column by the comma delimiter*df = df.withColumn("number\_array", split(col("numbers"), ","))  
  
*# Show the result*df.show(truncate=False)

**output:**

+---------+-------+------------+

|Alphabets|numbers|number\_array|

+---------+-------+------------+

|A |1,2,3 |[1, 2, 3] |

|B |4,5,6,7|[4, 5, 6, 7]|

+---------+-------+------------+

1. Splitting a string using a regular expression

from pyspark.sql import \*  
from pyspark.sql.functions import \*  
  
*# Create a SparkSession*spark = SparkSession.builder.appName("SplitFunctionScenario3").getOrCreate()  
  
*# Sample data*data = [(1,"A1B2C3"), (2,"X9Y10Z11")]  
  
*# Create a DataFrame*df = spark.createDataFrame(data, ["id","pattern"])  
  
*# Use split() with a regular expression to split the "pattern" column*df = df.withColumn("result", split(col("pattern"), "[0-9]"))  
  
*# Show the result*df.show(truncate=False)

**output:**

+---+--------+---------------+

|id |pattern |result |

+---+--------+---------------+

|1 |A1B2C3 |[A, B, C, ] |

|2 |X9Y10Z11|[X, Y, , Z, , ]|

+---+--------+---------------+

**array()Function**

In PySpark, the array() function is used to create an array column from the input values. An array column is a column that contains an array of elements. It is commonly used when working with nested data structures or when you want to group multiple values together in a single column.

from pyspark.sql.functions import \*  
from pyspark.sql import \*  
  
spark = SparkSession.builder.appName("array\_function").getOrCreate()  
  
my\_data = [(1,'Harsha','Vardhan'),  
 (2,'Kiran','Kumar'),  
 (3,'Mokshit','M')]  
  
df = spark.createDataFrame(my\_data,['id','first\_name','last\_name'])  
  
df\_with\_array = df.withColumn('name',array(col('first\_name'),col('last\_name')))  
  
df\_with\_array.show()

**output:**

+---+----------+---------+-----------------+

| id|first\_name|last\_name| name|

+---+----------+---------+-----------------+

| 1| Harsha| Vardhan|[Harsha, Vardhan]|

| 2| Kiran| Kumar| [Kiran, Kumar]|

| 3| Mokshit| M| [Mokshit, M]|

+---+----------+---------+-----------------+

**Array\_contains()Function**

Array\_contains() functions will check whether the element is present in the given array of elements.

from pyspark.sql import \*  
from pyspark.sql.functions import \*  
  
*# Create SparkSession*spark = SparkSession.builder.appName('array\_contains').getOrCreate()  
  
*# Create Data*data = [(1,'Harsha',['Azure','python']),  
 (2,'Kiran',['Python','Pyspark'])]  
  
*# Create DataFrame*df = spark.createDataFrame(data,['id','name','skills'])  
  
df1 = df.withColumn('PysparkSkill',array\_contains(col('skills'),'Pyspark'))  
  
df1.show()

**Output:**

+---+------+-----------------+------------+

| id| name| skills|PysparkSkill|

+---+------+-----------------+------------+

| 1|Harsha| [Azure, python]| false|

| 2| Kiran|[Python, Pyspark]| true|

+---+------+-----------------+------------+