databrickspracticas

(https://databricks.com)

1

```
// Importar librerías necesarias
import org.apache.spark.sql.{SparkSession, DataFrame}
import org.apache.spark.sql.functions._

// Crear una sesión de Spark
val spark = SparkSession.builder()
  .appName("Practica Scala")
  .config("spark.master", "local")
  .getOrCreate()

// Crear una tabla con datos ficticios
val data = Seq(
  (1, "a", 5, ""),
  (-10, "b", 15, "texto"),
  (0, "e", 3, "scala"),
  (20, "z", 200, "escalera"),
  (-5, "i", 9, "")
)

val df = spark.createDataFrame(data).toDF("num", "char", "year", "cadena")
df.show()
```

▸ df: org.apache.spark.sql.DataFrame = [num: integer, char: string ... 2 more fields]

```
+---+----+----+--------+
|num|char|year|  cadena|
+---+----+----+--------+
|  1|   a|   5|        |
|-10|   b|  15|   texto|
|  0|   e|   3|   scala|
| 20|   z| 200|escalera|
| -5|   i|   9|        |
+---+----+----+--------+

import org.apache.spark.sql.{SparkSession, DataFrame}
import org.apache.spark.sql.functions._
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@5c6d1acb
data: Seq[(Int, String, Int, String)] = List((1,a,5,""), (-10,b,15,texto), (0,e,3,scala), (20,z,200,escalera), (-5,i,
9,""))
df: org.apache.spark.sql.DataFrame = [num: int, char: string ... 2 more fields]
```

Estructuras condicionales

3

```
// 1. Dado un número, imprimir "Positivo", "Negativo" o "Cero".
println("1. Condicional: Positivo, Negativo o Cero")
df.withColumn("num_condicion", when(col("num") > 0, "Positivo")
  .when(col("num") < 0, "Negativo")
  .otherwise("Cero")).show()

// 2. Dado un carácter, imprimir "Es vocal" o "No es vocal".
println("2. Condicional: Es vocal o No es vocal")
df.withColumn("es_vocal", when(col("char").isin("a", "e", "i", "o", "u"), "Es vocal")
  .otherwise("No es vocal")).show()

// 3. Dado un número, imprimir "Dígito" o "Número de múltiples dígitos".
println("3. Condicional: Dígito o Número de múltiples dígitos")
df.withColumn("digito_condicion", when(abs(col("num")) < 10, "Dígito")
  .otherwise("Número de múltiples dígitos")).show()

// 4. Dada una cadena, imprimir "Cadena vacía" o "Cadena no vacía".
println("4. Condicional: Cadena vacía o no vacía")
df.withColumn("cadena_condicion", when(col("cadena") === "", "Cadena vacía")
  .otherwise("Cadena no vacía")).show()

// 5. Dado un año, imprimir "Año bisiesto" o "Año no bisiesto".
println("5. Condicional: Año bisiesto o no bisiesto")
df.withColumn("es_bisiesto", when((col("year") % 4 === 0 && col("year") % 100 =!= 0) || (col("year") % 400 === 0),
bisiesto")
  .otherwise("Año no bisiesto")).show()
```

```
1. Condicional: Positivo, Negativo o Cero
+---+----+----+--------+-------------+
|num|char|year|  cadena|num_condicion|
+---+----+----+--------+-------------+
|  1|   a|   5|        |     Positivo|
|-10|   b|  15|   texto|     Negativo|
|  0|   e|   3|   scala|         Cero|
| 20|   z| 200|escalera|     Positivo|
| -5|   i|   9|        |     Negativo|
+---+----+----+--------+-------------+

2. Condicional: Es vocal o No es vocal
+---+----+----+--------+-----------+
|num|char|year|  cadena|   es_vocal|
+---+----+----+--------+-----------+
|  1|   a|   5|        |   Es vocal|
|-10|   b|  15|   texto|No es vocal|
|  0|   e|   3|   scala|   Es vocal|
| 20|   z| 200|escalera|No es vocal|
| -5|   i|   9|        |   Es vocal|
+---+----+----+--------+-----------+
```

Estructuras de iteracion

```
5
```

```
// 1. Imprimir los números del 10 al 1 en orden descendente.
println("1. Iteración: Números del 10 al 1")
(10 to 1 by -1).foreach(println)

// 2. Imprimir los elementos de un arreglo de enteros.
println("2. Iteración: Elementos de un arreglo")
val arr = Array(1, 2, 3, 4, 5)
arr.foreach(println)

// 3. Calcular la potencia de un número utilizando un bucle for.
println("3. Iteración: Calcular la potencia de un número")
def potencia(base: Int, exponente: Int): Int = {
  var resultado = 1
  for (_ <- 1 to exponente) {
    resultado *= base
  }
  resultado
}
println(s"Potencia: ${potencia(2, 3)}") // Ejemplo: 2^3

// 4. Imprimir los números impares del 1 al 50 utilizando un bucle do-while.
println("4. Iteración: Números impares del 1 al 50")
var i = 1
do {
  if (i % 2 != 0) println(i)
  i += 1
} while (i <= 50)

// 5. Calcular la suma de los primeros n números naturales utilizando un bucle while.
println("5. Iteración: Suma de los primeros n números naturales")
def sumaNaturales(n: Int): Int = {
  var suma = 0
  var i = 1
  while (i <= n) {
    suma += i
    i += 1
  }
  suma
}
println(s"Suma de los primeros 10 números: ${sumaNaturales(10)}")
```

```
1. Iteración: Números del 10 al 1
10
9
8
7
6
5
4
3
2
1
2. Iteración: Elementos de un arreglo
1
2
3
4
5
3. Iteración: Calcular la potencia de un número
Potencia: 8
4. Iteración: Números impares del 1 al 50
1
```

Estructuras en Scala

7

```
1. Función: Obtener promedio
Promedio: 3.0
2. Función: Concatenar cadenas
Concatenación: Hola, mundo!
```

```
3. Función: Verificar si es palíndromo
¿Es palíndromo 'radar'? true
¿Es palíndromo 'scala'? false
4. Función: Duplicar elementos de una lista
Duplicar elementos: List(1, 1, 2, 2, 3, 3)
5. Función: Verificar si un número es capicúa
¿Es capicúa 121? true
¿Es capicúa 123? false
obtenerPromedio: (nums: List[Int])Double
concatenarCadenas: (cad1: String, cad2: String)String
esPalindromo: (cadena: String)Boolean
duplicarElementos: (lista: List[Int])List[Int]
esCapicua: (num: Int)Boolean
```