Apuntes SPARK SQL

1. Introducción a Spark

Apache Spark es un motor de procesamiento distribuido diseñado para trabajar con grandes volúmenes de datos de forma rápida y escalable. Permite trabajar con:

- Transformaciones en DataFrames
- Consultas con Spark SQL
- Algoritmos de Machine Learning (MLlib)
- Procesamiento en streaming

2. DataFrames en Spark

Un DataFrame es una estructura de datos distribuida organizada en columnas, similar a una tabla. Permite:

- Filtrar
- Seleccionar columnas
- Crear columnas nuevas
- Agrupar
- Ordenar
- Hacer joins

Spark es **inmutable** → cada operación devuelve un DataFrame nuevo.

```
//Crear un dataframe de ejemplo
val dfRaw = Seq(
  ("Bizkaia", "enero", 2200.0),
  ("bizkaia", "01",    2300.0),
  ("BIZKAIA", "Jan",   null.asInstanceOf[Double])
).toDF("provincia", "mes", "precio")
```

3. Función withColumn()

Sirve para crear o modificar columnas.

```
df.withColumn("nombre_columna", expresion)
```

Parámetro 1: nombre_columna

- Si no existe → se crea
- Si existe → se reemplaza

Parámetro 2: expresion

Puede contener:

- operaciones matemáticas
- expresiones lógicas
- funciones (lower, upper, trim…)
- condiciones con when / otherwise
- literales (lit())
- castings

```
// Crear nueva
df.withColumn("edad2", col("edad") * 2)

// Sobrescribir
df.withColumn("nombre", upper(col("nombre")))

// Condicional
df.withColumn("categoria",
  when(col("nota") >= 5, "Aprobado").otherwise("Suspenso")
)

// Casting
df.withColumn("mes_num", col("mes").cast("int"))
```

4. Función isin()

Sirve para comprobar si una columna está dentro de un conjunto de valores.

Ejemplo

```
df.filter(col("curso").isin("DAM", "DAW"))
//Equivalente a:
col("curso") === "DAM" || col("curso") === "DAW"
```

Usado frecuentemente tras un lower() para normalizar texto:

```
when(lower(col("provincia")).isin("bizkaia","vizcaya"), "Bizkaia")
```

5. Normalización de texto en Spark

Muy útil para "data wrangling":

```
//Pasar a minusculas
withColumn("prov_norm", lower(col("provincia")))

//Eliminar los espacios blancos
withColumn("prov_norm", trim(col("provincia")))
```

6. Expresiones condicionales: when / otherwise

Son el equivalente a un IF / ELSE.

```
df.withColumn("nivel",
          when(col("nota") >= 9, "Sobresaliente")
         .when(col("nota") >= 7, "Notable")
         .when(col("nota") >= 5, "Aprobado")
         .otherwise("Suspenso")
)
```

7. Funciones útiles

- lower(col()) → minúsculas
- upper(col()) → mayúsculas
- trim(col()) → elimina espacios
- coalesce() → primera columna no nula
- lit() → literal
- cast() → conversión de tipo

Ejemplo:

```
df.withColumn("precio_final", coalesce(col("precio"), lit(0)))
```

8. Spark SQL - Vistas

Podemos registrar cualquier DataFrame como una vista:

df.createOrReplaceTempView("tabla")

Y consultarlo en SQL:

SELECT provincia, AVG(precio) FROM tabla GROUP BY provincia;

9. Data Wrangling con Spark – Ejemplo con provincia.csv

Data wrangling = limpiar, transformar y preparar datos.

Tareas típicas con Spark:

- normalizar texto (lower, trim)
- transformar meses: Jan → 1, Feb → 2…
- unificar provincias con isin
- eliminar duplicados
- convertir tipos
- imputar valores faltantes
- detectar outliers

Ejemplo:

```
val dfProv = dfRaw.withColumn(
                "provincia_norm",
                when(lower(col("provincia")).isin("bizkaia","vizcaya","BIZKAIA"), "Bizkaia")
                .otherwise(col("provincia"))
              )

dfProv.show()


+---------+-----+------+--------------+
|provincia|  mes|precio|provincia_norm|
+---------+-----+------+--------------+
|  Bizkaia|enero|2200.0|       Bizkaia|
|  bizkaia|   01|2300.0|       Bizkaia|
|  BIZKAIA|  Jan|   0.0|       Bizkaia|
+---------+-----+------+--------------+
```

```
//Normalizar fechas
val dfMes = dfProv
  .withColumn("mes_trim", trim(col("mes")))
  .withColumn("mes_lower", lower(col("mes_trim")))
  .withColumn("mes_num", when(col("mes_lower").isin("enero", "jan", "1", "01"), lit(1))
    // .when(col("mes_lower").isin("febrero","feb","2","02"), lit(2))
    // ...
    // .when(col("mes_lower").isin("diciembre","dec","12"), lit(12))
  )
```

READY

10. Crear vistas para análisis en Zeppelin

```
dfFinal.createOrReplaceTempView("vivienda_limpia")
```

A continuación:

```
%sql
SELECT * FROM vivienda_limpia;
```