FUNCIONES DE AGREGADO

**Ejercicio Se nos proporcionan una serie de datos , (James", "Sales", 3000),**
**("Michael", "Sales", 4600),**
**("Robert", "Sales", 4100),**
**("Maria", "Finance", 3000),**
**("James", "Sales", 3000),**
**("Scott", "Finance", 3300),**
**("Jen", "Finance", 3900),**
**("Jeff", "Marketing", 3000),**
**("Kumar", "Marketing", 2000),**
**("Saif", "Sales", 4100)**

**Queremos aplicar funciones de agregado en SCALA SPARK**

a. **Mostrar los datos por pantalla en forma de tabla**
b. **Calcular la media de los salarios**
c. **Saber cuantos departamentos y salarios distintos tenemos**

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

import org.apache.spark.sql.Dataset
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._

object ejemplo {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().appName("MiApp").master("local[*]").getOrCreate()
    import spark.implicits._
    val simpleData = Seq(("James", "Sales", 3000),
      ("Michael", "Sales", 4600),
      ("Robert", "Sales", 4100),
      ("Maria", "Finance", 3000),
      ("James", "Sales", 3000),
      ("Scott", "Finance", 3300),
      ("Jen", "Finance", 3900),
      ("Jeff", "Marketing", 3000),
      ("Kumar", "Marketing", 2000),
      ("Saif", "Sales", 4100)
    )

   val df = simpleData.toDF("employee_name", "department", "salary")
   // df.show()
    df.select(collect_set("salary")).show(false)
    df.select(avg("Salary")).show(false)

    val df2 = df.select(countDistinct("department", "salary"))
    df2.show(false)
    println("Distinct Count of Department & Salary: "+df2.collect()(0)(0))
```

**Ejercicio. Repetir la misma operación pero ahora vamos a mostrar:**

a. **Mostrar el primer elemento**
b. **Mostrar el último**
c. **Mostrar el salario máximo y mínimo**

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

import org.apache.spark.sql.Dataset
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._

object ejemplo {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().appName("MiApp").master("local[*]").getOrCreate()
    import spark.implicits._
    val simpleData = Seq(("James", "Sales", 3000),
      ("Michael", "Sales", 4600),
      ("Robert", "Sales", 4100),
      ("Maria", "Finance", 3000),
      ("James", "Sales", 3000),
      ("Scott", "Finance", 3300),
      ("Jen", "Finance", 3900),
      ("Jeff", "Marketing", 3000),
      ("Kumar", "Marketing", 2000),
      ("Saif", "Sales", 4100)
    )

   val df = simpleData.toDF("employee_name", "department", "salary")
   // df.show()
    /*df.select(collect_set("salary")).show(false)
    df.select(avg("Salary")).show(false)

    val df2 = df.select(countDistinct("department", "salary"))
    df2.show(false)
    println("Distinct Count of Department & Salary: "+df2.collect()(0)(0))*/
    df.select(first("salary")).show(false)
    df.select(last("salary")).show(false)
    df.select(max("salary")).show(false)
    df.select(min("salary")).show(false)




  }
```

**OPERACIÓN JOIN Se nos facillitan dos seq de datos con las que vamos a practicar las distintas estructuras de join de las que disponems en SPARKsql**

```
   val emp = Seq((1,"Smith",-1,"2018","10","M",3000),
     (2,"Rose",1,"2010","20","M",4000),
     (3,"Williams",1,"2010","10","M",1000),
     (4,"Jones",2,"2005","10","F",2000),
     (5,"Brown",2,"2010","40","",-1),
     (6,"Brown",2,"2010","50","",-1)
   )
   val dept = Seq(("Finance",10),
     ("Marketing",20),
     ("Sales",30),
     ("IT",40)
   )
```

**Aplicar: Inner, Full, right y left join**

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

import org.apache.spark.sql.Dataset
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._


object ejemplo {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().appName("MiApp").master("local[*]").getOrCreate()
    import spark.implicits._
    val emp = Seq((1,"Smith",-1,"2018","10","M",3000),
      (2,"Rose",1,"2010","20","M",4000),
      (3,"Williams",1,"2010","10","M",1000),
      (4,"Jones",2,"2005","10","F",2000),
      (5,"Brown",2,"2010","40","",-1),
      (6,"Brown",2,"2010","50","",-1)
    )


    val empDF =
emp.toDF("emp_id","name","superior_emp_id","year_joined","emp_dept_id","gender","salary")
    empDF.show(false)

    val dept = Seq(("Finance",10),
      ("Marketing",20),
      ("Sales",30),
      ("IT",40)
    )
```

```
   val deptDF = dept.toDF("dept_name","dept_id")
   //deptDF.show(false)

// inner join muestra los que tengan en las dos tablas el mismo departamento NO SALE NI 30 NI 50
   empDF.join(deptDF,empDF("emp_dept_id") ===  deptDF("dept_id"),"inner")
     .show(false)
   //empDF.join(deptDF,empDF("emp_dept_id") ===  deptDF("dept_id"),"outer")
     //.show(false)
   //empDF.join(deptDF,empDF("emp_dept_id") ===  deptDF("dept_id"),"full")
     //.show(false)
   empDF.join(deptDF,empDF("emp_dept_id") ===  deptDF("dept_id"),"fullouter")
     .show(false)

   empDF.join(deptDF,empDF("emp_dept_id") ===  deptDF("dept_id"),"left")
     .show(false)
   empDF.join(deptDF,empDF("emp_dept_id") ===  deptDF("dept_id"),"leftouter")
     .show(false)

   empDF.join(deptDF,empDF("emp_dept_id") ===  deptDF("dept_id"),"right")
     .show(false)
   empDF.join(deptDF,empDF("emp_dept_id") ===  deptDF("dept_id"),"rightouter")

  }
}
```

**Mostrar el contenido de  un fichero CSV que habremos guardado en un dataframe y que mostraremos los campos ( 20 primeros que es el valor por defecto)**
**Los 5 primeros**

```
mport org.apache.spark.SparkConf
import org.apache.spark.SparkContext

import org.apache.spark.sql.Dataset
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SparkSession


object ejemplo {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().appName("MiApp").master("local[*]").getOrCreate()
//USANDO SINTAXIS PROGRAMATICA
    val df = spark.read.format("csv")
      .option("delimiter", ",")
      .load("1800.csv")

  df.show()
df.show(2)

//USANDO SINTAXIS SQL
val df = spark.sql("SELECT * FROM csv.`1800.csv`")
```

```
    df.show()
  }
}
```

**Filtrando datos de nuestro csv**

```
import org.apache.spark.SparkConf
import org.apache.spark.SparkContext

import org.apache.spark.sql.Dataset
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SparkSession


object ejemplo {
  def main(args: Array[String]): Unit = {
    val spark = SparkSession.builder().appName("MiApp").master("local[*]").getOrCreate()
    // forma prgramatica
    /*val df = spark.read.format("csv")
      .option("delimiter", ",")
      .load("1800.csv")*/
    // froma sql
    val df = spark.sql("SELECT * FROM csv.`1800.csv`  where _c3=-75 limit 5")
    df.show()



  }
}
```