PRÁCTICA SESIÓN 3 SPARK - PROYECTO 3

El objetivo del siguiente proyecto, es evaluar lo visto en la sesión y en los ejercicios.

Sumario

Ejercicio final Repaso (Proyecto 3).....2

Ejercicio final Repaso (Proyecto 3)

Trabajamos en una empresa con las siguientes caracteristicas

Empleados

emp_id	nombre	superior_emp_id	fecha_inicio	emp_dept_id	genero	sueldo
1	Jaime	-1	2018	10	M	1300
2	Maria	1	2010	20	F	1400
3	Rosa	1	2010	10	F	1000
4	Vanesa	2	null	10	F	1800
5	Leonardo	2	2010	40	M	1300
6	Paula	2	2013	20	F	1600
7	Jose Luis	2	2014	30	M	1200

Departamentos

dept_nombre	dept_id
Financiero	10
Marketing	20
Ventas	30
Desarrollo	40

Crea los dataframes necesarios con la información anterior

Adjunto las listas para facilitar la creación:

*empleados = [(1,"Jaime",-1,"2018","10","M",1300), *
*(2,"Maria",1,"2010","20","F",1400), *
*(3,"Rosa",1,"2010","10","F",1000), *
*(4,"Vanesa",2,None,"10","F",1800), *
*(5,"Leonardo",2,"2010","40","M",1300), *
*(6,"Paula",2,"2013","20","F",1600), *
*(7,"Jose Luis",2,"2014","30","F",1200) *
]

departamento = [("Financiero",10),("Marketing",20),("Ventas",30), ("Desarrollo",40)]

```
from pyspark.sql.types import
StructField,StructType,IntegerType,StringType,FloatType

est_emp = StructType([StructField('emp_id',IntegerType(),False),
          StructField('nom',StringType(),True),
          StructField('superior_emp_id',IntegerType(),True),
          StructField('data_i',StringType(),True),
          StructField('dept_id',StringType(),True),
          StructField('genere',StringType(),True),
          StructField('sou',IntegerType(),True)
         ])
est_dep = StructType([StructField('dept_nom',StringType(),False),
          StructField('dept_id',IntegerType(),False)
        ])
df_emp = spark.createDataFrame(empleados,schema=est_emp)
df_dep = spark.createDataFrame(departamento,schema=est_dep)
```

Necesitamos lo siguiente:

1. Saber ¿Cuál es el sueldo medio de cada departamento?

```
df_emp.select("dept_id","sou").groupBy("dept_id").avg().orderBy("dept_id").show()
```

2. Eliminar las filas que tengan el campo fecha_inicio a nulo

```
df_emp.na.drop(subset= "data_i").show()
```

3. A los empleados que tienen más de 10 años de antigüedad se les va a subir el sueldo al doble calcula con una función UDF el nuevo sueldo de los empleados y añádelo a una nueva columna Sueldo_nuevo

```
from pyspark.sql.functions import udf
def doble(s):
   return 2 * s
doble_udf = udf(lambda x: doble(x),IntegerType())
df_emp.where(df_emp["data_i"]<="2011").withColumn('sou_nou',doble_udf(df_emp['sou'])).show()
```

4. De los trabajadores cuya responsable es Maria ¿Quien es la persona que más cobra?

```
df_emp.where(df_emp["superior_emp_id"]==(df_emp.where(df_emp['nom']=='Maria').select('emp_id').collect()[0][0])).orderBy(df_emp['sou'].desc()).limit(1).show()
```

Ho he intentat també amb joins, que crec que quedaria més fàcil, però em donava problemes d'àlies (a pesar que els posava)

També ho podem passar tot a SQL utilitzant vistes temporals (per a mi més còmode)

```
df_emp.createTempView("emp")

df_dep.createTempView("dep")

spark.sql("SELECT e1.emp_id, e1.nom, e1.sou FROM emp e1 INNER JOIN emp e2 ON e1.superior_emp_id=e2.emp_id WHERE e2.nom='Maria' ORDER BY e1.sou DESC LIMIT 1").show()
```

5. Crea una vista que se pueda acceder a ella desde otra sesión que muestre el total que se gasta en sueldos por departamentos

```
df_emp.join(df_dep,df_emp.dept_id==df_dep.dept_id,'INNER').select('dept_nom','sou').groupBy('dept_nom').sum().createGlobalTempView("Departaments")

spark.sql("Select * from global_temp.Departaments").show()
```

6. Se acaba de contratar una jefa de ventas, llamada Elena Martínez, va a cobrar 1700, necesitamos añadirla a nuestra base de datos

```
df_emp_nova = spark.createDataFrame([(8,"Elena Martínez",-1,"2021","30","F",1700)],schema=est_emp)

df_emp = df_emp.union(df_emp_nova)
```