Actividad modulo 47 - Big Data I - Spark

○ Configuración de plataforma Spark

In [ ]:
```
# Importar sesion de Spark
from pyspark.sql import SparkSession
```

In [ ]:
```
spark = SparkSession \
    .builder \
    .appName("actividadmodulo47")\
    .getOrCreate()
```

○ Importación de datos de Housing a una estructura Spark

In [ ]:
```
path_archivo= "D:/WORK IN PROGRESS/Data Analytics course/parte 7 spark/week 47/Housing.csv"
# Lee el archivo CSV con read.csv
# Diferente del read_csv de padnas!
df = spark.read.csv(path_archivo)
```

In [ ]:
```
# Muestra las primeras 10 filas del archivo

df.show(10)
```

```
+--------+-----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------
+----------------+
|     _c0|  _c1|     _c2|      _c3|    _c4|     _c5|      _c6|     _c7|           _c8|           _c9|   _c10|    _c11
|            _c12|
+--------+-----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------
+----------------+
|   price| area|bedrooms|bathrooms|stories|mainroad|guestroom|basement|hotwaterheating|airconditioning|parking|prefarea
|furnishingstatus|
|13300000| 7420|       4|        2|      3|     yes|       no|      no|            no|           yes|      2|     yes
|       furnished|
|12250000| 8960|       4|        4|      4|     yes|       no|      no|            no|           yes|      3|      no
|       furnished|
|12250000| 9960|       3|        2|      2|     yes|       no|     yes|            no|            no|      2|     yes
|  semi-furnished|
|12215000| 7500|       4|        2|      2|     yes|       no|     yes|            no|           yes|      3|     yes
|       furnished|
|11410000| 7420|       4|        1|      2|     yes|      yes|     yes|            no|           yes|      2|      no
|       furnished|
|10850000| 7500|       3|        3|      1|     yes|       no|     yes|            no|           yes|      2|     yes
|  semi-furnished|
|10150000| 8580|       4|        3|      4|     yes|       no|      no|            no|           yes|      2|     yes
|  semi-furnished|
|10150000|16200|       5|        3|      2|     yes|       no|      no|            no|            no|      0|      no
|     unfurnished|
| 9870000| 8100|       4|        1|      2|     yes|      yes|     yes|            no|           yes|      2|     yes
|       furnished|
+--------+-----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------
+----------------+
only showing top 10 rows
```

In [ ]:
```
# Los encabezados se estan tomando cmo si fueran datos, hay que usar header
df = spark.read.csv(path_archivo, header=True)
```

In [ ]:
```
df.show(5)
```

```
+--------+----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------+
----------------+
|   price|area|bedrooms|bathrooms|stories|mainroad|guestroom|basement|hotwaterheating|airconditioning|parking|prefarea|
furnishingstatus|
+--------+----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------+
----------------+
|13300000|7420|       4|        2|      3|     yes|       no|      no|            no|           yes|      2|     yes|
furnished|
|12250000|8960|       4|        4|      4|     yes|       no|      no|            no|           yes|      3|      no|
furnished|
|12250000|9960|       3|        2|      2|     yes|       no|     yes|            no|            no|      2|     yes|
semi-furnished|
|12215000|7500|       4|        2|      2|     yes|       no|     yes|            no|           yes|      3|     yes|
furnished|
|11410000|7420|       4|        1|      2|     yes|      yes|     yes|            no|           yes|      2|      no|
furnished|
+--------+----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------+
----------------+
only showing top 5 rows
```

In [ ]:
```
df.dtypes
```

Out[ ]:
```
[('price', 'string'),
 ('area', 'string'),
 ('bedrooms', 'string'),
 ('bathrooms', 'string'),
 ('stories', 'string'),
 ('mainroad', 'string'),
 ('guestroom', 'string'),
 ('basement', 'string'),
 ('hotwaterheating', 'string'),
 ('airconditioning', 'string'),
 ('parking', 'string'),
 ('prefarea', 'string'),
 ('furnishingstatus', 'string')]
```

In [ ]:
```
# cuenta el numero de lineas
df.count()
```

Out[ ]:
```
545
```

In [ ]:
```
# cambia el tipo de dato de string to floattype

from pyspark.sql.types import IntegerType, FloatType
```

```
df= df.withColumn('price',df.price.cast(IntegerType()))
df= df.withColumn('area',df.area.cast(IntegerType()))
df= df.withColumn('bedrooms',df.bedrooms.cast(IntegerType()))
df= df.withColumn('bathrooms',df.bathrooms.cast(IntegerType()))
df= df.withColumn('stories',df.stories.cast(IntegerType()))
```

In [ ]:
```
# Imprime la informacion de columnas de una manera jerarquica
df.printSchema()
```

```
root
 |-- price: integer (nullable = true)
 |-- area: integer (nullable = true)
 |-- bedrooms: integer (nullable = true)
 |-- bathrooms: integer (nullable = true)
 |-- stories: integer (nullable = true)
 |-- mainroad: string (nullable = true)
 |-- guestroom: string (nullable = true)
 |-- basement: string (nullable = true)
 |-- hotwaterheating: string (nullable = true)
 |-- airconditioning: string (nullable = true)
 |-- parking: string (nullable = true)
 |-- prefarea: string (nullable = true)
 |-- furnishingstatus: string (nullable = true)
```

○ Selección de datos de housing con filtros simples:

1) listado completo de columnas ordenado por price

In [ ]:
```
# Se importa el objeto functions en F1 para seleccionar la columna
import pyspark.sql.functions as F1

# Se ordena por una columna
df.sort(F1.col('price').desc()).show(10)
```

```
+--------+-----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------
+----------------+
|   price| area|bedrooms|bathrooms|stories|mainroad|guestroom|basement|hotwaterheating|airconditioning|parking|prefarea
|furnishingstatus|
+--------+-----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------
+----------------+
|13300000| 7420|       4|        2|      3|     yes|       no|      no|            no|           yes|      2|     yes
|       furnished|
|12250000| 8960|       4|        4|      4|     yes|       no|      no|            no|           yes|      3|      no
|       furnished|
|12250000| 9960|       3|        2|      2|     yes|       no|     yes|            no|            no|      2|     yes
|  semi-furnished|
|12215000| 7500|       4|        2|      2|     yes|       no|     yes|            no|           yes|      3|     yes
|       furnished|
|11410000| 7420|       4|        1|      2|     yes|      yes|     yes|            no|           yes|      2|      no
|       furnished|
|10850000| 7500|       3|        3|      1|     yes|       no|     yes|            no|           yes|      2|     yes
|  semi-furnished|
|10150000| 8580|       4|        3|      4|     yes|       no|      no|            no|           yes|      2|     yes
|  semi-furnished|
|10150000|16200|       5|        3|      2|     yes|       no|      no|            no|            no|      0|      no
|     unfurnished|
| 9870000| 8100|       4|        1|      2|     yes|      yes|     yes|            no|           yes|      2|     yes
|       furnished|
| 9800000| 5750|       3|        2|      4|     yes|      yes|      no|            no|           yes|      1|     yes
|     unfurnished|
+--------+-----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------
+----------------+
only showing top 10 rows
```

2) para las casas con mayor numero de habitaciones, calcular el promedio de precio, y tamaño en m2

In [ ]:
```python
from pyspark.sql.functions import desc

df.groupBy('bedrooms').count().orderBy(desc("count")).show()

# Los apartamentos que tienen el mayor numero de habitaciones son los de 3 habitaciones
```

```
+--------+-----+
|bedrooms|count|
+--------+-----+
|       3|  300|
|       2|  136|
|       4|   95|
|       5|   10|
|       1|    2|
|       6|    2|
+--------+-----+
```

In [ ]:
```
# Tabla temporal
df.createOrReplaceTempView("apartamentos3habitaciones")
# Comando SQL
sql_str = "select bedrooms, avg(price), avg(area) from apartamentos3habitaciones where bedrooms=3 group by 1"
# Ejecuto SQL
spark.sql(sql_str).show()
```

```
+--------+-----------------+---------+
|bedrooms|       avg(price)|avg(area)|
+--------+-----------------+---------+
|       3|4954598.133333334|  5226.62|
+--------+-----------------+---------+
```

○ Agrupamiento en Spark, por número de habitaciones y baños, del precio. Ej: # habitaciones | # baños | precio promedio, esto por furnishingstatus

In [ ]:
```
# Tabla temporal
df.createOrReplaceTempView("agrupamientos")
# Comando SQL
sql_str = "select furnishingstatus, count(bedrooms) as numero_habitaciones, count(bathrooms) as numero_banos, avg(price
# Ejecuto SQL
spark.sql(sql_str).show()
```

```
+---------------+------------------+-----------+------------------+
|furnishingstatus|numero_habitaciones|numero_banos|   precio_promedio|
+---------------+------------------+-----------+------------------+
|  semi-furnished|               227|        227|  4907524.22907489|
|     unfurnished|               178|        178|4013831.4606741574|
|       furnished|               140|        140|         5495696.0|
+---------------+------------------+-----------+------------------+
```