Actividad Módulo 47 - Big Data Parte 1

Generar un archivo .pdf que contenga las salidas:

- Configuración de plataforma Spark
- Importación de datos de Housing a una estructura de Spark
- Selección de datos de housing con filtros simples:
  1. Listado completo de columnas
  2. Para el zipcode con mayor número de casas, calcular el promedio de precio y tamaño en m2
- Agrupamiento en Spark, por zipcode, por número de habitaciones y baños, precio promedio

In [ ]:
```
from pyspark.sql import SparkSession

# Create Spark Session
spark = SparkSession\
            .builder\
            .appName('ActividadMod47')\
            .getOrCreate()
```

In [ ]:
```
# Importación de datos de Housing
df = spark.read.csv('D:/Documentos/Documentos/DataAnalysis/ebac/Python/Modulo44/kc_
df.show(10)
```

```
+----------+---------------+---------+--------+---------+-----------+--------+
------+----------+----+---------+-----+----------+-------------+--------+
------------+-------+-------+--------+------------+----------+
|        id|           date|     price|bedrooms|bathrooms|sqft_living|sqft_lot|floo
rs|waterfront|view|condition|grade|sqft_above|sqft_basement|yr_built|yr_renovated|z
ipcode|    lat|    long|sqft_living15|sqft_lot15|
+----------+---------------+---------+--------+---------+-----------+--------+
------+----------+----+---------+-----+----------+-------------+--------+
------------+-------+-------+--------+------------+----------+
|7129300520|20141013T000000|   221900|       3|        1|       1180|    5650|
1|         0|   0|        3|    7|      1180|            0|    1955|           0|
98178|47.5112|-122.257|         1340|      5650|
|6414100192|20141209T000000|   538000|       3|     2.25|       2570|    7242|
2|         0|   0|        3|    7|      2170|          400|    1951|        1991|
98125| 47.721|-122.319|         1690|      7639|
|5631500400|20150225T000000|   180000|       2|        1|        770|   10000|
1|         0|   0|        3|    6|       770|            0|    1933|           0|
98028|47.7379|-122.233|         2720|      8062|
|2487200875|20141209T000000|   604000|       4|        3|       1960|    5000|
1|         0|   0|        5|    7|      1050|          910|    1965|           0|
98136|47.5208|-122.393|         1360|      5000|
|1954400510|20150218T000000|   510000|       3|        2|       1680|    8080|
1|         0|   0|        3|    8|      1680|            0|    1987|           0|
98074|47.6168|-122.045|         1800|      7503|
|7237550310|20140512T000000|1.225e+006|       4|      4.5|       5420|  101930|
1|         0|   0|        3|   11|      3890|         1530|    2001|           0|
98053|47.6561|-122.005|         4760|    101930|
|1321400060|20140627T000000|   257500|       3|     2.25|       1715|    6819|
2|         0|   0|        3|    7|      1715|            0|    1995|           0|
98003|47.3097|-122.327|         2238|      6819|
|2008000270|20150115T000000|   291850|       3|      1.5|       1060|    9711|
1|         0|   0|        3|    7|      1060|            0|    1963|           0|
98198|47.4095|-122.315|         1650|      9711|
|2414600126|20150415T000000|   229500|       3|        1|       1780|    7470|
1|         0|   0|        3|    7|      1050|          730|    1960|           0|
98146|47.5123|-122.337|         1780|      8113|
|3793500160|20150312T000000|   323000|       3|      2.5|       1890|    6560|
2|         0|   0|        3|    7|      1890|            0|    2003|           0|
98038|47.3684|-122.031|         2390|      7570|
+----------+---------------+---------+--------+---------+-----------+--------+
------+----------+----+---------+-----+----------+-------------+--------+
------------+-------+-------+--------+------------+----------+
only showing top 10 rows
```

In [ ]:

```
# Selección de datos de housing
# 1) Listado completo de columnas
df.printSchema()
```

```
root
 |-- id: string (nullable = true)
 |-- date: string (nullable = true)
 |-- price: string (nullable = true)
 |-- bedrooms: string (nullable = true)
 |-- bathrooms: string (nullable = true)
 |-- sqft_living: string (nullable = true)
 |-- sqft_lot: string (nullable = true)
 |-- floors: string (nullable = true)
 |-- waterfront: string (nullable = true)
 |-- view: string (nullable = true)
 |-- condition: string (nullable = true)
 |-- grade: string (nullable = true)
 |-- sqft_above: string (nullable = true)
 |-- sqft_basement: string (nullable = true)
 |-- yr_built: string (nullable = true)
 |-- yr_renovated: string (nullable = true)
 |-- zipcode: string (nullable = true)
 |-- lat: string (nullable = true)
 |-- long: string (nullable = true)
 |-- sqft_living15: string (nullable = true)
 |-- sqft_lot15: string (nullable = true)
```

In [ ]:

```python
# Para el zipcode con mayor número de casas, calcular el promedio de precio y tamañ
from pyspark.sql.types import FloatType, IntegerType, BooleanType

# Cast columns as the correct data type
df = df.withColumn('price', df.price.cast(FloatType()))
df = df.withColumn('bedrooms', df.bedrooms.cast(IntegerType()))
df = df.withColumn('bathrooms', df.bathrooms.cast(FloatType()))

df = df.withColumn('sqft_living', df.sqft_living.cast(IntegerType()))
df = df.withColumn('sqft_lot', df.sqft_lot.cast(IntegerType()))
df = df.withColumn('floors', df.floors.cast(IntegerType()))
df = df.withColumn('waterfront', df.waterfront.cast(BooleanType()))
df = df.withColumn('view', df.view.cast(BooleanType()))
df = df.withColumn('condition', df.condition.cast(IntegerType()))
df = df.withColumn('grade', df.grade.cast(IntegerType()))
df = df.withColumn('sqft_above', df.sqft_above.cast(IntegerType()))
df = df.withColumn('sqft_basement', df.sqft_basement.cast(IntegerType()))

df = df.withColumn('lat', df.lat.cast(FloatType()))
df = df.withColumn('long', df.long.cast(FloatType()))

df = df.withColumn('sqft_living15', df.sqft_living15.cast(IntegerType()))
df = df.withColumn('sqft_lot15', df.sqft_lot15.cast(IntegerType()))
```

In [ ]:
```
# Para el zipcode con mayor número de casas, calcular el promedio de precio y tamañ
df.createOrReplaceTempView('KC_HOUSING')

sql_str = """
            select  zipcode,
                    count(distinct id) as id_count,
                    avg(price) as avg_price,
                    avg(sqft_living) * 0.0929 as sqft_living_m2
            from KC_HOUSING
            group by zipcode
            order by count(distinct id) desc
            limit 3
"""
spark.sql(sql_str).show(10)
```

```
+-------+--------+-----------------+------------------+
|zipcode|id_count|        avg_price|    sqft_living_m2|
+-------+--------+-----------------+------------------+
|  98103|     600|584919.2109634551|153.36215946843853|
|  98038|     587|         366867.6|199.52274711864408|
|  98115|     576|619900.5471698113|  170.498907890223|
+-------+--------+-----------------+------------------+
```

In [ ]:
```
# Agrupamiento en Spark, por zipcode, por número de habitaciones y baños, precio pr

df.createOrReplaceTempView('KC_HOUSING')

sql_str = """
            select  zipcode,
                    bedrooms,
                    bathrooms,
                    avg(price) as avg_price
            from KC_HOUSING
            group by zipcode, bedrooms, bathrooms
            order by 1,2,3
"""

spark.sql(sql_str).show(20)
```

```
+-------+--------+---------+------------------+
|zipcode|bedrooms|bathrooms|         avg_price|
+-------+--------+---------+------------------+
|  98001|       0|      0.0|          139950.0|
|  98001|       1|      1.0|          166000.0|
|  98001|       1|      2.0|          171000.0|
|  98001|       2|      1.0|197428.57142857142|
|  98001|       2|      1.5|          350000.0|
|  98001|       2|     1.75|          246112.5|
|  98001|       2|      2.5|          214100.0|
|  98001|       2|     2.75|          239475.0|
|  98001|       3|     0.75|          363000.0|
|  98001|       3|      1.0|205182.80952380953|
|  98001|       3|      1.5|          224108.5|
|  98001|       3|     1.75| 260531.0810810811|
|  98001|       3|      2.0|256841.42857142858|
|  98001|       3|     2.25|          265999.0|
|  98001|       3|      2.5| 308581.8604651163|
|  98001|       3|     2.75|          255000.0|
|  98001|       3|      3.0|          309500.0|
|  98001|       4|      1.0|          229790.0|
|  98001|       4|      1.5|246406.85714285713|
|  98001|       4|     1.75| 251114.2857142857|
+-------+--------+---------+------------------+
only showing top 20 rows
```