AWS Glue Studio Notebook

You are now running a AWS Glue Studio notebook; To start using your notebook you need to start an AWS Glue Interactive Session.

Optional: Run this cell to see available notebook commands ("magics").

In [ ]:
```
%help
```

Run this cell to set up and start your interactive session.

In [ ]:
```
%idle_timeout 2880
%glue_version 3.0
%worker_type G.1X
%number_of_workers 5

import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
```

```
You are already connected to a glueetl session bee23146-fdf9-48dc-8695-d672caa6254d.

No change will be made to the current session that is set as glueetl. The session configuration change will apply to newly creat
ed sessions.
Current idle_timeout is 2880 minutes.
idle_timeout has been set to 2880 minutes.
You are already connected to a glueetl session bee23146-fdf9-48dc-8695-d672caa6254d.

No change will be made to the current session that is set as glueetl. The session configuration change will apply to newly creat
ed sessions.
Setting Glue version to: 3.0
```

```
You are already connected to a glueetl session bee23146-fdf9-48dc-8695-d672caa6254d.

No change will be made to the current session that is set as glueetl. The session configuration change will apply to newly creat
ed sessions.
Previous worker type: G.1X
Setting new worker type to: G.1X
You are already connected to a glueetl session bee23146-fdf9-48dc-8695-d672caa6254d.

No change will be made to the current session that is set as glueetl. The session configuration change will apply to newly creat
ed sessions.
Previous number of workers: 5
Setting new number of workers to: 5
```

○Exploración de la información de housing utilizando Python, y obteniendo la información del ejercicio anterior

Example: Create a DynamicFrame from a table in the AWS Glue Data Catalog and display its schema

In [ ]:
```
dyf = glueContext.create_dynamic_frame.from_catalog(database='ebac-dl-housing', table_name='module_49')
dyf.printSchema()
```

```
root
|-- price: long
|-- area: long
|-- bedrooms: long
|-- bathrooms: long
|-- stories: long
|-- mainroad: string
|-- guestroom: string
|-- basement: string
|-- hotwaterheating: string
|-- airconditioning: string
|-- parking: long
|-- prefarea: string
|-- furnishingstatus: string
```

Example: Convert the DynamicFrame to a Spark DataFrame and display a sample of the data

In [ ]:
```
df = dyf.toDF()
df.show()
```

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
9800000	13200	3	1	2	yes	no	yes	no	yes	2	yes	furnished
9681000	6000	4	3	2	yes	yes	yes	yes	no	2	no	semi-furnished
9310000	6550	4	2	2	yes	no	no	no	yes	1	yes	semi-furnished
9240000	3500	4	2	2	yes	no	no	yes	no	2	no	furnished
9240000	7800	3	2	2	yes	no	no	no	no	0	yes	semi-furnished
9100000	6000	4	1	2	yes	no	yes	no	no	2	no	semi-furnished
9100000	6600	4	2	2	yes	yes	yes	no	yes	1	yes	unfurnished
8960000	8500	3	2	4	yes	no	no	no	yes	2	no	furnished
8890000	4600	3	2	2	yes	yes	no	no	yes	2	no	furnished

```
| 8855000| 6420|      3|       2|      2|    yes|      no|      no|             no|            yes|      1|    yes|  semi-f
urnished|
+--------+-----+-------+--------+-------+-------+--------+--------+---------------+---------------+-------+-------+--------
--------+
only showing top 20 rows
```

○ 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|furnishi
ngstatus|
+--------+-----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------+--------
--------+
|13300000| 7420|       4|        2|      3|     yes|       no|      no|            no|           yes|      2|     yes|       f
urnished|
|12250000| 9960|       3|        2|      2|     yes|       no|     yes|            no|            no|      2|     yes|  semi-f
urnished|
|12250000| 8960|       4|        4|      4|     yes|       no|      no|            no|           yes|      3|      no|       f
urnished|
|12215000| 7500|       4|        2|      2|     yes|       no|     yes|            no|           yes|      3|     yes|       f
urnished|
|11410000| 7420|       4|        1|      2|     yes|      yes|     yes|            no|           yes|      2|      no|       f
urnished|
|10850000| 7500|       3|        3|      1|     yes|       no|     yes|            no|           yes|      2|     yes|  semi-f
urnished|
|10150000|16200|       5|        3|      2|     yes|       no|      no|            no|            no|      0|      no|     unf
urnished|
|10150000| 8580|       4|        3|      4|     yes|       no|      no|            no|           yes|      2|     yes|  semi-f
urnished|
| 9870000| 8100|       4|        1|      2|     yes|      yes|     yes|            no|           yes|      2|     yes|       f
urnished|
| 9800000| 5750|       3|        2|      4|     yes|      yes|      no|            no|           yes|      1|     yes|     unf
urnished|
+--------+-----+--------+---------+-------+--------+---------+--------+--------------+--------------+-------+--------+--------
--------+
only showing top 10 rows
```

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

In [ ]:
```
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|
|       6|    2|
|       1|    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) as preci
# Ejecuto SQL
spark.sql(sql_str).show()
```

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

○Incluir 3 análisis adicionales seleccionados por el estudiante, que respondan a preguntas que el negocio quisiera hacer. Incluir KPIs y datos que permitan a una persona sin conocer el negocio a fondo, dares cuenta de sus magnitudes (e.g. promedio del tamaño de una casa)

In [ ]:
```
from pyspark.sql.functions import avg
avg_area = df.agg(avg('area')).collect()[0][0]

avg_price =df.agg(avg('price')).collect()[0][0]

print(f"The average of the '{'area'}' is: {avg_area}")

print(f"The average of the '{'price'}' is: {avg_price}")
```

```
The average of the 'area' is: 5150.54128440367
The average of the 'price' is: 4766729.247706422
```

In [ ]:
```
some_columns= ['bedrooms','bathrooms','stories','mainroad','guestroom','basement','hotwaterheating','airconditioning','parking',

for column in some_columns:
    df_helper = df.groupBy(column).count().orderBy(desc("count"))
    print('El numero de ocurrencias que se tienen para la variable ',column, ' es:')
    df_helper.show()
```

```
El numero de ocurrencias que se tienen para la variable  bedrooms  es:
+--------+-----+
|bedrooms|count|
+--------+-----+
|       3|  300|
|       2|  136|
|       4|   95|
|       5|   10|
|       6|    2|
|       1|    2|
+--------+-----+

El numero de ocurrencias que se tienen para la variable  bathrooms  es:
+---------+-----+
|bathrooms|count|
+---------+-----+
|        1|  401|
|        2|  133|
|        3|   10|
|        4|    1|
+---------+-----+

El numero de ocurrencias que se tienen para la variable  stories  es:
+-------+-----+
|stories|count|
+-------+-----+
|      2|  238|
|      1|  227|
|      4|   41|
|      3|   39|
+-------+-----+

El numero de ocurrencias que se tienen para la variable  mainroad  es:
+--------+-----+
|mainroad|count|
+--------+-----+
|     yes|  468|
|      no|   77|
+--------+-----+

El numero de ocurrencias que se tienen para la variable  guestroom  es:
+---------+-----+
|guestroom|count|
+---------+-----+
```

```
|       no|  448|
|      yes|   97|
+---------+-----+

El numero de ocurrencias que se tienen para la variable  basement  es:
+--------+-----+
|basement|count|
+--------+-----+
|      no|  354|
|     yes|  191|
+--------+-----+

El numero de ocurrencias que se tienen para la variable  hotwaterheating  es:
+---------------+-----+
|hotwaterheating|count|
+---------------+-----+
|             no|  520|
|            yes|   25|
+---------------+-----+

El numero de ocurrencias que se tienen para la variable  airconditioning  es:
+---------------+-----+
|airconditioning|count|
+---------------+-----+
|             no|  373|
|            yes|  172|
+---------------+-----+

El numero de ocurrencias que se tienen para la variable  parking  es:
+-------+-----+
|parking|count|
+-------+-----+
|      0|  299|
|      1|  126|
|      2|  108|
|      3|   12|
+-------+-----+

El numero de ocurrencias que se tienen para la variable  prefarea  es:
+--------+-----+
|prefarea|count|
+--------+-----+
|      no|  417|
|     yes|  128|
```

```
+--------+-----+

El numero de ocurrencias que se tienen para la variable  furnishingstatus  es:
+----------------+-----+
|furnishingstatus|count|
+----------------+-----+
|  semi-furnished|  227|
|     unfurnished|  178|
|       furnished|  140|
+----------------+-----+
```

In [ ]: