Los resultados se escriben en la sintaxis de Mongo Shell.

For the owners with more than one apartment, show only price, rooms, description and location.

```
[
 {
  $match:
   {
    host_total_listings_count: {
     $gt: 1
    }
   }
 },
 {
  $project:
   {
    _id: 0,
    name: 1,
    price: 1,
    description: 1,
    location: {
     latitude: "$latitude",
     longitude: "$longitude"
    }
   }
 }
]
```

Yo he interpretado que se deben mostrar los apartamentos de aquellos hosts con un host_total_listing_count mayor que 1 y mostrar para cada apartamento el nombre, precio, rescripción y ubicación (latitud y longitud). Para ello diseño un pipeline con 2 etapas. La primera para filtrar aquellos hosts que tienen más de una propiedad, he supuesto que se “host_total_list_count” es el valor scrapeado de las propiedades obtenidas. Para la segunda etapa se enseñan el nombre, descripción y ubicación de la propiedad.

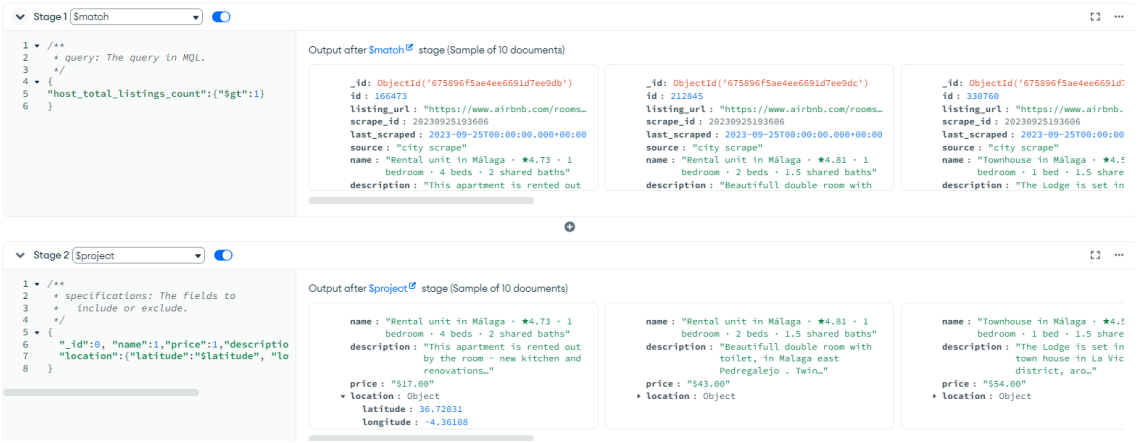

Figura 1Resultado del pipeline de la primera consulta.

Create a new collection with the prices ~~in pounds~~ changed to euros as numbers.

```
[
 {
  $addFields: {
   price: {
    $convert: {
     input: {
      $substr: [
       "$price",
       1,
       {
        $strLenCP: "$price"
       }
      ]
     },
     to: "double",
     onError: null,
     onNull: null
    }
   }
  }
 },
 {
  $addFields: {
   price: {
    $multiply: [
     "$price",
     0.95 // Assuming conversion rate $1 = €1.05
    ]
   }
  }
 },
 {
  $merge: {
   into: "hosting_with_price_euros",
   whenMatched: "merge",
   whenNotMatched: "insert"
  }
 }
]
```

Los precios guardados en el script tienen un símbolo de un dólar “$” y no de libra “£”. Así que he supuesto para crear una colección con precios en euros, pero se usa la conversión de dólares a euros. Por lo que se multiplicará por 0.95. En caso de que

el símbolo sea incorrecto en los datos scrapeados en el JSON. Simplemente habría que cambiar la conversión por el de las libras.

Figura 2 Valor del euro respecto al dólar.

Figura 3 Valor del euro respecto a la libra

1. En la primera etapa convertimos “Price” de cadena de texto a un número. Para ello usamos el operador “$convert “. Para no incluir el primer valor de la cadena de texto (que posee el símbolo $) usamos “$substr” que obtiene una sub-cadena desde la posición 1 de la cadena de texto (empieza desde 0) hasta la última posición de la sub cadena. (Se usa srtLenCP para obtener la longitud de la cadena de texto).

Figura 4 Primera etapa del pipeline

2. Pasamos de dólares a euros. Para ello multiplicamos todos los valores en este caso por 0.95. En caso de que fueran libras y el valor guardado en la base de datos tenga un símbolo incorrecto habría que múltiplicar por 1.2 (*Figura 3)*.

Figura 5 Segunda etapa de la agregación.

3. Por último, como queremos crear una nueva colección usamos un merge.

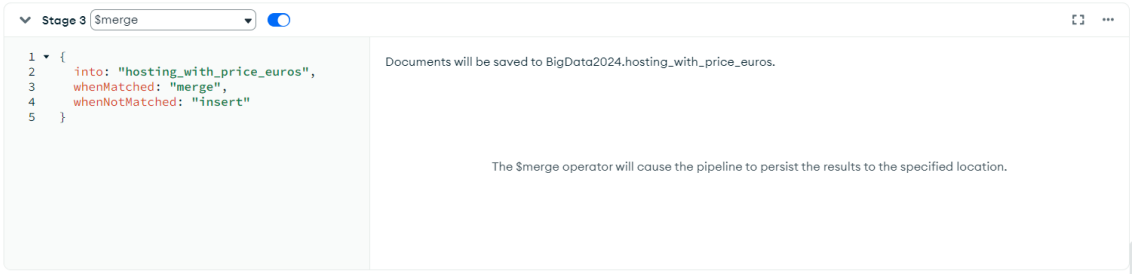

Figura 6 Se crea una colección llamada hosting_with_price_euros

Join the original collection with the one with prices in euros to add this price to the original collection.

```
[
 {
  $lookup: {
   from: "hosting_with_price_euros",
   localField: "id",
   foreignField: "id",
   as: "hosting_with_price_info"
  }
 },
 {
  $unwind: "$hosting_with_price_info"
 },
 {
  $addFields: {
   price_in_euros:
    "$hosting_with_price_info.price"
  }
 },
 {
  $project:
   {
    price: 1,
    price_in_euros: 1,
    name: 1,
    description: 1,
    host_id: 1,
    host_name: 1
   }
 }
]
```

Se divide en 4 etapas este pipeline:

1. Se realiza un LEFT JOIN con el operador $lookup: entre la colección actual y hosting_with_price_euros, con base en el campo id (cualquier campo único bastaría, por ejemplo “listing_url”).

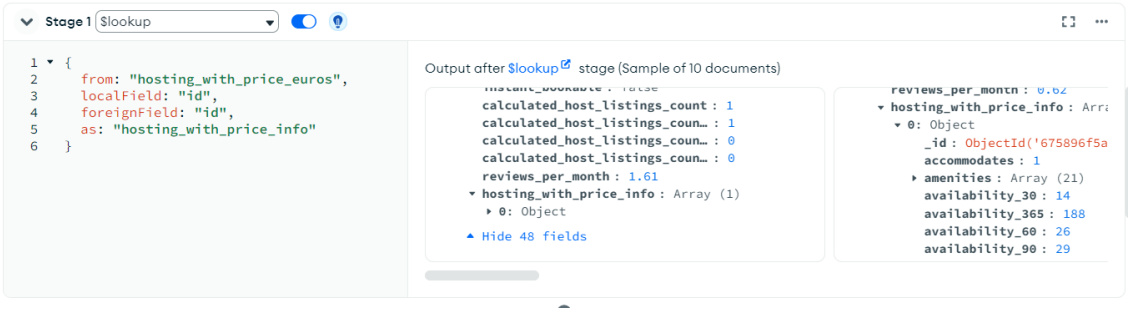

Figura 7 Uso de $lookup para unir tablas. El resultado está en el index 0 del array resultante.

2. Despliega el array resultante de $lookup en documentos individuales utilizando el operador $unwind.

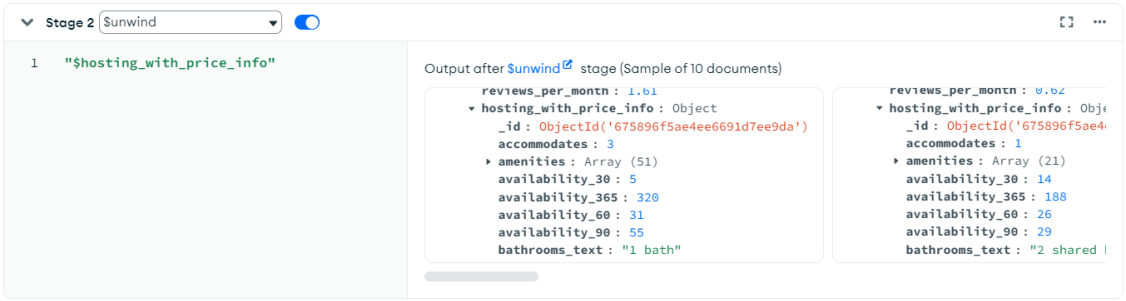

Figura 8 Gracias al $unwind ahora el hosting_with_price_info es un Object.

3. Se usa $addFields para añadir el campo price_in_euros.

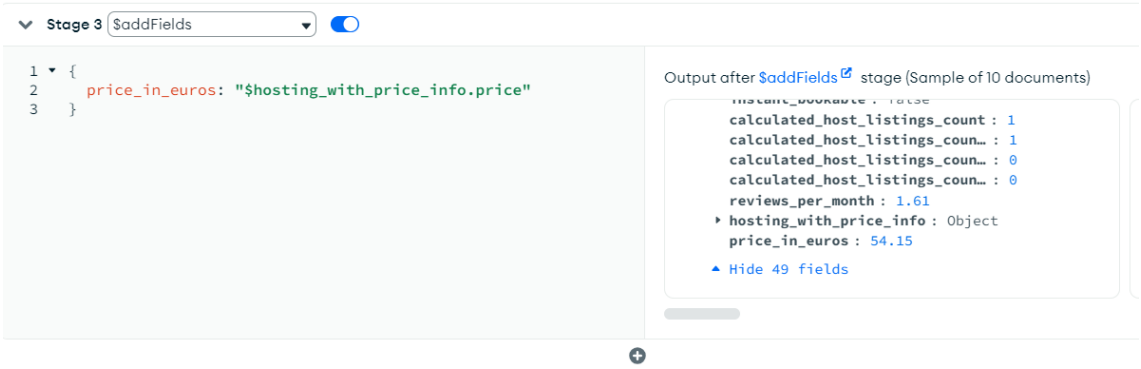

Figura 9 Añadimos el campo price_in_euros procedente del objeto hosting_with_price_info

4. Por último, con $project se seleciona os campos que campos mostramos al final.

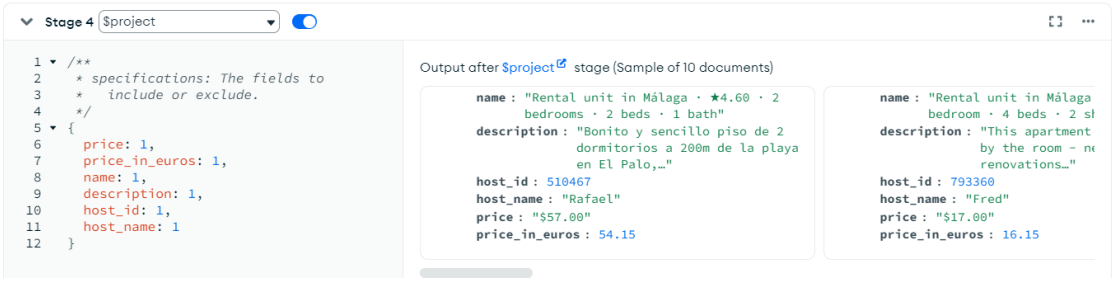

Figura 10 Se muestran el campo de price y price_in_euros en la proyección.

What is the average price in each neighbourhood?

```
[
 {
  $group: {
   _id: "$neighbourhood",
   averagePrice: {
    $avg: "$price"
   }
  }
 }
]
```

Partiendo de la base de que la colección ahora tiene el precio en euros. Se debe agrupar por vecindario y ver la media. El id de la agrupación en este caso será el vecindario “neighbourhood”.

Figura 11 Resultado del Group By.

Find the 10 cheaper owners by the average price of their aparments

```
[
 {
  $match: {
   price: { $ne: null }
  }
 },
 {
  $group: {
   _id: {
    neighbourhood: "$neighbourhood",
    host: "$host_id"
   },
   hosts: {
    $addToSet: {
     name: "$host_name",
     id: "$host_id"
    }
   },
   averagePrice: {
    $avg: "$price"
   }
  }
 },
 {
  $sort: {
   averagePrice: 1
  }
 },
 {
  $project: {
   _id: 0,
   host_id: {
    $first: "$hosts.id"
   },
   host_name: {
    $first: "$hosts.name"
   },
   averagePrice: 1
  }
 },
 {
  $limit: 10
 }

]
```

Partiendo de la base de los anteriores ejercicios tengo un campo llamado “Price” con un double.

1. En primer lugar, filtro por todos los precios que no sean nulos.

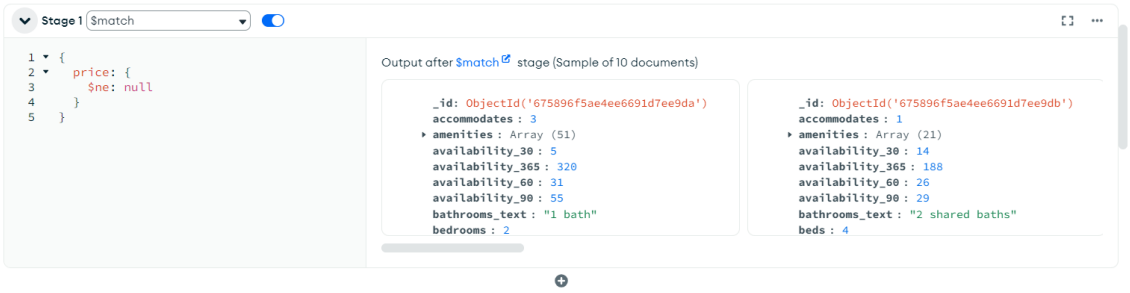

Figura 12 Obtengo todos los precios distintos de nulo.

2. Agrupo por vecindario y el ide de host. Utilizo el operador “$addToSet” para añadir los hosts. Calculo el promedio con el operador “$avg”

Figura 13 Agrupación

3. Se ordena el promedio de forma ascendente.

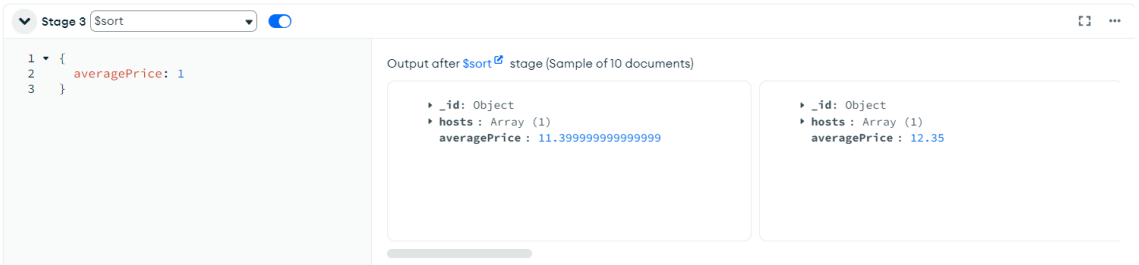

Figura 14 Sort de forma ascendente

4. Utilizo el “$project” para mostrar el id, nombre y precio medio de cada host.

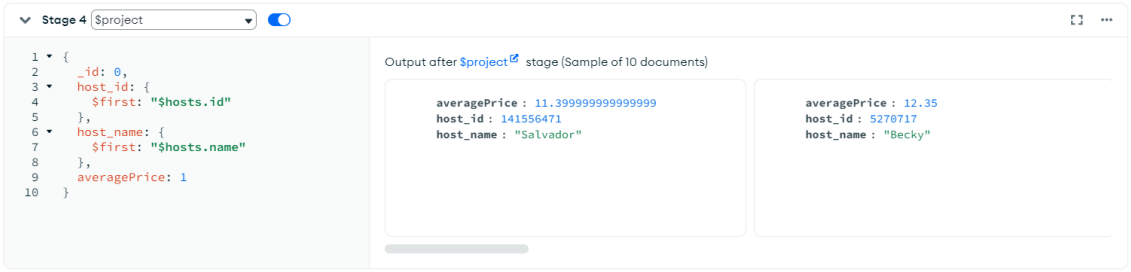

Figura 15 Proyección para mostrar el precio medio y el host.

5. Como me interesan solo los 10 mas baratos, utilizo el operador $limit y me quedo con esos hosts.