Lab: Set Up An Aggregation Pipeline

In this lab, we are going to be using aggregation to search through a database of sample Airbnb listings, and find out if there is a place we can stay **in the US**, that **has a pool**, and costs **less than $240/night**.

For this lab, we will assume that you have set up a free-tier database on MongoDB Atlas, and that you are familiar with the Mongo Query Language (MQL) syntax that allows us to work with the BSON documents in our collections.

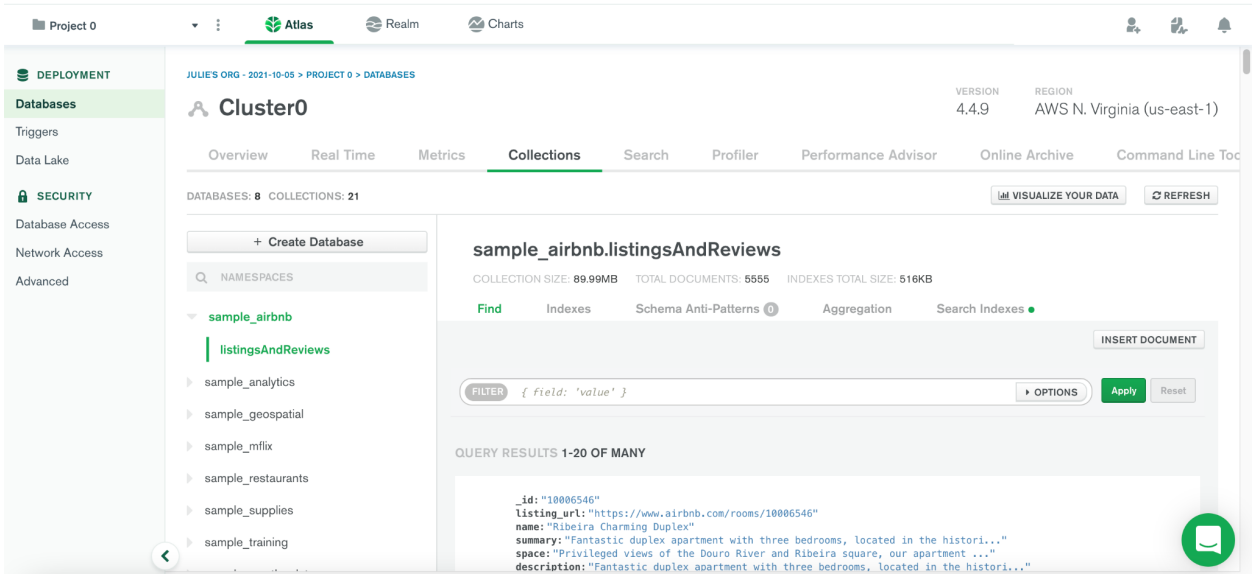
When setting up a new collection, Mongo provides the option to load in some sample datasets. We will be using the Mongo-provided Airbnb dataset.

Don't worry if the MQL syntax has been taking a while to get the hang of - it's always a little tricky dealing with syntax that has a lot of brackets to keep track of.

Step 1: Load in a sample dataset

When setting up a new collection in MongoDB, you will be asked if you wish to load sample data. You can also follow [these instructions] to load sample data into your database later.

It should look like this when correctly set up:



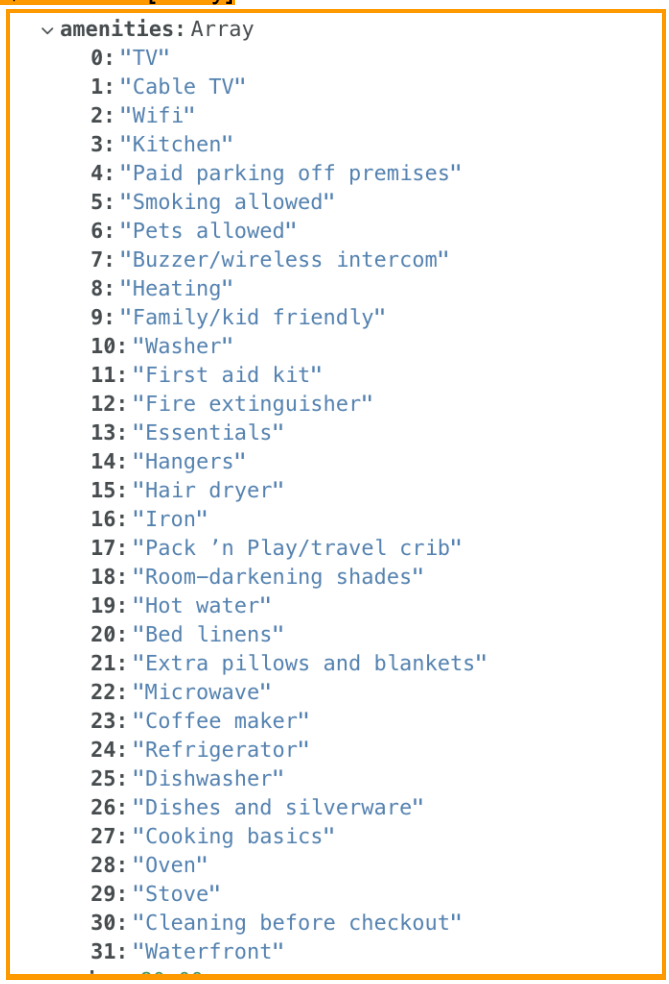
Step 2: Let's take a look at how the collection is structured.

We know that the pieces of information that we're going to be looking for in this lab are

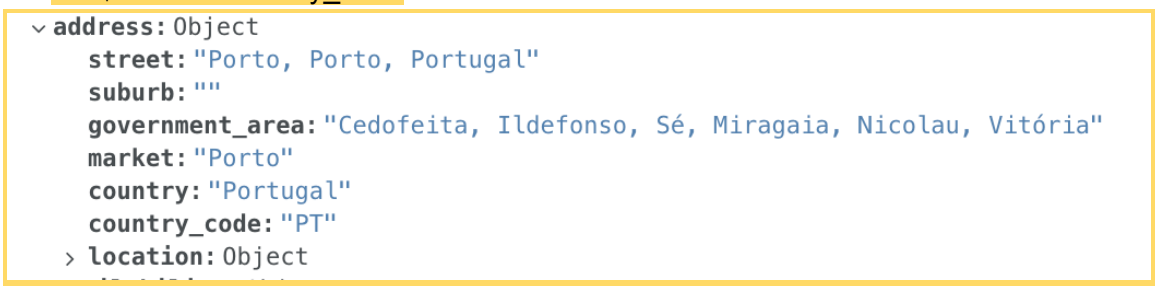
- Whether or not there is a pool
- In which country is this property located
- Cost per night

Which means that the keys that we will be looking at in our data are:

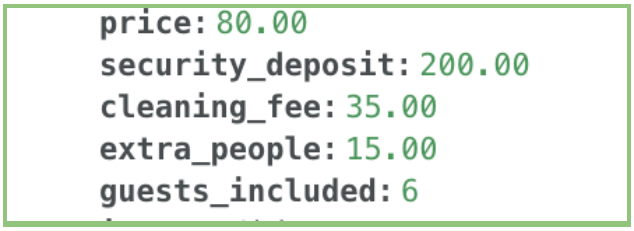
- $amenities[Array]



- $address.country_code



- $price + $cleaning_fee (because we all know this is where they get you)



Step 3: Plan which steps we will need to add to our aggregation pipeline

We are going to be filtering three times: First to keep only properties located in the US, second to keep only properties below $240 with the price + the cleaning fee, and third to find only those properties with a pool.

We will primarily be leaning on the $match operation to help us filter based on criteria being met in each document. First, we need to keep only properties that have a country_code in the address object that matches “US”

```
{ $match: { address.country_code: "US" } }
```

And we know that we will need to create a way to reference the real price for a one-night stay, which is the price + the cleaning cost, so we will add a new field.

```
{ $addFields: {total_price : { $add: [“$price”, "$cleaning_fee"] } } }
```

Then we will need to filter out the properties that cost more than $240 to stay for one night, so we will use $match again, as well as the less-than-or-equal-to operator.

```
{ $match: { “total_price”: {$lte: 240 } }
```

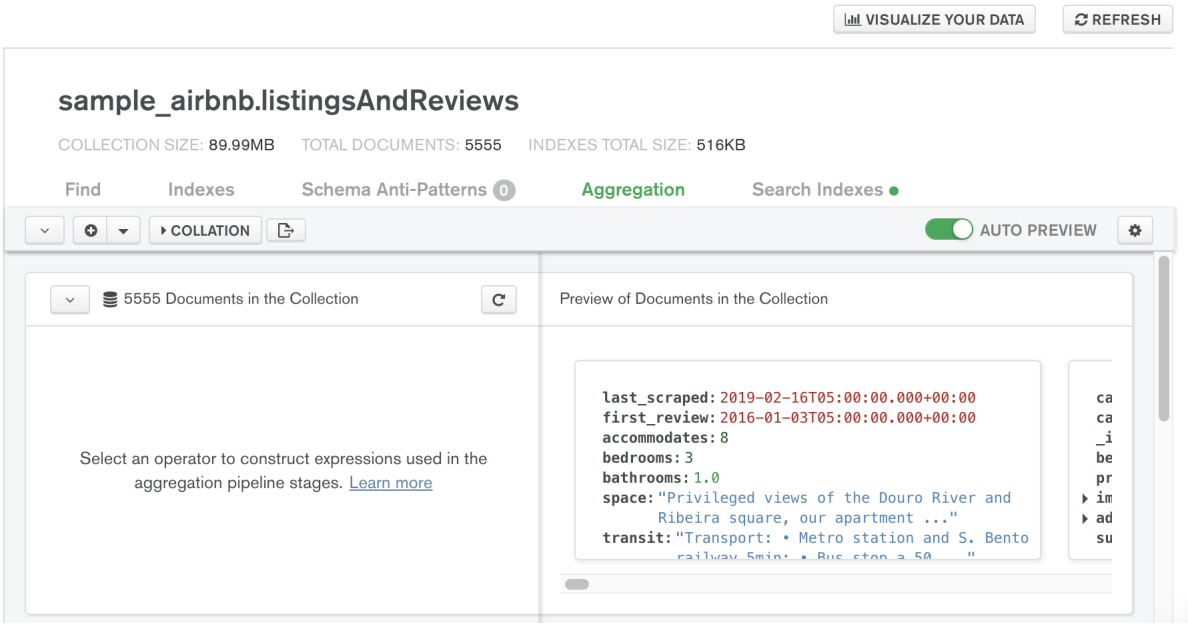
Because the $match operation cannot directly help us find an item that could be anywhere in an array, we will also use the $addFields operation to create a field to see if a property has “Pool” in it’s amenities array and be able to filter again by that.

```
{ $addFields: {has_pool : { $in: [“Pool”, "$amenities"] } } }
```

And we then want to use $match to filter once more, and hope that we have a document left that matches all of these criteria!

```
{ $match: { “has_pool”: true }
```

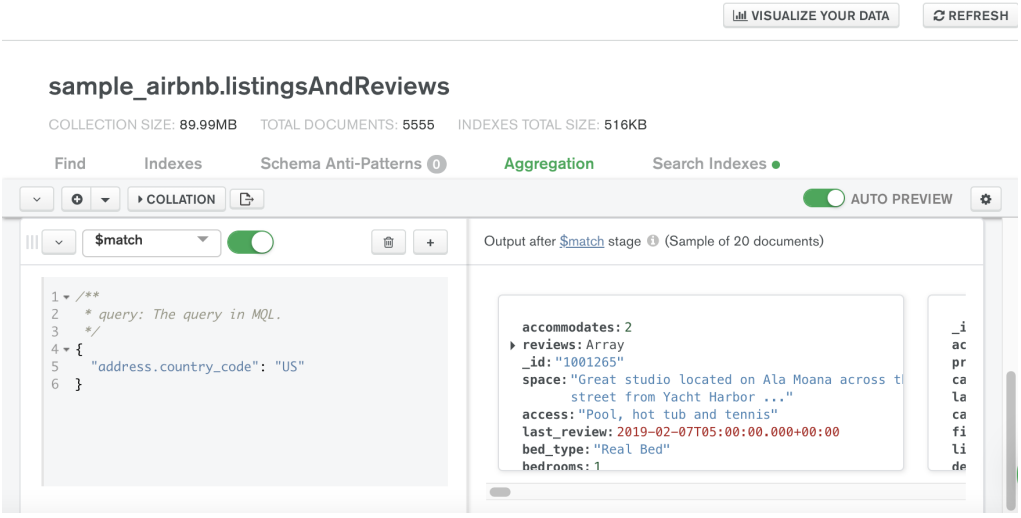
Step 4: Find the Aggregation tab in our collection view on MongoDB Atlas. It will also show you a preview of the documents in the collection.



Step 5: Add our first stage to filter down to only properties located in the US to our aggregate pipeline in MongoDB Atlas.

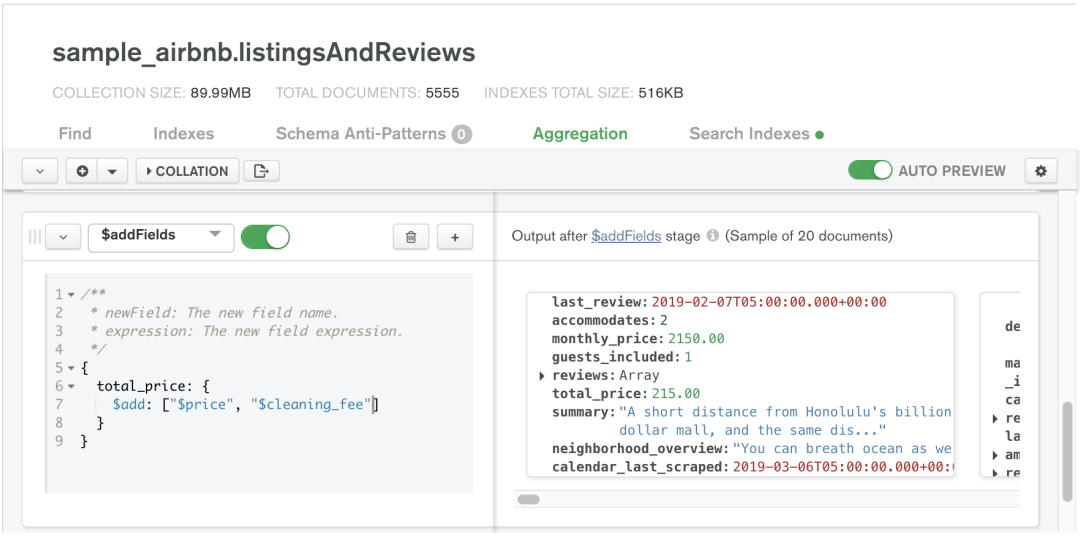
We're going to create a $match operation for the first stage. The MQL that we will put in the textbox is:

```
{
  "address.country_code": "US"
}
```



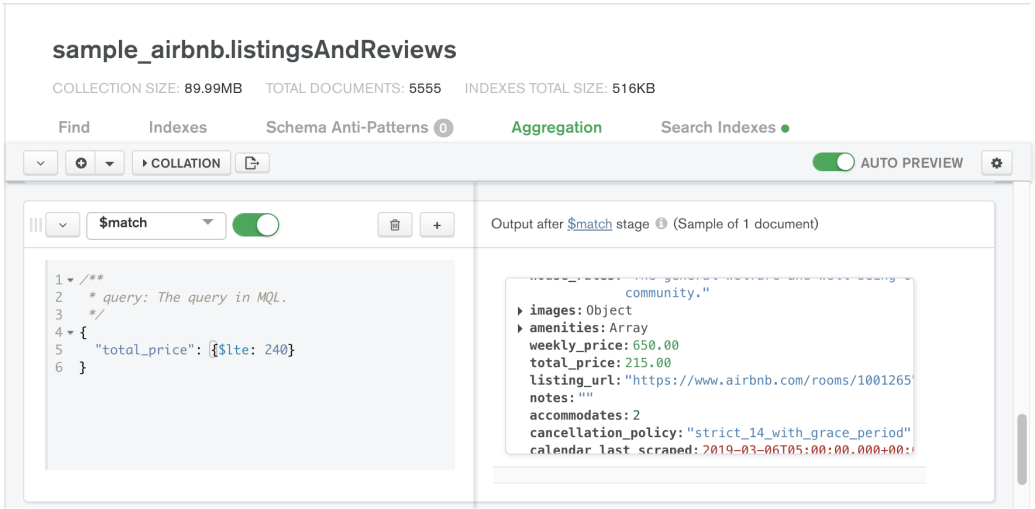
Step 6: Add a field so that we can see the price for a one-night stay, by adding the $price and the $cleaning_fee

```
{
  total_price: {
    $add: ["$price", "$cleaning_fee"]
  }
}
```



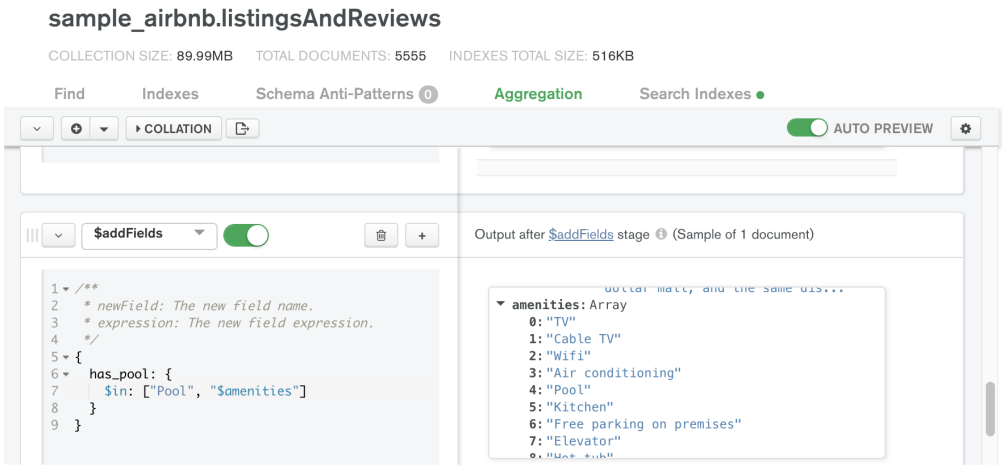
Step 7: Filter down to only properties that cost less than $240/night by using a $match function and the $lte (less-than-or-equal-to) operator on the $total_price field

```
{
  "total_price": {$lte: 240}
}
```



Step 8: Create a field to be able to check if a property has a pool listed in its amenities using the $in operator

```
{
  has_pool: {
    $in: ["Pool", "$amenities"]
  }
}
```



Step 9: Filter for only properties that have a pool by using $match on the $has_pool field

```
{
  "has_pool": true
}
```

Step 10: Check out our results and start planning a vacation to HAWAII!! Only one property in the database matched our criteria of being in the US, less than $240/night, and having a pool, and it sounds like a great Airbnb to visit!