**MongoDB Mini-project #4**

**Project Outline**

1. Access the timelog in the Google sheet for your team (see your email for the link) and sign in every time you work on this project. Sign in right now and create your first entry.
2. Answer the two data demands given below by writing MongoDB aggregation pipeline queries in Compass. Follow the instructions.
3. Create a new data demand and solution (query and example result with total number of documents) that utilizes all three data sets. Your data demand should be an English sentence and your query should utilize the aggregation pipeline in MongoDB Compass. Your new data demand should be different than the ones I have written (examples and assigned queries.)
4. If you are working with a partner, you must also complete a second original data demand and solution (same as the previous step). If you are working by yourself, you can skip the final data demand (stop at activity 3 above).

**Data Demands**

Notes about the data:

* *CAMIS* is the same as *restaurant\_id*
* *JURISDICTION NAME* is the same as *zipcode*

**Data Demand 1**

This is the high-level, English description of the final result:

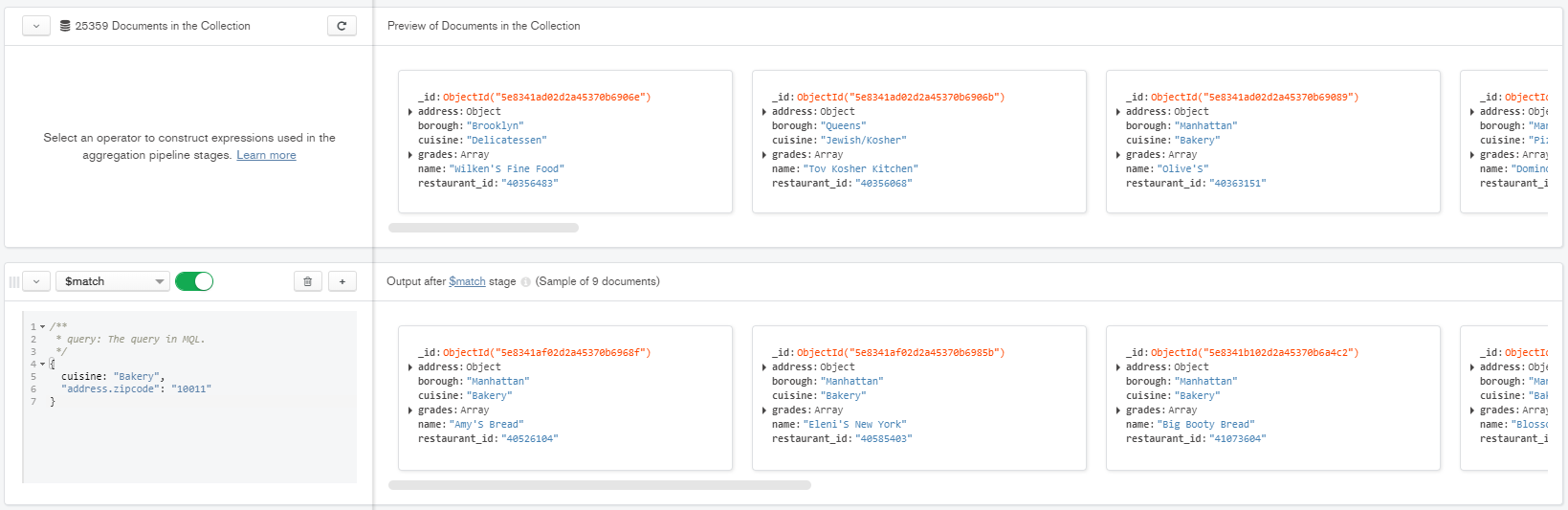
Give the names of bakeries located in zipcode 10011 that have at least one mouse violation in their inspections.

Unfortunately, we do not have the bandwidth to write this as one query. I have separated it into sequential steps that I was able to execute. Do the steps in sequence and do screen captures of your results.

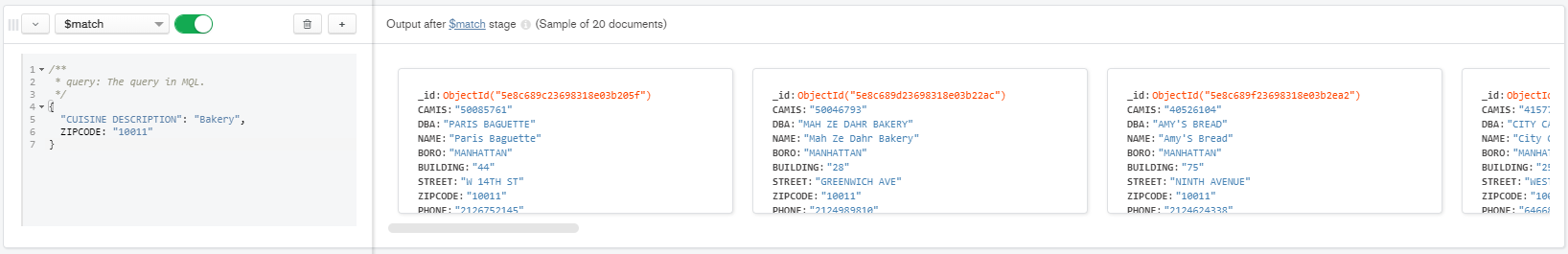
Give screenshots for each activity below (show the MongoDB Compass stage and results.)

We will first create two smaller collections using the aggregation pipeline and *SAVE/Create View* option:

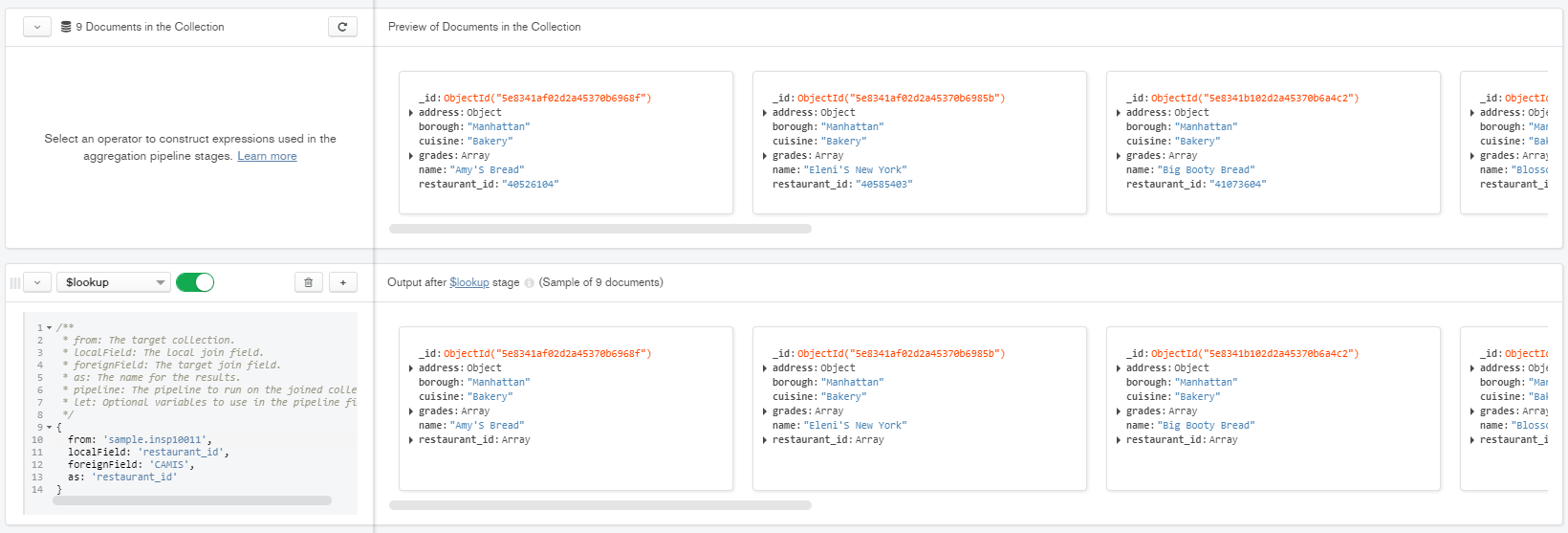
1. Create a view on the *restaurants* collection of bakeries (*cuisine* = “Bakery”) with *address.zipcode* = “10011”. Call this view *bakery10011*. It should have 9 documents.



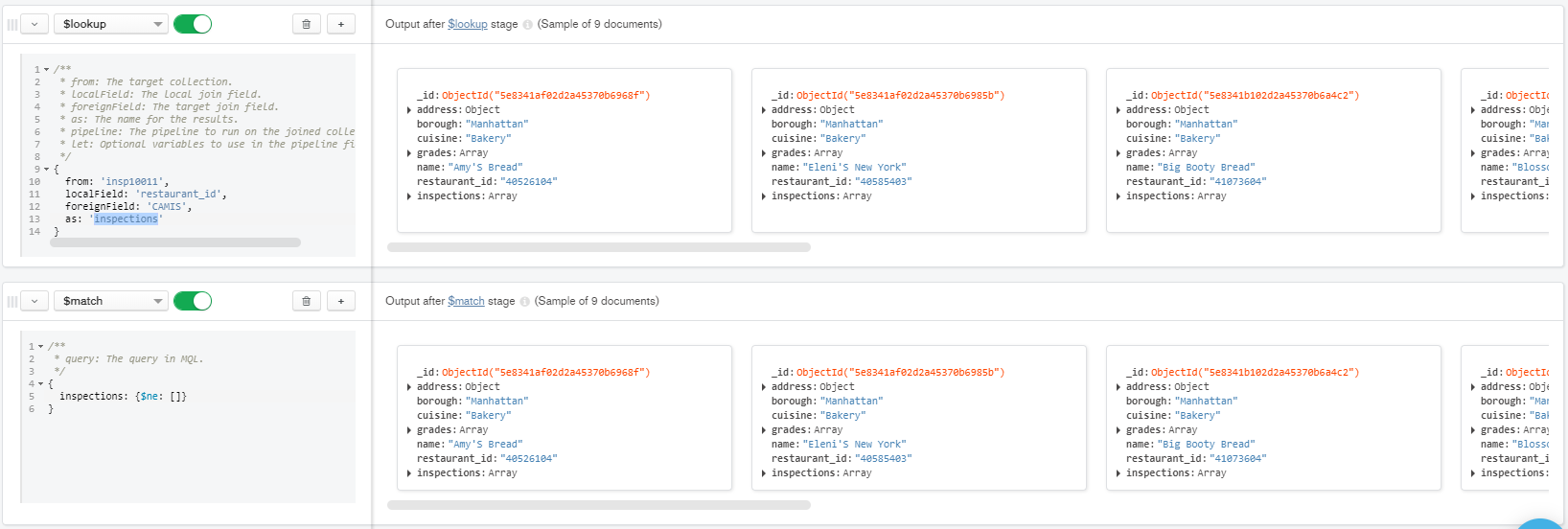
1. Create a view on the *inspections* collection for bakeries in zipcode 10011. Call it *insp10011*. It should have 100 documents.



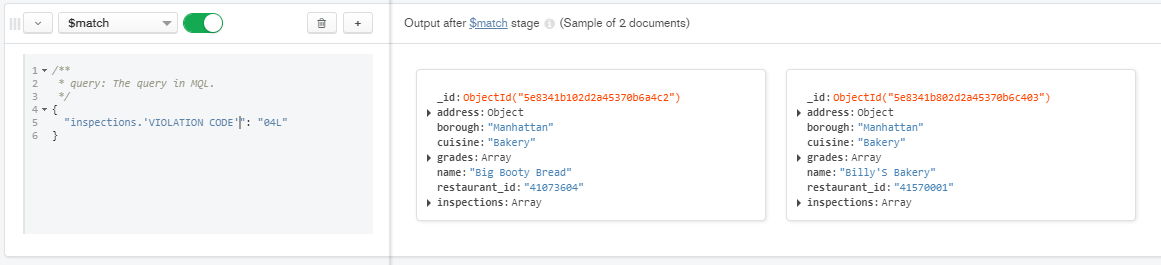
1. Go to the *bakery10011* view and join it with *insp10011* using $lookup (*restaurant\_id* = *CAMIS*). Your result should have 9 documents. The inspections will be nested in an array at the end of each document in the result.



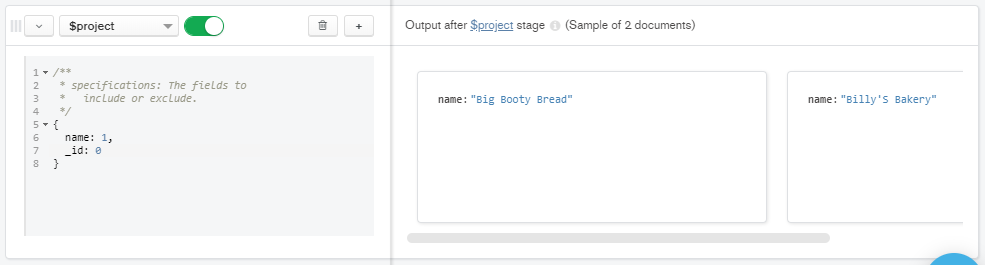
1. Add a stage to the pipeline to remove empty inspection arrays. Your result should have 6 documents.



1. Add a stage to filter the collection to keep only documents that have a mouse violation (use the code “04L”). This should give only 2 documents.



1. Add a stage to display just the restaurant names. (2 documents).



1. Google the restaurant names (in Manhattan, NY) and take a screenshot of their logos.



**Data Demand 2**

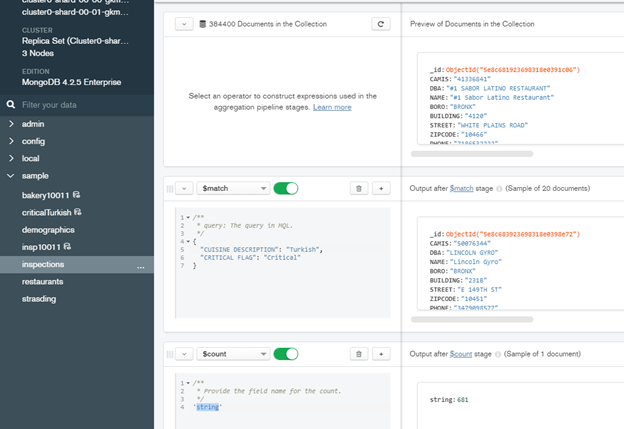
This is the high-level, English description of the final result:

Give the count per borough of critical inspections for Turkish restaurants in areas where the youth and community department demographic has at least 100 participants and at least 50 females counted.

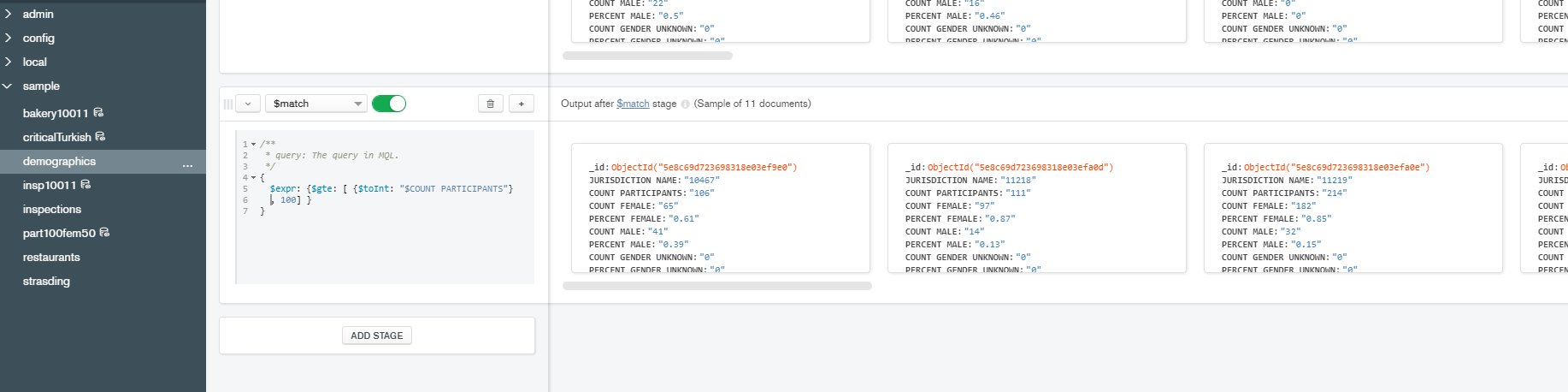
Do the steps in sequence and do screen captures of your results.

Give screenshots for each activity below (show the MongoDB Compass stage and results.)

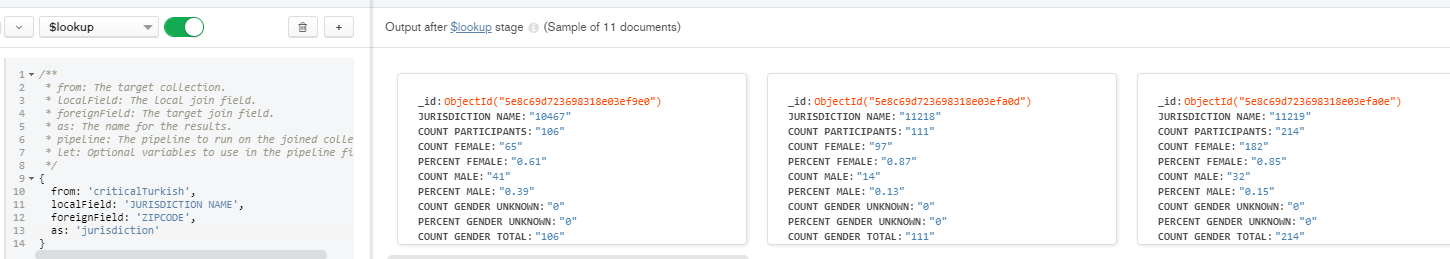
1. Create a view on *inspections* called *criticalTurkish*. The cuisine description should be “Turkish” and the critical flag should be “Critical”. (681 documents)



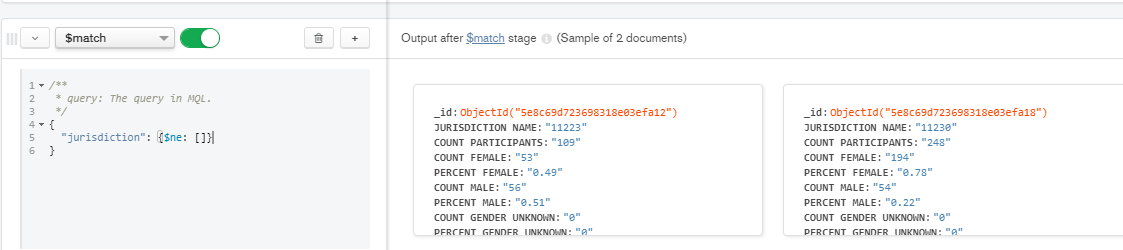
1. Create a view on *demographics* called *part100fem50.* The count of participants should be ≥ 100 and the count of females should be ≥ 50. (11 documents)



1. Join *part100fem50* with *criticalTurkish* on zipcode. (11 documents)



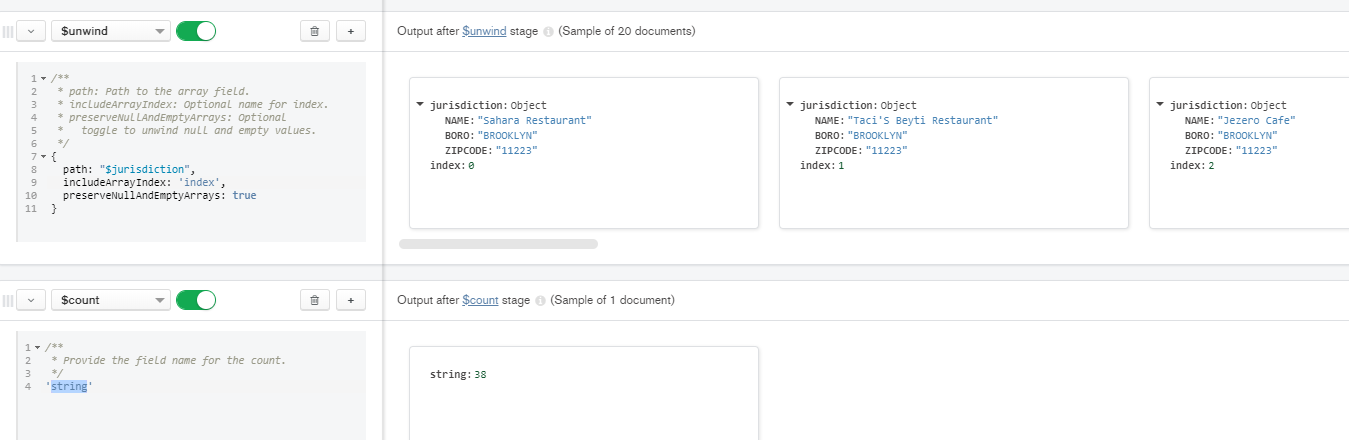
1. Remove any empty arrays. (2 documents)



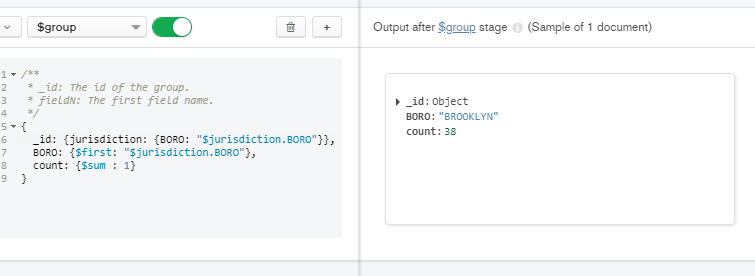
1. Project zipcode, borough, and restaurant name. (2 documents: borough and restaurant will be in an array in each.)



1. Unwind the array to create a different document for each borough and restaurant pair. (If you insert a $count stage here, you will see 38 documents are in the result; then delete the $count stage.)



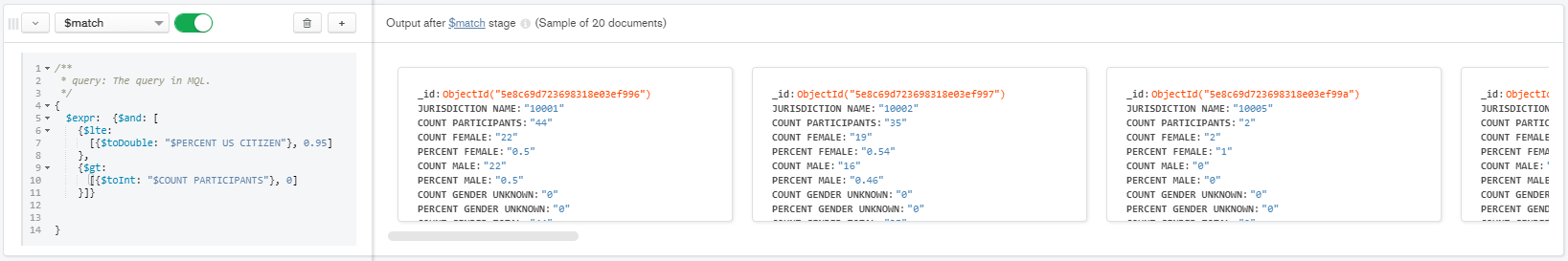
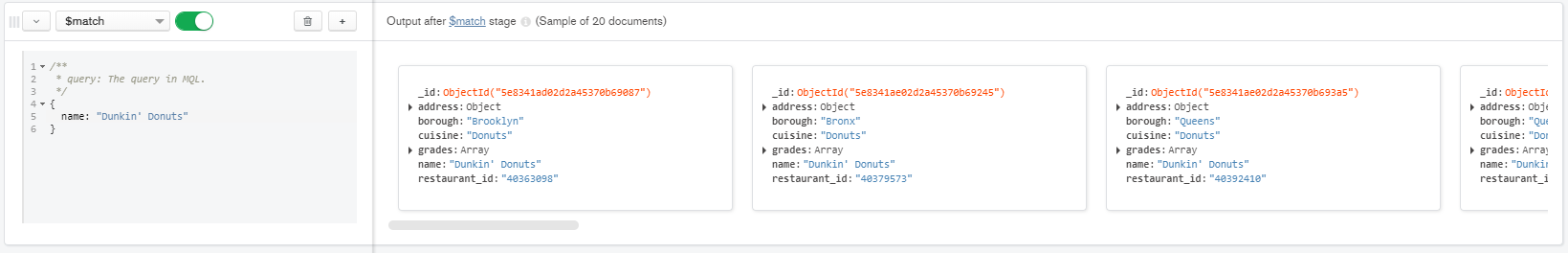
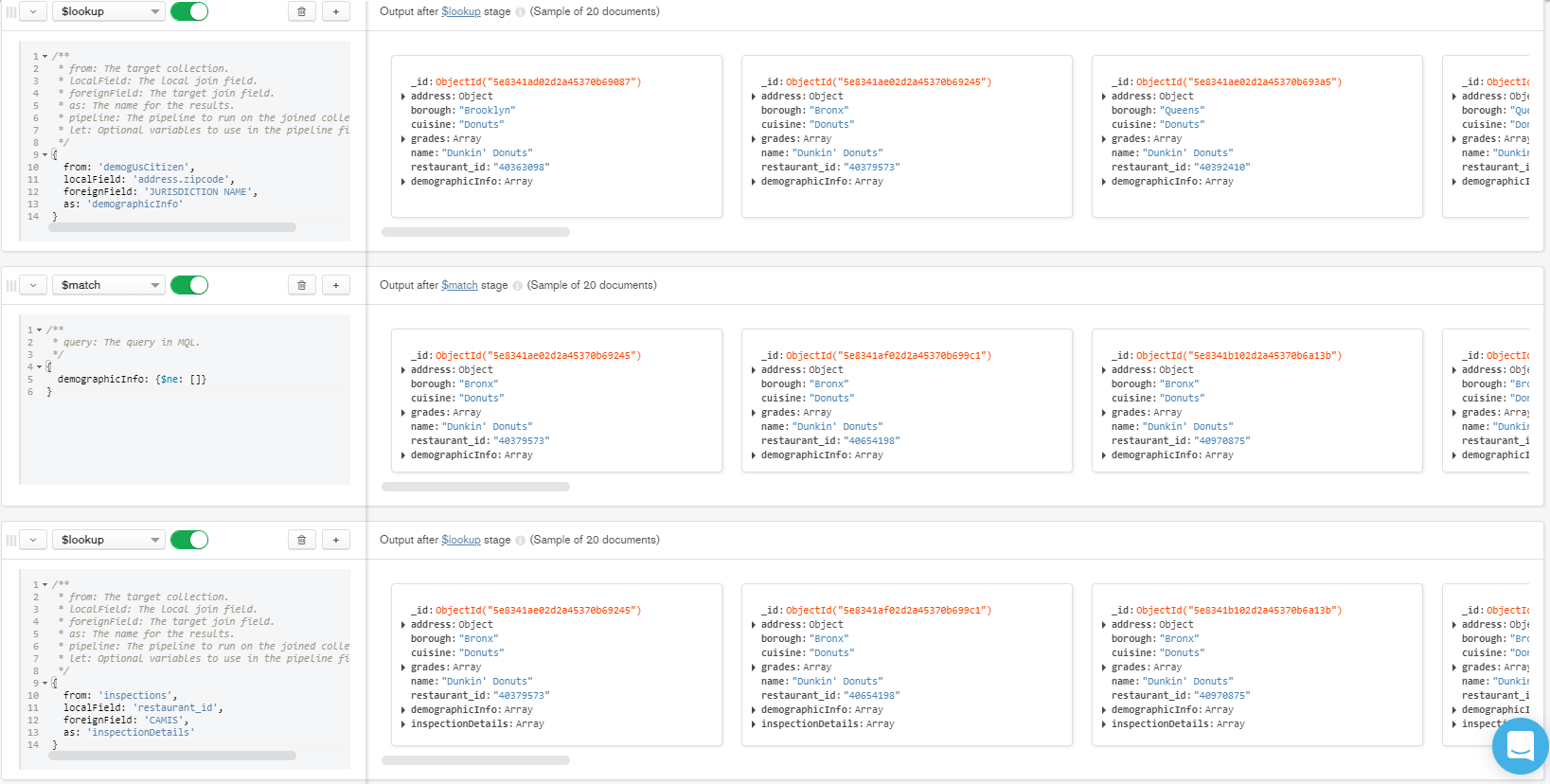
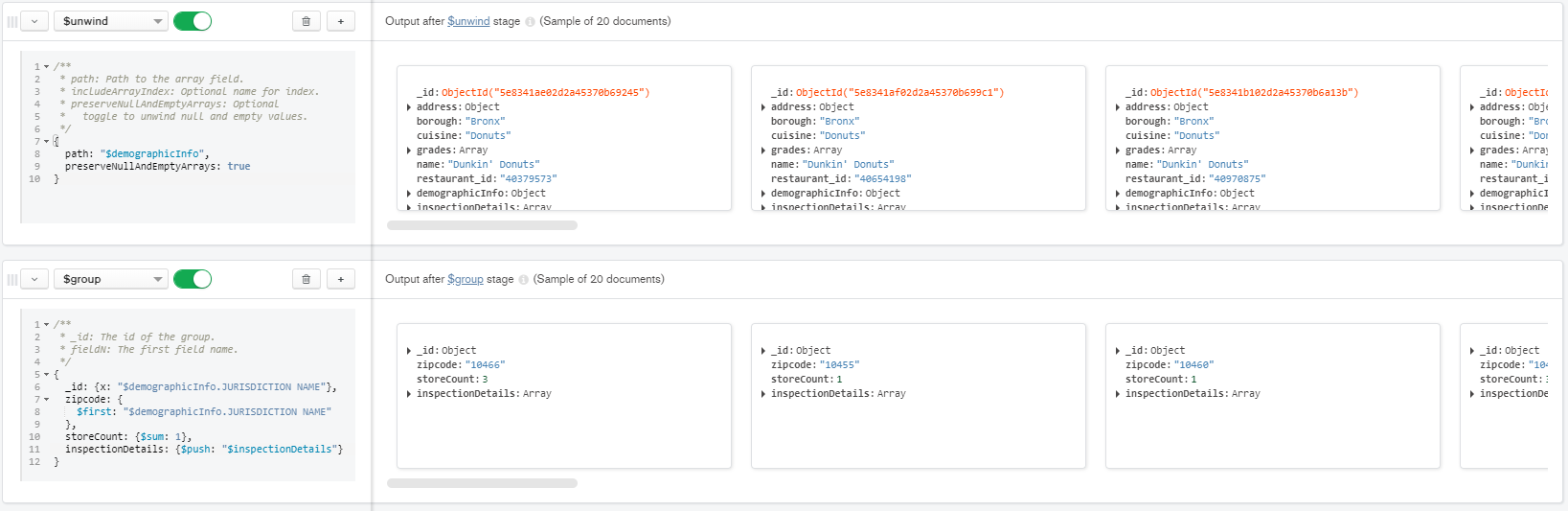
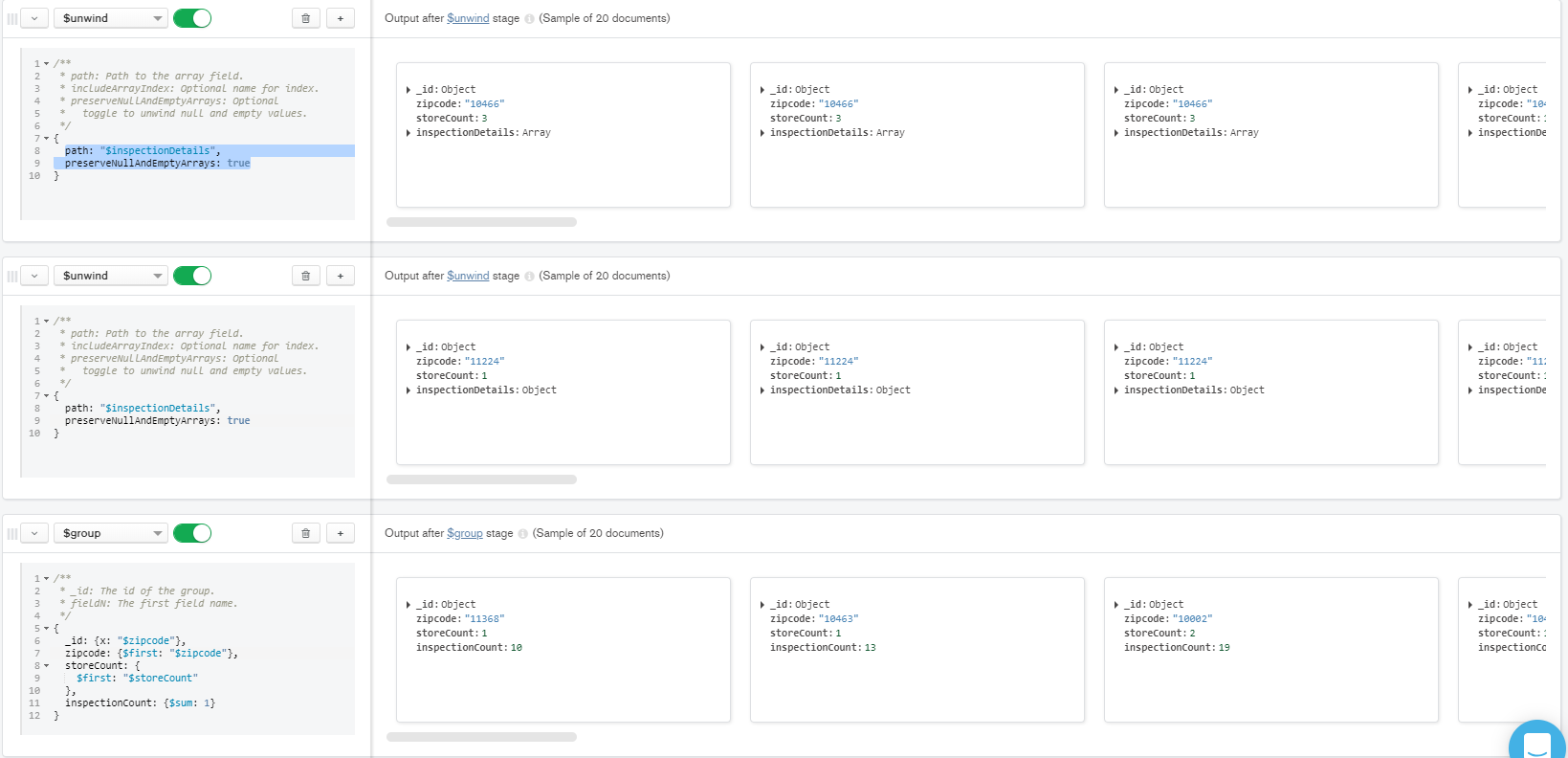
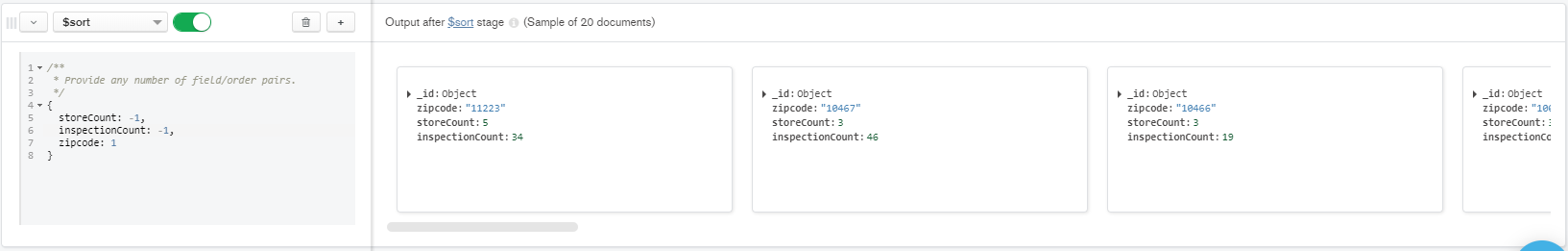
1. Group the documents by borough and count the number of documents per borough. (1 document)



**Data Demand 3**

Write your own data demand and create the solution for it. Include a step-by-step description of the aggregation pipeline stages and results. The result must be non-empty at each stage, and must join on the three different collections.

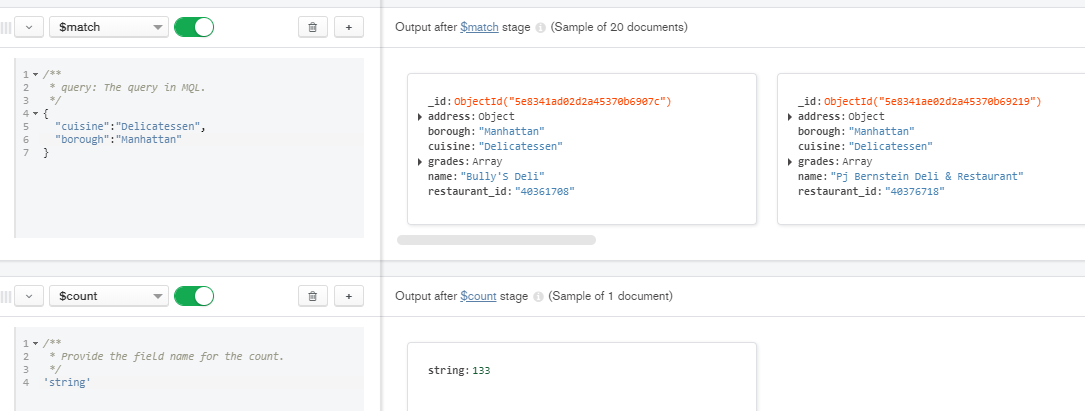
Give the zipcode and the number of Dunkin’ Donuts restaurants and the total number of inspections among the different stores within a zipcode where the local demographic is less than or equal to 95% US Citizens and the number of participants for that zipcode is more than 0.

1. Create a view of Demographics where the PERCENT US CITIZEN is ≤ 0.95 and COUNT PARTICIPANTS > 0 (You should have 36 documents)  
   
2. Create a view of Restaurants where the restaurant name is “Dunkin’ Donuts”. There are some Dunkin’ Donuts that are connected to Baskin’ Robbins but for this we are only interested in the individual stores. (206 Documents)  
   
3. Using lookup, join your created view of Dunkin’ Donuts to the View of Demographics and the inspections table the zipcodes, removing all unmatched elements. (49 Documents)  
   
4. Group the stores by zipcode and count the number of Dunkin’ Donuts that are present in that area. (29 Documents)  
   
5. Then unwind and group the results by each individual store inspection, counting how many inspections are present within the zipcode  
   
6. Sort the results in descending order based on store count and inspection count, then sort by zipcode in ascending order.  
   

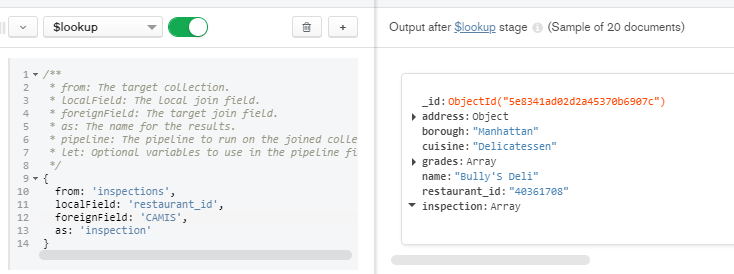
**Data Demand 4 (only for teams of 2; solo projects can omit this problem)**

Determine the count of all “C” reviewed Delicatessen cuisine restaurants in Manhattan, with their number of C reviews alongside their phone number.

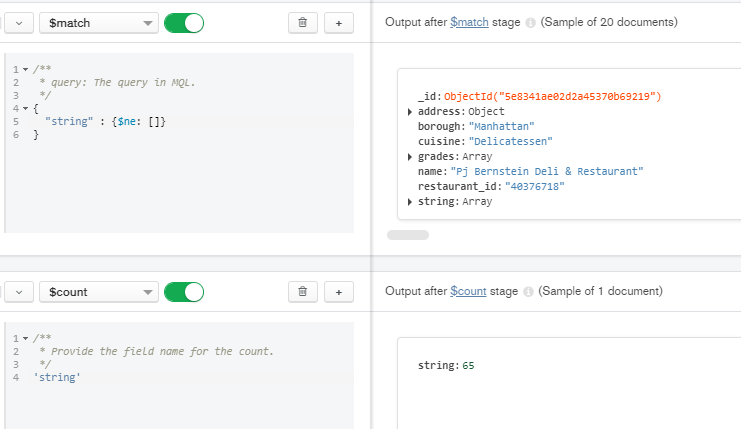
1. Determine all places serving Delicatessen cuisine in Manhattan. (133 documents) .



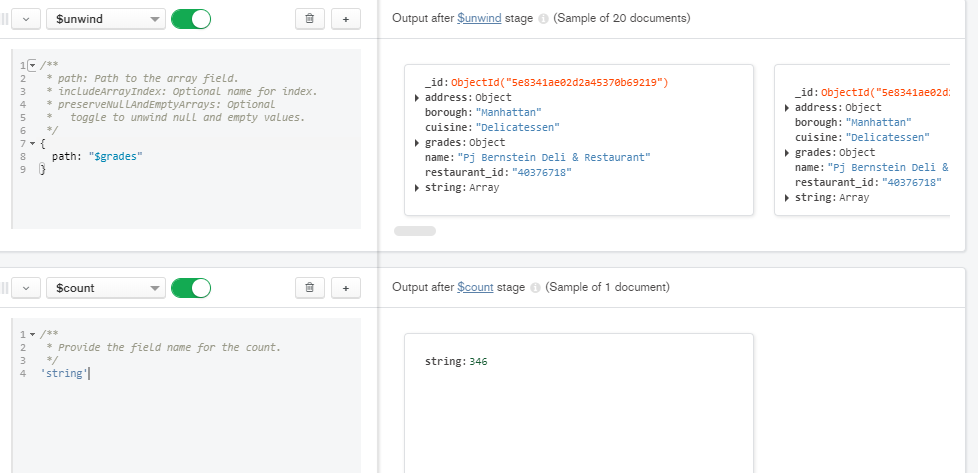
1. Using the $lookup function, join the view table created in step 1 with the inspections table on restaurant\_id and CAMIS. Use $out to create a copy of this. (133 rows)



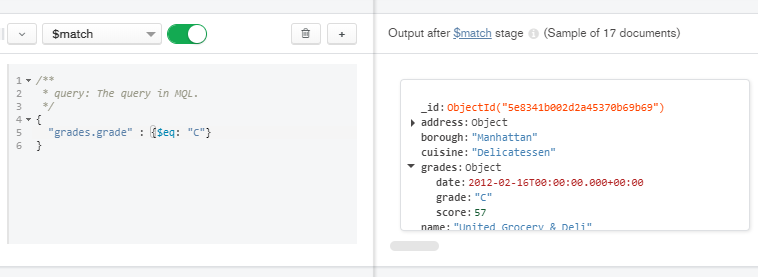
1. Remove all restaurants without any inspection information. (65 documents)



1. Seperate each review by $unwinding the grades Array(346 docuements)



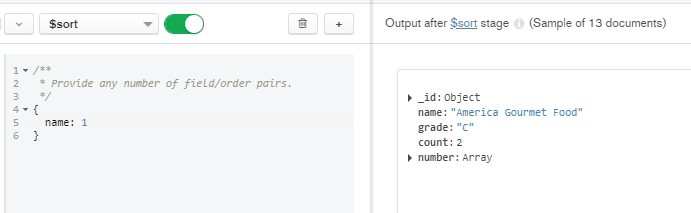
1. Filter by only restaurants that received a C rating (17 documents)



1. Group by name and grade for each restaurant with a count for each restaurant. Additionally, grab the number in an array using the “number: {$first: $string.PHONE}”. This will yield an entire array of numbers, but leave it this way for simplicity's sake (13 documents).



1. Sort Names in Ascending Order (13 documents)



**Submission**

Only one submission per team in Canvas is needed for the 3 (or 4) data demands. I will check your online timelogs in Google Drive, so there is no need to submit them in Canvas.