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:

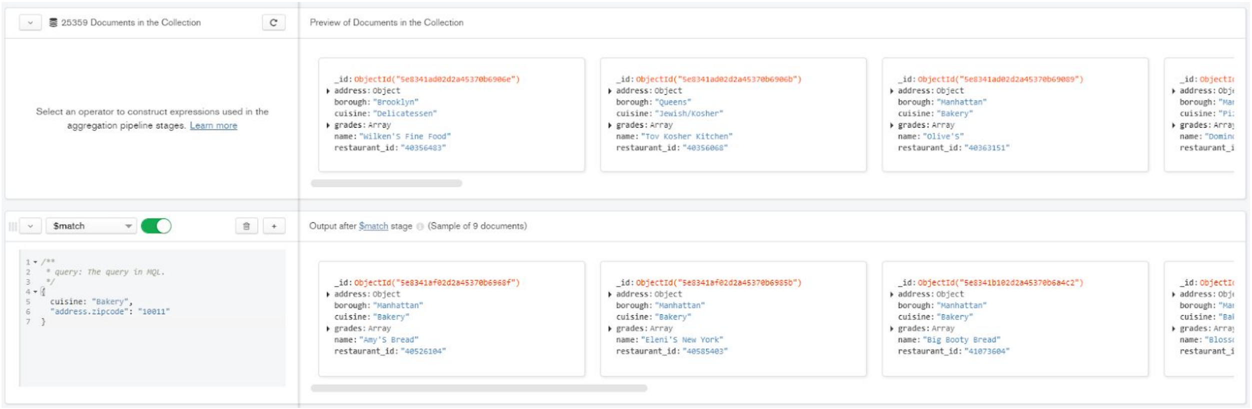
Give the names of bakeries located in zipcode 10011 that have at least one mouse violation in their inspections.

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.



25359 Documents in the Collection

Preview of Documents in the Collection

Select an operator to construct expressions used in the aggregation pipeline stages. Learn more

```
_id: ObjectId("5e8341ad02d2a45370b6906e")
▸ address: Object
  borough: "Brooklyn"
  cuisine: "Delicatessen"
▸ grades: Array
  name: "Wilken'S Fine Food"
  restaurant_id: "40356483"
```

```
_id: ObjectId("5e8341ad02d2a45370b6906b")
▸ address: Object
  borough: "Queens"
  cuisine: "Jewish/Kosher"
▸ grades: Array
  name: "Tov Kosher Kitchen"
  restaurant_id: "40356068"
```

```
_id: ObjectId("5e8341ad02d2a45370b69089")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Olive'S"
  restaurant_id: "40363151"
```

```
_id: ObjectI
▸ address: Obj
  borough: "Ma
  cuisine: "Pi
▸ grades: Array
  name: "Domin
  restaurant_i
```

$match

Output after $match stage (Sample of 9 documents)

```
/**
 * query: The query in MQL.
 */
{
  cuisine: "Bakery",
  "address.zipcode": "10011"
}
```

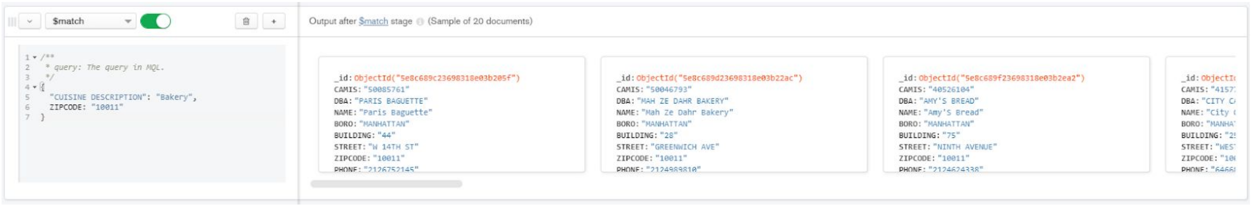
```
_id: ObjectId("5e8341af02d2a45370b6968f")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Amy'S Bread"
  restaurant_id: "40526104"
```

```
_id: ObjectId("5e8341af02d2a45370b6985b")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Eleni'S New York"
  restaurant_id: "40585403"
```

```
_id: ObjectId("5e8341b102d2a45370b6a4c2")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Big Booty Bread"
  restaurant_id: "41073604"
```

```
_id: ObjectI
▸ address: Obj
  borough: "Ma
  cuisine: "Ba
▸ grades: Array
  name: "Bloss
  restaurant_i
```

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



$match

Output after $match stage (Sample of 20 documents)

```
/**
 * query: The query in MQL.
 */
{
  "CUISINE DESCRIPTION": "Bakery",
  ZIPCODE: "10011"
}
```

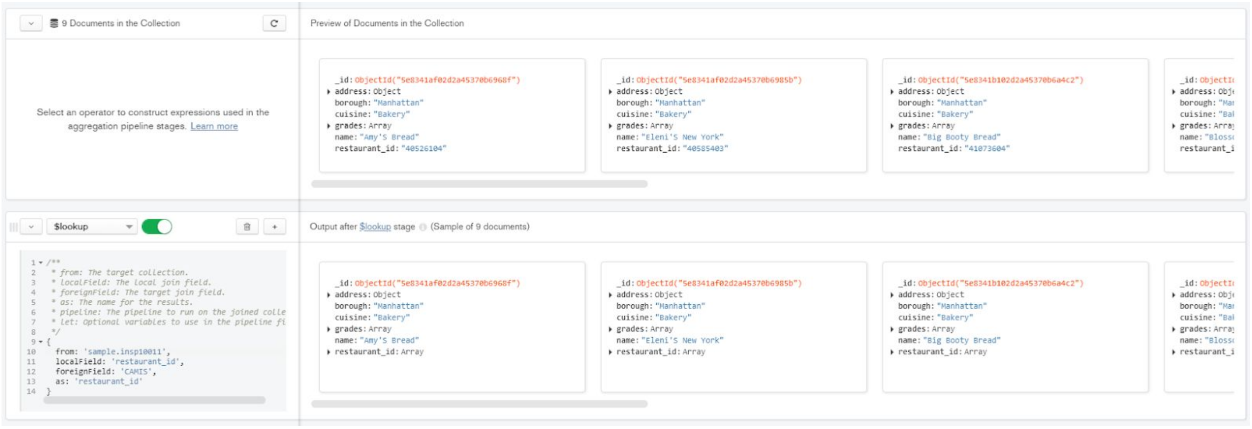
```
_id: ObjectId("5e8c689c23698318e03b205f")
CAMIS: "50085761"
DBA: "PARIS BAGUETTE"
NAME: "Paris Baguette"
BORO: "MANHATTAN"
BUILDING: "44"
STREET: "W 14TH ST"
ZIPCODE: "10011"
PHONE: "2126752145"
```

```
_id: ObjectId("5e8c689d23698318e03b22ac")
CAMIS: "50046793"
DBA: "MAH ZE DAHR BAKERY"
NAME: "Mah Ze Dahr Bakery"
BORO: "MANHATTAN"
BUILDING: "28"
STREET: "GREENWICH AVE"
ZIPCODE: "10011"
PHONE: "2124989810"
```

```
_id: ObjectId("5e8c689f23698318e03b2ea2")
CAMIS: "40526104"
DBA: "AMY'S BREAD"
NAME: "Amy'S Bread"
BORO: "MANHATTAN"
BUILDING: "75"
STREET: "NINTH AVENUE"
ZIPCODE: "10011"
PHONE: "2124624338"
```

```
_id: ObjectI
CAMIS: "4157
DBA: "CITY C
NAME: "City
BORO: "MANHA
BUILDING: "2
STREET: "WES
ZIPCODE: "10
PHONE: "6466
```

3. 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.



9 Documents in the Collection

Preview of Documents in the Collection

Select an operator to construct expressions used in the aggregation pipeline stages. Learn more

```
_id: ObjectId("5e8341af02d2a45370b6968f")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Amy'S Bread"
  restaurant_id: "40526104"
```

```
_id: ObjectId("5e8341af02d2a45370b6985b")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Eleni'S New York"
  restaurant_id: "40585403"
```

```
_id: ObjectId("5e8341b102d2a45370b6a4c2")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Big Booty Bread"
  restaurant_id: "41073604"
```

```
_id: ObjectI
▸ address: Obj
  borough: "Ma
  cuisine: "Ba
▸ grades: Array
  name: "Bloss
  restaurant_i
```

$lookup

Output after $lookup stage (Sample of 9 documents)

```
/**
 * from: The target collection.
 * localField: The local join field.
 * foreignField: The target join field.
 * as: The name for the results.
 * pipeline: The pipeline to run on the joined colle
 * let: Optional variables to use in the pipeline fi
 */
{
  from: 'sample.insp10011',
  localField: 'restaurant_id',
  foreignField: 'CAMIS',
  as: 'restaurant_id'
}
```

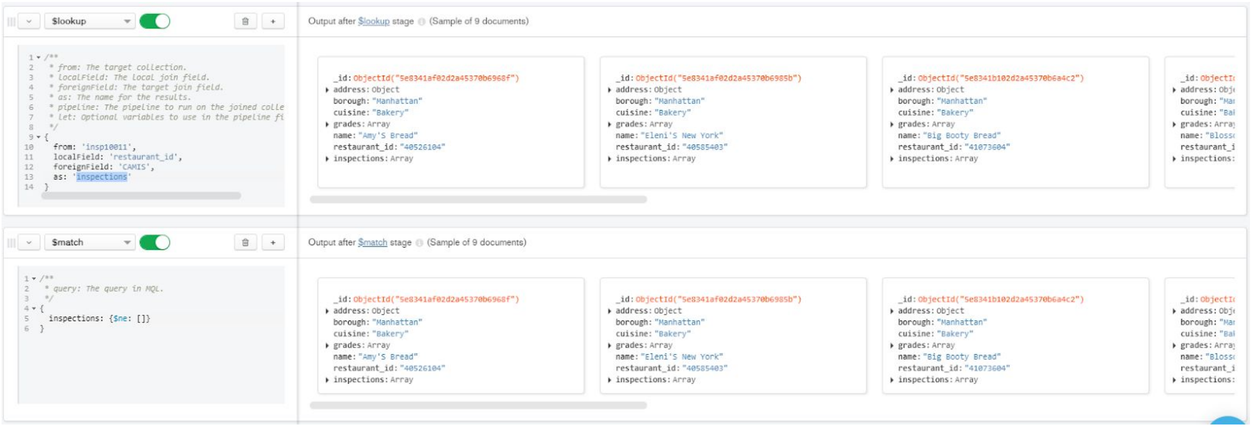
```
_id: ObjectId("5e8341af02d2a45370b6968f")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Amy'S Bread"
▸ restaurant_id: Array
```

```
_id: ObjectId("5e8341af02d2a45370b6985b")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Eleni'S New York"
▸ restaurant_id: Array
```

```
_id: ObjectId("5e8341b102d2a45370b6a4c2")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Big Booty Bread"
▸ restaurant_id: Array
```

```
_id: ObjectI
▸ address: Obj
  borough: "Ma
  cuisine: "Ba
▸ grades: Array
  name: "Bloss
▸ restaurant_i
```

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



$lookup

Output after $lookup stage (Sample of 9 documents)

```
/**
 * from: The target collection.
 * localField: The local join field.
 * foreignField: The target join field.
 * as: The name for the results.
 * pipeline: The pipeline to run on the joined colle
 * let: Optional variables to use in the pipeline fi
 */
{
  from: 'insp10011',
  localField: 'restaurant_id',
  foreignField: 'CAMIS',
  as: 'inspections'
}
```

```
_id: ObjectId("5e8341af02d2a45370b6968f")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Amy'S Bread"
  restaurant_id: "40526104"
▸ inspections: Array
```

```
_id: ObjectId("5e8341af02d2a45370b6985b")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Eleni'S New York"
  restaurant_id: "40585403"
▸ inspections: Array
```

```
_id: ObjectId("5e8341b102d2a45370b6a4c2")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Big Booty Bread"
  restaurant_id: "41073604"
▸ inspections: Array
```

```
_id: ObjectI
▸ address: Obj
  borough: "Ma
  cuisine: "Ba
▸ grades: Array
  name: "Bloss
  restaurant_i
▸ inspections:
```

$match

Output after $match stage (Sample of 9 documents)

```
/**
 * query: The query in MQL.
 */
{
  inspections: {$ne: []}
}
```

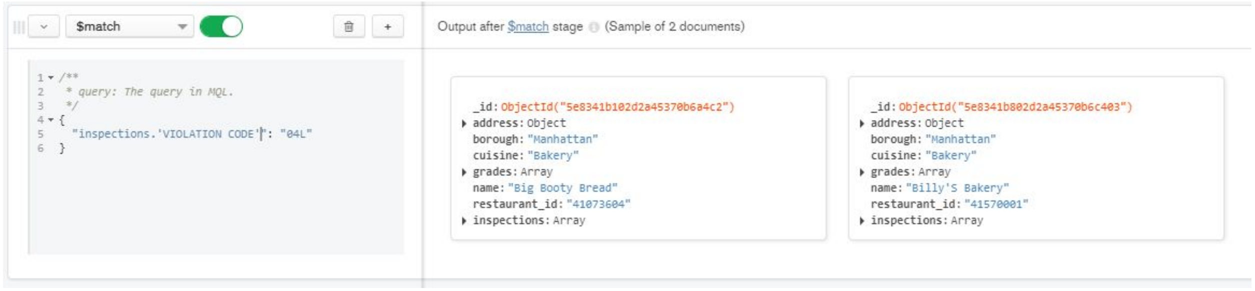
```
_id: ObjectId("5e8341af02d2a45370b6968f")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Amy'S Bread"
  restaurant_id: "40526104"
▸ inspections: Array
```

```
_id: ObjectId("5e8341af02d2a45370b6985b")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Eleni'S New York"
  restaurant_id: "40585403"
▸ inspections: Array
```

```
_id: ObjectId("5e8341b102d2a45370b6a4c2")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Bakery"
▸ grades: Array
  name: "Big Booty Bread"
  restaurant_id: "41073604"
▸ inspections: Array
```

```
_id: ObjectI
▸ address: Obj
  borough: "Ma
  cuisine: "Ba
▸ grades: Array
  name: "Bloss
  restaurant_i
▸ inspections:
```

5. 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.

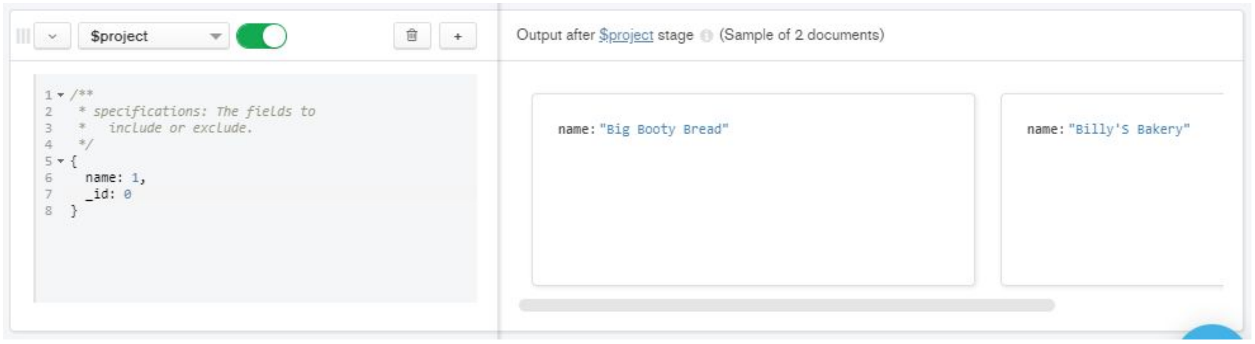


```
/**
 * query: The query in MQL.
 */
{
  "inspections.'VIOLATION CODE'": "04L"
}
```

```
_id: ObjectId("5e8341b102d2a45370b6a4c2")
▸ address: Object
borough: "Manhattan"
cuisine: "Bakery"
▸ grades: Array
name: "Big Booty Bread"
restaurant_id: "41073604"
▸ inspections: Array
```

```
_id: ObjectId("5e8341b802d2a45370b6c403")
▸ address: Object
borough: "Manhattan"
cuisine: "Bakery"
▸ grades: Array
name: "Billy'S Bakery"
restaurant_id: "41570001"
▸ inspections: Array
```

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



```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  name: 1,
  _id: 0
}
```

```
name: "Big Booty Bread"
```

```
name: "Billy'S Bakery"
```

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

BIG BOOTY
BIG BOOTY BREAD CO.
FRESH BAKED BREAD · COFFEE · SANDWICHES
Once You go BIG Booty You Never go Back!

Billy's
BAKERY

Data Demand 2

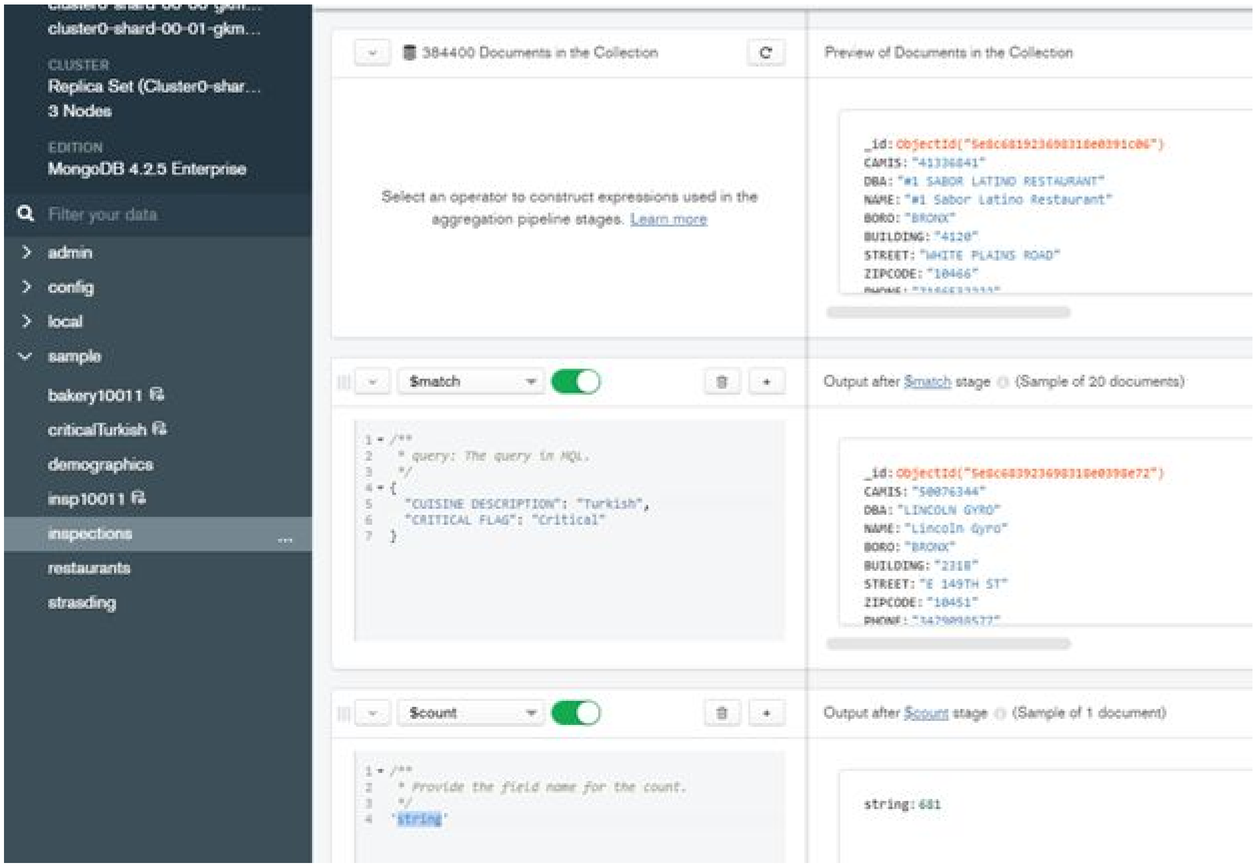
This is the high-level, English description of the final result:

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)


cluster0-shard-00-01-gkm...
CLUSTER
Replica Set (Cluster0-shar...
3 Nodes
EDITION
MongoDB 4.2.5 Enterprise
Filter your data
admin
config
local
sample
bakery10011
criticalTurkish
demographics
insp10011
inspections
restaurants
strasding
384400 Documents in the Collection
Preview of Documents in the Collection
Select an operator to construct expressions used in the aggregation pipeline stages. Learn more
$match
Output after $match stage (Sample of 20 documents)
$count
Output after $count stage (Sample of 1 document)

```
_id: ObjectId("5e8c6819236983l8e0391c06")
CAMIS: "41336841"
DBA: "#1 SABOR LATINO RESTAURANT"
NAME: "#1 Sabor Latino Restaurant"
BORO: "BRONX"
BUILDING: "4120"
STREET: "WHITE PLAINS ROAD"
ZIPCODE: "10466"
```

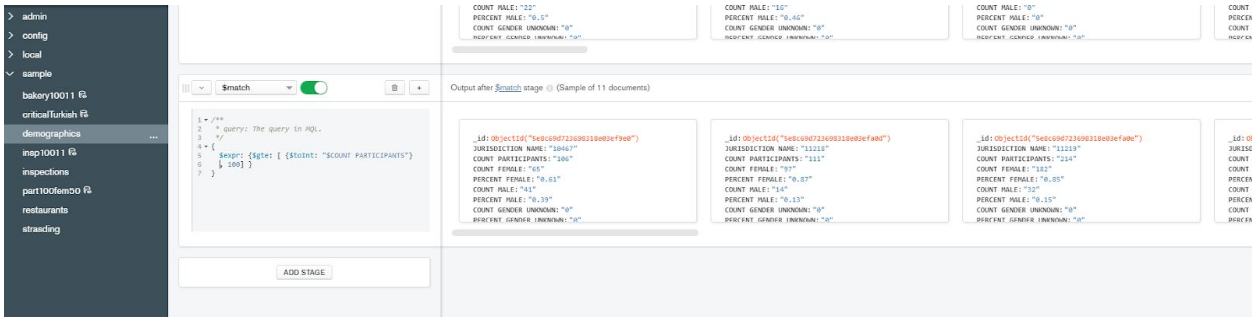
```
/**
 * query: The query in MQL.
 */
{
  "CUISINE DESCRIPTION": "Turkish",
  "CRITICAL FLAG": "Critical"
}
```

```
_id: ObjectId("5e8c6839236983l8e0398e72")
CAMIS: "50076344"
DBA: "LINCOLN GYRO"
NAME: "Lincoln Gyro"
BORO: "BRONX"
BUILDING: "2318"
STREET: "E 149TH ST"
ZIPCODE: "10451"
```

```
/**
 * Provide the field name for the count.
 */
'string'
```

```
string: 681
```

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



```
COUNT MALE: "22"
PERCENT MALE: "0.5"
COUNT GENDER UNKNOWN: "0"
```

```
COUNT MALE: "16"
PERCENT MALE: "0.46"
COUNT GENDER UNKNOWN: "0"
```

```
COUNT MALE: "0"
PERCENT MALE: "0"
COUNT GENDER UNKNOWN: "0"
```

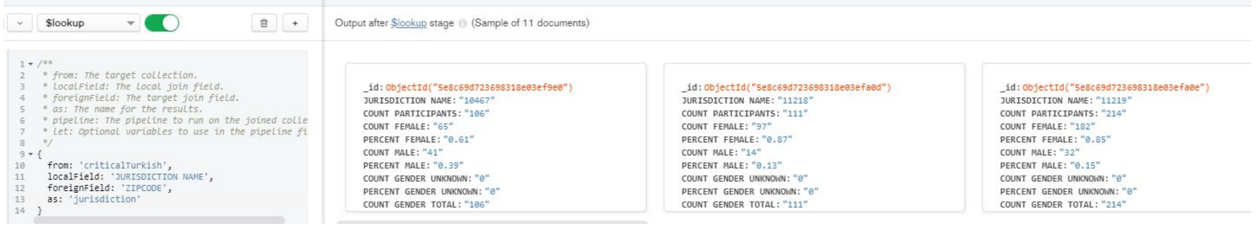
```
/**
 * query: The query in MQL.
 */
{
  $expr: {$gte: [ {$toInt: "$COUNT PARTICIPANTS"}
  , 100] }
}
```

```
_id: ObjectId("5e8c69d723698318e03ef9e0")
JURISDICTION NAME: "10467"
COUNT PARTICIPANTS: "106"
COUNT FEMALE: "65"
PERCENT FEMALE: "0.61"
COUNT MALE: "41"
PERCENT MALE: "0.39"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
```

```
_id: ObjectId("5e8c69d723698318e03efa0d")
JURISDICTION NAME: "11218"
COUNT PARTICIPANTS: "111"
COUNT FEMALE: "97"
PERCENT FEMALE: "0.87"
COUNT MALE: "14"
PERCENT MALE: "0.13"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
```

```
_id: ObjectId("5e8c69d723698318e03efa0e")
JURISDICTION NAME: "11219"
COUNT PARTICIPANTS: "214"
COUNT FEMALE: "182"
PERCENT FEMALE: "0.85"
COUNT MALE: "32"
PERCENT MALE: "0.15"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
```

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



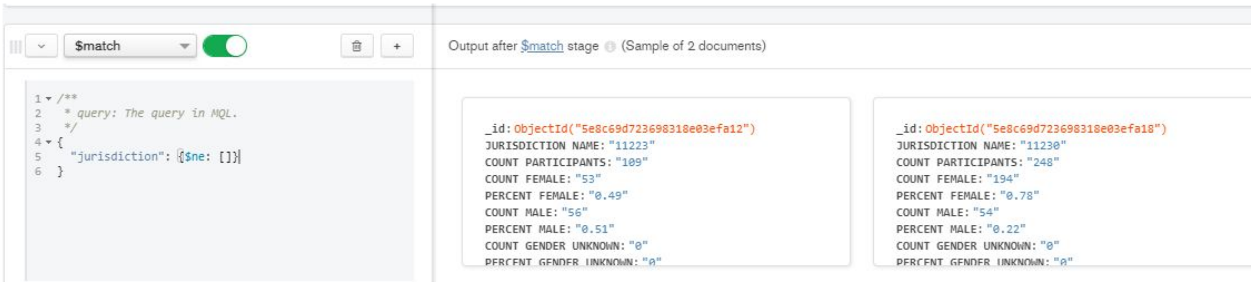
```
/**
 * from: The target collection.
 * localField: The local join field.
 * foreignField: The target join field.
 * as: The name for the results.
 * pipeline: The pipeline to run on the joined colle
 * let: Optional variables to use in the pipeline fi
 */
{
  from: 'criticalTurkish',
  localField: 'JURISDICTION NAME',
  foreignField: 'ZIPCODE',
  as: 'jurisdiction'
}
```

```
_id: ObjectId("5e8c69d723698318e03ef9e0")
JURISDICTION NAME: "10467"
COUNT PARTICIPANTS: "106"
COUNT FEMALE: "65"
PERCENT FEMALE: "0.61"
COUNT MALE: "41"
PERCENT MALE: "0.39"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
COUNT GENDER TOTAL: "106"
```

```
_id: ObjectId("5e8c69d723698318e03efa0d")
JURISDICTION NAME: "11218"
COUNT PARTICIPANTS: "111"
COUNT FEMALE: "97"
PERCENT FEMALE: "0.87"
COUNT MALE: "14"
PERCENT MALE: "0.13"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
COUNT GENDER TOTAL: "111"
```

```
_id: ObjectId("5e8c69d723698318e03efa0e")
JURISDICTION NAME: "11219"
COUNT PARTICIPANTS: "214"
COUNT FEMALE: "182"
PERCENT FEMALE: "0.85"
COUNT MALE: "32"
PERCENT MALE: "0.15"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
COUNT GENDER TOTAL: "214"
```

4. Remove any empty arrays. (2 documents)



$match

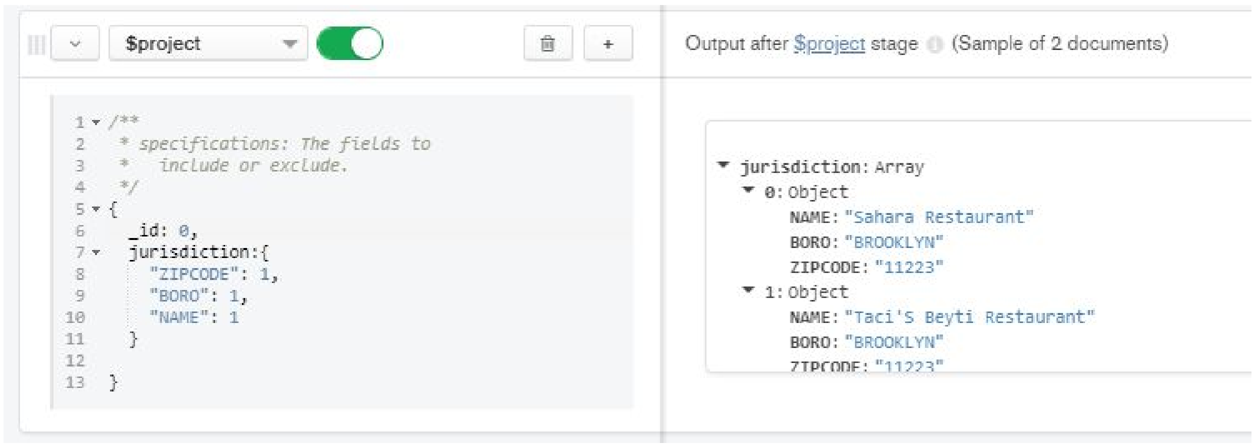
Output after $match stage (Sample of 2 documents)

```
/**
 * query: The query in MQL.
 */
{
  "jurisdiction": {$ne: []}
}
```

```
_id: ObjectId("5e8c69d723698318e03efa12")
JURISDICTION NAME: "11223"
COUNT PARTICIPANTS: "109"
COUNT FEMALE: "53"
PERCENT FEMALE: "0.49"
COUNT MALE: "56"
PERCENT MALE: "0.51"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
```

```
_id: ObjectId("5e8c69d723698318e03efa18")
JURISDICTION NAME: "11230"
COUNT PARTICIPANTS: "248"
COUNT FEMALE: "194"
PERCENT FEMALE: "0.78"
COUNT MALE: "54"
PERCENT MALE: "0.22"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
```

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



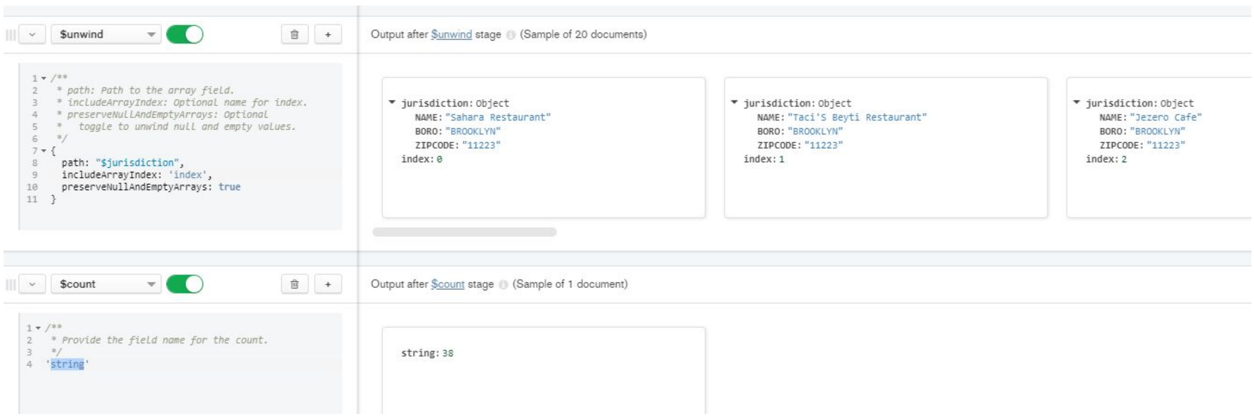
$project

Output after $project stage (Sample of 2 documents)

```
/**
 * specifications: The fields to
 *   include or exclude.
 */
{
  _id: 0,
  jurisdiction:{
    "ZIPCODE": 1,
    "BORO": 1,
    "NAME": 1
  }

}
```

```
▼ jurisdiction: Array
  ▼ 0: Object
      NAME: "Sahara Restaurant"
      BORO: "BROOKLYN"
      ZIPCODE: "11223"
  ▼ 1: Object
      NAME: "Taci'S Beyti Restaurant"
      BORO: "BROOKLYN"
      ZIPCODE: "11223"
```

6. 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.)



$unwind

Output after $unwind stage (Sample of 20 documents)

```
/**
 * path: Path to the array field.
 * includeArrayIndex: Optional name for index.
 * preserveNullAndEmptyArrays: Optional
 *   toggle to unwind null and empty values.
 */
{
  path: "$jurisdiction",
  includeArrayIndex: 'index',
  preserveNullAndEmptyArrays: true
}
```

```
▼ jurisdiction: Object
    NAME: "Sahara Restaurant"
    BORO: "BROOKLYN"
    ZIPCODE: "11223"
  index: 0
```

```
▼ jurisdiction: Object
    NAME: "Taci'S Beyti Restaurant"
    BORO: "BROOKLYN"
    ZIPCODE: "11223"
  index: 1
```

```
▼ jurisdiction: Object
    NAME: "Jezero Cafe"
    BORO: "BROOKLYN"
    ZIPCODE: "11223"
  index: 2
```

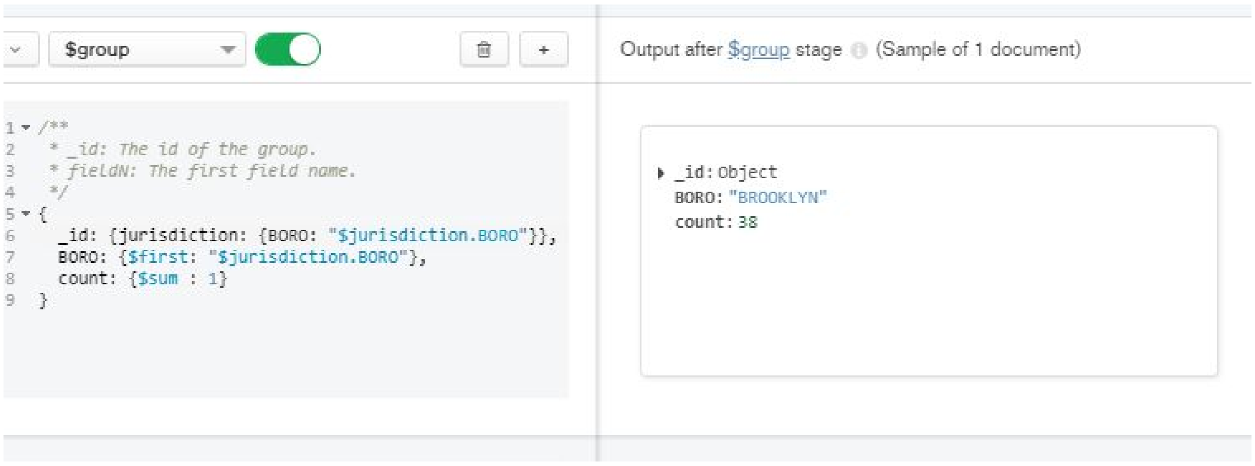
$count

Output after $count stage (Sample of 1 document)

```
/**
 * Provide the field name for the count.
 */
'string'
```

```
string: 38
```

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



$group

Output after $group stage (Sample of 1 document)

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: {jurisdiction: {BORO: "$jurisdiction.BORO"}},
  BORO: {$first: "$jurisdiction.BORO"},
  count: {$sum : 1}
}
```

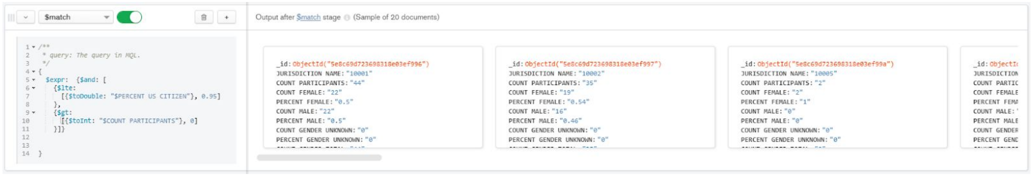
```
▸ _id: Object
  BORO: "BROOKLYN"
  count: 38
```

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)



$match

Output after $match stage (Sample of 20 documents)

```
/**
 * query: The query in MQL.
 */
{
  $expr:  {$and: [
    {$lte:
      [{$toDouble: "$PERCENT US CITIZEN"}, 0.95]
    },
    {$gt:
      [{$toInt: "$COUNT PARTICIPANTS"}, 0]
    }]}

}
```

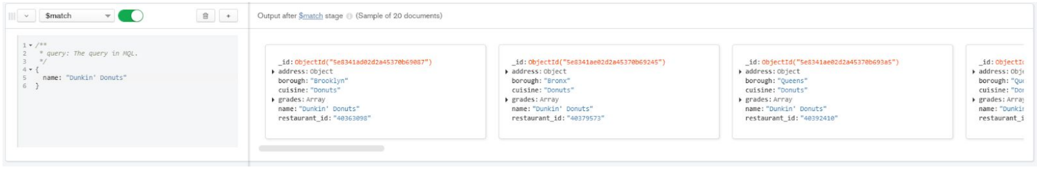
```
_id: ObjectId("5e8c69d723698318e03ef996")
JURISDICTION NAME: "10001"
COUNT PARTICIPANTS: "44"
COUNT FEMALE: "22"
PERCENT FEMALE: "0.5"
COUNT MALE: "22"
PERCENT MALE: "0.5"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
```

```
_id: ObjectId("5e8c69d723698318e03ef997")
JURISDICTION NAME: "10002"
COUNT PARTICIPANTS: "35"
COUNT FEMALE: "19"
PERCENT FEMALE: "0.54"
COUNT MALE: "16"
PERCENT MALE: "0.46"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
```

```
_id: ObjectId("5e8c69d723698318e03ef99a")
JURISDICTION NAME: "10005"
COUNT PARTICIPANTS: "2"
COUNT FEMALE: "2"
PERCENT FEMALE: "1"
COUNT MALE: "0"
PERCENT MALE: "0"
COUNT GENDER UNKNOWN: "0"
PERCENT GENDER UNKNOWN: "0"
```

```
_id: ObjectI
JURISDICTION
COUNT PARTIC
COUNT FEMALE
PERCENT FEMA
COUNT MALE:
PERCENT MALE
COUNT GENDER
PERCENT GENC
```

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)



$match

Output after $match stage (Sample of 20 documents)

```
/**
 * query: The query in MQL.
 */
{
  name: "Dunkin' Donuts"
}
```

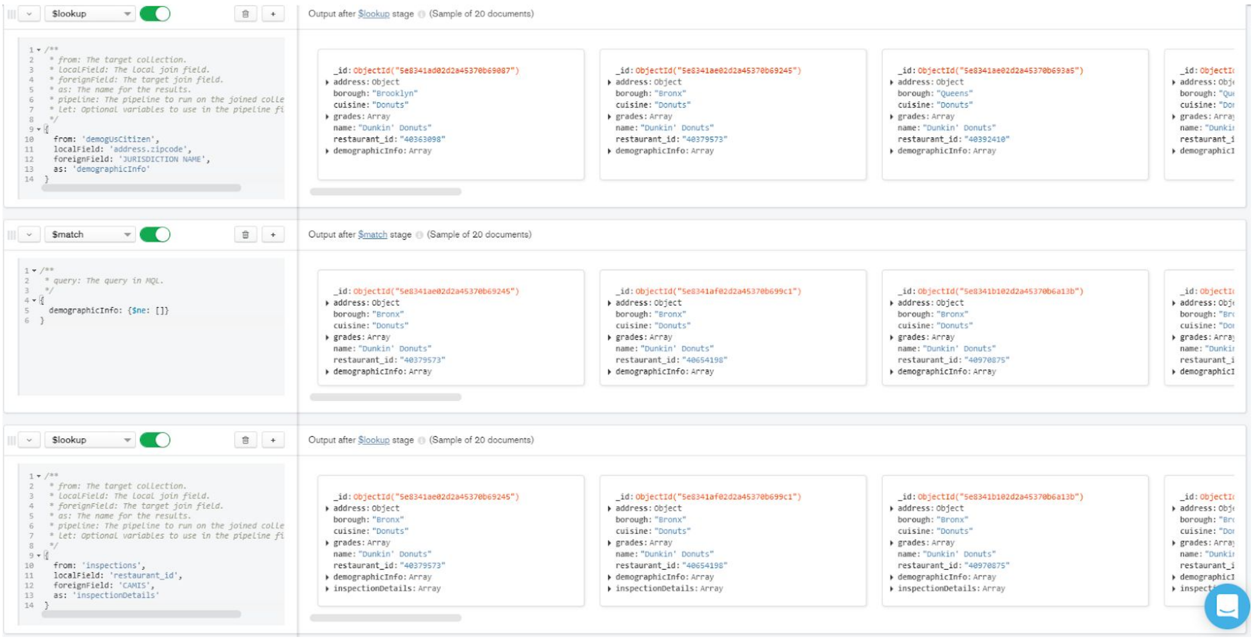
```
_id: ObjectId("5e8341ad02d2a45370b69087")
▸ address: Object
  borough: "Brooklyn"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40363098"
```

```
_id: ObjectId("5e8341ae02d2a45370b69245")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40379573"
```

```
_id: ObjectId("5e8341ae02d2a45370b693a5")
▸ address: Object
  borough: "Queens"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40392410"
```

```
_id: ObjectI
▸ address: Obj
  borough: "Qu
  cuisine: "Do
▸ grades: Arra
  name: "Dunki
  restaurant_i
```

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)



$lookup — Output after $lookup stage (Sample of 20 documents)

```
/**
 * from: The target collection.
 * localField: The local join field.
 * foreignField: The target join field.
 * as: The name for the results.
 * pipeline: The pipeline to run on the joined colle
 * let: Optional variables to use in the pipeline fi
 */
{
  from: 'demogUsCitizen',
  localField: 'address.zipcode',
  foreignField: 'JURISDICTION NAME',
  as: 'demographicInfo'
}
```

```
_id: ObjectId("5e8341ad02d2a45370b69087")
▸ address: Object
  borough: "Brooklyn"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40363098"
▸ demographicInfo: Array

_id: ObjectId("5e8341ae02d2a45370b69245")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40379573"
▸ demographicInfo: Array

_id: ObjectId("5e8341ae02d2a45370b693a5")
▸ address: Object
  borough: "Queens"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40392410"
▸ demographicInfo: Array
```

$match — Output after $match stage (Sample of 20 documents)

```
/**
 * query: The query in MQL.
 */
{
  demographicInfo: {$ne: []}
}
```

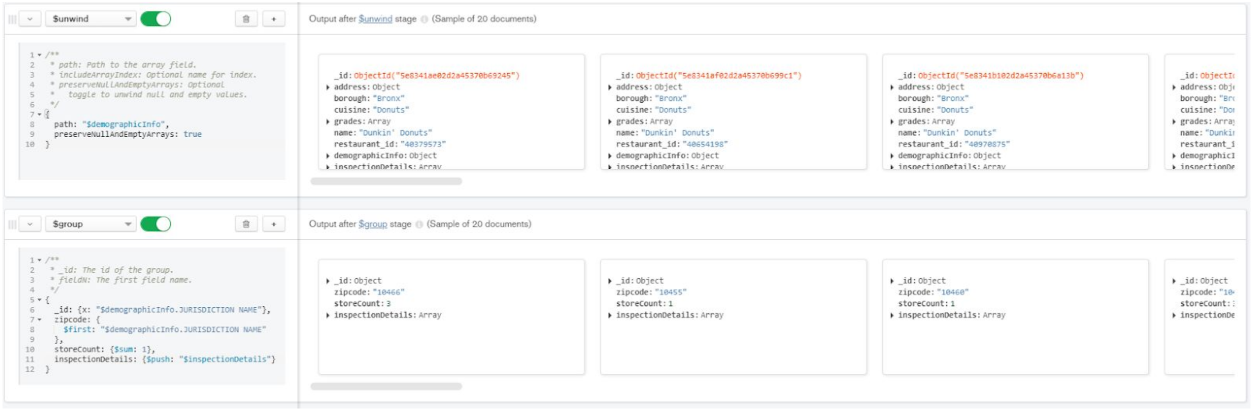
```
_id: ObjectId("5e8341ae02d2a45370b69245")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40379573"
▸ demographicInfo: Array

_id: ObjectId("5e8341af02d2a45370b699c1")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40654198"
▸ demographicInfo: Array

_id: ObjectId("5e8341b102d2a45370b6a13b")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40970875"
▸ demographicInfo: Array
```

$lookup — Output after $lookup stage (Sample of 20 documents)

```
/**
 * from: The target collection.
 * localField: The local join field.
 * foreignField: The target join field.
 * as: The name for the results.
 * pipeline: The pipeline to run on the joined colle
 * let: Optional variables to use in the pipeline fi
 */
{
  from: 'inspections',
  localField: 'restaurant_id',
  foreignField: 'CAMIS',
  as: 'inspectionDetails'
}
```

```
_id: ObjectId("5e8341ae02d2a45370b69245")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40379573"
▸ demographicInfo: Array
▸ inspectionDetails: Array

_id: ObjectId("5e8341af02d2a45370b699c1")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40654198"
▸ demographicInfo: Array
▸ inspectionDetails: Array

_id: ObjectId("5e8341b102d2a45370b6a13b")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40970875"
▸ demographicInfo: Array
▸ inspectionDetails: Array
```

4. Group the stores by zipcode and count the number of Dunkin’ Donuts that are present in that area. (29 Documents)



$unwind — Output after $unwind stage (Sample of 20 documents)

```
/**
 * path: Path to the array field.
 * includeArrayIndex: Optional name for index.
 * preserveNullAndEmptyArrays: Optional
 *   toggle to unwind null and empty values.
 */
{
  path: "$demographicInfo",
  preserveNullAndEmptyArrays: true
}
```

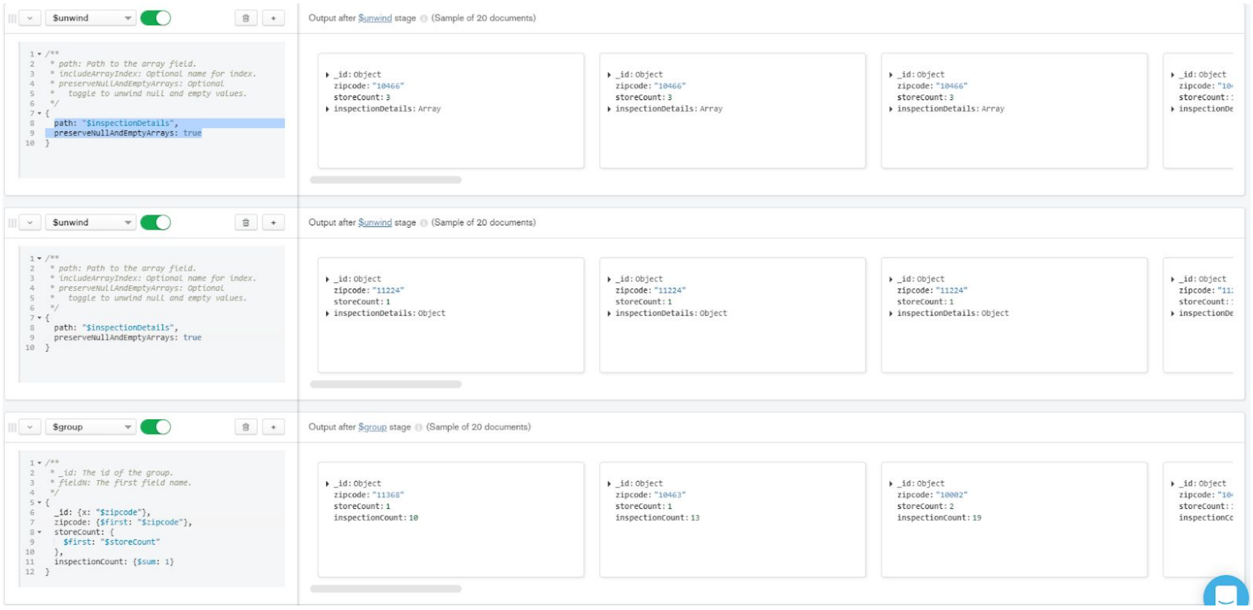
```
_id: ObjectId("5e8341ae02d2a45370b69245")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40379573"
▸ demographicInfo: Object
▸ inspectionDetails: Array

_id: ObjectId("5e8341af02d2a45370b699c1")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40654198"
▸ demographicInfo: Object
▸ inspectionDetails: Array

_id: ObjectId("5e8341b102d2a45370b6a13b")
▸ address: Object
  borough: "Bronx"
  cuisine: "Donuts"
▸ grades: Array
  name: "Dunkin' Donuts"
  restaurant_id: "40970875"
▸ demographicInfo: Object
▸ inspectionDetails: Array
```

$group — Output after $group stage (Sample of 20 documents)

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: {x: "$demographicInfo.JURISDICTION NAME"},
  zipcode: {
    $first: "$demographicInfo.JURISDICTION NAME"
  },
  storeCount: {$sum: 1},
  inspectionDetails: {$push: "$inspectionDetails"}
}
```

```
▸ _id: Object
  zipcode: "10466"
  storeCount: 3
▸ inspectionDetails: Array

▸ _id: Object
  zipcode: "10455"
  storeCount: 1
▸ inspectionDetails: Array

▸ _id: Object
  zipcode: "10460"
  storeCount: 1
▸ inspectionDetails: Array
```

5. Then unwind and group the results by each individual store inspection, counting how many inspections are present within the zipcode



$unwind — Output after $unwind stage (Sample of 20 documents)

```
/**
 * path: Path to the array field.
 * includeArrayIndex: Optional name for index.
 * preserveNullAndEmptyArrays: Optional
 *   toggle to unwind null and empty values.
 */
{
  path: "$inspectionDetails",
  preserveNullAndEmptyArrays: true
}
```

```
▸ _id: Object
  zipcode: "10466"
  storeCount: 3
▸ inspectionDetails: Array

▸ _id: Object
  zipcode: "10466"
  storeCount: 3
▸ inspectionDetails: Array

▸ _id: Object
  zipcode: "10466"
  storeCount: 3
▸ inspectionDetails: Array

▸ _id: Object
  zipcode: "10
  storeCount:
▸ inspectionDe
```

$unwind — Output after $unwind stage (Sample of 20 documents)

```
/**
 * path: Path to the array field.
 * includeArrayIndex: Optional name for index.
 * preserveNullAndEmptyArrays: Optional
 *   toggle to unwind null and empty values.
 */
{
  path: "$inspectionDetails",
  preserveNullAndEmptyArrays: true
}
```

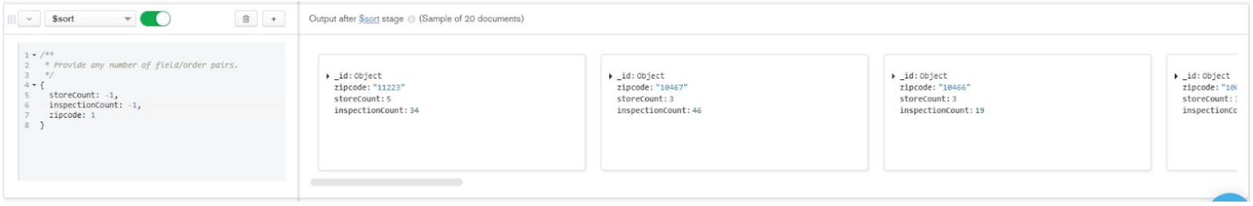
```
▸ _id: Object
  zipcode: "11224"
  storeCount: 1
▸ inspectionDetails: Object

▸ _id: Object
  zipcode: "11224"
  storeCount: 1
▸ inspectionDetails: Object

▸ _id: Object
  zipcode: "11224"
  storeCount: 1
▸ inspectionDetails: Object

▸ _id: Object
  zipcode: "11:
  storeCount:
▸ inspectionDe
```

$group — Output after $group stage (Sample of 20 documents)

```
/**
 * _id: The id of the group.
 * fieldN: The first field name.
 */
{
  _id: {x: "$zipcode"},
  zipcode: {$first: "$zipcode"},
  storeCount: {
    $first: "$storeCount"
  },
  inspectionCount: {$sum: 1}
}
```

```
▸ _id: Object
  zipcode: "11368"
  storeCount: 1
  inspectionCount: 10

▸ _id: Object
  zipcode: "10463"
  storeCount: 1
  inspectionCount: 13

▸ _id: Object
  zipcode: "10002"
  storeCount: 2
  inspectionCount: 19

▸ _id: Object
  zipcode: "10
  storeCount:
  inspectionCc
```

6. Sort the results in descending order based on store count and inspection count, then sort by zipcode in ascending order.



$sort — Output after $sort stage (Sample of 20 documents)

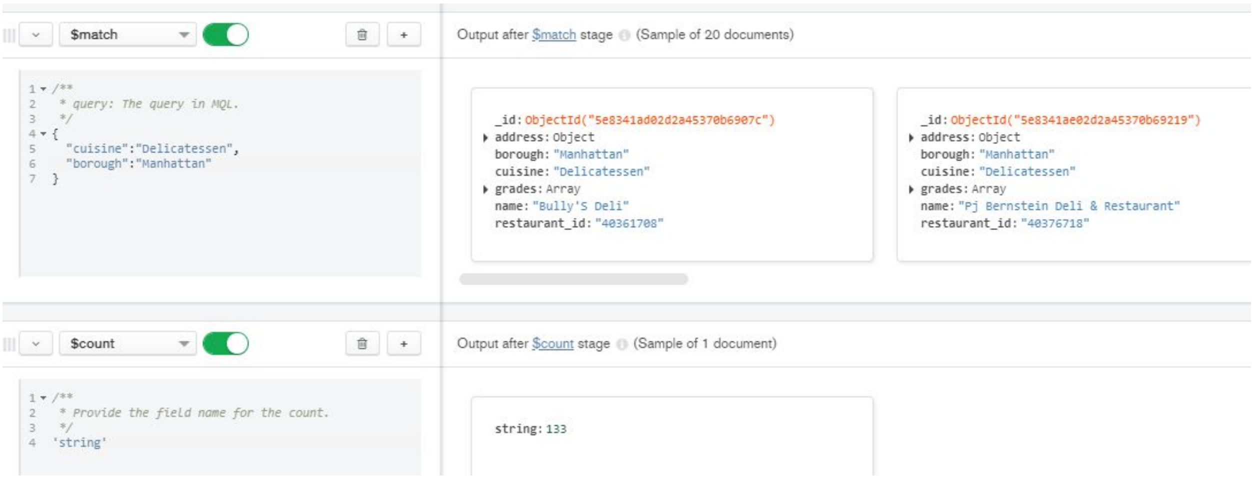
```
/**
 * Provide any number of field/order pairs.
 */
{
  storeCount: -1,
  inspectionCount: -1,
  zipcode: 1
}
```

```
▸ _id: Object
  zipcode: "11223"
  storeCount: 5
  inspectionCount: 34

▸ _id: Object
  zipcode: "10467"
  storeCount: 3
  inspectionCount: 46

▸ _id: Object
  zipcode: "10466"
  storeCount: 3
  inspectionCount: 19

▸ _id: Object
  zipcode: "10
  storeCount:
  inspectionCc
```

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) .



$match — Output after $match stage (Sample of 20 documents)

```
/**
 * query: The query in MQL.
 */
{
  "cuisine":"Delicatessen",
  "borough":"Manhattan"
}
```

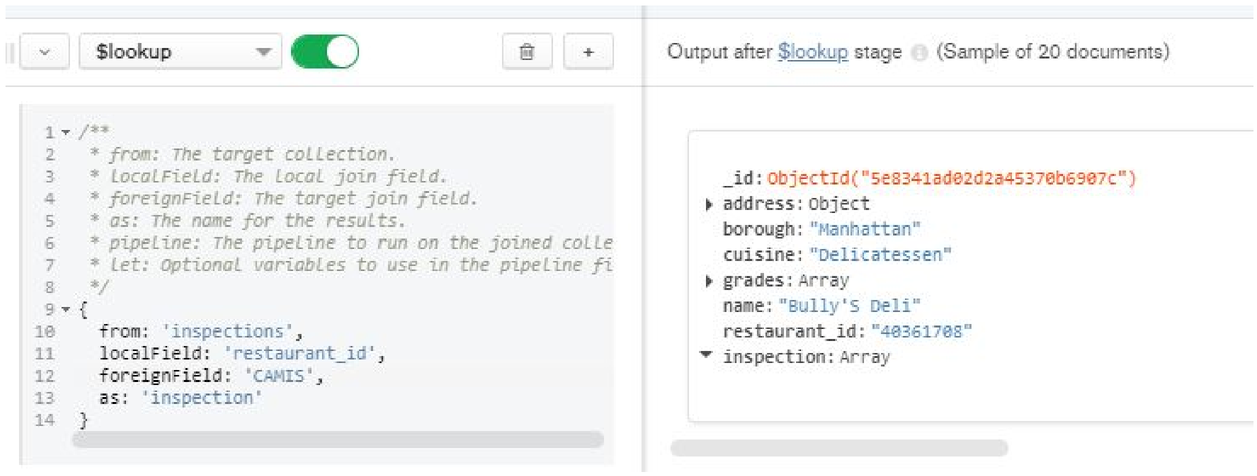
```
  _id: ObjectId("5e8341ad02d2a45370b6907c")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Delicatessen"
▸ grades: Array
  name: "Bully'S Deli"
  restaurant_id: "40361708"

  _id: ObjectId("5e8341ae02d2a45370b69219")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Delicatessen"
▸ grades: Array
  name: "Pj Bernstein Deli & Restaurant"
  restaurant_id: "40376718"
```

$count — Output after $count stage (Sample of 1 document)

```
/**
 * Provide the field name for the count.
 */
'string'
```

```
string: 133
```

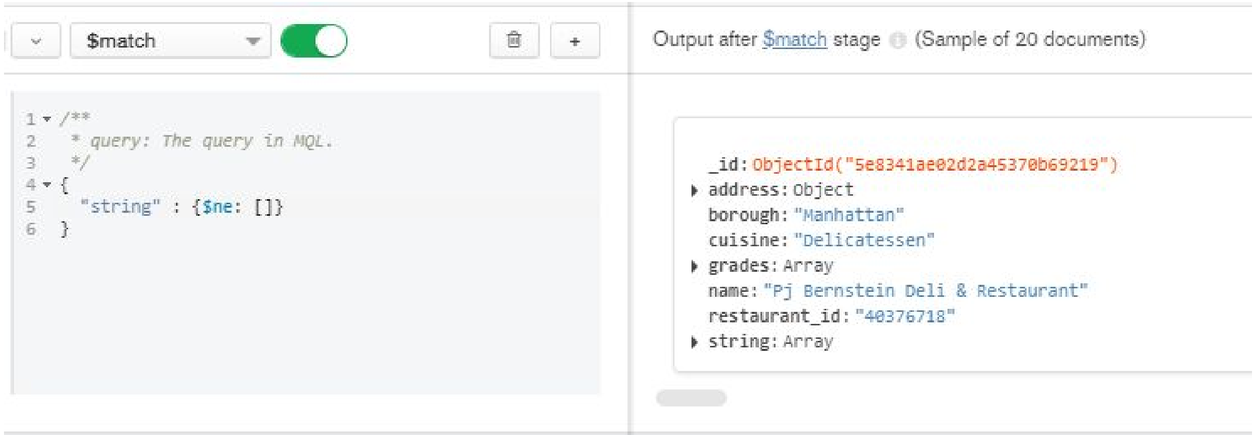
2. Using the $lookup function, join the view table created in step 1 with the inspections table on restaurant_id and CAMIS. Use $out to create a copy of this. (133 rows)



```
/**
 * from: The target collection.
 * localField: The local join field.
 * foreignField: The target join field.
 * as: The name for the results.
 * pipeline: The pipeline to run on the joined colle
 * let: Optional variables to use in the pipeline fi
 */
{
  from: 'inspections',
  localField: 'restaurant_id',
  foreignField: 'CAMIS',
  as: 'inspection'
}
```

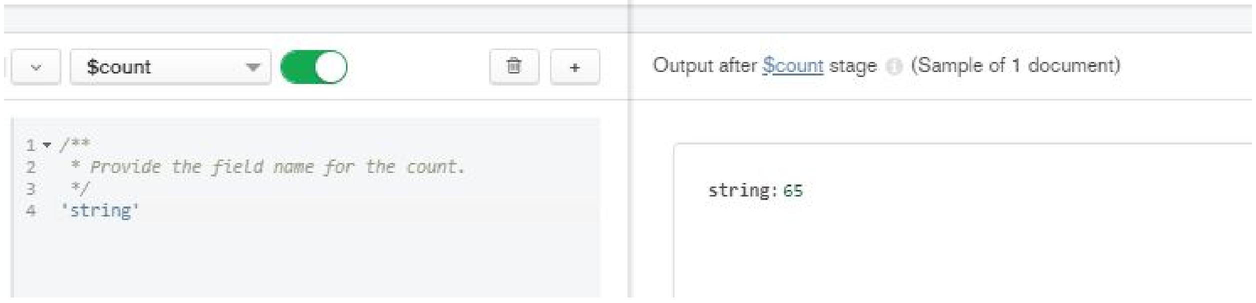
```
_id: ObjectId("5e8341ad02d2a45370b6907c")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Delicatessen"
▸ grades: Array
  name: "Bully'S Deli"
  restaurant_id: "40361708"
▾ inspection: Array
```

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



```
/**
 * query: The query in MQL.
 */
{
  "string" : {$ne: []}
}
```

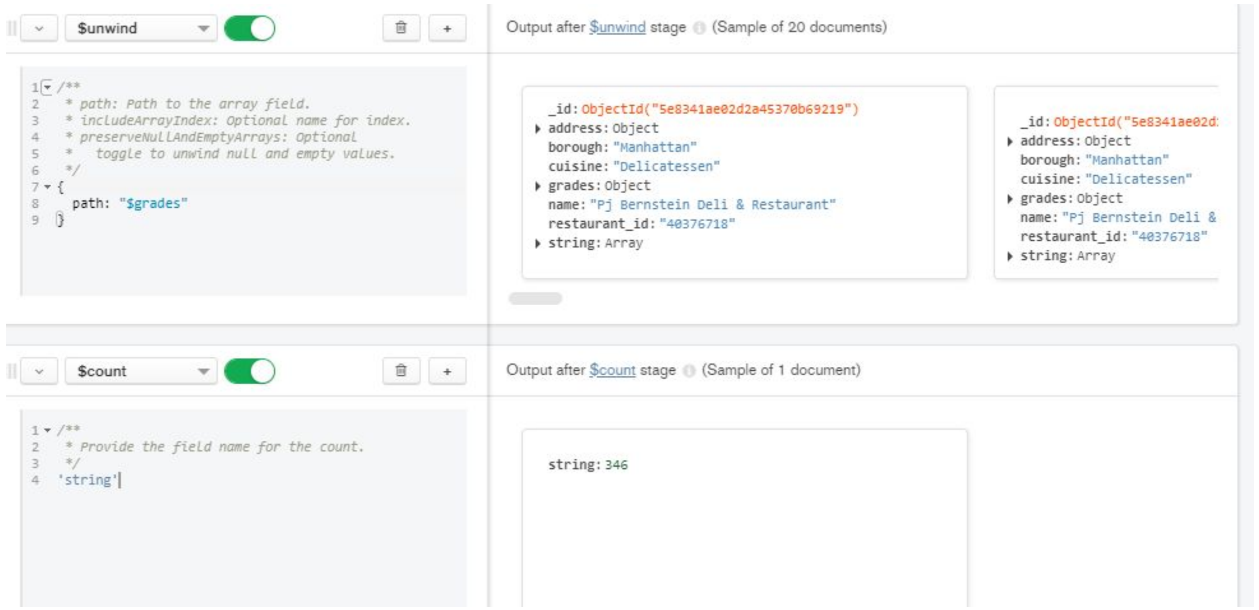
```
_id: ObjectId("5e8341ae02d2a45370b69219")
▸ address: Object
  borough: "Manhattan"
  cuisine: "Delicatessen"
▸ grades: Array
  name: "Pj Bernstein Deli & Restaurant"
  restaurant_id: "40376718"
▸ string: Array
```



```
/**
 * Provide the field name for the count.
 */
'string'
```

```
string: 65
```

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



```
/**
 * path: Path to the array field.
 * includeArrayIndex: Optional name for index.
 * preserveNullAndEmptyArrays: Optional
 *   toggle to unwind null and empty values.
 */
{
  path: "$grades"
}
```

Output after $unwind stage (Sample of 20 documents)

```
_id: ObjectId("5e8341ae02d2a45370b69219")
▸ address: Object
borough: "Manhattan"
cuisine: "Delicatessen"
▸ grades: Object
name: "Pj Bernstein Deli & Restaurant"
restaurant_id: "40376718"
▸ string: Array
```

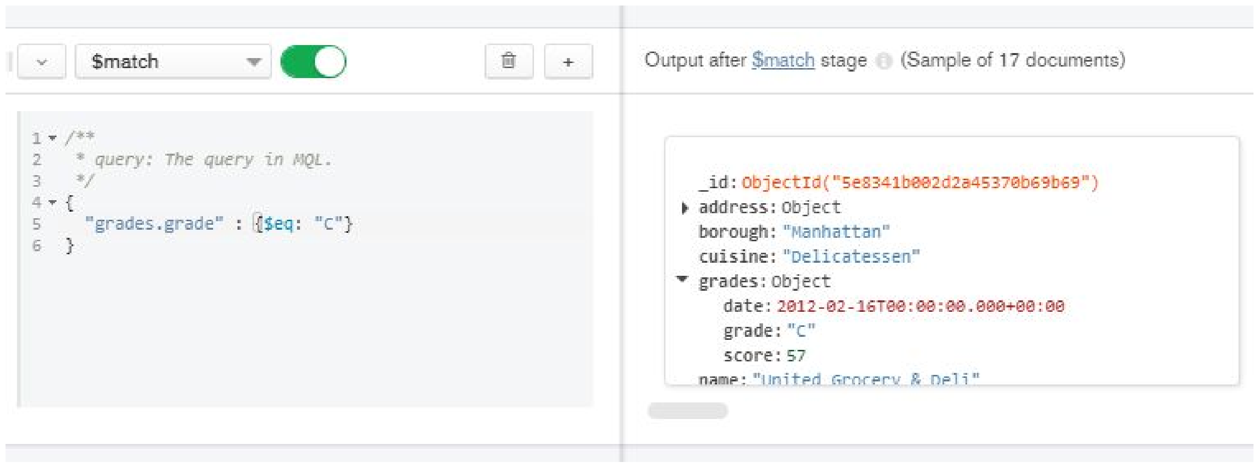
```
_id: ObjectId("5e8341ae02d
▸ address: Object
borough: "Manhattan"
cuisine: "Delicatessen"
▸ grades: Object
name: "Pj Bernstein Deli &
restaurant_id: "40376718"
▸ string: Array
```

```
/**
 * Provide the field name for the count.
 */
'string'
```

Output after $count stage (Sample of 1 document)

```
string: 346
```

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

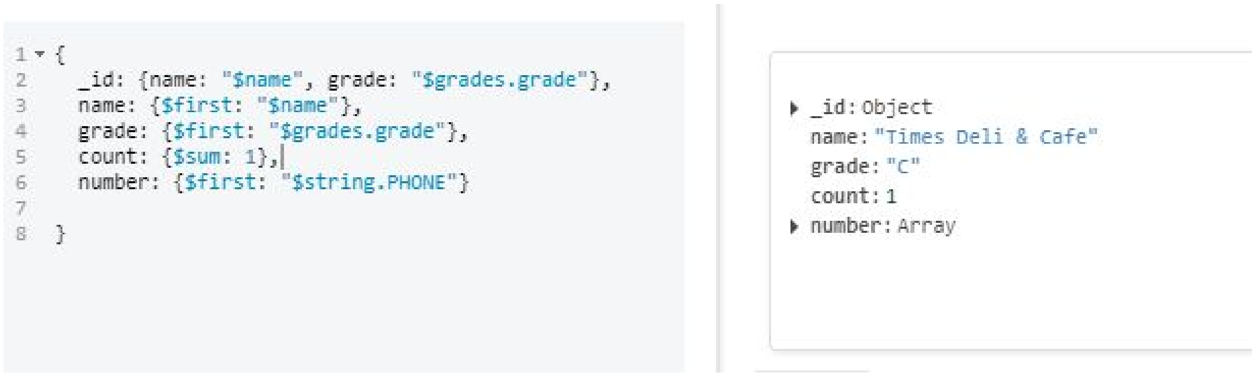


```
/**
 * query: The query in MQL.
 */
{
  "grades.grade" : {$eq: "C"}
}
```

Output after $match stage (Sample of 17 documents)

```
_id: ObjectId("5e8341b002d2a45370b69b69")
▸ address: Object
borough: "Manhattan"
cuisine: "Delicatessen"
▾ grades: Object
    date: 2012-02-16T00:00:00.000+00:00
    grade: "C"
    score: 57
name: "United Grocery & Deli"
```

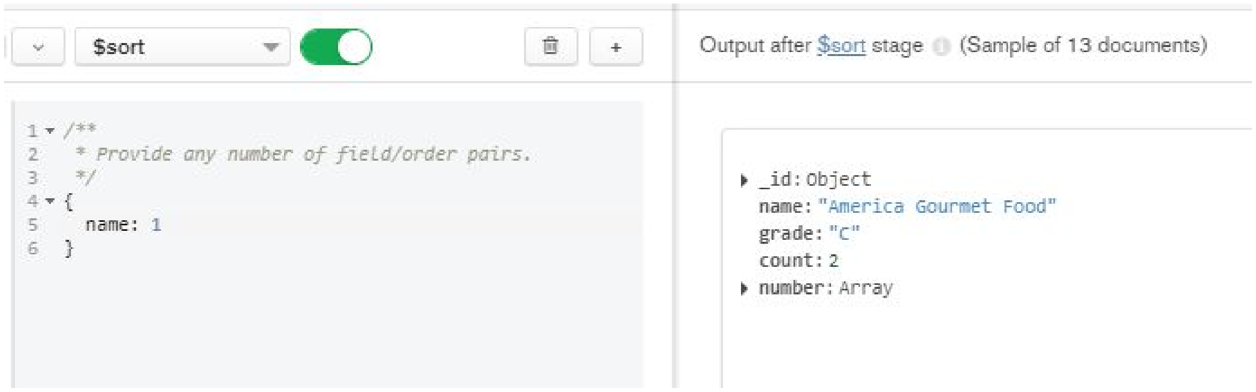
6. Group by name and grade for each restaurant with a count for each restaurant. Additionally, grab the number in an array using the “number: {$first: $string.PHONE}”. This will yield an entire array of numbers, but leave it this way for simplicity's sake (13 documents).



```
{
  _id: {name: "$name", grade: "$grades.grade"},
  name: {$first: "$name"},
  grade: {$first: "$grades.grade"},
  count: {$sum: 1},
  number: {$first: "$string.PHONE"}

}
```

```
▸ _id: Object
  name: "Times Deli & Cafe"
  grade: "C"
  count: 1
▸ number: Array
```

7. Sort Names in Ascending Order (13 documents)



$sort

Output after $sort stage (Sample of 13 documents)

```
/**
 * Provide any number of field/order pairs.
 */
{
  name: 1
}
```

```
▸ _id: Object
  name: "America Gourmet Food"
  grade: "C"
  count: 2
▸ number: Array
```

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.