|  |
| --- |
| Close-up image showing the leaf-sides of two oversized books side-by-side on a bookshelf, with additional books in soft focus background |
| MongoDB Project  Advanced Database |
| |  |  |  | | --- | --- | --- | | Reonne Benoy | 5/8/20 | Athlone Institute of Technology, Athlone | |

Contents

[Project Overview 2](#_Toc39859590)

[Database Creation 2](#_Toc39859591)

[Create Atlas Account 2](#_Toc39859592)

[Deploy a Free Tier Cluster 2](#_Toc39859593)

[Whitelist Connection IP Address 2](#_Toc39859594)

[Create the Database User for the Cluster 2](#_Toc39859595)

[Connect to Your Cluster using Mongo Shell 2](#_Toc39859596)

[Insert Data in the Cluster 4](#_Toc39859597)

[Insert using Atlas UI 4](#_Toc39859598)

[MongoDB Basic Queries 5](#_Toc39859599)

[Aggregate Pipeline Queries 24](#_Toc39859600)

[Compass and Atlas functions 31](#_Toc39859601)

[Youtube Video Links 38](#_Toc39859602)

[MongoDB Aggregation part -1 38](#_Toc39859603)

[MongoDB Aggregation part-2 38](#_Toc39859604)

[References 39](#_Toc39859605)

Table of Figures

[Figure 1: Using filter on Annual Turnover 31](#_Toc39859498)

[Figure 2: Schema view of the output 31](#_Toc39859499)

[Figure 3: Annual Turnover filter 31](#_Toc39859500)

[Figure 4: Applying filter on document 32](#_Toc39859501)

[Figure 5: Schema View of the output 32](#_Toc39859502)

[Figure 6: Applying filter on the collection 33](#_Toc39859503)

[Figure 7: Schema View of the output 33](#_Toc39859504)

[Figure 8: The Schema view of Nationality and Net\_Worth 34](#_Toc39859505)

[Figure 9: Applying multiple options for filter 34](#_Toc39859506)

[Figure 10: Schema view of the Output 35](#_Toc39859507)

[Figure 11: Schema View of Frequency 35](#_Toc39859508)

[Figure 12: Highlight the years in the output document 35](#_Toc39859509)

[Figure 13: Aggregation pipeline field box with the respective filter query 36](#_Toc39859510)

[Figure 14: Output document detail using Aggregation Pipeline 36](#_Toc39859511)

[Figure 15: Pipeline entire view 37](#_Toc39859512)

[Figure 16: Seperate field box for different Aggregate Stages 37](#_Toc39859513)

[Figure 17: Youtube Video of Aggregation Part-1 38](#_Toc39859514)

[Figure 18: Youtube Video of Aggregation Part-2 38](#_Toc39859515)

MongoDB

# Project Overview

The MongoDB project is a brief description and a practical approach of explanation of the various functionalities of Mongo DB. In this project, we see the creation of database on MongoDB through Mongo Atlas and Compass UI along with building clusters on the MongoDB web servers. The database creation is accompanied by data population and insertion of data. We follow through all the MongoDB query selectors and operators. The project document is sectioned into four parts:

1. Database Creation
2. MongoDB simple queries
3. MongoDB Aggregation Framework
4. MongoDB Queries using Compass

By the end of this project, a more than adequate knowledge of MongoDB will be acquired through practise and test runs. The project documentation will be included with video files of explanation of some queries.

# Database Creation

## Create Atlas Account

1. Sign up with Google by entering the Gmail address. Associated phone number could also be given.
2. In the next stage, enter the password and proceed.
3. Accept the terms of service and the Privacy Policy.
4. Submit for creation.
5. Login with the credentials.

## Deploy a Free Tier Cluster

1. After logging in to Atlas, click on Build a Cluster.
2. Select Starter Clusters and Create a Cluster.
3. Select the preferred Cloud Provider and Region.
4. Select M0 Sandbox for Cluster tier.
5. Enter the preferred name for the cluster and create Cluster for deployment.

## Whitelist Connection IP Address

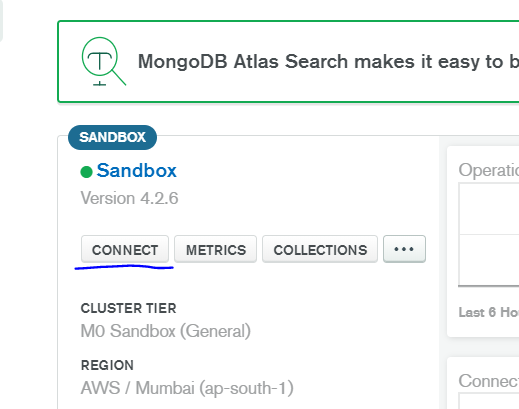
1. Open the connect dialog by clicking on the connect button.
2. Configure the whitelist entry by adding your current IP address.
3. Add IP address and this will configure the whitelist.

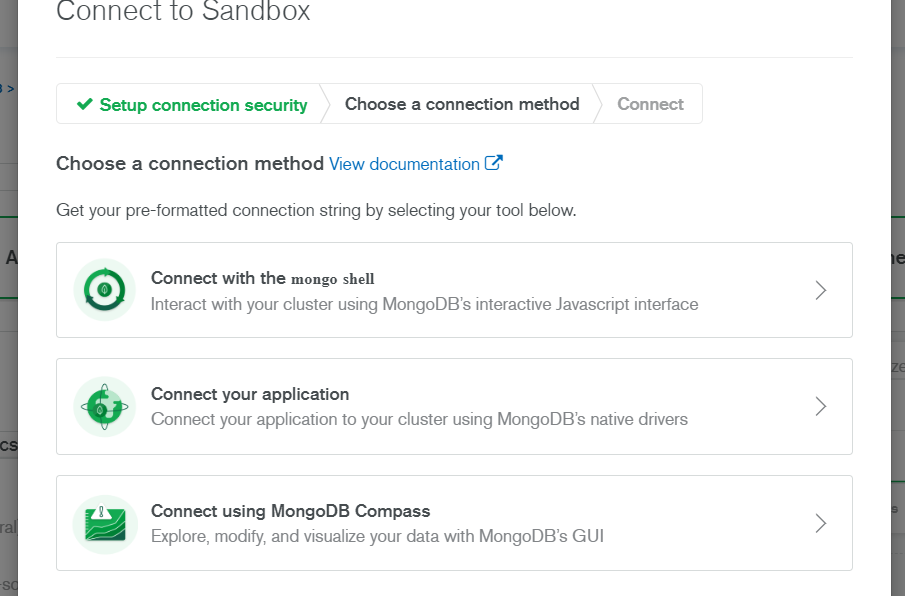
## Create the Database User for the Cluster

1. Open the connect dialog.
2. In the Create a MongoDB User step of the dialog, enter the Username and a Password.
3. Create MongoDB User.

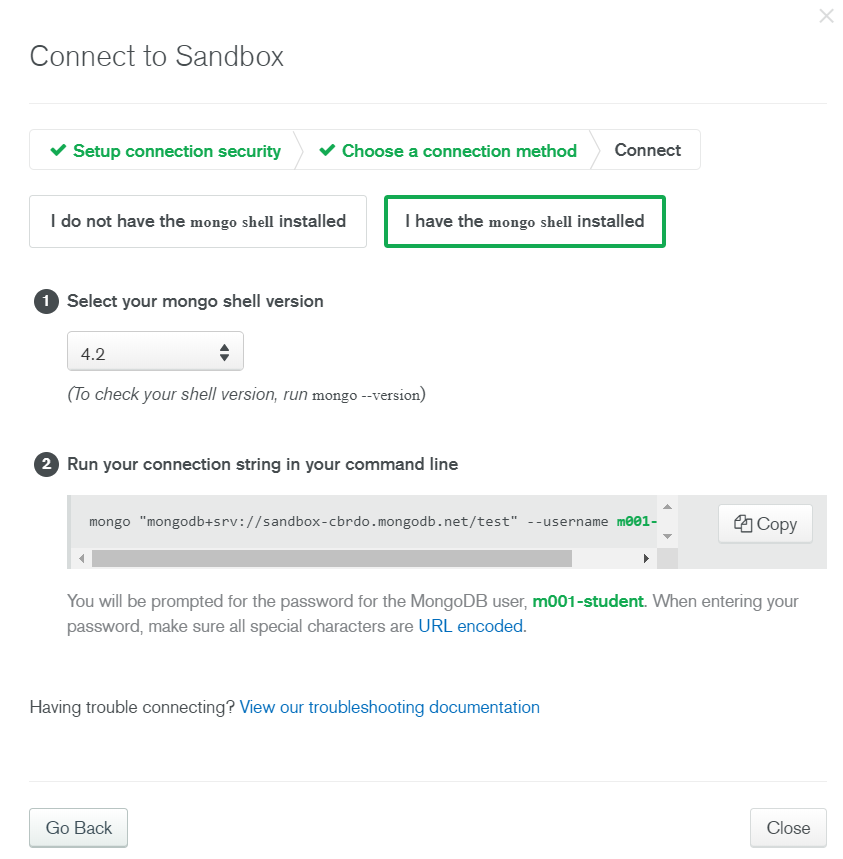
## Connect to Your Cluster using Mongo Shell

1. Download and Install the Shell.
2. Open the Connect Dialog and click on Connect with the Mongo Shell.





1. Click on I already have the Mongo Shell installed if it is installed and click on the Mongo version and operating system. If not, click on I do not have Mongo Shell installed and select the operating system. Then download Mongo Shell.
2. Add the mongo shell to the system path.
3. If the Mongo shell is already installed, there will be an option to copy the provided connection string to your clipboard and paste it in the terminal and run it.

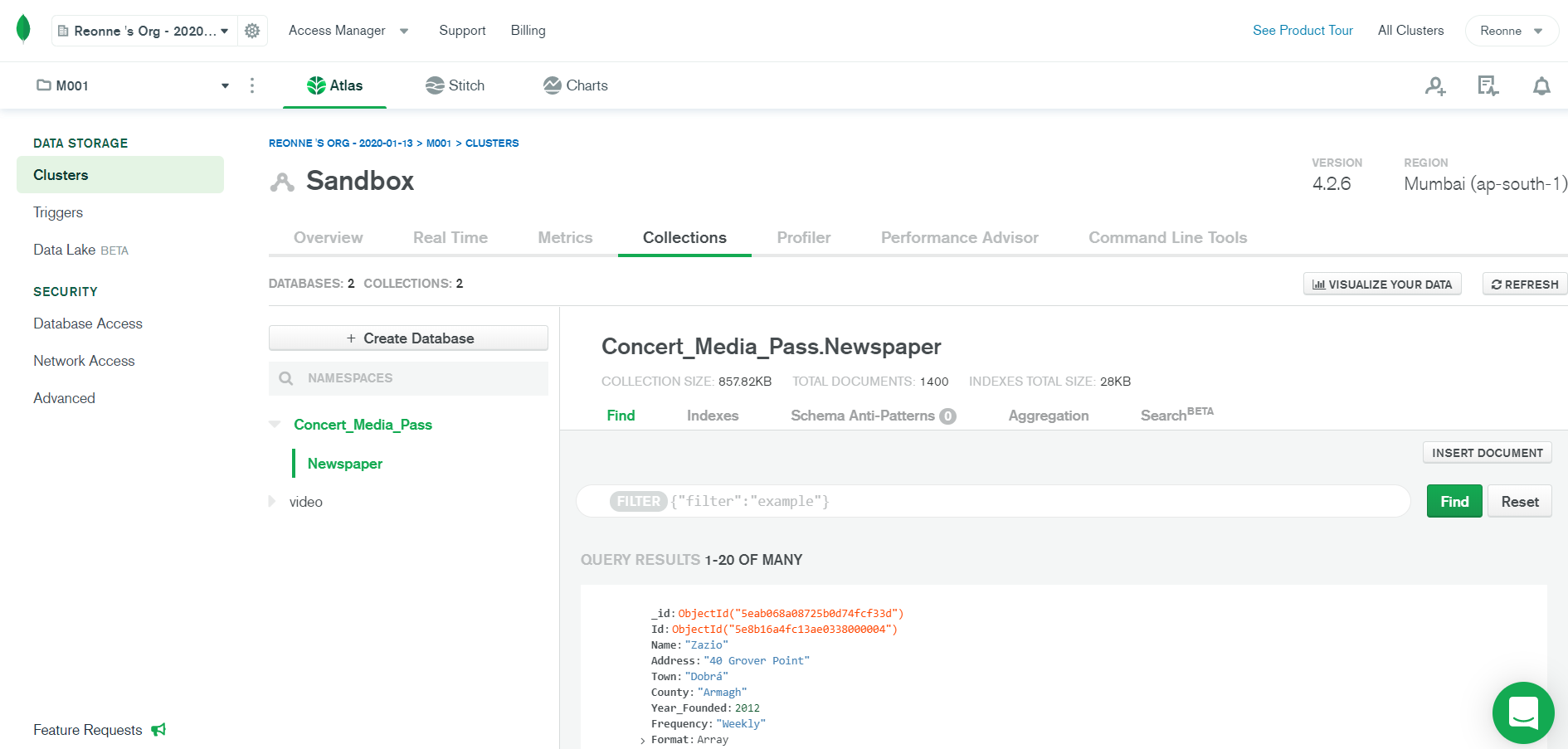


1. When prompted, enter your database user’s password.
2. Test the Mongo shell installation.

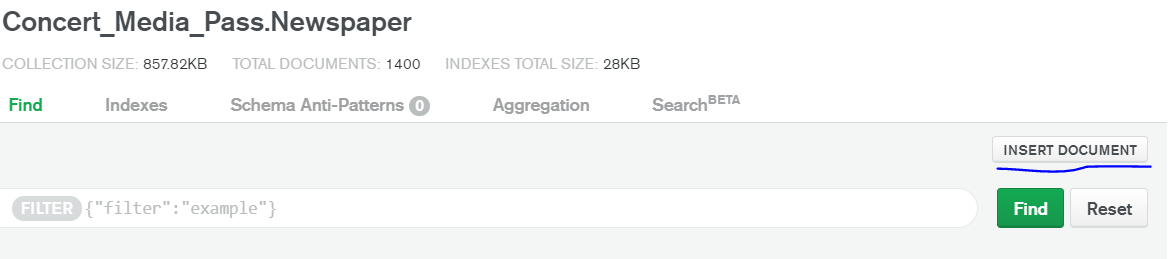
## Insert Data in the Cluster

## Insert using Atlas UI

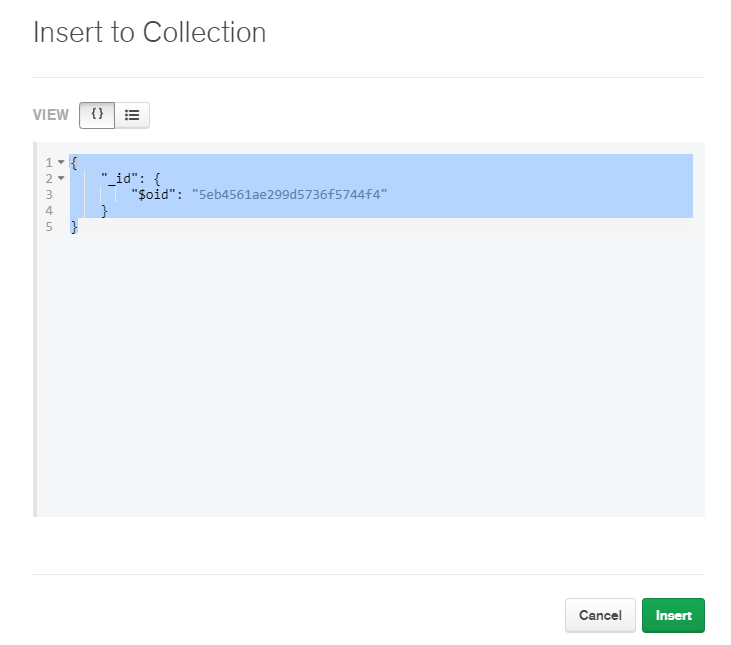
1. Open Data Explorer.



1. Click on **Insert Document** option.



1. Enter the document details and insert the document.



# MongoDB Basic Queries

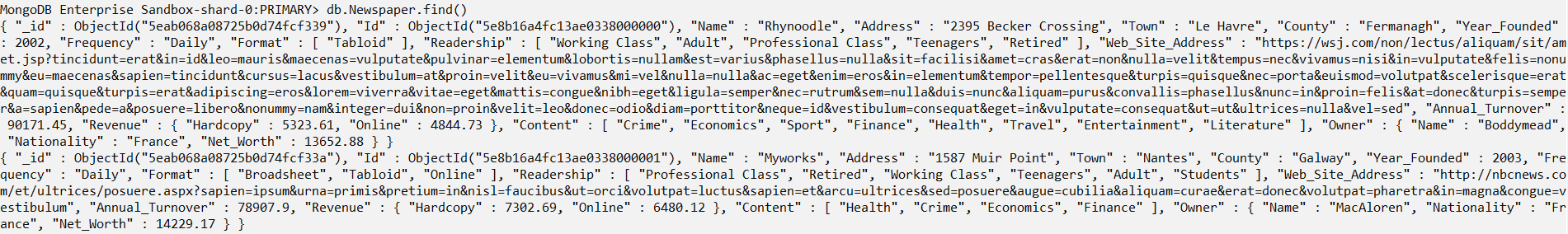
1. **Retrieve all the documents from the collection and arrange them in an easy to read format. Also return the total number of documents.**

As we have already populated our database with the appropriate data, we can retrieve data from it. First, we retrieve the number of documents in the database collection.

*db.Newspaper.find().count()*



*db.Newspaper.find()*



*db.Newspaper.find().pretty()*



1. **Find the documents of the newspaper company founded after 2012 where the frequency of newspaper delivery is monthly.**



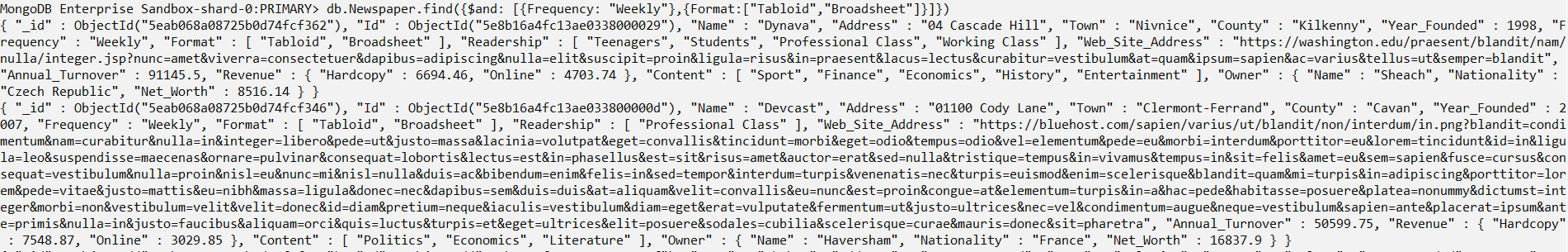
*db.Newspaper.find({$and:[{Frequency: "Monthly"},{Year\_Founded:{$gt:2012}}]})*



1. **Find all the documents where the frequency of newspaper delivery is weekly and the format of the newspaper is tabloid and broadsheet. Return the number of documents found.**



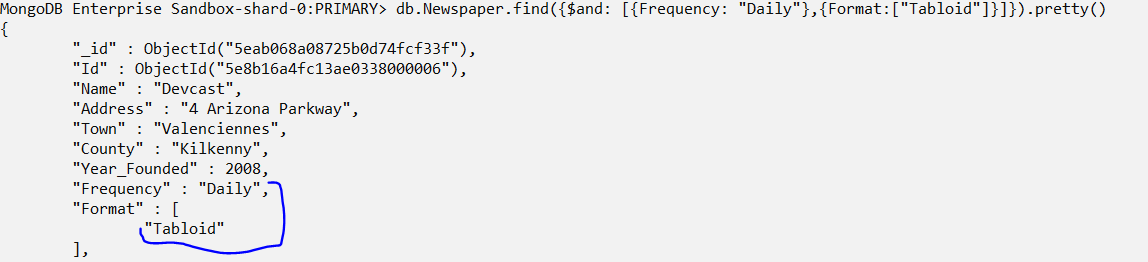
*db.Newspaper.find({$and:[{Frequency:"Weekly"},{Format:["Tabloid","Broadsheet"]}]})*

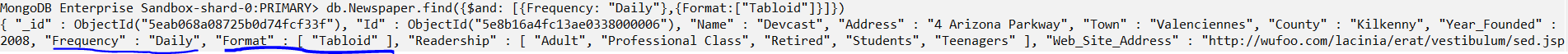


1. **Find all the documents where the newspaper is delivered daily and the format is tabloid. Return the number of documents and present them in an easy-to-read format.**



*db.Newspaper.find({$and: [{Frequency:"Daily"},{Format:["Tabloid"]}]})*

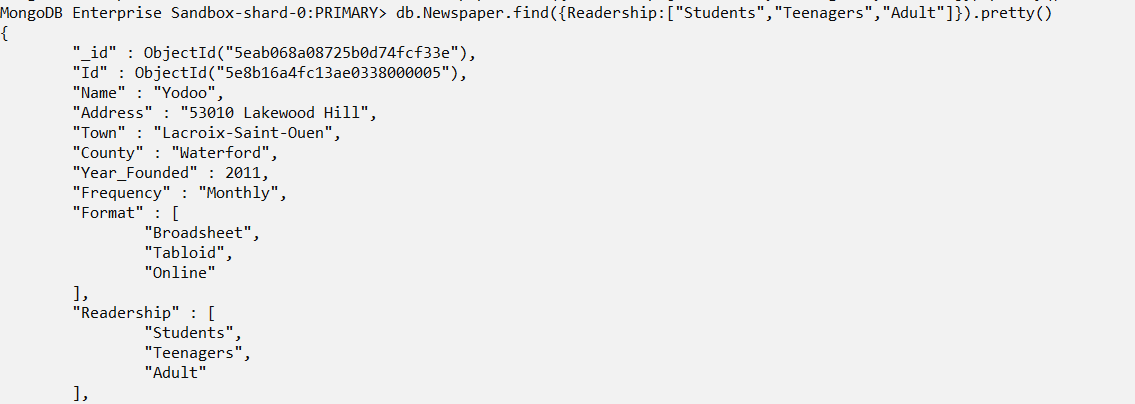




1. **Find the number of documents which is for students, teenagers and adult and in that sequence.**



*db.Newspaper.find({Readership:["Students","Teenagers","Adult"]})*



1. **Find the number of documents which is for students, teenagers and adult and are in any sequence.**



*db.Newspaper.find({Readership:{$all: ["Students","Teenagers","Adult"]}})*

1. **Find the number of documents which is for adults, retired, professional class and students in any sequence and also return the documents details.**



*db.Newspaper.find({Readership:{$all: ["Adult","Retired","Professional Class"]}})*



1. **Find the number of documents which are for adults, retired, professional class, and students, in that sequence.**

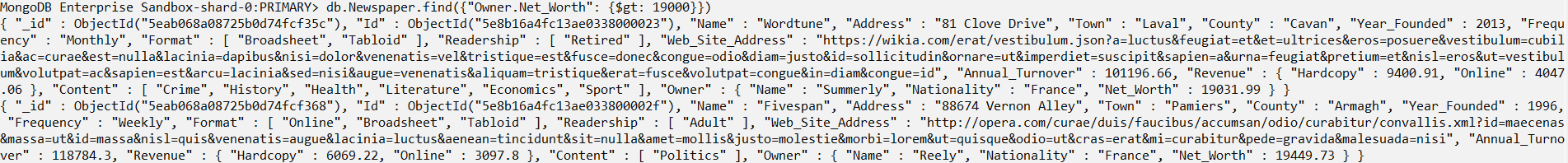


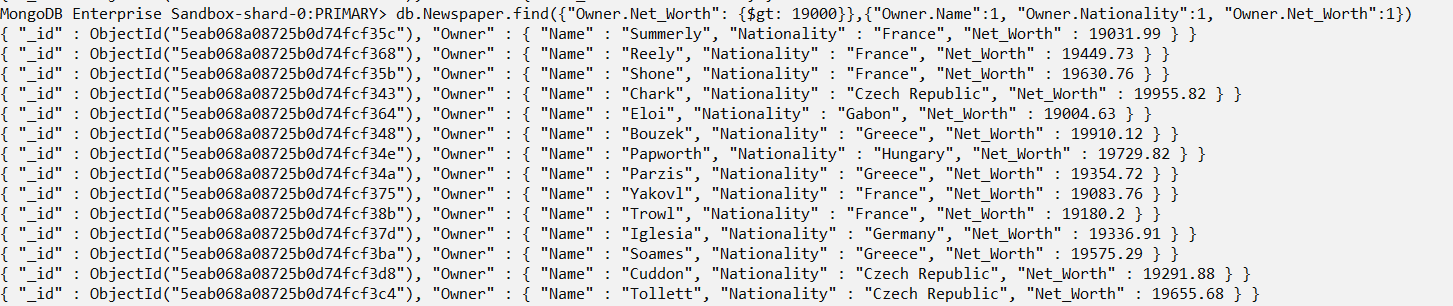
*db.Newspaper.find({Readership:["Adult","Retired","Professional Class","Students"]}).count()*

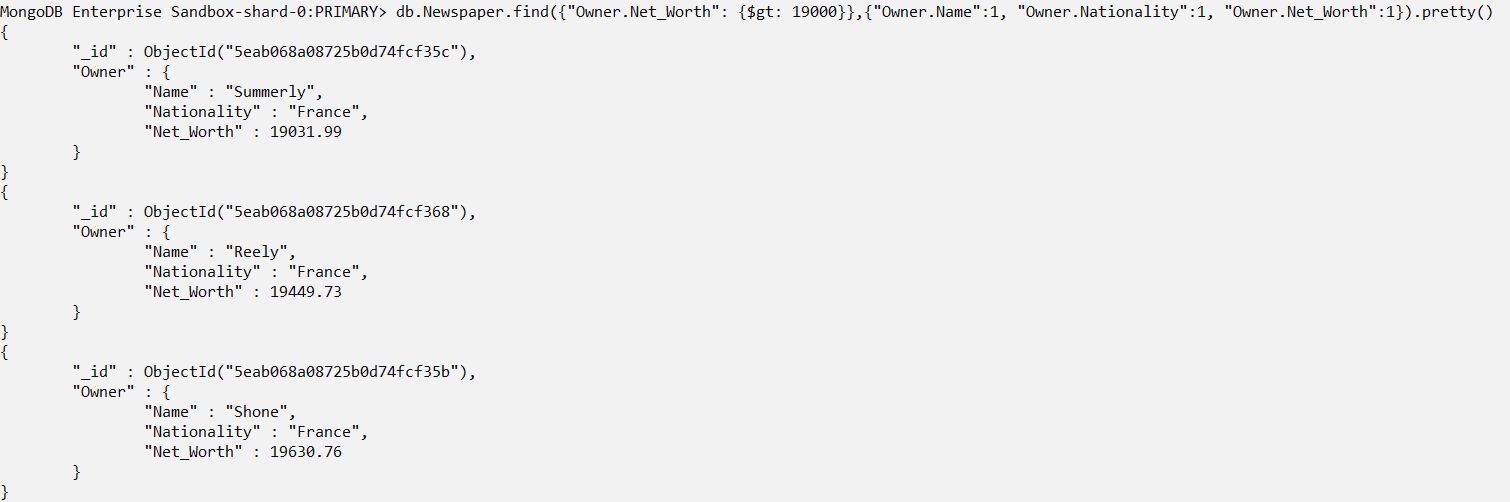
1. **Return the newspaper owner details of those whose net worth is greater than 19000 euros.**



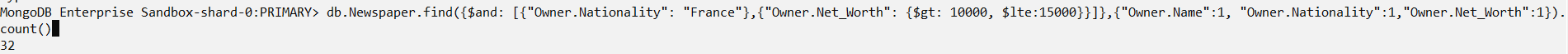
*db.Newspaper.find({"Owner.Net\_Worth":{$gt:19000}},{"Owner":1})*



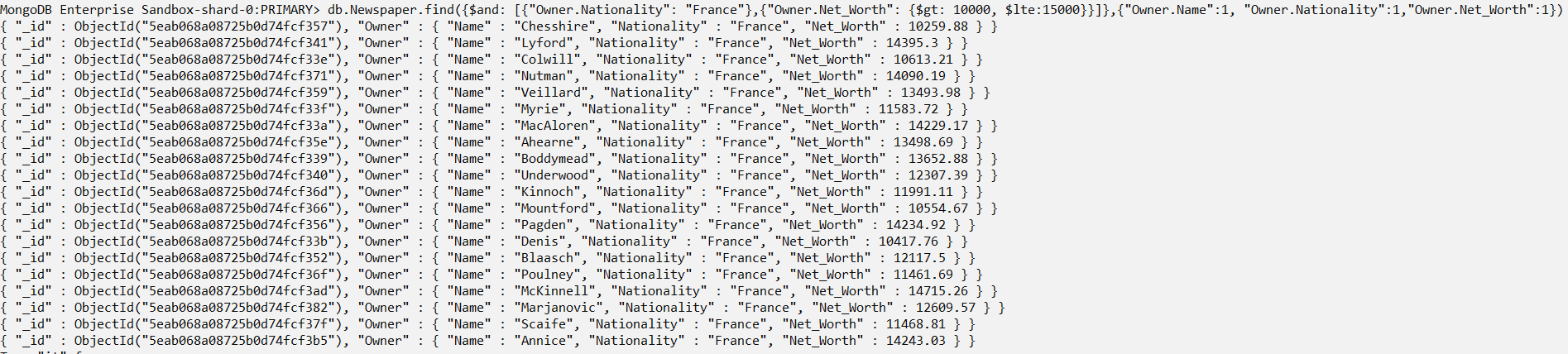




1. **Return the owner details of those whose net worth is greater than 10000 euros but less than 15000 euros and are from France.**



*db.Newspaper.find({$and:[{"Owner.Nationality":"France"},{"Owner.Net\_Worth":{$gt:10000,$lt:15000}}]},{"Owner":1})*

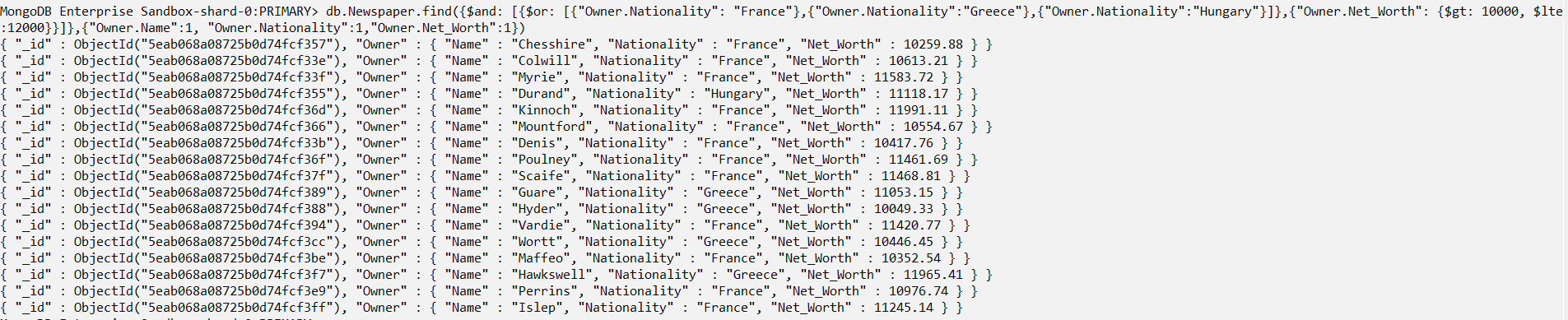




1. **Return the owner details of those whose net worth is greater than 10000 euros but less than 12000 euros and are from France, Greece or Hungary. Also return the number of documents.**



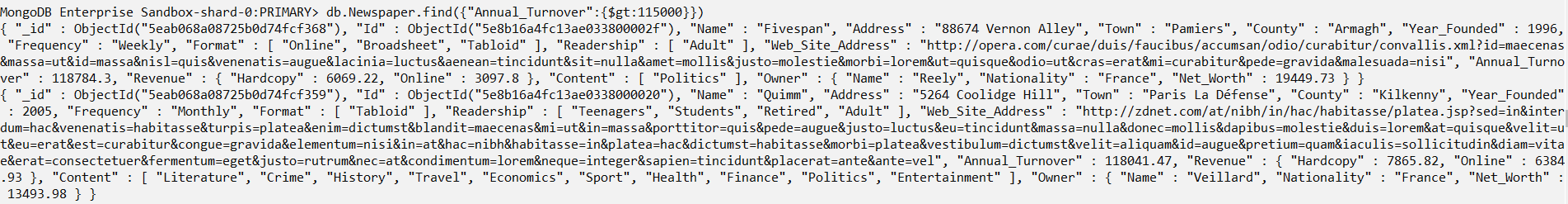
*db.Newspaper.find({$and:[{$or:[{"Owner.Nationality":"France"},{"Owner.Nationality":"Greece"},{"Owner.Nationality":"Hungary"}]},{"Owner.Net\_Worth": {$gt: 10000, $lte:12000}}]},{"Owner.Name":1, "Owner.Nationality":1,"Owner.Net\_Worth":1}).count()*



1. **Return the documents whose annual turnover is greater than 115000 euros.**



*db.Newspaper.find({"Annual\_Turnover":{$gt:115000}},{\_id:1,"Name":1,"Year\_Founded":1,"Format":1})*

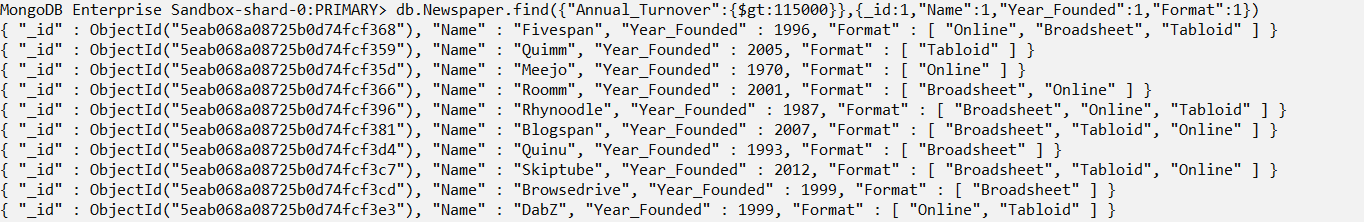


1. **Return the documents whose annual turnover is greater than 115000 euros and return the name, year founded and the format of the newpaper.**

*db.Newspaper.find({"Annual\_Turnover":{$gt:11500}},{\_id:1,"Name":1,"Year\_Founded":1,"Format":1}).count()*



*db.Newspaper.find({"Annual\_Turnover":{$gt:11500}},{\_id:1,"Name":1,"Year\_Founded":1,"Format":1})*



1. **Return the documents whose annual turnover is less than 15500 euros.**

*db.Newspaper.find({"Annual\_Turnover":{$lt:15500}}).count()*



*db.Newspaper.find({"Annual\_Turnover":{$lt:15500}})*



1. **Return the documents whose annual turnover is less than 15500 euros and return the name, year founded, the format and the frequency of the newspaper.**

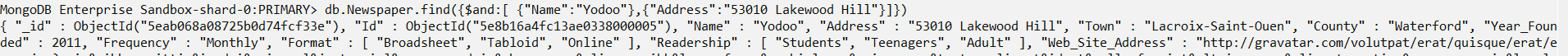
*db.Newspaper.find({"Annual\_Turnover":{$lt:15500}},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"Format":1}).count()*



*db.Newspaper.find({"Annual\_Turnover":{$lt:15500}},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"Format":1})*



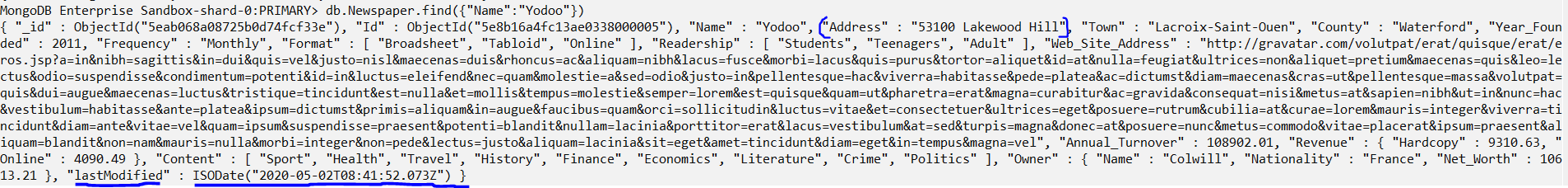
1. **Update the address of the newspaper Yodoo to 53100 Lakewood Hill.**

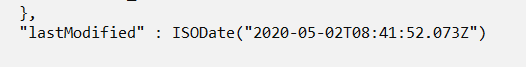


*db.Newspaper.updateOne({$and: [{"Name":"Yodoo"},{"Address":"53010 Lakewood Hill"}]},{$set:{"Address":"53100 Lakewood Hill"}, $currentDate:{lastModified:true}})*



*db.Newspaper.find({"Name":"Yodoo"})*

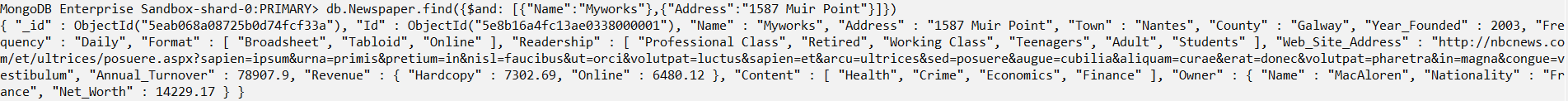






1. **Update the address and town of the newspaper Myworks to 1489 Muir Point and Le Havre.**

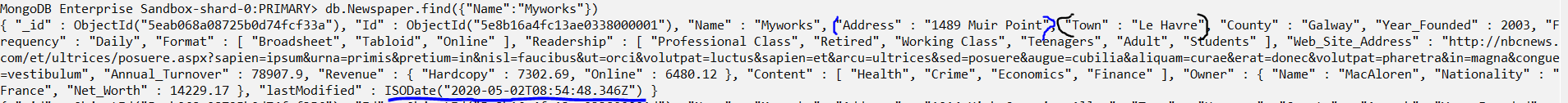
*db.Newspaper.find({$and: [{"Name":"Myworks"},{"Address":"1587 Muir Point"}]})*



*db.Newspaper.updateOne({$and: [{"Name":"Myworks"},{"Address":"1587 Muir Point"}]},{$set: {"Address":"1489 Muir Point","Town":"Le Havre"},$currentDate: {lastModified:true}})*

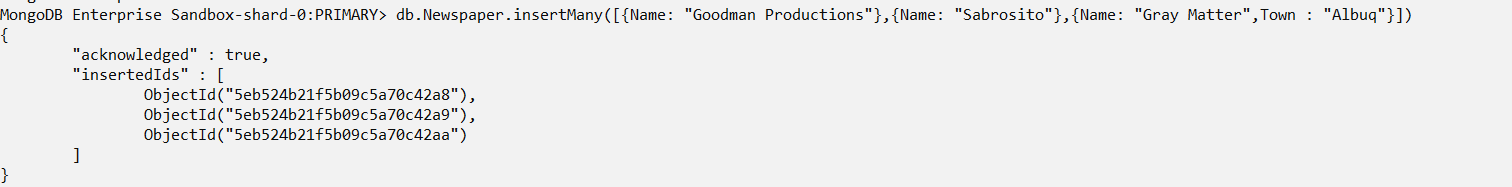


*db.Newspaper.find({"Name":"Myworks"})*



1. **Insert three documents into the collection.**

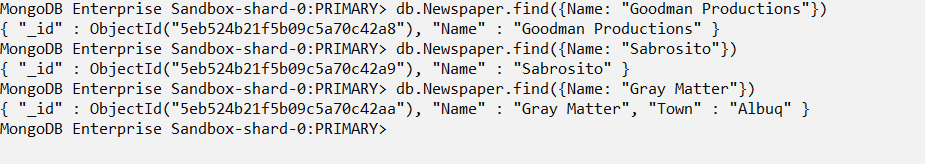
*db.Newspaper.insertMany([{Name: "Goodman Productions"},{Name: "Sabrosito"},{Name: "Gray Matter",Town : "Albuq"}])*



*db.Newspaper.find({Name: "Goodman Productions"})*

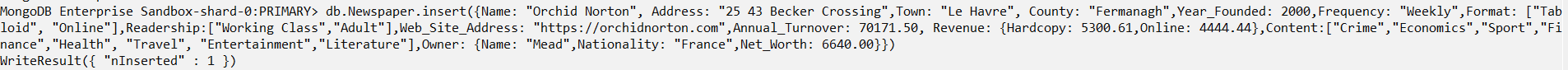
*db.Newspaper.find({Name: "Sabrosito"})*

*db.Newspaper.find({Name: "Gray Matter"})*

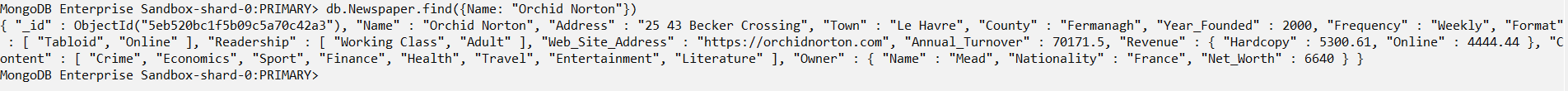


1. **Insert one document into the collection.**

*db.Newspaper.insert({Name: "Orchid Norton", Address: "25 43 Becker Crossing",Town: "Le Havre", County: "Fermanagh",Year\_Founded: 2000,Frequency: "Weekly",Format: ["Tabloid", "Online"],Readership:["Working Class","Adult"],Web\_Site\_Address: "https://orchidnorton.com",Annual\_Turnover: 70171.50, Revenue: {Hardcopy: 5300.61,Online: 4444.44},Content:["Crime","Economics","Sport","Finance","Health", "Travel", "Entertainment","Literature"],Owner: {Name: "Mead",Nationality: "France",Net\_Worth: 6640.00}})*



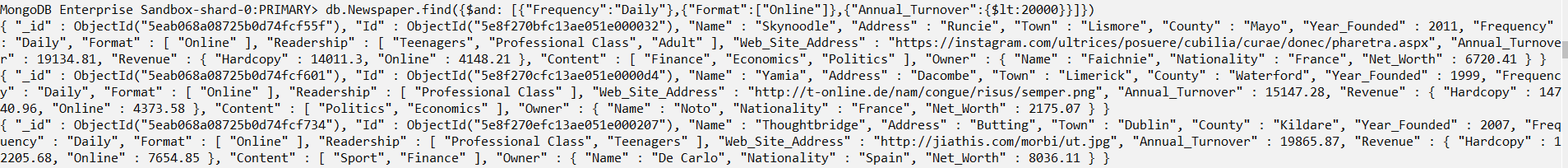
*db.Newspaper.find({Name: "Orchid Norton"})*



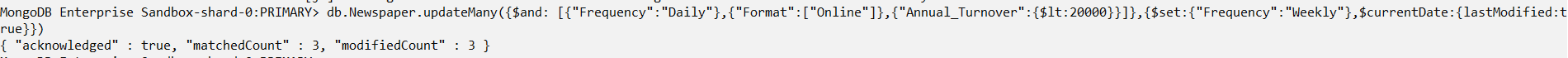
1. **Update frequency of delivery of the documents in the collection where the format is online and the annual turnover of the newspaper company is less than 20000 euros.**



*db.Newspaper.find({$and:[{"Frequency":"Daily"},{"Format":["Online"]},{"Annual\_Turnover":{$lt:20000}}]}).count()*

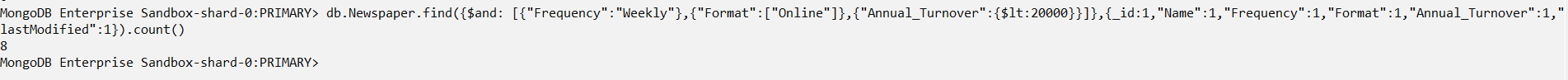
**

*db.Newspaper.updateMany({$and:[{"Frequency":"Daily"},{"Format":["Online"]},{"Annual\_Turnover”:{$lt:20000}}]},{$set:{"Frequency":"Weekly"},$currentDate:{lastModified:true}})*

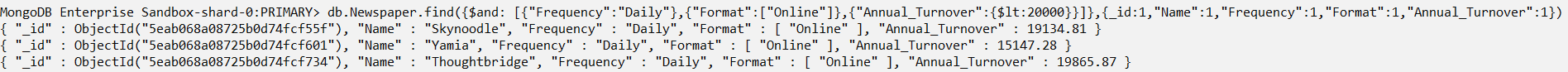




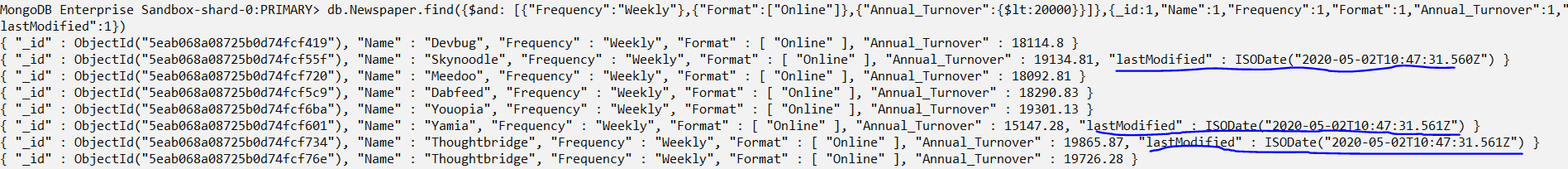
*db.Newspaper.find({$and:[{"Frequency":"Weekly"},{"Format":["Online"]},{"Annual\_Turnover":{$lt:2 0000}}]},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Annual\_Turnover":1,"lastModified":1}).count()*



*db.Newspaper.find({$and:[{"Frequency":"Daily"},{"Format":["Online"]},{"Annual\_Turnover":{$lt:20000}}]},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Annual\_Turnover":1})*

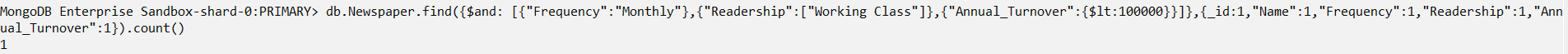


*db.Newspaper.find({$and:[{"Frequency":"Weekly"},{"Format":["Online"]},{"Annual\_Turnover":{$lt:20000}}]},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Annual\_Turnover":1,"lastModified":1})*



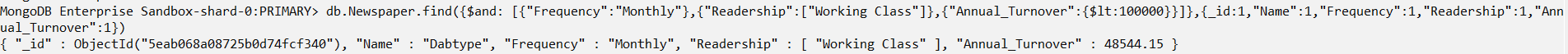
1. **Update the frequency of delivery in the documents from monthly where the newspapers are for Working class and the company annual turnover is less than 100000.**

*db.Newspaper.find({$and: [{"Frequency":"Monthly"},{"Readership":["Working Class"]},{"Annual\_Turnover":{$lt:100000}}]},{\_id:1,"Name":1,"Frequency":1,"Readership":1,"Annual\_Turnover":1}).count()*

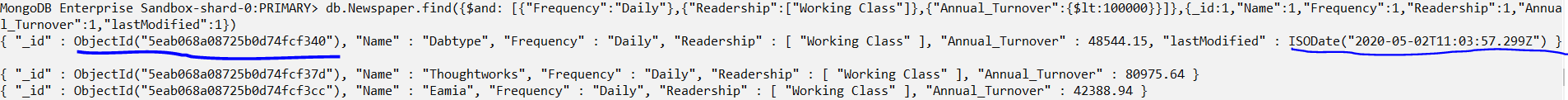


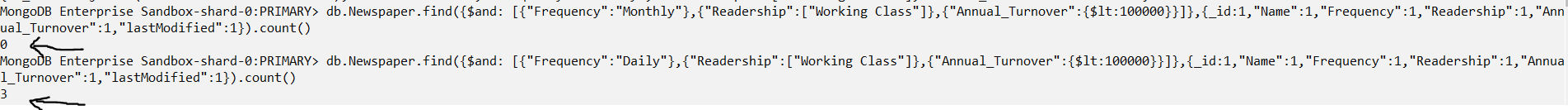
*db.Newspaper.updateMany({$and: [{"Frequency":"Monthly"},{"Readership":["Working Class"]},{"Annual\_Turnover":{$lt:100000}}]},{$set:{"Frequency":"Daily"},$currentDate:{lastModified:true}})*





*db.Newspaper.find({$and: [{"Frequency":"Daily"},{"Readership":["Working Class"]},{"Annual\_Turnover":{$lt:100000}}]},{\_id:1,"Name":1,"Frequency":1,"Readership":1,"Annual\_ Turnover":1})*



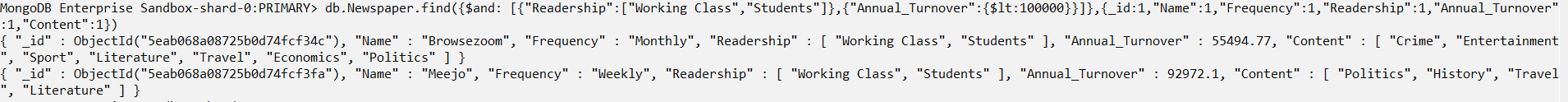


1. **Update the contents of the newspapers which are for the working class and students where the company annual turnover is less than 100000 euros.**

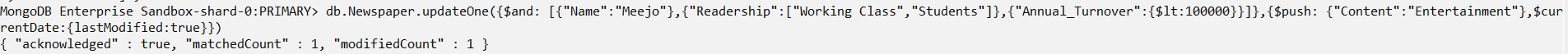
*db.Newspaper.find({$and: [{"Readership":["Working Class","Students"]},{"Annual\_Turnover":{$lt:100000}}]},{\_id:1,"Name":1,"Frequency":1,"Readership":1,"Annual\_Turnover":1,"Content":1}).count()*



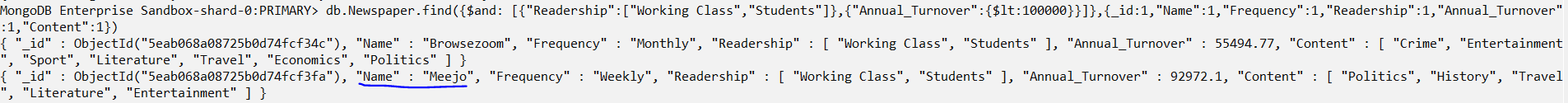
*db.Newspaper.find({$and: [{"Readership":["Working Class","Students"]},{"Annual\_Turnover":{$lt:100000}}]},{\_id:1,"Name":1,"Frequency":1,"Readership":1,"Annual\_Turnover":1,"Content":1})*



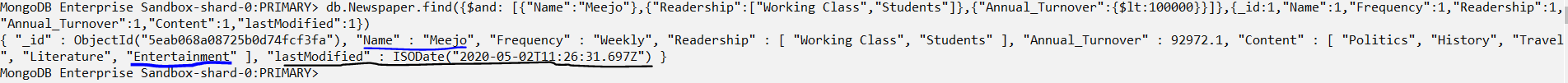
*db.Newspaper.updateOne({$and: [{"Name":"Meejo"},{"Readership":["Working Class","Students"]},{"Annual\_Turnover":{$lt:100000}}]},{$push: {"Content":"Entertainment"},$currentDate:{lastModified:true}})*



*db.Newspaper.find({$and: [{"Readership":["Working Class","Students"]},{"Annual\_Turnover":{$lt:100000}}]},{\_id:1,"Name":1,"Frequency":1,"Readership":1,"Annual\_Turnover":1,"Content":1})*

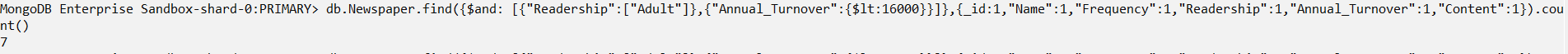


*db.Newspaper.find({$and: [{"Name":"Meejo"},{"Readership":["Working Class","Students"]},{"Annual\_Turnover":{$lt:100000}}]},{\_id:1,"Name":1,"Frequency":1,"Readership":1,"Annual\_Turnover":1,"Content":1})*

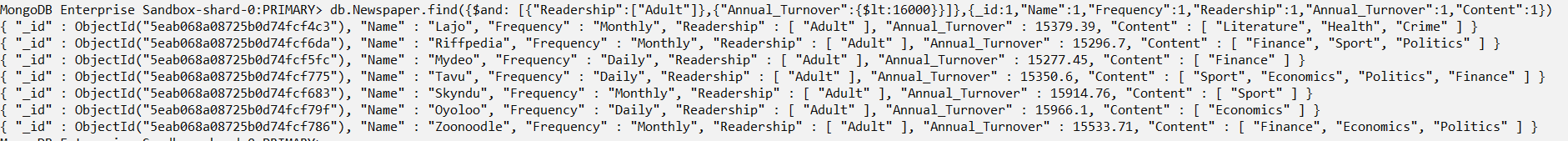


1. **Update the contents of the newspapers which are for the adults and the company annual turnover is less than 16000 euros.**

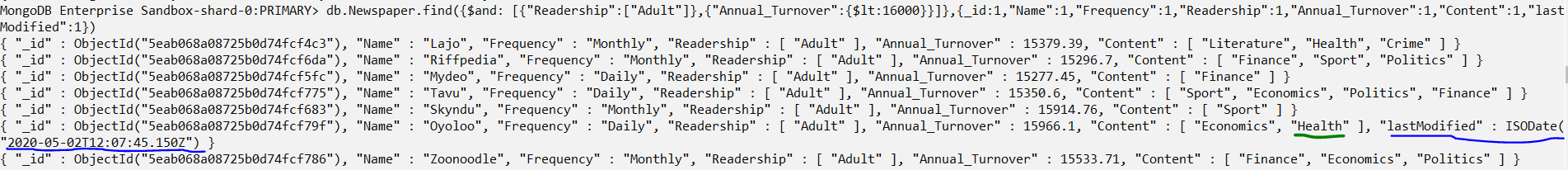
*db.Newspaper.find({$and:[{"Readership":["Adult"]},{"Annual\_Turnover":{$lt:16000}}]},{\_id:1,"Name":1,"Frequency":1,"Readership":1,"Annual\_Turnover":1,"Content":1}).count()*



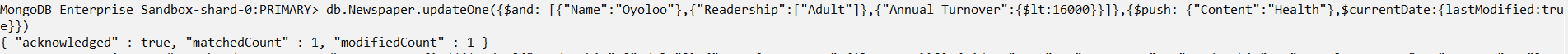
*db.Newspaper.find({$and:[{"Readership":["Adult"]},{"Annual\_Turnover":{$lt:16000}}]},{\_id:1,"Name":1,"Frequency":1,"Readership":1,"Annual\_Turnover":1,"Content":1})*



*db.Newspaper.updateOne({$and:[{"Name":"Oyoloo"},{"Readership":["Adult"]},{"Annual\_Turnover":{$lt:16000}}]},{$push: {"Content":"Health"},$currentDate:{lastModified:true}})*



*db.Newspaper.find({$and:[{"Readership":["Adult"]},{"Annual\_Turnover":{$lt:16000}}]},{\_id:1,"Name":1,"Frequency":1,"Readership":1,"Annual\_Turnover":1,"Content":1,"lastModified":1})*



1. **Update the format of the newspapers which are for Students, adults and retired (in that sequence) and the annual turnover of the company is greater than 70000 euros.**

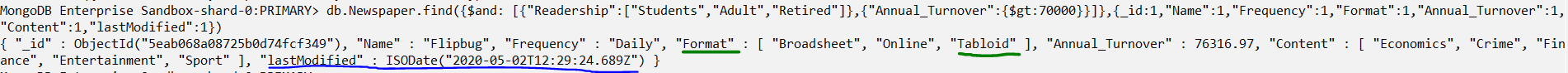
*db.Newspaper.find({$and:[{"Readership":["Students","Adult","Retired"]},{"Annual\_Turnover":{$gt:70000}}]},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Annual\_Turnover":1,"Content":1}).count()*



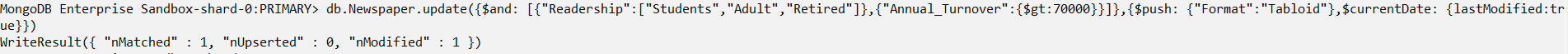
*db.Newspaper.find({$and:[{"Readership":["Students","Adult","Retired"]},{"Annual\_Turnover":{$gt:70000}}]},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Annual\_Turnover":1,"Content":1})*



*db.Newspaper.update({$and:[{"Readership":["Students","Adult","Retired"]},{"Annual\_Turnover":{$gt:70000}}]},{$push: {"Format":"Tabloid"},$currentDate: {lastModified:true}})*

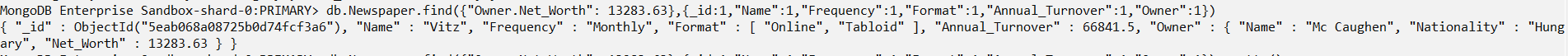


*db.Newspaper.find({$and:[{"Readership":["Students","Adult","Retired"]},{"Annual\_Turnover":{$gt:70000}}]},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Annual\_Turnover":1,"Content":1,"lastModified":1})*

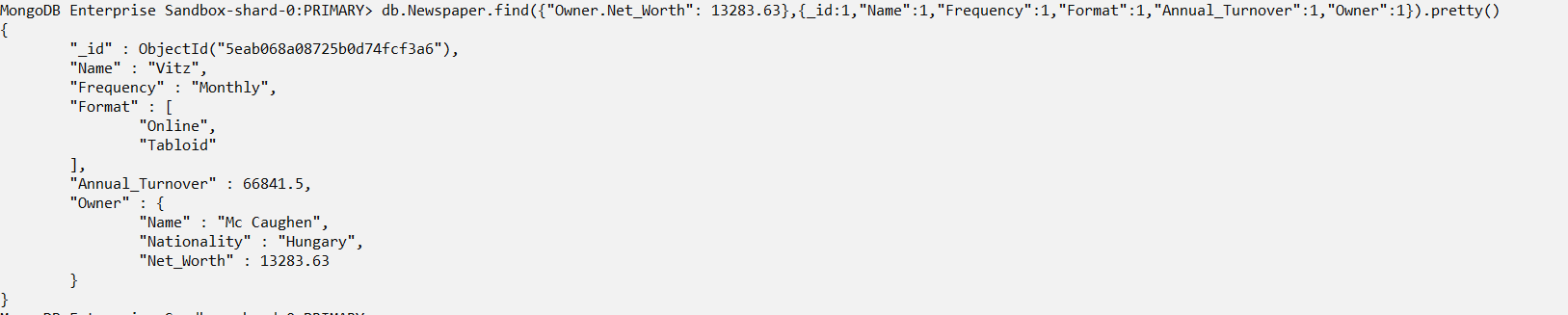


1. **Update the frequency of those whose net worth is equal to 13283,63.**

*db.Newspaper.find({"Owner.Net\_Worth":13283.63},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Annual\_Turnover":1,"Content":1,}).count()*



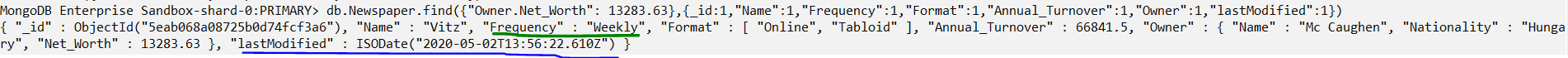
*db.Newspaper.find({"Owner.Net\_Worth":13283.63},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Annual\_Turnover":1,"Content":1,"Owner":1})*

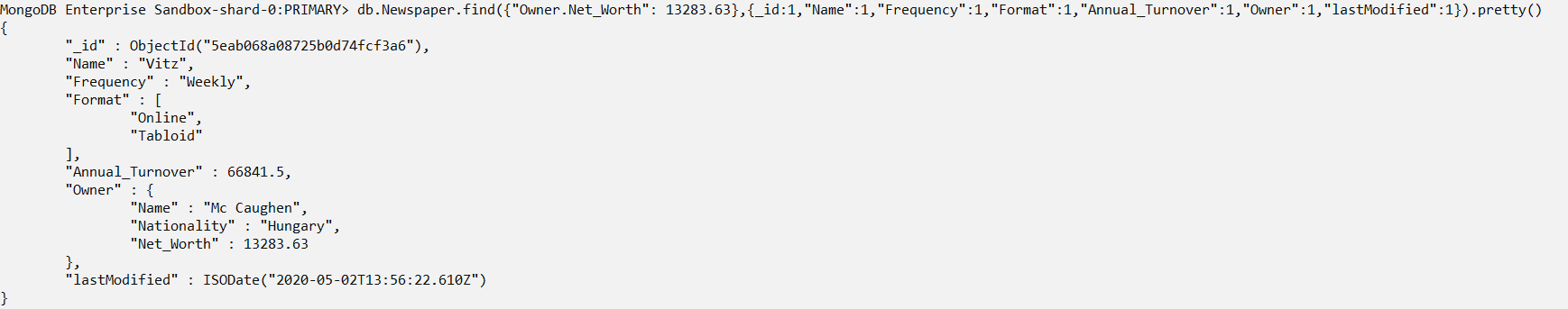


*db.Newspaper.update({"Owner.Net\_Worth":13283.63},{$set:{"Frequency":"Weekly"},$currentDate:{lastModified:true}})*



*db.Newspaper.find({"Owner.Net\_Worth":13283.63},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Annual\_Turnover":1,"Owner":1,"lastModified":1})*





1. **Sort the documents of the company founded after 2011 in the descending order of Annual turnover. The company must be delivering the newspapers monthly and the readers must be from either professional class or working class.**

*db.Newspaper.find({$and:[{"Year\_Founded":{$gt:2011}},{"Frequency":"Monthly"},{"Readership":{$in: ["Professional Class","Working Class"]}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"Readership":1}).sort({"Year\_Founded":-1}).count()*



*db.Newspaper.find({$and:[{"Year\_Founded":{$gt:2011}},{"Frequency":"Monthly"},{"Readership":{$in: ["Professional Class","Working Class"]}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"Readership":1}).sort({"Year\_Founded":-1})*

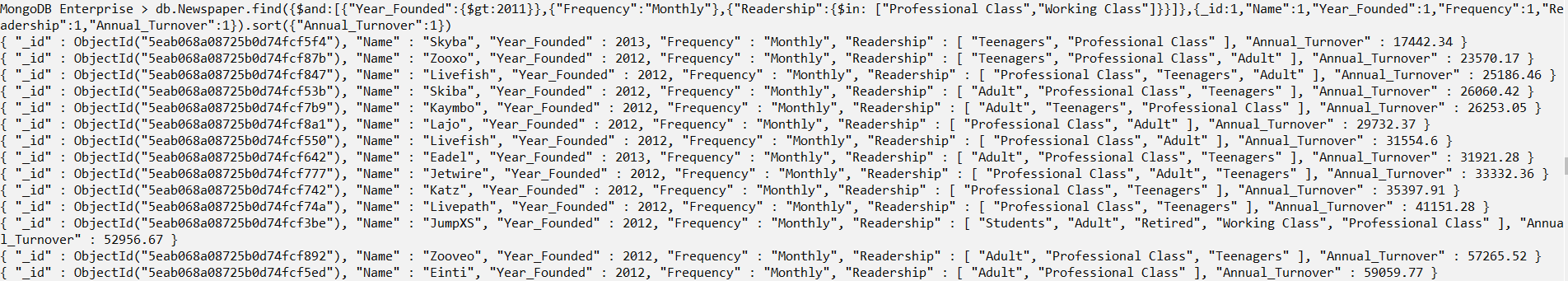


*db.Newspaper.find({$and:[{"Year\_Founded":{$gt:2011}},{"Frequency":"Monthly"},{"Readership":{$in: ["Professional Class","Working Class"]}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"Readership":1}).sort({"Annual\_Turnover":-1})*



1. **Sort the documents of the company founded after 2011 in the ascending order of Annual turnover. The company must be delivering the newspapers monthly and the readers must be from either professional class or working class.**

*db.Newspaper.find({$and:[{"Year\_Founded":{$gt:2011}},{"Frequency":"Monthly"},{"Readership":{$in: ["Professional Class","Working Class"]}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"Readership":1,"Annual\_Turnover":1}).sort({"Annual\_Turnover":1})*



1. **Sort the documents in the ascending order of owner’s net worth where the newspaper company is founded after 2011 and where the newspapers are delivered monthly. The readers must be either from professional class or working class.**



*db.Newspaper.find({$and:[{"Year\_Founded":{$gt:2011}},{"Frequency":"Monthly"},{"Readership":{$in: ["Professional Class","Working Class"]}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Owner":1}).sort({"Owner.Net\_Worth":1})*

1. **Sort the documents in descending order of owner’s net worth.**

*db.Newspaper.find({$and:[{"Year\_Founded":{$gt:2011}},{"Frequency":"Monthly"},{"Readership":{$in: ["Professional Class","Working Class"]}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Owner":1}).sort({"Owner.Net\_Worth":-1})*

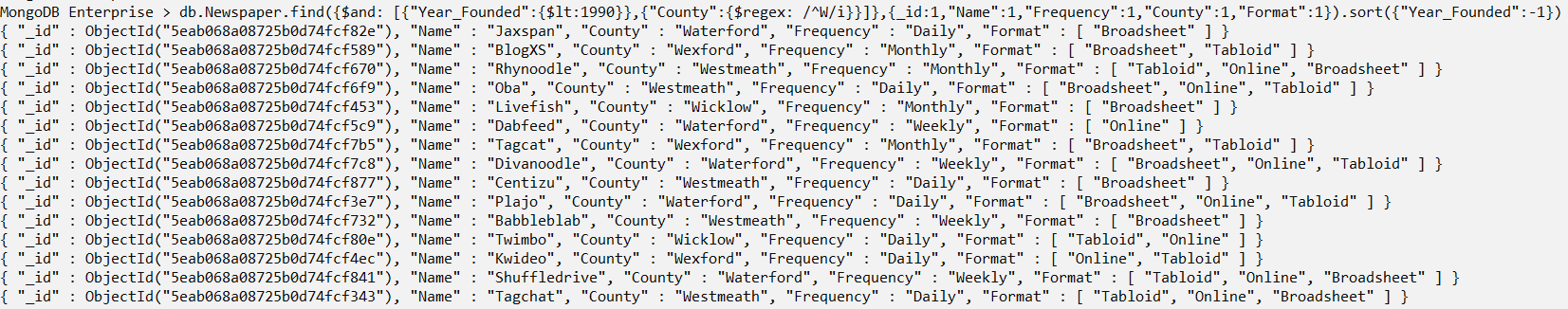


1. **Find all the documents in which the county name starts with the letter W. [Update Question]**

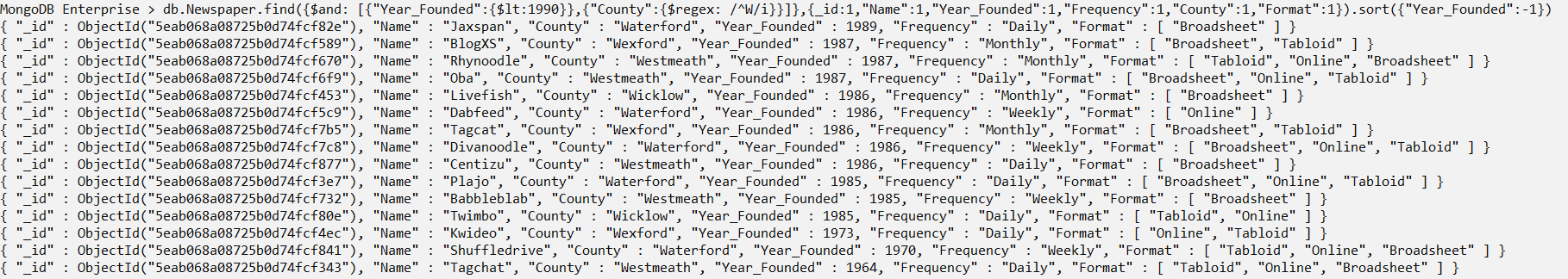
*db.Newspaper.find({$and: [{"Year\_Founded":{$lt:1990}},{"County":{$regex: /^W/i}}]},{\_id:1,"Name":1,"Frequency":1,"County":1,"Format":1}).sort({"Year\_Founded":-1}).count()*



*db.Newspaper.find({$and: [{"Year\_Founded":{$lt:1990}},{"County":{$regex: /^W/i}}]},{\_id:1,"Name":1,"Frequency":1,"County":1,"Format":1}).sort({"Year\_Founded":-1})*



*db.Newspaper.find({$and: [{"Year\_Founded":{$lt:1990}},{"County":{$regex: /^W/i}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"County":1,"Format":1}).sort({"Year\_Founded":-1})*

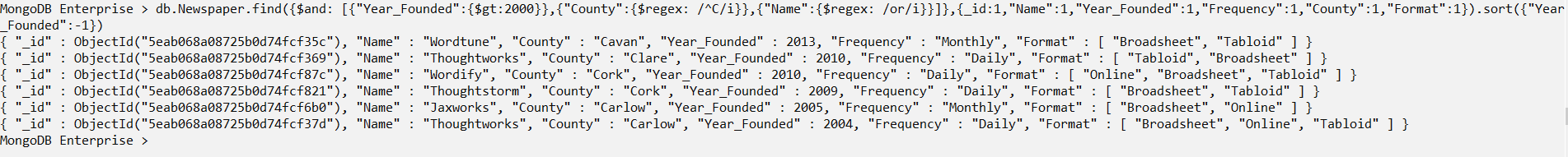


1. **Find all the documents for the newspaper companies found after 2000 in which the county name starts with the letter C, and the name of the newspaper has “or” in it.**

*db.Newspaper.find({$and: [{"Year\_Founded":{$gt:2000}},{"County":{$regex: /^C/i}},{"Name":{$regex: /or/i}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"County":1,"Format":1}).sort({"Year\_Founded":-1}).count()*

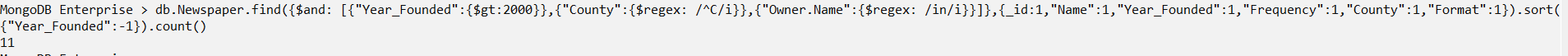


*db.Newspaper.find({$and: [{"Year\_Founded":{$gt:2000}},{"County":{$regex: /^C/i}},{"Name":{$regex: /or/i}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"County":1,"Format":1}).sort({"Year\_Founded":-1})*



1. **Find the documents in which the name of the newspaper starts with the letter R and the newspaper owner name consists of “in” in it, and is founded before the year 2000.**

*db.Newspaper.find({$and: [{"Year\_Founded":{$gt:2000}},{"County":{$regex: /^R/i}},{"Owner.Name":{$regex: /in/i}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"County":1,"Format":1}).sort({"Year\_Founded":-1}).count()*



*db.Newspaper.find({$and: [{"Year\_Founded":{$gt:2000}},{"County":{$regex: /^R/i}},{"Owner.Name":{$regex: /in/i}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"County":1,"Format":1}).sort({"Year\_Founded":-1})*



1. **Find the documents of the company founded after 2010 whose owner is from France and name of the owner starts with the letter “B”.**

*db.Newspaper.find({$and: [{"Year\_Founded":{$gt:2010}},{"Owner.Nationality":"France"},{"Owner.Name":{$regex: /^B/i}}]},{\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"County":1,"Owner":1}).sort({"Year\_Founded":-1}).count()*



*db.Newspaper.find({$and: [{"Year\_Founded":{$gt:2010}},{"Owner.Nationality":"France"},{"Owner.Name":{$regex: /^B/i}}]},{\_id:1,"Name":1,"Year\_Founded":1,"County":1,"Owner":1}).sort({"Year\_Founded":-1})*

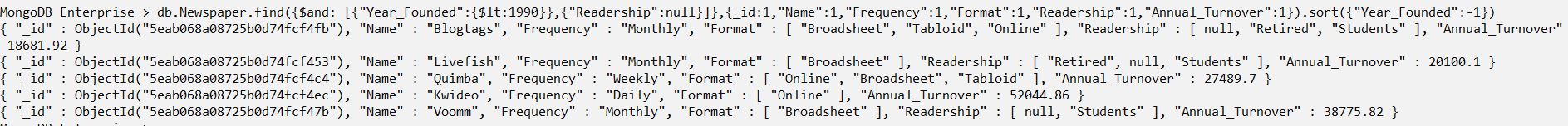


1. **Update the format information of the documents to “Tabloid” for the ones which have not specified the readership, where the newspaper companies found before 1990.**

*db.Newspaper.find({$and: [{"Year\_Founded":{$lt:1990}},{"Readership":null}]},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Readership":1,"Annual\_Turnover":1}).sort({"Year\_Founded":-1}).count()*



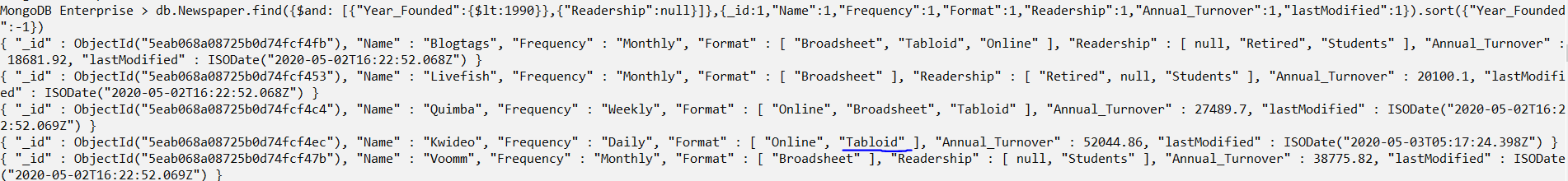
*db.Newspaper.find({$and: [{"Year\_Founded":{$lt:1990}},{"Readership":null}]},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Readership":1,"Annual\_Turnover":1}).sort({"Year\_Founded":-1})*



*db.Newspaper.update({$and: [{"Year\_Founded":{$lt:1990}},{"Readership":[null]}]},{$addToSet :{"Format":"Tabloid"},$currentDate:{lastModified:true}})*

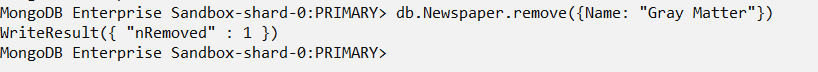


*db.Newspaper.find({$and: [{"Year\_Founded":{$lt:1990}},{"Readership":null}]},{\_id:1,"Name":1,"Frequency":1,"Format":1,"Readership":1,"Annual\_Turnover":1,"lastModified":1}).sort({"Year\_Founded":-1})*



1. **Remove the Document of the company Gray Matter.**

db.Newspaper.remove({Name: "Gray Matter"})

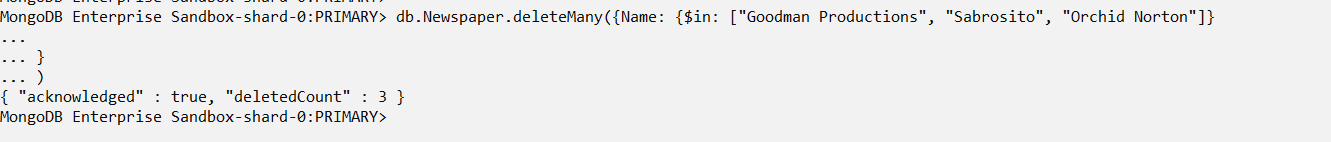


*db.Newspaper.find({Name: "Gray Matter"}).count()*



1. **Remove all the newly inserted documents.**

*db.Newspaper.deleteMany({Name: {$in: ["Goodman Productions", "Sabrosito", "Orchid Norton"]}})*



*db.Newspaper.find({Name: {$in: ["Goodman Productions", "Sabrosito", "Orchid Norton"]} }).count()*



# Aggregate Pipeline Queries

1. **Return the documents where the name of the Newspaper,whose owner’s net worth is between 3000 to 4000, and delivered daily in counties Leitrim, Meath, Kildare, Antrim, or Wexford. Display the name of the newspaper, the counties, readership details and the owner details.**

db.Newspaper.aggregate(

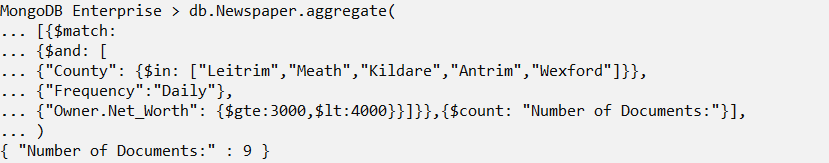
[{$match:

{$and: [

{"County": {$in: ["Leitrim","Meath","Kildare","Antrim","Wexford"]}},

{"Frequency":"Daily"},

{"Owner.Net\_Worth": {$gte:3000,$lt:4000}}]}},{$count: "Number of Documents:"}])



db.Newspaper.aggregate(

{$match:

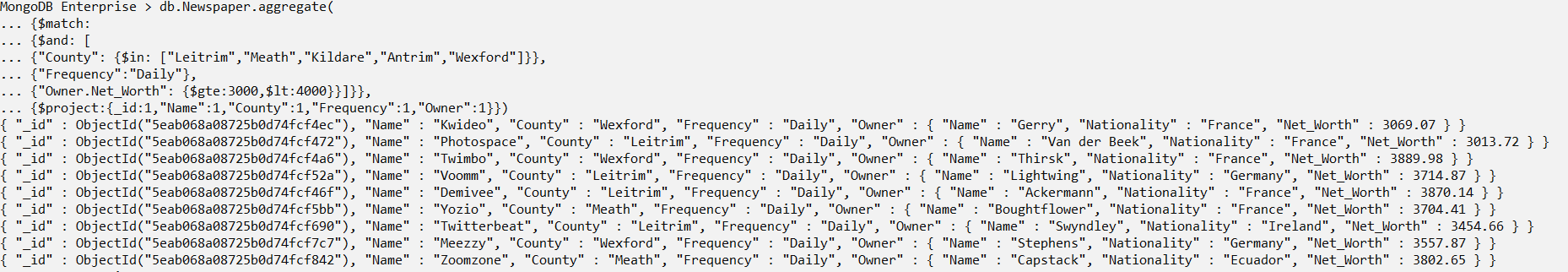
{$and: [

{"County": {$in: ["Leitrim","Meath","Kildare","Antrim","Wexford"]}},

{"Frequency":"Daily"},

{"Owner.Net\_Worth": {$gte:3000,$lt:4000}}]}}

{$project:{\_id:1,"Name":1,"County":1,"Frequency":1,"Owner":1}})



1. **Return the documents where the name of the Newspaper,whose owner’s net worth is between 3000 to 4000, starts with “Y” and is delivered daily in counties Leitrim, Meath, Kildare, Antrim, or Wexford. Display the name of the newspaper, the counties, readership details and the owner details. The output must each have different readership elements.**

db.Newspaper.aggregate(

{$match:

{$and: [

{"Name":{$regex: /^Y/i }},

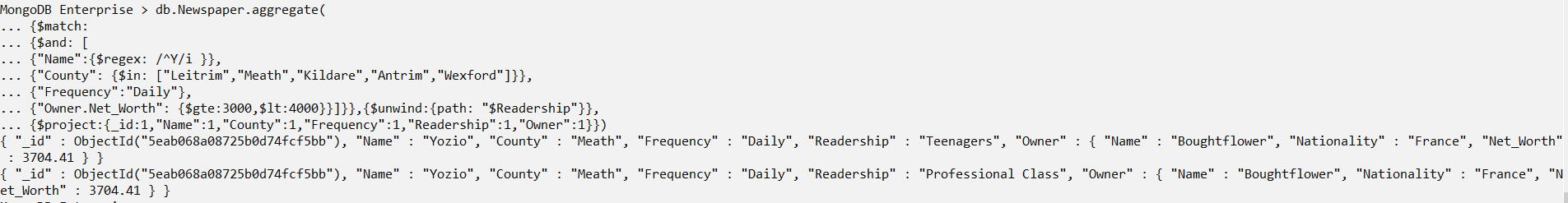
{"County": {$in: ["Leitrim","Meath","Kildare","Antrim","Wexford"]}},

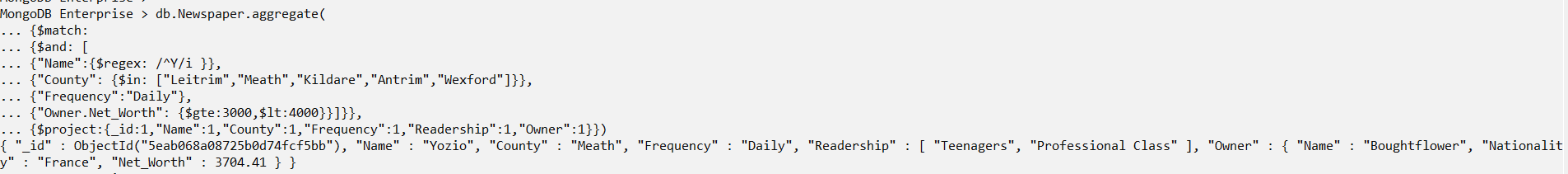
{"Frequency":"Daily"},

{"Owner.Net\_Worth": {$gte:3000,$lt:4000}}]}},

{$unwind:{path: "$Readership"}},

{$project:{\_id:1,"Name":1,"County":1,"Frequency":1,"Readership":1,"Owner":1}})





1. **Return the details of newspaper founded between the year 2002 and 2005 belonging to counties Leitrim, Meath, Kildare, Antrim or Wexford, for which the readership details has unspecified fields in them.**

db.Newspaper.aggregate(

{$match:

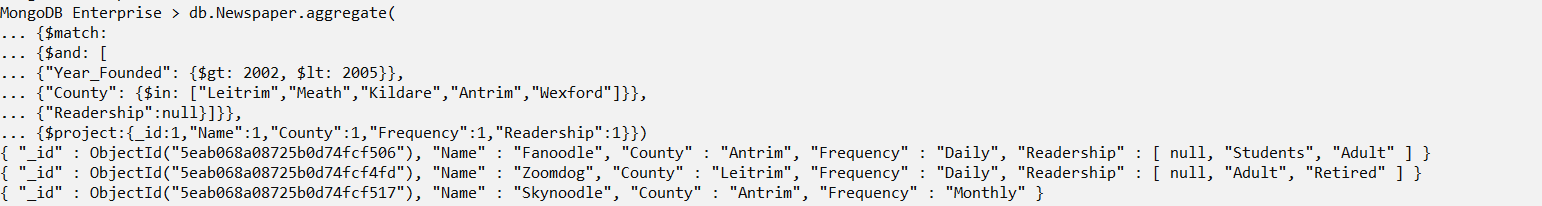
{$and: [

{"Year\_Founded": {$gt: 2002, $lt: 2005}},

{"County": {$in: ["Leitrim","Meath","Kildare","Antrim","Wexford"]}},

{"Readership":null}]}},

{$count: "Number Of Documents: "})



1. **Return the details of the newspaper founded between 2002 and 2005 in counties of Leitrim, Meath, Kildare, Antrim, or Wexford**

db.Newspaper.aggregate(

{$match:

{$and: [

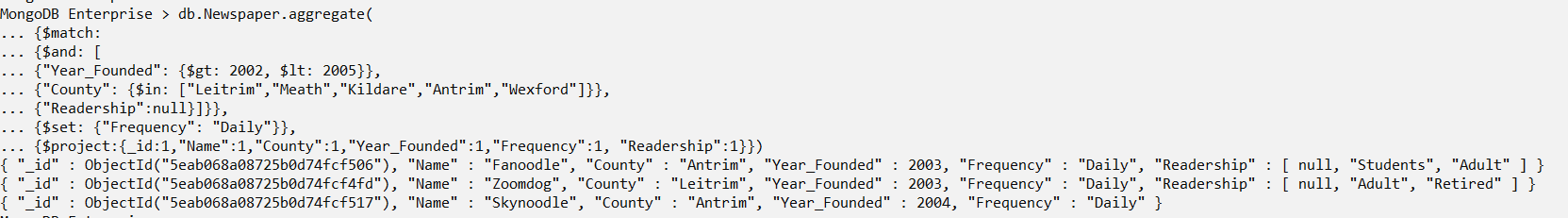
{"Year\_Founded": {$gt: 2002, $lt: 2005}},

{"County": {$in: ["Leitrim","Meath","Kildare","Antrim","Wexford"]}},

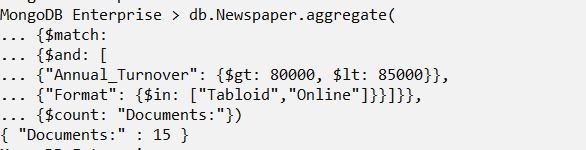
{"Readership":null}]}},

{$set: {"Frequency": "Daily"},

$currentDate: {lastModified:true}})



1. **Return the documents where the annual turnover of the newspaper company is between 80000 and 85000 with a format of either Tabloid or Online.**



db.Newspaper.aggregate(

{$match:

{$and:

[ {"Annual\_Turnover": {$gt: 80000, $lt: 85000}},

{"Format": {$in: ["Tabloid","Online"]}}]}}, {$project:{\_id:1,"Name":1,"County":1,"Year\_Founded":1,"Frequency":1, "Format":1}})



db.Newspaper.aggregate(

{$match:

{$and:

[ {"Annual\_Turnover": {$gt: 80000, $lt: 85000}},

{"Format": {$in: ["Tabloid","Online"]}}]}},

{$set: {"Frequency": "Daily"}},

{$project:{\_id:1,"Name":1,"County":1,"Year\_Founded":1,"Frequency":1, "Format":1}})



1. **Return the details of Taboid or Online paper companies with Annual Turnover between 40000 and 95000 euros sorted out in descending order**

db.Newspaper.aggregate(

{$match:

{$and:

[{"Annual\_Turnover": {$gt: 40000, $lt: 95000}},

{"Format": ["Tabloid","Online"]}]}}, {$project:{\_id:1,"Name":1,"County":1,"Annual\_Turnover":1,"Frequency":1, "Format":1}},

{$sort: {"Annual\_Turnover":-1}})



1. **Display the number of documents presented by each Newspaper company, founded between 1990 and 1995, and the total hardcopy amount of those companies.**

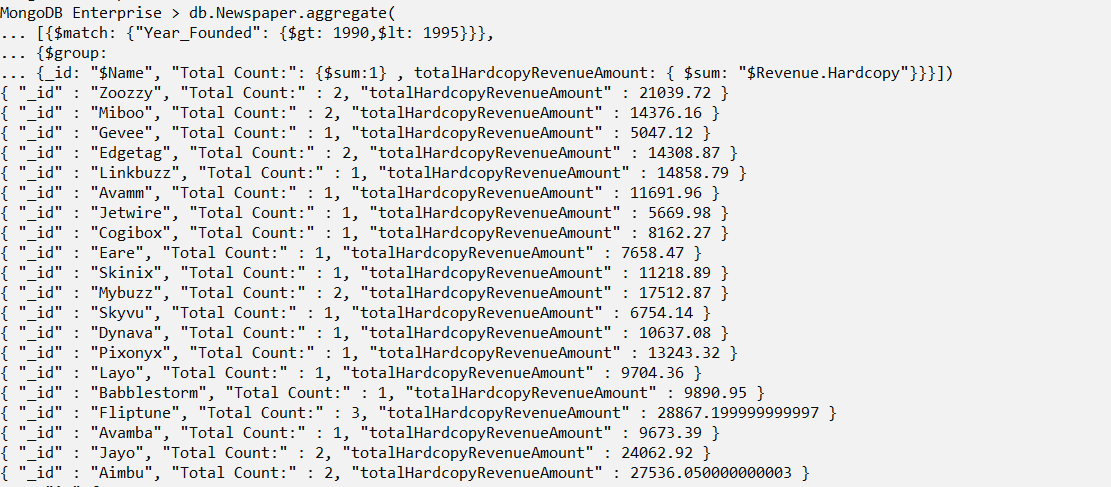
db.Newspaper.aggregate(

[{$match:

{"Year\_Founded": {$gt: 1990,$lt: 1995}}},

{$group: {\_id: "$Name", "Total Count:": {$sum:1} ,

totalHardcopyRevenueAmount: { $sum: "Revenue.Hardcopy"}}}])



1. **Display the average of the owner’s net worth from each nationality who owns a newspaper company founded between 1990 and 2000.**

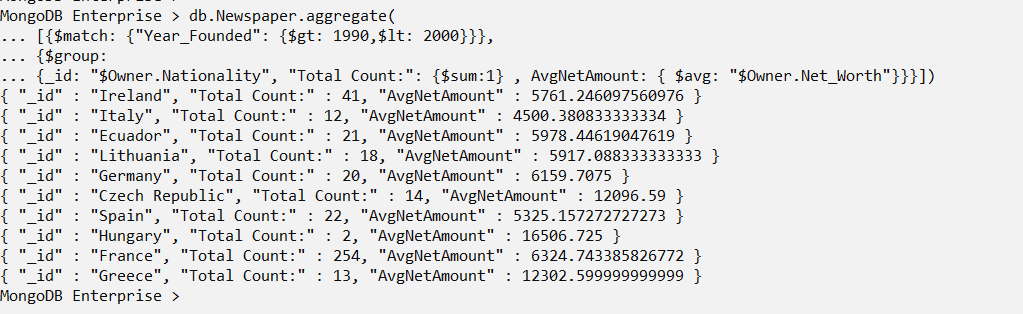
db.Newspaper.aggregate(

[{$match:

{"Year\_Founded": {$gt: 1990,$lt: 2000}}},

{$group: {\_id: "$Owner.Nationality", "Total Count:": {$sum:1} ,

AvgNetAmount: { $avg: "$Owner.Net\_Worth"}}}])



1. **Return the total net worth of the owners from each country greater than 30000 euros sorted out in descending order.**

db.Newspaper.aggregate(

[{$group:

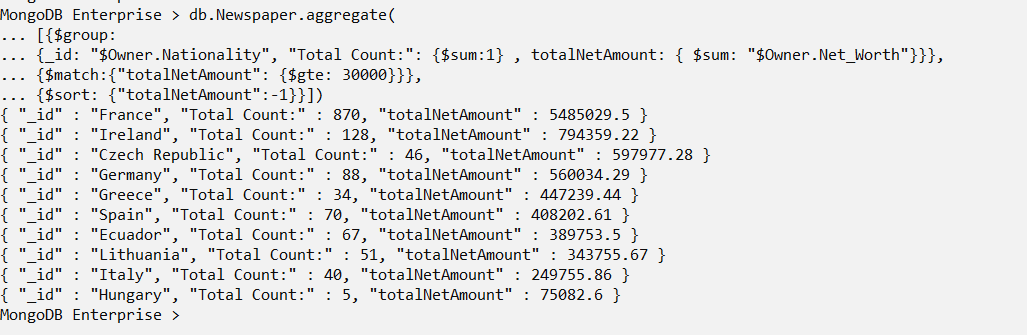
{\_id: "$Owner.Nationality",

"Total Count:": {$sum:1} ,

totalNetAmount: { $sum: "$Owner.Net\_Worth"}}},

{$match:{"totalNetAmount": {$gte: 30000}}},

{$sort: {"totalNetAmount":-1}}])



1. **Return the Annual turnover of the Newspaper company from each town, which is greater than 30000 euros, sorted out in descending order.**

db.Newspaper.aggregate(

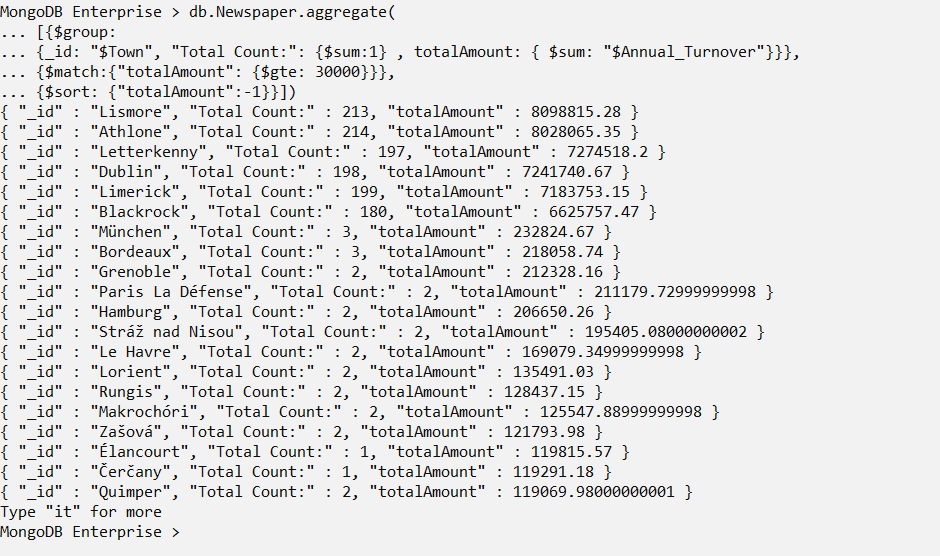
[{$group: {\_id: "$Town",

"Total Count:": {$sum:1} ,

totalAmount: { $sum: "$Annual\_Turnover"}}},

{$match:{"totalAmount": {$gte: 30000}}},

{$sort: {"totalAmount":-1}}])



# Compass and Atlas functions

1. **Return the documents whose annual turnover is less than 15500 euros.**

* **Enter the query in the filter option:** *{"Annual\_Turnover":{$lt:15500}}*

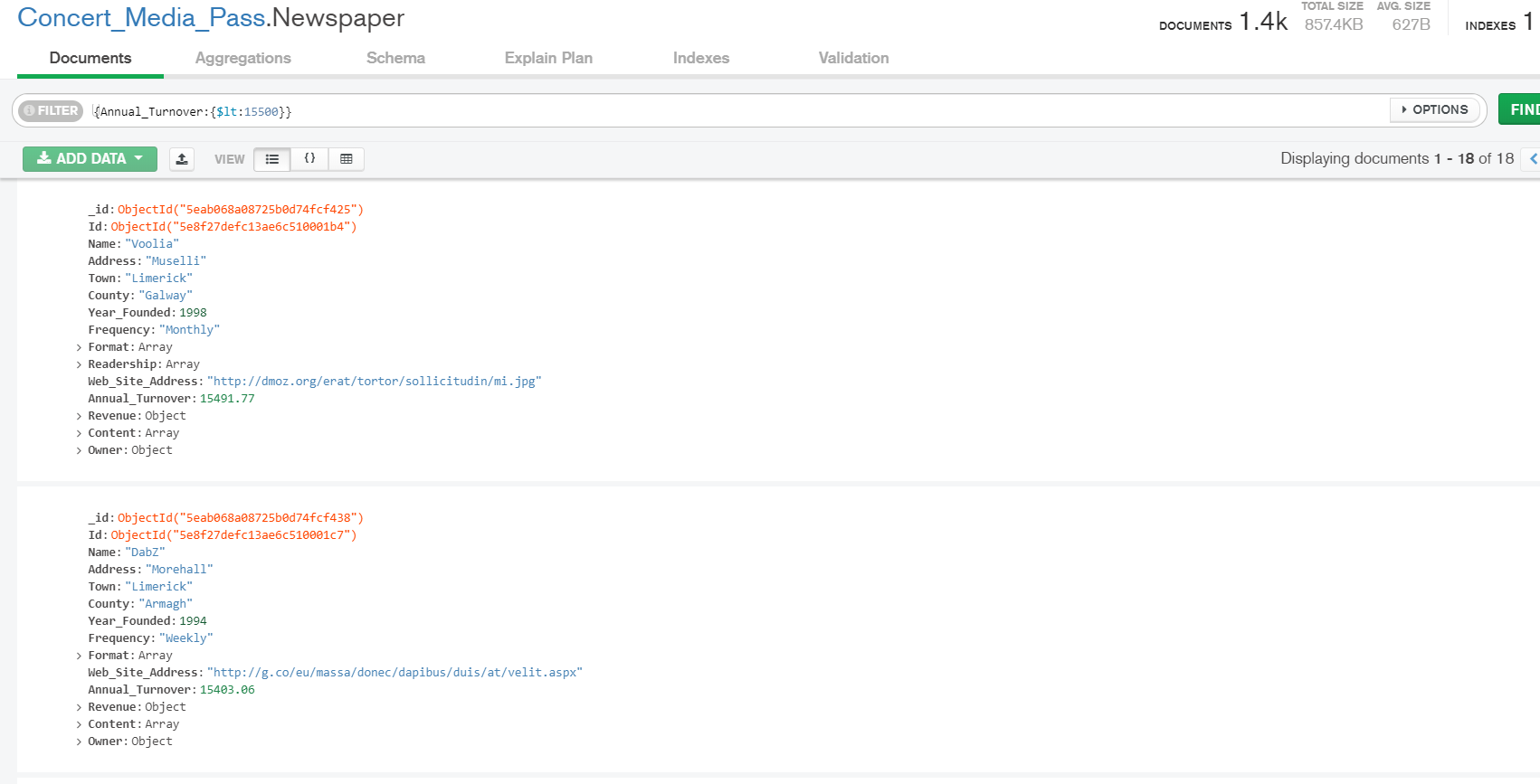


Figure : Using filter on Annual Turnover

* Click on the **Schema** View for visual representation of the output.

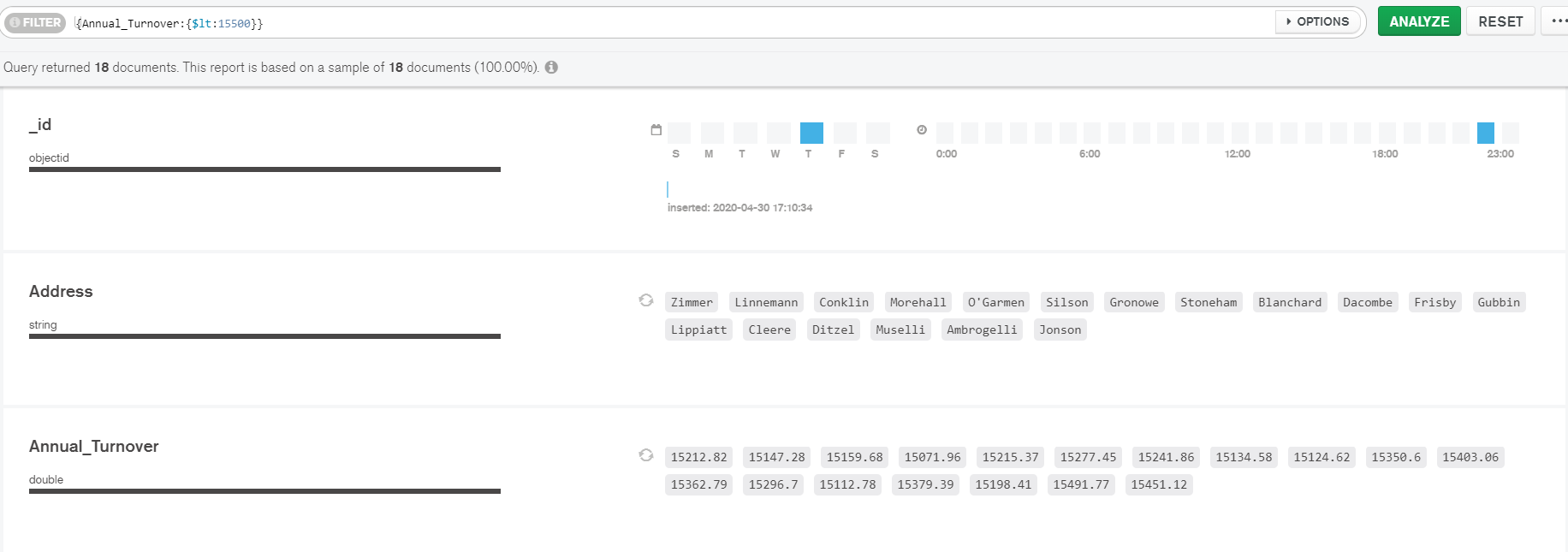


Figure : Schema view of the output

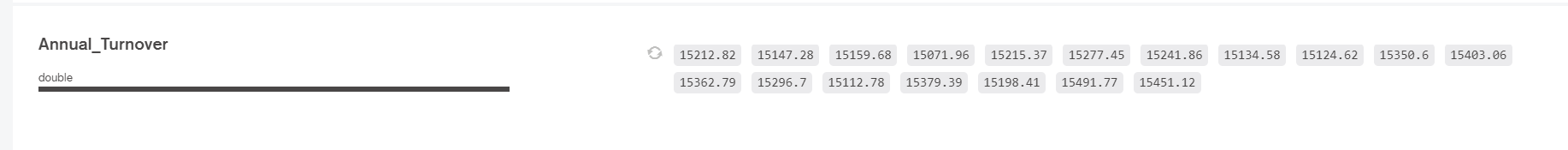


Figure : Annual Turnover filter

1. **Find the number of documents which is for adults, retired, professional class and students in any sequence and also return the documents details. (Corresponding Question 7 From Part-II)**

* **Enter the query in the filter path option:** *{Readership:{$all: ["Adult","Retired","Professional Class","Students"]}})*

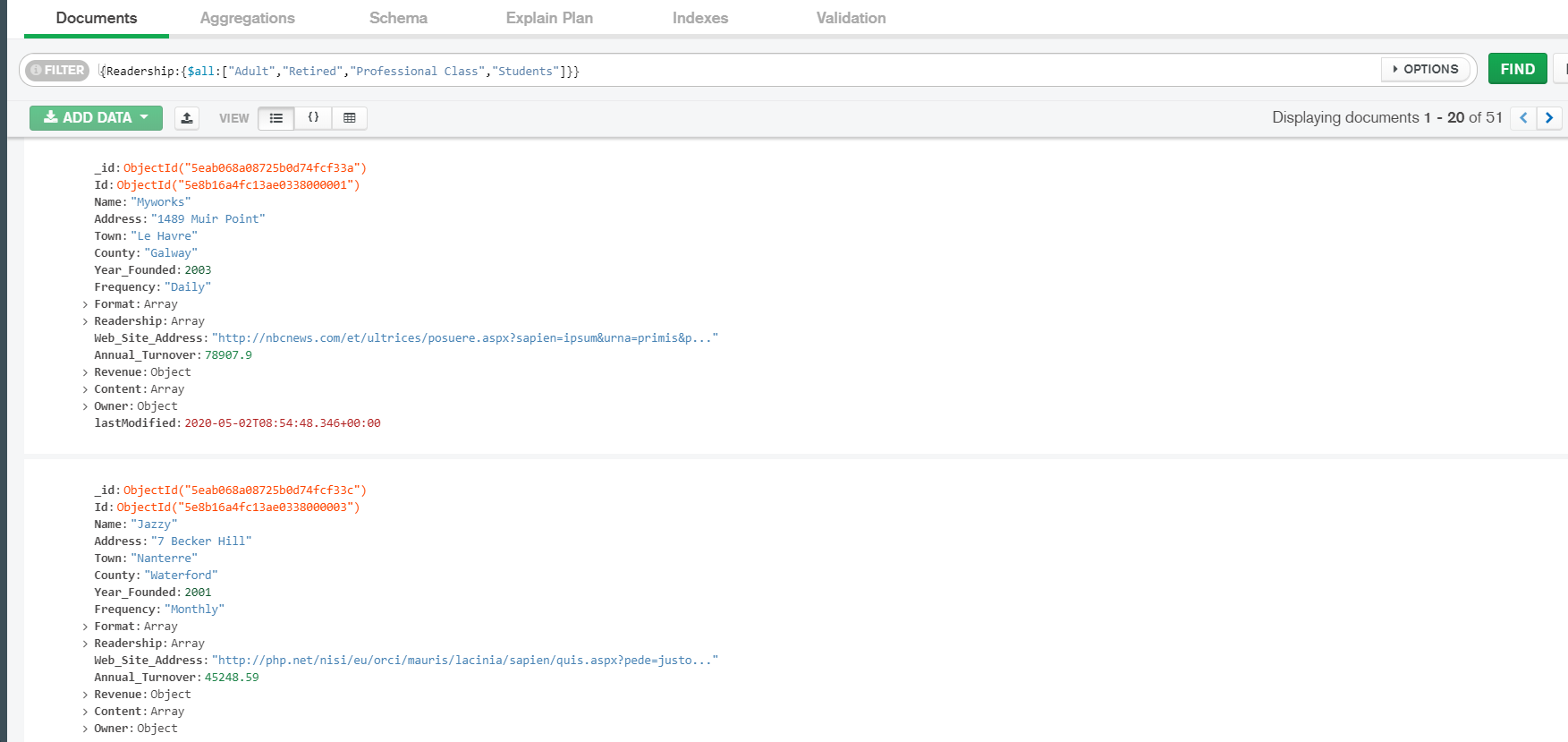


Figure : Applying filter on document

* Click on the **Schema** view to display the output visually.

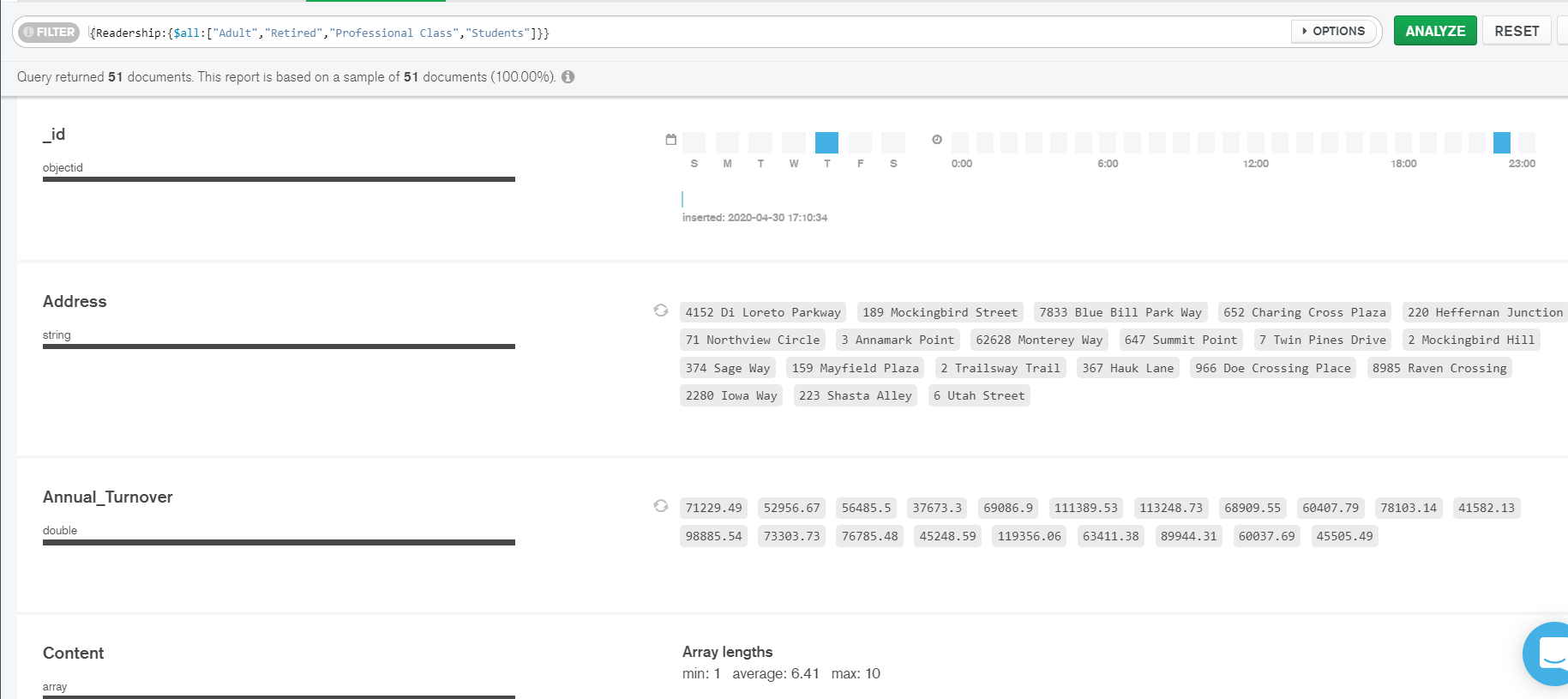


Figure : Schema View of the output

1. **Return the owner details of those whose net worth is greater than 10000 euros but less than 15000 euros and are from France.**

* **Query:** *{"Owner.Net\_Worth":{$gt:10000,$lt:15000}},{"Owner.Nationality":"France"}*

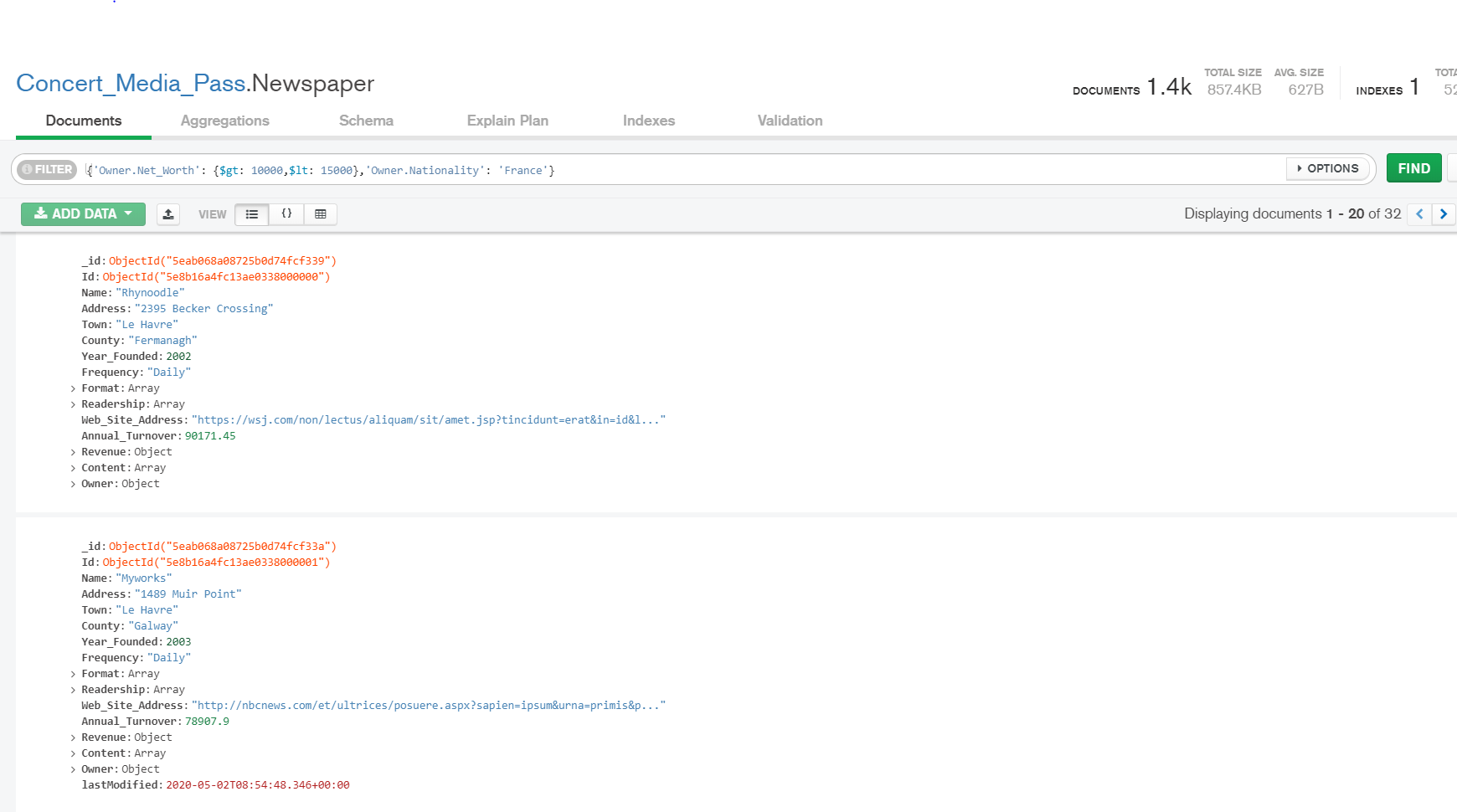


Figure : Applying filter on the collection

* Click on the **Schema** View

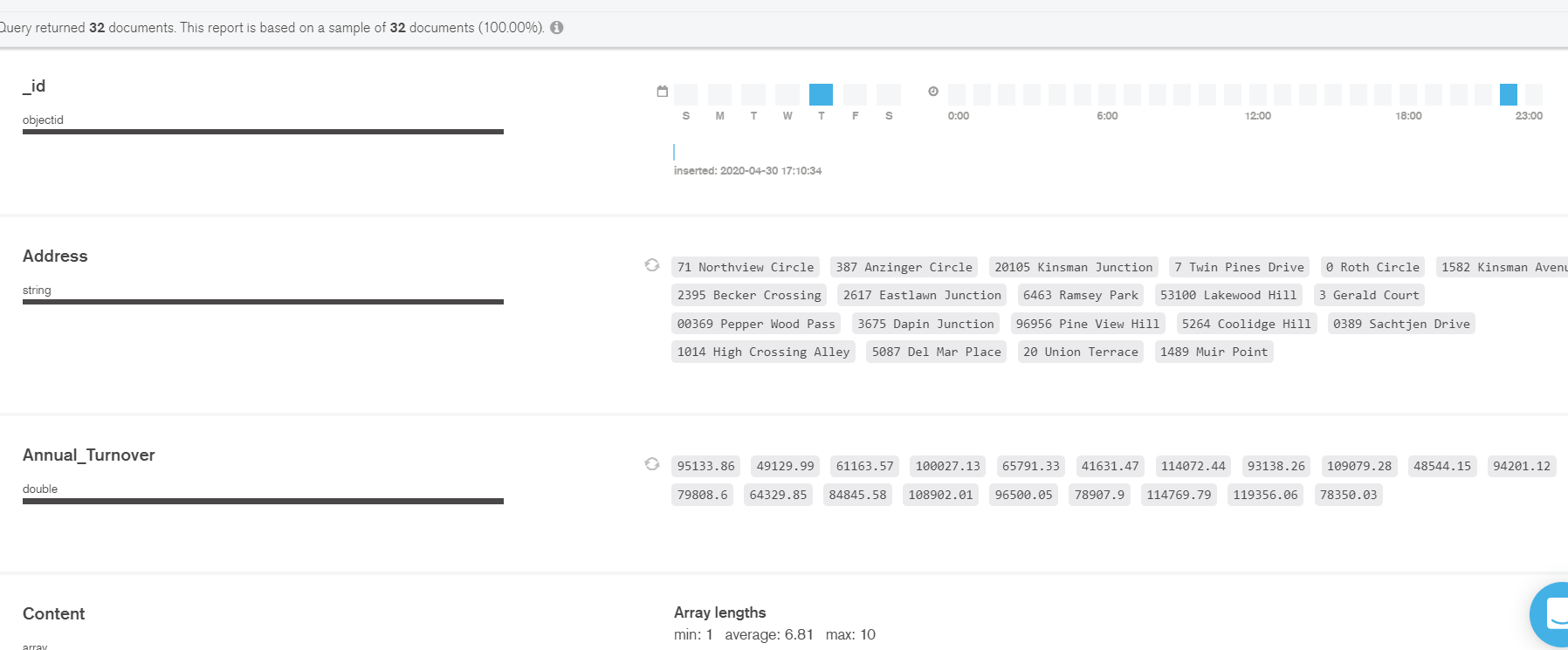


Figure : Schema View of the output

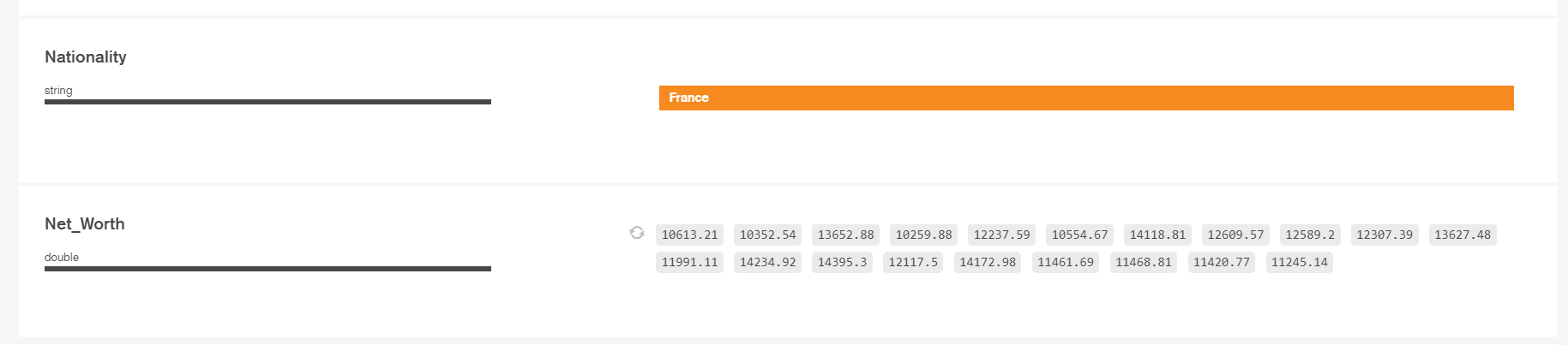


Figure : The Schema view of Nationality and Net\_Worth

1. **Sort the documents of the company founded after 2011 in the ascending order of Annual turnover. The company must be delivering the newspapers monthly and the readers must be from either professional class or working class.**

* Enter the Query:

***Filter***: *{"Year\_Founded":{$gt:2011}},{"Frequency":"Monthly"},{"Readership":{$in: ["Professional Class","Working Class"]}}*

***Project****: {\_id:1,"Name":1,"Year\_Founded":1,"Frequency":1,"Readership":1}*

***Sort****:{"Annual\_Turnover":-1}*

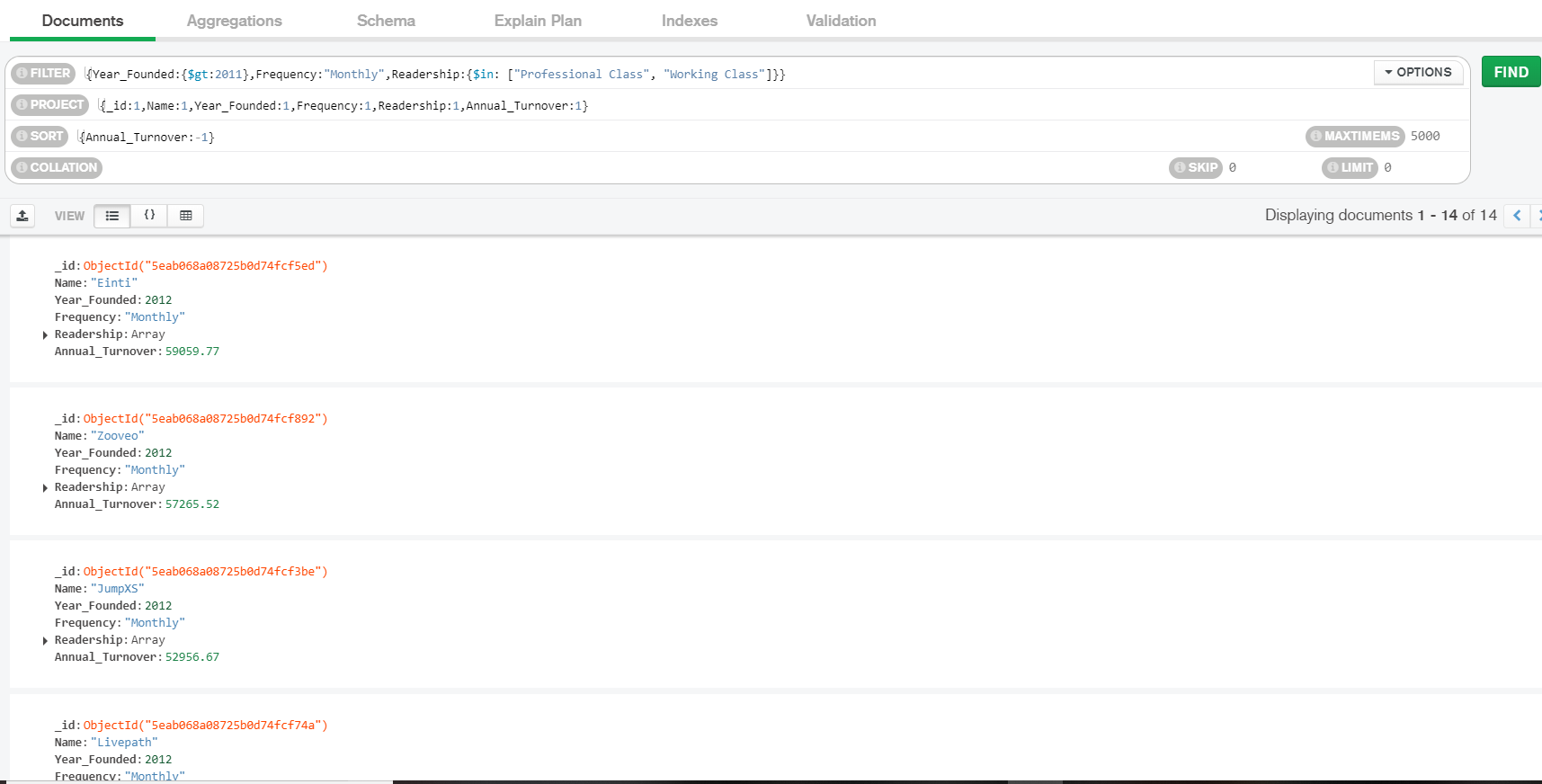


Figure : Applying multiple options for filter

* Click on the **Schema** view for visualizing the output.

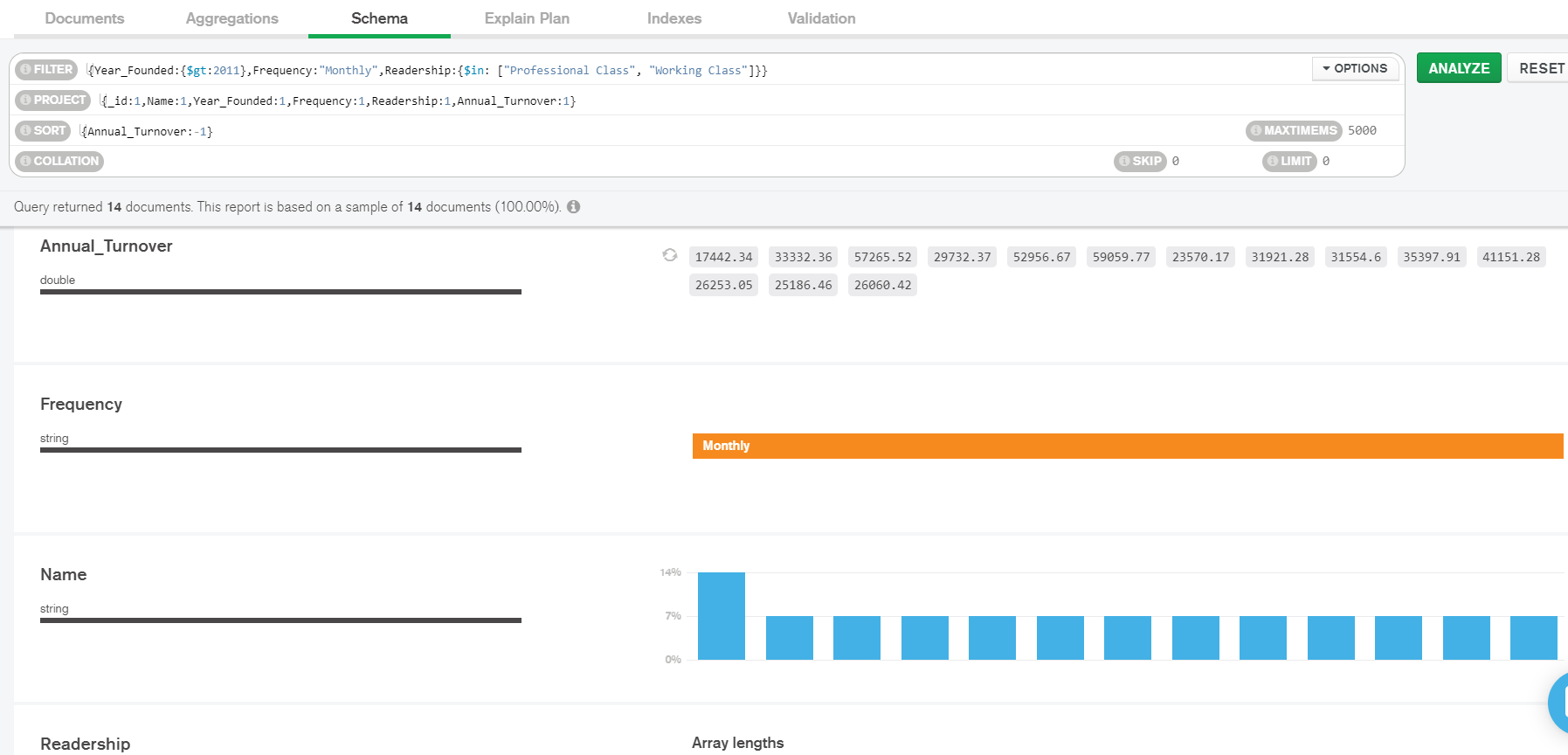


Figure : Schema view of the Output



Figure : Schema View of Frequency

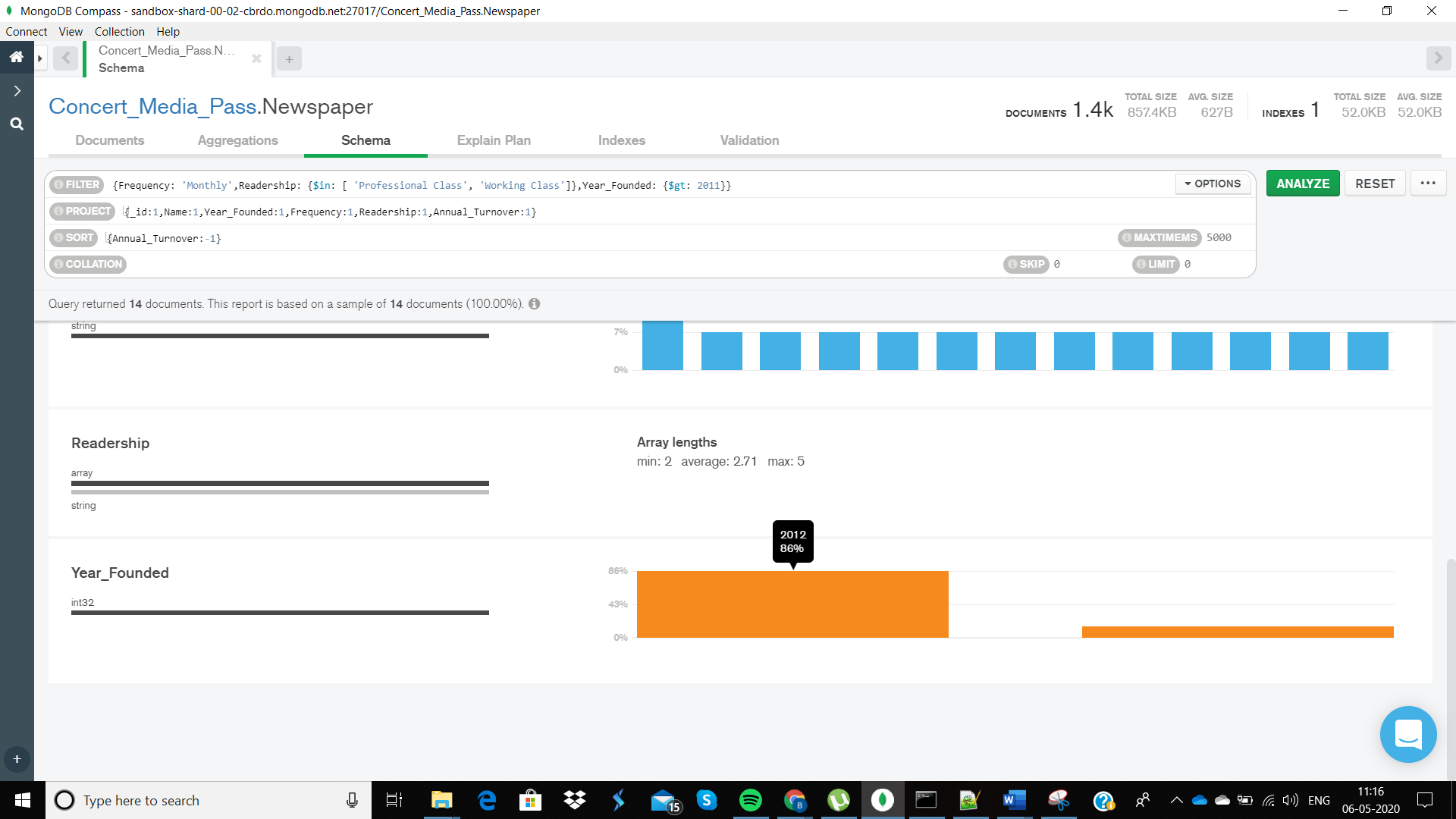


Figure : Highlight the years in the output document

1. **Return the details of paper companies with Annual Turnover between 40000 and 95000 euros.**

* Click on the Aggregation view option.
* In the field box, select the preferred stage and enter the query for that particular stage.

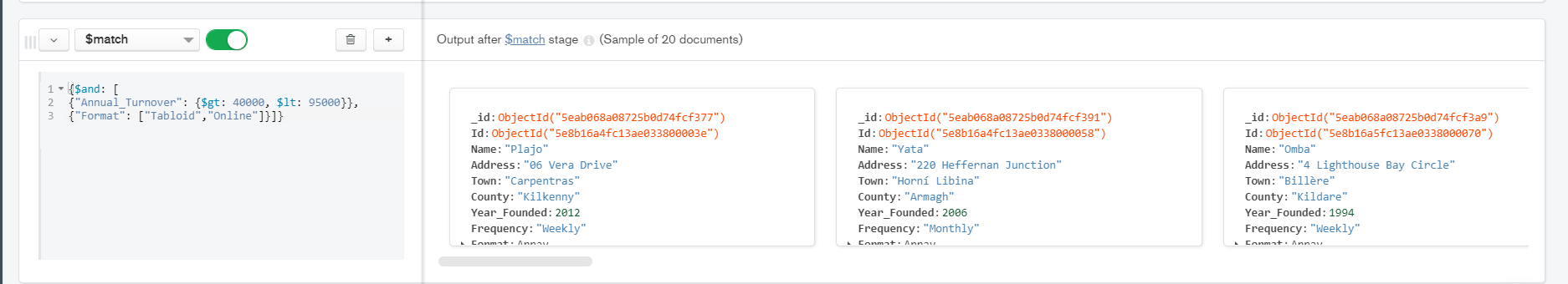


Figure : Aggregation pipeline field box with the respective filter query

* The output will be automatically generated.



Figure : Output document detail using Aggregation Pipeline

1. **Return the total net worth of the owners from each country greater than 30000 euros sorted out in descending order.**

* **Click on the Aggregation view and enter the query according to stages.**

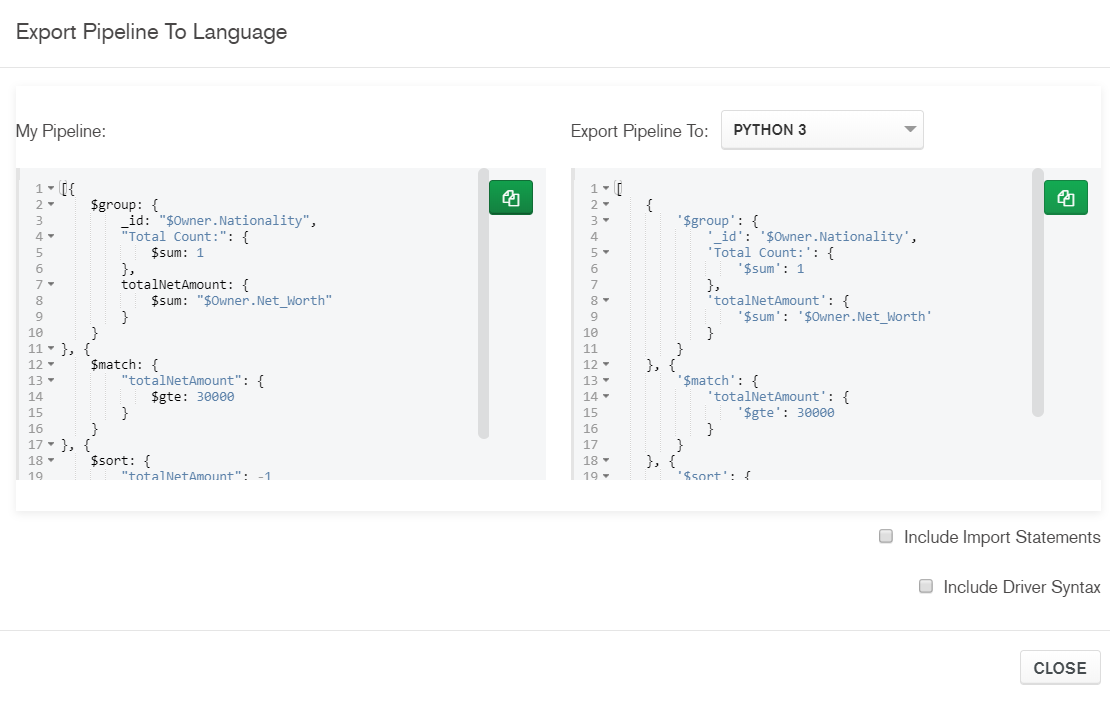


Figure : Pipeline entire view

* For three stages, enter the conditions in different field boxes as shown below. There is an option to add more stages after one.







Figure : Seperate field box for different Aggregate Stages

# Youtube Video Links

## MongoDB Aggregation part -1

Link: <https://youtu.be/de9BjennORI>

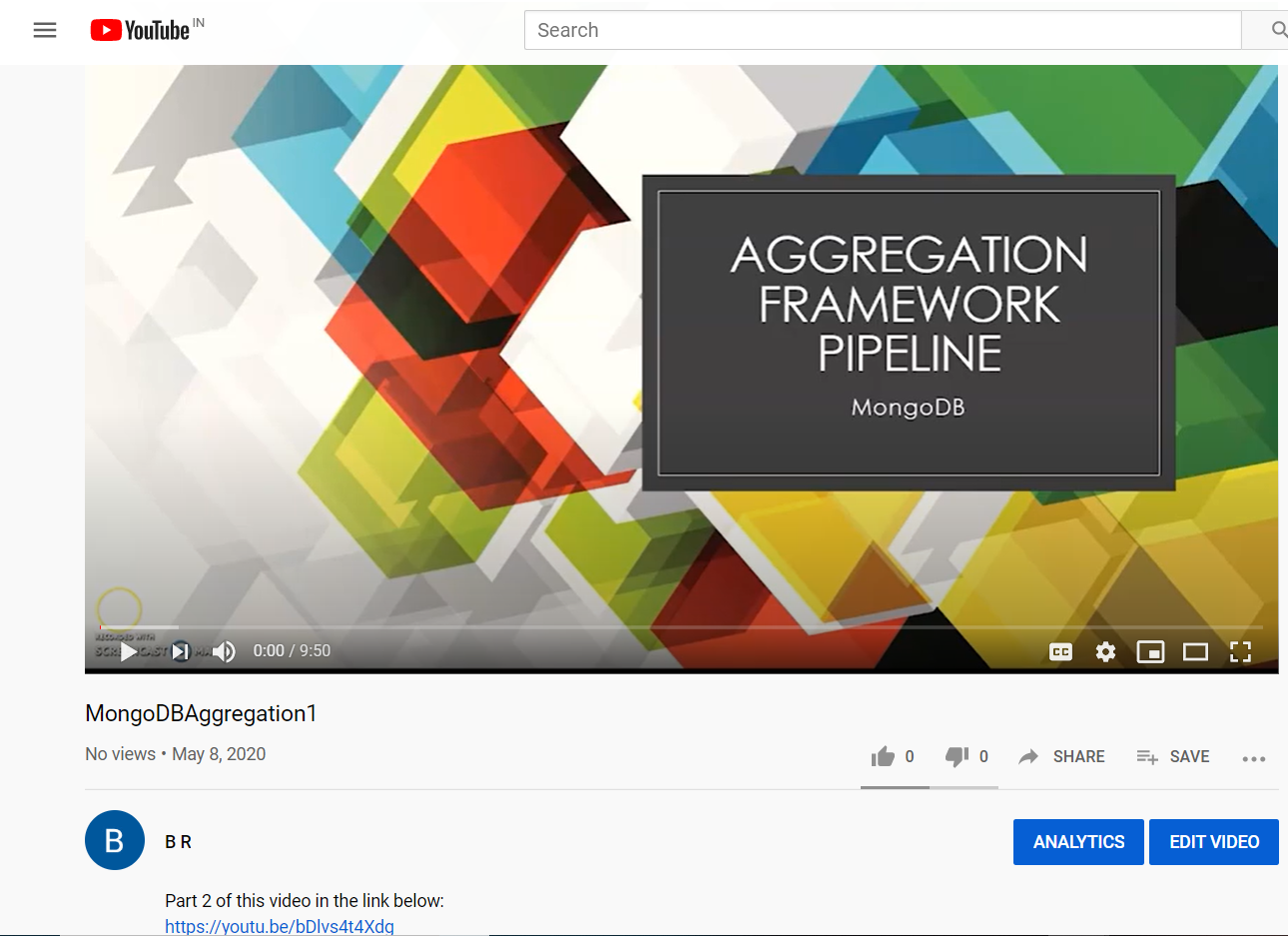


Figure : Youtube Video of Aggregation Part-1

## MongoDB Aggregation part-2

Link: <https://youtu.be/bDlvs4t4Xdg>

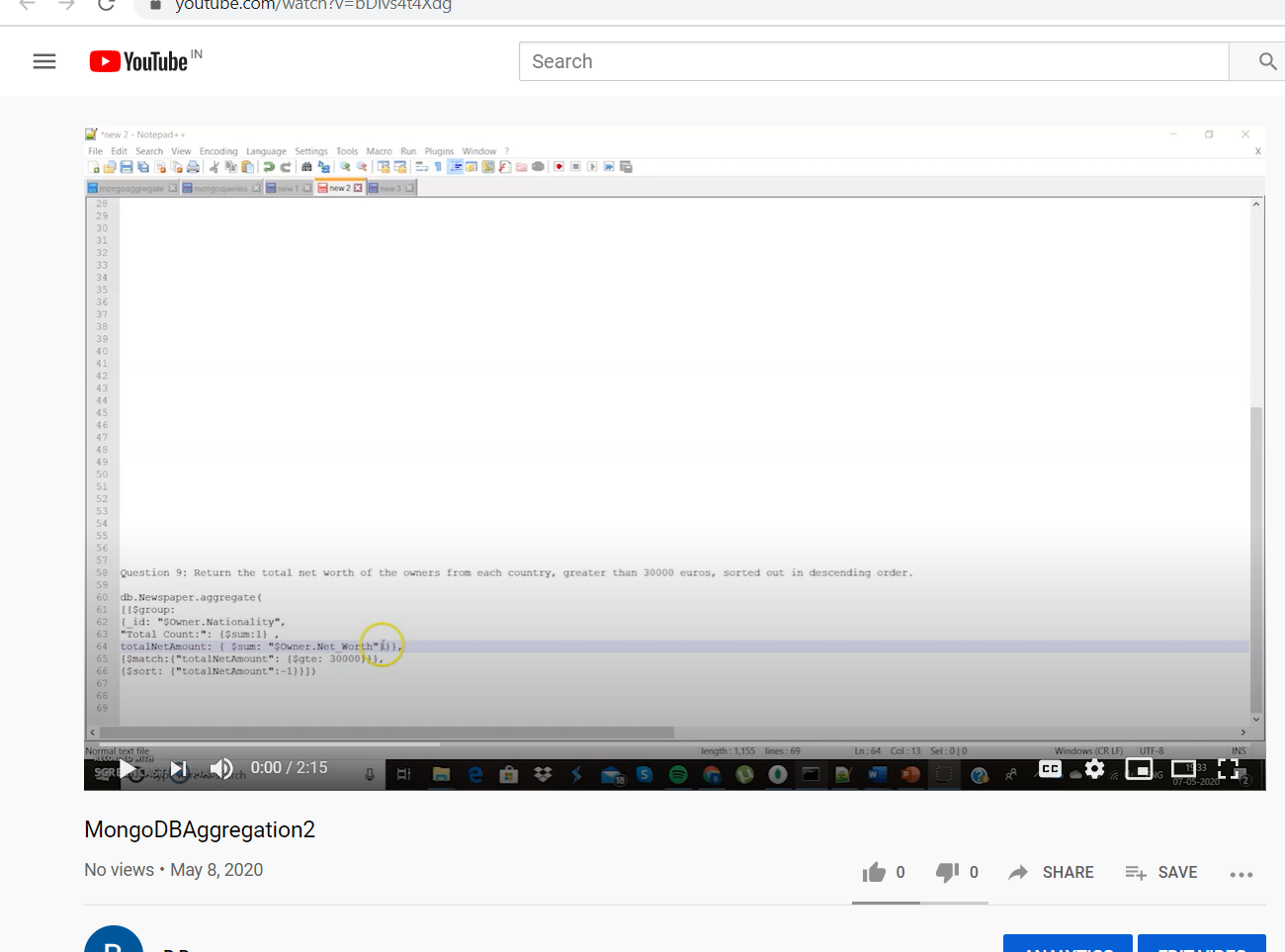


Figure : Youtube Video of Aggregation Part-2

# References

(n.d.). Retrieved from https://docs.mongodb.com/manual/reference/operator/update/addToSet/

*addToSet/*. (n.d.). Retrieved from mongodb: https://docs.mongodb.com/manual/reference/operator/update/addToSet/

*aggregation/merge*. (n.d.). Retrieved from https://docs.mongodb.com: https://docs.mongodb.com/manual/reference/operator/aggregation/merge/

*insert-data-into-your-cluster*. (n.d.). Retrieved from https://docs.atlas.mongodb.com: https://docs.atlas.mongodb.com/tutorial/insert-data-into-your-cluster/

*introduction-mongodb-1*. (n.d.). Retrieved from dzone: https://dzone.com/articles/introduction-mongodb-1

*method/db.collection.findAndModify*. (n.d.). Retrieved from mongodb: https://docs.mongodb.com/manual/reference/method/db.collection.findAndModify/

*mongodb-query-with-multiple-conditions*. (n.d.). Retrieved from stackoverflow.com: https://stackoverflow.com/questions/39389823/mongodb-query-with-multiple-conditions

*mongodb-query-with-multiple-conditions*. (n.d.). Retrieved from https://stackoverflow.com: https://stackoverflow.com/questions/39389823/mongodb-query-with-multiple-conditions

*operator/update/push/*. (n.d.). Retrieved from mongodb: https://docs.mongodb.com/manual/reference/operator/update/push/

*orderby*. (n.d.). Retrieved from mongodb: https://docs.mongodb.com/manual/reference/operator/meta/orderby/

*query-modifier*. (n.d.). Retrieved from mongodb.com: https://docs.mongodb.com/manual/reference/operator/query-modifier/

*reduce*. (n.d.). Retrieved from https://docs.mongodb.com: https://docs.mongodb.com/manual/reference/operator/aggregation/reduce/

*regex*. (n.d.). Retrieved from https://docs.mongodb.com: https://docs.mongodb.com/manual/reference/operator/query/regex/

*retrieve-only-queried-element-in-an-object-array-in-mongodb-collection*. (n.d.). Retrieved from blog.fullstacktraining: https://blog.fullstacktraining.com/retrieve-only-queried-element-in-an-object-array-in-mongodb-collection/

*retrieve-only-the-queried-element-in-an-object-array-in-mongodb-collection*. (n.d.). Retrieved from https://stackoverflow.com: https://stackoverflow.com/questions/3985214/retrieve-only-the-queried-element-in-an-object-array-in-mongodb-collection

*unwind*. (n.d.). Retrieved from https://docs.mongodb.com: https://docs.mongodb.com/manual/reference/operator/aggregation/unwind/

*update-documents-with-aggregation-pipeline*. (n.d.). Retrieved from https://docs.mongodb.com: https://docs.mongodb.com/manual/tutorial/update-documents-with-aggregation-pipeline/

*updating-data-objects-nested-in-document-arrays-in-mongodb-5ef007fa51b7*. (n.d.). Retrieved from https://medium.com: https://medium.com/@Saheedt/updating-data-objects-nested-in-document-arrays-in-mongodb-5ef007fa51b7