**Name:** Fredrick J. Sigalla

**Student Number:** A00277360

**Course Name:** MSc. In Data Analytics

**Module Name:** Advanced Databases

**Project Title:** Irish Dancing Lessons (IDL)

**Date of Submission:** 08th May 2020

Table of Contents

[PROJECT OVERVIEW 3](#_Toc39860250)

[DATABASE SETUP 4](#_Toc39860251)

[MONGO SHELL – SIMPLE QUERIES 8](#_Toc39860252)

[MONGO SHELL – AGGREGATION FRAMEWORK 35](#_Toc39860253)

[MONGODB COMPASS 53](#_Toc39860254)

[Overview 53](#_Toc39860255)

[Evidence of using mongodb compass/atlas with idl data 61](#_Toc39860256)

[APPENDIX 68](#_Toc39860257)

[REFERENCE 69](#_Toc39860258)

# PROJECT OVERVIEW

Irish Dancing Lessons (IDL) is the database which manages dancing lessons taking part at different counties in Ireland. Each document includes a dancing lesson that captures dancer’s details, lesson venue, lesson fee, lesson duration, teaching technique, progress of the lesson and the details of the teacher providing the lesson.

One venue can host one or more lessons, also one teacher can teach one or more lessons taking place at different venues, and lastly, a dancer can take only one dancing lesson at a time.

While a dancer can take only one dancing lesson at a time, a lesson may include one or more dancing styles.

Despite that every teacher has specialities in certain age group of dancers, occasionally a teacher can be assigned to provide a lesson to a dancer whose age is not within the teacher’s age group of specialities. For example, a teacher whose speciality is mainly teenagers, can occasionally be assigned to provide lesson to kids, and vice versa.

Also, one or more teaching technique can be employed during lesson delivery. For instance, a dancer can be taught by using video clip technique only or the combination of video clip, Graphics and Teacher Demonstration.

Lastly, every lesson must include one teacher and one dancer only.

# DATABASE SETUP

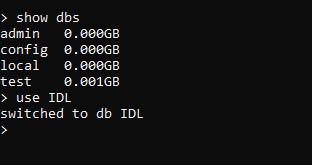
The Irish Dancing Lesson is managed by the database called IDL, which is the initials of the project name.

Running show dbs command to make sure the database is not already existing in local server

*show dbs*

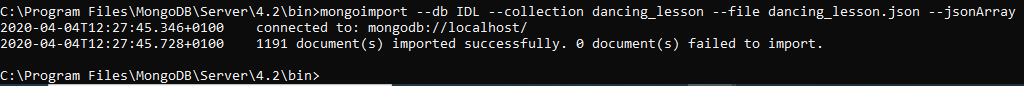
Running use IDL command to create a database named IDL

*use IDL*



Running command mongoimport to import a json file named dancing\_lesson, into MongoDB *IDL* database’s collection called *dancing\_lesson.*

*mongoimport –db IDL –collection dancing\_lesson –file dancing\_lesson.json jsonArray*



Confirming if the collection *dancing\_lesson* has been successfully created in the IDL database

*show collections*



Checking the first, last and any other arbitrary documents to make sure all fields of the document have been well imported as well as to check the structure of the document.

First document:

*db.dancing\_lesson.find().limit(1).pretty()*



Last document:

*db.dancing\_lesson.find().sort({\_id : -1}).limit(1).pretty()*



Another arbirtrary document:

db.dancing\_lesson.find({\_id : 567}).pretty()



# MONGO SHELL – SIMPLE QUERIES

**Query 1**

Show the number of lessons that have so far been added to the database.

db.dancing\_lesson.find().count()



**Query 2**

Use any arbitrary document to show the details that are recorded to the database for every new lesson.

db.dancing\_lesson.find(

{\_id : 45}

).pretty()



**Query 3**

Add one new dancer called Khumalo Nokwadzi to the database, the dancer should be assigned to the new teacher called Super Nyamwela. The lesson is to take place at Kwame Nkrumah hall located at Sligo county which is also the address of the dancer. The lesson is expected to last for 39 hours, costing 600.60 Euros. Since the teacher is not in the country, the lesson will be delivered by using Video Clips and Graphics which will be prepared by th teacher. Other dancer’s details include year of birth and height which are 1980 and 172.33 cm respectively. Other details like styles and teacher’s details will be updated later.

*Checking number of documents before adding a record*

db.dancing\_lesson.find().count()



*Adding a new document and checking the number of document to confirm the increase of one document.*

db.dancing\_lesson.insert(

{

"\_id" : 2000,

"Dancer" :{

"Name" : "Khumalo Nokwadzi",

"Year\_Born" : 1980,

"Height" : 172.33,

"Gender" : "Male",

"County" : "Sligo"

} ,

"Venue" : "Kwame Nkrumah",

"Lesson\_Duration" : 36,

"Lesson\_Fee" : 600.60,

"Teaching\_Technique" : ["Video Clip", "Graphics"],

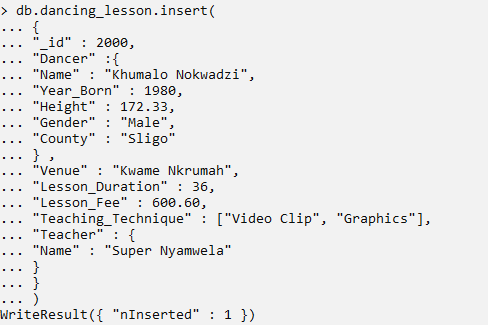
"Teacher" : {

"Name" : "Super Nyamwela"

}

}

)



db.dancing\_lesson.find().count()



**Query 4**

Every lesson include one dancer and one teacher. Even if the teacher can teach many lessons (dancers), a dancer can only be taught by once teacher. Therefore, use dancer’s name as a criteria to update details of the above added document by adding Baikoko as the dancer’s Style, progress poor, as well as the rest of the teacher’s details which include weekly working hours, dance type and age groups whose values are 20, “Baikoko, Mduara, Jigs” and “Young Adults” respectively.

db.dancing\_lesson.update(

{"Dancer.Name" : "Khumalo Nokwadzi"},

{

$set :

{

"Dancer.Styles" : ["Baikoko"],

"Progress" : "Poor",

"Teacher.Hours" : 20,

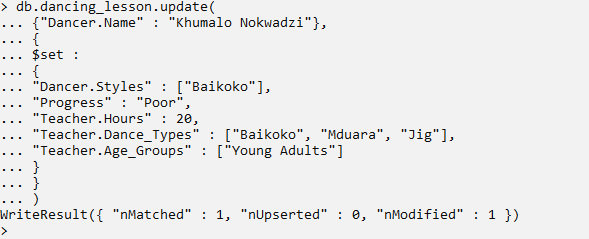
"Teacher.Dance\_Types" : ["Baikoko", "Mduara", "Jig"],

"Teacher.Age\_Groups" : ["Young Adults"]

}

}

)



db.dancing\_lesson.find({“Dancer.Name” : “Khumalo Nokwadzi”}).pretty()



**Query 5.**

Update progress of the lesson taken by Khumalo Nokwadzi to Excellent.

*Value of progress before an update*

db.dancing\_lesson.find({\_id : 2000}, {"Dancer.Name" : 1, "Progress" : 1})



db.dancing\_lesson.update(

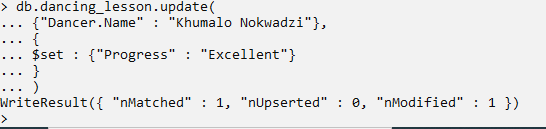
{"Dancer.Name" : "Khumalo Nokwadzi"},

{

$set : {"Progress" : "Excellent"}

}

)



*Value of progress after an update*

db.dancing\_lesson.find({\_id : 2000}, {"Dancer.Name" : 1, "Progress" : 1})



**Query 6**

Add three more new dancers with ids 2500, 3000, 3500 and names Ansu James, Rad Razmutte and Mac Show respectively. All three new dancers are to be taught by a teacher called Basia Bernard. The gender and Height for three new dancers are Female, Female, Male and 168cm, 170cm, 171cm respectively.

*Number of documents before adding three new dancers*

db.dancing\_lesson.find().count()



*Adding three new dancers (documents)*

var new\_dancers = [

{

"\_id" : 2500,

"Dancer" : {"Name" : "Ansu James", "Gender" : "Female", "Height" : 168},

"Teacher" : {"Name" : "Basia Bernard"}

},

{

"\_id" : 3000,

"Dancer" : {"Name" : "Rad Razmutte", "Gender" : "Female", "Height" : 170},

"Teacher" : {"Name" : "Basia Bernard"}

},

{

"\_id" : 3500,

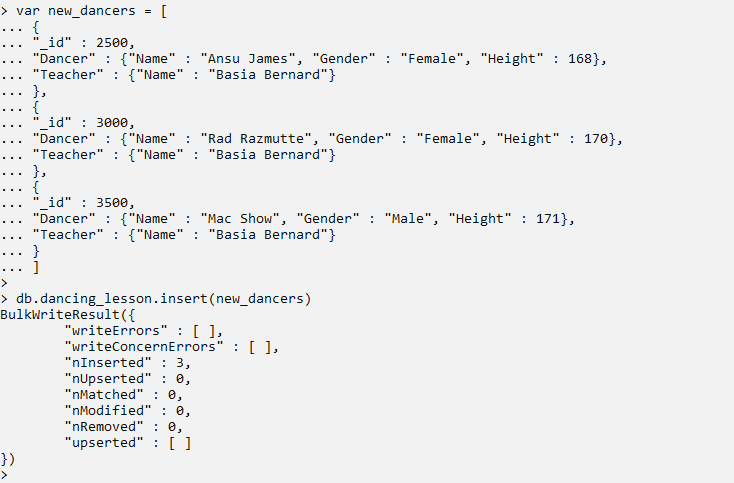
"Dancer" : {"Name" : "Mac Show", "Gender" : "Male", "Height" : 171},

"Teacher" : {"Name" : "Basia Bernard"}

}

]

db.dancing\_lesson.insert(new\_dancers)



Number of lessons (documents/dancers) after adding 3 three new dancers

db.dancing\_lesson.find().count()



**Query 7**

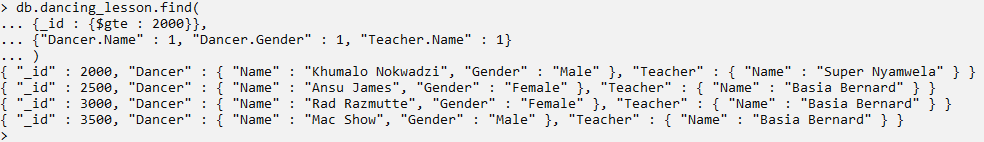
Show the name, gender and teacher of all the dancers with lesson ID greater or equal to 2000.

db.dancing\_lesson.find(

{\_id : {$gte : 2000}},

{"Dancer.Name" : 1, "Dancer.Gender" : 1, "Teacher.Name" : 1}

)



**Query 8**

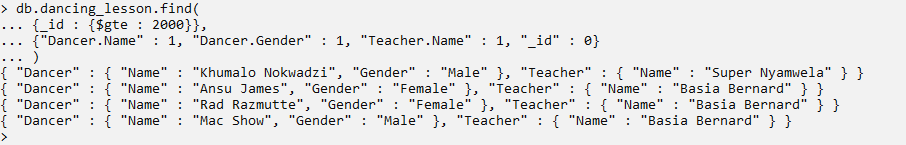
Show the name, gender and teacher of all the dancers with lesson ID greater or equal to 2000, don’t include the lesson ID as part of the output.

db.dancing\_lesson.find(

{\_id : {$gte : 2000}},

{"Dancer.Name" : 1, "Dancer.Gender" : 1, "Teacher.Name" : 1, "\_id" : 0}

)



**Query 9**

Delete all the four newly added dancers. It should be noted that, the Ids of the four dancers are 2000 2500, 300 and 3500.

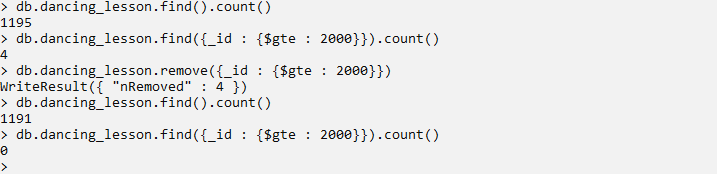
db.dancing\_lesson.find().count()

db.dancing\_lesson.find({\_id : {$gte : 2000}}).count()

db.dancing\_lesson.remove({\_id : {$gte : 2000}})

db.dancing\_lesson.find().count()

db.dancing\_lesson.find({\_id : {$gte : 2000}}).count()



**Query 10**

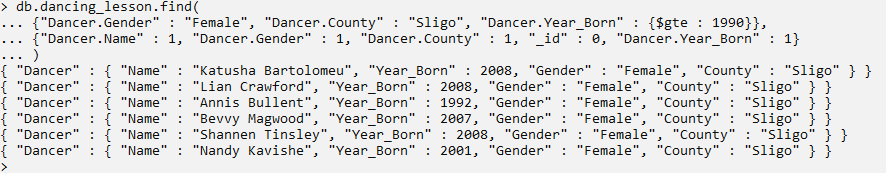
Show the names of all female dancers from county Sligo that were born as early as 1990.

db.dancing\_lesson.find(

{"Dancer.Gender" : "Female", "Dancer.County" : "Sligo", "Dancer.Year\_Born" : {$gte : 1990}},

{"Dancer.Name" : 1, "Dancer.Gender" : 1, "Dancer.County" : 1, "\_id" : 0, "Dancer.Year\_Born" : 1}

)



**Query 11**

Show the name and address of female dancers who have “Jig" as one of their dancing styles

db.dancing\_lesson.find(

{"Dancer.Styles" : "Jig", "Dancer.Gender" : "Female"},

{"Dancer.Name" : 1, "Dancer.County" : 1, "Dancer.Gender" : 1, "Dancer.Styles" : 1, \_id : 0}

)



**Query 12**

List the name and year of birth of all male dancers that are being taught by the teacher called Judith Reyes.

db.dancing\_lesson.find(

{

"Teacher.Name" : "Judith Reyes",

"Dancer.Gender" : "Male"

},

{

"Dancer.Name" : 1,

"Dancer.Year\_Born" : 1,

"Dancer.Gender" : 1,

"Teacher.Name" : 1,

"\_id" : 0

}

)

db.dancing\_lesson.find(

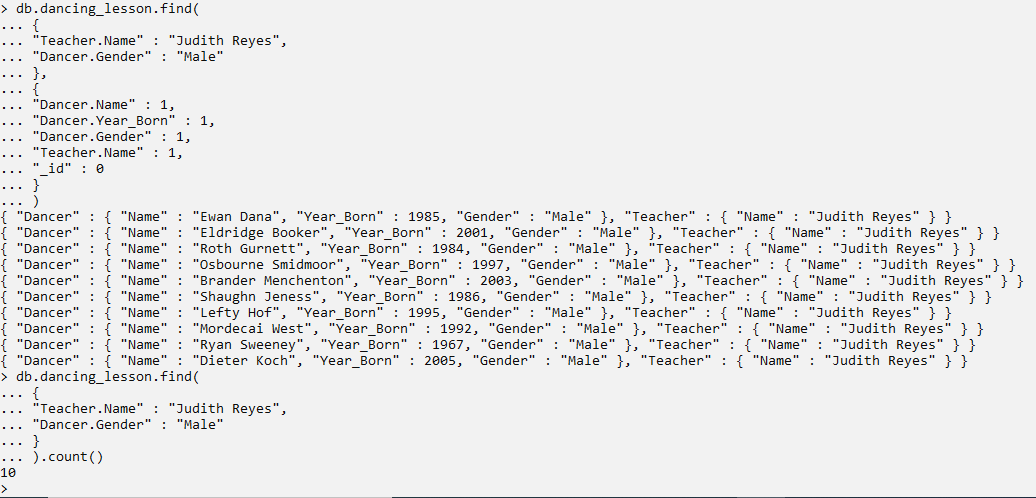
{

"Teacher.Name" : "Judith Reyes",

"Dancer.Gender" : "Male"

}

).count()

`

**Query 13**

Display the names of all the teacher and female dancers for all the lessons with excellent progress and whose lesson fee is less than 500 Euros.

db.dancing\_lesson.find(

{

"Dancer.Gender" : "Female",

"Progress" : "Excellent",

"Lesson\_Fee" : {$lt : 500}

},

{

"Dancer.Name" : 1,

"Teacher.Name" : 1,

"Progress" : 1,

"Lesson\_Fee" : 1,

"\_id" : 0

}

)

db.dancing\_lesson.find(

{

"Dancer.Gender" : "Female",

"Progress" : "Excellent",

"Lesson\_Fee" : {$lt : 500}

}

).count()



**Query 14**

List 10 lessons that have the highest lesson fee with their corresponding venue and the name of the teacher providing the lesson.

db.dancing\_lesson.find(

{},

{

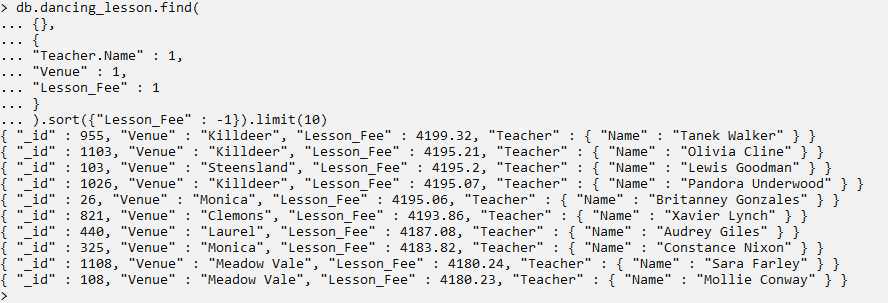
"Teacher.Name" : 1,

"Venue" : 1,

"Lesson\_Fee" : 1

}

).sort({"Lesson\_Fee" : -1}).limit(10)



**Query 15**

Display the name, gender, county and the height of the second shortest dancer.

db.dancing\_lesson(

{},

{

\_id : 0,

"Dancer.Name" : 1,

"Dancer.Gender" : 1,

"Dancer.County" : 1,

"Dancer.Height" : 1

}

).sort({"Dancer.Height" : 1}).skip(1).limit(1)



**Query 16**

Display the number of lessons that use “Video Clip” as one of the ways to deliver a lesson.

db.dancing\_lesson.find({"Teaching\_Technique" : "Video Clip"}).count()



**Query 17**

Display the number of lessons that use **only** “Video Clip” as a means to deliver a lesson.

db.dancing\_lesson.find({"Teaching\_Technique" : ["Video Clip"]}).count()



**Query 18**

Show the name of all female dancers from county Dublin who do four dancing styles.

db.dancing\_lesson.find(

{

"Dancer.Gender" : "Female",

"Dancer.County" : "Dublin",

"Dancer.Styles" : {$size : 4}

},

{

\_id : 0,

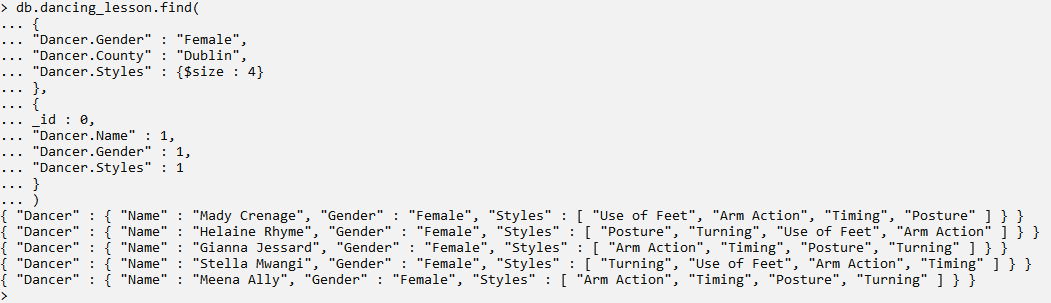
"Dancer.Name" : 1,

"Dancer.Gender" : 1,

"Dancer.Styles" : 1

}

)



**Query 19**

Show the details of all the lessons which are being held in North Tipperary whose fee is at least 3000. Limit the details to name of the dancer and the teacher teaching the lesson.

db.dancing\_lesson.find(

{$and :

[

{"Dancer.County" : "North Tipperary"},

{"Lesson\_Fee" :

{$gte : 3000}

}

]

},

{

\_id : 0,

"Dancer.Name" : 1,

"Teacher.Name" : 1,

"Dancer.County" : 1,

"Lesson\_Fee" : 1

}

).pretty()

db.dancing\_lesson.find(

{$and :

[

{"Dancer.County" : "North Tipperary"},

{"Lesson\_Fee" :

{$gte : 3000}

}

]

},

{

\_id : 0,

"Dancer.Name" : 1,

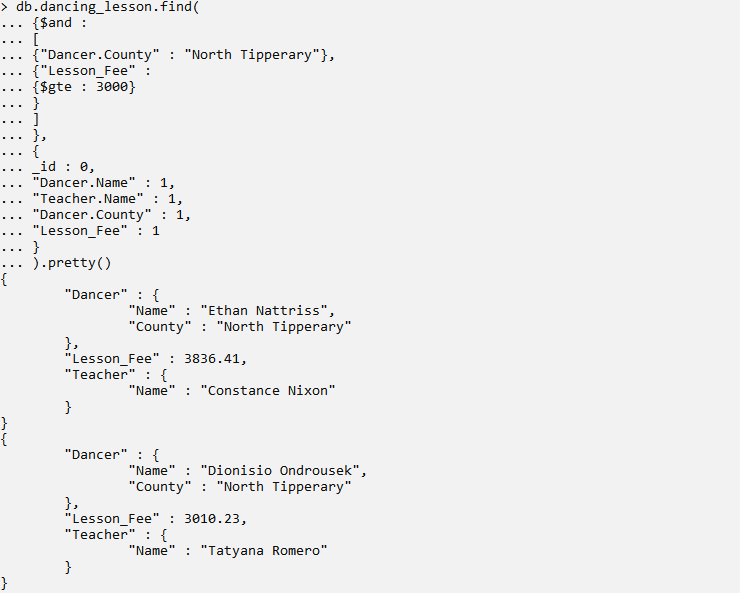
"Teacher.Name" : 1,

"Dancer.County" : 1,

"Lesson\_Fee" : 1

}

).count()







**Query 20**

Show the details of all the teachers who teaches only Salsa, nothing more, and yet they work for more than 30 hours per week. Since the teacher can teach more than one student, then only lesson for whose dancers are from Offaly Country are the one who should be displayed. Sort them by number of hours by starting with the one with the most number hours per week.

db.dancing\_lesson.find(

{

"Teacher.Dance\_Types" : ["Salsa"],

"Teacher.Hours" :

{

$gt : 30

},

"Dancer.County" : "Offaly"

},

{

\_id : 0,

"Teacher.Name" : 1,

"Teacher.Dance\_Types" : 1,

"Teacher.Hours" : 1,

"Dancer.Name" : 1,

"Dancer.County" : 1

}

).sort({"Teacher.Hours" : -1})

db.dancing\_lesson.find(

{

"Teacher.Dance\_Types" : ["Salsa"],

"Teacher.Hours" :

{

$gt : 30

},

"Dancer.County" : "Offaly"

},

{

\_id : 0,

"Teacher.Name" : 1,

"Teacher.Dance\_Types" : 1,

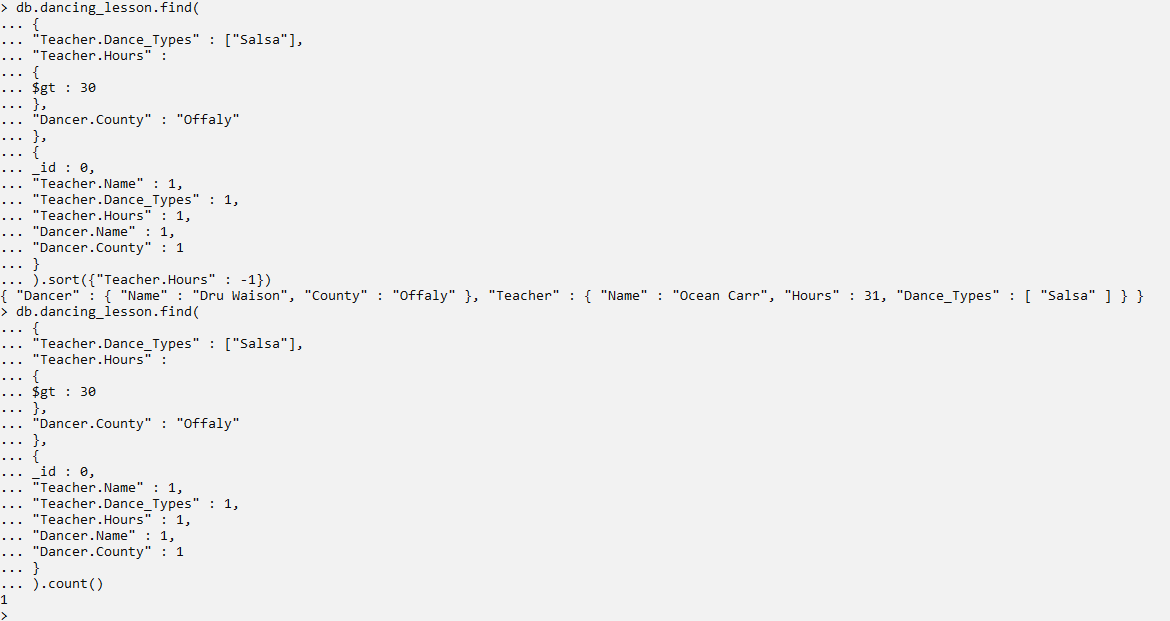
"Teacher.Hours" : 1,

"Dancer.Name" : 1,

"Dancer.County" : 1

}

).count()



**Query 21**

Show all the dancers from Leitrim whose first names starts with letter B

db.dancing\_lesson.find(

{

"Dancer.Name" :

{

$regex : /^Bz\*/

},

"Dancer.County" : "Leitrim"

},

{

"Dancer.Name" : 1,

"Dancer.County" : 1,

\_id : 0

}

)

db.dancing\_lesson.find(

{

"Dancer.Name" :

{

$regex : /^Bz\*/

},

"Dancer.County" : "Leitrim"

},

{

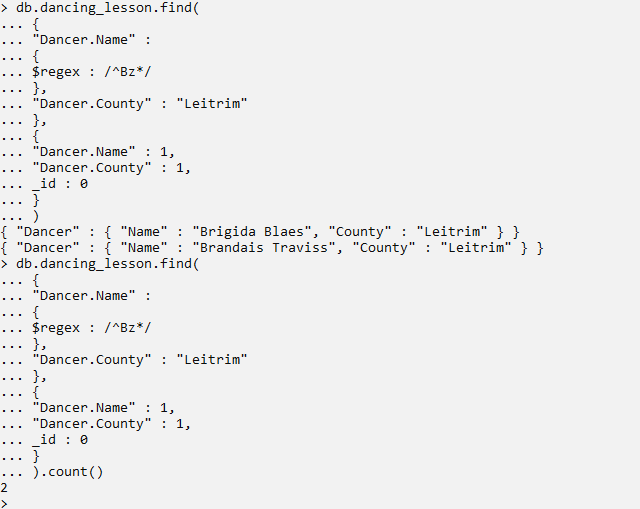
"Dancer.Name" : 1,

"Dancer.County" : 1,

\_id : 0

}

).count()



# MONGO SHELL – AGGREGATION FRAMEWORK

**Query 1**

Show how many dancing venues have so far been registered in the database and are generating income from the lesson fee.

db.dancing\_lesson.aggregate([

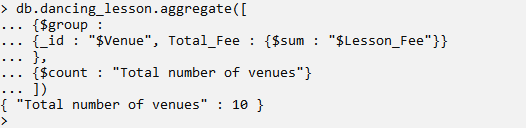
{$group :

{\_id : "$Venue", Total\_Fee : {$sum : "$Lesson\_Fee"}}

},

{$count : "Total number of venues"}

])



**Query 2**

Show how all the lessons have been allocated to all available venues. Display them by starting with the venues hosting the largest number of lessons to the one with the least number of lessons

db.dancing\_lesson.aggregate([

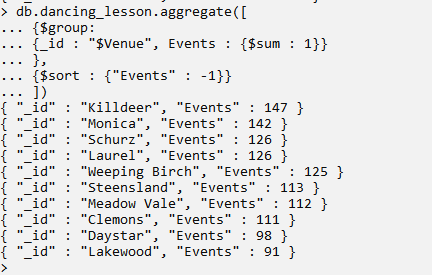
{$group:

{\_id : "$Venue", Events : {$sum : 1}}

},

{$sort : {"Events" : -1}}

])



**Query 3**

For all the lessons whose dancers are from Galway, show how they are distributed amongst the four progress categories so that to know the progress distribution of all the dancers from this county. Display them by first showing the category which reflect the progress of th majority of the dancers to the one with the least.

db.dancing\_lesson.aggregate([

{$match :

{"Dancer.County" : "Galway"}

},

{$project :

{"Dancer.County" : 1, "Progress" : 1, \_id : 0}

},

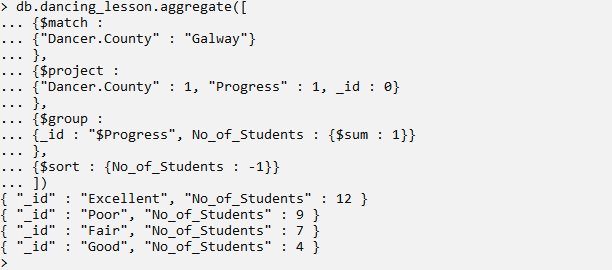
{$group :

{\_id : "$Progress", No\_of\_Students : {$sum : 1}}

},

{$sort : {No\_of\_Students : -1}}

])



**Query 4**

Query all the data from the database to check if the total amount of fee paid has an impact on lesson progress.

db.dancing\_lesson.aggregate([

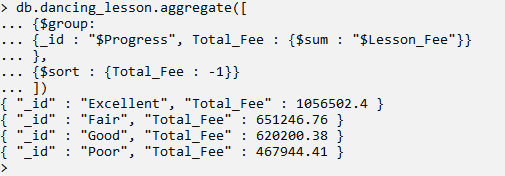
{$group:

{\_id : "$Progress", Total\_Fee : {$sum : "$Lesson\_Fee"}}

},

{$sort : {Total\_Fee : -1}}

])



**Query 5**

Show the number of female dancers from each county who are using Video Clip as one the learning techniques, provided that the total number of such group of dancers from a particular county is more than 10. Display the results by starting with the counties with the largest number of dancers first.

db.dancing\_lesson.aggregate([

{$match :

{

"Teaching\_Technique" : "Video Clip",

"Dancer.Gender" : "Female"

}

},

{$group :

{\_id : "$Dancer.County", No\_of\_Dancers : {$sum : 1}}

},

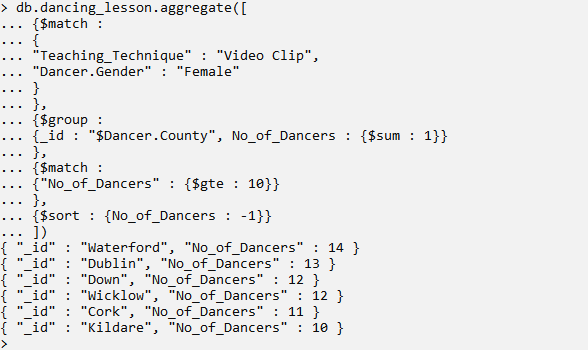
{$match :

{"No\_of\_Dancers" : {$gte : 10}}

},

{$sort : {No\_of\_Dancers : -1}}

])



**Query 6**

For all the male dancers from Dublin, show the average year of birth and how it’s related to the lesson progress. This can help to determine if the age of the dancers has anything to do the with dancer’s performance.

db.dancing\_lesson.aggregate([

{$match :

{"Dancer.Gender" : "Male"}

},

{$group :

{\_id :

{"Progress" : "$Progress"},

"Average\_Year\_Born" : {$avg : "$Dancer.Year\_Born"}

}

},

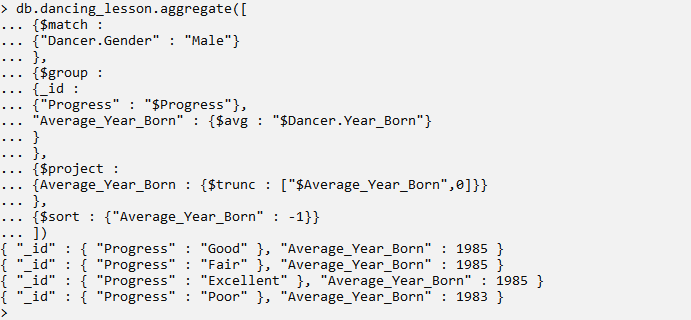
{$project :

{Average\_Year\_Born : {$trunc : ["$Average\_Year\_Born",0]}}

},

{$sort : {"Average\_Year\_Born" : -1}}

])



**Query 7**

Check for a dancers who have poor progress and yet are paying the highest lesson fee among all dancers with poor progress, add new information recommendating the dancer to be discontinued and place them them in a new collection called discontinued.

show collections

db.dancing\_lesson.aggregate([

{$match :

{"Progress" : "Poor"}

},

{$group :

{\_id : "$Dancer.Name", Max\_Fee : {$max : "$Lesson\_Fee"}}

},

{$sort : {"Max\_Fee" : -1}},

{$limit : 1},

{$addFields :

{"Recommendation" : "Discontinue"}

},

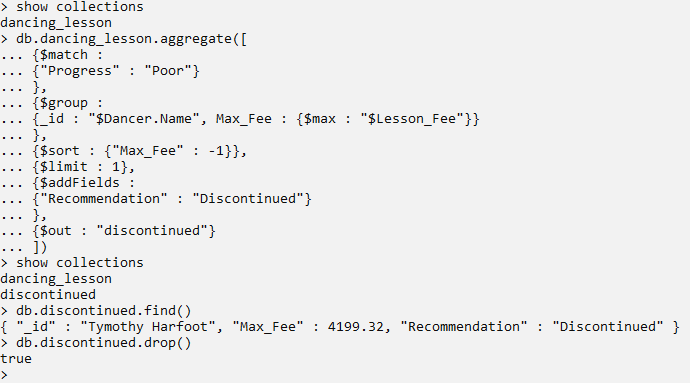
{$out : "discontinued"}

])

show collections

db.discontinued.find()

db.discontinued.drop()



**Query 8**

All the dancers from Sligo and Dublin who are doing only Jig dancing style, should be enrolled to one more dancing style called “Mduara” and added to a new collection called “Afro Dance”. Only the dancer’s name, address,date of birth and list of updated dancing styles should be added to the new collection.

show collections

db.dancing\_lesson.aggregate([

{$match :

{

"Dancer.County" : {$in : ["Sligo", "Dublin"]},

"Dancer.Styles" : ["Jig"]

}

},

{$project :

{

"\_id" : 0,

"Dancer.Name" : 1,

"Dancer.Year\_Born" : 1,

"Dancer.County" : 1,

"Dancer.Styles" : 1

}

},

{$addFields :

{"Dancer.Styles" :

{

$concatArrays : ["$Dancer.Styles", ["Mduara"]]

}

}

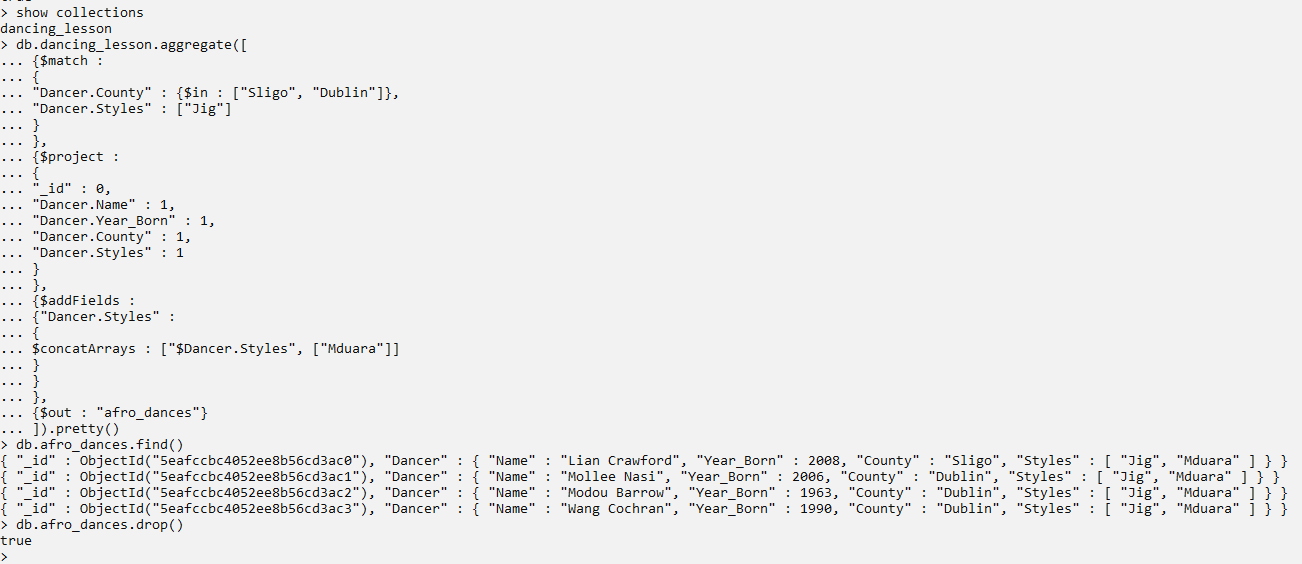
},

{$out : "afro\_dances"}

])

db.afro\_dances.find()

db.afro\_dances.drop()



**Query 9**

Show the average lesson fee spent on each available venue.

db.dancing\_lesson.aggregate([

{$group :

{\_id : "$Venue", "Average Fee" : {$avg : "$Lesson\_Fee"}}

},

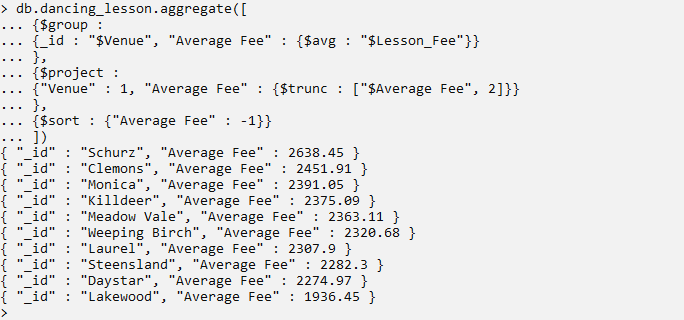
{$project :

{"Venue" : 1, "Average Fee" : {$trunc : ["$Average Fee", 2]}}

},

{$sort : {"Average Fee" : -1}}

])



**Query 10**

From the query above, by using the lesson fee of the venue with the lowest average lesson fee (Lakewood – 1936.36) as the reference point, show the average lesson fee of the for all the teachers who have excellent progress and teach for at least 30 hours a week.

db.dancing\_lesson.aggregate([

{$match :

{

"Progress" : "Excellent",

"Teacher.Hours" : {$gte : 30}

}

},

{$project :

{"Teacher.Name" : 1, "Lesson\_Fee" : 1}

},

{$group :

{\_id : "$Teacher.Name", "Average Fee" : {$avg : "$Lesson\_Fee"}}

},

{$project :

{"Teacher.Name" : 1, "Average Fee" : {$trunc : ["$Average Fee", 2]}}

},

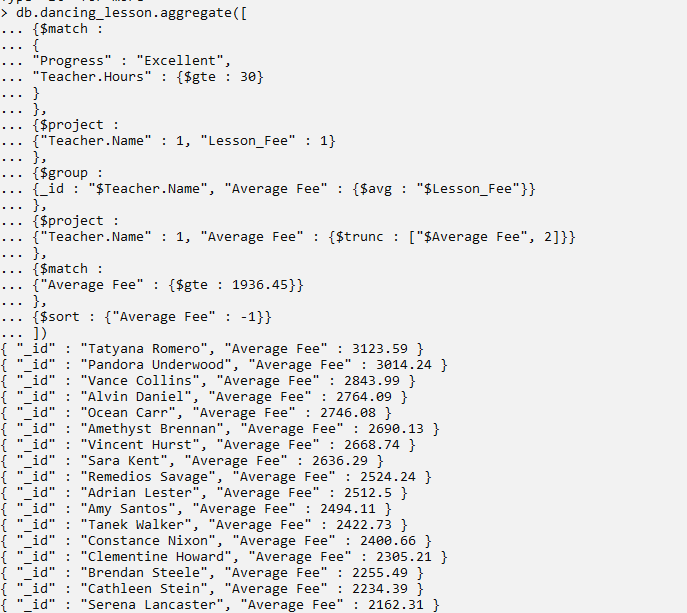
{$match :

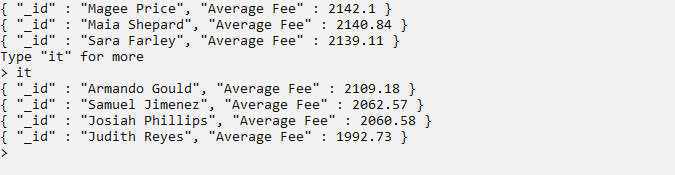
{"Average Fee" : {$gte : 1936.45}}

},

{$sort : {"Average Fee" : -1}}

])





**Query 11**

The two queries above show that the average maximum lesson fee by venue and teacher is 2638.45 and 3123.59 Euro respectively, where the most expensive venue is Schurz and the teacher is Tatyana Romeo exclusively. This show that, the teacher teaching a lesson has higher impact on the lesson than the venue. Since a lesson must take place in a certain venue, show the lesson fee by venue and teacher teaching the lesson combined for all the teachers with excellent progress and working for at least 30 hours per week. Categoriese them into 4000, 4100 and 4200 Euros. Show only the categories that have less than 10 documents.

db.dancing\_lesson.aggregate([

{$match :

{

"Progress" : "Excellent",

"Teacher.Hours" : {$gte : 30},

"Lesson\_Fee" : {$gte : 3700}

}

},

{$bucket :

{

groupBy : "$Lesson\_Fee",

boundaries : [4000, 4100, 4200],

default : "Other",

output : {

"count" : {$sum : 1},

"Lesson" : {

$push : {

"Venue and Teacher" :

{$concat : ["$Venue", " || ", "$Teacher.Name"]},

"Lesson Fee" : "$Lesson\_Fee"

}

}

}

}

},

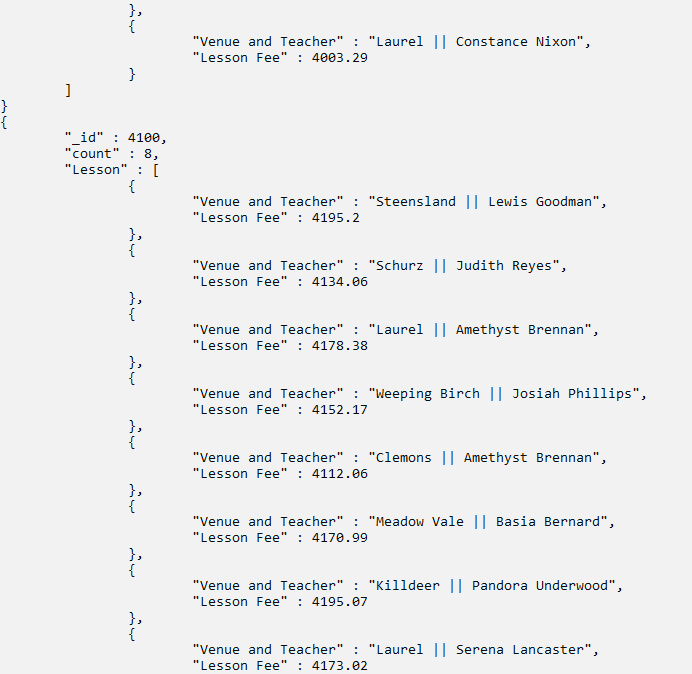
{

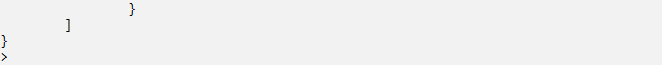
$match : {count : {$lt : 10}}

}

]).pretty()







**Query 12**

For all the lessons whose dancers are from Dublin county, show the details of the dancers who have poor progress on their lessons even though they are doing as few dance styles as not more than two. Display them by starting with the dancers doing the least number of dance styles.

db.dancing\_lesson.aggregate([

{$match :

{

"Dancer.County" : "Dublin",

"Progress" : "Poor"

}

},

{$project :

{

"Dancer.Name": 1,

"Dancer.Styles" : 1,

"NumberOfStyles" :

{$cond:

{if:

{$isArray: "$Dancer.Styles" }, then:

{ $size: "$Dancer.Styles" }, else: "NA"

}

},

"Progress" : 1,

"Dancer.County" : 1,

\_id : 0

}

},

{$match :

{"NumberOfStyles" : {$lte : 2}}

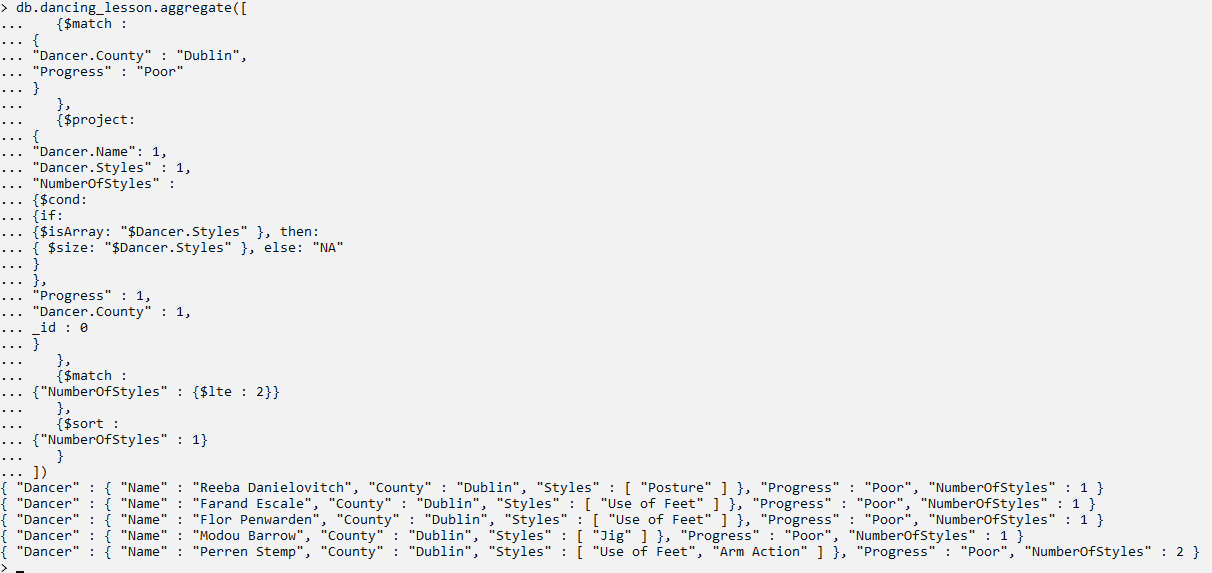
},

{$sort :

{"NumberOfStyles" : 1}

}

])



# MONGODB COMPASS

## Overview

MongoDB compass is the Graphical User Interface which is used to authenticate access and facilitate management of MongoDB instance. MongoDB Compass allow creation of databases, collections, and documents as well as execution of all other data manipulation functions.

Even though MongoDB Compass can be used to connect to a local MongoDB instance (database running on a local machine), but is commonly used to connect to a MongoDB SaaS (Software as a Service) platform commonly aliased as MongoDB Atlas.

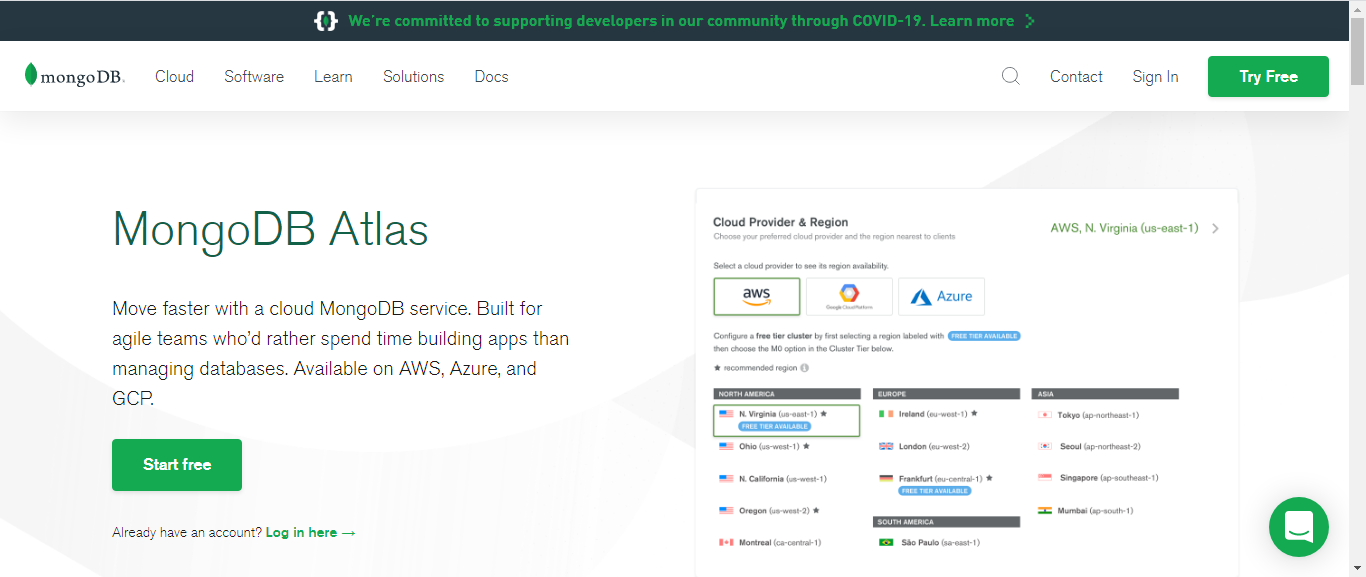
To use MongoDB Atlas, a user account is required, there is a free time unlimited subscription with limited services and a paid monthly subscription with unlimited services

## Setup

For this scenario, MongoDB Compass will be used to connect and manage a MongoDB instance on MongoDB Atlas.

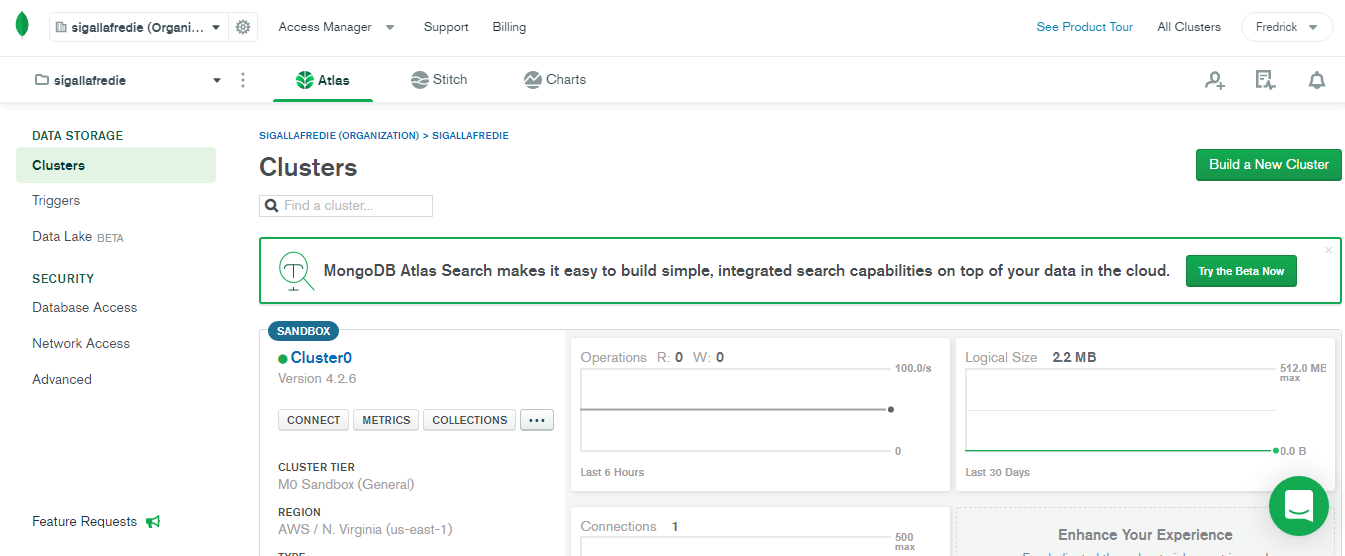
* To start with, a free subsbcrption account is created on MongoDB Atlas. MongoDB Atlas can be accessed through MongoDB official website (mongodb.com). All what is needed to create a free MongoDB Atlas account is a valid email address.

Figure 1.



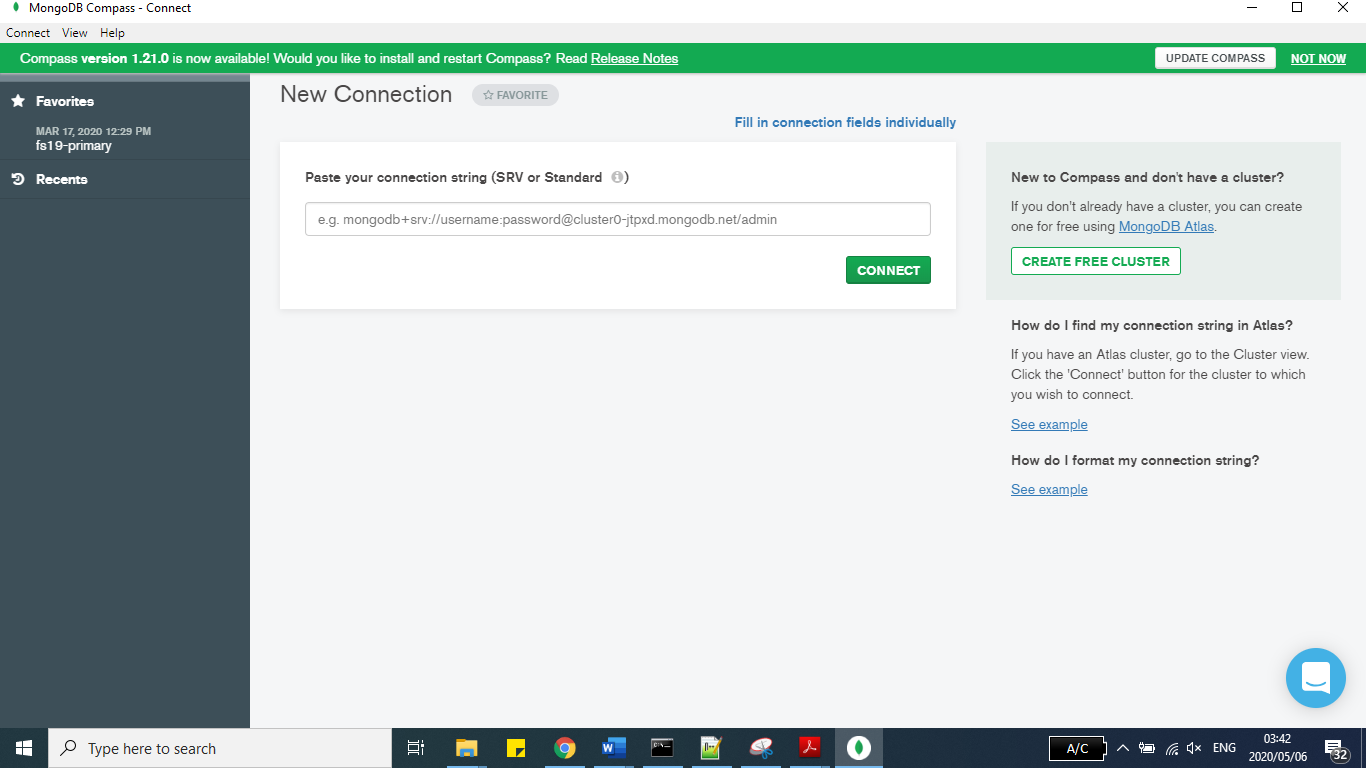
* After creating an account and login to MongoDB Atlas, just like any other SaaS platform, you need to create a cluster (node/droplet) which is basically a virtual machine.

Figure 2.



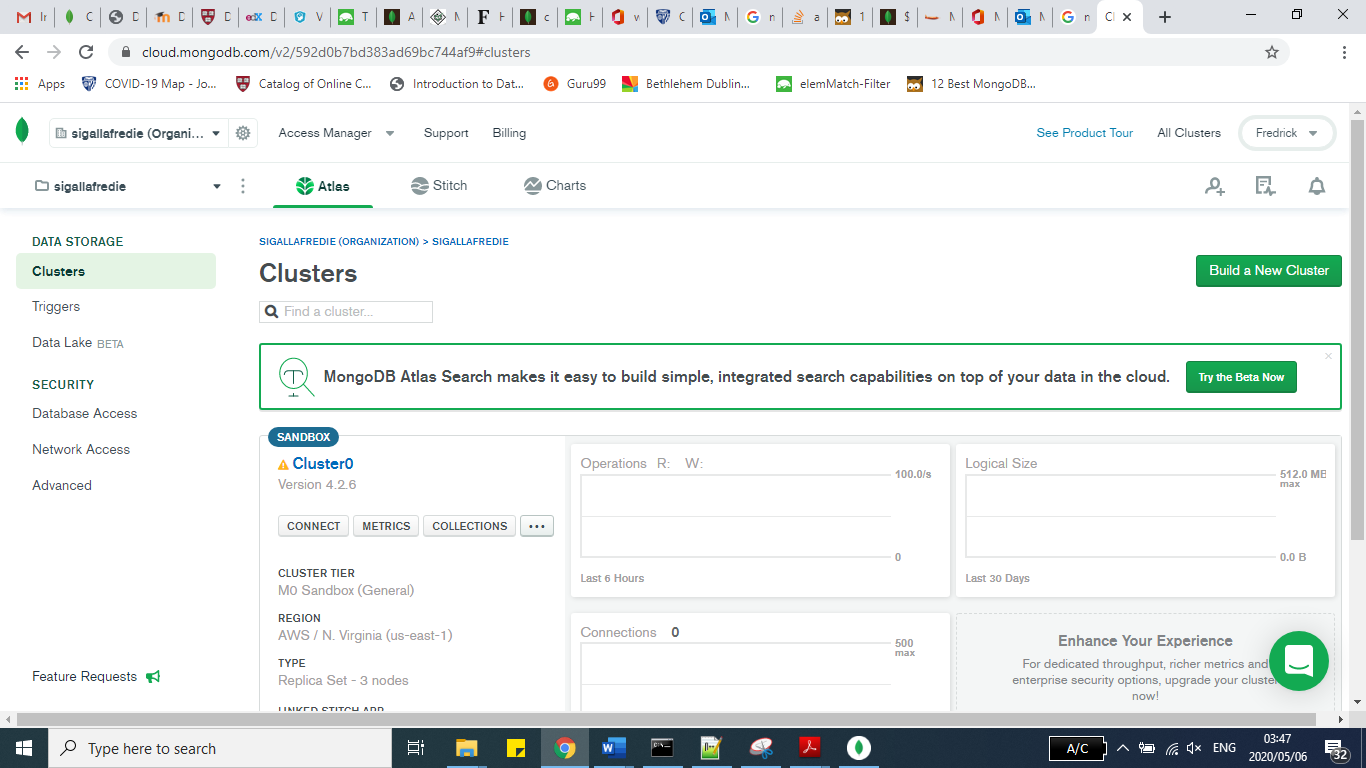
* After having MongoDB Atlas account and cluster all setup, MongoDB Compass is downloaded and install on the local machine. Launching a MongoDB Compass gives access to a connection interface

Figure 3



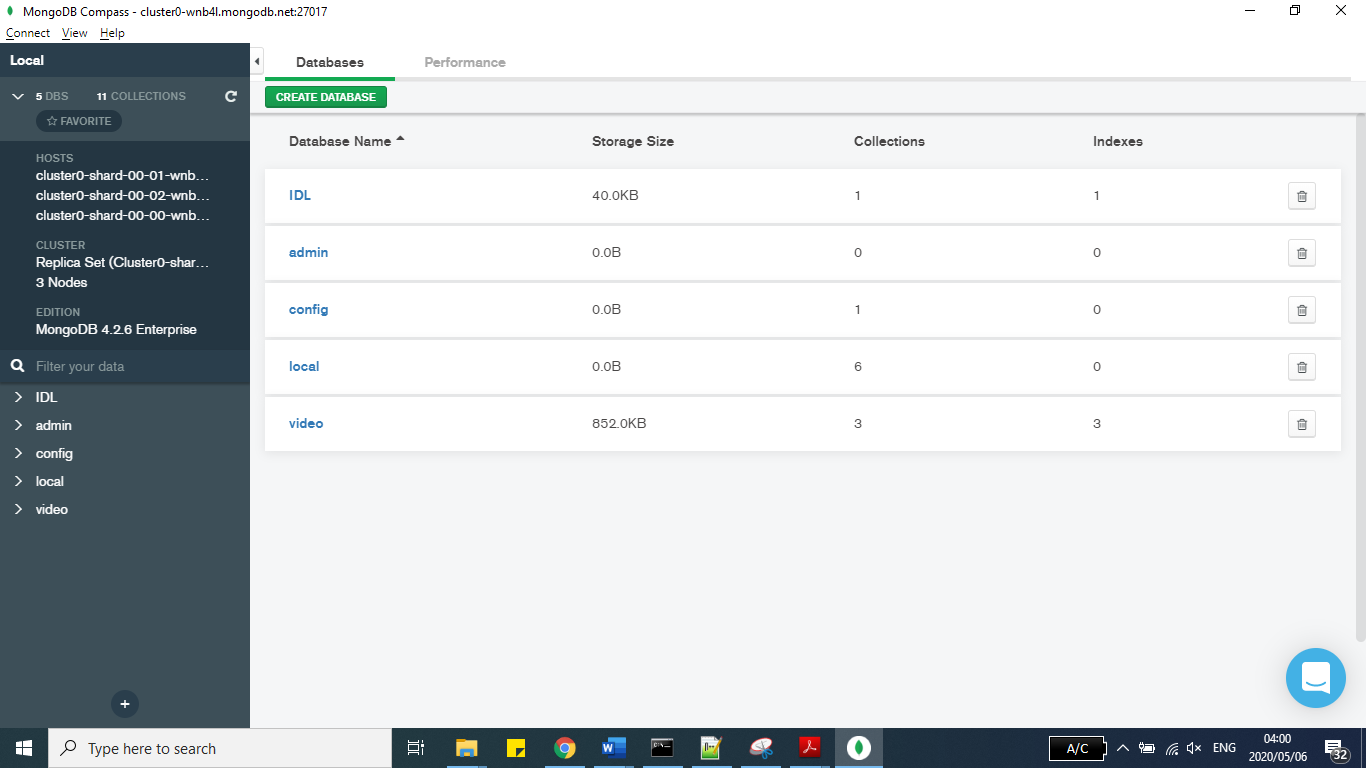
* Connection token to be used to connect MongoDB Compass to MongoDB Atlas can found on the MongoDB Atlas after pressing connect on the cluster created, which for this scenario, it’s cluster 0.

Figure 4.



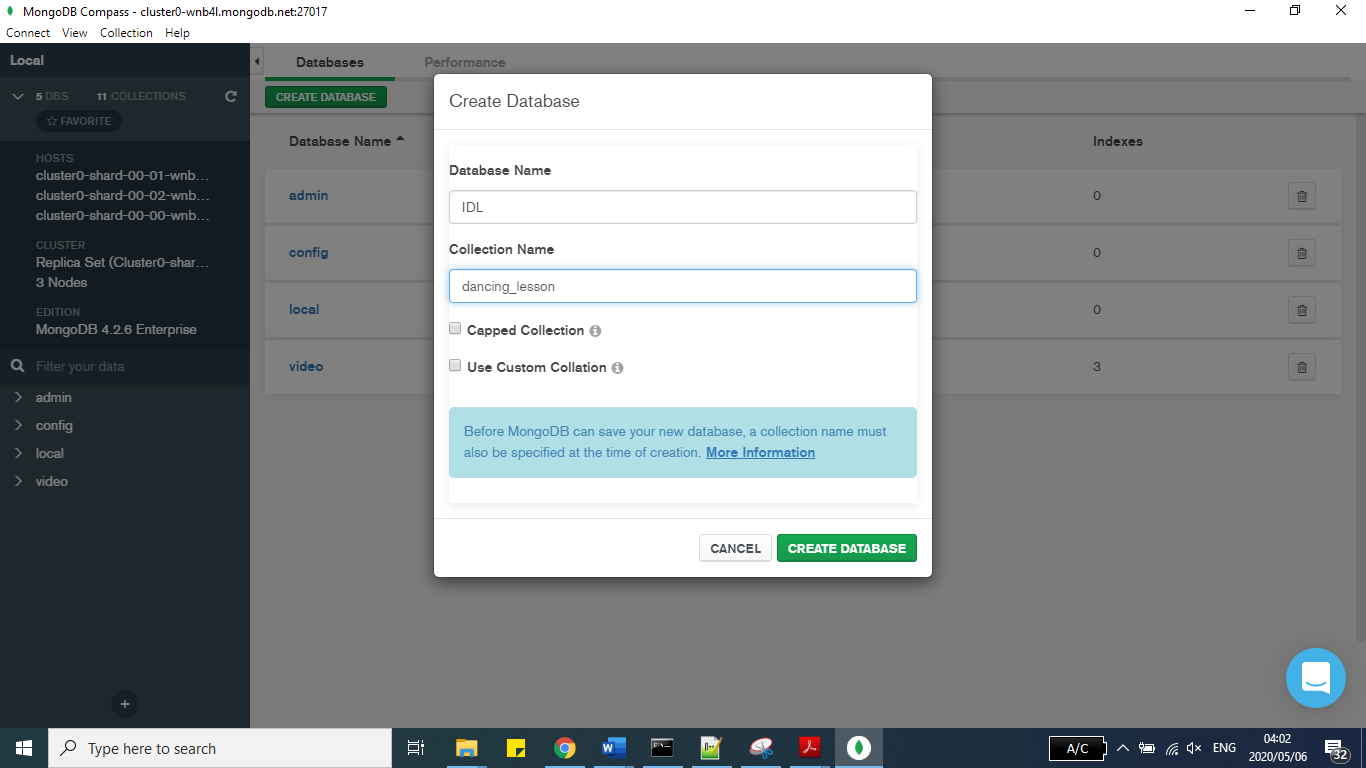
* After copying the token from MongoDB Atlas and pastig it on the connection page of the MongoDB Compass, the token substring *<password>* needs to be replaced with the password of your MongoDB Atlas account before proceeding with establishing connection. As long as there no errors on the connection token nor password, MongoDB Compass will connect to MongoDB Atlas and mirror all its basic content on the new interface. If there are any database already created previously, they will be seen as well. Clicking Create Database will provide an interface for creating a new database

Figure 5.



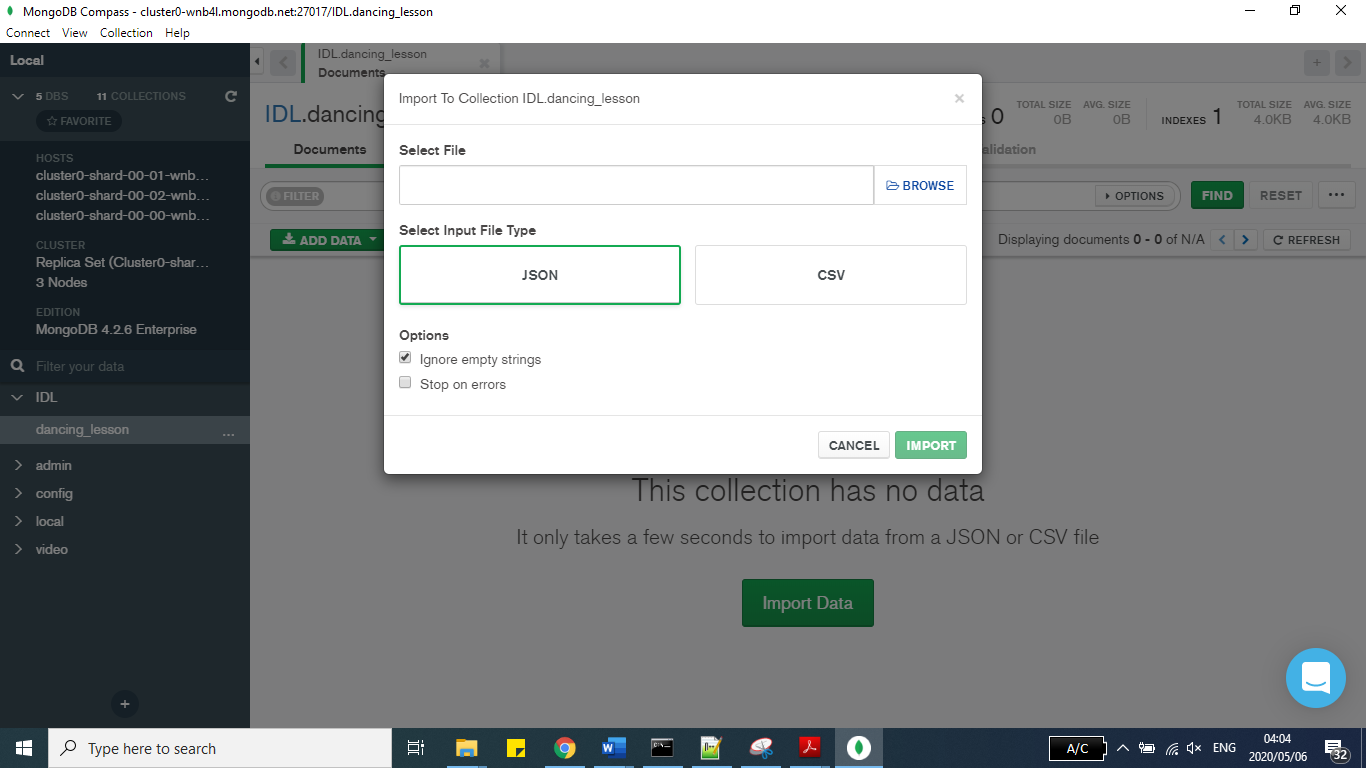
* At this interface, the new desired but descriptive database and collection and collection names are entered.

Figure 6



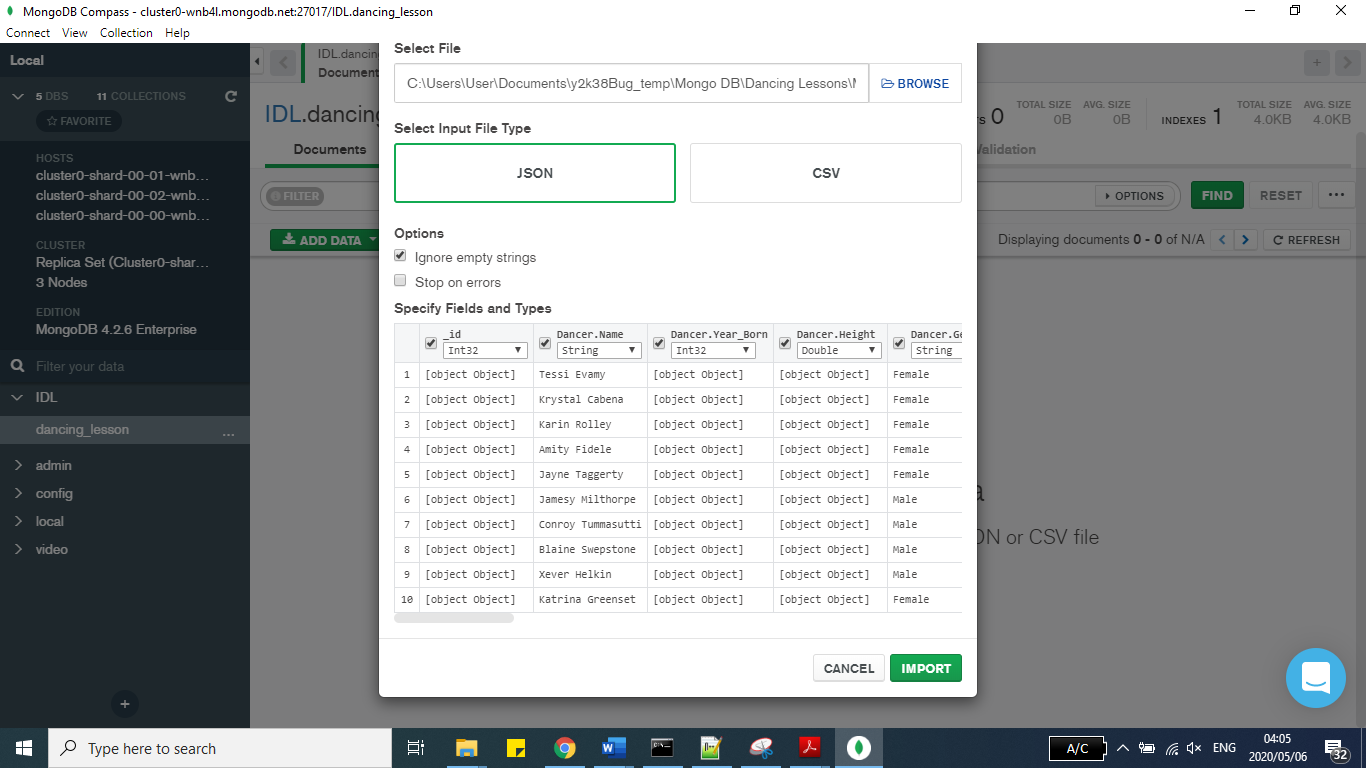
* After creating database and collection , the data file which contains the data is imported. As of version 1.12.0, MongoDB Compass (and Atlas) can import data which are either in JSON or CSV format only. For this case JSON format will be used.

Figure 7



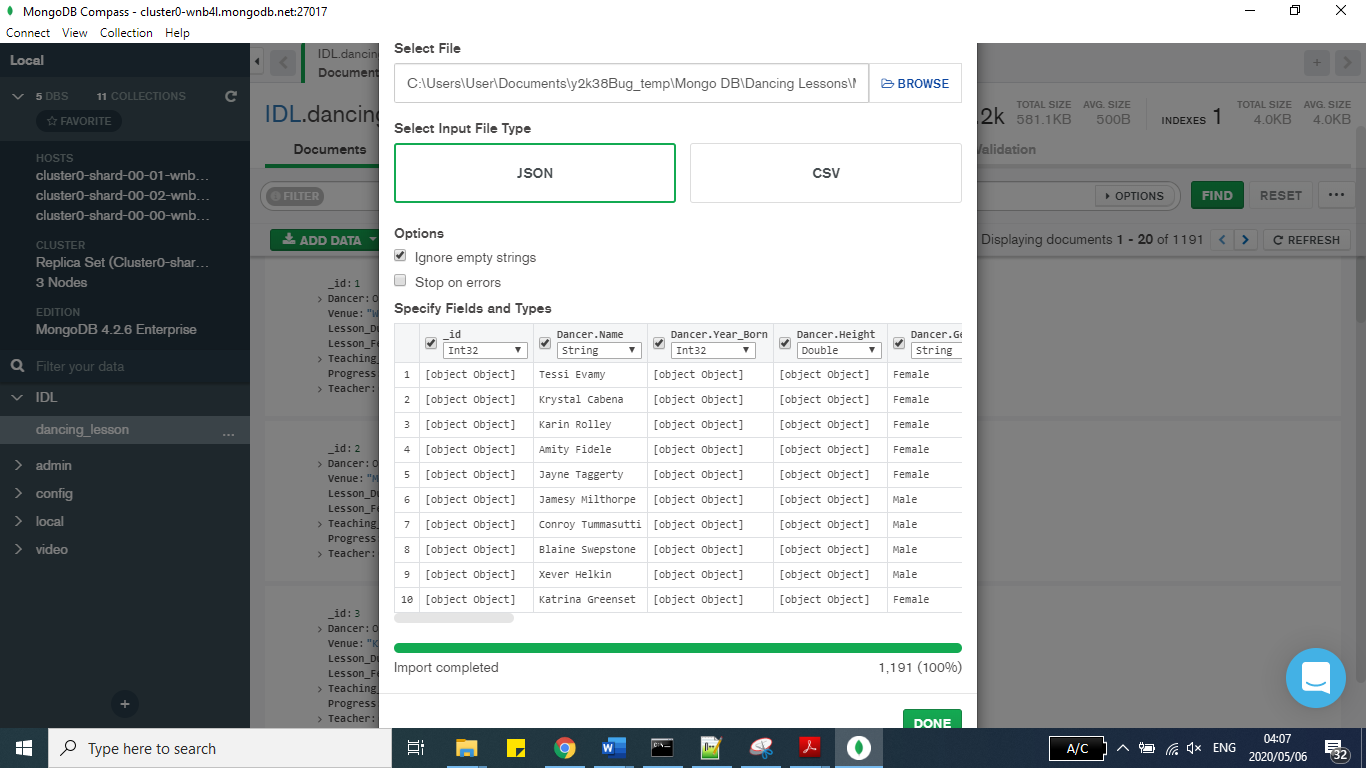
* While importing the data, “Stop on Errors” can be checked to stop the whole operation in case there are any errors in the data set. The erros can be punctuation errors mostly. If this options remains unchecked, MongoDB Compass will import all the documents except the ones after an error.

Figure 8



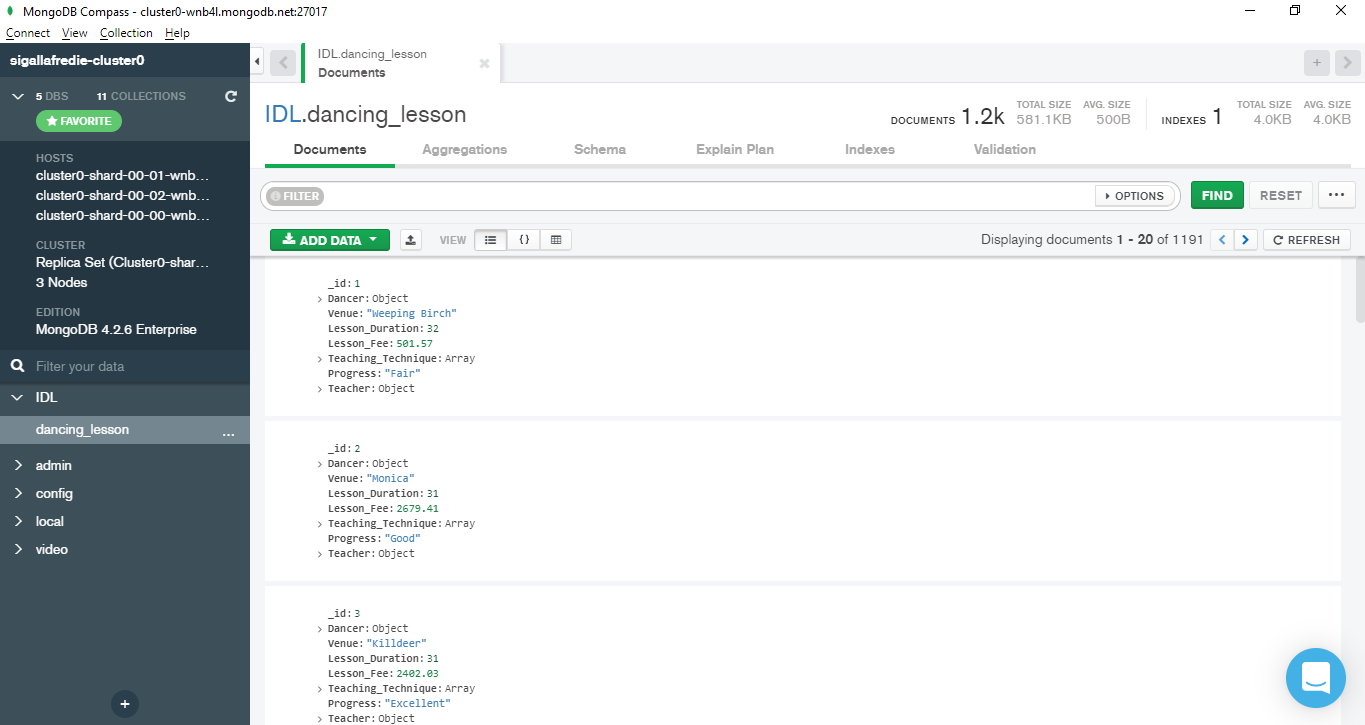
* After importing is complete, MongoDB Compass gives feedback on how many document have been successfully imported to the database without errors.

Figure 9



* After importing the data, the database is ready to use, with the capability of running all sort of operation from MongoDB Compass. Below snip is basically the page of MongoDB Compass after all the data has been imported and ready for running queries.

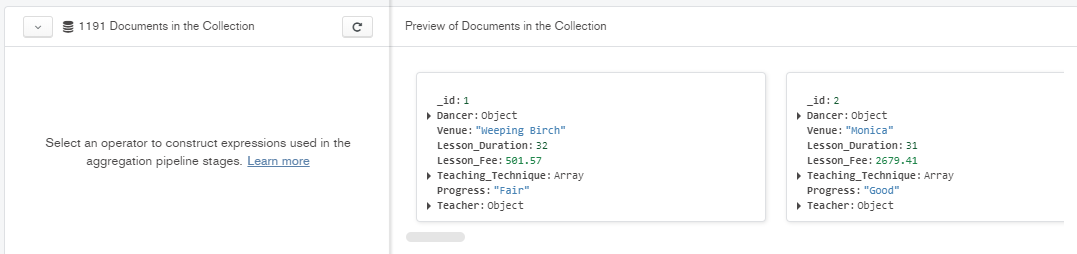
Figure 10

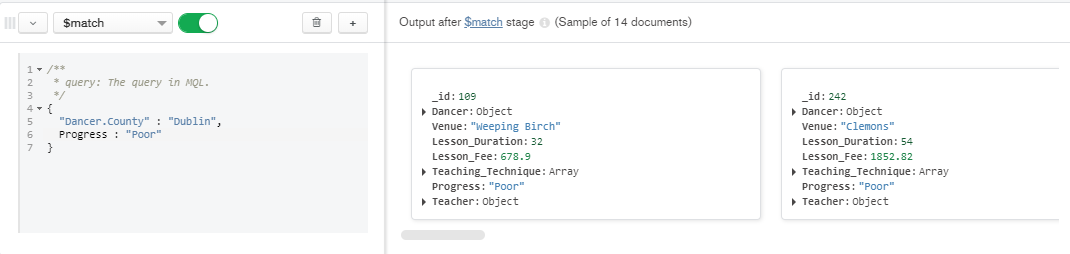


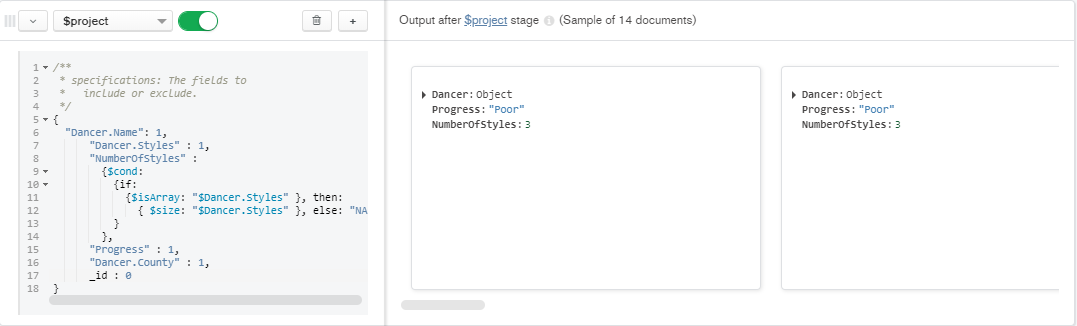
## Evidence of using mongodb compass/atlas with idl data

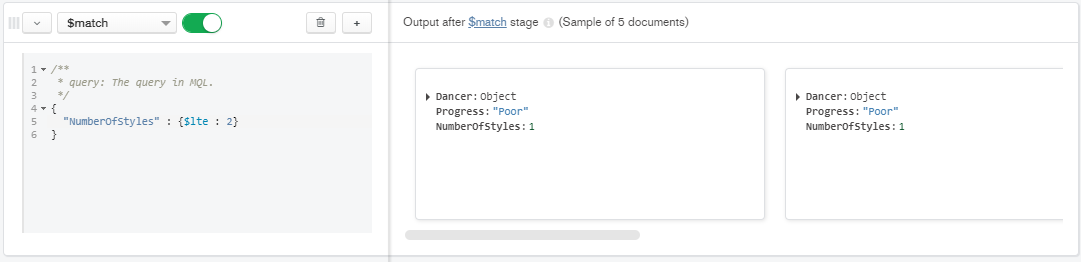
**Query 1 - (Query 12 from Aggregation Framework - Stage 3)**

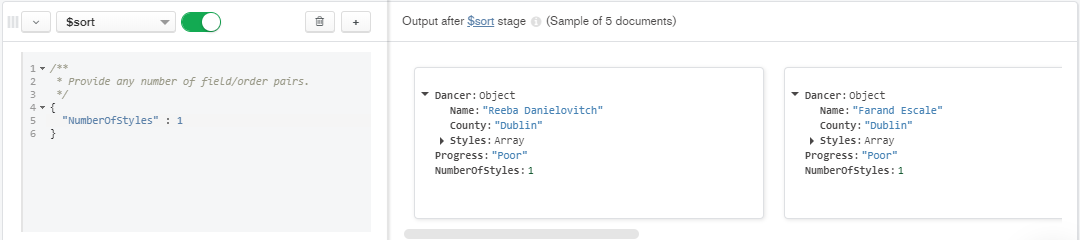
For all the lessons whose dancers are from Dublin county, which dancers have poor progress on their lessons even though they are doing as few dance styles as not more than two? Display them by starting with the dancers doing the least number of dance styles.





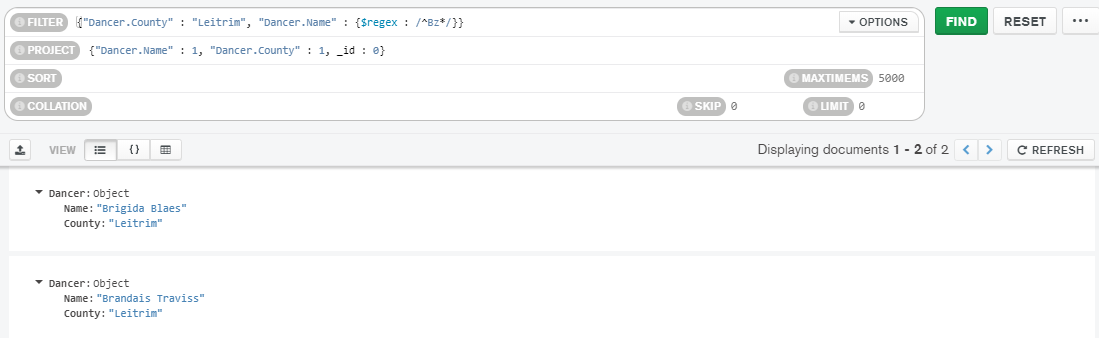






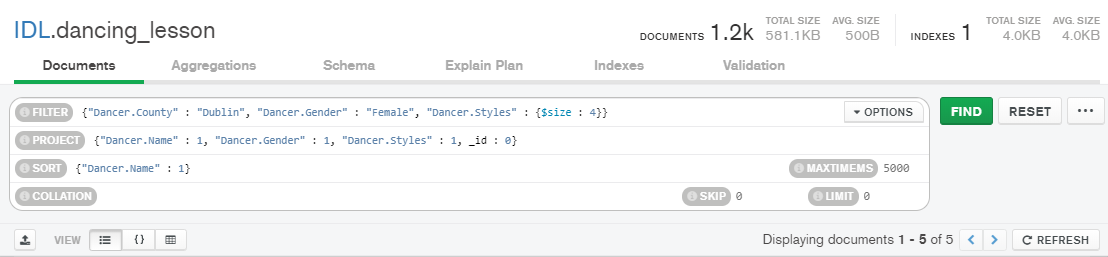
**Query 2 – (Query 21 from MongoFB Shell – Stage 2)**

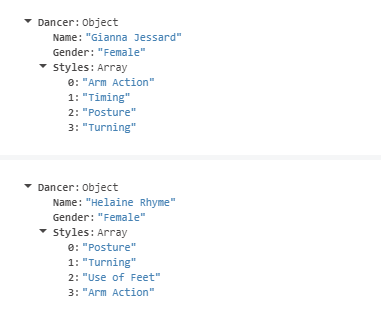
Show all the dancers from Leitrim county whose first names starts with letter B.



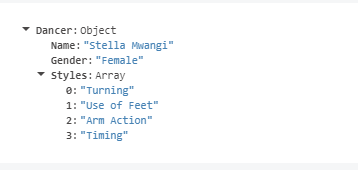
**Query 3 – (Query 18 from Mongo Shell – Stage 2)**

What are the names of female dancers from Dublin county who do exactly all four dancing styles? Sort their names alphabetically



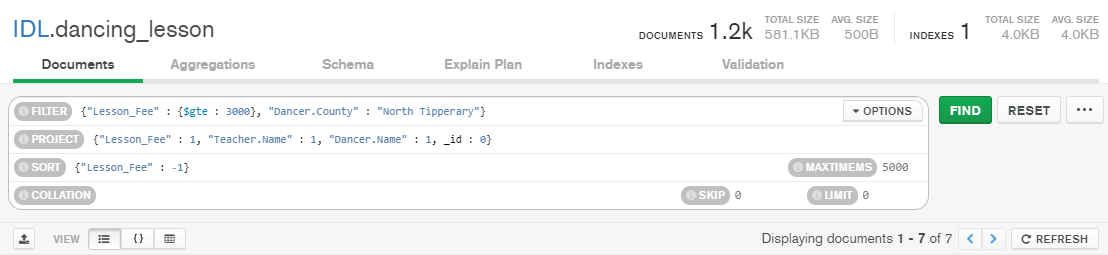


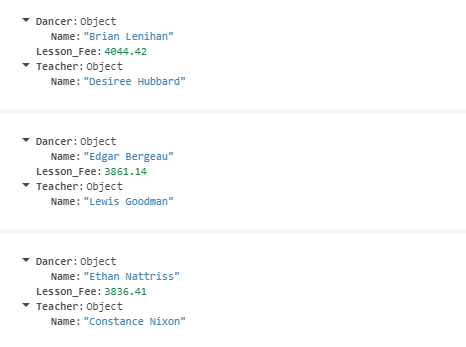




**Query 4 - (Query 19 from Mongo Shell – Stage 2)**

Show the details of all the lessons which are being held in North Tipperary whose fee is at least 3000 Euros. Limit the lesson details to name of the dancer and the teacher teaching the lesson. List the lessons with highest fee first.



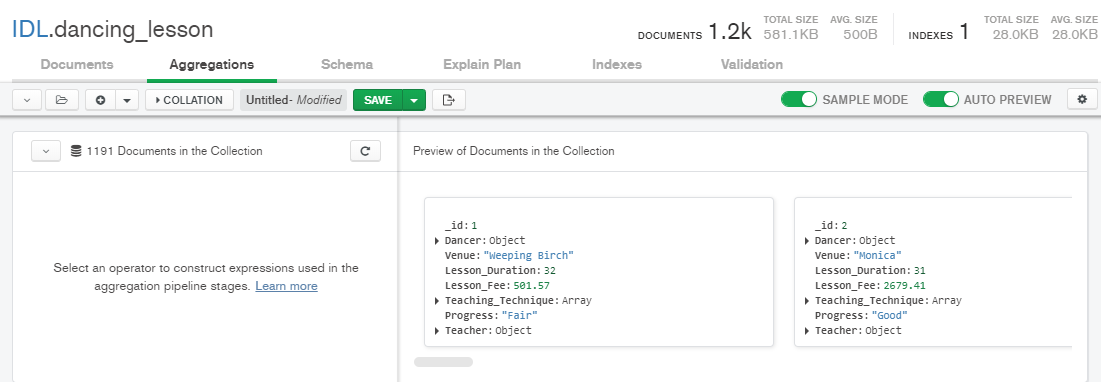


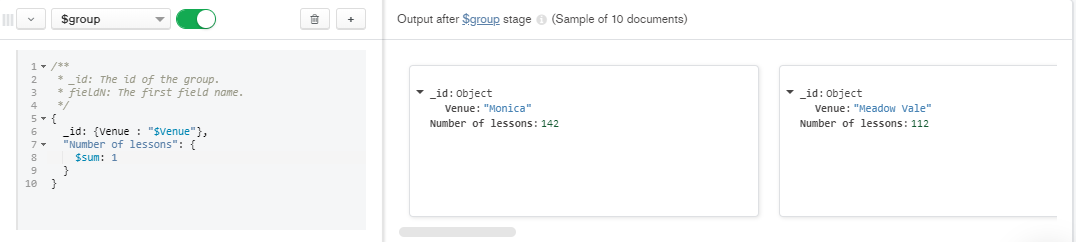




**Query 5 - (Query 2 from Mongo Aggregation Framework – Stage 3)**

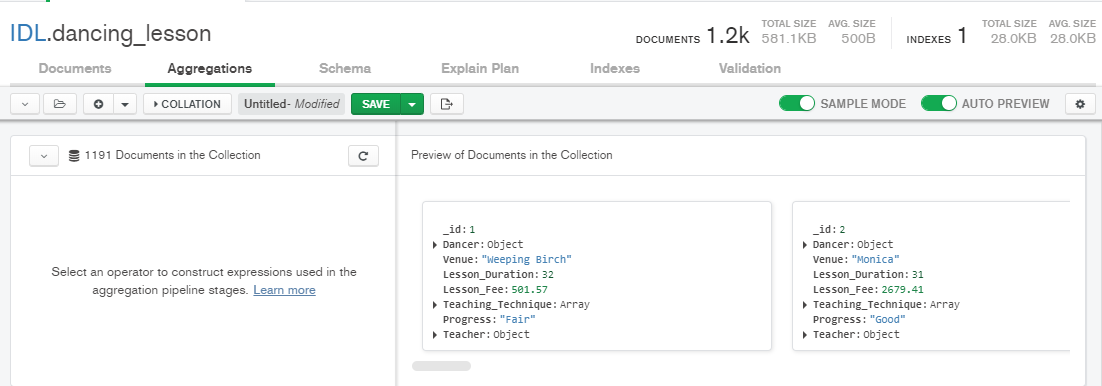
Show the distribution of lessons among available venues. Display them by starting with the venues hosting the largest number of lessons to the ones with the least number of lessons

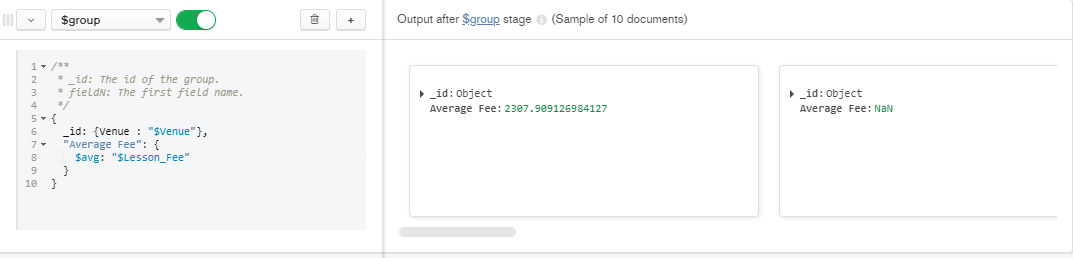


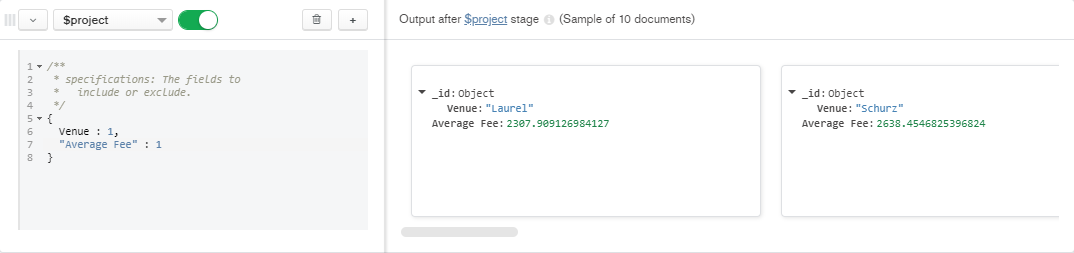


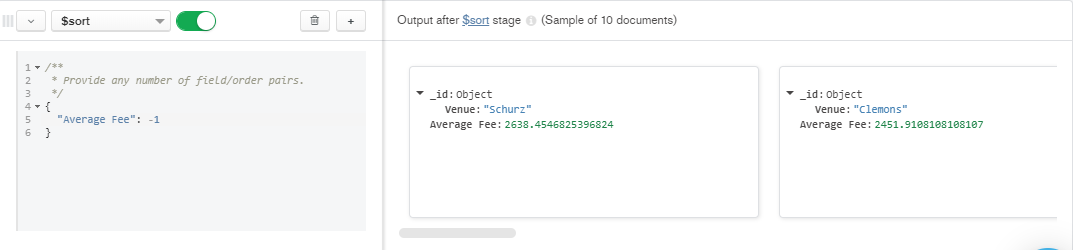
**Query 6 - (Query 9 from Mongo Aggregation Framework – Stage 3)**

What is the average lesson fee associated with each venue?









# APPENDIX

[Screencast1 – Aggregation Framework](https://youtu.be/4bca14l3iXw)

[Screencast2 – Aggregation Framework/Compass/Atlas](https://youtu.be/-ActbeU6KHY)

# REFERENCE

[1] MongoDB, Inc. (2020) *The Database for modern applications*. MongoDB Inc, Available at: <https://www.mongodb.com/> [Accessed 03 May 2020].

[2] ObjectRocket. (2019) *MongoDB Group by Multiple Fields Using Aggregation Function*. Rackspace Company, Available at: <https://kb.objectrocket.com/mongo-db/mongodb-group-by-multiple-fields-using-aggregation-function-464> [Accessed 08 April 2020].

[3] Studio 3T. (2020) *The Beginners Guide to MongoDB Aggregation*. Studio 3T, Available at: <https://studio3t.com/knowledge-base/articles/mongodb-aggregation-framework/> [Accessed 17 April 2020].