Achal Gupta

MSCDA | Advanced Databases | A00258772

10 May 2019

Abstract

Implementation of core CRUD MongoDB queries and aggregation pipeline.  
Introduction to MongoDB Compass and MongoDB Atlas cluster.

Private SWIMMING LESSON

MongoDB

Table of Contents

[Stage 1 - Populate data 2](#_Toc8242790)

[Stage 2 – CRUD Queries 3](#_Toc8242791)

[Stage 3 – Aggregation Pipeline 17](#_Toc8242792)

[Stage 4 – MongoDB tools 25](#_Toc8242793)

[Compass 25](#_Toc8242794)

[Step 1 - Download 25](#_Toc8242795)

[Step 2 – Run and Connect 25](#_Toc8242796)

[Step 3 – Documents Tab (CRUD) 28](#_Toc8242797)

[Step 4 - Aggregations Tab 31](#_Toc8242798)

[Step 5 - Schema Tab 33](#_Toc8242799)

[Step 6 – Performance Optimization using Compass 37](#_Toc8242800)

[Atlas 40](#_Toc8242801)

[Step 1 - Create and Configure Atlas Cluster 40](#_Toc8242802)

[Step 2 - Connect Atlas Cluster with Mongo Compass 48](#_Toc8242803)

[Step 3 - Connecting to Atlas cluster with Mongo Shell and loading PSL database. 51](#_Toc8242804)

[Step 4 - Example Atlas Cluster with Mongo Shell 54](#_Toc8242805)

[Step 5 - Visualizing PSL database in Compass and Atlas Cluster. 54](#_Toc8242806)

[Screencast Links 58](#_Toc8242807)

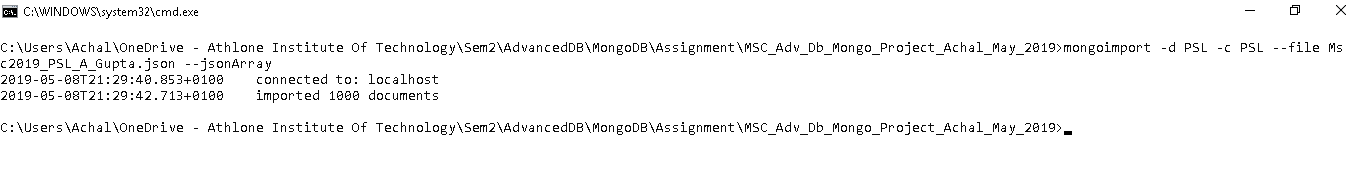
[Screencast 1 – Aggregation Pipeline in Mongo DB 58](#_Toc8242808)

[Screencast 2 – Running MongoDB Atlas cluster on Mongo DB compass and Mongo DB shell. 58](#_Toc8242809)

# Stage 1 - Populate data

Run the following command in command prompt.

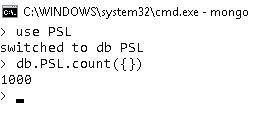
mongoimport **-**d PSL **-**c PSL **--**file Msc2019\_PSL\_A\_Gupta**.**json **--**jsonArray



Verify the document count.

Use PSL

db**.**PSL**.**count**({})**



# Stage 2 – CRUD Queries

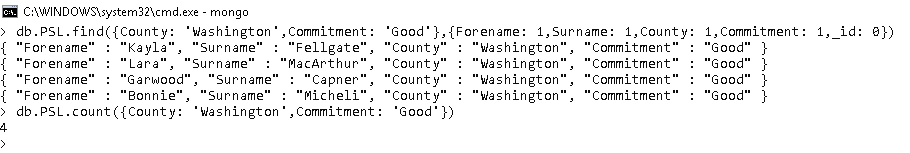
Q1) List the First Name, Last Name,County and Commitment of all the swimmers who lives in 'Washington' County and have 'Good' commitment towards the swimming lessons.

cls

db**.**PSL**.**find**({**County**:** 'Washington'**,**Commitment**:** 'Good'**},{**Forename**:** 1**,**Surname**:** 1**,**County**:** 1**,**Commitment**:** 1**,**\_id**:** 0**})**

db**.**PSL**.**count**({**County**:** 'Washington'**,**Commitment**:** 'Good'**})**

**---------------------------------**



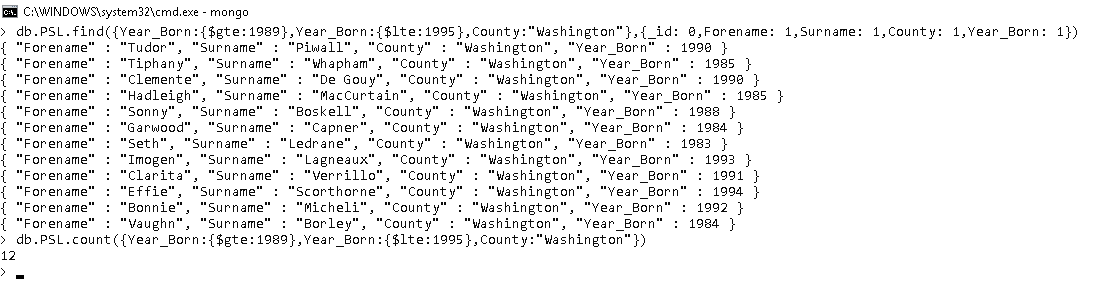
Q2) List ForeName, Surname, County and Birth year of swimmers who are in County Washington county and Born between 1989 and 1995 including.

cls

db**.**PSL**.**find**({**Year\_Born**:{**$gte**:**1989**},**Year\_Born**:{**$lte**:**1995**},**County**:**"Washington"**},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**County**:** 1**,**Year\_Born**:** 1**})**

db**.**PSL**.**count**({**Year\_Born**:{**$gte**:**1989**},**Year\_Born**:{**$lte**:**1995**},**County**:**"Washington"**})**

**---------------------------------**



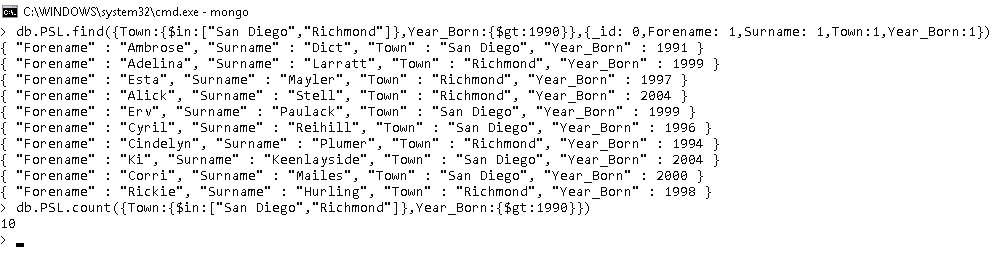
Q3) Give ForeName, Surname, Town and birth year of all the swimmers who are in either Town of San Diego or Richmond and are born after 1990.

cls

db**.**PSL**.**find**({**Town**:{**$in**:[**"San Diego"**,**"Richmond"**]},**Year\_Born**:{**$gt**:**1990**}},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Town**:**1**,**Year\_Born**:**1**})**

db**.**PSL**.**count**({**Town**:{**$in**:[**"San Diego"**,**"Richmond"**]},**Year\_Born**:{**$gt**:**1990**}})**

**---------------------------------**



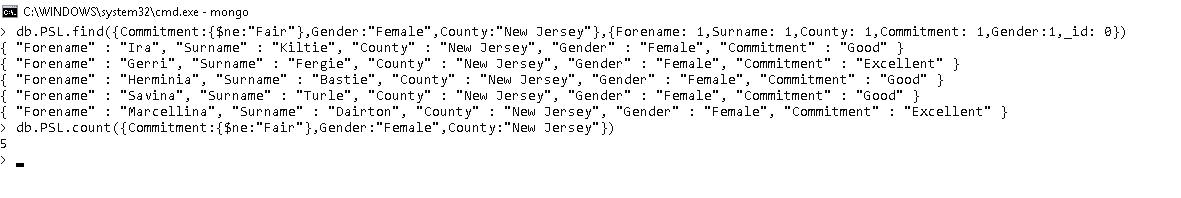
Q4) List Forename, Surname, County, Commitment and Gender of all Female swimmers who have commitment towards lesson other than Fair and are in County San Francisco

cls

db**.**PSL**.**find**({**Commitment**:{**$ne**:**"Fair"**},**Gender**:**"Female"**,**County**:**"New Jersey"**},{**Forename**:** 1**,**Surname**:** 1**,**County**:** 1**,**Commitment**:** 1**,**Gender**:**1**,**\_id**:** 0**})**

db**.**PSL**.**count**({**Commitment**:{**$ne**:**"Fair"**},**Gender**:**"Female"**,**County**:**"New Jersey"**})**

**---------------------------------**



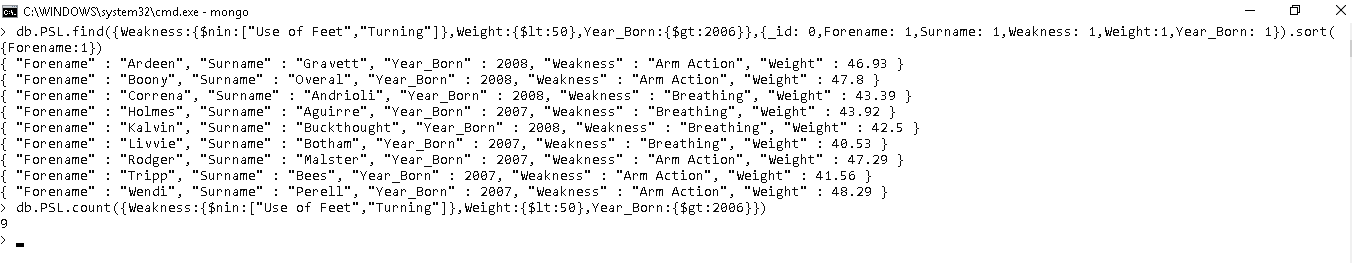
Q5) List Forename, Surname, Weakness, Weight and birth year of all swimmers who are neither weak in Turning nor in use of feet and have weight of less than 50 Kg and were born after 2006. Sort the results by ascending order of Forename of swimmers.

cls

db**.**PSL**.**find**({**Weakness**:{**$nin**:[**"Use of Feet"**,**"Turning"**]},**Weight**:{**$lt**:**50**},**Year\_Born**:{**$gt**:**2006**}},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Weakness**:** 1**,**Weight**:**1**,**Year\_Born**:** 1**}).**sort**({**Forename**:**1**})**

db**.**PSL**.**count**({**Weakness**:{**$nin**:[**"Use of Feet"**,**"Turning"**]},**Weight**:{**$lt**:**50**},**Year\_Born**:{**$gt**:**2006**}})**

**---------------------------------**



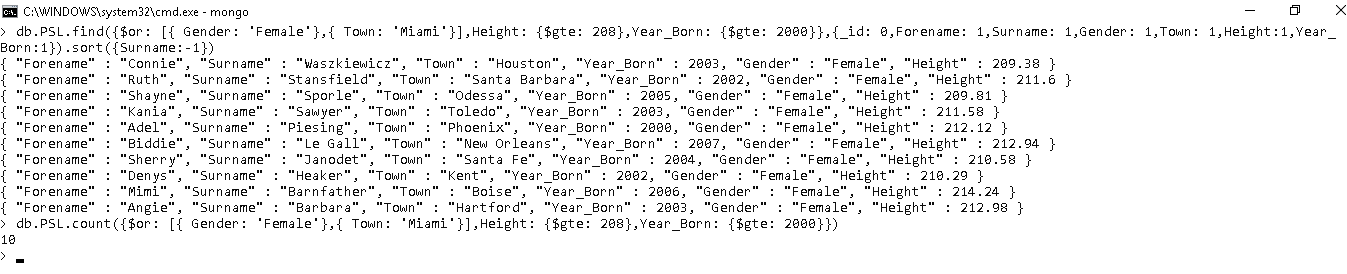
Q6) List Forename, Surname, Gender, Town, Height and birth year of swimmers who are either Female or are in town of Miami also must be greater than 208cm and were born on or after 2000. Sort the results by Surname of swimmer in descending order.

cls

db.PSL.find**({**$or**:** **[{** Gender**:** 'Female'**},{** Town**:** 'Miami'**}],**Height**:** **{**$gte**:** 208**},**Year\_Born**:** **{**$gte**:** 2000**}},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Gender**:** 1**,**Town**:** 1**,**Height**:**1**,**Year\_Born**:**1**}).**sort**({**Surname**:-**1**})**

db**.**PSL**.**count**({**$or**:** **[{** Gender**:** 'Female'**},{** Town**:** 'Miami'**}],**Height**:** **{**$gte**:** 208**},**Year\_Born**:** **{**$gte**:** 2000**}})**

**---------------------------------**



Q7) List Forename, Surname, Address and Town of swimmers located in Denver town and whose address does not contain word "drive" irrespective of case. Sort the results by Surname of swimmer in descending order.

cls

db.PSL.find**({**Address**:{**$not**:{**$regex**:**/drive/i**}},**Town**:**"Denver"**},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Address**:** 1**,**Town**:** 1**}).**sort**({**Surname**:-**1**})**

db**.**PSL**.**count**({**Address**:{**$not**:{**$regex**:**/drive/i**}},**Town**:**"Denver"**})**

**---------------------------------**



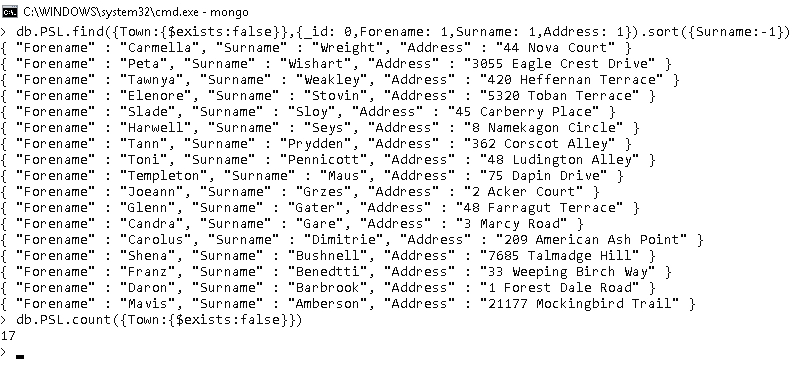
Q8) List Forename, Surname, Address of all swimmers whose town details are not recorded in database. Sort the results in descending order of surname of swimmers.

cls

db.PSL.find**({**Town**:{**$exists**:false}},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Address**:** 1**}).**sort**({**Surname**:-**1**})**

db**.**PSL**.**count**({**Town**:{**$exists**:false}})**

**---------------------------------**



Q9) Display Forename, Surname, Gender, Weight and Height of the swimmers located in County Washington. Calculate Adjusted weight of the swimmer using below logic and make sure Adjusted weight is less than 6 for the swimmers.

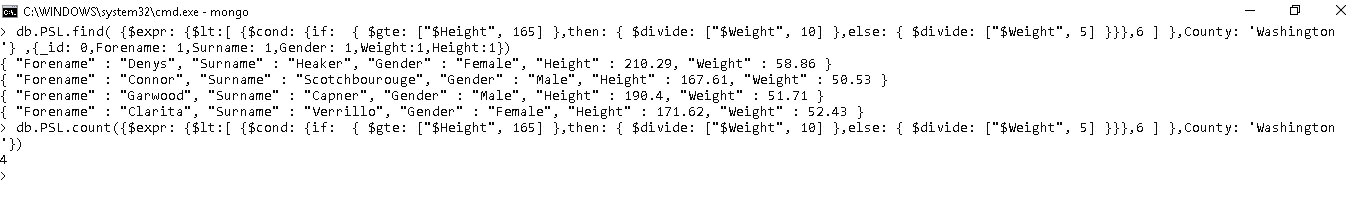
1. If the Height of swimmer is greater than or equal to 165 cm Adjusted weight is calculated by dividing weight by 10
2. Otherwise weight is divided by 5

cls

db.PSL.find**(** **{**$expr**:** **{**$lt**:[** **{**$cond**:** **{**if**:** **{** $gte**:** **[**"$Height"**,** 165**]** **},**then**:** **{** $divide**:** **[**"$Weight"**,** 10**]** **},**else**:** **{** $divide**:** **[**"$Weight"**,** 5**]** **}}},**6 **]** **},**County**:** 'Washington'**}** **,{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Gender**:** 1**,**Weight**:**1**,**Height**:**1**})**

db**.**PSL**.**count**({**$expr**:** **{**$lt**:[** **{**$cond**:** **{**if**:** **{** $gte**:** **[**"$Height"**,** 165**]** **},**then**:** **{** $divide**:** **[**"$Weight"**,** 10**]** **},**else**:** **{** $divide**:** **[**"$Weight"**,** 5**]** **}}},**6 **]** **},**County**:** 'Washington'**})**

**---------------------------------**



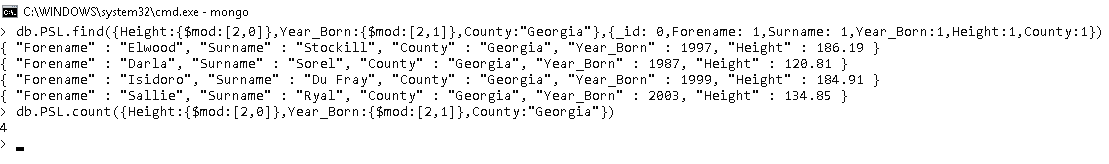
Q10) List Forename, Surname, birth year, height and county of all swimmers who were born in odd years and are located in Georgia.

cls

db.PSL.find**({**Height**:{**$mod**:[**2**,**0**]},**Year\_Born**:{**$mod**:[**2**,**1**]},**County**:**"Georgia"**},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Year\_Born**:**1**,**Height**:**1**,**County**:**1**})**

db**.**PSL**.**count**({**Height**:{**$mod**:[**2**,**0**]},**Year\_Born**:{**$mod**:[**2**,**1**]},**County**:**"Georgia"**})**

**---------------------------------**



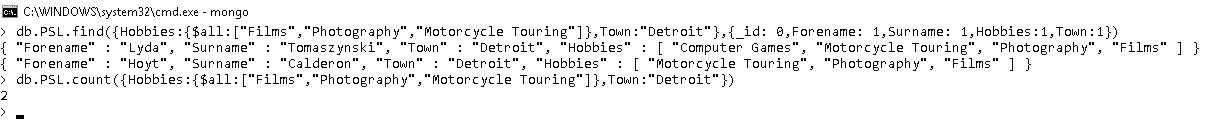
Q11) List Forename, Surname, Hobbies and Town of all the swimmers who are in Town Detroit and have Films, Photography, Motorcycle Touring as their hobbies their position in sequence is not important

cls

db.PSL.find**({**Hobbies**:{**$all**:[**"Films"**,**"Photography"**,**"Motorcycle Touring"**]},**Town**:**"Detroit"**},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Hobbies**:**1**,**Town**:**1**})**

db**.**PSL**.**count**({**Hobbies**:{**$all**:[**"Films"**,**"Photography"**,**"Motorcycle Touring"**]},**Town**:**"Detroit"**})**

**---------------------------------**



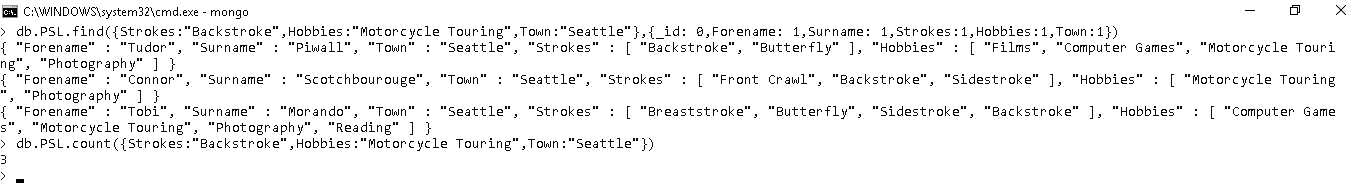
Q12) List Forename, Surname, Strokes, Hobbies and Town of all swimmers located in Seattle town and have atleast Backstroke as a Stoke and Motorcycle touring as Hobby, swimmer can have other values for Strokes and Hobbies also.

cls

db.PSL.find**({**Strokes**:**"Backstroke"**,**Hobbies**:**"Motorcycle Touring"**,**Town**:**"Seattle"**},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Strokes**:**1**,**Hobbies**:**1**,**Town**:**1**})**

db**.**PSL**.**count**({**Strokes**:**"Backstroke"**,**Hobbies**:**"Motorcycle Touring"**,**Town**:**"Seattle"**})**

**---------------------------------**



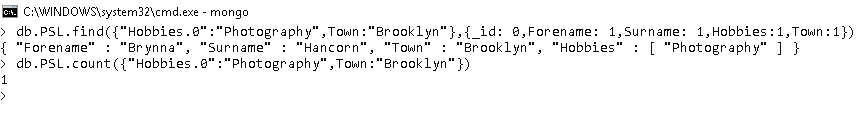
Q13) List Forename, Surname, Hobbies and Town of swimmers residing in the town of Brooklyn and have first hobby as Photography.

cls

db.PSL.find**({**"Hobbies.0"**:**"Photography"**,**Town**:**"Brooklyn"**},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Hobbies**:**1**,**Town**:**1**})**

db**.**PSL**.**count**({**"Hobbies.0"**:**"Photography"**,**Town**:**"Brooklyn"**})**

**---------------------------------**



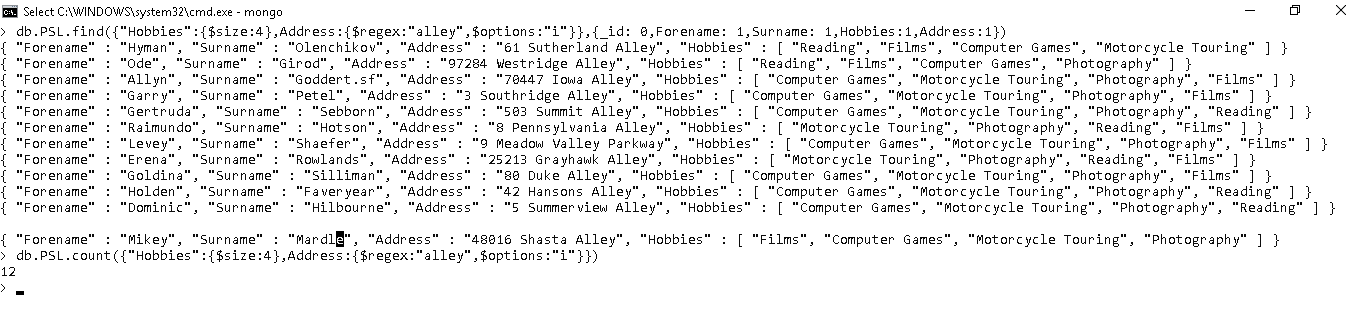
Q14) List Forename, Surname, Hobbies and Address of swimmers whose address contains word 'alley' irrespective of case and have exactly 4 hobbies.

cls

db.PSL.find**({**"Hobbies"**:{**$size**:**4**},**Address**:{**$regex**:**"alley"**,**$options**:**"i"**}},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Hobbies**:**1**,**Address**:**1**})**

db**.**PSL**.**count**({**"Hobbies"**:{**$size**:**4**},**Address**:{**$regex**:**"alley"**,**$options**:**"i"**}})**

**---------------------------------**



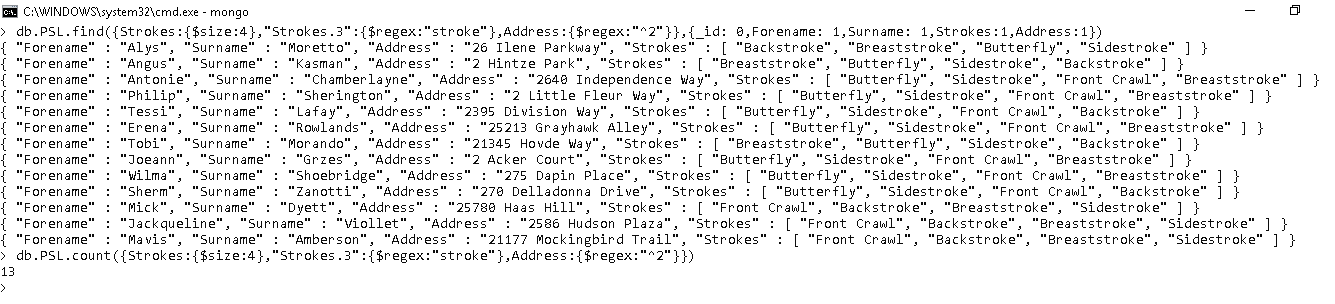
Q15) List Forename, Surname, Strokes and Address of all swimmers who have exactly 4 strokes and name of 4th stroke contains word "stroke". Limit the results to swimmers who have address starting with "2".

cls

db.PSL.find**({**Strokes**:{**$size**:**4**},**"Strokes.3"**:{**$regex**:**"stroke"**},**Address**:{**$regex**:**"^2"**}},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Strokes**:**1**,**Address**:**1**})**

db**.**PSL**.**count**({**Strokes**:{**$size**:**4**},**"Strokes.3"**:{**$regex**:**"stroke"**},**Address**:{**$regex**:**"^2"**}})**

**---------------------------------**



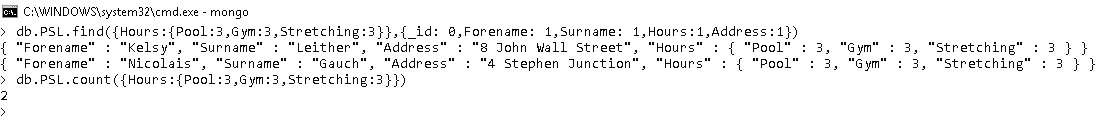
Q16) List Forename, Surname, training hours and Address of all swimmers who spend 3 hours on each Pool, Gym and Stretching.

cls

db.PSL.find**({**Hours**:{**Pool**:**3**,**Gym**:**3**,**Stretching**:**3**}},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Hours**:**1**,**Address**:**1**})**

db**.**PSL**.**count**({**Hours**:{**Pool**:**3**,**Gym**:**3**,**Stretching**:**3**}})**

**---------------------------------**



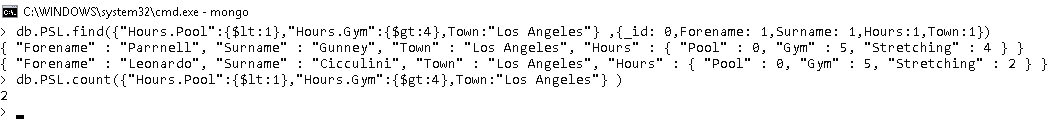
Q17) List Forename, Surname, training hours and Town of all swimmers who spent less than 1 hour training in pool and spent more than 4 hours in gym and are in Los Angeles town.

cls

db.PSL.find**({**"Hours.Pool"**:{**$lt**:**1**},**"Hours.Gym"**:{**$gt**:**4**},**Town**:**"Los Angeles"**}** **,{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Hours**:**1**,**Town**:**1**})**

db**.**PSL**.**count**({**"Hours.Pool"**:{**$lt**:**1**},**"Hours.Gym"**:{**$gt**:**4**},**Town**:**"Los Angeles"**}** **)**

**---------------------------------**



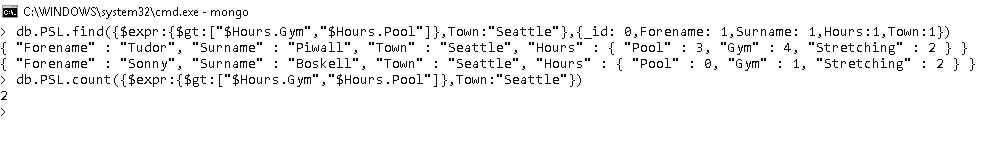
Q18) List Forename, Surname, Training hours and Town of all swimmers who are in town of Seattle and spend more hours training in gym than pool.

cls

db.PSL.find**({**$expr**:{**$gt**:[**"$Hours.Gym"**,**"$Hours.Pool"**]},**Town**:**"Seattle"**},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Hours**:**1**,**Town**:**1**})**

db**.**PSL**.**count**({**$expr**:{**$gt**:[**"$Hours.Gym"**,**"$Hours.Pool"**]},**Town**:**"Seattle"**})**

**---------------------------------**



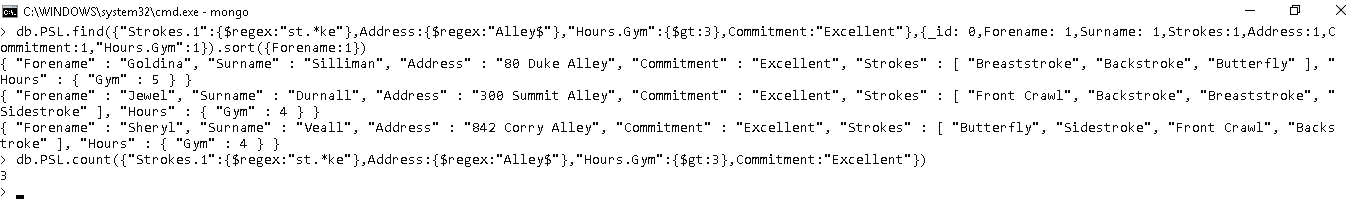
Q19) List Forename, Surname, Strokes, Address, Commitment and hours spent training in Gym of swimmers whose name of second stroke contains letters 'st' followed by 'ke' and Address ends with 'Alley'. Limit the result to swimmers who spent more than 3 hours in Gym and have Excellent Commitment

cls

db.PSL.find**({**"Strokes.1"**:{**$regex**:**"st.\*ke"**},**Address**:{**$regex**:**"Alley$"**},**"Hours.Gym"**:{**$gt**:**3**},**Commitment**:**"Excellent"**},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Strokes**:**1**,**Address**:**1**,**Commitment**:**1,"Hours.Gym":1**}).**sort**({**Forename**:**1**})**

db**.**PSL**.**count**({**"Strokes.1"**:{**$regex**:**"st.\*ke"**},**Address**:{**$regex**:**"Alley$"**},**"Hours.Gym"**:{**$gt**:**3**},**Commitment**:**"Excellent"**})**

**---------------------------------**



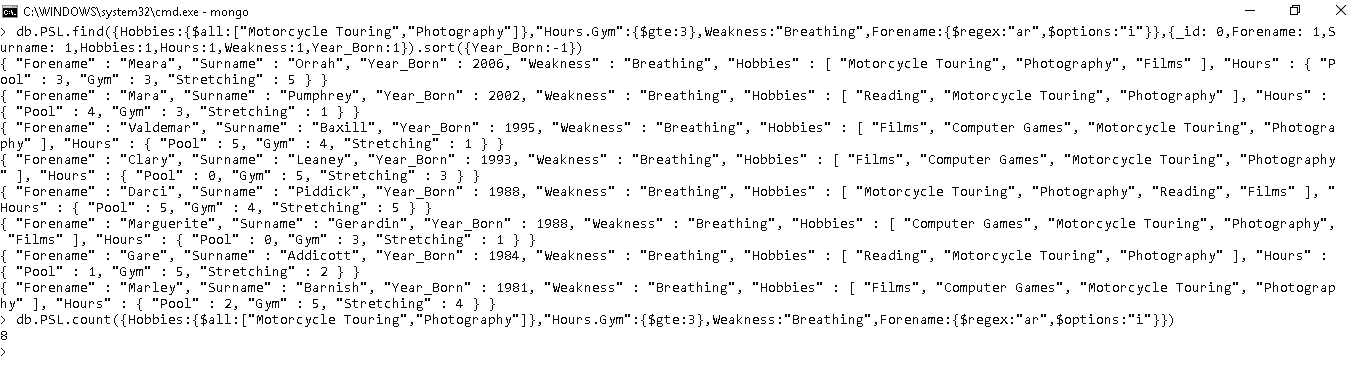
Q20) List Forename, Surname, Hobbies, Training hours, Weakness and birth year of swimmers whose Hobbies include Motorcyle Touring and Photography. Limit rows to swimmers who spent at least 3 hours in Gym training and have Breathing as weakness and forename contains letters 'ar' irrespective of case. Sort the results by birth year in reverse order.

cls

db.PSL.find**({**Hobbies**:{**$all**:[**"Motorcycle Touring"**,**"Photography"**]},**"Hours.Gym"**:{**$gte**:**3**},**Weakness**:**"Breathing"**,**Forename**:{**$regex**:**"ar"**,**$options**:**"i"**}},{**\_id**:** 0**,**Forename**:** 1**,**Surname**:** 1**,**Hobbies**:**1**,**Hours**:**1**,**Weakness**:**1**,**Year\_Born**:**1**}).**sort**({**Year\_Born**:-**1**})**

db**.**PSL**.**count**({**Hobbies**:{**$all**:[**"Motorcycle Touring"**,**"Photography"**]},**"Hours.Gym"**:{**$gte**:**3**},**Weakness**:**"Breathing"**,**Forename**:{**$regex**:**"ar"**,**$options**:**"i"**}})**

**---------------------------------**



Q21) Insert a new record in database with only 7 Key-Value pairs as given below.

Forename – Achal

Surname – Gupta

Town – Athlone

Gender – Male

Hours in Gym – 1

Hours in Pool – 4

Hobbies – Motorcycle Touring and Photography.

cls

try **{**

db**.**PSL**.**insertOne**(** **{** "Forename"**:** "Achal"**,**"Surname"**:**"Gupta" **,**"Town"**:** "Athlone"**,** "Gender"**:**"Male"**,**"Hours"**:{**"Gym"**:** 1**,** "Pool"**:**4**},** "Hobbies"**:[**"Motorcyle Touring"**,**"Photography"**]}** **);**

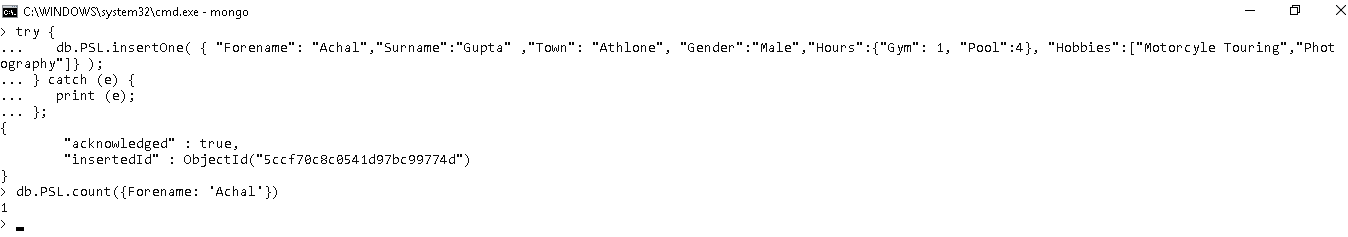
**}** catch **(**e**)** **{**

print **(**e**);**

**};**

db**.**PSL**.**count**({**Forename**:** 'Achal'**})**

**--------------------------------------------**



Q22) Insert one record in database with all the details.

cls

try **{**

db**.**PSL**.**insertOne**(** **{** "Forename" **:** "Anup"**,** "Surname" **:** "Singh"**,** "Address" **:** "4 Retreat Mews"**,** "Town" **:** "Athlone"**,** "County" **:** "Westmeath" **,** "Year\_Born" **:** 1992**,** "Gender" **:** "Male"**,** "Commitment" **:** "Fair"**,** "Weakness" **:** "Turning"**,** "Height" **:** 162.51**,** "Weight" **:** 67.43**,** "Strokes" **:** **[**"Front Crawl"**,** "Breaststroke"**,** "Sidestroke"**]** **,** "Hobbies" **:** **[**"Reading"**,** "Computer Games"**]** **,** "Hours" **:** **{**"Pool" **:**1**,**"Gym" **:** 5**,**"Stretching" **:** 3**}** **}**

**);**

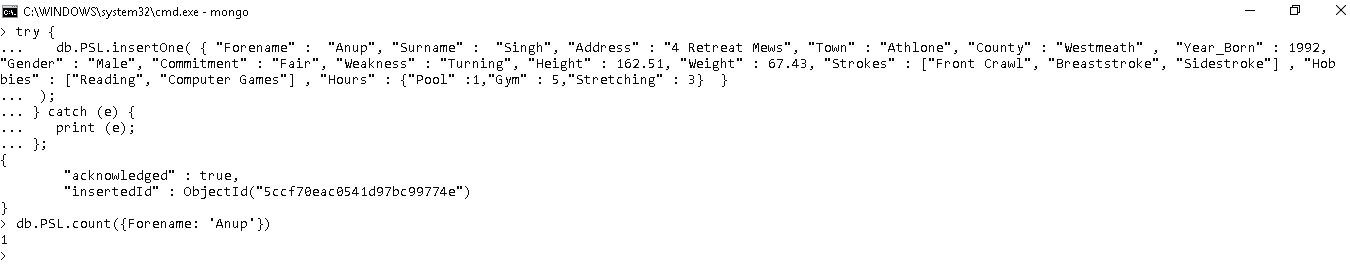
**}** catch **(**e**)** **{**

print **(**e**);**

**};**

db**.**PSL**.**count**({**Forename**:** 'Anup'**})**

**--------------------------------------------**



Q23) Insert more than one record in database with all the details.

cls

try **{**

db**.**PSL**.**insertMany**(** **[**

**{** "Forename" **:** "Reenu"**,** "Surname" **:** "Rathi"**,** "Address" **:** "63 Croi Oige"**,** "Town" **:** "Athlone"**,** "County" **:** "Westmeath" **,** "Year\_Born" **:** 1988**,** "Gender" **:** "Female"**,** "Commitment" **:** "Excellent"**,** "Weakness" **:** "Use of Feet"**,** "Height" **:** 186.28**,** "Weight" **:** 47.33**,** "Strokes" **:** **[**"Front Crawl"**]** **,** "Hobbies" **:** **[**"Reading"**,** "Films"**,** "Motorcycle Touring"**]** **,** "Hours" **:** **{**"Pool" **:**4**,**"Gym" **:** 3**,**"Stretching" **:** 4**}** **}** **,**

**{** "Forename" **:** "Radha"**,** "Surname" **:** "Singh"**,** "Gender" **:** "Female"**,** "Commitment" **:** "Excellent"**,**"Town" **:** "Athlone"**,** "County" **:** "Westmeath", "Weakness" **:** "Use of Feet"**,** "Height" **:** 201.7**,** "Weight" **:** 42.99**,** "Strokes" **:** **[**"Breaststroke"**,** "Butterfly"**]** **,** "Hobbies" **:** **[**"Reading"**,** "Films"**]** **,** "Hours" **:** **{**"Pool" **:**3**,**"Gym" **:** 3**,**"Stretching" **:** 1**}** **}**

**]);**

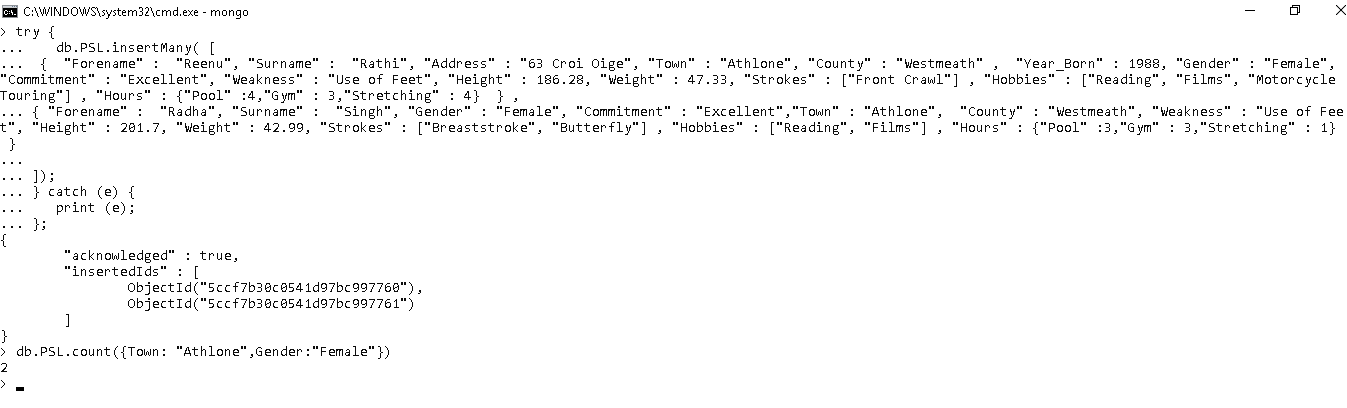
**}** catch **(**e**)** **{**

print **(**e**);**

**};**

db**.**PSL**.**count**({**Town**:** "Athlone"**,**Gender**:**"Female"**})**

**--------------------------------------------**



Q24) For swimmers with Forename "Achal" update town to Dublin and training hours in gym to 2 hours.

cls

db.PSL.find**({**Forename**:**"Achal"**});**

db**.**PSL**.**updateOne**(**

**{** "Forename"**:** "Achal" **},**

**{**

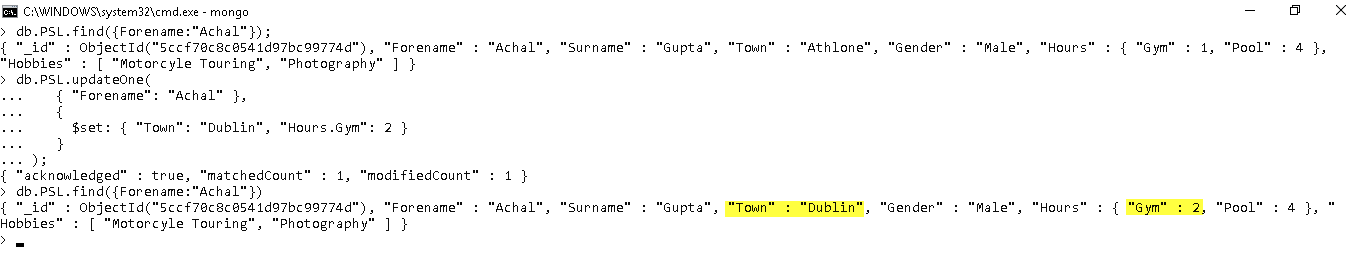
$set**:** **{** "Town"**:** "Dublin"**,** "Hours.Gym"**:** 2 **}**

**}**

**);**

db**.**PSL**.**find**({**Forename**:**"Achal"**})**

**--------------------------------------------**



Q25) Update training hours spend on pool to 0 for all female swimmers who are in county starting with word "West" and located in Town of Athlone or Dublin.

cls

db.PSL.find**({**County**:{**$regex**:**/^West/**},** Gender**:** "Female"**,**Town**:{**$in**:[**"Athlone"**,**"Dublin"**]}},{**Forename**:**1**,** Surname**:**1**,** Gender**:**1**,** Town**:**1**,** "Hours.Pool"**:**1**,**\_id**:**0**})**

db**.**PSL**.**updateMany**(**

**{**County**:{**$regex**:**/^West/**},** Gender**:** "Female"**,**Town**:{**$in**:[**"Athlone"**,**"Dublin"**]}},**

**{**

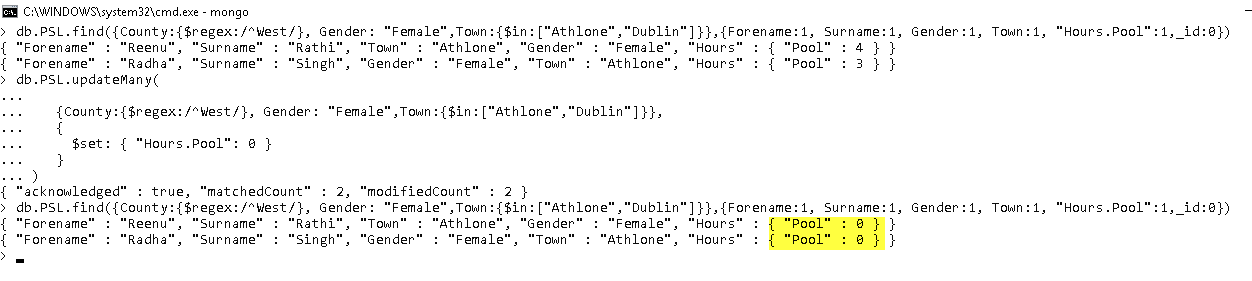
$set**:** **{** "Hours.Pool"**:** 0 **}**

**}**

**)**

db**.**PSL**.**find**({**County**:{**$regex**:**/^West/**},** Gender**:** "Female"**,**Town**:{**$in**:[**"Athlone"**,**"Dublin"**]}},{**Forename**:**1**,** Surname**:**1**,** Gender**:**1**,** Town**:**1**,** "Hours.Pool"**:**1**,**\_id**:**0**})**

**--------------------------------------------**



Q26) Replace the record for swimmer named Becky with following information. If the earlier record does not exists create the new entry.

County – Westmeath

Year\_Born – 1980

Hours in Stretching – 0

Strokes – Front Crawl

cls

db.PSL.find**({**Forename**:**"Becky"**});**

try **{**

db**.**PSL**.**replaceOne**(**

**{**"Forename" **:** "Becky"**},**

**{**"Forename"**:** "Becky"**,** "County" **:** "Westmeath"**,** "Year\_Born"**:** "1980"**,** "Hours" **:{**"Stretching"**:**0**},**"Strokes"**:[**"Front Crawl"**]},**

**{**upsert**:true}**

**);**

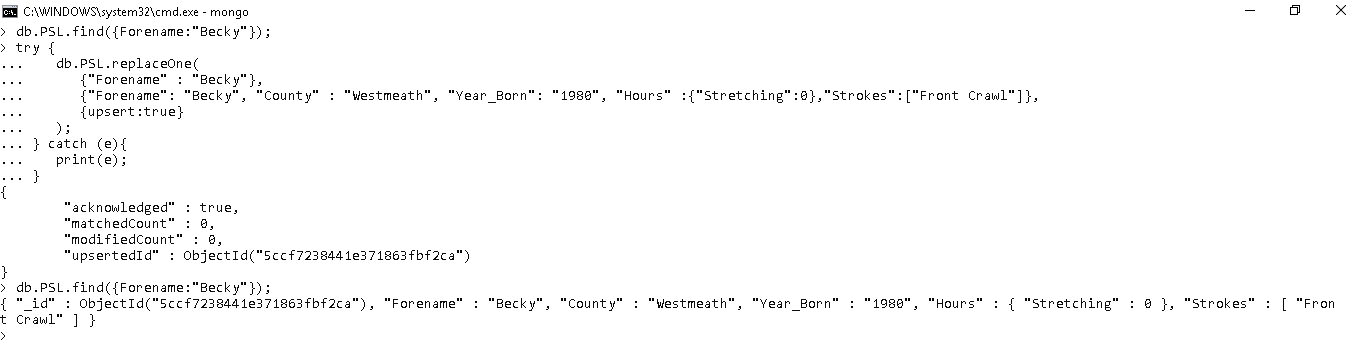
**}** catch **(**e**){**

print**(**e**);**

**}**

db**.**PSL**.**find**({**Forename**:**"Becky"**});**

**--------------------------------------------**



Q27) Increase Height by 2cm, decrease weight by 10 kg and increase number of training hours spend in pool by 1 for all swimmers whose last name is "Singh".

cls

db.PSL.find**({**Surname**:**"Singh",Gender:"Male"**},{**Surname**:**1**,**Height**:**1**,**Weight**:**1**,**Hours**:**1**});**

db**.**PSL**.**update**(**

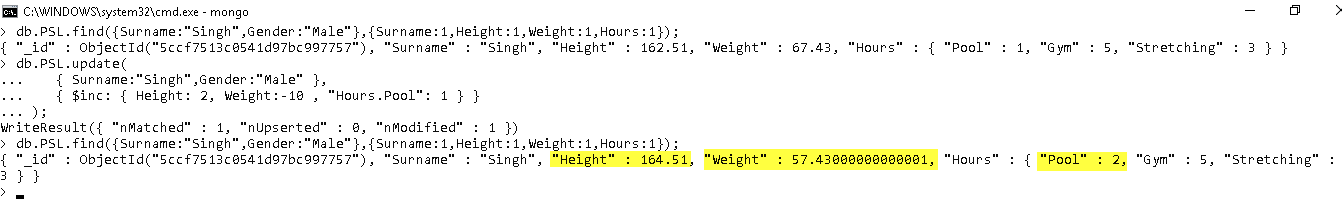
**{** Surname**:**"Singh",Gender:"Male" **},**

**{** $inc**:** **{** Height**:** 2**,** Weight**:-**10 **,** "Hours.Pool"**:** 1 **}** **}**

**);**

db**.**PSL**.**find**({**Surname**:**"Singh",Gender:"Male"**},{**Surname**:**1**,**Height**:**1**,**Weight**:**1**,**Hours**:**1**});**

**--------------------------------------------**



Q28) Add "Reading", "Films" and "Photography" to the Hobbies of swimmer Achal. Make sure if any of the above Hobby is already recorded in database it should not be repeated twice in the list.

cls

db.PSL.find**({**Forename**:**"Achal"**});**

db**.**PSL**.**update**(**

**{** "Forename"**:** "Achal" **},**

**{**

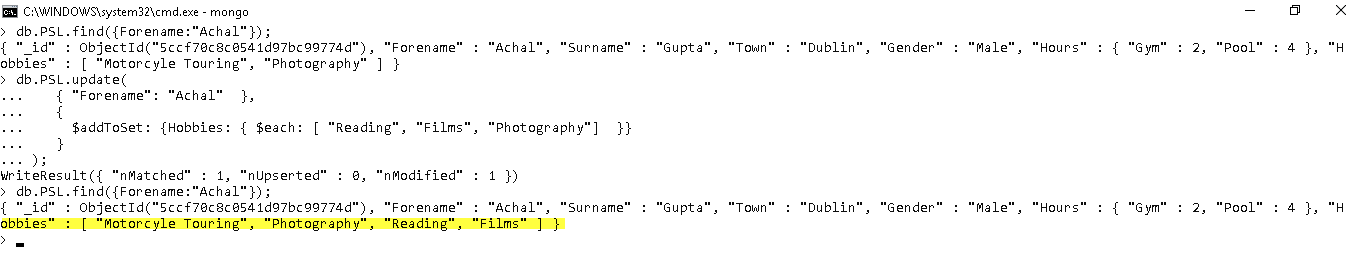
$addToSet**:** **{**Hobbies**:** **{** $each**:** **[** "Reading"**,** "Films"**,** "Photography"**]** **}}**

**}**

**);**

db**.**PSL**.**find**({**Forename**:**"Achal"**});**

**--------------------------------------------**



Q29) Delete only first occurrence of record of female swimmer who is in county starting with word "West" and located in Town of Athlone or Dublin.

cls

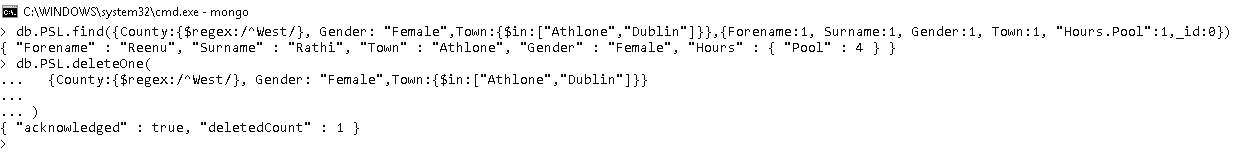
db.PSL.find**({**County**:{**$regex**:**/^West/**},** Gender**:** "Female"**,**Town**:{**$in**:[**"Athlone"**,**"Dublin"**]}},{**Forename**:**1**,** Surname**:**1**,** Gender**:**1**,** Town**:**1**,** "Hours.Pool"**:**1**,**\_id**:**0**})**

db**.**PSL**.**deleteOne**(**

**{**County**:{**$regex**:**/^West/**},** Gender**:** "Female"**,**Town**:{**$in**:[**"Athlone"**,**"Dublin"**]}}**

**)**

**--------------------------------------------**



Q30) Delete records of all swimmers who are located in County Westmeath or are located in Town of Athlone.

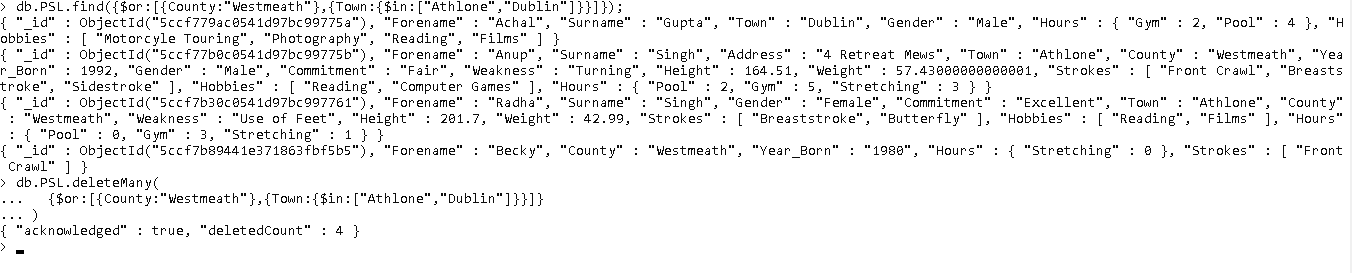
db**.**PSL**.**find**({**$or**:[{**County**:**"Westmeath"**},{**Town**:{**$in**:[**"Athlone"**,**"Dublin"**]}}]});**

db**.**PSL**.**deleteMany**(**

**{**$or**:[{**County**:**"Westmeath"**},{**Town**:{**$in**:[**"Athlone"**,**"Dublin"**]}}]}**

**)**

**--------------------------------------------**



# Stage 3 – Aggregation Pipeline

Q1) List the number of swimmers grouped on basis of their commitment. Exclude swimmers whose commitment is not recorded in database.

cls

db**.**PSL**.**aggregate**([**

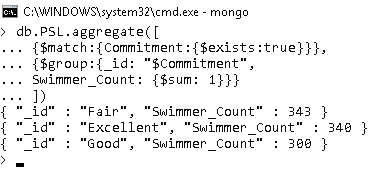
**{**$match**:{**Commitment**:{**$exists**:true}}},**

**{**$group**:{**\_id**:** "$Commitment"**,**

Swimmer\_Count**:** **{**$sum**:** 1**}}}**

**])**

**--------------------------------**



Q2) Divide swimmers in 3 categories based on their birth year and Count number of swimmers in each category.

cls

db**.**PSL**.**aggregate**([**

**{**$bucket**:{**

groupBy**:** "$Year\_Born"**,**

boundaries**:** **[**1980**,**1990**,**2000**,**2010**],**

default**:** "Other"**,**

output**:** **{**

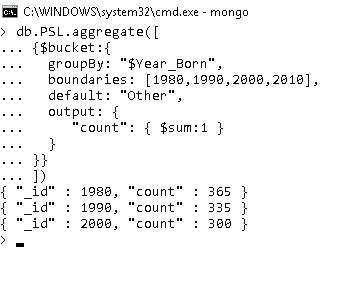
"count"**:** **{** $sum**:**1 **}**

**}**

**}}**

**])**

**--------------------------------**



Q3) Display Forename, Surname, Total hours spent training in Pool, Gym and Stretching for all swimmers located in County Hawaii.

cls

db**.**PSL**.**aggregate**([**

**{**$match**:{**

County**:**"Hawaii"

**}},**

**{**$addFields**:{**

"Hours.Total"**:** **{**$add**:[**"$Hours.Gym"**,**"$Hours.Pool"**,**"$Hours.Stretching"**]}**

**}},**

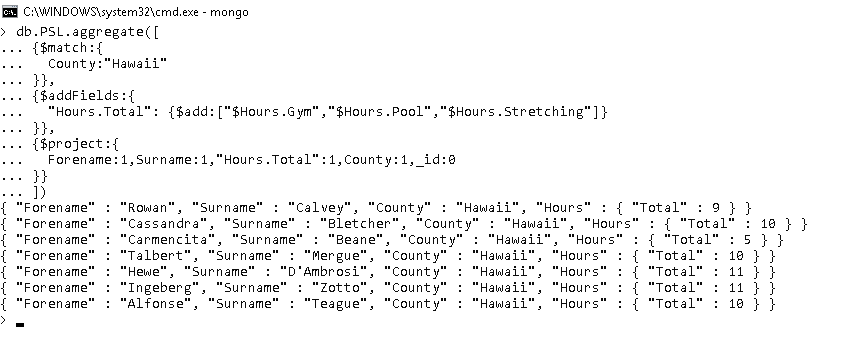
**{**$project**:{**

Forename**:**1**,**Surname**:**1**,**"Hours.Total"**:**1**,**County**:**1**,**\_id**:**0

**}}**

**])**

**--------------------------------**



Q4) Display count of swimmers for each of the Hobby. Also Divide height of swimmers into 5 buckets and calculate number of swimmers belonging to each bucket.

cls

db**.**PSL**.**aggregate**([**

**{**$facet**:{**

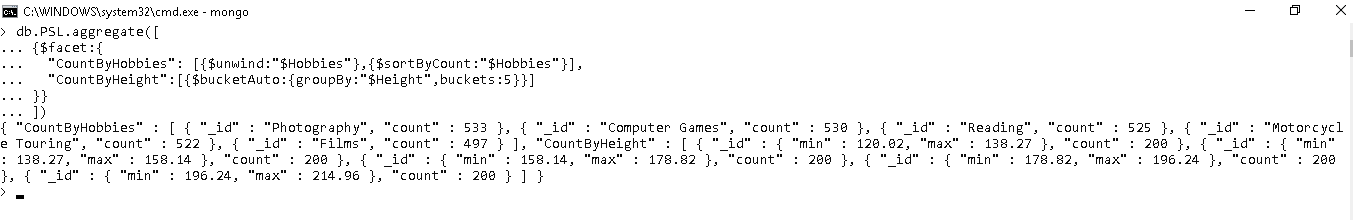
"CountByHobbies"**:** **[{**$unwind**:**"$Hobbies"**},{**$sortByCount**:**"$Hobbies"**}],**

"CountByHeight"**:[{**$bucketAuto**:{**groupBy**:**"$Height"**,**buckets**:**5**}}]**

**}}**

**])**

**--------------------------------**



Q5) Show details of all indexes created on PSL collection.

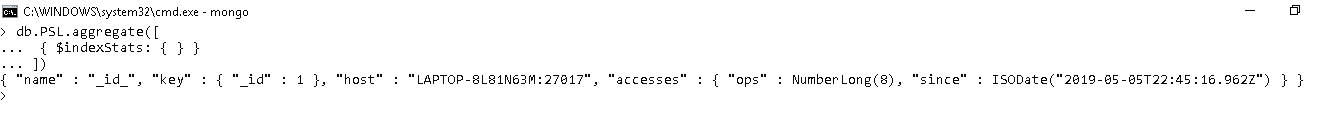
cls

db**.**PSL**.**aggregate**([**

**{** $indexStats**:** **{** **}** **}**

**])**

**--------------------------------**



Q6) List Forename, Surname, Height, County and Weakness of all swimmers whose height is same as the minimum height of swimmers located at Washington and have a weakness "Use of Feet".

cls

db**.**PSL**.**aggregate**([**

**{**$match**:{**County**:**"Washington"**}},**

**{**$group**:{**\_id**:** "$Weakness"**,**Min\_Height**:** **{**$min**:** "$Height"**}}},**

**{**$match**:{**\_id**:**"Use of Feet"**}},**

**{**$lookup**:{**

from**:** 'PSL'**,**

localField**:** 'Min\_Height'**,**

foreignField**:** 'Height'**,**

as**:** 'Swimmers\_MinHeight'

**}},**

**{**$project**:{**

"Swimmers\_MinHeight.Forename"**:**1**,**

"Swimmers\_MinHeight.Surname"**:**1**,**

"Swimmers\_MinHeight.Height"**:**1**,**

"Swimmers\_MinHeight.County"**:**1**,**

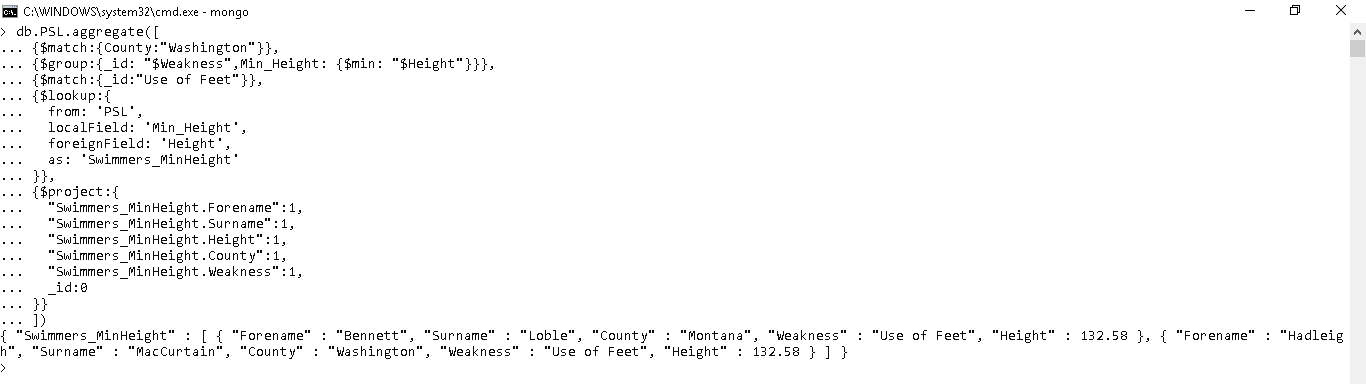
"Swimmers\_MinHeight.Weakness"**:**1**,**

\_id**:**0

**}}**

**])**

**--------------------------------**



Q7) Calculate Minimum, maximum, average and standard deviation of height categorized by weakness of swimmers.

cls

db**.**PSL**.**aggregate**([**

**{**$match**:{**

Weakness**:{**$exists**:true}**

**}},**

**{**$group**:{**

\_id**:** "$Weakness"**,**

Min\_Height**:** **{**$min**:**"$Height"**},**

Max\_Height**:** **{**$max**:**"$Height"**},**

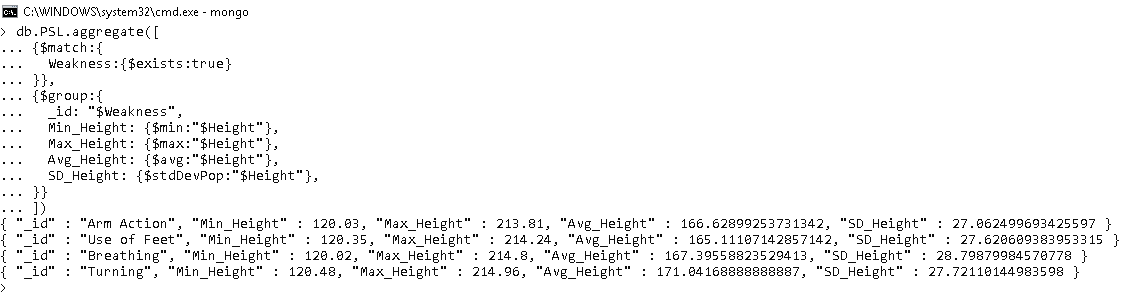
Avg\_Height**:** **{**$avg**:**"$Height"**},**

SD\_Height**:** **{**$stdDevPop**:**"$Height"**},**

**}}**

**])**

**--------------------------------**



Q8) List the most popular hobby and weakness categorised by birth year for swimmers born between 1989 and 2000. Popularity is calculated by counting number of users with each of the characteristics. Exclude records where Weakness is not recorded in database. Sort the results by birth year in reverse order.

cls

db**.**PSL**.**aggregate**([**

**{**$match**:{**

Year\_Born**:{**$gt**:**1989**,**$lt**:**2000**},**

Weakness**:{**$exists**:true}**

**}},**

**{**$unwind**:{**

path**:** "$Hobbies"

**}},**

**{**$group**:{**

\_id**:** **{**Year**:**"$Year\_Born"**,**Hobby**:**"$Hobbies"**,**Weakness**:**"$Weakness"**},**

Count\_Swimmers**:{**$sum**:**1**},**

**}},**

**{**$sort**:{**

"\_id.Year"**:**1**,** Count\_Swimmers**:-**1

**}},**

**{**$group**:{**

\_id**:**"$\_id.Year"**,**Hobby**:{**$first**:**"$\_id.Hobby"**},**

Weakness**:{**$first**:**"$\_id.Weakness"**}**

**}},**

**{**$sort**:{**

\_id**:-**1

**}},**

**])**

**--------------------------------**



Q9) Calculate number of male swimmers for each stroke. Limit the results to strokes where swimmer count is at least 250. Sort the results by Stroke name.

cls

db**.**PSL**.**aggregate**([**

**{**$match**:{**

Gender**:**"Male"

**}},**

**{**$unwind**:{**

path**:**"$Strokes"

**}},**

**{**$project**:{**

\_id**:**0**,**Strokes**:**1

**}},**

**{**$group**:{**

\_id**:**"$Strokes"**,** Swimmer\_Count**:{**$sum**:**1**}**

**}},**

**{**$match**:{**

Swimmer\_Count**:{**$gt**:**250**}**

**}},**

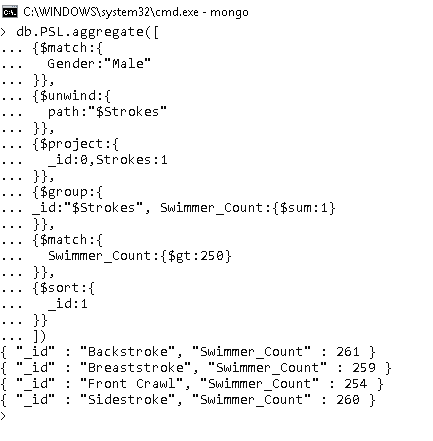
**{**$sort**:{**

\_id**:**1

**}}**

**])**

**--------------------------------**



Q10) Calculate average height and weight for each type of stroke, categorized further by weakness of swimmer. Display records where average height is at least 165 cm and average weight is at most 69 kg. Exclude records where weakness is not recorded for a swimmer in a database. Sort the results by average height and weight in descending order.

cls

db**.**PSL**.**aggregate**([**

**{**$match**:{**

Weakness**:{**$exists**:true}**

**}},**

**{**$unwind**:{**

path**:** "$Strokes"

**}},**

**{**$group**:{**

\_id**:{**Stroke**:**"$Strokes"**,**Weakness**:**"$Weakness"**},**

Avg\_Height**:{**$avg**:**"$Height"**},**

Avg\_Weight**:{**$avg**:**"$Weight"**}**

**}},**

**{**$match**:{**

Avg\_Height**:{**$gte**:**165**},**

Avg\_Weight**:{**$lte**:**69**}**

**}},**

**{**$sort**:{**

Avg\_Height**:-**1**,**Avg\_Weight**:-**1

**}},**

**])**

**--------------------------------**



Q11) Calculate maximum height of swimmers for each Hobby further categorized by stokes. Limit the records to groups where maximum height is greater than 190.5 cm. Sort the results by height.

cls

db**.**PSL**.**aggregate**([**

**{**$match**:{**

County**:**"Washington"

**}},**

**{**$unwind**:{**

path**:** "$Hobbies"

**}},**

**{**$unwind**:{**

path**:** "$Strokes"

**}},**

**{**$group**:{**

\_id**:{**Hobby**:**"$Hobbies"**,**Stroke**:**"$Strokes"**},**

Max\_Height**:{**$max**:**"$Height"**}**

**}},**

**{**$match**:{**

Max\_Height**:{**$gt**:**190.5**}**

**}},**

**{**$sort**:{**

Max\_Height**:**1

**}},**

**])**



# Stage 4 – MongoDB tools

## Compass

### Step 1 - Download

<https://www.mongodb.com/download-center/compass?jmp=hero>

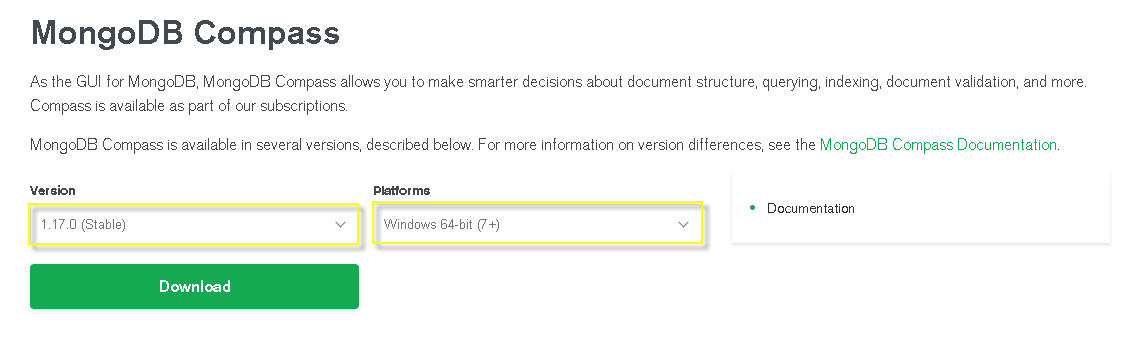


Figure 1: Download Compass

### Step 2 – Run and Connect

Click on mongodb-compass-1.17.0-win32-x64.exe file and wait for below window to open.

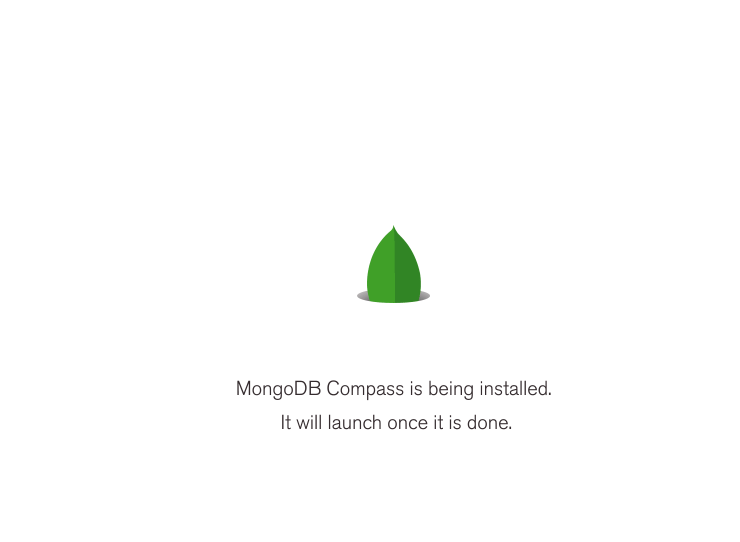


Figure 2: Installing Compass

Once the installation is finished fill in the following details and click connect.

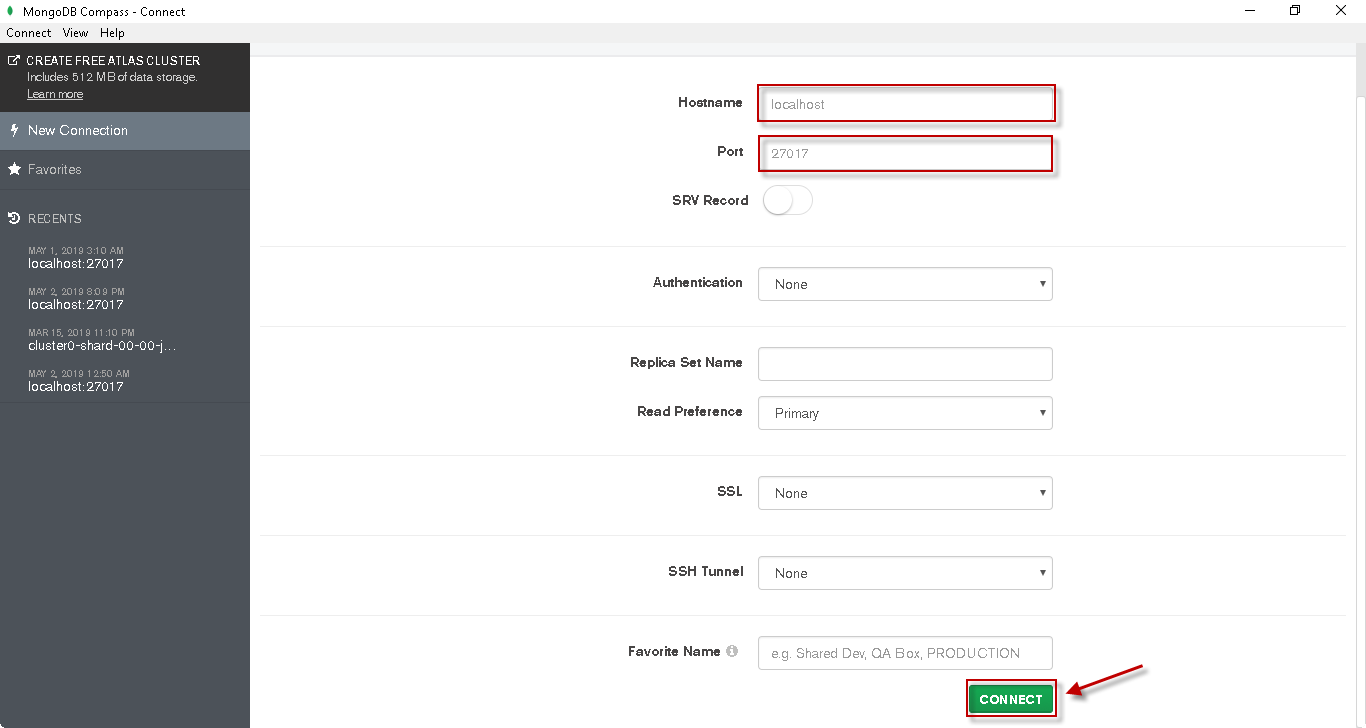


Figure 3: Connecting Compass

Once connected, click on the PSL database from the available list of databases.

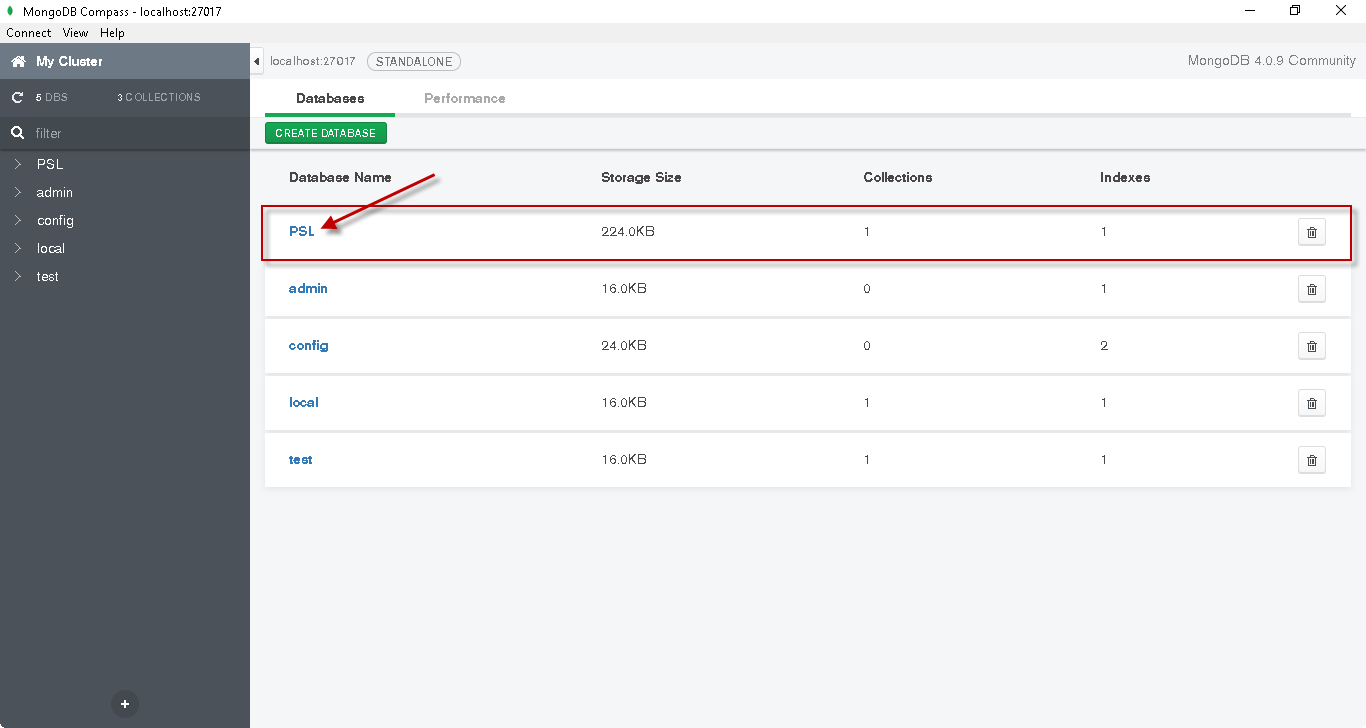


Figure 4: Access PSL Database

Next click on the PSL collection, we can see that the collection already have 1000 documents

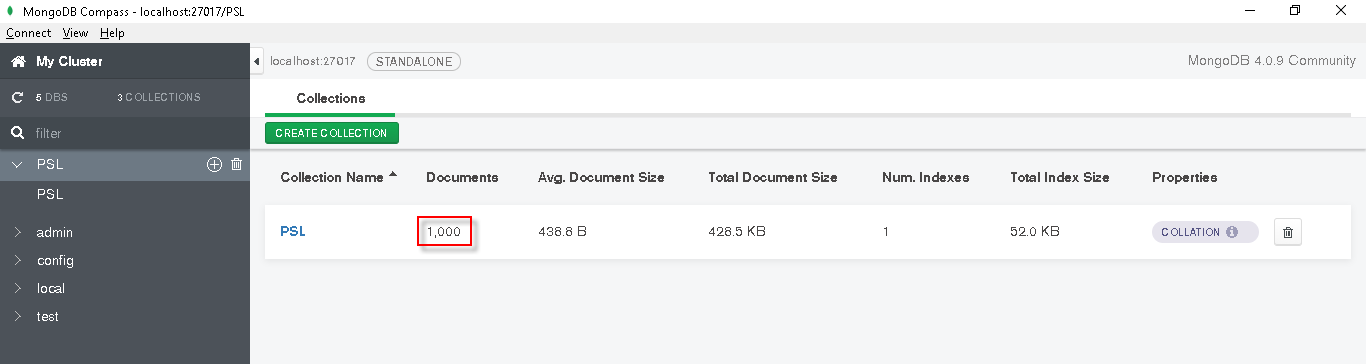


Figure 5: PSL Collection

Compass provide a very easy to use and read interface. It provides an interactive approach for performing CRUD and other operations on mongoDB.

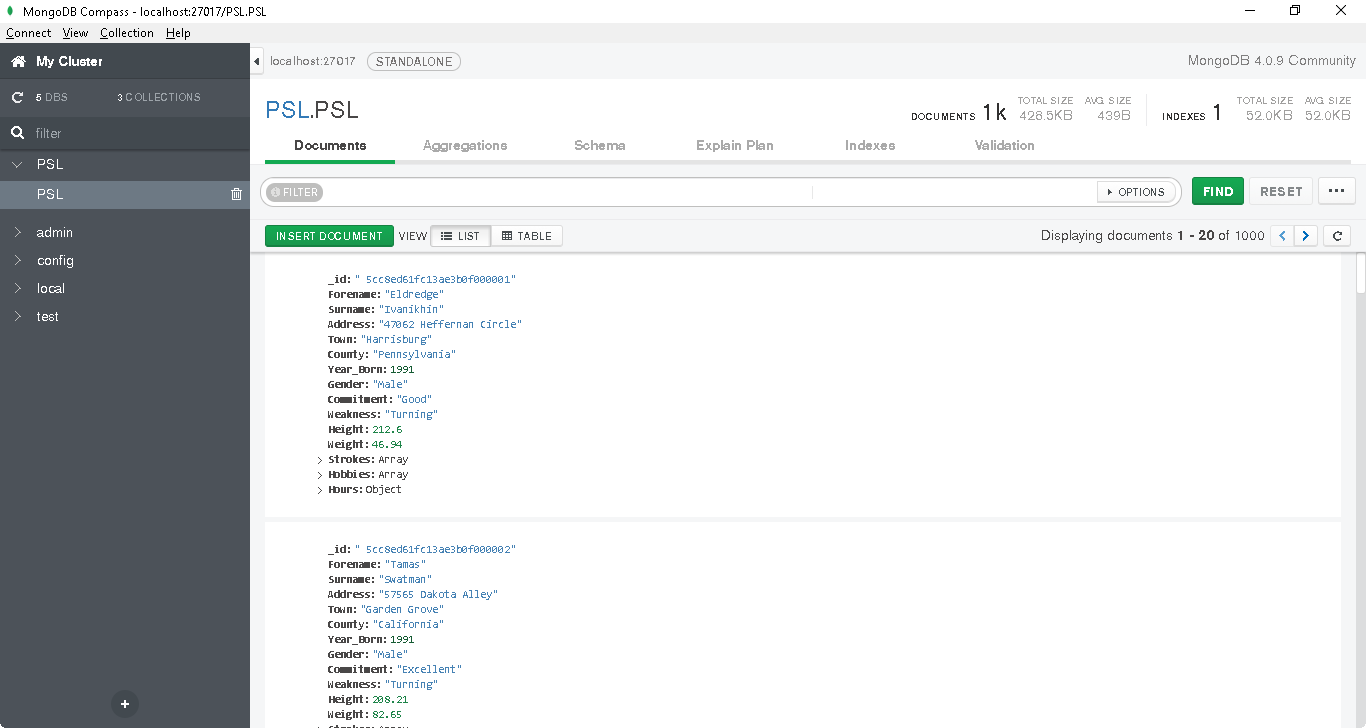


Figure 6: Visualize documents

### Step 3 – Documents Tab (CRUD)

#### Creating New Documents

Click on Insert Document button

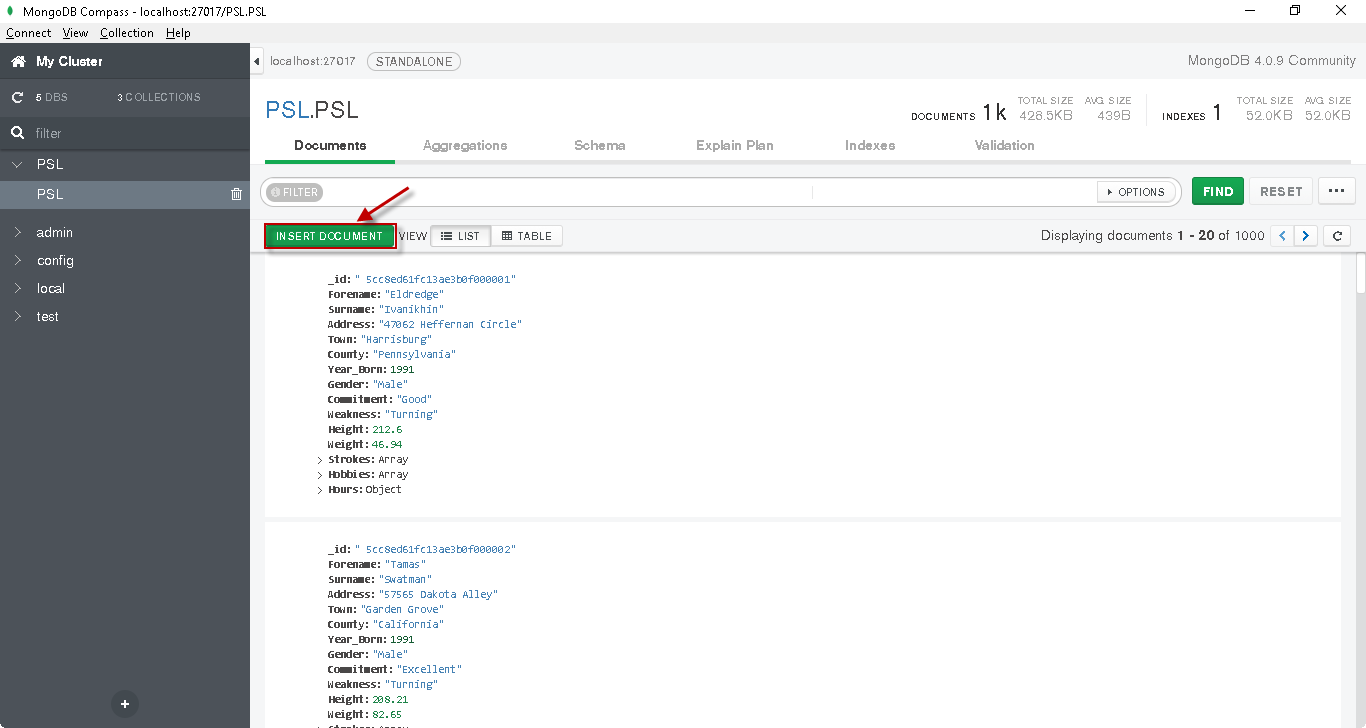


Figure 7: Insert Document - Step 1

Add the document details as required



Figure 8: Insert Document - Step 2

#### Read or Filtering Documents

Compass provide graphical interface for easy querying mongoDB. All the options of find() query like project, sort, collation, skip and limit can be edited and desired documents can be filtered out from the entire collection.

Results are shown directly in the window below the query which saves lot of time rebuilding queries and running them on command shell of mongoDB.

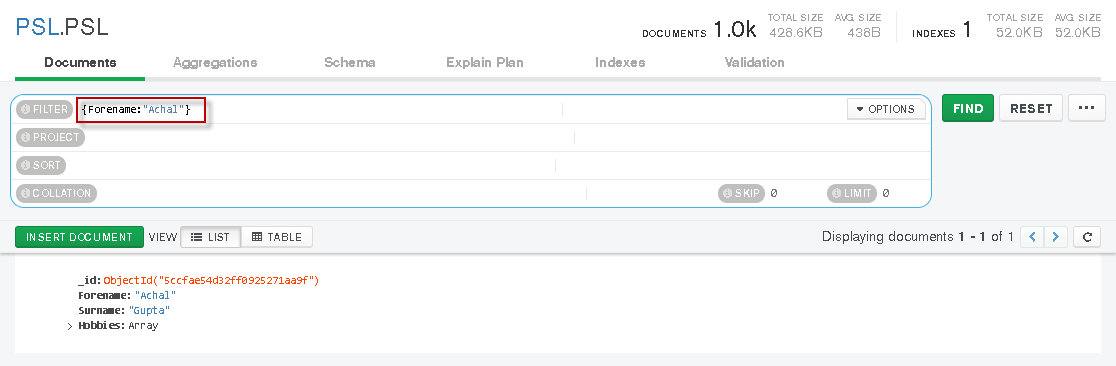


Figure 9: View inserted document

#### Updating Documents

Documents can be easily edited using compass just click the edit button which appears when you hover upon the document

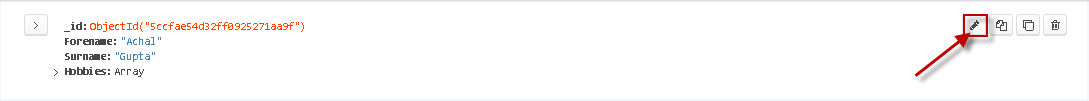


Figure 10: Update document - Step 1

Multiple changes can be added in a document interactively using compass. Below example shows update, delete and addition of fields in the document

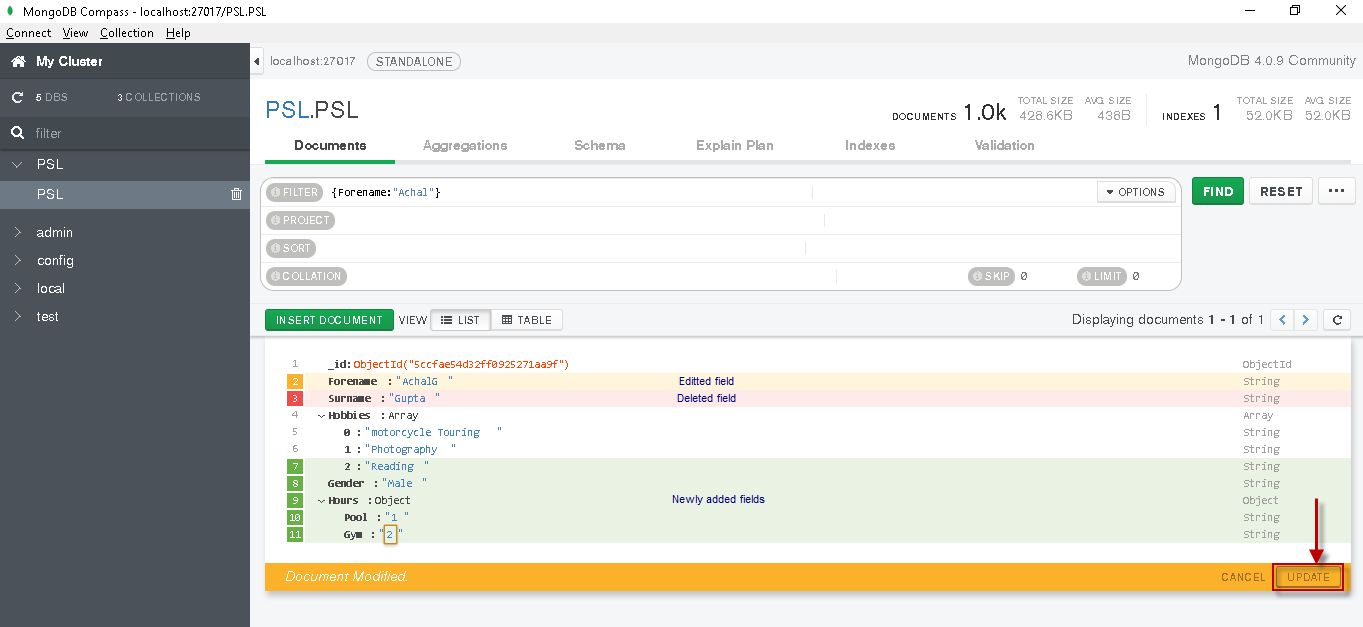


Figure 11: Update document - Step 2

Once the required changes are performed on the document all changes can be committed to mongoDB by clicking on the UPDATE button as shown above.

#### Deleting Documents

Individual documents can be deleted by click on the delete button visible when hoovered upon the document.

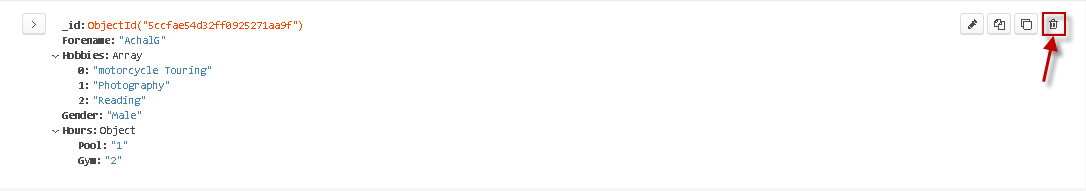


Figure 12: Delete document - Step 1

Compass will ask for a confirmation to delete the document, Click Delete button to delete the document.

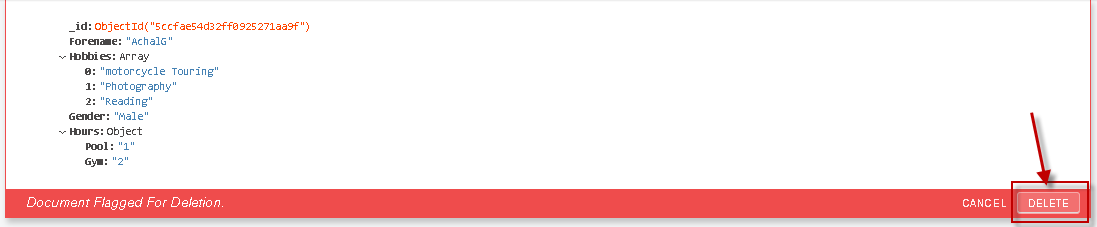


Figure 13: Delete document - Step 2

#### Tabular View

Compass also provide a tabular view of the data and hence visualizing the data is quite easy.

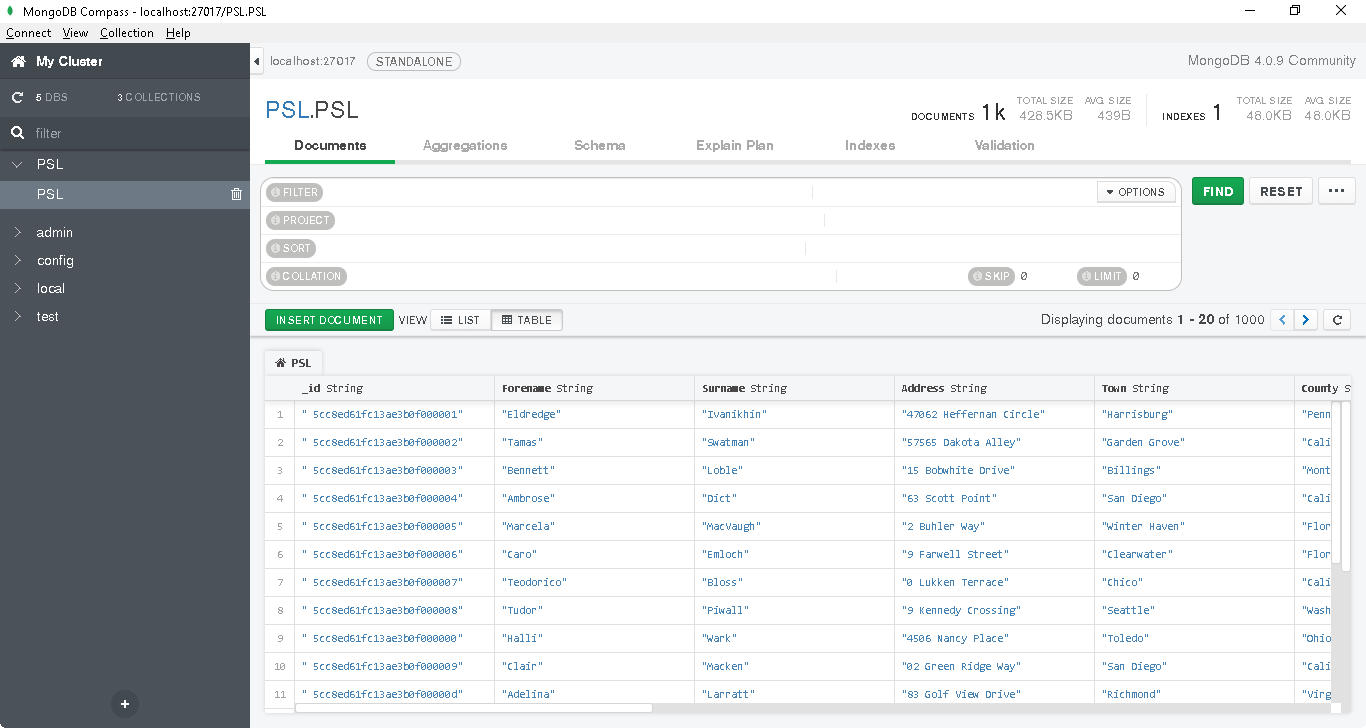


Figure 14: Tabular view

### Step 4 - Aggregations Tab

Creating Aggregation Pipeline is comparatively easy and fast in compass. Individual stages can be edited separately, and results of each stage can be viewed side by side to get a better prospective of pipeline. Moreover, Pipelines can be saved for future reference also.

#### Example -1

List the most popular hobby and weakness categorised by birth year for swimmers born between 1989 and 2000. Popularity is calculated by counting number of users with each of the characteristics. Exclude records where Weakness is not recorded in database. Sort the results by birth year in reverse order.

Overview of Pipeline – Below figure shows the architecture of pipeline. There are 6 stages overall starting from $match followed by $unwind, $group, $sort, $group and $sort.

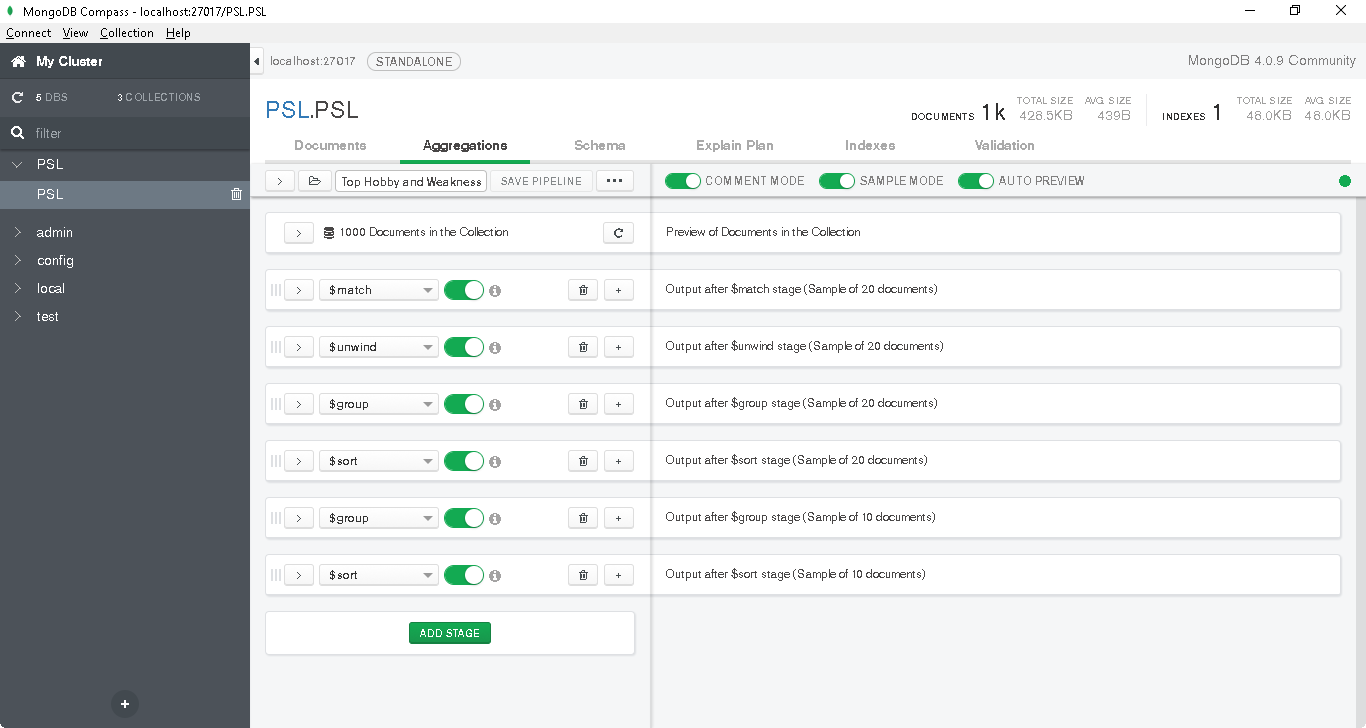


Figure 15: Example 1 Aggregation Pipeline in Compass - Overview

##### Stage -1

Filter records where birth year is between 1989 and 2000 and Weakness is recorded in database.

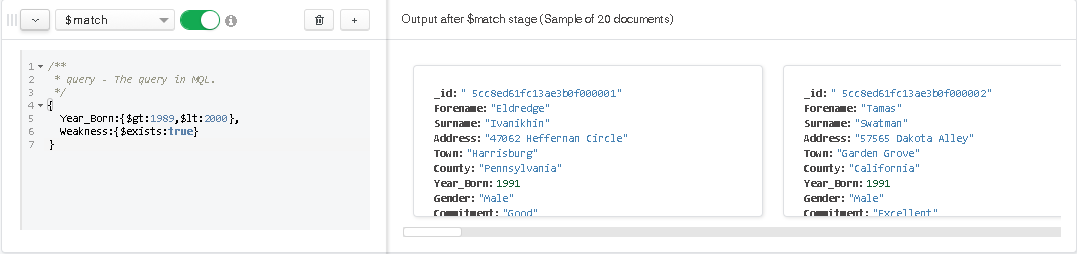


Figure 16: Example 1 Aggregation Pipeline in Compass - Stage 1

##### Stage – 2

Unwind the Hobbies Array

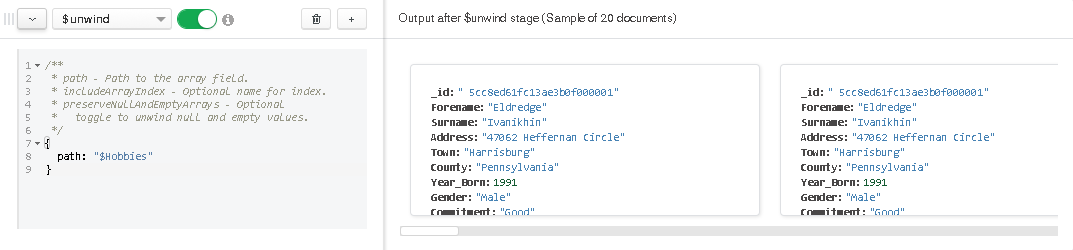


Figure 17: Example 1 Aggregation Pipeline in Compass - Stage 2

##### Stage – 3

Calculate count of swimmers categorized by Year\_born, hobbies and Weakness of swimmers.

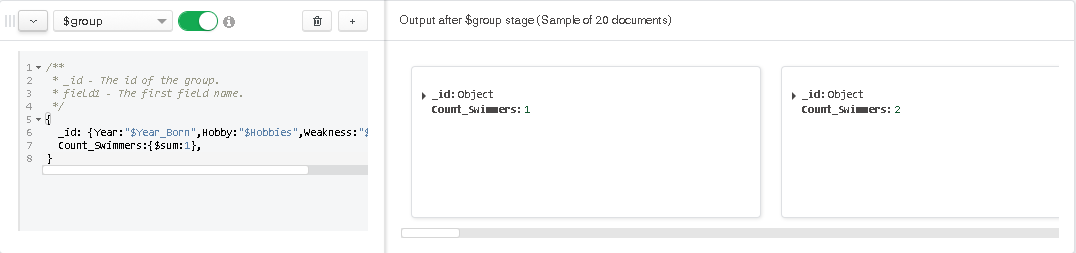


Figure 18: Example 1 Aggregation Pipeline in Compass - Stage 3

##### Stage – 4

Sort results by Year in ascending order and count of swimmers in descending order.

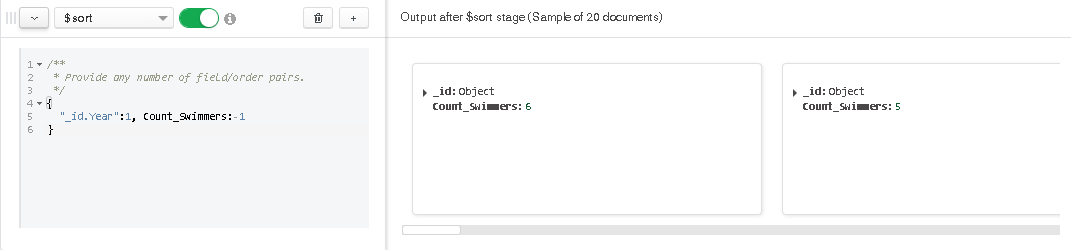


Figure 19: Example 1 Aggregation Pipeline in Compass - Stage 4

##### Stage – 5

Identify top Hobby and Weakness i.e. maximum swimmer count per year

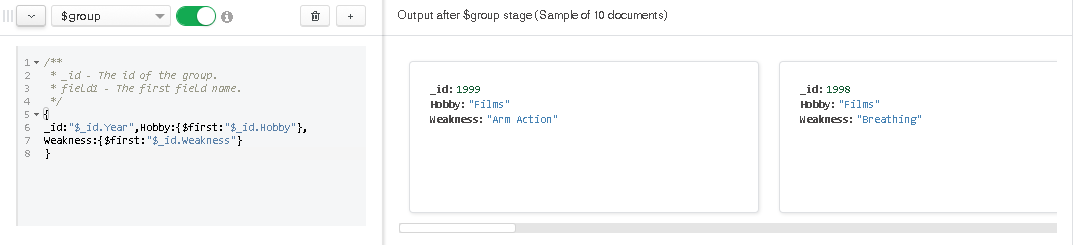


Figure 20: Example 1 Aggregation Pipeline in Compass - Stage 5

##### Stage – 6

Sort the results by year.

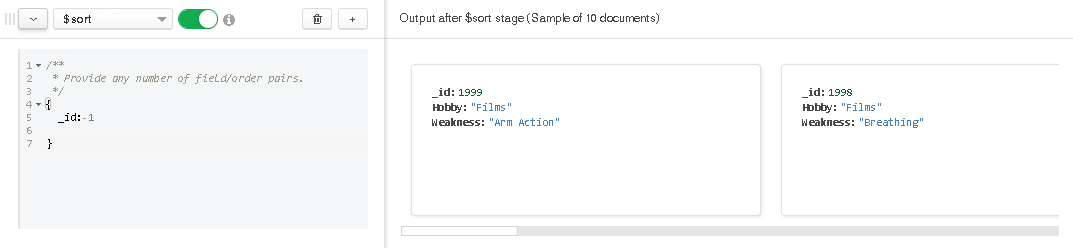


Figure 21: Example 1 Aggregation Pipeline in Compass - Stage 6

#### Example -2

Display count of swimmers for each of the Hobby. Also Divide height of swimmers into 5 buckets and calculate number of swimmers belonging to each bucket.

Below pipeline uses $facet stage to combine two stages $unwind and $bucketAuto to display the desired results.

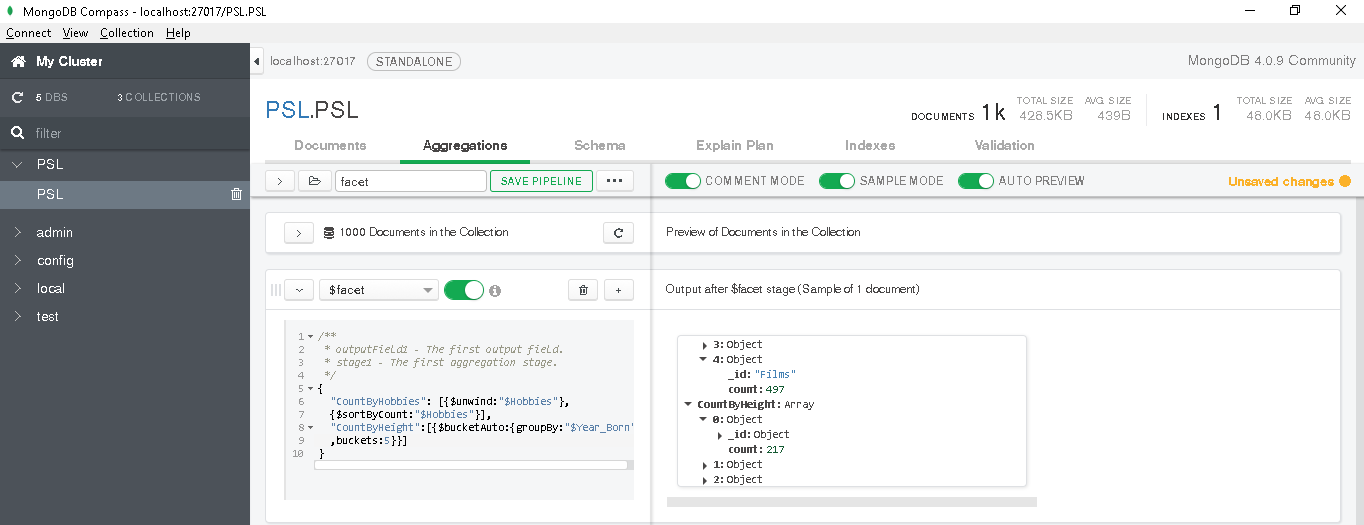


Figure 22: Example 2 Aggregation Pipeline in Compass

### Step 5 - Schema Tab

One of the most important features of Compass. This tab helps visualizing range of values present in each document of a collection.

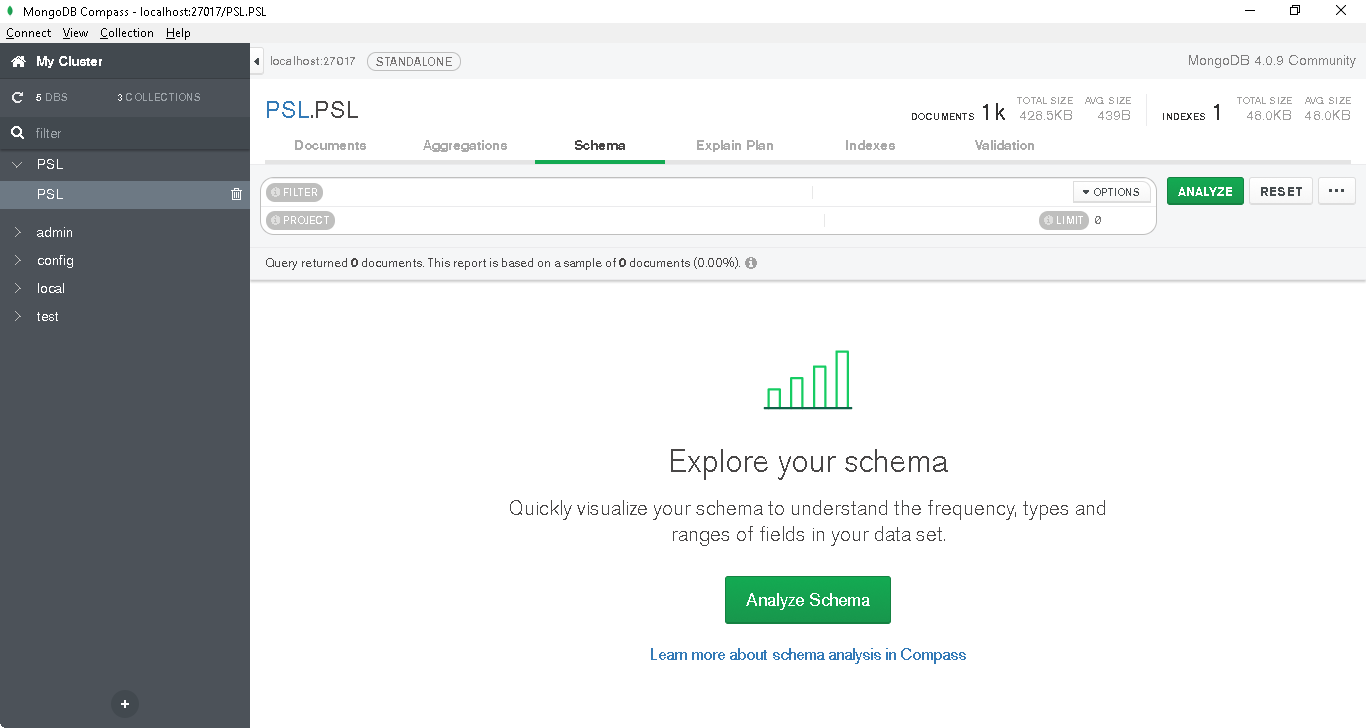


Figure 23: Schema Tab

Analysing String Values

Compass will automatically interactive visuals of string values for various fields. When hovered upon each bar will display the %out of total sample for specific value.

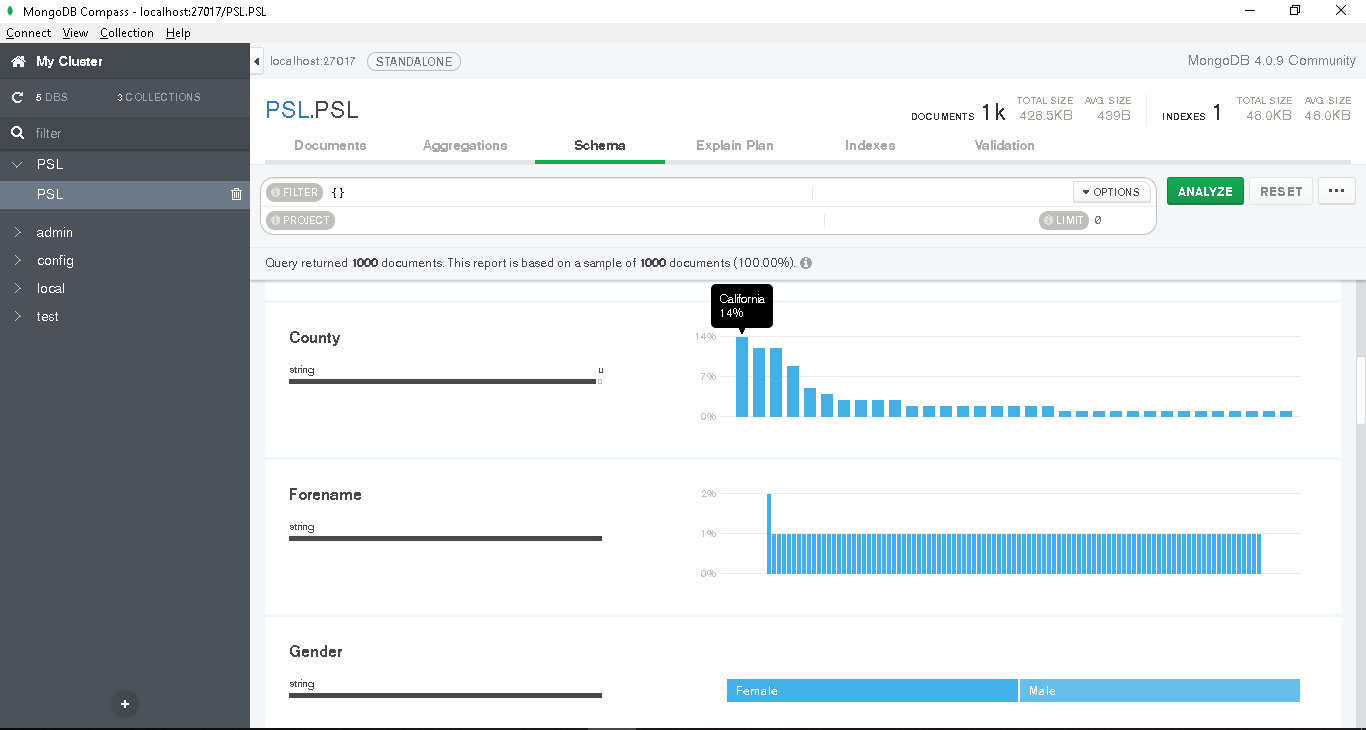


Figure 24: Schema Tab - Analysing String

Records can be filtered interactively by click single or multiple values from the Schema analysis.

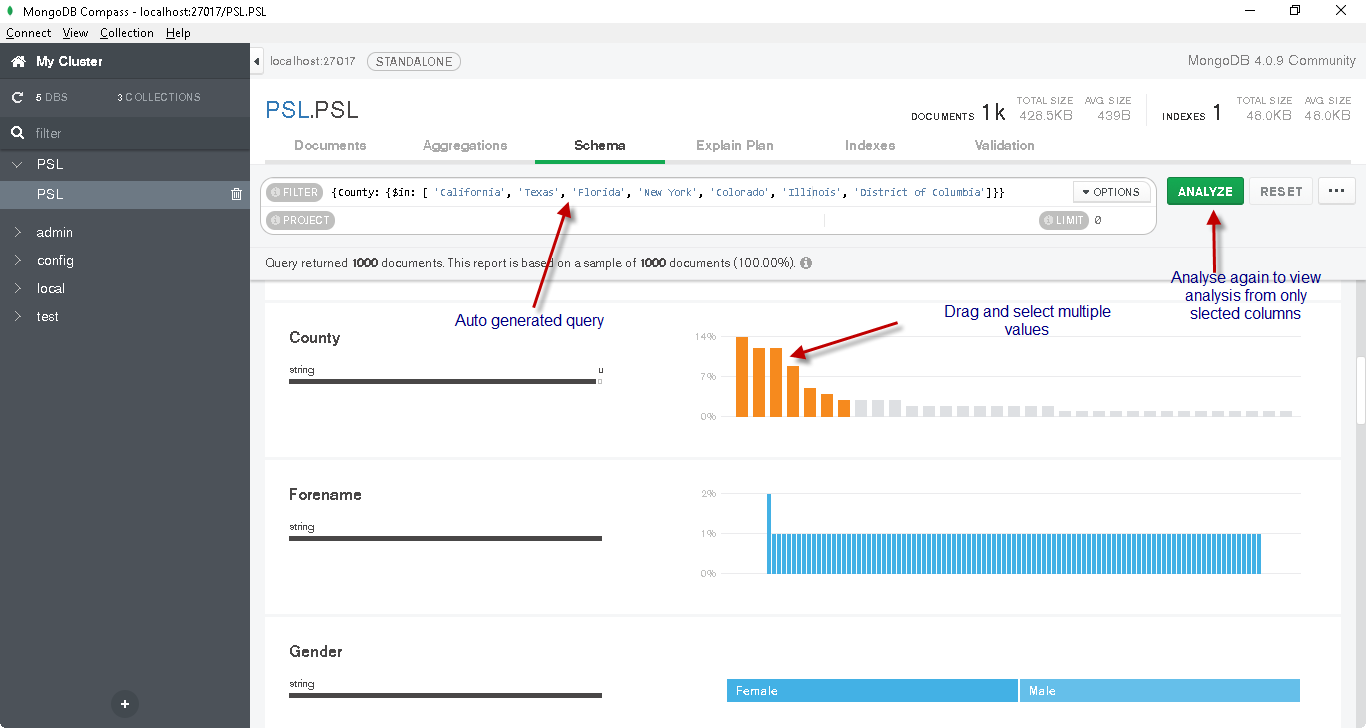


Figure 25: Schema Tab - Analysing String with filter

Once Analyse is clicked, visualizations will be changed and will use data from the filtered subset. Below screenshot shows analysis of filtered 456 documents.

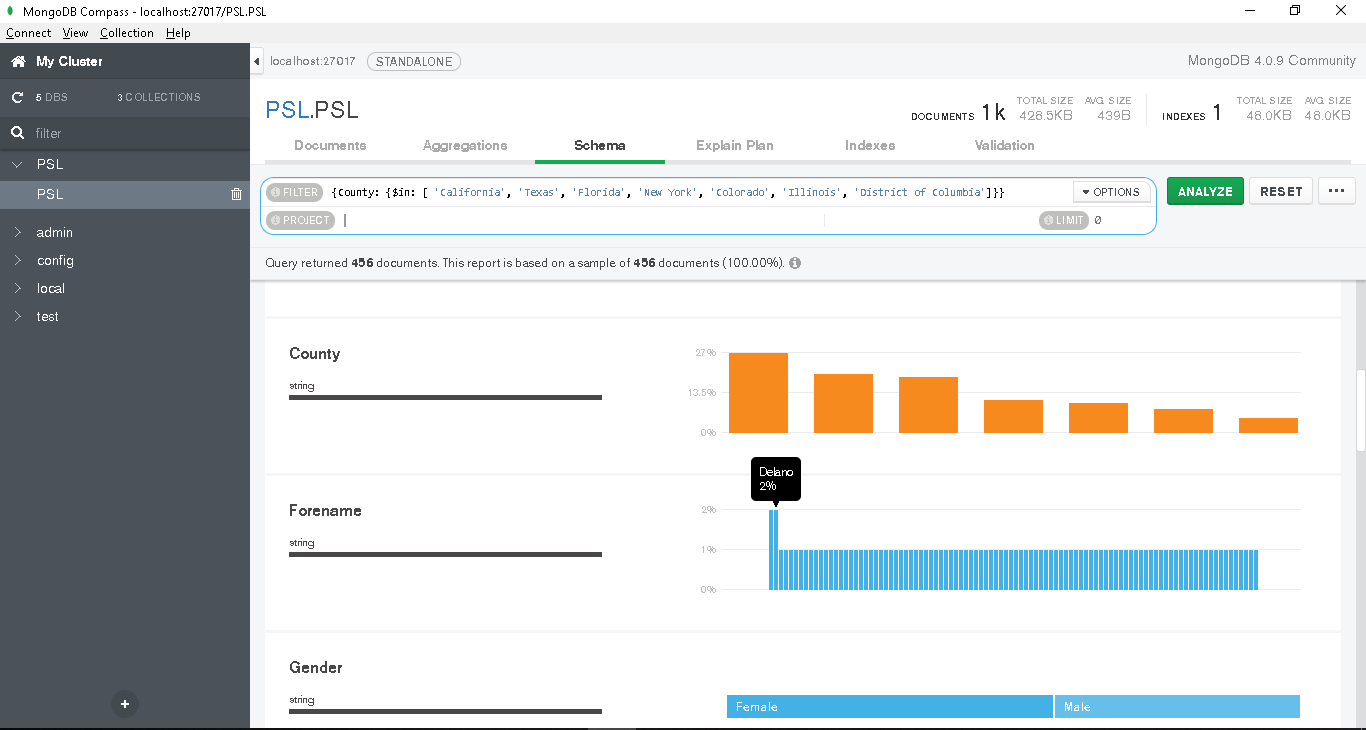


Figure 26: Schema Tab - Analysing String with filtered records

Analysis of Arrays

Arrays can be analysed on two granularities.

* Top Level Analysis – It will show minimum, maximum and average number of elements in array.
* Record level analysis – This analysis will show %out of total for individual elements.

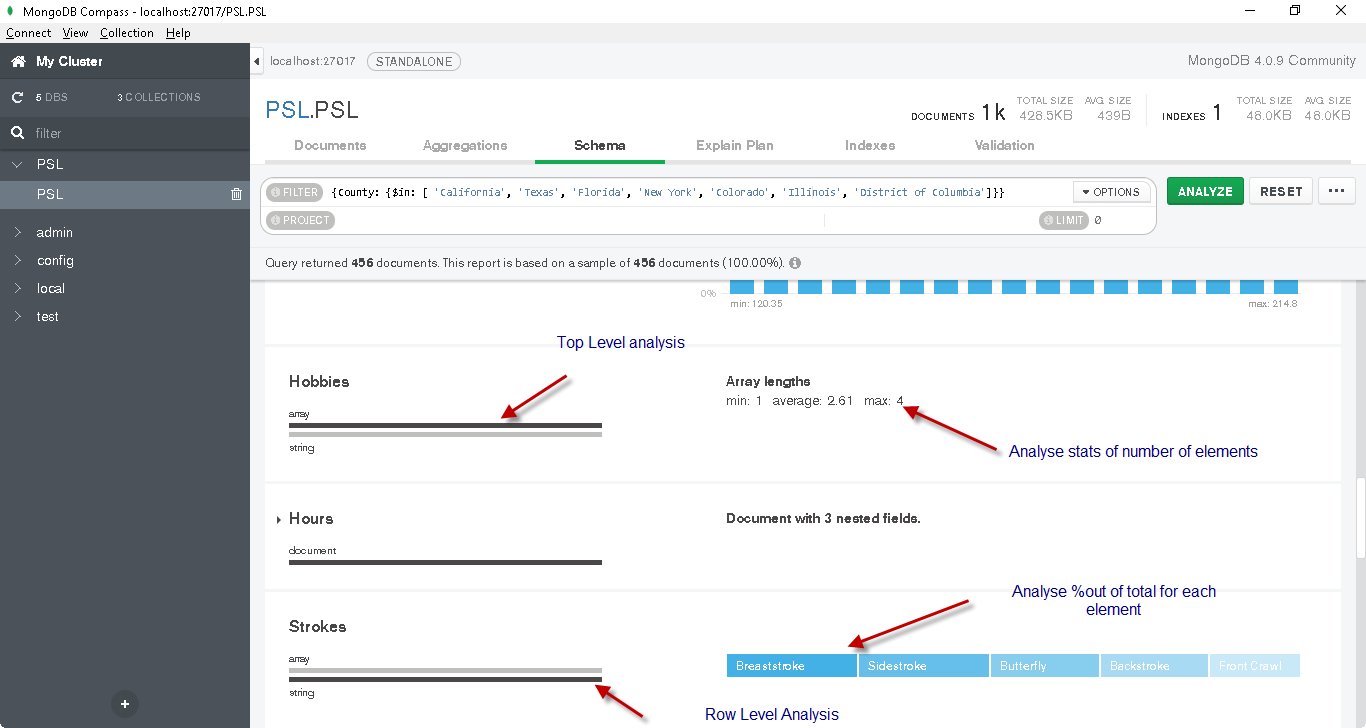


Figure 27: Schema Tab - Analysing Array

Analysis of Embeded Documents

At high level compass will display the depth of Embeded document. In our scenario, Compass displays 3 nested fields for training Hours.

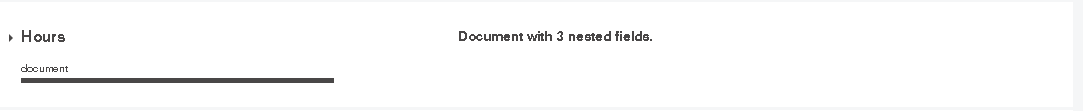


Figure 28: Schema Tab - Analysing Embeded Docs

Analysis can be expanded to visualize individual fields.

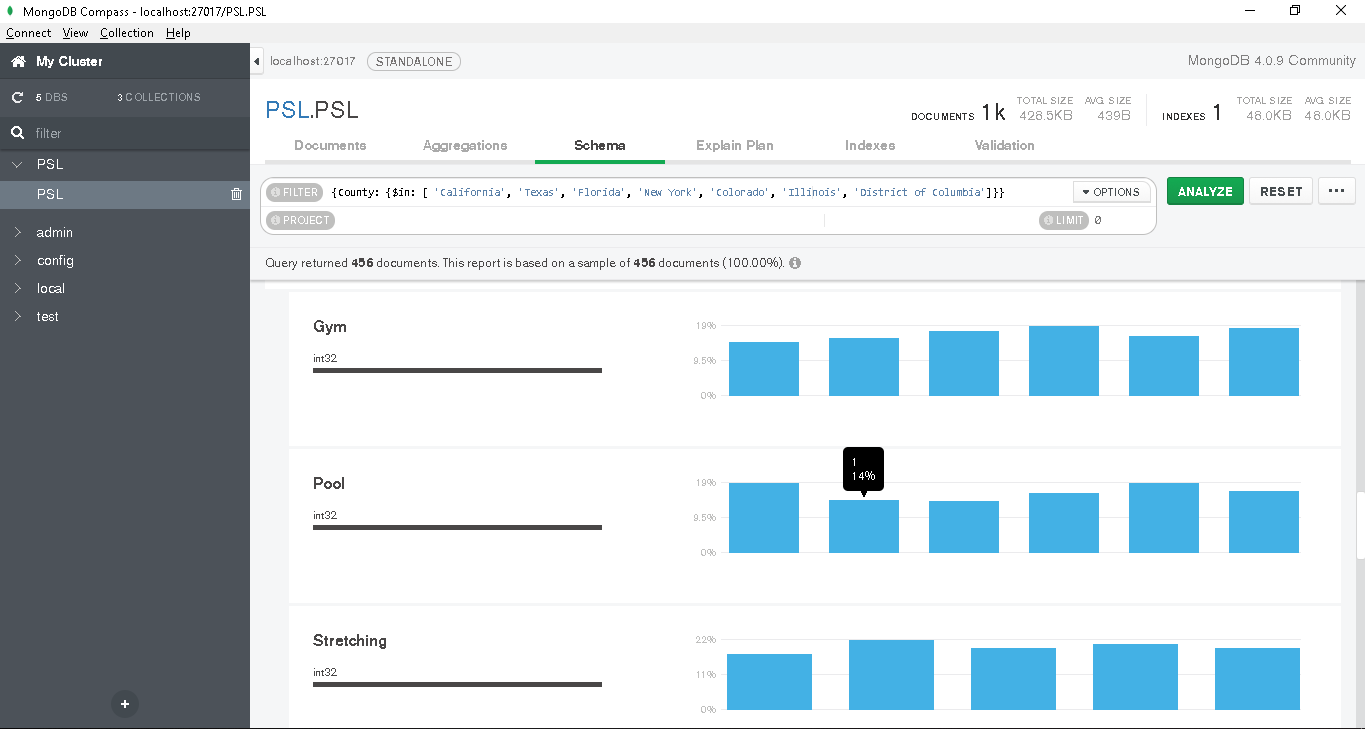


Figure 29: Schema Tab - Analysing Embeded Docs - Row Analysis

Again, filter can be applied by easily selecting values

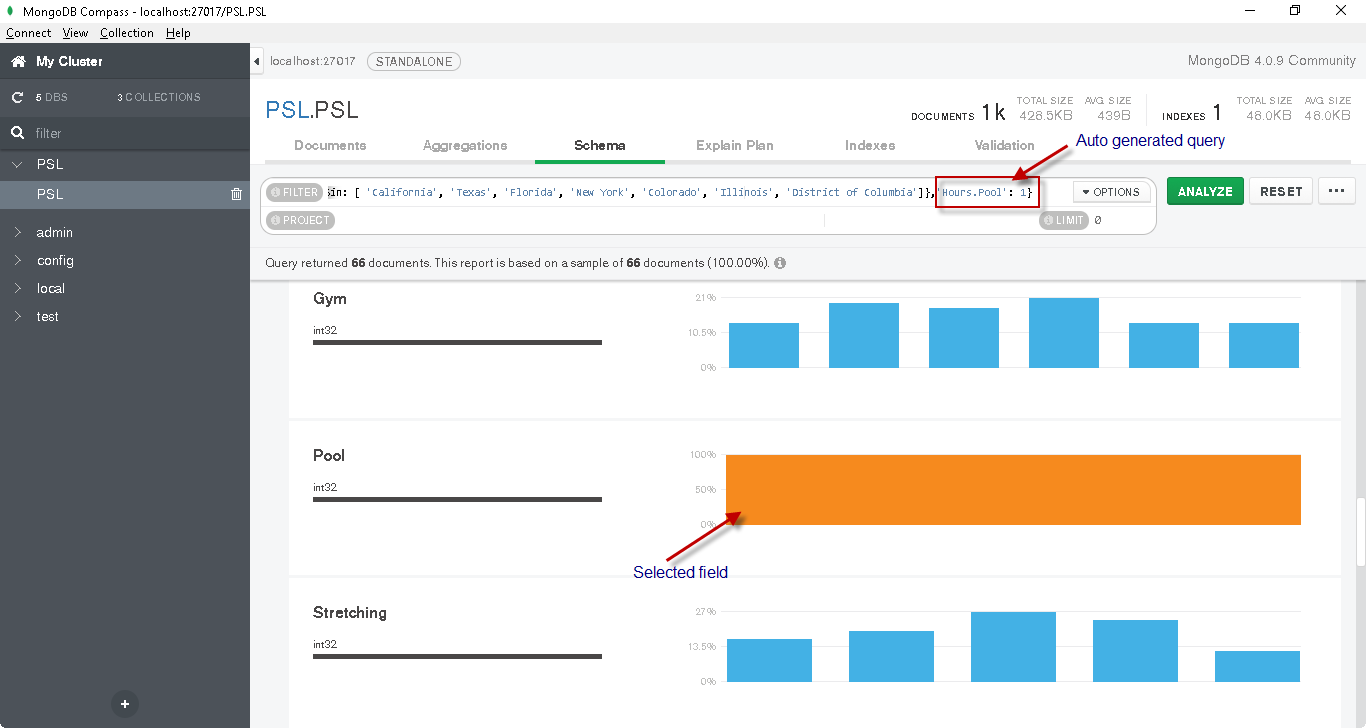


Figure 30: Schema Tab - Analysing Embeded Docs with filter

### Step 6 – Performance Optimization using Compass

#### Analyze Execution Plan

Go to Explain Plain tab in Compass. Add the query in filter and projects columns to display Address of swimmers whose address contains word "alley" irrespective of case.

Click on explain to visualize the execution plan.

**Key Observations:**

Total Execution time – 881ms

62 Documents returned.

No index available for query

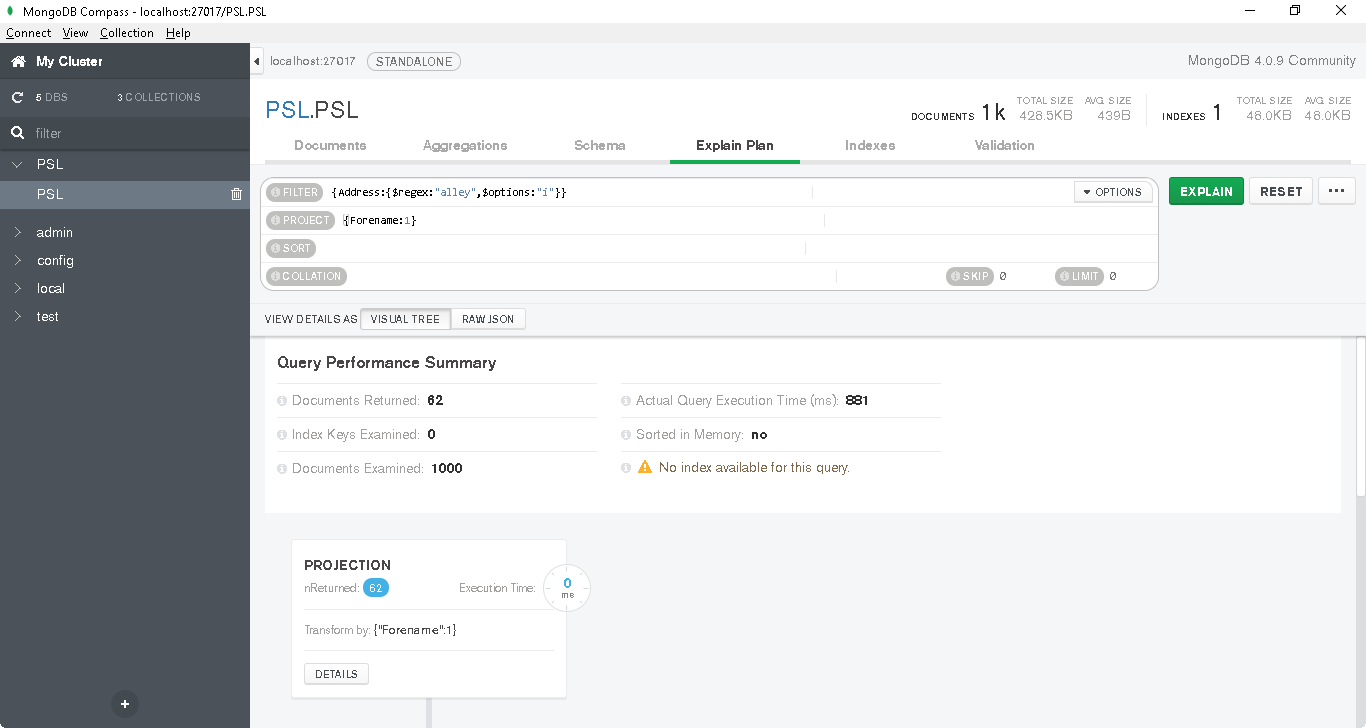


Figure 31: Initial Execution Plan

Column scan stage is visible in the detailed plan below which is highly time consuming,

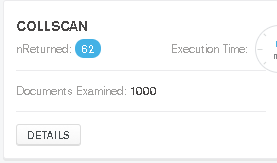


Figure 32: Initial Execution Plan details

#### Creating an index on Address from Indexes tab.

Click on Create Index button to create a new index on Address

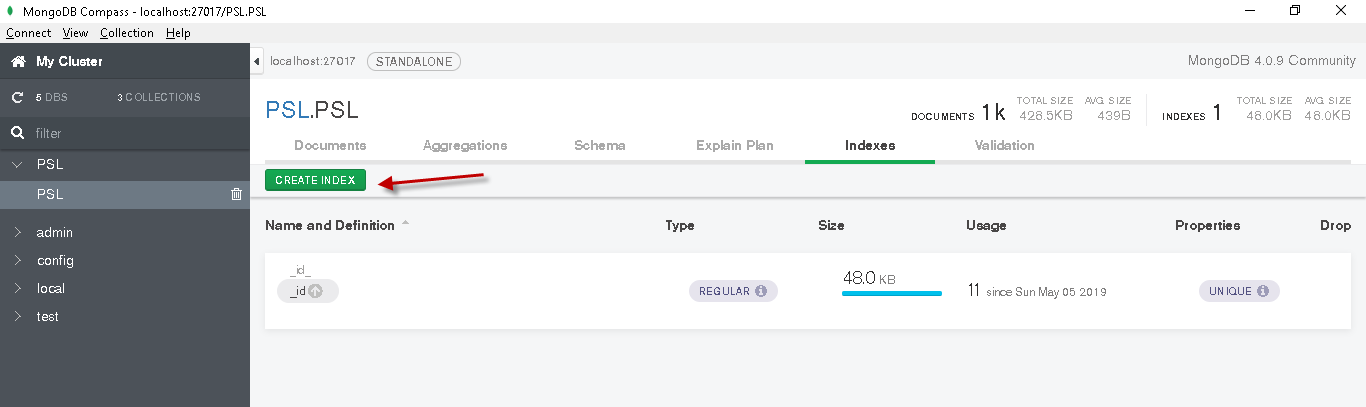


Figure 33: Creating Index Step 1

Fill in the following details and click on Create

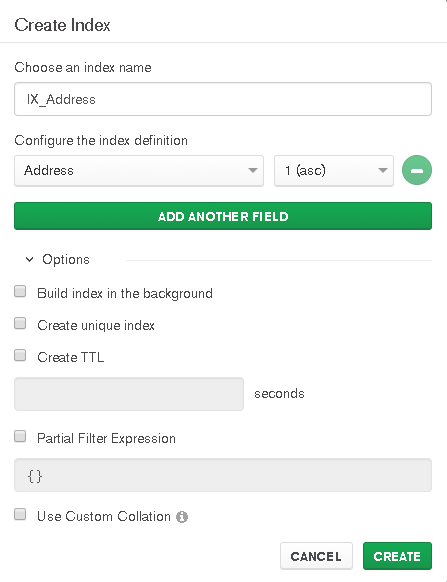


Figure 34: Creating Index Step 2

#### Evaluating performance change

Go back to Explain Plan tab in Compass and hit Explain button. Execution time is highly reduced and there is an index available for query.

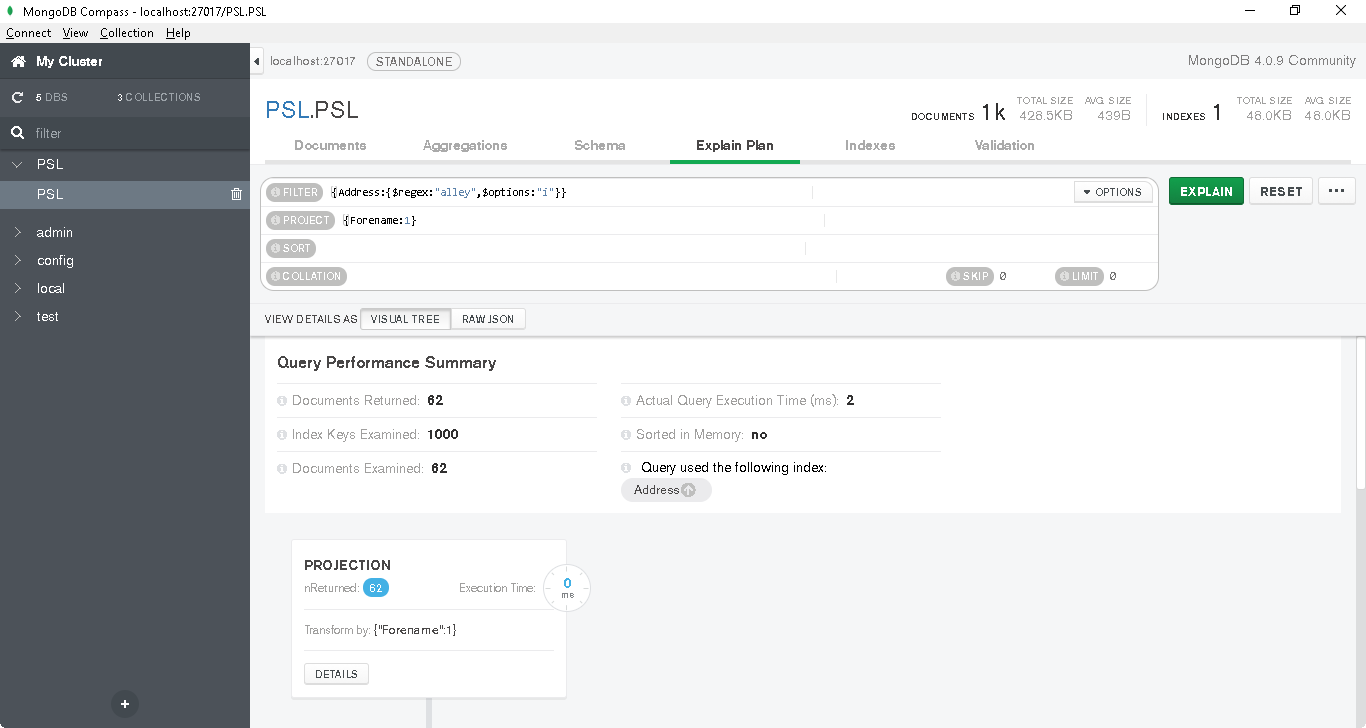


Figure 35: Updated Execution Plan

A new index scan stage is visible in the plan as compared to Column scan stage visible when no index was present.

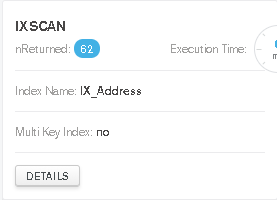


Figure 36: Updated Execution Plan details

## Atlas

### Step 1 - Create and Configure Atlas Cluster

Go to below URL and follow the instructions below.

<https://www.mongodb.com/cloud/atlas>

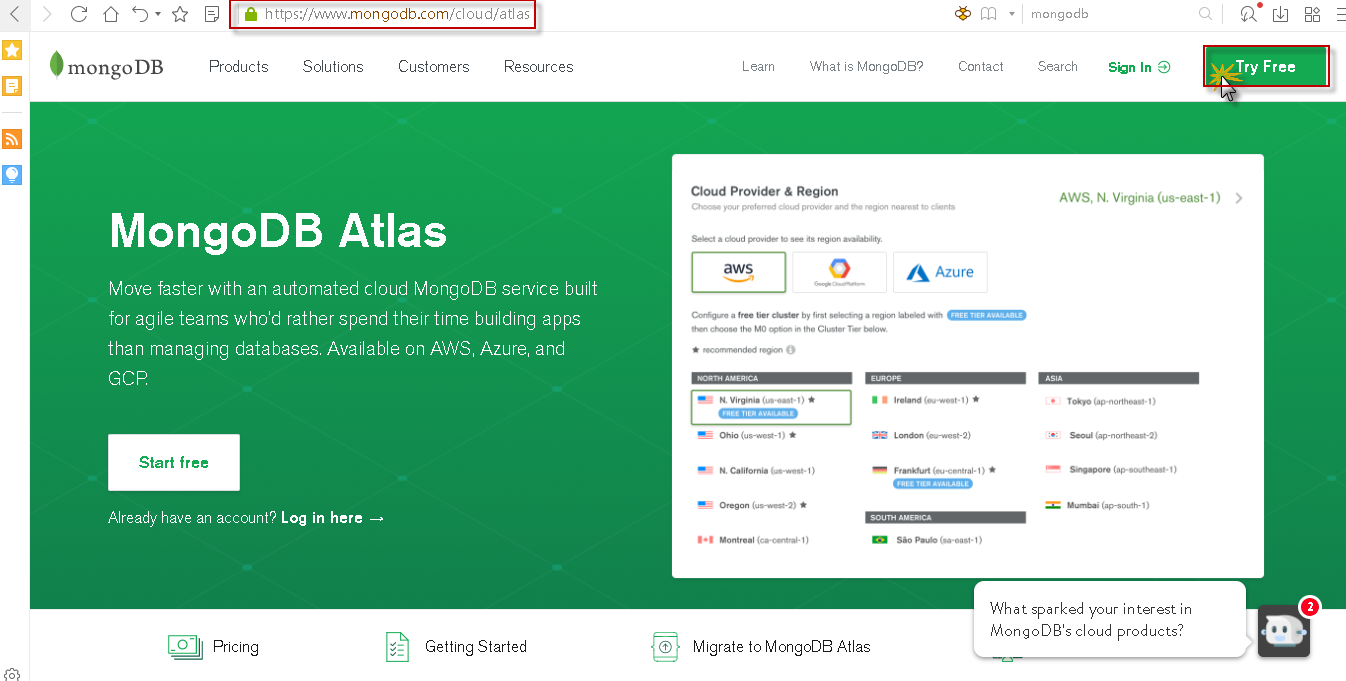


Figure 37: Launching Atlas

Fill in the registration details and create a password fulfilling the password requirements

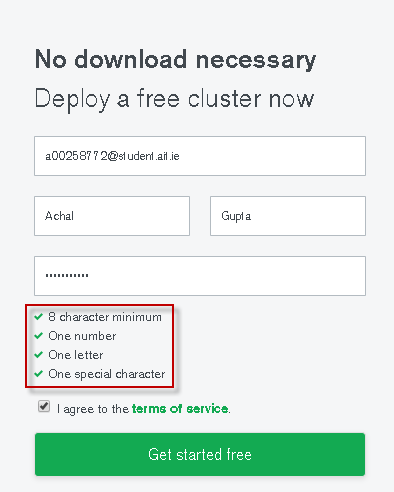


Figure 38: Registration for Atlas

As soon as the registration is successful, prompt will appear to create a new cluster. Click on the Build my first cluster button to continue.

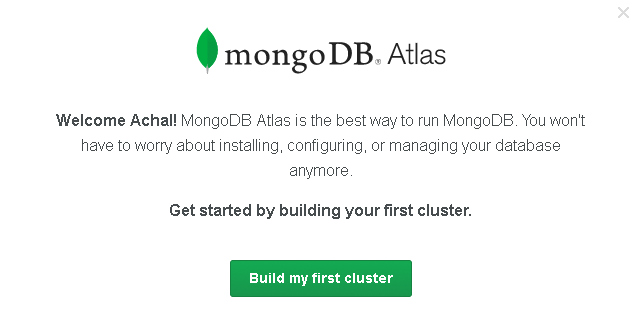


Figure 39: Creating Cluster Step 1

Chose the cloud provider from Amazon AWS, Google Cloud or Microsoft Azure. Here we are using Microsoft Azure cloud platform

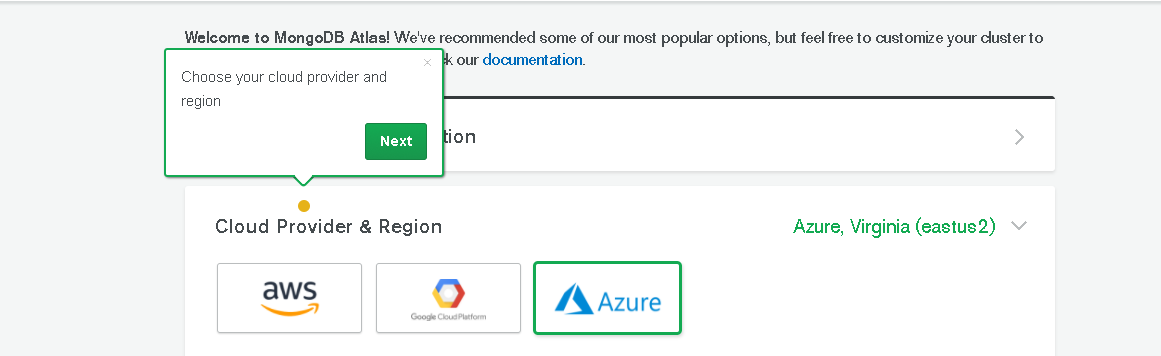


Figure 40: Creating Cluster Step 2

Next chose a Cloud server region. Make sure region has Free Tier Availability

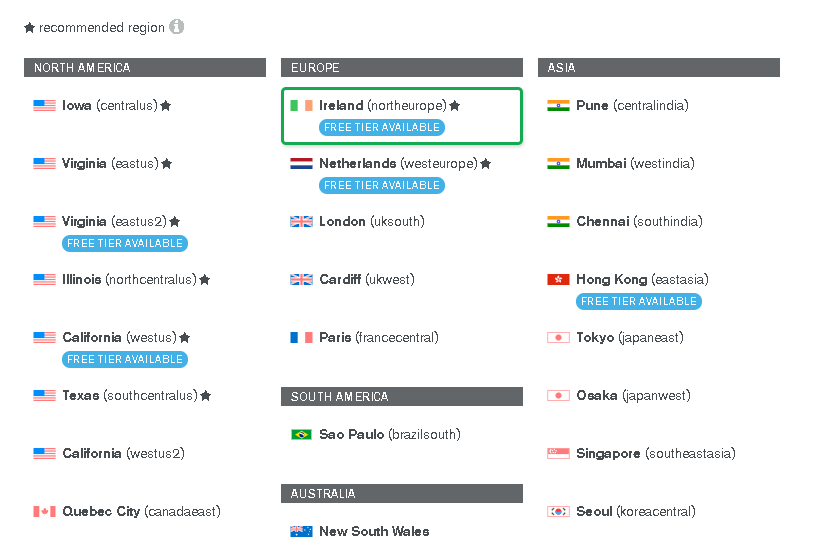


Figure 41: Creating Cluster Step 3

Next step is to customize cluster specs. Make sure to select M0 Sandbox cluster (Free).

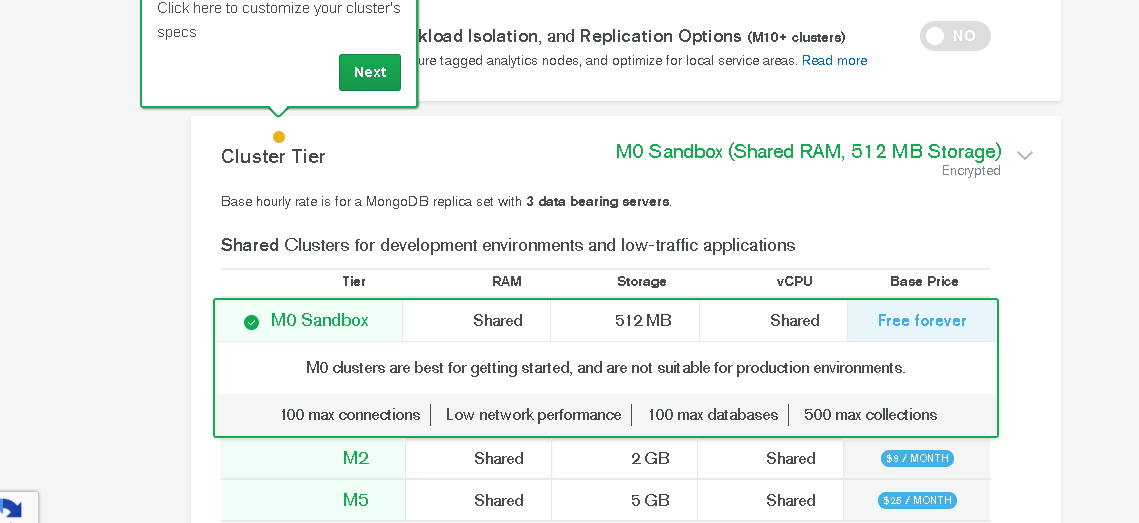


Figure 42: Creating Cluster Step 4

Next step would be to name the cluster and click on Create Cluster to launch cluster.

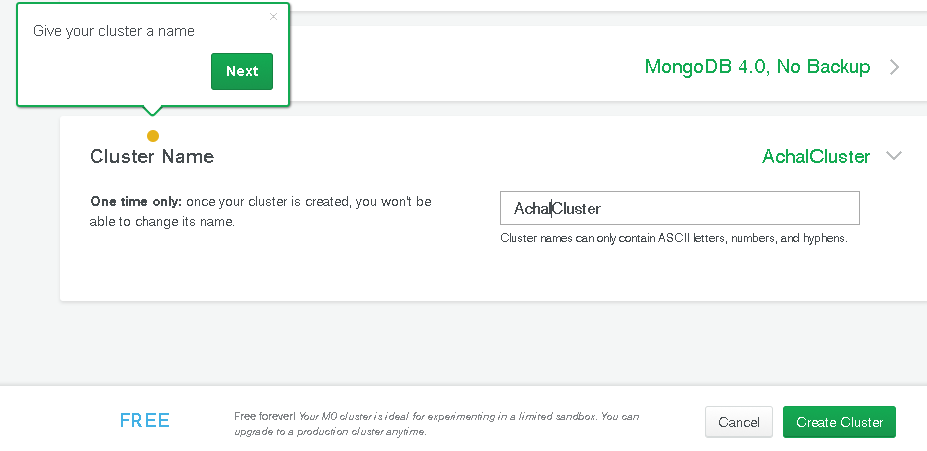


Figure 43: Creating Cluster Step 5

Wait for the cluster to set up, it can take from 7 to 10 minutes to set up a new cluster.

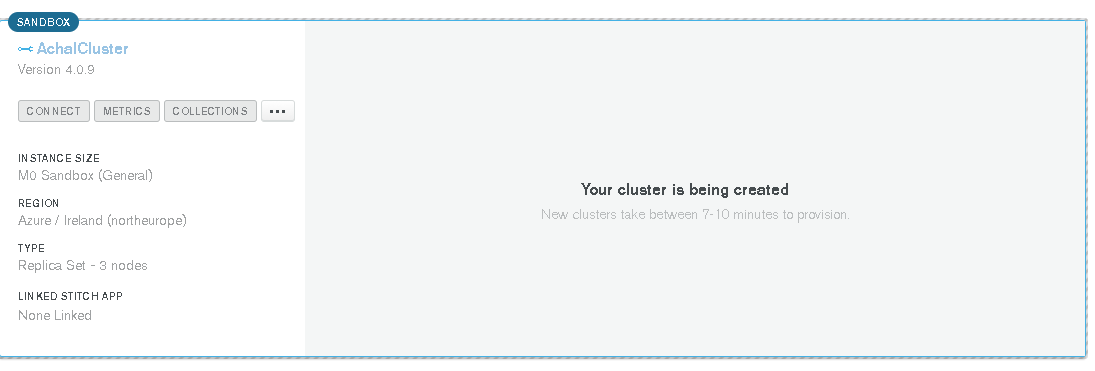


Figure 44: Creating Cluster Step 6

Once the Cluster is created first step is to create Database Users. Go to security tab

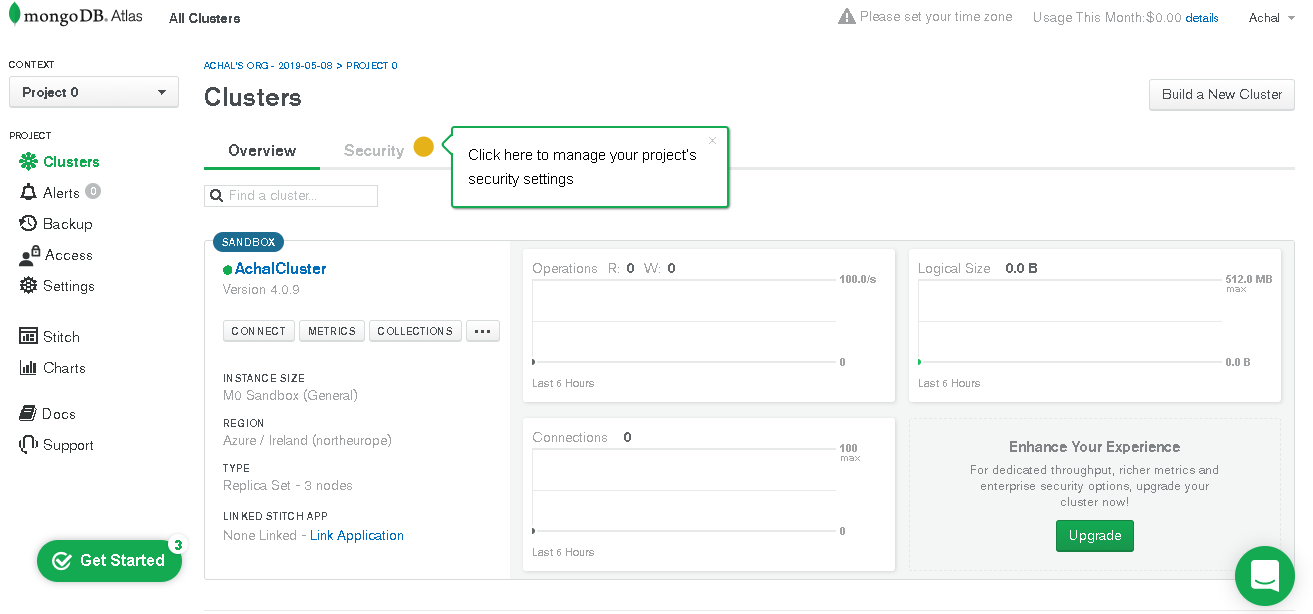


Figure 45: Configuring Cluster Step 1

Click add new user from the Security tab

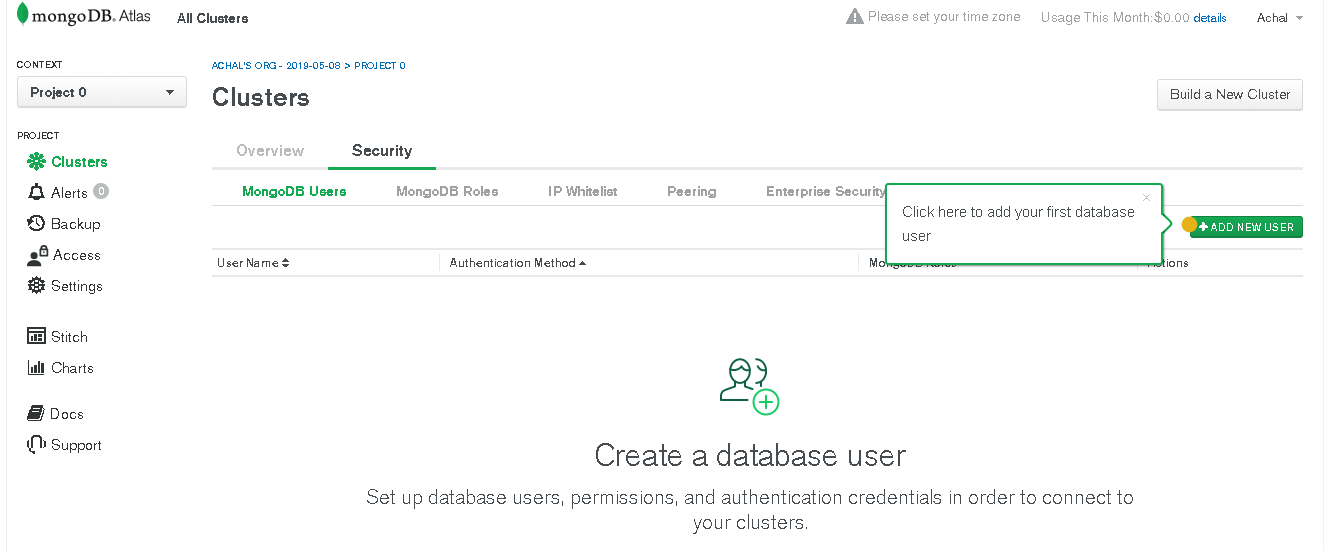


Figure 46: Configuring Cluster Step 2

Provide desired Username and Password, make sure to provide admin privileges and click add user.

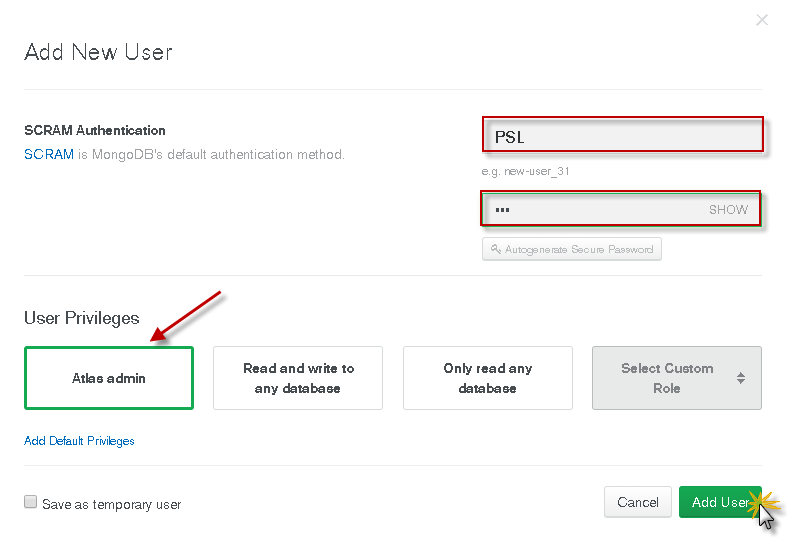


Figure 47: Configuring Cluster Step 2 Add DB User

Verify the user is created from MongoDB Users tab.

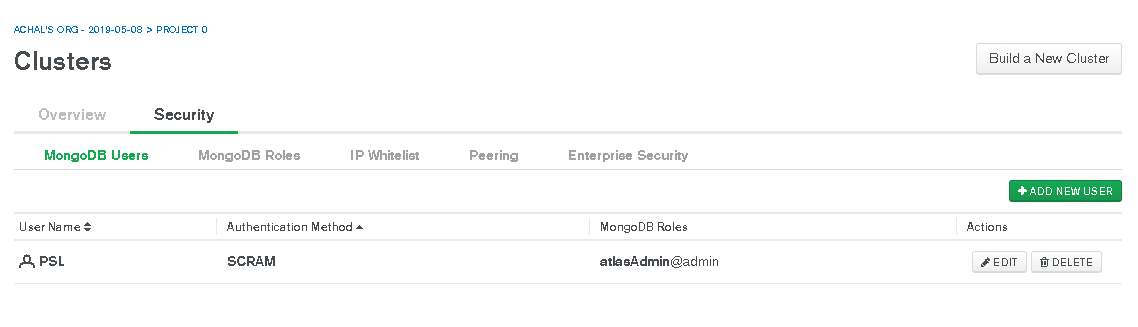


Figure 48: Configuring Cluster Step 3 Verify User

Once the user is created next step is to whitelist the IPs from which cluster needs to be accessed.

Click on IP Whitelist tab to proceed.

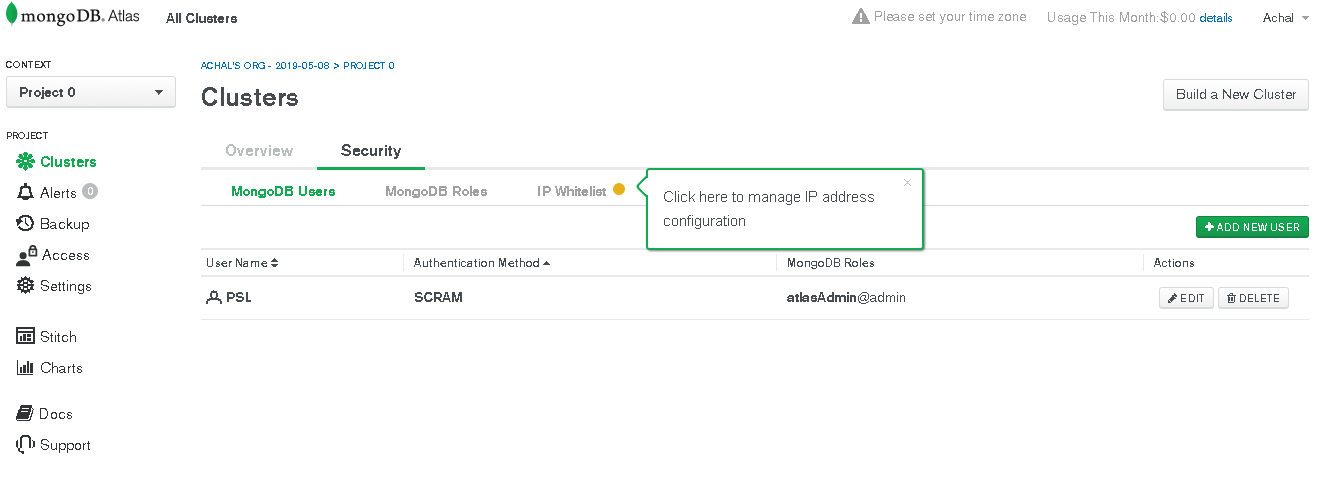


Figure 49: Configuring Cluster Step 4

Click on Add IP address button

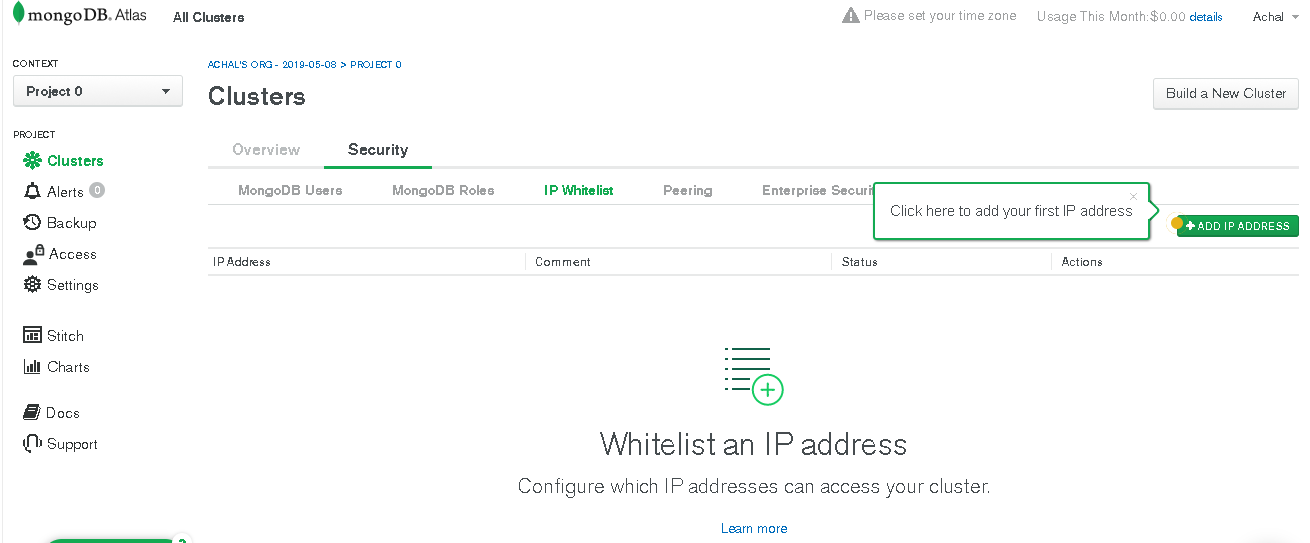


Figure 50: Configuring Cluster Step 5 IP Whitelist

Click on ALLOW ACCESS FROM ANYWHERE button to avoid unnecessary network issues. Although, this is not best practice but for our example we will allow access from everywhere. Click confirm to proceed.

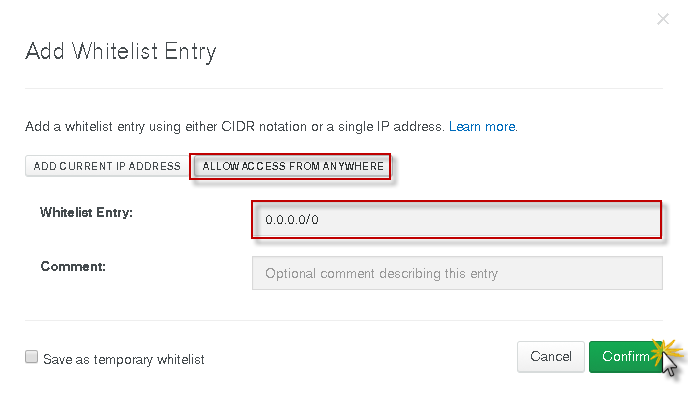


Figure 51: Configuring Cluster Step 6 Whitelist All

Verify the Active status in IP whitelist tab.

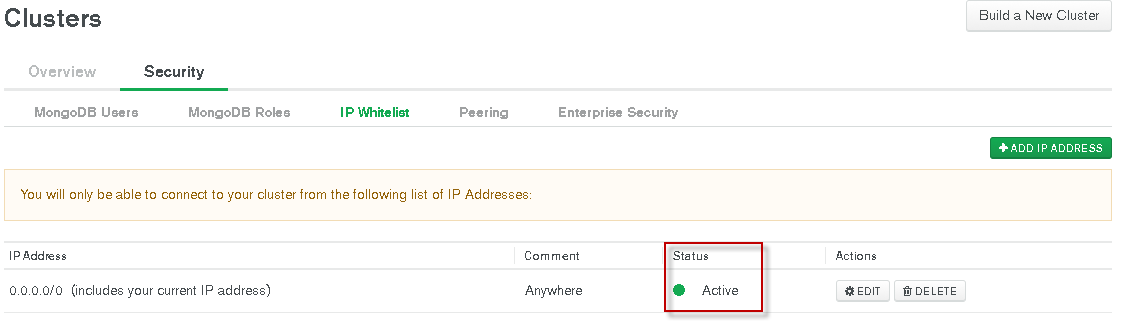


Figure 52: Configuring Cluster Step 7 Verify IP Status

### Step 2 - Connect Atlas Cluster with Mongo Compass

Now we are good to connect to Atlas cluster from Mongo shell or compass.

Head to overview tab and click on connect. MongoDB provides three options to connect to Atlas cluster. Click on Connect with MongoDB compass button to continue

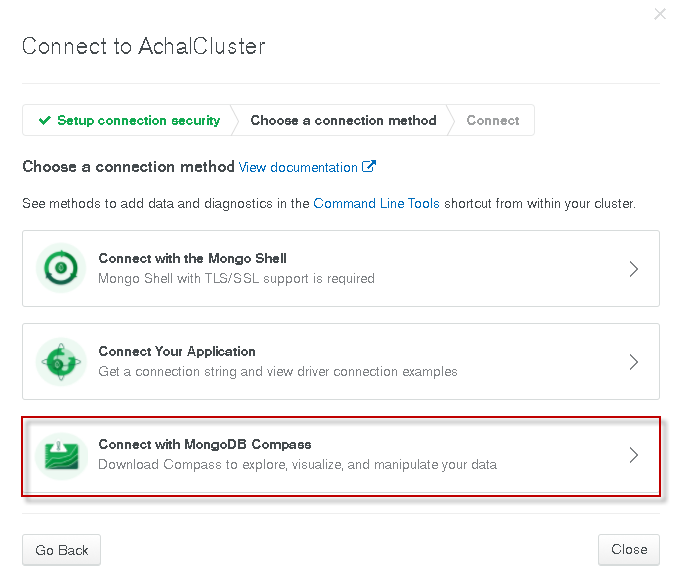


Figure 53: Connect Compass with Cluster Step 1

Select I have compass option, select the correct version as shown below and click on copy button to copy the connection string.

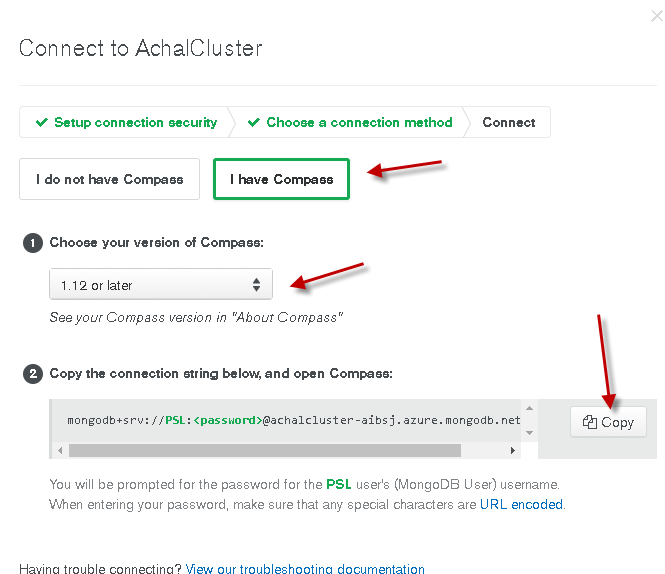


Figure 54: Connect Compass with Cluster Step 2

Now open MongoDB compass and click Yes on prompt which ask whether to use copied connection string or not. Notice that Host and Username is automatically filled by Compass. Enter the password for PSL Database user which was created initially during the Cluster setup.

(Current password – PSL)

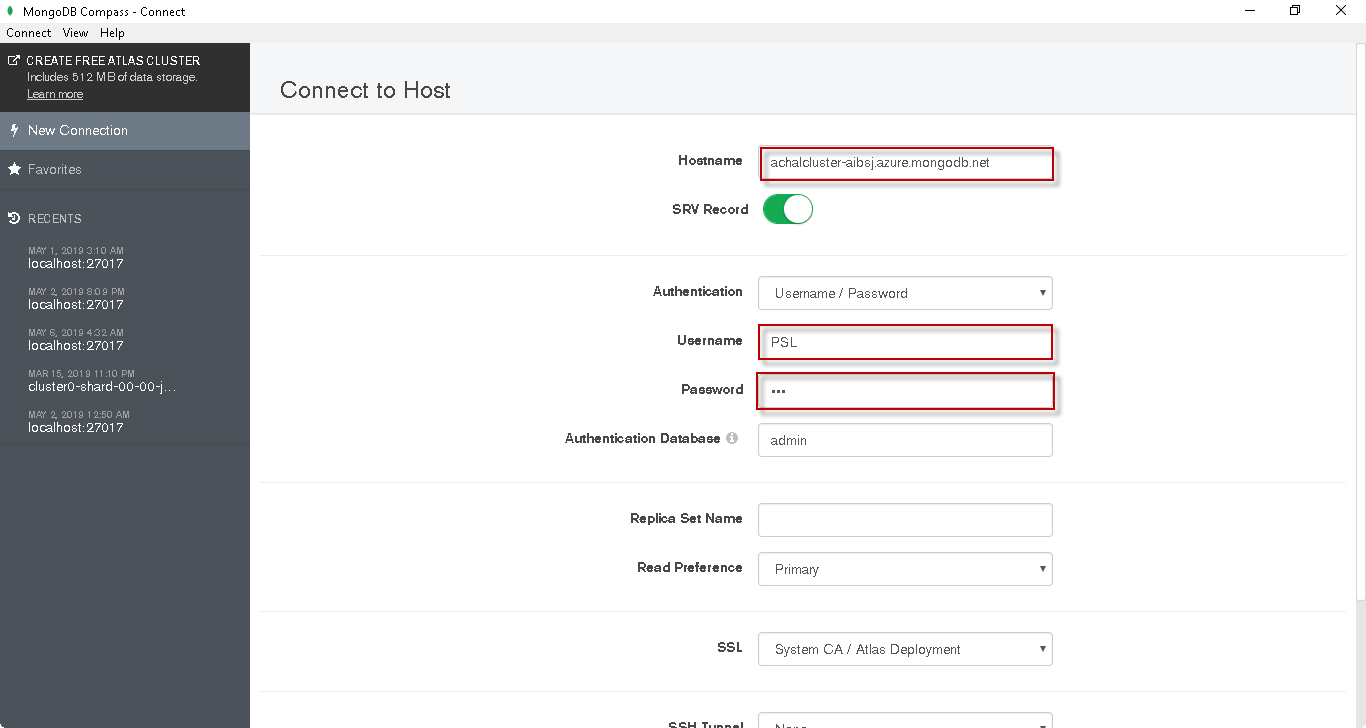


Figure 55: Connect Compass with Cluster Step 3

Give a Favourite Name to save connection details for future use and click Connect to connect to cluster.

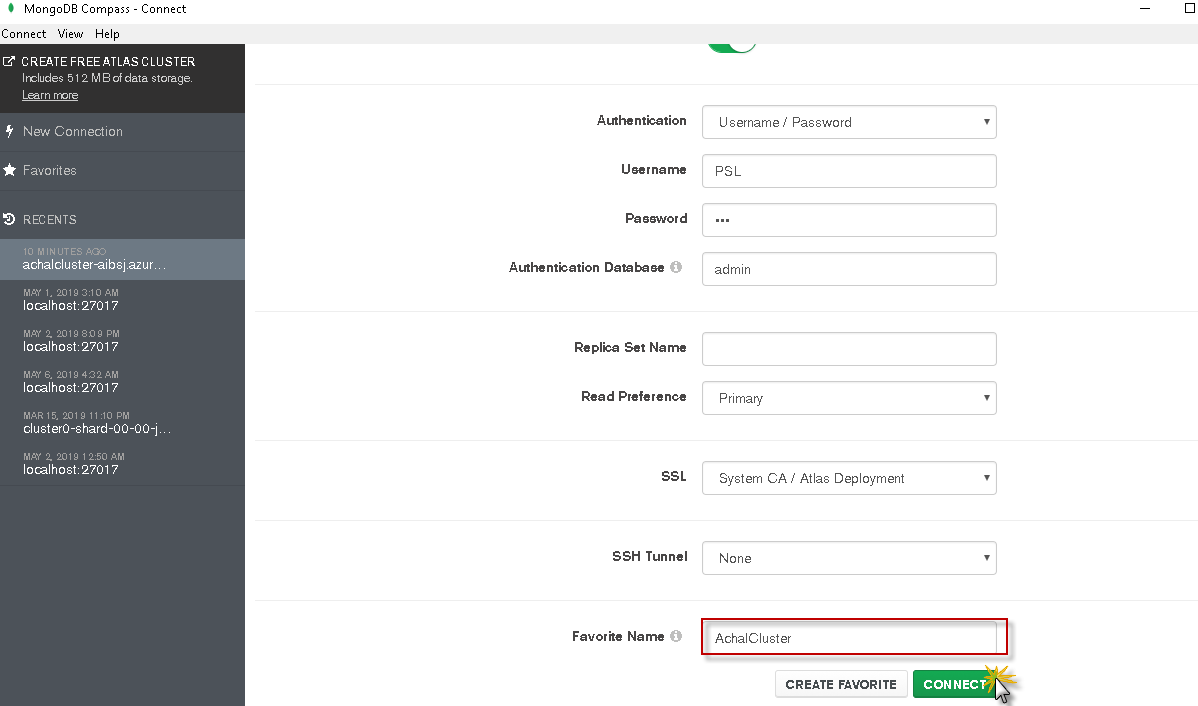


Figure 56: Connect Compass with Cluster Step 4

This will connect to the "AchalCluster" atlas cluster which has 3 nodes.

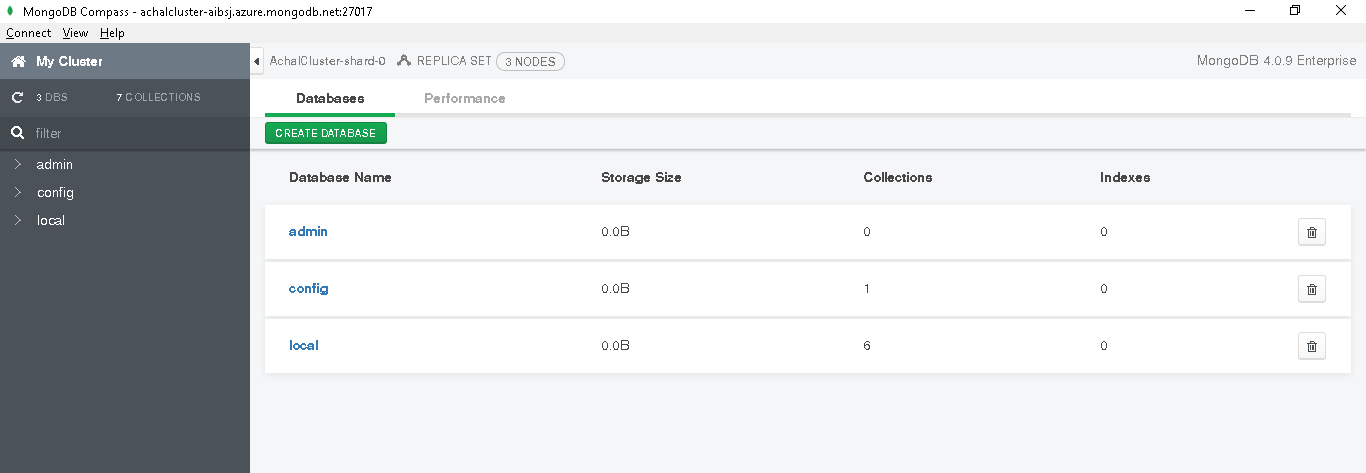


Figure 57: Connect Compass with Cluster Step 5

### Step 3 - Connecting to Atlas cluster with Mongo Shell and loading PSL database.

Go to overview tab from browser and click connect button. And select Connect with Mongo Shell option

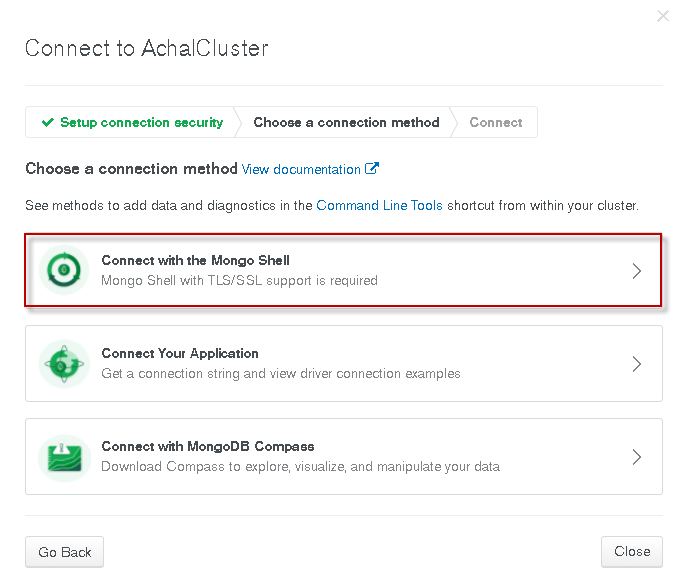


Figure 58: Connect Mongo Shell with Cluster Step 1

Select the appropriate shell version as shown below and click on copy connection string button,

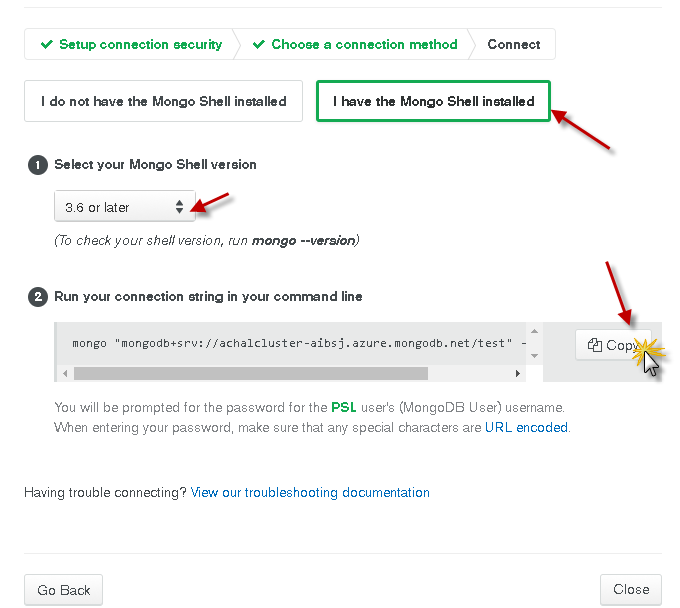


Figure 59: Connect Mongo Shell with Cluster Step 2

Navigate to Command Prompt and navigate to the location of PSL\_Cluster.js file.

Now paste the copied connection string and hit enter. Enter the Password for DB user PSL created during setup.

mongo "mongodb+srv://achalcluster-aibsj.azure.mongodb.net/test" **--**username PSL

(Current password - PSL)

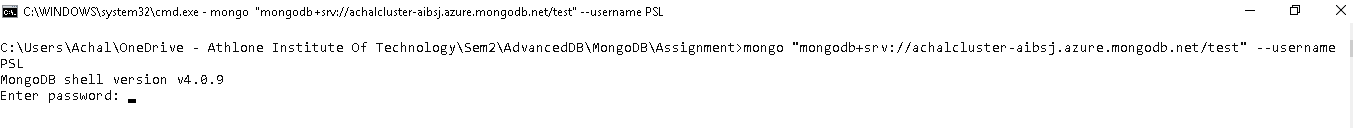


Figure 60: Connect Mongo Shell with Cluster Step 3

This will connect to one of the nodes of the cluster. Generally, the primary node is connected. Below we can see that Primary node is connected.



Figure 61: Connect Mongo Shell with Cluster Step 4

Run command **load("PSL\_Cluster.js")** on Mongo Shell. Output "true" will represent a successful load.



Figure 62: Load Data in Cluster via Mongo Shell

Verify the document count using below commands

use PSL;

db.PSL.count()

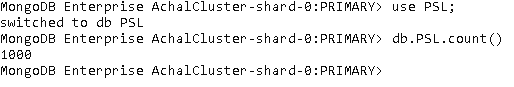


Figure 63: Verify Data

All the operations can be performed now. Let's run an example query.

### Step 4 - Example Atlas Cluster with Mongo Shell

List Forename, Surname, Height, County and Weakness of all swimmers whose height is same as the minimum height of swimmers located at Washington and have a weakness "Use of Feet".

cls

db**.**PSL**.**aggregate**([**

**{**$match**:{**County**:**"Washington"**}},**

**{**$group**:{**\_id**:** "$Weakness"**,**Min\_Height**:** **{**$min**:** "$Height"**}}},**

**{**$match**:{**\_id**:**"Use of Feet"**}},**

**{**$lookup**:{**

from**:** 'PSL'**,**

localField**:** 'Min\_Height'**,**

foreignField**:** 'Height'**,**

as**:** 'Swimmers\_MinHeight'

**}},**

**{**$project**:{**

"Swimmers\_MinHeight.Forename"**:**1**,**

"Swimmers\_MinHeight.Surname"**:**1**,**

"Swimmers\_MinHeight.Height"**:**1**,**

"Swimmers\_MinHeight.County"**:**1**,**

"Swimmers\_MinHeight.Weakness"**:**1**,**

\_id**:**0

**}}**

**])**

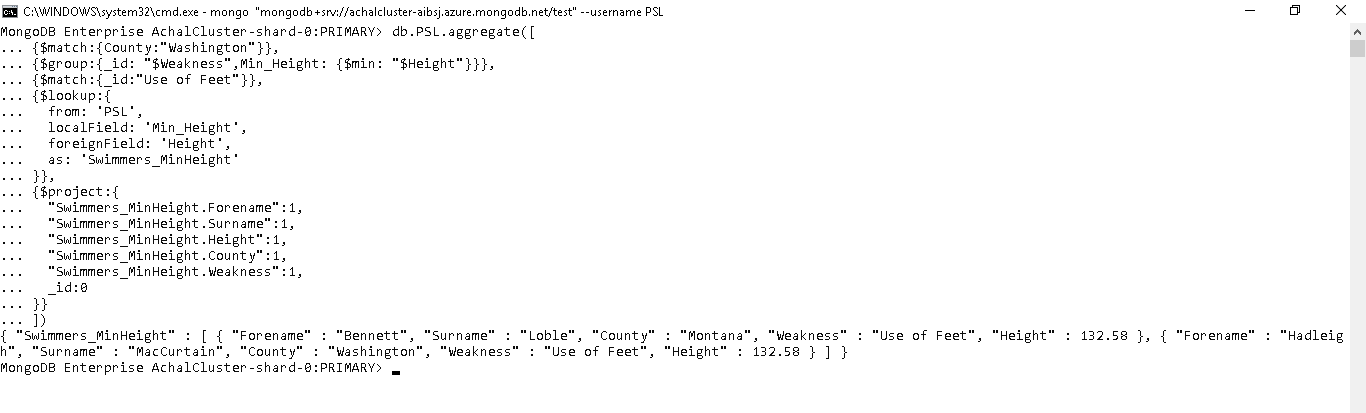


Figure 64: Example Atlas Cluster with Mongo Shell

### Step 5 - Visualizing PSL database in Compass and Atlas Cluster.

Refresh the already connected Cluster in compass and click on PSL database.

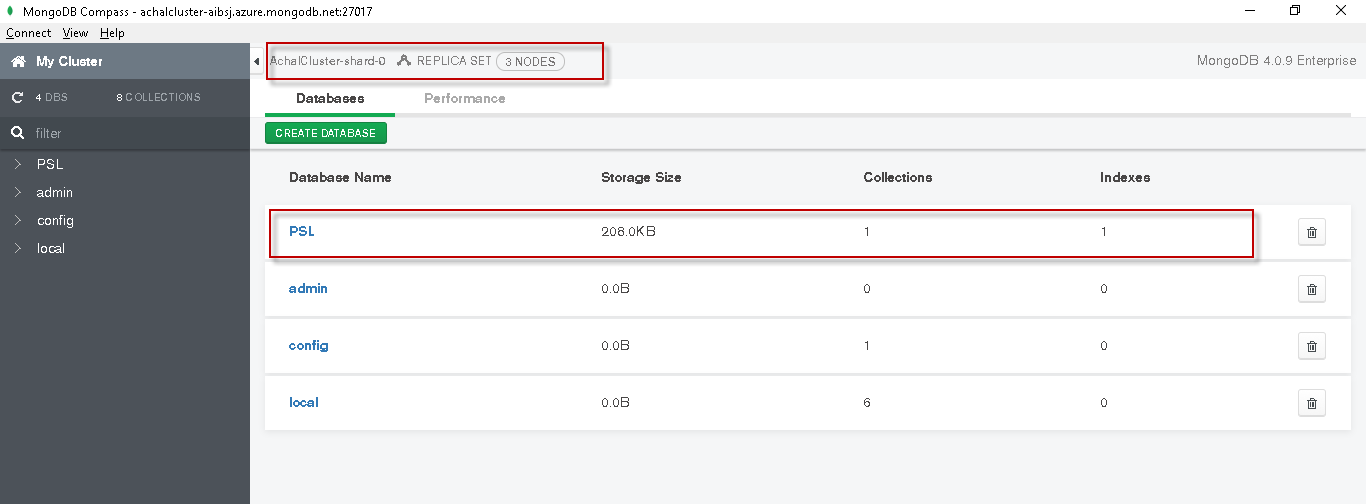


Figure 65: Visualizing PSL database in Compass and Atlas Cluster

Select the PSL collection.

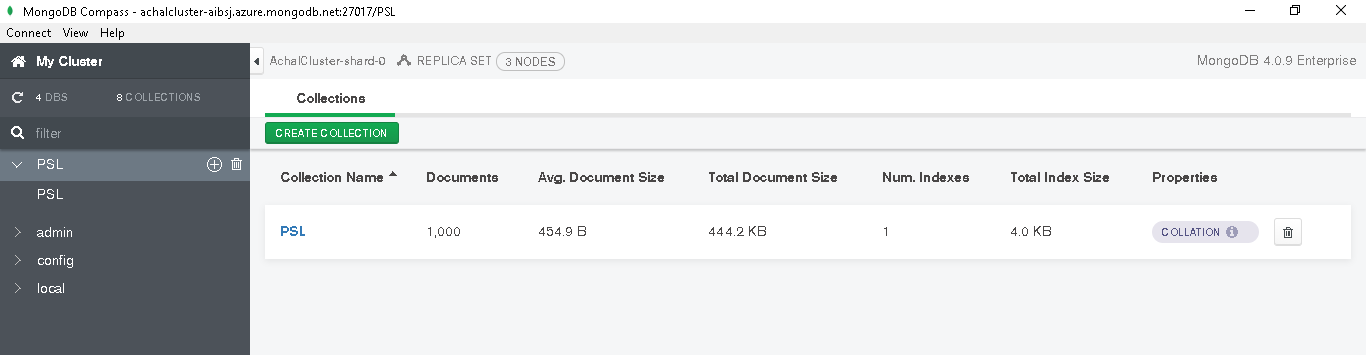


Figure 66: Visualizing PSL collection in Compass and Atlas Cluster

Compass will display 20 documents out of 1000 documents in PSL collection by default.

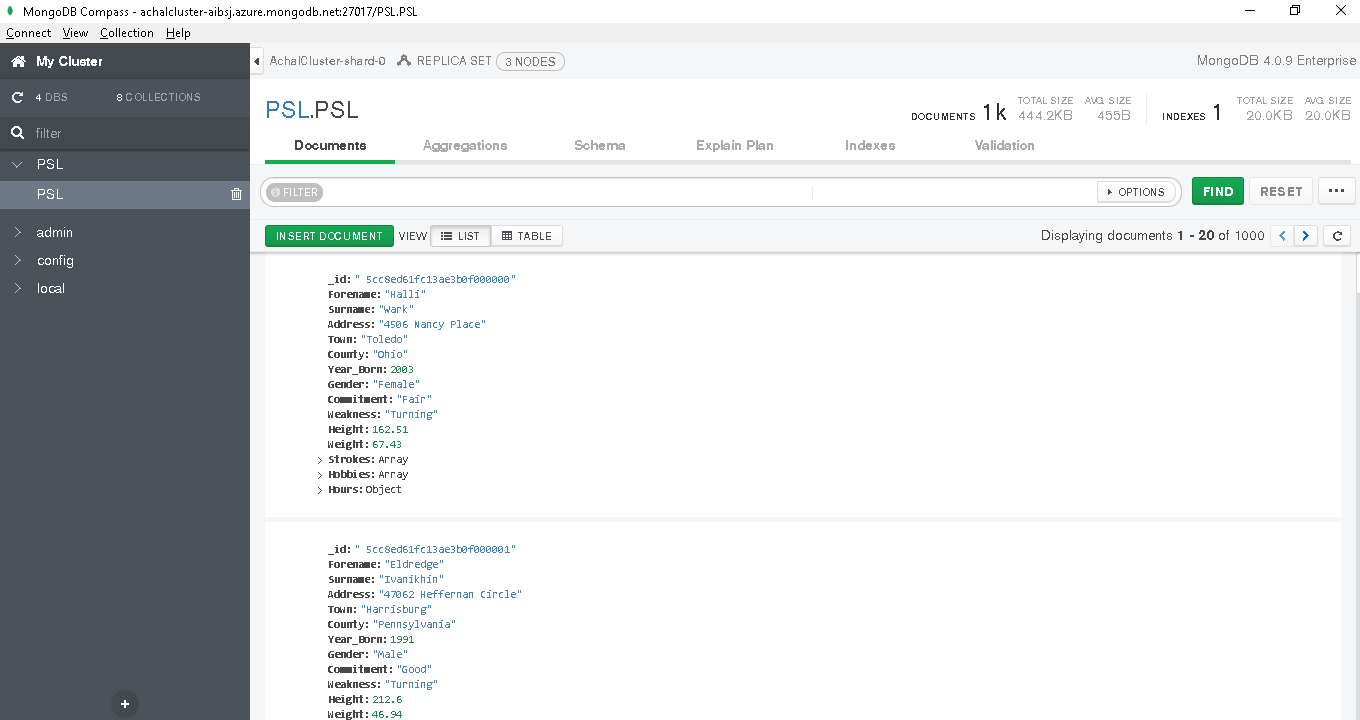


Figure 67: Visualizing PSL documents in Compass and Atlas Cluster

Now, all the operations can be performed directly on cloud cluster using compass.

Let's create an Aggregation pipeline using one of the previously used pipeline codes.

Example -1

List the most popular hobby and weakness categorised by birth year for swimmers born between 1989 and 2000. Popularity is calculated by counting number of users with each of the characteristics. Exclude records where Weakness is not recorded in database. Sort the results by birth year in reverse order.

We will use a new feature of compass which allows user to create pipeline from Mongo queries.

Head to aggregations tab and click on "…" button and select New Pipeline from text option as shown in screenshot below.

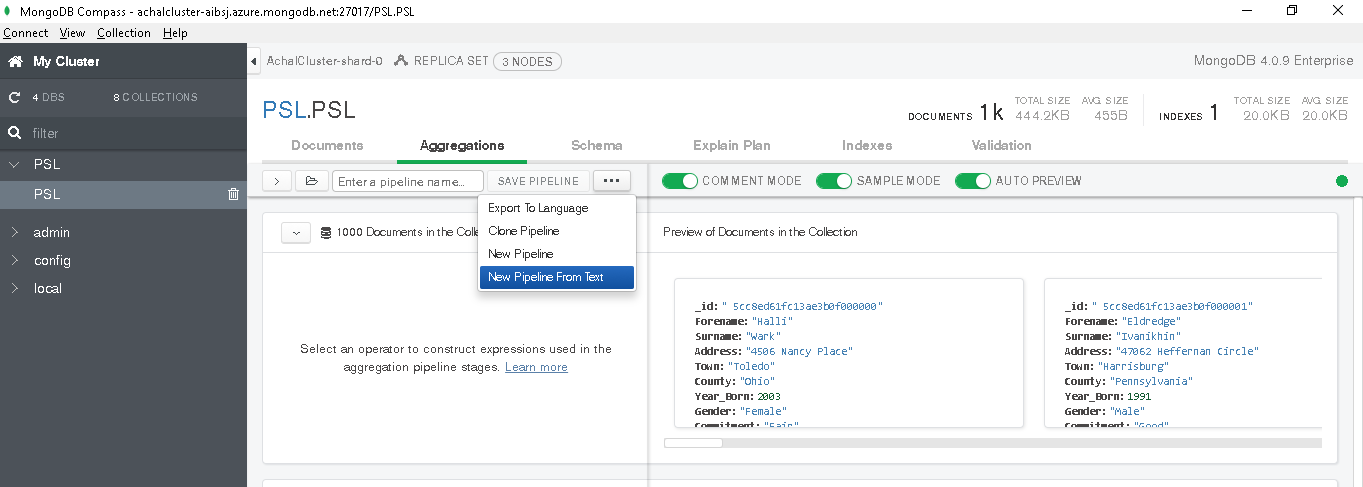


Figure 68: Aggregation Pipeline in Compass Cluster from text

Paste the below code in the window and click Create New

**[**

**{**$match**:{**

Year\_Born**:{**$gt**:**1989**,**$lt**:**2000**},**

Weakness**:{**$exists**:true}**

**}},**

**{**$unwind**:{**

path**:** "$Hobbies"

**}},**

**{**$group**:{**

\_id**:** **{**Year**:**"$Year\_Born"**,**Hobby**:**"$Hobbies"**,**Weakness**:**"$Weakness"**},**

Count\_Swimmers**:{**$sum**:**1**},**

**}},**

**{**$sort**:{**

"\_id.Year"**:**1**,** Count\_Swimmers**:-**1

**}},**

**{**$group**:{**

\_id**:**"$\_id.Year"**,**Hobby**:{**$first**:**"$\_id.Hobby"**},**

Weakness**:{**$first**:**"$\_id.Weakness"**}**

**}},**

**{**$sort**:{**

\_id**:-**1

**}},**

**]**

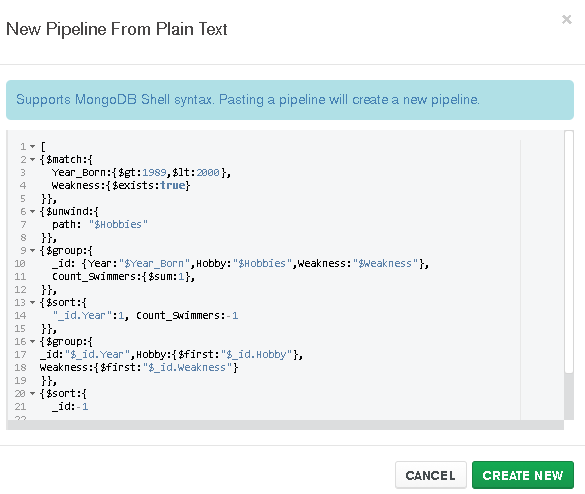


Figure 69: Aggregation Pipeline in Compass Cluster from text Step 2

Click Confirm to continue

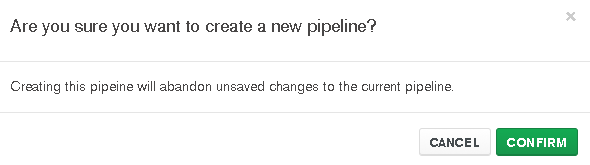


Figure 70: Aggregation Pipeline in Compass Cluster from text Step 3

Compass will automatically create multi stage pipeline from the code.

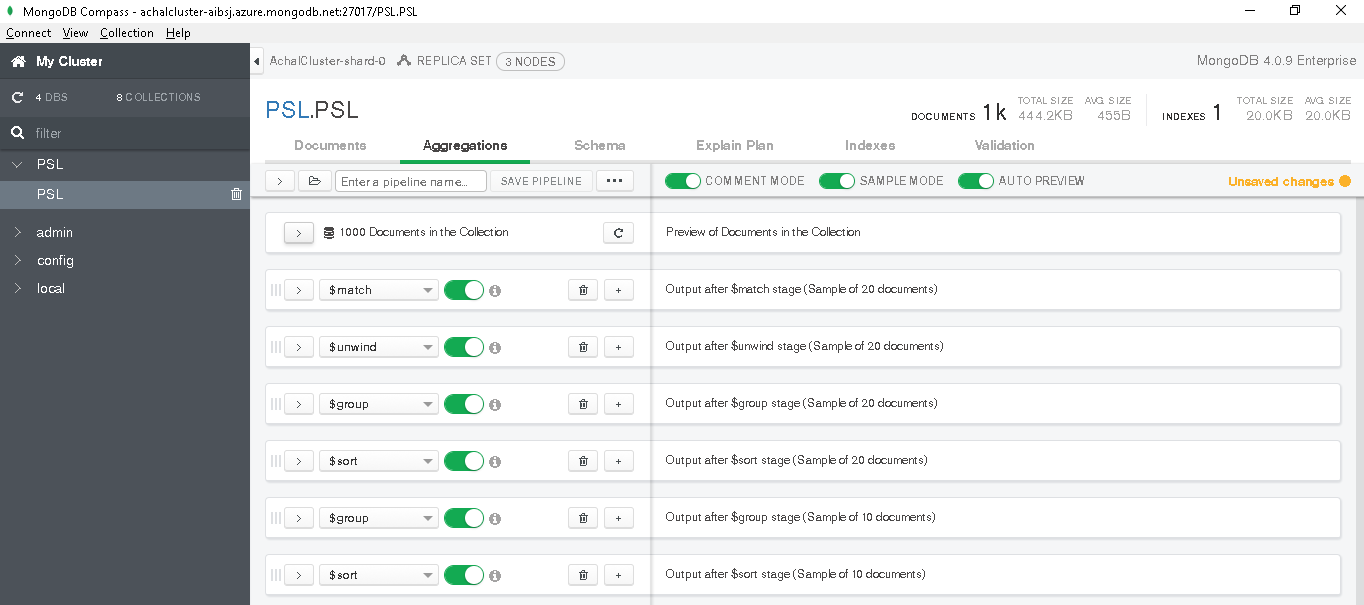


Figure 71: Aggregation Pipeline in Compass Cluster from text Step 4

Thus, Compass provides a great option for troubleshooting and modifying pipelines.

# Screencast Links

## Screencast 1 – Aggregation Pipeline in Mongo DB

<https://youtu.be/Rxp8yfA8fhk>

## Screencast 2 – Running MongoDB Atlas cluster on Mongo DB compass and Mongo DB shell.

<https://youtu.be/Rxp8yfA8fhk>