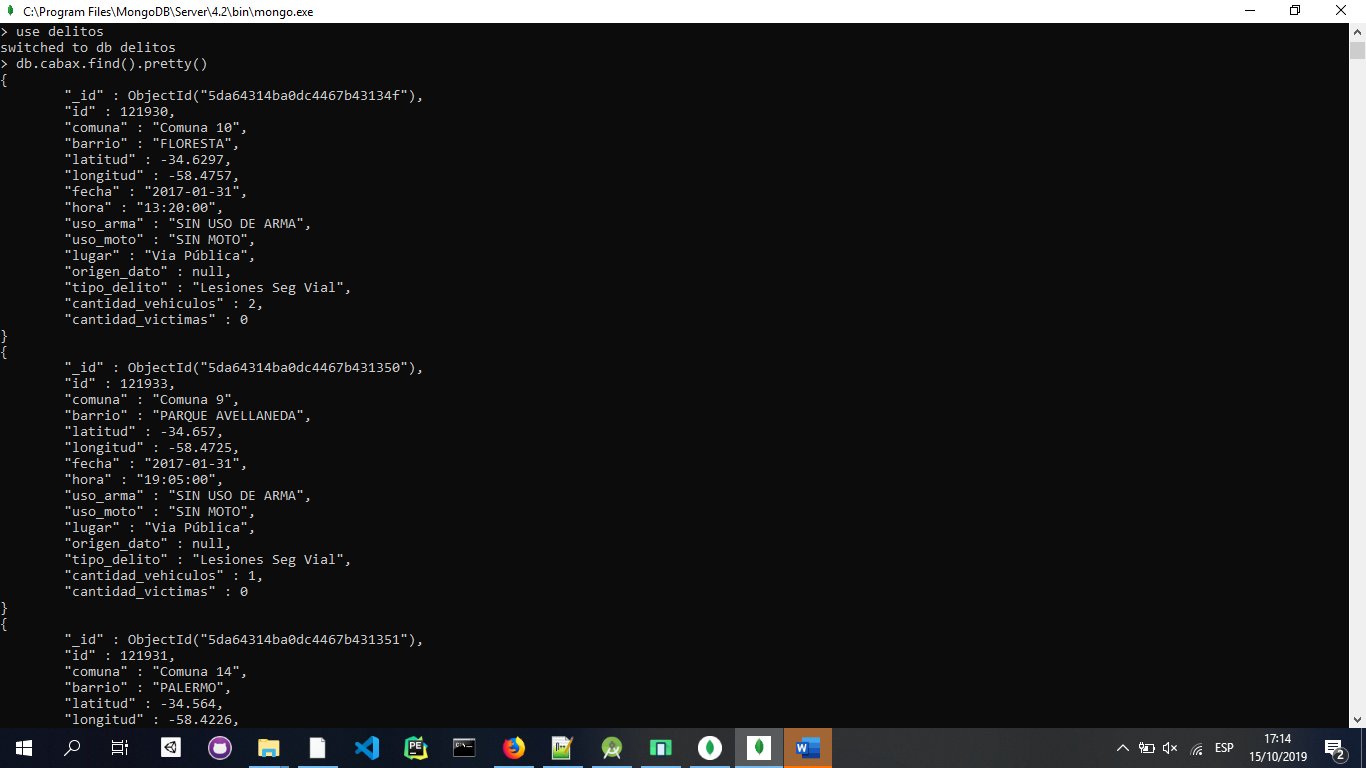
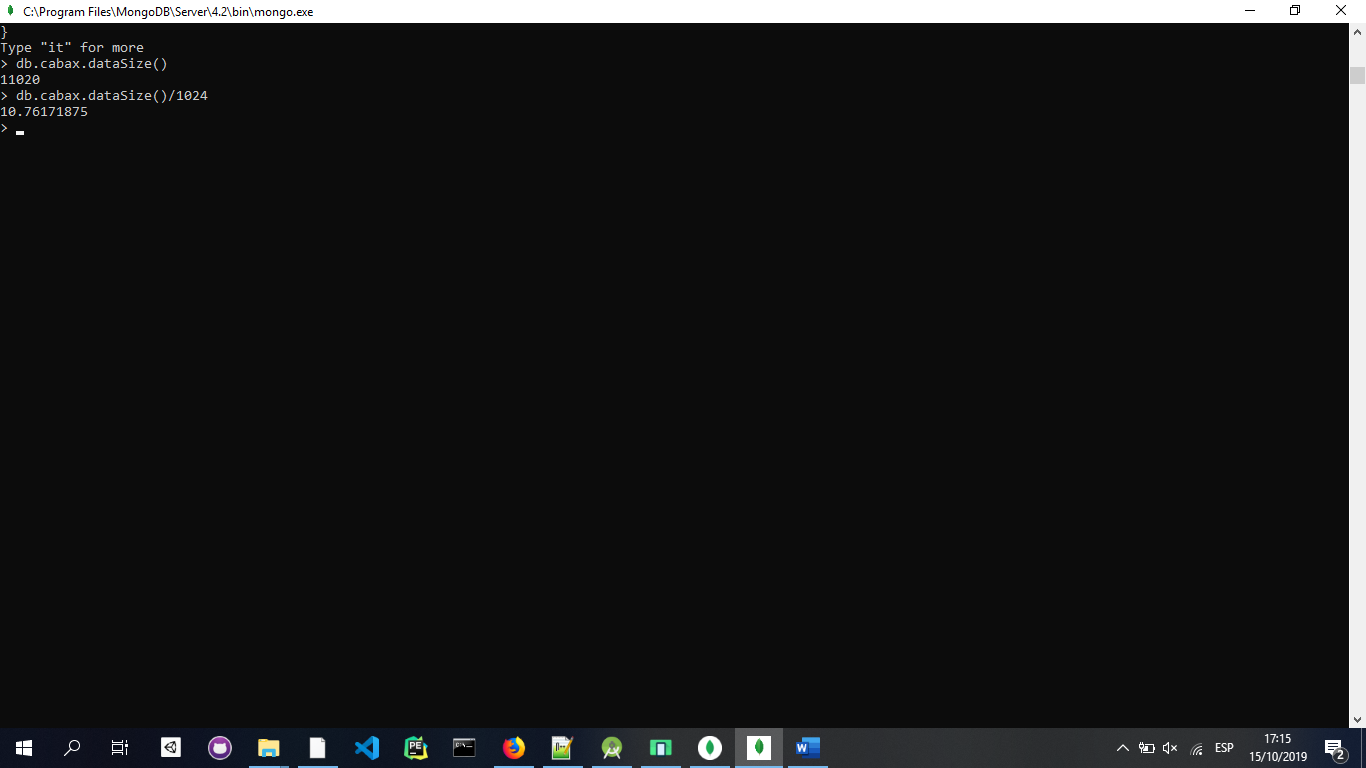
1. **Trabajo en clase: nuevos comandos sobre MongoDB**

Listamos todos los documentos de la base de datos agregada desde JSON para ver de que trata la colección:

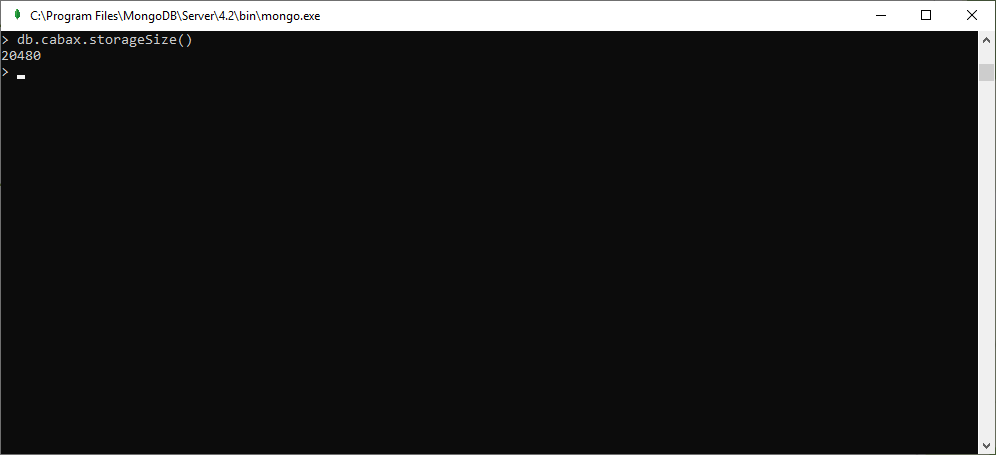
**db.cavax.find().pretty()**



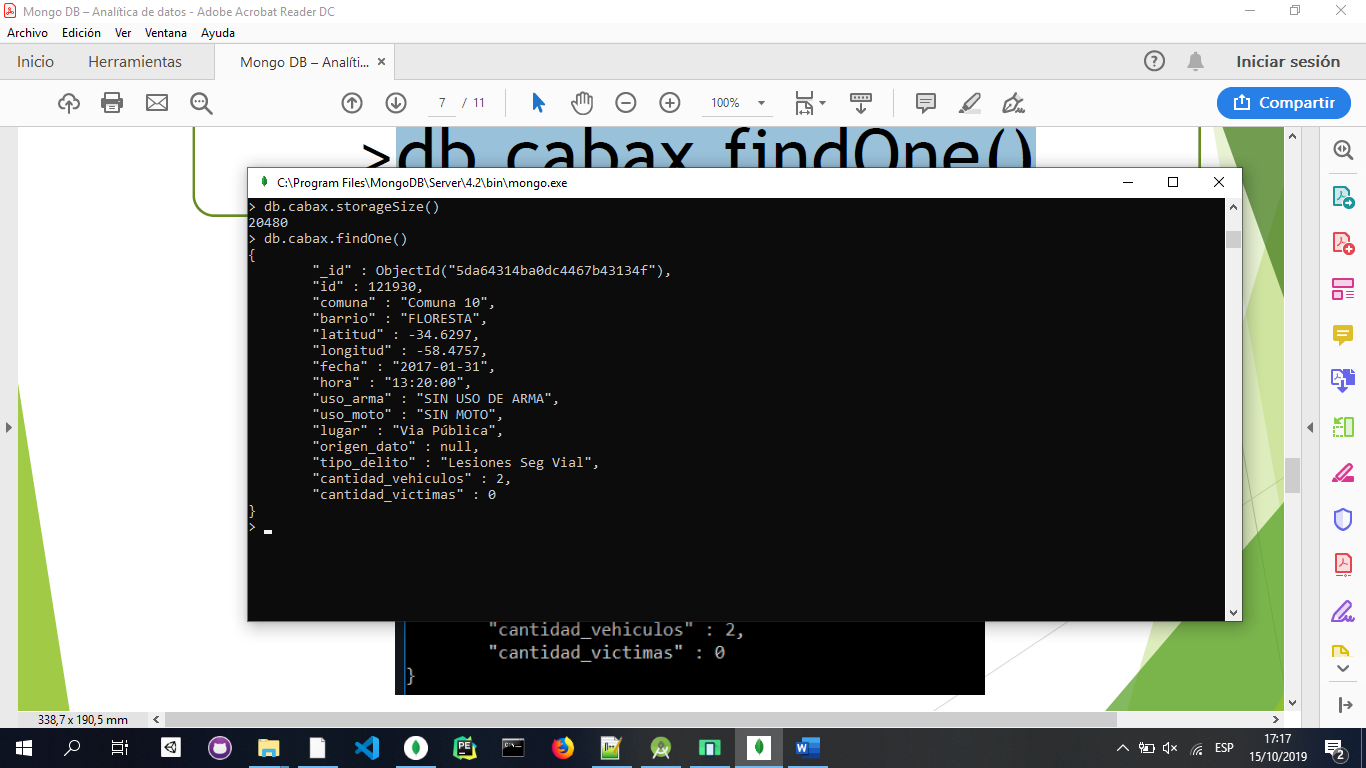
Es posible ver el tamaño de la colección en bytes usando: **db.cabax.dataSize(),** asi mismo podemos realizar una operación sobre esa instrucción. En el ejemplo de abajo se calcula el peso en KB



Para mostrar el total de documentos dentro de la colección se usa: **db.cabax.storageSize()**

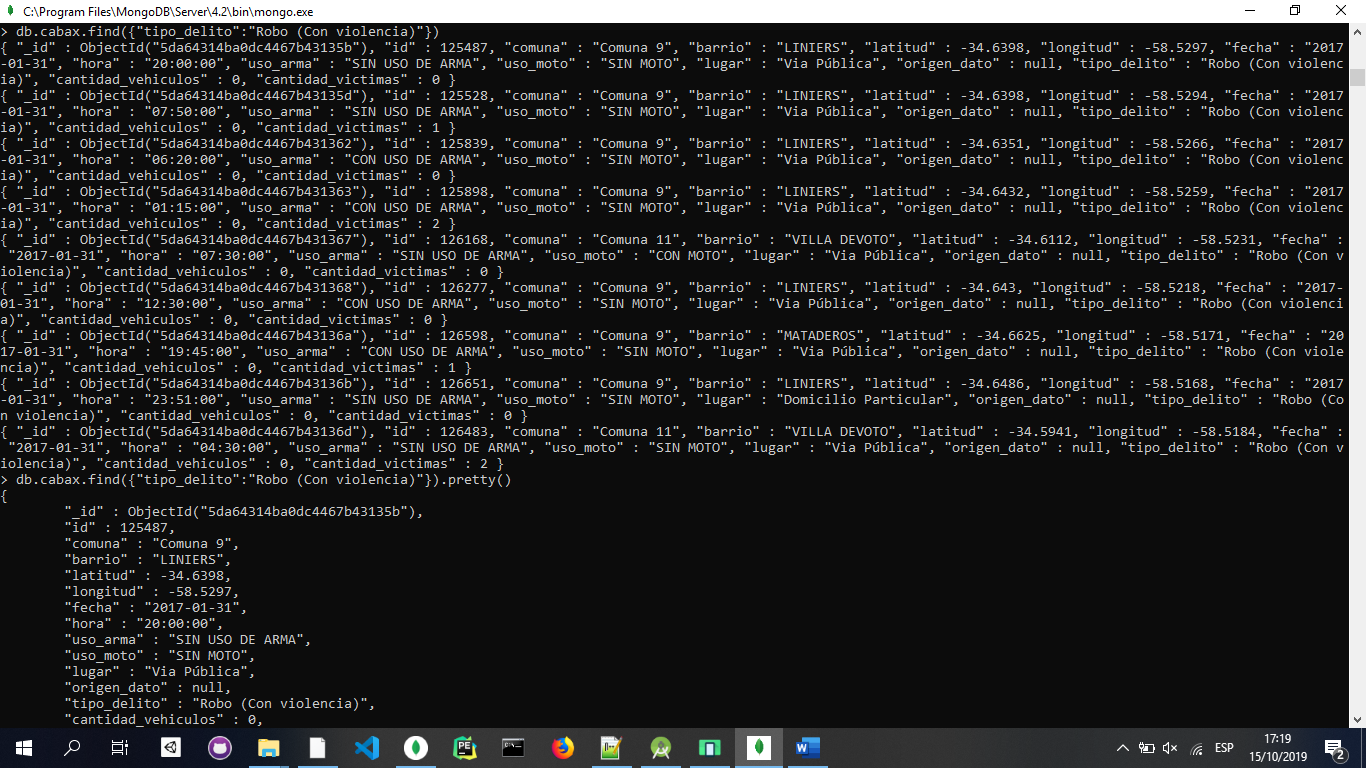


Para mostrar el primer elemento de la colección se usa: **db.cabax.findOne()**



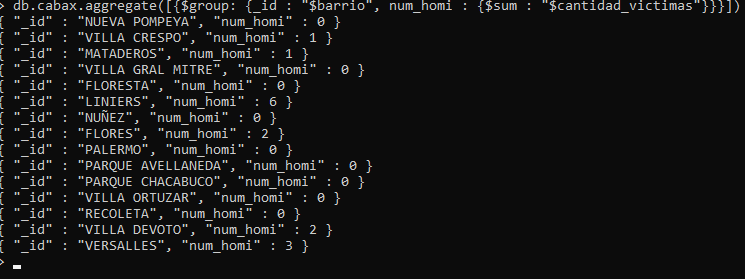
**Búsqueda con filtro de** Delitos en argentina que fueron del modo: Robo (Con violencia)

**db.cabax.find({"tipo\_delito":"Robo (Con violencia)"})**



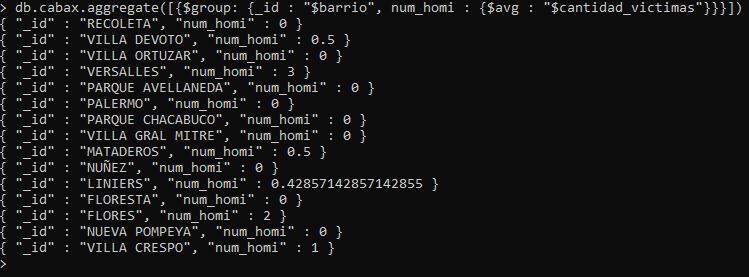
**Sumar** los registros de la cantidad de víctimas por barrio:

**db.cabax.aggregate([{$group: {\_id : "$barrio", num\_homi : {$sum : "$cantidad\_victimas"}}}])**



Operación de la Media de la Cantidad de víctimas por barrio:

db**.cabax.aggregate([{$group: {\_id : "$barrio", num\_homi : {$avg : "$cantidad\_victimas"}}}])**



1. **Revisar y aplicar 10 comandos en el dataset delitosar.json, que no se hayan visto en clase.**
   1. Describir y explicar los comandos utilizados. En el marco teórico.
   2. Aplicar y analizar los delitos del dataset
   3. Mostrar resultados
      1. **Count**

**Comando:** db.cabax.count({lugar:"Via Pública"})

**Descripción:** Se utiliza para contar los delitos que han ocurrido en la vía pública.

**Uso:**

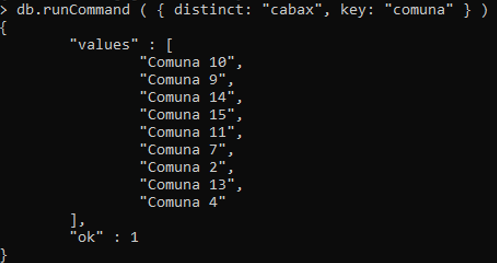


* + 1. **Distinct**

**Comando:** db.runCommand({distinct: “cabax”, key: “comuna”})

**Descripción:** Se utiliza para seleccionar todos los datos distintos de una clave en específico.

**Uso:**



* + 1. **Greater than or equal to**

**Comando:** db.cabax.find({cantidad\_victimas:{$gte:2}}).pretty()

**Descripción:** Encuentra todos los delitos cuyo número de víctimas fue mayor o igual a 2.

**Uso:**

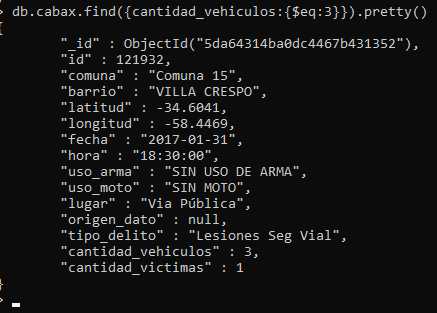


* + 1. **Less than**

**Comando:** db.cabax.find({cantidad\_vehiculos:{$eq:3}}).pretty()

**Descripción:** El comando busca los delitos en donde se hayan involucrado 3 vehículos exactamente.

**Uso:**



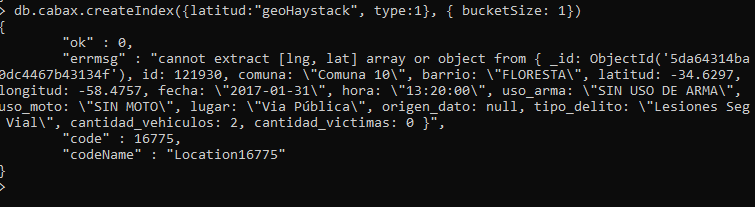
* + 1. **Create Index**

**Comando:** db.cabax.createIndex({latitud:"geoHaystack", type:1}, { bucketSize: 1})

**Descripción:** Crea un índice especializado para utilizar una búsqueda geográfica

**Uso:**

En este caso el comando no fue exitoso porque se necesita [longitud, latitud] y esos datos en los documentos de la colección **cabax** estan separados



* + 1. **GeoSearch**

**Comando:** db.runCommand({ geoSearch : "cabax", near: [ -34.6297, -58.4757 ], maxDistance : 6, search : { cantidad\_victimas : 1 }, limit : 10 })

**Descripción:** Busca los delitos alrededor del punto [ -34.6297, -58.4757] cuyo número de víctimas es 1.

**Uso:**

De la mano con el comando anterior, este comando necesita crear un índice “geoHaystack”, el cual no se pudo agregar en la parte anterior por la estructura de los documentos



* + 1. **Sort**

**Comando:** db.cabax.find({comuna: "Comuna 7"}).sort({cantidad\_victimas: 1}).pretty()

**Descripción:** El comando se utiliza para ordenar los resultados en orden ascendente.

**Uso:**

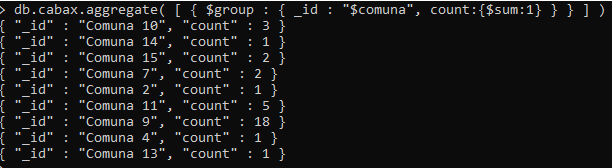


* + 1. **Group**

**Comando:** db.cabax.aggregate( [ { $group : { \_id : "$comuna", count:{$sum:1} } } ] )

**Descripción:** Cuenta la cantidad de delitos u los agrupa por comuna.

**Uso:**

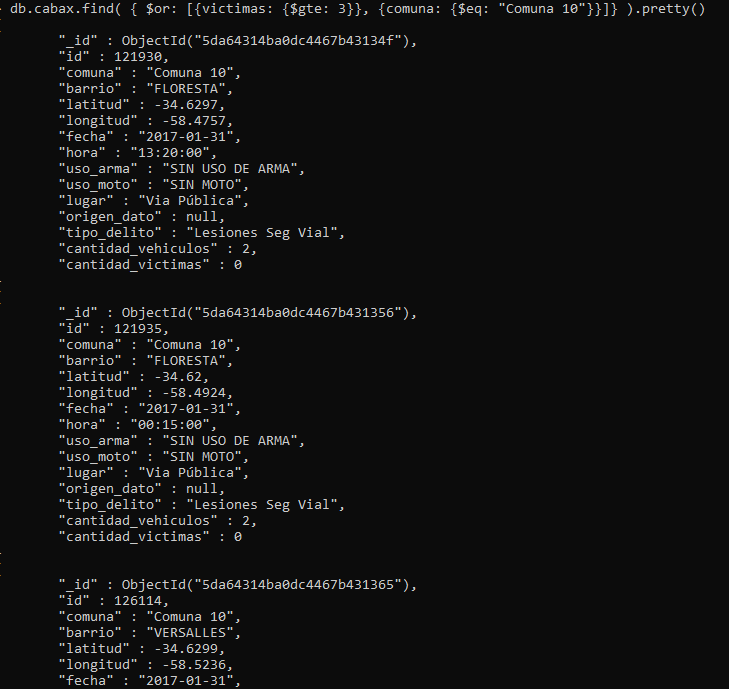


* + 1. **OR (Operación lógica)**

**Comando:** db.cabax.find( { $or: [{victimas: {$gte: 3}}, {comuna: {$eq: "Comuna 10"}}]} ).pretty()

**Descripción:** Selecciona los delitos que tengan víctimas iguales o mayores atres O que hayan ocurrido en la comuna 10.

**Uso:**



* + 1. **MapReduce**

**Comando:**

db.runCommand({   
 mapReduce: "cabax",   
 map: function(){

emit("MAP: " + this.comuna, (this.cantidad\_victimas+this.cantidad\_vehiculos)/2 )

},   
 reduce: function(key, values){

return Array.sum(values);

},   
 out: {inline:1}   
})

**Descripción:** Obtenemos la media entre la suma de victimas y vehículos robados por cada delito (MAP) y luego sumamos todos los valores generados en el vector reduciéndolo a un valor escalar (REDUCE).

**Uso:**

