

Tarea 4 - Almacenamiento y Consultas de Datos en Big Data

Autor: Samuel Solis Palacios

Tutor: Frank Rodríguez Achury

Universidad Nacional Abierta y A Distancia

ECBTI

Ingeniería De Sistemas

BigData

Grupo: 202016911_45

Cali, Valle Del Cauca

19 de noviembre del 2025

INTRODUCCIÓN

Las bases de datos NoSQL son conocidas por su flexibilidad y su capacidad para manejar grandes volúmenes de datos que no se ajustan a un esquema rígido. A diferencia de las bases de datos tradicionales, NoSQL permite trabajar con datos variados y semi-estructurados de una manera más eficiente. MongoDB, que es una base de datos orientada a documentos, guarda la información en formato JSON, lo que hace que sea más fácil consultarla, modificarla y analizarla. Con sus herramientas de filtrado y agregación, puedes explorar tendencias y generar información valiosa rápidamente, lo que la convierte en una opción perfecta para gestionar datos complejos y en constante cambio.

INVESTIGACIÓN

Realizar un ensayo comparando los diferentes tipos de bases de datos NoSQL (clave-valor, documentos, columnas, grafos). discuta las ventajas e inconvenientes de cada tipo, así como sus casos de uso más apropiados.

Las bases de datos NoSQL son bases de datos no relacionales para datos semiestructurados o sin un esquema estricto, nacieron como una solución a las limitaciones que tenían las bases de datos relacionales y se necesitaba un método más flexible para trabajar con los datos dando ventajas como, productividad del desarrollo de aplicaciones, mejora en datos a gran escala y su ejecución sobre clústeres.

Modelos de bases de datos NoSQL:

Las BDD SQL se categorizan según su modelo de datos. Existen 4 familias e incluso algunas bases de datos comparten sus características:

- **Bases de datos clave-valor:** Riak, Redis, Dynamo, Voldemort.

Se almacenan los datos en pares clave, valor. La clave es única y el valor puede ser cualquier tipo de dato o un objeto. Se suele usar un mecanismo tipo tabla hash o sistema de índices y la clave se mapea a un nodo que contiene el valor.

- Ventajas:

Puede almacenar grandes cantidades de datos, ofrece una opción rápida y escalable, tiene un modelo sencillo, y flexible con el tipo de dato que se puede almacenar.

- Desventajas:

No tiene capacidad de reversión, no está optimizado para la búsqueda haciendo que no se pueda filtrar fácilmente por campo del valor, no es una base de datos para modelos complejos es ideal cuando la lógica es más simple.

- Casos de uso ideales:

Caché de sesiones, configuraciones, perfil de usuario simple, Sistemas donde la clave es evidente (ID de usuario, nombre de recurso) y basta recuperar el valor completo, Aplicaciones con necesidad de latencia ultra-baja y mucho volumen de operaciones simples por segundo.

- **Bases de datos orientadas a documento:** MongoDB, CouchDB.

En este modelo los datos se almacenan como documentos, de forma semiestructurada de forma que los documentos tienen una clave que lo identifica y este modelo nos permite ver la información que hay dentro del documento y manipularla.

- Ventajas:

Flexibilidad en el esquema para modelos que evoluciona con el tiempo, muy bueno para consultas ya que se pueden indexar atributos, hacer agregaciones y los almacenes de documentos permiten la inserción, actualización y consulta de registros completos utilizando un formato (JSON).

- Desventajas:

Mas flexible que clave-valor, pero menos que las bases relacionales para relaciones complejas, si la consulta se vuelve muy compleja se puede volver ineficiente comparado con sistemas analíticos.

- Casos de uso ideales:

Catálogos de productos donde cada producto puede tener atributos diferentes, Logs, eventos, datos semiestructurados/variable que no encajan bien en tabla rígida.

- **Bases de datos basadas en columnas: Cassandra, Hypertable, HBase,**

SimpleDB.

En las bases de datos de familias de columnas, los datos se almacenan en celdas de columnas, agrupadas en familias de columnas. Estas bases de datos se implementan como mapas de mapas ordenados anidados multidimensionales. El mapa más interno constituye una versión de los datos identificados por una marca de tiempo y almacenados en una celda. Una celda se asigna a una columna que, a su vez, se asigna a una familia de columnas.

- Ventajas:

Escalada horizontal muy buena, puede almacenar matrices, excelente rendimiento para cargas intensas de escritura/lectura, logging, series temporales, IoT, etc.

- Desventajas:

Modelado más complejo que documentos o clave-valor, no está diseñado para operaciones analíticas complejas o relacionar muchos registros de diferente naturaleza, se puede encontrar menos intuitivo que modelos más simples.

- Casos de uso ideales:

Telemetria, sistemas de eventos a gran escala, IoT, analíticas operacionales en tiempo real.

- **Bases de datos de grafos: Neo4J, Infinite Graph.**

Un gráfico de propiedades consta de nodos y relaciones entre los nodos. Tanto los nodos como las relaciones pueden almacenar una serie de atributos llamados propiedades. Los nodos son las entidades del gráfico y se pueden etiquetar con etiquetas que se pueden usar para proporcionar contexto y metadatos al nodo. Las relaciones proporcionan conexiones dirigidas, bidireccionales y con nombre entre dos nodos. Puede haber más de una relación entre dos nodos y cada relación tiene una dirección, un tipo, un nodo inicial y un nodo final.

- Ventajas:

Muy buena para relaciones complejas y profundas, flexible en cuanto al esquema porque se pueden añadir nuevos tipos de nodos/relaciones sin grandes migraciones, es ideal para descubrimiento de patrones, recomendaciones, fraude, análisis de redes, etc.

- Desventajas:

No tan óptima para cargas de trabajo simples, modelar correctamente un grafo y escoger particionamiento, indexación, puede requerir más especialización y escalabilidad horizontal compleja.

FASE 2: MONGODB

Caso: Dataset (CSV) de los datos de homicidios por año. [HOMICIDIO | Datos Abiertos](#)

[Colombia](#)

Elegí esta colección de homicidios con documentos por evento o por agrupación geográfica, para hacer análisis de violencia, tendencias, mapas con densidad de homicidios, agregaciones por departamento, A demás que se tratan de casos reales ya que los datos se obtuvieron de fuentes oficiales.

Este es el diseño de los documentos:

```
{  
    "FECHA HECHO": "01/01/2003",  
    "COD_DEPTO": 11,  
    "DEPARTAMENTO": "BOGOTA D.C.",  
    "COD_MUNI": 11001,  
    "MUNICIPIO": "BOGOTA D.C.",  
    "ZONA": "URBANA",  
    "SEXO": "MASCULINO",  
    "CANTIDAD": 1  
}
```

1. Creación e importación.

Se crea la Base de datos (BD_), se crea una colección llamada “Homicidios” y se importan los documentos del dataset.

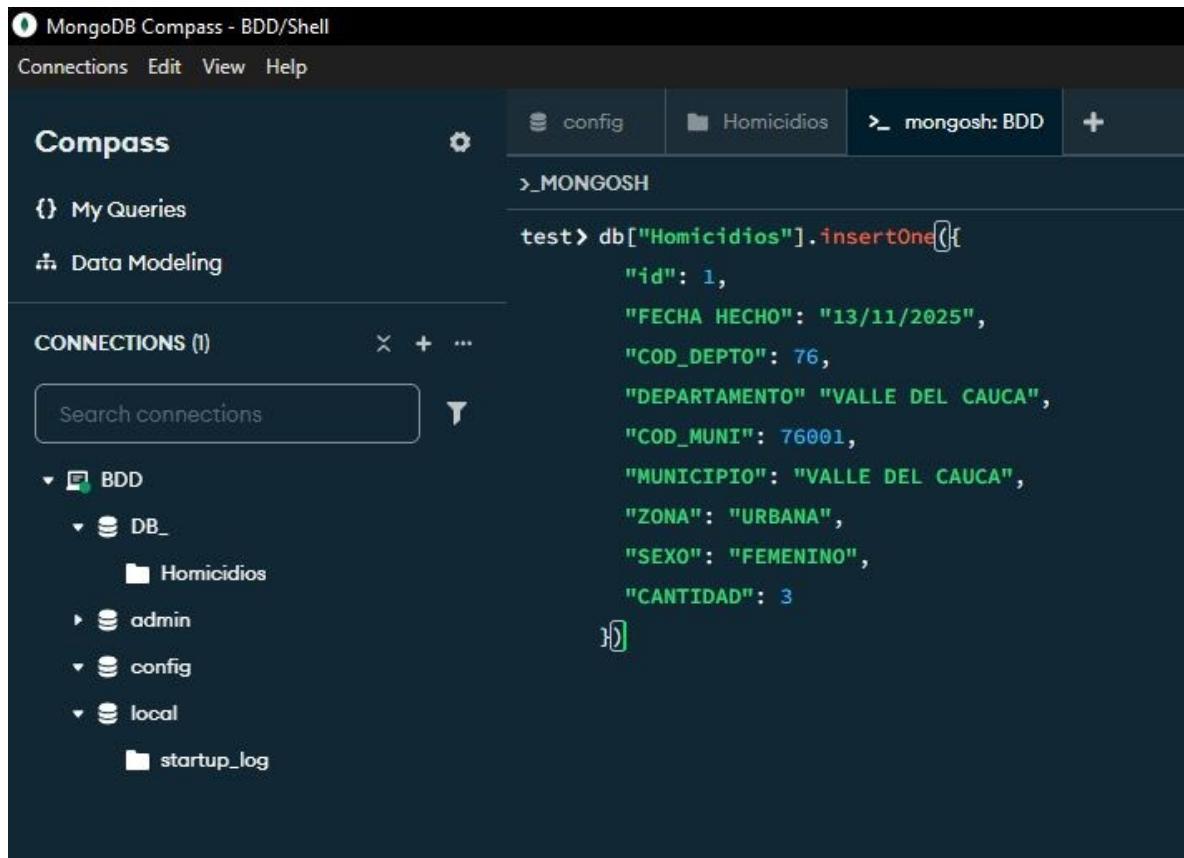
The screenshot shows the Compass MongoDB interface. On the left, the sidebar displays 'Connections' with 'BDD' expanded, showing 'DB_' and 'Homicidios'. The main area shows the 'Homicidios' collection under 'BDD > DB_ > Homicidios'. The 'Documents' tab is active, showing 331K documents. A specific document is selected, displaying its fields:

```
_id: ObjectId('692384fa72f43316dc8a83ea')
FECHA HECHO: "01/01/2003"
COD_DEPTO: 11
DEPARTAMENTO: "BOGOTA D.C."
COD_MUNI: 11001
MUNICIPIO: "BOGOTA D.C."
ZONA: "URBANA"
SEXO: "MASCULINO"
CANTIDAD: 1
```

2. Consultas básicas.

a. Insertar.

Si se requiere añadir un documento con la información de un caso nuevo utilizamos (insertOne).

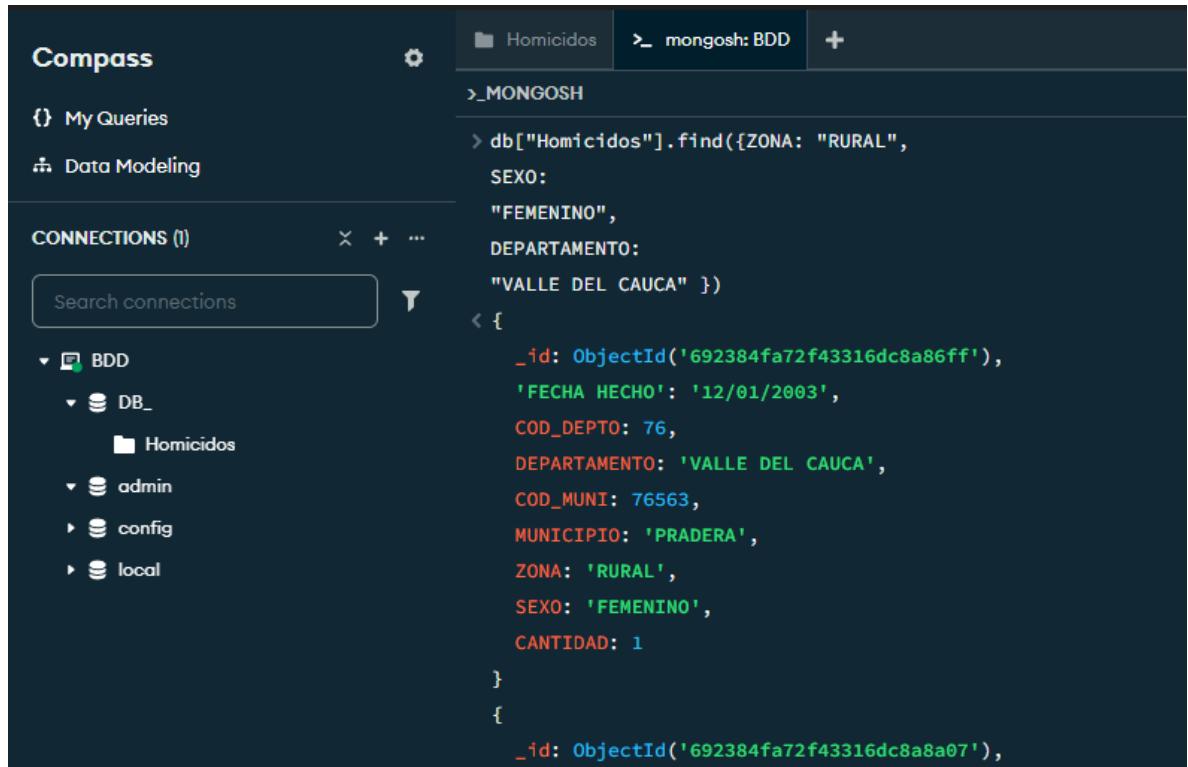


The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays connections: BDD (selected), DB_, Homicidios, admin, config, local, and startup_log. The main area shows the MONGOSH shell with the following command and its result:

```
test> db["Homicidios"].insertOne({  
    "id": 1,  
    "FECHA HECHO": "13/11/2025",  
    "COD_DEPTO": 76,  
    "DEPARTAMENTO": "VALLE DEL CAUCA",  
    "COD_MUNI": 76001,  
    "MUNICIPIO": "VALLE DEL CAUCA",  
    "ZONA": "URBANA",  
    "SEXO": "FEMENINO",  
    "CANTIDAD": 3  
})
```

b. Consultar:

Necesito consultar las muertes femeninas en Valle del cauca en zonas rurales:



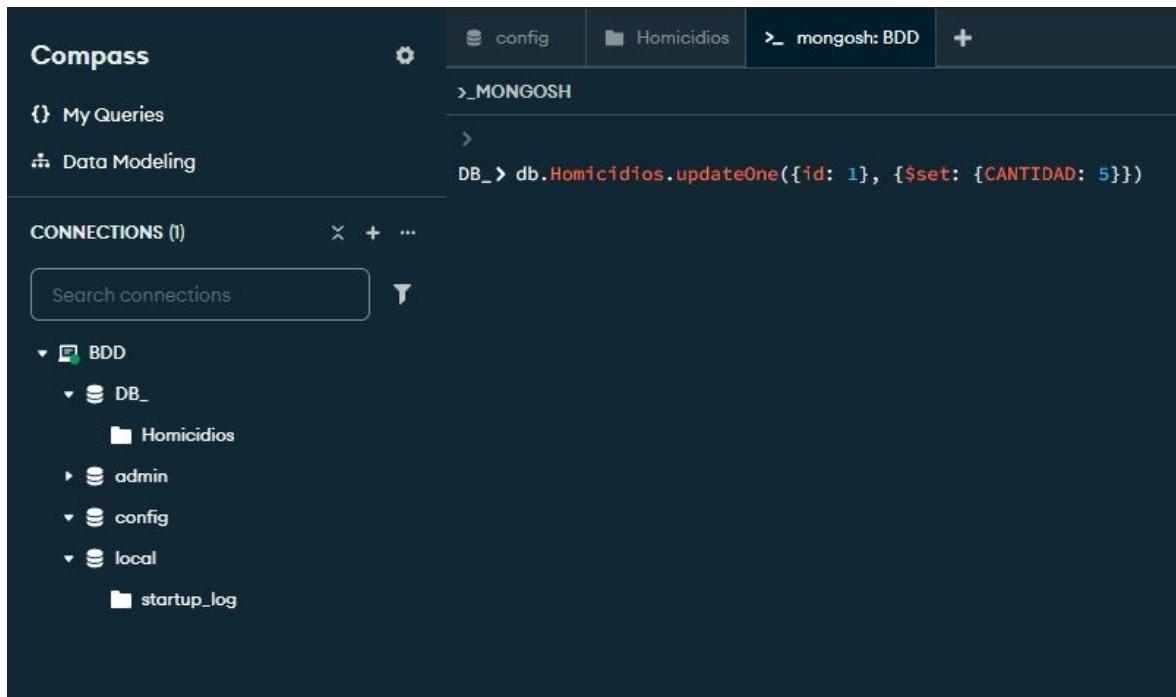
The screenshot shows the Compass MongoDB interface. On the left, the 'CONNECTIONS' sidebar lists 'BDD' (selected), 'DB_', 'admin', 'config', and 'local'. Under 'BDD', it shows 'Homicidios'. On the right, the main area displays a MongoDB query in the 'mongosh: BDD' tab:

```
>_MONGOSH
> db["Homicidios"].find({ZONA: "RURAL",
  SEXO:
    "FEMENINO",
  DEPARTAMENTO:
    "VALLE DEL CAUCA" })
< [
  {
    _id: ObjectId('692384fa72f43316dc8a86ff'),
    'FECHA HECHO': '12/01/2003',
    COD_DEPTO: 76,
    DEPARTAMENTO: 'VALLE DEL CAUCA',
    COD_MUNI: 76563,
    MUNICIPIO: 'PRADERA',
    ZONA: 'RURAL',
    SEXO: 'FEMENINO',
    CANTIDAD: 1
  }
]
{
  _id: ObjectId('692384fa72f43316dc8a8a07'),
```

c. *Actualización:*

Necesito modificar un archivo sobre la cantidad de personas que fueron asesinadas.

Utilizo (updateOne).



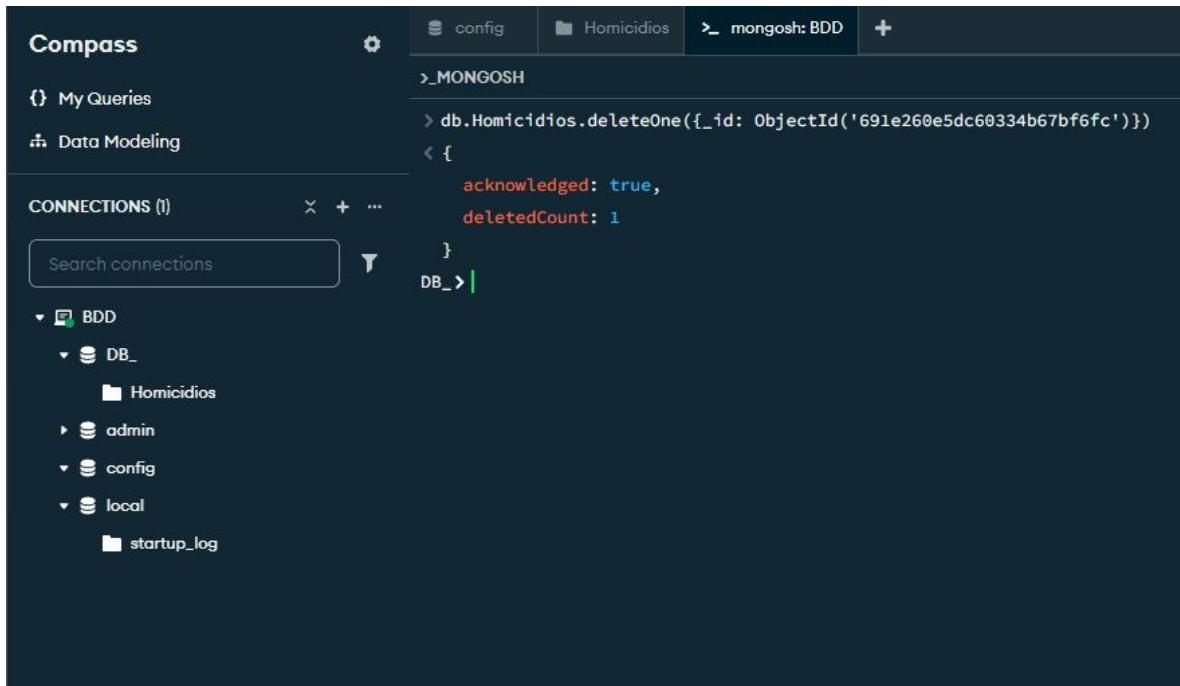
The screenshot shows the MongoDB Compass interface. In the top right, there are tabs for 'config', 'Homicidios', and 'mongosh: BDD'. Below them is a search bar with the text '>_MONGOSH'. Underneath is a code editor with the following command:

```
DB_> db.Homicidios.updateOne({id: 1}, {$set: {CANTIDAD: 5}})
```

On the left, the 'Data Modeling' section shows a tree view of connections. The 'BDD' connection is expanded, showing the 'DB_' database which contains 'Homicidios', 'admin', 'config', 'local', and 'startup_log' collections.

d. *Eliminar:*

Se pide que elimine un archivo erróneo de la base, utilizamos (deleteOne).



The screenshot shows the Compass MongoDB interface. On the left, there's a sidebar with 'My Queries' and 'Data Modeling' sections, and a 'CONNECTIONS (1)' section listing 'BDD'. The main area shows the 'mongosh: BDD' tab active, displaying the command and its execution results:

```
>_MONGOSH
> db.Homicidios.deleteOne({_id: ObjectId('691e260e5dc60334b67bf6fc')})
< {
  acknowledged: true,
  deletedCount: 1
}
DB_> |
```

3. Consultas con operadores.

a. *Municipios de valle de cauca que no sean ni Cali, ni yumbo.*

```

Homicidios > mongosh: BDD +
>_MONGOSH
DB_> db["Homicidios"].find({DEPARTAMENTO: "VALLE DEL CAUCA"}, { MUNICIPIO: {$ne: ["CALI", "YUMBO"] }})

```

b. *Solamente departamentos.*

```

Homicidios > mongosh: BDD +
>_MONGOSH
DB_> db["Homicidios"].find({ DEPARTAMENTO: { $in: ["ANTIOQUIA", "VALLE DEL CAUCA", "BOGOTÁ D.C."] } })

```

c. *Cantidad de muertes entre 2 y 5.*

```

Homicidios > mongosh: BDD +
>_MONGOSH
test> db.Homicidios.find({CANTIDAD: {"gte": 2, "lte": 5}})

```

d. *Muertes mayores a 10.*

```

Homicidios > mongosh: BDD +
>_MONGOSH
DB_> db["Homicidios"].find({{ CANTIDAD: { "$gt": 10 }}})

```

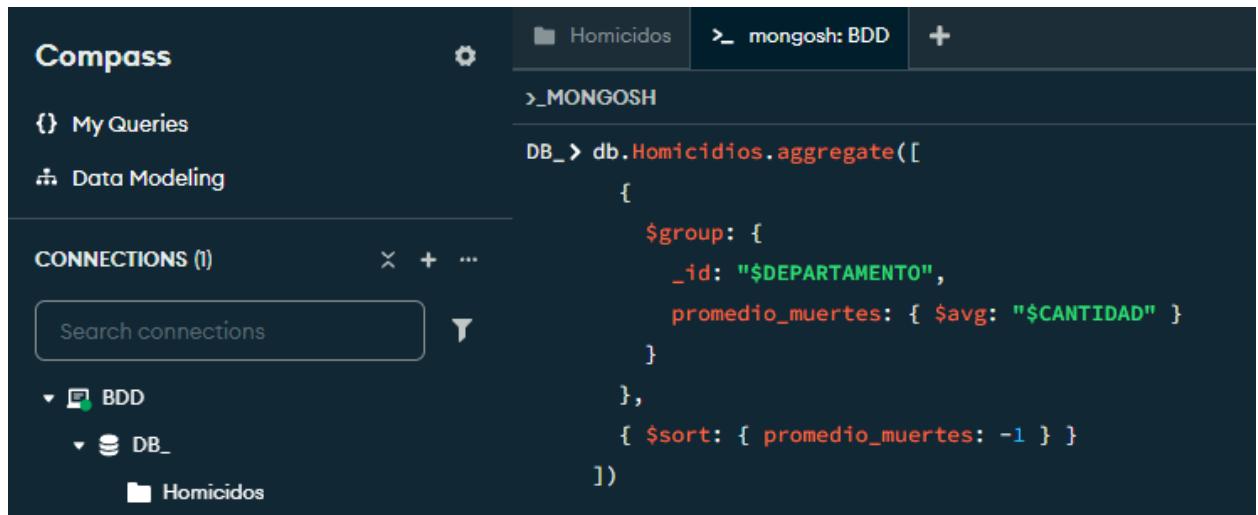
4. Consultas de agregación para calcular estadísticas.

- a. Sumar la cantidad de casos según el departamento con “\$sum”.



```
>_ mongosh: BDD | Homicidios | +
>_MONGOSH
> db.Homicidios.aggregate([{$group: {_id: "$DEPARTAMENTO", total: {$sum: 1}} }])
< [
    {
        "_id": "MAGDALENA",
        "total": 8149
    },
    {
        "_id": "BOGOTA D.C.",
        "total": 29141
    }
]
```

- b. Promedio de muertes por departamentos, utilizamos \$group, y \$avg.



```
>_ mongosh: BDD | Homicidios | +
>_MONGOSH
DB_> db.Homicidios.aggregate([
{
    $group: {
        _id: "$DEPARTAMENTO",
        promedio_muertes: { $avg: "$CANTIDAD" }
    },
    { $sort: { promedio_muertes: -1 } }
])
```

5. GitHub:

[Link del Repositorio](#)

CONCLUSION

MongoDB fue útil para trabajar con datasets como el de homicidios gracias a su modelo NoSQL basado en documentos, el cual ofrece una estructura flexible y fácil de adaptar. Al almacenar cada registro como un documento tipo JSON, se facilita la revisión, modificación y consulta sin depender de esquemas rígidos. Sus operadores permiten filtrar datos por rangos de fechas, cantidades o ubicaciones de manera intuitiva, mientras que el framework de agregación brinda herramientas para calcular promedios, totales y ordenar resultados de forma directa. En conjunto, MongoDB demuestra cómo las bases de datos NoSQL aportan flexibilidad, naturalidad en las consultas y eficiencia al explorar información.

BIBLIOGRAFÍA

MongoDB. What is NoSQL? NoSQL Databases Explained. Recuperado de [What Is NoSQL? NoSQL Databases Explained | MongoDB](#)

ScyllaDB. NoSQL Storage Types. Recuperado de [What is NoSQL Storage Types? Definition & FAQs | ScyllaDB](#)

Saurav Mitra. (2020, 29 de septiembre). What is NoSQL. Recuperado de [What is NoSQL](#)

AWS. ¿Qué es NoSQL? Recuperado de [¿Qué es una base de datos NoSQL? - Explicación de las bases de datos no relacionales - AWS](#)