### **Practico 6**

### **Parte 1**

### Ejercicio 1: Sistema de reservas / compra de tickets en una aerolínea

Para el sistema de reservas y compra de tickets de una aerolínea, la elección de una base de datos **relacional (SQL)** es más adecuada debido a varios factores esenciales que aseguran la integridad, consistencia y disponibilidad de la información crítica en este contexto.

El sistema de reservas maneja información estructurada, con relaciones complejas entre diferentes entidades como pasajeros, vuelos, asientos, etc. Estas relaciones requieren una estructura organizada y claramente definida, por ejemplo, la disponibilidad de asientos vinculada a vuelos y pasajeros específicos. Las bases de datos relacionales son la indicada para manejar este tipo de datos estructurados, ya que permiten organizar y relacionar fácilmente los datos.

La consistencia de los datos es un requisito esencial en un sistema de reservas, ya que debe garantizarse que los datos siempre estén actualizados y reflejen el estado real del sistema. Por ejemplo, si un asiento se reserva, debe quedar marcado inmediatamente como no disponible para evitar sobreventas. Las bases de datos relacionales proporcionan consistencia fuerte a través de sus propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que asegura que todas las operaciones se reflejen en tiempo real y estén sincronizadas mediante las transacciones. Esto permite que cada reserva se registre de manera confiable en el sistema, eliminando acciones no deseadas.

En un sistema de reservas, la atomicidad es fundamental. Cada reserva es una transacción que debe completarse en su totalidad o no ejecutarse en absoluto. Por ejemplo, si se realiza una reserva de múltiples asientos, todos deben quedar asignados correctamente o, en caso de error, ninguno debería registrarse. Las bases de datos relacionales cumplen con esta propiedad de atomicidad, garantizando que las transacciones se realicen de manera completa y segura.

El sistema de reservas es utilizado por múltiples usuarios en simultáneo, lo que significa que varias personas podrían intentar reservar el mismo asiento al mismo tiempo. Para evitar conflictos y garantizar que cada transacción se realice de forma aislada, es importante contar con un sistema que soporte un aislamiento fuerte de las transacciones. Las bases de datos relacionales manejan el control de concurrencia y el aislamiento mediante niveles transaccionales, protegiendo cada operación y asegurando que no haya interferencias entre transacciones simultáneas.

En un sistema de reservas de aerolíneas, la durabilidad de los datos es esencial. Una vez confirmada una reserva, la información debe persistir en el sistema, incluso si ocurre una falla del servidor o un corte de energía. Las bases de datos relacionales proporcionan durabilidad a través de mecanismos de respaldo y recuperación, lo que asegura que los datos de cada transacción completada se almacenen de manera permanente y estén disponibles siempre que se necesiten.

Aunque las bases de datos NoSQL se destacan por su capacidad de escalabilidad horizontal, las bases de datos relacionales también pueden escalarse mediante técnicas de partición de datos y replicación, especialmente en sistemas donde el rendimiento debe mantenerse estable. Para el sistema de reservas, donde el enfoque principal es la consistencia y el control transaccional, una base de datos relacional es capaz de cumplir con estos requerimientos al tiempo que mantiene un rendimiento adecuado durante picos de demanda.

### Ejercicio 2: Sistema de reporte de transacciones bancarias categorizadas por monto

Para un sistema que reporte las transacciones realizadas en un banco durante el último día y las categorice por franjas de monto, la elección de una base de datos relacional (SQL) es más adecuada debido a las necesidades de precisión, consistencia y facilidad para realizar consultas agregadas, que son características fundamentales en un entorno bancario.

El sistema de reporte maneja información financiera estructurada que incluye detalles sobre transacciones específicas como el monto, la fecha y el tipo de operación. Estas transacciones deben agruparse por monto, lo que requiere consultas estructuradas y filtradas en tiempo real para reportar de forma precisa las transacciones dentro de cada categoría establecida. Las bases de datos relacionales son las mejores para este tipo de datos estructurados y permiten categorizar y organizar fácilmente la información según criterios definidos.

La consistencia de los datos es crucial en un sistema bancario, ya que el reporte debe reflejar de forma precisa el estado de las transacciones del último día. Las propiedades ACID de las bases de datos relacionales aseguran que los datos sean confiables, lo cual es fundamental en un contexto donde los registros financieros deben ser completamente precisos y verificados. Esta consistencia fuerte garantiza que cualquier operación de transacción refleje el monto correcto y esté categorizada de forma precisa.

La atomicidad es otra propiedad clave para este sistema. Cada transacción bancaria debe registrarse como una operación completa e indivisible. Si, por ejemplo, una transacción no puede completarse en su totalidad, no debe registrarse en absoluto para evitar inexactitudes en el reporte. Las bases de datos relacionales garantizan esta atomicidad, lo que asegura que cada transacción financiera sea procesada correctamente y refleje un estado consistente en el sistema de reportes.

Dado que los reportes se generan diariamente y pueden ser accedidos por múltiples usuarios, es importante que las operaciones de consulta sean aisladas para evitar interferencias o lecturas inconsistentes en medio de una actualización de datos. Las bases de datos relacionales proporcionan niveles de aislamiento que permiten acceder a datos exactos y categorizados sin interferencias, incluso cuando se están registrando nuevas transacciones.

La durabilidad es esencial en un sistema bancario, ya que los registros de transacciones deben permanecer en el sistema incluso en caso de fallos o interrupciones. Las bases de datos relacionales ofrecen durabilidad a través de registros permanentes y de respaldo, garantizando que la información de las transacciones diarias esté siempre disponible y pueda recuperarse en cualquier circunstancia.

El tipo de consultas en este sistema se centra en operaciones de agrupación, conteo y filtrado de transacciones, lo cual requiere SQL para poder realizar agregaciones avanzadas, ordenar monto y generar reportes detallados. Estas consultas estructuradas y agregadas son perfectamente compatibles con una base de datos relacional, que ofrece optimización para operaciones como GROUP BY y ORDER BY en grandes volúmenes de datos.

### Ejercicio 3: Sistema de recomendación basado en compras anteriores

Para un sistema de recomendación que sugiere productos en base a compras anteriores de los clientes, la elección de una base de datos NoSQL es más adecuada debido a la necesidad de flexibilidad en el esquema, escalabilidad y rapidez en la recuperación de datos que permita un análisis eficiente y en tiempo real.

Este sistema maneja información semiestructurada, como el historial de compras, características de los productos y perfiles de clientes. La naturaleza de estos datos es altamente variable y se enriquece constantemente con nuevas transacciones, lo cual requiere una estructura flexible para incorporar nuevos tipos de datos sin problemas. Las bases de datos NoSQL, como MongoDB, permiten esta flexibilidad, ya que pueden almacenar información de productos, categorías y comportamientos de compra en documentos JSON, sin necesidad de un esquema rígido.

En términos de rendimiento, el sistema de recomendaciones debe poder acceder y analizar rápidamente el historial de compras para generar sugerencias en tiempo real. Dado que se trata de un sistema de alto rendimiento y consulta constante, una base de datos NoSQL optimiza el acceso a grandes volúmenes de datos mediante índices y consultas rápidas. Esto permite recomendar productos de manera inmediata, mejorando la experiencia del usuario.

La escalabilidad es un factor clave en este tipo de sistema, especialmente en plataformas de comercio electrónico con millones de usuarios y productos. Las bases de datos NoSQL están diseñadas para escalar horizontalmente, distribuyendo datos entre múltiples servidores de manera eficiente y permitiendo manejar el crecimiento en volumen de datos y usuarios sin afectar el rendimiento del sistema de recomendaciones.

Aunque la atomicidad de las operaciones no es una prioridad en este sistema, es fundamental que el sistema de recomendación funcione de manera efectiva incluso si algunas transacciones de compra aún no se han sincronizado completamente. Las bases de datos NoSQL son ideales aquí, ya que pueden manejar consistencia eventual, lo cual es suficiente para un sistema de recomendaciones que no depende de transacciones estrictas, sino de patrones y tendencias generales de compra.

En cuanto a la consistencia, este sistema puede tolerar una consistencia eventual, dado que las recomendaciones no requieren ser absolutamente precisas en todo momento. Las bases de datos NoSQL permiten que los datos se sincronicen en segundo plano, de manera que los usuarios pueden ver sugerencias relevantes basadas en sus compras recientes sin necesidad de una actualización instantánea en todas las partes del sistema.

El tipo de consultas que realiza este sistema es analítico y basado en patrones. Las consultas buscan patrones en los datos de compra para identificar productos relacionados y generar recomendaciones personalizadas. Este tipo de consultas y análisis de datos a gran escala son compatibles con bases de datos NoSQL, que permiten realizar búsquedas rápidas y flexibles en grandes volúmenes de datos semiestructurados, y que pueden optimizarse mediante técnicas de procesamiento en paralelo.

### Ejercicio 4: Sistema de análisis de ventas mensuales

Para un sistema de análisis de ventas mensuales, la elección entre una base de datos relacional (SQL) o una NoSQL dependerá en gran medida de la escala de la empresa, los requerimientos de procesamiento de datos y la infraestructura tecnológica. Se plantean tres casos para analizar:

1. Empresa multinacional con depósitos distribuidos y grandes niveles de venta

Para una empresa multinacional, que maneja un alto volumen de datos distribuidos en múltiples países, una base de datos NoSQL es una opción más adecuada debido a su flexibilidad, escalabilidad y rapidez en el análisis de grandes volúmenes de datos distribuidos geográficamente.

Este sistema maneja datos semiestructurados y estructurados, como registros de ventas, información de inventarios y datos de localización de depósitos, los cuales pueden variar según las regulaciones y requerimientos de cada país. Las bases de datos NoSQL, como MongoDB o Cassandra, permiten almacenar estos datos de forma flexible sin un esquema fijo, facilitando la integración de información de diferentes regiones.

La escalabilidad es crucial en este contexto, dado el gran volumen de datos que genera una empresa multinacional. Las bases de datos NoSQL están diseñadas para escalar horizontalmente, distribuyendo los datos entre múltiples servidores. Esto permite a la empresa gestionar sus datos de ventas en varias ubicaciones y ampliar la infraestructura de almacenamiento según sea necesario, sin comprometer el rendimiento del sistema.

En cuanto a la atomicidad y consistencia, una consistencia eventual es generalmente suficiente para análisis mensuales. Las bases de datos NoSQL permiten replicación de datos entre regiones, lo que garantiza que los datos estarán disponibles en tiempo real, aunque la actualización pueda no ser instantánea en todas las ubicaciones. Esto es adecuado para análisis que no requieren transacciones estrictamente sincronizadas.

El tipo de consultas en este sistema es principalmente analítico, enfocándose en tendencias de ventas, comparaciones por región y rendimiento mensual. Las bases de datos NoSQL, especialmente aquellas optimizadas para análisis de datos en tiempo real, permiten realizar consultas complejas y procesar grandes volúmenes de datos rápidamente.

Además, cuando una multinacional realiza un reporte mensual sobre sus ventas, no le cambia si hay una leve variante en los resultados, cosa contraria pasa en los siguientes dos casos:

Empresa nacional con menor escala de operaciones (por ejemplo, Conaprole o Salus)

En el caso de una empresa nacional que maneja datos centralizados dentro de un solo país, una base de datos relacional (SQL) es una opción viable, especialmente si la empresa prefiere un control transaccional más estricto y necesita consistencia en sus informes mensuales.

La información de ventas en una empresa nacional puede ser de gran volumen, pero generalmente sigue una estructura uniforme, lo cual se adapta bien a un modelo de base de datos relacional. Las bases de datos SQL permiten estructurar y relacionar datos de ventas, inventario y clientes de forma organizada, optimizando el análisis y el reporte mensual de forma precisa.

Para este tipo de sistema, el rendimiento es esencial, pero las bases de datos relacionales pueden manejar esta carga adecuadamente mediante la optimización de índices y consultas, especialmente si los datos están centralizados en una única infraestructura.

La consistencia y la atomicidad son críticas, dado que la empresa necesita que sus datos financieros y de ventas reflejen la realidad de forma precisa y en tiempo real, lo que no pasa en las multinacionales. Las bases de datos relacionales proporcionan consistencia fuerte mediante transacciones ACID, lo cual garantiza que todos los datos de ventas sean fiables y estén listos para análisis detallados y seguros.

Las consultas en este sistema son estructuradas y analíticas, enfocándose en tendencias de ventas y reportes mensuales. Las bases de datos SQL están diseñadas para consultas estructuradas, lo que facilita el análisis preciso de datos históricos y actuales.

Por último, un almacén local de pequeña escala, si el dueño lo cree necesario debería utilizar una base de datos relacional (SQL). Pero no se ve necesario en una almacén de barrio.

Ejercicio 5: Sistema que intenta alertarse ante una compra por valor 5000 USD por una persona que no suele utilizar su tarjeta.

Para un sistema que detecta y alerta ante compras inusuales por valor de 5000 USD realizadas por usuarios que no suelen utilizar su tarjeta, la elección de una base de datos NoSQL es más adecuada debido a la necesidad de escalabilidad, rendimiento en tiempo real y flexibilidad en el almacenamiento de datos de usuario y transacciones, así como en el análisis de patrones de comportamiento.

Este sistema maneja información semiestructurada que incluye datos de transacciones, perfiles de usuario y patrones de uso de tarjetas. Estos datos pueden variar, ya que se agregan constantemente nuevas transacciones y el comportamiento de los usuarios cambia con el tiempo. Una base de datos NoSQL, como MongoDB o Cassandra, permite una estructura flexible para almacenar tanto los datos de la compra como los patrones de uso de cada usuario, adaptándose a la variabilidad de los datos y permitiendo realizar cambios en el esquema sin afectar el rendimiento.

En términos de rendimiento, el sistema debe ser capaz de analizar y procesar grandes volúmenes de datos de transacciones en tiempo real para detectar patrones inusuales. Las bases de datos NoSQL son optimizadas para consultas rápidas y en tiempo real, lo que permite que el sistema genere alertas instantáneas ante una transacción sospechosa, mejorando la capacidad de respuesta para prevenir posibles fraudes.

La escalabilidad es otro factor clave, ya que el sistema debe procesar datos de millones de usuarios y sus respectivas transacciones a medida que estas ocurren. Las bases de datos NoSQL están diseñadas para escalar horizontalmente, lo que permite distribuir el procesamiento de datos en varios servidores y manejar el aumento de transacciones sin degradar el rendimiento, lo cual es fundamental para un sistema de detección de fraudes en una institución financiera.

En cuanto a la atomicidad y consistencia, este sistema puede tolerar una consistencia eventual, dado que las alertas pueden generarse aunque algunas transacciones recientes aún no se hayan sincronizado completamente en todo el sistema. Las bases de datos NoSQL permiten manejar grandes volúmenes de datos en tiempo real sin necesidad de transacciones estrictas, priorizando la velocidad de procesamiento sobre la consistencia inmediata en todas las réplicas.

El tipo de consultas en este sistema es analítico y centrado en el comportamiento. Las consultas buscan patrones inusuales en los datos de transacciones y verifican la frecuencia de uso de cada tarjeta. Estas consultas requieren flexibilidad para analizar datos en tiempo real y adaptar las reglas de detección. Las bases de datos NoSQL ofrecen una solución eficiente para este tipo de consultas basadas en patrones, ya que están optimizadas para realizar búsquedas y análisis complejos sin depender de un esquema rígido.

### Ejercicio 6: Sistema de compras online con la necesidad de registrar usuarios y detección de posibles cuentas fraudulentas

Para un sistema de compras en línea que requiere registrar usuarios y, simultáneamente, detectar si un nuevo cliente es potencialmente fraudulento (como cuando se crean varias cuentas desde la misma IP en un corto período de tiempo), la elección de una base de datos NoSQL es ideal. La flexibilidad, escalabilidad y capacidad para realizar análisis de patrones en tiempo real de este tipo de bases de datos son beneficios clave para este sistema.

Este sistema maneja datos semiestructurados y estructurados, que incluyen información de usuarios, actividad de IPs y patrones de registro. La naturaleza de estos datos puede variar con el tiempo, ya que se agregan continuamente nuevas cuentas y se almacenan eventos temporales de actividad de IP. Las bases de datos NoSQL, como MongoDB, permiten una estructura de datos flexible, lo cual es esencial para almacenar tanto la información de registro de usuario como los registros de actividad en un esquema que se puede adaptar según las necesidades.

En cuanto al rendimiento, el sistema debe poder verificar rápidamente los datos de registro y las direcciones IP asociadas para detectar patrones inusuales en tiempo real. Las bases de datos NoSQL están optimizadas para consultas rápidas y permiten ejecutar estas verificaciones y correlaciones casi instantáneamente, lo cual es crucial para una detección eficaz de cuentas fraudulentas al momento de registrar nuevos usuarios.

La escalabilidad es un factor fundamental, especialmente en plataformas de comercio electrónico que manejan grandes volúmenes de usuarios nuevos y transacciones diarias. Las bases de datos NoSQL están diseñadas para escalar horizontalmente, permitiendo al sistema distribuir la carga entre múltiples servidores y manejar el crecimiento de usuarios y actividad sin afectar el rendimiento, garantizando la eficacia del sistema de monitoreo y detección de fraudes.

Aunque la atomicidad no es un requisito estricto para este sistema, es importante que el análisis y la detección de cuentas fraudulentas se realicen de manera confiable, aunque algunas transacciones recientes puedan no estar completamente sincronizadas en todo el sistema. Las bases de datos NoSQL permiten una consistencia eventual, lo cual es suficiente para identificar patrones de posible fraude basados en la frecuencia de registros de IPs.

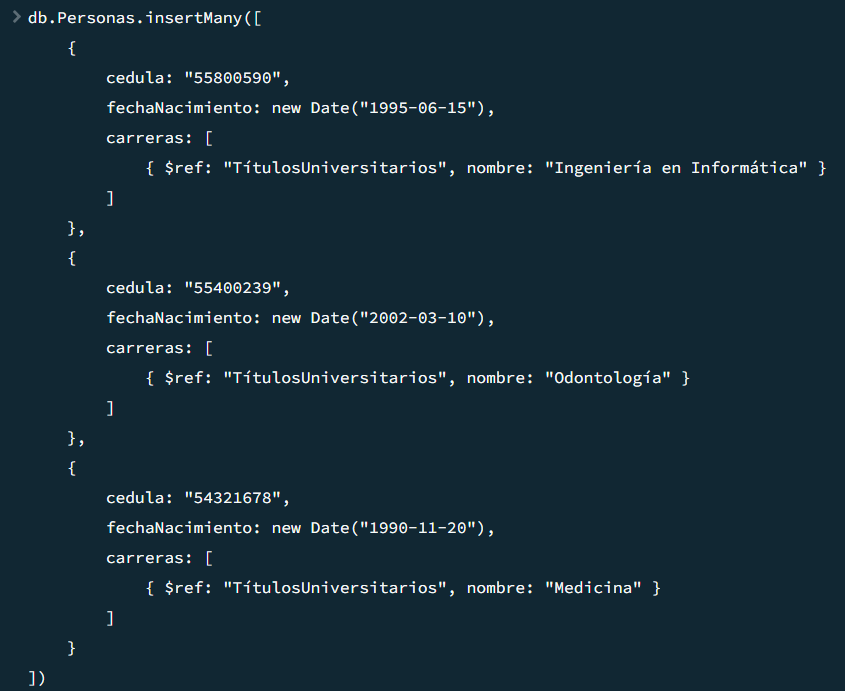
El tipo de consultas en este sistema se centra en la detección de patrones y en el análisis de comportamiento. Las consultas deben ser capaces de identificar y correlacionar múltiples registros de IP en tiempo real para activar alertas en caso de actividad sospechosa. Las bases de datos NoSQL permiten realizar este tipo de consultas de forma rápida y eficiente, optimizando la detección de patrones irregulares sin la necesidad de operaciones complejas o esquemas rígidos.

**Parte 2 Igual a Ignacio Aguerrondo, lo hicimos juntos.**

use practico\_6

--inserts:





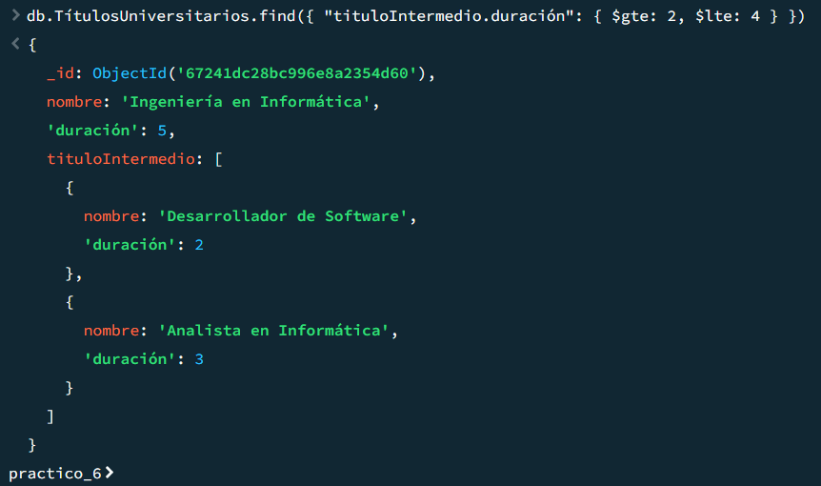
1. Buscar aquellos títulos universitarios cuya duración sea de al menos 2 años, pero que no supere los 4.

db.TítulosUniversitarios.find({ duración: { $gte: 2, $lte: 4 } })

Esto no devuelve ningún resultado ya que los títulos padre no tienen ninguno esta duración, por eso hice esta segunda consulta para chequearlo en los títulos intermedios:

db.TítulosUniversitarios.find({ "tituloIntermedio.duración": { $gte: 2, $lte: 4 } })

devolviendo esto:



también encontré que podía haber hecho las dos consultas juntas de esta forma:

db.TítulosUniversitarios.find({

$or: [

{ duración: { $gte: 2, $lte: 4 } },

{ "tituloIntermedio.duración": { $gte: 2, $lte: 4 } }

]

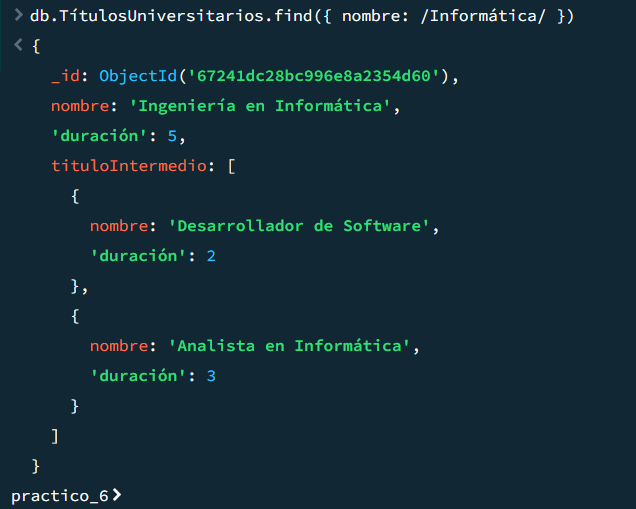
})

1. Buscar carrera cuyo nombre contenga la palabra "Informática".

Consulta:

db.TítulosUniversitarios.find({ nombre: /Informática/ })

Resultado:

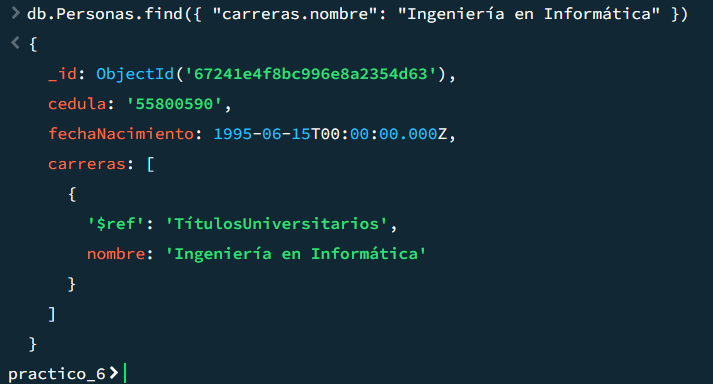


1. Busque todas las personas que estudian Ingeniería en Informática

Consulta:

db.Personas.find({ "carreras.nombre": "Ingeniería en Informática" })

Resultado:

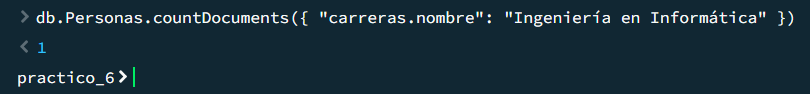


1. Cuente todas las personas que estudian Ingeniería en Informática.

Consulta:

db.Personas.countDocuments({ "carreras.nombre": "Ingeniería en Informática" })

Resultado:

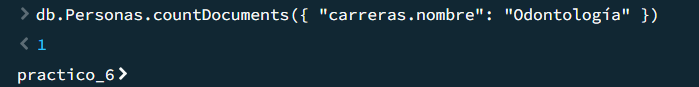


1. Cuente todas las personas que estudian Odontología.

Consulta:

db.Personas.countDocuments({ "carreras.nombre": "Odontología" })

Resultado:



1. Busque todas las personas mayores a 20 años.

Consulta:

const veinteañosatrás = new Date();

veinteañosatrás.setFullYear(veinteañosatrás.getFullYear() - 20);

db.Personas.find({ fechaNacimiento: { $lte: veinteañosatrás} })

Resultado:



1. Liste el nombre de todas las personas de la base de datos junto al título de la carrera que estudia y la duración del mismo.

Consulta:

db.Personas.aggregate([

{

$lookup: {

from: "TítulosUniversitarios",

localField: "carreras.nombre",

foreignField: "nombre",

as: "tituloDetalles"

}

},

{

$project: {

cedula: 1,

"tituloDetalles.nombre": 1,

"tituloDetalles.duración": 1

}

}

])

Resultado:



### Para Investigar:

* Se quiere obtener un reporte en que se diga, año por año, la cantidad de títulos, título por título.

Para poder hacer este reporte debería tener algún documento con año de graduación, teniendo que modificar cada documento añadiéndole un año de graduación en el caso de que fueran graduados, quedando un documento persona de ejemplo así:

{

cedula: "5908410",

fechaNacimiento: new Date("1995-06-15"),

carreras: [

{

$ref: "TítulosUniversitarios",

nombre: "Ingeniería en Informática",

añoGraduación: 2022

}

]

}

En mi caso añadí para la prueba estos dos documentos:

db.Personas.insertMany([

{

cedula: "12345678",

fechaNacimiento: new Date("1995-06-15"),

carreras: [

{

$ref: "TítulosUniversitarios",

nombre: "Ingeniería en Informática",

añoGraduación: 2022

}

]

},

{

cedula: "87654321",

fechaNacimiento: new Date("2000-09-12"),

carreras: [

{

$ref: "TítulosUniversitarios",

nombre: "Odontología",

añoGraduación: 2023

}

]

}

])

Luego de ya tener documentos con gente graduada podríamos realizar el reporte mediante esta consulta:

db.Personas.aggregate([

{ $unwind: "$carreras" },

{ $match: { "carreras.añoGraduación": { $exists: true } } },

{

$group: {

\_id: { año: "$carreras.añoGraduación", titulo: "$carreras.nombre" },

count: { $sum: 1 }

}

},

{ $sort: { "\_id.año": 1, "\_id.titulo": 1 } }

])

Y dando como resultado algo así:

