IRisk Lab Data Discovery and Consolidation - Task 2
Week 2 Report
Sep. 5
Rohit Valmeekam

What was accomplished

Dove Deeper into the benefits and downsides of using DQL as opposed to MongoDB

SQL (Structured Query Language)

SQL databases are known for their strong data integrity, consistency, and adherence to ACID (Atomicity, Consistency, Isolation, Durability) properties. This makes them highly reliable for storing sensitive data. They offer well-established security features, including role-based access control, encryption, and authentication mechanisms. These features are crucial for maintaining high levels of data security. SQL databases excel at handling complex data relationships through the use of structured tables with defined schemas. This can be advantageous when dealing with intricate insurance data structures. As stated last week, it is critical to have different relationships between "Insurance Category" and "Dataset Table" database tables to thoroughly display all the datasets in an organized manner. This is due to the nature of the datasets that were acquired by Team 1 and them being grouped by different insurance categories.

SQL provides a powerful and standardized query language that is familiar to most developers, making it easier to work with and maintain.

However, SQL databases require a fixed database schema, which can be restrictive when dealing with evolving or unstructured data. Changes to the schema can be challenging. Assuming our data pipeline is created correctly, this should not be an issue.

For certain high-write, high-throughput scenarios, SQL databases might not perform as well as NoSQL databases due to the overhead of enforcing ACID properties. While scalability options exist, scaling SQL databases can be more complex and costly than scaling NoSQL databases, particularly for extremely large data volumes.

MongoDB (NoSQL)

MongoDB's schema-less design allows you to store unstructured or semi-structured data easily. This flexibility can accommodate changing insurance data requirements. Might be useful if we decide to seriously alter the way the insurance data is stored.

MongoDB is also optimized for write-heavy workloads, making it suitable for applications with high data ingestion rates.MongoDB excels at horizontal scalability, distributing data across multiple nodes or clusters. This makes it well-suited for handling extremely large volumes of data.

However, MongoDB sacrifices some consistency in favor of performance and scalability. This may not be ideal for applications where absolute data consistency is crucial.

Complex queries and joins are less intuitive in MongoDB compared to SQL, which can make data retrieval and analysis more challenging.

Developers accustomed to SQL may face a learning curve when transitioning to MongoDB, particularly when dealing with complex data structures. Particularly for a lot of us here on Task 2, MongoDB presents a larger learning curve than SQL does.

To-Do List

1. Create the data pipeline in order to normalize the data between Task 1 data and our database system
2. Create the infrastructure for our database system