The Evolution of NoSQL

Web 2.0 refers to the second generation of the World Wide Web, characterized by the shift from static web pages to dynamic and interactive content( youtube, blogs, facebook,twitter etc).

The SQL **scalability issue** was recognized by Web 2.0 companies with huge, growing data and infrastructure needs, such as Google, Amazon, and Facebook.

They came up with their own solutions to the problem – technologies like [BigTable](https://cloud.google.com/bigtable/), [DynamoDB](https://aws.amazon.com/dynamodb/), and [Cassandra](http://cassandra.apache.org/).

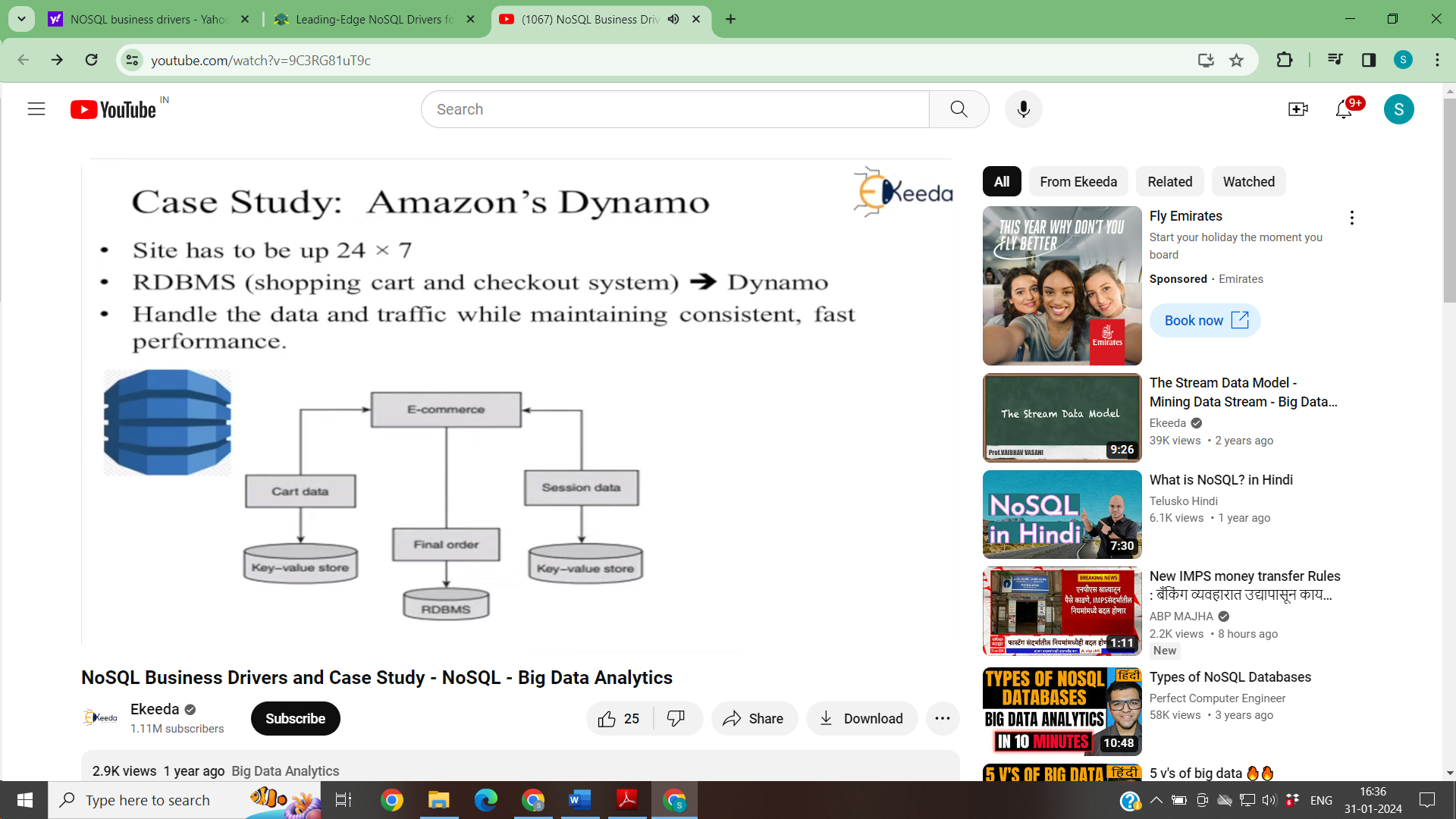


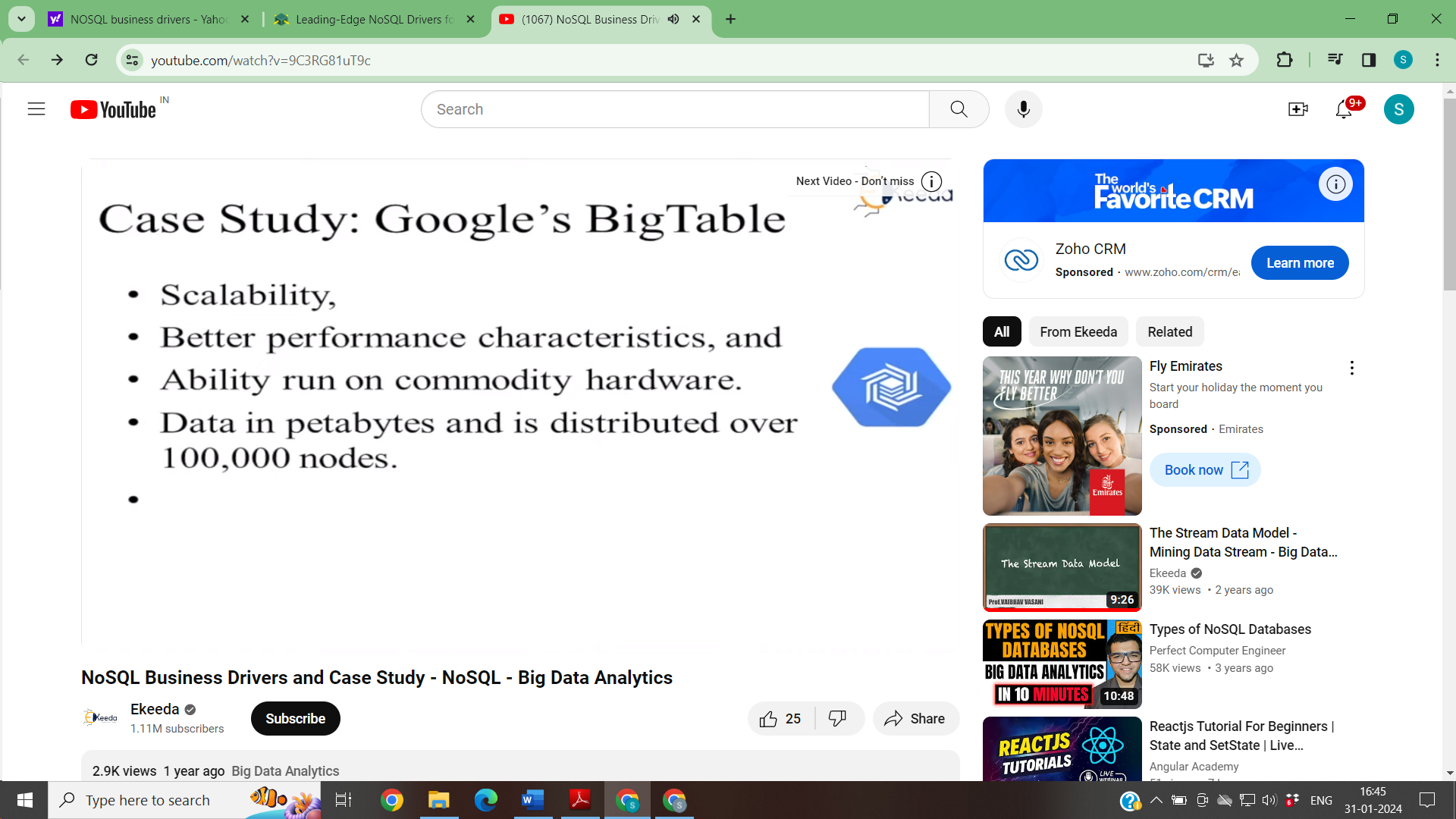
This growing interest resulted in a number of NoSQL Database Management Systems (DBMS’s), with a **focus on performance, reliability, and consistency.**

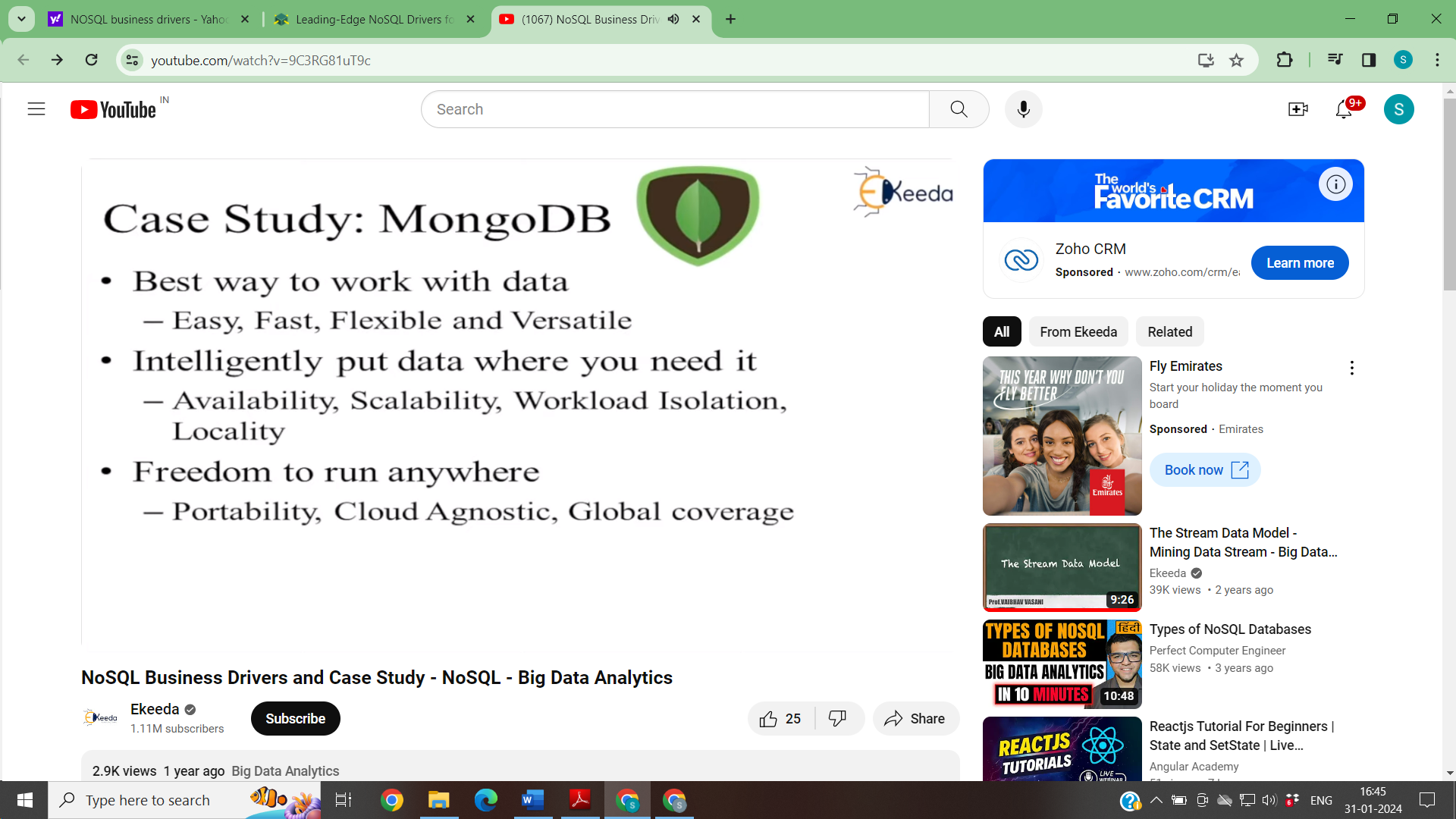
A number of **existing indexing structures were reused and improved** upon with the purpose of enhancing search and read performance.

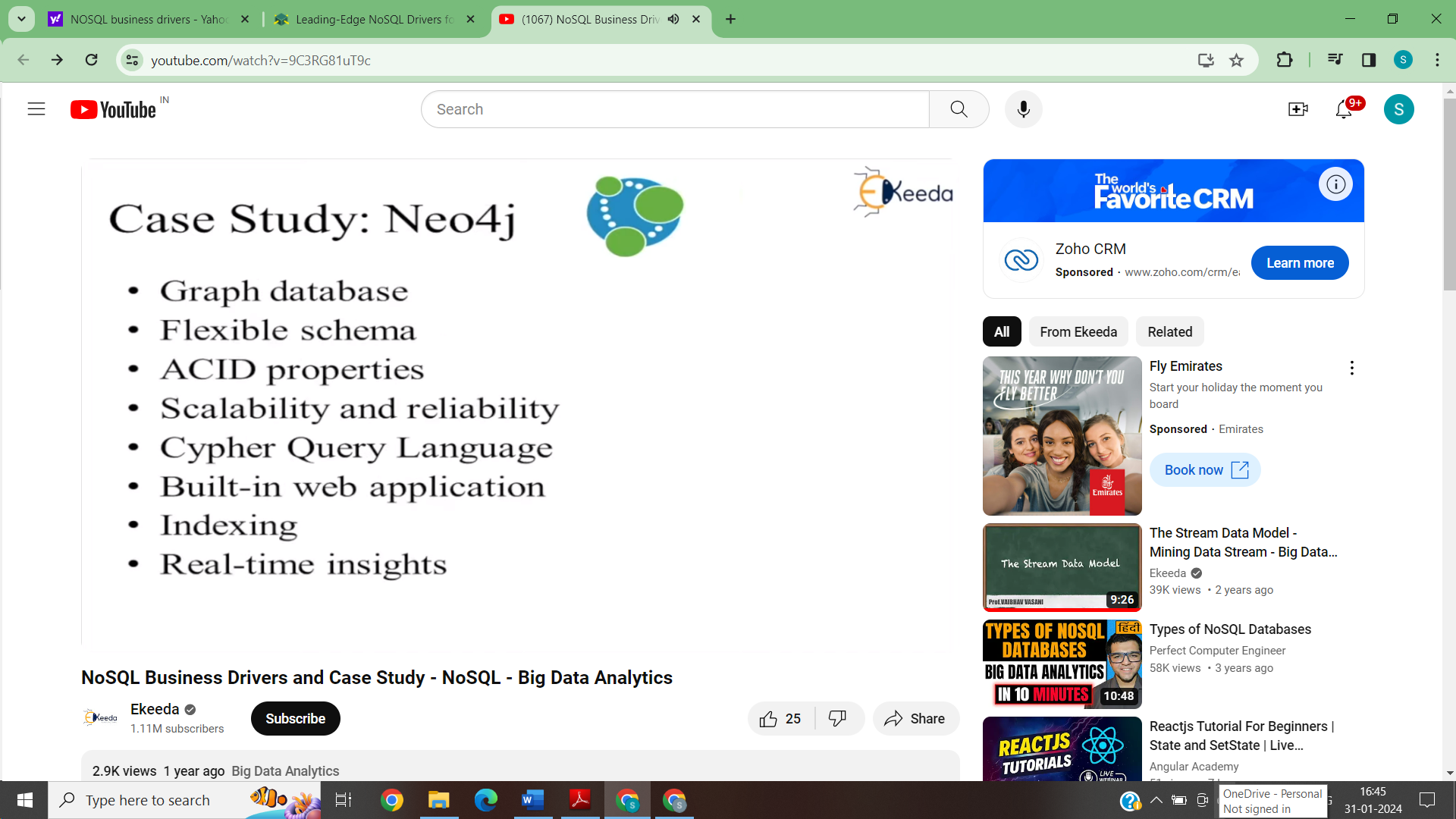
First, there were proprietary (closed source) types of NoSQL databases developed by big companies to meet their specific needs, such as **Google’s BigTable, which is believed to be the first NoSQL system, and Amazon’s DynamoDB.**

The success of these proprietary systems, initiated development of a number of similar open-source and proprietary database systems, the most popular ones being Hypertable, Cassandra, MongoDB, DynamoDB, HBase, and Redis.









NoSQL is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructured and semi-structured data.

Unlike traditional relational databases that use tables with pre-defined schemas to store data, NoSQL databases use flexible data models that can adapt to changes in data structures and are capable of scaling horizontally to handle growing amounts of data.

The term NoSQL originally referred to “non-SQL” or “non-relational” databases, but the term has since evolved to mean “not only SQL,” as NoSQL databases have expanded to include a wide range of different database architectures and data models.

It does not follow the rules of Relational Database Management Systems (RDBMS), and hence do not use traditional SQL statements to query your data.

Some famous examples are MongoDB, Neo4J, HyperGraphDB, etc.

**NoSQL databases are generally classified into four main categories:**

1. **Document database:**
2. In a document database, the data is stored in documents.

Each document is typically a nested structure of keys and values. The values can be atomic data types, or complex elements such as lists, arrays, nested objects, or child collections (for example, a collection in the document database is analogous to a table in a relational database(ordb), except there is no single schema enforced upon all documents).

1. Documents are retrieved by unique keys. It may also be possible to retrieve only parts of a document--for example, the cost of an item--to run queries such as aggregation, querying using examples based on a text string, or even full-text search. Most document databases also allow you to define secondary indexes.
2. You can transfer the application code object model directly into a document using several different formats. The most commonly used are JavaScript Object Notation (JSON), Binary JavaScript Object Notation (BSON), and Extensible Markup Language (XML).


          An example of a document data model.
        

*An example of a document data model*

AWS offers a specialized document database service called [Amazon DocumentDB (with MongoDB compatibility)](https://aws.amazon.com/documentdb/).

NoSQL databases are often used in applications where there is a high volume of data that needs to be processed and analyzed in real-time, such as social media analytics, e-commerce, and gaming.

They can also be used for other applications, such as content management systems, document management, and customer relationship management.

However, NoSQL databases **may not be suitable for all applications, as they may not provide the same level of data consistency and transactional guarantees as traditional relational database**s.

It is important to carefully evaluate the specific needs of an application when choosing a database management system.

In a relational database, we'd likely create two tables: one for Users and one for Hobbies.

Users

ID first\_name last\_name cell city

1 Leslie Yepp 8125552344 Pawnee

Hobbies

ID user\_id hobby

10 1 scrapbooking

11 1 eating waffles

12 1 working

In order to retrieve all of the information about a user and their hobbies, information from the Users table and Hobbies table will need to be joined together.

The data model we design for a NoSQL database will depend on the type of NoSQL database we choose. Let's consider how to store the same information about a user and their hobbies in a **document database like MongoDB**.

Code Snippet for MongoDB

{

"\_id": 1,

"first\_name": "Leslie",

"last\_name": "Yepp",

"cell": "8125552344",

"city": "Pawnee",

"hobbies": ["scrapbooking", "eating waffles", "working"]

}

In order to retrieve all of the information about a user and their hobbies, a single document can be retrieved from the database. No joins are required, resulting in faster queries.

**2. Key-value:**

A [key-value data store](https://aws.amazon.com/nosql/key-value/) is a type of database that stores data as a **collection of key-value pairs**. In this type of data store, each data item is identified by a unique key, and the value associated with that key can be anything, such as a string, number, object, or even another data structure.(MongoDB)


          A diagram showing an example of data stored as key-value pairs.
        

*An example of data stored as key-value pairs.*

AWS offers Amazon DynamoDB as a key-value managed database service.

1. **Wide-column** — A wide column data store is a type of NoSQL database that stores data in columns rather than rows, making it highly scalable and flexible. In a wide column data store, data is organized into column families, which are groups of columns that share the same attributes. Each row in a wide column data store is identified by a unique row key, and the columns in that row are further divided into column names and values.

Unlike traditional relational databases, which have a fixed number of columns and data types, wide column data stores allow for a variable number of columns and support multiple data types. The most significant benefit of having column-oriented databases is that you can store large amounts of data within a single column. This feature allows you to reduce disk resources and the time it takes to retrieve information from it.


          A diagram showing an example of the kind of data you might store in a wide-column
            data store.
        

*An example of the kind of data you might store in a wide-column data store*

AWS offers Amazon Keyspaces (for Apache’s Cassandra) as a wide-column managed database service.

1. **Graph** — Graph databases are used to store and query highly connected data. Data can be modelled in the form of entities (also referred to as nodes, or vertices) and the relationships between those entities (also referred to as edges). The strength or nature of the relationships also carry significant meaning in graph databases.

Users can then traverse the graph structure by starting at a defined set of nodes or edges and travel across the graph, along defined relationship types or strengths, until they reach some defined condition.

Results can be returned in the form of literals, lists, maps, or graph traversal paths.

Graph databases provide a set of query languages that contain syntax designed for traversing a graph structure, or matching a certain structural pattern.


          An example of a social network graph.
        

*An example of a social network graph. Given the people (nodes) and their relationships (edges), you can find out who the "friends of friends" of a particular person are—for example, the friends of Howard's friends.*

AWS offers [Amazon Neptune](https://aws.amazon.com/neptune/) as a managed graph database service (Neo4j is open source)

