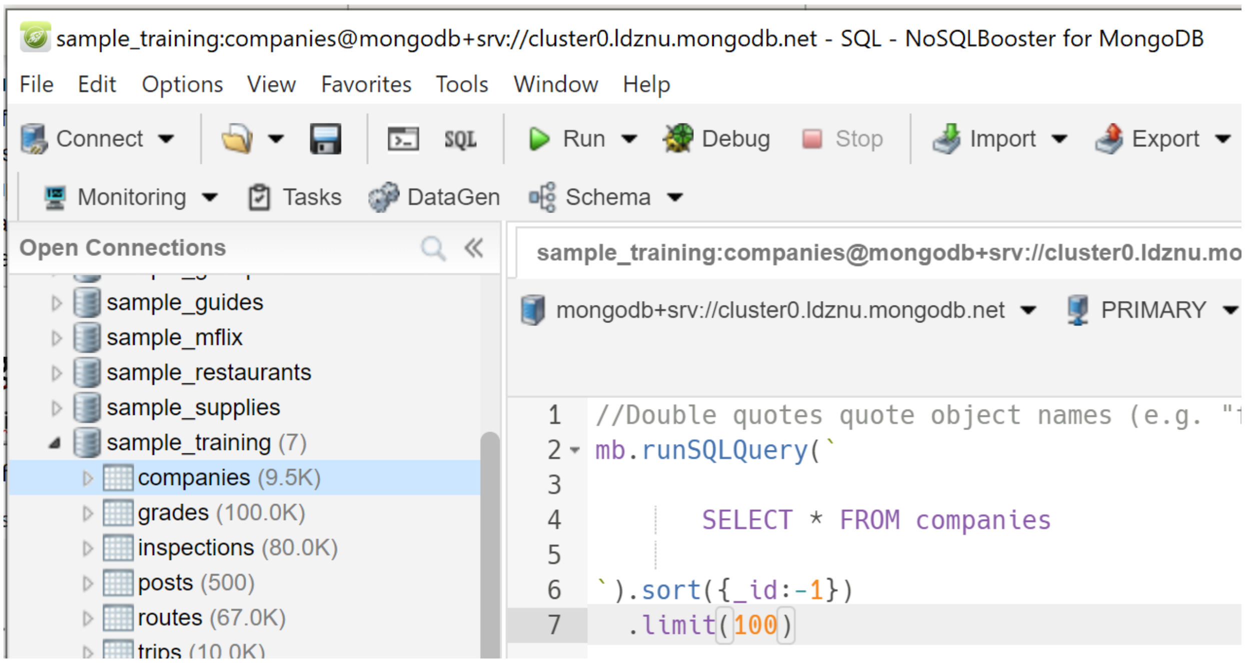
Tutorial – Week 11 SOLUTIONS

Objectives:

- Explore NoSQL database (MongoDB)
- Install and use a browser for MongoDB - NoSQLBooster
- Get understanding of JSON
- Revise theoretical concepts of NoSQL Databases

A. Download & Install
B. Connect to MongoDB
C. Explore MongoDB sample data using NoSQLbooster

3. The browser will show a default SQL query.



```
//Double quotes quote object names (e.g. "t
mb.runSQLQuery(`

      SELECT * FROM companies

`).sort({_id:-1})
  .limit(100)
```

What is the default SQL query?

SELECT * FROM companies; -- as on the screenshot above

If we select another collection, it will be SELECT * FROM <whatever collection selected>

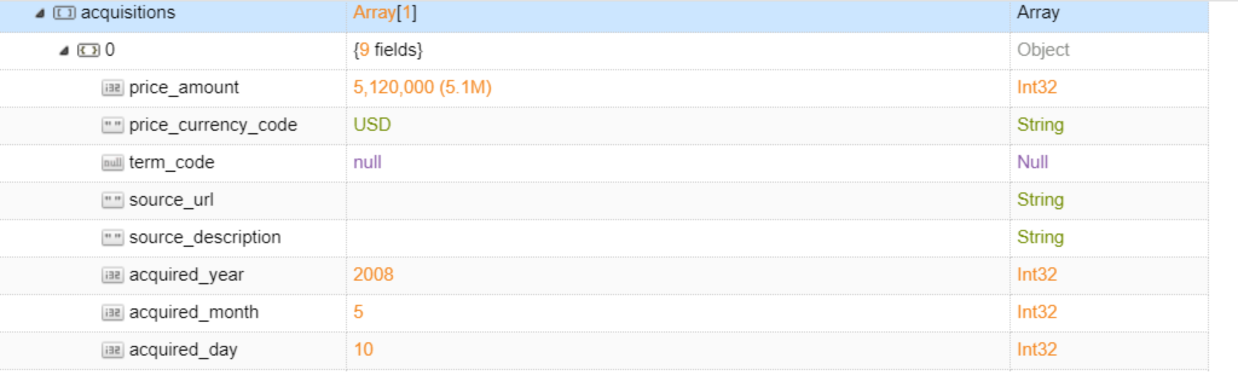
6. This is 'Tree' view look at data types, how many are there? What types are they and what is the highest level (or parent)? Why is it called 'Tree'?

Document at the top
It has branches or levels..

9. Now try 'JSON' view (far right).
Explore, note that data is in key:value pairs e.g. "name" : "EnteGreat Solutions" (and separated by commas). **Find 'Acquisitions', what data type is it?**

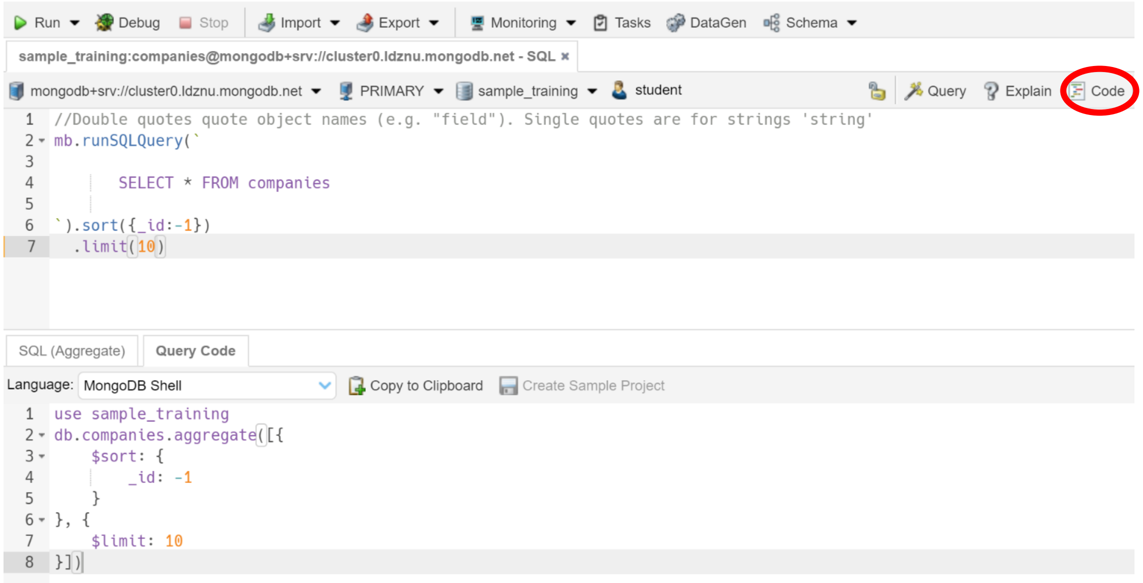
Go back to Tree view - an array

If you open it, you will see something like this screenshot



acquisitions	Array[1]	Array
0	{9 fields}	Object
price_amount	5,120,000 (5.1M)	Int32
price_currency_code	USD	String
term_code	null	Null
source_url		String
source_description		String
acquired_year	2008	Int32
acquired_month	5	Int32
acquired_day	10	Int32

10. Now try ‘Code’ (button circled at the far right) to see:



```
//Double quotes quote object names (e.g. "field"). Single quotes are for strings 'string'
mb.runSQLQuery(`

    SELECT * FROM companies

`).sort({_id:-1})
  .limit(10)
```

Language: MongoDB Shell

```
use sample_training
db.companies.aggregate([{
    $sort: {
        _id: -1
    }
}, {
    $limit: 10
}])
```

Obviously we have some sort of SQL above but what is the ‘Query Code’?

As per the bottom part of the screenshot above, click on Query Code to see JSON `db.companies.aggregate…`

12. Now search for companies founded after 2010 using SQL. How many are there?

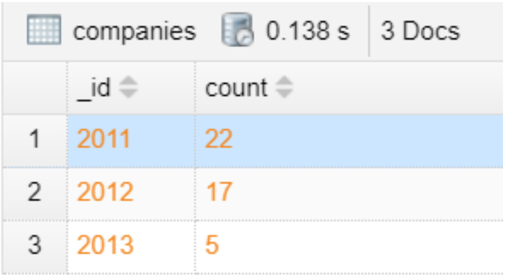
44

You have to write an SQL query and change limit back to e.g. 100.. or you can try figuring out JSON (optional)

13. You can achieve a one-click grouping/filtering of data fields.

Switch to tree view and right click on ‘founded_year’ then ‘Group by…’, then ‘COUNT…’ (see screenshot below)

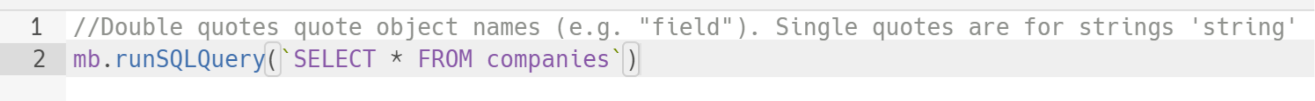
How many companies were founded after 2010? (and if the answer is different to the previous question.. review it).



companies 0.138 s 3 Docs

	_id	count
1	2011	22
2	2012	17
3	2013	5

14. //Double quotes quote object names (e.g. "field"). Single quotes are for strings 'string'



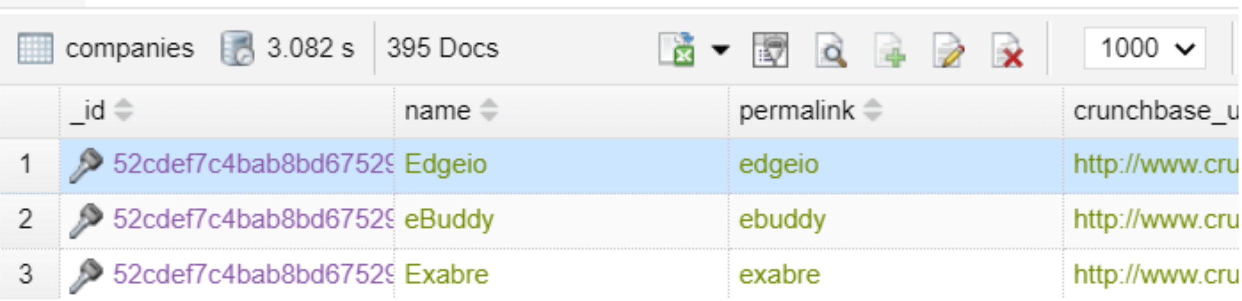
```
//Double quotes quote object names (e.g. "field"). Single quotes are for strings 'string'
mb.runSQLQuery(`SELECT * FROM companies`)
```

As above, on line 1, what does this mean?

so if you want to search for a city called Melbourne, it would have to be "city" 'Melbourne' (see movies exercises below, it's JavaScript)

Try a search to find companies that start with "E" (using SQL), how many are there?

395



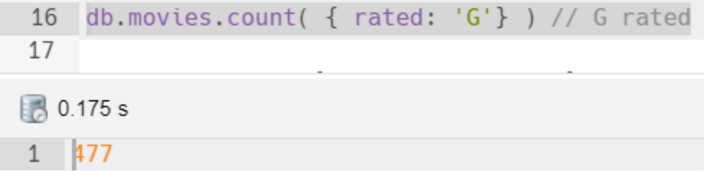
companies 3.082 s 395 Docs

	_id	name	permalink	crunchbase_u
1	52cdef7c4bab8bd6752	Edgeio	edgeio	http://www.cru
2	52cdef7c4bab8bd6752	eBuddy	ebuddy	http://www.cru
3	52cdef7c4bab8bd6752	Exabre	exabre	http://www.cru

18. Find all 'G' rated movies. How many are there?

the hint is to replace 'find' with something else, try count

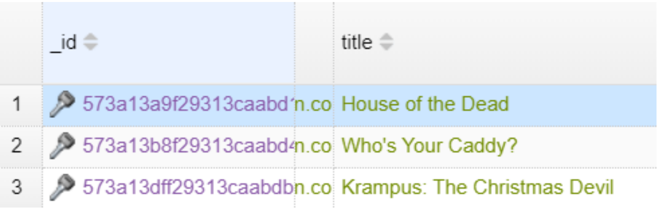
477



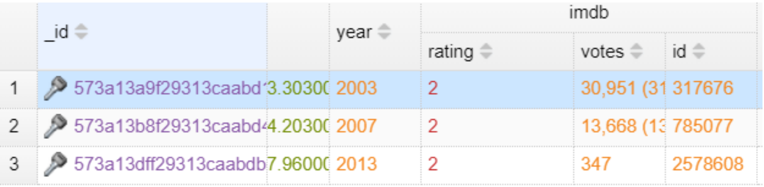
```
db.movies.count( { rated: 'G'} ) // G rated
```

```
0.175 s
1  477
```

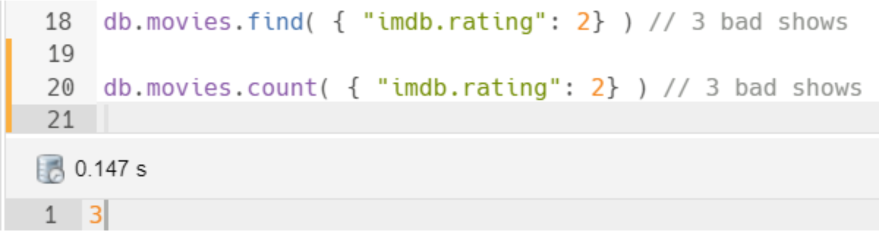
21. How many shows have a rating of 2?



	_id	title
1	573a13a9f29313caabd'n.co	House of the Dead
2	573a13b8f29313caabd'n.co	Who's Your Caddy?
3	573a13dff29313caabdbn.co	Krampus: The Christmas Devil



	_id		year	imdb		
				rating	votes	id
1	573a13a9f29313caabd	3.3030	2003	2	30,951 (31	317676
2	573a13b8f29313caabd	4.2030	2007	2	13,668 (13	785077
3	573a13dff29313caabdb	7.9600	2013	2	347	2578608



```
db.movies.find( { "imdb.rating": 2} ) // 3 bad shows

db.movies.count( { "imdb.rating": 2} ) // 3 bad shows
```

```
0.147 s
3
```

22. How to find the movies with best ‘rating’?

What is $gt?

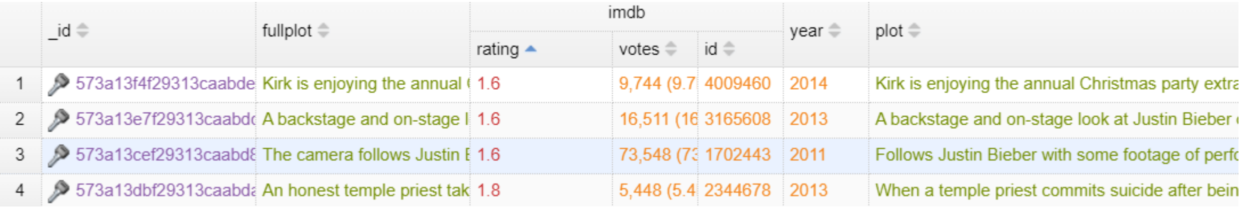
greater than (>)

23. Why are there TWO Shawshank Redemptions?

They have different number of votes

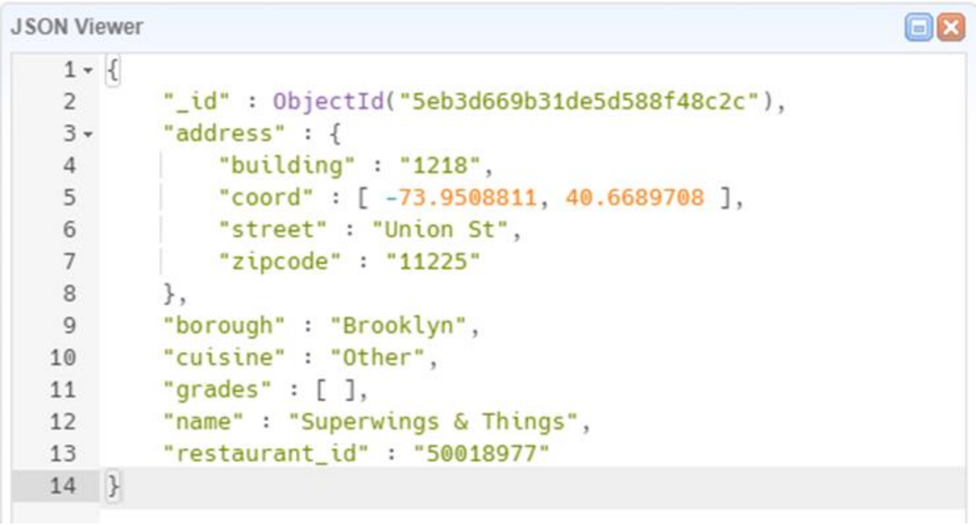
24. Who features in two of the worst movies (by imdb rating)?

So there are three rated 1.6, two feature Beiber



	_id	fullplot	imdb			year	plot
			rating	votes	id		
1	573a13f4f29313caabde	Kirk is enjoying the annual	1.6	9,744 (9.7	4009460	2014	Kirk is enjoying the annual Christmas party extra
2	573a13e7f29313caabd	A backstage and on-stage l	1.6	16,511 (16	3165608	2013	A backstage and on-stage look at Justin Bieber
3	573a13cef29313caabd	The camera follows Justin	1.6	73,548 (73	1702443	2011	Follows Justin Bieber with some footage of perf
4	573a13dbf29313caabd	An honest temple priest tak	1.8	5,448 (5.4	2344678	2013	When a temple priest commits suicide after bein

26. Find the restaurant ‘Superwings and Things”. Move your mouse pointer to the restaurant name, you will see a JSON pop-up describing the object. Press p to see the code in JSON viewer.



```
{
    "_id" : ObjectId("5eb3d669b31de5d588f48c2c"),
    "address" : {
        "building" : "1218",
        "coord" : [ -73.9508811, 40.6689708 ],
        "street" : "Union St",
        "zipcode" : "11225"
    },
    "borough" : "Brooklyn",
    "cuisine" : "Other",
    "grades" : [ ],
    "name" : "Superwings & Things",
    "restaurant_id" : "50018977"
}
```

27. You need to write code adding more attributes to this record. Note, your connection is read-only so you cannot add your changes to MongoDB.

You need to add details of the signature dish:
name “Sticky BBQ wings”
diet: Gluten-free, Halal

The code can be inserted anywhere, e.g. between cuisine and grades

```
        “signature dish” : {
                “name” : “Sticky BBQ wings”,
                “diet” : [“Gluten-free”, “Halal”]
        },
```

D. Exercise. Choosing NoSQL Database

Choosing a NoSQL database

Libraries store information about their collections in their catalogue.

Match each of the following statements to the type of NoSQL database that would be best for storing that library’s data. Select from the four types of NoSQL database discussed previously.

a. In one library, items are catalogued by author, title and publisher, as well as any number of other fields chosen by the cataloguer, such as physical description, subject codes and notes.

   A *column-family database* would be the best choice. Each row in a column-family table may have a different set of columns associated with it.

b. In another library, each catalogue record is stored in the MARC format (Figure 1), a coded text format that contains all the catalogue information for a particular item.

   A *document store* would be best suited to this task. Normally, document stores use a modern data interchange format such as JSON or XML, but industry-specific structured data formats such as MARC can be used with specialised document store systems.

c. A public library wishes to store cover photos of all its items, which might be in JPEG, PNG or PDF format, or stored as a URL.

   Key-value stores can store any kind of data. Each document in a document store should be made up of structured data – images are not structured data in the same way as JSON, so a document store is a poor choice.

d. A university library wishes to keep track of which published academic papers reference each other in order to help researchers measure their metrics.

   By storing papers as nodes and references as edges joining the nodes, a *graph database* can efficiently capture, and answer complex queries about, the relationships between papers.

```
LEADER 00000nam  22000001  4500
008    730220s1955    ilu      b    00000 eng
019    55007351
050 0  QA276.5|b.R3
082    311.22
110 20 Rand Corporation.
245 12 A million random digits|bwith 100,000 normal deviates.
260 0  Glencoe, Ill.,|bFree Press|c[1955]
300    xxv, 400, 200 p.|c28 cm.
504    Bibliography: p. xxiv-xxv.
650  0 Numbers, Random.
984    |cMS T 519 R152
```

Figure 1: An example of a MARC record. MARC is a very old format that predates NoSQL, JSON and even XML by several decades, yet it remains the industry standard in library data systems.

Key Concepts:

NOTE for students: *This is a brief summary of some of the concepts taught in lecture 22. The lectures contain detailed content related to these and many more concepts. These notes should be considered quick revision instead of a sole resource for the course material.*

- What are NoSQL databases?

  A **NoSQL database**, also referred to as a non-relational database, is an approach that provides a facility to store and retrieve data in formats other than tabular form. NoSQL databases do not depend on any particular structure such as tables, rows, columns or schemas to organize data; instead, they use a more flexible model. With the rapid evolution in the nature of data, the needs of next-generation data storage and analysis, and requirements of intensive but flexible data analysis using distributed systems, cloud computing and high-performance computing (HPC), traditional relational databases are unable to meet *performance*, *scalability* and *flexibility* requirements. Examples of unstructured but exponentially-growing data include chat data, messaging, large objects such as videos and images and many types of business documents.

- Types of NoSQL database

  There are four main categories of NoSQL databases:

  - Graph databases

    Graph databases are based on graph theory and utilize the concept of a *graph* to store, connect and query data. In a graph database, *nodes* are equivalent to rows or records in the relational database and represent entities such as accounts, people, items etc. Nodes are which are linked together by edges. Edges connect nodes and resemble the relational relationship between tables. Both nodes and edges can have properties associated with them.

    A well-known example of a graph database is *neo4j*, used by Airbnb, Microsoft, IBM, eBay and Walmart.

  - Key-value stores

Key-value stores are the most flexible NoSQL databases, and also the least structured, using a simple key-value structure to organize data. There is no schema and the data values can be of any data type. Similar to a dictionary structure, the key should be a unique identifier to allow retrieval of the associated value. The key can theoretically be anything, but certain limitations can be imposed by the DBMS such as the key size and key type to achieve better performance. The value however can be anything, such as images, long text, videos, binary data, lists, JSON data, numbers, etc.

Examples of key-value stores include Berkeley DB, Aerospike and Redis. Key-value stores are highly flexible and support massive scalability.

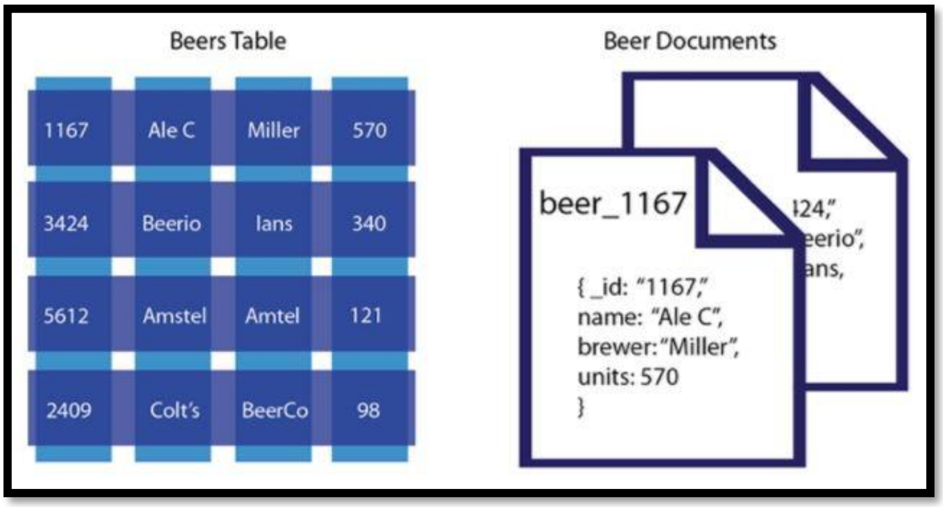
○ Column-family stores

Column-family stores, also referred to as wide-column stores or extensible record stores, are a type of key-value database. Column-family DBs, like relational databases, use tables, rows and columns but for each record the column names, their format and record keys can greatly vary, hence resulting in a schema-free structure. This enables organizations to store and query semi-structured data. Columns are created for each row instead of being predefined for a table.

Cassandra and Hbase are two important NoSQL wide-column databases used by Facebook in the past to handle messaging and inbox search. Other enterprises using wide column stores are Netflix, Twitter and Reddit.

○ Document stores

Document stores typically store data in JSON, XML or BSON documents (where JSON is by far the most dominant).



1167	Ale C	Miller	570
3424	Beerio	Ians	340
5612	Amstel	Amtel	121
2409	Colt's	BeerCo	98

```
{ _id: "1167,"
name: "Ale C",
brewer:"Miller",
units: 570
}
```

Figure S1: A comparison between a relational Beers table and a document store containing the same data. Adapted from https://developer.couchbase.com/comparing-document-vs-relational/

The resulting documents are independent components which can be distributed more easily. In addition, the storage does not require compliance with a set schema. Instead, each document can have its own structure and schema, allowing greater agility and flexibility as the business, website or app evolves. Even though there is no fixed schema, it is still possible to create indexes within documents. If indexing features associated with document databases are used to their fullest, they can provide fast and efficient querying of data.

MongoDB is an example of a document store.

- Advantages of NoSQL

There are four key advantages offered by NoSQL databases as compared to relational databases:

Flexible modelling – Instead of relying on a fixed schema, data types, row size and column names, NoSQL facilitates the implementation of flexible data models, making it more suited to coping with less structured data sources such as crowdsourced data.

Scalability – Capacity in a NoSQL database can be added and removed quickly using a horizontal scale-out methodology (adding inexpensive servers and connecting them to a database cluster). As a result, the cost and complexity associated with scaling up a relational database into a distributed database are avoided.

Performance – By achieving seamless scalability using the horizontal scale-out methodology, the enterprises can manage efficient reads, writes and storage of the data items when handling big data. Companies like LinkedIn, Facebook and Google have users around the world; therefore, they deploy data centres in different parts of the world and partition their users so that all of their users experience the fewest possible hops by being routed to the closest data centre.

High availability – With many businesses' customer and user engagement taking place mostly or entirely online, the availability of any application is a major concern for enterprises. Constant availability (24/7) is a challenge for relational databases. In contrast, NoSQL databases are typically stored in partitions and they divide data across multiple database instances without any shared resources. The automatic failover means that if nodes fail, the database can continue its read and write operations on a different node.