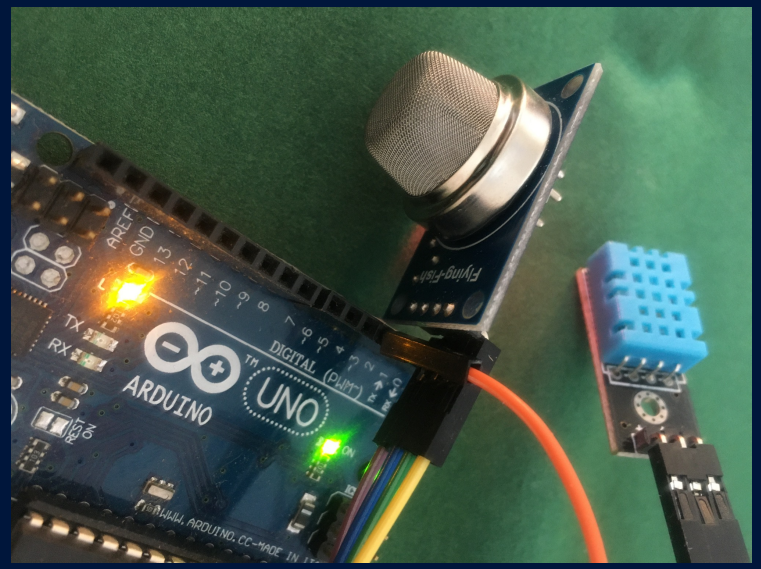
cphbusiness

COPENHAGEN BUSINESS ACADEMY

DATA ENGINEERING with MongoDB



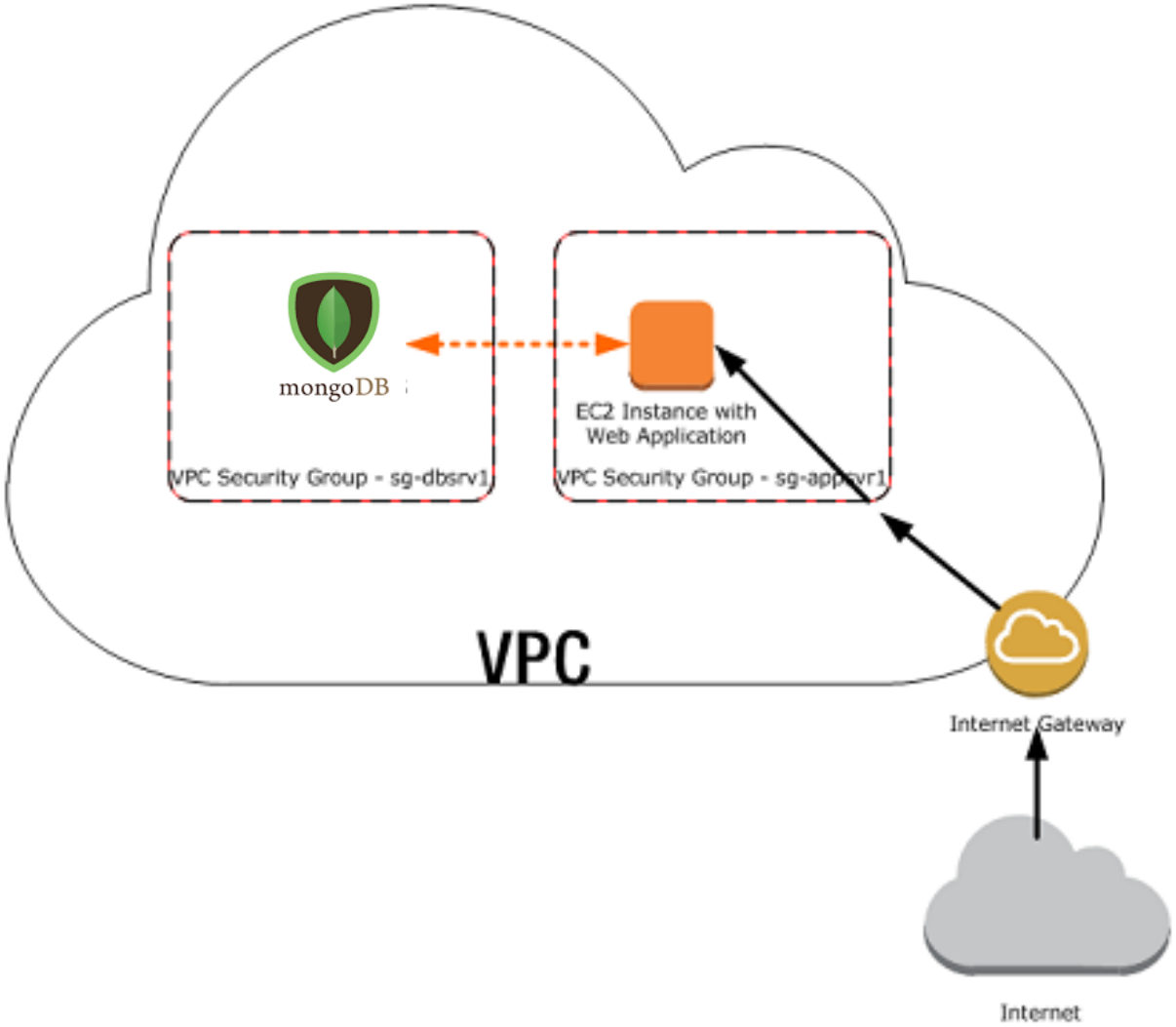
Agenda – MongoDB

- AWS – hvor er vi
  - OLA
  - Databaser
- MongoDB - overview
  - Anvendelse
  - NoSQL vs SQL
- MongoDB
  - CRUD
  - Temrinal eller GUI
  - Øvelser i mongosh
    - Forbind til min database
    - Indlæs restaurent
- Arbejde med aws-opgaven i OLA 5

Øvelser – MongoDB

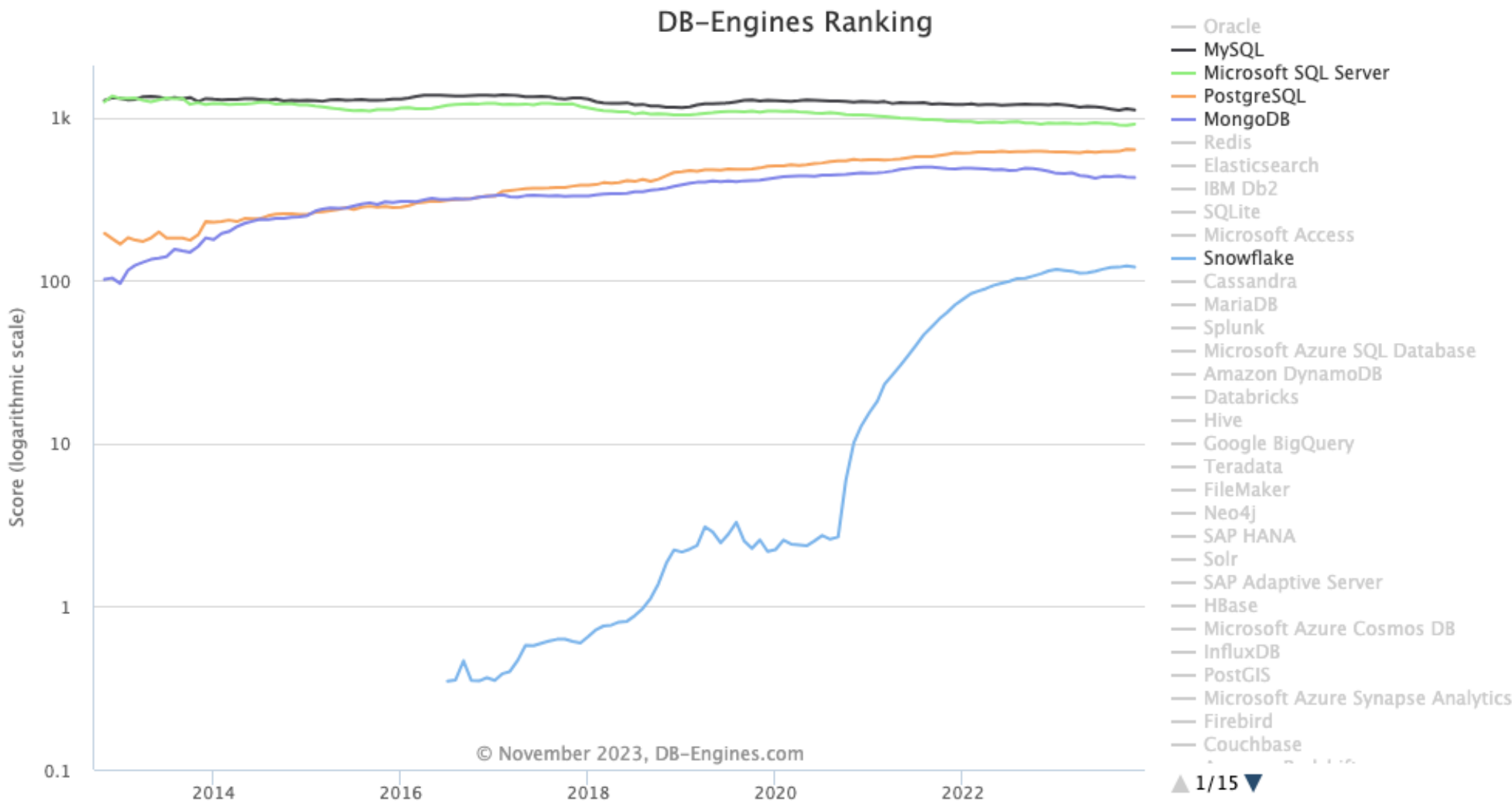
- One House skal modificeres så vardata bliver et array af pris-observationer
- Løs de første opgaver (1-9)
- Løs følgende
  - forbind til min boligsiden-database
  - hvor mange boliger?
  - hvor mange boliger over 4000000?
  - hvor mange over 200 m2?
  - hvor mange over 4000000 og i Charlottenlund?
  - hvor mange er over 200 kvm og bygget før 1950?
  - hvor mange er over 400000 og under 1000000?
  - hvor mange ligger i Odense?
  - hvor mange unikke postnummre?
  - hvor mange ligger i postnumre 8000,8200,8230,8340?
- Løse resten af opgaverne fra Word documentet

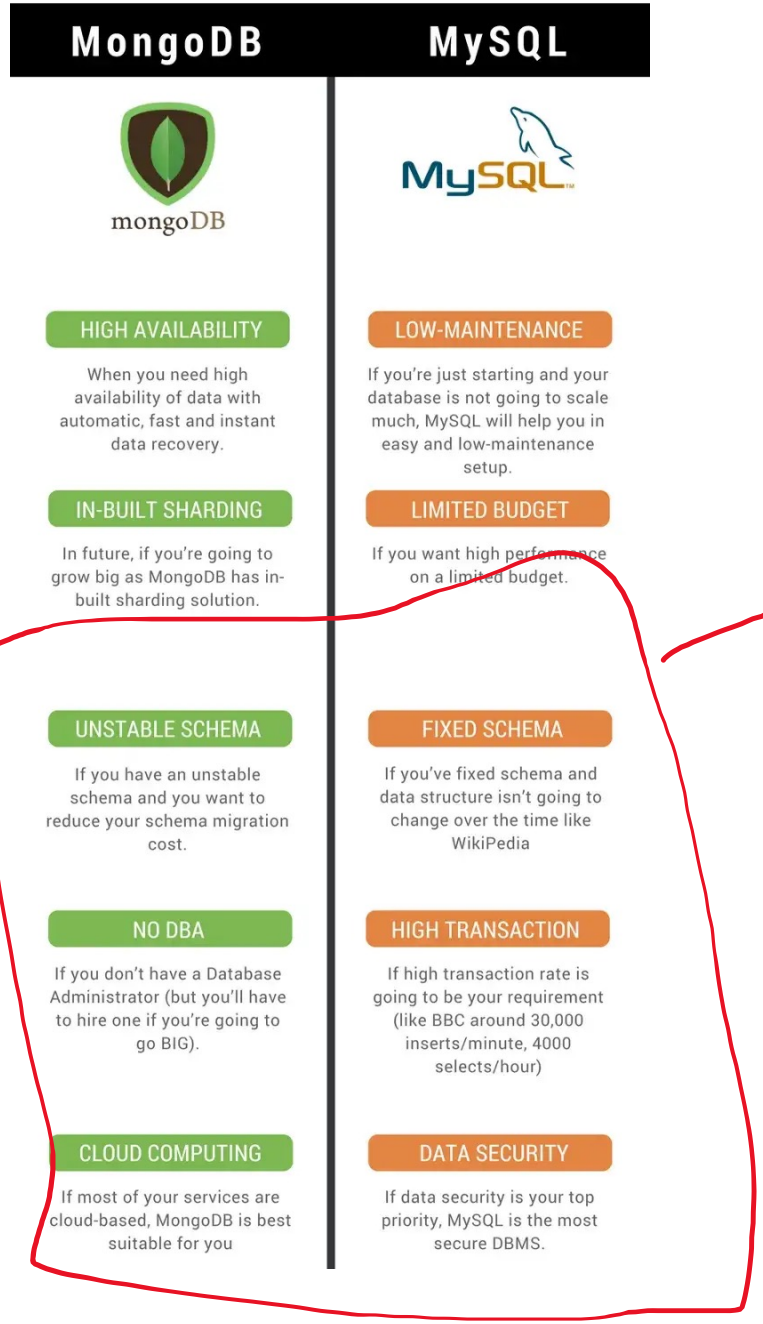
Bilbasen – slutmålet



- **MongoDB**
  - Create AWS EC2 instance
  - Get rdp-access
  - Client access
    - mongosh
  - Dev access
    - mongolite
- **SQL**
  - MySQL on EC2
  - Client access
    - MySQL-Workbench
  - Dev access
    - RMariaDB
- **Files**
  - Create AWS S3 bucket
  - Dev access
    - aws.s3

November 2023





UNSTABLE SCHEMA

If you have an unstable schema and you want to reduce your schema migration cost.

FIXED SCHEMA

If you've fixed schema and data structure isn't going to change over the time like WikiPedia

NO DBA

If you don't have a Database Administrator (but you'll have to hire one if you're going to go BIG).

HIGH TRANSACTION

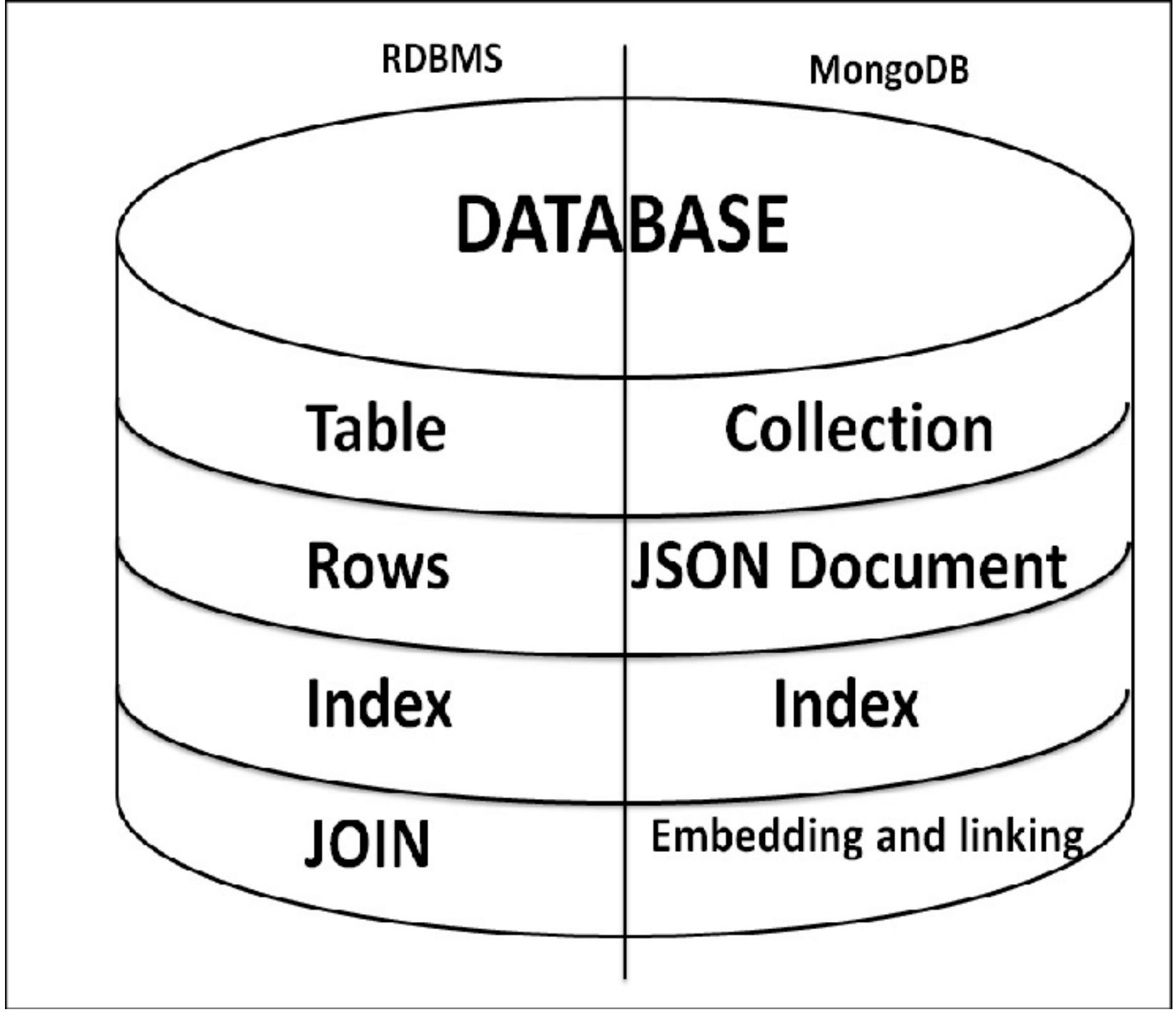
If high transaction rate is going to be your requirement (like BBC around 30,000 inserts/minute, 4000 selects/hour)

CLOUD COMPUTING

If most of your services are cloud-based, MongoDB is best suitable for you

DATA SECURITY

If data security is your top priority, MySQL is the most secure DBMS.


RDBMS
MongoDB
DATABASE
Table
Collection
Rows
JSON Document
Index
Index
JOIN
Embedding and linking

Data Models: Relational to Document

Relational

Person:

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rom

no relation

Car:

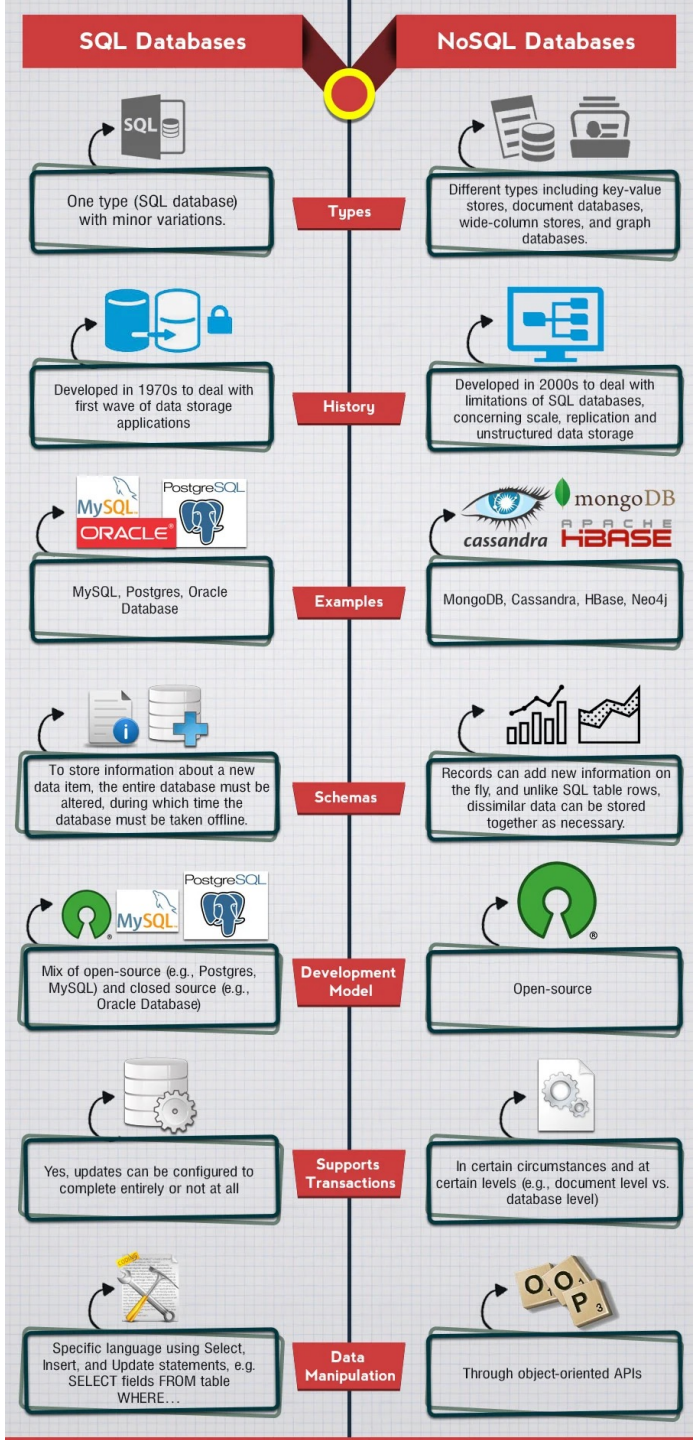
Car_ID	Model	Year	Value	Pers_ID
101	Bentley	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

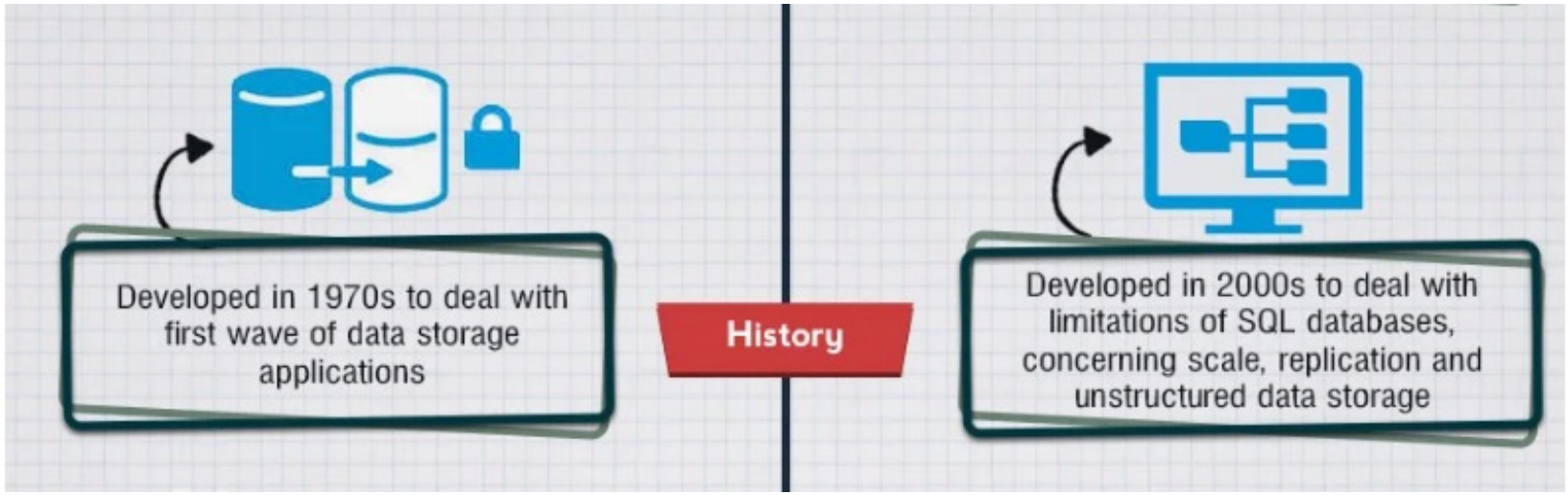
MongoDB Document

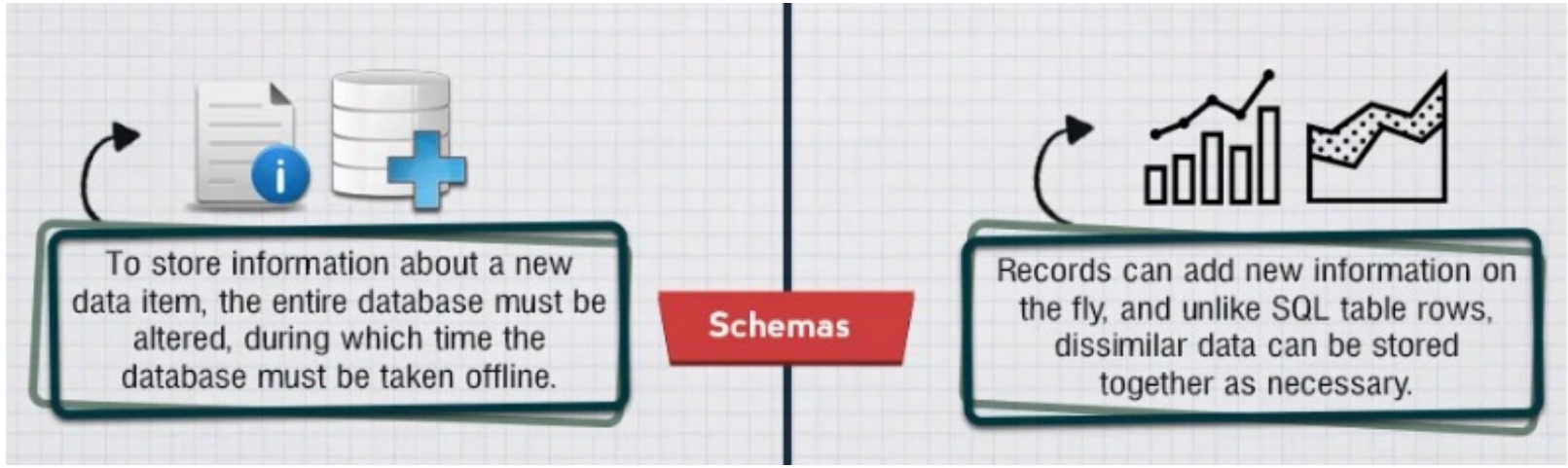
```
{
  first_name: 'Paul',
  surname: 'Miller'
  city: 'London',
  location: [45.123,47.232],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, ... },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, ... }
  ]
}
```


mongoDB

```
{    customer_id : 1,
     first_name : "Mark",
     last_name : "Smith",
     city : "San Francisco",
     phones: [ {
          type : “work”,
          number: “1-800-555-1212”
     },
     {    type : “home”,
          number: “1-800-555-1313”,
          DNC: true
     },
     {    type : “home”,
          number: “1-800-555-1414”,
          DNC: true
     }
   ]
}
```


SQL Databases
NoSQL Databases
One type (SQL database) with minor variations.
Types
Different types including key-value stores, document databases, wide-column stores, and graph databases.
Developed in 1970s to deal with first wave of data storage applications
History
Developed in 2000s to deal with limitations of SQL databases, concerning scale, replication and unstructured data storage
MySQL, Postgres, Oracle Database
Examples
MongoDB, Cassandra, HBase, Neo4j
To store information about a new data item, the entire database must be altered, during which time the database must be taken offline.
Schemas
Records can add new information on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary.
Mix of open-source (e.g., Postgres, MySQL) and closed source (e.g., Oracle Database)
Development Model
Open-source
Yes, updates can be configured to complete entirely or not at all
Supports Transactions
In certain circumstances and at certain levels (e.g., document level vs. database level)
Specific language using Select, Insert, and Update statements, e.g. SELECT fields FROM table WHERE...
Data Manipulation
Through object-oriented APIs


Developed in 1970s to deal with first wave of data storage applications
History
Developed in 2000s to deal with limitations of SQL databases, concerning scale, replication and unstructured data storage


To store information about a new data item, the entire database must be altered, during which time the database must be taken offline.
Schemas
Records can add new information on the fly, and unlike SQL table rows, dissimilar data can be stored together as necessary.

NoSQL Landskab

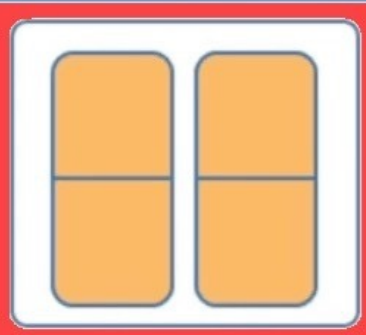
Key Value

Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

Document-Based

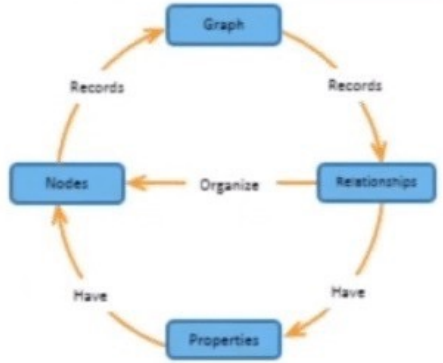
Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based

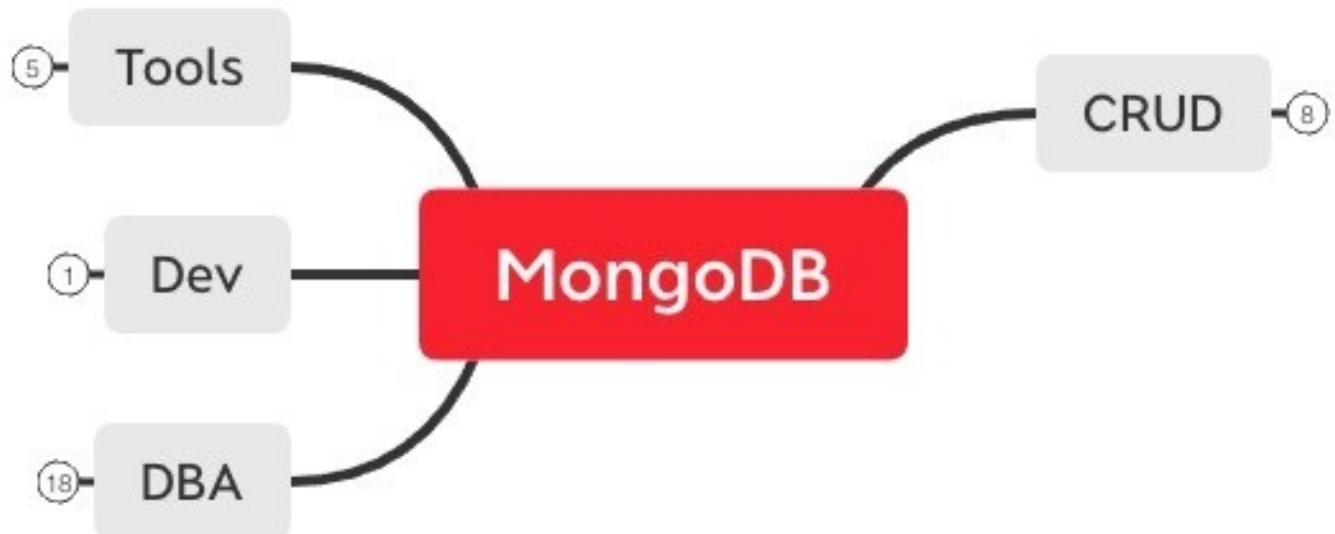


Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based



Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB


Tools
5
Dev
1
DBA
18
MongoDB
CRUD
8

MongoDB – CRUD-operations

C	• insert()
R	• find()
U	• update()
D	• remove()

Query Selectors : Comparison	
$gt:Val	Greater then Val
$gte:Val	Greater then equals Val
$lt:Val	Lower then Val
$lte:Val	Lower then equals Val
$all:Array	All Array elements are included in field array value
$in:Array	Elements with values contained in Array
$nin:Array	Elements with values Not contained in Array
$ne:Val	Not equal

Val can be any Scalar Integer, String, Date, etc

$exists

```
db.users.find({ name: { $exists: true } })
```

$not

```
db.users.find({ name: { $not: { $eq: "Kyle" } } })
```

```
db.users.insertOne(          ← collection
  {
    name: "sue",             ← field: value
    age: 26,                 ← field: value
    status: "pending"        ← field: value
  }                          } document
)
```

```
db.users.find(                          ← collection
   { age: { $gt: 18 } },                ← query criteria
   { name: 1, address: 1 }              ← projection
).limit(5)                              ← cursor modifier
```

```
db.users.updateMany(                    ← collection
   { age: { $lt: 18 } },                ← update filter
   { $set: { status: "reject" } }       ← update action
)
```

```
db.users.deleteMany(                    ← collection
   { status: "reject" }                 ← delete filter
)
```

R Mongolite – CRUD-operations

mongoDB

```
find(
        query = '{}',
        fields = '{"_id" : 0}',
        sort = '{}',
        skip = 0,
        limit = 0,
        handler = NULL,
        pagesize = 1000
)
```

Retrieve fields from records matching query. Default handler will return all data as a single dataframe.

```
insert(
        data,
        pagesize = 1000,
        stop_on_error =TRUE,
...)
```

Insert rows into the collection.
Argument 'data' must be a **data-frame**, named list (for single record) or character vector with json strings (one string for each row).

```
update(
        query = '{}',
        update = '{"$set":{}}',
        upsert = FALSE,
        multiple = FALSE
)
```

Modify fields of matching record(s) with value of the update argument

MongoDB – CRUD-operations

C	• insert()
R	• find()
U	• update()

https://github.com/jeroen/mongolite
https://jeroen.github.io/mongolite/index.html
https://cran.r-project.org/web/packages/mongolite/mongolite.pdf

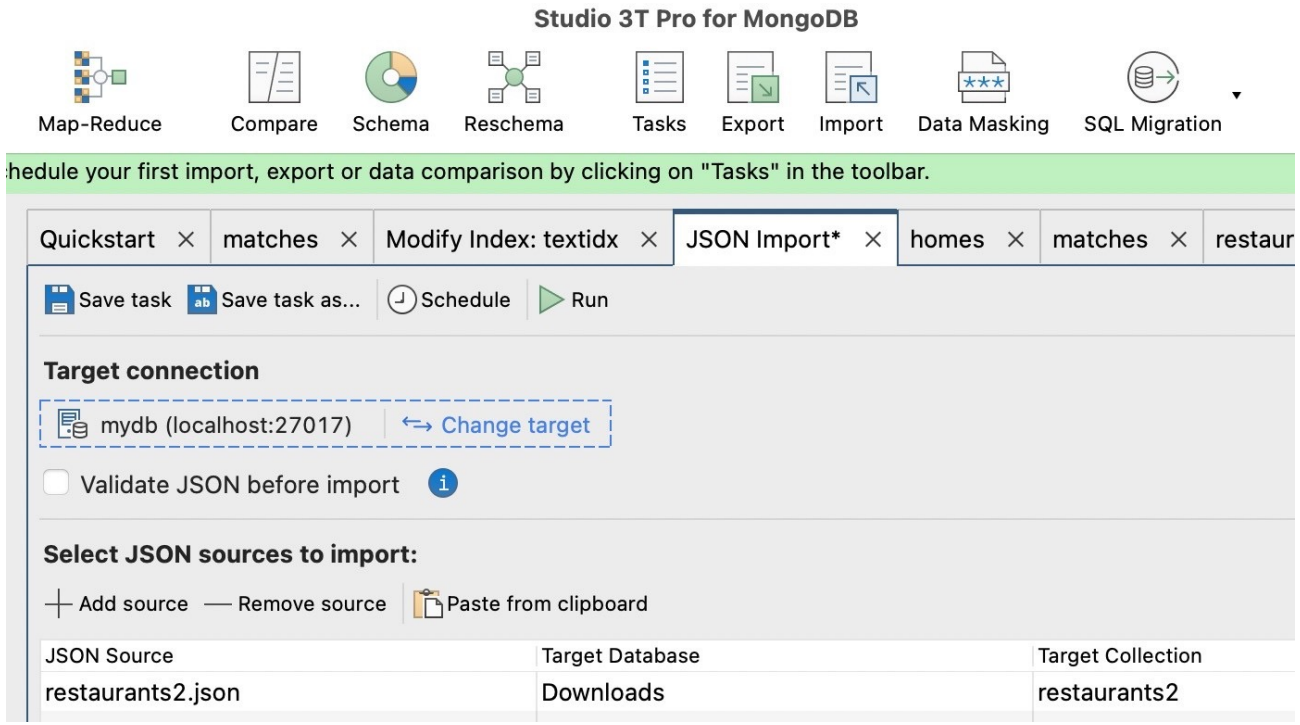
```
find(
  query = '{"domain" : { "$regex" : "horton" } }'
)
```

Retrieve fields from records matching query. Default handler will return all data as a single dataframe.

import data - monogsh tool

```
ls evt_5* | while read x; do mongoimport -d=wyscoutdutch -c=games $x; done
mongoimport --uri "mongodb+srv://soccer.cukt54z.mongodb.net/myFirstDatabase&authSource=admin" -d=wyscoutdutch
```

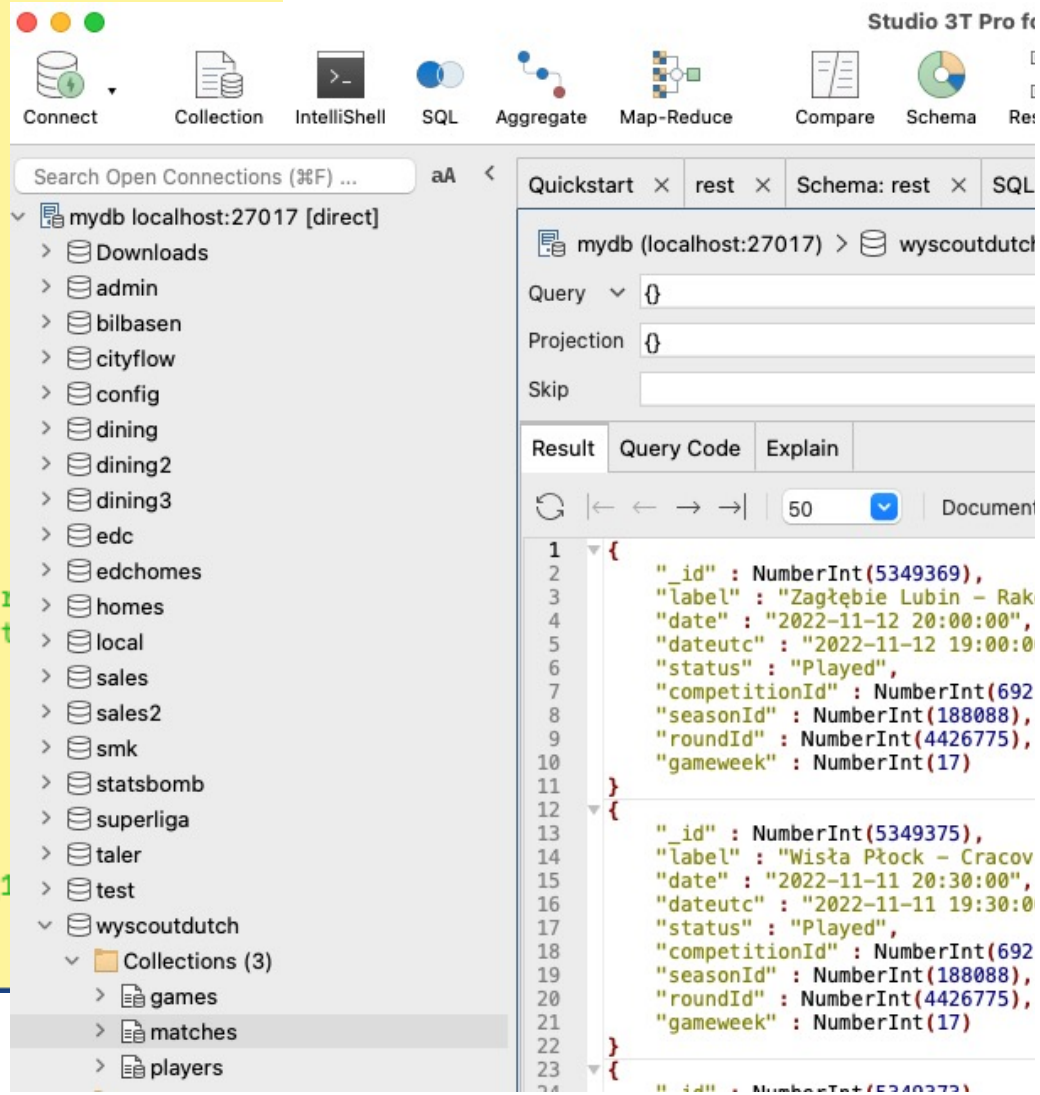
Or GUI



verify data - mongo shell

```
dining2> use wyscoutdutch
switched to db wyscoutdutch
wyscoutdutch> show collections;
games
matches
players
wyscoutdutch> db.players.findOne()
{
  _id: 869465,
  shortName: 'T. van Bergen',
  firstName: 'Thom',
  middleName: '',
  lastName: 'van Bergen',
  height: 0,
  weight: 0,
  birthDate: '2004-01-06',
  birthArea: { id: 528, alpha2code: 'NL', alpha3code: 'NLD', na
  passportArea: { id: 528, alpha2code: 'NL', alpha3code: 'NLD',
  role: { name: 'Forward', code2: 'FW', code3: 'FWD' },
  foot: null,
  currentTeamId: 23564,
  currentNationalTeamId: null,
  gender: 'male',
  status: 'active',
  imageDataURL: 'https://cdn5.wyscout.com/photos/players/public/ndplayer_1
}
wyscoutdutch>
```

Or GUI



work on data - CRUD shell/Gui

```
sales2> db.sales2.find({"storeLocation":/Lond/},{"storeLocation":1,"items.quantity":1,saleDate:1,_id:0}).limit(2)
[
  {
    saleDate: ISODate("2014-11-11T02:13:51.893Z"),
    items: [
      { quantity: 4 },
      { quantity: 7 },
      { quantity: 5 },
      { quantity: 5 },
      { quantity: 3 },
      { quantity: 5 },
      { quantity: 1 },
      { quantity: 2 },
      { quantity: 3 },
      { quantity: 1 }
    ],
    storeLocation: 'London'
  },
sales2> db.sales2.find({"storeLocation":/Lond/},{"storeLocation":1,max:{$max:"$items.quantity"},saleDate:1, _id:0}).limit(2)
[
  {
    saleDate: ISODate("2014-11-11T02:13:51.893Z"),
    storeLocation: 'London',
    max: 7
  },

sales2> db.sales2.updateOne({_id:ObjectId("5bd761dcae323e45a93ccff3")},{$set:{storeLocation:"Nyborg"}})
```

export data - mongodb tool

```
| => mongoexport -v --collection=sales --db=sales -q='{"storeLocation":"London"}' --pretty --jsonArray --out dumplondon.json
2022-03-05T12:44:49.136+0100    will listen for SIGTERM, SIGINT, and SIGKILL
2022-03-05T12:44:49.142+0100    connected to: mongodb://localhost/
2022-03-05T12:44:49.242+0100    exported 794 records
```

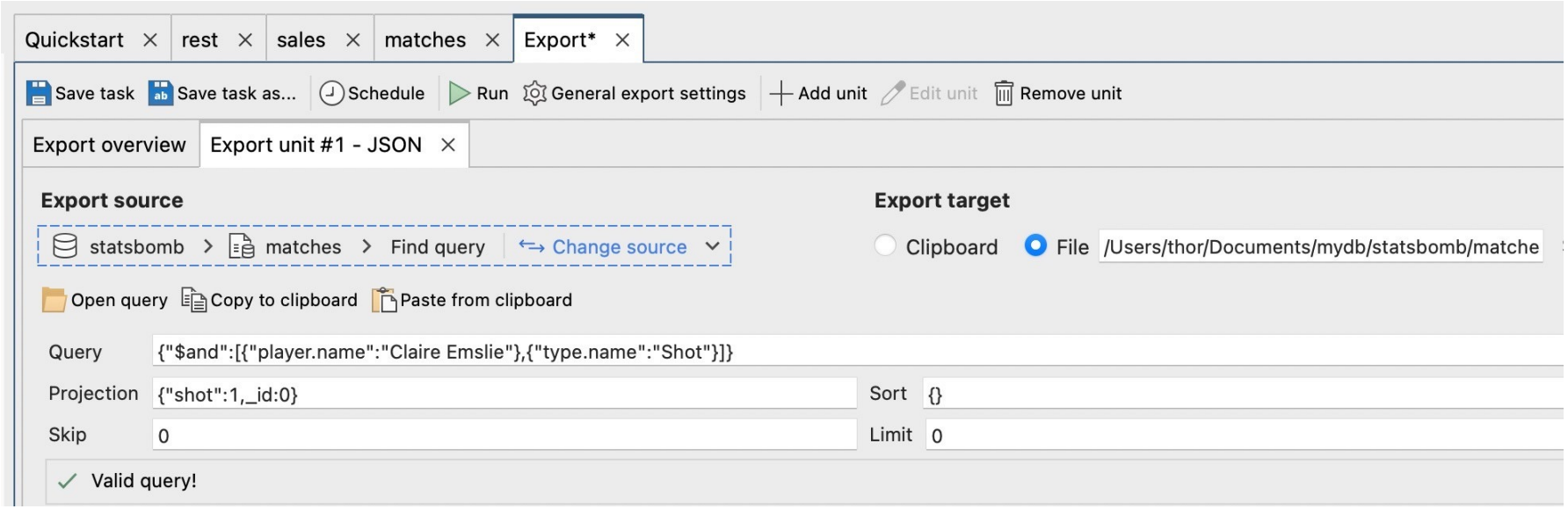
```
| => head dumplondon.json
[{
        "_id": {
                "$oid": "5bd761dcae323e45a93ccfed"
        },
        "saleDate": {
                "$date": "2015-09-02T16:11:59.565Z"
        },
        "items": [
                {
                        "name": "binder",
```

export data - mongodb tool

```
| => mongoexport -v --collection=sales --db=sales -q='{"storeLocation":"London"}' --pretty --jsonArray --out dumplondon.json
2022-03-05T12:44:49.136+0100    will listen for SIGTERM, SIGINT, and SIGKILL
2022-03-05T12:44:49.142+0100    connected to: mongodb://localhost/
2022-03-05T12:44:49.242+0100    exported 794 records
```

```
mongoexport --db=statsbomb --collection=matches --query='{"player.name":{ "$in": ["Claire Emslie","Lucy Graham"]}}' --out
dump.json
```

Or GUI



Update MongoDB from R

https://jeroen.github.io/mongolite/

```
restorig   10 obs. of 8 variables
rests      10 obs. of 9 variables
```

Untitled1*

```r
library(mongolite)

con <-mongo(
  collection = "rest",
  db = "dining2",
  url = "mongodb://localhost",
  verbose = TRUE
)

restorig <- con$find(limit=10)

numt=2
for (i in 1:ncol(rests)) {
  numt<-numt*0.95
  rests[i,"Viggo"]=numt[1]
}

for (i in 1:ncol(rests)) {
  key <- rests[i,6]
  val <- rests[i,9]
  con$update(
    query=paste0('{"restaurant_id": "',key,'"}'),
    update=paste0('{"$set":{"lbl": "',val,'"}}')
  )
}
```

From mongo

name	restaurant_id	age	lbl
Wendy'S	30112340	30112340	kurt
Wilken'S Fine Food	40356483	30112340	kurt
Brunos On The Boulevard	40356151	30112340	kurt
Riviera Caterer	40356018	30112340	kurt
Kosher Island	40356442	30112340	kurt
Taste The Tropics Ice Cream	40356731	30112340	kurt
Regina Caterers	40356649	30112340	kurt
Dj Reynolds Pub And Restaurant	30191841	30112340	NA
Morris Park Bake Shop	30075445	30112340	NA
May May Kitchen	40358429	30112340	NA

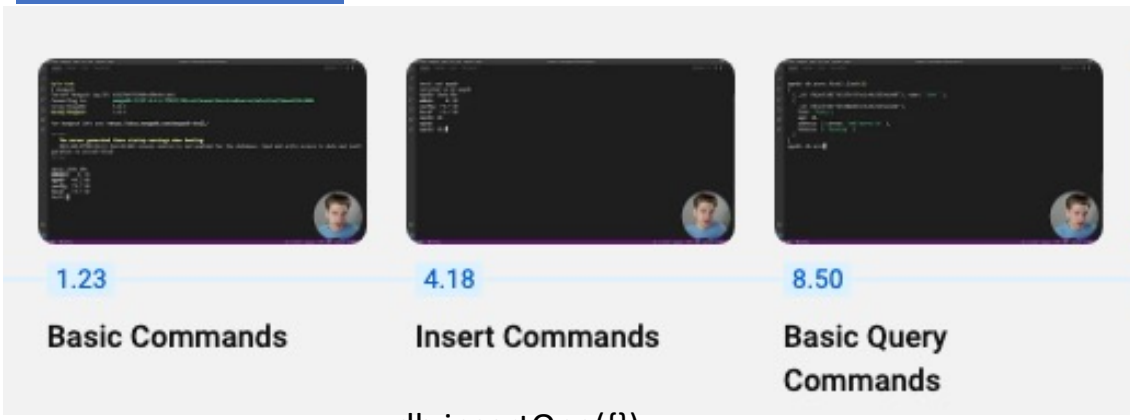
Modif dataframe

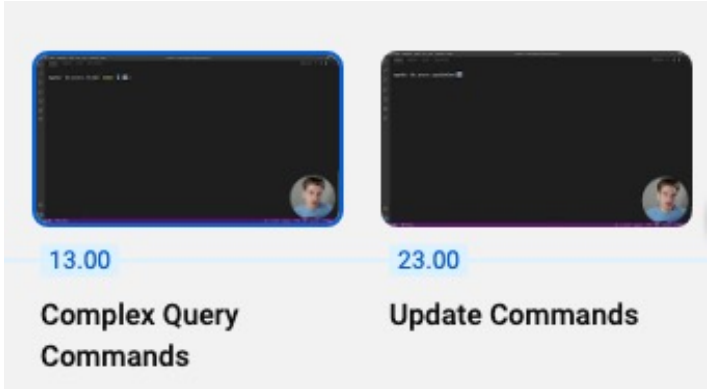
name	restaurant_id	age	lbl	Viggo
Wendy'S	30112340	30112340	kurt	1.900000
Wilken'S Fine Food	40356483	30112340	kurt	1.805000
Brunos On The Boulevard	40356151	30112340	kurt	1.714750
Riviera Caterer	40356018	30112340	kurt	1.629012
Kosher Island	40356442	30112340	kurt	1.547562
Taste The Tropics Ice Cream	40356731	30112340	kurt	1.470184
Regina Caterers	40356649	30112340	kurt	1.396675
Dj Reynolds Pub And Restaurant	30191841	30112340	NA	1.326841
Morris Park Bake Shop	30075445	30112340	NA	NA
May May Kitchen	40358429	30112340	NA	NA

Update mongi via R

Kyle

db.insertOne({}) db.find({},{})





show
use

db.insertOne({})
db.insertMany({})

db.find({},{}).limit(2)
.sort({})
.skip(3)

WHERE
df.find({name:”Ib”},{})
SELECT
df.find({},{name:1,_id:0})

WHERE
df.find({name: {$eq:”Ib” }})
$neq
$gt
$in:[]
$nin:[]
$exists: true
$gt:20,$le:2
db.find({$or: [{ age: {$lt:1}},{ name:”Ib”}]
db.find({$expr: {$gt:[”$d”,”$c”]}})
db.find({”vardata.pris”: { $gt:34000}})