MongoDB

databases: storage to store huge amount of data
Types:

1. Structured Database
   - Relational Database storing data in relational way
   - store it in tables
   - creating customer table, product tables making relation ships between them
   - creating employee table and department table and making relationship
   - relationship: one to many, many to one, many to many , one to one
   - example: MySQL, PostgreSQL, Oracle, MySQL Server, MS Access
2. Non-structured Database
   - its not following any structure but storing data in some form like JSON or BSON (binary JSON)
   - Big data for that we can use non-structured DB
   - eg. Mongo DB, Couch DB, Cassandra

MongoDB
Mongodb noSQL, open-Source, document-oriented DB.
Well designed to provide high performance, high availability and easy scalability
Developed By MongoDB Inc.
written in lang C++

Compare SQL with MongoDB

SQL	MongoDB
Table	Collection
Row	Document
Column	Field
join	embedded or linked Doc
Schema	Dynamic Schema

MongoDB Terminology

Database: Container of Collection

Db name is Ecommerce, then store collections of customer, products, orders, categories

Collection: Group where you store similar type of data

Customer collection can have customer data

product collection can have product data

Document:

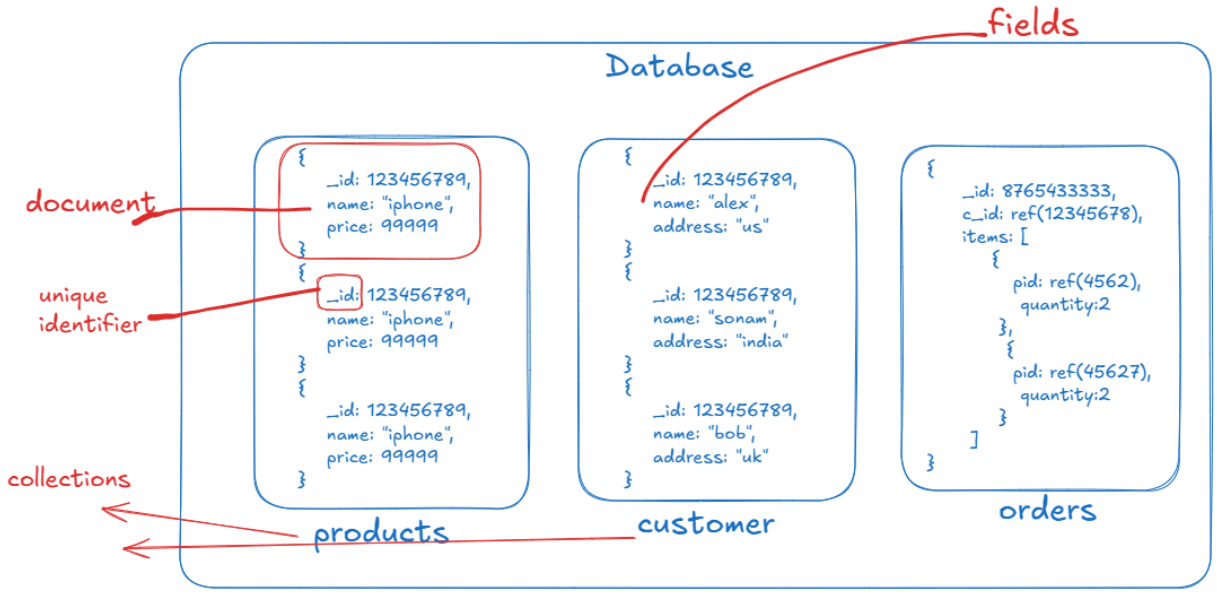
individual record

like one product object

one customer object

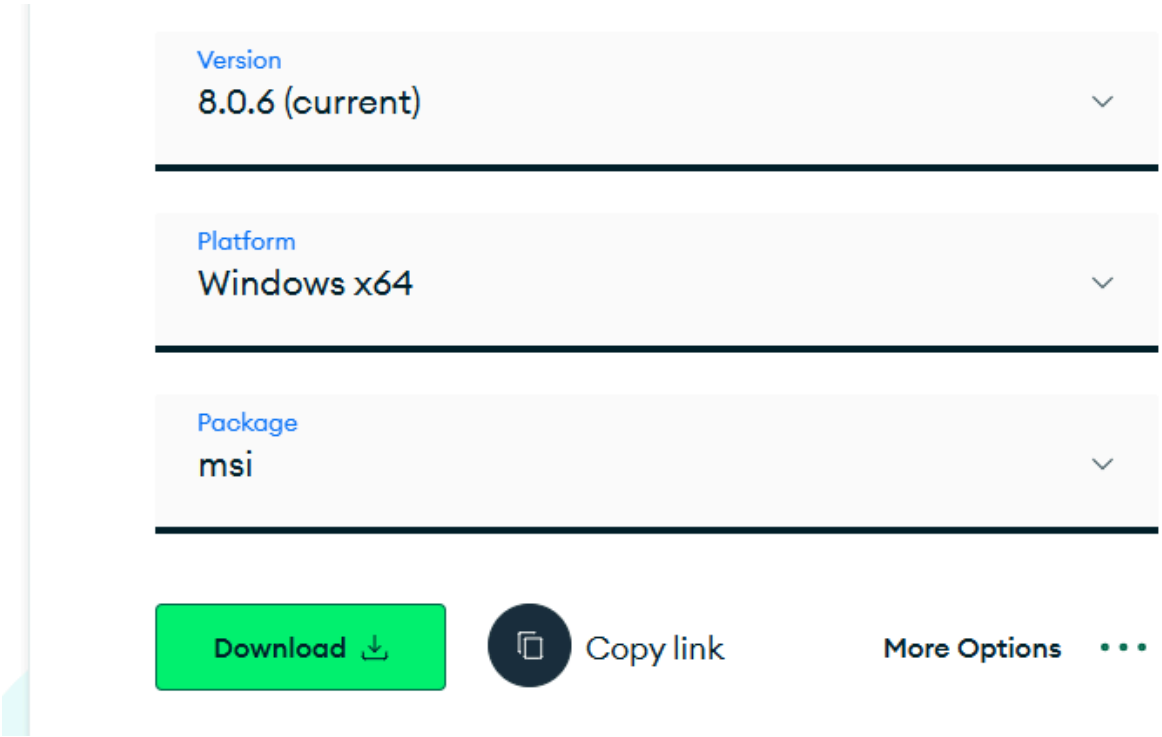
Field: a key-value pair.

_id: a unique identifier automatically added to each Document



How to use MongoDB database Locally.

https://www.mongodb.com/try/download/community (open this and scroll down a bit)



Download this.
just double click on that follow the installation process and finish.
Once you install this it will give you a prompt to install mongoDB Compass That also you install.
Once it is installed.
open mongodb Compass.
click on + icon and create connection by using default string.
click on open mongodb shell icon and execute commands.
to see the available databases: show databases
By default the database is test so if you want to create and use your Db then: **use skillacademy**

```
> use skillacademy
< switched to db skillacademy
> db.createCollection("student")
< { ok: 1 }
> db.student.insertOne({ name: "John Doe",age:22,course:"Devops"})
< {
    acknowledged: true,
    insertedId: ObjectId('67f14dcbaf34da616d09e2e7')
  }
> db.student.find()
< {
    _id: ObjectId('67f14dcbaf34da616d09e2e7'),
    name: 'John Doe',
    age: 22,
    course: 'Devops'
  }
```

What if we want to insert many Records:

```
> db.student.insertMany([
    {name:"Alice", age:24, course: "Full Stack"},
    {name:"Bob", age:21, course: "Data Science"},
    {name:"charlie", age:23, course: "DevOps"}
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('67f14ed5af34da616d09e2e8'),
      '1': ObjectId('67f14ed5af34da616d09e2e9'),
      '2': ObjectId('67f14ed5af34da616d09e2ea')
    }
  }
```

db.student.find().size() # get size in DB
db.student.find({name:'Alice'}) # find student whose name is Alice
db.student.find({age: {$gt:22}}) # find students whose age is gretaer than 22
db.student.find({course: {$ne:'Full Stack'}}) ## get All except Full Stack Student

```
db.student.find({course: {$eq:'Full Stack'}}) or
db.student.find({course:'Full Stack'}) ## both give Same result for full stack
Student
## Fetch limited Field
db.student.find({},{name:1,course:1}) ## it will fetch name, course, Id also
come by default
## if you want to skip ID
db.student.find({},{name:1,course:1,_id:0}) ## it will fetch only name and
course field

## Combine Conditions
db.student.find({
$or :
[
{age: {$gt:22}} ,
```

```
{course: {$eq:'Full Stack'}}
]
})

## And Operator
db.student.find({
$and :
[
{age: {$gt:22}} ,
{course: {$eq:'Full Stack'}}
]
})
```

Update Query:

```
db.student.updateOne(
{ name: 'John Doe'},
{ $set : {course: 'Cloud Computing'} }
)
## Update Student named John Doe Course Cloud Computing
## Add Some More Records
db.student.insertMany([
{name:"Catty", age:34, course: "Full Stack"},
{name:"Devid", age:31, course: "Data Science"},
{name:"Jack", age:45, course: "DevOps"}
])
## Update Many
db.student.updateMany(
{ course: 'Full Stack'},
{ $set : {course: 'MERN Stack'} }
)
##find students from the course data Science and cloud computing
db.student.find({course: {$in: ['Data Science','Cloud Computing']}})
## Anather way for above solution
db.student.find({$or:
[
{course: 'Data Science'},
{course: 'Cloud Computing'}
]
})
```