- db = avoir le nom de la base courante
- show dbs =liste des bases existantes
- use nom_database = changer la base courante (si elle n'existe pas il la crée)
- show collections = liste des collections existantes
- db.nom_collection.insert({ attr1 : " valeur1 ", attr2 : " valeur2 "}) = insérer un document avec deux attribut dans une collection (si elle n'existe pas elle sera créé)
  un champ _id soit il est inséré par l'user soit générer automatiquement et affiché
- db.nom_collection .find() = liste des documents de la collection
- db.nom_collection .find({critère1 : val1, critère2 : val2)}) = recherche selon des critères (les : au sens de égale) on pt ajouter les opérandes aux critères :
  db.produits.find({critère1: {$gte:525 }, cent:{$lt:90} }) // un seul paramètre
- les opérateurs :
- db.nom_collection .find().count() = calcule le nombre de résultats trouvé
- db.nom_collection.find({critère: 525} , {attribut1:1}) // deux paramètres : le premier normal le deuxième on l'ajoute pour mentionner qu'on ne veut afficher que cet attribut dans les résultat
  NB : le _id s'affiche aussi pour ne pas l'afficher on met comme deuxième attribut :
  {_id:0, attribut1:1}
- db. nom_collection.find({ attribut1 : {$in: [val1, val2, val3]}}).sort({attribu1:1})
  dans un premier temps on aura les résultats où l'attribut vaut val1 si on tape it pleusieurs fois
  on aura ceux où l'attribut vaut val1 ainsi de suite
  pour trier les résultat au sens contraire on utilise -1 à la place de 1

- db. nom_collection.update({critère: val} , {$set : {attribut:val2}}) // deux paramètres 1er là où la mise à jour doit etre effectuée et le 2ème la modification de la valeur de l'attribut ou son ajout s'il n'existe pas
  Cette commande ne modifie que le premier résultat vérifiant le critère
  Pour les mettre tous à jour on ajoute un 3ème paramètre
  db. nom_collection.update ({critère: val} , {$set : {attribut:val2}} , {multi:true})

- db. nom_collection.update ({critère: val} , {$unset : {attribut:val2}} , {multi:true}) // pour supprimer un attribut

- db. nom_collection.insert({attribut1 : val1 ,tab:['a','b','c']}) // exemple d'insertion d'un tableau pour l'afficher on fait un find avec le critère de attribut1 : val1

- db. nom_collection.update({attribut1:val1}, {$push : {tab : 'd'}}) // ajouter un élément au tableau
  pour ajouter plusieurs éléments à la fois on utilise $pushAll

- pour supprimer un élément on met au lieu de $push {$pop : {tab:1}} //supprime le dernier élément de tab
  pour supprimer le premier on met -1 au lieu de 1
- db. nom_collection.update({critere1: val1}, {$addToSet : {tab : 'b'}}) //ajoute b si seulement s'il n'existe pas dans le tableau sinon il ne l'ajoute pas

- db. nom_collection.remove({critere1: val1}) // supprimer tous les documents vérifiant le critère
- db.oldname.renameCollection("newname") //renomer une collection
- db.nom_collection.drop() // supprimer une collection ou avec remove()
- db.dropDatabase() = supprimer la base où on est
- db.users.update({_id: x }, {$pull: {name_collection: {_id: y}}}
  effacer un sous-document d'id y du doc d'id x (dans un tableau de sous-documents)
- db.name_collection.getIndexes() = lister les indexes de la collection
- db. name_collection.createIndex({email: 1}, {unique: true}) // unique est un parameter facultatif
  L'index ci-dessus considérera une absence de valeur comme une valeur. Il n'y a donc qu'un email qui pourra être vide ensuite mongo considérera les autres email sans valeur comme dupliqués et refusera de les ajouter. Pour empêcher ce comportement, on peut utiliser l'option sparse.
  db.users.createIndex({email: 1}, {unique: true, sparse: true})
  il est possible d'utiliser l'index de type text
  db.users.createIndex({firstname: "text"})

  il ne peut y avoir qu'un seul index text par collection
  cependant, on peut les combiner
  db.users.createIndex({firstname: "text", lastname: "text"}) //un seul paramètre