

# Rapport : Gestion des Étudiants dans MongoDB

**Module :** Base de Données MongoDB

**Nom:** Aya ouazzani akkar hammoudi

## SOMMAIRE :

1. Créer une base de données et des collections dans MongoDB pour stocker les informations des étudiants.

2. Générer et importer des données avec Mockaroo pour simuler des étudiants et leurs cours.

3. Manipuler les données : Utiliser des requêtes pour rechercher, trier et filtrer des documents, et appliquer des opérations de mise à jour.

4. Manipuler des tableaux : Gérer des champs de type tableau dans MongoDB, notamment pour ajouter, mettre à jour, et supprimer des données dans des tableaux.

## 1. Création de la base et des collections :

The screenshot shows two screenshots of the Compass MongoDB interface. The top screenshot displays the 'gestion\_etudiants' database in the Compass sidebar under 'CONNECTIONS'. It lists three databases: admin, config, and local. The bottom screenshot shows the 'gestion\_etudiants' database selected. Under the 'gestion\_etudiants' connection, a 'gestion\_etudiants' collection is selected. The interface shows a query builder with 'Documents 0', a search bar, and a note: 'This collection has no data'. A green 'Import data' button is visible at the bottom.

## 2. Génération des données avec Mockaroo:

Sur Mockaroo (<https://mockaroo.com>) :

Exemple :

The screenshot shows the Mockaroo data generator interface. It lists several fields with their types and options:

- Field Name:** numero\_inscription, **Type:** Custom List, **Options:** E123, E124, E125
- Field Name:** nom, **Type:** Text
- Field Name:** prenom, **Type:** Text
- Field Name:** date\_naissance, **Type:** Datetime, **Options:** from: 01/01/2000, to: 12/31/2005, format: m/d/yyyy
- Field Name:** cours, **Type:** JSON Array, **Options:** min elements: 0, max elements: 5, blank: 0%

Buttons at the bottom include "+ ADD ANOTHER FIELD", "GENERATE FIELDS USING AI...", and "+ ADD ANOTHER FIELD".

### 3. Importation des données dans MongoDB :

The screenshot shows the MongoDB Compass interface with the following details:

- CONNECTIONS:** gestion\_etudiants (selected), admin, config, local
- Find Results:** 25 documents found, pages 1-25 of 81.
- Document Preview:** Three documents are shown in expanded form:

```
_id: ObjectId('69139b61c698413ab8808fb2')
numero_inscription : "E126"
nom : "Durand"
prenom : "Alice"
date_naissance : "2004-09-12"
> cours : Array (4)
```

```
_id: ObjectId('69139b61c698413ab8808fb3')
numero_inscription : "E127"
nom : "Dubois"
prenom : "Emma"
date_naissance : "2004-03-15"
> cours : Array (3)
```

```
_id: ObjectId('69139b61c698413ab8808fb4')
numero_inscription : "E128"
nom : "Petit"
```

- Document Editor:** A detailed view of a document with fields: \_id, numero\_inscription, nom, prenom, date\_naissance, and cours.

### 4. Requêtes de recherche et affichage :

Afficher tous les étudiants : db.etudiants.find().pretty()

```

> use gestion_etudiants
< switched to db gestion_etudiants
> db["etudiants"].find()
< [
  {
    _id: ObjectId('69139b61c698413ab8808faf'),
    numero_inscription: 'E123',
    nom: 'Bernard',
    prenom: 'Léa',
    date_naissance: '2000-05-07',
    cours: [
      {
        nom: 'Informatique',
        code: 'INF101',
        note: 16.5
      },
      {
        nom: 'Physique',
        code: 'PHY101',
        note: 19.2
      },
      {
        nom: 'Chimie',
        code: 'CHM101',
        note: 15.9
      }
    ]
  }
]

```

Afficher les étudiants inscrits en Mathématiques : db.etudiants.find({ "cours.nom": "Mathématiques" }).pretty()

```

Type "it" for more
> db.etudiants.find({ "cours.nom": "Mathématiques" }).pretty()
< [
  {
    _id: ObjectId('69139b61c698413ab8808fb0'),
    numero_inscription: 'E124',
    nom: 'Simon',
    prenom: 'Emma',
    date_naissance: '2000-06-27',
    cours: [
      {
        nom: 'Informatique',
        code: 'INF101',
        note: 16.5
      },
      {
        nom: 'Chimie',
        code: 'CHM101',
        note: 15.9
      },
      {
        nom: 'Mathématiques',
        code: 'MATH101',
        note: 12.4
      }
    ]
  }
]

```

```
>_MONGOSH
{
  {
    nom: 'Physique',
    code: 'PHY101',
    note: 19.2
  },
  {
    nom: 'Chimie',
    code: 'CHM101',
    note: 15.9
  },
  {
    nom: 'Informatique',
    code: 'INF101',
    note: 16.5
  },
  {
    nom: 'Mathématiques',
    code: 'MATH101',
    note: 12.4
  }
}
```

Trier par nom (ordre alphabétique) : db.etudiants.find().sort({ nom: 1 }).pretty()

```
Type "it" for more
>db.etudiants.find().sort({ nom: 1 }).pretty()
< {
  _id: ObjectId('69139b61c698413ab8809009'),
  numero_inscription: 'E123',
  nom: 'Bernard',
  prenom: 'Alice',
  date_naissance: '2000-05-07',
  cours: [
    {
      nom: 'Informatique',
      code: 'INF101',
      note: 16.5
    },
    {
      nom: 'Physique',
      code: 'PHY101',
      note: 19.2
    },
    {
      nom: 'Chimie',
      code: 'CHM101',
    }
  ]
}
```

```
_id: ObjectId('69139b61c698413ab8809009'),
numero_inscription: 'E213',
nom: 'Bernard',
prenom: 'Alice',
date_naissance: '2003-03-06',
cours: [
  {
    nom: 'Chimie',
    code: 'CHM101',
    note: 15.9
  },
  {
    nom: 'Informatique',
    code: 'INF101',
    note: 16.5
  },
  {
    nom: 'Mathématiques',
    code: 'MATH101',
    note: 12.4
  }
]
```

Filtrer étudiants ayant note > 15 en Mathématiques : db.etudiants.find({ "cours.nom": "Mathématiques", "cours.note": { \$gt: 15 } }).pretty()

```
Type "it" for more
> db.etudiants.find({
  "cours.nom": "Mathématiques",
  "cours.note": { $gt: 15 }
}).pretty()
< {
  "_id": ObjectId('69139b61c698413ab8808fb0'),
  "numero_inscription": 'E124',
  "nom": 'Simon',
  "prenom": 'Emma',
  "date_naissance": '2000-06-27',
  "cours": [
    {
      "nom": 'Informatique',
      "code": 'INF101',
      "note": 16.5
    },
    {
      "nom": 'Chimie',
      "code": 'CHM101',
      "note": 15.9
    },
    {
      "nom": "Physique",
      "code": "PHY101",
      "note": 19.2
    },
    {
      "nom": 'Chimie',
      "code": 'CHM101',
      "note": 15.9
    },
    {
      "nom": 'Informatique',
      "code": 'INF101',
      "note": 16.5
    },
    {
      "nom": "Mathématiques"
    }
  ]
}
```

## 5. Mise à jour des documents :

Mettre à jour la note de Mathématiques pour un étudiant spécifique :

```
gestion_etudiants> db.etudiants.updateOne(
  { numero_inscription: "E123", "cours.nom": "Mathématiques" },
  { $set: { "cours.$.note": 19 } }
)
```

```
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 0,
  modifiedCount: 0,
  upsertedCount: 0
}
```

Ajouter un nouveau cours : db.etudiants.updateOne( { numero\_inscription: "E124" }, { \$push: { cours: { nom: "Anglais", code: "ENG101", note: 15 } } })

```
> db.etudiants.updateOne(  
  { numero_inscription: "E124" },  
  { $push: { cours: { nom: "Anglais", code: "ENG101", note: 15 } } } )  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}
```

Ajouter plusieurs dates de présence sans duplication : db.etudiants.updateOne( { numero\_inscription: "E125" }, { \$addToSet: { presences: { \$each: [ISODate("2023-09-15"), ISODate("2023-09-16")]} } } )

```
> db.etudiants.updateOne(  
  { numero_inscription: "E125" },  
  { $addToSet: { presences: { $each: ["2023-09-15", "2023-09-16"] } } } )  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}
```

Supprimer le dernier cours du tableau cours : db.etudiants.updateOne( { numero\_inscription: "E125" }, { \$pop: { cours: 1 } } )

```
> db.etudiants.updateOne(  
  { numero_inscription: "E125" },  
  { $pop: { cours: 1 } }  
)  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 1,  
  modifiedCount: 1,  
  upsertedCount: 0  
}
```

## 6. Suppression de documents

Supprimer un étudiant par numéro : db.etudiants.deleteOne({ numero\_inscription: "E125" })

```
> db.etudiants.deleteOne({ numero_inscription: "E125" })  
< {  
  acknowledged: true,  
  deletedCount: 1  
}
```

Supprimer tous les étudiants nés avant 2001-01-01 : db.etudiants.deleteMany({ date\_naissance: { \$lt: new Date("2001-01-01") } })

```
> db.etudiants.deleteMany({ date_naissance: { $lt: "2001-01-01" } })  
< {  
  acknowledged: true,  
  deletedCount: 18  
}
```