Lab 4: Initiation à MongoDB

L'objectif de cet atelier est de :

- Créer un compte MongoDB Atlas et déployer un cluster.
- Se connecter à la base de données avec MongoDB Compass ou MongoDB Shell.
- Créer une collection et insérer des documents.
- Exécuter des requêtes pour récupérer des données.
- Mettre à jour et supprimer des documents.
- Importation des fichiers
- Exemple Map Reduce

MongoDB est un SGBD NoSQL orienté documents. Il est particulièrement apprécié pour sa capacité à passer en mode distribué pour répartir le stockage et les traitements de données
Aussi, MongoDB fournit une API en python et autres langages pour pouvoir manipuler programmatiquement les données de la base.

1. Création d'un compte et déploiement d'un cluster

Pour faciliter l'utilisation de MongoDB, nous allons exploiter MongoDB Atlas. Il s'agit d'un service cloud entièrement géré qui simplifie le déploiement, l'évolutivité et la gestion des bases de données MongoDB.

1. Allez sur MongoDB Atlas.
2. Inscrivez-vous ou connectez-vous (à l'aide d'un compte gmail par exemple)
3. Cliquez sur **"Créer un cluster"** (Choisissez un cluster **gratuit** si disponible).
4. Sélectionnez **"M0 Sandbox"** et choisissez un fournisseur de cloud et une région.
5. Configurez votre base de données (Nom du cluster par défaut ou personnalisé).
6. Cliquez sur **"Créer un cluster"** (cela peut prendre quelques minutes).
7. Créer un utilisateur avec un mot de passe (garder les deux pour un usage ultérieur)

MongoDB fonctionne en mode client/serveur :

- Le serveur est en attente
- Le client : peut être :
  - l'interpréteur de commande (shell) mongo
  - une application UI pour simplifier l'usage telle que compass, RoboMongo ou bien Studio3T.
  - Un programme développé
  - l'interface MongoDB Atlas

2. Connexion à MongoDB Atlas

- Une fois le cluster créé, allez dans "Database" > "Connect".
- Choisissez "MongoDB Compass" (interface graphique) ou "MongoDB Shell" (ligne de commande).

- Copiez l'URI de connexion et utilisez-le pour vous connecter. (n'oublier pas de copier votre mot de passe). **Exemple :**

```
mongodb+srv://yalami:<db_password>@cluster0.apxc2.mongodb.net/
```

3. Création d'une base de données et d'une collection

Dans **MongoDB Shell** :

- Pour se placer dans une base:

```
use <nombase>
```

- Créer une collection nommée users

```
db.createCollection("users")
```

- Lister les collections de la base

```
show collections
```

- Insérer un document JSON dans la collection users

```
db.users.insertOne( { name: "Mohamed", age: 20, city: "Casablanca" })
```

- Insérer maintenant plusieurs tuples dans la collection users

```
db.users.insertMany([
 { name: "Alice", age: 25, city: "Paris" },
 { name: "Bob", age: 30, city: "Lyon" },
 { name: "Charlie", age: 28, city: "Marseille" }
])
```

4. Requêtes pour récupérer des données

- L'objet (javascript) implicite, db, permet de soumettre des demandes d'exécution de certaines méthodes. Afficher tous les utilisateurs de la collection:

```
db.users.find()
```

- insérer un autre document avec les information de votre choix:

```

```

- Compter le nombre de documents dans la collection:

```
db.users.count()
```

- Récupérer un utilisateur spécifique :

```
db.users.findOne({ name: "Alice" })
```

- Filtrer les utilisateurs par âge (age>20) :

```
db.users.find({ age: { $gt: 20 } })
```

- Modifier l'âge d'un utilisateur :

```
db.users.updateOne({ name: "Alice" }, { $set: { age: 26 } })
```

- Supprimer un utilisateur :

```
db.users.deleteOne({ name: "Charlie" })
```

- supprimer une collection:

```
db.users.drop()
```

5. Importation de fichier dans une collection

L'utilitaire d'import de MongoDB prend en entrée un tableau JSON contenant la liste des objets à insérer. Dans notre cas, nous allons utiliser des données sur les transactions bancaires.

Cet ensemble de données contient 1 000 transactions bancaires synthétiques, incluant divers types de transactions tels que les virements, les retraits et les dépôts. Chaque enregistrement contient des attributs détaillés tels que l'identifiant de la transaction, les identifiants des comptes émetteur et récepteur, le montant de la transaction, l'horodatage, le statut de la transaction (réussite ou échec), l'indicateur de fraude, la géolocalisation, l'appareil utilisé, l'identifiant de la tranche réseau, la latence, la bande passante de la tranche et le code PIN associé. Cet ensemble de données est conçu pour l'analyse des données financières, la détection des fraudes et la surveillance des performances réseau, offrant un riche ensemble de fonctionnalités pour les applications de machine learning liées à l'analyse des transactions et à la détection des fraudes.

Pour ce faire

- Télécharger le fichier transactions.json.
- Après avoir créé la base de données appelée « **BankDB** » et la collection « **transactions** »,importer le fichier dans la collection :
  - **Méthode 1** : depuis l'interface graphique,
  - **Méthode 2** : en utilisant la commande `mongoimport`. Ceci suppose l'installation de mongoshell sur votre machine

```
mongoimport --uri
"mongodb+srv://<username>:<password>@cluster0.mongodb.net/BankDB" \
 --collection transactions --file transactions.json --jsonArray
```

- Dans **MongoDB Shell** ou **MongoDB Compass, c**onnectez-vous à la base de données BankDB et afficher son contenu

```
use BankDB myDatabase
db.transactions.find().pretty()
```

- affiche les collections de la base de données courante.
- Afficher les informations sur une transaction pour comprendre la structure
- Combien y a-t-il de transactions dans la base de données ?
- Trouver toutes les transactions échouées ("Transaction Status": "Failed")
- Vérifier les transactions suspectes ("Fraud Flag" : "True")

- Nombre total de transactions par statut

```
db.transactions.aggregate([
  { $group: { _id: "$Transaction Status", total: { $count: {} } } }
])
```

- Montant moyen des transactions par type

```
db.transactions.aggregate([
  { $group: { _id: "$Transaction Type", avgAmount: { $avg: "$Transaction Amount" } } }
])
```

- Top 5 des comptes qui envoient le plus d'argent

```
db.transactions.aggregate([
  { $group: { _id: "$Sender Account ID", totalSent: { $sum: "$Transaction Amount" } } },
{  $sort: { totalSent: -1 } },
  { $limit: 5 }
])
```

- Calculer le montant total des transactions par type

```
db.transactions.aggregate([
  { $group: { _id: "$Transaction Type", total: { $sum: "$Transaction Amount" } } }
])
```

- Vérifier la latence moyenne par "Network Slice ID"

```
db.transactions.aggregate([
  { $group: { _id: "$Network Slice ID", avgLatency: { $avg: "$Latency (ms)" } } }
])
```

- Transactions échouées par appareil utilisé

```
db.transactions.aggregate([
  { $match: { "Transaction Status": "Failed" } },
  { $group: { _id: "$Device Used", totalFailed: { $count: {} } } }
])
```

- Trouver les transactions suspectes (fraude et montant élevé > 1000)

```
db.transactions.find({ "Fraud Flag": "True", "Transaction Amount": { $gt: 1000 } })
```

- Combien transactions ont un montant entre 100 et 200 ans ?

On veut maintenant mettre en place plusieurs **indicateurs clés** (KPI) pour surveiller l'activité financière, détecter les fraudes et analyser la performance des transactions.

- Afficher nombre total des transactions

- Afficher Montant total des transactions

```
db.transactions.aggregate([{ $group: { _id: null, totalAmount: { $sum: "$Transaction Amount" } } }])
```

- afficher **Taux d'échec des transactions (%)**

```
db.transactions.aggregate([
  { $group: { _id: "$Transaction Status", count: { $count: {} } } }
])
```

- Afficher le Nombre de transactions frauduleuses

```
db.transactions.countDocuments({ "Fraud Flag": "True" })
```

Pour plus d'informations
http://docs.mongodb.org/manual/reference/method/db.collection.find/

6. Manipuler MapReduce sur la collection transactions

Une fois connecté à MongoDB Atlas, vous pouvez utiliser MapReduce pour effectuer des analyses. L'exemple suivant montre comment utiliser MapReduce pour calculer le montant total des transactions par type de transaction (par exemple, "Dépôt", "Transfert", etc.).

Pour ce faire :

- Créer un nouveau notebook sur colab
- Exécutez la cellule suivante dans **Google Colab** pour installer **PyMongo** :

```
!pip install pymongo
```

- Ensuite, Se connecter à MongoDB Atlas

```
from pymongo import MongoClient
# Remplacez par votre URI MongoDB Atlas
uri = "mongodb+srv://<username>:<password>@cluster0.mongodb.net/bankdb?retryWrites=true&w=majority"
client = MongoClient(uri)
# Accéder à la base de données
db = client["bankdb"]
transactions_collection = db["transactions"]
# Vérification de la connexion
print("Connexion à MongoDB Atlas réussie!")
```

Etant donné que map_reduce() est obsolète, utilisez le framework d'agrégation de MongoDB pour additionner les montants des transactions par type de transaction

```
#creer un pipeline d' Aggregation  pour calculer le montant total transaction par type
pipeline = [
   {"$group": {"_id": "$Transaction Type", "totalAmount": {"$sum": "$Transaction Amount"}}}
]
# executer la requete d'agrégation
result = transactions_collection.aggregate(pipeline)
# afficher les resultats
```

```
for doc in result:
    print(f"Transaction Type: {doc['_id']}, Total Amount: {doc['totalAmount']}")
```

- Réaliser d'autres opération d'agrégation de votre choix

7. Dashboard avec MongoDB Atlas

MongoDB Atlas propose un outil appelé **MongoDB Charts**, qui permet de **visualiser et d'analyser les données** directement depuis votre base de données **sans besoin d'exportation**. Cet outil est idéal pour créer des **tableaux de bord interactifs** basés sur vos données en temps réel

1 Activer MongoDB Charts

1. Connectez-vous à **MongoDB Atlas** (https://www.mongodb.com/atlas).
2. Sélectionnez votre cluster.
3. Cliquez sur **"Charts"** dans le menu de gauche.
4. Créez un **nouveau tableau de bord**.

2 Connecter une Source de Données

1. Cliquez sur **"Add Data Source"**.
2. Sélectionnez votre base de données **(`bankdb`)** et votre collection **(`transactions`)**.
3. Configurez les autorisations si nécessaire.

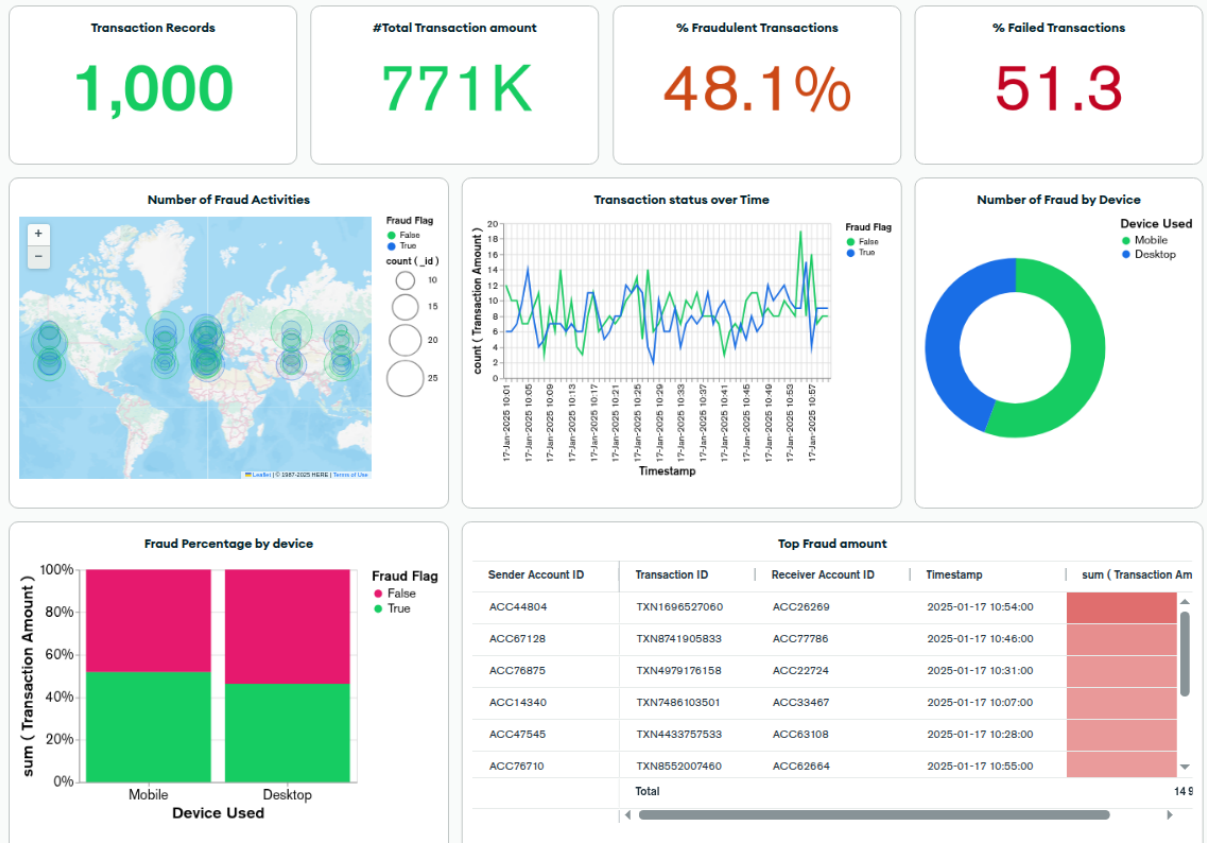
3 Créer des Visualisations

1. **Sélectionnez un type de graphique** (barres, lignes, camembert, etc.).
2. **Définissez les axes** :
   - **X-axis** : `"Transaction Type"`
   - **Y-axis** : `"Transaction Amount"` (somme des montants)
3. **Appliquez des filtres si nécessaire** (ex. afficher uniquement les transactions réussies).
4. **Enregistrez et ajoutez le graphique** à votre tableau de bord.

4 Personnaliser et Partager

- Ajoutez plusieurs graphiques à votre tableau de bord.
- Personnalisez les couleurs et les titres.
- **Partagez le tableau** avec d'autres utilisateurs via un lien.

Voici un exemple :


Transaction Records
1,000
#Total Transaction amount
771K
% Fraudulent Transactions
48.1%
% Failed Transactions
51.3
Number of Fraud Activities
Fraud Flag
False
True
count (_id)
10
15
20
25
Transaction status over Time
Fraud Flag
False
True
count (Transaction Amount)
Timestamp
Number of Fraud by Device
Device Used
Mobile
Desktop
Fraud Percentage by device
sum (Transaction Amount)
100%
80%
60%
40%
20%
0%
Mobile
Desktop
Device Used
Fraud Flag
False
True
Top Fraud amount
Sender Account ID
Transaction ID
Receiver Account ID
Timestamp
sum (Transaction Am
ACC44804 TXN1696527060 ACC26269 2025-01-17 10:54:00
ACC67128 TXN8741905833 ACC77786 2025-01-17 10:46:00
ACC76875 TXN4979176158 ACC22724 2025-01-17 10:31:00
ACC14340 TXN7486103501 ACC33467 2025-01-17 10:07:00
ACC47545 TXN4433757533 ACC63108 2025-01-17 10:28:00
ACC76710 TXN8552007460 ACC62664 2025-01-17 10:55:00
Total