

TP4_instalation-test_Gescom.Employes_Test-Paris

3 Installer MongoDB et faire quelques tests

3.1 Installer MongoDB

MongoDB est disponible pour Windows, Mac OS, Linux et Solaris. Rendez-vous sur la page officielle de téléchargement pour télécharger la version qui correspond à votre système d'exploitation. Dans mon cas, il s'agit d'un Windows 64 bits.

Pour démarrer MongoDB, exécutez le fichier mongod.exe (Windows) ou mongod (Unix), situé dans le dossier bin du dossier dans lequel MongoDB est installé. Lors de son lancement, MongoDB recherchera le dossier /data/bd à la racine de votre disque. Si ce dossier n'existe pas, créez-le.

Vous pouvez évidemment créer le dossier /data/db ailleurs; dans ce cas, il faudra spécifier son emplacement en utilisant le paramètre - -dbpath.

[[information]] | Pour plus de facilité, sous Windows, je vous conseille de vous créer un petit script .bat. Créez un nouveau fichier texte (.txt) avec le Bloc-Notes. Dedans, écrivez le

chemin vers mongod.exe, suivi du paramètre -- dbpath, comme ceci : | | bash | TITLE

MongoDB | "F:\Program Files\MongoDB\bin\mongod.exe" -- dbpath "F: \Program

Files\MongoDB\bin\data" | | Enregistrez le fichier, et changez l'extension en .bat. Un

double clic sur le fichier .bat lancera la commande, et donc le démarrage de MongoDB.

Lancez donc MongoDB, et voici ce que vous devriez obtenir. Si la dernière ligne affiche waiting for connections, c'est bon, MongoDB fonctionne et attend vos consignes!

3.2.1 Lister et choisir la base de données

Par défaut, MongoDB se connecte à la base de données appelée test. La commande show dbs permet d'afficher toutes les bases de données gérées par MongoDB. Voici ce que j'obtiens :

```
>_MONGOSH
> show dbs
< admin   40.00 KiB
  config  72.00 KiB
  local   72.00 KiB
```

Maintenant que nous connaissons toutes les bases de données accessibles, changeons de base, avec la commande use <nom-de-la-base>:

```
> use local
< switched to db local
```

[[question]] | Mais comment créer une base de données ?

Pour créer une base de données, par exemple la base tutoriel, il suffit de la sélectionner (même si elle n'existe pas) avec use, puis d'y insérer des données. Lors de l'insertion, la base sera créée.

Voyons donc comment insérer des données, mais auparavant, sélectionnez la base de données tutoriel :

```
> use tutoriel
< switched to db tutoriel
```

3.2.2 Insérer des documents

Nous sommes connectés à la base tutoriel, donc toutes les opérations que nous allons réaliser ici porteront sur cette base de données-là. Gardez cela en tête !

Pour insérer un document, rien de plus simple :

```
db.personnages.insert( { name : "Gordon Freeman", game : "Half-Life" } );
```

Cette simple commande insérera le document { name: "Gordon Freeman", game: "Half-

Life" } au sein de la collection personnages. Cette collection n'existe pas, mais sera créée automatiquement.

```
> db.personnage.insert({name : "Gordon Freeman", game : "Half-Life"});
< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("655cbcd68d9dea331d86a579")
  }
}
```

3.2.3 Faire des recherches

Pour afficher tous les documents d'une collection, il suffit de faire :

```
> db.personnages.find()
```

Cette commande affichera tous les documents de la collection personnages.

```
> db.personnage.find()
< {
  _id: ObjectId("655cbef08d9dea331d86a57a"),
  name: 'Gordon Freeman',
  game: 'Half-Life'
}
```

Des critères peuvent être spécifiés en paramètre de la méthode find (). Ces paramètres prennent la forme d'un objet, dont les propriétés à analyser correspondent à celles des documents :

```
> db.personnages.find( { name : "Gordon Freeman" } )
```

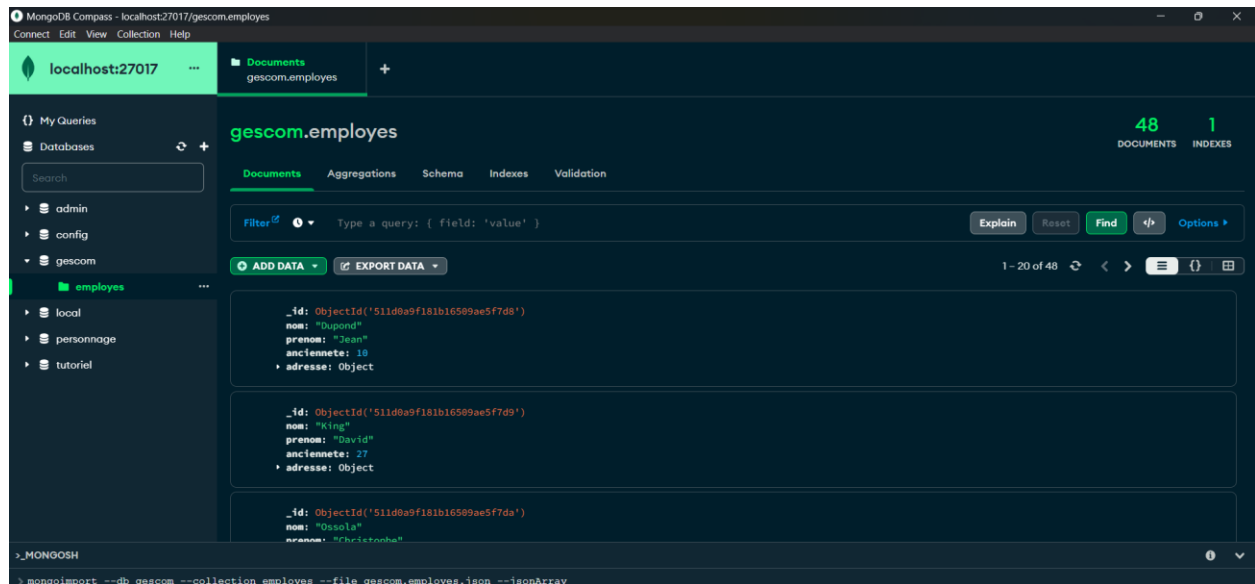
Cette commande trouvera donc le document dont la clef name vaut Gordon Freeman.

```
> db.personnage.find({name : "Gordon Freeman"})  
< {  
  _id: ObjectId("655cbef08d9dea331d86a57a"),  
  name: 'Gordon Freeman',  
  game: 'Half-Life'  
}  
personnage >
```

TP MongoDB

Dans MongoDB Compass:

- créer une nouvelle base de données « gescom »
- créer une nouvelles Collection « employes »
- importer le fichier « gescom.employes.json »

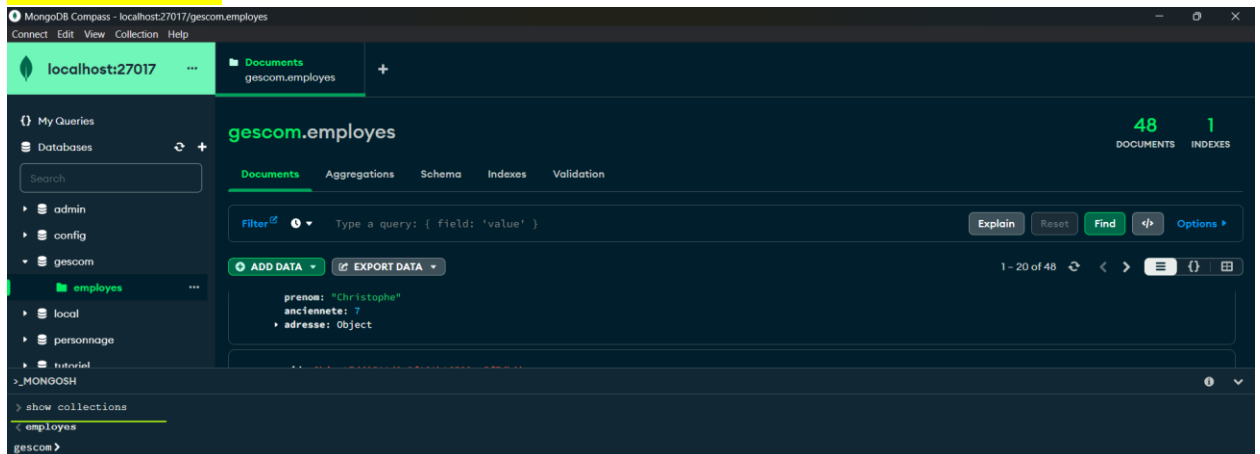


Exercice 1

Ecrire les requêtes MongoDB qui permettent de :

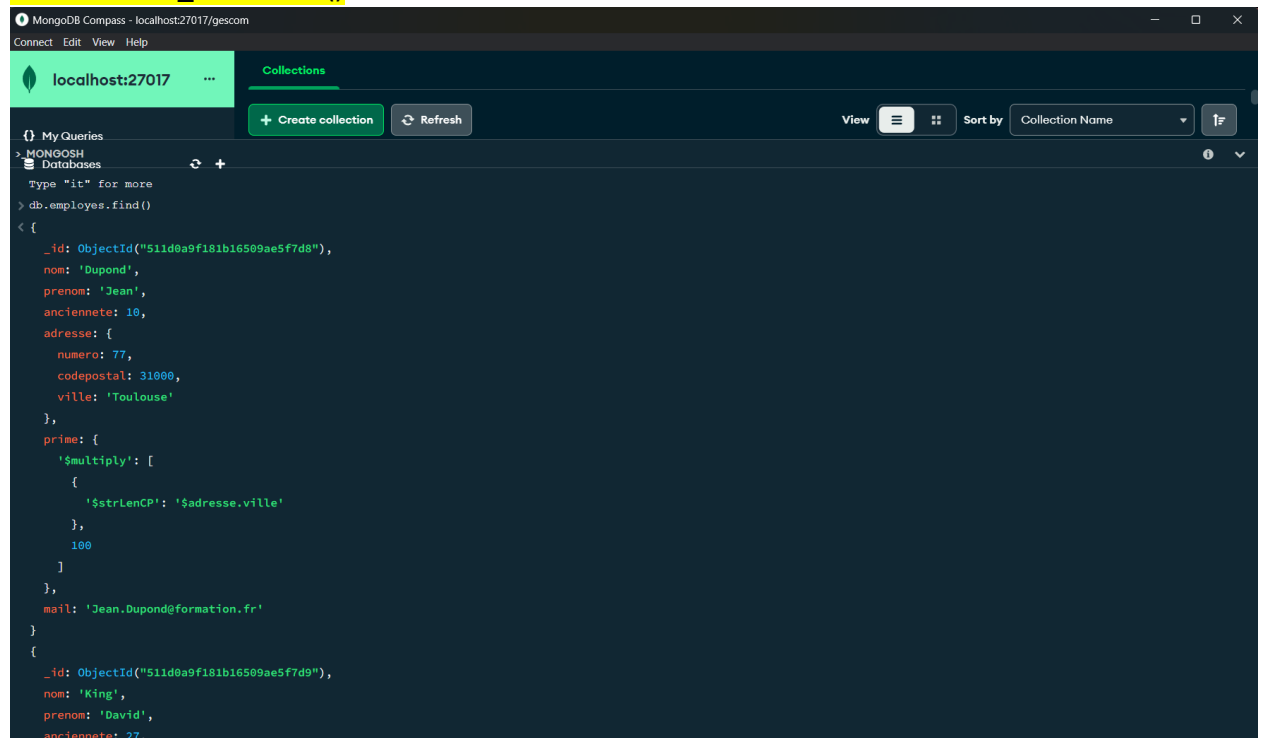
1. Afficher toutes les collections de la base

show collections



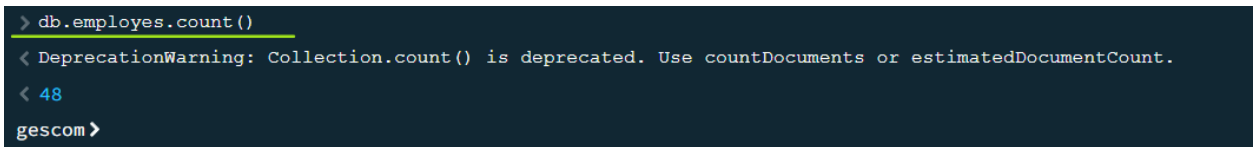
- Afficher tous les documents de la base :

`db.collection_name.find()`



- Compter le nombre de documents de la collection employés

`db.employees.count()`



- Insérer de deux manières différentes deux employés avec les champs nom, prénom et soit prime soit ancienneté

`> // Insertion 1`

`db.employees.insert({`

`nom: "Doe",`

`prenom: "John",`

`prime: 5000`

`});`

`// Insertion 2`

```
db.employees.insert ([
  { nom: "Smith", prenom: "Jane", anciennete: 3 },
  { nom: "Johnson", prenom: "Bob", anciennete: 7 }
]);
```

```
> // Insertion 1
db.employees.insert({
  nom: "Doe",
  prenom: "John",
  prime: 5000
});

// Insertion 2
db.employees.insert([
  { nom: "Smith", prenom: "Jane", anciennete: 3 },
  { nom: "Johnson", prenom: "Bob", anciennete: 7 }
]);

< DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6565faba9238bbe9fe4ba7bc"),
    '1': ObjectId("6565faba9238bbe9fe4ba7bd")
  }
}
gescom>
```

- Afficher la liste des employés dont le prénom est David

```
db.employees.find({ prenom: "David" })
```

```
>_MONGODB
{
}
}
> db.employees.find({ prenom: "David" })
< {
  _id: ObjectId("511d0a9f181b16509ae5f7d9"),
  nom: 'King',
  prenom: 'David',
  anciennete: 27,
  adresse: {
    numero: 78,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
{
  _id: ObjectId("511d0a9f181b16509ae5f7df"),
  nom: 'Ving',
  prenom: 'David',
  anciennete: 17,
  adresse: {
    numero: 28,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
}
```

- Afficher la liste des employés dont le prénom commence ou se termine par D

```
db.employees.find ({
  $or: [
```

```
{ prenom: /^D/ }, // commence par D
{ prenom: /D$/ } // se termine par D
}}
```

```
>_MONGOOSH
> db.employees.find({
  $or: [
    { prenom: /^D/ }, // commence par D
    { prenom: /D$/ } // se termine par D
  ]
})
< [
  {
    _id: ObjectId("511d0a9f181b16509ae5f7d9"),
    nom: 'King',
    prenom: 'David',
    anciennete: 27,
    adresse: {
      numero: 78,
      codepostal: 33000,
      ville: 'Bordeaux'
    }
  },
  {
    _id: ObjectId("511d0a9f181b16509ae5f7df"),
    nom: 'Ving',
    prenom: 'David',
    anciennete: 17,
    adresse: {
      numero: 28,

```

- Afficher la liste des personnes dont le prénom commence par D et contient exactement 5 lettres

```
db.employees.find ({prenom : /^D.{3}$/})
```

```
> db.employees.find({
  prenom: /^D.{3}$/
})
<
```

- Afficher la liste des personnes dont le prénom commence et se termine par une voyelle

```
db.employees.find({ prenom: /^[aeiou].*[aeiou]$/i })
```

```
>_MONGOOSH
<
> db.employees.find({
  prenom: /^[aeiouAEIOU].*[aeiouAEIOU]$/i
})
< [
  {
    _id: ObjectId("511d0aa0181b16509ae5f801"),
    nom: 'Baron',
    prenom: 'Elodie',
    anciennete: 16,
    adresse: {
      numero: 9,
      codepostal: 95000,
      ville: 'Foix'
    },
    tel: 96934634,
    prime: 1500
  },
  {
    _id: ObjectId("511d0aa0181b16509ae5f805"),
    nom: 'Batini',
    prenom: 'Orlando',
    anciennete: 2,
    adresse: {
      numero: 27,
      ville: 'Les Bains'
    }
  }
]
```


- Afficher la liste des personnes dont le prénom commence et se termine par une même lettre

```
db.employees.find({
  $where: "this.prenom.charAt(0) == this.prenom.charAt(this.prenom.length-1) "
})
```

```
> db.employees.find({
  $where: "this.prenom.charAt(0) == this.prenom.charAt(this.prenom.length-1) "
})
<
gescom>
```

- Afficher les nom et prénom de chaque employé ayant une ancienneté > 10

```
db.employees.find({ anciennete: { $gt: 10 } }, { nom: 1, prenom: 1, _id: 0 })
```

```
>_MONGOSH
> db.employees.find({
  anciennete: { $gt: 10 }
}, { nom: 1, prenom: 1, _id: 0 })
< [
  {
    nom: 'King',
    prenom: 'David'
  },
  {
    nom: 'Rupont',
    prenom: 'Jean'
  },
  {
    nom: 'Ving',
    prenom: 'David'
  },
  {
    nom: 'Bass',
    prenom: 'Vincent'
  },
  {
    nom: 'Kingaba',
    prenom: 'David'
  }
]
```

- Afficher les nom et adresse complète des employés ayant un attribut rue dans l'objet adresse

```
db.employees.find({ "adresse.rue": { $exists: true } }, { nom: 1, adresse: 1, _id: 0 })
```

```
>_MONGOSH
> db.employees.find({ "adresse.rue": { $exists: true } }, { nom: 1, adresse: 1, _id: 0 })
< [
  {
    nom: 'Monnin',
    adresse: {
      numero: 88,
      rue: 'General Leclerc',
      codepostal: 31000,
      ville: 'Toulouse'
    }
  },
  {
    nom: 'Motin',
    adresse: {
      numero: 67,
      rue: 'Jean Moulin',
      codepostal: 31000,
      ville: 'Toulouse'
    }
  },
  {
    nom: 'Mani',
    adresse: {
      numero: 47,
      rue: 'Lavoisier',

```

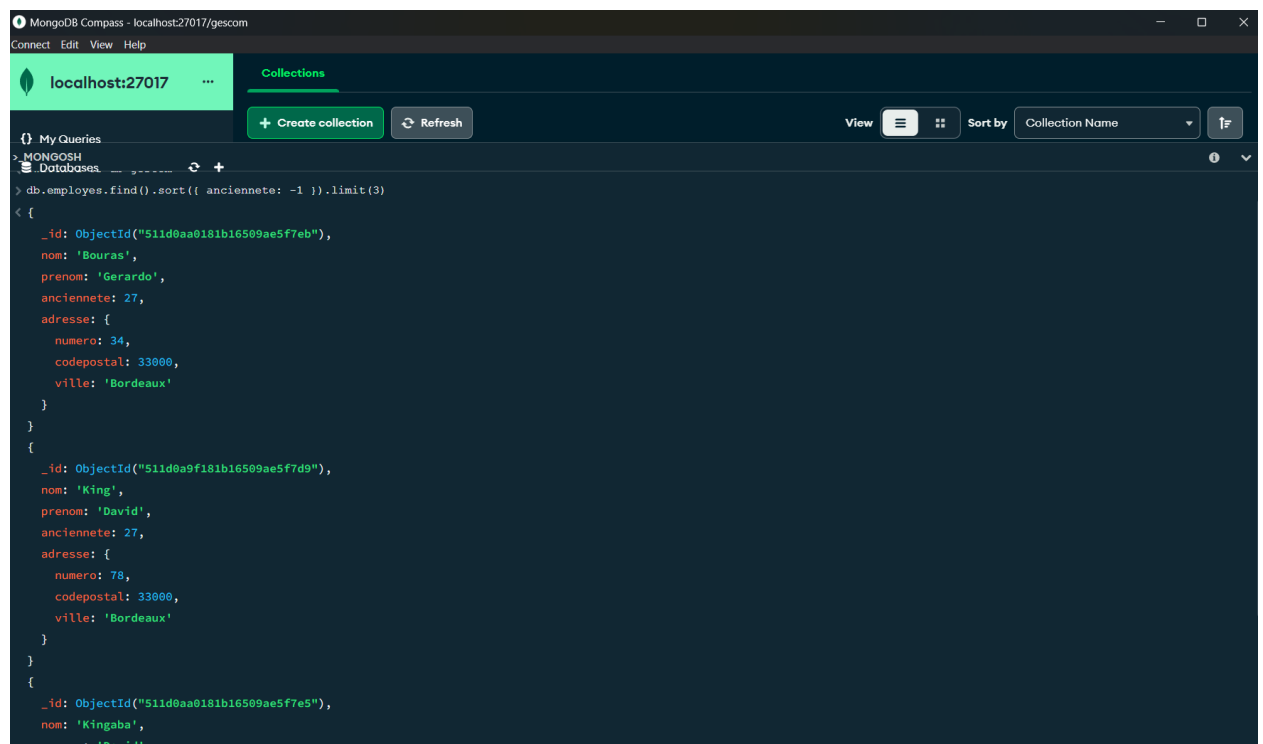
12. Incrémenter de 200 la prime des employés ayant déjà le champ prime

```
db.employees.updateMany({ prime: { $exists: true } }, { $inc: { prime: 200 } })
```

```
> db.employees.updateMany({ prime: { $exists: true } }, { $inc: { prime: 200 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 9,
  modifiedCount: 9,
  upsertedCount: 0
}
```

13. afficher les trois premières personnes ayant la plus grande valeur d'ancienneté

```
db.employees.find().sort({ anciennete: -1 }).limit(3)
```



```
MongoDB Compass - localhost:27017/gescom
Connect Edit View Help

localhost:27017 Collections
+ Create collection Refresh

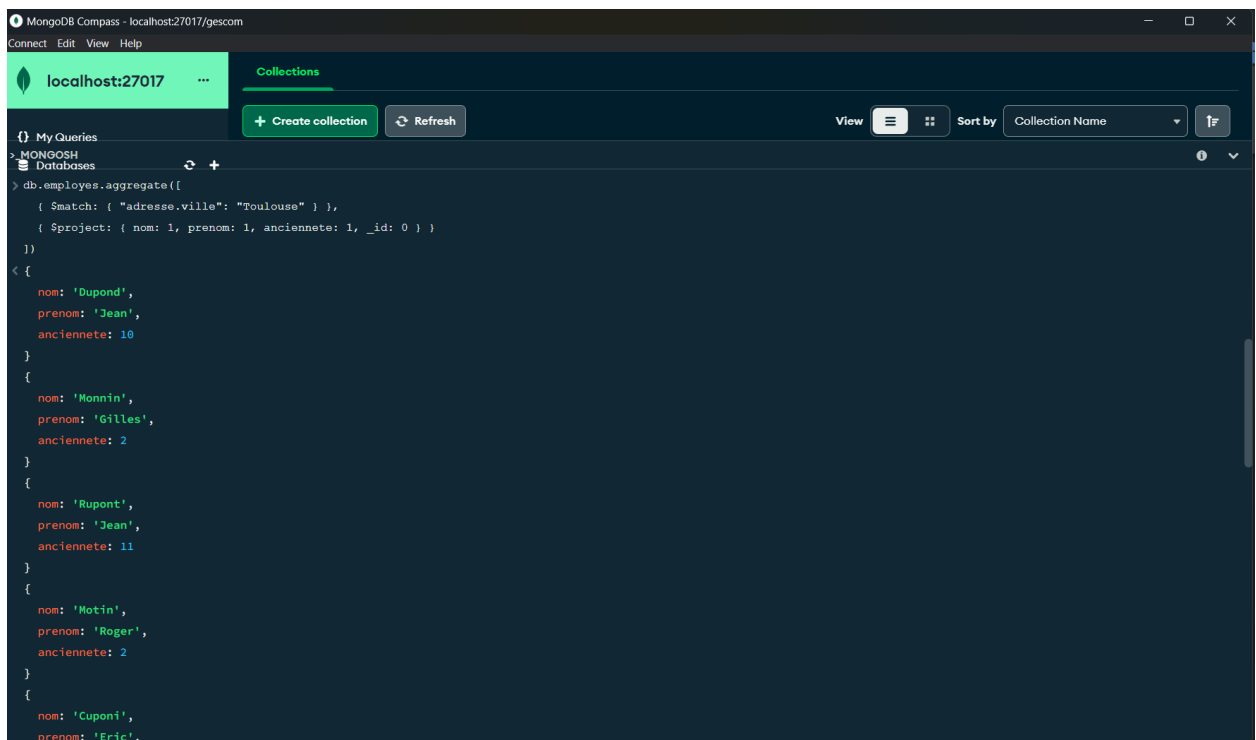
My Queries
> MONGODB
> Databases
> db.employees.find().sort({ anciennete: -1 }).limit(3)

{
  "_id": ObjectId("511d0aa0181b16509ae5f7eb"),
  "nom": "Bouras",
  "prenom": "Gerardo",
  "anciennete": 27,
  "adresse": {
    "numero": 34,
    "codepostal": 33000,
    "ville": "Bordeaux"
  }
}
{
  "_id": ObjectId("511d0a9f181b16509ae5f7d9"),
  "nom": "King",
  "prenom": "David",
  "anciennete": 27,
  "adresse": {
    "numero": 78,
    "codepostal": 33000,
    "ville": "Bordeaux"
  }
}
{
  "_id": ObjectId("511d0aa0181b16509ae5f7e5"),
  "nom": "Kingaba",
  "prenom": "David",
  "anciennete": 27,
  "adresse": {
    "numero": 12,
    "codepostal": 33000,
    "ville": "Bordeaux"
  }
}
```

14. regrouper les personnes dont la ville de résidence est Toulouse (afficher nom, prénom et ancienneté)

```
db.employees.aggregate([
  { $match: { "adresse.ville": "Toulouse" } },
  { $project: { nom: 1, prenom: 1, anciennete: 1, _id: 0 } }
])
```

NOUR ROUIS ING2 INFO



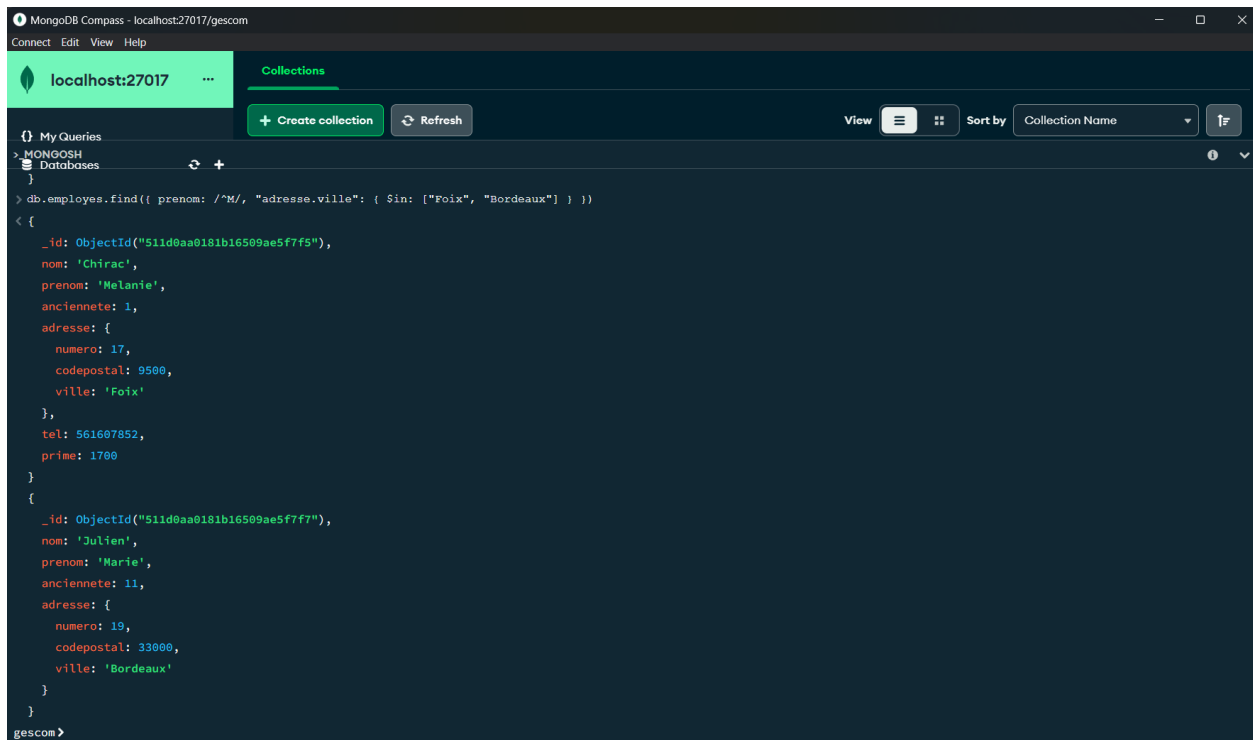
The screenshot shows the MongoDB Compass interface. The 'My Queries' tab is active, displaying an aggregate query for the 'employees' collection. The query filters for employees in 'Toulouse' and projects specific fields. The result shows five documents with names like 'Dupond', 'Monnin', 'Rupont', 'Motin', and 'Cupont'.

```
> db.employees.aggregate([
  { $match: { "adresse.ville": "Toulouse" } },
  { $project: { nom: 1, prenom: 1, anciennete: 1, _id: 0 } }
])
```

```
< {
  nom: 'Dupond',
  prenom: 'Jean',
  anciennete: 10
}
{
  nom: 'Monnin',
  prenom: 'Gilles',
  anciennete: 2
}
{
  nom: 'Rupont',
  prenom: 'Jean',
  anciennete: 11
}
{
  nom: 'Motin',
  prenom: 'Roger',
  anciennete: 2
}
{
  nom: 'Cupont',
  prenom: 'Eric',
  anciennete: 1
}
```

15. afficher les personnes dont le prénom commence par M et la ville de résidence est soit Foix soit Bordeaux

```
db.employees.find({ prenom: /^M/, "adresse.ville": { $in: ["Foix", "Bordeaux"] } })
```



The screenshot shows the MongoDB Compass interface with a 'find' query on the 'employees' collection. The query filters for employees whose first name starts with 'M' and whose city is either 'Foix' or 'Bordeaux'. The result shows two documents: one for 'Chirac' (Melanie) in Foix and one for 'Julien' (Marie) in Bordeaux.

```
> db.employees.find({ prenom: /^M/, "adresse.ville": { $in: ["Foix", "Bordeaux"] } })
```

```
< {
  _id: ObjectId("511d0aa0181b16509ae5f7f5"),
  nom: 'Chirac',
  prenom: 'Melanie',
  anciennete: 1,
  adresse: {
    numero: 17,
    codepostal: 9500,
    ville: 'Foix'
  },
  tel: 561607852,
  prime: 1700
}
{
  _id: ObjectId("511d0aa0181b16509ae5f7f7"),
  nom: 'Julien',
  prenom: 'Marie',
  anciennete: 11,
  adresse: {
    numero: 19,
    codepostal: 33000,
    ville: 'Bordeaux'
  }
}
```

16. mettre `à jour l'adresse de Dominique Mani : nouvelle adresse ({ numero : 20, ville : 'Marseille',codepostal : '13015' }). Attention, il n'y aura plus d'attribut rue dans adresse

```
db.employees.update({ nom: "Mani", prenom: "Dominique" }, { $set: { adresse: { numero: 20, ville: "Marseille", codepostal: "13015" } } })
```

```
> db.employees.update({ nom: "Mani", prenom: "Dominique" }, { $set: { adresse: { numero: 20, ville: "Marseille", codepostal: "13015" } } })
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
gescom>
```

17. attribuer une prime de 1 500 `à tous les employés n'ayant pas de prime et dont la ville de résidence est différente de Toulouse, Bordeaux et Paris. 2

```
db.employees.update({ prime: { $exists: false }, "adresse.ville": { $nin: ["Toulouse", "Bordeaux", "Paris"] } }, { $set: { prime: 1500 } }, { multi: true })
```

```
> db.employees.update({ prime: { $exists: false }, "adresse.ville": { $nin: ["Toulouse", "Bordeaux", "Paris"] } }, { $set: { prime: 1500 } }, { multi: true })
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 11,
  modifiedCount: 11,
  upsertedCount: 0
}
gescom>
```

18. remplacer le champ tel, pour les documents ayant un champ tel), par un tableau nommé téléphone contenant la valeur du champ tel (le champ tel est `à supprimer)

```
db.employees.find({ tel: { $exists: true } }).forEach(function (doc) {
  db.employees.update({ _id: doc._id }, { $set: { telephone: [doc.tel] }, $unset: { tel: 1 } });
});
```

```
> db.employees.find({ tel: { $exists: true } }).forEach(function (doc) {
  db.employees.update({ _id: doc._id }, { $set: { telephone: [doc.tel] }, $unset: { tel: 1 } });
});
gescom>
```

19. créer un champ prime pour les documents qui n'en disposent pas et de l'affecter `à 100 * nombre

de caractère du nom de la ville

```
db.employees.update({ prime: { $exists: false } }, { $set: { prime: { $multiply: [{ $strLenCP: "$adresse.ville" }, 100] } } }, { multi: true })
```

```
> db.employees.update({ prime: { $exists: false } }, { $set: { prime: { $multiply: [{ $strLenCP: "$adresse.ville" }, 100] } } }, { multi: true })
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 31,
  modifiedCount: 31,
  upsertedCount: 0
}
```

20. créer un champ mail dont la valeur est égale soit `a nom.prénom@formation.fr pour les employés

ne disposant pas d'un champ téléphone, soit `a prenom.nom@formation.fr (nom et prénom sont

`a remplacer par les vraies valeurs de chaque employé)

```
db.employees.find().forEach(function (doc) {
```

```
  var email = doc.telephone ? doc.nom + "." + doc.prenom + "@formation.fr" : doc.prenom + "." + doc.nom + "@formation.fr";
```

```
  db.employees.update({ _id: doc._id }, { $set: { mail: email } });
```

```
});
```

```
> db.employees.find().forEach(function (doc) {
  var email = doc.telephone ? doc.nom + "." + doc.prenom + "@formation.fr" : doc.prenom + "." + doc.nom + "@formation.fr";
  db.employees.update({ _id: doc._id }, { $set: { mail: email } });
});
```

21. calculer et afficher la somme de l'ancienneté pour les employés disposant du même prénom

```
db.employees.aggregate([
```

```
  { $group: { _id: "$prenom", totalAnciennete: { $sum: "$anciennete" } } }
```

```
]);
```

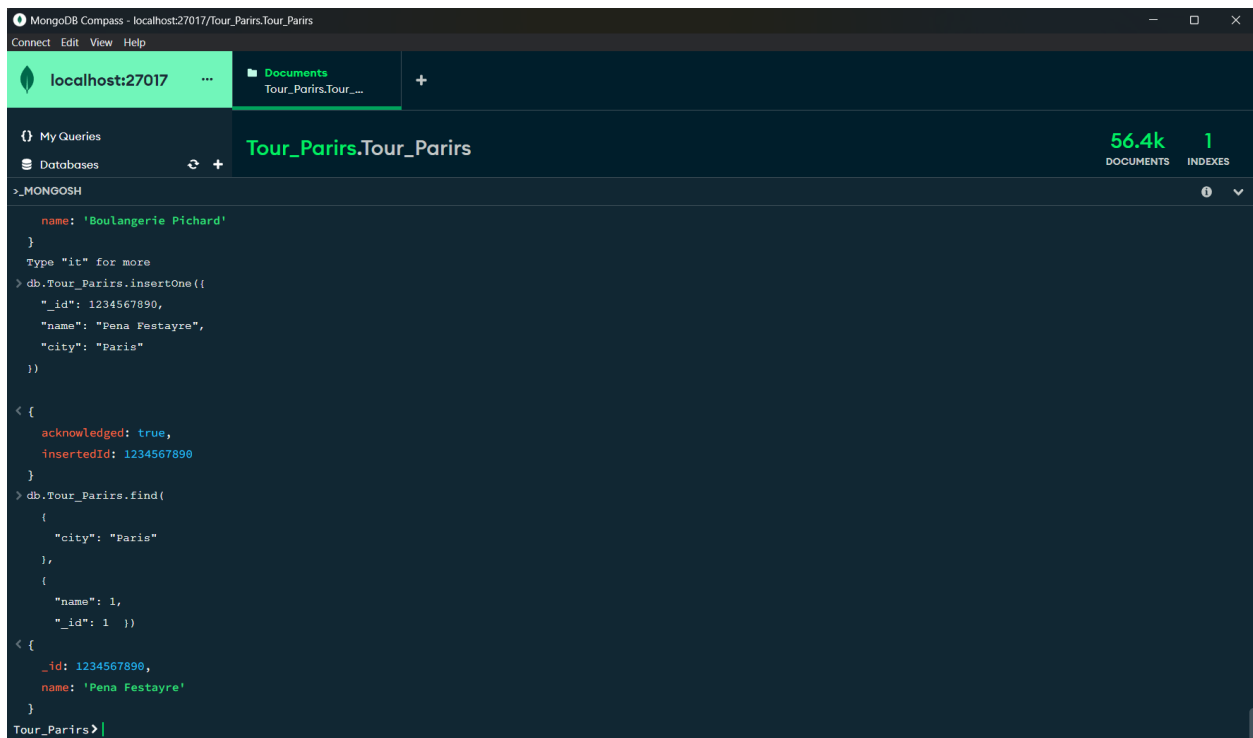
```
> MONGOSH
Databases
> db.employees.aggregate([
  { $group: { _id: "$prenom", totalAnciennete: { $sum: "$anciennete" } } }
])
< {
  _id: 'Sylvie',
  totalAnciennete: 5
}
{
  _id: 'Roland',
  totalAnciennete: 17
}
{
  _id: 'James',
  totalAnciennete: 5
}
{
  _id: 'John',
  totalAnciennete: 0
}
{
  _id: 'Marc',
  totalAnciennete: 7
}
{
  _id: 'Bassil',
  totalAnciennete: 7
}
}
```

TEST_Tour Paris :

1. Insérer un nouveau lieu ayant la description suivante : _id: 1234567890, name : Pena Festayre et city: Paris ;

```
db.Tour_Paris.insertOne({
  "_id": 1234567890,
  "name": "Pena Festayre",
  "city": "Paris"
})
```

NOUR ROUIS ING2 INFO



2. Donner le nom des lieux dont la catégorie est "accommodation" ;

```
db.Tour_Paris.find(
```

```
{
```

```
  "category": "accommodation"
```

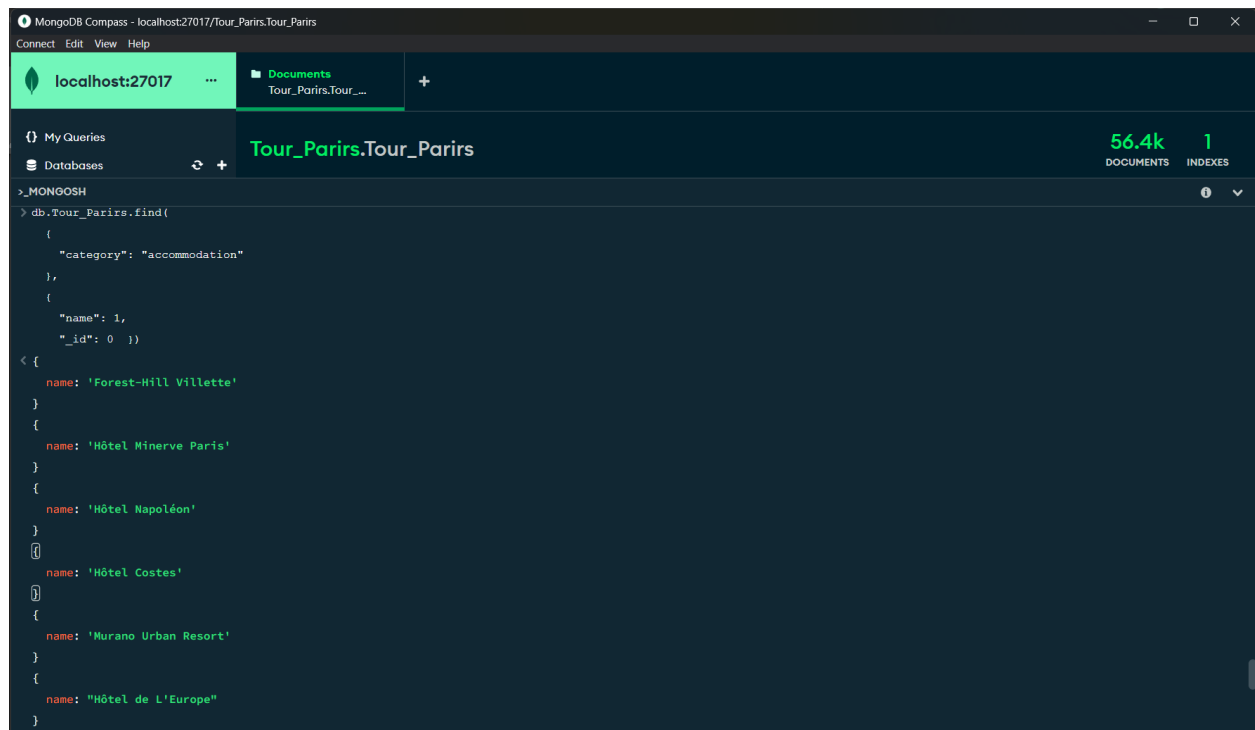
```
},
```

```
{
```

```
  "name": 1,
```

```
  "_id": 0 })
```

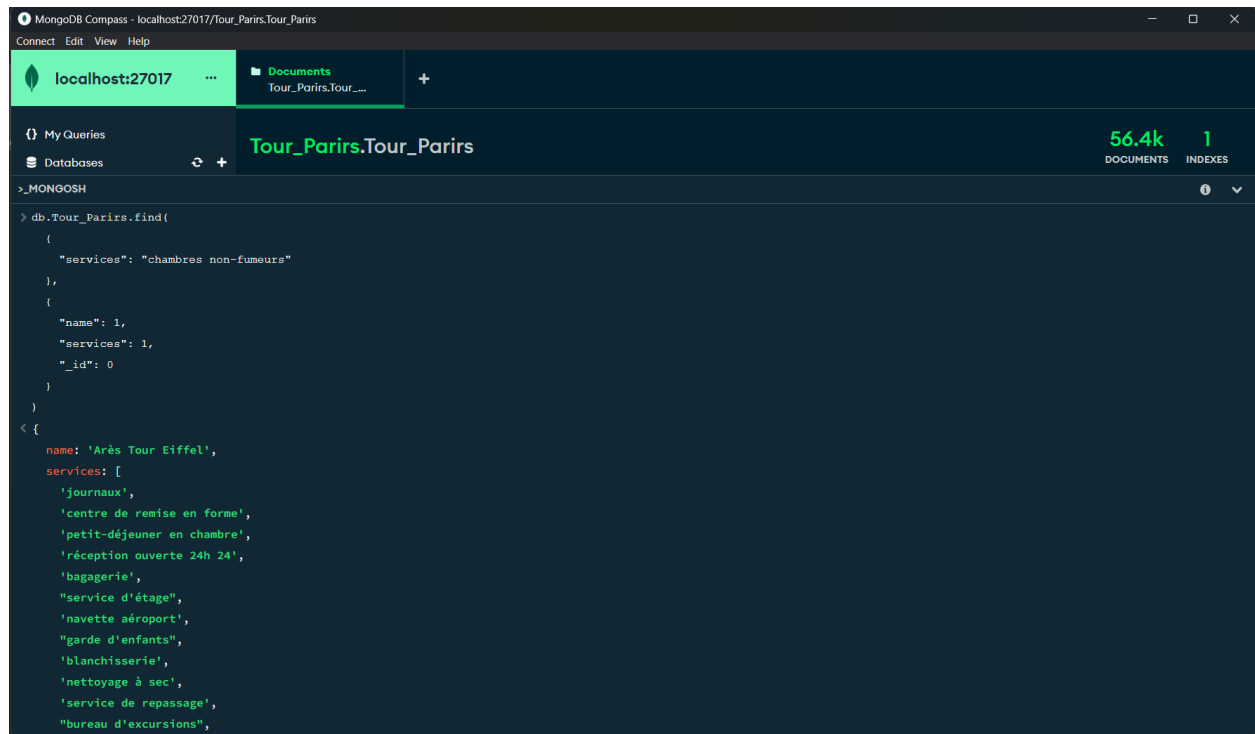
NOUR ROUIS ING2 INFO



4. Nom et services des lieux ayant un service "chambres non-fumeurs".

```
db.Tour_Paris.find(
{
  "services": "chambres non-fumeurs"
},
{
  "name": 1,
  "services": 1,
  "_id": 0
})
```


NOUR ROUIS ING2 INFO

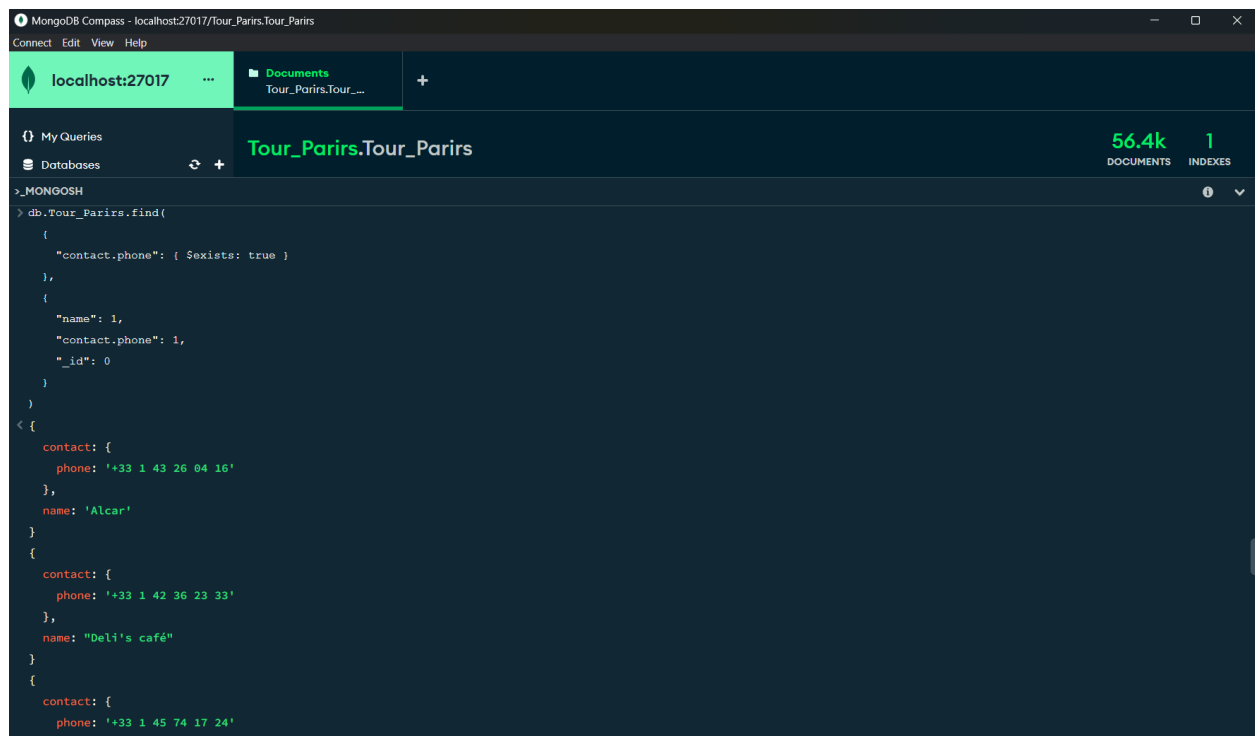


The screenshot shows the MongoDB Compass interface. The top bar indicates the connection to 'localhost:27017/Tour_Parirs.Tour_Parirs'. The left sidebar shows 'My Queries' and 'Databases'. The main panel displays the 'Documents' tab for the 'Tour_Parirs.Tour_Parirs' collection, showing 56.4k documents and 1 index. The MONGODB shell contains a query:

```
> db.Tour_Parirs.find(
  {
    "services": "chambres non-fumeurs"
  },
  {
    "name": 1,
    "services": 1,
    "_id": 0
  }
)
```

 The result shows a document for 'Arès Tour Eiffel' with a list of services including 'journaux', 'centre de remise en forme', 'petit-déjeuner en chambre', 'réception ouverte 24h 24', 'bagagerie', 'service d'étage', 'navette aéroport', 'garde d'enfants', 'blanchisserie', 'nettoyage à sec', 'service de repassage', and 'bureau d'excursions'.

3. Donner le nom et numéro de téléphone des lieux ayant un numéro de téléphone renseigné;



The screenshot shows the MongoDB Compass interface. The top bar indicates the connection to 'localhost:27017/Tour_Parirs.Tour_Parirs'. The left sidebar shows 'My Queries' and 'Databases'. The main panel displays the 'Documents' tab for the 'Tour_Parirs.Tour_Parirs' collection, showing 56.4k documents and 1 index. The MONGODB shell contains a query:

```
> db.Tour_Parirs.find(
  {
    "contact.phone": { $exists: true }
  },
  {
    "name": 1,
    "contact.phone": 1,
    "_id": 0
  }
)
```

 The result shows three documents, each with a 'contact' object containing a 'phone' field and a 'name' field. The names are 'Alcar', 'Deli's café', and an unnamed entry.

5. Nom et services des lieux proposant 4 services au moins;

`db.Tour_Parirs.find(`

```
{
  "services": { $exists: true, $not: { $size: 0 }, $size: { $gte: 4 } }
},
{
  "name": 1,
  "services": 1,
  "_id": 0
}
)
```

6. Donner la liste distincte des catégories.

```
db.Tour_Paris.distinct("category")
```

```
> db.Tour_Paris.distinct("category")
< [ 'accommodation', 'attraction', 'poi', 'restaurant' ]
Tour_Paris> |
```

7. Donner les adresses des lieux de catégorie "accommodation" avec un service "blanchisserie", projeter le résultat sur le nom et numéro de téléphone, et trier sur le nom;

```
My Queries
Databases
+ Tour_Paris.Tour_Paris 56.4k DOCUMENTS 1 INDEXES
> MONGODB
> db.Tour_Paris.distinct("category")
< [ 'accommodation', 'attraction', 'poi', 'restaurant' ]
> db.Tour_Paris.find(
  {
    "category": "accommodation",
    "services": "blanchisserie"
  },
  {
    "name": 1,
    "contact.phone": 1,
    "location.address": 1,
    "_id": 0
  }
).sort({"name": 1})
< {
  contact: {
    phone: '+33 1 47 70 78 34'
  },
  name: '9HOTEL OPERA',
  location: {
    address: 'Paris, France 14, Rue Papillon, 09. Opera - Haussmann, 75009 Paris'
  }
}
{
  contact: {
    phone: '+33 1 47 70 78 34'
  },
  name: 'HOTEL OPERA',
  location: {
    address: 'Paris, France 14, Rue Papillon, 09. Opera - Haussmann, 75009 Paris'
  }
}
```

```
db.Tour_Paris.find(
```

```
{
  "category": "accommodation",
  "services": "blanchisserie"
},
{
  "name": 1,
  "contact.phoneNumber": 1,
  "location.address": 1,
  "_id": 0
}
).sort({"name": 1})
```

8. Nombre de lieux de catégorie "accommodation" et ayant un service "chambres non-fumeurs";

```
db.Tour_Paris.count(
{
  "category": "accommodation",
  "services": "chambres non-fumeurs" })
```

```
> db.Tour_Paris.count(
  {
    "category": "accommodation",
    "services": "chambres non-fumeurs" })
< DeprecationWarning: Collection.count() is deprecated. Use countDocuments or estimatedDocumentCount.
< 1313
Tour_Paris>
```

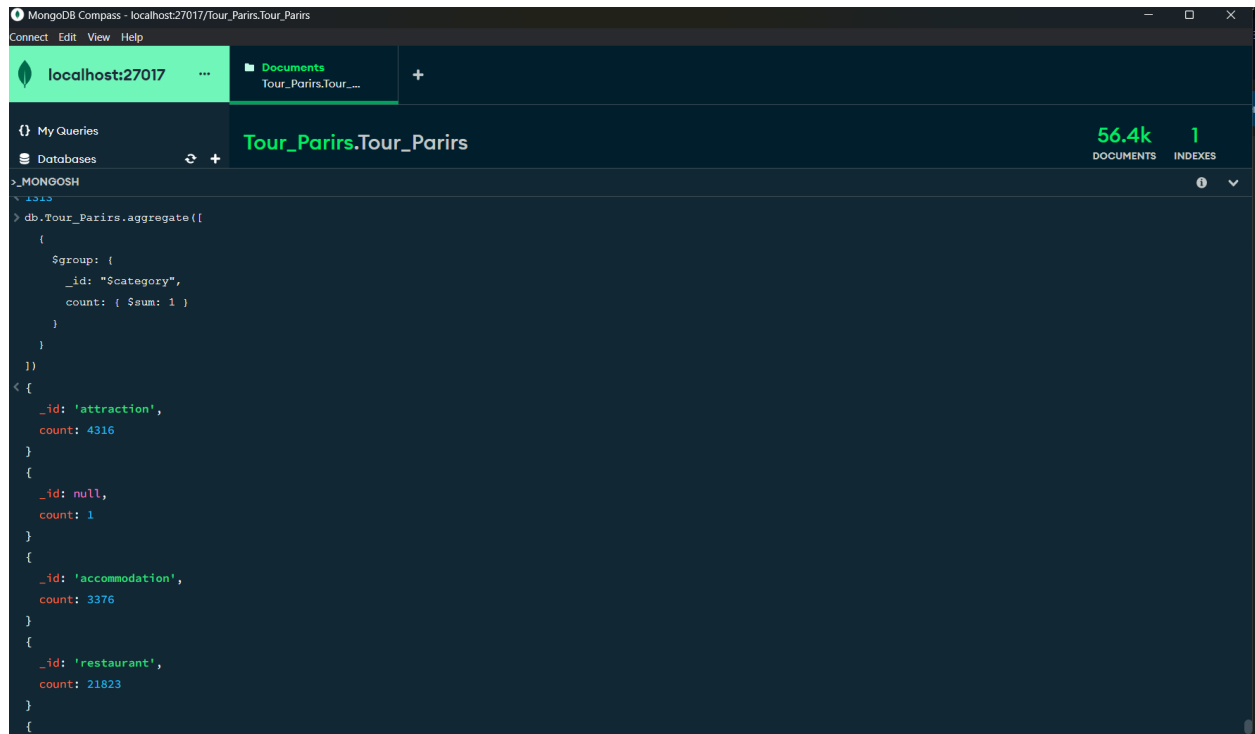
9. Donner le nombre de lieux par catégorie;

```
db.Tour_Paris.aggregate([
{
  $group: {
    _id: "$category",
    count: { $sum: 1 }
  }
})
```

NOUR ROUIS ING2 INFO

```
}
```

```
])
```



The screenshot shows the MongoDB Compass interface. The top bar indicates the connection to 'localhost:27017/Tour_Paris.Tour_Paris'. The left sidebar shows 'My Queries' and 'Databases'. The main panel displays the 'Tour_Paris.Tour_Paris' collection with 56.4k documents and 1 index. The aggregation query is as follows:

```
> db.Tour_Paris.aggregate([
  {
    $group: {
      _id: "$category",
      count: { $sum: 1 }
    }
  }
])
```

The result set shows three documents:

```
{
  "_id": "attraction",
  "count": 4316
}
{
  "_id": null,
  "count": 1
}
{
  "_id": "accommodation",
  "count": 3376
}
{
  "_id": "restaurant",
  "count": 21823
}
```

10. Pour les lieux de catégorie "accommodation", donner le nombre de lieux pour chaque service ;

```
db.Tour_Paris.aggregate([
```

```
{
```

```
  $match: {
```

```
    "category": "accommodation"
```

```
  }
```

```
},
```

```
{
```

```
  $unwind: "$services"
```

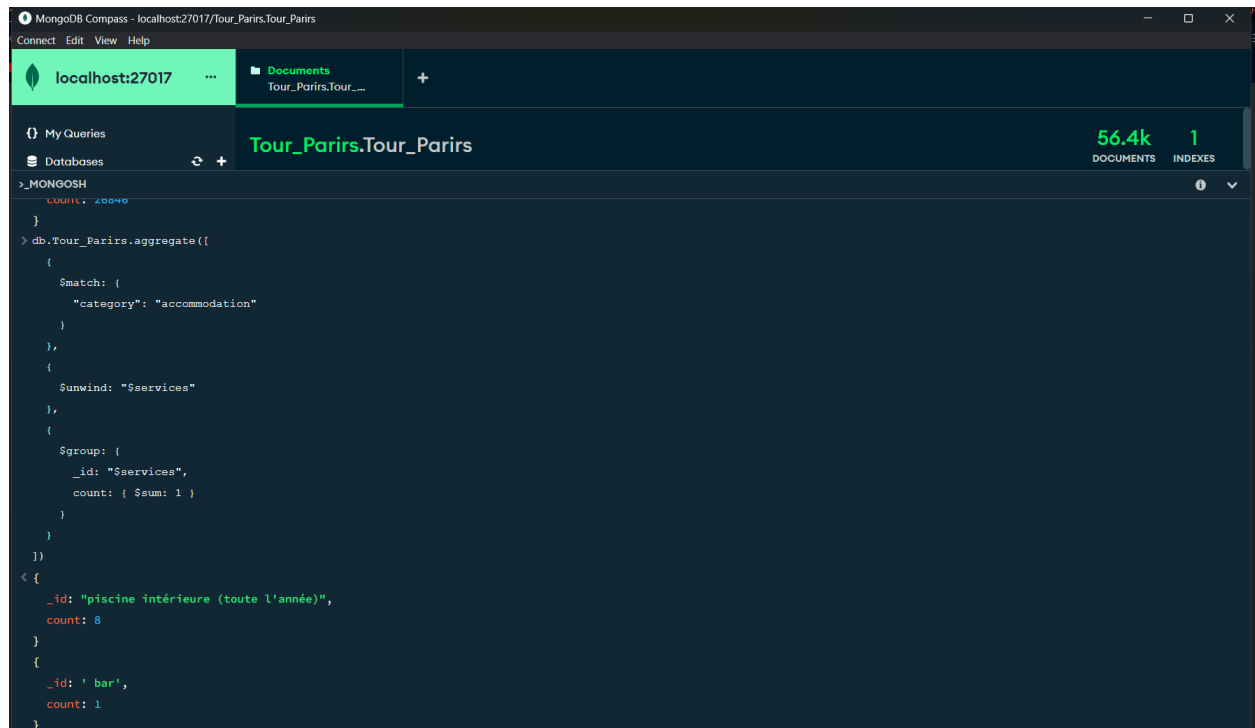
```
},
```

```
{
```

```
  $group: {
```

NOUR ROUIS ING2 INFO

```
    _id: "$services",  
    count: { $sum: 1 }  
  }  
}  
])
```



11. Ajouter aux lieux une information sur le nombre des reviews (clé : lieu.nb_reviews) ;

```
db.Tour_Paris.updateMany(  
  {},  
  {  
    $set: {  
      "nb_reviews": { $size: "$reviews" }  
    }  
  }  
)
```

```
> db.Tour_Paris.updateMany(  
  {},  
  {  
    $set: {  
      "nb_reviews": { $size: "$reviews" }  
    }  
  }  
)  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 56362,  
  modifiedCount: 56362,  
  upsertedCount: 0  
}
```

12. Afficher pour chaque nombre de reviews, le nombre de documents correspondants.

```
db.Tour_Paris.aggregate([  
  {  
    $group: {  
      _id: "$nb_reviews",  
      count: { $sum: 1 }  
    }  
  }  
])
```

```
> db.Tour_Parirs.aggregate([
  {
    $group: {
      _id: "$nb_reviews",
      count: { $sum: 1 }
    }
  }
])
< {
  _id: {
    '$size': '$reviews'
  },
  count: 56362
}
```