PROJET 7

OPENCLASSROOMS: FORMATION

INGÉNIEUR DATA

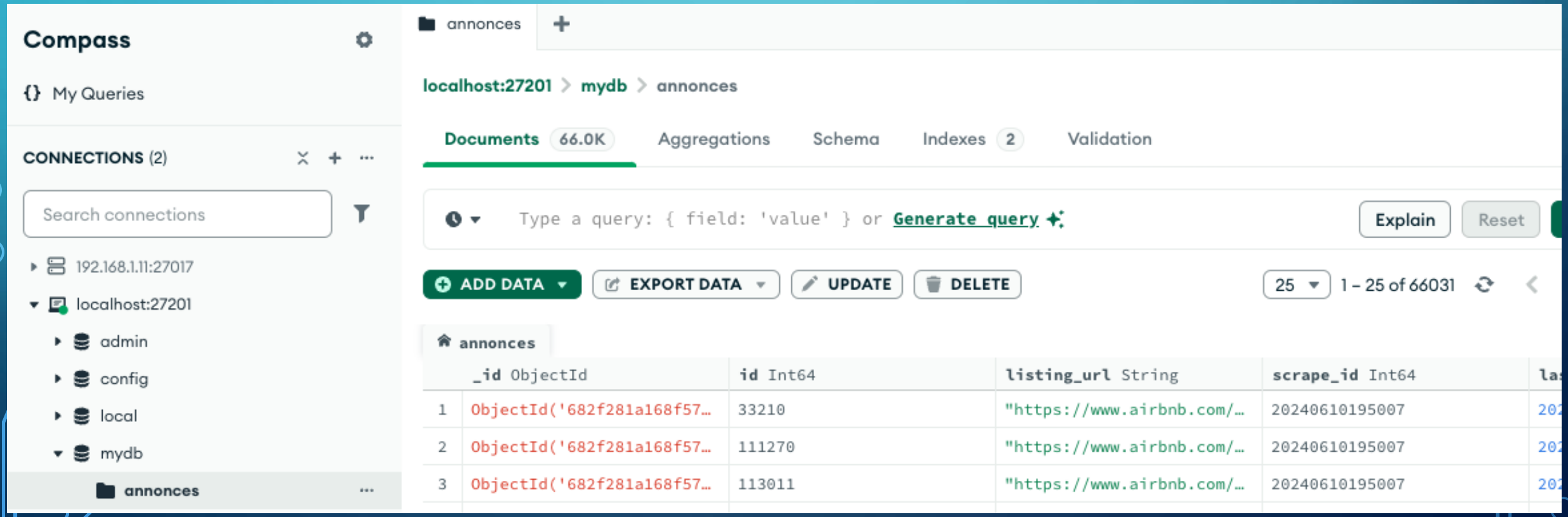
Contexte:

Data Engineer dans l’association NosCités.

But du projet:

Explorer et analyser une sauvegarde de données pour vérifier si elle correspond bien aux données attendues.

- Filtrage du jeu de données en ne gardant que « Paris, France » comme lieu d'annonces
- Import du fichier csv des données filtrées avec MongoDB Compass



Pourquoi choisir une base de données NoSQL?

Cas d'usage typique du NoSQL:

- Gros volume (dans notre cas, environ 70000 documents),
- Champs très variés (pas de structure fixe imposée),
- Lecture rapide.

Le format JSON-like (BSON) de MongoDB est idéal pour modéliser :

- des sous-objets (host, reviews, etc.),
- des tableaux (amenities, host_verifications, etc.).

Nombre de documents

```
> db["annonces"].countDocuments()
< 66031
```

66031

Nombre de logements ayant des disponibilités

```
> db["annonces"].countDocuments({has_availability: true})
< 62581
```

62581

Nombre d'annonces par type de location

```
> db["annonces"].aggregate([
    { $group: { _id: "$room_type", count: { $sum: 1 } } },
    { $sort: { count: -1} }
  ])
< {
    _id: 'Entire home/apt',
    count: 59201
  }
  {
    _id: 'Private room',
    count: 6067
  }
  {
    _id: 'Hotel room',
    count: 513
  }
  {
    _id: 'Shared room',
    count: 250
  }
```

Type location	Nombre
Entire home/apt	59201
Private room	6067
Hotel room	513
Shared room	250

Nombre d'annonces par type de location

```
> db["annonces"].aggregate([
    { $group: { _id: "$property_type", count: { $sum: 1 } } },
    { $sort: { count: -1} }
  ])
< {
    _id: 'Entire rental unit',
    count: 55483
  }
  {
    _id: 'Private room in rental unit',
    count: 4252
  }
  {
    _id: 'Entire condo',
    count: 1862
  }
```

Type location	Nombre
Entire rental unit	55483
Private room in rental unit	4252
Entire condo	1862
Entire loft	816
Room in boutique hotel	789
Entire home	478
…	…
Private room in guest suite	29

Les 5 annonces de location avec le plus d'évaluations et leurs nombres

```
> db["annonces"].find({}, { name: 1, number_of_reviews: 1 })
    .sort({ number_of_reviews: -1 })
    .limit(5)
< {
    _id: ObjectId('682f281d168f577b587397d9'),
    name: 'Sweet & cosy room next to Canal Saint Martin ♥',
    number_of_reviews: 3067
  }
  {
    _id: ObjectId('682f281e168f577b5873aa6e'),
    name: 'Double/Twin Room, close to Opera and the Louvre with breakfast included',
    number_of_reviews: 2620
  }
  {
    _id: ObjectId('682f2820168f577b5873c4f2'),
    name: 'Bed in Dorm of 8 Beds "The Big One" in Paris',
    number_of_reviews: 2294
  }
  {
    _id: ObjectId('682f281f168f577b5873c383'),
```

Nombre d'évaluations
3067
2620
2294
2105
1716

Nombre total d’hôtes différents

```
> db["annonces"].aggregate([
    { $group: { _id: "$host_id" } },
    { $count: "total_hosts" }
  ])
```

50772

ou

```
> db["annonces"].distinct("host_id").length
< 50772
```

Nombre de locations réservables instantanément et la proportion

```
> db["annonces"].aggregate([
    {
      $facet: {
        total: [ { $count: "value" }],
        instant: [ { $match: { instant_bookable: true } }, { $count: "value" }]
      }
    },
    {
      $project: { total: { $arrayElemAt: ["$total.value", 0] }, instant: { $arrayElemAt: ["$instant.value", 0] } }
    },
    {
      $project: {
        instant_bookable_count: "$instant",
        proportion: {
          $multiply: [
            { $divide: ["$instant", "$total"] },
            100
          ]
        }
      }
    }
  ])
```

12598

soit

19.08 %

Nombre d’hôtes ayant plus de 100 annonces sur les plateformes, leurs noms et la proportion

```
> db["annonces"].aggregate([
    {
      $group: { _id: "$host_id", host_name: { $first: "$host_name" }, total_listings: { $first: "$host_total_listings_count" } }
    },
    {
      $facet: {
        highVolume: [ { $match: { total_listings: { $gt: 100 } } }, { $project: { _id: 0, host_name: 1 } }, { $sort: { host_name: 1 } } ],
        totalHosts: [ { $count: "count" } ]
      }
    },
    {
      $project: { highVolume: 1, countHighVolume: { $size: "$highVolume" }, total: { $arrayElemAt: ["$totalHosts.count", 0] } }
    },
    {
      $project: {
        highVolume: 1, countHighVolume: 1,
        proportion: {
          $multiply: [
            { $divide: ["$countHighVolume", "$total"] },
            100
          ]
        }
      }
    }
  ])
```

host_name
Pierre De WeHost
Blueground
Reservation Desk
Ludovic
…

62 hôtes

0,12% des hôtes

Nombre de super hôtes différents et la proportion

```
> db["annonces"].aggregate([
    {
      "$group": { "_id": "$host_id", "is_superhost": { "$first": "$host_is_superhost" } }
    },
    {
      "$facet": { "superhosts": [ { "$match": { "is_superhost": true } } ], "all_hosts": [ { "$count": "count" } ] }
    },
    {
      "$project": { "countSuperhosts": { "$size": "$superhosts" }, "totalHosts": { "$arrayElemAt": ["$all_hosts.count", 0] } }
    },
    {
      "$project": {
        "countSuperhosts": 1,
        "totalHosts": 1,
        "proportion": {
          "$multiply": [
            { "$divide": ["$countSuperhosts", "$totalHosts"] },
            100
          ]
        }
      }
    }
  ])
```

7509 super hôtes

soit

14.79 %

Taux de réservation moyen par mois par type de logement

```python
# Récupération des documents
cursor = collection.find(
    { "reviews_per_month": { "$ne": None } },  # exclut les valeurs nulles
    { "room_type": 1, "reviews_per_month": 1, "_id": 0 }
)
# Conversion en liste de dictionnaires
data = list(cursor)
# Création du DataFrame Polars
df = pl.from_dicts(data)
# Calcul du taux moyen de réservation par type de logement
result = (
   df.group_by("room_type")
      .agg(pl.col("reviews_per_month").mean().alias("avg_reviews_per_month"))
      .sort("avg_reviews_per_month", descending=True)
)
```

Type logement	Taux moyen
Shared room	2.20
Private room	1.32
Entire home/apt	1.15
Hotel room	1.04

Taux de réservation moyen par mois par type de logement

```python
# Récupération des documents
cursor = collection.find(
    { "reviews_per_month": { "$ne": None } },  # exclut les valeurs nulles
    { "property_type": 1, "reviews_per_month": 1, "_id": 0 }
)
# Conversion en liste de dictionnaires
data = list(cursor)
# Création du DataFrame Polars
df = pl.from_dicts(data)
# Calcul du taux moyen de réservation par type de logement
result = (
   df.group_by("property_type")
      .agg(pl.col("reviews_per_month").mean().alias("avg_reviews_per_month"))
      .sort("avg_reviews_per_month", descending=True)
)
```

Type logement	Taux moyen
Room in hostel	5.88
Private room in boat	5.34
Shared room in hostel	4.61
…	…
Earthen home	0.2

Médiane du nombre d'avis pour tous les logements

```python
# Récupérer les champs utiles et filtrer les nulls
cursor = collection.find(
    { "number_of_reviews": { "$ne": None } },
    { "number_of_reviews": 1, "_id": 0 }
)
# Convertir en DataFrame Polars
df = pl.from_dicts(list(cursor))
# Calcul de la médiane
median_reviews = df.select(pl.col("number_of_reviews").median()).item()
```

→ 4

Médiane du nombre d'avis par catégorie d'hôte

```python
# Récupérer les données utiles : nombre d'avis et statut superhost
cursor = collection.find(
    {
        "number_of_reviews": { "$ne": None },
        "host_is_superhost": { "$ne": None }
    },
    {
        "number_of_reviews": 1, "host_is_superhost": 1, "_id": 0
    }
)
# Création du DataFrame Polars
df = pl.from_dicts(list(cursor))
# Calcul du taux moyen de réservation par type de logement
result = (
    df.group_by("host_is_superhost")
      .agg(pl.col("number_of_reviews").median().alias("median_reviews"))
)
```

Super hôte	Médiane
Oui	27
Non	3

Densité de logements par quartier de Paris

```python
# Charger les quartiers
cursor = collection.find(
    { "neighbourhood_cleansed": { "$ne": None } },
    { "neighbourhood_cleansed": 1, "_id": 0 }
)
# Création du DataFrame
df = pl.from_dicts(list(cursor))
# Compter les logements par quartier
result = (
    df.group_by("neighbourhood_cleansed")
      .agg(pl.len().alias("nombre_logements"))
      .sort("nombre_logements", descending=True)
)
```

Quartier	Nombre	Quartier	Nombre
Buttes-Montmartre	7772	Reuilly	2709
Popincourt	6141	Observatoire	2367
Vaugirard	5030	Gobelins	2174
Entrepôt	4829	Bourse	2045
Batignolles-Monceau	4584	Hôtel-de-Ville	1972
Buttes-Chaumont	4135	Panthéon	1927
Ménilmontant	4067	Élysée	1796
Passy	3872	Luxembourg	1746
Opéra	3108	Palais-Bourbon	1705
Temple	2726	Louvre	1326

Quartiers avec le plus fort taux de réservation par mois

```python
# Extraire les champs nécessaires
cursor = collection.find(
    {
        "neighbourhood_cleansed": { "$ne": None },
        "reviews_per_month": { "$ne": None }
    },
    {
        "neighbourhood_cleansed": 1,
        "reviews_per_month": 1,
        "_id": 0
    }
)
# Charger dans un DataFrame Polars
df = pl.from_dicts(list(cursor))
# Calcul du taux moyen de réservation par quartier
result = (
    df.group_by("neighbourhood_cleansed")
      .agg(pl.col("reviews_per_month").mean()
           .alias("moyenne_rsv_mensuelle"))
      .sort("moyenne_rsv_mensuelle", descending=True)
)
```

Quartier	Nombre	Quartier	Nombre
Bourse	1.57	Panthéon	1.09
Louvre	1.36	Passy	1.09
Élysée	1.28	Luxembourg	1.06
Hôtel-de-Ville	1.24	Gobelins	1.02
Palais-Bourbon	1.21	Observatoire	1.02
Temple	1.21	Popincourt	1.00
Vaugirard	1.17	Batignolles-Monceau	0.97
Opéra	1.11	Buttes-Montmartre	0.97
Reuilly	1.11	Buttes-Chaumont	0.83
Entrepôt	1.10	Ménilmontant	0.83

Connection de Power BI à la base de données MongoDB

- Installation d'un driver ODBC (CData MongoDB Power BI Connector) s'interfaçant avec MongoDB
- Power BI utilise une connection ODBC

MongoDB Compass - Home/local.annonces

Compass

My Queries

CONNECTIONS (1)

Search connections

- Home
  - admin
  - config
  - local
    - annonces
    - startup_log
  - mongosqld_data

annonces

Home > local > annonces

Open MongoDB shell

Documents 95.9K | Aggregations | Schema | Indexes 1 | Validation

Type a query: { field: 'value' } or Generate query

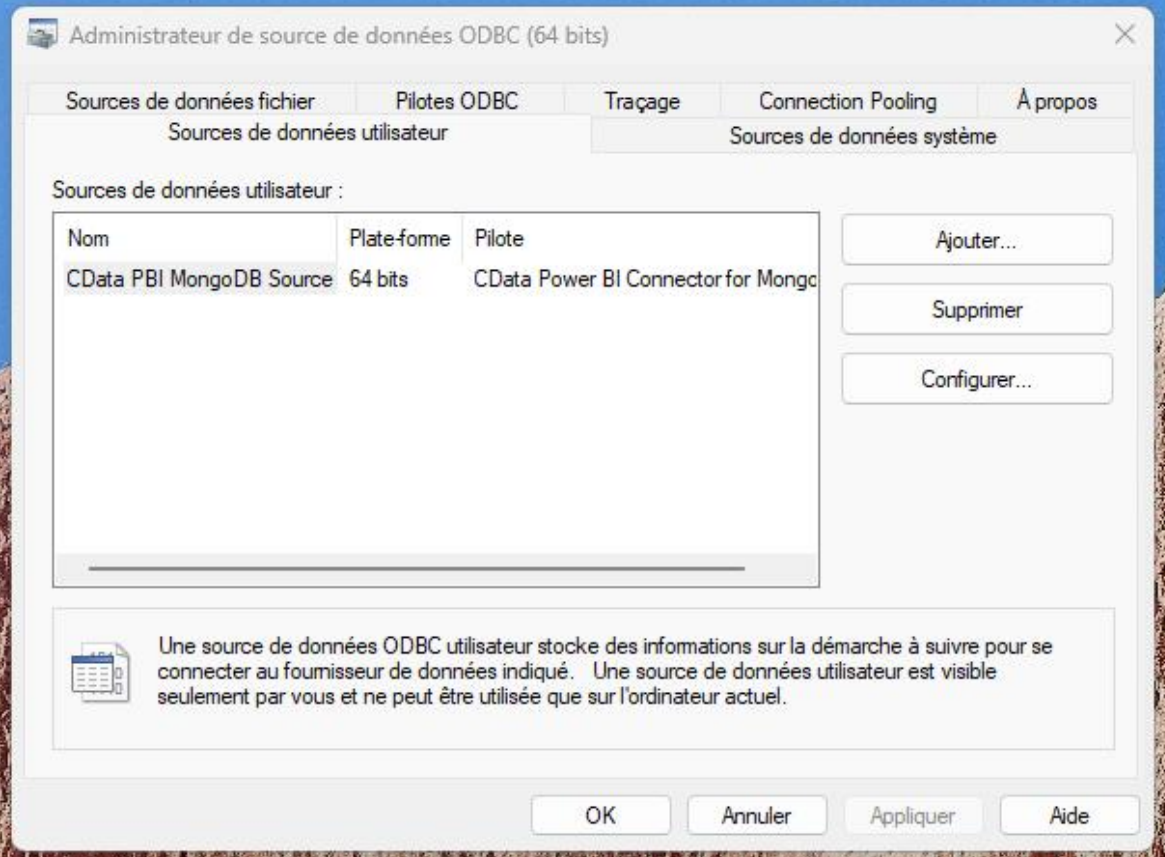
Explain | Reset | Find | Options

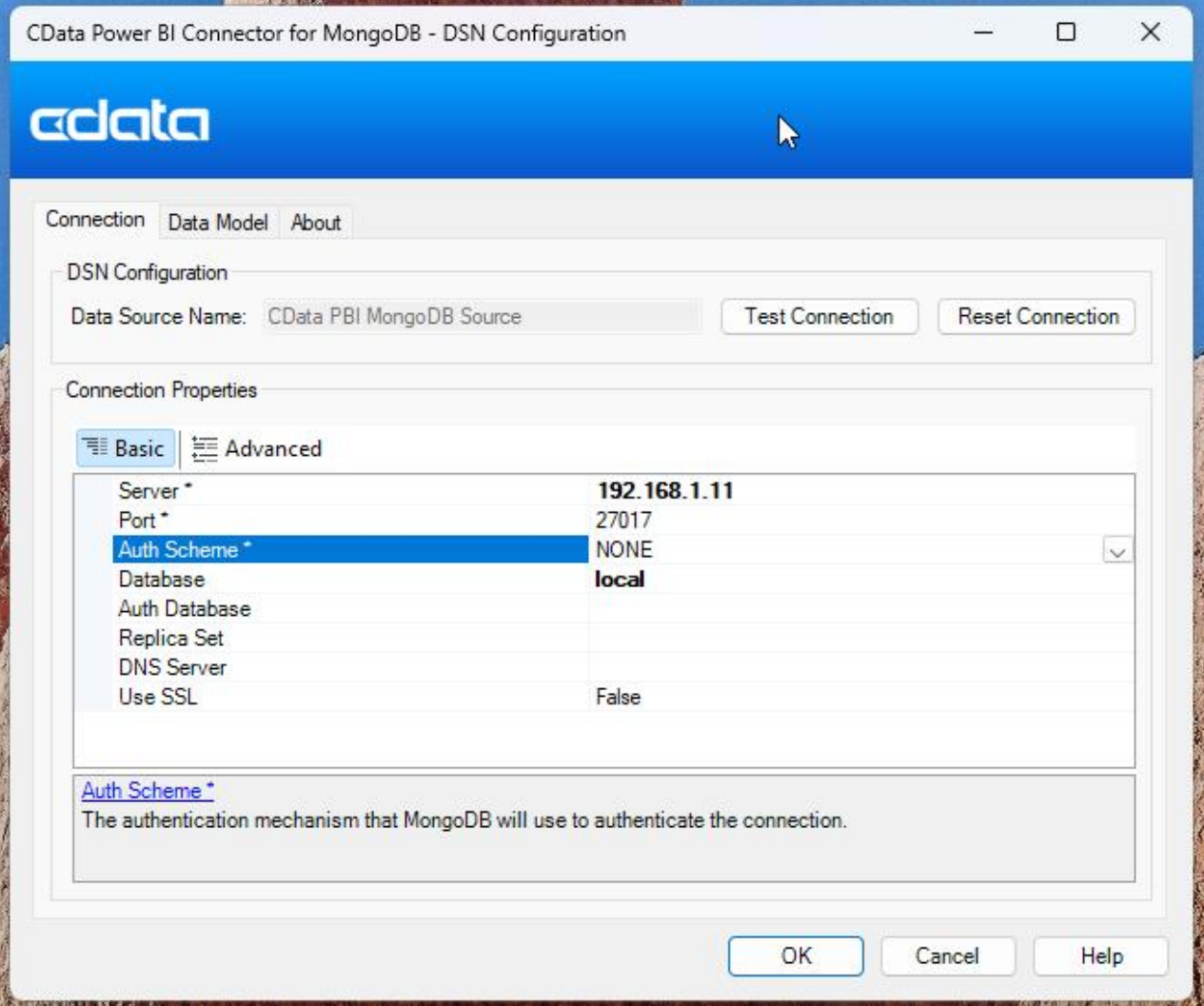
ADD DATA | EXPORT DATA | UPDATE | DELETE

25 | 1 – 25 of 95885

annonces

	_id ObjectId	id Int64	listing_url String	scrape_id Int64	last_scraped Date
1	ObjectId('6821aee53bd5a5…	80260	"https://www.airbnb.com/…	20240610195007	2024-06-13T00:00:00.0
2	ObjectId('6821aee53bd5a5…	3109	"https://www.airbnb.com/…	20240610195007	2024-06-12T00:00:00.0
3	ObjectId('6821aee53bd5a5…	80301	"https://www.airbnb.com/…	20240610195007	2024-06-13T00:00:00.0
4	ObjectId('6821aee53bd5a5…	5396	"https://www.airbnb.com/…	20240610195007	2024-06-13T00:00:00.0
5	ObjectId('6821aee53bd5a5…	7397	"https://www.airbnb.com/…	20240610195007	2024-06-13T00:00:00.0


Administrateur de source de données ODBC (64 bits)
Sources de données fichier
Pilotes ODBC
Traçage
Connection Pooling
À propos
Sources de données utilisateur
Sources de données système
Sources de données utilisateur :
Nom
Plate-forme
Pilote
CData PBI MongoDB Source
64 bits
CData Power BI Connector for Mongc
Ajouter...
Supprimer
Configurer...
Une source de données ODBC utilisateur stocke des informations sur la démarche à suivre pour se connecter au fournisseur de données indiqué. Une source de données utilisateur est visible seulement par vous et ne peut être utilisée que sur l'ordinateur actuel.
OK
Annuler
Appliquer
Aide


CData Power BI Connector for MongoDB - DSN Configuration
cdata
Connection
Data Model
About
DSN Configuration
Data Source Name:
CData PBI MongoDB Source
Test Connection
Reset Connection
Connection Properties
Basic
Advanced
Server *
192.168.1.11
Port *
27017
Auth Scheme *
NONE
Database
local
Auth Database
Replica Set
DNS Server
Use SSL
False
Auth Scheme *
The authentication mechanism that MongoDB will use to authenticate the connection.
OK
Cancel
Help

super_host • Dernière enregistrement : Aujourd'hui à 13:39

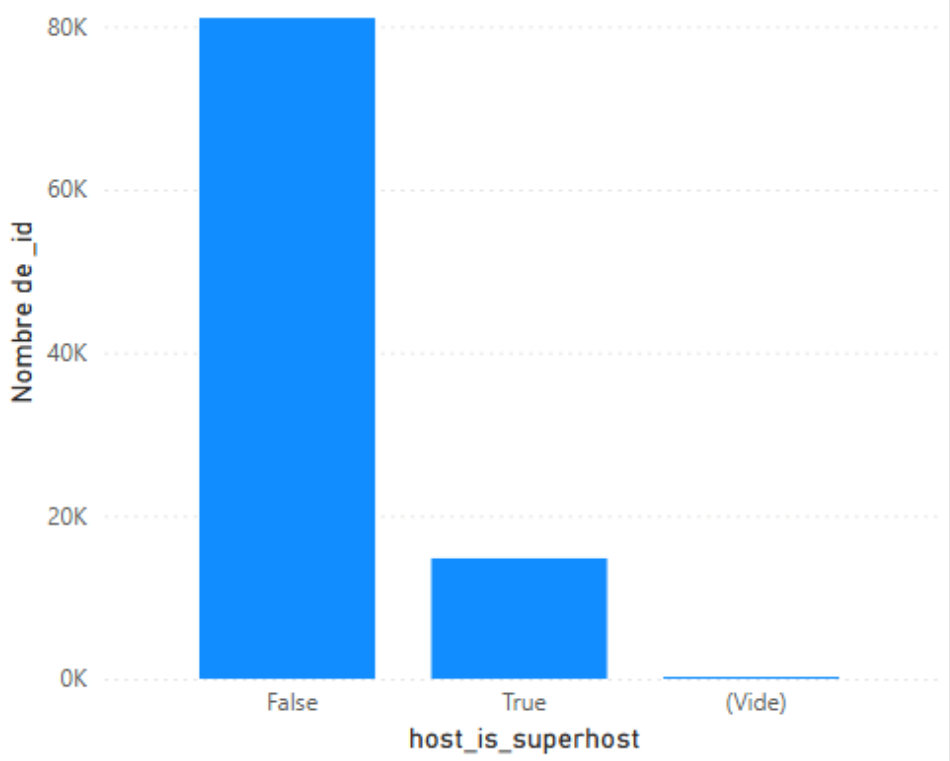
Se connecter

Fichier | Accueil | Insérer | Modélisation | Afficher | Optimiser | Aide | Format | Données / Explorer

Coller | Obtenir les données | Classeur Excel | Catalogue OneLake | SQL Server | Entrer des données | Dataverse | Sources récentes | Transformer les données | Actualiser | Nouveau visuel | Zone de texte | Plus de visuels | Nouveau calcul de visuel | Nouvelle mesure | Mesure rapide | Confidentialité | Publier | Préparer les

Presse-papiers | Données | Requêtes | Insérer | Calculs | Confidentialité | Partager

Retour au rapport | NOMBRE DE _ID PAR HOST_IS_SUPERHOST



Filtres

Rechercher

Filtres sur ce visuel

host_is_superhost
est (Tout)

Nombre de _id
est (Tout)

Ajouter des champs de do...

Filtres dans cette page

Ajouter des champs de do...

Filtres dans toutes les pages

Ajouter des champs de do...

Visualisations

Générer un élément visuel

Axe X

host_is_superhost

Axe Y

Nombre de _id

Légende

Ajouter des champs de données ici

Petits multiples

Données

Rechercher

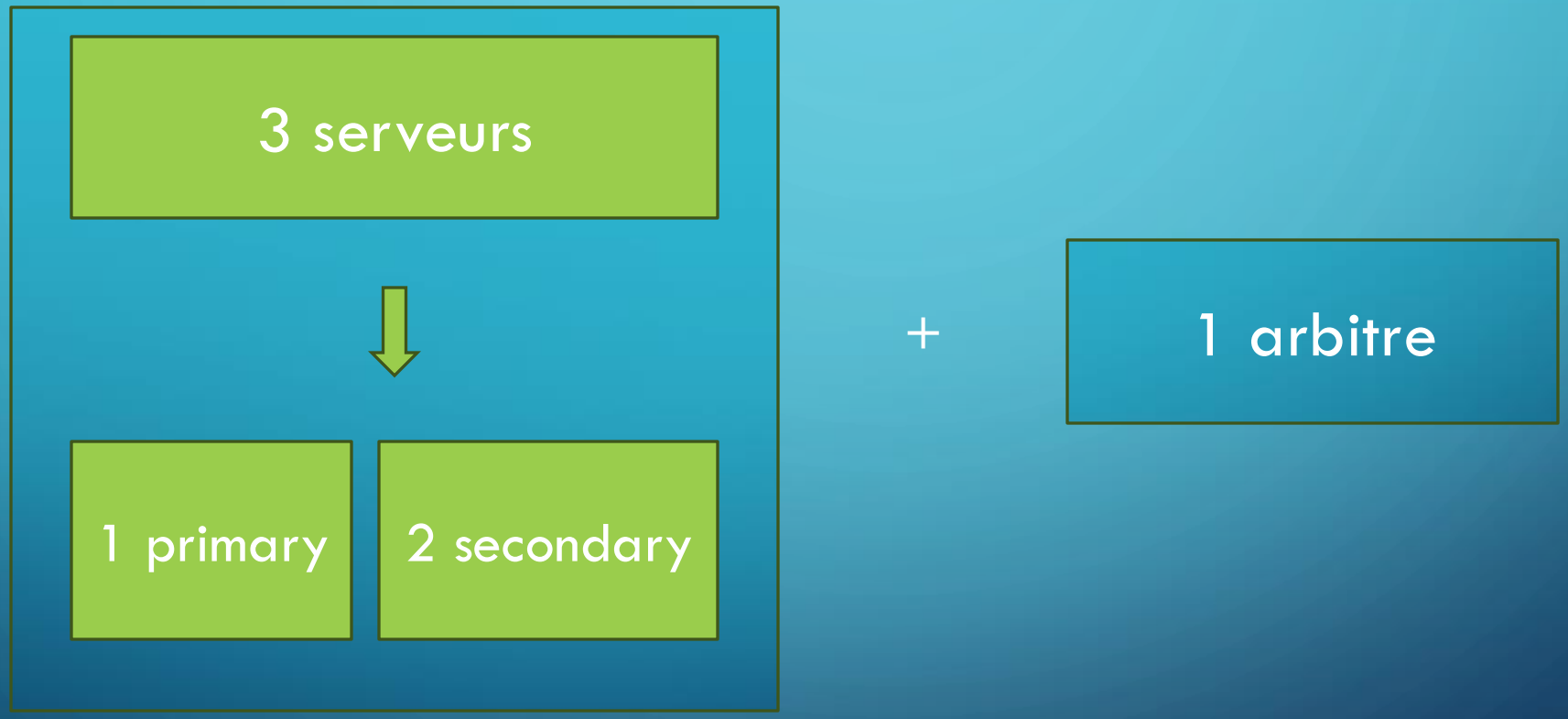
- maximum_minimum_nights
- maximum_nights
- maximum_nights_avg_ntm
- minimum_maximum_nights
- minimum_minimum_nights
- minimum_nights
- minimum_nights_avg_ntm
- name
- neighborhood_overview
- neighbourhood
- neighbourhood_cleansed
- number_of_reviews
- number_of_reviews_l30d
- number_of_reviews_ltm
- picture_url
- price
- property_type

Page 1 sur 1

Import des annonces de « Lyon, France » avec MongoDB Compass

72702 annonces (Paris + Lyon)

Tolérance à la panne: mise en place d'un ReplicaSet



Exemple d'un script créant un ReplicaSet

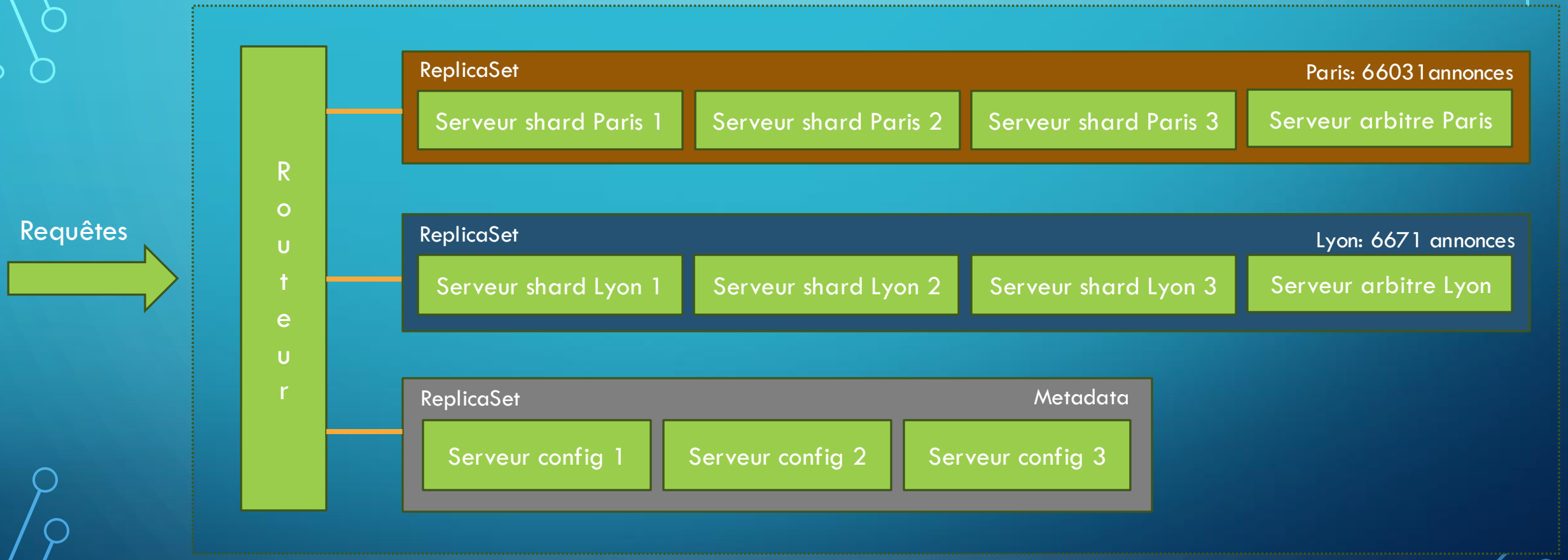
```
#!/bin/bash

# Variables
BASE_DIR="/data"
RS_NAME=rs0
BINDIP=0.0.0.0

# Création des dossiers
echo "Création des répertoires de données..."
mkdir -p "$BASE_DIR/rs0" "$BASE_DIR/rs1" "$BASE_DIR/rs2" "$BASE_DIR/arb"

# Lancement des instances mongod
echo "Lancement des instances mongod..."
mongod --replSet $RS_NAME --port 27017 --dbpath "$BASE_DIR/rs0" --bind_ip $BINDIP --fork --logpath "$BASE_DIR/rs0/mongod.log"
mongod --replSet $RS_NAME --port 27018 --dbpath "$BASE_DIR/rs1" --bind_ip $BINDIP --fork --logpath "$BASE_DIR/rs1/mongod.log"
mongod --replSet $RS_NAME --port 27019 --dbpath "$BASE_DIR/rs2" --bind_ip $BINDIP --fork --logpath "$BASE_DIR/rs2/mongod.log"
mongod --replSet $RS_NAME --port 30000 --dbpath "$BASE_DIR/arb" --bind_ip $BINDIP --fork --logpath "$BASE_DIR/arb/mongod.log"
# Attente de démarrage
sleep 5
echo "Initialisation du ReplicaSet..."
mongosh --port 27017 --eval "
  rs.initiate({
    _id: '$RS_NAME',
    members: [
      { _id: 0, host: 'localhost:27017' },
      { _id: 1, host: 'localhost:27018' },
      { _id: 2, host: 'localhost:27019' },
      { _id: 3, host: 'localhost:30000', arbiterOnly: true }
    ]
  });
"
```

DISTRIBUTION DES DONNÉES



CONFIGURATION ROUTEUR

```
sharding:
configDB: cfgReplSet/localhost:27018,localhost:27019,localhost:27020

net:
bindIp: 0.0.0.0
port: 27017

systemLog:
destination: file
logAppend: true
path: /data/sharding/mongos.log

processManagement:
fork: true
```

CONFIGURATION SERVEUR CONFIGURATION 1

```
sharding:
clusterRole: configsvr

replication:
replSetName: cfgReplSet

net:
bindIp: 0.0.0.0
port: 27018

storage:
dbPath: /data/sharding/config1

systemLog:
destination: file
logAppend: true
path: /data/sharding/config1.log

processManagement:
fork: true
```

CONFIGURATION SERVEUR SHARD LYON 1

```
sharding:
clusterRole: shardsvr

replication:
replSetName: rsLyon

net:
bindIp: 0.0.0.0
port: 27101

storage:
dbPath: /Users/nicolas/Openclassrooms/projet 7/data/sharding/shardLyon-res0

systemLog:
destination: file
logAppend: true
path: /Users/nicolas/Openclassrooms/projet 7/data/sharding/shardLyon-res0.log

processManagement:
fork: true
```

CONFIGURATION SERVEUR SHARD PARIS 1

```
sharding:
clusterRole: shardsvr

replication:
replSetName: rsParis

net:
bindIp: 0.0.0.0
port: 27201

storage:
dbPath: /Users/nicolas/Openclassrooms/projet 7/data/sharding/shardParis-res0

systemLog:
destination: file
logAppend: true
path: /Users/nicolas/Openclassrooms/projet 7/data/sharding/shard1-res0.log

processManagement:
fork: true
```

Initialisation du sharding

```
rs.initiate({
  _id: "cfgReplSet", configsvr: true,
  members: [ { _id: 0, host: "192.166.1.11:27018" }, { _id: 1, host: "192.166.1.11:27019" }, { _id: 2, host: "192.166.1.11:27020" }]
});
rs.initiate({
  _id: "rsLyon",
  members: [{ _id: 0, host: "192.168.1.11:27101" }, { _id: 1, host: "192.168.1.11:27102" }, { _id: 2, host: "192.168.1.11:27103" },
                { _id: 3, host: "192.168.1.11:27110", arbiterOnly: true }]
});
rs.initiate({
 _id: "rsParis",
 members: [ { _id: 0, host: "192.168.1.11:27201" }, { _id: 1, host: "192.168.1.11:27202" }, { _id: 2, host: "192.168.1.11:27203" },
               { _id: 3, host: "192.168.1.11:27210", arbiterOnly: true }]
});
db.adminCommand({
  setDefaultRWConcern: 1, defaultWriteConcern: { w: "majority" }, defaultReadConcern: { level: "local" }
})

sh.addShard("rsLyon/192.168.1.11:27101,192.168.1.11:27102,192.168.1.11:27103,192.168.1.11:27110")
sh.addShard("rsParis/192.168.1.11:27201,192.168.1.11:27202,192.168.1.11:27203,192.168.1.11:27210")

use mydb;
db.annonces.createIndex({ host_location: 1 });
sh.enableSharding("mydb");
sh.shardCollection("mydb.annonces", { host_location: 1 });
sh.splitAt("mydb.annonces", { host_location: "Paris, France" });
sh.moveChunk("mydb.annonces", { host_location: "Lyon, France" }, "rsLyon");
sh.moveChunk("mydb.annonces", { host_location: "Paris, France" }, "rsParis");
```

SHARDING STATUS

shardedDataDistribution

```
[
    {
    ns: 'mydb.annonces',
    shards: [
      {
        shardName: 'rsLyon',
        numOrphanedDocs: 0,
        numOwnedDocuments: 6671,
        ownedSizeBytes: 22754781,
        orphanedSizeBytes: 0
      },
      {
        shardName: 'rsParis',
        numOrphanedDocs: 0,
        numOwnedDocuments: 66031,
        ownedSizeBytes: 228995508,
        orphanedSizeBytes: 0
      }
    ]
  }
]
```

Sharding • Dernière enregistrement : Aujourd'hui à 19:07

Se connecter

Fichier | Accueil | Insérer | Modélisation | Afficher | Optimiser | Aide | Format | Données / Explorer

Coller | Couper | Copier | Reproduire la mise en forme (Presse-papiers)

Obtenir les données | Classeur Excel | Catalogue OneLake | SQL Server | Entrer des données | Dataverse | Sources récentes (Données)

Transformer les données | Actualiser (Requêtes)

Nouveau visuel | Zone de texte | Plus de visuels (Insérer)

Nouveau calcul de visuel | Nouvelle mesure rapide (Calculs)

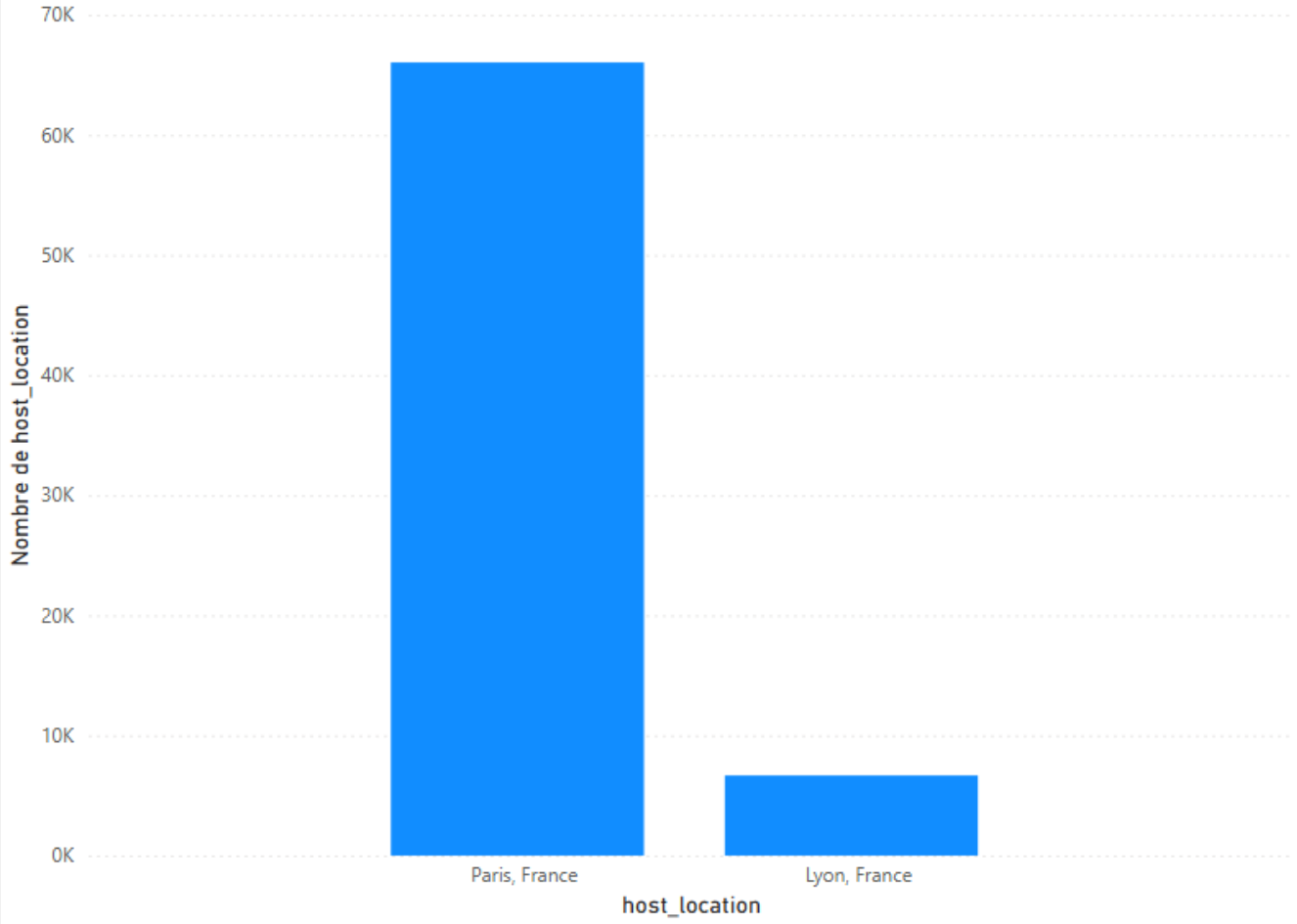
Confidentialité

Publier (Partager)

Préparer les données pour l'IA | Copi (Copilot)

Retour au rapport

NOMBRE DE HOST_LOCATION PAR HOST_LOCATION



Filtres

Rechercher

Filtres sur ce visuel

host_location est (Tout)

Nombre de host_location est (Tout)

Ajouter des champs de données ici

Filtres dans cette page

Ajouter des champs de données ici

Filtres dans toutes les pages

Ajouter des champs de données ici

Visualisations

Générer un élément visuel

Axe X

host_location

Axe Y

Nombre de host_location

Légende

Ajouter des champs de données ici

Petits multiples

Ajouter des champs de données ici

Info-bulles

Ajouter des champs de données ici

Extraire

Interrapport

Garder tous les filtres

Données

Rechercher

annonces

_id

acco

amer

avail

avail

avail

avail

bath

bath

bedr

beds

calcu

calcu

calcu

calcu

calen

desc

first_

has_a

host_

host_

Page 1 sur 1

Mode de stockage : DirectQuery (cliquer pour changer)