******

***Compte rendu :***

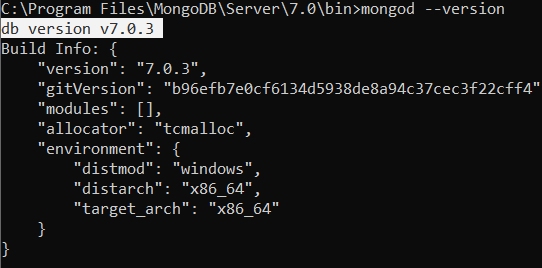
***TP MongoDB***

***Réalisé par :***

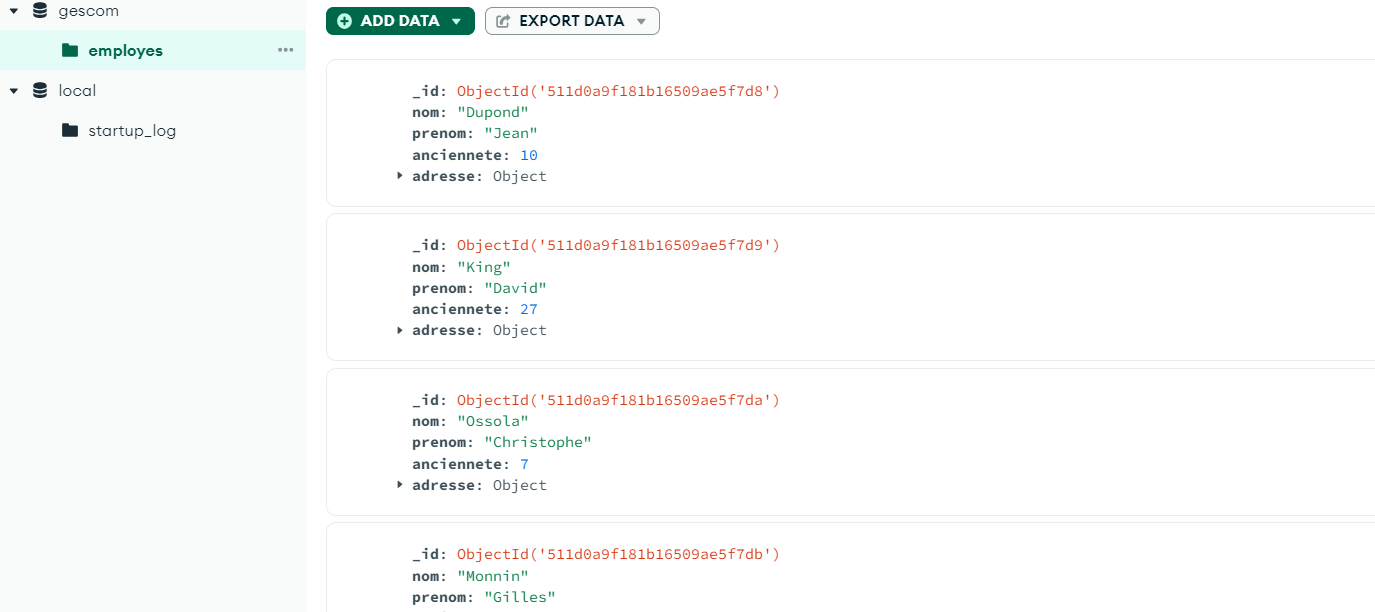
***Riahi Farah***

Objectifs du TP :

* On vérifie l’installation de mongo sur notre machine :

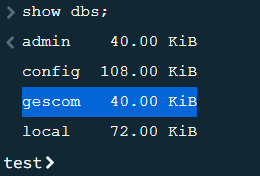
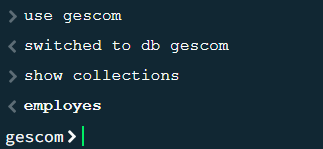


* » Importation de fichier « gescom.employes.json :



Exercice :

1. afficher toutes les collections de la base :

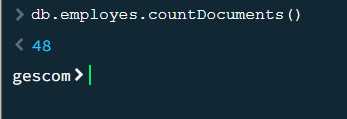
* On passe à la base gescom et on affiche les collections , en utilisant la commande « show collections »

2. afficher tous les documents de la base :



* Affichage liste des employés.

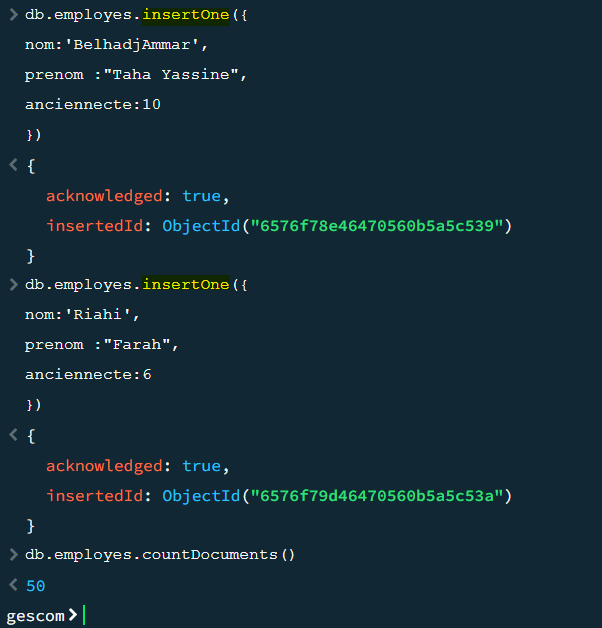
3. compter le nombre de documents de la collection employés :



* Les employés sont au nombre de 48.

4. insérer de deux manières différentes deux employés avec les champs nom, prénom et soit prime soit ancienneté :

- En utilisant une insertion un par un : insertOne



- En utilisant plusieurs insertions à la fois : insertMany



5. afficher la liste des employés dont le prénom est David :



* On utilise la méthode find() qui chercher tous les employés dont le nom est David
* Suite des employés :



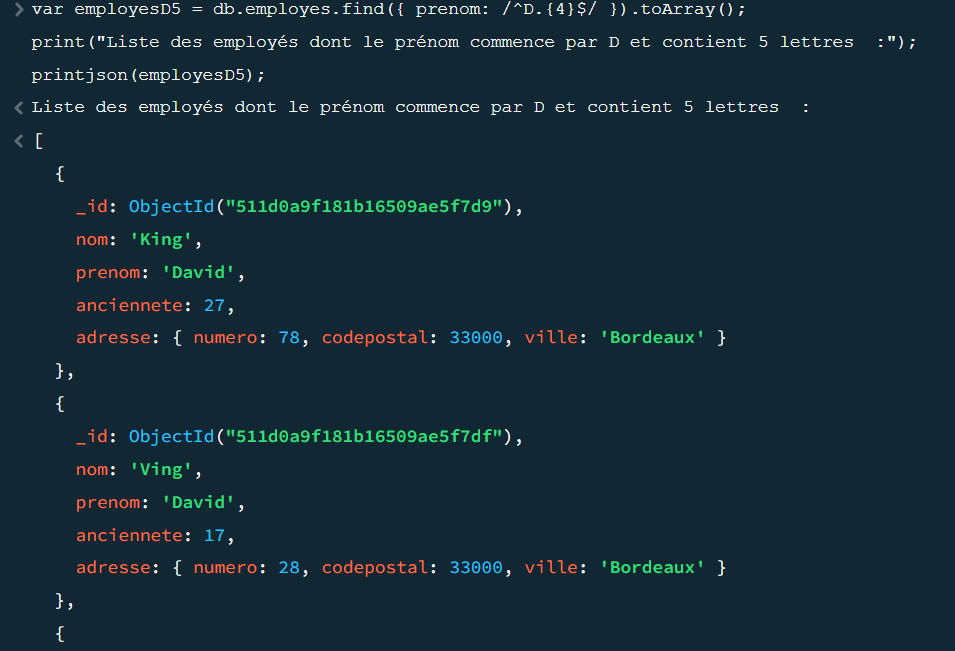
6. afficher la liste des employés dont le prénom commence ou se termine par D :



* On cherche à l’aide de l’expression régulière :
* /^ D : qui signifie commence par D
* | : ou
* D $/ : se termine par D
* Suite des employés :



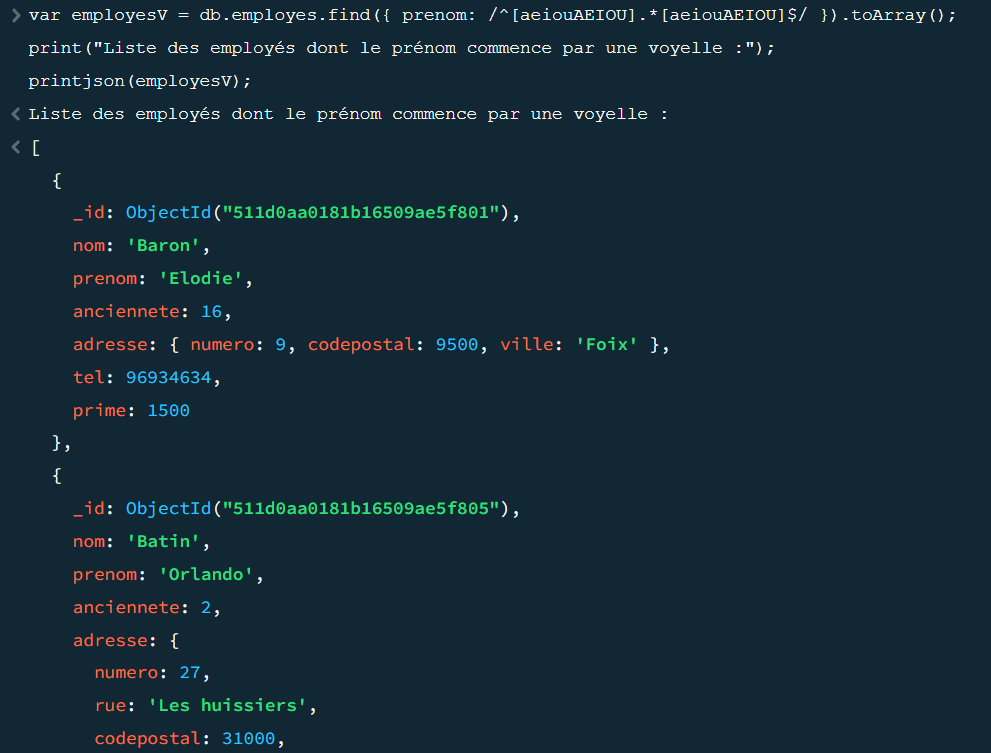
7. afficher la liste des personnes dont le prénom commence par D et contient exactement 5 lettres :



* On cherche à l’aide de l’expression régulière :
* /^ D : qui signifie commence par D
* **.** correspond à n'importe quel caractère (sauf un saut de ligne)
* **{4}** recherche exactement quatre caractères après la lettre "D"
* $/ : se termine
* Suite des employés :



8. afficher la liste des personnes dont le prénom commence et se termine par une voyelle :



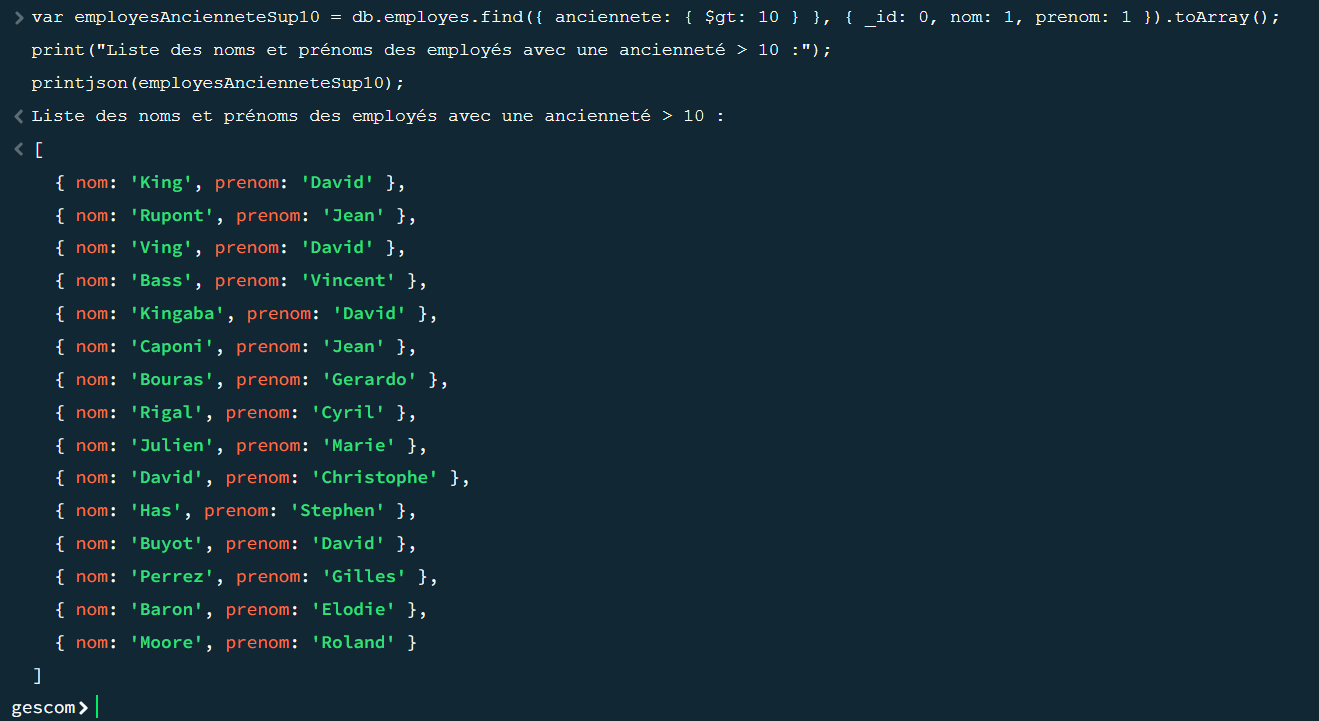
* On cherche à l’aide de l’expression régulière :
* /^ [aeiouAEIOU]: qui signifie commence par l’un des lettres aeiouAEIOU
* **.\*** correspond à n'importe quel caractère répété zéro ou plusieurs fois
* $/ : se termine

9. afficher la liste des personnes dont le prénom commence et se termine par une même lettre :



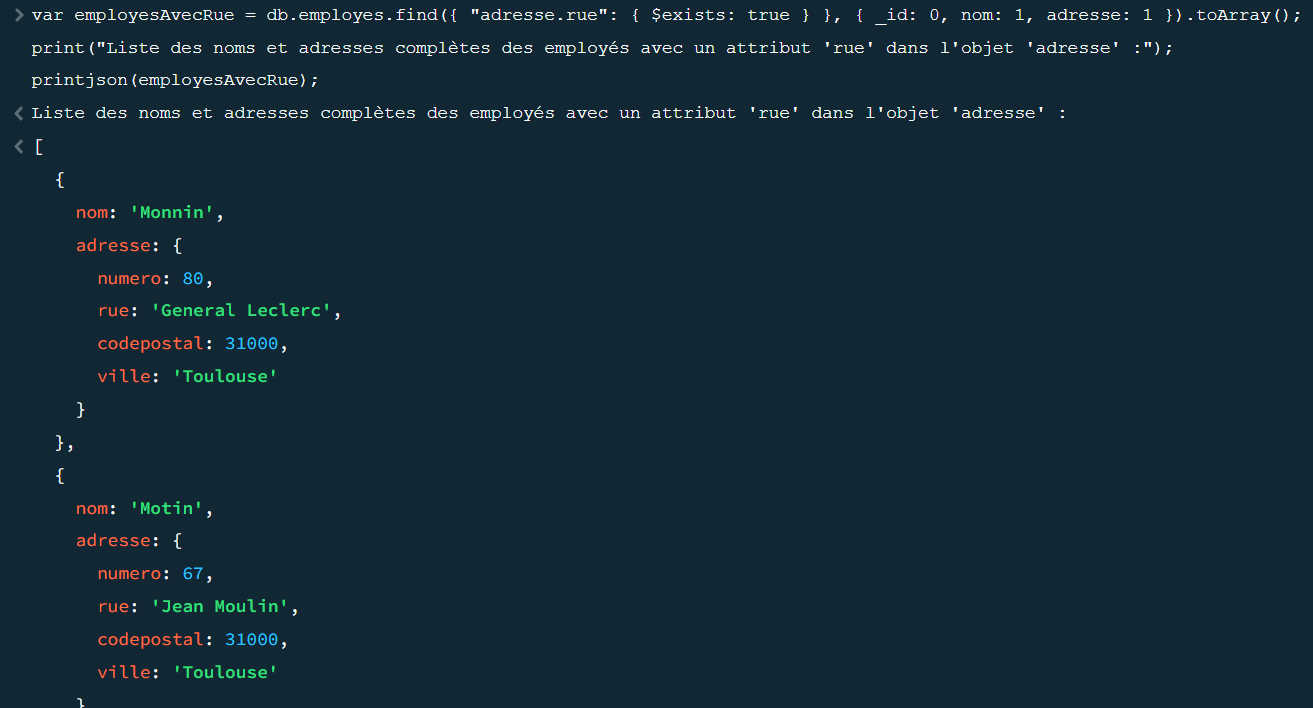
* **(\w)** : C'est une capture de groupe. **\w** correspond à n'importe quel caractère de mot (lettre, chiffre ou souligné).
* **1** : C'est une référence arrière au premier groupe de capture **(\w)**. Il fait correspondre le même caractère que celui capturé précédemment.
* **i** : C'est un indicateur d'option pour effectuer une correspondance insensible à la casse, ce qui signifie que la casse des lettres n'est pas prise en compte

10. afficher les nom et prénom de chaque employé ayant une ancienneté > 10 :

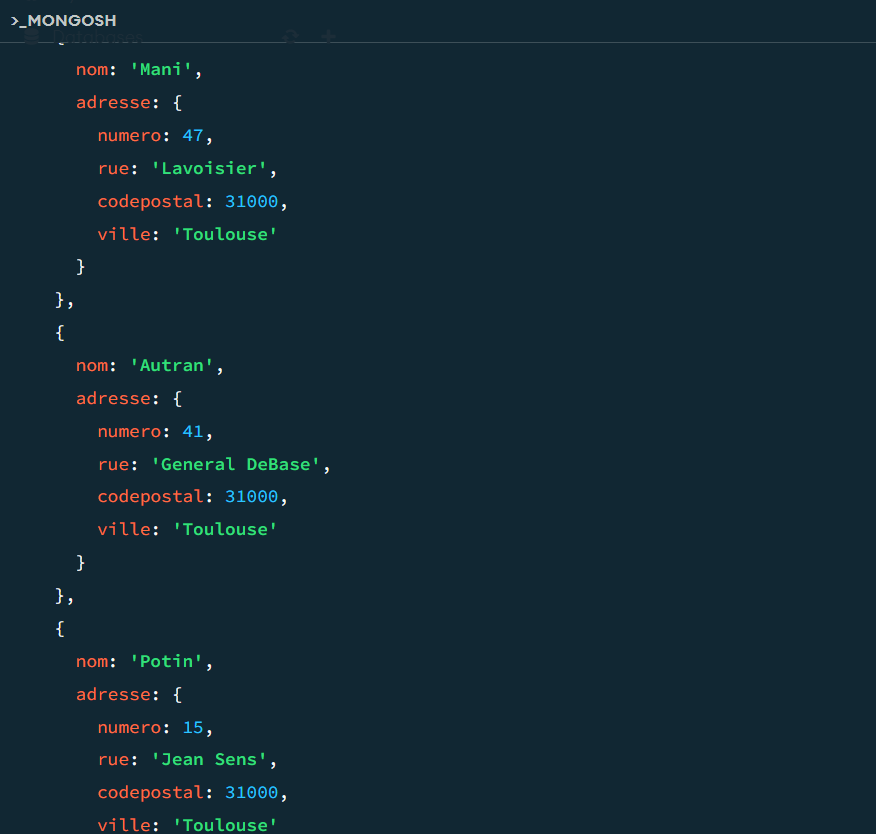


* **{ $gt: 10 } }** : C'est la condition de filtrage. **$gt : supérieur à**
* **{ \_id: 0, nom: 1, prenom: 1 }** : C'est la projection des champs. Cette partie spécifie quels champs vous souhaitez inclure dans les résultats : 0 : ne pas inclure ; 1 inclure

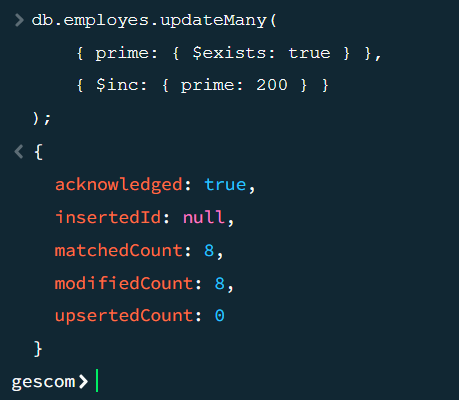
11. afficher les nom et adresse complète des employés ayant un attribut rue dans l’objet adresse :



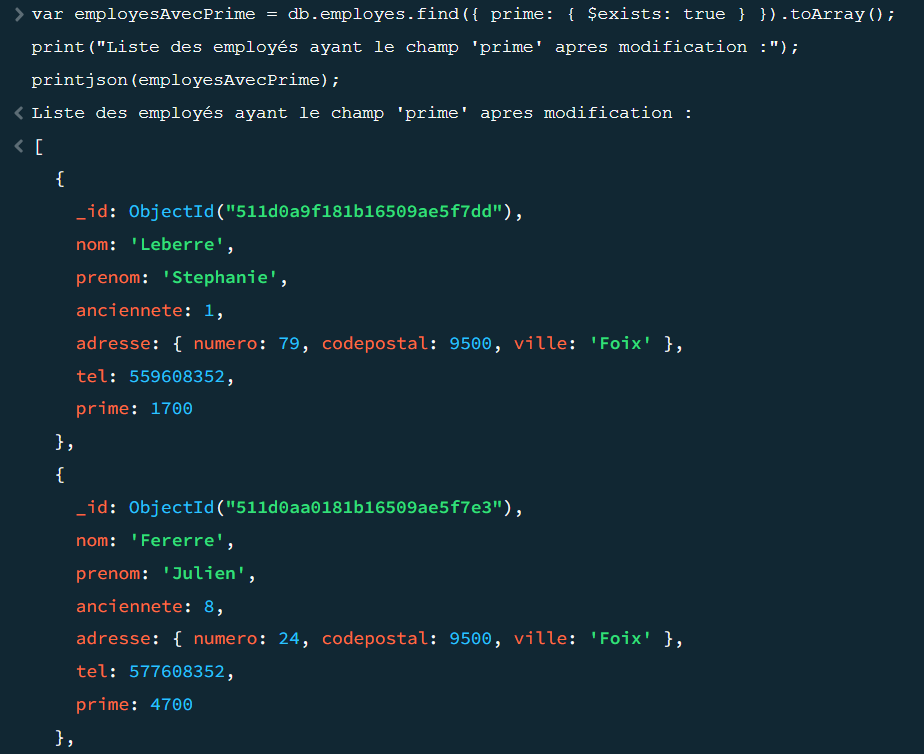
* $adresse.rue : $exists : true : signifie que l’attribut rue existe dans l’objet adresse .



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

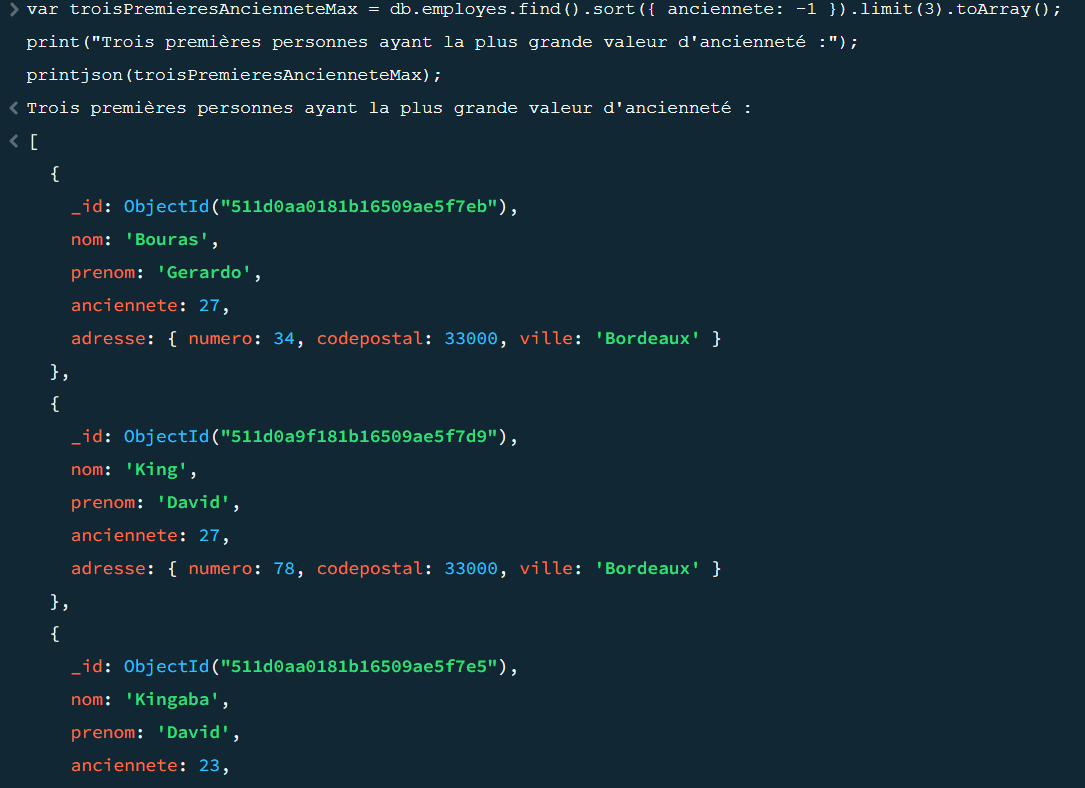


* $inc : incrementer
* Aperçu d’affichage après l’incrementation :



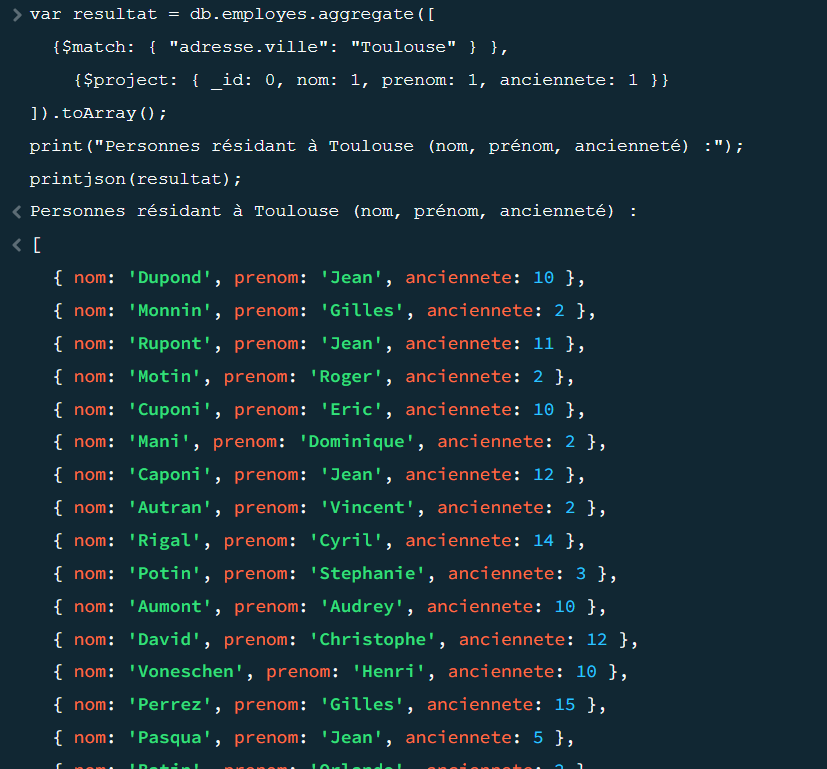


13. afficher les trois premières personnes ayant la plus grande valeur d’ancienneté :



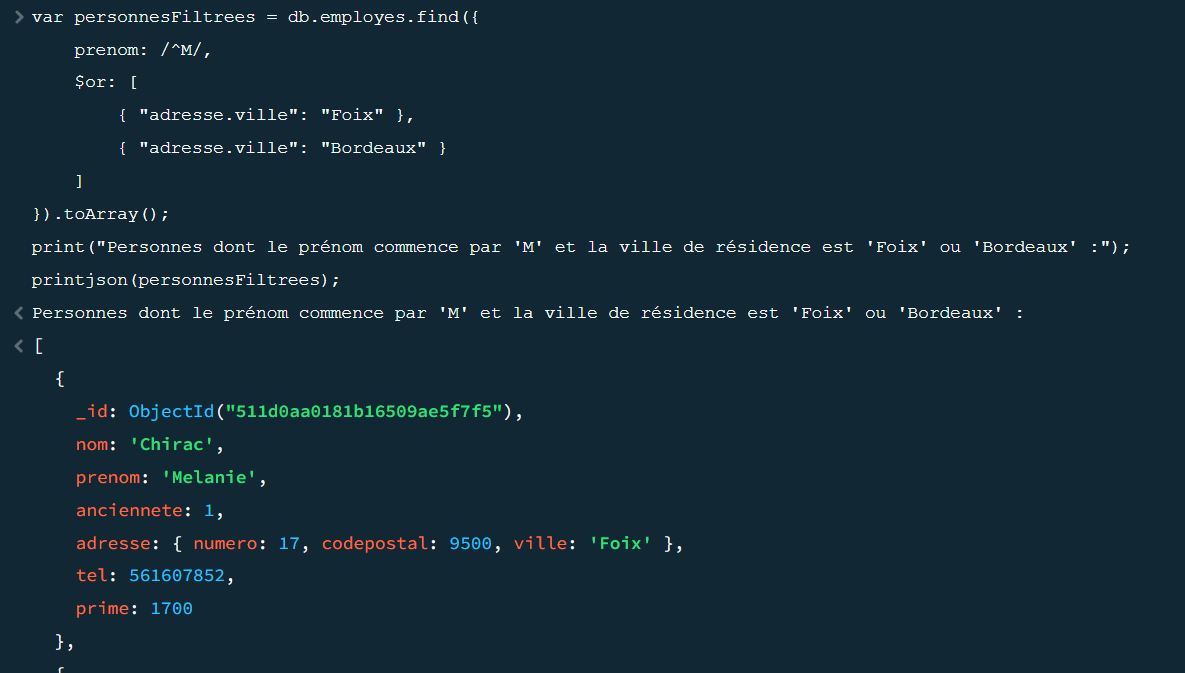
* **.sort({ anciennete: -1 })** : Trie les documents par ordre décroissant de l'ancienneté.
* **.limit(3)** : Limite les résultats aux trois premiers documents.

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



* L'utilisation de **aggregate()** est particulièrement puissante pour effectuer des opérations complexes sur les données, telles que le regroupement, le filtrage, la transformation, et d'autres opérations d'agrégation.
* **$match: { "adresse.ville": "Toulouse" }** : Filtre les documents pour inclure uniquement ceux dont la ville de résidence est Toulouse.
* **$project: { \_id: 0, nom: 1, prenom: 1, anciennete: 1 }** : Projette les champs que vous souhaitez afficher dans les résultats, tout en excluant l'identifiant **\_id**.

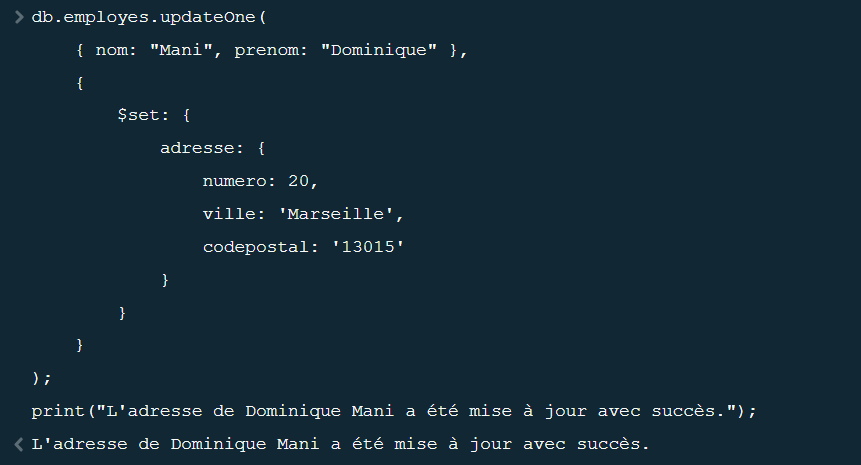
15. afficher les personnes dont le prénom commence par M et la ville de résidence est soit Foix soit Bordeaux :



* **prenom: /^M/** : Cela spécifie que le prénom doit commencer par la lettre "M". L'utilisation de **^** dans une expression régulière signifie "commence par".
* **$or: [...]** : Cela utilise l'opérateur logique **$or** pour spécifier que la ville de résidence doit être soit "Foix" soit "Bordeaux".
* **{ "adresse.ville": "Foix" }** et **{ "adresse.ville": "Bordeaux" }** : Ces expressions spécifient les deux conditions possibles pour la ville de résidence



16. mettre à jour l’adresse de Dominique Mani : nouvelle adresse ({ numéro : 20, ville : ’Marseille’, code postal : ’13015’ }) :



* **{ nom: 'Mani', prenom: 'Dominique' }** : C'est la condition pour sélectionner le document de Dominique Mani.
* **$set** : C'est l'opérateur pour définir de nouveaux champs ou mettre à jour des champs existants.
* **"adresse.numero": 20**, **"adresse.ville": 'Marseille'**, **"adresse.codepostal": '13015'** : Ce sont les nouvelles valeurs que vous souhaitez définir pour les champs de l'adresse.

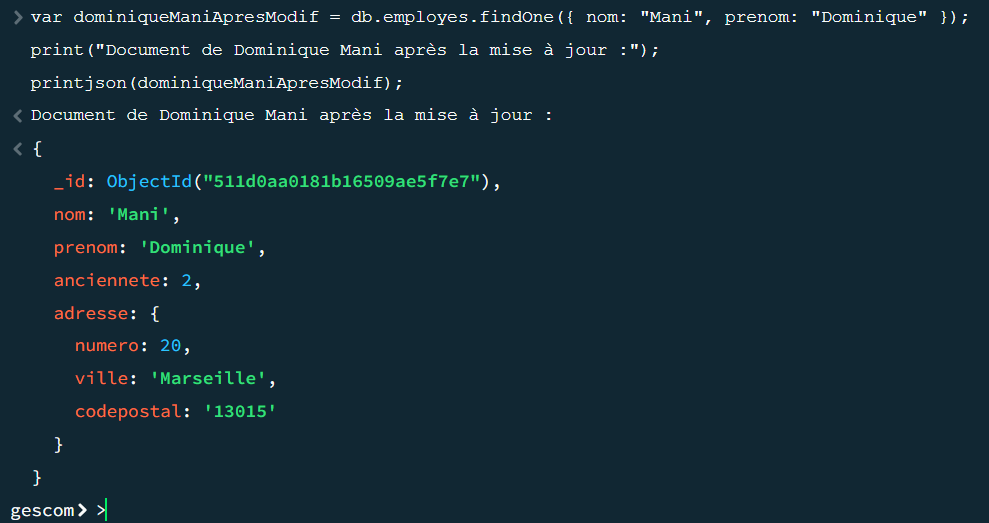
Affichage de l’employe Mani Dominique avant l’update :



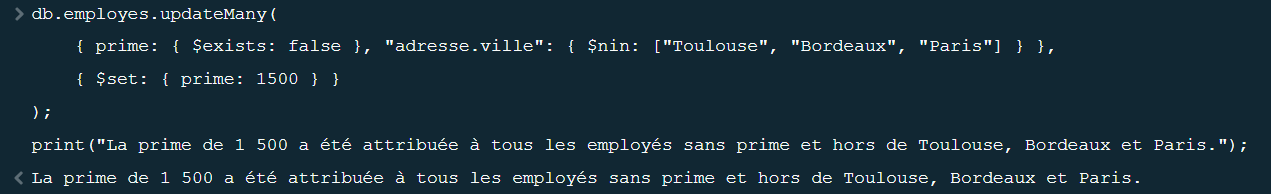
Affichage de l’employe Mani Dominique après l’update :



* Fonction d’affichage

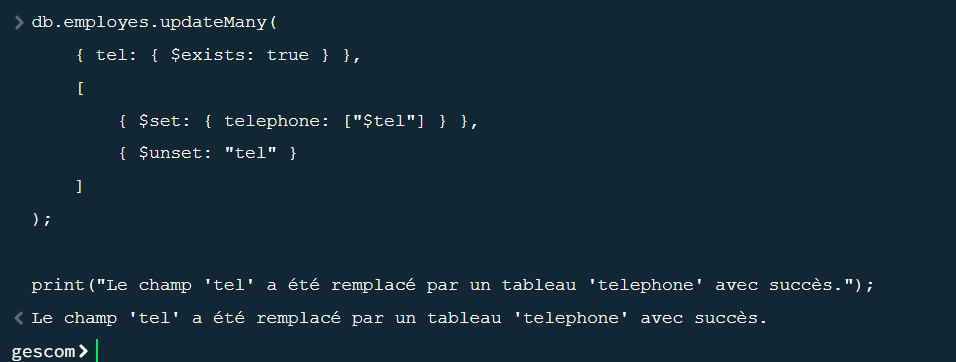


17. attribuer une prime de 1 500 `a tous les employés n’ayant pas de prime et dont la ville de résidence est différente de Toulouse, Bordeaux et Paris :



* **{ prime: { $exists: false } }** : C'est la condition pour sélectionner les employés qui n'ont pas de prime.
* **"adresse.ville": { $nin: ["Toulouse", "Bordeaux", "Paris"] }** : C'est la condition pour sélectionner les employés dont la ville de résidence n'est pas Toulouse, Bordeaux ou Paris.
* **$set: { prime: 1500 }** : C'est l'opération pour attribuer une prime de 1 500 € à ces employés.

18. remplacer le champ tel, pour les documents ayant un champ tel, par un tableau nommé téléphone contenant la valeur du champ tel :

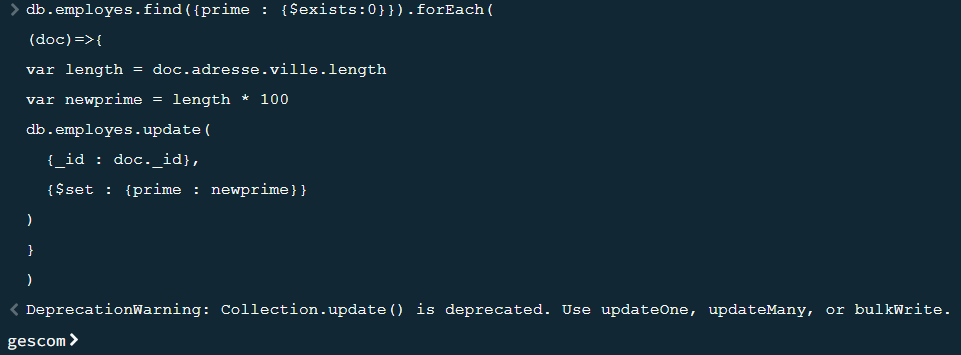


* **{ tel: { $exists: true } }** : C'est la condition pour sélectionner les documents qui ont un champ **tel**.
* **$set: { telephone: ["$tel"] }** : C'est l'opération pour créer un champ **telephone** contenant un tableau avec la valeur du champ **tel**.
* **$unset: { tel: "" }** : C'est l'opération pour supprimer le champ **tel** après avoir créé le champ **telephone**.

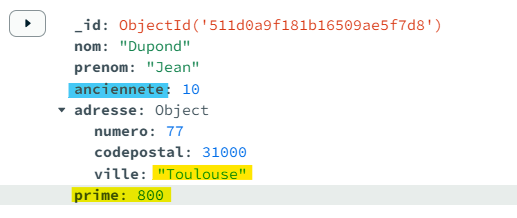
Voila un exmple d’affichage après la modification :



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 :

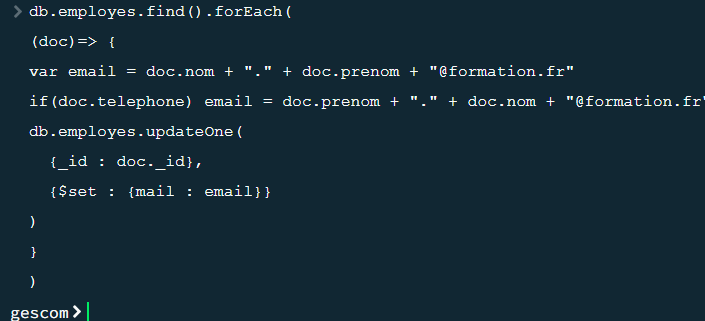


On vérifie avec un exmple :



* Voici un exemple d’un employé qui n’avait pas le champ prime, mais qui dispose maintenant une valeur de prime = length(« Toulouse »)\*100 = 8\*100 = 800.

20. créer un champ mail dont la valeur est égale soit à nom.prenom@formation.fr pour les employés ne disposant pas d’un champ ttéléphone, soit à prenom.nom@formation.fr (nom et prénom sont à remplacer par les vraies valeurs de chaque employé) :





* Voilà un affichage.

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

