**Rapport**

**Introduction**

Ce projet a pour objectif de manipuler une base de données NoSQL MongoDB en utilisant le langage Python avec le module PyMongo.

Il s’agit d’une mini-application e-commerce qui permet de gerer les informations sur les produits, les clients, et les commandes.

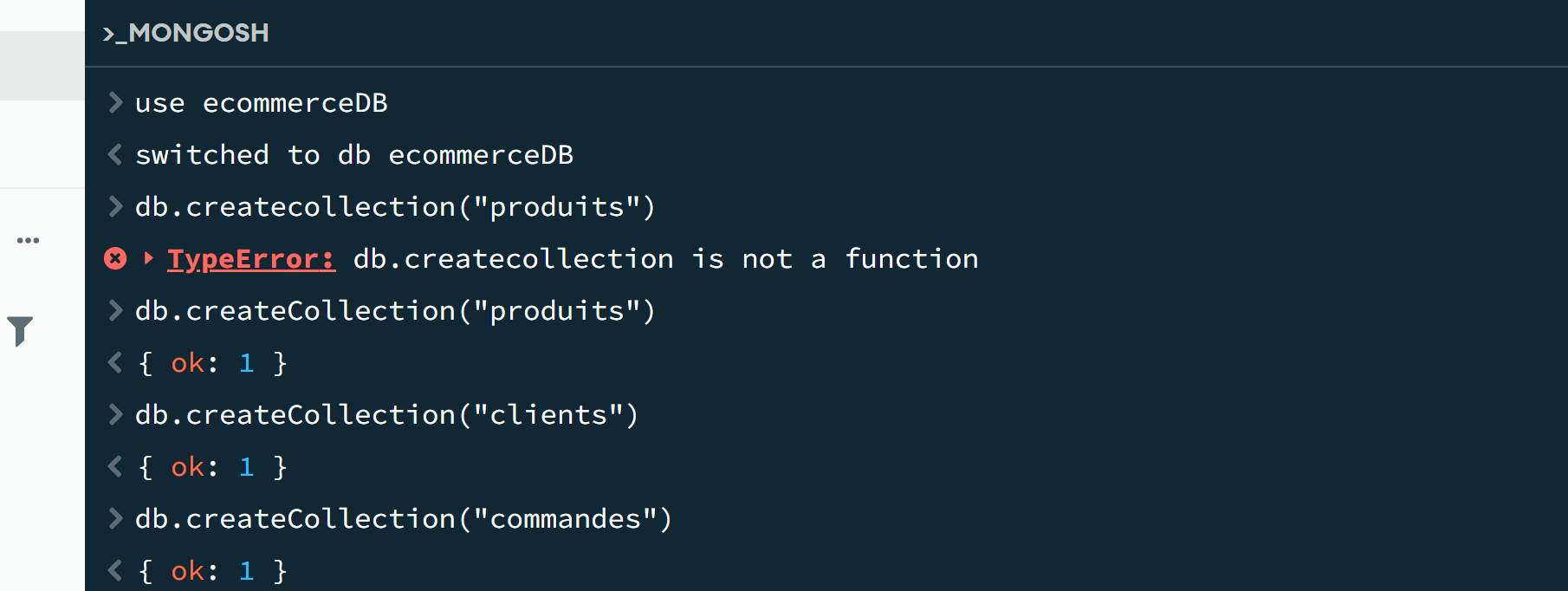
A travers ce travail, on apprend àaeffectuer differentes operations comme l’ajout, la mise a jour, la suppression, et la recherche des donnees dans MongoDB, tout en appliquant les notions vues en cours.

1. Créer les collections : produits, clients et commandes dans une base de données

**ecommerceDB**.

On crée la base de données ecommerceDB et trois collections :

Produits-clients-commandes



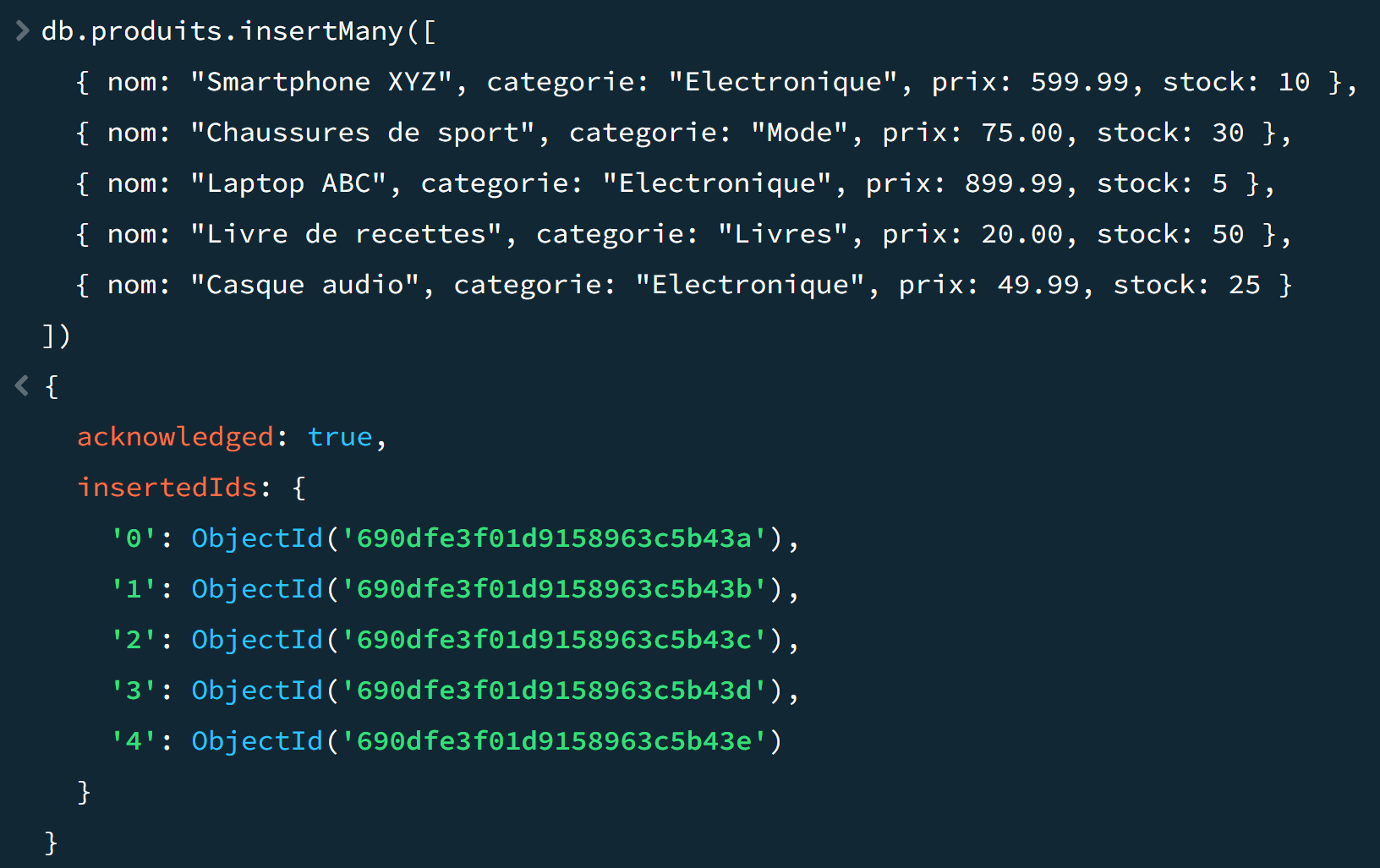
1. Ajouter dans la base de donnée

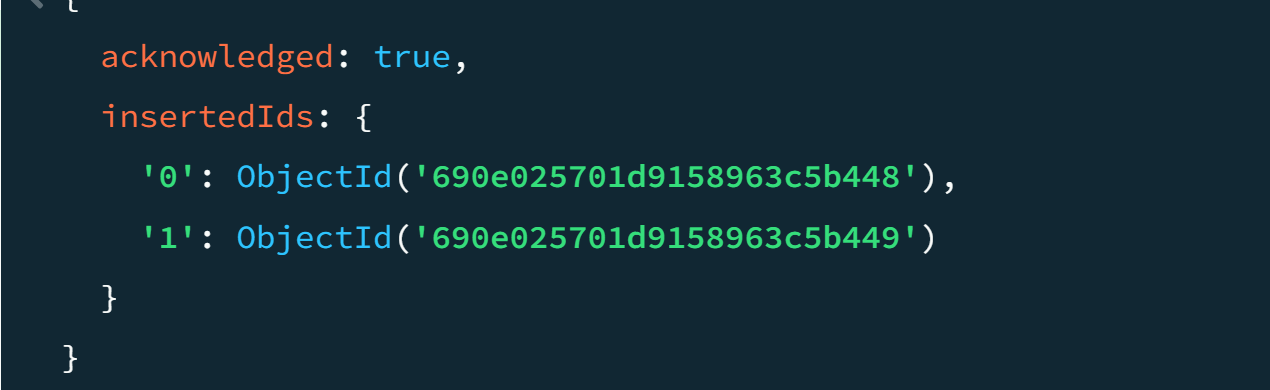
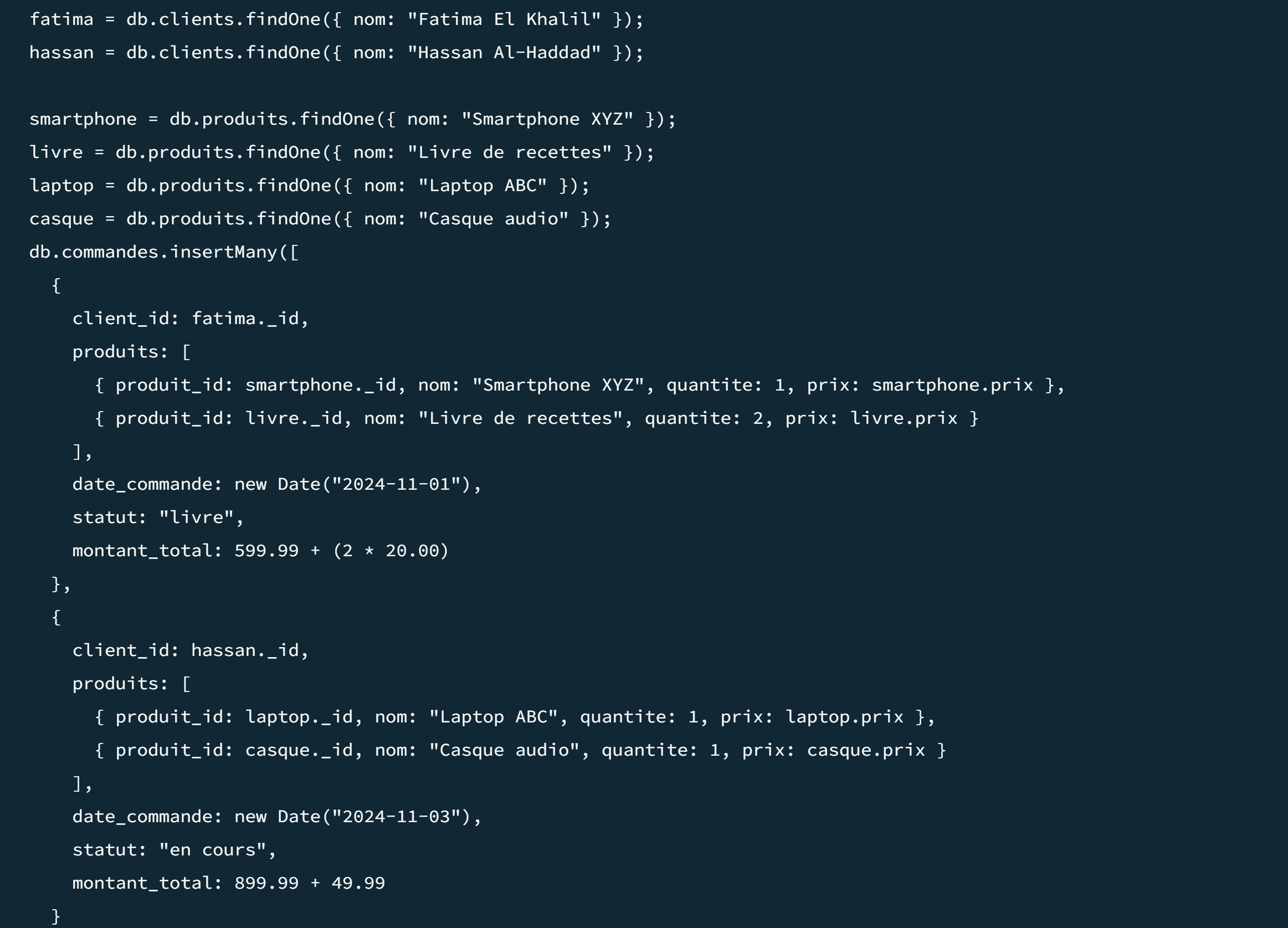
Dans cette etape, nous avons insere les donnees initiales dans les trois collections principales de la base ecommerceDB :Produits- Clients-Commandes

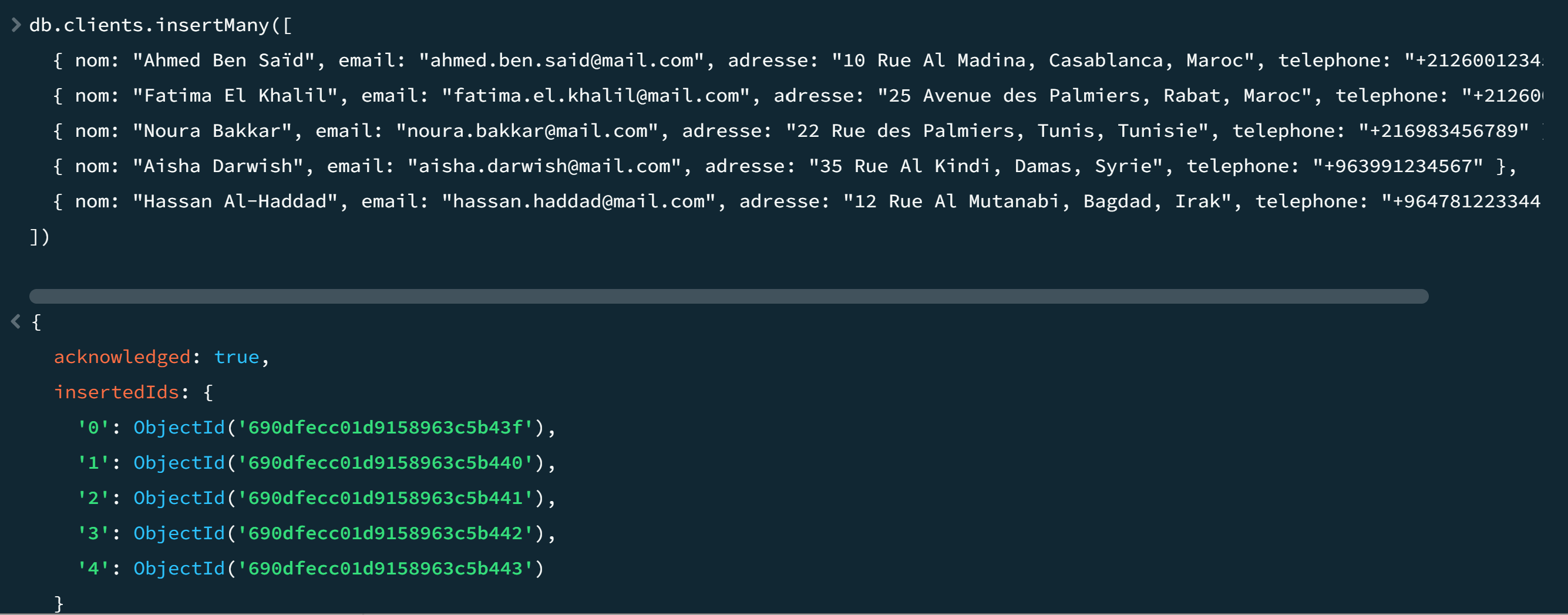
Chaque collection contient des documents structures selon les attributs specifies

Les commandes font référence a des clients et à des produits existants, ce qui permet de creer des relations logiques entre les collections.L’insertion des données a ete realisee a l’aide de la commande :

**db.collection.insertMany**







1. Afficher tous les produits en stock.

L’objectif de cette étape est d’afficher tous les produits disponibles dans le stock.

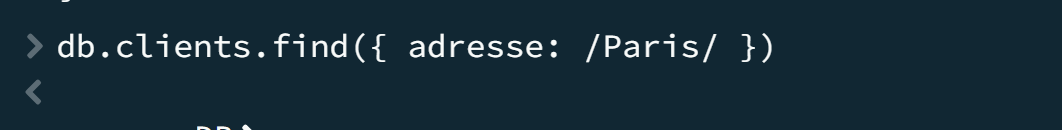
Pour cela, une requête de type find() a été utilisée sur la collection produits avec une condition sur le champ stock



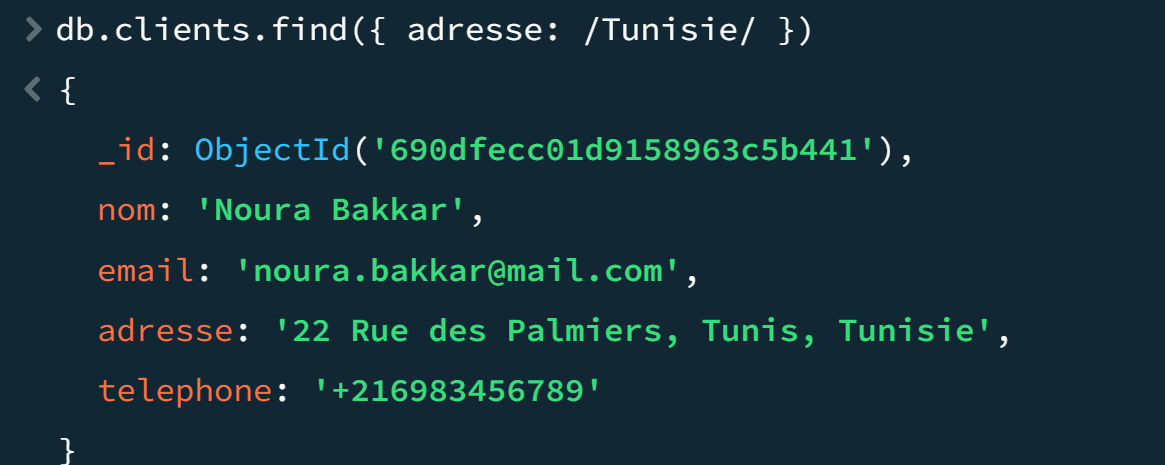
1. Rechercher tous les clients dont l'adresse contient "Paris".

L’objectif de cette etape est de trouver tous les clients dont l’adresse contient le mot “Paris”.

Pour cela, on effectue une recherche dans la collection clients a l’aide d’un filtre utilisant l’opérateur $regex.



Exemple



1. Afficher toutes les commandes d’un client particulier (par exemple, Alice Dupont).

L’objectif de cette question est d’afficher toutes les commandes passées par un client particulier (par exemple : Alice Dupont ou Fatima El Khalil).



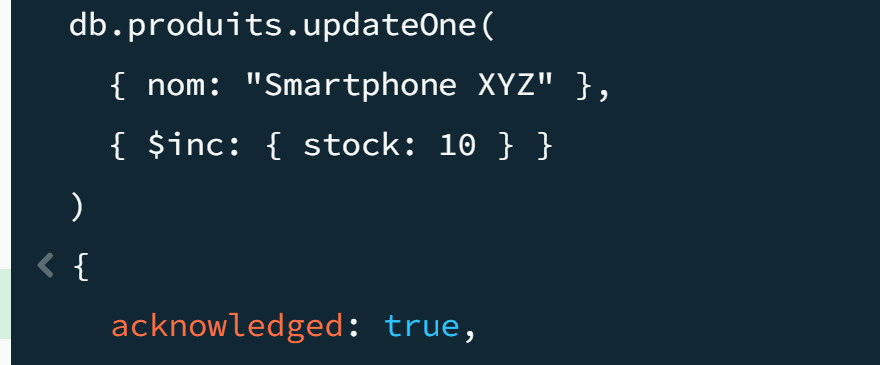
1. Rechercher toutes les commandes livrées.

L’objectif de cette question est d’afficher uniquement les commandes dont le statut est "livree" dans la collection commandes.



1. Le stock du produit "Smartphone XYZ" doit être mis à jour suite à une nouvelle livraison (ajoutez 10 unités de stock).

Augmenter le stock du produit "Smartphone XYZ" suite a une nouvelle livraison de 10 unites.



8. Modifier le statut de la commande de Hassan Al-Haddad pour passer de "en cours" à

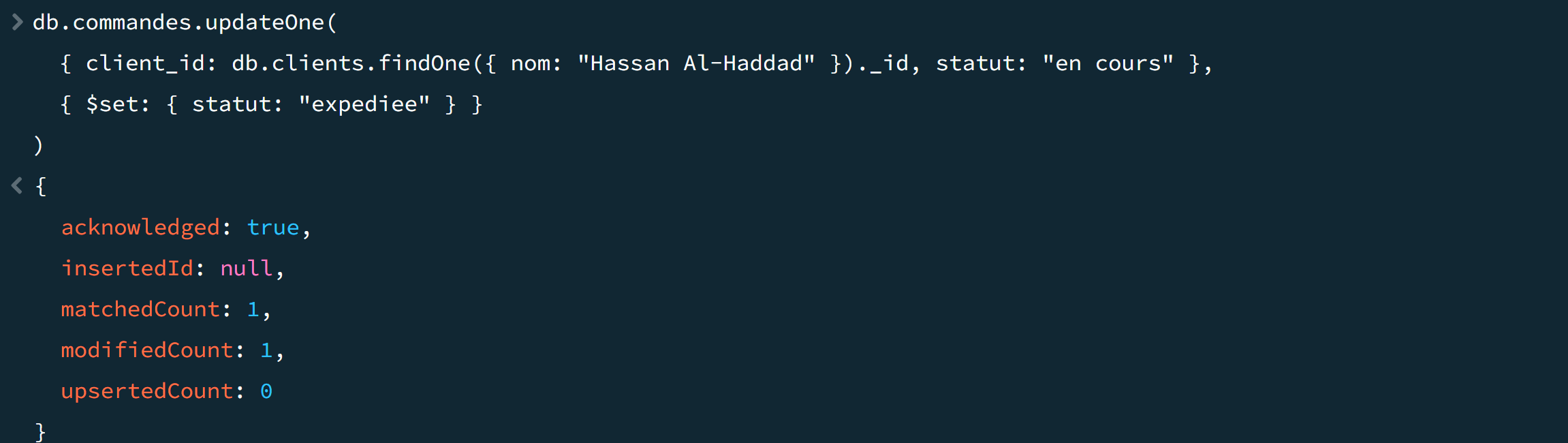
"expédiée".

Objectif :

Changer le statut de la commande de "Hassan Al-Haddad" de "en cours" à "expédiée".

updateOne() permet de mettre à jour le premier document correspondant aux critères donnés.

On utilise $set pour modifier uniquement le champ statut sans toucher aux autres champs.





1. Supprimer un produit qui est épuisé (par exemple, un produit dont le stock est égal à 0).

Objectif :

Supprimer de la collection produits tout produit dont le stock est égal à 0.

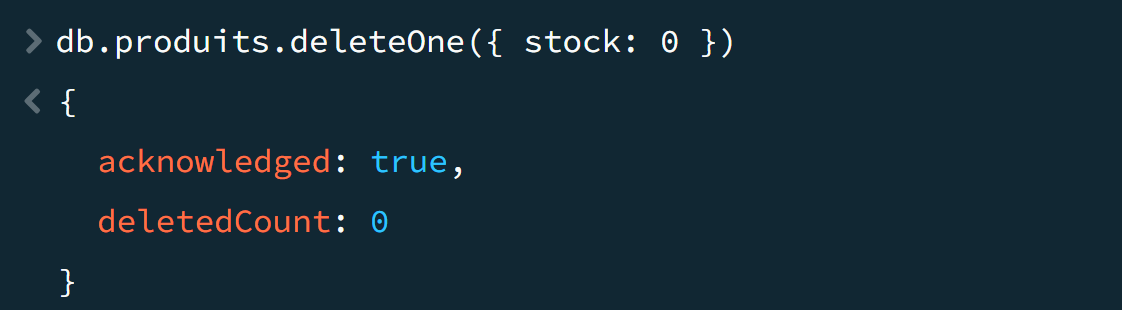
Commande MongoDB :

db.produits.deleteOne({ stock: 0 })

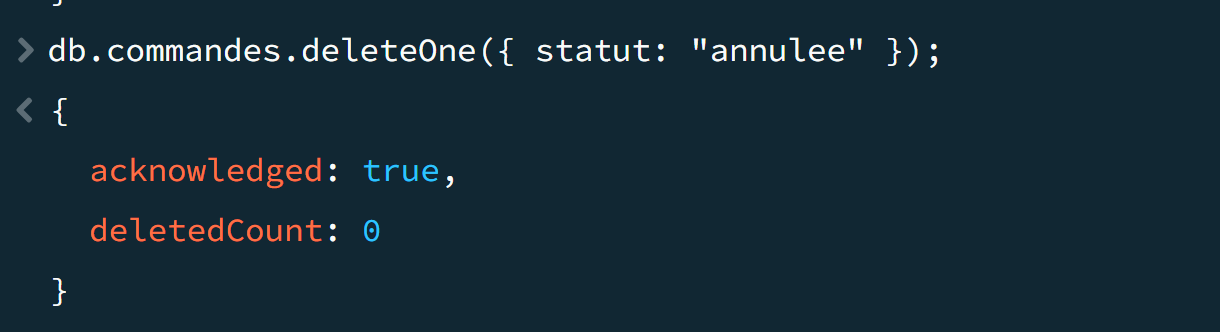
Explication :

deleteOne() supprime le premier document qui correspond au critère { stock: 0 }.

Si plusieurs produits sont épuisés et qu’on veut tous les supprimer, on peut utiliser deleteMany({ stock: 0 }).



1. Supprimer une commande annulée.



Objectif :

Supprimer une commande specifique qui a ete annulee de la collection commandes.

Commande :

db.commandes.deleteOne({ statut: "annulee" })

Explication :

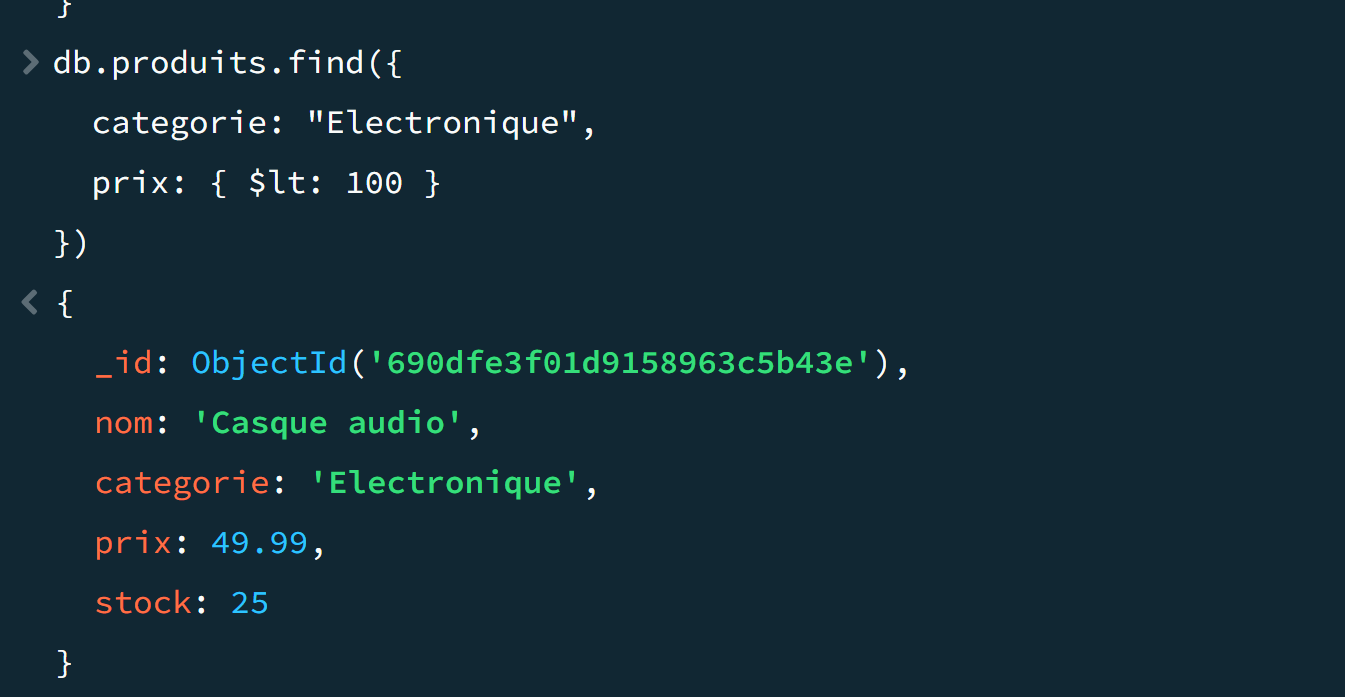
deleteOne() supprime la première commande trouvee avec le statut "annulee".

Si vous voulez supprimer toutes les commandes annulées, on utilise deleteMany({ statut: "annulée" }).

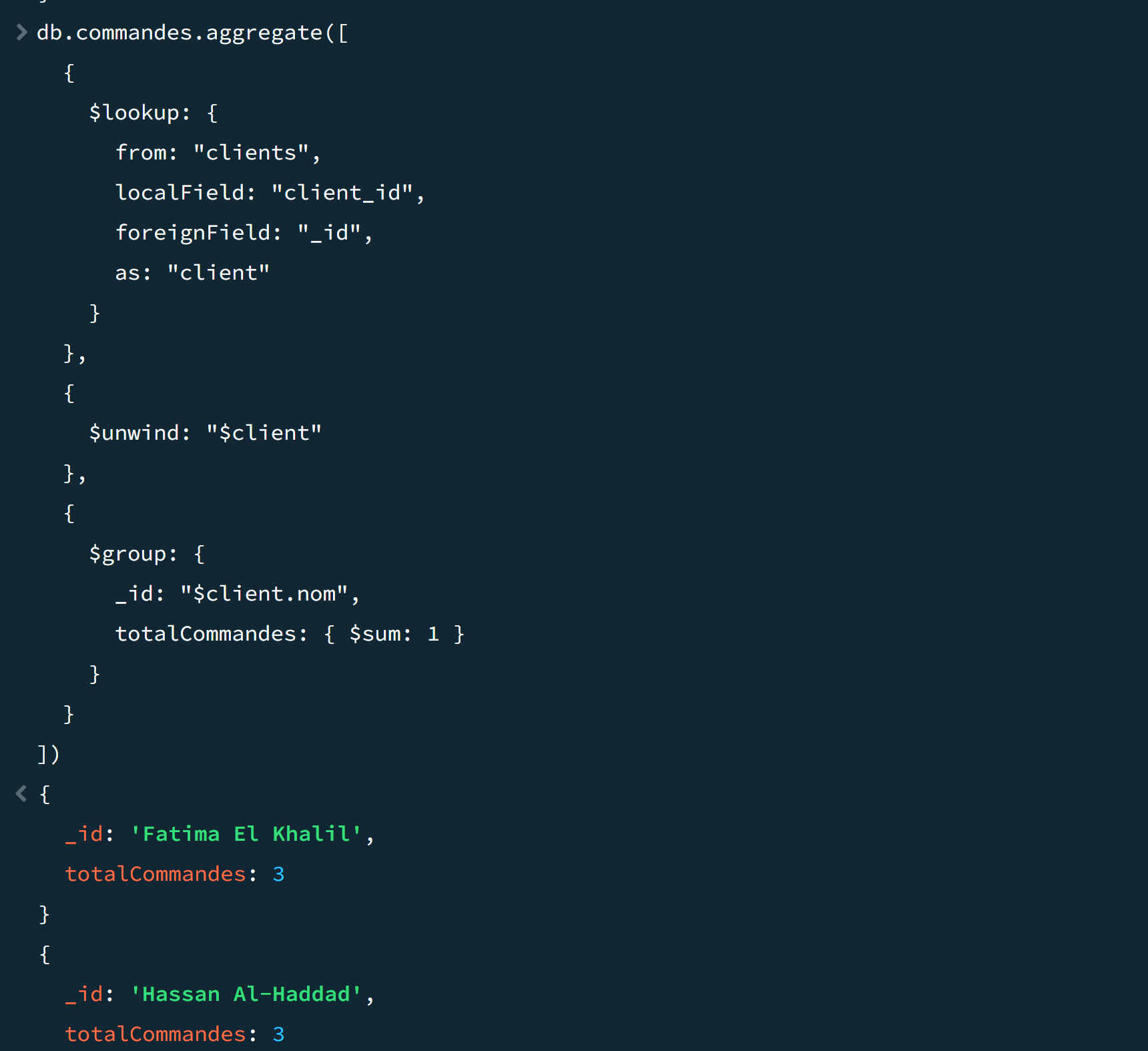
1. Trouver tous les produits de la catégorie "Électronique" dont le prix est inférieur à 100.

Objectif :

Identifier tous les produits appartenant a la categorie "Électronique" et dont le prix est inferieur a 100. Cela permet de filtrer les produits bon marche dans une categorie specifique.



1. Compter le nombre total de commandes passées par chaque client.



Objectif :

Savoir combien de commandes chaque client a passé. Cela permet de suivre l’activité des clients et de faire des statistiques sur les achats.

Remarque

Le total de commande pour chaque client et 3 parce que j'ai insere trois commandes par chaque client.

1. Afficher les 5 produits les plus chers dans la collection.

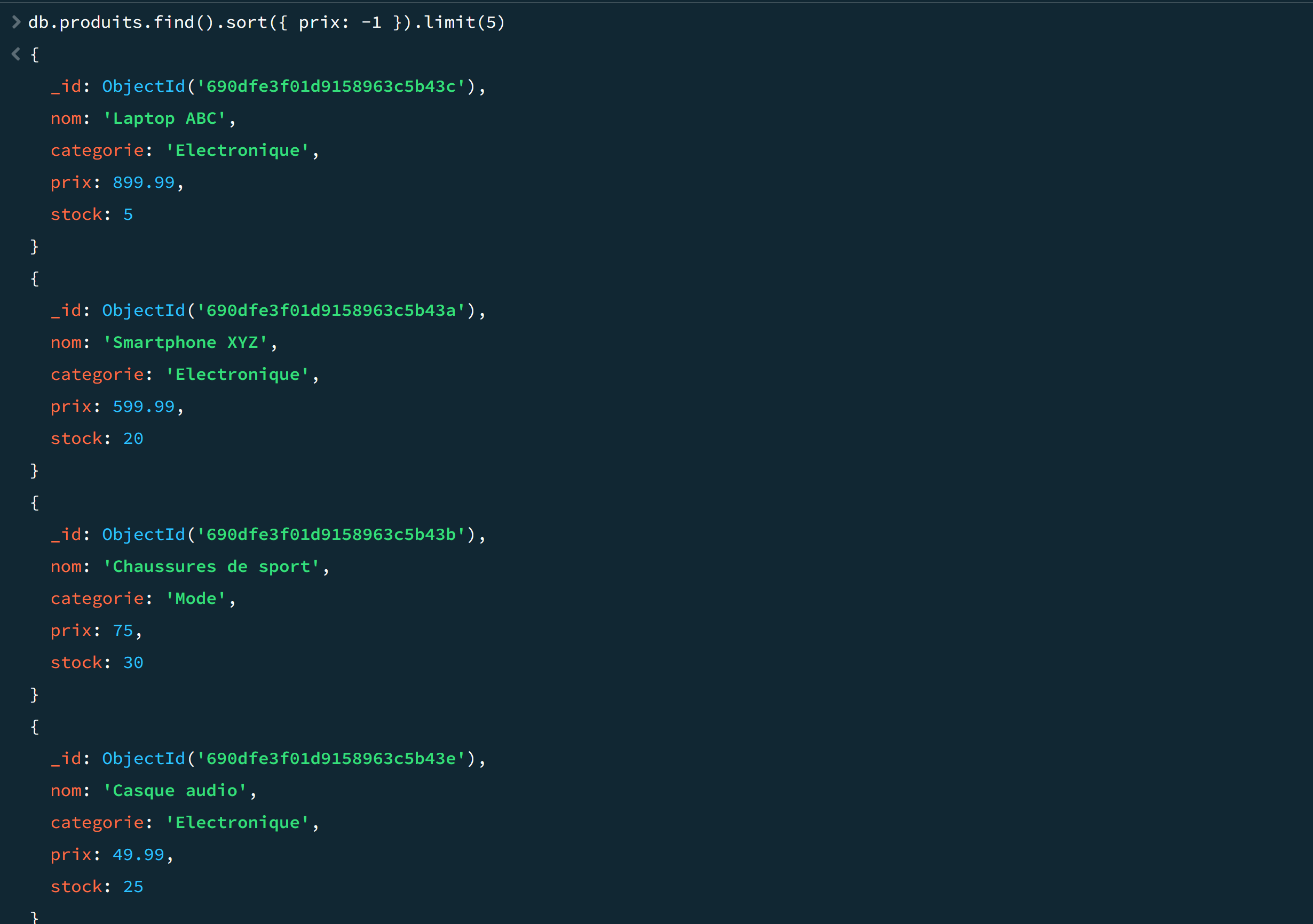
Identifier les produits les plus chers pour mieux gérer les ventes et les promotions.

Comment faire avec MongoDB :

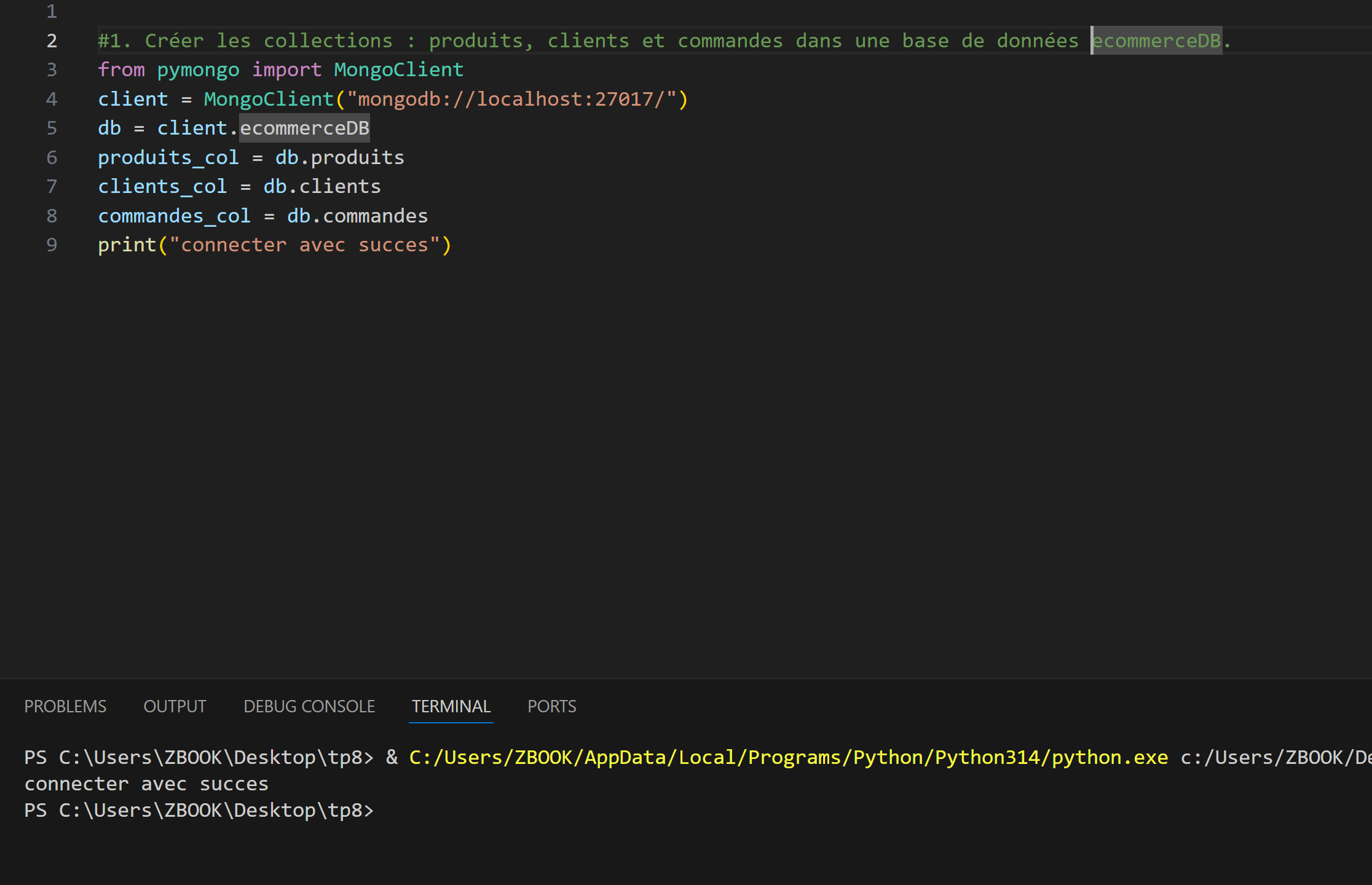
Utiliser find() pour récupérer tous les produits.

Trier avec sort({ prix: -1 }) pour avoir les prix du plus élevé au plus bas.

Limiter le résultat à 5 avec limit



***Partie II : Pymongo***



1. Créez une commande

Permettre à l’utilisateur d’ajouter une nouvelle commande dans la base de données ecommerceDB via Python.

Explication simple :

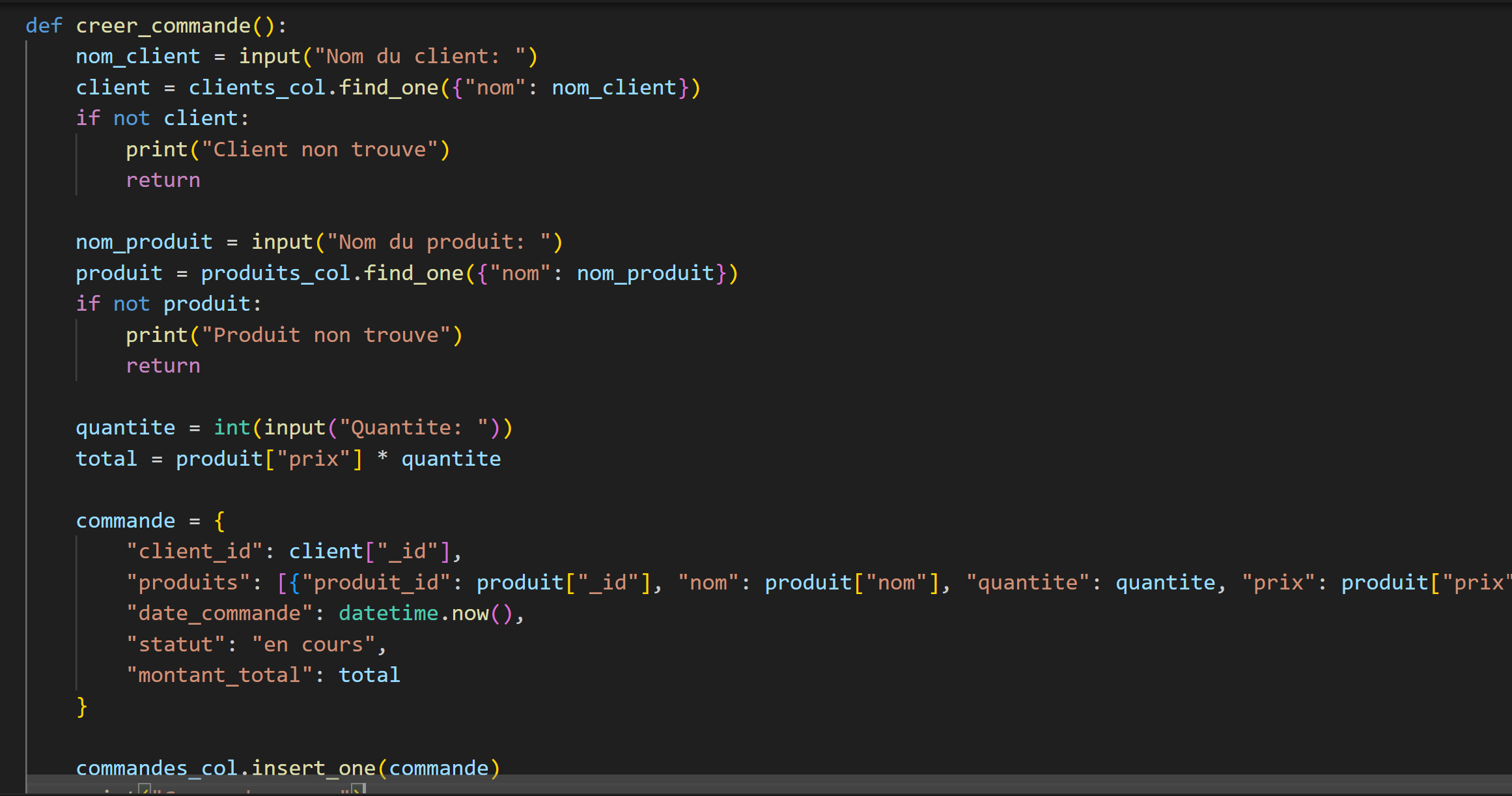
L’utilisateur saisit le nom du client.

Le script vérifie si le client existe dans la collection clients.

L’utilisateur saisit les produits souhaités et les quantités.

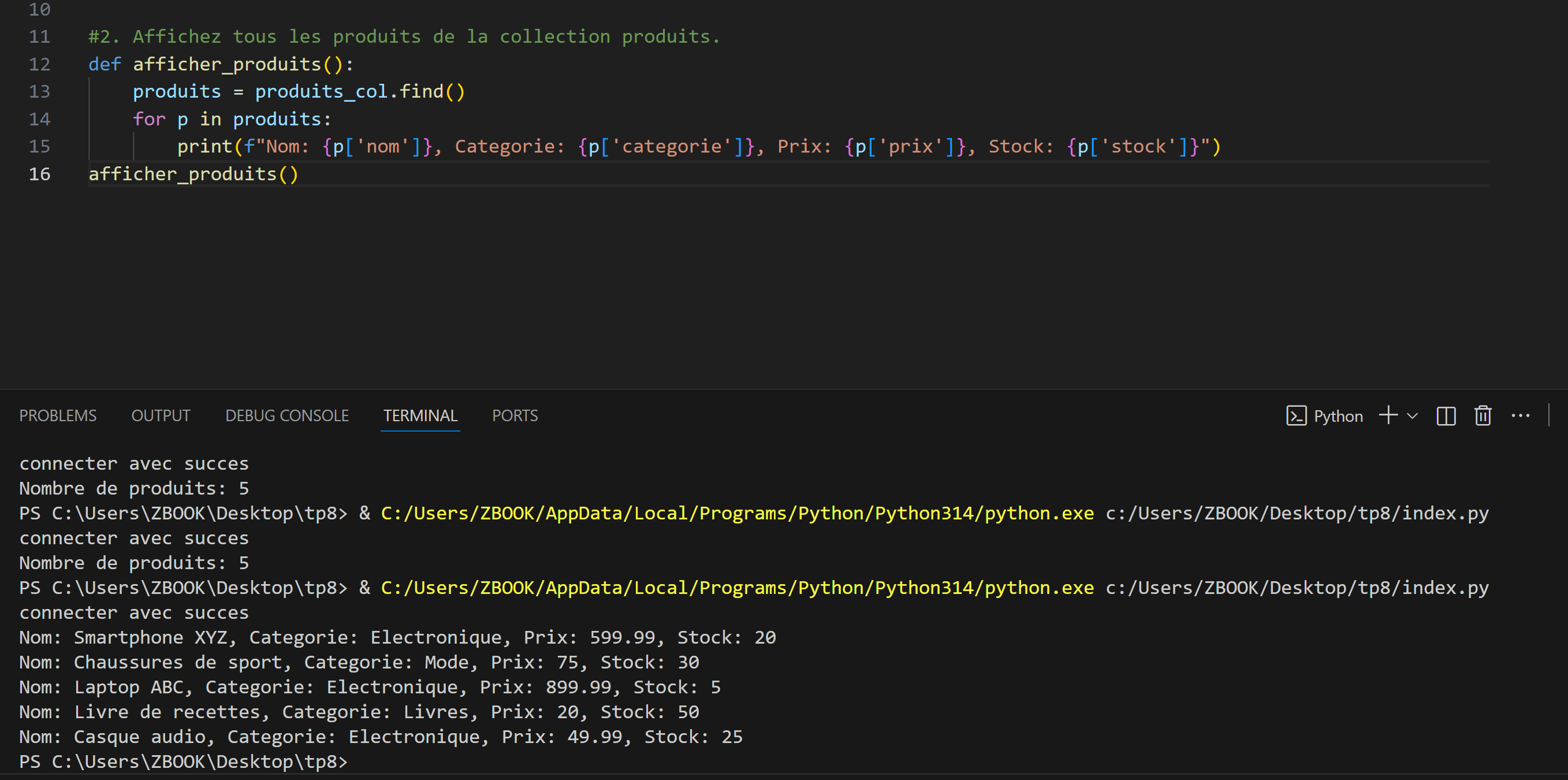
Le script calcule le montant total de la commande.

La commande est ajoutée à la collection commandes avec la date du jour et le statut "en cours".









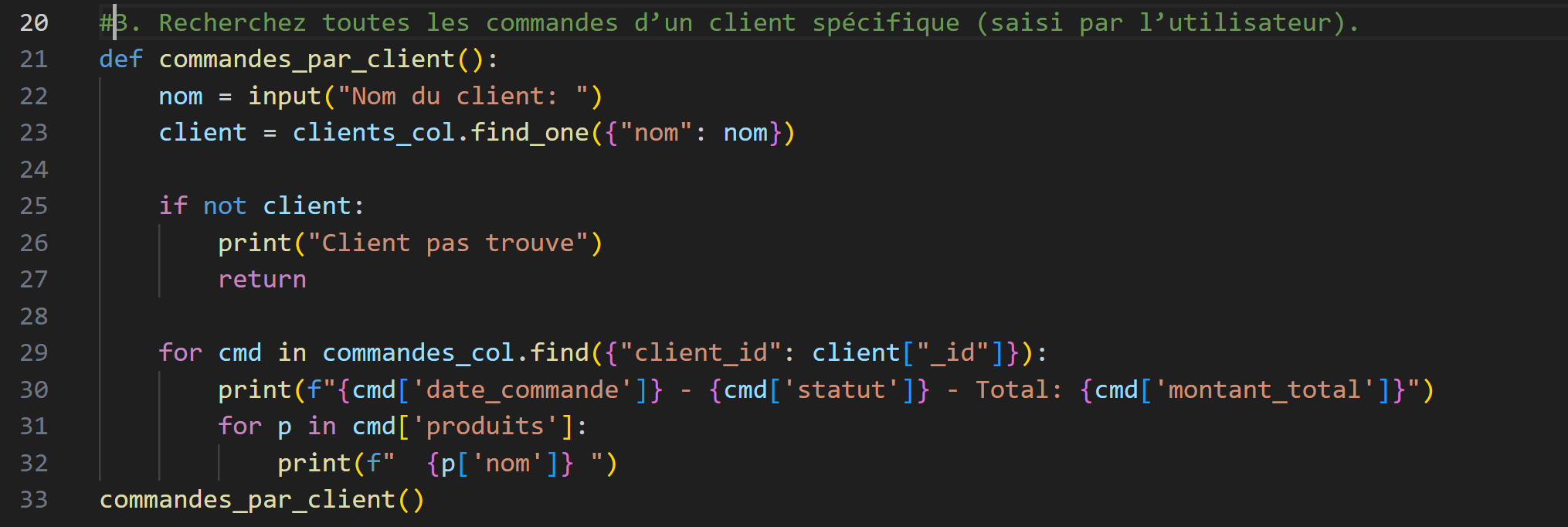
Objectif :

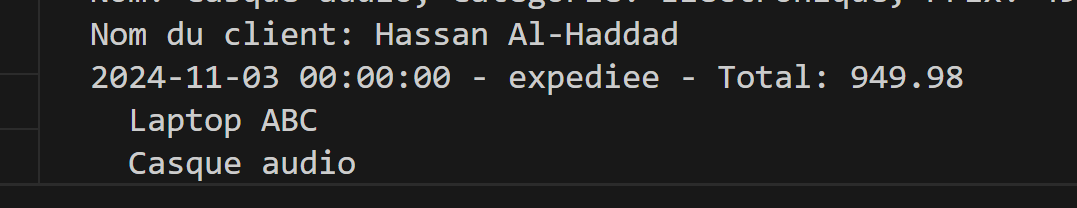
Permettre à l’utilisateur de visualiser tous les produits stockes dans la base ecommerceDB.

Explication :

On recupere tous les documents de la collection produits.

Pour chaque produit, on affiche ses attributs : nom, categorie, prix et stock disponible.





Objectif :

Permettre a l’utilisateur de voir toutes les commandes passees par un client donne.

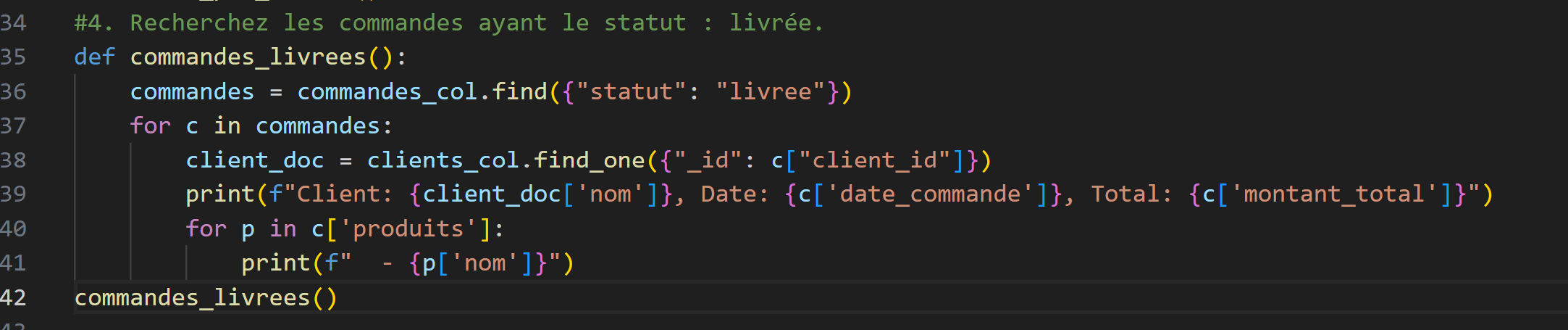
Explication :

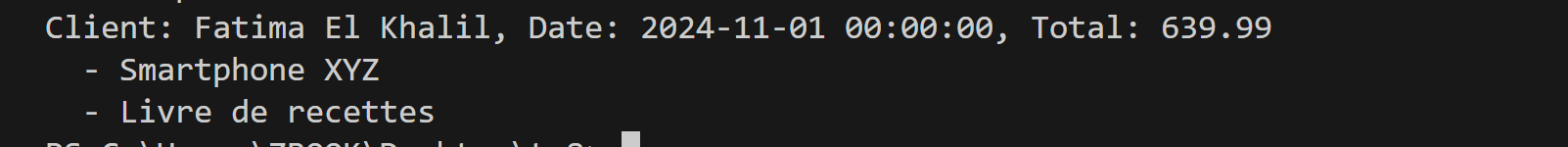
On demande à l’utilisateur de saisir le nom du client.

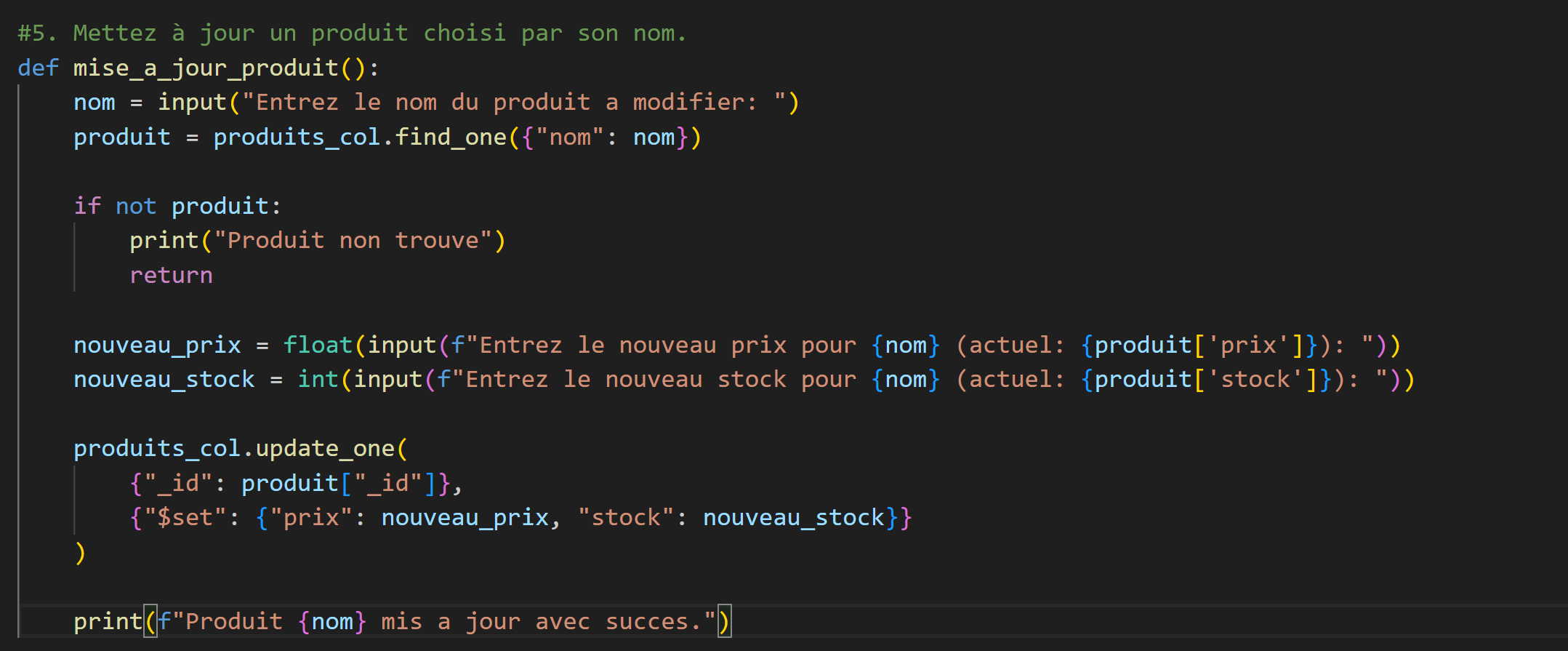
On recherche le document correspondant dans la collection clients.

Ensuite, on racupere toutes les commandes dans commandes où client\_id correspond a l’ID du client.

Pour chaque commande, on affiche la date, le statut, le montant total et les details des produits commandes.







Objectif :

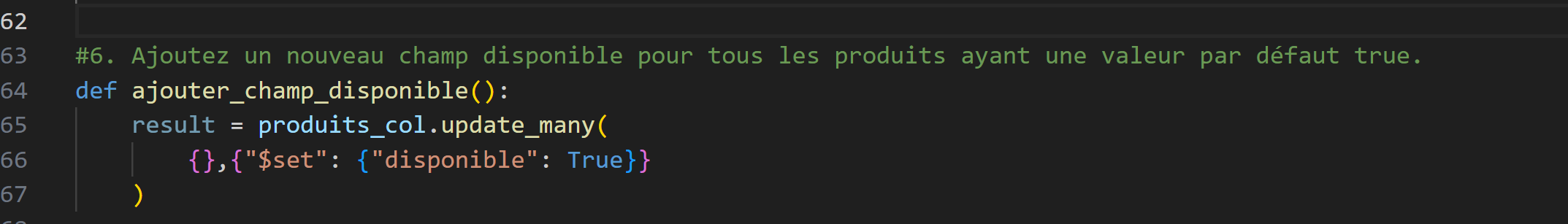
Lister toutes les commandes qui ont ete livrees afin de suivre les commandes terminees.

Explication :

On filtre la collection commandes avec statut = "livree".

Pour chaque commande, on récupère le nom du client grace a client\_id.

On affiche les informations importantes : nom du client, date de commande, montant total et details des produits.



Objectif :

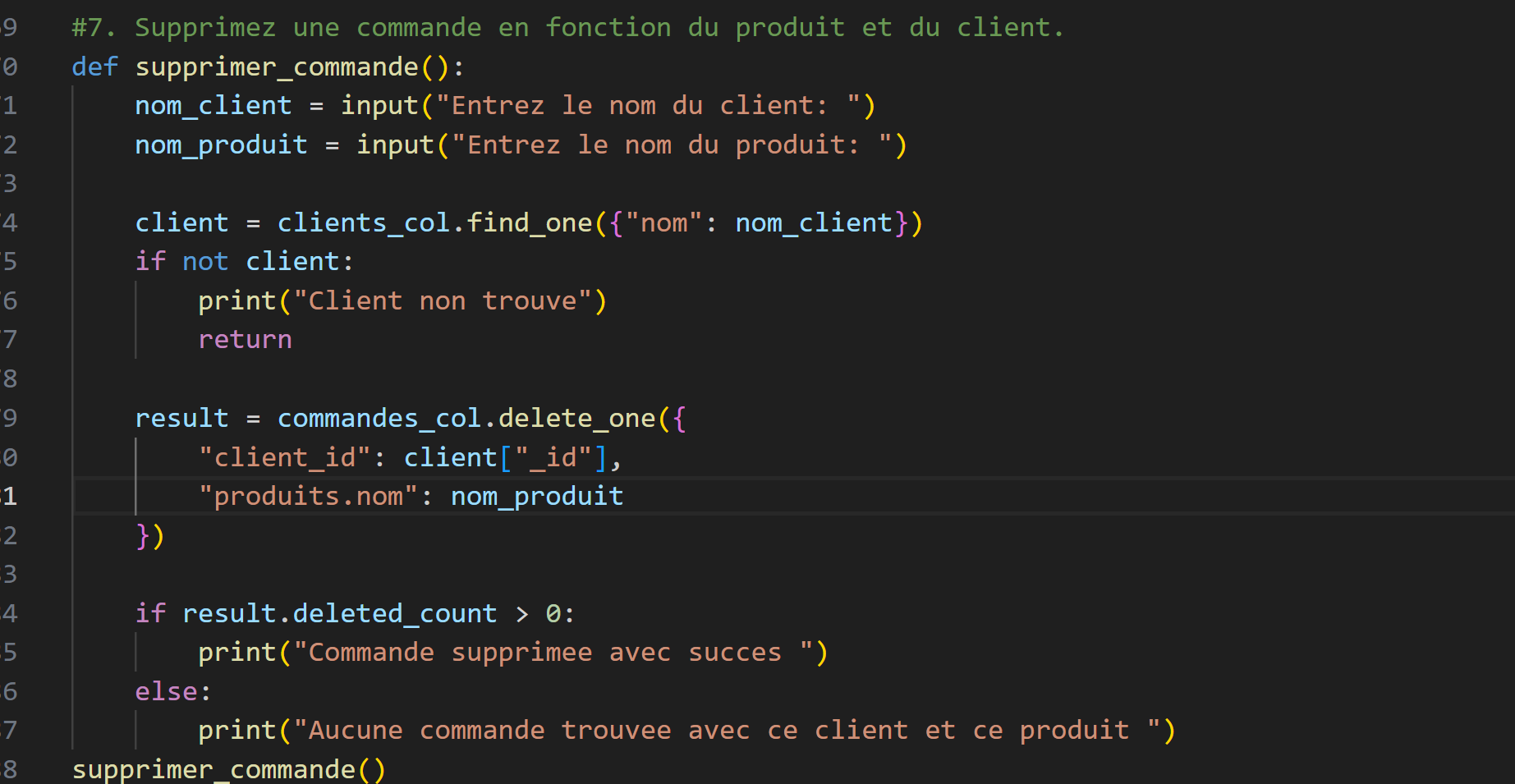
Ajouter un champ supplémentaire a tous les documents de la collection produits pour indiquer si le produit est disponible a la vente.

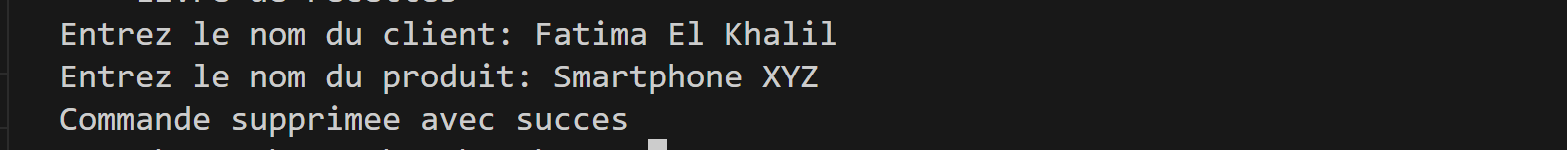
Explication :

On utilise update\_many pour modifier tous les documents de la collection.

Le champ disponible est ajouté avec une valeur par defaut True.

Cela permet de filtrer facilement les produits disponibles par la suite.





Objectif :

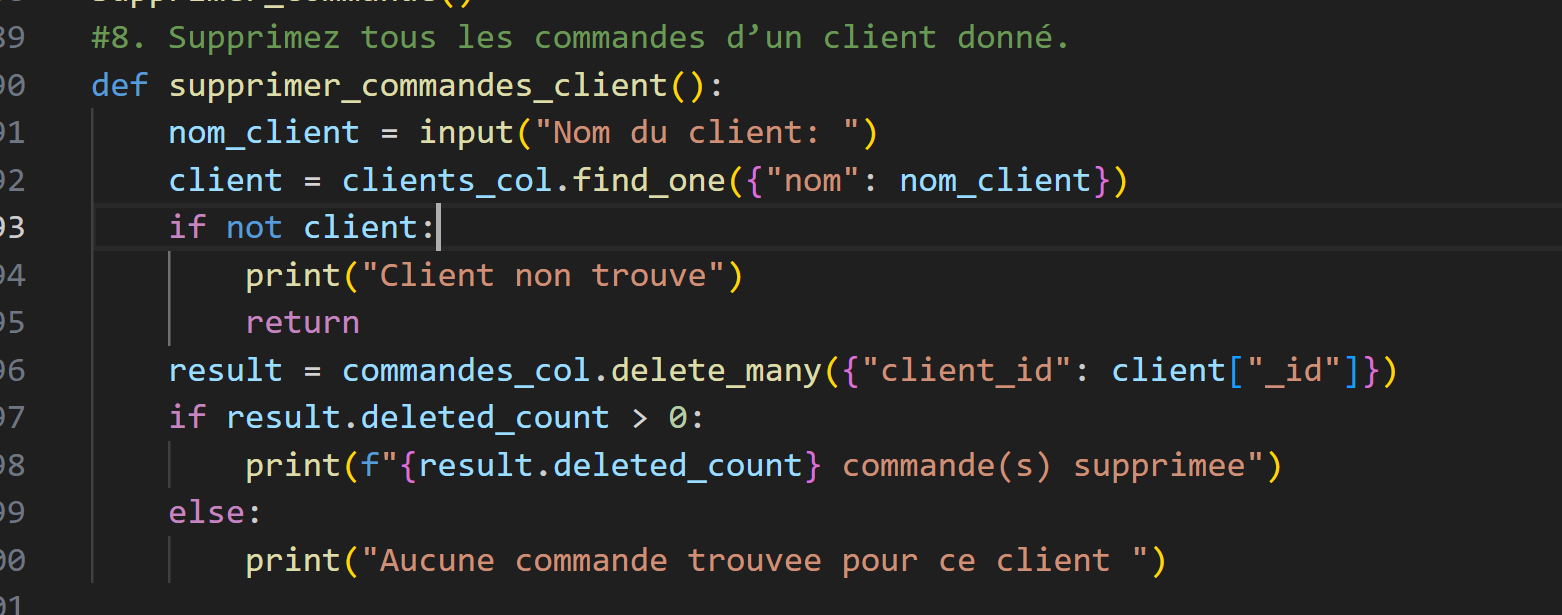
Permettre la suppression d’une commande spéecifique en ciblant un client et un produit precis.

Explication :

On demande a l’utilisateur le nom du client et le nom du produit.

On recupere l’\_id du client depuis la collection clients.

On utilise delete\_one pour supprimer une commande contenant ce produit pour ce client.



Objectif :

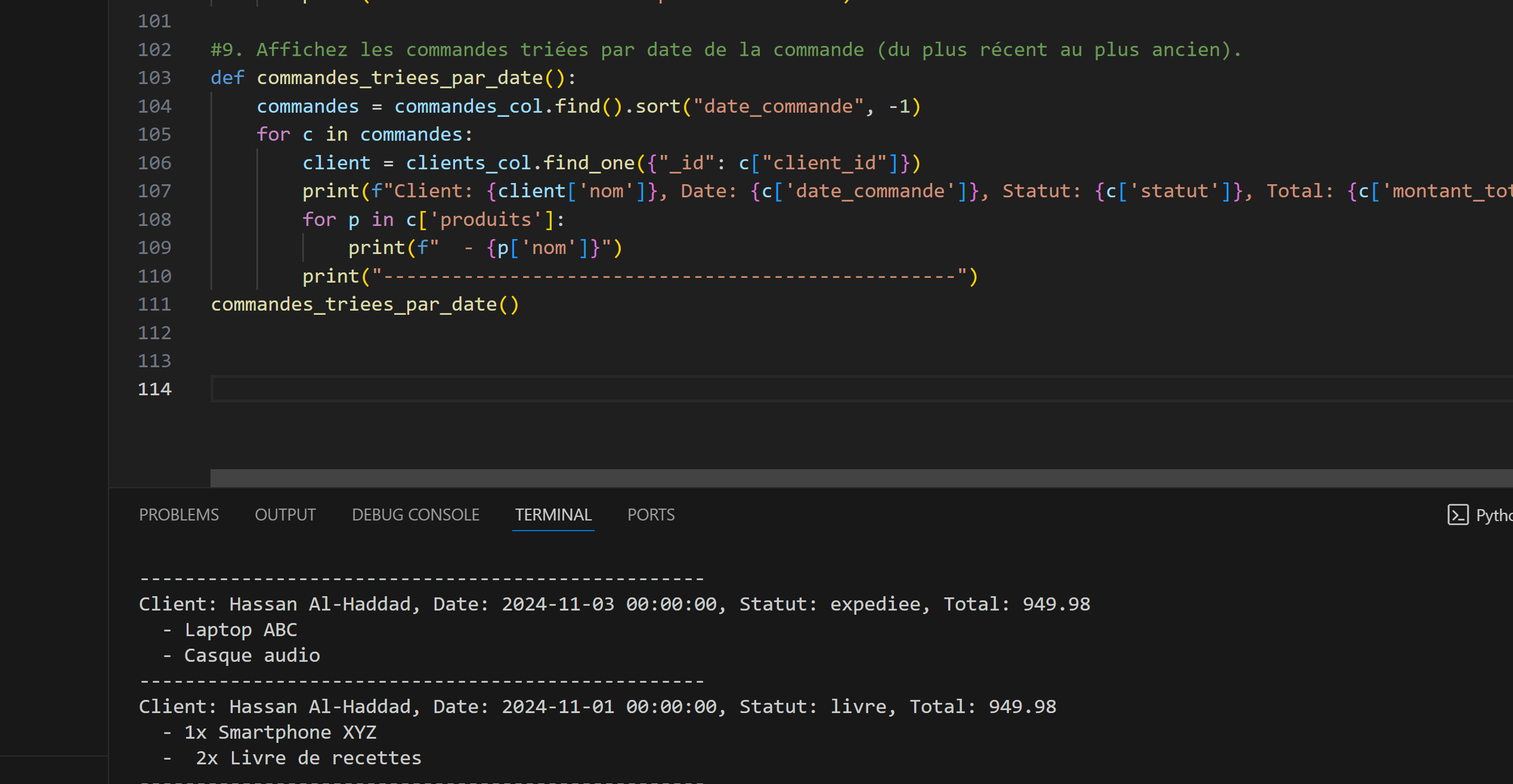
Supprimer toutes les commandes associees a un client specifique.

Explication :

L’utilisateur saisit le nom du client.

On recupere l’\_id du client depuis la collection clients.

On utilise delete\_many pour supprimer toutes les commandes ou client\_id correspond a cet \_id.



Objectif :

Lister toutes les commandes en affichant d’abord les plus récentes. Cela permet de suivre facilement l’activite recente des clients.

Explication :

On recupere toutes les commandes avec find().

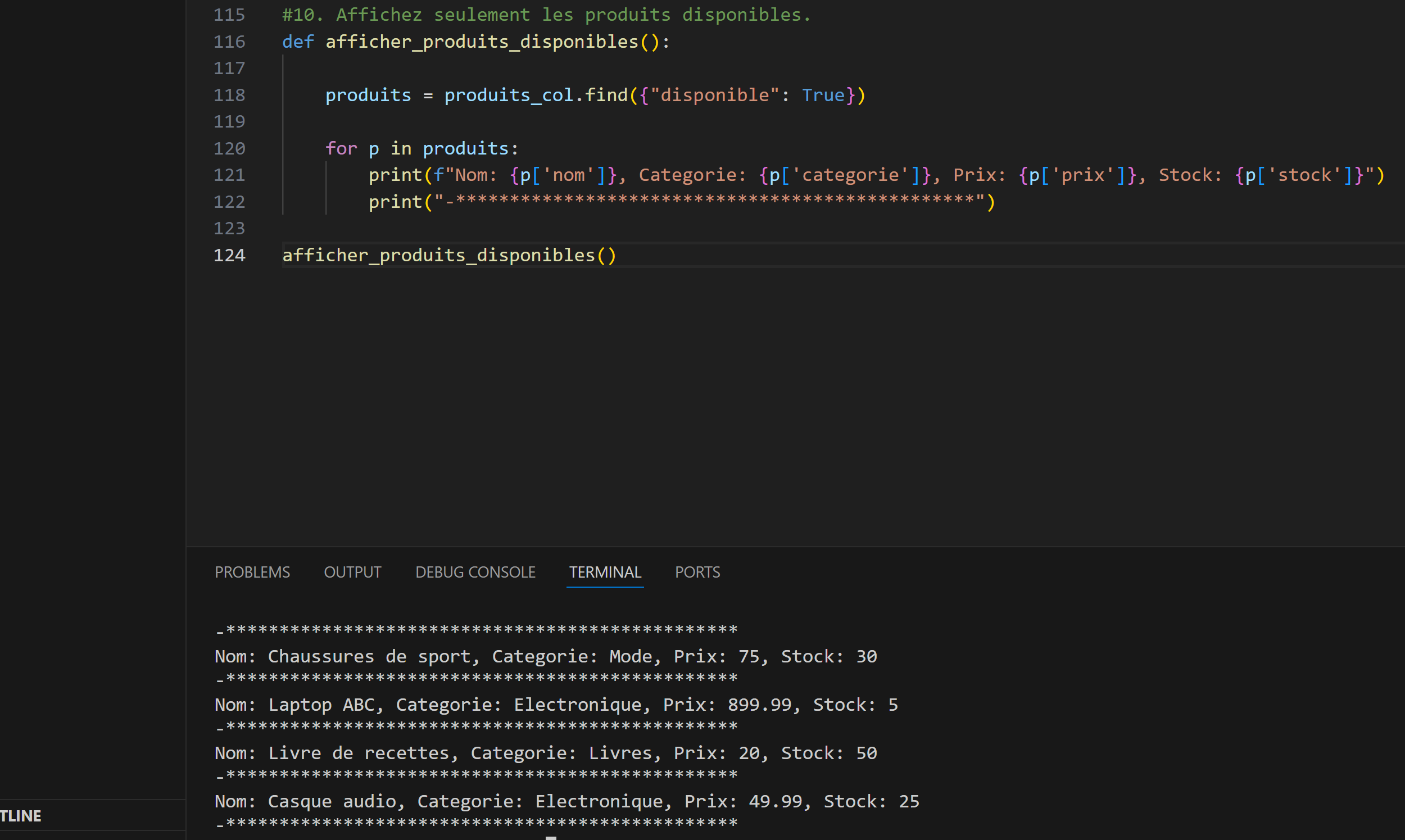
On utilise sort("date\_commande", -1) pour trier par date décroissante (-1 = du plus recent au plus ancien).

Pour chaque commande, on peut affiche

Le nom du client (recupere-via client\_id)

La date de la commande

Le statut -Le montant total



Objectif :

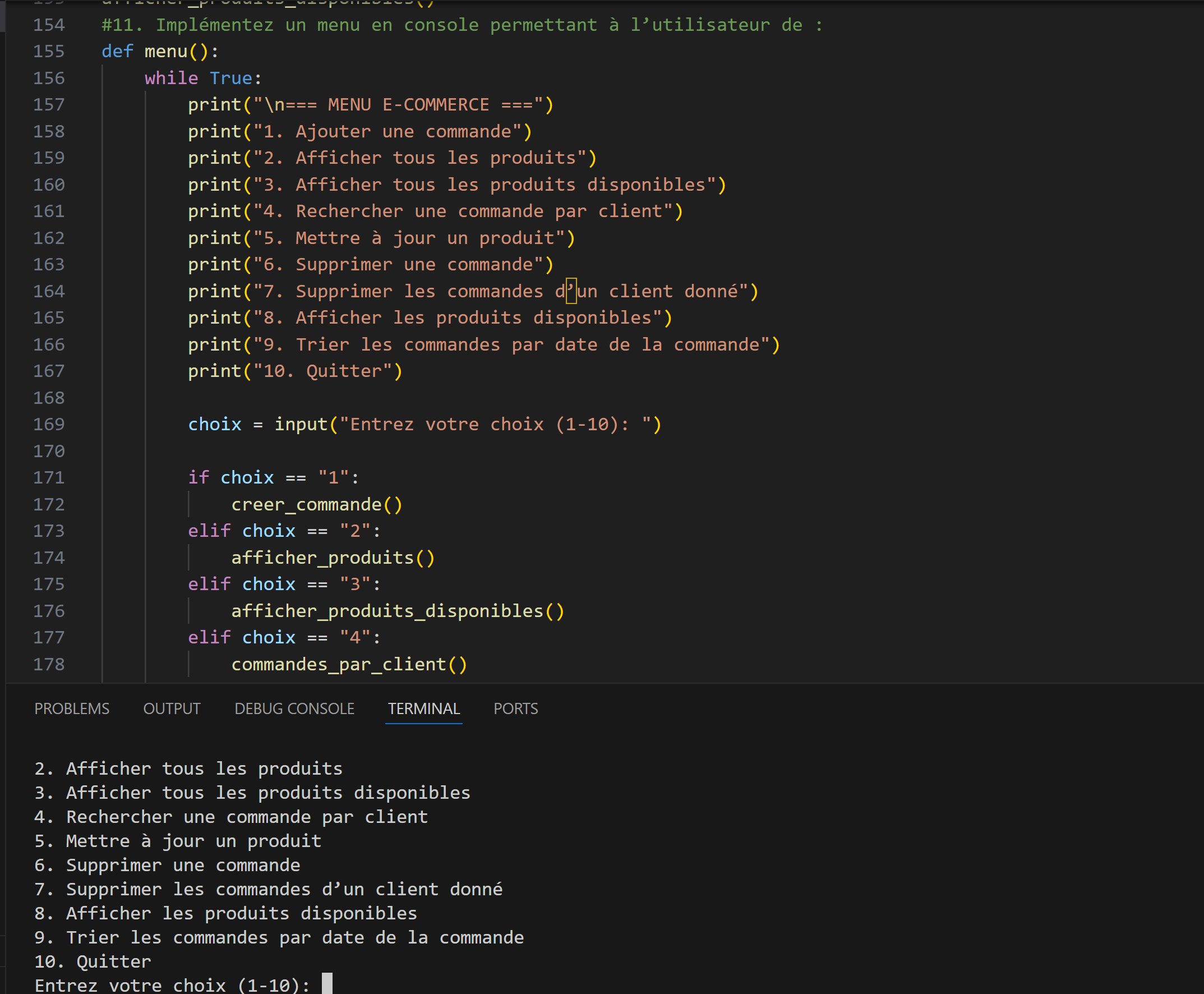
Lister uniquement les produits qui sont actuellement disponibles a la vente, en filtrant ceux qui ont le champ disponible = True. Cela permet de connaitre rapidement le stock actif.

Explication :

On utilise find({"disponible": True}) pour ne recuperer que les produits disponibles.

Pour chaque produit, on peut afficher :

Nom-Categorie- Prix -Stock



Objectif :

Permettre a l’utilisateur d’interagir facilement avec la mini-application e-commerce via un menu textuel. Chaque option correspond a une fonction deja creer ( creer\_commande()afficher\_produits() etc... ).

Explication :

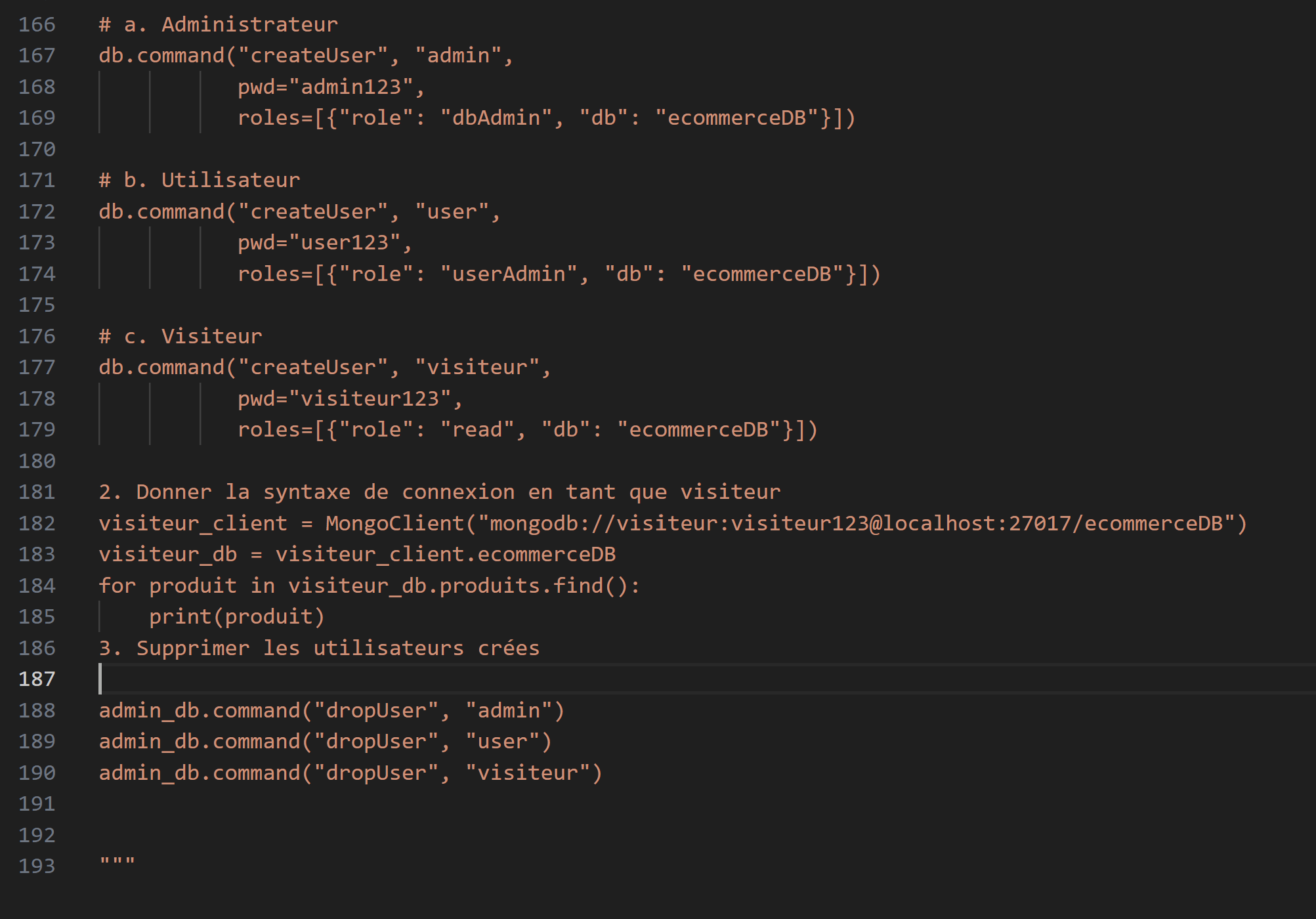
On utilise une boucle while True pour que le menu reste actif jusqu’a ce que l’utilisateur choisisse de quitter.

On affiche les options numerotees.

Selon le choix de l’utilisateur, on appelle la fonction correspondante.

L’option “Quitter” permet de sortir de la boucle.

***Partie III : Gestion des utilisateurs :***



Objectif :

Assurer la securite et le controle d’acces dans la base de donnees MongoDB en créant differents types d’utilisateurs avec des roles precis.

1. Creation de 3 utilisateurs

Administrateur : role dbAdmin, a tous les droits d’administration sur la base ecommerceDB.

Utilisateur : role userAdmin, peut gerer les utilisateurs et les permissions mais pas forcement les donnees.

Visiteur : role read, peut seulement lire les donnees de la base.

2.Connexion en tant que visiteur

3. Suppression des utilisateurs crees