def corriger\_non\_conformite(produit):

if produit.non\_conforme:

# Effectuer les actions nécessaires pour corriger la non-conformité

produit.correction\_effectuee = True

produit.non\_conforme = False

produit.enregistrer() # Sauvegarder les modifications dans la base de données

# Ajouter d'autres actions spécifiques, telles que l'envoi de notifications, etc.

print("La non-conformité a été corrigée avec succès.")

else:

print("Le produit est conforme. Aucune correction nécessaire.")

# Exemple d'utilisation

mon\_produit = Produit()

mon\_produit.non\_conforme = True

corriger\_non\_conformite(mon\_produit)

class Produit:

def \_\_init\_\_(self):

self.non\_conforme = False

self.correction\_effectuee = False

def enregistrer(self):

# Code pour enregistrer les modifications dans une base de données

# par exemple, en utilisant un ORM (Object-Relational Mapping) ou des requêtes SQL

# Exemple fictif : mise à jour d'une colonne "non\_conforme" dans une table "produits"

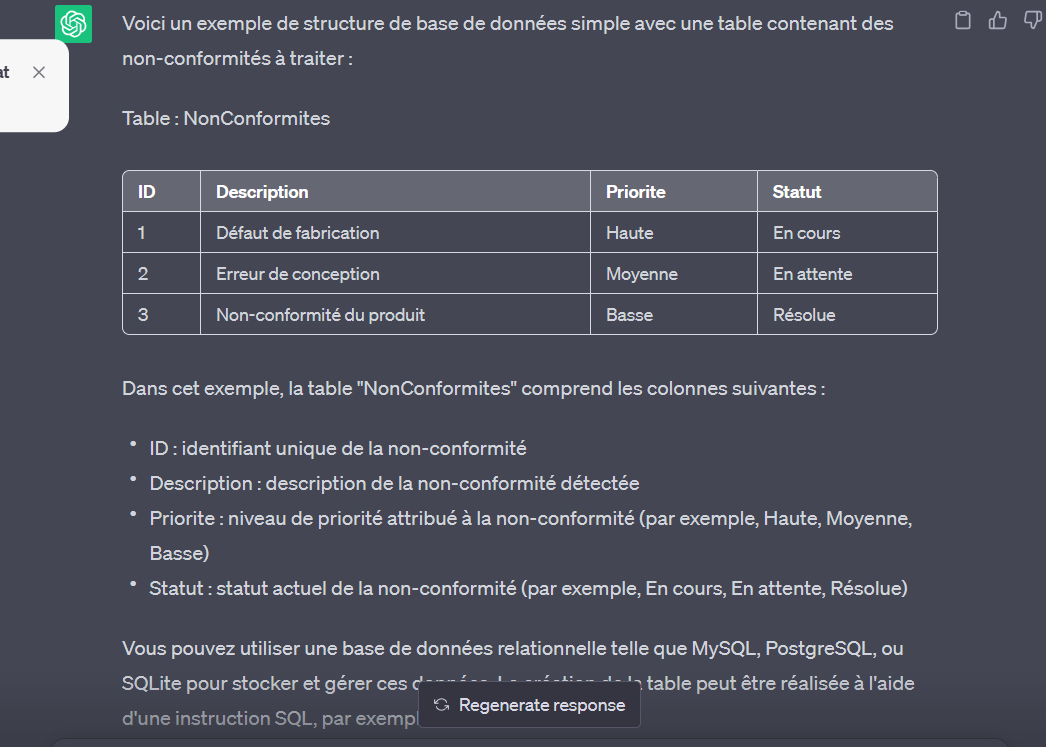
sql = "UPDATE produits SET non\_conforme = ? WHERE id = ?"

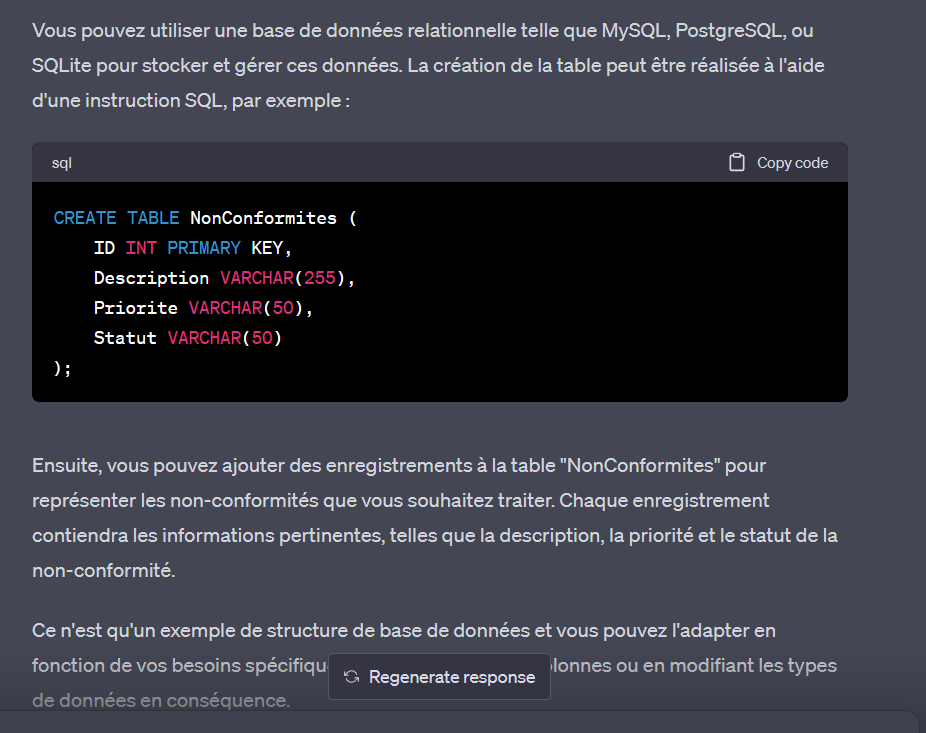
parametres = (self.non\_conforme, self.id)

execute\_sql(sql, parametres) # Exécution de la requête SQL avec les paramètres appropriés

# Autres opérations nécessaires pour la sauvegarde des données

print("Modifications enregistrées dans la base de données.")







import sqlite3

# Fonction pour récupérer les non-conformités depuis la base de données

def obtenir\_non\_conformites():

conn = sqlite3.connect('ma\_base\_de\_donnees.db') # Remplacez 'ma\_base\_de\_donnees.db' par le nom de votre base de données

cursor = conn.cursor()

# Exécution de la requête SQL pour obtenir les non-conformités

cursor.execute("SELECT ID, Description, Priorite, Statut FROM NonConformites")

non\_conformites = cursor.fetchall()

conn.close()

return non\_conformites

# Fonction pour résoudre une non-conformité spécifique

def resoudre\_non\_conformite(non\_conformite\_id):

conn = sqlite3.connect('ma\_base\_de\_donnees.db') # Remplacez 'ma\_base\_de\_donnees.db' par le nom de votre base de données

cursor = conn.cursor()

# Exécution de la requête SQL pour mettre à jour le statut de la non-conformité

cursor.execute("UPDATE NonConformites SET Statut = 'Résolue' WHERE ID = ?", (non\_conformite\_id,))

conn.commit()

conn.close()

print("La non-conformité a été résolue avec succès.")

# Fonction principale pour traiter les non-conformités en fonction des avis des clients

def traiter\_non\_conformites():

non\_conformites = obtenir\_non\_conformites()

# Affichage des non-conformités disponibles

print("Non-conformités à traiter :")

for nc in non\_conformites:

print(f"ID : {nc[0]}, Description : {nc[1]}, Priorité : {nc[2]}, Statut : {nc[3]}")

# Demander à l'utilisateur d'entrer l'ID de la non-conformité à résoudre

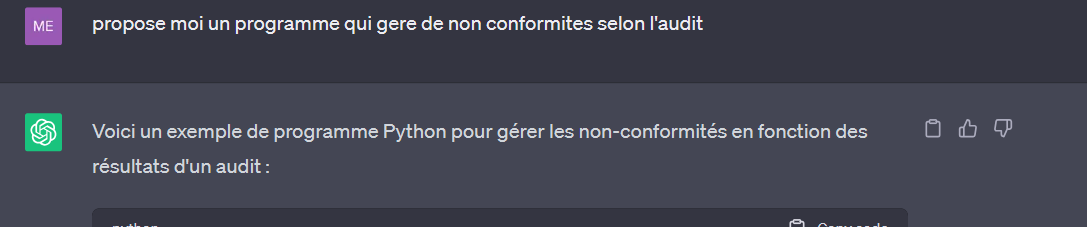
non\_conformite\_id = input("Entrez l'ID de la non-conformité à résoudre : ")

# Résoudre la non-conformité

resoudre\_non\_conformite(non\_conformite\_id)

# Appel de la fonction principale pour traiter les non-conformités

traiter\_non\_conformites ()



import sqlite3

# Fonction pour récupérer les non-conformités depuis la base de données

def obtenir\_non\_conformites():

conn = sqlite3.connect('ma\_base\_de\_donnees.db') # Remplacez 'ma\_base\_de\_donnees.db' par le nom de votre base de données

cursor = conn.cursor()

# Exécution de la requête SQL pour obtenir les non-conformités

cursor.execute("SELECT ID, Description, Priorite, Statut FROM NonConformites")

non\_conformites = cursor.fetchall()

conn.close()

return non\_conformites

# Fonction pour mettre à jour le statut d'une non-conformité suite à un audit

def mettre\_a\_jour\_non\_conformite(non\_conformite\_id, statut):

conn = sqlite3.connect('ma\_base\_de\_donnees.db') # Remplacez 'ma\_base\_de\_donnees.db' par le nom de votre base de données

cursor = conn.cursor()

# Exécution de la requête SQL pour mettre à jour le statut de la non-conformité

cursor.execute("UPDATE NonConformites SET Statut = ? WHERE ID = ?", (statut, non\_conformite\_id))

conn.commit()

conn.close()

print("Le statut de la non-conformité a été mis à jour avec succès.")

# Fonction principale pour traiter les non-conformités en fonction des résultats de l'audit

def traiter\_non\_conformites\_audit():

non\_conformites = obtenir\_non\_conformites()

# Affichage des non-conformités disponibles

print("Non-conformités à traiter :")

for nc in non\_conformites:

print(f"ID : {nc[0]}, Description : {nc[1]}, Priorité : {nc[2]}, Statut : {nc[3]}")

# Demander à l'utilisateur d'entrer l'ID de la non-conformité à mettre à jour

non\_conformite\_id = input("Entrez l'ID de la non-conformité à mettre à jour : ")

# Demander à l'utilisateur d'entrer le nouveau statut de la non-conformité

nouveau\_statut = input("Entrez le nouveau statut de la non-conformité : ")

# Mettre à jour le statut de la non-conformité

mettre\_a\_jour\_non\_conformite(non\_conformite\_id, nouveau\_statut)

# Appel de la fonction principale pour traiter les non-conformités selon l'audit

traiter\_non\_conformites\_audit()

