Projet IN513 Base de Données

PHASE I

**1 :Dans le cas où il y'a un problème avec le rendu, comme du code manquant ou un problème de format/conversion, voici le lien contenant toutes les sources qui ont permis de créer ce projet : https://github.com/uvsq22200574/Projet-IN513*

Il se peut que certaines parties de texte soient en blanc, donc invisibles.

Le sujet de la base de données représentera une pharmacie, le choix est motivé par la faisabilité du projet en étant seul à travailler sur celui-ci tout en permettant un sujet intéressant pour le stockage et le traitement des données.

Ainsi, nous pouvons déduire qu'une pharmacie possède un cahier des charges sur plusieurs aspects, tels que les médicaments (identifiant, type, nom commercial, stock), les clients (ordonnances, sécurité sociale, mutuelle), de même que certains aspects comme la nécessité d'une ordonnance, le taux de prise en charge, etc. En détails, la pharmacie doit pouvoir assurer la vente de médicaments en suivant différentes contraintes.

I. Cahier des charges :

La pharmacie doit pouvoir suivre le stock de médicaments. De plus, elle doit pouvoir vérifier qu'une vente est possible dans le sens où les quantités demandées, le médicament, l'ordonnance et les informations du client permettent que la transaction ait lieu.

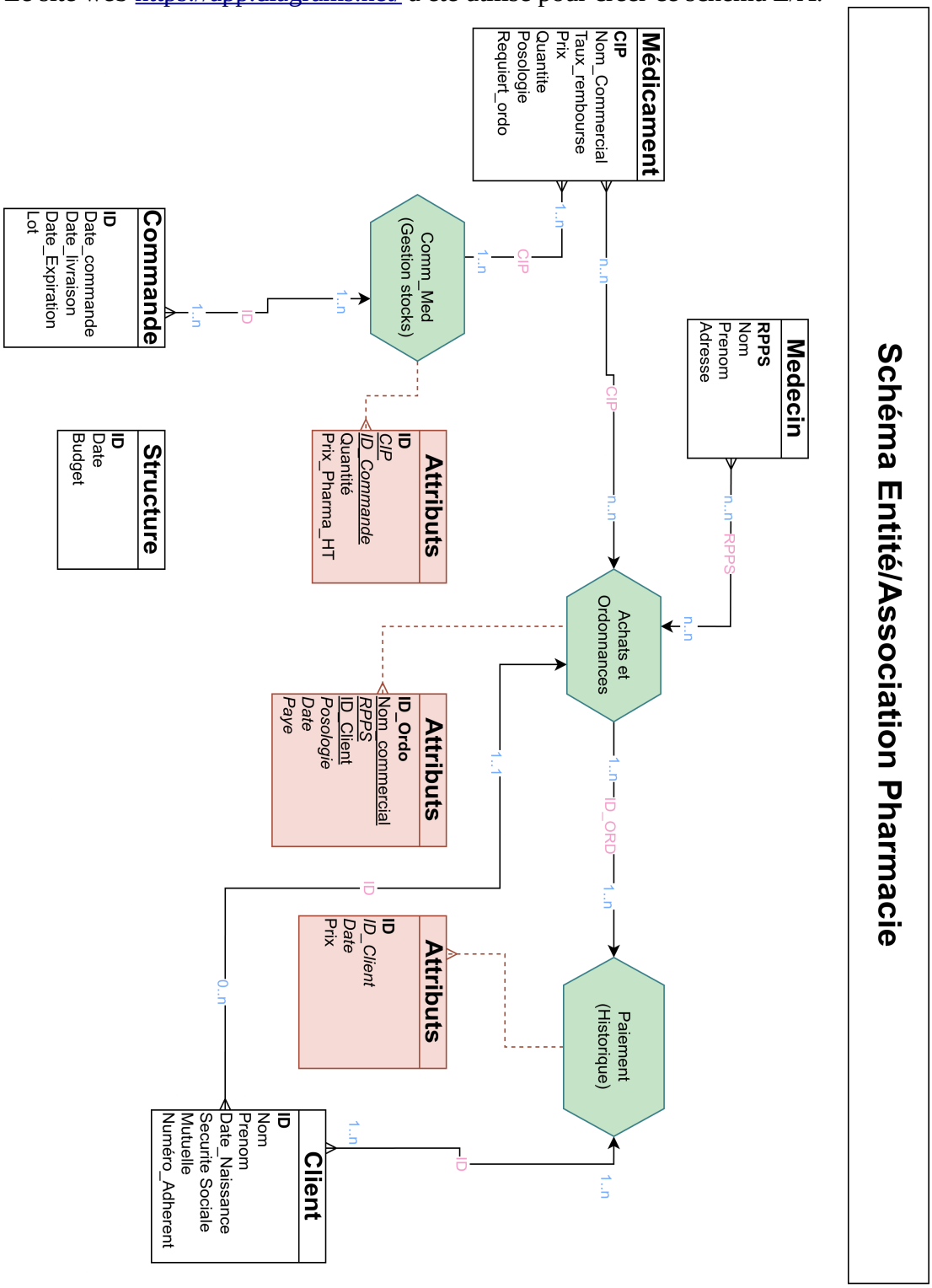
En ce qui concerne les ordonnances, il faut pouvoir identifier quel médecin l'a délivré, à qui elle est destinée et quels médicaments sont prescrits.

Mais au-delà des bases, une pharmacie est un commerce qui doit être rentable. On doit ainsi pouvoir suivre l'évolution financière du commerce, comme les ventes quotidiennes, les médicaments les plus/moins demandés, quels sont les marges et pouvoir comparer avec des prévisions ou des objectifs fixés.

Tout cela doit se faire dans le respect des contraintes et des droits, un employé ne doit avoir accès uniquement aux données en relation avec son poste et son rôle, et les données doivent être cohérentes, utiles et correctement stockées et manipulées.

II. Schéma Entité/Association :

Le site web https://app.diagrams.net/ à été utilisé pour créer ce schéma E/A.



III. Contraintes d'intégrité :

Comme rappelé précédemment, un médicament ne peut être vendu qu'en respectant les règles. Cela signifie qu'on ajoute une transaction dans la base de données uniquement lorsque qu'un médicament respecte les conditions suivantes : le médicament est en stock, il est vendu avec une ordonnance valide si besoin, le client à payer le prix correct[2] et a fourni sa carte vitale avec les informations sur sa sécurité sociale et sa mutuelle. La transaction n'est possiblement enregistrée que par un caissier ou un utilisateur ayant les permissions adéquates.

Les médicaments ne doivent, idéalement, jamais être en rupture de stock, cela ne signifie pas qu'un médicament ne peut pas être en rupture. Chaque médicament doit avoir un prix indiqué, on fait la distinction entre le prix payé par le client et le prix du médicament pour la pharmacie. Enfin, chaque lot de médicament à une date de péremption. On ne peut garder de lot expiré. On suppose qu'on ne possède pas de code CIP datant d'avant 2011.

Ainsi on peut avoir des alertes lorsque le stock d'un médicament est trop faible ou si une transaction ne respecte pas une règle spécifique (Exemple : Le médicament est vendu sans avoir fourni d'ordonnance alors que c'est requis).

Pour enregistrer un client dans la base de données, s'il possède une carte vitale avec son numéro de sécurité sociale on doit utiliser les chiffres pour vérifier qu'elle est valide, avec la syntaxe suivante *(Genre 1|2) - YY - MM - Département - Commune|code INSEE - Numéro d'ordre - Clé de contrôle, longueur de 15, clé=97-(NIR[:-2]% 97))*, et que l'employé n'a pas fait une erreur de saisie. On inclus également la mutuelle et le numéro d'adhérent, si le client en à une.

*2 : Il s'agit d'une condition valide dans le contexte mais en pratique ce n'est pas possible de l'implémenter dans la base de données.

IV. Droits et Vues : (2 max)

Pour les interactions avec les données, on doit pouvoir accéder uniquement aux données nécessaires avec les permissions qui sont accordées à l'utilisateur. Cela doit inclure le caissier, le chargé des rayons, le chargé des stocks, l'administrateur de la base de données, et enfin le directeur de la pharmacie. On ne part pas du principe que les clients puissent interroger la base de données pour obtenir des informations.

Tous les utilisateurs ont le droit de consulter les **stocks** et les **prix**. Le caissier, en plus de ces accès a le droit d'accéder aux informations des **ordonnances** mais pas aux détails liés à la **sécurité sociale** ou la **mutuelle** du client par exemple (On suppose que la saisie d'une entrée pour un nouveau client se fait automatiquement par le système mais ici on ne peut pas). Il doit juste savoir que les informations sont valides et que la transaction peut avoir lieu. Le chargé des stocks à accès à toutes les informations liées aux **stocks** et aux **livraisons** ainsi que le **budget** de la structure. L'administrateur à **tous les droits**, de même pour le directeur si on part du principe que son autorité sur le commerce le permet. Afin de limiter des erreurs de manipulation ou des attaques ciblées sur celui-ci, il ne pourra que consulter les données sensibles.

*3 :Puisque je ne peux pas utiliser la VM avec toutes les fonctionnalités, ces droits/utilisateurs ne peuvent pas être inclus dans la phase II. Bien que les vues soient possibles, dans le contexte où j'ai travaillé, elles n'ont pas beaucoup d'utilité, il n'y en aura donc pas.

V. Requêtes :

1. Ajouter un paiement et mettre à jour les tables
2. Quelles sont les 5 dernières transactions ?
3. Quel est le médicament le moins cher et qui est en stock ?
4. Quels sont les médicaments les plus/moins en stock ?
5. Quel client à effectuer le plus d'achat sur l'année et pour quel montant ?
6. Quel client à dépenser le plus sur l'année et pour quel montant ?
7. Quel est le chiffre d'affaires du mois/entre deux dates?
8. Quel livraisons sont en attente (commandé mais pas livré) ?
9. Quels sont les lots de médicaments expirés à retirer ?
10. Quelle est la liste de toutes les ordonnances délivrées par un médecin spécifique ?
11. Quels médicaments ont été commandés dans une commande spécifique ?
12. Mettre à jour la commande en définissant la date de livraison, la date d'expiration est définie et le stock est mis à jour
13. Quel est le prix d'une commande ?
14. Quel est le prix total (hors ss et mutuelle) d'une ordonnance?
15. Quels médicaments ont été commandés mais pas encore livré ?
16. Qui sont les clients qui ont une sécurité sociale mais pas de mutuelle ?
17. Quels sont les médicaments les plus/moins bien remboursés par la SS ?
18. Quel est l'historique des achats d'un client spécifique ?
19. Combien un client dépense en moyenne ?
20. Qui sont les clients qui ont dépenser plus que la moyenne ?

VI. Tables et colonnes

1) ***Médicament*** {
   CIP - CHAR(13) PK,
   Nom_Commercial - VARCHAR(128),
   Taux_remboursement - NUMBER(5, 2) $0 \leqslant x \leqslant 100$,
   Prix - NUMBER(6, 3), // Prix de vente
   Quantite - NUMBER $\geqslant 0$,
   Posologie - NUMBER(7, 3) > 0,
   Requiert_ord - BOOL,
   }
2) ***Client*** {
   ID - INT PK,
   Nom - VARCHAR (128),
   Prenom - VARCHAR (128),
   Date_Naissance - DATE
   Sécurité Sociale - CHAR(15),
   Mutuelle - VARCHAR(32),
   Numéro_Adherent - NUMBER(12), // Rien si pas de mutuelle
   }
3) ***Commande*** {
   ID - INT PK,
   Date_commande - TIMESTAMP,
   Date_livraison - TIMESTAMP,
   Date_expiration - TIMESTAMP,
   Lot - CHAR(8),
   }
4) ***Medecin*** {
   RPPS - CHAR(11) PK,
   Nom - VARCHAR(128),
   Prénom - VARCHAR(128),
   Adresse - VARCHAR(128)
   }
5) ***Structure*** { // Pour garder en mémoire le budget
   ID - INT PK,
   Date - DATE,
   Budget - NUMBER(10, 2) $\geqslant 0$,
   }

Relations Many to Many :

1) ***Comm_Med (Médicament + Commande)*** {
   ID - INT PK,
   CIP - CHAR(13) FK,
   ID_Commande - NUMBER FK,
   Quantite - NUMBER > 0,
   Prix_HT_UNIIT - NUMBER(6, 2)
   }

2) ***Achats_Ordonnances*** {
   ID_Ordo - INT PK,
   Nom_commercial - VARCHAR(128) FK,
   RPPS - CHAR(11) FK,
   ID_Client - NUMBER FK,
   Posologie - NUMBER(4, 2) > 0,
   Date - TIMESTAMP,
   Paye - BOOL,
   }
3) ***Paiement (Client + Ordonnance)*** {
   ID - INT PK,
   ID_Client - INT FK,
   Date - TIMESTAMP,
   Prix - NUMBER(6, 2) ⩾ 0,
   }

PHASE II

I. Création des tables :

Avant de commencer les opérations sur la base de données ou l'implémentation d'un jeu de données, il faut définir les tables et les type des colonnes. Voici donc les commandes SQL qui permettent de créer ces tables :

```
-- Création des tables principales
CREATE TABLE Medicament (
    CIP VARCHAR2(13) PRIMARY KEY NOT NULL CHECK (LENGTH(CIP) = 13),
    Nom_Commercial VARCHAR2(128) UNIQUE NOT NULL,
    Taux_remboursement NUMBER(5, 2) CHECK (Taux_remboursement BETWEEN 0 AND 100),
    Prix NUMBER(6, 3),
    Quantite NUMBER CHECK (Quantite ⩾ 0), -- boites ou flacons
    Posologie NUMBER(7, 3) CHECK (Posologie > 0), -- mg ou ml
    Requiert_ord VARCHAR2(5) DEFAULT 'False' CHECK (Requiert_ord IN ('False', 'True')) -- BOOLEAN
);

CREATE TABLE Client (
    ID NUMBER PRIMARY KEY NOT NULL,
    Nom VARCHAR2(128) NOT NULL,
    Prenom VARCHAR2(128) NOT NULL,
    Date_Naissance DATE NOT NULL,
    Securite_Sociale VARCHAR2(15) CHECK(LENGTH(Securite_Sociale) = 15) UNIQUE,
    Mutuelle VARCHAR2(32),
    Numero_Adherent NUMBER(12)

);

CREATE TABLE Commande (
    ID NUMBER PRIMARY KEY NOT NULL,
    Date_commande TIMESTAMP,
    Date_livraison TIMESTAMP,
    Date_expiration TIMESTAMP,
    Lot VARCHAR2(8) CHECK (LENGTH(Lot) = 8)
);

CREATE TABLE Medecin (
    RPPS VARCHAR2(11) PRIMARY KEY NOT NULL CHECK (LENGTH(RPPS) = 11),
    Nom VARCHAR2(128) NOT NULL,
    Prenom VARCHAR2(128) NOT NULL,
    Adresse VARCHAR2(128)
);

CREATE TABLE Structure (
    ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    dDate DATE UNIQUE,
    Budget NUMBER(10, 2) CHECK (Budget ⩾ 0)
);

-- Création des tables de relations many-to-many
CREATE TABLE Comm_Med (
    ID NUMBER PRIMARY KEY NOT NULL,
    CIP VARCHAR2(13) NOT NULL CHECK (LENGTH(CIP) = 13),
    ID_Commande NUMBER NOT NULL,
    Quantite NUMBER CHECK (Quantite > 0), -- Boite ou flacon
    Prix_HT_UNIT NUMBER(6, 2),
    FOREIGN KEY (CIP) REFERENCES Medicament(CIP),
    FOREIGN KEY (ID_Commande) REFERENCES Commande(ID)
);

CREATE TABLE Achats_Ordonnances (
    ID_Ordo NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    Nom_commercial VARCHAR2(128) NOT NULL,
    RPPS VARCHAR2(11),
    ID_Client NUMBER NOT NULL,
    Posologie NUMBER(6, 2) CHECK (Posologie > 0),
    dDate DATE,
    Paye VARCHAR2(5) DEFAULT 'False' CHECK (Paye IN ('False', 'True')),
    FOREIGN KEY (Nom_commercial) REFERENCES Medicament(Nom_Commercial),
    FOREIGN KEY (RPPS) REFERENCES Medecin(RPPS),
    FOREIGN KEY (ID_Client) REFERENCES Client(ID),
    CONSTRAINT chk_RPPS_length CHECK (RPPS IS NULL OR LENGTH(RPPS) = 11)
);

CREATE TABLE Paiement (
    ID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    ID_Client NUMBER NOT NULL,
```

```
    dDate TIMESTAMP,
    Prix NUMBER(6, 2) CHECK (Prix ⩾ 0),
    FOREIGN KEY (ID_Client) REFERENCES Client(ID)
);

-- Environnement
ALTER SESSION SET NLS_TIME_FORMAT = 'HH24:MI:SS';
ALTER SESSION SET NLS_DATE_FORMAT = 'YYYY/MM/DD';
ALTER SESSION SET NLS_TIMESTAMP_FORMAT = 'YYYY/MM/DD HH24:MI:SS';
ALTER SESSION SET TIME_ZONE = 'Europe/Paris';
```

Le type VARCHAR2 est souvent utilisé car il permet de stocker des chaînes de caractères à taille variable sans espaces avant/après, ce qui permet de mesurer la longueur des chaînes, par rapport au type CHAR. La longueur maximale est définie soit par la taille maximale si celle-ci est connue à l'avance, comme le numéro de sécurité sociale, ou alors une valeur suffisamment grande que la plupart des cas sont couverts. Les dates sont représentées par un type TIMESTAMP quand on à besoin de plus de précision, sinon on utilise le type DATE. Les booléens sont représentés par une chaîne de caractère qui prend comme valeur soit 'False' soit 'True', puisqu'il ne semble pas y avoir de type dédié. Les nombres flottants sont représentés par NUMBER(x, y), ce qui correspond à des nombres de longueur total x et de précision y. Avec ces types, on couvre la plupart des besoins de cette BDD.

Avant de pouvoir peupler les tables avec des données, on doit s'assurer que celles-ci soit en accord avec les contraintes d'intégrité. Les types sont déjà un bon début mais c'est très insuffisant. Puisque nous travaillons en PL/SQL, on utilisera des triggers qui se déclenchent avant l'insertion et qui vérifie qu'aucune erreur n'ait été commise. Voici la liste de tous les triggers nécessaires au maintien des contraintes d'intégrité ainsi que leurs description en langage naturel:

II. Création des triggers :

1) Nombre Sécurité Sociale

Le premier trigger concerne la création d'une entrée dans la table CLIENT. Il faut ainsi s'assurer que le numéro de sécurité sociale soit valide et que les dates correspondent. Pour cela il faut remplir les conditions suivantes :

- Le numéro est de longueur 15
- La chaîne de caractères ne contient que des chiffres
- Les dates de naissance dans le numéro et dans la colonne sont les mêmes
- Le numéro à la fin est correct, en calculant la somme de contrôle

On obtient ainsi le code suivant, ainsi que des insertions de test :

```
-- Trigger 1
-- Ce trigger permet de s'assurer que les numéros de sécurité sociale généré soit corrects
-- On vérifie d'abord la longueur, puis les caratères, puis la date puis la clé
CREATE OR REPLACE TRIGGER validate_securite_sociale
BEFORE INSERT OR UPDATE ON Client FOR EACH ROW
DECLARE
    numeric_part CHAR(13);  -- Pour obtenir les 13 premiers chiffres
    check_digits CHAR(2);   -- Pour obtenir la clé de vérification
    date_digits CHAR(4);    -- Pour obtenir la date
    birth_date CHAR(4);          -- Pour obtenir la date
    calculated_check NUMBER; -- Le résultat du calcul de la clé de vérification
BEGIN
    -- Vérification de la longueur
    IF LENGTH(:NEW.Securite_Sociale) ≠ 15 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Le numéro de Sécurité Sociale doit être long de 15 caractères.');
    END IF;
    -- Vérification de la présence de chiffres seulement
    numeric_part := SUBSTR(:NEW.Securite_Sociale, 1, 13);
    IF NOT REGEXP_LIKE(numeric_part, '^[0-9]+$') THEN
        RAISE_APPLICATION_ERROR(-20002, 'Le numéro de Sécurité Sociale doit contenir uniquement des chiffres.');
    END IF;
    -- Vérification des dates
    date_digits := SUBSTR(:NEW.Securite_Sociale, 2, 4);
    birth_date := TO_CHAR(:NEW.Date_Naissance, 'YYMM');
    IF TO_NUMBER(birth_date) ≠ TO_NUMBER(date_digits) THEN
        RAISE_APPLICATION_ERROR(-20003, 'Le numéro de Sécurité Sociale et la date de naissance ne correspondent
pas.');
    END IF;
    -- Vérification de la clé de controle
    check_digits := TO_NUMBER(SUBSTR(:NEW.Securite_Sociale, 14, 2));
    calculated_check := 97 - MOD(TO_NUMBER(numeric_part), 97);
    IF check_digits ≠ calculated_check THEN
        RAISE_APPLICATION_ERROR(-20004, 'Le numéro de Sécurité Sociale contient des erreurs ! La clé vaut: ' ||
check_digits || ' elle devrait être de: ' || calculated_check || '.');
    END IF;
END;
/

-- Valide
INSERT INTO Client VALUES (0,'Assis', 'Hugo', TO_DATE('2004/07/05', 'YYYY/MM/DD'), '104073417200192',
'Mutuelle1', 123456);
-- Echec longueur
INSERT INTO Client VALUES (1,'Assis', 'Hugo', TO_DATE('2004/07/05', 'YYYY/MM/DD'), '10407341720192',
'Mutuelle1', 123456);
-- Echec type
INSERT INTO Client VALUES (2,'Assis', 'Hugo', TO_DATE('2004/07/05', 'YYYY/MM/DD'), '104072B30500126',
'Mutuelle1', 123456);
-- Echec date
INSERT INTO Client VALUES (3,'Assis', 'Hugo', TO_DATE('2004/07/05', 'YYYY/MM/DD'), '101073417200145',
'Mutuelle1', 123456);
INSERT INTO Client VALUES (4,'Assis', 'Hugo', TO_DATE('2004/07/05', 'YYYY/MM/DD'), '104017815824749',
'Mutuelle1', 123456);
INSERT INTO Client VALUES (5,'Assis', 'Hugo', TO_DATE('2004/07/05', 'YYYY/MM/DD'), '101017815824702',
'Mutuelle1', 123456);
-- Echec clé
INSERT INTO Client VALUES (0,'Assis', 'Hugo', TO_DATE('2004/07/05', 'YYYY/MM/DD'), '104073417200169',
'Mutuelle1', 123456);
```

2) Dates commande

Permet de vérifier que les dates de livraison et d'expiration soit après la date de commande. Voici le code :

```
-- Trigger 2
-- Ce trigger permet de vérifier que les dates des commandes aient du sens
-- Comme par exemple on ne peut pas être livré avant la date de commande
CREATE OR REPLACE TRIGGER create_commande
BEFORE INSERT OR UPDATE ON Commande FOR EACH ROW
BEGIN
    -- Vérifier que Date_livraison ⩾ Date_commande
    IF :NEW.Date_commande IS NULL THEN
        RAISE_APPLICATION_ERROR(-20005, 'La commande doit avoir une date.');
    END IF;

    IF :NEW.Date_livraison IS NOT NULL AND :NEW.Date_livraison < :NEW.Date_commande THEN
        RAISE_APPLICATION_ERROR(-20006, 'La livraison doit s''effectuer après la commande.');
    END IF;

    -- Vérifier que Date_expiration ⩾ Date_livraison
    IF :NEW.Date_livraison IS NOT NULL AND :NEW.Date_expiration IS NOT NULL AND :NEW.Date_expiration
< :NEW.Date_livraison THEN
        RAISE_APPLICATION_ERROR(-20007, 'La commande doit expirer plus tard que la date de
livraison.');
    END IF;
    IF :NEW.Date_expiration IS NOT NULL AND :NEW.Date_expiration < :NEW.Date_commande THEN
    RAISE_APPLICATION_ERROR(-20008, 'La commande doit expirer après la date de commande.');
    END IF;
END;
/
```

3) Empêche la modification de commandes dont la date de livraison est déjà définie. Cela donne le code suivant :

```
-- trigger 3
-- Empêche la modification de commandes déjà conclues
CREATE OR REPLACE TRIGGER check_commande_delivered
BEFORE INSERT ON Comm_Med
FOR EACH ROW
DECLARE
    v_livraison TIMESTAMP;
BEGIN
    -- Récupère la date de livraison pour la commande spécifiée
    SELECT Date_livraison INTO v_livraison
    FROM Commande
    WHERE ID = :NEW.ID_Commande;

    -- Vérifie si la date de livraison est déjà renseignée
    IF v_livraison IS NOT NULL THEN
        RAISE_APPLICATION_ERROR(-20009, 'Impossible d''ajouter un médicament à une commande déjà
livrée.');
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20010, 'La commande spécifiée n''existe pas.');
END;
/
```

4) Quand on met une date de livraison, il faut appliquer une date d’expiration. Elle est automatiquement calculée pour être de 2 semaines. Voici le code :

```
-- trigger 4
-- Si une livraison devient datée, on applique automatiquement la date d'expiration, qui est fixée à 15
jours.
CREATE OR REPLACE TRIGGER set_expiration_date
BEFORE UPDATE OF Date_livraison ON Commande
FOR EACH ROW
BEGIN
    -- Vérifie si la date de livraison est définie et applique la date d'expiration à j+15
    IF :NEW.Date_livraison IS NOT NULL THEN
        :NEW.Date_expiration := :NEW.Date_livraison + INTERVAL '15' DAY;
    END IF;
END;
/
```

5) Lors de la création d’une commande, il faut attribuer un numéro de lot pour pouvoir identifier aisément les boites. Il faut qu’il soit unique. Voici le code :

```
-- trigger 5
-- Génère automatiquement un lot quand on créer une commande, on vérifie également qu'il n'existe pas
déjà.
CREATE OR REPLACE TRIGGER generate_lot
BEFORE INSERT ON Commande FOR EACH ROW
DECLARE
    v_lot VARCHAR2(8);
    v_count NUMBER;
BEGIN
    -- Boucle pour générer un Lot unique
    LOOP
        -- Générer une chaîne aléatoire avec lettres et chiffres
        v_lot := SUBSTR(
            DBMS_RANDOM.STRING('U', 6) || DBMS_RANDOM.STRING('X', 6),
```

```
            DBMS_RANDOM.VALUE(1, 5),
            8
        );

        -- On récupère le nombre de lots similaires à celui généré
        SELECT COUNT(*) INTO v_count
        FROM Commande
        WHERE Lot = v_lot;

        -- Sortir de la boucle si le lot est unique
        EXIT WHEN v_count = 0;
    END LOOP;

    -- Assigner le Lot généré
    :NEW.Lot := v_lot;
END;
/
```

6) Modifie automatiquement les stocks pour rajouter les médicaments commandés. Voici donc le code :

```
-- trigger 6
-- Met à jour les quantité de médicaments et le budget quand on ajoute la date de livraison (Une
livraison à eu lieue donc)
-- Ne met pas à jour les stocks ou le budget quand on modifie la date de livraison.
CREATE OR REPLACE TRIGGER update_stock_and_budget
AFTER UPDATE OF Date_livraison ON Commande FOR EACH ROW
DECLARE
    v_cip VARCHAR2(13);
    v_qty NUMBER;
    price NUMBER(10, 2);
BEGIN
    -- Si la date de livraison précédente existait déjà, on ne fait rien
    IF :OLD.Date_livraison IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('La date de livraison existait déjà, aucune mise à jour du budget ou des
stocks.');
    ELSE
        -- Parcourt les lignes de Comm_Med associées à cette commande
        FOR rec IN (SELECT CIP, Quantite, Prix_HT_UNIT FROM Comm_Med WHERE ID_Commande = :NEW.ID) LOOP
            -- Met à jour la quantité dans la table Medicament
            UPDATE Medicament SET Quantite = Quantite + rec.Quantite WHERE CIP = rec.CIP;

            -- Sinon, met à jour le budget existant
            price := rec.QUANTITE * rec.PRIX_HT_UNIT;
            UPDATE Structure SET Budget = Budget - price WHERE dDate = TRUNC(SYSDATE);
            DBMS_OUTPUT.PUT_LINE('Le prix du médicament '|| rec.cip ||' est:' || price || '€');

        END LOOP;
    END IF;
END;
/
```

7) Ce trigger permet de segmenter le budget pour chaque jour, permettant de suivre l'évolution. Voici le code :

```
-- trigger 7
-- Met à jour la date dès qu'on met à jour le budget de la structure
CREATE OR REPLACE TRIGGER update_budget
BEFORE UPDATE ON Structure
DECLARE
    last_date DATE;
    last_budget NUMBER(10, 2);
BEGIN
    -- Si la dernière date n'est pas aujourd'hui, insère une nouvelle ligne
    -- Récupère la dernière date et le dernier budget
    SELECT dDate, Budget INTO last_date, last_budget FROM Structure WHERE dDate = (SELECT MAX(dDate)
FROM Structure);
    IF last_date =/= TRUNC(SYSDATE) THEN
        INSERT INTO Structure (dDate, Budget) VALUES (TRUNC(SYSDATE), last_budget);
    END IF;
END;
/
```

8) Ce dernier trigger permet de gérer toutes les actions en rapport avec les paiements, tel que la mise à jour des stocks, du budget, les vérifications de validité, les calculs etc. Voici le code :

```
-- trigger 8
-- Puisque sans ID d'ordonnance on ne peut pas vendre de médicaments qui en requiert, il n'est pas
nécessaire de vérifier cela.

CREATE OR REPLACE TRIGGER process_paiement
BEFORE INSERT ON Paiement FOR EACH ROW
DECLARE
    total_price NUMBER(6, 2) := 0;
    total_price_client NUMBER(6, 2) := 0;

    price NUMBER (6, 2) := 0;
    quantite_per_med NUMBER := 0;
```

```
    taux_remboursement NUMBER(5, 2);

    quantite_meds NUMBER (6, 2);
    mutuelle VARCHAR2(32);
    securite_Sociale VARCHAR2(15);
BEGIN
    -- Récupération des informations du client
    SELECT Securite_Sociale, Mutuelle INTO securite_Sociale, mutuelle FROM CLIENT WHERE ID
= :NEW.ID_Client;

    -- On informe si le client est affilié à la sécurité sociale
    IF securite_Sociale IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Le client est affilié à la sécurité sociale.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('/!\ Le client n''est pas affilié à la sécurité sociale.');
    END IF;

    -- On informe si le client est affilié à une mutuelle
    IF mutuelle IS NOT NULL THEN
        DBMS_OUTPUT.PUT_LINE('Le client est affilié à la mutuelle ' || mutuelle || '.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('/!\ Le client n''est pas affilié à une mutuelle.');
    END IF;

    -- Pour chaque médicament de l'ordonnance
    FOR ordo IN (SELECT ID_Ordo, Nom_Commercial, Posologie, dDate FROM Achats_Ordonnances
WHERE :NEW.ID_Client = Achats_Ordonnances.ID_Client AND Paye = 'False') LOOP
        -- Vérification de la date de l'ordonnance
        IF SYSDATE - ordo.dDate >= 14 THEN
            RAISE_APPLICATION_ERROR(-20011, 'L''ordonnance est expirée.');
            -- Vider les messages précédents
            DBMS_OUTPUT.DISABLE;
            DBMS_OUTPUT.ENABLE;
            RAISE;
        END IF;

        -- Calcul du prix et des quantités demandées
        SELECT ordo.Posologie / Medicament.Posologie INTO quantite_per_med FROM Medicament WHERE
Nom_Commercial = ordo.Nom_Commercial;
        SELECT Prix * quantite_per_med INTO price FROM Medicament WHERE Nom_Commercial =
ordo.Nom_Commercial;
        SELECT Quantite, Taux_remboursement INTO quantite_meds, taux_remboursement FROM Medicament
WHERE ordo.Nom_Commercial = Nom_Commercial;

        IF quantite_meds - quantite_per_med >= 0 THEN
            DBMS_OUTPUT.PUT_LINE('Le client à demandé ' || quantite_per_med || ' boites de '||
ordo.Nom_Commercial ||', ce qui fait ' || price ||'€.');
            -- Mise à jour des stocks
            UPDATE Medicament SET Quantite = Quantite - quantite_per_med WHERE ordo.Nom_Commercial =
Nom_Commercial;
            UPDATE Achats_Ordonnances SET Paye = 'True' WHERE ordo.ID_Ordo = ID_Ordo;
            total_price := total_price + price;
            IF securite_Sociale IS NOT NULL AND mutuelle IS NOT NULL THEN
                total_price_client := 0;
            END IF;
            IF securite_Sociale IS NOT NULL AND mutuelle IS NULL THEN
                total_price_client := total_price_client + price * taux_remboursement / 100;
            END IF;
            IF securite_Sociale IS NULL AND mutuelle IS NOT NULL THEN
                total_price_client := total_price_client + price * 1 - (taux_remboursement / 100);
            END IF;
            IF securite_Sociale IS NULL AND mutuelle IS NULL THEN
                total_price_client := total_price_client + price;
            END IF;
        ELSE
            DBMS_OUTPUT.PUT_LINE('/!\ Le stock de '|| ordo.Nom_Commercial ||' ('|| quantite_meds
||'/'|| quantite_per_med ||') est insuffisant. Il n''est pas comptabilisé.');
        END IF;

    END LOOP;
    -- Completion de la ligne
    :NEW.Prix := total_price;
    :NEW.dDate := SYSTIMESTAMP;
    -- Mise à jour du budget
    UPDATE Structure SET Budget = Budget + total_price WHERE dDate = TRUNC(SYSDATE);
    DBMS_OUTPUT.PUT_LINE('Le prix final est ' || total_price || '€.');
    DBMS_OUTPUT.PUT_LINE('Le client doit payer ' || total_price_client || '€.');
END;
/
```

III. Jeu de données :

Les outils suivants ont été créer pour générer les jeux de données et vérifié leur cohérence :

- Python, avec le ficher gen.py
- La librairie Faker
- Le site https://mon-convertisseur.fr/calculateur-cle-numero-securite-sociale.php
- Le site https://www.insee.fr/fr/information/6051727 pour les données relatives aux départements
- Le modèle d'IA ChatGPT pour la correction de syntaxe et d'erreurs ainsi que la génération de la table des médicaments, car il semble que ces données ne soient pas très accessible sans devoir faire des recherches approfondies ou du nettoyage et de l'interprétation. Par conséquent, les données essayent de se rapprocher le plus possible de la réalité mais sont très probablement fictives. Cela à en général servi aussi à trouver des renseignements sur certaines fonctionnalités de la BDD Oracle.

Voici toutes les commandes d'insertions, l'outil de chargement massif n'étant pas disponible sur Oracle Live :

```
-- Insertions de données
INSERT INTO Structure (dDate, Budget) VALUES (TO_DATE('2024/12/05', 'YYYY/MM/DD'), 1000.50);

DELETE FROM Client WHERE ID ≥ 0;

INSERT INTO Client VALUES (0,'Alves','Simone',TO_DATE('1936/12/11',
'YYYY/MM/DD'),236126252555738,'Lopes',322103821859);
INSERT INTO Client VALUES (1,'Delahaye','Renée',TO_DATE('1973/11/22',
'YYYY/MM/DD'),273116300389818,'Lopes',264417325153);
INSERT INTO Client VALUES (2,'Bousquet','Thibault',TO_DATE('1934/08/27', 'YYYY/MM/DD'),134084301295609,'Labbé
Jacquot et Fils',448402954223);
INSERT INTO Client VALUES (3,'Da Silva','François',TO_DATE('2005/07/15', 'YYYY/MM/DD'),105079201284310,'Pottier
Lemoine SA',115194896849);
INSERT INTO Client VALUES (4,'Chevrolet','Sophie',TO_DATE('1949/08/20', 'YYYY/MM/DD'),NULL,NULL,NULL);
INSERT INTO Client VALUES (5,'Gomes','Jérôme',TO_DATE('1950/01/17', 'YYYY/MM/DD'),150017908159224,'Paris
S.A.',542332145059);
INSERT INTO Client VALUES (6,'Delorme','Élodie',TO_DATE('1955/04/26', 'YYYY/MM/DD'),255044623149449,'Colas
S.A.R.L.',352719028965);
INSERT INTO Client VALUES (7,'Lecomte','Emmanuelle',TO_DATE('1976/06/09', 'YYYY/MM/DD'),276068941961485,'Paris
S.A.',315782243799);
INSERT INTO Client VALUES (8,'Briand','Michelle',TO_DATE('1934/09/15', 'YYYY/MM/DD'),234096306386632,'Labbé
Jacquot et Fils',192360872179);
INSERT INTO Client VALUES (9,'Davis','Erica',TO_DATE('1946/07/08', 'YYYY/MM/DD'),NULL,NULL,NULL);
INSERT INTO Client VALUES (10,'Lebrun','Margaux',TO_DATE('1982/12/08',
'YYYY/MM/DD'),282126251687762,'Lesage',514494056788);
INSERT INTO Client VALUES (11,'박','성민',TO_DATE('1972/03/10', 'YYYY/MM/DD'),172038403173209,NULL,NULL);
INSERT INTO Client VALUES (12,'Couturier','Élisabeth',TO_DATE('1972/07/17', 'YYYY/MM/DD'),272070733453202,'Labbé
Jacquot et Fils',845737251037);
INSERT INTO Client VALUES (13,'Boyer','Victoire',TO_DATE('1950/06/08',
'YYYY/MM/DD'),250064902373833,'Jourdan',1420896819 27);
INSERT INTO Client VALUES (14,'Mendès','Eugène',TO_DATE('1963/10/09',
'YYYY/MM/DD'),163101434191352,'Jourdan',115999943650);
INSERT INTO Client VALUES (15,'Rivière','Sylvie',TO_DATE('1999/12/01', 'YYYY/MM/DD'),299123715939241,'Paris
S.A.',196180842154);
INSERT INTO Client VALUES (16,'Perrot','Laurence',TO_DATE('1949/08/25', 'YYYY/MM/DD'),249088841394060,'Colas
S.A.R.L.',371800976532);
INSERT INTO Client VALUES (17,'Fernandes','Michel',TO_DATE('1999/05/23', 'YYYY/MM/DD'),199050279581149,'Pottier
Lemoine SA',195437814623);
INSERT INTO Client VALUES (18,'Arnaud','Manon',TO_DATE('1946/06/06', 'YYYY/MM/DD'),246066704697082,'Nicolas
Descamps SARL',145640726442);
INSERT INTO Client VALUES (19,'Allard','Philippe',TO_DATE('1948/10/17', 'YYYY/MM/DD'),148103528833263,'Labbé
Jacquot et Fils',158347474690);
INSERT INTO Client VALUES (20,'Valette','Émile',TO_DATE('1939/01/28', 'YYYY/MM/DD'),139015212113267,'Nicolas
Descamps SARL',754780762482);
INSERT INTO Client VALUES (21,'Arnaud','Marthe',TO_DATE('1978/04/01',
'YYYY/MM/DD'),278040808181652,'Jourdan',763526336065);
INSERT INTO Client VALUES (22,'Dijoux','Matthieu',TO_DATE('1995/01/21', 'YYYY/MM/DD'),195011468947575,'Pottier
Lemoine SA',281705622727);
INSERT INTO Client VALUES (23,'Bègue','William',TO_DATE('1947/03/22',
'YYYY/MM/DD'),147035121730081,'Lesage',236716071219);
```

```
INSERT INTO Client VALUES (24,'Traore','Colette',TO_DATE('1963/03/02',
'YYYY/MM/DD'),263034803447188,'Lopes',348880205455);
INSERT INTO Client VALUES (25,'Besnard','Monique',TO_DATE('1968/08/09', 'YYYY/MM/DD'),268080734942367,'Labbé
Jacquot et Fils',195914569172);
INSERT INTO Client VALUES (26,'Teixeira','Madeleine',TO_DATE('1950/06/05',
'YYYY/MM/DD'),250069525062152,'Lopes',562095782460);
INSERT INTO Client VALUES (27,'Lacroix','Gabriel',TO_DATE('1986/01/24', 'YYYY/MM/DD'),186011220243588,'Pottier
Lemoine SA',975704398506);
INSERT INTO Client VALUES (28,'Chauvet','Claudine',TO_DATE('1944/11/06', 'YYYY/MM/DD'),244111300257538,'Colas
S.A.R.L.',247096491001);
INSERT INTO Client VALUES (29,'엄','진우',TO_DATE('1990/09/14', 'YYYY/MM/DD'),190091900418903,NULL,NULL);

DELETE FROM Medecin WHERE RPPS ≥ 0;

INSERT INTO Medecin VALUES (32801131885,'Maréchal','Anouk','21, rue Simone Gomes 90873 Marion');
INSERT INTO Medecin VALUES (80000050525,'Aubert','Valérie','634, rue de Humbert 40817 Saint Célina');
INSERT INTO Medecin VALUES (44882930295,'Allain','Guillaume','48, rue Besson 08752 Gomes');
INSERT INTO Medecin VALUES (11733531222,'Vallée','Margot','60, chemin Grégoire Bailly 53729 Launay');
INSERT INTO Medecin VALUES (83977507080,'Le Goff','Michelle','622, rue Nicole Fernandes 78306 LoiseauVille');
INSERT INTO Medecin VALUES (29452752713,'Guillaume','Marcel','46, avenue de Delorme 19810 Marchand');
INSERT INTO Medecin VALUES (38274513141,'Mary','Danielle','84, avenue Philippe 84214 Lagarde');
INSERT INTO Medecin VALUES (10852261065,'Mahe','Étienne','44, rue Guibert 25465 Duval');
INSERT INTO Medecin VALUES (58747568756,'Alexandre','Henri','85, rue Adèle Charrier 97391 Lecoq');
INSERT INTO Medecin VALUES (30379166296,'Alves','Sébastien','89, boulevard Langlois 72963 Neveu');
INSERT INTO Medecin VALUES (51280855754,'Lenoir','Marianne','8, rue de Toussaint 97355 Fournier-la-Forêt');
INSERT INTO Medecin VALUES (88220503674,'Cordier','Capucine','82, rue de Lemoine 92496 Weberdan');
INSERT INTO Medecin VALUES (30045822729,'Roussel','Théodore','75, boulevard Launay 90199 Reynaud');
INSERT INTO Medecin VALUES (51522934825,'Baudry','Élisabeth','4, chemin Claude Carlier 66290 LucasVille');
INSERT INTO Medecin VALUES (79363064483,'Moreau','Cécile','avenue Rodrigues 71959 Bruneau');
INSERT INTO Medecin VALUES (17979908410,'Lefebvre','Alice','41, rue Lambert 19600 Sainte Jacques');
INSERT INTO Medecin VALUES (92661073315,'Barbier','Roger','75, rue de Gomez 31711 Chartierboeuf');
INSERT INTO Medecin VALUES (30864955446,'Cordier','Maurice','88, rue Devaux 59842 BoyerVille');
INSERT INTO Medecin VALUES (86750317269,'Chauveau','Denis','rue Michèle Charrier 11239 Lambert-sur-Guilbert');
INSERT INTO Medecin VALUES (40644897348,'Allain','Émile','698, rue Claudine Collet 97438 Saint Dorothéeboeuf');
INSERT INTO Medecin VALUES (43927171825,'Gay','Caroline','rue Olivier Marchal 87664 Louis-sur-Robert');
INSERT INTO Medecin VALUES (97740383732,'Bernier','Victoire','rue Boyer 71864 Leduc');
INSERT INTO Medecin VALUES (89070747399,'Chartier','Cécile','47, rue Berger 11408 Martin');
INSERT INTO Medecin VALUES (39734841774,'Rey','Inès','2, rue de Blanchet 75185 Sainte Diane');
INSERT INTO Medecin VALUES (34666413852,'Benoit','Margot','713, rue Michèle Dumont 35366 Jacquet-sur-Perrier');
INSERT INTO Medecin VALUES (83625354668,'Martins','Émile','1, rue de Dijoux 42443 Daniel');
INSERT INTO Medecin VALUES (64188706957,'Dijoux','Margot','89, rue Gillet 76500 Normand');
INSERT INTO Medecin VALUES (39009010342,'Delannoy','Aimé','8, avenue de Millet 44893 BourgeoisVille');
INSERT INTO Medecin VALUES (59537221837,'Faure','Suzanne','39, boulevard Marie Poirier 69137 De Sousa');
INSERT INTO Medecin VALUES (27997895058,'Bruneau','Étienne','62, rue de Girard 57443 Sainte Corinne');

DELETE FROM Medicament WHERE CIP ≥ 0;

INSERT INTO Medicament VALUES ('3400935042818','Doliprane500',65,2.000,10,500,'False');
INSERT INTO Medicament VALUES ('3400935042870','Doliprane1000',65,2.500,10,1000,'False');
INSERT INTO Medicament VALUES ('3400930018967','Spasfon Lyoc',30,4.500,10,80,'False');
INSERT INTO Medicament VALUES ('3400932768953','Ibuprofène200',65,1.800,10,200,'False');
INSERT INTO Medicament VALUES ('3400932769011','Ibuprofène400',65,2.300,10,400,'False');
INSERT INTO Medicament VALUES ('3400932958392','Efferalgan500',65,2.100,10,500,'False');
INSERT INTO Medicament VALUES ('3400933174791','Efferalgan1000',65,2.800,10,1000,'False');
INSERT INTO Medicament VALUES ('3400935042467','Humex Rhume',0,5.500,10,500,'False');
INSERT INTO Medicament VALUES ('3400933848603','Gaviscon',30,3.200,10,250,'False');
INSERT INTO Medicament VALUES ('3400937888911','Smecta',30,4.000,10,3000,'False');
INSERT INTO Medicament VALUES ('3400934707619','Augmentin',65,9.000,10,1000,'True');
INSERT INTO Medicament VALUES ('3400935329681','Amoxicilline',65,8.000,10,500,'True');
INSERT INTO Medicament VALUES ('3400930115738','Zithromax',65,14.500,10,250,'True');
INSERT INTO Medicament VALUES ('3400935273792','Paroxetine',65,12.000,10,20,'True');
INSERT INTO Medicament VALUES ('3400932277886','Levothyrox',65,4.000,10,0.05,'True');
INSERT INTO Medicament VALUES ('3400930049056','Xanax',65,3.500,10,0.25,'True');
INSERT INTO Medicament VALUES ('3400930253187','Lexomil',65,5.000,10,6,'True');
INSERT INTO Medicament VALUES ('3400930568267','Valium',65,2.700,10,10,'True');
INSERT INTO Medicament VALUES ('3400930004356','Magné B6',65,4.000,10,475,'False');
INSERT INTO Medicament VALUES ('3400933661165','Fervex Adultes',0,6.200,10,750,'False');
INSERT INTO Medicament VALUES ('3400934756877','Strepsils Menthol',0,4.300,10,32,'False');
INSERT INTO Medicament VALUES ('3400937007258','Imodium',30,3.800,10,2,'False');
INSERT INTO Medicament VALUES ('3400937748654','Tiorfan',65,7.000,10,100,'True');
INSERT INTO Medicament VALUES ('3400932651712','Clamoxyl',65,10.500,10,1000,'True');
INSERT INTO Medicament VALUES ('3400933226252','Spifen',65,4.500,10,400,'False');
INSERT INTO Medicament VALUES ('3400930007764','Nurofen',65,4.000,10,400,'False');
INSERT INTO Medicament VALUES ('3400933695672','Orocal',65,3.200,10,500,'False');
INSERT INTO Medicament VALUES ('3400932858999','Pradaxa',65,60.000,10,150,'True');
INSERT INTO Medicament VALUES ('3400933163474','Eliquis',65,85.000,10,5,'True');
INSERT INTO Medicament VALUES ('3400930249746','Lovenox',65,35.000,10,40,'True');

DELETE FROM Comm_med WHERE ID_COMMANDE ≥ 0;
DELETE FROM Commande WHERE ID ≥ 0;
-- Commandes expirées
INSERT INTO Commande (ID, Date_commande, Date_livraison, Date_expiration) VALUES (0, TO_TIMESTAMP('2024/09/26',
'YYYY/MM/DD HH24:MI:SS'), TO_TIMESTAMP('2024/10/06 09:48:05', 'YYYY/MM/DD HH24:MI:SS'), TO_TIMESTAMP('2024/10/21
00:00:00', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Commande (ID, Date_commande, Date_livraison, Date_expiration) VALUES (1, TO_TIMESTAMP('2024/11/27',
'YYYY/MM/DD HH24:MI:SS'), TO_TIMESTAMP('2024/12/10 08:45:25', 'YYYY/MM/DD HH24:MI:SS'), TO_TIMESTAMP('2024/12/25
00:00:00', 'YYYY/MM/DD HH24:MI:SS'));
```

```
INSERT INTO Commande (ID, Date_commande, Date_livraison, Date_expiration) VALUES (2, TO_TIMESTAMP('2024/11/20',
'YYYY/MM/DD HH24:MI:SS'), TO_TIMESTAMP('2024/12/01 15:48:15', 'YYYY/MM/DD HH24:MI:SS'), TO_TIMESTAMP('2024/12/16
00:00:00', 'YYYY/MM/DD HH24:MI:SS'));
-- Commandes faites mais non livrées
INSERT INTO Commande (ID, Date_commande, Date_livraison, Date_expiration) VALUES (3, TO_TIMESTAMP('2024/12/08',
'YYYY/MM/DD HH24:MI:SS'), NULL, NULL);
INSERT INTO Commande (ID, Date_commande, Date_livraison, Date_expiration) VALUES (4, TO_TIMESTAMP('2024/12/07',
'YYYY/MM/DD HH24:MI:SS'), NULL, NULL);
INSERT INTO Commande (ID, Date_commande, Date_livraison, Date_expiration) VALUES (5, TO_TIMESTAMP('2024/11/20',
'YYYY/MM/DD HH24:MI:SS'), NULL, NULL);
-- Commandes faites et livrées
INSERT INTO Commande (ID, Date_commande, Date_livraison, Date_expiration) VALUES (6, TO_TIMESTAMP('2024/09/24',
'YYYY/MM/DD HH24:MI:SS'), TO_TIMESTAMP('2024/10/05 06:18:06', 'YYYY/MM/DD HH24:MI:SS'), NULL);
INSERT INTO Commande (ID, Date_commande, Date_livraison, Date_expiration) VALUES (7, TO_TIMESTAMP('2024/09/04',
'YYYY/MM/DD HH24:MI:SS'), TO_TIMESTAMP('2024/09/17 04:59:09', 'YYYY/MM/DD HH24:MI:SS'), NULL);
INSERT INTO Commande (ID, Date_commande, Date_livraison, Date_expiration) VALUES (8, TO_TIMESTAMP('2024/09/22',
'YYYY/MM/DD HH24:MI:SS'), TO_TIMESTAMP('2024/10/02 10:34:32', 'YYYY/MM/DD HH24:MI:SS'), NULL);

DELETE FROM Achats_Ordonnances WHERE ID_ORDO >= 0;

INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES (0,'Spasfon
Lyoc',NULL,320.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(0,'Ibuprofène400',NULL,1200.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(0,'Efferalgan500',NULL,1000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(0,'Levothyrox',51522934825,0.20,TO_DATE('2024/12/06', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(0,'Efferalgan1000',NULL,2000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(1,'Nurofen',NULL,1200.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(1,'Eliquis',83977507080,20.00,TO_DATE('2024/12/12', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(1,'Efferalgan1000',NULL,5000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(1,'Lexomil',83977507080,30.00,TO_DATE('2024/12/12', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(2,'Tiorfan',64188706957,400.00,TO_DATE('2024/12/18', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(2,'Valium',64188706957,50.00,TO_DATE('2024/12/18', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(2,'Xanax',64188706957,0.75,TO_DATE('2024/12/18', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(3,'Doliprane1000',NULL,5000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(3,'Lexomil',38274513141,30.00,TO_DATE('2024/12/10', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES (4,'Magné
B6',NULL,1900.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(4,'Efferalgan1000',NULL,3000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(4,'Efferalgan500',NULL,1000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(5,'Doliprane1000',NULL,2000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES (5,'Strepsils
Menthol',NULL,64.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(6,'Xanax',86750317269,0.50,TO_DATE('2024/12/14', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(6,'Doliprane1000',NULL,2000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(6,'Efferalgan500',NULL,1000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(6,'Clamoxyl',86750317269,4000.00,TO_DATE('2024/12/14', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(7,'Orocal',NULL,2500.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(7,'Levothyrox',30864955446,0.10,TO_DATE('2024/12/12', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(8,'Ibuprofène400',NULL,1600.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(8,'Efferalgan1000',NULL,2000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(8,'Amoxicilline',10852261065,1000.00,TO_DATE('2024/12/18', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(9,'Lovenox',86750317269,120.00,TO_DATE('2024/12/17', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(9,'Augmentin',86750317269,2000.00,TO_DATE('2024/12/17', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(9,'Valium',86750317269,40.00,TO_DATE('2024/12/17', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(10,'Spifen',NULL,2000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(10,'Efferalgan1000',NULL,5000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(11,'Lexomil',44882930295,12.00,TO_DATE('2024/12/12', 'YYYY/MM/DD HH24:MI:SS'));
```

```
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(11,'Clamoxyl',44882930295,3000.00,TO_DATE('2024/12/12', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(11,'Ibuprofène200',NULL,400.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES (11,'Humex
Rhume',NULL,2500.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES (11,'Strepsils
Menthol',NULL,64.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES (12,'Strepsils
Menthol',NULL,64.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(12,'Lovenox',51522934825,160.00,TO_DATE('2024/12/15', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(13,'Efferalgan500',NULL,1500.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(13,'Orocal',NULL,1000.00,SYSDATE);
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(14,'Eliquis',83625354668,25.00,TO_DATE('2024/12/14', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(14,'Zithromax',83625354668,750.00,TO_DATE('2024/12/14', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(14,'Tiorfan',83625354668,500.00,TO_DATE('2024/12/14', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(14,'Augmentin',83625354668,2000.00,TO_DATE('2024/12/14', 'YYYY/MM/DD HH24:MI:SS'));
INSERT INTO Achats_Ordonnances (Id_Client, Nom_commercial, RPPS, Posologie, dDate) VALUES
(14,'Amoxicilline',83625354668,2000.00,TO_DATE('2024/12/14', 'YYYY/MM/DD HH24:MI:SS'));

-- MANUAL

INSERT INTO Comm_Med (ID, CIP, ID_Commande, Quantite, Prix_HT_UNIT) VALUES (0, '3400935042870', 3, 100, 1.5);
INSERT INTO Comm_Med (ID, CIP, ID_Commande, Quantite, Prix_HT_UNIT) VALUES (1, '3400933174791', 3, 13, 1.8);
INSERT INTO Comm_Med (ID, CIP, ID_Commande, Quantite, Prix_HT_UNIT) VALUES (2, '3400933163474', 3, 2, 80);

INSERT INTO Comm_Med (ID, CIP, ID_Commande, Quantite, Prix_HT_UNIT) VALUES (3, '3400932858999', 4, 3, 30);

INSERT INTO Paiement (ID_Client, dDate) VALUES (4, SYSDATE);
INSERT INTO Paiement (ID_Client, dDate) VALUES (8, SYSDATE);
INSERT INTO Paiement (ID_Client, dDate) VALUES (9, SYSDATE);
INSERT INTO Paiement (ID_Client, dDate) VALUES (12, SYSDATE);
INSERT INTO Paiement (ID_Client, dDate) VALUES (16, SYSDATE);
INSERT INTO Paiement (ID_Client, dDate) VALUES (20, SYSDATE);
INSERT INTO Paiement (ID_Client, dDate) VALUES (8, SYSDATE);
-- On ajoute encore une fois le client 8 car cela créera un autre achat, de valeur 0.
```

IV. Requêtes SQL :

Voici la liste des requêtes SQL comme décrit en phase I. Voici le code :

```
-- REQUETES SQL
-- #01. Ajouter un paiement et mettre à jour les tables
INSERT INTO Paiement (ID_Client, dDate) VALUES (0, SYSDATE);
-- Va générer une erreur car l'ordonnance associée est expirée
INSERT INTO Paiement (ID_Client, dDate) VALUES (11, SYSDATE);
SELECT * FROM Structure;
-- #02. Quelles sont les 5 dernières transactions ?
SELECT * FROM Paiement ORDER BY dDate DESC FETCH FIRST 5 ROWS ONLY;
-- #03. Quel est le médicament le moins cher et qui est en stock ?
SELECT Nom_Commercial, Quantite, Prix FROM Medicament WHERE Quantite > 0 ORDER BY Prix ASC FETCH FIRST 1 ROWS
ONLY;
-- #04. Quels sont les médicaments les plus/moins en stock ?
SELECT Nom_Commercial, Quantite FROM Medicament ORDER BY Quantite ASC;
-- #05. Quel client à effectuer le plus d'achat sur l'année et pour quel montant ?
SELECT ID_Client, COUNT(*) AS Nombre_Achats FROM Paiement
WHERE EXTRACT(YEAR FROM dDate) = 2024
GROUP BY ID_Client ORDER BY Nombre_Achats DESC FETCH FIRST 1 ROWS ONLY;
-- #06. Quel client à dépenser le plus sur l'année et pour quel montant ?
SELECT ID_Client, SUM(Prix) AS Total_Achats FROM Paiement
WHERE EXTRACT(YEAR FROM dDate) = 2024
GROUP BY ID_Client ORDER BY Total_Achats DESC FETCH FIRST 1 ROWS ONLY;
-- #07. Quel est le chiffre d'affaires du mois/entre deux dates?
SELECT SUM(Budget) AS Chiffre_Affaire_Mois FROM Structure WHERE TRUNC(dDate, 'MM') = TRUNC(SYSDATE, 'MM');
SELECT SUM(Budget) AS Chiffre_Affaire FROM Structure WHERE dDate BETWEEN TO_DATE('2024/01/01', 'YYYY/MM/DD') AND
TO_DATE('2024/12/31', 'YYYY/MM/DD');
-- #08. Quel livraisons sont en attente (commandé mais pas livré) ?
SELECT * FROM Commande WHERE Date_livraison IS NULL;
-- #09. Quels sont les lots de médicaments expirés à retirer ?
SELECT ID, Date_expiration, Lot FROM Commande WHERE Date_expiration ⩽ SYSTIMESTAMP;
-- #10. Quelle est la liste de toutes les ordonnances délivrées par un médecin spécifique ?
SELECT * FROM Achats_Ordonnances WHERE RPPS = 86750317269;
SELECT * FROM Achats_Ordonnances WHERE RPPS = 83625354668;
-- #11. Quels médicaments ont été commandés dans une commande spécifique ?
SELECT * FROM Comm_Med WHERE ID_COMMANDE = 3;
-- #12. Mettre à jour la commande en définissant la date de livraison, la date d'expiration est définie et le
stock est mis à jour
UPDATE Commande SET Date_livraison = SYSTIMESTAMP WHERE Commande.ID = 3;
SELECT * FROM Structure
-- #13. Quel est le prix d'une commande ?
SELECT SUM(QUANTITE * PRIX_HT_UNIT) AS Prix_Total FROM Comm_Med WHERE ID_COMMANDE = 3;
```

```
-- #14. Quel est le prix total (hors ss et mutuelle) d'une ordonnance?
SELECT ao.ID_Client, SUM(m.Prix * ao.Posologie / m.Posologie) AS total_price FROM Achats_Ordonnances ao
JOIN Medicament m ON ao.Nom_commercial = m.Nom_Commercial
GROUP BY ao.ID_Client;
-- #15. Quels médicaments ont été commandés mais pas encore livré ?
SELECT ID_Commande, Nom_Commercial FROM Comm_Med JOIN Commande ON Comm_Med.ID_Commande = Commande.ID JOIN
Medicament ON Comm_med.CIP = Medicament.CIP WHERE Commande.Date_livraison IS NULL;
-- #16. Qui sont les clients qui ont une sécurité sociale mais pas de mutuelle ?
SELECT ID, Nom, Prenom, Date_Naissance FROM Client WHERE Securite_Sociale IS NOT NULL AND Mutuelle IS NULL;
-- #17. Quels sont les médicaments les plus/moins bien remboursés par la SS ?
SELECT CIP, Nom_Commercial, Taux_remboursement FROM Medicament ORDER BY Taux_remboursement;
-- #18. Quel est l'historique des achats d'un client spécifique ?
SELECT dDate, ID_Client, Prix FROM Paiement WHERE ID_Client = 8 ORDER BY dDATE;
-- #19. Combien un client dépense en moyenne ?
SELECT ROUND(AVG(m.Prix * ao.Posologie / m.Posologie), 2) AS Depense_moyenne_clients FROM Achats_Ordonnances ao
JOIN Medicament m ON ao.Nom_commercial = m.Nom_Commercial;
-- #20. Qui sont les clients qui ont dépenser plus que la moyenne ?
SELECT c.ID, c.Nom, c.Prenom, ROUND(SUM(m.Prix * ao.Posologie / m.Posologie), 2) AS Depense_totale FROM Client c
JOIN Achats_Ordonnances ao ON c.ID = ao.ID_Client
JOIN Medicament m ON ao.Nom_Commercial = m.Nom_Commercial
GROUP BY c.ID, c.Nom, c.Prenom
HAVING SUM(m.Prix * ao.Posologie / m.Posologie) >
(SELECT AVG(m.Prix * ao.Posologie / m.Posologie) FROM Achats_Ordonnances ao JOIN Medicament m ON
ao.Nom_commercial = m.Nom_Commercial);
```

V. Méta-Données

Voici les commandes répondant aux consignes :

```
-- Meta-Données

-- Contraintes d'integrité

SELECT
c.TABLE_NAME,c.CONSTRAINT_NAME,c.CONSTRAINT_TYPE,c.STATUS,c.DEFERRABLE,c.REFERENCED_TABLE_NAME,cc.COLUMN_NAME
FROM USER_CONSTRAINTS c

JOIN USER_CONS_COLUMNS cc ON c.CONSTRAINT_NAME = cc.CONSTRAINT_NAME

ORDER BY c.TABLE_NAME,c.CONSTRAINT_TYPE,c.CONSTRAINT_NAME;

-- Triggers

SELECT t.TABLE_NAME,t.TRIGGER_NAME,t.TRIGGER_TYPE,t.TRIGGERING_EVENT,t.STATUS,t.DESCRIPTION FROM USER_TRIGGERS t

ORDER BY t.TABLE_NAME, t.TRIGGER_NAME;


-- Index

SELECT i.TABLE_NAME,i.INDEX_NAME,i.UNIQUENESS,c.COLUMN_NAME FROM USER_INDEXES i

JOIN USER_IND_COLUMNS c ON i.INDEX_NAME = c.INDEX_NAME

ORDER BY i.TABLE_NAME, i.INDEX_NAME;


-- Utilisateurs

SELECT grantee, table_name, privilege

FROM USER_TAB_PRIVS ORDER BY grantee, table_name;
```

Les deux autres scripts permettent d’obtenir les index, une structure gérer par la base de donnée pour s’assurer par exemple que certains clés sont uniques, pour optimiser les opérations de jointures ou bien lors des filtrages. Le second script permet d’obtenir les rôles utilisateurs, même si sur Oracle Live il n’y en a qu’un seul.