**Projet de Fin de Module : Bases de Données Avancées & PL/SQL﻿**  
Titre du Projet : Conception et Réalisation d'une Marketplace

E-commerce Multi-vendeurs﻿

**Réalisé par :**

El Houcine Rahmouni, Zakaria Oulamine

**Filière :**

S5 Sciences de l'Informatique

**Encadré par :**

Hamza Er-rahmouny

**Année universitaire** : 2025/2026

**Remerciements﻿**

Avant d'entamer la présentation de ce travail, nous tenons à exprimer notre profonde gratitude et nos plus sincères remerciements à notre professeur, M. Hamza Er-rahmouny, pour ses conseils précieux, sa disponibilité et son encadrement rigoureux tout au long de la réalisation de ce projet.﻿

Nos remerciements s'étendent également à l'Université Cadi Ayyad et au corps professoral de la Faculté des Sciences Semlalia, qui nous ont fourni un environnement d'apprentissage stimulant et les ressources nécessaires à notre formation.﻿

Enfin, nous souhaitons souligner l'excellente collaboration et l'esprit d'équipe qui ont animé notre groupe de travail. Ce projet est le fruit d'un effort commun et d'un engagement partagé.﻿

# Sommaire

Contents

[**Remerciements** 1](#_Toc211974635)

[Sommaire 2](#_Toc211974636)

[**Introduction Générale** 3](#_Toc211974637)

[**1.** **Contexte Général du Projet﻿** 3](#_Toc211974638)

[**2.** **Objectifs du Projet﻿** 3](#_Toc211974639)

[**Chapitre 1 : Étude Préalable et Analyse des Besoins﻿** 4](#_Toc211974640)

[**1.1.** **Contexte Général du Projet﻿** 4](#_Toc211974641)

[**1.2.** **Objectifs du Projet﻿** 4](#_Toc211974642)

[**1.2.1. Exigences Fonctionnelles﻿** 5](#_Toc211974643)

[**1.2.2. Exigences Techniques﻿** 5](#_Toc211974644)

[**Conclusion du chapitre﻿** 5](#_Toc211974645)

[**Chapitre 2 : Conception du Système d'Information** 6](#_Toc211974646)

[**2.1.** **Choix de la Méthodologie de Conception** 6](#_Toc211974647)

[**2.1.1. Présentation de la méthode MERISE** 6](#_Toc211974648)

[**2.1.2.Présentation de la méthode UML** 7](#_Toc211974649)

[**2.1.3. Justification du choix de MERISE** 8](#_Toc211974650)

[**2.2. Modèle Conceptuel de Données (MCD)** 8](#_Toc211974651)

[**2.2.1. Description et justification des entités du système e-commerce multi-vendeurs** 8](#_Toc211974652)

[**2.3. Modèle Logique de Données (MLD)** 14](#_Toc211974653)

# **Introduction Générale**

## **Contexte Général du Projet﻿**

Le présent rapport est élaboré dans le cadre du projet de fin de module de "Bases de Données Avancées & PL/SQL", une composante essentielle du cinquième semestre de la filière Sciences de l'Informatique. Ce projet représente une opportunité fondamentale de mettre en application les concepts théoriques et les compétences pratiques que nous avons acquis en matière de modélisation, de conception de bases de données relationnelles et de programmation avancée.﻿

## **Objectifs du Projet﻿**

L'objectif principal qui nous a été confié est la conception et la réalisation d'une plateforme de commerce électronique de type marketplace multi-vendeurs. Le système doit permettre à divers vendeurs de gérer leurs propres boutiques virtuelles, d'ajouter et de suivre leurs produits, tout en offrant aux clients une expérience d'achat fluide. Le cœur technique du projet repose sur une base de données Oracle robuste et une logique métier complexe qui sera implémentée en PL/SQL, conformément aux exigences du cahier des charges.﻿

# **Chapitre 1 : Étude Préalable et Analyse des Besoins﻿**

**Introduction du chapitre﻿**

Ce premier chapitre est consacré à l'étude préalable du projet. Il a pour objectif de définir le cadre général, de présenter le contexte métier et d'identifier de manière claire et précise les besoins fonctionnels et techniques auxquels notre solution devra répondre. Cette phase est fondamentale car elle conditionne la pertinence et le succès des étapes de conception et de développement qui suivront.﻿

## **Contexte Général du Projet﻿**

Le projet s'inscrit dans un contexte économique moderne où la digitalisation des activités commerciales est devenue une nécessité stratégique. Il nous est demandé de réaliser une plateforme de type **marketplace e-commerce multi-vendeurs**. Contrairement à un site e-commerce classique qui ne présente les produits que d'une seule entité, une marketplace est un espace en ligne qui met en relation une multitude de vendeurs indépendants avec une large base de clients. Cette approche permet d'offrir une gamme de produits très variée et de créer un écosystème commercial dynamique.﻿

Sur le plan académique, ce projet constitue une application pratique et intégrale des concepts étudiés dans le module "Bases de Données Avancées & PL/SQL". Il nous permettra de mobiliser nos compétences en modélisation de données via la méthode MERISE, en programmation de la logique métier avec PL/SQL (triggers, procédures, fonctions), et en développement d'interfaces web riches avec Oracle APEX.﻿

## **Objectifs du Projet﻿**

À partir de l'analyse du cahier des charges, nous avons identifié un ensemble d'objectifs principaux que notre application doit atteindre pour être considérée comme fonctionnelle et complète. Ces objectifs se décomposent en exigences fonctionnelles et non fonctionnelles.﻿

### **1.2.1. Exigences Fonctionnelles﻿**

Le système doit impérativement prendre en charge les fonctionnalités suivantes :﻿

* **Gestion des Vendeurs :﻿** Permettre aux vendeurs de créer et gérer leur profil, ainsi que de mettre en ligne et suivre leurs propres produits.﻿
* **Gestion des Produits et Catégories :﻿** Organiser les produits dans un catalogue structuré par catégories pour faciliter la navigation des clients.﻿
* **Gestion des Clients :﻿** Offrir aux clients la possibilité de créer un compte, de parcourir les produits, de les ajouter à un panier et de passer une commande.﻿
* **Gestion des Commandes :﻿** Assurer le traitement complet d'une commande, de sa création à sa validation, même si elle contient des produits de plusieurs vendeurs différents.﻿
* **Paiement et Facturation :﻿** Calculer automatiquement le montant total d'une commande, en appliquant les remises éventuelles, et permettre au client d'effectuer le paiement, générant ainsi une facture.﻿
* **Gestion de l'Expédition :﻿** Permettre le suivi de l'état d'avancement de l'expédition de chaque commande (en attente, en cours, livrée) en association avec un transporteur.﻿
* **Gestion des Avis Clients :﻿** Donner la possibilité aux clients de noter et de laisser des commentaires sur les produits qu'ils ont achetés.﻿

### **1.2.2. Exigences Techniques﻿**

Le cahier des charges impose l'utilisation exclusive de la stack technologique Oracle :﻿

* Système de Gestion de Base de Données : **Oracle Database XE**.﻿
* Langage de la logique métier : **PL/SQL**.﻿
* Framework de développement de l'interface : **Oracle APEX**.﻿
* Outil de modélisation : **Draw.io** pour les diagrammes MERISE ou UML.﻿

**Conclusion du chapitre﻿**  
Cette étude préalable nous a permis de cerner avec précision le périmètre du projet et de définir une liste claire d'objectifs. Le chapitre suivant sera consacré à la conception détaillée du système d'information en réponse à ces exigences.﻿

# **Chapitre 2 : Conception du Système d'Information**

**Introduction du chapitre**

Après avoir défini le périmètre et les objectifs de notre projet dans le chapitre précédent, nous abordons maintenant la phase de conception. Ce chapitre est le cœur de notre travail de modélisation. Il a pour but de traduire les besoins fonctionnels identifiés en une structure de données formelle et robuste. Pour ce faire, nous allons justifier le choix de notre méthodologie de conception, puis présenter en détail les modèles conceptuel et logique des données qui serviront de fondation à notre base de données Oracle.

## **Choix de la Méthodologie de Conception**

### **2.1.1. Présentation de la méthode MERISE**

1. **Séparation claire des niveaux d’abstraction :**
   * Le **MCD (Modèle Conceptuel de Données)** décrit les informations du système sans se préoccuper de la technique (c’est le “quoi”).
   * Le **MLD (Modèle Logique de Données)** adapte le modèle à un SGBD précis (le “comment”).
   * Cette séparation permet de **mieux comprendre le système avant de le coder**.
2. **Cohérence et fiabilité :**  
   La méthode Merise permet de **détecter les incohérences et les redondances** dans les données avant la création de la base.  
   Ainsi, les erreurs sont corrigées très tôt, ce qui améliore la **qualité et la fiabilité du modèle.**
3. **Communication facilitée :**  
   Les diagrammes (MCD, MLD) sont **visuels et faciles à comprendre**, ce qui facilite la communication entre les développeurs, les analystes et les utilisateurs.
4. **Gestion des dépendances fonctionnelles :**  
   Merise permet d’identifier clairement les **liens logiques entre les entités**, les **contraintes d’intégrité** et les **cardinalités.**
5. **Documentation complète :**  
   Le **dictionnaire de données** fournit une description précise de chaque donnée (nom, type, taille, signification, contraintes...), ce qui constitue une **base de documentation fiable** pour tout le projet.

### **2.1.2.Présentation de la méthode UML**

**Objectifs différents :**

* La méthode UML est principalement conçue pour modéliser l’ensemble d’un système logiciel, incluant les comportements, les interactions et la logique orientée objet.
* La méthode Merise, quant à elle, est spécialisée dans la modélisation des données et des traitements d’un système d’information.

Ainsi, pour un projet centré sur la conception d’une base de données, Merise s’avère plus pertinente et mieux adaptée.

Adaptation au modèle relationnel

* UML décrit des classes et des objets, ce qui correspond davantage à la programmation orientée objet (Java, C++, etc.).
* Merise, en revanche, décrit des entités, des associations et des attributs, ce qui correspond directement à un modèle relationnel (tables, clés, relations).

Par conséquent, la transformation du MCD → MLD → SQL est plus simple, logique et claire avec Merise.

Simplicité et clarté pour la modélisation de données

* Merise propose une notation simple et intuitive basée sur les entités, les associations et les cardinalités.
* UML, en comparaison, peut devenir plus complexe et verbeuse pour les utilisateurs qui souhaitent uniquement concevoir une base de données sans programmation orientée objet.

Pertinence pédagogique et méthodologique

* Dans les contextes éducatifs ou administratifs, comme les projets d’analyse, de gestion ou de comptabilité, Merise offre une démarche méthodologique claire et structurée.
* UML, de son côté, est davantage utilisée dans les projets de développement orienté objet ou dans les architectures logicielles complexes.

### **2.1.3. Justification du choix de MERISE**

Dans le cadre de notre projet, nous avons choisi d’utiliser la méthode Merise plutôt que la méthode UML.

Ce choix s’explique par la nature de notre travail, qui porte principalement sur la modélisation et la conception d’une base de données relationnelle.

La méthode Merise est particulièrement adaptée à ce type de projet, car elle permet de décrire de manière claire et progressive les données (MCD, MLD, MPD) et leurs relations.

À l’inverse, UML est davantage orientée vers la programmation orientée objet et la modélisation de systèmes logiciels complets, ce qui dépasse les besoins de notre projet.

**Ainsi, l’utilisation de Merise nous a permis d’obtenir une modélisation plus simple, cohérente et directement exploitable pour la création de la base de données.**

## **2.2. Modèle Conceptuel de Données (MCD)**

Le MCD est une représentation abstraite de la structure des données du système, indépendante de toute technologie. Il identifie les objets d'information principaux (entités) et les liens sémantiques qui les unissent (relations).

### **2.2.1. Description et justification des entités du système e-commerce multi-vendeurs**

**1. VENDEUR**

* Justification : Cette entité représente les vendeurs de la plateforme. Chaque vendeur dispose d'un compte personnel lui permettant de gérer sa boutique et ses produits.
* Attributs : id\_vendeur (PK), nom, email, telephone, nom\_boutique, date\_inscription, status\_compte

**2. PRODUIT**

* Justification : Le produit constitue l'élément central de la plateforme. Chaque produit possède des caractéristiques propres et appartient à un seul vendeur.
* Attributs : id\_produit (PK), nom, description, quantite\_stock, date\_creation

**3. CATEGORIE**

* Justification : Les produits sont regroupés par catégories pour faciliter la navigation et la recherche.
* Attributs : id\_categorie (PK), nom, description

**4. CLIENT**

* Justification : Le client est l'utilisateur final de la plateforme. Il consulte les produits et effectue des achats.
* Attributs : id\_client (PK), nom, email, telephone, date\_inscription

**5. COMMANDE**

* Justification : Une commande enregistre chaque achat effectué par un client.
* Attributs: id\_commande (PK), date, status, montant\_total, adress

**6. PAIEMENT**

* Justification : Permet d'assurer le suivi des transactions financières liées aux commandes.
* Attributs : id\_paiement (PK), montant, mode\_paiement, status, date

**7. TRANSPORTEUR**

* Justification : Cette entité représente les sociétés chargées de la livraison.
* Attributs : id\_transport (PK), nom, contact, site\_web

**8. FACTURE\_VENDEUR**

* Justification : Cette entité représente la facture générée pour chaque client lorsqu'il achat un produit.
* Attributs : id\_facture (PK), date\_facture, montant\_total, statut\_paiement, mode\_paiement

**2.2.2. Relations & cardinalités (MCD Merise - description)**

VENDEUR ---- (Proposer) ---- PRODUIT

* Type : Association simple
* Cardinalités : VENDEUR (0,N) ---> PROPOSER ---> (1,1) PRODUIT

Un vendeur peut proposer 0 à N produits. Un produit appartient obligatoirement à un seul vendeur.

CATEGORIE ---- (Classer) ---- PRODUIT

* Type : Association simple
* Cardinalités : PRODUIT (1,1) ---> CLASSER ---> (1,N) CATEGORIE

Un produit appartient à une seule catégorie. Une catégorie peut contenir 1 à N produits.

PRODUIT ---- (inclure) ---- COMMANDE

* Type : Association porteuse (relation n-n)
* Attributs portés : quantite, prix\_unitaire, remis
* Cardinalités : PRODUIT (0,N) ---> INCLURE ---> (1,N) COMMANDE

Une commande contient au moins un produit. Un produit peut être inclus dans 0 à N commandes.

CLIENT ---- (Passer) ---- COMMANDE

* Type : Association simple
* Cardinalités : CLIENT (0,N) ---> PASSER ---> (1,1) COMMANDE

Un client peut passer 0 à N commandes. Une commande est passée par un seul client.

CLIENT ---- (evaluer) ---- PRODUIT

* Type : Association porteuse (relation n-n)
* Attributs portés : note, commentaire, date\_avis
* Cardinalités : CLIENT (0,N) ---> EVALUER ---> (0,N) PRODUIT

Un client peut évaluer 0 à N produits. Un produit peut recevoir 0 à N évaluations.

COMMANDE ---- (Regler) ---- PAIEMENT

* Type : Association simple
* Cardinalités : COMMANDE (1,N) ---> REGLER ---> (1,1) PAIEMENT

Un paiement concerne une seule commande. Une commande peut avoir 1 à N paiements.

COMMANDE ---- (Expédier) ---- TRANSPORTEUR

* Type : Association porteuse
* Attributs portés : numero\_suivi, statut\_avancement, date\_expedition
* Cardinalités : COMMANDE (1,N) ---> EXPEDIER ---> (1,1) TRANSPORTEUR

Un transporteur peut expédier 1 à N commandes. Une commande est expédiée par un seul transporteur.

CLIENT ---- (Recevoir) ---- FACTURE\_VENDEUR

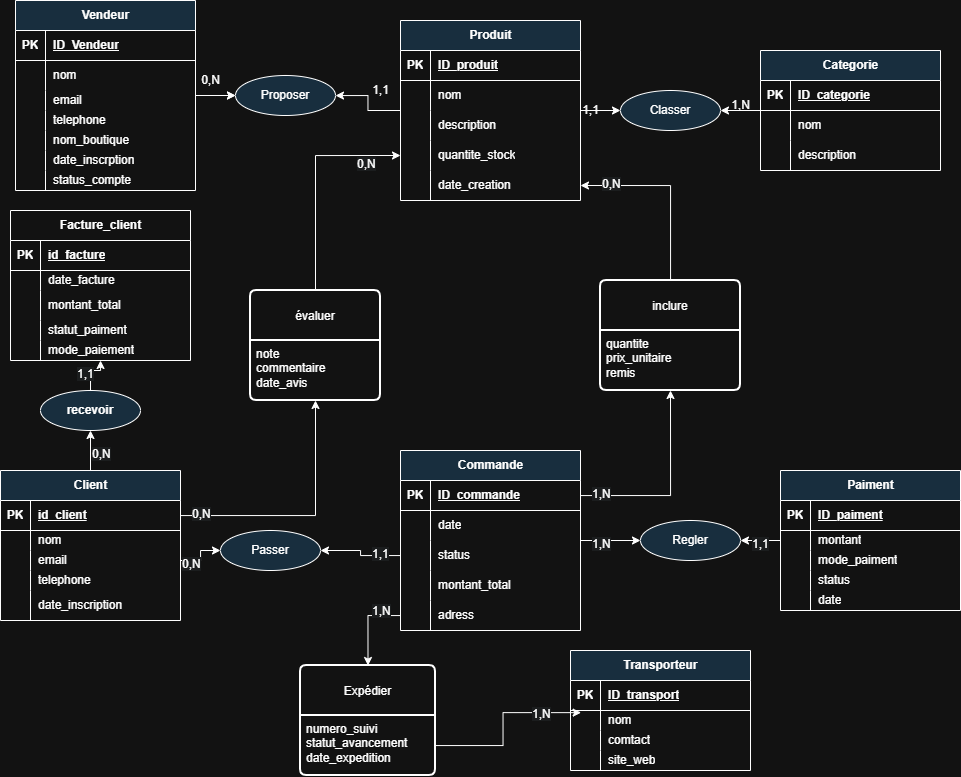
* Type : Association simple
* Cardinalités : CLIENT (1,1) ---> RECEVOIR ---> (1,N) FACTURE\_VENDEUR

Une facture concerne un seul CLIENT. Un CLIENT peut recevoir 1 à N factures.

**Remarque : la conception du cycle Client-Produit-Commande**

Nous avons gardé un cycle entre Client, Commande et Produit. Ce choix n'est pas une erreur, mais une décision réfléchie qui correspond aux besoins de la plateforme. En effet, le client a deux actions différentes dans le système : il peut **acheter** un produit (via l'association inclure dans une commande) et il peut aussi **donner son avis** sur ce produit (via l'association evaluer). Ces deux fonctions sont séparées et chacune a ses propres règles. Cette séparation nous permet de suivre indépendamment les achats et les avis, ce qui est important pour gérer la satisfaction des clients et améliorer la qualité de la plateforme.

**2.2.3. MCD**



**Légende du MCD:**

| **Symbole** | **Type** | **Description** |
| --- | --- | --- |
|  | Entité | Représente un objet réel du système (Client, Produit, Commande, etc.) |
|  | Association simple | Représente un lien entre deux entités (Proposer, Passer, Classer) |
|  | Association porteuse | Association qui porte des données propres (inclure, evaluer, Expédier) |

### 

### **2.3. Modèle Logique de Données (MLD)**

Le schéma relationnel pour notre base de données Oracle sera le suivant (PK : Clé Primaire,

FK : Clé Étrangère) :

* **Les associations simple :(0,N) | (1,N)**

1. **VENDEUR**

VENDEUR (id\_vendeur pk , nom, email, téléphone, nom\_boutique, date\_inscription)

🡪 chaque vendeur est un compte autonome, indépendant des autres entités. Il devient clé étrangère seulement dans PRODUIT .

1. **CLIENT**

CLIENT (id\_client PK, nom, prénom, email, téléphone)

🡪 Chaque client est indépendant.Il devient clé étrangère dans COMMANDE, FACTURE\_CLIENT .

1. **CATEGORIE**

CATEGORIE (id\_categorie PK, nom\_categorie, description)

🡪 permet de créer une hiérarchie de catégories (sous-catégories) , Il devient clé étrangère dans PRODUIT.

1. **TRANSPORTEUR**

TRANSPORTEUR(id\_transporteur PK, nom\_transporteur, telephone, email)

🡪 chaque transporteur est indépendant. Il devient clé étrangère uniquement dans association porteuse EXPEDITION.

* **Les associations simple avec clé étrangère : (0,1) | (1,1)**

1. **PRODUIT**

PRODUIT (id\_produit PK, nom\_produit, description, prix, quantite\_stock, #id\_vendeur FK, #id\_categorie FK)

🡪 chaque produit appartient à un vendeur et une catégorie.

1. **COMMANDE**

COMMANDE (id\_commande PK, date\_commande, statut, #id\_client FK)

🡪 relie chaque commande à un client et aux adresses concernées.

1. **PAIMENT**

PAIMENT (id\_paiement PK, montant, methode, statut, date\_paiement, #id\_commande FK)

🡪 1 commande → 0,N paiements possibles, permet de suivre l’état du paiement.

1. **FACTURE\_CLIENT**

FACTURE\_VENDEUR(id\_facture(PK),date\_facture,montant\_total, statut\_paiement, mode\_paiement)

* **Les associations pourteuses d’attributs : (0,N) | (1,N) & (0,N) | (1,N)**

1. **Evaluer(‘AVIS’)**

Evaluer(‘AVIS’) ((id\_client FK, id\_produit FK)PK, note, commentaire, date\_avis)

🡪 permet de stocker les évaluations des clients pour chaque produit.

1. **Contient (‘COMMANDE\_ITEM’)**

Contient(‘COMMANDE\_ITEM’)((id\_commande FK,id\_produit FK)PK,quantite, prix\_unitaire, remis)

🡪 détaille chaque produit acheté dans une commande multi-vendeur.

1. **Expedier**

Expedier ((id\_transporteur FK, id\_commande FK)PK, date\_expedition, numero\_suivi, statut\_expedition)

🡪 relie une commande à un transporteur.

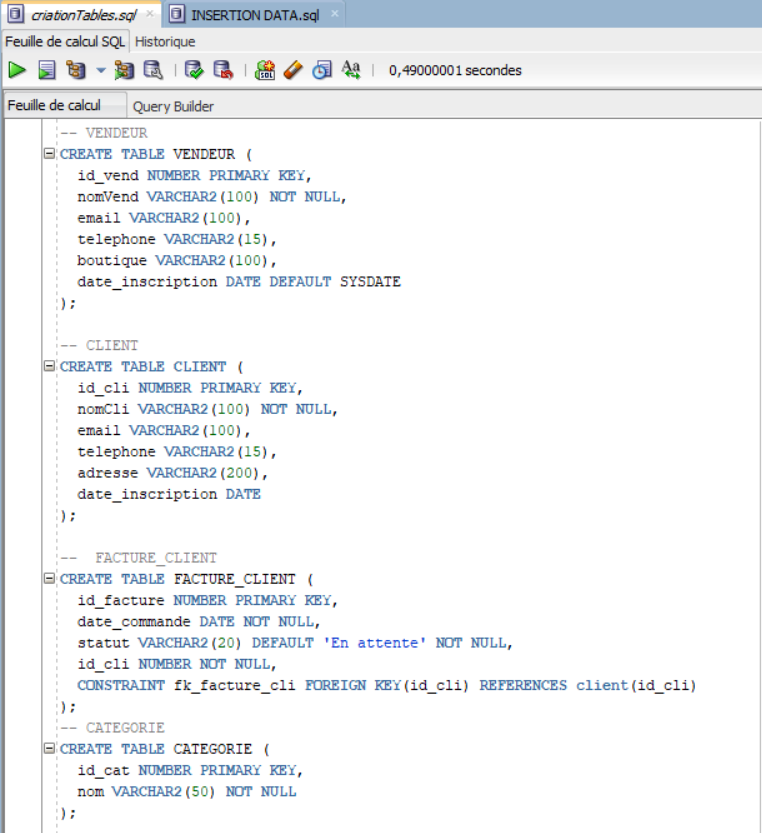
**Justifcation** :

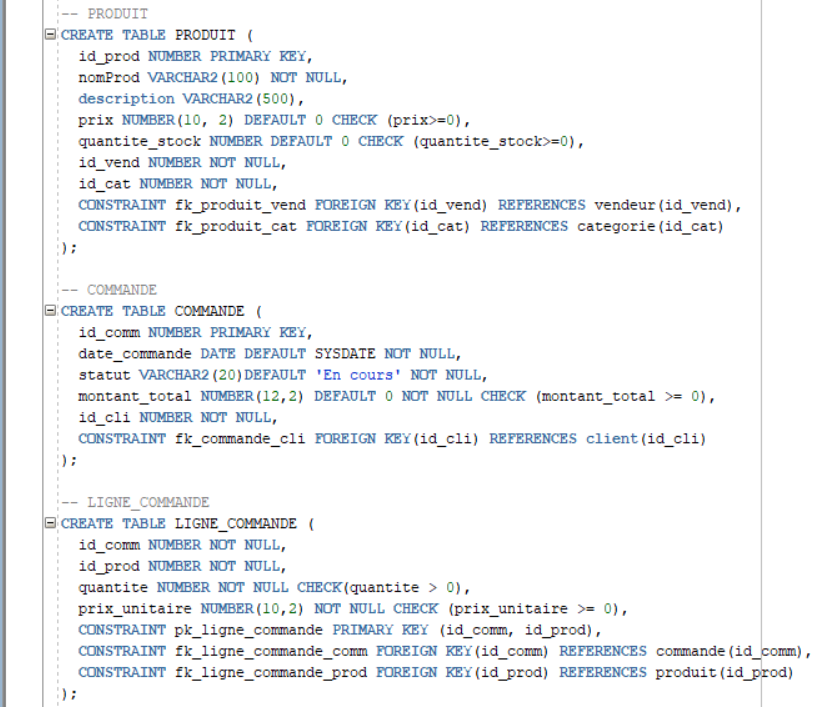
est une association porteuse car contient des attributs depends les deux entites.Comme ,en ‘Evaluer’ on’a attribut note, commentaire et la date\_avis depend de client et produit donc si on ne achete pas des produit alors avis pas important .

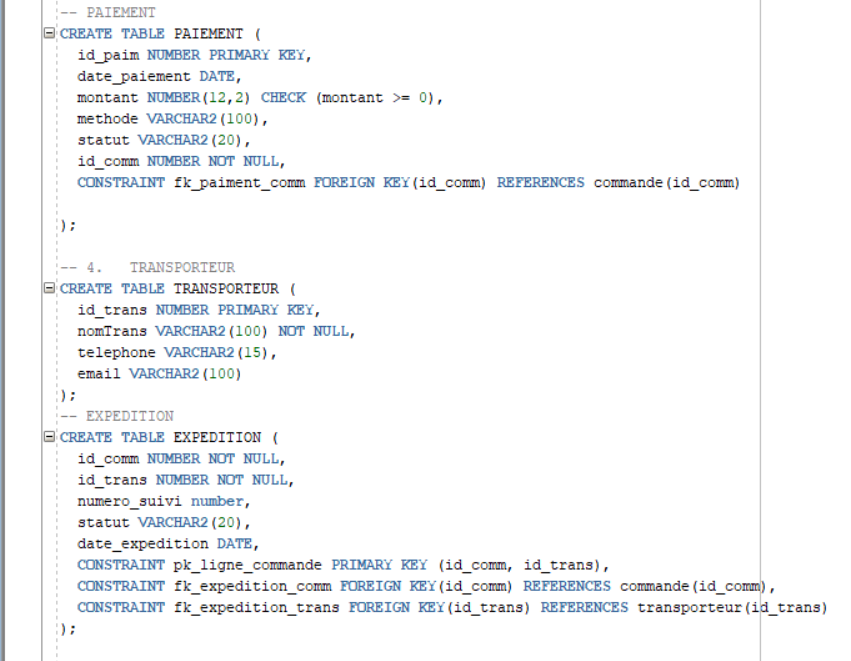
Et meme justification pour les autres .

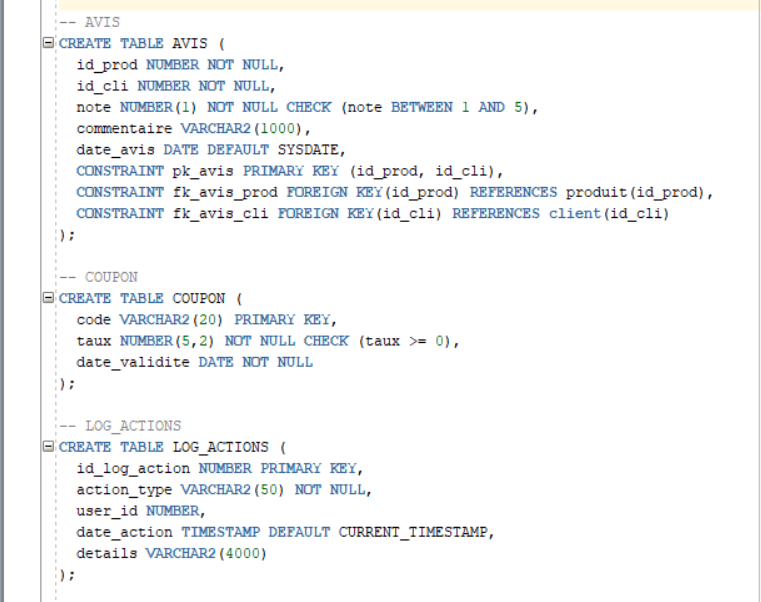
**Chapitre3 :Scripts SQL pour la création des tables et l’insertion des données**

**3.1.Création des tables**

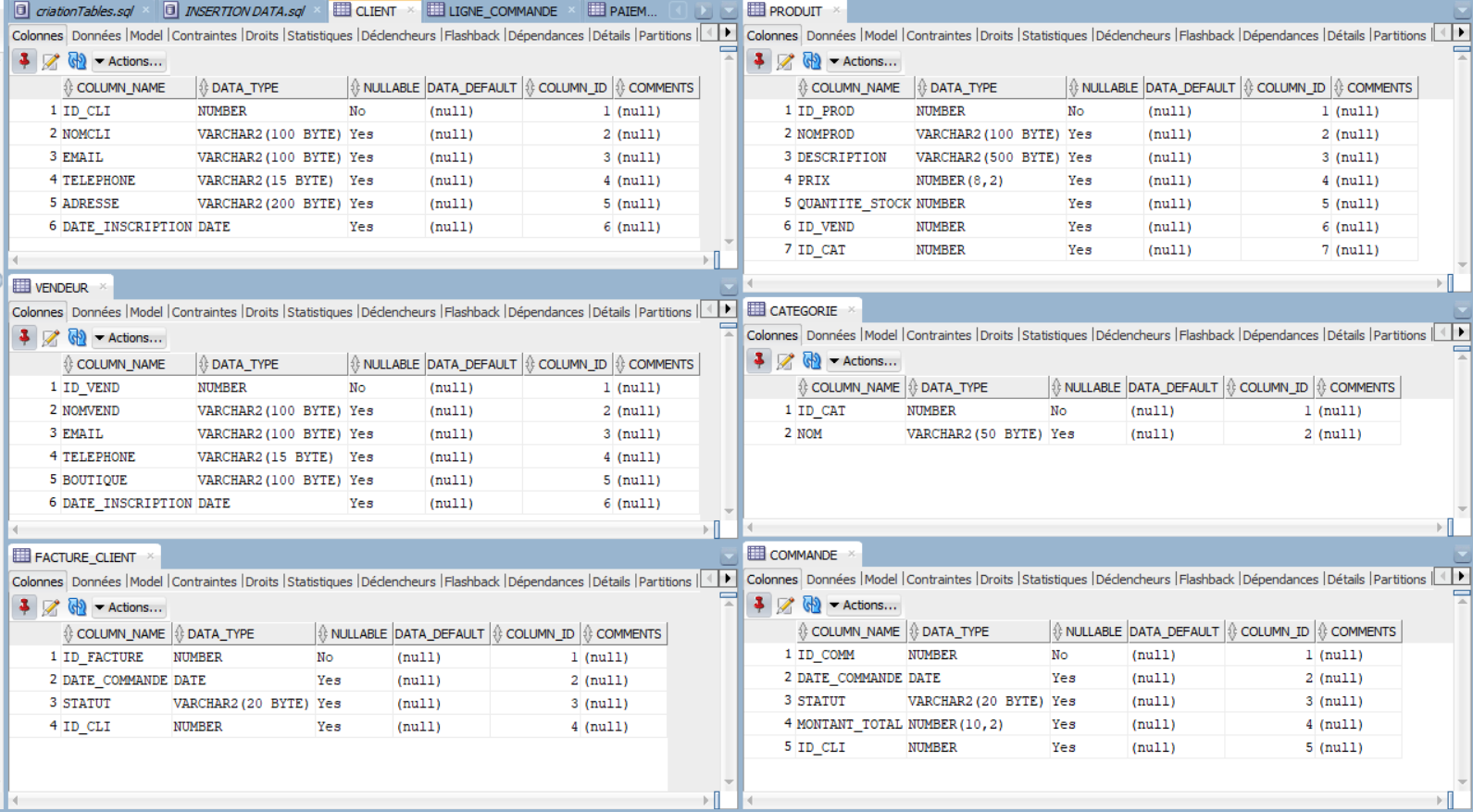


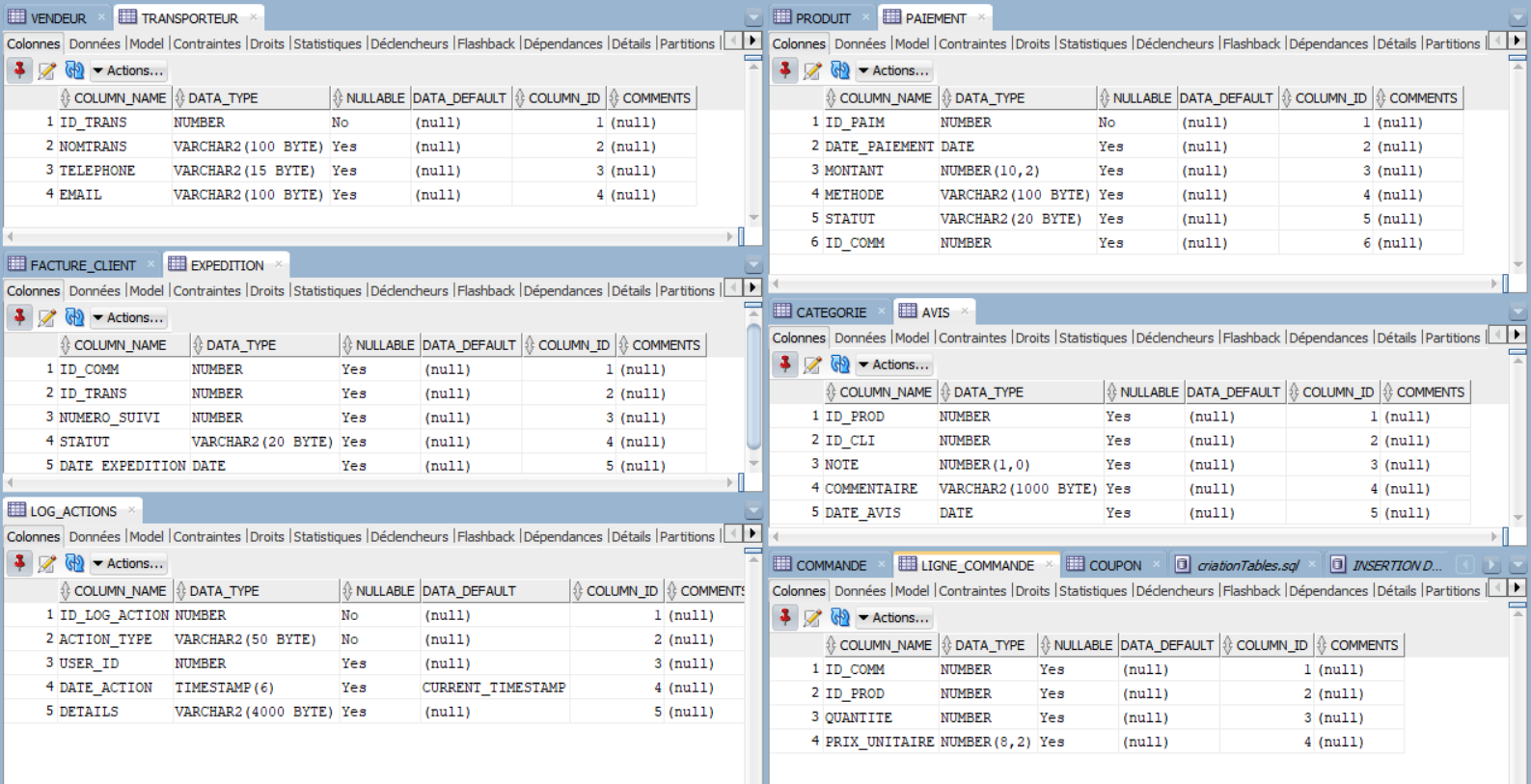


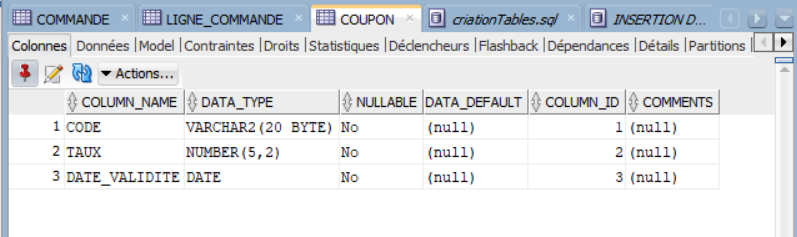




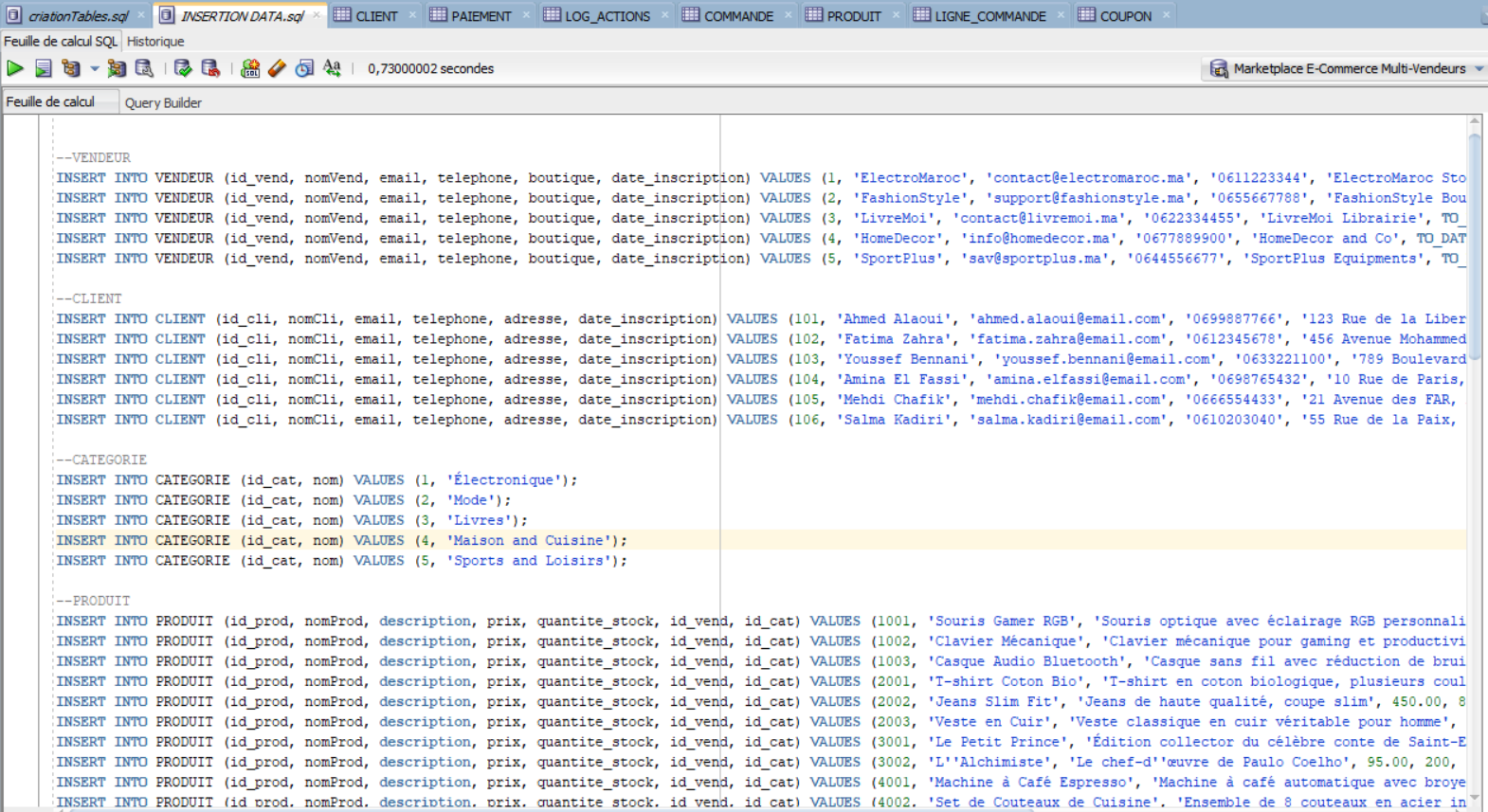
**3.1.Teste la création des tables**

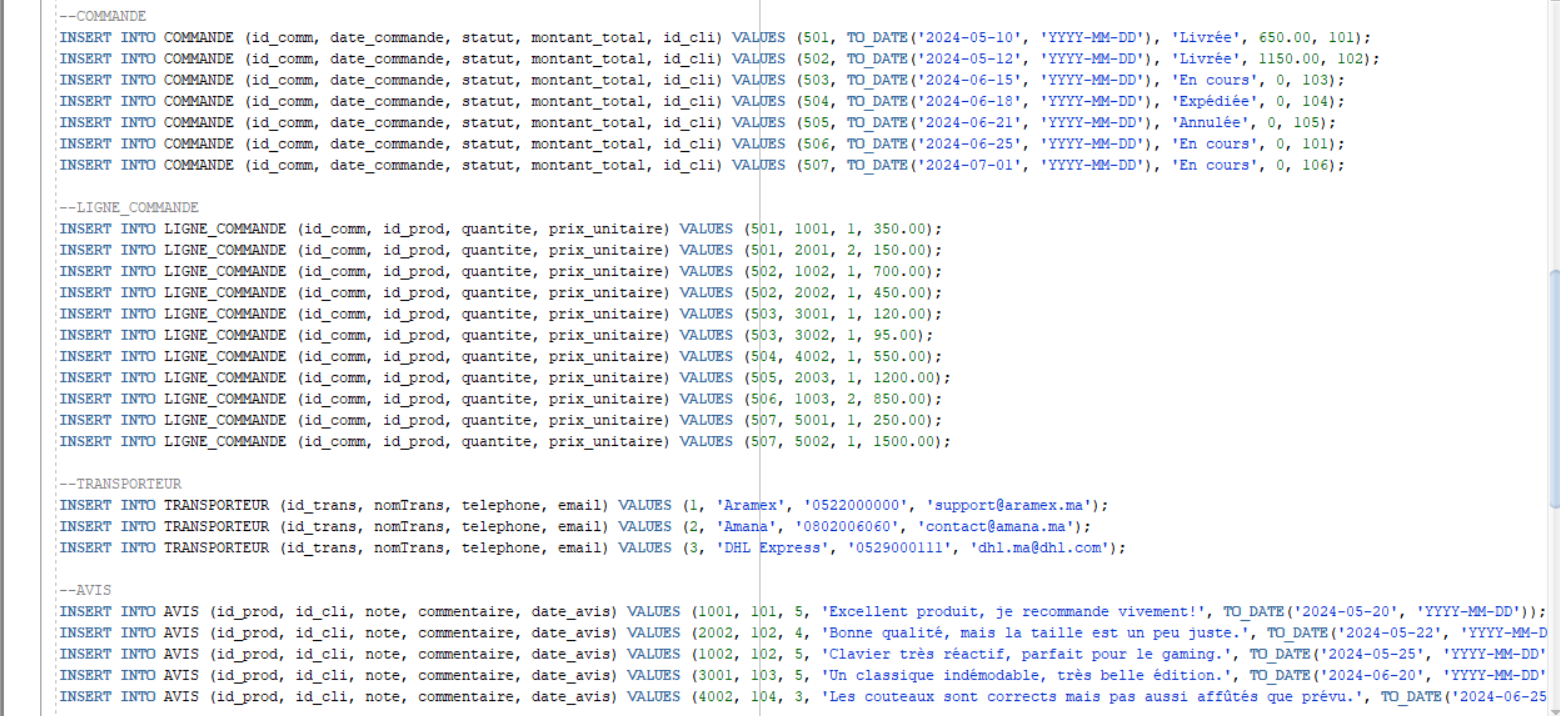






**3.4.Insertion des donnees :**

****

****

****

**3.1.Teste l’insertion des tables**

