1. Conception de la base de données

- *Définir les entités principales : **Client, Jeu, Achat, Éditeur, Panier, Paiement**, etc.*
- *Créer un **diagramme de classes UML** représentant les relations.*
- *Définir les **attributs** de chaque table.*
- *Prévoir les **contraintes** : clés primaires, étrangères, uniques.*
- *Prévoir les **index** utiles.*

***Outil utilisé** : `Draw.io` (pour le diagramme UML)*

---

2. Génération de données

- *Créer un script pour générer :*
  - ***1000 clients***
  - ***100 jeux***
  - *Un **historique d'achats réaliste** (plusieurs achats par client)*
- *Exporter les données en **CSV** ou insérer directement dans **PostgreSQL**.*

***Outils utilisés** :*

- `Python 3` *(avec les bibliothèques* `Faker`*,* `pandas`*,* `csv`*)*
  OU
- `PostgreSQL` *(si génération directe via SQL)*

---

3. Développement sur PostgreSQL

- *Créer le **schéma SQL** (DDL).*
- *Importer les données.*
- *Écrire **20 scripts PL/pgSQL** correspondant à des cas d'usage :*
  - *Total des achats d'un client*
  - *Top ventes*
  - *Ajout d'un jeu*
  - *etc.*
- *Inclure **1 script récursif** (ex : recommandations basées sur des achats similaires).*
- *Implémenter des **fonctions** (stockées, scalaires, d'agrégation).*
- *Implémenter des **triggers** (ex : mise à jour du stock après un achat).*
- *Tester les scripts.*

***Outil utilisé** : `PostgreSQL` (inclut `psql`, le shell SQL)*

---

4. Portage Oracle

- *Adapter le schéma PostgreSQL au **format Oracle**.*
- *Réécrire les fonctions, triggers et scripts en **PL/SQL**.*
- *Adapter la **syntaxe** aux particularités Oracle.*
- *Importer les données dans **Oracle**.*
- *Tester l'équivalence des 20 scripts dans Oracle.*

***Outils utilisés** :*

- `Oracle Database XE`
- `SQL Developer`

---

5. Déploiement Cassandra

- *Adapter le **modèle relationnel** en modèle orienté colonnes (Cassandra).*
- *Créer les **keyspaces et tables** en CQL.*
- *Réécrire les **requêtes principales** (non relationnelles).*
- *Déployer sur un **cluster de 10 nœuds Cassandra**.*
- *Importer une **version aplatie** des données.*
- *Tester les cas d'usage réalisables.*

***Outils utilisés** :*

- `Apache Cassandra`
- `cqlsh` *(inclus avec Cassandra)*

---

6. Benchmarking

- *Choisir **3 à 5 requêtes** à comparer.*
- *Utiliser un **benchmark de type TPC-C** pour PostgreSQL et Oracle.*
- *Chronométrer les **temps d'exécution** de chaque requête sur les mêmes données.*
- *Comparer les **performances PostgreSQL vs Oracle**.*

***Outils utilisés** :*

- `PostgreSQL`
- `Oracle Database XE`

---

7. Conversion Neo4j

- *Redéfinir le modèle sous forme de **graphe** :*
  - *Nœuds : **Clients, Jeux, Achats***
- *Convertir les données en **CSV ou Cypher** pour import.*

- *Créer les **relations** : `(Client)-[ACHÈTE]->(Jeu)`*
- *Importer les données dans **Neo4j**.*
- *Reproduire quelques **cas d'usage** avec des requêtes Cypher.*

***Outil utilisé** : `Neo4j Desktop`*

---

8. Documentation finale

- *Rédiger un document récapitulant :*
  - *Le **modèle de données** (schéma, classes)*
  - *Les **scripts** (exemples, rôles)*
  - *Les **adaptations Oracle et Cassandra***
  - *Les **résultats de benchmark***
  - *La **représentation graphe (Neo4j)***
- *Inclure des **captures d'écran**, **schémas**, et **résultats chiffrés**.*
- *Fournir tous les scripts : **SQL / PLpgSQL / PL/SQL / CQL / Cypher***

***Outil utilisé** : `LibreOffice Writer` (ou tout éditeur de texte compatible PDF)*