ANTISECHE

1. Objectif du projet

- Concevoir une base de données pour un service de location de véhicules électriques.
- Créer les tables, insérer des données, écrire des requêtes, ajouter des triggers.

2. Structure de la base (tables)

- **stations** : lieux où sont les véhicules
- **vehicules** : infos des voitures (marque, modèle, autonomie, état...)
- **clients** : utilisateurs du service
- **techniciens** : personnel de maintenance
- **reservations** : réservations des clients
- **paiements** : paiements liés aux réservations
- **maintenance** : interventions techniques

Clé étrangère principale : vehicules.id_station → stations.id_station

3. Ordre d'exécution des fichiers SQL

1. create_tables.sql → crée les tables
2. insert_test_data.sql → insère les données
3. queries.sql → exécute les requêtes
4. triggers_functions.sql → installe les triggers

4. Comment j'ai exécuté le projet

Via le terminal psql :

Code

\i 'chemin/create_tables.sql'

\i 'chemin/insert_test_data.sql'

\i 'chemin/queries.sql'

\i 'chemin/triggers_functions.sql'

Le CSV n'est **pas importé automatiquement**. Les données viennent de insert_test_data.sql.

5. Requêtes importantes (résultats attendus)

- **Afficher tous les véhicules** → 3 lignes (Renault, Tesla, Peugeot)
- **Compter les véhicules** → 3
- **Véhicules disponibles** → Renault Zoé
- **Véhicules en maintenance** → Peugeot e-208
- **Véhicules en service** → Tesla Model 3
- **Tri par autonomie** → Tesla > Renault > Peugeot
- **Nombre par marque** → 1 chacun
- **Autonomie moyenne par marque** → valeurs identiques aux autonomies
- **Répartition par localisation** → Paris / Montreuil / La Défense

6. Triggers

Exemple :

- Empêcher une réservation si le véhicule est en maintenance.
- Mettre à jour automatiquement l'état d'un véhicule après une maintenance.

Le trigger se déclenche **avant** ou **après** un INSERT / UPDATE.

7. Questions possibles du jury

« Pourquoi ces tables ? »

→ Pour représenter les entités du service : véhicules, stations, clients, réservations, paiements, maintenance.

« Pourquoi ces relations ? »

→ Pour assurer la cohérence : un véhicule appartient à une station, une réservation appartient à un client et un véhicule.

« Pourquoi un trigger ? »

→ Pour automatiser des règles métier (ex : empêcher une réservation impossible).

« Comment exécuter les scripts ? »

→ Avec \i dans psql ou via pgAdmin (Query Tool).

« D'où viennent les données ? »

→ Du fichier insert_test_data.sql, pas du CSV.

8. Conclusion rapide

- Base fonctionnelle
- Données cohérentes
- Requêtes correctes
- Triggers opérationnels
- Projet complet et propre