TP SQL –SELECT / INSERT / UPDATE / DELETE

CONTEXTE DE L'ENTREPRISE

Vous travaillez pour **Tech&Go**, une petite boutique informatique en ligne.
Le responsable vous envoie régulièrement de petites demandes.
Votre mission : comprendre ce qu'il souhaite et écrire **les requêtes SQL** correspondantes.

1. Ouvre **SQLiteStudio**.
2. Clique sur **Database** → **Add a Database**.
3. Choisis un fichier, par exemple :
   tp_sql_boutique.db
4. Une fois la base créée, exécute le script suivant pour générer les tables :

```
CREATE TABLE Clients (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nom TEXT NOT NULL,
    prenom TEXT NOT NULL,
    email TEXT UNIQUE NOT NULL
);
```

Que fait la commande précédente :

Cela créé une table Clients avec l'id en clé primaire, des noms, prénoms et emails sans texte vide

```
CREATE TABLE Produits (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nom TEXT NOT NULL,
    prix REAL NOT NULL
);
```

Que fait la commande précédente :

Cela créé une table Produits avec l'id en clé primaire, des noms et des prix sans texte vide

```
CREATE TABLE Commandes (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    id_client INTEGER NOT NULL,
    id_produit INTEGER NOT NULL,
    quantite INTEGER NOT NULL,
    date_commande TEXT NOT NULL,
    FOREIGN KEY(id_client) REFERENCES Clients(id),
    FOREIGN KEY(id_produit) REFERENCES Produits(id)
```

);

Que fait la commande précédente :

Cela créé une table Commandes avec l'id en clé primaire, des ids client et produit en clé étrangères, avec la quantité et les dates de commandes sans texte vide.

Vous venez de monter une mini-boutique.

INSERT INTO Clients (nom, prenom, email) VALUES ('Dupont', 'Jean', 'jean.dupont@example.com');

INSERT INTO Clients (nom, prenom, email) VALUES ('Martin', 'Sophie', 'sophie.martin@example.com');

INSERT INTO Clients (nom, prenom, email) VALUES ('Bernard', 'Lucas', 'lucas.bernard@example.com');

INSERT INTO Clients (nom, prenom, email) VALUES ('Petit', 'Emma', 'emma.petit@example.com');

INSERT INTO Clients (nom, prenom, email) VALUES ('Durand', 'Nina', 'nina.durand@example.com');

Que fait la commande précédente :
Cela insert des nom, prénom et emails de clients

INSERT INTO Produits (nom, prix) VALUES ('Clavier mécanique', 79.90);

INSERT INTO Produits (nom, prix) VALUES ('Souris gamer', 49.90);

INSERT INTO Produits (nom, prix) VALUES ('Écran 24 pouces', 159.00);

INSERT INTO Produits (nom, prix) VALUES ('Casque audio', 89.00);

INSERT INTO Produits (nom, prix) VALUES ('Tapis de souris XXL', 19.90);

Que fait la commande précédente :
Cela insert des valeurs nom produit prix etc dans la table produit

INSERT INTO Commandes (id_client, id_produit, quantite, date_commande)

VALUES (1, 1, 1, '2025-01-10');

INSERT INTO Commandes (id_client, id_produit, quantite, date_commande)

VALUES (2, 3, 2, '2025-01-10');

INSERT INTO Commandes (id_client, id_produit, quantite, date_commande)

VALUES (4, 5, 1, '2025-01-11');

Que fait la commande précédente :

Cela insert des données dont l id client produit, la quantité et la date de commande (table de jointure)

5. Le responsable veut vérifier que tous les clients sont bien enregistrés. Affichez la liste complète des clients.

```
SELECT *
FROM Clients
```

6. Il souhaite contacter les clients : il ne veut que leurs emails. Affichez uniquement les emails de tous les clients.

```
SELECT email
FROM Clients
```

7. Il veut voir le catalogue, mais trié du moins cher au plus cher. Affichez tous les produits triés par prix croissant.

```
SELECT *
FROM Produits
ORDER BY prix;
```

8. Il prépare une promo “haut de gamme”. Affichez les produits dont le prix est supérieur à 100 €.

```
SELECT *
FROM Produits
WHERE prix >=100;
```

9. Il veut connaître les commandes du client numéro 3. Listez toutes les commandes du client dont l’id = 3.

```
SELECT *
FROM Commandes, Clients
WHERE Clients.id=3;
```

10. Il veut analyser les ventes “enrichies”. Affichez les commandes avec : nom du client + nom du produit + quantité.

```
SELECT *
FROM Commandes, Clients, Produits
WHERE Commandes.id_client=Clients.id
AND Commandes.id_produit=Produits.id;
```

11. Il soupçonne que certains produits ne se vendent pas. Trouvez les produits qui n'ont jamais été commandés.

```
SELECT *
FROM Produits
WHERE id NOT IN (SELECT id_Produit FROM Commandes);
```

12. Il veut repérer les produits “entrée de gamme”. Trouvez les produits dont le prix est inférieur à 30 €.

```
SELECT *
FROM Produits
WHERE prix<=30;
```

13. Il prépare un mailing ciblé. Affichez les clients dont le nom commence par "D".

```
SELECT *
FROM Clients
WHERE nom LIKE "D%";
```

14. Il doit faire un rapport des commandes du 10 janvier 2025. Affichez toutes les commandes passées à la date du 2025-01-10.

```
SELECT *
FROM Commandes
WHERE date_commande="2025-01-10";
```

15. Un nouveau client vient de créer un compte.

    Ajoutez le client :

    - Nom : Leroy
    - Prénom : Jade
    - Email : jade.leroy@example.com

```
INSERT
INTO Clients(nom, prenom, email)
VALUES ('Leroy', 'Jade', 'jade.leroy@example.com');
```

16. Le magasin ajoute un produit “Webcam HD” à 59 €.

    Insérez ce produit.

```
INSERT
INTO Produits(nom, prix)
VALUES (' Webcam HD ' , 59);
```

17. Le client “Leroy Jade” vient d’effectuer une commande :

    - Produit : id = 2
    - Quantité = 1
    - Date du jour

    Insérez cette commande.

```
INSERT
INTO Commandes(id_client, id_produit,quantite,date_commande)
VALUES (6,2,1, '04/12/2025');
```

18. Deux nouveaux produits haut de gamme arrivent (prix > 250 €).
    Ajoutez-les dans la base.

```
INSERT
INTO Produits(nom, prix)
VALUES (' Laptop ' , 599);

INSERT
INTO Produits(nom, prix)
VALUES (' PC Gamer ' , 999);
```

19. Un client a changé d'adresse email.
    Modifiez l'email du client id = 1 avec : new.dupont@example.com

```
UPDATE Clients
SET email='new.dupont@example.com'
WHERE id=1;
```

20. Le prix du "Casque audio" baisse de 10 €.
    Réduisez son prix.

```
UPDATE Produits
SET prix=prix-10
WHERE nom='Casque audio';
```

21. L'entreprise augmente de 5 % le prix de tous les produits.
    Mettez à jour tous les prix.

```
UPDATE Produits
SET prix=prix*1.05;
```

22. Le responsable veut corriger une erreur :

    Le produit “Tapis de souris XXL” doit être renommé “Tapis XXL Gaming”.**

    Mettez à jour son nom.

```
UPDATE Produits
SET nom='Tapis XXL Gaming'
WHERE nom='Tapis de souris XXL';
```

23. Le client id = 4 s’est trompé : il avait commandé 1 unité mais en voulait 2.

    Modifiez la quantité de sa commande.

```
UPDATE Commandes
SET quantite=2
WHERE id_client=4;
```

24. Un client demande la suppression totale de son compte (RGPD).

    Supprimez le client dont l’id = 5.

```
DELETE
FROM Commandes
WHERE id_client=5;

DELETE
FROM Clients
WHERE id=5;
```

25. Le magasin retire un produit du catalogue : “Souris gamer”.

    Supprimez ce produit (s’il n’est pas utilisé en commande).

```
DELETE
FROM Produits

DELETE
FROM Commandes
WHERE id_produit=2;
```

26. Nettoyage de la base.

    Supprimez toutes les commandes de quantité 1.

```
DELETE
FROM Commandes
WHERE quantite=1;
```

27. Suppression des commandes trop anciennes.

    Supprimez toutes les commandes antérieures au 11 janvier 2025.

```
DELETE
FROM Commandes
WHERE date_commande<'2025-01-11';
```

28. Un test a créé une fausse commande id = 99.

    Supprimez-la.

```
DELETE
FROM Commandes
WHERE id=99 ;
```

29. Le responsable veut connaître la somme totale générée par les commandes.

    Calculez le montant total des ventes (prix × quantité).

```
SELECT SUM(prix*quantite)
FROM Commandes,Produits
WHERE Produits.id=Commandes.id_produit;
```

30. Il veut identifier les “meilleurs clients” :

    Trouvez le client qui a commandé le plus de produits (quantité totale).

```
SELECT c.id_client,
    c.nom_client,
    SUM(lc.quantite) AS quantite_totale
```

```
FROM Clients c
JOIN Commandes co ON co.id_client = c.id_client
JOIN LignesCommandes lc ON lc.id_commande = co.id_commande
GROUP BY c.id_client, c.nom_client
ORDER BY quantite_totale DESC
LIMIT 1;
```

31. Il veut connaître la commande la plus récente.

    Affichez la commande dont la date est la plus récente.

```
SELECT *
FROM Commandes
ORDER BY date_commande DESC
LIMIT 1;
```