Soit La base de données **Le_Traditionnel** définie par la représentation textuelle de ses tables suivante :

`Client (email, tel, nom)`

`Réservation (idReservation, espace, numTable, email#, dateR, heureP, nbPersonnes)`

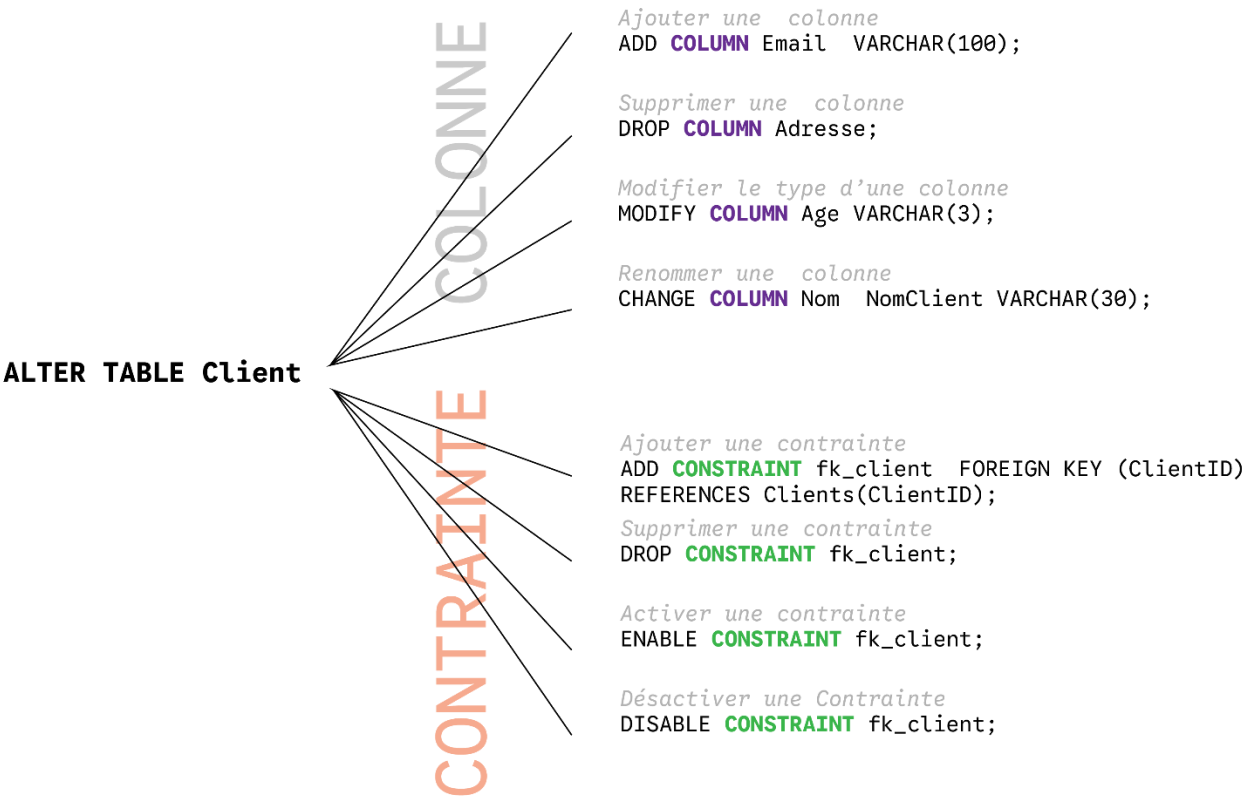
`Produit (idProduit, nomProduit, prix, qteStock)`

`Commande (idCommande, idReservation#)`

`LigneCommande (idCommande#, idProduit#, qte, caracteristiques)`

Tableau de description des colonnes

Champs	Type	Description	Contrainte
email	*Chaine de 30 caractères*	*Email d'un client*	*Clé primaire*
tel	*Numérique de 8 chiffres*	*Téléphone d'un client*	
nom	*Chaine de 20 caractères*	*Nom d'un client*	
idReservation	*entier*	*Identifiant d'une réservation*	*Clé primaire auto-incrémenté*
espace	*Chaine de 10 caractères*	*Nom d'un espace*	
numTable	*Entier de 2 chiffres*	*Numéro de table dans un espace*	
dateR	*Date*	*Date d'une réservation*	
heureP	*heure*	*Heure d'une réservation*	
nbPersonnes	*Entier de 2 chiffres*	*Nombre de personnes pour une réservation*	
idProduit	*entier*	*Identifiant d'un produit*	*Clé primaire auto-incrémenté*
nomProduit	*Chaine de 20 caractères*	*Nom d'un produit*	
prix	*décimale (10,3)*	*Prix d'un produit*	*Un réel strictement positif*
qteStock	*Entier de 3 chiffres*	*Quantité de stock d'un produit*	*Un entier positif*
idCommande	*entier*	*Identifiant d'une commande*	*Clé primaire auto-incrémenté*
qte	*entier de 3 chiffres*	*Quantité d'un produit commandée*	*Un entier strictement positif*
caracteristiques	*Chaine de 40 caractères*	*Caractéristiques d'un produit commandé*	


ALTER TABLE Client
COLONNE
CONTRAINTE
Ajouter une colonne
ADD COLUMN Email VARCHAR(100);
Supprimer une colonne
DROP COLUMN Adresse;
Modifier le type d'une colonne
MODIFY COLUMN Age VARCHAR(3);
Renommer une colonne
CHANGE COLUMN Nom NomClient VARCHAR(30);
Ajouter une contrainte
ADD CONSTRAINT fk_client FOREIGN KEY (ClientID)
REFERENCES Clients(ClientID);
Supprimer une contrainte
DROP CONSTRAINT fk_client;
Activer une contrainte
ENABLE CONSTRAINT fk_client;
Désactiver une Contrainte
DISABLE CONSTRAINT fk_client;

✅ 0. CREATION DE LA BASE DE DONNEES / LES TABLES

```sql
CREATE DATABASE BD_Le_Traditionnel;
-- Table n°1 : Client -- Contient les informations des clients.
CREATE TABLE Client (
email VARCHAR(30) PRIMARY KEY,
tel VARCHAR(8),
nom VARCHAR(20)
);
-- Table n°2 : Produit -- Contient le catalogue des produits.
CREATE TABLE Produit (
idProduit INT PRIMARY KEY AUTO_INCREMENT,
nomProduit VARCHAR(20),
prix DECIMAL(10, 3) CHECK (prix > 0),
typeProduit VARCHAR(20),
-- Contrainte : Le prix doit être strictement positif
qteStock INT CHECK (qteStock >= 0 AND qteStock <= 999)
-- Contrainte : La quantité en stock doit être un entier positif (0-999)
);
-- Table n°3 : Reservation -- Enregistre les réservations des clients. Dépend de la table
'Client'.
CREATE TABLE Reservation (
idReservation INT PRIMARY KEY AUTO_INCREMENT,
espace VARCHAR(10),
-- Contrainte : "Entier de 2 chiffres" pour la table (0-99)
numTable INT CHECK (numTable >= 0 AND numTable <= 99),
email VARCHAR(30),
dateR DATE,
heureP TIME,
-- Contrainte : "Entier de 2 chiffres" pour le nombre de personnes (1-99)
nbPersonnes INT CHECK (nbPersonnes > 0 AND nbPersonnes <= 99),
-- CLÉ ÉTRANGÈRE NOMMÉE vers le client qui a réservé
FOREIGN KEY FK_Reservation_Client (email) REFERENCES Client(email)
ON DELETE CASCADE ON UPDATE CASCADE
);
-- Table n°4 : Commande
```

```sql
-- Fait le lien entre une réservation et une commande passée. Dépend de la table 'Reservation'.
CREATE TABLE Commande (
idCommande INT PRIMARY KEY AUTO_INCREMENT,
idReservation INT,
-- CLÉ ÉTRANGÈRE NOMMÉE vers la réservation associée
FOREIGN KEY FK_Commande_Reservation (idReservation) REFERENCES Reservation(idReservation)
);
-- Table n°5 : LigneCommande
-- Table de jonction qui détaille les produits commandés. Dépend des tables 'Commande' et 'Produit'.
CREATE TABLE LigneCommande (
idCommande INT,
idProduit INT,
-- Contrainte : "Entier de 3 chiffres" et "strictement positif" (1-999)
qte INT CHECK (qte > 100 AND qte <= 999),
caracteristiques VARCHAR(40),
-- Clé primaire composite
PRIMARY KEY (idCommande, idProduit),
-- CLÉS ÉTRANGÈRES NOMMÉES
FOREIGN KEY FK_LigneCommande_Commande (idCommande) REFERENCES Commande(idCommande),
FOREIGN KEY FK_LigneCommande_Produit (idProduit) REFERENCES Produit(idProduit)
);
```

✅ 0. EXEMPLE 01 (CREATION TABLE SANS CONTRAINTE / AJOUT DES CONTRAINTES)

```sql
-- Création de la table : Client sans aucune contrainte
CREATE TABLE Client (
    email VARCHAR(30),
    tel VARCHAR(8),
    nom VARCHAR(20)
);

-- Ajout de la clé primaire à la table client
ALTER TABLE client
    ADD CONSTRAINT ajout_pk_client PRIMARY KEY (email);
```

```sql
-- Création de la table n°2 : Produit sans aucune contrainte
CREATE TABLE Produit (
    idProduit INT,
    nomProduit VARCHAR(20),
    prix DECIMAL(10, 3),
    qteStock INT
  );

-- Ajout des contraintes de la table n°2 : Produit
ALTER TABLE produit
    -- 1. Ajout de la clé primaire
    ADD PRIMARY KEY (idProduit),

    -- 2. Ajout de la contrainte de vérification (CHECK)
    ADD CONSTRAINT verif_prix
        CHECK (prix > 10);
```

✅ 0. AJOUT / SUPPRESSION / MODIFICATION DU NOM ET TYPE D'UN CHAMP

```sql
-- Ajout d'une column age à la table client
ALTER TABLE client
    ADD COLUMN age int ;

-- Modification du nom d'une column (age) -> date_naissance
ALTER TABLE client
    CHANGE age date_naissance INT;

-- Modification du type d'une column (date_naissance)
ALTER TABLE client
    MODIFY date_naissance DATE ;

-- Supression d'une column (date_naissance)
ALTER TABLE client
    Drop COLUMN date_naissance;

-- Supression d'une contrainte
```

```sql
ALTER TABLE produit
    DROP CONSTRAINT verif_prix;
```

✅ 0. SUPRESSION DES DONNEES

```sql
-- Supprimer en précisant les colonnes
DELETE FROM client WHERE tel="21620443";

-- Supression sans préciser les colonnes
DELETE FROM Clients;
```

✅ 0. MISE A JOUR DES DONNEES

```sql
-- Mise à jour des informations du client dont l'email est
 "ali.benali@example.com"
UPDATE client
SET
    nom = 'Ali Ben Ali Updated',    -- Nouveau nom
    tel = '99887766'                -- Nouveau téléphone
WHERE
    email = 'ali.benali@example.com';   -- Condition pour trouver la ligne
 à modifier

-- Mise à jour de toutes les lignes de la table client
UPDATE client
SET tel = '11111111';   -- Cette valeur sera appliquée à TOUTES les lignes


-- Modifier la valeur existante du champ tel pour un client précis
UPDATE client
SET tel = '55443322'         -- Nouvelle valeur
WHERE email = 'ali.benali@example.com';   -- Ligne à modifier
```

✅ 0. REQUÊTES AVEC INSERTION DES DONNEES

```
Insertion des enregistrements dans les tables


-- 1. Table Client - 7 enregistrements
INSERT INTO Client (email, tel, nom) VALUES
('amine@gmail.com', '55123456', 'Amine'),
('sara@gmail.com', '55223344', 'Sara'),
('mohamed@gmail.com', '55334455', 'Mohamed'),
('nadia@gmail.com', '55445566', 'Nadia'),
('youssef@gmail.com', '55556677', 'Youssef'),
('rim@gmail.com', '55667788', 'Rim'),
('khaled@gmail.com', '55778899', 'Khaled');
-- 2. Table Réservation – 7 enregistrement
INSERT INTO Reservation (idReservation, espace, numTable, email, dateR, heureP,
nbPersonnes) VALUES
(1, 'espace01', 5, 'amine@gmail.com', '2026-11-25', '19:00', 2),
(2, 'espace02', 12, 'sara@gmail.com', '2026-11-25', '20:00', 4),
(3, 'espace03', 1, 'mohamed@gmail.com', '2026-11-26', '21:00', 3),
(4, 'espace04', 3, 'nadia@gmail.com', '2026-11-26', '18:30', 2),
(5, 'espace05', 14, 'youssef@gmail.com', '2026-11-27', '19:45', 5),
(6, 'espace02', 2, 'rim@gmail.com', '2026-11-27', '20:30', 2),
(7, 'espace03', 7, 'khaled@gmail.com', '2026-11-28', '21:00', 3);


-- 3. Table Produit – 7 enregistrements
INSERT INTO Produit (idProduit, nomProduit, prix, qteStock, typeProduit) VALUES
(1, 'Eau Minérale', 1.500, 200, 'Boisson'),
(2, 'Boisson Gazeuse', 3.000, 150 , 'Boisson'),
(3, 'Sandwich', 8.000, 80, 'Nourriture'),
(4, 'Café Expresso', 2.000, 250, 'Boisson'),
(5, 'Jus Orange', 4.000, 120, 'Boisson'),
(6, 'Thé Menthe', 2.500, 180, 'Boisson'),
(7, 'Croissant', 2.000, 90 , 'Nourriture');


-- 4. Table Commande – 7 enregistrements
INSERT INTO Commande (idCommande, idReservation) VALUES
(1, 1),
(2, 2),
```

```
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7);

-- 5. Table LigneCommande – 7 enregistrements
INSERT INTO LigneCommande (idCommande, idProduit, qte, caracteristiques) VALUES
(1, 1, 220, 'fraîche'),
(2, 2, 150, 'bouteille 33cl'),
(3, 3, 289, 'sans mayonnaise'),
(4, 4, 140, 'serré'),
(5, 2, 210, 'canette'),
(6, 1, 107, '500ml'),
(7, 3, 166, 'sans sauce');
```

✅ 2. REQUÊTES AVEC FONCTIONS AGREGATS (BD LE TRADITIONNEL)

✅ 1. Nombre total de PRODUITS

```
SELECT COUNT(*) AS nb_produits
FROM Produit;
```

✅ 2. Prix moyen des produits

```
SELECT AVG(prix) AS prix_moyen
FROM Produit;
```

✅ 3. Prix maximum

```
SELECT MAX(prix) AS prix_max
FROM Produit;
```

✅ 4. Prix minimum

```
SELECT MIN(prix) AS prix_min
FROM Produit;
```

✅ 5. Quantité totale en stock

```
SELECT SUM(qteStock) AS quantite_totale_stock
FROM Produit;
```

✅ 6. Nombre de produits dont le prix > 3

(Toujours une bonne combinaison avec agrégat + condition)

```
SELECT COUNT(*) AS nb_produits_chers
```

```
FROM Produit
WHERE prix > 3;
```

✅ 7. Prix total de tous les produits en stock

(utile pour calculer la valeur du stock)

```
SELECT SUM(prix * qteStock) AS valeur_stock
FROM Produit;
```

Pour rendre ça intéressant, on va ajouter une colonne logique typeProduit (catégorie).

Exemples avec typeProduit :

- Eau Minérale → *Boisson*
- Boisson gazeuse → *Boisson*
- Sandwich → *Nourriture*
- Café expresso → *Boisson*
- Jus orange → *Boisson*
- Thé menthe → *Boisson*
- Croissant → *Nourriture*

🟦 1. Nombre de produits par type

```
SELECT typeProduit, COUNT(*) AS nb_produits
FROM Produit
GROUP BY typeProduit;
```

🟦 2. Prix moyen par type

```
SELECT typeProduit, AVG(prix) AS prix_moyen
FROM Produit
GROUP BY typeProduit;
```

🟦 3. Quantité totale en stock par type

```
SELECT typeProduit, SUM(qteStock) AS total_stock
FROM Produit
GROUP BY typeProduit;
```

🟦 4. Le prix maximum par type

```
SELECT typeProduit, MAX(prix) AS prix_max
FROM Produit
GROUP BY typeProduit;
```

🟦 5. Valeur totale du stock par type

(très utilisée en gestion)

```
SELECT typeProduit, SUM(prix * qteStock) AS valeur_stock
FROM Produit
GROUP BY typeProduit;
```

🟦 6. Afficher seulement les types ayant plus de 1 produit

(utilisation de HAVING)

```
SELECT typeProduit, COUNT(*) AS nb_produits
FROM Produit
GROUP BY typeProduit
HAVING COUNT(*) > 1;
```

🟦 7. Type de produit le plus cher (parmi chaque type)

```
SELECT typeProduit, nomProduit, prix
FROM Produit p
WHERE prix = (
    SELECT MAX(prix)
    FROM Produit
    WHERE typeProduit = p.typeProduit
);
```

✅ 3. REQUÊTES AVEC FONCTIONS DE DATE

✓ 3.1. Sélectionner les réservations d'aujourd'hui

```
SELECT * FROM Reservation
WHERE dateR = DATE(NOW());
```

✓ 3.2. Lister les réservations du mois courant

```
SELECT * FROM Reservation
```

```
WHERE MONTH(dateR) = MONTH(NOW())
  AND YEAR(dateR) = YEAR(NOW());
```

✓ 3.3. Réservations pour le mois de février

```
SELECT * FROM Reservation
WHERE MONTH(dateR) = 2;
```

✓ 3.4. Afficher l'année de chaque réservation

```
SELECT idReservation, dateR, YEAR(dateR) AS Annee
FROM Reservation;
```

✓ 3.5. Extraire le jour, mois et année de chaque réservation

```
SELECT idReservation, DAY(dateR) AS jour, MONTH(dateR) AS mois,
       YEAR(dateR)  AS annee
FROM Reservation;
```

✓ 3.6. Réservations dans les 7 prochains jours

```
SELECT * FROM Reservation
WHERE dateR BETWEEN DATE(NOW()) AND ADDDATE(NOW(), 7);
```

✓ 3.7. Nombre de jours restants avant chaque réservation

```
SELECT idReservation, dateR,
       DATEDIFF(dateR, NOW()) AS Jours_Restants
FROM Reservation;
```

✓ 3.8. Ajouter 7 jours à la date de réservation

```
SELECT idReservation, dateR,
       ADDDATE(dateR, INTERVAL 7 DAY) AS nouvelleDate
FROM Reservation;
```

✅ 4. FONCTIONS D'AGRÉGATION

✓ 4.1. Nombre total de réservations

```
SELECT COUNT(*) AS Total_Reservations
FROM Reservation;
```

✓ 4.2. Nombre total de personnes réservées

```
SELECT SUM(nbPersonnes) AS Total_Personnes
```

```
FROM Reservation;
```

✓ 4.3. Moyenne des personnes par réservation

```
SELECT AVG(nbPersonnes) AS Moyenne_Personnes
FROM Reservation;
```

✓ 4.4. Maximum de personnes dans une réservation

```
SELECT MAX(nbPersonnes) AS Max_Personnes
FROM Reservation;
```

✓ 4.5. Minimum de personnes dans une réservation

```
SELECT MIN(nbPersonnes) AS Min_Personnes
FROM Reservation;
```

✅ 5. GROUP BY + HAVING

✓ 5.1. Nombre de réservations par espace

```
SELECT espace, COUNT(*) AS NB_Reservations
FROM Reservation
GROUP BY espace;
```

✓ 5.2.1. Total de personnes par espace

```
SELECT espace, SUM(nbPersonnes) AS Total_Pers
FROM Reservation
GROUP BY espace;
```

✓ 5.2.2 Nombre total de personnes par jour

```
SELECT dateR, SUM(nbPersonnes) AS totalPersonnes
FROM Reservation
GROUP BY dateR;
```

✓ 5.3. Moyenne des personnes par espace

```
SELECT espace, AVG(nbPersonnes) AS Moyenne
FROM Reservation
GROUP BY espace;
```

✓ 5.4. Espaces qui ont plus de 10 personnes réservées au total (HAVING)

```
SELECT espace, SUM(nbPersonnes) AS Total
FROM Reservation
GROUP BY espace
HAVING SUM(nbPersonnes) > 10;
```

✓ 5.5. Jours avec plus de 2 réservations

```
SELECT dateR, COUNT(*) AS NB
FROM Reservation
GROUP BY dateR
HAVING COUNT(*) > 2;
```

✓ 5.6. Espaces avec une moyenne de plus de 5 personnes

```
SELECT espace, AVG(nbPersonnes) AS moyennePersonnes
FROM Reservation
GROUP BY espace
HAVING AVG(nbPersonnes) > 5;
```

🎁 6. Date + Group By

✓ Réservations par mois

```
SELECT MONTH(dateR) AS Mois, COUNT(*) AS NB
FROM Reservation
GROUP BY MONTH(dateR);
```

✓ Total personnes par mois

```
SELECT MONTH(dateR) AS Mois, SUM(nbPersonnes) AS TotalPersonnes
FROM Reservation
GROUP BY MONTH(dateR);
```