1. CRUD sur les tables

Les tables sont les entités relationnelles d'une base de données relationnelles.

Lister les tables

```
SHOW TABLES
```

Création d'une table

```
CREATE TABLE customers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    lastname VARCHAR(255),
    firstname VARCHAR(255),
    address VARCHAR(255),
    city VARCHAR(255)
);

CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    reference VARCHAR(255) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    customer_id INT,
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);
```

Supprimer une table

```
DROP TABLE customers;
```

Vider une table

```
TRUNCATE TABLE customers;
```

Renommer une table

```
RENAME TABLE customers TO clients
```

2. CRUD sur les colonnes

Une table est composée de colonnes. Chaque colonne a un type et peut avoir une contrainte.

Lister les colonnes

```
SHOW COLUMNS FROM clients
```

Ajouter une colonne

```
ALTER TABLE customers
ADD email VARCHAR(255);
```

Supprimer une colonne

```
ALTER TABLE customers
DROP COLUMN email;
```

Modifier le type d'une colonne

```
ALTER TABLE customers
MODIFY  COLUMN email VARCHAR(50)
```

Changer le nom d'une colonne

```
ALTER TABLE customers
RENAME COLUMN email TO email_address;
```

3. CRUD sur les lignes

Une table stocke ses données sous une liste de données dont la structure respecte la structure de la table (les valeurs doivent respecter le type et les contraintes des colonnes).

Lister les lignes

```
SELECT * FROM customers;
```

Insertion d'une ligne

```
INSERT INTO customers(lastname, firstname, address, city, email)
VALUES ('Dupont', 'Jean', '33 avenue du Maine', 'Paris', 'jean@dupont.com');

INSERT INTO orders(reference, customer_id) VALUES ('32443DFEWE', 1);
```

Modification d'une ligne

```
UPDATE customers SET firstname = 'Koki' WHERE id = 1
```

Suppression d'une ligne

```
DELETE FROM customers WHERE id = 1
```

4. Constraints

Les contraintes permettent d'assurer l'intégrité de la base de données en interdisant les données partielles ou incompatibles.

Ajouter des contraintes

```
ALTER TABLE orders
ADD CONSTRAINT FK__order_customer
FOREIGN KEY (customer_id) REFERENCES customers(id);
```

Supprimer des contraintes

```
ALTER TABLE orders
DROP CONSTRAINT  FK__order_customer;
```

La contrainte NOT NULL

```
CREATE TABLE orders
(
    reference VARCHAR(50) NOT NULL
)
```

La contrainte UNIQUE

```
CREATE TABLE orders
(
    reference VARCHAR(50) UNIQUE
)
```

La contrainte PRIMARY KEY

```
CREATE TABLE orders
(
    id INT PRIMARY KEY
)
```

La contrainte FOREIGN KEY

```
ALTER TABLE orders
ADD CONSTRAINT FK__order_customer
FOREIGN KEY (customer_id) REFERENCES customers(id);
```

La contrainte DEFAULT

```
CREATE TABLE orders
(
    note VARCHAR(255) DEFAULT ''
)
```

5. Aggregates

Les fonctions d'agrégation permettent d'avoir 1 seul indicateur par groupe de ligne au sein d'une même table.

```
SELECT MAX(price) FROM orders GROUP BY customer_id

SELECT MIN(price) FROM orders GROUP BY customer_id

SELECT AVG(price) FROM orders GROUP BY customer_id

SELECT SUM(price) FROM orders GROUP BY customer_id

SELECT COUNT(price) FROM orders GROUP BY customer_id
```

6. Jointure

Une jointure permet de lier plusieurs tables.

```
SELECT * FROM customers c LEFT OUTER JOIN orders o ON c.id = o.customer_id

SELECT * FROM customers c LEFT JOIN orders o ON c.id = o.customer_id

SELECT * FROM customers c RIGHT OUTER JOIN orders o ON c.id = o.customer_id

SELECT * FROM customers c RIGHT JOIN orders o ON c.id = o.customer_id

SELECT * FROM customers c INNER JOIN orders o ON c.id = o.customer_id

SELECT * FROM customers c JOIN orders o ON c.id = o.customer_id
```

7. Vues

Une vue permet de stocker une requête sous forme d'entité (tableau) séparée.

```
CREATE VIEW v_co AS
SELECT c.lastname, o.reference, o.price
FROM customers c JOIN orders o ON c.id = o.customer_id;
```

8. Procédures

Les procédures permettent de créer des fonctions qui pourront être appelées dynamiquement dans les requêtes.

```
DELIMITER //

CREATE PROCEDURE GetAllProducts()
BEGIN
    SELECT *  FROM products;
END //

DELIMITER ;
```

9. Transactions

Un transaction premet d'assurer les caractéristiques ACID d'un ensemble de requêtes.

```
START TRANSACTION;
INSERT INTO orders(reference, customer_id) VALUES('DSDF12324', 2);
INSERT INTO orders(reference, customer_id) VALUES('DSDF12324', 'ER');
COMMIT;
```

10. Indexes

Les indexes permettent d'améliorer la performance en recherche des requêtes qui consultent les colonnes indexées.

Créer un index

```
CREATE INDEX idx_lastname
ON customers (lastname);
```

Supprimer un index

```
DROP INDEX idx_lastname
```