Es gibt eine Vielzahl von SQL-Syntaxen, die in verschiedenen Datenbankverwaltungssystemen (DBMS) verwendet werden, um Datenbanken zu erstellen, zu ändern, Daten zu manipulieren und Abfragen durchzuführen. Hier ist eine umfassende Liste der grundlegenden SQL-Syntaxen:

## 1. Datenbankverwaltung

- **CREATE DATABASE**: Erstellt eine neue Datenbank.

```
CREATE DATABASE datenbankname;
```

- **DROP DATABASE**: Löscht eine bestehende Datenbank.

```
DROP DATABASE datenbankname;
```

- **USE**: Wählt eine Datenbank zur Nutzung aus.

```
USE datenbankname;
```

## 2. Tabellenverwaltung

- **CREATE TABLE**: Erstellt eine neue Tabelle.

```
CREATE TABLE tabellenname (
    spaltenname1 datentyp constraint,
    spaltenname2 datentyp constraint,
    ...
);
```

- **ALTER TABLE**: Ändert eine bestehende Tabelle.
  - Spalte hinzufügen:

```
ALTER TABLE tabellenname ADD spaltenname datentyp;
```

  - Spalte ändern:

```
ALTER TABLE tabellenname MODIFY spaltenname neuer_datentyp;
```

  - Spalte löschen:

```
ALTER TABLE tabellenname DROP COLUMN spaltenname;
```

- **DROP TABLE**: Löscht eine bestehende Tabelle.

```
DROP TABLE tabellenname;
```

- **TRUNCATE TABLE**: Entfernt alle Daten aus einer Tabelle, ohne die Tabellenstruktur zu löschen.

```
TRUNCATE TABLE tabellenname;
```

## 3. Datenmanipulation (DML)

- **INSERT INTO**: Fügt neue Daten in eine Tabelle ein.

```
INSERT INTO tabellenname (spalte1, spalte2, ...)
VALUES (wert1, wert2, ...);
```

- **UPDATE**: Ändert bestehende Daten in einer Tabelle.

```
UPDATE tabellenname
SET spalte1 = wert1, spalte2 = wert2, ...
WHERE bedingung;
```

- **DELETE FROM**: Löscht Daten aus einer Tabelle.

```
DELETE FROM tabellenname WHERE bedingung;
```

## 4. Datenabfrage (DQL)

- **SELECT**: Wählt Daten aus einer oder mehreren Tabellen aus.

- Grundlegende Abfrage:

```
SELECT spalte1, spalte2, ... FROM tabellenname;
```

- Mit Bedingung:

```
SELECT spalte1, spalte2, ... FROM tabellenname WHERE bedingung;
```

- Mit Gruppierung:

```
SELECT spalte1, COUNT(*) FROM tabellenname GROUP BY spalte1;
```

- Mit Sortierung:

```
SELECT spalte1, spalte2, ... FROM tabellenname ORDER BY spalte1
ASC|DESC;
```

- **JOIN**: Kombiniert Daten aus mehreren Tabellen.
  - INNER JOIN:

```
SELECT t1.spalte1, t2.spalte2
FROM tabelle1 t1
INNER JOIN tabelle2 t2 ON t1.spalte = t2.spalte;
```

  - LEFT JOIN:

```
SELECT t1.spalte1, t2.spalte2
FROM tabelle1 t1
LEFT JOIN tabelle2 t2 ON t1.spalte = t2.spalte;
```

  - RIGHT JOIN:

```
SELECT t1.spalte1, t2.spalte2
FROM tabelle1 t1
RIGHT JOIN tabelle2 t2 ON t1.spalte = t2.spalte;
```

  - FULL JOIN:

```
SELECT t1.spalte1, t2.spalte2
FROM tabelle1 t1
FULL OUTER JOIN tabelle2 t2 ON t1.spalte = t2.spalte;
```

## 5. Transaktionen

- **BEGIN TRANSACTION**: Startet eine neue Transaktion.

```
BEGIN TRANSACTION;
```

- **COMMIT**: Bestätigt alle Änderungen der aktuellen Transaktion.

```
COMMIT;
```

- **ROLLBACK**: Macht alle Änderungen der aktuellen Transaktion rückgängig.

```
ROLLBACK;
```

## 6. Rechteverwaltung

- **GRANT**: Gewährt Benutzerrechte.

```
GRANT privilegien ON objekt TO benutzer;
```

- **REVOKE**: Entzieht Benutzerrechte.

```
REVOKE privilegien ON objekt FROM benutzer;
```

## 7. Views

- **CREATE VIEW**: Erstellt eine Ansicht (View).

```
CREATE VIEW viewname AS
SELECT spalte1, spalte2, ...
```

```
FROM tabellenname
WHERE bedingung;
```

- **DROP VIEW**: Löscht eine bestehende Ansicht.

```
DROP VIEW viewname;
```

## 8. Indexe

- **CREATE INDEX**: Erstellt einen Index auf einer Tabelle.

```
CREATE INDEX indexname ON tabellenname (spalte1, spalte2, ...);
```

- **DROP INDEX**: Löscht einen bestehenden Index.

```
DROP INDEX indexname ON tabellenname;
```

## 9. Stored Procedures und Funktionen

- **CREATE PROCEDURE**: Erstellt eine gespeicherte Prozedur.

```
CREATE PROCEDURE procedurename
AS
BEGIN
    -- SQL-Anweisungen
END;
```

- **CREATE FUNCTION**: Erstellt eine benutzerdefinierte Funktion.

```
CREATE FUNCTION functionname (@parameternamen datentyp)
RETURNS datentyp
AS
BEGIN
    -- SQL-Anweisungen
    RETURN wert;
END;
```

- **EXEC**: Führt eine gespeicherte Prozedur aus.

```
EXEC procedurename @param1 = wert1, @param2 = wert2;
```

## 10. Trigger

- **CREATE TRIGGER**: Erstellt einen Trigger, der auf eine Tabelle angewendet wird.

```
CREATE TRIGGER triggername
ON tabellenname
AFTER INSERT|UPDATE|DELETE
AS
BEGIN
    -- SQL-Anweisungen
END;
```