If you're writing your own **pet database** with **bare minimum SQL support**, you should focus on implementing **essential SQL features** to make it usable for basic CRUD operations. Here's a practical **minimal SQL subset** to support:

1️⃣ Table Management

These commands allow users to create, delete, and inspect table structures.

✅ Minimum Support:

```
CREATE TABLE users (id INT PRIMARY KEY, name VARCHAR(255), age INT);
DROP TABLE users;
```

✅ Essential Concepts:

- `CREATE TABLE` → Define table schema.
- `DROP TABLE` → Remove table.

2️⃣ Basic Data Manipulation (CRUD)

These are the core commands for inserting, reading, updating, and deleting data.

✅ INSERT (Create)

```
INSERT INTO users (id, name, age) VALUES (1, 'John', 25);
```

✅ SELECT (Read)

```
SELECT * FROM users;  -- Get all rows
SELECT name FROM users WHERE age > 20;  -- Conditional retrieval
```

✅ UPDATE (Modify)

```
UPDATE users SET age = 26 WHERE id = 1;
```

✅ DELETE (Remove)

```
DELETE FROM users WHERE id = 1;
```

✅ Essential Concepts:

- `INSERT INTO ... VALUES (...)` → Add new records.
- `SELECT ... FROM ... WHERE ...` → Retrieve data.
- `UPDATE ... SET ... WHERE ...` → Modify records.
- `DELETE FROM ... WHERE ...` → Remove records.

3️⃣ Constraints (Minimal Support)

At a minimum, you should handle:

- `PRIMARY KEY` (Unique row identifier)
- `NOT NULL` (Ensure values are not empty)

✅ **Example:**

```
CREATE TABLE users (
    id INT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    age INT
);
```

4️⃣ Basic Indexing (For Efficiency)

Even a simple indexing system can improve performance.

✅ **Example:**

```
CREATE INDEX idx_users_name ON users(name);
```

- `CREATE INDEX` allows fast lookups.

5️⃣ Basic Transactions

To ensure data consistency, support simple transactions.

✅ **Example:**

```
BEGIN;
    INSERT INTO users (id, name, age) VALUES (2, 'Alice', 30);
    UPDATE users SET age = 31 WHERE id = 2;
COMMIT;
```

- `BEGIN ... COMMIT` ensures **atomicity**.

🔥 Absolute Minimum SQL Engine Scope

If you want a **working but minimal SQL implementation**, support:

Feature	SQL Example
Table Creation	`CREATE TABLE users (id INT PRIMARY KEY, name VARCHAR(255));`
Insert Data	`INSERT INTO users VALUES (1, 'Alice');`
Select Data	`SELECT * FROM users;`
Update Data	`UPDATE users SET name = 'Bob' WHERE id = 1;`
Delete Data	`DELETE FROM users WHERE id = 1;`
Basic Constraints	`PRIMARY KEY`, `NOT NULL`
Simple WHERE Filtering	`SELECT * FROM users WHERE name = 'Alice';`

Optional (For More Power)

If you want to go beyond the bare minimum:

- **Joins** (`SELECT * FROM users INNER JOIN orders ON users.id = orders.user_id;`)
- **Foreign Keys** (`FOREIGN KEY (user_id) REFERENCES users(id);`)
- **Aggregates** (`COUNT`, `SUM`, `AVG`)
- **Basic JSON Support** (`JSON` column type)

Final Thoughts

For a **pet database**, start simple:

1. **Support `CREATE TABLE`, `INSERT`, `SELECT`, `UPDATE`, and `DELETE`.**
2. **Ensure `PRIMARY KEY` and `WHERE` filtering works.**
3. **Optimize with indexing (`CREATE INDEX`).**
4. **Consider basic transactions (`BEGIN ... COMMIT`).**

Would you like a **minimal SQL parser** example? 🚀