*Here, I implemented the steps to design and implement a database for the NoteApp application:*

**1. Design database schema:**

```
CREATE TABLE users (
  user_id INT PRIMARY KEY,
  name VARCHAR(100)
);

CREATE TABLE notes (
  note_id INT PRIMARY KEY,
  user_id INT,
  title VARCHAR(100),
  content TEXT,
  created_date DATE,
  FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

This creates two tables - users and notes with the necessary columns and relationships. The foreign key establishes the linkage between notes and users.

**2. Write SQL queries:**

```
-- Get all notes for a user
SELECT * FROM notes WHERE user_id = 123;

-- Get a specific note
SELECT * FROM notes WHERE note_id = 456;

-- Get all users
SELECT * FROM users;

-- Create a new note
INSERT INTO notes (user_id, title, content, created_date)
VALUES (123, 'Shopping List', 'Milk, Bread, Apples', '2023-02-28');
```

**3. Implement CRUD operations:**

```
# Create
def create_user(name):
  sql = "INSERT INTO users (name) VALUES (%s)"
  cursor.execute(sql, (name,))

def create_note(user_id, title, content):
  sql = "INSERT INTO notes (user_id, title, content, created_date) VALUES (%s, %s, %s,
CURDATE())"
  cursor.execute(sql, (user_id, title, content))

# Read
def get_user(user_id):
  sql = "SELECT * FROM users WHERE user_id = %s"
  cursor.execute(sql, (user_id,))
  return cursor.fetchone()

def get_notes(user_id):
  sql = "SELECT * FROM notes WHERE user_id = %s"
  cursor.execute(sql, (user_id,))
  return cursor.fetchall()

# Update
def update_note(note_id, title, content):
  sql = "UPDATE notes SET title = %s, content = %s WHERE note_id = %s"
  cursor.execute(sql, (title, content, note_id))

# Delete
def delete_note(note_id):
  sql = "DELETE FROM notes WHERE note_id = %s"
  cursor.execute(sql, (note_id,))
```

This shows basic CRUD operations to create, read, update and delete data from the database using SQL and Python.