Day 1: Create ScoutConnect SQLite Database

Method 1: Using Python Script (Recommended)

Create a Python script to initialize your database:

```
# create_database.py
import sqlite3
import os
from datetime import datetime

def create_database():
    """Create empty ScoutConnect SQLite database with basic setup."""

    # Database file path
    db_path = "scoutconnect.db"

    # Remove existing database if it exists
    if os.path.exists(db_path):
        print(f"Removing existing database: {db_path}")
        os.remove(db_path)

    # Create new database connection
    print(f"Creating new database: {db_path}")
    conn = sqlite3.connect(db_path)
    cursor = conn.cursor()

    # Enable foreign key constraints
    cursor.execute("PRAGMA foreign_keys = ON;")

    # Create a simple metadata table to track database info
    cursor.execute("""
        CREATE TABLE database_info (
            id INTEGER PRIMARY KEY,
            version TEXT NOT NULL,
            created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
            description TEXT
        )
    """)
```

```
    # Insert initial metadata
    cursor.execute("""
        INSERT INTO database_info (version, description)
        VALUES (?, ?)
    """, ("0.1.0", "ScoutConnect initial database setup"))

    # Commit changes and close
    conn.commit()
    conn.close()

    print(f"✅ Database created successfully!")
    print(f"📍 Location: {os.path.abspath(db_path)}")
    print(f"📊 Size: {os.path.getsize(db_path)} bytes")

    return db_path

def verify_database(db_path):
    """Verify database was created correctly."""
    try:
        conn = sqlite3.connect(db_path)
        cursor = conn.cursor()

        # Check if we can query the metadata table
        cursor.execute("SELECT * FROM database_info")
        result = cursor.fetchone()

        if result:
            print(f"✅ Database verification successful!")
            print(f"   Version: {result[1]}")
            print(f"   Created: {result[2]}")
            print(f"   Description: {result[3]}")

        conn.close()
        return True

    except Exception as e:
        print(f"❌ Database verification failed: {e}")
        return False

if __name__ == "__main__":
    print("🚀 ScoutConnect Database Setup - Day 1")
    print("=" * 50)
```

```
    db_path = create_database()
    verify_database(db_path)

    print("\n📋 Next Steps:")
    print("1. Add database to .gitignore (already done if you followed setup)")
    print("2. Update .env file with database path")
    print("3. Start designing your database schema for Day 2")
```

Method 2: Using SQLite Command Line

If you have SQLite installed on your system:

```
# Create empty database using SQLite CLI
sqlite3 scoutconnect.db "SELECT 'Database created successfully';"

# Verify it was created
ls -la scoutconnect.db

# Optional: Add a simple table to verify functionality
sqlite3 scoutconnect.db << EOF
CREATE TABLE database_info (
    id INTEGER PRIMARY KEY,
    version TEXT NOT NULL DEFAULT '0.1.0',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO database_info (version) VALUES ('0.1.0');

.schema
.quit
EOF
```

Method 3: Using Python Interactive Shell

Open Python and run these commands:

```
import sqlite3
import os

# Create database
conn = sqlite3.connect('scoutconnect.db')
```

```
print(f"Database created: {os.path.abspath('scoutconnect.db')}")

# Create a simple test table
cursor = conn.cursor()
cursor.execute("""
    CREATE TABLE IF NOT EXISTS database_info (
        id INTEGER PRIMARY KEY,
        version TEXT DEFAULT '0.1.0',
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    )
""")

cursor.execute("INSERT INTO database_info (version) VALUES ('0.1.0')")
conn.commit()

# Verify
cursor.execute("SELECT * FROM database_info")
print("Database contents:", cursor.fetchall())

conn.close()
print("✅ Database setup complete!")
```

Method 4: Integration with Your FastAPI Project

Update your existing project structure to include database initialization:

```
# app/database.py (update existing file)
import sqlite3
import os
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
from app.config import settings

def create_empty_database():
    """Create empty SQLite database if it doesn't exist."""
    db_path = "scoutconnect.db"

    if not os.path.exists(db_path):
        print(f"Creating database: {db_path}")
        conn = sqlite3.connect(db_path)

        # Enable foreign key constraints
```

```
        conn.execute("PRAGMA foreign_keys = ON;")

        # Create metadata table
        conn.execute("""
            CREATE TABLE database_info (
                id INTEGER PRIMARY KEY,
                version TEXT NOT NULL DEFAULT '0.1.0',
                created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
                last_updated TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            )
        """)

        # Insert initial record
        conn.execute("""
            INSERT INTO database_info (version) VALUES ('0.1.0')
        """)

        conn.commit()
        conn.close()
        print(f"✅ Database created successfully!")
    else:
        print(f"Database already exists: {db_path}")

# Create SQLite engine
engine = create_engine(
    settings.database_url,
    connect_args={"check_same_thread": False}
)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()

def get_db():
    """Dependency to get database session."""
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# Initialize database on import
create_empty_database()
```

Method 5: Using a Dedicated Setup Script

Create a comprehensive setup script for your project:

```
# scripts/setup_database.py
import sqlite3
import os
import sys
from datetime import datetime
from pathlib import Path

# Add the app directory to Python path
sys.path.append(str(Path(__file__).parent.parent / "app"))

def setup_database():
    """Complete database setup for ScoutConnect."""

    print("🏗️ ScoutConnect Database Setup")
    print("=" * 40)

    # Configuration
    db_name = "scoutconnect.db"
    db_path = Path(db_name)

    # Check if database already exists
    if db_path.exists():
        response = input(f"Database {db_name} already exists. Overwrite? (y/N): ")
        if response.lower() != 'y':
            print("Setup cancelled.")
            return
        db_path.unlink()
        print(f"Removed existing database: {db_name}")

    # Create database
    print(f"Creating database: {db_name}")
    conn = sqlite3.connect(str(db_path))
    cursor = conn.cursor()

    # Enable foreign key constraints and other pragmas
    cursor.execute("PRAGMA foreign_keys = ON;")
    cursor.execute("PRAGMA journal_mode = WAL;")  # Better for concurrent access
    cursor.execute("PRAGMA synchronous = NORMAL;")  # Balance between safety and speed

    # Create database metadata table
```

```
cursor.execute("""
    CREATE TABLE database_metadata (
        id INTEGER PRIMARY KEY,
        version TEXT NOT NULL,
        created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
        last_migration TIMESTAMP,
        description TEXT,
        environment TEXT DEFAULT 'development'
    )
""")

# Insert initial metadata
cursor.execute("""
    INSERT INTO database_metadata
    (version, description, environment)
    VALUES (?, ?, ?)
""", ("0.1.0", "ScoutConnect initial database setup", "development"))

# Create a simple health check table
cursor.execute("""
    CREATE TABLE health_check (
        id INTEGER PRIMARY KEY,
        status TEXT DEFAULT 'healthy',
        last_check TIMESTAMP DEFAULT CURRENT_TIMESTAMP
    )
""")

cursor.execute("INSERT INTO health_check (status) VALUES ('initialized')")

# Commit and close
conn.commit()
conn.close()

# Verify setup
db_size = db_path.stat().st_size
print(f"✅ Database created successfully!")
print(f"   📍 Location: {db_path.absolute()}")
print(f"   📊 Size: {db_size} bytes")
print(f"   🕐 Created: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")

# Test connection
try:
    conn = sqlite3.connect(str(db_path))
    cursor = conn.cursor()
```

```
        cursor.execute("SELECT version, created_at FROM database_metadata LIMIT 1")
        result = cursor.fetchone()
        conn.close()

        print(f"  ✅ Connection test successful!")
        print(f"  🏷️  Version: {result[0]}")
        print(f"  📅 Created: {result[1]}")

    except Exception as e:
        print(f"  ❌ Connection test failed: {e}")
        return False

    print("\n📋 Next Steps for Day 2:")
    print("1. Design your database schema (Users, Players, Evaluations, etc.)")
    print("2. Create SQLAlchemy models")
    print("3. Set up database migrations")
    print("4. Add the database to your FastAPI app")

    return True

if __name__ == "__main__":
    setup_database()
```

Quick Start (Recommended Approach)

For Day 1, I recommend using **Method 1** (Python script). Here's what to do:

Create the script:

```
# In your project root directory
touch create_database.py
```

1.
2. **Copy the Python script** from Method 1 above into `create_database.py`

Run the script:

```
python create_database.py
```

3.

Verify the database was created:

```
ls -la scoutconnect.db
```

4.

Update your .env file (if needed):

```
echo "DATABASE_URL=sqlite:///./scoutconnect.db" >> .env
```

5.

Database File Checklist

- [ ] Database file created (`scoutconnect.db`)
- [ ] Database is in `.gitignore` (to avoid committing to Git)
- [ ] Database path is configured in `.env`
- [ ] Basic metadata table exists for versioning
- [ ] Foreign key constraints are enabled
- [ ] Database is accessible and functional

Troubleshooting

Common Issues:

1. **Permission denied**: Make sure you have write permissions in the directory
2. **Module not found**: Ensure you're running Python from the correct directory
3. **Database locked**: Close any existing connections to the database

Verification Commands:

```
# Check if database exists and get info
sqlite3 scoutconnect.db ".schema"

# Check database size
ls -lh scoutconnect.db

# Test basic query
sqlite3 scoutconnect.db "SELECT * FROM database_info;"
```

Your empty database is now ready for Day 2 when you'll start designing the actual schema! 🎉