**Project: Movie & Actor Connection Manager**

**Repository Setup (Initial Steps for your local Git):**

**Initial Git Repository Setup (Pre-Project Steps)**

These steps are performed *before* you even run npm init.

1. **Create a New Remote Repository (e.g., on GitHub):**
   * Go to GitHub (or GitLab, Bitbucket, etc.).
   * Click "New repository".
   * Give it a name (e.g., neo4j-movie-app).
   * Choose public or private.
   * **Do NOT** initialize with a README.md or .gitignore or license. Leave it completely empty.
   * Click "Create repository".
2. **Clone the Empty Remote Repository Locally:**
   * After creating the repository on GitHub, it will show you instructions to clone it. Copy the HTTPS or SSH URL.
   * Open your terminal/command prompt.
   * Navigate to the directory where you want to store your project (e.g., C:\Users\USER\Documents\course taken\neo4j\simple projects\).
   * Run the clone command:

Bash

git clone <your\_repository\_url>

(e.g., git clone https://github.com/your-username/neo4j-movie-app.git)

* + Navigate into the newly cloned (empty) project directory:

Bash

cd neo4j-movie-app

* + **Note:** Because you cloned an *empty* repository, you **do not** need to run git init or git branch -M main here. The clone command handles that.

Bash

cd neo4j-movie-app

git init

git branch -M main # Renames default branch to main

**Commit 1: Initial Project Setup**

* **Message:** feat: Initialize Node.js project and install Neo4j driver
* **Description:**
  + Set up package.json with basic project info.
  + Install neo4j-driver dependency.
  + Configure package.json for ES modules ("type": "module").
* **Files Affected:** package.json

**Commit 2: Basic Neo4j Connection & Test**

* **Message:** feat: Add basic Neo4j connection logic to app.js
* **Description:**
  + Create app.js with Neo4j driver import.
  + Configure connection URI, user, and password.
  + Implement testConnection function to verify database connectivity with a simple RETURN query.
  + Set up session handling (opening and closing).
* **Files Affected:** app.js

**Commit 3: Database Information & Cleanup**

* **Message:** refactor: Improve connection test to display database name
* **Description:**
  + Explicitly specify database name (neo4j) when creating a session.
  + Modify testConnection to use db.name() in the return query, confirming the connected database.
  + Add .gitignore to exclude node\_modules/.
* **Files Affected:** app.js, .gitignore

**Commit 4: Implement Movie Node Creation**

* **Message:** feat: Add function to MERGE Movie nodes
* **Description:**
  + Create a new async function addMovie that uses MERGE to create or update Movie nodes.
  + Include ON CREATE SET for properties like released and tagline.
  + Demonstrate usage of addMovie in testConnection or a temporary main function.
* **Files Affected:** app.js (or a new services/movieService.js if you prefer to structure it now)

**Commit 5: Implement Person Node Creation**

* **Message:** feat: Add function to MERGE Person nodes
* **Description:**
  + Create a new async function addPerson that uses MERGE to create or update Person nodes.
  + Include ON CREATE SET for properties like born.
  + Demonstrate usage of addPerson.
* **Files Affected:** app.js (or the services module)

**Commit 6: Add Unique Constraints for Nodes**

* **Message:** feat: Add unique constraints for Movie title and Person name
* **Description:**
  + Implement an async function createConstraints to set up UNIQUE constraints on Movie(title) and Person(name).
  + Call this function once when the application starts or on initial setup. This is crucial for MERGE performance and data integrity.
* **Files Affected:** app.js (or a db/setup.js file)

**Commit 7: Implement Relationship Creation (ACTED\_IN)**

* **Message:** feat: Add function to MERGE ACTED\_IN relationships
* **Description:**
  + Create an async function addActorToMovie to MATCH existing Person and Movie nodes.
  + MERGE the ACTED\_IN relationship between them.
  + Use ON CREATE SET for relationship properties like roles.
  + Demonstrate usage.
* **Files Affected:** app.js (or the services module)

**Commit 8: Implement Basic Query Functions (Reads)**

* **Message:** feat: Add functions to query movie and actor data
* **Description:**
  + Implement getMovieByTitle(title): MATCH (m:Movie {title: $title}) RETURN m.
  + Implement getPersonByName(name): MATCH (p:Person {name: $name}) RETURN p.
  + Demonstrate usage.
* **Files Affected:** app.js (or the services module)

**Commit 9: Implement Relationship Query Functions**

* **Message:** feat: Add functions to query relationships (movies by actor, actors in movie)
* **Description:**
  + Implement getMoviesByActor(actorName): MATCH (p:Person)-[:ACTED\_IN]->(m:Movie) ....
  + Implement getActorsInMovie(movieTitle): MATCH (p:Person)-[:ACTED\_IN]->(m:Movie) ....
  + Demonstrate usage.
* **Files Affected:** app.js (or the services module)

**Commit 10: Refactor into Modules & Clean Execution**

* **Message:** refactor: Organize database operations into dedicated service module
* **Description:**
  + Move all addMovie, addPerson, addActorToMovie, and query functions into a new file (e.g., src/movieService.js or db/index.js).
  + Update app.js to import and use these functions.
  + Create a clear main execution block in app.js to demonstrate all functionalities.
  + Handle driver.close() gracefully when all operations are complete.
* **Files Affected:** app.js, src/movieService.js (or similar), potentially new src/ or db/ directories.

**Commit 11: Documentation**

* **Message:** docs: Add basic README.md
* **Description:**
  + Provide a brief overview of the project.
  + Include setup instructions (npm install, Neo4j credentials).
  + Explain how to run the application.
* **Files Affected:** README.md

This structured approach will make your Git history clear and your project development manageable. Let's start with the first few commits (Steps 1-3) to get your project initialized and the enhanced connection test working.