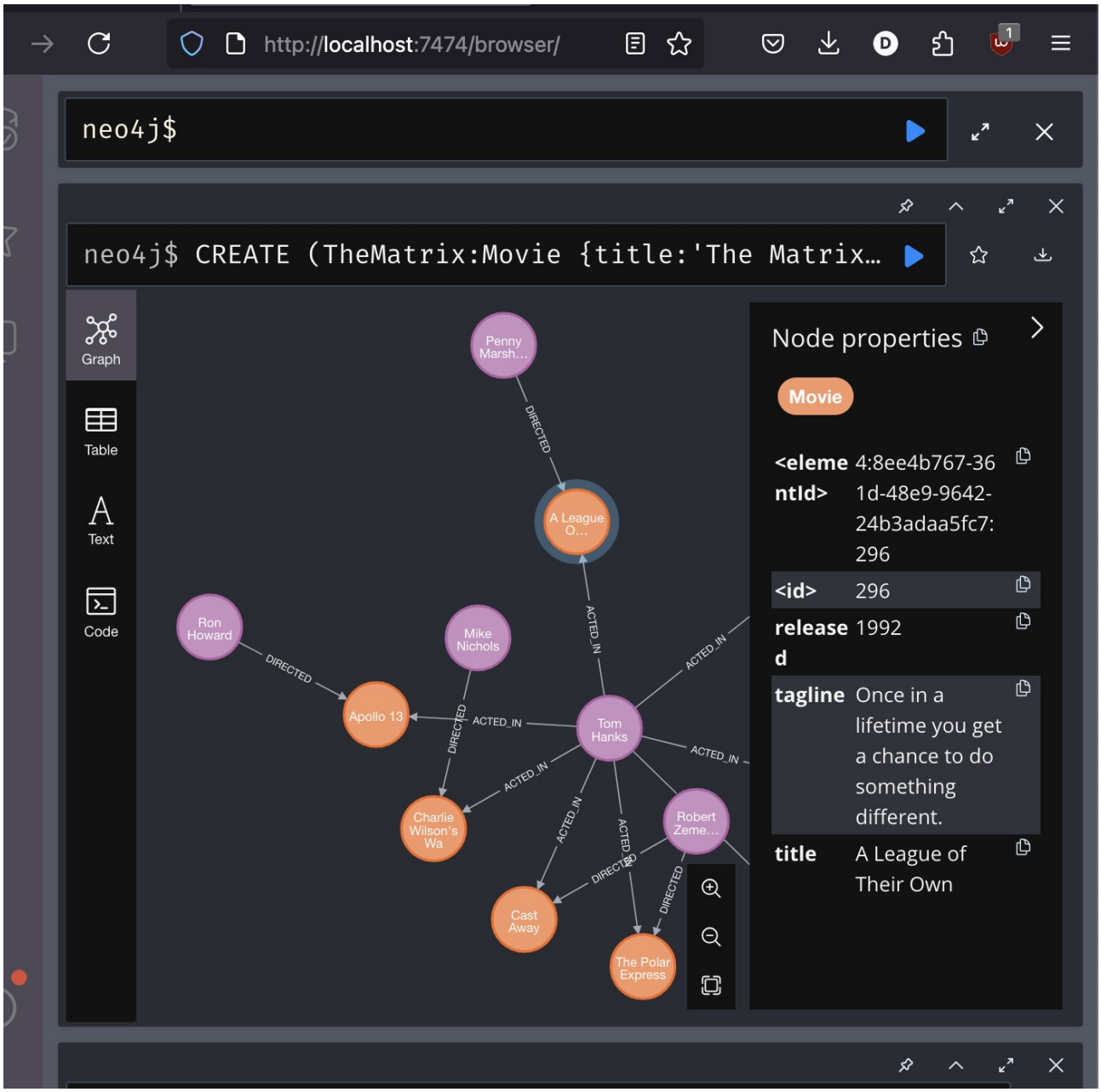
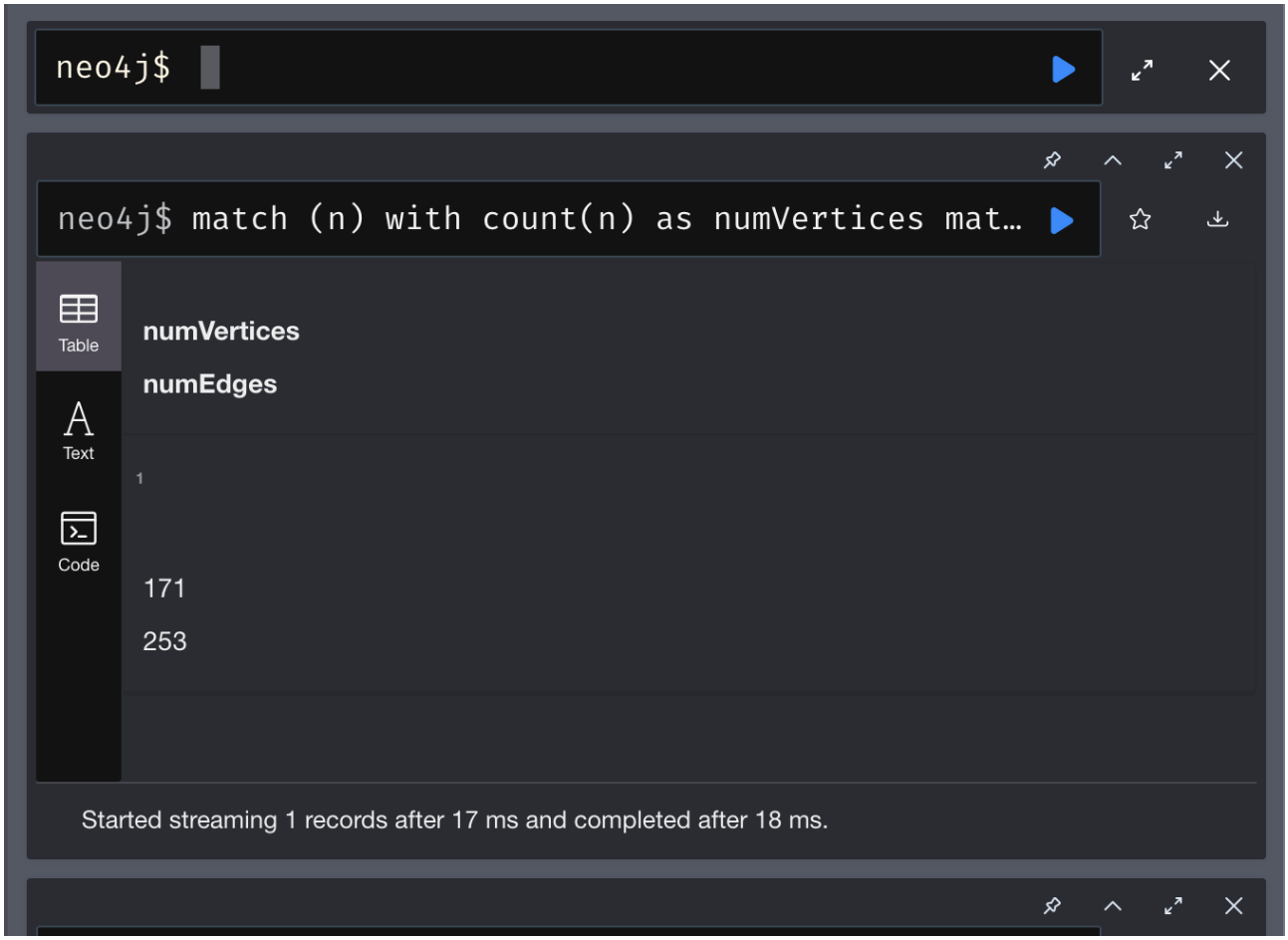
Daniel Soden Assignment 6 neo4j

1.

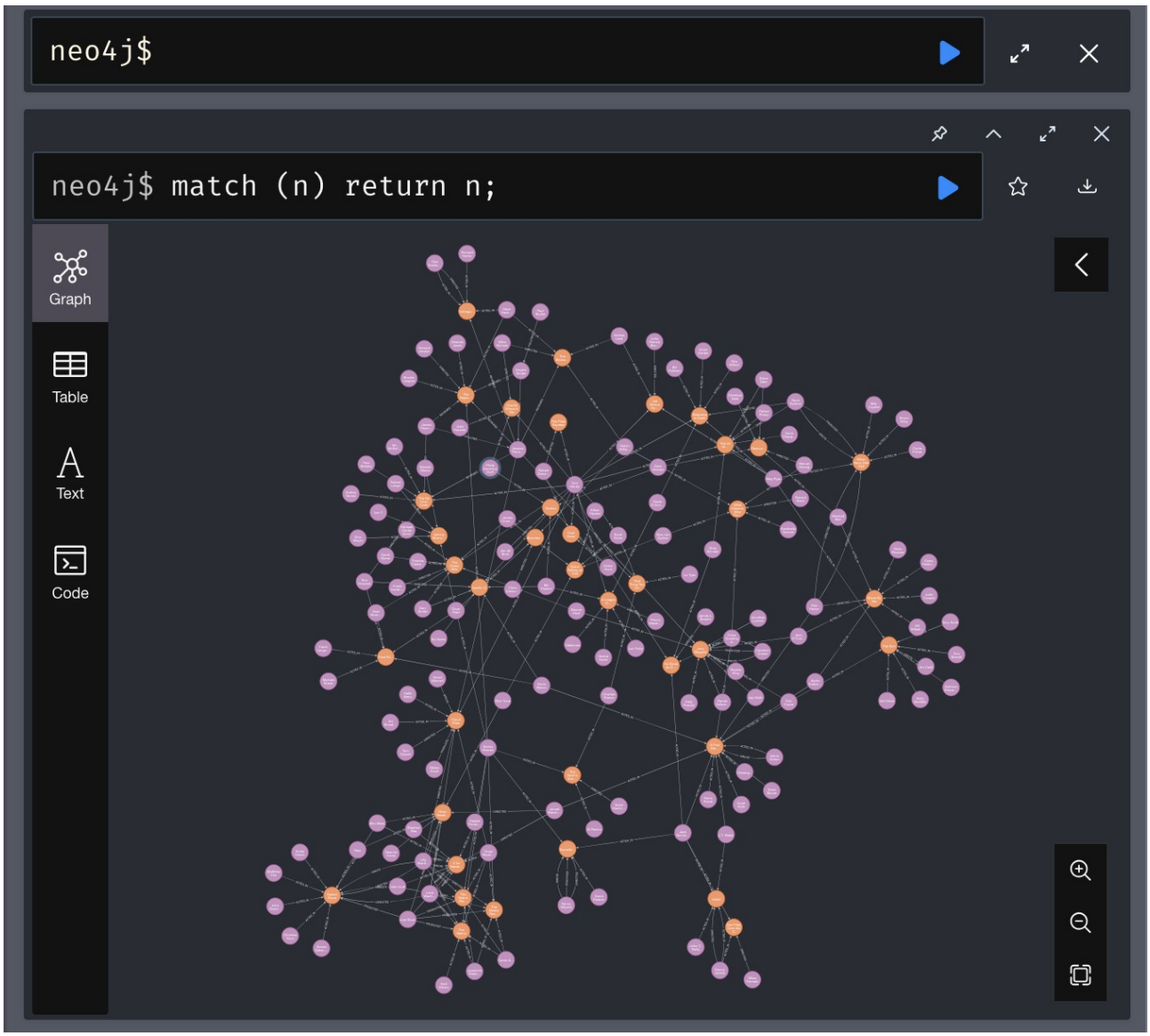
I started by execcuting play movie graph and then in the second page, I was prompted to execute this CREATE graph command:



Then after running this command I was able to see the number of vertices, ie the nodes and the number of edges being the amount of relationships in the graph.

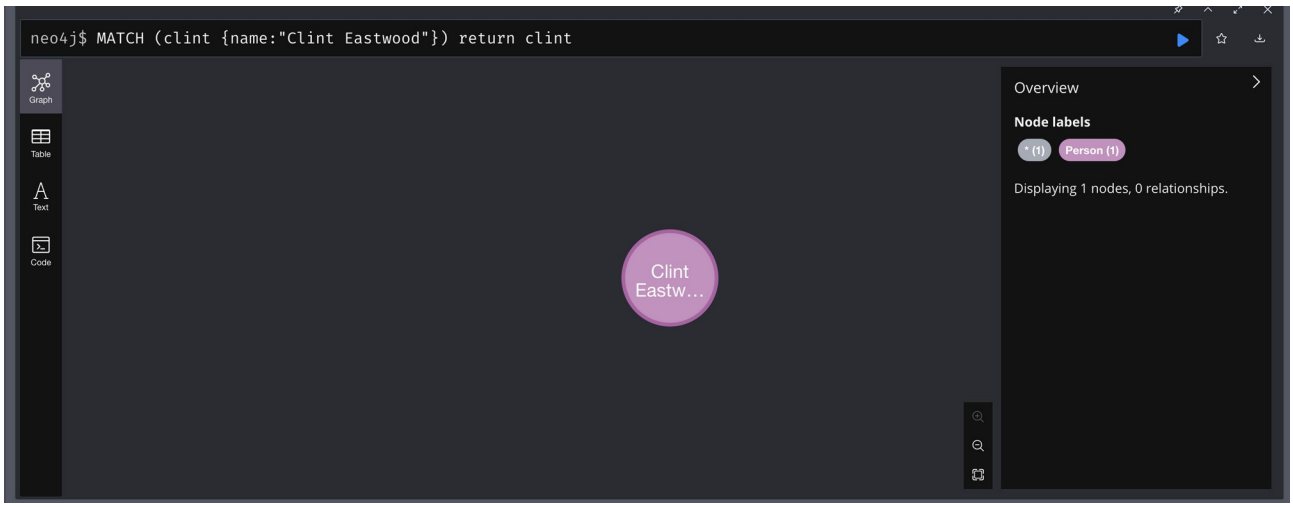


Then to view all the nodes and their relationships I ran this command:

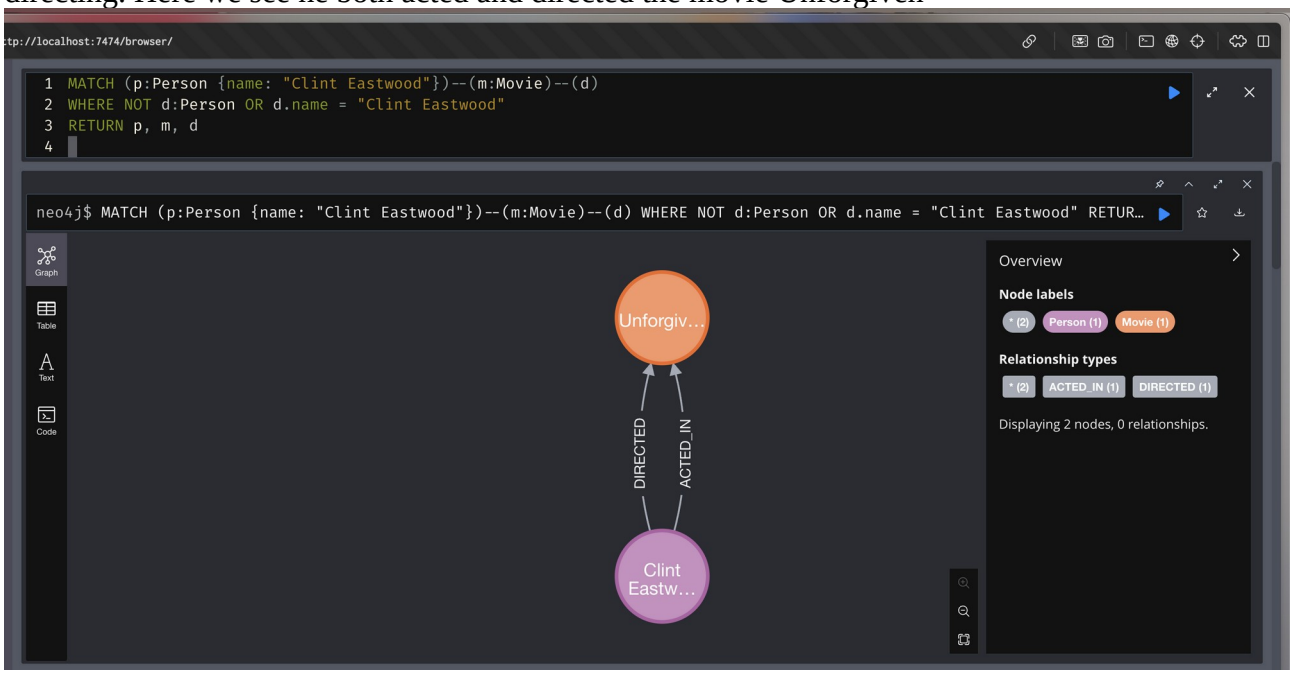


Now that our database is all setup and verified, we can start on the core db queries.

Before inserting the data relating to the movie Bridges of Mason County, I began checking if these nodes actually already existed for the director or actors. As seen above Clint Eastwood is already a person here so it stopped me from creating a dummy node.

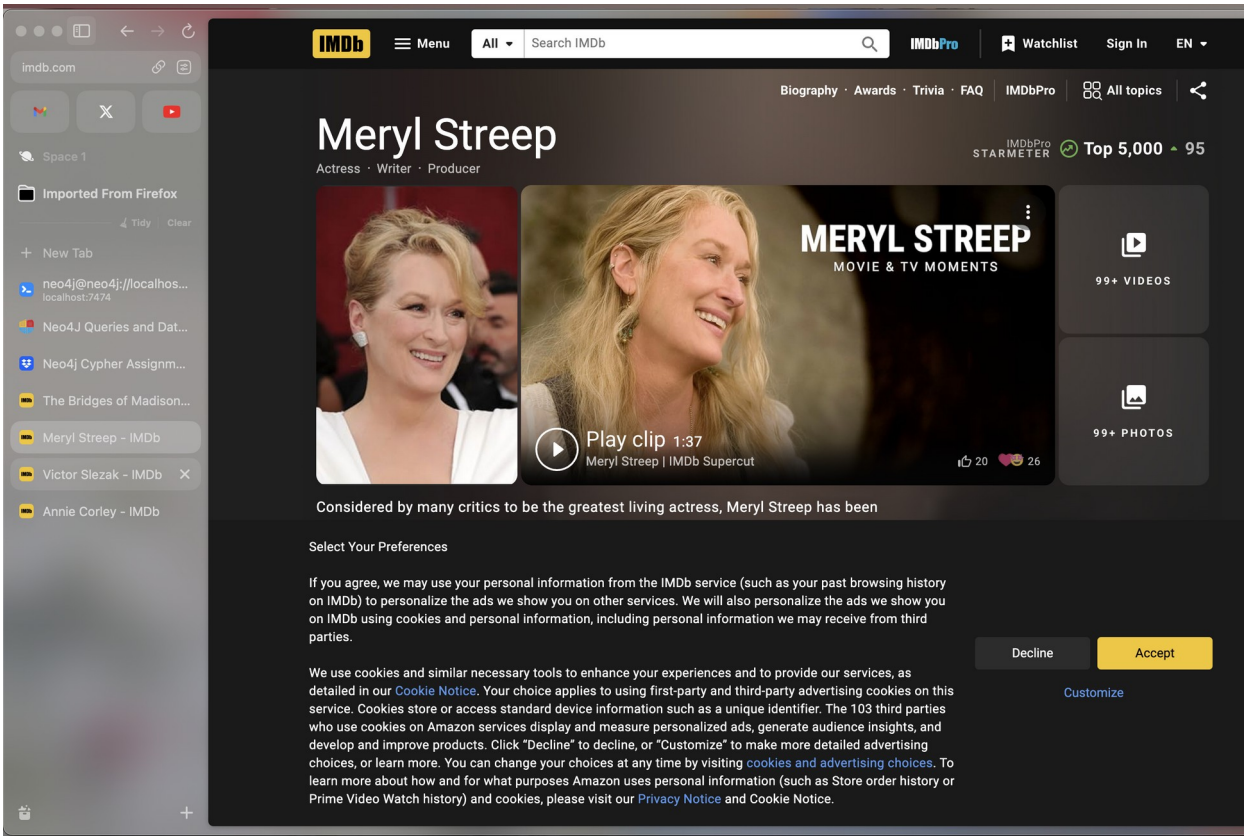


Then I checked what kind of connections Clint Eastwood already had in terms of acting and directing. Here we see he both acted and directed the movie Unforgiven

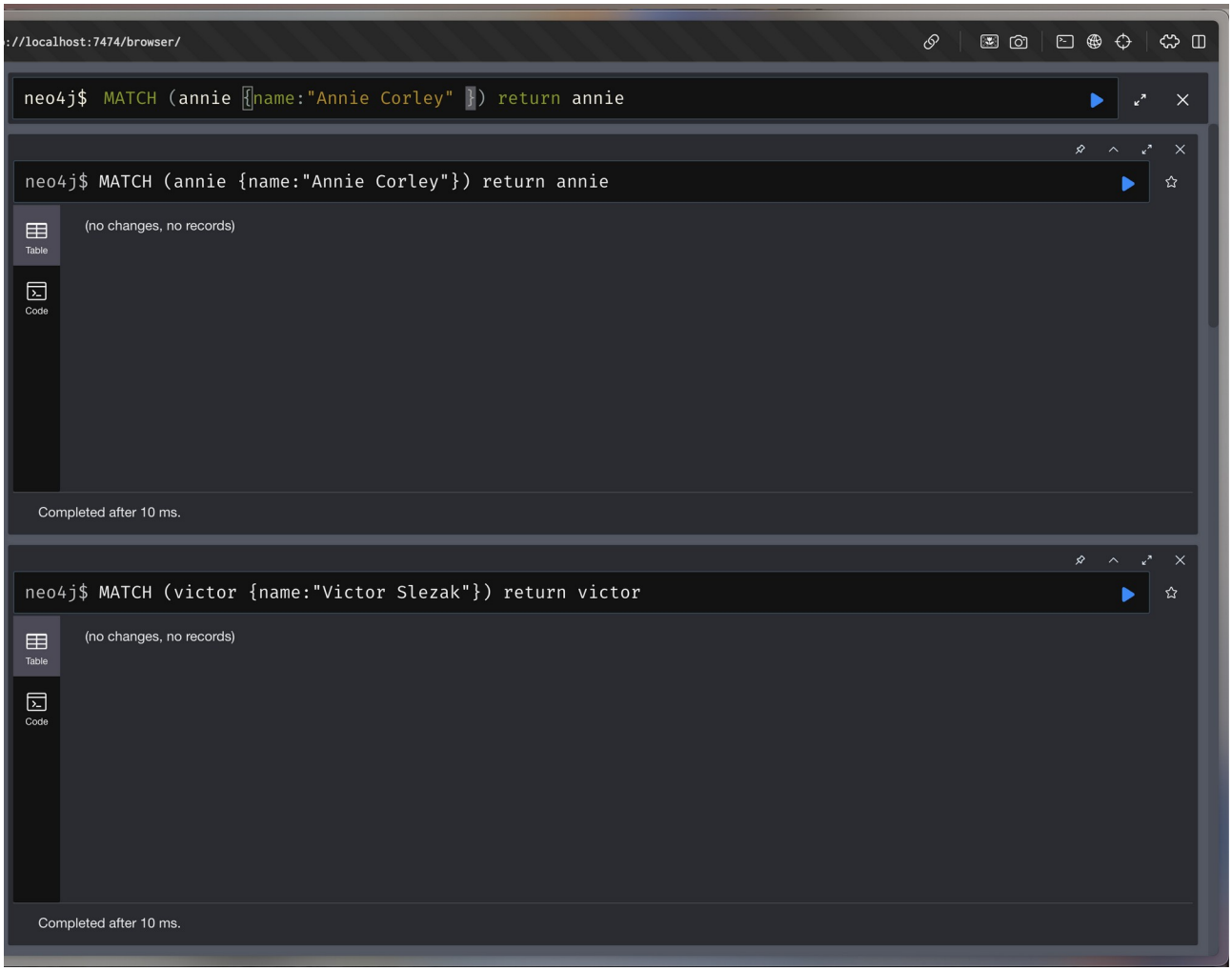


Then as seen below in the main tabs and sub tabs I picked the top three actors in this movie:

- Meryl Streep
- Victor Sizeak
- Annie Corley



Then I performed the same check as to make sure they wouldn't become dummy nodes:

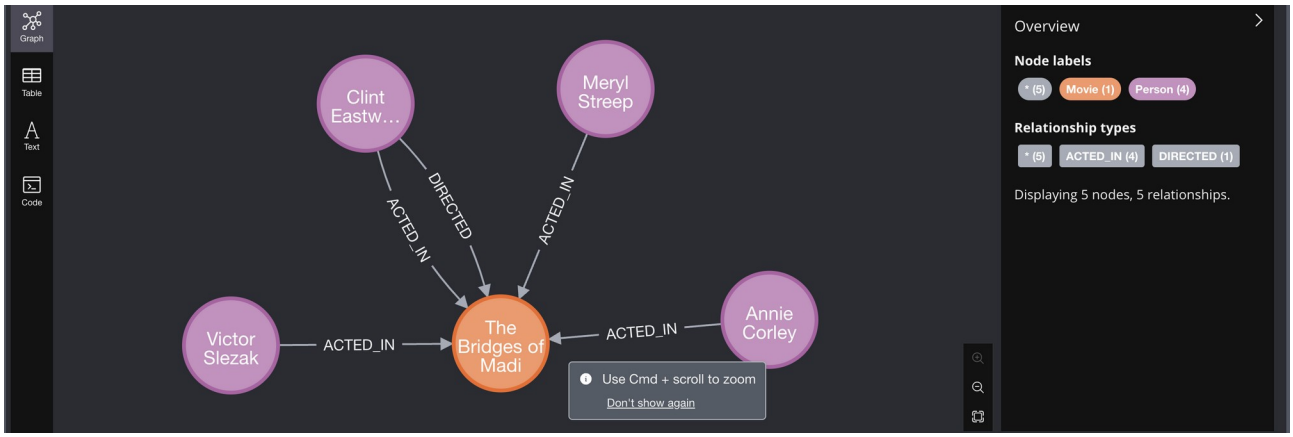


Each of them had no records so I created their vertices and connected them to the new Movie:

```
MATCH (ClintE:Person {name: 'Clint Eastwood'})  // create clint eastwood variable for fixing usage
CREATE (TheBridges:Movie {
  title: 'The Bridges of Madison County',
  released: 1995,
  tagline: 'This kinda certainty comes but just once in a lifetime.'
})
CREATE (MerylStr:Person {name: 'Meryl Streep', born: 1949})
CREATE (AnnieCorl:Person {name: 'Annie Corley', born: 1952})
CREATE (VictorSlez:Person {name: 'Victor Slezak', born: 1957})
CREATE
  (MerylStr)-[:ACTED_IN {roles: ['Francesca Johnson']}]->(TheBridges),
  (AnnieCorl)-[:ACTED_IN {roles: ['Carolyn Johnson']}]->(TheBridges),
  (VictorSlez)-[:ACTED_IN {roles: ['Michael Johnson']}]->(TheBridges),
MERGE (ClintE)-[:ACTED_IN {roles: ['Robert Kincaid']}]->(TheBridges)
MERGE (ClintE)-[:DIRECTED]->(TheBridges);
```

Rather than manually adding each entry, I decided in a separate .cypher file, to create the structure of the adding statement and then manually paste this in as to avoid dummy nodes.

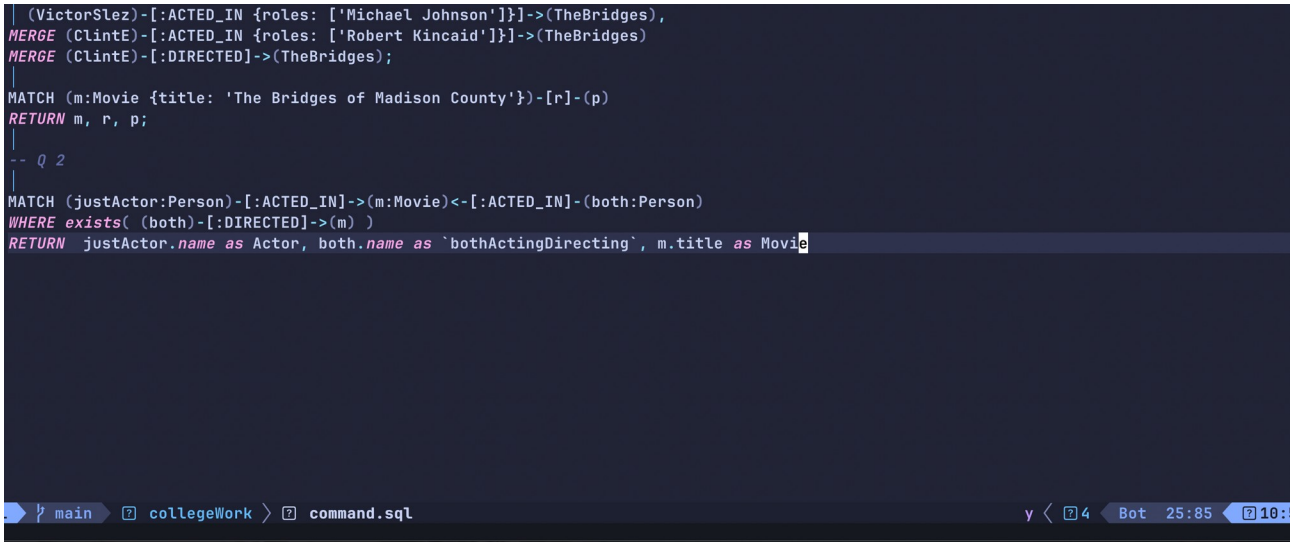
Successfully created person and movie relationships.



Now we have no dummy nodes through the use of merge and making Clint Eastwood a variable in this query through a match statement.

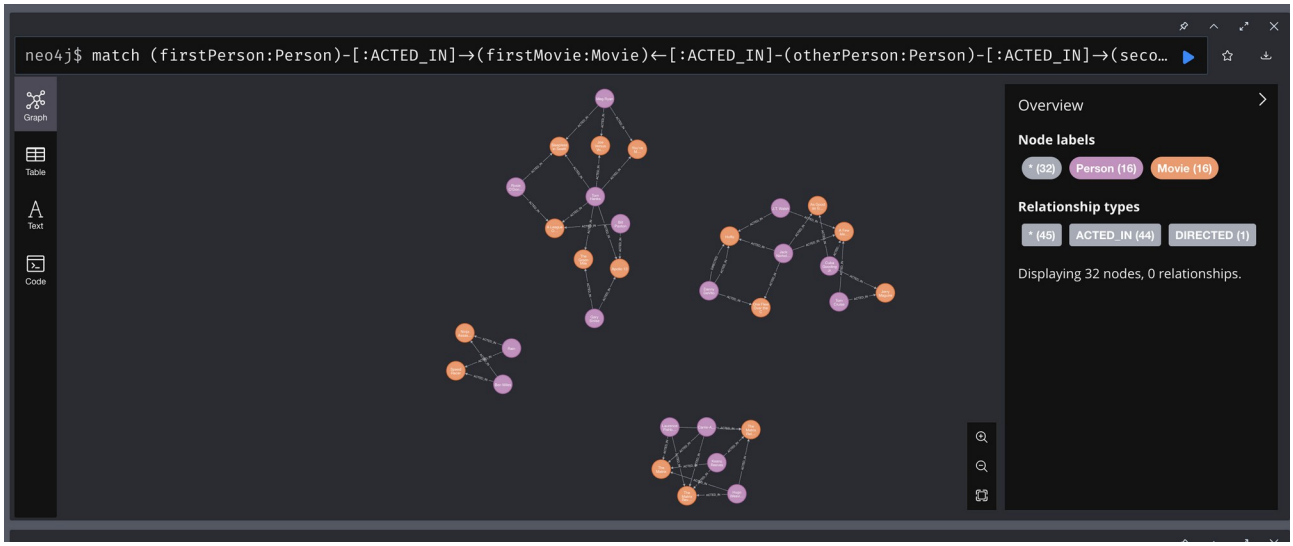
2.
To find this we need to use a match statement that outputs three elements: - the movie itself, the actors in this movie and a person acting and directing.

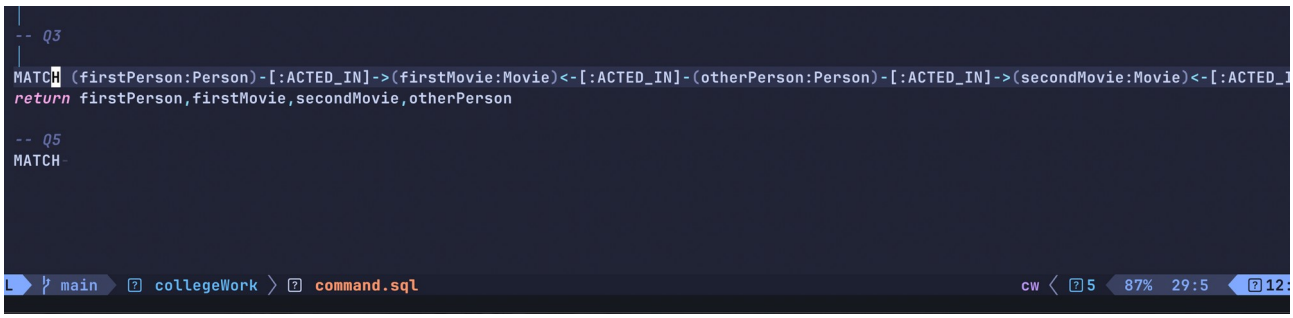
This is my query for that:



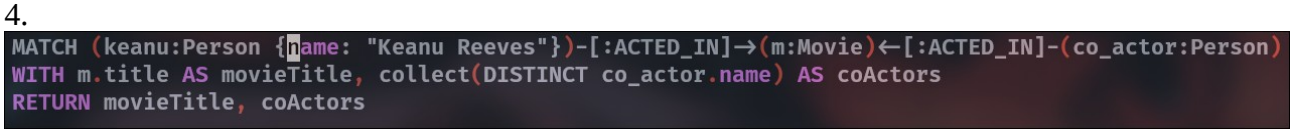
This output does provide duplicates and is only available in text/table format, but in a node format, it would be increasingly difficult to read.

3.

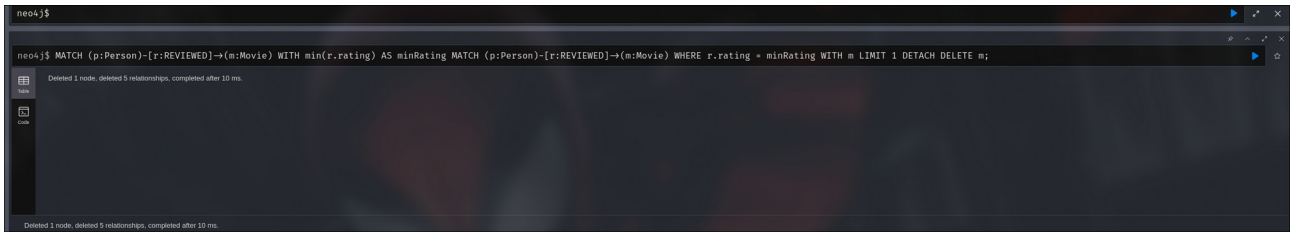




4.



5.



For deleting the movie with the lowest rating, we need to delete both the movie itself, and the leftover relationships:
I kept a limit of 1 just so it only deletes the movie with THE lowest rating.

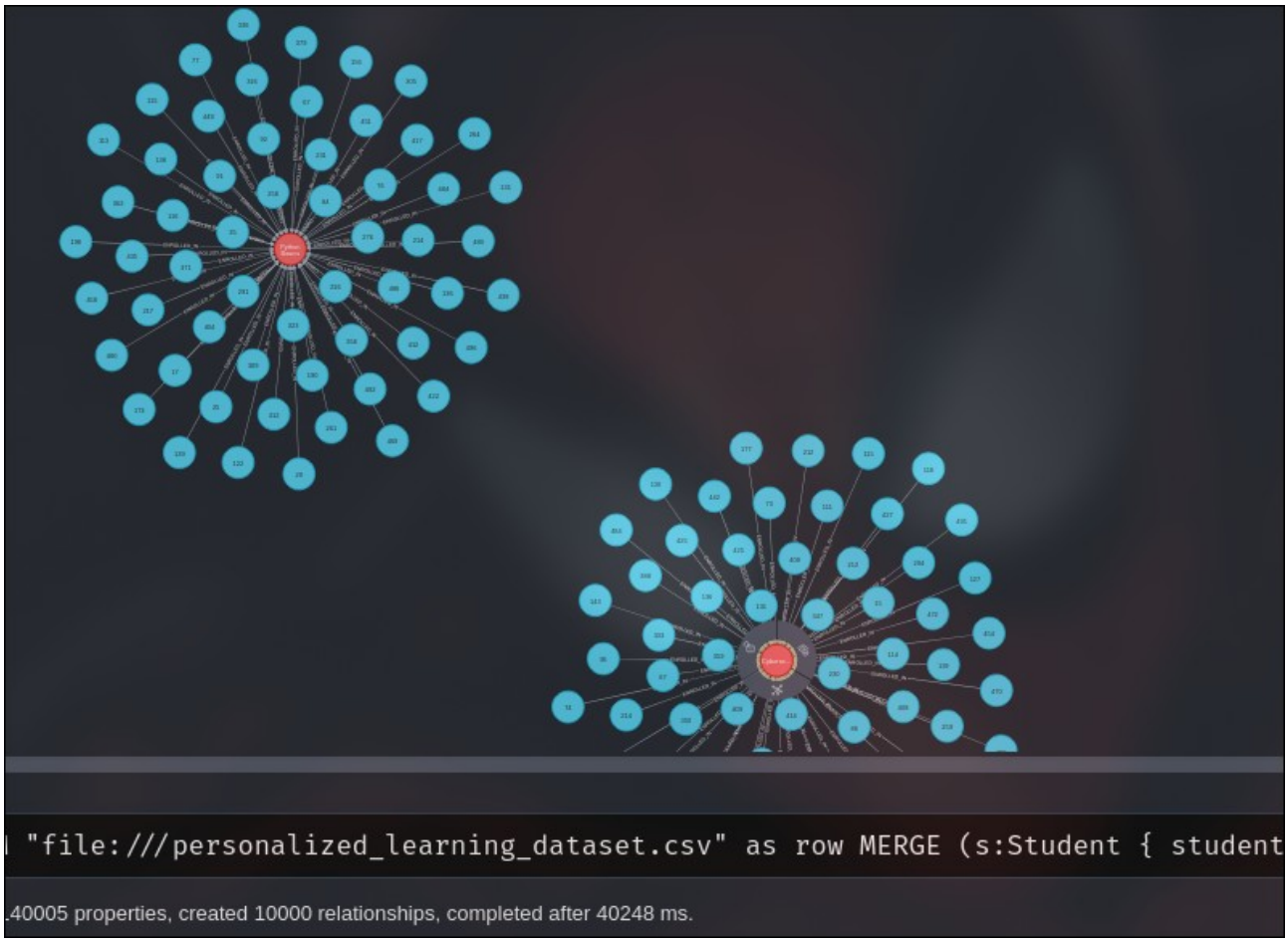
```
-- Q5

MATCH (p:Person)-[r:REVIEWED]→(m:Movie)
WITH min(r.rating) AS minRating
MATCH (p:Person)-[r:REVIEWED]→(m:Movie)
WHERE r.rating = minRating
WITH m
LIMIT 1
DETACH DELETE m;
```

Part 2 (Personalized Learning Database)

- Before simply importing this dataset, we first need to decipher the csv data. From the pdf, we have two distint entities, students and courses. I initially assumed there may be occurrences of students being listed for multiple courses, but upon looking through the file, the ids for students only ever appear once. I was able to then use this command and import the csv:

```
-- Import for next question
LOAD CSV WITH HEADERS FROM "file:///personalized_learning_dataset.csv" as row
MERGE (s:Student {
  studentId: row.Student_ID,
  age: toInteger(row.Age),
  gender: row.Gender,
  educationLevel: row.Education_Level,
  timeSpentOnVideos: toInteger(row.`Time_Spent_on_Videos`),
  quizAttempts: toInteger(row.Quiz_Attempts),
  quizScores: toFloat(row.Quiz_Scores) / 100,
  forumParticipation: toInteger(row.Forum_Participation),
  assignmentCompletionRate: toFloat(row.Assignment_Completion_Rate) / 100,
  engagementLevel: row.Engagement_Level,
  finalExamScore: toFloat(row.Final_Exam_Score) / 100,
  learningStyle: row.Learning_Style,
  feedbackScore: toFloat(row.Feedback_Score),
  dropOutChance: row.Dropout_Likelihood
})
MERGE (c:Course {name: row.Course_Name})
CREATE (s)-[:ENROLLED_IN]->(c);
```



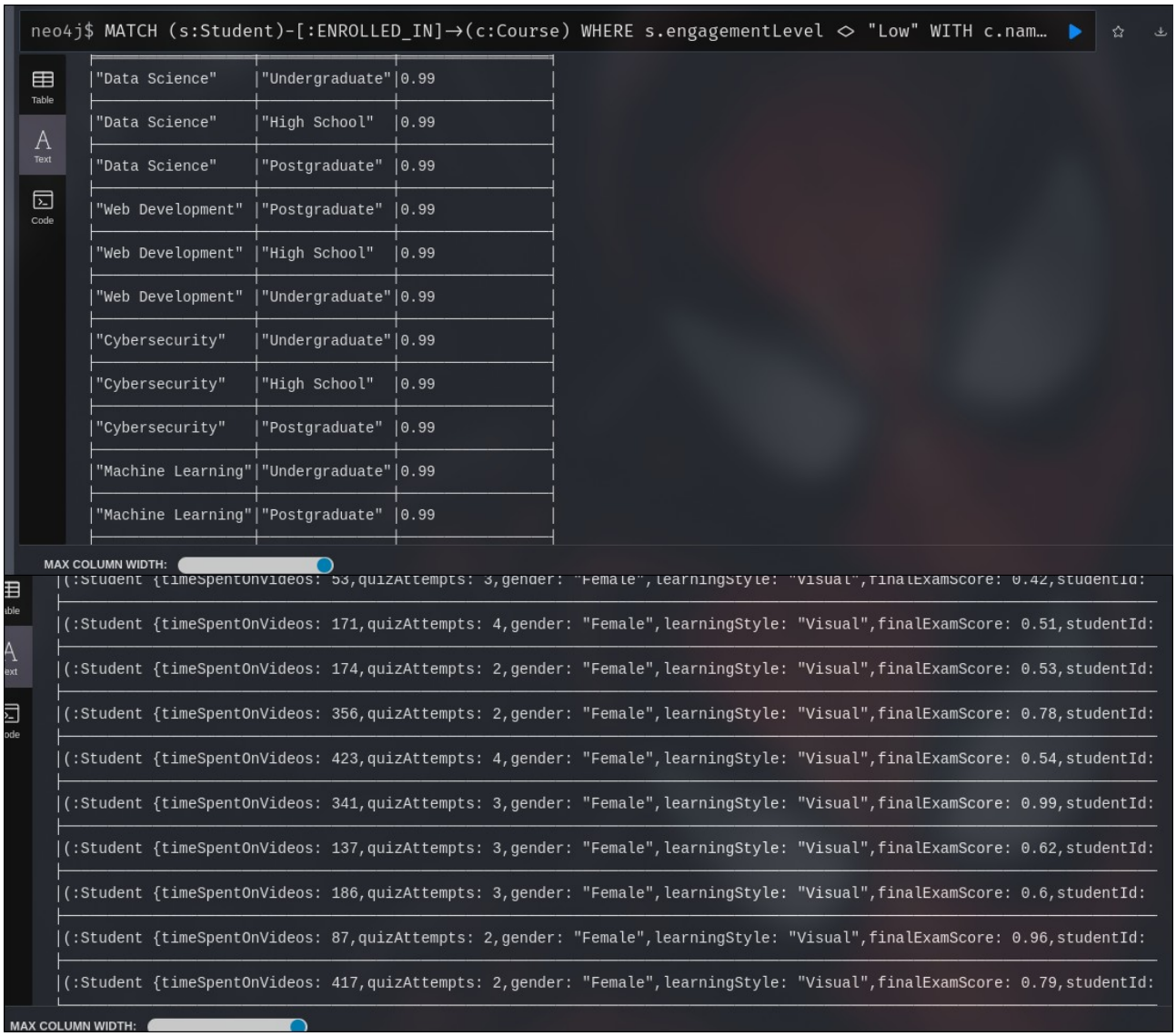
```
| "file:///personalized_learning_dataset.csv" as row MERGE (s:Student { student
```

.40005 properties, created 10000 relationships, completed after 40248 ms.

As seen below, we have hundreds of students each connected to python basics and cyber-security respectfully.

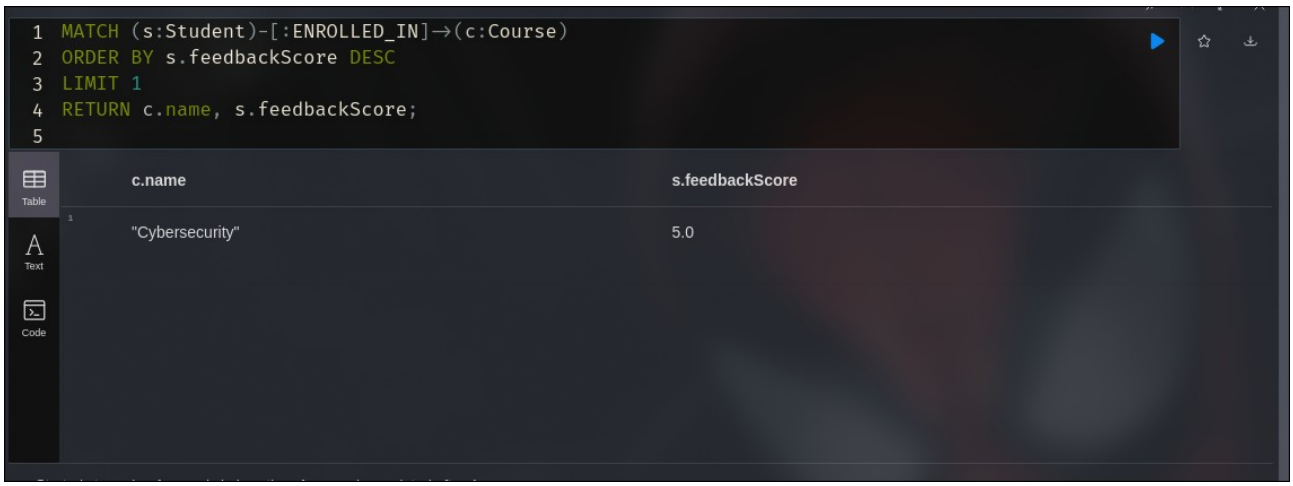
1.

2.




```
MATCH (s:Student)-[:ENROLLED_IN]→(c:Course)
WHERE s.engagementLevel <> "Low"
WITH c.name AS CourseName, s.educationLevel AS EducationLevel, MAX(s.assignmentCompletionRate) AS completionRate
RETURN DISTINCT CourseName, EducationLevel, MaxCompletionRate
ORDER BY completionRate DESC;
```

3.



This simply gives the score any has achieved however a more accurate depiction would be an average:

4.

```
MATCH (s:Student)-[:ENROLLED_IN]→(c:Course)
    WHERE c.name = "Cybersecurity" AND s.engagementLevel="Low"
        AND s.finalExamScore < 0.4 AND s.gender = "Male"
    return COUNT(s);
```

```
neo4j$ MATCH (s:Student)-[:ENROLLED_IN]→(c:Course) WHERE c.name = "Cybersecurity" AND s.engage…
```

Table | Text | Code

	COUNT(s)
1	29

Started streaming 1 records after 5 ms and completed after 6 ms.

5.

```
MATCH (s:Student)
WHERE s.dropOutChance = "Yes" AND s.engagementLevel="Low"
WITH s,collect(DISTINCT s.learningStyle) as learnStyle, collect(DISTINCT s.educationLevel)
as edLevel
    return learnStyle, edLevel, COUNT(s);
```

Table | Text | Code

	learnStyle	edLevel	COUNT(s)
7	["Kinesthetic"]	["Postgraduate"]	23
8	["Auditory"]	["Postgraduate"]	28
9	["Reading/Writing"]	["High School"]	30
10	["Reading/Writing"]	["Undergraduate"]	42
11	["Kinesthetic"]	["High School"]	26
12	["Visual"]	["Postgraduate"]	13

Started streaming 12 records after 7 ms and completed after 18 ms.