Classifying actors on the basis of Awards

Akanksha Chaturvedi
Huzefa Shabbir Sadikot
Nagul Meera Shaik

Problem Statement

We have to create nodes using the graph database neo4j and show the relationship between various nodes.

Abstract:

The Neo4j graph database management system was used in this project to represent and analyse a complex network of data, and the results are presented in this report. Large datasets must be imported into Neo4j for the project in order to build a graph database that shows the connections between the data's elements. To gain useful insights from the data, the Cypher query language was used to query the graph database. The report gives a broad overview of the project's goals, approach, and outcomes. It emphasizes the advantages of modelling and analysing complicated networks of data using a graph database like Neo4j. The report offers suggestions for future work as well as a discussion of some of the difficulties encountered during the project.

Introduction:

This Neo4j project provides a powerful and flexible solution for analysing complex relationships in the entertainment industry. It offers a new and innovative way to manage and analyse large and complex networks of data, which can provide valuable insights for actors, directors, producers, and other stakeholders in the industry.
In the entertainment industry, movies generate a vast amount of data, with thousands of actors being nominated for various roles each year. This data is often dispersed across multiple sources, making it difficult to manage and analyse effectively. Additionally, the relationships between actors, movies, and awards are often complex and interconnected, which further complicates the analysis.
The Neo4j project that we have undertaken aims to model and analyse the relationships between actors who have acted in movies and have been nominated for a certain role. This project utilizes the power and flexibility of Neo4j, a graph database management system, to represent these relationships as nodes and edges.

Brief overview of Neo4j:

The graph database management system Neo4j offers a potent and adaptable method for modelling and analysing intricate networks of data. Neo4j employs a graph data model to represent data as nodes and edges, in contrast to conventional relational databases that store data in tables.

Nodes in Neo4j stand for entities, whereas edges represent the connections between those entities. Properties on nodes and edges can be used to record extra information about the entities and relationships.
Users can query the graph database using Neo4j's flexible query language Cypher, which has an easy-to-understand syntax. Data can be retrieved, altered, and the connections between nodes and edges can be shown via cypher queries.

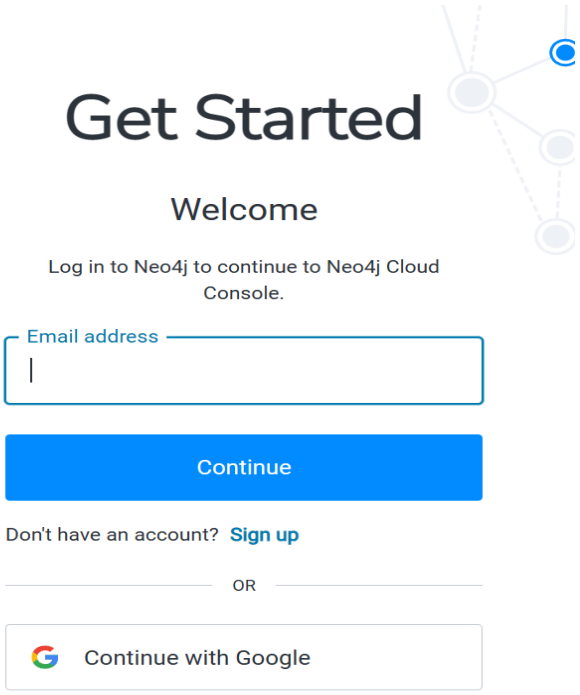
ADVANTAGES OF Neo4j:

1. **Flexible and expressive data model**: Neo4j's graph data model allows for a more flexible and expressive way to represent complex relationships between entities than traditional relational databases.
2. **Performance:** Neo4j is optimized for querying graph data, which makes it faster and more efficient than traditional relational databases for certain types of queries.
3. **Scalability:** Neo4j can scale horizontally to handle large volumes of data and high query loads, making it a great choice for applications that require high scalability.

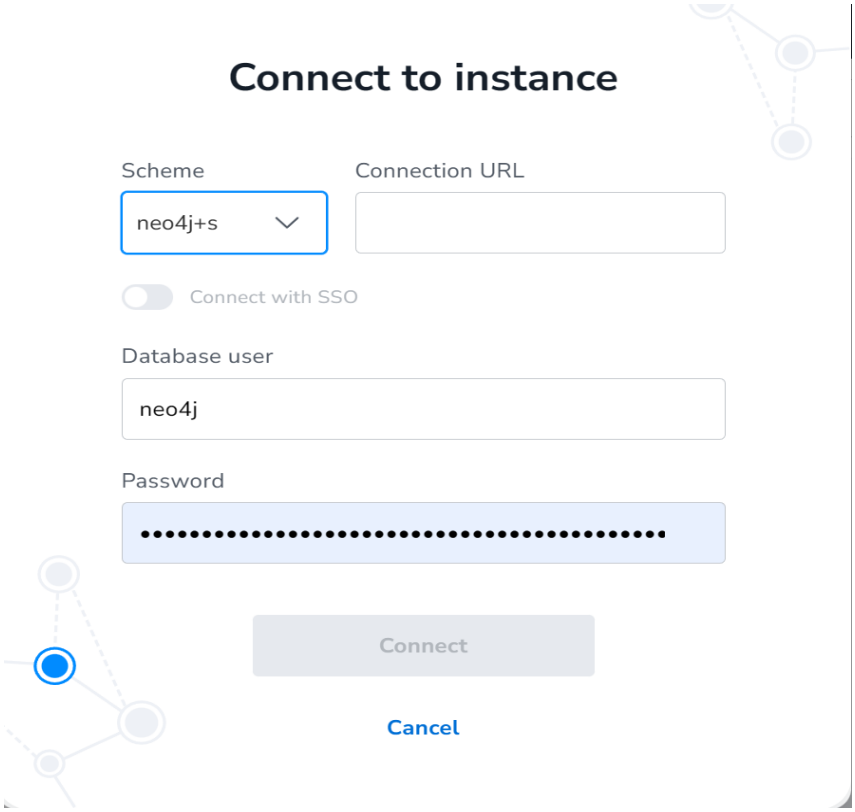
Project Introduction:

We are making a project on the academy awards for the actors, directors who are nominated for a particular role. We have formed a relationship between the same.
We have used graph database neo4j for this purpose because it gives us clarity about the relationship and nodes.
We have used cypher to make our codes and for the coding language we are using cql.
Cypher is a declarative query language used to interact with the Neo4j graph database management system. It is designed to be expressive, readable, and easy to learn, with a syntax that is like SQL and other database query languages. In Cypher, queries are expressed using ASCII art-style patterns that represent the nodes and relationships in the graph.
CQL, on the other hand, stands for Cassandra Query Language, which is a query language used to interact with the Apache Cassandra NoSQL database. CQL is a SQL-like language with a similar syntax to SQL, but it is designed specifically for Cassandra's data model.
While both Cypher and CQL are query languages used in different types of databases, they have distinct syntaxes and are designed for different data models. Cypher is used specifically for Neo4j's graph data model, while CQL is used for Cassandra's column-family data model.
Let’s begin with our project:

STEP 1: First login into the neo4j graph database using your credentials.



STEP 2: After login you will get a console where you can type your queries. Here we will first create the csv file using the create commands. First create the instance and connect it using the credentials.

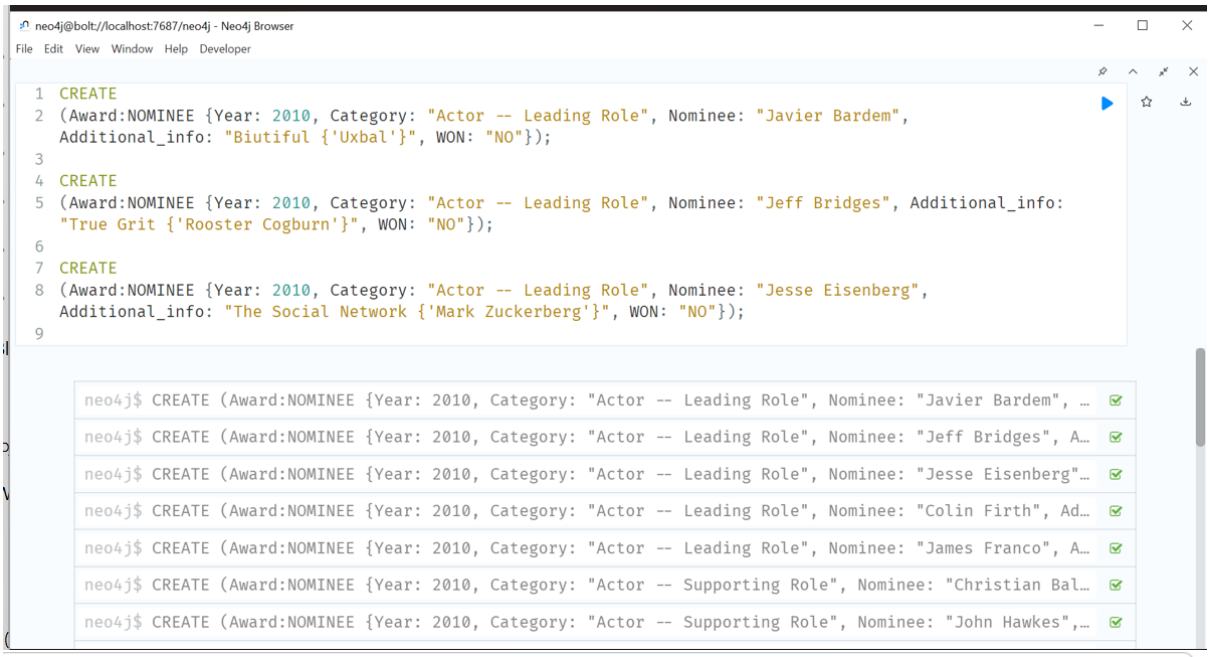


MODELLING DATABASE:

Modelling the graph database involves various steps like identifying the entities and relationship in your model, defining the node labels and edges, define the relationship types, adding properties to nodes and relationships, creating the graphs, and querying the graph.

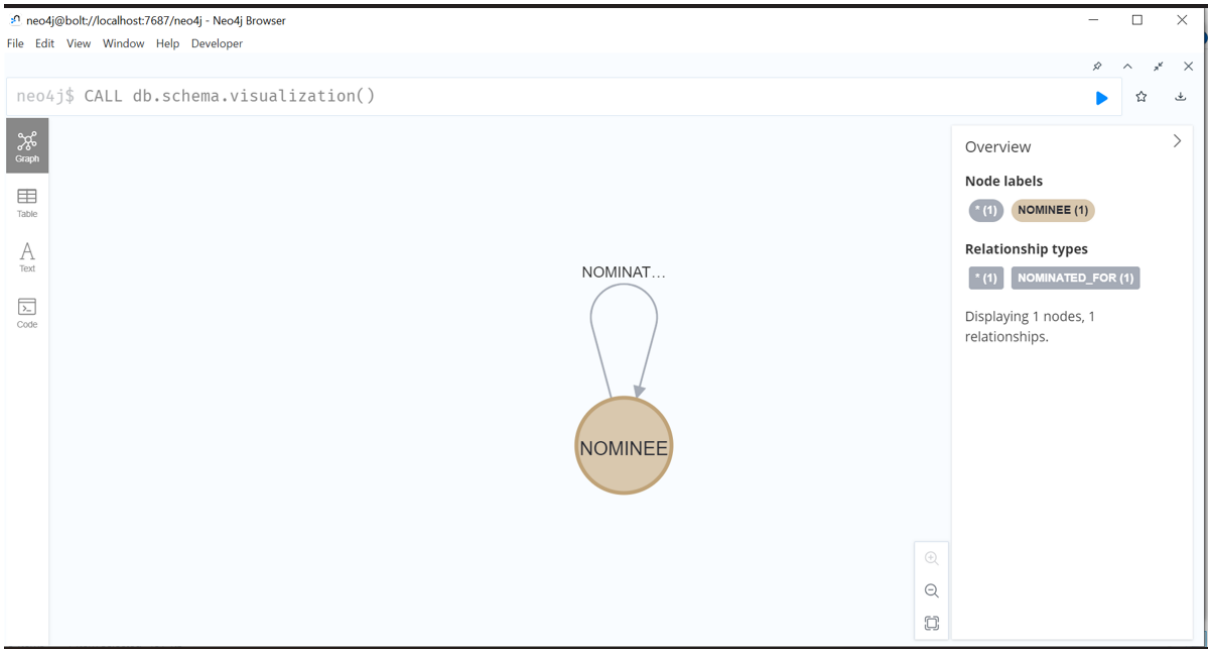
Here, first we have created the database, created the labels for nodes and defined the relationships between the nodes.

STEP 3: First create the database using the following commands. Here we have created the nodes.




```
1  CREATE
2  (Award:NOMINEE {Year: 2010, Category: "Actor --
   Leading Role", Nominee: "Javier Bardem",
   Additional_info: "Biutiful {'Uxbal'}", WON:
   "NO"});
3
4  CREATE
5  (Award:NOMINEE {Year: 2010, Category: "Actor --
   Leading Role", Nominee: "Jeff Bridges",
   Additional_info: "True Grit {'Rooster Cogburn'}",
   WON: "NO"});
6
```

Database schema will look like this:



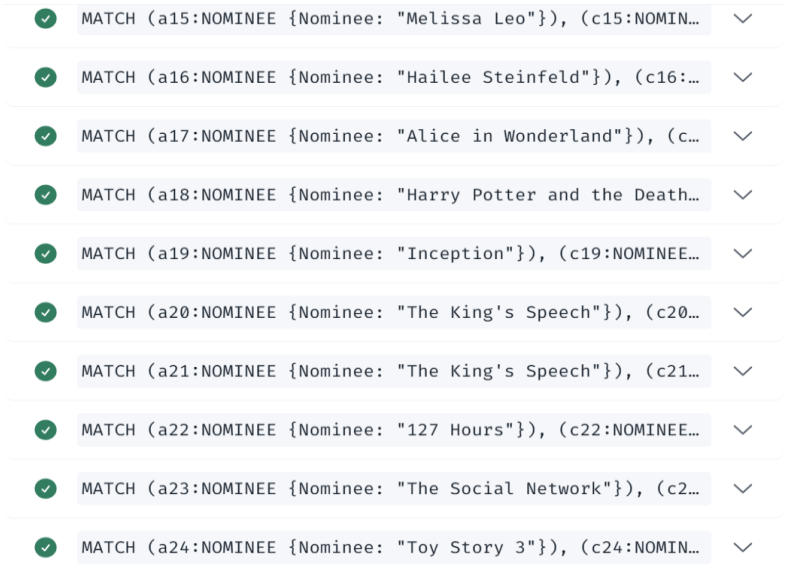
STEP 4: After creating the nodes, we need to create the relationship between the nodes. We can create the relationship using the following commands. We have created relationship between actors and awards they were nominated for.

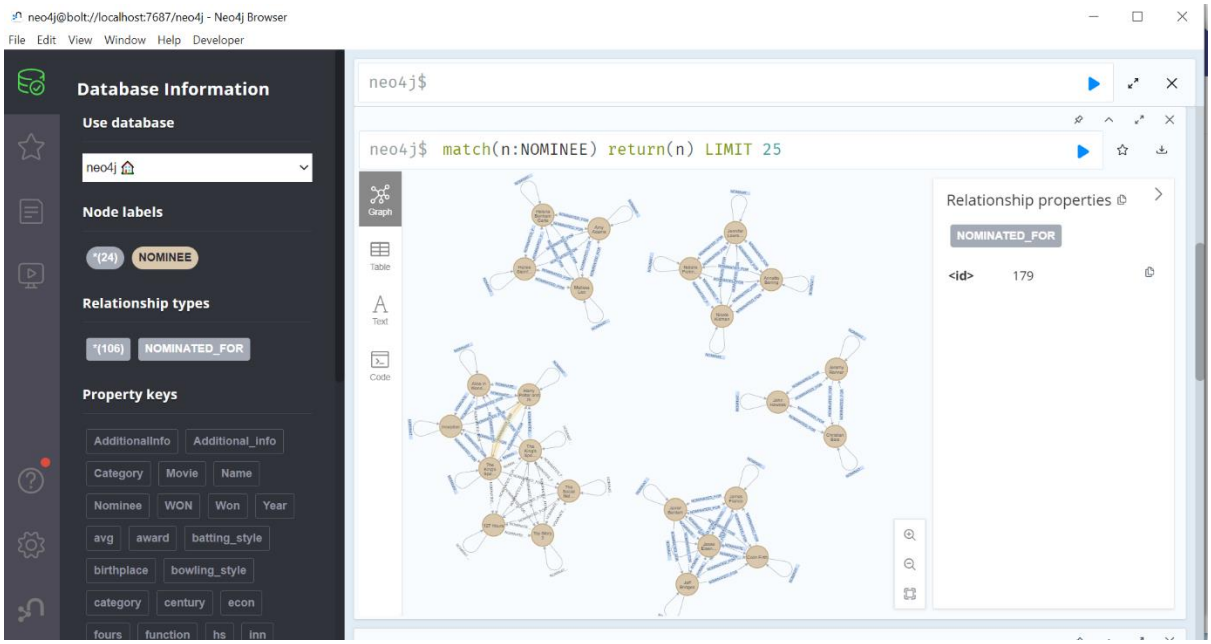
```
MATCH (a1:NOMINEE {Nominee: "Javier Bardem"}), (c1:NOMINEE
{Category: "Actor -- Leading Role"})
CREATE (a1)-[:NOMINATED_FOR]→(c1);

MATCH (a2:NOMINEE {Nominee: "Jeff Bridges"}), (c2:NOMINEE
{Category: "Actor -- Leading Role"})
CREATE (a2)-[:NOMINATED_FOR]→(c2);

MATCH (a3:NOMINEE {Nominee: "Jesse Eisenberg"}), (c3:NOMINEE
{Category: "Actor -- Leading Role"})
CREATE (a3)-[:NOMINATED_FOR]→(c3);

```




neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
Database Information
Use database
neo4j
Node labels
*(24)
NOMINEE
Relationship types
*(106)
NOMINATED_FOR
Property keys
AdditionalInfo
Additional_info
Category
Movie
Name
Nominee
WON
Won
Year
avg
award
batting_style
birthplace
bowling_style
category
century
econ
fours
function
hs
inn
neo4j$
neo4j$ match(n:NOMINEE) return(n) LIMIT 25
Graph
Table
Text
Code
Relationship properties
NOMINATED_FOR
<id> 179

MODEL WITH LOOPS:

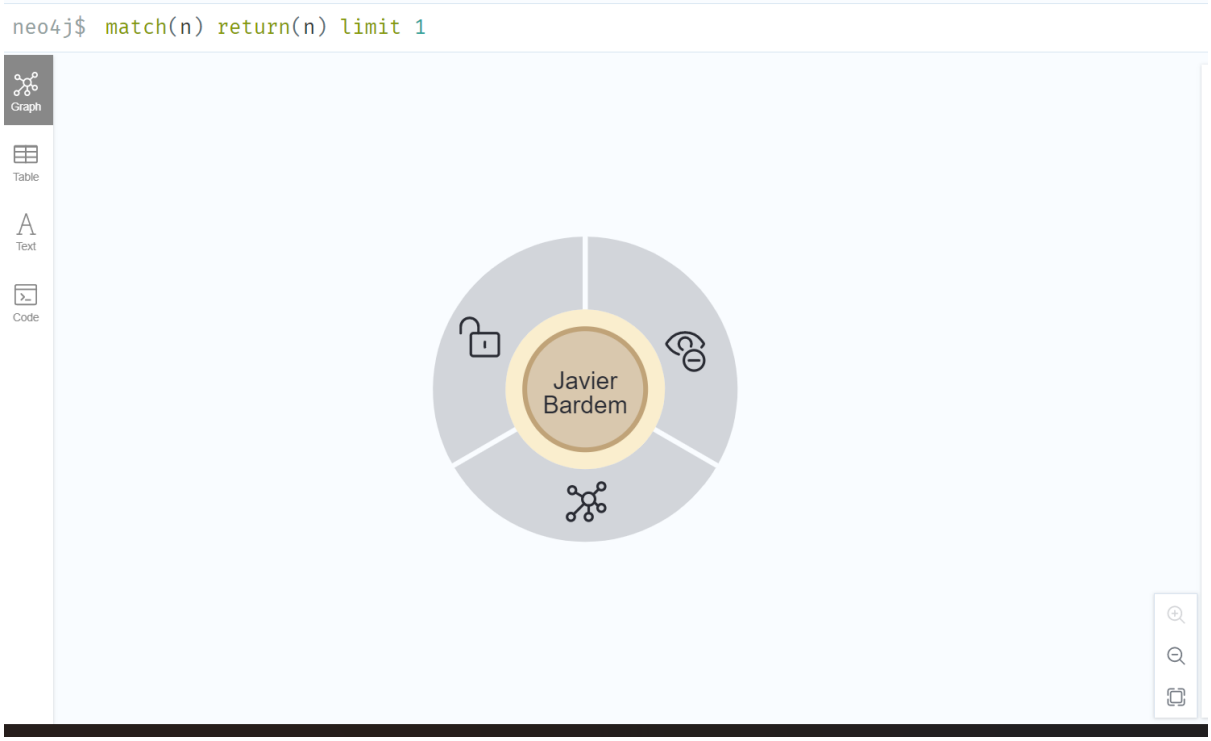
By running the following commands we can check the property of nodes that our model have some cycles. We have used the command **MATCH (n)**

WHERE (n)-[:RELATIONSHIP*]->(n)

RETURN n

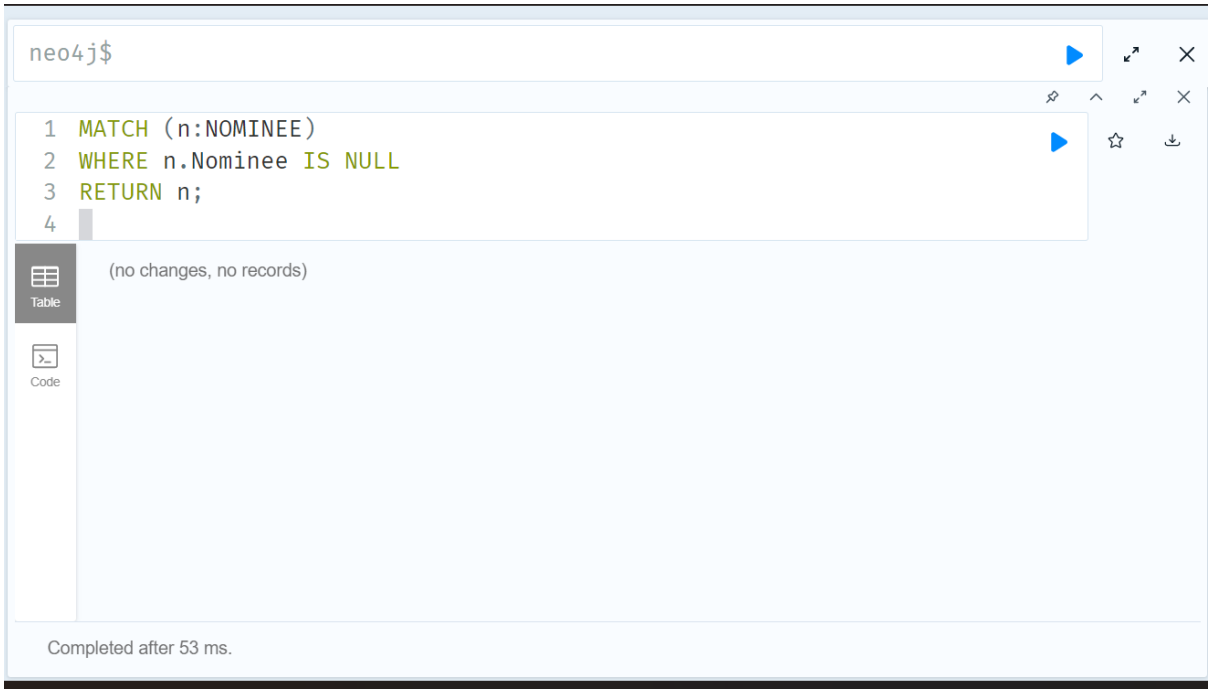
To check whether our model have loops or not. In this query, n is a placeholder for any node in your graph. The [:RELATIONSHIP*] syntax specifies that we want to match any path that includes zero or more relationships. The -> and <- arrows indicate the direction of the relationship. The WHERE clause filters the results to only include nodes that have a relationship that forms a cycle with itself. Finally, the RETURN clause returns all nodes that match the query.

If this query returns any nodes, it means that your graph has loops or cycles. You can inspect the results to see which nodes are involved in the cycles and how they are connected. You can also modify the query to include additional constraints or filters to further refine your search for cycles in your graph.



CLEANING THE DATA FOR ANY NULL VALUES:

Step 5: We will check for the null values. In the following we can see that there are no null values present in our database.



TRANSFORMING THE DATA:

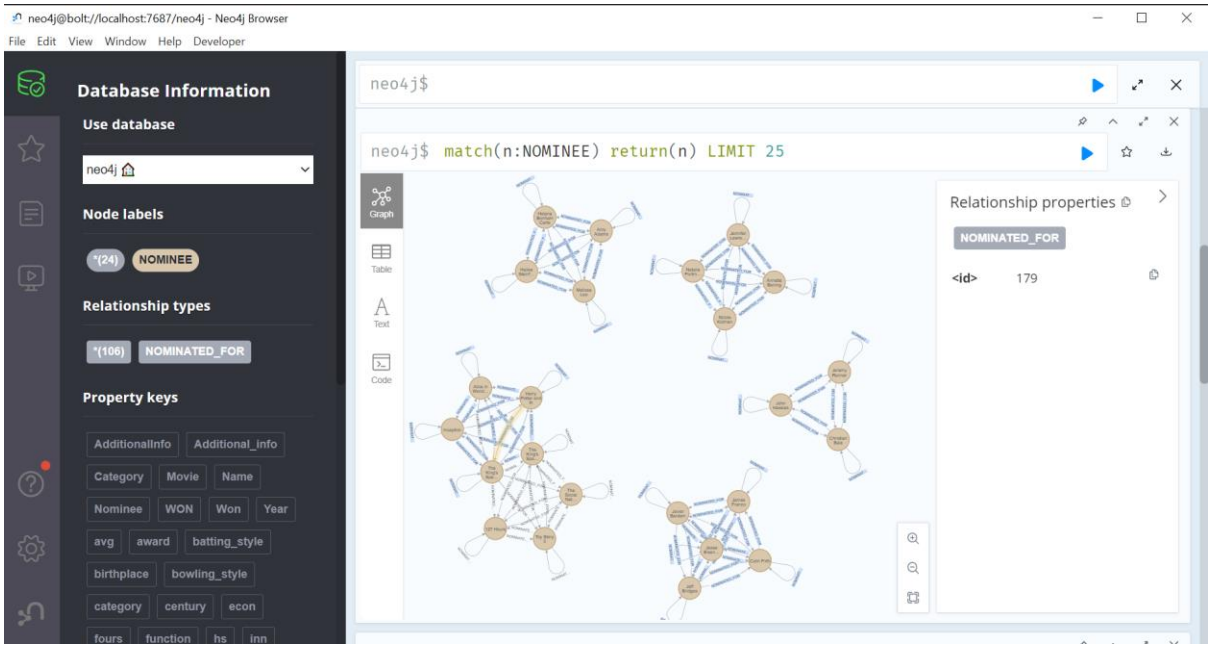
To transform data in a Neo4j graph database, you can use the Cypher query language to create new nodes, relationships, and properties based on existing data. Here are some common techniques for data transformation in Neo4j:

- Create new nodes: You can create new nodes in your graph by selecting data from existing nodes and relationships and creating new nodes with the CREATE clause.
- Update existing nodes: You can update properties on existing nodes using the SET clause.
- Delete nodes and relationships: You can delete nodes and relationships from your graph using the DELETE clause.
- Create new relationships: You can create new relationships between existing nodes using the CREATE clause.

STEP 6: After checking the null values let's look for the relationship between the nodes.

```
1  MATCH (a1:NOMINEE {Nominee: "Javier Bardem"}),
   (c1:NOMINEE {Category: "Actor -- Leading Role"})
2  CREATE (a1)-[:NOMINATED_FOR]→(c1);
3
4  MATCH (a2:NOMINEE {Nominee: "Jeff Bridges"}),
   (c2:NOMINEE {Category: "Actor -- Leading Role"})
5  CREATE (a2)-[:NOMINATED_FOR]→(c2);
6
7  MATCH (a3:NOMINEE {Nominee: "Jesse Eisenberg"}),
   (c3:NOMINEE {Category: "Actor -- Leading Role"})
8  CREATE (a3)-[:NOMINATED_FOR]→(c3);
9
```

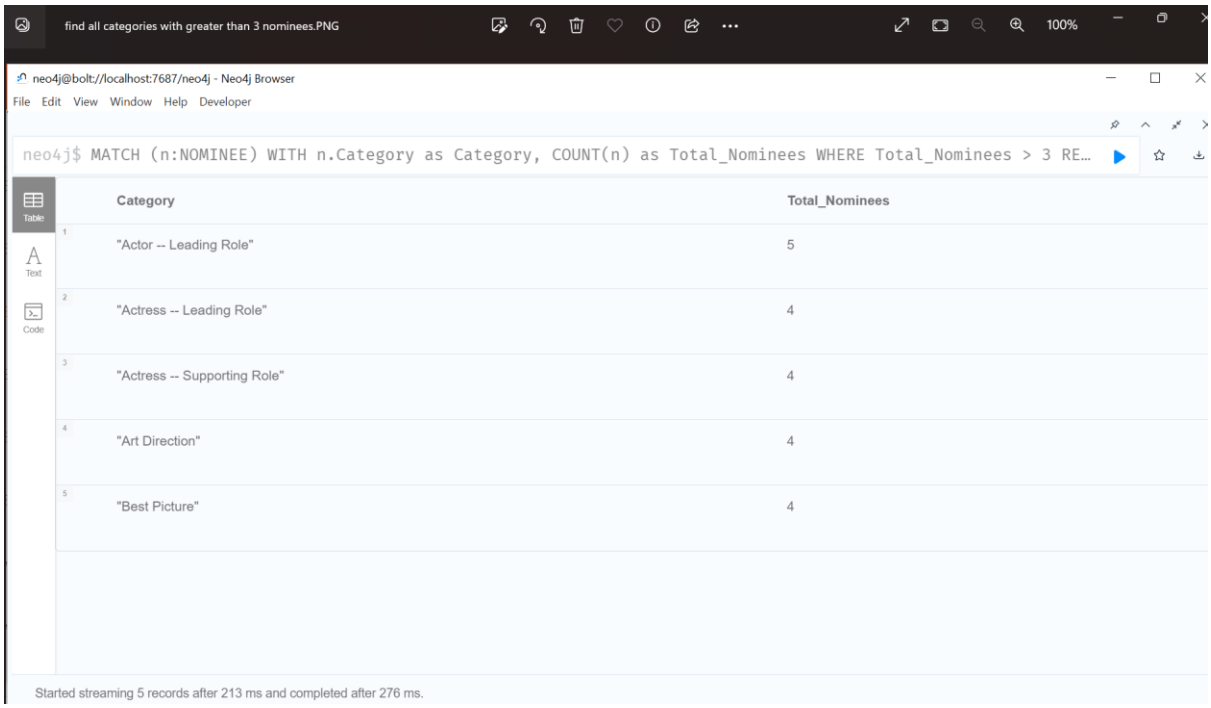
```
MATCH (a3:NOMINEE {Nominee: "Jesse Eisenberg"}), (c3:NOM…
MATCH (a4:NOMINEE {Nominee: "Colin Firth"}), (c4:NOMINEE…
MATCH (a5:NOMINEE {Nominee: "James Franco"}), (c5:NOMINE…
MATCH (a6:NOMINEE {Nominee: "Christian Bale"}), (c6:NOMI…
MATCH (a7:NOMINEE {Nominee: "John Hawkes"}), (c7:NOMINEE…
```


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
Database Information
Use database
neo4j
Node labels
*(24)
NOMINEE
Relationship types
*(106)
NOMINATED_FOR
Property keys
AdditionalInfo
Additional_Info
Category
Movie
Name
Nominee
WON
Won
Year
avg
award
batting_style
birthplace
bowling_style
category
century
econ
fours
function
hs
inn
neo4j$
neo4j$ match(n:NOMINEE) return(n) LIMIT 25
Graph
Table
Text
Code
Relationship properties
NOMINATED_FOR
<id> 179

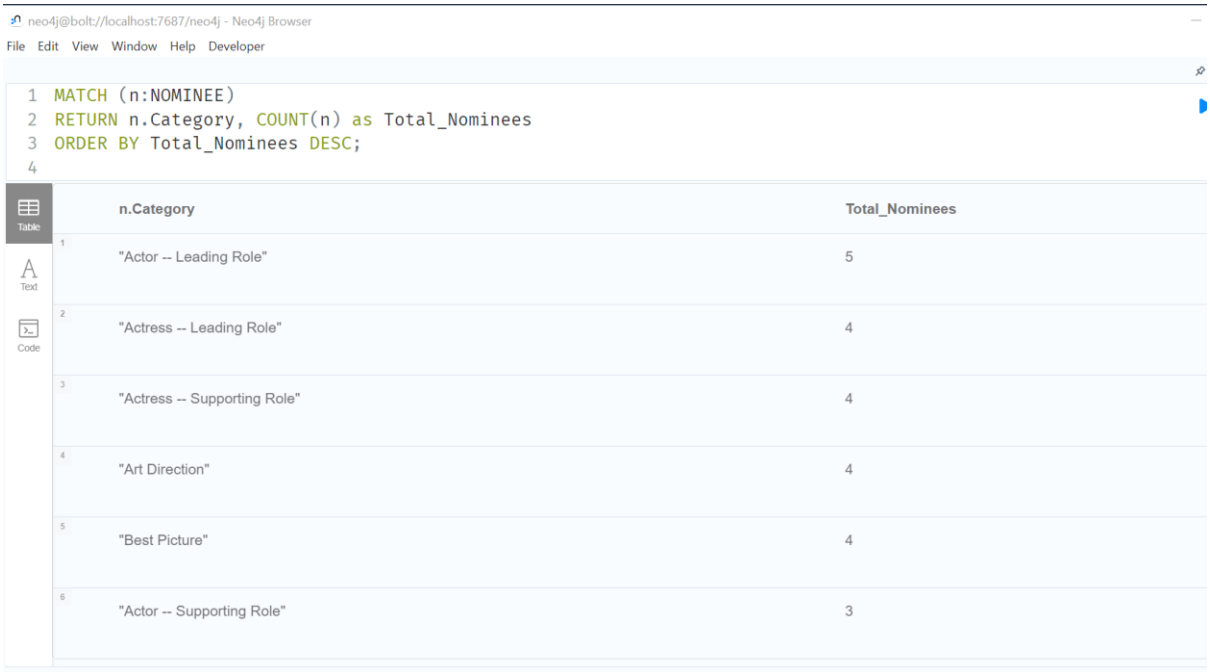
PERFORMING AGGREGATION OPERATIONS:

To perform aggregation operations in Neo4j, you can use the Cypher query language to group nodes by common properties and calculate aggregate values like counts, sums, averages, and more. Here are some examples of common aggregation operations in Neo4j:

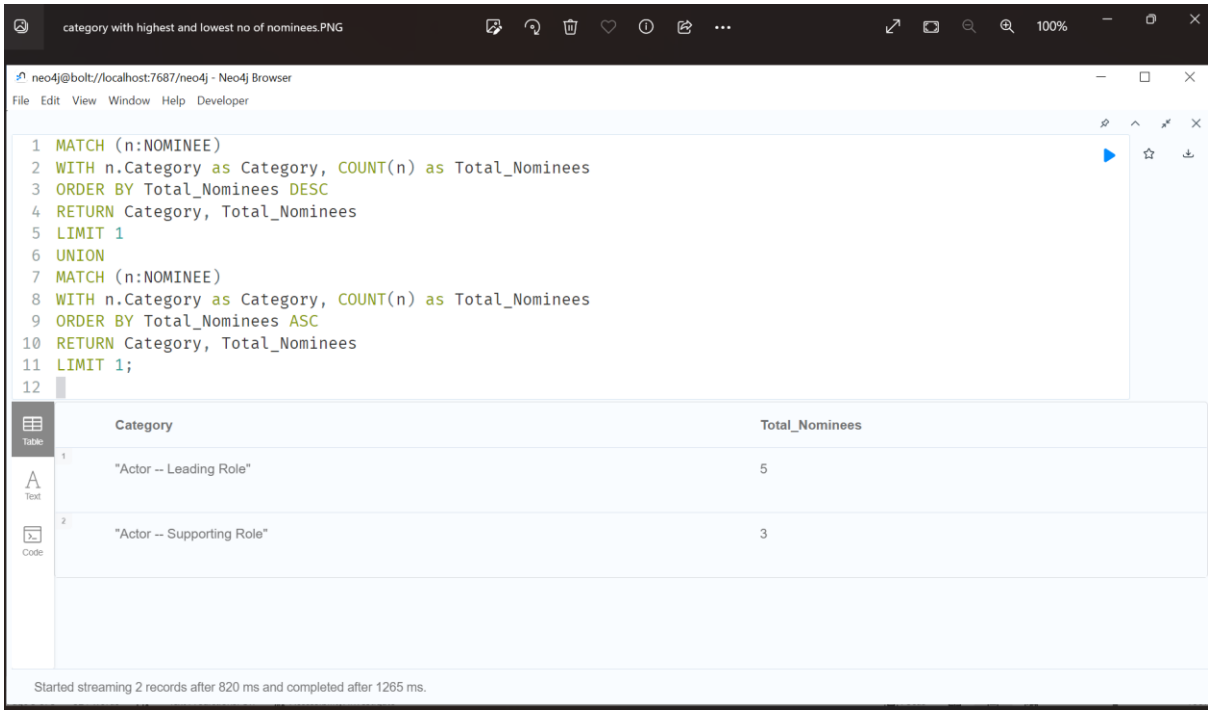
STEP 7: Let's see the counts of all the nodes and relationships we have created. Below we can see all the actors more than 3 times nominated. Here we have used COUNT() function to count the number of nodes to match a certain pattern.



Below is the total number of nominees in each category. Here, we have used ORDER BY() clause to group the nodes by common properties.



Below is the number of highest and lowest number of nominees. Here, we have used UNION () clause to calculate the sum and average of the property across a group of functions.



As we can see from above output, we have total number of people who won the awards is 6 and the people who lost it is 18.

Conclusion:

In conclusion, the Neo4j project for showing the relationship between the nodes of actors who acted in a movie and were nominated for a certain role has been a success. By utilizing the graph data model and Cypher query language, we were able to represent and analyze the complex relationships between the entities in the data.

Through the project, we were able to identify the different movies and roles that actors were nominated for, as well as the different awards that they received. The project has also allowed us to gain insights into the patterns and trends in the data, which can be useful in various applications such as the film industry, marketing, and research.

Overall, Neo4j has proven to be a powerful and flexible tool for managing and analyzing graph data. Its performance, scalability, and ease of use make it an ideal choice for a wide range of applications, including those that require the analysis of complex networks of data such as the relationships between actors and movies.