GRAPH DATABASE

**ABSTRACT**

This report presents an in-depth analysis of a comprehensive dataset of over 5,000 movies from IMDb, examining various factors such as budget, rating, and profitability. The data was stored in Neo4j and processed using a combination of queries and visualizations. The analysis identified key findings such as the highest grossing and highest/lowest rated movies (including the actors and directors associated with them), the most successful genre, the movie with the highest number of critic reviews, the directors with the most number of movies, and those who directed the highest budget movies between 2009 and 2013. The study also includes CRUD operations performed in the databases, with Python used for analysis.In addition, the report discusses the use of containerization and orchestration in Neo4j. Overall, this analysis provides valuable insights into trends and preferences in the movie industry, allowing for a deeper understanding of the factors that contribute to a movie's success.

Top of Form

Bottom of Form

**Cleaning the dataset using python:**

* The jupyter notebook file will be uploaded.
* All the null values are removed
* The genre column is split into multiple columns with Boolean values

**Data Model for the IMDB dataset:**

A picture containing diagram

Description automatically generated

We have created three nodes namely: actor, movie\_title and director and the properties are as follows:

* Movie\_title: id, title, title\_year, imdb\_rating, budget, gross, duration, number of reviews by critics and number of reviews by users
* Actor:id, name, actor\_facebook\_likes, id
* Director: director\_facebook\_likes,name id

The relationship between actor and movie\_title is “acted\_in” as many actors can be a part of many movies and a movie is directed by a director, the relationship between movie\_title and director is directed by-from movie\_title to director.

**Creating three nodes and relationships:**

**Query:**

LOAD CSV WITH HEADERS FROM "file:///imdb.csv" AS row

MERGE (m:Movie {title: row.movie\_title})

ON CREATE SET m.duration = toInteger(row.duration),

m.director\_name = row.director\_name,

m.imdb\_score = toFloat(row.imdb\_score),

m.title\_year = toInteger(row.title\_year),

m.num\_critic\_for\_reviews = toInteger(row.num\_critic\_for\_reviews),

m.num\_user\_for\_reviews = toInteger(row.num\_user\_for\_reviews),

m.budget = toFloat(row.budget),

m.gross = toFloat(row.gross)

MERGE (a:Actor {name: row.actor\_1\_name})

ON CREATE SET a.actor\_facebook\_likes = toInteger(row.actor\_1\_facebook\_likes)

MERGE (m)-[:ACTS\_IN]->(a)

MERGE (d:Director {name: row.director\_name})

ON CREATE SET d.director\_facebook\_likes = toInteger(row.director\_facebook\_likes)

MERGE (d)-[:DIRECTED]->(m)

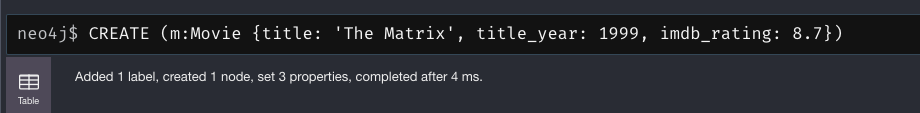
Graphical user interface, text, website

Description automatically generated

****

**Adding a node:**

CREATE (m:Movie {title: 'The Matrix', title\_year: 1999, imdb\_rating: 8.7})



**Adding a relationship:**

MATCH (m:Movie {title: 'The Matrix'})

MATCH (a:Actor {name: 'Keanu Reeves'})

MERGE (a)-[:ACTS\_IN]->(m)

Graphical user interface, text

Description automatically generated

MATCH (a:Actor)-[r:ACTS\_IN]->(m:Movie {title: 'The Matrix'}) RETURN a, r, m

Graphical user interface

Description automatically generated

**Updating a node:**

MATCH (m:Movie {title: 'The Matrix'}) SET m.title\_year = 1998

**Graphical user interface, text

Description automatically generated**

MATCH (m:Movie {title: 'The Matrix'})

RETURN m

**Graphical user interface, application, Teams

Description automatically generated**

**Deleting a node with the movie\_title “Quantum\_of\_solace”**

MATCH (m:Movie {title: 'Quantum of Solace'})

DETACH DELETE m

Graphical user interface, text, application

Description automatically generated

**Checking if the node still exists:**

MATCH (m:Movie {title: 'Quantum of Solace'})

RETURN m

Graphical user interface, text

Description automatically generated

**QUERIES:**

**1.Top 10 highest grossing movies:**

**QUERY:**

MATCH (m:Movie)WHERE toInteger(m.gross) IS NOT NULL

RETURN DISTINCT m.title, toInteger(m.gross)

ORDER BY toInteger(m.gross) DESC

A picture containing table

Description automatically generated

**2.Query to show the profit of the movies each year**

**QUERY:**

MATCH (m:Movie)

WHERE toInteger(m.gross) IS NOT NULL AND toInteger(m.budget) IS NOT NULL AND toInteger(m.title\_year) IS NOT NULL

WITH m, toInteger(m.gross) - toInteger(m.budget) AS profit

RETURN m.title\_year AS title\_year, sum(profit) AS total\_profit

ORDER BY title\_year

Graphical user interface, text, application, chat or text message

Description automatically generated

**3.Top 5 highest IMDB rating movies**

**QUERY:**

MATCH (m:Movie)

WHERE m.imdb\_score IS NOT NULL

RETURN m.title, m.imdb\_score

ORDER BY m.imdb\_score DESC

LIMIT 5

**A picture containing table

Description automatically generated**

**4.Top 5 years which had the highest budget:**

**QUERY:**

MATCH (m:Movie)

WHERE m.budget IS NOT NULL AND m.title\_year IS NOT NULL

WITH m.title\_year AS year, SUM(m.budget) AS total\_budget

RETURN year, total\_budget

ORDER BY total\_budget DESC

LIMIT 5

Text

Description automatically generated

**5.The name of the movie with the highest number of critic reviews and its profit:**

**QUERY:**

MATCH (m:Movie)

WHERE m.num\_user\_for\_reviews IS NOT NULL AND m.num\_critic\_for\_reviews IS NOT NULL

RETURN m.title AS movie\_title, m.num\_critic\_for\_reviews AS critic\_reviews, (toInteger(m.gross)-toInteger(m.budget)) AS movie\_profit

ORDER BY m.num\_critic\_for\_reviews DESC

LIMIT 1

**Graphical user interface, text

Description automatically generated**

**6.Name of the directors who have directed the highest budget movie from year 2009-2013:**

**QUERY:**MATCH (d:Director)-[:DIRECTED]->(m:Movie)

WHERE m.title\_year >= 2009 AND m.title\_year <= 2013 AND m.budget IS NOT NULL

WITH d, m

ORDER BY m.budget DESC

RETURN d.name AS director\_name, m.title AS movie\_title, m.budget AS movie\_budget

LIMIT 5

Text

Description automatically generated

**7.The top 5 directors who have directed the most number of movies and their imdb ratings;**

**QUERY:**

MATCH (d:Director)-[:DIRECTED]->(m:Movie)

WITH d, count(m) AS num\_movies, avg(m.imdb\_score) AS avg\_score

ORDER BY num\_movies DESC

LIMIT 5

RETURN d.name, num\_movies, avg\_score

A picture containing text

Description automatically generated

**CONTAINERIZATION**

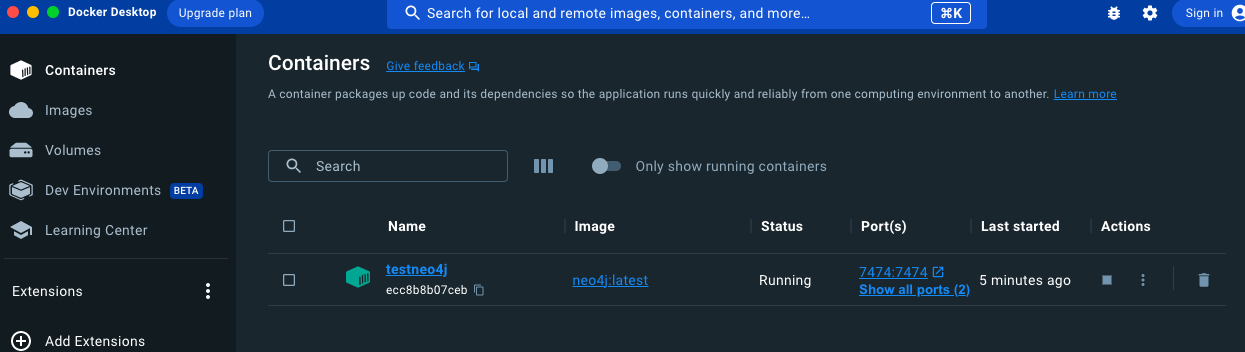
**Running the database in a container:**

This command creates a new container named "testneo4j" with Neo4j database running inside it. The container is configured to use the CSV file located at "/Users/poojamanjunatha/Desktop/imdb.csv" as the data source for Neo4j's import tool.

**Text

Description automatically generated**

**The container is visible:**

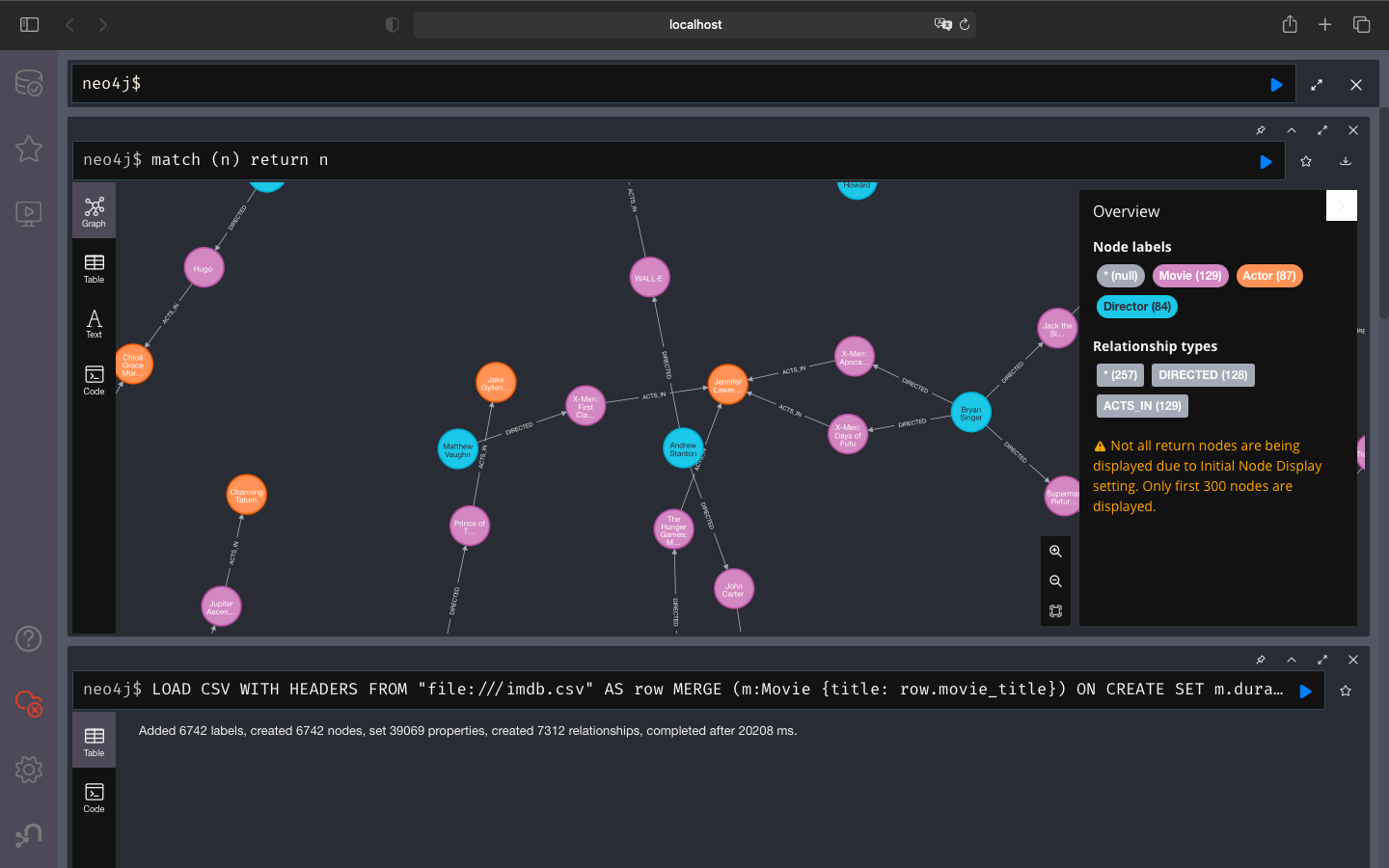
****

**Database creation:**

**Graphical user interface, text

Description automatically generated**

**All the nodes and relationships are created:**



**ORCHESTRATION:**

**Docker Swarm:**

**Getting the IP address of the container created above**

docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps -q --filter "name=neo4j")

****

**Creating a swarm node:**

docker swarm init --advertise-addr 172.17.0.2

**Text

Description automatically generated**

**Creating a .yml file:**

Text

Description automatically generated

**Deployment:**

This command deploys a new stack called my-stack using the my-docker-compose.yml file as the configuration for the services. It creates a new network named my-stack\_default and a new service called my-stack\_database based on the configuration in the compose file. The my-stack\_database service will use the neo4j image and will be deployed as a single replica on the manager node.

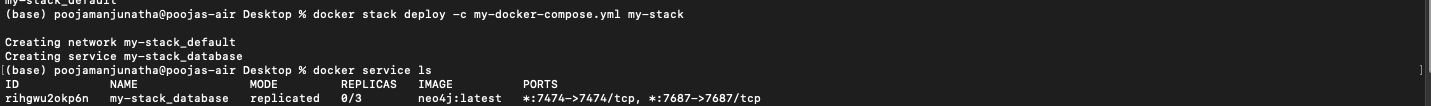


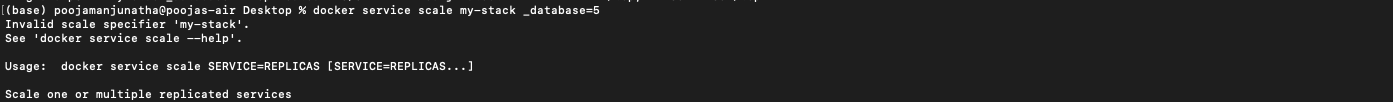
**This is a Docker Compose file that defines a service called "neo4j". The service is based on the "neo4j:latest" Docker image**

Text

Description automatically generated

**This command shows the number of replicas:**





**Accessing the Neo4j web interface running in a container by opening a web browser and navigate to** [**http://192.168.86.28:7474**](http://192.168.86.28:7474)

A screenshot of a computer

Description automatically generated with medium confidence