**Lab Task of Graph Database Neo4j**

1. **Access Neo4j SandBox by the following URI:**

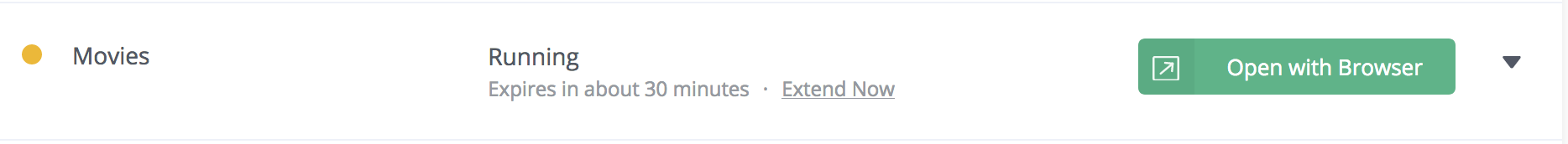
https://sandbox.neo4j.com/login

1. **Launch the free Sandbox:**

Graphical user interface, website

Description automatically generated

1. **Select Movies database:**



1. **Nodes and Relationships in Neo4j**

* **Query:**

**Match (m:Movie) where m.released > 2000 RETURN m limit 5**

**Expected Result**: The above query will return all the movies that were released after the year 2000 limiting the result to 5 items.

Chart, bubble chart

Description automatically generated

* **Query:**

**MATCH (p:Person)-[d:DIRECTED]-(m:Movie) where m.released > 2010 RETURN p,d,m**

**Expected Result**: The above query will return all Person nodes who directed a movie that was released after 2010.

Chart, bubble chart

Description automatically generated

1. **Labels**

* **Query:**

**MATCH (p:Person) RETURN p limit 20**

**Expected Result**: it will return only Person Nodes (limiting to 20 items) while

Chart, bubble chart

Description automatically generated

* **Query:**

**MATCH (n) RETURN n limit 20**

**Expected Result**: it will return all kinds of nodes (limiting to 20 items) while

Chart, bubble chart

Description automatically generated

1. **Properties**

* **Query:**

**MATCH (m:Movie) return m.title, m.released**

**Expected Result** - This will return Movie nodes but with only the title and released properties.

Graphical user interface, application

Description automatically generated

1. **Create a Node**

* **Query:**

**Create (p:Person {name: 'YourName'}) RETURN p**

**Expected Result** - The above statement will create a new Person node with property name having value YourName.

1. **Finding Nodes with Match and Where Clause**

* **Query:**

**Match (p:Person {name: 'Tom Hanks'}) RETURN p**

**Expected Result** - The above statement will return a Person node with property name having value Tom Hanks.

Diagram

Description automatically generated

1. **Merge Clause**

The Merge clause is used to either

1. match the existing nodes and bind them or
2. create new node(s) and bind them

It is a combination of Match and Create and additionally allows to specify additional actions if the data was matched or created.

* **Query:**

**MERGE (p:Person {name: 'John Doe'})**

**ON MATCH SET p.lastLoggedInAt = timestamp()**

**ON CREATE SET p.createdAt = timestamp()**

**Return p**

**Expected Result** - The above statement will create the Person node if it does not exist. If the node already exists, then it will set the property lastLoggedInAt to the current timestamp. If node did not exist and was newly created instead, then it will set the createdAt property to the current timestamp.

Graphical user interface, text, application

Description automatically generated

1. **Create a Relationship**

* **Query:**

**MATCH (p:Person), (m:Movie)**

**WHERE p.name = "Tom Hanks" and m.title = "Cloud Atlas"**

**CREATE (p)-[w:WATCHED]->(m)**

**RETURN type(w)**

**Expected Result** - The above statement will create a relationship :WATCHED between the existing Person and Movie nodes and return the type of relationship (i.e WATCHED).

1. **Relationship Types**

In Neo4j, there can be 2 kinds of relationships - **incoming** and **outgoing**.

To denote an outgoing or an incoming relationship in cypher, we use → or ←.

**MATCH (p:Person)-[r:ACTED\_IN]->(m:Movie) RETURN p,r,m**

**or**

**MATCH (p:Person)-[r:ACTED\_IN]-(m:Movie) RETURN p,r,m**

Diagram

Description automatically generated

1. **Advcanced Cypher queries**

* **Finding who directed “Cloud Atlas” movie**

**MATCH (m:Movie {title: 'Cloud Atlas'})<-[d:DIRECTED]-(p:Person) return p.name**

* **Finding all people who have co-acted with Tom Hanks in any movie**

**MATCH (tom:Person {name: "Tom Hanks"})-[:ACTED\_IN]->(:Movie)<-[:ACTED\_IN]-(p:Person) return p.name**

* **Finding all people related to the movie Cloud Atlas in any way**

**MATCH (p:Person)-[relatedTo]-(m:Movie {title: "Cloud Atlas"}) return p.name, type(relatedTo)**

* **Finding Movies and Actors that are 3 hops away from Kevin Bacon.**

**MATCH (p:Person {name: 'Kevin Bacon'})-[\*1..3]-(hollywood) return DISTINCT p, hollywood**

in the above query, hollywood refers to any node in the database (in this case Person and Movie nodes)

A picture containing chart

Description automatically generated