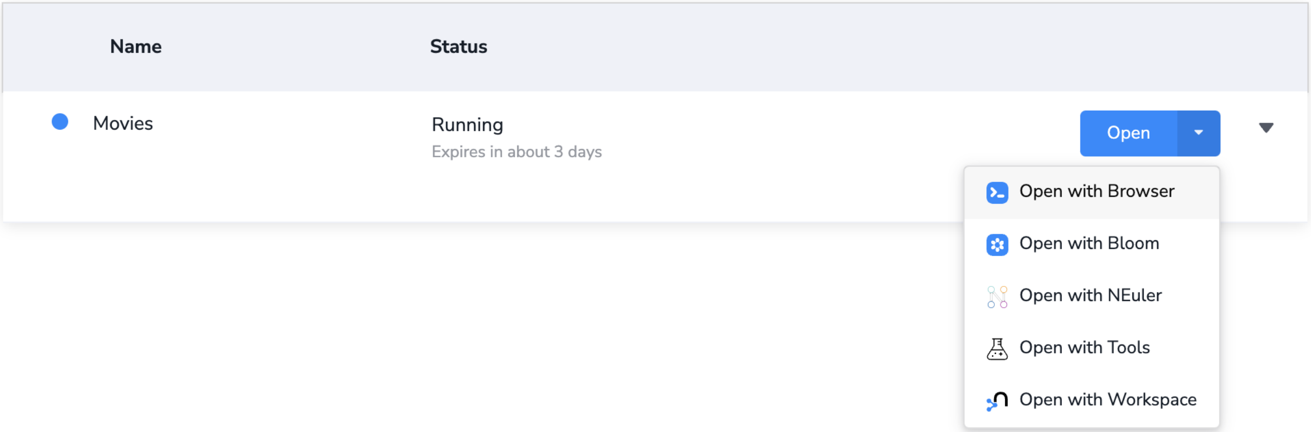
Exercises on Graph Databases

Mehwish Alam

2023

1 Connecting to Neo4j

For this exercise you will be using Neo4j which is a Graph Database Management System built by Neo4j Inc. Neo4j Sandbox - that will be used in today's exercises - is a free, cloud-based instance of Neo4j, where you can learn about Neo4j, test your ideas, or play around with the pre-built data examples. For accessing the Neo4j Sandbox visit this link: `https://neo4j.com/sandbox/`. Click on "Launch the Free Sandbox", setup your account, "Open with Browser" as shown in Figure 1 and choose Movie Graph as shown in Figure 2. After selecting the Movie Graph you will see several queries from 1-8 (see Figure 3). Read and run the queries to understand the results of the queries.
Note: The queries in "Data Profiling" can be used for getting information related to the statistics of the graph.



Name	Status
● Movies	Running Expires in about 3 days

Figure 1: Launching the Sandbox with the Example of Movie Graph

2 Creating a Shakespeare Graph

Now we will create our own graph database based on Figure 4 by clicking on the first database icon on the left in Figure 2. The query for creating this graph is given in the text file accompanied with these exercises.

The figure shows a graph representation of the value chain surrounding the production and consumption of Shakespearean literature. It contains high-quality information about Shakespeare and some of his plays, together with the details of one of the companies that has recently performed the plays, plus a theatrical venue, and some geospatial data and a review. Overall, the graph describes and connects three different domains with differently

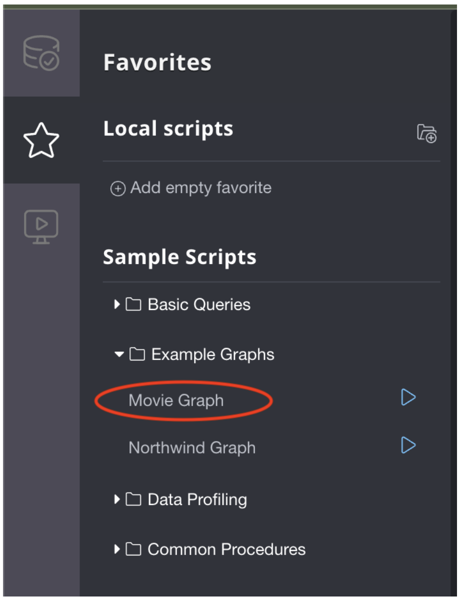
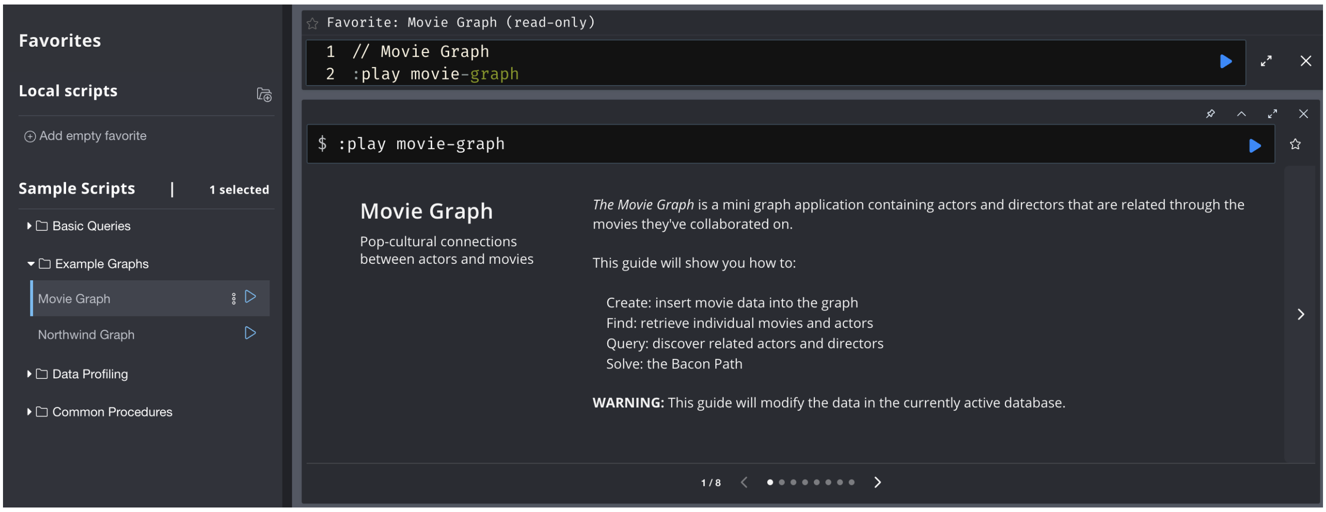


Figure 2: Choose the pre-built Movie Graph



```
// Movie Graph
:play movie-graph
```

```
$ :play movie-graph
```

Movie Graph

Pop-cultural connections between actors and movies

The Movie Graph is a mini graph application containing actors and directors that are related through the movies they've collaborated on.

This guide will show you how to:

Create: insert movie data into the graph
Find: retrieve individual movies and actors
Query: discover related actors and directors
Solve: the Bacon Path

WARNING: This guide will modify the data in the currently active database.

Figure 3: Queries on Movie Graph

formatted relationships: dotted for the literary domain, solid for the theatrical domain, and dashed for the geospatial domain.

Looking first at the literary domain, there is a node that represents Shakespeare himself, with a label Author and properties firstname:'William' and lastname:'Shakespeare'. This node is connected to a pair of nodes, each of which is labeled Play, representing the plays Julius Caesar (title:'Julius Caesar') and The Tempest (title:'The Tempest'), via relationships named WROTE_PLAY. Reading this subgraph left-to-right, following the direction of the relationship arrows, tells us that the author William Shakespeare wrote the plays Julius Caesar and The Tempest. Each WROTE_PLAY relationship has a date property, which tells us that Julius Caesar was written in 1599 and The Tempest in 1610. It's a trivial matter to see how the rest of the Shakespeare's works — the plays and the poems — can be represented in the graph simply by adding more nodes to represent each work, and joining them to the Shakespeare node via WROTE_PLAY and WROTE_POEM relationships.

Turning next to the theatrical domain, some information about the Royal Shakespeare Company was added (often known simply as the RSC) in the form of a node with the label Company and a property key name whose value is RSC. The theatrical domain is connected to the literary domain. In this graph, this is reflected by the fact that the RSC has PRODUCED versions of Julius Caesar and The Tempest. In turn, these theatrical productions are connected to the plays in the literary domain, using PRODUCTION_OF relationships. The graph also captures details of specific performances.

The graph also allows to capture reviews of specific performances. This sample graph includes just one review, for the July 29 performance, written by the user Billy. It can be seen in the interplay of the performance, rating, and user nodes. In this case we have a node labeled User representing Billy (with property name:'Billy') whose outgoing WROTE_REVIEW relationship connects to a node representing his review. The Review node contains a numeric rating property and a freetext review property. The review is linked to a specific Performance through an outgoing REVIEW_OF relationship. To scale this up to many users, many reviews, and many performances, more nodes can simply be added with the appropriate labels and more identically named relationships to the graph.

The third domain, that of geospatial data, comprises a simple hierarchical tree of places. This geospatial domain is connected to the other two domains at several points in the graph. The City of Stratford upon Avon (with property name:'Strat ford upon Avon') is connected to the literary domain as a result of its being Shakespeare's birthplace (Shakespeare was BORN_IN Stratford). It is connected to the theatrical domain insofar as it is home to the RSC (the RSC is BASED_IN Stratford). To learn more about Stratford upon Avon's geography, we can follow its outgoing COUNTRY relationship to discover it is in the Country named England. Looking at the labels on the nodes to which the Theatre Royal is connected, for example, it can be seen that it is located on Grey Street, which is in the City of Newcastle, which is in the County of Tyne and Wear, which ultimately is in the Country of England—just like Stratford upon Avon.

Exercises: Write the queries for the following:

1. Finds all the Shakespeare performances at Newcastle's Theatre Royal.
2. Finds all the Shakespeare performances at Newcastle's Theatre Royal after 1608.
3. How many Shakespeare performances were at Newcastle's Theatre Royal?
4. Rank plays by number of performances.
5. Find the plays written by Shakespeare, and order them based on the year in which they were written. (**HINT:** Use WITH and collect())

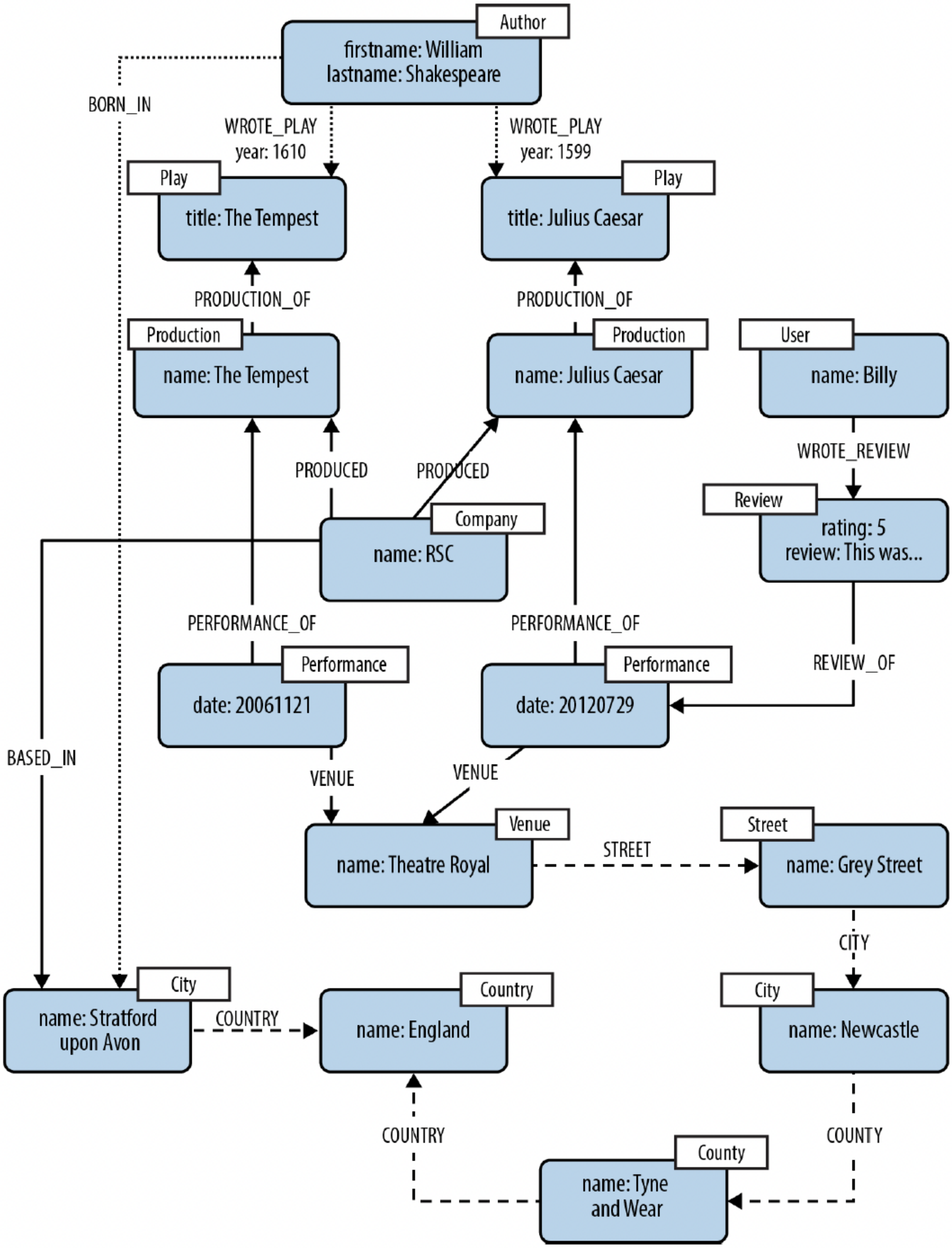


Figure 4: Shakespearean Literature