

Star-Wars Hands-On Workshop

Lab Exercises

Mark Quinsland
Senior Field Engineer
22-September 2022



Notes on Hands-On Exercises:

The full text of the exercises and the data files are all available from the Github repo:

https://github.com/mquinz/star_wars_demo

The exercises use the pre-release Data Importer tool that is currently available only on Neo4j Aura. It is assumed that you have a login & password for Neo4j Aura.

If you are unable to create an Aura account, please let me know and I will set you up with 'Plan B'

We will also be using the browser-based Neo4j modeling tool called Arrows and is available at <https://arrows.app/>

The Arrows app is a simple tool takes only a few minutes to learn but is invaluable for designing Neo4j data models.

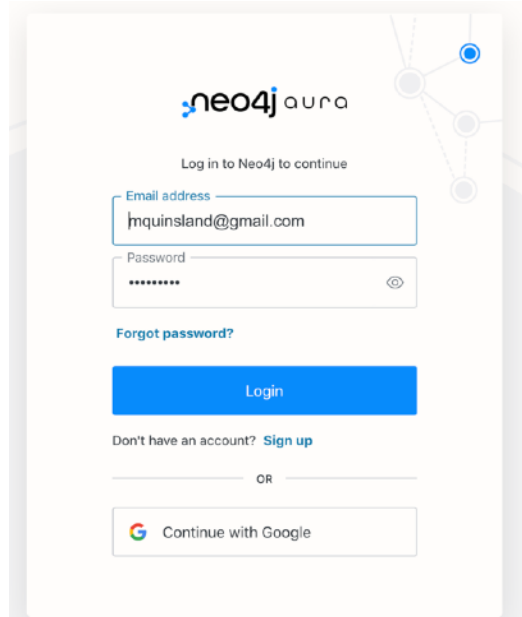
Lab 0 - Connecting to Aura



Lab 0 - Creating a New Database in Aura

Login - or create a free account

- Proceed to <https://login.neo4j.com>
- Enter username and password
- Create a new account if necessary

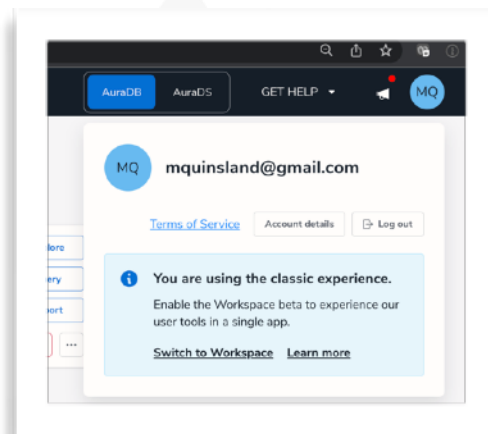


The screenshot shows the Neo4j Aura login interface. At the top is the Neo4j Aura logo. Below it is the text "Log in to Neo4j to continue". There are two input fields: "Email address" with the value "mquinsland@gmail.com" and "Password" with masked characters "*****". To the right of the password field is an eye icon. Below the password field is a link "Forgot password?". A blue "Login" button is centered below the links. Below the button is the text "Don't have an account? Sign up". A horizontal line with "OR" in the center separates the login section from the Google sign-in section. The Google sign-in section features the Google logo and the text "Continue with Google".

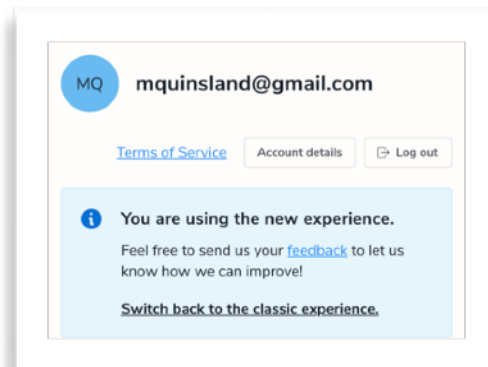
Lab 0 - Creating a New Database in Aura

Switch to New Workspace Experience

- After logging in, click on your initials in the menu bar
- You will see one of these messages to indicate whether you are in the classic version, or the new Workspace version.
- We will be using the new Workspace version. (it's in beta testing so things may get rough!)



Or



Lab 0 - Creating a New Database in Aura

Create a new Instance

- Click on the New Instance Button
- Select the Movies instance
- Your new instance will be named Instance01 and a password will be generated. Copy the password to the clipboard and download the password file
- Please remember where you downloaded the password file!

The image shows two screenshots from the Neo4j Aura interface. The top screenshot is the 'Instances' page, which has a 'New Instance' button. The bottom screenshot is the 'Credentials for Instance01' dialog. It displays the username 'neo4j' and a generated password 'GmLkylkCwpmDaOnlcC1u1EV1Y2hDMxx3wr'. There is a 'Download' button next to the password. A message states: 'We strongly advise changing this initial password.' At the bottom, there is a checkbox that is checked, with the text: 'I confirm I have have copied or downloaded the above credentials, as this password will not be available after this point'. A 'Continue' button is at the bottom right.

Instances [New Instance](#)

Credentials for Instance01

Username: neo4j

Generated password

GmLkylkCwpmDaOnlcC1u1EV1Y2hDMxx3wr [Download](#)

We strongly advise changing this initial password.

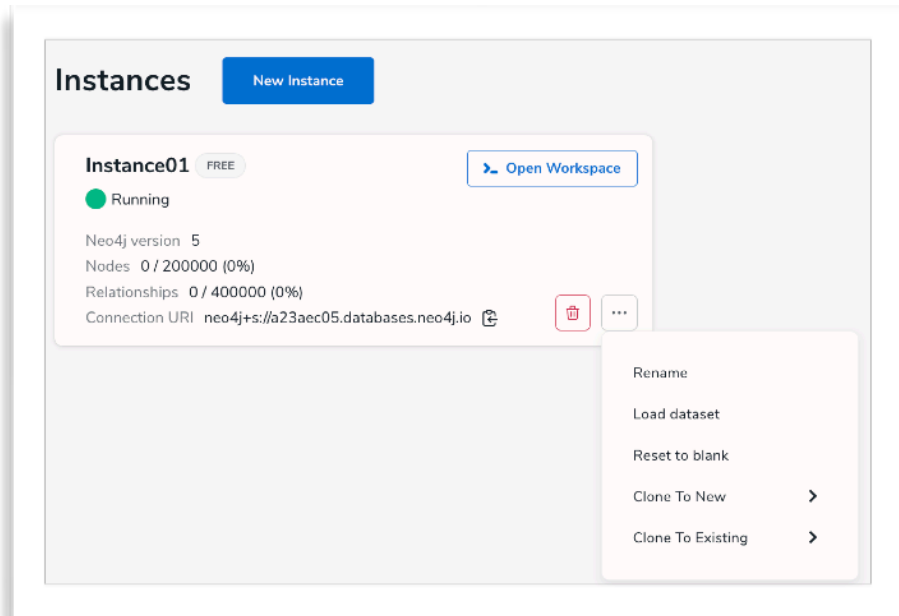
☒ I confirm I have have copied or downloaded the above credentials, as this password will not be available after this point

[Continue](#)

Lab 0 - Creating a New Database in Aura

Rename the new Instance

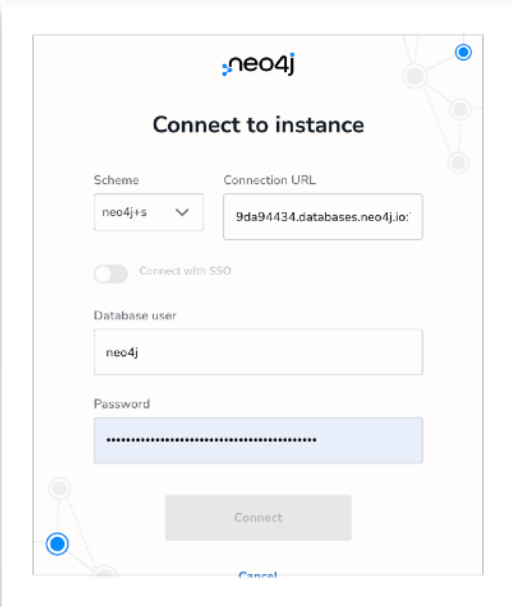
- In a moment or two, the instance will be created and the status will change to 'Running'
- Change the name to something useful by clicking on the ... button next to the trash can.
- Some basic statistics will be shown
- The instance will be free to use for its lifetime. After periods of inactivity it will be suspended, but resuming it requires only a simple click.



Lab 0 - Creating a New Database in Aura

Testing the Connection

- Click on the Open Workspace button to display the Connect to Instance dialog.
- Paste your password into the appropriate field.
- If you forgot your password and failed to download it, you must delete the instance and start over.

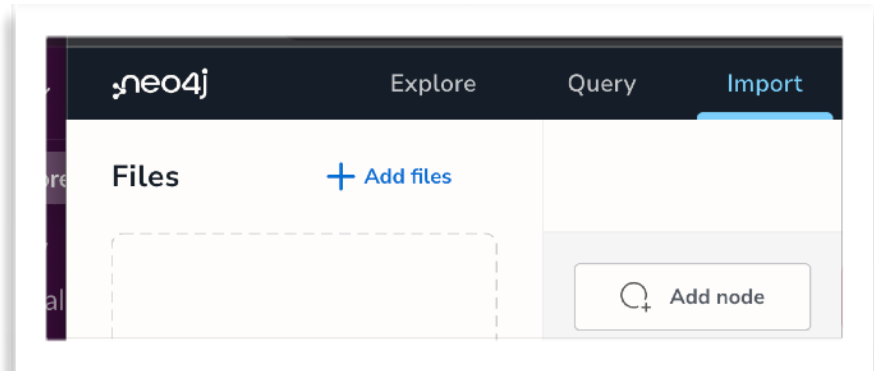


The screenshot shows the 'Connect to instance' dialog box in the Neo4j Aura console. The dialog has a light gray background and a white border. At the top, the Neo4j logo is on the left, and a decorative network diagram is on the right. The title 'Connect to instance' is centered. Below the title, there are two input fields: 'Scheme' with a dropdown menu showing 'neo4j+s' and 'Connection URL' with a text box containing '9da94434.databases.neo4j.io:'. Below these is a toggle switch for 'Connect with SSO'. Further down is a 'Database user' text box with 'neo4j' entered. Below that is a 'Password' text box with a masked password (dots). At the bottom center is a 'Connect' button. The bottom left corner has a small network diagram, and the bottom right corner has the 'Canal' logo.

Lab 0 - Creating a New Database in Aura

Viewing the Workspace

- The Aura Developer Workspace has 3 main tabs.
- Explore: Uses Neo4j Bloom to analyze and visualize the graph without writing code
- Query: Write and execute Cypher queries to create, read, update, and delete data.
- Import: Use the Neo4j Import tool to load CSV files without writing code
- We will start on the Query tab



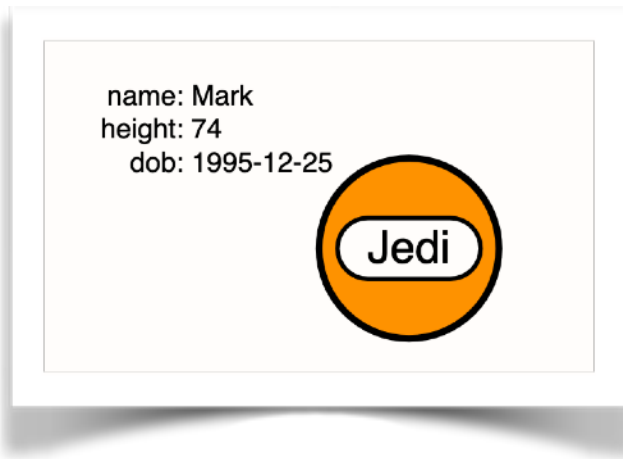
Lab 1 - Data Modeling with Arrows



Lab 1 - Data Modeling using Arrows

Creating A New Node

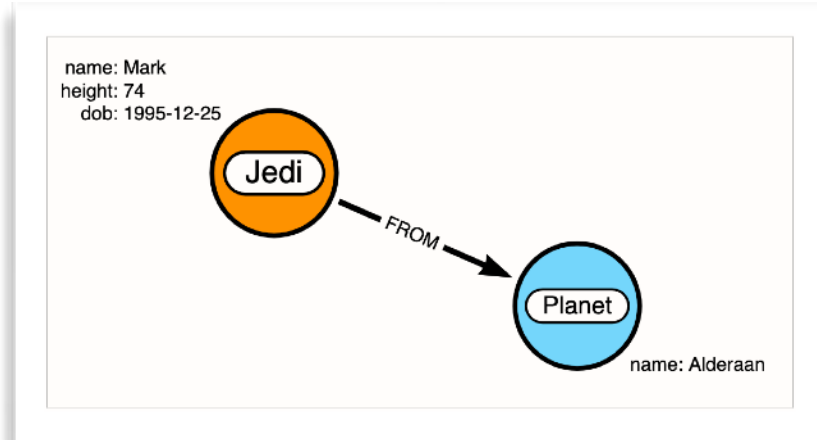
- Bring up <https://arrows.app> on a browser
- Tap on the empty node to highlight it
- On the properties tab, give the new node a label (not a caption)
- Give the node a property:value pair (name: your name)
- Give the node 2-3 additional properties
- Give the node a color



Lab 1 - Data Modeling using Arrows

Creating A Related Node

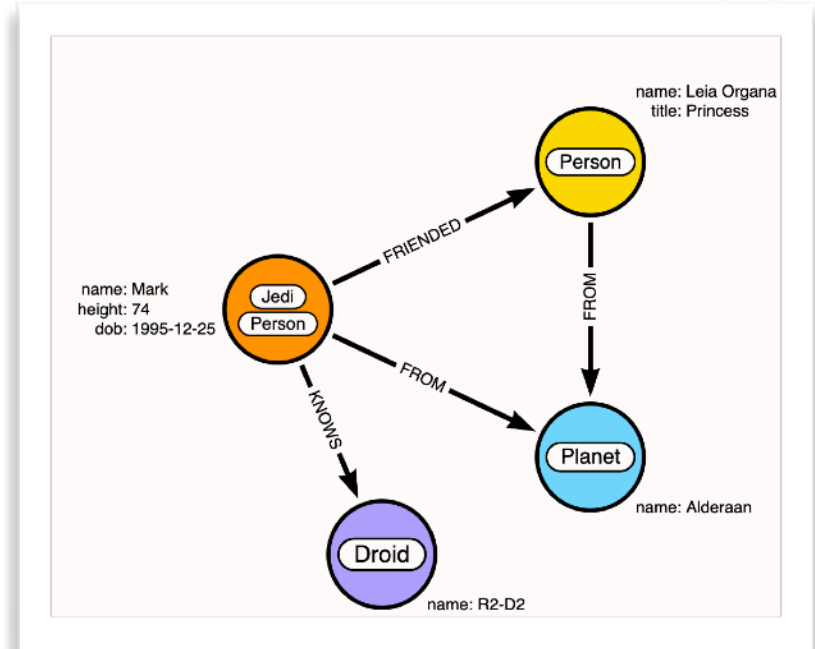
1. Hover over the border of the source node until the border shows that it is selected
2. Click on the selection border and drag it away from the node. You will see a new node appear.
3. Drop the new node away from other nodes
4. On the properties tab, give the new node a label (not a caption)
5. Give the node a property:value pair (name: your name)
6. Select the relationship and give it a type.



Lab 1 - Data Modeling using Arrows

Creating Related Nodes

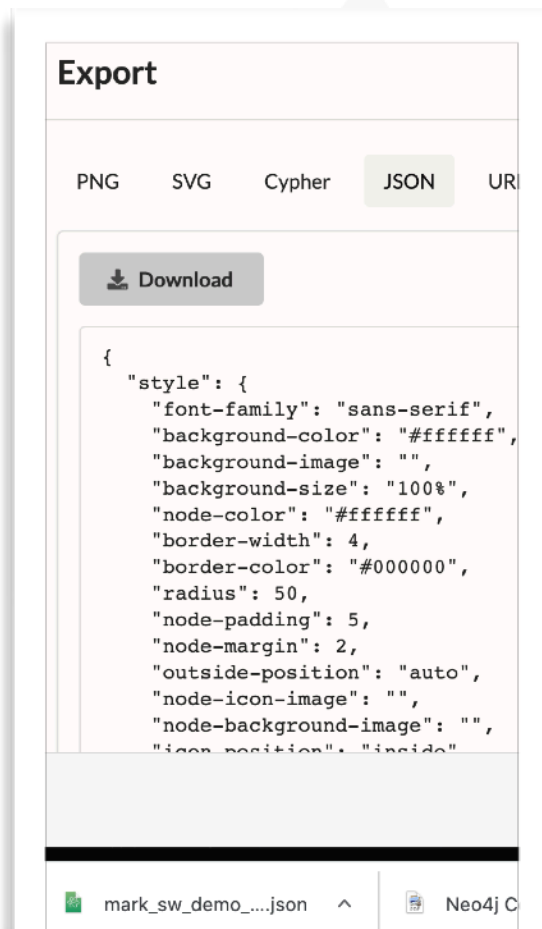
1. Hover over the border of the source node until the border shows that it is selected
2. Click on the selection border and drag it away from the node. You will see a new node appear.
3. Drop the new node away from other nodes
4. On the properties tab, give the new node a label (not a caption)
5. Give the node a property:value pair (name: your name)
6. Select the relationship and give it a type.
7. Repeat for 2-3 more nodes



Lab 1 - Data Modeling using Arrows

Exporting a Model

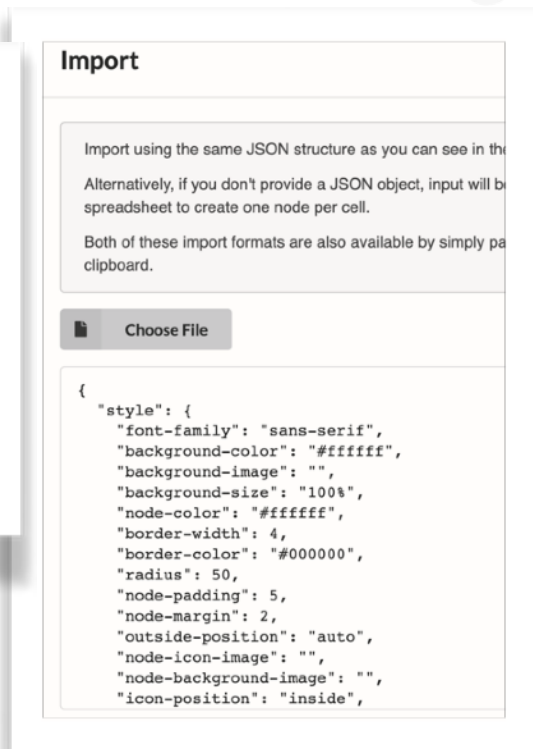
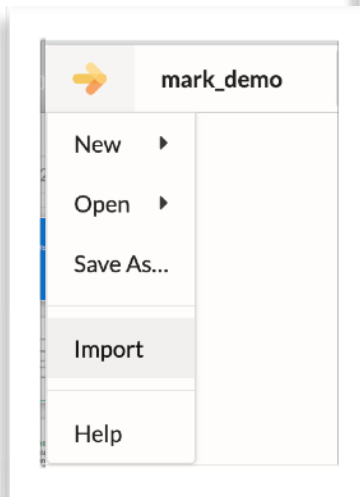
1. Give the model a name
2. Click on the Download/Export button
3. Select the JSON tab
4. Click on the Download button
5. Select a safe place to store the JSON file.



Lab 1 - Data Modeling using Arrows

Importing a Model that was previously exported

1. Click on the yellow arrow at the top left corner
2. Select the Import item
3. Choose the JSON file to be uploaded.
4. View the new nodes / relationships.



Lab 1 - Data Modeling using Arrows

Testing the Model

1. Click on the Download/Export button
2. Select the Cypher tab
3. Select the Create option
4. Review the Cypher code
5. Copy the code to clipboard
6. Paste it into a browser
7. Execute the Cypher to create the nodes
8. View the newly created data in the browser

Export

PNG SVG **Cypher** JSON URL GraphQL

Cypher Clause: ☐ CREATE ☐ MATCH ☒ MERGE

MERGE query behaviour depends on what data is already present in the database. You may need to edit the query to achieve exactly the behaviour you are looking for. Please see [MERGE documentation](#) for guidance.



Copy to clipboard



Run in Neo4j Browser

```
MERGE (n2:Person {name: "Leia Organa", title: "Princess"})<-[:FRIENDED]-(n0:Jedi:Person {name: "Mark", height: 74, dob: "1995-12-25"})-[:FROM]->(:Planet {name: "Alderaan"})<-[:FROM]-(n2)
MERGE (n0)-[:KNOWS]->(:Droid {name: "R2-D2"})
```


Lab 1 - Data Modeling using Arrows

Viewing the Data in Neo4j Browser

1. Copy the Cypher text that was generated by the Export window
2. Switch to your browser and bring up the Aura Developer Workbench tab
3. Paste the code into the execution window
4. Click on the small blue arrow to execute the command

```
1 CREATE (n6:Person {name: "Leia Organa", title:
  "Princess"})←[:FRIENDED]-(n4:Jedi:Person {name:
  "Mark", height: 74, dob: "1995-12-25"})-[:FROM]→
  (:Planet {name: "Alderaan"})←[:FROM]-(n6),
2 (n4)-[:KNOWS]→(:Droid {name: "R2-D2"})
```

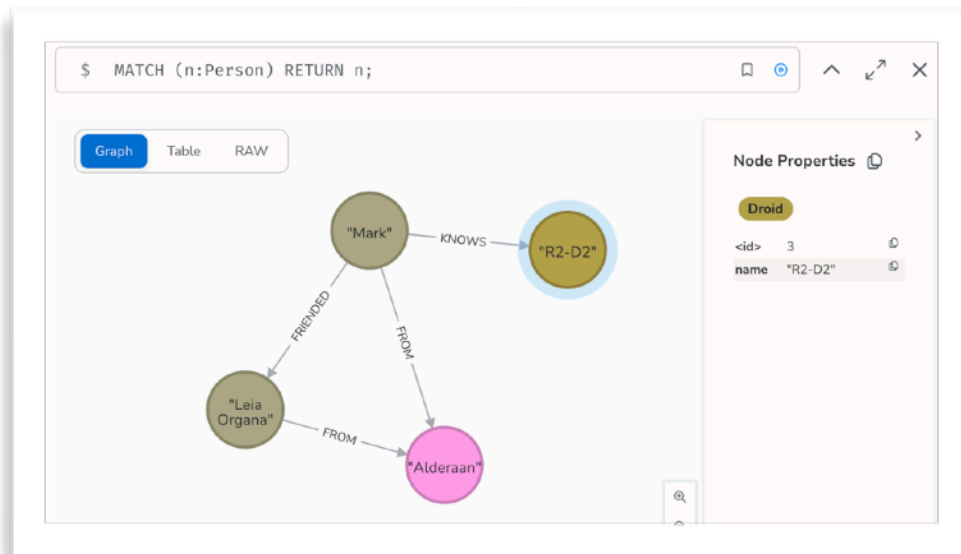


Lab 1 - Data Modeling using Arrows

Viewing the Data in Neo4j Browser

1. Run a simple Cypher command to view your data
2. Double-click on the nodes to see related nodes appear.

```
MATCH (n:Person) RETURN n;
```



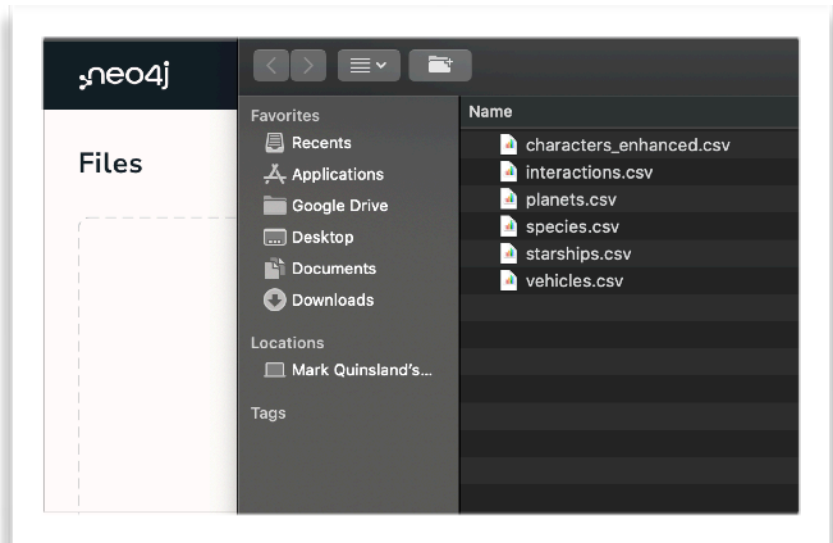
Lab 2 - Using Workspace Importer



Lab 2 - Using Workspace Importer

Selecting CSV files to Import

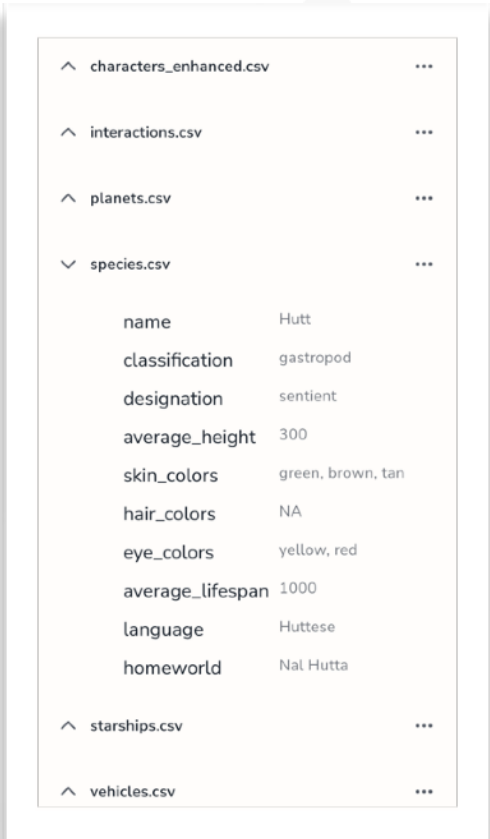
- Download the data files from
- https://github.com/mquinz/star_wars_demo/blob/master/data/star_wars_data.zip
- Unzip the files
- Click on the + Add files link on the Workspace Import tab or drag-and-drop the files onto the files tab.



Lab 2 - Using Workspace Importer

Previewing Data From CSV files

- After adding the 6 CSV files, expand/collapse the file names to preview the data in the files
- The importer will attempt to determine data types for the values, but you can override/correct this in the next step.



The screenshot shows a workspace importer interface with a list of CSV files on the left and a preview of the 'species.csv' file on the right. The files listed are 'characters_enhanced.csv', 'interactions.csv', 'planets.csv', 'species.csv', 'starships.csv', and 'vehicles.csv'. The 'species.csv' file is expanded, showing a table with columns and values.

| species.csv | |
|------------------|-------------------|
| name | Hutt |
| classification | gastropod |
| designation | sentient |
| average_height | 300 |
| skin_colors | green, brown, tan |
| hair_colors | NA |
| eye_colors | yellow, red |
| average_lifespan | 1000 |
| language | Huttese |
| homeworld | Nal Hutta |

Lab 2 - Using Workspace Importer

Adding Nodes

- Click on the Add Node Button and an empty node type will appear.
- In the Mapping Details tab, enter the name of a Label e.g. Character
- Select a CSV file that contains the data to be loaded for that Label
- Click on the 'Select from file' link and choose the property names to be loaded.
- Edit data types or field names
- In the ID field, choose which property value should be used as the primary key of each node

The screenshot shows the Workspace Importer interface. The 'Files' tab is active, showing a list of files. The 'Select from file' dialog is open, showing a list of properties from a CSV file. The 'Mapping Details' tab is also visible, showing the 'Properties' section with a table of mapped properties.

Files + Add files

Select from file ☒ Select all

| | | |
|-------------------------------------|-------------|---------------------|
| <input checked="" type="checkbox"/> | name | Yoda |
| <input checked="" type="checkbox"/> | gender | male |
| <input checked="" type="checkbox"/> | birthYear | 896BBY |
| <input checked="" type="checkbox"/> | hairColor | white |
| <input checked="" type="checkbox"/> | eyecolor | brown |
| <input checked="" type="checkbox"/> | height | 66 |
| <input checked="" type="checkbox"/> | homePlanet | NA |
| <input checked="" type="checkbox"/> | weight | 17 |
| <input checked="" type="checkbox"/> | socialName | YODA |
| <input checked="" type="checkbox"/> | species | Yoda's species |
| <input checked="" type="checkbox"/> | pagerank | 2.6774272604793796 |
| <input checked="" type="checkbox"/> | betweenness | 184.5754407354513 |
| <input checked="" type="checkbox"/> | closeness | 0.47619047619047616 |
| <input checked="" type="checkbox"/> | louvain | 127 |

Mapping Details

Label ⓘ

Character

File ⓘ

characters_enhanced.csv

Properties Mapping

+ Select from file + Add new

| | | | |
|------------|--------|--|--|
| name | string | | |
| gender | string | | |
| birthYear | string | | |
| hairColor | string | | |
| eyecolor | string | | |
| height | string | | |
| homePlanet | string | | |
| weight | string | | |
| socialName | string | | |

ID ⓘ

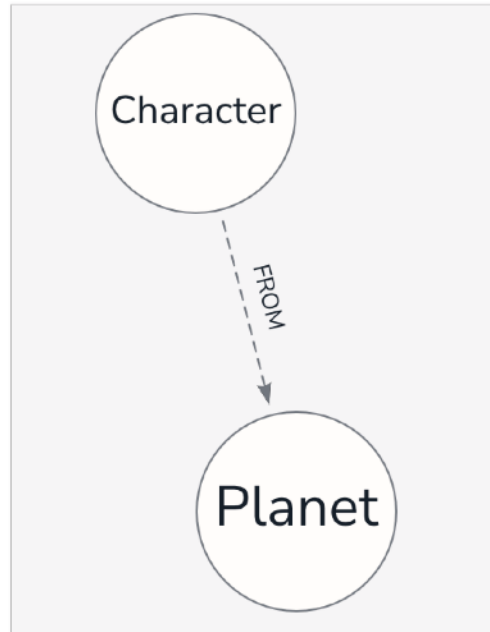
name

Cancel Confirm

Lab 2 - Using Workspace Importer

Creating Relationships Between Nodes

- Select a Node type that will serve as the 'From' node
- Hover over its border until the border color changes and a + indicator appears.
- Drag the + indicator to the node that will be the 'To' node.
- Double-Click on the relationship line to select it and type in the relationship type
- The line will appear broken until the relationship info is completed




Lab 2 - Using Workspace Importer

Creating Relationships Between Nodes

- Complete the remaining fields on the Mapping Details tab.
- Type: The name of the relationship
- File: Which file has the source/target information for the relationship
- From: Which field contains the ID of the From node
- To: Which field contains the ID of the To node
- Properties: Any fields that should be used as relationship properties.

Mapping Details

Type ⓘ 

FROM

File ⓘ

characters_enhanced.csv ▼

From ⓘ Character (name)

name ▼

To ⓘ Planet (name)

homePlanet ▼

Properties Mapping

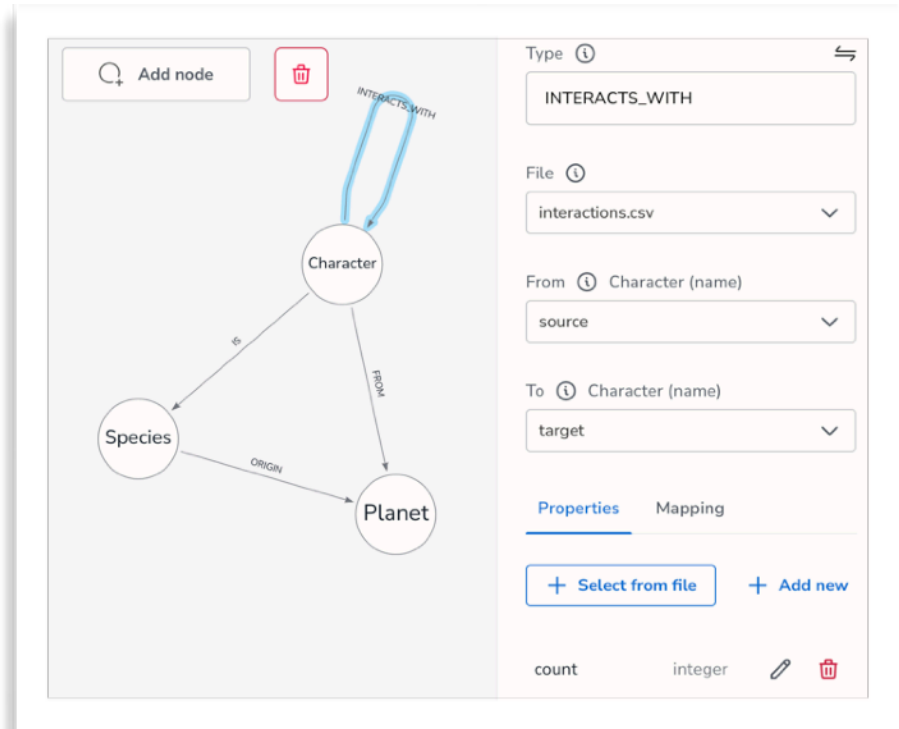
[+ Select from file](#) [+ Add new](#)

No properties defined

Lab 2 - Using Workspace Importer

Creating Self-Join Relationship

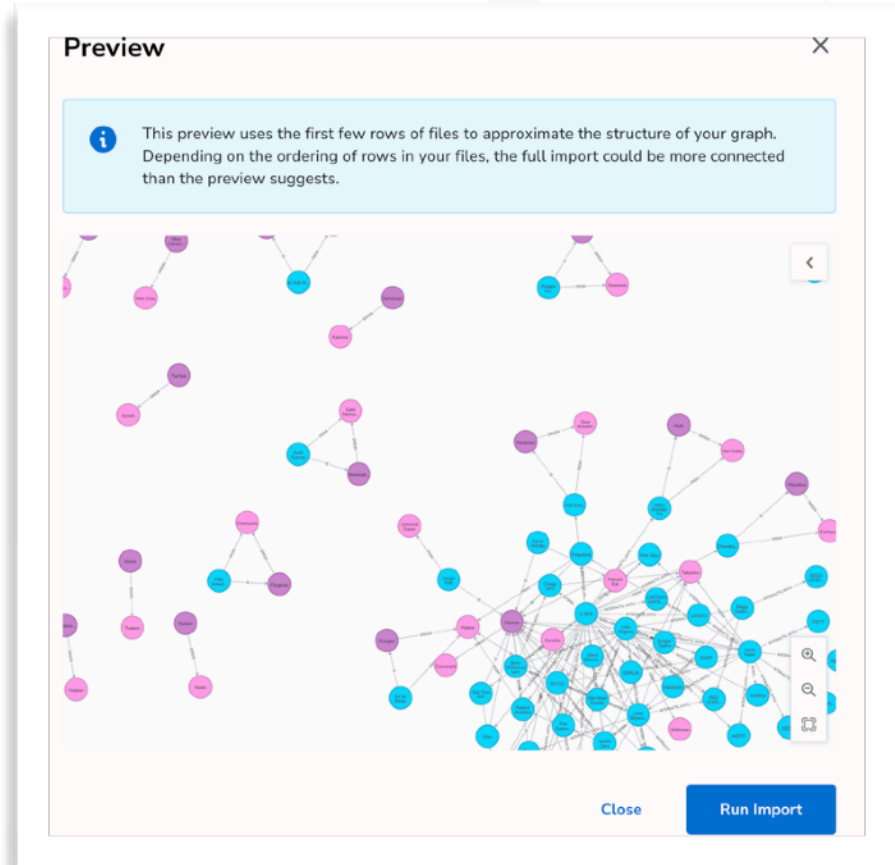
- Select the Character node and hover over its border until the + indicator appears
- Drag the selection + indicator away from the Character node, and then drag it back and drop it on the Character node.
- Fill out the mapping details.
- Type: INTERACTS_WITH
- File: interactions.csv
- From: source
- To: target
- Properties: count



Lab 2 - Using Workspace Importer

Preview the Import

- Click on the Preview button
- A sampling of records from each of the files will be used to give an idea of how the nodes and relationships will look.
- If everything looks okay, click on the Run Import button.



Lab 2 - Using Workspace Importer

Run the Import

- Click on the Run Import button
- A summary of the import will be shown.
- Each Node Label and each Relationship Type will be summarized.
- If all looks good, click on the Start Exploring button.

Import results

✓ Import completed without errors • Total time: 00:00:04

| Species | species.csv | | | | | | Show cypher |
|------------|-------------|-----------|---------------|----------------|--------------|-------------|-----------------------------|
| Time Taken | File Size | File Rows | Nodes Created | Properties Set | Labels Added | Query Count | Query Time |
| 00:00:01 | 3.2 KiB | 37 | 37 | 346 | 37 | 1 | 00:00:00 |

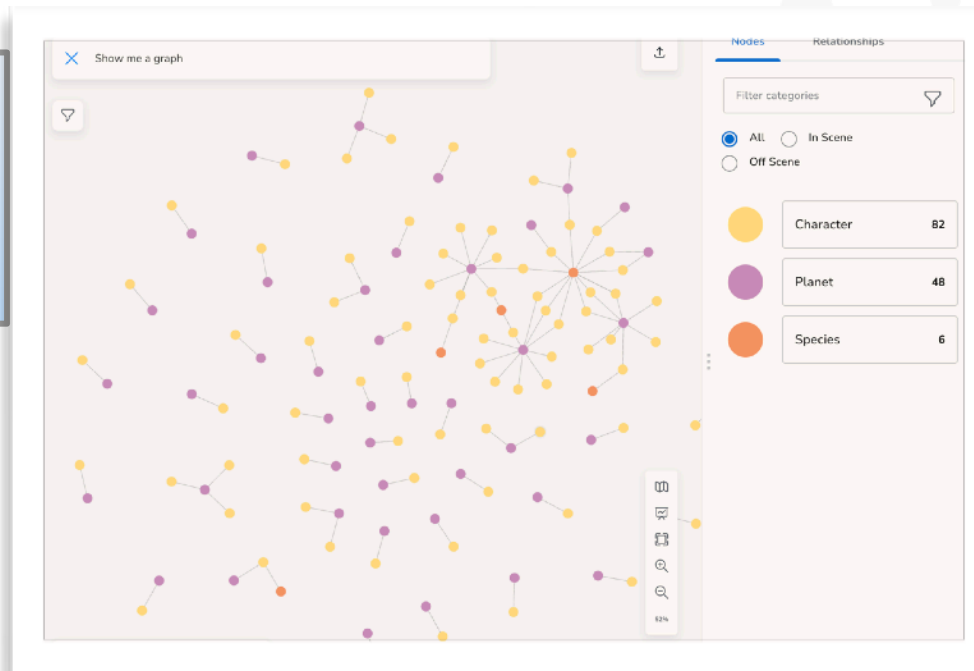
| FROM | characters_enhanced.csv | | | | | | Show query |
|------------|-------------------------|-----------|-----------------------|----------------|-------------|------------|----------------------------|
| Time Taken | File Size | File Rows | Relationships Created | Properties Set | Query Count | Query Time | |
| 00:00:00 | 16.0 KiB | 151 | 82 | 0 | 1 | 00:00:00 | |

[Close](#)[Start Exploring](#)

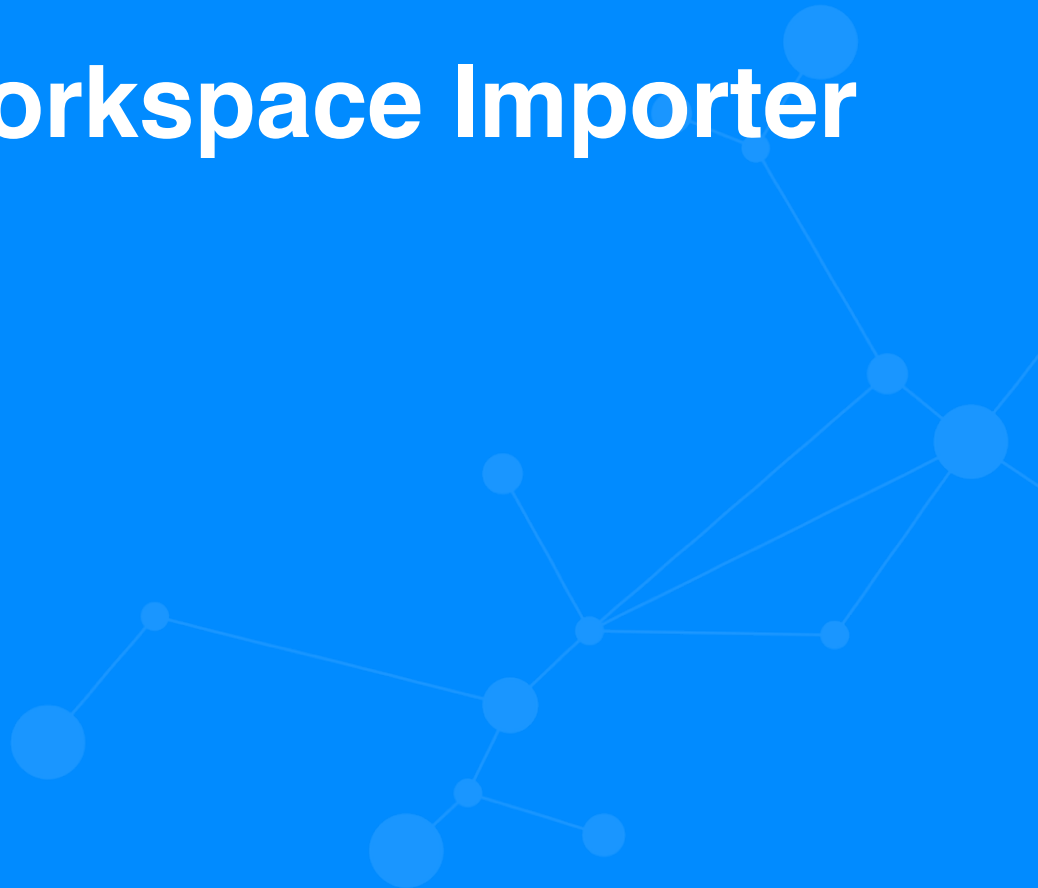
Lab 2 - Using Workspace Importer

Viewing your Handiwork

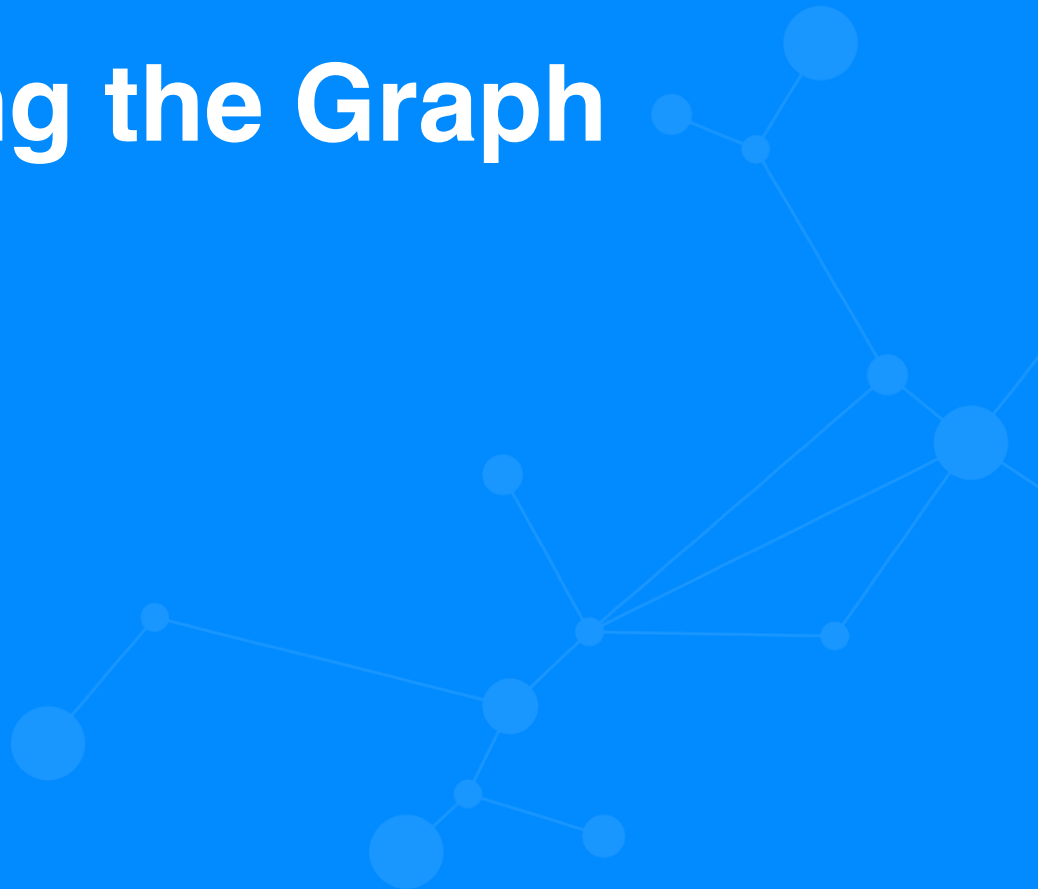
- Click on the Start Exploring button
- This will show the Explore tab which uses Neo4j Bloom to display the newly imported data



Lab 2 - Using Workspace Importer



Lab 3 - Querying the Graph



Lab 3 - Basic Queries - The Match Statement

- Most queries start with a MATCH statement which is used for pattern matching.
- It is generally the most important part of the query
- All nodes/relationships/paths that meet the filtering criteria are selected.
- Selected items are then passed to the next part of the query

```
// Return all characters  
MATCH (c:Character)  
RETURN c
```

```
// Return a specific character  
MATCH (c:Character)  
WHERE c.name = "Han Solo"  
RETURN c
```

Lab 3 - Basic Queries - The Match Statement

- Most queries start with a MATCH statement
- Used for pattern matching - similar in some ways to a WHERE clause
- All nodes/relationships/paths that meet the filtering criteria are selected
- Selected items are then passed to the next part of the query
- It is generally the most important part of the query

```
// Return all Characters from Alderaan
MATCH (c:Character)-[FROM]->(p:Planet)
WHERE p.name = "Alderaan"
RETURN c
```

```
// Return a specific Character
MATCH (c:Character)
WHERE c.name = "Han Solo"
RETURN c
```


Lab 3 - Basic Queries - The Match Statement

Node Properties Used as a Filter

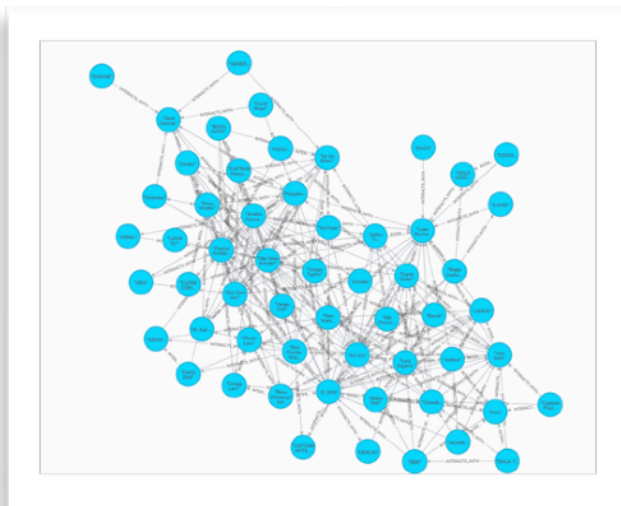
```
// Return all Characters that interacted with R2-D2  
MATCH (c:Character)<-[INTERACTS_WITH]-(other:Character)  
WHERE c.name = "R2-D2"  
RETURN other
```



Lab 3 - Basic Queries - The Match Statement

Unlike SQL, Variable-Length Queries are simple

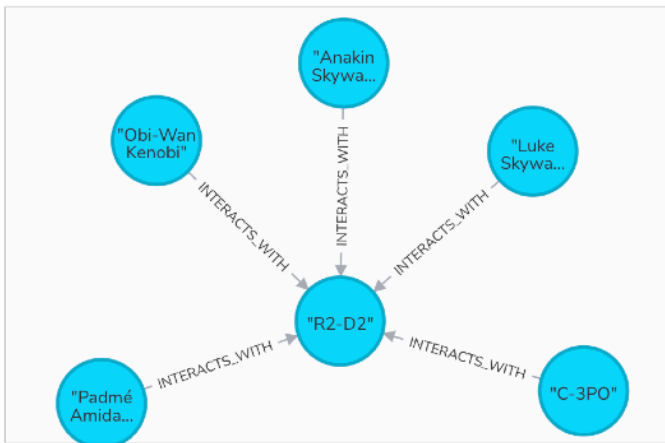
```
// Variable length query - interacts_with 3 levels down  
MATCH p=(c:Character)<-[INTERACTS_WITH*..3]-(other:Character)  
WHERE c.name = "R2-D2"  
RETURN p
```



Lab 3 - Basic Queries - The Match Statement

Relationship Properties Can Also be Used

```
// Who does R2-D2 interact with the most  
MATCH p=(c:Character)-[i:INTERACTS_WITH]-(other:Character)  
WHERE c.name = "R2-D2"  
RETURN p ORDER BY i.count DESC LIMIT 5
```



Lab 3 - Basic Queries - The Match Statement

Aggregation Values Used as a Filter

```
// Which characters interact with the most characters
MATCH (c:Character)-[INTERACTS_WITH]-(other)
WITH c.name as character, count(other) as others
WHERE others > 20
RETURN character, others
ORDER BY others desc LIMIT 5
```

| character | others |
|--------------------------|--------|
| "Anakin Skywalker" | 44 |
| "Obi-Wan Kenobi" | 39 |
| "C-3PO" | 38 |
| "Padmé Amidala" | 36 |
| Showing 1-5 of 5 results | |

Lab 3 - Basic Queries - The Match Statement

String Comparison Used as a Filter

```
// Which characters have Darth as part of their name  
MATCH (c:Character)  
WHERE c.name STARTS WITH "Darth"  
RETURN c
```

"Darth
Vader"

"Darth
Maul"

Lab 4 - Updating the Graph



Lab 4 - Update Queries - Creating New Nodes

Using the CREATE Statement

- New nodes can be created using the CREATE statement
- Unless uniqueness constraints are used, duplicates may result

```
// Create a new Character  
CREATE (c:Character {name:"Mark Q",  
height:74,hair_colors:"Fleshtone"})  
RETURN c
```



Lab 4 - Update Queries - Creating New Nodes

Using the MERGE Statement

- New nodes can also be created using the MERGE statement
- Neo will check first to see if any nodes match before creating a new node

```
// Carefully create a new Character using MERGE
MERGE (c:Character {name:"Mark Q"})
ON CREATE SET c +=
{height:74,hair_colors:"Fleshtone"}
RETURN c
```



Lab 4 - Update Queries - Creating New Relationships

Using the MERGE Statement

- New relationships can also be created using the MERGE statement
- Use the MATCH statement to find the “from” and “to” nodes
- Use the MERGE statement to create a new relationship (if necessary)

MATCH

(r2:Character),

(me:Character)

WHERE r2.name = 'R2-D2' **AND** me.name = 'Mark Q'

MERGE (me)-[r:FOLLOWS]->(r2)

RETURN r2,r,me

