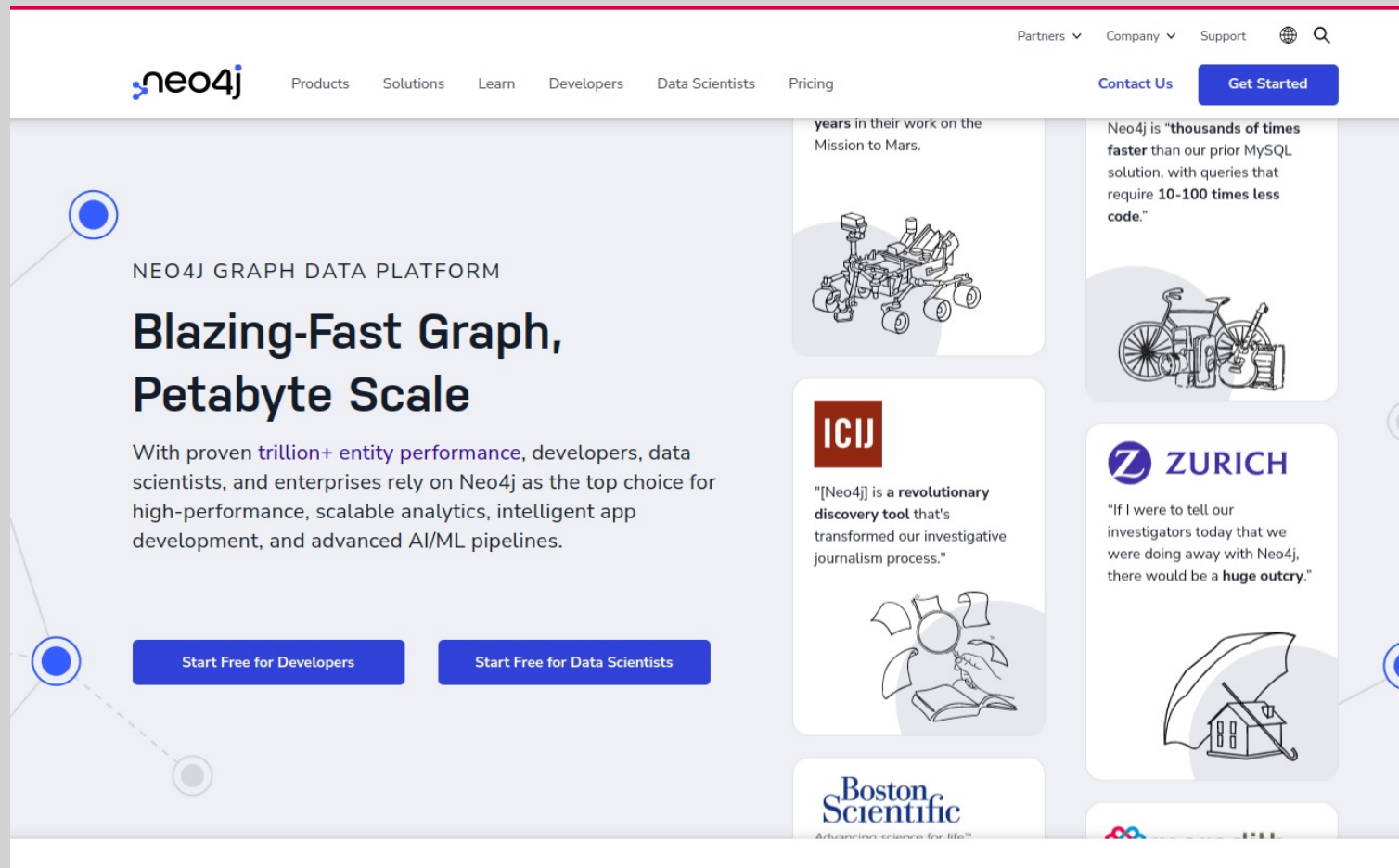


Neo4J

Databases 2023

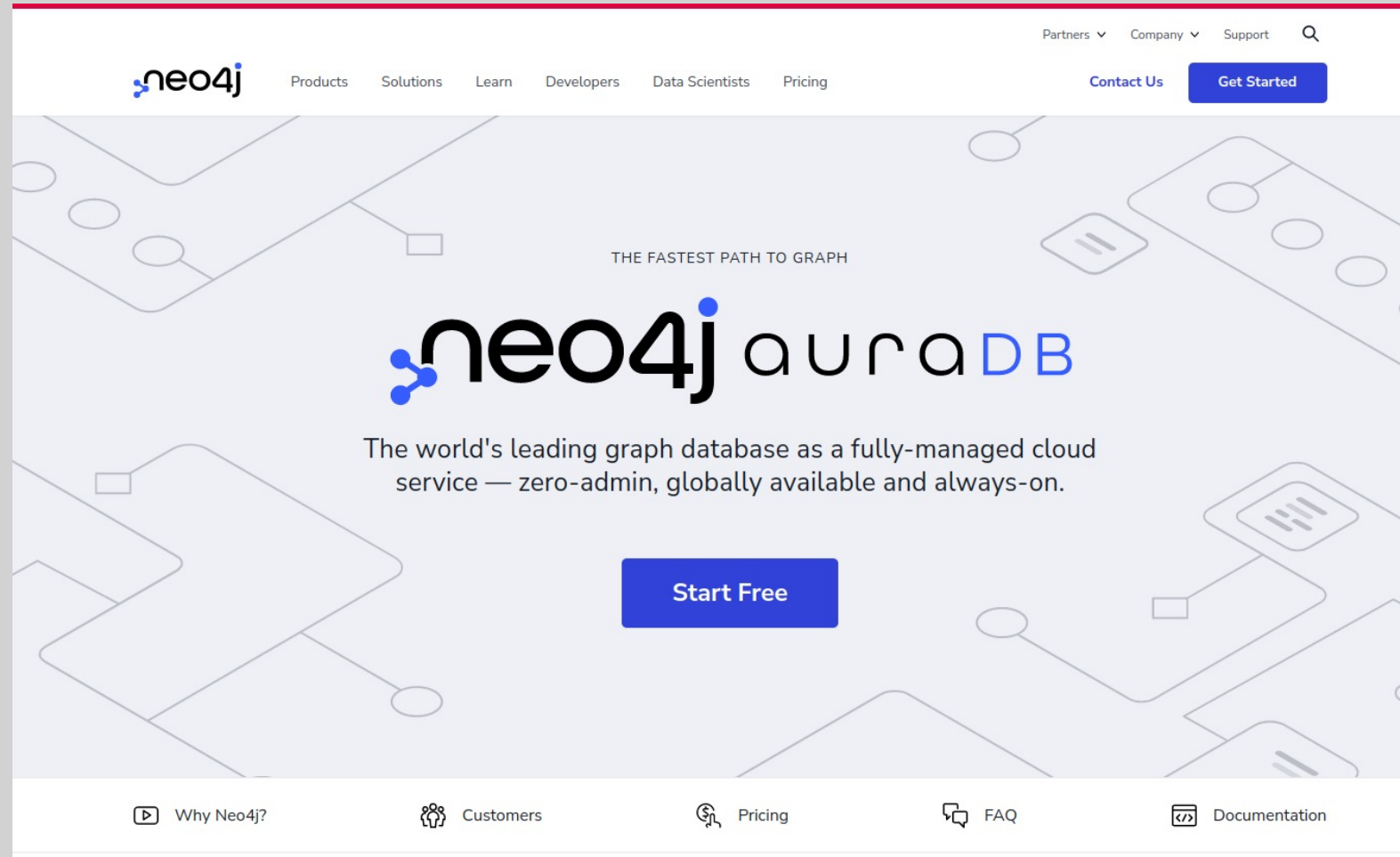
How to Start

- Go to **neo4j.com** and press get started



How to Start

- Then press Start free



How to Start

- Then press Register

Welcome to Neo4j Aura

A fully managed, cloud-native graph data platform.

Neo4j AuraDB and AuraDS make it easy to build fast, scalable, and intelligent applications in the cloud, without managing complex infrastructure. Aura solutions offer:

- ✓ Lightning-fast query performance
- ✓ Fully-managed updates and patches
- ✓ Easy scalability, on-demand
- ✓ Robust security, reliability and ACID compliance
- ✓ Built-in tools to learn, build, and visualize

Start Free with AuraDB and join the largest graph community.
No Credit Card Required.



Create Your Account


Sign Up to Neo4j to continue to Neo4j Cloud Console.

Email address

Continue

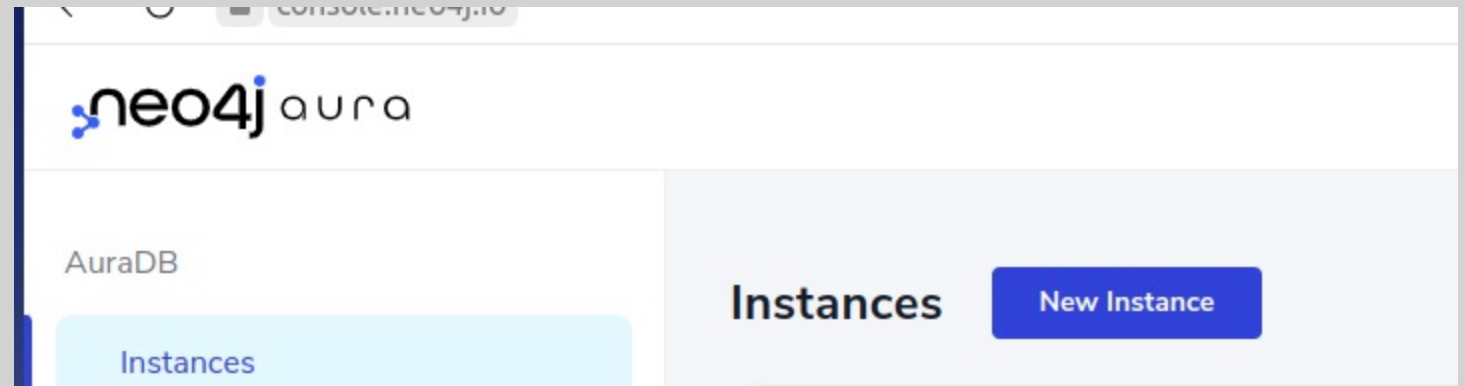
Already have an account? [Log in](#)

OR

 Continue with Google

How to Start







- Then New instance



How to Start

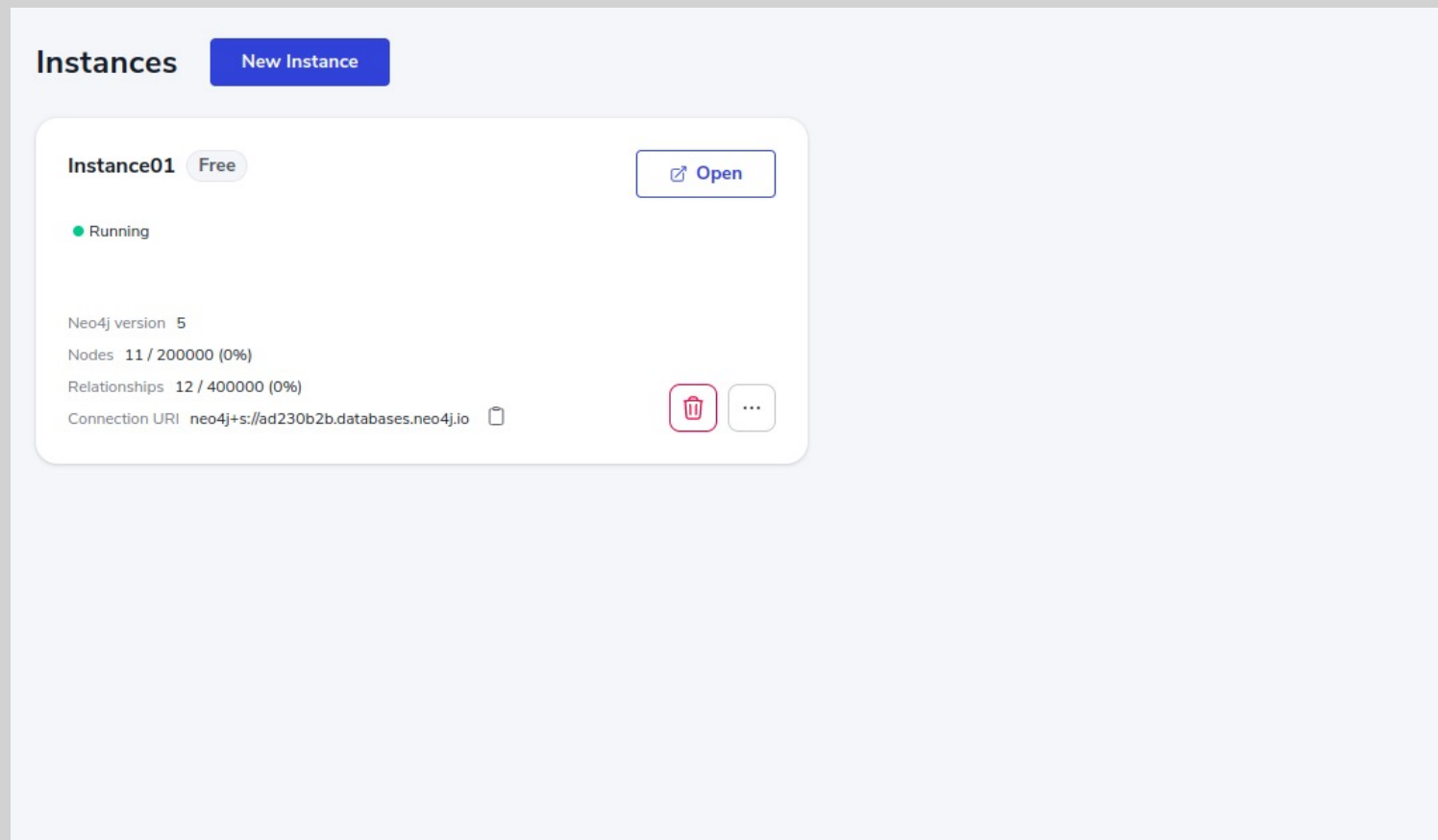
- Select “Start here” instance and store login and password

Get started by picking a dataset or an empty instance

 Start Here An introduction to graph databases and Neo4j for new users BEGINNER Free	 Movies Small example graph of popular movies and actors Free	 Graph based Recommendations Generate personalized real-time recommendations using a dataset of movie reviews Free
 Graphs for Cybersecurity Cybersecurity, Active Directory environment auditing and analysis of possible attack paths using graph Free	 StackOverflow Data StackOverflow Analysis - Questions, Answers, Tags, Comments Free	 Empty instance Load or create your own data in a blank instance Free

How to start

- After the instance is loaded, enter to the instance (press open)



How to Add Data

- Create node

```
CREATE(p:Fighter{name:'Khabib Nurmagomedov', weight:'155'}), (pp:Fighter{name:'Rafael Dos Anjos', weight:'155'})
```

- Create relation to existing nodes

```
MATCH (a:Fighter), (b:Fighter)
```

```
WHERE a.name = 'Khabib Nurmagomedov' AND b.name = 'Rafael Dos Anjos'
```

```
CREATE (a)-[r:beats]→(b)
```

- Create node and relations

```
CREATE(p:Fighter{name:'Khabib Nurmagomedov', weight:'155'}), (pp:Fighter{name:'Rafael Dos Anjos', weight:'155'}), (p)-[:beats]->(pp)
```

- Example to drop nodes

Drop all: `MATCH (n) DETACH DELETE n`

Drop all relations: `MATCH () - [r:beats] -> () DELETE r`

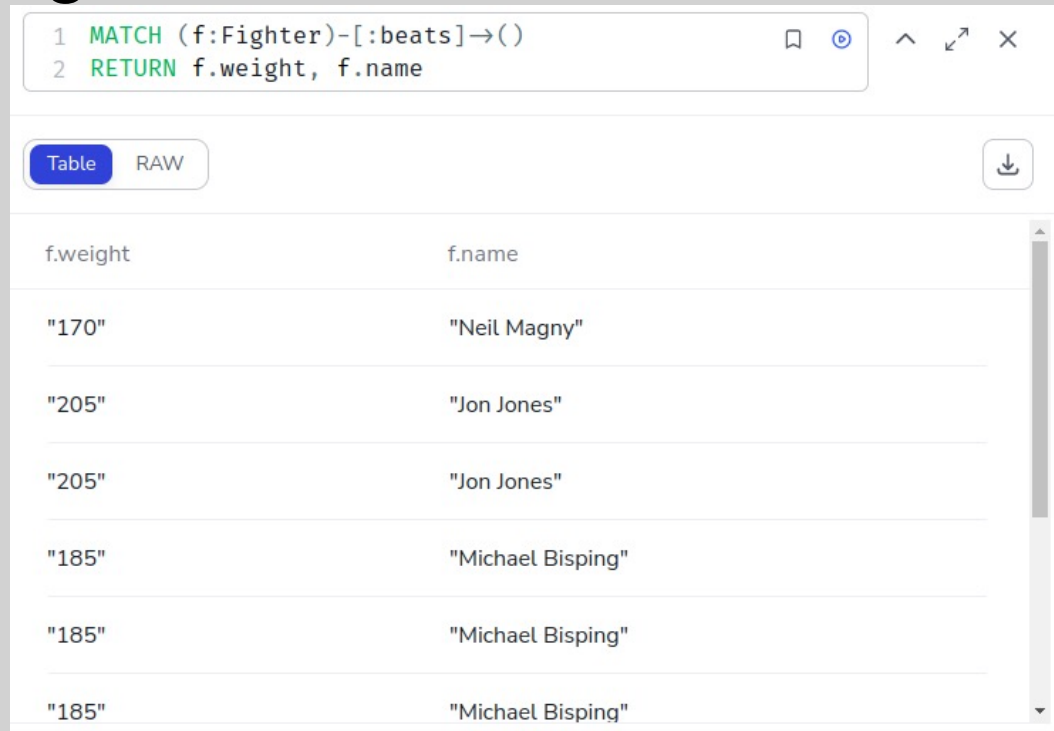
Drop all nodes: `MATCH (n:Fighter) DELETE n`

Select Example

1. Consider the following query: Return all weight fighters grouped by weight who at least have one win.

Select Example

1. Consider the following query: Return all weight fighters grouped by weight who at least have one win.
 - a. list of all fighters who at least have one win > list of winners



The screenshot shows a query editor with two lines of code: `1 MATCH (f:Fighter)-[:beats]→()` and `2 RETURN f.weight, f.name`. Below the editor are tabs for 'Table' (selected) and 'RAW', and a download icon. The results table has two columns: 'f.weight' and 'f.name'. It contains six rows of data, showing fighters grouped by weight.

f.weight	f.name
"170"	"Neil Magny"
"205"	"Jon Jones"
"205"	"Jon Jones"
"185"	"Michael Bisping"
"185"	"Michael Bisping"
"185"	"Michael Bisping"

Select Example

1. Consider the following query: Return all weight fighters grouped by weight who at least have one win.

b. list of winners grouped by weight

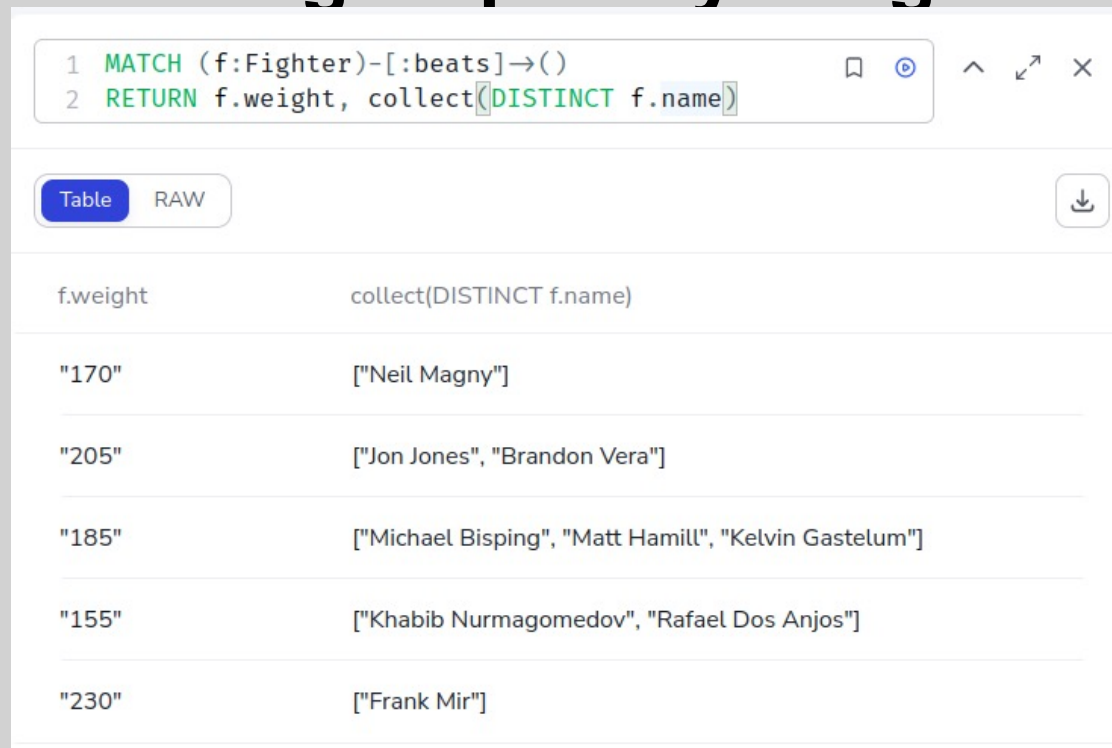
```
1 MATCH (f:Fighter)-[:beats]→()
2 RETURN f.weight, collect(f.name)
```

f.weight	collect(f.name)
"170"	["Neil Magny"]
"205"	["Jon Jones", "Jon Jones", "Brandon Vera"]
"185"	["Michael Bisping", "Michael Bisping", "Michael Bisping", "Matt Hamill", "Kelvin Gastelum"]
"155"	["Khabib Nurmagomedov", "Rafael Dos Anjos"]
"230"	["Frank Mir"]

Select Example

1. Consider the following query: Return all weight fighters grouped by weight who at least have one win.

c. list of winners grouped by weight without duplicates



The screenshot shows a query editor with two lines of code: `1 MATCH (f:Fighter)-[:beats]→()` and `2 RETURN f.weight, collect(DISTINCT f.name)`. Below the editor are tabs for 'Table' (selected) and 'RAW', and a download icon. The results table has two columns: 'f.weight' and 'collect(DISTINCT f.name)'. It contains six rows of data representing different weight classes and the names of fighters who have won in that class.

f.weight	collect(DISTINCT f.name)
"170"	["Neil Magny"]
"205"	["Jon Jones", "Brandon Vera"]
"185"	["Michael Bisping", "Matt Hamill", "Kelvin Gastelum"]
"155"	["Khabib Nurmagomedov", "Rafael Dos Anjos"]
"230"	["Frank Mir"]

Exercise 1

- MMA Math.. If fighter A beats fighter C and C beats B that means A can beat B.
 - Using Neo4J-empty project, build a representation of the relationship between the following fighters (fighter is a Node{name,weight}, beat is a relationship).
 - Khabib Nurmagomedov(155) > Rafael Dos Anjos (155)
 - Rafael Dos Anjos > Neil Magny(170)
 - Jon Jones(205) > Daniel Cormier(205)
 - Michael Bisping (185)> Matt Hamill (185)
 - Jon Jones > Brandon Vera (205)
 - Brandon Vera > Frank Mir (230)
 - Frank Mir > Brock Lesnar(230)
 - Neil Magny > Kelvin Gastelum(185)
 - Kelvin Gastelum > Michael Bisping
 - Michael Bisping > Matt Hamill
 - Michael Bisping > Kelvin Gastelum
 - Matt Hamill > Jon Jones
- **Submit a single file with the query or queries which create the database and a figure with a graph.**

Exercise 2

- Write a *Cypher* queries to:
 - 1) Return all **middle/welter/light** weight fighters (155, 170, 185) who at least have one win and grouped by weight.
 - 2) Return fighters who had **1-1 record** with each other. Use **Count** from the aggregation functions. Pair repetitions is accepted. Expected two column response. Hint: “*MATCH (f:Fighter)-[b:beats]->(f2:Fighter) - [b2:beats] -> (f3:Fighter) ...*” this part of the query might be useful.
 - 3) Return all fighters who has the greatest number of fights.
 - 4) Return **undefeated fighters (0 losses)** and **defeated fighters (0 wins)**. Expected two rows response.
- **Submit Queries and Screenshots with responses of Queries.**

Useful Links

- NEO4J Tutorial - <https://neo4j.com/developer/get-started/>
- Aggregation funcs - <https://neo4j.com/docs/cypher-manual/current/functions/aggregating/>
- Clauses - <https://neo4j.com/docs/cypher-manual/current/clauses/>
- Why we should use the neo4j - https://www.youtube.com/watch?v=_D19h5s73Co
- NEO4J - <https://neo4j.com/>
- <https://habr.com/ru/post/219441/>