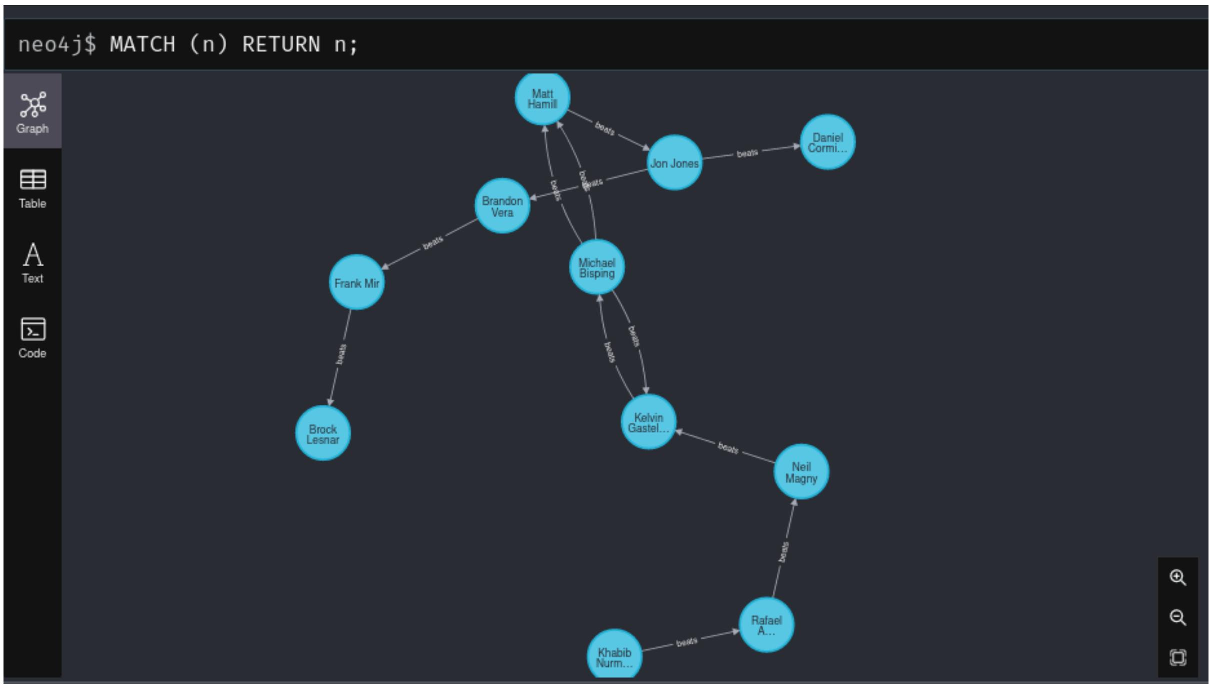
Part 1

Creating the fighters and relations:

```
Create(kn:Fighter {name: 'Khabib Nurmagomedov',weight:'155'}),(rda:Fighter {name: 'Rafael Dos Anjos', weight:'155'}),(nm: Fighter {name:'Neil Magny', weight:'170'}), (jj: Fighter {name:'Jon Jones', weight:'205'}), (dc: Fighter {name:'Daniel Cormier', weight:'205'}), (mb: Fighter {name:'Michael Bisping', weight:'185'}), (mh: Fighter {name:'Matt Hamill', weight:'185'}), (bv: Fighter {name:'Brandon Vera', weight:'205'}), (fm: Fighter {name:'Frank Mir', weight:'230'}), (bl: Fighter {name:'Brock Lesnar', weight:'230'}), (kg: Fighter {name:'Kelvin Gastelum', weight:'185'}), (kn)-[:beats]->(rda), (rda)-[:beats]->(nm), (jj)-[:beats]->(dc), (mb)-[:beats]->(mh), (jj)-[:beats]->(bv), (bv)-[:beats]->(fm), (fm)-[:beats]->(bl), (nm)-[:beats]->(kg), (kg)-[:beats]->(mb), (mb)-[:beats]->(mh), (mb)-[:beats]->(kg), (mh)-[:beats]->(jj)
```

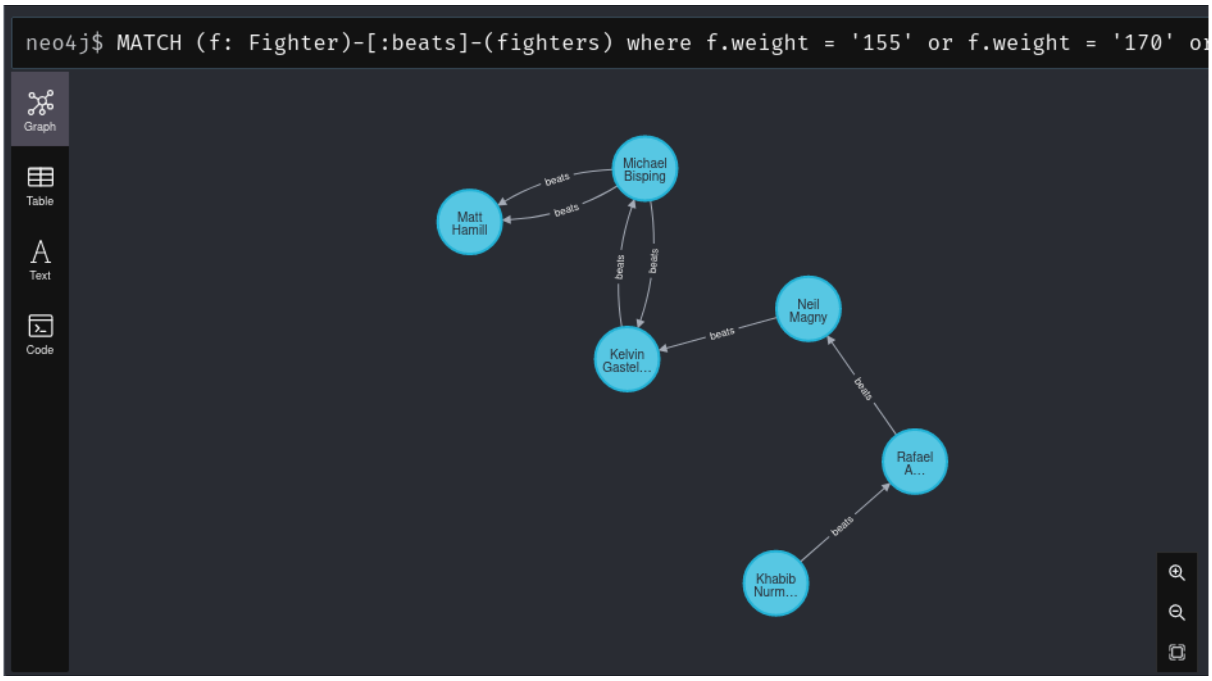


```
neo4j$ MATCH (n) RETURN n;
```

Part 2

1) Return all middle/Walter/light weight fighters (155,170,185) who at least have one win.

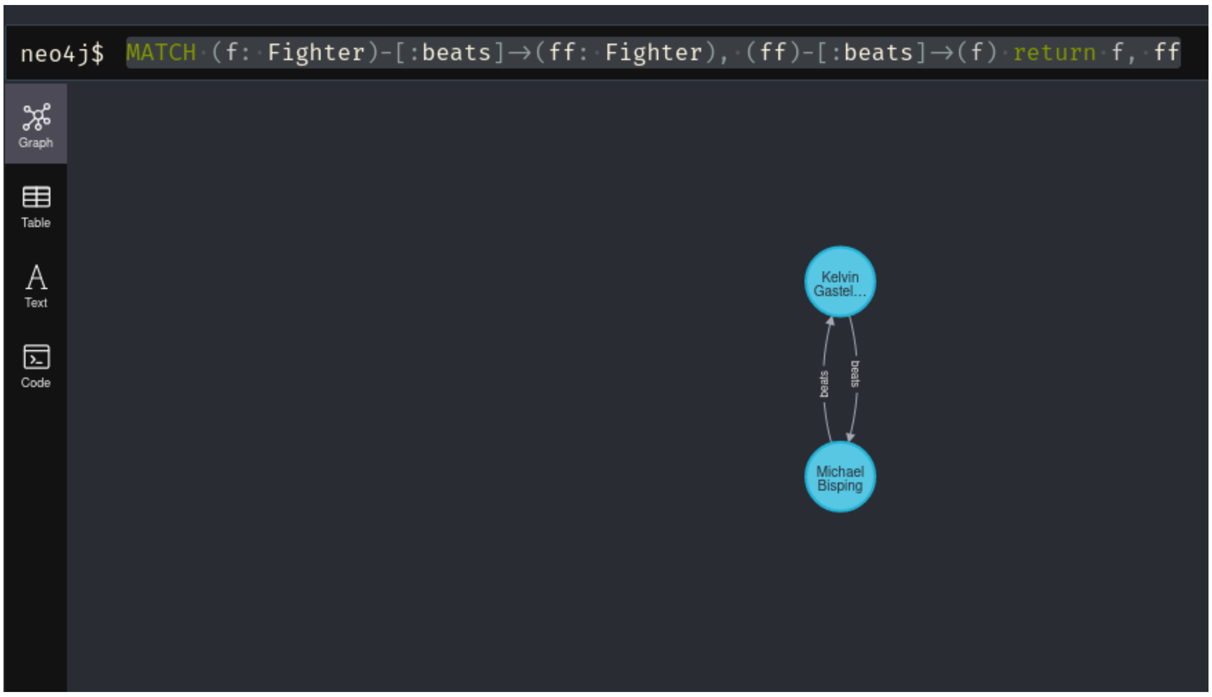
```
MATCH (f: Fighter)-[:beats]-(fighters) where f.weight = '155' or f.weight = '170' or f.weight = '185' return f
```



```
neo4j$ MATCH (f: Fighter)-[:beats]-(fighters) where f.weight = '155' or f.weight = '170' o
```

2) Return fighters who had 1-1 record with each other. Use Count from the aggregation functions

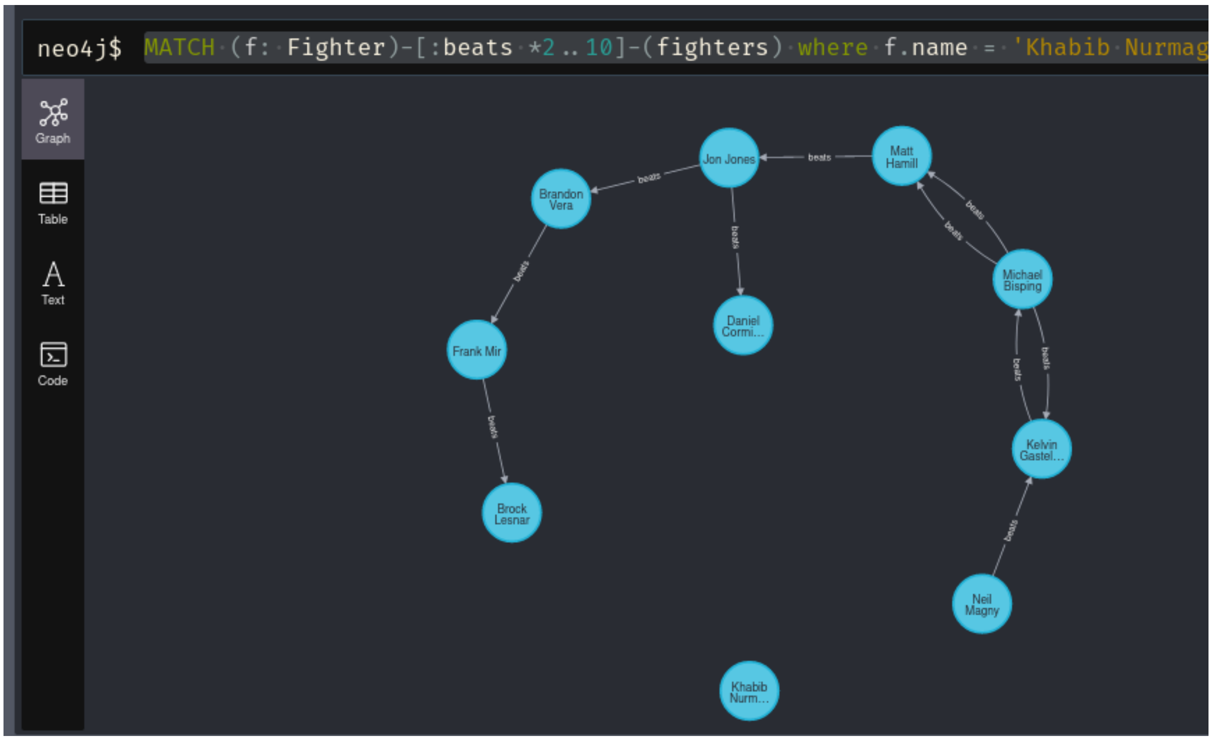
```
MATCH (f: Fighter)-[:beats]->(ff: Fighter), (ff)-[:beats]->(f) return f, ff
```



```
neo4j$ MATCH (f: Fighter)-[:beats]→(ff: Fighter), (ff)-[:beats]→(f) return f, ff
```

3) Return all fighter that can “Khabib Nurmagomedov” beat them and he didn’t have a fight with them yet.

```
MATCH (f: Fighter)-[:beats *2..10]-(fighters) where f.name = 'Khabib Nurmagomedov' return
f, fighters
```



```
neo4j$ MATCH (f: Fighter)-[:beats *2..10]-(fighters) where f.name = 'Khabib Nurmag
```