COMS W4111-002/V002 (Spring 2023) Introduction to Databases

</span>

Homework 3c: Non-Programming

Overview

- There are three parts to HW 3:
  - `3a` : Written questions
  - `3b:` A set of common practical tasks.
  - `3c:` Programming and Non-Programming track specific tasks.
- This notebook define part `3c, non-programming.`
- You will implement additional data loading and transformation, and write queries to enable visualization.
- These tasks are common "data engineering" as part of data science, operations research, etc.

Setup

In [1]:
```
import pandas as pd
```

In [2]:
```
%load_ext sql
```

In [3]:
```
%sql mysql+pymysql://root:Edy990127@localhost
```

Out[3]:
```
'Connected: root@None'
```

Note: Please use _S22_W4111_3c for you SQL database.

In [4]:
```
%sql use zz_S22_W4111_HW3_c;
```

Out[4]:
```
 * mysql+pymysql://root:***@localhost
0 rows affected.
[]
```

In [5]:
```
%sql show tables from zz_S22_W4111_HW3_c;
```

```
 * mysql+pymysql://root:***@localhost
2 rows affected.
```

Out[5]:

Tables_in_zz_s22_w4111_hw3_c
character_relationships
episodes_scenes_all

Note:

```
- Using the helper files I provided caused more confusion than help. So,
we will use the APIs directly.
- Set your URLs and passwords below for your work, but do not include in
your submission.
```

In [6]:
```
from py2neo import data, Graph, NodeMatcher, Node, Relationship, RelationshipMatcher
```

In [7]:
```
neo_g = g = Graph("neo4j+s://13f30c6c.databases.neo4j.io:7687",
                  auth=("neo4j", "isZCQgcAMZ9UusAH_w-hoqAtB7TsjJnXiEwrvbuCbGA"))
```

In [8]:
```
cypher_q = "match (n:GoT_Character {characterName: $c_name}) return n"
```

In [9]:
```
result = neo_g.run(cypher_q, c_name='Jon Snow')
```

In [10]:
```
for r in result:
    print(r)
```

In [11]:
```
q2 = """
    match (n:GoT_Character {characterName: 'Sansa Stark'})-[:SIBLINGS]-(s)-[:KILLED]->(
    return s.characterName, v.characterName
"""
```

In [12]:
```
result = neo_g.run(q2)
```

In [13]:
```
r_list = []
for r in result:
    r_list.append(dict(r))
```

In [14]:
```
v_df = pd.DataFrame(r_list)
v_df
```

Out[14]: —

In [15]:
```
from pymongo import MongoClient
```

```
In [16]: client = MongoClient("mongodb+srv://de2418:Edy990127@aaaa.1raes.mongodb.net/test?retryW
```

```
In [17]: filter={"seasonNum": 1}
         p_clause = {
             "seasonNum": 1,
             "episodeNum": 1,
             "episodeAirDate": 1,
             "episodeTitle": 1,
             "episodeDescription": 1
         }

         result = client['CU_Example_GoT']['episodes'].find(
             filter, p_clause
         )
```

```
In [18]: result = list(result)
         result_df = pd.DataFrame(result)
         result_df
```

```
Out[18]: —
```

```
In [19]: from sqlalchemy import create_engine
```

```
In [20]: engine = create_engine("mysql+pymysql://root:Edy990127@localhost")
```

```
In [21]: import json
```

```
In [22]: def load_json(fn):
             result = None
             with open(fn, "r") as in_file:
                 result = json.load(in_file)
             return result
```

```
In [23]: characters = load_json("./data/characters.json")
```

```
In [24]: episodes = load_json("./data/episodes.json")
```

```
In [25]: locations = load_json("./data/locations.json")
```

```
In [26]: characters_groups = load_json("./data/characters-groups.json")
```

```
In [27]: from pymongo import MongoClient
         client = MongoClient("mongodb+srv://de2418:Edy990127@aaaa.1raes.mongodb.net/GoT?retryWr
```

Additional Data Loading

- You loaded the character information into MongoDB.
- The document for a character may contain sections the lists relationships between characters. For example, the following code snippet shows the following relationships:
  - `Aegon Targaryen - PARENTS -> Elia Martell`
  - `Aegon Targaryen - PARENTS -> Rhaegar Targaryen`
  - `Aegon Targaryen - SIBLINGS -> Rhaenys Targaryen`
  - `Aegon Targaryen - SIBLINGS -> Jon Snow`
  - `Aegon Targaryen - KILLED_BY -> Gregor Clegane`

In [28]:
```
filter={
    'characterName': 'Aegon Targaryen'
}

result = client['Example_GoT']['characters'].find(
  filter=filter
)
result
```

Out[28]:
```
<pymongo.cursor.Cursor at 0x254af22eb50>
```

In [29]:
```
result = list(result)
```

In [30]:
```
result
```

Out[30]:
```
[]
```

In [31]:
```
non_complex = []
```

In [32]:
```
result = client['GoT']['characters'].find(
)
```

In [33]:
```
for c in result:
    for k,v in c.items():
        if type(v) in [str, int, bool]:
            non_complex.append(k)
```

In [34]:
```
non_complex = set(non_complex)
non_complex
```

Out[34]:
```
{'actorLink',
 'actorName',
 'characterImageFull',
 'characterImageThumb',
 'characterLink',
 'characterName',
 'houseName',
 'kingsguard',
 'nickname',
 'royal'}
```

In [35]:
```
character_project = {'actorLink':1,
                     'actorName':1,
                     'characterImageFull':1,
                     'characterImageThumb':1,
                     'characterLink':1,
                     'characterName':1,
                     'houseName':1,
                     'kingsguard':1,
                     'nickname':1,
                     'royal':1}
```

In [36]:
```
characters = result = client['GoT']['characters'].find(
    None,
    {'actorLink':1,
     'actorName':1,
     'characterImageFull':1,
     'characterImageThumb':1,
     'characterLink':1,
     'characterName':1,
     'houseName':1,
     'kingsguard':1,
     'nickname':1,
     'royal':1}
)
```

In [37]:
```
characters = list(result)
```

In [38]:
```
characters[0]
```

Out[38]:
```
{'_id': ObjectId('625e15debfd4b732a108d58f'),
 'characterName': 'Addam Marbrand',
 'characterLink': '/character/ch0305333/',
 'actorName': 'B.J. Hogg',
 'actorLink': '/name/nm0389698/'}
```

In [39]:
```
check_q = "match (c:GoT {characterName : $c_name}) return c"
result = neo_g.run(check_q, c_name = 'Addam Marbrand')
```

In [40]:
```
if len(result.data()) == 0:
    q = "create (c:GoT_Character {characterName : $c_name, UUID: $u_id})"
    neo_g.run(q, c_name = characters[11]['characterName'], u_id = str(characters[11]['_
```

In [41]:
```
result
```

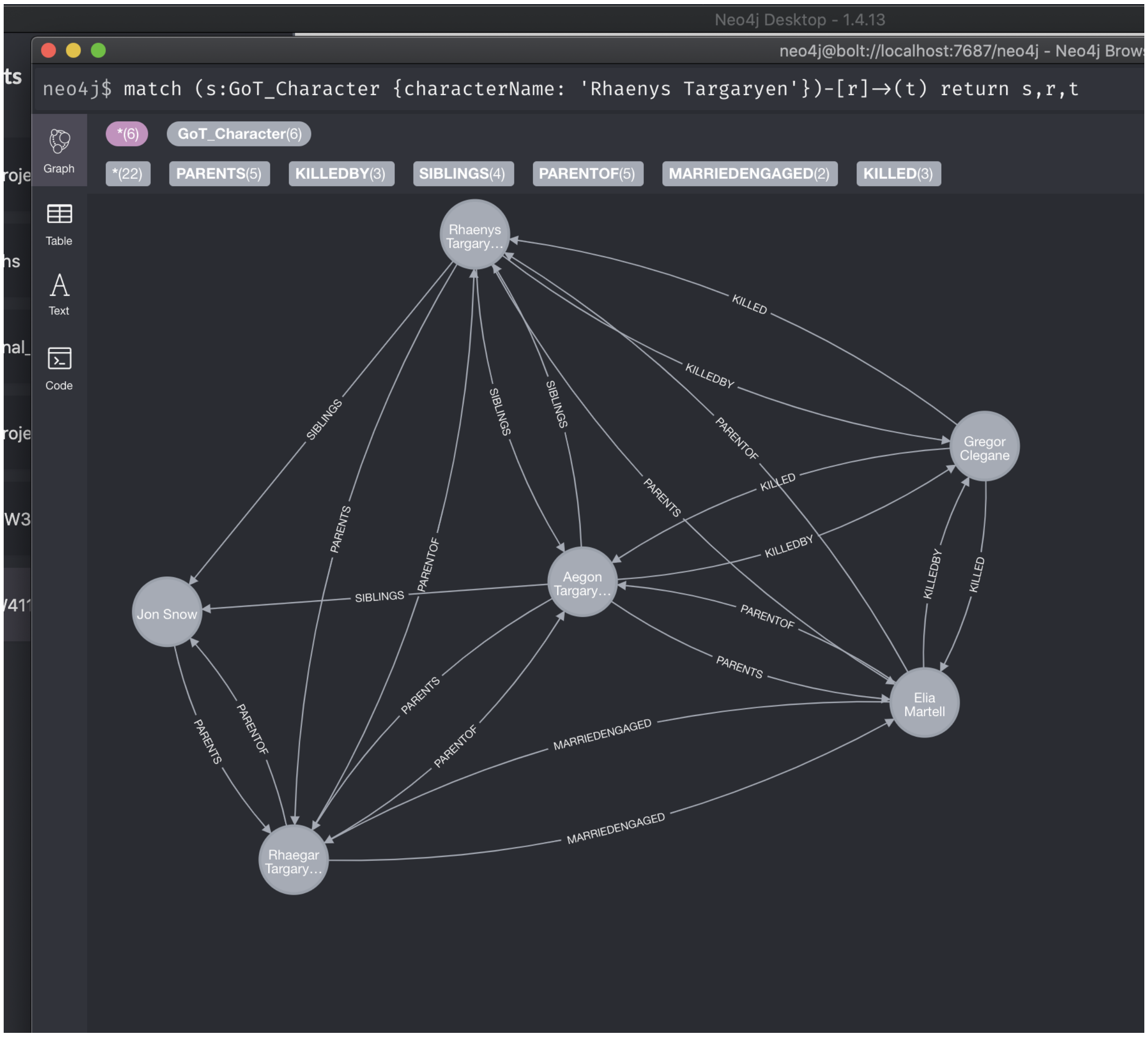
Out[41]: (No data)

- The array below lists the fields in `characters` that document relationships.

- **Note:** There is a data error. The documents contain relationships "sibling" and "siblings," we will fix this later.

- The first task is to write code that uses MongoDB aggregation(s) to load the relationship information into a MySQL database.
- The table I created looks like:

- The second task is to use the created table to load the information into your Neo4j graph.
- My loaded data looks like.



```
neo4j$ match (s:GoT_Character {characterName: 'Rhaenys Targaryen'})-[r]→(t) return s,r,t
```

- The following query also displays the information:

- The tasks are:
  - Write the MongoDB aggregations to load the data into MySQL. You can also use Pandas and SQLAlchemy to write the data to MySQL.

- Write a simple program the queries the relationships table you created and loads the information into Neo4j using the format from my examples.

In [42]:
```
# MongoDB Aggregations
#
# Put your code here
```

In [43]:
```
_relationships = [
    "sibling",
    "marriedEngaged",
    "servedBy",
    "killed",
    "killedBy",
    "abducted",
    "abductedBy",
    "allies",
    "parentOf",
    "guardianOf",
    "guardedBy",
    "siblings",
    "parents",
    "serves"
]
```

In [44]:
```
def get_relationship_pairs(relationship_name):
    result = client["GoT"]['characters'].aggregate([
        {
            '$unwind': {
                'path': '$' + relationship_name
            }
        }, {
            '$project': {
                'sourceCharacterName': '$characterName',
                'targetCharacterName': '$' + relationship_name,
                'relationshipType': relationship_name
            }
        }
    ])
    return result
```

In [45]:
```
import pandas as pd
from sqlalchemy import create_engine
sql_engine = create_engine('mysql+pymysql://root:Edy990127@localhost')
```

In [46]:
```
df = pd.DataFrame(columns = ['_id', "sourceCharacterName", "targetCharacterName", "rela
```

In [47]:
```
for i in _relationships:
    res = get_relationship_pairs(i)
    res = list(res)
    df = df.append(pd.DataFrame(res, index = None))
```

In [48]:
```
df.reset_index(drop=True, inplace=True)
```

In [49]:
```
def create_cool_relationship(s_character, relationship, t_character):
    cypher_q1 = """
        match (c:GoT_Character {characterName: $s_c}),
                (t:GoT_Character {characterName: $t_c})
        create (s)-[:"""

    cypher_q2 = """]->(t)"""

    full_cypher = cypher_q1 + relationship + cypher_q2
    print(full_cypher)
    neo_g.run(full_cypher, s_c = s_character, t_c = t_character)
```

In [50]:
```
create_cool_relationship('Aegon Targaryen','siblings','Rhaenys Targaryen')
```

```
        match (c:GoT_Character {characterName: $s_c}),
                (t:GoT_Character {characterName: $t_c})
        create (s)-[:siblings]->(t)
```

In [51]:
```
df.to_sql(
    "character_relationships", con=sql_engine, if_exists="replace", index=False,
    schema="zz_S22_W4111_HW3_c")
```

In [52]:
```
# Load neo4j
# Put your code here.
#
```

In [53]:
```
from py2neo import data, Graph, NodeMatcher, Node, Relationship, RelationshipMatcher
g = Graph('neo4j+s://13f30c6c.databases.neo4j.io:7687',
          auth = ("neo4j", "isZCQgcAMZ9UusAH_w-hoqAtB7TsjJnXiEwrvbuCbGA"))
```

In [54]:
```
q = "match (p:Person) where p.name=$name return p"
res = g.run(q, name = "Tom Hanks")
print(type(res))
for r in res:
    print(type(r))
    print(r)
    print("Labels = ", r['p'].labels)
    print("Properties = ", dict(r['p']))
```

```
<class 'py2neo.cypher.Cursor'>
```

In [55]:
```
charactername = pd.unique(df.sourceCharacterName.append(df.targetCharacterName))
```

In [56]:
```
cypher_q3 = """
        match (s:GoT_CharacterName {CharacterName: $s_name}),
                (t:GoT_CharacterName {CharacterName: $t_name})
        create (s)-[:"""
cypher_q4 = """]->(t)"""
```

In [57]:
```
cypher_g = "create (c:GoT_CharacterName {CharacterName: $name}) return c"
```

In [58]:
```
for i in range(len(charactername)):
    result = g.run(cypher_g, name = charactername[i])
```

In [59]:
```
for i in range(len(df.targetCharacterName)):
    g.run(cypher_q3 + df.relationshipType[i] + cypher_q4,
                   s_name = df.sourceCharacterName[i],
                   t_name = df.targetCharacterName[i])
```

Tests: Put some tests that demonstrate that you have correctly loaded the data.

In [60]:
```
%sql SELECT * FROM zz_S22_W4111_HW3_c.character_relationships where targetCharacterName
```

```
 * mysql+pymysql://root:***@localhost
11 rows affected.
```

Out[60]:

_id	sourceCharacterName	targetCharacterName	relationshipType
625e15dfbfd4b732a108d5ce	Elia Martell	Rhaegar Targaryen	marriedEngaged
625e15e1bfd4b732a108d646	Lyanna Stark	Rhaegar Targaryen	marriedEngaged
625e15e3bfd4b732a108d6ae	Robert Baratheon	Rhaegar Targaryen	killed
625e15e1bfd4b732a108d646	Lyanna Stark	Rhaegar Targaryen	abductedBy
625e15debfd4b732a108d592	Aerys II Targaryen	Rhaegar Targaryen	parentOf
625e15e3bfd4b732a108d6a8	Rhaella Targaryen	Rhaegar Targaryen	parentOf
625e15debfd4b732a108d5bb	Daenerys Targaryen	Rhaegar Targaryen	siblings
625e15e5bfd4b732a108d6f2	Viserys Targaryen	Rhaegar Targaryen	siblings
625e15debfd4b732a108d590	Aegon Targaryen	Rhaegar Targaryen	parents
625e15e0bfd4b732a108d60a	Jon Snow	Rhaegar Targaryen	parents
625e15e3bfd4b732a108d6a9	Rhaenys Targaryen	Rhaegar Targaryen	parents

In [61]:
```
cypher_qq = """
match (s:GoT_CharacterName {CharacterName: 'Rhaegar Targaryen'})-[r]->(t) return s,r,t
"""

result = neo_g.run(cypher_qq)
result = list(result)
```

In [62]:
```
simple_r = []
for r in result:
    simple_r.append(
        {
            "sourceCharacterName": r['s']['CharacterName'],
            "relationshipType": ",".join(list(set(r['r'].types()))),
            "targetCharacterName": r['t']['CharacterName']
        }
    )
```

In [63]:

```
simple_r_df = pd.DataFrame(simple_r)
simple_r_df
```

Out[63]:

	sourceCharacterName	relationshipType	targetCharacterName
0	Rhaegar Targaryen	parents	Aerys II Targaryen
1	Rhaegar Targaryen	parents	Rhaella Targaryen
2	Rhaegar Targaryen	siblings	Viserys Targaryen
3	Rhaegar Targaryen	siblings	Daenerys Targaryen
4	Rhaegar Targaryen	parentOf	Aegon Targaryen
5	Rhaegar Targaryen	parentOf	Rhaenys Targaryen
6	Rhaegar Targaryen	parentOf	Jon Snow
7	Rhaegar Targaryen	abducted	Lyanna Stark
8	Rhaegar Targaryen	killedBy	Robert Baratheon
9	Rhaegar Targaryen	marriedEngaged	Lyanna Stark
10	Rhaegar Targaryen	marriedEngaged	Elia Martell

Some Interesting Queries

- The zip file for the HW contains a file "scenes_all.csv."
- The following code will read the CSV file and create a table in your database. Make sure you set the correct database name.

In [64]:

```
df = pd.read_csv("./scenes_all.csv")
```

In [65]:

```
df = df[['seasonNum', 'episodeNum', 'sceneNum', 'sceneStartTime', 'sceneEndTime',
            'sceneLocation', 'sceneSubLocation', 'characterName']]
```

In [66]:

```
df.to_sql("episodes_scenes_all", schema="zz_S22_W4111_HW3_c", con=sql_engine, index=Fal
```

- We can now do some tests.

In [67]:

```
%%sql

select * from zz_S22_W4111_HW3_c.episodes_scenes_all
    where episodeNum=1 and sceneNum=1;
```

```
 * mysql+pymysql://root:***@localhost
14 rows affected.
```

Out[67]:

seasonNum	episodeNum	sceneNum	sceneStartTime	sceneEndTime	sceneLocation	sceneSubLocation
1	1	1	0:00:40	0:01:45	The Wall	Castle Black
1	1	1	0:00:40	0:01:45	The Wall	Castle Black
1	1	1	0:00:40	0:01:45	The Wall	Castle Black
2	1	1	0:02:29	0:03:08	The Crownlands	King's Landing
2	1	1	0:02:29	0:03:08	The Crownlands	King's Landing
2	1	1	0:02:29	0:03:08	The Crownlands	King's Landing
2	1	1	0:02:29	0:03:08	The Crownlands	King's Landing
3	1	1	0:00:06	0:00:17	North of the Wall	Fist of the First Men
4	1	1	0:00:07	0:01:47	The Crownlands	King's Landing
5	1	1	0:05:30	0:07:07	The Westerlands	Outside Casterly Rock
5	1	1	0:05:30	0:07:07	The Westerlands	Outside Casterly Rock
6	1	1	0:07:04	0:08:03	The Wall	Castle Black
7	1	1	0:03:54	0:04:20	The Riverlands	The Twins
8	1	1	0:04:39	0:04:50	The North	Outside Winterfell

Query 1:

- Use the table you created. Produce a table of the form:

`(characterName, sceneLocation, sceneSubLocation)`

- The shows the total time a character spent in locations and sub-locations.

- My answer looks like ... ...

In [68]:

```
%%sql
select characterName, sceneLocation, sceneSubLocation,
sum(TIME_TO_SEC(sceneEndTime) - TIME_TO_SEC(sceneStartTime)) as time_in_Location
from zz_S22_W4111_HW3_c.episodes_scenes_all
group by characterName, sceneLocation, sceneSubLocation
order by time_in_Location desc
LIMIT 25
```

```
 * mysql+pymysql://root:***@localhost
```

```
25 rows affected.
```

Out[68]:

characterName	sceneLocation	sceneSubLocation	time_in_Location
Cersei Lannister	The Crownlands	King's Landing	23426
Tyrion Lannister	The Crownlands	King's Landing	18216
Jon Snow	The Wall	Castle Black	11873
Sansa Stark	The North	Winterfell	10423
Jaime Lannister	The Crownlands	King's Landing	10026
Sansa Stark	The Crownlands	King's Landing	8713
Lord Varys	The Crownlands	King's Landing	7814
Joffrey Baratheon	The Crownlands	King's Landing	7660
Margaery Tyrell	The Crownlands	King's Landing	7595
Bran Stark	The North	Winterfell	7398
Jon Snow	The North	Winterfell	7113
Samwell Tarly	The Wall	Castle Black	6770
Daenerys Targaryen	Meereen	None	6732
Tywin Lannister	The Crownlands	King's Landing	6537
Grand Maester Pycelle	The Crownlands	King's Landing	6459
Petyr Baelish	The Crownlands	King's Landing	6434
Eddard Stark	The Crownlands	King's Landing	6081
Theon Greyjoy	The North	Winterfell	5744
Missandei	Meereen	None	5517
Tommen Baratheon	The Crownlands	King's Landing	5385
Arya Stark	The North	Winterfell	5272
Arya Stark	Braavos	None	5011
Tyrion Lannister	Meereen	None	4975
Gregor Clegane	The Crownlands	King's Landing	4713
Tyrion Lannister	The Crownlands	Dragonstone	4466

- Using the preceding query, write a query that shows the percentage of time spent in locations for a character and has the total number of scenes.
- The percentage of time is the time in a location, sub-location compared to total time on screen.
- My query below shows an answer for characters with at least 50 scenes.

In [69]:

```
%%sql
select b.characterName, c.sceneLocation, c.sceneSubLocation, b.no_of_scene,
c.time_in_Location, b.total_time, round((c.time_in_Location/b.total_time) * 100, 1) as
(select a.characterName as characterName, sum(a.time_in_Location) as total_time, sum(a.
from
(select characterName, sceneLocation, sceneSubLocation, count(*) as no_of_scenes,
sum(TIME_TO_SEC(sceneEndTime) - TIME_TO_SEC(sceneStartTime)) as time_in_Location
from zz_S22_W4111_HW3_c.episodes_scenes_all
group by characterName, sceneLocation, sceneSubLocation
order by time_in_Location desc) as a
group by characterName) as b
join
(select characterName, sceneLocation, sceneSubLocation, count(*) as no_of_scenes,
sum(TIME_TO_SEC(sceneEndTime) - TIME_TO_SEC(sceneStartTime)) as time_in_Location
from zz_S22_W4111_HW3_c.episodes_scenes_all
group by characterName, sceneLocation, sceneSubLocation
order by time_in_Location desc) as c
on b.characterName = c.characterName
having no_of_scene >= 50
order by characterName, time_percent
LIMIT 50
```

```
 * mysql+pymysql://root:***@localhost
50 rows affected.
```

Out[69]:

characterName	sceneLocation	sceneSubLocation	no_of_scene	time_in_Location	total_time	time_percen
Alliser Thorne	The Wall	Castle Black	51	4070	4070	100.
Arya Stark	The North	The Kingsroad South to King's Landing	360	16	24315	0.
Arya Stark	The North	Winter Town	360	59	24315	0.
Arya Stark	The Sunset Sea	None	360	81	24315	0.
Arya Stark	The North	Outside Winterfell	360	63	24315	0.
Arya Stark	The Vale	The Eyrie	360	128	24315	0.
Arya Stark	The Riverlands	Away from the Twins	360	130	24315	0.
Arya Stark	The Vale	Coast of the Vale	360	157	24315	0.
Arya Stark	The Riverlands	Red Fork	360	167	24315	0.
Arya Stark	The Riverlands	Outside Harrenhal	360	185	24315	0.
Arya Stark	The Riverlands	South to King's Landing	360	249	24315	1.
Arya Stark	The Riverlands	North to the Red Fork	360	246	24315	1.
Arya Stark	The Riverlands	The Twins	360	292	24315	1.
Arya Stark	The Riverlands	To The Twins	360	431	24315	1.
Arya Stark	The Vale	To The Eyrie	360	503	24315	2.
Arya Stark	The Crownlands	Outside King's Landing	360	751	24315	3.
Arya Stark	The Riverlands	Crossroads Inn	360	1011	24315	4.
Arya Stark	The Riverlands	The Kingsroad	360	1047	24315	4.

characterName	sceneLocation	sceneSubLocation	no_of_scene	time_in_Location	total_time	time_percen
Arya Stark	The Riverlands	Hollow Hill	360	1275	24315	5.
Arya Stark	The Riverlands	To The Eyrie	360	1722	24315	7.
Arya Stark	The Riverlands	Harrenhal	360	2141	24315	8.
Arya Stark	The Crownlands	King's Landing	360	3378	24315	13.
Arya Stark	Braavos	None	360	5011	24315	20.
Arya Stark	The North	Winterfell	360	5272	24315	21.
Barristan Selmy	Meereen	Outside Meereen	77	86	6577	1.
Barristan Selmy	The Crownlands	The Kingswood	77	113	6577	1.
Barristan Selmy	Yunkai	None	77	337	6577	5.
Barristan Selmy	Yunkai	Outside Yunkai	77	857	6577	13.
Barristan Selmy	Astapor	None	77	904	6577	13.
Barristan Selmy	The Crownlands	King's Landing	77	1822	6577	27.
Barristan Selmy	Meereen	None	77	2458	6577	37.
Beric Dondarrion	North of the Wall	The Wall	85	17	5171	0.
Beric Dondarrion	The North	Last Hearth	85	224	5171	4.
Beric Dondarrion	The Crownlands	King's Landing	85	245	5171	4.
Beric Dondarrion	The Riverlands	Forest	85	282	5171	5.
Beric Dondarrion	The Wall	Eastwatch	85	312	5171	6.
Beric Dondarrion	The Riverlands	To The Eyrie	85	382	5171	7.
Beric Dondarrion	The North	Winterfell	85	880	5171	17.
Beric Dondarrion	The Riverlands	Hollow Hill	85	1046	5171	20.
Beric Dondarrion	North of the Wall	The Haunted Forest	85	1783	5171	34.
Bran Stark	Dorne	None	248	30	14346	0.
Bran Stark	North of the Wall	The Wall	248	89	14346	0.
Bran Stark	North of the Wall	The Lands of Always Winter	248	199	14346	1.

characterName	sceneLocation	sceneSubLocation	no_of_scene	time_in_Location	total_time	time_percen
Bran Stark	The North	The Wolfswood	248	273	14346	1.
Bran Stark	The Crownlands	King's Landing	248	278	14346	1.
Bran Stark	North of the Wall	The Haunted Forest	248	313	14346	2.
Bran Stark	The Wall	Nightfort	248	414	14346	2.
Bran Stark	The North	The Gift	248	429	14346	3.
Bran Stark	North of the Wall	Outside the Three-Eyed Raven	248	477	14346	3.
Bran Stark	Dorne	Tower of Joy	248	497	14346	3.

- Using the Neo4j Graph. Wrote a function that returns the characters related to a source character by a list of relationships.

In [70]:

```
def get_by_relationships(characterName, relationship_list):
    simple_r = []
    cypher_q1 = """
        match (s:GoT_CharacterName {CharacterName: '"""

    cypher_q2 = """' })-[r:"""

    cypher_q3 = """]->(t)  return s, r, t"""


    full_cypher = []
    result = list()
    for i in range(len(relationship_list)):
        full_cypher.append(cypher_q1 + characterName + cypher_q2 + relationship_list[i]
        res = neo_g.run(full_cypher[i])
        res = list(res)
        result.append(res)
        for r in res:
            simple_r.append(
                {
                    "sourceCharacterName": r['s']['CharacterName'],
                    "relationshipType": ",".join(list(set(r['r'].types()))),
                    "targetCharacterName": r['t']['CharacterName']
                }
            )
    return simple_r
```

In [71]:

```
result = get_by_relationships("Arya Stark", ["siblings", "parents"])
```

In [72]:

```
result
```

Out[72]:

```
[{'sourceCharacterName': 'Arya Stark',
  'relationshipType': 'siblings',
  'targetCharacterName': 'Rickon Stark'},
 {'sourceCharacterName': 'Arya Stark',
  'relationshipType': 'siblings',
  'targetCharacterName': 'Bran Stark'},
 {'sourceCharacterName': 'Arya Stark',
  'relationshipType': 'siblings',
  'targetCharacterName': 'Sansa Stark'},
 {'sourceCharacterName': 'Arya Stark',
  'relationshipType': 'siblings',
  'targetCharacterName': 'Robb Stark'},
 {'sourceCharacterName': 'Arya Stark',
  'relationshipType': 'parents',
  'targetCharacterName': 'Catelyn Stark'},
 {'sourceCharacterName': 'Arya Stark',
  'relationshipType': 'parents',
  'targetCharacterName': 'Eddard Stark'}]
```