MONASH University
Information Technology

ASSESSMENT COVER SHEET

Student ID number	27135519
Given Name	Hongliang
Family name	Tang

Unit Name and Code:	**Advanced Database FIT5137**
Campus:	**Clayton**
Assignment Title:	**Assignment 2**
Name of Lecturer:	Dr. Agnes Haryanto
Name of Tutor:	Shuyi Sun
Tutorial Day and Time:	**Friday 8am**
Phone Number:	**0449288911**
Email Address:	**htan79@student.monash.edu**

Has any part of this assignment been previously submitted as part of another unit/course? ☐ Yes [✓ No]

Due Date:	**20/10/2021**	Date Submitted:	25/10/2021

All work must be submitted by the due date. If an extension of work is granted this must be specified with the signature of the lecturer/tutor.

Extension granted until (date)____________________ Signature of lecturer/tutor ___________________________

Please note that it is your responsibility to retain copies of your assessments.

Intentional plagiarism or collusion amounts to cheating under Part 7 of the Monash University (Council) Regulations

Plagiarism: Plagiarism means taking and using another person's ideas or manner of expressing them and passing them off as one's own. For example, by failing to give appropriate acknowledgement. The material used can be from any source (staff, students or the internet, published and unpublished works).

Collusion: Collusion means unauthorised collaboration with another person on assessable written, oral or practical work and includes paying another person to complete all or part of the work.

Where there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Associate Dean (Education) or delegate, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Discipline Panel for a hearing.

Student Statement:

- I have read the university's Student Academic Integrity Policy and Procedures.
- I understand the consequences of engaging in plagiarism and collusion as described in Part 7 of the Monash University (Council) Regulations http://adm.monash.edu/legal/legislation/statutes
- have taken proper care to safeguard this work and made all reasonable efforts to ensure it could not be copied.
- No part of this assignment has been previously submitted as part of another unit/course.
- I acknowledge and agree that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
  - i. provide to another member of faculty and any external marker; and/or
  - ii. submit it to a text matching software; and/or
  - iii. submit it to a text matching software which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.
- I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

Signature ....[illegible]........ Date………25/10/2021……………………………

* delete (iii) if not applicable

C. Tasks

C.1. Database Design

Identification of potential nodes and edges

Nodes:

Address (contain city and state properties from ufo csv), City, County, Day, Hour, Month, Shape, State, UfoInfo, WeatherCondition, WindDirection, Year, Fog, Hail, Snow, Thunder, Tornado, Rain

18 nodes

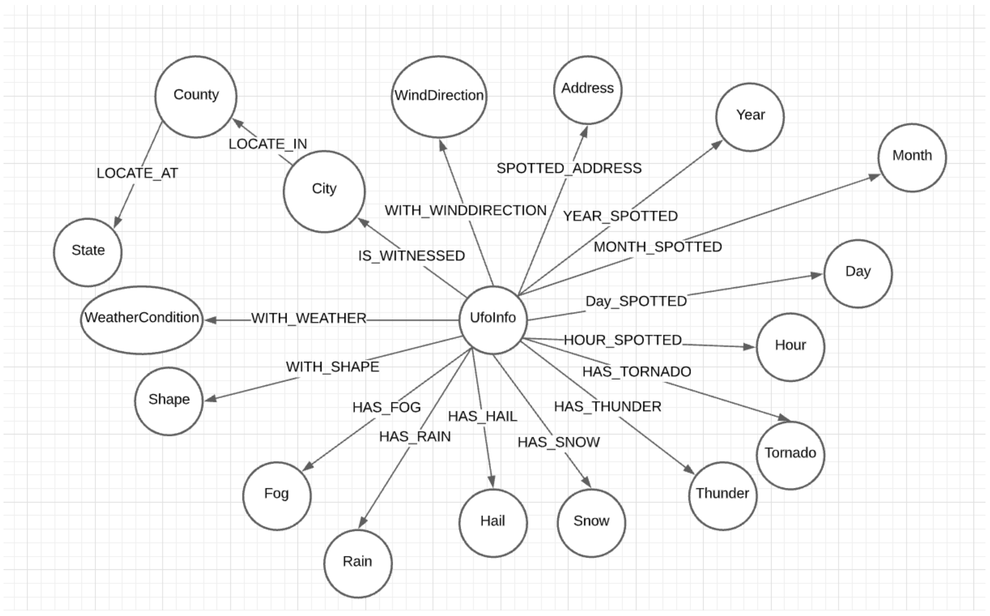
Edges:

Day_SPOTTED, HOUR_SPOTTED, IS_WITNESSED,LOCATE_AT,LOCATE_IN,MONTH_SPOTTED,SPOTTED_ADDRESS,WITH_SHAPE,WITH_WEATHER,WITH_WINDDIRECTION,YEAR_SPOTTED,

HAS_HAIL, HAS_RAIN, HAS_THUNDER, HAS_FOG, HAS_TORNADO, HAS_SNOW

17 relationships

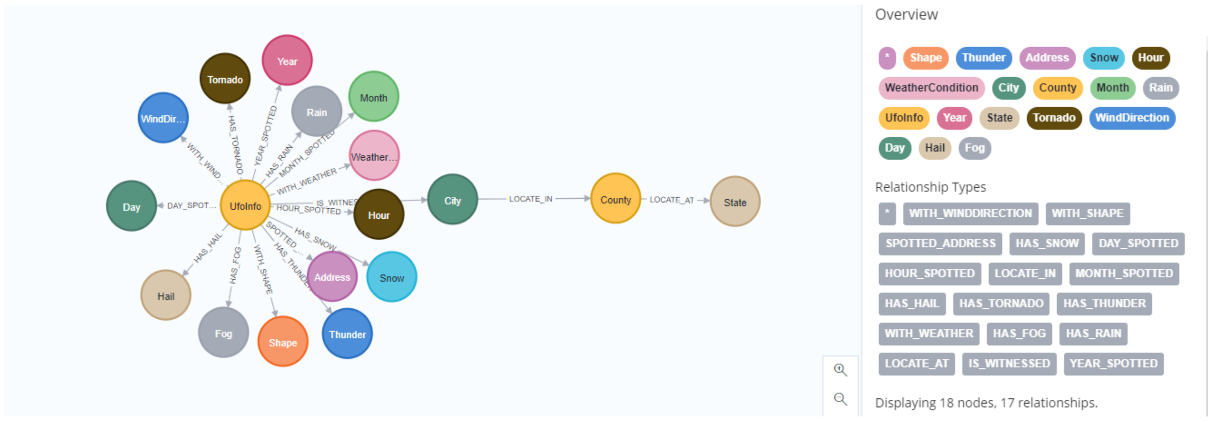
Graph illustration



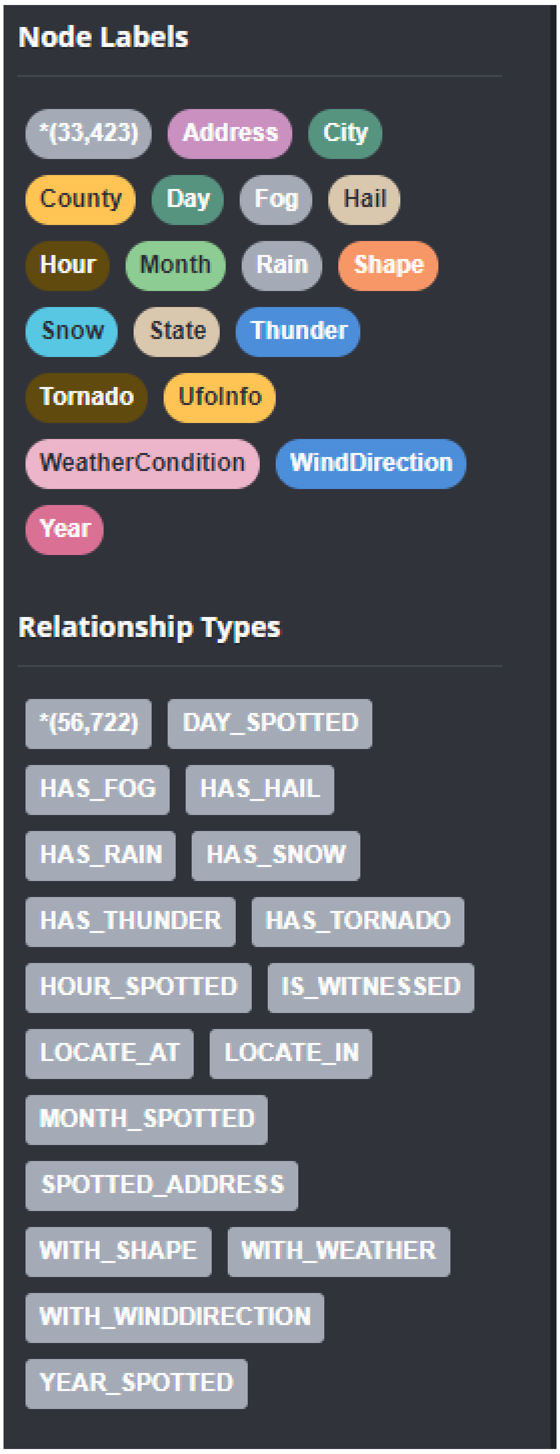
Nodes for the time are day,month,year,hour and those values are repeated multiple times, so it will be better to create node for those columns in ufo_a2.csv to reduce duplication in graph. Those time attributes will be related to the ufo through edges like YEAR_SPOTTED, MONTH_SPOTTED, Day_SPOTTED and HOUR_SPOTTED. The Address node is used to represent the address where the ufo was been recorded and this will be used to match with the geo location in states_a2.csv. The edge SPOTTED_ADDRESS is used to link ufo to the address. The weather condition and wind direction are extracted to create node for the same reason that I want to reduce repetitiveness in the data. Those two nodes are link to ufo those edge WITH_WEATHER and WITH_WINDDIRECTION. Properties like text, summary, duration, temp, heatindex, windchill, vis, dewpt, windspeed, humidity, wgust are information describing the observation on the ufo, I use the ufoInfo node to contain all the information. Boolean values like Fog, Hail, Snow, Thunder, Tornado, Rain are extracted, and I created node for each one of those. This helps reduce duplicative values. All those nodes will then be connected to UfoInfo with edges HAS_HAIL, HAS_RAIN, HAS_THUNDER, HAS_FOG, HAS_TORNADO, HAS_SNOW. Finally, I used the state_a2.csv to create 3 nodes, City,County and State.

Task C.1's code and screenshots of results.

```
//C.1. Database Design.
//import state.csv
load csv with headers from 'file:///states_a2.csv' as row
with row
merge (states:State{stateName:trim(toUpper(row.state))})  // state contain
counties
merge
(cities:City{cityName:trim(toUpper(row.city)),latitude:toFloat(row.lat),longit
ude:toFloat(row.lng)})-
[:LOCATE_IN]->(counties:County{countyName:trim(toUpper(row.countyName))}) -
[:LOCATE_AT]->(states);//county contain cities, same name county in different
state is still different county


// duration, text and summary has empty value, so case when is needed
otherwise will return error
//if line.xxx is used.
//assume 1 is true 0 is false
Load csv with headers from "file:///ufo_a2.csv" as line
with line
create (ufo:UfoInfo)
set ufo.duration = trim(COALESCE(line.duration,'Not recorded')),
    ufo.text = trim(COALESCE(line.text,'Not recorded')),
    ufo.summary = trim(COALESCE(line.summary,'Not recorded')),
    ufo.pressure = (case trim(line.pressure)
                    when 'NA' then null else toFloat(trim(line.pressure))END),
    ufo.temp = (case trim(line.temp)
                    when 'NA' then null else toFloat(trim(line.temp))END),
    ufo.heatindex = (case trim(line.heatindex)
                    when 'NA' then null else
toFloat(trim(line.heatindex))END),
    ufo.windchill = (case trim(line.windchill)
                    when 'NA' then null else
toFloat(trim(line.windchill))END),
    ufo.vis = (case trim(line.vis)
                    when 'NA' then null else toFloat(trim(line.vis))END),
    ufo.dewpt = (case trim(line.dewpt)
                    when 'NA' then null else toFloat(trim(line.dewpt))END),
    ufo.precip = (case trim(line.precip)
                    when 'NA' then null else toFloat(trim(line.precip))END),
    ufo.wspd = (case trim(line.wspd)
                    when 'NA' then null else toFloat(trim(line.wspd))END),
    ufo.hum = (case trim(line.hum)
                    when 'NA' then null else toFloat(trim(line.hum))END),
    ufo.wgust = (case trim(line.wgust)
                    when 'NA' then null else toFloat(trim(line.wgust))END)
//bool
create (hail:Hail)
set hail.status = (case trim(line.hail) when 1 then true else false end)
merge (ufo)-[:HAS_HAIL]->(hail)

create (rain:Rain)
set rain.status = (case trim(line.rain) when 1 then true else false end)
merge (ufo)-[:HAS_RAIN]->(rain)

create (thunder:Thunder)
set thunder.status = (case trim(line.thunder) when 1 then true else false end)
merge (ufo)-[:HAS_THUNDER]->(thunder)

create (fog:Fog)
set fog.status = (case trim(line.fog) when 1 then true else false end)
merge (ufo)-[:HAS_FOG]->(fog)

create (tornado:Tornado)
set tornado.status = (case trim(line.tornado) when 1 then true else false end)
merge (ufo)-[:HAS_TORNADO]->(tornado)

create (snow:Snow)
set snow.status = (case trim(line.snow) when 1 then true else false end)
merge (ufo)-[:HAS_SNOW]->(snow)

// year month day hour
merge (years:Year{year:toInteger(trim(line.year))})
merge (ufo)-[:YEAR_SPOTTED]->(years)
merge (months:Month{month:toInteger(trim(line.month))})
merge (ufo)-[:MONTH_SPOTTED]->(months)
merge (days:Day{day:toInteger(trim(line.day))})
merge (ufo)-[:DAY_SPOTTED]->(days)
merge (hours:Hour{hour:toInteger(trim(line.hour))})
merge (ufo)-[:HOUR_SPOTTED]->(hours)

//shape https://community.neo4j.com/t/load-csv-with-empty-cells/5091/4 handle
null
//https://neo4j.com/developer/kb/conditional-cypher-execution/
foreach(ignoreMe in case when exists(line.shape) then [1] else [] END |
merge(shapes:Shape{shape:toUpper(trim(line.shape))}) // changed toupper
merge(ufo)-[:WITH_SHAPE]->(shapes)
)
//windir
merge(wdire:WindDirection{direction:line.wdire})
merge (ufo)-[:WITH_WINDDIRECTION]->(wdire)

//weather condition
//conds
foreach(ignoreMe in case when exists(line.conds) then [1] else [] END |
merge(conds:WeatherCondition{weather:line.conds})
merge(ufo)-[:WITH_WEATHER]->(conds)
)

//location
merge(address:Address
{city:trim(toUpper(line.city)),state:trim(toUpper(line.state))})
merge(ufo)-[:SPOTTED_ADDRESS]->(address);


//link ufo to Geo location

match(ufo:UfoInfo)-[s:SPOTTED_ADDRESS]->(address:Address)
match (ci:City) -[:LOCATE_IN]-> (co:County)-[:LOCATE_AT]->(st:State)
match (ci:City{cityName:address.city}) -[:LOCATE_IN]-> (co:County)-
[:LOCATE_AT]->(st:State {stateName:address.state})
create (ufo) -[:IS_WITNESSED]-> (ci);
```



Displaying 18 nodes, 17 relationships.

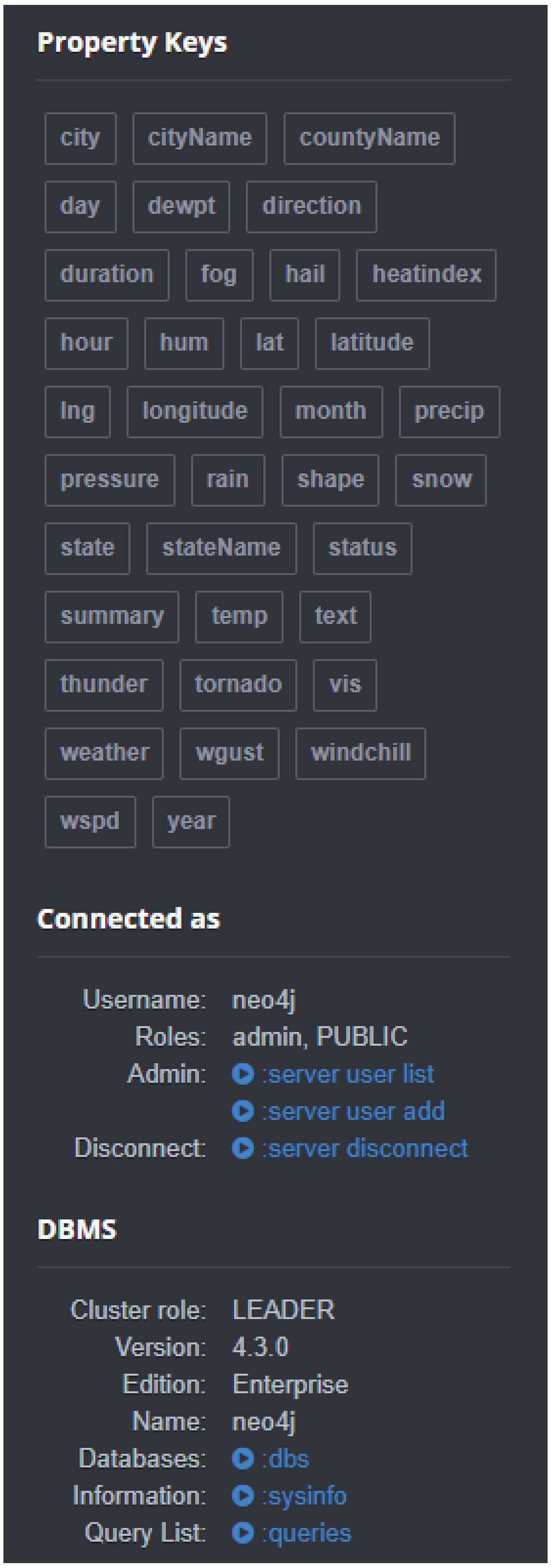


Node Labels

*(33,423) Address City County Day Fog Hail Hour Month Rain Shape Snow State Thunder Tornado UfoInfo WeatherCondition WindDirection Year

Relationship Types

*(56,722) DAY_SPOTTED HAS_FOG HAS_HAIL HAS_RAIN HAS_SNOW HAS_THUNDER HAS_TORNADO HOUR_SPOTTED IS_WITNESSED LOCATE_AT LOCATE_IN MONTH_SPOTTED SPOTTED_ADDRESS WITH_SHAPE WITH_WEATHER WITH_WINDDIRECTION YEAR_SPOTTED



Property Keys

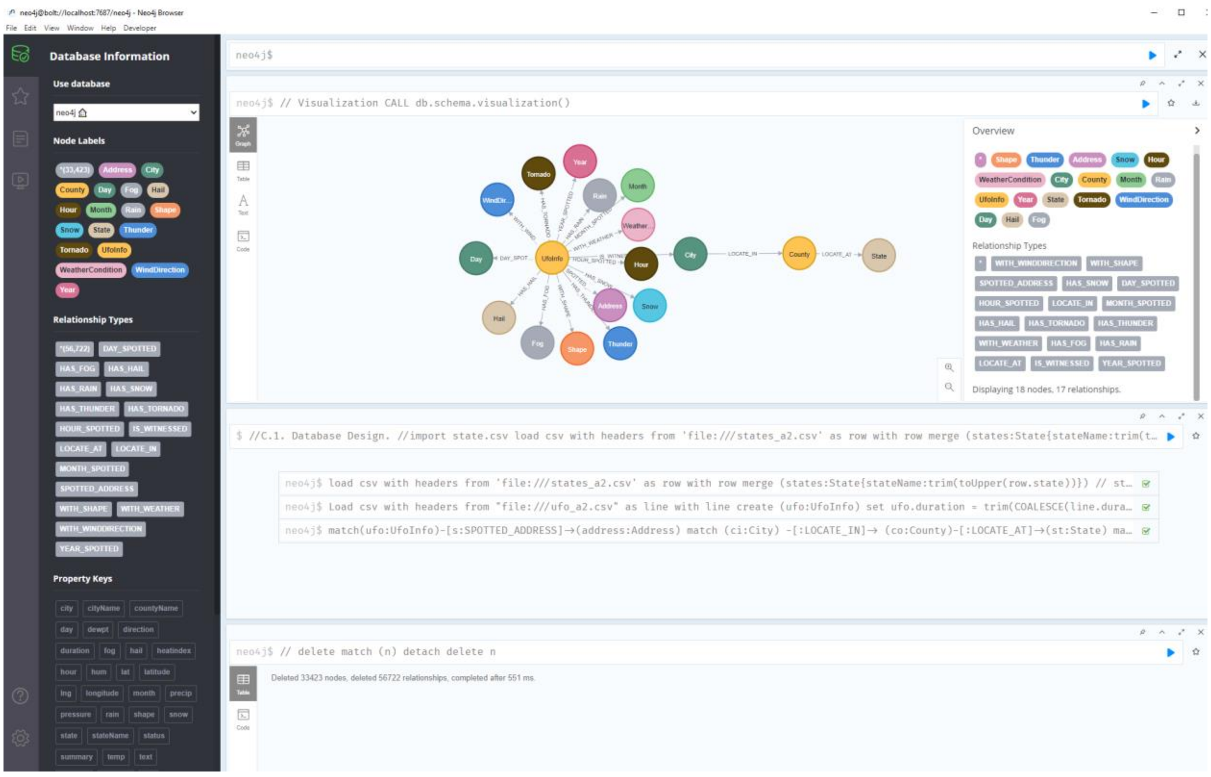
city cityName countyName
day dewpt direction
duration fog hail heatindex
hour hum lat latitude
lng longitude month precip
pressure rain shape snow
state stateName status
summary temp text
thunder tornado vis
weather wgust windchill
wspd year

Connected as

Username:	neo4j
Roles:	admin, PUBLIC
Admin:	:server user list
	:server user add
Disconnect:	:server disconnect

DBMS

Cluster role:	LEADER
Version:	4.3.0
Edition:	Enterprise
Name:	neo4j
Databases:	:dbs
Information:	:sysinfo
Query List:	:queries


neo4j@bolt://localhost:7687/neo4j - Neo4j Browser
File Edit View Window Help Developer
Database Information
Use database
Node Labels
Relationship Types
Property Keys
Overview
Displaying 18 nodes, 17 relationships.

```
neo4j$ // Visualization CALL db.schema.visualization()
```

```
$ //C.1. Database Design. //import state.csv load csv with headers from 'file:///states_a2.csv' as row with row merge (states:State{stateName:trim(t…
neo4j$ load csv with headers from 'file:///states_a2.csv' as row with row merge (states:State{stateName:trim(toUpper(row.state))}) // st…
neo4j$ Load csv with headers from "file:///ufo_a2.csv" as line with line create (ufo:UfoInfo) set ufo.duration = trim(COALESCE(line.dura…
neo4j$ match(ufo:UfoInfo)-[s:SPOTTED_ADDRESS]→(address:Address) match (ci:City) -[:LOCATE_IN]→ (co:County)-[:LOCATE_AT]→(st:State) ma…
```

```
neo4j$ // delete match (n) detach delete n
Deleted 33423 nodes, deleted 56722 relationships, completed after 551 ms.
```

Explaination on graph design

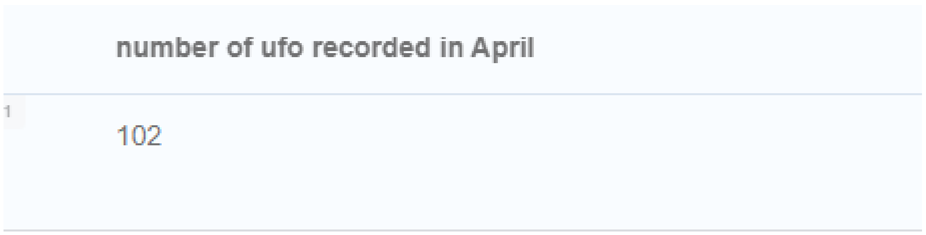
I designed this graph to minimize data duplication issue and linked all nodes directly or indirectly to UfoInfo node which can help me to answer all queries required in Task2. There are 4 parts, the Time part are nodes that represent time properties which are Year, Month, Day, Hour. Those will help me to match and search time related information for the observed ufo. The Address and location part can help to located where the ufo was observed. The boolean part, like Hail, Snow, Thunder, Fog, Tornado, Rain can help me to answer whether the special environmental condition exist or not. The wind and weather part can help me to illustrate the wind direction and weather condition when the ufo was spotted. UfoInfo node contains numeric information and text information about each observation of the Ufo. (136words)

C.2. Queries.

I created two indices for this part and specifically for question 4 and 6. In question 4, I need to search for text that contain ‘at high speeds’. Create a fulltext index on the field text will allow me to perform the search quickly. According to the neo4j manual “A full-text index allows you to write queries that match within the *contents* of indexed string properties." In question 6, I created a composite index for both properties, the latitude and longitude. It helps faster retrieval operation on database when I query and compare the distance for those geo-locations.

1.

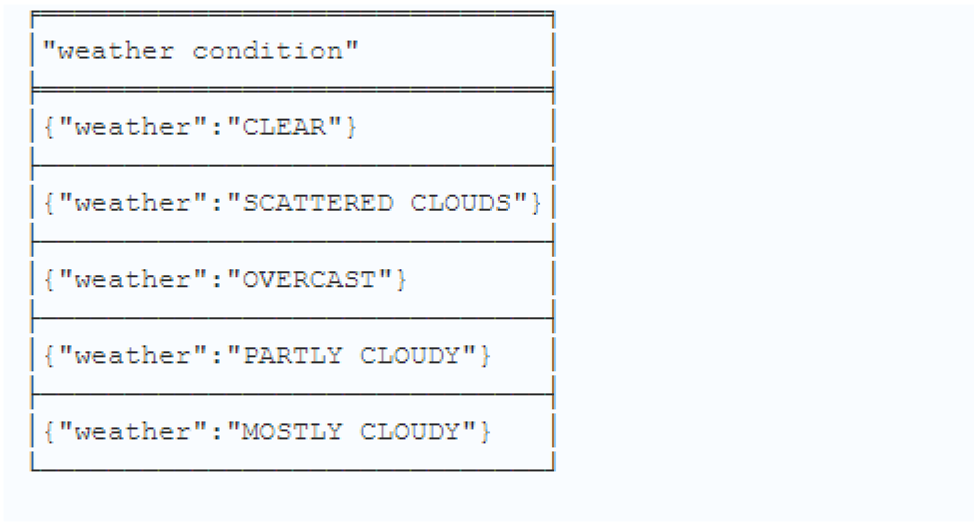
```
match (ufo:UfoInfo) -[:MONTH_SPOTTED]-> (months:Month)
where months.month = 4
return count(ufo) as `number of ufo recorded in April`;
```



	number of ufo recorded in April
1	102

2.

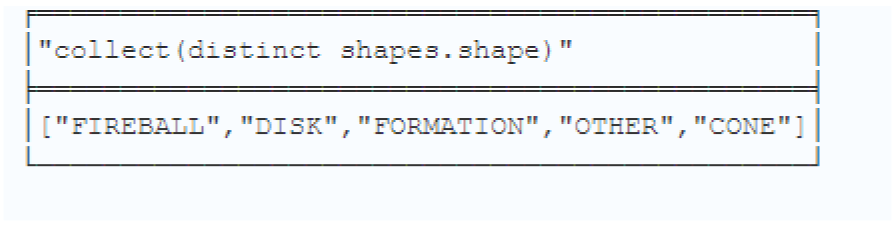
```
match(shapes:Shape{shape:'CIRCLE'})<-[:WITH_SHAPE]-(ufo:UfoInfo)-
[:WITH_WEATHER]->(weather:WeatherCondition),
(years:Year)<-[:YEAR_SPOTTED]-(ufo)-[:IS_WITNESSED]->(ci:City)-
[:LOCATE_IN]->(co:County)-[:LOCATE_AT]->(st:State{stateName:'AZ'})
where years.year<2014
return distinct weather as `weather condition`;
```



```
╒════════════════════════════════╕
│"weather condition"             │
╞════════════════════════════════╡
│{"weather":"CLEAR"}             │
├────────────────────────────────┤
│{"weather":"SCATTERED CLOUDS"}  │
├────────────────────────────────┤
│{"weather":"OVERCAST"}          │
├────────────────────────────────┤
│{"weather":"PARTLY CLOUDY"}     │
├────────────────────────────────┤
│{"weather":"MOSTLY CLOUDY"}     │
└────────────────────────────────┘
```

3.

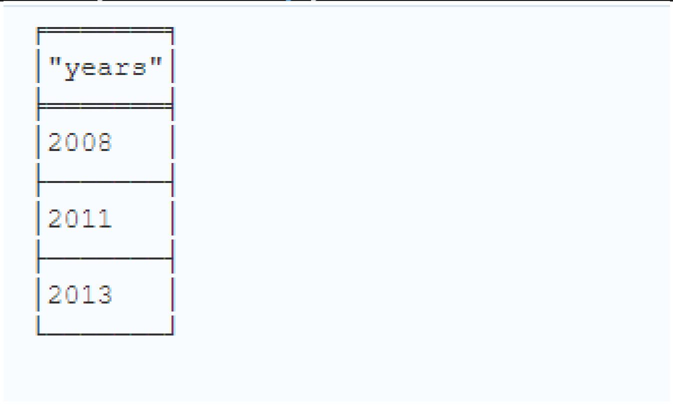
```
MATCH (shapes:Shape)<-[:WITH_SHAPE]-(ufo:UfoInfo)-
[:YEAR_SPOTTED]->(years:Year)
WITH collect(years.year) as years, shapes
WHERE NOT 2000 in years AND 2015 in years
RETURN collect(distinct shapes.shape);
```



```
╒══════════════════════════════════════════════╕
│"collect(distinct shapes.shape)"              │
╞══════════════════════════════════════════════╡
│["FIREBALL","DISK","FORMATION","OTHER","CONE"]│
└──────────────────────────────────────────────┘
```

4.

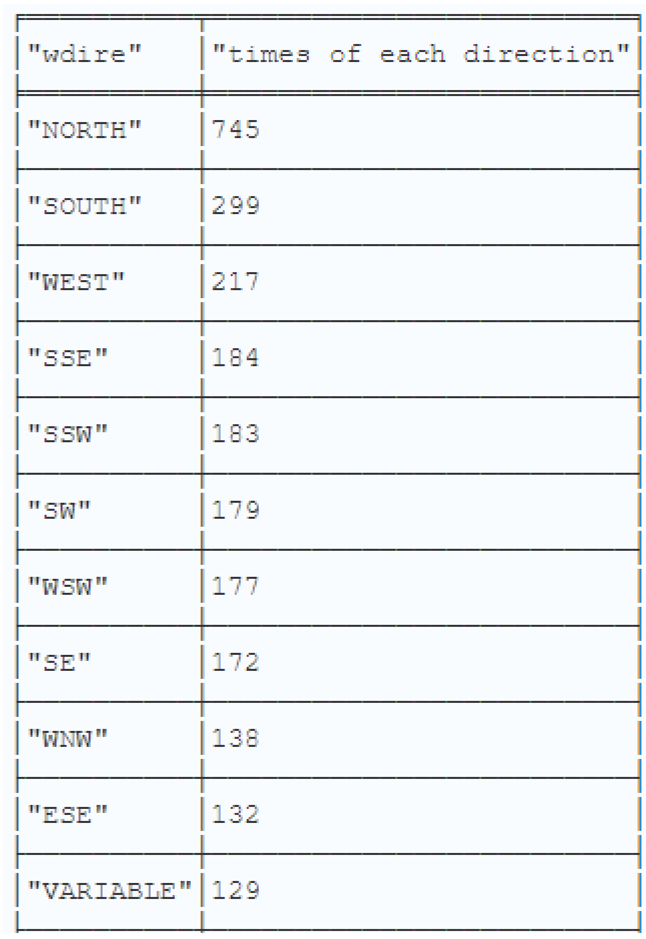
```
CREATE FULLTEXT INDEX textIndex FOR (n:UfoInfo) ON EACH [n.text];

CALL db.index.fulltext.queryNodes("textIndex", "'at high speeds'") YIELD node,
score
match (node)-->(y:Year)
with distinct y.year as `years`
return years order by years ASC
```



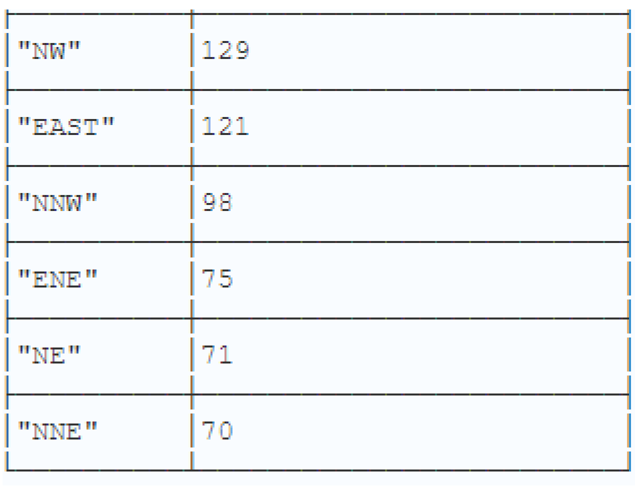
```
╒═══════╕
│"years"│
╞═══════╡
│2008   │
├───────┤
│2011   │
├───────┤
│2013   │
└───────┘
```

5.

```
match (years:Year)<-[:YEAR_SPOTTED]-(u:UfoInfo)-
[:WITH_WINDDIRECTION]->(wdire:WindDirection)
with wdire, count(*) as `times of each direction`
order by `times of each direction` desc
return wdire.direction as wdire,`times of each direction`;
```



"wdire"	"times of each direction"
"NORTH"	745
"SOUTH"	299
"WEST"	217
"SSE"	184
"SSW"	183
"SW"	179
"WSW"	177
"SE"	172
"WNW"	138
"ESE"	132
"VARIABLE"	129



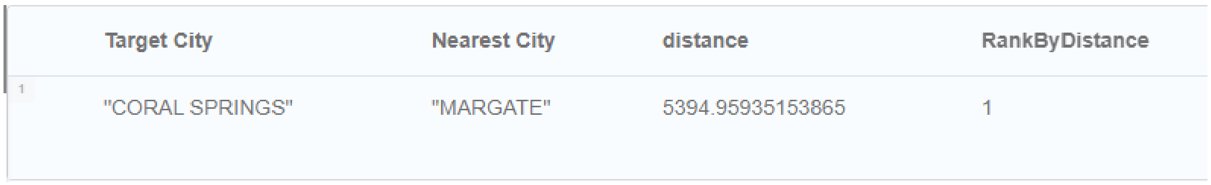
"NW"	129
"EAST"	121
"NNW"	98
"ENE"	75
"NE"	71
"NNE"	70

6.

```
create index city_location_index for (c:City) on (c.latitude,c.longitude);
```

```
match (city:City{cityName:'CORAL SPRINGS'}) -
->(counties:County{countyName:'BROWARD'}) -->(states:State{stateName:'FL'})
with city, point({longitude:city.longitude,latitude:city.latitude}) as location1
match (citydup:City)
with location1,city,citydup,point({longitude:citydup.longitude,latitude:citydup.latitude})
 as location2
```

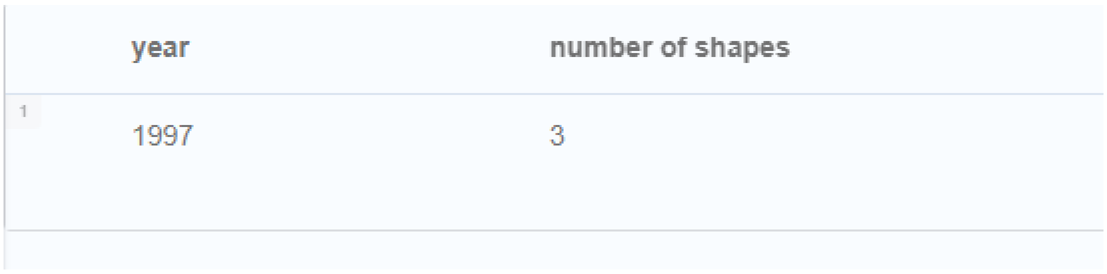
```
where location1 <> location2
with city.cityName as place1, citydup.cityName as place2, distance(location1,locatio
n2) as distance
order by distance asc
limit 1
with place1, collect(place2) as pl2, distance
unwind pl2 as place
return place1 as `Target City`, place as `Nearest City`, distance, apoc.coll.indexOf(pl
2,place) + 1 as RankByDistance;
```



	Target City	Nearest City	distance	RankByDistance
1	"CORAL SPRINGS"	"MARGATE"	5394.95935153865	1

7.

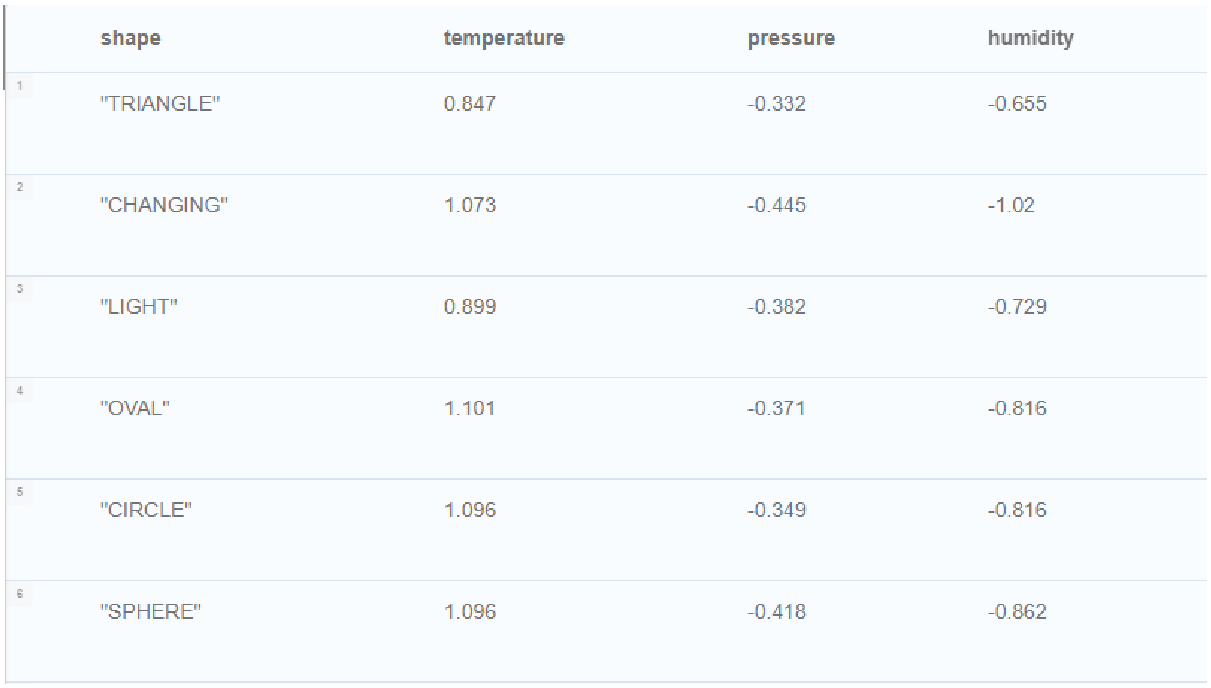
```
match (y:Year) <--(u:UfoInfo)-->(s:Shape)
with y,count(distinct s.shape) as number
order by number asc
limit 1
return y.year as year, number as `number of shapes`;
```



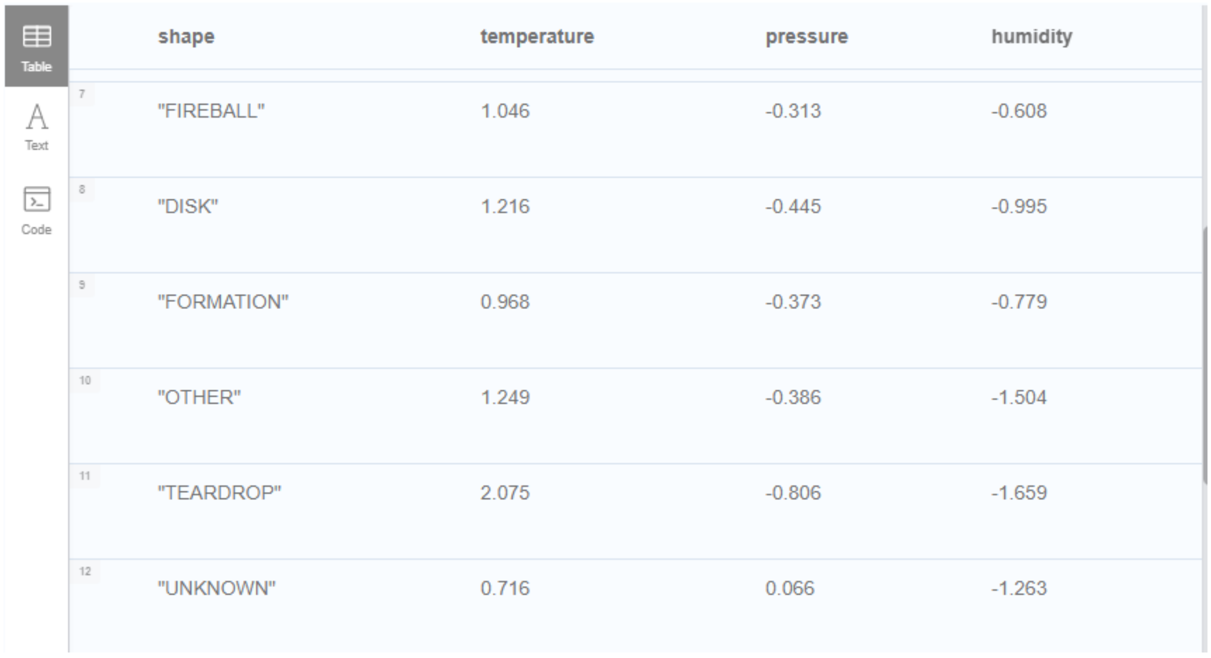
	year	number of shapes
1	1997	3

8.

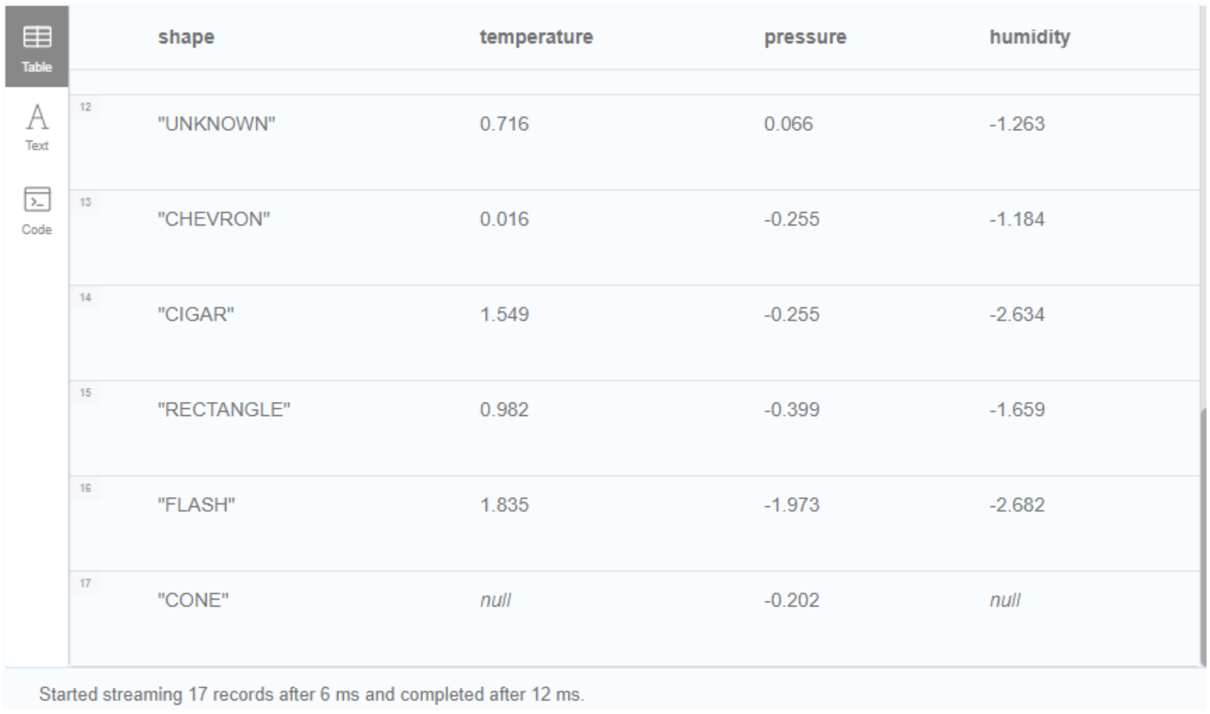
```
match (u:UfoInfo) --> (s:Shape)
with s.shape as shape, round(avg(u.temp),3) as temperature, round(avg(u.pressure),3) as press
ure, round(avg(u.hum),3) as humidity
return shape, temperature, pressure, humidity;
```



	shape	temperature	pressure	humidity
1	"TRIANGLE"	0.847	-0.332	-0.655
2	"CHANGING"	1.073	-0.445	-1.02
3	"LIGHT"	0.899	-0.382	-0.729
4	"OVAL"	1.101	-0.371	-0.816
5	"CIRCLE"	1.096	-0.349	-0.816
6	"SPHERE"	1.096	-0.418	-0.862



	shape	temperature	pressure	humidity
7	"FIREBALL"	1.046	-0.313	-0.608
8	"DISK"	1.216	-0.445	-0.995
9	"FORMATION"	0.968	-0.373	-0.779
10	"OTHER"	1.249	-0.386	-1.504
11	"TEARDROP"	2.075	-0.806	-1.659
12	"UNKNOWN"	0.716	0.066	-1.263

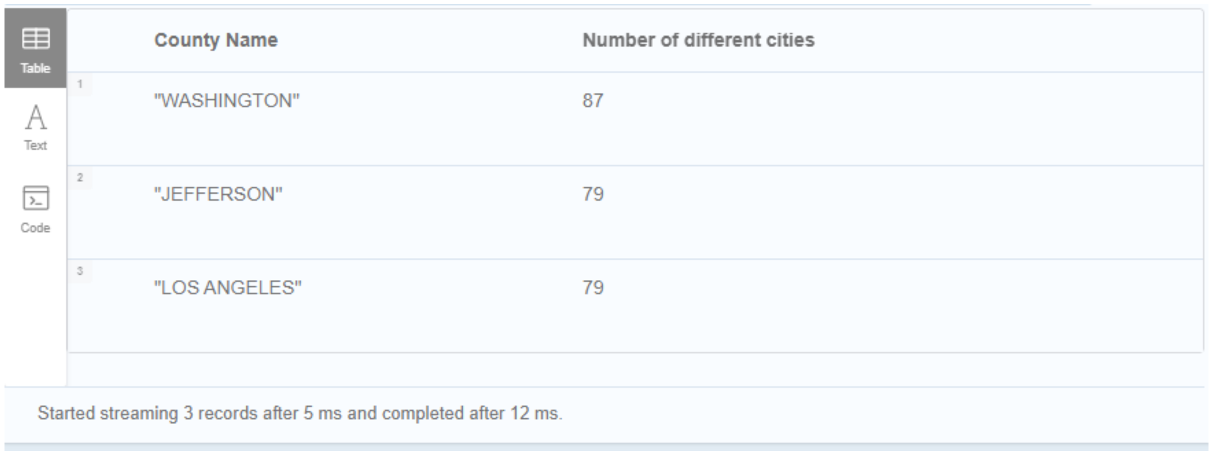


	shape	temperature	pressure	humidity
12	"UNKNOWN"	0.716	0.066	-1.263
13	"CHEVRON"	0.016	-0.255	-1.184
14	"CIGAR"	1.549	-0.255	-2.634
15	"RECTANGLE"	0.982	-0.399	-1.659
16	"FLASH"	1.835	-1.973	-2.682
17	"CONE"	*null*	-0.202	*null*

Started streaming 17 records after 6 ms and completed after 12 ms.

9.

```
match (c:County) <--(ci:City)
with c, count(ci.cityName) as number
order by number desc
limit 3
return c.countyName as `County Name`, number as `Number of different cities`;
```

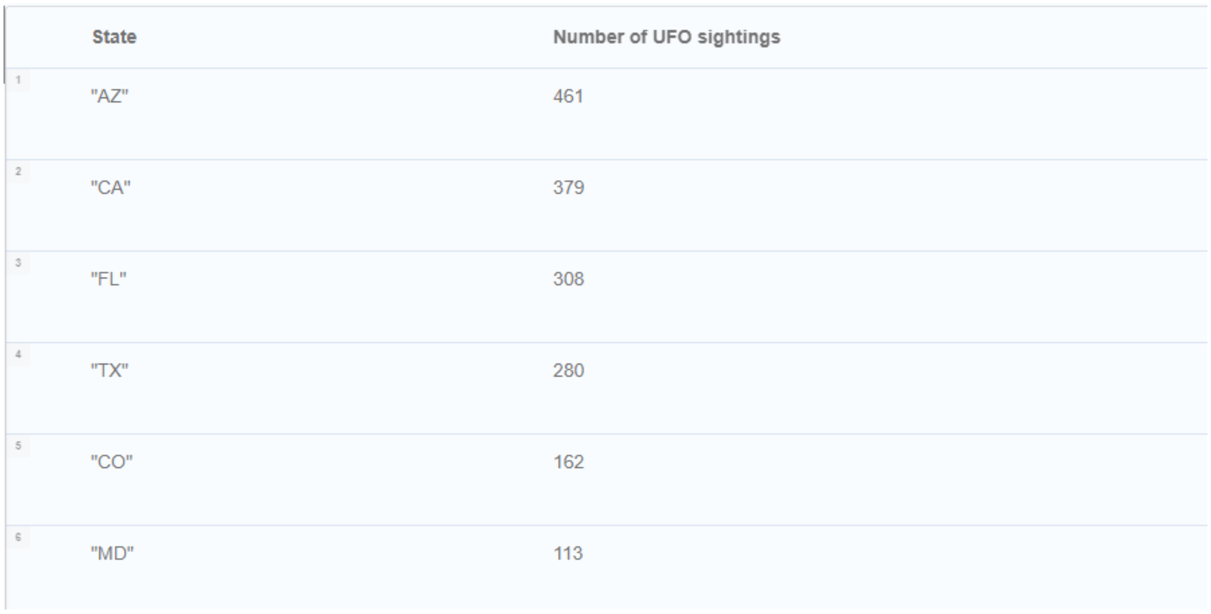


	County Name	Number of different cities
1	"WASHINGTON"	87
2	"JEFFERSON"	79
3	"LOS ANGELES"	79

Started streaming 3 records after 5 ms and completed after 12 ms.

10.

```
match (u:UfoInfo)-->(ci:City)-->(co:County)-->(s:State)
with s, count(u) as number
order by number desc
return s.stateName as State, number as `Number of UFO sightings`;
```

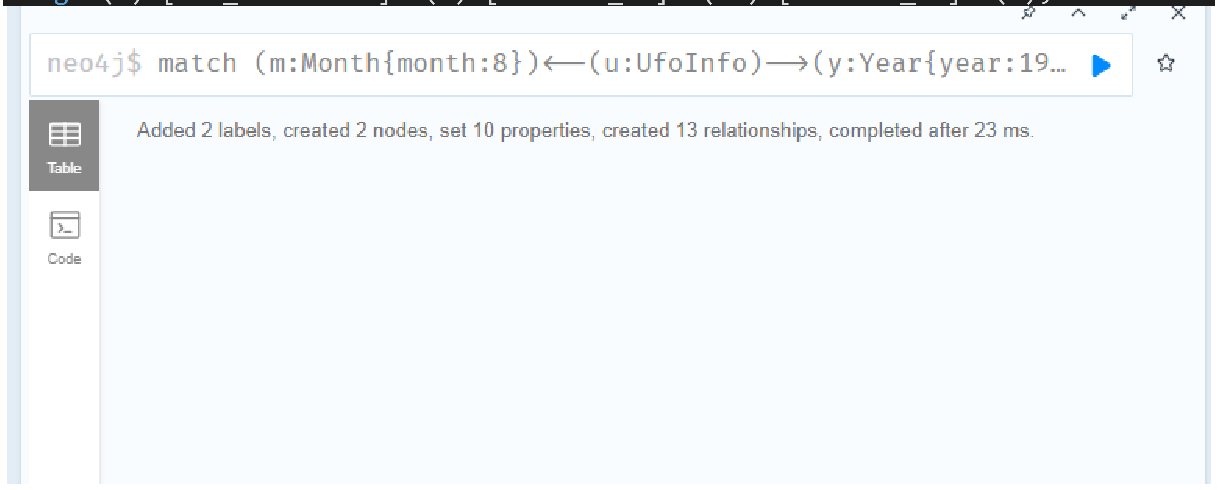


	State	Number of UFO sightings
1	"AZ"	461
2	"CA"	379
3	"FL"	308
4	"TX"	280
5	"CO"	162
6	"MD"	113

C.3. Database Modifications.

1.

```
match (m:Month{month:8})<--(u:UfoInfo)-->(y:Year{year:1998}),
(d:Day{day:14})<--(u)-->(h:Hour{hour:16}), (rain:Rain)<--(u)-->(hail:Hail),
(thunder:Thunder)<--(u)-->(fog:Fog),
(tornado:Tornado)<--(u)-->(snow:Snow)
with u,rain,hail,thunder,fog,tornado,snow
create (n:UfoInfo{duration:'25 minutes', text:"Awesome lights were seen in the
sky",summary:"Awesome lights",
        pressure:u.pressure, temp:u.temp,
heatindex:u.heatindex,windchill:u.windchill,
        vis:u.vis, dewpt:u.dewpt, precip:u.precip, wspd:u.wspd,
hum:u.hum,wgust:u.wgust
        })
with u,rain,hail,thunder,fog,tornado,snow,n
merge (y:Year{year:2021})
merge (m:Month{month:1})
merge (d:Day{day:14})
merge (h:Hour{hour:23})
merge (n)-[:YEAR_SPOTTED]->(y)
merge (n)-[:MONTH_SPOTTED]->(m)
merge (n)-[:DAY_SPOTTED]->(d)
merge (n)-[:HOUR_SPOTTED]->(h)
merge (n)-[:HAS_HAIL]->(hail)
merge (n)-[:HAS_RAIN]->(rain)
merge (n)-[:HAS_THUNDER]->(thunder)
merge (n)-[:HAS_FOG]->(fog)
merge (n)-[:HAS_TORNADO]->(tornado)
merge (n)-[:HAS_SNOW]->(snow)
with n
match (c:City{cityName:'HIGHLAND'})-
[:LOCATE_IN]->(co:County{countyName:'LAKE'})-
[:LOCATE_AT]->(s:State{stateName:'IN'})
merge (n)-[:IS_WITNESSED]->(c)-[:LOCATE_IN]->(co)-[:LOCATE_AT]->(s);
```



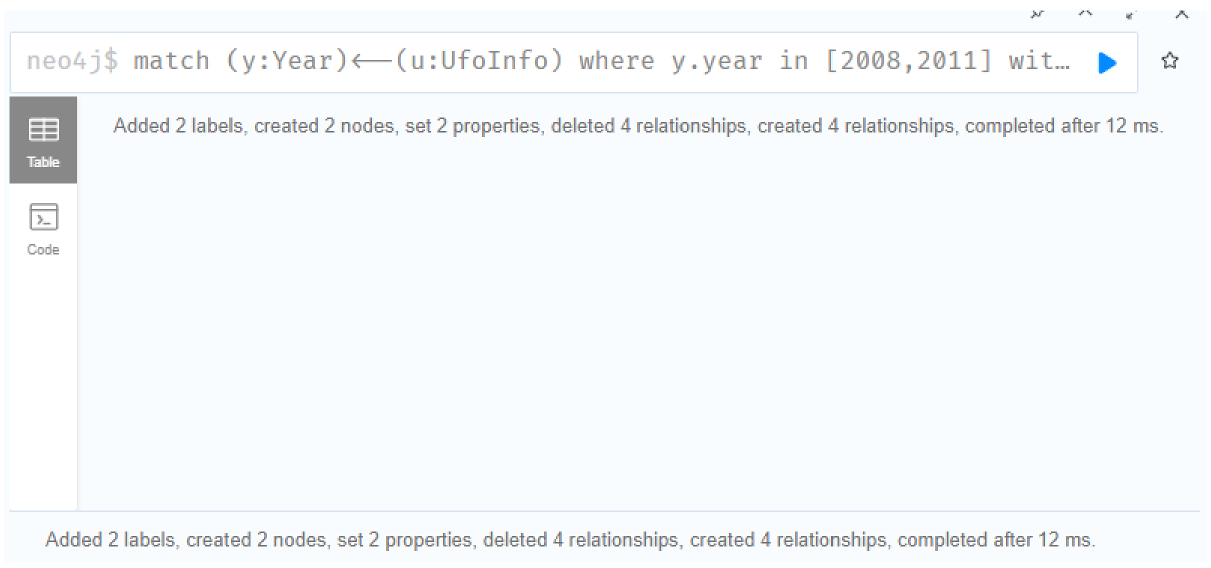
```
neo4j$ match (m:Month{month:8})⟵(u:UfoInfo)⟶(y:Year{year:19…
```

Added 2 labels, created 2 nodes, set 10 properties, created 13 relationships, completed after 23 ms.

2.

```
match (y:Year)<--(u:UfoInfo)
where y.year in [2008,2011]
with u
match (u)-[r:WITH_SHAPE]->(s:Shape{shape:'UNKNOWN'})
delete r
with u
merge (newshape:Shape{shape:'FLYING SAUCER'})
merge (u)-[:WITH_SHAPE]->(newshape)
with u
match (u)-[r1:WITH_WEATHER]->(conds:WeatherCondition{weather:'CLEAR'})
delete r1
with u
merge (conds1:WeatherCondition{weather:'SUNNY/CLEAR'})
merge (u)-[:WITH_WEATHER]->(conds1);
```

3.

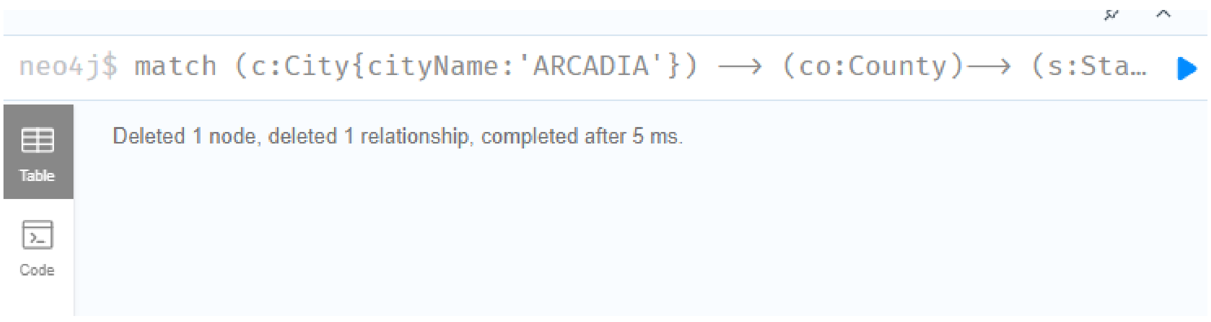

```
neo4j$ match (y:Year)⟵(u:UfoInfo) where y.year in [2008,2011] wit…
```

Added 2 labels, created 2 nodes, set 2 properties, deleted 4 relationships, created 4 relationships, completed after 12 ms.

Added 2 labels, created 2 nodes, set 2 properties, deleted 4 relationships, created 4 relationships, completed after 12 ms.

3.

```
match (c:City{cityName:'ARCADIA'}) --> (co:County)--> (s:State{stateName:'FL'})
detach delete c;
```


```
neo4j$ match (c:City{cityName:'ARCADIA'}) ⟶ (co:County)⟶ (s:Sta…
```

Deleted 1 node, deleted 1 relationship, completed after 5 ms.