<u>Assignment 5</u>

Neo4j

Biraj Parikh

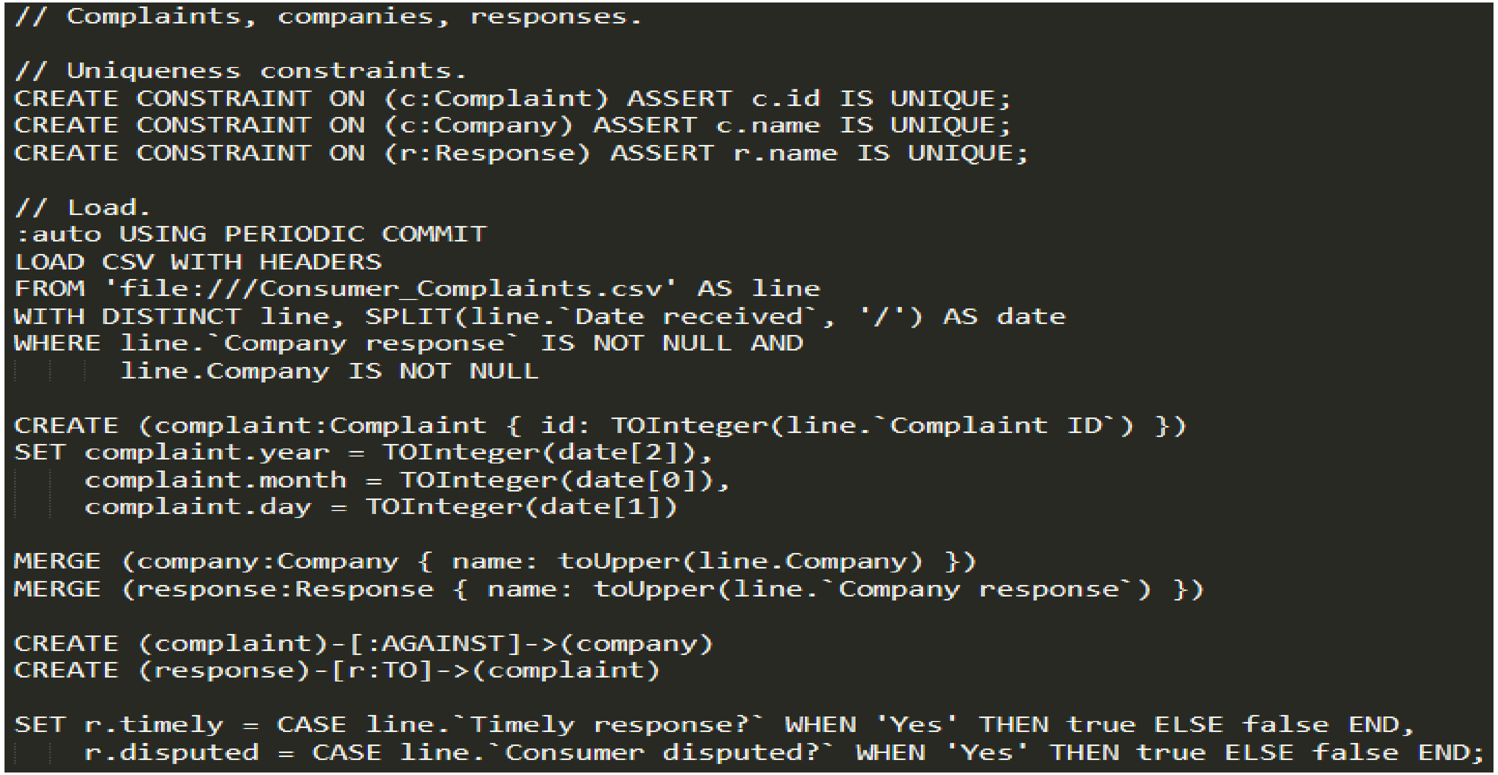
About the data:

Description:

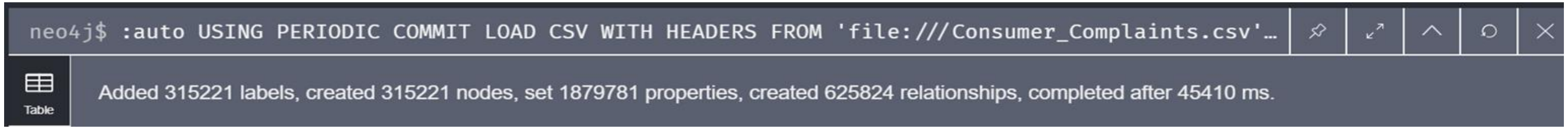
The Consumer Complaint Database is a collection of complaints about consumer financial products and services that we sent to companies for response. Complaints are published after the company responds, confirming a commercial relationship with the consumer, or after 15 days, whichever comes first.

Data Source: https://catalog.data.gov/dataset/consumer-complaint-database

Loading the data

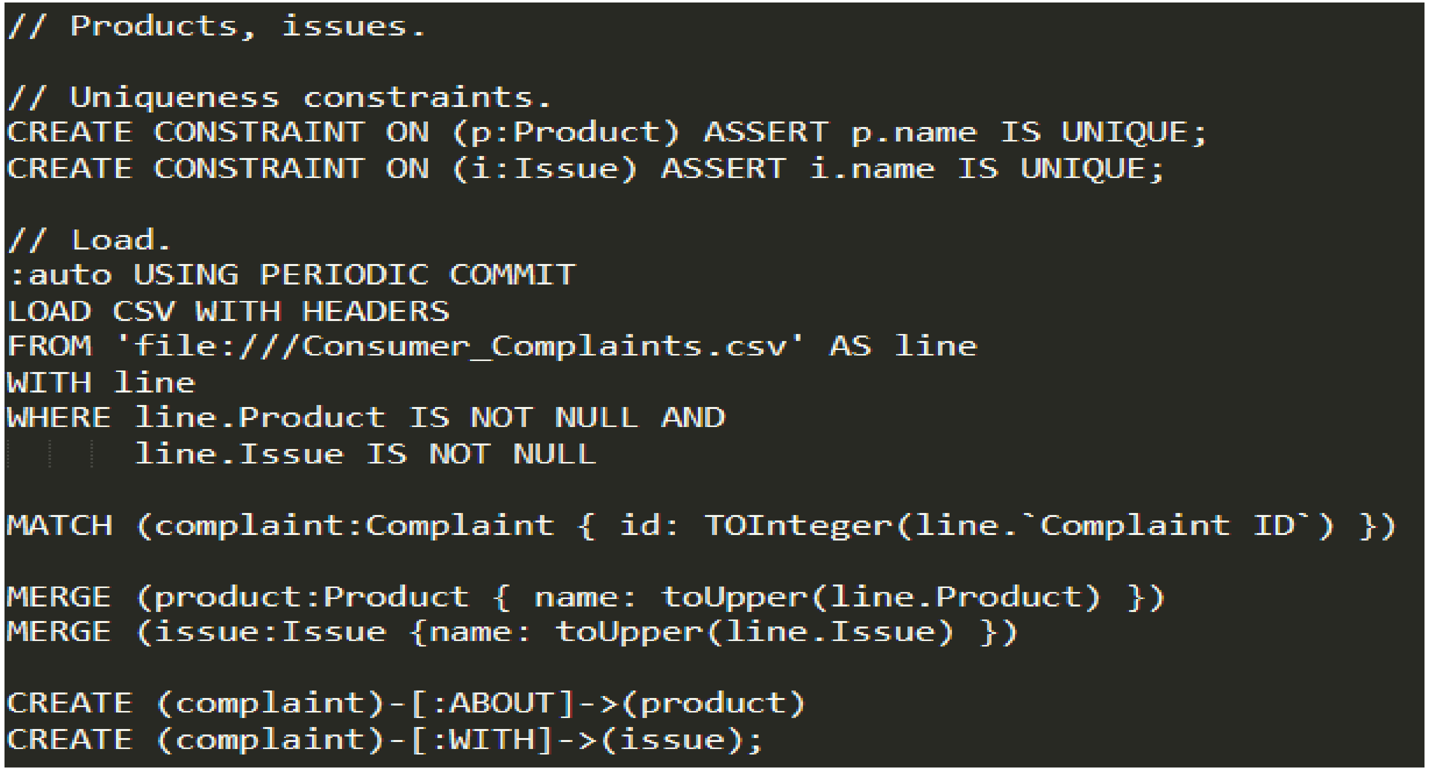


```
// Complaints, companies, responses.

// Uniqueness constraints.
CREATE CONSTRAINT ON (c:Complaint) ASSERT c.id IS UNIQUE;
CREATE CONSTRAINT ON (c:Company) ASSERT c.name IS UNIQUE;
CREATE CONSTRAINT ON (r:Response) ASSERT r.name IS UNIQUE;

// Load.
:auto USING PERIODIC COMMIT
LOAD CSV WITH HEADERS
FROM 'file:///Consumer_Complaints.csv' AS line
WITH DISTINCT line, SPLIT(line.`Date received`, '/') AS date
WHERE line.`Company response` IS NOT NULL AND
      line.Company IS NOT NULL

CREATE (complaint:Complaint { id: TOInteger(line.`Complaint ID`) })
SET complaint.year = TOInteger(date[2]),
    complaint.month = TOInteger(date[0]),
    complaint.day = TOInteger(date[1])

MERGE (company:Company { name: toUpper(line.Company) })
MERGE (response:Response { name: toUpper(line.`Company response`) })

CREATE (complaint)-[:AGAINST]->(company)
CREATE (response)-[r:TO]->(complaint)

SET r.timely = CASE line.`Timely response?` WHEN 'Yes' THEN true ELSE false END,
    r.disputed = CASE line.`Consumer disputed?` WHEN 'Yes' THEN true ELSE false END;
```

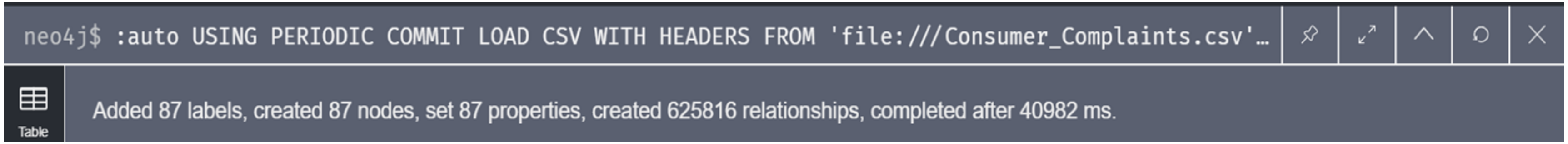


```
neo4j$ :auto USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM 'file:///Consumer_Complaints.csv'…
```

```
Added 315221 labels, created 315221 nodes, set 1879781 properties, created 625824 relationships, completed after 45410 ms.
```

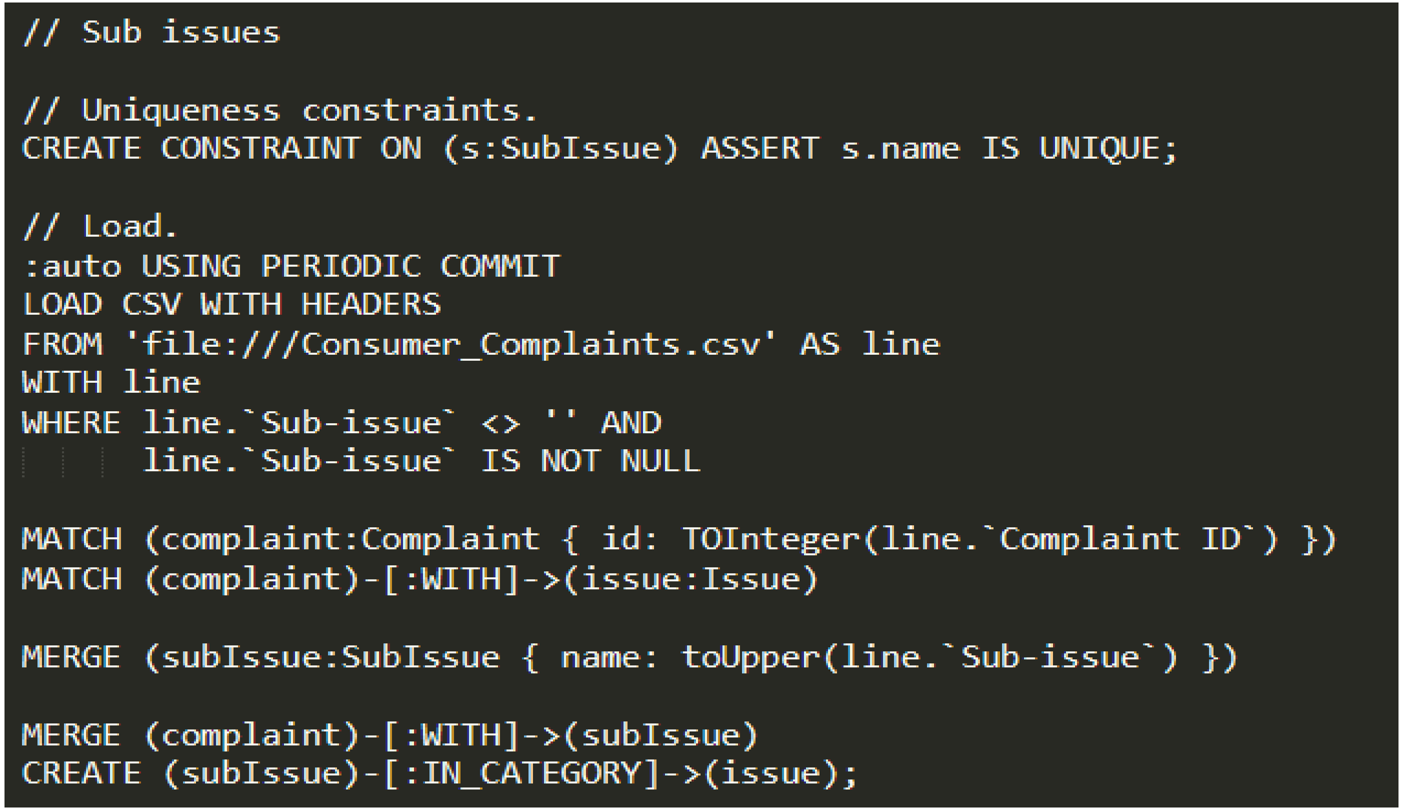


```
// Products, issues.

// Uniqueness constraints.
CREATE CONSTRAINT ON (p:Product) ASSERT p.name IS UNIQUE;
CREATE CONSTRAINT ON (i:Issue) ASSERT i.name IS UNIQUE;

// Load.
:auto USING PERIODIC COMMIT
LOAD CSV WITH HEADERS
FROM 'file:///Consumer_Complaints.csv' AS line
WITH line
WHERE line.Product IS NOT NULL AND
      line.Issue IS NOT NULL

MATCH (complaint:Complaint { id: TOInteger(line.`Complaint ID`) })

MERGE (product:Product { name: toUpper(line.Product) })
MERGE (issue:Issue {name: toUpper(line.Issue) })

CREATE (complaint)-[:ABOUT]->(product)
CREATE (complaint)-[:WITH]->(issue);
```

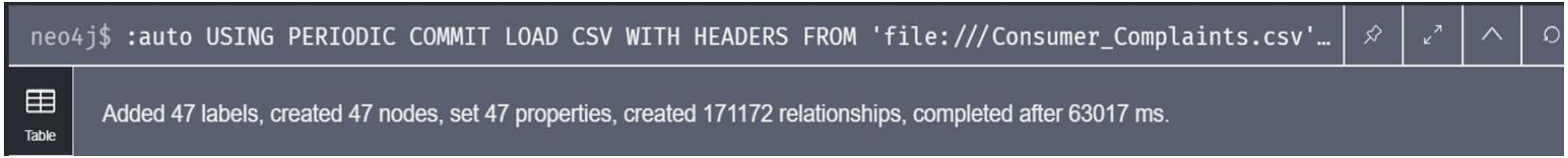


```
neo4j$ :auto USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM 'file:///Consumer_Complaints.csv'…
```

Added 87 labels, created 87 nodes, set 87 properties, created 625816 relationships, completed after 40982 ms.

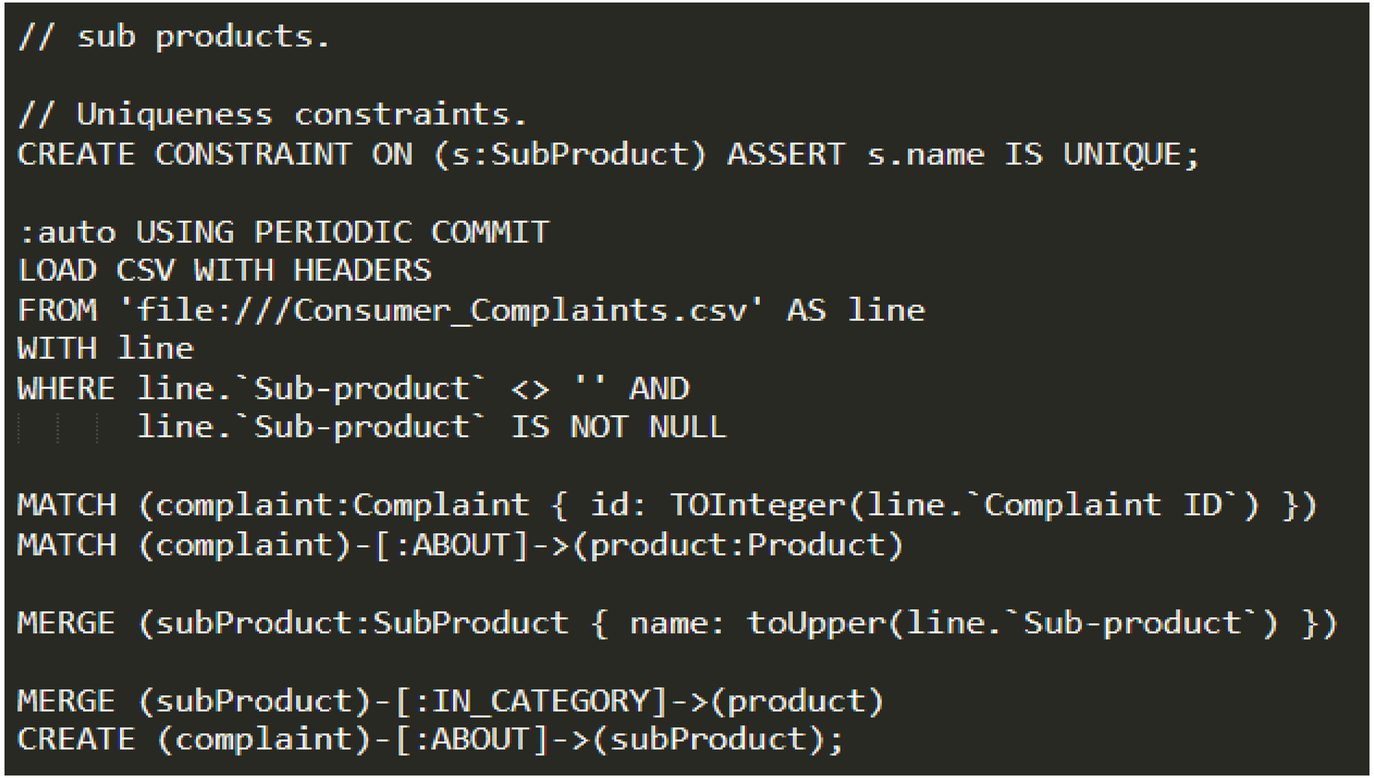


```
// Sub issues

// Uniqueness constraints.
CREATE CONSTRAINT ON (s:SubIssue) ASSERT s.name IS UNIQUE;

// Load.
:auto USING PERIODIC COMMIT
LOAD CSV WITH HEADERS
FROM 'file:///Consumer_Complaints.csv' AS line
WITH line
WHERE line.`Sub-issue` <> '' AND
      line.`Sub-issue` IS NOT NULL

MATCH (complaint:Complaint { id: TOInteger(line.`Complaint ID`) })
MATCH (complaint)-[:WITH]->(issue:Issue)

MERGE (subIssue:SubIssue { name: toUpper(line.`Sub-issue`) })

MERGE (complaint)-[:WITH]->(subIssue)
CREATE (subIssue)-[:IN_CATEGORY]->(issue);
```

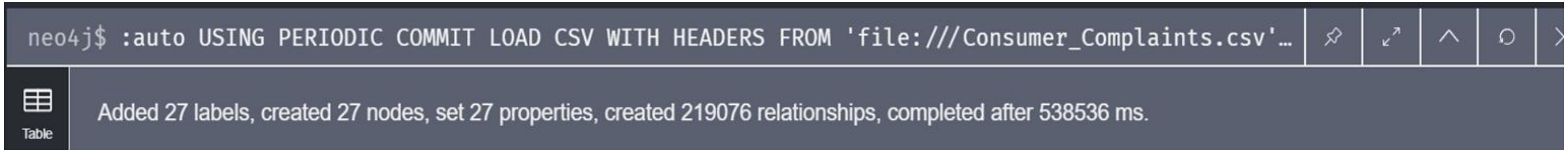


```
neo4j$ :auto USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM 'file:///Consumer_Complaints.csv'…
```

Added 47 labels, created 47 nodes, set 47 properties, created 171172 relationships, completed after 63017 ms.



```
// sub products.

// Uniqueness constraints.
CREATE CONSTRAINT ON (s:SubProduct) ASSERT s.name IS UNIQUE;

:auto USING PERIODIC COMMIT
LOAD CSV WITH HEADERS
FROM 'file:///Consumer_Complaints.csv' AS line
WITH line
WHERE line.`Sub-product` <> '' AND
      line.`Sub-product` IS NOT NULL

MATCH (complaint:Complaint { id: TOInteger(line.`Complaint ID`) })
MATCH (complaint)-[:ABOUT]->(product:Product)

MERGE (subProduct:SubProduct { name: toUpper(line.`Sub-product`) })

MERGE (subProduct)-[:IN_CATEGORY]->(product)
CREATE (complaint)-[:ABOUT]->(subProduct);
```

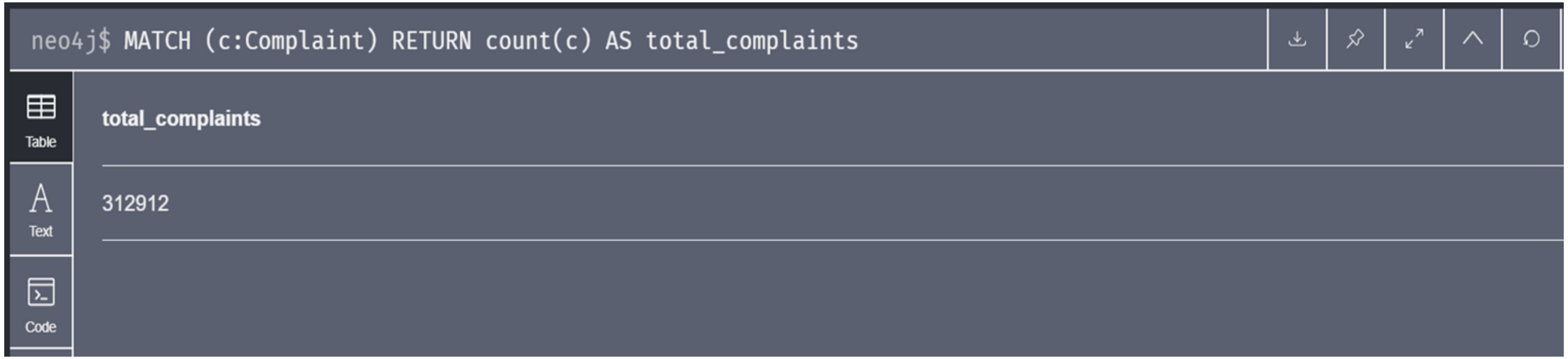


```
neo4j$ :auto USING PERIODIC COMMIT LOAD CSV WITH HEADERS FROM 'file:///Consumer_Complaints.csv'…
```

Added 27 labels, created 27 nodes, set 27 properties, created 219076 relationships, completed after 538536 ms.

1. ***Let's look at the total complaints***

```
// Total complaints
MATCH (c:Complaint)
RETURN count(c) AS total_complaints
```

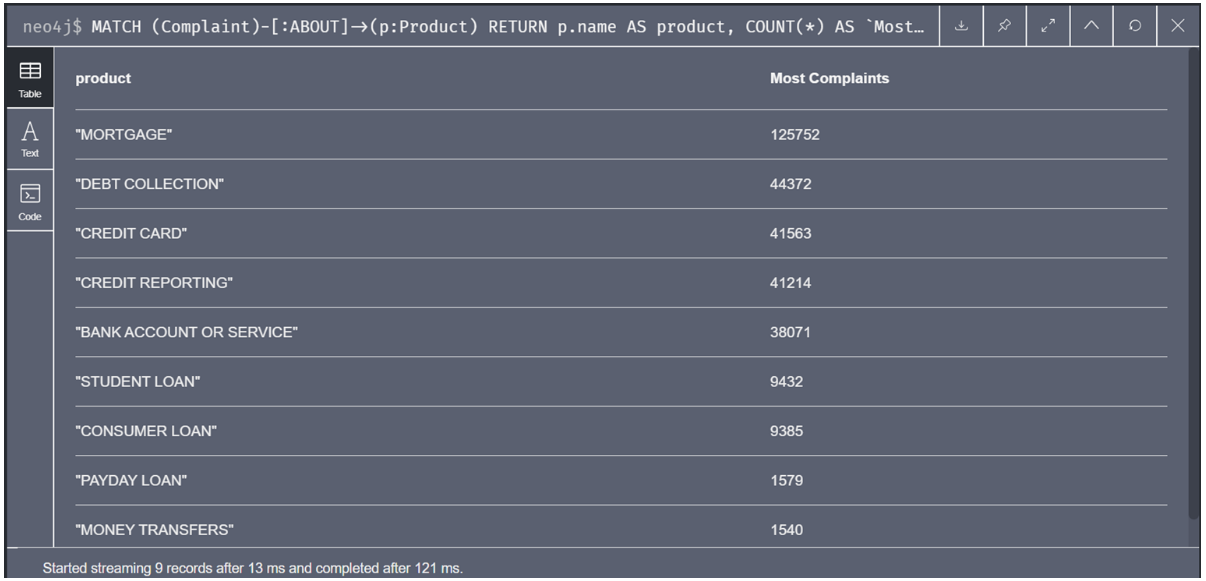


```
neo4j$ MATCH (c:Complaint) RETURN count(c) AS total_complaints
```

```
total_complaints
312912
```

2. Top product having most complaints

```
// Top product having most complaints
MATCH (Complaint)-[:ABOUT]->(p:Product)
RETURN p.name AS product, COUNT(*) AS `Most Complaints`
ORDER BY `Most Complaints` DESC
LIMIT 25;
```



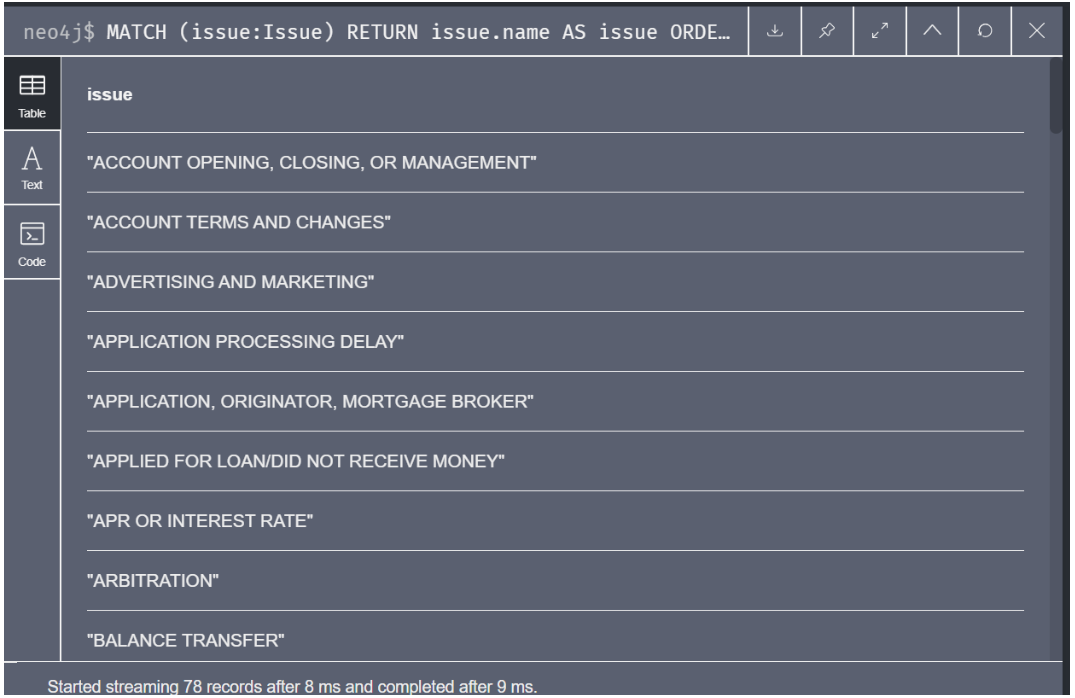
```
neo4j$ MATCH (Complaint)-[:ABOUT]→(p:Product) RETURN p.name AS product, COUNT(*) AS `Most…
```

product	Most Complaints
"MORTGAGE"	125752
"DEBT COLLECTION"	44372
"CREDIT CARD"	41563
"CREDIT REPORTING"	41214
"BANK ACCOUNT OR SERVICE"	38071
"STUDENT LOAN"	9432
"CONSUMER LOAN"	9385
"PAYDAY LOAN"	1579
"MONEY TRANSFERS"	1540

Started streaming 9 records after 13 ms and completed after 121 ms.

3. Finding all issues

```
// All issues.
MATCH (issue:Issue)
RETURN issue.name AS issue
ORDER BY issue;
```



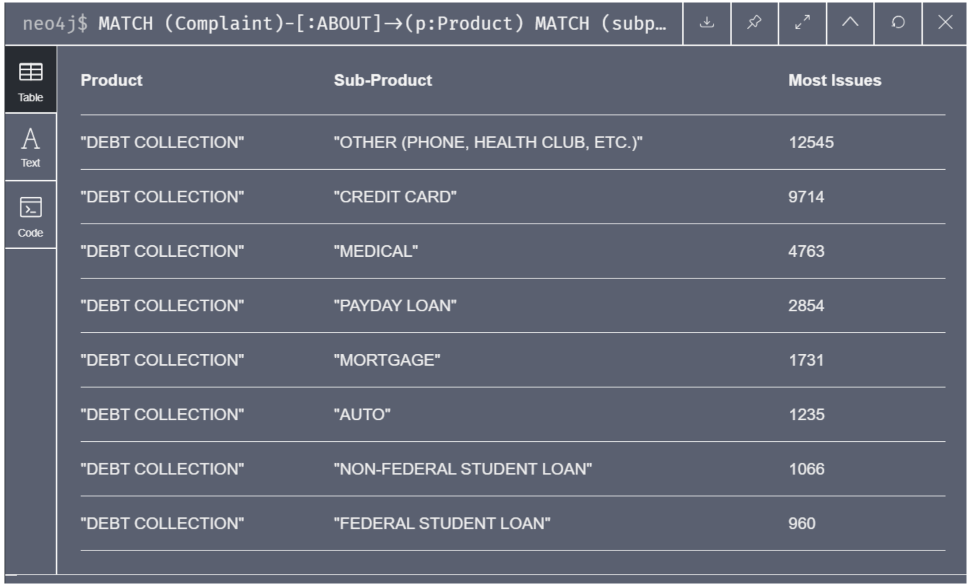
```
neo4j$ MATCH (issue:Issue) RETURN issue.name AS issue ORDE…
```

```
issue
"ACCOUNT OPENING, CLOSING, OR MANAGEMENT"
"ACCOUNT TERMS AND CHANGES"
"ADVERTISING AND MARKETING"
"APPLICATION PROCESSING DELAY"
"APPLICATION, ORIGINATOR, MORTGAGE BROKER"
"APPLIED FOR LOAN/DID NOT RECEIVE MONEY"
"APR OR INTEREST RATE"
"ARBITRATION"
"BALANCE TRANSFER"
```

Started streaming 78 records after 8 ms and completed after 9 ms.

4. Finding Products having Sub-Product with most issues

```
// Finding product having sub-products with most issues
MATCH (Complaint)-[:ABOUT]->(p:Product)
MATCH (subproduct:SubProduct)-[:IN_CATEGORY]->(p:Product)
MATCH (Complaint)-[:ABOUT]->(subproduct:subProduct)
MATCH (Complaint)-[:WITH]->(subissue:SubIssue)
RETURN p.name AS Product, subproduct.name AS `Sub-Product`, COUNT(*) AS `Most Issues`
ORDER BY `Most Issues` DESC;
```

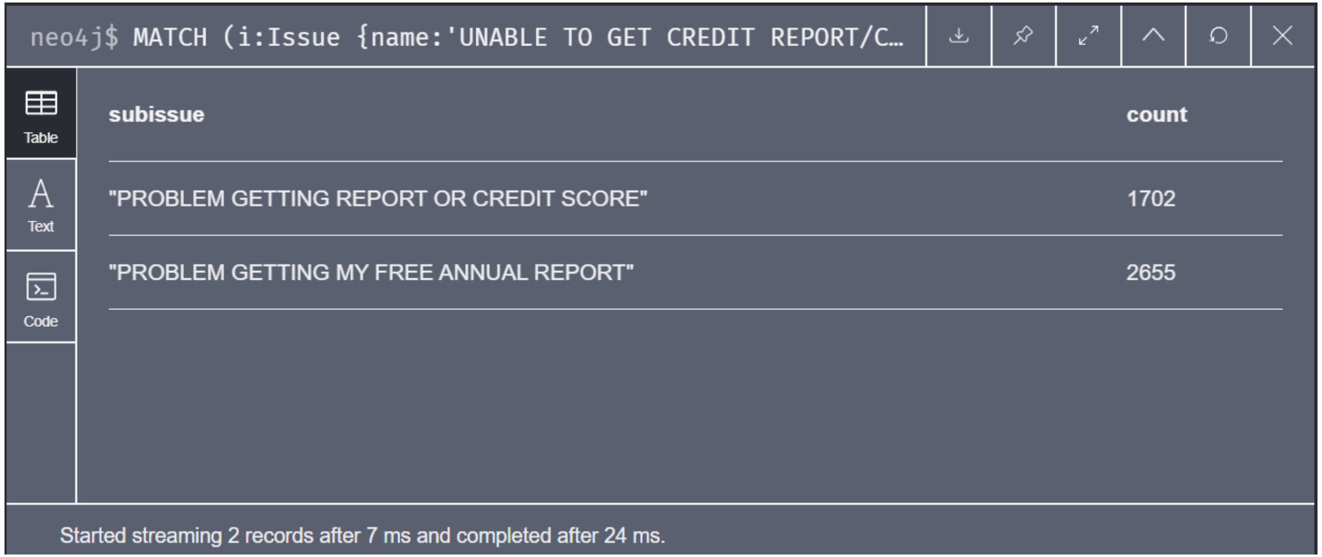


```
neo4j$ MATCH (Complaint)-[:ABOUT]→(p:Product) MATCH (subp…
```

Product	Sub-Product	Most Issues
"DEBT COLLECTION"	"OTHER (PHONE, HEALTH CLUB, ETC.)"	12545
"DEBT COLLECTION"	"CREDIT CARD"	9714
"DEBT COLLECTION"	"MEDICAL"	4763
"DEBT COLLECTION"	"PAYDAY LOAN"	2854
"DEBT COLLECTION"	"MORTGAGE"	1731
"DEBT COLLECTION"	"AUTO"	1235
"DEBT COLLECTION"	"NON-FEDERAL STUDENT LOAN"	1066
"DEBT COLLECTION"	"FEDERAL STUDENT LOAN"	960

5. Number of Sub-issues having “Unable to get report/credit score” issue

```
// All sub-issues within the 'Unable to get credit report/credit score' issue.
MATCH (i:Issue {name:'UNABLE TO GET CREDIT REPORT/CREDIT SCORE'})
MATCH (sub:SubIssue)-[:IN_CATEGORY]->(i)
RETURN sub.name AS subissue, COUNT(*) as count
ORDER BY count;
```



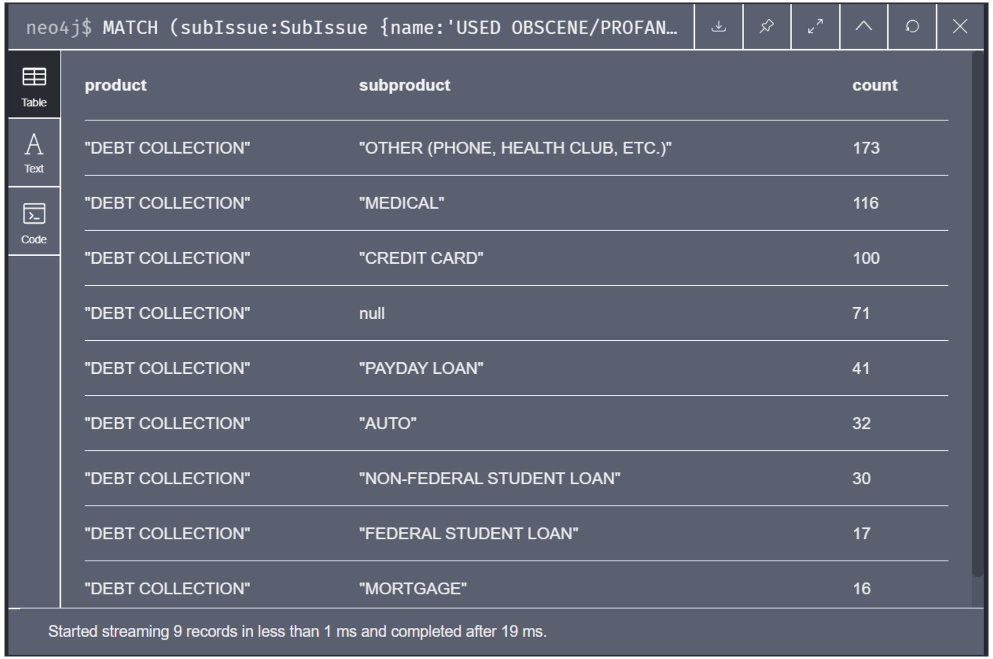
```
neo4j$ MATCH (i:Issue {name:'UNABLE TO GET CREDIT REPORT/C…
```

subissue	count
"PROBLEM GETTING REPORT OR CREDIT SCORE"	1702
"PROBLEM GETTING MY FREE ANNUAL REPORT"	2655

Started streaming 2 records after 7 ms and completed after 24 ms.

6. Finding product and sub-product having “obscene / abusive language” sub-issue

```
MATCH (subIssue:SubIssue {name:'USED OBSCENE/PROFANE/ABUSIVE LANGUAGE'})
MATCH (complaint:Complaint)-[:WITH]->(subIssue)
MATCH (complaint)-[:ABOUT]->(p:Product)
OPTIONAL MATCH (complaint)-[:ABOUT]->(sub:SubProduct)
RETURN p.name AS product, sub.name AS subproduct, COUNT(*) AS count
ORDER BY count DESC;
```



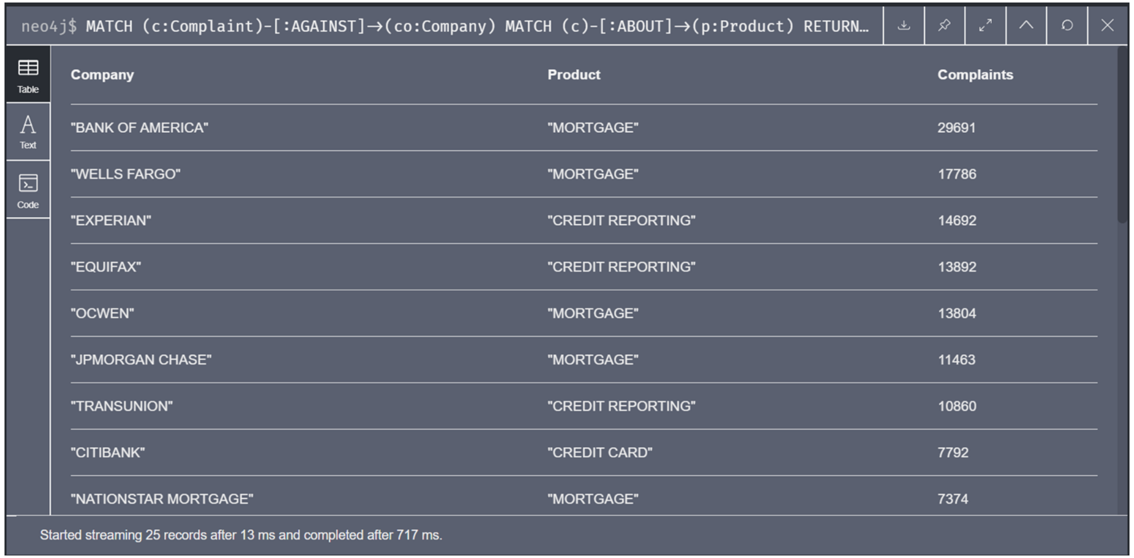
```
neo4j$ MATCH (subIssue:SubIssue {name:'USED OBSCENE/PROFAN…
```

product	subproduct	count
"DEBT COLLECTION"	"OTHER (PHONE, HEALTH CLUB, ETC.)"	173
"DEBT COLLECTION"	"MEDICAL"	116
"DEBT COLLECTION"	"CREDIT CARD"	100
"DEBT COLLECTION"	null	71
"DEBT COLLECTION"	"PAYDAY LOAN"	41
"DEBT COLLECTION"	"AUTO"	32
"DEBT COLLECTION"	"NON-FEDERAL STUDENT LOAN"	30
"DEBT COLLECTION"	"FEDERAL STUDENT LOAN"	17
"DEBT COLLECTION"	"MORTGAGE"	16

Started streaming 9 records in less than 1 ms and completed after 19 ms.

7. Companies having products with most complaints

```
// Top Companies with products having most number of complaints
MATCH (c:Complaint)-[:AGAINST]->(co:Company)
MATCH (c)-[:ABOUT]->(p:Product)
RETURN co.name AS Company, p.name AS Product, COUNT(*) AS `Complaints`
ORDER BY `Complaints` DESC
LIMIT 25;
```



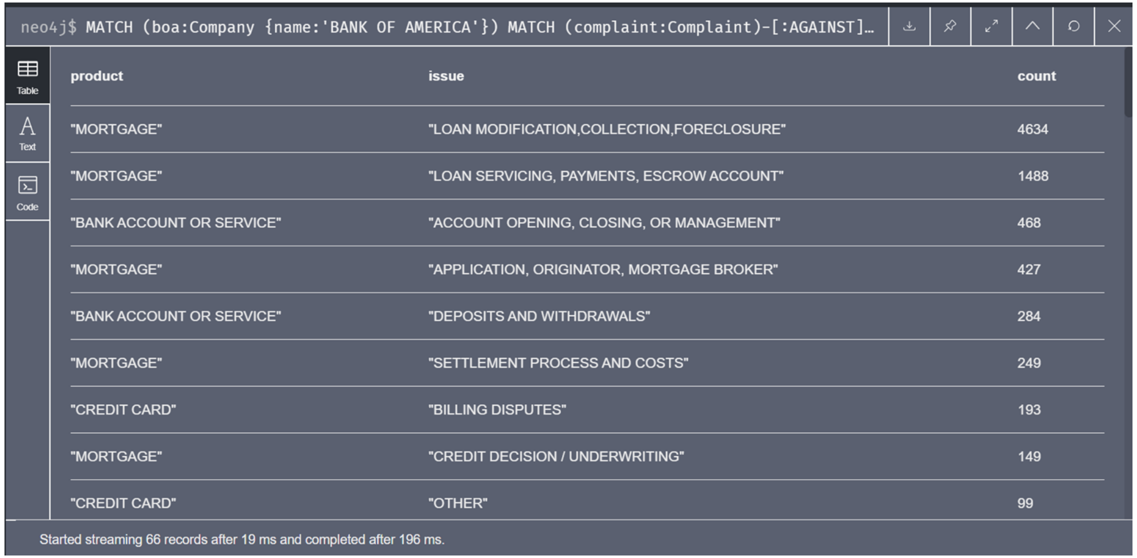
```
neo4j$ MATCH (c:Complaint)-[:AGAINST]→(co:Company) MATCH (c)-[:ABOUT]→(p:Product) RETURN…
```

Company	Product	Complaints
"BANK OF AMERICA"	"MORTGAGE"	29691
"WELLS FARGO"	"MORTGAGE"	17786
"EXPERIAN"	"CREDIT REPORTING"	14692
"EQUIFAX"	"CREDIT REPORTING"	13892
"OCWEN"	"MORTGAGE"	13804
"JPMORGAN CHASE"	"MORTGAGE"	11463
"TRANSUNION"	"CREDIT REPORTING"	10860
"CITIBANK"	"CREDIT CARD"	7792
"NATIONSTAR MORTGAGE"	"MORTGAGE"	7374

Started streaming 25 records after 13 ms and completed after 717 ms.

8. Exploring “Bank Of America’s” product and issues

```
// Top product and issue combinations with disputed responses at Bank Of America
MATCH (boa:Company {name:'BANK OF AMERICA'})
MATCH (complaint:Complaint)-[:AGAINST]->(boa)
MATCH (:Response)-[:TO {disputed:true}]->(complaint)
MATCH (complaint)-[:ABOUT]->(p:Product)
MATCH (complaint)-[:WITH]->(i:Issue)
RETURN p.name AS product, i.name AS issue, COUNT(*) AS count
ORDER BY count DESC;
```



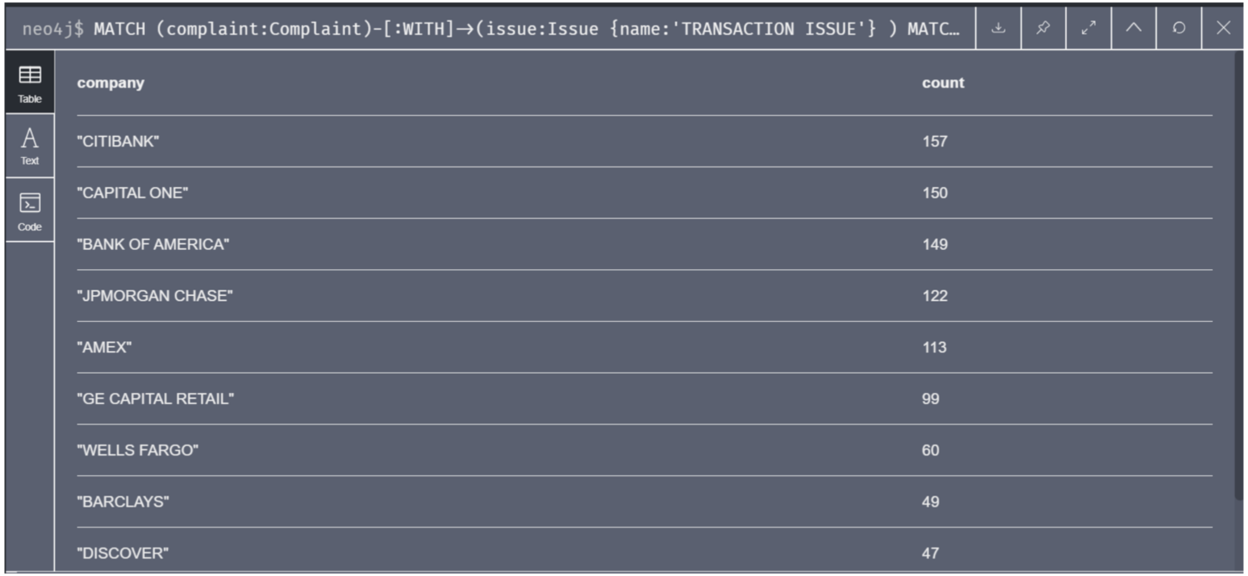
```
neo4j$ MATCH (boa:Company {name:'BANK OF AMERICA'}) MATCH (complaint:Complaint)-[:AGAINST]…
```

product	issue	count
"MORTGAGE"	"LOAN MODIFICATION,COLLECTION,FORECLOSURE"	4634
"MORTGAGE"	"LOAN SERVICING, PAYMENTS, ESCROW ACCOUNT"	1488
"BANK ACCOUNT OR SERVICE"	"ACCOUNT OPENING, CLOSING, OR MANAGEMENT"	468
"MORTGAGE"	"APPLICATION, ORIGINATOR, MORTGAGE BROKER"	427
"BANK ACCOUNT OR SERVICE"	"DEPOSITS AND WITHDRAWALS"	284
"MORTGAGE"	"SETTLEMENT PROCESS AND COSTS"	249
"CREDIT CARD"	"BILLING DISPUTES"	193
"MORTGAGE"	"CREDIT DECISION / UNDERWRITING"	149
"CREDIT CARD"	"OTHER"	99

Started streaming 66 records after 19 ms and completed after 196 ms.

9. Top Company having “Transaction Issue”

```
// Top company associated with the Transaction issue.
MATCH (complaint:Complaint)-[:WITH]->(issue:Issue {name:'TRANSACTION ISSUE'} )
MATCH (complaint)-[:AGAINST]->(company:Company)
RETURN company.name AS company, COUNT(*) AS count
ORDER BY count DESC
LIMIT 10;
```

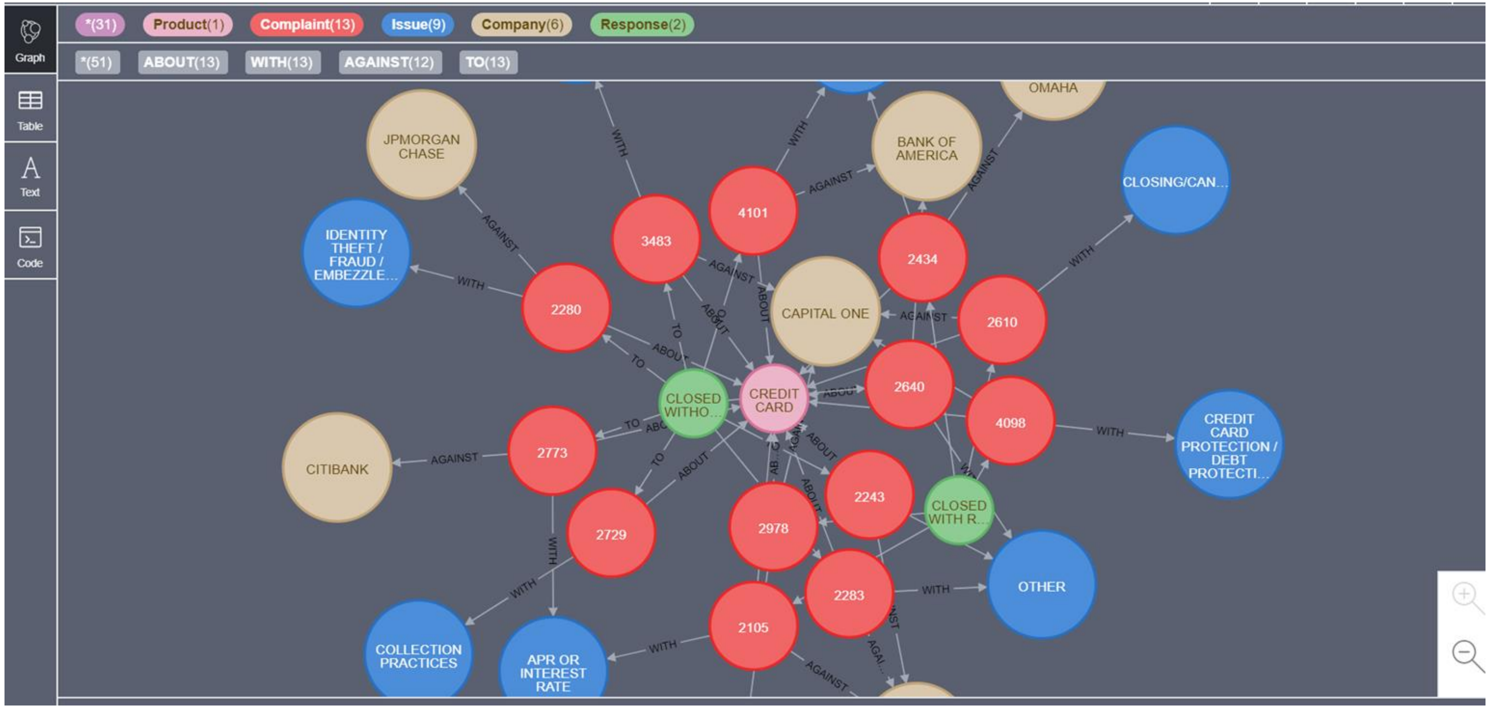


```
neo4j$ MATCH (complaint:Complaint)-[:WITH]→(issue:Issue {name:'TRANSACTION ISSUE'} ) MATC…
```

company	count
"CITIBANK"	157
"CAPITAL ONE"	150
"BANK OF AMERICA"	149
"JPMORGAN CHASE"	122
"AMEX"	113
"GE CAPITAL RETAIL"	99
"WELLS FARGO"	60
"BARCLAYS"	49
"DISCOVER"	47

10. *Exploring second degree connection with product having "card" in it*

```
// Exploring second degree connection with product having "card" in it
MATCH p=(pro:Product) – [*..2] – () WHERE pro.name CONTAINS "CARD" RETURN
p LIMIT 50;
```



11. Finding Products having responses that are timely disputed

```
MATCH (c:Complaint)-[:ABOUT]->(p:Product)
MATCH (r:Response)-[:TO]->(c:Complaint)
MATCH (r:Response)-[:TO {disputed:true}]->(:Complaint)
MATCH (r:Response)-[:TO {timely:true}]->(:Complaint)
RETURN p.name AS product, r.name AS response, COUNT(*) AS count
ORDER BY count DESC;
```

Output: Out of Memory Error

References:

- https://www.youtube.com/watch?v=Eh_79goBRUk
- https://github.com/nicolewhite/neo4j-complaints

THANK YOU.