# Big Data Architecture Lab 3 Report

**Task 1: data import**

1. Go to the home folder of neo4j then start it:

   ./bin/neo4j start

   Open Cypher:

   ./bin/cypher-shell

```
(base) cherilyndeMacBook-Pro:NEO4J_HOME cherilyn$ ./bin/cypher-shell
username: neo4j
password: ******
Connected to Neo4j 3.5.11 at bolt://localhost:7687 as user neo4j.
Type :help for a list of available commands or :exit to exit the shell.
Note that Cypher queries must end with a semicolon.
```

   Then import the CSV files:

   LOAD CSV FROM "file:///boston-crime-incident-reports-10k.csv" AS row
   RETURN row

   We can see all the contents in this file and the last line shows:
   10000 rows available after 21 ms, consumed after another 573 ms

   LOAD CSV FROM "file:///boston-offense-codes-lookup.csv" AS row
   RETURN row

   We can see all the contents in this file and the last line shows:
   577 rows available after 28 ms, consumed after another 49 ms

2. We need to add nodes and labels first.

   USING PERIODIC COMMIT
   LOAD CSV WITH HEADERS FROM "file:///boston-crime-incident-reports-10k.csv" AS row
   CREATE(:Report {iNumber: row.INCIDENT_NUMBER, oCode: row.OFFENSE_CODE, oCodeGroup: row.OFFENSE_CODE_GROUP, oDescrip: row.OFFENSE_DESCRIPTION, district: row.DISTRICT, reportArea: row.REPORTING_AREA, shoot: row.SHOOTING, date: row.OCCURRED_ON_DATE, year: row.YEAR, month: row.MONTH, day: row.DAY_OF_WEEK, hour: row.HOUR, ucr: row.UCR_PART, street: row.STREET, lat: row.Lat, long: row.Long, location: row.Location})

   Result: 0 rows available after 1406 ms, consumed after another 0 ms
   Added 9999 nodes, Set 156405 properties, Added 9999 labels

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///boston-offense-codes-lookup.csv" AS row
CREATE(:Code {code: row.CODE, name: row.NAME})
```

Result: 0 rows available after 43 ms, consumed after another 0 ms
Added 576 nodes, Set 1152 properties, Added 576 labels

Now we need to set relationship between the two imported files. The relationship is that, every incident has several offense codes.

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM "file:///boston-crime-incident-reports-10k.csv" AS row
MATCH (r:Report {iNumber: row.INCIDENT_NUMBER})
MATCH (c:Code {code: row.OFFENSE_CODE})
MERGE (r)-[:OFFEND]->(c)
```

Result: 0 rows available after 1070 ms, consumed after another 0 ms
Created 23332 relationships

**Task 2: data querying**

1. We can set the condition in MATCH or in WHERE.

```
neo4j> MATCH (r:Report)
       WHERE r.oCodeGroup = 'Drug Violation'
       RETURN COUNT(DISTINCT r.iNumber);
+--------------------------+
| COUNT(DISTINCT r.iNumber) |
+--------------------------+
| 330                      |
+--------------------------+

1 row available after 26 ms, consumed after another 0 ms
neo4j> MATCH (r:Report {oCodeGroup: "Drug Violation"})
       RETURN COUNT(DISTINCT r.iNumber);
+--------------------------+
| COUNT(DISTINCT r.iNumber) |
+--------------------------+
| 330                      |
+--------------------------+

1 row available after 24 ms, consumed after another 0 ms
```

2. In fact, I didn't understand this sentence. We should find the offense names of group Investigate Person. Or we need to find all the offense names that are related to the incidents who also offend Investigate Person.
   For the 1st case, we can simply return its description because the description is the same with the offense name.

   MATCH (r:Report {oCodeGroup: "Investigate Person"})
   RETURN DISTINCT r.oDescrip

   Or we can use the two data.

   MATCH (r:Report {oCodeGroup: "Investigate Person"})
   MATCH (c:Code)
   WHERE c.code = r.oCode
   RETURN DISTINCT c.name

```
neo4j> MATCH (r:Report {oCodeGroup: "Investigate Person"})
       RETURN DISTINCT r.oDescrip;
+----------------------+
| r.oDescrip           |
+----------------------+
| "INVESTIGATE PERSON" |
+----------------------+

1 row available after 16 ms, consumed after another 13 ms
neo4j> MATCH (r:Report {oCodeGroup: "Investigate Person"})
       MATCH (c:Code)
       WHERE c.code = r.oCode
       RETURN DISTINCT c.name;
+----------------------+
| c.name               |
+----------------------+
| "INVESTIGATE PERSON" |
+----------------------+

1 row available after 33 ms, consumed after another 56 ms
```

   For the 2nd case, we need to match the relationships.

   MATCH (r:Report)--(c:Code)
   WHERE r.oCodeGroup = "Investigate Person"
   RETURN DISTINCT c.name

```
neo4j> MATCH (r:Report)--(c:Code)
       WHERE r.oCodeGroup = "Investigate Person"
       RETURN DISTINCT c.name;
+-------------------------------------------------+
| c.name                                          |
+-------------------------------------------------+
| "INVESTIGATE PERSON"                            |
| "TRESPASSING"                                   |
| "PROPERTY - LOST"                               |
| "HARASSMENT"                                    |
| "VAL - VIOLATION OF AUTO LAW - OTHER"           |
| "INVESTIGATE PROPERTY"                          |
| "VANDALISM"                                     |
| "THREATS TO DO BODILY HARM"                     |
| "ASSAULT SIMPLE - BATTERY"                      |
| "ASSAULT & BATTERY"                             |
| "MISSING PERSON - NOT REPORTED - LOCATED"       |
| "VERBAL DISPUTE"                                |
| "SICK/INJURED/MEDICAL - PERSON"                 |
| "SIMPLE ASSAULT"                                |
| "ASSAULT - SIMPLE"                              |
| "ASSAULT - AGGRAVATED - BATTERY"                |
| "DISORDERLY PERSON"                             |
| "DISORDERLY CONDUCT"                            |
| "ASSAULT D/W - OTHER"                           |
| "WARRANT ARREST"                                |
| "REPORT AFFECTING OTHER DEPTS."                 |
| "FRAUD - FALSE PRETENSE"                        |
| "FRAUD - FALSE PRETENSE / SCHEME"               |
| "LARCENY SHOPLIFTING $200 & OVER"               |
| "LARCENY SHOPLIFTING"                           |
| "SICK/INJURED/MEDICAL - POLICE"                 |
| "LARCENY THEFT FROM BUILDING"                   |
| "LARCENY IN A BUILDING $200 & OVER"             |
| "DRUGS - POSS CLASS D"                          |
| "DRUGS - POSS CLASS D - INTENT TO MFR DIST DISP" |
| "DRUGS - POSS CLASS B - INTENT TO MFR DIST DISP" |
+-------------------------------------------------+

31 rows available after 1 ms, consumed after another 31 ms
```

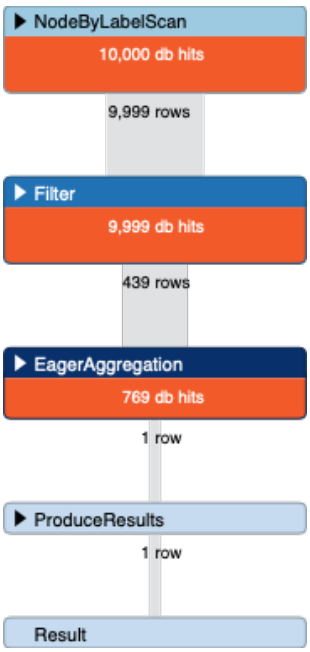
3. We can run the commands and see which operators are doing most of the work with PROFILE.
   With EXPLAIN we can see the execution plan without running the statement. The statement will always return an empty result and make no changes to the database.

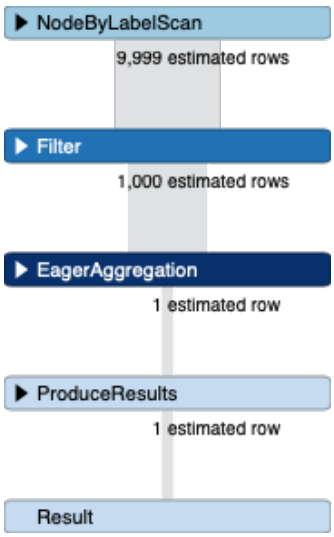
   PROFILE MATCH (r:Report {oCodeGroup: "Drug Violation"})
   RETURN COUNT(r)

   EXPLAIN MATCH (r:Report {oCodeGroup: "Drug Violation"})
   RETURN COUNT(r)

```
neo4j> PROFILE MATCH (r:Report {oCodeGroup: "Drug Violation"})
       RETURN COUNT(DISTINCT r.iNumber);
+--------------------------+
| COUNT(DISTINCT r.iNumber) |
+--------------------------+
| 330                      |
+--------------------------+

+-----------------------------------------------------------------------------------------------+
| Plan      | Statement   | Version      | Planner | Runtime       | Time | DbHits | Rows |
+-----------------------------------------------------------------------------------------------+
| "PROFILE" | "READ_ONLY" | "CYPHER 3.5" | "COST"  | "INTERPRETED" | 14   | 20768  | 1    |
+-----------------------------------------------------------------------------------------------+


+-------------------+----------------+------+---------+----------+-----------------------+--------------------------------+
| Operator          | Estimated Rows | Rows | DB Hits | Cache H/M | Identifiers           | Other                          |
+-------------------+----------------+------+---------+----------+-----------------------+--------------------------------+
| +ProduceResults   |              1 |    1 |       0 |      0/0 | COUNT(DISTINCT r.iNumber) |                            |
| |                 +----------------+------+---------+----------+-----------------------+--------------------------------+
| +EagerAggregation |              1 |    1 |     769 |      0/0 | COUNT(DISTINCT r.iNumber) |                            |
| |                 +----------------+------+---------+----------+-----------------------+--------------------------------+
| +Filter           |           1000 |  439 |    9999 |      0/0 | r                     | r.oCodeGroup = $`  AUTOSTRING0` |
| |                 +----------------+------+---------+----------+-----------------------+--------------------------------+
| +NodeByLabelScan  |           9999 | 9999 |   10000 |      0/0 | r                     | :Report                        |
+-------------------+----------------+------+---------+----------+-----------------------+--------------------------------+

1 row available after 14 ms, consumed after another 0 ms
```

```
neo4j> EXPLAIN MATCH (r:Report {oCodeGroup: "Drug Violation"})
       RETURN COUNT(DISTINCT r.iNumber);
+--------------------------+
| COUNT(DISTINCT r.iNumber) |
+--------------------------+
+--------------------------+

+-------------------------------------------------------------------------+
| Plan      | Statement   | Version      | Planner | Runtime       | Time |
+-------------------------------------------------------------------------+
| "EXPLAIN" | "READ_ONLY" | "CYPHER 3.5" | "COST"  | "INTERPRETED" | 1    |
+-------------------------------------------------------------------------+


+-------------------+----------------+-----------------------+--------------------------------+
| Operator          | Estimated Rows | Identifiers           | Other                          |
+-------------------+----------------+-----------------------+--------------------------------+
| +ProduceResults   |              1 | COUNT(DISTINCT r.iNumber) |                            |
| |                 +----------------+-----------------------+--------------------------------+
| +EagerAggregation |              1 | COUNT(DISTINCT r.iNumber) |                            |
| |                 +----------------+-----------------------+--------------------------------+
| +Filter           |           1000 | r                     | r.oCodeGroup = $`  AUTOSTRING0` |
| |                 +----------------+-----------------------+--------------------------------+
| +NodeByLabelScan  |           9999 | r                     | :Report                        |
+-------------------+----------------+-----------------------+--------------------------------+

0 rows available after 1 ms, consumed after another 0 ms
```

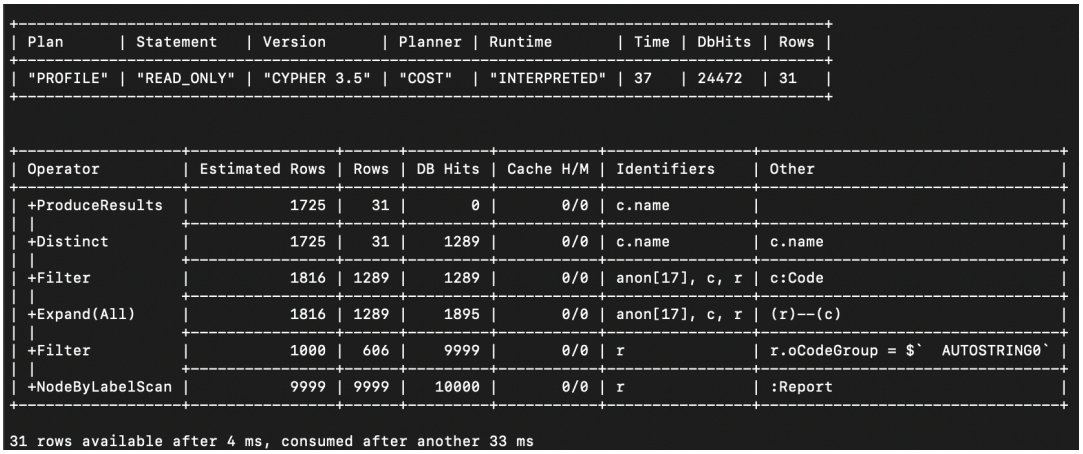


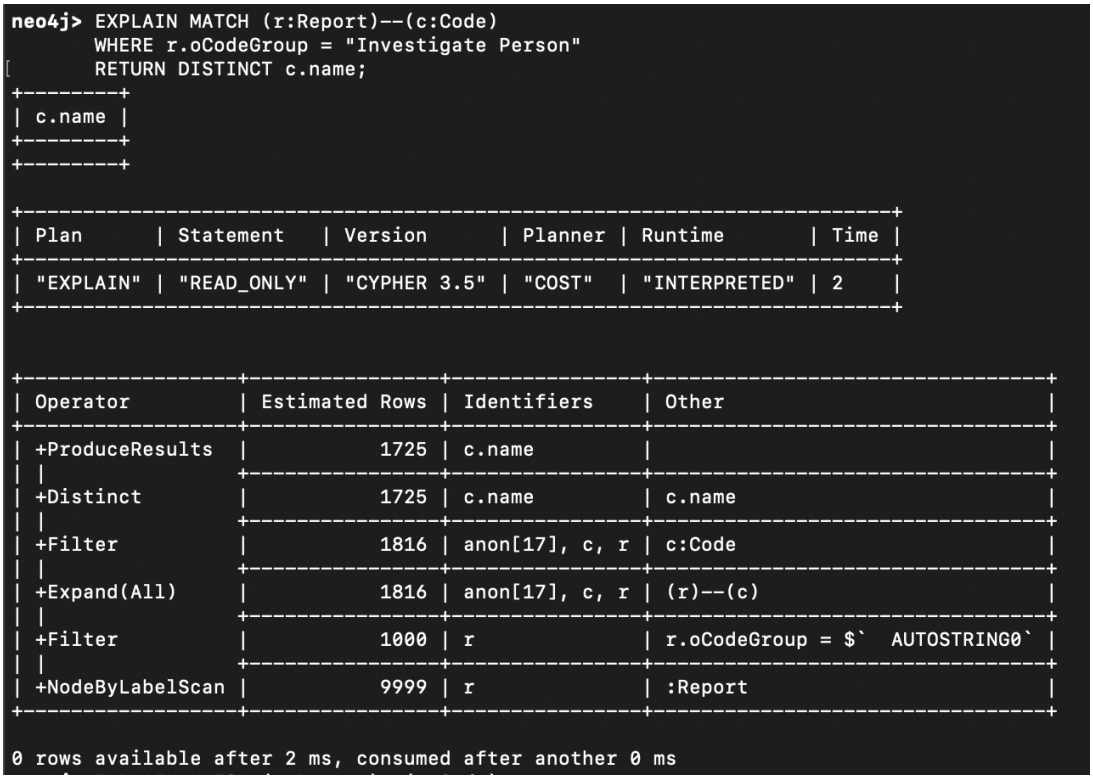

The commands have been run with PROFILE so there are exact result and runtime. But with EXPLAIN, there is only estimated result and there isn't runtime.

```
PROFILE MATCH (r:Report)--(c:Code)
WHERE r.oCodeGroup = "Investigate Person"
RETURN DISTINCT c.name

EXPLAIN MATCH (r:Report)-[:OFFEND]-(c:Code)
WHERE r.oCodeGroup = "Investigate Person"
RETURN DISTINCT c.name
```

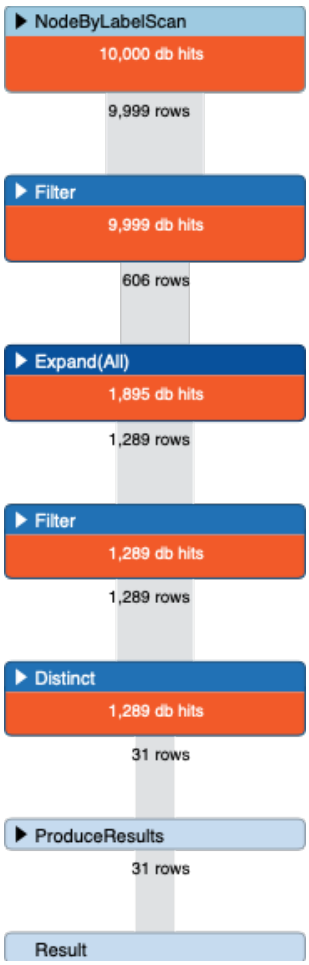
Here I just put a profile result screenshot because the return output is the same as above and it's too long.

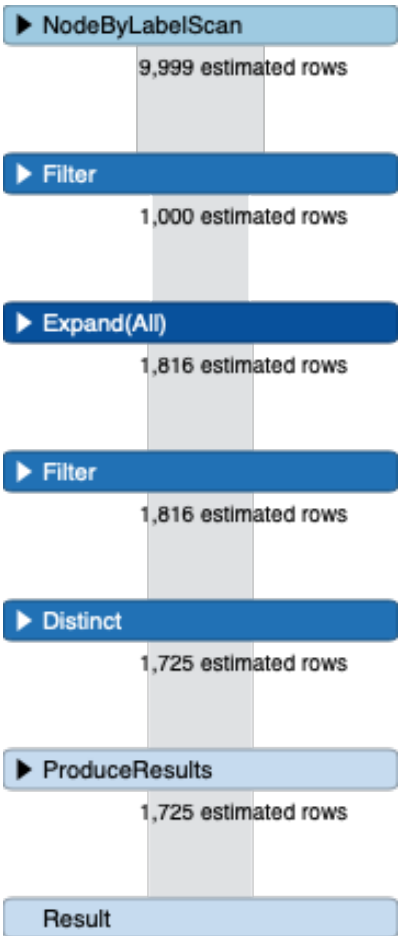


```
+-------------------------------------------------------------------------------+
| Plan      | Statement   | Version      | Planner | Runtime       | Time | DbHits | Rows |
+-------------------------------------------------------------------------------+
| "PROFILE" | "READ_ONLY" | "CYPHER 3.5" | "COST"  | "INTERPRETED" | 37   | 24472  | 31   |
+-------------------------------------------------------------------------------+


+-----------------+----------------+------+---------+----------+---------------+-------------------------------+
| Operator        | Estimated Rows | Rows | DB Hits | Cache H/M | Identifiers   | Other                         |
+-----------------+----------------+------+---------+----------+---------------+-------------------------------+
| +ProduceResults |           1725 |   31 |       0 |      0/0 | c.name        |                               |
| |               +----------------+------+---------+----------+---------------+-------------------------------+
| +Distinct       |           1725 |   31 |    1289 |      0/0 | c.name        | c.name                        |
| |               +----------------+------+---------+----------+---------------+-------------------------------+
| +Filter         |           1816 | 1289 |    1289 |      0/0 | anon[17], c, r | c:Code                       |
| |               +----------------+------+---------+----------+---------------+-------------------------------+
| +Expand(All)    |           1816 | 1289 |    1895 |      0/0 | anon[17], c, r | (r)--(c)                     |
| |               +----------------+------+---------+----------+---------------+-------------------------------+
| +Filter         |           1000 |  606 |    9999 |      0/0 | r             | r.oCodeGroup = $`  AUTOSTRING0` |
| |               +----------------+------+---------+----------+---------------+-------------------------------+
| +NodeByLabelScan |          9999 | 9999 |   10000 |      0/0 | r             | :Report                       |
+-----------------+----------------+------+---------+----------+---------------+-------------------------------+

31 rows available after 4 ms, consumed after another 33 ms
```



```
neo4j> EXPLAIN MATCH (r:Report)--(c:Code)
       WHERE r.oCodeGroup = "Investigate Person"
       RETURN DISTINCT c.name;
+--------+
| c.name |
+--------+
+--------+

+-------------------------------------------------------------------------+
| Plan      | Statement   | Version      | Planner | Runtime       | Time |
+-------------------------------------------------------------------------+
| "EXPLAIN" | "READ_ONLY" | "CYPHER 3.5" | "COST"  | "INTERPRETED" | 2    |
+-------------------------------------------------------------------------+


+-----------------+----------------+---------------+---------------------------------+
| Operator        | Estimated Rows | Identifiers   | Other                           |
+-----------------+----------------+---------------+---------------------------------+
| +ProduceResults |           1725 | c.name        |                                 |
| |               +----------------+---------------+---------------------------------+
| +Distinct       |           1725 | c.name        | c.name                          |
| |               +----------------+---------------+---------------------------------+
| +Filter         |           1816 | anon[17], c, r | c:Code                         |
| |               +----------------+---------------+---------------------------------+
| +Expand(All)    |           1816 | anon[17], c, r | (r)--(c)                       |
| |               +----------------+---------------+---------------------------------+
| +Filter         |           1000 | r             | r.oCodeGroup = $`  AUTOSTRING0` |
| |               +----------------+---------------+---------------------------------+
| +NodeByLabelScan |          9999 | r             | :Report                         |
+-----------------+----------------+---------------+---------------------------------+

0 rows available after 2 ms, consumed after another 0 ms
```



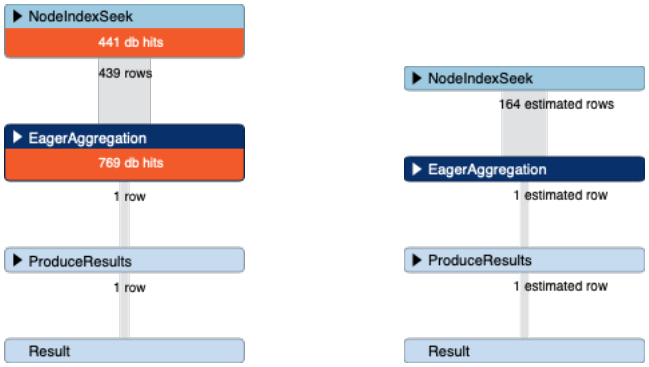




4. CREATE INDEX ON :Report(oCodeGroup);
   Result: 0 rows available after 4 ms, consumed after another 0 ms
   Added 1 indexes

   With this index, we run again the first querying with PROFILE and EXPLAIN.

```
neo4j> PROFILE MATCH (r:Report {oCodeGroup: "Drug Violation"})
       RETURN COUNT(DISTINCT r.iNumber);
+---------------------------+
| COUNT(DISTINCT r.iNumber) |
+---------------------------+
| 330                       |
+---------------------------+

+-------------------------------------------------------------------------------------+
| Plan      | Statement   | Version      | Planner | Runtime       | Time | DbHits | Rows |
+-------------------------------------------------------------------------------------+
| "PROFILE" | "READ_ONLY" | "CYPHER 3.5" | "COST"  | "INTERPRETED" | 3    | 1209   | 1    |
+-------------------------------------------------------------------------------------+


+------------------+----------------+------+---------+-----------+---------------------------+-----------------+---------------------+
| Operator         | Estimated Rows | Rows | DB Hits | Cache H/M | Identifiers               | Ordered by      | Other               |
+------------------+----------------+------+---------+-----------+---------------------------+-----------------+---------------------+
| +ProduceResults  |              1 |    1 |       0 |       0/0 | COUNT(DISTINCT r.iNumber) |                 |                     |
| |                +----------------+------+---------+-----------+---------------------------+-----------------+---------------------+
| +EagerAggregation |              1 |    1 |     769 |       0/0 | COUNT(DISTINCT r.iNumber) |                 |                     |
| |                +----------------+------+---------+-----------+---------------------------+-----------------+---------------------+
| +NodeIndexSeek   |            164 |  439 |     440 |       0/0 | r                         | r.oCodeGroup ASC | :Report(oCodeGroup) |
+------------------+----------------+------+---------+-----------+---------------------------+-----------------+---------------------+

1 row available after 3 ms, consumed after another 0 ms
```

```
neo4j> EXPLAIN MATCH (r:Report {oCodeGroup: "Drug Violation"})
       RETURN COUNT(DISTINCT r.iNumber);
+---------------------------+
| COUNT(DISTINCT r.iNumber) |
+---------------------------+
+---------------------------+

+-------------------------------------------------------------------------+
| Plan      | Statement   | Version      | Planner | Runtime       | Time |
+-------------------------------------------------------------------------+
| "EXPLAIN" | "READ_ONLY" | "CYPHER 3.5" | "COST"  | "INTERPRETED" | 3    |
+-------------------------------------------------------------------------+


+-------------------+----------------+---------------------------+-------------------+----------------------+
| Operator          | Estimated Rows | Identifiers               | Ordered by        | Other                |
+-------------------+----------------+---------------------------+-------------------+----------------------+
| +ProduceResults   |              1 | COUNT(DISTINCT r.iNumber) |                   |                      |
| |                 +----------------+---------------------------+-------------------+----------------------+
| +EagerAggregation |              1 | COUNT(DISTINCT r.iNumber) |                   |                      |
| |                 +----------------+---------------------------+-------------------+----------------------+
| +NodeIndexSeek    |            164 | r                         | r.oCodeGroup ASC  | :Report(oCodeGroup)  |
+-------------------+----------------+---------------------------+-------------------+----------------------+

0 rows available after 3 ms, consumed after another 0 ms
```




Now it uses NodeIndexSeek in the first step instead of NodeByLabelScan. And there is no longer a step named “Filter”. It becomes more efficient than before.

For the second querying it also improved a lot.

```
+-----------------------------------------------------------------------------------+
| Plan      | Statement   | Version      | Planner | Runtime       | Time | DbHits | Rows |
+-----------------------------------------------------------------------------------+
| "PROFILE" | "READ_ONLY" | "CYPHER 3.5" | "COST"  | "INTERPRETED" | 31   | 5081   | 31   |
+-----------------------------------------------------------------------------------+


+-----------------+----------------+------+---------+-----------+-----------------+-----------------+----------------------+
| Operator        | Estimated Rows | Rows | DB Hits | Cache H/M | Identifiers     | Ordered by      | Other                |
+-----------------+----------------+------+---------+-----------+-----------------+-----------------+----------------------+
| +ProduceResults |            363 |   31 |       0 |       0/0 | c.name          | r.oCodeGroup ASC |                      |
| |               +----------------+------+---------+-----------+-----------------+-----------------+----------------------+
| +Distinct       |            363 |   31 |    1289 |       0/0 | c.name          | r.oCodeGroup ASC | c.name               |
| |               +----------------+------+---------+-----------+-----------------+-----------------+----------------------+
| +Filter         |            383 | 1289 |    1289 |       0/0 | anon[17], c, r  | r.oCodeGroup ASC | c:Code               |
| |               +----------------+------+---------+-----------+-----------------+-----------------+----------------------+
| +Expand(All)    |            383 | 1289 |    1895 |       0/0 | anon[17], c, r  | r.oCodeGroup ASC | (r)--(c)             |
| |               +----------------+------+---------+-----------+-----------------+-----------------+----------------------+
| +NodeIndexSeek  |            164 |  606 |     608 |       0/0 | r               | r.oCodeGroup ASC | :Report(oCodeGroup)  |
+-----------------+----------------+------+---------+-----------+-----------------+-----------------+----------------------+

31 rows available after 22 ms, consumed after another 9 ms
neo4j> EXPLAIN MATCH (r:Report)-[:OFFEND]-(c:Code)
       WHERE r.oCodeGroup = "Investigate Person"
       RETURN DISTINCT c.name;
+--------+
| c.name |
+--------+
+--------+

+-------------------------------------------------------------------------+
| Plan      | Statement   | Version      | Planner | Runtime       | Time |
+-------------------------------------------------------------------------+
| "EXPLAIN" | "READ_ONLY" | "CYPHER 3.5" | "COST"  | "INTERPRETED" | 17   |
+-------------------------------------------------------------------------+


+-----------------+----------------+----------------+-----------------+--------------------------+
| Operator        | Estimated Rows | Identifiers    | Ordered by      | Other                    |
+-----------------+----------------+----------------+-----------------+--------------------------+
| +ProduceResults |            363 | c.name         | r.oCodeGroup ASC |                          |
| |               +----------------+----------------+-----------------+--------------------------+
| +Distinct       |            363 | c.name         | r.oCodeGroup ASC | c.name                   |
| |               +----------------+----------------+-----------------+--------------------------+
| +Filter         |            383 | anon[17], c, r | r.oCodeGroup ASC | c:Code                   |
| |               +----------------+----------------+-----------------+--------------------------+
| +Expand(All)    |            383 | anon[17], c, r | r.oCodeGroup ASC | (r)-[anon[17]:OFFEND]-(c) |
| |               +----------------+----------------+-----------------+--------------------------+
| +NodeIndexSeek  |            164 | r              | r.oCodeGroup ASC | :Report(oCodeGroup)      |
+-----------------+----------------+----------------+-----------------+--------------------------+

0 rows available after 17 ms, consumed after another 0 ms
```

5. Show all the incidents and their offense names of Part One in Sunday order by their occurred time.

   MATCH (r:Report)--(c:Code)
   WHERE r.ucr = "Part One" AND r.day = "Sunday"
   RETURN r.iNumber, r.date, c.name
   ORDER BY r.date

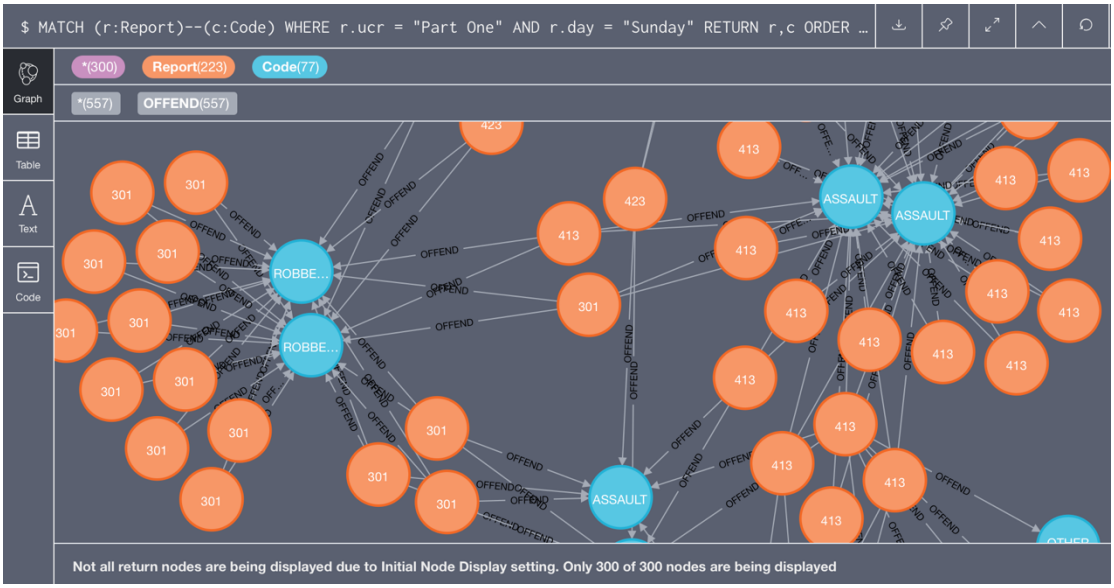
| "I192077812" | "2019-09-22 21:00:00" | "LARCENY NON-ACCESSORY FROM VEH. $200 & OVER" |
|---|---|---|
| "I192076656" | "2019-09-22 21:00:00" | "LARCENY THEFT FROM MV - NON-ACCESSORY" |
| "I192076656" | "2019-09-22 21:00:00" | "LARCENY NON-ACCESSORY FROM VEH. $200 & OVER" |
| "I192076531" | "2019-09-22 21:54:00" | "LARCENY IN A BUILDING $200 & OVER" |
| "I192076531" | "2019-09-22 21:54:00" | "LARCENY THEFT FROM BUILDING" |
| "I192076533" | "2019-09-22 22:00:00" | "AUTO THEFT" |
| "I192076533" | "2019-09-22 22:00:00" | "AUTO THEFT" |
| "I192076826" | "2019-09-22 22:30:00" | "B&E NON-RESIDENCE DAY - FORCIBLE" |
| "I192076826" | "2019-09-22 22:30:00" | "BURGLARY - COMMERICAL - FORCE" |
| "I192076529" | "2019-09-22 22:44:00" | "LARCENY OTHER $200 & OVER" |
| "I192076529" | "2019-09-22 22:44:00" | "LARCENY ALL OTHERS" |
| "I192076779" | "2019-09-22 23:00:00" | "ROBBERY - STREET" |
| "I192076779" | "2019-09-22 23:00:00" | "ROBBERY - FIREARM - BANK" |
| "I192078600" | "2019-09-29 00:08:40" | "ASSAULT - AGGRAVATED - BATTERY" |
| "I192078600" | "2019-09-29 00:08:40" | "ASSAULT D/W - OTHER" |
| "I192078590" | "2019-09-29 00:32:00" | "B&E RESIDENCE DAY - NO FORCE" |
| "I192078590" | "2019-09-29 00:32:00" | "BURGLARY - RESIDENTIAL - NO FORCE" |
| "I192078609" | "2019-09-29 01:01:00" | "ROBBERY - FIREARM - BANK" |
| "I192078609" | "2019-09-29 01:01:00" | "ROBBERY - STREET" |
| "I192078603" | "2019-09-29 01:30:00" | "ASSAULT & BATTERY" |
| "I192078603" | "2019-09-29 01:30:00" | "ROBBERY - FIREARM - BANK" |
| "I192078603" | "2019-09-29 01:30:00" | "ASSAULT SIMPLE - BATTERY" |
| "I192078603" | "2019-09-29 01:30:00" | "ROBBERY - STREET" |
| "I192078613" | "2019-09-29 02:30:00" | "ASSAULT & BATTERY D/W - OTHER ON POLICE OFFICER" |
| "I192078613" | "2019-09-29 02:30:00" | "ASSAULT & BATTERY" |
| "I192078613" | "2019-09-29 02:30:00" | "ASSAULT SIMPLE - BATTERY" |
| "I192078613" | "2019-09-29 02:30:00" | "ASSAULT - AGGRAVATED" |
| "I192078622" | "2019-09-29 03:04:00" | "LARCENY ALL OTHERS" |
| "I192078622" | "2019-09-29 03:04:00" | "LARCENY OTHER $200 & OVER" |
| "I192078636" | "2019-09-29 04:40:00" | "BURGLARY - RESIDENTIAL - ATTEMPT" |
| "I192078636" | "2019-09-29 04:40:00" | "B&E RESIDENCE DAY - ATTEMPT FORCE" |

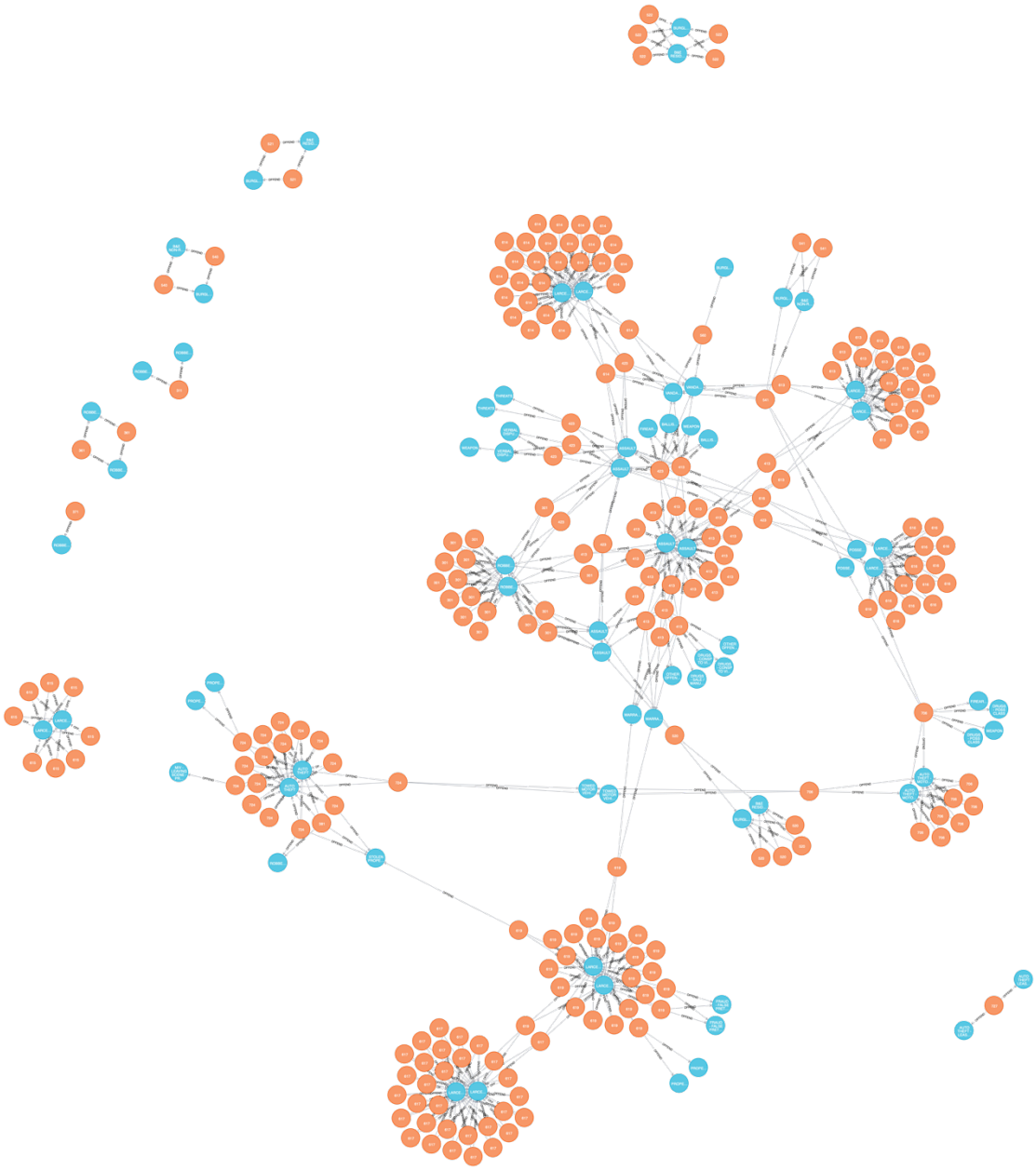
593 rows available after 49 ms, consumed after another 3 ms

**Task 3: results visualization**

In order to get the graph, we enter the command lines in Neo4j Browser:
MATCH (r:Report)--(c:Code)
WHERE r.ucr = "Part One" AND r.day = "Sunday"
RETURN r,c
ORDER BY r.date

Then we can get the graph:


$ MATCH (r:Report)--(c:Code) WHERE r.ucr = "Part One" AND r.day = "Sunday" RETURN r,c ORDER …
Graph
Table
Text
Code
*(300)
Report(223)
Code(77)
*(557)
OFFEND(557)
ROBBE…
ROBBE…
ASSAULT
ASSAULT
ASSAULT
OFFEND
301
413
423
Not all return nodes are being displayed due to Initial Node Display setting. Only 300 of 300 nodes are being displayed



With this visualized result, we can see easily the nodes and relationships in the graph. It is convenient to use the graph to find the relationship network between users. It can also find the shortest path between two users. The query speed is very fast.
Here the relationship is very simple and not important to us, so it’s not very practical and readable to use the graph.