# **Philly Graph DB Meetup**

# **Querying Open Civic Data Using Cypher & Neo4j**

### **[Philly Graph DB](http://meetu.ps/e/Fzq5M/1Fh5w/f)** [**Meetup Page**](http://meetu.ps/e/Fzq5M/1Fh5w/f) **[OpenDataPhilly.com](https://www.opendataphilly.org/)** **[PPD Datasets Cleaned Google Drive](https://drive.google.com/drive/folders/1iJnBiUgt9J8TGbME4fzZz97zGCklWHBM?usp=sharing)** **[PPD Complaints Datasets - Open Data Philly](https://www.opendataphilly.org/dataset/police-complaints)**

[**Scrolly Telly Example**](http://philadelphia.maps.arcgis.com/apps/MapJournal/index.html?appid=d498be2dde18426193679f5e9ce0e6e5)

[**Mondeling Your Data with Arrows Tool**](http://www.apcjones.com/arrows/)

## **Getting a Neo4j Database**

There are multiple ways to get setup with your own Neo4j database.

You can install the Desktop client, setup a full version on a server, or use Neo4j’s publicly available Sandboxes.

**Neo4j Desktop User Interface Guide:** <https://neo4j.com/developer/guide-neo4j-desktop/#_installing_and_starting_neo4j_desktop>

**Getting Started with the Neo4j desktop app:**

1. **Download Neo4j:** <https://neo4j.com/download/>
2. Open the [Neo4j Desktop App]
3. Scroll down and click [New Graph]
4. Click on [Create a local graph]
5. Name Graph [PPD Complaints] and set password to [password]
6. In the new Graph Box your created Click [Manage]
7. Next click [Open Folder] and select the [Import] Folder for the selection of folders keep that folder open and go to your browser.
8. Past this link to the google drive with all of the datasets for the PPD complaints <https://drive.google.com/drive/folders/1iJnBiUgt9J8TGbME4fzZz97zGCklWHBM?usp=sharing> and [Download] all 3 complaint CSV’s for the google drive folder.
9. Go to your desktop and and unzip the CSV’s you just downloaded and drag them into the import folder you should still have open.
10. Once the CSV’s are in your import folder [Close] the folder and go back to your neo4j desktop app and [Click] the Play button(looks like an arrowhead), then [Click] the [Open Browser] button. And sew Neo4j Browser will popup.
11. From your new Neo4j browser Highlight and copy each section below in green and past them one section at a time in to your Neo4j browser Command Line bar at the top and [Click] [Play] after each section pasted into the browser command line bar.
12. Once you have pasted all 6 sections in green you’ll be ready to start querying your datasets in the Neo4j Browser. You can also view your data as nodes by clicking on the Database symbol at the top left of your Neo4j Browser. In the new pop out you’ll see each data set as a tag, click on one to view the visualization of your data

**Intro to Cypher:** <https://neo4j.com/developer/cypher-query-language/>

**OpenData Philly Website:**  <https://www.opendataphilly.org/>

## **Using a Neo4j Sandbox**

### Make a Sandbox at Neo4j: <https://neo4j.com/sandbox-v2/>

1. Click Get Started Then Create an account or sign in
2. Select the one named “**Blank Sandbox**” and click Launch Sandbox
3. On the Database info section, under the Get Started with your Neo4j Sandbox text, Click “Visit the Neo4j Browser” to connect to the database.
4. Run the Queries below, which download and process the CSV into the database

**Complaints**:

LOAD CSV WITH HEADERS FROM 'http://cypherphil.ly/content/ppd\_complaints.csv' AS line

CREATE (:PPD\_Complaints { cap\_number: line.cap\_number, date\_received: line.date\_received, dist\_occurrence: line.dist\_occurrence, general\_cap\_classification: line.general\_cap\_classification, summary: line.summary })

**Findings**:

LOAD CSV WITH HEADERS FROM 'http://cypherphil.ly/content/ppd\_complaint\_disciplines.csv' AS line

CREATE (:PPD\_Complaint\_Disciplines { cap\_number: line.cap\_number, po\_initials: line.po\_initials, po\_race: line.po\_race, po\_sex: line.po\_sex, allegations\_investigated: line.allegations\_investigated, investigative\_findings: line.investigative\_findings, disciplinary\_findings: line.disciplinary\_findings })

**Complainants**:

LOAD CSV WITH HEADERS FROM 'http://cypherphil.ly/content/ppd\_complaint\_complainants.csv' AS line

CREATE (:PPD\_Complaint\_Complainants { cap\_number: line.cap\_number, complainant\_sex: line.complainant\_sex, complainant\_race: line.complainant\_race, complainant\_age: line.complainant\_age, complainant\_initials: line.complainant\_initials })

**Neo4j Create and load template and Guide:**

CREATE ←(This creates the entity Labels from header names from your datasets)

(`0`:->PPD\_Complaints <--(Your Dataset file name goes here)

(CSV header titles go here)-->{**cap\_number**:"string", **dist\_occurrence**:"string", **general\_cap\_classification**:"string", **summary**:"string" })

LOAD CSV WITH HEADERS FROM 'file:///ppd\_complaints.csv' AS line <--(This Line loads the contents of the CSV into the Neo4j app project database)

CREATE (:PPD\_Complaints←(dataset name) (CSV headers)--> { **cap\_number**: line.**cap\_number**, **dist\_occurrence**: line.**dist\_occurrence**, **general\_cap\_classification**: line.**general\_cap\_classification**, **summary**: line.**summary** })<--(This Line created nodes for the CSV you loaded into the dataset)

**Example Blank Template:**

CREATE

(`0`:csv\_name\_here

{**first\_header\_title**:"string", **second\_header\_title**:"string", **thrid\_header\_title**:"string", **forth\_header\_title**:"string" })

LOAD CSV WITH HEADERS FROM 'file:///csv\_name\_here.csv' AS line

CREATE (:csv\_name\_here { **first\_header\_title**: line.**first\_header\_title**, **second\_header\_title**: line.**second\_header\_title**, **thrid\_header\_title**: line.**thrid\_header\_title**, **forth\_header\_title**: line.**forth\_header\_title** })

**Complaints Against Police**

# As part of the Philadelphia Police Department's (PPD) accountability processes, PPD has released two datasets: The Complaints Against Police (CAP) dataset documents the civilian complaints alleging police misconduct and the CAP Findings dataset provides demographic details of the police officer involved, the allegations, and the status of the PPD's Internal Affairs Division's investigation of and findings (if available) about the allegation.

Includes data from 2013 to the present year. Updated monthly.

Datasets:

<http://cypherphil.ly/content/ppd_complaints.csv>

<http://cypherphil.ly/content/ppd_complaint_disciplines.csv>

<http://cypherphil.ly/content/ppd_complaint_complainants.csv>

**DataSet 1 formatting: (Copy and Paste Code in green One section at a time into your local neo4j browser instance)**

**ppd\_complaints:**

CREATE

(`0`:PPD\_Complaints

{**cap\_number**:"string", **date\_received**:"string", **dist\_occurrence**:"string", **general\_cap\_classification**:"string", **summary**:"string" })

LOAD CSV WITH HEADERS FROM 'file:///ppd\_complaints.csv' AS line

CREATE (:PPD\_Complaints { **cap\_number**: line.**cap\_number**, **date\_received**: line.**date\_received**, **dist\_occurrence**: line.**dist\_occurrence**, **general\_cap\_classification**: line.**general\_cap\_classification**, **summary**: line.**summary** })

**DataSet 2 formatting: (Copy and Paste Code in green One section at a time into your local neo4j browser instance)**

**Note: this data set has multiple rows for cap\_number, which is what Neo4j uses to join the nodes. So there will be some rows that cannot join to the complaints and complainants data.**

**ppd\_complaint\_disciplines:**

CREATE

(`0`:PPD\_Complaint\_Disciplines

{cap\_number:"string", po\_initials:"string", po\_race:"string", po\_sex:"string", allegations\_investigated:"string", investigative\_findings:"string", disciplinary\_findings:"string" })

LOAD CSV WITH HEADERS FROM 'file:///ppd\_complaint\_disciplines.csv' AS line

CREATE (:PPD\_Complaint\_Disciplines { **cap\_number**: line.**cap\_number**, **po\_initials**: line.**po\_initials**, **po\_race**: line.**po\_race**, **po\_sex**: line.**po\_sex**, **allegations\_investigated**: line.**allegations\_investigated**, **investigative\_findings**: line.**investigative\_findings**, **disciplinary\_findings**: line.**disciplinary\_findings** })

**DataSet 3 formatting: (Copy and Paste Code in green One section at a time into your local neo4j browser instance)**

**ppd\_complaint\_complainants:**

CREATE

(`0`:PPD\_Complaint\_Complainants

{**cap\_number**:"string", **complainant\_sex**:"string", **complainant\_race**:"string", **complainant\_age**:"string", **complainant\_initials**:"string" })

LOAD CSV WITH HEADERS FROM 'file:///ppd\_complaint\_complainants.csv' AS line

CREATE (:PPD\_Complaint\_Complainants { **cap\_number**: line.**cap\_number**, **complainant\_sex**: line.**complainant\_sex**, **complainant\_race**: line.**complainant\_race**, **complainant\_age**: line.**complainant\_age**, **complainant\_initials**: line.**complainant\_initials** })

**Sample Queries for PPD Datasets**

1st Example Query (Finds [District] with the Highest [Cap Number] occurrences in [Descending Order])

MATCH (n:PPD\_Complaints)

WITH COUNT(DISTINCT n.cap\_number) AS occurrences, toInteger(n.dist\_occurrence) AS district

RETURN occurrences, district

ORDER BY occurrences DESC

2nd Example Query (Counts [Complaints] with the Highest occurrence by [Type] and lists out in [Descending Order])

MATCH (n)

WITH COUNT(DISTINCT n.cap\_number) as complaints, n.general\_cap\_classification as type

RETURN type, complaints

ORDER BY complaints DESC

3rd Keyword Search in [Complaints] [summary] Label (assault)

MATCH (n:PPD\_Complaints)

WHERE n.summary CONTAINS 'assault'

RETURN n

Keyword Search in [Complaints] [summary] label (any regular expression of variation of Assault)

MATCH (n:PPD\_Complaints)

WHERE n.summary =~ '(?i).\*assault.\*'

RETURN n

**PPD Complaint Relationships****This section will help you start building relationships in in your graph**

To Delete A Relationship Label and node relationship

MATCH (a:PPD\_Complaints)-[r:HaveA]->(b:PPD\_Complaint\_Complainants) ←- Example Relationship Types Label to be removed

DELETE r

1st Relationship(Complaints Has A Complainant/s):

MATCH (a:PPD\_Complaints),(b:PPD\_Complaint\_Complainants)

WHERE a.cap\_number = b.cap\_number

CREATE (a)-[:HAS\_A]->(b)

RETURN a,b

2nd Relationship(Complaints Has A Disciplines):

MATCH (a:PPD\_Complaints),(b:PPD\_Complaint\_Disciplines)

WHERE a.cap\_number = b.cap\_number

CREATE (a)-[:HAS\_A]->(b)

RETURN a,b

**Data Annotations**This section will help you start annotating your data to quick search your data for specific qualities.

To Delete A node Label.

MATCH (n)

REMOVE n:PoliceOfficers ←- Example Label to be removed

1st Annotation (Complaints where summer has assault keyword in any variation):

MATCH (n:PPD\_Complaints)

WHERE n.summary =~ '(?i).\*assault.\*'

SET n :PPD\_ASSAULT

RETURN n

# **311 Service and Information Requests** This represents all service and information requests since December 8th, 2014 submitted to Philly311 via the 311 mobile application, calls, walk-ins, emails, the 311 website or social media.

CREATE

(`0`:public\_cases\_fc

{**the\_geom**:"string", **lon**:"string", **objectid**:"string", **service\_request\_id**:"string", **status**:"string", **status\_notes**:"string", **service\_name**:"string", **service\_code**:"string", **the\_geom\_webmercator**:"string", **requested\_datetime**:"string", **updated\_datetime**:"string", **expected\_datetime**:"string", **address**:"string", **zipcode**:"string", **media\_url**:"string", **lat**:"string", **service\_notice**:"string"})

USING PERIODIC COMMIT 500 LOAD CSV WITH HEADERS FROM 'file:///public\_cases\_fc.csv' AS line

CREATE (:public\_cases\_fc { **the\_geom**: line.**the\_geom**, **lon**: line.**lon**, **objectid**: line.**objectid**, **service\_request\_id**: line.**service\_request\_id, service\_request\_id**: line.**service\_request\_id**, **status**: line.**status**, **status\_notes**: line.**status\_notes**, **service\_name**: line.**service\_name**, **service\_code**: line.**service\_code**, **the\_geom\_webmercator**: line.**the\_geom\_webmercator**, **requested\_datetime**: line.**requested\_datetime**, **updated\_datetime**: line.**updated\_datetime**, **expected\_datetime**: line.**expected\_datetime**, **address**: line.**address**, **zipcode**: line.**zipcode**, **media\_url**: line.**media\_url**, **lat**: line.**lat**, **service\_notice**: line.**service\_notice** })