***Chicago Police Department Project Proposal - Checkpoint 3 Queries***

**Question #8: *Do complaints tend to be filed against a cluster of police officers? How do these potential clusters change over time?***

We want to cluster the officers that are involved in the same complaints. To do this, we first ran a query to list out pairs of officers that were involved in the same complaint. To test this, we limited our analysis to allegations that happened in 2015, to keep the amount of data manageable. We used this query (table named officer\_allegation\_network.csv):

*select n1.incident\_date, n1.allegation\_id, n1.disciplined, n1.officer\_id as "officer1", n2.officer\_id as "officer2"*

*from (select da.incident\_date, doa.id, doa.allegation\_id, doa.officer\_id, doa.disciplined*

*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n1*

*cross join (select da.incident\_date, doa.allegation\_id, doa.officer\_id, doa.disciplined*

*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n2*

*where n1.allegation\_id=n2.allegation\_id and n1.officer\_id != n2.officer\_id and n1.incident\_date > '2015-01-01' and n1.incident\_date < '2016-01-01';*

To use the connected components analysis, it needs a list of all of the officers in the analysis. This was acquired with the query (tabled named 2015\_allegation\_network\_officers.csv):

*select distinct n1.officer\_id*

*from (select da.incident\_date, doa.id, doa.allegation\_id, doa.officer\_id, doa.disciplined*

*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n1*

*cross join (select da.incident\_date, doa.allegation\_id, doa.officer\_id, doa.disciplined*

*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n2*

*where n1.allegation\_id=n2.allegation\_id and n1.officer\_id != n2.officer\_id and n1.incident\_date > '2015-01-01' and n1.incident\_date < '2016-01-01';*

Then, we import each of these datasets into spark (databricks platform). Using the queries above, we can create a vertices and edges representation. The first query represents edges, and the second query represents vertices.

In order to use GraphX in Python, we utilized the Graphframe Package via the Library import functionality in Databricks. Using the “Maven Coordinate” option, Graphframe can be installed / attached directly to the cluster that we are currently working on, eliminating the need for additional configuration. Then the following code was used to create the graph, and run both connected components and a PageRank.

from functools import reduce

from pyspark.sql.functions import col, lit, when

from graphframes import \*

vertices = spark.table('allegation\_network\_officers\_csv')

vertices = vertices.withColumnRenamed("officer\_id", "id")

edges = spark.table('officer\_allegation\_network\_csv')

edges = edges.drop("incident\_date","allegation\_id","disciplined")

edges = edges.withColumnRenamed("officer1", "src")

edges = edges.withColumnRenamed("officer2", "dst")

g = GraphFrame(vertices, edges)

sc.setCheckpointDir("/tmp/")

result = g.connectedComponents()

result.show()

results = g.pageRank(resetProbability=0.15, tol=0.01)

display(results.vertices)

**Question #9: *Are we able to see if once police officers potentially "get away" with certain behaviors, they are much more likely to keep committing the same acts repeatedly?***

We needed a group of officers that we consider as ‘bad offenders’ and compare to their peers. To get the list of ‘bad offenders’, we used the top 7% of officers with the highest number of complaints per year normalized by their district average. This was done by taking officers from the top three buckets from figure 1 of our checkpoint #2 analysis (table named ‘Checkpoint\_3\_bad\_officers.csv). We uploaded this file into postgres as a new table called ‘bad\_officers’.

With this, we then wrote a query to calculate the time between each complaint for all of the officers. We used this query (output named officer\_allegation\_intervals.csv):

*select extract(epoch from n2.incident\_date - n1.incident\_date)/86400 as "allegation\_interval\_days", case when n1.officer\_id in (select bo.id from bad\_officers bo) then 'True' else 'False' end as bad\_officer*

*from (select \*, row\_number() over (order by doa."officer\_id", da.incident\_date asc)*

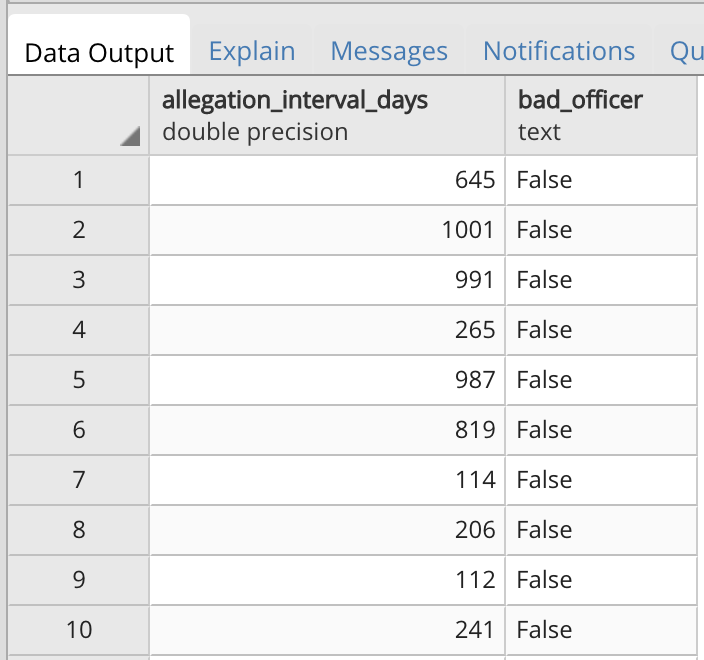
*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n1, (select \*, row\_number() over (order by doa."officer\_id", da.incident\_date asc)*

*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n2*

*where n1.row\_number=n2.row\_number-1 and n1.officer\_id=n2.officer\_id;*



When creating a histogram of this, we found some outliers with the interval being around 16000 causing the graph to be very wide. So we added in a threshold to only look at intervals less than 1000 days (output named Filtered\_Intervals.csv):

*select extract(epoch from n2.incident\_date - n1.incident\_date)/86400 as "allegation\_interval\_days", case when n1.officer\_id in (select bo.id from bad\_officers bo) then 'True' else 'False' end as bad\_officer*

*from (select \*, row\_number() over (order by doa."officer\_id", da.incident\_date asc)*

*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n1, (select \*, row\_number() over (order by doa."officer\_id", da.incident\_date asc)*

*from data\_allegation da*

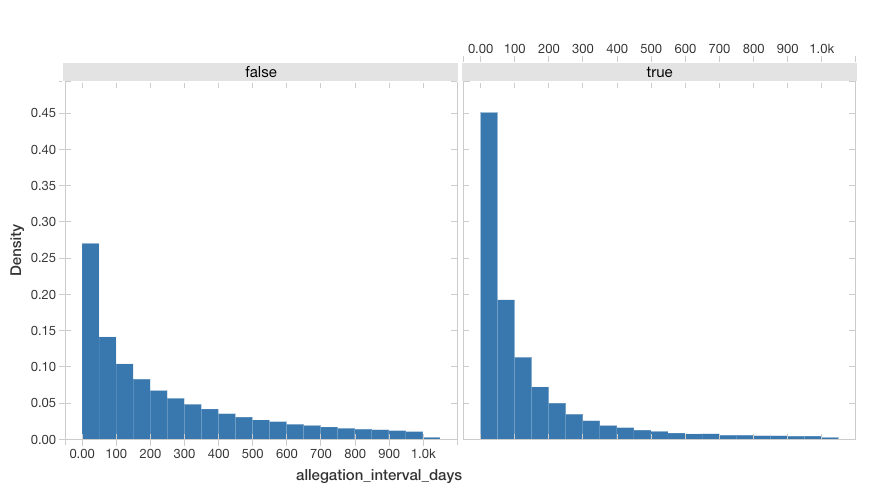
*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n2*

*where n1.row\_number=n2.row\_number-1 and n1.officer\_id=n2.officer\_id and extract(epoch from n2.incident\_date - n1.incident\_date)/86400 < 1000;*

In Spark, we displayed this as a histogram with:

df2 = spark.table("Filtered\_Intervals\_csv")

display(df2)



To dig a bit deeper, we wanted to see if there was going to be a shift in the distribution when looking at the intervals between complaints when the officer got disciplined for the first allegation. To get these intervals, we used this query (output named Disciplined\_Intervals.csv):

*select extract(epoch from n2.incident\_date - n1.incident\_date)/86400 as "allegation\_interval\_days", case when n1.officer\_id in (select bo.id from bad\_officers bo) then 'True' else 'False' end as bad\_officer*

*from (select \*, row\_number() over (order by doa."officer\_id", da.incident\_date asc)*

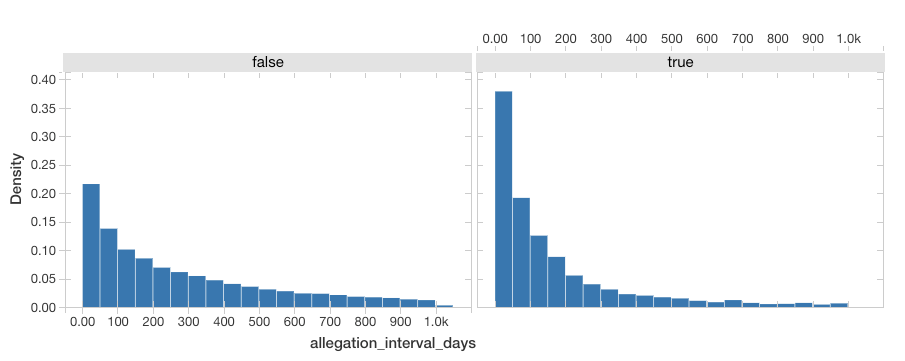
*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n1, (select \*, row\_number() over (order by doa."officer\_id", da.incident\_date asc)*

*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n2*

*where n1.row\_number=n2.row\_number-1 and n1.officer\_id=n2.officer\_id and n1.disciplined = False and extract(epoch from n2.incident\_date - n1.incident\_date)/86400 < 1000;*



And to get the counter-set of intervals between complaints where the officer ***did not*** get disciplined for the first allegation, we used the query:

*select extract(epoch from n2.incident\_date - n1.incident\_date)/86400 as "allegation\_interval\_days", case when n1.officer\_id in (select bo.id from bad\_officers bo) then 'True' else 'False' end as bad\_officer*

*from (select \*, row\_number() over (order by doa."officer\_id", da.incident\_date asc)*

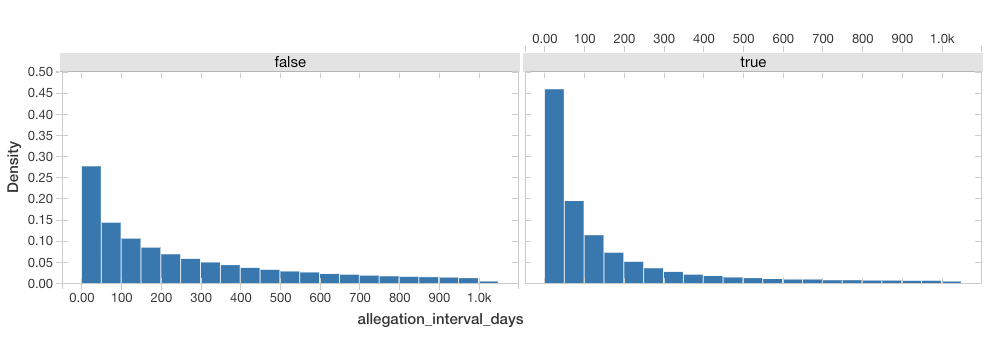
*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n1, (select \*, row\_number() over (order by doa."officer\_id", da.incident\_date asc)*

*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n2*

*where n1.row\_number=n2.row\_number-1 and n1.officer\_id=n2.officer\_id and n1.disciplined = True and extract(epoch from n2.incident\_date - n1.incident\_date)/86400 < 1000;*



To get the overall averages, we used the query:

*select avg(extract(epoch from n2.incident\_date - n1.incident\_date)/86400) as "allegation\_interval\_days"*

*from (select \*, row\_number() over (order by doa."officer\_id", da.incident\_date asc)*

*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n1, (select \*, row\_number() over (order by doa."officer\_id", da.incident\_date asc)*

*from data\_allegation da*

*inner join data\_officerallegation doa on da.id=doa.allegation\_id) n2*

*where n1.row\_number=n2.row\_number-1 and n1.officer\_id=n2.officer\_id and n1.disciplined = True and n1.officer\_id not in (select bo.id from bad\_officers bo);*

Where we changed the ‘not in’ to ‘in’ to switch between sets of officers, and changed the ‘n1.disciplined’ requirement to True and False.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Overall Average Interval (Days)** | **Average Interval After Disciplined Complaint (Days)** | **Average Interval After Non-Disciplined Complaint (Days)** |
| **Repeat Offenders** | 145 | 177 | 142 |
| **Non-Repeat Offenders** | 398 | 481 | 393 |