The Brave Ducks

Checkpoint 4: Graph Analytics

Overview: With the following exploration of the Chicago Polic Database, we inspect co-offending officers and if they have any interesting relationships within each pair of officers. The exploration of the following questions is done via graph analytics, conducted using Apache Spark and GraphX tools, processed in the attached Google Colab notebook.

Questions we seek to answer with the graph analytics:

1. Are there occurrences of co-offending officers on the same misconduct report and how often do the same co-offending officers repeat?
2. What is the salary, rank, and race relationship between the topmost pair of co-offending officers? This analysis can then be repeated for each pair, as needed. For this report, we will provide discussion for the topmost offending pair.
    - do the officers have comparable salaries within 10% of each other?
    - are the officers of similar ranking or is one a higher ranking officer, possibly socially pressuring a lower ranking officer into committing misconduct?
    - are the officers the same race?
3. How many unique co-offending relationships does each officer have? It is straightforward to compute total misconduct of an individual officer but will be more meaningful to understanding how many unique relationships an officer has in which he/she allegedly commits a misconduct.
    - we will use the Triangle Count algorithm to count the unique relationships connecting officer_id nodes
4. Who is the ring-leader (most important) co-offending officer?

```
# install java
!apt-get install openjdk-8-jdk-headless -qq > /dev/null

# install spark (change the version number if needed)
!wget -q https://archive.apache.org/dist/spark/spark-3.2.0/spark-3.2.0-bin-hadoop3.2.tgz

# unzip the spark file to the current folder
!tar xf spark-3.2.0-bin-hadoop3.2.tgz

# set your spark folder to your system path environment.
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.2.0-bin-hadoop3.2"

# install findspark using pip
!pip install -q findspark

# install pyspark
!pip3 install pyspark==3.2.0

# install graphframes
!pip3 install graphframes
```

```
    Requirement already satisfied: pyspark==3.2.0 in /usr/local/lib/python3.7/dist-packages (3.2.0)
    Requirement already satisfied: py4j==0.10.9.2 in /usr/local/lib/python3.7/dist-packages (from pyspark==3.2.0) (0.10.9.2)
    Requirement already satisfied: graphframes in /usr/local/lib/python3.7/dist-packages (0.6)
    Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from graphframes) (1.19.5)
    Requirement already satisfied: nose in /usr/local/lib/python3.7/dist-packages (from graphframes) (1.3.7)
```

Download the graphframes jar file from: Graphframe jar file:

Upload it in the Google Colab Files folder. Can be found in the left pane of this window.

```
!cp -v /content/graphframes-0.8.2-spark3.2-s_2.12.jar $SPARK_HOME/jars/
```

```
    '/content/graphframes-0.8.2-spark3.2-s_2.12.jar' -> '/content/spark-3.2.0-bin-hadoop3.2/jars/graphframes-0.8.2-spark3.2-s_2.1
```

```
#import the packages
from pyspark import *
from pyspark.sql import *
from graphframes import *
import findspark
import pandas as pd

findspark.init()

# Start a Spark session
spark = SparkSession.builder.master("local[*]").getOrCreate()
```

```
import psycopg2
```

```
# access the postgresql server
conn = psycopg2.connect(
    host="codd04.research.northwestern.edu",
    port = "5433",
    database="postgres",
    user="cpdbstudent",
    password="DataSci4AI")
```

```
cursor = conn.cursor()
```

CPDB

Analyze the police officers connection with police misconducts, i.e. (illegal search and use of force)

```
edges_query = "SELECT da1.officer_id src, da2.officer_id dst, COUNT(DISTINCT da1.allegation_id) relationship \
FROM data_officerallegation da1, \
     data_officerallegation da2, \
     data_allegationcategory dcat \
WHERE da1.allegation_id = da2.allegation_id \
  AND da1.allegation_category_id = dcat.id \
  AND da1.officer_id < da2.officer_id \
  AND (dcat.category like 'Illegal Search' or dcat.category like 'Use Of Force') \
GROUP BY da1.officer_id, da2.officer_id \
ORDER BY count(*) DESC;"
```

Following query creates nodes and edges to answer the questions.

- **nodes**: id, officer name and misconduct count
- **edges**: src(officer1 id), dist(officer2 id) and relationship(misconduct count)

```
nodes_query = "SELECT da.officer_id id, doff.first_name || ' ' || doff.last_name officer_name, doff.race, doff.currer
FROM data_officerallegation da, \
data_officer doff, \
data_allegationcategory dcat \
WHERE da.allegation_category_id = dcat.id \
AND doff.id = da.officer_id \
AND (dcat.category like 'Illegal Search' or dcat.category like 'Use Of Force') \
AND current_salary is not null \
GROUP BY da.officer_id, officer_name, doff.race, doff.current_salary, doff.rank; "
```

```
cursor.execute(edges_query)
edges = cursor.fetchall()
print("shape is: " + str(len(edges)))

df_edges = pd.DataFrame(edges)
colnames = [desc[0] for desc in cursor.description]
df_edges.columns = colnames

print(df_edges.shape)
```

```
    shape is: 92794
    (92794, 3)
```

```
cursor.execute(nodes_query)
nodes = cursor.fetchall()
print("shape is: " + str(len(nodes)))

df_nodes = pd.DataFrame(nodes)
colnames = [desc[0] for desc in cursor.description]
df_nodes.columns = colnames

print(df_nodes.shape)
```

```
    shape is: 13951
    (13951, 6)
```

```
edges_·=·spark.createDataFrame(df_edges)
```

```
nodes·=·spark.createDataFrame(df_nodes)
```

```
cpdb·=·GraphFrame(nodes,·edges_)
```

The Results

```
cpdb.vertices.show()
```

```
+---+----------------+-------------+--------------+--------------------+----------------------+
| id|    officer_name|         race|current_salary|                rank|total_misconduct_count|
+---+----------------+-------------+--------------+--------------------+----------------------+
|  1|   Jeffery Aaron|        White|        101442|  Sergeant of Police|                     2|
|  2|    Karina Aaron|     Hispanic|         94122|Police Officer as...|                     4|
|  4|   Carmel Abbate|        White|         74946|Police Officer as...|                     2|
|  6|  Anthony Abbate|        White|         70656|      Police Officer|                     2|
|  7|    Terry Abbate|        White|         93354|      Police Officer|                     3|
|  8|      Leon Abbey|        Black|         73116|      Police Officer|                     1|
| 11|    Laura Abbott|        White|         73476|Police Officer as...|                     2|
| 13|     Dale Abbott|        White|         85278|      Police Officer|                     2|
| 14|Elizabeth Abbott|        White|         82878|      Police Officer|                     1|
| 16|Aziz Abdelmajeid|Asian/Pacific|         84054|  Sergeant of Police|                     9|
| 17| Moulay Abdullah|        Black|         83706|      Police Officer|                     1|
| 18|   Jason Abejero|Asian/Pacific|         90024|      Police Officer|                     1|
| 20|   Kenneth Abels|        White|        106068|  Sergeant of Police|                     2|
| 33|   Ricardo Abreu|     Hispanic|         74946|Police Officer as...|                    10|
| 34|     Floyd Abron|        Black|         90024|      Police Officer|                     5|
| 38|Abdalla Abuzanat|Asian/Pacific|         97440|Police Officer as...|                     5|
| 39|Rosemary Accardo|        White|         92316|      Police Officer|                    10|
| 41|Jennifer Accardo|        White|         87006|      Police Officer|                     2|
| 42|  Thomas Accardo|        White|         90024|      Police Officer|                     6|
| 44|   Marco Acevedo|     Hispanic|        100980|Police Officer as...|                    10|
+---+----------------+-------------+--------------+--------------------+----------------------+
only showing top 20 rows
```

Question 1: Are there occurrences of co-offending officers on the same misconduct report and how often do the same co-offending officers repeat?

The table below represents a graph with source node being officer_id1 and destination node being officer_id2, while their relationship is the total count of co-offending misconduct.

```
cpdb.edges.show()
```

```
+-----+-----+------------+
|  src|  dst|relationship|
+-----+-----+------------+
|12478|32166|          36|
| 8562|27778|          34|
| 2725|21703|          29|
| 1553|10724|          28|
| 3605|14442|          28|
| 8562|18206|          28|
|12074|12825|          28|
|32265|32347|          27|
| 8562|23841|          26|
|31882|32401|          25|
|13361|20150|          25|
| 1553|16699|          24|
|23841|27778|          24|
|32016|32213|          24|
|14731|27602|          23|
|14045|15502|          23|
|12479|20713|          22|
|17285|17397|          21|
|18206|27778|          21|
| 8658|13788|          21|
+-----+-----+------------+
only showing top 20 rows
```

Question 2: What is the salary, rank, and race relationship between the topmost pair of co-offending officers?

- do the officers have comparable salaries within 10% of each other?
- are the officers of similar ranking or is one a higher ranking officer, possibly socially pressuring a lower ranking officer into committing misconduct?
- are the officers the same race?

To gain more insight into the top most co-offending pair of officers, we inspect the graph vertices to see the exact names, salary, rank, and race of the officers with the most total misconduct counts. As we can see below, the officers Ronald Holt and Emmet Mc Clendon have the most co-offending misconduct complaints. There is a more than 10% discrepency between their salaries and their officer rankings are significantly different, implying some hierarchical relationship between officer Holt and officer McClendon. Further, we note that both officers in this pair are of race black.

```
cpdb.vertices.filter('id=12478').show()
cpdb.vertices.filter('id=32166').show()
```

```
+-----+------------+-----+--------------+---------------+---------------------+
+-----+------------+-----+--------------+----------------+----------------------+
```

```
+-----+-----------------+-----+--------------+------------------+----------------------+
|   id|     officer_name| race|current_salary|              rank|total_misconduct_count|
+-----+-----------------+-----+--------------+------------------+----------------------+
|32166|Emmett Mc Clendon|Black|        107988|Sergeant of Police|                    64|
+-----+-----------------+-----+--------------+------------------+----------------------+
```

Question 3: How many unique co-offending relationships does each officer have?

Background on the Triangle Count algorithm:

The Triangle Count algorithm counts the number of triangles for each node in the graph. A triangle is a set of three nodes where each node has a relationship to the other two. In graph theory terminology, this is sometimes referred to as a 3-clique. The Triangle Count algorithm in the GDS library only finds triangles in undirected graphs.

Triangle counting has gained popularity in social network analysis, where it is used to detect communities and measure the cohesiveness of those communities.

(source: https://neo4j.com/docs/graph-data-science/current/algorithms/triangle-count/#:~:text=The%20Triangle%20Count%20algorithm%20counts,to%20as%20a%203%2Dclique):

We see from our triangle count computation on the graph that officer Vincent Stinar has the highest number of unique co-offending relationships with other officers. And although we understood the Triangle Count algorithm to compute the unique relationship each officer has, as described above, the computation result we get from the code does not make immediate intuitive sense. We do a sanity-check on the computation by printing out the noted officer id '32356' as being the officer with the most unique co-offending relationships, being officer Vincent Stinar. We note that the TC algorithm computes he has 1514 unique co-offending relationships but the following computation for officer Stinar's total misconduct count is much lower, at total count equal to 12. This does not make intuitive sense for officer Vincent Stinar to have 1514 unique co-offending relationships but only 12 counts of misconduct. Upon further investigation and debugging of the code, we are unable to identify any specific bug in the code to rectify this error.

[Edit]: We previously noted that officer_id 32356, Officer Vincent Stinar, has the greatest number of co-offending relationships as analyzed by the trianglecount algorithm. It was also noted that these are unique co-offending triangular relationships and that it did not make intuitive sense to have such a large trianglecount result, 1514, while having only 12 total counts of misconduct allegations. After further review of the Triangle Count algorithm, we find that the analysis is not of *unique* co-offending relationships, but repeated count of relationships. This would then mean it is reasonable to see a larger triangle count result even if the total misconduct count is much lower.

```
tc_cpdb = cpdb.triangleCount()

tc_cpdb.select("id", "count").sort(['count'], ascending=[0]).show()
cpdb.vertices.filter('id=32356').show()
```

```
+-----+-----+
|   id|count|
+-----+-----+
|32356| 1514|
|31536| 1485|
|32390| 1437|
|25230| 1417|
|22554| 1390|
| 2375| 1369|
| 6704| 1366|
|21364| 1347|
|30337| 1316|
|25983| 1312|
| 2201| 1287|
|13272| 1277|
|13093| 1263|
|10724| 1237|
|28384| 1230|
| 9648| 1204|
| 7032| 1196|
|12947| 1186|
| 6852| 1182|
| 2356| 1179|
+-----+-----+
only showing top 20 rows

+-----+--------------+-----+--------------+--------------+----------------------+
|   id|  officer_name| race|current_salary|          rank|total_misconduct_count|
+-----+--------------+-----+--------------+--------------+----------------------+
|32356|Vincent Stinar|White|         90024|Police Officer|                    12|
+-----+--------------+-----+--------------+--------------+----------------------+
```

da1.officer_id < da2.officer_id" is modified to "AND da1.officer_id <> da2.ifficer_id".

```
edges_query = "SELECT da1.officer_id src, da2.officer_id dst, COUNT(DISTINCT da1.allegation_id) relationship \
FROM data_officerallegation da1, \
     data_officerallegation da2, \
     data_allegationcategory dcat \
WHERE da1.allegation_id = da2.allegation_id \
  AND da1.allegation_category_id = dcat.id \
  AND da1.officer_id <> da2.officer_id \
  AND (dcat.category like 'Illegal Search' or dcat.category like 'Use Of Force') \
GROUP BY da1.officer_id, da2.officer_id \
ORDER BY count(*) DESC;"
```

```
nodes_query = "SELECT da.officer_id id, doff.first_name || ' ' || doff.last_name officer_name, doff.race, doff.currer
FROM data_officerallegation da, \
data_officer doff, \
data_allegationcategory dcat \
WHERE da.allegation_category_id = dcat.id \
AND doff.id = da.officer_id \
AND (dcat.category like 'Illegal Search' or dcat.category like 'Use Of Force') \
AND current_salary is not null \
GROUP BY da.officer_id, officer_name, doff.race, doff.current_salary, doff.rank; "
```

```
cursor.execute(edges_query)
edges = cursor.fetchall()
print("shape is: " + str(len(edges)))

df_edges = pd.DataFrame(edges)
colnames = [desc[0] for desc in cursor.description]
df_edges.columns = colnames

print(df_edges.shape)
```

```
shape is: 185528
(185528, 3)
```

```
edges_ = spark.createDataFrame(df_edges)
```

```
nodes = spark.createDataFrame(df_nodes)
```

```
cpdb = GraphFrame(nodes, edges_)
```

New graph for use in PageRank:

```
cpdb.vertices.show()
```

```
+---+----------------+-------------+--------------+--------------------+----------------------+
| id|    officer_name|         race|current_salary|                rank|total_misconduct_count|
+---+----------------+-------------+--------------+--------------------+----------------------+
|  1|   Jeffery Aaron|        White|        101442|  Sergeant of Police|                     2|
|  2|    Karina Aaron|     Hispanic|         94122|Police Officer as...|                     4|
|  4|   Carmel Abbate|        White|         74946|Police Officer as...|                     2|
|  6|  Anthony Abbate|        White|         70656|      Police Officer|                     2|
|  7|    Terry Abbate|        White|         93354|      Police Officer|                     3|
|  8|      Leon Abbey|        Black|         73116|      Police Officer|                     1|
| 11|    Laura Abbott|        White|         73476|Police Officer as...|                     2|
| 13|     Dale Abbott|        White|         85278|      Police Officer|                     2|
| 14|Elizabeth Abbott|        White|         82878|      Police Officer|                     1|
| 16|Aziz Abdelmajeid|Asian/Pacific|         84054|  Sergeant of Police|                     9|
| 17| Moulay Abdullah|        Black|         83706|      Police Officer|                     1|
| 18|   Jason Abejero|Asian/Pacific|         90024|      Police Officer|                     1|
| 20|   Kenneth Abels|        White|        106068|  Sergeant of Police|                     2|
| 33|   Ricardo Abreu|     Hispanic|         74946|Police Officer as...|                    10|
| 34|     Floyd Abron|        Black|         90024|      Police Officer|                     5|
| 38|Abdalla Abuzanat|Asian/Pacific|         97440|Police Officer as...|                     5|
| 39|Rosemary Accardo|        White|         92316|      Police Officer|                    10|
| 41|Jennifer Accardo|        White|         87006|      Police Officer|                     2|
| 42|  Thomas Accardo|        White|         90024|      Police Officer|                     6|
| 44|   Marco Acevedo|     Hispanic|        100980|Police Officer as...|                    10|
+---+----------------+-------------+--------------+--------------------+----------------------+
only showing top 20 rows
```

```
cpdb.edges.show()
```

```
|27778| 8562|          34|
| 2725|21703|          29|
|21703| 2725|          29|
|18206| 8562|          28|
|12074|12825|          28|
|12825|12074|          28|
|14442| 3605|          28|
| 8562|18206|          28|
|10724| 1553|          28|
| 3605|14442|          28|
| 1553|10724|          28|
|32265|32347|          27|
|32347|32265|          27|
|23841| 8562|          26|
| 8562|23841|          26|
|13361|20150|          25|
|31882|32401|          25|
+-----+-----+------------+
only showing top 20 rows
```

Result of ring leader analysis using the PageRank algorithm

We use the PageRank algorithm on our graph to highlight which officers perform as ring leaders in the perspective of the data.

As visualized above, the pagerank algorithm shows that officer Glenn Evans has the most influence on other officers in co-offending misconduct cases with a PageRank of 9.22 and total misconduct count of 73. Although at first glance the total misconduct count may seem less than other entries, we must note that PageRank will highlight which officer has the most commonality with other officers in committing misconduct offenses; thereby being "ring leaders". Other officers may commit additional misconduct offenses separately as individuals and thereby increasing their overall misconduct count.

To further highlight this point, note that Officer Jerome Finnigan, officer_id 8562, has the highest total of misconduct allegations yet is only the 4th highest ranking "ring-leader" as analyzed by PageRank.

```
pr_cpdb = cpdb.pageRank(resetProbability=0.15, tol=0.01)
#look at the pagerank score for every vertex
pr_cpdb.vertices.orderBy('pagerank', ascending=False).show()
```

```
+-----+------------------+--------+--------------+--------------------+----------------------+------------------+
|   id|      officer_name|    race|current_salary|                rank|total_misconduct_count|          pagerank|
+-----+------------------+--------+--------------+--------------------+----------------------+------------------+
| 8138|       Glenn Evans|   Black|        125190|Lieutenant of Police|                    73| 9.224336490995743|
|17816|        Edward May|   White|         86130|      Police Officer|                    91| 8.781453450615974|
|13303|     David Jarmusz|   White|        127596|           Commander|                    29| 7.753717937769441|
| 8562|   Jerome Finnigan|   White|         73116|      Police Officer|                   116| 7.205109026215454|
|32255|     Gerardo Perez|Hispanic|         93354|      Police Officer|                    38| 6.999806910246934|
| 2375|Marvin Bonnstetter|   White|        101958|  Sergeant of Police|                    52| 6.925574000926473|
|16567|    Baudilio Lopez|Hispanic|        111474|  Sergeant of Police|                    55| 6.727067377907265|
| 9821|       Mark George|   White|        111474|  Sergeant of Police|                    41| 6.375518197546371|
|27392|     Robert Stasch|   White|        125190|Lieutenant of Police|                    23|6.3229051514833765|
|31859|         Eric Cato|   Black|        111474|  Sergeant of Police|                    51| 6.255302627009381|
|20959|     James O Grady|   White|        154932|           Commander|                    40| 5.905031155488198|
|16699|        John Lucid|   White|        111474|  Sergeant of Police|                    36| 5.880121795999848|
|25306|     James Sanchez|Hispanic|        162684|Lieutenant of Police|                    69| 5.878604323220897|
|31834|   Michael Bocardo|Hispanic|        111474|  Sergeant of Police|                    37|5.8610254549394565|
|10528|      Bernard Graf|   White|         87354|Police Officer as...|                    35| 5.761828279300101|
|27778|     Carl Suchocki|   White|         90024|      Police Officer|                    55|5.7371383579010855|
|32164|   Tamara Matthews|   Black|         93354|  Sergeant of Police|                    62| 5.699532705181115|
|29445|         Luis Vega|Hispanic|         93240|      Police Officer|                    36| 5.692547823980591|
|27270|     Michael Stack|   White|        107988|  Sergeant of Police|                    40| 5.550108151684686|
|32237|    Louis Ortoneda|Hispanic|         93354|      Police Officer|                    38| 5.522336960533268|
+-----+------------------+--------+--------------+--------------------+----------------------+------------------+
only showing top 20 rows
```