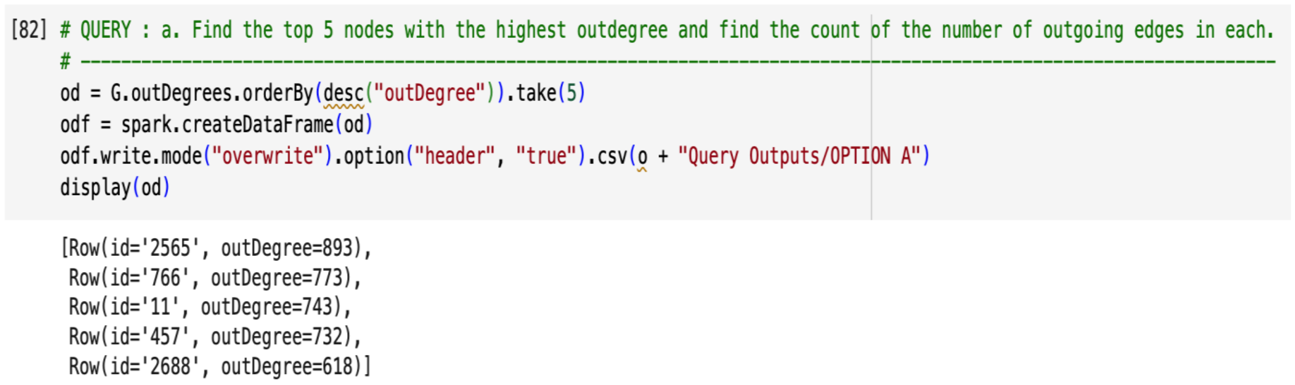
QUESTION 2 :

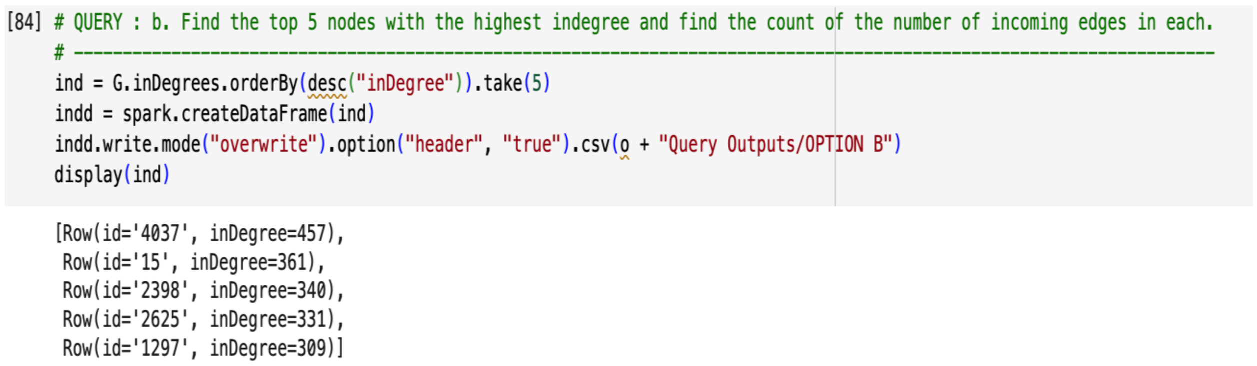
1. Find the top 5 nodes with the highest outdegree and and the count of the number of outgoing edges in each



```
[82] # QUERY : a. Find the top 5 nodes with the highest outdegree and find the count of the number of outgoing edges in each.
     # -----------------------------------------------------------------------------------------------------------------------
     od = G.outDegrees.orderBy(desc("outDegree")).take(5)
     odf = spark.createDataFrame(od)
     odf.write.mode("overwrite").option("header", "true").csv(o + "Query Outputs/OPTION A")
     display(od)

     [Row(id='2565', outDegree=893),
      Row(id='766', outDegree=773),
      Row(id='11', outDegree=743),
      Row(id='457', outDegree=732),
      Row(id='2688', outDegree=618)]
```

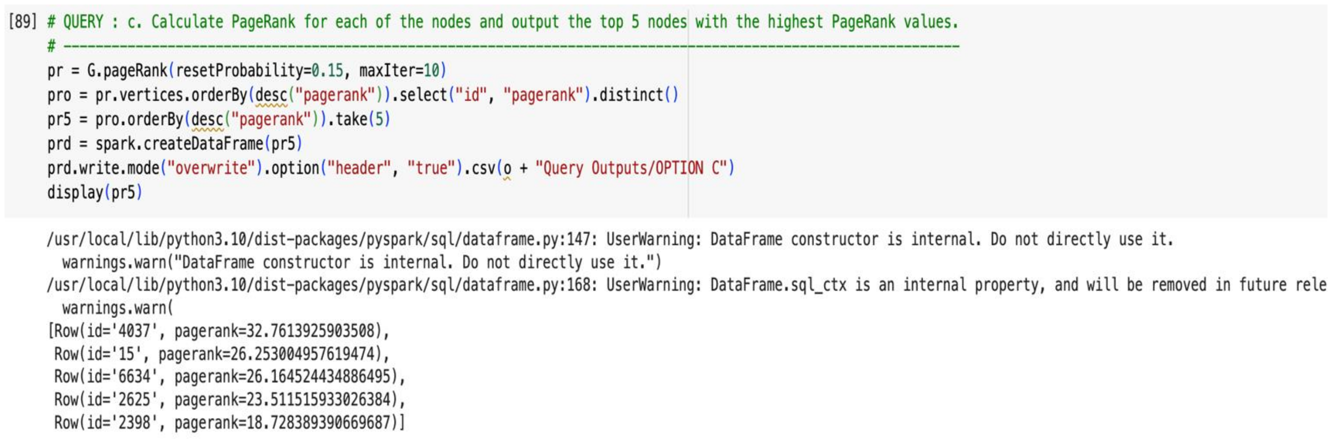
2. Find the top 5 nodes with the highest indegree and nd the count of the number of incoming edges in each



```
[84] # QUERY : b. Find the top 5 nodes with the highest indegree and find the count of the number of incoming edges in each.
     # -----------------------------------------------------------------------------------------------------------------------
     ind = G.inDegrees.orderBy(desc("inDegree")).take(5)
     indd = spark.createDataFrame(ind)
     indd.write.mode("overwrite").option("header", "true").csv(o + "Query Outputs/OPTION B")
     display(ind)

     [Row(id='4037', inDegree=457),
      Row(id='15', inDegree=361),
      Row(id='2398', inDegree=340),
      Row(id='2625', inDegree=331),
      Row(id='1297', inDegree=309)]
```

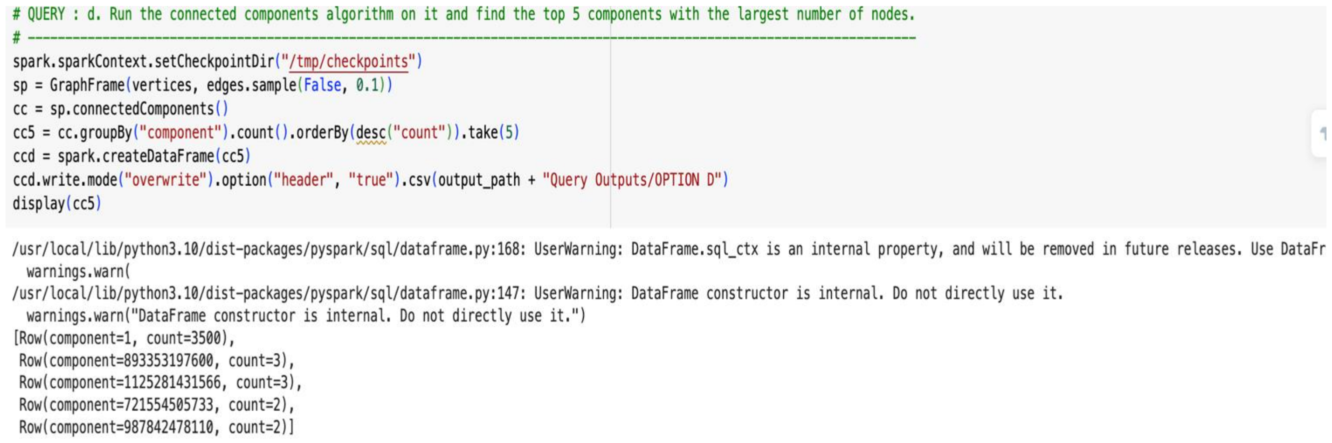
3. Calculate PageRank for each of the nodes and output the top 5 nodes with the highest PageRank values. You are free to define any suitable parameters.



```
[89] # QUERY : c. Calculate PageRank for each of the nodes and output the top 5 nodes with the highest PageRank values.
     # ---------------------------------------------------------------------------------------------------------------
     pr = G.pageRank(resetProbability=0.15, maxIter=10)
     pro = pr.vertices.orderBy(desc("pagerank")).select("id", "pagerank").distinct()
     pr5 = pro.orderBy(desc("pagerank")).take(5)
     prd = spark.createDataFrame(pr5)
     prd.write.mode("overwrite").option("header", "true").csv(o + "Query Outputs/OPTION C")
     display(pr5)

     /usr/local/lib/python3.10/dist-packages/pyspark/sql/dataframe.py:147: UserWarning: DataFrame constructor is internal. Do not directly use it.
       warnings.warn("DataFrame constructor is internal. Do not directly use it.")
     /usr/local/lib/python3.10/dist-packages/pyspark/sql/dataframe.py:168: UserWarning: DataFrame.sql_ctx is an internal property, and will be removed in future rele
       warnings.warn(
     [Row(id='4037', pagerank=32.7613925903508),
      Row(id='15', pagerank=26.253004957619474),
      Row(id='6634', pagerank=26.164524434886495),
      Row(id='2625', pagerank=23.511515933026384),
      Row(id='2398', pagerank=18.728389390669687)]
```

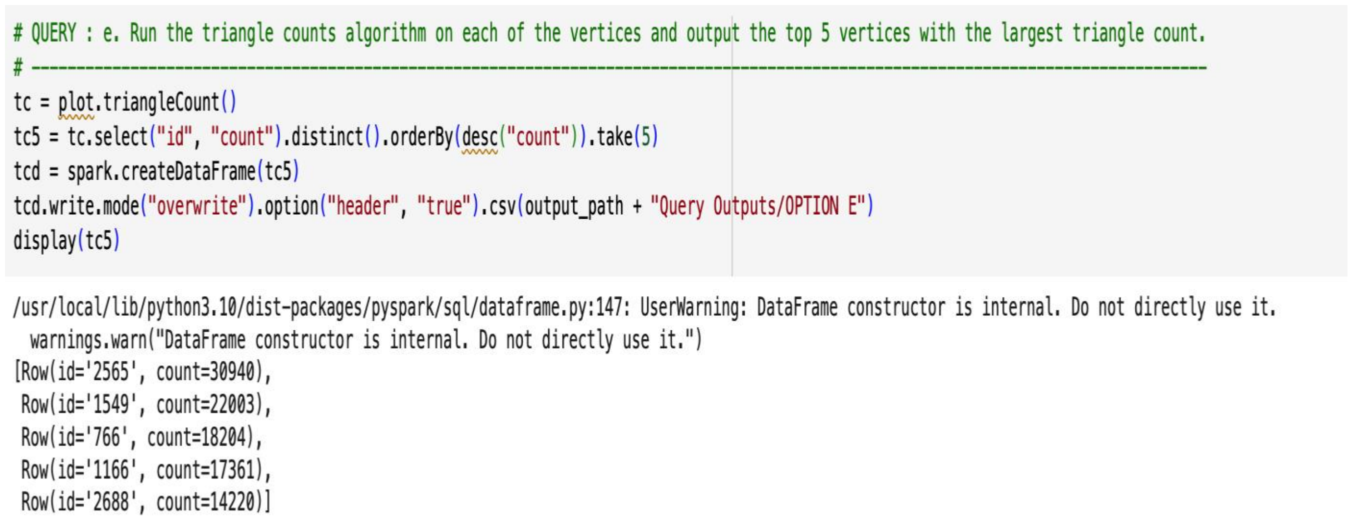
4. Run the connected components algorithm on it and nd the top 5 components with the largest number of nodes.



```
# QUERY : d. Run the connected components algorithm on it and find the top 5 components with the largest number of nodes.
# ------------------------------------------------------------------------------------------------------------------------
spark.sparkContext.setCheckpointDir("/tmp/checkpoints")
sp = GraphFrame(vertices, edges.sample(False, 0.1))
cc = sp.connectedComponents()
cc5 = cc.groupBy("component").count().orderBy(desc("count")).take(5)
ccd = spark.createDataFrame(cc5)
ccd.write.mode("overwrite").option("header", "true").csv(output_path + "Query Outputs/OPTION D")
display(cc5)
```

```
/usr/local/lib/python3.10/dist-packages/pyspark/sql/dataframe.py:168: UserWarning: DataFrame.sql_ctx is an internal property, and will be removed in future releases. Use DataFr
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/pyspark/sql/dataframe.py:147: UserWarning: DataFrame constructor is internal. Do not directly use it.
  warnings.warn("DataFrame constructor is internal. Do not directly use it.")
[Row(component=1, count=3500),
 Row(component=893353197600, count=3),
 Row(component=1125281431566, count=3),
 Row(component=721554505733, count=2),
 Row(component=987842478110, count=2)]
```

5. Run the triangle counts algorithm on each of the vertices and output the top 5 vertices with the largest triangle count. In case of ties, you can randomly select the top 5 vertices.



```
# QUERY : e. Run the triangle counts algorithm on each of the vertices and output the top 5 vertices with the largest triangle count.
# ------------------------------------------------------------------------------------------------------------------------
tc = plot.triangleCount()
tc5 = tc.select("id", "count").distinct().orderBy(desc("count")).take(5)
tcd = spark.createDataFrame(tc5)
tcd.write.mode("overwrite").option("header", "true").csv(output_path + "Query Outputs/OPTION E")
display(tc5)
```

```
/usr/local/lib/python3.10/dist-packages/pyspark/sql/dataframe.py:147: UserWarning: DataFrame constructor is internal. Do not directly use it.
  warnings.warn("DataFrame constructor is internal. Do not directly use it.")
[Row(id='2565', count=30940),
 Row(id='1549', count=22003),
 Row(id='766', count=18204),
 Row(id='1166', count=17361),
 Row(id='2688', count=14220)]
```

SUMMARY :-

1. The analysis shows that the nodes with the highest outdegree have the most outgoing edges, indicating that they are highly well-liked or important inside the network.
   The highest outdegree is id-2565.

2. Similar to this, nodes with the highest indegree also have the most incoming edges, demonstrating their significance and strong connections.
   The highest indegree is id-4037.
3. The most important or core nodes in the network can be identified using the PageRank algorithm. The most important and influential nodes in the network are probably the ones with the highest PageRank ratings.
   Out of all the nodes, id-4037 has the greatest PageRank.
4. Finding clusters or communities within a network is made easier with the help of the connected component method.
   Node 1 has the highest count among the top 5 connected components, whereas the next four have much lower counts.
5. Lastly, the triangle counts method can help find nodes that are involved in multiple triangles and have a strong link, which can provide information about how cliques or sub-communities form inside the network.
   The maximum triangular count, 30940, is found in id-2565.