Project: PySpark: DataFrames / SparkSQL + GraphFrames / GraphX

Student :

Presented by MANICKAM RAVISEKAR ,
Master of Science in Computer Science, 19599 , Fall Semester 2022

Professor : Dr Henry Chung
TA : Liang

SAN FRANCISCO BAY
UNIVERSITY
47671 WestingHouse Dr.,
Fremont, CA 94539

ACKNOWLEDGEMENT

One of our master's degree Project for PySpark: DataFrames / SparkSQL + GraphFrames / GraphX,

Is an Interesting, which made me to learn new things, it is useful in designing and applying using Scala, PySpark: DataFrames / SparkSQL + GraphFrames / GraphX.

For deploying this project, I would like to thank Dr. Henry Chang an TA Liang for providing all the required input .

Also, for all I would like to always pray to Almighty for giving us wisdom and power to understand things.

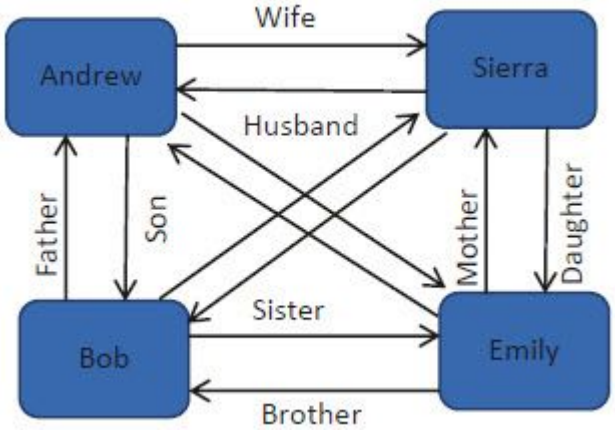
Content

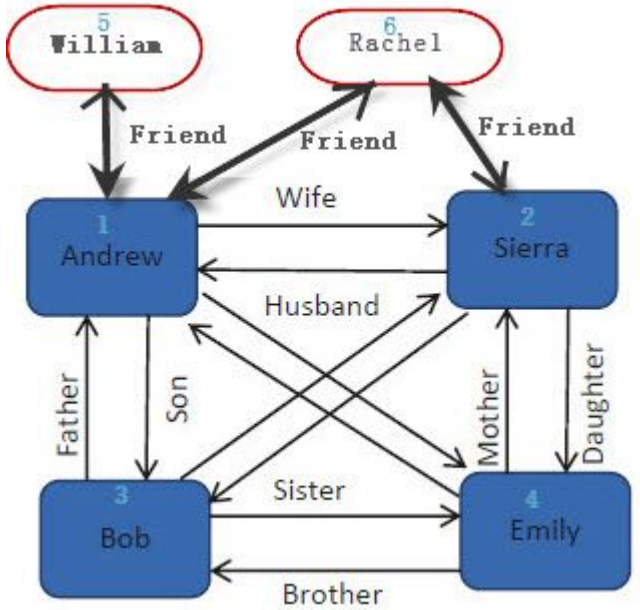
Index :

Abstract

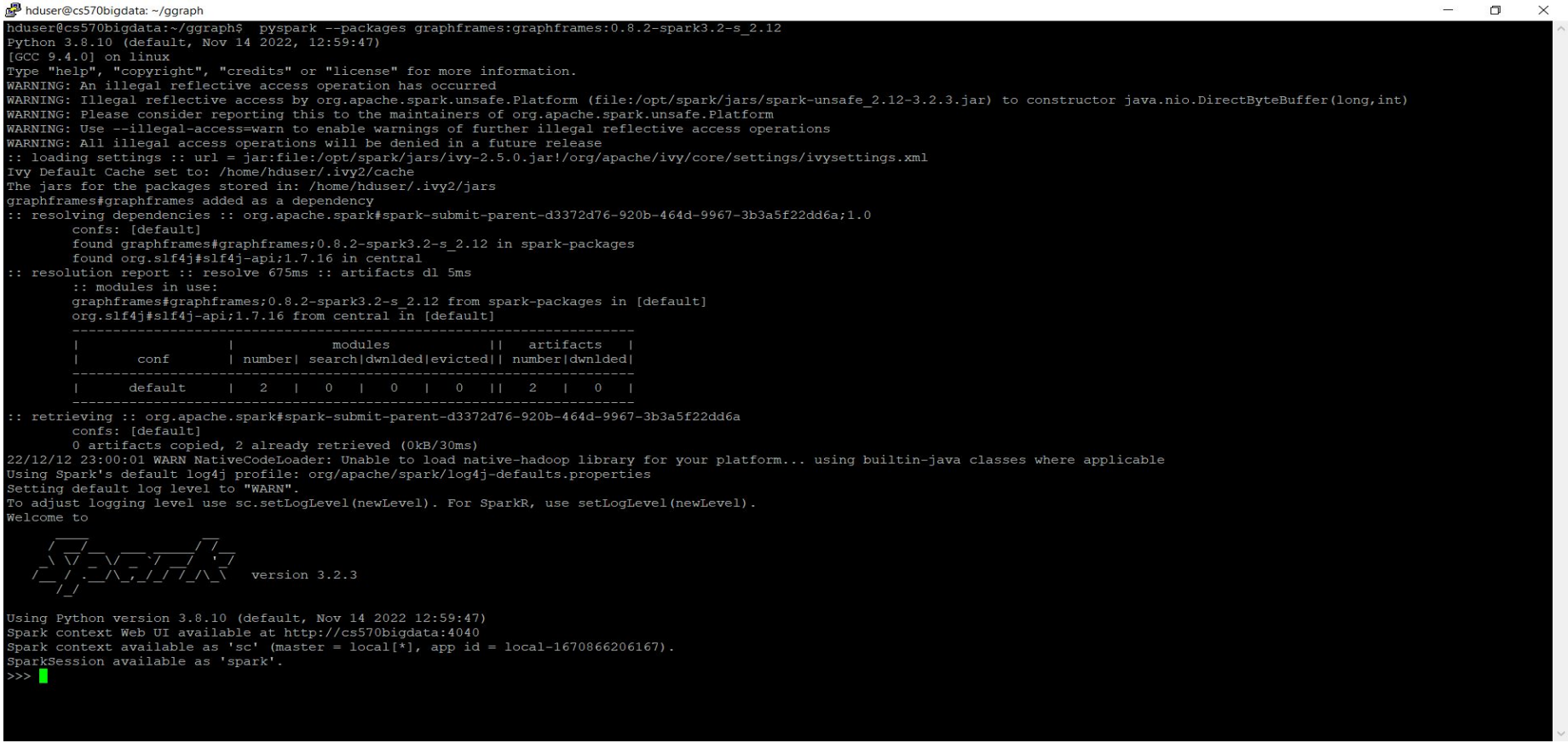
Graph Frames are an abstraction of Data Frames that are used to do Graph Analytics. Graph Analytics stems from the mathematical Graph Theory. Graph Theory is an especially important theory used to be relationships between entities, which we can use to perform various analyses. You are using Graph Theory in your everyday life when using Google. Google introduced the PageRank algorithm that is based on Graph Theory. It tries to show the most influential website that suits your search in the best way

Dataset For Family and Friend's Relationship network



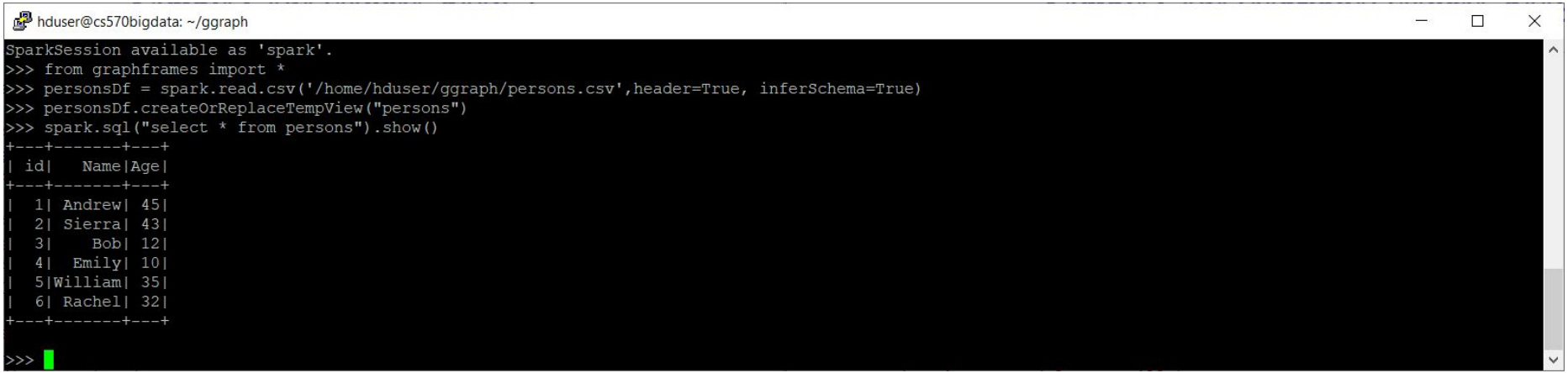


pyspark --packages graphframes:graphframes:0.8.2-spark3.2-s_2.12



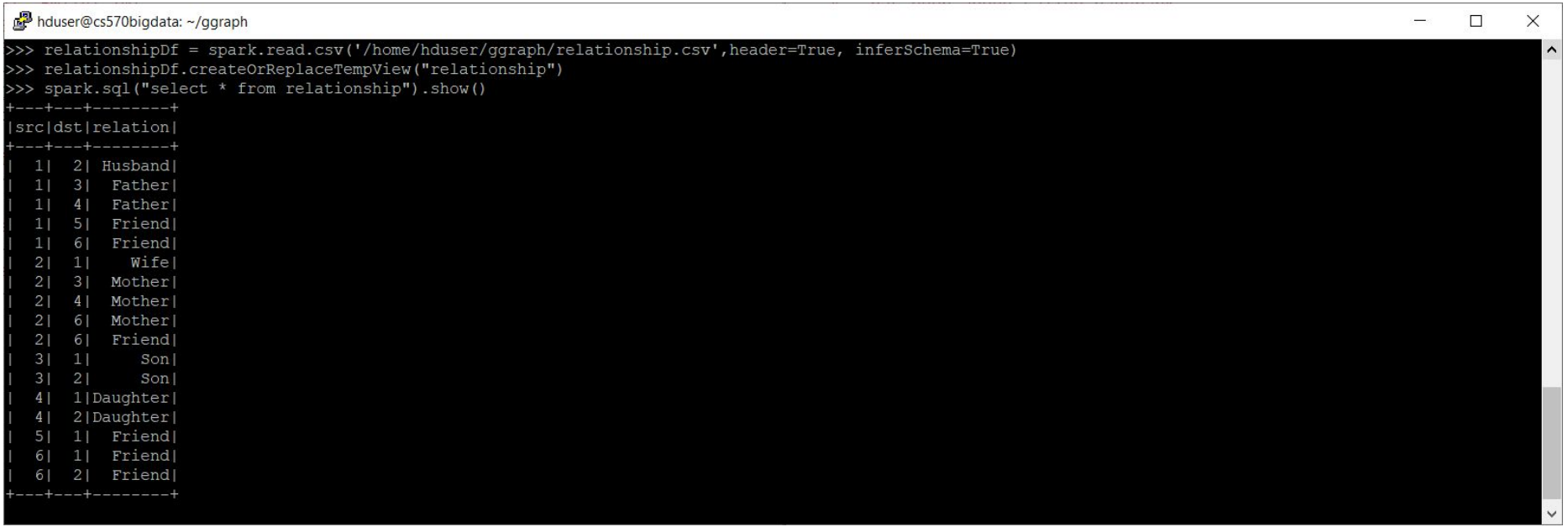
```
#######################################################
# Create GraphFrames
#######################################################

#######################################################
#     person dataframe : id, Name, age
#######################################################
```



```
# relationship dataframe : src, dst, relation
relationshipDf = spark.read.csv('/home/hduser/ggraph/relationship.csv',header=True, inferSchema=True)

relationshipDf.createOrReplaceTempView("relationship")
spark.sql("select * from relationship").show()
```



```
# - Create a GraphFrame from both person and relationship dataframes
#     >>> graph
#     GraphFrame(v:[id: int, Name: string ... 1 more field], e:[src:
#     int, dst: int ... 1 more field])
# - A GraphFrame that contains v and e.
#   + The v represents vertices and e represents edges.
graph = GraphFrame(personsDf, relationshipDf)
```

```
# - Degrees represent the number of edges that are connected to a vertex.
#   + GraphFrame supports inDegrees and outDegrees.
#     - inDegrees give you the number of incoming links to a vertex.
#     - outDegrees give the number of outgoing edges from a node.
# - Find all the edges connected to Andrew.
#     +--+------+
#     |id|degree|
#     +--+------+
#     | 1|    10|
#     +--+------+
graph.degrees.filter("id = 1").show()
```

```
# Find the number of incoming links to Andrew
#     +--+--------+
#     |id|inDegree|
#     +--+--------+
#     | 1|       5|
#     +--+--------+
```

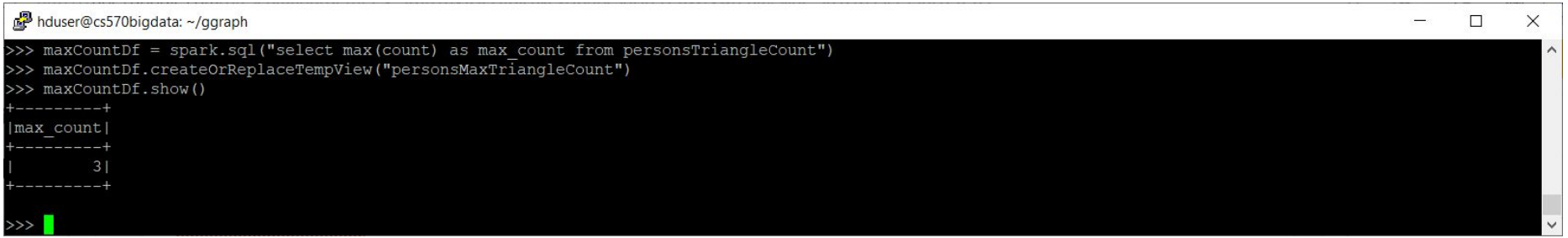
```
graph.inDegrees.filter("id = 1").show()
```

```
# Find the number of links coming out from Andrew using the outDegrees
#     +--+---------+
#     |id|outDegree|
#     +--+---------+
#     | 1|        5|
#     +--+---------+
graph.outDegrees.filter("id = 1").show()
```

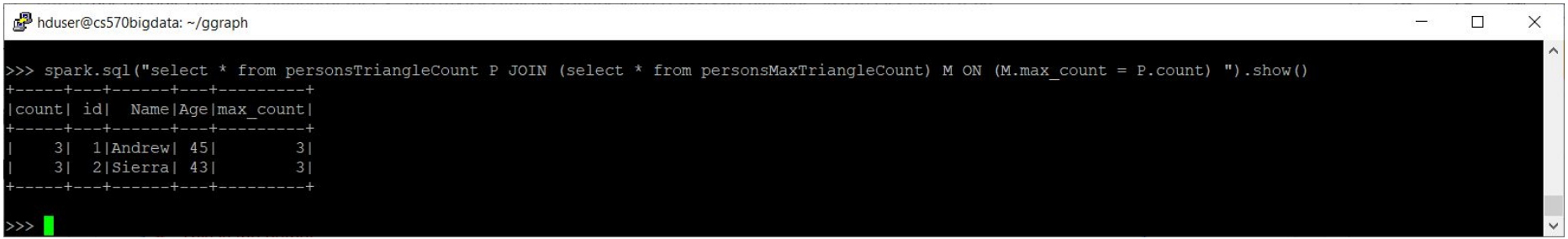
```
#########################################################################
# Apply Triangle Counting in a GraphFrame
#########################################################################
# - Find how many triangle relationships the vertex is participating in
# - A vertex is part of a triangle when it has two adjacent vertices with an
#   edge between them. For example, Andrew has 3 triangle count.
#   + Andrew - Sierra - Bob
#   + Andrew - Emily - Bib
#   + Andrew - Sierra - Emily
# - A new column count is added in the output that represents the triangle count.
#   + The output shows that Andrew and Sierra have the maximum triangle counts,
#     since they are involved in 3 kinds of relationships
#     - Andrew as father, friend, and husband and Sierra as mother,
#       friend, and wife.
```

```
# Create a "personsTriangleCount" SQL table from the
# personsTriangleCountDf DataFrame
personsTriangleCountDf.createOrReplaceTempView("personsTriangleCount")
```


```
maxCountDf = spark.sql("select max(count) as max_count from personsTriangleCount")
maxCountDf.createOrReplaceTempView("personsMaxTriangleCount")
```



```
# - Join  with the Persons DataFrame to be able to
#   view the person's details
# - This is the output
#    +-----+---+------+---+---------+
#    |count| id| Name |Age|max_count|
#    +-----+---+------+---+---------+
#    |    3|  1|Andrew| 45|        3|
#    |    3|  2|Sierra| 43|        3|
#    +-----+---+------+---+---------+
```



Conclusion

Graph Theory is used in various sciences, computer science also tends to solve a lot of problems with Graph Theory. Some of the applications of Graph Theory include social media problems, travel, chip design, and many other fields.

Social media app, such as LinkedIn, which needs to connect millions of people eventually it will create enormous number of edges. To be able to apply analytics on this kind of data, regular databases will not suffice. With a regular database, we would need to apply self-joins so many times and applying self-joins will literally bring our database down so the optimal solution is graph theory application.