

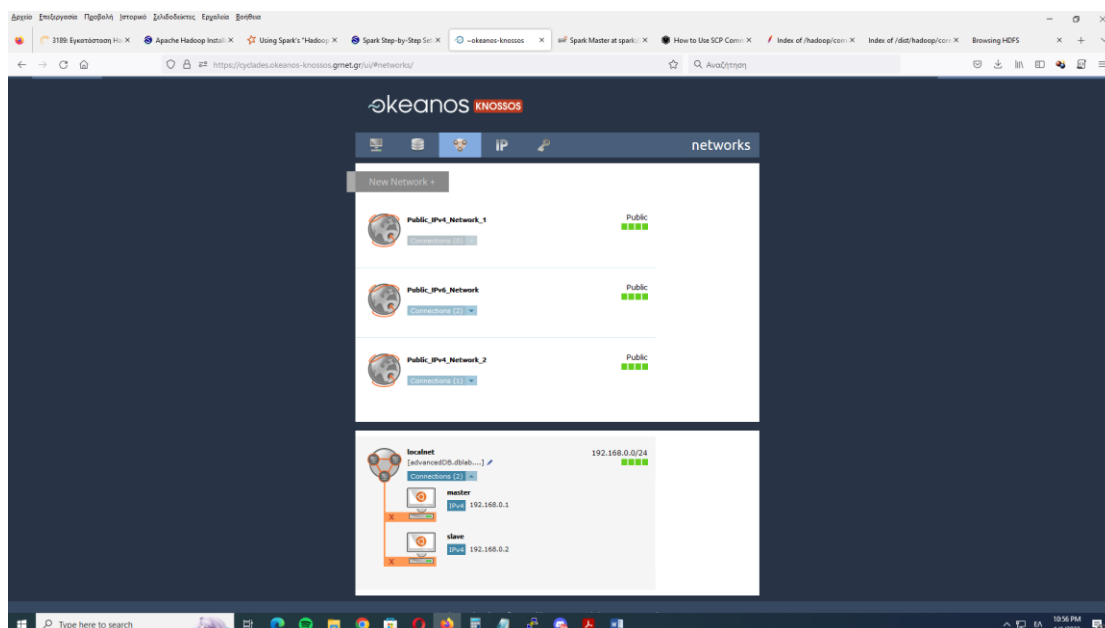
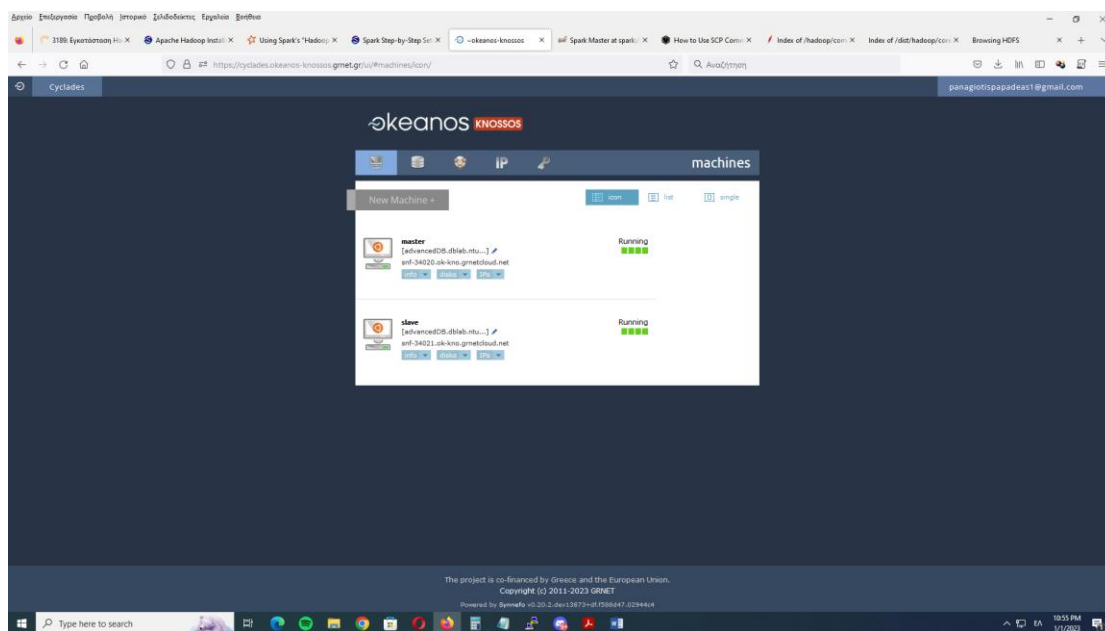
Προχωρημένα θέματα βάσεων δεδομένων

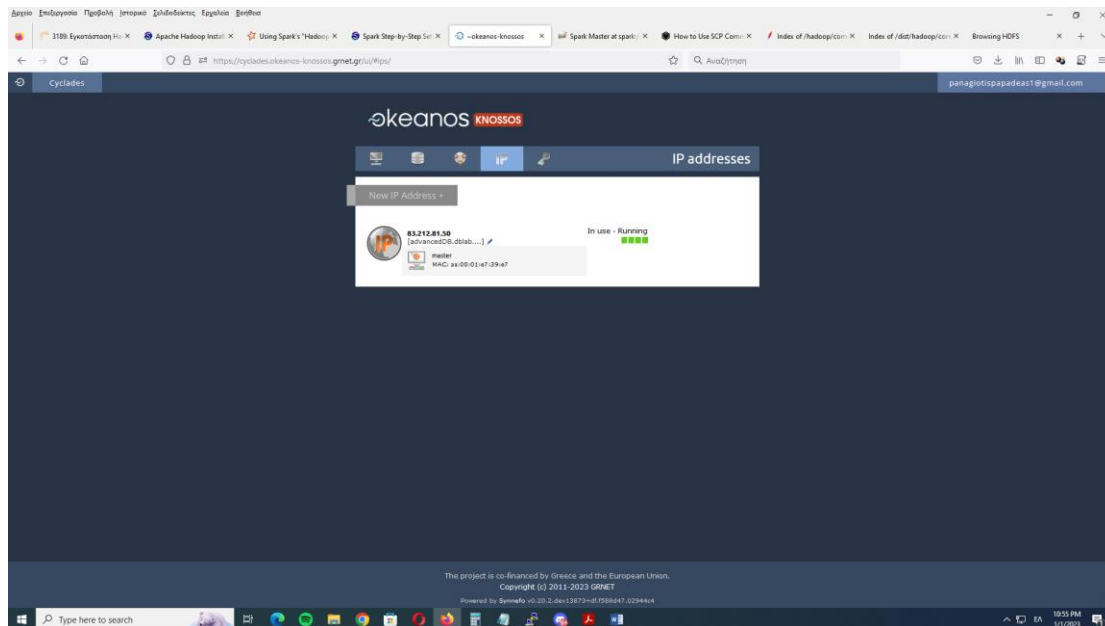
Εξαμηνιαία εργασία

ΠΑΝΑΓΙΩΤΗΣ ΠΑΠΑΔΕΑΣ A.M. 03118039

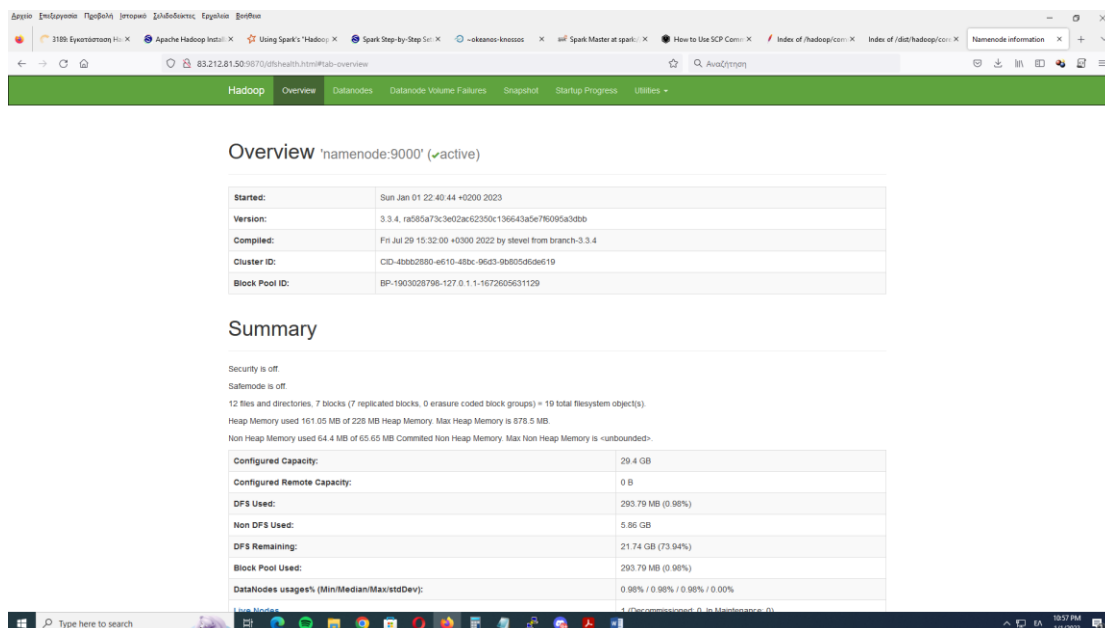
ΚΡΙΣ ΚΟΥΤΣΗ A.M. 03118905

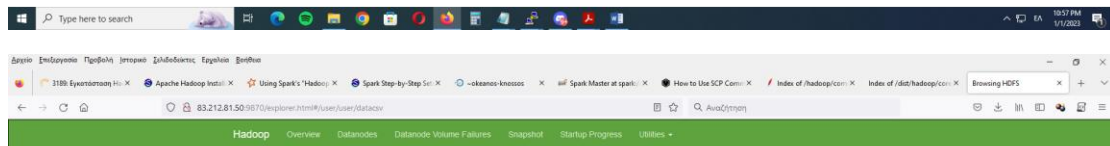
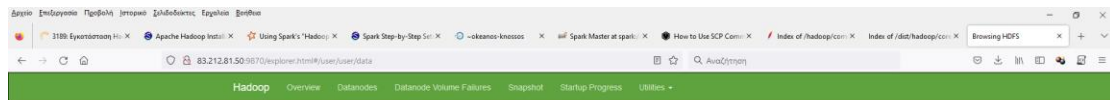
Αρχικά κατασκευάζουμε τα vms στο okeanos σύμφωνα με τις οδηγίες του pdf.
Δημιουργούμε ένα master node και ένα slave, τα οργανώνουμε σε τοπικό δίκτυο για να μπορούν να επικοινωνούν μεταξύ τους και αναθέτουμε μια public IP στο master node.



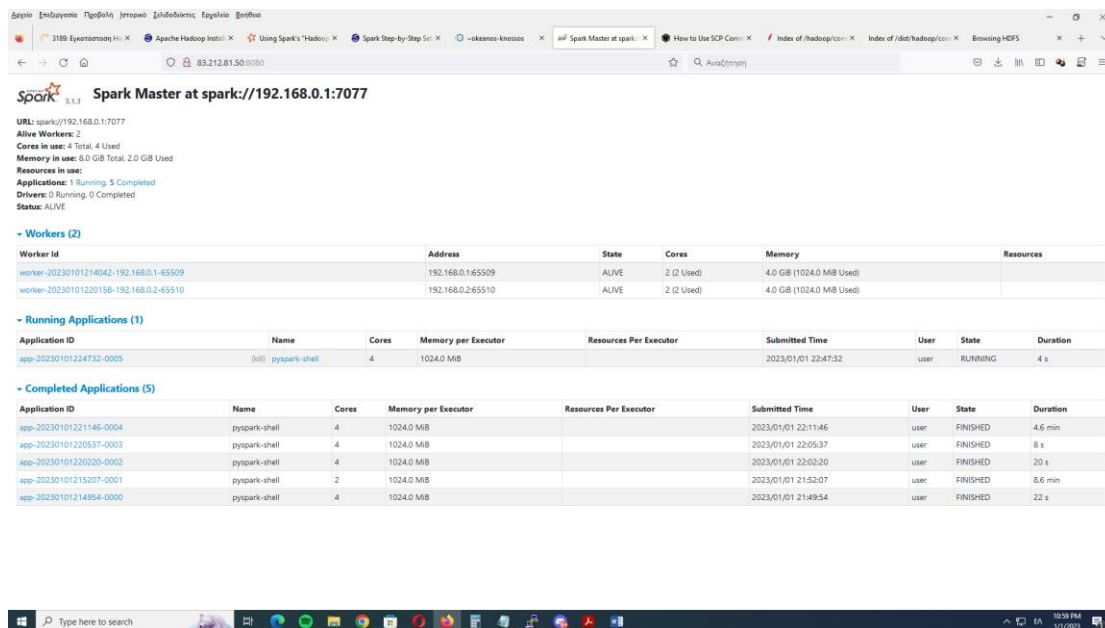


Στη συνέχεια εγκαθιστούμε το Hadoop (HDFS). Ορίζουμε ως namenode το master και ως datanodes το master και το slave. Ανεβάζουμε τα αρχεία με τα δεδομένα μέσω της εντολής: `hdfs dfs -put` στα αντίστοιχα folders και χτυπάμε την public διεύθυνση του Master (83.212.81.50) στη θύρα 9870 για να δούμε ότι το hdfs έχει στηθεί σωστά και υπάρχουν τα αντίστοιχα αρχεία





Αντίστοιχα εγκαθιστούμε το apache spark. Ορίζουμε ως master node το master και ως workers τον master και τον slave. Αργότερα απενεργοποιούμε τον ένα worker για να δούμε τη διαφορά του χρόνου στα queries. Χτυπάμε την public διεύθυνση του Master (83.212.81.50) στη θύρα 8080 και βλέπουμε ότι το apache spark έχει στηθεί σωστά και υπάρχουν 2 ενεργοί workers.



Φτιάχνουμε ένα ργθιοη εκτελέσιμο για όλα τα queries. Αρχικά διαβάζουμε τα δεδομένα μας μέσω του hdfs που δημιουργήσαμε.

```
parDF1=spark.read.parquet('hdfs://192.168.0.1:9000/user/user/data')
```

Φιλτράρουμε τυχόν δεδομένα που βρίσκονται εκτός του διαστήματος ενδιαφέροντος δηλαδή εκτός του διαστήματος 01/2022 – 06/2022.

```
parDF1 = parDF1.filter((year(parDF1['tpep_pickup_datetime']) == "2022") & (month(parDF1['tpep_pickup_datetime']) < "7"))
```

Τέλος δημιουργούμε το rdd από το dataframe μας.

```
rdd1 = parDF1.rdd
```

Αντίστοιχα για το csv.

```
schema2 = "LocationID INT, Borough STRING, Zone STRING, service_zone STRING"
#create dataframe for csv
DF2 = spark.read.csv('hdfs://192.168.0.1:9000/user/user/datacsv', schema = schema2)
```

Ο κώδικας για κάθε query φαίνεται αναλυτικά παρακάτω:

Q1:

```
parDF1.join(DF2new, parDF1.DOLocationID == DF2new.DOLocationID,"inner").filter((month(parDF1['tpep_pickup_datetime']) == "3") & (DF2new['Zone'] == "Battery Park")).agg({"tip_amount":"max"}).show()
```

Q1sql:

```
sqlDF1 = spark.sql("select max(tip_amount) from yellowtrip INNER JOIN location ON yellowtrip.DOLocationID == location.DOLocationID where month(tpep_pickup_datetime) = 3 and Zone = 'Battery Park'")
sqlDF1.show()
```

Q2:

```
parDF1.groupby(month("tpep_pickup_datetime").alias("month")).max("tolls_amount").show()
```

Q3:

```
parDF1.filter(parDF1['PULocationID']!=parDF1['DOLocationID']).groupBy((floor((dayofyear("tpep_pickup_datetime")-1)/15)+1).alias("nth 15 days of year")).mean('trip_distance','total_amount').show()
```

Q3rdd:

```
def convert(row):
    day_str = row.tpep_pickup_datetime
    dist = row.trip_distance
    amount = row.total_amount
    #dt = datetime.strptime(day_str, '%Y-%m-%d %H:%M:%S')
    dayofyear = int(day_str.strftime("%j"))
    fifteen = ((dayofyear-1)//15)+1
    return (fifteen, (dist,amount,1))

rddx = rdd1.filter(lambda x: x.PULocationID != x.DOLocationID).map(convert)

#for x in rddx.collect():
#    print(x)

rddfinal = rddx.reduceByKey(lambda a,b: (a[0]+b[0],a[1]+b[1],a[2]+b[2])).mapValues(lambda x: (x[0]/x[2],x[1]/x[2]))

for x in rddfinal.collect():
    print(x)
```

Q4:

```
windowdays = Window.partitionBy(dayofweek("tpep_pickup_datetime")).orderBy(col("passenger_count").desc())
df1days = parDF1.withColumn("row_num",row_number().over(windowdays))
df1days.filter(col("row_num") <= 3).select(hour("tpep_pickup_datetime").alias("rush hours"),dayofweek("tpep_pickup_datetime").alias("day")).show(50)
```

Q5:

```
windowmonths = Window.partitionBy(month("tpep_pickup_datetime")).orderBy(col("percent").desc())
df1months2 = df1months.withColumn("row_num",row_number().over(windowmonths))
df1months2.filter(col("row_num") <= 5).select(dayofmonth("tpep_pickup_datetime").alias("day"),month("tpep_pickup_datetime").alias("month")).show(50)
```

Εκτελούμε το αρχείο με τα queries μέσω της εντολής: `python3.8 queriesfinal.py` και βλέπουμε τα παρακάτω αποτελέσματα με τους αντίστοιχους χρόνους για όλα τα queries.

Για 2 Workers:

Q1

```
+-----+
|max(tip_amount)|
+-----+
|          40.0|
+-----+

TimeQ1: 15.89511251449585
+-----+
|max(tip_amount)|
+-----+
|          40.0|
+-----+

TimeQ1sql: 7.532622575759888
```

Q2

```
+-----+-----+
|month|max(tolls_amount)|
+-----+-----+
|    1|             193.3|
|    6|            800.09|
|    3|             235.7|
|    5|            813.75|
|    4|            911.87|
|    2|              95.0|
+-----+-----+

TimeQ2: 10.774070024490356
```

Q3

```
+-----+-----+-----+
|nth 15 days of year|avg(trip_distance)| avg(total_amount)|
+-----+-----+-----+
|              7| 5.679323077938295|21.515559094583587|
|              6| 5.532999252101388|21.121659489582353|
|              9| 6.249697852127242|21.921570348909114|
|              5| 6.60698631990843|20.692357713183547|
|              1| 5.576410377852007|19.903702637879007|
|             10| 7.9990632224691165|22.806499070460386|
|              3| 5.950485844928121|19.553891327960553|
|             12| 6.153370128239474|22.352167683521646|
|              8| 5.800344707645977|21.428088376232783|
|             11| 6.378971191608972|22.452110839872283|
|              2| 4.804840472309411| 19.03660791389491|
|              4| 6.1857672125677| 20.17207809365826|
|             13| 5.811220970695942| 22.16938397436561|
+-----+-----+-----+

TimeQ3: 10.495319128036499
(6, (5.532999252101388, 21.121659489582353))
(12, (6.153370128239474, 22.352167683521646))
(7, (5.679323077938295, 21.515559094583587))
(13, (5.811220970695942, 22.16938397436561))
(1, (5.576410377852007, 19.903702637879007))
(8, (5.800344707645977, 21.428088376232783))
(2, (4.804840472309411, 19.03660791389491))
(9, (6.249697852127242, 21.921570348909114))
(3, (5.950485844928121, 19.553891327960553))
(4, (6.1857672125677, 20.17207809365826))
(10, (7.9990632224691165, 22.806499070460386))
(5, (6.60698631990843, 20.692357713183547))
(11, (6.378971191608972, 22.452110839872283))
TimeQ3rdd: 194.3226819038391
```

Q4

```
+-----+-----+
|rush hours|day|
+-----+-----+
|          |16| 1|
|          |22| 1|
|          |11| 1|
|          |18| 6|
|          |19| 6|
|          |19| 6|
|          |22| 3|
|          |20| 3|
|          |21| 3|
|          |21| 5|
|          | 2| 5|
|          | 1| 5|
|          | 9| 4|
|          |14| 4|
|          | 4| 4|
|          | 6| 7|
|          |13| 7|
|          |14| 7|
|          |20| 2|
|          | 1| 2|
|          |20| 2|
+-----+-----+

TimeQ4: 26.119625329971313
```

Q5

```
+-----+-----+
|day|month|
+-----+-----+
| 9| 1|
|31| 1|
|31| 1|
| 1| 1|
| 3| 1|
|25| 6|
|13| 6|
|10| 6|
|16| 6|
|13| 6|
|18| 3|
|21| 3|
|26| 3|
| 5| 3|
|18| 3|
|12| 5|
|12| 5|
|20| 5|
|15| 5|
|16| 5|
|12| 4|
| 2| 4|
|21| 4|
|12| 4|
| 3| 4|
|21| 2|
|13| 2|
| 9| 2|
|27| 2|
|24| 2|
+-----+-----+

TimeQ5: 26.428637981414795
```

Στη συνέχεια εκτελούμε το script: stopworker.sh που δημιουργήσαμε στο slave node για να απενεργοποιήσουμε τον ένα worker στο spark.

```
user@snf-34021:~$ ./stopworker.sh
stopping org.apache.spark.deploy.worker.Worker
user@snf-34021:~$
```

The screenshot shows the Spark Master web interface at `spark://192.168.0.1:7077`. The interface displays the following information:

- URL:** `spark://192.168.0.1:7077`
- Alive Workers:** 1
- Cores in use:** 2 Total, 2 Used
- Memory in use:** 4.0 GB Total, 1024.0 MB Used
- Resources in use:**
- Applications:** 1 Running, 6 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20230101214042-192.168.0.1-65509	192.168.0.1:65509	ALIVE	2 (2 Used)	4.0 GB (1024.0 MB Used)	
worker-20230101220158-192.168.0.2-65510	192.168.0.2:65510	DEAD	2 (0 Used)	4.0 GB (0.0 B Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20230101232119-0006	[yml] pyspark-shell	2	1024.0 MB		2023/01/01 23:21:19	user	RUNNING	6 s

Completed Applications (6)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20230101224732-0005	pyspark-shell	4	1024.0 MB		2023/01/01 22:47:32	user	FINISHED	4.5 min
app-20230101221146-0004	pyspark-shell	4	1024.0 MB		2023/01/01 22:11:46	user	FINISHED	4.6 min
app-20230101220937-0003	pyspark-shell	4	1024.0 MB		2023/01/01 22:09:37	user	FINISHED	8 s
app-20230101220220-0002	pyspark-shell	4	1024.0 MB		2023/01/01 22:02:20	user	FINISHED	20 s
app-20230101215207-0001	pyspark-shell	2	1024.0 MB		2023/01/01 21:52:07	user	FINISHED	8.6 min
app-20230101214954-0000	pyspark-shell	4	1024.0 MB		2023/01/01 21:49:54	user	FINISHED	22 s

Ξανατρέχουμε το εκτελέσιμο αρχείο `queriesfinal.py` με 1 worker αυτή τη φορά και βλέπουμε πάλι τα αποτελέσματα και τους αντίστοιχους χρόνους για κάθε query

Για 1 Worker:

Q1

```
+-----+
|max(tip_amount)|
+-----+
|          40.0|
+-----+

TimeQ1: 23.549365997314453

+-----+
|max(tip_amount)|
+-----+
|          40.0|
+-----+

TimeQ1sq1: 17.263962030410767
```


Q2

```
+-----+
|month|max(tolls_amount)|
+-----+
| 1|193.3|
| 6|800.09|
| 3|235.7|
| 5|813.75|
| 4|911.87|
| 2|95.0|
+-----+

TimeQ2: 20.07305335998535
```

Q3

```
+-----+
|nth 15 days of year|avg(trip_distance)| avg(total_amount)|
+-----+
| 7|5.679323077938295|21.515559094583587|
| 6|5.532999252101388|21.121659489582353|
| 9|6.249697852127242| 21.9215703489091|
| 5|6.60698631990843|20.652357713183547|
| 1|5.576410377852007|19.903702637879007|
|10|7.9990632224691165|22.806498070460386|
| 3|5.950485844928086|19.553891327979455|
|12|6.153370128239474|22.352167683521646|
| 8|5.800344707645977|21.428088376232783|
|11|6.378971191608972|22.452110839872283|
| 2|4.804840472309411| 19.03660791389491|
| 4| 6.1857672125677| 20.17207809365826|
|13|5.811220970695942| 22.16938397436561|
+-----+

TimeQ3: 27.347949981689453
(6, (5.532999252101388, 21.121659489582353))
(9, (6.249697852127242, 21.9215703489091))
(12, (6.153370128239474, 22.352167683521646))
(3, (5.950485844928086, 19.553891327979455))
(4, (6.1857672125677, 20.17207809365826))
(7, (5.679323077938295, 21.515559094583587))
(10, (7.9990632224691165, 22.806498070460386))
(13, (5.811220970695942, 22.16938397436561))
(1, (5.576410377852007, 19.903702637879007))
(5, (6.60698631990843, 20.652357713183547))
(8, (5.800344707645977, 21.428088376232783))
(11, (6.378971191608972, 22.452110839872283))
(2, (4.804840472309411, 19.03660791389491))
TimeQ3rdd: 440.64205503463745
```

Q4

```
+-----+
|rush hours|day|
+-----+
| 16| 1|
| 23| 1|
| 21| 1|
| 18| 6|
| 19| 6|
| 19| 6|
| 21| 3|
| 19| 3|
| 11| 3|
| 21| 5|
| 2| 5|
| 20| 5|
| 9| 4|
| 18| 4|
| 14| 4|
| 13| 7|
| 14| 7|
| 6| 7|
| 20| 2|
| 17| 2|
| 17| 2|
+-----+

TimeQ4: 74.01037645339966
```

Q5

```
+---+-----+
|day|month|
+---+-----+
| 9| 1|
|31| 1|
|31| 1|
| 1| 1|
| 3| 1|
|25| 6|
|13| 6|
|10| 6|
|16| 6|
|13| 6|
|18| 3|
|21| 3|
|26| 3|
| 5| 3|
|18| 3|
|12| 5|
|12| 5|
|20| 5|
|15| 5|
|16| 5|
|12| 4|
| 2| 4|
|21| 4|
|12| 4|
| 3| 4|
|21| 2|
|13| 2|
| 9| 2|
|27| 2|
|24| 2|
+---+-----+
TimeQ5: 88.72045373916626
```

Συνοψίζουμε τα αποτελέσματα με τους χρόνους για κάθε query για έναν και δύο workers αντίστοιχα

Time (sec)	One worker	Two workers
Q1	23.54	15.89
Q1sql	17.26	7.53
Q2	20.07	10.77
Q3	27.34	10.49
Q3rdd	440.64	194.32
Q4	74.01	26.11
Q5	88.72	26.42

Βλέπουμε ότι οι χρόνοι μειώνονται σημαντικά με 2 workers γεγονός που αποδεικνύει ότι το setup έχει γίνει σωστά και αποδεικνύει την βελτίωση της επίδοσης μέσω της αύξησης των workers στο apache spark. Ειδικά στο query με rdd η βελτίωση είναι σημαντική.