ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Εξαμηνιαία Εργασία

Ακαδημαϊκό Έτος 2022-2023

Ομάδα 58:

Ντούσης Οδυσσέας 03119441

Συριωγωνάκης Αλέξανδρος 03117044

1.

Εγκατάσταση HDFS:

Εγκαταστάθηκε η έκδοση hadoop-3.3.4 σε cluster 2 κόμβων (master at 192.168.0.1, worker at 192.168.0.2). Για το setup του hadoop και του yarn ακουλουθήθηκαν τα βήματα που περιγράφονται στους οδηγούς:

https://sparkbyexamples.com/hadoop/apache-hadoop-installation/
https://sparkbyexamples.com/hadoop/yarn-setup-and-run-map-reduce-program/

Ως Namenode ορίστηκε ο master, ενημερώθηκαν τα xml files με τις κατάλληλες τοπικές παραμέτρους και δημιουργήθηκαν τα master και workers files, τα οποία περιέχουν τις ip διευθύνσεις του master και των worker αντίστοιχα

Η έναρξη λειτουργίας του HDFS γίνεται με την εντολή start-dfs.sh

Το cluster είναι λειτουργικό και μπορεί να προσπελαστεί από το web interface {master_public_ip}:9870

Με την εντολή hdfs hdfs dfs -mkdir data δημιουργείται το directory στο οποίο θα φορτωθούν τα parquet files και το csv file. Αυτό γίνεται στη συνέχεια με την εντολή hdfs dfs -put data/{filename}.

Στη συνέχεια γίνεται configuration του yarn, και ενημερώνονται κατάλληλα τα xml files, όπως περιγράφει ο παραπάνω οδηγός. Το yarn ενεργοποιείται με την εντολή start-yarn.sh

Για την εγκατάσταση του spark αξιοποιήθηκε ο οδηγός:

https://sparkbyexamples.com/spark/spark-setup-on-hadoop-yarn/

καθώς και ο οδηγός Spark_installation που μας δόθηκε

Εγκαταστάθηκε η έκδοση spark-3.3.1-bin-hadoop3

Συγκεκριμένα ακολουθήθηκαν τα βήματα 1 εως 8 του οδηγού και επίσης τροποποιήθηκαν τα αρχεία

→ $SPARK_HOME/conf/spark-defaults.conf:

spark.master yarn
spark.driver.memory 1024m
spark.yarn.am.memory 1024m
spark.executor.memory 4g

→ $SPARK_HOME/conf/spark-env.sh:

SPARK_MASTER_HOST='192.168.0.1'

→ $SPARK_HOME/conf/workers
192.168.0.1

192.168.0.2

Εκκινούμε τον master με την εντολή start-master.sh και τους workers με την εντολή start-worker.sh spark://192.168.0.1:7077

Μπορούμε να έχουμε επισκόπηση των ενεργών worker στο {master_public_ip}:8080

Στη συνέχεια για την εκτέλεση των python scripts αρκεί να εκτελεστεί η:

python3.8 {file_name}.py

Για κάθε query φορτώνουμε τα parquet αρχεια και το csv σε δύο dataframe:

```
df = spark.read.parquet("hdfs://192.168.0.1:9000/data/yellow_tripdata_2022-0*.parquet")
lt = spark.read.option("header",True).csv("hdfs://192.168.0.1:9000/data/taxi+_zone_lookup.csv")
```

και μπορούμε να τα μετατρέψουμε σε rdd απλά: rdd1 = df.rdd

Dataframe API

Τα ερωτήματα Q1-Q5 υλοποιήθηκαν μέσω της διεπαφής DataFrame. Στον παρακάτω παρουσιάζονται οι χρόνοι για 1 και 2 workers (Για την περίπτωση του ενός worker κρατάμε ανοιχτό μόνο τον worker που βρίσκεται στον master)

	1 worker(master)	1 worker(worker)	2 workers
Q1	44.57967948913574	42.66677141189575	37.508095502853394
Q2	94.00861716270447	85.0381863117218	62.7083158493042
Q3	35.5038902759552	52.79745531082153	34.581276416778564
Q4	26.636648416519165	30.47820782661438	25.893094539642334
Q5	24.81599473953247	23.463637113571167	25.81048822402954

RDD API

Το ερώτημα Q3 υλοποιήθηκε και σε RDD. Οι χρόνοι εκτέλεσης για έναν και δυο workers:

	1 worker	2 workers
Q3	923.3246314525604	588.0243837833405

Σημείωση:

Στο ερώτημα Q3 η στήλη half αναφέρεται αν οι υπολογισμοί γίνονται στο πρώτο (0) ή στο δεύτερο (1) δεκαπενθήμερο του μήνα

Στο ερώτημα Q4 η στήλη hour δηλώνει το διάστημα στο οποί η ώρα έχει αυτή την τιμή (δηλαδή για hour = 23 εννοείται το διάστημα 23:00 εως 23:59)

Queries Dump (from show()):

Q1:

```
+--------+--------------------+--------------------+---------------+------------+----------+------------------
+------------+------------+------------+-----------+-----+-------+----------+------------+--------------------
+------------+-------------------+-----------+----------+---------+------------+------------+
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distance|RatecodeID|
store_and_fwd_flag|PULocationID|DOLocationID|payment_type|fare_amount|extra|mta_tax|
tip_amount|tolls_amount|improvement_surcharge|total_amount|congestion_surcharge|airport_fee|
LocationID|  Borough|       Zone|service_zone|
+--------+--------------------+--------------------+---------------+------------+----------+------------------
+------------+------------+------------+-----------+-----+-------+----------+------------+--------------------
+------------+-------------------+-----------+----------+---------+------------+------------+
|      2| 2022-03-17 12:27:47|  2022-03-17 12:27:58|            1.0|         0.0|       1.0|                N|         12|
12|          1|        2.5|  0.0|    0.5|      40.0|         0.0|                 0.3|       45.8|                2.5|        0.0|
12|Manhattan|Battery Park| Yellow Zone|
+--------+--------------------+--------------------+---------------+------------+----------+------------------
+------------+------------+------------+-----------+-----+-------+----------+------------+--------------------
+------------+-------------------+-----------+----------+---------+------------+------------+
```

Q2:

```
+--------+--------------------+--------------------+---------------+------------+----------+------------------
+------------+------------+------------+-----------+-----+-------+----------+------------+--------------------
+------------+-------------------+-----------+-----+
|VendorID|tpep_pickup_datetime|tpep_dropoff_datetime|passenger_count|trip_distance|RatecodeID|
store_and_fwd_flag|PULocationID|DOLocationID|payment_type|fare_amount|extra|mta_tax|
tip_amount|tolls_amount|improvement_surcharge|total_amount|congestion_surcharge|airport_fee|
month|
+--------+--------------------+--------------------+---------------+------------+----------+------------------
+------------+------------+------------+-----------+-----+-------+----------+------------+--------------------
+------------+-------------------+-----------+-----+
|      1| 2022-01-22 11:39:07|  2022-01-22 12:31:09|            1.0|        33.4|       1.0|                Y|         70|
265|          4|       88.0|  0.0|    0.5|       0.0|       193.3|                 0.3|      282.1|                0.0|        0.0|
1|
|      1| 2022-02-18 02:33:30|  2022-02-18 02:35:28|            1.0|         1.3|       1.0|                N|        265|
265|          1|        3.0|  0.5|    0.5|     19.85|        95.0|                 0.3|     119.15|                0.0|        0.0|
2|
|      1| 2022-03-11 20:08:32|  2022-03-11 20:09:45|            1.0|         0.0|       1.0|                N|        265|
265|          1|        2.5|  1.0|    0.5|      48.0|       235.7|                 0.3|      288.0|                0.0|        0.0|
3|
|      1| 2022-04-29 04:31:21|  2022-04-29 04:32:30|            2.0|         0.0|       1.0|                N|        249|
249|          3|        3.0|  3.0|    0.5|       0.0|      911.87|                 0.3|     918.67|                2.5|        0.0|
4|
|      1| 2022-05-21 16:47:48|  2022-05-21 17:05:47|            1.0|         2.4|       3.0|                N|        239|
```

```
246|          3|       31.5|  0.0|    0.0|       0.0|      813.75|                0.3|      845.55|                0.0|       0.0|
5|
|       1| 2022-06-12 16:51:46|  2022-06-12 17:56:48|           9.0|        22.0|       1.0|                N|
142|         132|           2|       67.5|  2.5|    0.5|       0.0|      800.09|                0.3|      870.89|                2.5|
0.0|    6|
+--------+--------------------+---------------------+--------------+------------+----------+-----------------
+------------+------------+------------+-----------+-----+-------+----------+------------+--------------------
+------------+-------------------+-----------+-----+
```

Q3:

```
+-----+-----+-----------------+-----------------+
|month| half|avg(Trip_distance)| avg(Total_amount)|
+-----+-----+-----------------+-----------------+
|    1|false| 5.576429554927403| 19.90405084638674|
|    1| true| 5.097880367275346| 19.14882164234129|
|    2|false| 6.248888338463885|19.491979067237448|
|    2| true| 5.849460516243601| 20.18769180439039|
|    3|false| 6.480491651211442|20.652292316598395|
|    3| true| 5.556947478917816|21.120927430034417|
|    4|false| 5.679323077938295|21.515559094583587|
|    4| true| 5.800341831534024|21.428117467145515|
|    5|false| 6.249697852127242|21.921570348909114|
|    5| true| 7.906694182348757|22.771948777963715|
|    6|false| 6.315157336730177|22.466305309343248|
|    6| true| 6.174138574511356|22.331380641103525|
+-----+-----+-----------------+-----------------+
```

Q4:

```
+---+----+-----------------+
|day|hour|       congestion|
+---+----+-----------------+
|  1|   0|1.5299385566844705|
|  1|   1| 1.527827635720992|
|  1|   2|1.5080607476635515|
|  2|   0|1.4679941240391194|
|  2|   1|1.4442867916810471|
|  2|   2|1.4231993989051486|
|  3|   0|1.4200313882151518|
|  3|   1|1.4175124740006593|
|  3|   2|1.4104520814693964|
|  4|   1| 1.408844324695756|
|  4|   0|  1.40126318475672|
|  4|   2|1.4011194528306927|
|  5|  23| 1.405379910176621|
|  5|   1|1.4026163620497012|
```

5	0	1.401039516028603
6	23	1.4755868330850932
6	22	1.4448114645797336
6	2	1.4236408968027283
7	23	1.52260296010296
7	22	1.506814483948998
7	0	1.4993284322949447
+---+----+-----------------+

Q5:

+-----+---+--------------------+
|month|day|average tip percentage|
+-----+---+--------------------+
1	9	0.4578674775487535
1	31	0.4393563580769872
1	1	0.2906301939811919
1	29	0.2405951845436878
1	16	0.23377299918217617
2	21	0.25981657452765006
2	13	0.2457206838940141
2	9	0.23904535643411468
2	10	0.23339615899346813
2	27	0.23300679951545766
3	18	0.29671341612657676
3	21	0.2757992602492
3	26	0.22708845953721593
3	5	0.22555461372495167
3	12	0.22100859110807622
4	12	0.4836884410450817
4	2	0.31175092883996536
4	21	0.3044861250236238
4	3	0.24463727704754118
4	30	0.21996769659947238
5	12	0.32402658973195914
5	20	0.2603403609036704
5	16	0.23659110789277535
5	15	0.220524452470084
5	6	0.2183200616188207
6	13	0.38451369937243063
6	25	0.32913073292653017
6	10	0.27397637812780157
6	16	0.2553497575787421
6	20	0.24242914593518236
+-----+---+--------------------+

Queries Dump (Collect()):

Q1: Row(VendorID=2, tpep_pickup_datetime=datetime.datetime(2022, 3, 17, 12, 27, 47), tpep_dropoff_datetime=datetime.datetime(2022, 3, 17, 12, 27, 58), passenger_count=1.0, trip_distance=0.0, RatecodeID=1.0, store_and_fwd_flag='N', PULocationID=12, DOLocationID=12, payment_type=1, fare_amount=2.5, extra=0.0, mta_tax=0.5, tip_amount=40.0, tolls_amount=0.0, improvement_surcharge=0.3, total_amount=45.8, congestion_surcharge=2.5, airport_fee=0.0, LocationID='12', Borough='Manhattan', Zone='Battery Park', service_zone='Yellow Zone')

Q2: Row(VendorID=1, tpep_pickup_datetime=datetime.datetime(2022, 1, 22, 11, 39, 7), tpep_dropoff_datetime=datetime.datetime(2022, 1, 22, 12, 31, 9), passenger_count=1.0, trip_distance=33.4, RatecodeID=1.0, store_and_fwd_flag='Y', PULocationID=70, DOLocationID=265, payment_type=4, fare_amount=88.0, extra=0.0, mta_tax=0.5, tip_amount=0.0, tolls_amount=193.3, improvement_surcharge=0.3, total_amount=282.1, congestion_surcharge=0.0, airport_fee=0.0, month=1)
Row(VendorID=1, tpep_pickup_datetime=datetime.datetime(2022, 2, 18, 2, 33, 30), tpep_dropoff_datetime=datetime.datetime(2022, 2, 18, 2, 35, 28), passenger_count=1.0, trip_distance=1.3, RatecodeID=1.0, store_and_fwd_flag='N', PULocationID=265, DOLocationID=265, payment_type=1, fare_amount=3.0, extra=0.5, mta_tax=0.5, tip_amount=19.85, tolls_amount=95.0, improvement_surcharge=0.3, total_amount=119.15, congestion_surcharge=0.0, airport_fee=0.0, month=2)
Row(VendorID=1, tpep_pickup_datetime=datetime.datetime(2022, 3, 11, 20, 8, 32), tpep_dropoff_datetime=datetime.datetime(2022, 3, 11, 20, 9, 45), passenger_count=1.0, trip_distance=0.0, RatecodeID=1.0, store_and_fwd_flag='N', PULocationID=265, DOLocationID=265, payment_type=1, fare_amount=2.5, extra=1.0, mta_tax=0.5, tip_amount=48.0, tolls_amount=235.7, improvement_surcharge=0.3, total_amount=288.0, congestion_surcharge=0.0, airport_fee=0.0, month=3)
Row(VendorID=1, tpep_pickup_datetime=datetime.datetime(2022, 4, 29, 4, 31, 21), tpep_dropoff_datetime=datetime.datetime(2022, 4, 29, 4, 32, 30), passenger_count=2.0, trip_distance=0.0, RatecodeID=1.0, store_and_fwd_flag='N', PULocationID=249, DOLocationID=249, payment_type=3, fare_amount=3.0, extra=3.0, mta_tax=0.5, tip_amount=0.0, tolls_amount=911.87, improvement_surcharge=0.3, total_amount=918.67, congestion_surcharge=2.5, airport_fee=0.0, month=4)
Row(VendorID=1, tpep_pickup_datetime=datetime.datetime(2022, 5, 21, 16, 47, 48), tpep_dropoff_datetime=datetime.datetime(2022, 5, 21, 17, 5, 47), passenger_count=1.0, trip_distance=2.4, RatecodeID=3.0, store_and_fwd_flag='N', PULocationID=239, DOLocationID=246, payment_type=3, fare_amount=31.5, extra=0.0, mta_tax=0.0, tip_amount=0.0, tolls_amount=813.75, improvement_surcharge=0.3, total_amount=845.55, congestion_surcharge=0.0, airport_fee=0.0, month=5)
Row(VendorID=1, tpep_pickup_datetime=datetime.datetime(2022, 6, 12, 16, 51, 46), tpep_dropoff_datetime=datetime.datetime(2022, 6, 12, 17, 56, 48), passenger_count=9.0, trip_distance=22.0, RatecodeID=1.0, store_and_fwd_flag='N', PULocationID=142, DOLocationID=132, payment_type=2, fare_amount=67.5, extra=2.5, mta_tax=0.5, tip_amount=0.0, tolls_amount=800.09, improvement_surcharge=0.3, total_amount=870.89, congestion_surcharge=2.5, airport_fee=0.0, month=6)

Q3 Dataframe: Row(month=1, half=False, avg(Trip_distance)=5.576429554927403, avg(Total_amount)=19.90405084638674)
Row(month=1, half=True, avg(Trip_distance)=5.097880367275346, avg(Total_amount)=19.14882164234129)
Row(month=2, half=False, avg(Trip_distance)=6.248888338463885, avg(Total_amount)=19.491979067237448)
Row(month=2, half=True, avg(Trip_distance)=5.849460516243601, avg(Total_amount)=20.18769180439039)
Row(month=3, half=False, avg(Trip_distance)=6.480491651211442, avg(Total_amount)=20.652292316598395)
Row(month=3, half=True, avg(Trip_distance)=5.556947478917816, avg(Total_amount)=21.120927430034417)
Row(month=4, half=False, avg(Trip_distance)=5.679323077938295, avg(Total_amount)=21.515559094583587)
Row(month=4, half=True, avg(Trip_distance)=5.800341831534024, avg(Total_amount)=21.42811746714552)
Row(month=5, half=False, avg(Trip_distance)=6.249697852127242, avg(Total_amount)=21.921570348909114)
Row(month=5, half=True, avg(Trip_distance)=7.906694182348757, avg(Total_amount)=22.771948777963715)
Row(month=6, half=False, avg(Trip_distance)=6.315157336730177, avg(Total_amount)=22.466305309343248)
Row(month=6, half=True, avg(Trip_distance)=6.174138574511356, avg(Total_amount)=22.331380641103525)

Q3 RDD: [((1, 0), (5.576429554927422, 19.90405084621702)), ((1, 1), (5.097880367275371, 19.148821642159373))]
[((2, 0), (6.248888338463784, 19.491979067027476)), ((2, 1), (5.849460516243568, 20.187691804185548))]
[((3, 0), (6.480491651211576, 20.652292316351936)), ((3, 1), (5.556947478917783, 21.120927429871042))]
[((4, 0), (5.6793230779383785, 21.515559094396902)), ((4, 1), (5.800341831534326, 21.428117466951257))]
[((5, 0), (6.2496978521272695, 21.921570348700058)), ((5, 1), (7.906694182348883, 22.771948777835313))]
[((6, 0), (6.3151573367302145, 22.46630530920166)), ((6, 1), (6.1741385745113275, 22.331380640904253))]

“””
where:
x[0][0] = month
x[0][1] = 0 for first half of month, 1 for second
x[1][0] = average Trip_distatnce
x[1][1] = average Total_Amount
“””

Q4: Row(day=1, hour=0, cong=1.5299385566844705)
Row(day=1, hour=1, cong=1.527827635720992)
Row(day=1, hour=2, cong=1.5080607476635515)
Row(day=2, hour=0, cong=1.4679941240391194)
Row(day=2, hour=1, cong=1.4442867916810471)
Row(day=2, hour=2, cong=1.4231993989051486)
Row(day=3, hour=0, cong=1.4200313882151518)
Row(day=3, hour=1, cong=1.4175124740006593)
Row(day=3, hour=2, cong=1.4104520814693964)
Row(day=4, hour=1, cong=1.408844324695756)
Row(day=4, hour=0, cong=1.40126318475672)
Row(day=4, hour=2, cong=1.4011194528306927)
Row(day=5, hour=23, cong=1.405379910176621)
Row(day=5, hour=1, cong=1.4026163620497012)
Row(day=5, hour=0, cong=1.401039516028603)
Row(day=6, hour=23, cong=1.4755868330850932)
Row(day=6, hour=22, cong=1.4448114645797336)
Row(day=6, hour=2, cong=1.4236408968027283)
Row(day=7, hour=23, cong=1.52260296010296)
Row(day=7, hour=22, cong=1.506814483948998)
Row(day=7, hour=0, cong=1.4993284322949447)

Q5: Row(month=1, day=9, average tip percentage=0.4578674775487535)
Row(month=1, day=31, average tip percentage=0.4393563580769872)
Row(month=1, day=1, average tip percentage=0.2906301939811919)
Row(month=1, day=29, average tip percentage=0.2405951845436878)
Row(month=1, day=16, average tip percentage=0.23377299918217617)
Row(month=2, day=21, average tip percentage=0.25981657452765006)
Row(month=2, day=13, average tip percentage=0.2457206838940141)
Row(month=2, day=9, average tip percentage=0.23904535643411468)
Row(month=2, day=10, average tip percentage=0.23339615899346813)
Row(month=2, day=27, average tip percentage=0.23300679951545766)
Row(month=3, day=18, average tip percentage=0.29671341612657676)
Row(month=3, day=21, average tip percentage=0.2757992602492)
Row(month=3, day=26, average tip percentage=0.22708845953721593)
Row(month=3, day=5, average tip percentage=0.22555461372495167)
Row(month=3, day=12, average tip percentage=0.22100859110807622)
Row(month=4, day=12, average tip percentage=0.4836884410450817)
Row(month=4, day=2, average tip percentage=0.31175092883996536)
Row(month=4, day=21, average tip percentage=0.3044861250236238)
Row(month=4, day=3, average tip percentage=0.24463727704754118)
Row(month=4, day=30, average tip percentage=0.21996769659947238)
Row(month=5, day=12, average tip percentage=0.32402658973195914)
Row(month=5, day=20, average tip percentage=0.2603403609036704)

Row(month=5, day=16, average tip percentage=0.23659110789277535)
Row(month=5, day=15, average tip percentage=0.220524452470084)
Row(month=5, day=6, average tip percentage=0.2183200616188207)
Row(month=6, day=13, average tip percentage=0.38451369937243063)
Row(month=6, day=25, average tip percentage=0.32913073292653017)
Row(month=6, day=10, average tip percentage=0.27397637812780157)
Row(month=6, day=16, average tip percentage=0.2553497575787421)
Row(month=6, day=20, average tip percentage=0.24242914593518236)

Query Code:

Q1:

```
df = spark.read.parquet("hdfs://192.168.0.1:9000/data/yellow_tripdata_2022-0*.parquet")
lt = spark.read.option("header",True).csv("hdfs://192.168.0.1:9000/data/taxi+_zone_lookup.csv")

w = df.where(df.tpep_pickup_datetime.contains("2022-03"))\
.join(lt,df.DOLocationID == lt.LocationID,"inner")\
.where(col("Zone")=="Battery Park")\
.orderBy(desc(col("Tip_amount")))

w1 = w.collect()

w.show(1)
```

Q2:

```
df = spark.read.parquet("/data/yellow_tripdata_2022-0*.parquet")

windowM =Window.partitionBy("month").orderBy(desc(col("Tolls_amount")))

w = df.withColumn("month", month("tpep_pickup_datetime"))\
.withColumn("row",row_number().over(windowM))\
.filter(col("row")==1).drop("row")\
.orderBy(asc(col("month")))

w1 = w.collect()

w.show(6)
```

Q3 DataFrame:

```
df = spark.read.parquet("/data/yellow_tripdata_2022-0*.parquet")
lt = spark.read.option("header",True).csv("hdfs://192.168.0.1:9000/data/taxi+_zone_lookup.csv")

w =
df.select("tpep_pickup_datetime","Trip_distance","Total_amount","PULocationID","DOLocationID")\
.join(lt.withColumnRenamed("Borough",
"SBorough").withColumnRenamed("Zone","SZone"),df.PULocationID ==
lt.LocationID,"inner").drop("LocationID","service_zone")\
.join(lt.withColumnRenamed("Borough",
"DBorough").withColumnRenamed("Zone","DZone"),df.DOLocationID ==
```

```
lt.LocationID,"inner").drop("LocationID","service_zone")\
.where((col("PULocationID")!=col("DOLocationID")) & (col("SZone")!=col("DZone")))\
.withColumn("month", month("tpep_pickup_datetime"))\
.withColumn("half", dayofmonth("tpep_pickup_datetime")>15)\
.groupBy("month","half")\
.agg(avg("Trip_distance"),avg("Total_amount"))\
.orderBy("month","half")

w1 = w.collect()

w.show(12)
```

Q3 RDD:

```
df = spark.read.parquet("/data/yellow_tripdata_2022-0*.parquet")
lt = spark.read.option("header",True).csv("hdfs://192.168.0.1:9000/data/taxi+_zone_lookup.csv")

temp1 = lt.withColumnRenamed("Borough",
"SBorough").withColumnRenamed("Zone","SZone").drop("service_zone")
temp2 = lt.withColumnRenamed("Borough",
"DBorough").withColumnRenamed("Zone","DZone").drop("service_zone")

rdd = df.rdd
rddt1 = temp1.rdd.map(lambda x: (int(x[0]),x[2]))
rddt2 = temp2.rdd.map(lambda x: (int(x[0]),x[2]))

initrdd = rdd.filter(lambda x: (x[7]!=x[8]))\
.map(lambda x:(x[7], ( (x[8],(x[4],x[16])),(x[1].month,0 if x[1].day<=15 else 1) ) ))

rdd1 = initrdd.filter(lambda x: (x[1][1][0]==1))
rdd2 = initrdd.filter(lambda x: (x[1][1][0]==2))
rdd3 = initrdd.filter(lambda x: (x[1][1][0]==3))
rdd4 = initrdd.filter(lambda x: (x[1][1][0]==4))
rdd5 = initrdd.filter(lambda x: (x[1][1][0]==5))
rdd6 = initrdd.filter(lambda x: (x[1][1][0]==6))


w1 = rdd1.join(rddt1)\
.map(lambda x:(x[1][0][0][0],(((x[1][0][0][1][0],x[1][0][0][1][1]),(x[1][0][1][0],x[1][0][1][1])),x[1]
[1])))\
.join(rddt2)\
.filter(lambda x: (x[1][0][1]!=x[1][1]))\
.map(lambda x:((x[1][0][0][1][0],x[1][0][0][1][1]),((x[1][0][0][0][0],x[1][0][0][0][1]),1)))\
.reduceByKey(lambda x,y: ((x[0][0]+y[0][0],x[0][1]+y[0][1]),x[1]+y[1]))\
.map(lambda x: (x[0],(x[1][0][0]/x[1][1],x[1][0][1]/x[1][1])))\
.sortByKey(ascending=True)\
.collect()

w2 = rdd2.join(rddt1)\
```

```
.map(lambda x:(x[1][0][0][0],(((x[1][0][0][1][0],x[1][0][0][1][1]),(x[1][0][1][0],x[1][0][1][1])),x[1]
[1])))\
.join(rddt2)\
.filter(lambda x: (x[1][0][1]!=x[1][1]))\
.map(lambda x:((x[1][0][0][1][0],x[1][0][0][1][1]),((x[1][0][0][0][0],x[1][0][0][0][1]),1)))\
.reduceByKey(lambda x,y: ((x[0][0]+y[0][0],x[0][1]+y[0][1]),x[1]+y[1]))\
.map(lambda x: (x[0],(x[1][0][0]/x[1][1],x[1][0][1]/x[1][1])))\
.sortByKey(ascending=True)\
.collect()

w3 = rdd3.join(rddt1)\
.map(lambda x:(x[1][0][0][0],(((x[1][0][0][1][0],x[1][0][0][1][1]),(x[1][0][1][0],x[1][0][1][1])),x[1]
[1])))\
.join(rddt2)\
.filter(lambda x: (x[1][0][1]!=x[1][1]))\
.map(lambda x:((x[1][0][0][1][0],x[1][0][0][1][1]),((x[1][0][0][0][0],x[1][0][0][0][1]),1)))\
.reduceByKey(lambda x,y: ((x[0][0]+y[0][0],x[0][1]+y[0][1]),x[1]+y[1]))\
.map(lambda x: (x[0],(x[1][0][0]/x[1][1],x[1][0][1]/x[1][1])))\
.sortByKey(ascending=True)\
.collect()

w4 = rdd4.join(rddt1)\
.map(lambda x:(x[1][0][0][0],(((x[1][0][0][1][0],x[1][0][0][1][1]),(x[1][0][1][0],x[1][0][1][1])),x[1]
[1])))\
.join(rddt2)\
.filter(lambda x: (x[1][0][1]!=x[1][1]))\
.map(lambda x:((x[1][0][0][1][0],x[1][0][0][1][1]),((x[1][0][0][0][0],x[1][0][0][0][1]),1)))\
.reduceByKey(lambda x,y: ((x[0][0]+y[0][0],x[0][1]+y[0][1]),x[1]+y[1]))\
.map(lambda x: (x[0],(x[1][0][0]/x[1][1],x[1][0][1]/x[1][1])))\
.sortByKey(ascending=True)\
.collect()

w5 = rdd5.join(rddt1)\
.map(lambda x:(x[1][0][0][0],(((x[1][0][0][1][0],x[1][0][0][1][1]),(x[1][0][1][0],x[1][0][1][1])),x[1]
[1])))\
.join(rddt2)\
.filter(lambda x: (x[1][0][1]!=x[1][1]))\
.map(lambda x:((x[1][0][0][1][0],x[1][0][0][1][1]),((x[1][0][0][0][0],x[1][0][0][0][1]),1)))\
.reduceByKey(lambda x,y: ((x[0][0]+y[0][0],x[0][1]+y[0][1]),x[1]+y[1]))\
.map(lambda x: (x[0],(x[1][0][0]/x[1][1],x[1][0][1]/x[1][1])))\
.sortByKey(ascending=True)\
.collect()

w6 = rdd6.join(rddt1)\
.map(lambda x:(x[1][0][0][0],(((x[1][0][0][1][0],x[1][0][0][1][1]),(x[1][0][1][0],x[1][0][1][1])),x[1]
[1])))\
.join(rddt2)\
.filter(lambda x: (x[1][0][1]!=x[1][1]))\
.map(lambda x:((x[1][0][0][1][0],x[1][0][0][1][1]),((x[1][0][0][0][0],x[1][0][0][0][1]),1)))\
```

```
.reduceByKey(lambda x,y: ((x[0][0]+y[0][0],x[0][1]+y[0][1]),x[1]+y[1]))\
.map(lambda x: (x[0],(x[1][0][0]/x[1][1],x[1][0][1]/x[1][1])))\
.sortByKey(ascending=True)\
.collect()

print(w1)
print(w2)
print(w3)
print(w4)
print(w5)
print(w6)
```

Q4:

```
df = spark.read.parquet("/data/yellow_tripdata_2022-0*.parquet")

windowC = Window.partitionBy("day").orderBy(col("congestion").desc(),col("day").asc())

w = df.select("tpep_pickup_datetime","Passenger_count")\
.withColumn("day", dayofweek("tpep_pickup_datetime"))\
.withColumn("hour", hour("tpep_pickup_datetime"))\
.groupBy("day","hour")\
.agg(avg("Passenger_count").alias("congestion"))\
.withColumn("row",row_number().over(windowC))\
.filter(col("row") <= 3)\
.drop("row")\
.drop("tpep_pickup_datetime")\
.drop("Passenger_count")

w1 = w.collect()

w.show(21)
```

Q5:

```
df = spark.read.parquet("/data/yellow_tripdata_2022-0*.parquet")

windowC = Window.partitionBy("month").orderBy(col("average tip
percentage").desc(),col("month").asc())

w = df.select("tpep_pickup_datetime","Fare_amount","Tip_amount")\
.withColumn("day", dayofmonth(df.tpep_pickup_datetime))\
.withColumn("month", month(df.tpep_pickup_datetime))\
.withColumn("tipp",df.tip_amount/df.fare_amount)\
.drop("tpep_pickup_datetime")\
.drop("Fare_amount")\
.drop("Tip_amount")\
.groupBy("month","day")\
.agg(avg("tipp").alias("average tip percentage"))\
```

```
.withColumn("row",row_number().over(windowC))\
.filter(col("row") <= 5)\
.drop("row")

w1 = w.collect()

w.show(30)
```

Github Repository:

https://github.com/ntua-el17044/advDB_t58