Αναφορά Άσκησης

Συστήματα Διαχείρισης Δεδομένων Μεγάλου Όγκου

Εργαστηριακή Άσκηση 2023/24

Ονομα	**Επώνυμο**	**ΑΜ**
ΧΡΙΣΤΙΝΑ-ΕΛΕΑΝΝΑ	ΣΑΜΑΡΑ	1084622

Βεβαιώνω ότι είμαι συγγραφέας της παρούσας εργασίας και ότι έχω αναφέρει ή παραπέμψει σε αυτήν, ρητά και συγκεκριμένα, όλες τις πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, προτάσεων ή λέξεων, είτε αυτές μεταφέρονται επακριβώς (στο πρωτότυπο ή μεταφρασμένες) είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για το συγκεκριμένο μάθημα/σεμινάριο/πρόγραμμα σπουδών.

Έχω ενημερωθεί ότι σύμφωνα με τον εσωτερικό κανονισμό λειτουργίας του Πανεπιστημίου Πατρών άρθρο 50§6, τυχόν προσπάθεια αντιγραφής ή εν γένει φαλκίδευσης της εξεταστικής και εκπαιδευτικής διαδικασίας από οιονδήποτε εξεταζόμενο, πέραν του μηδενισμού, συνιστά βαρύ πειθαρχικό παράπτωμα.

Υπογραφή

ΧΡΙΣΤΙΝΑ-ΕΛΕΑΝΝΑ ΣΑΜΑΡΑ

17 / 09 / 2024

Συνημμένα αρχεία κώδικα

Μαζί με την παρούσα αναφορά υποβάλλουμε τα παρακάτω αρχεία κώδικα

Αρχείο	Αφορά το ερώτημα	Περιγραφή/Σχόλιο
sdmd.ipynb	**1**	Παραγωγή δεδομένων εξομοίωσης σε csv
vehicles_data.csv	**1**	Το csv που παράχθηκε από την εξομοίωση
producer.py	**1**	Παραγωγός δεδομένων
consumer.py	**1**	Καταναλωτής δεδομένων

sparkjob.py	**2**	Καταναλωτής δεδομένων και επεξεργασία δεδομένων σε Spark
filtered_vehicles_data.csv	**2**	Το csv που παράχθηκε με τα επεξεργασμένα δεδομένα
mongo_spark.py	**3**	Αποθήκευση των δεδομένων σε συλλογή της MongoDB
check_mongo.py	**3**	Query 1
check_mongo2.py	**3**	Query 2
check_mongo3.py	**3**	Query 3

Τεχνικά χαρακτηριστικά περιβάλλοντος λειτουργίας

Χαρακτηριστικό	**Τιμή**
CPU model	*12th Gen Intel(R) Core(TM) i7-12700H (Host System)*
CPU clock speed	*2.30 GHz (Host System)*
Physical CPU cores	*14 (Host System)*
Logical CPU cores	*20 (Host System)*
RAM	*16 (VM)*
Secondary Storage Type	*SSD (VM)*

Ερώτημα 1: Παραγωγή δεδομένων

1) Παραγωγή αρχείου csv από την εξομοίωση

Για την παραγωγή του αρχείου csv με τα δεδομένα της εξομοίωσης χρησιμοποιήθηκε η συνάρτηση vehicles_to_pandas(), όπως φαίνεται και στο παρακάτω screenshot.

```
df = W.analyzer.vehicles_to_pandas()

# save df to vehicles_data.csv
df.to_csv("vehicles_data.csv", index=False)
```

2) Εγκατάσταση και αρχικοποίηση kafka

Η εγκατάσταση του kafka έγινε ακολουθώντας τις οδηγίες της επίσημης ιστοσελίδας, κατεβάζοντας και εξάγοντας το tgz αρχείο σε έναν φάκελο με το όνομα kafka.

```
christina@christina-virtualbox:~/Downloads/kafka$ bin/kafka-topics.sh --version
3.7.0
```

Για την αρχικοποίηση του περιβάλλοντος kafka εκτελούνται με την σειρά και σε διαφορετικά terminals οι ακόλουθες εντολές:

bin/zookeeper-server-start.sh config/zookeeper.properties ***(έναρξη υπηρεσίας Zookeeper)***

bin/kafka-server-start.sh config/server.properties ***(έναρξη υπηρεσίας kafka broker)***

bin/kafka-topics.sh --create --topic vehicles --bootstrap-server localhost:9092 ***(δημιουργία θέματος-topic)***

Για την αποφυγή ορισμένων warnings κατά την εκτέλεση του project, το topic που δημιουργήθηκε ονομάστηκε **vehicles** αντί vehicle_positions.

3) Kafka producer

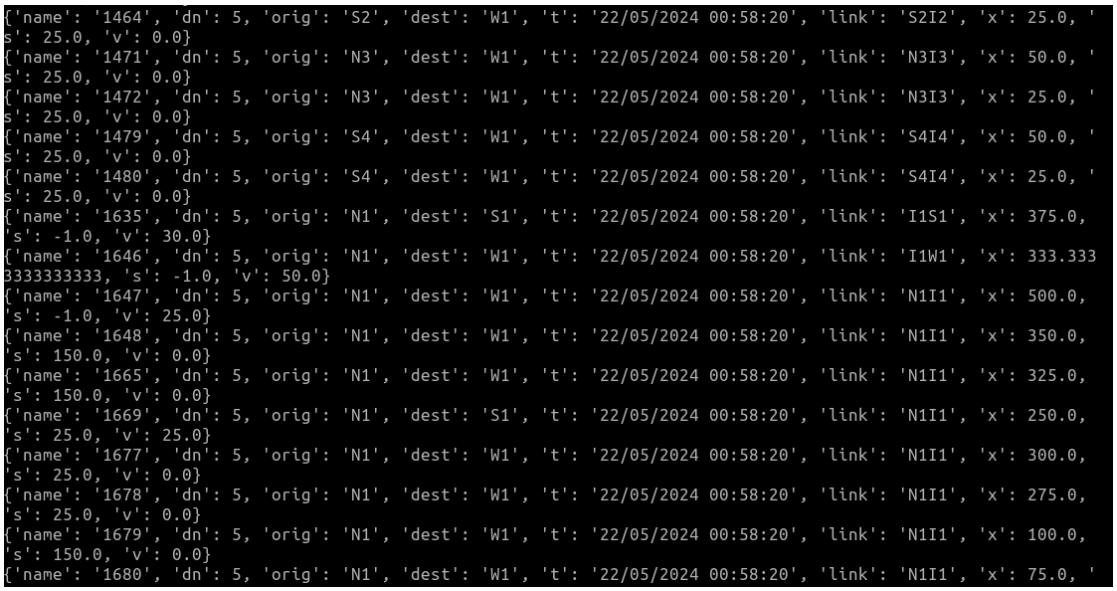
Ο κώδικας του producer διαβάζει τα δεδομένα από το αρχείο CSV που δημιουργήθηκε νωρίτερα και στέλνει τις εγγραφές στο θέμα **vehicles** Kafka. Οι εγγραφές αποστέλλονται σε προκαθορισμένο ρυθμό (κάθε 100 δευτερόλεπτα), με βάση την χρονική στιγμή που δημιουργήθηκαν και το πρόγραμμα διακόπτεται για μερικά δευτερόλεπτα μεταξύ των επαναλήψεων. Ο κώδικας προσαρμόζει τους τύπους δεδομένων κάθε γραμμής πριν τα στείλει στο Kafka, ενώ επίσης τα εκτυπώνει για εύκολο debugging. Μετά το τέλος της διαδικασίας, ο παραγωγός Kafka κλείνει.

```
christina@christina-virtualbox:~/Downloads/kafka$ bin/kafka-topics.sh --creat
e --topic vehicles --bootstrap-server localhost:9092
Created topic vehicles.
christina@christina-virtualbox:~/Downloads/kafka$
```

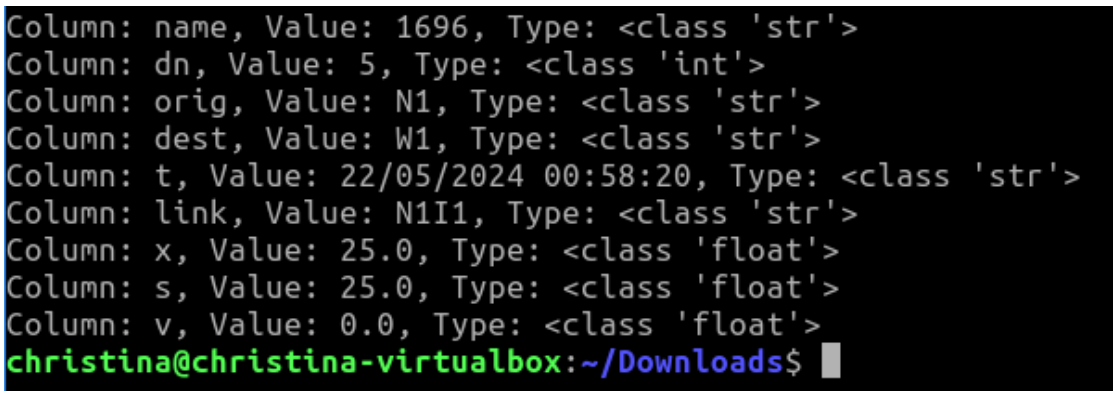
Ο producer εκτελείται με την ακόλουθη εντολή:

python3 producer.py

Στιγμιότυπο παραγωγής και αποστολής μηνυμάτων:



Τα στοιχεία του τελευταίου μηνύματος:

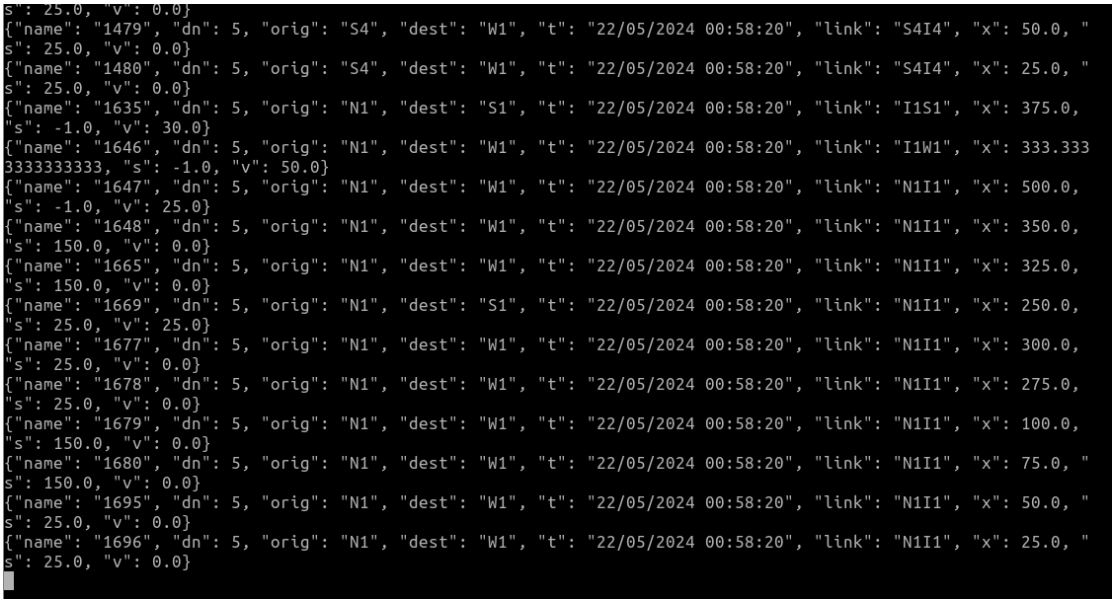


4) Simple kafka consumer

Ο consumer εκτελείται με την ακόλουθη εντολή:

spark-submit --packages org.apache.spark:spark-streaming-kafka-0-10_2.12:3.0.1,org.apache.spark:spark-sql-kafka-0-10_2.12:3.0.1 ~/Downloads/consumer.py localhost:9092 subscribe vehicles

Τα μηνύματα όπως φαίνονται από τον kafka consumer:



Ο kafka consumer δεν σταματάει, μένει σε κατάσταση αναμονής μέχρι να τερματιστεί από τον χρήστη.

Ερώτημα 2: Κατανάλωση και επεξεργασία με Spark

Το αρχείο κατανάλωσης και επεξεργασίας Spark εκτελείται με την ακόλουθη εντολή:

spark-submit --packages org.apache.spark:spark-streaming-kafka-0-10_2.12:3.0.1,org.apache.spark:spark-sql-kafka-0-10_2.12:3.0.1 ~/Downloads/sparkjob.py localhost:9092 subscribe vehicles

Η διεργασία επεξεργάζεται εισερχόμενα δεδομένα σε πραγματικό χρόνο και παράγει ένα Spark DataFrame με τα πεδία time, link, vcount και vspeed (χρονική στιγμή, ακμή, πλήθος οχημάτων ακμής, μέση ταχύτητα ακμής).

Αρχικά ορίστηκε το σχήμα των εισερχόμενων δεδομένων (name, link, dn, orig, dest, t, x, s, v) και η στήλη “t” μετατράπηκε σε στήλη “time” με τον σωστό τύπο. Έπειτα έγινε η ομαδοποίηση των δεδομένων ανά χρονική στιγμή και ακμή, και ο υπολογισμός τόσο της μέσης ταχύτητας των οχημάτων (vspeed) όσο και του πλήθους των οχημάτων (vcount).

Τα αποτελέσματα παρουσιάζονται στην κονσόλα και αποθηκεύονται στη μνήμη για περαιτέρω επεξεργασία. Τέλος, εξάγονται επίσης σε αρχείο CSV, για δυνατότητα περαιτέρω ανάλυσης.

Τα δεδομένα έτσι όπως εκτυπώνονται στην κονσόλα:

```
Batch: 2
-------------------------------------------
+-------------------+----+------------------+------+
|               time|link|            vspeed|vcount|
+-------------------+----+------------------+------+
|2024-05-22 00:11:40|I3I2|               0.0|    16|
|2024-05-22 00:10:00|E1I4|3.3333333333333335|    15|
|2024-05-22 00:10:00|N1I1|               5.0|    10|
|2024-05-22 00:13:20|E1I4|               0.0|    18|
|2024-05-22 00:11:40|N3I3| 4.090909090909091|    11|
|2024-05-22 00:11:40|E1I4|               0.0|    20|
|2024-05-22 00:11:40|S2I2| 4.090909090909091|    11|
|2024-05-22 00:15:00|S2I2|3.3333333333333335|    12|
|2024-05-22 00:13:20|I3I2|               0.0|    16|
|2024-05-22 00:11:40|S4I4| 4.090909090909091|    11|
|2024-05-22 00:13:20|N3I3|3.3333333333333335|    12|
|2024-05-22 00:13:20|I2I1|1.1538461538461537|    13|
|2024-05-22 00:13:20|S2I2|3.3333333333333335|    12|
|2024-05-22 00:10:00|I3I2|               0.0|    16|
|2024-05-22 00:11:40|N1I1|               5.0|    10|
|2024-05-22 00:11:40|I4N4|              30.0|     1|
|2024-05-22 00:15:00|N1I1|               5.0|    10|
|2024-05-22 00:15:00|I1W1|              12.5|     4|
|2024-05-22 00:13:20|S4I4|3.3333333333333335|    12|
|2024-05-22 00:10:00|I1W1|              12.5|     4|
+-------------------+----+------------------+------+
```

Τα δεδομένα στο αρχείο CSV που δημιουργείται:

	time	link	vspeed	vcount
2	2024-05-22T00:11:40.000+03:00	I3I2	0	16
3	2024-05-22T00:26:40.000+03:00	S2I2	4.09090909090909	11
4	2024-05-22T00:10:00.000+03:00	E1I4	3.33333333333333	15
5	2024-05-22T00:21:40.000+03:00	I1S1	30	1
6	2024-05-22T00:23:20.000+03:00	I3I2	0	16
7	2024-05-22T00:23:20.000+03:00	S2I2	3.33333333333333	12
8	2024-05-22T00:20:00.000+03:00	I2I1	1.15384615384615	13
9	2024-05-22T00:26:40.000+03:00	N1I1	5	10
10	2024-05-22T00:26:40.000+03:00	I3I2	0	16
11	2024-05-22T00:10:00.000+03:00	N1I1	5	10
12	2024-05-22T00:13:20.000+03:00	E1I4	0	18
13	2024-05-22T00:11:40.000+03:00	N3I3	4.09090909090909	11
14	2024-05-22T00:20:00.000+03:00	E1I4	0	19
15	2024-05-22T00:25:00.000+03:00	N3I3	3.33333333333333	12
16	2024-05-22T00:16:40.000+03:00	I1W1	12.5	4
17	2024-05-22T00:11:40.000+03:00	E1I4	0	20
18	2024-05-22T00:11:40.000+03:00	S2I2	4.09090909090909	11
19	2024-05-22T00:30:00.000+03:00	I3I2	0	3
20	2024-05-22T00:20:00.000+03:00	I4N4	30	1
21	2024-05-22T00:18:20.000+03:00	I4N4	30	1
22	2024-05-22T00:21:40.000+03:00	S2I2	4.09090909090909	11
23	2024-05-22T00:21:40.000+03:00	N1I1	5	10
24	2024-05-22T00:18:20.000+03:00	E1I4	0	19
25	2024-05-22T00:30:00.000+03:00	I4I3	0	9
26	2024-05-22T00:20:00.000+03:00	N3I3	3.33333333333333	12
27	2024-05-22T00:23:20.000+03:00	I2I1	1.15384615384615	13
28	2024-05-22T00:15:00.000+03:00	S2I2	3.33333333333333	12
29	2024-05-22T00:21:40.000+03:00	N3I3	4.09090909090909	11
30	2024-05-22T00:25:00.000+03:00	N1I1	5	10

Ερώτημα 3: Αποθήκευση σε MongoDB

Το αρχείο αποθήκευσης σε Mongo εκτελείται με την εξής εντολή:

(πρώτα η εκκίνηση της mongo): *sudo systemctl start mongod*

spark-submit --packages org.apache.spark:spark-streaming-kafka-0-10_2.12:3.0.1,org.apache.spark:spark-sql-kafka-0-10_2.12:3.0.1,org.mongodb.spark:mongo-spark-connector_2.12:10.3.0,org.mongodb:mongodb-driver-sync:5.1.0 ~/Downloads/mongo_spark.py localhost:9092 subscribe vehicles

Η αποθήκευση των δεδομένων από Spark σε MongoDB πραγματοποιήθηκε μέσω του MongoDB Connector. Δημιουργήθηκαν δύο συλλογές στη βάση δεδομένων bigdata:

- raw_data: Αποθηκεύονται τα ωμά δεδομένα από την πηγή, όπως προέρχονται από το Kafka.
- aggregated_data: Αποθηκεύονται τα επεξεργασμένα δεδομένα.

Αφού διαβαστούν τα δεδομένα από το Kafka και μετατραπούν σε DataFrame με τη χρήση του Spark, τα δεδομένα αποθηκεύονται στη συλλογή raw_data της βάσης MongoDB μέσω της παρακάτω συνάρτησης:

def saveRawData(message, bid): message.write \ .format("mongodb") \ .mode("append") \ .option("database", "bigdata") \ .option("collection", "raw_data") \ .save() message.show()

Ομοίως, μετά την επεξεργασία των δεδομένα για να υπολογιστούν οι μέσες ταχύτητες και το πλήθος οχημάτων ανά χρονική στιγμή και ακμή σε spark (όπως είδαμε στο προηγούμενο ερώτημα), αποθηκεύονται στη συλλογή aggregated_data:

def saveProcessedData(message, bid): message.write\.format("mongodb")\.mode("append")\.option("database","bigdata") \.option("collection","aggregated_data") \.save() message.show()

Τα ωμά δεδομένα:

name	dn	orig	dest	t	link	x	s	v
946	5	E1	W1	22/05/2024 00:58:20	E1I4	425.0	75.0	0.0
947	5	E1	W1	22/05/2024 00:58:20	E1I4	425.0	75.0	0.0
976	5	E1	W1	22/05/2024 00:58:20	E1I4	350.0	150.0	0.0
977	5	E1	W1	22/05/2024 00:58:20	E1I4	350.0	75.0	0.0
986	5	E1	W1	22/05/2024 00:58:20	E1I4	350.0	75.0	0.0
995	5	E1	W1	22/05/2024 00:58:20	E1I4	275.0	75.0	0.0
996	5	E1	W1	22/05/2024 00:58:20	E1I4	275.0	75.0	0.0
997	5	E1	W1	22/05/2024 00:58:20	E1I4	275.0	75.0	0.0
1014	5	E1	W1	22/05/2024 00:58:20	E1I4	200.0	75.0	0.0
1015	5	E1	W1	22/05/2024 00:58:20	E1I4	200.0	75.0	0.0
1016	5	E1	W1	22/05/2024 00:58:20	E1I4	200.0	75.0	0.0
1017	5	E1	W1	22/05/2024 00:58:20	E1I4	125.0	75.0	0.0
1034	5	E1	W1	22/05/2024 00:58:20	E1I4	125.0	75.0	0.0
1035	5	E1	W1	22/05/2024 00:58:20	E1I4	125.0	75.0	0.0
1052	5	E1	W1	22/05/2024 00:58:20	E1I4	50.0	75.0	0.0
1053	5	E1	W1	22/05/2024 00:58:20	E1I4	50.0	75.0	0.0
1054	5	E1	W1	22/05/2024 00:58:20	E1I4	50.0	75.0	0.0
1283	5	S4	W1	22/05/2024 00:58:20	I2I1	425.0	-1.0	0.0
1284	5	S4	W1	22/05/2024 00:58:20	I2I1	350.0	75.0	0.0
1289	5	S4	W1	22/05/2024 00:58:20	I2I1	350.0	75.0	0.0

Τα επεξεργασμένα δεδομένα:

time	link	vspeed	vcount
2024-05-22 00:55:00	S2I2	3.3333333333333335	12
2024-05-22 00:55:00	I3S3	30.0	1
2024-05-22 00:56:40	S4I4	3.3333333333333335	12
2024-05-22 00:55:00	N3I3	3.3333333333333335	12
2024-05-22 00:55:00	I2I1	2.5	12
2024-05-22 00:56:40	E1I4	0.0	18
2024-05-22 00:55:00	I1W1	12.5	4
2024-05-22 00:55:00	N1I1	5.0	10
2024-05-22 00:56:40	S2I2	3.3333333333333335	12
2024-05-22 00:56:40	I1W1	12.5	4
2024-05-22 00:56:40	I4I3	0.0	18
2024-05-22 00:55:00	I3I2	0.0	16
2024-05-22 00:55:00	I4N4	30.0	1
2024-05-22 00:55:00	S4I4	3.3333333333333335	12
2024-05-22 00:56:40	I4N4	30.0	1
2024-05-22 00:56:40	N1I1	5.0	10
2024-05-22 00:55:00	E1I4	0.0	19
2024-05-22 00:56:40	I3I2	0.0	15
2024-05-22 00:56:40	I3S3	30.0	1
2024-05-22 00:56:40	N3I3	3.3333333333333335	12

ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΡΩΤΗΜΑΤΩΝ

Τα 3 ερωτήματα εκτελούνται σε διαφορετικά αρχεία. (check_mongo1, check_mongo2 και check_mongo3).

Η εκτέλεση των ερωτημάτων γίνεται με την εξής εντολή:

*spark-submit --packages org.apache.spark:spark-streaming-kafka-0-10_2.12:3.0.1,org.apache.spark:spark-sql-kafka-0-10_2.12:3.0.1,org.mongodb.spark:mongo-spark-connector_2.12:10.3.0,org.mongodb:mongodb-driver-sync:4.8.1 ~/Downloads/**check_mongo1.py 00:00:00 00:5:55:00***

Τα δύο τελευταία ορίσματα είναι η χρονική περίοδος για την οποία θέλουμε να εκτελέσουμε το ερώτημα.

ΕΡΩΤΗΜΑ 1

Αρχικά γίνεται η σύνδεση με την MongoDB και λαμβάνονται οι τιμές έναρξης και λήξης που δίνονται από τον χρήστη κατά την εκτέλεση, οι οποίες μετατρέπονται σε μορφή ISO. Έπειτα διαβάζονται τα δεδομένα από τη MongoDB με βάση το aggregation pipeline, το οποίο φιλτράρει τα δεδομένα χρονικά και επιστρέφει τα αποτελέσματα ταξινομημένα ανά το πλήθος οχημάτων (vcount).

```
+--------------------+-------------------+----+------+------+
|                 _id|               time|link|vspeed|vcount|
+--------------------+-------------------+----+------+------+
|66e9aac6b4c74b381...|2024-05-22 00:05:00|E1I4|  35.0|     1|
+--------------------+-------------------+----+------+------+
```

ΕΡΩΤΗΜΑ 2

Οι χρονικές τιμές έναρξης και λήξης (start_time και end_time) δίνονται ως είσοδος και μετατρέπονται σε μορφή ISO. Στην συνέχεια ορίζεται το pipeline για τη MongoDB, το οποίο φιλτράρει τις εγγραφές βάσει του χρονικού διαστήματος και ταξινομεί τα δεδομένα κατά φθίνουσα σειρά ταχύτητας. Επιστρέφεται το όχημα με τη μέγιστη ταχύτητα στο καθορισμένο διάστημα.

```
+--------------------+-------------------+----+------+------+
|                 _id|               time|link|vspeed|vcount|
+--------------------+-------------------+----+------+------+
|66e9aac7b4c74b381...|2024-05-22 00:03:20|E1I4|  50.0|     1|
+--------------------+-------------------+----+------+------+
```

ΕΡΩΤΗΜΑ 3

Στο συγκεκριμένο ερώτημα γίνεται χρήση παραθύρων (Spark Window Functions) και υπολογίζονται οι μεταβολές στην τιμή x (θέση) για κάθε όχημα με βάση το πεδίο χρόνου. Χρησιμοποιείται η συνάρτηση lag για να πάρει την προηγούμενη τιμή x_lag και κατόπιν να υπολογιστεί η απόσταση που διανύθηκε. Στο τέλο εκτυπώνεται το όχημα που έχει διανύσει τη μεγαλύτερη συνολική απόσταση, ταξινομώντας τα δεδομένα με βάση το άθροισμα της απόστασης (sum(x_lag)) και επιλέγοντας το όχημα με την μεγαλύτερη τιμή.

```
Result: Vehicle with the longest distance is 473 with 2400.0 total distance.
```

Σχολιασμός αποτελεσμάτων

Το project ανέδειξε την αποτελεσματικότητα του Apache Spark και της MongoDB για την επεξεργασία και ανάλυση **μεγάλων όγκων δεδομένων σε πραγματικό χρόνο**. Η χρήση αυτών των εργαλείων μας επέτρεψε να διαχειριστούμε και να επεξεργαστούμε δεδομένα από Kafka streams, να τα αποθηκεύσουμε στη MongoDB και να εκτελέσουμε σύνθετες αναλύσεις. Η εφαρμογή του Spark Streaming για την επεξεργασία των δεδομένων και οι προηγμένες λειτουργίες του PySpark (π.χ. aggregations), αποδείχθηκαν κρίσιμες για την διαχείριση και ανάλυση των δεδομένων σε ροή, επιβεβαιώνοντας τη δύναμη αυτών των τεχνολογιών για την επεξεργασία μεγάλων δεδομένων σε πραγματικό χρόνο.

Βιβλιογραφία

- Toruseo. *UXsim Documentation*. https://toruseo.jp/UXsim/docs/index.html
- Toruseo. *UXsim GitHub Repository*. https://github.com/toruseo/UXsim
- Kafka Python. *Kafka-Python Documentation*. https://kafka-python.readthedocs.io/en/master/index.html
- Apache Kafka. *Apache Kafka Quickstart*. https://kafka.apache.org/quickstart
- Hesbon. *Apache Kafka with Python*. https://dev.to/hesbon/apache-kafka-with-python-laa
- Apache Spark. *PySpark Getting Started*. https://spark.apache.org/docs/latest/api/python/getting_started/install.html
- Kattepogu, S. *Python Spark Transformations on Kafka Data*. https://sandeepkattepogu.medium.com/python-spark-transformations-on-kafka-data-8a19b498b32c
- Expedia Group Tech. *Working with JSON in Apache Spark*. https://medium.com/expedia-group-tech/working-with-json-in-apache-spark-1ecf553c2a8c
- MongoDB. *MongoDB Community Edition Download*. https://www.mongodb.com/try/download/community
- MongoDB. *MongoDB Spark Connector Documentation*. https://www.mongodb.com/docs/spark-connector/current/
- Bobcares. *PySpark and MongoDB Pipeline*. https://bobcares.com/blog/pyspark-mongodb-pipeline/
- MongoDB. *Getting Started with MongoDB, PySpark, and Jupyter Notebook*. https://www.mongodb.com/blog/post/getting-started-with-mongodb-pyspark-and-jupyter-notebook
- Walters, R. *Mongo-Spark-Jupyter GitHub Repository*. https://github.com/RWaltersMA/mongo-spark-jupyter