


Συστήματα Διαχείρισης Δεδομένων Μεγάλου Όγκου

Εργαστηριακή Άσκηση 2024/25

Όνομα	Επώνυμο	ΑΜ
Χριστιάνα	Μουσελέ	1090068

Βεβαιώνω ότι είμαι συγγραφέας της παρούσας εργασίας και ότι έχω αναφέρει ή παραπέμπει 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  
  
01 / 06 / 2025

Συνημμένα αρχεία κώδικα

Μαζί με την παρούσα αναφορά υποβάλλουμε τα παρακάτω αρχεία κώδικα

Αρχείο	Αφορά το ερώτημα	Περιγραφή/Σχόλιο
<i>producer.py</i>	1	Υλοποιείται ο <i>KafkaProducer</i>
<i>consumer.py</i>	1	Υλοποιείται ο <i>KafkaConsumer</i>
<i>kafka_spark_stream.py</i>	2 , 3	Υλοποιείται το ερώτημα 2 καθώς και το πρώτο μέρος του ερωτήματος 3
<i>query_mongo.py</i>	3	Υλοποιείται το δεύτερο μέρος του ερωτήματος 3

## Τεχνικά χαρακτηριστικά περιβάλλοντος λειτουργίας

Χαρακτηριστικό	Τιμή
CPU model	<i>Intel Core i7-10750H</i>
CPU clock speed	<i>2.5 GHz (βάση)</i>
Physical CPU cores	<i>6</i>
Logical CPU cores	<i>12</i>
RAM	<i>8 GB</i>
Secondary Storage Type	<i>SSD</i>

## Περιεχόμενα

Τεχνικά χαρακτηριστικά περιβάλλοντος λειτουργίας .....	2
Ερώτημα 1: Παραγωγή δεδομένων .....	3
Ερώτημα 2: Κατανάλωση και επεξεργασία με Spark.....	4
Ερώτημα 3: Αποθήκευση σε MongoDB .....	5
Σχολιασμός αποτελεσμάτων .....	11
Βιβλιογραφία .....	11

## Ερώτημα 1: Παραγωγή δεδομένων

Για την παραγωγή δεδομένων τρέχουμε το script που δίνεται (hotel\_sim.py) όπου και δημιουργείται ένα αρχείο csv (hotel\_clickstream.csv) το οποίο περιέχει χρονικά δεδομένα από ενέργειες χρηστών σε ιστοσελίδα ξενοδοχείου.

```
user_id,session_id,timestamp,event_type,booking_id,check_in_date,check_out_date,hotel_id,item_id,location,num_guests,page_url,payment_method,price,room_type,total_price
user_00466,session_user_00466_002_904c4d,2025-05-11T19:00:37.906064,page_view,,,,,,https://hotelbooking.example.com/,...
user_00753,session_user_00753_001_f87479,2025-05-11T19:01:49.906064,page_view,,,,,,https://hotelbooking.example.com/,...
user_00812,session_user_00812_004_9507f9,2025-05-11T19:01:57.906064,page_view,,,,,,https://hotelbooking.example.com/,...
user_00323,session_user_00323_003_1aa0f8,2025-05-11T19:06:07.906064,page_view,,,,,,https://hotelbooking.example.com/,...
user_00223,session_user_00223_001_f7aca9,2025-05-11T19:06:12.906064,page_view,,,,,,https://hotelbooking.example.com/,...
user_00282,session_user_00282_001_b497c0,2025-05-11T19:08:37.906064,page_view,,,,,,https://hotelbooking.example.com/,...
user_00753,session_user_00753_001_f87479,2025-05-11T19:08:37.906064,logout,,,,,,https://hotelbooking.example.com/logout,...
user_00466,session_user_00466_002_904c4d,2025-05-11T19:09:58.906064,search_hotels,,2025-08-28,2025-08-31,,Dubai,4,...
user_00323,session_user_00323_003_1aa0f8,2025-05-11T19:10:14.906064,search_hotels,,2026-04-28,2026-05-04,,Singapore,4,...
user_00812,session_user_00812_004_9507f9,2025-05-11T19:10:41.906064,search_hotels,,2025-07-09,2025-07-17,,Singapore,4,...
user_00323,session_user_00323_003_1aa0f8,2025-05-11T19:11:34.906064,view_hotel_details,,,HOTEL_0015,,,https://hotelbooking.example.com/hotel/HOTEL_0015,...
user_00466,session_user_00466_002_904c4d,2025-05-11T19:11:36.906064,view_hotel_details,,,HOTEL_0297,,,https://hotelbooking.example.com/hotel/HOTEL_0297,...
user_00323,session_user_00323_003_1aa0f8,2025-05-11T19:11:48.906064,view_hotel_details,,,HOTEL_0151,,,https://hotelbooking.example.com/hotel/HOTEL_0151,...
user_00812,session_user_00812_004_9507f9,2025-05-11T19:12:02.906064,view_hotel_details,,,HOTEL_0367,,,https://hotelbooking.example.com/hotel/HOTEL_0367,...
user_00323,session_user_00323_003_1aa0f8,2025-05-11T19:13:27.906064,view_hotel_details,,,HOTEL_0103,,,https://hotelbooking.example.com/hotel/HOTEL_0103,...
user_00466,session_user_00466_002_904c4d,2025-05-11T19:13:27.906064,view_hotel_details,,,HOTEL_0298,,,https://hotelbooking.example.com/hotel/HOTEL_0298,...
user_00812,session_user_00812_004_9507f9,2025-05-11T19:13:33.906064,view_hotel_details,,,HOTEL_0417,,,https://hotelbooking.example.com/hotel/HOTEL_0417,...
user_00323,session_user_00323_003_1aa0f8,2025-05-11T19:13:38.906064,view_hotel_details,,,HOTEL_0200,,,https://hotelbooking.example.com/hotel/HOTEL_0200,...
user_00087,session_user_00087_002_d6c34b,2025-05-11T19:14:09.906064,page_view,,,,,,https://hotelbooking.example.com/,...
user_00323,session_user_00323_003_1aa0f8,2025-05-11T19:14:14.906064,view_hotel_details,,,HOTEL_0041,,,https://hotelbooking.example.com/hotel/HOTEL_0041,...
user_00812,session_user_00812_004_9507f9,2025-05-11T19:14:45.906064,view_hotel_details,,,HOTEL_0343,,,https://hotelbooking.example.com/hotel/HOTEL_0343,...
user_00466,session_user_00466_002_904c4d,2025-05-11T19:14:49.906064,view_hotel_details,,,HOTEL_0248,,,https://hotelbooking.example.com/hotel/HOTEL_0248,...
user_00812,session_user_00812_004_9507f9,2025-05-11T19:14:56.906064,view_hotel_details,,,HOTEL_0275,,,https://hotelbooking.example.com/hotel/HOTEL_0275,...
user_00323,session_user_00323_003_1aa0f8,2025-05-11T19:15:11.906064,add_to_cart,,2026-04-28,2026-05-04,HOTEL_0041,6308b887-0e1e-4e90-833b-8ccd49826928,Singapore,,https://
```

Αφού έχουμε κατεβάσει το kafka , κάνουμε τον kafka server να τρέχει :

```
christiana@Christiana:~$ cd kafka_2.13-4.0.0
christiana@Christiana:~/kafka_2.13-4.0.0$ bin/kafka-server-start.sh config/server.properties
[2025-06-01 16:30:43,802] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)
[2025-06-01 16:30:45,615] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)
```

Δημιουργήσαμε το kafka topic με την εντολή :

```
bin/kafka-topics.sh --create --topic website_actions --bootstrap-server localhost:9092
```

Ο Kafka Producer διαβάζει τα δεδομένα, υπολογίζει τη χρονική μετατόπιση και αποστέλλει κάθε εγγραφή στο Kafka topic (website\_actions) ανά 5 δευτερόλεπτα, χρησιμοποιώντας JSON serialization και μεμονωμένες αποστολές ανά event.

```
christiana@Christiana:~$ cd 'Big Data Project'
christiana@Christiana:~/Big Data Project$ python3 producer.py
[INFO] Εκκίνηση η εφαρμογή στο Kafka topic 'website_actions'...
[SENT] 2025-06-01T16:41:22.553956 - page_view
[SENT] 2025-06-01T16:42:34.553956 - page_view
[SENT] 2025-06-01T16:42:42.553956 - page_view
[SENT] 2025-06-01T16:46:52.553956 - page_view
[SENT] 2025-06-01T16:46:57.553956 - page_view
[SENT] 2025-06-01T16:49:22.553956 - page_view
[SENT] 2025-06-01T16:49:22.553956 - logout
[SENT] 2025-06-01T16:50:43.553956 - search_hotels
[SENT] 2025-06-01T16:50:59.553956 - search_hotels
[SENT] 2025-06-01T16:51:26.553956 - search_hotels
[SENT] 2025-06-01T16:52:19.553956 - view_hotel_details
[SENT] 2025-06-01T16:52:21.553956 - view_hotel_details
[SENT] 2025-06-01T16:52:33.553956 - view_hotel_details
[SENT] 2025-06-01T16:52:47.553956 - view_hotel_details
[SENT] 2025-06-01T16:53:22.553956 - view_hotel_details
[SENT] 2025-06-01T16:54:12.553956 - view_hotel_details
[SENT] 2025-06-01T16:54:18.553956 - view_hotel_details
[SENT] 2025-06-01T16:54:23.553956 - view_hotel_details
[SENT] 2025-06-01T16:54:54.553956 - page_view
[SENT] 2025-06-01T16:54:59.553956 - view_hotel_details
[SENT] 2025-06-01T16:55:30.553956 - view_hotel_details
[SENT] 2025-06-01T16:55:34.553956 - view_hotel_details
[SENT] 2025-06-01T16:55:41.553956 - view_hotel_details
[SENT] 2025-06-01T16:55:56.553956 - add_to_cart
[SENT] 2025-06-01T16:56:14.553956 - page_view
[SENT] 2025-06-01T16:56:49.553956 - logout
[SENT] 2025-06-01T16:57:00.553956 - view_hotel_details
[SENT] 2025-06-01T16:57:33.553956 - page_view
[SENT] 2025-06-01T16:58:20.553956 - search_hotels
[SENT] 2025-06-01T16:58:50.553956 - logout
[SENT] 2025-06-01T16:58:58.553956 - view_hotel_details
[SENT] 2025-06-01T16:59:52.553956 - view_cart
[SENT] 2025-06-01T17:00:13.553956 - view_hotel_details
[SENT] 2025-06-01T17:00:23.553956 - logout
[SENT] 2025-06-01T17:00:51.553956 - add_to_cart
[SENT] 2025-06-01T17:01:06.553956 - page_view
[SENT] 2025-06-01T17:01:40.553956 - start_checkout
```

Ο Kafka Consumer, από την άλλη, συνδέεται στο ίδιο topic, λαμβάνει τα events καθώς φτάνουν και τα εμφανίζει σε πραγματικό χρόνο στην κονσόλα, επιτρέποντας την επιβεβαίωση της επιτυχούς ροής δεδομένων από τον producer στον consumer.

```
christiana@Christiana: ~/kafka x  christiana@Christiana: ~/Big Data Project x  christiana@Christiana: ~/Big Data Project x  + - v
christiana@Christiana:~/kafka$ cd 'Big Data Project'
christiana@Christiana:~/Big Data Project$ python3 consumer.py
[INFO] Περιμένουμε μηνύματα από το Kafka topic 'website_actions'...

[RECEIVED] 2025-06-01T16:41:22.553956 - page_view
[RECEIVED] 2025-06-01T16:42:34.553956 - page_view
[RECEIVED] 2025-06-01T16:42:42.553956 - page_view
[RECEIVED] 2025-06-01T16:46:52.553956 - page_view
[RECEIVED] 2025-06-01T16:46:57.553956 - page_view
[RECEIVED] 2025-06-01T16:49:22.553956 - page_view
[RECEIVED] 2025-06-01T16:49:22.553956 - logout
[RECEIVED] 2025-06-01T16:50:43.553956 - search_hotels
[RECEIVED] 2025-06-01T16:50:59.553956 - search_hotels
[RECEIVED] 2025-06-01T16:51:26.553956 - search_hotels
[RECEIVED] 2025-06-01T16:52:19.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:52:21.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:52:33.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:52:47.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:53:22.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:54:12.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:54:18.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:54:23.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:54:54.553956 - page_view
[RECEIVED] 2025-06-01T16:54:59.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:55:30.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:55:34.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:55:41.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:55:56.553956 - add_to_cart
[RECEIVED] 2025-06-01T16:56:14.553956 - page_view
[RECEIVED] 2025-06-01T16:56:49.553956 - logout
[RECEIVED] 2025-06-01T16:57:00.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:57:33.553956 - page_view
[RECEIVED] 2025-06-01T16:58:20.553956 - search_hotels
[RECEIVED] 2025-06-01T16:58:50.553956 - logout
[RECEIVED] 2025-06-01T16:58:58.553956 - view_hotel_details
[RECEIVED] 2025-06-01T16:59:52.553956 - view_hotel_details
[RECEIVED] 2025-06-01T17:00:13.553956 - view_hotel_details
[RECEIVED] 2025-06-01T17:00:23.553956 - logout
[RECEIVED] 2025-06-01T17:00:51.553956 - add_to_cart
[RECEIVED] 2025-06-01T17:01:06.553956 - page_view
[RECEIVED] 2025-06-01T17:01:40.553956 - start_checkout
```

## Ερώτημα 2: Κατανάλωση και επεξεργασία με Spark

Αφού έχουμε εγκαταστήσει το PySpark τρέχουμε το πρόγραμμα με την εντολή :

```
spark-submit \
```

```
--packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.4.2,org.mongodb.spark:mongo-  
spark-connector_2.12:10.2.0 \
```

```
kafka_spark_stream.py
```

Το script `kafka_spark_stream.py` χρησιμοποιεί το Spark Structured Streaming για να διαβάσει και να επεξεργαστεί δεδομένα σε πραγματικό χρόνο από το Kafka topic `website_actions`. Αρχικά, δημιουργείται ένα `SparkSession` και ρυθμίζεται να διαβάζει συνεχώς νέα δεδομένα από το Kafka, τα οποία καταφθάνουν σε μορφή JSON. Μέσω ενός `schema`, τα δεδομένα αναλύονται ώστε κάθε πεδίο να είναι κατάλληλα δομημένο.

Το stream καθαρίζεται φιλτράροντας συγκεκριμένους τύπους ενεργειών (`search_hotels`, `complete_booking`) και αγνοούνται εγγραφές χωρίς τοποθεσία. Στη συνέχεια, εφαρμόζεται ομαδοποίηση κατά τοποθεσία με χρήση `watermarking` για διαχείριση καθυστερημένων δεδομένων, και υπολογίζονται οι βασικοί δείκτες:

- Πλήθος αναζητήσεων (`search_volume`)
- Πλήθος κρατήσεων (`bookings_volume`)
- Συνολικό ποσό συναλλαγών (`sales_volume`)

Επιπλέον, υπολογίζεται πόσα λεπτά έχουν περάσει από την έναρξη της εξομοίωσης(`time`) .

Τέλος, το αποτέλεσμα εμφανίζεται στην κονσόλα κάθε 5 λεπτά μέσω του `writeStream().format("console")`, επιβεβαιώνοντας την ορθή ροή, επεξεργασία και δυναμική ενημέρωση των δεδομένων από το Spark.

```
christiana@Christiana: ~/kafka, x christiana@Christiana: ~/Big Di, x christiana@Christiana: ~/Big Di, x christiana@Christiana: ~/Big Di, x christiana@Christiana: ~
[INFO] Writing batch 16 with 1 rows to MongoDB...
[INFO] Writing batch 17 with 1 rows to MongoDB...

Batch: 4
-----
|time|destination_name|search_volume|bookings_volume|sales_volume|
-----|-----|-----|-----|-----|
|15|London|1|0|0.0|
|15|Sydney|1|0|0.0|
|15|Rome|0|1|1530.65|
|15|Rio de Janeiro|0|1|1996.43|
-----

[INFO] Writing batch 4 with 2 rows to MongoDB...
[INFO] Writing batch 18 with 1 rows to MongoDB...
[INFO] Writing batch 19 with 1 rows to MongoDB...
[INFO] Writing batch 20 with 2 rows to MongoDB...
[INFO] Writing batch 21 with 1 rows to MongoDB...
[INFO] Writing batch 22 with 1 rows to MongoDB...
[INFO] Writing batch 23 with 1 rows to MongoDB...
[INFO] Writing batch 24 with 1 rows to MongoDB...

Batch: 5
-----
|time|destination_name|search_volume|bookings_volume|sales_volume|
-----|-----|-----|-----|-----|
|20|Singapore|0|1|1985.63|
|20|London|1|0|0.0|
|20|Sydney|1|0|0.0|
|20|Rome|0|1|1530.65|
|20|Rio de Janeiro|0|1|1996.43|
-----

[INFO] Writing batch 5 with 1 rows to MongoDB...
[INFO] Writing batch 25 with 1 rows to MongoDB...
[INFO] Writing batch 26 with 1 rows to MongoDB...

Batch: 6
-----
```

### Ερώτημα 3: Αποθήκευση σε MongoDB

Αφού έχουμε επιτυχώς εγκαταστήσει το Mongo , δημιουργούμε ένα φάκελο που θα αποθηκεύονται τα δεδομένα του (mkdir -p ~/mongo-data) και το εκκινούμε :

```
PS C:\Users\xrist> ubuntu
christiana@Christiana:~$ mongod --dbpath ~/mongo-data
{"t":{"_id":"2025-06-01T17:42:04.457+03:00"},"s":"I","c":"CONTR
```

Η αποθήκευση των δεδομένων στο MongoDB πραγματοποιείται σε δύο στάδια: πρώτα εισάγονται τα ωμά δεδομένα (raw events) όπως λαμβάνονται από το Kafka stream, και έπειτα αποθηκεύονται τα συγκεντρωτικά αποτελέσματα (aggregated stats) που παράγει το Spark.

Η βάση δεδομένων ονομάζεται **clickstream**, και περιλαμβάνει δύο βασικές συλλογές:

- **raw\_events**: αποθηκεύει κάθε μεμονωμένο event από το Kafka stream, μαζί με όλα τα μεταδεδομένα του.

```
59 #-----ερωτημα 3 : Mongo raw data-----
60 def write_raw_to_mongo(batch_df, batch_id):
61     print(f"[INFO] Writing batch {batch_id} with {batch_df.count()} rows to MongoDB...")
62     batch_df.write \
63         .format("mongodb") \
64         .option("uri", "mongodb://localhost:27017") \
65         .option("database", "clickstream") \
66         .option("collection", "raw_events") \
67         .mode("append") \
68         .save()
69
70 raw_query = df.json.writeStream \
71     .foreachBatch(write_raw_to_mongo) \
72     .outputMode("append") \
73     .start()
74
75 #-----
```

- **aggregated\_stats**: αποθηκεύει περιοδικά αποτελέσματα από το Spark, τα οποία προκύπτουν από ομαδοποιήσεις και υπολογισμούς όπως αριθμός αναζητήσεων, κρατήσεων και συνολικές πωλήσεις ανά προορισμό.

```

113 #-----Ερωτημα 3 : Mongo aggregated data-----
114 def write_agg_to_mongo(batch_df, batch_id):
115     print(f"[INFO] Writing batch {batch_id} with {batch_df.count()} rows to MongoDB...")
116     batch_df.write \
117         .format("mongodb") \
118         .option("uri", "mongodb://localhost:27017") \
119         .option("database", "clickstream") \
120         .option("collection", "aggregated_stats") \
121         .mode("append") \
122         .save()
123
124 agg_query = final.writeStream \
125     .foreachBatch(write_agg_to_mongo) \
126     .outputMode("update") \
127     .trigger(processingTime="5 minutes") \
128     .start()
129 #-----

```

Το Spark script, μέσω της μεθόδου `foreachBatch`, στέλνει περιοδικά τα δεδομένα στις αντίστοιχες συλλογές:

- `write_raw_to_mongo`: αποθηκεύει τα ωμά δεδομένα στην `raw_events`.
- `write_agg_to_mongo`: αποθηκεύει τα στατιστικά στην `aggregated_stats`.

Η διαδικασία είναι *real-time*, με κάθε batch να αποστέλλεται όταν ολοκληρώνεται η επεξεργασία, χωρίς απώλειες.

```

[INFO] Writing batch 4 with 2 rows to MongoDB...
[INFO] Writing batch 18 with 1 rows to MongoDB...
[INFO] Writing batch 19 with 1 rows to MongoDB...
[INFO] Writing batch 20 with 2 rows to MongoDB...
[INFO] Writing batch 21 with 1 rows to MongoDB...
[INFO] Writing batch 22 with 1 rows to MongoDB...
[INFO] Writing batch 23 with 1 rows to MongoDB...
[INFO] Writing batch 24 with 1 rows to MongoDB...

```

Για την απάντηση των ερωτημάτων, δημιουργήθηκε ένα ξεχωριστό Python script (`query_mongo.py`) που συνδέεται με τη MongoDB μέσω του `MongoClient`. Τα δεδομένα αντλούνται από τη συλλογή `raw_events` της βάσης `clickstream`, ενώ καθορίζεται και το χρονικό διάστημα ενδιαφέροντος (μεταβλητές `start_date` και `end_date`).

### Ερώτημα 1 – Πόλη με τις περισσότερες κρατήσεις:

Με τη χρήση της εντολής `aggregate()`, το script φιλτράρει μόνο τα γεγονότα `complete_booking` (`event_type`) που έγιναν εντός του χρονικού διαστήματος και έχουν καταχωρημένη πόλη (`location`). Έπειτα, γίνεται ομαδοποίηση με βάση την πόλη και υπολογίζεται το πλήθος κρατήσεων. Τέλος, ταξινομούνται τα αποτελέσματα ώστε να επιστραφεί η πόλη με τις περισσότερες κρατήσεις.

```

query_mongo.py
1  from pymongo import MongoClient
2  from datetime import datetime
3  from bson.son import SON
4  from collections import defaultdict
5
6  #Σύνδεση με MongoDB
7  client = MongoClient("mongodb://localhost:27017/")
8  db = client["clickstream"]
9  collection = db["raw_events"]
10
11 # Ορισμός χρονικού διαστήματος
12 start_date = datetime(2025, 5, 29, 0, 0)
13 end_date = datetime(2025, 6, 1, 23, 59)
14
15
16 # Ερώτημα 1: Πόλη με τις περισσότερες κρατήσεις
17 bookings = collection.aggregate([
18     {"$match": {
19         "event_type": "complete_booking",
20         "event_time": {"$gte": start_date, "$lte": end_date},
21         "location": {"$ne": ""}
22     }},
23     {"$group": {"_id": "$location", "count": {"$sum": 1}}},
24     {"$sort": {"count": -1}},
25     {"$limit": 1}
26 ])
27
28 print("1. Πόλη με τις περισσότερες κρατήσεις:")
29 for doc in bookings:
30     print(f"    {doc['_id']} με {doc['count']} κρατήσεις")
31
32 # -----
33

```

## Ερώτημα 2 - Πόλη με τις περισσότερες αναζητήσεις:

Αντίστοιχα, φιλτράρονται τα γεγονότα search\_hotels και με παρόμοια λογική (χρήση \$match, \$group, \$sort, \$limit), το script εντοπίζει την πόλη που είχε τις περισσότερες αναζητήσεις ξενοδοχείων μέσα στο καθορισμένο χρονικό διάστημα.

```

# Ερώτημα 2: Πόλη με τις περισσότερες αναζητήσεις
✓ searches = collection.aggregate([
✓     {"$match": {
        "event_type": "search_hotels",
        "event_time": {"$gte": start_date, "$lte": end_date},
        "location": {"$ne": ""}
    }},
    {"$group": {"_id": "$location", "count": {"$sum": 1}}},
    {"$sort": {"count": -1}},
    {"$limit": 1}
✓ ])

print("\n2. Πόλη με τις περισσότερες αναζητήσεις:")
✓ for doc in searches:
    print(f"    {doc['_id']} με {doc['count']} αναζητήσεις")

# -----

```

### Ερώτημα 3 - Μέση διάρκεια παραμονής ανά πόλη:

Για τις κρατήσεις (complete\_booking) που περιλαμβάνουν έγκυρες ημερομηνίες check\_in\_date και check\_out\_date, ο κώδικας διατρέχει τα δεδομένα με find() και υπολογίζει τη διαφορά των ημερομηνιών για κάθε κράτηση. Οι τιμές αποθηκεύονται σε λίστα ανά πόλη (durations dictionary) και στο τέλος υπολογίζεται ο μέσος όρος παραμονής για κάθε πόλη.

```
# Ερώτημα 3: Μέση διάρκεια παραμονής ανά πόλη (σε μέρες)
✓ bookings_cursor = collection.find({
    "event_type": "complete_booking",
    "event_time": {"$gte": start_date, "$lte": end_date},
    "location": {"$ne": ""},
    "check_in_date": {"$ne": ""},
    "check_out_date": {"$ne": ""}
})

durations = defaultdict(list)

✓ for doc in bookings_cursor:
✓     try:
        check_in = datetime.strptime(doc["check_in_date"], "%Y-%m-%d")
        check_out = datetime.strptime(doc["check_out_date"], "%Y-%m-%d")
        stay_length = (check_out - check_in).days
        if stay_length > 0:
            durations[doc["location"]].append(stay_length)
✓     except:
        continue

print("\n3. Μέση διάρκεια παραμονής ανά πόλη:")
✓ for city, stays in durations.items():
    avg = sum(stays) / len(stays)
    print(f"    {city}: {avg:.2f} μέρες")
```

Για να δούμε τα αποτελέσματα πρώτα ανοίγουμε το mongo shell , βλέπουμε αρχικά τις βάσεις δεδομένων που έχουμε και παρατηρούμε ότι έχει ήδη δημιουργηθεί η clickstream . Στην συνέχεια βλέπουμε τις συλλογές που δημιουργήθηκαν και παρατηρούμε ότι έχει δημιουργηθεί η raw\_events και η aggregated\_stats :



```

christiana@Christiana:~$ mongosh
Current Mongosh Log ID: 683c7713ab41843597c59f34
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=10000
Using MongoDB:      7.0.20
Using Mongosh:      2.5.1

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

-----
  The server generated these startup warnings when booting
  2025-06-01T17:42:04.496+03:00: Using the XFS filesystem is strongly recommended on the platform in this mode
  2025-06-01T17:42:08.479+03:00: Access control is not enabled for the database. Remote systems can
  2025-06-01T17:42:08.479+03:00: This server is bound to localhost. Remote systems can connect to
  2025-06-01T17:42:08.479+03:00: Soft rlimits for open file descriptors too low: rlimit:
  2025-06-01T17:42:08.480+03:00: For customers running MongoDB 7.0, we suggest changing the
  -----

test> show dbs
admin          40.00 KiB
appdb          40.00 KiB
clickstream    152.00 KiB
config         72.00 KiB
local          72.00 KiB
test> use clickstream
switched to db clickstream
clickstream> show collections
aggregated_stats
raw_events
clickstream>

```

Με την εντολή `db.raw_events.find().pretty()` εκτυπώνουμε το περιεχόμενο της `raw_events` συλλογής :

```

christiana@Christiana:~$ mongosh
Current Mongosh Log ID: 683c7713ab41843597c59f34
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=10000
Using MongoDB:      7.0.20
Using Mongosh:      2.5.1

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

-----
  The server generated these startup warnings when booting
  2025-06-01T17:42:04.496+03:00: Using the XFS filesystem is strongly recommended on the platform in this mode
  2025-06-01T17:42:08.479+03:00: Access control is not enabled for the database. Remote systems can
  2025-06-01T17:42:08.479+03:00: This server is bound to localhost. Remote systems can connect to
  2025-06-01T17:42:08.479+03:00: Soft rlimits for open file descriptors too low: rlimit:
  2025-06-01T17:42:08.480+03:00: For customers running MongoDB 7.0, we suggest changing the
  -----

test> use clickstream
switched to db clickstream
clickstream> show collections
aggregated_stats
raw_events
clickstream> db.raw_events.find().pretty()
{
  "_id" : ObjectId("683c7713ab41843597c59f34"),
  "user_id" : "user_00000",
  "session_id" : "session_00000_00000_00000",
  "timestamp" : "2025-06-01T17:40:36.553Z",
  "event_type" : "login",
  "booking_id" : "",
  "check_in_date" : "",
  "check_out_date" : "",
  "hotel_id" : "",
  "item_id" : "",
  "location" : "",
  "mkt_guests" : {
    "page_url" : "https://hotelbooking.example.com/login",
    "payment_method" : "",
    "price" : null,
    "room_type" : "",
    "total_price" : null,
    "event_time" : ISODate("2025-06-01T17:40:36.553Z")
  }
},
{
  "_id" : ObjectId("683c7713ab41843597c59f34"),
  "user_id" : "user_00000",
  "session_id" : "session_00000_00000_00000",
  "timestamp" : "2025-06-01T17:40:27.553Z",
  "event_type" : "room_view",
  "booking_id" : "",
  "check_in_date" : "",
  "check_out_date" : "",
  "hotel_id" : "",
  "item_id" : "",
  "location" : "",
  "mkt_guests" : {
    "page_url" : "https://hotelbooking.example.com/room",
    "payment_method" : "",
    "price" : null,
    "room_type" : "",
    "total_price" : null,
    "event_time" : ISODate("2025-06-01T17:40:27.553Z")
  }
},
{
  "_id" : ObjectId("683c7713ab41843597c59f34"),
  "user_id" : "user_00000",
  "session_id" : "session_00000_00000_00000",
  "timestamp" : "2025-06-01T17:40:19.553Z",
  "event_type" : "room_checkout",
  "booking_id" : "",
  "check_in_date" : "",
  "check_out_date" : "",
  "hotel_id" : "",
  "item_id" : "",
  "location" : "",
  "mkt_guests" : {
    "page_url" : "https://hotelbooking.example.com/checkout",
    "payment_method" : "",
    "price" : null,
    "room_type" : "",
    "total_price" : null,
    "event_time" : ISODate("2025-06-01T17:40:19.553Z")
  }
},
{
  "_id" : ObjectId("683c7713ab41843597c59f34")
}

```

Αντίστοιχα και για την `aggregated_stats` συλλογή :



## Σχολιασμός αποτελεσμάτων

Μέσα από την εκπόνηση του project είχα την ευκαιρία να εφαρμόσω στην πράξη τεχνολογίες όπως το Kafka, το Apache Spark και τη MongoDB. Αν και στην αρχή αντιμετώπισα δυσκολίες στην κατανόηση ορισμένων εννοιών, κυρίως στη σύνδεση μεταξύ των εργαλείων, η διαδικασία με βοήθησε να αποκτήσω καλύτερη εικόνα για το πώς γίνεται η ανάλυση και η αποθήκευση δεδομένων σε πραγματικά σενάρια. Επιπλέον, είχα για πρώτη φορά την ευκαιρία να δουλέψω με τη MongoDB και να εξασκηθώ στη χρήση της μέσω Python για επεξεργασία και εκτέλεση ερωτημάτων. Συνολικά, το project συνέβαλε σημαντικά στην ενίσχυση των γνώσεών μου και με βοήθησε να αισθάνομαι πιο άνετα με αυτά τα εργαλεία.

## Βιβλιογραφία

Χρησιμοποιήθηκαν όλες οι πηγές που παρέχονταν στην εκφώνηση της άσκησης και επιπλέον :

- [MongoDB Crash Course](#)
- [Structured Streaming Programming Guide - Spark 4.0.0 Documentation](#)
- [Introduction to Structured Streaming in Apache Spark \(PySpark\) + Kafka | by Dinesh Kumar A S | Medium](#)