FitRec Workout Analysis Using MongoDB and PySpark

MUNAL BARAILI

900006725

Introduction

This project focuses on designing and testing a simple non-relational data system using MongoDB and PySpark. The dataset used for this assignment originates from a much larger collection of recorded fitness activities. Because full dataset is approximately 5GB too large for storage limits of the free tier of MongoDB Atlas, a cloud-hosted service provided by MongoDB Inc. (MongoDB Inc., n.d.) I worked with a smaller representative sample containing 1,710 workout sessions and around 855,000 individual trackpoints, roughly 1% of the full data. This subset was sufficient to demonstrate required database design, querying, and PySpark integration steps.

Each workout record in the subset includes metadata such as user ID, gender, and sport type, together with sequential measurements like GPS coordinates, altitude, timestamps, and heart rate. This structure follows the format documented in FitRec dataset (Ni, Muhlstein & McAuley, 2019). In this project, data was reorganised into a cleaner database structure and imported into MongoDB, where analytical queries were performed. The processed collections were then integrated with PySpark to enable further exploration and verification. This workflow helped clarify both the characteristics of dataset and practical process of moving data between storage and computation layers.

Database Design

At the start, I examined the structure of the dataset closely. Each workout included long arrays of trackpoints typically 500 points per workout making it clear that embedding all this information inside a single MongoDB document would lead to very large documents that would be inefficient to query. Because of this, I chose to separate the data into two collections.

The first collection, *workouts*, stores the high-level metadata for each session, such as the sport, gender, and workout ID. The second collection, *trackpoints*, stores the time-series measurements for each workout, with each sensor reading recorded as its own document. This structure forms a simple logical relationship where each workout is the parent and its trackpoints are the children. I also created a third collection called *summary*, which holds aggregated statistics such as the number of workouts per sport and average heart rate by sport.

MongoDB was the preferred choice for this project because it naturally handles semi-structured data and allows each workout to have an arbitrary number of trackpoints. A relational schema with strict table structures would have required multiple joins and would not have been as flexible. To make the database more efficient, I added indexes on the workout ID, user ID, and on the trackpoints' workout ID paired with sequence number. This improved the speed of queries that needed to retrieve time-ordered data.

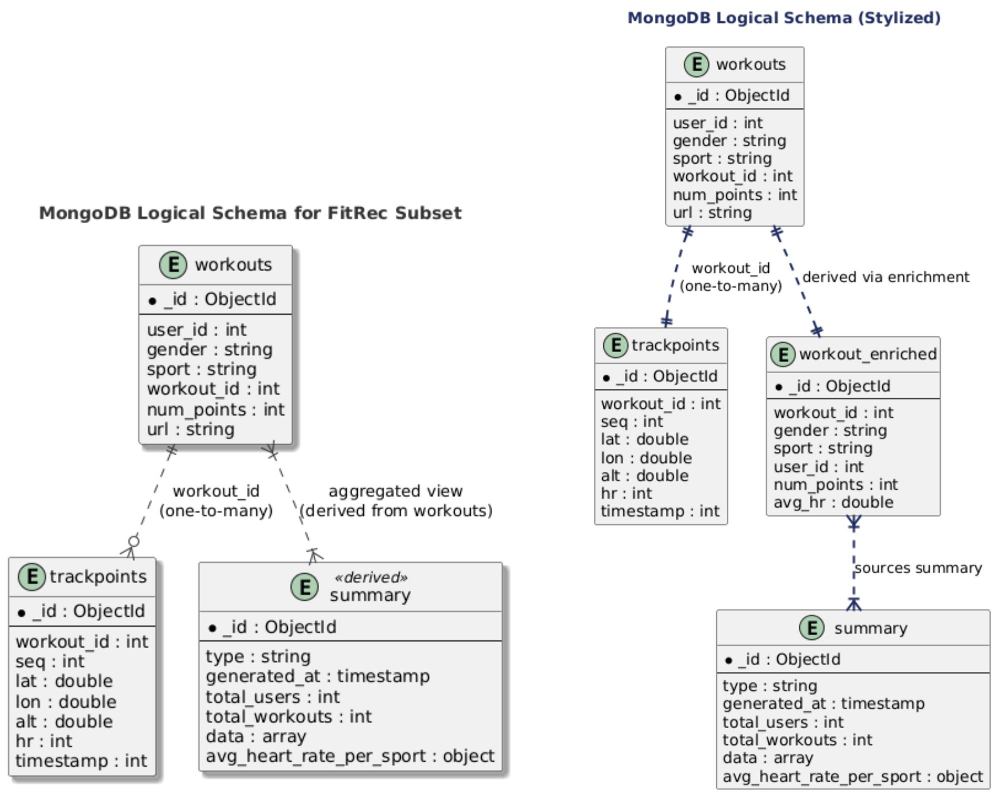
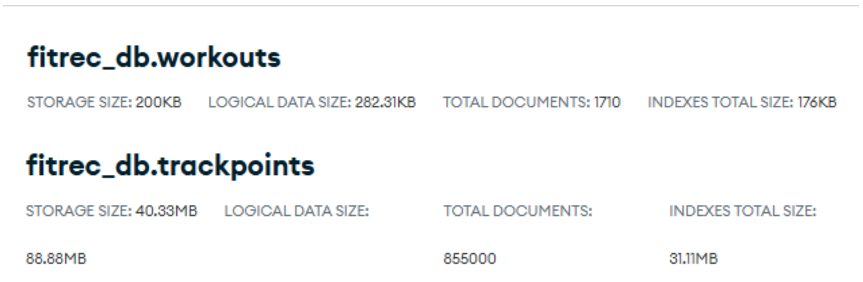


Fig 1. Beginning Modelling Schema (left), After spark session update schema (right)

Data Preparation and Population

Before uploading the data into MongoDB, I applied light cleaning to ensure that the structure was consistent. After confirming fields and verifying shape of the sample, I inserted all 1710 workout records into *workouts* collection and then inserted all 855,000 trackpoints using batch operations. The batching process helped avoid performance issues and made the import more efficient.



fitrec_db.workouts

STORAGE SIZE: 200KB LOGICAL DATA SIZE: 282.31KB TOTAL DOCUMENTS: 1710 INDEXES TOTAL SIZE: 176KB

fitrec_db.trackpoints

STORAGE SIZE: 40.33MB LOGICAL DATA SIZE: 88.88MB TOTAL DOCUMENTS: 855000 INDEXES TOTAL SIZE: 31.11MB

Fig 2 Screenshot of MongoDB Atlas for workouts and trackpoints

Once the data was inserted, I created a summary document that stores results of common queries such as the distribution of sports, total number of users, and average heart rate values. This helped demonstrate how derived or analytical information can be kept in the database for quick reference.

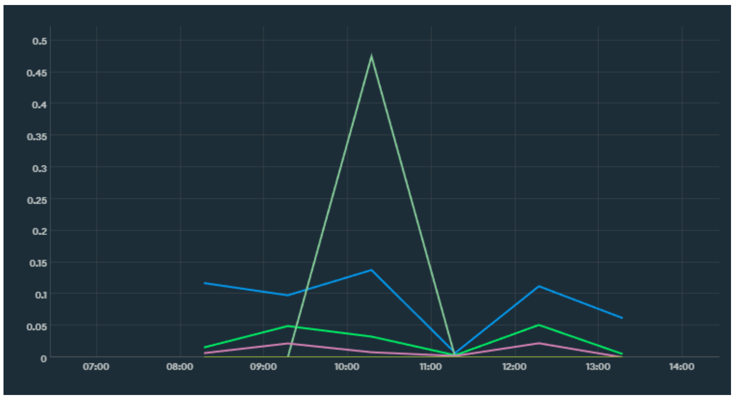


Fig 3 Cluster Metrics showing different activities (query, write, command inside the cluster).

MongoDB Query Demonstration

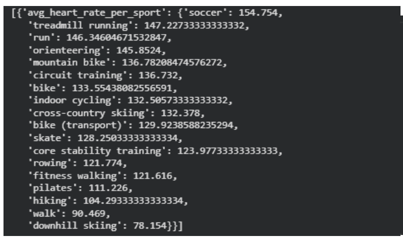
Once the data was loaded into MongoDB, a set of queries was carried out to explore the structure of the collections and to verify that both basic retrievals and more advanced analytical operations worked as intended. The process began with simple filtering tasks. For example, workouts belonging to female users were retrieved using a projection that displayed only the sport and user ID fields. This made it easy to get an initial sense of what activities were represented within that subset of users.

After confirming that basic filters were working properly, aggregation queries were used to explore the dataset at a broader level. One of these queries grouped the workouts by sport to calculate how many sessions belonged to each category.

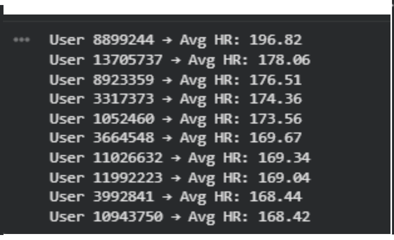
This formed the basis for small "sport distribution" summary that was later stored in its own collection. A new document containing this distribution, along with a timestamp, was inserted into the *summary* collection.

This was later updated to include additional fields such as the total number of workouts and the number of unique users in the database. These insert and update steps demonstrated that the database was functioning correctly beyond simple read operations.

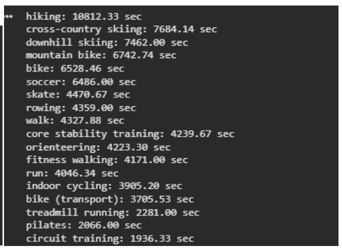
More detailed analysis involved combining the *workouts* and *trackpoints* collections. Using MongoDB's $lookup operator, the time-series heart-rate data stored in *trackpoints* was joined with the corresponding workout metadata. This allowed the calculation of the average heart rate for each sport by grouping the joined documents. The results showed clear differences between activities; for example, sports such as soccer tended to produce much higher average heart rates, while activities like downhill skiing appeared on the lower end. This query demonstrated how MongoDB's aggregation framework can be used to compute meaningful statistics across collections without changing the underlying schema.



```
[{'avg_heart_rate_per_sport': {'soccer': 154.754,
   'treadmill running': 147.22733333333332,
   'run': 146.34604671532847,
   'orienteering': 145.8524,
   'mountain bike': 136.78208474576272,
   'circuit training': 136.732,
   'bike': 133.55438082556591,
   'indoor cycling': 132.50573333333332,
   'cross-country skiing': 132.378,
   'bike (transport)': 129.9238588235294,
   'skate': 128.25033333333334,
   'core stability training': 123.97733333333333,
   'rowing': 121.774,
   'fitness walking': 121.616,
   'pilates': 111.226,
   'hiking': 104.29333333333334,
   'walk': 90.469,
   'downhill skiing': 78.154}}]
```



```
User 8899244 → Avg HR: 196.82
User 13705737 → Avg HR: 178.06
User 8923359 → Avg HR: 176.51
User 3317373 → Avg HR: 174.36
User 1052460 → Avg HR: 173.56
User 3664548 → Avg HR: 169.67
User 11026632 → Avg HR: 169.34
User 11992223 → Avg HR: 169.04
User 3992841 → Avg HR: 168.44
User 10943750 → Avg HR: 168.42
```

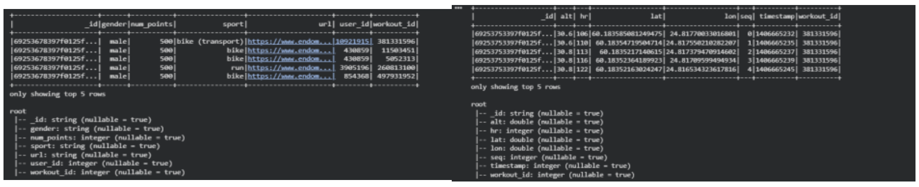


```
hiking: 10812.33 sec
cross-country skiing: 7684.14 sec
downhill skiing: 7462.00 sec
mountain bike: 6742.74 sec
bike: 6528.46 sec
soccer: 6486.00 sec
skate: 4470.67 sec
rowing: 4359.00 sec
walk: 4327.88 sec
core stability training: 4239.67 sec
orienteering: 4223.30 sec
fitness walking: 4171.00 sec
run: 4046.34 sec
indoor cycling: 3905.20 sec
bike (transport): 3705.53 sec
treadmill running: 2281.00 sec
pilates: 2066.00 sec
circuit training: 1936.33 sec
```

Throughout these steps, additional queries were used to verify that the summary document had been updated correctly, to confirm the number of unique users, and to check that the new fields were reflected in subsequent reads.

PySpark Integration

Integration with PySpark formed the second major part of the project. After configuring the MongoDB Spark Connector (version 10.x), the *workouts* and *trackpoints* collections were loaded directly into Spark DataFrames. The connector recognised the fields correctly, and both schemas appeared as expected when inspected in the notebook. Spark handled the trackpoints collection which contained more than 850,000 sensor readings efficiently, confirming that the dataset size was well suited for a distributed processing environment.



```
+--------------------+------+----------+----------------+--------------------+--------+----------+
|                 _id|gender|num_points|           sport|                 url| user_id|workout_id|
+--------------------+------+----------+----------------+--------------------+--------+----------+
|69253678397f0125f...|  male|       500|bike (transport)|https://www.endom...|10921915| 381331596|
|69253678397f0125f...|  male|       500|            bike|https://www.endom...|  430859|  11503451|
|69253678397f0125f...|  male|       500|            bike|https://www.endom...|  430859|   5052313|
|69253678397f0125f...|  male|       500|             run|https://www.endom...| 3905196| 260813100|
|69253678397f0125f...|  male|       500|            bike|https://www.endom...|  854368| 497931952|
+--------------------+------+----------+----------------+--------------------+--------+----------+
only showing top 5 rows

root
 |-- _id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- num_points: integer (nullable = true)
 |-- sport: string (nullable = true)
 |-- url: string (nullable = true)
 |-- user_id: integer (nullable = true)
 |-- workout_id: integer (nullable = true)
```

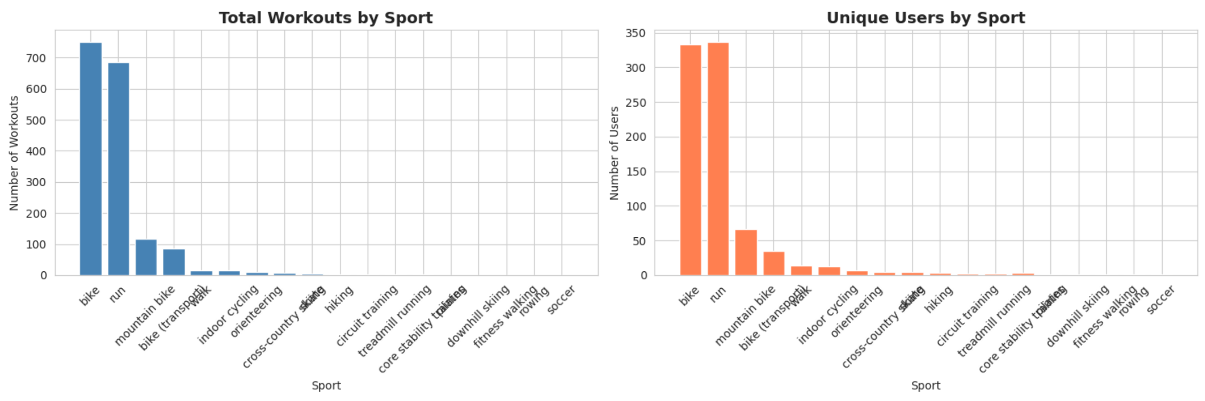
```
+--------------------+----+---+-----------------+------------------+---+----------+----------+
|                 _id| alt| hr|              lat|               lon|seq| timestamp|workout_id|
+--------------------+----+---+-----------------+------------------+---+----------+----------+
|69253753397f0125f...|30.6|106|60.183585081249475| 24.81770033016801|  0|1406665232| 381331596|
|69253753397f0125f...|30.6|110| 60.18354719504714|24.817550210282207|  1|1406665235| 381331596|
|69253753397f0125f...|30.8|113|  60.1835217140615|24.817379470914602|  2|1406665237| 381331596|
|69253753397f0125f...|30.8|116| 60.18352364189923| 24.81709599494934|  3|1406665239| 381331596|
|69253753397f0125f...|30.8|122| 60.18352163024247|24.816534323617816|  4|1406665245| 381331596|
+--------------------+----+---+-----------------+------------------+---+----------+----------+
only showing top 5 rows

root
 |-- _id: string (nullable = true)
 |-- alt: double (nullable = true)
 |-- hr: integer (nullable = true)
 |-- lat: double (nullable = true)
 |-- lon: double (nullable = true)
 |-- seq: integer (nullable = true)
 |-- timestamp: integer (nullable = true)
 |-- workout_id: integer (nullable = true)
```

Fig 4 Spark Session Workout_df printSchema and trackpoints_df

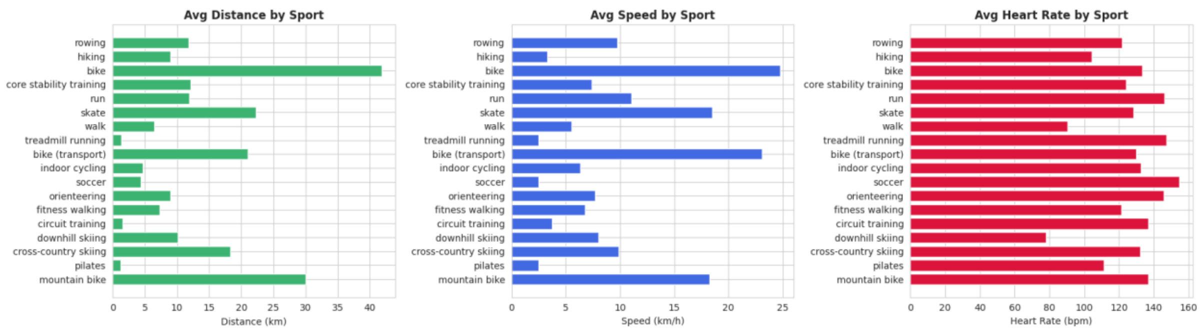
workouts_df.count(), trackpoints_df.count() = (1710, 855000)

The first set of queries in PySpark involved recreating some of the checks that were initially done in MongoDB. By grouping workouts by sport, Spark produced the same distribution seen earlier, with biking and running appearing as the most frequent activities.

Additional statistics, such as the number of unique users in each sport and the average number of trackpoints per session, were also computed. These early results provided a general overview of the dataset and helped confirm that the data had been loaded correctly.

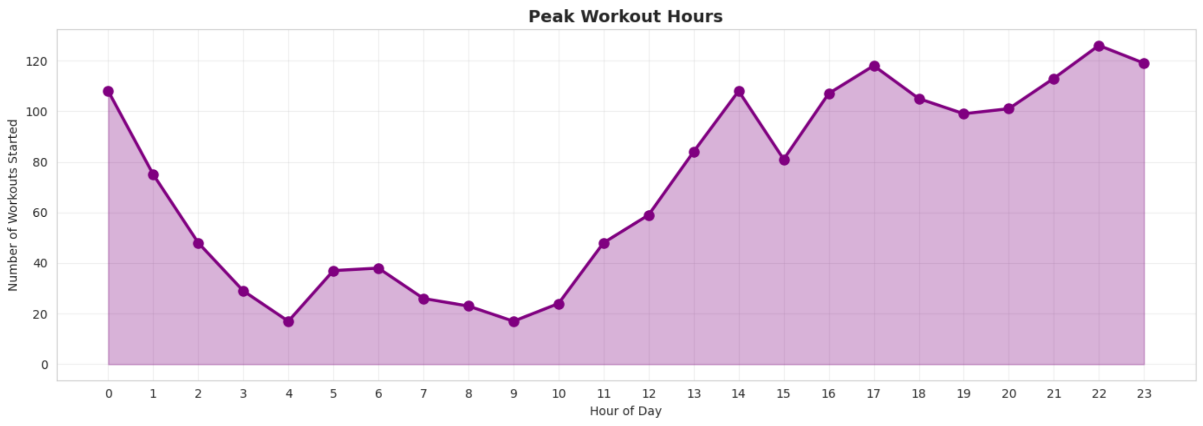


To support more detailed analysis, the trackpoints DataFrame was enriched with calculated fields. These included the distance between each pair of GPS coordinates, changes in elevation, time differences, estimated speed in km/h, and human-readable timestamps converted from UNIX time. A sample of the enriched DataFrame is shown in Figure 5. These additional features allowed for a more complete understanding of each workout's characteristics.

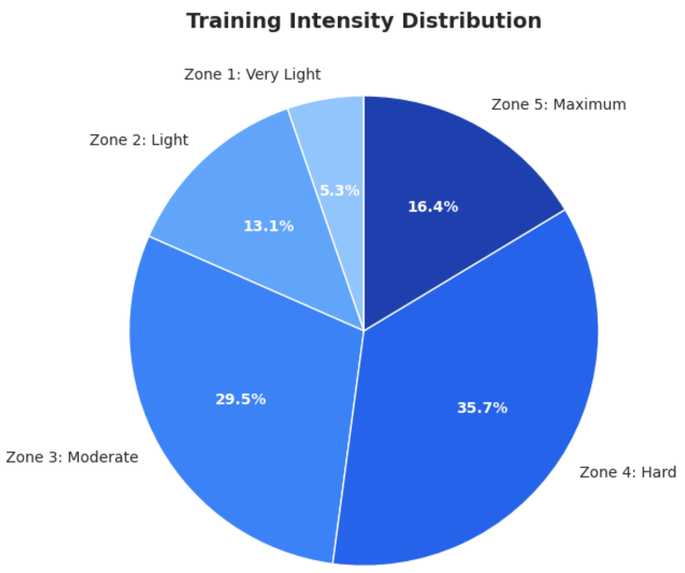


Using these enriched measurements, a summary table was generated at the workout level. For each workout, Spark calculated values such as total distance, average and maximum speed, average and maximum heart rate, elevation gain, duration, and average pace. When this summary was joined back with the original workout metadata, it became possible to explore differences across sports, such as which activities tended to be longer or more intense. These comparisons helped clarify how different exercise types behaved in the sample dataset.

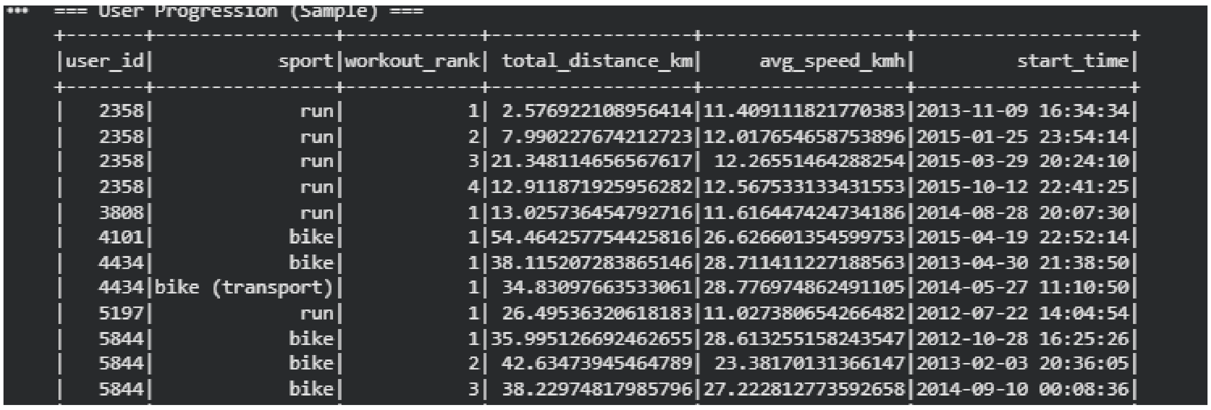
Temporal patterns were also examined by extracting the hour of day from the workout start time. This revealed a strong concentration of activity in the evening, particularly between 5 PM and 10 PM.



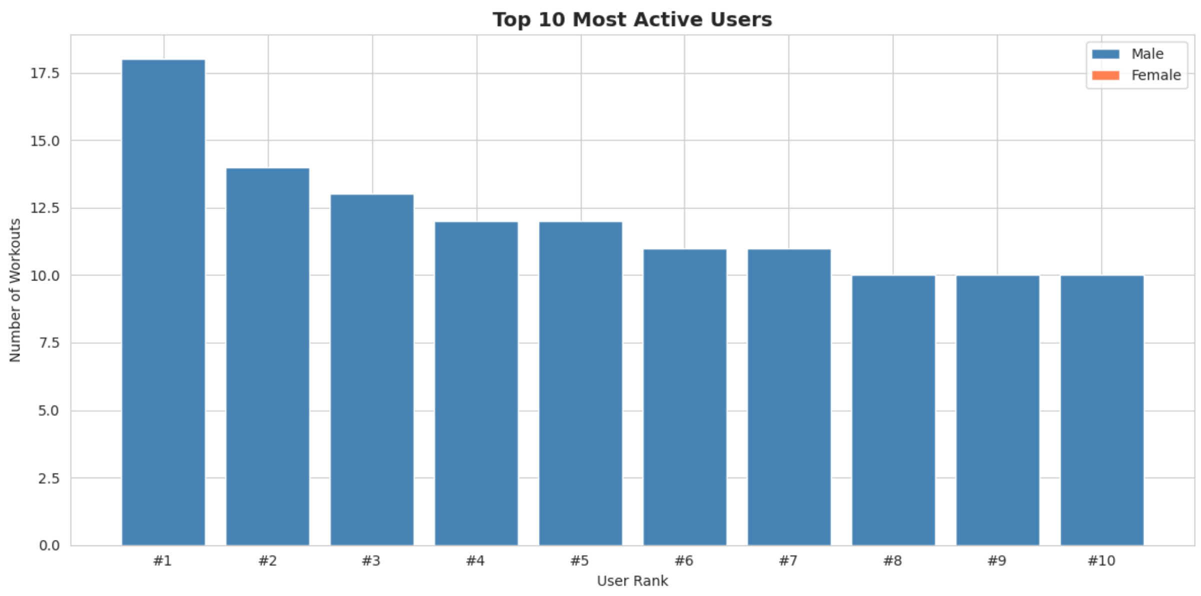
To investigate workout intensity, each heart-rate measurement was assigned to a zone ranging from “very light” to “maximum.” Counting how many trackpoints fell into each zone provided a clearer view of how strenuous the workouts were overall.



Finally, a simple progression analysis was performed by ordering each user’s workouts chronologically and comparing their distances and speeds over time. Although the dataset was limited, this approach helped illustrate how PySpark can be used to track behavioural trends across multiple sessions.



```
=== User Progression (Sample) ===
+-------+----------------+------------+------------------+------------------+-------------------+
|user_id|           sport|workout_rank| total_distance_km|     avg_speed_kmh|         start_time|
+-------+----------------+------------+------------------+------------------+-------------------+
|   2358|             run|           1| 2.576922108956414|11.409111821770383|2013-11-09 16:34:34|
|   2358|             run|           2| 7.990227674212723|12.017654658753896|2015-01-25 23:54:14|
|   2358|             run|           3|21.348114656567617| 12.26551464288254|2015-03-29 20:24:10|
|   2358|             run|           4|12.911871925956282|12.567533133431553|2015-10-12 22:41:25|
|   3808|             run|           1|13.025736454792716|11.616447424734186|2014-08-28 20:07:30|
|   4101|            bike|           1|54.464257754425816|26.626601354599753|2015-04-19 22:52:14|
|   4434|            bike|           1|38.115207283865146|28.711411227188563|2013-04-30 21:38:50|
|   4434|bike (transport)|           1| 34.83097663533061|28.776974862491105|2014-05-27 11:10:50|
|   5197|             run|           1| 26.49536320618183|11.027380654266482|2012-07-22 14:04:54|
|   5844|            bike|           1|35.995126692462655|28.613255158243547|2012-10-28 16:25:26|
|   5844|            bike|           2| 42.63473945464789| 23.38170131366147|2013-02-03 20:36:05|
|   5844|            bike|           3| 38.22974817985796|27.222812773592658|2014-09-10 00:08:36|
```

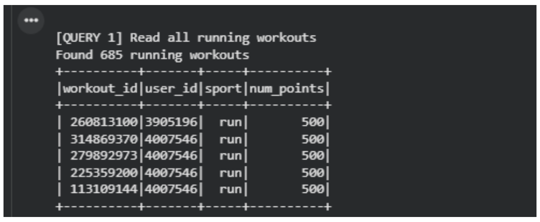


Top 10 Most Active Users
Male
Female
Number of Workouts
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
#1
#2
#3
#4
#5
#6
#7
#8
#9
#10
User Rank

Python Query Execution Through PySpark

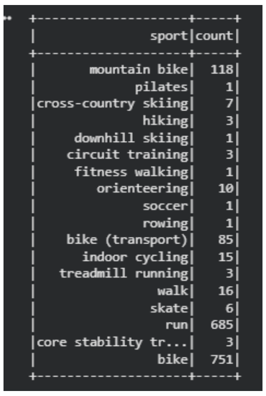
To demonstrate Python-level querying using PySpark, I executed a series of read and write operations against the MongoDB collections. These queries served mainly to verify that Spark could retrieve targeted subsets of data, perform aggregations, and write processed results back into the database.

The first query filtered the workouts collection to return all sessions labelled as *running*. Spark identified 685 running workouts, and a five-row preview.



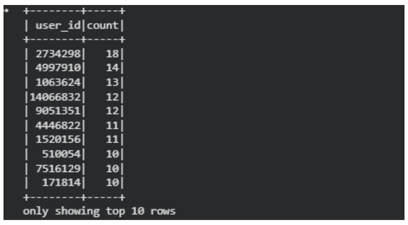
```
[QUERY 1] Read all running workouts
Found 685 running workouts
+----------+-------+-----+----------+
|workout_id|user_id|sport|num_points|
+----------+-------+-----+----------+
| 260813100|3905196|  run|       500|
| 314869370|4007546|  run|       500|
| 279892973|4007546|  run|       500|
| 225359200|4007546|  run|       500|
| 113109144|4007546|  run|       500|
+----------+-------+-----+----------+
```

Similar filtering and grouping queries were used to examine the distribution of sports, confirming that *bike* and *run* were the most frequent activities in this subset of the dataset.



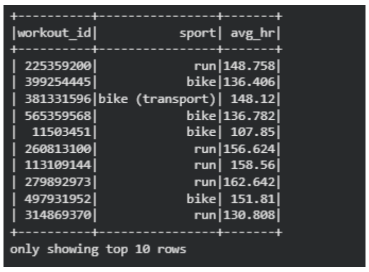
```
+--------------------+-----+
|               sport|count|
+--------------------+-----+
|        mountain bike|  118|
|             pilates|    1|
|cross-country skiing|    7|
|              hiking|    3|
|     downhill skiing|    1|
|    circuit training|    3|
|     fitness walking|    1|
|         orienteering|   10|
|              soccer|    1|
|              rowing|    1|
|    bike (transport)|   85|
|       indoor cycling|   15|
|    treadmill running|    3|
|                walk|   16|
|               skate|    6|
|                 run|  685|
|core stability tr...|    3|
|                bike|  751|
+--------------------+-----+
```

I then grouped the workouts by user ID to identify the most active individuals. Spark reported that the top user had 18 workouts, followed by several users with between 10 and 14 workouts. This simple aggregation helped verify that the underlying collection loaded correctly and matched earlier database results.



```
+--------+-----+
| user_id|count|
+--------+-----+
| 2734298|   18|
| 4997910|   14|
| 1063624|   13|
|14066832|   12|
| 9051351|   12|
| 4446822|   11|
| 1520156|   11|
|  510054|   10|
| 7516129|   10|
|  171814|   10|
+--------+-----+
only showing top 10 rows
```

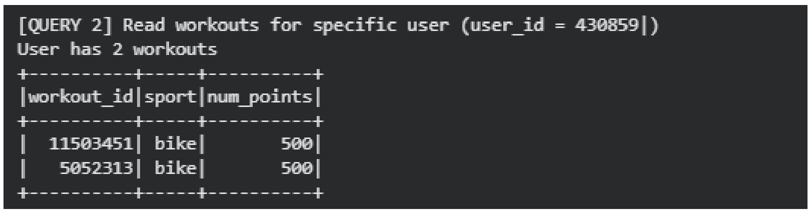
To incorporate physiological data, I computed the average heart rate for each workout by grouping the *trackpoints* collection by workout ID. These averages were then joined with the workout metadata, producing a combined view of sport type and corresponding intensity.



```
+----------+-----------------+-------+
|workout_id|            sport| avg_hr|
+----------+-----------------+-------+
| 225359200|              run|148.758|
| 399254445|             bike|136.406|
| 381331596|bike (transport)| 148.12|
| 565359568|             bike|136.782|
|  11503451|             bike| 107.85|
| 260813100|              run|156.624|
| 113109144|              run| 158.56|
| 279892973|              run|162.642|
| 497931952|             bike| 151.81|
| 314869370|              run|130.808|
+----------+-----------------+-------+
only showing top 10 rows
```

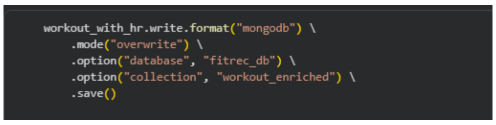
Activities such as soccer, treadmill running, and running showed some of the highest average values, while walking and skiing appeared among the lowest.

Trackpoint-level queries were also executed. For example, retrieving all points for a single workout returned 500 sequential GPS and heart-rate entries, confirming that ordering and schema alignment were preserved in Spark.



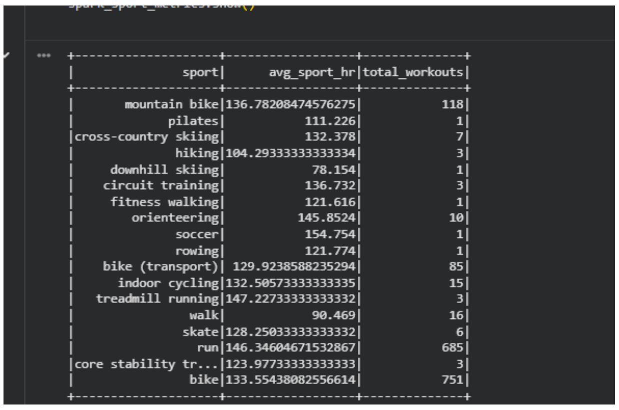
```
[QUERY 2] Read workouts for specific user (user_id = 430859|)
User has 2 workouts
+----------+-----+----------+
|workout_id|sport|num_points|
+----------+-----+----------+
|  11503451| bike|       500|
|   5052313| bike|       500|
+----------+-----+----------+
```

To confirm write capabilities, the enriched workout dataset containing metadata along with computed average heart rate was saved back into MongoDB as a new collection.



```
workout_with_hr.write.format("mongodb") \
    .mode("overwrite") \
    .option("database", "fitrec_db") \
    .option("collection", "workout_enriched") \
    .save()
```

A final group-by operation in Spark then calculated sport-level summary statistics (e.g., average heart rate and workout counts), producing results consistent with earlier patterns observed in both Spark and MongoDB.



```
+-------------------+----------------+--------------+
|              sport|    avg_sport_hr|total_workouts|
+-------------------+----------------+--------------+
|      mountain bike|136.78208474576275|         118|
|            pilates|         111.226|            1|
|cross-country skiing|         132.378|            7|
|             hiking|104.29333333333334|           3|
|     downhill skiing|          78.154|            1|
|    circuit training|         136.732|            3|
|     fitness walking|         121.616|            1|
|        orienteering|        145.8524|           10|
|              soccer|         154.754|            1|
|              rowing|         121.774|            1|
|    bike (transport)| 129.9238588235294|          85|
|      indoor cycling|132.50573333333335|          15|
|   treadmill running|147.22733333333332|           3|
|                walk|          90.469|           16|
|               skate|128.25033333333332|            6|
|                 run|146.34604671532867|          685|
|core stability tr...|123.97733333333333|            3|
|                bike|133.55438082556614|          751|
+-------------------+----------------+--------------+
```

Together, these queries demonstrated that PySpark could reliably read from and write to MongoDB, perform filtering and aggregation at scale, and support the analytical tasks needed for this project. The full code implementations appear in the accompanying notebook, while the report highlights only the most representative examples.

Results and Discussion

Across both MongoDB and PySpark, the dataset showed consistent patterns. Cycling and running were the most common activities, with running generally producing higher heart rates. Enriched features revealed additional behaviours such as a preference for evening workouts and the presence of high-intensity sessions in sports like running and orienteering. While the data quality was mostly stable, some anomalies such as abrupt GPS jumps suggest opportunities for further cleaning. The chosen architecture performed well: MongoDB handled semi-structured data efficiently, and PySpark processed the large trackpoint dataset without issues.

Conclusion

This project demonstrated complete workflow of designing, loading, and analysing a non-relational fitness dataset using MongoDB and PySpark. Splitting the data into workouts and trackpoints collections worked effectively for time-series processing, while PySpark enabled richer metric calculations and exploration of behavioural patterns. Although a few sensor anomalies were present, the dataset remained reliable for analysis. The system performed well for project scale, and future improvements could include additional cleaning steps, more derived fields, or visual dashboards. Overall, the project provided practical experience in integrating database systems with distributed analytical tools in a real-world context.