Contents

1 TDT4225 - Report 1
1.1 Assignment 3 - MongoDB 1
1.1.1 Introduction 1
1.1.2 Results 1
1.1.3 Discussion 7

1 TDT4225 - Report

1.1 Assignment 3 - MongoDB

- **Group**: 89
- **Students**: Deidier Simone

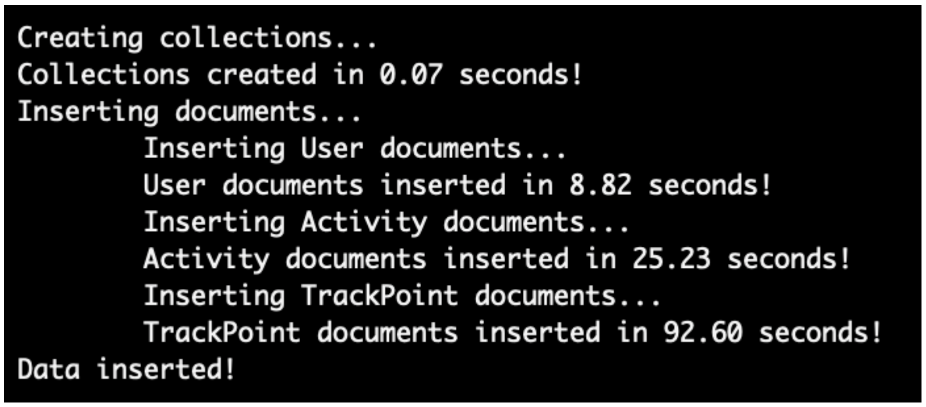
1.1.1 Introduction

This project centers on analyzing the **Geolife GPS Trajectory dataset** using *MongoDB* and *Python.* Using the same dataset as a previous project with *MySQL*, this analysis explores differences in database design, query construction, and performance when managing **GPS trajectory data** with a NoSQL database. Like before, the dataset tracks the outdoor movements of **182 users**, and we set up a MongoDB database with collections for **users**, **activities**, and **trackpoints**. Data cleaning and filtering were conducted to ensure only activities with fewer than **2,500 trackpoints** were included. While implementing equivalent queries, including those related to user activities and transportation modes, this project highlights key differences between *SQL* and *MongoDB*. We examine how NoSQL's flexible document structure and aggregation framework affect query design and data handling, assessing MongoDB's strengths and limitations in managing large, nested datasets and performing complex query analyses.

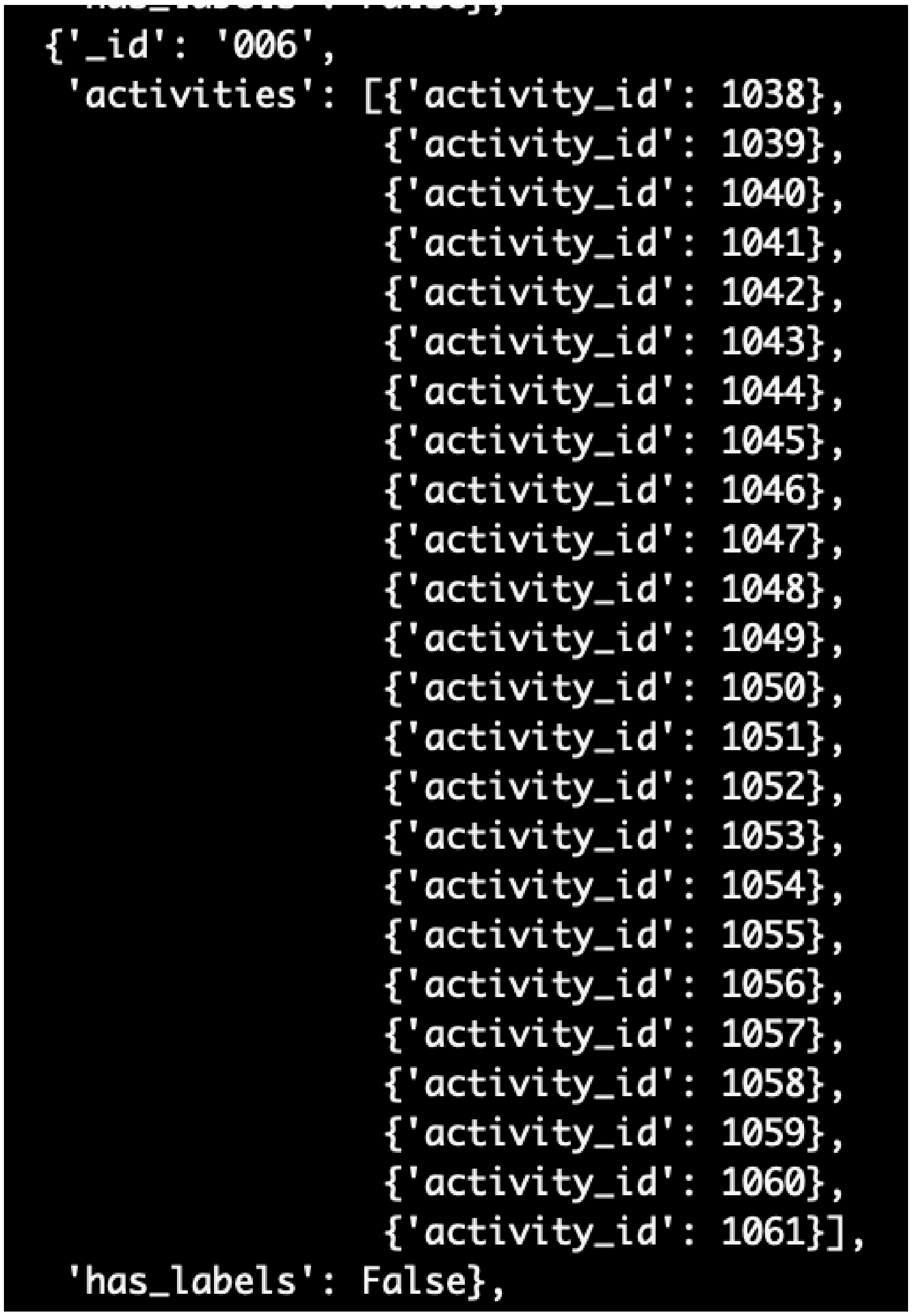
1.1.2 Results

- **DATA INSERTION**

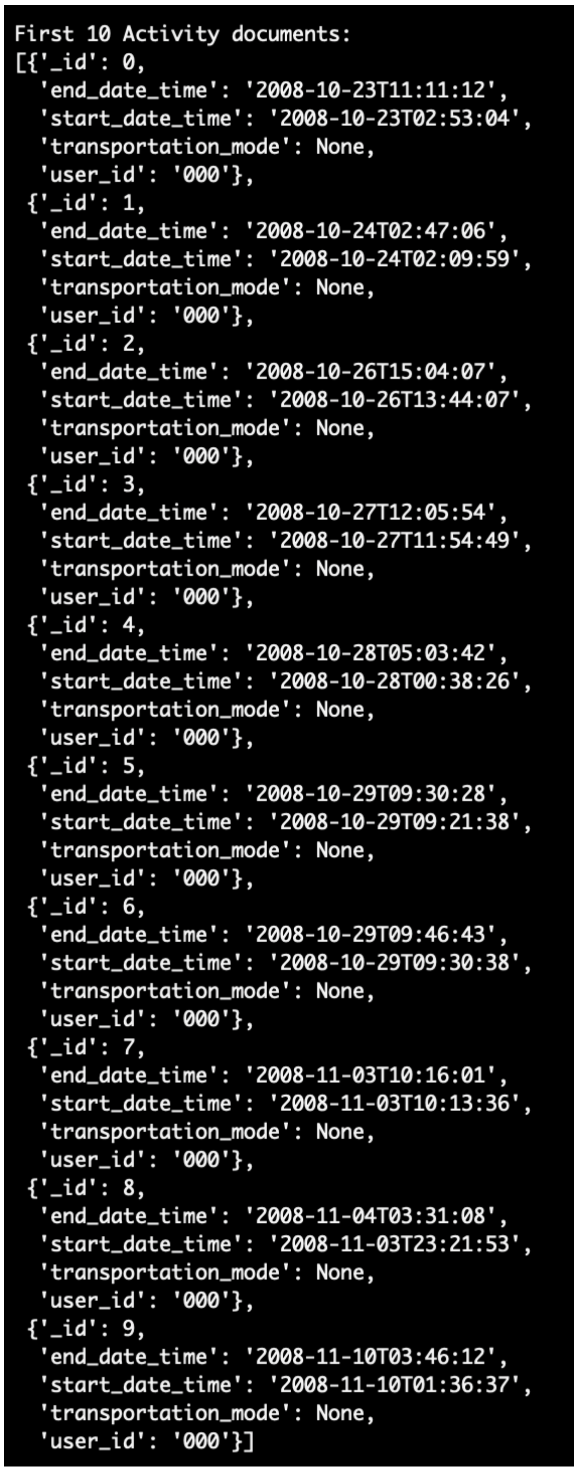
As we can see from the results, even without specific optimizations, *MongoDB* demonstrates significantly faster data insertion compared to *MySQL*. In particular, the time required to insert documents into the **TrackPoint collection**, which contains over **nine million data points**, is approximately half the time taken with *MySQL*. This performance boost is due to MongoDB's document-oriented structure, which allows it to handle large data volumes more efficiently. While in the SQL-based approach, a *batch entry* strategy was necessary to reduce insertion time, MongoDB achieves faster insertion without requiring additional techniques, reflecting its advantages in handling high data throughput (*for the user screenshot, I captured only one user's data as the output format occupies significant space in the terminal; running the code displays the data for all ten users, but here only one is shown for clarity*).



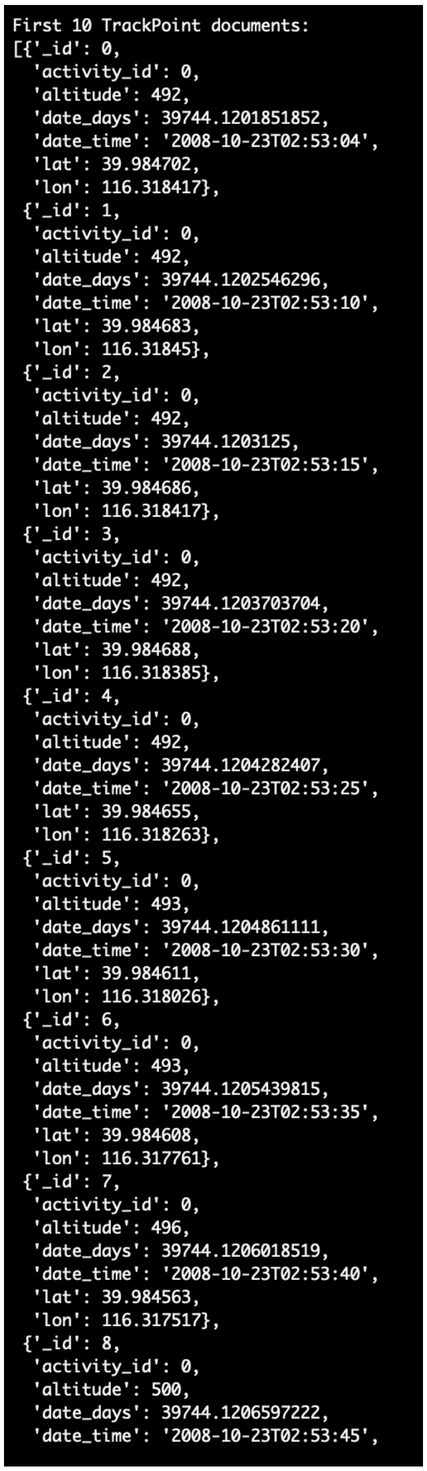
```
Creating collections...
Collections created in 0.07 seconds!
Inserting documents...
        Inserting User documents...
        User documents inserted in 8.82 seconds!
        Inserting Activity documents...
        Activity documents inserted in 25.23 seconds!
        Inserting TrackPoint documents...
        TrackPoint documents inserted in 92.60 seconds!
Data inserted!
```



```
{'_id': '006',
 'activities': [{'activity_id': 1038},
                {'activity_id': 1039},
                {'activity_id': 1040},
                {'activity_id': 1041},
                {'activity_id': 1042},
                {'activity_id': 1043},
                {'activity_id': 1044},
                {'activity_id': 1045},
                {'activity_id': 1046},
                {'activity_id': 1047},
                {'activity_id': 1048},
                {'activity_id': 1049},
                {'activity_id': 1050},
                {'activity_id': 1051},
                {'activity_id': 1052},
                {'activity_id': 1053},
                {'activity_id': 1054},
                {'activity_id': 1055},
                {'activity_id': 1056},
                {'activity_id': 1057},
                {'activity_id': 1058},
                {'activity_id': 1059},
                {'activity_id': 1060},
                {'activity_id': 1061}],
 'has_labels': False},
```



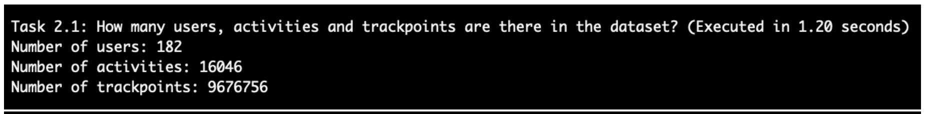
```
First 10 Activity documents:
[{'_id': 0,
  'end_date_time': '2008-10-23T11:11:12',
  'start_date_time': '2008-10-23T02:53:04',
  'transportation_mode': None,
  'user_id': '000'},
 {'_id': 1,
  'end_date_time': '2008-10-24T02:47:06',
  'start_date_time': '2008-10-24T02:09:59',
  'transportation_mode': None,
  'user_id': '000'},
 {'_id': 2,
  'end_date_time': '2008-10-26T15:04:07',
  'start_date_time': '2008-10-26T13:44:07',
  'transportation_mode': None,
  'user_id': '000'},
 {'_id': 3,
  'end_date_time': '2008-10-27T12:05:54',
  'start_date_time': '2008-10-27T11:54:49',
  'transportation_mode': None,
  'user_id': '000'},
 {'_id': 4,
  'end_date_time': '2008-10-28T05:03:42',
  'start_date_time': '2008-10-28T00:38:26',
  'transportation_mode': None,
  'user_id': '000'},
 {'_id': 5,
  'end_date_time': '2008-10-29T09:30:28',
  'start_date_time': '2008-10-29T09:21:38',
  'transportation_mode': None,
  'user_id': '000'},
 {'_id': 6,
  'end_date_time': '2008-10-29T09:46:43',
  'start_date_time': '2008-10-29T09:30:38',
  'transportation_mode': None,
  'user_id': '000'},
 {'_id': 7,
  'end_date_time': '2008-11-03T10:16:01',
  'start_date_time': '2008-11-03T10:13:36',
  'transportation_mode': None,
  'user_id': '000'},
 {'_id': 8,
  'end_date_time': '2008-11-04T03:31:08',
  'start_date_time': '2008-11-03T23:21:53',
  'transportation_mode': None,
  'user_id': '000'},
 {'_id': 9,
  'end_date_time': '2008-11-10T03:46:12',
  'start_date_time': '2008-11-10T01:36:37',
  'transportation_mode': None,
  'user_id': '000'}]
```



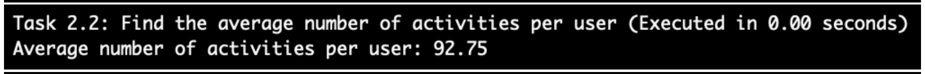
```
First 10 TrackPoint documents:
[{'_id': 0,
  'activity_id': 0,
  'altitude': 492,
  'date_days': 39744.1201851852,
  'date_time': '2008-10-23T02:53:04',
  'lat': 39.984702,
  'lon': 116.318417},
 {'_id': 1,
  'activity_id': 0,
  'altitude': 492,
  'date_days': 39744.1202546296,
  'date_time': '2008-10-23T02:53:10',
  'lat': 39.984683,
  'lon': 116.31845},
 {'_id': 2,
  'activity_id': 0,
  'altitude': 492,
  'date_days': 39744.1203125,
  'date_time': '2008-10-23T02:53:15',
  'lat': 39.984686,
  'lon': 116.318417},
 {'_id': 3,
  'activity_id': 0,
  'altitude': 492,
  'date_days': 39744.1203703704,
  'date_time': '2008-10-23T02:53:20',
  'lat': 39.984688,
  'lon': 116.318385},
 {'_id': 4,
  'activity_id': 0,
  'altitude': 492,
  'date_days': 39744.1204282407,
  'date_time': '2008-10-23T02:53:25',
  'lat': 39.984655,
  'lon': 116.318263},
 {'_id': 5,
  'activity_id': 0,
  'altitude': 493,
  'date_days': 39744.1204861111,
  'date_time': '2008-10-23T02:53:30',
  'lat': 39.984611,
  'lon': 116.318026},
 {'_id': 6,
  'activity_id': 0,
  'altitude': 493,
  'date_days': 39744.1205439815,
  'date_time': '2008-10-23T02:53:35',
  'lat': 39.984608,
  'lon': 116.317761},
 {'_id': 7,
  'activity_id': 0,
  'altitude': 496,
  'date_days': 39744.1206018519,
  'date_time': '2008-10-23T02:53:40',
  'lat': 39.984563,
  'lon': 116.317517},
 {'_id': 8,
  'activity_id': 0,
  'altitude': 500,
  'date_days': 39744.1206597222,
  'date_time': '2008-10-23T02:53:45',
```

- **QUERIES**

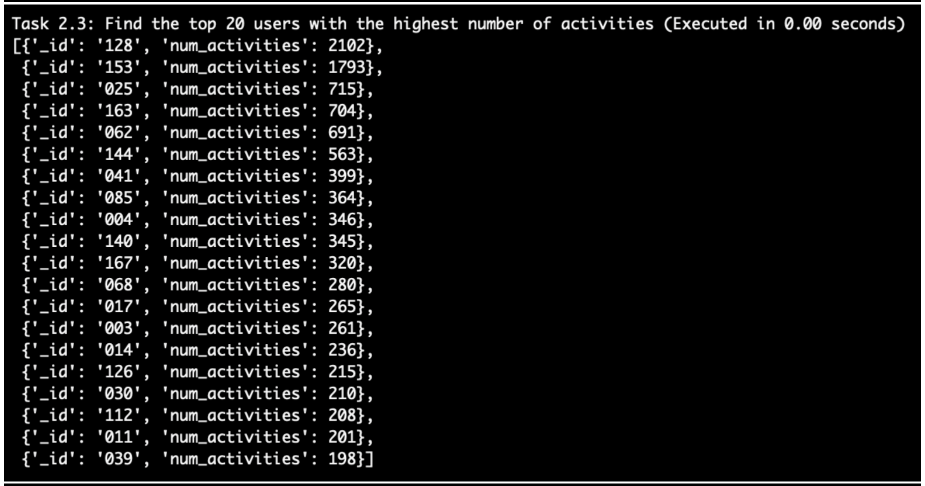
As observed from the execution times of the first seven queries, *MongoDB* performs exceptionally well, with speeds comparable to *MySQL* for these simple queries. For straightforward operations such as counting, filtering, or basic aggregations, MongoDB matches MySQL's efficiency, executing queries in just a few seconds or even instantaneously. This demonstrates MongoDB's capability to handle simple queries effectively, delivering performance on par with relational databases for basic data retrieval tasks.



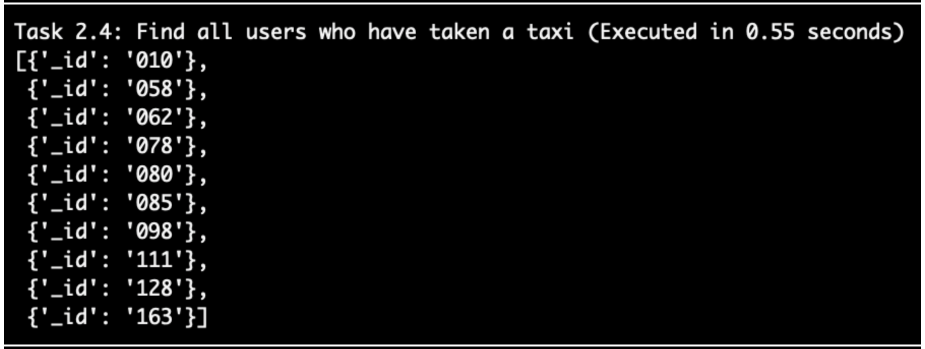
```
Task 2.1: How many users, activities and trackpoints are there in the dataset? (Executed in 1.20 seconds)
Number of users: 182
Number of activities: 16046
Number of trackpoints: 9676756
```



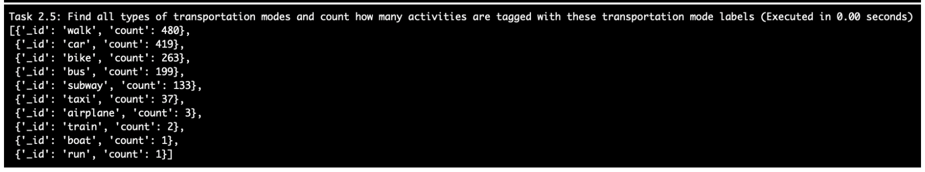
```
Task 2.2: Find the average number of activities per user (Executed in 0.00 seconds)
Average number of activities per user: 92.75
```



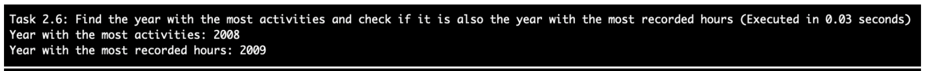
```
Task 2.3: Find the top 20 users with the highest number of activities (Executed in 0.00 seconds)
[{'_id': '128', 'num_activities': 2102},
 {'_id': '153', 'num_activities': 1793},
 {'_id': '025', 'num_activities': 715},
 {'_id': '163', 'num_activities': 704},
 {'_id': '062', 'num_activities': 691},
 {'_id': '144', 'num_activities': 563},
 {'_id': '041', 'num_activities': 399},
 {'_id': '085', 'num_activities': 364},
 {'_id': '004', 'num_activities': 346},
 {'_id': '140', 'num_activities': 345},
 {'_id': '167', 'num_activities': 320},
 {'_id': '068', 'num_activities': 280},
 {'_id': '017', 'num_activities': 265},
 {'_id': '003', 'num_activities': 261},
 {'_id': '014', 'num_activities': 236},
 {'_id': '126', 'num_activities': 215},
 {'_id': '030', 'num_activities': 210},
 {'_id': '112', 'num_activities': 208},
 {'_id': '011', 'num_activities': 201},
 {'_id': '039', 'num_activities': 198}]
```



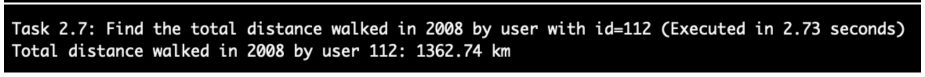
```
Task 2.4: Find all users who have taken a taxi (Executed in 0.55 seconds)
[{'_id': '010'},
 {'_id': '058'},
 {'_id': '062'},
 {'_id': '078'},
 {'_id': '080'},
 {'_id': '085'},
 {'_id': '098'},
 {'_id': '111'},
 {'_id': '128'},
 {'_id': '163'}]
```



```
Task 2.5: Find all types of transportation modes and count how many activities are tagged with these transportation mode labels (Executed in 0.00 seconds)
[{'_id': 'walk', 'count': 480},
 {'_id': 'car', 'count': 419},
 {'_id': 'bike', 'count': 263},
 {'_id': 'bus', 'count': 199},
 {'_id': 'subway', 'count': 133},
 {'_id': 'taxi', 'count': 37},
 {'_id': 'airplane', 'count': 3},
 {'_id': 'train', 'count': 2},
 {'_id': 'boat', 'count': 1},
 {'_id': 'run', 'count': 1}]
```

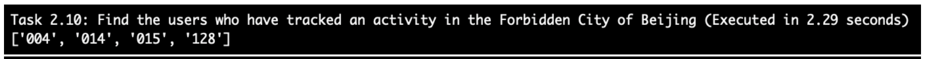


```
Task 2.6: Find the year with the most activities and check if it is also the year with the most recorded hours (Executed in 0.03 seconds)
Year with the most activities: 2008
Year with the most recorded hours: 2009
```

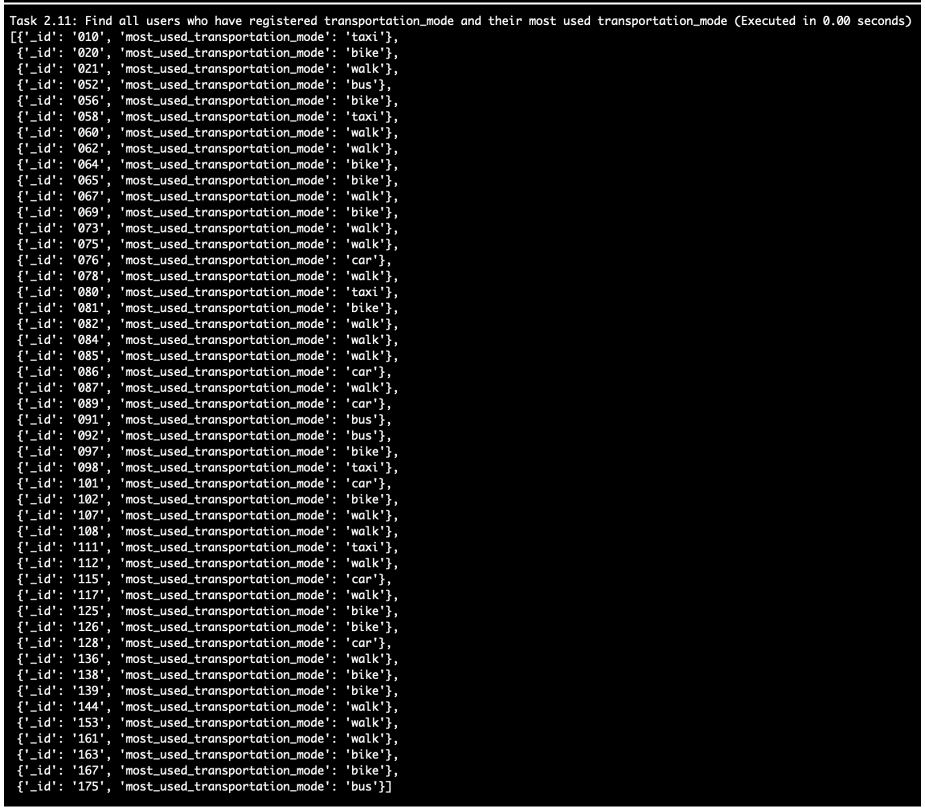


```
Task 2.7: Find the total distance walked in 2008 by user with id=112 (Executed in 2.73 seconds)
Total distance walked in 2008 by user 112: 1362.74 km
```

For more complex queries, particularly those requiring **multiple joins** across collections (*such as joining with the **TrackPoint collection** containing over **nine million documents***), MongoDB presented significant challenges. Unlike *MySQL*, MongoDB lacks inherent optimization mechanisms for complex joins and aggregations on large datasets, leading to notably longer execution times. For instance, queries 8 and 9 could not produce results within a reasonable timeframe, exceeding **half an hour** of processing time. Even after attempting to optimize with smaller query segments and more efficient Python code, the execution time remained prohibitively long. This highlights MongoDB's limitations in handling complex, join-intensive queries compared to the optimized capabilities of a relational DBMS.



```
Task 2.10: Find the users who have tracked an activity in the Forbidden City of Beijing (Executed in 2.29 seconds)
['004', '014', '015', '128']
```



```
Task 2.11: Find all users who have registered transportation_mode and their most used transportation_mode (Executed in 0.00 seconds)
[{'_id': '010', 'most_used_transportation_mode': 'taxi'},
 {'_id': '020', 'most_used_transportation_mode': 'bike'},
 {'_id': '021', 'most_used_transportation_mode': 'walk'},
 {'_id': '052', 'most_used_transportation_mode': 'bus'},
 {'_id': '056', 'most_used_transportation_mode': 'bike'},
 {'_id': '058', 'most_used_transportation_mode': 'taxi'},
 {'_id': '060', 'most_used_transportation_mode': 'walk'},
 {'_id': '062', 'most_used_transportation_mode': 'walk'},
 {'_id': '064', 'most_used_transportation_mode': 'bike'},
 {'_id': '065', 'most_used_transportation_mode': 'bike'},
 {'_id': '067', 'most_used_transportation_mode': 'walk'},
 {'_id': '069', 'most_used_transportation_mode': 'bike'},
 {'_id': '073', 'most_used_transportation_mode': 'walk'},
 {'_id': '075', 'most_used_transportation_mode': 'walk'},
 {'_id': '076', 'most_used_transportation_mode': 'car'},
 {'_id': '078', 'most_used_transportation_mode': 'walk'},
 {'_id': '080', 'most_used_transportation_mode': 'taxi'},
 {'_id': '081', 'most_used_transportation_mode': 'bike'},
 {'_id': '082', 'most_used_transportation_mode': 'walk'},
 {'_id': '084', 'most_used_transportation_mode': 'walk'},
 {'_id': '085', 'most_used_transportation_mode': 'walk'},
 {'_id': '086', 'most_used_transportation_mode': 'car'},
 {'_id': '087', 'most_used_transportation_mode': 'walk'},
 {'_id': '089', 'most_used_transportation_mode': 'car'},
 {'_id': '091', 'most_used_transportation_mode': 'bus'},
 {'_id': '092', 'most_used_transportation_mode': 'bus'},
 {'_id': '097', 'most_used_transportation_mode': 'bike'},
 {'_id': '098', 'most_used_transportation_mode': 'taxi'},
 {'_id': '101', 'most_used_transportation_mode': 'car'},
 {'_id': '102', 'most_used_transportation_mode': 'bike'},
 {'_id': '107', 'most_used_transportation_mode': 'walk'},
 {'_id': '108', 'most_used_transportation_mode': 'walk'},
 {'_id': '111', 'most_used_transportation_mode': 'taxi'},
 {'_id': '112', 'most_used_transportation_mode': 'walk'},
 {'_id': '115', 'most_used_transportation_mode': 'car'},
 {'_id': '117', 'most_used_transportation_mode': 'walk'},
 {'_id': '125', 'most_used_transportation_mode': 'bike'},
 {'_id': '126', 'most_used_transportation_mode': 'bike'},
 {'_id': '128', 'most_used_transportation_mode': 'car'},
 {'_id': '136', 'most_used_transportation_mode': 'walk'},
 {'_id': '138', 'most_used_transportation_mode': 'bike'},
 {'_id': '139', 'most_used_transportation_mode': 'bike'},
 {'_id': '144', 'most_used_transportation_mode': 'walk'},
 {'_id': '153', 'most_used_transportation_mode': 'walk'},
 {'_id': '161', 'most_used_transportation_mode': 'walk'},
 {'_id': '163', 'most_used_transportation_mode': 'bike'},
 {'_id': '167', 'most_used_transportation_mode': 'bike'},
 {'_id': '175', 'most_used_transportation_mode': 'bus'}]
```

1.1.3 Discussion

The instructions provided in the assignment document were once again very helpful, offering clear guidance through each step. However, this project posed a greater challenge for me because it was my first experience working with a *NoSQL database*, specifically *MongoDB*. Unlike previous experience with relational databases like *MySQL*, I had to learn MongoDB's syntax, data structures, and the aggregation pipeline to perform complex queries, which was a steep learning curve. Additionally, working independently meant that balancing tasks like learning new syntax, writing optimized code, and testing complex queries was particularly demanding. As part of my learning process, I found MongoDB's resources on handling **one-to-one**, **one-to-many**, and **one-to-squillions** relationships extremely helpful. These resources helped me understand how to model complex relationships within MongoDB, a skill that proved essential for working with large datasets like the **TrackPoint collection**, which contains over **nine million documents**. Modeling and handling these relationships effectively in MongoDB required different strategies than those in SQL, and these guides helped me understand best practices for organizing and accessing related data efficiently. A critical point in this project was handling **expensive queries**. For complex queries involving large datasets, MongoDB's performance proved to be a limitation. Query execution times were lengthy, especially for operations that required data from multiple collections and sequential processing. Even after MongoDB returned the data, additional time was needed in Python to process and format results, further extending the total execution time. Comparing this project to the MySQL project, I gained insight into several key differences between *SQL* and *NoSQL* databases:

- **Data Flexibility**: MongoDB's document-oriented structure allows for flexible schema designs, which is advantageous for storing unstructured or semi-structured data. This flexibility is particularly useful when the data model may evolve over time, as documents can vary in structure without requiring a predefined schema.
- **Data Insertion Speed**: MongoDB showed a clear advantage in data insertion speeds. Without any additional optimization techniques, inserting data into MongoDB collections was significantly faster than MySQL, especially for the **TrackPoint collection** with over **nine million documents**.
- **Modeling Complex Relationships**: In MongoDB, I learned to apply various techniques to handle complex relationships effectively, especially *one-to-squillions* relations. These techniques, such as embedding data within documents or using references based on data access patterns, helped to improve both storage efficiency and retrieval speed, particularly for high-volume data like track points.
- **Query Complexity and Optimization**: While MySQL handled complex, join-heavy queries relatively well due to built-in optimizations and indexing options, MongoDB struggled with queries requiring extensive joins or aggregations across collections. In MongoDB, such queries required

more detailed planning and experimentation with the aggregation pipeline. Complex queries that require **window functions** (like `ROW_NUMBER()`) in SQL are notably challenging to reproduce efficiently in MongoDB.

- **Scalability and ACID Transactions**: MongoDB's flexibility and scalability come at the cost of some traditional *DBMS* features, such as robust support for **ACID transactions**. This can complicate operations that require strict transactional integrity, a feature where relational databases often excel.

This project provided me with a better understanding of the fundamental differences between *SQL* and *NoSQL* databases, highlighting each approach's unique advantages and limitations. I learned how to structure and query data in MongoDB, utilizing the aggregation pipeline to replicate complex operations, and how to improve Python code to manage the results when database processing times are prolonged. Additionally, studying best practices for modeling relationships in MongoDB expanded my understanding of how to effectively structure data in a document-oriented database, especially for one-to-many and one-to-squillions relationships. This experience not only expanded my technical skills with NoSQL but also deepened my understanding of choosing the right database type depending on the application's requirements and data structure.