THE UNIVERSITY
OF TEXAS AT DALLAS

PROJECT – 3

GROUP - 18

(BUAN 6320: Database Foundations for Business Analytics)

Abhishek Arya	**axa220149@utdallas.edu**
Manan Mistry	**mxm220108@utdallas.edu**
Tanishka Patel	**txp210040@utdallas.edu**
Het Shah	**hrs220002@utdallas.edu**
Shekhar Vashist	**sxv220008@utdallas.edu**

Project made under the guidance of: Farzad Kamalzadeh

SECTION 1: INDEXING AND QUERY TIMING

1.1 List all the current indexes in your database and the columns they are associated with along with the index type.

We have 5 tables in our database and below are the indexes that we have for each table along with the column names and index type.

Table	Key	Type	Column
hierarchy	Primary	BTREE	hierarchy5_id
productdetails	Primary	BTREE	product_id, product_length
	hierarchy5_id_idx	BTREE	hierarchy5_id
promotion	Primary	BTREE	promo_id
	prmo_id_idx	BTREE	promo_id
sales	Primary	BTREE	sales_id
	store_id_idx	BTREE	store_id
	product_id_idx	BTREE	product_id
	promo_id_idx	BTREE	promo_id
storedetails	Primary	BTREE	store_id
	store_id_index	BTREE	store_id

1.2 Explain what is in common between these columns (why these columns are indexed automatically by the database management system)

These indexes are the primary keys and the foreign keys which we incorporated while creating our schema and that is why the database management system has automatically added them.

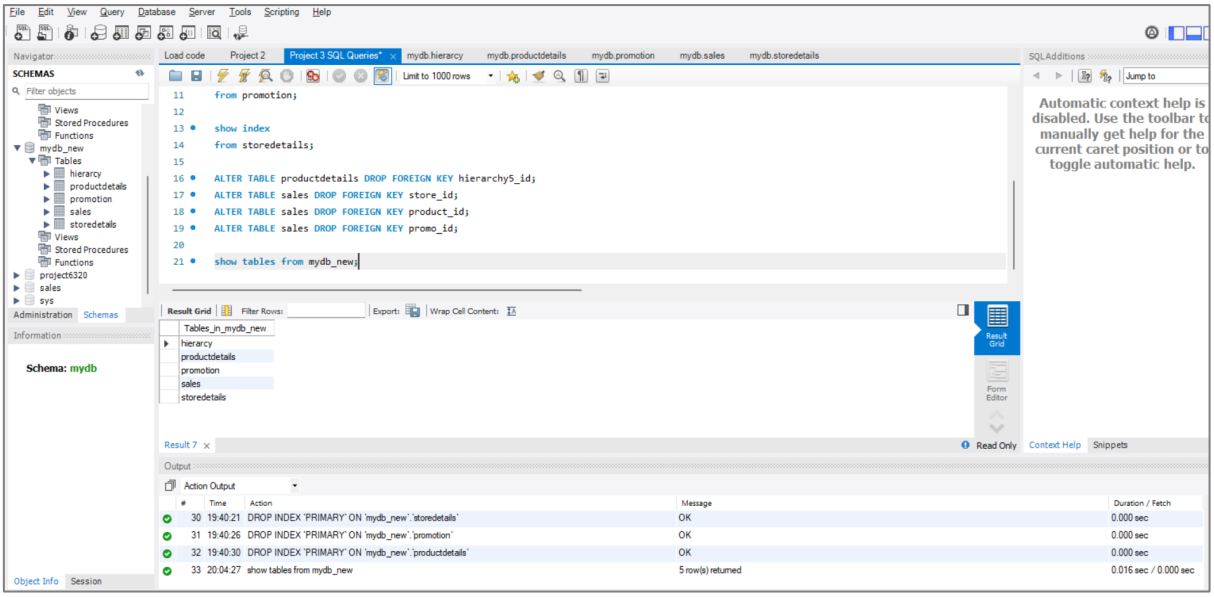
1.3 Make a copy of your database and delete all the indexes there (you might need to delete foreign keys before you can delete some of the indexes) – now you have two databases: database A with indexes and database B without any indexes.

We have created a duplicate database and named it “mydb_new”. So, we have 2 databases “mydb” and “mydb_new”. As mentioned, we have removed the indexes from the mydb_new and to do that we had to remove the foreign key constraints which we did by using the ALTER query.

Query used to remove the foreign keys:

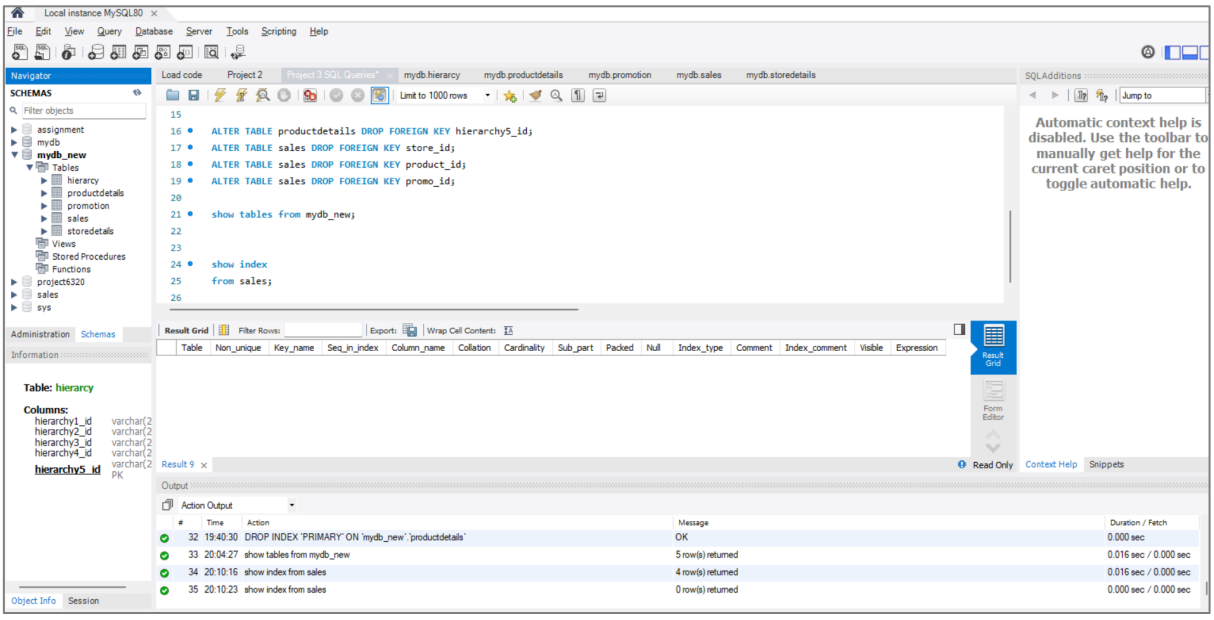
```
ALTER TABLE productdetails DROP FOREIGN KEY hierarchy5_id;
ALTER TABLE sales DROP FOREIGN KEY store_id;
ALTER TABLE sales DROP FOREIGN KEY product_id;
ALTER TABLE sales DROP FOREIGN KEY promo_id;
```

Screen shot for the tables we have in our mydb_new database.

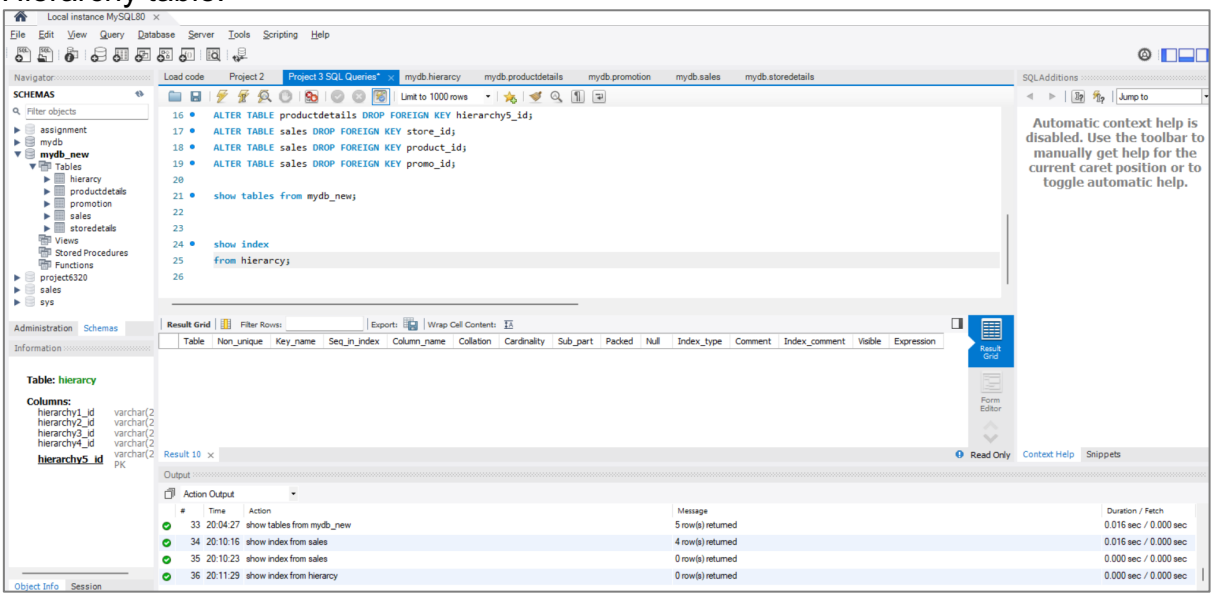


Screenshots after checking the indexes from our new database:

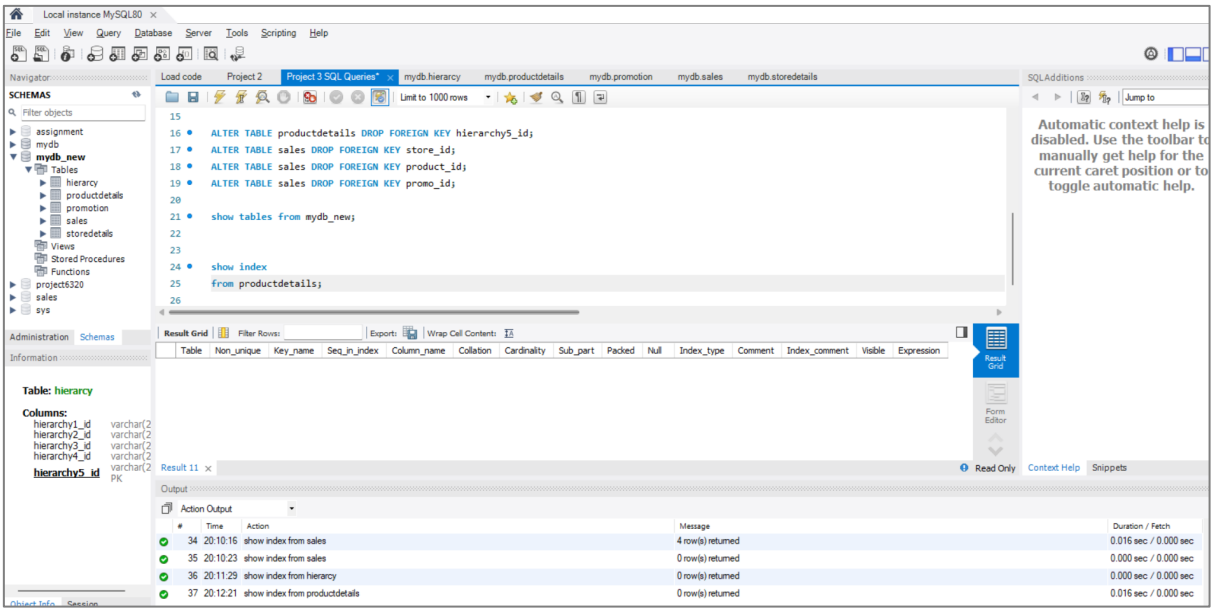
Sales Table:



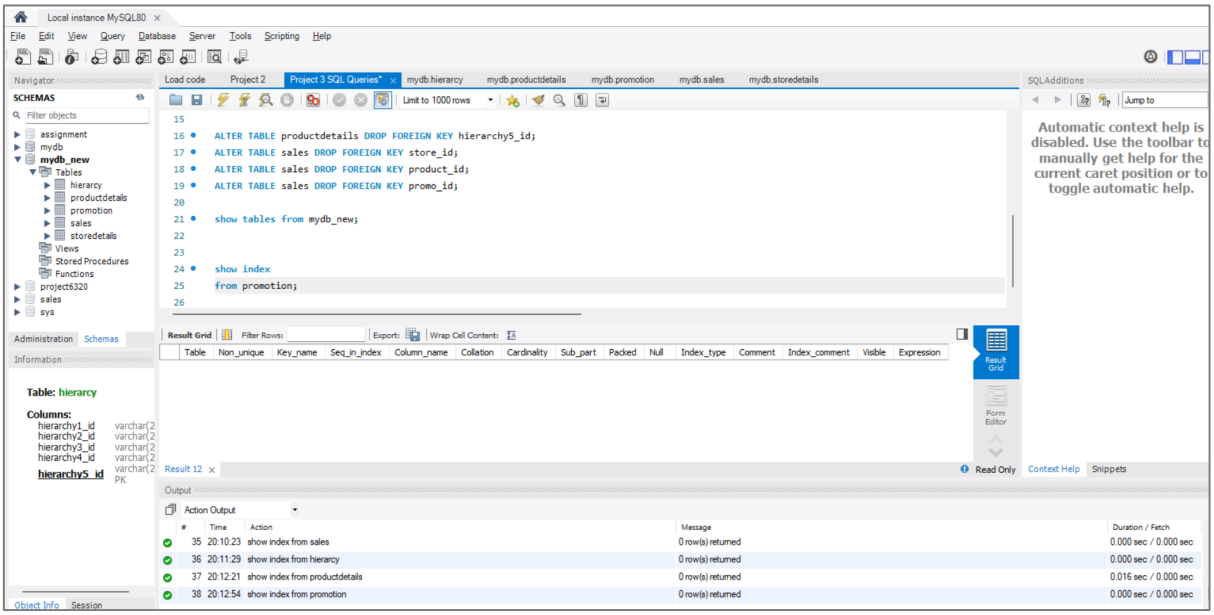
Hierarchy table:



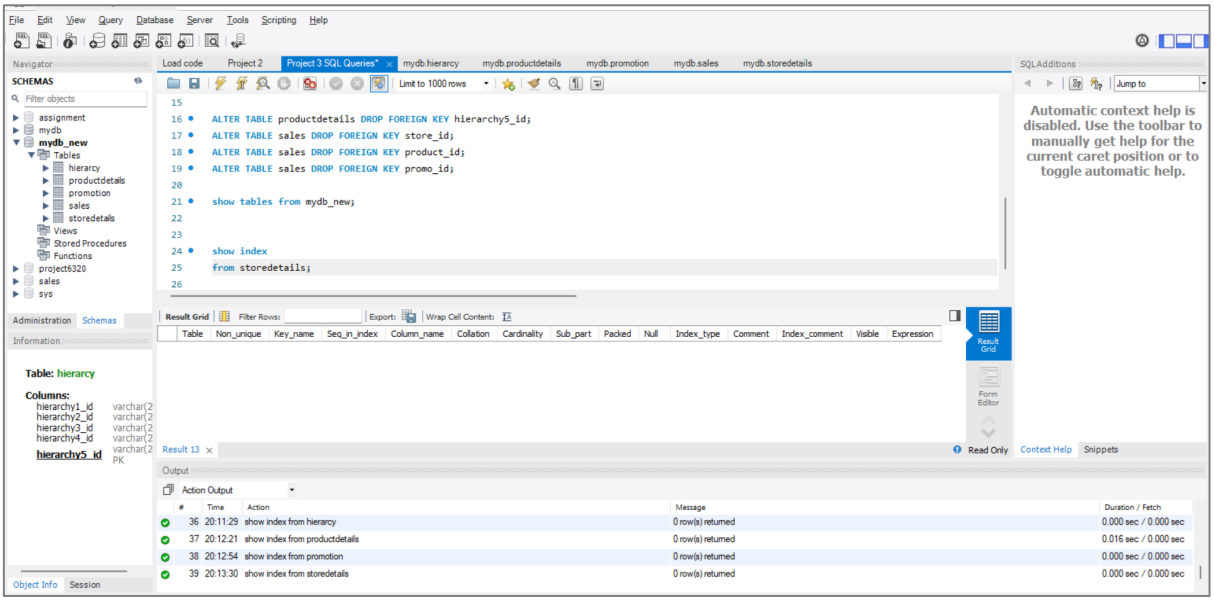
Productdetails table:



Promotion Table:



Storedetails table:



1.4 Write at least 5 queries (with JOINs between your tables)

```
SELECT productdetails.hierarchy5_id , hierarchy4_id, cluster_id
FROM hierarcy
JOIN productdetails
ON hierarcy.hierarchy5_id = productdetails.hierarchy5_id;
```

```
SELECT T1.product_id , T2.store_id, T2.store_size
FROM sales as T1
JOIN storedetails as T2
```

```
on T1.store_id = T2.store_id;


SELECT *
FROM sales as T1
JOIN productdetails as T2
on T1.product_id = T2.product_id;


SELECT *
FROM productdetails p
LEFT JOIN hierarcy h
ON p.hierarchy5_id = h.hierarchy5_id;


SELECT *
FROM sales as T1
JOIN promotion as T2
on T1.promo_id = T2.promo_id;
```

1.5 Execute and time these queries on both databases and report your findings (repeat timing for each query at least 10 times and average the times)

As described, we performed the above-mentioned 5 joins on both the databases and repeated them 10 times. Following are the durations for these queries along with the average durations.

Join 1		
S. No.	**Duration**	
	mydb	**mydb_new**
1	0.0092590	0.0143370
2	0.0042458	0.0048398
3	0.0047048	0.0029120
4	0.0047063	0.0029545
5	0.0061550	0.0034040
6	0.0049965	0.0029810
7	0.0047528	0.0027848
8	0.0056295	0.0035785
9	0.0044935	0.0026965
10	0.0043695	0.0018758
Average Time	**0.0053313**	**0.0042364**

Join 2		
S. No.	Duration	
	mydb	mydb_new
1	0.0648010	0.0066825
2	0.0232860	0.0029335
3	0.0254210	0.0032828
4	0.0121758	0.0031370
5	0.0032605	0.0032295
6	0.0028740	0.0030093
7	0.0081933	0.0019200
8	0.0052275	0.0030133
9	0.0034510	0.0020208
10	0.0031825	0.0028923
Average Time	**0.0151873**	**0.0032121**

Join 3		
S. No.	Duration	
	mydb	mydb_new
1	0.0077500	0.0069988
2	0.0079078	0.0058010
3	0.0042440	0.0050835
4	0.0049578	0.0056370
5	0.0046858	0.0042435
6	0.0044440	0.0055780
7	0.0048793	0.0054120
8	0.0047220	0.0050660
9	0.0048728	0.0060500
10	0.0057980	0.0052830
Average Time	**0.0054261**	**0.0055153**

Join 4		
S. No.	Duration	
	mydb	mydb_new
1	0.0045880	0.0033020
2	0.0041988	0.0036558
3	0.0045960	0.0037380
4	0.0045670	0.0035850
5	0.0044900	0.0040505
6	0.0039505	0.0043750
7	0.0049200	0.0039093
8	0.0044860	0.0037168
9	0.0036445	0.0028508
10	0.0043068	0.0039043
Average Time	**0.0043748**	**0.0037087**

Join 5		
S. No.	Duration	
	mydb	mydb_new
1	0.0047143	0.0060633
2	0.0043740	0.0037553
3	0.0044875	0.0033308
4	0.0039323	0.0040460
5	0.0042280	0.0038518
6	0.0036055	0.0040115
7	0.0040215	0.0035223
8	0.0041393	0.0040298
9	0.0035778	0.0040843
10	0.0038728	0.0030980
Average Time	**0.0040953**	**0.0039793**

Findings: As we can see, the average duration of the new database has given smaller durations which means that the queries are taking less time to perform in the database with no indexes as compared to the database with indexes.
Out of 5 joins there was only 1 join query (i.e., join 3) where mydb_new's duration was a little higher than the mydb other wise mydb_new has lesser duration in rest of the join queries.

1.6 Select some columns from database A (columns that are not already indexed) and create an index on them.

We have added the following indexes for the columns:

Table	Index	Column
hierarchy	idx_hierarcy_hierarchy4_id	hierarchy4_id
productdetails	idx_productdetails_cluster_id	cluster_id
sales	idx_sales_product_id	product_id
storedetails	idx_storedetails_storetype_id	storetype_id

1.7 Write a query for each column – the query should include the column in the WHERE clause in a condition.

Table: hierarchy

```
SELECT *
FROM hierarcy
WHERE hierarchy4_id = "H00000400";
```

Table: productdetails

```
SELECT *
FROM productdetails
WHERE cluster_id = "cluster_4";
```

Table: sales

```
SELECT *
FROM sales
WHERE product_id = "P0001";
```

Table: storedetails

```
SELECT *
FROM storedetails
WHERE storetype_id = "ST04";
```

1.8 Execute and time these queries on both databases and report your findings (repeat timing for each query at least 10 times and average the times)

Table: hierarcy		
S. No.	**Duration**	
	mydb	**mydb_new**
1	0.0011288	0.0055335
2	0.0019793	0.0005200
3	0.0005063	0.0005088
4	0.0003670	0.0006133
5	0.0003380	0.0004685
6	0.0003433	0.0006643
7	0.0004320	0.0004875
8	0.0003295	0.0005448
9	0.0003798	0.0005650
10	0.0007000	0.0005310
Average Time	**0.0006504**	**0.0010437**

Table: productdetails		
S. No.	Duration	
	mydb	mydb_new
1	0.0009948	0.0054110
2	0.0004148	0.0007455
3	0.0004050	0.0007740
4	0.0004728	0.0007400
5	0.0005270	0.0011115
6	0.0004880	0.0008088
7	0.0003865	0.0007418
8	0.0004785	0.0006983
9	0.0004108	0.0007315
10	0.0004175	0.0010785
Average Time	**0.0004996**	**0.0012841**

Table: sales		
S. No.	Duration	
	mydb	mydb_new
1	0.0030898	0.5782740
2	0.0021630	0.3881418
3	0.0030580	0.4212490
4	0.0018688	0.4032643
5	0.0018480	0.3667640
6	0.0019165	0.3753288
7	0.0018225	0.4124388
8	0.0026118	0.4310423
9	0.0018078	0.3882908
10	0.0019483	0.4552420
Average Time	**0.0022134**	**0.4220036**

Table: storedetails		
S. No.	Duration	
	mydb	mydb_new
1	0.0024603	0.0044748
2	0.0005505	0.0006833
3	0.0005988	0.0004403
4	0.0005240	0.0003680
5	0.0006048	0.0005415
6	0.0004660	0.0005228
7	0.0005288	0.0004973
8	0.0004798	0.0004260
9	0.0004520	0.0003800
10	0.0005983	0.0005263
Average Time	**0.0007263**	**0.0008860**

Findings: The results show that the database with indexes i.e., “mydb” has very less average duration for queries as compared to the database without indexes i.e., “mydb_new”.
This implies that creating indexes will make the queries run more efficiently, especially for the tables with large data like we can see for the sales table in our data which has large data and the difference in the average duration is quite visible.

1.9 Make a conclusion based on your findings in this part.

After comparing the results of JOINS and WHERE queries we can conclude that creating indexes will make queries run more efficiently.
Even though the average durations for JOIN queries were not that significantly less for the database with index but the difference is quite visible for the WHERE queries in database with indexes.
Fast data retrieval is one of the most important benefits of adding indexes hence, with indexes in place, MySQL can locate and retrieve data much faster, which can lead to significant improvements in query performance.

SECTION 2: MONGODB AND MQL

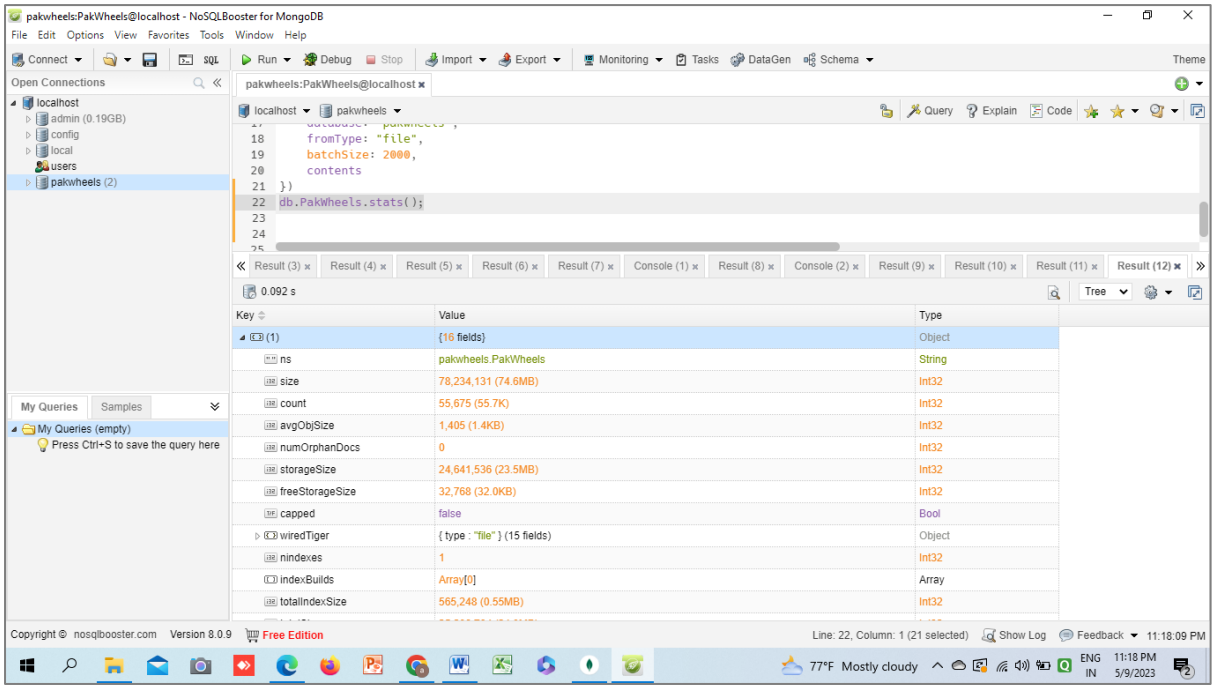
2.1. Explore your dataset and familiarize yourself with the dataset and its content

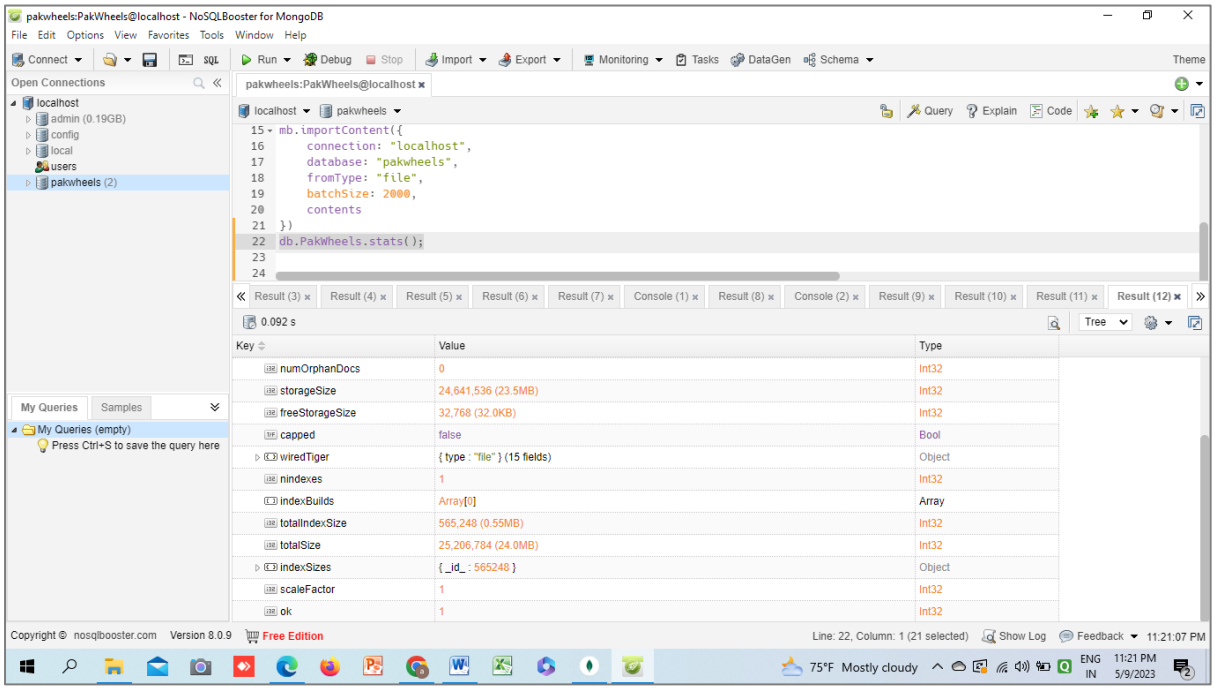
We have taken Pakistan's Largest PakWheels Automobiles Listings dataset to answer the following questions. It was in a json format and while importing in the no sql booster for MongoDB it automatically changes into table format. Looking into the unstructured data it is large in size, total documents count = 55675; size = 74.6 MB and average object size = 1405(1.4 KB). We can find many attribute/ fields like {@type, brand (@type, name)}, model, description , item condition, modelDate, manufacturer, FuelType, name, image, vehicleTransmission, color, bodyType, vehicleEngine, mileageFromOdometer, sellerLocation, postedFrom, keywords, extraFeatures , Features , price , priceCurrency .

The screenshots below shows the statistics of dataset that we have taken in this study.

We use the following query to obtain the stats:

db.PakWheels.stats();


pakwheels:PakWheels@localhost - NoSQLBooster for MongoDB
File Edit Options View Favorites Tools Window Help
Connect Run Debug Stop Import Export Monitoring Tasks DataGen Schema Theme
Open Connections
localhost
admin (0.19GB)
config
local
users
pakwheels (2)
pakwheels:PakWheels@localhost
localhost pakwheels
Query Explain Code
fromType: "file",
batchSize: 2000,
contents
})
db.PakWheels.stats();
Result (3) Result (4) Result (5) Result (6) Result (7) Console (1) Result (8) Console (2) Result (9) Result (10) Result (11) Result (12)
0.092 s
Tree
Key Value Type
(1) {16 fields} Object
ns pakwheels.PakWheels String
size 78,234,131 (74.6MB) Int32
count 55,675 (55.7K) Int32
avgObjSize 1,405 (1.4KB) Int32
numOrphanDocs 0 Int32
storageSize 24,641,536 (23.5MB) Int32
freeStorageSize 32,768 (32.0KB) Int32
capped false Bool
wiredTiger { type : "file" } (15 fields) Object
nindexes 1 Int32
indexBuilds Array[0] Array
totalIndexSize 565,248 (0.55MB) Int32
My Queries Samples
My Queries (empty)
Press Ctrl+S to save the query here
Copyright © nosqlbooster.com Version 8.0.9 Free Edition
Line: 22, Column: 1 (21 selected) Show Log Feedback 11:18:09 PM
77°F Mostly cloudy ENG IN 11:18 PM 5/9/2023


pakwheels:PakWheels@localhost - NoSQLBooster for MongoDB
File Edit Options View Favorites Tools Window Help
Connect Run Debug Stop Import Export Monitoring Tasks DataGen Schema Theme
Open Connections
localhost
admin (0.19GB)
config
local
users
pakwheels (2)
pakwheels:PakWheels@localhost
localhost pakwheels
Query Explain Code
mb.importContent({
connection: "localhost",
database: "pakwheels",
fromType: "file",
batchSize: 2000,
contents
})
db.PakWheels.stats();
Result (3) Result (4) Result (5) Result (6) Result (7) Console (1) Result (8) Console (2) Result (9) Result (10) Result (11) Result (12)
0.092 s
Tree
Key Value Type
numOrphanDocs 0 Int32
storageSize 24,641,536 (23.5MB) Int32
freeStorageSize 32,768 (32.0KB) Int32
capped false Bool
wiredTiger { type : "file" } (15 fields) Object
nindexes 1 Int32
indexBuilds Array[0] Array
totalIndexSize 565,248 (0.55MB) Int32
totalSize 25,206,784 (24.0MB) Int32
indexSizes { _id_ : 565248 } Object
scaleFactor 1 Int32
ok 1 Int32
My Queries Samples
My Queries (empty)
Press Ctrl+S to save the query here
Copyright © nosqlbooster.com Version 8.0.9 Free Edition
Line: 22, Column: 1 (21 selected) Show Log Feedback 11:21:07 PM
75°F Mostly cloudy ENG IN 11:21 PM 5/9/2023

2.2. Explain why it is better to use non-relational databases such as MongoDB to work with such a dataset (explain in the context of your dataset)

As we can see from the above screenshots the unstructured data is large in size around 75 MB with total document counts of 55675, one of the main advantages of non-relational databases is their ability to handle large volumes of unstructured or semi-structured data, which can be difficult to manage in a traditional relational database. This is because non-relational databases allow for flexible schema design and can store data in a variety of formats, such as JSON, BSON, or XML.

Another advantage of non-relational databases is their ability to scale horizontally, meaning that they can easily distribute data across multiple servers to handle high levels of traffic and ensure high availability. This is particularly useful for datasets that are constantly growing or that need to be accessed by many users at the same time**. Our dataset is of the same nature where new data can be added for each consecutive coming years whether related to car's bodytype, brand, mileage From Odometer or color.** So we can say that our dataset is lively and it would expand horizontally and for that non-relational database is a must.

As our dataset is large and unstructured, non-relational databases can provide faster read and write performance compared to relational databases, especially when dealing with large datasets. This is because non-relational databases can store data in a way that is optimized for fast retrieval and can use techniques such as sharing and indexing to speed up queries.

In summary, if your dataset is large and unstructured or semi-structured, needs to be accessed by many users simultaneously, or requires high performance and scalability, then a non-relational database such as MongoDB might be a better choice than a traditional relational database. However, the choice of database depends on many factors, including the nature of the data, the application requirements, and the available resources, so it's important to carefully evaluate the options before making a decision.

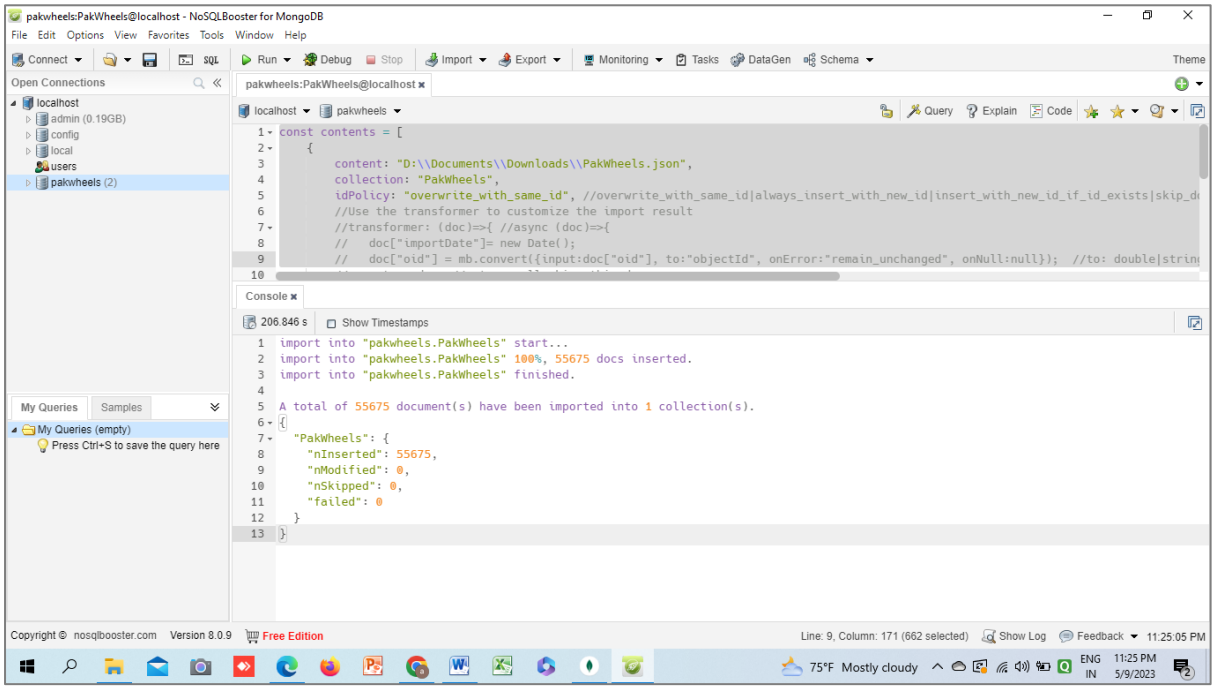
2.3. Import your dataset into MongoDB using either MongoDB Compass or MongoDB Database Tools*

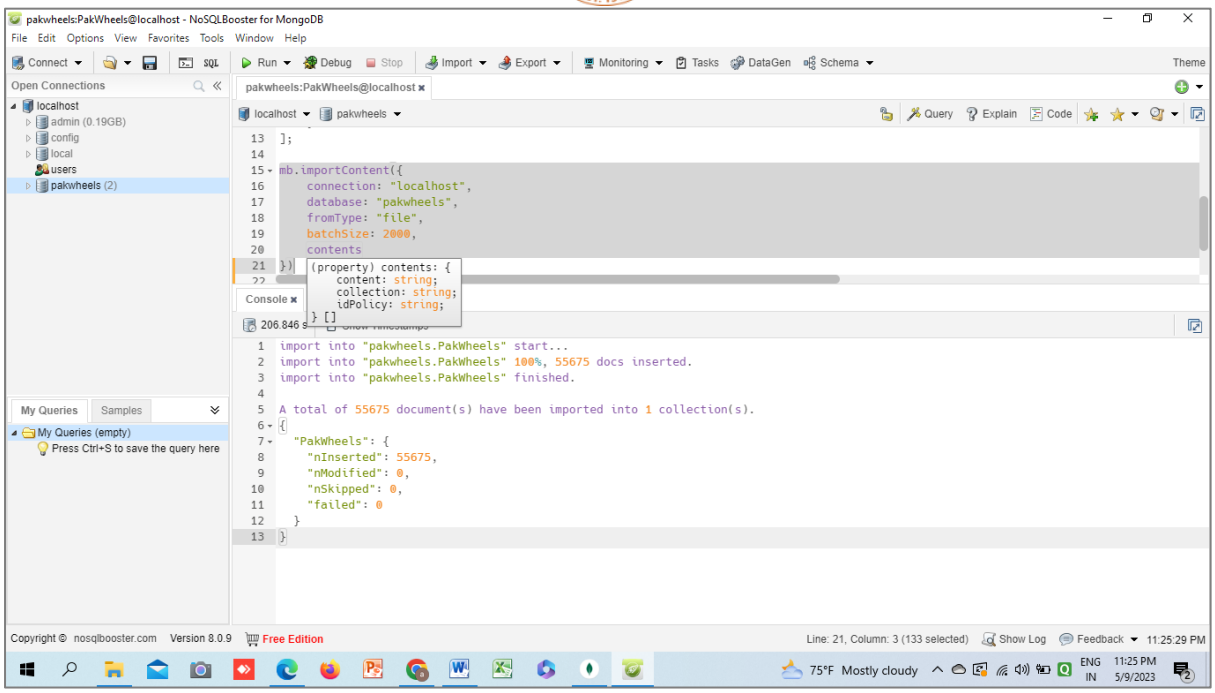
NoSQLBooster is one of the most popular cross-platform GUI (Graphical User Interface) **tools for MongoDB that houses a built-in** MongoDB Script Debugger, Comprehensive Server Monitoring Tools, Query Code Generator, and advanced IntelliSense support**.**

After importing the JSON dataset into the No SQL Booster MongoDB we found out the following output

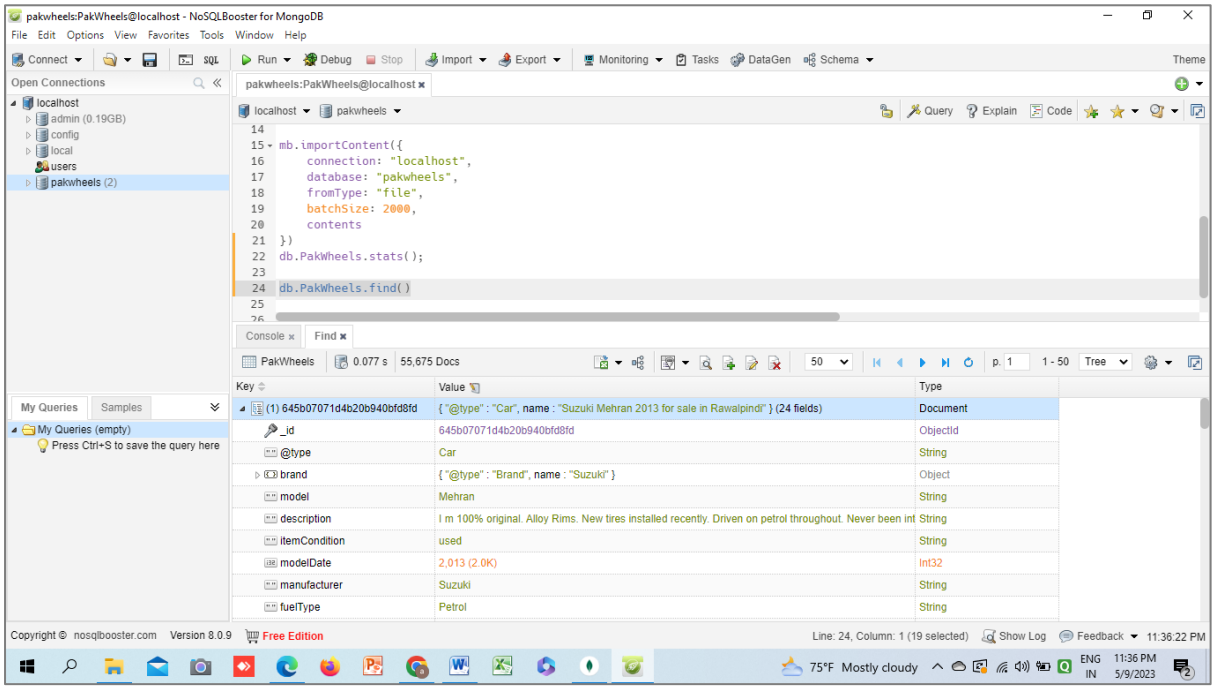
```
import into "pakwheels.PakWheels" start...
import into "pakwheels.PakWheels" 100%, 55675 docs inserted.
import into "pakwheels.PakWheels" finished.

A total of 55675 document(s) have been imported into 1 collection(s).
{
  "PakWheels": {
    "nInserted": 55675,
    "nModified": 0,
    "nSkipped": 0,
    "failed": 0
  }
}
```

The below screenshots shows the same:





By using db.PakWheels.find() function we can find out all the fields, values in all the documents in a collection. Screenshot is below:



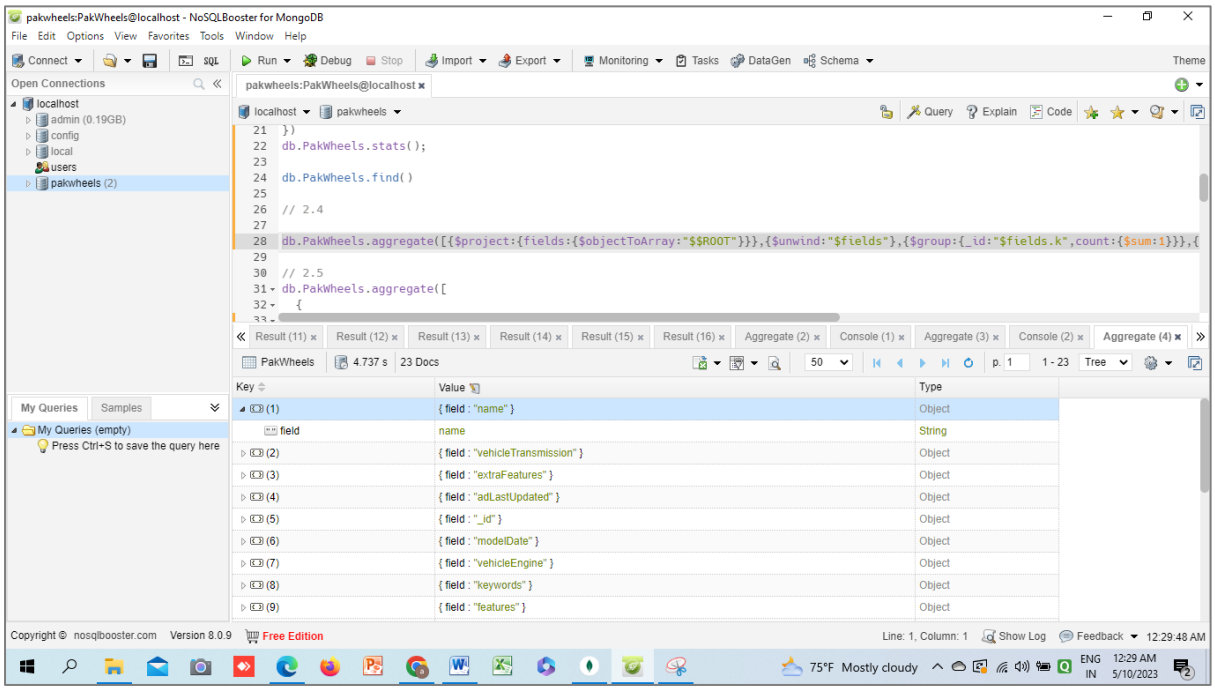
2.4. List some of the attributes (field/properties) of your database which are common among all documents.

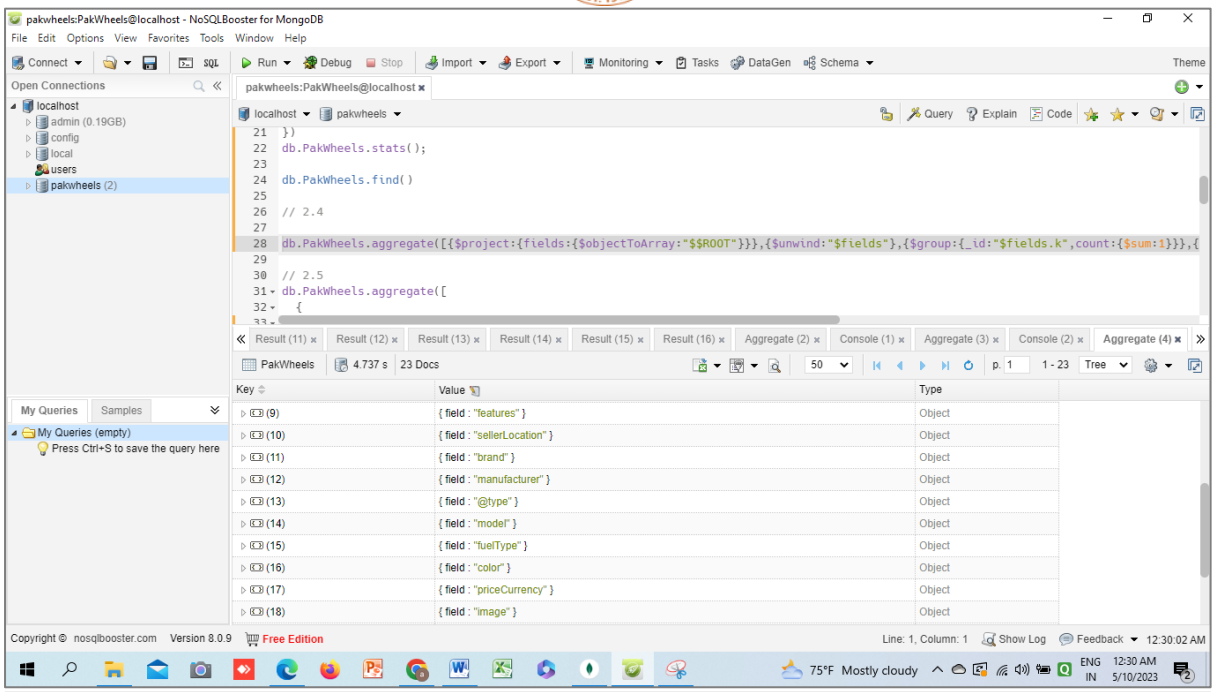
For this we use the following query to find out the common fields among all the documents:

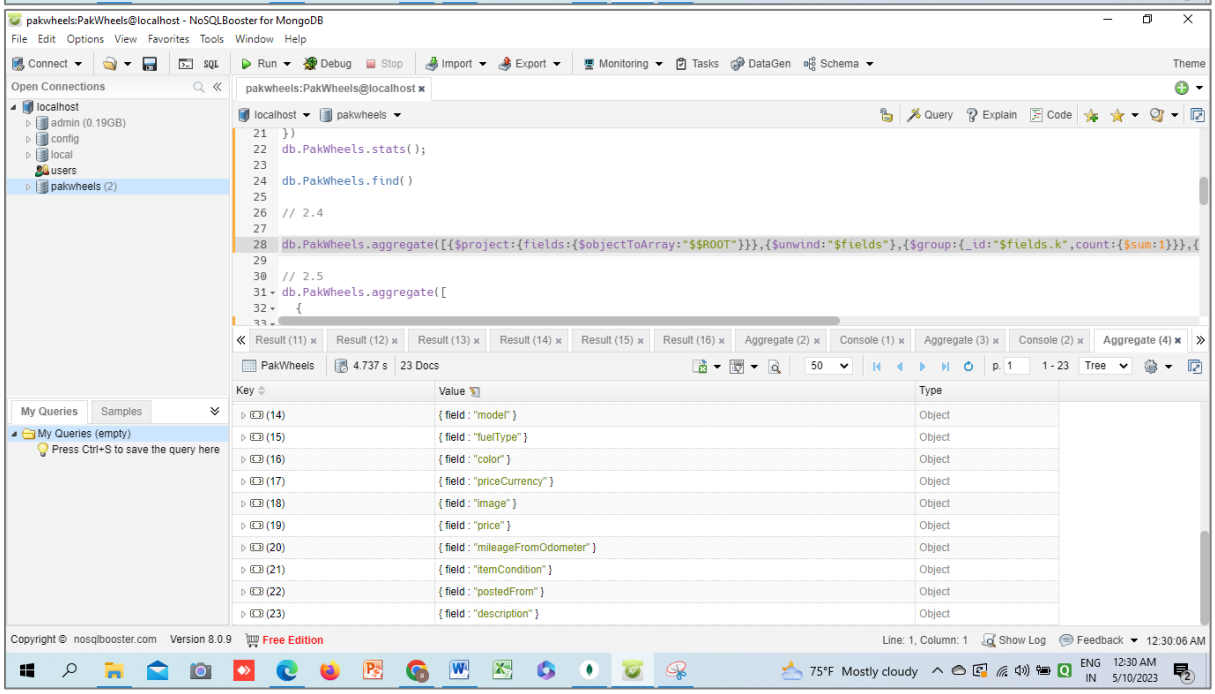
Query:

db.PakWheels.aggregate([{$project:{fields:{$objectToArray:"$$ROOT"}}},{$unwind:"$fields"},{$group:{_id:"$fields.k",count:{$sum:1}}},{$match:{count:{$gte:db.PakWheels.count()}}},{$project:{_id:0, field:"$_id"}}])

The below screenshots shows the output of the above query which shows that all the attributes are common except “bodyType”. 23 attributes are common among all the documents.



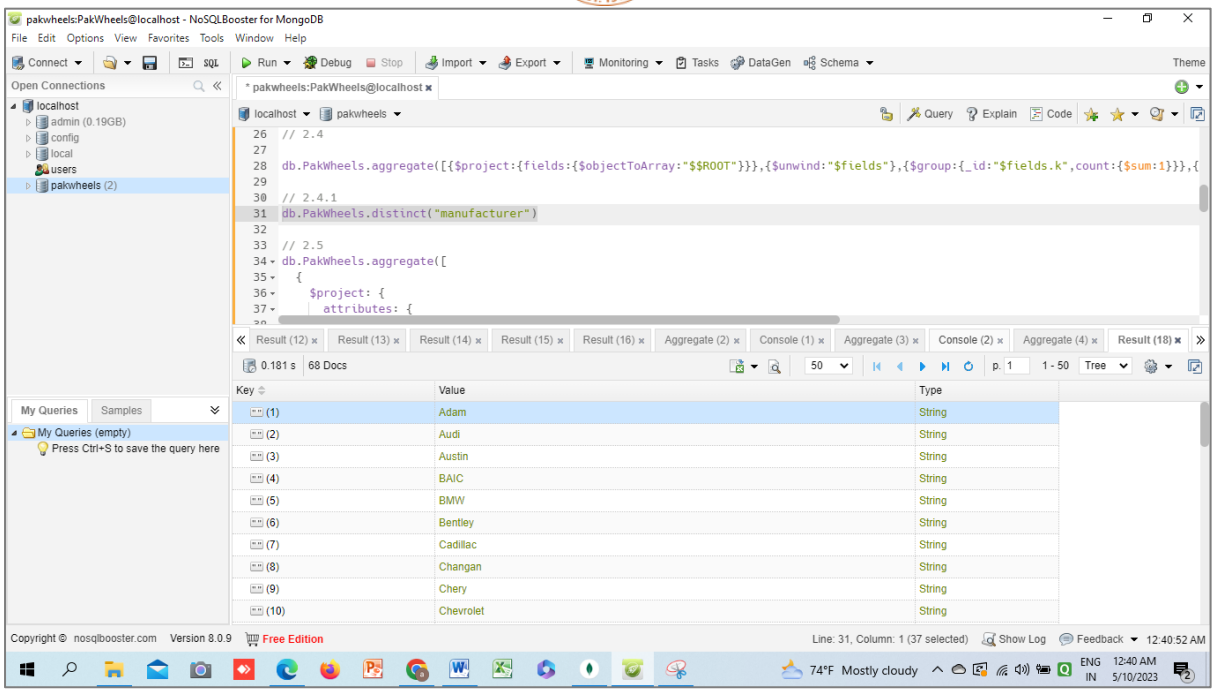


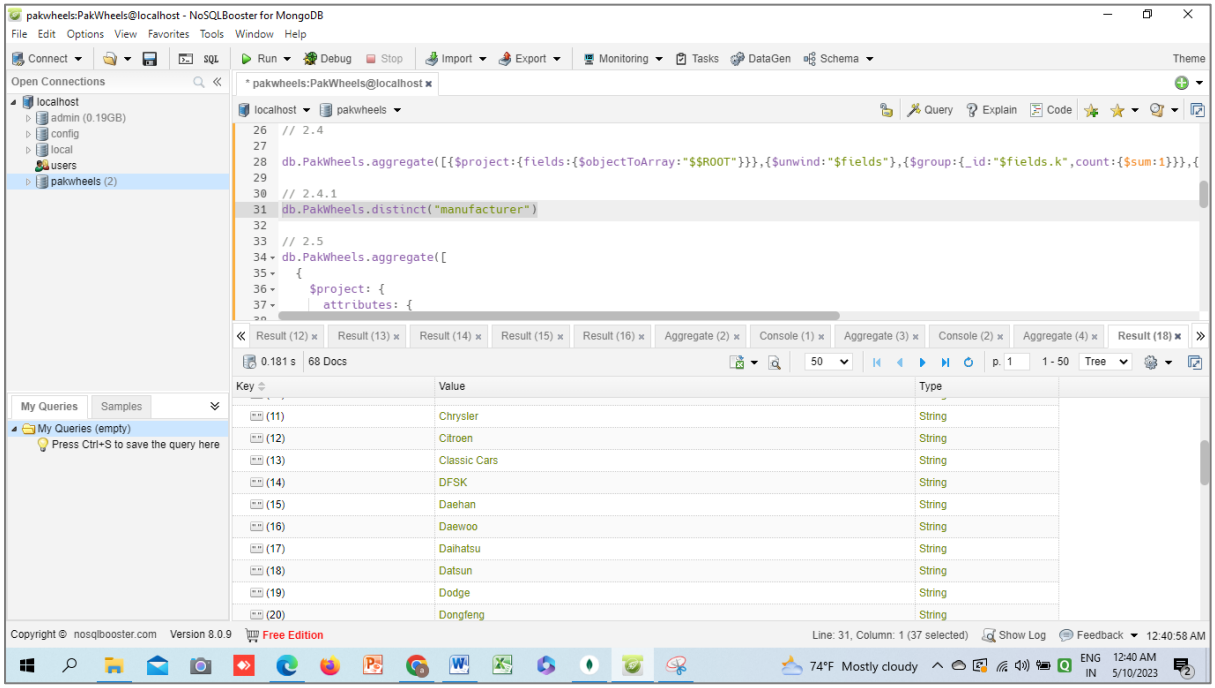


2.4.1. For these fields, provide some of the values they contain in the database

In case of “manufacturer” attribute total 50 values are there some of them are given below in the output console in the below screenshot

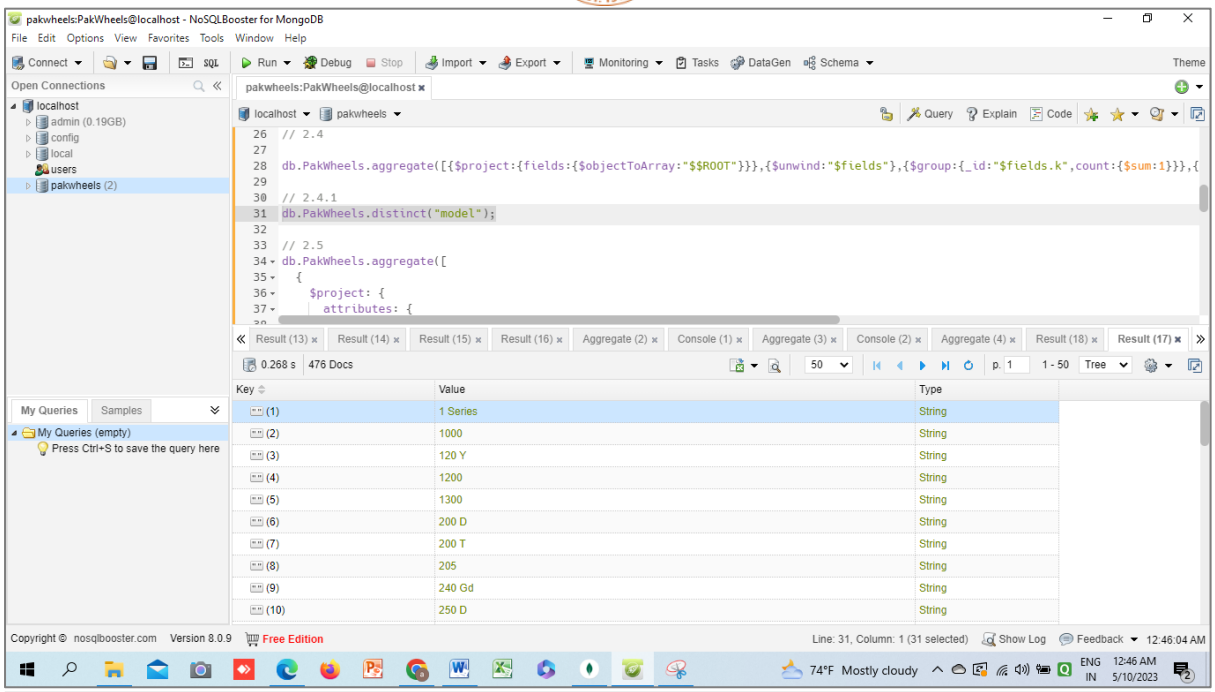
db.PakWheels.distinct("manufacturer");

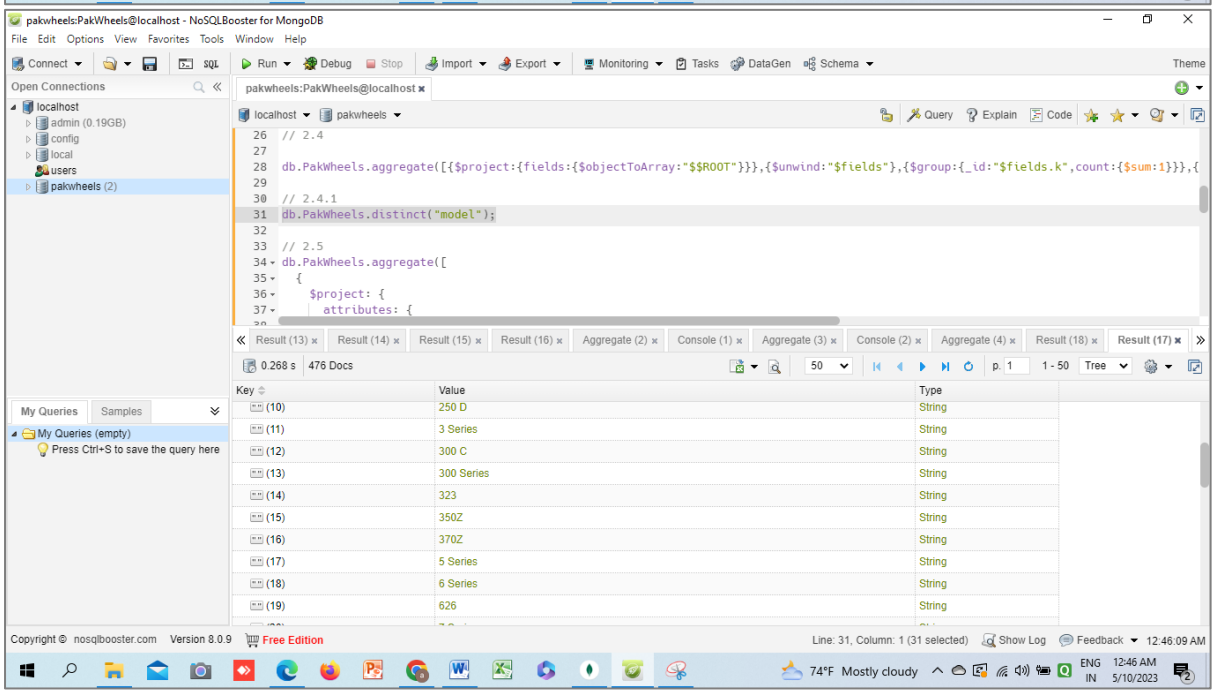




In case of “model” attribute total 50 values are there some of them are given below in the output console in the below screenshot

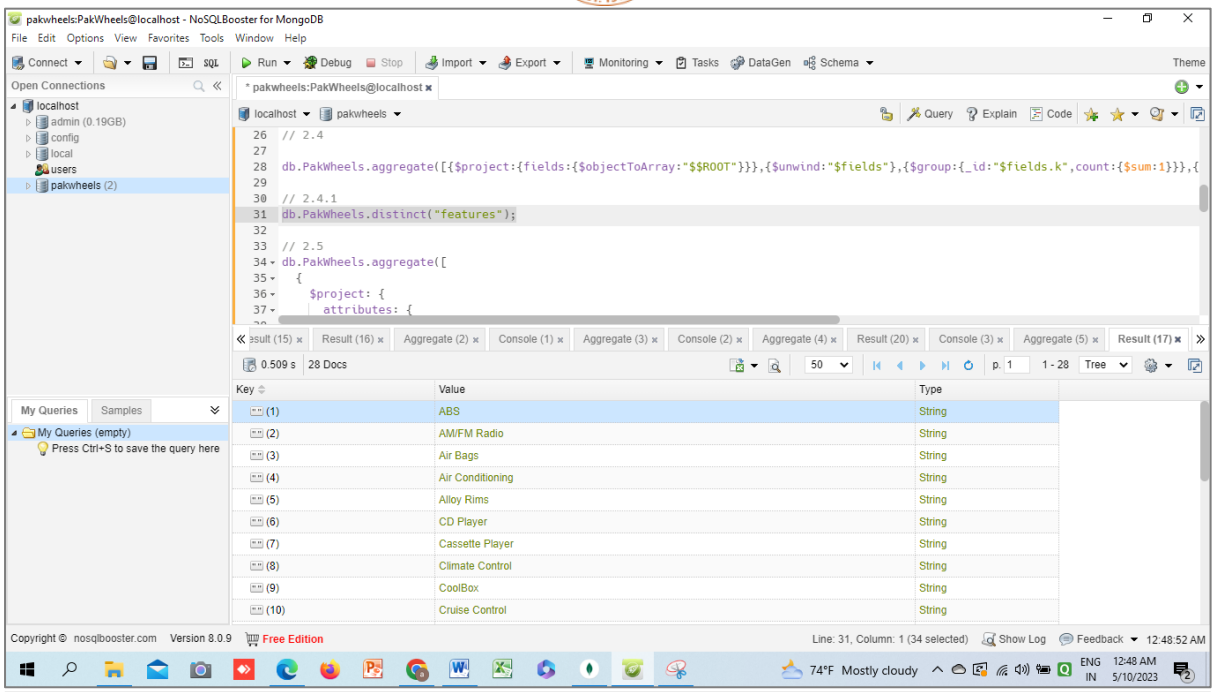
db.PakWheels.distinct("model");

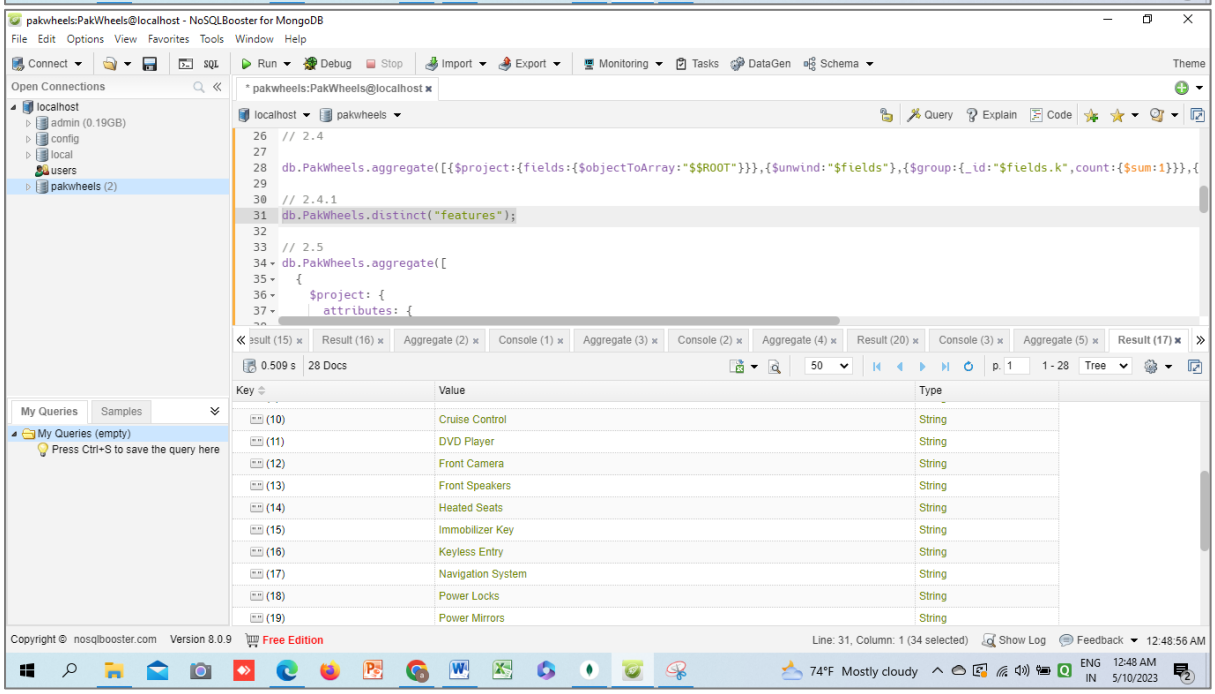


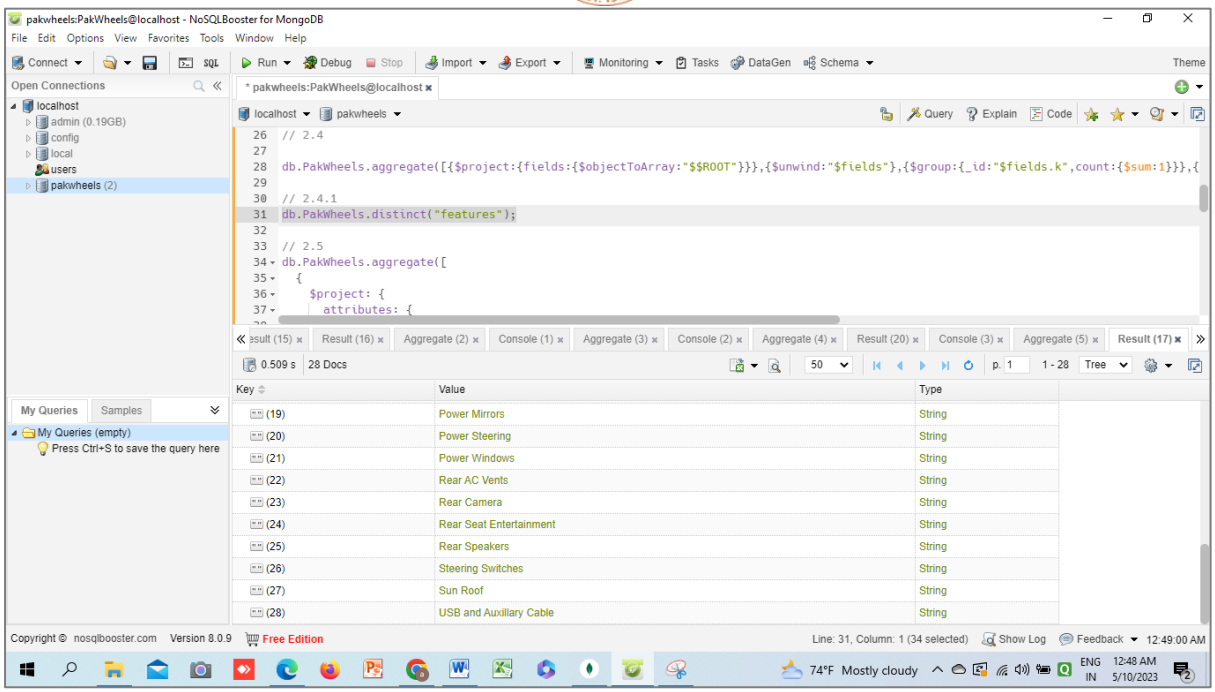


In case of “features” attribute total 28 values are there some of them are given below in the output console in the below screenshot

db.PakWheels.distinct("features");


pakwheels:PakWheels@localhost - NoSQLBooster for MongoDB
db.PakWheels.distinct("features");
0.509 s
28 Docs
Key
Value
Type
(1) ABS String
(2) AM/FM Radio String
(3) Air Bags String
(4) Air Conditioning String
(5) Alloy Rims String
(6) CD Player String
(7) Cassette Player String
(8) Climate Control String
(9) CoolBox String
(10) Cruise Control String


pakwheels:PakWheels@localhost - NoSQLBooster for MongoDB
db.PakWheels.distinct("features");
0.509 s
28 Docs
Key
Value
Type
(10) Cruise Control String
(11) DVD Player String
(12) Front Camera String
(13) Front Speakers String
(14) Heated Seats String
(15) Immobilizer Key String
(16) Keyless Entry String
(17) Navigation System String
(18) Power Locks String
(19) Power Mirrors String



Similarly, we can find out the rest of the values of these common attributes.

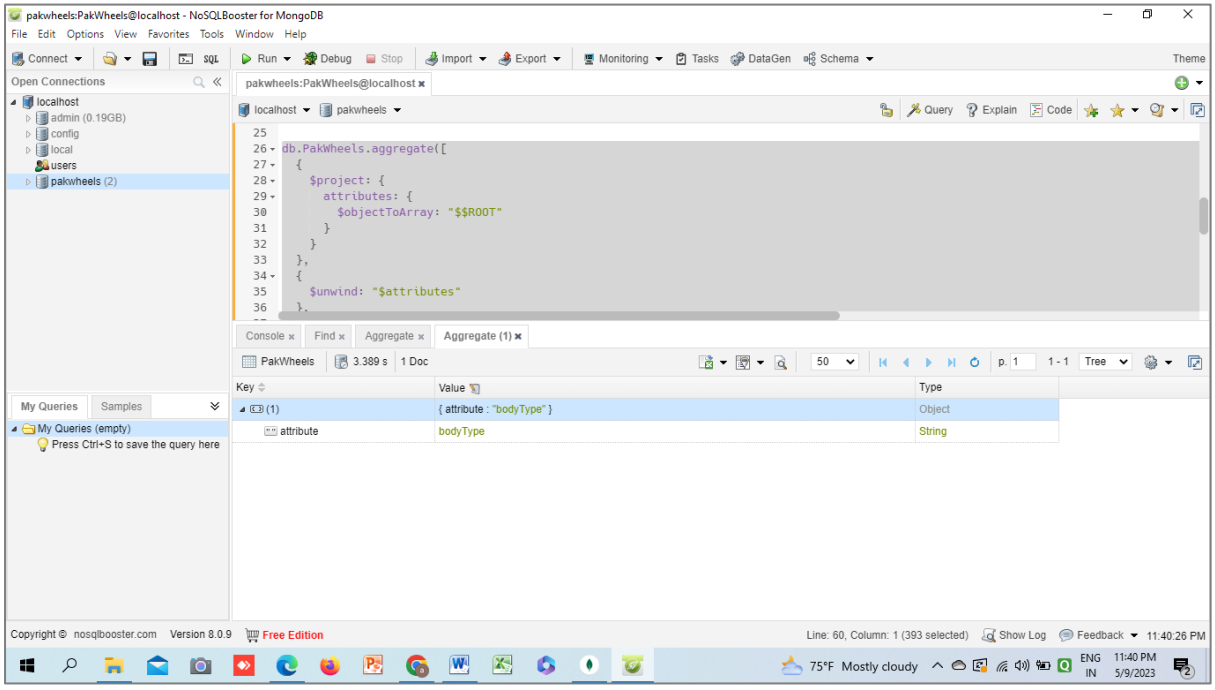
2.5. List some of the attributes (field/properties) of your database which are not common among all the documents

We used the following query to find out the not common attributes among all the documents in the database:

```
db.PakWheels.aggregate([
  {
    $project: {
      attributes: {
        $objectToArray: "$$ROOT"
      }
    }
  },
  {
    $unwind: "$attributes"
  },
  {
    $group: {
      _id: "$attributes.k",
      count: {
        $sum: 1
      }
    }
  },
  {
    $match: {
```

```
      count: {
        $ne: 55675
      }
    }
  },
  {
    $project: {
      attribute: "$_id",
      _id: 0
    }
  }
]);
```

The below screenshot is showing the output of the above query:



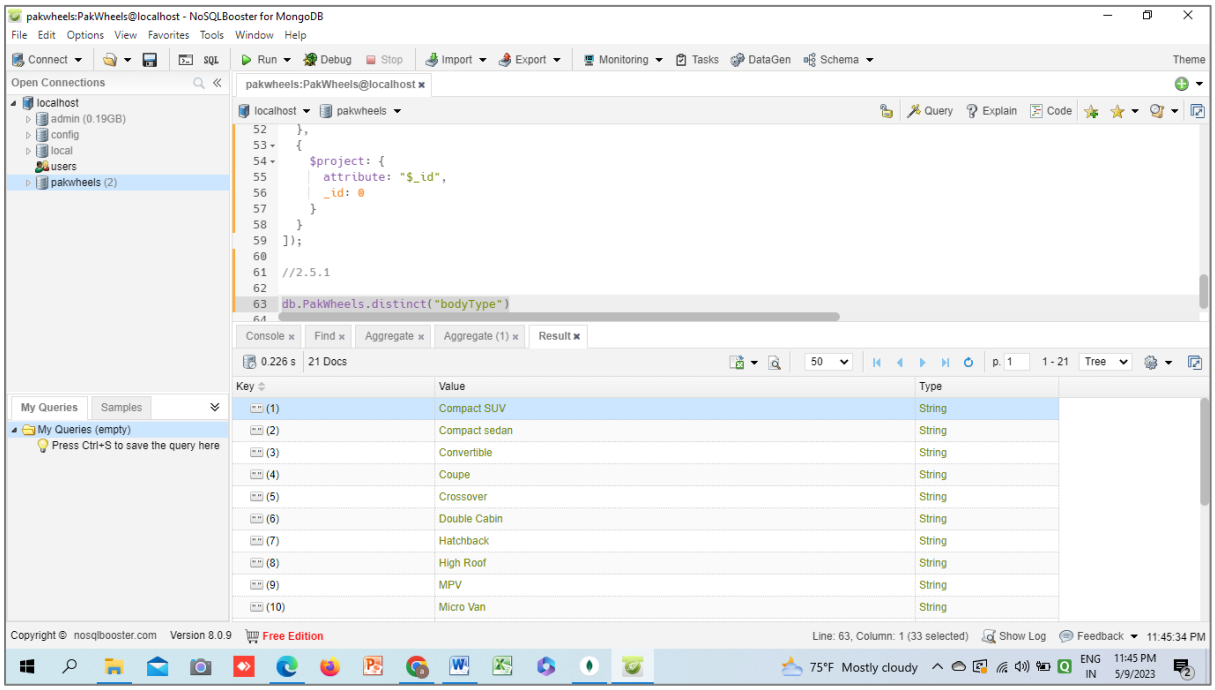
Using this we get "***bodyType***" as an attribute which is not common in all documents.

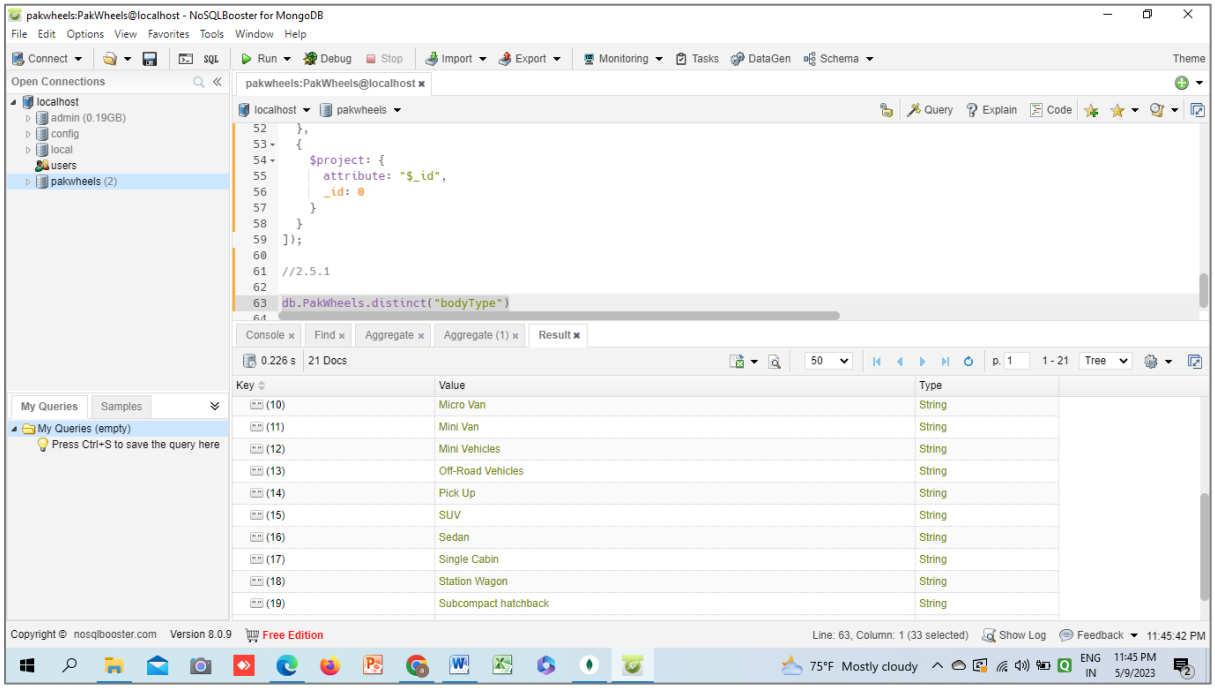
2.5.1. For these fields, provide some of the values they contain in the database

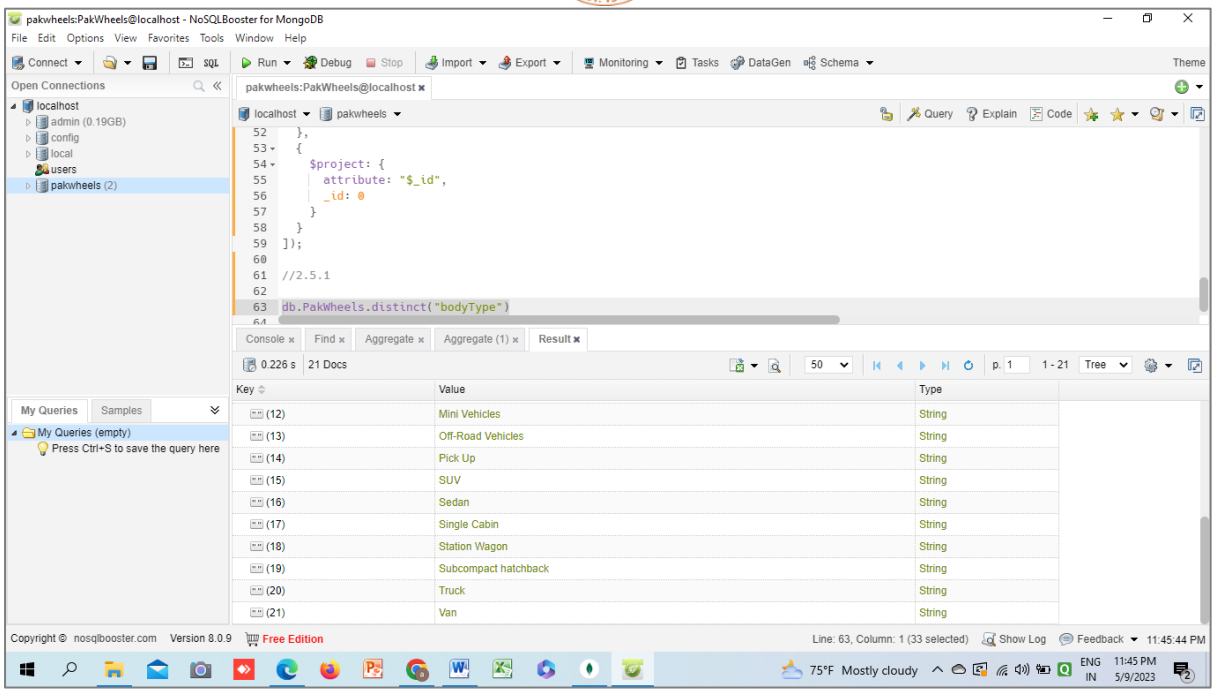
As we know that “bodyType” attribute is the uncommon one among all the documents, some of the values it contains in the database are as follows with screenshots attached:

Total 21 values are inside the “bodyType” attribute some of them are as follows: Compact SUV, Compact Sedan, Convertible, Coupe , Crossover , Double Cabin, Hatchback, High Roof, MPV , Micro Van, Mini-Van, Mini Vehicles, Off-Road Vehicles, Pick Up, SUV , Sedan, Van, Truck, and so on.

The below screenshots shows the output of query **db.PakWheels.distinct("bodyType")**







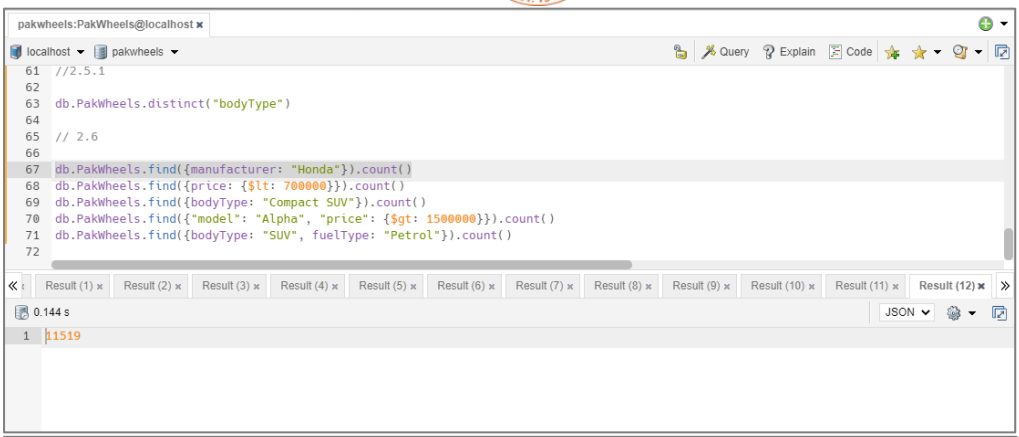
2.6. Write at least 5 queries using the key-value pairs you found in the previous steps to narrow down the result (provide the queries and results in your report)

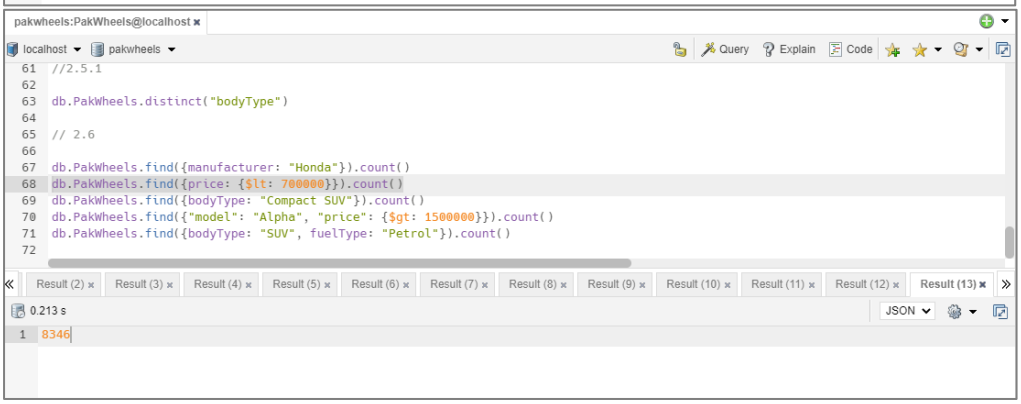
For 2.6 we can use these 5 queries

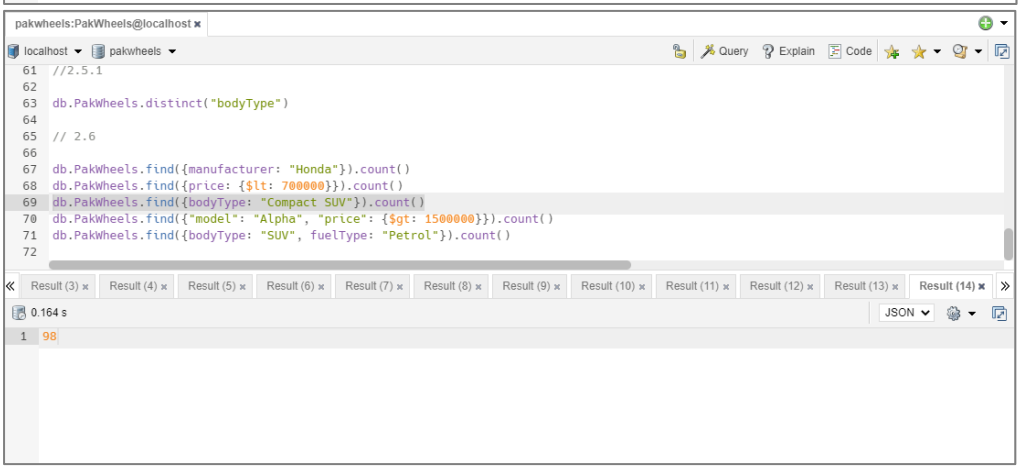
```
db.PakWheels.find({manufacturer: "Honda"}).count()
db.PakWheels.find({price: {$lt: 700000}}).count()
db.PakWheels.find({bodyType: "Compact SUV"}).count()
db.PakWheels.find({"model": "Alpha", "price": {$gt: 1500000}}).count()
db.PakWheels.find({bodyType: "SUV", fuelType: "Petrol"}).count()
```

1. Count of Manufacturer as Honda is **11519.**
2. Count of price less than 700000 are **8346.**
3. Count of bodyType as “Compact SUV” is **98.**
4. Count of model as Alpha and price greater than 1500000 is **1.**
5. Count of bodyType as SUV and fuelType as Petrol is **2766.**

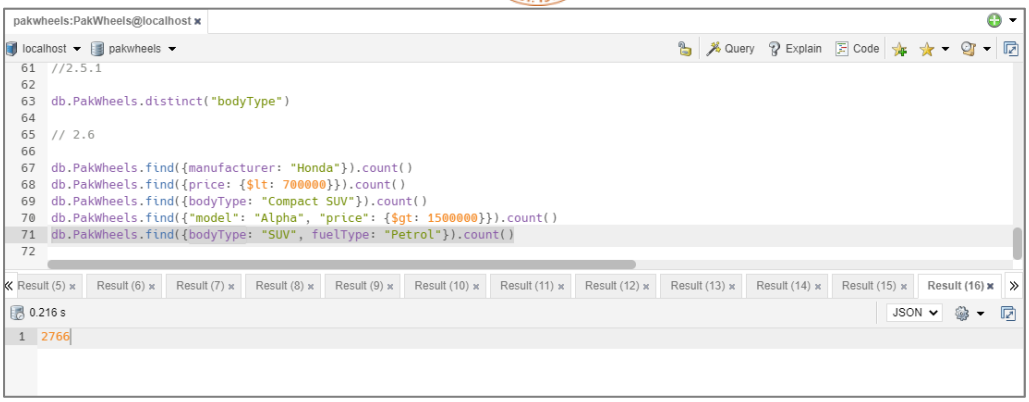
The below screenshots shows the output of above 5 queries:


pakwheels:PakWheels@localhost
localhost
pakwheels
Query
Explain
Code
61 //2.5.1
62
63 db.PakWheels.distinct("bodyType")
64
65 // 2.6
66
67 db.PakWheels.find({manufacturer: "Honda"}).count()
68 db.PakWheels.find({price: {$lt: 700000}}).count()
69 db.PakWheels.find({bodyType: "Compact SUV"}).count()
70 db.PakWheels.find({"model": "Alpha", "price": {$gt: 1500000}}).count()
71 db.PakWheels.find({bodyType: "SUV", fuelType: "Petrol"}).count()
72
Result (1)
Result (2)
Result (3)
Result (4)
Result (5)
Result (6)
Result (7)
Result (8)
Result (9)
Result (10)
Result (11)
Result (12)
0.144 s
JSON
1 11519


pakwheels:PakWheels@localhost
localhost
pakwheels
Query
Explain
Code
61 //2.5.1
62
63 db.PakWheels.distinct("bodyType")
64
65 // 2.6
66
67 db.PakWheels.find({manufacturer: "Honda"}).count()
68 db.PakWheels.find({price: {$lt: 700000}}).count()
69 db.PakWheels.find({bodyType: "Compact SUV"}).count()
70 db.PakWheels.find({"model": "Alpha", "price": {$gt: 1500000}}).count()
71 db.PakWheels.find({bodyType: "SUV", fuelType: "Petrol"}).count()
72
Result (2)
Result (3)
Result (4)
Result (5)
Result (6)
Result (7)
Result (8)
Result (9)
Result (10)
Result (11)
Result (12)
Result (13)
0.213 s
JSON
1 8346


pakwheels:PakWheels@localhost
localhost
pakwheels
Query
Explain
Code
61 //2.5.1
62
63 db.PakWheels.distinct("bodyType")
64
65 // 2.6
66
67 db.PakWheels.find({manufacturer: "Honda"}).count()
68 db.PakWheels.find({price: {$lt: 700000}}).count()
69 db.PakWheels.find({bodyType: "Compact SUV"}).count()
70 db.PakWheels.find({"model": "Alpha", "price": {$gt: 1500000}}).count()
71 db.PakWheels.find({bodyType: "SUV", fuelType: "Petrol"}).count()
72
Result (3)
Result (4)
Result (5)
Result (6)
Result (7)
Result (8)
Result (9)
Result (10)
Result (11)
Result (12)
Result (13)
Result (14)
0.164 s
JSON
1 98

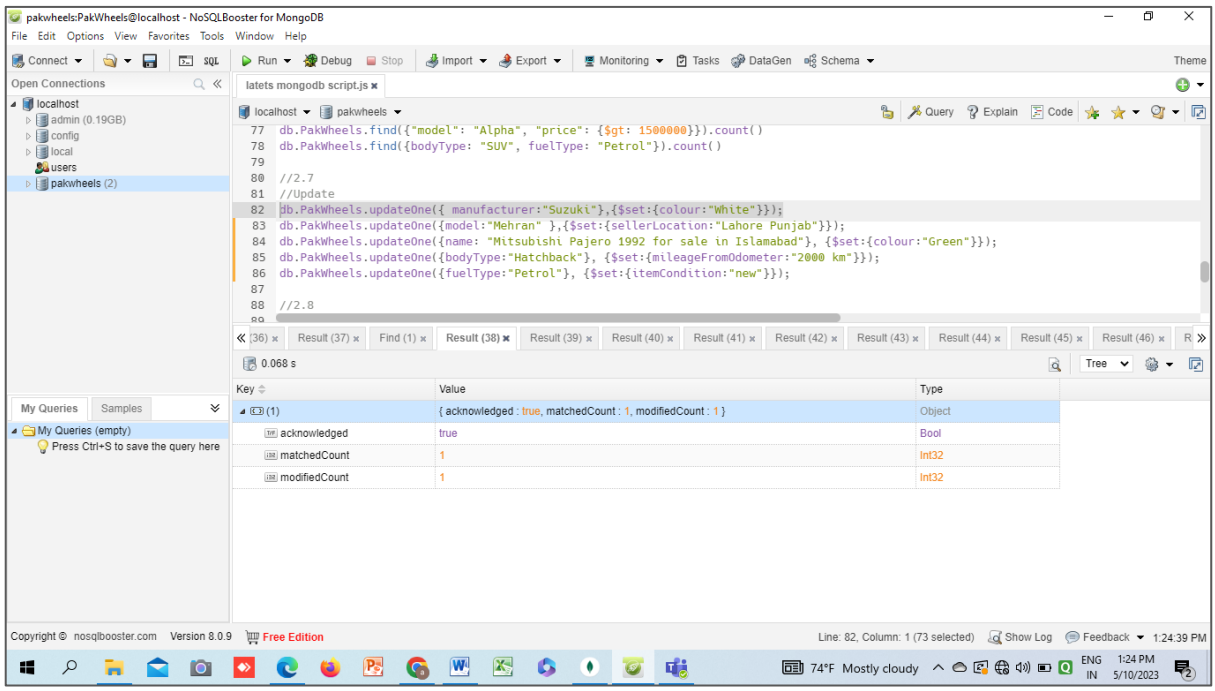

pakwheels:PakWheels@localhost
localhost
pakwheels
Query
Explain
Code
61 //2.5.1
62
63 db.PakWheels.distinct("bodyType")
64
65 // 2.6
66
67 db.PakWheels.find({manufacturer: "Honda"}).count()
68 db.PakWheels.find({price: {$lt: 700000}}).count()
69 db.PakWheels.find({bodyType: "Compact SUV"}).count()
70 db.PakWheels.find({"model": "Alpha", "price": {$gt: 1500000}}).count()
71 db.PakWheels.find({bodyType: "SUV", fuelType: "Petrol"}).count()
72
Result (4)
Result (5)
Result (6)
Result (7)
Result (8)
Result (9)
Result (10)
Result (11)
Result (12)
Result (13)
Result (14)
Result (15)
0.188 s
JSON
1 1



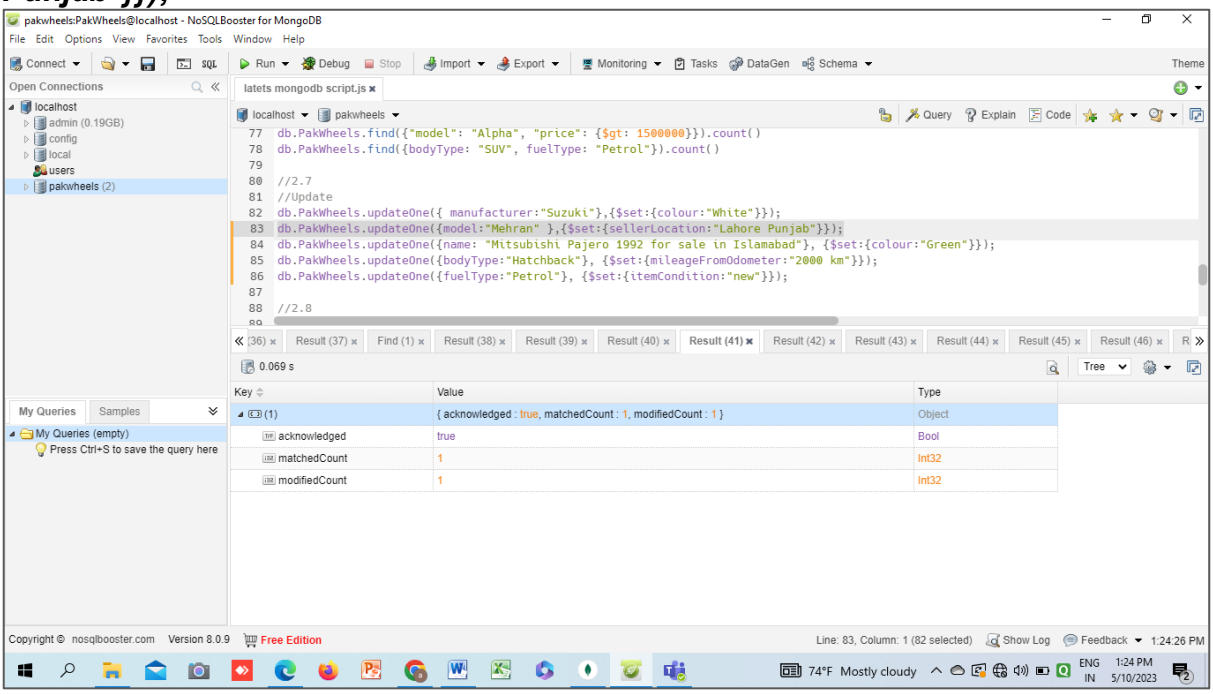
2.7. Write at least 5 update queries to update some of the values in your database (provide the queries and results in your report)

Below are screenshots along with the update queries.

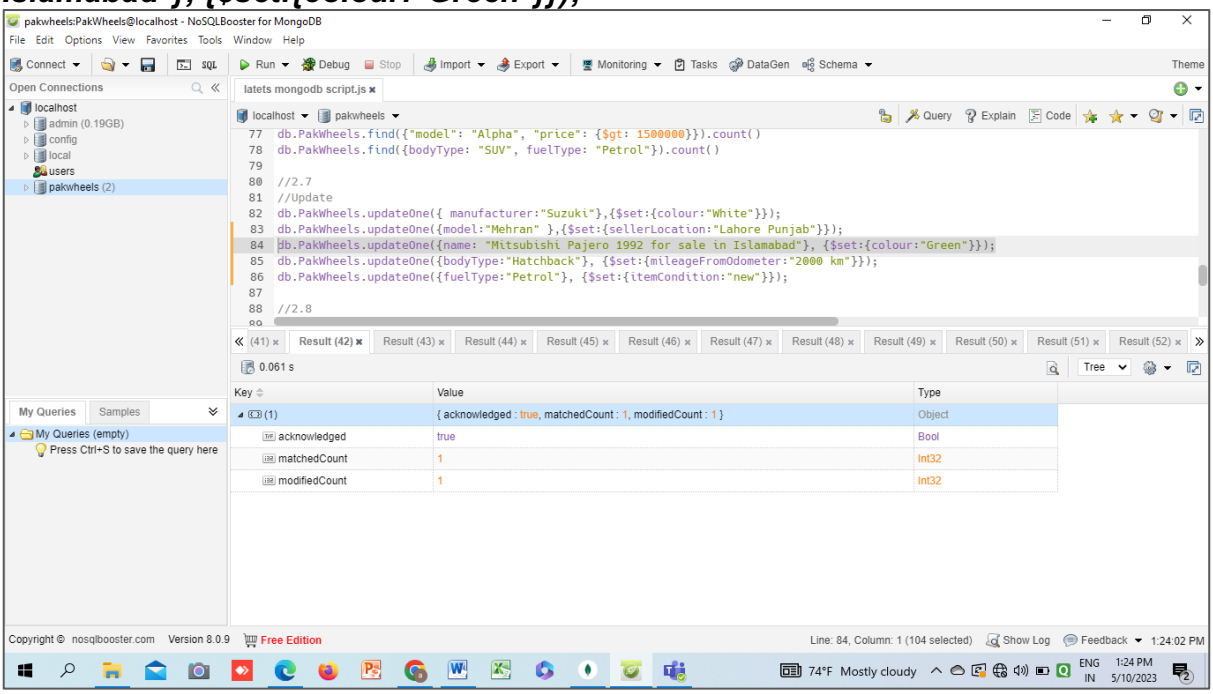
db.PakWheels.updateOne({ manufacturer:"Suzuki"},{$set:{colour:"White"}});



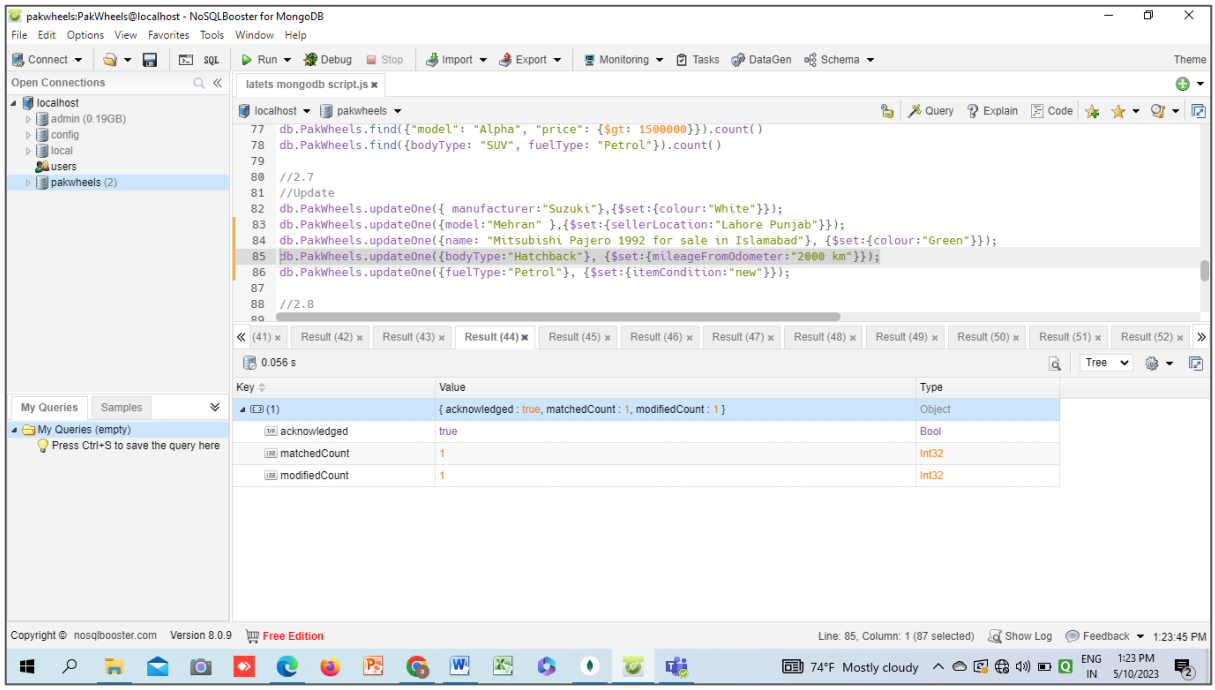
db.PakWheels.updateOne({model:"Mehran" },{$set:{sellerLocation:"Lahore Punjab"}});



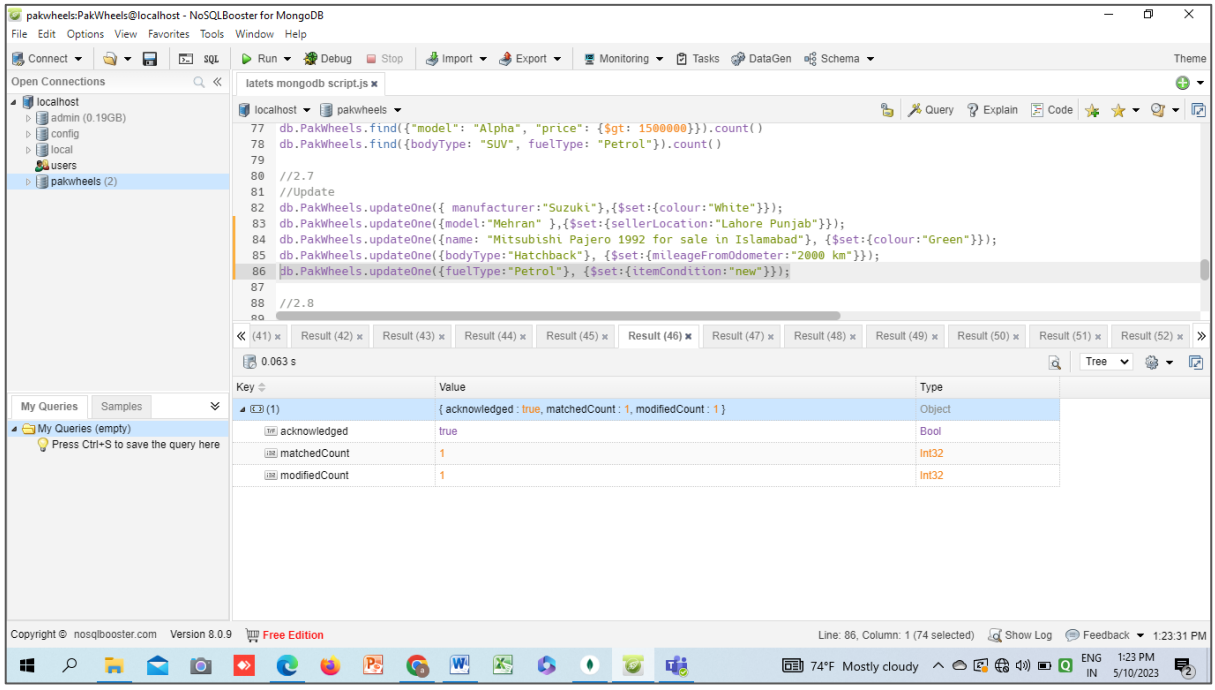
db.PakWheels.updateOne({name: "Mitsubishi Pajero 1992 for sale in Islamabad"}, {$set:{colour:"Green"}});



db.PakWheels.updateOne({bodyType:"Hatchback"}, {$set:{mileageFromOdometer:"2000 km"}});



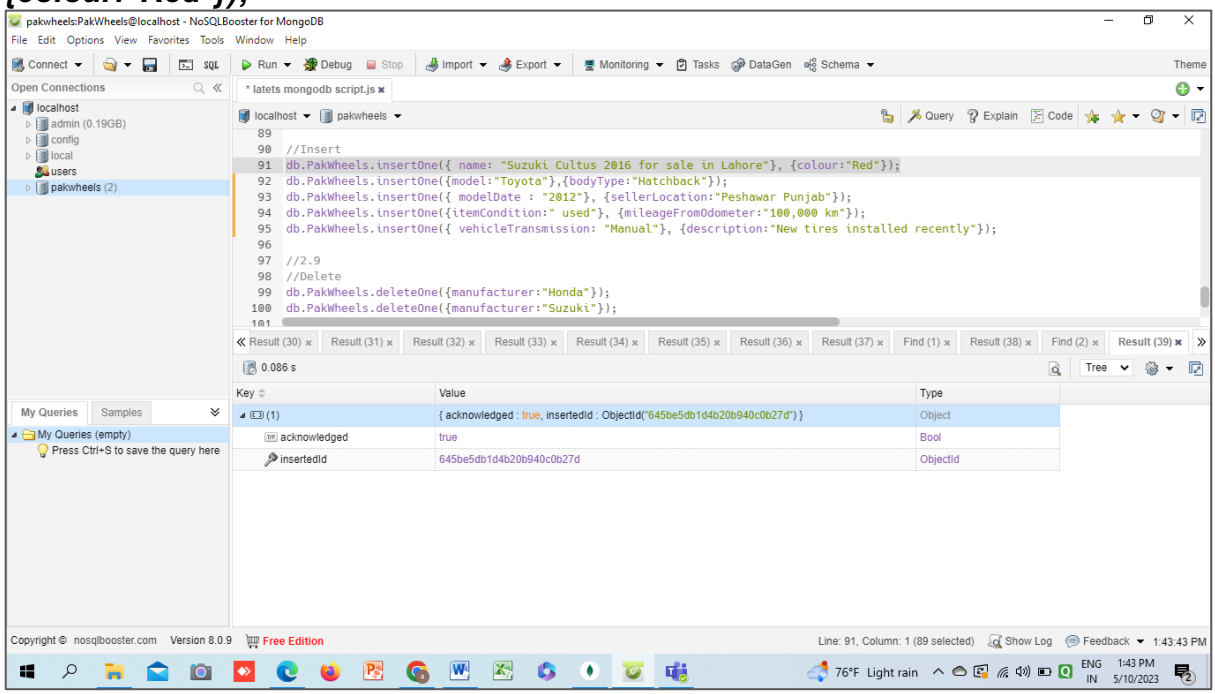
db.PakWheels.updateOne({fuelType:"Petrol"}, {$set:{itemCondition:"new"}});



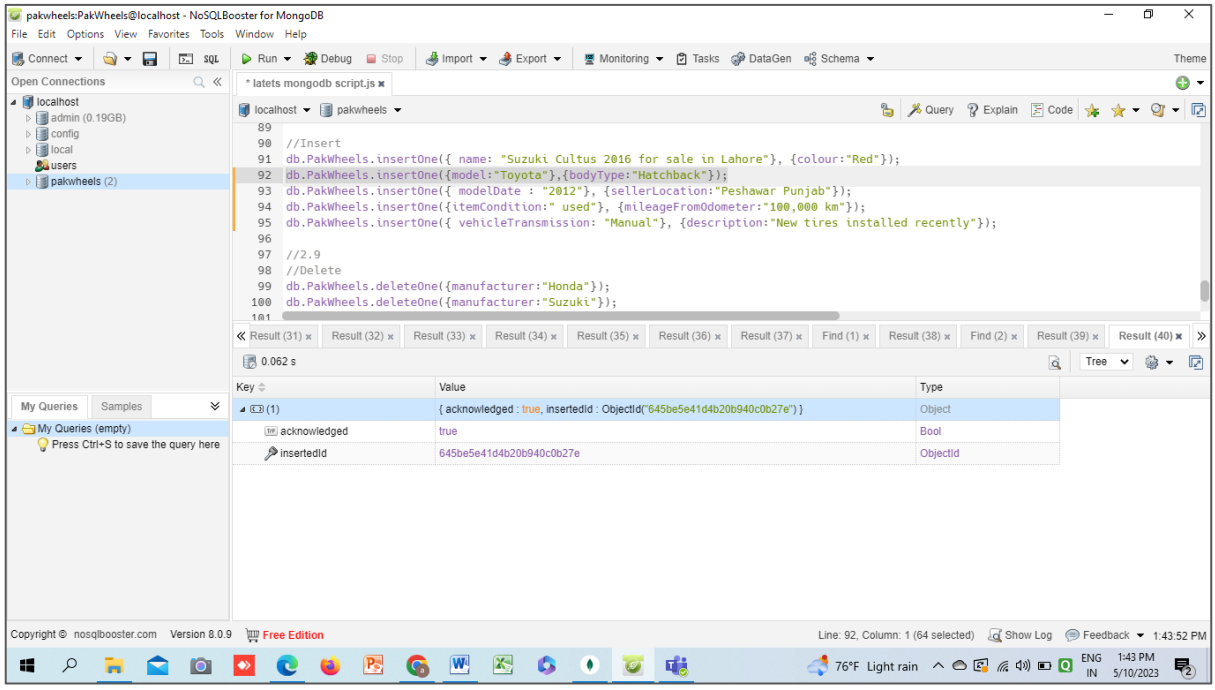
2.8. Write at least 5 queries to insert new documents into your database (provide the queries and results in your report)

Below are screenshots along with the insert queries.

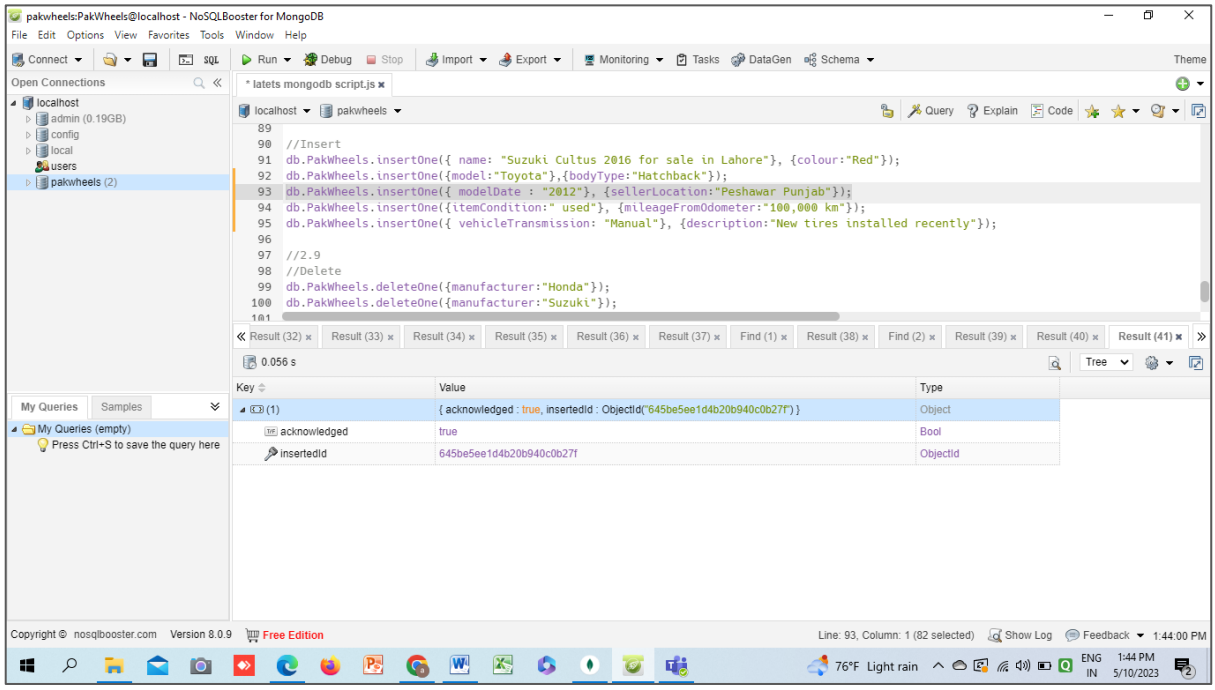
db.PakWheels.insertOne({ name: "Suzuki Cultus 2016 for sale in Lahore"}, {colour:"Red"});



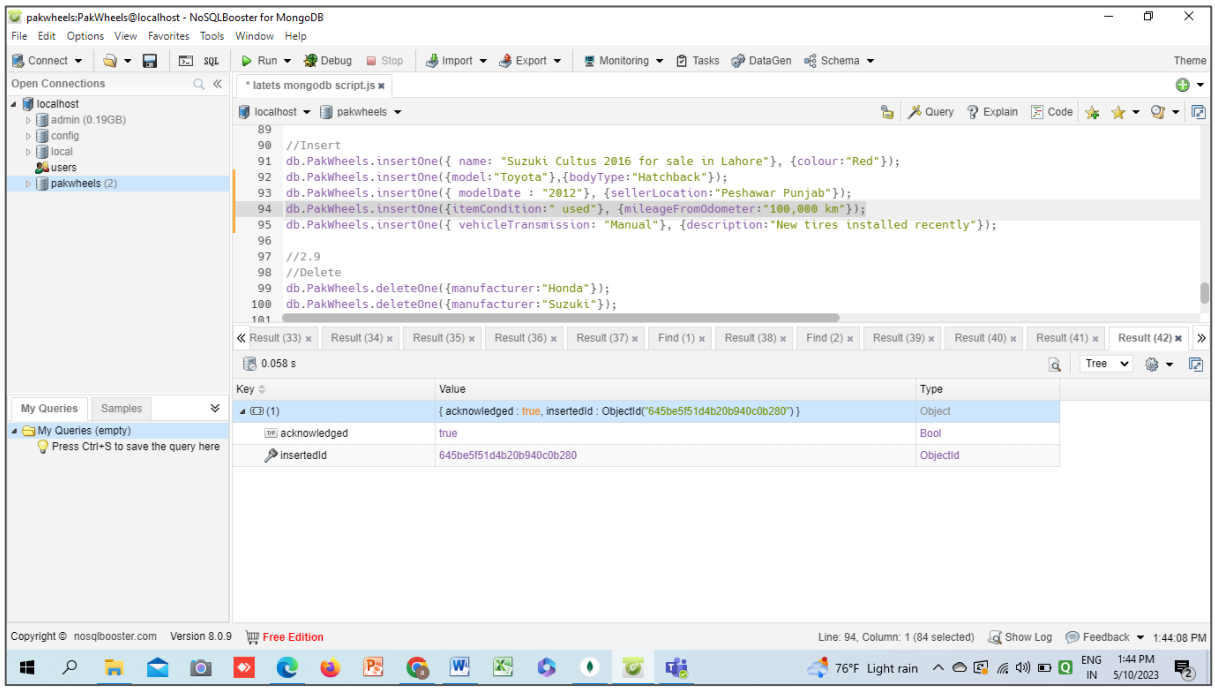
db.PakWheels.insertOne({model:"Toyota"},{bodyType:"Hatchback"});



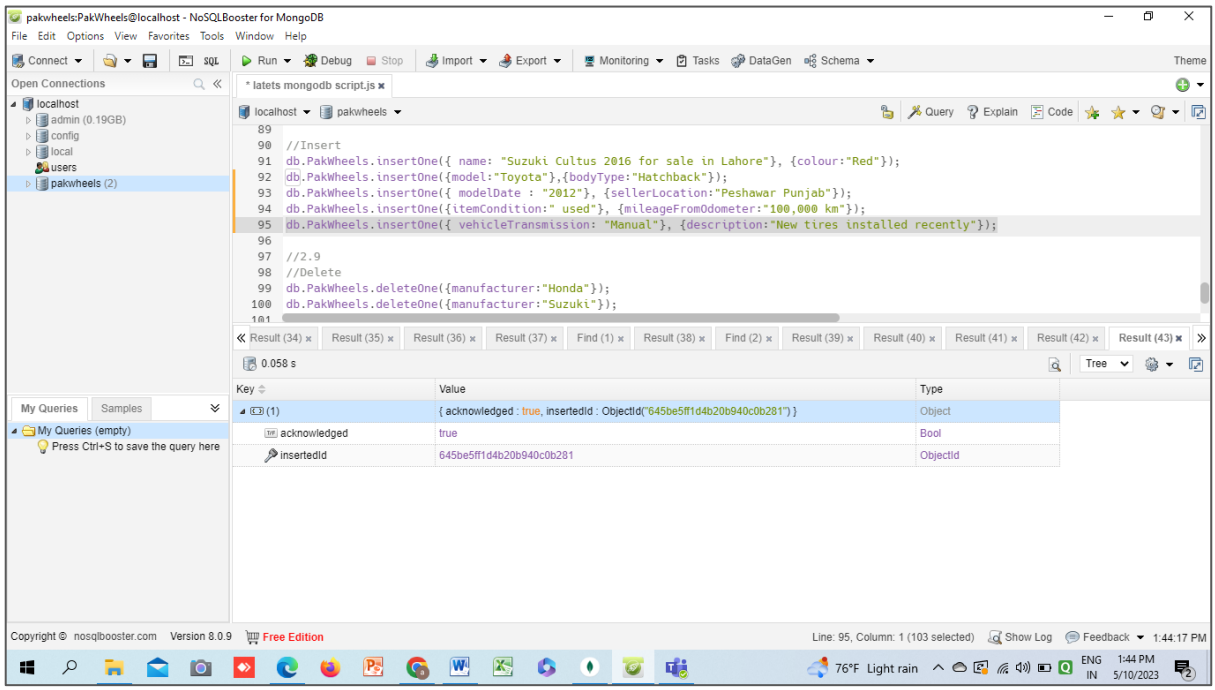
db.PakWheels.insertOne({ modelDate : "2012"}, {sellerLocation:"Peshawar Punjab"});



db.PakWheels.insertOne({itemCondition:" used"}, {mileageFromOdometer:"100,000 km"});



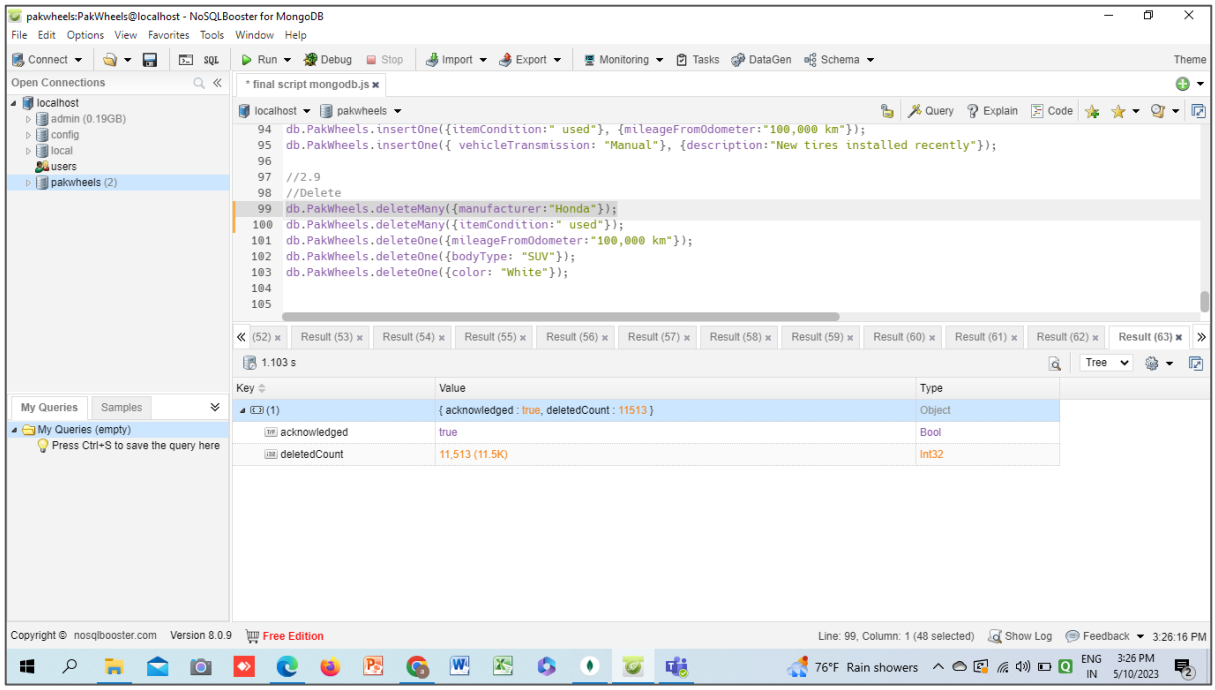
db.PakWheels.insertOne({ vehicleTransmission: "Manual"}, {description:"New tires installed recently"});



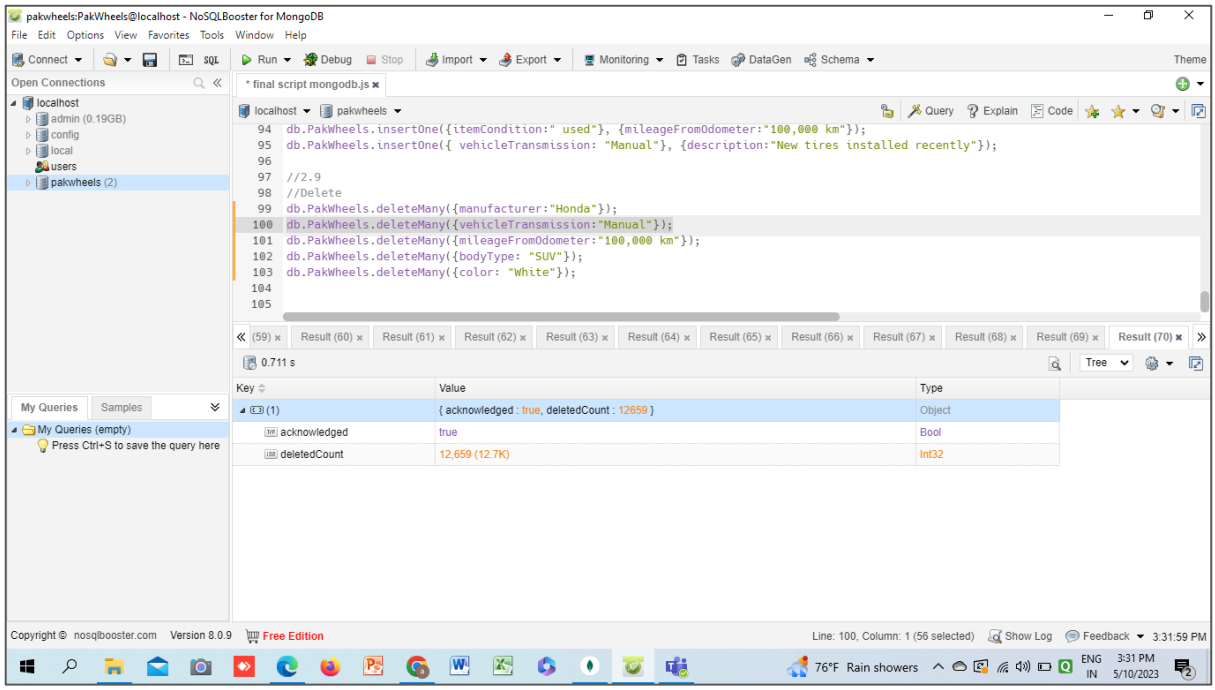
2.9. Write at least 5 delete queries to remove documents from your database (provide the queries and results in your report)

Below are screenshots along with the delete queries.

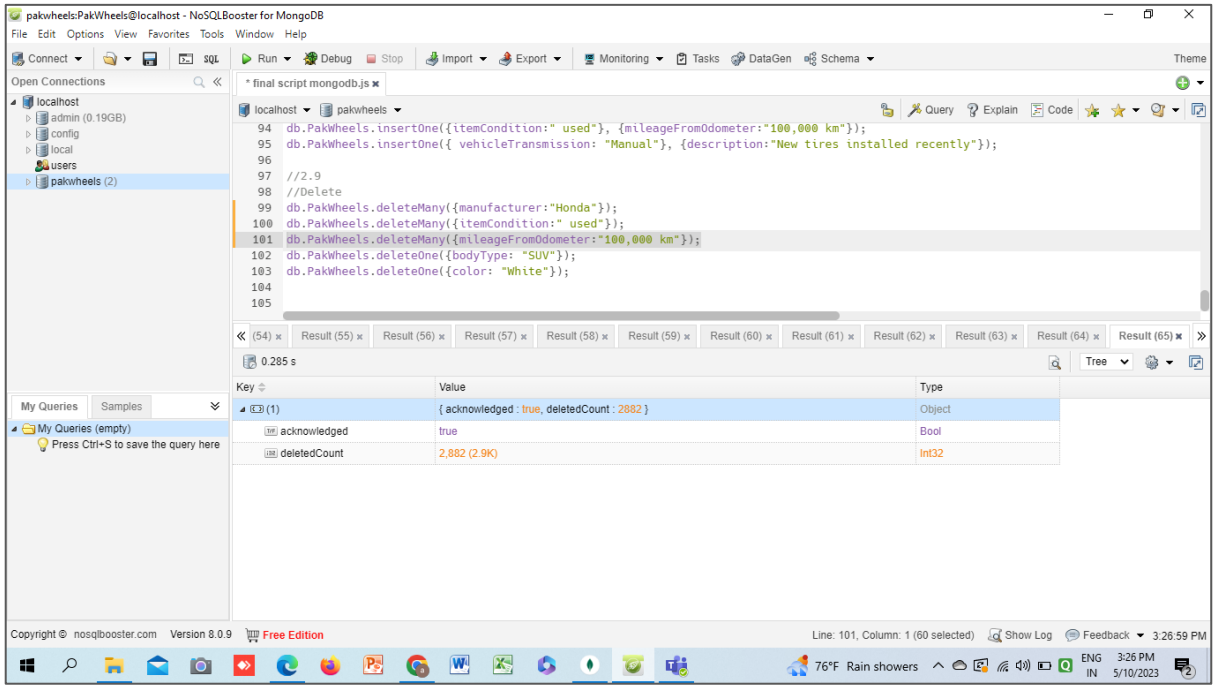
db.PakWheels.deleteMany({manufacturer:"Honda"});



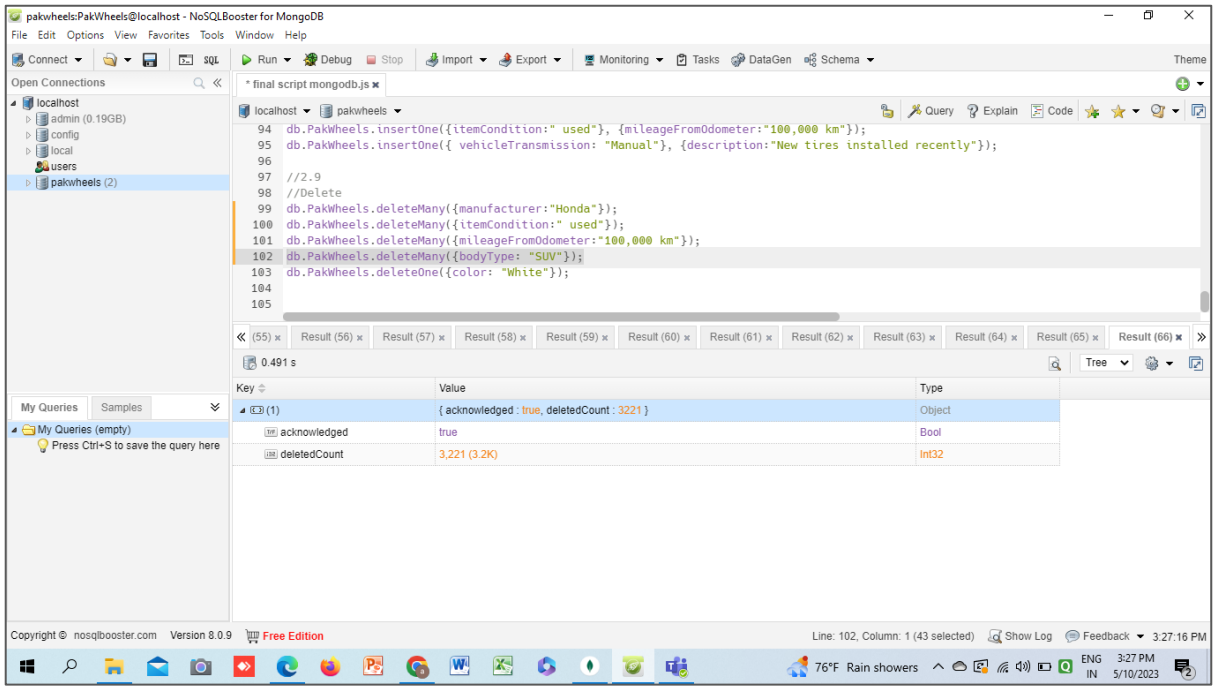
db.PakWheels.deleteMany({vehicleTransmission:"Manual"});



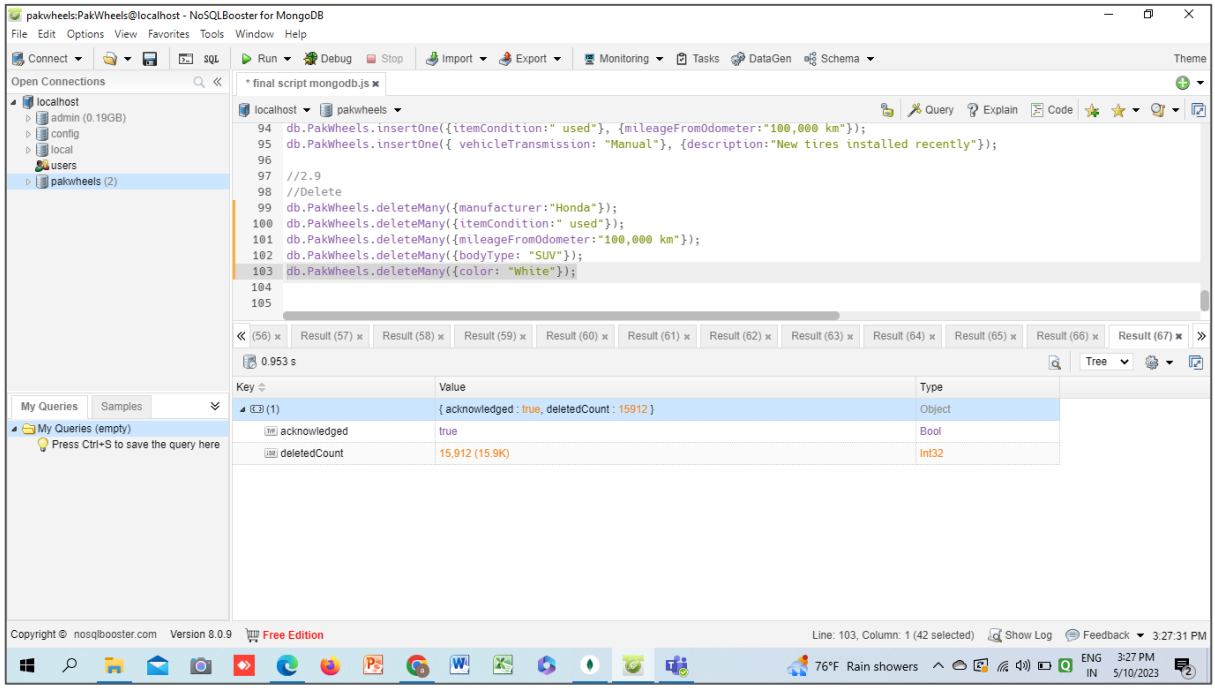
db.PakWheels.deleteMany({mileageFromOdometer:"100,000 km"});



db.PakWheels.deleteMany({bodyType: "SUV"});



db.PakWheels.deleteMany({color: "White"});



*****End of Document*****