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.

```
1 •  SELECT DISTINCT
2        TABLE_NAME,
3        INDEX_NAME,
4        INDEX_TYPE
5    FROM INFORMATION_SCHEMA.STATISTICS
6    WHERE TABLE_SCHEMA = 'new_schema';
7
```

100% 14:13

Result Grid Filter Rows: Search Export:

TABLE_NAME	INDEX_NAME	INDEX_TYPE
Accident	Location_Id_idx	BTREE
Accident	PRIMARY	BTREE
City	City	BTREE
GPS	PRIMARY	BTREE
Location	idx_city	BTREE
Location	PRIMARY	BTREE
Weather	PRIMARY	BTREE

There are 5 tables in our database and indexes for each table is shown below:(name the table for each snippet)

Table Accident:

```
10 •  SHOW index
11    from Accident
12    from new_schema;
13
```

100% 17:12

Result Grid Filter Rows: Search Export:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Accident	0	PRIMARY	1	ID	A	0	NULL	NULL		BTREE			YES	NULL
Accident	1	Location_Id_idx	1	Location_Id	A	0	NULL	NULL	YES	BTREE			YES	NULL

Table GPS:

```
18 •  SHOW index
19    from GPS
20    from new_schema;
21
```

100% 17:20

Result Grid Filter Rows: Search Export:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
GPS	0	PRIMARY	1	Location_id	A	0	NULL	NULL		BTREE			YES	NULL

Table Location:

```
22 •  SHOW index
23    from Location
24    from new_schema;
25
```

100% 17:24

Result Grid Filter Rows: Search Export:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Location	0	PRIMARY	1	Location_id	A	961	NULL	NULL		BTREE			YES	NULL
Location	1	idx_city	1	City	A	0	NULL	NULL	YES	BTREE			YES	NULL

Table City:

```
14 •  SHOW index
15    from City
16    from new_schema;
17
```

100% 17:16

Result Grid Filter Rows: Search Export:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
City	1	City	1	City	A	0	NULL	NULL		BTREE			YES	NULL

Table Weather:

```
26 •  SHOW index
27    from Weather
28    from new_schema;
```

100% 17:28

Result Grid | Filter Rows: Search | Export:

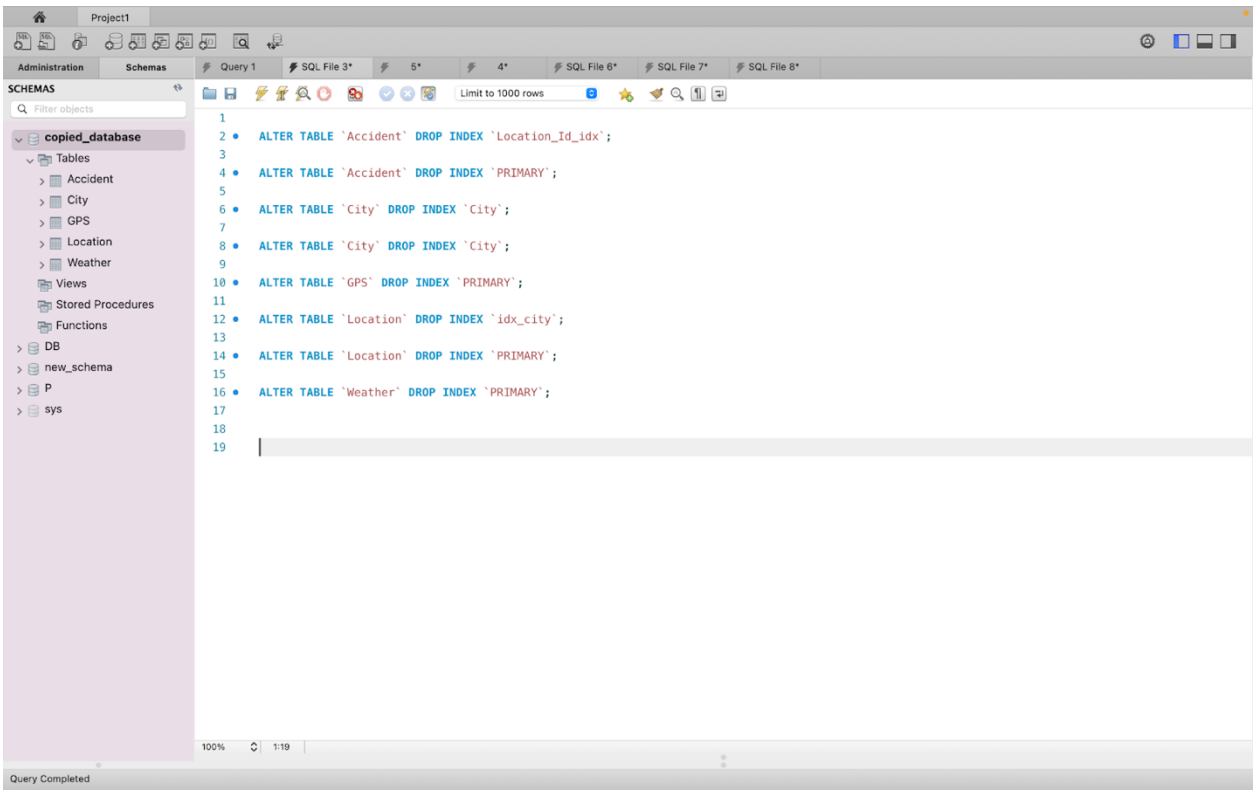
Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
Weather	0	PRIMARY	1	Location_id	A	1000	NULL	NULL		BTREE			YES	NULL
Weather	0	PRIMARY	2	Weather_Timestamp	A	0	NULL	NULL		BTREE			YES	NULL

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

Indexes in a database table allow for fast access to data by using one or more fields. By creating indexes based on specific fields, the database can quickly retrieve information either randomly or in an ordered fashion. Primary keys are automatically indexed in MySQL, and they are stored in B-trees. This means that primary keys are implicitly indexed, along with other types of indexes.

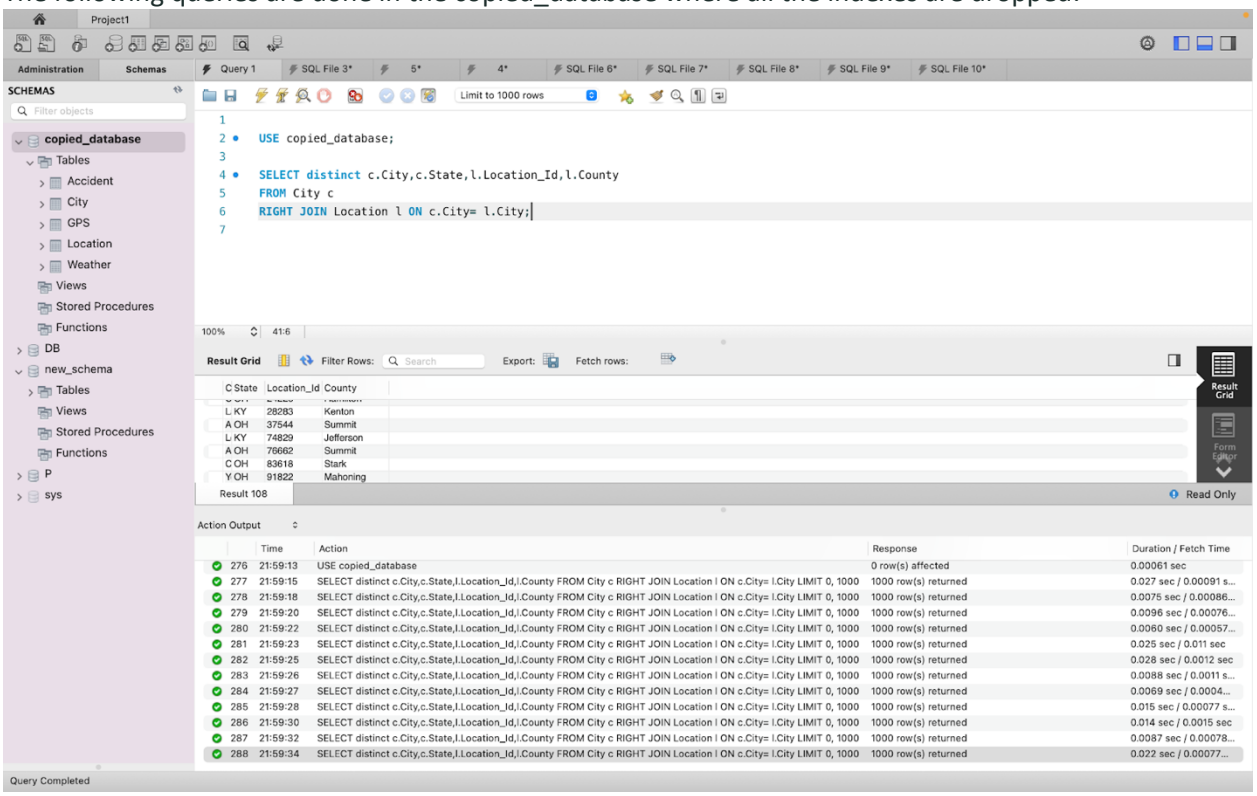
1.3. Make a copy of your database and delete all the indexes there (you might need to delete foreign keys before you can delete some of the indexes) – now you have two databases: database A with indexes and database B without any indexes.

Created a database called copied_database. Exported everything from the original database as a self-contained file(.sql) using:
Server->Data Export->Select database to export-> Export to self-contained file
Further, imported that file into the copied database using:
Server->Data Import->Import from a self-contained file-> Select Default Target Schema
Further, all the foreign keys (from table:city) are deleted so that we can delete all the indexes.

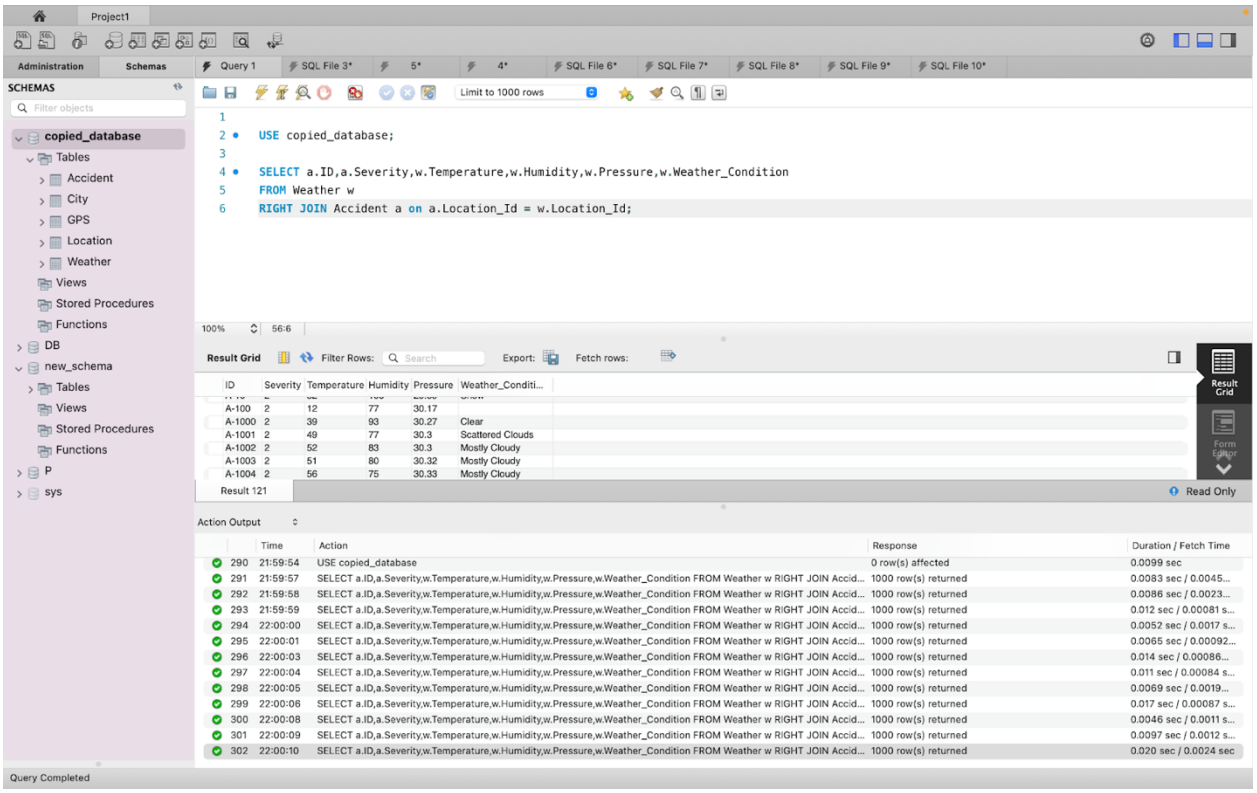


1.4. Write at least 5 queries (with JOINs between your tables)

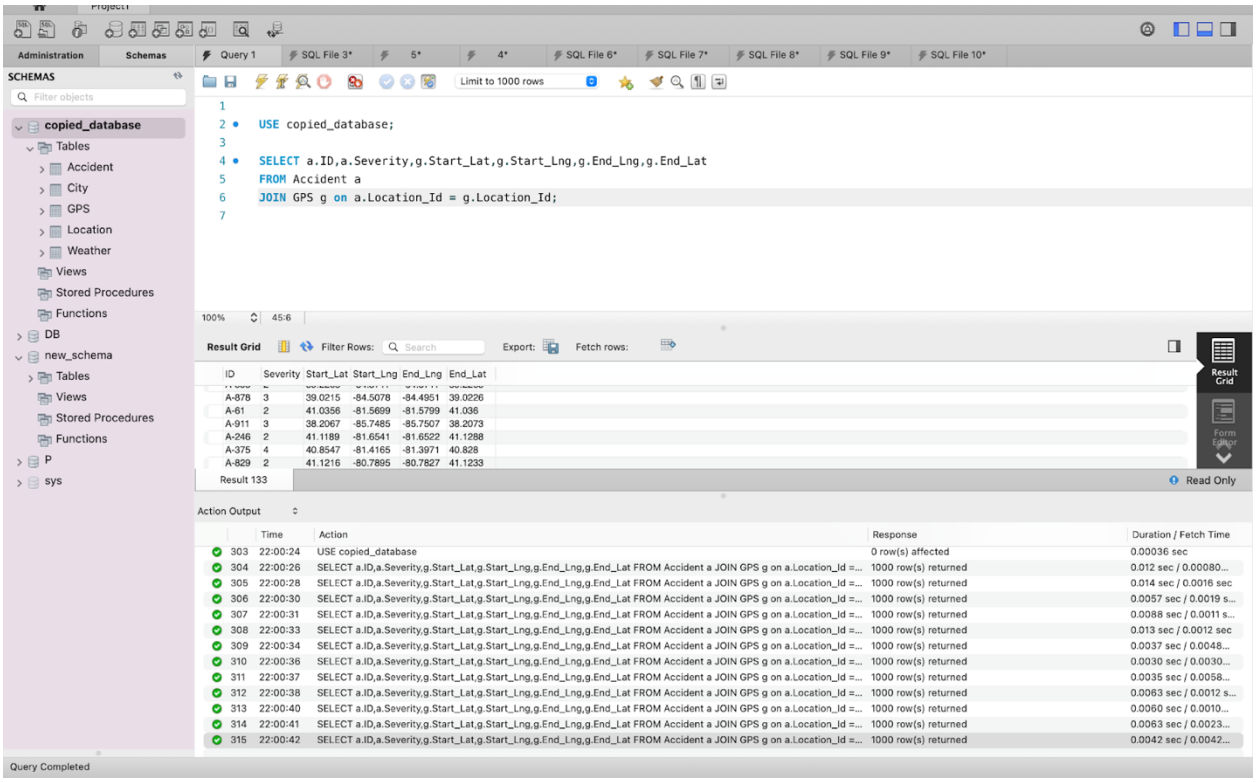
The SQL JOIN clause is utilized to access and retrieve data from multiple tables, based on logical connections between them. This clause specifies how SQL Server should select and combine entries from different tables, using information from other sources. Essentially, JOINS enable the retrieval of data from related tables in a database.
The following queries are done in the copied_database where all the indexes are dropped:



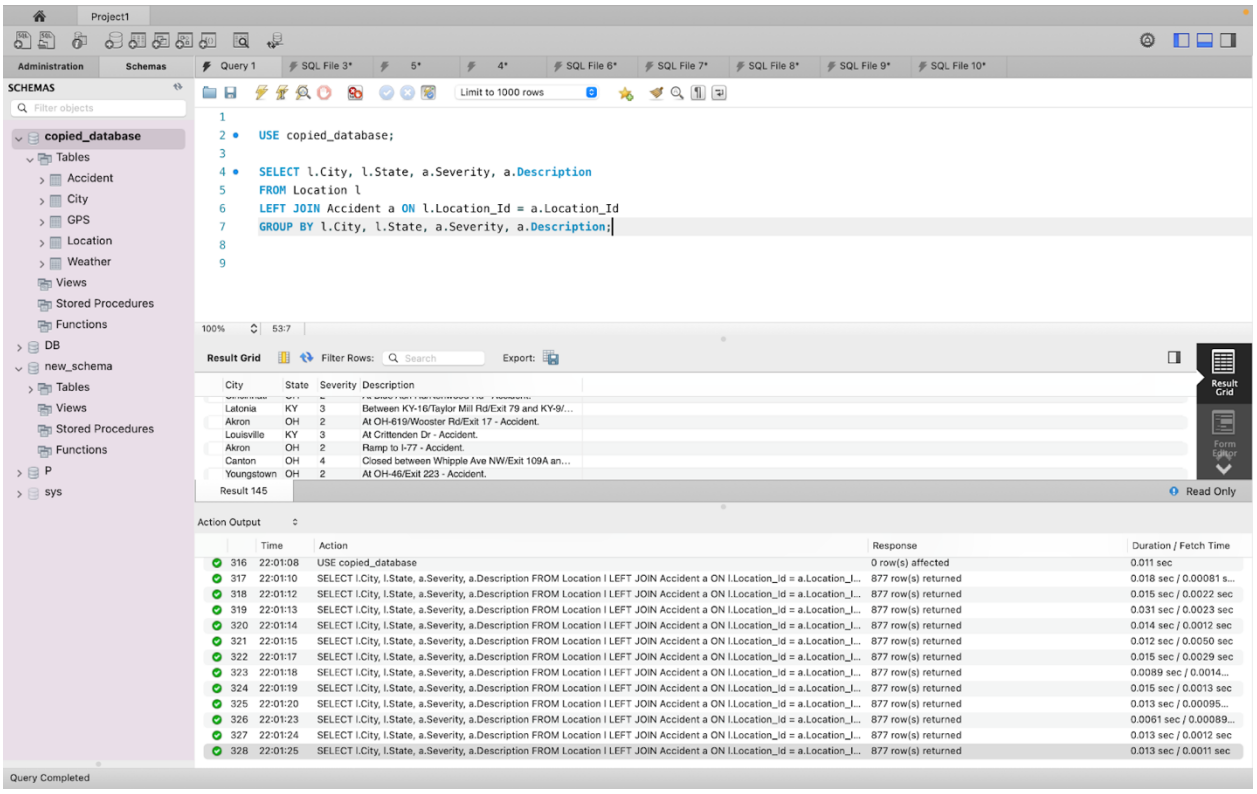
Average timing (Duration/Fetch) of this query run 10 times without index – 0.0167 sec



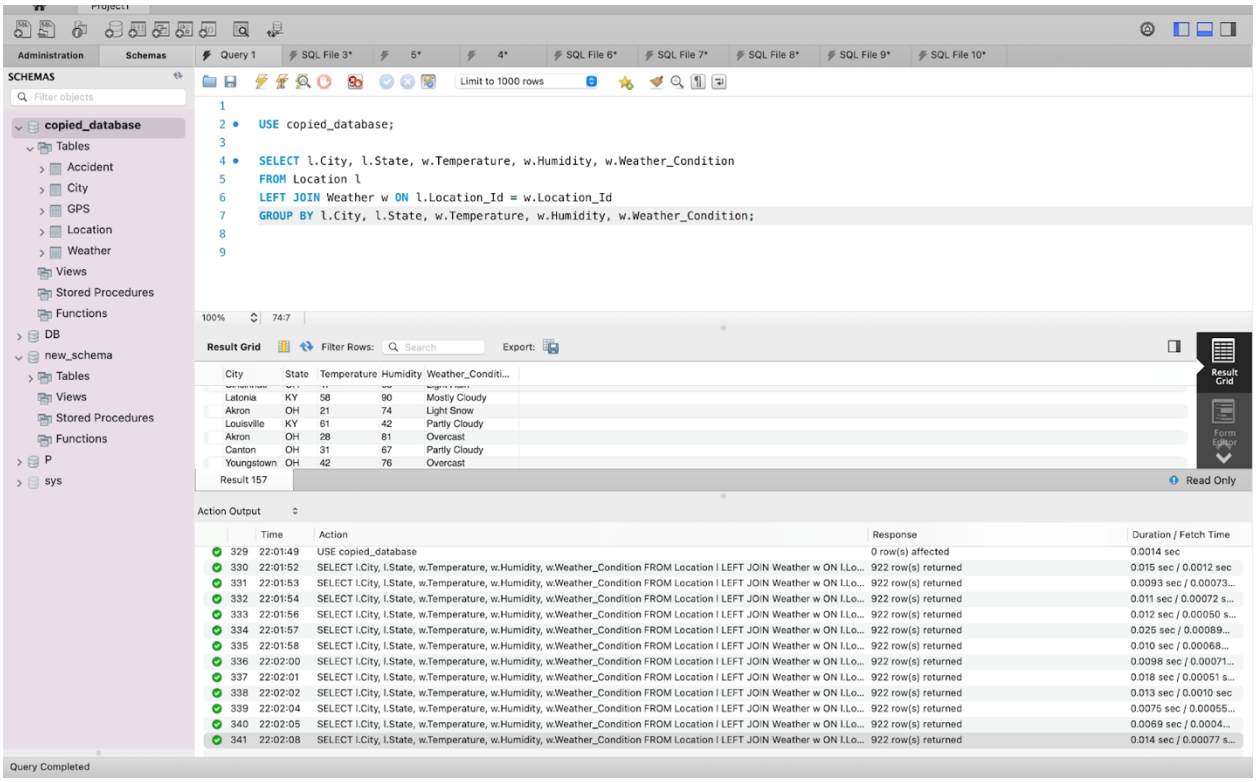
Average timing (Duration/Fetch) of this query run 10 times without index – 0.01069 sec



Average timing (Duration/Fetch) of this query run 10 times without index – 0.01202 sec

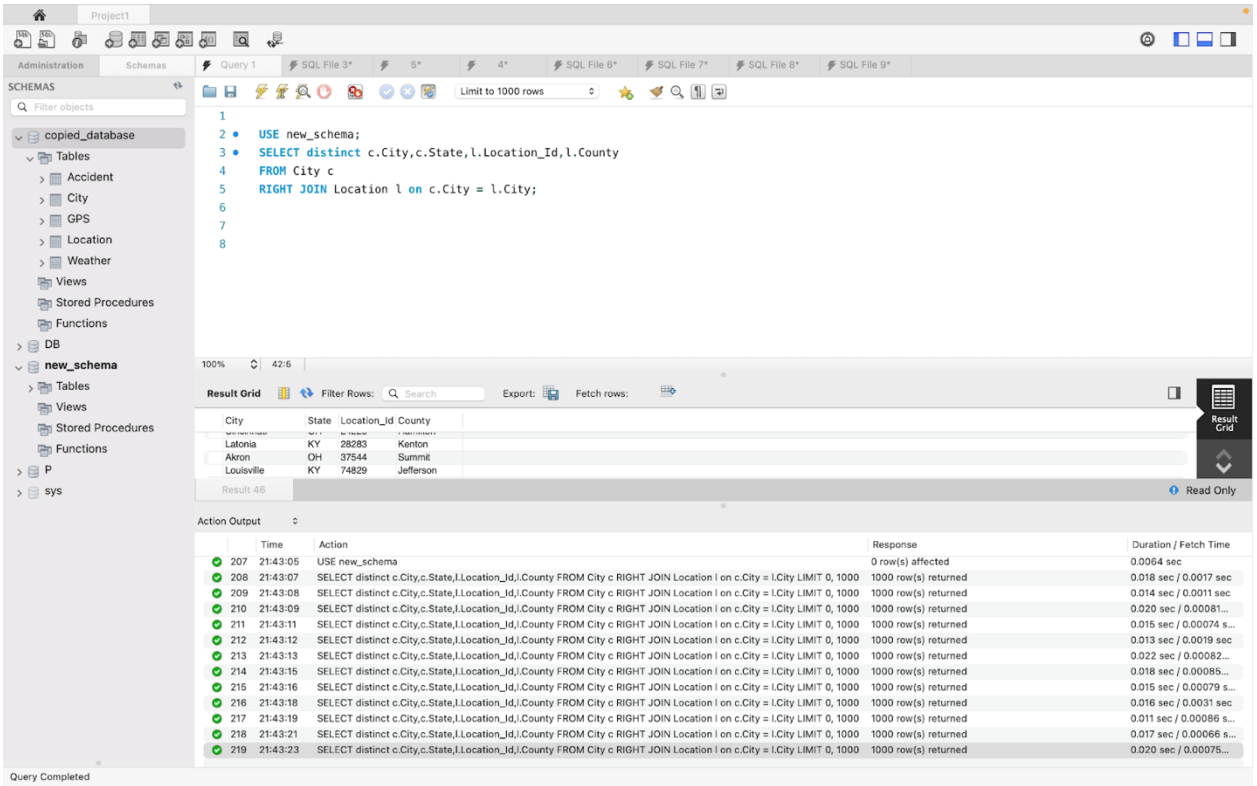


Average timing (Duration/Fetch) of this query run 10 times without index – 0.0237 sec

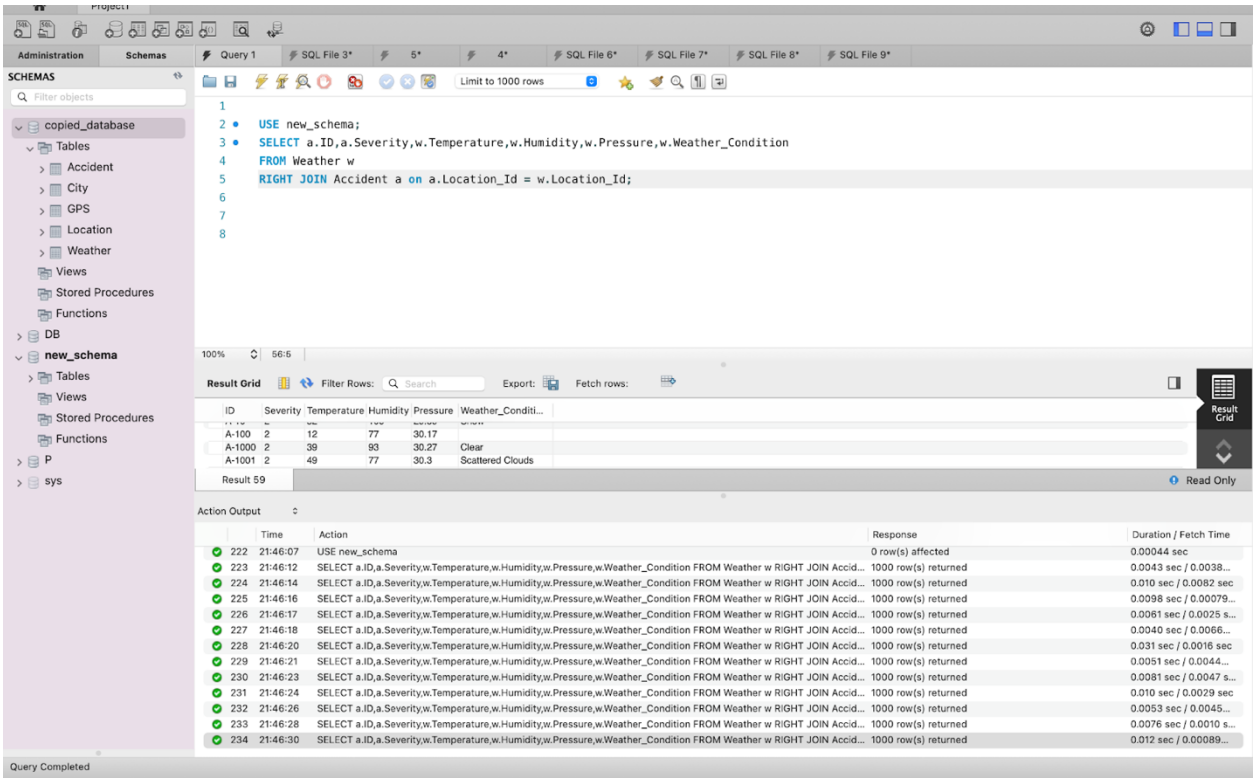


Average timing (Duration/Fetch) of this query run 10 times without index – 0.0216 sec

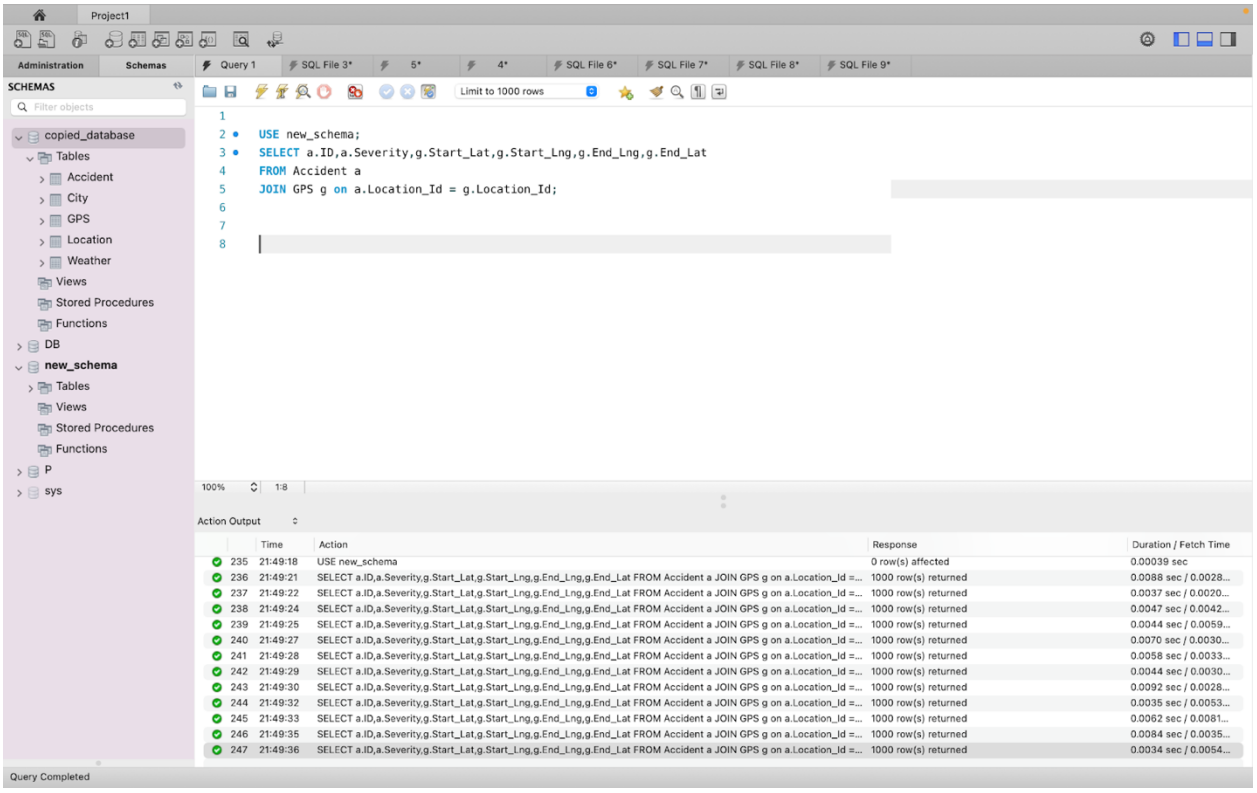
1.5. Execute and time these queries on both databases and report your findings (repeat timing for each query at least 10 times and average the times)



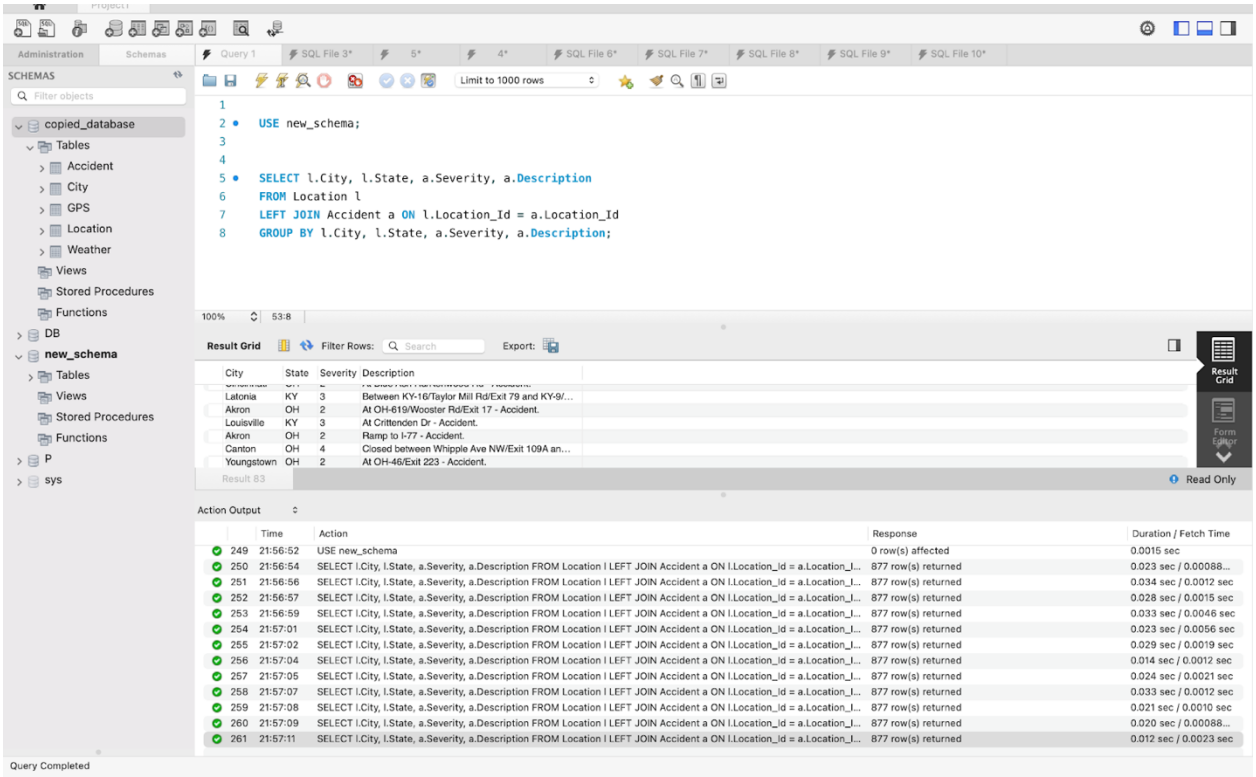
Average timing (Duration/Fetch) of this query run 10 times without index – 0.0144 sec



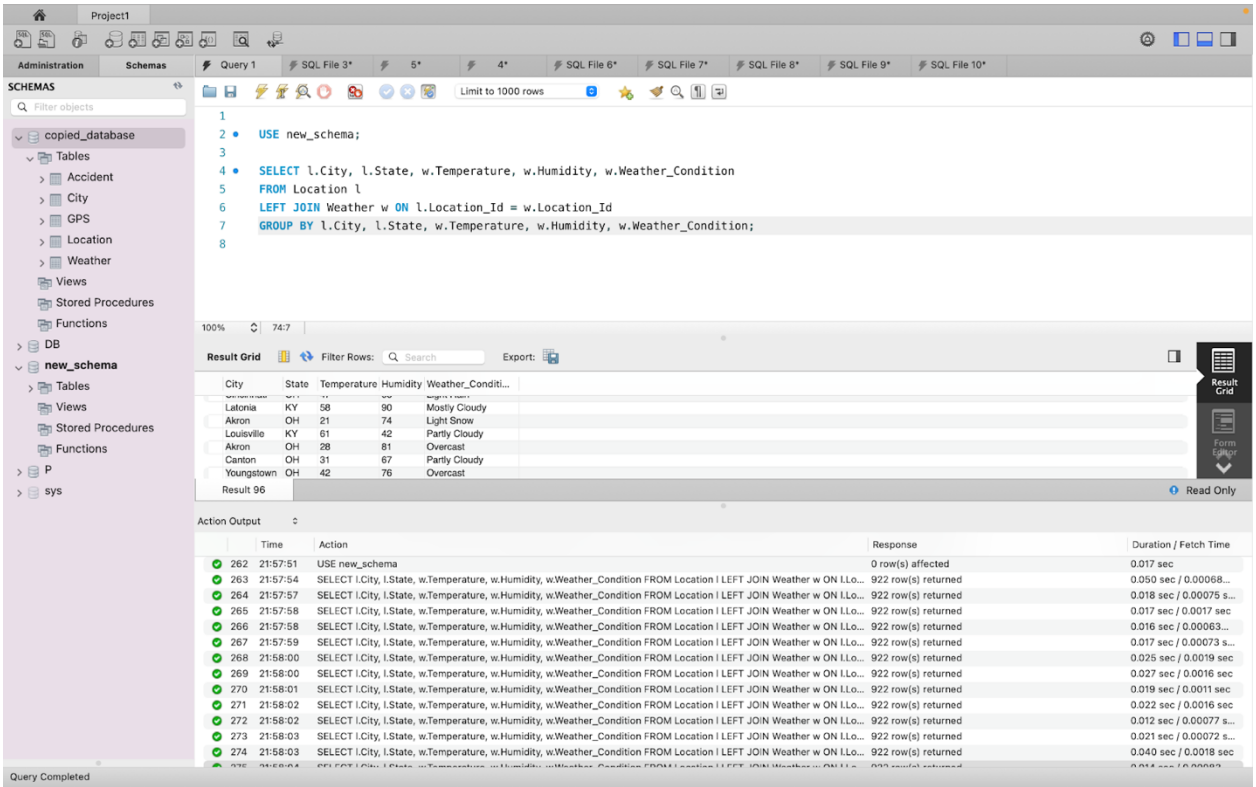
Average timing (Duration/Fetch) of this query run 10 times without index – 0.0099 sec



Average timing (Duration/Fetch) of this query run 10 times without index – 0.0605 sec



Average timing (Duration/Fetch) of this query run 10 times without index – 0.0141 sec



Average timing (Duration/Fetch) of this query run 10 times without index – 0.01272 sec

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

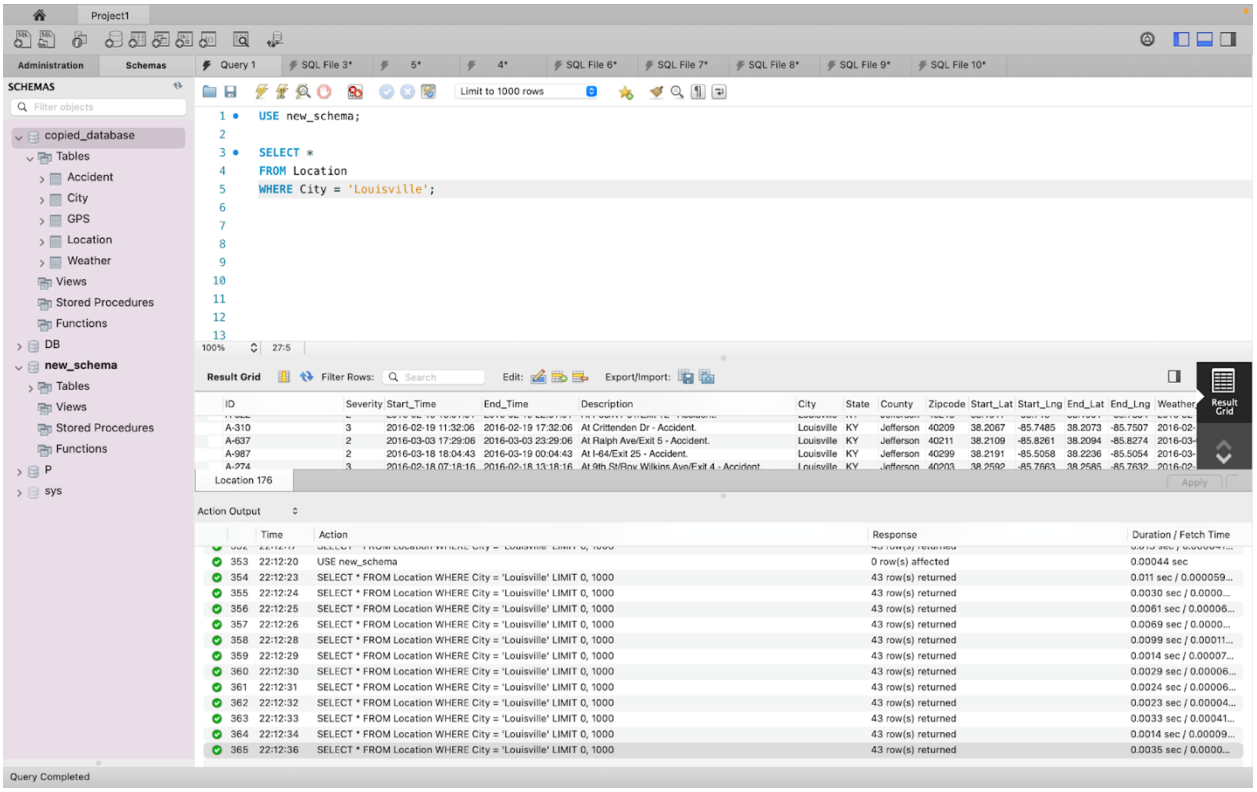
```
25
26 •  CREATE INDEX idx_city
27    ON Location (City);
```

100% 1:25

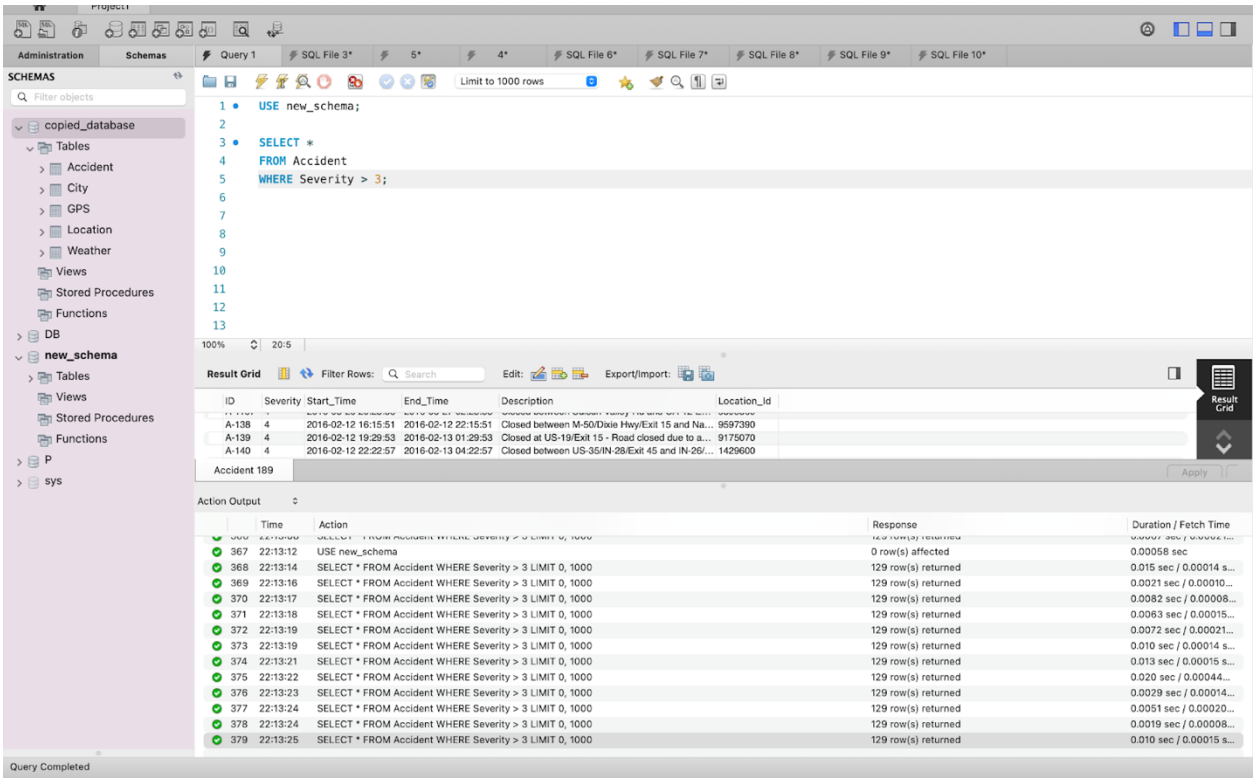
Action Output

	Time	Action	Response	Duration / Fetch Time
339	22:02:04	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Lo...	922 row(s) returned	0.0075 sec / 0.00055...
340	22:02:05	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Lo...	922 row(s) returned	0.0069 sec / 0.0004...
341	22:02:08	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Lo...	922 row(s) returned	0.014 sec / 0.00077 s...
342	22:03:47	CREATE INDEX idx_city ON Location (City)	0 row(s) affected Records: 0 Duplicates: 0 Warnings...	0.126 sec

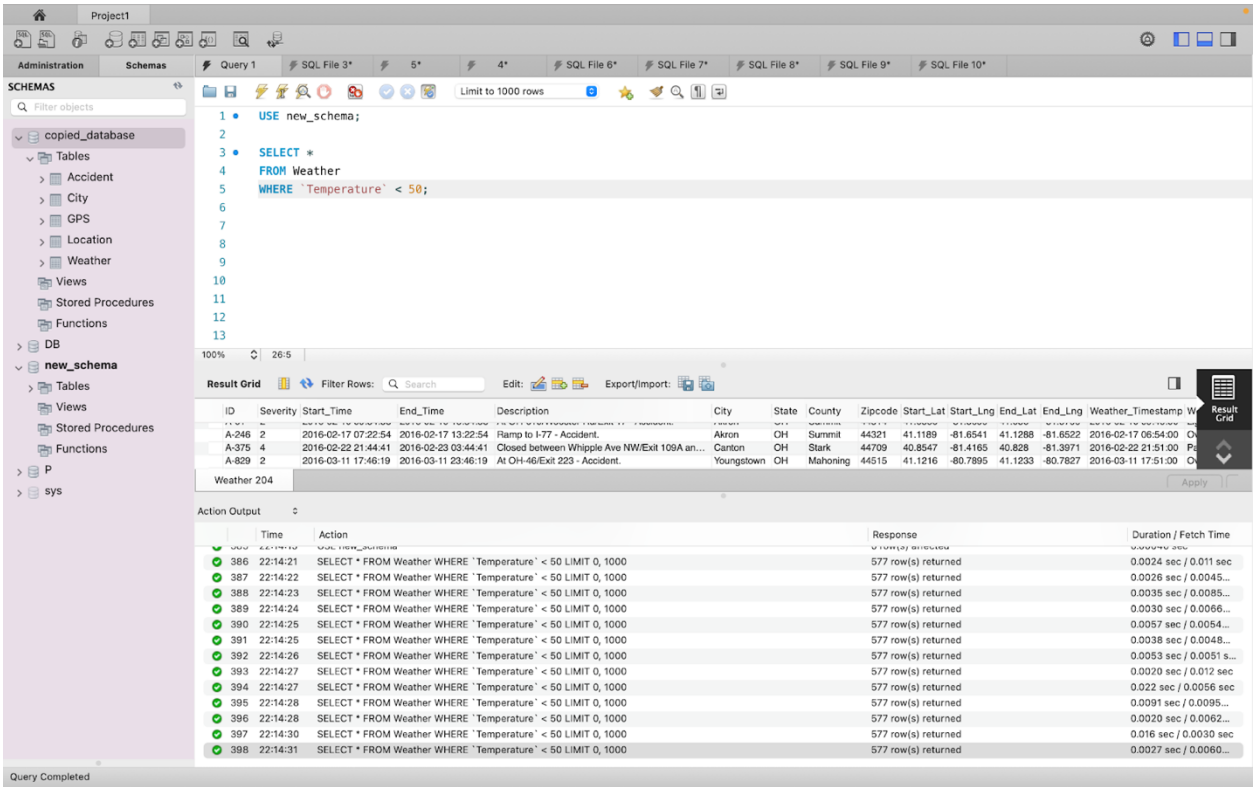
1.7. Write a query for each column – the query should include the column in the WHERE clause in a condition.



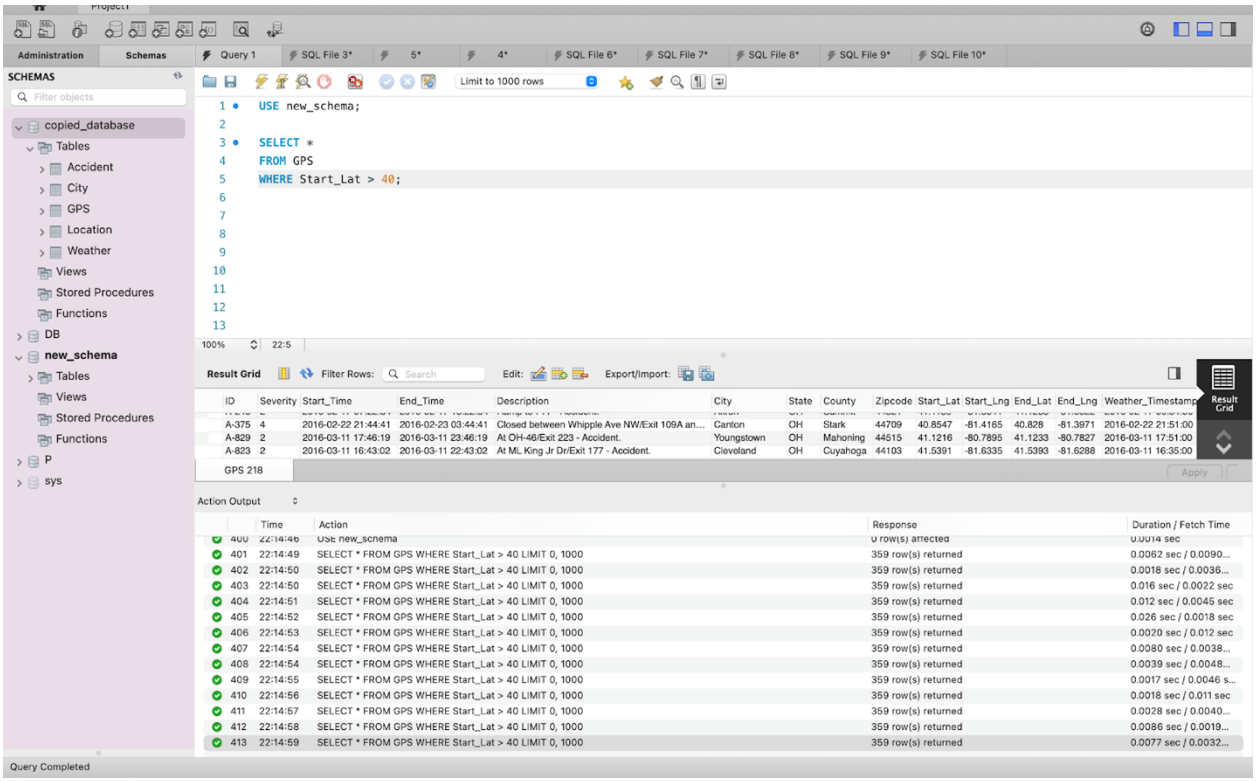
Average timing (Duration/Fetch) of this query run 10 times without index – 0.00362 sec



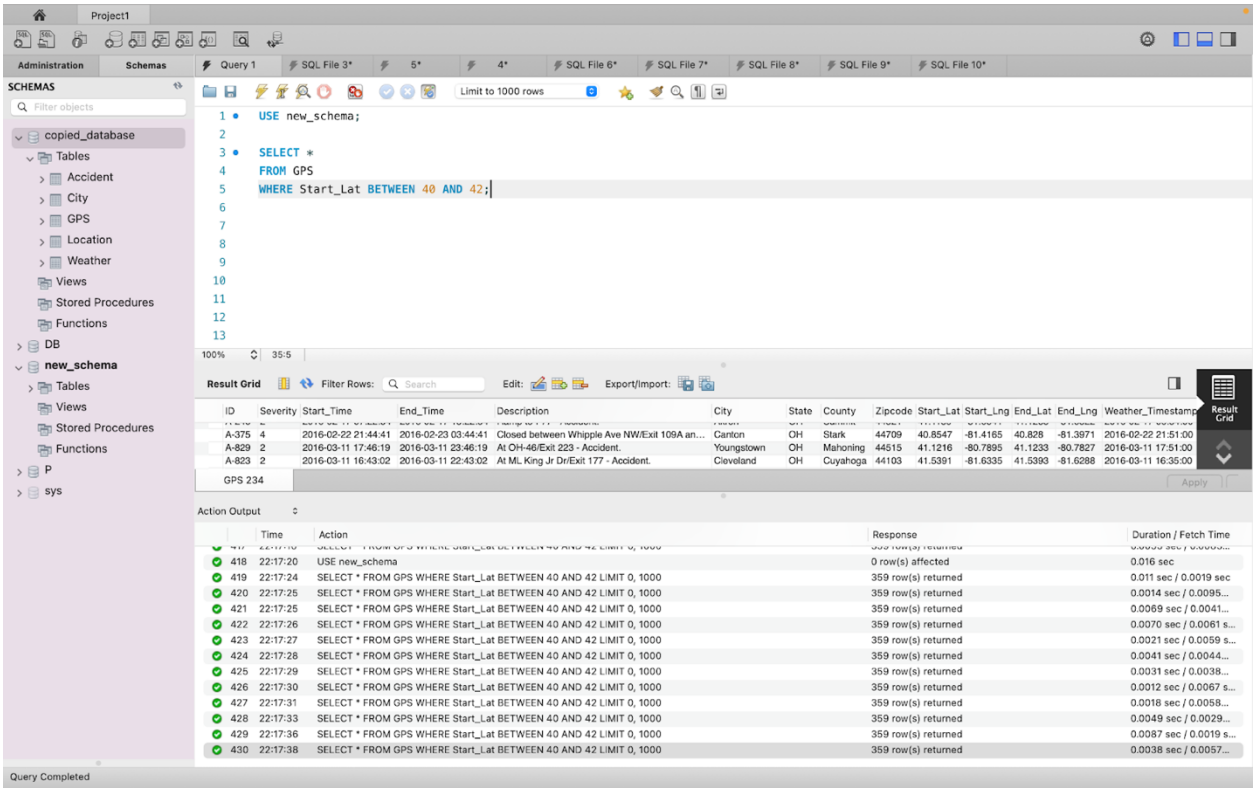
Average timing (Duration/Fetch) of this query run 10 times without index – 0.00672 sec



Average timing (Duration/Fetch) of this query run 10 times without index – 0.00379 sec

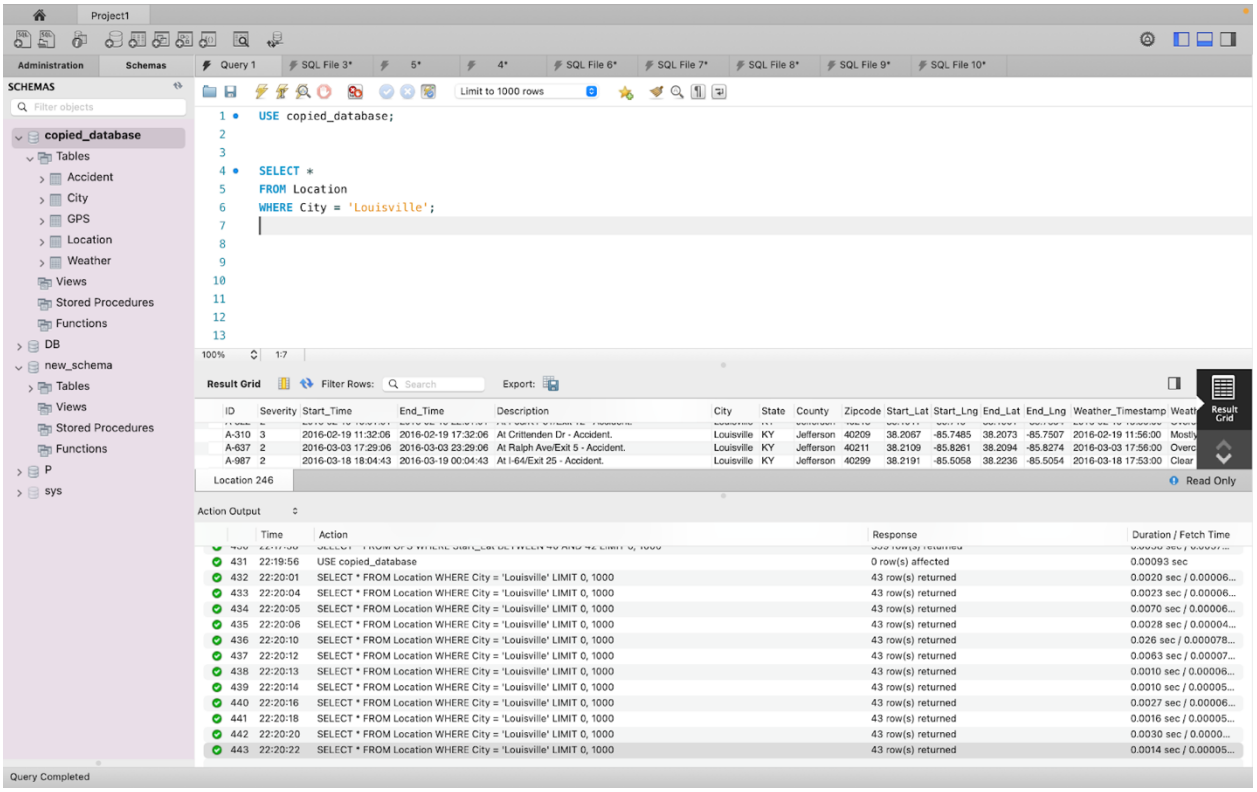


Average timing (Duration/Fetch) of this query run 10 times without index – 0.00738 sec

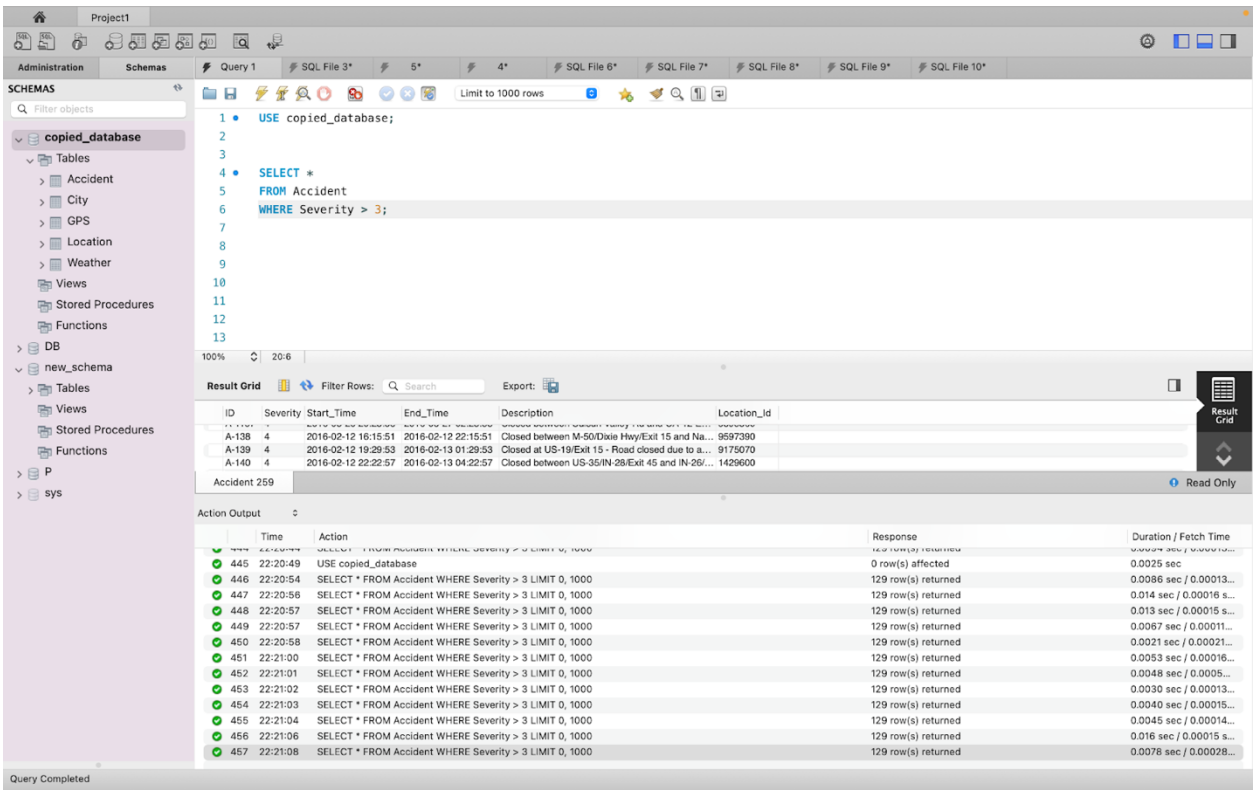


Average timing (Duration/Fetch) of this query run 10 times without index – 0.00337 sec

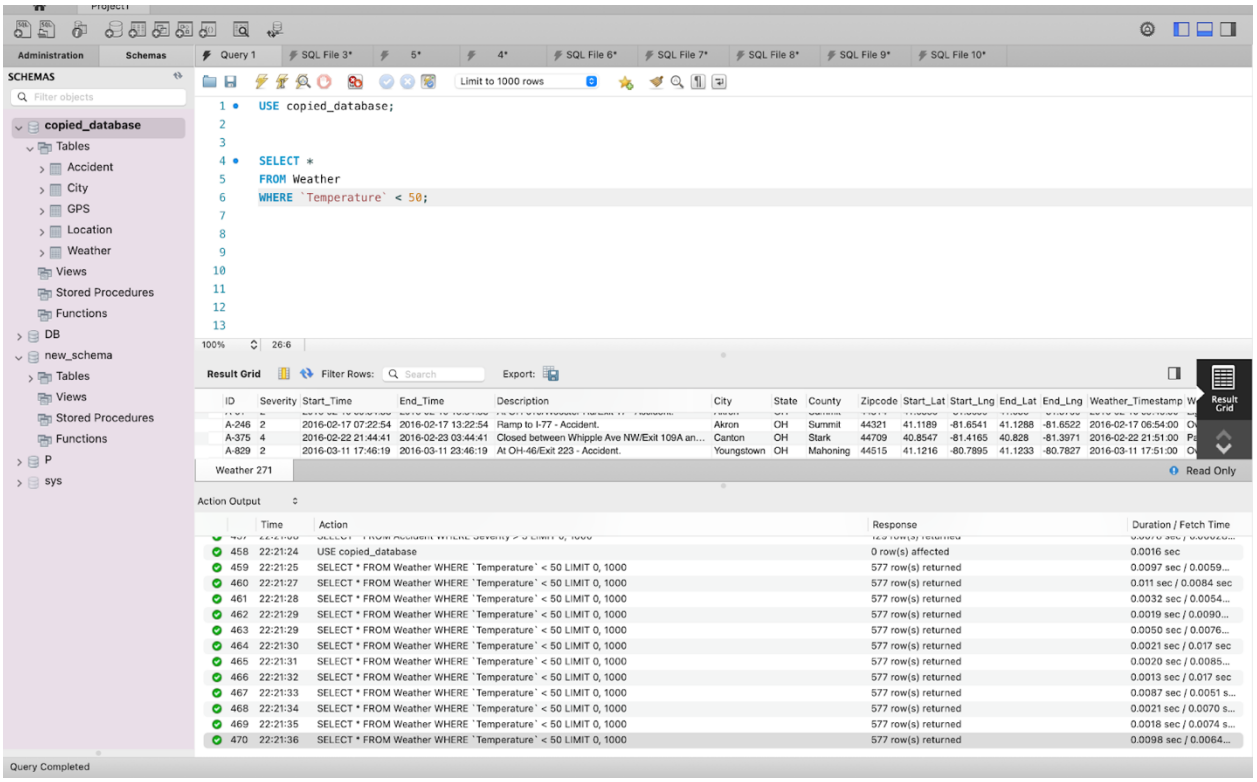
1.8. Execute and time these queries on both databases and report your findings (repeat timing for each query at least 10 times and average the times) 1.9. Make a conclusion based on your findings in this part.



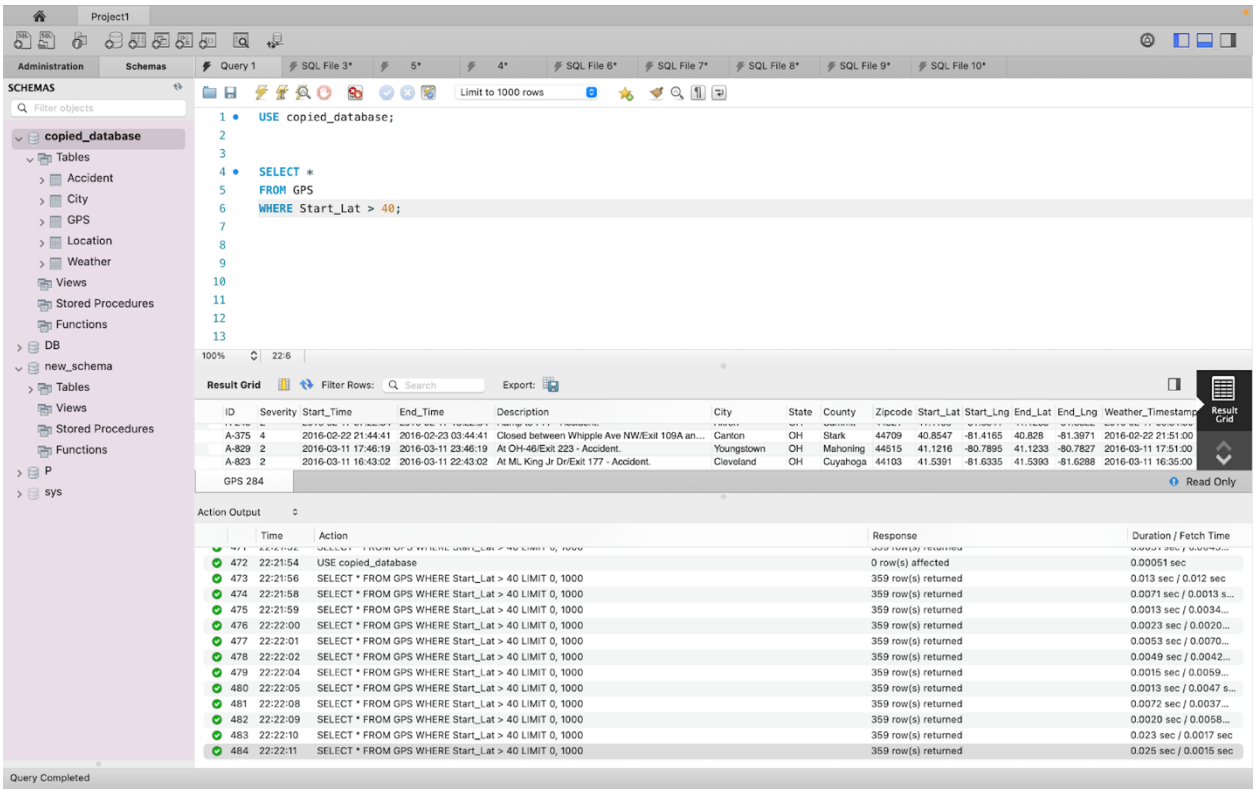
Average timing (Duration/Fetch) of this query run 10 times without index – 0.00541 sec



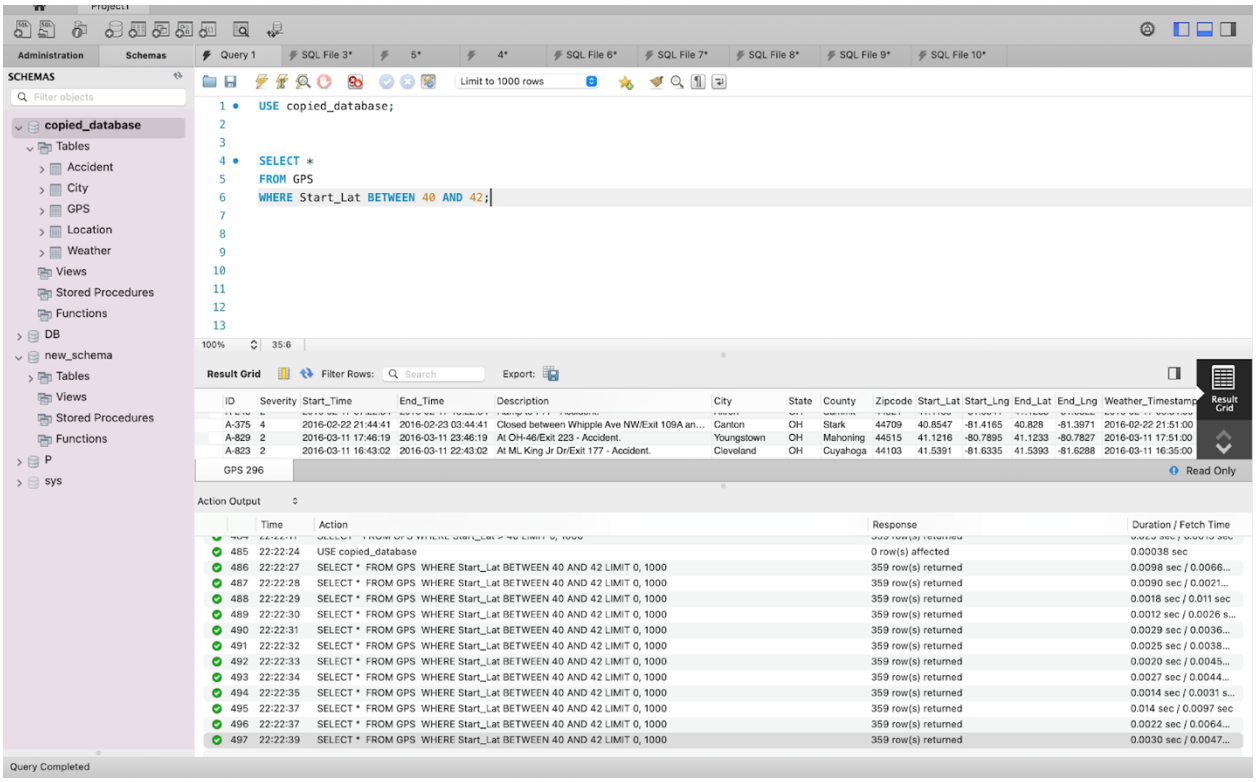
Average timing (Duration/Fetch) of this query run 10 times without index – 0.00846 sec



Average timing (Duration/Fetch) of this query run 10 times without index – 0.00716 sec



Average timing (Duration/Fetch) of this query run 10 times without index – 0.00745 sec



Average timing (Duration/Fetch) of this query run 10 times without index – 0.00436 sec

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

To avoid searching through every row in a database table every time it is accessed, indexes are used to quickly locate data. These indexes are created based on columns in the table, allowing for fast random searches and efficient retrieval of ordered items. Our research shows that the average time it takes to query the copied_database, which has no indexes, is longer compared to the new_schema, which has automatic or custom indexes. The absence of indexes in copied_database causes longer query times. Therefore, using indexes is the best method to quickly locate data in a database.

MongoDB and MQL

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

The PakWheels dataset contains information about the largest online marketplace in Pakistan for buying and selling cars. It gathers many brand-new, used, and certified pre-owned vehicles from many dealers and individual sellers. The dataset has fields such as brand of datatype **object**, model of datatype **string**, modelDate of datatype **integer** and so on.

- @type: Vehicle type
- model: Vehicle model
- itemCondition: vehicle condition
- modelDate: vehicle model release date
- manufacturer: vehicle manufacturer
- fuelType vehicle fuel type
- image: ad/vehicle cover image
- vehicleTransmission: vehicle transmission type (manual, automatic)
- color: vehicle color
- bodyType: vehicle body type
- mileageFromOdometer: km's the vehicle has driven.

- sellerLocation: seller location
- postedFrom: platform ad posted from
- features: list of features available in the car. e.g.: AC, Power windows, etc.
- price: Asking price.
- priceCurrency: Currency. e.g.: PKR, USD.
- brand: {
    - @type:
    - name: vehicle brand name
    }
- vehicleEngine: {
    - @type:
    - engineDisplacement: km's the vehicle has driven
    }
- extraFeatures: {
    - RegisteredIn: city the vehicle is registered in.
    - Color: vehicle color
    - Assembly: vehicle assembly location (local, imported
    - EngineCapacity: engine capacity in cubic centimeters(cc)
    - BodyType: vehicle body type
    - LastUpdated: Ad last updated
    }

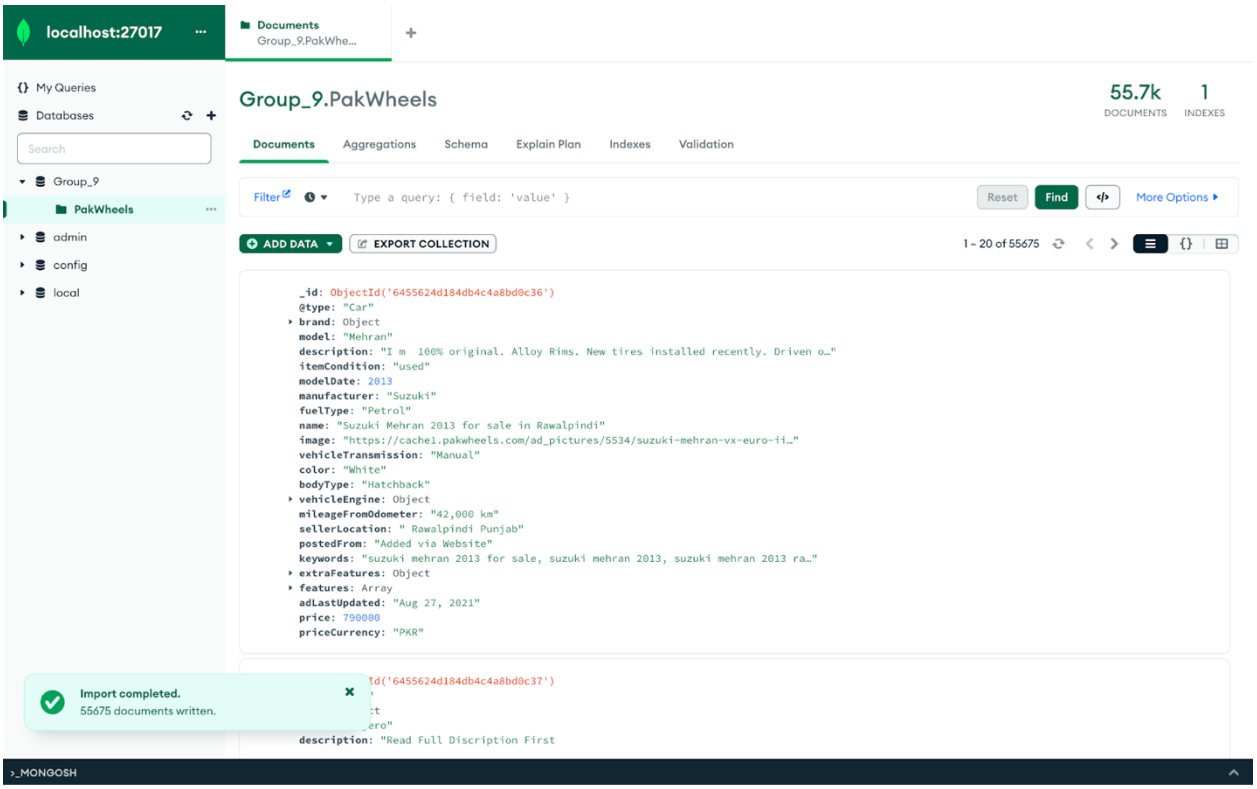
2.2. Explain why it is better to use non-relational databases such as MongoDB to work with such a dataset (explain in the context of your dataset)

In the case of the PakWheels dataset, non-relational databases such as MongoDB can be a better option to work with as compared to traditional relational databases such as SQL.

The primary reason for this is that the PakWheels dataset is unstructured and contains many fields with varying data types. The dataset contains a mix of numeric, string, and text data types, which makes it challenging to fit it into a pre-defined schema. In a traditional relational database, we would need to define a fixed schema for the data beforehand, and any deviation from the schema would require updating the schema, which can be time-consuming and error prone.

On the other hand, non-relational databases such as MongoDB are designed to handle unstructured data efficiently. MongoDB is a document-based database that stores data in JSON-like documents, which can be nested and flexible. In the context of the PakWheels dataset, we can store each row as a document, and each column can be a field within the document. Since MongoDB does not require a pre-defined schema, we can easily add or remove fields as needed, which makes it easier to handle the varying data types and the unstructured nature of the data.
Another advantage of using MongoDB in the context of the PakWheels dataset is its ability to scale horizontally. As the dataset grows, we can add more servers to the cluster and distribute the load across them, which can help improve performance and scalability.

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



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

- **_id:** The unique identifier for each document in the collection.
- **name:** The make and model of the car.
- **modelDate**: The year the car was manufactured.
- **mileageFromOdometer**: The number of miles the car has been driven.
- **vehicleTransmission**: The type of transmission (automatic or manual).
- **engineDisplacement:** The engine capacity of the car in cc.
- **fuelType:** The type of engine (petrol, diesel).
- **Color**: The color of the car.
- **Assembly**: The country where the car was assembled.
- **Body Type:** The type of body (sedan, hatchback, SUV, etc.).
- **Price**: The asking price of the car.
- **Description**: A brief description of the car.
- **Ad Title:** The title of the ad for the car.
- **image**: The URL of the ad for the car.

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

- **Name**: Honda Civic, Toyota Corolla, Suzuki Cultus, Hyundai Santro, etc.
- **modelDate**: 2010, 2015, 2018, 2021, etc.
- **mileageFromOdometer**: 50,000 km, 100,000 km, 150,000 km, 200,000 km, etc.
- **vehicleTransmission**: Automatic, Manual.
- **fuelType**: Petrol, Diesel.
- **Color**: White, Black, Silver, Blue, Red, etc.
- **Assembly**: Local, Imported.
- **Body Type:** Sedan, Hatchback, SUV, Pickup, etc.
- **Price**: 500,000 PKR, 1,000,000 PKR, 1,500,000 PKR, 2,000,000 PKR, etc.

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

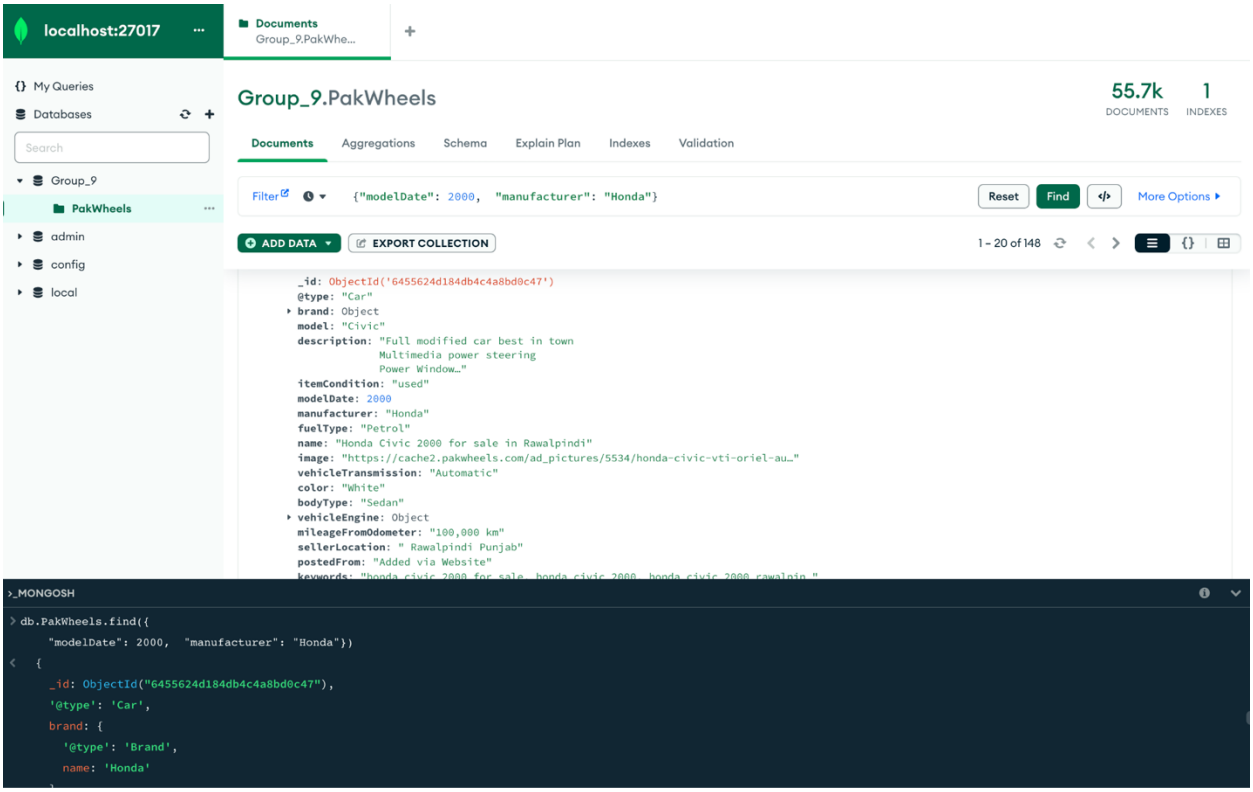
- **Sunroof**: Some cars may have sunroofs, while others may not.
- **Navigation System**: Some cars may have built-in navigation systems, while others may not.
- **Keyless entry**: Some cars may have a keyless entry installed, while others may not.
- **Power Locks:** Some cars may have power locks installed, while others may not.
- **Alloy Rims:** Some cars may have alloy rims, while others may have steel rims.
- **Airbag**: Some cars may have airbags, while others may not.
- **CD Player:** Some cars may have a CD player installed, while others may not.
- **Immobilizer Key:** Some cars may have an immobilizer key to prevent theft, while others may not.

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

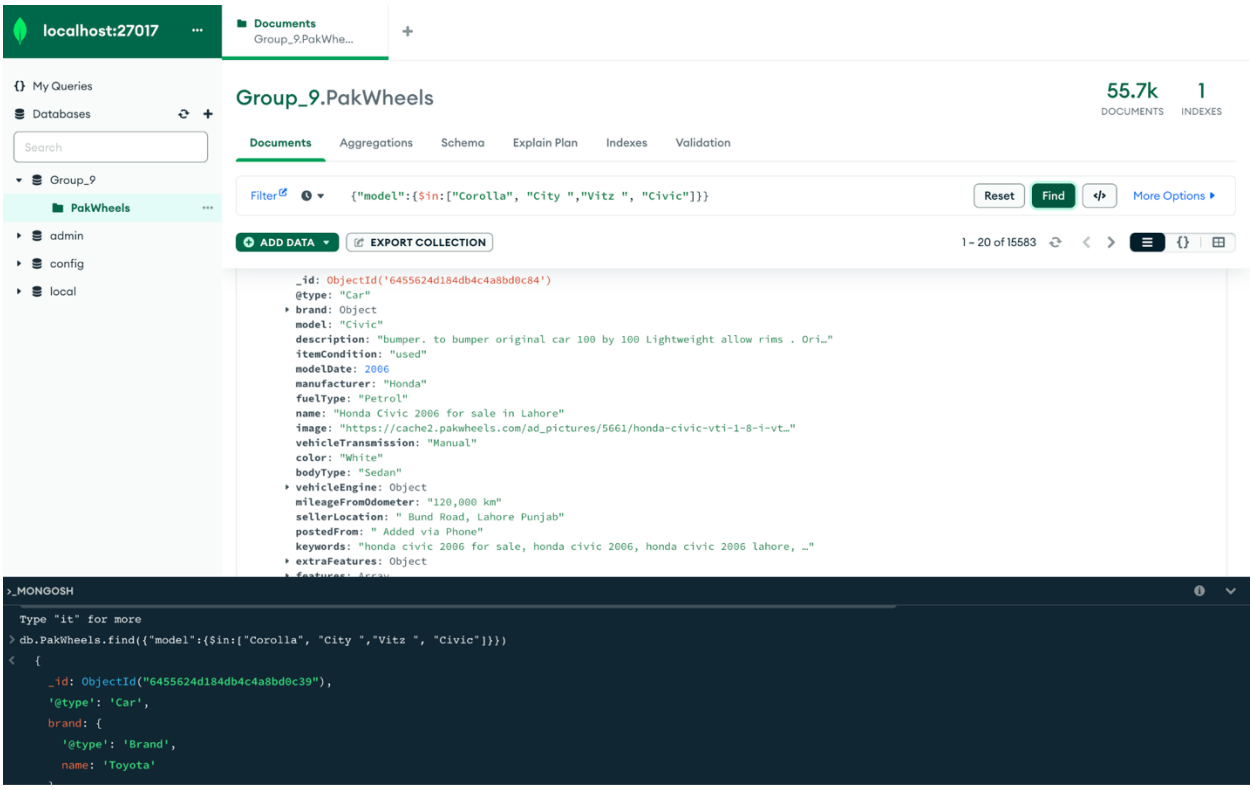
- **Sunroof**: Yes, No
- **Navigation System**: Yes, No
- **Keyless entry**: Yes, No
- **Power Locks**: Yes, No
- **Alloy Rims**: Yes, No
- **Airbag**: Yes, No
- **CD Player**: Yes, No
- **Immobilizer Key:** Yes, No

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)

Query 1: To find the bikes which are made in 2000 with Honda as manufacturer.
db.PakWheels.find({ "modelDate": 2000, "manufacturer": "Honda" })

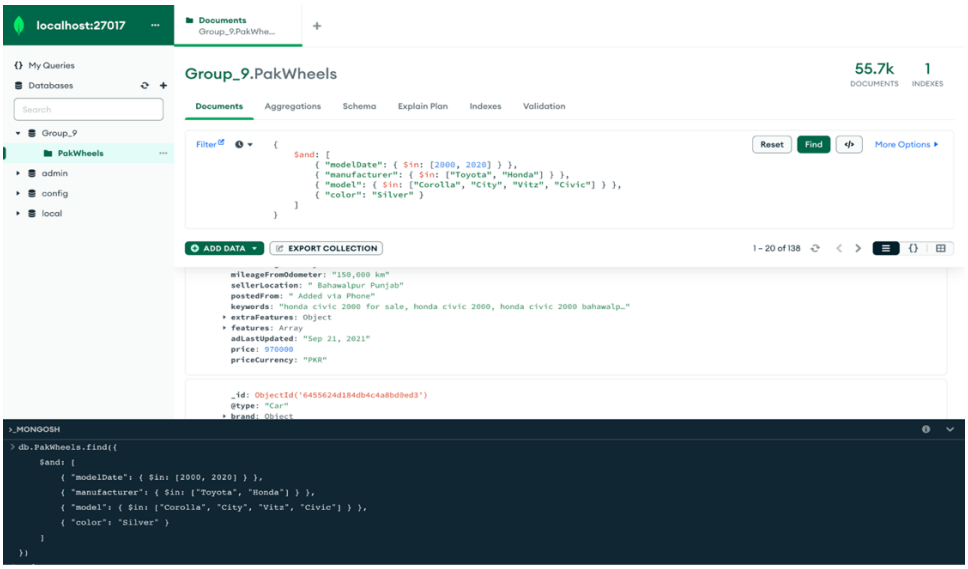


Query 2: To find all the cars of models: Corolla, City, Vitz, Civic
db.PakWheels.find({"model":{$in:["Corolla", "City ","Vitz ", "Civic"]}})



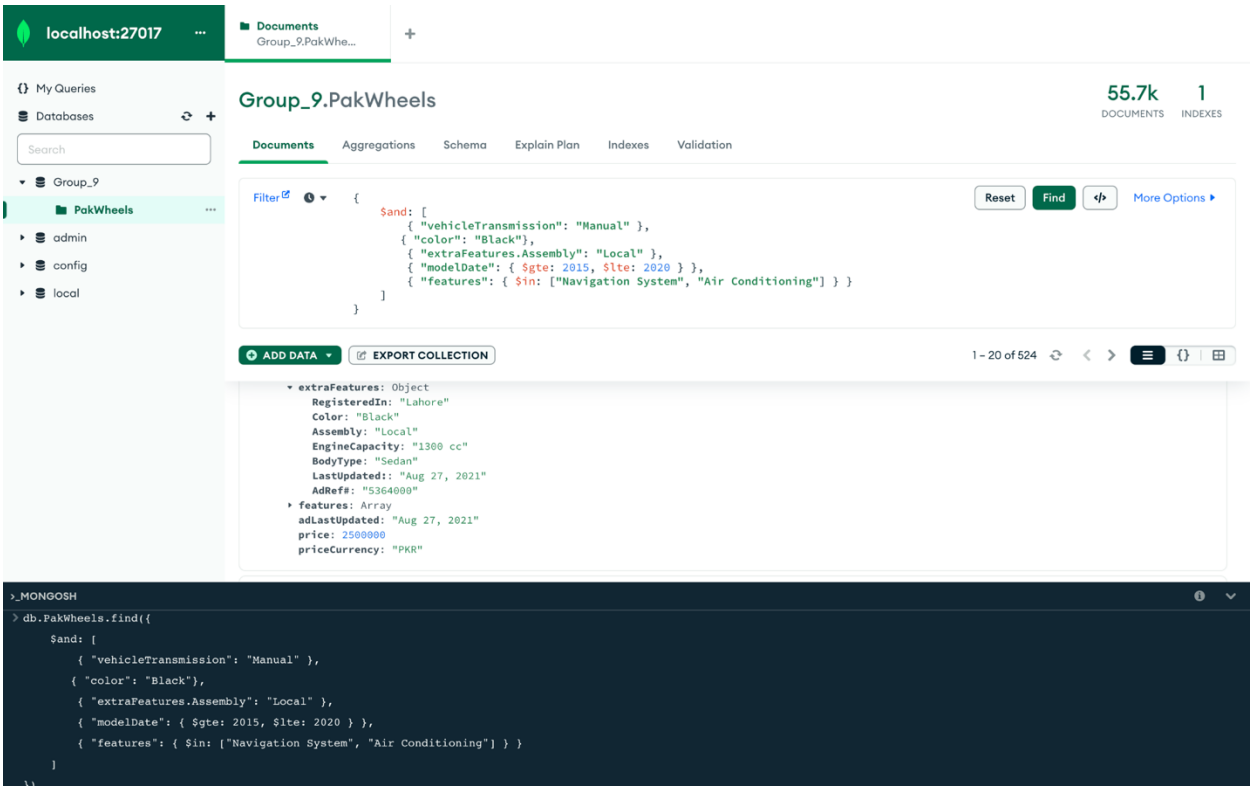
Query 3: To find the cars of models: Corolla, City, Vitz, Civic made in 2000, 2020 with Toyota, Honda as manufacturer and silver color:

```
db.PakWheels.find({
   $and: [
      { "modelDate": { $in: [2000, 2020] } },
      { "manufacturer": { $in: ["Toyota", "Honda"] } },
      { "model": { $in: ["Corolla", "City", "Vitz", "Civic"] } },
      { "color": "Silver" }
   ]
})
```



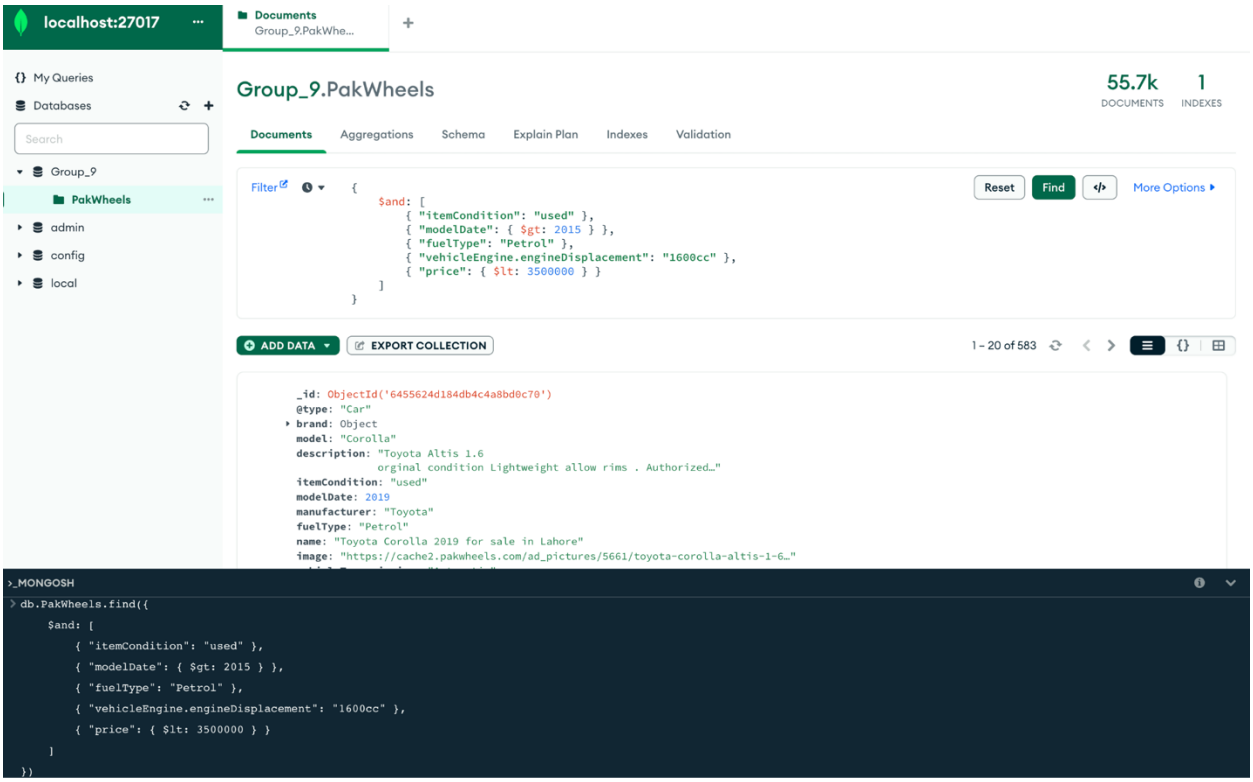
Query 4: To find all the used cars with Navigation System =, Air Conditioning from 2015 to 2020.

```
db.PakWheels.find({
   $and: [
      { "vehicleTransmission": "Manual" },
     { "color": "Black"},
      { "extraFeatures.Assembly": "Local" },
      { "modelDate": { $gte: 2015, $lte: 2020 } },
      { "features": { $in: ["Navigation System", "Air Conditioning"] } }
   ]
})
```



Query 5: To find all the used petrol cars manufactured after 2015 of price less than 3.5 million PKR(Pakistani Rupees)

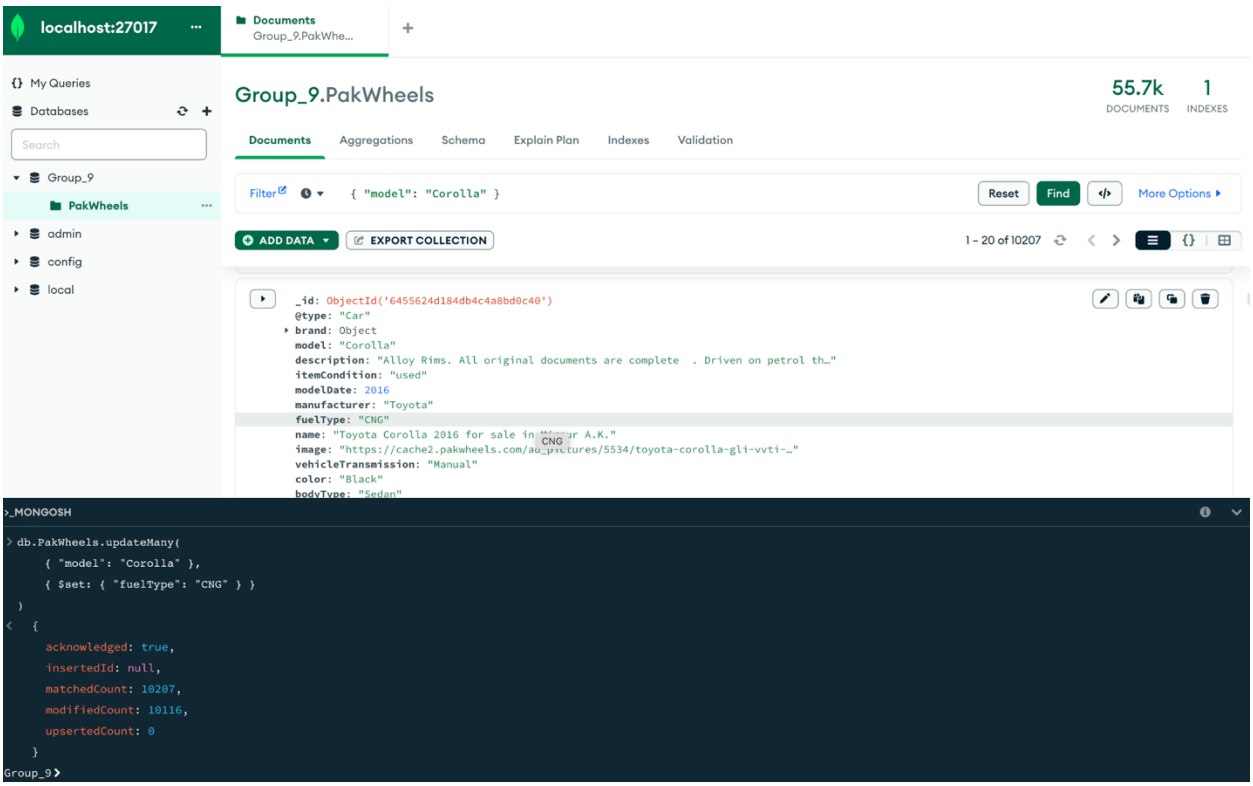
```
db.PakWheels.find({
   $and: [
      { "itemCondition": "used" },
      { "modelDate": { $gt: 2015 } },
      { "fuelType": "Petrol" },
      { "vehicleEngine.engineDisplacement": "1600cc" },
      { "price": { $lt: 3500000 } }
   ]
})
```



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)

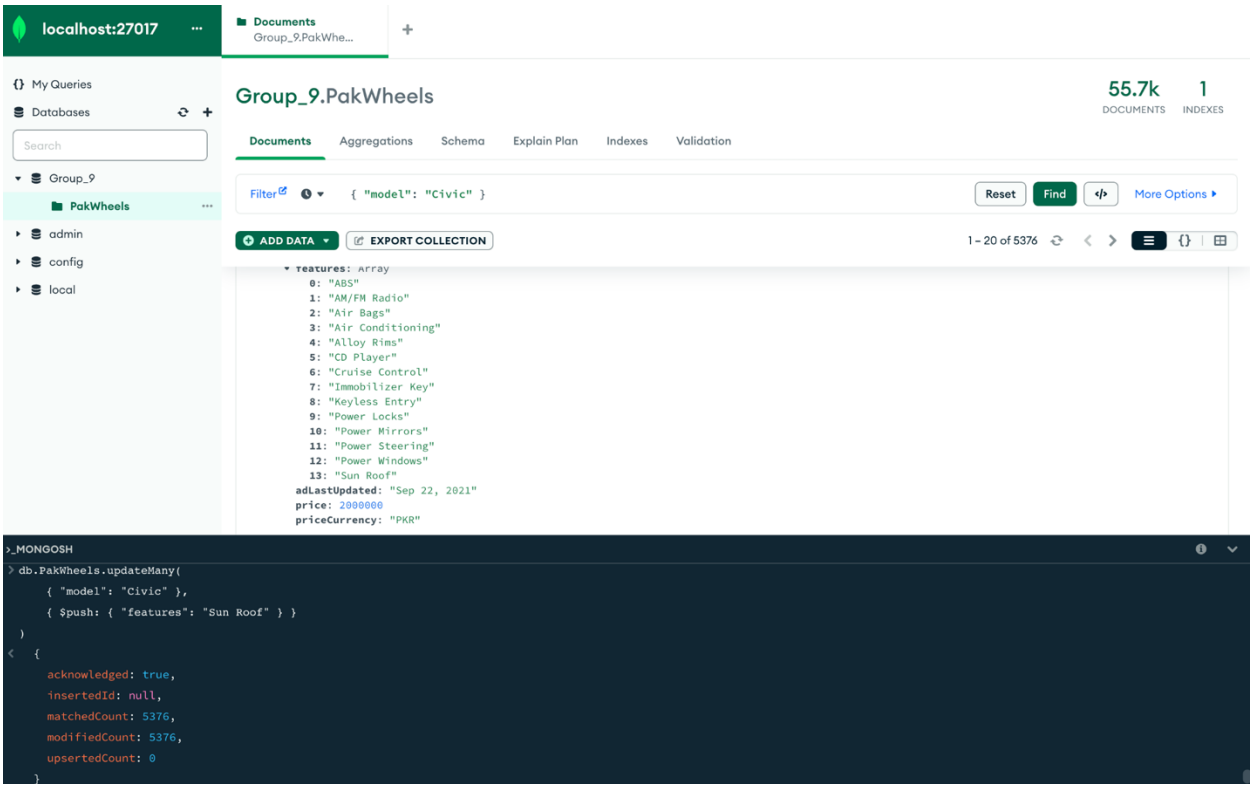
Query 1: Update the fuelType of all Toyota Corolla cars to "CNG"

```
db.PakWheels.updateMany(
  { "model": "Corolla" },
  { $set: { "fuelType": "CNG" } })
```



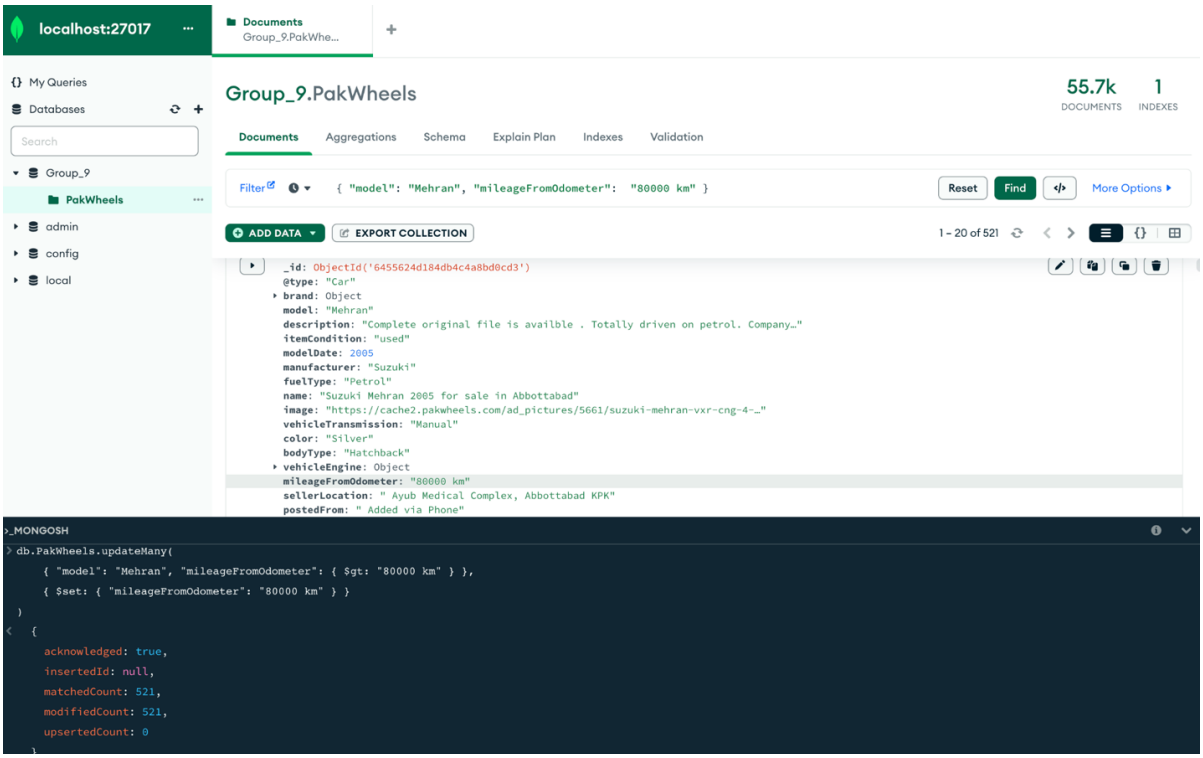
Query 2: Add "Sunroof" to the extraFeatures array of all Honda Civic cars:

```
db.PakWheels.updateMany(
  { "model": "Civic" },
  { $push: { "features": "Sun Roof" } })
```



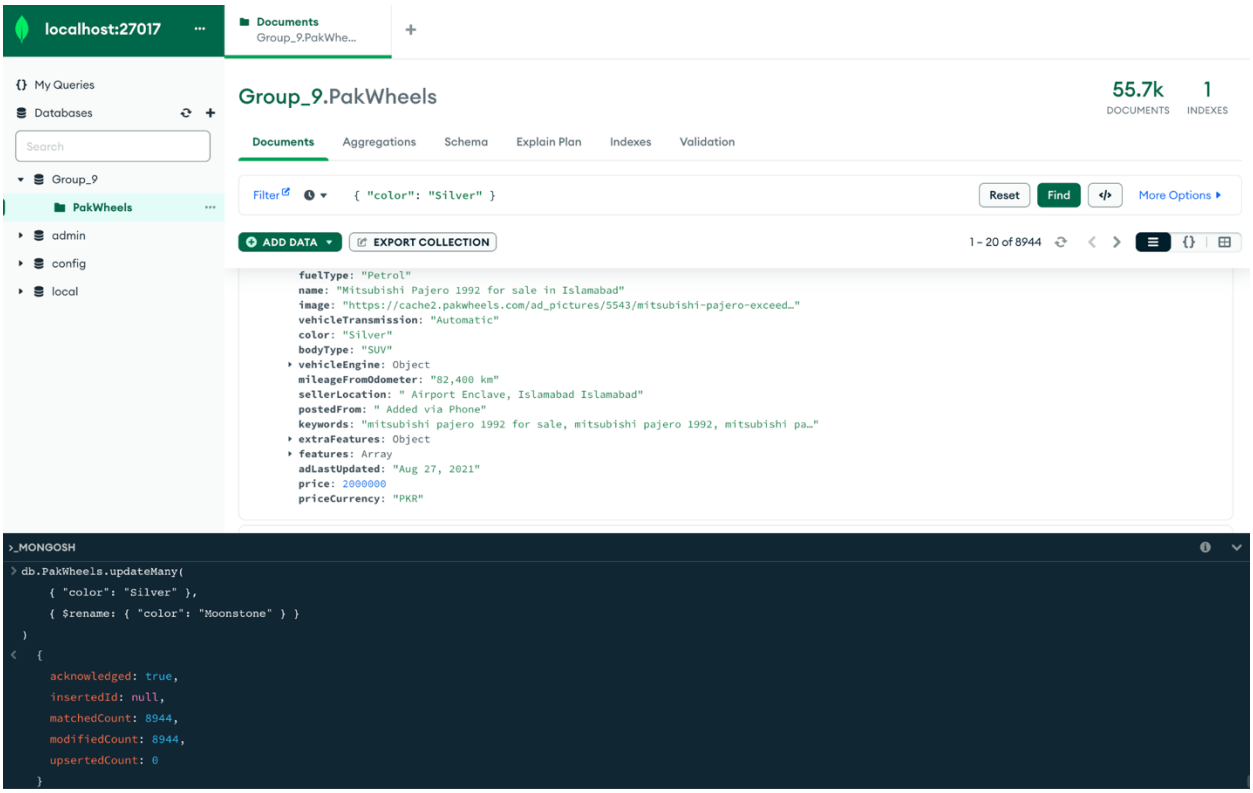
Query 3: Update the mileage of all Suzuki Mehran cars with a mileage greater than 80,000 km to 80,000 km:

```
db.PakWheels.updateMany(
  { "model": "Mehran", "mileageFromOdometer": { $gt: "80000 km" } },
  { $set: { "mileageFromOdometer": "80000 km" } })
```



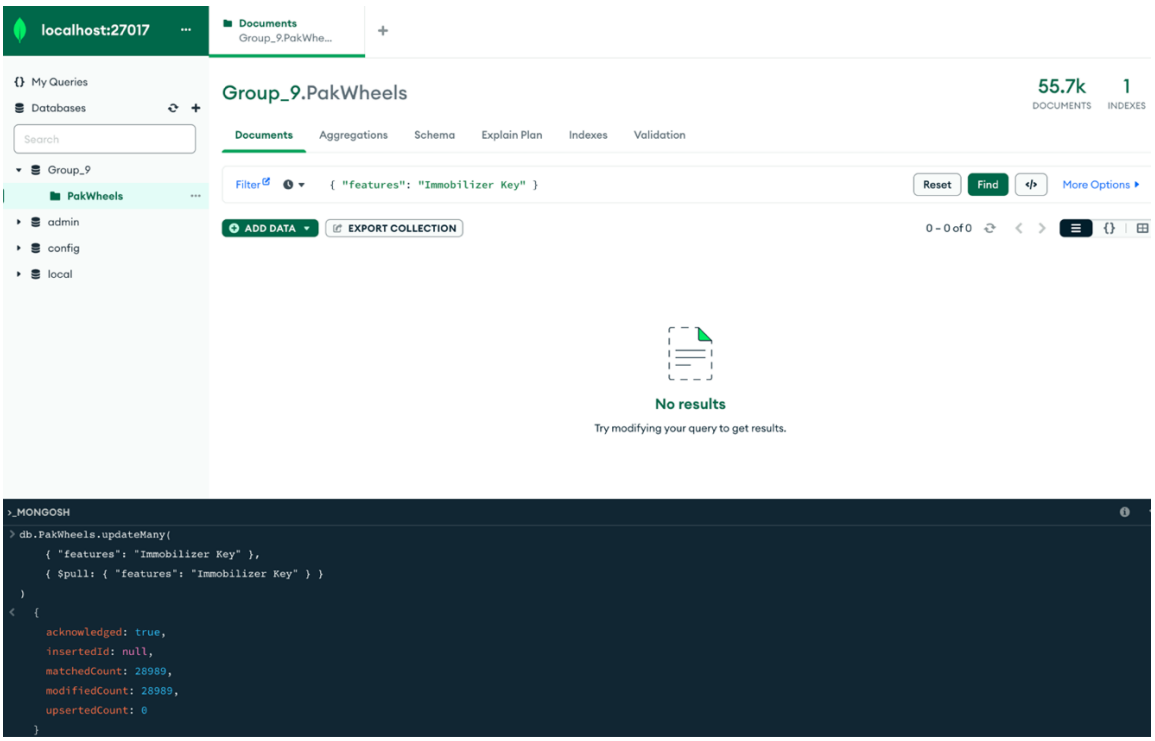
Query 4: Rename the color "Silver" to "Moonstone" for all cars:

```
db.PakWheels.updateMany(
   { "color": "Silver" },
   { $rename: { "color": "Moonstone" } })
```



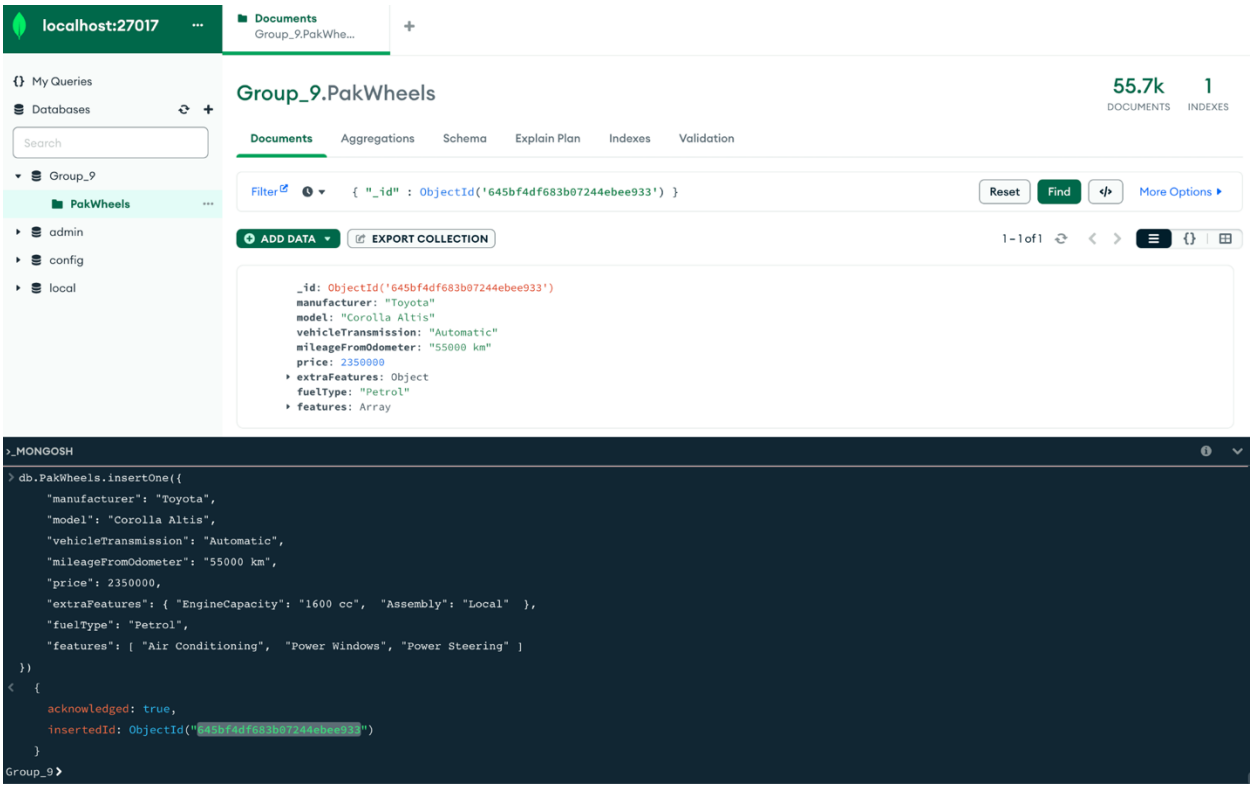
Query 5: Remove "Immobilizer Key" from the features array of all cars:

```
db.PakWheels.updateMany(
   { "features": "Immobilizer Key" },
   { $pull: { "features": "Immobilizer Key" } } )
```

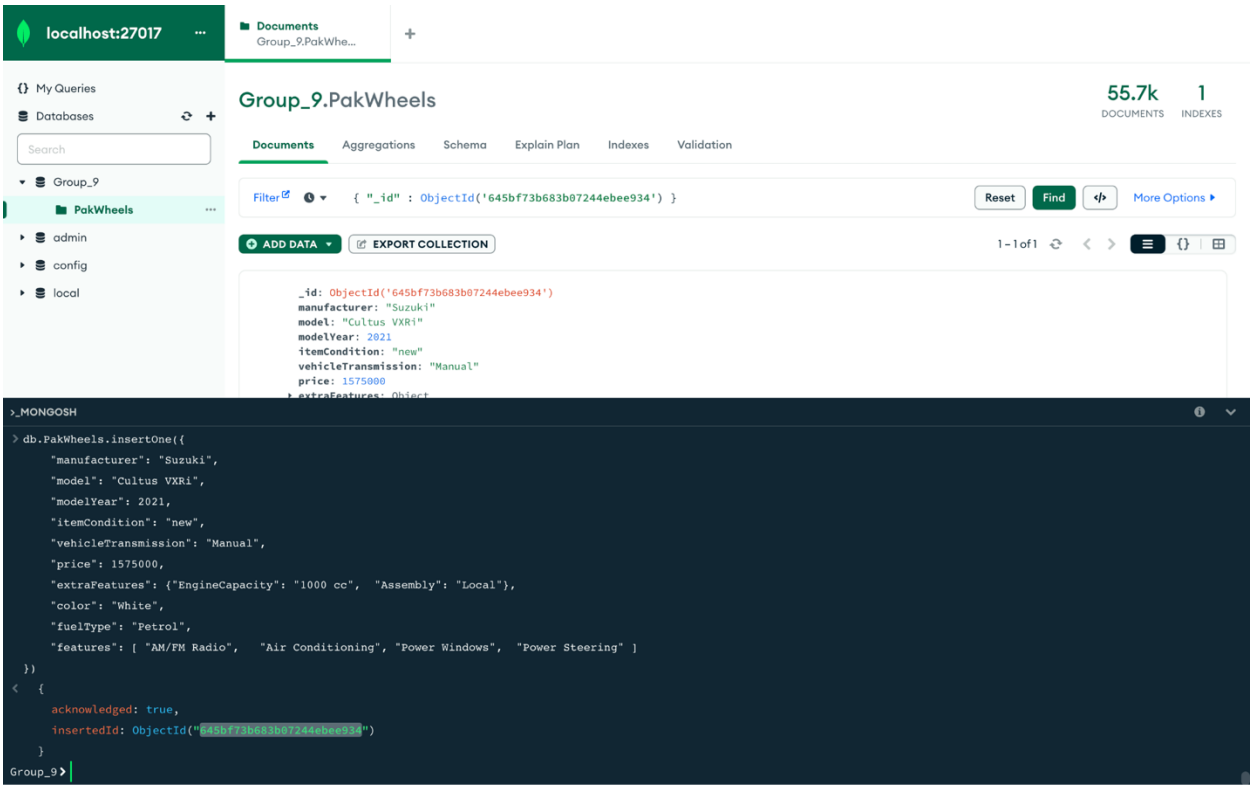


2.8. Write at least 5 queries to insert new documents into your database (provide the queries and results in your report)

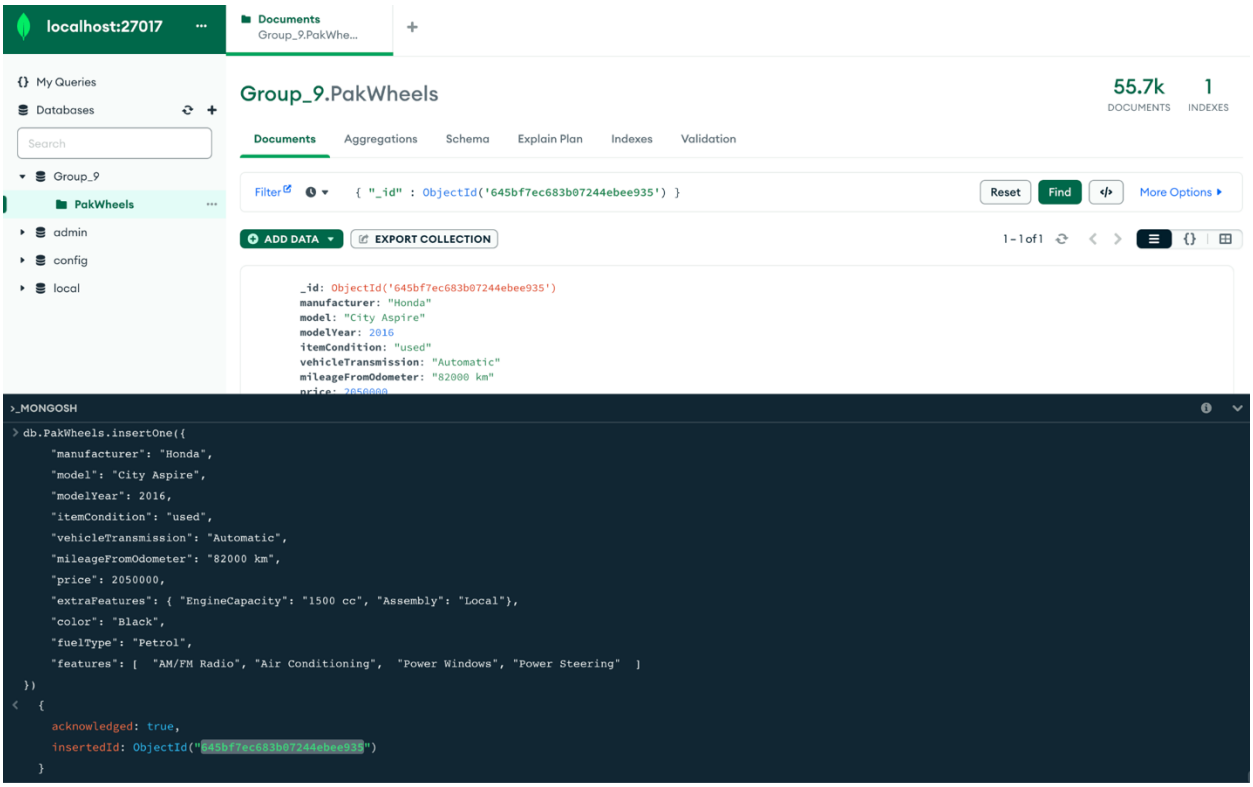
Object Id: 645bf4df683b07244ebee933



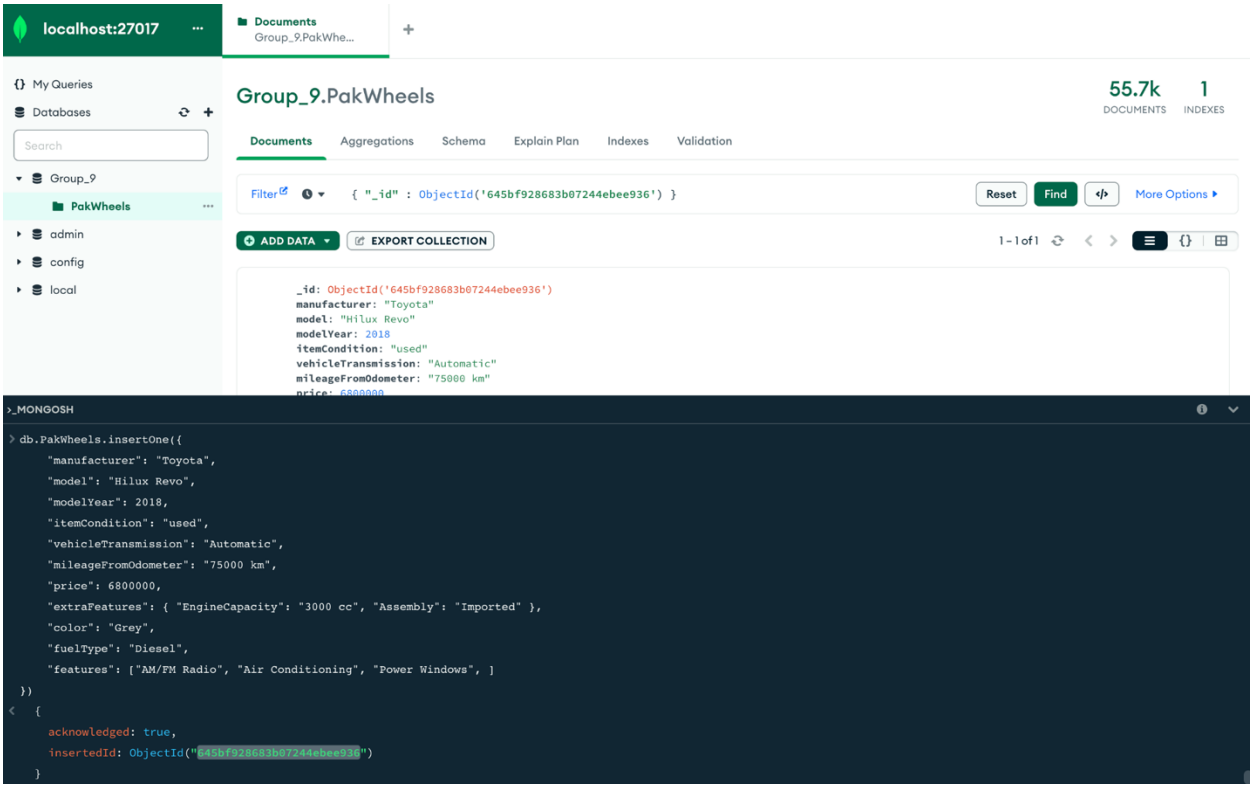
ObjectId: 645bf73b683b07244ebee934



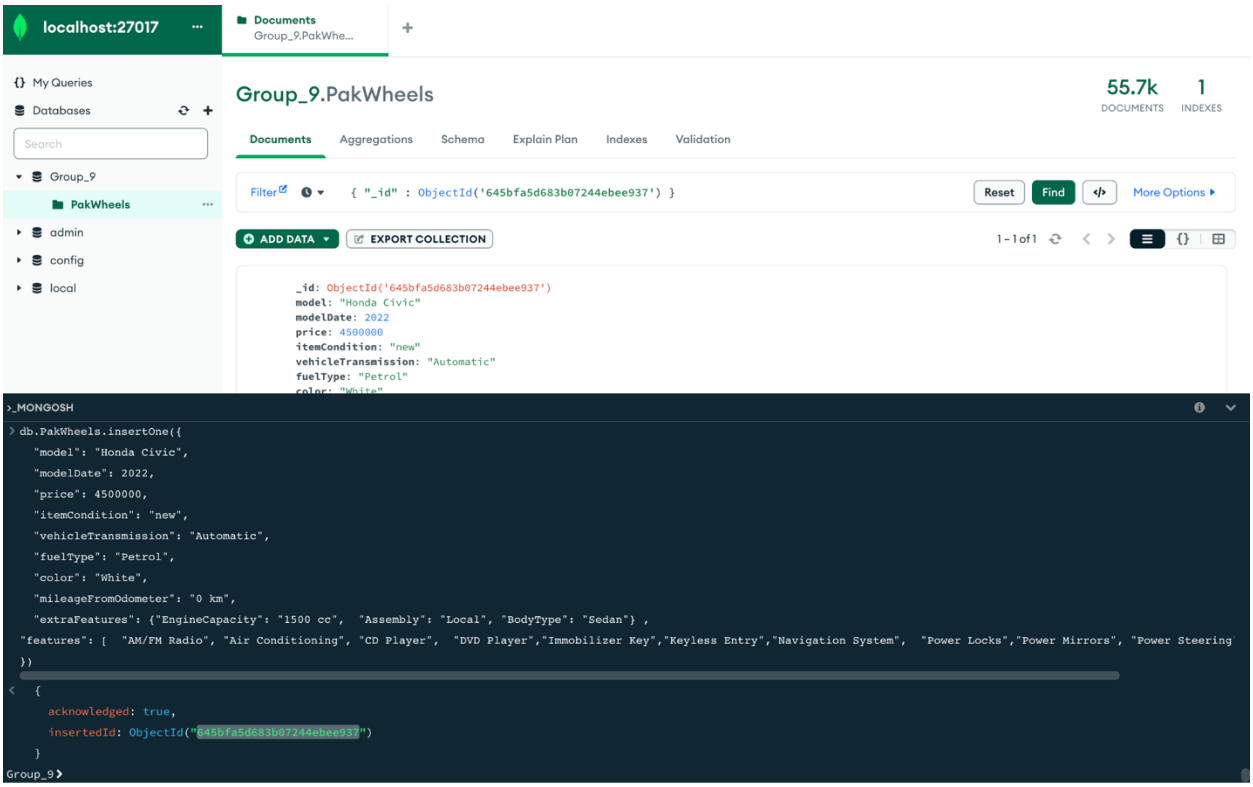
Object Id: 645bf7ec683b07244ebee935



Object Id:
645bf928683b07244ebee936

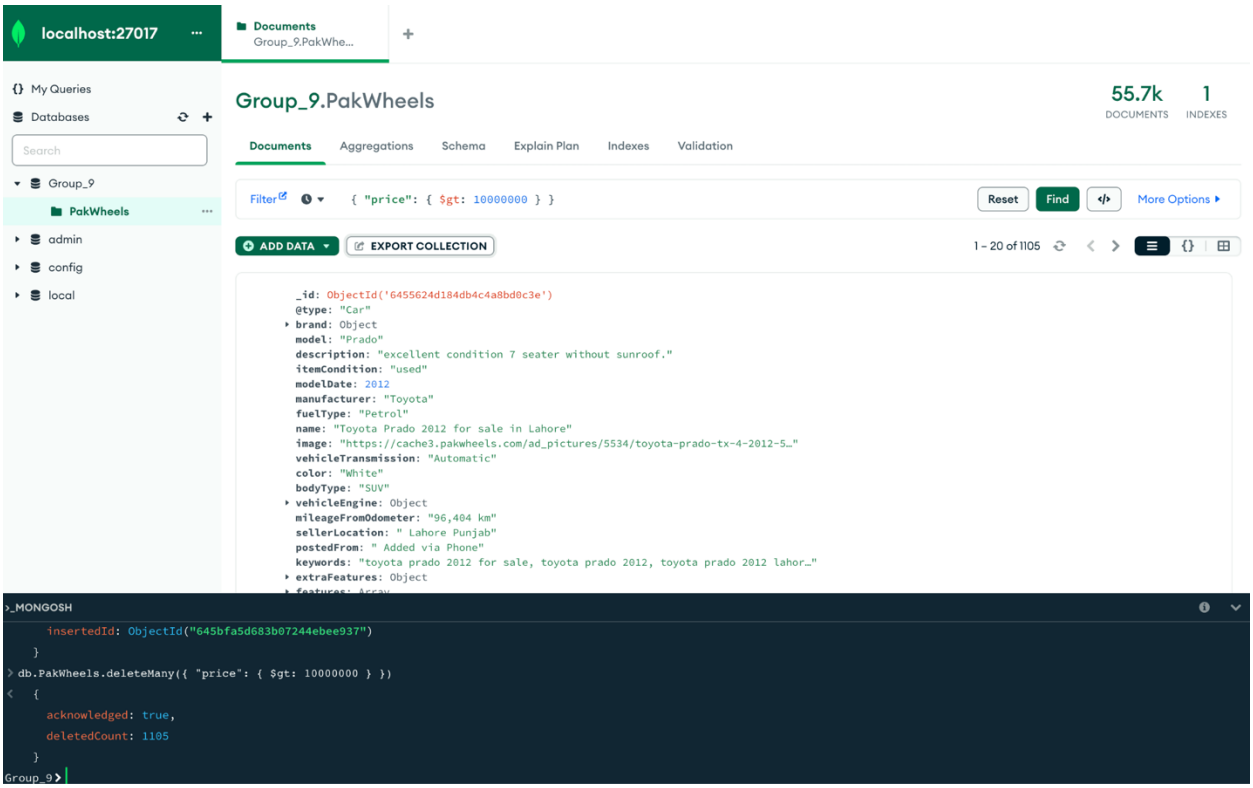


Object Id: 645bfa5d683b07244ebee937

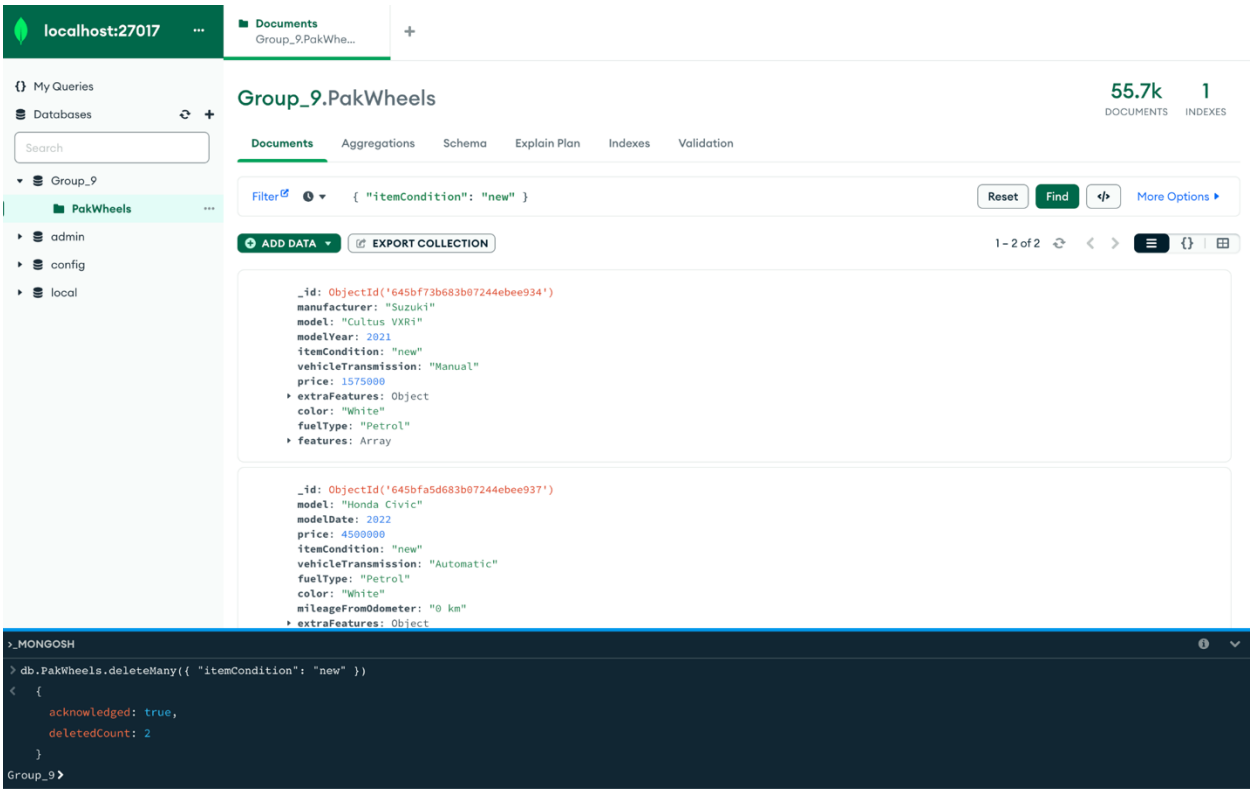


2.9. Write at least 5 delete queries to remove documents from your database (provide the queries and results in your report)

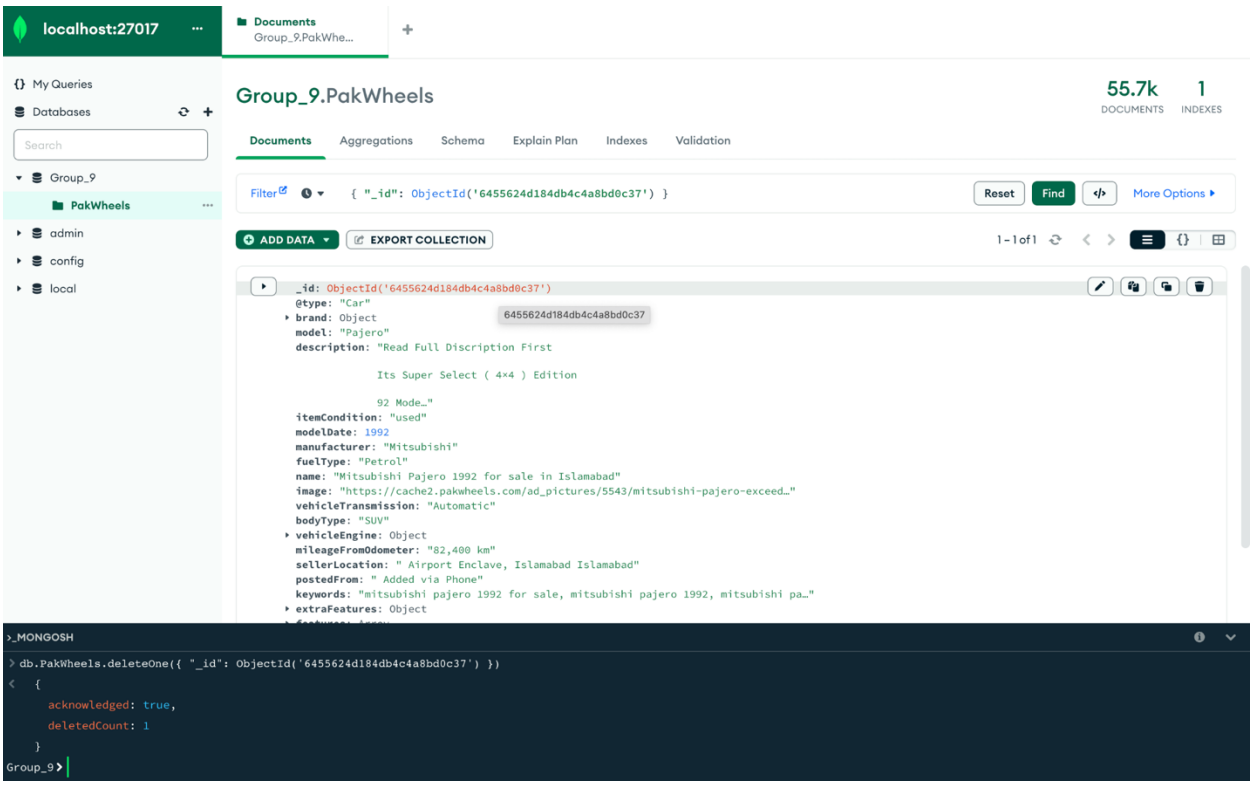
Query 1: Delete all documents where the price is greater than 10000000:



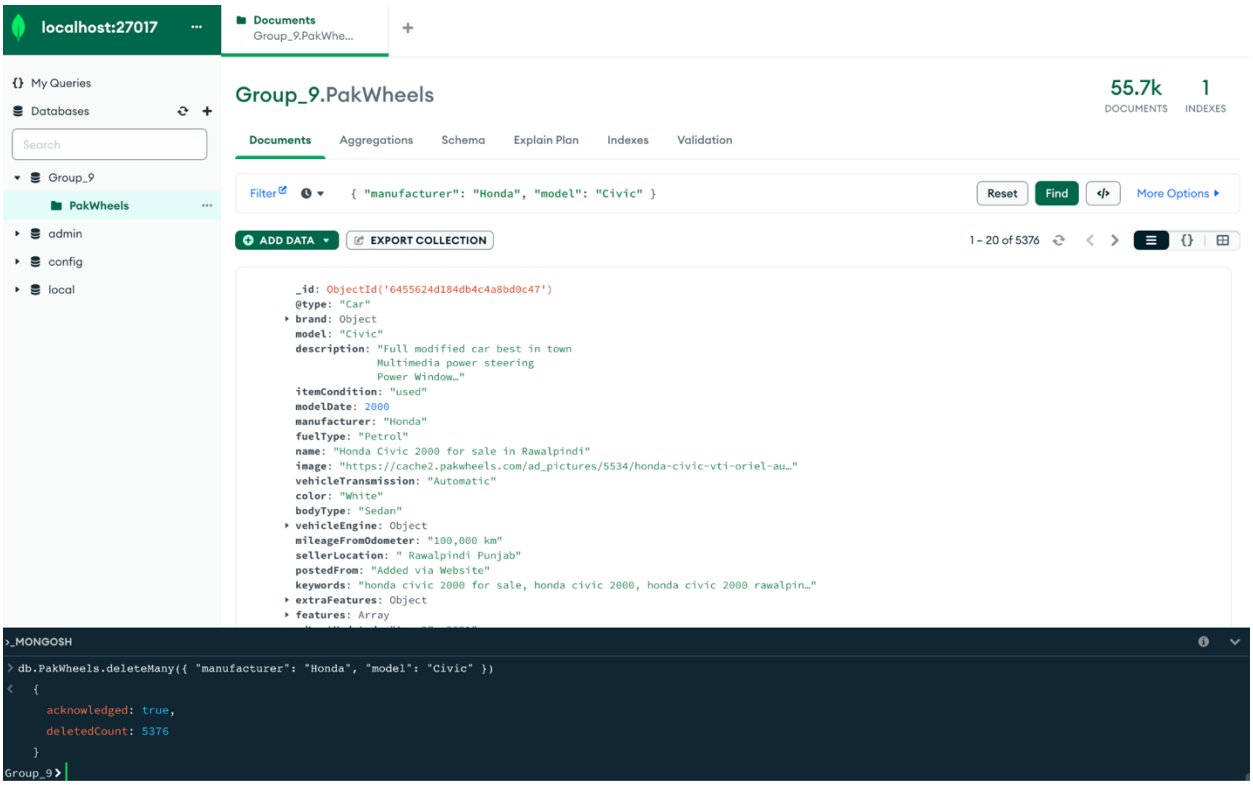
Query 2: Delete all documents where the vehicleCondition is "New":



Query 3: Delete the document where the _id is "6455624d184db4c4a8bd0c37":



Query 4: Delete all documents where the manufacturer is "Honda" and the model is "Civic":



Query 5: Delete all documents where the year is less than or equal to 2010 and the vehicleTransmission is "Automatic":