

# 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: Q 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: Q 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	HULL	HULL		BTREE			YES	HULL
Accident	1	Location_Id_idx	1	Location_Id	A	0	HULL	HULL	YES	BTREE			YES	HULL

### Table GPS:

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

100% ◇ | 17:20 |

Result Grid Filter Rows: Q 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	HULL	HULL		BTREE			YES	HULL

### Table Location:

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

100% ◇ | 17:24 |

Result Grid Filter Rows: Q 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	HULL	HULL		BTREE			YES	HULL
Location	1	idx_city	1	City	A	0	HULL	HULL	YES	BTREE			YES	HULL

### Table City:

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

100% ◇ | 17:16 |

Result Grid Filter Rows: Q 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	HULL	HULL		BTREE			YES	HULL

## Table Weather:

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

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	HULL	HULL		BTREE			YES	HULL
Weather	0	PRIMARY	2	Weather_Timestamp	A	0	HULL	HULL		BTREE			YES	HULL

### 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.

The screenshot shows the MySQL Workbench interface with the 'copied\_database' schema selected in the left sidebar. The main pane displays a query editor containing the following SQL code:

```
1
2 • ALTER TABLE `Accident` DROP INDEX `Location_Id_idx`;
3
4 • ALTER TABLE `Accident` DROP INDEX `PRIMARY`;
5
6 • ALTER TABLE `City` DROP INDEX `City`;
7
8 • ALTER TABLE `City` DROP INDEX `City`;
9
10 • ALTER TABLE `GPS` DROP INDEX `PRIMARY`;
11
12 • ALTER TABLE `Location` DROP INDEX `idx_city`;
13
14 • ALTER TABLE `Location` DROP INDEX `PRIMARY`;
15
16 • ALTER TABLE `Weather` DROP INDEX `PRIMARY`;
17
18
19
```

The status bar at the bottom indicates "Query Completed".

#### 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:

The screenshot shows the SQL Server Management Studio interface. The left pane displays the schema browser for the 'copied\_database'. The 'Tables' node is expanded, showing 'Accident', 'City', 'GPS', 'Location', 'Weather', 'Views', 'Stored Procedures', and 'Functions'. The 'Result Grid' tab is selected, showing the results of a query that joins the 'City' and 'Location' tables. The 'Action Output' tab shows the execution history of the query, detailing each step's duration and fetch time.

```

1
2 • USE copied_database;
3
4 • SELECT distinct c.City,c.State,l.Location_Id,l.County
  FROM City c
    RIGHT JOIN Location l ON c.City= l.City;
7

```

C State	Location_Id	County
L KY	28283	Kenton
A OH	37544	Summit
L KY	74829	Jefferson
A OH	75662	Summit
C OH	83618	Stark
Y OH	91822	Mahoning

Result 108

Action Output

Time	Action	Response	Duration / Fetch Time
21:59:13	USE copied_database	0 row(s) affected	0.00061 sec
21:59:15	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.027 sec / 0.00091 s...
21:59:18	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.0075 sec / 0.00086...
21:59:20	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.0096 sec / 0.00076...
21:59:22	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.0050 sec / 0.00057...
21:59:23	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.025 sec / 0.011 sec
21:59:25	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.028 sec / 0.0012 sec
21:59:26	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.0088 sec / 0.0011 s...
21:59:27	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.0069 sec / 0.0004...
21:59:28	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.015 sec / 0.00077...
21:59:30	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.014 sec / 0.0015 sec
21:59:32	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.0087 sec / 0.00078...
21:59:34	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l ON c.City= l.City LIMIT 0, 1000	1000 row(s) returned	0.022 sec / 0.00077...

Query Completed

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

The screenshot shows the SQL Server Management Studio interface. The left pane displays the schema browser for the 'copied\_database'. The 'Tables' node is expanded, showing 'Accident', 'City', 'GPS', 'Location', 'Weather', 'Views', 'Stored Procedures', and 'Functions'. The 'Result Grid' tab is selected, showing the results of a query that joins the 'Weather' and 'Accident' tables. The 'Action Output' tab shows the execution history of the query, detailing each step's duration and fetch time.

```

1
2 • USE copied_database;
3
4 • SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition
  FROM Weather w
    RIGHT JOIN Accident a ON a.location_Id = w.Location_Id;
6

```

ID	Severity	Temperature	Humidity	Pressure	Weather_Condition
A-100	2	12	77	30.17	
A-1000	2	39	93	30.27	Clear
A-1001	2	49	77	30.3	Scattered Clouds
A-1002	2	52	83	30.3	Mostly Cloudy
A-1003	2	51	80	30.32	Mostly Cloudy
A-1004	2	56	75	30.33	Mostly Cloudy

Result 121

Action Output

Time	Action	Response	Duration / Fetch Time
21:59:54	USE copied_database	0 row(s) affected	0.0099 sec
21:59:57	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.0083 sec / 0.0045...
21:59:57	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.0086 sec / 0.0023...
21:59:59	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.012 sec / 0.0081 s...
22:00:00	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.0052 sec / 0.0017 s...
22:00:01	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.0065 sec / 0.0092...
22:00:03	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.014 sec / 0.00086...
22:00:04	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.011 sec / 0.00084 s...
22:00:05	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.0069 sec / 0.0019...
22:00:08	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.017 sec / 0.00087 ...
22:00:08	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.0046 sec / 0.0011 s...
22:00:09	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.0097 sec / 0.0012 s...
22:00:10	SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accide...	1000 row(s) returned	0.020 sec / 0.0024 sec

Query Completed

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

Project1

Administration Schemas

Schemas copied\_database

```

1
2 • USE copied_database;
3
4 • SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat
5   FROM Accident a
6   JOIN GPS g ON a.Location_Id = g.Location_Id;
7

```

Result Grid

ID	Severity	Start_Lat	Start_Lng	End_Lng	End_Lat
A-874	3	39.0215	-84.5074	-84.4951	39.0226
A-61	2	41.0356	-81.5694	-81.5799	41.036
A-911	3	38.2004	-81.7489	-81.7501	38.2073
A-206	2	41.0189	-81.5101	-81.5059	41.0196
A-375	4	40.8547	-81.4165	-81.3971	40.858
A-829	2	41.1216	-80.7895	-80.7827	41.1233

Result 133

Action Output

Time	Action	Response	Duration / Fetch Time
303 22:00:24	USE copied_database	0 row(s) affected	0.00036 sec
304 22:00:26	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.012 sec / 0.00080 sec
305 22:00:28	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.014 sec / 0.0016 sec
306 22:00:30	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.0057 sec / 0.0019 sec
307 22:00:31	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.008 sec / 0.0011 sec
308 22:00:33	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.013 sec / 0.0012 sec
309 22:00:34	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.0037 sec / 0.0048 sec
310 22:00:36	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.0030 sec / 0.0030 sec
311 22:00:37	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.0035 sec / 0.0058 sec
312 22:00:38	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.0063 sec / 0.0012 sec
313 22:00:40	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.0060 sec / 0.0010 sec
314 22:00:41	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.0063 sec / 0.0023 sec
315 22:00:42	SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	1000 row(s) returned	0.0042 sec / 0.0042 sec

Query Completed

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

Project1

Administration Schemas

Schemas copied\_database

```

1
2 • USE copied_database;
3
4 • SELECT l.City, l.State, a.Severity, a.Description
5   FROM Location l
6   LEFT JOIN Accident a ON l.Location_Id = a.Location_Id
7   GROUP BY l.City, l.State, a.Severity, a.Description;
8
9

```

Result Grid

City	State	Severity	Description
Lima	OH	3	At Whipple Ave NW/Exit 79 and KY-9...
Alton	KY	2	At CH-619/Weston Rd/Exit 17 - Accident
Louisville	KY	3	At Crittenton Dr - Accident
Akron	OH	2	Ramp to I-77 - Accident
Canton	OH	4	Closed between Whipple Ave NW/Exit 109A an...
Youngstown	OH	2	At OH-46/Exit 223 - Accident

Result 145

Action Output

Time	Action	Response	Duration / Fetch Time
316 22:01:08	USE copied_database	0 row(s) affected	0.011 sec
317 22:01:10	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.018 sec / 0.00081 sec
318 22:01:12	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.015 sec / 0.0022 sec
319 22:01:13	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.031 sec / 0.0023 sec
320 22:01:14	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.014 sec / 0.0012 sec
321 22:01:15	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.012 sec / 0.0050 sec
322 22:01:17	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.015 sec / 0.0029 sec
323 22:01:18	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.0089 sec / 0.0014 sec
324 22:01:19	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.015 sec / 0.0013 sec
325 22:01:20	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.013 sec / 0.00098 sec
326 22:01:23	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.0061 sec / 0.00089 sec
327 22:01:24	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.013 sec / 0.0012 sec
328 22:01:25	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.013 sec / 0.0011 sec

Query Completed

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

```

1
2 • USE copied_database;
3
4 • SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition
5   FROM Location l
6   LEFT JOIN Weather w ON l.Location_Id = w.Location_Id
7   GROUP BY l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition;
8
9

```

Result Grid

City	State	Temperature	Humidity	Weather_Condition
Latonia	KY	58	90	Mostly Cloudy
Akron	OH	21	74	Light Snow
Louisville	KY	61	42	Partly Cloudy
Akron	OH	28	81	Overcast
Canton	OH	31	67	Partly Cloudy
Youngstown	OH	42	76	Overcast

Action Output

Action	Time	Response	Duration / Fetch Time
329	22:01:49	USE copied_database	0.0014 sec
330	22:01:52	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.0012 sec
331	22:01:53	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.00073...
332	22:01:54	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.00072 s...
333	22:01:56	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.00050 s...
334	22:01:57	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.00089...
335	22:01:58	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.00068...
336	22:02:00	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.00071...
337	22:02:01	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.00051 s...
338	22:02:02	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.0010 sec
339	22:02:04	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.00055...
340	22:02:05	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.0004...
341	22:02:08	SELECT i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location i LEFT JOIN Weather w ON i.Location_Id = w.Location_Id GROUP BY i.City, i.State, w.Temperature, w.Humidity, w.Weather_Condition	0.00077 s...

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)

```

1
2 • USE new_schema;
3 • SELECT distinct c.City,c.State,l.Location_Id,l.County
4   FROM City c
5   RIGHT JOIN Location l on c.City = l.City;
6
7
8

```

Result Grid

City	State	Location_Id	County
Latonia	KY	28283	Kenton
Akron	OH	37544	Summit
Louisville	KY	74829	Jefferson

Action Output

Action	Time	Response	Duration / Fetch Time
207	21:43:05	USE new_schema	0.0064 sec
208	21:43:07	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
209	21:43:08	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
210	21:43:09	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
211	21:43:11	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
212	21:43:12	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
213	21:43:13	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
214	21:43:15	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
215	21:43:16	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
216	21:43:18	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
217	21:43:19	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
218	21:43:21	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned
219	21:43:23	SELECT distinct c.City,c.State,l.Location_Id,l.County FROM City c RIGHT JOIN Location l on c.City = l.City LIMIT 0, 1000	1000 row(s) returned

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

Schemas

```

1
2 • USE new_schema;
3 • SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition
4 FROM Weather w
5 RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;
6
7
8

```

Tables

ID	Severity	Temperature	Humidity	Pressure	Weather_Condition
A-100	2	12	77	30.17	
A-1000	2	39	93	30.27	Clear
A-1001	2	48	77	30.3	Scattered Clouds

Action Output

Action	Time	Response	Duration / Fetch Time
USE new_schema	21:46:07	0 row(s) affected	0.00044 sec
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:12	1000 row(s) returned	0.0043 sec / 0.0038...
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:14	1000 row(s) returned	0.010 sec / 0.0082 sec
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:16	1000 row(s) returned	0.0098 sec / 0.00079...
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:17	1000 row(s) returned	0.0061 sec / 0.0025...
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:18	1000 row(s) returned	0.0040 sec / 0.0066...
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:20	1000 row(s) returned	0.031 sec / 0.0016 sec
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:21	1000 row(s) returned	0.0051 sec / 0.0044...
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:23	1000 row(s) returned	0.0081 sec / 0.0047...
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:24	1000 row(s) returned	0.010 sec / 0.0029 sec
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:26	1000 row(s) returned	0.0053 sec / 0.0045...
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:28	1000 row(s) returned	0.0076 sec / 0.0010 s...
SELECT a.ID,a.Severity,w.Temperature,w.Humidity,w.Pressure,w.Weather_Condition FROM Weather w RIGHT JOIN Accident a ON a.Location_Id = w.Location_Id;	21:46:30	1000 row(s) returned	0.012 sec / 0.0089...

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

Schemas

```

1
2 • USE new_schema;
3 • SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat
4 FROM Accident a
5 JOIN GPS g ON a.Location_Id = g.Location_Id;
6
7
8

```

Action Output

Action	Time	Response	Duration / Fetch Time
USE new_schema	21:49:18	0 row(s) affected	0.00039 sec
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:21	1000 row(s) returned	0.0088 sec / 0.0028...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:22	1000 row(s) returned	0.0037 sec / 0.0020...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:24	1000 row(s) returned	0.0047 sec / 0.0042...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:25	1000 row(s) returned	0.0044 sec / 0.0059...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:27	1000 row(s) returned	0.0070 sec / 0.0030...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:28	1000 row(s) returned	0.0058 sec / 0.0033...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:29	1000 row(s) returned	0.0044 sec / 0.0030...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:30	1000 row(s) returned	0.0092 sec / 0.0028...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:32	1000 row(s) returned	0.0035 sec / 0.0053...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:33	1000 row(s) returned	0.0062 sec / 0.0081...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:35	1000 row(s) returned	0.0084 sec / 0.0035...
SELECT a.ID,a.Severity,g.Start_Lat,g.Start_Lng,g.End_Lng,g.End_Lat FROM Accident a JOIN GPS g ON a.Location_Id =...	21:49:36	1000 row(s) returned	0.0034 sec / 0.0054...

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

Project1

Administration Schemas

SCHEMAS copied\_database

Tables Accident City GPS Location Weather Views Stored Procedures Functions DB new\_schema

1  
2 • USE new\_schema;  
3  
4  
5 • SELECT l.City, l.State, a.Severity, a.Description  
6 FROM Location l  
7 LEFT JOIN Accident a ON l.Location\_Id = a.Location\_Id  
8 GROUP BY l.City, l.State, a.Severity, a.Description;

100% 53:8

Result Grid Filter Rows: Search Export:

City	State	Severity	Description
Latonia	KY	3	Between KY-16/Taylor Mill Rd/Exit 79 and KY-9/...
Akron	OH	2	At OH-619/Wooster Rd/Exit 17 - Accident.
Louisville	KY	3	At Criminal Dr - Accident.
Akron	OH	2	From 10 to 17 -
Canton	OH	4	Closed between Whipple Ave NW/Exit 109A an...
Youngstown	OH	2	At OH-46/Exit 223 - Accident.

Result 83

Action Output

Action	Time	Action	Response	Duration / Fetch Time
249	21:56:52	USE new_schema	0 row(s) affected	0.0015 sec
250	21:56:54	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.023 sec / 0.00088...
251	21:56:56	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.034 sec / 0.0012 sec
252	21:56:57	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.028 sec / 0.0015 sec
253	21:56:59	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.033 sec / 0.0046 sec
254	21:57:01	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.023 sec / 0.0056 sec
255	21:57:02	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.029 sec / 0.0019 sec
256	21:57:04	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.014 sec / 0.0012 sec
257	21:57:05	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.024 sec / 0.0021 sec
258	21:57:07	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.033 sec / 0.0012 sec
259	21:57:08	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.021 sec / 0.0010 sec
260	21:57:09	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.020 sec / 0.00088...
261	21:57:11	SELECT l.City, l.State, a.Severity, a.Description FROM Location l LEFT JOIN Accident a ON l.Location_Id = a.Location_Id...	877 row(s) returned	0.012 sec / 0.0023 sec

Query Completed

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

Project1

Administration Schemas

SCHEMAS copied\_database

Tables Accident City GPS Location Weather Views Stored Procedures Functions DB new\_schema

1  
2 • USE new\_schema;  
3  
4 • SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather\_Condition  
5 FROM Location l  
6 LEFT JOIN Weather w ON l.Location\_Id = w.Location\_Id  
7 GROUP BY l.City, l.State, w.Temperature, w.Humidity, w.Weather\_Condition;

100% 74:7

Result Grid Filter Rows: Search Export:

City	State	Temperature	Humidity	Weather_Conditi...
Latonia	KY	93	90	Partly Cloudy
Akron	OH	21	74	Light Snow
Louisville	KY	61	42	Partly Cloudy
Akron	OH	28	81	Overcast
Canton	OH	31	67	Partly Cloudy
Youngstown	OH	42	76	Overcast

Result 96

Action Output

Action	Time	Action	Response	Duration / Fetch Time
262	21:57:51	USE new_schema	0 row(s) affected	0.017 sec
263	21:57:54	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.050 sec / 0.00068...
264	21:57:57	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.018 sec / 0.00075 s...
265	21:57:58	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.017 sec / 0.0017 sec
266	21:57:58	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.016 sec / 0.00063...
267	21:57:59	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.017 sec / 0.00073 s...
268	21:58:00	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.025 sec / 0.0019 sec
269	21:58:00	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.027 sec / 0.0016 sec
270	21:58:01	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.019 sec / 0.0011 sec
271	21:58:02	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.022 sec / 0.0016 sec
272	21:58:02	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.012 sec / 0.00077 s...
273	21:58:03	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.021 sec / 0.00072 s...
274	21:58:03	SELECT l.City, l.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location l LEFT JOIN Weather w ON l.Location_Id = w.Location_Id...	922 row(s) returned	0.040 sec / 0.0018 sec

Query Completed

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 I.City, I.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location I LEFT JOIN Weather w ON I.LocationID = w.LocationID; 922 row(s) returned 0.0009 sec / 0.0005...
340 22:02:05 SELECT I.City, I.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location I LEFT JOIN Weather w ON I.LocationID = w.LocationID; 922 row(s) returned 0.0069 sec / 0.0004...
341 22:02:08 SELECT I.City, I.State, w.Temperature, w.Humidity, w.Weather_Condition FROM Location I LEFT JOIN Weather w ON I.LocationID = w.LocationID; 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.

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Schemas' tree shows 'copied\_database' and 'new\_schema'. In the main area, a query editor window displays:

```

1 • USE new_schema;
2
3 • SELECT *
4   FROM Location
5   WHERE City = 'Louisville';
6
7
8
9
10
11
12
13

```

Below the query editor is a 'Result Grid' table titled 'Location 176' with 176 rows. The columns are: ID, Severity, Start\_Time, End\_Time, Description, City, State, County, Zipcode, Start\_Lat, Start\_Lng, End\_Lat, End\_Lng, Weather, and Accid. The data includes various accident entries in Louisville, KY.

At the bottom, an 'Action Output' table lists the execution history:

Time	Action	Response	Duration / Fetch Time
353 22:12:20	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.00044 sec
354 22:12:23	USE new_schema	0 row(s) affected	0.011 sec / 0.000059...
355 22:12:24	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.030 sec / 0.0000...
356 22:12:25	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.0061 sec / 0.00006...
357 22:12:26	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.0089 sec / 0.0000...
358 22:12:28	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.0099 sec / 0.0001...
359 22:12:29	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.0014 sec / 0.00007...
360 22:12:30	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.0029 sec / 0.00006...
361 22:12:31	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.0024 sec / 0.00006...
362 22:12:32	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.0023 sec / 0.00004...
363 22:12:33	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.0033 sec / 0.00041...
364 22:12:34	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.0014 sec / 0.00009...
365 22:12:36	SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	43 row(s) returned	0.0035 sec / 0.0000...

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

Project1

Administration Schemas

SCHEMAS

copied\_database

- Tables
  - Accident
  - City
  - GPS
  - Location
  - Weather
- Views
- Stored Procedures
- Functions
- DB

new\_schema

- Tables
- Views
- Stored Procedures
- Functions
- P
- sys

Query 1

```

1 • USE new_schema;
2
3 • SELECT *
4   FROM Accident
5   WHERE Severity > 3;
6
7
8
9
10
11
12
13

```

Result Grid

ID	Severity	Start_Time	End_Time	Description	Location_Id
A-138	4	2016-02-12 16:15:51	2016-02-12 22:15:51	Closed between M-50/Dale Hwy/Exit 15 and N...	5097390
A-139	4	2016-02-12 19:29:53	2016-02-13 01:29:53	Closed at US-19/Exit 15 - Road closed due to a...	9175070
A-140	4	2016-02-12 22:22:57	2016-02-13 04:22:57	Closed between US-35/IN-28/Exit 45 and IN-26...	1426600

Action Output

Time	Action	Response	Duration / Fetch Time
360 22:13:12	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	0 row(s) affected	0.00058 sec
361 22:13:12	USE new_schema	129 row(s) returned	0.015 sec / 0.00014 sec
362 22:13:16	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.021 sec / 0.00010 sec
363 22:13:17	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.028 sec / 0.00008 sec
364 22:13:18	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.035 sec / 0.00015 sec
365 22:13:19	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.0372 sec / 0.00021 sec
366 22:13:19	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.041 sec / 0.00014 sec
367 22:13:21	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.043 sec / 0.00015 sec
368 22:13:22	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.045 sec / 0.00014 sec
369 22:13:23	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.047 sec / 0.00014 sec
370 22:13:24	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.051 sec / 0.00020 sec
371 22:13:24	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.0519 sec / 0.00008 sec
372 22:13:25	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.0519 sec / 0.00008 sec
373 22:13:25	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.0519 sec / 0.00008 sec
374 22:13:25	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.0519 sec / 0.00008 sec
375 22:13:25	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.0519 sec / 0.00008 sec
376 22:13:25	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.0519 sec / 0.00008 sec
377 22:13:25	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.0519 sec / 0.00008 sec
378 22:13:25	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.0519 sec / 0.00008 sec
379 22:13:25	SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	129 row(s) returned	0.0519 sec / 0.00008 sec

Query Completed

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

Project1

Administration Schemas

SCHEMAS

copied\_database

- Tables
  - Accident
  - City
  - GPS
  - Location
  - Weather
- Views
- Stored Procedures
- Functions
- DB

new\_schema

- Tables
- Views
- Stored Procedures
- Functions
- P
- sys

Query 1

```

1 • USE new_schema;
2
3 • SELECT *
4   FROM Weather
5   WHERE `Temperature` < 50;
6
7
8
9
10
11
12
13

```

Result Grid

ID	Severity	Start_Time	End_Time	Description	City	State	County	Zipcode	Start_Lat	Start_Lng	End_Lat	End_Lng	Weather_Timestamp	Wind_Direction
A-246	2	2016-02-17 07:22:54	2016-02-17 12:22:54	Temp to 177 - Accident.	Akron	OH	Summit	44321	41.1189	-81.6541	41.1268	-81.6522	2016-02-17 08:54:00	W
A-375	4	2016-02-22 21:44:41	2016-02-23 03:44:41	Closed between Whipple Ave NW/Exit 109A an...	Canton	OH	Stark	44709	40.8547	-81.4165	40.828	-81.3971	2016-02-22 21:51:00	N
A-829	2	2016-03-11 17:46:19	2016-03-11 23:46:19	At OH-46/Exit 223 - Accident.	Youngstown	OH	Mahoning	44515	41.1216	-80.7895	41.1233	-80.7827	2016-03-11 17:51:00	W

Action Output

Time	Action	Response	Duration / Fetch Time
386 22:14:21	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0024 sec / 0.011 sec
387 22:14:22	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0026 sec / 0.0045 sec
388 22:14:23	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0035 sec / 0.0085 sec
389 22:14:24	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0030 sec / 0.0066 sec
390 22:14:25	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0057 sec / 0.0054 sec
391 22:14:25	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0038 sec / 0.0048 sec
392 22:14:26	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0053 sec / 0.0051 sec
393 22:14:27	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0020 sec / 0.012 sec
394 22:14:27	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.022 sec / 0.0056 sec
395 22:14:28	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0091 sec / 0.009 sec
396 22:14:28	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0020 sec / 0.0062 sec
397 22:14:30	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.016 sec / 0.0030 sec
398 22:14:31	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0027 sec / 0.0060 sec

Query Completed

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

Project1

Administration Schemas

Schemas copied\_database

```

1 • USE new_schema;
2
3 • SELECT *
4   FROM GPS
5   WHERE Start_Lat > 40;
6
7
8
9
10
11
12
13

```

Result Grid

ID	Severity	Start_Time	End_Time	Description	City	State	County	Zipcode	Start_Lat	Start_Lng	End_Lat	End_Lng	Weather_Timestamp
A-375	4	2016-02-22 21:44:41	2016-02-23 03:44:41	Closed between Whipple Ave NW/Exit 109A an...	Canton	OH	Stark	44709	40.8547	-81.4165	40.828	-81.3971	2016-02-22 21:51:00
A-829	2	2016-03-11 17:46:19	2016-03-11 23:46:19	At OH-46/Exit 223 - Accident.	Youngstown	OH	Mahoning	44515	41.1216	-80.7895	41.1233	-80.7827	2016-03-11 17:51:00
A-823	2	2016-03-11 16:43:02	2016-03-11 22:43:02	At ML King Jr Dr/Exit 177 - Accident.	Cleveland	OH	Cuyahoga	44103	41.5391	-81.6335	41.5393	-81.6288	2016-03-11 16:35:00

GPS 218

Action Output

Time	Action	Response	Duration / Fetch Time
40U 22:14:48	USE new_schema	0 row(s) affected	0.0014 sec
401 22:14:49	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0062 sec / 0.0090...
402 22:14:50	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0018 sec / 0.0036...
403 22:14:50	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.016 sec / 0.0022 sec
404 22:14:51	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.012 sec / 0.0045 sec
405 22:14:52	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.026 sec / 0.0018 sec
406 22:14:53	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0020 sec / 0.012 sec
407 22:14:54	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0080 sec / 0.0038...
408 22:14:54	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0039 sec / 0.0048...
409 22:14:55	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0017 sec / 0.0046 ...
410 22:14:56	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0018 sec / 0.011 sec
411 22:14:57	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0028 sec / 0.0040...
412 22:14:58	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0086 sec / 0.0019...
413 22:14:59	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0077 sec / 0.0032...

Query Completed

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

Project1

Administration Schemas

Schemas copied\_database

```

1 • USE new_schema;
2
3 • SELECT *
4   FROM GPS
5   WHERE Start_Lat BETWEEN 40 AND 42;
6
7
8
9
10
11
12
13

```

Result Grid

ID	Severity	Start_Time	End_Time	Description	City	State	County	Zipcode	Start_Lat	Start_Lng	End_Lat	End_Lng	Weather_Timestamp
A-375	4	2016-02-22 21:44:41	2016-02-23 03:44:41	Closed between Whipple Ave NW/Exit 109A an...	Canton	OH	Stark	44709	40.8547	-81.4165	40.828	-81.3971	2016-02-22 21:51:00
A-829	2	2016-03-11 17:46:19	2016-03-11 23:46:19	At OH-46/Exit 223 - Accident.	Youngstown	OH	Mahoning	44515	41.1216	-80.7895	41.1233	-80.7827	2016-03-11 17:51:00
A-823	2	2016-03-11 16:43:02	2016-03-11 22:43:02	At ML King Jr Dr/Exit 177 - Accident.	Cleveland	OH	Cuyahoga	44103	41.5391	-81.6335	41.5393	-81.6288	2016-03-11 16:35:00

GPS 234

Action Output

Time	Action	Response	Duration / Fetch Time
40U 22:17:20	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0000 sec / 0.0000...
418 22:17:20	USE new_schema	0 row(s) affected	0.016 sec
419 22:17:24	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.011 sec / 0.0019 sec
420 22:17:25	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0014 sec / 0.0095...
421 22:17:25	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0069 sec / 0.0041...
422 22:17:26	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0070 sec / 0.0061 s...
423 22:17:27	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0021 sec / 0.0059 ...
424 22:17:28	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0041 sec / 0.0044...
425 22:17:29	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0031 sec / 0.0038...
426 22:17:30	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0012 sec / 0.0067 s...
427 22:17:31	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0018 sec / 0.0058...
428 22:17:33	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0049 sec / 0.0029...
429 22:17:36	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0087 sec / 0.0019 ...
430 22:17:38	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0038 sec / 0.0057...

Query Completed

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.**

The screenshot shows the SQL Server Management Studio interface. A query window displays the following T-SQL code:

```

1 • USE copied_database;
2
3
4 • SELECT *
5   FROM Location
6   WHERE City = 'Louisville';
7
8
9
10
11
12
13

```

The results grid shows 246 rows from the Location table where City is Louisville. The Action Output pane below the results grid displays the execution history for this query, showing 433 individual actions with their corresponding times and responses.

Action	Time	Response	Duration / Fetch Time
SELECT * FROM Location WHERE City = 'Louisville';	22:19:56	0 row(s) affected	0.00093 sec
USE copied_database			
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:01	43 row(s) returned	0.0020 sec / 0.00006...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:04	43 row(s) returned	0.0023 sec / 0.00006...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:05	43 row(s) returned	0.0070 sec / 0.00006...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:06	43 row(s) returned	0.0028 sec / 0.00004...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:10	43 row(s) returned	0.026 sec / 0.00007...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:12	43 row(s) returned	0.0065 sec / 0.00007...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:13	43 row(s) returned	0.0010 sec / 0.00006...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:14	43 row(s) returned	0.0010 sec / 0.00005...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:16	43 row(s) returned	0.0027 sec / 0.00006...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:18	43 row(s) returned	0.0016 sec / 0.00005...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:20	43 row(s) returned	0.0030 sec / 0.0000...
SELECT * FROM Location WHERE City = 'Louisville' LIMIT 0, 1000	22:20:22	43 row(s) returned	0.0014 sec / 0.00005...

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

The screenshot shows the SQL Server Management Studio interface. A query window displays the following T-SQL code:

```

1 • USE copied_database;
2
3
4 • SELECT *
5   FROM Accident
6   WHERE Severity > 3;
7
8
9
10
11
12
13

```

The results grid shows 259 rows from the Accident table where Severity is greater than 3. The Action Output pane below the results grid displays the execution history for this query, showing 457 individual actions with their corresponding times and responses.

Action	Time	Response	Duration / Fetch Time
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:20:49	0 row(s) returned	0.0025 sec
USE copied_database			
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:20:54	129 row(s) returned	0.0086 sec / 0.00013...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:20:58	129 row(s) returned	0.014 sec / 0.00016...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:20:57	129 row(s) returned	0.013 sec / 0.00015 s...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:20:57	129 row(s) returned	0.0087 sec / 0.00011...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:20:58	129 row(s) returned	0.0021 sec / 0.00021...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:21:00	129 row(s) returned	0.0055 sec / 0.00016...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:21:01	129 row(s) returned	0.0048 sec / 0.0005...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:21:02	129 row(s) returned	0.0030 sec / 0.00013...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:21:03	129 row(s) returned	0.0040 sec / 0.00015...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:21:04	129 row(s) returned	0.0045 sec / 0.00014...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:21:06	129 row(s) returned	0.016 sec / 0.00015 s...
SELECT * FROM Accident WHERE Severity > 3 LIMIT 0, 1000	22:21:08	129 row(s) returned	0.0078 sec / 0.00028...

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

Project1

Administration Schemas Query 1 SQL File 3\* 5\* 4\* SQL File 6\* SQL File 7\* SQL File 8\* SQL File 9\* SQL File 10\*

**SCHEMAS**

copied\_database

- Tables
  - Accident
  - City
  - GPS
  - Location
  - Weather
- Views
- Stored Procedures
- Functions

DB

new\_schema

- Tables
- Views
- Stored Procedures
- Functions

P

sys

1 • USE copied\_database;

2

3

4 • SELECT \*

5 FROM Weather

6 WHERE `Temperature` < 50;

7

8

9

10

11

12

13

Result Grid Filter Rows: Search Export:

100% 26:6

ID	Severity	Start_Time	End_Time	Description	City	State	County	Zipcode	Start_Lat	Start_Lng	End_Lat	End_Lng	Weather_Timestamp	Weather_Condition
A-246	2	2016-02-17 07:22:54	2016-02-17 13:22:54	Ramp to I-77 - Accident.	Akron	OH	Summit	44321	41.189	-81.6541	41.268	-81.6522	2016-02-17 06:54:00	Overcast
A-375	4	2016-02-22 21:44:41	2016-02-23 03:44:41	Closed between Whipple Ave NW/Exit 109A an...	Canton	OH	Stark	44709	40.8547	-81.4165	40.828	-81.3971	2016-02-22 21:51:00	Partly Cloudy
A-829	2	2016-03-11 17:46:19	2016-03-11 23:46:19	At OH-46/Exit 223 - Accident.	Youngstown	OH	Mahoning	44515	41.1216	-80.7895	41.1233	-80.7827	2016-03-11 17:51:00	Cloudy

Weather 271

Action Output

Time	Action	Response	Duration / Fetch Time
450 22:21:24	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.0000...
458 22:21:24	USE copied_database	0 row(s) affected	0.0016 sec
459 22:21:25	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0097 sec / 0.0059...
460 22:21:27	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.011 sec / 0.0084 sec
461 22:21:28	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0032 sec / 0.0054...
462 22:21:29	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.019 sec / 0.0090...
463 22:21:29	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0050 sec / 0.0076...
464 22:21:30	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0021 sec / 0.017 sec
465 22:21:31	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0020 sec / 0.0085...
466 22:21:32	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0013 sec / 0.017 sec
467 22:21:33	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0087 sec / 0.0051 sec
468 22:21:34	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0021 sec / 0.0070 sec
469 22:21:35	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.018 sec / 0.0074 sec
470 22:21:36	SELECT * FROM Weather WHERE `Temperature` < 50 LIMIT 0, 1000	577 row(s) returned	0.0098 sec / 0.0064...

Query Completed

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

Project1

Administration Schemas Query 1 SQL File 3\* 5\* 4\* SQL File 6\* SQL File 7\* SQL File 8\* SQL File 9\* SQL File 10\*

**SCHEMAS**

copied\_database

- Tables
  - Accident
  - City
  - GPS
  - Location
  - Weather
- Views
- Stored Procedures
- Functions

DB

new\_schema

- Tables
- Views
- Stored Procedures
- Functions

P

sys

1 • USE copied\_database;

2

3

4 • SELECT \*

5 FROM GPS

6 WHERE Start\_Lat > 40;

7

8

9

10

11

12

13

Result Grid Filter Rows: Search Export:

100% 22:6

ID	Severity	Start_Time	End_Time	Description	City	State	County	Zipcode	Start_Lat	Start_Lng	End_Lat	End_Lng	Weather_Timestamp	Weather_Condition
A-375	4	2016-02-22 21:44:41	2016-02-23 03:44:41	Closed between Whipple Ave NW/Exit 109A an...	Canton	OH	Stark	44709	40.8547	-81.4165	40.828	-81.3971	2016-02-22 21:51:00	Partly Cloudy
A-829	2	2016-03-11 17:46:19	2016-03-11 23:46:19	At OH-46/Exit 223 - Accident.	Youngstown	OH	Mahoning	44515	41.1216	-80.7895	41.1233	-80.7827	2016-03-11 17:51:00	Cloudy
A-823	2	2016-03-11 16:43:02	2016-03-11 22:43:02	At ML King Jr Dr/Exit 177 - Accident.	Cleveland	OH	Cuyahoga	44103	41.5391	-81.6335	41.5393	-81.6288	2016-03-11 16:55:00	Cloudy

GPS 284

Action Output

Time	Action	Response	Duration / Fetch Time
472 22:21:54	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	0 row(s) returned	0.0000 sec / 0.0000...
473 22:21:56	USE copied_database	0 row(s) affected	0.00051 sec
474 22:21:58	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.013 sec / 0.012 sec
475 22:21:59	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0071 sec / 0.0013 s...
476 22:22:00	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0019 sec / 0.0034...
477 22:22:01	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0023 sec / 0.0020...
478 22:22:02	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0053 sec / 0.0070...
479 22:22:04	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0049 sec / 0.0042...
480 22:22:05	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0015 sec / 0.0059...
481 22:22:08	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0013 sec / 0.0047 s...
482 22:22:09	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0072 sec / 0.0037...
483 22:22:10	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.0020 sec / 0.0058...
484 22:22:11	SELECT * FROM GPS WHERE Start_Lat > 40 LIMIT 0, 1000	359 row(s) returned	0.023 sec / 0.0017 sec

Query Completed

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

```

USE copied_database;
SELECT *
FROM GPS
WHERE Start_Lat BETWEEN 40 AND 42;

```

ID	Severity	Start_Time	End_Time	Description	City	State	County	Zipcode	Start_Lat	Start_Lng	End_Lat	End_Lng	Weather_Timestamp
A-375	4	2016-02-22 21:44:41	2016-02-23 03:44:41	Closed between Whipple Ave NW/Exit 109A an...	Canton	OH	Stark	44709	40.8547	-81.4165	40.8628	-81.3971	2016-02-22 21:51:00
A-829	2	2016-03-11 17:46:19	2016-03-11 23:46:19	At OH-46/Exit 223 - Accident.	Youngstown	OH	Mahoning	44515	41.1216	-80.7895	41.1233	-80.7827	2016-03-11 17:51:00
A-823	2	2016-03-11 16:43:02	2016-03-11 22:43:02	At Ml. King Jr Dr/Exit 177 - Accident.	Cleveland	OH	Cuyahoga	44103	41.5391	-81.6335	41.5393	-81.6288	2016-03-11 16:35:00

GPS 296

Action	Time	Action	Response	Duration / Fetch Time
485	22:22:24	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	0 row(s) affected	0.00038 sec
486	22:22:27	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0098 sec / 0.0066...
487	22:22:28	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0090 sec / 0.0021...
488	22:22:29	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0018 sec / 0.011 sec
489	22:22:30	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0012 sec / 0.0026 s...
490	22:22:31	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0029 sec / 0.0036...
491	22:22:32	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0025 sec / 0.0038...
492	22:22:33	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0020 sec / 0.0045...
493	22:22:34	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0027 sec / 0.0044...
494	22:22:35	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0014 sec / 0.0031 s...
495	22:22:37	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.014 sec / 0.0097 sec
496	22:22:37	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0022 sec / 0.0064...
497	22:22:39	SELECT * FROM GPS WHERE Start_Lat BETWEEN 40 AND 42 LIMIT 0, 1000	359 row(s) returned	0.0030 sec / 0.0047...

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.

The screenshot shows the MongoDB Compass interface connected to the database 'Group\_9'. The 'PakWheels' collection is selected. The top right corner indicates there are 55.7k documents and 1 index. A document preview is shown with fields like '\_id', 'type', 'brand', 'model', 'description', 'itemCondition', 'modelDate', 'manufacturer', 'fuelType', 'name', 'image', 'vehicleTransmission', 'color', 'bodyType', 'vehicleEngine', 'mileageFromOdometer', 'sellerLocation', 'postedFrom', 'keywords', 'extraFeatures', 'features', 'adLastUpdated', 'price', and 'priceCurrency'. Below the document preview, a green checkmark icon and the text 'Import completed. 55675 documents written.' are displayed. The bottom status bar shows the connection is to 'MONGOSH'.

## 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" })
```

The screenshot shows the MongoDB Compass interface. On the left, the sidebar lists databases: 'localhost:27017' (selected), 'Databases' (Group\_9.PakWheels), 'Group\_9' (PakWheels selected), 'admin', 'config', and 'local'. The main area displays the 'Group\_9.PakWheels' collection. At the top right, it shows '55.7k DOCUMENTS' and '1 INDEXES'. Below the header, there are tabs for 'Documents', 'Aggregations', 'Schema', 'Explain Plan', 'Indexes', and 'Validation'. A 'Filter' dropdown is set to '{ "modelDate": 2000, "manufacturer": "Honda" }'. The results pane shows one document with the following details:

```
_id: ObjectId('6455624d184db4c4a8bd0c47')
@type: 'Car'
brand: Object
model: 'Civic'
description: "Full modified car best in town
    Multimedia power steering
    Power Window..."
itemCondition: "used"
modelDate: 2000
manufacturer: "Honda"
fuelType: "Petrol"
name: "Honda Civic 2000 for sale in Rawalpindi"
image: "https://cache2.pakwheels.com/ad_pictures/5534/honda-civic-vti-oriental-au...
vehicleTransmission: "Automatic"
color: "White"
bodyType: "Sedan"
vehicleEngine: Object
mileageFromOdometer: "100,000 km"
sellerLocation: " Rawalpindi Punjab"
postedFrom: "Added via Website"
keywords: "honda civic 2000 for sale, honda civic 2000, honda civic 2000 rawalpindi"
```

At the bottom, the MONGOSH command line shows the query:

```
> db.PakWheels.find({
  "modelDate": 2000, "manufacturer": "Honda"})
< {
  _id: ObjectId("6455624d184db4c4a8bd0c47"),
  '@type': 'Car',
  brand: {
    '@type': 'Brand',
    name: 'Honda'
```

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

```

  {
    "_id": ObjectId('6455624d184db4c4a8bd0c84'),
    "@type": "Car",
    brand: Object,
    model: "Civic",
    description: "bumper, to bumper original car 100 by 100 Lightweight allow rims . Ori_",
    itemCondition: "used",
    modelDate: 2006,
    manufacturer: "Honda",
    fuelType: "Petrol",
    name: "Honda Civic 2006 for sale in Lahore",
    image: "https://cache2.pakwheels.com/ad_pictures/5661/honda-civic-vti-1-8-i-vt...",
    vehicleTransmission: "Manual",
    color: "White",
    bodyType: "Sedan",
    vehicleEngine: Object,
    mileageFromOwner: "120,000 km",
    sellerLocation: "Bund Road, Lahore Punjab",
    postedFrom: "Added via Phone",
    keywords: "honda civic 2006 for sale, honda civic 2006, honda civic 2006 lahore, ...",
    extraFeatures: Object,
    features: Array
  }
  > MONGOSH
  Type "it" for more
  > db.PakWheels.find({"model":{$in:["Corolla", "City ", "Vitz ", "Civic"]}})
  < [
      {
        _id: ObjectId("6455624d184db4c4a8bd0c39"),
        '@type': 'Car',
        brand: {
          '@type': 'Brand',
          name: 'Toyota'
        }
      }
    ]
  
```

**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" }
  ]
})
  
```

```

  {
    "$and": [
      { "modelDate": { $in: [2000, 2020] } },
      { "manufacturer": { $in: ["Toyota", "Honda"] } },
      { "model": { $in: ["Corolla", "City", "Vitz", "Civic"] } },
      { "color": "Silver" }
    ]
  }
  > MONGOSH
  > 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"] } }  
    ]  
})
```

localhost:27017 ... Documents Group\_9.PakWheels 55.7k 1 DOCUMENTS INDEXES

My Queries Databases +

Search

Group\_9 PakWheels ...

admin config local

Documents Aggregations Schema Explain Plan Indexes Validation

Filter ⚙ ⚙ { \$and: [  
 { "vehicleTransmission": "Manual" },  
 { "color": "Black" },  
 { "extraFeatures.Assembly": "Local" },  
 { "modelDate": { \$gte: 2015, \$lte: 2020 } },  
 { "features": { \$in: ["Navigation System", "Air Conditioning"] } }  
]

Reset Find More Options ↗

ADD DATA EXPORT COLLECTION 1 - 20 of 524

extraFeatures: Object  
 RegisteredIn: "Lahore"  
 Color: "Black"  
 Assembly: "Local"  
 EngineCapacity: "1380 cc"  
 BodyType: "Sedan"  
 LastUpdated: "Aug 27, 2021"  
 AdRef#: "53640909"  
features: Array  
 adlastUpdated: "Aug 27, 2021"  
 price: "2500000"  
 priceCurrency: "PKR"

> MONGOSH

```
> 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 } }  
    ]  
})
```

The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays databases: My Queries, Databases (with Group\_9 selected), and PakWheels. The main area is titled "Group\_9.PakWheels" and shows a search bar with a filter: { \$and: [ { "itemCondition": "used" }, { "modelDate": { \$gt: 2015 } }, { "fuelType": "Petrol" }, { "vehicleEngine.engineDisplacement": "1600cc" }, { "price": { \$lt: 3500000 } } ]}. Below the filter, there are "ADD DATA" and "EXPORT COLLECTION" buttons. The results pane shows a single document: \_id: ObjectId('6455624d184db4c4a8bd0c70'), @type: "Car", brand: Object, model: "Corolla", description: "Toyota Altis 1.6 original condition Lightweight allow rims . Authorized...", itemCondition: "used", modelDate: 2019, manufacturer: "Toyota", fuelType: "Petrol", name: "Toyota Corolla 2019 for sale in Lahore", image: "https://cache2.pakwheels.com/ad\_pictures/5661/toyota-corolla-altis-1-6...". At the bottom, the MONGOSH shell shows the command 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" } }
```

The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays databases: My Queries, Databases (with Group\_9 selected), and PakWheels. The main area is titled "Group\_9.PakWheels" and shows a search bar with a filter: { "model": "Corolla" }. Below the filter, there are "ADD DATA" and "EXPORT COLLECTION" buttons. The results pane shows a single document: \_id: ObjectId('6455624d184db4c4a8bd0c40'), @type: "Car", brand: Object, model: "Corolla", description: "Alloy Rims. All original documents are complete . Driven on petrol th...", itemCondition: "used", modelDate: 2016, manufacturer: "Toyota", fuelType: "CNG", name: "Toyota Corolla 2016 for sale in Lahore A.K.", image: "https://cache2.pakwheels.com/ad\_pictures/5534/toyota-corolla-gli-vvti-...", vehicleTransmission: "Manual", color: "Black", bodyType: "Sedan". At the bottom, the MONGOSH shell shows the command db.PakWheels.updateMany( { "model": "Corolla" }, { \$set: { "fuelType": "CNG" } } )

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

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

localhost:27017 ... Documents Group\_9.PakWheels

My Queries Databases +

Search

Group\_9 PakWheels ...

Documents Aggregations Schema Explain Plan Indexes Validation

Filter ⚙ { "model": "Civic" } Reset Find More Options ▾

ADD DATA EXPORT COLLECTION

1 - 20 of 5376

... **\_id: ObjectId('6455624d184db4c4a8bd0cd3')** ...

... **features: array**

0: "ABS"  
1: "AM/FM Radio"  
2: "Air Bags"  
3: "Air Conditioning"  
4: "Alloy Rims"  
5: "CD Player"  
6: "Cruise Control"  
7: "Immobilizer Key"  
8: "Keyless Entry"  
9: "Power Locks"  
10: "Power Mirrors"  
11: "Power Steering"  
12: "Power Windows"  
13: "Sun Roof"  
adlastUpdated: "Sep 22, 2021"  
price: 200000  
priceCurrency: "PKR"

>\_MONGOSH

> db.PakWheels.updateMany(  
 { "model": "Civic" },  
 { \$push: { "features": "Sun Roof" } })  
)  
< {  
 acknowledged: true,  
 insertedId: null,  
 matchedCount: 5376,  
 modifiedCount: 5376,  
 upsertedCount: 0  
}

**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" } })
```

localhost:27017 ... Documents Group\_9.PakWheels

My Queries Databases +

Search

Group\_9 PakWheels ...

Documents Aggregations Schema Explain Plan Indexes Validation

Filter ⚙ { "model": "Mehran", "mileageFromOdometer": "80000 km" } Reset Find More Options ▾

ADD DATA EXPORT COLLECTION

1 - 20 of 521

... **\_id: ObjectId('6455624d184db4c4a8bd0cd3')** ...

... **brand: Object**

model: "Mehran"  
description: "Complete original file is available . Totally driven on petrol. Company..."  
itemCondition: "used"  
modelDate: 2005  
manufacturer: "Suzuki"  
fuelType: "Petrol"  
name: "Suzuki Mehran 2005 for sale in Abbottabad"  
image: "https://cache2.pakwheels.com/ad\_pictures/5661/suzuki-mehran-vxr-cng-4-..."  
vehicleTransmission: "Manual"  
color: "#Silver"  
bodyType: "Hatchback"  
vehicleEngine: Object  
mileageFromOdometer: "80000 km"  
sellerLocation: "Ayub Medical Complex, Abbottabad KPK"  
postedFrom: "Added via Phone"

>\_MONGOSH

> db.PakWheels.updateMany(  
 { "model": "Mehran", "mileageFromOdometer": { \$gt: "80000 km" } },  
 { \$set: { "mileageFromOdometer": "80000 km" } })  
)  
< {  
 acknowledged: true,  
 insertedId: null,  
 matchedCount: 521,  
 modifiedCount: 521,  
 upsertedCount: 0  
}

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

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

```
>_MONGOSH  
db.PakWheels.updateMany(  
  { "color": "Silver" },  
  { $rename: { "color": "Moonstone" } })  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 8944,  
  modifiedCount: 8944,  
  upsertedCount: 0  
}
```

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

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

```
>_MONGOSH  
db.PakWheels.updateMany(  
  { "features": "Immobilizer Key" },  
  { $pull: { "features": "Immobilizer Key" } })  
< {  
  acknowledged: true,  
  insertedId: null,  
  matchedCount: 28989,  
  modifiedCount: 28989,  
  upsertedCount: 0  
}
```

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

Object Id: 645bf4df683b07244ebbee933

The screenshot shows the MongoDB Compass interface. On the left, the sidebar lists databases: My Queries, Databases (Group\_9 selected), and local. Under Group\_9, the collection 'PakWheels' is selected. The main area displays the 'Documents' tab for the 'Group\_9.PakWheels' collection. A search bar contains the query: { "\_id" : ObjectId('645bf4df683b07244ebbee933') }. Below the search bar are buttons for 'Reset', 'Find', and 'More Options'. A table shows one document with the following details:

```

_id: ObjectId("645bf4df683b07244ebbee933")
manufacturer: "Toyota"
model: "Corolla Altis"
vehicleTransmission: "Automatic"
mileageFromOdometer: "55000 km"
price: 2350000
extraFeatures: Object
fuelType: "Petrol"
features: Array

```

Below the table, the MONGOSH shell shows the command used to insert the document:

```

> db.PakWheels.insertOne({
  "manufacturer": "Toyota",
  "model": "Corolla Altis",
  "vehicleTransmission": "Automatic",
  "mileageFromOdometer": "55000 km",
  "price": 2350000,
  "extraFeatures": { "EngineCapacity": "1600 cc", "Assembly": "Local" },
  "fuelType": "Petrol",
  "features": [ "Air Conditioning", "Power Windows", "Power Steering" ]
})
{
  acknowledged: true,
  insertedId: ObjectId("645bf4df683b07244ebbee933")
}

```

ObjectId: 645bf73b683b07244ebbee934

The screenshot shows the MongoDB Compass interface. The sidebar and collection selection are identical to the previous screenshot. The main area displays the 'Documents' tab for the 'Group\_9.PakWheels' collection. A search bar contains the query: { "\_id" : ObjectId('645bf73b683b07244ebbee934') }. Below the search bar are buttons for 'Reset', 'Find', and 'More Options'. A table shows one document with the following details:

```

_id: ObjectId("645bf73b683b07244ebbee934")
manufacturer: "Suzuki"
model: "Cultus VXRI"
modelYear: 2021
itemCondition: "new"
vehicleTransmission: "Manual"
price: 1575000
extraFeatures: Object

```

Below the table, the MONGOSH shell shows the command used to insert the document:

```

> db.PakWheels.insertOne({
  "manufacturer": "Suzuki",
  "model": "Cultus VXRI",
  "modelYear": 2021,
  "itemCondition": "new",
  "vehicleTransmission": "Manual",
  "price": 1575000,
  "extraFeatures": { "EngineCapacity": "1000 cc", "Assembly": "Local" },
  "color": "White",
  "fuelType": "Petrol",
  "features": [ "AM/FM Radio", "Air Conditioning", "Power Windows", "Power Steering" ]
})
{
  acknowledged: true,
  insertedId: ObjectId("645bf73b683b07244ebbee934")
}

```

Object Id: 645bf7ec683b07244ebbe935

The screenshot shows the MongoDB Compass interface. The top navigation bar indicates the connection is to 'localhost:27017' and the database is 'Group\_9.PakWheels'. The collection selected is 'PakWheels'. The document count is 55.7k documents and 1 index. A search bar is present, and the left sidebar shows the database structure with 'admin', 'config', and 'local' databases, and 'PakWheels' collection selected.

**Documents** tab selected. Filter: `{ "_id" : ObjectId('645bf7ec683b07244ebbe935') }`. ADD DATA button is visible. Document details:

```
_id: ObjectId('645bf7ec683b07244ebbe935')
manufacturer: "Honda"
model: "City Aspire"
modelYear: 2016
itemCondition: "used"
vehicleTransmission: "Automatic"
mileageFromOdometer: "82000 km"
price: 2050000,
```

MONGOSH shell output:

```
> db.PakWheels.insertOne({
  "manufacturer": "Honda",
  "model": "City Aspire",
  "modelYear": 2016,
  "itemCondition": "used",
  "vehicleTransmission": "Automatic",
  "mileageFromOdometer": "82000 km",
  "price": 2050000,
  "extraFeatures": { "EngineCapacity": "1500 cc", "Assembly": "Local" },
  "color": "Black",
  "fuelType": "Petrol",
  "features": [ "AM/FM Radio", "Air Conditioning", "Power Windows", "Power Steering" ]
})
< 
  acknowledged: true,
  insertedId: ObjectId("645bf7ec683b07244ebbe935")
```

Object Id:

645bf928683b07244ebbe936

The screenshot shows the MongoDB Compass interface, identical to the previous one but with a different object ID. The top navigation bar indicates the connection is to 'localhost:27017' and the database is 'Group\_9.PakWheels'. The collection selected is 'PakWheels'. The document count is 55.7k documents and 1 index. A search bar is present, and the left sidebar shows the database structure with 'admin', 'config', and 'local' databases, and 'PakWheels' collection selected.

**Documents** tab selected. Filter: `{ "_id" : ObjectId('645bf928683b07244ebbe936') }`. ADD DATA button is visible. Document details:

```
_id: ObjectId('645bf928683b07244ebbe936')
manufacturer: "Toyota"
model: "Hilux Revo"
modelYear: 2018
itemCondition: "used"
vehicleTransmission: "Automatic"
mileageFromOdometer: "75000 km"
price: 6800000,
```

MONGOSH shell output:

```
> db.PakWheels.insertOne({
  "manufacturer": "Toyota",
  "model": "Hilux Revo",
  "modelYear": 2018,
  "itemCondition": "used",
  "vehicleTransmission": "Automatic",
  "mileageFromOdometer": "75000 km",
  "price": 6800000,
  "extraFeatures": { "EngineCapacity": "3000 cc", "Assembly": "Imported" },
  "color": "Grey",
  "fuelType": "Diesel",
  "features": [ "AM/FM Radio", "Air Conditioning", "Power Windows", ]
```

```
)
< 
  acknowledged: true,
  insertedId: ObjectId("645bf928683b07244ebbe936")
```

Object Id: 645bfa5d683b07244ebbee937

The screenshot shows the MongoDB Compass interface. On the left, the sidebar lists databases: Group\_9, admin, config, local, and PakWheels (selected). The main area displays the 'Group\_9.PakWheels' collection with 55.7k documents and 1 index. A document is selected with the following fields:

```
_id: ObjectId('645bfa5d683b07244ebbee937')
model: "Honda Civic"
modelDate: 2022
price: 4500000
itemCondition: "new"
vehicleTransmission: "Automatic"
fuelType: "Petrol"
color: "White"
mileageFromOdometer: "0 km",
extraFeatures: {"EngineCapacity": "1500 cc", "Assembly": "Local", "BodyType": "Sedan"} ,
features: [ "AM/FM Radio", "Air Conditioning", "CD Player", "DVD Player", "Immobilizer Key", "Keyless Entry", "Navigation System", "Power Locks", "Power Mirrors", "Power Steering" ]
```

In the mongo shell at the bottom, a document is inserted with the same fields and values.

**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:

The screenshot shows the MongoDB Compass interface. The 'PakWheels' collection has 55.7k documents and 1 index. A delete query is run:

```
Filter { "price": { $gt: 10000000 } }
```

The mongo shell shows the result of the delete operation:

```
1 - 20 of 1105
{
  acknowledged: true,
  deletedCount: 1105
}
```

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

**Documents** Group\_9.PakWheels

Filter: { "itemCondition": "new" }

```
_id: ObjectId('645bf73b683b07244ebbe934')
manufacturer: "Suzuki"
model: "Vitara VXRI"
modelYear: 2021
itemCondition: "new"
vehicleTransmission: "Manual"
price: 1575000
extraFeatures: Object
color: "White"
fuelType: "Petrol"
features: Array

_id: ObjectId('645bfa5d683b07244ebbe937')
model: "Honda Civic"
modelDate: 2022
price: 4500000
itemCondition: "new"
vehicleTransmission: "Automatic"
fuelType: "Petrol"
color: "White"
mileageFromOdometer: "0 km"
extraFeatures: Object
```

MONGOSH

```
> db.PakWheels.deleteMany({ "itemCondition": "new" })
< 
  acknowledged: true,
  deletedCount: 2
>
```

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

**Documents** Group\_9.PakWheels

Filter: { "\_id": ObjectId('6455624d184db4c4a8bd0c37') }

```
_id: ObjectId('6455624d184db4c4a8bd0c37')
@type: "Car"
brand: Object
model: "Pajero"
description: "Read Full Description First
  Its Super Select ( 4x4 ) Edition
  92 Model"
itemCondition: "used"
modelDate: 1992
manufacturer: "Mitsubishi"
fuelType: "Petrol"
name: "Mitsubishi Pajero 1992 for sale in Islamabad"
image: "https://cache2.pakwheels.com/ad_pictures/5543/mitsubishi-pajero-exceed_"
vehicleTransmission: "Automatic"
bodyType: "SUV"
vehicleEngine: Object
mileageFromOdometer: "82,400 km"
sellerLocation: "Airport Enclave, Islamabad Islamabad"
postedFrom: "Added via Phone"
keywords: "mitsubishi pajero 1992 for sale, mitsubishi pajero 1992, mitsubishi pa_"
extraFeatures: Object
```

MONGOSH

```
> db.PakWheels.deleteOne({ "_id": ObjectId('6455624d184db4c4a8bd0c37') })
< 
  acknowledged: true,
  deletedCount: 1
>
```

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

The screenshot shows the MongoDB Compass interface. On the left, the sidebar lists databases: My Queries, Databases, Group\_9, and local. Under Group\_9, the 'PakWheels' collection is selected. The main area displays the 'Documents' tab for the 'PakWheels' collection. A filter bar at the top shows the query: { "manufacturer": "Honda", "model": "Civic" }. Below the filter, a preview pane shows a single document:

```

{
  "_id": ObjectId('6455624d184db4c4a8bd0c47'),
  "type": "Car",
  "brand": Object,
  "model": "Civic",
  "description": "Full modified car best in town\n  Multimedia power steering\n  Power Window..",
  "itemCondition": "used",
  "modelDate": 2000,
  "manufacturer": "Honda",
  "fuelType": "Petrol",
  "name": "Honda Civic 2000 for sale in Rawalpindi",
  "image": "https://cache2.pakwheels.com/ad_pictures/5534/honda-civic-vti-oriel-au..",
  "vehicleTransmission": "Automatic",
  "color": "White",
  "bodyType": "Sedan",
  "vehicleEngine": Object,
  "mileageFromOdometer": "100,000 km",
  "sellerLocation": "Rawalpindi Punjab",
  "postedFrom": "Added via Website",
  "keywords": "honda civic 2000 for sale, honda civic 2000, honda civic 2000 rawalp..",
  "extraFeatures": Object,
  "features": Array
}

```

At the bottom of the interface, the MONGOSH command line shows the execution of the delete operation:

```

> db.PakWheels.deleteMany({ "manufacturer": "Honda", "model": "Civic" })
< 
  acknowledged: true,
  deletedCount: 5376
>

```

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

The screenshot shows the MongoDB Compass interface. The sidebar and collection selection are identical to the previous query. The main area displays the 'Documents' tab for the 'PakWheels' collection. A filter bar at the top shows the query: { "modelDate": { \$lte: 2010 }, "vehicleTransmission": "Automatic" }. Below the filter, a preview pane shows a single document:

```

{
  "_id": ObjectId('6455624d184db4c4a8bd0c39'),
  "type": "Car",
  "brand": Object,
  "model": "Corolla",
  "description": "Axio luxel alfa 2007 model full option sensor bumpers with original bu..",
  "itemCondition": "used",
  "modelDate": 2007,
  "manufacturer": "Toyota",
  "fuelType": "CNG",
  "name": "Toyota Corolla 2007 for sale in Peshawar",
  "image": "https://cache1.pakwheels.com/ad_pictures/5533/toyota-corolla-2007-5533..",
  "vehicleTransmission": "Automatic",
  "color": "White",
  "bodyType": "Object",
  "mileageFromOdometer": "71,000 km",
  "sellerLocation": "Ring Road, Peshawar KPK",
  "postedFrom": "Added via Phone",
  "keywords": "toyota corolla 2007 for sale, toyota corolla 2007, toyota corolla 2007..",
  "extraFeatures": Object,
  "features": Array,
  "adLastUpdated": "Aug 27, 2021",
  "price": 3500000,
  "priceCurrency": "PKR"
}

```

At the bottom of the interface, the MONGOSH command line shows the execution of the delete operation:

```

> db.PakWheels.deleteMany({ "modelDate": { $lte: 2010 }, "vehicleTransmission": "Automatic" })
< 
  acknowledged: true,
  deletedCount: 5781
>

```