**CSCI 5408 Data Management Warehousing and Analytics**

**Assignment 1**

**Search Query Implementation using Relational Database and Elastic Search**

**Date of Submission: May 24, 2018**

**Hemanth Kurra (B00784050)**

**Srisaichand Singamaneni (B00792835)**

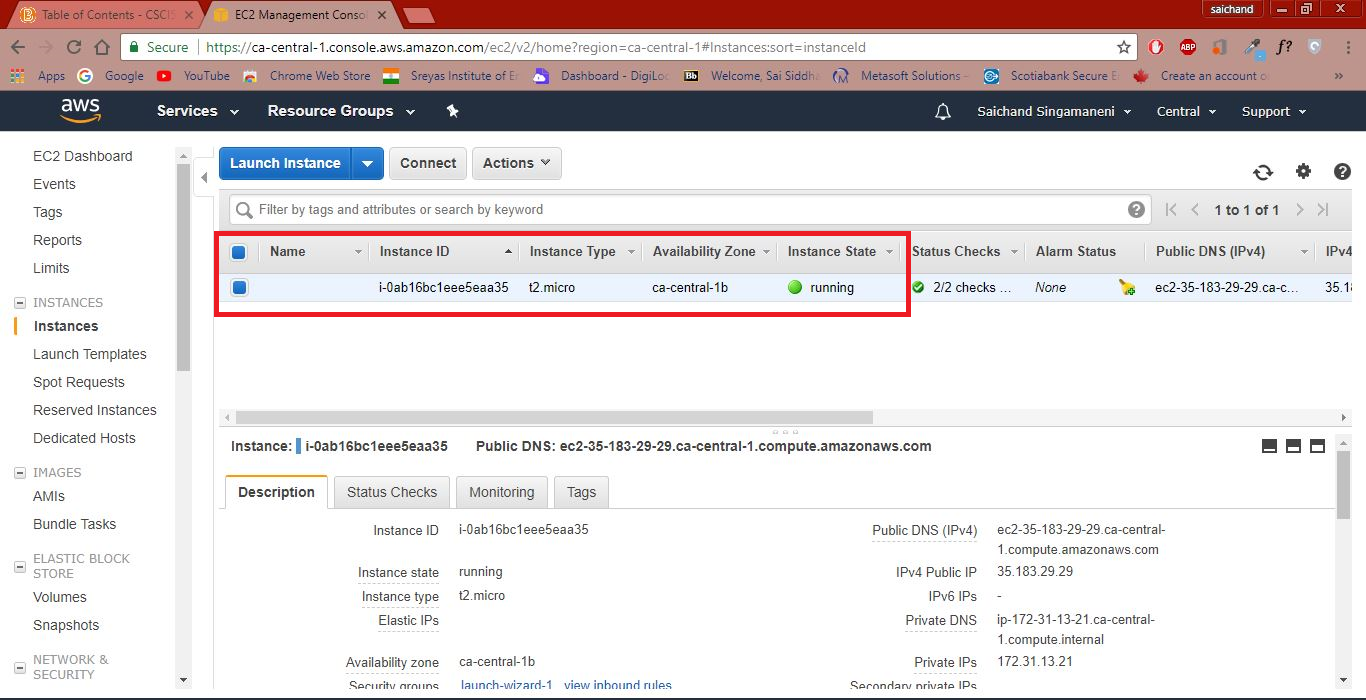
**GitHub URL:** <https://github.com/singamanenisrisai/MySQL-and-Elastic-Search.git>

**1. TASK DESCRIPTION**:

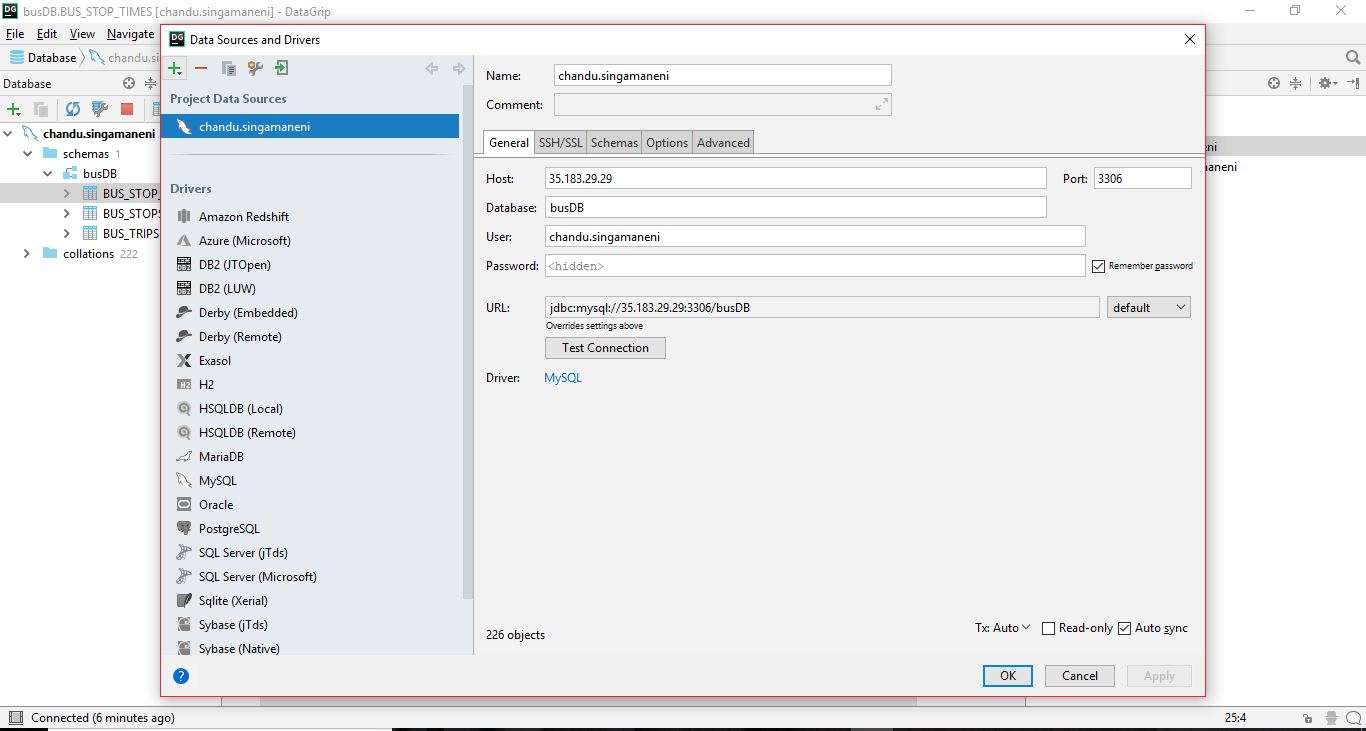
The purpose of the task is to analyse the most efficient data retrieval tool by comparing relational database system and cloud database. The Halifax transit data was used for the analysis.

**Applications & Requirements**

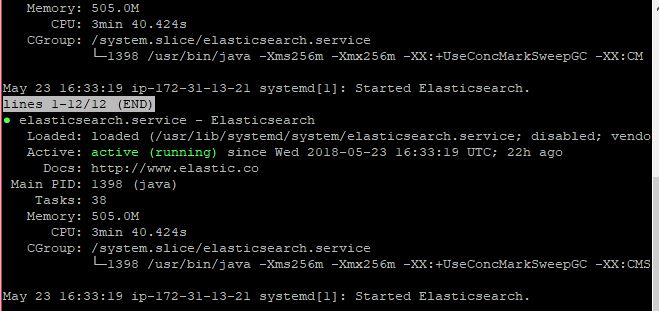
* Amazon AWS cloud service
* Virtual machine



* DataGrip 2018.1

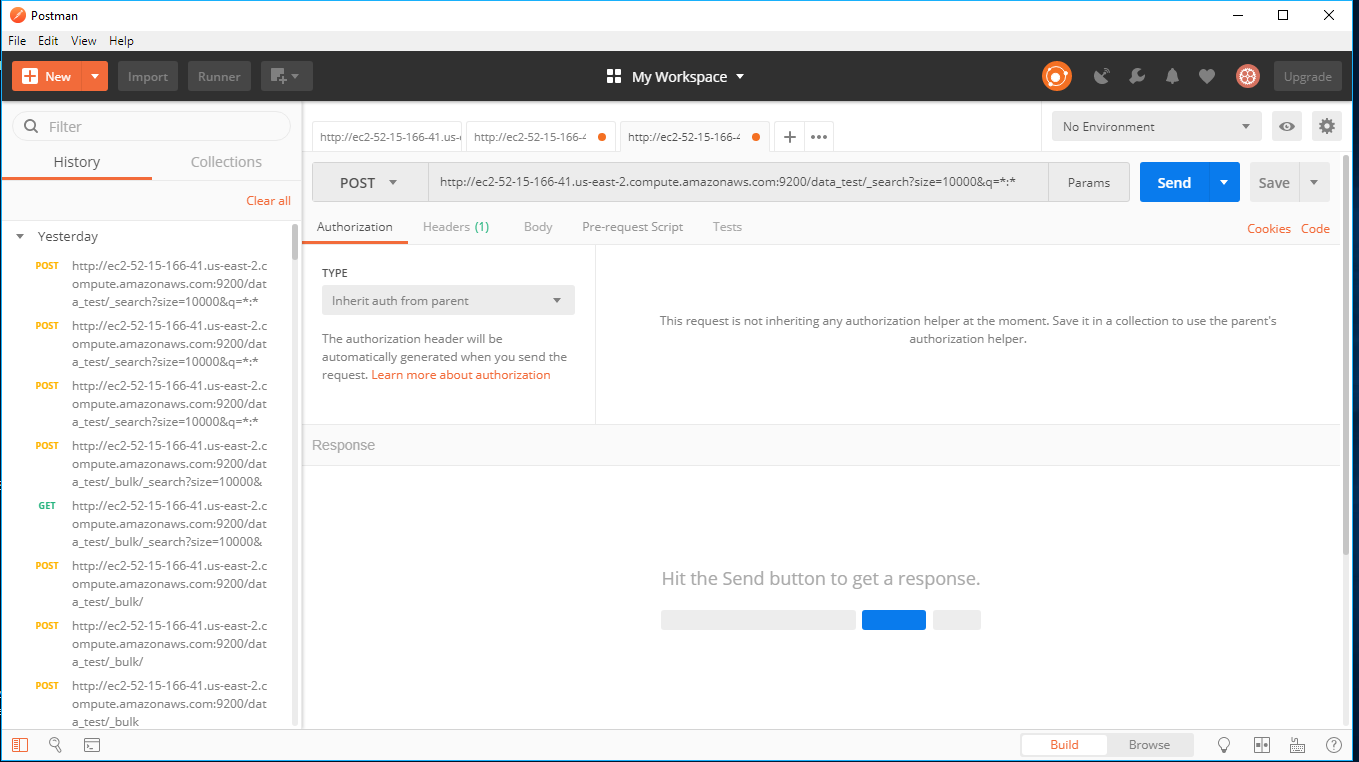


* Elastic Search



* PostMan

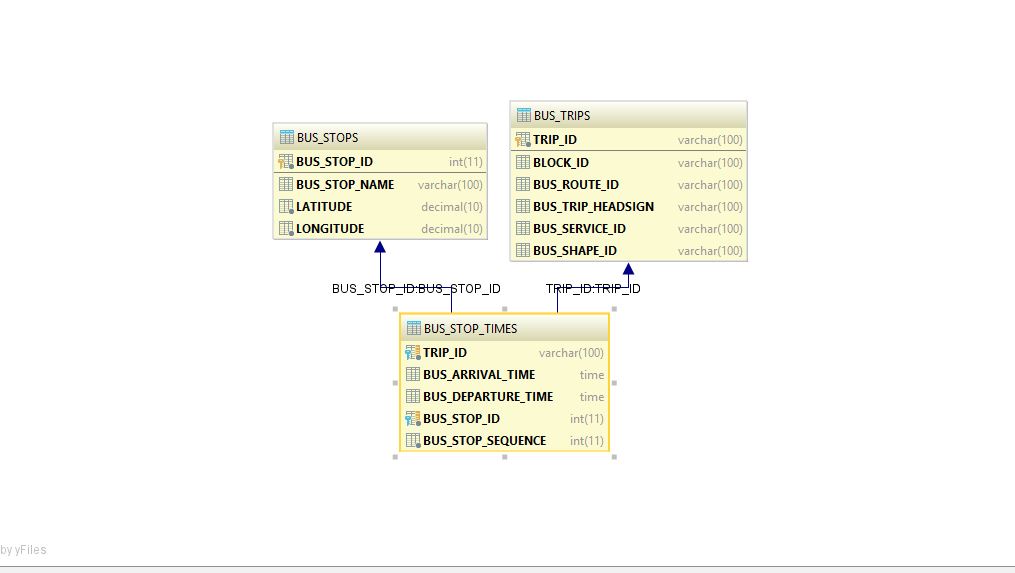
The Postman API development application used to interact with elastic search in the AWS.



We also generated RSA keys by using PuTTYgen on Windows for secure SSH authentication with OpenSSH. This is used for secure SSH access to the cloud server by using public-private key pair.

**2. RELATIONAL DATABASE** **DESIGN**:

Data Grip 2018 is used for relational data base. As we encounter some issues while importing large csv file in MySQL workbench, we found that Data Grip provides a dedicated UI for importing csv and tsv files and better performance when compared to workbench.



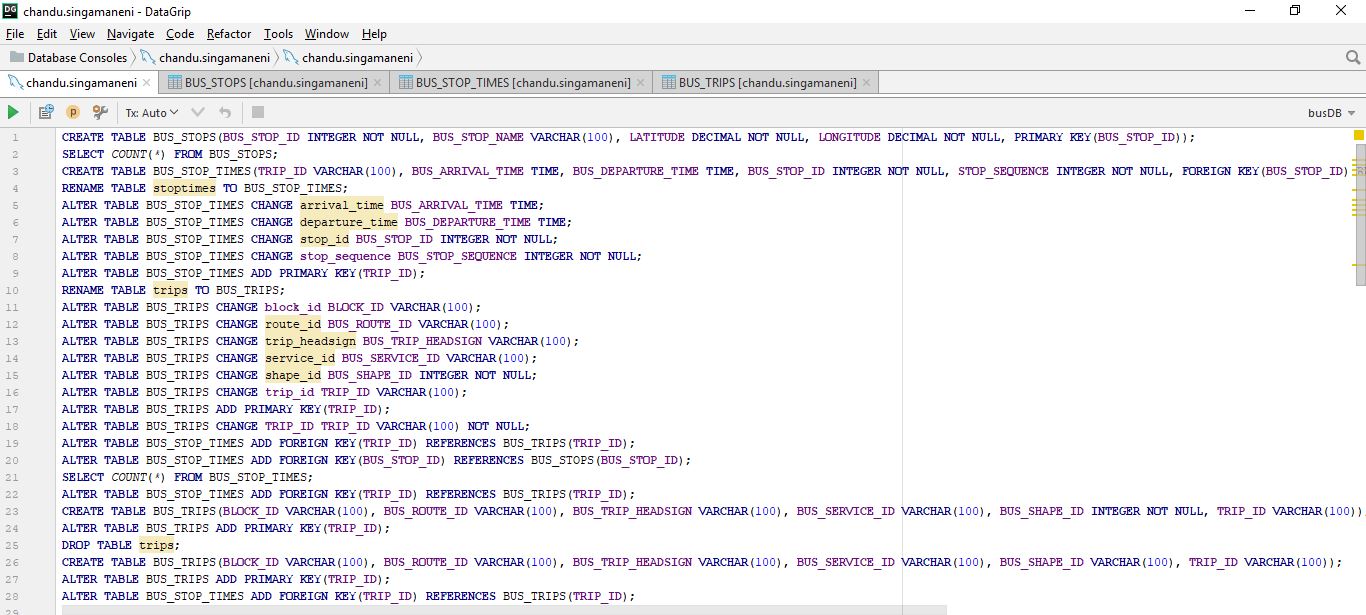
**Screenshot: ER Diagram of the relational databases**

**Table1:** BUS\_STOPS (Primary Key: BUS\_STOP\_ID)

**Table2:** BUS\_TRIPS (Primary Key: TRIP\_ID)

**Table3:** BUS\_STOP\_TIMES (Foreign Keys: BUS\_STOP\_ID, TRIP\_ID)

**Data Formatting and setup**



**3. APPLICATION QUERIES:**

The comparison is made between the RDBMS and Elastic search in Amazon services (Amazon AWS).

1. ***Find all buses for a particular Bus Stop***

***1.Input: Bus Stop Name***

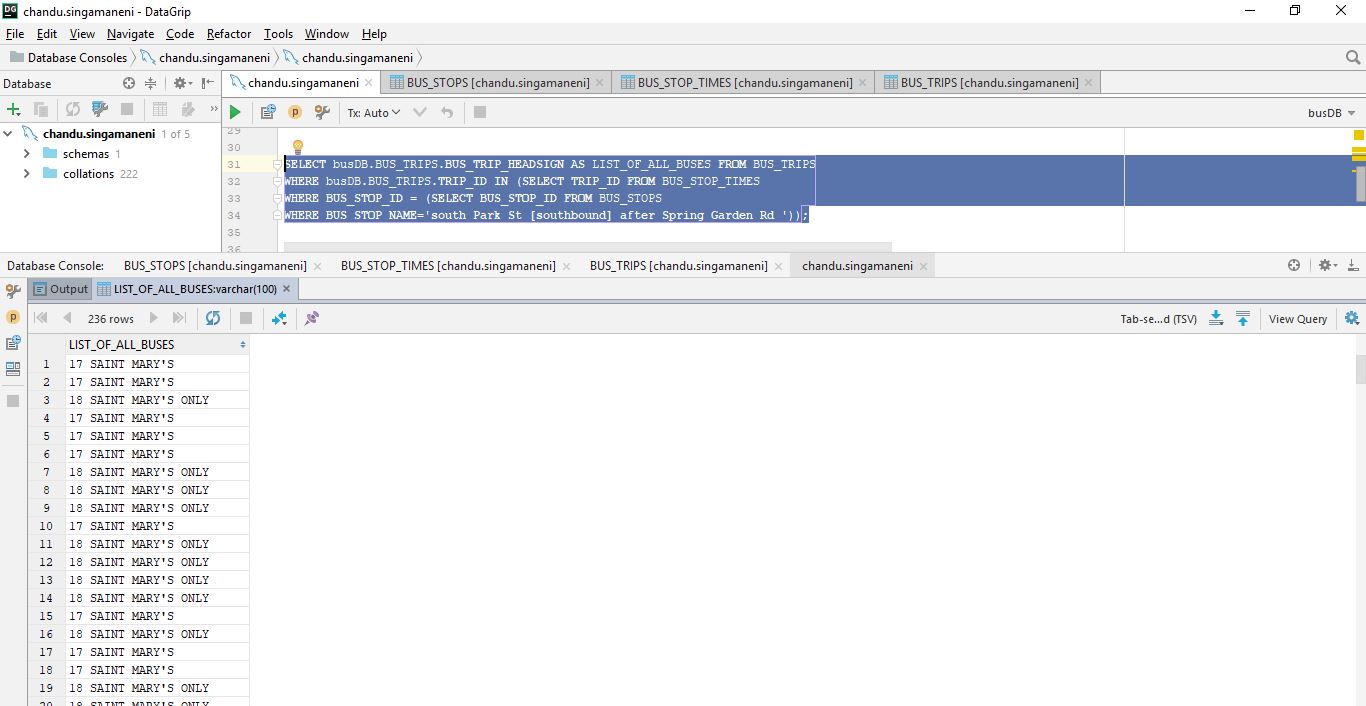
***2.Output: List of all buses, response time for the search query***

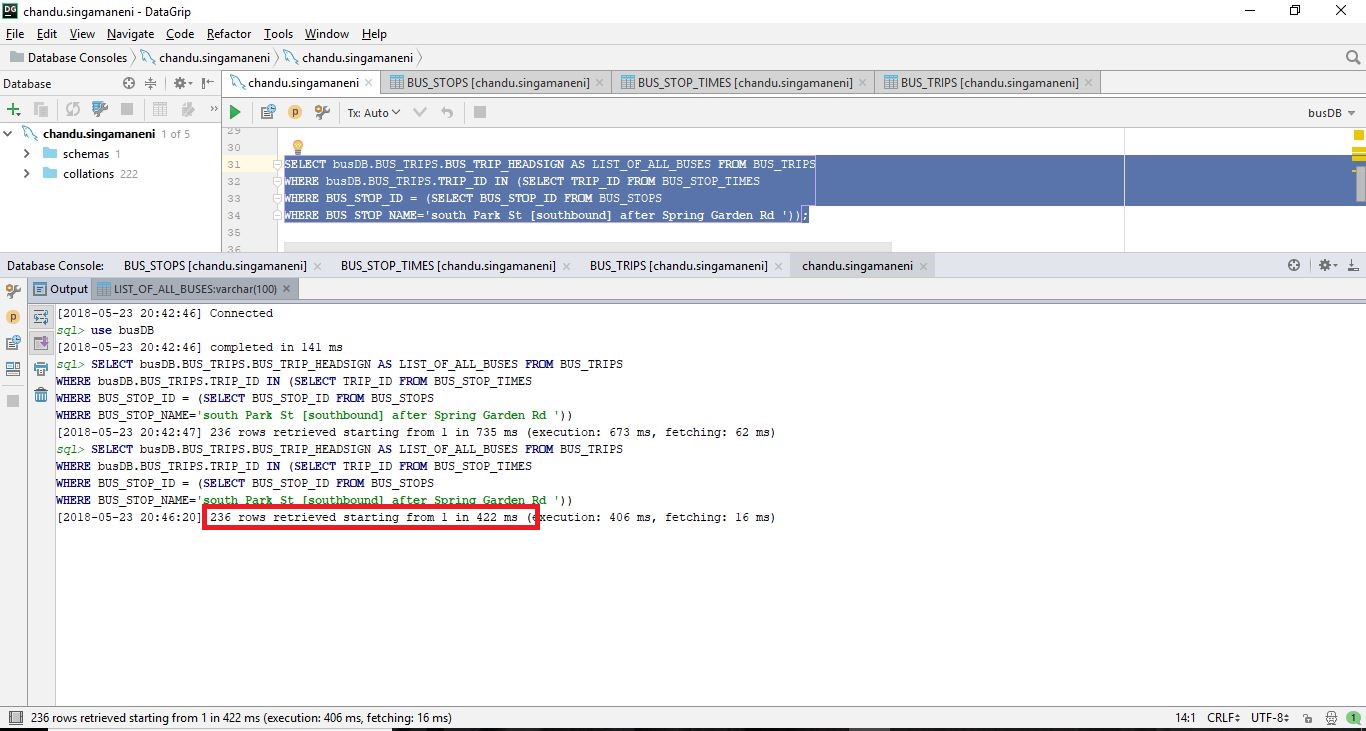
**SQL Query**

SELECT busDB.BUS\_TRIPS.BUS\_TRIP\_HEADSIGN AS LIST\_OF\_ALL\_BUSES FROM BUS\_TRIPS WHERE busDB.BUS\_TRIPS.TRIP\_ID IN (SELECT TRIP\_ID FROM BUS\_STOP\_TIMES WHERE BUS\_STOP\_ID = (SELECT BUS\_STOP\_ID FROM BUS\_STOPS

WHERE BUS\_STOP\_NAME='south Park St [southbound] after Spring Garden Rd '));

**SQL Output**





**Response Time: 422ms**

**Elastic Search Query**

**Sub Query 1:**

URL: http://ec2-35-183-29-29.ca-central-1.compute.amazonaws.com:9200/bus\_stops/\_search

{

"\_source" : ["name\_stop", "stop\_id"],

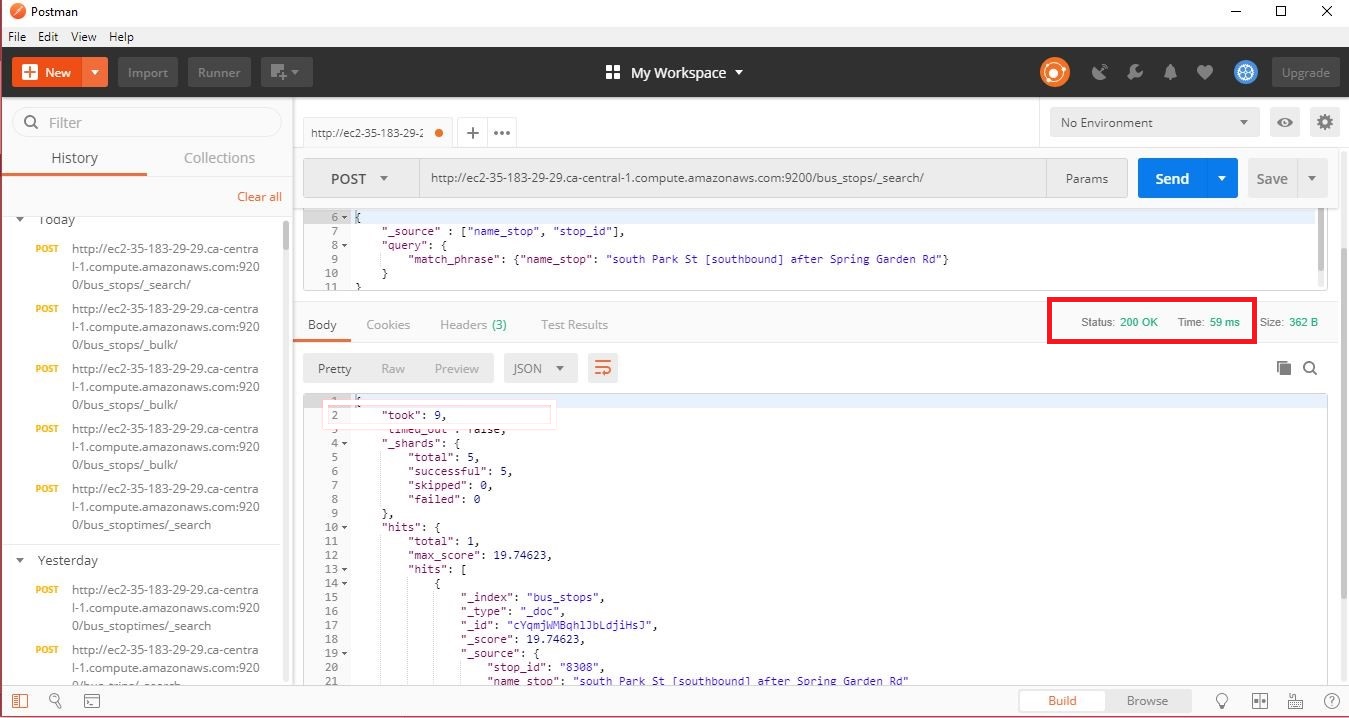
"query": {

"match\_phrase": {"name\_stop": "south Park St [southbound] after Spring Garden Rd"}

}

}

**Output**



**Response Time: 59ms**

**Sub Query 2:**

URL: http://ec2-35-183-29-29.ca-central-1.compute.amazonaws.com:9200/bus\_stoptimes/\_search

{

"\_source" : ["trip\_id", "stop\_id"],

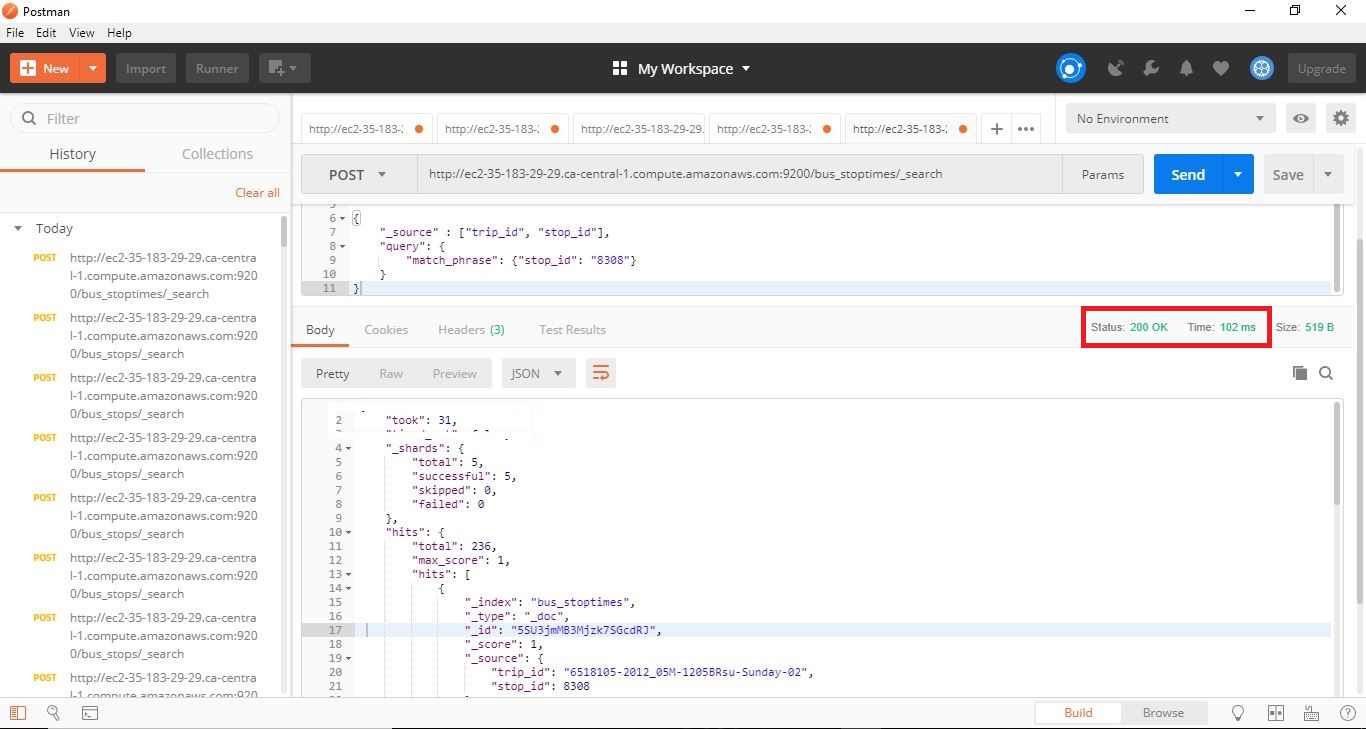
"query": {

"match\_phrase": {"stop\_id": "8308"}

}

}

**Output**



**Response Time: 102ms**

**Sub Query 3:**

URL: http://ec2-35-183-29-29.ca-central-1.compute.amazonaws.com:9200/bus\_trips/\_search

{

"\_source" : ["trip\_id", "trip\_headsign"],

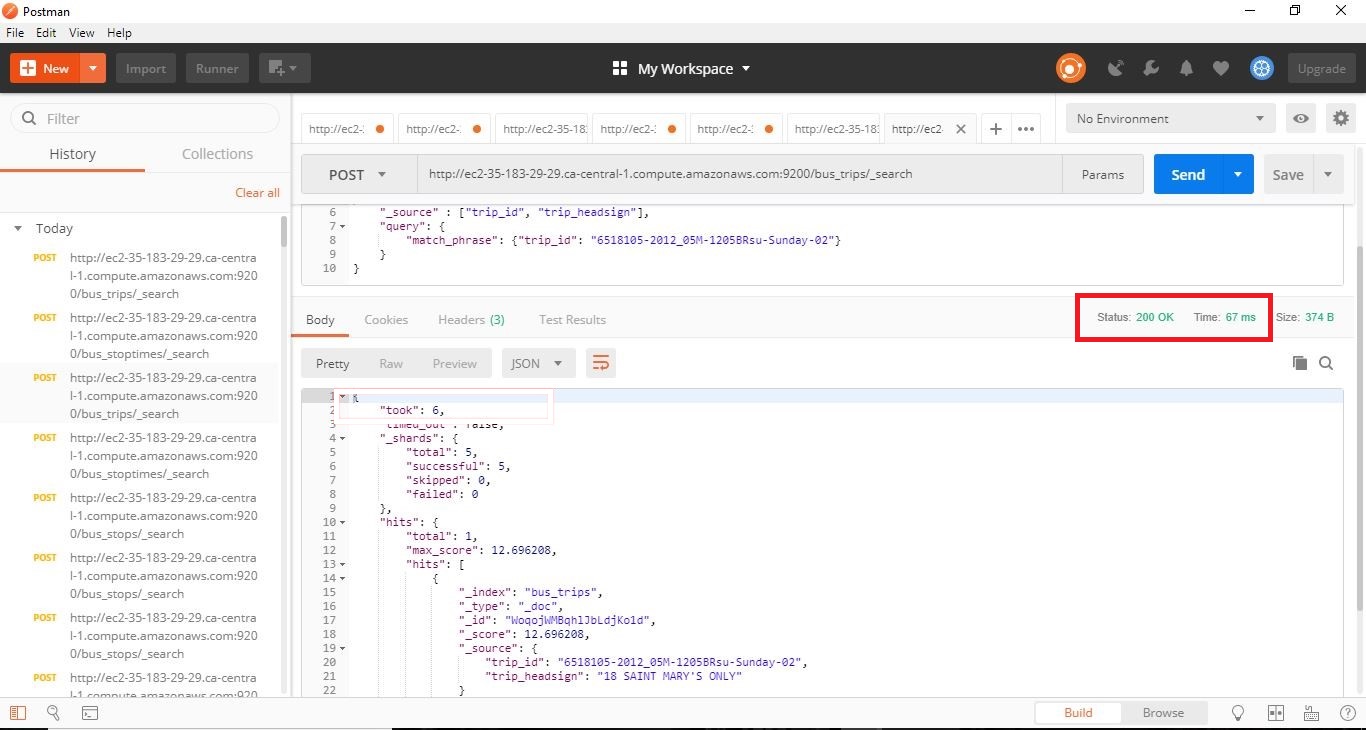
"query": {

"match\_phrase": {"trip\_id": "6518105-2012\_05M-1205BRsu-Sunday-02"}

}

}

**Output**



**Response Time: 67ms**

When compared both SQL and Elastic search response times (422ms, 228ms), the elastic search is more time efficient.

1. ***Find buses between two-time ranges***

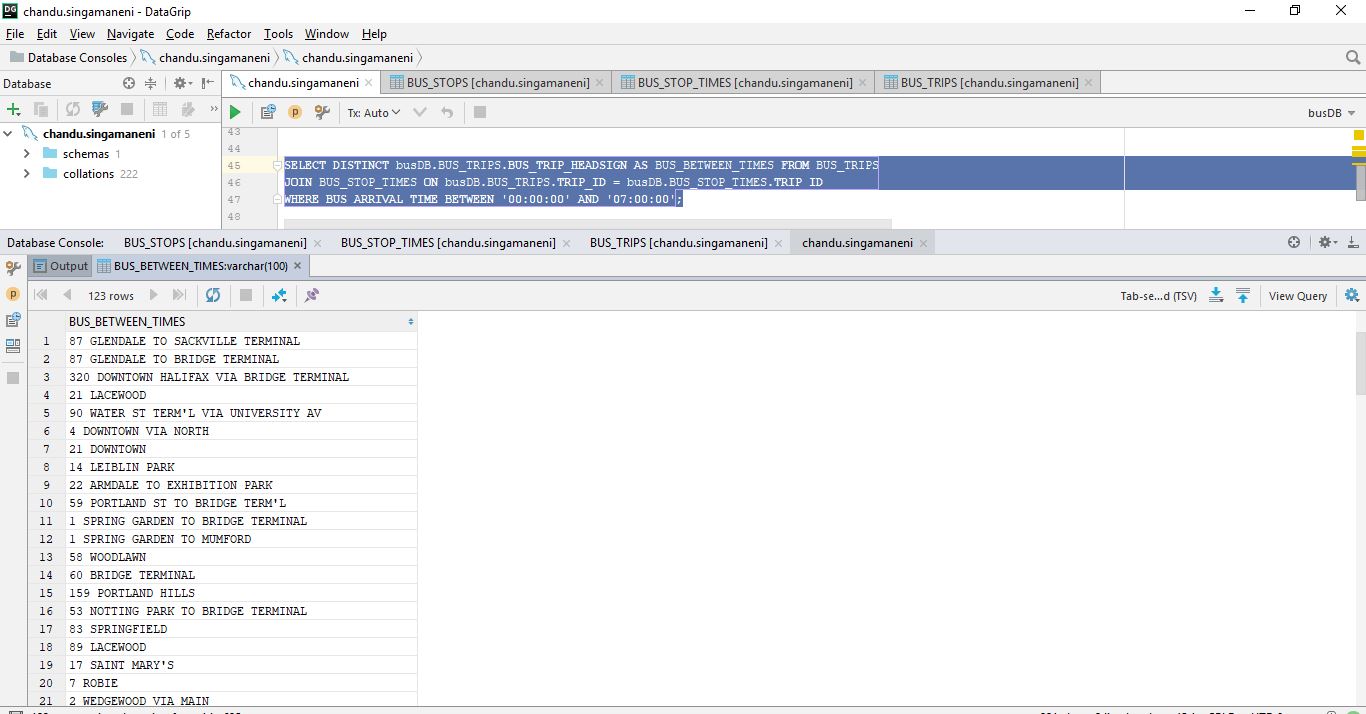
***1.Input: Time Range 1 (hh:mm:ss), Time Range 2 (hh:mm:ss)***

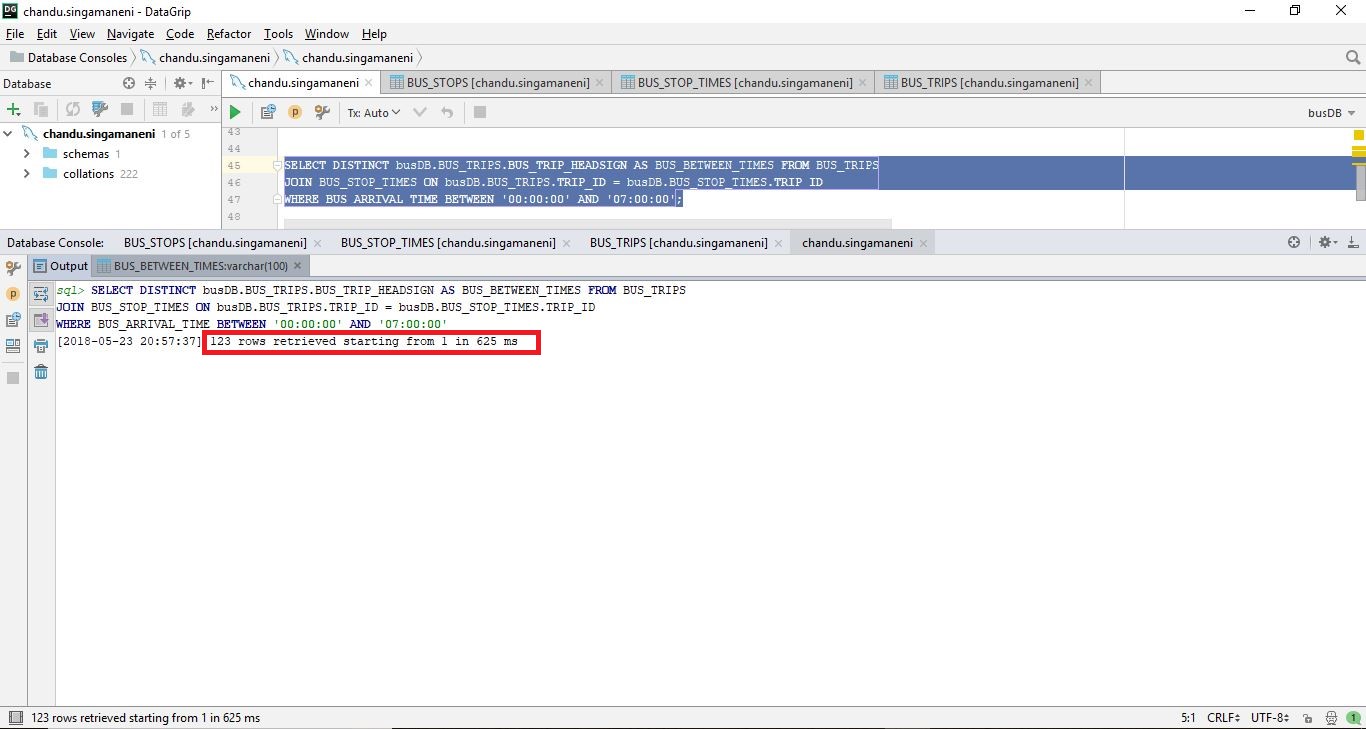
***2.Output: List of all buses, response time for the search query***

**SQL Query**

SELECT DISTINCT busDB.BUS\_TRIPS.BUS\_TRIP\_HEADSIGN AS BUS\_BETWEEN\_TIMES FROM BUS\_TRIPS JOIN BUS\_STOP\_TIMES ON busDB.BUS\_TRIPS.TRIP\_ID = busDB.BUS\_STOP\_TIMES.TRIP\_ID WHERE BUS\_ARRIVAL\_TIME BETWEEN '00:00:00' AND '07:00:00';

**SQL Output**





**Response Time: 625ms**

**Elastic Search Query**

**Sub Query 1:**

URL: http://ec2-35-183-29-29.ca-central-1.compute.amazonaws.com:9200/bus\_stoptimes/\_search

{

"\_source" : ["trip\_id", "arrival\_time"],

"query": {

"range": {"arrival\_time": {

"gte": "13:00:00",

"lte": "20:57:00"

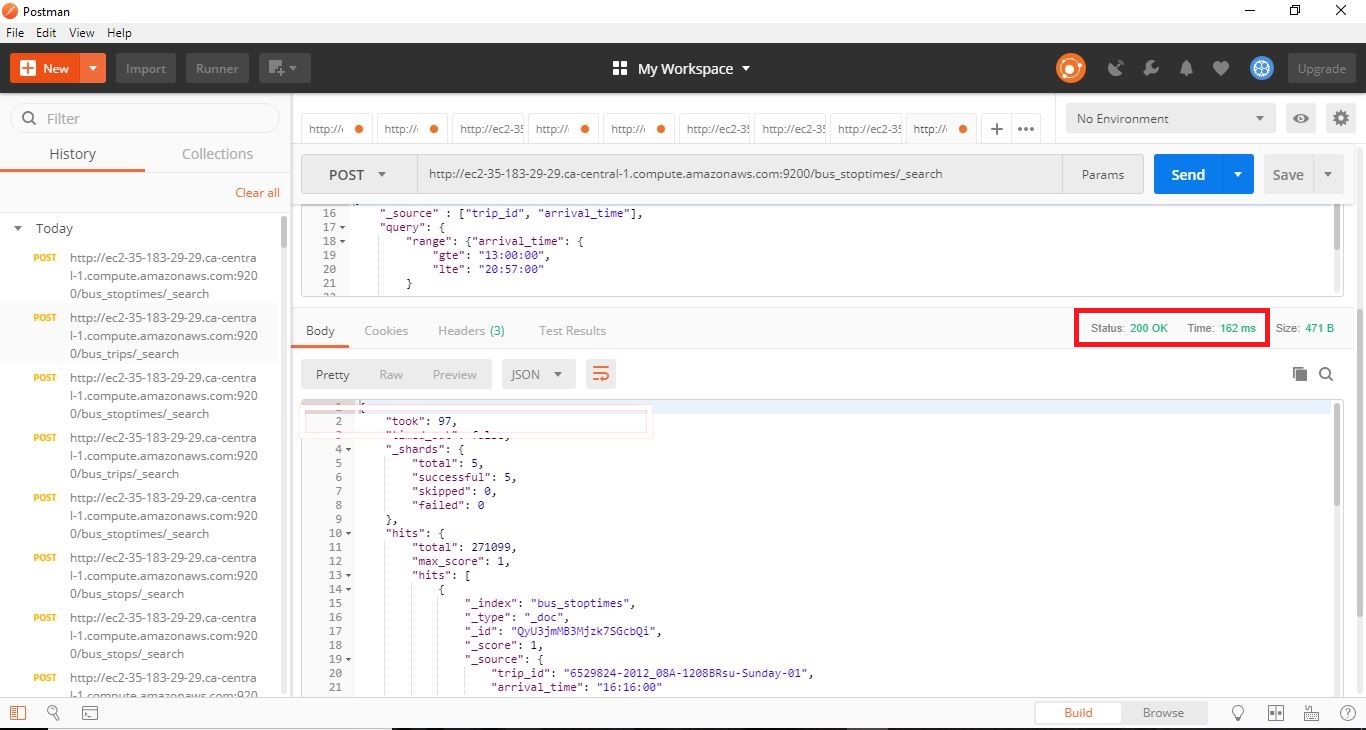
}

}

}

}

**Output**



**Response Time: 162ms**

**Sub Query 2:**

URL: http://ec2-35-183-29-29.ca-central-1.compute.amazonaws.com:9200/bus\_trips/\_search

{ "\_source" : ["trip\_id", "trip\_headsign"],

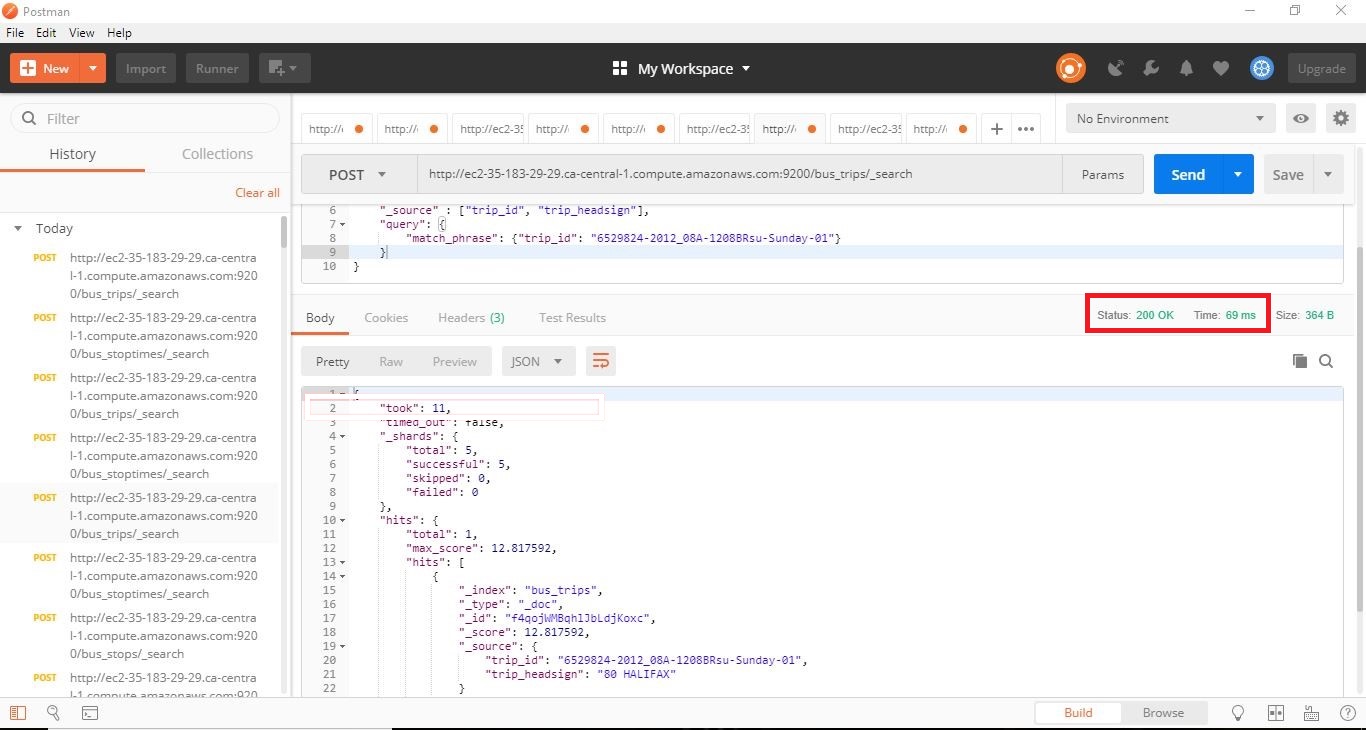
"query": {

"match\_phrase": {"trip\_id": "6529824-2012\_08A-1208BRsu-Sunday-01"}

}

}

**Output**



**Response Time: 69ms**

When compared both SQL and Elastic search response times (625ms, 231ms), the elastic search is more time efficient.

1. ***Find route information of a particular bus on a particular route***

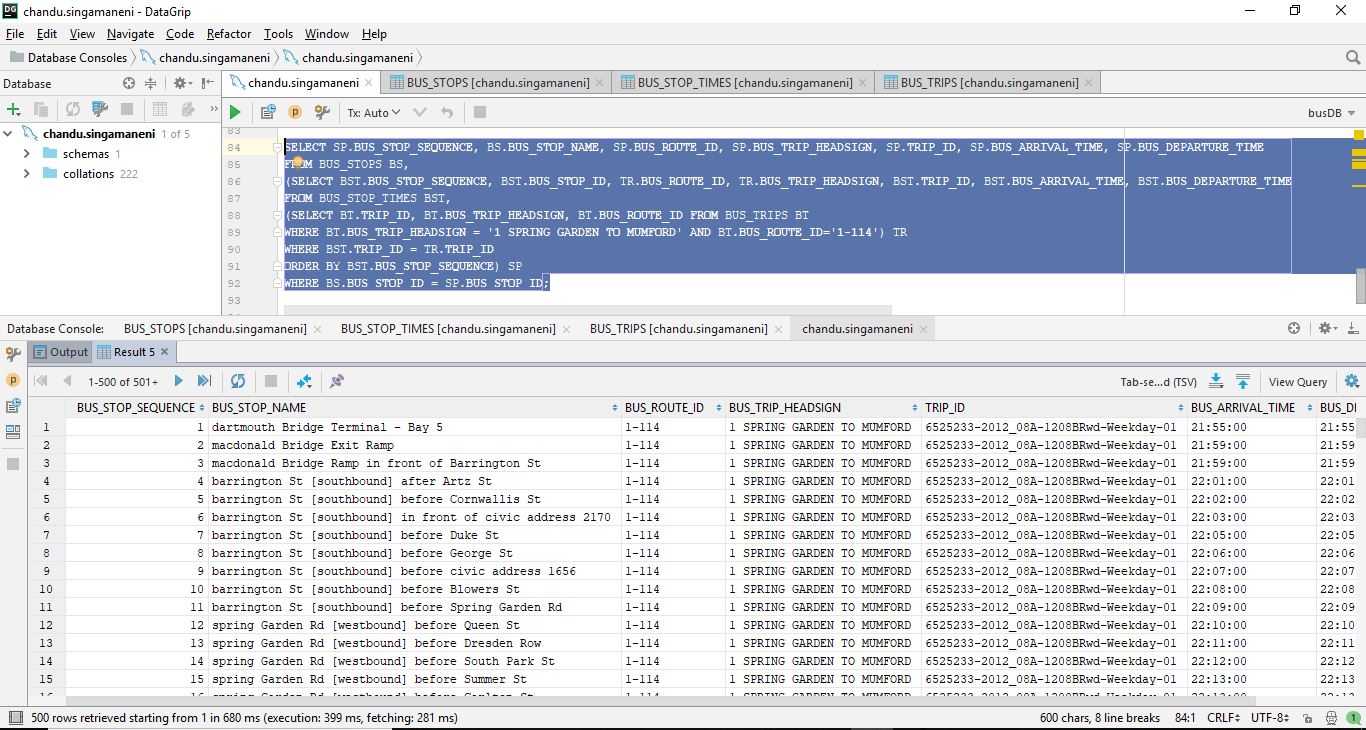
***1.Input: Bus Name, Route Name***

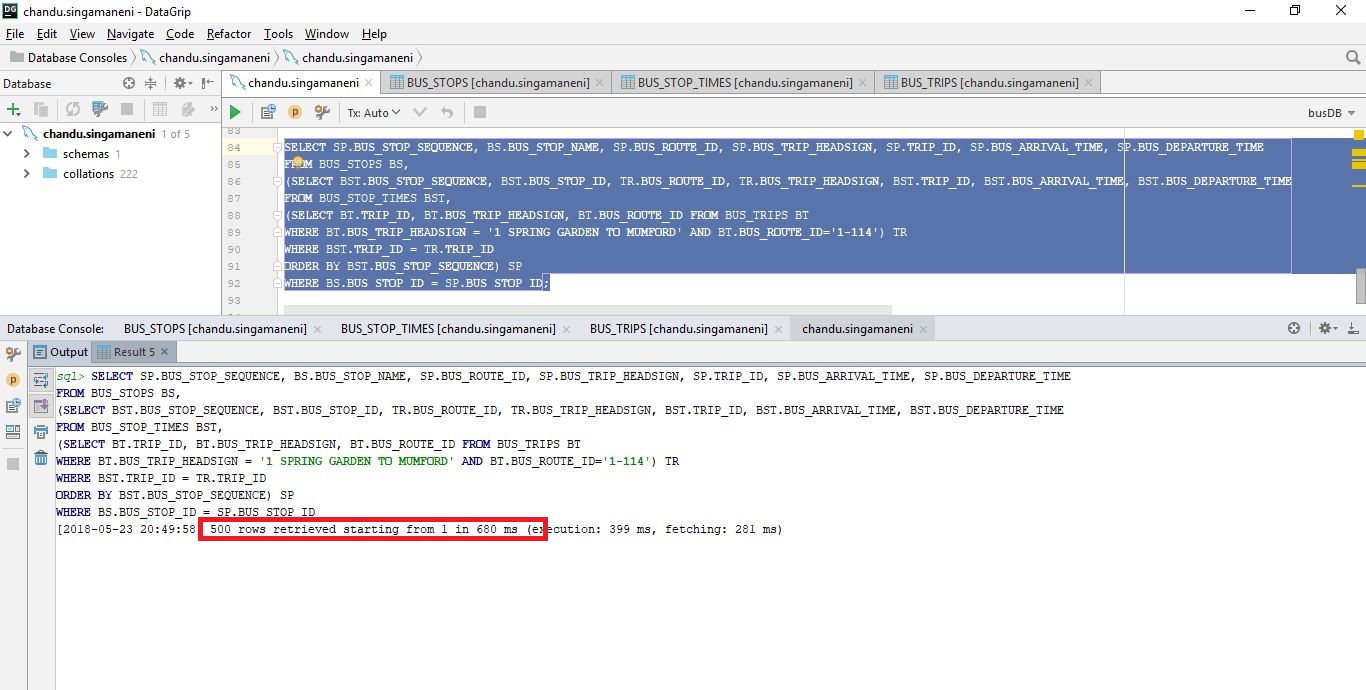
***2.Output: List of all routes, response time for the search query***

**SQL Query**

SELECT SP.BUS\_STOP\_SEQUENCE, BS.BUS\_STOP\_NAME, SP.BUS\_ROUTE\_ID, SP.BUS\_TRIP\_HEADSIGN, SP.TRIP\_ID, SP.BUS\_ARRIVAL\_TIME, SP.BUS\_DEPARTURE\_TIME FROM BUS\_STOPS BS,(SELECT BST.BUS\_STOP\_SEQUENCE, BST.BUS\_STOP\_ID, TR.BUS\_ROUTE\_ID, TR.BUS\_TRIP\_HEADSIGN, BST.TRIP\_ID, BST.BUS\_ARRIVAL\_TIME, BST.BUS\_DEPARTURE\_TIME FROM BUS\_STOP\_TIMES BST,(SELECT BT.TRIP\_ID, BT.BUS\_TRIP\_HEADSIGN, BT.BUS\_ROUTE\_ID FROM BUS\_TRIPS BT WHERE BT.BUS\_TRIP\_HEADSIGN = '1 SPRING GARDEN TO MUMFORD' AND BT.BUS\_ROUTE\_ID='1-114') TR WHERE BST.TRIP\_ID = TR.TRIP\_ID ORDER BY BST.BUS\_STOP\_SEQUENCE) SP WHERE BS.BUS\_STOP\_ID = SP.BUS\_STOP\_ID;

**SQL Output**





**Response Time: 680ms**

**Elastic Search Query**

**Sub Query 1:**

URL: http://ec2-35-183-29-29.ca-central-1.compute.amazonaws.com:9200/bus\_trips/\_search

{

"\_source" : ["trip\_id", "trip\_headsign"],

"query": {

"bool": {

"should":[

{"match": {"trip\_headsign": "7 GOTTINGEN"}},

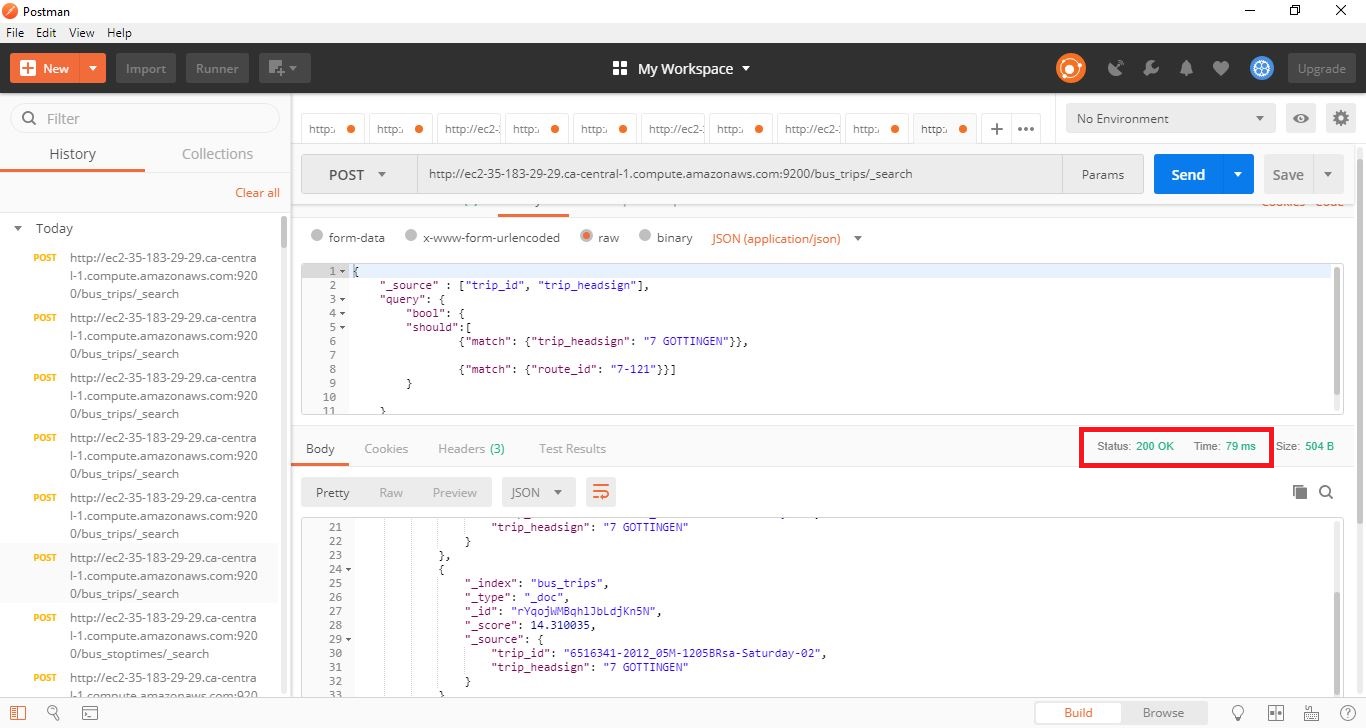
{"match": {"route\_id": "7-121"}}]

}

}

}

**Output**



**Response Time: 79ms**

**Sub Query 2:**

URL: http://ec2-35-183-29-29.ca-central-1.compute.amazonaws.com:9200/bus\_stoptimes/\_search

{

"\_source" : ["trip\_id", "stop\_id"],

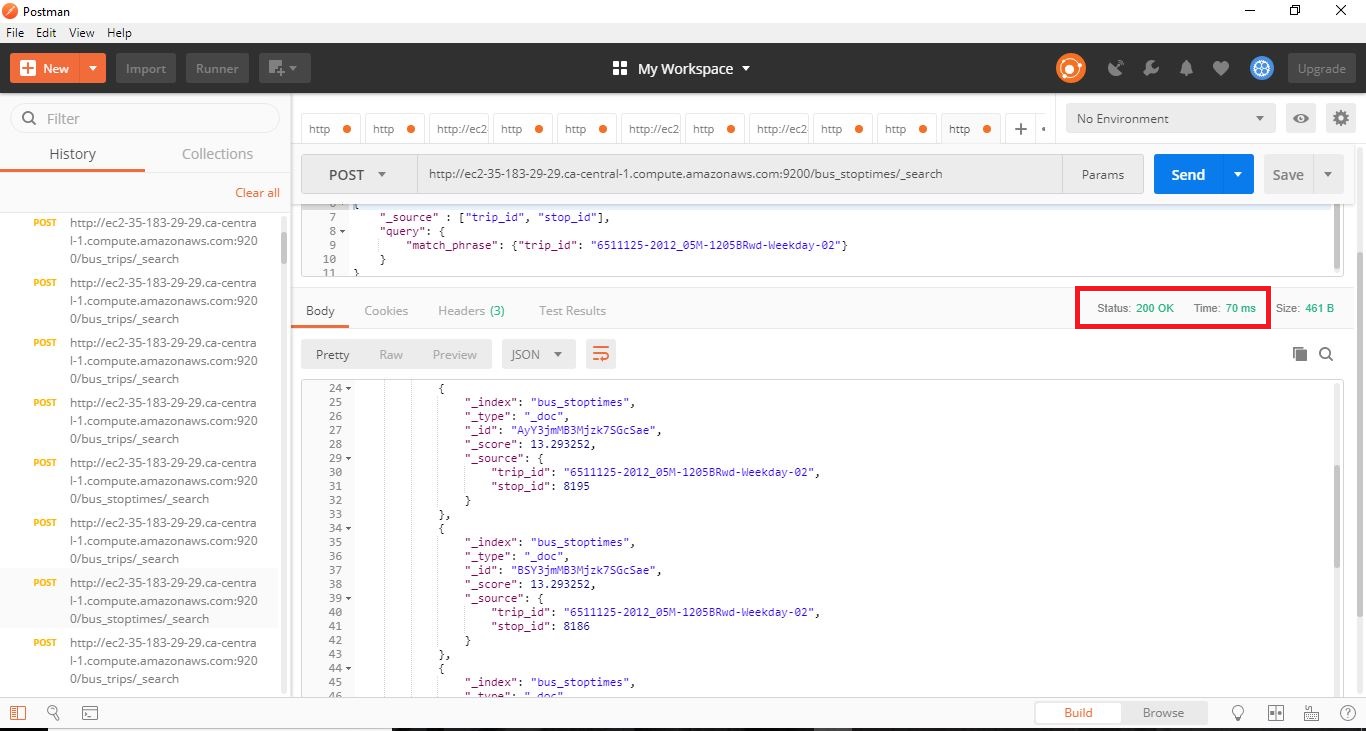
"query": {

"match\_phrase": {"trip\_id": "6511125-2012\_05M-1205BRwd-Weekday-02"}

}

}

**Output**



**Response Time: 70ms**

**Sub Query 3:**

URL: http://ec2-35-183-29-29.ca-central-1.compute.amazonaws.com:9200/bus\_stops/\_search

{

"\_source" : ["name\_stop", "stop\_id"],

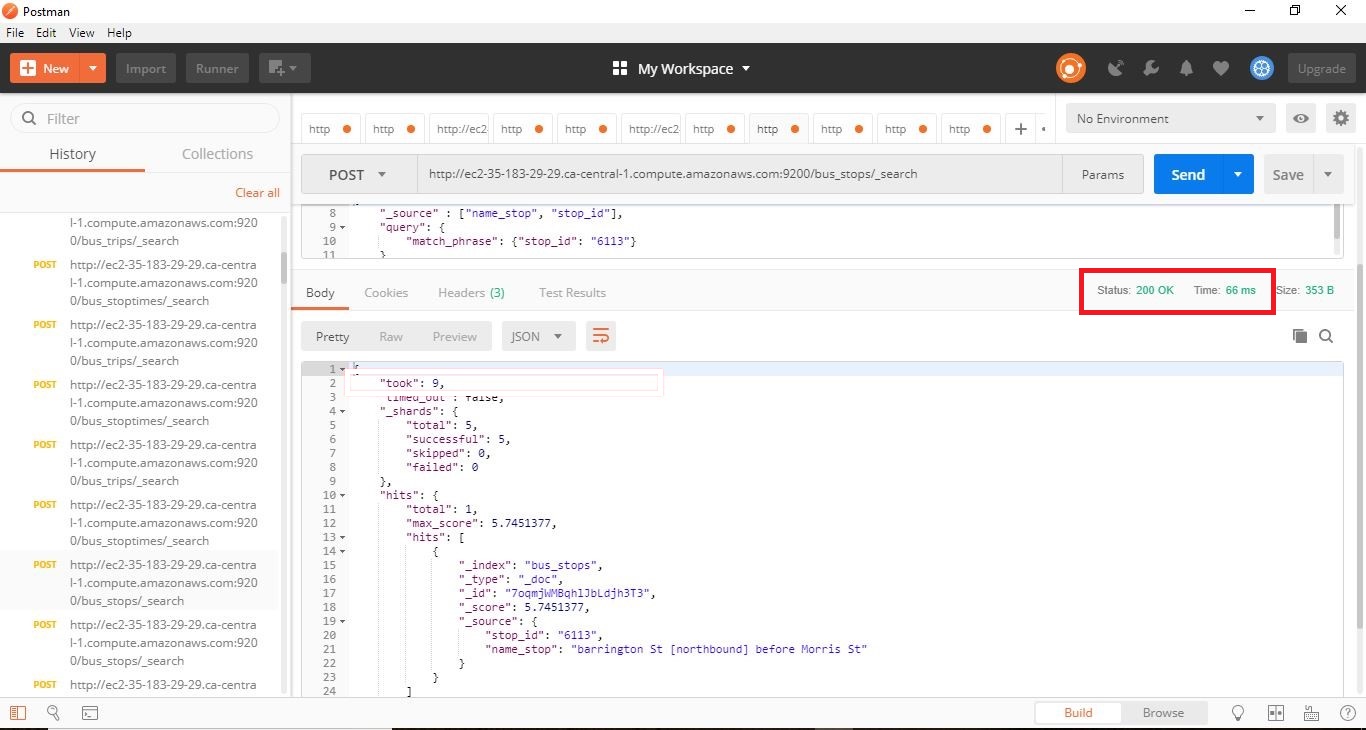
"query": {

"match\_phrase": {"stop\_id": "6113"}

}

}

**Output**



**Response Time: 66ms**

When compared both SQL and Elastic search response times (680ms, 215ms), the elastic search is more time efficient.

1. ***Find top 3 bus stops that are the busiest throughout the day in terms of bus routes. (Hint: The bus stops with high volume of bus routes and close time gaps would be considered as busiest).***

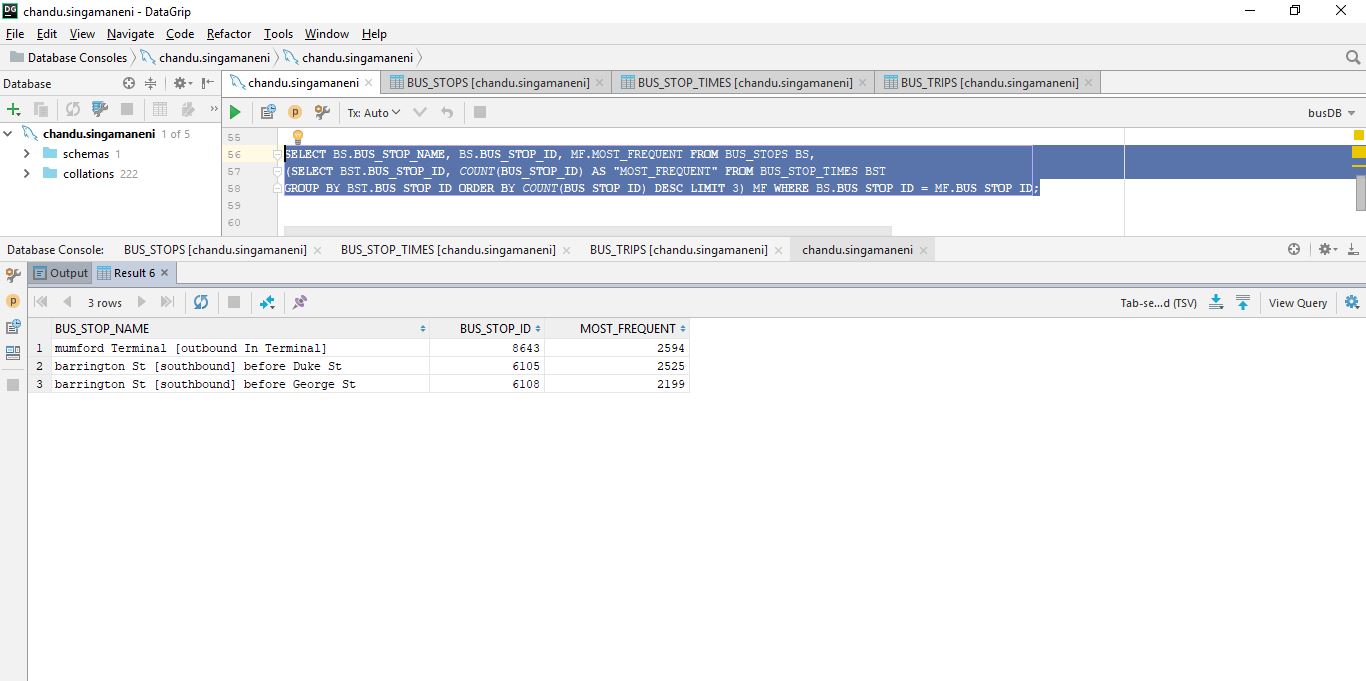
***1.Input: None***

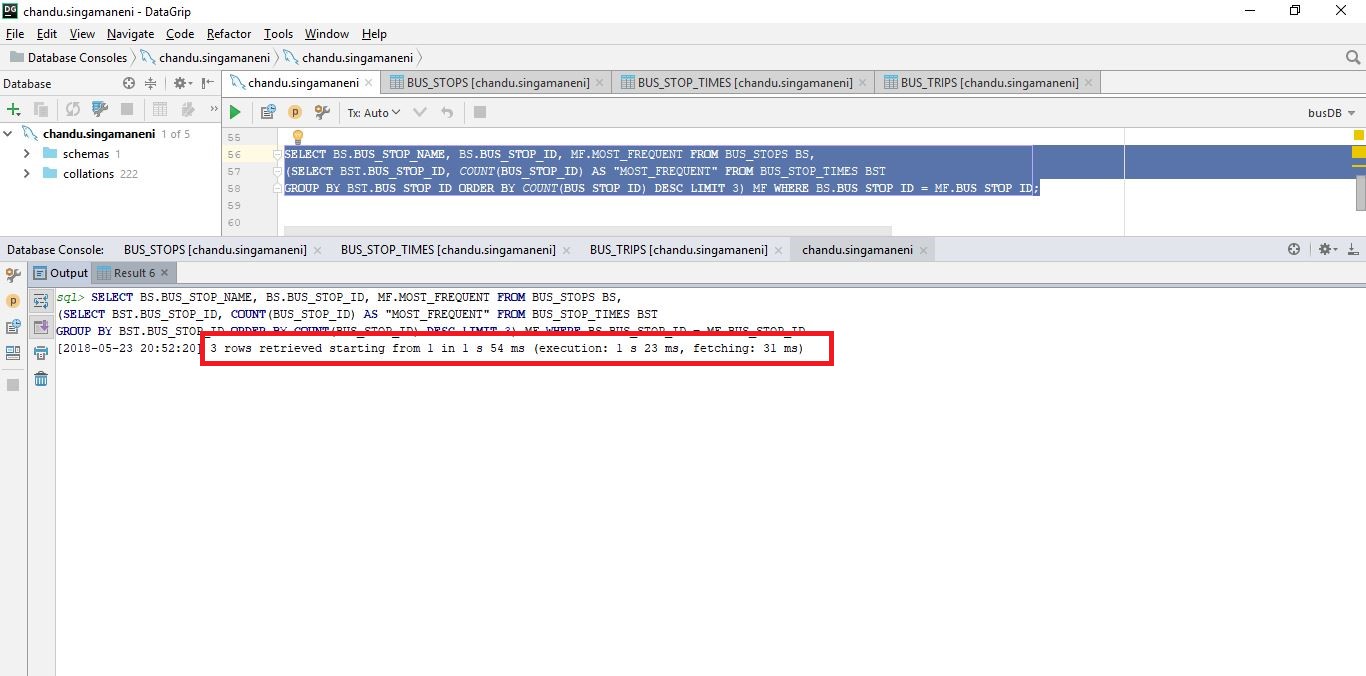
***2.Output: List of Bus Name, response time for the search query***

**SQL Query**

SELECT BS.BUS\_STOP\_NAME, BS.BUS\_STOP\_ID, MF.MOST\_FREQUENT FROM BUS\_STOPS BS,(SELECT BST.BUS\_STOP\_ID, COUNT(BUS\_STOP\_ID) AS "MOST\_FREQUENT" FROM BUS\_STOP\_TIMES BST GROUP BY BST.BUS\_STOP\_ID ORDER BY COUNT(BUS\_STOP\_ID) DESC LIMIT 3) MF WHERE BS.BUS\_STOP\_ID = MF.BUS\_STOP\_ID;

**SQL Output**





**Response Time: 1054ms**

**Elastic Search Query**

**Sub Query 1:**

URL: http://ec2-35-183-29-29.ca-central-1.compute.amazonaws.com:9200/bus\_stoptimes/\_search

{

"size":0,

"aggs": {

"top-terms-aggregation": {

"terms":{

"field":"stop\_id",

"size":3

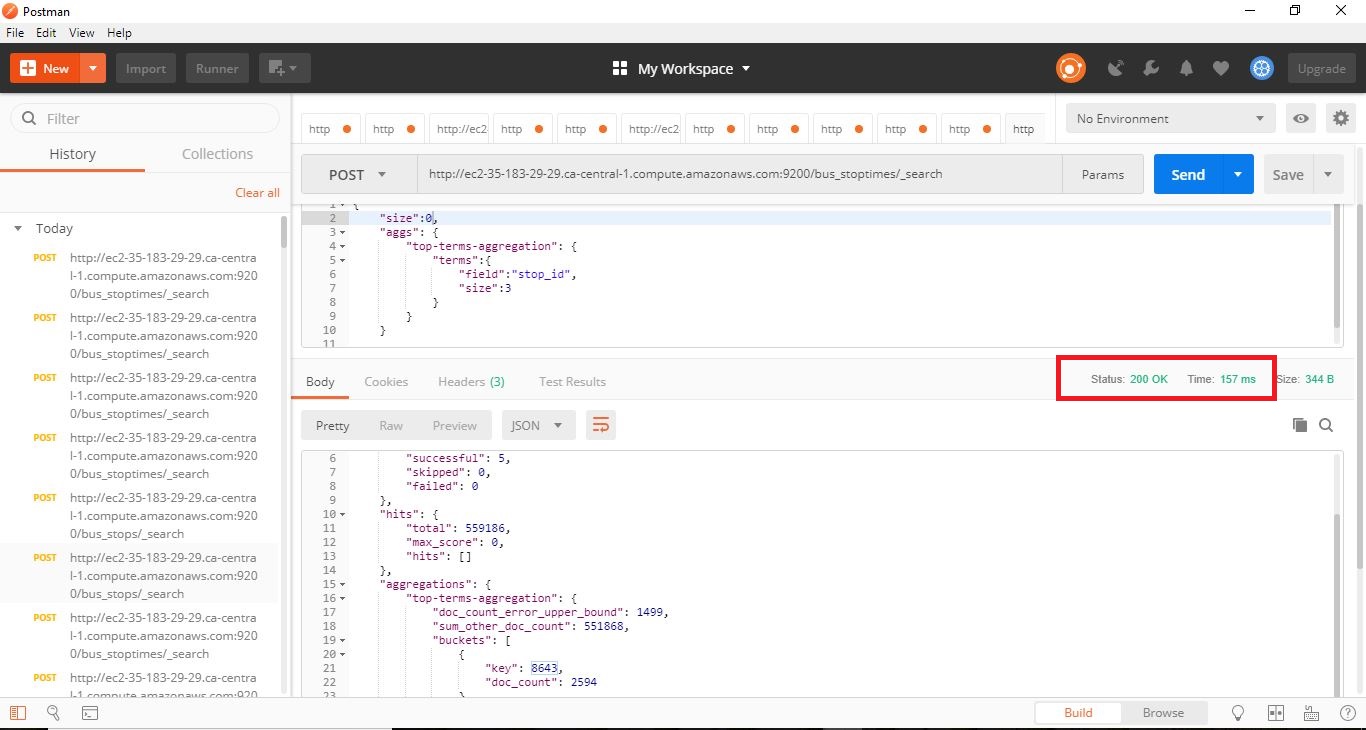
}

}

}

}

**Output**



**Response Time: 157ms**

**Sub Query 2:**

URL: http://ec2-35-183-29-29.ca-central-1.compute.amazonaws.com:9200/bus\_stops/\_search

{

"\_source": ["name\_stop", "stop\_id"],

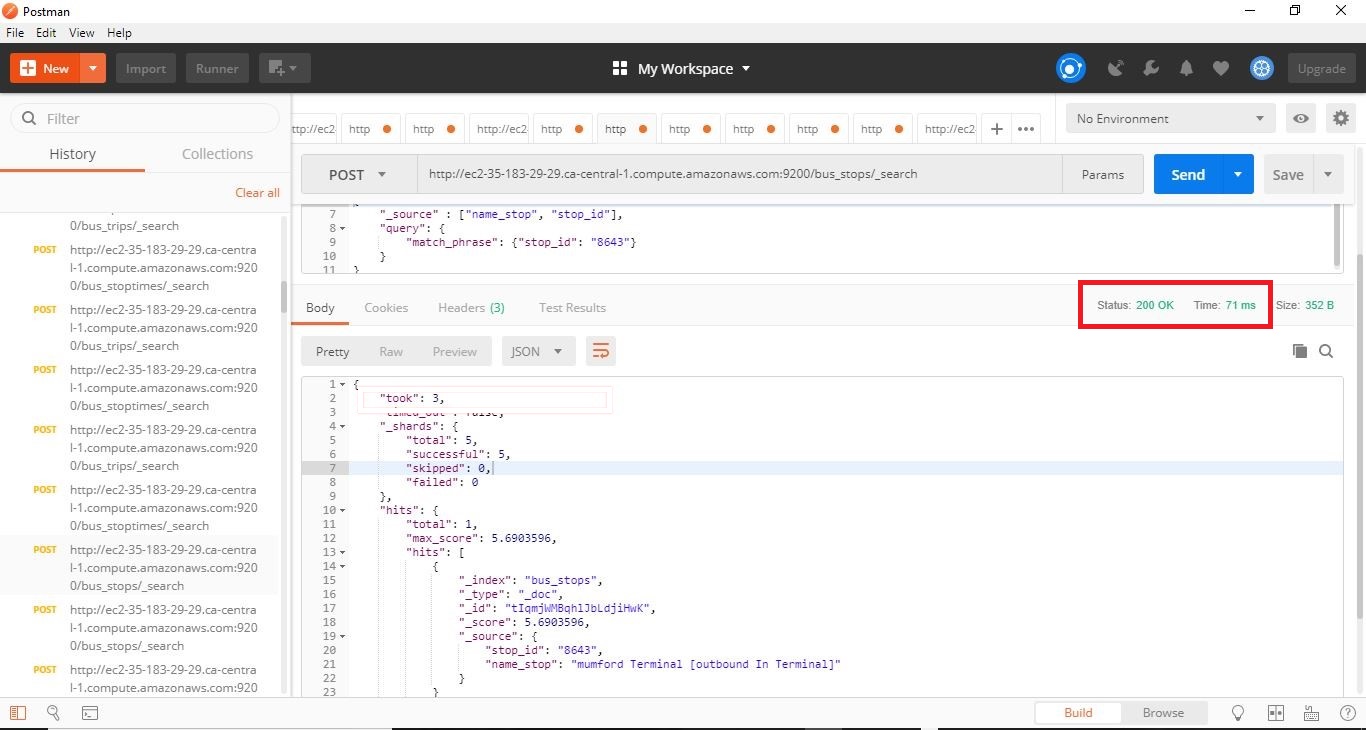
"query": {

"match\_phrase": {"stop\_id": "8643"}

}

}

**Output**



**Response Time: 71ms**

When compared both SQL and Elastic search response times (1054ms, 228ms), the elastic search is more time efficient.

**4. TEST RESULTS:**

The below diagram is the timeline for the time taken by all the queries to execute and fetch the data from the server.

The below diagrams are the comparison of the performance of MySQL and Elastic Search query on the basis execution and fetch time

Elastic search query is 1.8 faster than SQL Query

Elastic search query is 2.7 faster than SQL Query

Elastic search query is 3.1 faster than SQL Query

Elastic search query is 4.6 faster than SQL Query

**5. SUMMARY:**

This assignment helped us to explore different applications such as Amazon could services, MySQL Workbench, Data Grip, Elastic search and Postman. The Halifax transit data was used in classical relational database and compared with the elastic search i.e., NoSQL database to analyse the performance and execution of queries in retrieving data. After using all these tools, we had an opportunity to learn how to use the infrastructure services on amazon cloud and implementation and connectivity of different databases on cloud. Finally, comparing both the response times of MySQL and Elastic search, we analysed that queries in elastic search executed more rapidly and is better and faster mechanism to retrieve and store data. Although, the implementation in traditional databases are easier than the NoSQL databases.

**References**

“Amazon Web Services (AWS) - Cloud Computing Services,” Amazon. [Online]. Available: https://aws.amazon.com/. [Accessed: 23-May-2018].

“DataGrip: Cross-Platform IDE for Databases & SQL by JetBrains,” JetBrains. [Online]. Available: https://www.jetbrains.com/datagrip/. [Accessed: 23-May-2018].

“Must match multiple values,” Stack Overflow. [Online]. Available: https://stackoverflow.com/questions/35583781/must-match-multiple-values?utm\_medium=organic&utm\_source=google\_rich\_qa&utm\_campaign=google\_rich\_qa. [Accessed: 23-May-2018].

“MySQL WorkbenchDownload Now »,” MySQL. [Online]. Available: https://www.mysql.com/products/workbench/. [Accessed: 23-May-2018].

“Open Source Search & Analytics · Elasticsearch,” Open Source Search & Analytics · Elasticsearch. [Online]. Available: https://www.elastic.co/. [Accessed: 23-May-2018].

“Postman,” Debugging and logs. [Online]. Available: https://www.getpostman.com/. [Accessed: 24-May-2018].