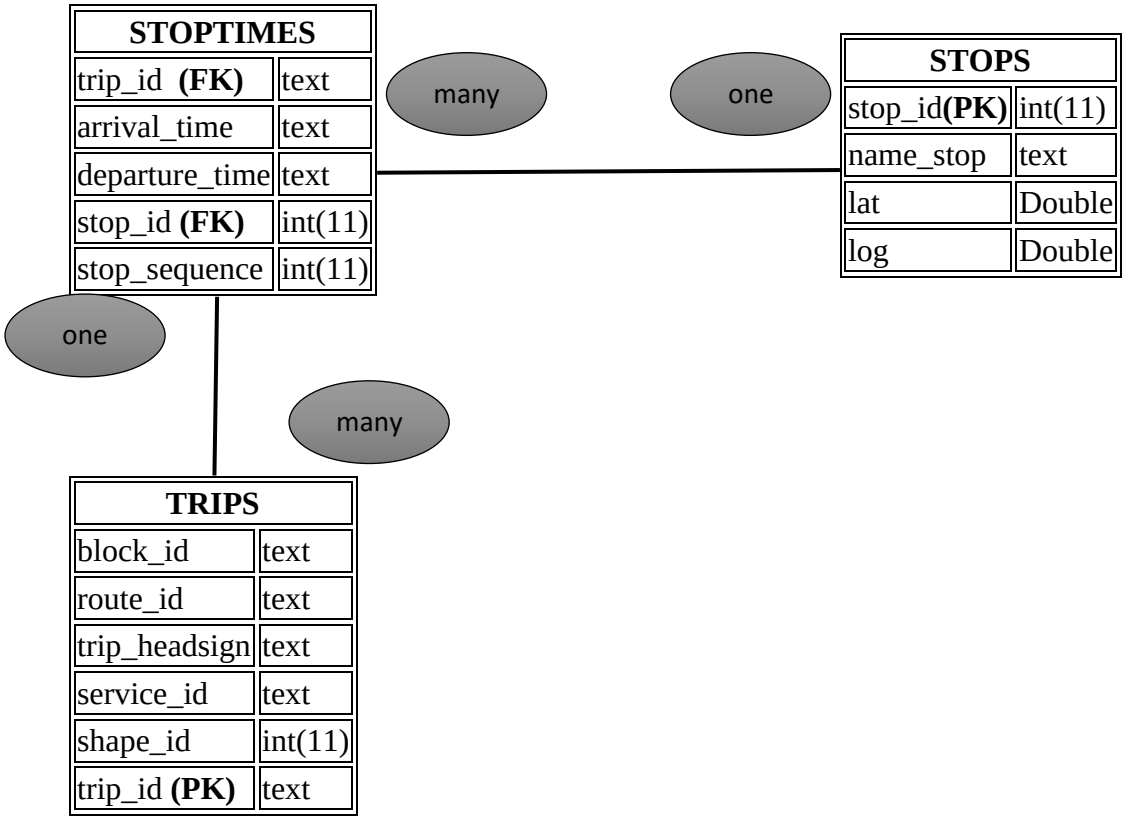
CSCI 5408
Assignment 1: Query Implementation using Relational Database and Elastic Search

1. TASK DESCRIPTION

The task is to get the bus routes dataset and perform set of queries using MySQL in local machine and Elastic search in AWS EC2 instance.

This task is mainly to experiment and understand the difference between SQL and NoSQL queries in terms of performance, efficiency, time consumption, resource utilization and other performance categories.

2. RELATIONAL DATABASE DESIGN



Normalization

The above tables are already decomposed and there are no redundant data in the tables, hence the tables cannot be normalised further.

3. MYSQL QUERIES

a. Find all buses for a particular Bus Stop

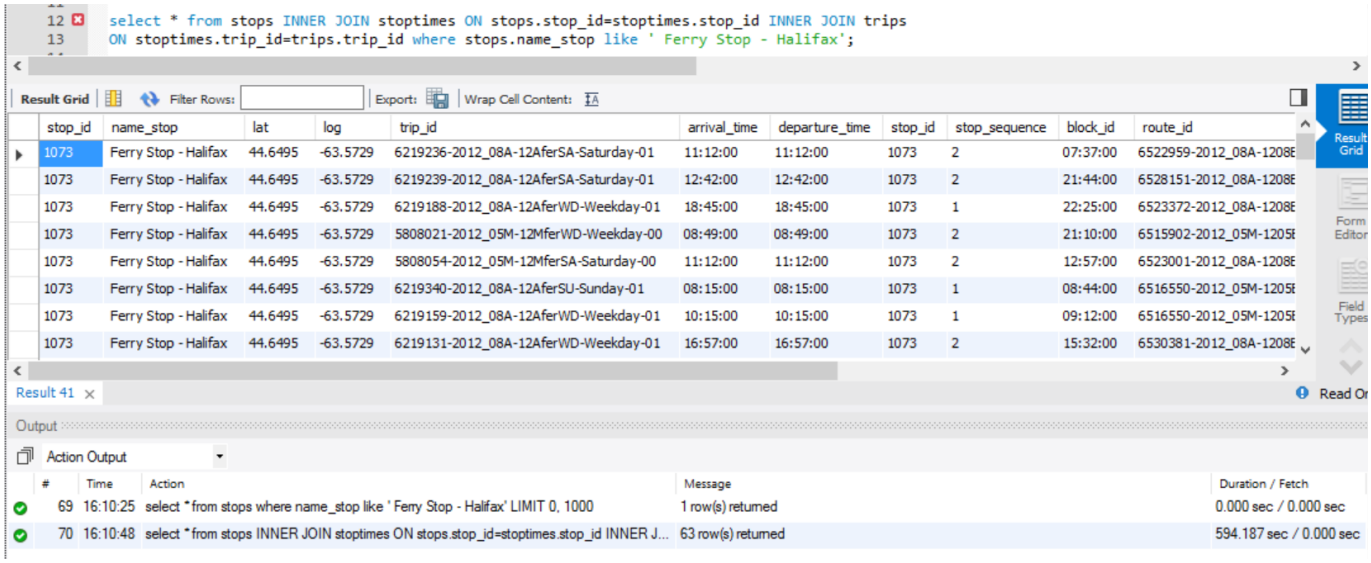
Input: Bus Stop Name

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

Fetching all Columns

select * from stops INNER JOIN stoptimes ON stops.stop_id=stoptimes.stop_id INNER JOIN trips

ON stoptimes.trip_id=trips.trip_id where stops.name_stop like ' Ferry Stop - Halifax';



```
select * from stops INNER JOIN stoptimes ON stops.stop_id=stoptimes.stop_id INNER JOIN trips
ON stoptimes.trip_id=trips.trip_id where stops.name_stop like ' Ferry Stop - Halifax';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

stop_id	name_stop	lat	log	trip_id	arrival_time	departure_time	stop_id	stop_sequence	block_id	route_id
1073	Ferry Stop - Halifax	44.6495	-63.5729	6219236-2012_08A-12AferSA-Saturday-01	11:12:00	11:12:00	1073	2	07:37:00	6522959-2012_08A-1208E
1073	Ferry Stop - Halifax	44.6495	-63.5729	6219239-2012_08A-12AferSA-Saturday-01	12:42:00	12:42:00	1073	2	21:44:00	6528151-2012_08A-1208E
1073	Ferry Stop - Halifax	44.6495	-63.5729	6219188-2012_08A-12AferWD-Weekday-01	18:45:00	18:45:00	1073	1	22:25:00	6523372-2012_08A-1208E
1073	Ferry Stop - Halifax	44.6495	-63.5729	5808021-2012_05M-12MferWD-Weekday-00	08:49:00	08:49:00	1073	2	21:10:00	6515902-2012_05M-1205E
1073	Ferry Stop - Halifax	44.6495	-63.5729	5808054-2012_05M-12MferSA-Saturday-00	11:12:00	11:12:00	1073	2	12:57:00	6523001-2012_08A-1208E
1073	Ferry Stop - Halifax	44.6495	-63.5729	6219340-2012_08A-12AferSU-Sunday-01	08:15:00	08:15:00	1073	1	08:44:00	6516550-2012_05M-1205E
1073	Ferry Stop - Halifax	44.6495	-63.5729	6219159-2012_08A-12AferWD-Weekday-01	10:15:00	10:15:00	1073	1	09:12:00	6516550-2012_05M-1205E
1073	Ferry Stop - Halifax	44.6495	-63.5729	6219131-2012_08A-12AferWD-Weekday-01	16:57:00	16:57:00	1073	2	15:32:00	6530381-2012_08A-1208E

Result 41

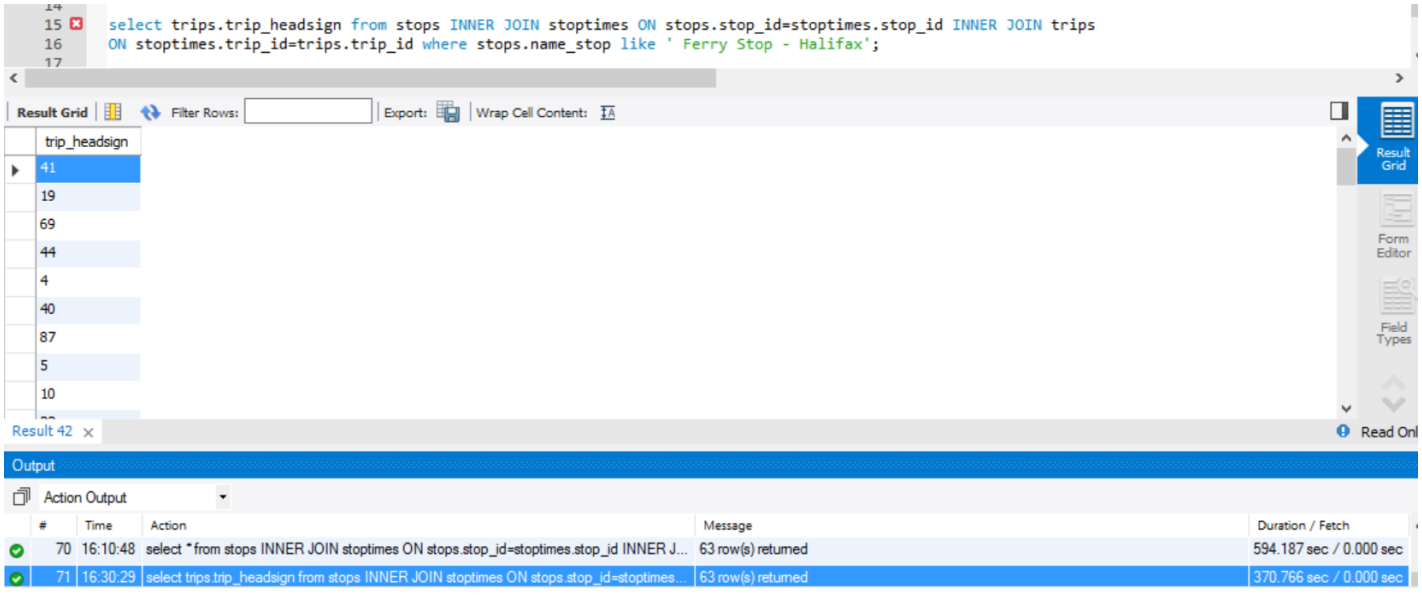
Output

Action Output

#	Time	Action	Message	Duration / Fetch
69	16:10:25	select * from stops where name_stop like ' Ferry Stop - Halifax' LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
70	16:10:48	select * from stops INNER JOIN stoptimes ON stops.stop_id=stoptimes.stop_id INNER J...	63 row(s) returned	594.187 sec / 0.000 sec

select trips.trip_headsign from stops INNER JOIN stoptimes ON stops.stop_id=stoptimes.stop_id INNER JOIN trips ON stoptimes.trip_id=trips.trip_id where stops.name_stop like ' Ferry Stop - Halifax';

Fetching only the list of buses for a particular bus_stop the execution time shown in snapshot is 370.766 sec.



```
select trips.trip_headsign from stops INNER JOIN stoptimes ON stops.stop_id=stoptimes.stop_id INNER JOIN trips
ON stoptimes.trip_id=trips.trip_id where stops.name_stop like ' Ferry Stop - Halifax';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

trip_headsign
41
19
69
44
4
40
87
5
10

Result 42

Output

Action Output

#	Time	Action	Message	Duration / Fetch
70	16:10:48	select * from stops INNER JOIN stoptimes ON stops.stop_id=stoptimes.stop_id INNER J...	63 row(s) returned	594.187 sec / 0.000 sec
71	16:30:29	select trips.trip_headsign from stops INNER JOIN stoptimes ON stops.stop_id=stoptimes...	63 row(s) returned	370.766 sec / 0.000 sec

b. Find buses between two time ranges

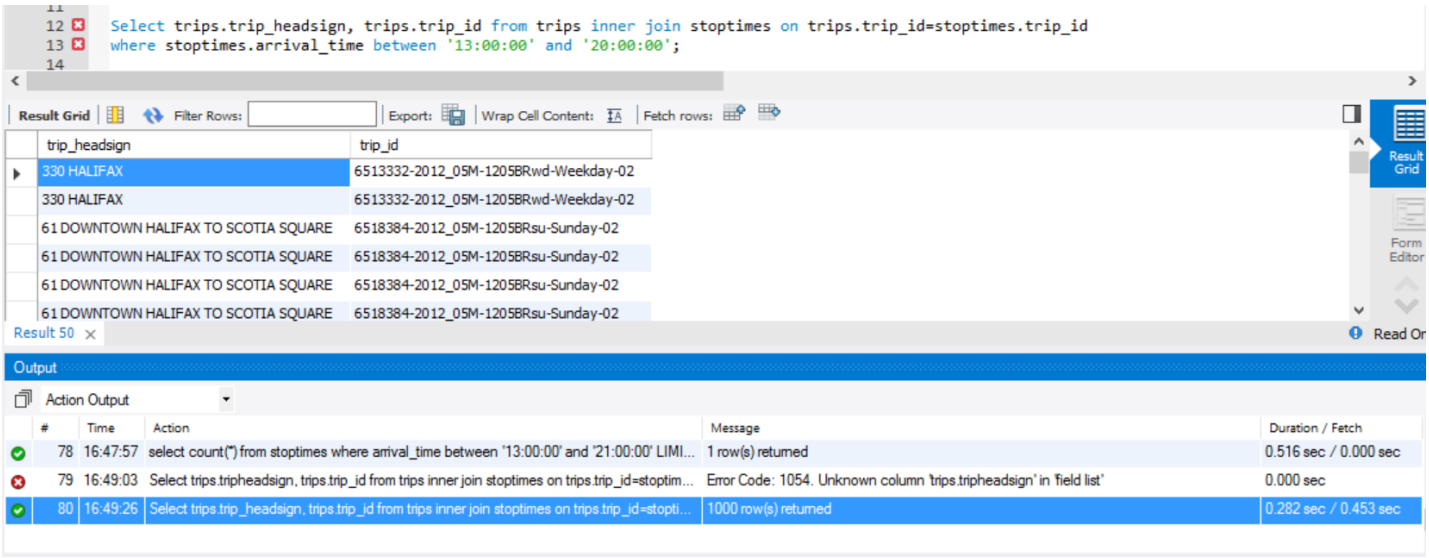
1. Input: Time Range 1 (hh:mm:ss), Time Range 2 (hh:mm:ss)

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

Select trips.trip_headsign, trips.trip_id from trips

inner join stoptimes on trips.trip_id=stoptimes.trip_id

where stoptimes.arrival_time between '13:00:00' and '20:00:00';

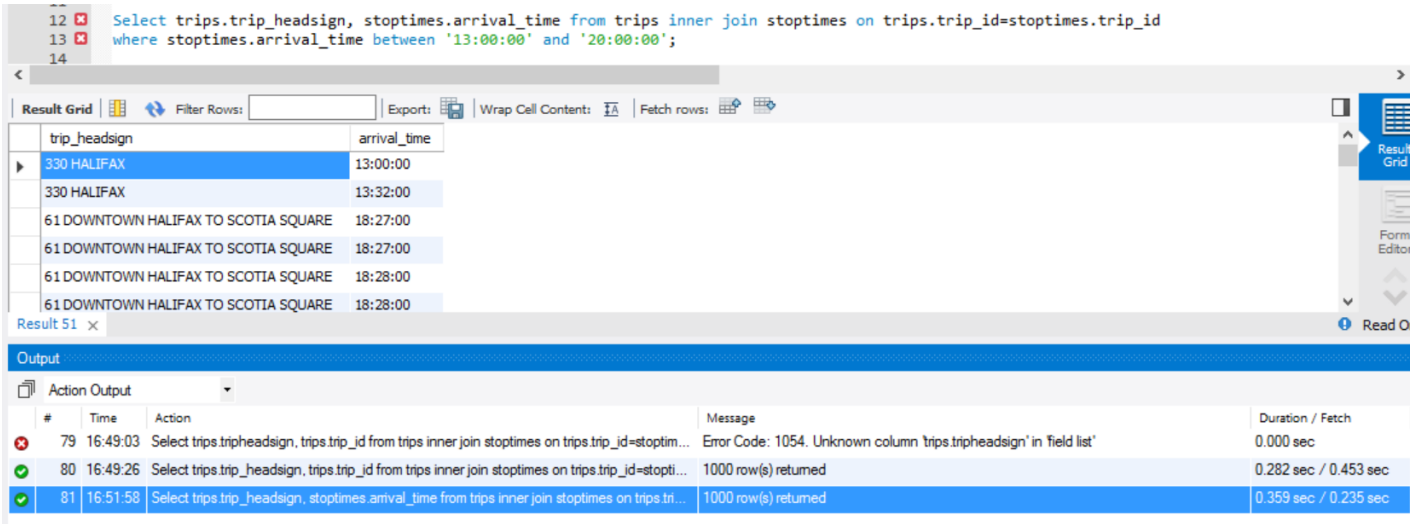


Select trips.trip_headsign, stoptimes.arrival_time from trips

inner join stoptimes on trips.trip_id=stoptimes.trip_id

where stoptimes.arrival_time between '13:00:00' and '20:00:00';

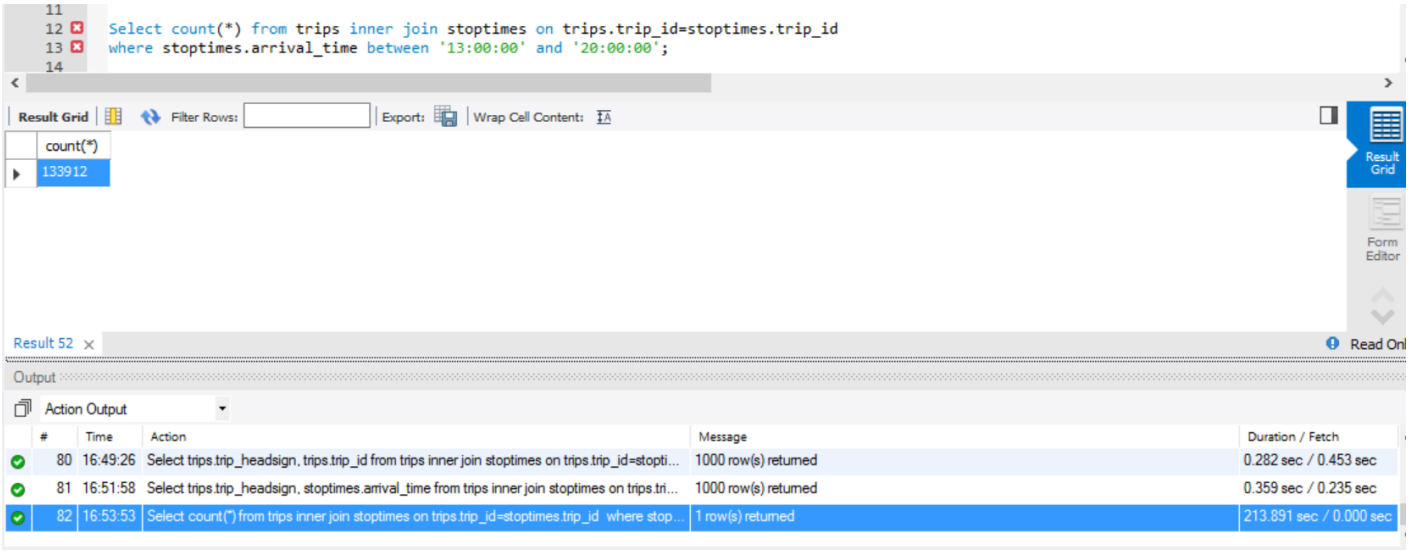
This query is to select list of all buses between time interval as shown in the query, it takes 0.359 seconds to execute the query as shown below.



Select count(*) from trips

inner join stoptimes on trips.trip_id=stoptimes.trip_id

where stoptimes.arrival_time between '13:00:00' and '20:00:00';



It takes 213 seconds to fetch all the records, the time difference between different execution of queries is because MYSQL is set to display only 1000 records at a time so it does not fetch all the matching records. This should be considered while comparing the performance.

c. 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

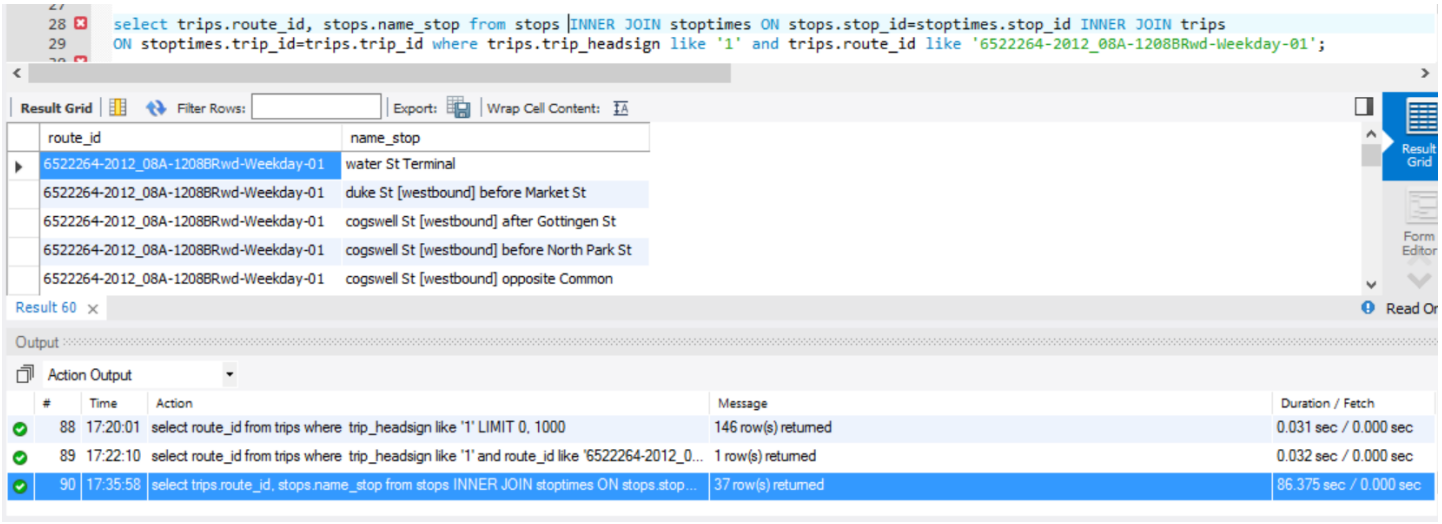
select trips.route_id, stops.name_stop from stops

INNER JOIN stoptimes ON stops.stop_id=stoptimes.stop_id

INNER JOIN trips ON stoptimes.trip_id=trips.trip_id

where trips.trip_headsign like '1' and trips.route_id like '6522264-2012_08A-1208BRwd-Weekday-01';

Below query fetches list of all the routes given the bus name and route name (route_id) as given in the question, time taken is 86 seconds as shown below.

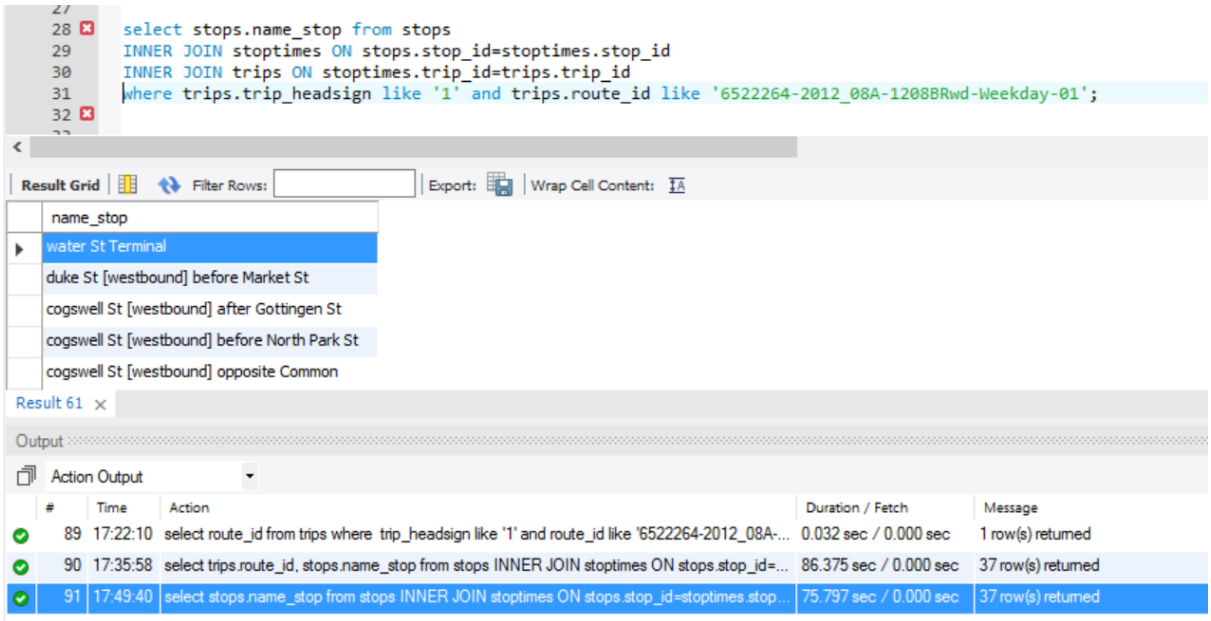


select stops.name_stop from stops

INNER JOIN stoptimes ON stops.stop_id=stoptimes.stop_id

INNER JOIN trips ON stoptimes.trip_id=trips.trip_id

where trips.trip_headsign like '1' and trips.route_id like '6522264-2012_08A-1208BRwd-Weekday-01';

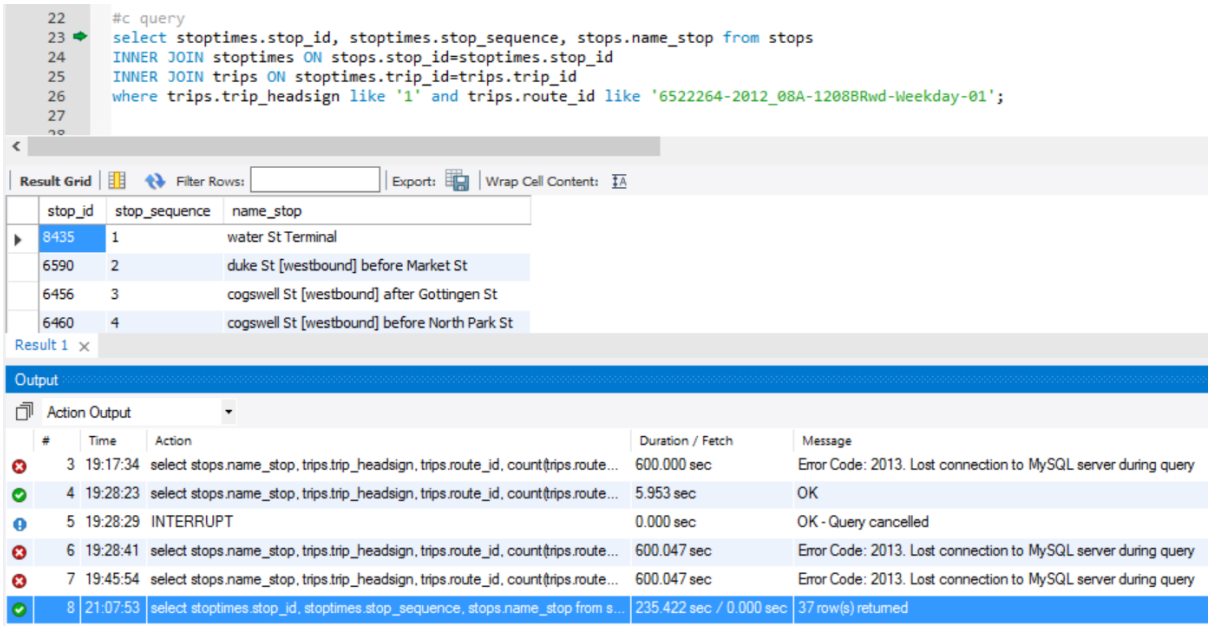


select stoptimes.stop_id, stoptimes.stop_sequence, stops.name_stop from stops

INNER JOIN stoptimes ON stops.stop_id=stoptimes.stop_id

INNER JOIN trips ON stoptimes.trip_id=trips.trip_id

where trips.trip_headsign like '1' and trips.route_id like '6522264-2012_08A-1208BRwd-Weekday-01';



Time duration: 235 seconds.

d. 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.

select stops.name_stop, trips.trip_headsign, trips.route_id, count(trips.route_id) from trips

INNER JOIN stoptimes ON trips.trip_id=stoptimes.trip_id

inner join stops on stops.stop_id=stoptimes.stop_id

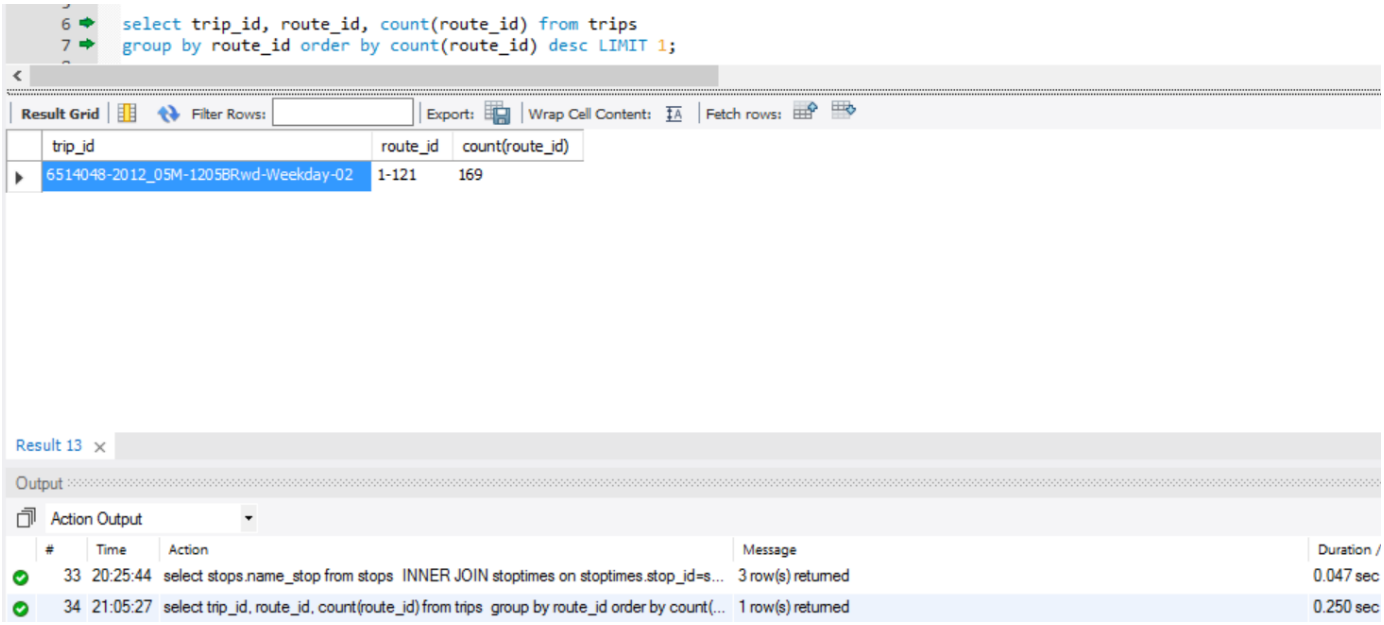
group by(trips.route_id) order by count(trips.route_id) desc LIMIT 3;

The above query as a single query is not getting executed. The query runs for 600 seconds after which it says connection lost. Hence, I am splitting the above query as two parts as shown below

select trip_id, route_id, count(route_id) from trips

group by route_id order by count(route_id) desc LIMIT 1;

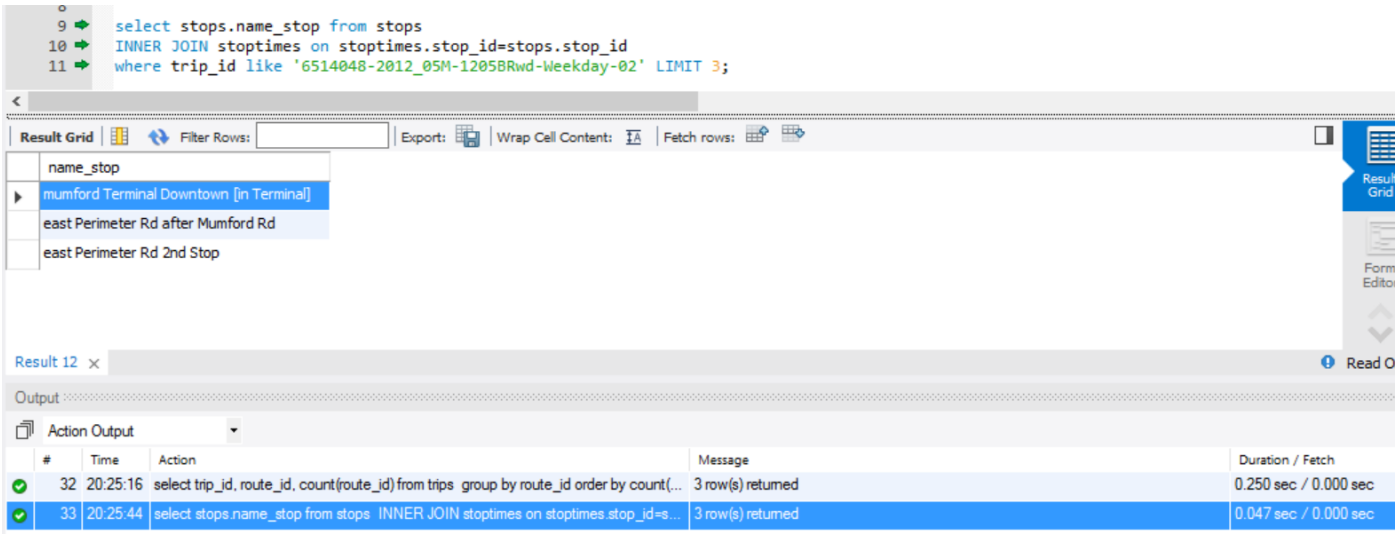
The above query is collecting the trip_id which has the maximum number of routes. This query takes 0.250 seconds to execute.



select trip_id, route_id, count(route_id) from trips

group by route_id order by count(route_id) desc LIMIT 1;

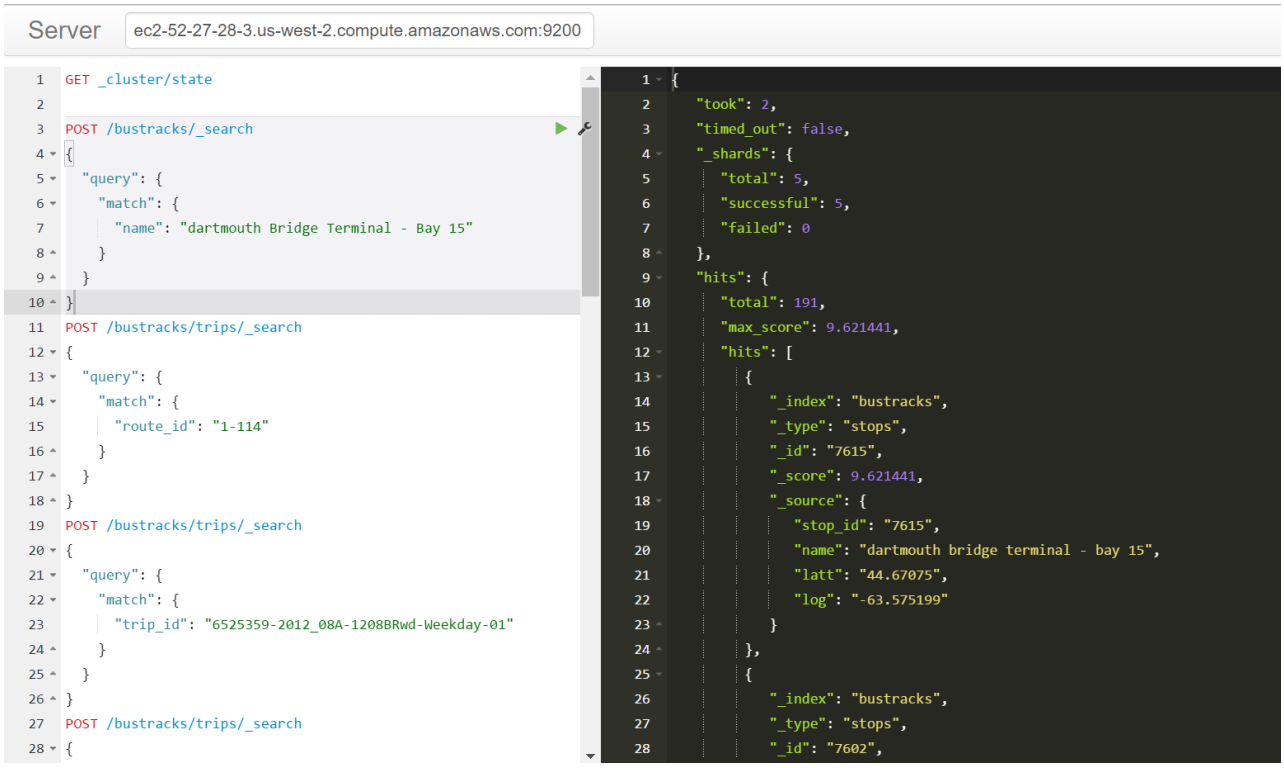
Three bus stops that are part of the trip_id with maximum number of routes is selected below. The query takes 0.047 seconds to execute as shown below.



4. ELASTICSEARCH QUERIES

CLUSTER STATE - USING SENSE EXTENTION

SAMPLE QUERY



The queries are written and executed in python. Python directly connected to instance using below code snippet. Hence the code and output are stored in file and embedded below.

from elasticsearch import Elasticsearch

es = Elasticsearch(['ec2-52-27-28-3.us-west-2.compute.amazonaws.com:9200'])

Query A – Python code with elastic search embedded

Execution time – 4 seconds


Query A - text file.txt


Query A Python Code.py

Output for Query A


Output of Query A.txt

Query B – Python code with elastic search embedded

Execution time – 5 seconds


Query B - text file.txt


Query B Python Code.py

Output for Query B


Output of Query B.txt

Query C – Python code with elastic search embedded

Execution time – 10 seconds


Query C - text file.txt


Query C Python Code.py

Output for Query C


Output of Query C.txt

Query D – Python code with elastic search embedded

Execution time – 400 seconds


Query D - text file.txt


Query D Python Code.py

Output for Query D


Output of Query D.txt

5. COMPARISON OF PERFORMANCE AND EVALUATION

Time duration in seconds		
Query	**SQL**	**NoSQL**
A	370	4
B	213	5
C	235	10
D	0.29/600	400

The time difference between execution of some MYSQL and NOSQL queries is because, MYSQL is set to display only 1000 records at a time so it does not fetch all the matching records. This should be considered while comparing the performance.

Taking this onto consideration, the NOSQL gives better performance

LOCAL MACHINE CPU UTILIZATION FOR MYSQL WHILE EXECUTING A COMPLEX QUERY

Windows Start-Up Application	0%	0.1 MB	0 MB/s	0 Mbps
Client Server Runtime Process	0.2%	0.6 MB	0 MB/s	0 Mbps
Client Server Runtime Process	0%	0.5 MB	0 MB/s	0 Mbps
Windows Session Manager	0%	0.1 MB	0 MB/s	0 Mbps
Task Manager	0.8%	12.8 MB	0 MB/s	0 Mbps
Service Host: Local System (Net...	0.8%	69.5 MB	0 MB/s	0 Mbps
Service Host: Local Service (Net...	0%	6.2 MB	0 MB/s	0 Mbps
appmodel (2)	0%	2.5 MB	0.1 MB/s	0 Mbps
Service Host: Windows Image A...	0%	0.5 MB	0.1 MB/s	0 Mbps
AVG Scanning Core Module - S...	0.2%	19.2 MB	0.1 MB/s	0 Mbps
AVG Resident Shield Service	0%	6.2 MB	0.1 MB/s	0 Mbps
System and compressed memory	1.0%	406.4 MB	0.1 MB/s	0 Mbps
MySQL Workbench	0.1%	21.2 MB	0.1 MB/s	0 Mbps
Google Chrome	0%	61.3 MB	0.1 MB/s	0 Mbps
VProtect Application (32 bit)	0%	6.9 MB	0.1 MB/s	0 Mbps
Cortana	5.5%	88.0 MB	0.1 MB/s	0 Mbps
mysqld	77.2%	14.7 MB	0.2 MB/s	0 Mbps
Microsoft Windows Search Inde...	6.8%	38.1 MB	0.3 MB/s	0 Mbps

Here the CPU utilization is 77% and mysqlid utilizes 14 MB of memory and MySQL Workbench utilizes 21 MB of memory.

AWS INSTANCE CPU UTILIZATION GRAPH SHOWN BELOW

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS
	i-07d1bf5a660da2b4c	t2.micro	us-west-2b	running	2/2 checks ...	None	ec2-52-27-28-3.(

6. SUMMARY OF OBSERVATIONS:

- Elasticsearch is efficient while searching indexed text and it goes slow while used inside a script to join the output of one search inside another.
- In short, this is not very efficient for perfroming joins which needs extra support of scripts or scripting languages like python.

- Using scripts adds a lot of additional overhead apart from searching.
- MySQL is efficient in performing joins and is less time consuming only with respect to joins.

TOOLS: MySQL Workbench, Python (import elasticsearch package)

7. AWS CONFIGURATION

PUBLIC IP of EC2 instance created: **52.27.28.3**

DNS of EC2 instance created: **ec2-52-27-28-3.us-west-2.compute.amazonaws.com**

TEAM:

ANURAG TENKOTI (B00738753)

YAMUNA JAYABALAN (B00741912)

8. TASK BREAKDOWN

ANURAG: SQL Queries - A, B, NOSQL , AWS Configuration

YAMUNA: SQL Queries – C, D, NOSQL Queries – A, B, C, D

REPORT: YAMUNA