BIG DATA IN HEALTH USING DIFFERENT DATA MODELS-A COMPARISON OF PERFORMANCE

A PROJECT REPORT

OF PROJECT-2 (IT 892)

BACHELOR OF TECHNOLOGY
in
Information Technology

(From **Maulana Abul Kalam Azad University of Technology, West Bengal**)

SUBMITTED BY

Arunava Lahiri (13000216122)

Ayush Singhi (13000216113)

Ishmita Basu (13000216098)

Parna Chakraborty (13000216076)

Under the Guidance of

Prof. Poly Sil Sen

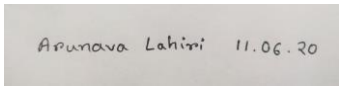
Department of Information Technology
Techno Main Salt Lake
Kolkata -700091

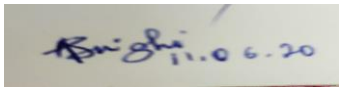
Department of Information Technology
Techno Main Salt Lake
EM-4/1, Salt Lake, Kolkata-700091

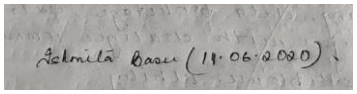
BONAFIDE CERTIFICATE

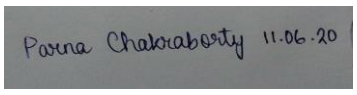
Certified that this report for the project titled "Health Care Data Analysis using Big Data" is a part of the final year project work being carried out by "Arunava Lahiri, Ayush Singhi, Ishmita Basu, Parna Chakraborty" as partial fulfillment for the degree of Bachelor of Technology in Information Technology, Maulana Abul Kalam University of Technology, West Bengal, under my supervision.

Full Signature of the Candidates (with date

1. 

2. 

3. 

4. 

(Signature of the Supervisor)

Dr. Subhamita Mukherjee
HOD, Information Technology

Department of Information Technology
Techno India, Salt Lake.
Kolkata -700091.

ACKNOWLEDGEMENT

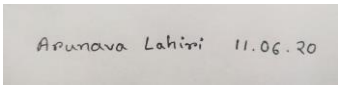
It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our respected and esteemed person Prof. Poly Sil Sen of the Dept. of Information Technology, Techno Main, Salt Lake, for his/ her valuable guidance, encouragement and helping us to pursue with our project work. His/ Her useful suggestions for this work and co-operative behavior are sincerely acknowledged.

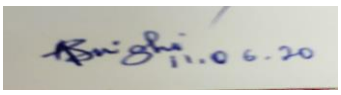
We would like to express our sincere thanks to the teaching fraternity of the Department of Information Technology, for giving us this opportunity to undertake this project and also supporting us whole heartedly.

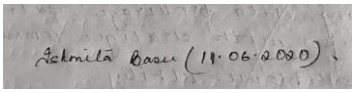
We also wish to express our gratitude to the HOD and all our teachers of the Department of Information Technology for their kind hearted support, guidance and utmost endeavor to groom and develop our academic skills.

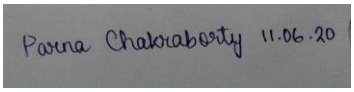
At the end we would like to express our sincere thanks to all our friends and others who helped us directly or indirectly during the effort in shaping this concept till now.

Full Signature of the Candidates (with date)

1. 

2. 

3. 

4. 

ABSTRACT

Big data in healthcare is important as it can be used in the prediction of outcome of diseases prevention of co-morbidities, mortality and saving the cost of medical treatment. In many countries, big data has becoming an important database where information generated could be used for treatment and management of diseases. [1]

The projected growth rate of volume of data being generated in the healthcare industry will soon being zettabyte or yottabyte scale. The traditional RDBMS is not designed to scale up to meet the exploding amount of data, whereas Cassandra offers the flexibility to scale up to any size at the lowest cost, just by adding nodes or clusters to the environment whenever required [2].

The objective of our project is to propose a feasible solution by computing and learning the data. It aims to foster the research, availability and accessibility in the field of health care. This project also provides measurable benefits to improve the field of health care. Data can be both structured and unstructured like clinical analysis, patient history, reports, medical emergencies and so on.

1.Health data is stored in CASSANDRA using health data files considering the same set of queries

2.Next the performances of these queries in Cassandra are observed using CQL.

3.Different use cases have been considered and have been performed using health tracker data functionalities of Cassandra.

TABLE OF CONTENT

TITLE	PAGE NO.
Title Page	
Certificate	i
Abstract	ii
Acknowledgement	iii
Table of Contents	iv
1. Introduction	1
2. Literature Review	4
3. Definition of The Problem	7
4. Software Design	9
5. Software and Hardware Requirements	11
6. Code Templates	13
6.1 Methodology ……..	14
6.2. Guidelines ……	16
6.2.1 Column Family Design …….	16
6.2.2 Guideline For Running The Java Code …….	17
6.3 Code Explanation . ……	19
6.4 Query 1 . …….	22
6.4.1 Code ……	22
6.4.2 Output ……..	23
6.5 Query 2 ……	24
6.5.1 Code ……	24
6.5.2 Output ……	26

TABLE OF CONTENT

TITLE **PAGE NO.**

6.6 Code for Query 3 …….. 27
6.6.1 Code …… 27
6.6.2 Output …… 28

6.7 Code for Query 4 …… 29
6.7.1 Code ……. 29
6.7.2 Output …… 30

7. Output Screens 31
7.1 Query 1 ……. 32
7.1.1 Output in Console …….. 32
7.1.2 Output showing Time of Execution …… 33
7.1.3 Graph …….. 34

7.2 Query 2 …….. 35
7.2.1 Output in Console ……. 35
7.2.2 Output showing Time of Execution ……. 36
7.2.3 Graph …….. 37

7.3 Query 3 …….. 38
7.3.1 Output in Console ……. 38
7.3.2 Output showing Time of Execution …….. 39
7.3.3 Graph …… 40

7.4 Query 4 …… 41
7.4.1 Output in Console …….. 41
7.4.2 Output showing Time of Execution …….. 42
7.4.3 Graph ……. 43

8. Conclusions 44

9. References 46

Chapter – I

INTRODUCTION

Big Data has changed the way we manage, analyze and leverage data in any industry and has the potential to contribute in many areas of the Healthcare industry.

The Healthcare sector is booming at a faster rate and the necessity to manage patient care and innovate medicines has increased synonymously. With the rise in such needs, newer technologies are being adopted in the industry. [6]

At the moment there are good initiatives but this is not enough to keep up with the demand of healthcare services and the rising cost. Relevant improvements can be:

1. Electronic health records which serve customers.
2. Structuring data and information for service optimization.
3. Accurate information about patient can reduce mistakes.
4. Cost optimization through efficiency of new e-health services
5. Increased customer satisfaction
6. Analysis of big datasets for R&D purposes

Using Big Data is both a technological and strategic issue. Besides cost -effectiveness, the healthcare sector eds to achieve improvements in combining data from many sources, with customers, competitors and the market through data driven decision making.

The motivation behind this project is to use large amount of data that will help gather a wider range of information about a patient and help provides better services.

Health sector is a sector where input data is increasing day by day exponentially.

Complexity of maintaining data with respect to its diversity will going to be

higher. Storing of information in digital form accessible and transferable, wherever and whenever needed is essential. Healthcare service providers are sitting on massive pools of information that have been collated from disparate sources and saved across multiple formats and systems. This includes individual patient information generated by past medical and treatment records, wearable devices, smart sensors, and mobile apps, as well as a facility-wide overview of medical data across multiple timeframes.[7]

Chapter – II

LITERATURE SURVEY

Structured Data: Structured Data comprises clearly defined data types whose pattern makes them easily searchable. It has been organized into a formatted repository that is typically, a database.

Big Data in Healthcare

Bernard Mars said “Big Data will leave no sector untouched as it continues to change the way we think about everything from sales to human resources, and medicine and healthcare is no different. For years, the basis of most medical research and discovery has been the collection and analysis of data: who gets sick, how they get sick and why. But now, with sensors in every smartphone and doctors able to share information across disciplines, the quantity and quality of the data available is greater than ever before, which means that the potential for breakthroughs and change is growing just as exponentially.” [3]

Advantages of using Big Data in Healthcare

a) Advanced patient care: Since Big Data works on a huge environment of data, it helps in providing a wider view on the patient entity.

b) Early Intervention: The overall goal of big data in healthcare is to use predictive analysis to find and address medical issues before they turn into larger problems, making the entire process more efficient

c) Fraud Detection: Big data is useful in fighting this because it can access a huge amount of data to find inconsistencies in submitted claims and flag potentially fraudulent claims for further review [4].

Disadvantages of using Big Data in Healthcare

- **a)** Traditional storage can cost lot of money to store big data.
- **b)** Lots of big data is unstructured.
- **c)** Big data analysis violates principles of privacy.
- **d)** It can be used for manipulation of customer records.
- **e)** It may increase social stratification.
- **f)** Big data analysis is not useful in short run. It needs to be analyzed for longer duration to leverage its benefits.
- **g)** Big data analysis results are misleading sometimes.
- **h)** Speedy updates in big data can mismatch real figures [5].

Chapter – III

DEFINITION OF THE PROBLEM

Aging populations and lifestyle changes pose increasing pressures on healthcare systems around the world. Big data in healthcare is important as it can be used in the prediction of outcome of diseases prevention of co-morbidities, mortality and saving the cost of medical treatment.

Majority of the healthcare information are unstructured and unmannered which is so vast and complex. It is nowadays even difficult to manipulate data quickly as its tremendous increase in volume day by day. Data can reside in database in many ways. Of the many ways,

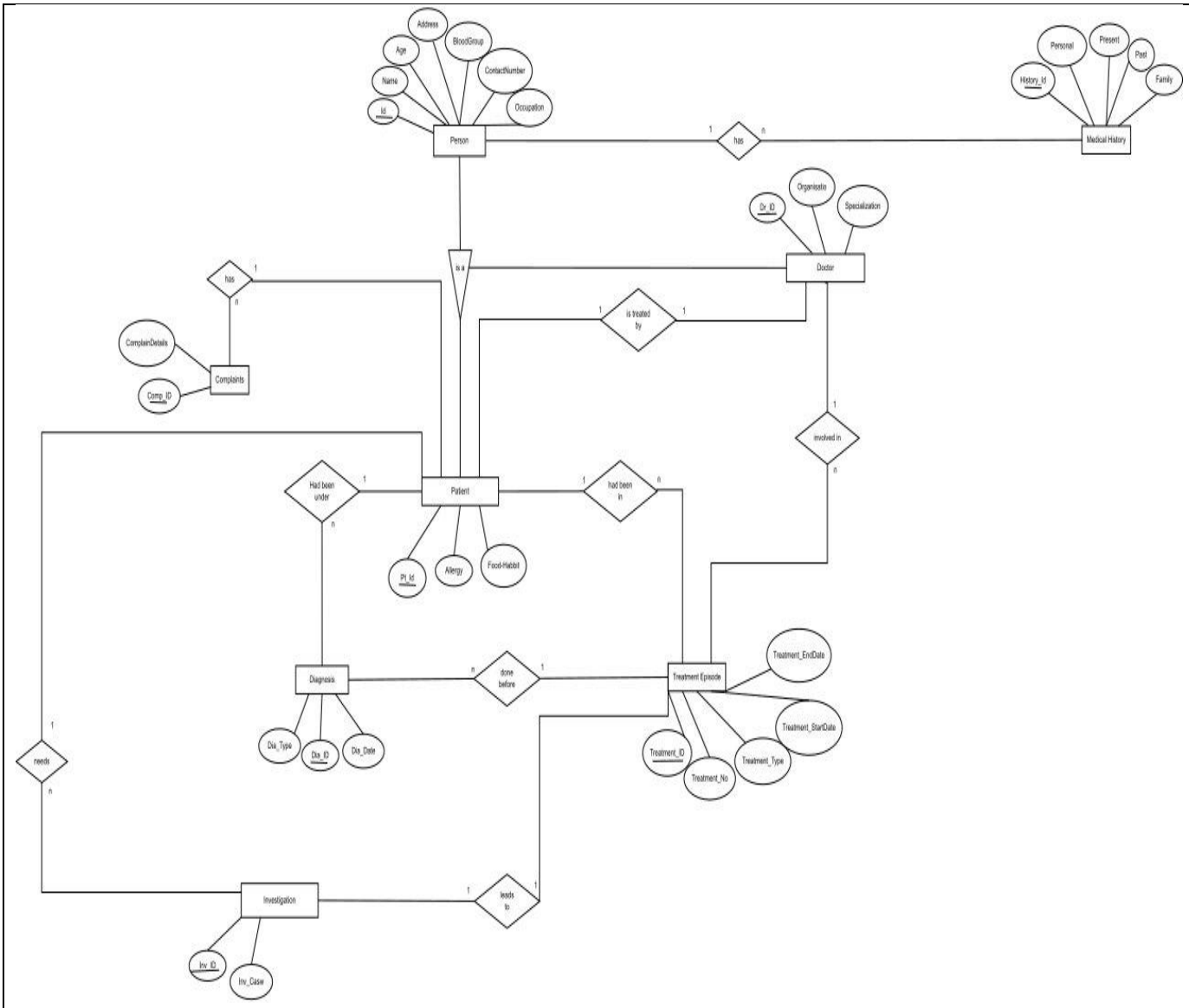
one can be relational and one denormalized.

The objective of our project is to propose a feasible solution by computing and learning the data. It aims to foster the research, availability and accessibility in the field of health care. This

project also provides measurable benefits to improve the field of health care.

Chapter – IV

SOFTWARE DESIGN


Address
Age
BloodGroup
Name
Id
ContactNumber
Occupation
Person
1
has
n
Personal
Present
Past
History_Id
Family
Medical History
Dr_ID
Organisatio
Specialization
Doctor
is a
has
1
n
ComplainDetails
Comp_ID
Complaints
1
is treated by
1
involved in
n
Had been under
1
n
Patient
1
had been in
n
Pt_Id
Allergy
Food-Habbit
needs
1
n
Diagnosis
n
done before
1
Dia_Type
Dia_ID
Dia_Date
Treatment Episode
Treatment_EndDate
Treatment_StartDate
Treatment_ID
Treatment_No
Treatment_Type
Investigation
1
leads to
1
Inv_ID
Inv_Case

Chapter – V

SOFTWARE AND HARDWARE REQUIREMENTS

5.1 HARDWARE REQUIREMENTS

- Laptops with minimum 4GB RAM
- Processor Requirements (minimum 1.5GHz)

5.2 SOFTWARE REQUIREMENTS

- Apache Cassandra
- Eclipse (Java IDE)
- Maven
- Python 2.7
- jdk 1.8

Chapter – VI

CODE TEMPLATES

6.1 METHODOLOGY

The methods used in this project in order to identify next steps for providing better solution in context of health data are as follows:

a) Intake: The scope of data is understood and processed.

b) Initiate: Different platforms and tools required for the project is understood as well.

Cassandra - This is a big data platform to perform operations on the data set. It has embedded CQL to perform operations on our data set.

Programming Language – Stronghold over a particular programming language specifically, JAVA to connect and put the data set on to our platform. The language is user friendly and easy to comprehend.

Documentation – Each step involves proper noting and formulating of all the activities done

c) Aim: Our aim is to test the performance of a data model and storing the health data in NoSQL store. In our design we have used the concept of only denormalization and tested the performance of the queries

6.2 GUIDELINES

6.2.1 COLUMN FAMILY DESIGN

Query Number	Design of the Column Family	Datatype
1	• drId (PK) • specialization • worksForOrganisation • Drname	• Int • Varchar • Varchar • Varchar
2	• ptId (PK) • age • gender • FoodHabit • Name • Religion	• Int • Int • Varchar • Varchar • Varchar • Varchar
3	• drId(PK) • drName • Specialisation • worksForOrganisation • Boolean_var(CK)	• Int • Varchar • Varchar • Varchar • Varchar
4	• ptId • Name • complaintStatus • complaintId(PK) • complaintDet	• Int • Varchar • Varchar • Varchar • Varchar

6.2.2 GUIDELINE FOR RUNNING THE JAVA CODE

The required steps are as follows :

- Cassandra 3.3.0 is installed
- System Requirement: JDK 1.8, Python 2.7
- Path link is added to the environment variables section
- Maven 3.0 is installed [9].
- Path link is added to the environment variables section
- To start Cassandra, “cassandra-f” command is written in the command prompt
- In a new command prompt, to start CQL “cqlsh” is entered.
- In cql command we have to type “Create Keyspace ecommerce with replication = {‘class’:’SimpleStrategy’, ‘replication_factor’ :1}; And then “use ecommerce”. This will create a keyspace called ecommerce; [10]
- Go to command prompt and type “mvn archetype:generate -DgroupId=com.cass.app -DartifactId=app-cass-tutorials -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false” in the workspace .This command will create a folder called “app-cass-tutorials”;
- Go to eclipse ->file -> maven projects->existing->browse->path at the folder has been created.
- Search in google for datastax dependency for maven
- Add the dependency in the pom.xml file.
- Then right click on source and create new java class. This will create the .class java file
- Right this on java fil

After writing the code :

- Run it.
- Go to cql command prompt.
- Run the queries. [11]

6.3 CODE WITH EXPLANATION

```
package com.cass.app;
//Name of the domain created by maven code used in command prompt

import com.datastax.driver.core.Cluster; //Java core driver or api name is datastax [13],[14]

import com.datastax.driver.core.Session;

public class FirstJava { //class where we are coding the connection

private static Cluster cluster;
// Cluster type object variable in used in main method and called without creating object ,that's why it is static

private static Session session;
// Session type object variable in used in main method and called without creating object ,that's why it is static

public static Cluster connect(String node) //function to connect Cluster with Session [8]

{
return Cluster.builder().addContactPoint(node).build();
// Cluster.builder() is helper class to build cluster instances
// Cluster.builder().addContactPoint(node).build()" is the main entry point of the driver
}

public static void main(String args[])
{
cluster = connect(&quot;localhost&quot;); //connecting to our pc hostname
session= cluster.connect(&quot;aru&quot;); //connecting to keyspace which we created database

session.execute(&quot;CREATE TABLE DOCTOR(drId int PRIMARY KEY,drName
```

varchar,specialization varchar,organization varchar); ");
//session.execute() is called to execute whatever we want to execute in database and passed as
//parameter as a string to the function

session.execute("INSERT INTO DOCTOR
(drId,drName,specialization,organization)Values(1006,'Dr.Parna Chakraborty
','GENERAL','CMRI');");

session.execute("INSERT INTO DOCTOR

(drId,drName,specialization,organization)Values(1007,'Dr.Ishmita Basu
','PATHOLOGY','MEDICA');");

session.execute("INSERT INTO DOCTOR

(drId,drName,specialization,organization)Values(1008,'Dr.Ayush Singhi
','HEART','NRS');");

session.execute("INSERT INTO DOCTOR

(drId,drName,specialization,organization)Values(1009,'Dr.Arunava Lahiri
','EYE','BB EYE');");

session.execute("INSERT INTO DOCTOR
(drId,drName,specialization,organization)Values(1010,'Dr.Bhaskar Dibakar
Choudhury
','NERVE','CMRI');");

session.execute("CREATE TABLE PERSON(perid int PRIMARY KEY,pername

varchar,age int,bloodgrp varchar,address varchar,contact varchar,occupation varchar);

");

session.execute("INSERT INTO

PERSON(perid,pername,age,bloodgrp,address,contact,occupation)values(1001,'Namr
ata
Das',22,'B+','9/m.n.k
road','8954246548','post office employee'); ");

session.execute("INSERT INTO

PERSON(perid,pername,age,bloodgrp,address,contact,occupation)values(1002,'Diptes
h
Das',24,'O+','72/c g.t
road','2134564678','accountant'); ");

session.execute("INSERT INTO

PERSON(perid,pername,age,bloodgrp,address,contact,occupation)values(1003,'Madh
abi
Sen',27,'A+','12/z c.t
road','7845126985','bank employee'); ");

session.execute("INSERT INTO

PERSON(perid,pername,age,bloodgrp,address,contact,occupation)values(1004,'Gullu
Majumdar',43,'AB+','56/a b.t
road','8987454122','driver'); ");

session.execute("CREATE TABLE PATIENT(patid int PRIMARY KEY,imm_details
varchar,allergies varchar,foodhabit varchar,address varchar,contact varchar,occupation
varchar); ");

//session.execute("CREATE TABLE PATIENT(person_id int PRIMARY KEY,name
varchar,occupation varchar,addres varchar,email varchar,contact int,gender varchar,dob
varchar,blood_specialization varchar);");

session.close(); **//closing the session object variable;**

cluster.close(); **//closing the cluster object variable;**
}
}

6.4 QUERY 1 : FIND ALL DOCTORS NAMES AND THEIR DETAILS WITH SPECIALIZATION STRING CARDIOLOGIST.

6.4.1 Code

```
static void table1()
{
//session.execute(&amp;quot;create keyspace cas with replication =
{&amp;#39;class&amp;#39;:&amp;#39;SimpleStrategy&amp;#39;,&amp;#39;replication_factor&amp;#39;:1};&amp;quot;);
session.execute(&amp;quot;use cas&amp;quot;);
//session.execute(&amp;quot;create table cas.table1(Specialisation varchar, Organisation varchar,
Dr_Name
varchar , Dr_Id int PRIMARY KEY);&amp;quot;);
String s1,s2,s3;
int s4;
int n=0;
while(n&amp;lt;10000)
{
String strCQL = &amp;quot;INSERT INTO table1 (Specialisation, Organisation, Dr_Name,
Dr_Id) VALUES
(?,?,?,?)&amp;quot;;
s1 = generate_specialisation();
s2 = generate_organisation();
s3 = generate_name();
s4 = gen_id();

PreparedStatement preparedStatement = session.prepare(strCQL);
BoundStatement boundStatement = preparedStatement.bind(s1,s2,s3,s4);
session.execute(boundStatement);
n++;
}

}
```

6.4.2 Output code

```
static void output1() {

String dr_name =null,organisation=null,specialisation=null;
int dr_id;
LocalTime myObj = LocalTime.now();
System.out.println(myObj);
com.datastax.driver.core.ResultSet resultset1 = session.execute(&amp;quot;select dr_id ,
dr_name ,
organisation from cas.table1 where Specialisation = &amp;#39;Cardiologist&amp;#39; ALLOW
FILTERING&amp;quot;);
for(Row row:resultset1)
{
dr_id = row.getInt(&amp;quot;dr_id&amp;quot;);
organisation = row.getString(&amp;quot;organisation&amp;quot;);
//specialisation = row.getString(&amp;quot;specialisation&amp;quot;);
dr_name = row.getString(&amp;quot;dr_name&amp;quot;);
System.out.println(&amp;quot;DR_ID : &amp;quot;+dr_id);
System.out.println(&amp;quot;Organisation: &amp;quot;+organisation);
//System.out.println(&amp;quot;Specialisation: &amp;quot;+specialisation);
System.out.println(&amp;quot;DOCTOR NAME : &amp;quot;+dr_name);

}
LocalTime myObj2 = LocalTime.now();

System.out.println(&amp;quot;Local Time Before Execution : &amp;quot;+myObj);
System.out.println(&amp;quot;Local Time After Execution : &amp;quot;+ myObj2);
long duration = Duration.between(myObj, myObj2).toMillis();
System.out.println(&amp;quot;Duration of execution : &amp;quot;+duration +&amp;quot;ms&amp;quot;);

}
```

6.5 QUERY 2 : FIND ALL INFORMATION OF PATIENTS LIKE DEMOGRAPHIC INFO, RELIGION, FOOD HABIT ETC, FOR A PATIENT.

6.5.1 Code

```
static void table2()
{
//session.execute(&amp;quot;create keyspace cas with replication =
{&amp;#39;class&amp;#39;:&amp;#39;SimpleStrategy&amp;#39;,&amp;#39;replication_factor&amp;#39;:1};&amp;quot;);
session.execute(&amp;quot;use cas&amp;quot;);
//session.execute(&amp;quot;create table cas.table2(age int, foothabit varchar, patient_name
varchar , gender
varchar, religion varchar, patient_id int PRIMARY KEY);&amp;quot;);

String s2,s3,s4,s5;
int s1;
int s6;
for(int i=0;i&amp;lt;10000;i++)
{

String strCQL = &amp;quot;INSERT INTO table2 (age, foothabit, patient_name, gender,
religion, patient_id)
VALUES (?,?,?,?,?,?)&amp;quot;;
s1 = generate_age();
s2 = generate_random_string(40);//foodhabit [12]
s3 = generate_name();
s4 = generate_gender();
s5 = generate_religion();
s6= gen_id();

PreparedStatement preparedStatement = session.prepare(strCQL);
BoundStatement boundStatement = preparedStatement.bind(s1,s2,s3,s4,s5,s6);
session.execute(boundStatement);

}

}
```

```
static String generate_com_status() {
String charac6[]=
{&amp;quot;Recovered&amp;quot;,&amp;quot;Ongoing&amp;quot;,&amp;quot;Cronic&amp;quot;,&amp;quot;Not
Recoverd&amp;quot;};


Random rand=new Random();
return charac6[rand.nextInt(charac6.length)];
}
```

6.5.2 Output

```
static void output2() {

String patient_name =null,gender=null,foothabit=null,religion=null;
int age,patient_id;
LocalTime myObj = LocalTime.now();
System.out.println(&quot;Local Time Before Execution : &quot;+myObj);
com.datastax.driver.core.ResultSet resultset1 = session.execute(&quot;select * from
cas.table2&quot;);

System.out.println(&quot;OUTPUT FOR QUERY 2&quot;);
for(Row row:resultset1)
{
patient_id = row.getInt(&quot;patient_id&quot;);
age = row.getInt(&quot;age&quot;);
religion = row.getString(&quot;religion&quot;);
gender = row.getString(&quot;gender&quot;);
patient_name = row.getString(&quot;patient_name&quot;);
foothabit = row.getString(&quot;foothabit&quot;);
System.out.println(&quot;Patient ID : &quot;+patient_id);
System.out.println(&quot;Patient Name : &quot;+patient_name);
System.out.println(&quot;Religion : &quot;+religion);
System.out.println(&quot;FoodHabit : &quot;+foothabit);

System.out.println(&quot;Gender : &quot;+gender);
System.out.println(&quot;Age : &quot;+age);

}
LocalTime myObj2 = LocalTime.now();
System.out.println(&quot;Local Time Before Execution : &quot;+myObj);
System.out.println(&quot;Local Time After Execution : &quot;+ myObj2);
long duration = Duration.between(myObj, myObj2).toMillis();
System.out.println(&quot;Duration of execution : &quot;+duration +&quot;ms&quot;);

}
```

6.6 QUERY 3 : FIND ALL THE COMPLAINT DETAILS OF A PATIENT.

6.6.1 Code

```
static void table8() {

//session.execute(&amp;quot;create keyspace cas with replication =
{&amp;#39;class&amp;#39;:&amp;#39;SimpleStrategy&amp;#39;,&amp;#39;replication_factor&amp;#39;:1};&amp;quot;);
session.execute(&amp;quot;use cas&amp;quot;);
//session.execute(&amp;quot;create table cas.table8(pt_id int, pt_name varchar , com_det
varchar, com_status
varchar, com_id int PRIMARY KEY);&amp;quot;);

String s3,s4,s5;
int s1,s2;
for(int i=0;i&amp;lt;10000;i++)
{
String strCQL = &amp;quot;INSERT INTO table8 (pt_id, pt_name,
com_det,com_status,com_id) VALUES
(?,?,?,?,?)&amp;quot;;
s1 = gen_id();
s2 = gen_id();
s3 = generate_name();
s4 = generate_random_string(40);//complaint details
s5 = generate_com_status();

PreparedStatement preparedStatement = session.prepare(strCQL);
BoundStatement boundStatement = preparedStatement.bind(s1,s3,s4,s5,s2);
session.execute(boundStatement);

}

}
```

6.6.2 Output

```
static void output8() {

String pt_name =null,com_det=null,com_status=null;
int pt_id,com_id;
//pt_id, pt_name, com_det,com_status,com_id
LocalTime myObj = LocalTime.now();
System.out.println(&quot;Local Time Before Execution : &quot;+myObj);
System.out.println(&quot;Find all the complaint details of a Patient&quot;);
com.datastax.driver.core.ResultSet resultset1 = session.execute(&quot;select * from
cas.table8&quot;);

System.out.println(&quot;OUTPUT FOR QUERY 8&quot;);
for(Row row:resultset1)
{
pt_id = row.getInt(&quot;pt_id&quot;);
com_id = row.getInt(&quot;com_id&quot;);
com_det = row.getString(&quot;com_det&quot;);
com_status = row.getString(&quot;com_status&quot;);
pt_name = row.getString(&quot;pt_name&quot;);
System.out.println(&quot;Patient ID : &quot;+pt_id);
System.out.println(&quot;Patient Name : &quot;+pt_name);
System.out.println(&quot;Complaint ID : &quot;+com_id);
System.out.println(&quot;Complaint Details : &quot;+com_det);
System.out.println(&quot;Complaint Status : &quot;+com_status);

}
LocalTime myObj2 = LocalTime.now();
System.out.println(&quot;Local Time Before Execution : &quot;+myObj);
System.out.println(&quot;Local Time After Execution : &quot;+ myObj2);
long duration = Duration.between(myObj, myObj2).toMillis();
System.out.println(&quot;Duration of execution : &quot;+duration +&quot;ms&quot;);

}
```

6.7 CODE FOR QUERY 4 : FIND ALL THE DOCTORS NAMES WHO ARE ATTACHED WITH AN ORGANIZATION.

6.7.1 Code

```
static void table7() {

//session.execute(&amp;quot;create keyspace cas with replication =
{&amp;#39;class&amp;#39;:&amp;#39;SimpleStrategy&amp;#39;,&amp;#39;replication_factor&amp;#39;:1};&amp;quot;);
session.execute(&amp;quot;use cas&amp;quot;);
session.execute(&amp;quot;create table cas.table7(dr_name varchar, specialisation varchar ,
organisation
varchar,dr_id int ,boolean_val varchar, PRIMARY KEY(dr_id,boolean_val));&amp;quot;);

String s2,s3,s4,s5;
int s1;
for( long i=0;i&amp;lt;10000;i++)
{
String strCQL = &amp;quot;INSERT INTO table7(dr_name , specialisation, organisation,
dr_id,boolean_val )
VALUES (?,?,?,?,?)&amp;quot;;
s1 = gen_id();
s2 = generate_name();
s3 = generate_specialisation();
s4 = generate_organisation();
if(s4!=&amp;quot;NA&amp;quot;)
s5=&amp;quot;YES&amp;quot;;
else
s5=&amp;quot;NO&amp;quot;;
PreparedStatement preparedStatement = session.prepare(strCQL);
BoundStatement boundStatement = preparedStatement.bind(s2,s3,s4,s1,s5);
session.execute(boundStatement);

}

}
```

6.7.2 Output

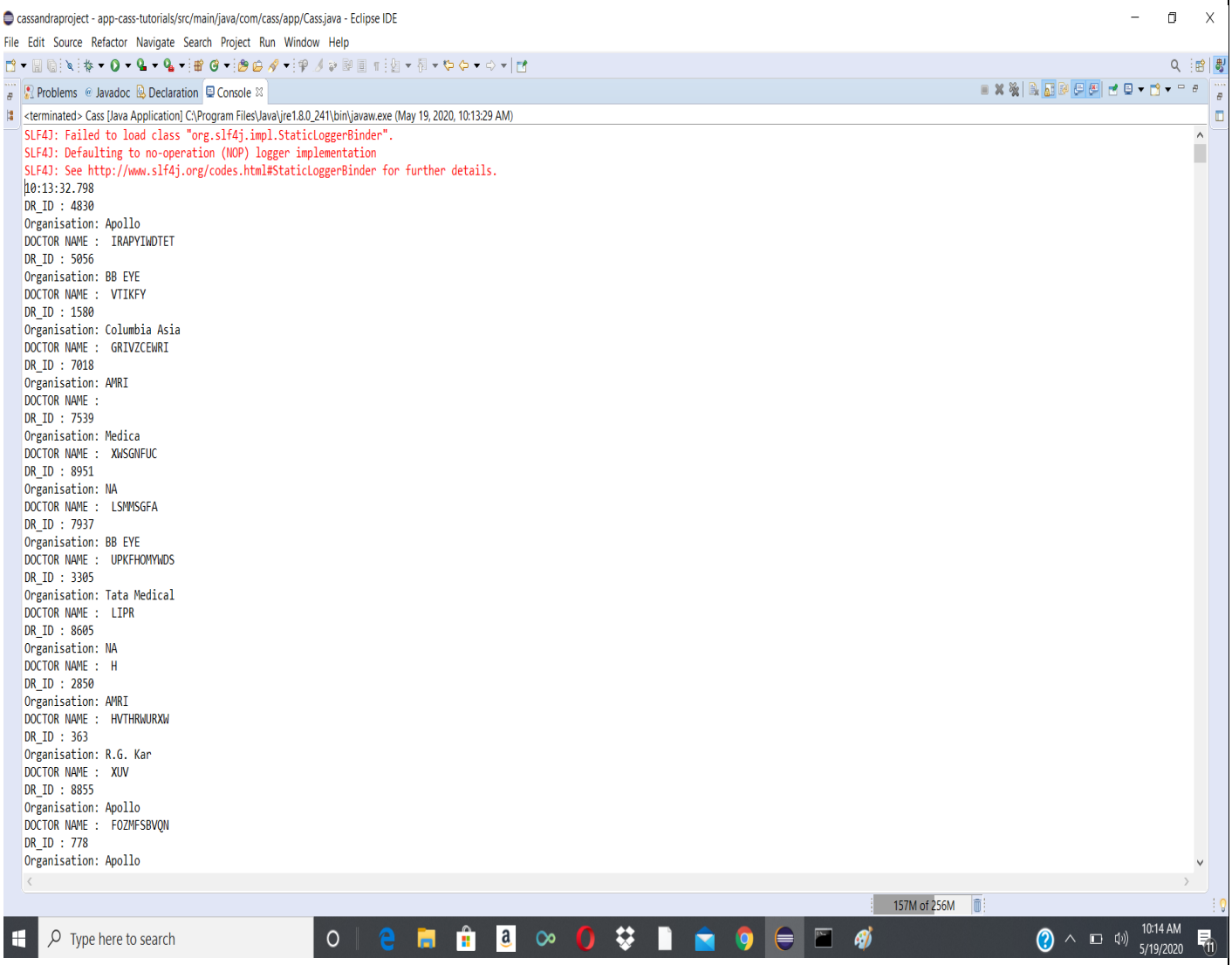
```
static void output7() {

String dr_name =null,specialisation=null,organisation=null;
int dr_id;

LocalTime myObj = LocalTime.now();
System.out.println(&quot;Local Time Before Execution : &quot;+myObj);
//Find all the doctors names who are attached with an organization.
com.datastax.driver.core.ResultSet resultset1 = session.execute(&quot;select
dr_id,dr_name,specialisation,organisation from cas.table7 where
boolean_val=&#39;YES&#39; ALLOW FILTERING
&quot;);

System.out.println(&quot;OUTPUT FOR QUERY 7&quot;);
for(Row row:resultset1)
{
dr_id = row.getInt(&quot;dr_id&quot;);

specialisation = row.getString(&quot;specialisation&quot;);
organisation = row.getString(&quot;organisation&quot;);
dr_name = row.getString(&quot;dr_name&quot;);
System.out.println(&quot;DOCTOR ID : &quot;+dr_id);
System.out.println(&quot;DOCTOR Name : &quot;+dr_name);

System.out.println(&quot;Specialisation : &quot;+specialisation);
System.out.println(&quot;Organisation : &quot;+organisation);

}
LocalTime myObj2 = LocalTime.now();
System.out.println(&quot;Local Time Before Execution : &quot;+myObj);
System.out.println(&quot;Local Time After Execution : &quot;+ myObj2);
long duration = Duration.between(myObj, myObj2).toMillis();
System.out.println(&quot;Duration of execution : &quot;+duration +&quot;ms&quot;);

}
```

Chapter – VII

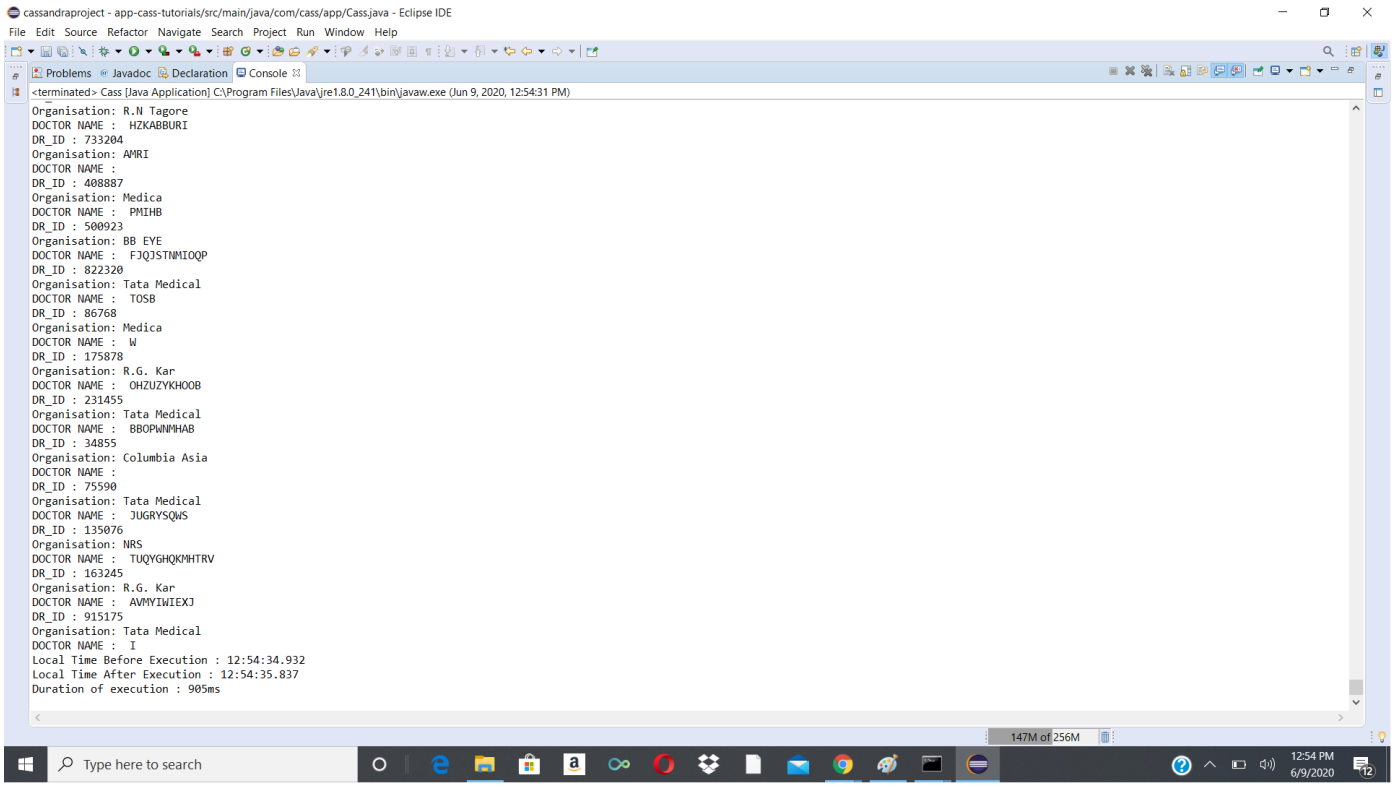
OUTPUT SCREENS

7.1 QUERY 1 : : FIND ALL DOCTORS NAMES AND THEIR DETAILS WITH SPECIALIZATION STRING CARDIOLOGIST.

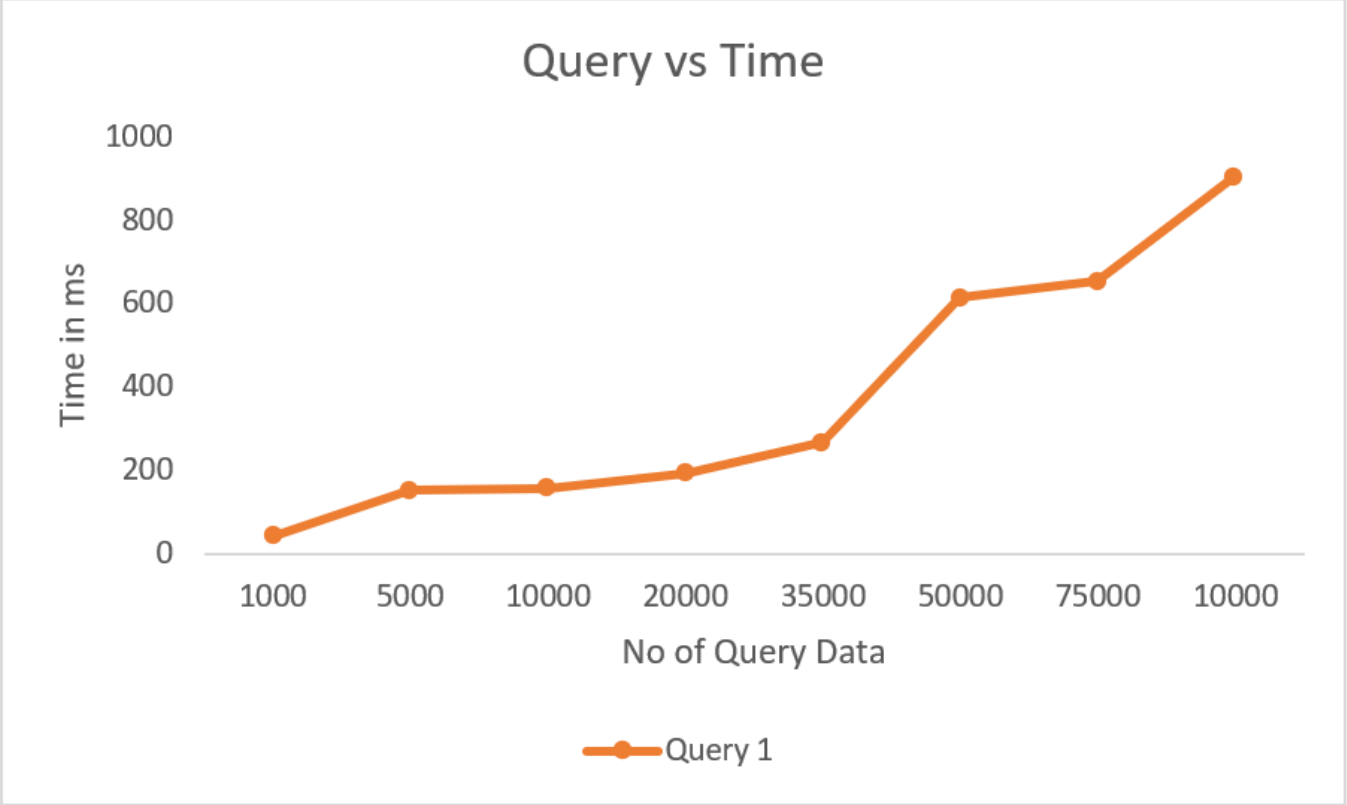
7.1.1 Ouput in Console



7.1.2 Ouput showing Time of Execution

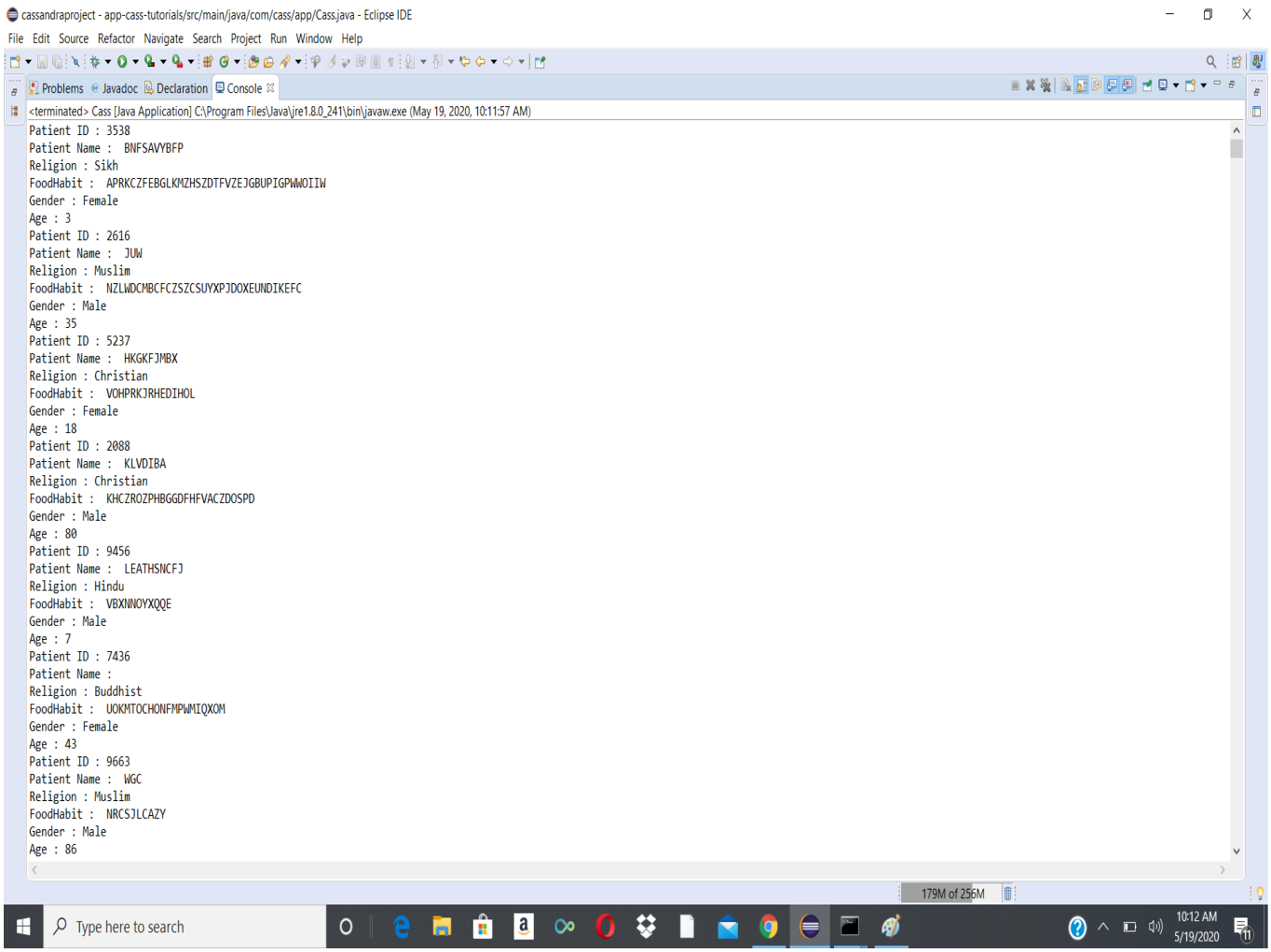


7.1.3 Graph

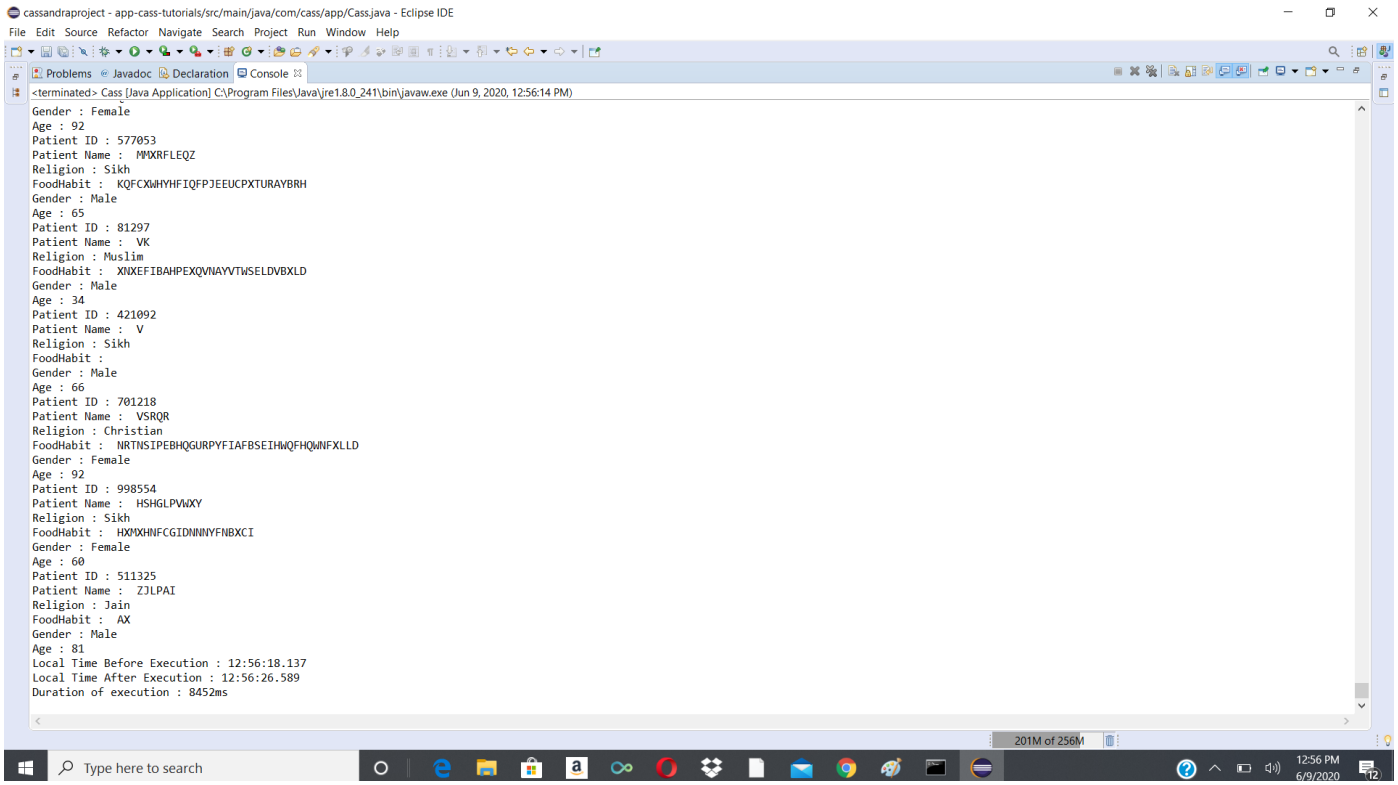

Query vs Time
1000
800
600
400
200
0
Time in ms
1000
5000
10000
20000
35000
50000
75000
10000
No of Query Data
Query 1

7.2 QUERY 2 : FIND ALL INFORMATION OF PATIENTS LIKE DEMOGRAPHIC INFO, RELIGION, FOOD HABIT ETC, FOR A PATIENT.

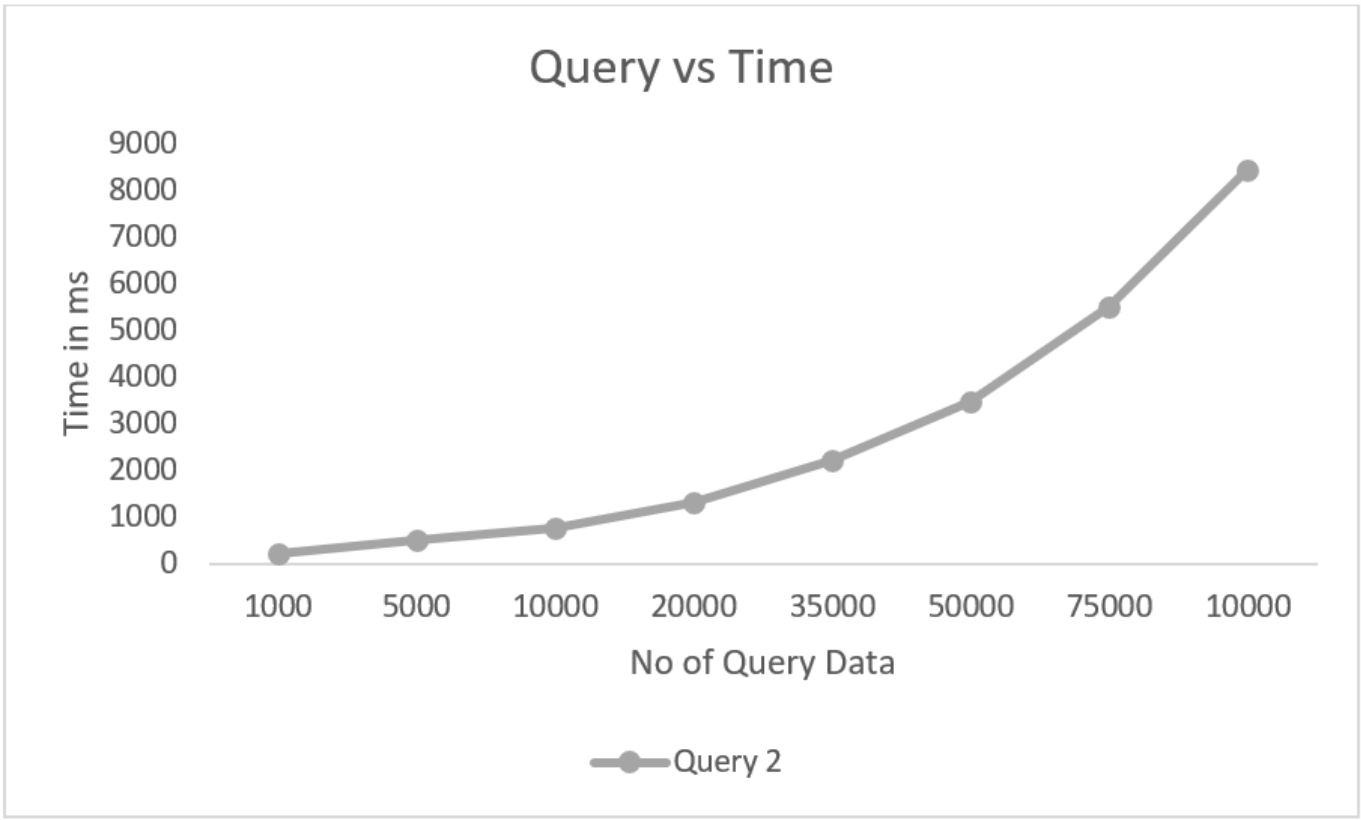
7.2.1 Output in Console



7.2.2 Output Showing Execution Time

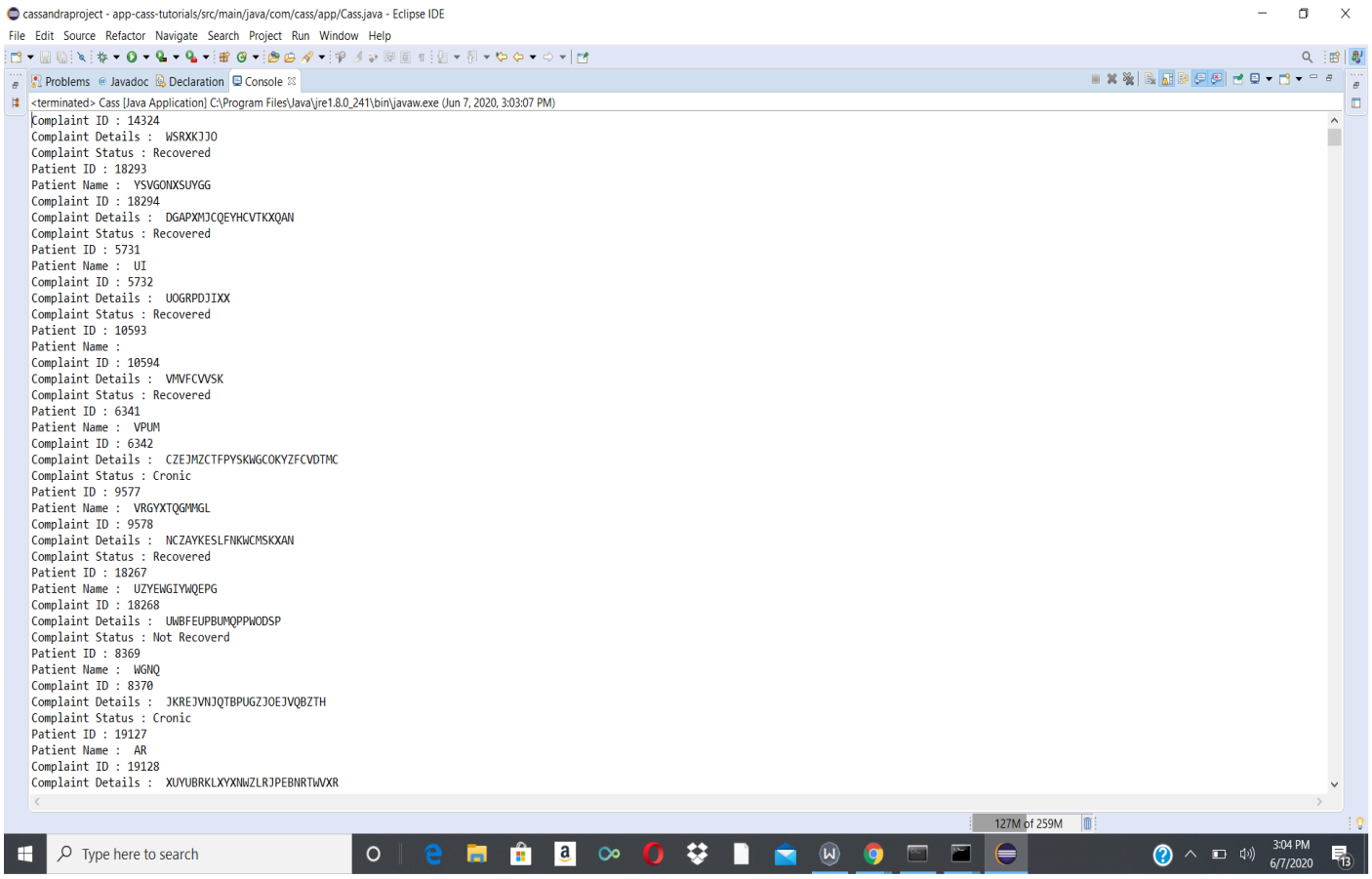


7.2.3 Graph

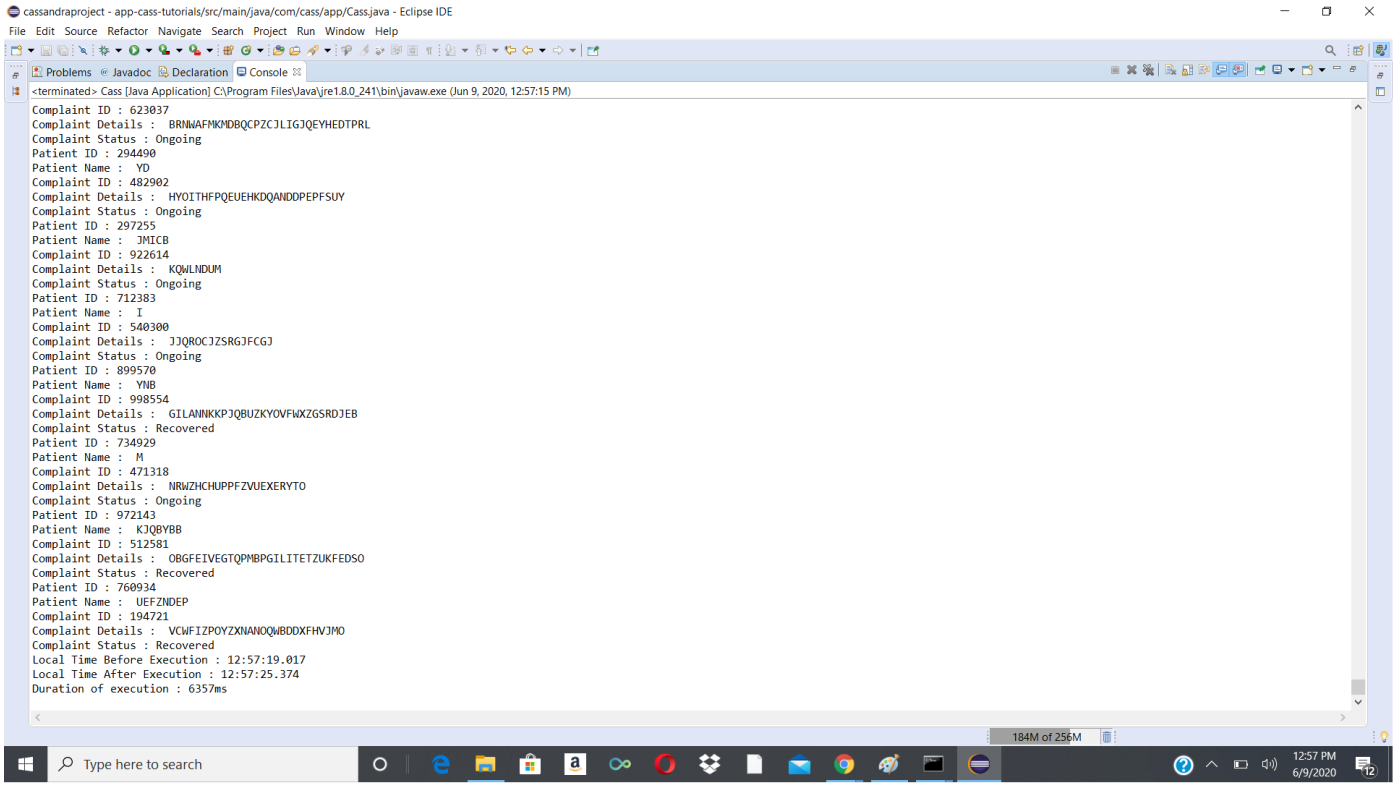

Query vs Time
Time in ms
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1000
5000
10000
20000
35000
50000
75000
10000
No of Query Data
Query 2

7.3 QUERY 3 : FIND ALL THE COMPLAINT DETAILS OF A PATIENT.

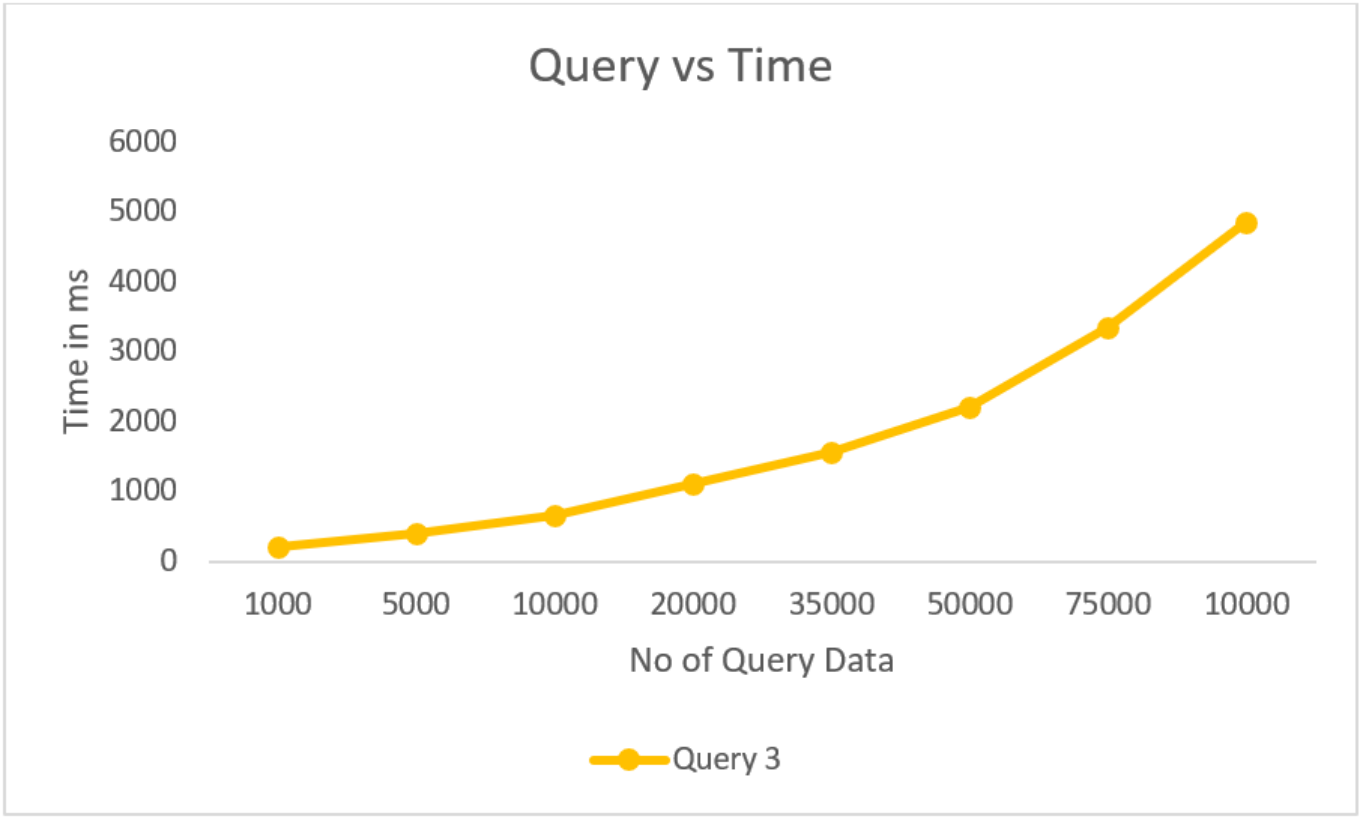
7.3.1 Output in Console



7.3.2 Output Showing Execution Time

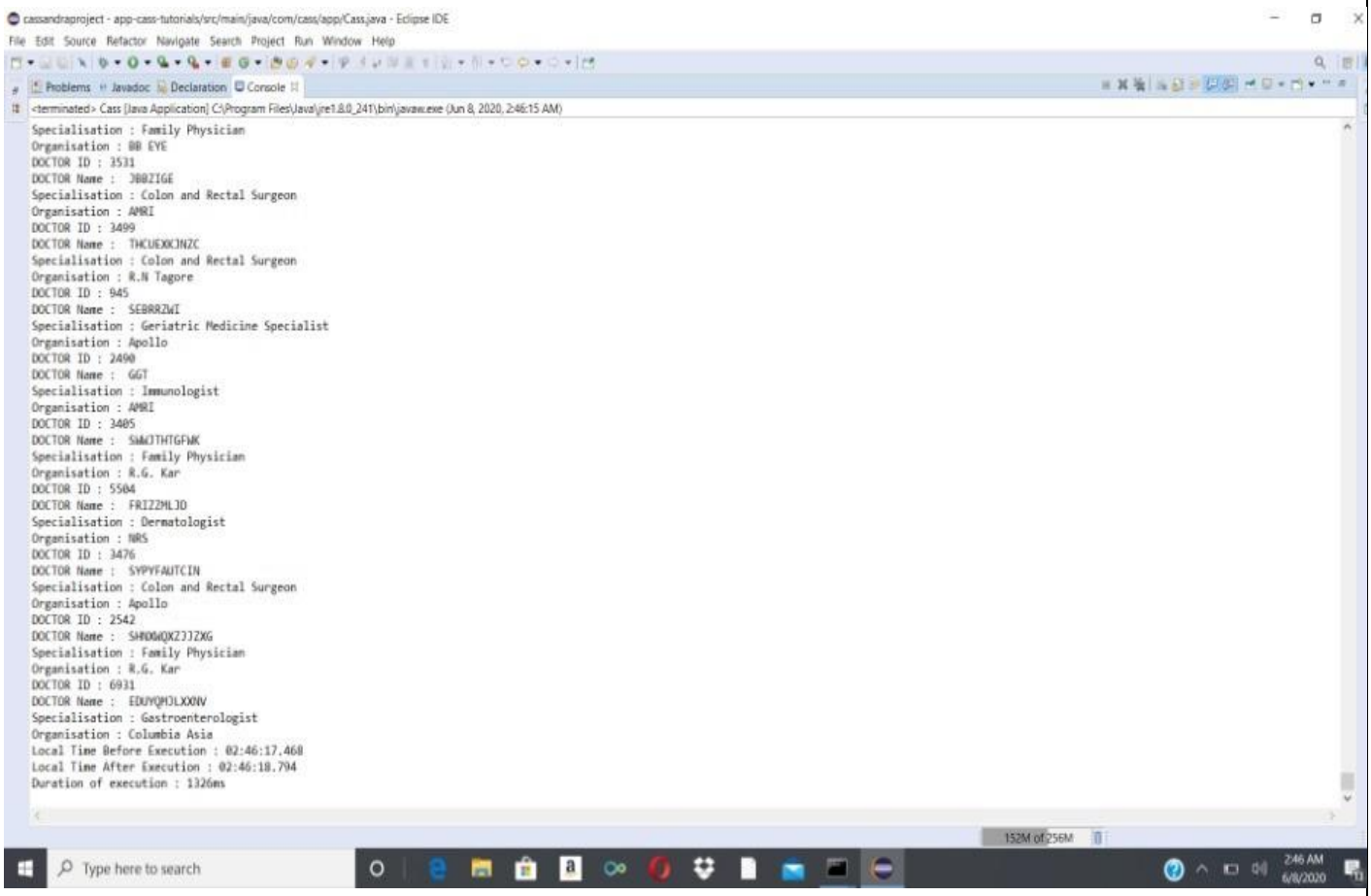


7.3.3 Graph

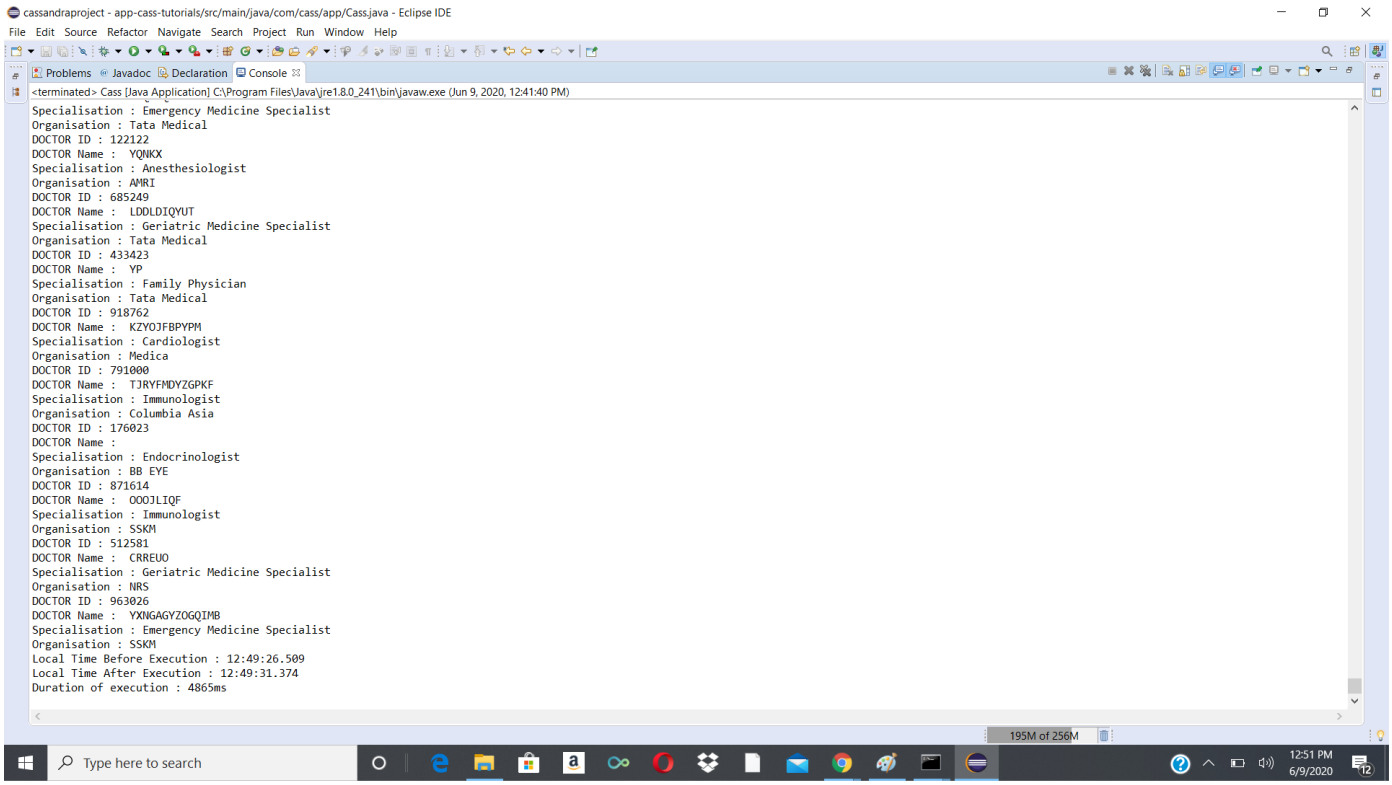

Query vs Time
6000
5000
4000
3000
2000
1000
0
Time in ms
1000
5000
10000
20000
35000
50000
75000
10000
No of Query Data
Query 3

7.4 QUERY 4 : FIND ALL THE DOCTORS NAMES WHO ARE ATTACHED WITH AN ORGANIZATION.

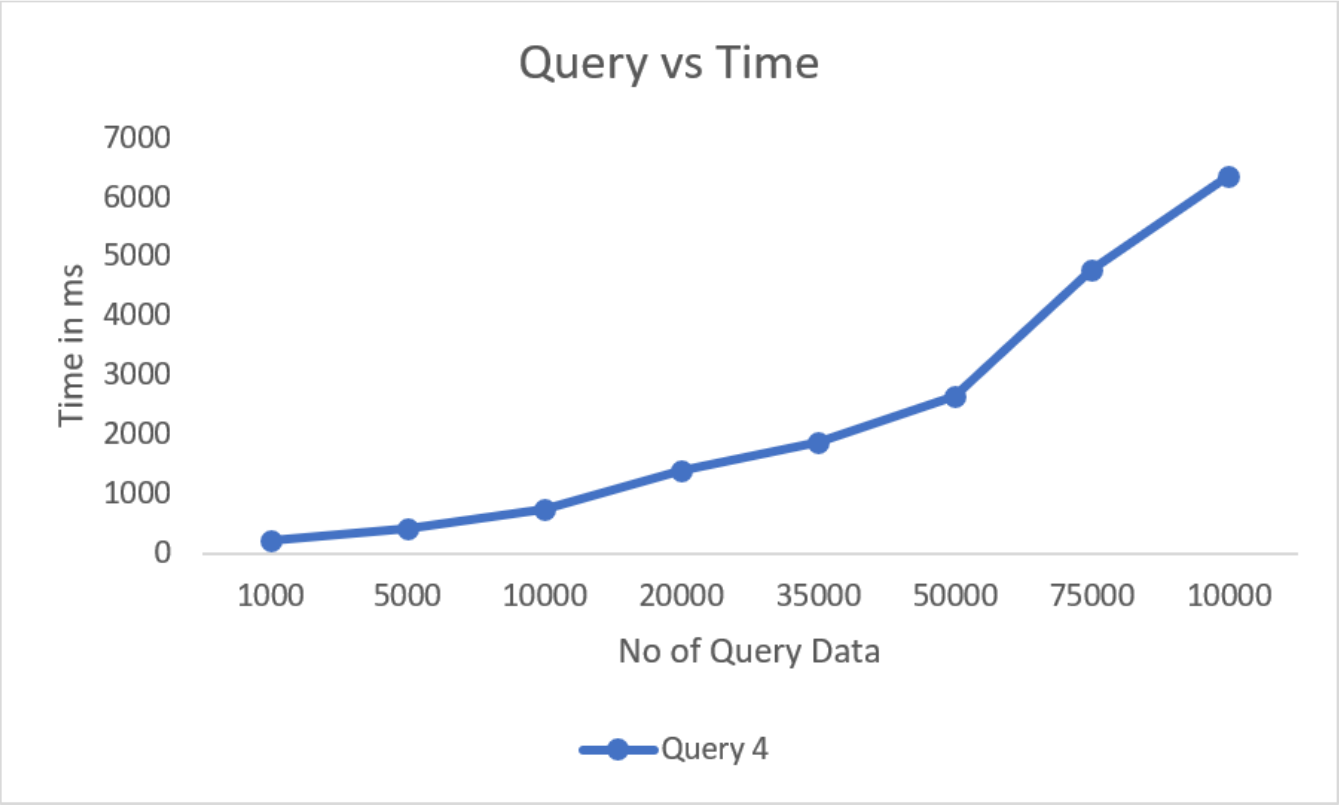
7.4.1 Output in Console



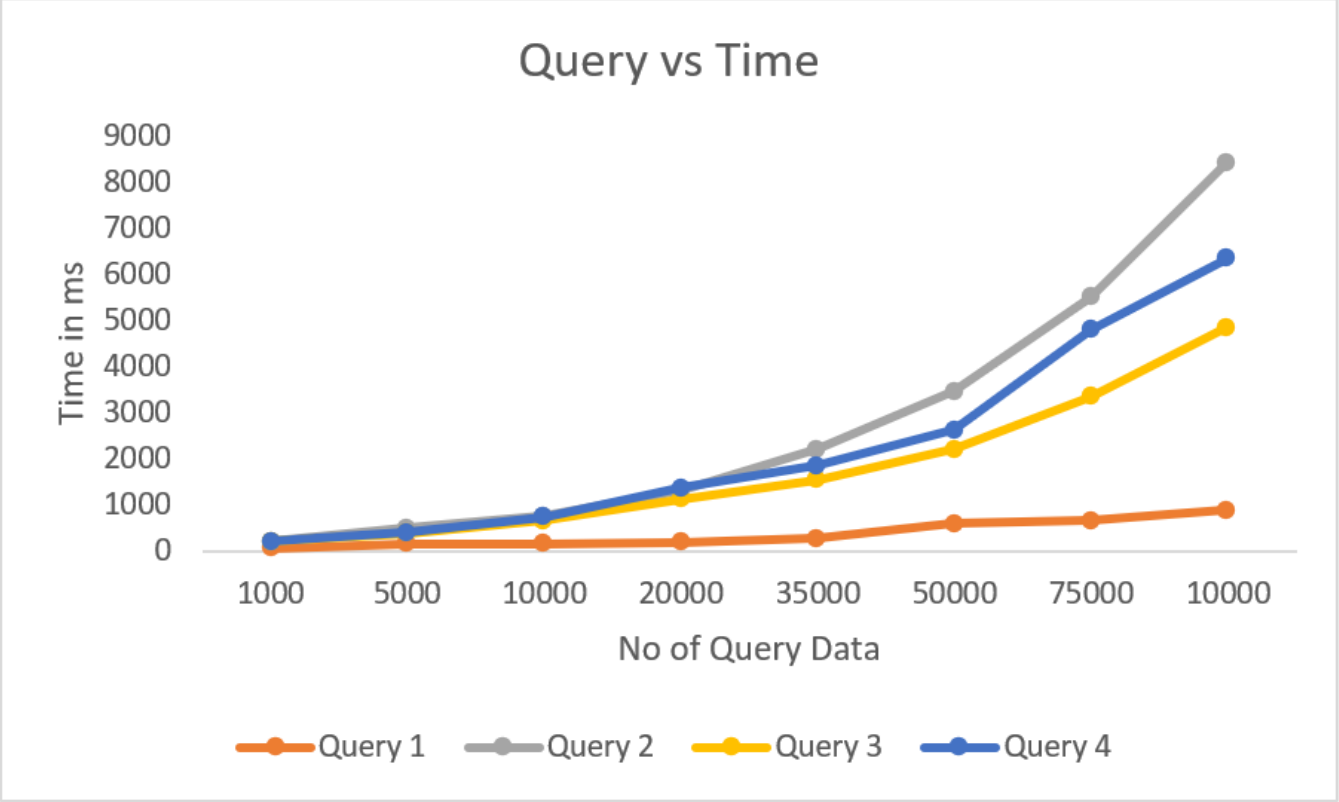
7.4.2 Output Showing Execution Time



7.4.3 Graph


Query vs Time
Time in ms
7000
6000
5000
4000
3000
2000
1000
0
1000
5000
10000
20000
35000
50000
75000
10000
No of Query Data
Query 4

7.5 GRAPH PLOT



Chapter – VIII

CONCLUSIONS

This project aimed for performance testing for a particular set of queries. This is not an application-based project, instead it is more of a research-based project. A person in account coming for the research will be helped by this considering that he or she looks up for a query and will know how much time it requires to run that particular query, that is, its timestamp. The project, since a research-based project, is almost never ending and has chances of expansion at every level. We tried implementing only a small part of it which included executing some queries and calculating the time required for calculation. This has more scope

of expansion.

Chapter – IX

REFERENCES

1) https://healthcare-communications.imedpub.com/the-usefulness-and-challenges-of-big-data-in-healthcare.php?aid=22237

2) http://www.healthtechzone.com/topics/healthcare/articles/2016/11/18/427248-pros-cons-big-data-the-healthcare-industry.htm

3) https://www.forbes.com/sites/bernardmarr/2016/05/24/big-data-a-game-changer-in-healthcare/#1ba824d6525b

4) https://www.expresshealthcare.in/features/big-data-analytics-and-indian-healthcare/162330/

5) https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-Big-Data.html#:~:text=Drawbacks%20or%20disadvantages%20of%20Big%20Data&text=%E2%9E%A8Lots%20of%20big%20data,It%20may%20increase%20social%20stratification.

6) https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0217-0

7) https://www.hindawi.com/journals/bmri/2015/370194/

8) https://www.datastax.com/blog/2015/03/how-do-joins-apache-cassandratm-and-datastax-enterprise

9) https://docs.datastax.com/en/studio/6.8/studio/studioStartStop.html

10) https://mvnrepository.com/artifact/org.apache.cassandra

11)https://docs.datastax.com/en/dse/5.1/cql/cql/cql_reference/cql_commands/cqlCommandsTOC.html

12) https://stackoverflow.com/questions/22600345/how-to-generate-random-code-and-check-whether-it-exist-in-database-or-not

13) https://www.youtube.com/watch?v=kFF5x4OpvW4

14) https://docs.datastax.com/en/dse/6.7/dse-dev/datastax_enterprise/spark/sparkJavaApi.html

15) https://saumitra.me/blog/how-cassandra-stores-data-on-filesystem/