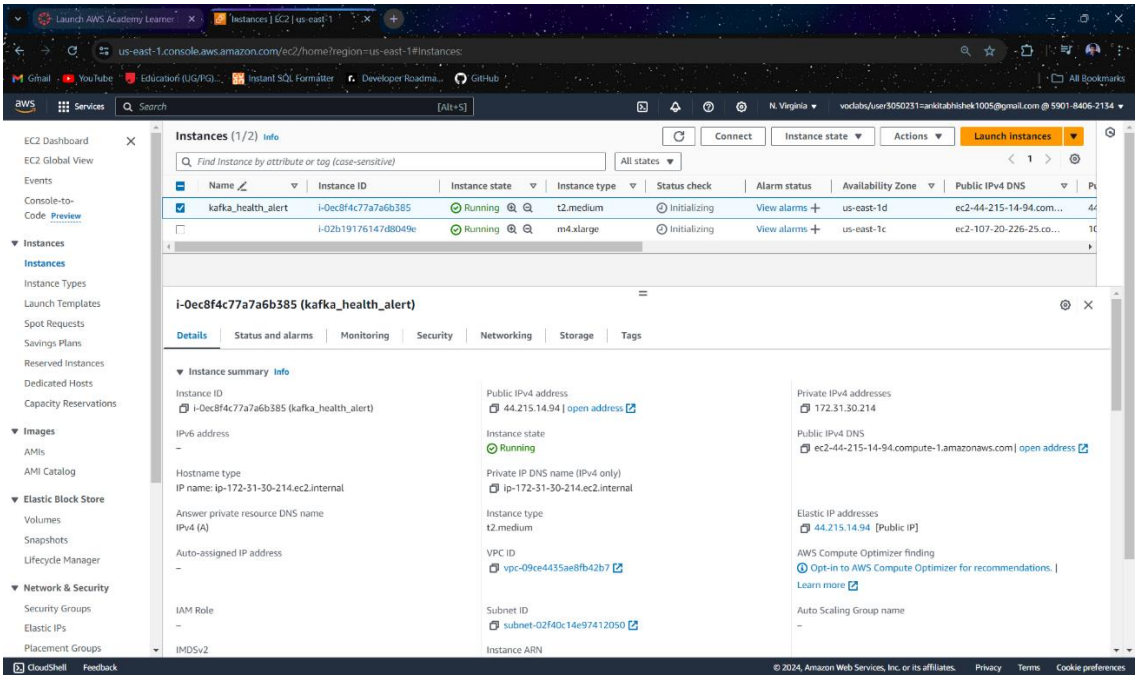
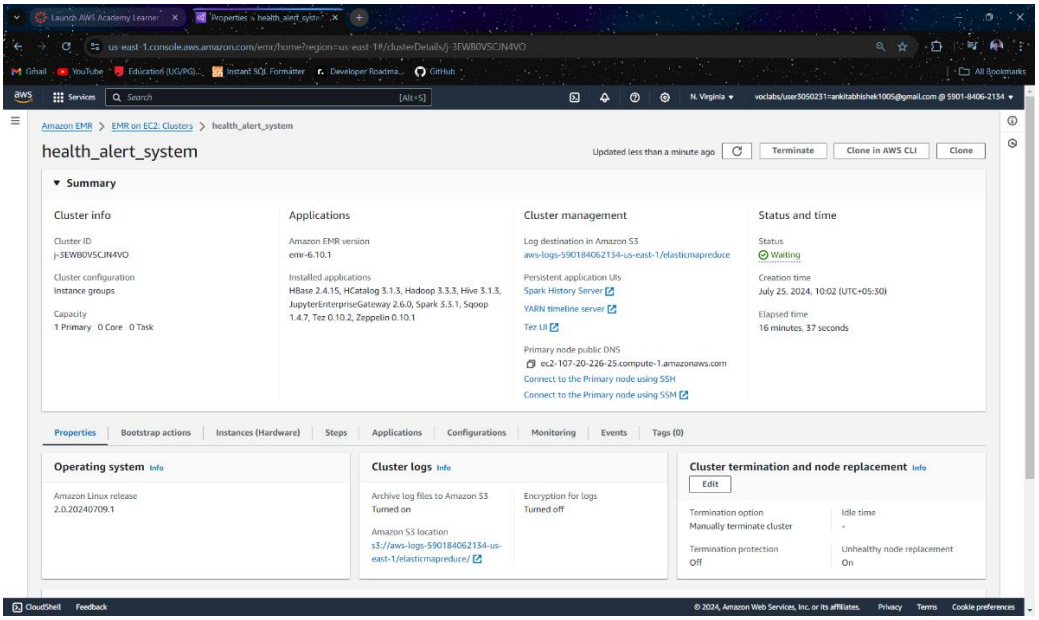
CODE LOGIC

STEP1: CREATION OF KAFKA CLUSTER AND EMR CLUSTER

1. Create a kafka cluster with the help of ec2 instance. (Referred with PDF attached in the Apache Kafka modules). Make the required configuration needed to run kafka instance. Kafka is pre-installed on below mentioned ec2 cluster with the selection of **ami-06c41d8b5a6ddd3c2** while creating **Amazon Machine Image** as pdf within modules.



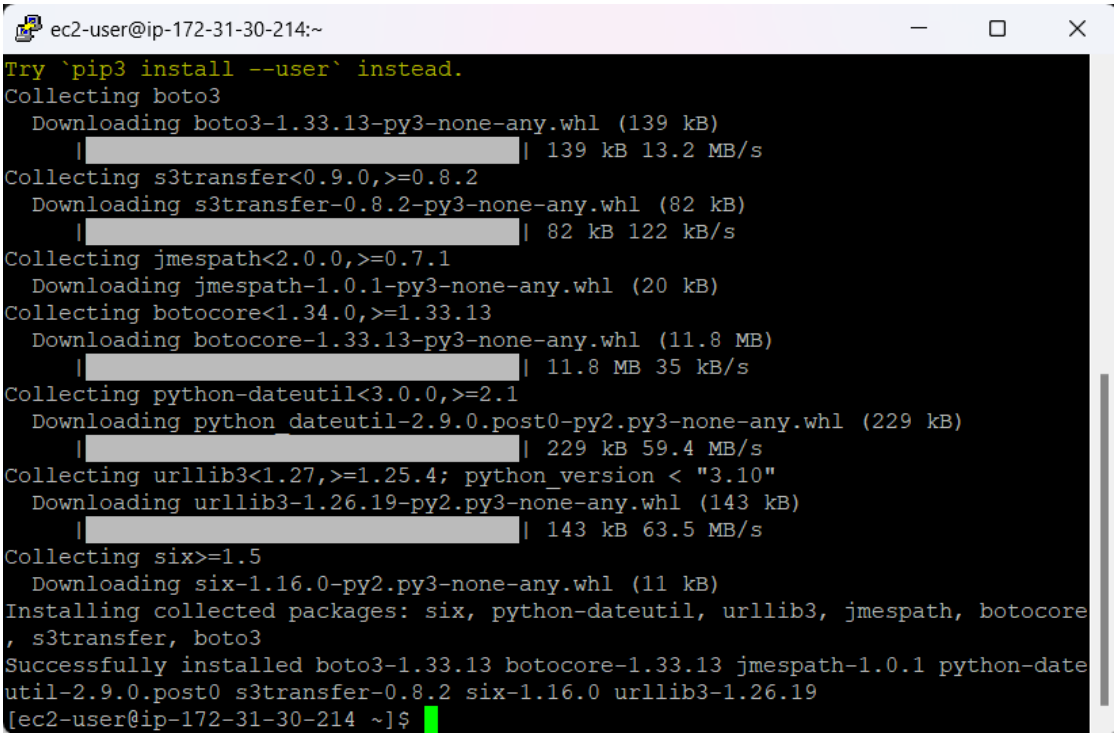
2. Create an EMR instance with required below mentioned libraries (Referred with PDF attached in the modules).
   Spark 3.3.1, Sqoop 1.4.7, HBase 2.4.15, HCatalog 3.1.3, Hadoop 3.3.3, Hive 3.1.3, JupyterEnterpriseGateway 2.6.0, Zeppelin 0.10.1



INSTALLING REQUIRED PACKAGES ON KAFKA CLUSTER

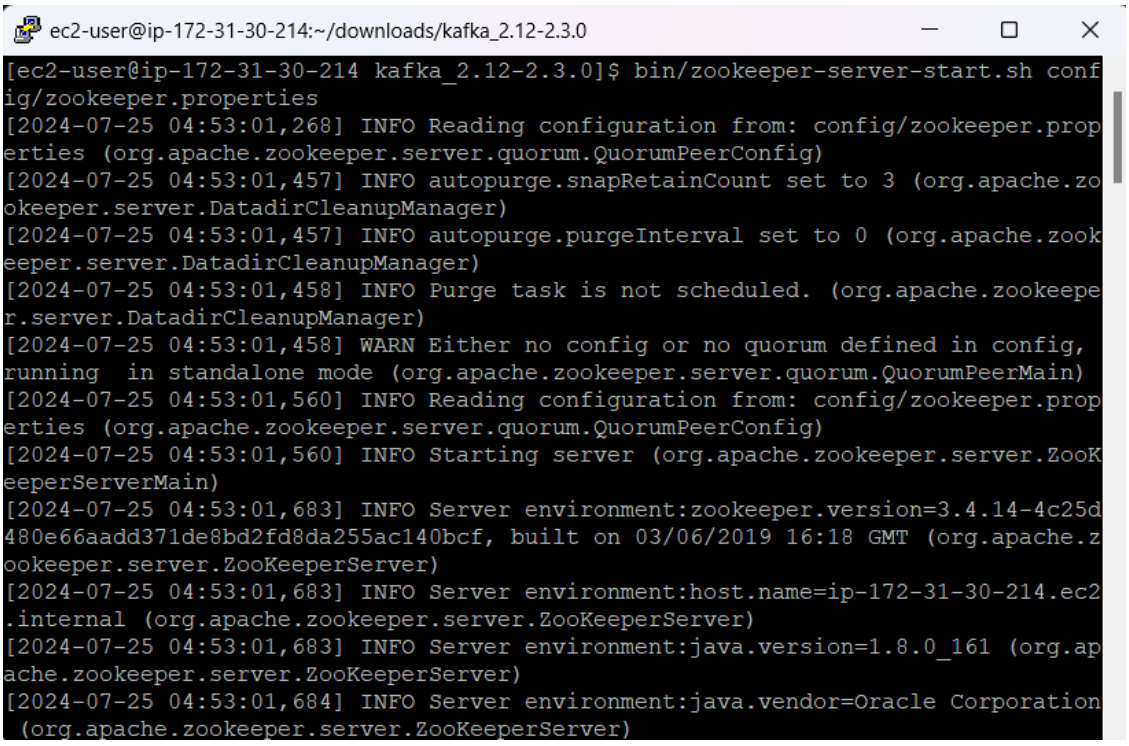
Sudo pip3 install kafka-python
Sudo pip3 install mysql-connector
Sudo pip3 install boto3

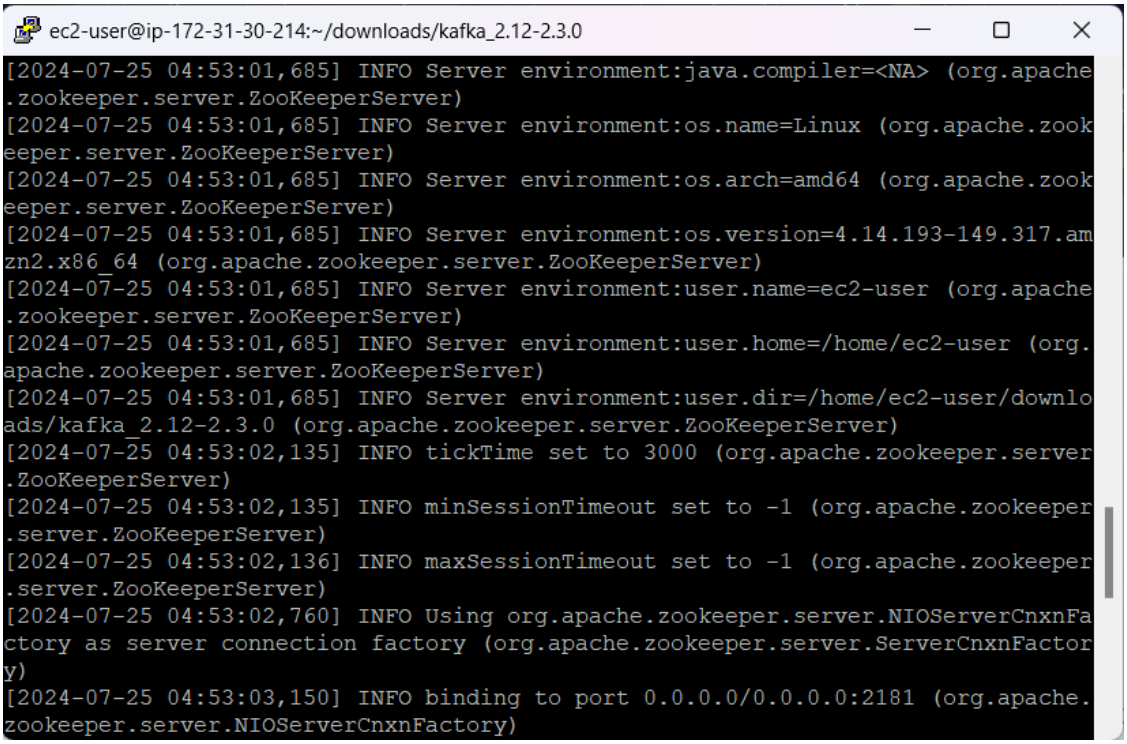
```
ec2-user@ip-172-31-30-214:~
[ec2-user@ip-172-31-30-214 ~]$ sudo pip3 install kafka-python
WARNING: Running pip install with root privileges is generally not a good idea.
Try `pip3 install --user` instead.
Collecting kafka-python
  Downloading kafka_python-2.0.2-py2.py3-none-any.whl (246 kB)
     |████████████████████████████████| 246 kB 36.2 MB/s
Installing collected packages: kafka-python
Successfully installed kafka-python-2.0.2
[ec2-user@ip-172-31-30-214 ~]$ sudo pip3 install mysql-connector
WARNING: Running pip install with root privileges is generally not a good idea.
Try `pip3 install --user` instead.
Collecting mysql-connector
  Downloading mysql-connector-2.2.9.tar.gz (11.9 MB)
     |████████████████████████████████| 11.9 MB 69 kB/s
Using legacy 'setup.py install' for mysql-connector, since package 'wheel' is no
t installed.
Installing collected packages: mysql-connector
    Running setup.py install for mysql-connector ... done
Successfully installed mysql-connector-2.2.9
[ec2-user@ip-172-31-30-214 ~]$ sudo pip3 install boto3
WARNING: Running pip install with root privileges is generally not a good idea.
Try `pip3 install --user` instead.
Collecting boto3
  Downloading boto3-1.33.13-py3-none-any.whl (139 kB)
     |████████████████████████████████| 139 kB 13.2 MB/s
```



STATEMENT FOR STARTING KAFKA SERVER

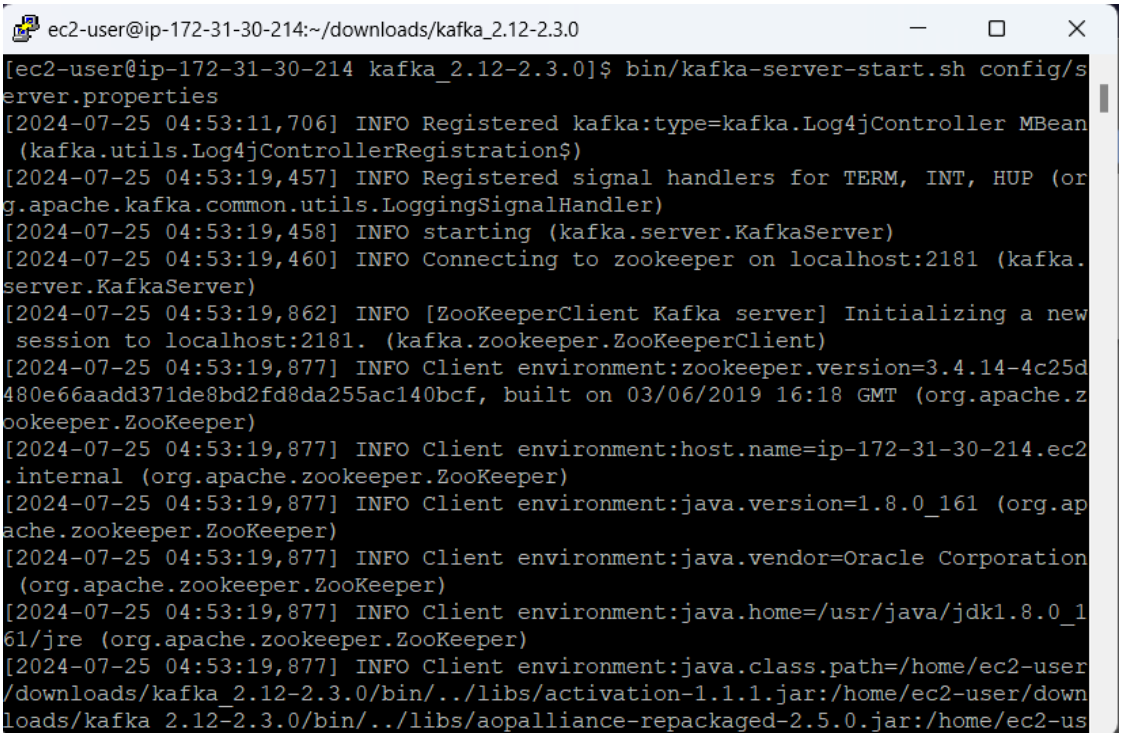
1. **STARTING ZOOKEEPER SERVER**:
   Inside cd downloads/kafka_2.12-2.3.0 run
   bin/zookeeper-server-start.sh config/zookeeper.properties

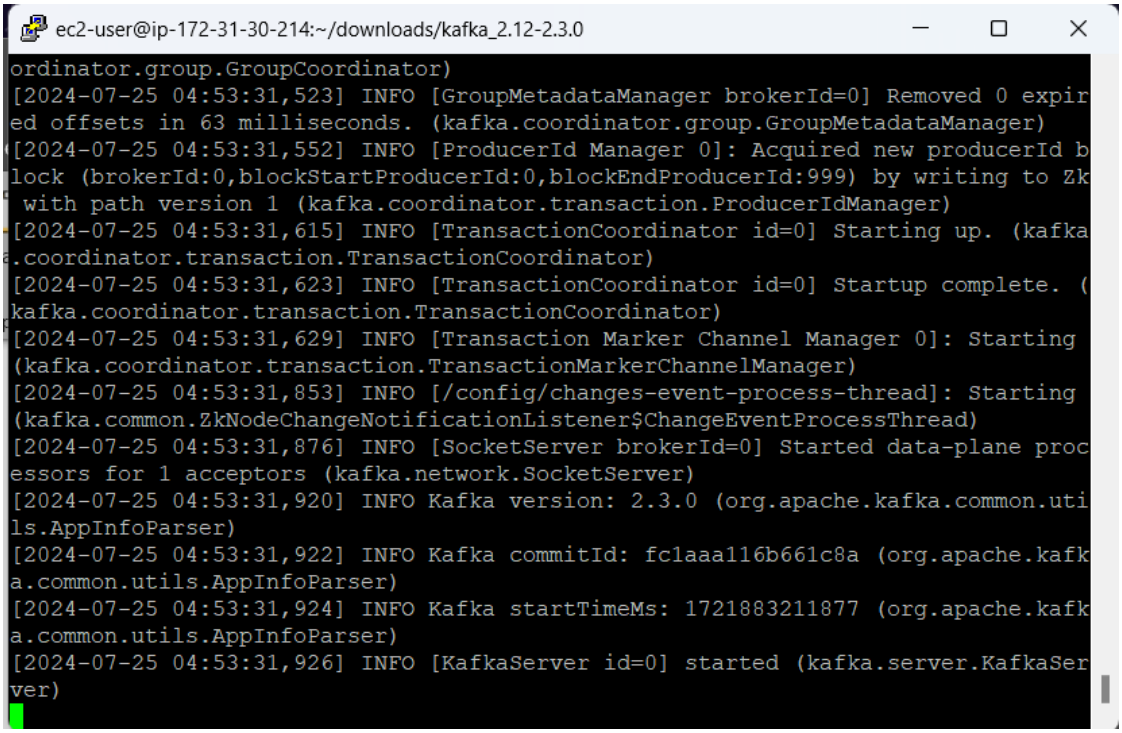




2. **STARTING KAFKA SERVER:**

Into another putty Session of kafka cluster inside cd downloads/kafka_2.12-2.3.0 run **bin/kafka-server-start.sh config/server.properties**





STATEMENT TO CREATE TOPICS

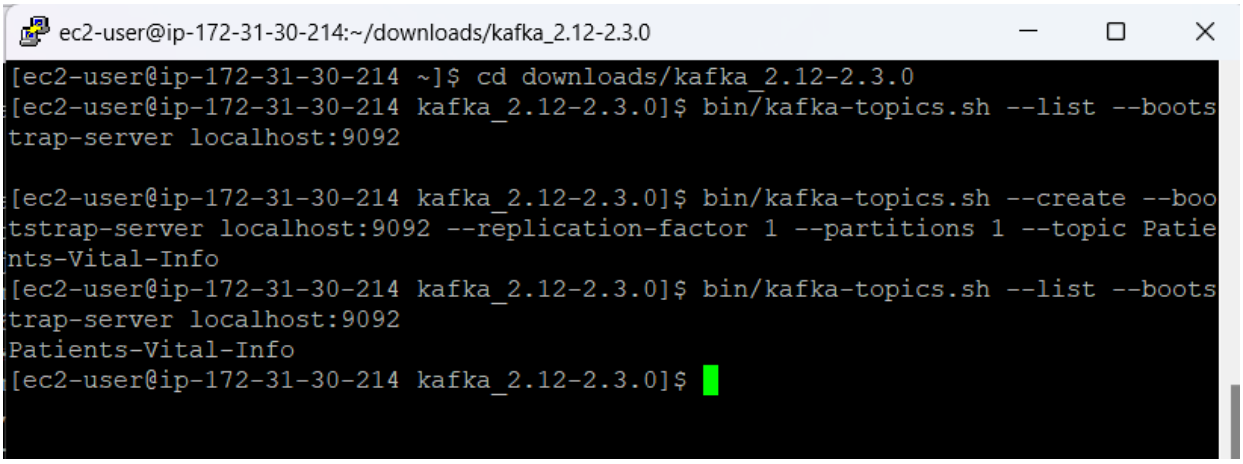
To create topic in kafka server, the command used is

bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic Patients-Vital-Info

STATEMENT TO LIST TOPICS

To list the created topic inside cd downloads/kafka_2.12-2.3.0, the command used is

bin/kafka-topics.sh --list --bootstrap-server localhost:9092



STEP 2: EXECUTING PRODUCER APPLICATION AND CONSUMER APPLICATION:

Producer application which is file named as **kafka_produce_patient_vitals.py** is built on the **python language** which will consume data residing on rds with below mentioned credentials:
Hostname = "upgraddetest.cyaielc9bmnf.us-east-1.rds.amazonaws.com"
username = "student"
password = "STUDENT123"
dbname = "testdatabase".

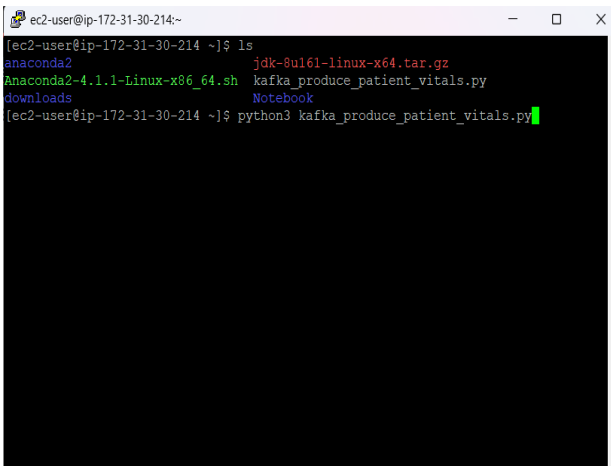
Consumer Application which is file named as **kafka_spark_patient_vitals.py** is built on the **Apache PySpark** language which will consume data being produced with the help of above mentioned producer application

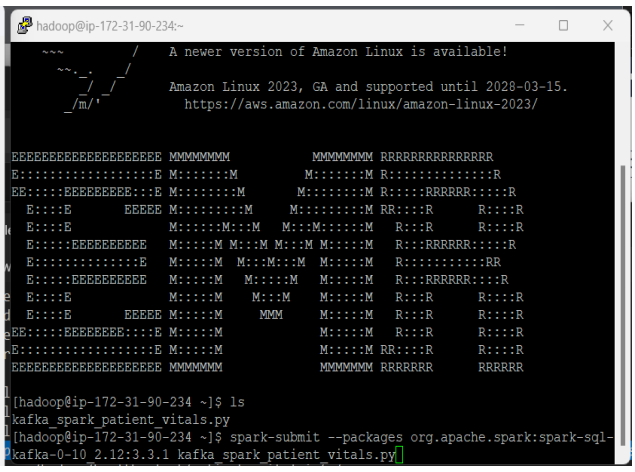
NOTE: Run the producer application on ec2 Kafka cluster after starting the consumer application on EMR cluster created with Spark, Hive and another libraries

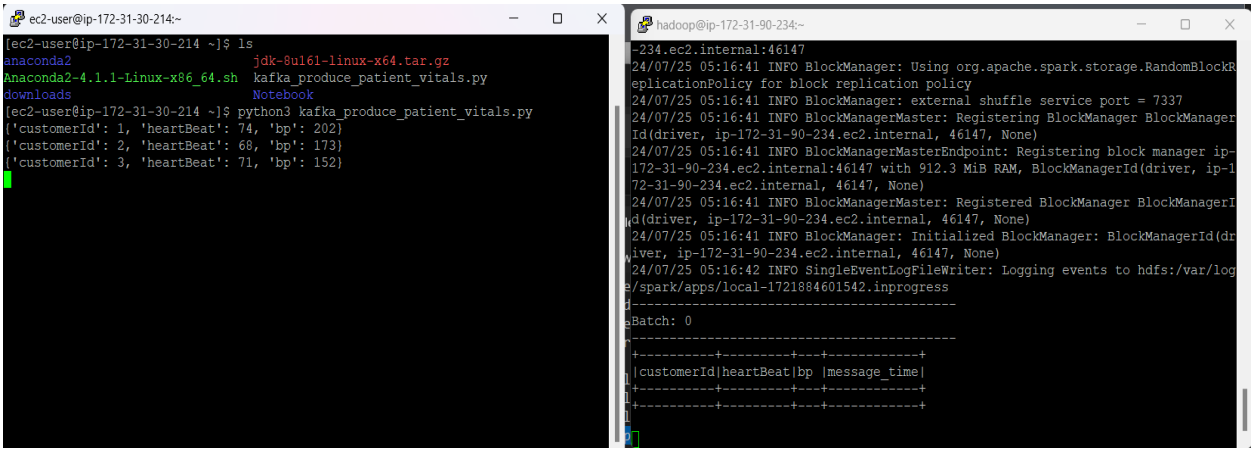
STATEMENT FOR EXECUTING PRODUCER APPLICATION AND CONSUMER APPLICATION

Spark Submitting Job to Consume Message from The Topic Patients-Vital-Info And Stored To HDFS Location

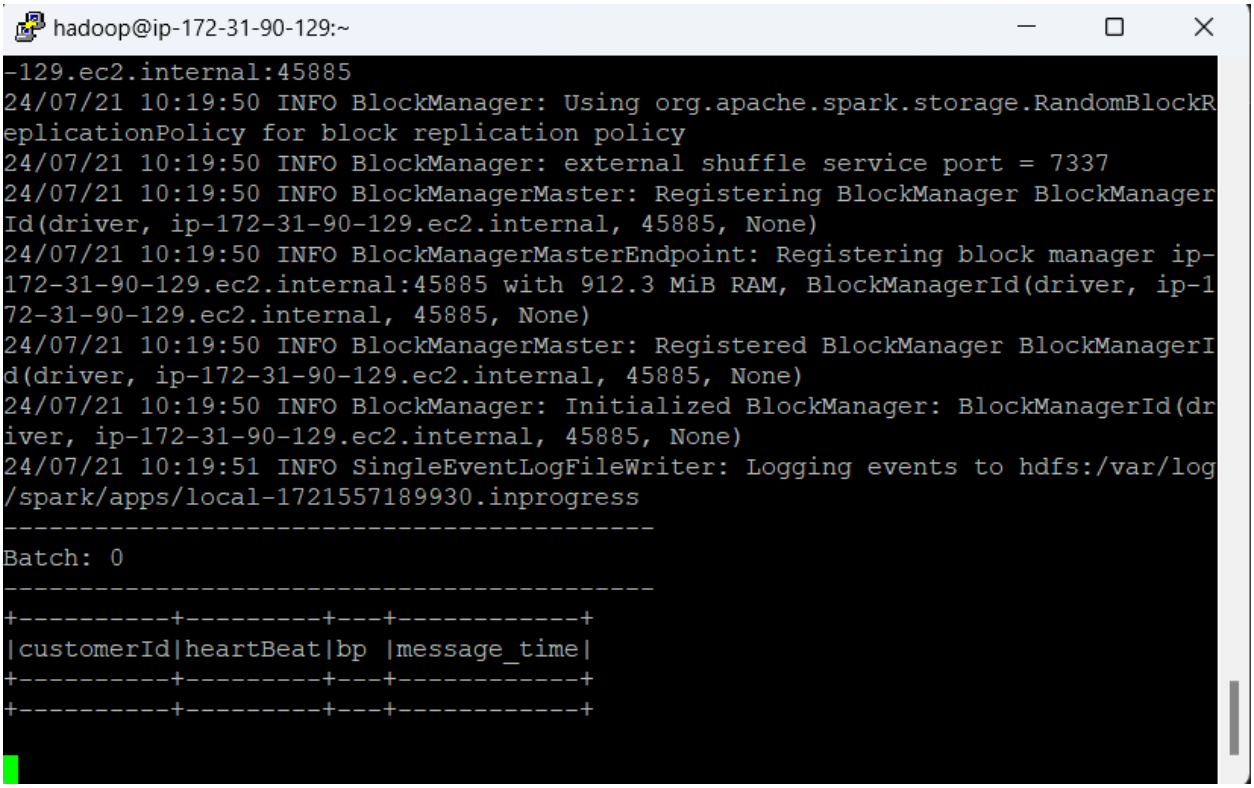
For Producer application: **python3 kafka_produce_patients_vitals.py**



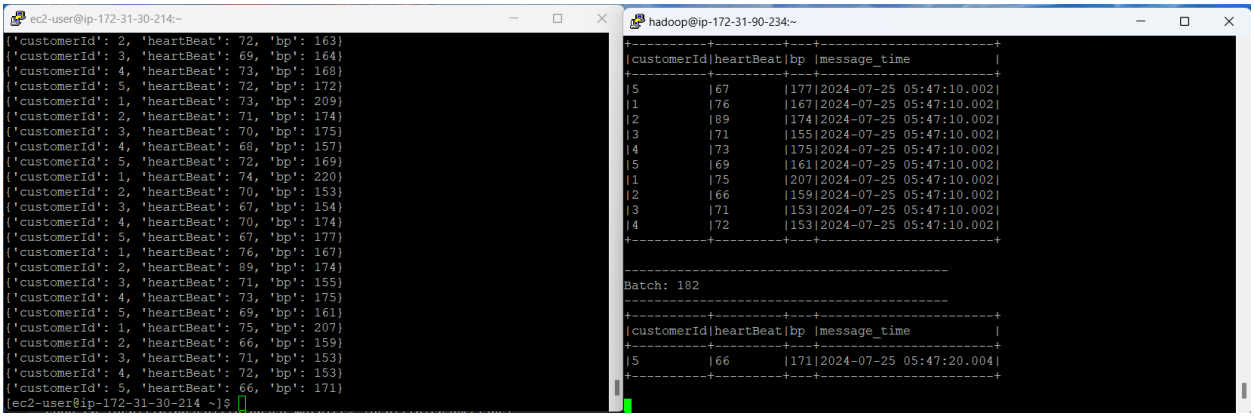




For Consumer Application: **spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.1 kafka_spark_patient_vitals.py**

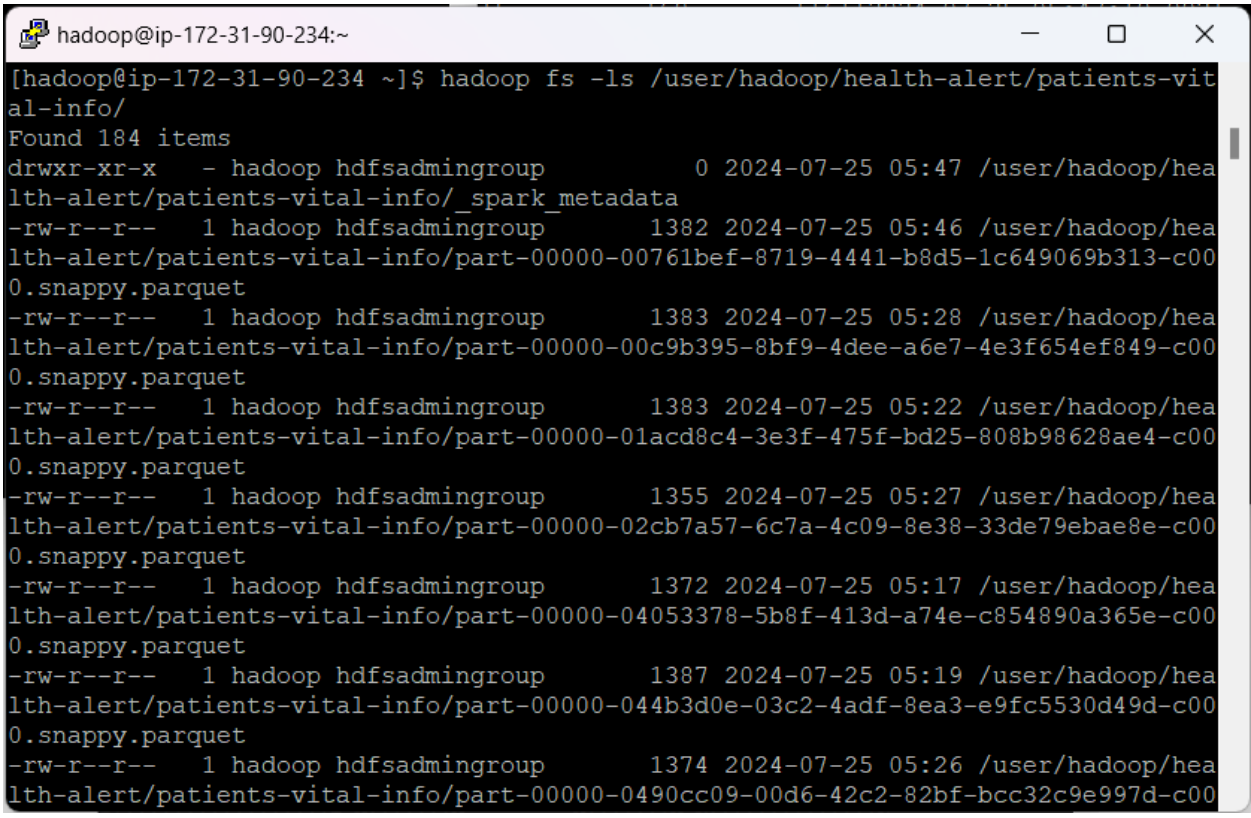


After 30 minutes when all 1800 data being streamed and saved to Parquet file of the required HDFS location



STATEMENT TO CHECK DATA STORED IN HDFS LOCATION:

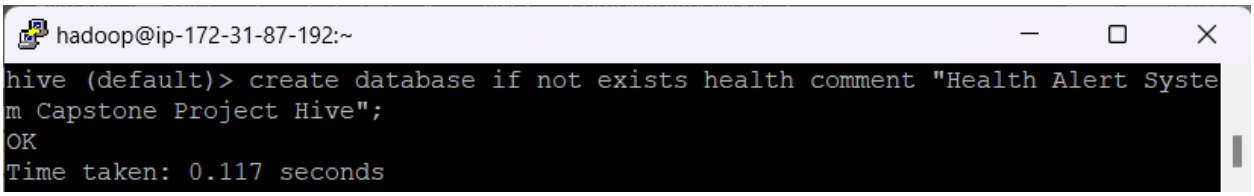
hadoop fs -ls /user/hadoop/health-alert/patients-vital-info/



So till now task 1 and task 2 was accomplished and patients-vital-info was stored in hdfs from rds through kafka.

STEP 3: Build an external hive table on HDFS and view data
STATEMENTS TO CREATE DATABASE FOR PATIENTS' VITAL INFORMATION

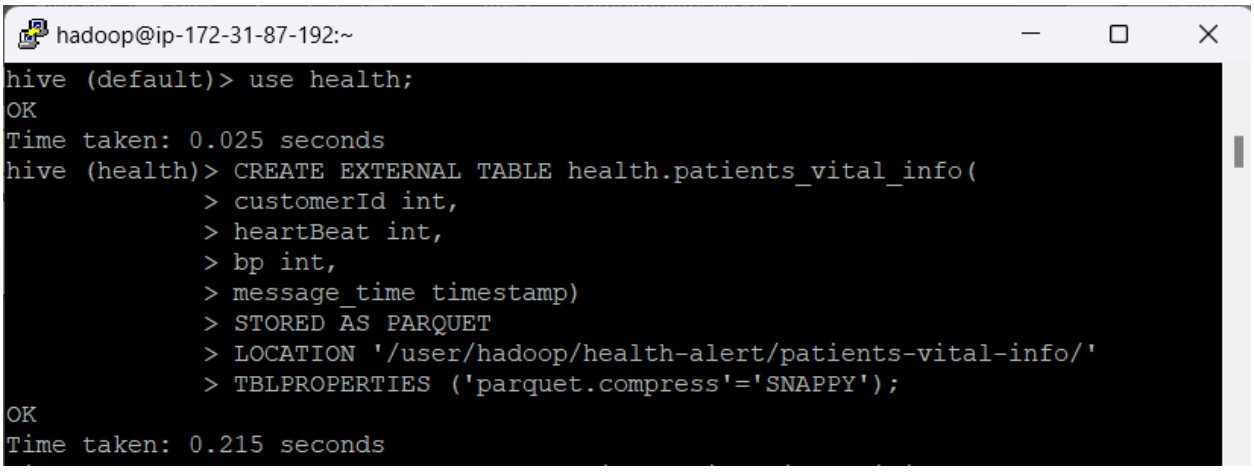
create database if not exists health comment "Health Alert System Capstone Project ` Database";



STATEMENTS TO CREATE EXTERNAL HIVE TABLE FOR PATIENTS' VITAL INFORMATION

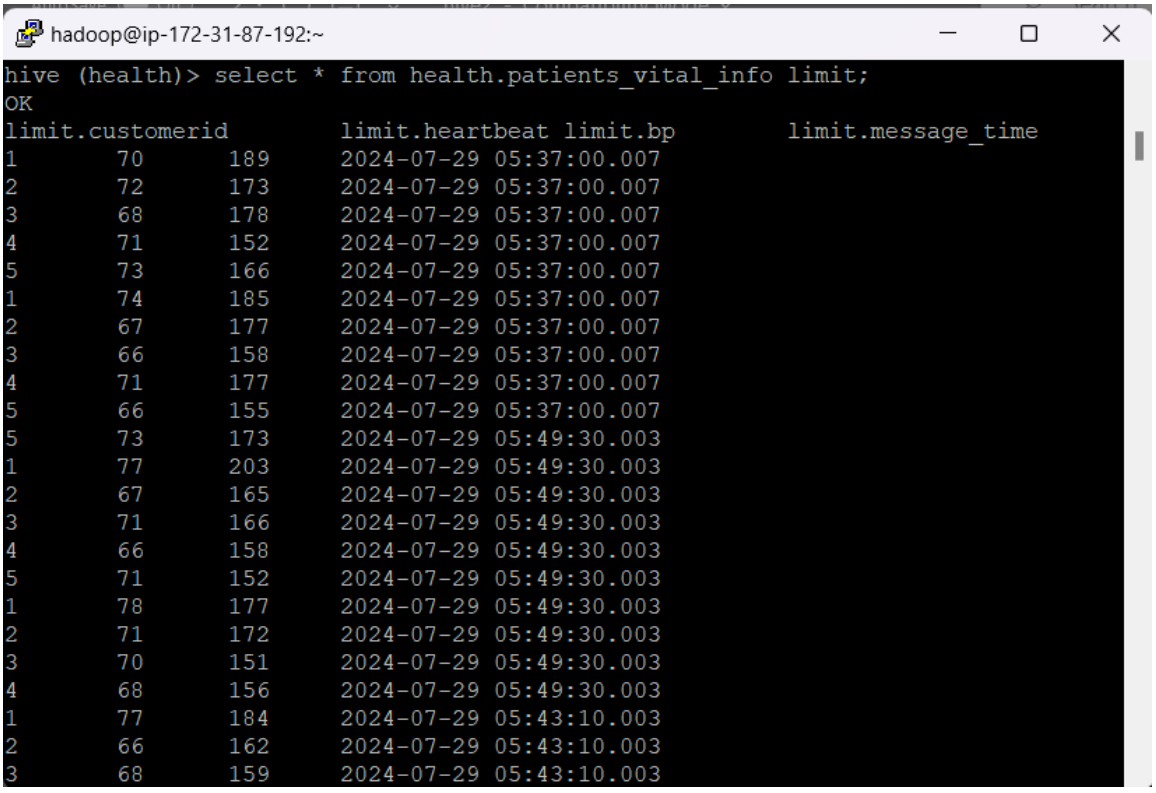
use health;

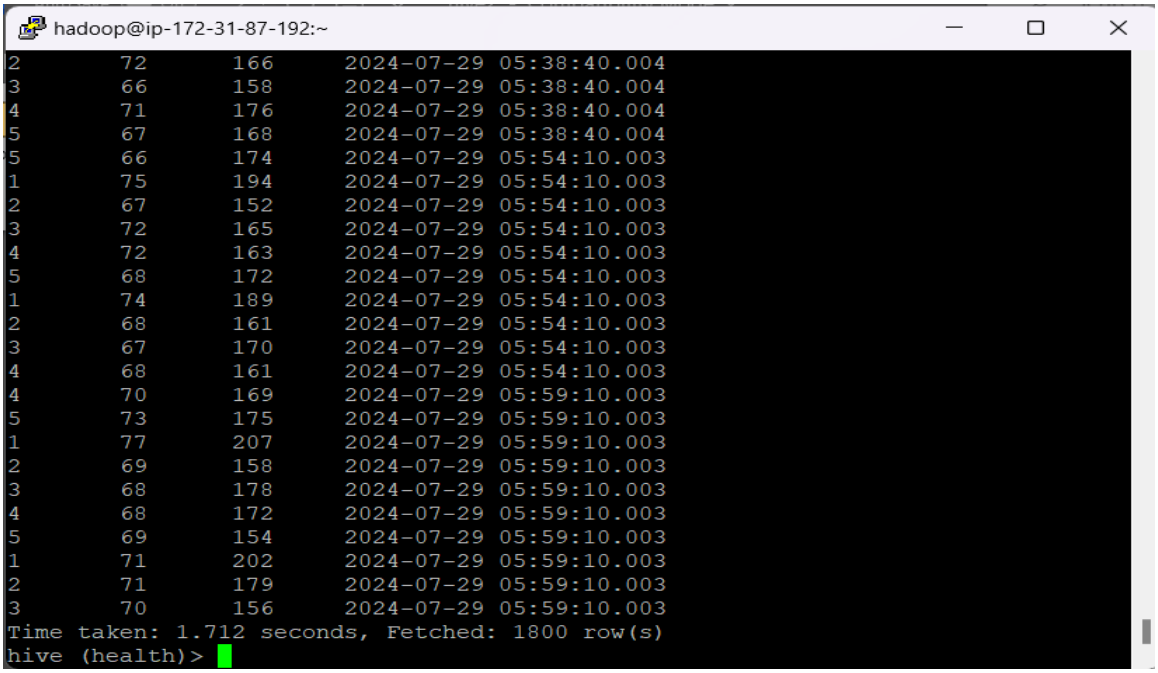
CREATE EXTERNAL TABLE health.patients_vital_info(
customerId int,
heartBeat int,
bp int,
message_time timestamp)
STORED AS PARQUET
LOCATION '/user/hadoop/health-alert/patients-vital-info/'
TBLPROPERTIES ('parquet.compress'='SNAPPY');



STATEMENTS TO VIEW PATIENTS' VITAL INFORMATION HIVE TABLE:

select * from health.patients_vital_info limit;





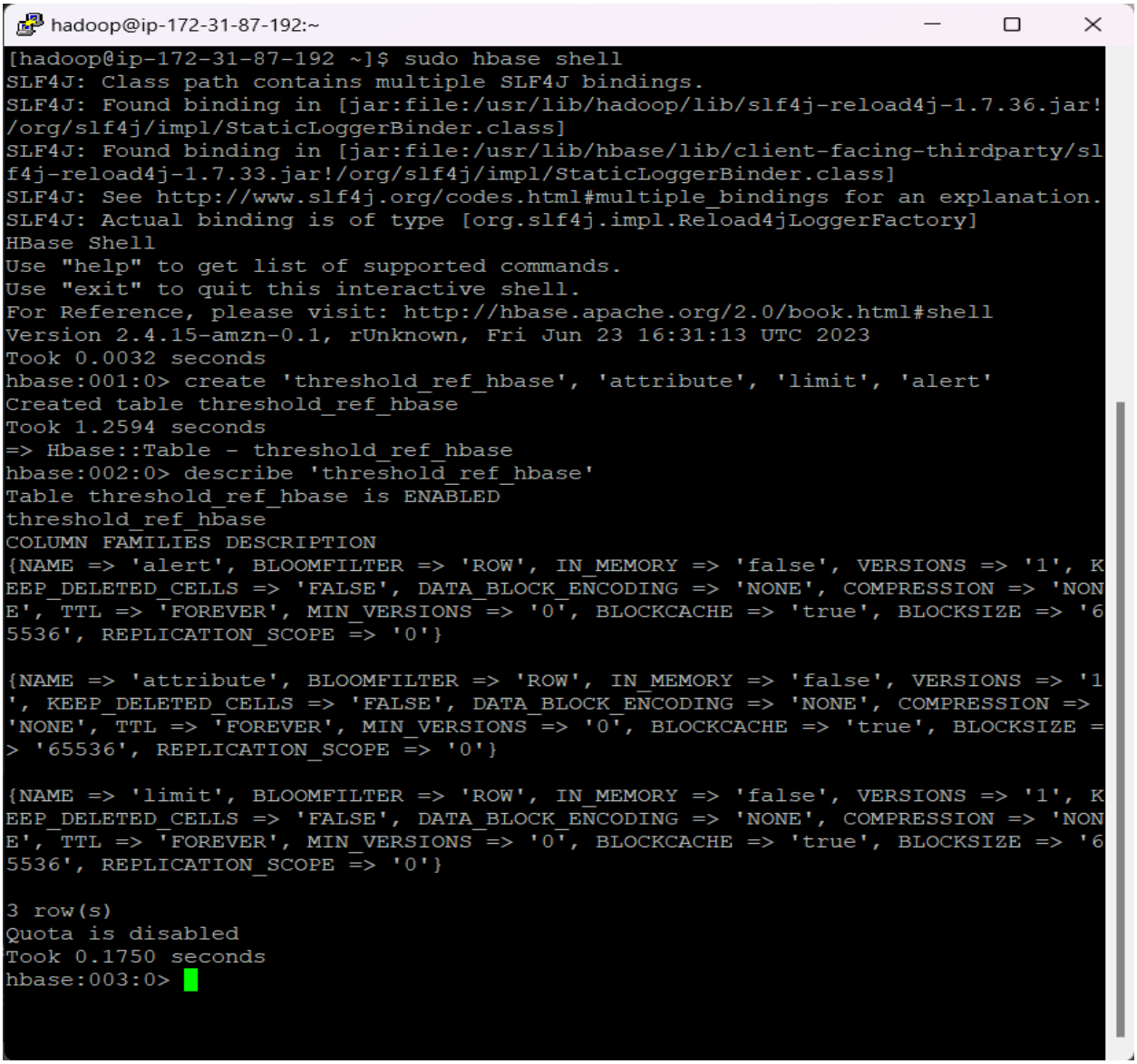
STEP 4: Created hbase table with 3 families (attribute, limit, alert) and inserted 12 records into hbase table:

STATEMENTS TO CREATE HBASE THRESHOLD TABLE

create 'threshold_ref_hbase', 'attribute', 'limit', 'alert'

STATEMENTS TO VIEW HBASE THRESHOLD TABLE ABOVE MENTIONED 3 FAMILIES:

describe 'threshold_ref_hbase'

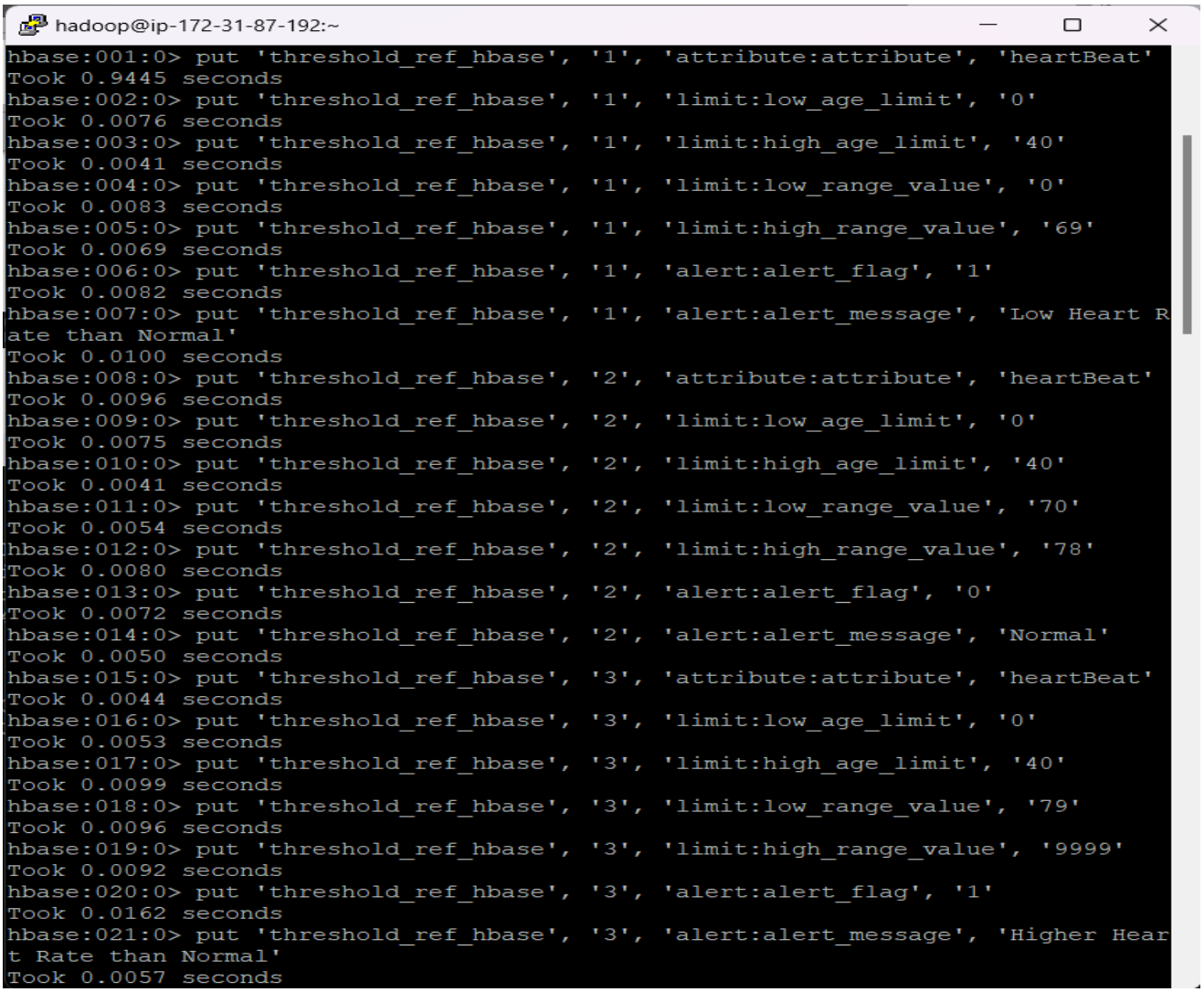


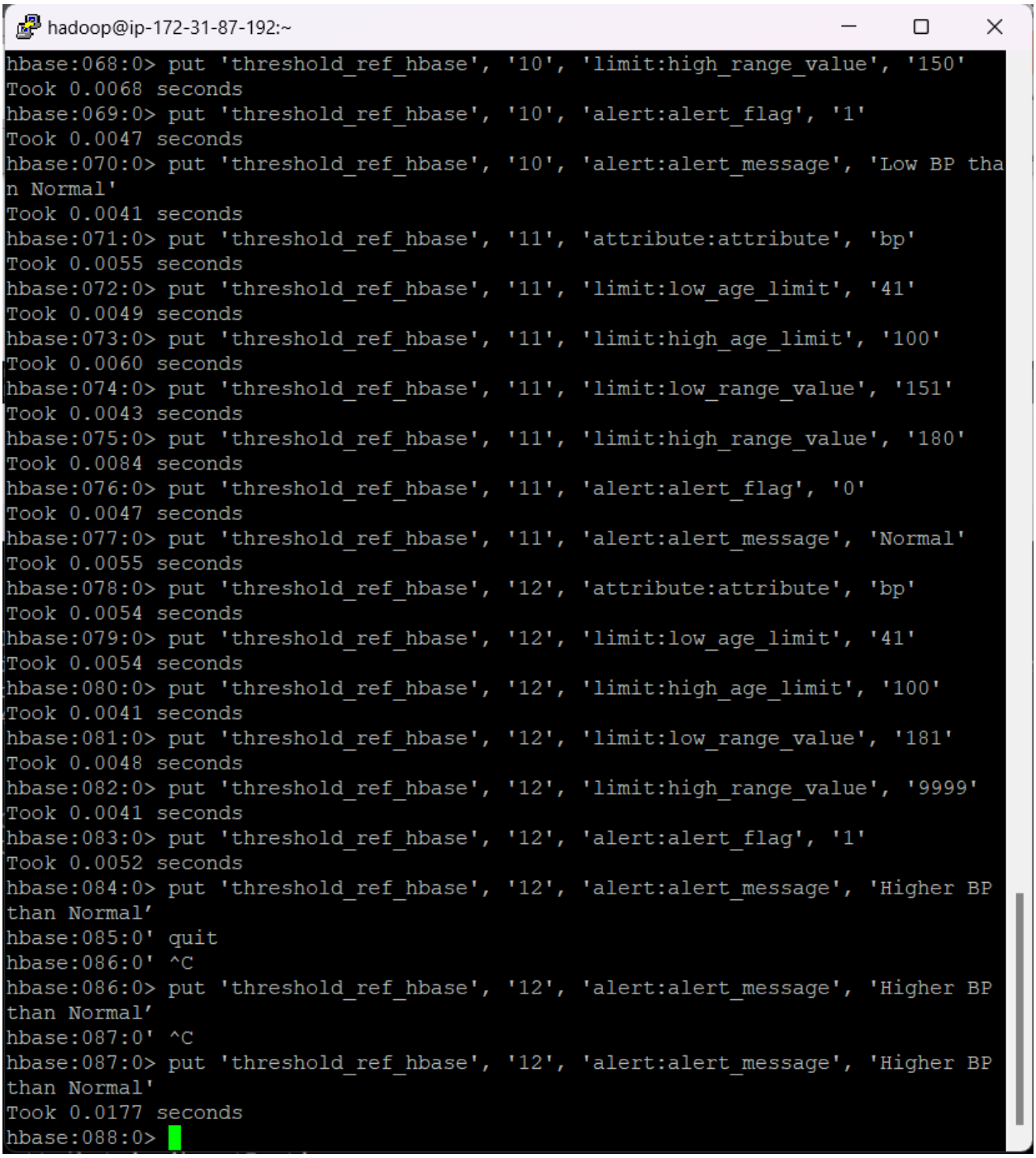
STATEMENTS TO INSERT THRESHOLD PARAMETERS INTO HBASE THRESHOLD TABLE:

```
put 'threshold_ref_hbase', '1', 'attribute:attribute', 'heartBeat'
put 'threshold_ref_hbase', '1', 'limit:low_age_limit', '0'
put 'threshold_ref_hbase', '1', 'limit:high_age_limit', '40'
put 'threshold_ref_hbase', '1', 'limit:low_range_value', '0'
put 'threshold_ref_hbase', '1', 'limit:high_range_value', '69'
put 'threshold_ref_hbase', '1', 'alert:alert_flag', '1'
put 'threshold_ref_hbase', '1', 'alert:alert_message', 'Low Heart Rate than Normal'

put 'threshold_ref_hbase', '2', 'attribute:attribute', 'heartBeat'
put 'threshold_ref_hbase', '2', 'limit:low_age_limit', '0'
put 'threshold_ref_hbase', '2', 'limit:high_age_limit', '40'
put 'threshold_ref_hbase', '2', 'limit:low_range_value', '70'
put 'threshold_ref_hbase', '2', 'limit:high_range_value', '78'
put 'threshold_ref_hbase', '2', 'alert:alert_flag', '0'
put 'threshold_ref_hbase', '2', 'alert:alert_message', 'Normal'

put 'threshold_ref_hbase', '3', 'attribute:attribute', 'heartBeat'
put 'threshold_ref_hbase', '3', 'limit:low_age_limit', '0'
put 'threshold_ref_hbase', '3', 'limit:high_age_limit', '40'
put 'threshold_ref_hbase', '3', 'limit:low_range_value', '79'
put 'threshold_ref_hbase', '3', 'limit:high_range_value', '9999'
put 'threshold_ref_hbase', '3', 'alert:alert_flag', '1'
put 'threshold_ref_hbase', '3', 'alert:alert_message', 'Higher Heart Rate than
Normal'

put 'threshold_ref_hbase', '4', 'attribute:attribute', 'bp'
put 'threshold_ref_hbase', '4', 'limit:low_age_limit', '0'
put 'threshold_ref_hbase', '4', 'limit:high_age_limit', '40'
put 'threshold_ref_hbase', '4', 'limit:low_range_value', '0'
put 'threshold_ref_hbase', '4', 'limit:high_range_value', '160'
put 'threshold_ref_hbase', '4', 'alert:alert_flag', '1'
put 'threshold_ref_hbase', '4', 'alert:alert_message', 'Low BP than Normal'

put 'threshold_ref_hbase', '5', 'attribute:attribute', 'bp'
put 'threshold_ref_hbase', '5', 'limit:low_age_limit', '0'
put 'threshold_ref_hbase', '5', 'limit:high_age_limit', '40'
put 'threshold_ref_hbase', '5', 'limit:low_range_value', '161'
put 'threshold_ref_hbase', '5', 'limit:high_range_value', '220'
put 'threshold_ref_hbase', '5', 'alert:alert_flag', '0'
put 'threshold_ref_hbase', '5', 'alert:alert_message', 'Normal'
```

```
put 'threshold_ref_hbase', '6', 'attribute:attribute', 'bp'
put 'threshold_ref_hbase', '6', 'limit:low_age_limit', '0'
put 'threshold_ref_hbase', '6', 'limit:high_age_limit', '40'
put 'threshold_ref_hbase', '6', 'limit:low_range_value', '221'
put 'threshold_ref_hbase', '6', 'limit:high_range_value', '9999'
put 'threshold_ref_hbase', '6', 'alert:alert_flag', '1'
put 'threshold_ref_hbase', '6', 'alert:alert_message', 'Higher BP than Normal'

put 'threshold_ref_hbase', '7', 'attribute:attribute', 'heartBeat'
put 'threshold_ref_hbase', '7', 'limit:low_age_limit', '41'
put 'threshold_ref_hbase', '7', 'limit:high_age_limit', '100'
put 'threshold_ref_hbase', '7', 'limit:low_range_value', '0'
put 'threshold_ref_hbase', '7', 'limit:high_range_value', '65'
put 'threshold_ref_hbase', '7', 'alert:alert_flag', '1'
put 'threshold_ref_hbase', '7', 'alert:alert_message', 'Low Heart Rate than Normal'

put 'threshold_ref_hbase', '8', 'attribute:attribute', 'heartBeat'
put 'threshold_ref_hbase', '8', 'limit:low_age_limit', '41'
put 'threshold_ref_hbase', '8', 'limit:high_age_limit', '100'
put 'threshold_ref_hbase', '8', 'limit:low_range_value', '66'
put 'threshold_ref_hbase', '8', 'limit:high_range_value', '73'
put 'threshold_ref_hbase', '8', 'alert:alert_flag', '0'
put 'threshold_ref_hbase', '8', 'alert:alert_message', 'Normal'

put 'threshold_ref_hbase', '9', 'attribute:attribute', 'heartBeat'
put 'threshold_ref_hbase', '9', 'limit:low_age_limit', '41'
put 'threshold_ref_hbase', '9', 'limit:high_age_limit', '100'
put 'threshold_ref_hbase', '9', 'limit:low_range_value', '74'
put 'threshold_ref_hbase', '9', 'limit:high_range_value', '9999'
put 'threshold_ref_hbase', '9', 'alert:alert_flag', '1'
put 'threshold_ref_hbase', '9', 'alert:alert_message', 'Higher Heart Rate than
Normal'

put 'threshold_ref_hbase', '10', 'attribute:attribute', 'bp'
put 'threshold_ref_hbase', '10', 'limit:low_age_limit', '41'
put 'threshold_ref_hbase', '10', 'limit:high_age_limit', '100'
put 'threshold_ref_hbase', '10', 'limit:low_range_value', '0'
put 'threshold_ref_hbase', '10', 'limit:high_range_value', '150'
put 'threshold_ref_hbase', '10', 'alert:alert_flag', '1'
put 'threshold_ref_hbase', '10', 'alert:alert_message', 'Low BP than Normal'

put 'threshold_ref_hbase', '11', 'attribute:attribute', 'bp'
put 'threshold_ref_hbase', '11', 'limit:low_age_limit', '41'
put 'threshold_ref_hbase', '11', 'limit:high_age_limit', '100'
```

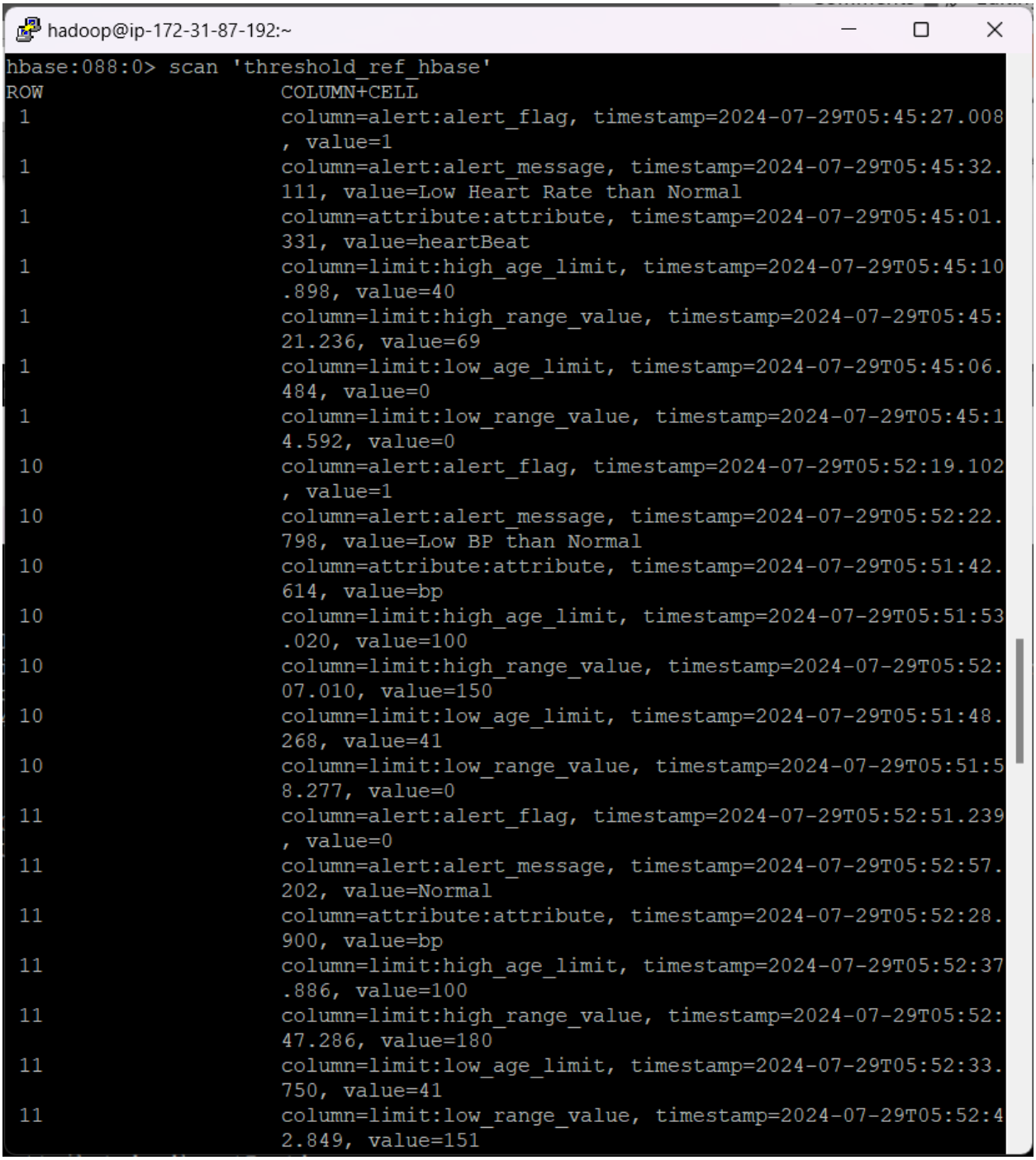
```
put 'threshold_ref_hbase', '11', 'limit:low_range_value', '151'
put 'threshold_ref_hbase', '11', 'limit:high_range_value', '180'
put 'threshold_ref_hbase', '11', 'alert:alert_flag', '0'
put 'threshold_ref_hbase', '11', 'alert:alert_message', 'Normal'

put 'threshold_ref_hbase', '12', 'attribute:attribute', 'bp'
put 'threshold_ref_hbase', '12', 'limit:low_age_limit', '41'
put 'threshold_ref_hbase', '12', 'limit:high_age_limit', '100'
put 'threshold_ref_hbase', '12', 'limit:low_range_value', '181'hive
put 'threshold_ref_hbase', '12', 'limit:high_range_value', '9999'
put 'threshold_ref_hbase', '12', 'alert:alert_flag', '1'
put 'threshold_ref_hbase', '12', 'alert:alert_message', 'Higher BP than Normal'
```





STATEMENTS TO VIEW HBASE THRESHOLD TABLE ABOVE MENTIONED AFTER DATA INSERTION:

scan 'threshold_ref_hbase'



STEP 5: STATEMENTS TO CREATE EXTERNAL HIVE TABLE FOR THRESHOLD TABLE PAREMETERS:

```
CREATE EXTERNAL TABLE IF NOT EXISTS threshold_ref_hive(
   ref_id INT,
   attribute VARCHAR(20),
   low_age_limit INT,
   high_age_limit INT,
   low_range_value INT,
   high_range_value INT,
   alert_flag INT,
   alert_message VARCHAR(255))
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES (
   "hbase.columns.mapping" = ":key,
   attribute:attribute,
   limit:low_age_limit,
   limit:high_age_limit,
   limit:low_range_value,
   limit:high_range_value,
   alert:alert_flag,
   alert:alert_message")
TBLPROPERTIES ("hbase.table.name" = "threshold_ref_hbase");
```

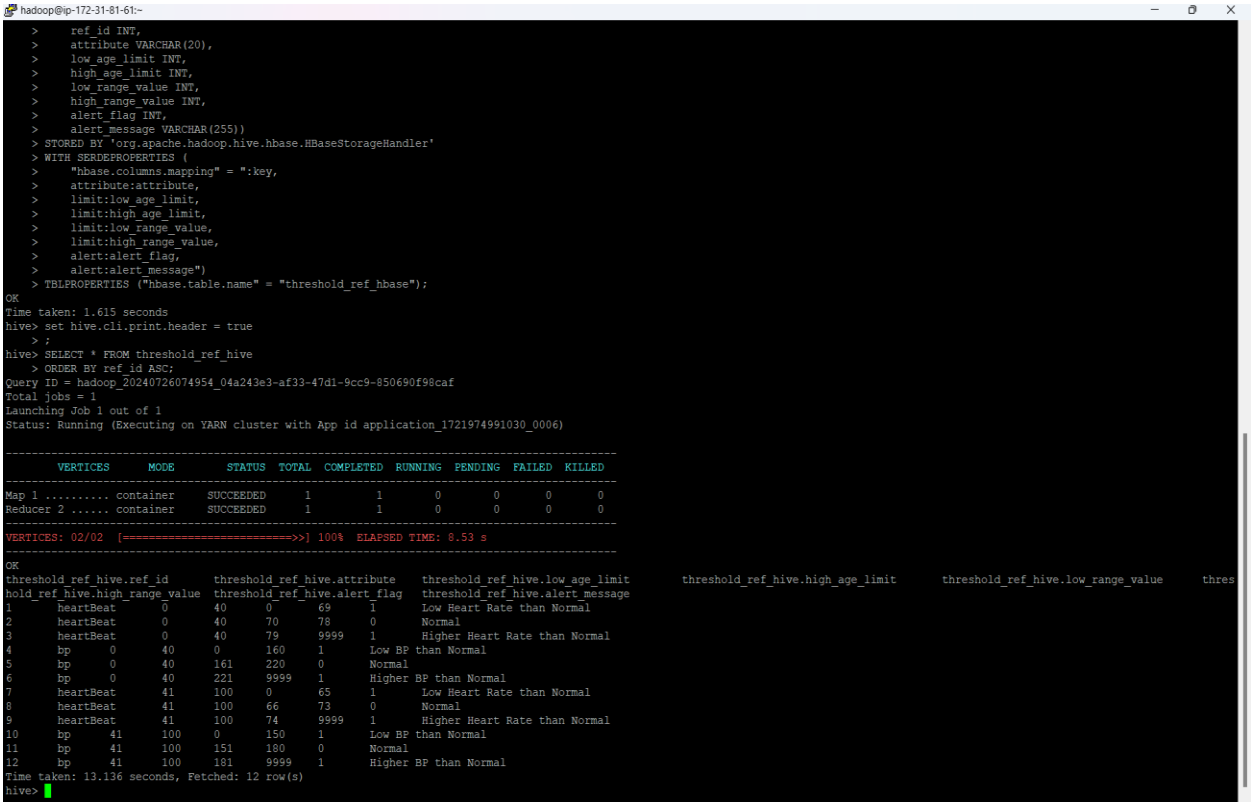
```
hadoop@ip-172-31-81-61:~
[hadoop@ip-172-31-81-61 ~]$ hive
Hive Session ID = 5c12e684-32c8-4638-9c84-72f3f1724dfb

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: true
hive> show tables;
OK
Time taken: 0.964 seconds
hive> CREATE EXTERNAL TABLE IF NOT EXISTS threshold_ref_hive(
    >     ref_id INT,
    >     attribute VARCHAR(20),
    >     low_age_limit INT,
    >     high_age_limit INT,
    >     low_range_value INT,
    >     high_range_value INT,
    >     alert_flag INT,
    >     alert_message VARCHAR(255))
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES (
    >     "hbase.columns.mapping" = ":key,
    >     attribute:attribute,
    >     limit:low_age_limit,
    >     limit:high_age_limit,
    >     limit:low_range_value,
    >     limit:high_range_value,
    >     alert:alert_flag,
    >     alert:alert_message")
    > TBLPROPERTIES ("hbase.table.name" = "threshold_ref_hbase");
OK
Time taken: 1.615 seconds
```

STATEMENT FOR VIEWING HEADER OF EACH COLUMNS:

```
set hive.cli.print.header=true;
```

STATEMENTS TO VIEW THRESHOLD DATA

```
SELECT * FROM threshold_ref_hive
ORDER BY ref_id ASC;
```



```
hadoop@ip-172-31-81-61:~
    >     ref_id INT,
    >     attribute VARCHAR(20),
    >     low_age_limit INT,
    >     high_age_limit INT,
    >     low_range_value INT,
    >     high_range_value INT,
    >     alert_flag INT,
    >     alert_message VARCHAR(255))
    > STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
    > WITH SERDEPROPERTIES (
    >     "hbase.columns.mapping" = ":key,
    >     attribute:attribute,
    >     limit:low_age_limit,
    >     limit:high_age_limit,
    >     limit:low_range_value,
    >     limit:high_range_value,
    >     alert:alert_flag,
    >     alert:alert_message")
    > TBLPROPERTIES ("hbase.table.name" = "threshold_ref_hbase");
OK
Time taken: 1.615 seconds
hive> set hive.cli.print.header = true
    > ;
hive> SELECT * FROM threshold_ref_hive
    > ORDER BY ref_id ASC;
Query ID = hadoop_20240726074954_04a243e3-af33-47d1-9cc9-850690f98caf
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1721974991030_0006)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 8.53 s
----------------------------------------------------------------------------------------------
OK
threshold_ref_hive.ref_id       threshold_ref_hive.attribute    threshold_ref_hive.low_age_limit        threshold_ref_hive.high_age_limit       threshold_ref_hive.low_range_value      thres
hold_ref_hive.high_range_value  threshold_ref_hive.alert_flag   threshold_ref_hive.alert_message
1       heartBeat       0       40      0       69      1       Low Heart Rate than Normal
2       heartBeat       0       40      70      78      0       Normal
3       heartBeat       0       40      79      9999    1       Higher Heart Rate than Normal
4       bp      0       40      0       160     1       Low BP than Normal
5       bp      0       40      161     220     0       Normal
6       bp      0       40      221     9999    1       Higher BP than Normal
7       heartBeat       41      100     0       65      1       Low Heart Rate than Normal
8       heartBeat       41      100     66      73      0       Normal
9       heartBeat       41      100     74      9999    1       Higher Heart Rate than Normal
10      bp      41      100     0       150     1       Low BP than Normal
11      bp      41      100     151     180     0       Normal
12      bp      41      100     181     9999    1       Higher BP than Normal
Time taken: 13.136 seconds, Fetched: 12 row(s)
hive>
```

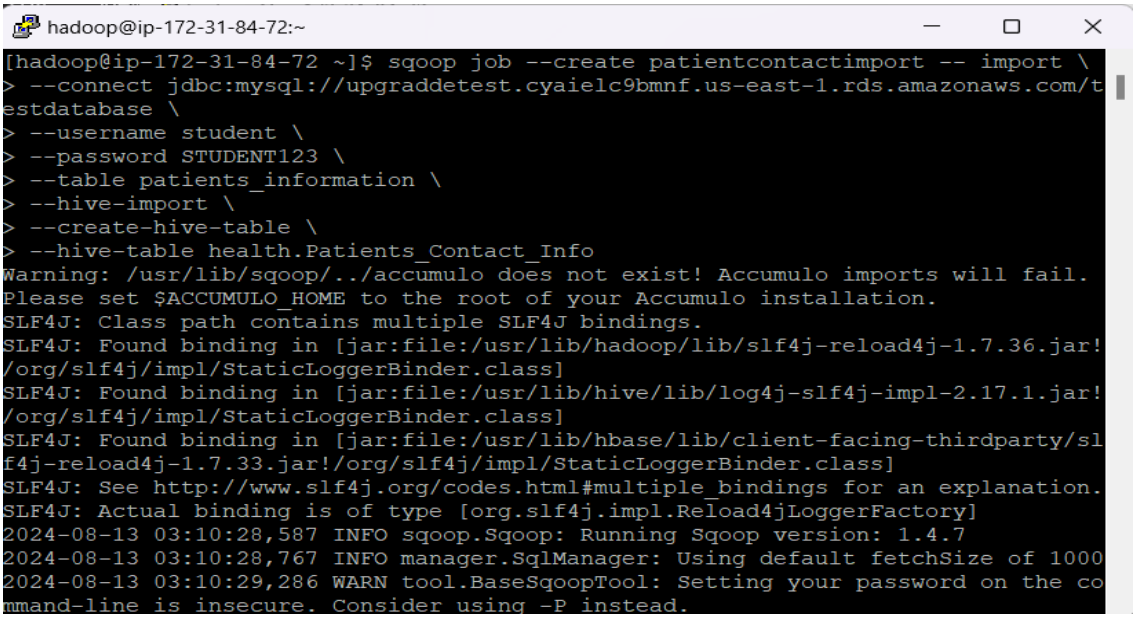
So till now task 2 was accomplished and we have patients_vital_info stored in hdfs and threshold_ref_hive table storing the threshold parameters.

STEP 6: Extract patient info using sqoop into hive table.

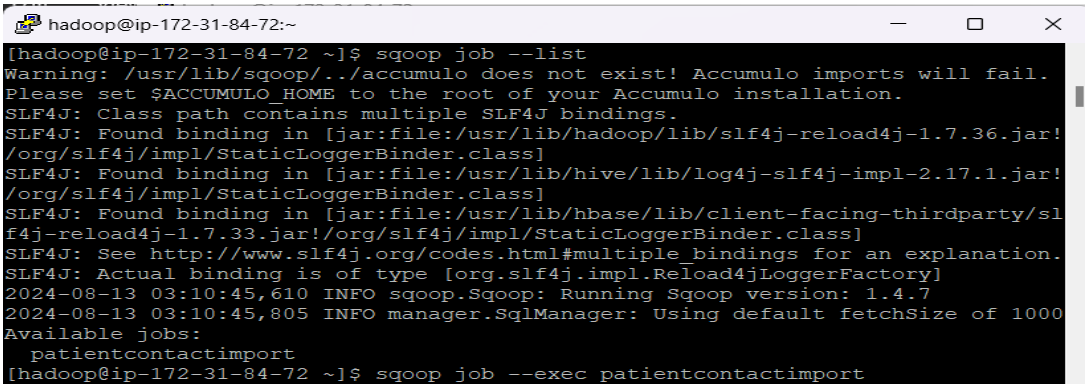
SQOOP COMMAND

SQOOP COMMAND FOR 'PATIENTCONTACTIMPORT' JOB CREATION:

```
sqoop job --create patientcontactimport -- import \
--connect jdbc:mysql://upgraddetest.cyaielc9bmnf.us-east-
1.rds.amazonaws.com/testdatabase \
--username student \
--password STUDENT123 \
--table patients_information \
--hive-import \
--create-hive-table \
--hive-table health.Patients_Contact_Info
```

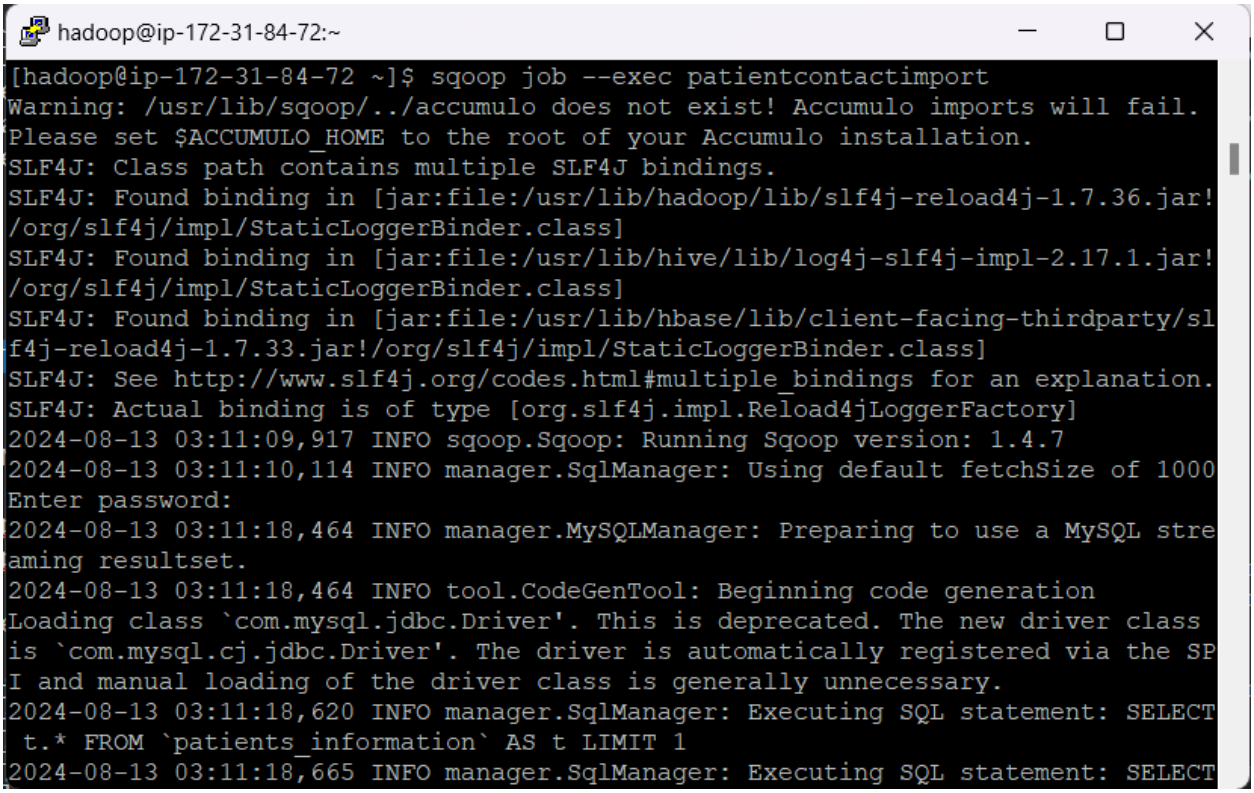



```
sqoop job –-list
```



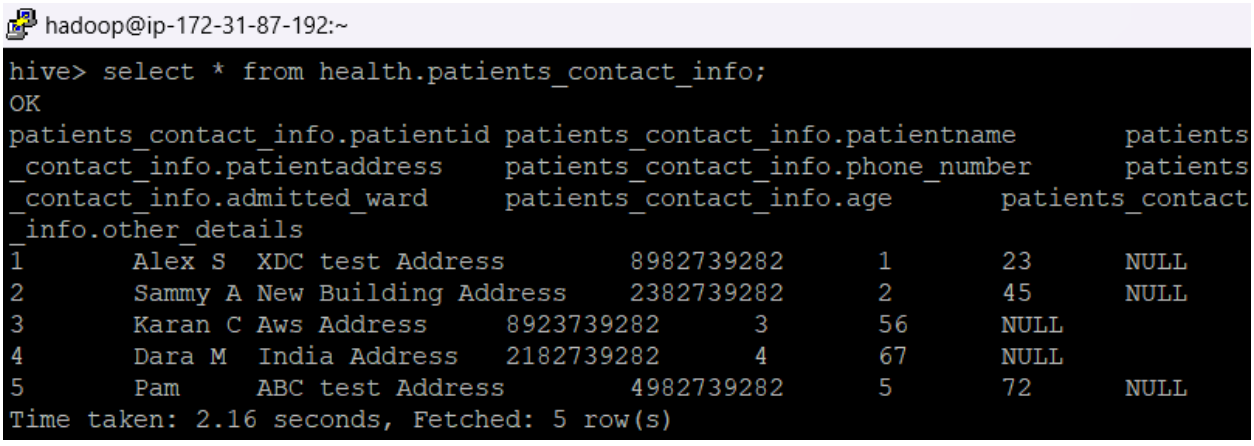
SQOOP COMMAND FOR 'PATIENTCONTACTIMPORT' JOB EXECUTION:

sqoop job --exec patientcontactimport



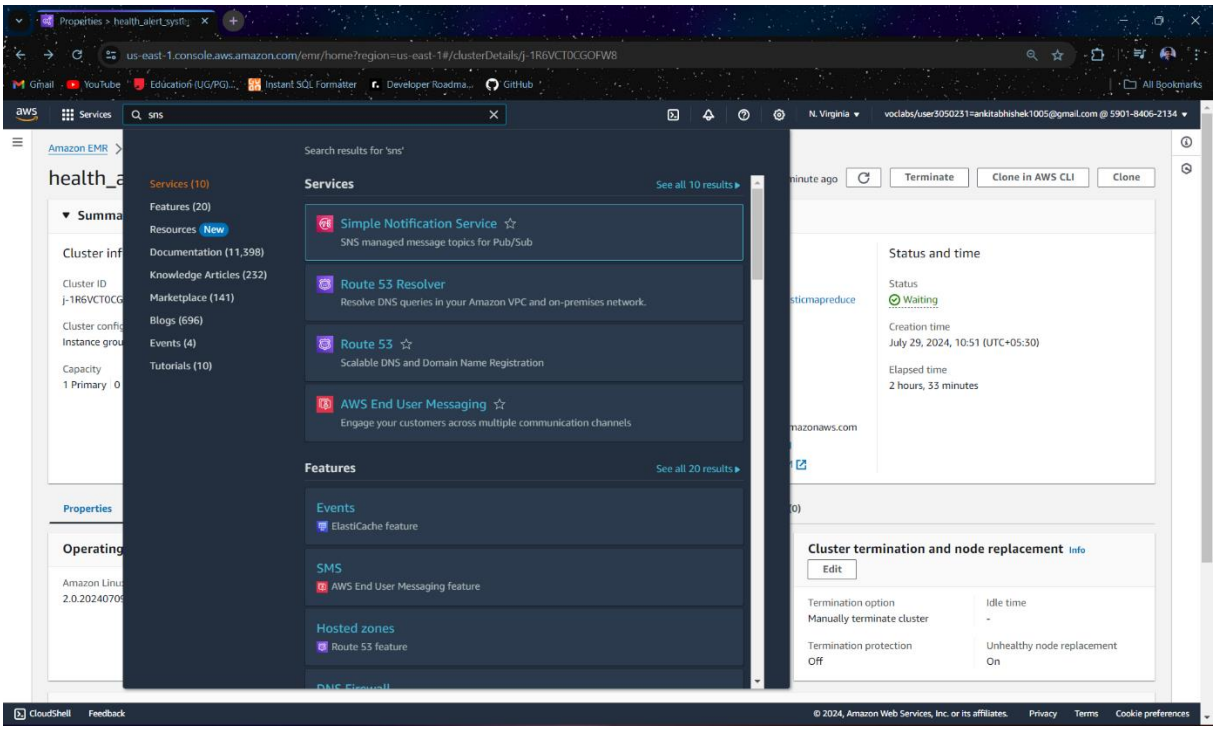
SNAPSHOT OF DATA COLLECTED:

select * from health.patients_contact_info;

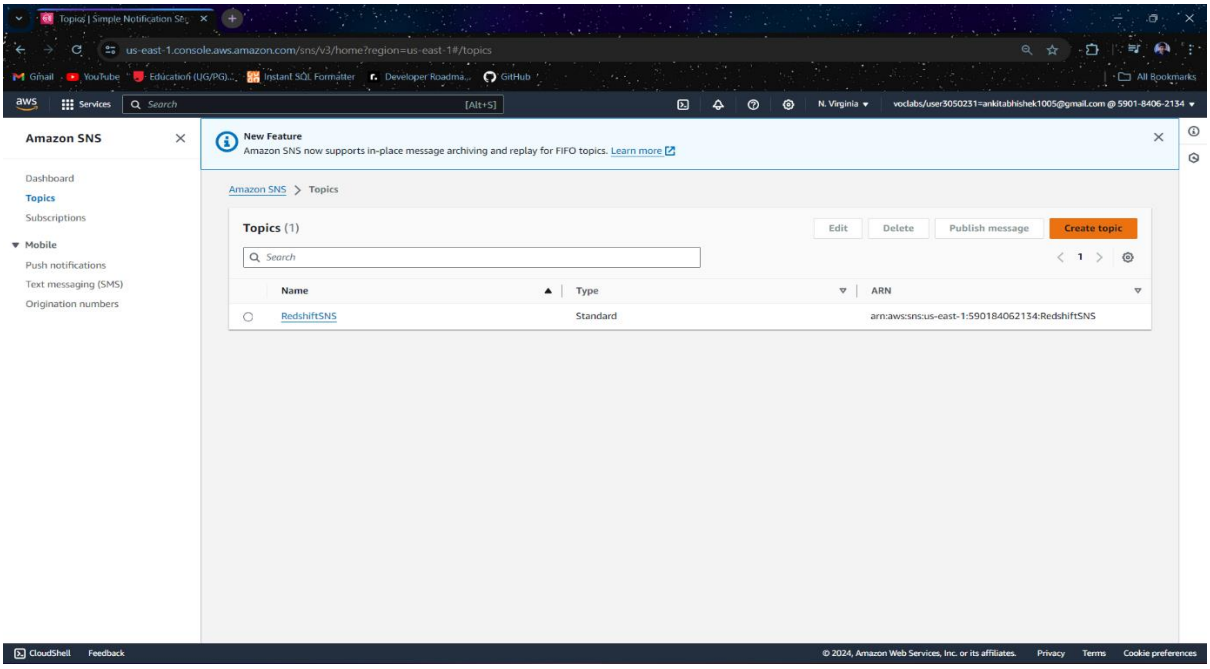


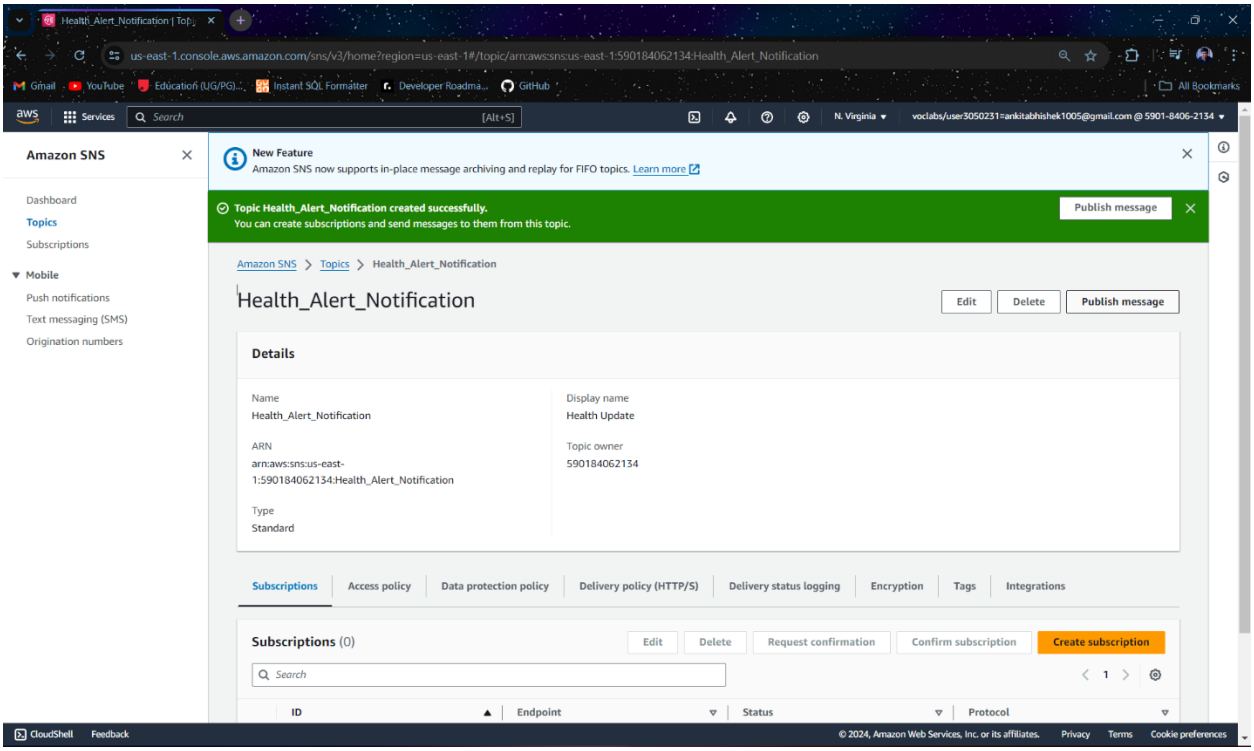
STEP 7: SNS configuration to the assessment email ID

STEP 1: SEARCH FOR AMAZON SNS (SIMPLE NOTIFICATION SERVICE):

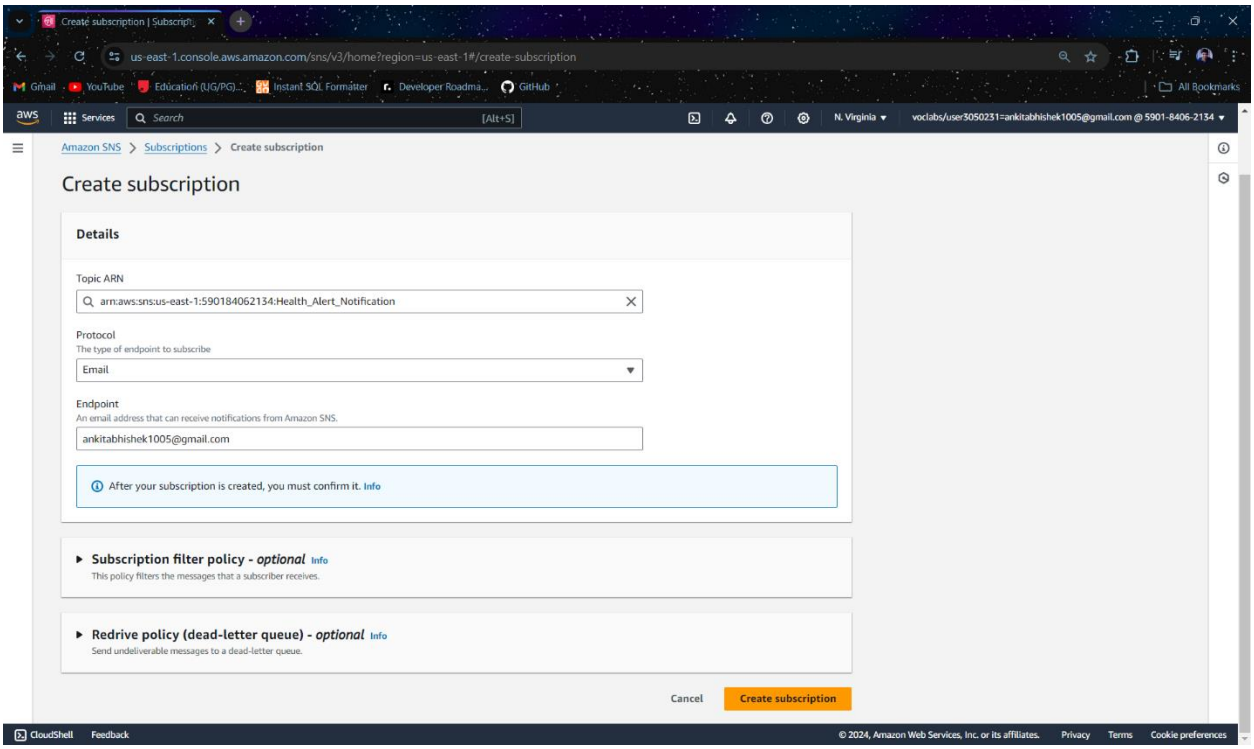


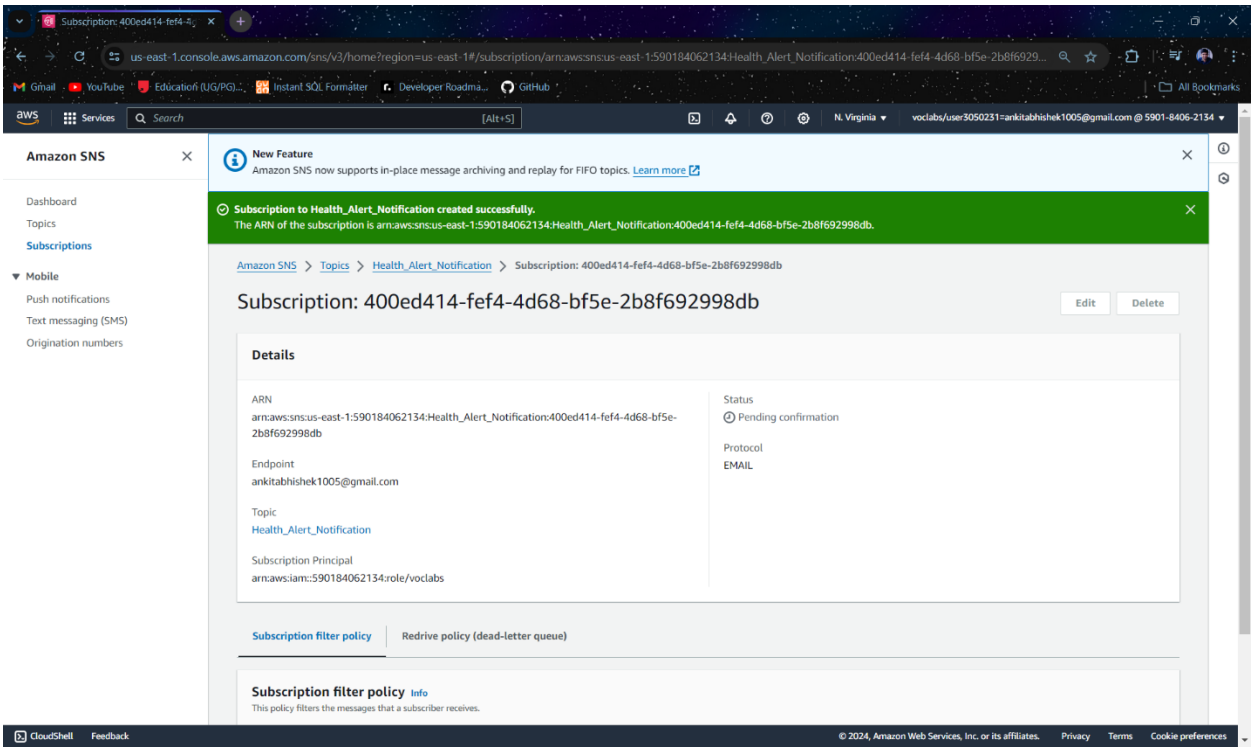
STEP 2: CREATE TOPIC 'HEALTH_ALERT_NOTIFICATION':



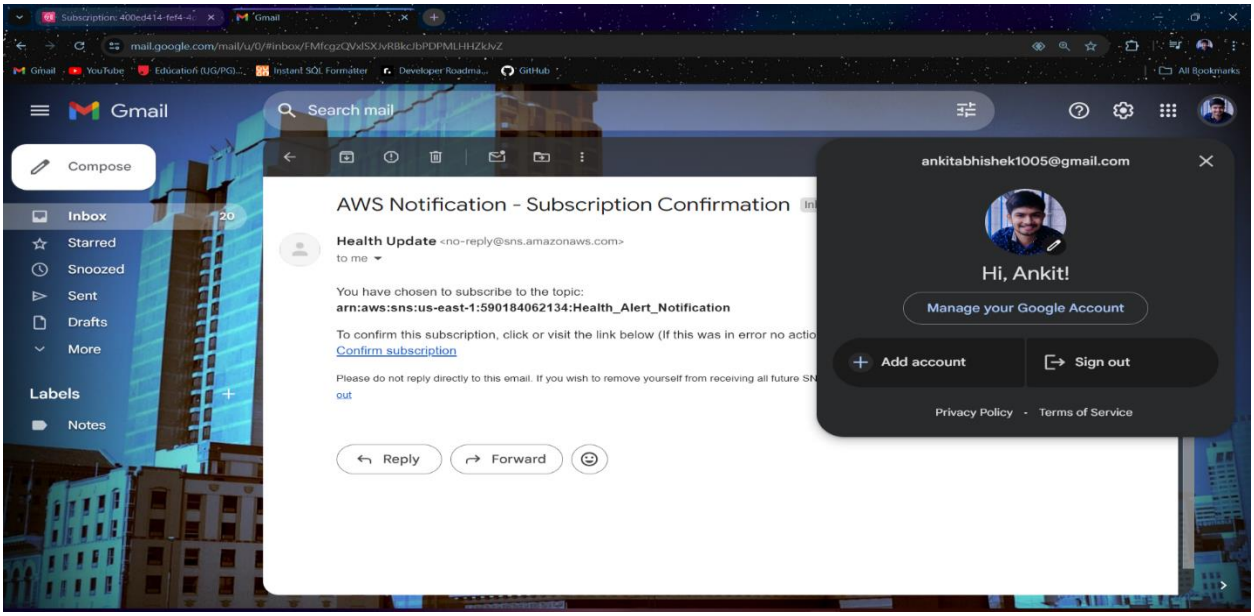


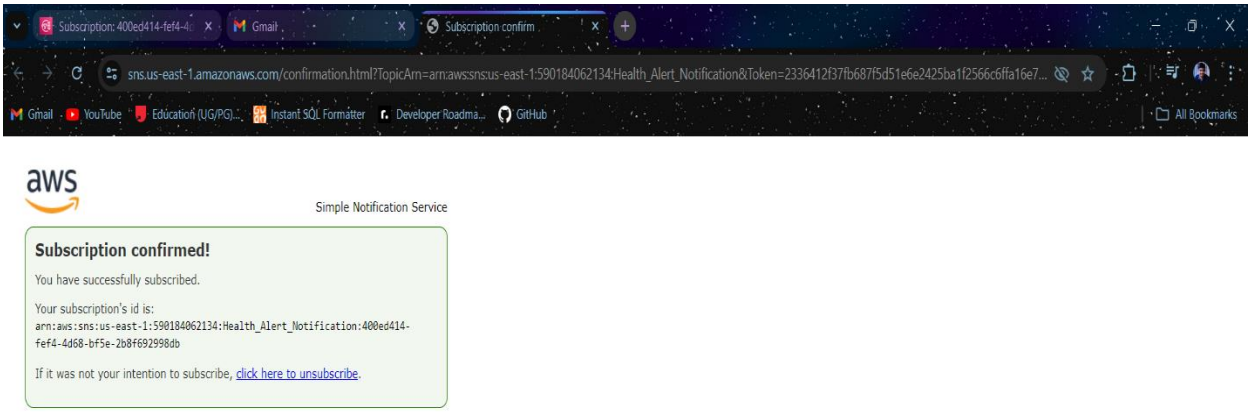
STEP 3: NOW CREATE SUBSCRIPTION FOR COLLECTING HEALTH_ALERTS:
Select Protocol: Email
Inside Endpoint: **Provided email id** where we want to get alert from the Kafka_Topic



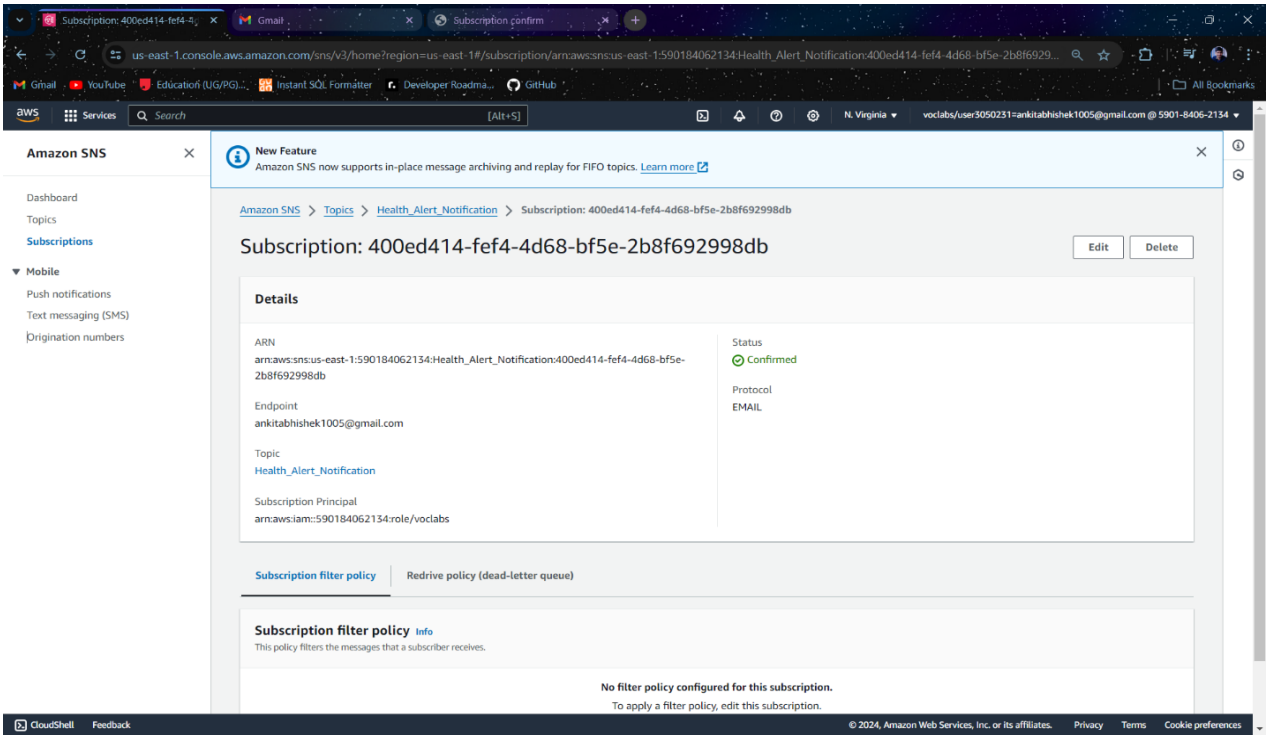


STEP 4: CONFIRM FROM THE EMAIL ID WHERE **SUBSCRIPTION CONFIRMATION EMAIL** ARRIVED:



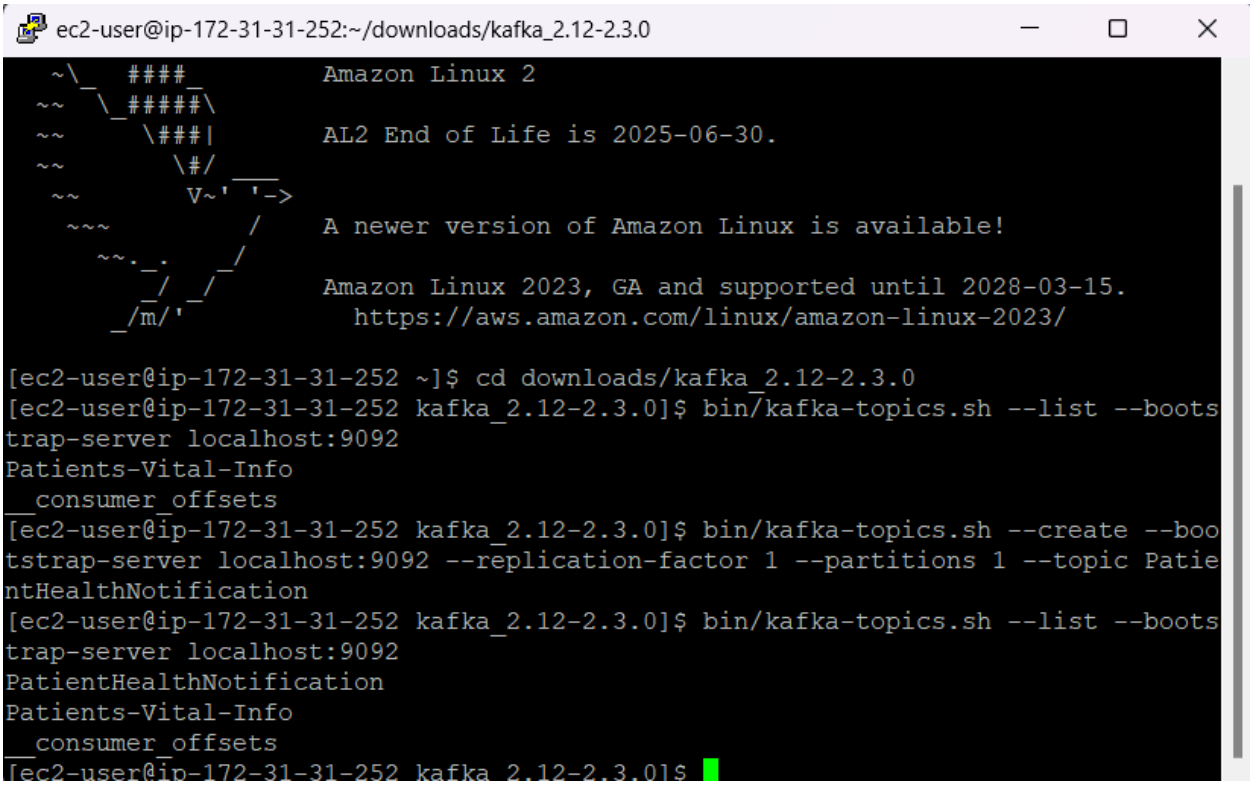


STEP 5: AS **SATUS OF SUBSCRIPTION** IS SWITCHED FROM **PENDING CONFIRMATION** TO **CONFIRMED:**



STEP 8: Creation of PatientHealthNotification topic for receiving the alerts to the KafkaQueue from where SNS wiill consume

bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic PatientHealthNotification



STEP 9: EXECUTION OF PRODUCER SAPRK APPLICATION AND CONSUMER APPLICATION

Execution of Spark streaming application to read data from HDFS and compare it with hbase and transfer the output data to PatientHealthNotification topic:

```
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.1
kafka_spark_generate_alerts.py
```

Execution of Consumer application to receive email regarding generated alerts received on PatientHealthNotification topic

```
python3 kafka_consume_alerts.py
```

STEP 10: Soon we were able to receive Alerts for first user as soon as Higher BP than normal was caught up by the spark streaming application as mentioned in below screenshot: