

## IoT Project

### Instructions

Every screenshot requested in this workbook is compulsory and carries 8 marks.

Your AWS account ID must be clearly visible in every screenshot using the AWS console; missing id or using someone else's id is not permitted. Such cases will be considered as plagiarism and severe penalty will be imposed.

All screenshots must be in the order mentioned under "Expected Screenshots" for every step

DO NOT WAIT UNTIL THE LAST MINUTE. The program office will not extend the project submission deadline under any circumstances.

The file should be renamed in the format BATCH\_FIRSTNAME\_LASTNAME\_PROJECT1.  
For example: ACSE\_Batch\_VIJAY\_DWIVEDI\_PROJECT1.docx

### Resource Clean Up

Cloud is always a pay per use model and all resources/services that we consume are chargeable. Cleaning up when you've completed your lab or project is always necessary. This is true whether you're doing a lab or implementing a project at your workplace.

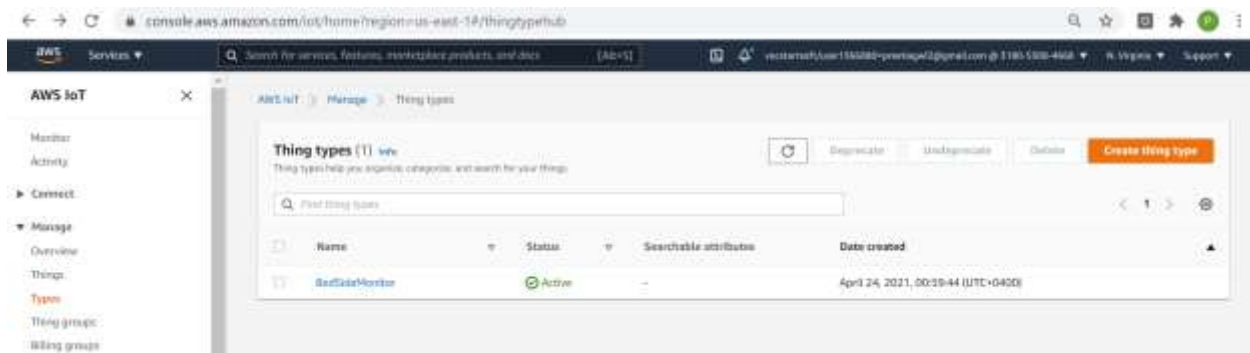
After completing the lab, make sure to delete each resource created in reverse chronological order.

## Creating Things, DynamoDB table

Step number	a
Step name	IoT Core and Things
Instructions	<ol style="list-style-type: none"> <li>1) Goto IoT core and create the types of things you wish to create.</li> <li>2) Create at least one group and parent group that will be attached to things.</li> <li>3) Create a policy that will be attached with newly created IoT devices.</li> <li>4) Create at least two devices on the IoT core to send and receive data.</li> </ol>
Expected screenshots	<ol style="list-style-type: none"> <li>1) Screenshot of successfully create thing type</li> <li>2) Screenshot of successfully created group.</li> <li>3) Screenshot of successfully created policy page.</li> <li>4) Screenshot of successfully created thing.</li> </ol>

<Insert Screenshot a(1) here>

## Thing Type



console.aws.amazon.com/iot/home?region=us-east-1#/thingtypehub

AWS IoT Manage Thing types

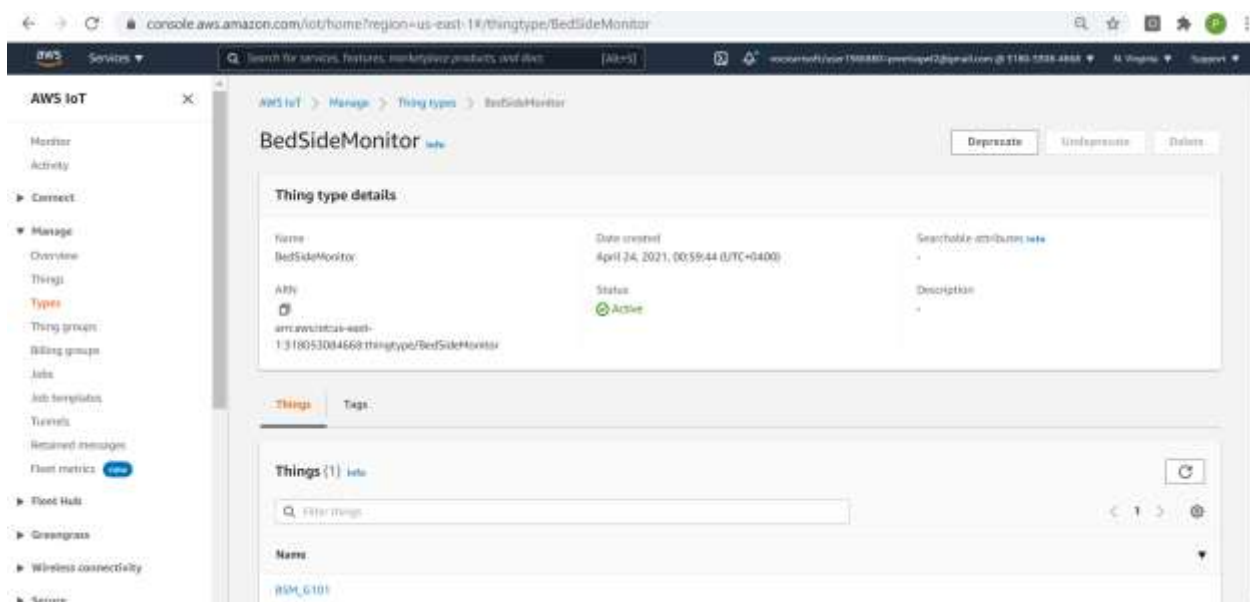
Thing types (1) [info](#)

Thing types help you organize categories and search for your things.

Filter thing types

Name	Status	Searchable attributes	Date created
BedSideMonitor	Active	-	April 24, 2021, 00:39:44 (UTC+0400)

Buttons: Deprecate, Undeprecate, Delete, Create thing type



console.aws.amazon.com/iot/home?region=us-east-1#/thingtype/BedSideMonitor

AWS IoT Manage Thing types BedSideMonitor

BedSideMonitor [info](#)

Buttons: Deprecate, Undeprecate, Delete

Thing type details

Name	BedSideMonitor	Date created	April 24, 2021, 00:39:44 (UTC+0400)	Searchable attributes	<a href="#">info</a>
ARN	arn:aws:iot:us-east-1:318053084660:thingtype/BedSideMonitor				
Status	Active				
Description	-				

Buttons: Things, Tags

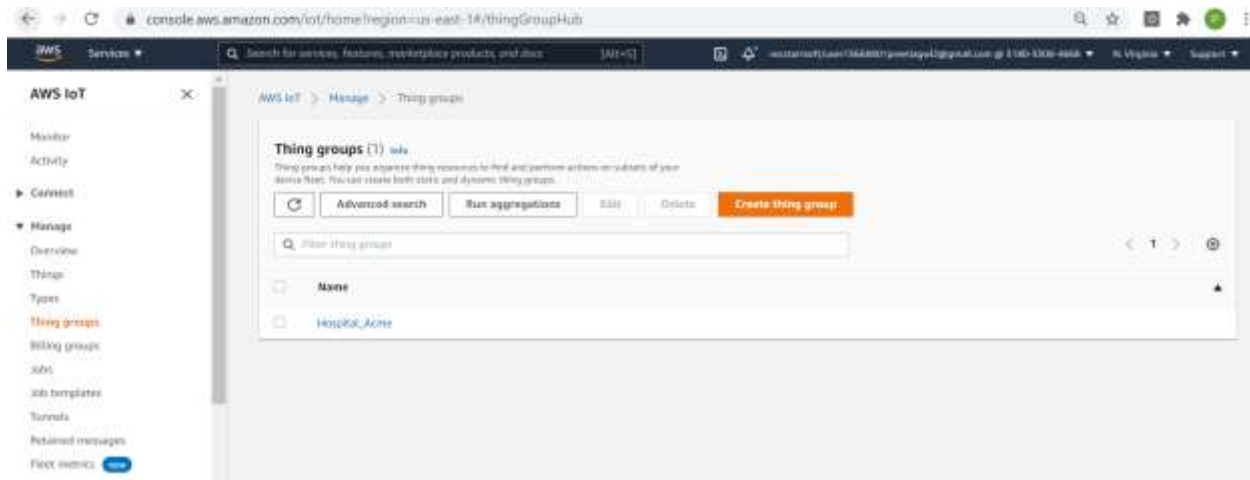
Things (1) [info](#)

Filter things

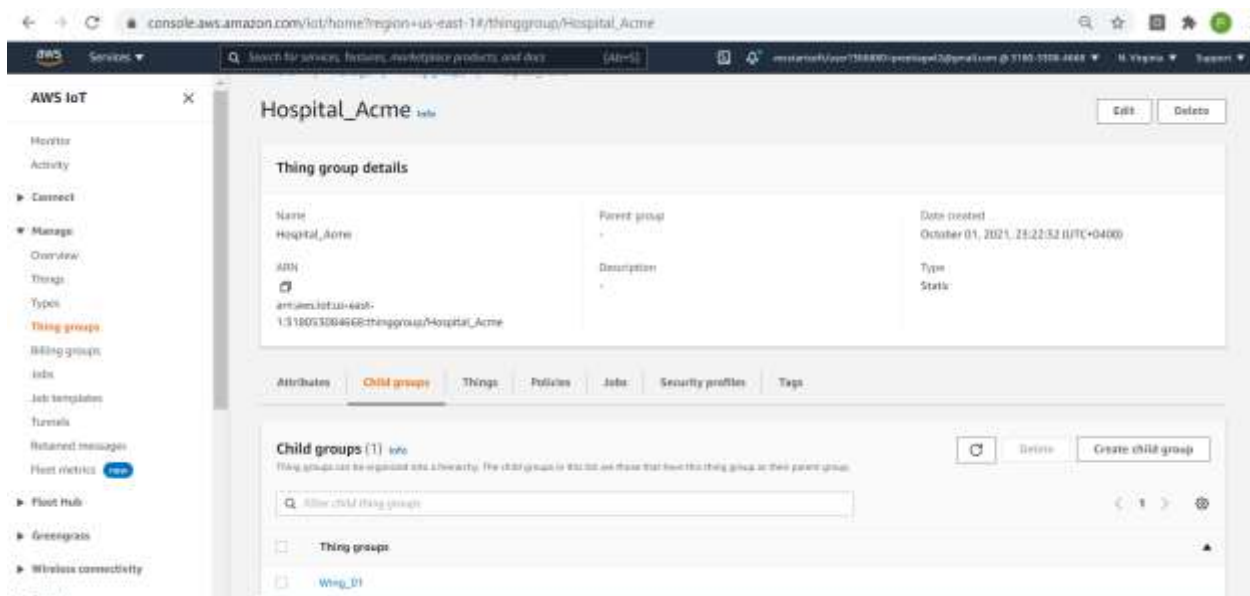
Name
BSM_6101

<Insert Screenshot a(2) here>

Parent group –



Child group –



console.aws.amazon.com/iot/home?region=us-east-1#/thinggroup/Wing\_01

**AWS IoT** Services Search for services, features, marketplace products, and docs US (+0) Account: user130000@gmail.com @ 910-100-0000 N. Virginia Support

**Wing\_01** info Edit Delete

**Thing group details**

<b>Name</b> Wing_01	<b>Parent group</b> Hospital_Acme	<b>Date created</b> October 01, 2021, 23:22:43 (UTC+0400)
<b>ARN</b> arn:aws:iot-us-east-1:310053064066:thinggroup/Wing_01		<b>Type</b> Static

**Attributes** Child groups Things Policies Jobs Security profiles Tags

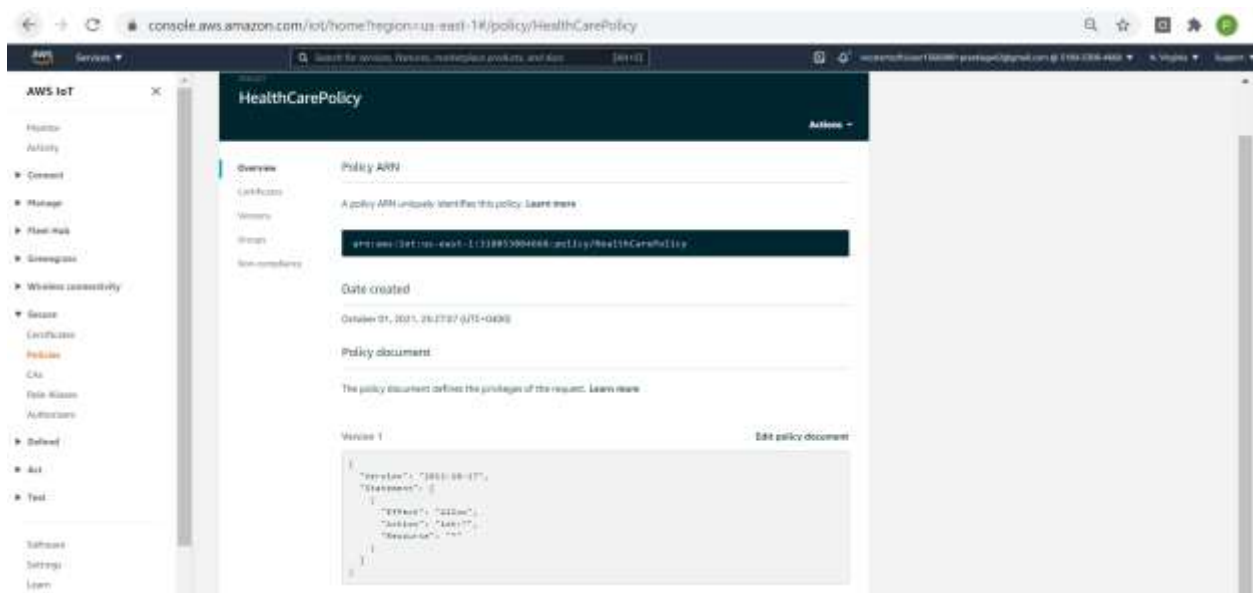
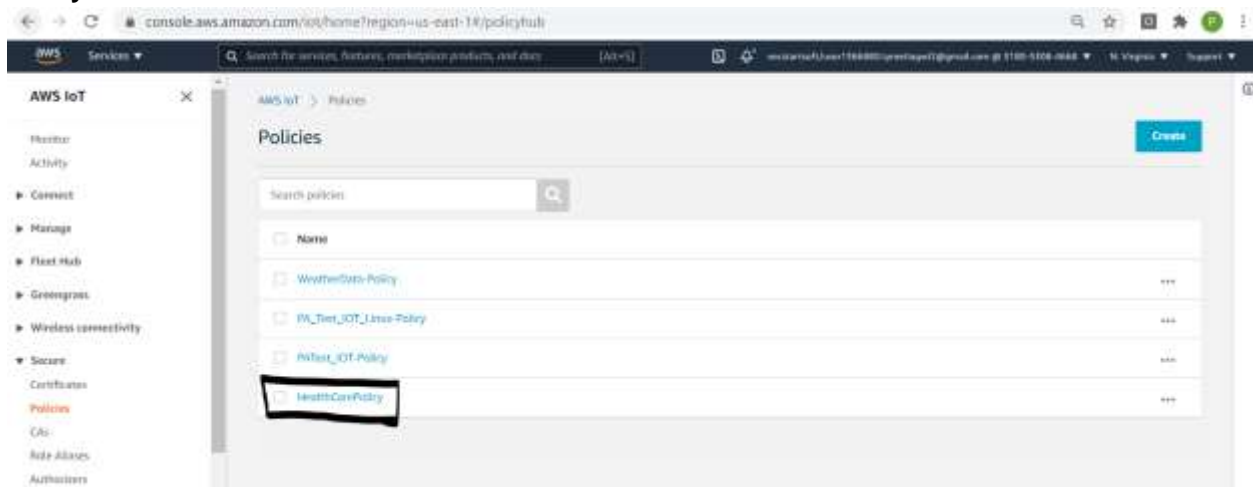
**Attributes (0)** info

Attributes store metadata about the thing group. Attributes can be used to filter lists of thing groups.

Key	Value
No attributes There are no attributes to display.	

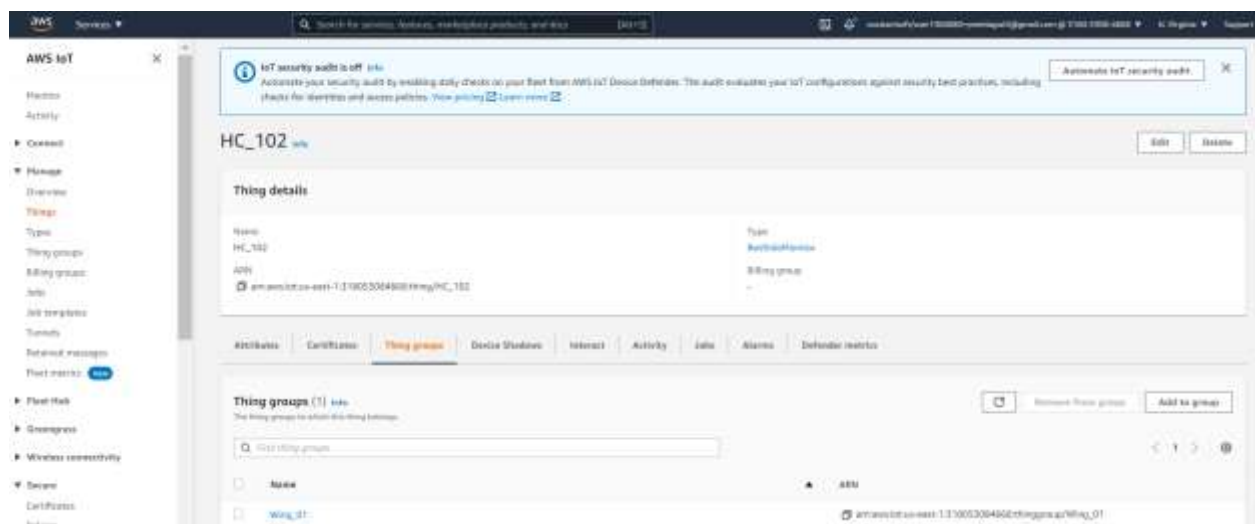
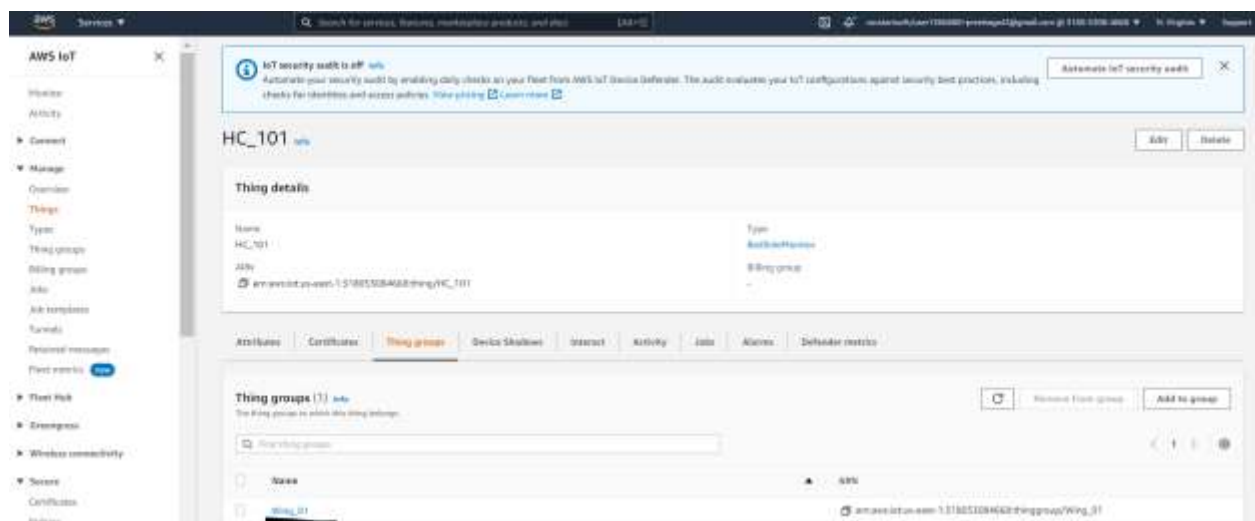
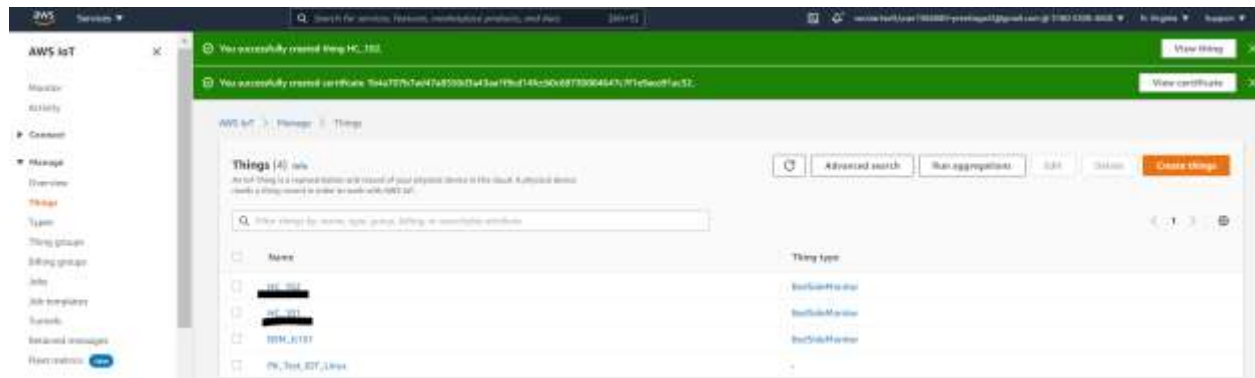
<Insert Screenshot a(3) here>

## Policy-



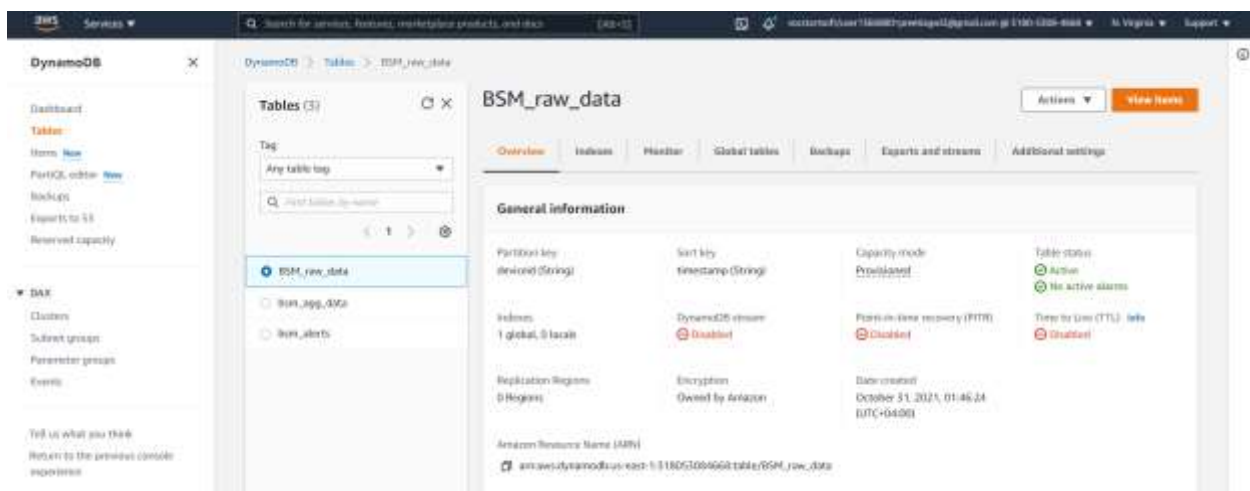
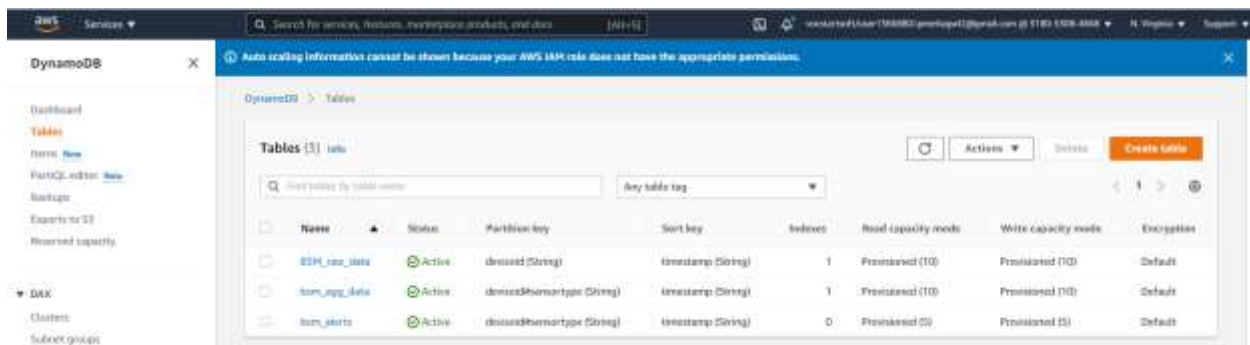
**<Insert Screenshot a(4) here>**

## Things –

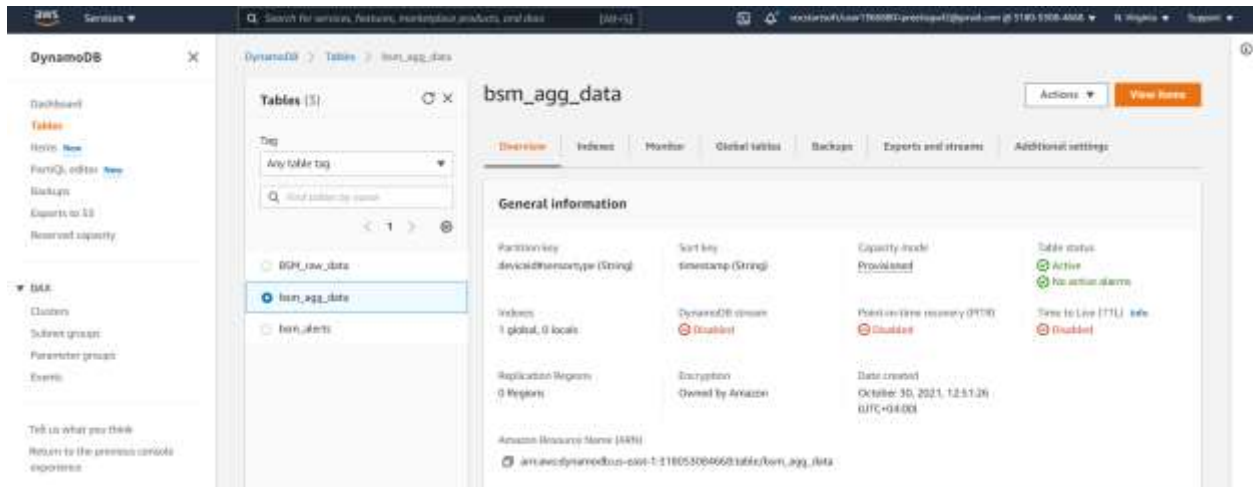


Step number	b
Step name	DynamoDB
Instructions	1) Goto DynamoDB console or use python script to create the table to store the raw_data, aggregate_data and anomaly-data with appropriate partition key and sort key 2) Assign the name to the table to store the raw data 3) The name of the tables should be clearly visible in the screenshot.
Expected screenshots	1. Screenshot of successfully created table to hold the raw data. (Empty table) 2. Screenshot of successfully created table to hold the aggregated/ data. (Empty table) 3. Screenshot of successfully created table to hold the anomaly data. (Empty table)

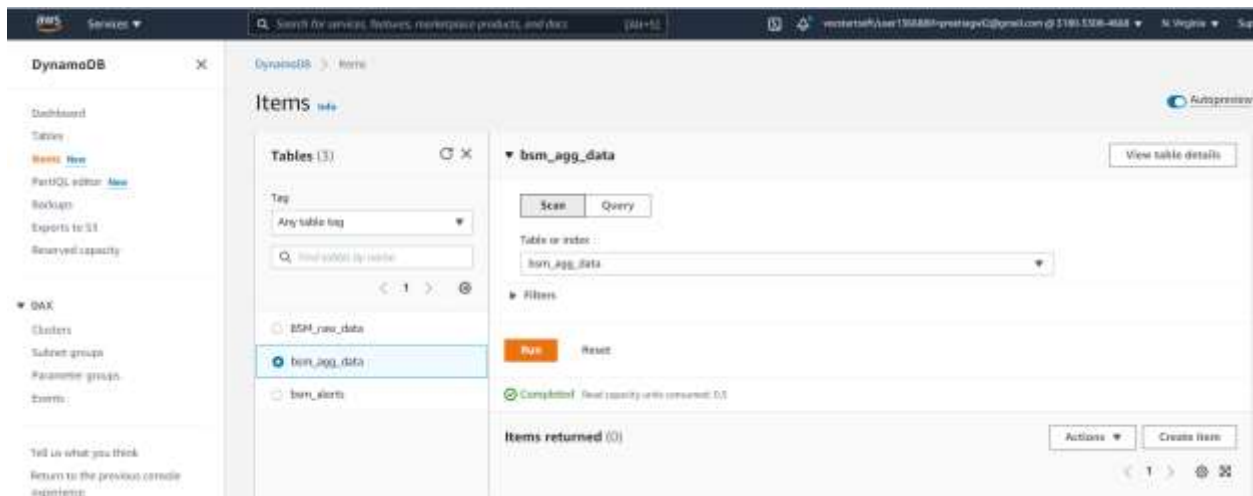
<Insert Screenshot b(1) here>



<Insert Screenshot b(2) here>



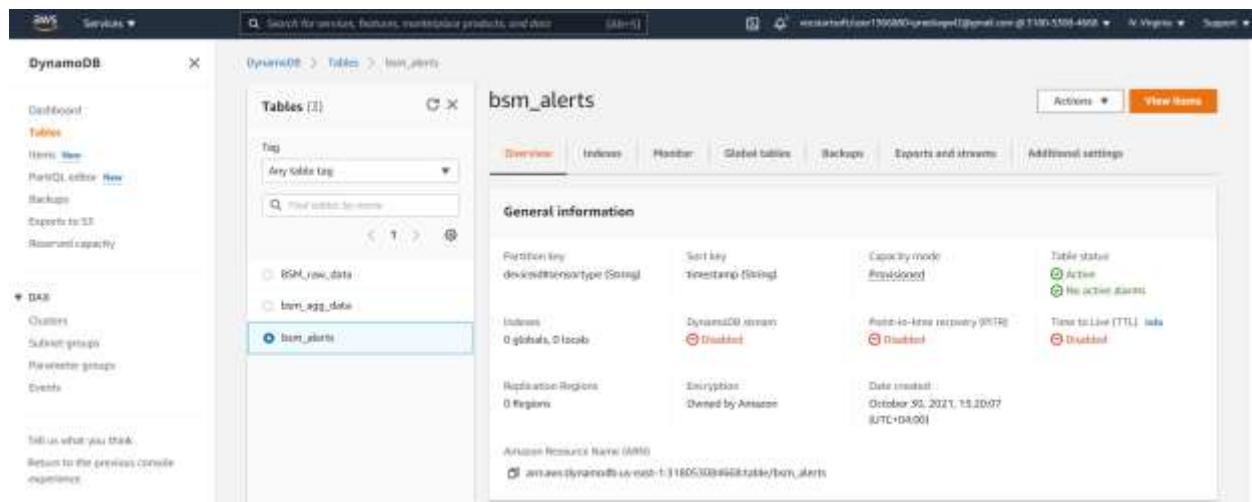
The screenshot shows the AWS Management Console interface for the **bsm\_agg\_data** table in DynamoDB. The left sidebar contains navigation links for Dashboard, Tables, Items, PartiQL editor, Backups, Exports to S3, and Reserved capacity. The main content area displays the table's general information, including its partition key (deviceid), sort key (timestamp), capacity mode (Provisioned), and table status (Active). It also shows the number of indexes (1 global, 0 local), replication regions (0), and encryption status (Enabled by Amazon). The table was created on October 30, 2021, at 12:51:26 UTC+04:00.



The screenshot shows the AWS Management Console interface for the **bsm\_agg\_data** table in DynamoDB, specifically the **Items** view. The left sidebar is identical to the previous screenshot. The main content area displays the table's name and a search bar. Below the search bar, there are buttons for **Scan** and **Query**. The **Table or index** dropdown is set to **bsm\_agg\_data**. The **Filters** section is empty. The **Run** button is highlighted, and the status indicates **Completed** with a message: "Read capacity units consumed: 0.5". The **Items returned** section shows **0** items.

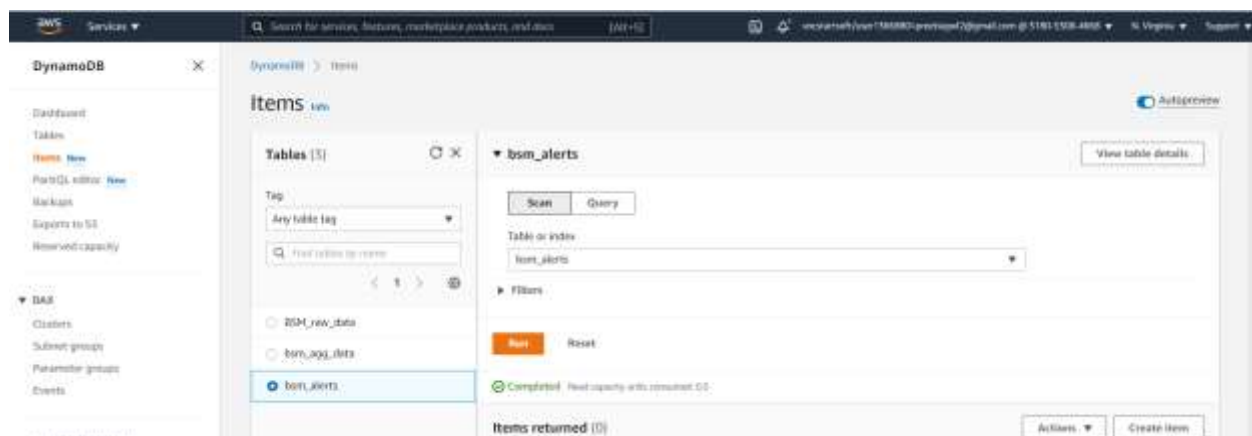


<Insert Screenshot b(3) here>



The screenshot shows the AWS DynamoDB console interface. On the left, there's a navigation menu with options like Dashboard, Tables, Items, PartiQL editor, Backups, Exports to S3, and Reserved capacity. The main area displays the 'bsm\_alerts' table details. The 'General information' section includes:

- Partition key: deviceid (String)
- Sort key: timestamp (String)
- Capacity mode: Provisioned
- Table status: Active
- Indices: 0 global, 0 local
- DynamoDB stream: Disabled
- Point-in-time recovery (PITR): Disabled
- Time to Live (TTL): Disabled
- Replication Regions: 0 Regions
- Encryption: Enabled by Amazon
- Date created: October 30, 2021, 15:20:07 (UTC+08:00)
- Amazon Resource Name (ARN): arn:aws:dynamodb:us-east-1:160530846888:table/bsm\_alerts



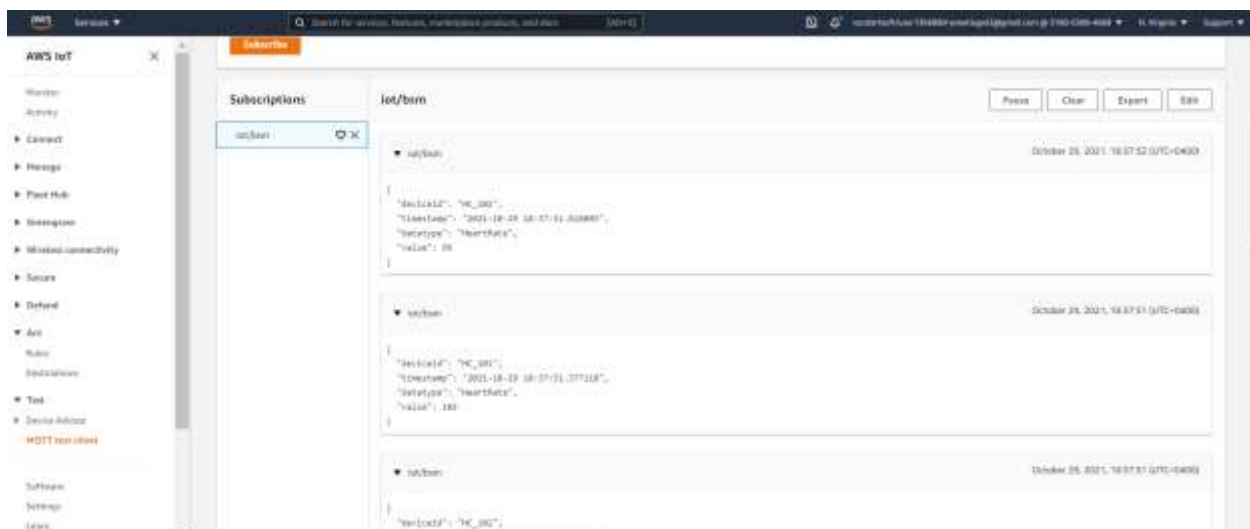
The screenshot shows the AWS DynamoDB console interface, specifically the 'Items' view for the 'bsm\_alerts' table. The 'Items' section displays a table with columns for 'Scan' and 'Query'. Below the table, there's a 'Filter' section with a 'Reset' button. The 'Items returned' section shows 0 items. The 'Actions' section includes a 'Create item' button.

Step number	c
Step name	Rule creation to push raw data
Instructions	<p>1) Goto IoT core and create rule to push simulated data in the raw data table created in the above step</p> <p>2) Once the rule is created, run the python script to push data on AWS MQTT. If the created rule is valid then data will be available in the previously created table.</p>
Expected screenshots	<p>1. Screenshot of successfully created rule to push the raw data</p> <p>2. Screenshot of data available in the AWS MQTT</p>

<Insert Screenshot c(1) here>

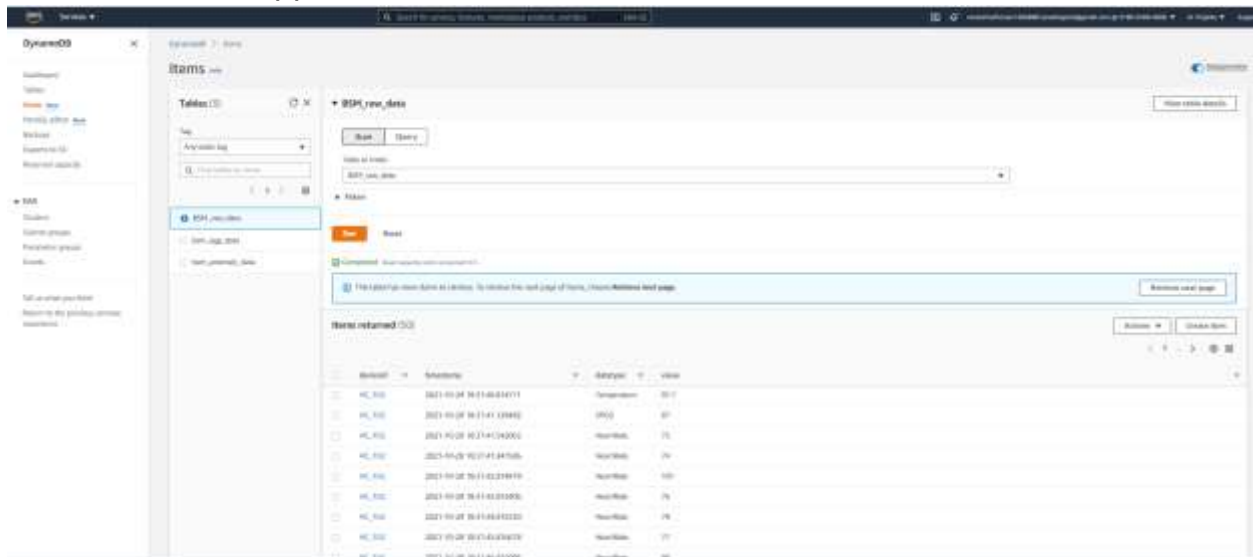


<Insert Screenshot c(2) here>

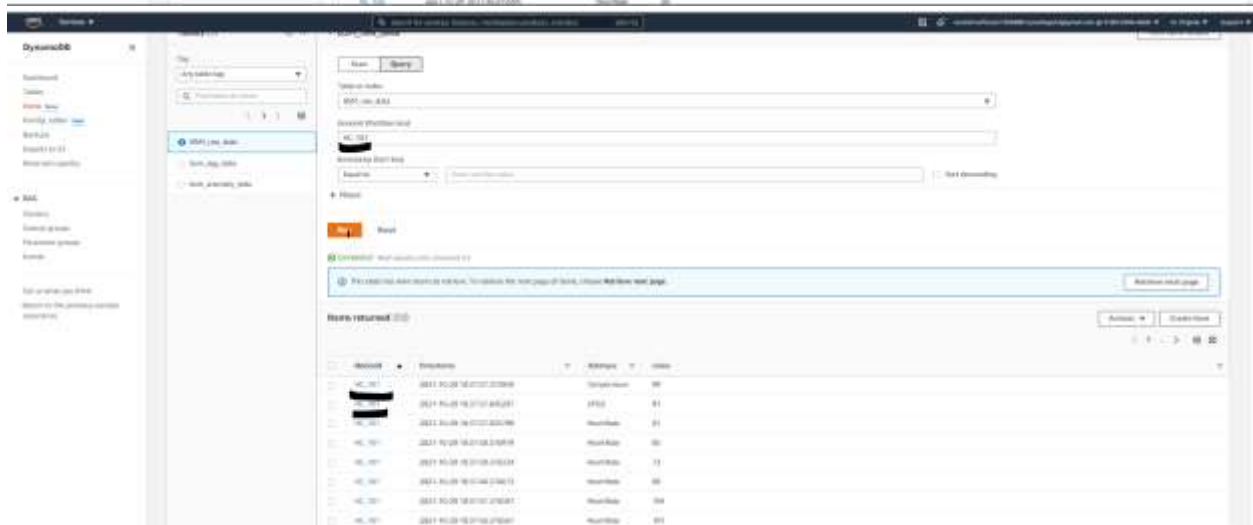


Step number	d
Step name	Data in the DynamoDB table
Instructions	<p>1) Once the tables are created and raw data rule is created to push the data</p> <p>2) Perform aggregation as mentioned in Task-2 of the problem statement, once done the data should be there in the newly created table.</p> <p>3) Perform anomaly as per the task-3 available in the problem statement and then push the data in the newly created table.</p>
Expected screenshots	<p>1. Screenshot of populated raw data in the table</p> <p>2. Screenshot of populated aggregated data in the table</p> <p>3. Screenshot of populated anomaly data in the table</p>

<Insert Screenshot d(1) here>



RecordId	Timestamp	Category	Value
HL_101	2021-05-28 10:14:00.000	Temperature	30.0
HL_101	2021-05-28 10:14:10.000	HRMS	97
HL_101	2021-05-28 10:14:15.000	HeartRate	75
HL_101	2021-05-28 10:14:20.000	HeartRate	76
HL_101	2021-05-28 10:14:25.000	HeartRate	76
HL_101	2021-05-28 10:14:30.000	HeartRate	76
HL_101	2021-05-28 10:14:35.000	HeartRate	76
HL_101	2021-05-28 10:14:40.000	HeartRate	76
HL_101	2021-05-28 10:14:45.000	HeartRate	77
HL_101	2021-05-28 10:14:50.000	HeartRate	78

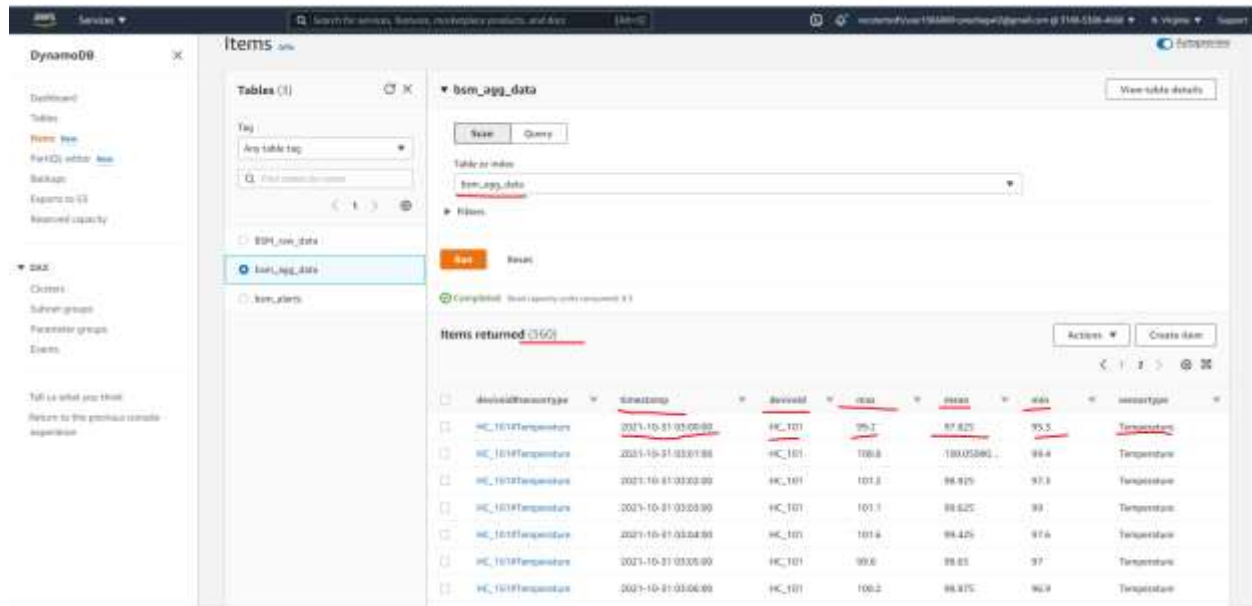


RecordId	Timestamp	Category	Value
HL_101	2021-05-28 10:14:00.000	Temperature	30
HL_101	2021-05-28 10:14:10.000	HRMS	91
HL_101	2021-05-28 10:14:15.000	HeartRate	81
HL_101	2021-05-28 10:14:20.000	HeartRate	80
HL_101	2021-05-28 10:14:25.000	HeartRate	71
HL_101	2021-05-28 10:14:30.000	HeartRate	80
HL_101	2021-05-28 10:14:35.000	HeartRate	84
HL_101	2021-05-28 10:14:40.000	HeartRate	84
HL_101	2021-05-28 10:14:45.000	HeartRate	84
HL_101	2021-05-28 10:14:50.000	HeartRate	84

<Insert Screenshot d(2) here>

Total 360 records inserted into bsm\_agg\_data table for 2 devices HC\_101 and HC\_102 for the 3 sensor types – HeartRate, Temperature and SPO2 for 1 hour.

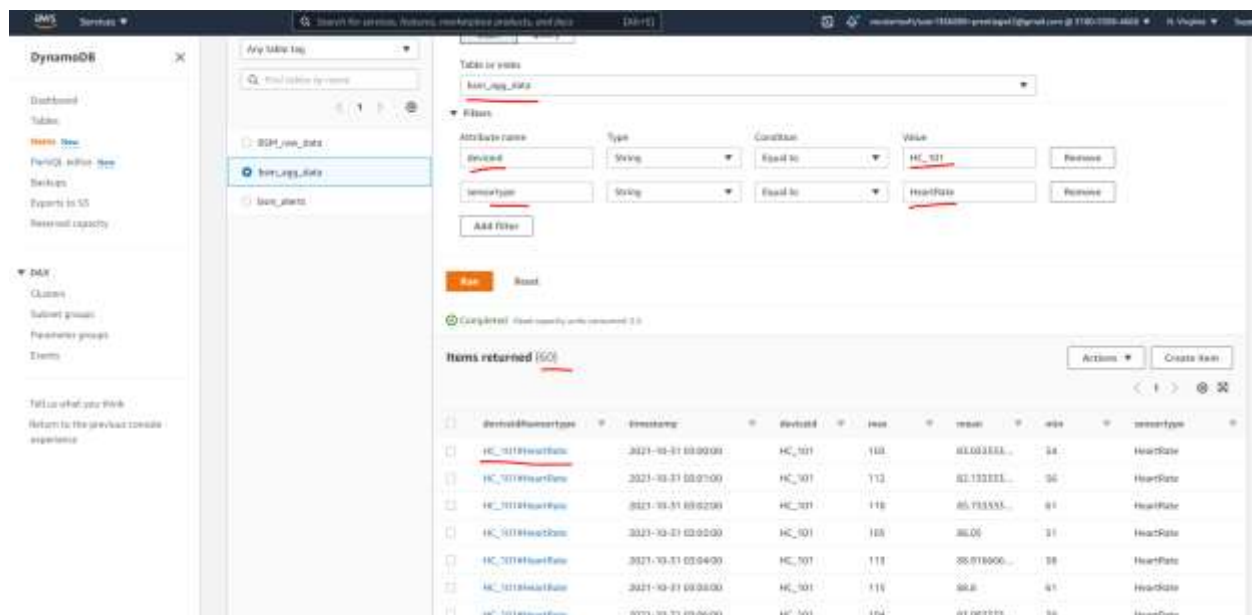
The table has composite partition key of 'deviceid#sensortype' and sort key of 'timestamp'



The screenshot shows the AWS DynamoDB console interface. On the left, the 'DynamoDB' sidebar is visible with options like Dashboard, Tables, Items, etc. The main panel displays the 'bsm\_agg\_data' table. The 'Table or index' dropdown is set to 'bsm\_agg\_data'. Below it, the 'Items' section shows a list of 360 items. The table has the following columns: deviceid#sensortype, timestamp, deviceid, heart, mean, min, and sensortype. The data is filtered to show items for device HC\_101 and sensor type Temperature.

deviceid#sensortype	timestamp	deviceid	heart	mean	min	sensortype
HC_101#Temperature	2021-10-31 03:00:00	HC_101	75.1	87.825	75.5	Temperature
HC_101#Temperature	2021-10-31 03:01:00	HC_101	100.8	100.0586	98.4	Temperature
HC_101#Temperature	2021-10-31 03:02:00	HC_101	101.3	88.815	97.3	Temperature
HC_101#Temperature	2021-10-31 03:03:00	HC_101	101.1	88.625	93	Temperature
HC_101#Temperature	2021-10-31 03:04:00	HC_101	101.4	88.425	97.6	Temperature
HC_101#Temperature	2021-10-31 03:05:00	HC_101	99.6	88.85	97	Temperature
HC_101#Temperature	2021-10-31 03:06:00	HC_101	100.2	88.375	96.8	Temperature

HeartRate data – 60 records for HeartRate aggregation inserted into the bsm\_agg\_data table as calculated with start time '10/31/2021 03:00:00' and end time '10/31/2021 04:00:00' for device HC\_101. Similar created for HC\_102.



The screenshot shows the AWS DynamoDB console interface. On the left, the 'DynamoDB' sidebar is visible. The main panel displays the 'bsm\_agg\_data' table. The 'Table or index' dropdown is set to 'bsm\_agg\_data'. Below it, the 'Items' section shows a list of 60 items. The table has the following columns: deviceid#sensortype, timestamp, deviceid, heart, mean, min, and sensortype. The data is filtered to show items for device HC\_101 and sensor type HeartRate.

deviceid#sensortype	timestamp	deviceid	heart	mean	min	sensortype
HC_101#HeartRate	2021-10-31 03:00:00	HC_101	110	81.03333...	84	HeartRate
HC_101#HeartRate	2021-10-31 03:01:00	HC_101	113	82.13333...	86	HeartRate
HC_101#HeartRate	2021-10-31 03:02:00	HC_101	116	85.73333...	89	HeartRate
HC_101#HeartRate	2021-10-31 03:03:00	HC_101	108	86.05	81	HeartRate
HC_101#HeartRate	2021-10-31 03:04:00	HC_101	118	88.91666...	88	HeartRate
HC_101#HeartRate	2021-10-31 03:05:00	HC_101	119	88.8	85	HeartRate
HC_101#HeartRate	2021-10-31 03:06:00	HC_101	104	81.03333...	80	HeartRate

Temperature data - 60 records for Temperature aggregation inserted into the bsm\_agg\_data table as calculated with start time '10/29/2021 18:37:00' and end time '10/29/2021 19:37:00' for device HC\_101. Similar created for HC\_102.

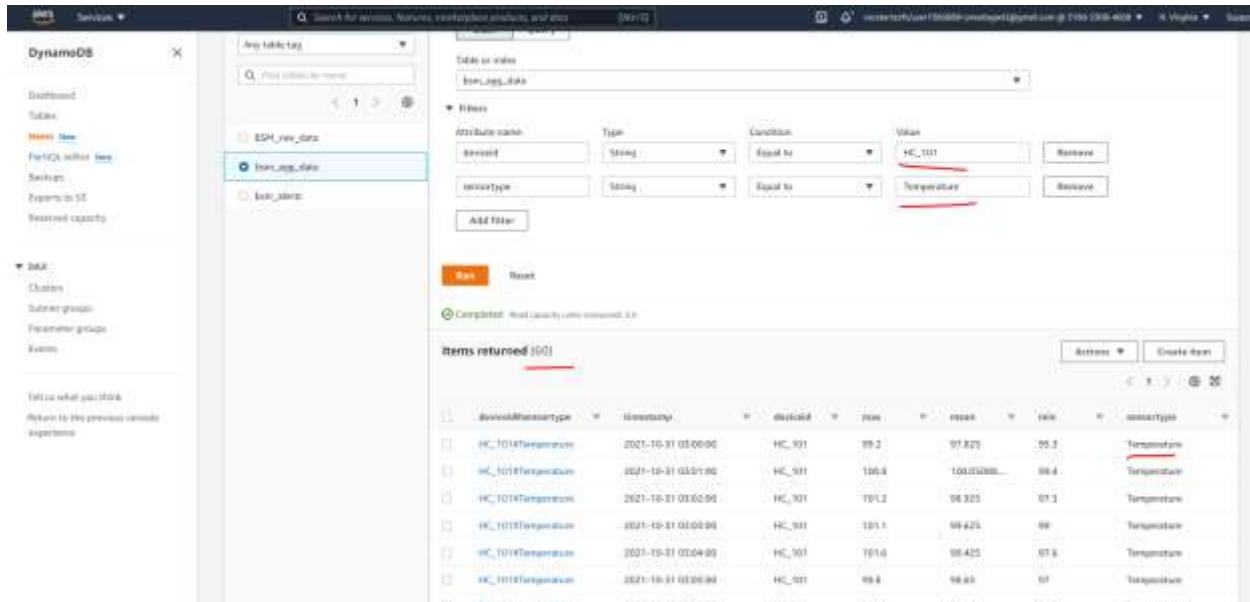


Table: bsm\_agg\_data

Filters:

- deviceid: String, Equal to, HC\_101
- sensorType: String, Equal to, Temperature

Items returned: 60

deviceid	sensorType	timestamp	min	max	avg	sensorType
HC_101	Temperature	2021-10-31 03:00:00	99.2	99.825	99.5	Temperature
HC_101	Temperature	2021-10-31 03:01:00	100.5	100.925	99.4	Temperature
HC_101	Temperature	2021-10-31 03:02:00	101.2	101.325	97.1	Temperature
HC_101	Temperature	2021-10-31 03:03:00	101.1	99.425	99	Temperature
HC_101	Temperature	2021-10-31 03:04:00	101.6	99.425	97.8	Temperature
HC_101	Temperature	2021-10-31 03:05:00	99.8	98.85	97	Temperature

SPO2 data - 60 records for SPO2 aggregation inserted into the bsm\_agg\_data table as calculated with start time '10/29/2021 18:37:00' and end time '10/29/2021 19:37:00' for device HC\_101. Similar created for HC\_102.

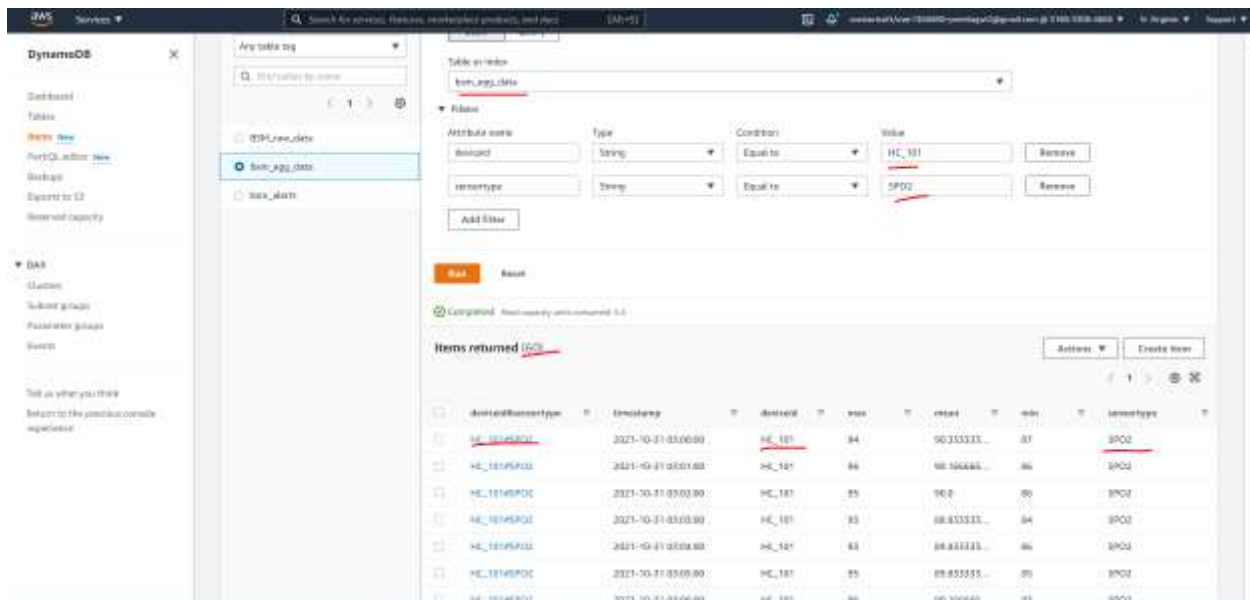


Table: bsm\_agg\_data

Filters:

- deviceid: String, Equal to, HC\_101
- sensorType: String, Equal to, SPO2

Items returned: 60

deviceid	sensorType	timestamp	min	max	avg	sensorType
HC_101	SPO2	2021-10-31 03:00:00	94	90.333333	97	SPO2
HC_101	SPO2	2021-10-31 03:01:00	86	90.966666	86	SPO2
HC_101	SPO2	2021-10-31 03:02:00	85	90.8	80	SPO2
HC_101	SPO2	2021-10-31 03:03:00	83	88.833333	84	SPO2
HC_101	SPO2	2021-10-31 03:04:00	83	88.833333	86	SPO2
HC_101	SPO2	2021-10-31 03:05:00	85	89.855555	80	SPO2

bsm\_alerts.csv