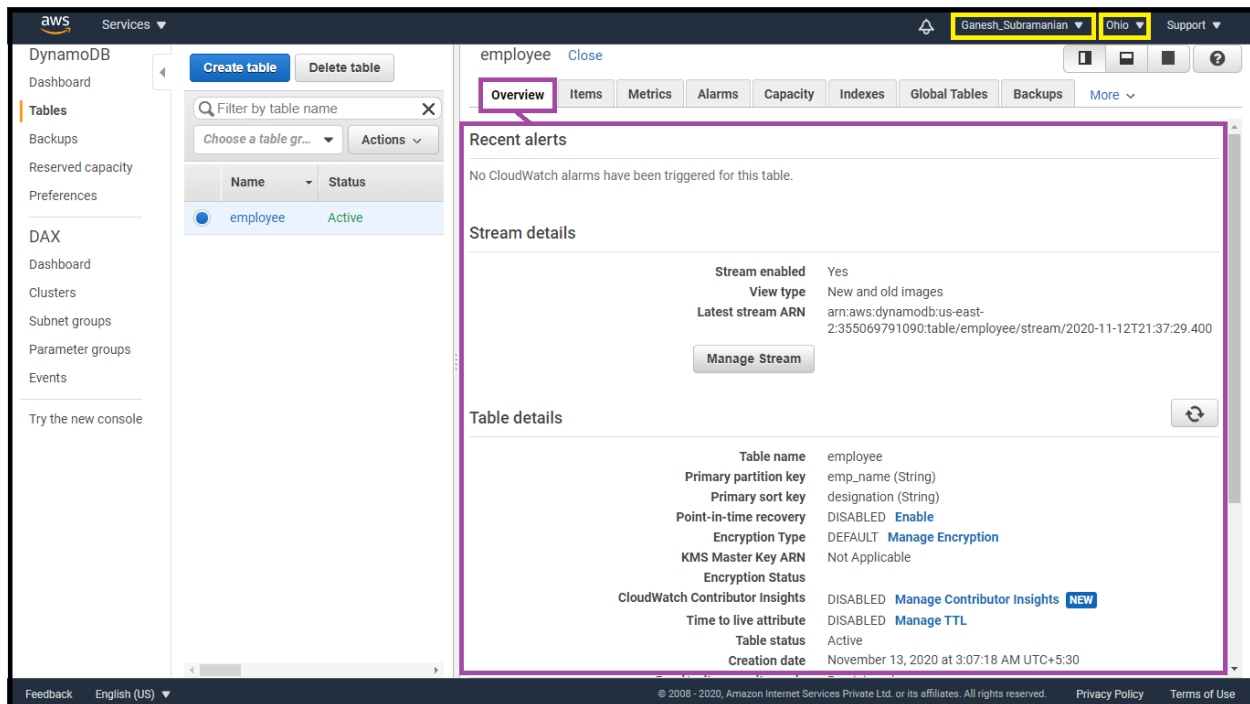


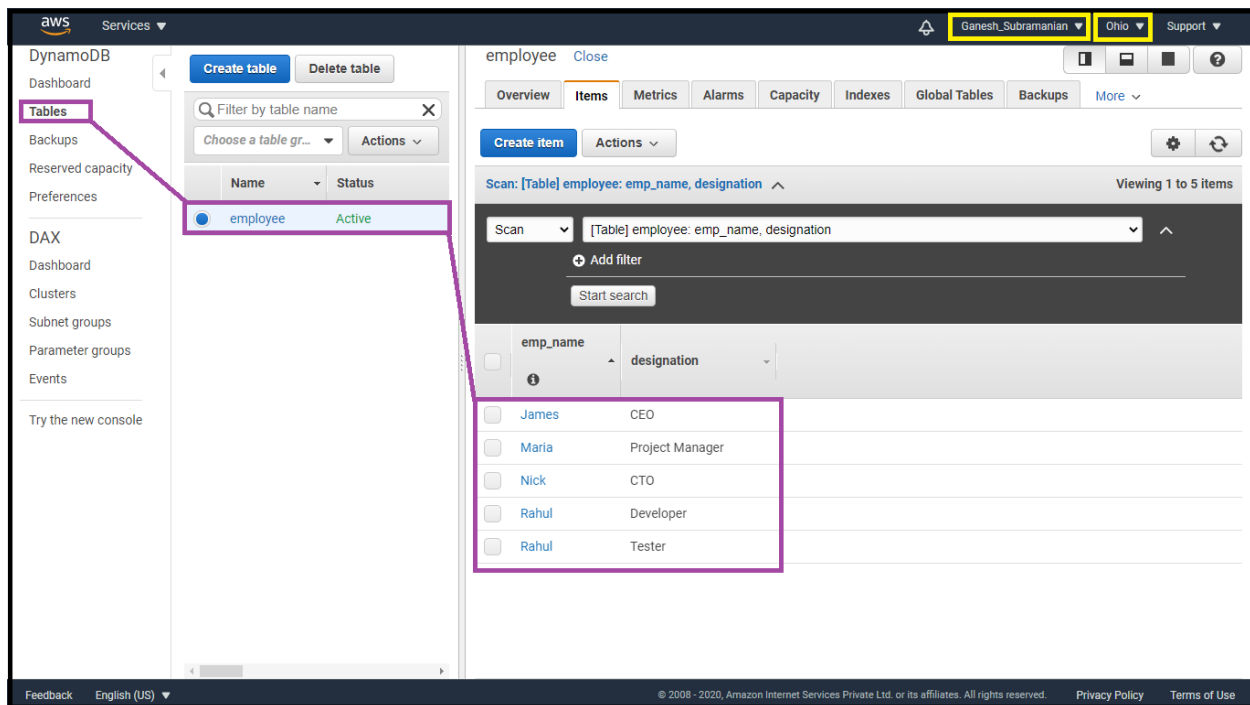
Question 1:

Task 1: Create a dynamo DB table with minimum two disaster recovery zones and verify replication.

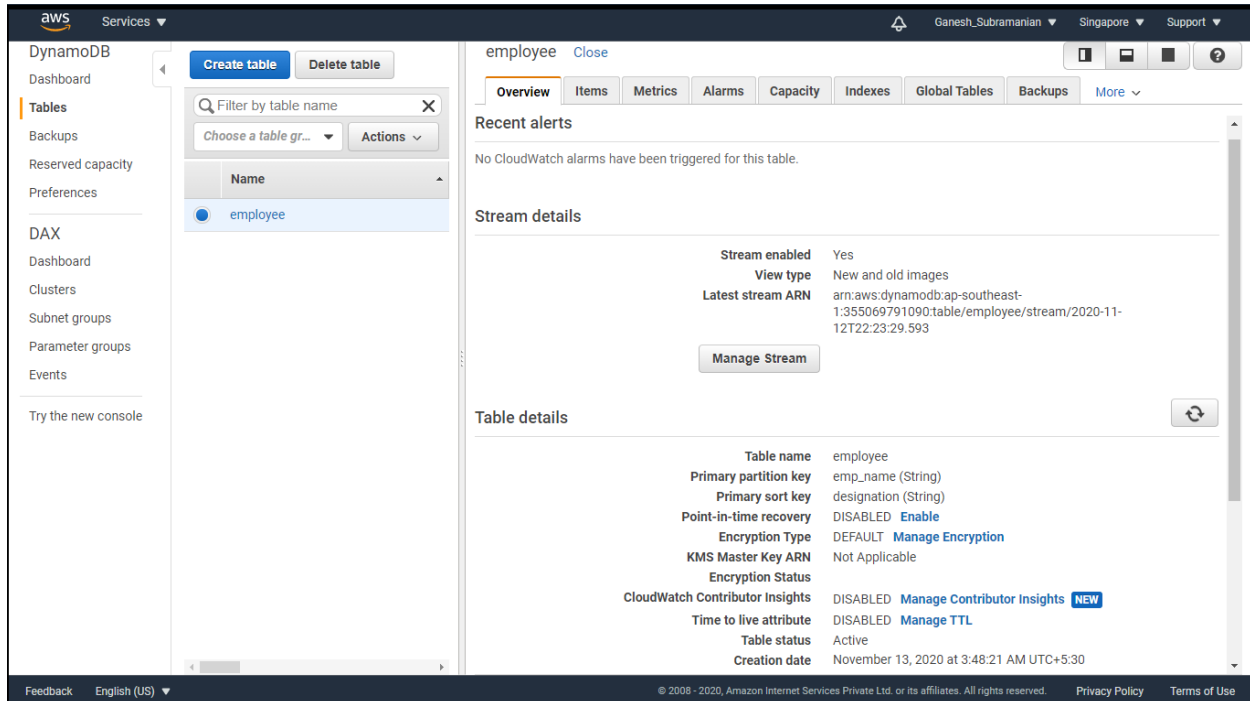
Screenshot 1: Disaster recovery regions with the table



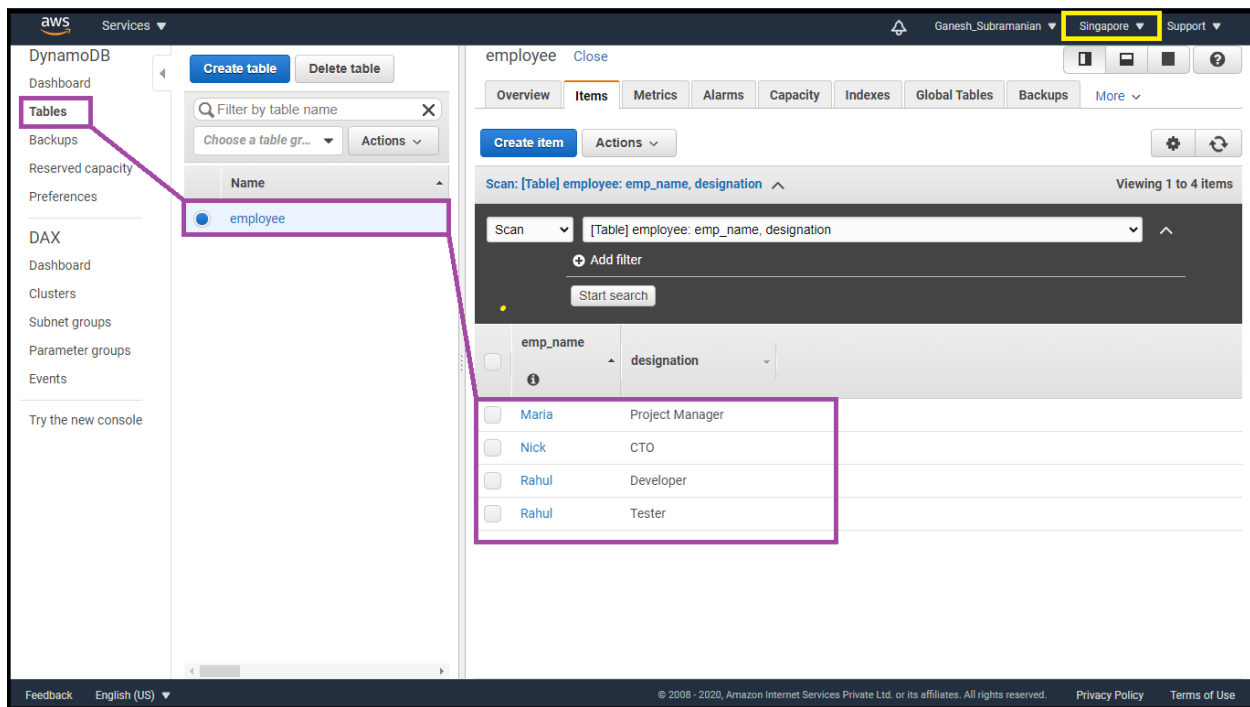
⇒ Disaster Recovery Regions Created with Table Details: - “**employee**” table created in **Ohio** Region (Home) with three disaster recovery regions i.e. **Sydney** and **Singapore**



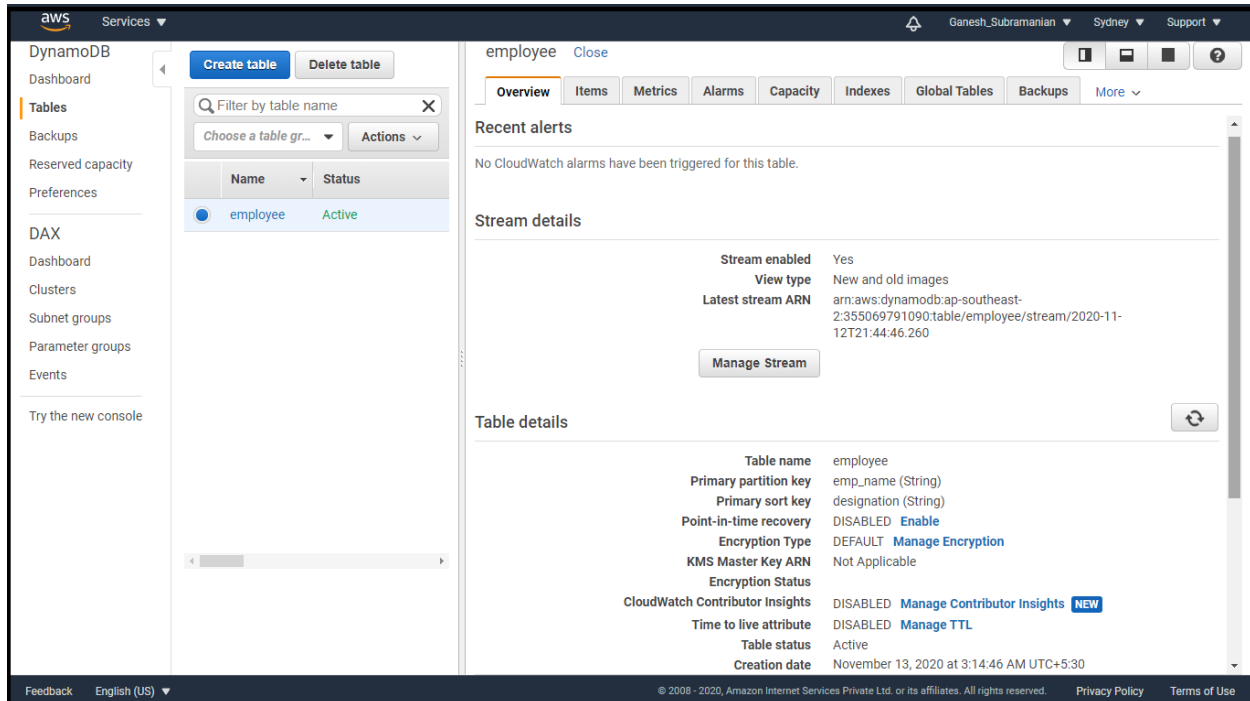
⇒ Disaster Recovery Regions Created with Table Details: - “employee” table created in **Ohio** Region (Home) with three disaster recovery regions i.e. **Sydney** and **Singapore**



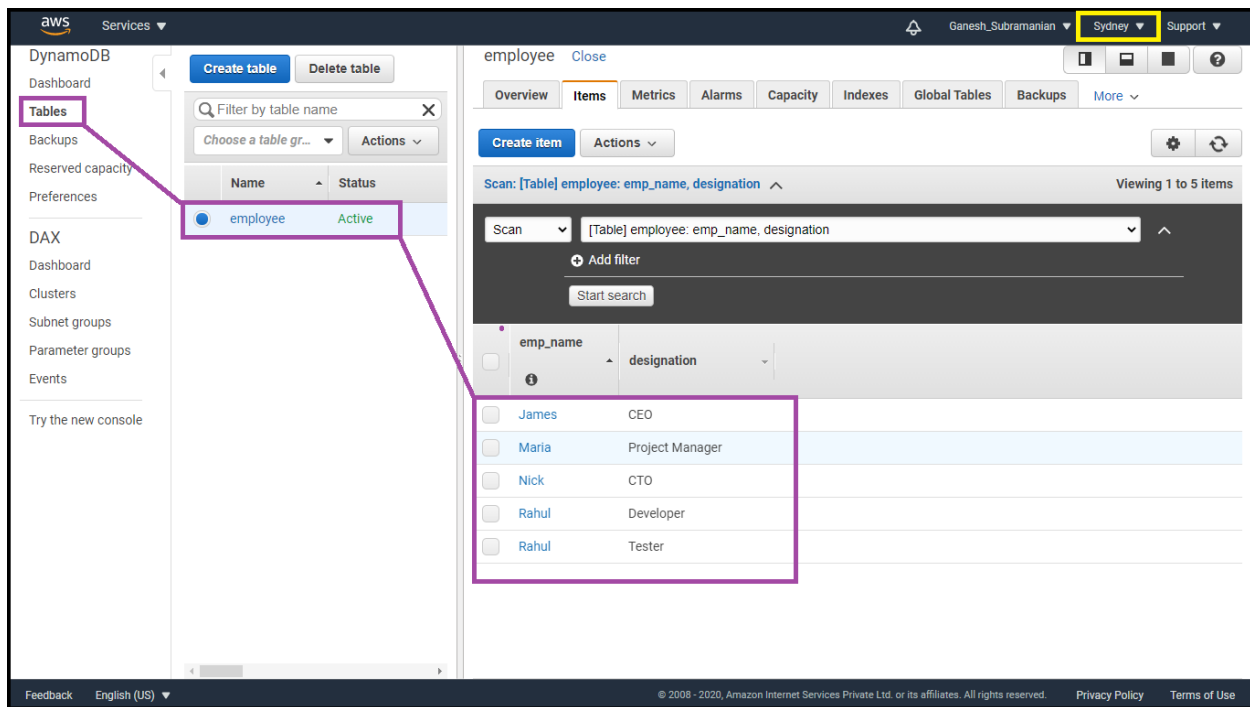
⇒ Disaster Recovery Regions Created with Table Details: - “employee” table created in **Ohio** Region (Home) with three disaster recovery regions i.e. **Sydney** and **Singapore**



⇒ Disaster Recovery Regions Created with Table Details: - “**employee**” table created in **Ohio** Region (Home) with three disaster recovery regions i.e. **Sydney** and **Singapore**

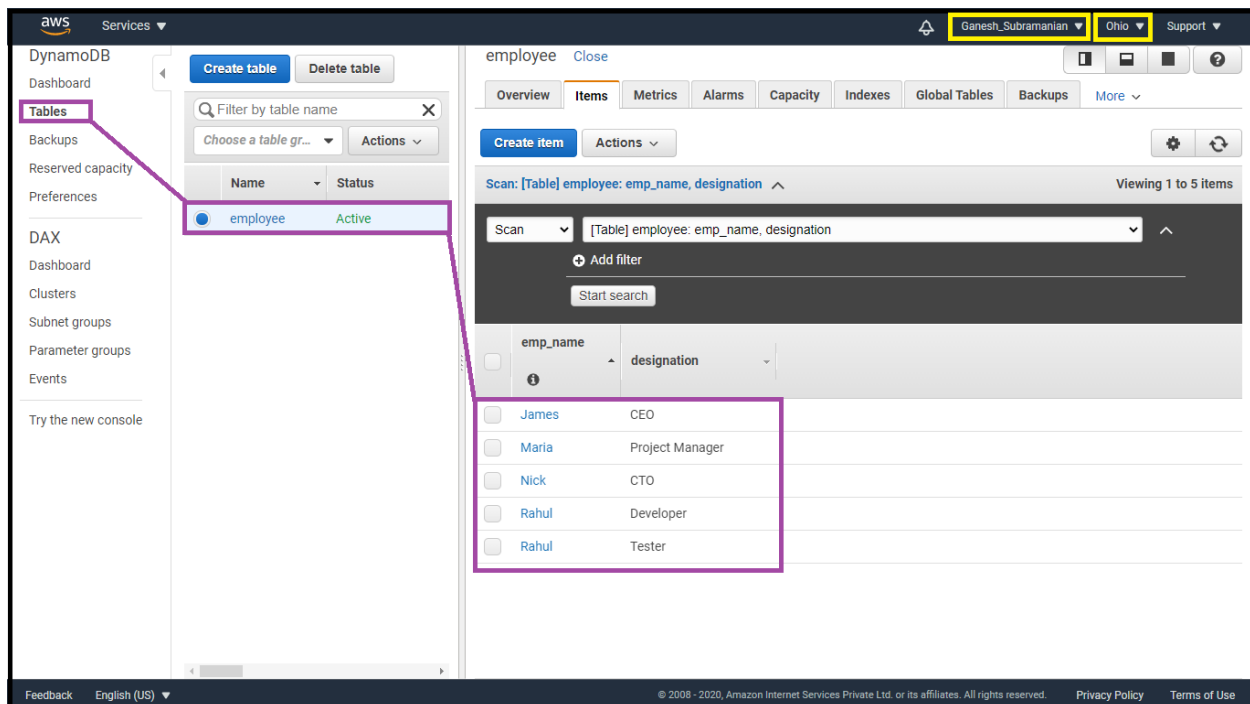


⇒ Disaster Recovery Regions Created with Table Details: - “**employee**” table created in **Ohio** Region (Home) with three disaster recovery regions i.e. **Sydney** and **Singapore**

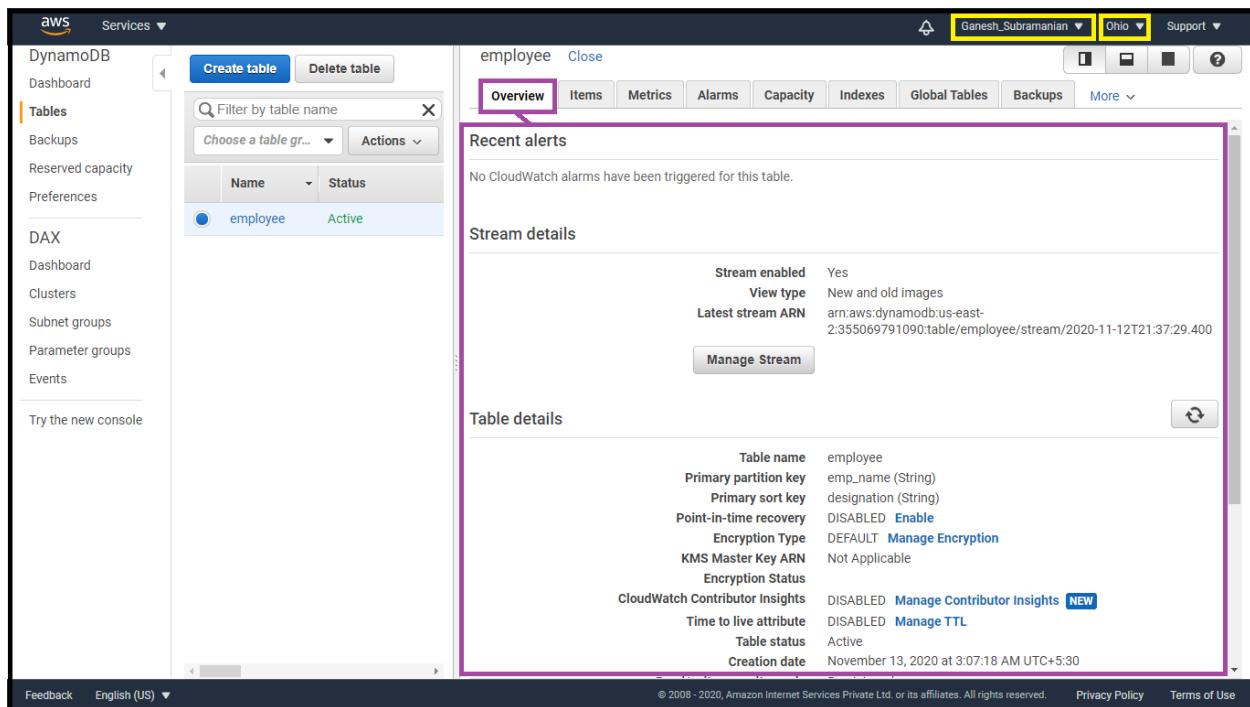


⇒ Disaster Recovery Regions Created with Table Details: - “**employee**” table created in **Ohio** Region (Home) with three disaster recovery regions i.e. **Sydney** and **Singapore**

Screenshot 2: Home region with all items displayed

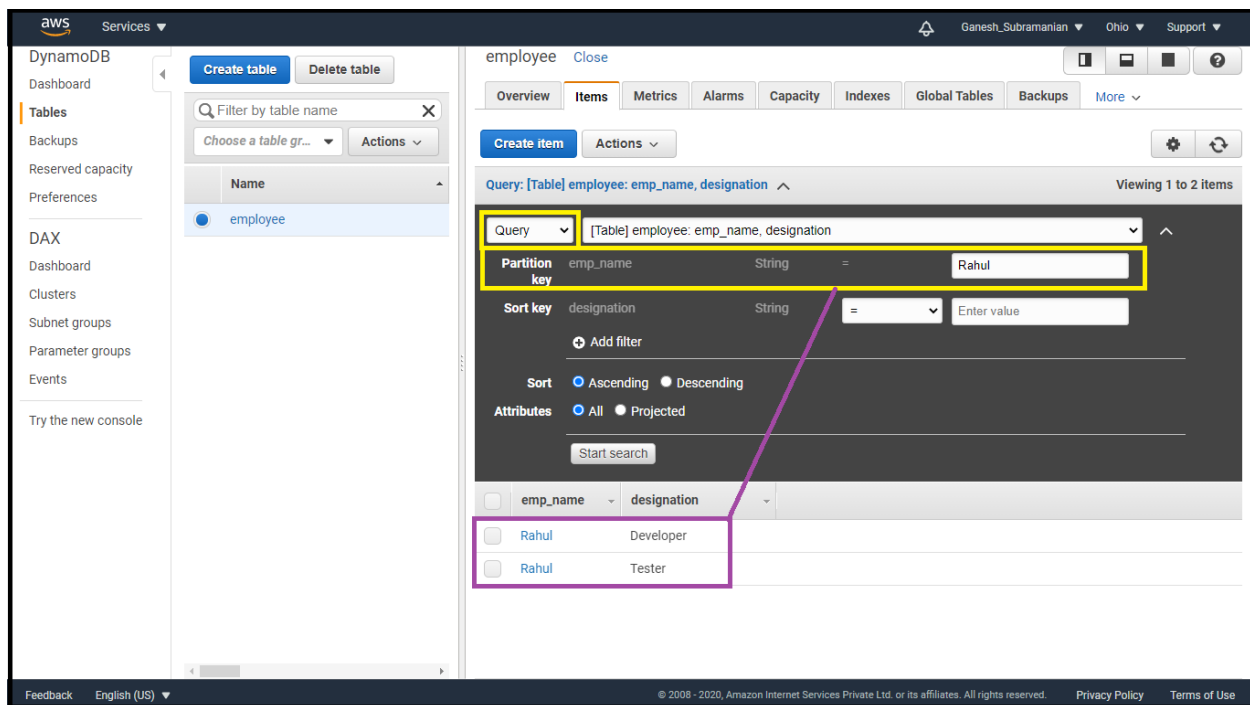


⇒ Home region with all items displayed: - Home Region **Ohio**, “**employee**” table created and all details displayed

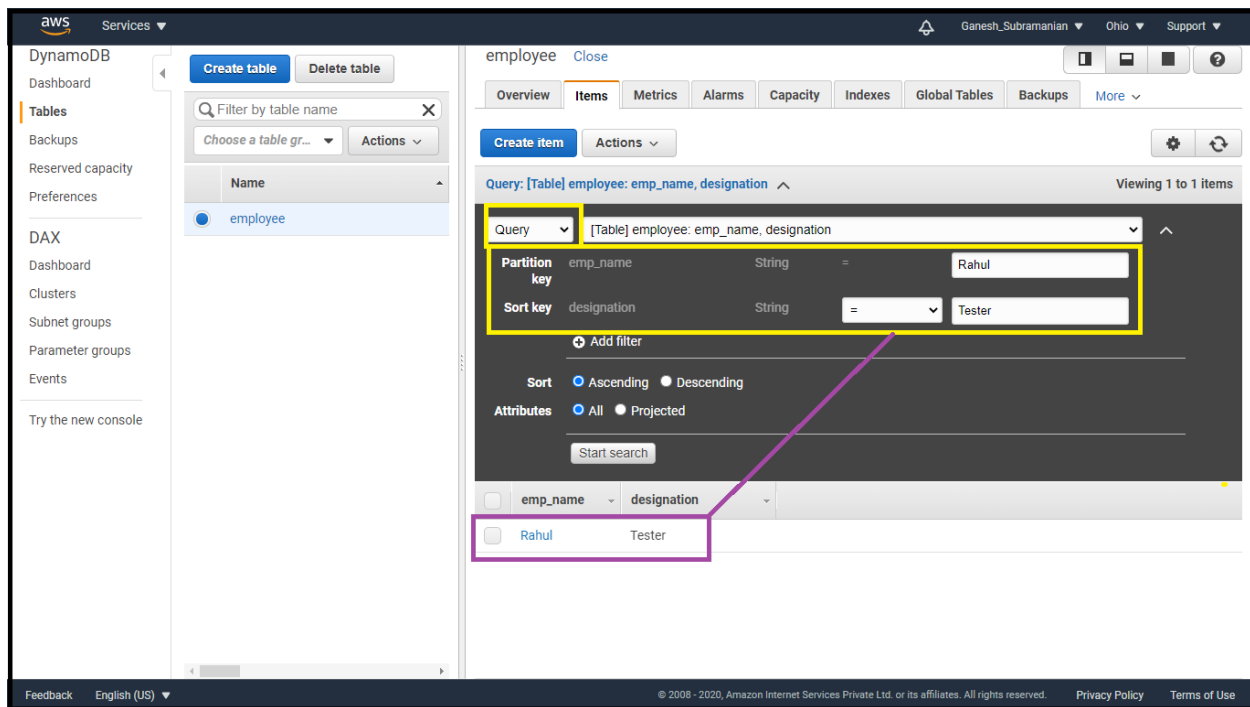


⇒ Home region with all items displayed: - Home Region **Ohio** and all required Overview details displayed

Screenshot 3: Use query to fetch few items

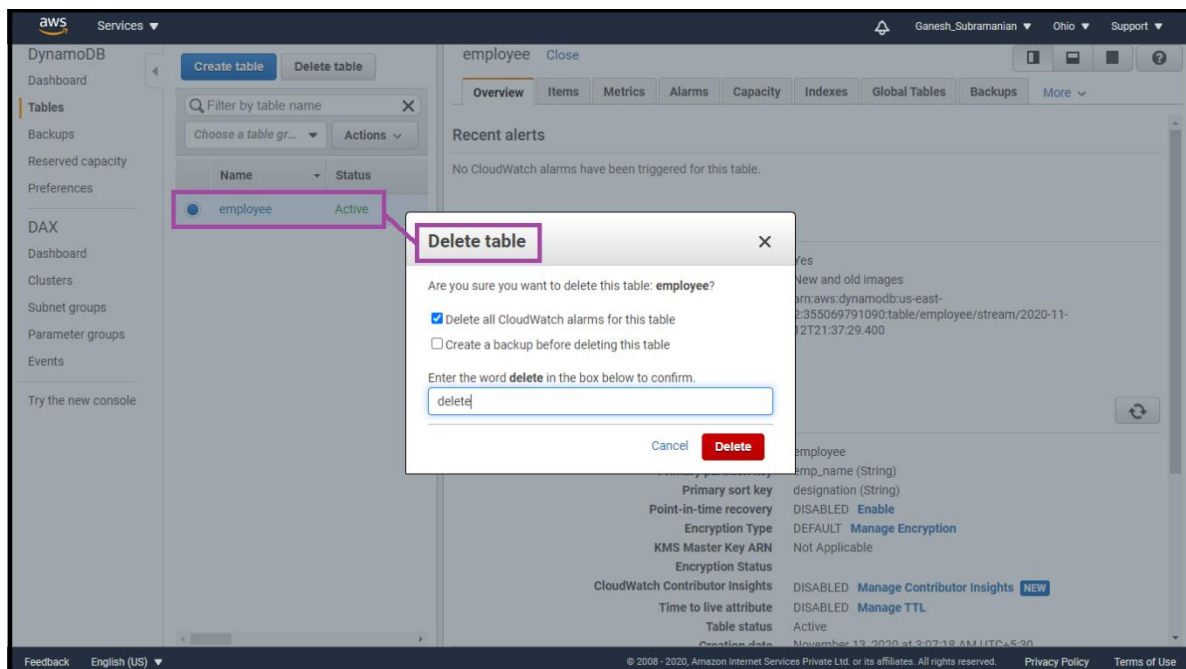


⇒ Query to fetch items: - where Data is fetched based on **Partition key** only. i.e. “**emp_name = Rahul**” and it fetches only Two data based on the available details in the DB

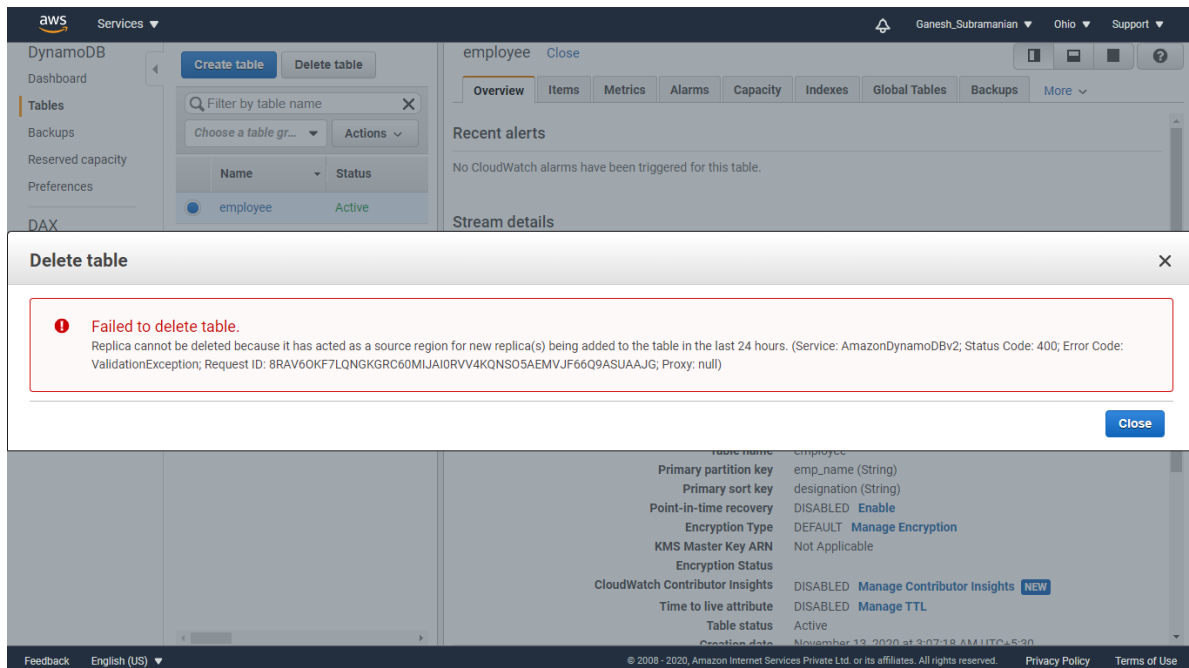


⇒ Query to fetch items: - where Data is fetched based on **Partition key** and Sort key i.e. **“emp_name = Rahul”** and **“designation = Tester”** and it fetches only One data based on the available details in the DB

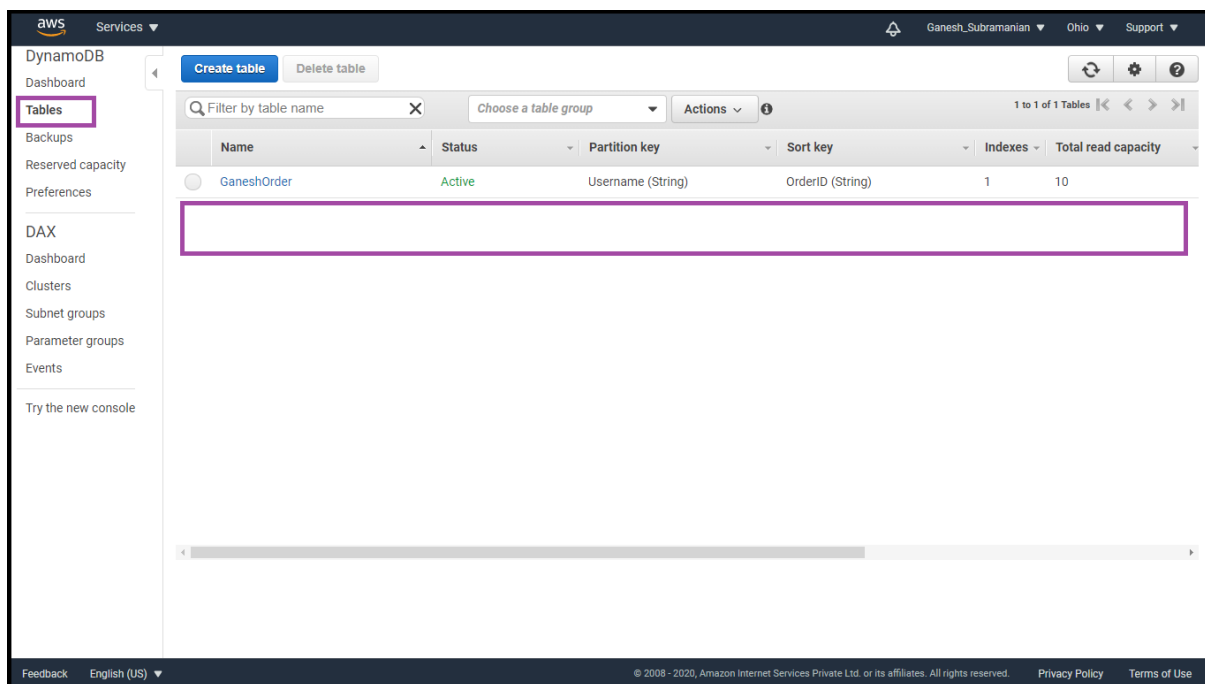
Screenshot 4: deletion and verification



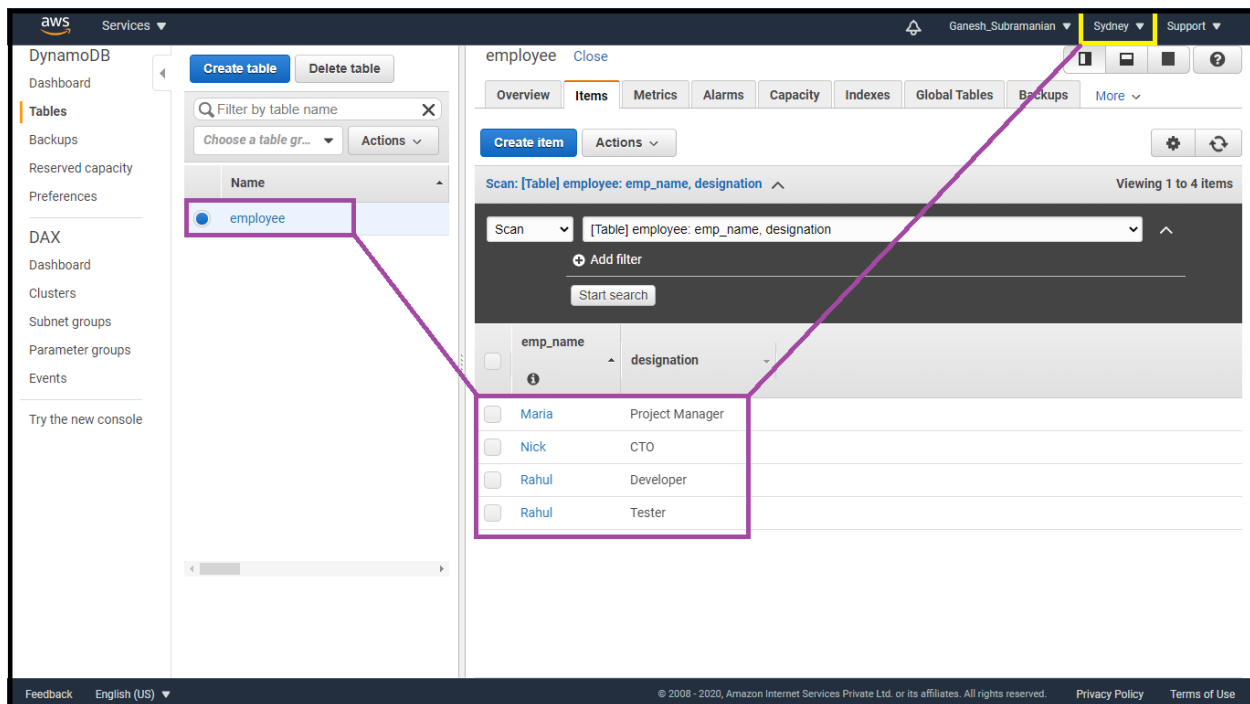
⇒ Deletion of DB: - DB is Deleted from the **Ohio (Home) Region**



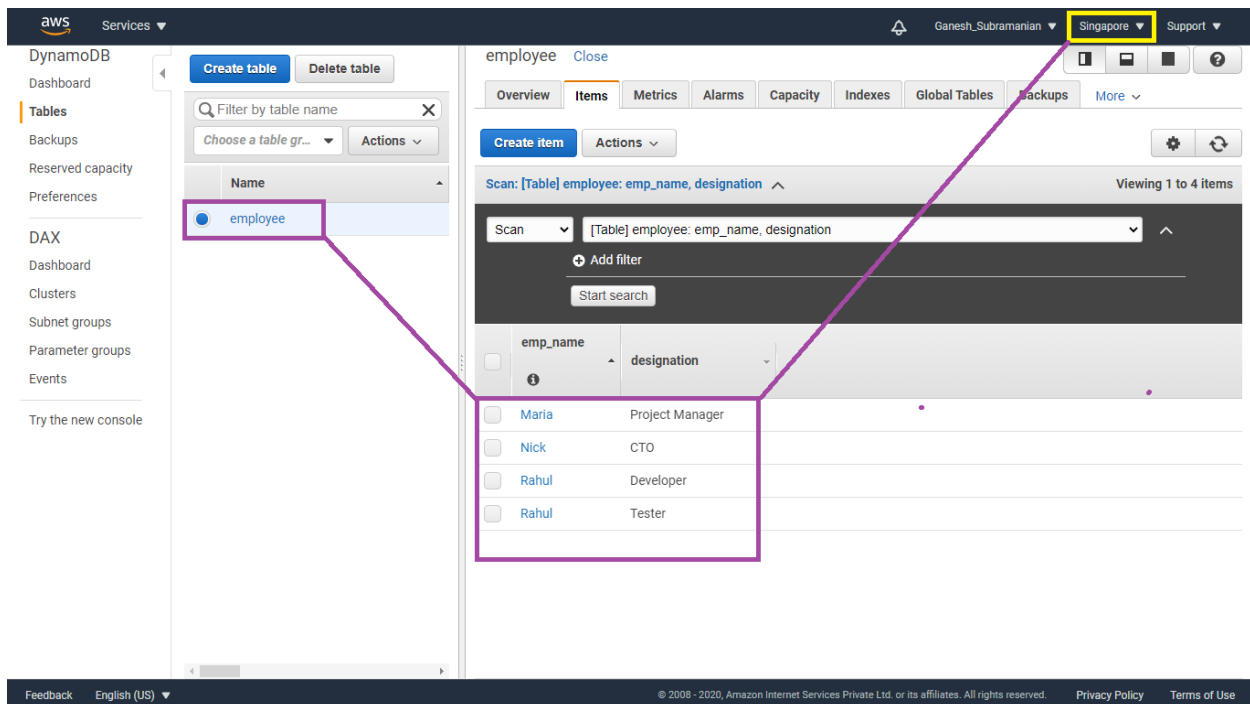
⇒ Deletion of DB: - A Warning Message is being displayed one trying to delete the DB from the **Ohio (Home) Region** due to less than 24 hours.



⇒ Successful Deletion of DB after 24 hours: - The Employee Table is completely deleted and is not available in the Ohio Region.



⇒ Deletion of Details: - DB is Still intact in **Sydney** region



⇒ Deletion of Details: - DB is Still intact in **Singapore** region

Task 2: Creating a dynamo DB table with global secondary indexes and fetching data using global secondary indexes.

Screenshot 1: Table with its items displayed

The screenshot shows the AWS DynamoDB console. On the left, the 'Tables' tab is selected in the left-hand navigation pane. The main panel displays the 'GaneshOrder' table. The 'Items' tab is selected, showing a scan of the 'ReturnDate-UserAmount-index'. The scan results are displayed in a table with the following columns: Username, OrderID, ReturnDate, and UserAmount. The items are:

Username	OrderID	ReturnDate	UserAmount
Onur	2333345453	20160104	410.12
Onur	2176567678	20150103	419.45
Tolga	3456123478	20180106	451.78
Tolga	1112323234	20170105	333.12
Yohan	2345678901	20190107	250.89

Screenshot 2: creating global secondary index

The screenshot shows the AWS DynamoDB console. On the left, the 'Tables' tab is selected in the left-hand navigation pane. The main panel displays the 'GaneshOrder' table. The 'Indexes' tab is selected, showing the 'Create index' button. The index details are shown in a table with the following columns: Name, Status, Type, Partition key, Sort key, Attributes, and Read capacity. The index is:

Name	Status	Type	Partition key	Sort key	Attributes	Read capacity
ReturnDate-UserAmount	Active	GSI	ReturnDate (String)	UserAmount (String)	ALL	5

⇒ **Creation of Global Secondary Index:** - The highlighted section shows the successful creation of GSI

Screenshot 3: Scan with Global Secondary Index

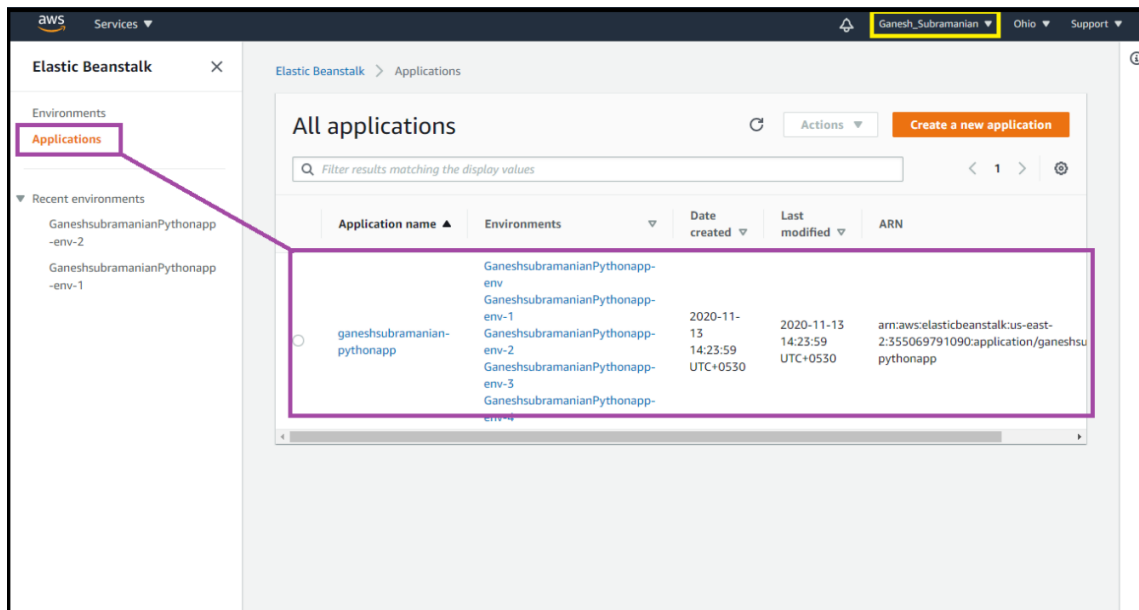
The screenshot displays the AWS Management Console interface for a DynamoDB table named 'GaneshOrder'. The left sidebar shows the 'Tables' menu highlighted. The main panel shows the 'Scan' operation for the table. The scan is configured with the GSI '[Index] ReturnDate-UserAmount-index: ReturnDate, UserAmount' and a filter for 'UserAmount' between 330 and 400. The scan results show two items: Yohan and Tolga.

Username	OrderID	ReturnDate	UserAmount
Yohan	4512309823	20200108	398.56
Tolga	1112323234	20170105	333.12

- ⇒ **Scan with Global Secondary Index:** - The highlighted section shows the successful scanned details display on performing the Scan of **GSI** feature filter creation of GSI i.e. where UserAmount is between 330/- to 400/- which eventually displayed on two rows of data.

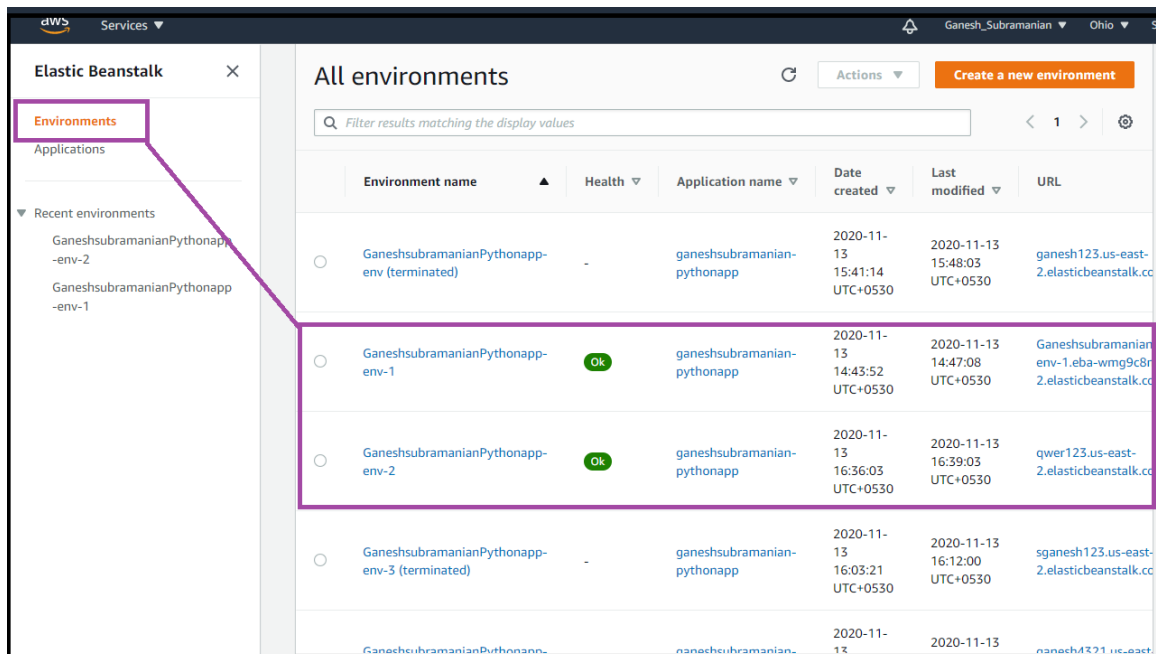
Task 3: Deploying a python application in elastic beanstalk

Screenshot 1: Application page

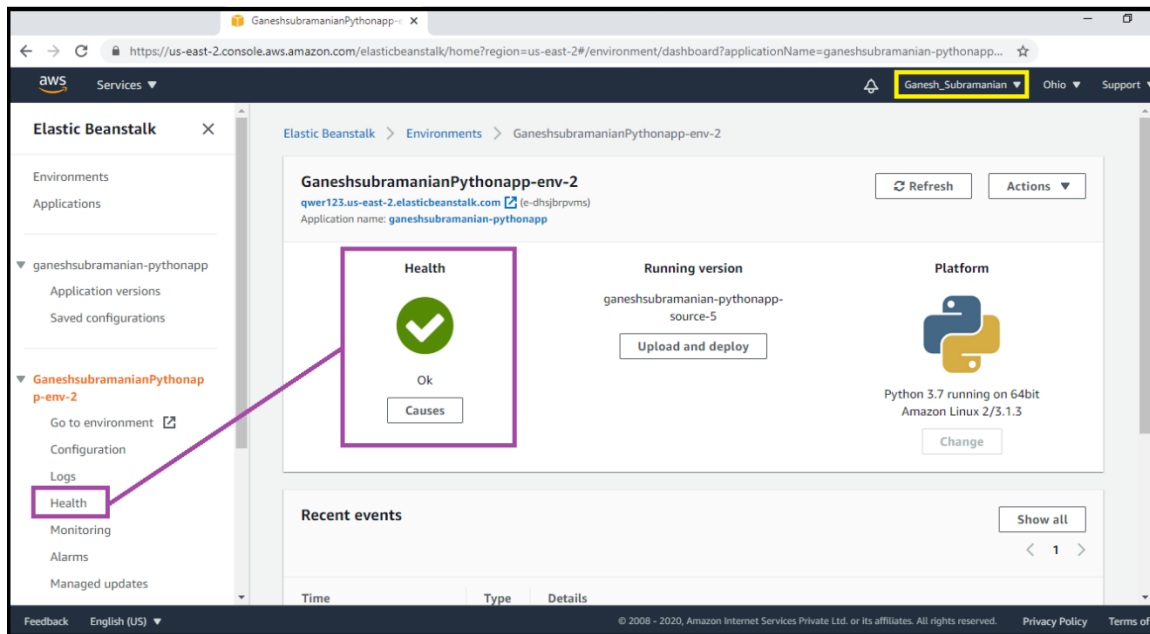


⇒ New Elastic Beanstalk Application creation: - “**ganeshsubramanianpythonapp**” has been created

Screenshot 2: Env list page

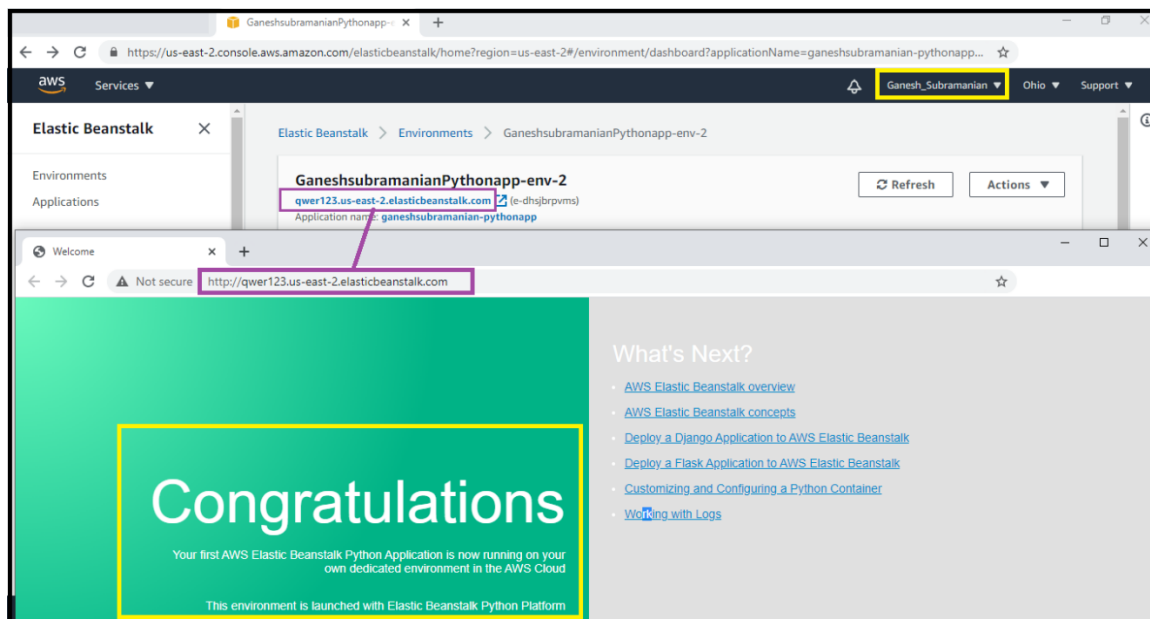


Screenshot 3: Env health status page



⇒ Health Status is Healthy: - After uploading the python file the Health Status is Healthy as Highlighted

Screenshot 4: Web page launched using the elastic beanstalk env



⇒ Launching of Web Page using Elastic Beanstalk environment: - On Clicking on the public URL the Web Page is displayed which indicates the same has been successfully launched using Elastic Beanstalk Environment