

# Demo\_1(UsingConsole):

Creating Table :

The screenshot shows the AWS Management Console interface for creating a new table in Amazon DynamoDB. The top navigation bar includes the AWS logo, 'Services', and 'Resource Groups'. The left-hand navigation pane lists various DynamoDB features: Dashboard, Tables, Backups, Reserved capacity, DAX, Clusters, Subnet groups, Parameter groups, and Events. The main content area is titled 'Create table' and includes a description of Amazon DynamoDB, a 'Create table' button, a 'Recent alerts' section (showing no alerts), and a 'Total capacity for US East (N. Virginia)' section with a table of capacity metrics. Below this is a 'Service health' section with a table showing the current status of the service as 'operating normally'.

**Create table**

Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability.

[Create table](#)

**Recent alerts**

No CloudWatch alarms have been triggered. [View all in CloudWatch](#)

**Total capacity for US East (N. Virginia)**

<b>Provisioned read capacity</b>	5	<b>Reserved read capacity</b>	0
<b>Provisioned write capacity</b>	5	<b>Reserved write capacity</b>	0

**Service health**

Current Status	Details
Amazon DynamoDB (N. Virginia)	Service is operating normally

[View complete service health details](#)

Provide:

Name of the Table

Primary Key

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name\*

Primary key\* Partition key

String

Add sort key

### Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

- Use default settings
  - No secondary indexes.
  - Provisioned capacity set to 5 reads and 5 writes.
  - Basic alarms with 80% upper threshold using SNS topic "dynamodb".
  - On-Demand Backup and Restore Enabled **NEW**

**i** You do not have the required role to enable Auto Scaling by default. Please refer to [documentation](#).

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Default settings: (Default settings will cost us around 2.91 \$ per month )

No secondary key

Read capacity units = 5

Write capacity units = 5

We can also configuring read/Write throughput(Provisioned read capacity units/Provisioned write capacity units)

## Provisioned capacity

Table

Read capacity units  Write capacity units

Estimated cost \$2.91 / month ([Capacity calculator](#))

## Auto Scaling

Read capacity  Write capacity

Same settings as read

Target utilization  %  %

Minimum provisioned capacity  units  units

Maximum provisioned capacity  units  units

Apply same settings to global secondary indexes  Apply same settings to global secondary indexes

IAM Role I authorize DynamoDB to scale capacity using the following role:

DynamoDB AutoScaling Service Linked Role

Role Name\*

a) Example of Primary Key(Partition): Creating Table Order  
Primary key is compulsory element to create Table.  
Sort part of primary Key is optional

1.

## Create DynamoDB table Tutorial ?

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

**Table name\***  i

**Primary key\*** Partition key  
  i

Add sort key

### Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

- No secondary indexes.
- Auto Scaling capacity set to 70% target utilization, at minimum capacity of 5 reads and 5 writes
- On-Demand Backup and Restore Enabled **NEW!**

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Cancel Create

2. To Create Item in Table , Click Create Item as highlighted in below screenshot.

The screenshot shows the AWS DynamoDB console interface for a table named 'Order'. The 'Items' tab is selected, and the 'Create Item' button is highlighted with a red box. The console displays a search bar for items, a filter section, and a table with one column labeled 'Order\_ID'. A tooltip at the bottom explains that an item consists of one or more attributes, each with a name, data type, and value, and that only the primary key attributes are required for reading or writing.

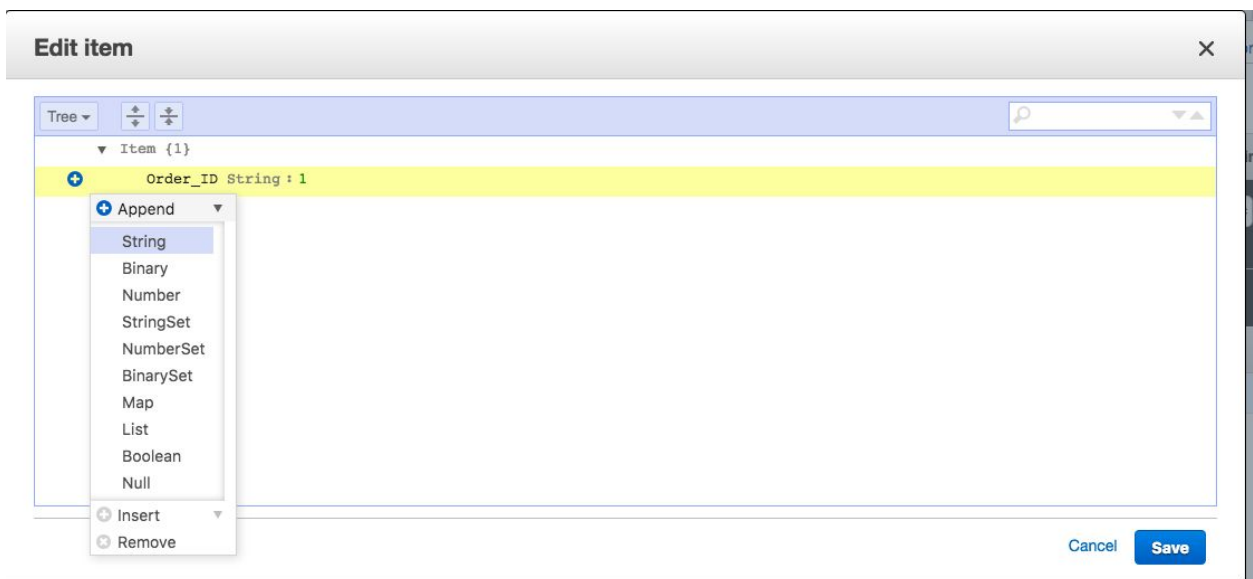
3. Below Window will appear:

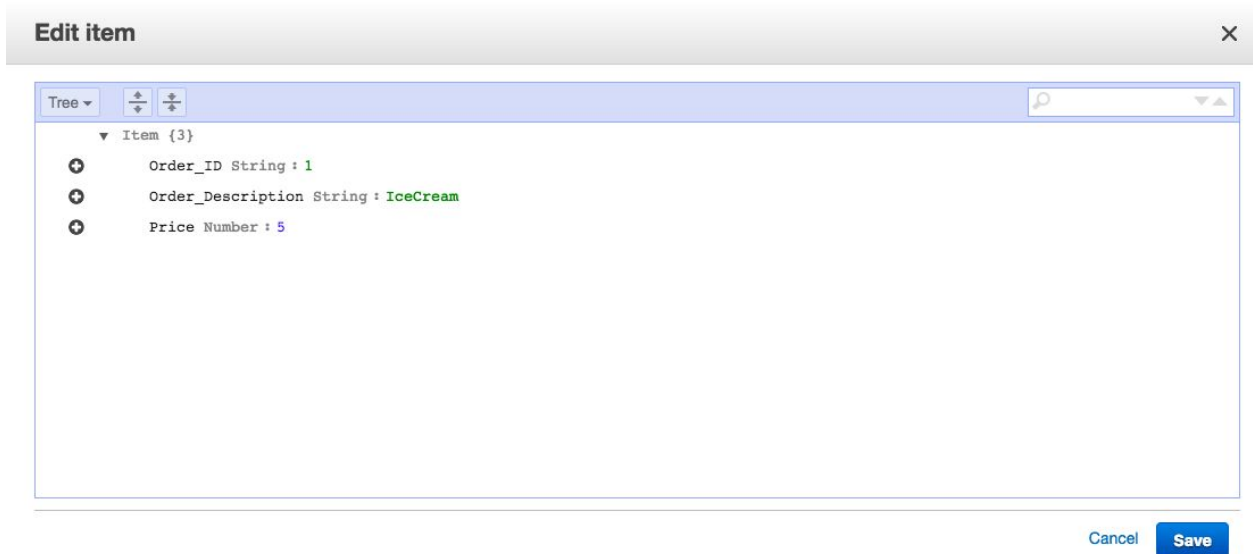
We can give Input in Tree format or Test format.

And give Value in ORder Id.

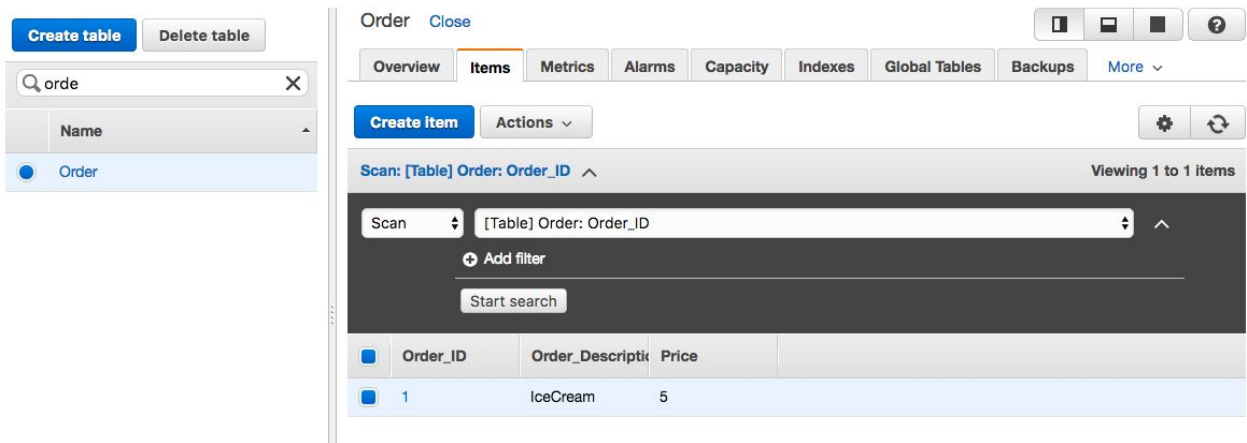


4. We can add more elements to Item:

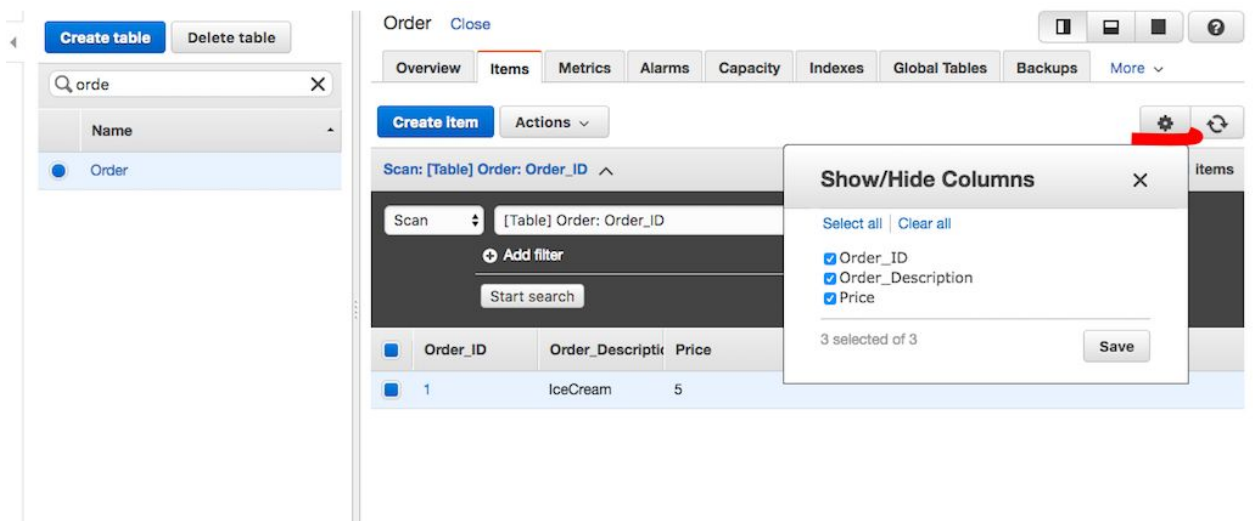




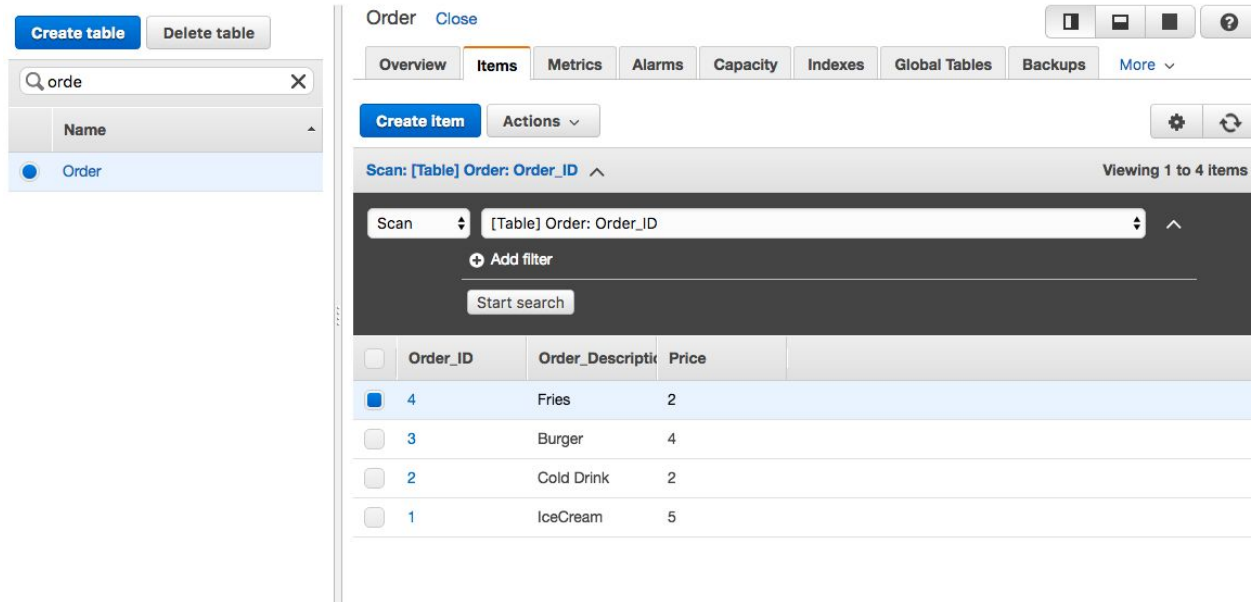
5. The table looks like this now:



6. If we click on settings we can show/hide Columns of table:



## 7. Adding few more elements in table:



The screenshot displays a database management interface. On the left, a search bar contains the text 'orde' and a dropdown menu shows 'Order' selected. The main interface shows a table named 'Order' with the following columns: 'Order\_ID', 'Order\_Descriptio', and 'Price'. The table contains four rows of data:

Order_ID	Order_Descriptio	Price
4	Fries	2
3	Burger	4
2	Cold Drink	2
1	IceCream	5

## b) Example of Primary Key(Partition+Sort):

### 1. Creating Table Project

## Create DynamoDB table

Tutorial ?

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

**Table name\***  ⓘ

**Primary key\*** Partition key

String ⓘ

Add sort key

Number ⓘ

### Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

- No secondary indexes.
- Auto Scaling capacity set to 70% target utilization, at minimum capacity of 5 reads and 5 writes
- On-Demand Backup and Restore Enabled **NEW!**

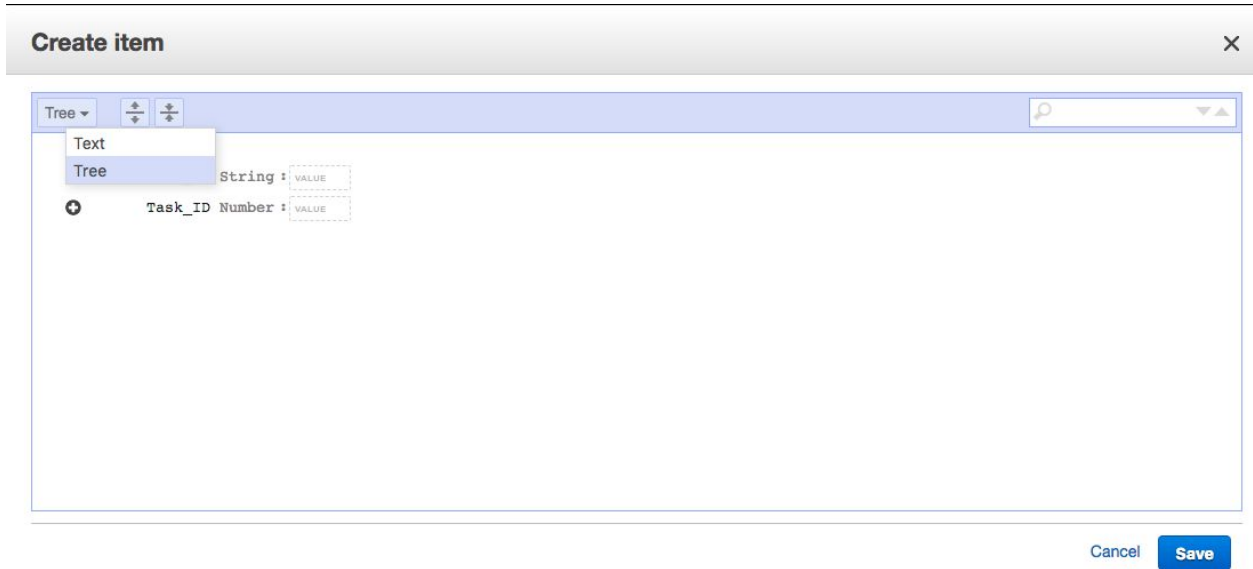
Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

## 2. Below Table is created:

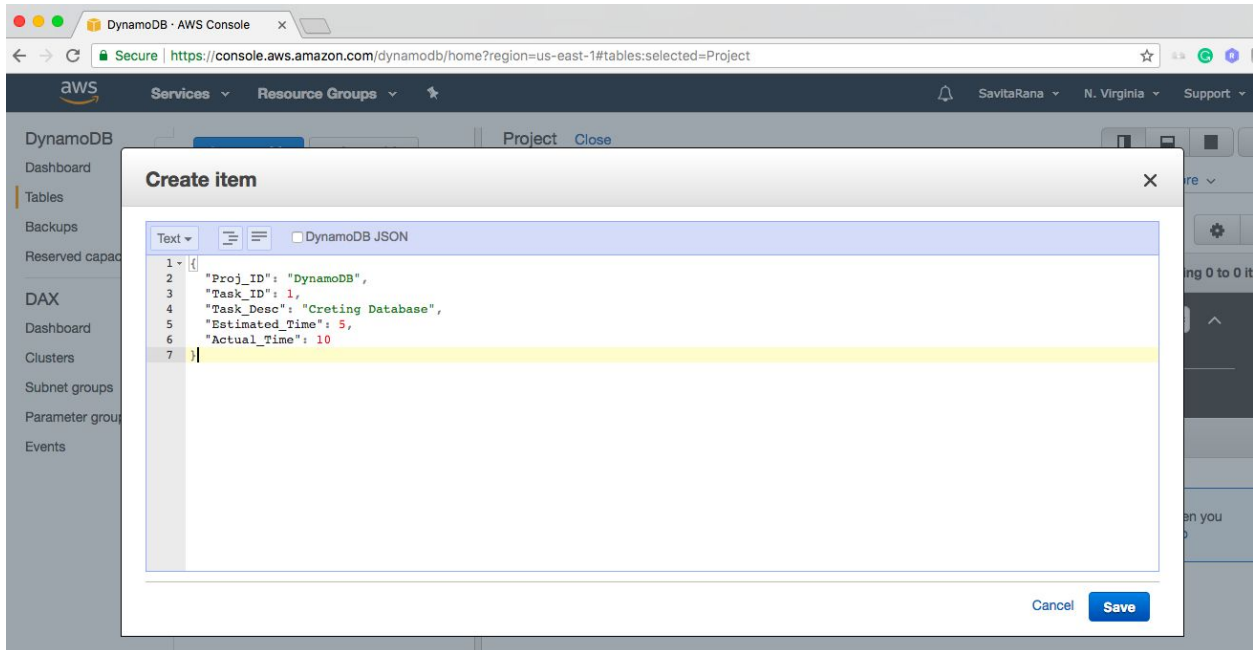
The screenshot shows the AWS DynamoDB console for a table named 'Project'. On the left, there is a search bar with 'Pro' entered and a list of results showing 'Project' selected. The main console area has a 'Project' header with a 'Close' button and navigation tabs: Overview, Items, Metrics, Alarms, Capacity, Indexes, Global Tables, Backups, and More. Below the tabs are 'Create Item' and 'Actions' buttons. A search filter is applied: 'Scan: [Table] Project: Proj\_ID, Task\_ID'. The search results show 'Viewing 0 to 0 items' and a search bar with the same filter. Below the search bar is an 'Add filter' button and a 'Start search' button. At the bottom, there is a message box: 'An item consists of one or more attributes. Each attribute consists of a name, a data type, and a value. When you read or write an item, the only attributes that are required are those that make up the primary key. More info'.

3. Again to add item we need to click "Create Item".  
And then in text window, we can add text format as well





4. There is option for Json as well



5. Thus below Item in Table is created:

The screenshot shows the AWS IAM console interface. On the left, there is a sidebar with a search bar containing 'Pro' and a list of items with 'Project' selected. The main panel shows the 'Project' details page, with the 'Items' tab selected. A table displays one item:

Proj_ID	Task_ID	Actual_Time	Estimated_Time	Task_Des
Dynamo_DB	1	10	5	Creating Database

## 6. Adding few more items in Table:

The screenshot shows the AWS IAM console interface with the 'Items' tab selected. The table now displays six items:

Proj_ID	Task_ID	Actual_Time	Estimated_Time	Task_Des
Software_Dev	3	24	12	Testing
Software_Dev	2	30	15	Software Development
Software_Dev	1	10	10	Creating Req
Dynamo_DB	3	15	10	Experiment for throughput
Dynamo_DB	2	6	5	CRUD Operations
Dynamo_DB	1	10	5	Creating Database

Number of Partitions:

By Capacity :  $\text{Total RCU}/3000 + \text{Total WCU}/1000$  By Size:

$\text{Total Size}/10\text{GB}$  Total Partitions

$\text{CEILING}(\text{MAX}(\text{Capacity}, \text{Size}))$

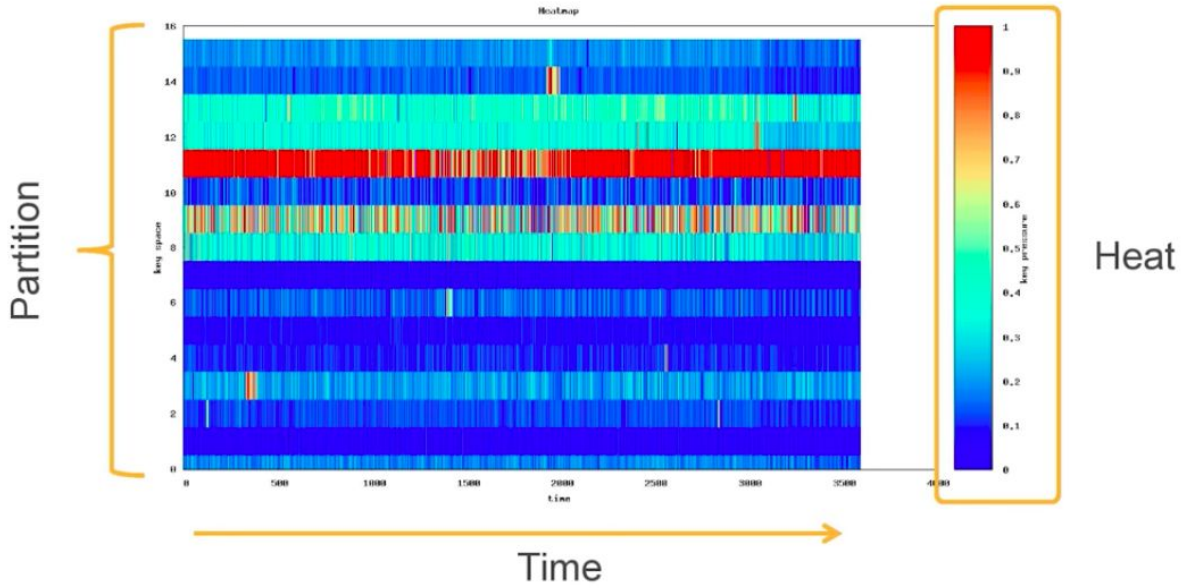
We can provide throughput for each partition

To get the most out of Dynamo DB throughput, create tables where the hash key element has a large number of distinct values, and values are requested fairly uniformly as randomly as possible.

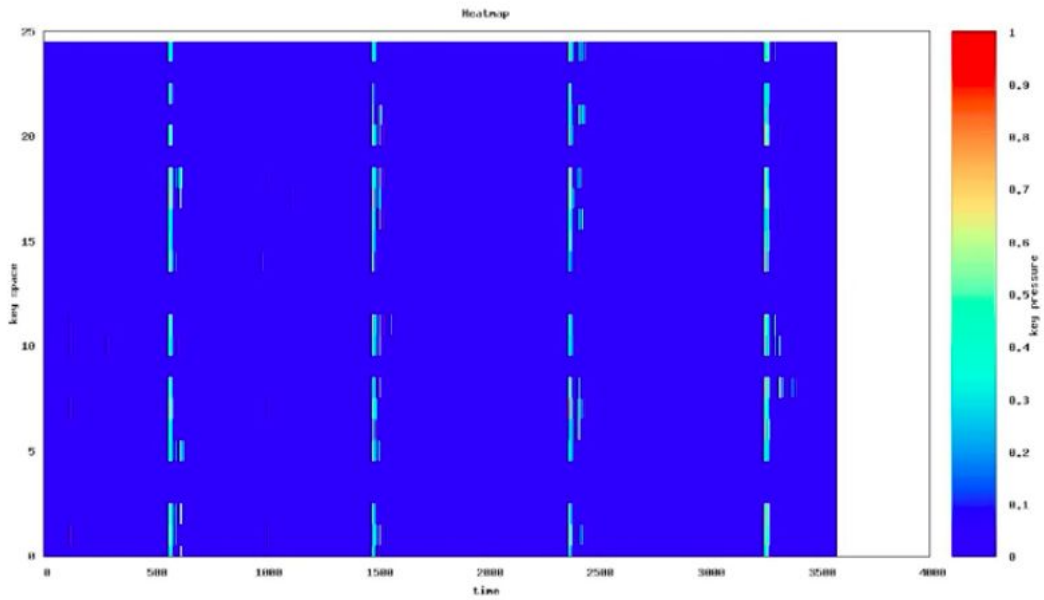
Space: access is evenly spread over the key-space

Time: requests arrive evenly spaced in time

Bad Example of Partition :

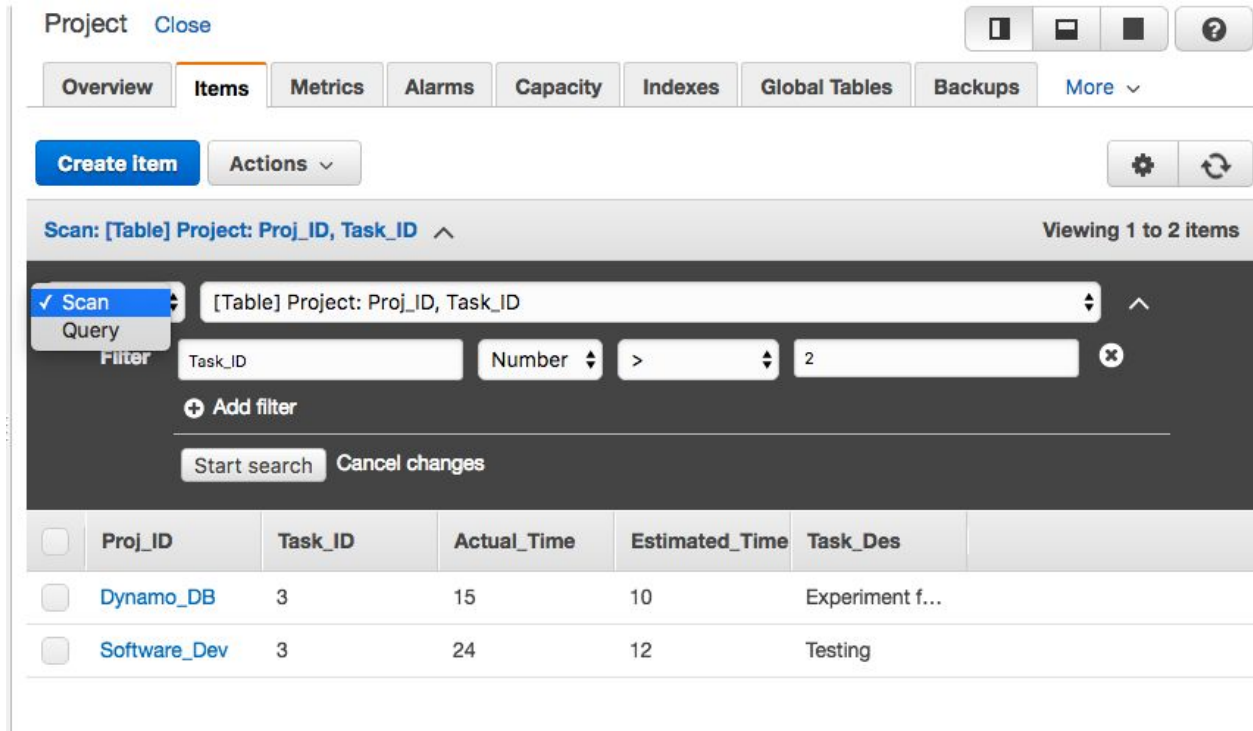


Good Example of Partition:



C. Scan and Query operation on Project Table:

## 1. Displaying Task\_Ids > 2

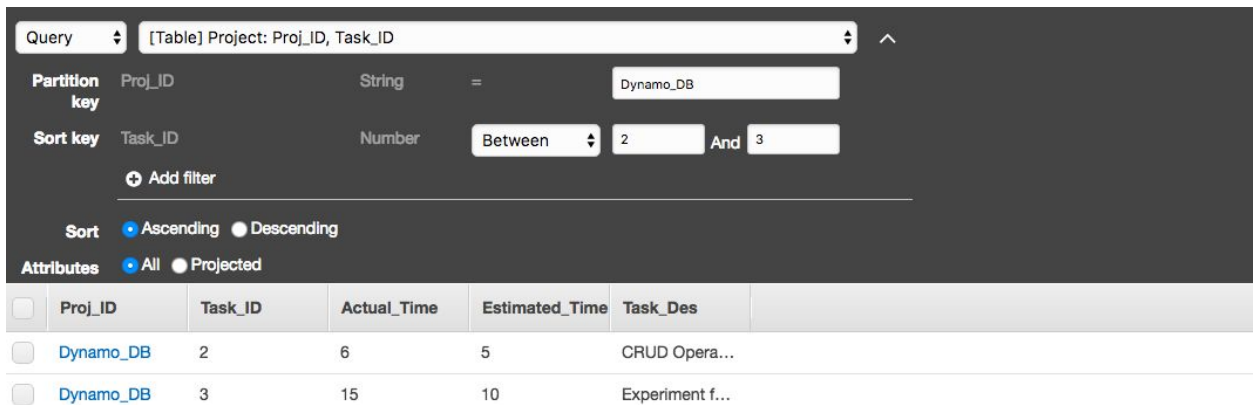


The screenshot shows the AWS IAM console interface for a table named "[Table] Project: Proj\_ID, Task\_ID". The "Items" tab is selected, and a scan operation is in progress. The scan filter is set to "Task\_ID" with the operator ">" and the value "2". The results table shows two items:

Proj_ID	Task_ID	Actual_Time	Estimated_Time	Task_Des
Dynamo_DB	3	15	10	Experiment f...
Software_Dev	3	24	12	Testing

A scan operation scans the entire table. You can specify filters to apply to the results to refine the values returned to you, after the complete scan.

## 2. Query (Project Dynamo\_DB and Task between 2 to 3)



The screenshot shows the AWS IAM console interface for a table named "[Table] Project: Proj\_ID, Task\_ID". The "Query" tab is selected, and a query operation is in progress. The query filters are set to "Proj\_ID" = "Dynamo\_DB" and "Task\_ID" between "2" and "3". The results table shows two items:

Proj_ID	Task_ID	Actual_Time	Estimated_Time	Task_Des
Dynamo_DB	2	6	5	CRUD Opera...
Dynamo_DB	3	15	10	Experiment f...

A query operation searches only primary key attribute values and supports a subset of comparison operators on key attribute values to refine the search process.

## d) Creating Global Secondary Index:

Option1 : We can specify Global Secondary Index, While creating Table Initially.

### Create DynamoDB table Tutorial ?

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

**Table name\***  ?

**Primary key\*** Partition key

?

Add sort key

#### Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

#### Secondary indexes

Name	Type	Partition key	Sort key	Projected Attributes
------	------	---------------	----------	----------------------

[+ Add index](#)

Option2: Or, After Table is created, Go to Indexes tab and click on Create Index:

1

The screenshot shows the AWS IAM console interface for a table named 'Order'. The 'Indexes' tab is selected, and the 'Create Index' button is highlighted. A tooltip explains that Global Secondary Indexes (GSI) allow querying over any field in a DynamoDB table.

Order Close

Overview Items Metrics Alarms Capacity **Indexes** Global Tables Backups Triggers Access control Tags

**Create Index** Delete Index Settings Refresh

Name	Status	Type	Partition key	Sort key	Attributes	Read capacity	Write capacity
------	--------	------	---------------	----------	------------	---------------	----------------

Global Secondary Indexes (GSI) allow you to query efficiently over any field (attribute) in your DynamoDB table. GSIs can treat any table attribute as a key, even attributes not present in all items. [More info](#)

2. The below window will appear.

Here we can provide Primary key for Index and also optional sort key as we created in case of Primary Index

There extra charge for creating Secondary Index and we can also change the read and write capacity for the new partition.

### Create index ✕

**Primary key\*** Partition key

String ⓘ

Add sort key

**Index name\***  ⓘ

**Projected attributes** All ⓘ

---

Read capacity units

Write capacity units

Estimated cost \$2.91 / month ([Capacity calculator](#))

---

**Auto Scaling: Read Capacity**  Apply same settings as table

**Auto Scaling: Write Capacity**  Apply same settings as table

---

Approximate creation time is **5 minutes**. Additional write capacity may decrease creation time. A notification will be sent to the SNS topic dynamodb once the index creation is complete. Basic Alarms with 80% upper threshold using SNS topic 'dynamodb' will be automatically created. Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced configuration for alarms can be done in the alarms tab.

---

Cancel Create Index

3. We selected Price index from Order table as secondary Index:

As we can see when we increase read/ write capacity of table price increased.

## Create index ✕

**Primary key\*** Partition key

Price Number ⓘ

Add sort key

**Index name\*** Price-index ⓘ

**Projected attributes** All ⓘ

---

Read capacity units Write capacity units

10 10

Estimated cost \$5.81 / month ([Capacity calculator](#))

---

**Auto Scaling: Read Capacity**  Apply same settings as table

**Auto Scaling: Write Capacity**  Apply same settings as table

---

Approximate creation time is 5 minutes. Additional write capacity may decrease creation time. A notification will be sent to the SNS topic dynamodb once the index creation is complete. Basic Alarms with 80% upper threshold using SNS topic 'dynamodb' will be automatically created. Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced configuration for alarms can be done in the alarms tab.

---

Cancel Create Index

4. For this Activity , We kept the read and write capacity same as before i.e, 5.



The Index is created:

Order Close

Overview Items Metrics Alarms Capacity **Indexes** Global Tables Backups Triggers Access control Tags

Create index Delete Index

Name	Status	Type	Partition key	Sort key	Attributes	Read capacity	Write capacity
Price-index	Creating	GSI	Price (Number)	-	ALL	5	5

5. Now in Table we can see 2 indexes.

Order Close

Overview **Items** Metrics Alarms Capacity Indexes Global Tables Backups More

Create item Actions

Scan: [Table] Order: Order\_ID Viewing 1 to 4 items

Scan [Table] Order: Order\_ID [Index] Price-index: Price

Start search

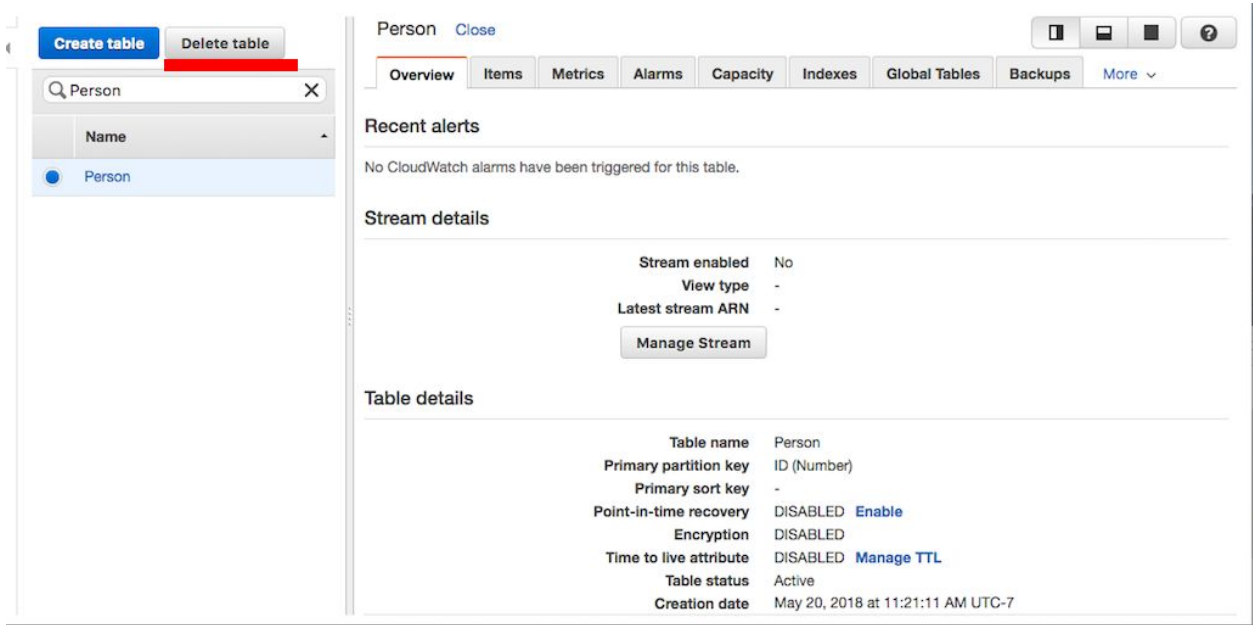
Order_ID	Order_Descriptio	Price
2	Cold Drink	2
1	IceCream	5
4	Fries	2
3	Burger	4

We can now perform scan and query operations using secondary index as well.

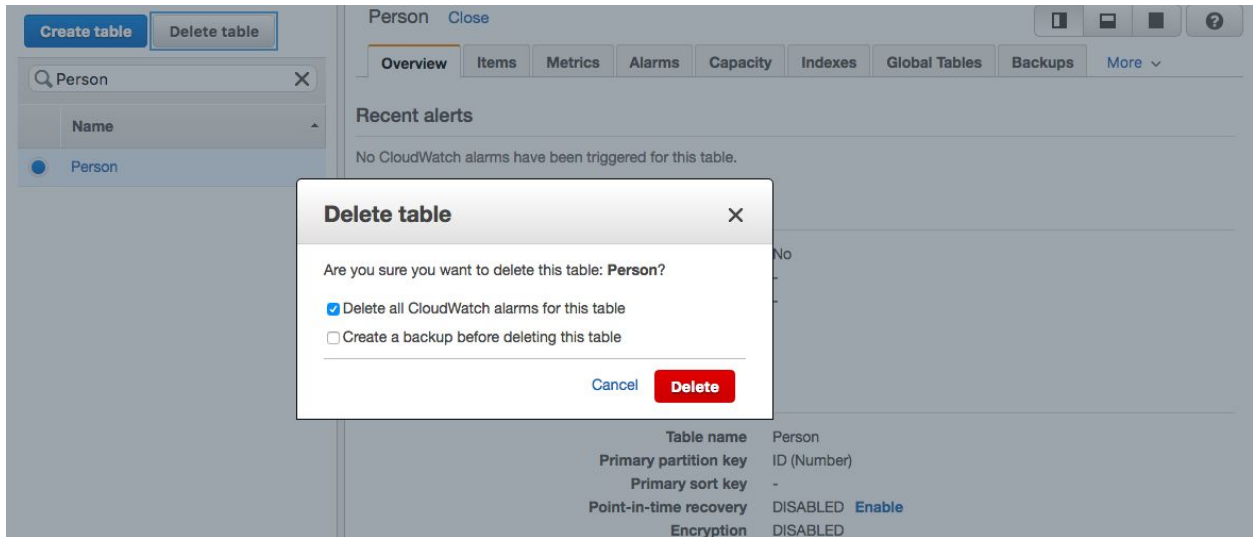
e) Delete Table

To Delete Table Person, We just need to select that table and click on Delete.

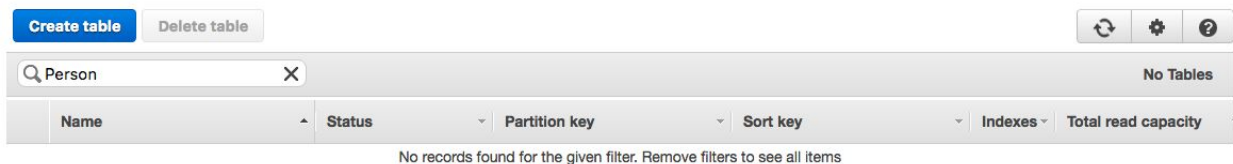




You can Choose to select backup:



Deleted Successfully.



## Demo\_2(Programming API):

### Set up and getting access to DynamoDB programmatically:

- 1) Create AWS account and set up AWS access Key in order to use the AWS SDKs and then configure your credentials to access DynamoDB programmatically.
- 2) To set up AWS SDK for Java, first install a java development environment. If you are using Eclipse IDE then install AWS Toolkit for eclipse ,AWS SDK for java and set up your AWS security credentials.
- 3) Now, you will be able to access the DynamoDB programmatically.

The steps and process is clearly explained in the following link

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.Java.htm>  
!

The following operations are performed in the DynamoDB for Demo:

- 1) Create table
- 2) Insert a new item to the table
- 3) Retrieve an item from the table
- 4) Update an item in the table
- 5) Delete an item
- 6) Upload a csv file into DynamoDB as another table