**Dynamo db using python**

Create table: done using **create\_table .**

Ex

Step1. Make.py file  
 Step 2. Import boto

Step3 you must provide : table name, primary key(key schema,attribute definitions, throughput provision(ReadCapacityUnits,WriteCapacityUnits)

Step 4- run file by : python filename.py

Load Sample Data(just like seeder in python django or adding data to table through external json file): done using **put\_item**

**Ex**

Step1. py file  
 Step 2. Import boto

Step3 you must use put\_item to add data in a specific table

Step 4- run file by : python filename.py

Create/Insert Item : done using **put\_item**

**Ex :** table = dynamodb.Table('Movies')

response = table.put\_item(

Item={

'year': year,

'title': title,

}

)

Read Item : done using **get\_item (just like a get() in django orm)**

In this we need to specify a key to filter  
Ex:

table = dynamodb.Table('Movies')

try:

response = table.get\_item(Key={'year': year, 'title': title})

except ClientError as e:

print(e.response['Error']['Message'])

else:

return response['Item']

Update Item : done using **update\_item (just like a update() in django orm)**

In this we need to specify a key to filter

Ex: table = dynamodb.Table('Movies')

response = table.update\_item(

Key={

'year': year,

'title': title

},

UpdateExpression="set info.rating=:r, info.plot=:p, info.actors=:a",

ExpressionAttributeValues={

':r': Decimal(rating),

':p': plot,

':a': actors

},

ReturnValues="UPDATED\_NEW"

)

Update Item Conditionally : done using **update\_item (just like a update() in django orm) but with a condition expression**

**If the condition evaluates to true, the update succeeds; otherwise, the update is not performed.**

**Ex:**

try:

response = table.update\_item(

Key={

'year': year,

'title': title

},

UpdateExpression="remove info.actors[0]",

ConditionExpression="size(info.actors) > :num",

ExpressionAttributeValues={':num': actor\_count},

ReturnValues="UPDATED\_NEW"

)

except ClientError as e:

if e.response['Error']['Code'] == "ConditionalCheckFailedException":

print(e.response['Error']['Message'])

else:

raise

else:

return response

Delete Item : done using **delete\_item (just like a delete() in django orm)**

In this we need to specify a key to filter

Ex:

try:

response = table.delete\_item(

Key={

'year': year,

'title': title

},

ConditionExpression="info.rating <= :val",

ExpressionAttributeValues={

":val": Decimal(rating)

}

)

except ClientError as e:

if e.response['Error']['Code'] == "ConditionalCheckFailedException":

print(e.response['Error']['Message'])

else:

raise

else:

return response

Increment an Atomic Counter:

DynamoDB supports atomic counters, which use the update\_item method to increment or decrement the value of an existing attribute without interfering with other write requests. (All write requests are applied in the order in which they are received.)

Query data: : done using **query (just like a filter() in django orm)**

In this we need to specify a key to filter

Ex:

table = dynamodb.Table('Movies')

response = table.query(

KeyConditionExpression=Key('year').eq(year)

)

**NOTE:The Boto 3 SDK constructs a ConditionExpression for you when you use the Key and Attr functions imported from boto3.dynamodb.conditions. You can also specify a ConditionExpression as a string.**

**NOTE- if you want only some fields to be returned in response(just like we use values in django orm):  
Then we can use ProjectionExpression for that:**

**Ex:** table = dynamodb.Table('Movies')

response = table.query(

ProjectionExpression="#yr, title, info.genres, info.actors[0]",

ExpressionAttributeNames={"#yr": "year"},

KeyConditionExpression=

Key('year').eq(year) & Key('title').between(title\_range[0], title\_range[1])

)

return response['Items']

Scan data: : done using **scan (just like a all() in django orm)**

**The scan method reads every item in the entire table and returns all the data in the table. You can provide an optional filter\_expression so that only the items matching your criteria are returned. However, the filter is applied only after the entire table has been scanned.**

In this we can specify a **filter\_expression** to filter.(optional)

Ex:

table = dynamodb.Table('Movies')

response = table.scan(

'FilterExpression': Key('year').between(\*year\_range),

'ProjectionExpression': "#yr, title, info.rating",

'ExpressionAttributeNames': {"#yr": "year"}

)

Delete a table : done using **delete**

**Ex:**

table = dynamodb.Table('Movies')

table.delete()

.