

**HU Extension**

Handed out: 10/10/2015

**Assignment 06**

**E-90 Cloud Computing**

Due by 11:59 PM EST on Friday, 10/16/2015

Place all of your narratives and illustrations in a single Word or PDF document named E90\_LastNameFirstNameHW06.docx [.pdf]. Use this assignment as the initial template. Add your steps and your code below problem statements used for that problem. Upload your homework file and your working code (e.g., filename.java) into your Assignment 6 folder. Do not include executables.

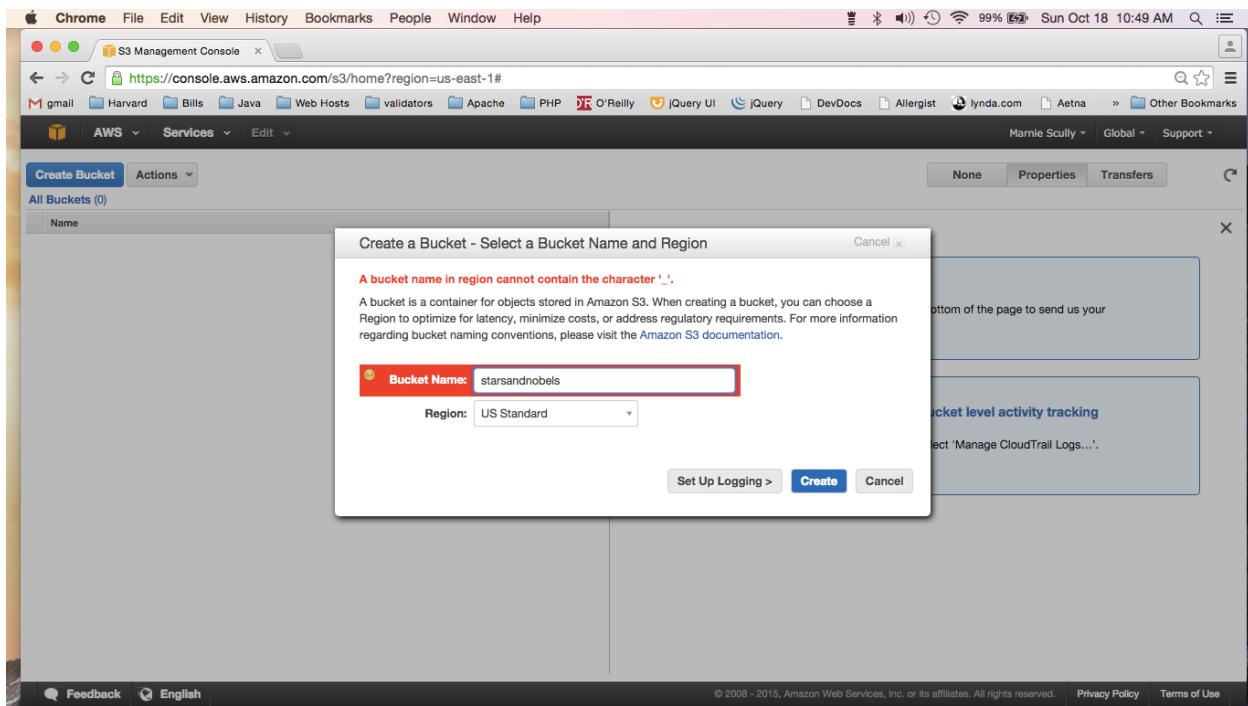
**Problem 1.**

- I. Manually populate an S3 bucket with Images and Resumes of movie stars and Nobel laureates:
  - a. Create folder structures with two folders: **stars** and **nobels**. Inside either folder you should have subfolders **images** and **resumes**.
  - b. From Google Images fetch images (pictures) of three movie stars and three Nobel winners. Produce 3+3 MS Word documents with bogus resumes for selected movie stars and Nobel winners on your own. The resumes should be very short; three sentences each.
  - c. Upload those images and resume into your S3 bucket manually.
  - d. **Manually** grant general public access to those images and resumes.
- II. Create one SimpleDB domain and programmatically insert one row for each movie star and each Nobel laureates - write your program either in Java or .Net or your favorite language supported by AWS:
  - a. Use their full names with hyphens between the first and the last name as the item keys.
  - b. In rows for movie stars record full name, most popular movie, S3 URL of the picture of the star and S3 URL of his or her resume.
  - c. Use those same attributes for the Nobel laureates and add the year they won the prize and the field of science in which they got the prize as two additional attributes.
  - d. Demonstrate that you can change (correct) the year of one Nobel prize award, programmatically.
  - e. Demonstrate that you can delete one movie star from SimpleDB domain programmatically.
  - f. Capture the content of your database as displayed in the Database Development perspective.

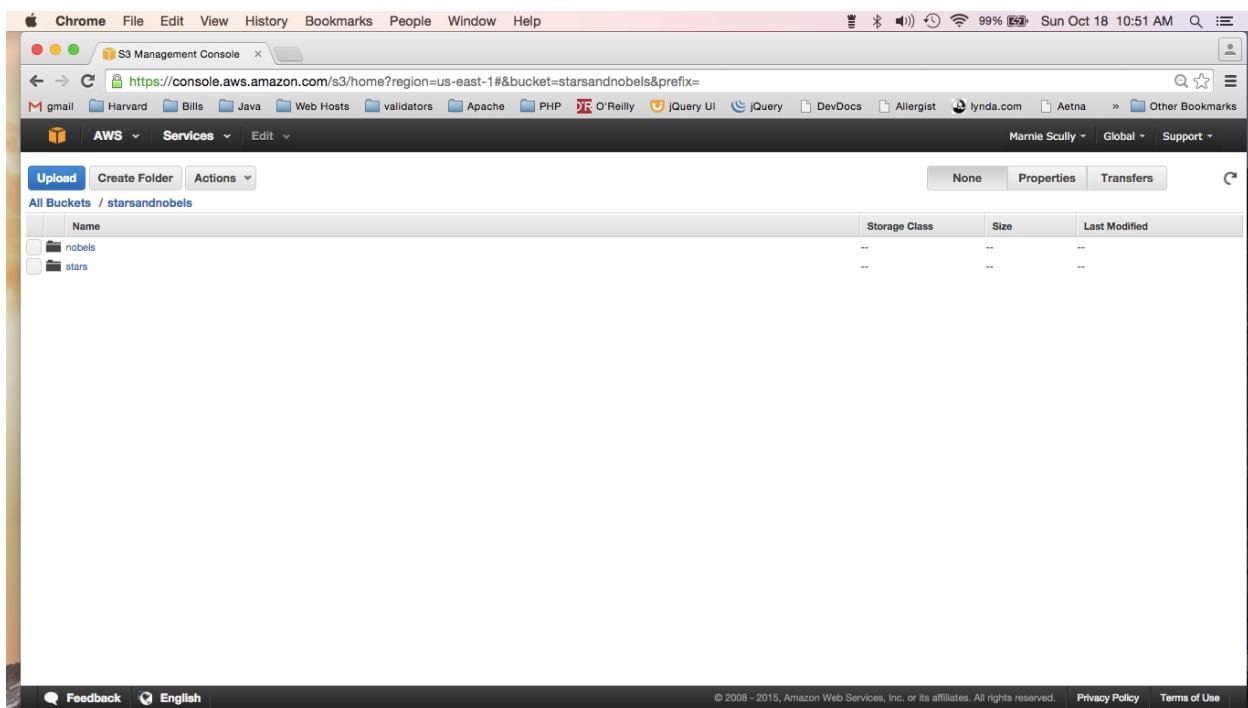
Provide working code and capture all stages of testing. Write your program either in Java or .Net or your favorite language supported by AWS. **[30 Points]**

## HW6\_ScullyMarnie

### Create a S3 Bucket named starsandnobels



Create folders within bucket for stars, nobels, with subfolders images and resumes in each.



## HW6\_ScullyMarnie

The screenshot shows the AWS S3 Management Console interface. The URL in the address bar is <https://console.aws.amazon.com/s3/home?region=us-east-1#&bucket=starsandnobels&prefix=nobels/>. The page displays the contents of the 'nobels' folder within the 'starsandnobels' bucket. There are two subfolders: 'images' and 'resumes'. A navigation bar at the top includes 'Upload', 'Create Folder', and 'Actions' buttons, along with tabs for 'None', 'Properties', and 'Transfers'. The bottom of the screen shows standard browser controls like Feedback and English language selection.

Upload all images and resumes to proper folders

The screenshot shows the AWS S3 Management Console interface with the URL <https://console.aws.amazon.com/s3/home?region=us-east-1#&bucket=starsandnobels&prefix=stars/images/>. A modal dialog box titled 'Upload - Select Files and Folders' is open, prompting the user to 'Drag and drop files and folders to upload here.' Three files are listed for upload: 'dustin.jpeg' (6.3 KB), 'marilyn.jpeg' (7 KB), and 'meryl.jpg' (957 KB). Below the list are 'Add Files' and 'Remove Selected Files' buttons. At the bottom of the dialog, it says 'Number of files: 3 Total upload size: 970.4 KB'. The background shows the same S3 bucket structure as the previous screenshot, with the 'stars' folder containing 'images' and 'resumes' subfolders.

## HW6\_ScullyMarnie

Grant Public access to the images and resumes within stars and novels

The screenshot shows the AWS S3 Management Console interface. The URL is https://console.aws.amazon.com/s3/home?region=us-east-1#&bucket=starsandnobels&prefix=. The left sidebar shows 'All Buckets / starsandnobels'. Inside the bucket, there are two objects: 'nobels' and 'stars'. A context menu is open over the 'stars' object, with 'Make Public' highlighted. The main pane shows '2 items selected' with details: Bucket: starsandnobels, Selected: 2. Below the main pane, there's a preview area showing a file named 'curie.docx'.

### Create a SimpleDB domain

```
String myDomain = "starsandnobels";
System.out.println("Creating domain called " + myDomain + ".\n");
sdb.createDomain(new CreateDomainRequest(myDomain));
```

### Insert a row for each star and laureate

```
/* Creates an array of SimpleDB ReplaceableItems populated with sample data. */
private static List<ReplaceableItem> createSampleData() {
    List<ReplaceableItem> sampleData = new ArrayList<ReplaceableItem>();

    sampleData.add(new ReplaceableItem("Dustin-Hoffman").withAttributes(
        new ReplaceableAttribute("Full Name", "Dustin Hoffman", true),
        new ReplaceableAttribute("Type", "star", true),
        new ReplaceableAttribute("Most Popular Movie", "Tootsie", true),
        new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/
stars/images/dustin.jpeg", true),
        new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/
stars/resumes/dustin.docx", true)));
```

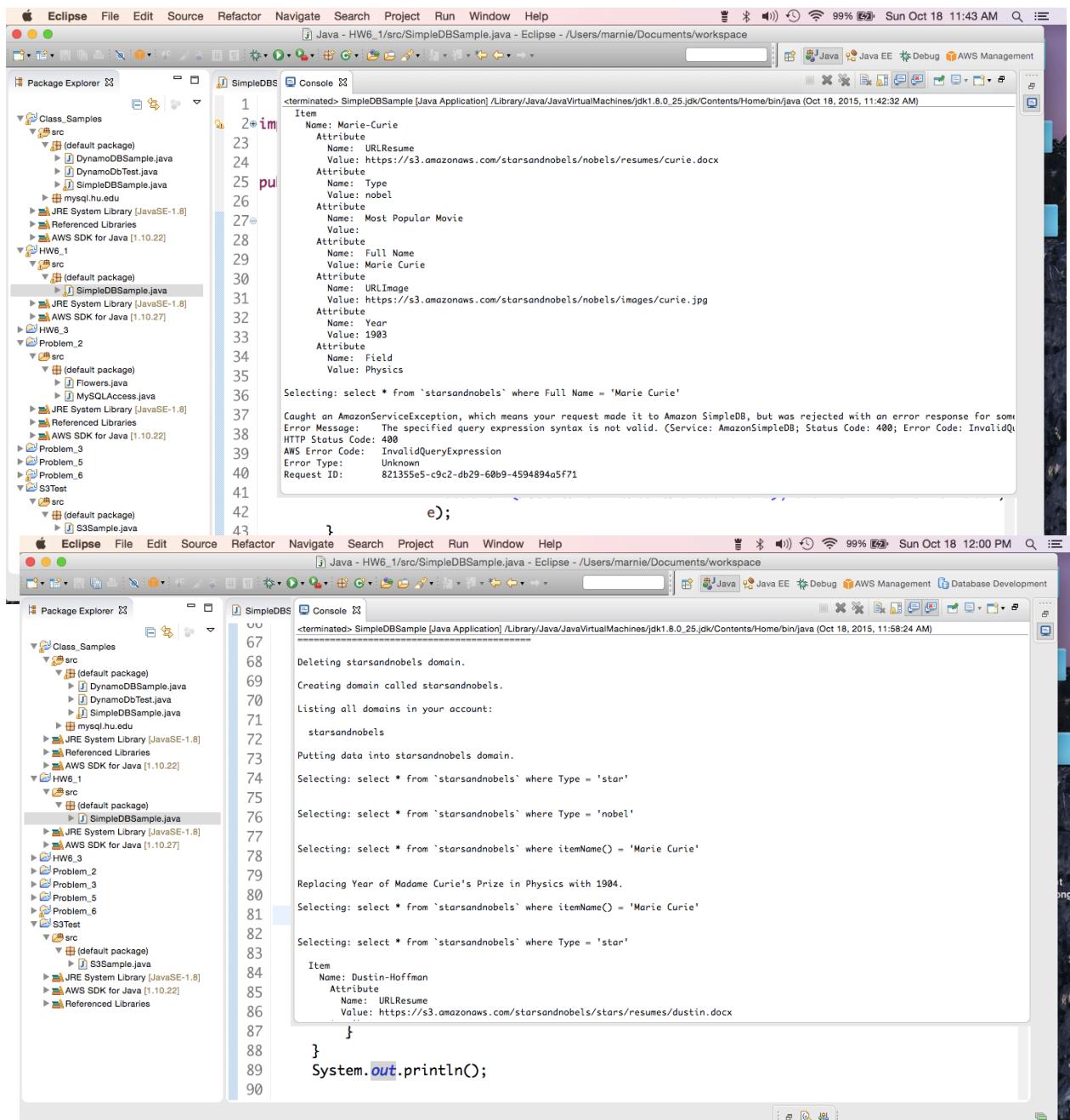
## HW6\_ScullyMarnie

```
sampleData.add(new ReplaceableItem("Marilyn-Monroe").withAttributes(  
    new ReplaceableAttribute("Full Name", "Marilyn Monroe", true),  
    new ReplaceableAttribute("Type", "star", true),  
    new ReplaceableAttribute("Most Popular Movie", "Some Like it Hot", true),  
    new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/  
stars/images/marilyn.jpeg", true),  
    new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/  
stars/resumes/marilyn.docx", true)));  
  
sampleData.add(new ReplaceableItem("Meryl-Streep").withAttributes(  
    new ReplaceableAttribute("Full Name", "Meryl Streep", true),  
    new ReplaceableAttribute("Type", "star", true),  
    new ReplaceableAttribute("Most Popular Movie", "Sophie's Choice", true),  
    new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/  
stars/images/meryl.jpeg", true),  
    new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/  
stars/resumes/meryl.docx", true)));  
  
sampleData.add(new ReplaceableItem("Marie-Curie").withAttributes(  
    new ReplaceableAttribute("Full Name", "Marie Curie", true),  
    new ReplaceableAttribute("Type", "nobel", true),  
    new ReplaceableAttribute("Most Popular Movie", "", true), // the instructions said to  
include this for the nobels too  
    new ReplaceableAttribute("Year", "1903", true),  
    new ReplaceableAttribute("Field", "Physics", true),  
    new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/  
nobels/images/curie.jpg", true),  
    new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/  
nobels/resumes/curie.docx", true)));  
  
sampleData.add(new ReplaceableItem("Eugene-Oneil").withAttributes(  
    new ReplaceableAttribute("Full Name", "Eugene O'Neil", true),  
    new ReplaceableAttribute("Type", "nobel", true),  
    new ReplaceableAttribute("Most Popular Movie", "", true),  
    new ReplaceableAttribute("Year", "1936", true),  
    new ReplaceableAttribute("Field", "Literature", true),  
    new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/  
nobels/images/oneil.jpg", true),  
    new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/  
nobels/resumes/oneil.docx", true)));  
  
sampleData.add(new ReplaceableItem("George-Bernard-Shaw").withAttributes(  
    new ReplaceableAttribute("Full Name", "George Bernard Shaw", true),
```

## HW6\_ScullyMarnie

```
new ReplaceableAttribute("Type", "nobel", true),
new ReplaceableAttribute("Most Popular Movie", "", true),
new ReplaceableAttribute("Year", "1925", true),
new ReplaceableAttribute("Field", "Literature", true),
new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/
nobels/images/shaw.jpg", true),
new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/
nobels/resumes/shaw.docx", true)));
return sampleData;
}
```

## Some testing as I developed



The screenshot shows two instances of the Eclipse IDE running side-by-side, each displaying a Java application console.

**Top Console (Oct 18, 2015, 11:42:32 AM):**

```
Item
Name: Marie-Curie
Attribute
Name: URLResume
Value: https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx
Attribute
Name: Type
Value: nobel
Attribute
Name: Most Popular Movie
Value:
Attribute
Name: Full Name
Value: Marie Curie
Attribute
Name: URLImage
Value: https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpg
Attribute
Name: Year
Value: 1903
Attribute
Name: Field
Value: Physics

Selecting: select * from `starsandnobels` where Full Name = 'Marie Curie'

Caught an AmazonServiceException, which means your request made it to Amazon SimpleDB, but was rejected with an error response for some reason.
Error Message: The specified query expression syntax is not valid. (Service: AmazonSimpleDB; Status Code: 400; Error Code: InvalidQueryExpression; Request ID: 821355e5-c9c2-db29-60b9-4594894a5f71)
```

**Bottom Console (Oct 18, 2015, 11:58:24 AM):**

```
Deleting starsandnobels domain.

Creating domain called starsandnobels.

Listing all domains in your account:

starsandnobels

Putting data into starsandnobels domain.

Selecting: select * from `starsandnobels` where Type = 'star'

Selecting: select * from `starsandnobels` where Type = 'nobel'

Selecting: select * from `starsandnobels` where itemName() = 'Marie Curie'

Replacing Year of Madame Curie's Prize in Physics with 1904.

Selecting: select * from `starsandnobels` where itemName() = 'Marie Curie'

Selecting: select * from `starsandnobels` where Type = 'star'

Item
Name: Dustin-Hoffman
Attribute
Name: URLResume
Value: https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx

}
System.out.println();
```

## HW6\_ScullyMarnie

### Demonstrate editing a value in an item and Demonstrate deleting an Item

#### (Output in Eclipse)

---

---

Getting Started with Amazon SimpleDB

---

---

Deleting starsandnobels domain.

Creating domain called starsandnobels.

Listing all domains in your account:

starsandnobels

Putting data into starsandnobels domain.

Selecting: select \* from `starsandnobels`

Item

Name: Dustin-Hoffman

Attribute

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx>

Attribute

Name: Type

Value: star

Attribute

Name: Most Popular Movie

Value: Tootsie

Attribute

Name: Full Name

Value: Dustin Hoffman

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg>

Item

Name: Eugene-Oneil

Attribute

Name: URLResume

## HW6\_ScullyMarnie

Value: <https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx>

Attribute

Name: Type

Value: nobel

Attribute

Name: Most Popular Movie

Value:

Attribute

Name: Full Name

Value: Eugene O'Neil

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/nobels/images/oneil.jpg>

Attribute

Name: Year

Value: 1936

Attribute

Name: Field

Value: Literature

Item

Name: George-Bernard-Shaw

Attribute

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/nobels/resumes/shaw.docx>

Attribute

Name: Type

Value: nobel

Attribute

Name: Most Popular Movie

Value:

Attribute

Name: Full Name

Value: George Bernard Shaw

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/nobels/images/shaw.jpg>

Attribute

Name: Year

Value: 1925

Attribute

Name: Field

Value: Literature

Item

## HW6\_ScullyMarnie

Name: Marie-Curie

Attribute

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx>

Attribute

Name: Type

Value: nobel

Attribute

Name: Most Popular Movie

Value:

Attribute

Name: Full Name

Value: Marie Curie

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpg>

Attribute

Name: Year

Value: 1903

Attribute

Name: Field

Value: Physics

Item

Name: Marilyn-Monroe

Attribute

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx>

Attribute

Name: Type

Value: star

Attribute

Name: Most Popular Movie

Value: Some Like it Hot

Attribute

Name: Full Name

Value: Marilyn Monroe

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg>

Item

Name: Meryl-Streep

Attribute

Name: URLResume

## HW6\_ScullyMarnie

Value: <https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx>

Attribute

Name: Type

Value: star

Attribute

Name: Most Popular Movie

Value: Sophie's Choice

Attribute

Name: Full Name

Value: Meryl Streep

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg>

Selecting: select \* from `starsandnobels` where itemName() = 'Marie-Curie'

Item

Name: Marie-Curie

Attribute

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx>

Attribute

Name: Type

Value: nobel

Attribute

Name: Most Popular Movie

Value:

Attribute

Name: Full Name

Value: Marie Curie

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpg>

Attribute

Name: Year

Value: 1903

Attribute

Name: Field

Value: Physics

Replacing Year of Madame Curie's Prize in Physics with 1904.

Selecting: select \* from `starsandnobels` where itemName() = 'Marie-Curie'

## HW6\_ScullyMarnie

Item

Name: Marie-Curie

Attribute

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx>

Attribute

Name: Type

Value: nobel

Attribute

Name: Most Popular Movie

Value:

Attribute

Name: Full Name

Value: Marie Curie

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpg>

Attribute

Name: Year

Value: 1904

Attribute

Name: Field

Value: Physics

Selecting: select \* from `starsandnobels` where Type = 'star'

Item

Name: Dustin-Hoffman

Attribute

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx>

Attribute

Name: Type

Value: star

Attribute

Name: Most Popular Movie

Value: Tootsie

Attribute

Name: Full Name

Value: Dustin Hoffman

Attribute

Name: URLImage

## HW6\_ScullyMarnie

Value: <https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg>

Item

Name: Marilyn-Monroe

Attribute

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx>

Attribute

Name: Type

Value: star

Attribute

Name: Most Popular Movie

Value: Some Like it Hot

Attribute

Name: Full Name

Value: Marilyn Monroe

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg>

Item

Name: Meryl-Streep

Attribute

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx>

Attribute

Name: Type

Value: star

Attribute

Name: Most Popular Movie

Value: Sophie's Choice

Attribute

Name: Full Name

Value: Meryl Streep

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg>

Deleting Dustin Hoffman.

Selecting: select \* from `starsandnobels` where Type = 'star'

Item

Name: Marilyn-Monroe

Attribute

## HW6\_ScullyMarnie

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx>

Attribute

Name: Type

Value: star

Attribute

Name: Most Popular Movie

Value: Some Like it Hot

Attribute

Name: Full Name

Value: Marilyn Monroe

Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg>

Item

Name: Meryl-Streep

Attribute

Name: URLResume

Value: <https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx>

Attribute

Name: Type

Value: star

Attribute

Name: Most Popular Movie

Value: Sophie's Choice

Attribute

Name: Full Name

Value: Meryl Streep

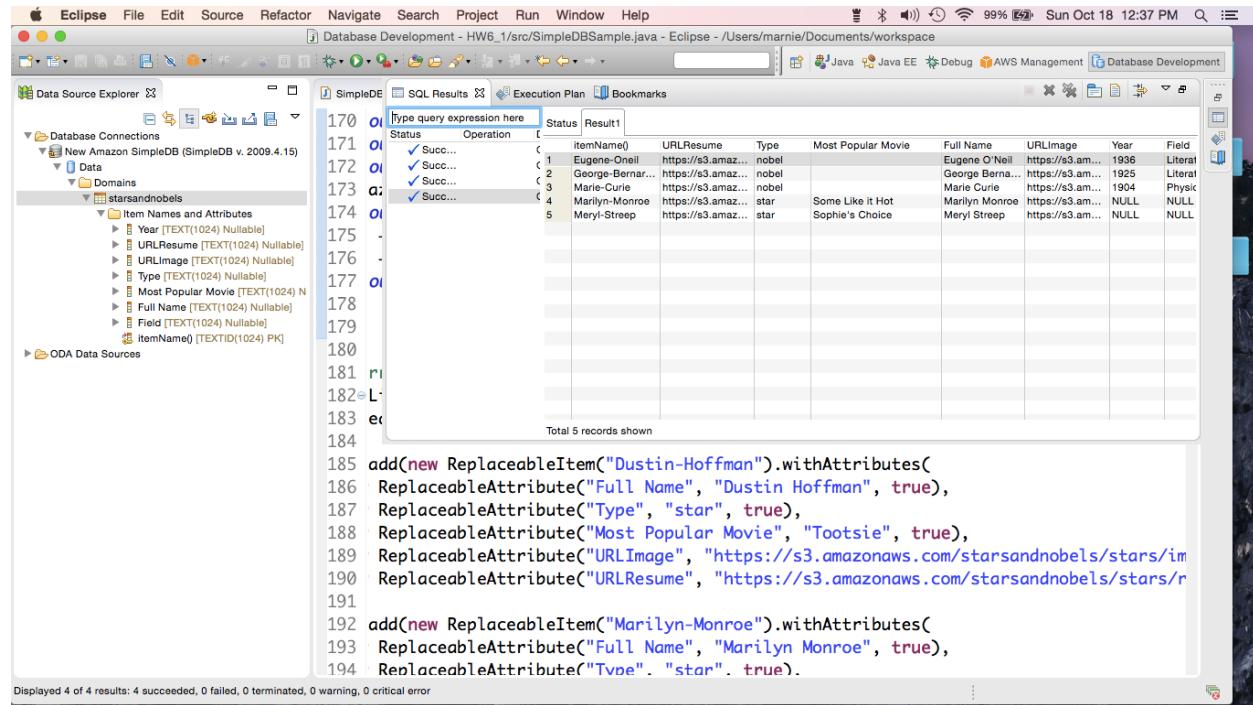
Attribute

Name: URLImage

Value: <https://s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg>

## HW6\_ScullyMarnie

### Database after changes in Database Development Perspective



The screenshot shows the Eclipse IDE interface with the Database Development perspective selected. In the center, there is a SQL Results view displaying a table with 5 records. The table has columns: Status, Operation, ItemName, URLResume, Type, Most Popular Movie, Full Name, URLImage, Year, and Field. The data is as follows:

Status	Operation	ItemName	URLResume	Type	Most Popular Movie	Full Name	URLImage	Year	Field
✓ Succ...	1	Eugene-O'Neill	https://s3.amaz...	nobel		Eugene O'Neil	https://s3.amaz...	1936	Literat
✓ Succ...	2	George-Bernar...	https://s3.amaz...	nobel		George Bernar...	https://s3.amaz...	1925	Literat
✓ Succ...	3	Marie-Curie	https://s3.amaz...	nobel		Marie Curie	https://s3.amaz...	1904	Physic
✓ Succ...	4	Marilyn-Monroe	https://s3.amaz...	star	Some Like it Hot	Marilyn Monroe	https://s3.amaz...	NULL	NULL
	5	Meryl-Streep	https://s3.amaz...	star	Sophie's Choice	Meryl Streep	https://s3.amaz...	NULL	NULL

Below the table, the SQL code used to perform the operation is shown:

```
170 oI type query expression here
171 oI Status Operation
172 oI ✓ Succ...
173 oI ✓ Succ...
174 oI ✓ Succ...
175 -
176 -
177 oI
178
179
180
181 rI
182 L:
183 ec
184
185 add(new ReplaceableItem("Dustin-Hoffman").withAttributes(
186 ReplaceableAttribute("Full Name", "Dustin Hoffman", true),
187 ReplaceableAttribute("Type", "star", true),
188 ReplaceableAttribute("Most Popular Movie", "Tootsie", true),
189 ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/stars/im
190 ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/stars/r
191
192 add(new ReplaceableItem("Marilyn-Monroe").withAttributes(
193 ReplaceableAttribute("Full Name", "Marilyn Monroe", true),
194 ReplaceableAttribute("Type". "star". true)).
```

Displayed 4 of 4 results: 4 succeeded, 0 failed, 0 terminated, 0 warning, 0 critical error

### Problem 2.

DynamoDB version. Use the same S3 bucket, images and resumes created in Problem 1.

Use AWS DynamoDB Console for all the work

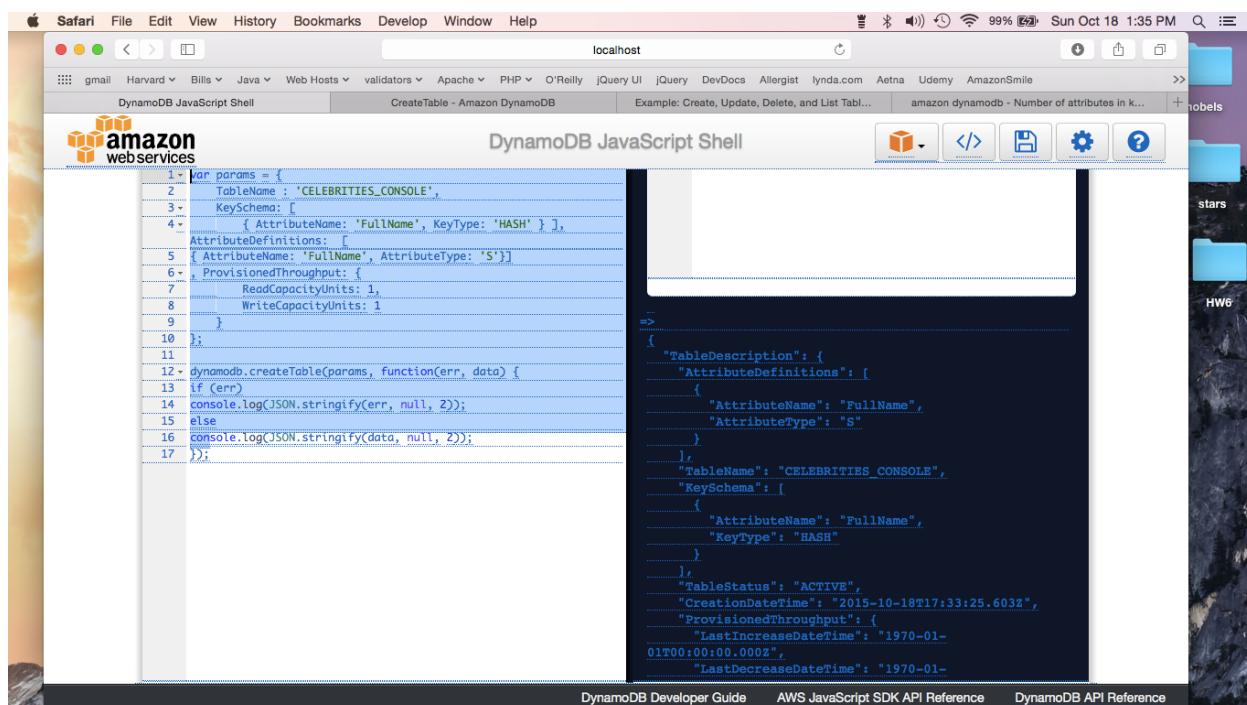
- a) Create a new table, "CELEBRITIES\_CONSOLE", where primary key is of a Hash type, and is constructed as "firstName" + "-" + "lastName". (for example, "Julia-Roberts")
  - b) Insert three rows for movie stars, with attributes: full name (= firstName + " " + lastName), most popular movie, S3 URL of the picture of the star and S3 URL of his or her resume
  - c) Insert three rows for Noble laureates, with attributes: full name (same format as above), S3 URL of the picture of the lauret, S3 URL of his or her resume, award year, science field in which they got the prize
  - d) Explore the table in the Console and search by providing filters on ID - both scans and get by ID types of queries
  - e) Update an award year of one of the Nobles rows - demo that the data is updated.
- Capture all stages of testing. **[15 Points]**

## HW6\_ScullyMarnie

In DynamoDBLocal place this code and run. See results on right pane.

```
var params = {
  TableName : 'CELEBRITIES_CONSOLE',
  KeySchema: [
    { AttributeName: 'FullName', KeyType: 'HASH' } ],
  AttributeDefinitions: [
  { AttributeName: 'FullName', AttributeType: 'S'}]
, ProvisionedThroughput: {
  ReadCapacityUnits: 1,
  WriteCapacityUnits: 1
}
};

dynamodb.createTable(params, function(err, data) {
if (err)
  console.log(JSON.stringify(err, null, 2));
else
  console.log(JSON.stringify(data, null, 2));
});
```



```
DynamoDB JavaScript Shell

1 var params = {
2   TableName : 'CELEBRITIES_CONSOLE',
3   KeySchema: [
4     { AttributeName: 'FullName', KeyType: 'HASH' },
5     AttributeDefinitions: [
6       { AttributeName: 'FullName', AttributeType: 'S'}]
7   , ProvisionedThroughput: {
8     ReadCapacityUnits: 1,
9     WriteCapacityUnits: 1
10 }
11 };
12 dynamodb.createTable(params, function(err, data) {
13 if (err)
14   console.log(JSON.stringify(err, null, 2));
15 else
16   console.log(JSON.stringify(data, null, 2));
17 });

{
  "TableDescription": {
    "AttributeDefinitions": [
      {
        "AttributeName": "FullName",
        "AttributeType": "S"
      }
    ],
    "TableName": "CELEBRITIES_CONSOLE",
    "KeySchema": [
      {
        "AttributeName": "FullName",
        "KeyType": "HASH"
      }
    ],
    "TableStatus": "ACTIVE",
    "CreationDateTime": "2015-10-18T17:33:25.603Z",
    "ProvisionedThroughput": {
      "LastIncreaseDateTime": "1970-01-01T00:00:00Z",
      "LastDecreaseDateTime": "1970-01-
```

## HW6\_ScullyMarnie

Insert All items in DB

The screenshot shows a Mac OS X desktop with a Safari browser window open to the 'localhost' address. The title bar says 'localhost'. The tab bar includes 'DynamoDB JavaScript Shell', 'CreateTable - Amazon DynamoDB', and 'Example: Create, Update, Delete, and List Tabl...'. The main content area is titled 'DynamoDB JavaScript Shell' and contains a code editor with the following JavaScript code:

```
1 var params = { RequestItems: {
2   'CELEBRITIES_CONSOLE': [
3     { PutRequest: {
4       Item: {
5         'FullName': 'Marilyn-Monroe',
6         'PopularMovie': 'Some Like it Hot',
7         'URLPic': 'https://s3.amazonaws.com/starsandnobels/stars
8 /images/marilyn.jpeg',
9         'URLRes': 'https://s3.amazonaws.com/starsandnobels/stars
10 /resumes/marilyn.docx',
11         'Year': 0,
12         'Field': 'none'
13       }
14     }}
15   ]
16 };
17
18 });
19 dynamodb.batchWriteItem(params, function (err, data) {
20 if (err)
21   console.log(JSON.stringify(err, null, 2));
22 else
23   console.log(JSON.stringify(data, null, 2));
24 });
25
26
27
28
29
30 }
```

The right side of the interface has a toolbar with icons for file operations like save, copy, and paste. Below the toolbar is a preview pane showing the JSON structure of the inserted item.

Query Table by PrimaryKey

The screenshot shows a Mac OS X desktop with a Safari browser window open to the 'localhost' address. The title bar says 'localhost'. The tab bar includes 'DynamoDB JavaScript Shell', 'CreateTable - Amazon DynamoDB', and 'Example: Create, Update, Delete, and List Tabl...'. The main content area is titled 'DynamoDB JavaScript Shell' and contains a code editor with the following JavaScript code:

```
1 var params = {
2   TableName: "CELEBRITIES_CONSOLE",
3   Key: {
4     "FullName": "Marie-Curie"
5   }
6 };
7 dynamodb.getItem(params, function(err, data) {
8   if (err)
9     console.log(JSON.stringify(err, null, 2));
10 else
11   console.log(JSON.stringify(data, null, 2));
12 });
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30 }
```

The right side of the interface has a toolbar with icons for file operations like save, copy, and paste. Below the toolbar is a preview pane showing the JSON structure of the queried item.

## HW6\_ScullyMarnie

### Query with filter

A screenshot of a Mac OS X desktop showing a Safari browser window. The title bar says "localhost". The address bar shows "CreateTable - Amazon DynamoDB". The main content area is titled "DynamoDB JavaScript Shell". On the left, there is a code editor with the following JavaScript code:

```
1 var params = {
2   TableName: "CELEBRITIES_CONSOLE",
3   ProjectionExpression: "Field",
4   KeyConditionExpression: "FullName = :fullname",
5   FilterExpression: "FullName >= :fullname",
6   ExpressionAttributeValues: {
7     ":fullname": "Marie-Curie"
8   },
9 }
10 dynamodb.query(params, function(err, data) {
11   if (err)
12     console.log(JSON.stringify(err, null, 2));
13   else
14     console.log(JSON.stringify(data, null, 2));
15 });

```
"retryable": false
```
}
>
```
1 var params = {};
2
3 dynamodb.query(params, function(err, data) {
4   if (err)
5     console.log(JSON.stringify(err, null, 2));
6   else
7     console.log(JSON.stringify(data, null, 2));
8 });
```
}
>
{
  "Items": [
    {
      "Field": "Physics"
    }
  ],
  "Count": 1,
  "ScannedCount": 1
}
```
}
>
```

The right side of the interface shows a preview pane with a single item from the query result:

```
{
  "Field": "Physics"
}
```

Below the preview pane, there are several icons for interacting with the code and results.

At the bottom of the browser window, there are links to "DynamoDB Developer Guide", "AWS JavaScript SDK API Reference", and "DynamoDB API Reference".

### Query using Scan

A screenshot of a Mac OS X desktop showing a Safari browser window. The title bar says "localhost". The address bar shows "CreateTable - Amazon DynamoDB". The main content area is titled "DynamoDB JavaScript Shell". On the left, there is a code editor with the following JavaScript code:

```
1 var params = {
2   TableName: "CELEBRITIES_CONSOLE"
3 };
4
5 dynamodb.scan(params, function(err, data) {
6   if (err)
7     console.log(JSON.stringify(err, null, 2));
8   else
9     console.log(JSON.stringify(data, null, 2));
10 });

```
<
```
}
>
{
  {
    "URLPic": "https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpeg",
    "Field": "Physics",
    "URLRes": "https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx",
    "Year": 1903,
    "PopularMovie": "none",
    "FullName": "Marie-Curie"
  },
  {
    "URLPic": "https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg",
    "Field": "none",
    "URLRes": "https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx",
    "Year": 0,
    "PopularMovie": "Tootsie",
    "FullName": "Dustin-Hoffman"
  },
  {
    "URLPic": "https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg",
    "Field": "none",
    "URLRes": "https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx"
  }
}
```
}
>
```

The right side of the interface shows a preview pane with three items from the scan result:

```

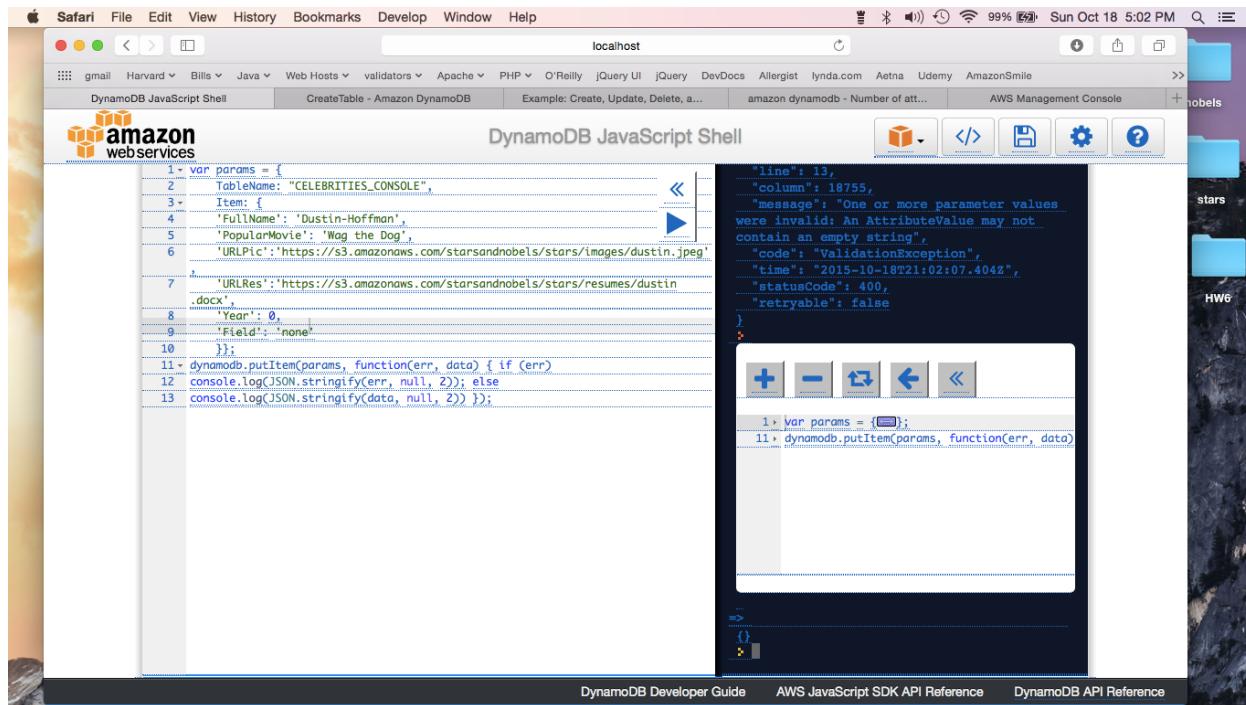
{
  {
    "URLPic": "https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpeg",
    "Field": "Physics",
    "URLRes": "https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx",
    "Year": 1903,
    "PopularMovie": "none",
    "FullName": "Marie-Curie"
  },
  {
    "URLPic": "https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg",
    "Field": "none",
    "URLRes": "https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx",
    "Year": 0,
    "PopularMovie": "Tootsie",
    "FullName": "Dustin-Hoffman"
  },
  {
    "URLPic": "https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg",
    "Field": "none",
    "URLRes": "https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx"
  }
}
```

Below the preview pane, there are several icons for interacting with the code and results.

At the bottom of the browser window, there are links to "DynamoDB Developer Guide", "AWS JavaScript SDK API Reference", and "DynamoDB API Reference".

## HW6\_ScullyMarnie

Demonstrate the ability to update a row



The screenshot shows a Safari browser window with the title bar "Safari File Edit View History Bookmarks Develop Window Help". The address bar shows "localhost" and the tab "DynamoDB JavaScript Shell". The main content area displays the "DynamoDB JavaScript Shell" interface. On the left, there is a code editor with the following JavaScript code:

```
1 - var params = {  
2 -   TableName: "CELEBRITIES_CONSOLE",  
3 -   Item: {  
4 -     'FullName': 'Dustin-Hoffman',  
5 -     'PopularMovie': 'Wag the Dog',  
6 -     'URLPic': 'https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg',  
7 -     'URLRes': 'https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx',  
8 -     'Year': 0,  
9 -     'Field': 'none'  
10 -   };  
11 - dynamodb.putItem(params, function(err, data) { if (err)  
12 -   console.log(JSON.stringify(err, null, 2)); else  
13 -   console.log(JSON.stringify(data, null, 2));});
```

On the right, the terminal window shows the error output:

```
"line": 13,  
"column": 18755,  
"message": "One or more parameter values were invalid: AnAttributeValue may not contain an empty string",  
"code": "ValidationException",  
"time": "2015-10-18T21:02:07.404Z",  
"statusCode": 400,  
"retryable": false  
}
```

Below the terminal are several small icons for navigation and file operations. At the bottom of the shell window, there are links to "DynamoDB Developer Guide", "AWS JavaScript SDK API Reference", and "DynamoDB API Reference".

And Now the same problem with the AWS DynamoDB console (because I'm not sure which one you want honestly)

### Choose DynamoDB

#### Database

-  **RDS**  
Managed Relational Database Service
-  **DynamoDB**  
Predictable and Scalable NoSQL Data Store
-  **ElastiCache**  
In-Memory Cache
-  **Redshift**  
Managed Petabyte-Scale Data Warehouse Service

#### Networking

# HW6\_ScullyMarnie

## Click “Create Table”

The screenshot shows the "Amazon DynamoDB Getting Started" page. At the top, there is a section titled "To start using Amazon DynamoDB, create a table" with a prominent "Create Table" button. Below this, a note states: "Note: Your table will be created in the US East (N. Virginia) region." To the right, there is an "Additional Resources" sidebar with links to various guides and forums. Further down, there is a "How do I create a table?" section with three steps: 1. Pick Primary Key (with a database icon), 2. Set Provisioned Throughput (with an envelope icon), and 3. Create your table with alarms (with a database plus icon). Each step has a "Learn More" link below it. On the far right, there is a "Watch the video" section featuring a thumbnail for an "Introduction to Amazon DynamoDB" video.

Name the table CELEBRITIES\_CONSOLE with a hash type primary key for the full name of the celebrity

The screenshot shows the "Create DynamoDB table" wizard. The first step is to enter the "Table name\*" field, which contains "CELEBRITIES\_CONSOLE". The "Primary key\*" field is set to "Item key" with the attribute "FullName" of type "String". There is also an unchecked option to "Add sort key". Below this, the "Table settings" section provides default settings for the table. It includes a checked checkbox for "Use default settings" and a list of three items: "No secondary indexes.", "Provisioned capacity set to 5 reads and 5 writes.", and "Basic alarms with 80% upper threshold using SNS topic "dynamodb"".

## HW6\_ScullyMarnie

The screenshot shows the AWS DynamoDB 'Create table' wizard. At the top, there's a 'Table settings' section with a note about default settings and a checkbox for 'Use default settings'. Below it is a 'Secondary indexes' section with a table header and a '+ Add index' button. The main part is 'Provisioned capacity' with fields for 'Read capacity units' (1 Table) and 'Write capacity units' (1 Table), and an 'Estimated cost' of \$0.59 / month. A note about additional charges follows. At the bottom are 'Cancel' and 'Create' buttons.

## Create the items in the DB

The screenshot shows the AWS DynamoDB 'Copy item' dialog. It displays a tree view of five items under 'Item {5}': 'FullName', 'MostPopularMovie', 'Name', 'URLPic', and 'URLRes'. The 'URLRes' item has its value highlighted in yellow. The background shows the 'CELEBRITIES\_CONSOLE' table with one item listed: 'URLPic' with value 'https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg'. At the bottom are 'Cancel' and 'Save' buttons.

# HW6\_ScullyMarnie

After inserting all records explore the table in the console

FullName	MostPopularMovie	Name	URLPic	URLRes	Field	Year
Marilyn-Monroe	Some Like it Hot	Marilyn Monroe	https://s3.amazo...	https://s3.amazo...		
Marie-Curie	none	Marie-Curie	https://s3.amazo...	https://s3.amazo...	Physics	1903
Dustin-Hoffman	Tootsie	Dustin Hoffman	https://s3.amazo...	https://s3.amazo...		
Eugene O'Neil	none	Eugene-O'Neill	https://s3.amazo...	https://s3.amazo...	Literature	1936
George Bernard Sh	none	George-Bernard...	https://s3.amazo...	https://s3.amazo...	Literature	1925
Meryl-Streep	Out of Africa	Meryl Streep	https://s3.amazo...	https://s3.amazo...		

Search by filters on ID

Scan filter by Field = "Literature"

FullName	Field	MostPopularM	Name	URLPic	URLRes	Year
Eugene O'Neil	Literature	none	Eugene-O'Neill	https://s3.am...	https://s3.am...	1936
George Bernard	Literature	none	George-Bern...	https://s3.am...	https://s3.am...	1925

## HW6\_ScullyMarnie

Query get by ID= “Marilyn-Monroe”

The screenshot shows the AWS DynamoDB Items page for the 'CELEBRITIES\_CONSOLE' table. A modal dialog is open, displaying a query configuration. The 'Item key' dropdown is set to 'FullName'. The value is 'String' and the condition is 'Equal to' with the value 'Marilyn-Monroe'. Below the modal, the table header includes columns: FullName, MostPopularM, Name, URLPic, and URLRes. A single item is listed: Marilyn Monroe, with the URLPic and URLRes values partially visible as 'https://s3.am...'. The status bar at the bottom indicates 'Viewing 1 to 1 items'.

This screenshot shows the same AWS DynamoDB Items page after the query has been executed. The modal dialog is no longer present. The table now displays one item: Marilyn Monroe, with the URLPic and URLRes values partially visible as 'https://s3.am...'. The status bar at the bottom indicates 'Viewing 1 to 1 items'.

# HW6\_ScullyMarnie

## Marie Curie before Update

The screenshot shows the AWS DynamoDB console for the 'CELEBRITIES\_CONSOLE' table. The table has the following data:

FullName	MostPopularMovie	Name	URLPic	URLRes	Field	Year
Marilyn-Monroe	Some Like It ...	Marilyn Monroe	https://s3.am...	https://s3.am...		
Marie-Curie	none	Marie-Curie	https://s3.am...	https://s3.am...	Physics	1903
Dustin-Hoffman	Tootsie	Dustin Hoffman	https://s3.am...	https://s3.am...		
Eugene O'Neill	none	Eugene-O'Neill	https://s3.am...	https://s3.am...	Literature	1936
George Bernard Shaw	none	George-Bern...	https://s3.am...	https://s3.am...	Literature	1925
Meryl-Streep	Out of Africa	Meryl Streep	https://s3.am...	https://s3.am...		

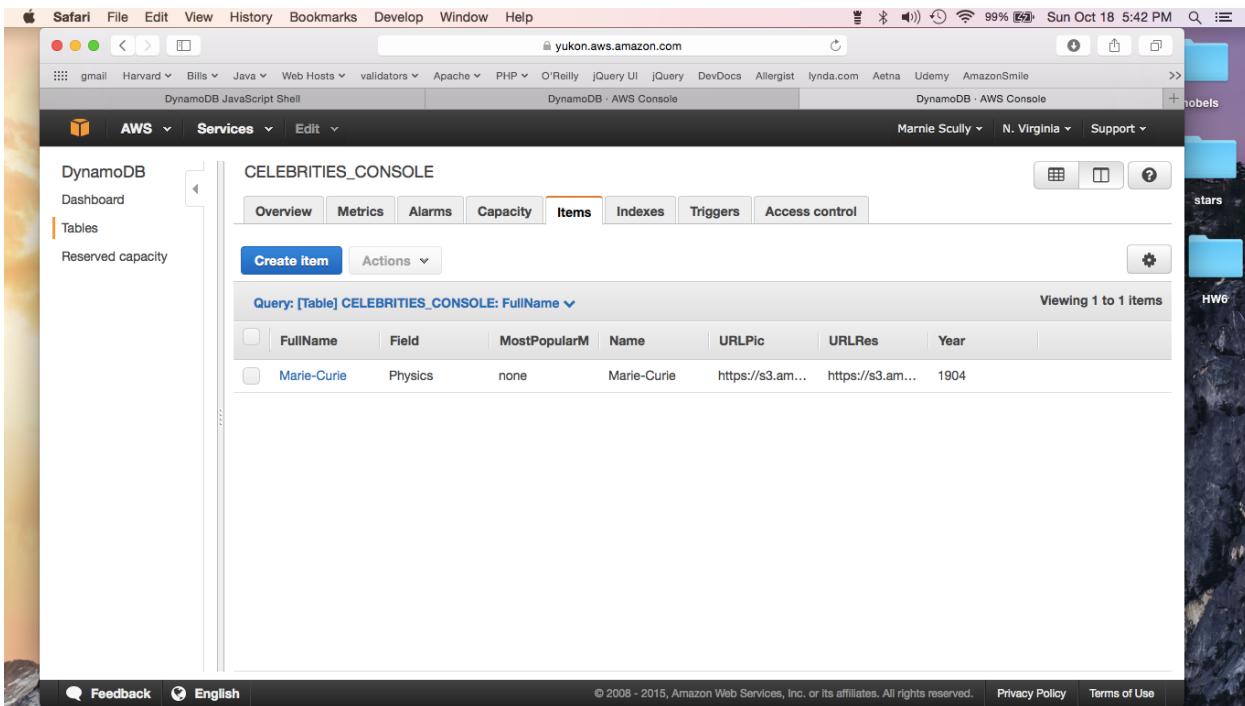
## Editing

The screenshot shows the 'Edit item' dialog for the Marie-Curie entry. The 'Field' value is being updated from 'Physics' to 'Chemistry'. The other fields remain the same.

Field	Type	Value
Field	String	Chemistry
FullName	String	Marie-Curie
MostPopularMovie	String	none
Name	String	Marie-Curie
URLPic	String	https://s3.amazonaws.com/starsandnobels/nobelimages/curie.jpeg
URLRes	String	https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx
Year	Number	1904

## HW6\_ScullyMarnie

After Update



The screenshot shows a Safari browser window displaying the AWS DynamoDB Items page for the 'CELEBRITIES\_CONSOLE' table. The URL is 'yukon.aws.amazon.com/DynamoDB/AWS\_Console'. The table has columns: FullName, Field, MostPopularM, Name, URLPic, URLRes, and Year. A single item is listed: Marie-Curie (Field: Physics, MostPopularM: none, Name: Marie-Curie, URLPic: https://s3.am..., URLRes: https://s3.am..., Year: 1904). The interface includes tabs for Overview, Metrics, Alarms, Capacity, Items (selected), Indexes, Triggers, and Access control, along with buttons for Create Item and Actions.

### Problem 3.

Use AWS SDK (Java or any other) to do the same work as in Problem 2, with minor changes:

- a) Same - except name the table "CELEBRITIES\_SDK"
- b) Same
- c) Same
- d) Query both CELEBRITIES\_CONSOLE and CELEBRITIES\_SDK tables via SDK APIs - demonstrate that you see the inserted data
- e) Same

Provide working code and capture all stages of testing. **[25 Points]**

#### DynamoDBSample.java

```
import java.util.HashMap;
import java.util.Map;

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.AWS Credentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
```

## HW6\_ScullyMarnie

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.ComparisonOperator;
import com.amazonaws.services.dynamodbv2.model.Condition;
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;
import com.amazonaws.services.dynamodbv2.model.DescribeTableRequest;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.PutItemRequest;
import com.amazonaws.services.dynamodbv2.model.PutItemResult;
import com.amazonaws.services.dynamodbv2.model.ScalarAttributeType;
import com.amazonaws.services.dynamodbv2.model.ScanRequest;
import com.amazonaws.services.dynamodbv2.model.ScanResult;
import com.amazonaws.services.dynamodbv2.model.TableDescription;
import com.amazonaws.services.dynamodbv2.util.Tables;
```

```
/**
 * Marnie Scully HW6 Problem 3
 */
public class DynamoDBSample {

    /*
     * Before running the code:
     *   Fill in your AWS access credentials in the provided credentials
     *   file template, and be sure to move the file to the default location
     *   (/Users/marnie/.aws/credentials) where the sample code will load the
     *   credentials from.
     *   https://console.aws.amazon.com/iam/home?#security_credential
     *
     * WARNING:
     *   To avoid accidental leakage of your credentials, DO NOT keep
     *   the credentials file in your source directory.
     */
}
```

```
static AmazonDynamoDBClient dynamoDB;
```

```
/*
 * The only information needed to create a client are security credentials
 * consisting of the AWS Access Key ID and Secret Access Key. All other
 * configuration, such as the service endpoints, are performed
```

## HW6\_ScullyMarnie

```
* automatically. Client parameters, such as proxies, can be specified in an
* optional ClientConfiguration object when constructing a client.
*
* @see com.amazonaws.auth.BasicAWSCredentials
* @see com.amazonaws.auth.ProfilesConfigFile
* @see com.amazonaws.ClientConfiguration
*/
private static void init() throws Exception {
    /*
     * The ProfileCredentialsProvider will return your [marniescully]
     * credential profile by reading from the credentials file located at
     * (/Users/marnie/.aws/credentials).. .
     */
    AWSCredentials credentials = null;
    try {
        credentials = new ProfileCredentialsProvider("marniescully").getCredentials();
    } catch (Exception e) {
        throw new AmazonClientException(
            "Cannot load the credentials from the credential profiles file. " +
            "Please make sure that your credentials file is at the correct " +
            "location (/Users/marnie/.aws/credentials).. , and is in valid format.", e);
    }
    dynamoDB = new AmazonDynamoDBClient(credentials);
    Region usEast1 = Region.getRegion(Regions.US_EAST_1);
    dynamoDB.setRegion(usEast1);
}

public static void main(String[] args) throws Exception {
    init();

    try {
        String tableName = "CELEBRITIES_SDK";

        // Create table if it does not exist yet
        if (Tables.doesTableExist(dynamoDB, tableName)) {
            System.out.println("Table " + tableName + " is already ACTIVE");
        } else {
            // Create a table with a primary hash key named 'name', which holds a string
            CreateTableRequest createTableRequest = new
CreateTableRequest().withTableName(tableName)
                .withKeySchema(new
KeySchemaElement().withAttributeName("name").withKeyType(KeyType.HASH))
```

## HW6\_ScullyMarnie

```
.withAttributeDefinitions(new
AttributeDefinition().withAttributeName("name").withAttributeType(ScalarAttributeType.S))
    .withProvisionedThroughput(new
ProvisionedThroughput().withReadCapacityUnits(1L).withWriteCapacityUnits(1L));
    TableDescription createdTableDescription =
dynamoDB.createTable(createTableRequest).getTableDescription();
    System.out.println("Created Table: " + createdTableDescription);

    // Wait for it to become active
    System.out.println("Waiting for " + tableName + " to become ACTIVE... ");
    Tables.awaitTableToBecomeActive(dynamoDB, tableName);
}

// Describe our new table
DescribeTableRequest describeTableRequest = new
DescribeTableRequest().withTableName(tableName);
    TableDescription tableDescription =
dynamoDB.describeTable(describeTableRequest).getTable();
    System.out.println("Table Description: " + tableDescription);

// Add an item
Map<String, AttributeValue> item = newItem("Marilyn-Monroe", 0, "none", "https://
s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg",
                                              "https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx",
"Some Like it Hot");
    PutItemRequest putItemRequest = new PutItemRequest(tableName, item);
    PutItemResult putItemResult = dynamoDB.putItem(putItemRequest);
    System.out.println("Result: " + putItemResult);

// Add another item
item = newItem("Dustin-Hoffman", 0, "none", "https://s3.amazonaws.com/
starsandnobels/stars/images/dustin.jpeg",
                                              "https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx",
"Tootsie");
    putItemRequest = new PutItemRequest(tableName, item);
    putItemResult = dynamoDB.putItem(putItemRequest);
    System.out.println("Result: " + putItemResult);

// Add another item
item = newItem("Meryl-Streep", 0, "none", "https://s3.amazonaws.com/starsandnobels/
stars/images/meryl.jpeg",
                                              "https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx",
"Out of Africa");
```

## HW6\_ScullyMarnie

```
putItemRequest = new PutItemRequest(tableName, item);
putItemResult = dynamoDB.putItem(putItemRequest);
System.out.println("Result: " + putItemResult);

// Add another item
item = newItem("Marie-Curie", 1903, "Physics", "https://s3.amazonaws.com/
starsandnobels/nobels/images/curie.jpeg",
                "https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx",
                "none");
putItemRequest = new PutItemRequest(tableName, item);
putItemResult = dynamoDB.putItem(putItemRequest);
System.out.println("Result: " + putItemResult);

// Add another item
item = newItem("George-Bernard-Shaw", 1925, "Literature", "https://
s3.amazonaws.com/starsandnobels/nobels/images/shaw.jpeg",
                "https://s3.amazonaws.com/starsandnobels/nobels/resumes/shaw.docx",
                "none");
putItemRequest = new PutItemRequest(tableName, item);
putItemResult = dynamoDB.putItem(putItemRequest);
System.out.println("Result: " + putItemResult);

// Add another item
item = newItem("Eugene-Oneil", 1936, "Literature", "https://s3.amazonaws.com/
starsandnobels/nobels/images/oneil.jpeg",
                "https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx",
                "none");
putItemRequest = new PutItemRequest(tableName, item);
putItemResult = dynamoDB.putItem(putItemRequest);
System.out.println("Result: " + putItemResult);

// Scan CELEBRITIES_SDK
// Scan items for Laureates with a prize year attribute greater than -1 (which would be all)
HashMap<String, Condition> scanFilter = new HashMap<String, Condition>();
Condition condition = new Condition()
    .withComparisonOperator(ComparisonOperator.GT.toString())
    .withAttributeValueList(new AttributeValue().withN("-1"));
scanFilter.put("year", condition);
ScanRequest scanRequest = new ScanRequest(tableName).withScanFilter(scanFilter);
ScanResult scanResult = dynamoDB.scan(scanRequest);
System.out.println("CELEBRITIES_SDK Result: " + scanResult);

// Scan CELEBRITIES_CONSOLE Will only return Laureates
```

## HW6\_ScullyMarnie

```
// Scan items for Laureates with a prize year attribute greater than -1 (which would be all)
tableName = "CELEBRITIES_CONSOLE";
scanFilter = new HashMap<String, Condition>();
condition = new Condition()
    .withComparisonOperator(ComparisonOperator.GT.toString())
    .withAttributeValueList(new AttributeValue().withN("1900"));
scanFilter.put("Year", condition);
scanRequest = new ScanRequest(tableName).withScanFilter(scanFilter);
scanResult = dynamoDB.scan(scanRequest);
System.out.println("CELEBRITIES_CONSOLE Result: " + scanResult);

// Changing the year of nobel prize for Eugene O'Neil
tableName = "CELEBRITIES_SDK";
item = newItem("Eugene-Oneil", 1946, "Literature", "https://s3.amazonaws.com/
starsandnobels/nobels/images/oneil.jpeg",
               "https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx",
               "none");
putItemRequest = new PutItemRequest(tableName, item);
putItemResult = dynamoDB.putItem(putItemRequest);
System.out.println("Changing O'Neil Year: ");

// View CELEBRITIES_SDK to see last update
scanRequest = new ScanRequest(tableName);
scanResult = dynamoDB.scan(scanRequest);
System.out.println("CELEBRITIES_SDK Result: " + scanResult);

} catch (AmazonServiceException ase) {
    System.out.println("Caught an AmazonServiceException, which means your request
made it "
        + "to AWS, but was rejected with an error response for some reason.");
    System.out.println("Error Message: " + ase.getMessage());
    System.out.println("HTTP Status Code: " + ase.getStatusCode());
    System.out.println("AWS Error Code: " + ase.getErrorCode());
    System.out.println("Error Type: " + ase.getErrorType());
    System.out.println("Request ID: " + ase.getRequestId());
} catch (AmazonClientException ace) {
    System.out.println("Caught an AmazonClientException, which means the client
encountered "
        + "a serious internal problem while trying to communicate with AWS,
        + "such as not being able to access the network.");
    System.out.println("Error Message: " + ace.getMessage());
}
```

## HW6\_ScullyMarnie

```
}

// I edited this function to have the fields of the CELEBRITIES_SDK table
private static Map<String,AttributeValue> newItem(String name, int year, String field, String URLPic, String URLRes, String popularMovie) {
    Map<String,AttributeValue> item = new HashMap<String,AttributeValue>();
    item.put("name", new AttributeValue(name));
    item.put("year", new AttributeValue().withN(Integer.toString(year)));
    item.put("field", new AttributeValue(field));
    item.put("URLPic", new AttributeValue(URLPic));
    item.put("URLRes", new AttributeValue(URLRes));
    item.put("popularMovie", new AttributeValue(popularMovie));
    return item;
}

}
```

### Output in Eclipse Console

```
Table CELEBRITIES_SDK is already ACTIVE
Table Description: {AttributeDefinitions: [{AttributeName: name,AttributeType: S}],TableName: CELEBRITIES_SDK,KeySchema: [{AttributeName: name,KeyType: HASH}],TableStatus: ACTIVE,CreationDateTime: Sun Oct 18 18:23:18 EDT 2015,ProvisionedThroughput: {NumberOfDecreasesToday: 0,ReadCapacityUnits: 1,WriteCapacityUnits: 1},TableSizeBytes: 0,ItemCount: 0,TableArn: arn:aws:dynamodb:us-east-1:413513583861:table/CELEBRITIES_SDK,}
Result: {}
CELEBRITIES_SDK Result: {Items: [{URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx}, field={S: Literature}, year={N: 1936}, name={S: Eugene-Oneil}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/oneil.jpeg}, popularMovie={S: none}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx}, field={S: none}, year={N: 0}, name={S: Marilyn-Monroe}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg}, popularMovie={S: Some Like it Hot}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx}, field={S: Physics}, year={N: 1903}, name={S: Marie-Curie}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpeg}, popularMovie={S: none}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx}, field={S: none}}]}
```

## HW6\_ScullyMarnie

```
year={N: 0}, name={S: Dustin-Hoffman}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg}, popularMovie={S: Tootsie}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/shaw.docx}, field={S: Literature}, year={N: 1925}, name={S: George-Bernard-Shaw}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/shaw.jpeg}, popularMovie={S: none}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx}, field={S: none}, year={N: 0}, name={S: Meryl-Streep}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg}, popularMovie={S: Out of Africa}}}, Count: 6, ScannedCount: 6}
```

**CELEBRITIES\_CONSOLE Result:** {Items: [{Field={S: Physics}, URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx}, Year={N: 1904}, MostPopularMovie={S: none}, FullName={S: Marie-Curie}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpeg}, Name={S: Marie-Curie}, {Field={S: Literature}, URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx}, Year={N: 1936}, MostPopularMovie={S: none}, FullName={S: Eugene O'Neil}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/oneil.jpeg}, Name={S: Eugene-Oneil}, {Field={S: Literature}, URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/shaw.docx}, Year={N: 1925}, MostPopularMovie={S: none}, FullName={S: George Bernard Shaw}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/shaw.jpeg}, Name={S: George-Bernard-Shaw}}}, Count: 3, ScannedCount: 6}

**Changing O'Neil Year:**

**CELEBRITIES\_SDK Result:** {Items: [{URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx}, field={S: Literature}, year={N: 1946}, name={S: Eugene-Oneil}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/oneil.jpeg}, popularMovie={S: none}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx}, field={S: none}, year={N: 0}, name={S: Marilyn-Monroe}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg}, popularMovie={S: Some Like it Hot}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx}, field={S: Physics}, year={N: 1903}, name={S: Marie-Curie}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpeg}, popularMovie={S: none}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx}, field={S: none}, year={N: 0}, name={S: Dustin-Hoffman}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg}, popularMovie={S: Tootsie}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/shaw.docx}, field={S: Literature}, year={N: 1925}, name={S: George-Bernard-Shaw}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/shaw.jpeg}, popularMovie={S: none}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx}, field={S: none}, year={N: 0}, name={S: Meryl-Streep}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg}, popularMovie={S: Out of Africa}}}, Count: 3, ScannedCount: 6}

## HW6\_ScullyMarnie

s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg}, popularMovie={S: Out of Africa,}},Count: 6,ScannedCount: 6,}

### ScreenShot of Eclipse Console

The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The title bar indicates the project is "Java - HW6\_3/src/DynamoDBSample.java" and the current date and time are "Sun Oct 18 6:35 PM". The left side features a "Console" view showing command-line output and a "Project Explorer" view showing the file structure. The right side has a "DynamoDBSample.java" editor window displaying Java code. The code is related to DynamoDB operations, specifically scanning a table named "CELEBRITIES\_SDK". The console output shows the results of these operations, including URLs for resumes and the year of the Nobel Prize.

```
<terminated> - DynamoDBSample [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java (Oct 18, 2015, 6:35:34 PM)
Table CELEBRITIES_SDK is already ACTIVE
Table Description: {AttributeDefinitions: [{AttributeName: name,AttributeType: S}],TableName: CELEBRITIES_SDK,KeySchema: [{AttributeName: name,KeyType: HASH}]
Result: []
CELEBRITIES_SDK Result: {Items: [{URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx,}, field={S: Literature,}, year={N: 1982}}]}
CELEBRITIES_CONSOLE Result: {Items: [{Field={S: Physics,}, URLRes={S: https://s3.amazonaws.com/starsandnobels/resumes/curie.docx,}, Year={N: 1903}}]}
Changing O'Neil Year:
CELEBRITIES_SDK Result: {Items: [{URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx,}, field={S: Literature,}, year={N: 1982}}]}
|
```

```
154 .wi
155 .wi
156 scanFil
157 ScanReq
158 ScanRes
159 System.
160
161 // Scan
162 // Scan
163 tableName
164 scanFil
165 conditi
166 .wi
167 .wi
168 scanFil
169 scanReq
170 scanRes
171 System.
172
173 // Chang
174 tableName
175 item =
176 "ht
177 putItem
178 putItem
```

### Problem 4.

Create a Lambda function that will monitor DynamoDB stream of table CELEBRITIES\_SDK and write all operations performed on that table into CloudWatch logs. Use Java class developed in Problem 3 to generate load on table CELEBRITIES\_SDK. Comment out the section of the class where you create the table

## HW6\_ScullyMarnie

itself. Capture all steps of your development and testing. [20 Points]

### Create Lambda Function

Screenshot of the AWS Lambda 'Configure event sources' step. The page title is 'Lambda > New function using blueprint dynamodb-process-stream'. On the left, a sidebar shows 'Step 1: Select blueprint', 'Step 2: Configure event sources' (which is selected), 'Step 3: Configure function', and 'Step 4: Review'. The main content area is titled 'Configure event sources' and contains the following configuration:

- Event source type: DynamoDB
- DynamoDB table: CELEBRITIES\_SDK
- Batch size: 100
- Starting position: Trim horizon

A note at the bottom states: "In order to read from the DynamoDB stream, your execution role must have proper permissions." At the bottom right are 'Cancel', 'Previous', 'Skip', and a blue 'Next' button.

Screenshot of the AWS Lambda 'Step 4. Review' step. The page title is 'DynamoDB - AWS Console'. The main content area shows the configuration for the Lambda function:

**Event sources**  
DynamoDB Batch size: 100, Starting position: TRIM\_HORIZON, DynamoDB table: CELEBRITIES\_SDK

**Enable event source**  
 Enable now  Enable later

**Lambda function**

Name	CELEBRITIES_SDK_LAMBDA
Description	An Amazon DynamoDB trigger that logs the updates made to a table.
Runtime	NodeJS
Handler	index.handler
Role	lambda_dynamo_streams
Memory (MB)	128
Timeout	3

At the bottom right are 'Cancel', 'Previous', and a blue 'Create function' button.

## HW6\_ScullyMarnie

The screenshot shows the AWS Lambda console in a Safari browser window. The URL is `console.aws.amazon.com`. The page title is "AWS Lambda". The navigation bar includes "Services" and "Edit". The main content area shows a Lambda function named "CELEBRITIES\_SDK\_LAMBDA". A green message box at the top says: "Congratulations! Your Lambda function "CELEBRITIES\_SDK\_LAMBDA" has been successfully created and configured with DDB: CELEBRITIES\_SDK as an event source." Below this, there are tabs for "Code", "Configuration", "Event sources", "API endpoints", and "Monitoring". The "Event sources" tab is selected. It displays a table with one row:

Event source	ARN	State	Details
DDB: CELEBRITIES_SDK	arn:aws:dynamodb:us-east-1:413513583861:table/CELEBRITIES_SDK/stream/2015-10-18T22:45:43.767	Enabled	Batch size: 100, Last result: No records processed

At the bottom left, there is a link to "Add event source". The right side of the screen shows a sidebar with various AWS services and a file browser.

## Before Load

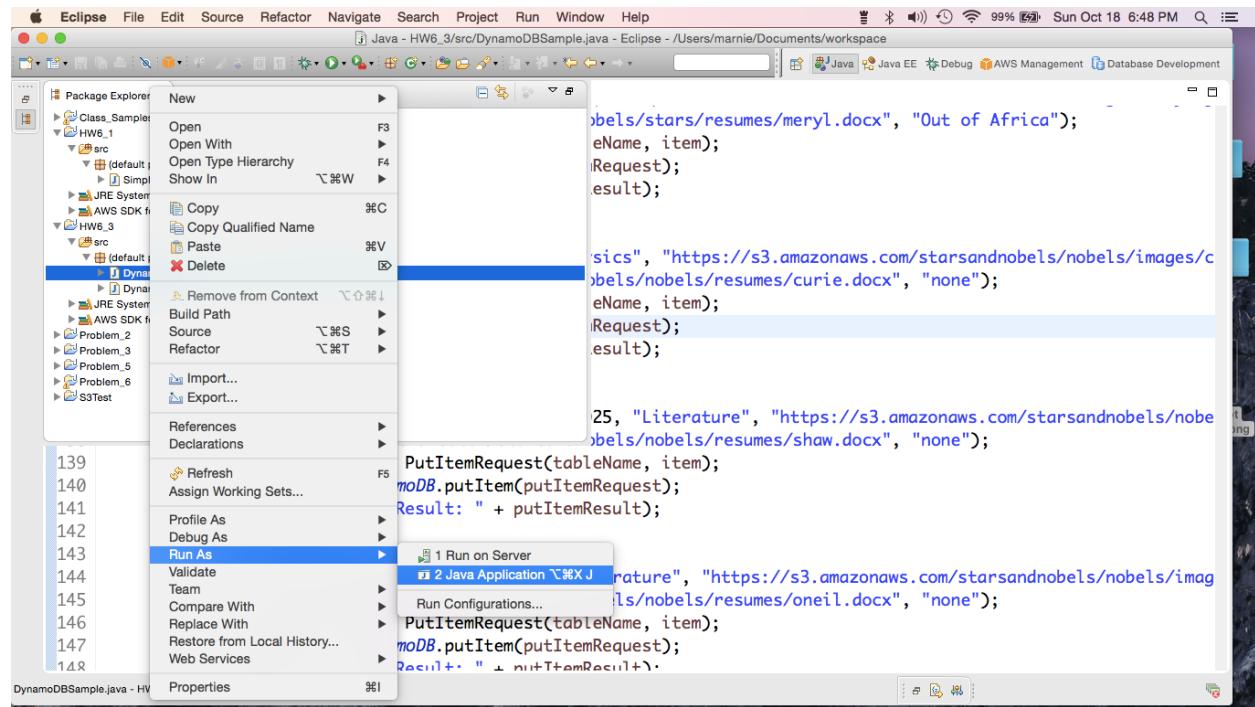
The screenshot shows the "Monitoring" tab of the AWS Lambda console. It displays four CloudWatch metrics charts for the last 24 hours:

- Invocation count:** Shows 1 invocation at 19:00, 0 invocations at 3:00, and 0 invocations at 11:00.
- Invocation duration:** Shows 1 invocation at 19:00, 0 invocations at 3:00, and 0 invocations at 11:00.
- Invocation errors:** Shows 1 invocation at 19:00, 0 invocations at 3:00, and 0 invocations at 11:00.
- Throttled invocations:** Shows 1 invocation at 19:00, 0 invocations at 3:00, and 0 invocations at 11:00.

A link "View logs in CloudWatch" is located above the charts. The right side of the screen shows a sidebar with various AWS services and a file browser.

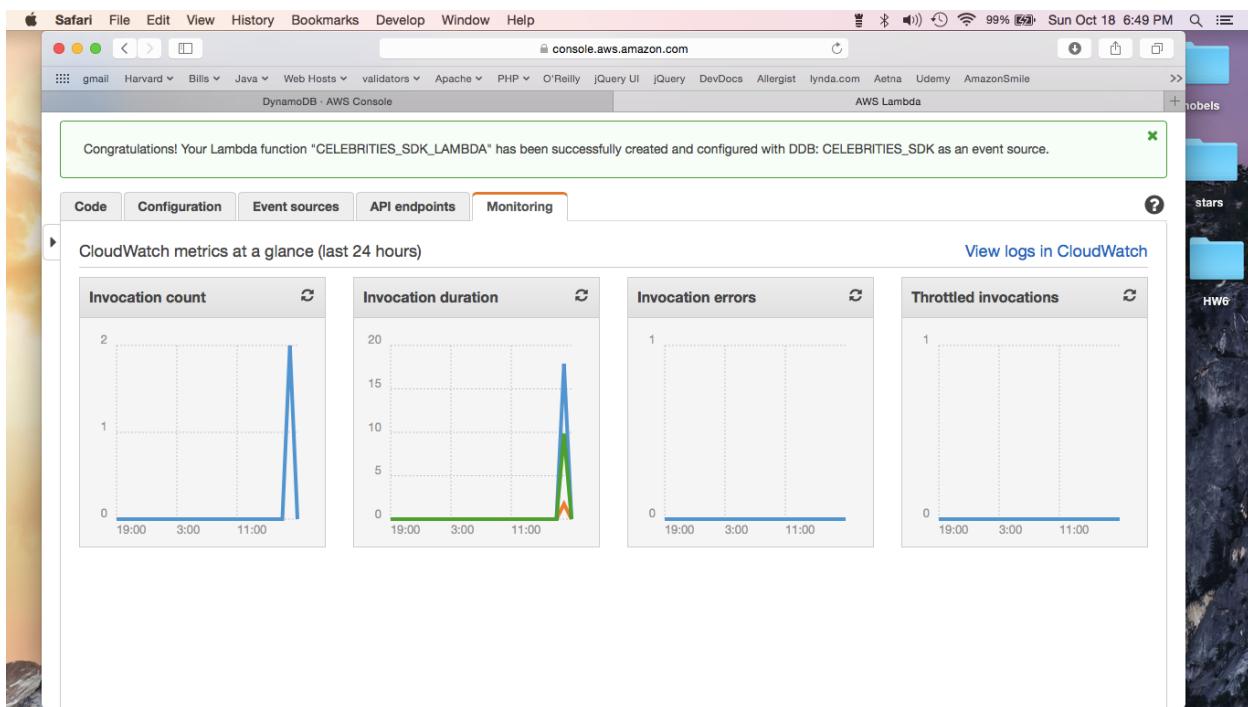
## HW6\_ScullyMarnie

Run Code From Problem 3 again



## HW6\_ScullyMarnie

### CloudWatch after Problem 3 ran



### Problem 5.

When done, delete all DynamoDB tables and Lambda functions, programmatically or using AWS CLI. Provide working code and capture all stages of testing. **[10 Points]**

### Delete Lambda Function from AWS CLI

Delete Tables programmatically with DynamoDBTest.java (modified)

```
Marnie's-MacBook-Air:~ marnie$ aws lambda delete-function --function-name CELEBRITIES_SDK_LAMBDA
Marnie's-MacBook-Air:~ marnie$
```

### Delete Tables Programmatically

#### Output in Eclipse

Starting DynamoDbManager...

Connected to DynamoDB OK

Before table deletion:

My DynamoDB tables:

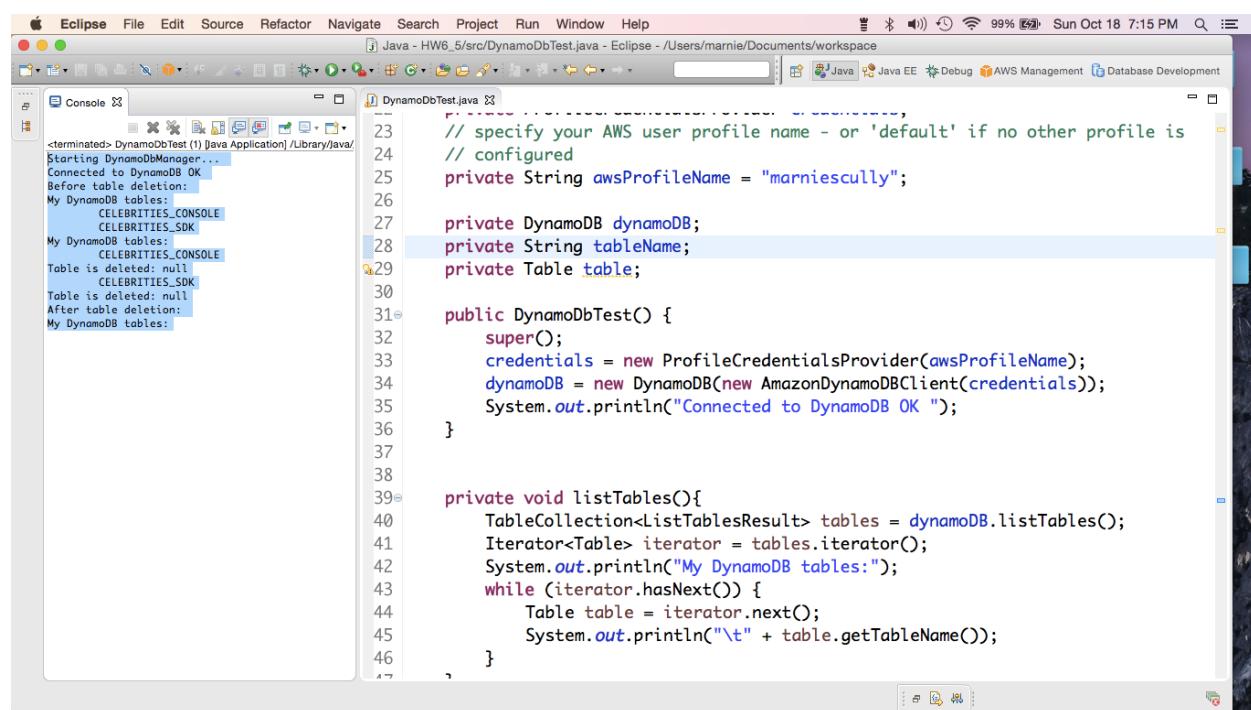
```
CELEBRITIES_CONSOLE
CELEBRITIES_SDK
```

My DynamoDB tables:

## HW6\_ScullyMarnie

```
CELEBRITIES_CONSOLE  
Table is deleted: null  
CELEBRITIES_SDK  
Table is deleted: null  
After table deletion:  
My DynamoDB tables:
```

Screen Shot of Eclipse



## DynamoDBTest.java

```
import java.util.ArrayList;  
import java.util.Iterator;  
  
import com.amazonaws.auth.profile.ProfileCredentialsProvider;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient;  
import com.amazonaws.services.dynamodbv2.document.DynamoDB;  
import com.amazonaws.services.dynamodbv2.document.Item;  
import com.amazonaws.services.dynamodbv2.document.Table;  
import com.amazonaws.services.dynamodbv2.document.TableCollection;  
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;  
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;  
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;  
import com.amazonaws.services.dynamodbv2.model.KeyType;
```

## HW6\_ScullyMarnie

```
import com.amazonaws.services.dynamodbv2.model.ListTablesResult;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.TableDescription;

public class DynamoDbTest {

    private ProfileCredentialsProvider credentials;
    // specify your AWS user profile name - or 'default' if no other profile is
    // configured
    private String awsProfileName = "marniescully";

    private DynamoDB dynamoDB;
    private String tableName;
    private Table table;

    public DynamoDbTest() {
        super();
        credentials = new ProfileCredentialsProvider(awsProfileName);
        dynamoDB = new DynamoDB(new AmazonDynamoDBClient(credentials));
        System.out.println("Connected to DynamoDB OK ");
    }

    private void listTables(){
        TableCollection<ListTablesResult> tables = dynamoDB.listTables();
        Iterator<Table> iterator = tables.iterator();
        System.out.println("My DynamoDB tables:");
        while (iterator.hasNext()) {
            Table table = iterator.next();
            System.out.println("\t" + table.getTableName());
        }
    }

    private void deleteTables() {

        TableCollection<ListTablesResult> tables = dynamoDB.listTables();
        Iterator<Table> iterator = tables.iterator();
        System.out.println("My DynamoDB tables:");
        while (iterator.hasNext()) {
            Table table = iterator.next();
            System.out.println("\t" + table.getTableName());
            table.delete();
            try {
                table.waitForDelete();
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                System.out.println("Failed to delete table: " + tableName);
                e.printStackTrace();
            }
            System.out.println("Table is deleted: " + tableName);
        }
    }
}
```

## HW6\_ScullyMarnie

```
public static void main(String[] args) {  
    System.out.println("Starting DynamoDbManager... ");  
    DynamoDbTest ddb = new DynamoDbTest();  
    System.out.println("Before table deletion: ");  
    ddb.listTables();  
    ddb.deleteTables();  
    System.out.println("After table deletion: ");  
    ddb.listTables();  
    /* */  
}  
}
```