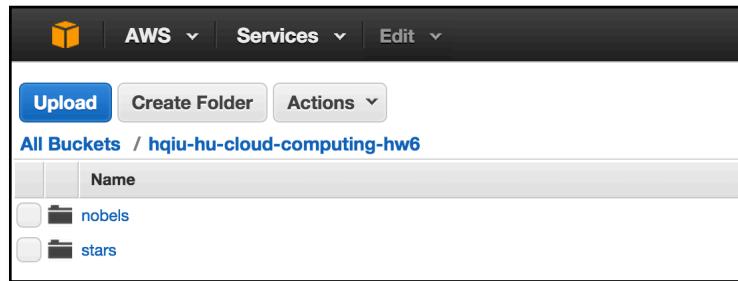
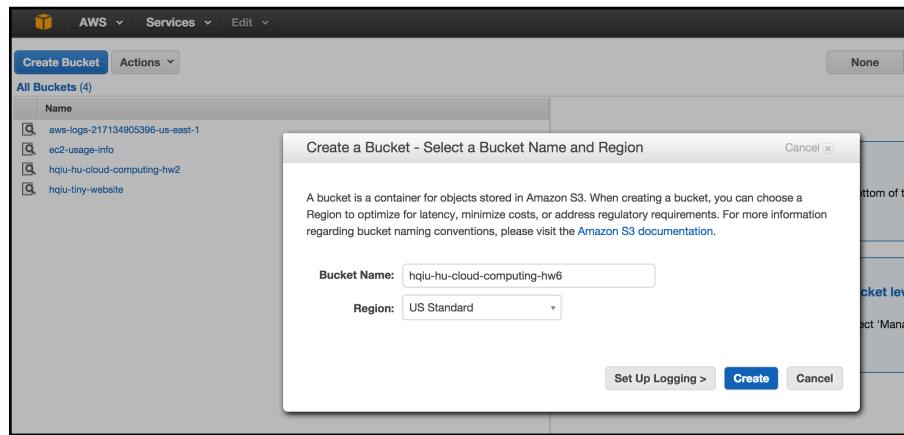


Place all of your narratives and illustrations in a single Word or PDF document named E90_LastNameFirstNameHW06.docx [.pdf]. Use this assignment as the initial template. Add your steps and your code below problem statements used for that problem. Upload your homework file and your working code (e.g., 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**.

Create a S3 bucket from the AWS console. Set the Region to “US Standard”. Create the hierarchical structure in the folder.



- 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.

Create some pictures and fake resumes.



- Upload those images and resume into your S3 bucket manually.
- Manually grant general public access to those images and resumes.

Upload corresponding images and resumes to “stars” and “nobels” folder. In “Properties”->“Permissions”, grant “open”, “view” and “edit” permission to “everyone”.

Screenshot of the AWS S3 console showing the properties of the file 'Img_Angelina_Jolie.jpg'. The 'Permissions' tab is open, showing that the file is currently set to 'Everyone' with 'Open/Download', 'View Permissions', and 'Edit Permissions' checked.

Object: Img_Angelina_Jolie.jpg

Properties

Permissions

Grantee: Everyone Open/Download View Permissions Edit Permissions

Save Cancel

Screenshot of the AWS S3 console showing the properties of the file 'Resume_Angelina_Jolie.docx'. The 'Permissions' tab is open, showing that the file is currently set to 'Everyone' with 'Open/Download', 'View Permissions', and 'Edit Permissions' checked.

Object: Resume_Angelina_Jolie.docx

Properties

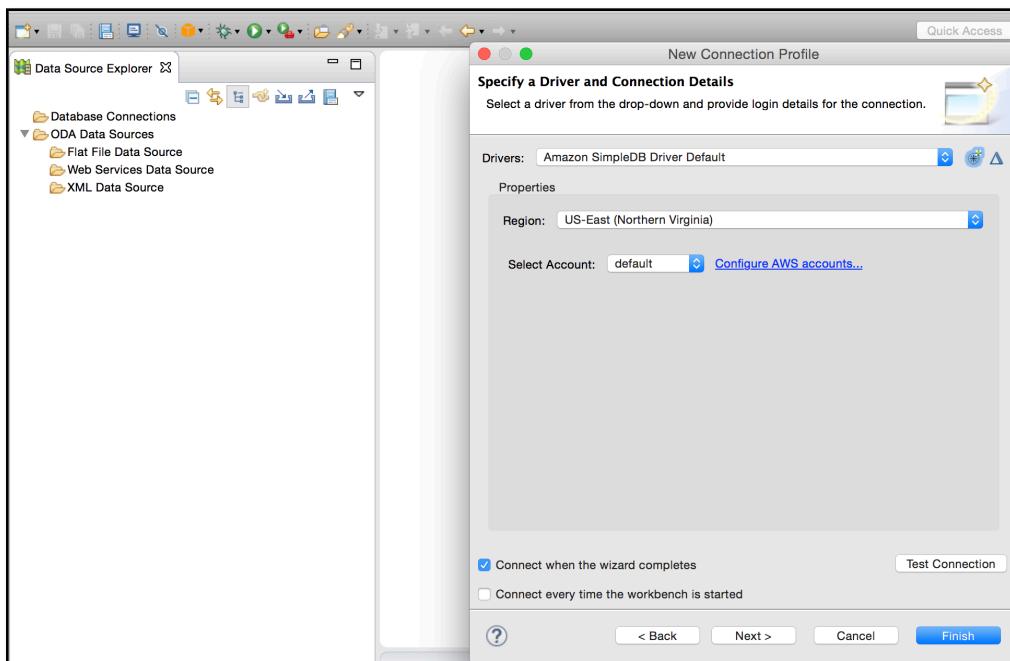
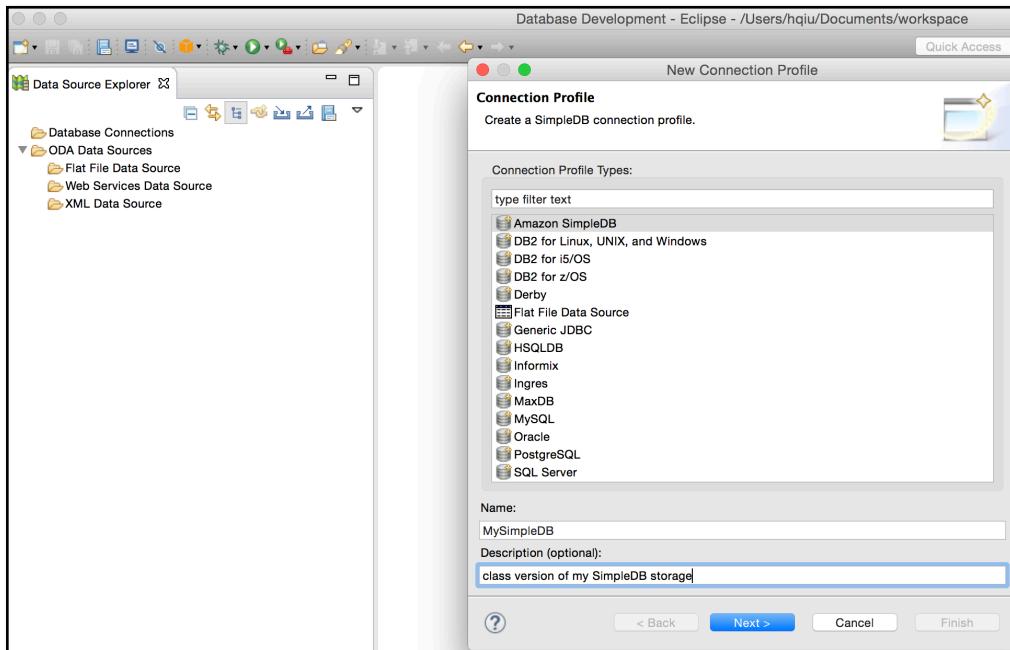
Permissions

Grantee: Everyone Open/Download View Permissions Edit Permissions

Save Cancel

- 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:

Create a new SimpleDB project in Eclipse.



Take the example code as an example. First do some initialization. Set up the credentials for connecting the SimpleDB. I use the “default” credentials here.

```
public class SimpleDB {  
    public static void main(String[] args) throws Exception {  
        AWS Credentials credentials = null;  
        try {  
            credentials = new ProfileCredentialsProvider("default").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/hqiu/.aws/credentials), and is in valid format.",  
                e);  
        }  
        AmazonSimpleDB sdb = new AmazonSimpleDBClient(credentials);  
        Region usEast1 = Region.getRegion(Regions.US_EAST_1);  
        sdb.setRegion(usEast1);  
  
        System.out.println("=====");  
        System.out.println("Getting Started with Amazon SimpleDB");  
        System.out.println("=====\\n");  
    }  
}
```

- 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.

First create a domain in the Simple DB.

```
try {  
    // Create a domain  
    String myDomain = "Celebrities";  
    System.out.println("Creating domain called " + myDomain + ".\\n");  
    sdb.createDomain(new CreateDomainRequest(myDomain));  
  
    // List domains  
    System.out.println("Listing all domains in your account:\\n");  
    for (String domainName : sdb.listDomains().getDomainNames()) {  
        System.out.println(" " + domainName);  
    }  
    System.out.println();  
}
```

Insert three rows for movie stars and another three rows for Nobel laureates using the API “BatchPutAttributesRequest()”. The primary key of the domain is the full names of the celebrities with hyphens between the first and last name. In function “createSampleData()”, the parameters in the “ReplaceableItem ()” is the primary key. The attributes are set by “ReplaceableAttribute()”.

```
// Put data into a domain  
System.out.println("Putting data into " + myDomain + " domain.\\n");  
sdb.batchPutAttributes(new BatchPutAttributesRequest(myDomain, createSampleData()));
```

```

/* 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("Angelina-Jolie").withAttributes(
        new ReplaceableAttribute("Name", "Angelina Jolie Pitt", true),
        new ReplaceableAttribute("Movie", "Lara Croft: Tomb Raider", true),
        new ReplaceableAttribute("PictureURL", "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_Angelin",
        new ReplaceableAttribute("ResumeURL", "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/resumes/Resume_AngelinaJolie"));

    sampleData.add(new ReplaceableItem("Audrey-Hepburn").withAttributes(
        new ReplaceableAttribute("Name", "Audrey Hepburn", true),
        new ReplaceableAttribute("Movie", "Roman Holiday", true),
        new ReplaceableAttribute("PictureURL", "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_AudreyHepburn",
        new ReplaceableAttribute("ResumeURL", "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/resumes/Resume_AudreyHepburn"));

    sampleData.add(new ReplaceableItem("Jennifer-Aniston").withAttributes(
        new ReplaceableAttribute("Name", "Jennifer Joanna Aniston", true),
        new ReplaceableAttribute("Movie", "The Good Girl", true),
        new ReplaceableAttribute("PictureURL", "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_JenniferAniston",
        new ReplaceableAttribute("ResumeURL", "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/resumes/Resume_JenniferAniston"));

    sampleData.add(new ReplaceableItem("Albert-Einstein").withAttributes(
        new ReplaceableAttribute("Name", "Albert Einstein", true),
        new ReplaceableAttribute("Movie", "Any Movie", true),
        new ReplaceableAttribute("PictureURL", "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/images/Img_AlbertEinstein",
        new ReplaceableAttribute("ResumeURL", "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/resumes/Resume_AlbertEinstein",
        new ReplaceableAttribute("YearOfNobel", "1921", true),
        new ReplaceableAttribute("FieldOfScience", "Physics", true)));
}

```

Check that we can select some data that we just inserted.

```

// Select data from a domain
// Notice the use of backticks around the domain name in our select expression.
String selectExpression = "select * from `"+ myDomain + "` where Movie = 'Roman Holiday'";
System.out.println("Selecting: " + selectExpression + "\n");
SelectRequest selectRequest = new SelectRequest(selectExpression);
for (Item item : sdb.select(selectRequest).getItems()) {
    System.out.println(" Item");
    System.out.println(" Name: " + item.getName());
    for (Attribute attribute : item.getAttributes()) {
        System.out.println("     Attribute");
        System.out.println("         Name: " + attribute.getName());
        System.out.println("         Value: " + attribute.getValue());
    }
}
System.out.println();

```

- d. Demonstrate that you can change (correct) the year of one Nobel prize award, programmatically.

Update the “YearOfNobel” of “Albert-Einstein”. First setup a “replaceableAttributes” and then invoke the “putAttributes()” request.

```

// Replace/Change the year of Nobel prize award of a Nobel laureate.
System.out.println("Replacing YearOfNobel of Albert-Einstein with 1922.\n");
List<ReplaceableAttribute> replaceableAttributes = new ArrayList<ReplaceableAttribute>();
replaceableAttributes.add(new ReplaceableAttribute("YearOfNobel", "1922", true));
sdb.putAttributes(new PutAttributesRequest(myDomain, "Albert-Einstein", replaceableAttributes));

```

- e. Demonstrate that you can delete one movie star from SimpleDB domain programmatically.

Delete one movie star using “deleteAttributes()” with the primary key.

```

// Delete a movie star and all of its attributes
System.out.println("Deleting star Jennifer Aniston.\n");
sdb.deleteAttributes(new DeleteAttributesRequest(myDomain, "Jennifer-Aniston"));

```

- f. Capture the content of your database as displayed in the Database Development perspective.

Check from the “database perspective” in Eclipse. We have deleted one actor before, so we only have five celebrities here.

The screenshot shows the Eclipse Database Development perspective with the following components:

- Database Connections:** A tree view showing "MySimpleDB (SimpleDB v. 2009.4.15)" under "Data" > "Domains" > "Celebrities". The "Item Names and Attributes" section lists "YearOfNobel", "ResumeURL", "PictureURL", "Name", "Movie", and "FieldOfScience" as nullable TEXT fields, along with the primary key "itemName()".
- SQL Results (itemName()):** A table showing the names of the five remaining celebrities: Albert-Einstein, Angelina-Jolie, Audrey-Hepburn, Hermann-Muller, and Werner-Heisenberg.
- SQL Results (FieldOfScience):** A table showing the field of science for each celebrity: Physics, NULL, NULL, Physiology or Medicine, and Physics.
- SQL Results (PictureURL):** A table showing the URLs for the five celebrities' pictures: https://s3.amazonaws.com/hqi-hu-cloud-computing-hw6/nobels/images/img_Albert_Einstein.jpg, https://s3.amazonaws.com/hqi-hu-cloud-computing-hw6/stars/images/img_Angelina_Jolie.jpg, https://s3.amazonaws.com/hqi-hu-cloud-computing-hw6/stars/images/img_Audrey_Hepburn.jpg, https://s3.amazonaws.com/hqi-hu-cloud-computing-hw6/nobels/images/img_Hermann_Joseph_Muller.jpg, and https://s3.amazonaws.com/hqi-hu-cloud-computing-hw6/nobels/images/img_Werner_Karl_Heisenberg.jpg.

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]**

Results from the Eclipse console.

The screenshot shows a terminal window within an IDE. The title bar indicates it's a Java application. The console output is as follows:

```

<terminated> MySimpleDB [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Oct 16, 2015, 7:11:12 AM)
=====
Getting Started with Amazon SimpleDB
=====

Creating domain called Celebrities.

Listing all domains in your account:

Celebrities
ElasticMapReduce-2015-05

Putting data into Celebrities domain.

Selecting: select * from `Celebrities` where Movie = 'Roman Holiday'

Item
Name: Audrey-Hepburn
Attribute
Name: Name
Value: Audrey Hepburn
Attribute
Name: PictureURL
Value: https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_Audrey_Hepburn.jpg
Attribute
Name: Movie
Value: Roman Holiday
Attribute
Name: ResumeURL
Value: https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/resumes/Resume_Audrey_Hepburn.docx

Replacing YearOfNobel of Albert-Einstein with 1922.

Deleting star Jennifer Aniston.

```

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")

The screenshot shows the 'Create DynamoDB table' wizard. The first step is completed, showing the table name 'CELEBRITIES_CONSOLE' and a primary key 'Item key' named 'CelebrityName' of type String. The second step, 'Table settings', is shown with the 'Use default settings' checkbox checked. The note below states: '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.' The note also specifies: '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.' At the bottom are 'Cancel' and 'Create' buttons.

Create the Dynamo DB table, table name is “CELEBRITIES_CONSOLE”, primary key is “CelebrityName”. The table setting (provisioned throughput, etc.) uses the default settings here. Once the table is created, enable the “Manage Streams”.

The screenshot shows the AWS DynamoDB console with the table "CELEBRITIES_CONSOLE" selected. A modal dialog titled "Manage Stream" is open, showing options for "View type": "New and old images" (selected), "Keys only - only the key attributes of the modified item", "New image - the entire item, as it appears after it was modified", and "Old image - the entire item, as it appeared before it was modified". At the bottom right of the dialog are "Cancel" and "Enable" buttons. The "Enable" button is highlighted in blue.

The screenshot shows the "Stream details" section for the "CELEBRITIES_CONSOLE" table. It displays the following information:

- Stream enabled:** Yes
- View type:** New and old images
- Latest stream ARN:** arn:aws:dynamodb:us-east-1:217134905396:table/CELEBRITIES_CONSOLE/stream/2015-10-15T20:32:28.777

A "Manage Stream" button is located at the bottom of this section.

The screenshot shows the "Overview" tab for the "CELEBRITIES_CONSOLE" table. It displays the following table details:

Table name	CELEBRITIES_CONSOLE
Primary item key	CelebrityName (String)
Primary sort key	-
Table status	Updating
Creation date	October 15, 2015 at 4:31:10 PM UTC-4
Provisioned read capacity units	5
Provisioned write capacity units	5
Last decrease time	-
Last increase time	-
Storage size (in bytes)	0 bytes
Item count	0
Region	US East (N. Virginia)
Amazon Resource Name (ARN)	arn:aws:dynamodb:us-east-1:217134905396:table/CELEBRITIES_CONSOLE

A note at the bottom states: "Storage size and item count are not updated in real-time. They are updated periodically, roughly every six hours."

Set the Access Control. Enable all the APIs when “Login with Amazon”.

```

1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Action": [
7          "dynamodb:BatchGetItem",
8          "dynamodb:BatchWriteItem",
9          "dynamodb:DeleteItem",
10         "dynamodb:GetItem",
11         "dynamodb:PutItem",
12         "dynamodb:Query",
13         "dynamodb:UpdateItem"
14       ],
15       "Resource": [
16         "arn:aws:dynamodb:us-east-1:217134905396:table/CELEB
17       ],
18       "Condition": {
19         "ForAllValues:StringEquals": {
20           "dynamodb:LeadingKeys": [
21             ...
22           ]
23         }
24       }
25     }
26   }
  
```

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

In the dashboard, select “Tables”, we will see the table we just created.

Name	CelebrityName
CELEBRITIES_CONSOLE	Angelina Jolie

Select “Create Item” and create new items into the table. Click “Save”, we will see the item we just created immediately.

CelebrityName	FullName	Movie	PictureURL	ResumeURL
Angelina-Jolie	Angelina Jolie Pitt	Lara Croft: Tomb ...	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/images/Img_Angelina_Jolie.jpg	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/resumes/Resume_Angelina_Jolie.docx

We can copy the item and edit it to another new item.

Copy item

Tree ▾

- Item {5}
 - CelebrityName String : Audrey-Hepburn
 - FullName String : Audrey Hepburn
 - Movie String : Roman Holiday
 - PictureURL String : https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/images/Img_Audrey_Hepburn.jpg
 - ResumeURL String : https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/resumes/Resume_Audrey_Hepburn.docx

Cancel Save

Add three stars and three Nobel laureates. The Nobel laureates have two more attributes “YearOfNobel” and “FieldOfScience”.

CelebrityName	FullName	Movie	PictureURL	ResumeURL	FieldOfScience	YearOfNobel
Werner-Heisenberg	Werner Karl Heisenberg	Any Movie	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/images/Img_Werner_Heisenberg.jpg	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/resumes/Resume_Werner_Heisenberg.docx	Physics	1932
Hermann-Muller	Hermann Joseph Muller	Any Movie	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/images/Img_Hermann_Muller.jpg	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/resumes/Resume_Hermann_Muller.docx	Physiology or Medicine	1946
Albert-Einstein	Albert Einstein	Any Movie	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/images/Img_Albert_Einstein.jpg	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/resumes/Resume_Albert_Einstein.docx	Physics	1921
Jennifer-Aniston	Jennifer Joanna Aniston	The Good Girl	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/images/Img_Jennifer_Aniston.jpg	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/resumes/Resume_Jennifer_Aniston.docx		
Audrey-Hepburn	Audrey Hepburn	Roman Holiday	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/images/Img_Audrey_Hepburn.jpg	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/resumes/Resume_Audrey_Hepburn.docx		
Angelina-Jolie	Angelina Jolie Pitt	Lara Croft: Tomb ...	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/images/Img_Angelina_Jolie.jpg	https://s3.amazonaws.com/hgiu-hu-cloud-computing-hw6/stars/resumes/Resume_Angelina_Jolie.docx		

d) Explore the table in the Console and search by providing filters on ID - both scans and get by ID types of queries

First try to scan the table and give some searching criteria. We want to filter the item whose “CelebrityName” equals to “Jennifer-Aniston”. Hit “Start”, only one item will be filtered out.

CELEBRITIES_CONSOLE

Scan: [Table] CELEBRITIES_CONSOLE: CelebrityName ^

Filter: CelebrityName String Equal to Jennifer-Aniston

CelebrityName	FullName	Movie	PictureURL	ResumeURL
Jennifer-Aniston	Jennifer Joan...	The Good Girl	https://s3.am...	https://s3.am...

CELEBRITIES_CONSOLE

Scan: [Table] CELEBRITIES_CONSOLE: CelebrityName ^

Viewing 1 to 1 items

CelebrityName	FullName	Movie	PictureURL	ResumeURL
Jennifer-Aniston	Jennifer Joan...	The Good Girl	https://s3.am...	https://s3.am...

Try to search the items whose “YearOfNobel” is less than 1940.

CELEBRITIES_CONSOLE

Query: [Table] CELEBRITIES_CONSOLE: CelebrityName ^

Filter: YearOfNobel Number Less than 1940

CelebrityName	FieldOfScience	FullName	Movie	PictureURL	ResumeURL	YearOfNobel
Werner-Heisen	Physics	Werner Karl ...	Any Movie	https://s3.am...	https://s3.am...	1932
Albert-Einstein	Physics	Albert Einstein	Any Movie	https://s3.am...	https://s3.am...	1921

CELEBRITIES_CONSOLE

Scan: [Table] CELEBRITIES_CONSOLE: CelebrityName ^

Viewing 1 to 2 items

CelebrityName	FieldOfScience	FullName	Movie	PictureURL	ResumeURL	YearOfNobel
Werner-Heisen	Physics	Werner Karl ...	Any Movie	https://s3.am...	https://s3.am...	1932
Albert-Einstein	Physics	Albert Einstein	Any Movie	https://s3.am...	https://s3.am...	1921

For the Query, we should give a primary key while doing query. The functionality of filter is identical to the previous one.

CELEBRITIES_CONSOLE

Overview Metrics Alarms Capacity **Items** Indexes Triggers Access control

Create item Actions ▾

Query: [Table] CELEBRITIES_CONSOLE: CelebrityName ▾ Viewing 1 to 1 items

Query [Table] CELEBRITIES_CONSOLE: CelebrityName

Item key CelebrityName String Equal to Audrey-Hepburn

Add filter

Sort Ascending Descending

Attributes All Projected

Start Begins a new search Cancel changes

CELEBRITIES_CONSOLE

Overview Metrics Alarms Capacity **Items** Indexes Triggers Access control

Create item Actions ▾

Query: [Table] CELEBRITIES_CONSOLE: CelebrityName ▾ Viewing 1 to 1 items

CelebrityName	FullName	Movie	PictureURL	ResumeURL
Audrey-Hepburn	Audrey Hepb...	Roman Holiday	https://s3.am...	https://s3.am...

- e) Update an award year of one of the Nobles rows - demo that the data is updated.
Capture all stages of testing. **[15 Points]**

CELEBRITIES_CONSOLE

Overview Metrics Alarms Capacity **Items** Indexes Triggers Access control

Create item Actions ▾

Scan: [Table] CELEBRITIES_CONSOLE: CelebrityName ▾ Viewing 1 to 6 items

Duplicate Edit Delete

CelebrityName	FullName	Movie	PictureURL	ResumeURL	FieldOfScience	YearOfNobel
Audrey-Hepburn	Audrey Hepb...	Roman Holiday	https://s3.am...	https://s3.am...		
Jennifer-Aniston	Jennifer Joan...	The Good Girl	https://s3.am...	https://s3.am...		
Angelina-Jolie	Angelina Joli...	Lara Croft: T...	https://s3.am...	https://s3.am...		
Werner-Heisenberg	Werner Karl ...	Any Movie	https://s3.am...	https://s3.am...	Physics	1932
Hermann-Muller	Hermann Jos...	Any Movie	https://s3.am...	https://s3.am...	Physiology or...	1946
Albert-Einstein	Albert Einstein	Any Movie	https://s3.am...	https://s3.am...	Physics	1921

Edit item

Tree ▾ Item (7)

- Albert-Einstein String : Albert-Einstein
- FieldOfScience String : Physics
- FullName String : Albert Einstein
- Movie String : Any Movie
- PictureURL String : https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/img_Albert-Einstein.jpg
- ResumeURL String : https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/resume/Resume_Albert-Einstein.docx
- YearOfNobel Number : 1920

Cancel Save

We have two ways to edit the table. We can either select “Actions”->“Edit” to modify the value, or click on the value to change it. Either way can change “YearOfNobel” from “1921” to “1920”.

<input type="checkbox"/>	Hermann-Muller	Hermann Jos...	Any Movie	https://s3.am...	https://s3.am...	Physiology or...	1946
<input checked="" type="checkbox"/>	Albert-Einstein	Albert Einstein	Any Movie	https://s3.am...	https://s3.am...	1920	1921

<input type="checkbox"/>	Hermann-Muller	Hermann Jos...	Any Movie	https://s3.am...	https://s3.am...	Physiology or...	1946
<input type="checkbox"/>	Albert-Einstein	Albert Einstein	Any Movie	https://s3.am...	https://s3.am...	Physics	1920

Problem 3.

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

First create a Dynamo DB project in Eclipse. We can choose the Dynamo DB template. It will generate the template code automatically. The code is the same with the Dynamo DB Sample code. Set up the credential information in the init() step.

```
private static void init() throws Exception {
    /*
     * The ProfileCredentialsProvider will return your [default]
     * credential profile by reading from the credentials file located at
     * (/Users/hqiu/.aws/credentials).
     */
    AWSCredentials credentials = null;
    try {
        credentials = new ProfileCredentialsProvider("default").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/hqiu/.aws/credentials), and is in valid format.",
            e);
    }
    dynamoDB = new AmazonDynamoDBClient(credentials);
    Region usEast1 = Region.getRegion(Regions.US_EAST_1);
    dynamoDB.setRegion(usEast1);
}
```

- a) Same - except name the table "CELEBRITIES_SDK"

Create a table named “CELEBRITIES_SDK” using “CreateTableRequest()”.

```
try {
    tableName = "CELEBRITIES_SDK";

    // Create table if it does not exist yet
    if (Tables.doesTableExist(dynamoDB, tableName)) {
        System.out.println("\n\nTable " + 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("CelebrityName").withKeyType(KeyType.HASH))
            .withAttributeDefinitions(new AttributeDefinition().withAttributeName("CelebrityName").withAttributeType(ScalarAttributeType.STRING))
            .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);
    }
}
```

The “KeySchema” is the primary key “CelebrityName”. It’s a hash type. We can also set the provisioned throughput through this step.

- b) Same
- c) Same

I used two different ways to insert rows into the table. The first method “addStar()” uses Dynamo DB class to get the table, and put the item directly into the table with the primary key. The second method “addNobel()” uses the template code, it invokes the method “PutItemRequest()” of the Dynamo DB client and insert the row with table name.

```
private static void addStar(String keyName, String fullName, String movie, String picURL, String resumeURL) {  
    System.out.println("Adding stars: " + keyName + "/" + fullName + "/" + movie +  
        "/" + picURL + "/" + resumeURL);  
  
    /*  
     Map<String, AttributeValue> item = new HashMap<String, AttributeValue>();  
     item.put("CelebrityName", new AttributeValue(keyName));  
     item.put("Name", new AttributeValue(fullName));  
     item.put("Movie", new AttributeValue(movie));  
     item.put("PictureURL", new AttributeValue(picURL));  
     item.put("ResumeURL", new AttributeValue(resumeURL));  
  
     PutItemRequest putItemRequest = new PutItemRequest(tableName, item);  
     dynamoDB.putItem(putItemRequest);  
    */  
  
    Item item = new Item().withPrimaryKey("CelebrityName", keyName)  
        .withString("Name", fullName)  
        .withString("Movie", movie)  
        .withString("PictureURL", picURL)  
        .withString("ResumeURL", resumeURL);  
    table.putItem(item);  
}  
  
private static void addNobel(String keyName, String fullName, String movie, String picURL, String resumeURL,  
    Integer year, String field) {  
  
    System.out.println("Adding nobel laureates: " + keyName + "/" + fullName + "/" + movie +  
        "/" + picURL + "/" + resumeURL + "/" + year + "/" + field);  
  
    Map<String, AttributeValue> item = new HashMap<String, AttributeValue>();  
    item.put("CelebrityName", new AttributeValue(keyName));  
    item.put("Name", new AttributeValue(fullName));  
    item.put("Movie", new AttributeValue(movie));  
    item.put("PictureURL", new AttributeValue(picURL));  
    item.put("ResumeURL", new AttributeValue(resumeURL));  
    item.put("YearOfNobel", new AttributeValue().withN(Integer.toString(year)));  
    item.put("FieldOfScience", new AttributeValue(field));  
  
    PutItemRequest putItemRequest = new PutItemRequest(tableName, item);  
    dynamoDB.putItem(putItemRequest);  
  
    // PutItemResult putItemResult = dynamoDB.putItem(putItemRequest);  
    // System.out.println("Result: " + putItemResult.toString());  
}
```

```
addStar("Jennifer-Aniston", "Jennifer Joanna Aniston", "The Good Girl",  
    "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_Jennifer_Aniston.jpg",  
    "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/resumes/Resume_Jennifer_Aniston.docx");  
  
addNobel("Albert-Einstein", "Albert Einstein", "Any Movie",  
    "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/images/Img_Albert_Einstein.jpg",  
    "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/resumes/Resume_Albert_Einstein.docx",  
    1921, "Physics");
```

- d) Query both CELEBRITIES_CONSOLE and CELEBRITIES_SDK tables via

SDK APIs - demonstrate that you see the inserted data

First query the data using the “query()” method. Similar to the AWS console, query through the “Query” should give the primary key/schema key, which is the “CelebrityName” here. We can also add some filter criteria here. Since the “YearOfNobel” of “Albert-Einstein” is 1921, it will meet the requirements and be filtered out.

```
// Query items for Nobel laureates with a primary key Albert-Einstein from CELEBRITIES_SDK
QuerySpec spec = new QuerySpec()
    .withHashKey("CelebrityName", "Albert-Einstein")
    .withFilterExpression("YearOfNobel between :v_start_yn and :v_end_yn")
    .withValueMap(new ValueMap()
        .withInt(":v_start_yn", 1920)
        .withInt(":v_end_yn", 1922));
ItemCollection<QueryOutcome> items = table.query(spec);

System.out.println("\nQuery items for Nobel laureates with a primary key Albert-Einstein from table " + tableName + ":");
for (Item item: items) {
    System.out.println(item);
}
```

Query the data using the “Scan” facility. The “ScanRequest” class needs to be created with a table name. The first “scan” will query the data from table “CELEBRITIES_SDK”. It will scan for items whose “YearOfNobel” is less than 1940. The second “scan” will query the data from table “CELEBRITIES_CONSOLE”. It will search for items whose “movie” name is “Roman Holiday”.

```
// Scan items for Nobel laureates with a year attribute smaller than 1940 from CELEBRITIES_SDK
HashMap<String, Condition> scanFilter = new HashMap<String, Condition>();
Condition condition = new Condition()
    .withComparisonOperator(ComparisonOperator.LT)
    .withAttributeValueList(new AttributeValue().withN("1940"));
scanFilter.put("YearOfNobel", condition);
ScanRequest scanRequest = new ScanRequest(tableName).withScanFilter(scanFilter);
ScanResult scanResult = dynamoDB.scan(scanRequest);
System.out.println("\nScan items for Nobel laureates with a YearOfNobel attribute smaller than 1940 from table " + tableName + ":");
System.out.println("Result: " + scanResult);

// Scan items for Stars with a specific movie name from CELEBRITIES_CONSOLE.
scanFilter = new HashMap<String, Condition>();
condition = new Condition()
    .withComparisonOperator(ComparisonOperator.EQ.toString())
    .withAttributeValueList(new AttributeValue().withS("Roman Holiday"));
scanFilter.put("Movie", condition);
scanRequest = new ScanRequest("CELEBRITIES_CONSOLE").withScanFilter(scanFilter);
scanResult = dynamoDB.scan(scanRequest);
System.out.println("\nScan items for Stars with a movie named \"Roman Holiday\" from table CELEBRITIES_CONSOLE:");
System.out.println("Result: " + scanResult);
```

e) Same

Update the item by invoking the method “updateItem()” of the Dynamo DB “table” class.

```
public static void updateYearOfNobel(String keyName, int year) {
    System.out.println("\nUpdate the celebrity's attribute \\'YearOfNobel\\' to " + year + ":");

    System.out.println("Before update, the \\\"YearOfNobel\\\" is " + getYearOfNobel(keyName) + ".");
    table.updateItem("CelebrityName", keyName,
        new AttributeUpdate("YearOfNobel").put(year));

    System.out.println("The newly updated \\\"YearOfNobel\\\" is " + getYearOfNobel(keyName) + ".");
}

public static int getYearOfNobel(String keyName) {
    GetItemOutcome outcome = table.getItemOutcome(new GetItemSpec()
        .withPrimaryKey("CelebrityName", keyName)
        .withConsistentRead(true));

    Item item = outcome.getItem();
    return item.getInt("YearOfNobel");
}
```

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

Results from the Eclipse Console. The results below shows that we created a new table called “CELEBRITIES_SDK”. We added a few rows into the table. The “Query” function queries out “Albert-Einstein”, whose “YearOfNobel” is between 1920 and 1921. The two “Scan” function select out the items whose “YearOfNobel” is less than 1940 and whose “movie” is “Roman Holiday” respectively. The program also updates “YearOfNobel” to 1920 in the end.

```
Markers Properties Servers Data Source Explorer Snippets Problems Console <terminated> MyDynamoDB [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Oct 16, 2015, 1:05:28 PM)
Before table creation:
My DynamoDB tables:
    CELEBRITIES_CONSOLE
Created Table: {AttributeDefinitions: [{AttributeName: CelebrityName,AttributeType: S}],TableName: CELEBRITIES_SDK,KeySchema: [{AttributeName: CelebrityName,KeyType: HASH}]
Waiting for CELEBRITIES_SDK to become ACTIVE...
Table Description: {AttributeDefinitions: [{AttributeName: CelebrityName,AttributeType: S}],TableName: CELEBRITIES_SDK,KeySchema: [{AttributeName: CelebrityName,Ke

After table creation:
My DynamoDB tables:
    CELEBRITIES_CONSOLE
    CELEBRITIES_SDK

Adding stars: Angelina-Jolie/Angelina Jolie/Pitt/Lara Croft: Tomb Raider/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_Angelina_Jolie.jpg/ht
Adding stars: Audrey-Hepburn/Audrey Hepburn/Roman Holiday/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_Audrey_Hepburn.jpg/https://s3.amazo
Adding stars: Jennifer-Aniston/Jennifer Joanna Aniston/The Good Girl/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_Jennifer_Aniston.jpg/htt
Adding nobel laureates: Albert-Einstein/Albert Einstein/Any Movie/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/images/Img_Albert_Einstein.jpg/https://s3.amaz
Adding nobel laureates: Hermann-Muller/Hermann Joseph Muller/Any Movie/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/images/Img_Hermann_Joseph_Muller.j
Adding nobel laureates: Werner-Heisenberg/Werner Karl Heisenberg/Any Movie/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/images/Img_Werner_Karl_Heise

Query items for Nobel laureates with a primary key Albert-Einstein from table CELEBRITIES_SDK:
{ Item: {Movie=Any Movie, CelebrityName=Albert-Einstein, FieldOfScience=Physics, ResumeURL=https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/resumes/Resu

Scan items for Nobel laureates with a YearOfNobel attribute smaller than 1940 from table CELEBRITIES_SDK:
Result: {Items: [{Movie={S: Any Movie,}, CelebrityName={S: Werner-Heisenberg,}, FieldOfScience={S: Physics,}, ResumeURL={S: https://s3.amazonaws.com/hqiu-hu-cloud-}

Scan items for Stars with a movie name "Roman Holiday" from table CELEBRITIES_CONSOLE:
Result: {Items: [{Movie={S: Roman Holiday,}, CelebrityName={S: Audrey-Hepburn,}, ResumeURL={S: https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/resumes/R

Update the celebrity's attribute "YearOfNobel" to 1920:
Before update, the "YearOfNobel" is 1921.
The newly updated "YearOfNobel" is 1920.
```

The results below are almost the same with the previous one. It just skipped to create the table since the table already exists. It also shows the table is deleted in the end. The code to delete the table will be shown in Problem 5.

```
Markers Properties Servers Data Source Explorer Snippets Problems Console 
[terminated= MyDynamoDB [Java Application] /Library/Java/JavaVirtualMachines/dk1.8.0_45.jdk/Contents/Home/bin/java (Oct 16, 2015, 1:02:50 PM)
Before table creation:
My DynamoDB tables:
    CELEBRITIES_CONSOLE
    CELEBRITIES_SDK

Table CELEBRITIES_SDK is already ACTIVE
Table Description: {AttributeDefinitions: [{AttributeName: CelebrityName,AttributeType: S}],TableName: CELEBRITIES_SDK,KeySchema: [{AttributeName: CelebrityName,KeyType: HASH}]

After table creation:
My DynamoDB tables:
    CELEBRITIES_CONSOLE
    CELEBRITIES_SDK

Adding stars: Angelina-Jolie/Angelina Jolie Pitt/Lara Croft: Tomb Raider/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_Angelina_Jolie.jpg/ht
Adding stars: Audrey-Hepburn/Audrey Hepburn/Roman Holiday/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_Audrey_Hepburn.jpg/https://s3.amazo
Adding stars: Jennifer-Aniston/Jennifer Joanna Aniston/The Good Girl/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/images/Img_Jennifer_Aniston.jpg/ht
Adding nobel laureates: Albert-Einstein/Albert Einstein/Any Movie/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/images/Img_Albert_Einstein.jpg/https:
Adding nobel laureates: Hermann-Muller/Hermann Joseph Muller/Any Movie/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/images/Img_Hermann_Joseph_Muller
Adding nobel laureates: Werner-Heisenberg/Werner Karl Heisenberg/Any Movie/https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/images/Img_Werner_Karl_Heise

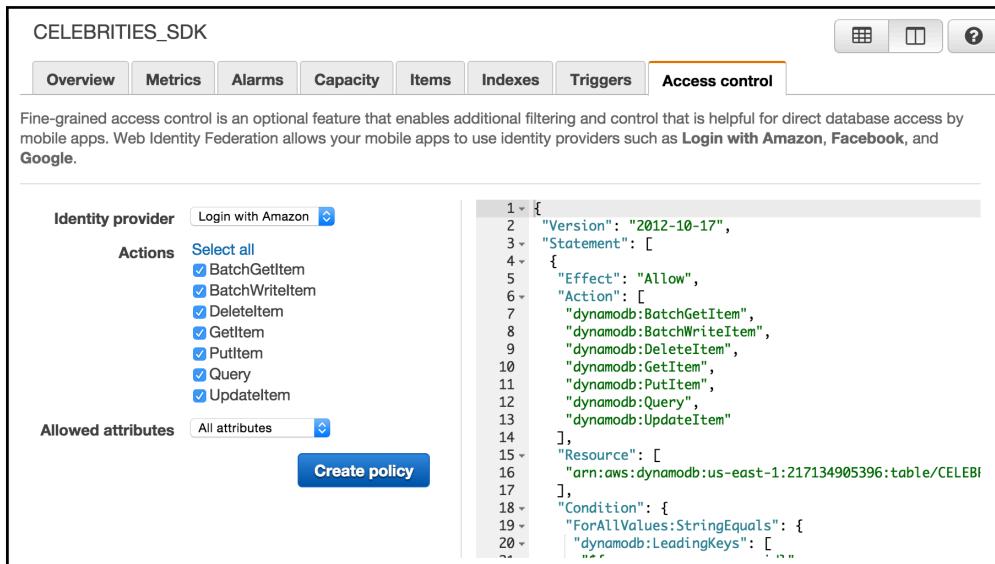
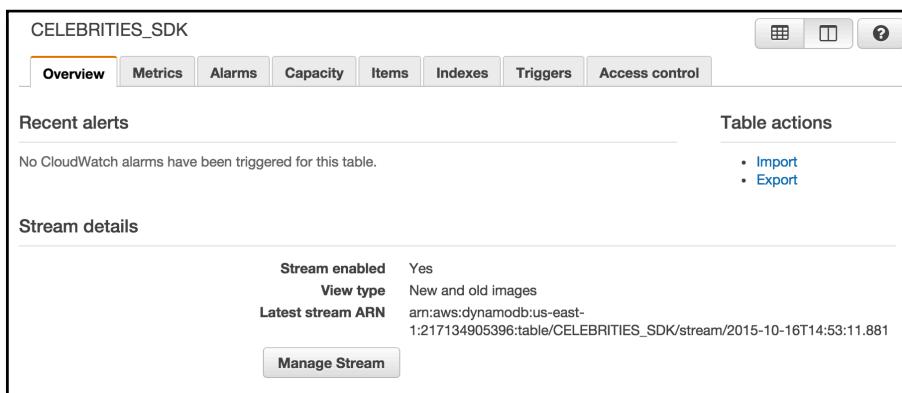
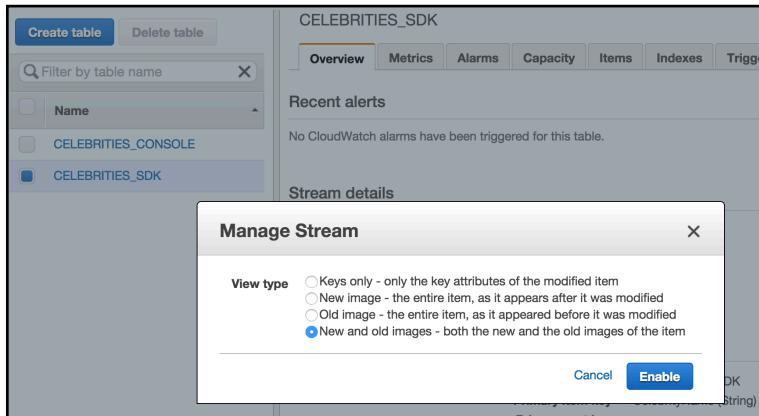
Query items for Nobel laureates with a primary key Albert-Einstein from table CELEBRITIES_SDK:
{ Item: {Movie=Any Movie, CelebrityName=Albert-Einstein, FieldOfScience=Physics, ResumeURL=https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/resumes/Resu
Scan items for Nobel laureates with a YearOfNobel attribute smaller than 1940 from table CELEBRITIES_SDK:
Result: {Items: [{Movie={S: Any Movie,}, CelebrityName={S: Werner-Heisenberg,}, FieldOfScience={S: Physics,}, ResumeURL={S: https://s3.amazonaws.com/hqiu-hu-cloud-}

Scan items for Stars with a movie named "Roman Holiday" from table CELEBRITIES_CONSOLE:
Result: {Items: [{Movie={S: Roman Holiday,}, CelebrityName={S: Audrey-Hepburn,}, ResumeURL={S: https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/stars/resumes/R
Update the celebrity's attribute "YearOfNobel" to 1920:
Before update, the "YearOfNobel" is 1921.
The newly updated "YearOfNobel" is 1920.
Table is deleted: CELEBRITIES_SDK
```

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

- a) Enable the streams, choose the “new and old images”. Set the “Access Control”.



- b) Go to AWS Lambda service and create a Lambda function. On the “Select blueprint” screen, choose the second one on the first row “dynamodb-process-stream”.

Select blueprint		
s3-get-object-python	dynamodb-process-stream	microservice-http-endpoint
An Amazon S3 trigger that retrieves metadata for the object that has been updated. python2.7 - s3	An Amazon DynamoDB trigger that logs the updates made to a table. nodejs - dynamodb	A simple backend (read/write to DynamoDB) with a RESTful API endpoint using Amazon API Gateway. nodejs - api-gateway
node-exec	simple-mobile-backend	kinesis-process-record-python
Demonstrates running an external process using the Node.js child_process module. nodejs	A simple mobile backend (read/write to DynamoDB). nodejs - mobile	An Amazon Kinesis stream processor that logs the data being published. python2.7 - kinesis

- c) On the “Configure event sources” screen, select “CELEBRITIES_SDK” table.

Step 1: Select blueprint		Configure event sources	
Step 2: Configure event sources	Step 3: Configure function	Event source type	DynamoDB
	Step 4: Review	DynamoDB table	CELEBRITIES_SDK
		Batch size	100
		Starting position	Trim horizon

In order to read from the DynamoDB stream, your execution role must have proper permissions.

- d) Configure the function, provide a function name and leave the function unmodified.

Step 1: Select blueprint		Configure function	
Step 2: Configure event sources	Step 3: Configure function	Name*	MyLambdaFunction
	Step 4: Review	Description	An Amazon DynamoDB trigger that logs the updates made to a table.
		Runtime*	Node.js
Lambda function code			
Provide the code for your function. Use the editor if your code does not require custom libraries (other than the aws-sdk). If you need custom libraries, you can upload your code and libraries as a .ZIP file. Learn more about deploying Lambda functions.			
Code entry type <input checked="" type="radio"/> Edit code inline <input type="radio"/> Upload a .ZIP file <input type="radio"/> Upload a .ZIP from Amazon S3			
<pre> 1 console.log('Loading function'); 2 3 exports.handler = function(event, context) { 4 //console.log('Received event:', JSON.stringify(event, null, 2)); 5 event.Records.forEach(function(record) { 6 console.log(record.eventID); 7 console.log(record.eventName); 8 console.log('DynamoDB Record: %j', record.dynamodb); 9 }); 10 context.succeed("Successfully processed " + event.Records.length + " records."); 11 }; </pre>			

- e) Adjust handler and role. On the bottom of “Configure function” screen leave “Handler” as is, as “Role” select suggested value “DynamoDB event stream role”. New screen opens asking for IAM role.

Lambda function handler and role

Handler* index.handler

Role* ✓ Create new role
Basic execution role
S3 execution role
Kinesis execution role
* DynamoDB event stream role
Basic with DynamoDB
Use existing role lambda_dynamo_streams

Suggested role: DynamoDB event stream role

Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. Learn more about how Lambda pricing works.

Memory (MB)* 128

Timeout* 0 min 3 sec

Accept, allow suggested IAM role.

AWS Lambda requires access to your resources

AWS Lambda uses an IAM role that grants your custom code permissions to access AWS resources it needs.

▼ Hide Details

Role Summary ?

Role Provides Lambda and DynamoDB access to AWS

Description Services and Resources.

IAM Role lambda_dynamo_streams

Don't Allow Allow

Leave advanced settings. Select “Next”.

Lambda function handler and role

Handler* index.handler

Role* lambda_dynamo_streams

Ensure that popups are enabled to create a new role. Learn more about Lambda execution roles.

Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. Learn more about how Lambda pricing works.

Memory (MB)* 128

Timeout* 0 min 3 sec

* These fields are required.

Cancel Previous Next

- f) Review and select “Enable now”. Hit “Create Function”. The “Lambda Function” is ready to use.

Lambda > New function using blueprint dynamodb-process-stream

Step 1: Select blueprint
Step 2: Configure event sources
Step 3: Configure function
Step 4: Review

Review

Please review your Lambda function details. You can go back to edit changes for each section. When you are ready, click **Create function** to complete the setup process.

Event sources

DynamoDB Batch size: 100, Starting position: TRIM_HORIZON, DynamoDB table: CELEBRITIES_SDK

Enable event source Enable now Enable later i

Lambda function

Name MyLambdaFunction

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

Create function

Lambda > Functions > MyLambdaFunction ARN - arn:aws:lambda:us-east-1:217134905396:function:MyLambdaFunction

Test Actions ▾

Congratulations! Your Lambda function "MyLambdaFunction" has been successfully created and configured with DDB: CELEBRITIES_SDK as an event source.

Event sources

Event source	ARN	State	Details
DDB: CELEBRITIES_SDK	arn:aws:dynamodb:us-east-1:217134905396:table/CELEBRITIES_SDK/stream/2015-10-16T14:53:11.881	Enabled	Batch size: 100, Last result: No records processed

Add event source

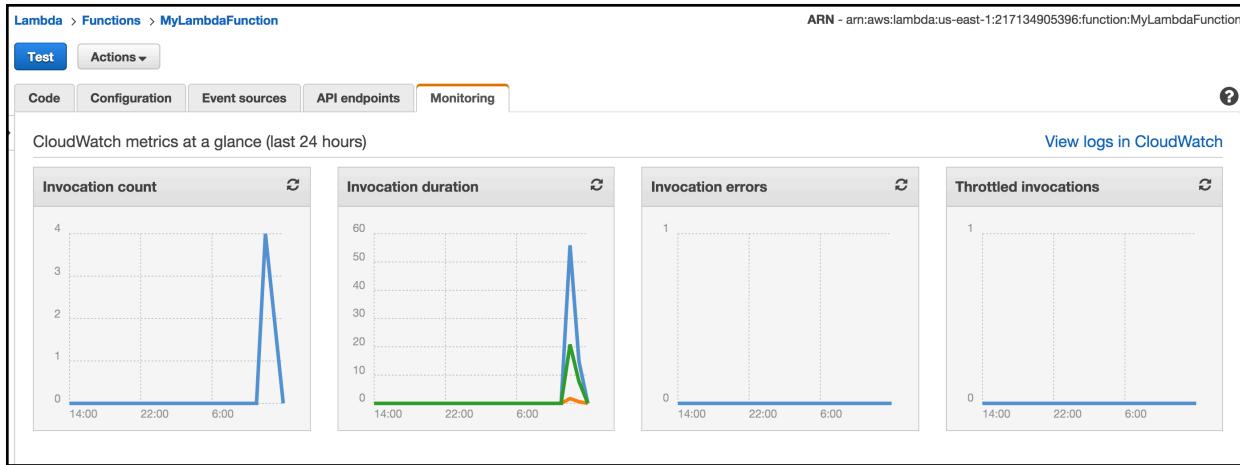
In DynamoDB Dashboard, select the table “CELEBRITIES_SDK”. We can also see that the lambda function was in the “Triggers” Tab of the DynamoDB table “CELEBRITIES_SDK”.

CELEBRITIES_SDK

Triggers

Function name	State	Last result
MyLambdaFunction	Enabled	No records processed

- g) In AWS Lambda Dashboard, select the lambda function “MyLambdaFunction” we just created. Double click on it and select “Monitoring” Tab. CloudWatch graphs will show up.



- h) On the top right of the page, select “View logs in CloudWatch”. We will see the logs below. Re-run the Java class developed in Problem 3 to generate load on table “CELEBRITIES_SDK”. Double click the log, we can see the comparison of “newImage” and “oldImage” here. The “YearOfNobel” is updated from 1921 to 1920.



```

"NewImage": {
    "Movie": {
        "S": "Any Movie"
    },
    "CelebrityName": {
        "S": "Albert-Einstein"
    },
    "FieldOfScience": {
        "S": "Physics"
    },
    "ResumeURL": {
        "S": "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/resumes/Resume_Albert_Einstein.docx"
    },
    "PictureURL": {
        "S": "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/images/Img_Albert_Einstein.jpg"
    },
    "YearOfNobel": {
        "N": "1920"
    },
    "Name": {
        "S": "Albert Einstein"
    }
},
"OldImage": {
    "Movie": {
        "S": "Any Movie"
    },
    "CelebrityName": {
        "S": "Albert-Einstein"
    },
    "FieldOfScience": {
        "S": "Physics"
    },
    "ResumeURL": {
        "S": "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/resumes/Resume_Albert_Einstein.docx"
    },
    "PictureURL": {
        "S": "https://s3.amazonaws.com/hqiu-hu-cloud-computing-hw6/nobels/images/Img_Albert_Einstein.jpg"
    },
    "YearOfNobel": {
        "N": "1921"
    }
},

```

Since we didn't recreate the table, there's no log for creating tables here.

Problem 5.

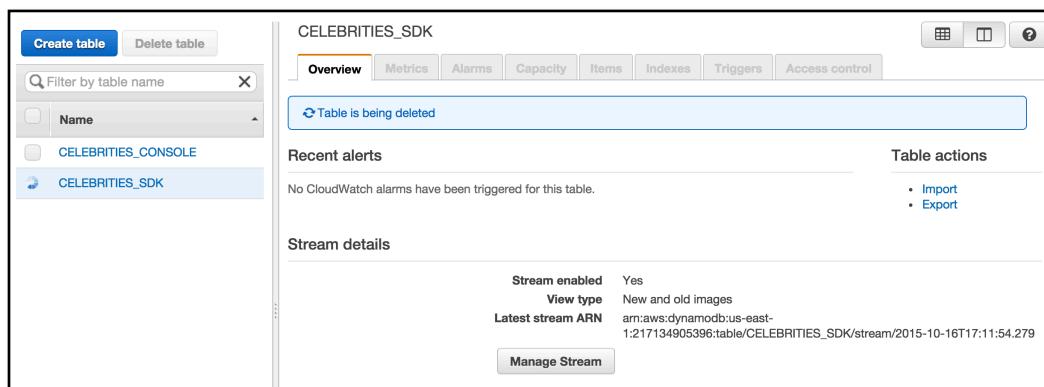
When done, delete all DynamoDB tables and Lambda functions, programmatically or using AWS CLI. Provide working code and capture all stages of testing. [10 Points]

- a) Delete the DynamoDB tables through Java SDK. It basically gets the "table" object and calls the "delete()" function.

```
private static void deleteTable(String tableName) {
    Table table = new DynamoDB(dynamoDB).getTable(tableName);

    if (table == null)
        return;
    table.delete();

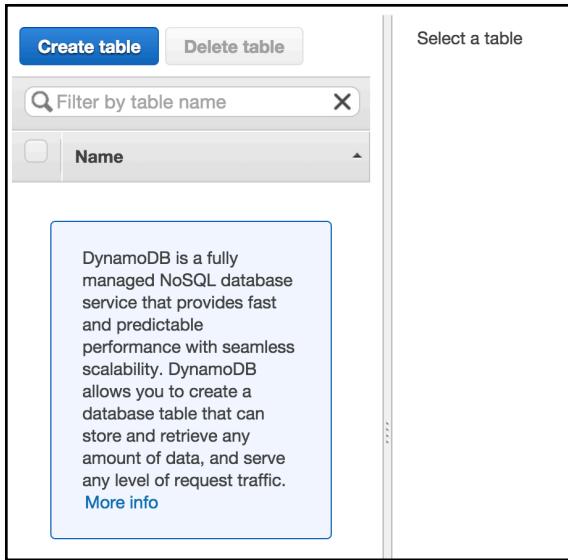
    try {
        table.waitForDelete();
    } catch (InterruptedException e) {
        System.out.println("Failed to delete table: " + tableName);
        e.printStackTrace();
    }
    System.out.println("Table is deleted: " + tableName);
}
```



- b) Delete the Dynamo DB tables through AWS CLI.

```
hqiu@bos-mpdei>> aws dynamodb delete-table --table-name CELEBRITIES_CONSOLE
{
    "TableDescription": {
        "TableArn": "arn:aws:dynamodb:us-east-1:217134905396:table/CELEBRITIES_CONSOLE",
        "ProvisionedThroughput": {
            "NumberOfDecreasesToday": 0,
            "WriteCapacityUnits": 1,
            "ReadCapacityUnits": 1
        },
        "TableSizeBytes": 0,
        "TableName": "CELEBRITIES_CONSOLE",
        "TableStatus": "DELETING",
        "StreamSpecification": {
            "StreamViewType": "NEW_AND_OLD_IMAGES",
            "StreamEnabled": true
        },
        "LatestStreamLabel": "2015-10-15T20:32:28.777",
        "ItemCount": 0,
        "LatestStreamArn": "arn:aws:dynamodb:us-east-1:217134905396:table/CELEBRITIES_CONSOLE/stream/2015-10-15T20:32:28.777"
    }
}
```

After delete:



c) Delete the lambda function through AWS CLI.

A screenshot of the AWS Lambda console under the 'Functions' section. It shows one Lambda function named 'MyLambdaFunction'. The details are as follows:

Function name	Description	Code size	Memory (MB)	Timeout (s)
MyLambdaFunction	An Amazon DynamoDB trigger that logs the updates made to a table.	365 bytes	128	3

```
hqiu@bos-mpdei>> aws lambda delete-function --function-name MyLambdaFunction --region us-east-1  
hqiu@bos-mpdei>>
```

After delete:

A screenshot of the AWS Lambda landing page. It features the AWS Lambda logo (a stylized orange block icon) and the text 'AWS Lambda'. Below that, a paragraph explains what Lambda is: 'AWS Lambda is a compute service that runs developers' code in response to events and automatically manages the compute resources for them, making it easy to build applications that respond quickly to new information.' At the bottom are 'Get Started Now' and 'Learn more about AWS Lambda' buttons.