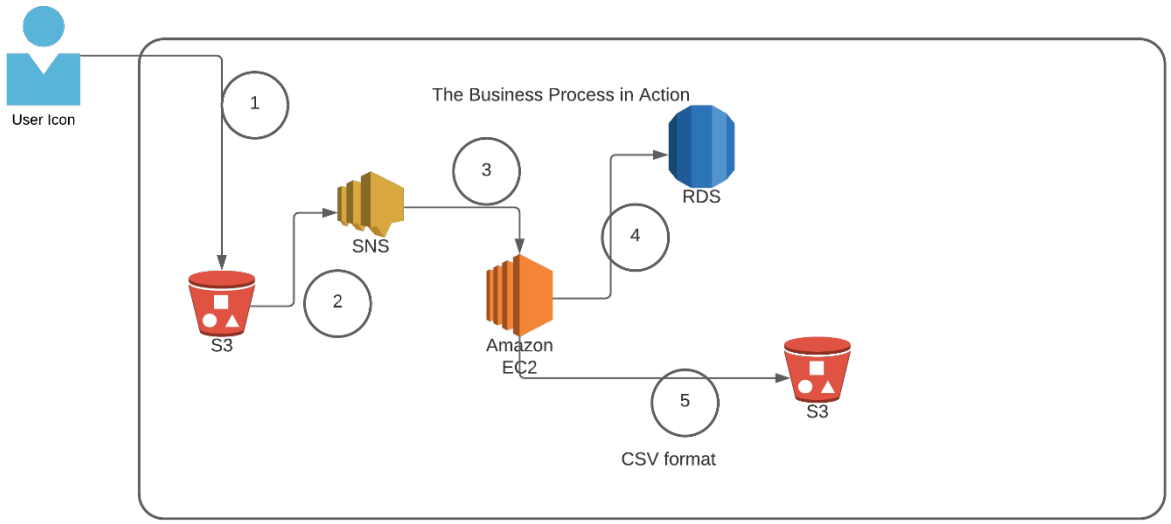
Declaration	
Questions in this exercise are intentionally complex and could be convoluted or confusing. This is by design and to simulate real life situations where customers seldom give crystal clear requirements and ask unambiguous questions.	

I have read the above statement and agree to these conditions	
I AGREE	
	Srikanta Ghosh

Instructions
Every screenshot requested in this workbook is compulsory and carries 1 marks
Your AWS account ID must be clearly visible in every screenshot using the AWS console; missing id or using someone else's id is not permitted. Such cases will be considered as plagiarism and severe penalty will be imposed.
All screenshots must be in the order mentioned under "Expected Screenshots" for every step
DO NOT WAIT UNTIL THE LAST MINUTE. The program office will not extend the project submission deadline under any circumstances.
The file should be renamed in the format BATCH_FIRSTNAME_LASTNAME_PROJECT1. For example: PGPCCMAY18_VIJAY_DWIVEDI_PROJECT1.pdf

Resource Clean Up
Cloud is always pay per use model and all resources/services that we consume are chargeable. Cleaning up when you’ve completed your lab or project is always necessary. This is true whether you’re doing a lab or implementing a project at your workplace.
After completing the lab, make sure to delete each resource created in reverse chronological order. Each AWS Academy session lasts for 4 hours by default, although you can extend a session to run longer by pressing the start button to reset your session timer. At the end of each session, any resources you created in the account will be preserved. Some AWS resources, such as EC2 instances, may be automatically shut down, while other resources, such as RDS instances will be left running.

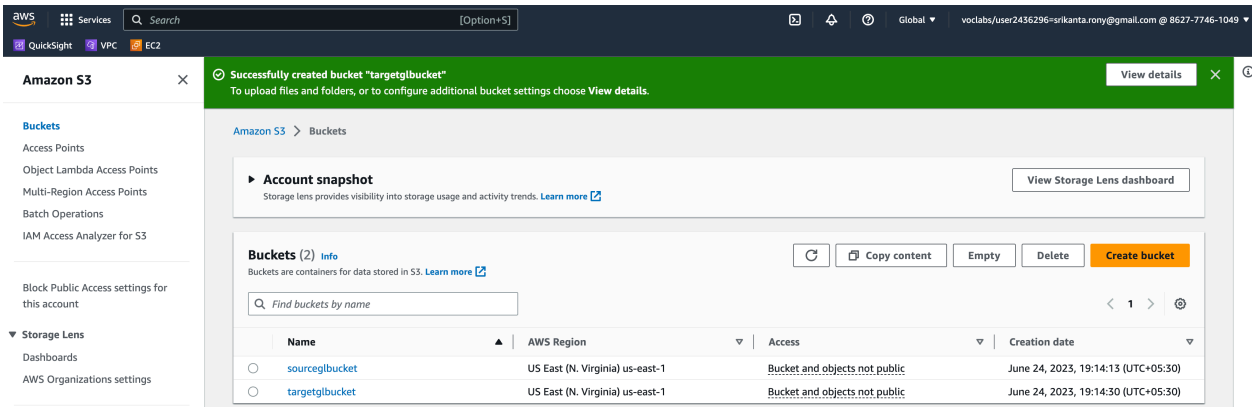
Architecture diagram



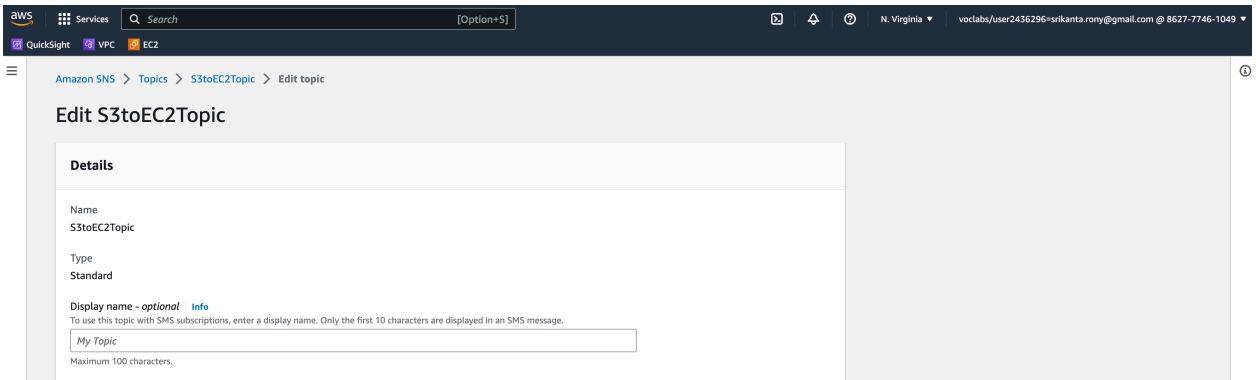
Architecture Implementation	
1	The customer uploads the invoice data to S3 bucket in a text format as per their guidelines and policies. This bucket will have a policy to auto delete any content that is more than 1 day old (24 hours).
2	An event will trigger in the bucket that will place a message in SNS topic
3	A custom program running in EC2 will subscribe to the SNS topic and get the message placed by S3 event
4	The program will use S3 API to read from the bucket, parse the content of the file and create a CSV record and save the details in an RDS database
5	The program will use S3 API to write CSV record to destination S3 bucket as new S3 object.
Note	The custom program codebase and sample invoice have been shared along with this workbook on the LMS.

Step 1: SNS and S3 topic creation

Step number	a
Step name	Creation of Source and target buckets
Instructions	1) Navigate to S3 using the Services button at the top of the screen 2) Select "Create Bucket" 3) Enter a source bucket name and use the default options for the rest of the fields 4) Click on "Create Bucket' 5) Repeat the above steps to create a target bucket
Expected screenshots	1) Screen showing created S3 source and target buckets



Step number	b
Step name	Creation of SNS subscription
Instructions	1) Navigate to SNS -> Topics 2) Click on "Create Topic" 3) Enter the following fields Name : S3toEC2Topic The other options can be ignored for now 4) Click on Create Topic
Expected screenshots	1) Creation of SNS topic



Step number	c
Step name	Modification of SNS Access Policy

Instructions	1) Navigate to SNS -> Topics and select the topic created in the previous step
	2) Note down the ARN shown in the topic details
	2) Click on Edit and select "Access Policy".
	3) Replace the text in the JSON editor with the following
	{
	"Version": "2012-10-17",
	"Id": "example-ID",
	"Statement": [
	{
	"Sid": "example-statement-ID",
	"Effect": "Allow",
	"Principal": {
	"AWS":"*"
	},
	"Action": [
	"SNS:Publish"
	],
	"Resource": "**SNS-topic-ARN**",
	"Condition": {
	"ArnLike": { "aws:SourceArn": "arn:aws:s3:*:*:**bucket-name**" },
	"StringEquals": { "aws:SourceAccount": "**bucket-owner-account-id**" }
	}
	}
	]
	}
	
	4) Replace the bold text with the SNS topic ARN, source bucket name and your AWS account ID respectively.
	5) Click on Save Changes

Expected screenshots	1) JSON Editor screen

Access policy - *optional* Info

This policy defines who can access your topic. By default, only the topic owner can publish or subscribe to the topic.

JSON editor

```
2   "Version": "2012-10-17",
3   "Id": "example-ID",
4   "Statement": [
5     {
6       "Sid": "example-statement-ID",
7       "Effect": "Allow",
8       "Principal": {
9         "AWS": "*"
10      },
11      "Action": "SNS:Publish",
12      "Resource": "arn:aws:sns:us-east-1:862777461049:S3toEC2Topic",
13      "Condition": {
14        "StringEquals": {
15          "aws:SourceAccount": "862777461049"
16        },
```

Step number	d
Step name	Configuring SNS notifications for S3
Instructions	1) Navigate to S3 and select the source bucket created in Step 1 (a) 2) Select Properties and scroll down to Event Notifications and select it 3) Select "Create Event Notification" 4) Fillup the details as follows Name : S3PutEvent Select PUT from the list of radio buttons Destination : Select SNS Topic SNS : Select S3ToEC2Topic 5) Save Changes
Expected screenshots	1) Event Configuration Screen

Event notifications (1)

Send a notification when specific events occur in your bucket. Learn more

Edit | Delete | Create event notification

Name	Event types	Filters	Destination type	Destination
S3PutEvent	Put	-	SNS topic	S3toEC2Topic

Amazon EventBridge

For additional capabilities, use Amazon EventBridge to build event-driven applications at scale using S3 event notifications. Learn more or see EventBridge pricing

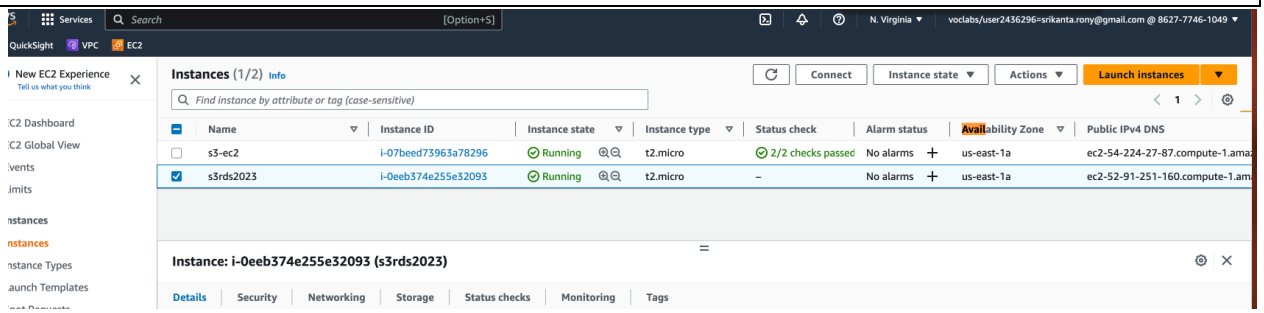
Edit

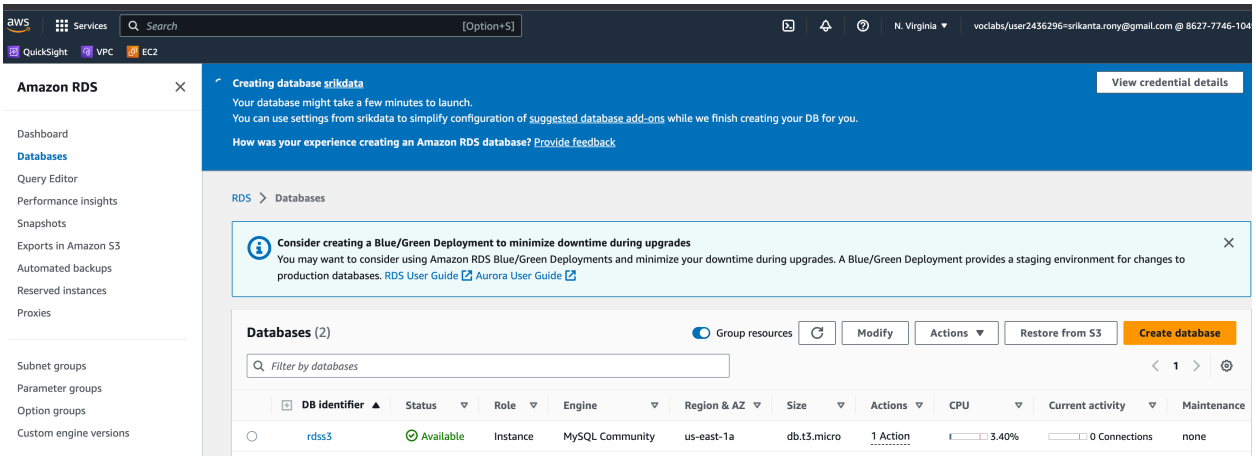
Send notifications to Amazon EventBridge for all events in this bucket

Off

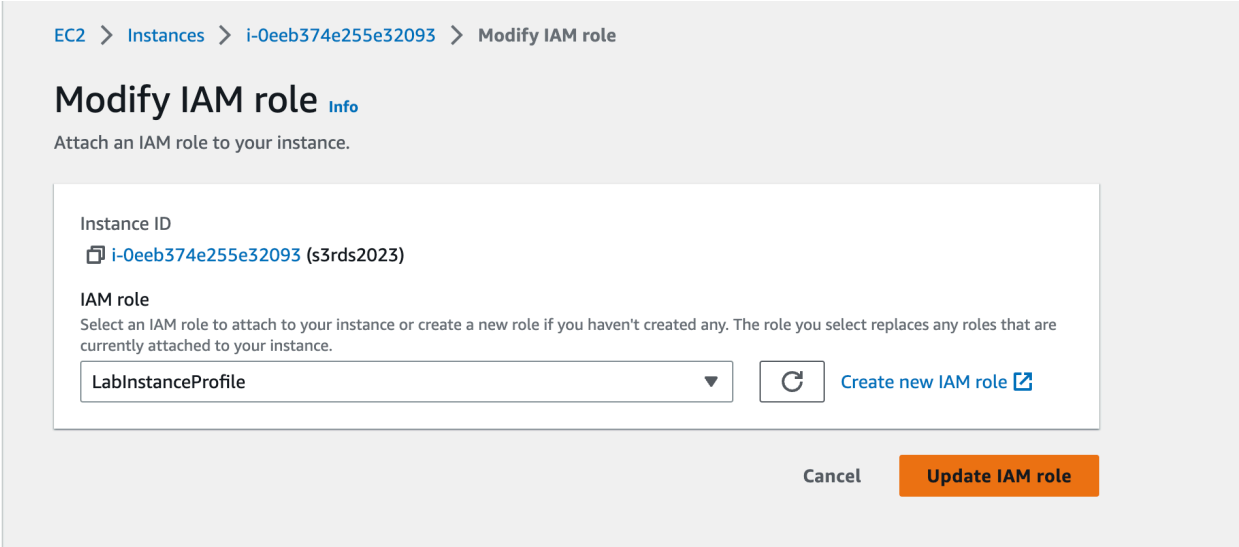
Step 2: Run the custom program in the EC2 instance

Step number	a
Step name	Creation of the EC2 instance and RDS instance
Instructions	1) Navigate to EC2 -> Instances 2) Create an EC2 instance with the following parameters AMI : Amazon Linux 2 VPC : Default Security group : Ports 22 and 8080 should be opened 3) Navigate to RDS 4) Create an RDS instance with the following parameters: Engine type : MySql Template : Dev/Test Set the username and password as required DB Instance class : Burstable Instance type : t3.micro Storage type : General purpose SSD (gp2) Public Access : Yes VPC Security group : Create New () Under Additional Configuration, add an initial database name. Take note of this name as it will be required later. Uncheck “Enable Enhanced Monitoring” Ensure that the security group created by the RDS deployment has port 3306 open for all incoming connections from all sources.
Expected screenshots	1) List of instances after creation of EC2 instance 2) List of RDS instances





Step number	b
Step name	Assignment of IAM role for EC2 instance
Instructions	1) Navigate back to EC2- > Instances 2) Select the EC2 instance created in the previous step and select Actions-> Security -> Modify IAM role 3) Select the role LabInstanceProfile from the dropdown and click on Save
Expected screenshots	1) Modify IAM role screen



Step number	c
Step name	Configuration and Uploading of custom program
Instructions	1) Download the file **docproc-new.zip** on your machine 2) Unzip the downloaded file 3) Enter the unzipped folder and open the file views.py in the API folder using a text editor 4) In line number 19-24, modify the target bucket name to the one created in Step 2 (a) and modify the hostname, username, password and database variables to the values set while creating the RDS database and save the file 5) Copy the folder docproc-new to the home folder of the EC2 instance created in Step 3(a) using scp. Use the command given below *scp -i* pem *-r ./docproc-new ec2-user@*ip *:/home/ec2-user*

Expected screenshots	1) Modifying of the views.py file to point to the target bucket	2)Copying the folder to the EC2 instance

```
from django.http import HttpResponse
from django.views.decorators.csrf import csrf_exempt
from botocore.exceptions import ClientError
from boto3.dynamodb.conditions import Key, Attr
import json
import boto3
import datetime
import mysql.connector


hostname = 'srikdata.c1hsic3suj1k.us-east-1.rds.amazonaws.com'
username = 'srikanta'
password = 'Annie1712'
database = 'ms_rds'

s3_target_bucket = 'targetglbucket'


# ***********************************************************************************
# Below methods are the handlers for the web http endpoint
```

```
(base) srikantaghosh@MAC-P9400769DT Downloads % scp -i s3rds.pem -r ./docproc-new ec2-user@52.91.251.160:/home/ec2-user
.DS_Store                                                                                                   100% 6148    24.1KB/s   00:00
__init__.py                                                                                                 100%    0     0.0KB/s   00:00
settings.py                                                                                                 100% 3129    12.3KB/s   00:00
urls.py                                                                                                     100%  798     3.1KB/s   00:00
wsgi.py                                                                                                     100%  392     1.5KB/s   00:00
db.sqlite3                                                                                                  100%   37KB  29.5KB/s   00:01
__init__.py                                                                                                 100%    0     0.0KB/s   00:00
models.py                                                                                                   100%  122     0.5KB/s   00:00
__init__.py                                                                                                 100%    0     0.0KB/s   00:00
apps.py                                                                                                     100%  146     0.6KB/s   00:00
admin.py                                                                                                    100%  128     0.5KB/s   00:00
tests.py                                                                                                    100%  125     0.5KB/s   00:00
views.py                                                                                                    100% 7121    27.7KB/s   00:00
manage.py                                                                                                   100%  805     3.1KB/s   00:00
(base) srikantaghosh@MAC-P9400769DT Downloads %
```

Step 3: Creation and Verification of SNS subscription and Generation of CSV file

Step number	a
Step name	Starting the EC2 custom program

Instructions	1) Log into the EC2 instance using SSH 2) Run the followng commands after successful SSH to start the server sudo cp -r docproc-new /opt sudo chown ec2-user:ec2-user -R /opt cd /opt/docproc-new sudo yum update sudo yum install python-pip -y python -m pip install --upgrade pip setuptools sudo pip install virtualenv virtualenv ~/.virtualenvs/djangodev source ~/.virtualenvs/djangodev/bin/activate pip install django pip install boto3 pip install mysql-connector-python-rf python manage.py runserver 0:8080 **Keep this terminal window open throughout the rest of the exercise**
Expected screenshots	1) Server in waiting state

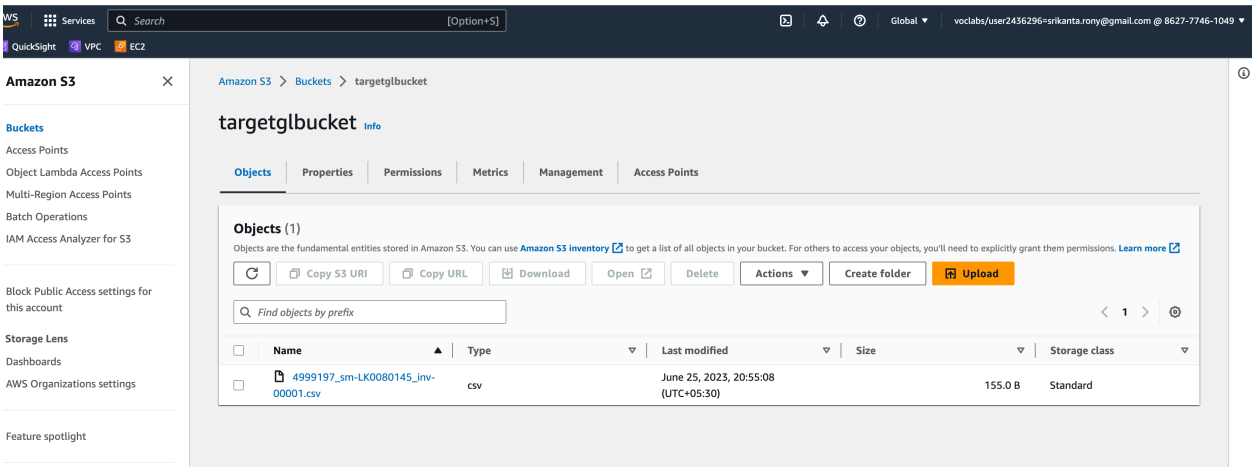
```
Collecting pytz
  Downloading pytz-2023.3-py2.py3-none-any.whl (502 kB)
     |################################| 502 kB 11.3 MB/s
Installing collected packages: pytz, django
Successfully installed django-1.11.29 pytz-2023.3
(djangodev) [ec2-user@ip-172-31-81-211 docproc-new]$ pip install boto3
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip c
 be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functionality.
Collecting boto3
  Downloading boto3-1.17.112-py2.py3-none-any.whl (131 kB)
     |################################| 131 kB 27.5 MB/s
Collecting jmespath<1.0.0,>=0.7.1
  Downloading jmespath-0.10.0-py2.py3-none-any.whl (24 kB)
Collecting s3transfer<0.5.0,>=0.4.0
  Downloading s3transfer-0.4.2-py2.py3-none-any.whl (79 kB)
     |################################| 79 kB 14.8 MB/s
Collecting botocore<1.21.0,>=1.20.112
  Downloading botocore-1.20.112-py2.py3-none-any.whl (7.7 MB)
     |################################| 7.7 MB 44.5 MB/s
Collecting futures<4.0.0,>=2.2.0; python_version == "2.7"
  Downloading futures-3.4.0-py2-none-any.whl (16 kB)
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     |################################| 247 kB 59.1 MB/s
Collecting urllib3<1.27,>=1.25.4
  Downloading urllib3-1.26.16-py2.py3-none-any.whl (143 kB)
     |################################| 143 kB 60.8 MB/s
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: jmespath, futures, six, python-dateutil, urllib3, botocore, s3transfer, boto3
Successfully installed boto3-1.17.112 botocore-1.20.112 futures-3.4.0 jmespath-0.10.0 python-dateutil-2.8.2 s3transfer-0.4.2 six-1.16.0 urllib3-1.26.16
(djangodev) [ec2-user@ip-172-31-81-211 docproc-new]$ pip install mysql-connector-python-rf
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in pip c
 be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support pip 21.0 will remove support for this functionality.
Collecting mysql-connector-python-rf
  Downloading mysql-connector-python-rf-2.2.2.tar.gz (11.9 MB)
     |################################| 11.9 MB 25.2 MB/s
Building wheels for collected packages: mysql-connector-python-rf
  Building wheel for mysql-connector-python-rf (setup.py) ... done
  Created wheel for mysql-connector-python-rf: filename=mysql_connector_python_rf-2.2.2-cp27-cp27mu-linux_x86_64.whl size=249459 sha256=f7bf39049b38057a01b16b0bbd2553052a7629a6d2c2c55851b91be0efbbd963
  Stored in directory: /home/ec2-user/.cache/pip/wheels/3b/b5/d4/5d0e3338625186eb2fbf75908b5b178b859aa8e1fd1291a0fa
Successfully built mysql-connector-python-rf
Installing collected packages: mysql-connector-python-rf
Successfully installed mysql-connector-python-rf-2.2.2
(djangodev) [ec2-user@ip-172-31-81-211 docproc-new]$ python manage.py runserver 0:8080
Performing system checks...

System check identified no issues (0 silenced).
June 25, 2023 - 15:06:21
Django version 1.11.29, using settings 'docproc.settings'
Starting development server at http://0:8080/
Quit the server with CONTROL-C.
```

Step number	b
Step name	Creation of SNS subscription

Instructions	1) Navigate to SNS in the AWS Console and select the topic S3ToEC2Topic 2) Click on Create Subscription 3) Enter the following details Protocol : HTTP Endpoint : http://52.91.251.160:8080/sns where <host> in the public IP of the EC2 instance Click on Create Subscription 4) In the EC2 terminal window, look for the field "SubscribeURL" and copy the entire link given **Note: If a message is seen "ValueError: No JSON object could be decoded", it can be safely ignored** 5) Paste that link into a browser window to verify the SNS subscription (Ignore any messages received in the web browser)
Expected screenshots	1) Subscription URL in EC2 terminal Window

```
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-01d088a6f77103d0fe307c0069e40ed6.pem"
}
Request method = POST
The S3 JSON is  {
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "37fc880c-bbbb-44Be-8584-fc6b0ecefb4c",
  "Token" : "2336412f37fb687f5d51e6e2425c464de3d3f2d44b900e140d377a7bb231f913b3814e5258e94d54e2babd24823b0c66cbf59dc4ef563cc7af2413ce2a805a9a0e9f08b97c57d51ed1391350acbb34cef5b24ac2c3a734005885 3dada13ad95cc629ff1deba7dd41b55ccad570cb7894",
  "TopicArn" : "arn:aws:sns:us-east-1:862777461049:S3toEC2Topic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-east-1:862777461049:S3toEC2Topic.\nTo confirm the subscription, visit the SubscribeURL included in this message.",
  "SubscribeURL" : "https://sns.us-east-1.amazonaws.com/?Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-east-1:862777461049:S3toEC2Topic&Token=2336412f37fb687f5d51e6e2425c464de3d3f2d44b900e140d377a7bb231f913b3814e5258e94d54e2babd24823b0c66cbf59dc4ef563cc7af2413ce2a805a9a0e9f08b97c57d51ed1391350acbb34cef5b24ac2c3a734005885 3dada13ad95cc629ff1deba7dd41b55ccad570cb7894",
  "Timestamp" : "2023-06-25T15:23:20.400Z",
  "SignatureVersion" : "1",
  "Signature" : "X93Xbt35VAIQtDKks/UhG5fwKbuYCMwc1XyyDFnSDqNCQVhPmda4HWQ5tqlm6OhaxsyYIZa0fkésXcTTekplnOxAt2kMyukfy581kFxlYFq2je/OTE1eaFnuN3U/busSeJ4+FvwGs6SjMVK8GUViqWZonSRMjgf5VicB6xWD+jEacrwZRhLfO5baWHtFwzEpluuVXFEgdgBFSFR119rBBJLPDhS655q03WL6fKOrcBSahkPumUMFub/F7AZ5271fSD9LA2YMLN9CtK3RG5nL+bQSoKDAw6+2SPvpoy6VRrRoU4Q8zSYudjv7iz8XkDMLw9nLsrAPLHQDGGz0vasHPg==",
  "SigningCertURL" : "https://sns.us-east-1.amazonaws.com/SimpleNotificationService-01d088a6f77103d0fe307c0069e40ed6.pem"
}
Internal Server Error: /sns
Traceback (most recent call last):
```


Step number	c
Step name	Generation of CSV file
Instructions	1) Download the file **docproc-invoice.txt** provided with this workbook 2) Navigate to S3 in the AWS Console 3) Upload the sample invoice file to the source S3 bucket using the default options 4) Verify that a CSV file is generated in the target S3 bucket. This may take a few minutes 5) (Optional) Login to the RDS instance using your preferred MySQL client and check the table created inside the specified database.
Expected screenshots	1) Generated CSV file in the target S3 bucket



Answer the following questions

Q1 Which of the following properties of an AWS resource is sufficient and necessary to uniquely identify it across all of AWS?

a) ARN

b) Region and ARN

c) ARN and Account number

d) Depends on the resource used

Enter your answer here | a) ARN |

Q2 Which of the following step numbers in Step 1 allowed S3 to publish to the SNS topic created?

a) 1(a)

b) 1(c)

c) 1(d)

d) 1(b)

Enter your answer here | b) 1(c) |

Q3 Which port is being used by SNS to send the notification to the custom program?

a) 8081

b) 80

c) 8080

d) 8065

Enter your answer here | c) 8080

Q4 How many IAM roles can be attached to an EC2 instance at a time?

a) 2

b) 3

c) 1

d) Depends on the policies required

Enter your answer here | c) 1

Q5 As a product manager, how would you describe the benefits of this architecture to an client, as compared to an equivalent on-premises architecture?

1) Flexible message delivery over multiple transport protocols.
2) Inexpensive, pay-as-you-go model with no up-front costs
3) Very reliable as compared to on-premises architecture.
4) Less or minimal coding required
5) It offers built-in security, backup and restores, and in-memory caching
6) Easy integration with applications and Simple APIs.
7) With Amazon Athena, you pay only for the queries that you run.

Grades distribution	
MCQs	10 (2.5 mark each)
Subjective questions	6 marks

Implementation screenshots	24 marks (2 marks each)
Total	40 marks