

```
In [16]: print("SteelEye Data Engineer Assignment")

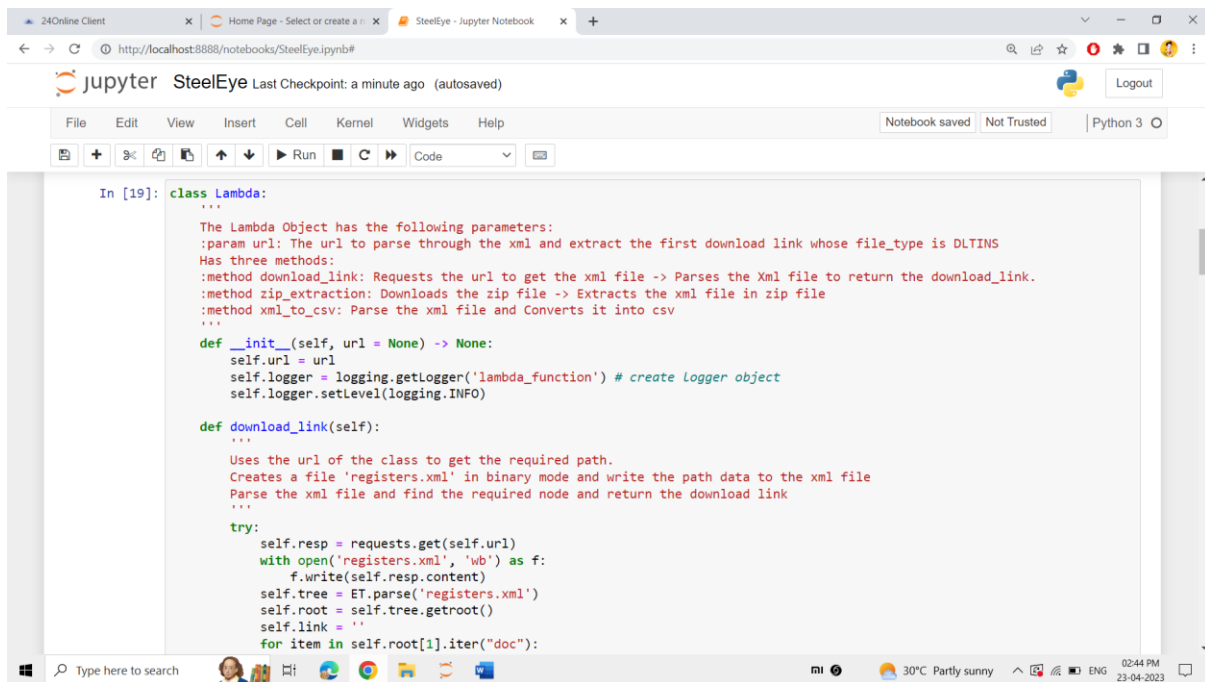
SteelEye Data Engineer Assignment

In [17]: My_Details = {
          "Name"       : "Jaya Vardhan Swarna",
          "Registration Number" : 12002055,
          "University"  : "Lovely Professional University"
        }
        print(My_Details)

{'Name': 'Jaya Vardhan Swarna', 'Registration Number': 12002055, 'University': 'Lovely Professional University'}

In [ ]:

In [18]: #import all the required Libraries
import csv
import requests
import xml.etree.ElementTree as ET
import zipfile
import pandas as pd
import boto3
from io import StringIO
import logging
```



```
In [19]: class Lambda:
...
    The Lambda Object has the following parameters:
    :param url: The url to parse through the xml and extract the first download link whose file_type is DLTINS
    Has three methods:
    :method download_link: Requests the url to get the xml file -> Parses the Xml file to return the download_link.
    :method zip_extraction: Downloads the zip file -> Extracts the xml file in zip file
    :method xml_to_csv: Parse the xml file and Converts it into csv
...
    def __init__(self, url = None) -> None:
        self.url = url
        self.logger = logging.getLogger('lambda_function') # create Logger object
        self.logger.setLevel(logging.INFO)

    def download_link(self):
        ...
        Uses the url of the class to get the required path.
        Creates a file 'registers.xml' in binary mode and write the path data to the xml file
        Parse the xml file and find the required node and return the download link
        ...
        try:
            self.resp = requests.get(self.url)
            with open('registers.xml', 'wb') as f:
                f.write(self.resp.content)
            self.tree = ET.parse('registers.xml')
            self.root = self.tree.getroot()
            self.link = ''
            for item in self.root[1].iter("doc"):
                ...
```

24Online Client x Home Page - Select or create a x SteelEye - Jupyter Notebook x +

← → ↻ http://localhost:8888/notebooks/SteelEye.ipynb#

Jupyter SteelEye Last Checkpoint: a minute ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

```
for item in self.root[1].iter("doc"):
    if item.find("str[@name = 'file_type']").text == 'DLTINS':
        self.link = item.find("str[@name='download_link']").text
        break
if not self.link:
    raise Exception('Could not find download link for file_type DLTINS')
return self.link
except Exception as e:
    self.logger.error(f"Error in download_link: {e}")
    raise e

def zip_extraction(self, link = None):
    """
    :param link: url link to download the zip file
    Uses the link to request the link
    Create a file 'zip_file.zip' and write the content into the file
    Extract the zip file and save the name of the file from the namelist and return it
    """
    try:
        self.zip_file = requests.get(self.link)
        with open('zip_file.zip', 'wb') as f:
            f.write(self.zip_file.content)
        self.xml_file = ''
        with zipfile.ZipFile('zip_file.zip', 'r') as f:
            self.xml_file = f.namelist()[0]
            f.extractall('')
        if not self.xml_file:
            raise Exception('Could not extract xml file from zip')
    
```

Type here to search 30°C Partly sunny 02:44 PM 23-04-2023

24Online Client x Home Page - Select or create a x SteelEye - Jupyter Notebook x +

← → ↻ http://localhost:8888/notebooks/SteelEye.ipynb#

Jupyter SteelEye Last Checkpoint: a minute ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

```
        raise Exception('Could not extract xml file from zip')
    return self.xml_file
except Exception as e:
    self.logger.error(f"Error in zip_extraction: {e}")
    raise e

def xml_to_csv(self, xml = None):
    self.new = ET.parse(xml) #parse xml
    self.test = self.new.getroot()

    self.pattern = 'FinInstrmGnlAttrbts' #required node
    self.children = ['Id', 'FullNm', 'ClsstctnTp', 'CmmdtyDerivInd', 'NtnlCcy'] #required children nodes

    self.tag = 'Issr' #required node

    self.rows = []
    self.cols = [self.pattern + '.' + k for k in self.children]
    self.cols.append(self.tag)

    self.parent = 'TermntdRcrd' #parent node

    for i in self.test.iter():
        if self.parent in i.tag: # If parent is found
            self.entry = [None for x in range(len(self.cols))] # Initialise array of required elements
            for child in i: # If required child has been found
                for c in child: # Get the required grand-children
                    for k in range(len(self.children)):

```

Type here to search 30°C Partly sunny 02:45 PM 23-04-2023

The screenshot shows a Jupyter Notebook titled 'SteelEye' with the following code:

```
if self.pattern in child.tag:
    for c in child:
        for k in range(len(self.children)):
            if self.children[k] in c.tag:
                self.entry[k] = c.text

if self.tag in child.tag:
    self.entry[5] = child.text
self.rows.append(self.entry)

self.df = pd.DataFrame(self.rows, columns=self.cols)
return self.df
```

In []: `if __name__ == '__main__':`

```
url = "https://registers.esma.europa.eu/solr/esma_registers_firds_files/select?q=*%&q=publication_date:%5B2021-01-17T00:00:00Z%5D"
p = Lambda(url) #create an object for class Lambda

#Requirement 2: From the xml, please parse through to the first download Link whose file_type is DLTINS and download the zip
zip_link = p.download_link()

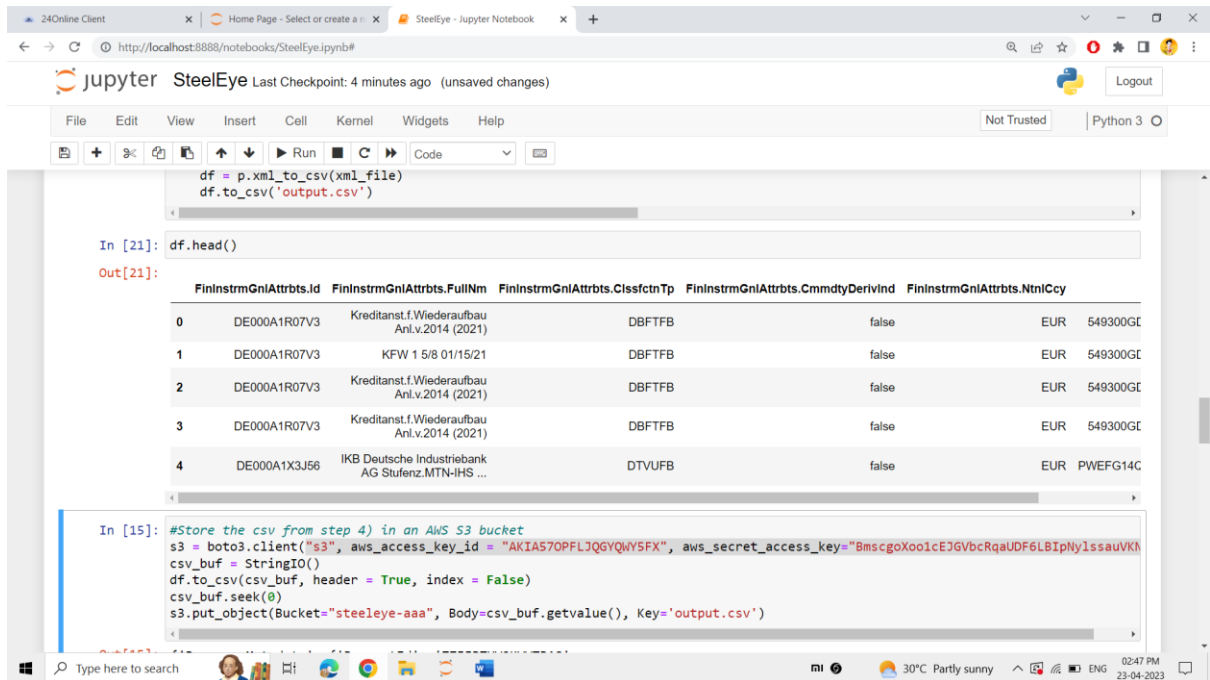
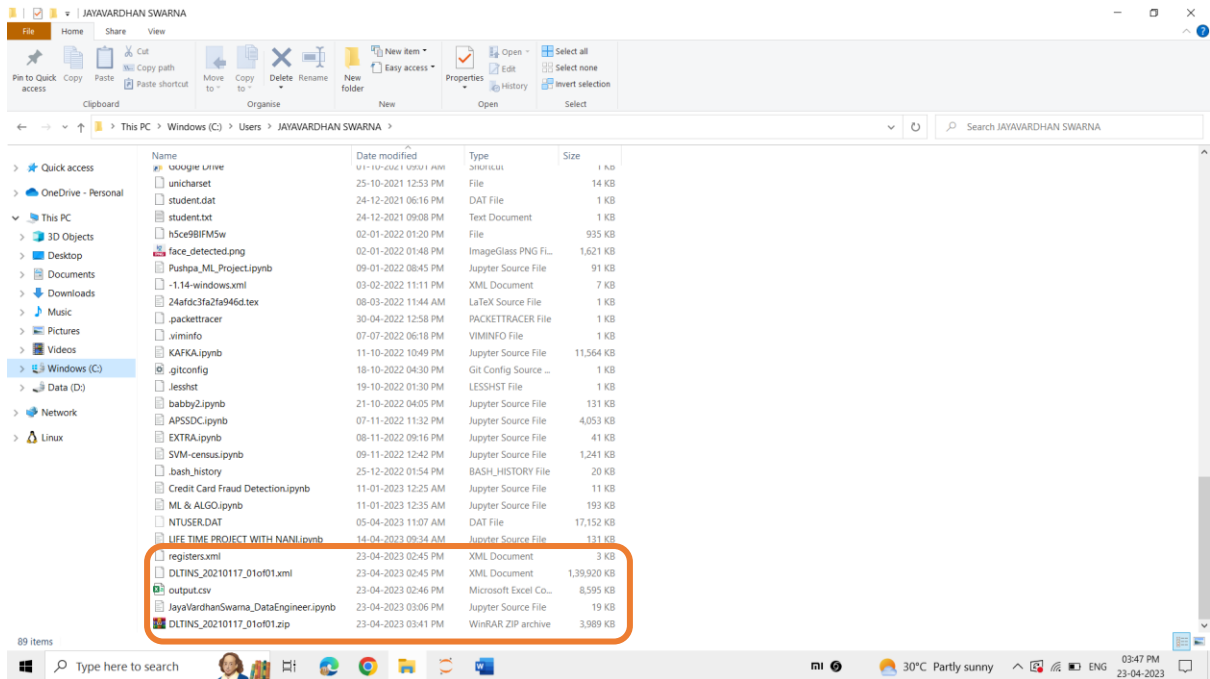
#Requirement 3: Extract the xml from the zip.
xml_file = p.zip_extraction(zip_link)

#Requirement 4: Convert the contents of the xml into a CSV
df = p.xml_to_csv(xml_file)
df.to_csv("output.csv")
```

All the zip, xml & csv files are downloaded.

The screenshot shows a Jupyter Notebook file browser with the following files and folders:

| File/Folder | Time | Size |
|-----------------------------------|---------------------|---------|
| APSSDC.ipynb | 6 months ago | 4.15 MB |
| EXTRA.ipynb | 5 months ago | 41.6 kB |
| SVM-census.ipynb | 5 months ago | 1.27 MB |
| Music | 4 months ago | |
| Credit Card Fraud Detection.ipynb | 3 months ago | 10.7 kB |
| ML & ALGO.ipynb | 3 months ago | 197 kB |
| Postman | 3 months ago | |
| Videos | a month ago | |
| OneDrive | a month ago | |
| LIFE TIME PROJECT WITH NANI.ipynb | 9 days ago | 134 kB |
| data | 3 days ago | |
| extracted | 2 days ago | |
| Downloads | a day ago | |
| registers.xml | a minute ago | 2.77 kB |
| zip_file.zip | a minute ago | 4.43 MB |
| DLTINS_20210117_01of01.xml | a minute ago | 143 MB |
| output.csv | a minute ago | 8.8 MB |
| SteelEye.ipynb | Running seconds ago | 19.4 kB |



Creating the iam User

The screenshot shows the AWS IAM console in the 'us-east-1' region. The left sidebar displays the 'Identity and Access Management (IAM)' menu with options like Dashboard, User groups, Roles, Policies, and Access reports. The main content area is titled 'Users (1)' and shows a table with one user: 'corestack-b7695'. The table columns are User name, Groups, Last activity, MFA, Password age, and Active key age. The user 'corestack-b7695' has no groups, last activity 'Never', no MFA, and a password age of '906 days ago'. Buttons for 'Add users', 'Delete', and 'Info' are visible at the top right of the table.

Iam user named as “steeleye-user1”

The screenshot shows the 'Create user' wizard in the AWS IAM console. The first step is 'Specify user details'. The 'User name' field is filled with 'steeleye-user1'. Below the field, a note states: 'The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , _ @ _ (hyphen)'. There is an unchecked checkbox for 'Provide user access to the AWS Management Console - optional'. A blue information box at the bottom states: 'If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more'. 'Cancel' and 'Next' buttons are at the bottom right.

Attaching the “S3FullAccess policy” to the “steeleye-user1”

The screenshot shows the AWS IAM console 'Set permissions' page. The left sidebar indicates the current step is 'Set permissions'. The main content area has a heading 'Set permissions' and a sub-heading 'Permissions options'. There are three radio buttons: 'Add user to group', 'Copy permissions', and 'Attach policies directly'. The 'Attach policies directly' option is selected. Below this, there is a section 'Permissions policies (1/1078)' with a search bar and a table of policies. The table has columns for 'Policy name', 'Type', and 'Attached entities'. The policy 'AmazonS3FullAccess' is selected.

Permissions options

- ☐ Add user to group
- ☐ Copy permissions
- ☒ Attach policies directly

Permissions policies (1/1078)

Choose one or more policies to attach to your new user.

Filter distributions by text, property or value 11 matches

Clear filters

| Policy name | Type | Attached entities |
|--|-------------|-------------------|
| <input type="checkbox"/> AmazonDMSRedshiftS3Role | AWS managed | 0 |
| <input checked="" type="checkbox"/> AmazonS3FullAccess | AWS managed | 0 |

The screenshot shows the AWS IAM console 'Review and create' page. The left sidebar indicates the current step is 'Review and create'. The main content area has a heading 'User details' and a sub-heading 'Permissions summary'. The 'User details' section shows the user name 'steeleye-user1', console password type 'None', and 'Require password reset' set to 'No'. The 'Permissions summary' section shows a table with columns for 'Name', 'Type', and 'Used as'. The policy 'AmazonS3FullAccess' is listed. There is also a 'Tags - optional' section with an 'Add new tag' button.

User details

User name: steeleye-user1

Console password type: None

Require password reset: No

Permissions summary

| Name | Type | Used as |
|--------------------|-------------|--------------------|
| AmazonS3FullAccess | AWS managed | Permissions policy |

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Cancel Previous Create user

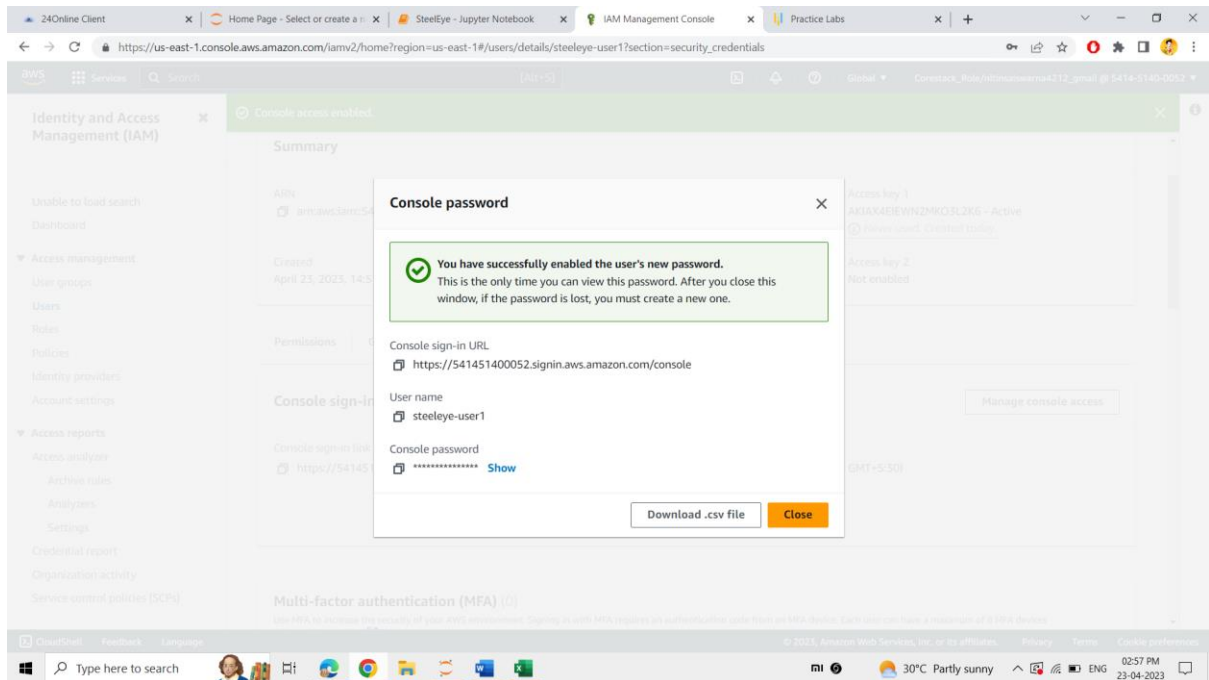
Steeleye-user1 has been created successfully with the property of “s3fullaccess”

The screenshot shows the AWS IAM Management Console interface. A green banner at the top indicates 'User created successfully'. The left sidebar shows the 'Users' link under 'Access management'. The main content area displays a table of users:

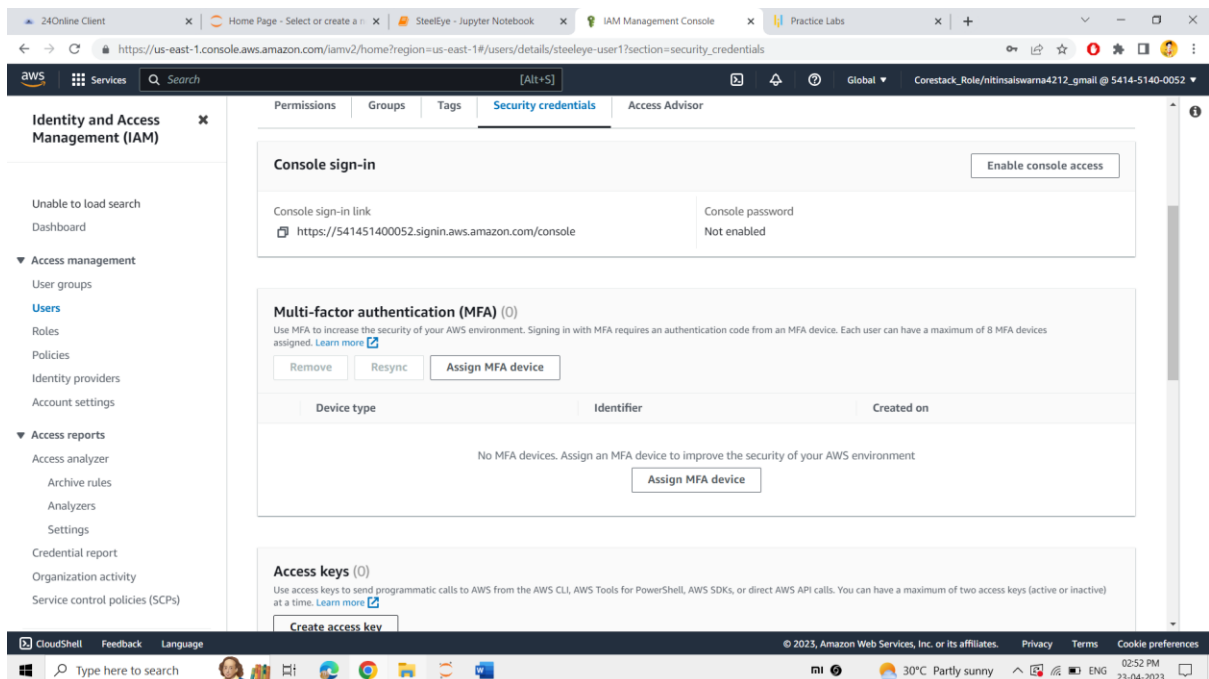
| | User name | Groups | Last activity | MFA | Password age | Active key age |
|--------------------------|-----------------|--------|---------------|------|--------------|----------------|
| <input type="checkbox"/> | corestack-b7695 | None | Never | None | 906 days ago | - |
| <input type="checkbox"/> | steeleye-user1 | None | Never | None | None | - |

Creating the password for the steeleye-user

The screenshot shows the 'Manage console access' dialog for the 'steeleye-user1' user. The dialog prompts to set a password, with options for 'Keep existing password', 'Autogenerated password', and 'Custom password'. The 'Custom password' option is selected, and a password field is visible. The dialog also includes a checkbox for 'User must create new password at next sign-in'.



To automate using the boto3 library, Access key is necessary.



Creating the Access key for the steel-user1

The screenshot shows the AWS IAM console interface. The breadcrumb navigation is IAM > Users > steel-user1 > Create access key. On the left, a sidebar shows three steps: Step 1: Access key best practices & alternatives (selected), Step 2 - optional: Set description tag, and Step 3: Retrieve access keys. The main content area is titled 'Access key best practices & alternatives' and includes a warning: 'Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.' Below this, there are five radio button options: 'Command Line Interface (CLI)', 'Local code' (selected), 'Application running on an AWS compute service', 'Third-party service', and 'Application running outside AWS'. Each option has a brief description of its use case. At the bottom of the console, there is a footer with 'CloudShell', 'Feedback', 'Language', and copyright information for Amazon Web Services, Inc. The Windows taskbar at the very bottom shows the search bar, task view, and several application icons.

The screenshot shows the AWS IAM console interface after the access key has been created. A green banner at the top states: 'Access key created. This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' The breadcrumb navigation remains the same. The sidebar now highlights Step 3: Retrieve access keys. The main content area is titled 'Retrieve access keys' and includes a warning: 'If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.' Below this, there is a table with two columns: 'Access key' and 'Secret access key'. The 'Access key' column contains the value 'AKIA4EIEWN2MKO3L2K6' with a copy icon. The 'Secret access key' column contains a masked value '*****' with a 'Show' link. Below the table, there is a section titled 'Access key best practices' with a bulleted list: 'Never store your access key in plain text, in a code repository, or in code.', 'Disable or delete access key when no longer needed.', 'Enable least-privilege permissions.', and 'Rotate access keys regularly.' A link to 'Best practices for managing AWS access keys' is provided. At the bottom right, there are buttons for 'Download .csv file' and 'Done'. The footer and Windows taskbar are identical to the previous screenshot.

Logging into the steeleye-user aws account

The screenshot displays the AWS Management Console interface. The top navigation bar shows the user is logged in as 'steeleye-user1' in the 'ap-northeast-1' region. The main content area is titled 'Console Home' and features a 'Recently visited' section with links to S3, Lambda, and EC2. A 'Welcome to AWS' sidebar on the right provides links to 'Getting started with AWS', 'Training and certification', and 'What's new with AWS?'. The bottom of the console shows a footer with '© 2023, Amazon Web Services, Inc. or its affiliates.' and links to 'Privacy', 'Terms', and 'Cookie preferences'.

The browser window shows the URL: https://ap-southeast-2.signin.aws.amazon.com/oauth?client_id=am%3Aaws%3Asignin%3A%3Aconsole%2Fcanvas&code_challenge=TNyIK...

Sign in as IAM user

Account ID (12 digits) or account alias
541451400052

IAM user name
steeleye-user1

Password
.....

☐ Remember this account

Sign in

[Sign in using root user email](#)
[Forgot password?](#)

[English](#)

[Terms of Use](#) [Privacy Policy](#) © 1996-2023 Amazon Web Services, Inc. or its affiliates

AWS Training and Certification
Propel your career. Get AWS certified
[LEARN MORE](#)

Console Home [Info](#)

[Reset to default layout](#) [+ Add widgets](#)

Recently visited [Info](#)

- S3
- Lambda
- EC2

[View all services](#)

Welcome to AWS

- [Getting started with AWS](#)
Learn the fundamentals and find valuable information to get the most out of AWS.
- [Training and certification](#)
Learn from AWS experts and advance your skills and knowledge.
- [What's new with AWS?](#)
Discover new AWS services, features, and Regions.

AWS Health [Info](#)

Cost and usage [Info](#)

CloudShell Feedback Language © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Creation of Steeleyedata bucket to store the csv file.

The image consists of two screenshots from the AWS S3 console, showing the process of creating and managing a bucket named 'steeleyedata'.

Top Screenshot: Create bucket page

The URL is `https://s3.console.aws.amazon.com/s3/bucket/create?region=ap-northeast-1`. The page title is "Create bucket". Below the title, it says "Buckets are containers for data stored in S3. [Learn more](#)".

General configuration

- Bucket name:** `steeleyedata`. A note states: "Bucket name must be globally unique and must not contain spaces or uppercase letters. See [rules for bucket naming](#)".
- AWS Region:** `Asia Pacific (Tokyo) ap-northeast-1`.
- Copy settings from existing bucket - optional:** Only the bucket settings in the following configuration are copied. A button labeled "Choose bucket" is present.

Object Ownership

- ACLs disabled (recommended):** All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using...
- ACLs enabled:** Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be...

Bottom Screenshot: S3 Management Console

The URL is `https://s3.console.aws.amazon.com/s3/buckets?region=ap-northeast-1`. A green banner at the top says "Successfully created bucket 'steeleyedata'". Below the banner, the page title is "Amazon S3 > Buckets".

Account snapshot

Last updated: Oct 14, 2022 by Storage Lens. Metrics are generated every 24 hours. [Learn more](#)

| Total storage | Object count | Average object size |
|---------------|--------------|---------------------|
| 1.2 MB | 5 | 244.9 KB |

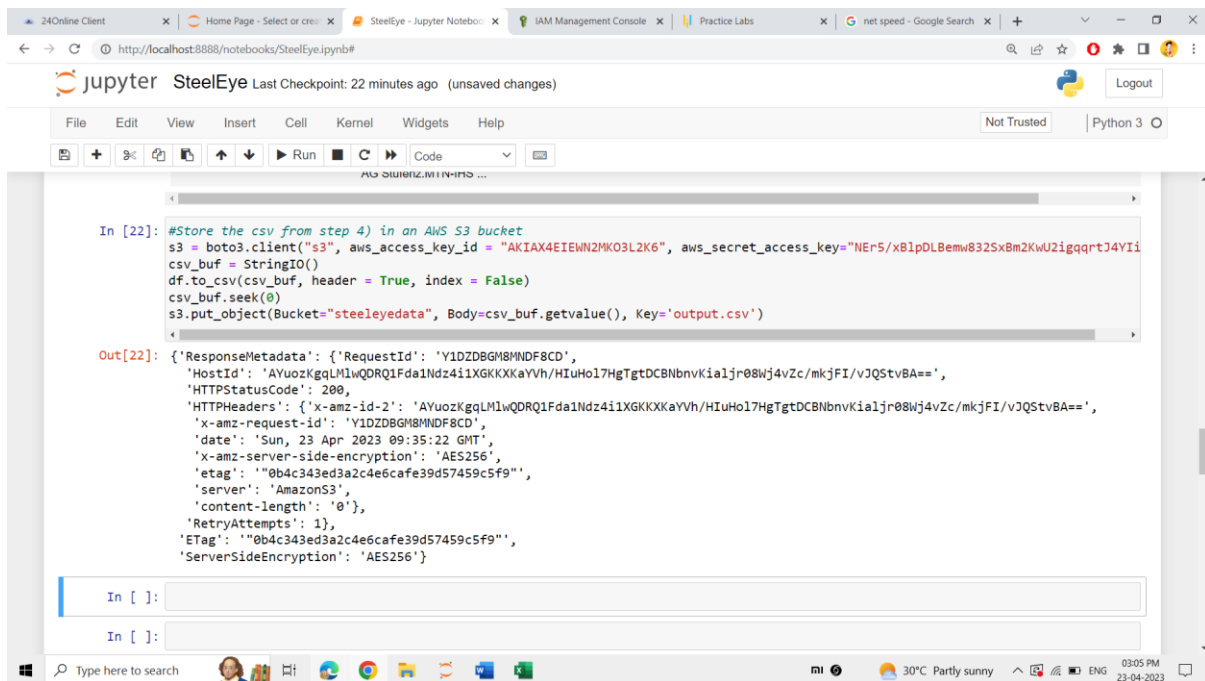
Buckets (1)

Buckets are containers for data stored in S3. [Learn more](#)

Find buckets by name

| Name | AWS Region | Access | Creation date |
|---------------------------|-------------------------------------|-------------------------------|--------------------------------------|
| <code>steeleyedata</code> | Asia Pacific (Tokyo) ap-northeast-1 | Bucket and objects not public | April 23, 2023, 15:01:40 (UTC+05:30) |

The given code helps in uploading the csv file into the bucket of the steeleye-user.

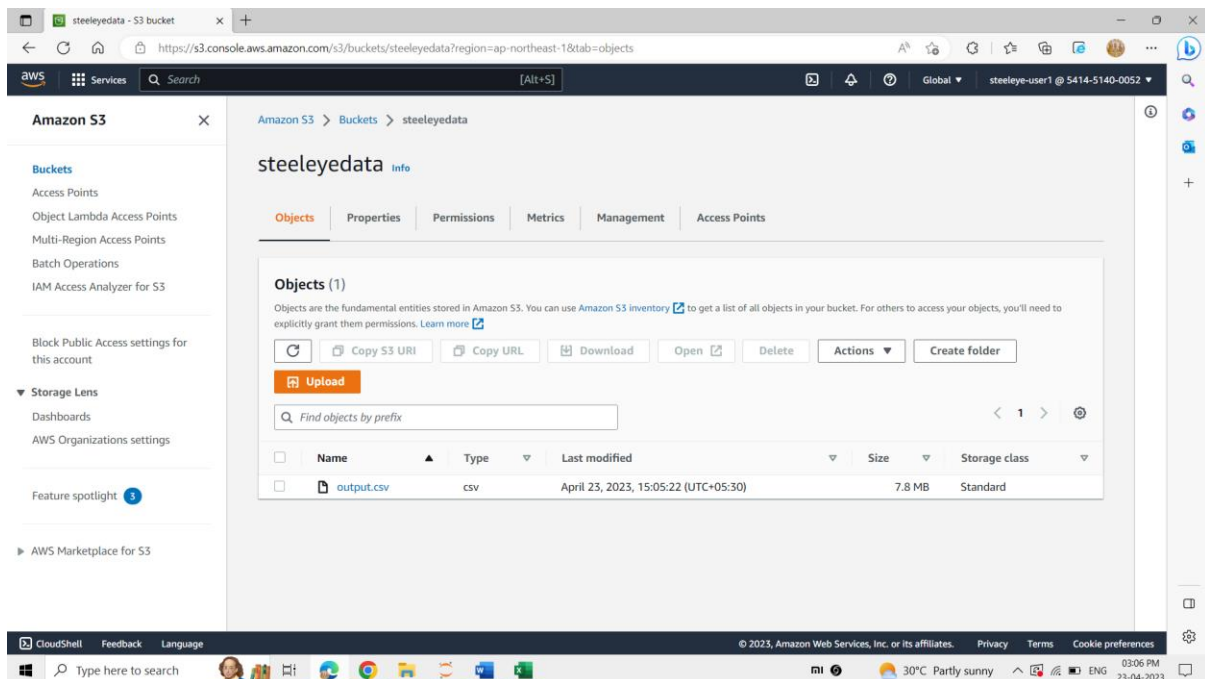


The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [22]: #Store the csv from step 4) in an AWS S3 bucket
s3 = boto3.client("s3", aws_access_key_id = "AKIA4EIEWN2MK03L2K6", aws_secret_access_key="NEr5/xB1pDLBemw8325x8m2KwU2igqqrTj4YiI")
csv_buf = StringIO()
df.to_csv(csv_buf, header = True, index = False)
csv_buf.seek(0)
s3.put_object(Bucket="steeleyedata", Body=csv_buf.getvalue(), Key='output.csv')
```

```
Out[22]: {'ResponseMetadata': {'RequestId': 'Y1DZDBG8M8MND8F8CD',
'HostId': 'AYuozKgqLMlwQDRQ1Fda1Nd24i1XGKKXKaYVh/HIuHo17HgTgtDCBNbnvKialjr08Wj4vZc/mkjFI/vJQStvBA==',
'HTTPStatusCode': 200,
'HTTPHeaders': {'x-amz-id-2': 'AYuozKgqLMlwQDRQ1Fda1Nd24i1XGKKXKaYVh/HIuHo17HgTgtDCBNbnvKialjr08Wj4vZc/mkjFI/vJQStvBA==',
'x-amz-request-id': 'Y1DZDBG8M8MND8F8CD',
'date': 'Sun, 23 Apr 2023 09:35:22 GMT',
'x-amz-server-side-encryption': 'AES256',
'etag': '"0b4c343ed3a2c4e6cafe39d57459c5f9"',
'server': 'AmazonS3',
'content-length': '0'},
'RetryAttempts': 1},
'ETag': '"0b4c343ed3a2c4e6cafe39d57459c5f9"',
'ServerSideEncryption': 'AES256'}
```

When the above code was started executing the csv file started to upload into the steeleyedata bucket.



The file was uploaded successfully.

...Thanking You...