**Cloud Computing**

| S.No | Title | Page No | Sign |
|---|---|---|---|
| 1 | a)Working with Google Drive to make spreadsheet<br><br>b)Working with Google Drive to make notes | | |
| 2 | Launch a Linux Virtual Machine. | | |
| 3 | To host a static website | | |
| 4 | Exploring Google cloud for the following<br>a) Storage<br>b) Sharing of data<br>c) manage your calendar, to-do lists<br>d) a document editing tool | | |
| 5 | Working and installation of Google App Engine | | |
| 6 | Working and installation of Microsoft Azure | | |
| 7 | To Connect Amazon Redshift with S3 bucket | | |
| 8 | To Create and Query a NoSQL Table | | |

| Ex.No:1 | A) Working with Google Drive to Create Spreadsheets and Notes |
|---|---|
| | |

**AIM:**

To create, edit, and manage spreadsheets and notes using Google Drive for efficient data organization and collaboration.

**ALGORITHM:**

**Step 1:** Sign in to your Google account and open Google Drive.

**Step 2:** Click on the **"New"** button and select **Google Sheets** to create a new spreadsheet.

**Step 3:** Rename the file by clicking on the default title at the top.

**Step 4:** Add, edit, and format text, numbers, or formulas as required in the spreadsheet.

**Step 5:** Utilize built-in functions and formulas for calculations and data analysis.

**Step 6:** Move the file to a specific folder by clicking the **"Move"** icon.

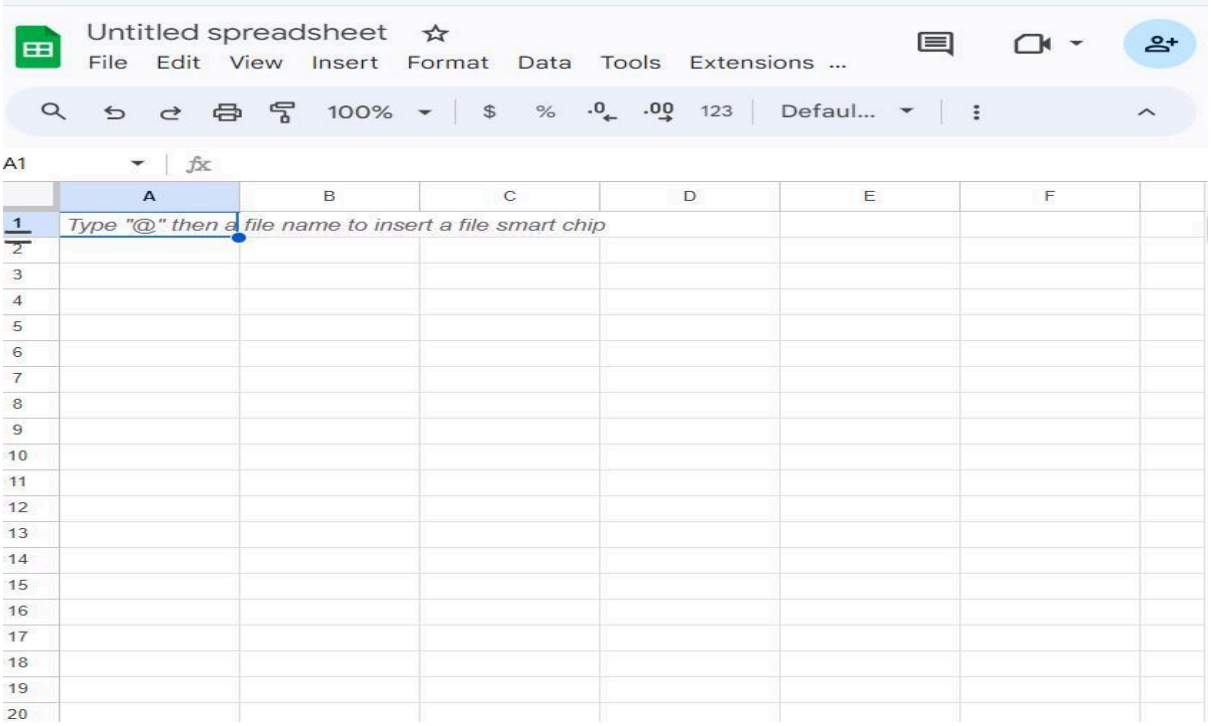
**Step 7:** Share the spreadsheet with others by clicking the **"Share"** button and granting appropriate access.

**Step 8:** Use the **commenting** and **notes** features to add annotations or collaborate on data.

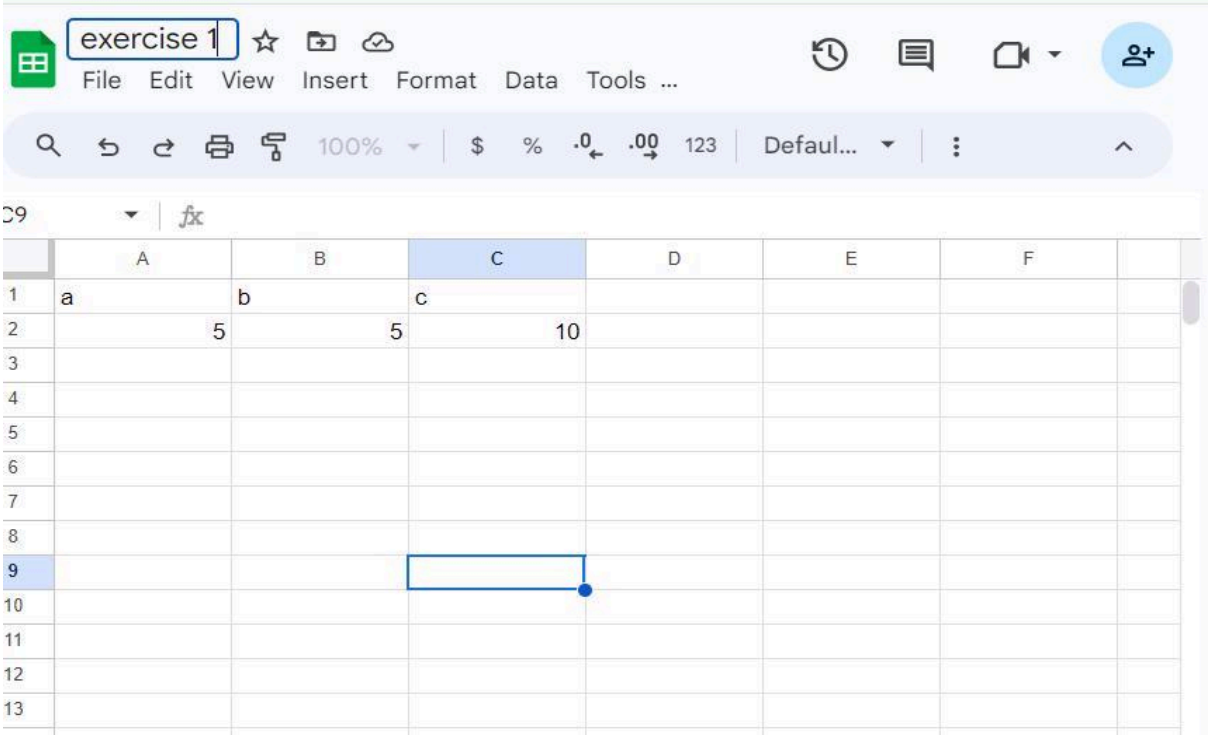
**Step 9:** Close the Google Spreadsheet and sign out of your Google account for security.

**PROGRAM:**

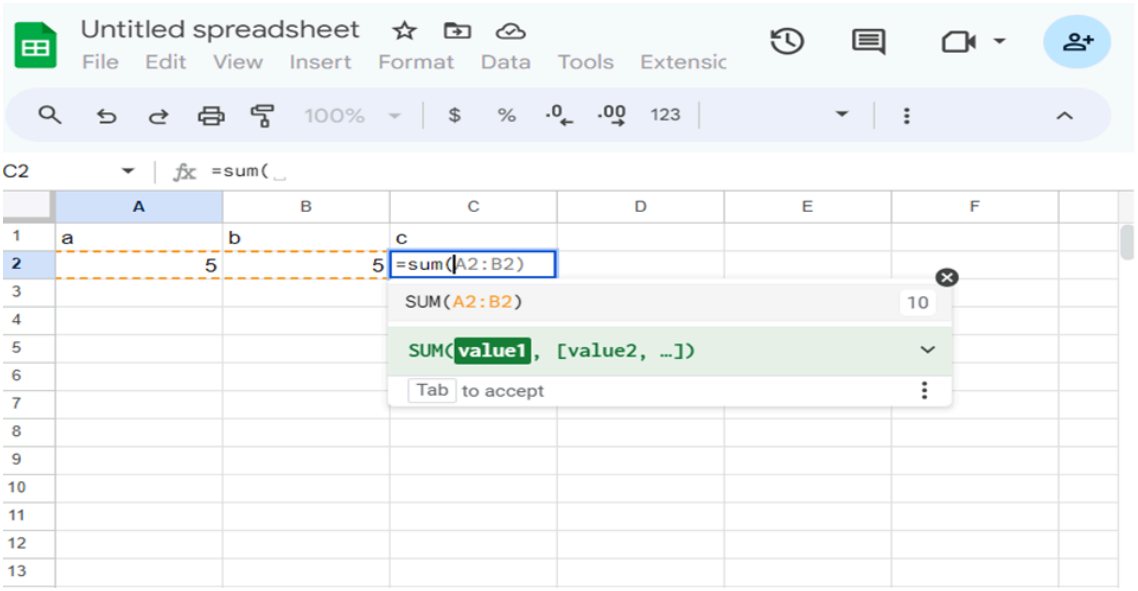
- Open a new blank spreadsheet or from a template.



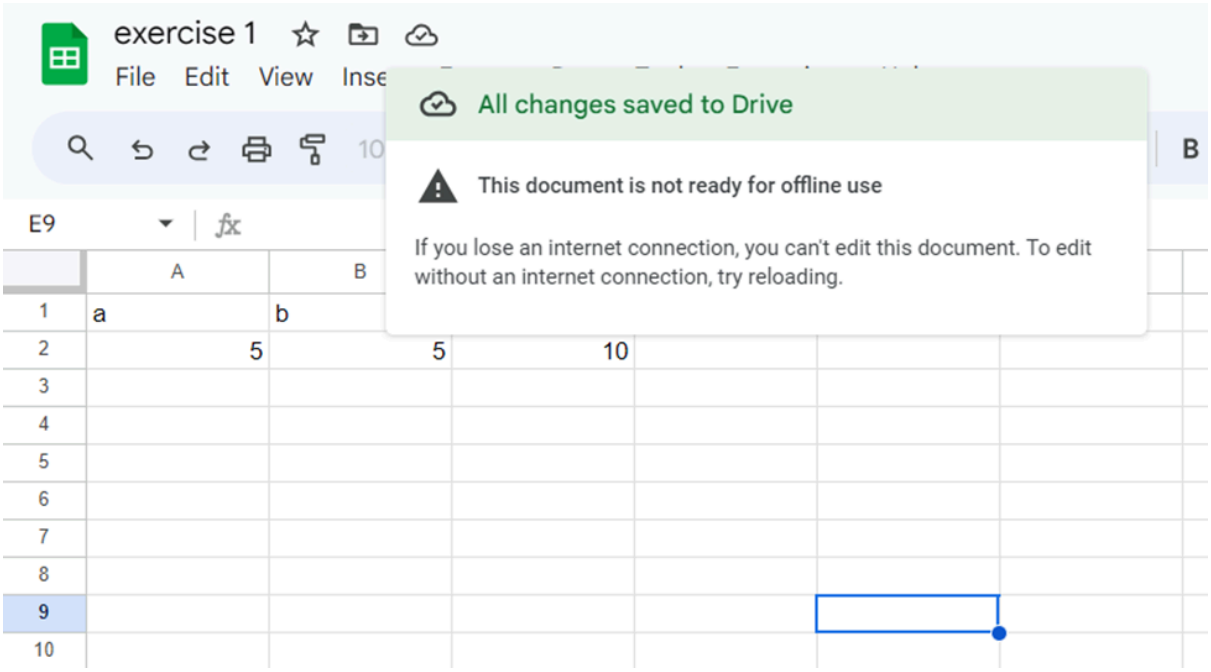

- We can change the filename, by double clicking the spreadsheet filename as untitled.



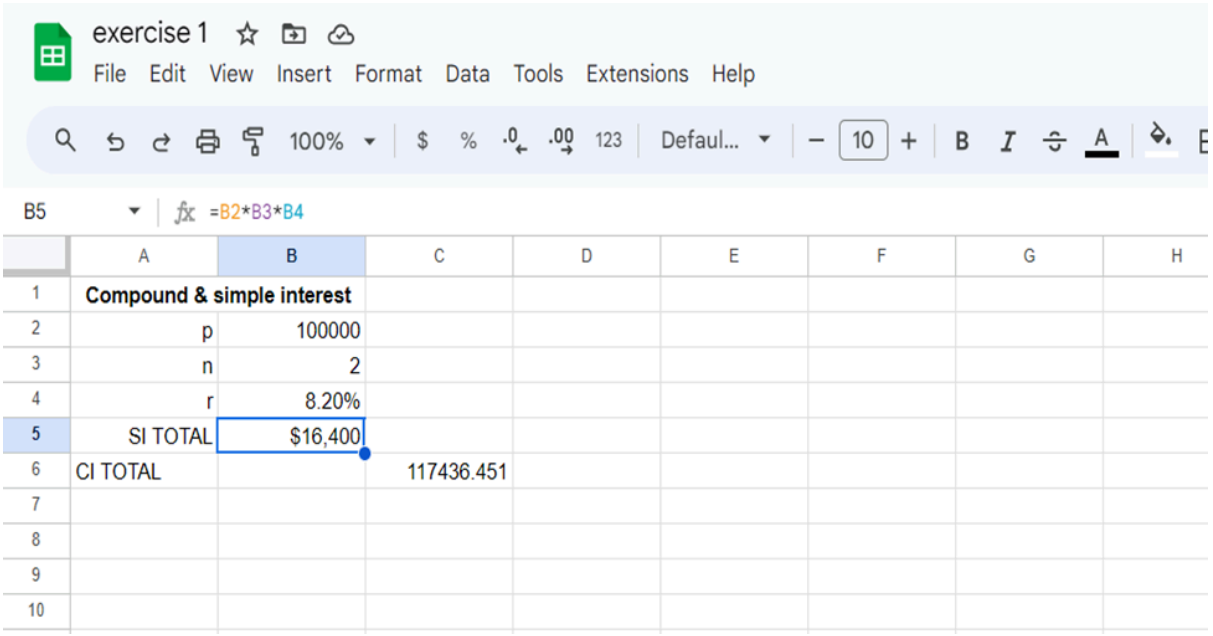

- Click a cell that's empty or double click a cell that isn't empty and start typing. When we done, press enter.

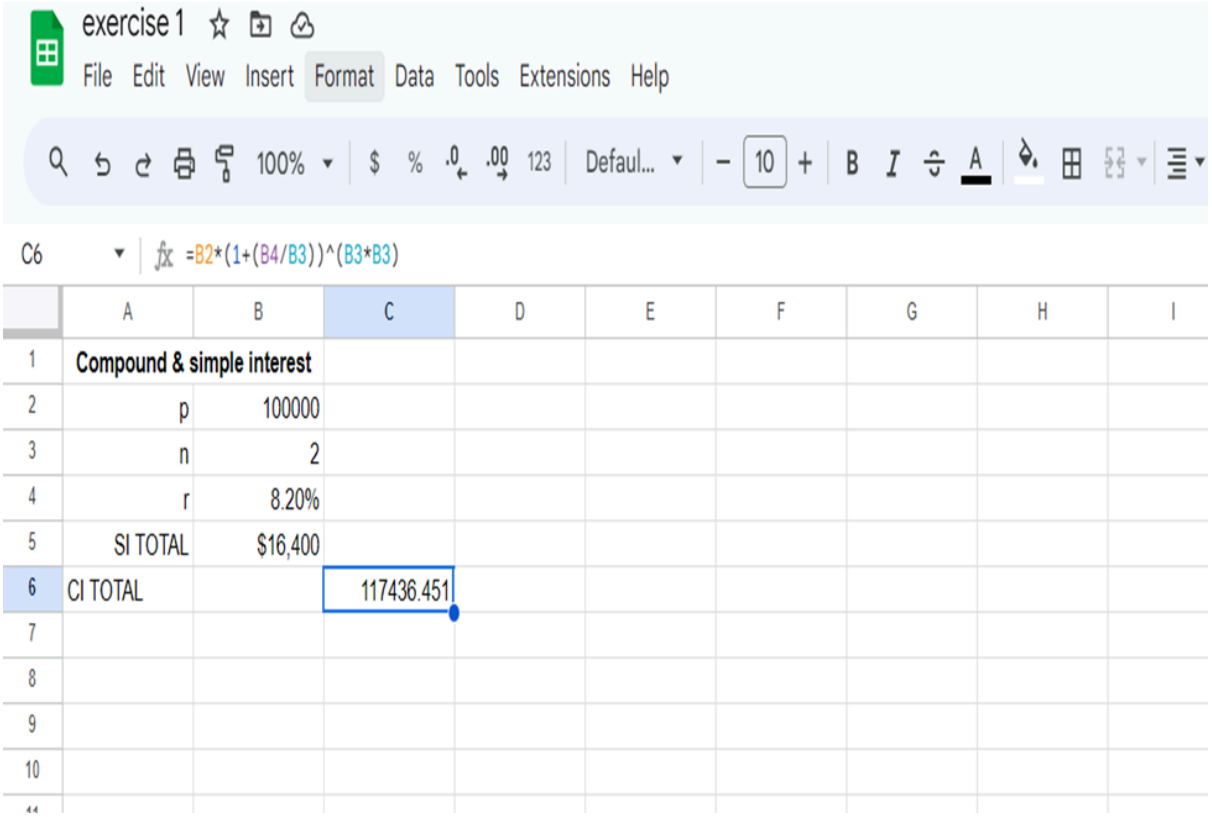



- All changes are automatically saved by Google instead of using save option in normal spreadsheet.
- We can also see the spreadsheet status by clicking this icon.

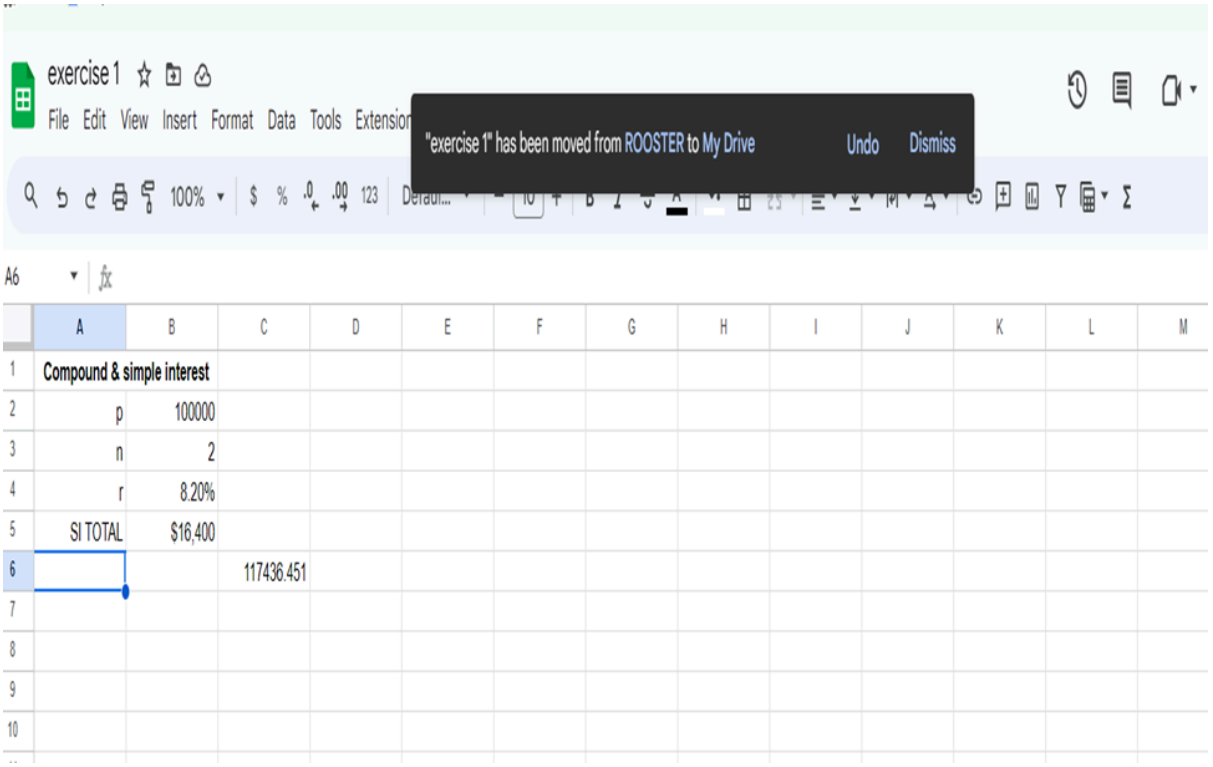



- Use the formula and functions to the spreadsheet
- Type an equal sign (=) in a cell and type in the function you want to use.

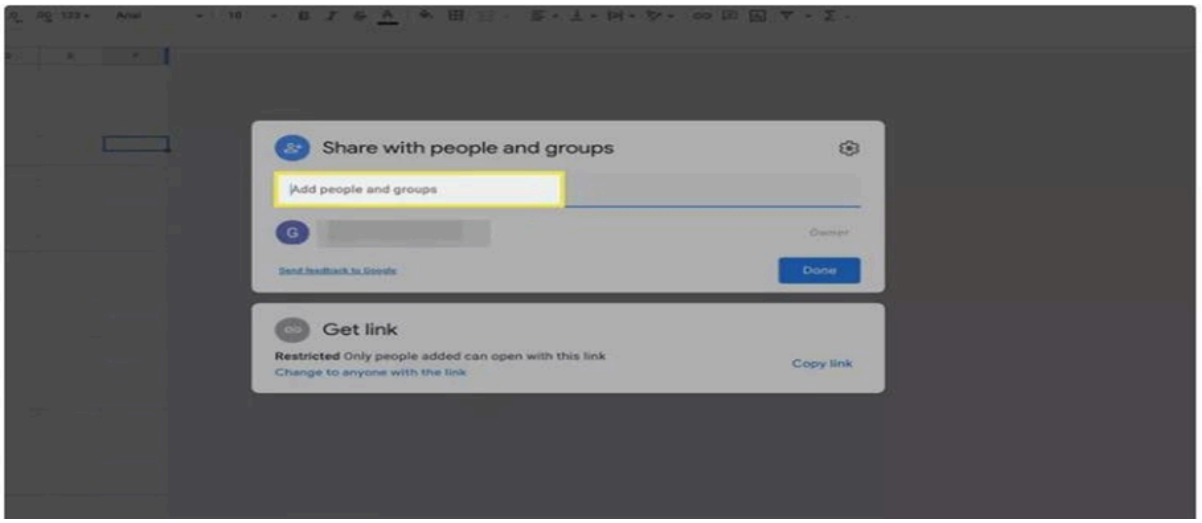


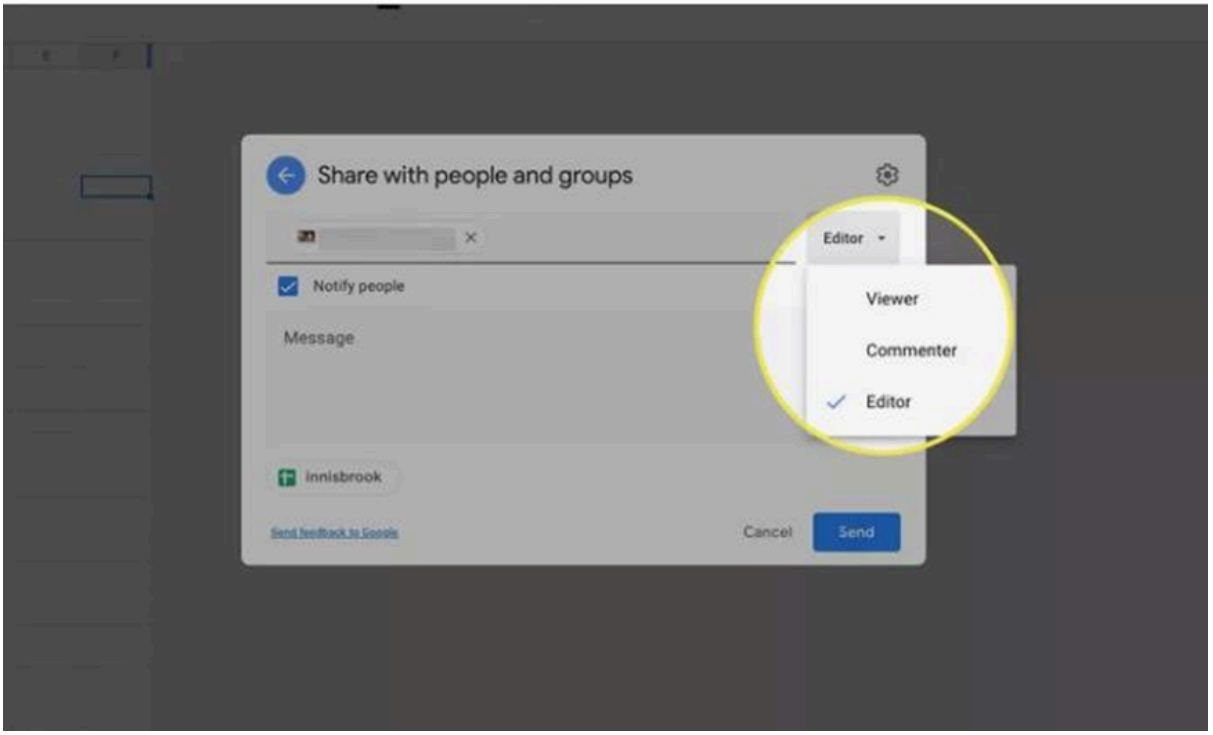

- Move our file into destination folder by using  icon.



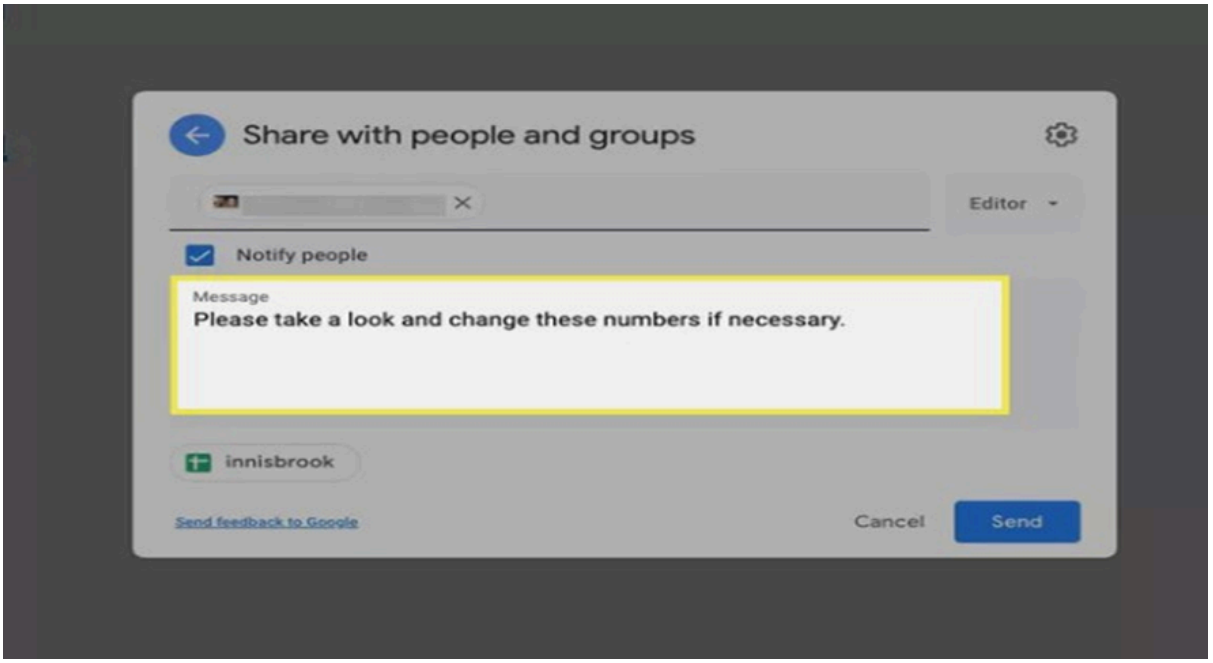

- Share files from Google Drive
    - On your computer, go to Google Drive.
    - Select the file you want to share Share.
    - Enter the email address you want to share with. If you use a work or personal account, you can share with suggested recipients.
    - In the Share with People and Groups dialog box, add the email addresses of the people you want to invite to view, comment on, or edit your Google Sheets file.



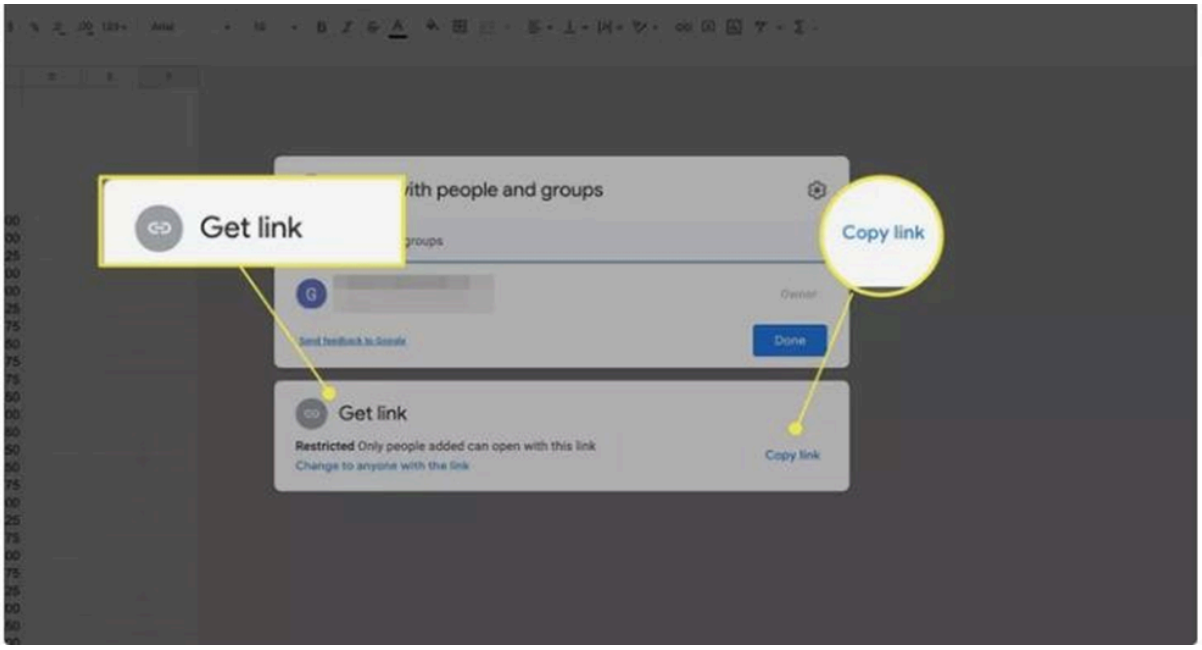
- Decide how people can use your file. Select one:
    - ❖ Viewer
    - ❖ Commenter
    - ❖ Editor




- Add a note to accompany the invitation, then select Send.



- Alternatively, open your Google Sheets file, select Share, and in the Get link box, choose Copy Link. The link is copied to your clipboard, and you can paste it into an email message to send to recipients that way.




- To stop sharing a Google Sheets file, select Share. In the drop-down menu next to the collaborator's name, select Remove.
- Click Send or share button. This shares the spreadsheet with the selected recipients. Anyone who receives the link can click it to open and edit the spreadsheet.

**RESULT:**

Thus, the above program has been executed successfully

| Ex.No:1 | B) Working with Google Drive to Create Notes |
|---|---|
| | |

**AIM:**

To create, edit, format, and share notes using Google Drive for easy documentation and collaboration.

**ALGORITHM:**

**Step 1:** Sign in to your Google account and open Google Drive.

**Step 2:** Click on the **"New"** button and select **Google Docs** to create a new document.

**Step 3:** Rename the file by clicking on the default title at the top.

**Step 4:** Add, edit, and format text in the document as required.

**Step 5:** Insert tables and images using the **"Insert"** option.

**Step 6:** Add a table with one row and two columns, then insert an image in each column.

**Step 7:** Move the document to a specific folder by clicking the **"Move"** icon.

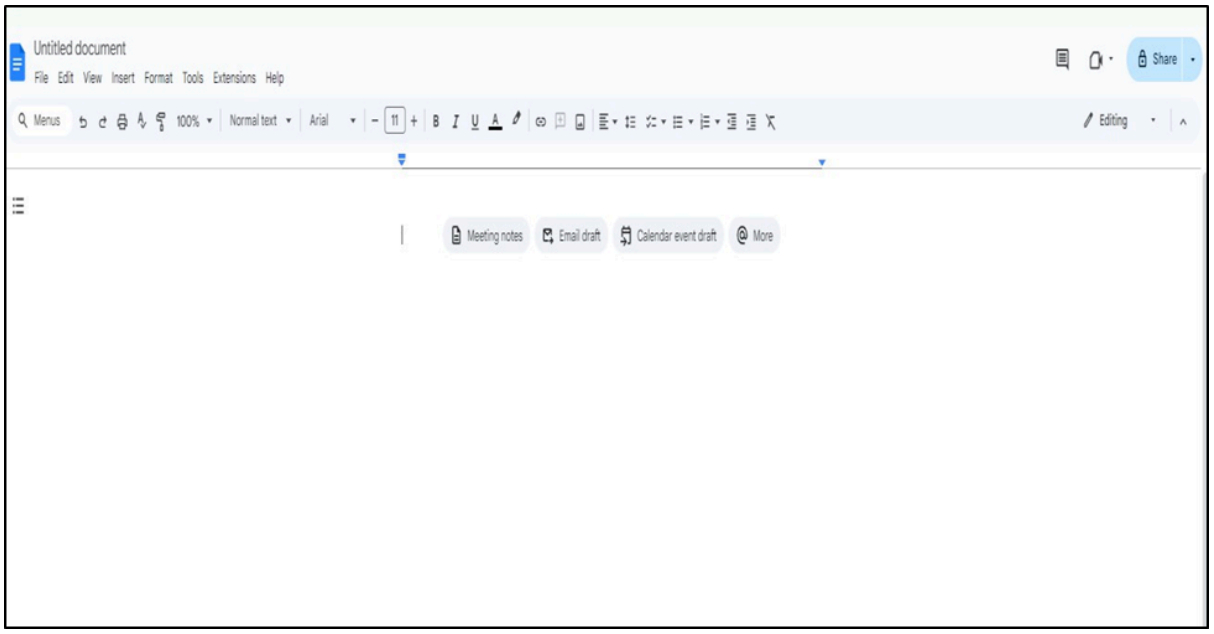
**Step 8:** Share the document by clicking the **"Share"** button and selecting access permissions such as **Can Edit**, **Can Comment**, or **Can View**.

**Step 9:** Share the document link via email or social media.

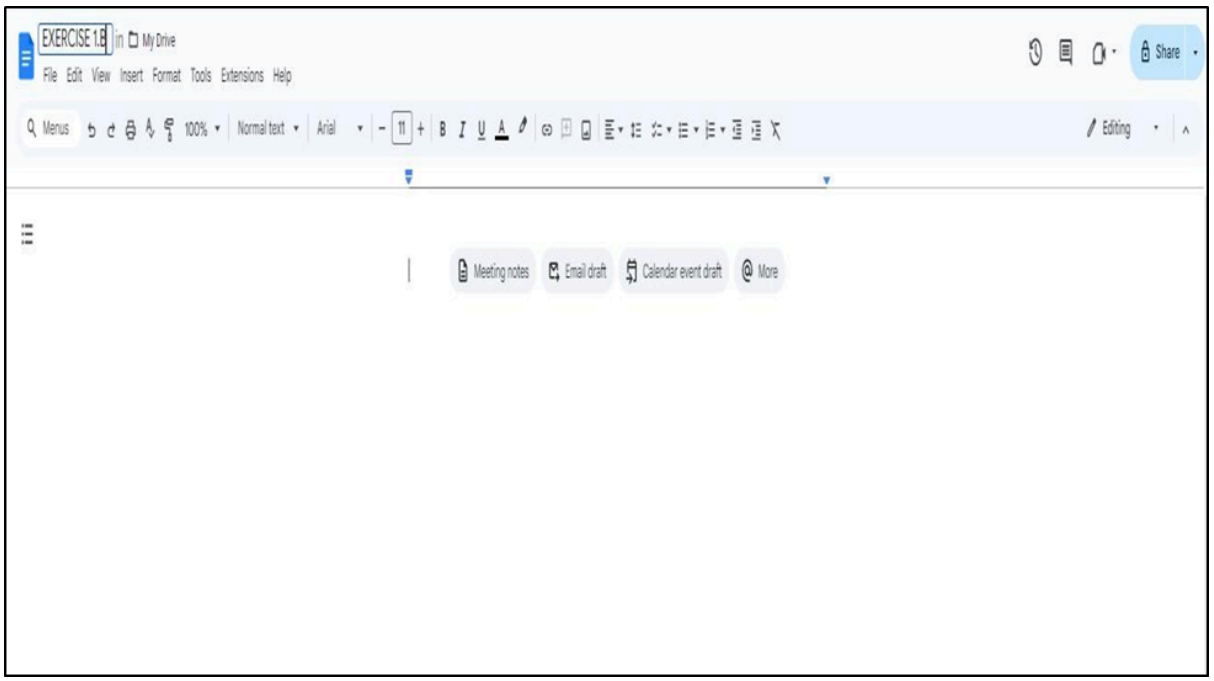
**Step 10:** After completing the process, close Google Docs and sign out of the Google account.

**PROGRAM:**

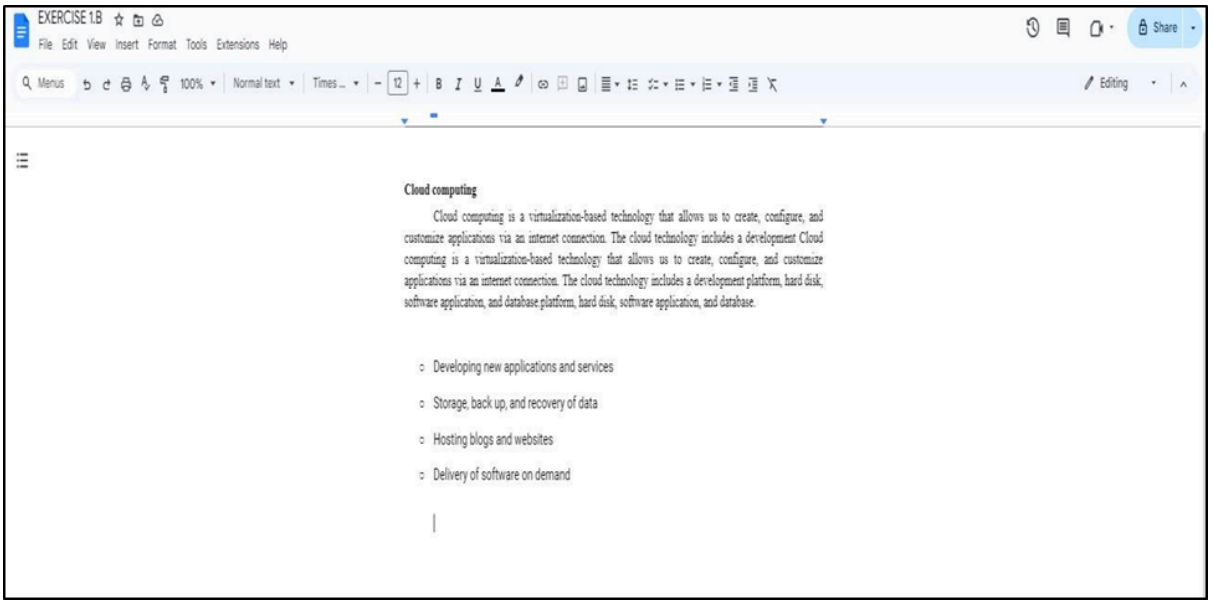
- Create a new Google document.



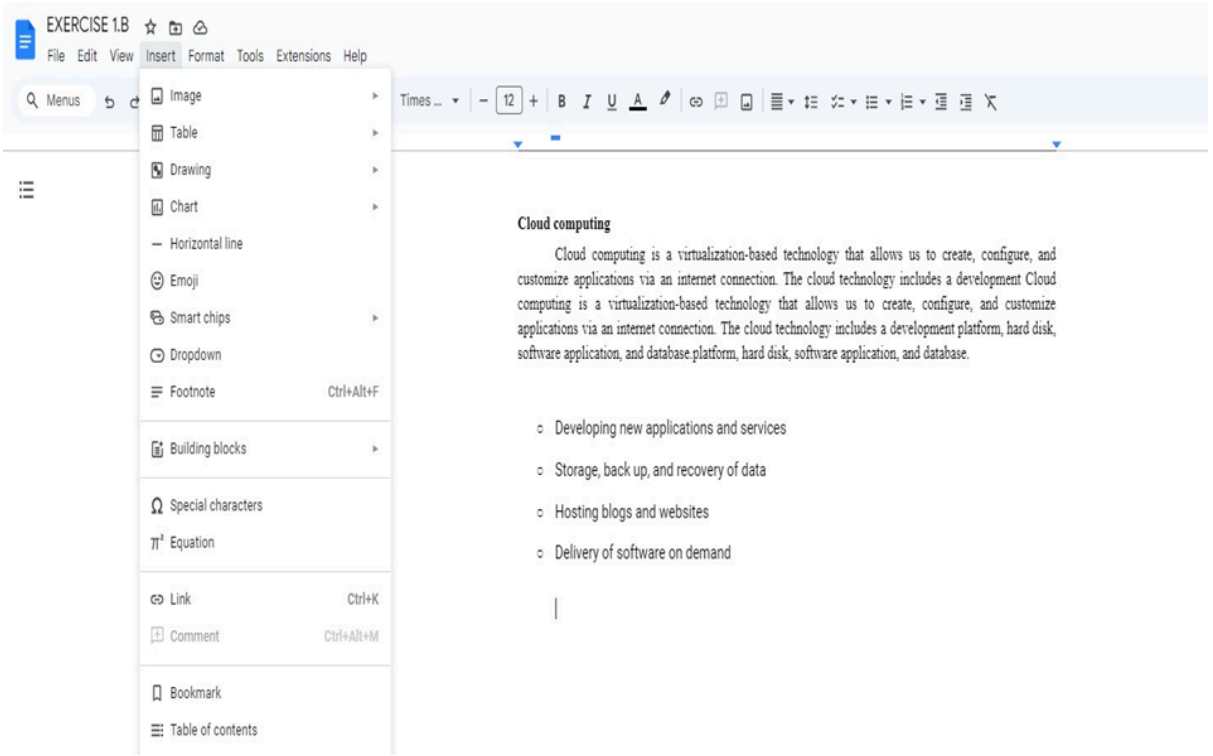

- Rename the file appropriately.

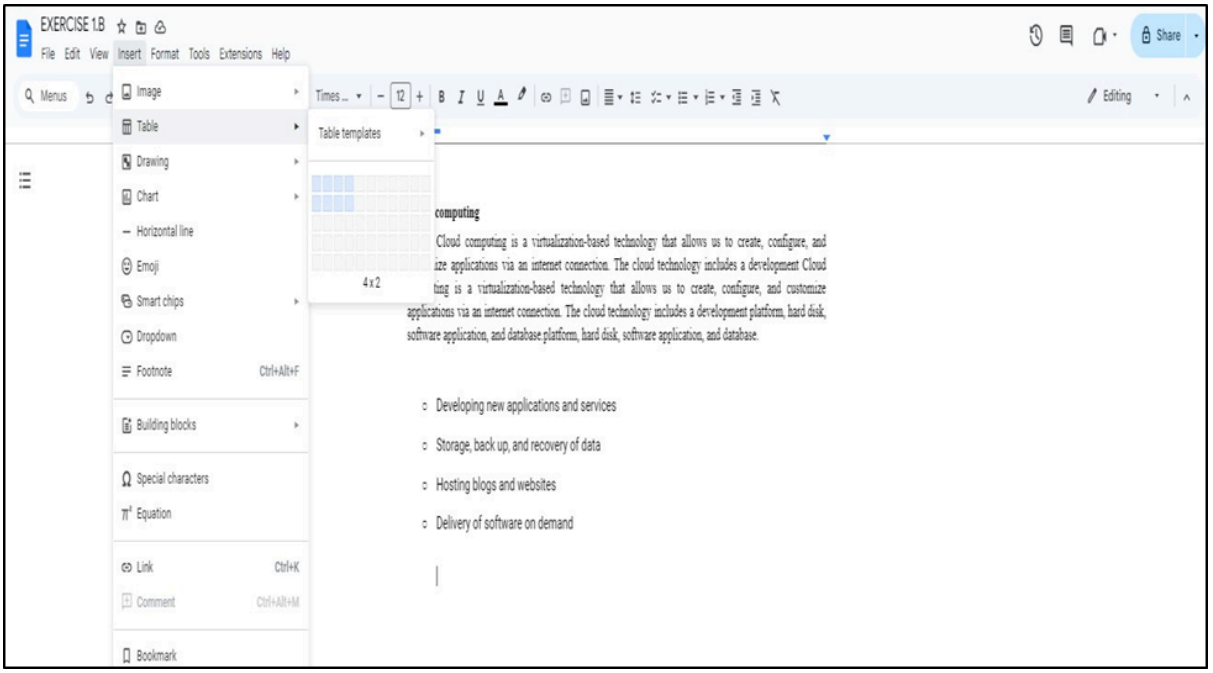


- Add, edit, and format text in the document.

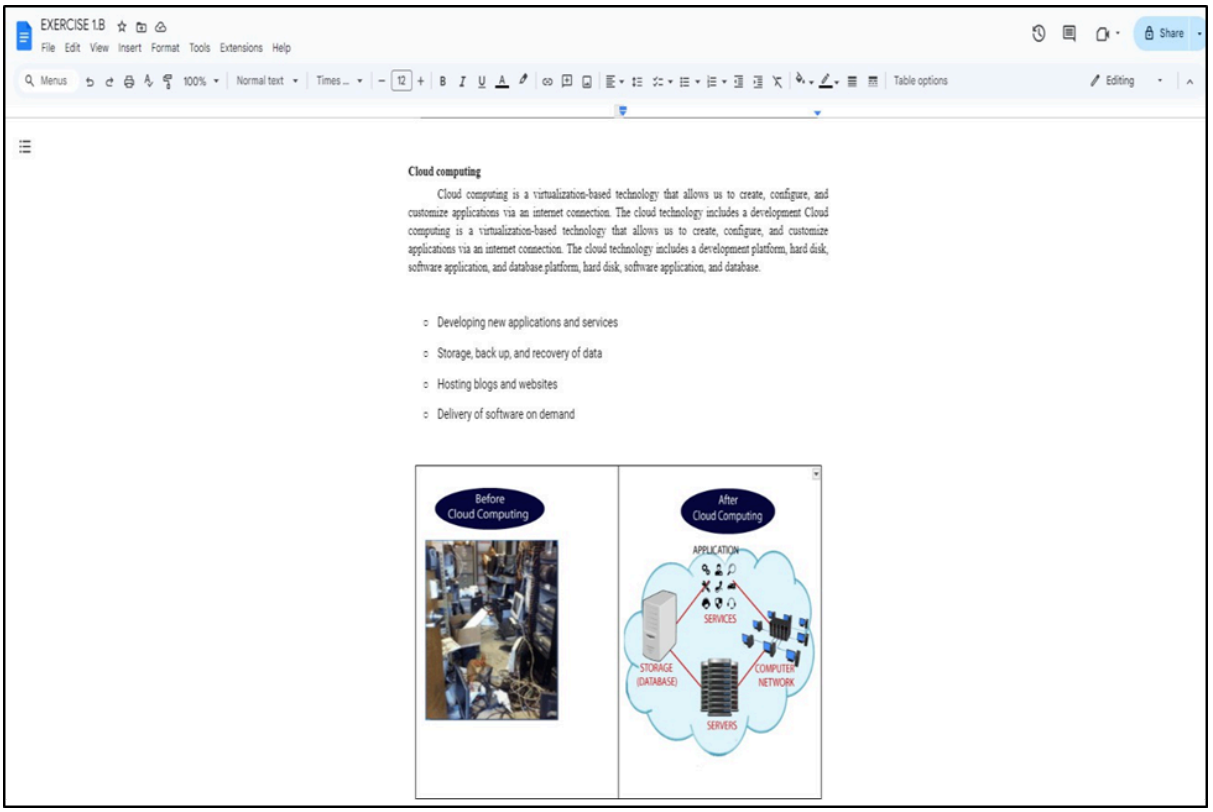



- Insert tables and images using the **Insert** option.

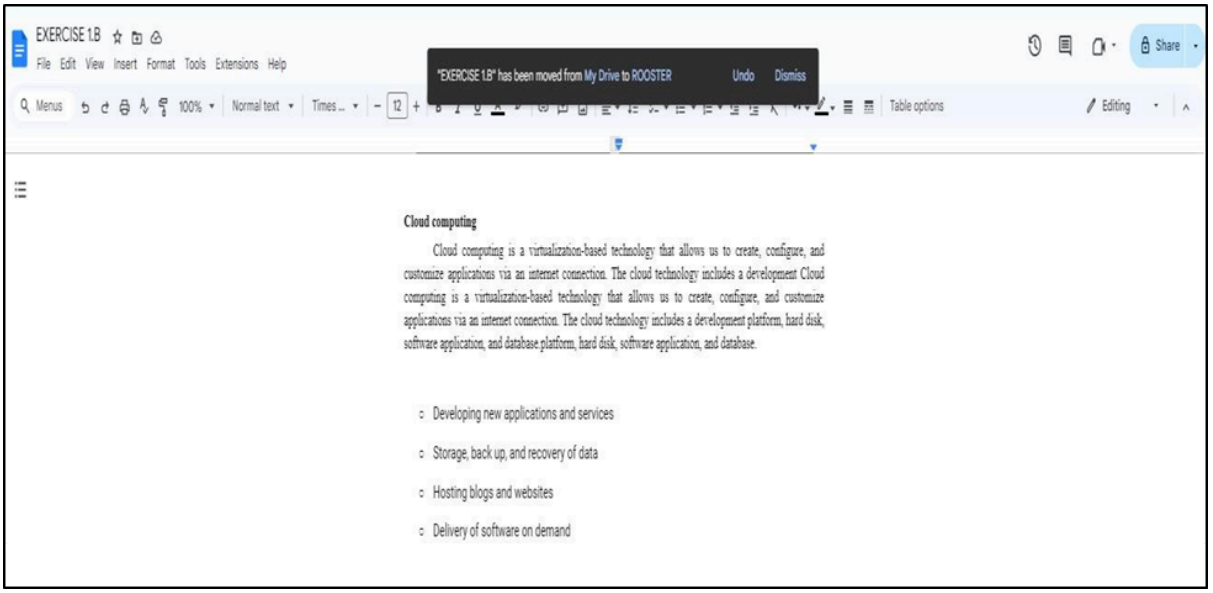




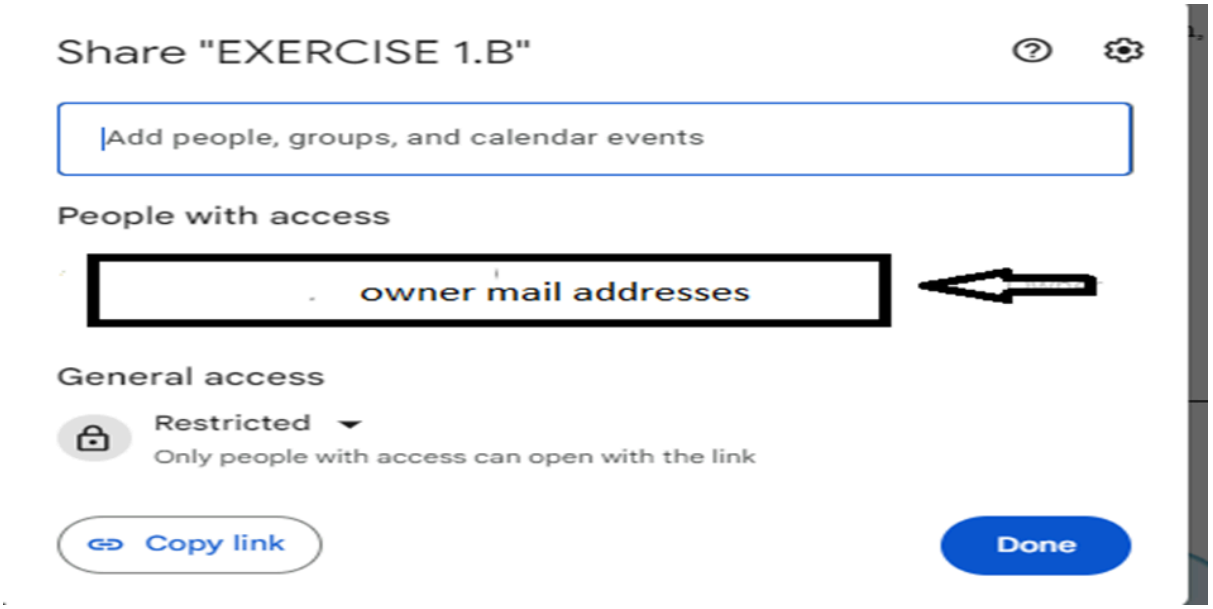

- We inserted one row and two columns. After that we insert a picture in each column by using insert command.



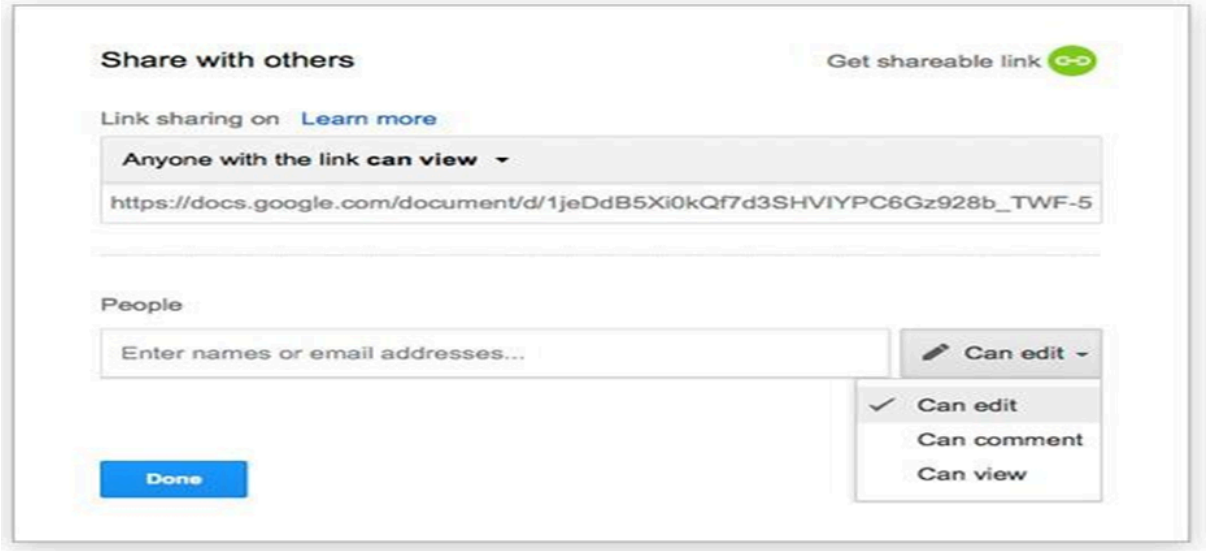
- We use to move our file into destination folder by using clicking this icon.



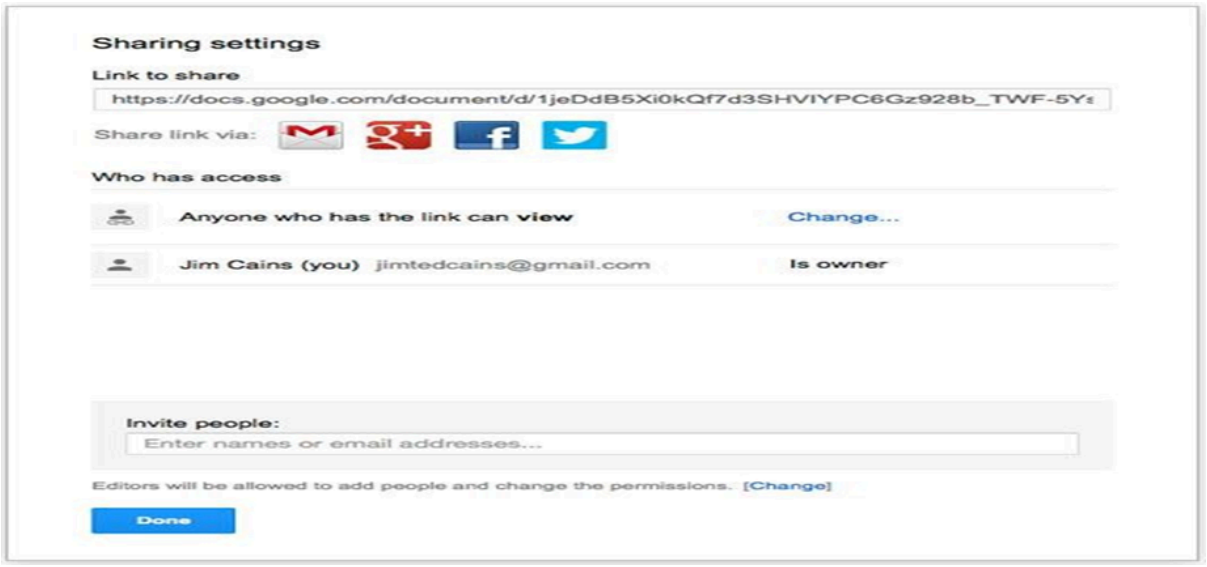

- Share the file with others by selecting appropriate access permissions.




- Share the document via email, link, or social media.




- Click Send or share button. This shares the documents with the selected recipients. Anyone who receives the link can click it to open and edit the document.
- We can also share the file with social media also.




- Close the document and sign out after completing the work.

**RESULT:**

Thus, the document creation, editing, and sharing process using Google Drive was successfully executed.

| Ex.No:2 | Launching a Linux Virtual Machine |
|---|---|
| | |

**AIM:**

To set up and launch a Linux virtual machine using VirtualBox.

## ALGORITHM:

**Step 1:** Open **VirtualBox** and click on **New** to create a virtual machine.

**Step 2:** Provide a name for the virtual machine, select the operating system type, version, and storage location, then click **Next**.

**Step 3:** Allocate the desired **RAM** for the virtual machine using the memory slider.

**Step 4:** Create a **virtual hard disk** by selecting **Create a virtual hard disk now**, then click **Next**.

**Step 5:** Choose the **VDI (VirtualBox Disk Image)** format and select **Dynamically Allocated** storage.

**Step 6:** Specify the virtual hard disk file location and size, then click **Create**.

**Step 7:** Select the newly created virtual machine and open **Settings**. Navigate to **Storage**, click on **Controller: IDE**, and select **Add** to attach an ISO image.

**Step 8:** Browse and select the **ISO image** of the Linux distribution for installation, then click **OK**.

**Step 9:** Click **Start** to boot the virtual machine from the selected ISO image.

**Step 10:** Proceed with the Linux installation by following on-screen instructions, selecting installation options, and configuring user settings.

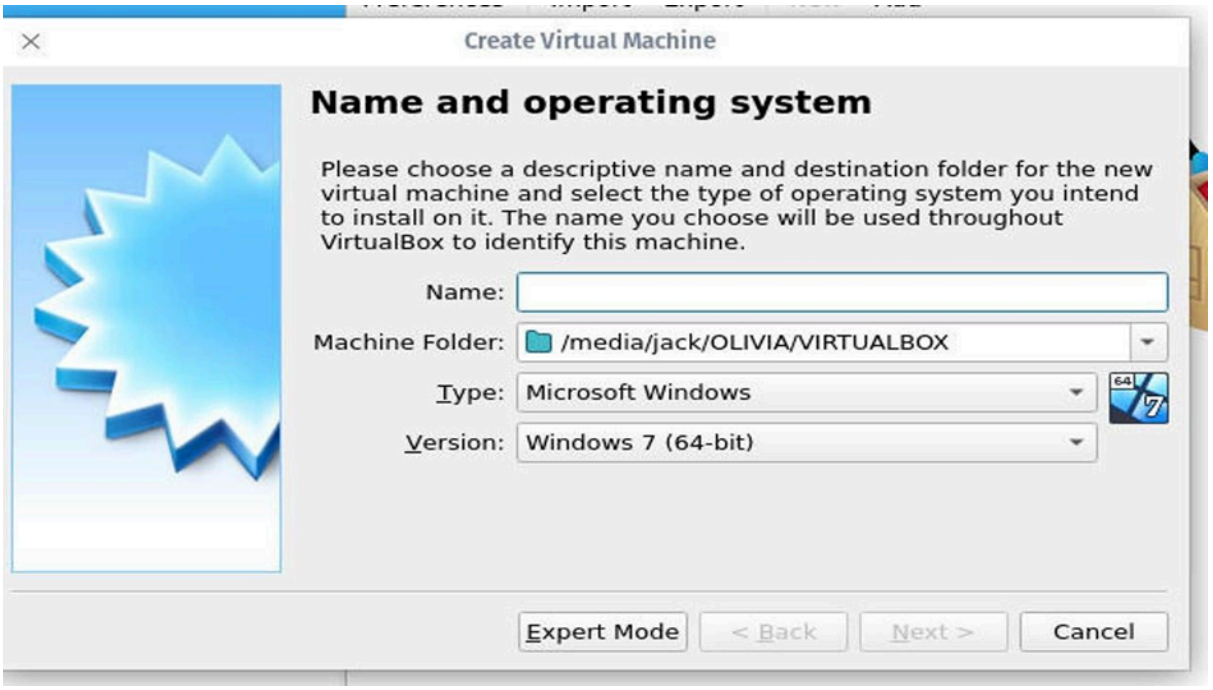
**Step 11:** Complete the installation, restart the virtual machine, and verify that the Linux OS is running properly.

**Step 12:** The virtual machine setup is now complete, providing an open-source Linux platform for further use.

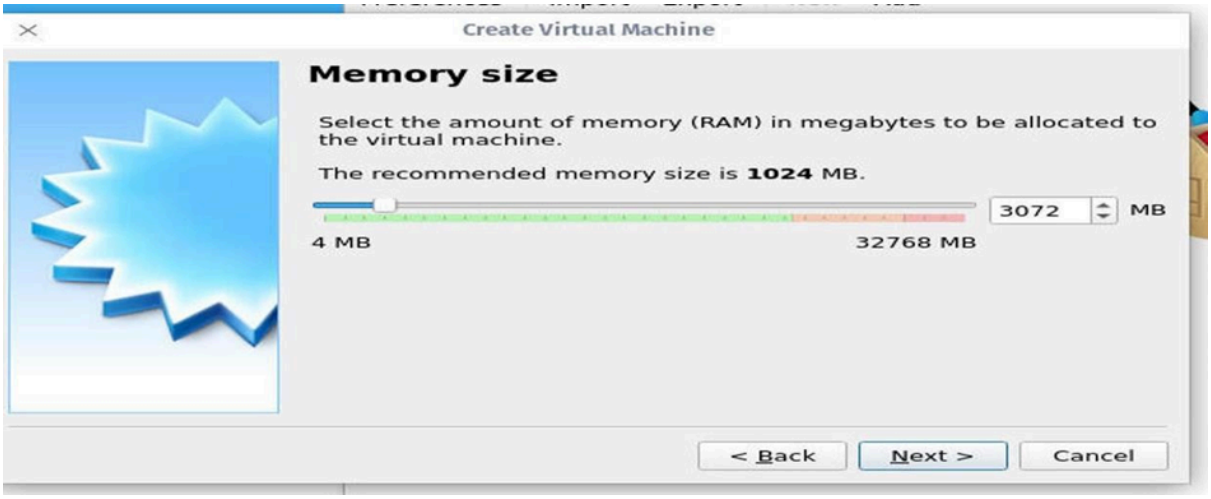
## PROGRAM IMPLEMENTATION:

- Open **VirtualBox** and create a new virtual machine.


- In the **Create Virtual Machine** window, enter a **name** for the VM, specify the **type** as Linux, and choose the correct **version** of your Linux distribution.

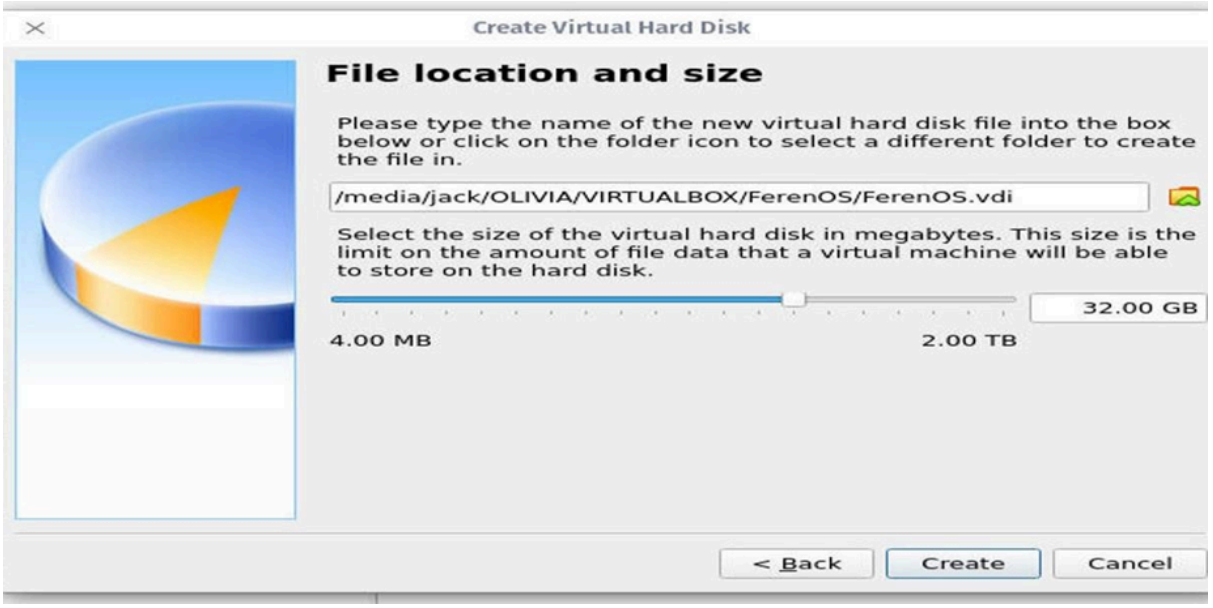


- Use the **Memory Size** slider to allocate the required RAM (recommended **2GB or more** for better performance), then click **Next**.

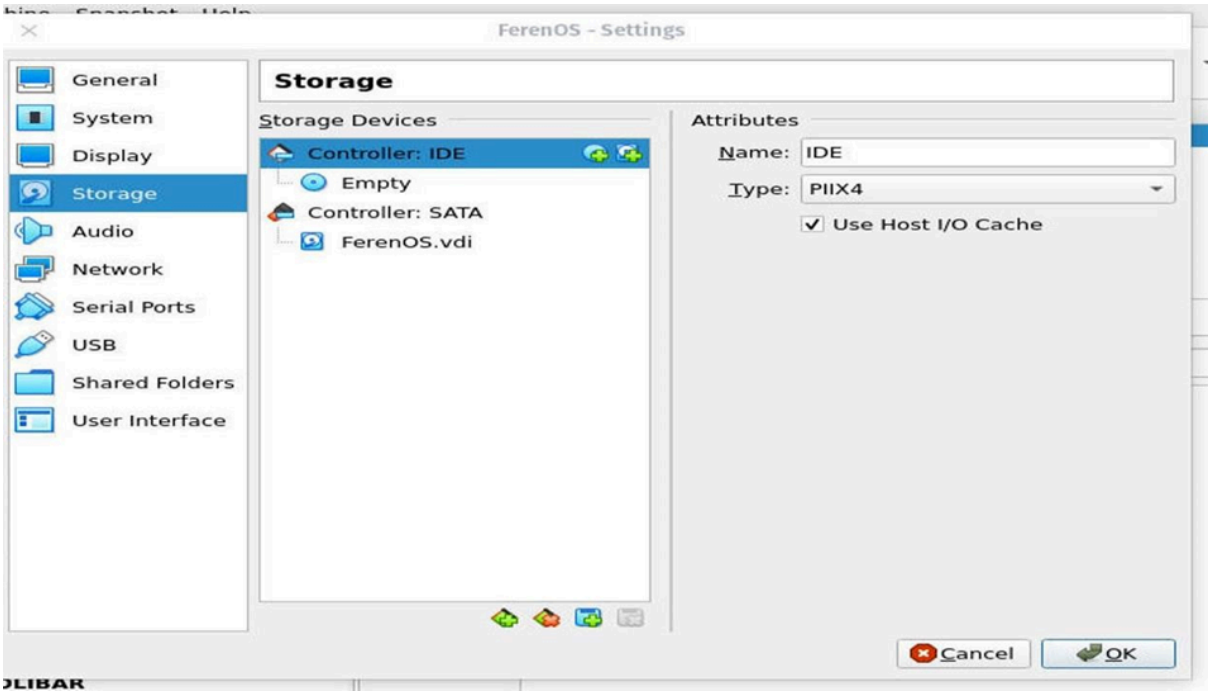



- Select **Create a virtual hard disk now** and click **Create** to proceed.

- Choose **VDI (VirtualBox Disk Image)** as the hard disk file type, then select **Dynamically Allocated** for flexible storage usage.
- Use the **slider** to allocate at least **20GB of disk space**, select the folder where the virtual hard disk file will be stored, then click **Create**.

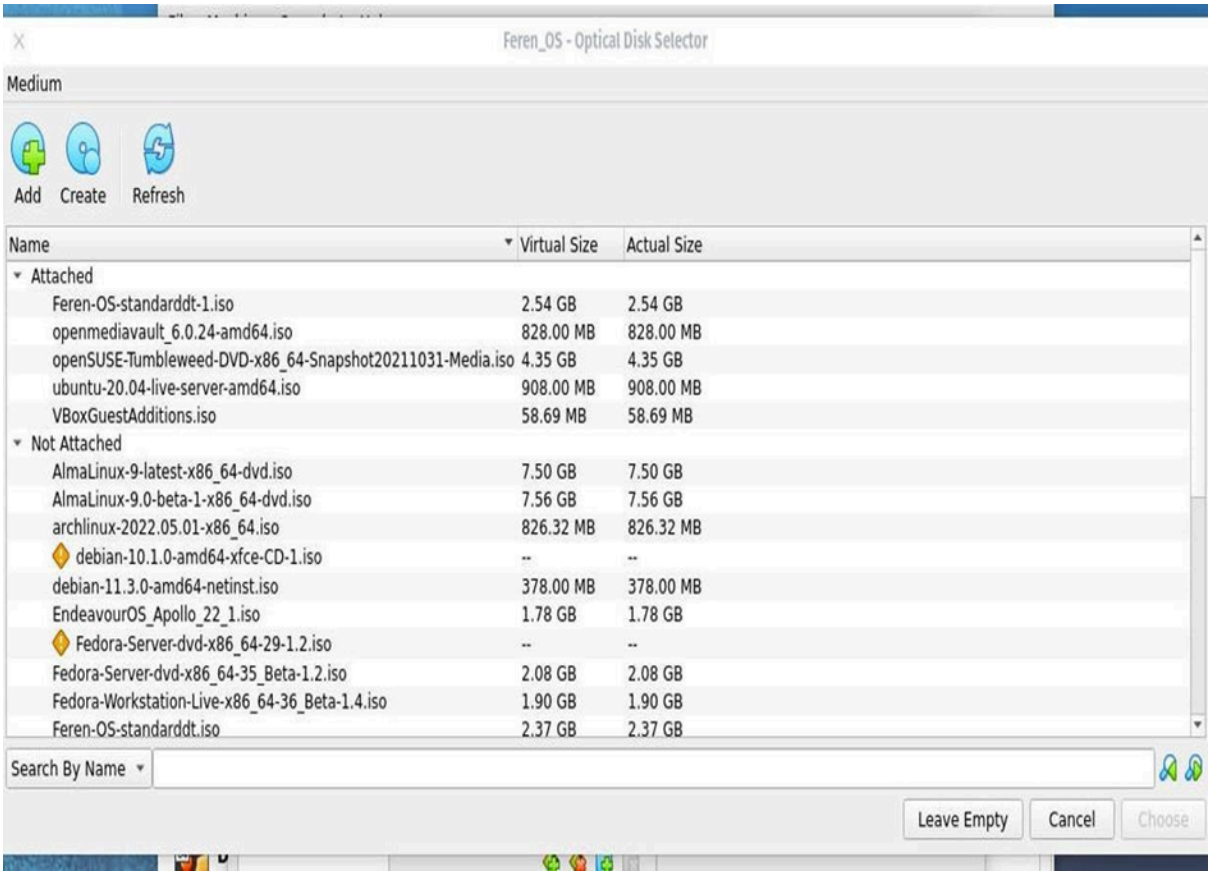



- Back in the **VirtualBox main window**, select the newly created virtual machine, click **Settings**, and go to the **Storage** tab. Click **Controller: IDE**, then select **Add Optical Drive** to mount the Linux **ISO image**.
- Click **Choose a disk file**, browse to the **ISO image file**, select it, and click **OK**.



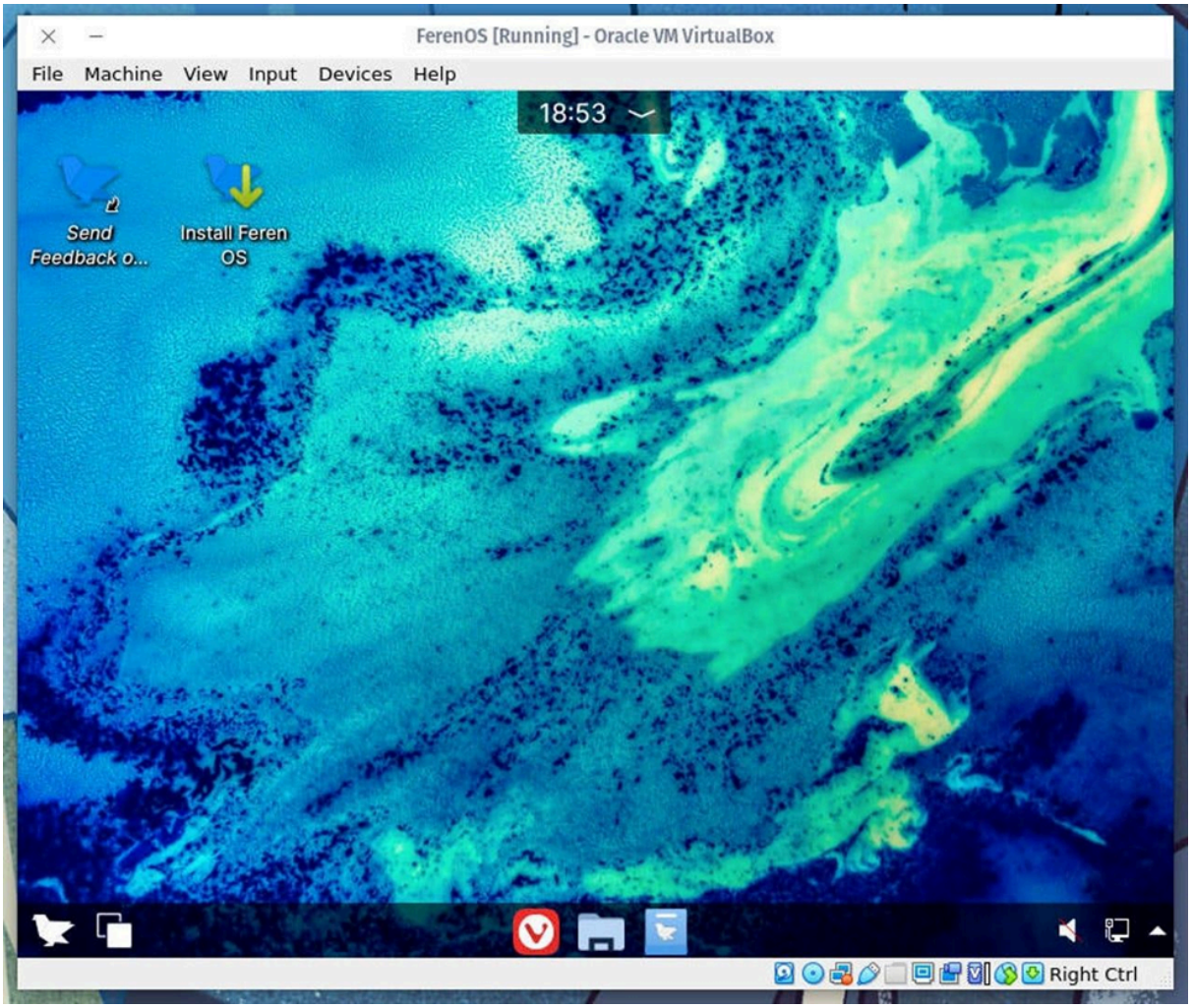
- Click **Start** to boot the virtual machine. The **Linux installer** will launch automatically.

- Follow the installation process by selecting the preferred **language**, **time zone**, **username**, and **password**.




- Click **Install** to proceed. Once installation is complete, restart the virtual machine and remove the **ISO image** from the virtual drive to prevent reinstallation.

- The Linux virtual machine is now successfully set up and ready for use.




**RESULT:**

Thus, the Linux virtual machine was successfully created, configured, and launched using VirtualBox.

| Ex.No:3 | **Hosting a Static Website Using Google Cloud Console** |
|---|---|
| | |

**AIM:**

To deploy and host a static website using Google Cloud Console and Cloud Storage.

**ALGORITHM:**

**Step 1:** Open **Google Cloud Console**, navigate to the project list, and create a **New Project** with a suitable name.

**Step 2:** In the **Navigation Menu**, go to **Cloud Storage → Buckets**, then click **Create** to set up a new storage bucket.

**Step 3:** Provide a **unique name** for the bucket, select **Multi-Region** as the location type, and choose **Standard Storage**.

**Step 4:** Uncheck **Enforce public access prevention**, leave other settings as default, and click **Create** to finalize the bucket setup.

**Step 5:** Upload the **website files** (e.g., index.html, styles.css, script.js) into the created bucket using the **Upload Files** option.

**Step 6:** Configure **permissions** by clicking on the **Permissions** tab, selecting **Grant Access**, and adding **"allUsers"** under **New Principals**. Assign the **Storage Object Viewer** role to allow public access.

**Step 7:** Click **Save**, then confirm by selecting **Allow Public Access** to make the website files readable by anyone.

**Step 8:** Add **index and error pages** by going to the **Edit Website Configuration** section in the bucket settings. Set index.html as the **Main Page** and error.html as the **Error Page**, then click **Save**.

**Step 9:** Verify the website is live by visiting:
https://storage.googleapis.com/YOUR_BUCKET_NAME/index.html

**Step 10:** If necessary, test by accessing incorrect URLs to check if the **error page** (error.html) is displayed properly.

**Step 11:** The static website is now successfully hosted on Google Cloud Storage and accessible via a public URL.

**PROGRAM:**

## Index.html: (Main Web Page)

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>Calculator</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <div class="calculator">

        <input type="text" id="display" readonly>

        <div class="buttons">

            <button onclick="appendToDisplay('1')">1</button>

            <button onclick="appendToDisplay('2')">2</button>

            <button onclick="appendToDisplay('3')">3</button>

            <button onclick="appendToDisplay('+')">+</button>

            <button onclick="appendToDisplay('4')">4</button>

            <button onclick="appendToDisplay('5')">5</button>

            <button onclick="appendToDisplay('6')">6</button>

            <button onclick="appendToDisplay('-')">-</button>

            <button onclick="appendToDisplay('7')">7</button>

            <button onclick="appendToDisplay('8')">8</button>
```

```
            <button onclick="appendToDisplay('9')">9</button>
            <button onclick="appendToDisplay('*')">*</button>
            <button onclick="appendToDisplay('0')">0</button>
            <button onclick="clearDisplay()">C</button>
            <button onclick="calculateResult()">=</button>
            <button onclick="appendToDisplay('/')">/</button>
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

**styles.css (Styling for the Calculator):**

```
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
}

.calculator {
    background: white;
    padding: 20px;
```

```
    border-radius: 10px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
}

.buttons {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    gap: 10px;
}

button {
    padding: 15px;
    font-size: 18px;
    border: none;
    background: #007bff;
    color: white;
    cursor: pointer;
    border-radius: 5px;
}

button:hover {
    background: #0056b3;
}
```

**script.js (Calculator Logic)**

```
function appendToDisplay(value) {
    document.getElementById("display").value += value;
}
function clearDisplay() {
    document.getElementById("display").value = "";
}
function calculateResult() {
    try {
        document.getElementById("display").value =
eval(document.getElementById("display").value);
    } catch (error) {
        alert("Invalid Expression");
    }
}
```

**error.html (Error Page for 404 Not Found):**

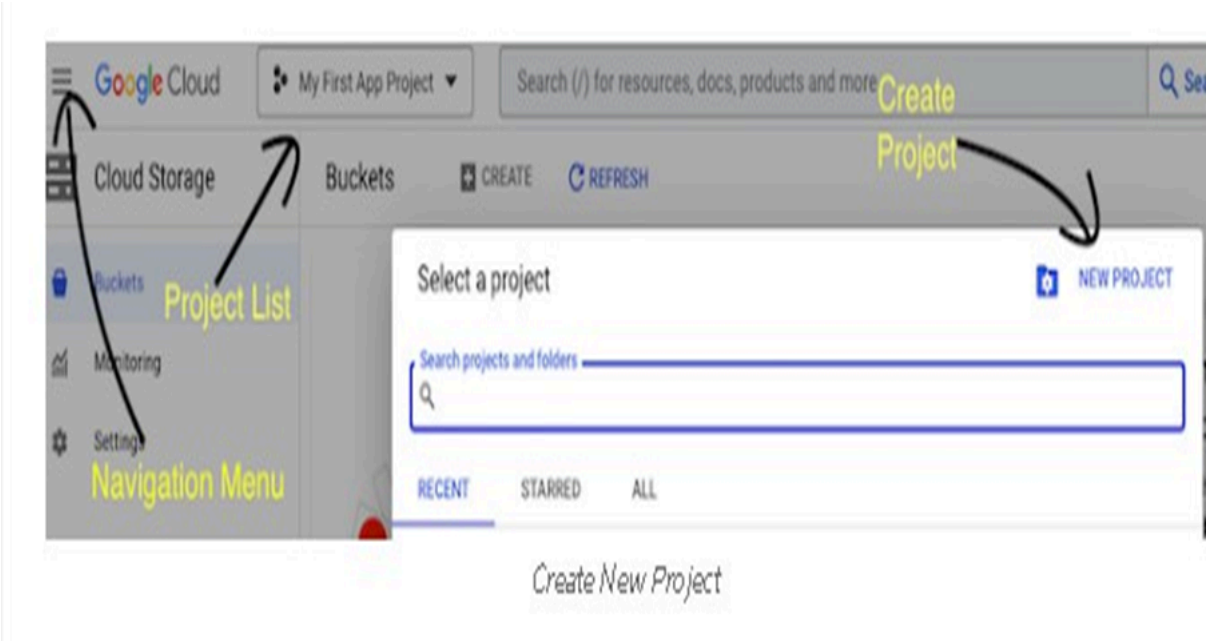
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Error Page</title>
    <style>
        body {
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    flex-direction: column;
    font-family: Arial, sans-serif;
}
h1 {
    font-size: 48px;
    color: red;
    margin-bottom: 20px;
}
p {
    font-size: 24px;
    margin-bottom: 30px;
}
a {
    font-size: 20px;
    text-decoration: none;
    color: #007bff;
}
a:hover {
    text-decoration: underline;
}
```

```
    </style>
</head>
<body>
    <h1>Oops! Something went wrong.</h1>
    <p>We're sorry, but an error occurred while processing your
request.</p>
    <a href="index.html">Back to Home Page</a>
</body>
</html>
```

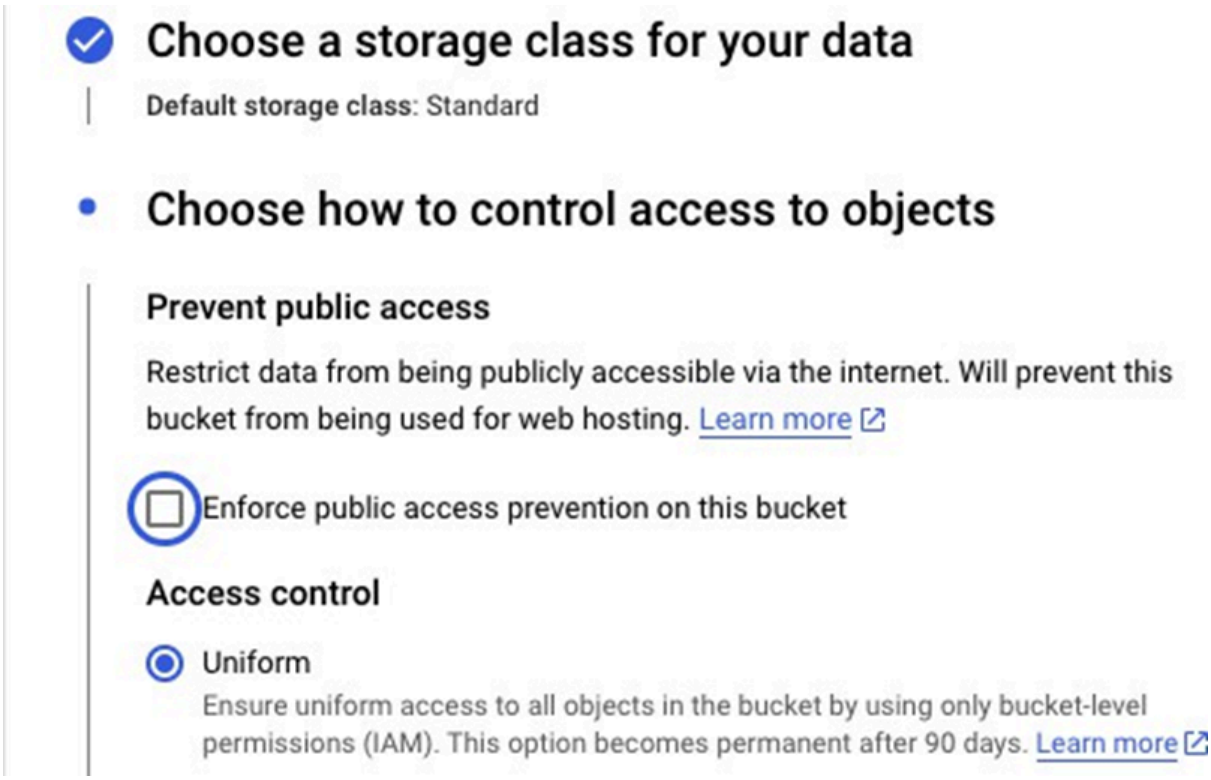
**DEPLOYMENT PROCESS:**

- Go to Google Cloud Console, create a New Project, and select it from the Project List.

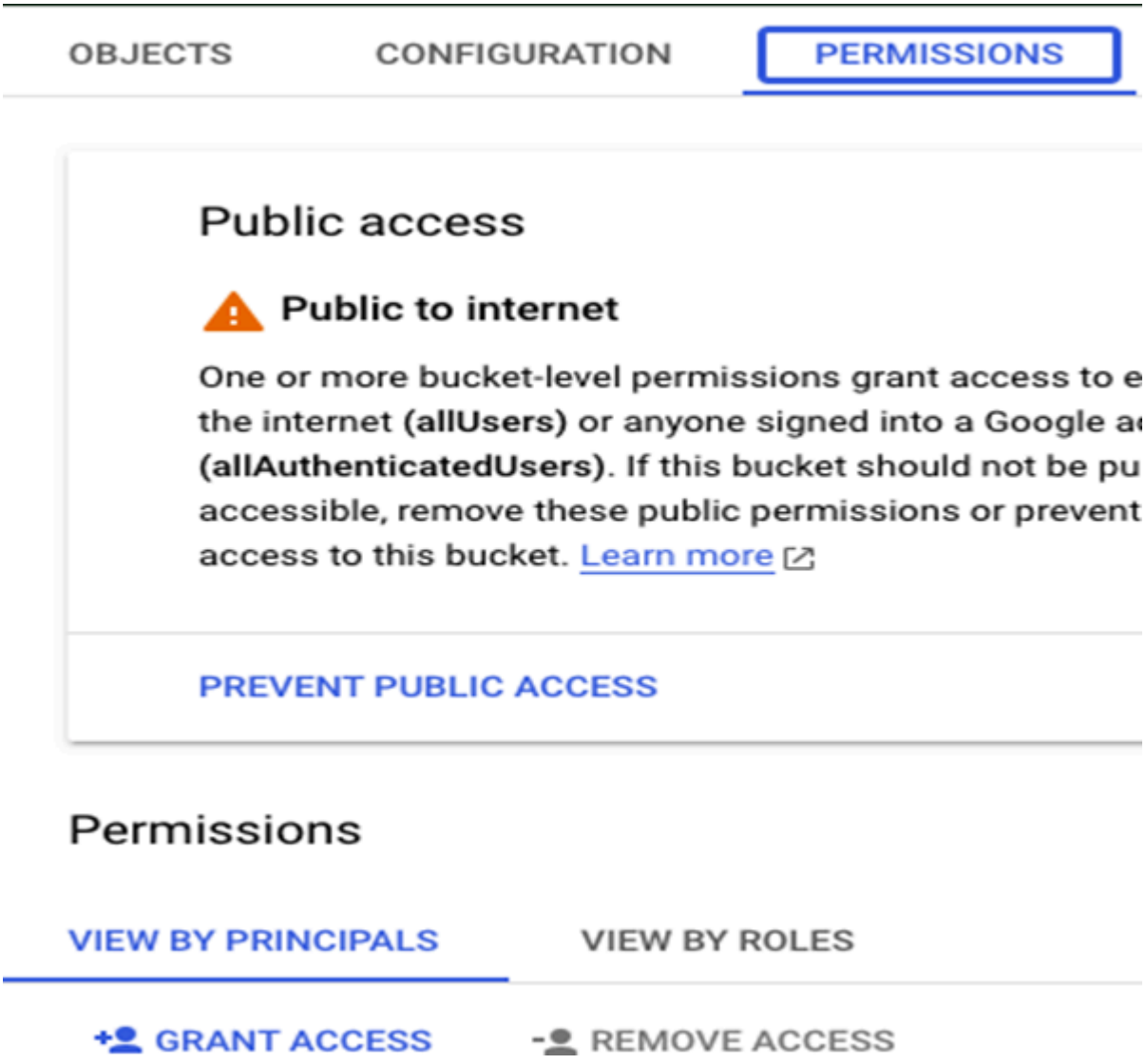



- **Navigate to Cloud Storage → Buckets** and click **Create** to set up a new **storage bucket**.

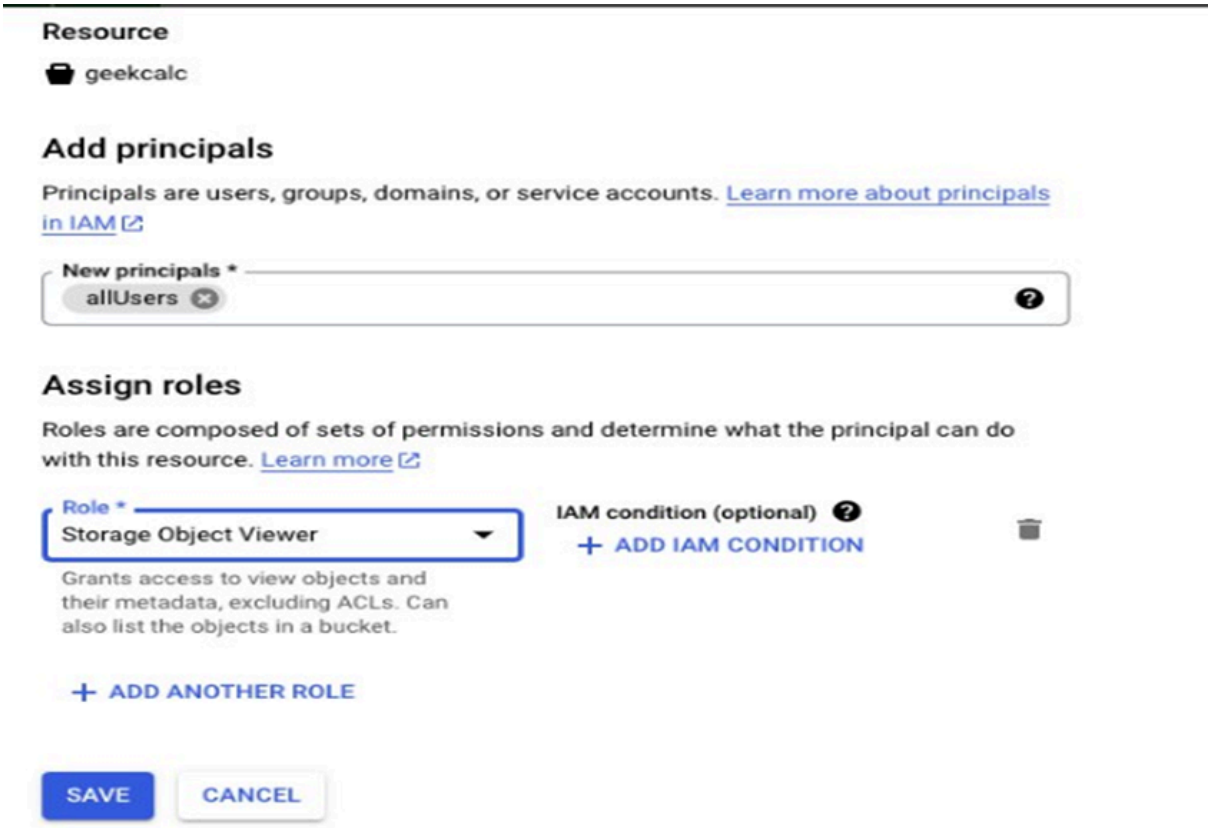

- **Provide a unique name**, select **Multi-Region**, and choose **Standard Storage**, then **Create** the bucket.




- **Upload the required files** (`index.html`, `styles.css`, `script.js`, `error.html`) to the bucket.
- **Configure permissions** in the **Permissions Tab**, add **"allUsers"**, and assign the **Storage Object Viewer** role.



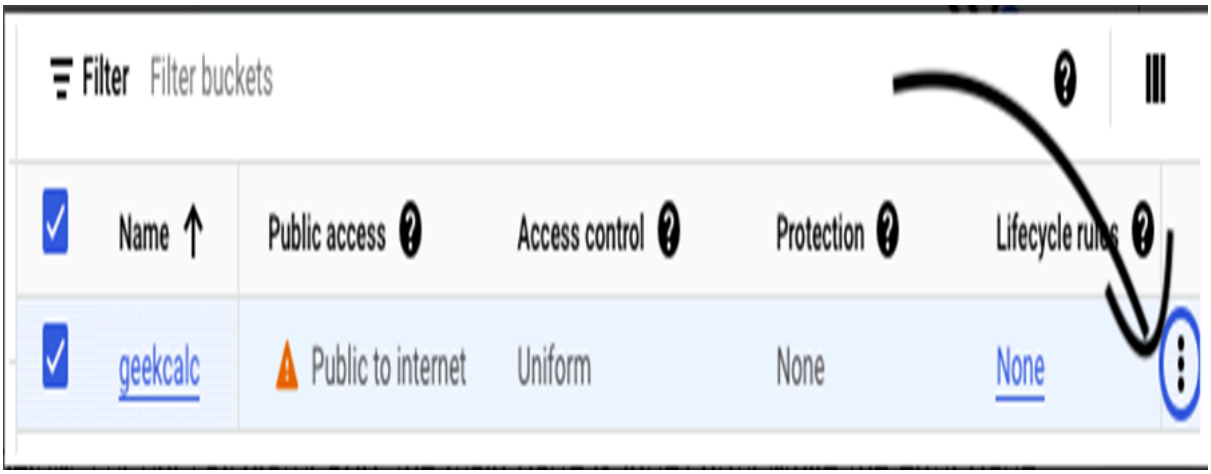

- **Click Save**, confirm **Allow Public Access**, and ensure the files are publicly rea0dable.



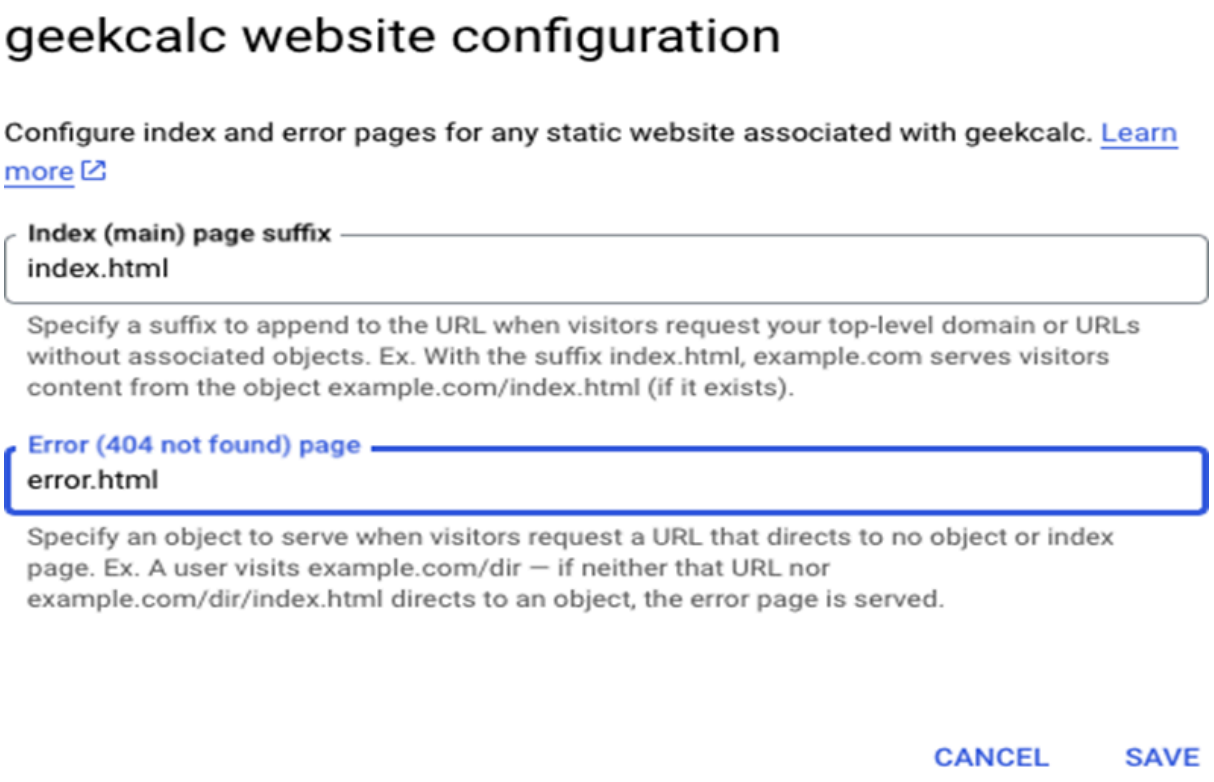

- **Access the website** using the following URL format:

  https://storage.googleapis.com/YOUR_BUCKET_NAME/index.html

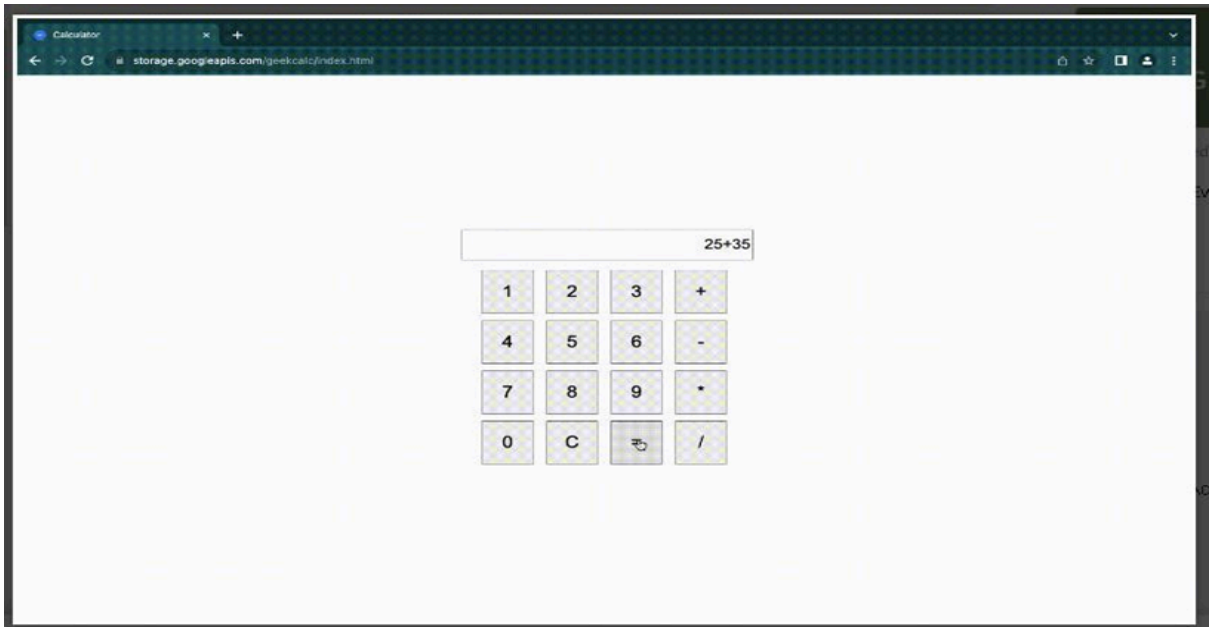
- Now, go to Cloud Storage -> Bucket, click on the Bucket menu button as shown below and select Edit website configuration.



- In the website configuration, specify the main page and error page as shown below. For our calculator app, the main page is index.html while the error page is error.html. Click the save button.
- Verify that the website is live and test the error page by entering a wrong URL.




- **The static website is now successfully hosted and accessible to the public**.




**RESULT:**

Thus, the static website was successfully created, deployed, and hosted on **Google Cloud Storage**.

| Ex.No:4 | **A) Exploring Google Cloud for Storage** |
|---|---|
| | |

**AIM:**

To utilize **Google Cloud Storage** for data storage, backup, security, and lifecycle management.

**ALGORITHM:**

**Step 1:** Start by logging into **Google Cloud Console** and navigating to the **Cloud Storage** section.

**Step 2:** Click on **Create Bucket**, assign a unique name, choose the appropriate **storage class**, and select a **location** for storing data.

**Step 3:** Upload data to the **Cloud Storage Bucket** using either the **Google Cloud Console** (by clicking "Upload Files") or the **gsutil command-line tool**.

**Step 4:** Back up data to the Cloud Storage bucket using either the **Cloud Storage Transfer Service** or the **gsutil tool**, ensuring secure storage.

**Step 5:** Secure the data by setting up **IAM policies** or **Access Control Lists (ACLs)** to manage user permissions and access.

**Step 6:** Enable **object versioning** to maintain multiple versions of files in case of accidental deletions or updates.

**Step 7:** Implement **lifecycle management policies** to automatically transition data between **storage classes** or delete outdated files after a specified period.

**Step 8:** Use **Coldline or Nearline Storage** for infrequently accessed data to reduce storage costs.

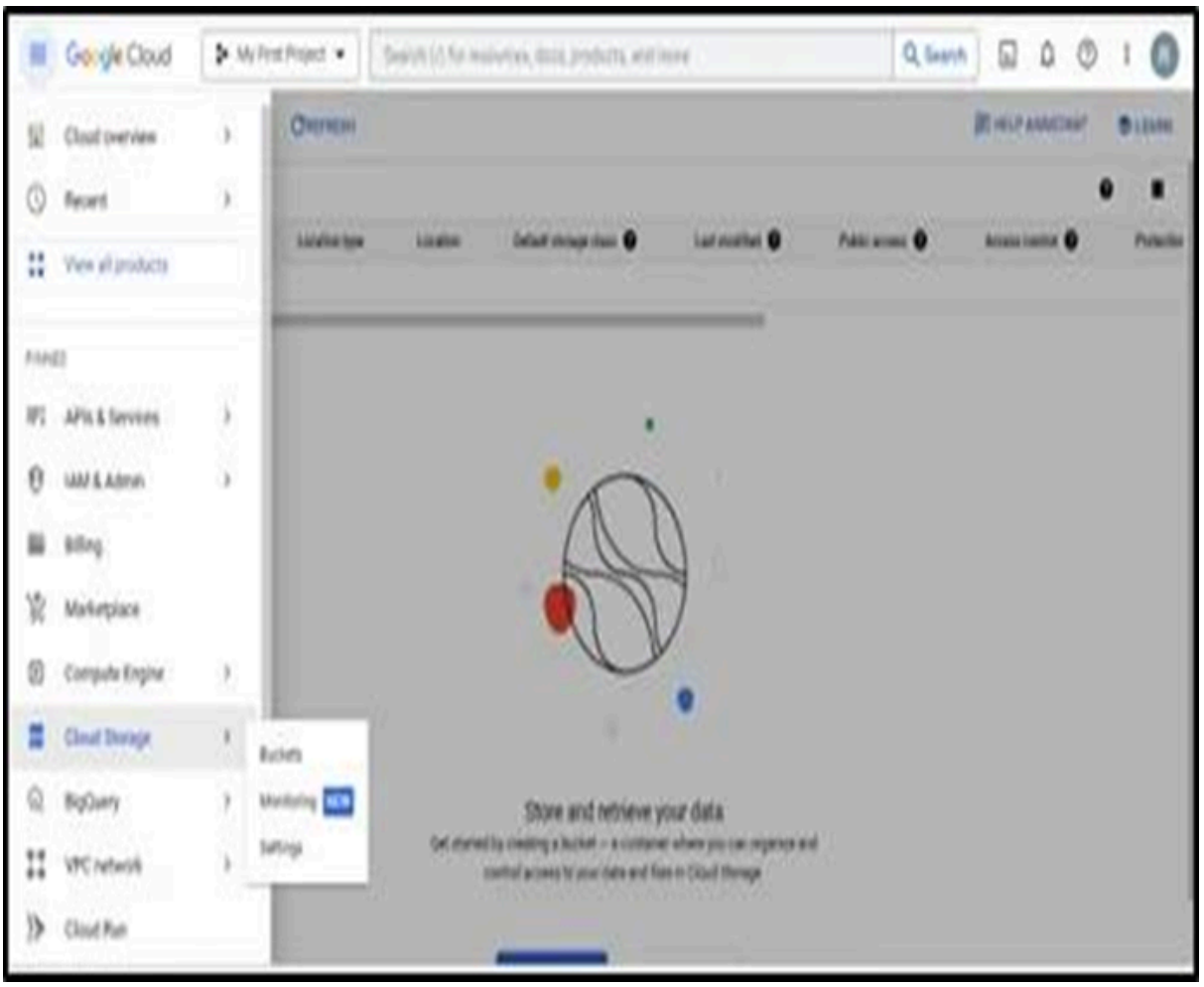
**Step 9:** Monitor and analyze storage usage using **Google Cloud's monitoring tools** to optimize resource utilization and detect anomalies.

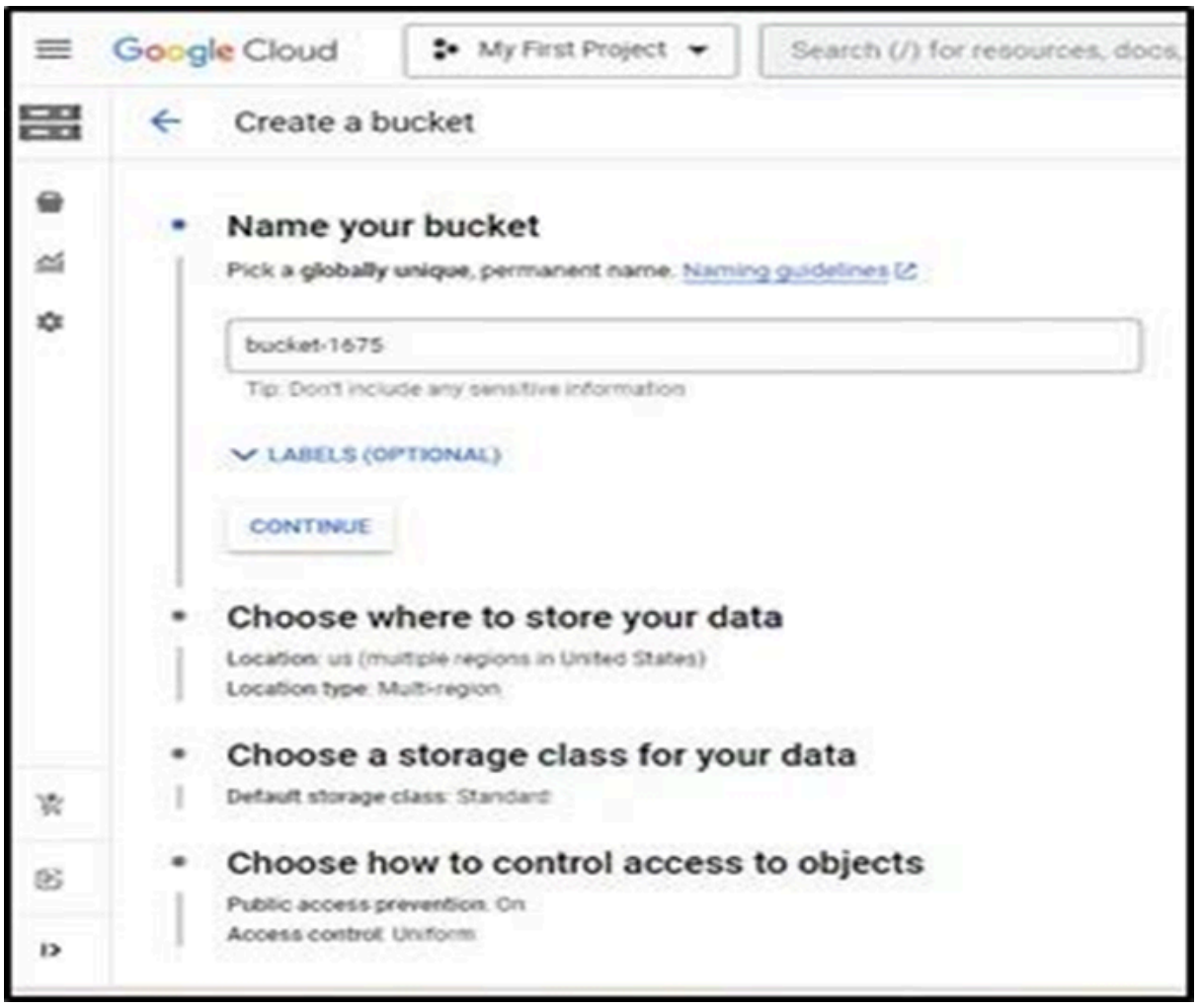
**Step 10:** The process is successfully completed, ensuring an efficient and secure storage solution.

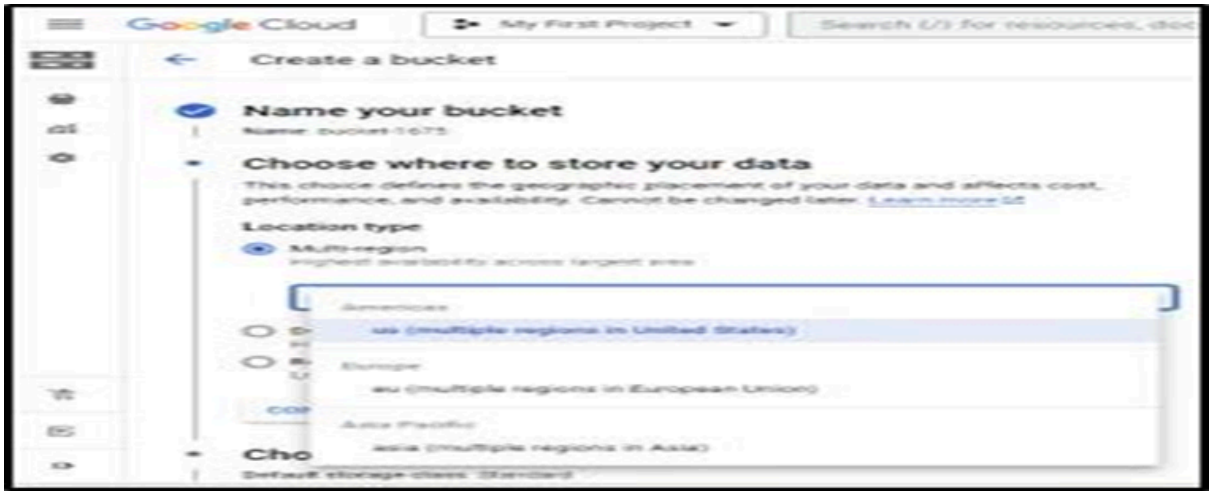
**PROCEDURE:**

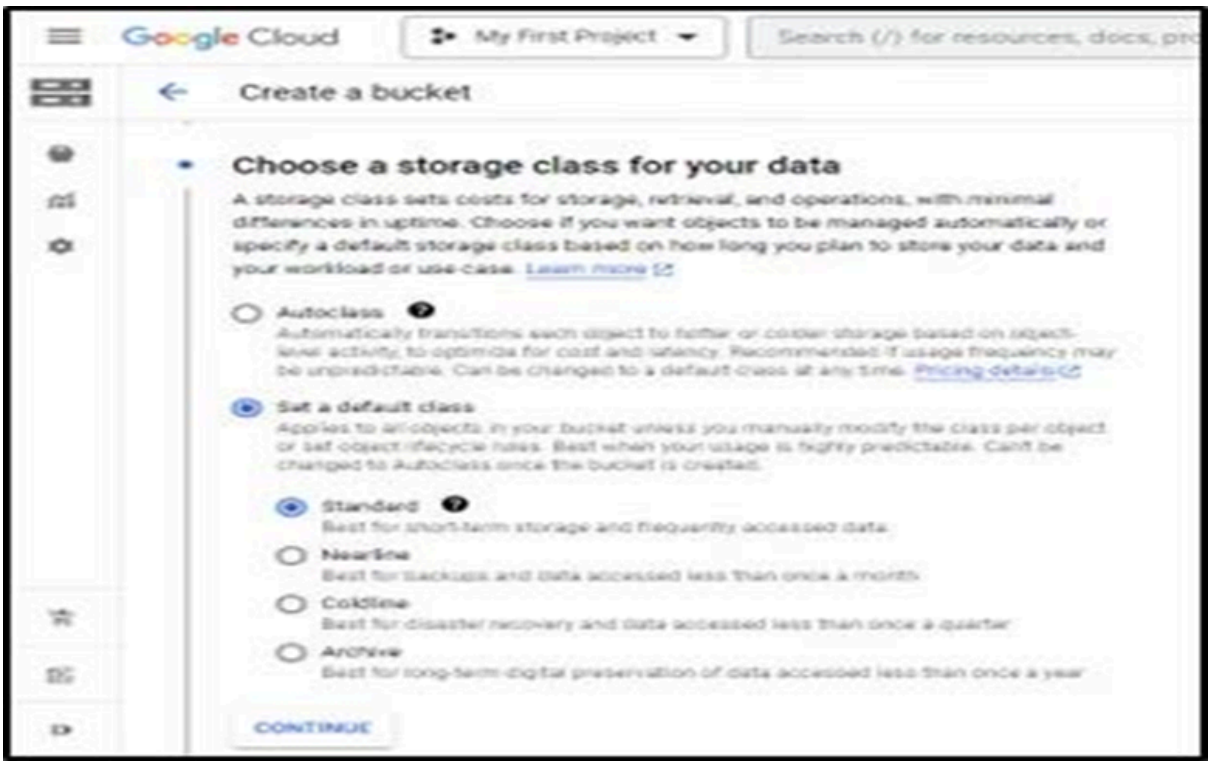
**1. Creating a Google Cloud Storage Bucket**

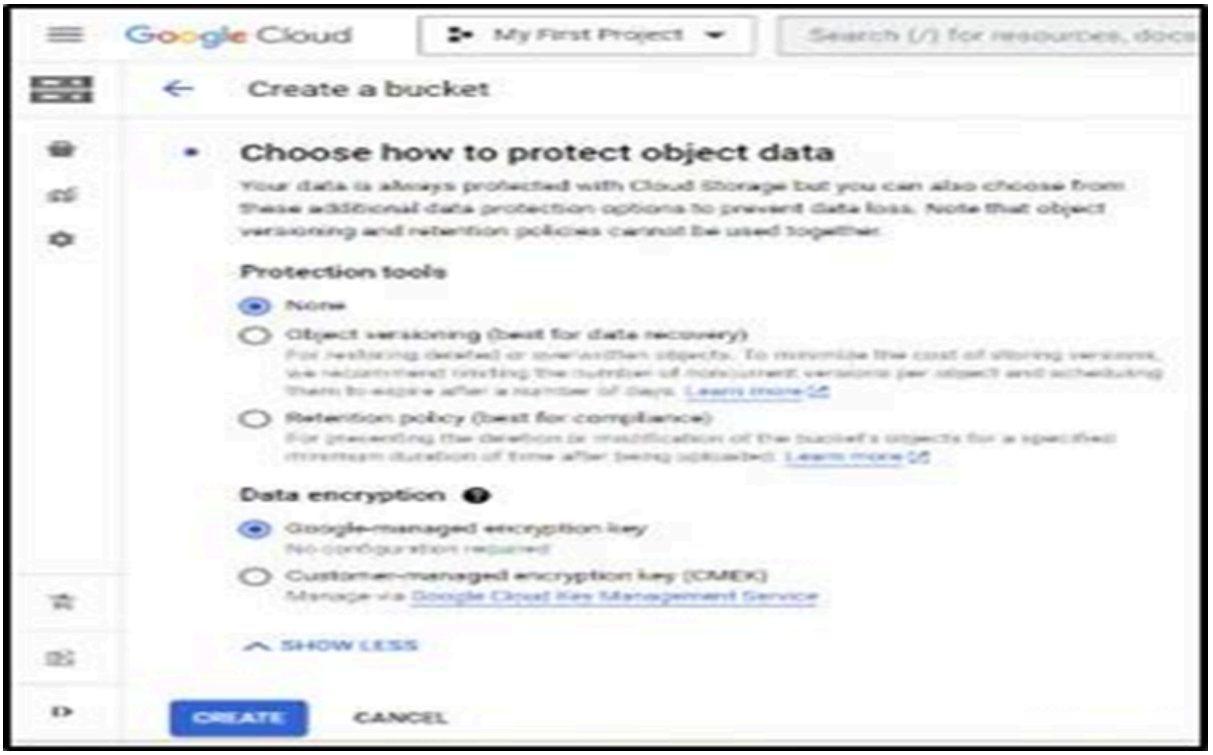
- Log in to **Google Cloud Console** and navigate to the **Cloud Storage** section.
- Click on **Create Bucket** and enter a **unique name**.
- Choose the **storage class** (Standard, Nearline, Coldline) based on data access frequency.
- Select the **bucket location** and click **Create**.




Google Cloud
My First Project
Search (/) for resources, docs,
Create a bucket
Name your bucket
Pick a globally unique, permanent name. Naming guidelines
bucket-1675
Tip: Don't include any sensitive information
LABELS (OPTIONAL)
CONTINUE
Choose where to store your data
Location: us (multiple regions in United States)
Location type: Multi-region
Choose a storage class for your data
Default storage class: Standard
Choose how to control access to objects
Public access prevention: On
Access control: Uniform


Google Cloud
Create a bucket
Name your bucket
Choose where to store your data
Location type
Multi-region




**2. Uploading Data to the Cloud Storage Bucket**

- Using the **Google Cloud Console**, navigate to your bucket and click **Upload Files**.

Using **gsutil command-line tool**, install and configure gsutil, then run:
`gsutil cp myfile.txt gs://[Bucket_Name]/`

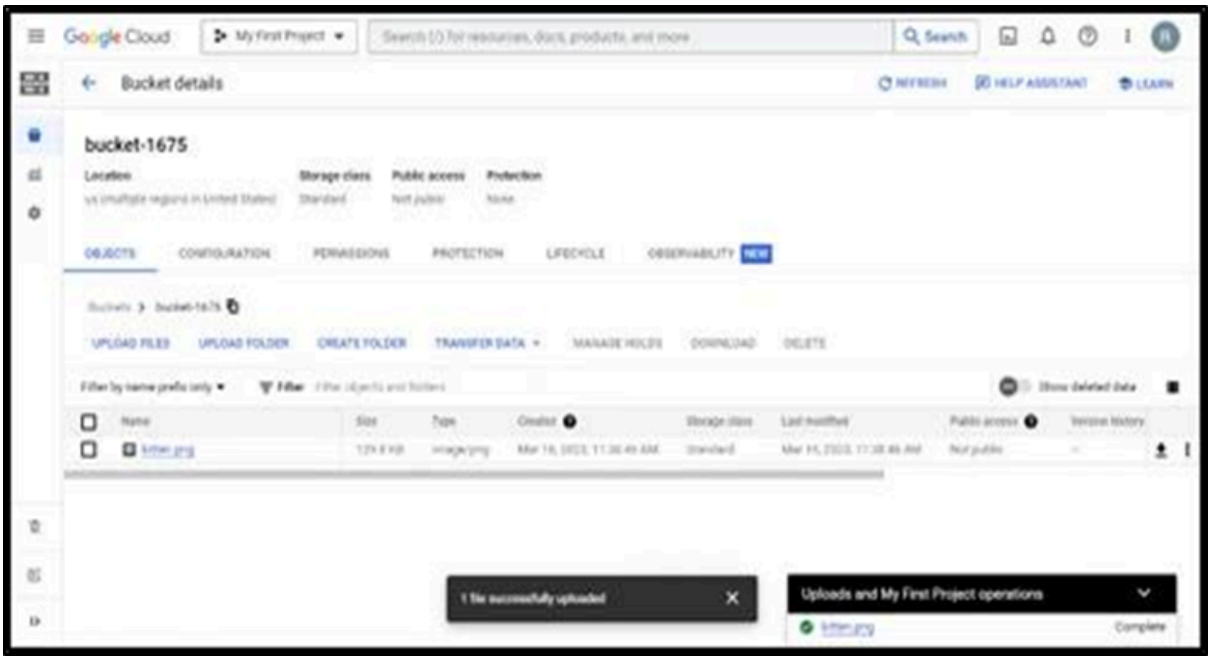
- This command uploads myfile.txt to the specified Cloud Storage bucket.

**3. Backing Up Data**

- Use **Cloud Storage Transfer Service** to automate data transfers from on-premises systems or between Cloud Storage buckets.

Use **gsutil rsync** to synchronize local directories with Cloud Storage:
`gsutil rsync -r /local_directory gs://[Bucket_Name]/`

- This ensures that any updates made in the local directory are reflected in the Cloud Storage bucket.




**4. Securing Data in Cloud Storage**

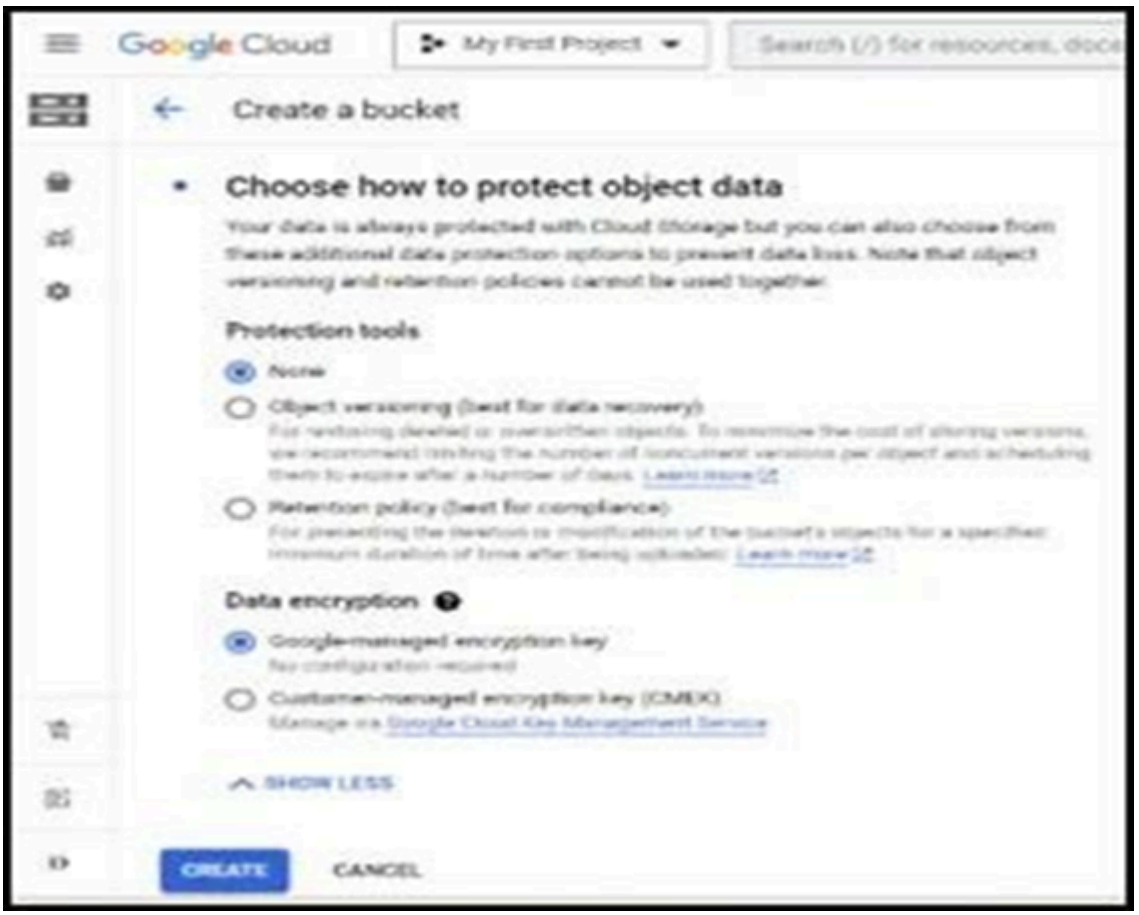
- By default, all objects in Cloud Storage are private. Grant access using **IAM policies** or **ACLs** to specific users or groups.
- Enable **object versioning** to store multiple versions of files, which is useful for recovery in case of accidental deletion.

**5. Implementing Lifecycle Management**

- Google Cloud Storage allows users to set up **lifecycle rules** that automatically transition data to **Nearline or Coldline Storage** or delete files after a specific period.

Example lifecycle rule (move objects to Coldline Storage after 30 days):

```
{
  "rule": [{
    "action": { "type": "SetStorageClass", "storageClass":
"COLDLINE" },
    "condition": { "age": 30 }
  }]
}
```




**6. Using Cost-Effective Storage Classes**

- **Standard Storage** for frequently accessed data.
- **Nearline Storage** for data accessed once a month.
- **Coldline Storage** for long-term archival storage (rarely accessed).

**7. Monitoring and Analyzing Data Usage**

- Use **Google Cloud Monitoring** to track **storage usage**, **data access patterns**, and detect anomalies.
- Set up **billing alerts** to optimize storage costs based on usage patterns.

**RESULT:**

Thus, the process of utilizing **Google Cloud Storage** for **data storage, backup, security, and lifecycle management** was successfully implemented.

| Ex.No:4 | B) Exploring Google Cloud for Data Sharing |
|---|---|
| | |

**AIM:**

To configure **Google Cloud Storage** for **data sharing** and manage **data access permissions** efficiently.

**ALGORITHM:**

**Step 1:** Start the process by signing in to the **Google Admin Console** with a **super administrator** account.

**Step 2:** Navigate to **Menu → Account → Account Settings → Legal and Compliance** to access sharing options.

**Step 3:** Click on **Sharing Options**, and choose **Enabled** to allow data sharing across Google Cloud services.

**Step 4:** If data sharing needs to be disabled, select **Disabled** to prevent any new data from being shared.

**Step 5:** When sharing is disabled, previously shared data will be deleted as per the **Google Cloud Admin Activity Audit Log Retention Policy**.

**Step 6:** Click **Save** to apply changes and update the data-sharing settings.

**Step 7:** Ensure that **super administrators** have access to all features in the Google Admin Console and Admin API.

**Step 8:** Super admins can manage **user roles**, **password resets**, **file ownership transfers**, and enforce security measures like **2-Step Verification**.

**Step 9:** Admins can also **assign roles, manage API access, configure SAML apps**, and monitor **Google Calendar resource permissions**.

**Step 10:** Confirm that at least **one super administrator** is available to receive billing and account notifications.

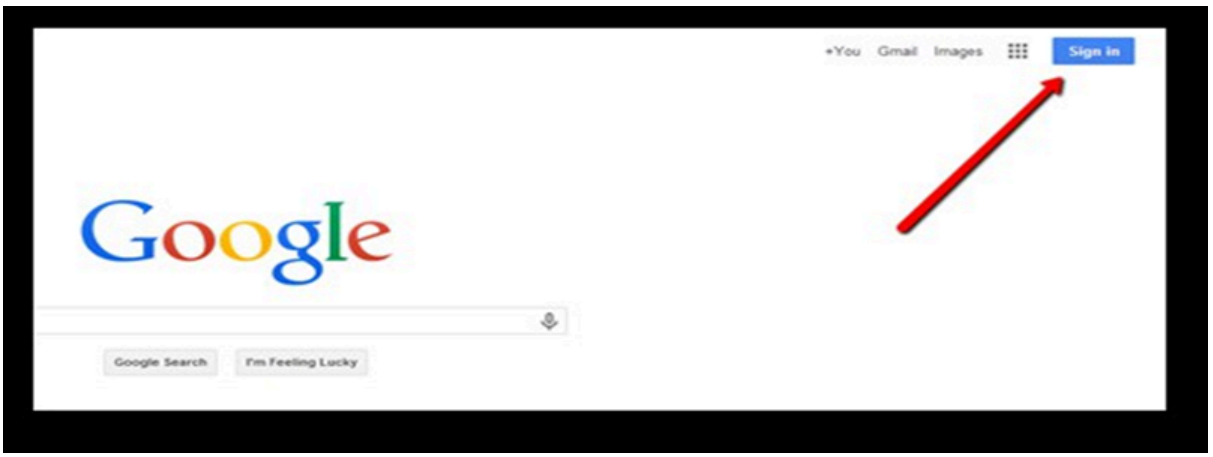
**Step 11:** Ensure that **super admins can reset their passwords** to maintain continuous access to critical services.

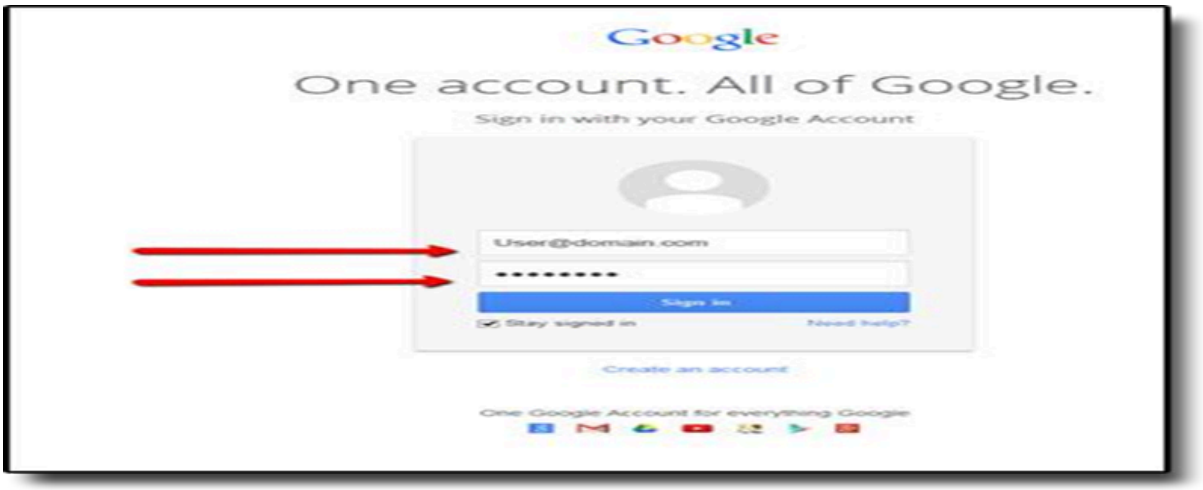
**Step 12:** Stop the process after verifying that **data-sharing policies** have been successfully Updated.

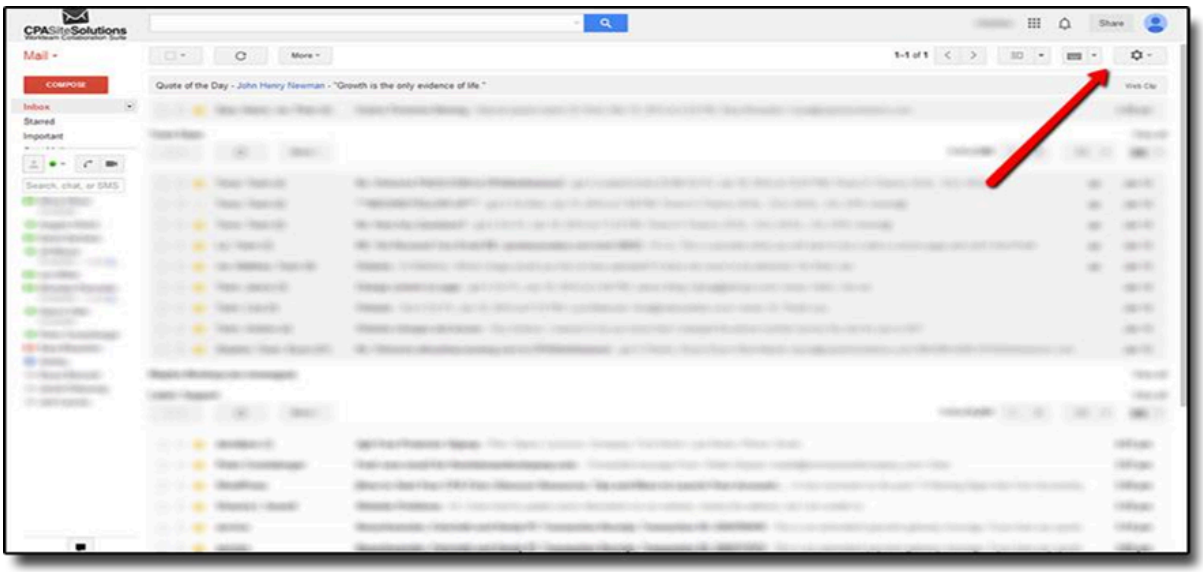
**PROCEDURE:**

**1. Signing into the Google Admin Console**

- Open a web browser and go to **admin.google.com**.
- Sign in using a **Google Admin Account** with **super administrator privileges** (not ending in @gmail.com).
- If you forget your password, follow the password reset process.
- Ensure that only authorized admins have access to manage organizational services.

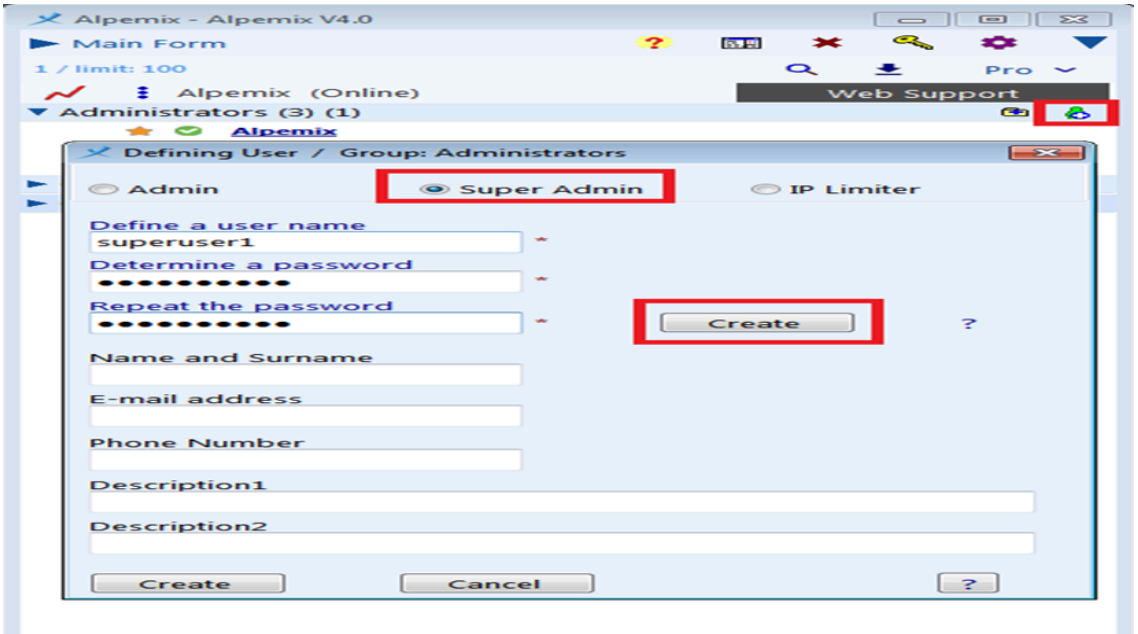


## 2. Accessing Sharing Settings

- In the **Google Admin Console**, navigate to **Menu → Account → Account Settings → Legal and Compliance**.
- This section provides **full control** over organizational data sharing policies.

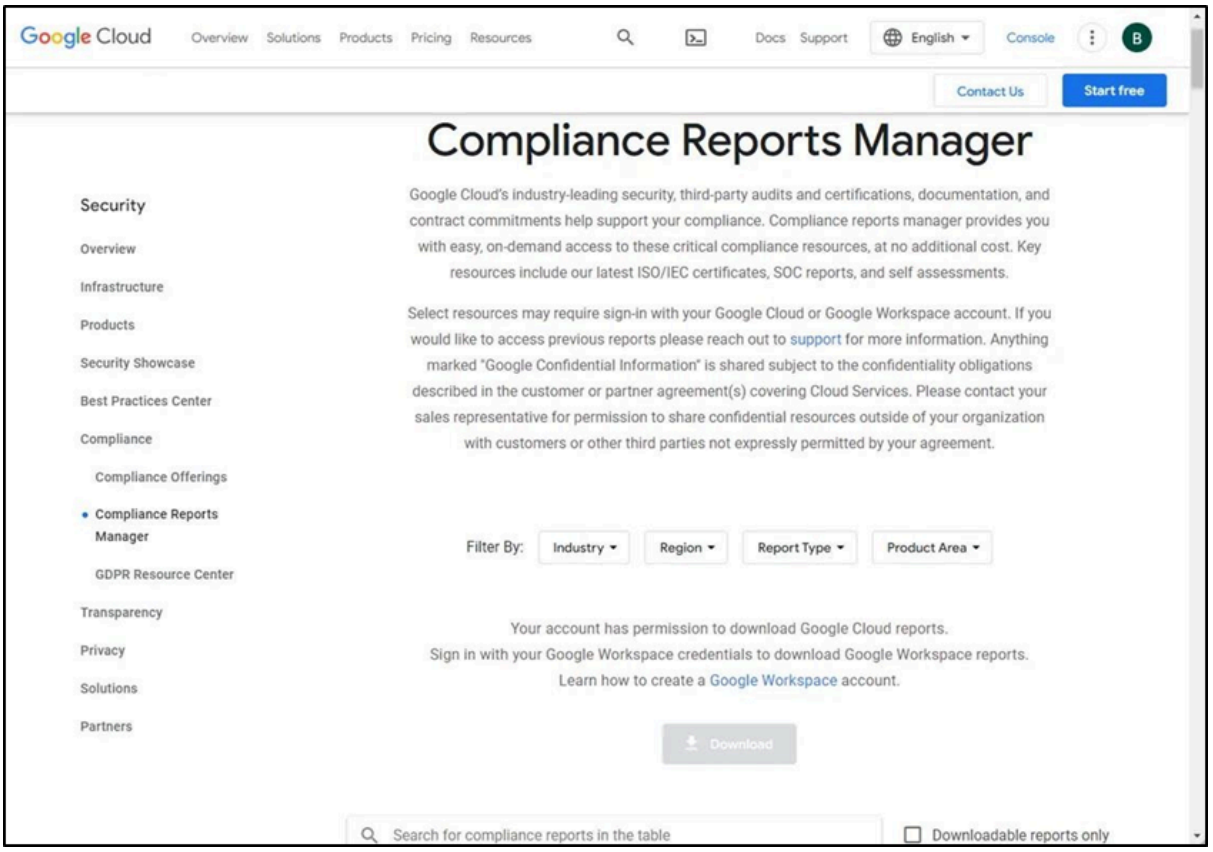
## 3. Managing Super Administrator Permissions

- Super Admins can:
  - Assign **administrator roles** and manage other admin privileges.
  - Reset passwords and manage **user access control**.
  - Transfer ownership of files before **deleting users**.
  - Enable **2-Step Verification** for security.
  - Install and manage **Google Workspace Marketplace Apps**.
  - Configure **Google Calendar resource access levels**.
  - Set up **Google as a SAML identity provider** for authentication.



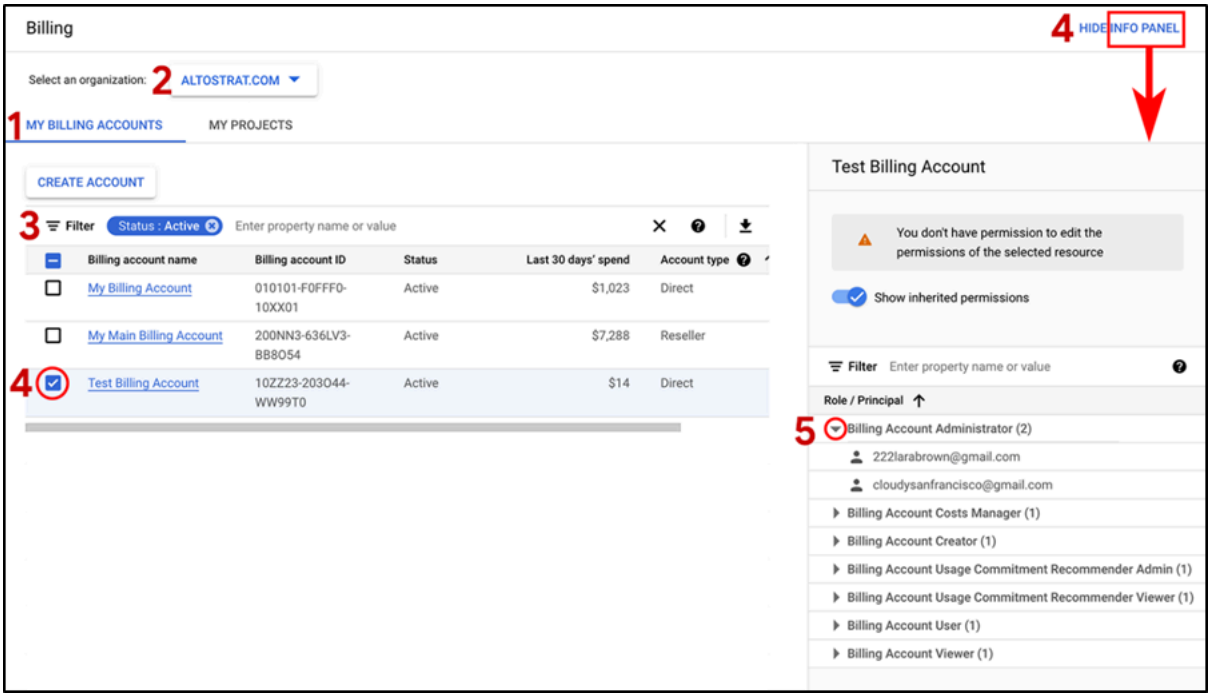
### 4. Configuring Data Sharing

- Click on **Sharing Options** in the Admin Console.
- To **enable data sharing**, select **Enabled** and save changes.
- To **disable data sharing**, select **Disabled** to **prevent** any new data from being shared.




### 5. Handling Existing Shared Data

- Once sharing is **disabled**, any previously shared data will be **deleted** as per the **Google Cloud Admin Activity Audit Log Retention Policy**.
- Admins should ensure that necessary data is backed up before disabling sharing.




### 6. Saving and Verifying Changes

- Click **Save** to confirm the updates in data-sharing settings.
- Verify that all applied policies are active and correctly implemented.

**RESULT:**

Thus, the **Google Cloud Storage data-sharing settings** were successfully configured and applied.

| Ex.No:4 | C) Exploring Google Cloud for Managing Calendar and To-Do Lists |
|---|---|
| | |

**AIM:**

To manage Google Calendar and To-Do Lists using **Google Cloud APIs** for scheduling and task automation.

**ALGORITHM:**

**Step 1:** Start the process by setting up a **Google Cloud Project** and enabling **billing** if required.

**Step 2:** Navigate to **API & Services → Library**, and enable **Google Calendar API** and **Google Tasks API**.

**Step 3:** Configure **OAuth 2.0 credentials** by selecting "Web Application," setting up consent screen details, and obtaining **Client ID and Client Secret**.

**Step 4:** Use **Google Calendar API** to fetch, create, update, and delete events.

**Step 5:** Write and execute **Python code** using the **Google API client library** to list upcoming calendar events.

**Step 6:** Schedule tasks by inserting new calendar events using the **insert()** method and modifying them with **update()**.

**Step 7:** Interact with **Google Tasks API** to retrieve, add, or update tasks using HTTP requests or API methods.

**Step 8:** Implement **Python code** using the **Google Tasks API** to list and manage tasks programmatically.

**Step 9:** Set up **Google Cloud Functions** to automate interactions with Calendar and Tasks APIs.

**Step 10:** Deploy and test a **Node.js-based Google Cloud Function** that retrieves calendar events dynamically.

**Step 11:** Debug and ensure that the cloud function is running as expected with appropriate API triggers.

**Step 12:** Stop the process after verifying successful task scheduling and event management.

**PROCEDURE:**

**1. Setting Up Google Cloud Project**

- Go to the **Google Cloud Console** and create a **new project**.
- Navigate to **API & Services → Library** and enable **Google Calendar API** and **Google Tasks API**.

**2. Configuring OAuth 2.0 Credentials**

- In **API & Services**, go to **Credentials** and select **Create Credentials → OAuth 2.0 Client ID**.
- Choose **Web Application**, enter authorized redirect URIs, and save **Client ID & Secret**.

**3. Interacting with Google Calendar API**

- Use the **Google API client library** to interact with Calendar events.
- Fetch calendar events using **Python** and display upcoming events.

```
from google.oauth2 import service_account
from googleapiclient.discovery import build

SCOPES = ['https://www.googleapis.com/auth/calendar']
SERVICE_ACCOUNT_FILE = 'path/to/service.json'

creds =
service_account.Credentials.from_service_account_file(SERVICE_ACCOUN
T_FILE, scopes=SCOPES)
service = build('calendar', 'v3', credentials=creds)

events_result = service.events().list(calendarId='primary',
maxResults=10, singleEvents=True, orderBy='startTime').execute()
events = events_result.get('items', [])

if not events:
    print('No upcoming events found.')

for event in events:
    start = event['start'].get('dateTime',
event['start'].get('date'))
    print(start, event['summary'])
```

**4. Scheduling and Managing Tasks**

- Create, update, and delete tasks in **Google Tasks API**.
- Execute Python code to fetch the task list.

```
from google.oauth2 import service_account
from googleapiclient.discovery import build

SCOPES = ['https://www.googleapis.com/auth/tasks']
SERVICE_ACCOUNT_FILE = 'path/to/service.json'

creds =
service_account.Credentials.from_service_account_file(SERVICE_ACCOUN
T_FILE, scopes=SCOPES)
service = build('tasks', 'v1', credentials=creds)

tasklists = service.tasklists().list().execute()
tasklist_id = tasklists['items'][0]['id']
tasks = service.tasks().list(tasklist=tasklist_id).execute()

for task in tasks.get('items', []):
    print(task['title'])
```

**5. Automating Tasks Using Google Cloud Functions**

- Go to **Cloud Functions** in **Google Cloud Console**.
- Click **Create Function**, set the trigger type to **HTTP**, and deploy the function.

```
const { google } = require('googleapis');

exports.manageCalendar = async (req, res) => {
    const calendar = google.calendar({ version: 'v3', auth:
'YOUR_API_KEY' });
    const response = await calendar.events.list({
        calendarId: 'primary',
        maxResults: 10,
        singleEvents: true,
        orderBy: 'startTime',
    });

    res.status(200).send(response.data.items);
};
```

**6. Deploying and Debugging Cloud Functions**

- Deploy the function using **Google Cloud Console** or **Cloud Shell**.
- Verify responses and ensure proper event listing.

**Output:**

```csharp
Complete quarterly report
Prepare presentation for client
Follow up on invoices
```


```
[
  {
    "kind": "calendar#event",
    "etag": "\"1234567890\"",
    "id": "event-id-1",
    "status": "confirmed",
    "htmlLink": "https://www.google.com/calendar/event?eid=event-id-1",
    "created": "2024-07-15T12:00:00Z",
    "updated": "2024-07-15T12:30:00Z",
    "summary": "Project Kickoff Meeting",
    "description": "Kickoff meeting for the new project",
    "start": {
      "dateTime": "2024-08-01T09:00:00-07:00",
      "timeZone": "America/Los_Angeles"
    },
    "end": {
      "dateTime": "2024-08-01T10:00:00-07:00",
      "timeZone": "America/Los_Angeles"
    }
  },
  // Additional event objects...
]
```


**RESULT:**

Thus, **Google Calendar and To-Do Lists were successfully managed** using **Google Cloud APIs**, and cloud functions were deployed for automation.

| Ex.No:4 | **D) Exploring Google Cloud for a Document Editing Tool** |
|---|---|
| | |

**AIM:**

To utilize Google Cloud tools and APIs for document creation, editing, and management using **Google Docs API**.

**ALGORITHM:**

**Step 1:** Start the process.

**Step 2:** Set up a **Google Cloud Project** in the **Google Cloud Console**.

**Step 3:** Create a new project or select an existing one.

**Step 4:** Enable **Google Docs API** from **API & Services → Library**.

**Step 5:** Navigate to **API & Services → Credentials** and create **OAuth 2.0 Client ID credentials**.

**Step 6:** Configure the **OAuth consent screen** and save the credentials.

**Step 7:** Install the **Google API Client Library** using pip.

**Step 8:** Implement Python code to create and modify a **Google Doc** programmatically.

**Step 9:** Run the script to generate a document and insert text.

**Step 10:** Use **Google Cloud Functions** for automated document creation.

**Step 11:** Deploy and execute the function to verify the output.

**Step 12:** Stop the process after successful execution.

**PROCEDURE:**

**1. Setting Up Google Cloud Project**

- Go to **Google Cloud Console** and create a **new project**.
- Navigate to **API & Services → Library** and enable **Google Docs API**.

**2. Creating OAuth 2.0 Credentials**

- Navigate to **API & Services → Credentials** and click **Create Credentials → OAuth 2.0 Client ID**.
- Configure the **consent screen** and set the application type to **Web Application**.

**3. Installing Google API Client Library**

- Install the **Google API client library** using pip:

```
pip install google-api-python-client
```

**4. Writing Python Code to Create and Modify a Google Doc**

```
from google.oauth2 import service_account

from googleapiclient.discovery import build


SCOPES = ['https://www.googleapis.com/auth/documents']

SERVICE_ACCOUNT_FILE = 'path/to/service.json'


creds =
service_account.Credentials.from_service_account_file(SERVICE_ACCOUN
T_FILE, scopes=SCOPES)

service = build('docs', 'v1', credentials=creds)

# Create a new document

document = service.documents().create(body={'title': 'Sample
Document'}).execute()

document_id = document.get('documentId')

print(f'Created document with ID: {document_id}')
```

```
# Insert text into the document

requests = [

    {

        'insertText': {

            'location': {'index': 1},

            'text': 'Hello, world!'

        }

    }

]

service.documents().batchUpdate(documentId=document_id,
body={'requests': requests}).execute()

print('Added text to document')
```

**5. Expected Output**

- The script will create a new **Google Doc** titled **"Sample Document"** and insert the text **"Hello, world!"** into it.

**6. Automating Document Creation with Google Cloud Functions**

- Use **Google Cloud Functions** to create documents dynamically using Node.js.

```
const { google } = require('googleapis');

exports.createDocument = async (req, res) => {

    const auth = new google.auth.GoogleAuth({

        keyFile: 'path/to/service.json',

        scopes: ['https://www.googleapis.com/auth/documents']

    });

    const docs = google.docs({ version: 'v1', auth });

    const response = await docs.documents.create({
```

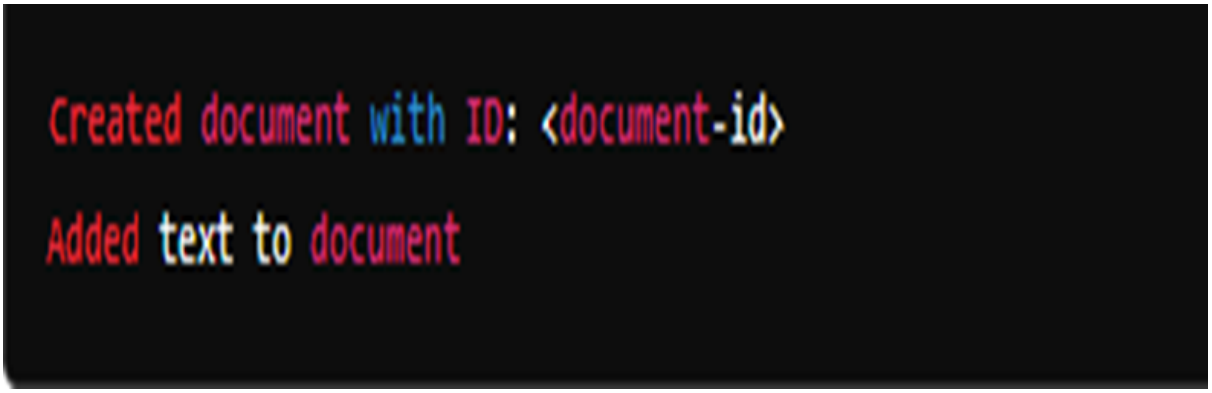
```
        requestBody: { title: 'New Document' }
    });
    res.status(200).send(`Document created with ID:
${response.data.documentId}`);
};
```

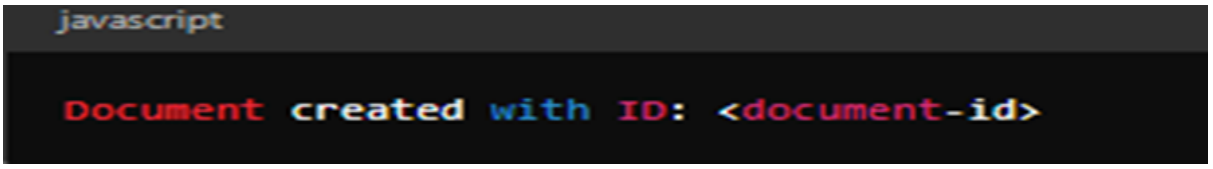
**7. Deploying and Testing the Cloud Function**

- Deploy the function in **Google Cloud Console** and trigger it via **HTTP request** to verify document creation.

**OUTPUT:**

The script will create a new Google Doc titled "Sample Document" and insert the text "Hello, world!" into it.




**RESULT:**

Thus, the **Google Docs API** was successfully used to **create, edit, and automate document management** using **Python and Google Cloud Functions**

| Ex.No:5 | **Working and Installation of Google App Engine** |
|---|---|
| | |

**AIM:**

To install and work with **Google App Engine** using **Python** for deploying a web application.

**ALGORITHM**

**Step 1:** Start the process.

**Step 2:** Install **Python** on your system if not already installed.

**Step 3:** Install **Google Cloud SDK** to manage Google Cloud resources.

**Step 4:** Sign in to **Google Cloud Console** and create a **new project** or use an existing one.

**Step 5:** Authenticate your account and set up your project using the command-line interface.

**Step 6:** Write a **Python script** for a simple web application using **Flask**.

**Step 7:** Deploy the application using **Google App Engine (GAE)**.

**Step 8:** Access the deployed application at **https://YOUR_PROJECT_ID.REGION_ID.r.appspot.com**.

**Step 9:** Stop the process after successful execution.

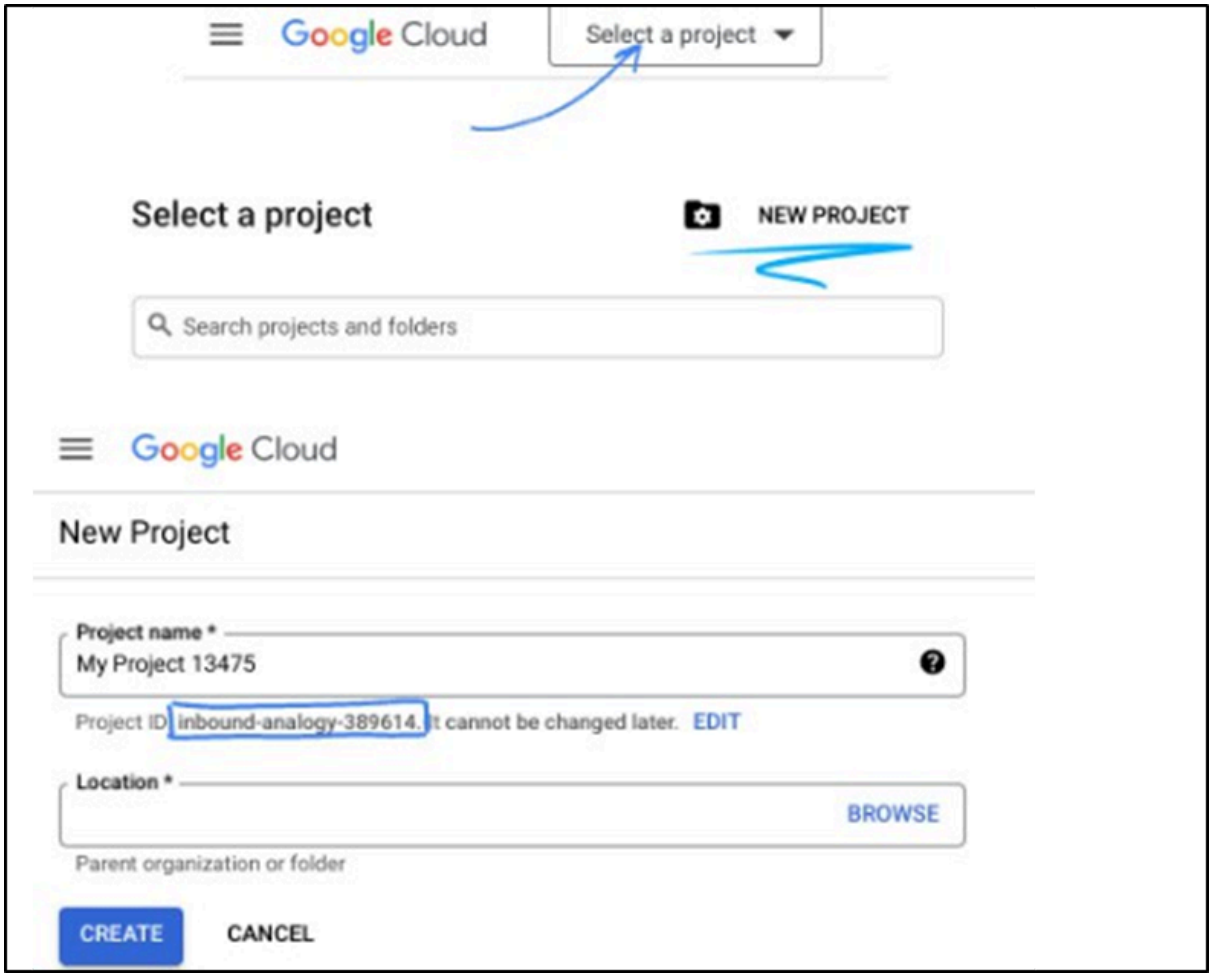
**PROCEDURE**

**1. Installing Python and Google Cloud SDK**

Ensure Python is installed by running the command:

```
python --version
```

- Download and install **Google Cloud SDK** from:
  https://cloud.google.com/sdk/docs/install




**2. Creating a Google Cloud Project**

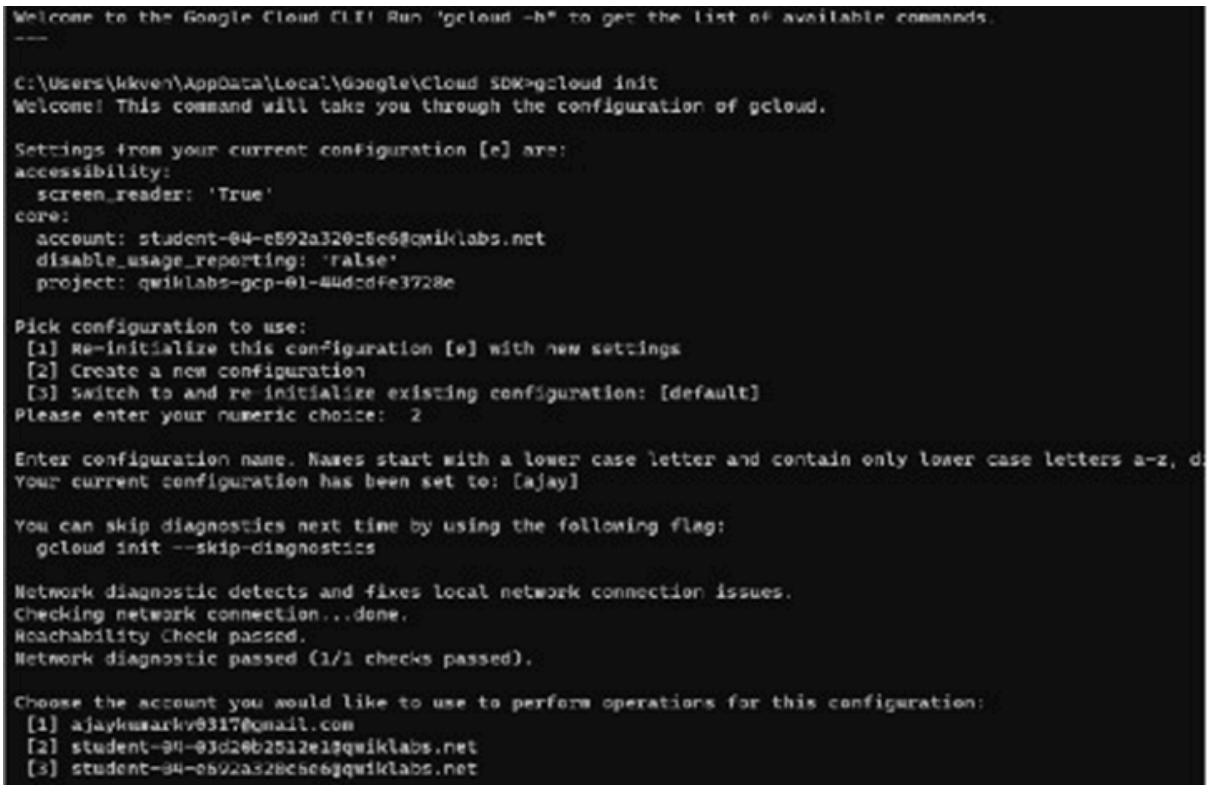
- Sign in to **Google Cloud Console**: https://console.cloud.google.com
- Create a **new project** and note the **Project ID**.

**3. Authenticating and Setting Up the Project**

Open the terminal and authenticate your Google account:

```
gcloud auth login
```

- Set your **Google Cloud Project**:
  `gcloud config set project YOUR_PROJECT_ID`




**4. Writing a Python Script for a Web Application**

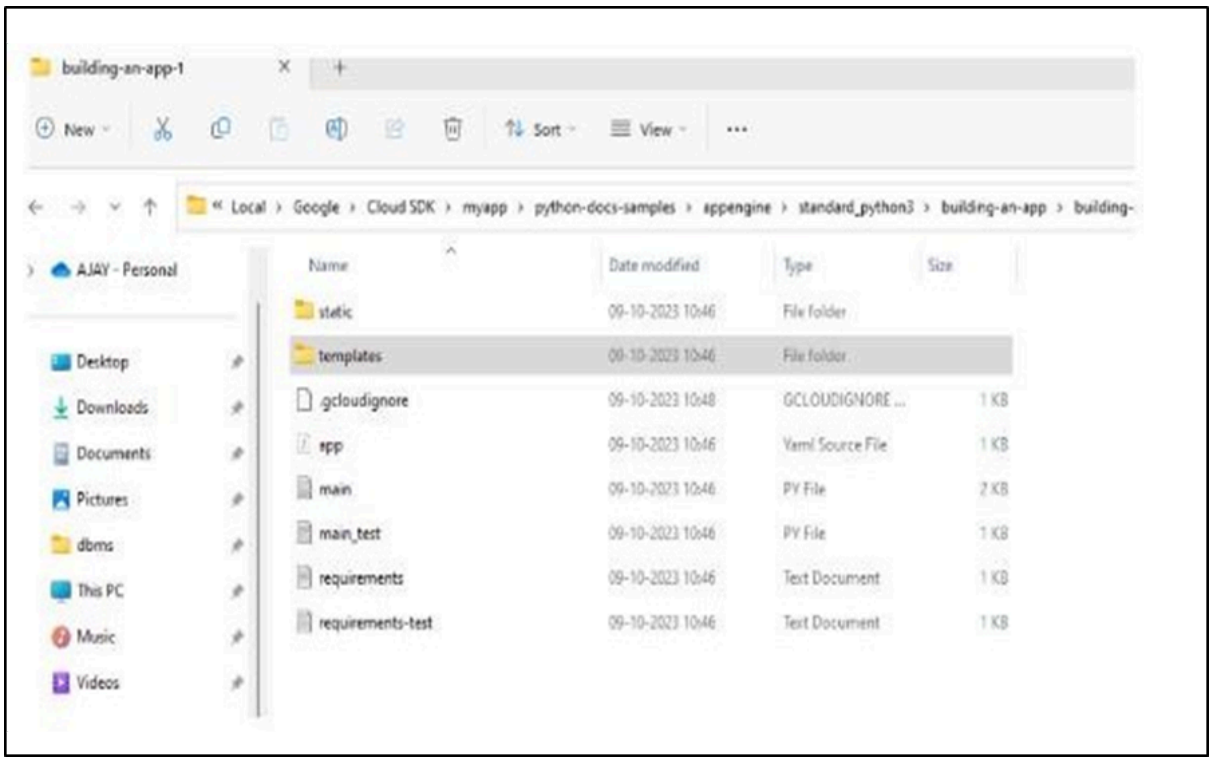
Create a file named **main.py** and write the following Flask application:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, Welcome to Google App Engine!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080)
```





### 5. Creating an App Engine Configuration File

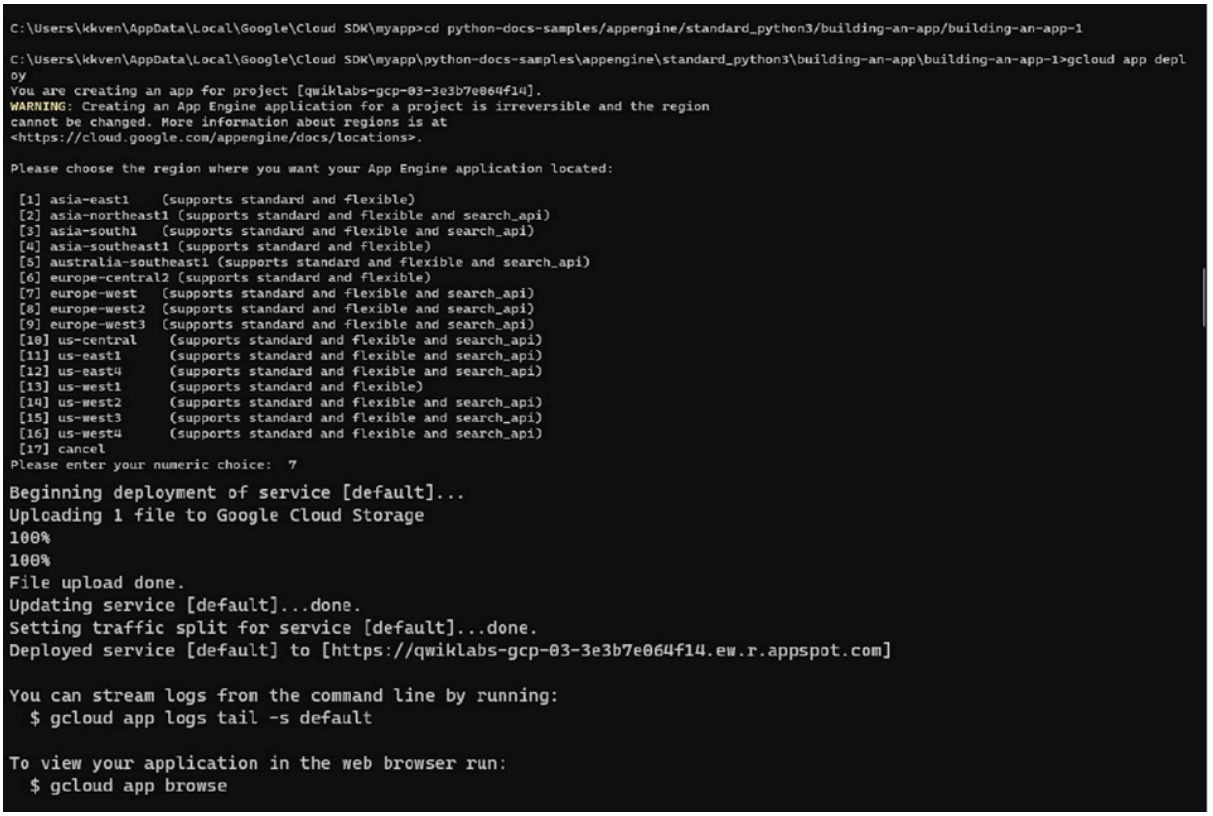
Create an **app.yaml** file in the same directory:

```
runtime: python39
entrypoint: gunicorn -b :$PORT main:app
```

### 6. Deploying the Application to Google App Engine

Run the following command to deploy the application:

```
gcloud app deploy
```




### 7. Accessing the Deployed Web Application

After deployment, access the application at:
`https://YOUR_PROJECT_ID.REGION_ID.r.appspot.com`

- Replace **YOUR_PROJECT_ID** with your actual project ID and **REGION_ID** with the selected region.

**RESULT:**

Thus, the **Google App Engine** was successfully installed, configured, and used to deploy a **Python web application**.

| Ex.No:6 | **Working and Installation of Microsoft Azure** |
|---|---|
| | |

**AIM:**

To install and work with **Microsoft Azure** for managing cloud resources efficiently.

**ALGORITHM**

**Step 1:** Start the process.

**Step 2:** Open a web browser and sign in to the **Azure Portal** at https://portal.azure.com.

**Step 3:** Familiarize yourself with the **Azure Portal Dashboard**, including sections like **Home, All Services, Dashboard, and Notifications**.

**Step 4:** Navigate to **Resource Groups** to manage and organize Azure resources effectively.

**Step 5:** Create a new **Resource Group**, assign a meaningful name, and select a location.

**Step 6:** Understand how **Azure Resource Groups** help in organizing related cloud resources efficiently.

**Step 7:** Go to **Subscriptions** and explore details such as **subscription type, status, and resource usage**.

**Step 8:** Manage **Access Control (IAM)** to add users, assign roles, and set permissions for different services.

**Step 9:** Explore **All Services** in the left-hand menu to get an overview of available Azure services.

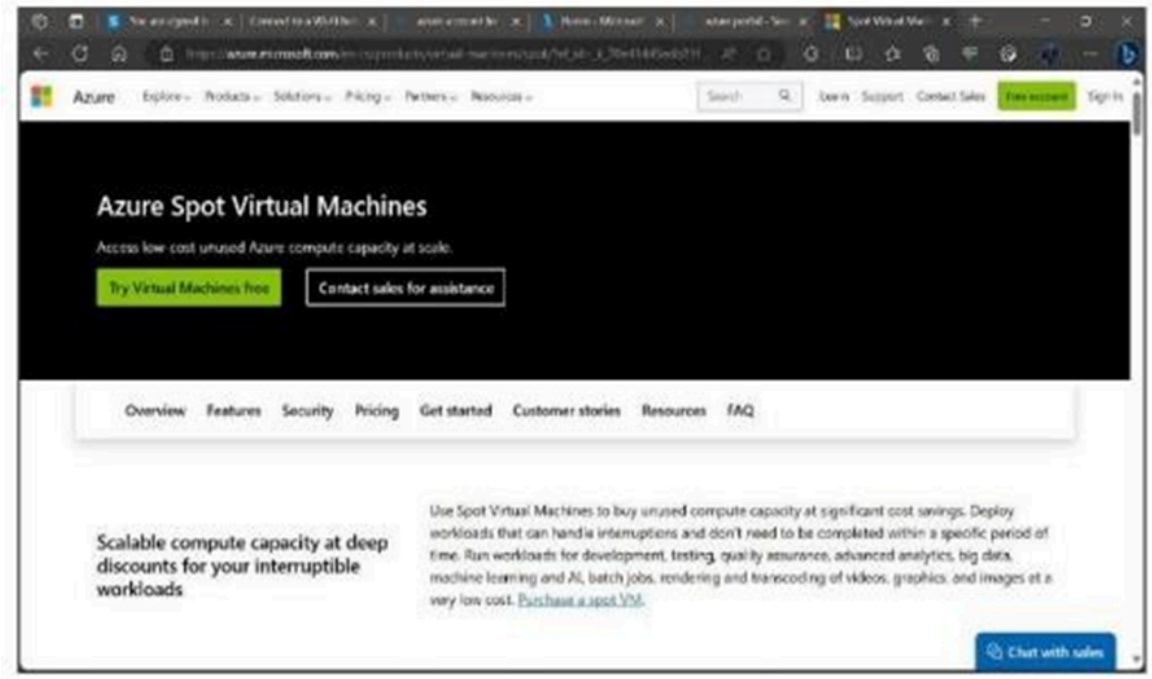
**Step 10:** Navigate to the **Azure Marketplace** to deploy third-party solutions and services.

**Step 11:** Save changes, log out of your **Azure Portal**, and complete the session.

**Step 12:** Stop the process after successful execution.

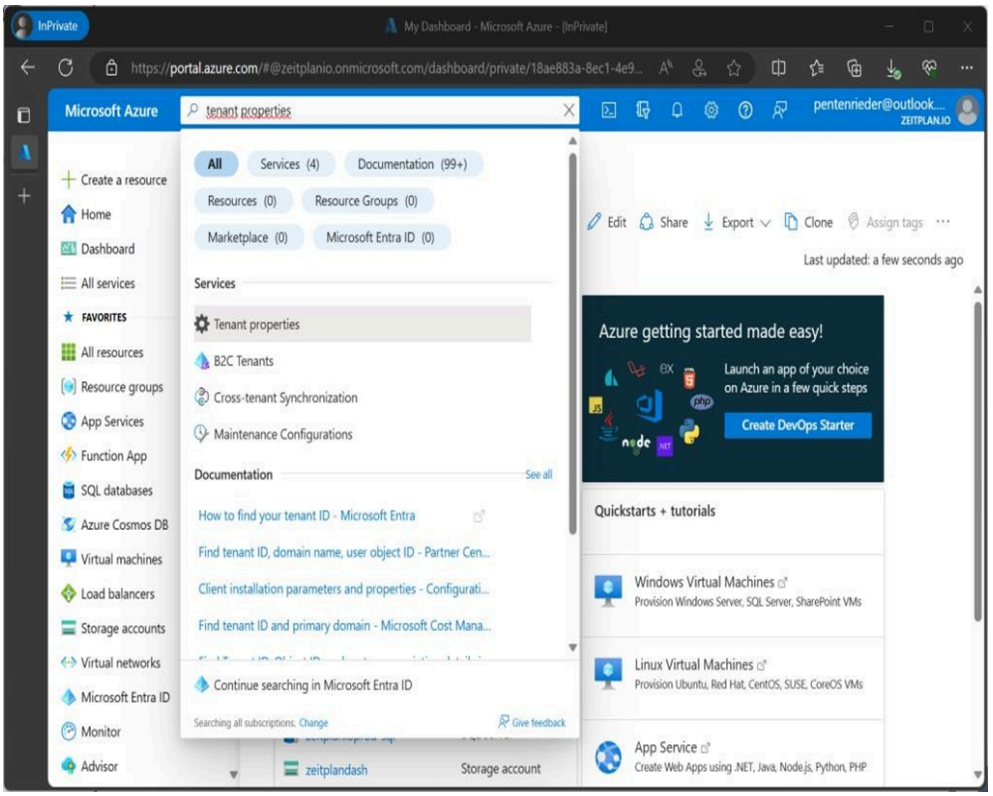
## PROCEDURE

1. **Signing into Azure Portal**
    - Open a web browser and visit https://portal.azure.com.
    - Enter your Microsoft Azure credentials and log in.



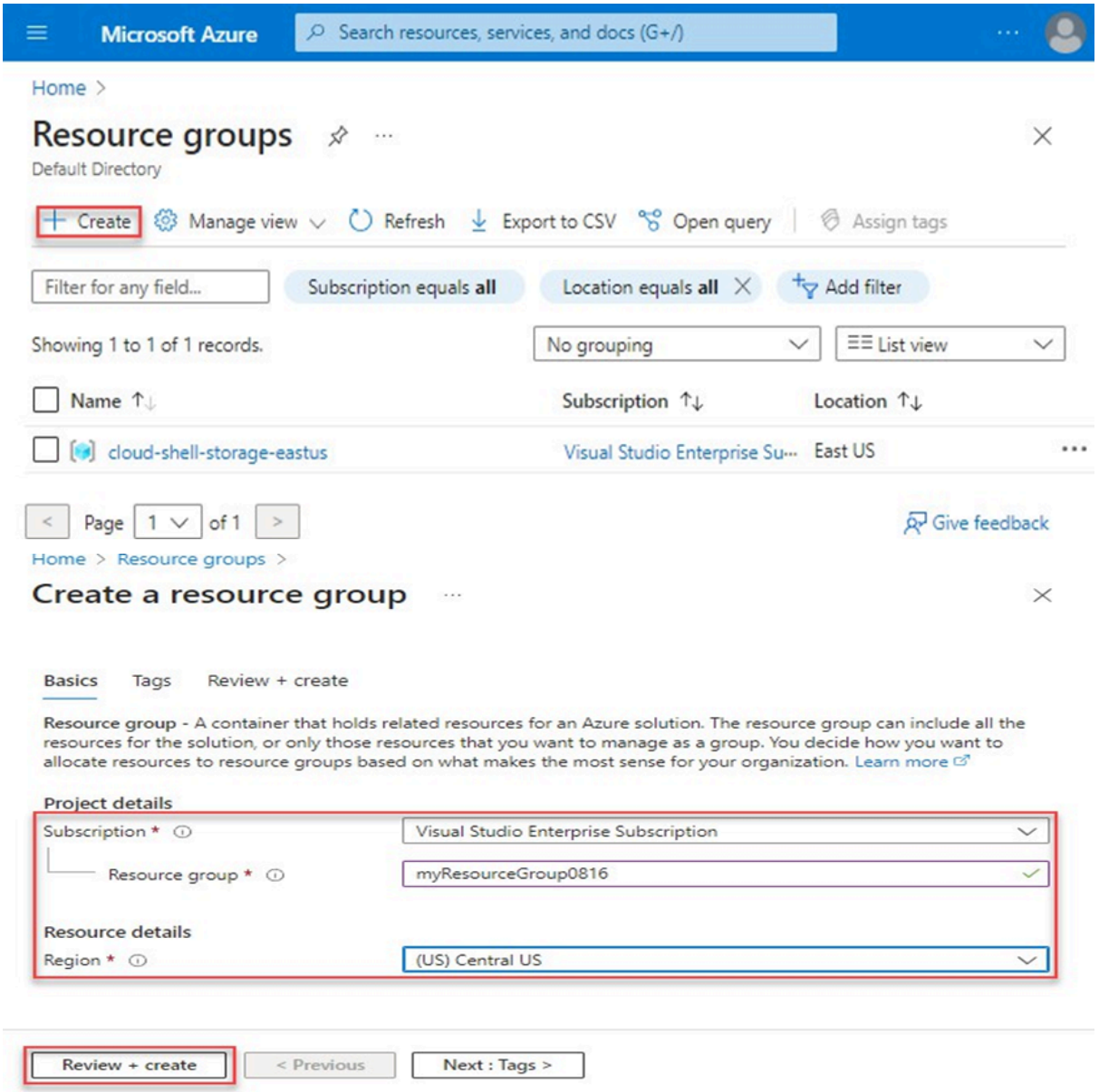

**Exploring the Azure Portal Interface**

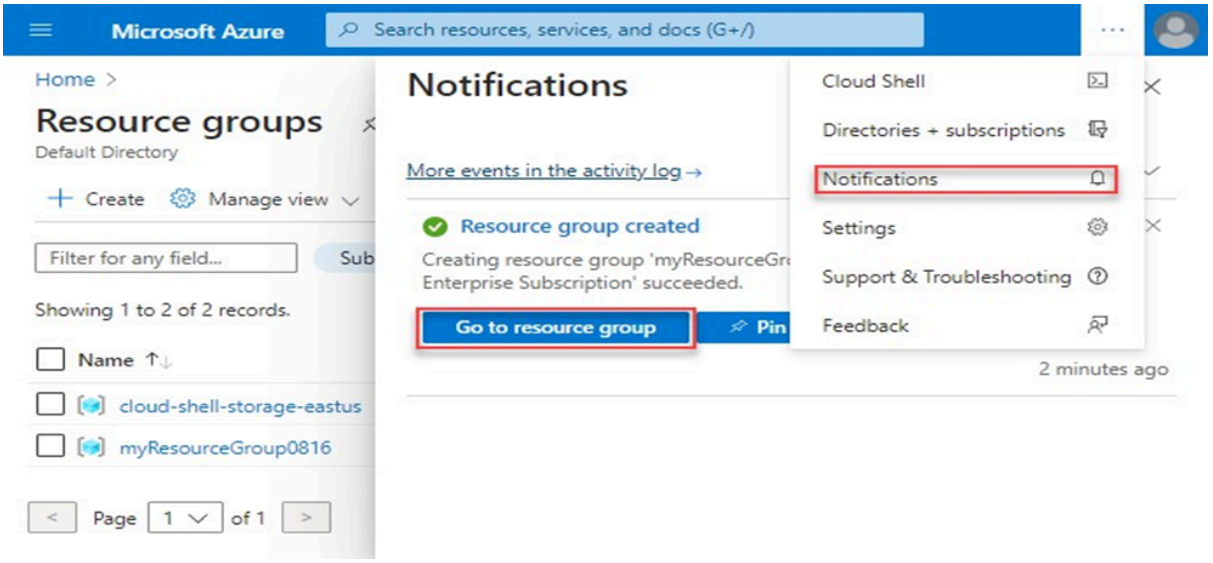
- The **Azure Portal Dashboard** provides access to multiple cloud services.
- Use the **left-hand menu** to navigate between **Home, All Services, Dashboard, and Notifications**.



### Creating and Managing Resource Groups

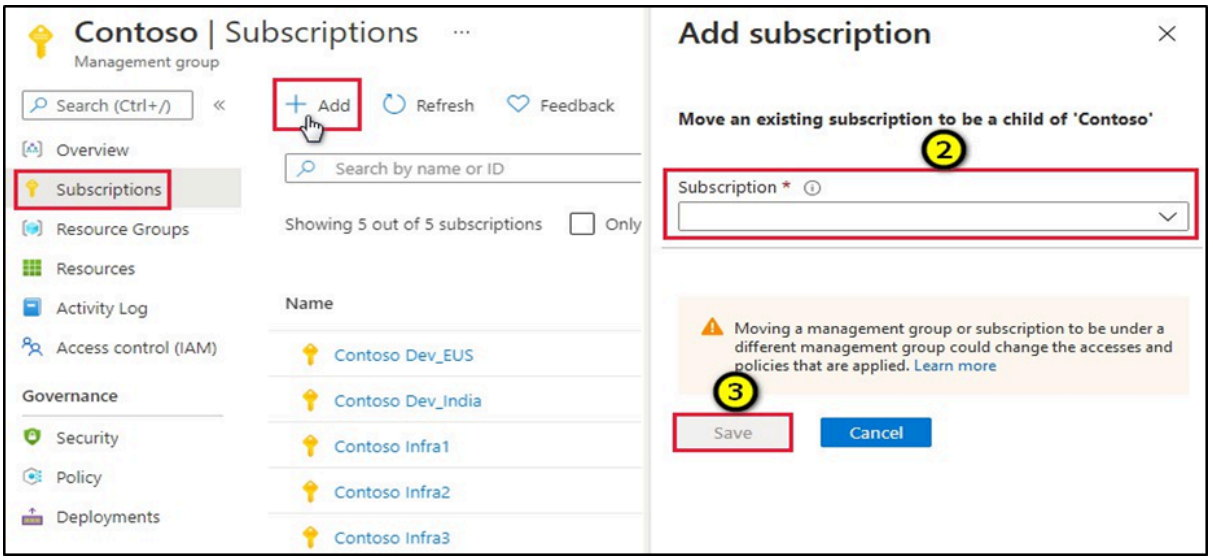
- Click on **"Resource Groups"** in the left-hand menu.
- View existing resource groups or create a new one by clicking **"+ Add"**.
- Enter a meaningful **name** and select a **location** for the resource group.
- Understand how **resource groups** help in managing related Azure resources.






**Managing Azure Subscriptions**

- Click on **"Subscriptions"** to view and manage your **Azure Subscription** details.
- Explore **subscription type, status, and resource usage**.

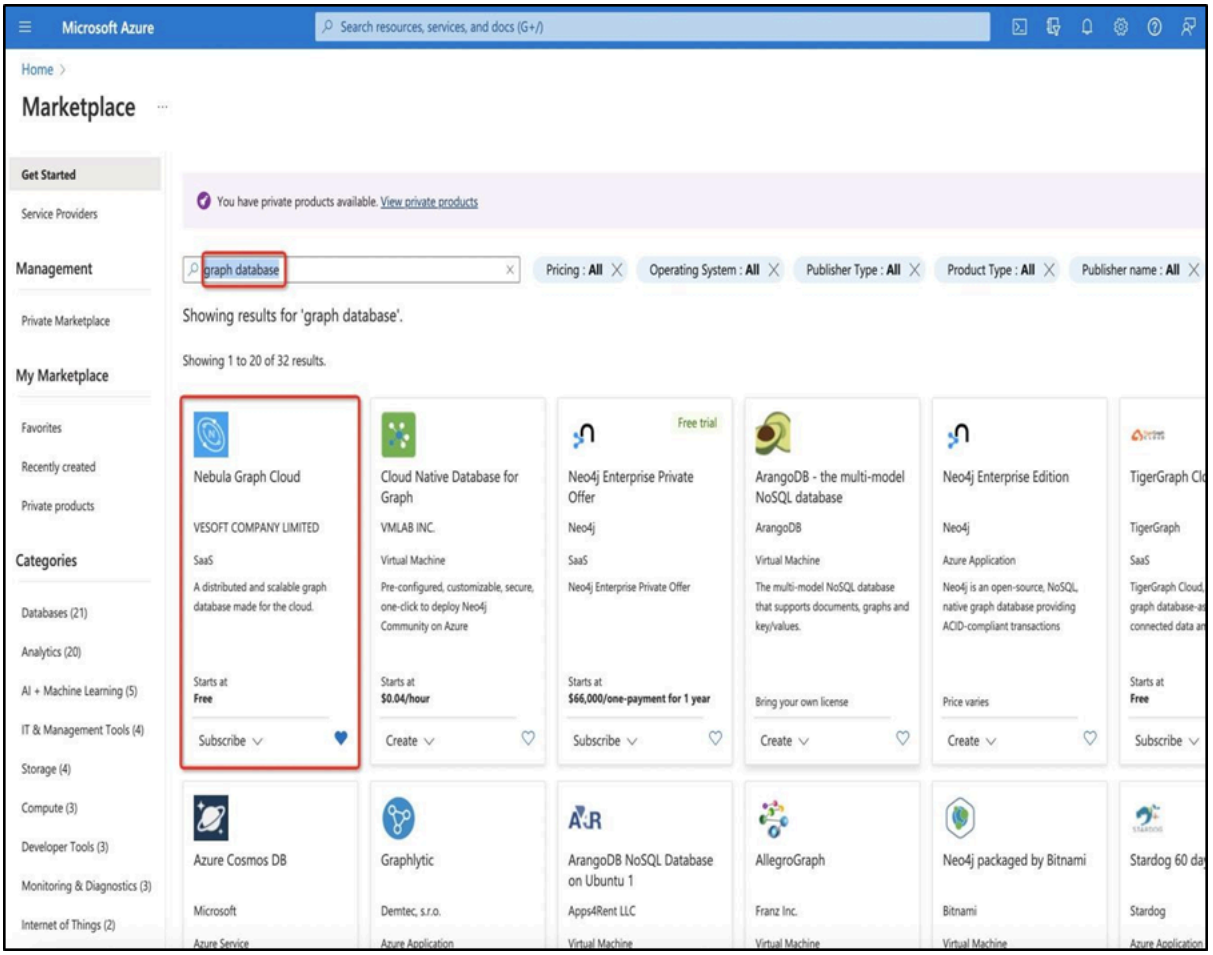



**Setting Up Access Control (IAM)**

- Under the **"Subscriptions"** tab, click **"Access Control (IAM)"**.
- Add users, assign roles, and configure permissions for various **Azure services**.

**Exploring Azure Services and Marketplace**

- Click **"All Services"** in the left-hand menu to explore all available **Azure services**.
- Visit the **Azure Marketplace** to deploy **third-party applications and solutions**.




**Finalizing Setup**

- Save the necessary changes and log out of your **Azure account**.

**RESULT:**

Thus, the **Microsoft Azure Portal** was successfully explored, including **resource group management, access control, and marketplace services**.

| Ex.No:7 | **Connecting Amazon Redshift with S3 Bucket** |
|---|---|
| | |

**AIM:**

To connect **Amazon Redshift** with **Amazon S3** to handle and transfer data efficiently.

**ALGORITHM**

**Step 1:** Log in to the **AWS Console**, navigate to **Amazon Redshift**, and choose the option to create a cluster.

**Step 2:** Configure the **Redshift Cluster** by selecting **node type, number of nodes, admin username, and password**.

**Step 3:** Click on **Create Cluster** and wait for the cluster status to change to **available**.

**Step 4:** Create an **IAM Role** in the AWS IAM service to allow **Redshift to access S3**.

**Step 5:** Choose **AWS service** as the trusted entity, select **Redshift** as the use case, and attach the **AmazonS3FullAccess** policy.

**Step 6:** Assign a name to the IAM role and create it.

**Step 7:** Associate the IAM role with the **Redshift cluster** by selecting **Manage IAM Roles** under cluster settings.

**Step 8:** Access the **Query Editor** in the Redshift dashboard and connect to the database using the **admin username and password**.

**Step 9:** Create a **table in the Redshift database** by specifying the required columns.

**Step 10:** Create an **S3 Bucket** and upload the **.txt file** containing data.

**Step 11:** Copy data from S3 to Redshift using the following command:

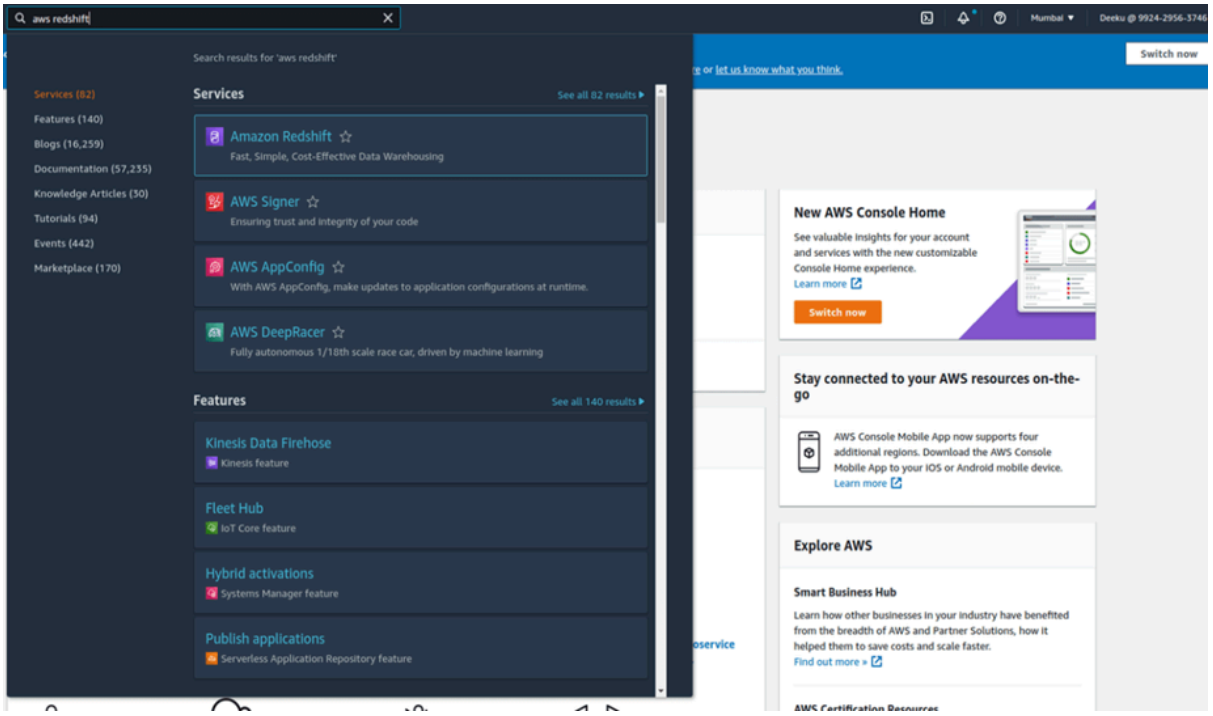
```
COPY Orders FROM 's3://your-bucket-name/your-data-file.txt'
IAM_ROLE 'your-iam-role-arn'
REGION 'us-east-1'
IGNOREHEADER 1;
```

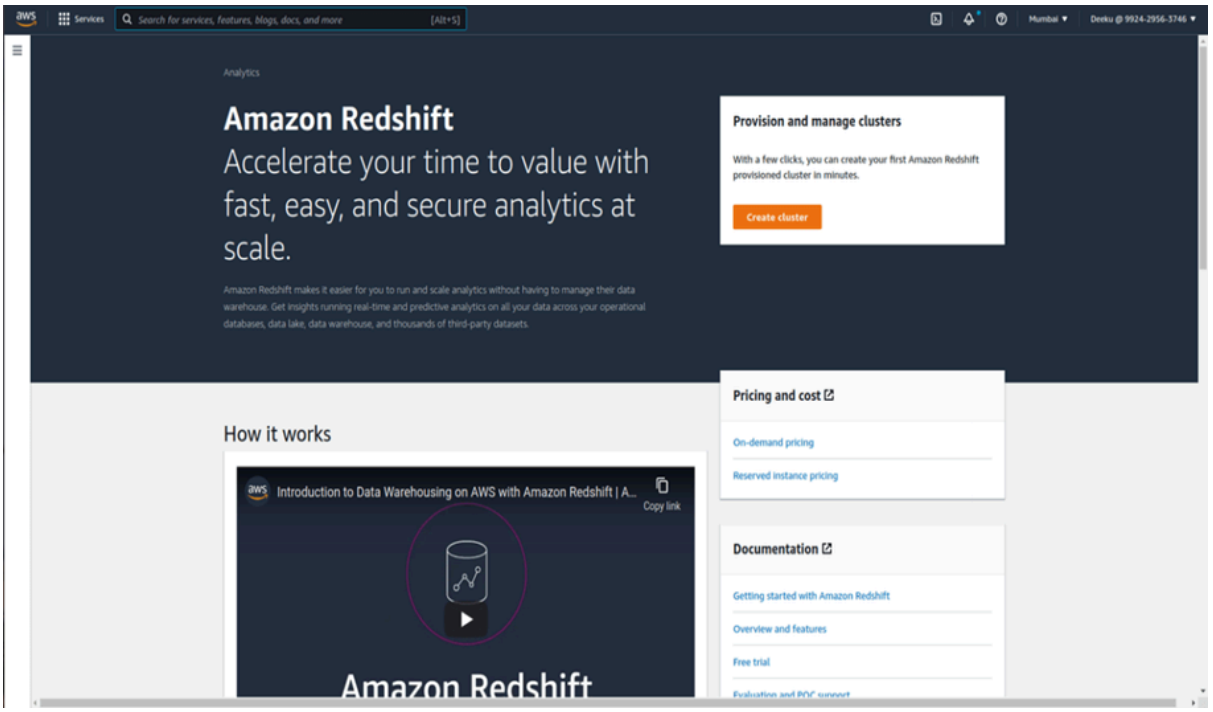
**Step 12:** Verify the data import in Redshift by running a **SELECT query**.

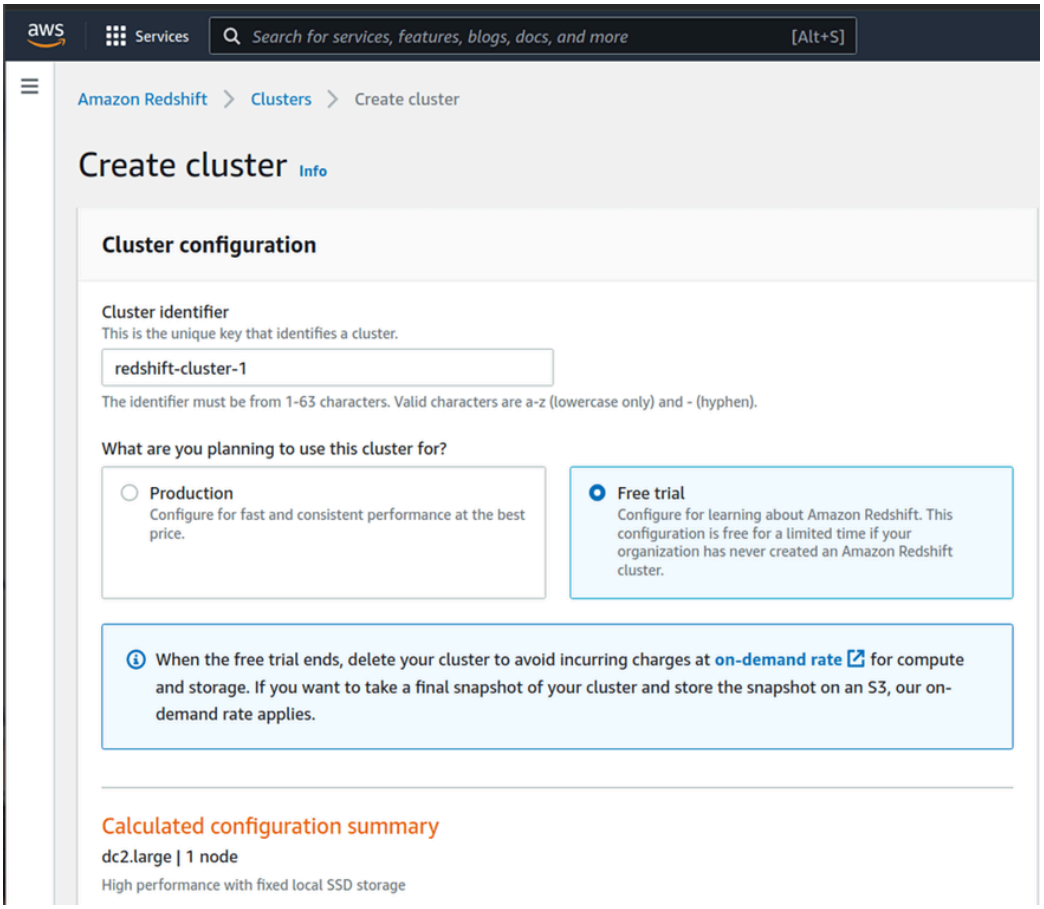
## PROGRAM IMPLEMENTATION:

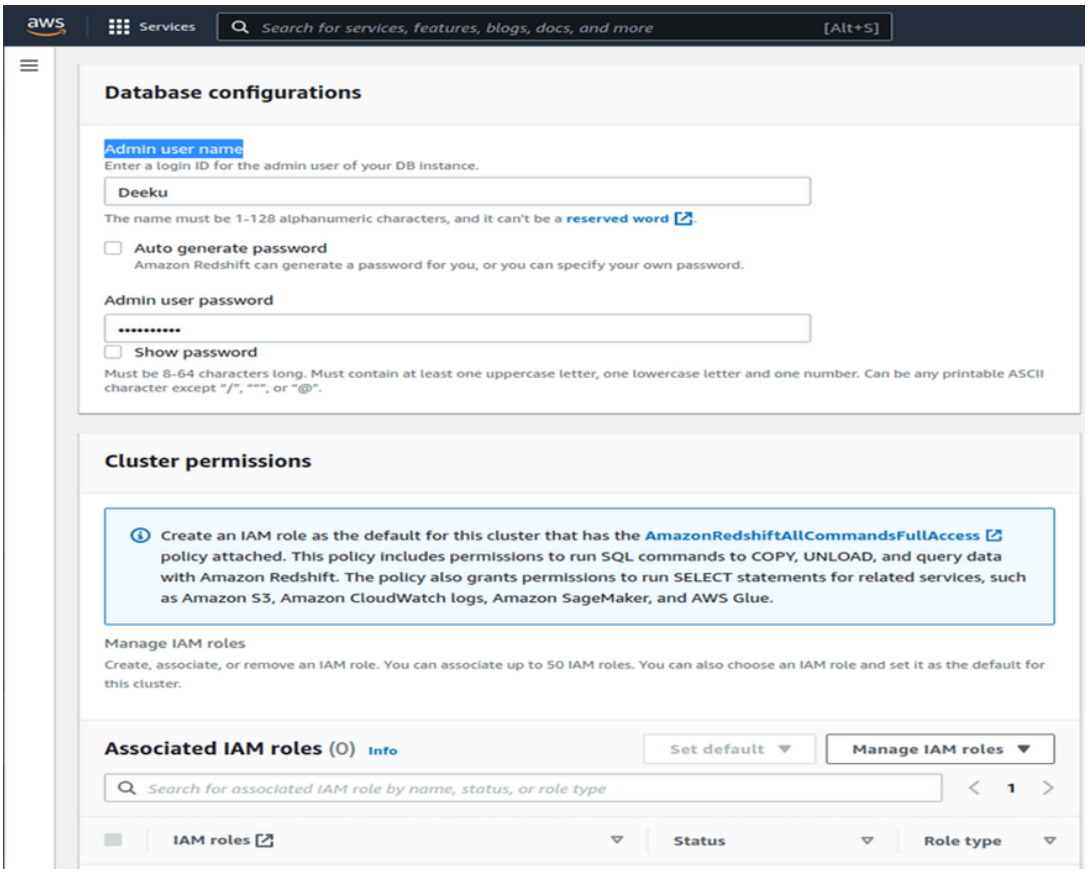
### 1. Creating an Amazon Redshift Cluster

- Log in to the AWS Console and navigate to Amazon Redshift.
- Click on Create Cluster and configure the cluster settings (node type, number of nodes, database name, admin username, and password).
- Click Create Cluster and wait until the cluster status changes to available.



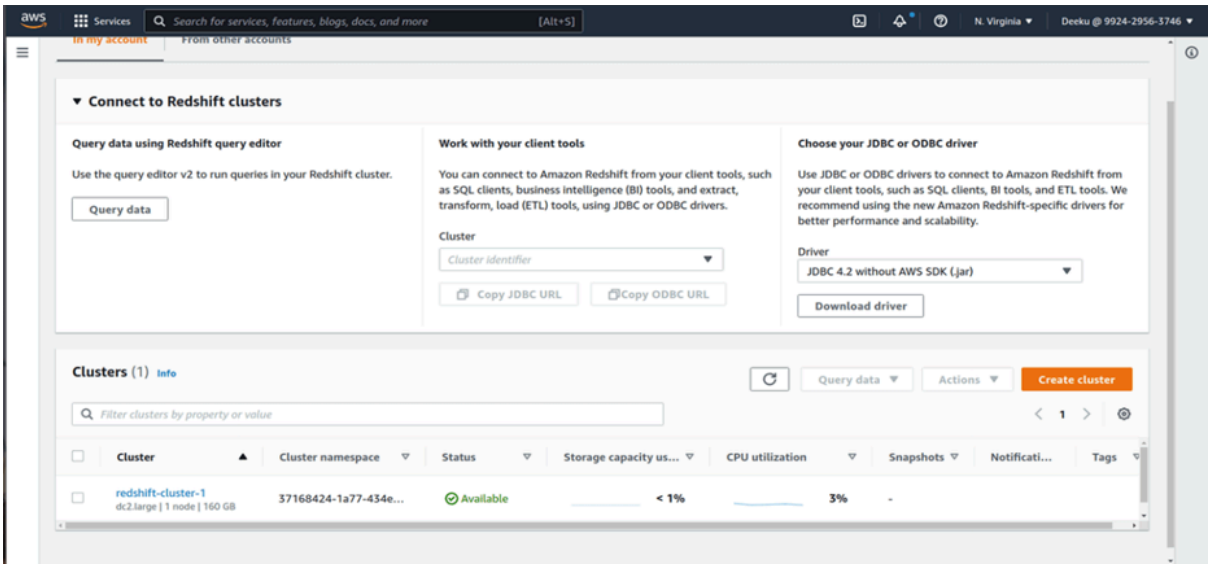




aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Amazon Redshift
Clusters
Create cluster
Create cluster
Info
Cluster configuration
Cluster identifier
This is the unique key that identifies a cluster.
redshift-cluster-1
The identifier must be from 1-63 characters. Valid characters are a-z (lowercase only) and - (hyphen).
What are you planning to use this cluster for?
Production
Configure for fast and consistent performance at the best price.
Free trial
Configure for learning about Amazon Redshift. This configuration is free for a limited time if your organization has never created an Amazon Redshift cluster.
When the free trial ends, delete your cluster to avoid incurring charges at on-demand rate for compute and storage. If you want to take a final snapshot of your cluster and store the snapshot on an S3, our on-demand rate applies.
Calculated configuration summary
dc2.large | 1 node
High performance with fixed local SSD storage


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Database configurations
Admin user name
Enter a login ID for the admin user of your DB instance.
Deeku
The name must be 1-128 alphanumeric characters, and it can't be a reserved word.
Auto generate password
Amazon Redshift can generate a password for you, or you can specify your own password.
Admin user password
Show password
Must be 8-64 characters long. Must contain at least one uppercase letter, one lowercase letter and one number. Can be any printable ASCII character except "/", """, or "@".
Cluster permissions
Create an IAM role as the default for this cluster that has the AmazonRedshiftAllCommandsFullAccess policy attached. This policy includes permissions to run SQL commands to COPY, UNLOAD, and query data with Amazon Redshift. The policy also grants permissions to run SELECT statements for related services, such as Amazon S3, Amazon CloudWatch logs, Amazon SageMaker, and AWS Glue.
Manage IAM roles
Create, associate, or remove an IAM role. You can associate up to 50 IAM roles. You can also choose an IAM role and set it as the default for this cluster.
Associated IAM roles (0)
Info
Set default
Manage IAM roles
Search for associated IAM role by name, status, or role type
1
IAM roles
Status
Role type

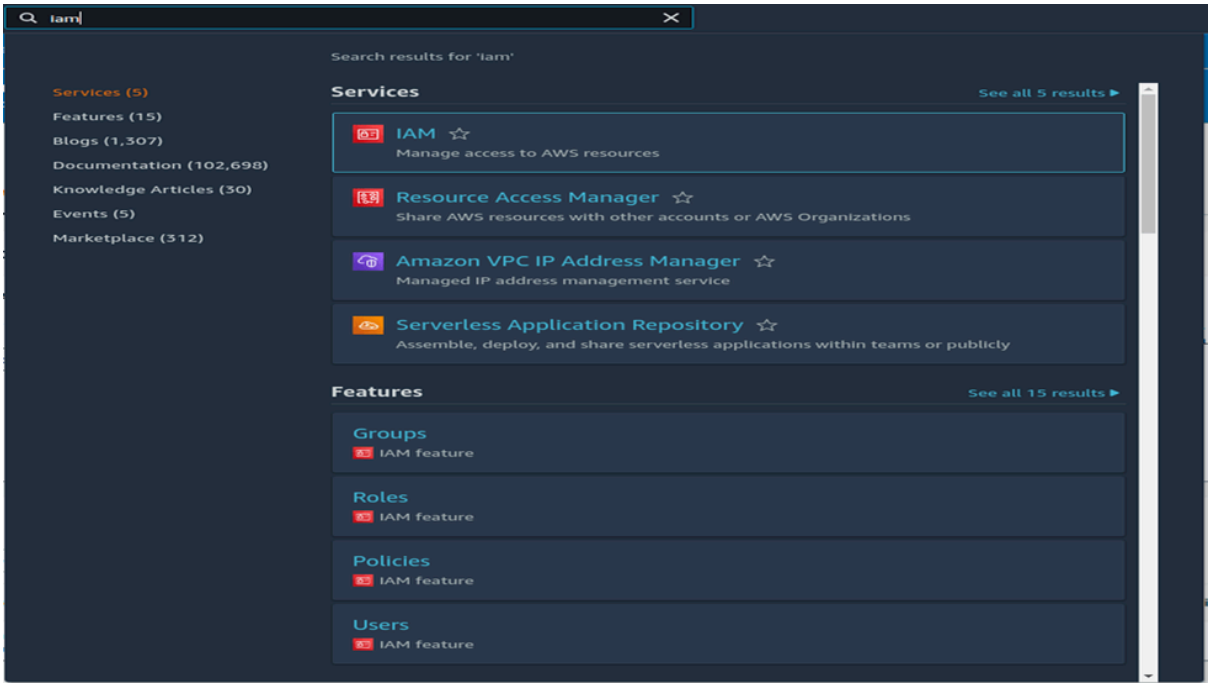
2. Creating an IAM Role for Redshift

- Open AWS IAM Service, click on Create Role, and choose AWS Service as the trusted entity.
- Select Redshift as the use case and attach the AmazonS3FullAccess policy.
- Assign a name to the role and click Create Role.




**3. Associating IAM Role with Redshift**

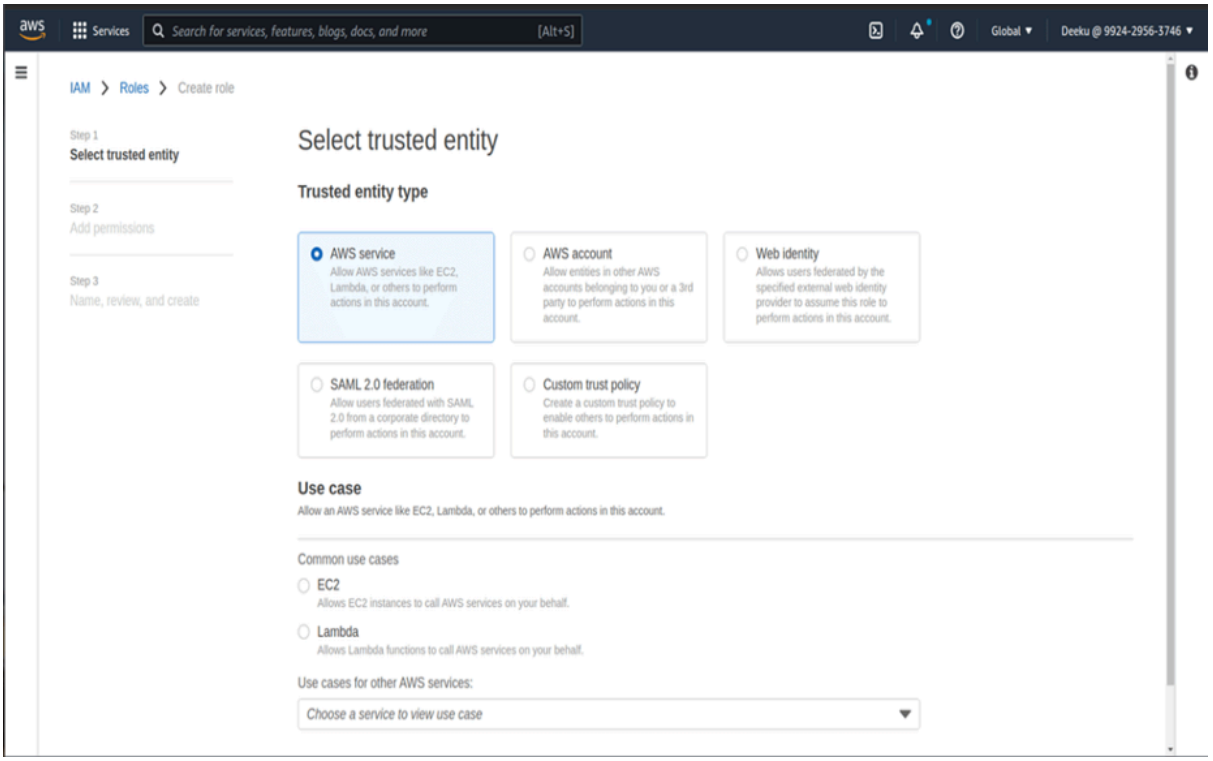
- Navigate to the **Redshift cluster**, select **Manage IAM Roles**, and attach the created IAM role.



### 4. Creating a Table in Redshift

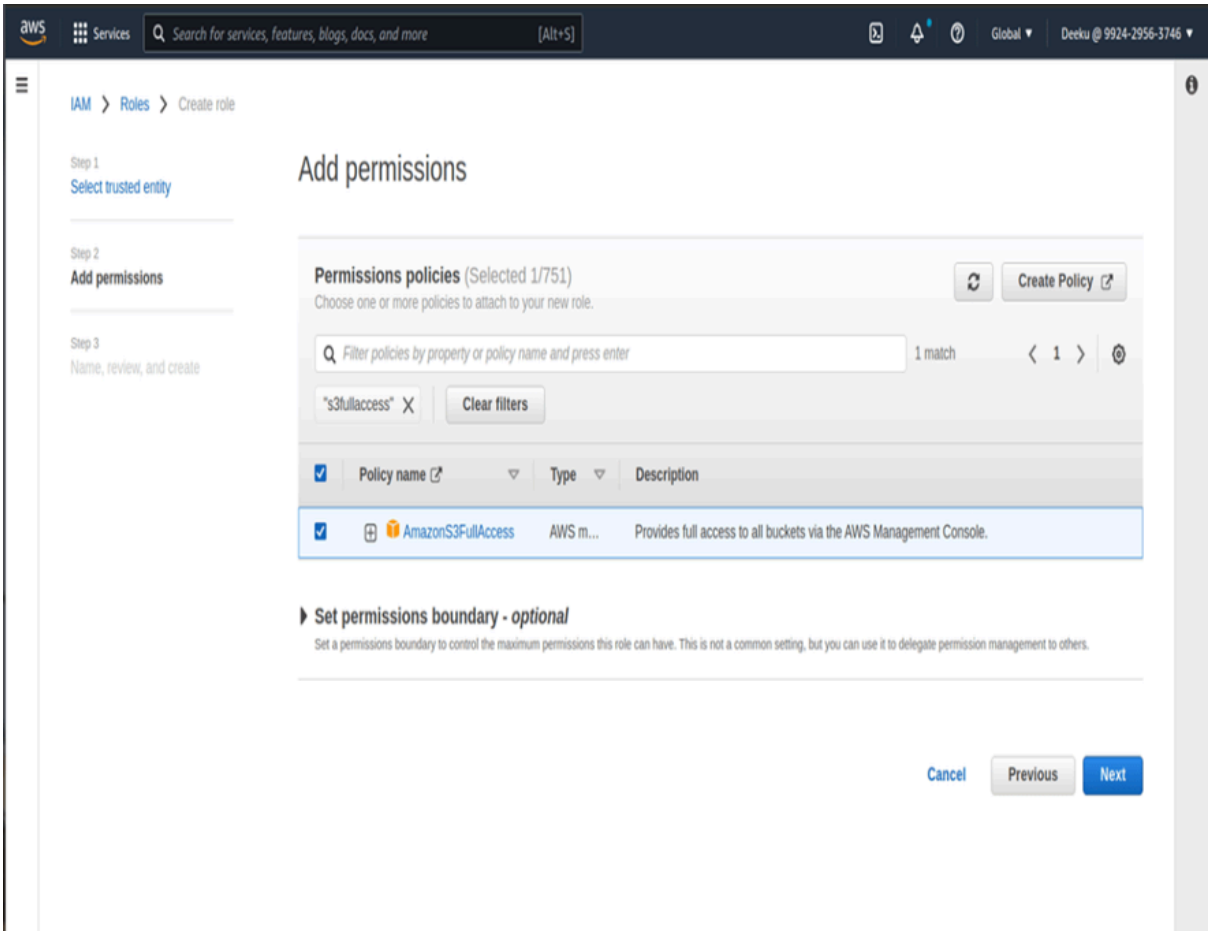
- Open the **Redshift Query Editor**, connect to the database using **admin credentials**, and run the following SQL command to create a table:

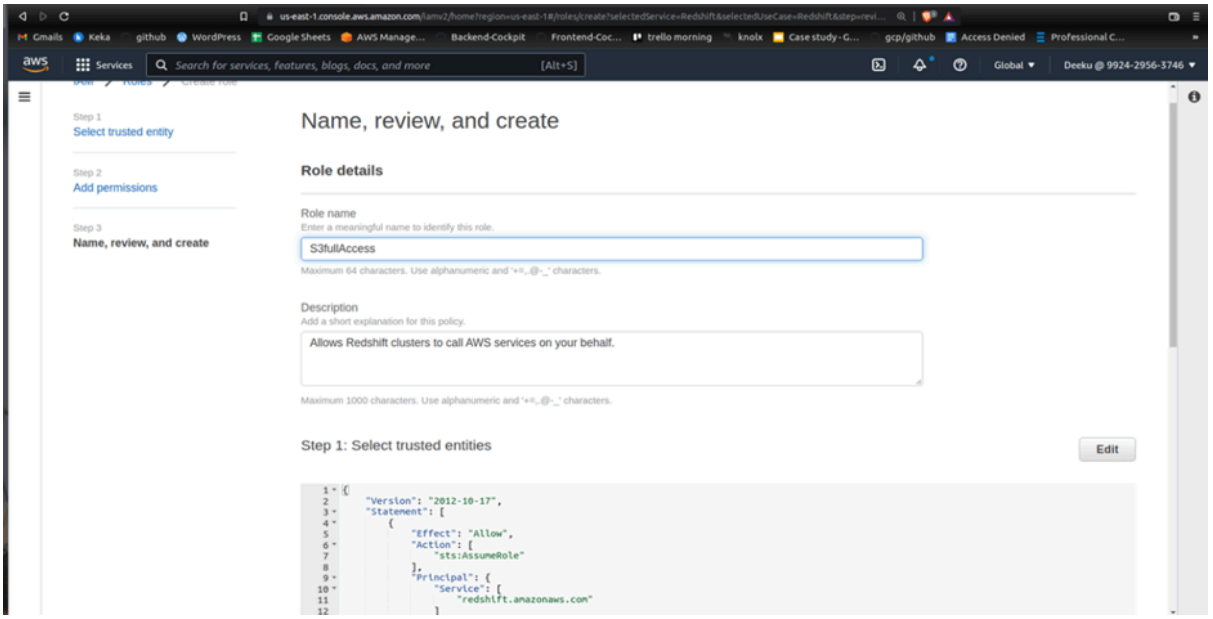
```
CREATE TABLE Orders (
    OrderID INT,
    CustomerName VARCHAR(255),
    OrderAmount DECIMAL(10,2),
    OrderDate DATE
);
```



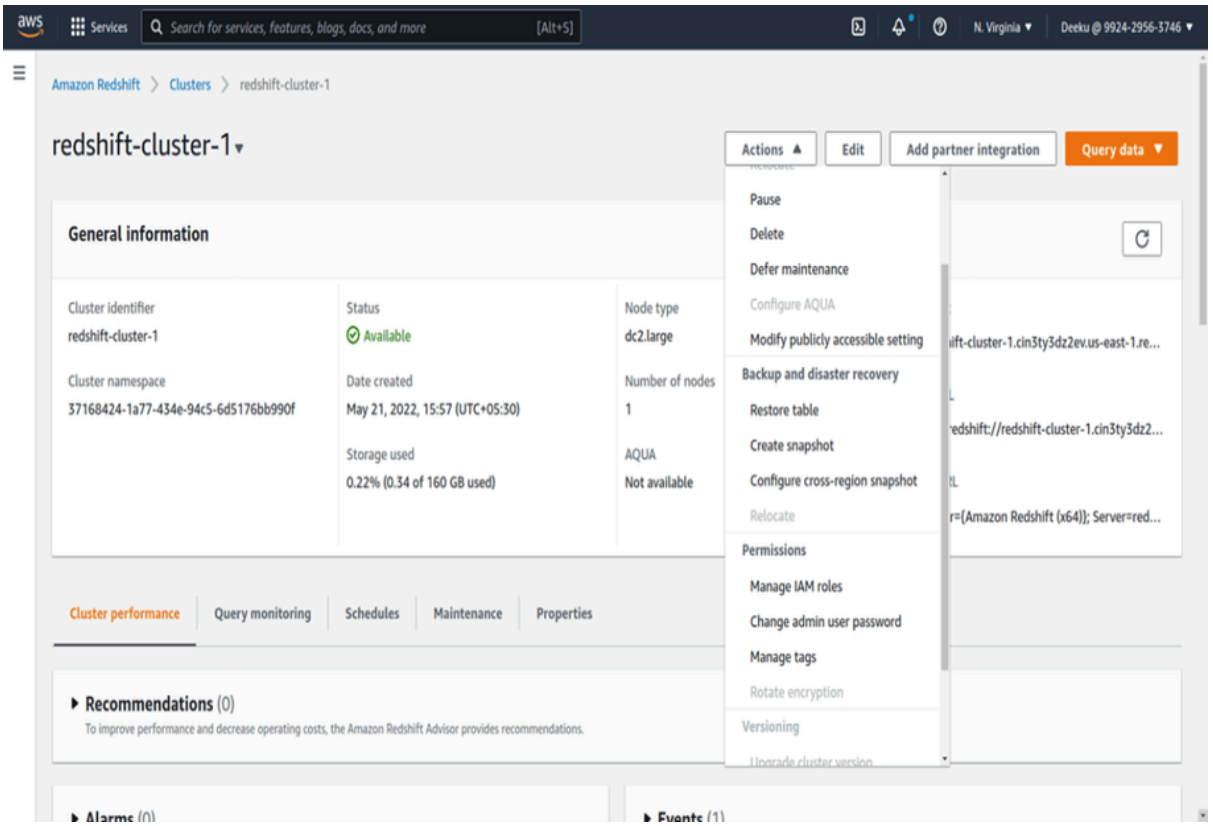

### 5. Uploading Data to S3 Bucket

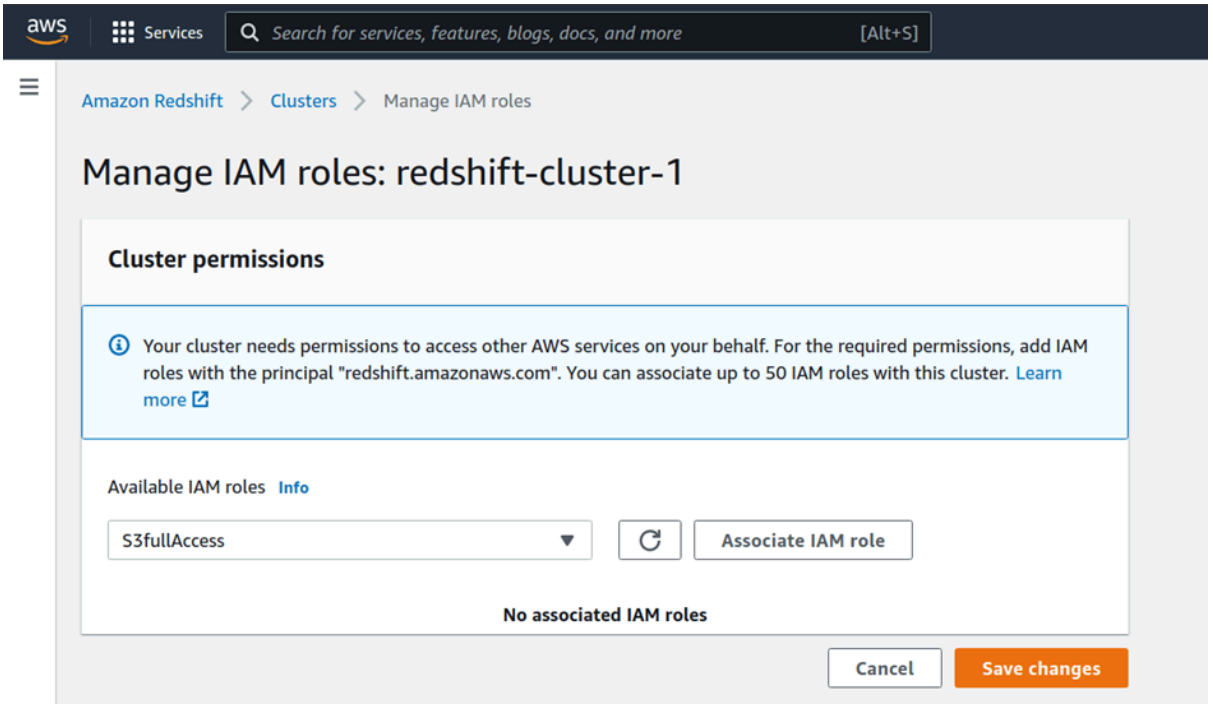
- Go to **S3 Service** in AWS, create a new **S3 Bucket**, and upload the **data file (orders.txt)**.


Services
Search for services, features, blogs, docs, and more
[Alt+S]
Global
Deeku @ 9924-2956-3746
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Add permissions
Permissions policies (Selected 1/751)
Choose one or more policies to attach to your new role.
Create Policy
Filter policies by property or policy name and press enter
1 match
1
"s3fullaccess"
Clear filters
Policy name
Type
Description
AmazonS3FullAccess
AWS m...
Provides full access to all buckets via the AWS Management Console.
Set permissions boundary - optional
Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting, but you can use it to delegate permission management to others.
Cancel
Previous
Next

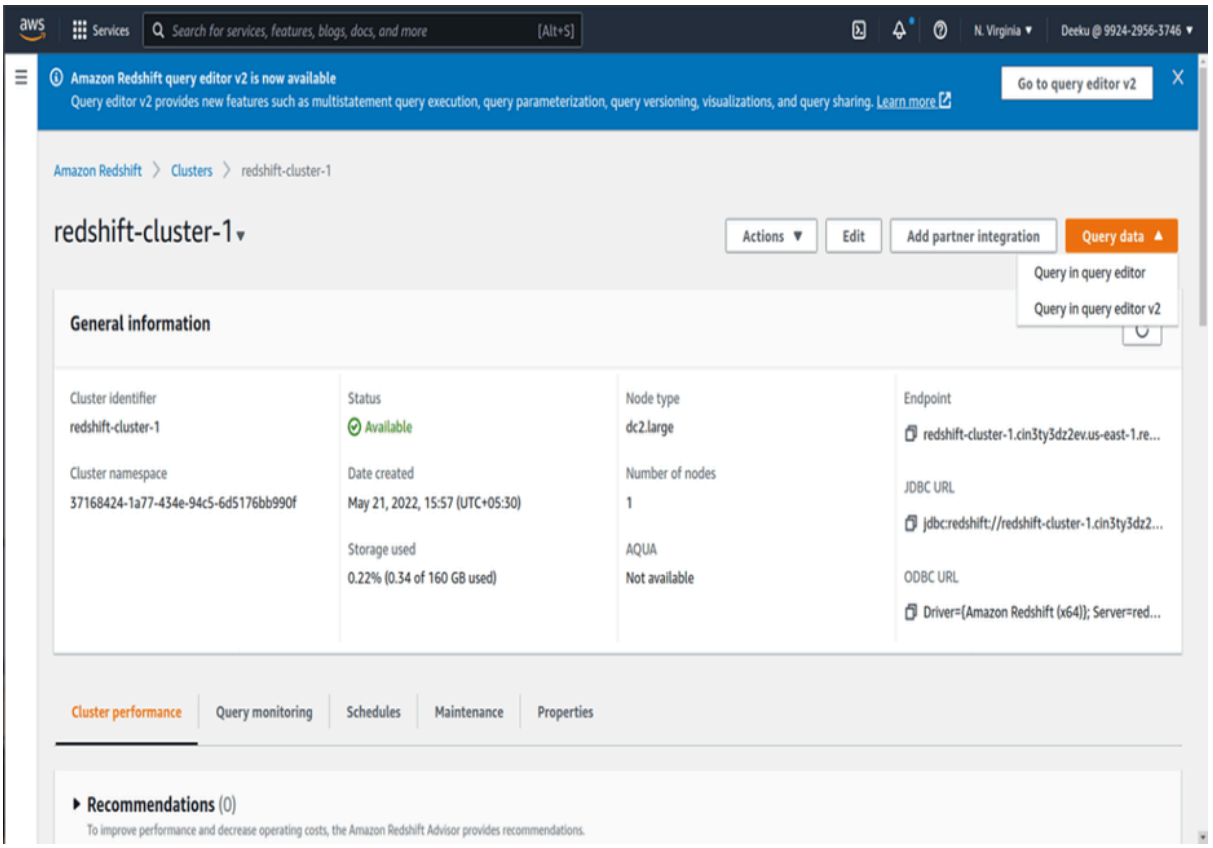

Services
Search for services, features, blogs, docs, and more
[Alt+S]
Global
Deeku @ 9924-2956-3746
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
S3fullAccess
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this policy.
Allows Redshift clusters to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Step 1: Select trusted entities
Edit
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"sts:AssumeRole"
],
"Principal": {
"Service": [
"redshift.amazonaws.com"

- Here, we need to associate IAM role with Redshift. In the create Redshift cluster, under actions, there is an option as *manage IAM role*. There we can associate the created role by giving the role name.



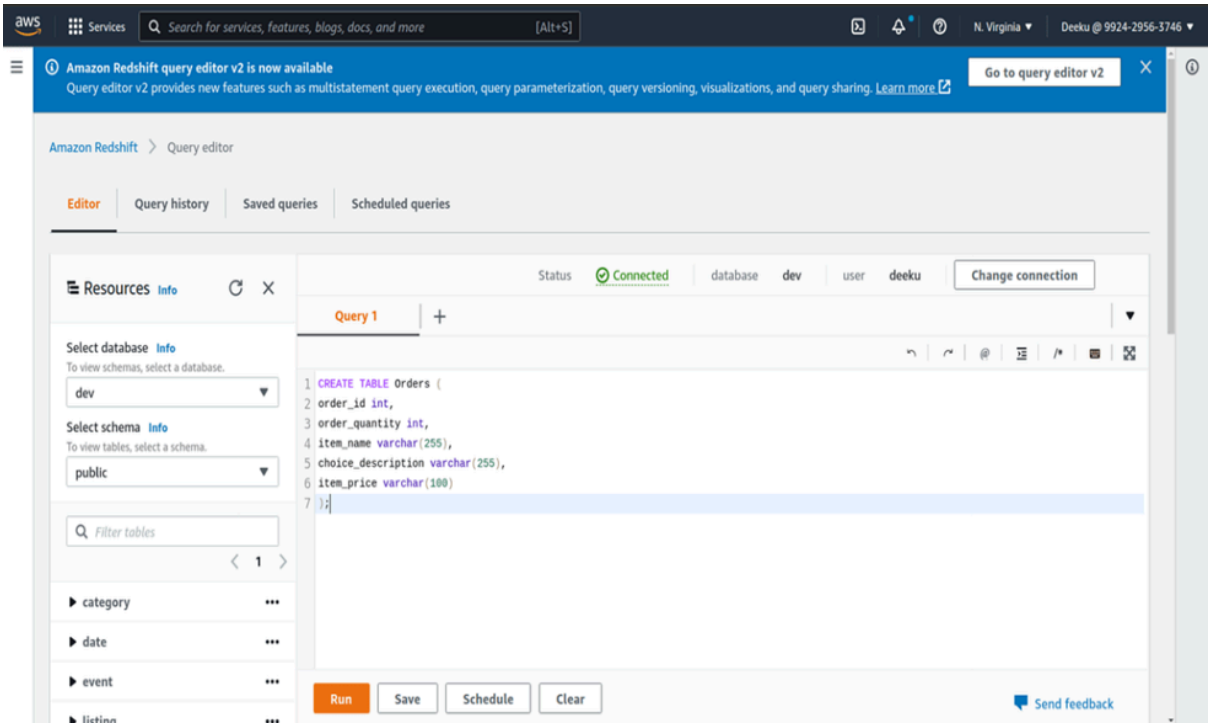
- In Redshift dashboard, we have query editor which is used to query the table. Similarly to access the query editor, connect to the database created when we launched Redshift cluster.



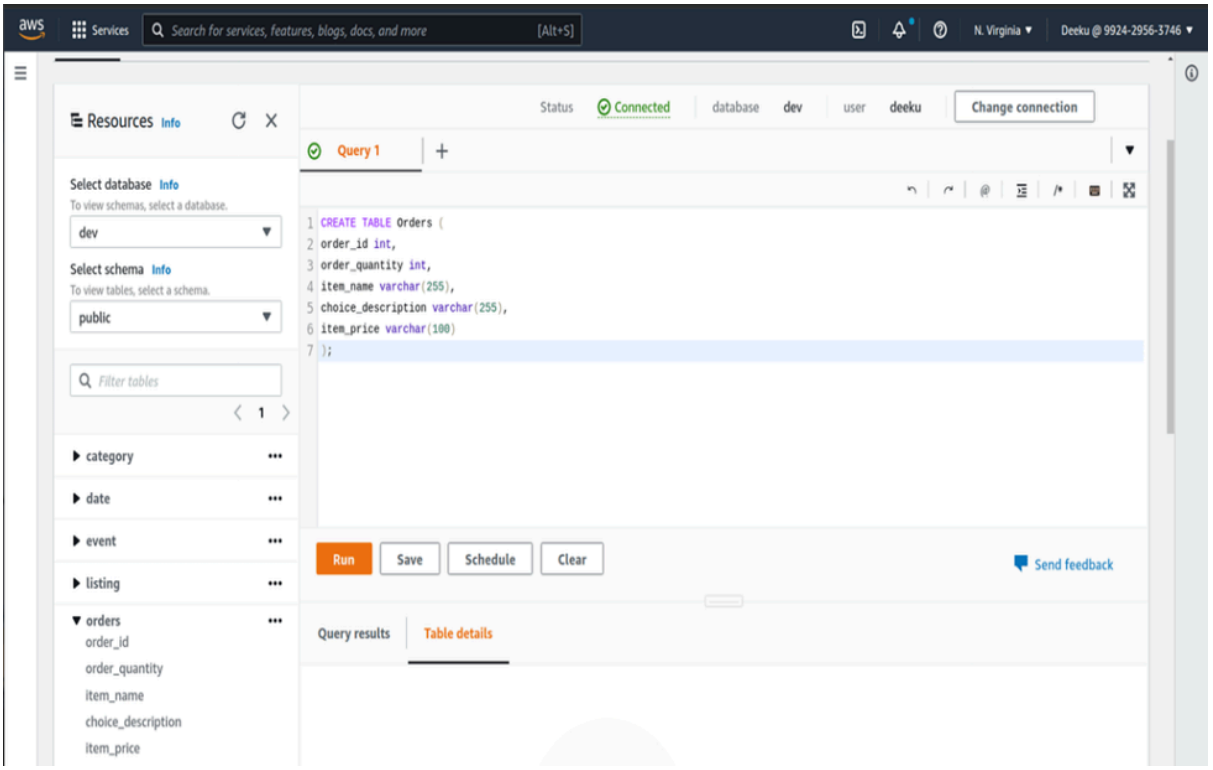

### 6. Copying Data from S3 to Redshift

- Use the **COPY command** in Redshift to load data from S3:

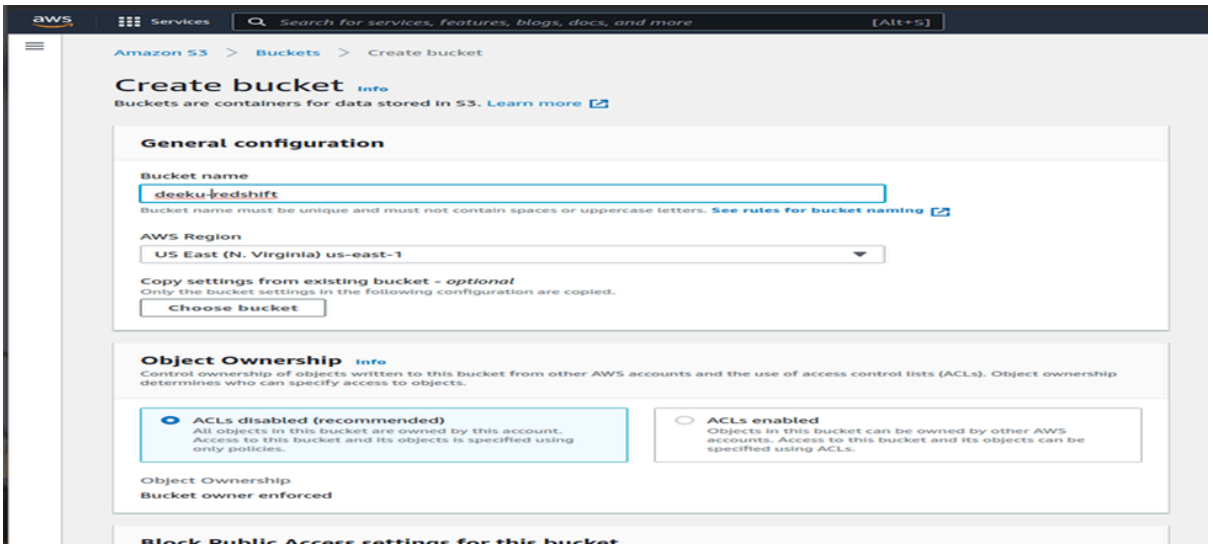
```
COPY Orders FROM 's3://your-bucket-name/orders.txt'
IAM_ROLE 'your-iam-role-arn'
REGION 'us-east-1'
IGNOREHEADER 1;
```



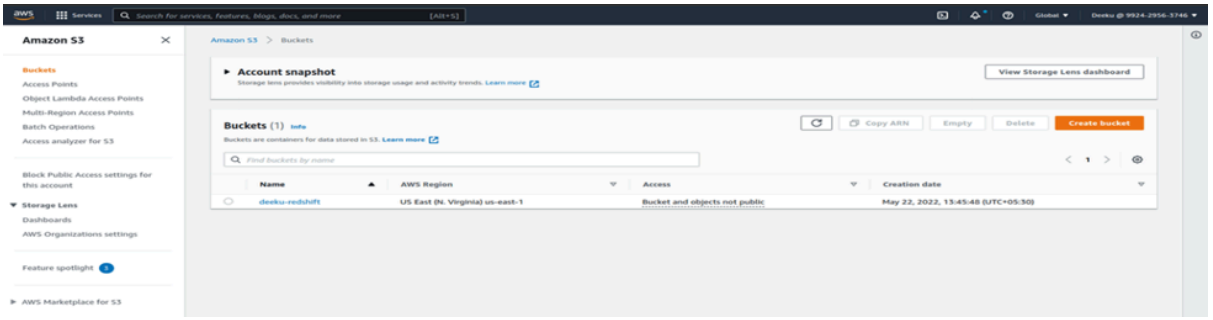

- Run the query in Query editor to create the table name *orders* like this



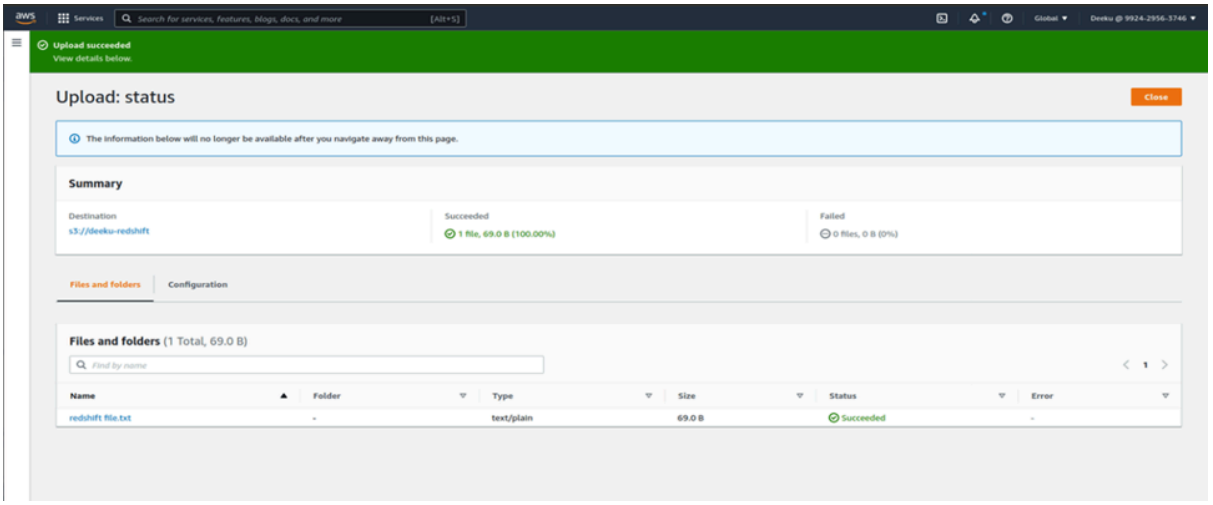
*Create S3 Bucket and upload txt file in that :*




- Once you have select create bucket option you will see , your bucket is created like this




- _Upload .txt file into that S3 Bucket as:



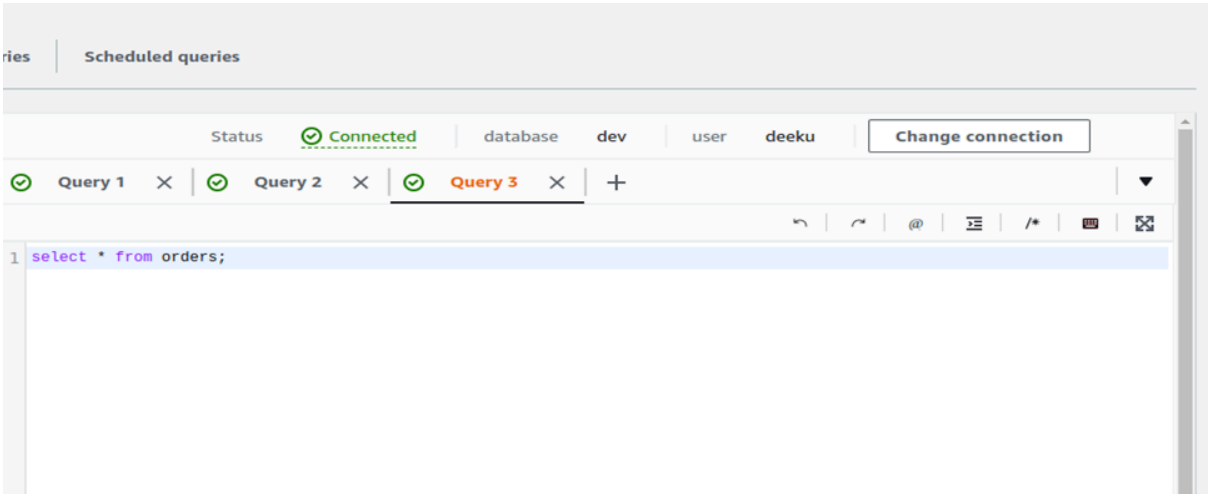
- We are at the final step. Now we have to copy data from S3 to Redshift use the following command:

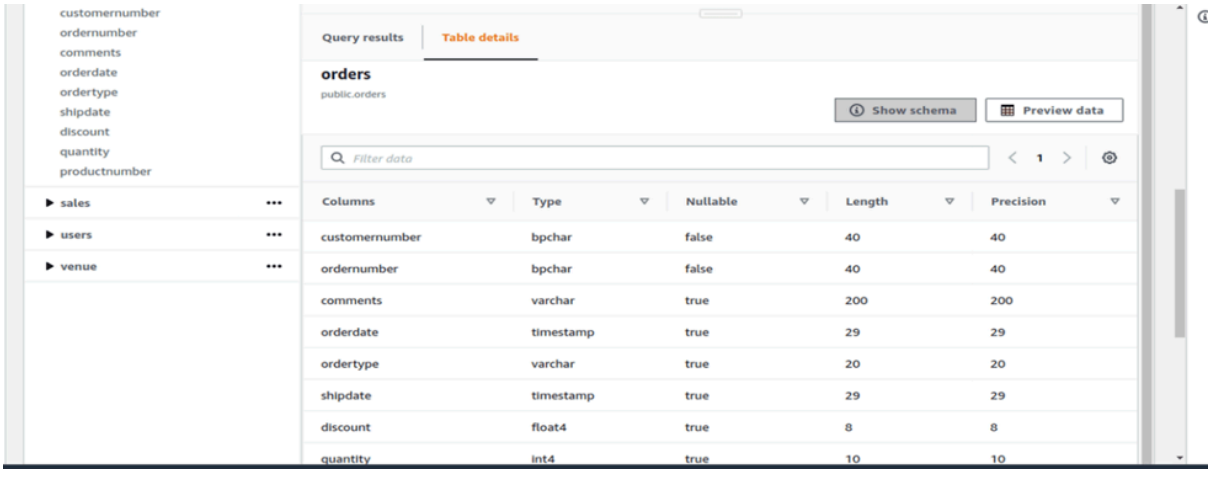
```
copy Orders from 's3://deeku-redshift/redshift
file.txt' iam_role ' ' region 'us- east-1'
IGNOREHEADER 1;
```

**7. Verifying Data Import**

- Run the following SQL query in **Redshift Query Editor** to verify that the data has been successfully imported:

```
SELECT * FROM Orders;
```




**RESULT:**

Thus, the **Amazon Redshift** was successfully connected to **S3**, and data was imported successfully.

| Ex.No:8 | **Creating and Querying a NoSQL Table in AWS DynamoDB** |
|---|---|
| | |

**AIM:**

To create, query, and manage a **NoSQL** database table using **AWS DynamoDB**.

**ALGORITHM**

**Step 1:** Log in to the **AWS Management Console**, navigate to **DynamoDB**, and click **Create Table**.

**Step 2:** Enter the **Table Name** and specify the **Primary Key** (Partition Key and optional Sort Key).

**Step 3:** Click **Create Table** and wait for the table to be available.

**Step 4:** Navigate to the **Items** section and click **Create Item** to insert new records.

**Step 5:** Enter data for each attribute based on the table schema and click **Save**.

**Step 6:** To query data, select the **Query** option from the drop-down menu below the **Create Item** section.

**Step 7:** Enter the **Partition Key** value to filter and retrieve specific records.

**Step 8:** To delete an item, switch the query mode to **Scan**, check the item's checkbox, click **Actions**, and select **Delete**.

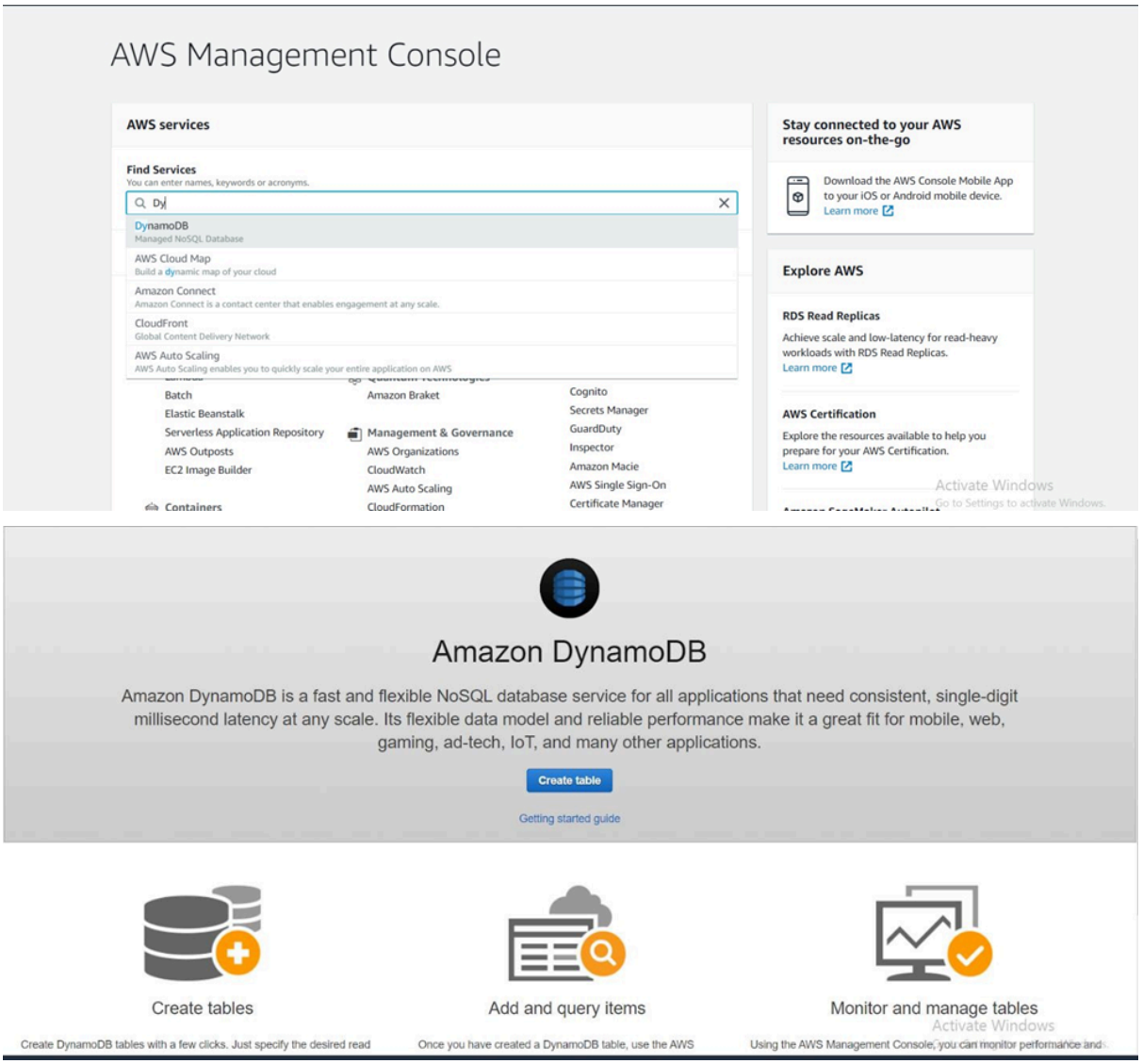
**Step 9:** To delete the entire table, go to the **DynamoDB Dashboard**, select the table, and click **Delete Table**.

**Step 10:** Verify the successful deletion of the table.

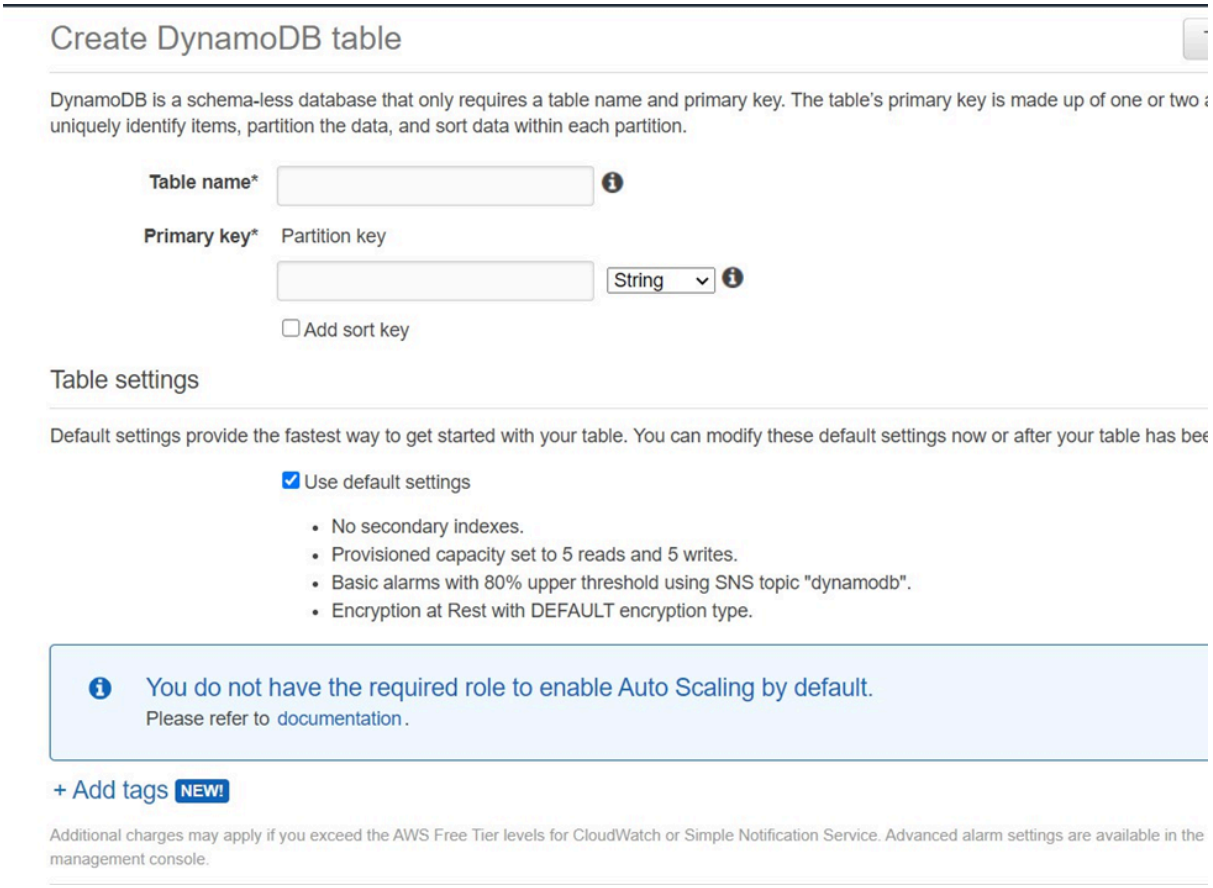
**PROGRAM :**

**1. Creating a DynamoDB Table**

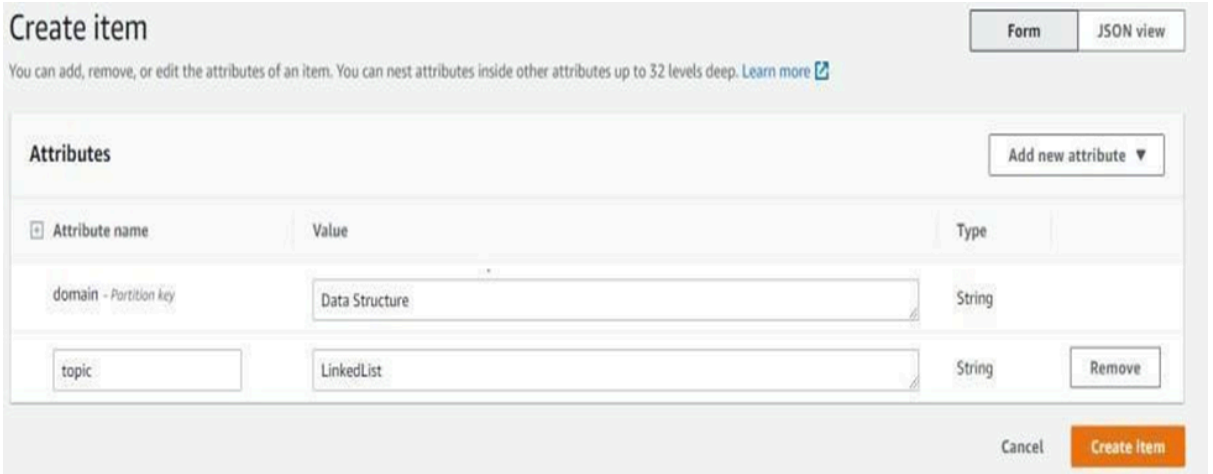
- Open the **AWS Management Console**, search for **DynamoDB**, and click **Create Table**.
- Enter the **Table Name** and specify the **Primary Key**.
- Click **Create Table** and wait for it to be available.




- Now, write your table name in the table name column, and the primary key of your table.

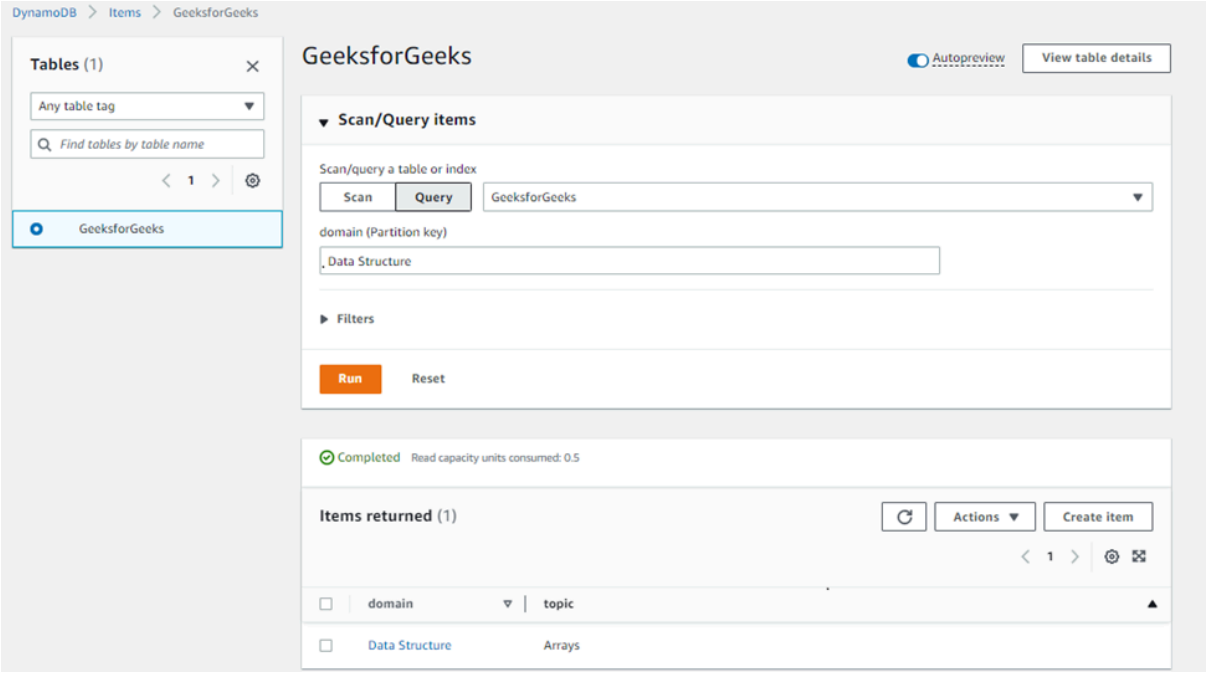



- Now click on the create table button
- Now go to the items section and click on create items.




## 2. Inserting Items into the Table

- Go to the **Items** section, click **Create Item**, and fill in the required data fields.
- Repeat this step to insert multiple records.
- Click **Save** to add the items.

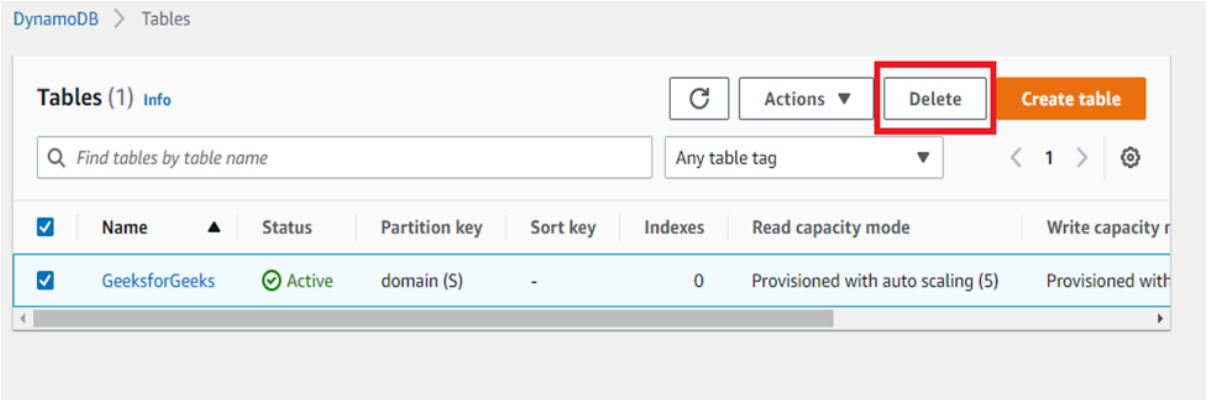



### 3. Querying the Table

- Below the **Create Item** section, select the **Query** option from the drop-down menu.
- Enter the **Partition Key** value to retrieve records that match the query.
- Click **Run** to view the results.

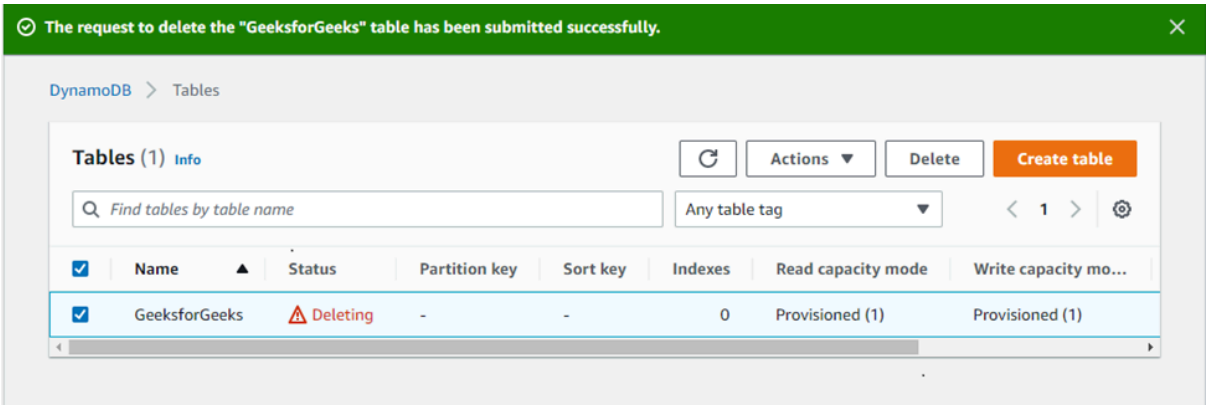
### 4. Deleting an Item from the Table

- Switch the **Query Mode** to **Scan**.
- Check the checkbox of the item to be deleted.
- Click **Actions**, then select **Delete Item**.




### 5. Deleting the DynamoDB Table

- Go to the **DynamoDB Dashboard** and click on the **Tables** tab.
- Select the table you want to delete and click **Delete Table**.
- Confirm deletion and verify that the table is removed.




**RESULT:**

Thus, a **NoSQL table** was successfully created, queried, and managed using **AWS DynamoDB**.