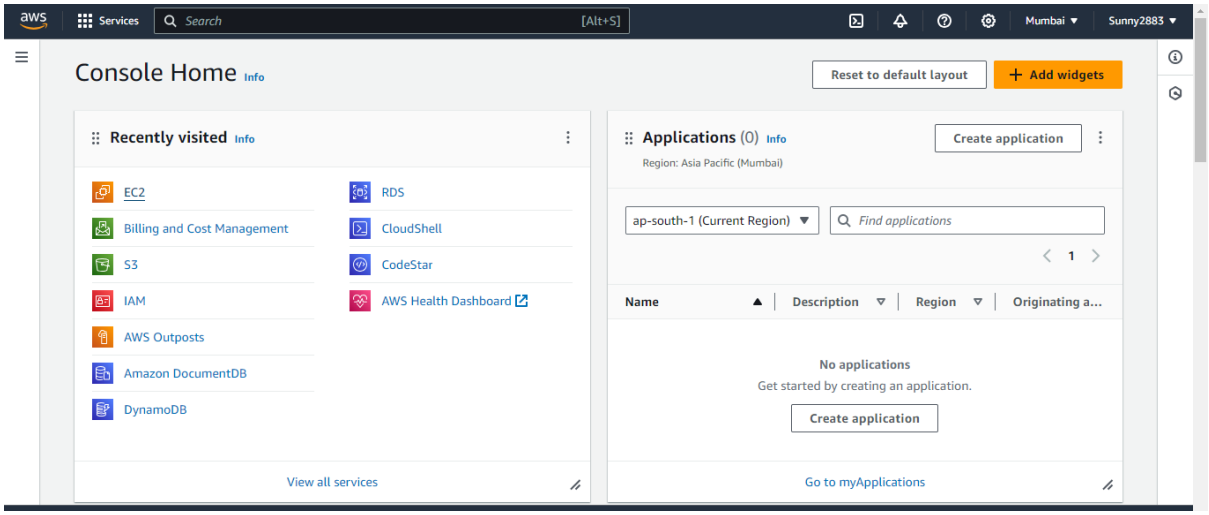
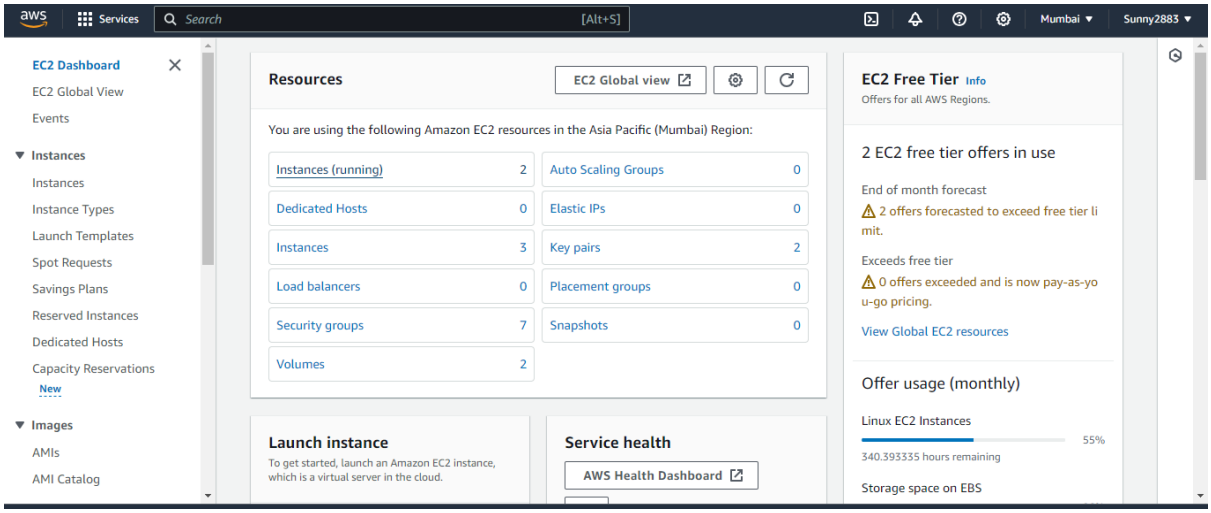
TASK 4: Have to create Linux backup with the use of S3 with specific requirements.

Requirements: Create a backup of the database, insert dummy data, and implement a retention rule to maintain backups for 30 days, with backups taken on the first day of every month and on the first day of every year.

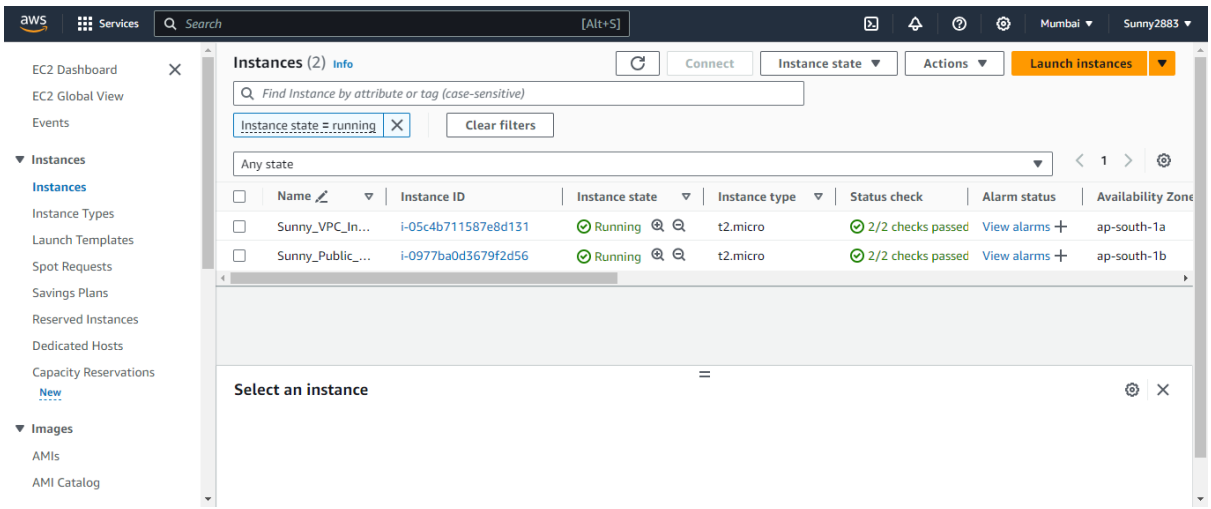
Step1: Go to the AWS Management Console and sign in using your AWS credentials.



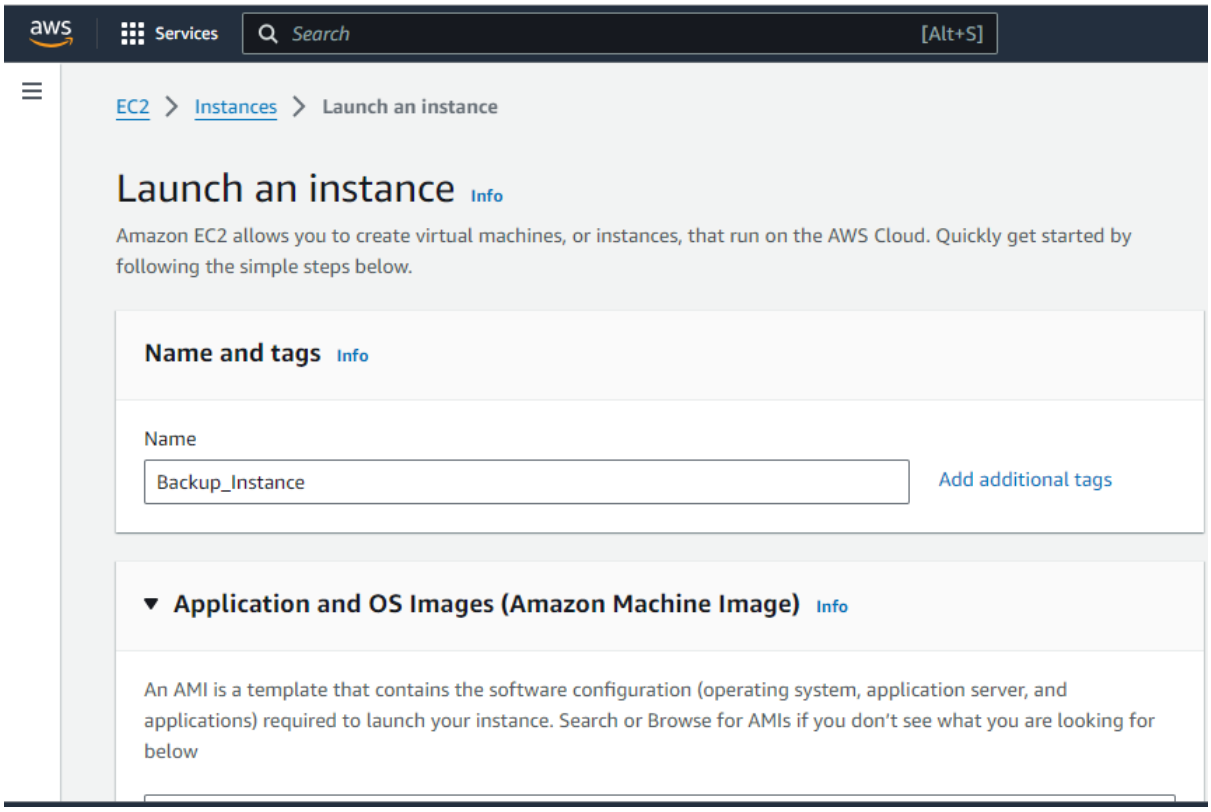
Step2: From the AWS Management Console, navigate to the EC2 service.



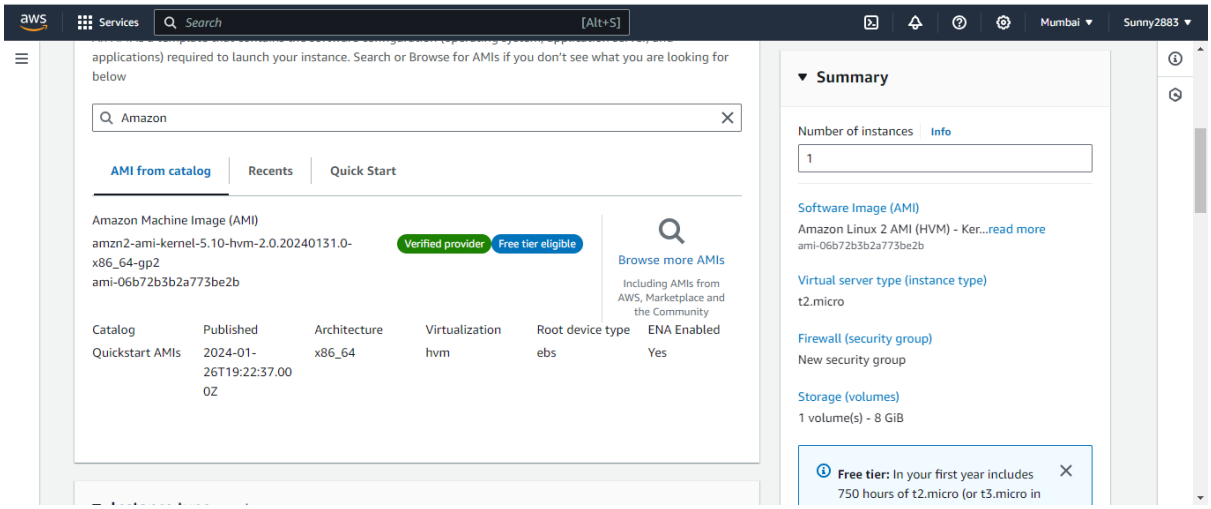
Step3: Click on the "Launch Instance" button to begin the process of creating a new EC2 instance.



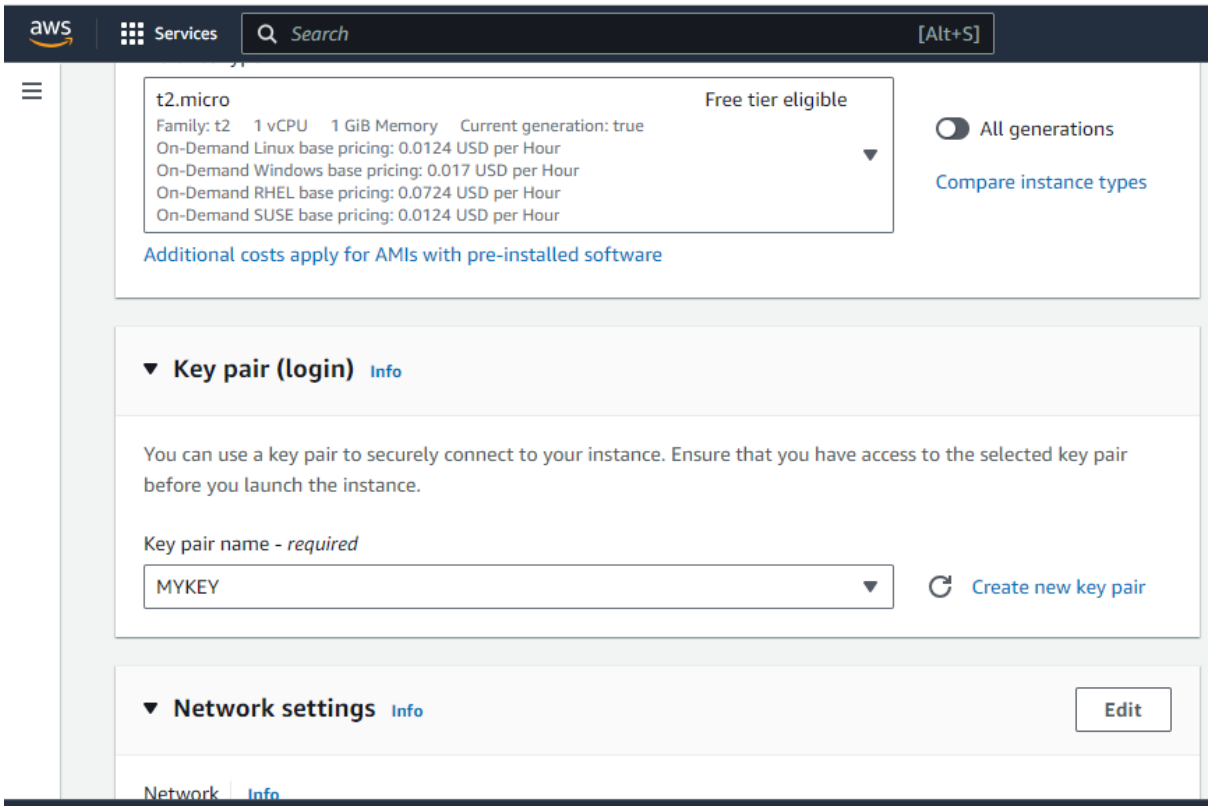
Step4: Give name and additional tag(optional) for your instance.



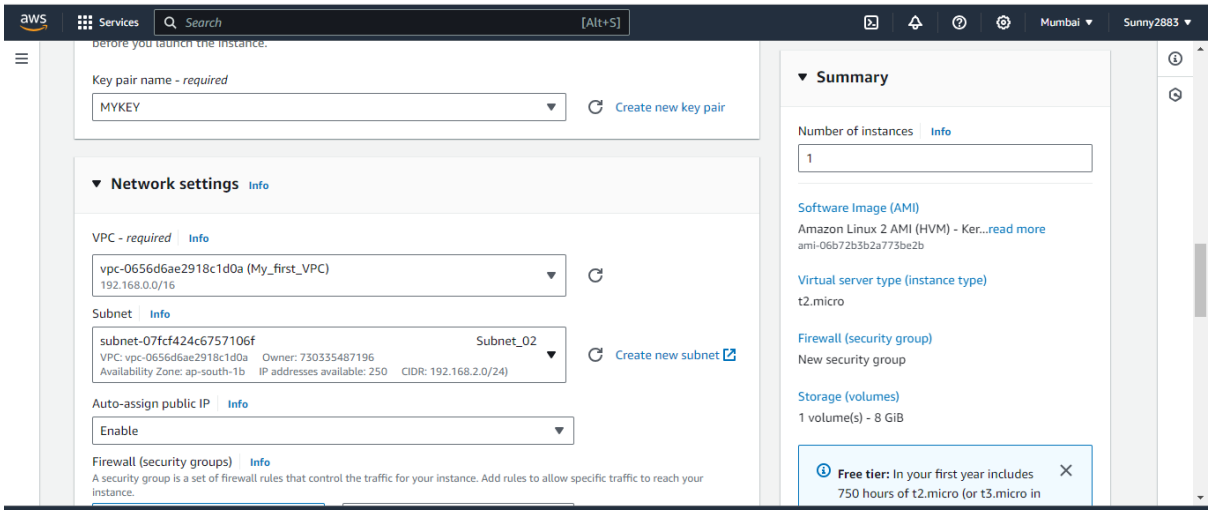
Step5: In the "Choose an Amazon Machine Image (AMI)" step, select "Amazon Linux 2 AMI". You can search for it in the search bar or navigate through the options.



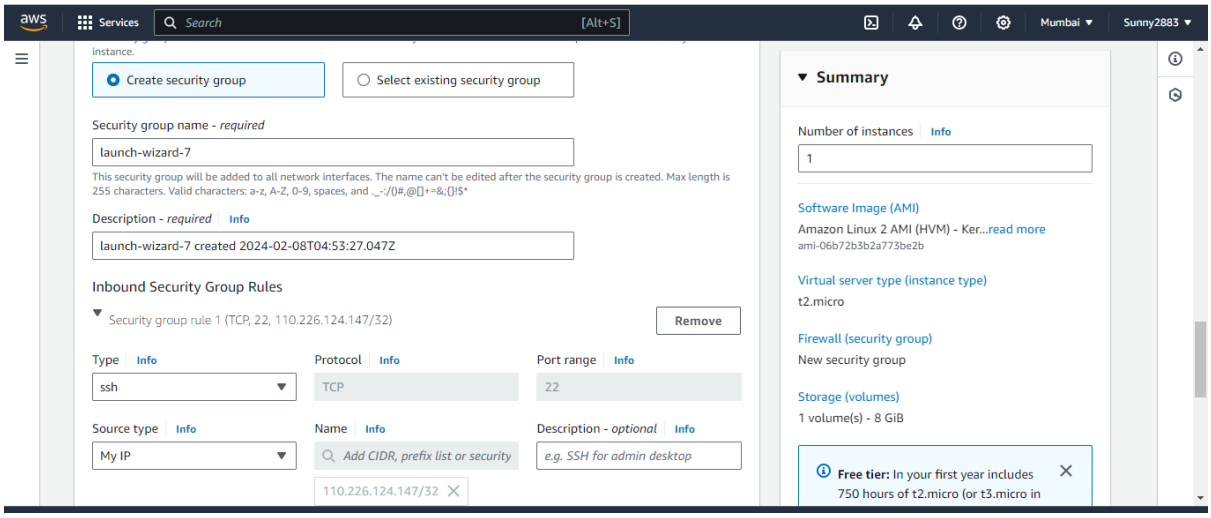
Step6: Select "t2.micro" as the instance type. This is a free tier eligible instance type.



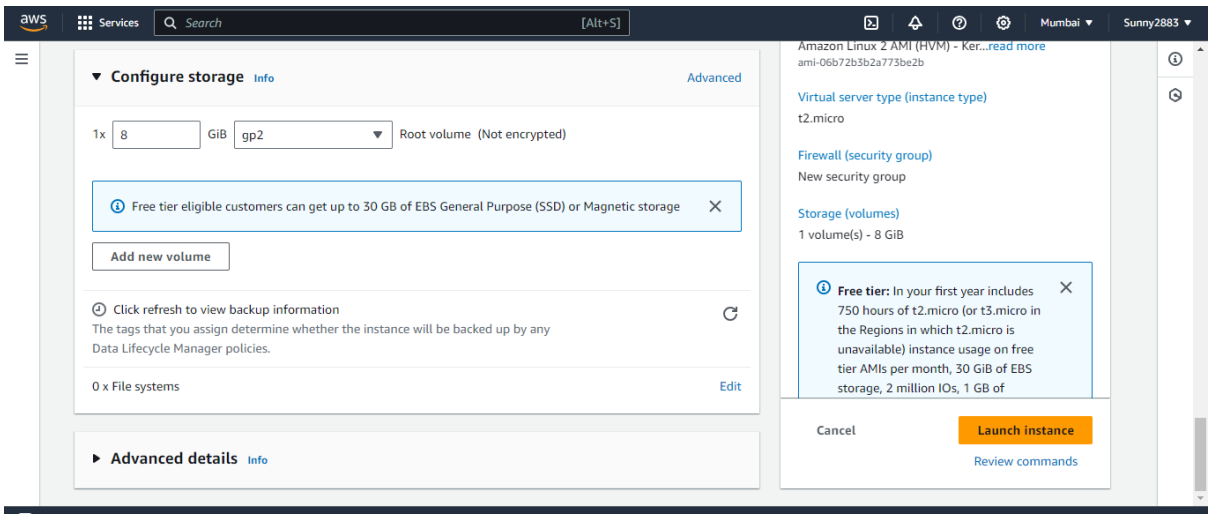
Step7: Select Key pair If you don't have an existing key pair, create a new one. This key pair will be used to securely connect to your EC2 instance via SSH.



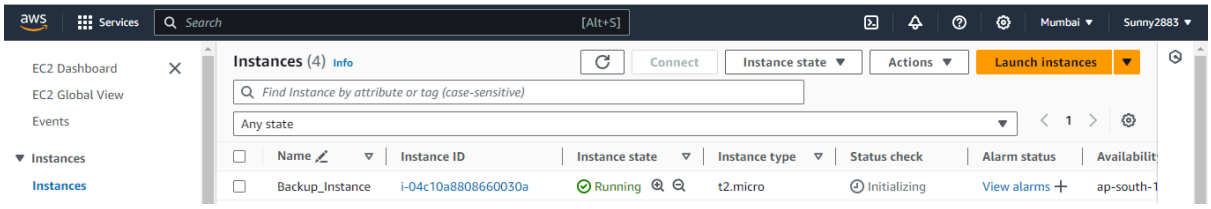
Step:8 Create a new security group or select an existing one. This group defines the firewall rules that control the traffic to your instance.



Step9: Review the configuration of your instance to ensure everything is set up as desired.Click "Launch" when you're ready.



Step10: You'll be redirected to the Instances dashboard where you can view the status of your newly launched instance.



Step11: Once the instance state transitions to "running", you can connect to it using SSH. You'll need the public IP address or public DNS name of the instance along with the private key associated with the key pair you selected.

ssh -i C:\Users\promact\Downloads\MYKEY.pem ec2-user@13.201.131.34

```
[ec2-user@ip-192-168-2-59 ~]$ ssh -i "C:\Users\promact\Downloads\MYKEY.pem" ec2-user@13.201.131.34
```

Step12:Once the connection is established you can install postgresql. PostgreSQL packages for Amazon Linux 2 are available in the PostgreSQL yum repository. Use the following commands to add the repository and install PostgreSQL 14:

sudo amazon-linux-extras install postgresql14

```
[ec2-user@ip-192-168-2-59 ~]$ sudo amazon-linux-extras install postresql14
```

Step13: Check version.

psql --version

```
[ec2-user@ip-192-168-2-59 ~]$ psql --version
psql (PostgreSQL) 14.10
```

Step14: Install PostgreSQL Server:

sudo yum install postgresql-server

```
[ec2-user@ip-192-168-2-59 ~]$ sudo yum install postgresql-server
```

Step15: Initialize PostgreSQL Database:

sudo postgresql-setup initdb

```
[ec2-user@ip-192-168-2-59 ~]$ sudo postgresql-setup initdb
```

Step16: Start PostgreSQL Service:

sudo systemctl start postgresql

```
sudo systemctl start postgresql-14
```

Step17: Enable PostgreSQL Service to Start on Boot:

sudo systemctl enable postgresql

```
sudo systemctl enable postgresql-14
```

Step18: First, access the PostgreSQL prompt by running

Sudo -u postgres psql

```
[ec2-user@ip-192-168-2-59 ~]$ sudo -u postgres psql
```

Step19: To create a new database, run the following command:

CREATE DATABASE backup;

```
[ec2-user@ip-192-168-2-59 ~]$ CREATE DATABASE backup;
```

Step20: After creating the database, connect to it using the \c command:

Step21: Now, create a table within the database.

CREATE TABLE student (

roll_number SERIAL PRIMARY KEY,

name VARCHAR(100),

class VARCHAR(20),

branch VARCHAR(50),

address TEXT,

gender VARCHAR(10),

university VARCHAR(100),

state VARCHAR(50),

sport VARCHAR(50),

language VARCHAR(50)

);

```
backup=> CREATE TABLE student (
backup(>     roll_number SERIAL PRIMARY KEY,
backup(>     name VARCHAR(100),
backup(>     class VARCHAR(20),
backup(>     branch VARCHAR(50),
backup(>     address TEXT,
backup(>     gender VARCHAR(10),
backup(>     university VARCHAR(100),
backup(>     state VARCHAR(50),
backup(>     sport VARCHAR(50),
backup(>     language VARCHAR(50)
backup(> );
CREATE TABLE
```

Step22: Use \dt to list the tables.

```
backup-> \dt
        List of relations
 Schema |  Name   | Type  | Owner
--------+---------+-------+-------
 public | student | table | sunny
(1 row)
```

Step23: You can now insert data into the table using the INSERT INTO command. Here's an example:

INSERT INTO student (name, class, branch, address, gender, university, state, sport, language) VALUES

('John Doe', '10th', 'Science', '123 Main St, City', 'Male', 'XYZ University', 'State1', 'Football', 'English'),

('Jane Smith', '12th', 'Commerce', '456 Elm St, Town', 'Female', 'ABC University', 'State2', 'Basketball', 'French'),

('Alice Johnson', '11th', 'Arts', '789 Oak St, Village', 'Female', 'PQR University', 'State3', 'Swimming', 'Spanish'),

('Bob Williams', '9th', 'Science', '101 Maple St, City', 'Male', 'XYZ University', 'State1', 'Tennis', 'English'),

('Emily Brown', '11th', 'Commerce', '202 Oak St, Town', 'Female', 'ABC University', 'State2', 'Soccer', 'French'),

('Michael Wilson', '10th', 'Arts', '303 Elm St, Village', 'Male', 'PQR University', 'State3', 'Cricket', 'Spanish'),

('Olivia Davis', '12th', 'Science', '404 Maple St, City', 'Female', 'XYZ University', 'State1', 'Badminton', 'English'),

('William Jones', '9th', 'Commerce', '505 Elm St, Town', 'Male', 'ABC University', 'State2', 'Table Tennis', 'French'),

('Emma Taylor', '11th', 'Arts', '606 Oak St, Village', 'Female', 'PQR University', 'State3', 'Hockey', 'Spanish'),

('Liam Martinez', '10th', 'Science', '707 Maple St, City', 'Male', 'XYZ University', 'State1', 'Chess', 'English'),

('Ava Hernandez', '12th', 'Commerce', '808 Elm St, Town', 'Female', 'ABC University', 'State2', 'Volleyball', 'French'),

('Noah Lopez', '9th', 'Arts', '909 Oak St, Village', 'Male', 'PQR University', 'State3', 'Athletics', 'Spanish'),

('Sophia Gonzales', '10th', 'Science', '111 Pine St, City', 'Female', 'XYZ University', 'State1', 'Golf', 'English'),

('Mason Perez', '12th', 'Commerce', '222 Cedar St, Town', 'Male', 'ABC University', 'State2', 'Boxing', 'French'),

('Isabella Adams', '9th', 'Arts', '333 Birch St, Village', 'Female', 'PQR University', 'State3', 'Martial Arts', 'Spanish'),

('Logan Cook', '10th', 'Science', '444 Pine St, City', 'Male', 'XYZ University', 'State1', 'Skating', 'English'),

('Amelia Green', '12th', 'Commerce', '555 Cedar St, Town', 'Female', 'ABC University', 'State2', 'Karate', 'French'),

('Benjamin Scott', '9th', 'Arts', '666 Birch St, Village', 'Male', 'PQR University', 'State3', 'Taekwondo', 'Spanish'),

('Elijah King', '10th', 'Science', '777 Pine St, City', 'Male', 'XYZ University', 'State1', 'Judo', 'English'),

('Charlotte Hill', '12th', 'Commerce', '888 Cedar St, Town', 'Female', 'ABC University', 'State2', 'Wrestling', 'French');

```
backup=>
backup=> INSERT INTO student (name, class, branch, address, gender, university, state, sport, language) VALUES
backup-> ('John Doe', '10th', 'Science', '123 Main St, City', 'Male', 'XYZ University', 'State1', 'Football', 'English'),
backup-> ('Jane Smith', '12th', 'Commerce', '456 Elm St, Town', 'Female', 'ABC University', 'State2', 'Basketball', 'French'),
backup-> ('Alice Johnson', '11th', 'Arts', '789 Oak St, Village', 'Female', 'PQR University', 'State3', 'Swimming', 'Spanish'),
backup-> ('Bob Williams', '9th', 'Science', '101 Maple St, City', 'Male', 'XYZ University', 'State1', 'Tennis', 'English'),
backup-> ('Emily Brown', '11th', 'Commerce', '202 Oak St, Town', 'Female', 'ABC University', 'State2', 'Soccer', 'French'),
backup-> ('Michael Wilson', '10th', 'Arts', '303 Elm St, Village', 'Male', 'PQR University', 'State3', 'Cricket', 'Spanish'),
backup-> ('Olivia Davis', '12th', 'Science', '404 Maple St, City', 'Female', 'XYZ University', 'State1', 'Badminton', 'English'),
backup-> ('William Jones', '9th', 'Commerce', '505 Elm St, Town', 'Male', 'ABC University', 'State2', 'Table Tennis', 'French'),
backup-> ('Emma Taylor', '11th', 'Arts', '606 Oak St, Village', 'Female', 'PQR University', 'State3', 'Hockey', 'Spanish'),
backup-> ('Liam Martinez', '10th', 'Science', '707 Maple St, City', 'Male', 'XYZ University', 'State1', 'Chess', 'English'),
backup-> ('Ava Hernandez', '12th', 'Commerce', '808 Elm St, Town', 'Female', 'ABC University', 'State2', 'Volleyball', 'French'),
backup-> ('Noah Lopez', '9th', 'Arts', '909 Oak St, Village', 'Male', 'PQR University', 'State3', 'Athletics', 'Spanish'),
backup-> ('Sophia Gonzales', '10th', 'Science', '111 Pine St, City', 'Female', 'XYZ University', 'State1', 'Golf', 'English'),
backup-> ('Mason Perez', '12th', 'Commerce', '222 Cedar St, Town', 'Male', 'ABC University', 'State2', 'Boxing', 'French'),
backup-> ('Isabella Adams', '9th', 'Arts', '333 Birch St, Village', 'Female', 'PQR University', 'State3', 'Martial Arts', 'Spanish'),
backup-> ('Logan Cook', '10th', 'Science', '444 Pine St, City', 'Male', 'XYZ University', 'State1', 'Skating', 'English'),
backup-> ('Amelia Green', '12th', 'Commerce', '555 Cedar St, Town', 'Female', 'ABC University', 'State2', 'Karate', 'French'),
backup-> ('Benjamin Scott', '9th', 'Arts', '666 Birch St, Village', 'Male', 'PQR University', 'State3', 'Taekwondo', 'Spanish'),
backup-> ('Elijah King', '10th', 'Science', '777 Pine St, City', 'Male', 'XYZ University', 'State1', 'Judo', 'English'),
backup-> ('Charlotte Hill', '12th', 'Commerce', '888 Cedar St, Town', 'Female', 'ABC University', 'State2', 'Wrestling', 'French');
INSERT 0 20
```

Step24: Use the pg_dump command to create a backup of your database. The basic syntax for pg_dump is:

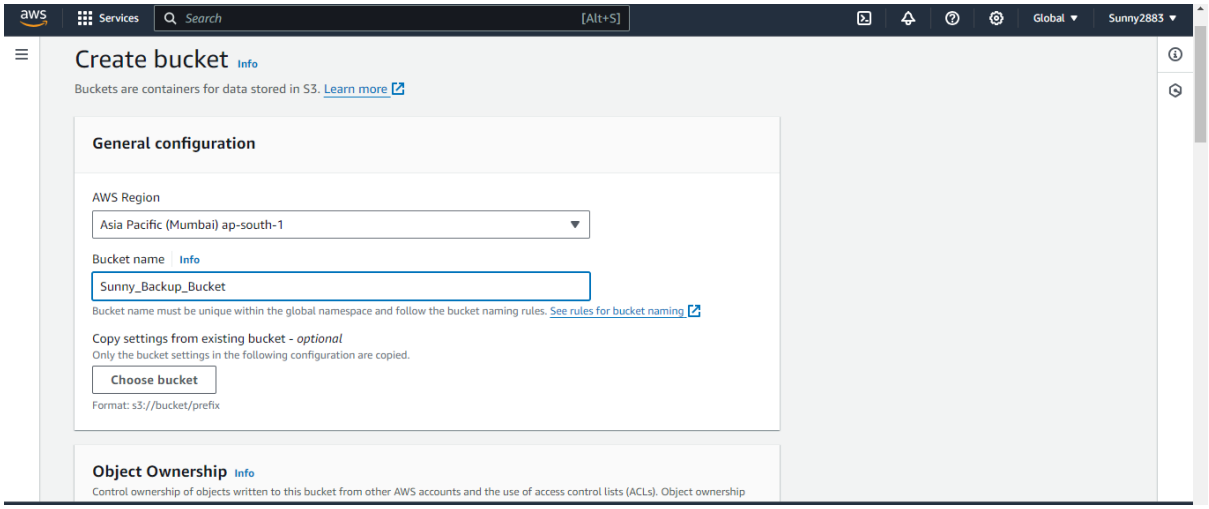
pg_dump -U sunny -d backup -h localhost > backup_file.sql

```
backup=> \q
[ec2-user@ip-192-168-2-59 ~]$ pg_dump -U sunny -d backup -h localhost > backup_file.sql
Password:
```

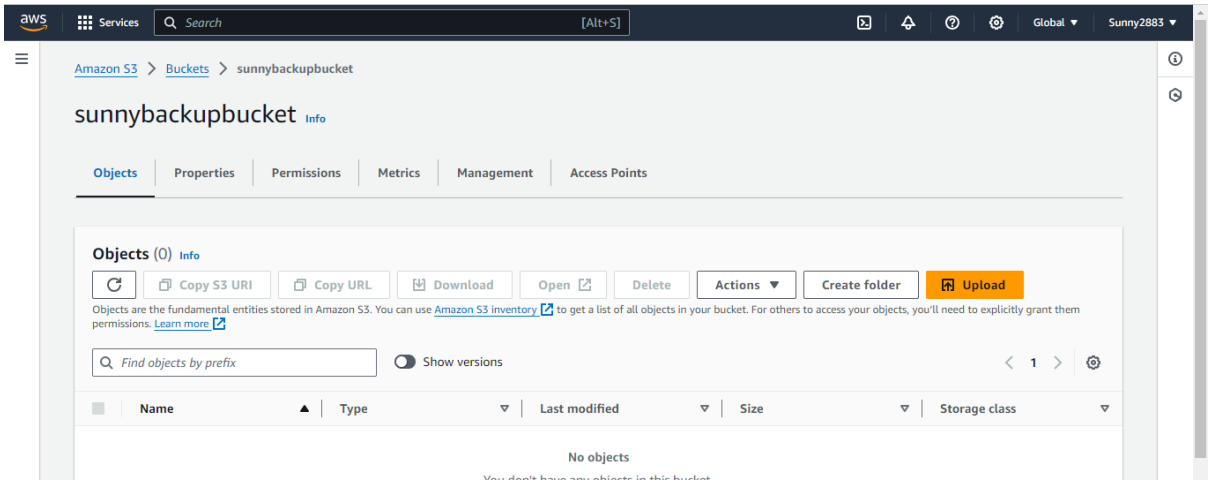
Step25:list the file using ls command;

```
Password:
[ec2-user@ip-192-168-2-59 ~]$ ls
backup_file.sql  backup.json  backup.sql
[ec2-user@ip-192-168-2-59 ~]$
```

Step26: If you haven't already created an S3 bucket, you can create one using the AWS Management Console or the AWS CLI. I have created using AWS management console.



Step27: Configure the bucket;



Step28: Once your S3 bucket is created, you can upload your PostgreSQL database backup file to the bucket using the aws s3 cp command. For example:

```
[ec2-user@ip-192-168-2-59 ~]$ aws s3 cp ~/backup_file.sql s3://sunnybackupbucket/
upload: ./backup_file.sql to s3://sunnybackupbucket/backup_file.sql
```

Step29: create a json file (backup.json) and add the retention rule for 30 days.

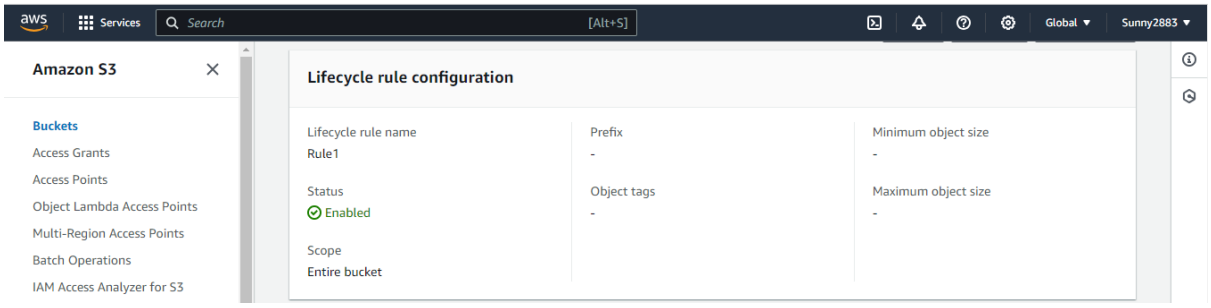
```
GNU nano 2.9.8                                                    backup.json

{
  "Rules": [
    {
      "ID": "Rule1",
      "Status": "Enabled",
      "Filter": {
        "Prefix": ""
      },
      "Expiration": {
        "Days": 30
      }
    }
  ]
}
```

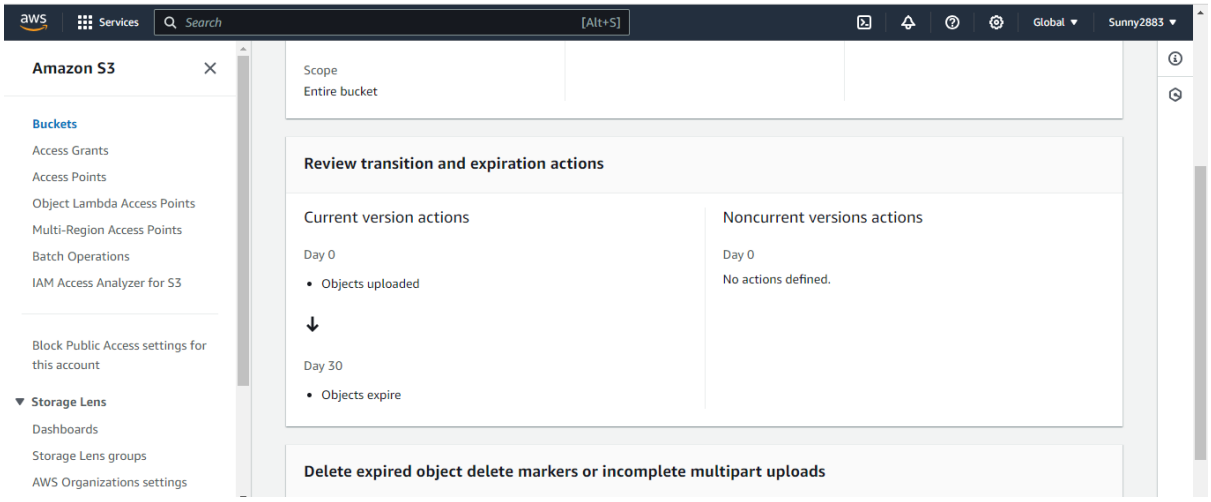
Step30: Run the command for the retention rule;

aws s3api put-bucket-lifecycle-configuration --bucket sunnybackupbucket --lifecycle-configuration file://backup.json

```
[ec2-user@ip-192-168-2-59 ~]$ aws s3api put-bucket-lifecycle-configuration --bucket sunnybackupbucket --lifecycle-configuration file://backup.json
```

Step31: Configure the retention rule:





Steps for "To automate the backup process, schedule it to run automatically on the first day of each month and the first day of each year."

Step1: Create a shell script file, and add the following content.

```
#!/bin/bash
perform_backup() {
   # Define database connection parameters
   db_user="sunny"
   db_name="backup"
   backup_file="/backup_$(date +'%Y%m%d').sql"
   sudo touch "$backup_file"
      sudo chmod 777 "$backup_file"
    pg_dump --dbname=postgresql://sunny:sunny2883@127.0.0.1:5432/backup -f "$backup_file"
```

```
    if [ $? -eq 0 ]; then
        echo "Database backup successful."

        aws s3 cp "$backup_file" s3://sunnybackupbucket/

        if [ $? -eq 0 ]; then
            echo "Backup file uploaded to S3 successfully."
        else
            echo "Failed to upload backup file to S3."
        fi
    else
        echo "Failed to create database backup."
    fi
}
```

```
#!/bin/bash

# Function to perform backup and upload to S3
perform_backup() {
    # Define database connection parameters
    db_user="sunny"
    db_name="backup"
    backup_file="/backup_$(date +'%Y%m%d').sql"
    sudo touch "$backup_file"
        sudo chmod 777 "$backup_file"
    # Perform database backup using pg_dump
     pg_dump --dbname=postgresql://sunny:sunny2883@127.0.0.1:5432/backup  -f "$backup_file"

    # Check if backup file was created successfully
    if [ $? -eq 0 ]; then
        echo "Database backup successful."

        # Upload backup file to S3 bucket
        aws s3 cp "$backup_file" s3://sunnybackupbucket/

        # Check if upload was successful
        if [ $? -eq 0 ]; then
            echo "Backup file uploaded to S3 successfully."
        else
            echo "Failed to upload backup file to S3."
        fi
    else
        echo "Failed to create database backup."
    fi
}
                                                    [ Read 40 lines ]
```

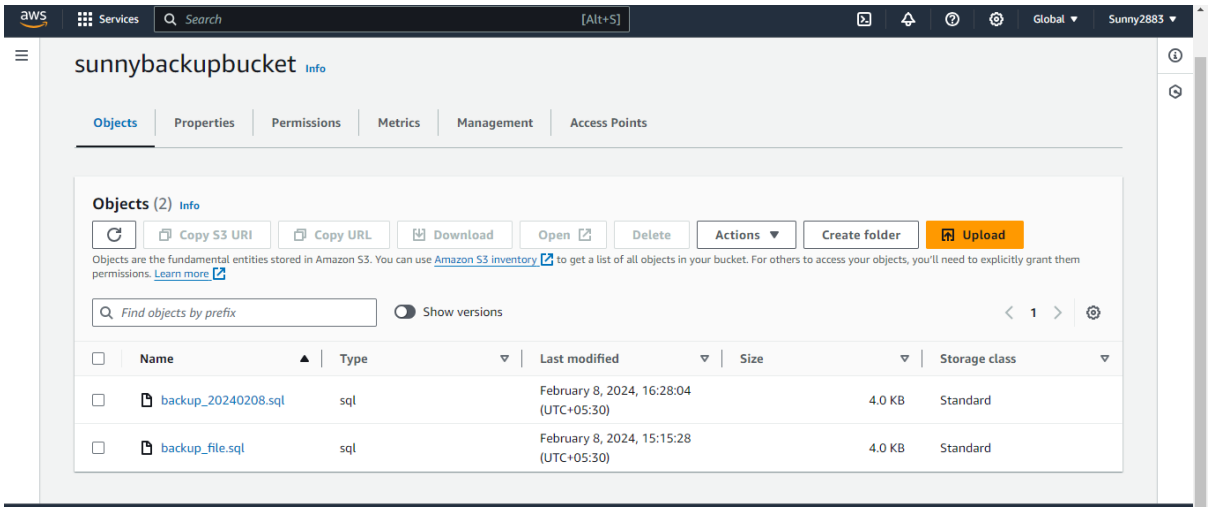
After executing the backup_script.sh file:

```
[ec2-user@ip-192-168-2-59 ~]$ nano backup_script.sh
[ec2-user@ip-192-168-2-59 ~]$ ./backup_script.sh
Database backup successful.
upload: ../../backup_20240208.sql to s3://sunnybackupbucket/backup_20240208.sql
Backup file uploaded to S3 successfully.
[ec2-user@ip-192-168-2-59 ~]$
```

Step2: Open the crontab file for editing Add a cron job for the first day of every month, Add a cron job for the first day of every year:

```
Command Prompt        ec2-user@ip-192-168-2-59:~
0 0 1 * * ~/backup_script.sh
0 0 1 1 * ~/backup_script.sh
~
~
~
~
~
```

Once the crontab file executed automatic backup will start:



ADDITIONAL REQUIRMENTS:

Additional project requirements: Download a new database from the internet, retrieve it, and save it to the instance. Create a backup file of the database and store it in an S3 bucket. Automate the backup process to occur every hour, and apply a retention rule to retain backups for one day.

*** For additional requirments I have created a different s3 bucket named **mybackupbucketsunny** and used same EC2 instance ***

Step1: Identify the Database:

Identify the specific database you want to download. This could be a database available online or provided by a source.

Step2: Download the Database File:

Use a tool like wget or curl to download the database file from the internet

scp -i C:\Users\promact\Downloads\MYKEY.pem "C:\Users\promact\Downloads\northwind.sql" ec2-user@13.201.131.34:/home/ec2-user

northwind.sql

```
[ec2-user@ip-192-168-2-59 ~]$ sudo psql -U sunny -d backup -f northwind.sql -h localhost
```

Step3: Restore the extracted file in to the database which is created using command:

sudo psql -U sunny -d backup -f northwind.sql -h localhost

```
[ec2-user@ip-192-168-2-59 ~]$ sudo psql -U sunny -d backup -f northwind.sql -h localhost
```

You are now connected to the database backup as user sunny:

```
backup=# \dt
                List of relations
 Schema |          Name          | Type  | Owner
--------+------------------------+-------+-------
 public | categories             | table | sunny
 public | customer_customer_demo | table | sunny
 public | customer_demographics  | table | sunny
 public | customers              | table | sunny
 public | employee_territories   | table | sunny
 public | employees              | table | sunny
 public | order_details          | table | sunny
 public | orders                 | table | sunny
 public | products               | table | sunny
 public | region                 | table | sunny
 public | shippers               | table | sunny
 public | suppliers              | table | sunny
 public | territories            | table | sunny
 public | us_states              | table | sunny
(14 rows)
```

Step4: Use the pg_dump command to create a backup of your database. The basic syntax for pg_dump is:

pg_dump -U sunny -d backup -h localhost >backup_file3.sql

```
[ec2-user@ip-192-168-2-59 ~]$ pg_dump -U sunny -d backup -h localhost >backup_file3.sql
```

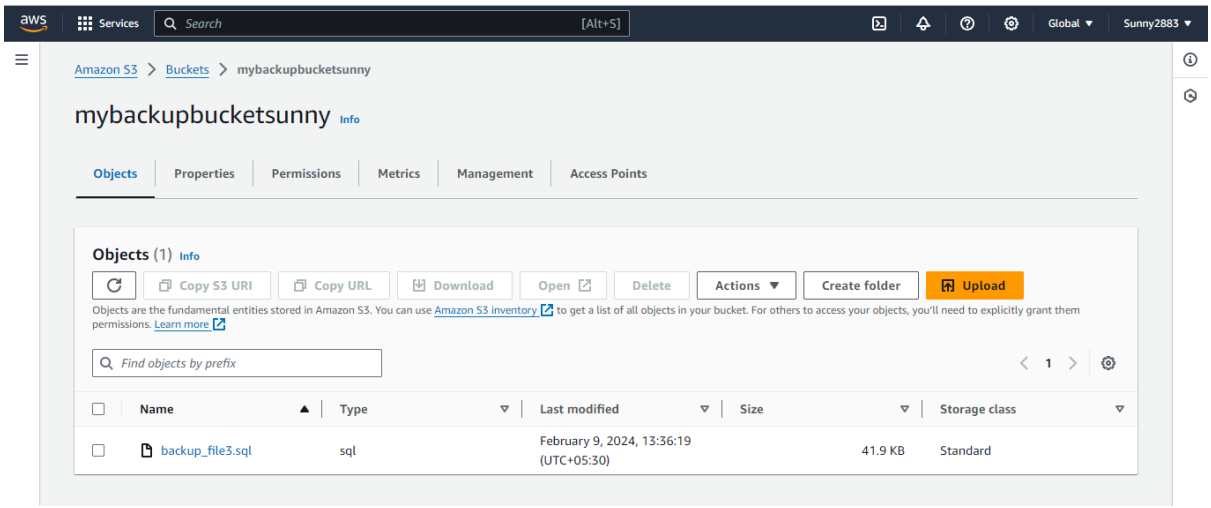
Step5: If your S3 bucket is already created, you can upload your PostgreSQL database backup file to the bucket using the AWS s3 cp command:

aws s3 cp ~/backup_file3.sql s3://mybackupbucketsunny/

```
ec2-user@ip-192-168-2-59 ~]$ aws s3 cp ~/backup_file3.sql s3://mybackupbucketsunny/
pload: ./backup_file3.sql to s3://mybackupbucketsunny/backup_file3.sql
ec2-user@ip-192-168-2-59 ~]$
```

Step6: Backup File Stored to S3 Bucket Successfully

Confirm that the backup file has been successfully stored in the S3 bucket.



Step7: Create a Backup_script3.sh File

Create a new shell script named backup_script3.sh using your preferred text editor. You can use the nano command to create an empty file and then edit it:

nano backup_script3.sh

```
[ec2-user@ip-192-168-2-59 ~]$ nano backup_script3.sh
```

Step8: Change Permissions of Backup_script2.sh File

Change the permissions of the backup_script2.sh file to make it executable. Use the chmod command to modify the file permissions:

sudo chmod 777 backup_script3.sh

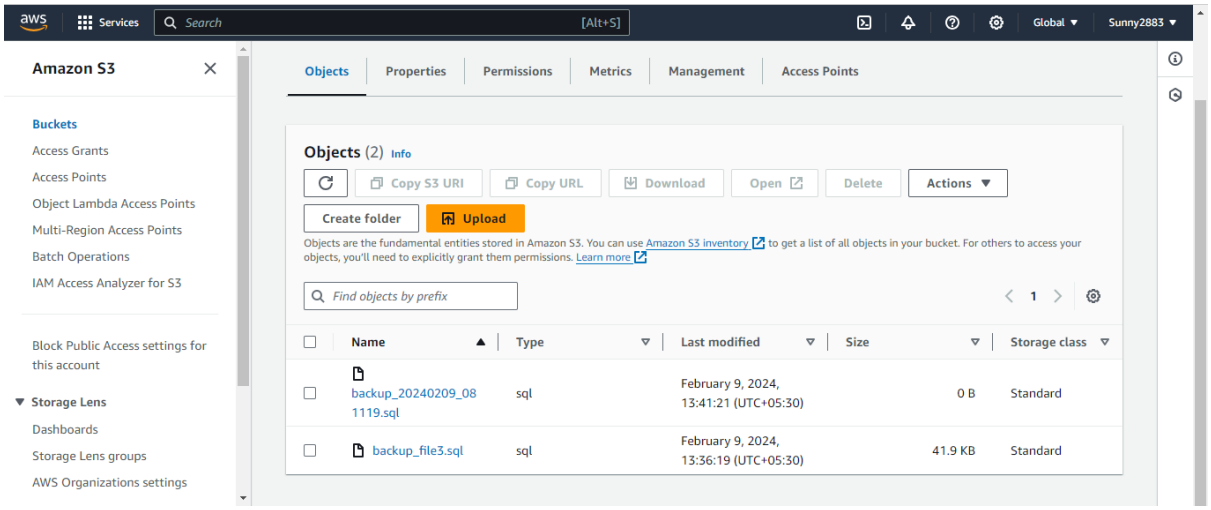
```
[ec2-user@ip-192-168-2-59 ~]$ sudo chmod 777 backup_script3.sh
```

Step9: Execute backup_script2.sh file.

./backup_script3.sh

```
[ec2-user@ip-192-168-2-59 ~]$ ./backup_script3.sh
```

Backup file stored to s3 successfully.



Step10: write a shell script:

nano backup_script.sh

```
#!/bin/bash
perform_backup() {
 # Define database connection parameters
 db_user="sunny"
 db_name="backup"
 backup_file3="/backup_$(date +'%Y%m%d_%H%M%S').sql"
 sudo touch "$backup_file3"
 sudo chmod 777 "$backup_file3"
 pg_dump --
dbname=postgresql://sunny:pswd@127.0.0.1:5432/$db_name -f
"$backup_file3"
 if [ $? -eq 0 ]; then
 echo "Database backup successful."
 aws s3 cp "$backup_file3" s3://mybackupbucketsunny/
 if [ $? -eq 0 ]; then
```

echo "Backup file uploaded to S3 successfully."

else

echo "Failed to upload backup file to S3."

fi

else

echo "Failed to create database backup."

fi

}

perform_backup

```
GNU nano 2.9.8                                    backup_script3.sh

#!/bin/bash
perform_backup() {
 # Define database connection parameters
 db_user="sunny"
 db_name="backup"
 backup_file3="/backup_$(date +'%Y%m%d_%H%M%S').sql"
 sudo touch "$backup_file3"
 sudo chmod 777 "$backup_file3"
 pg_dump --
dbname=postgresql://sunny:[redacted]@0.0.1:5432/$db_name -f
"$backup_file3"
 if [ $? -eq 0 ]; then
 echo "Database backup successful."
 aws s3 cp "$backup_file3" s3://mybackupbucketsunny/
 if [ $? -eq 0 ]; then
 echo "Backup file uploaded to S3 successfully."
 else
 echo "Failed to upload backup file to S3."
 fi
 else
 echo "Failed to create database backup."
 fi
}
perform_backup
```

Step11: Execute the file:

./backup_script3.sh

```
[ec2-user@ip-192-168-2-59 ~]$ ./backup_script3.sh
```

Step12: Create a backup.json file for retention of 1 day:

```
[ec2-user@ip-192-168-2-59 ~]$ nano backup3.json
```

{

"Rules": [

```
    {
      "Status": "Enabled",
      "ID": "BackupRetentionRule",
      "Filter": {
        "Prefix": ""
      },
      "Expiration": {
        "Days": 1
      }
    }
  ]
}
```

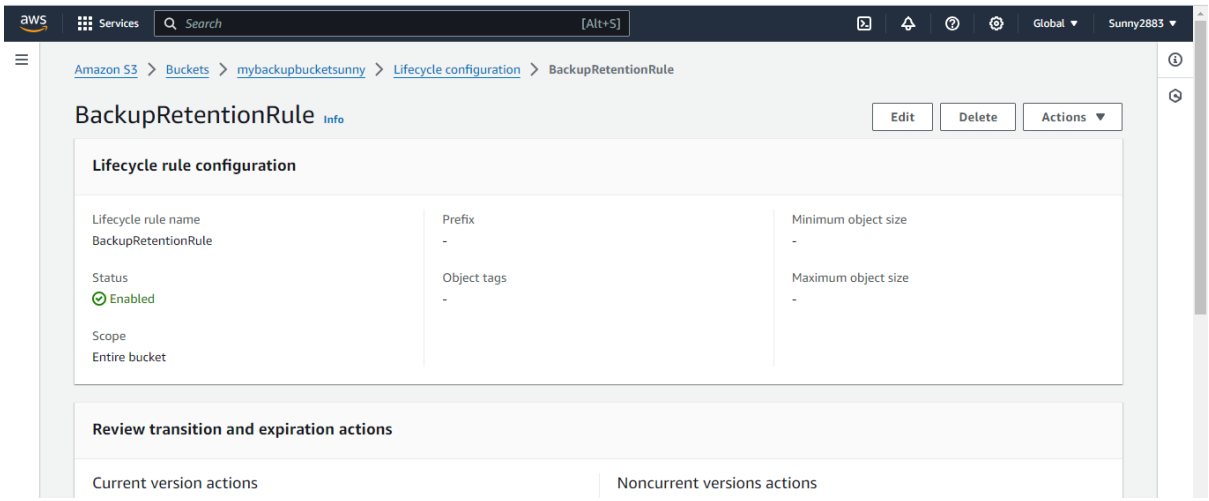
```
{
  "Rules": [
    {
      "Status": "Enabled",
      "ID": "BackupRetentionRule",
      "Filter": {
        "Prefix": ""
      },
      "Expiration": {
        "Days": 1
      }
    }
  ]
}
```

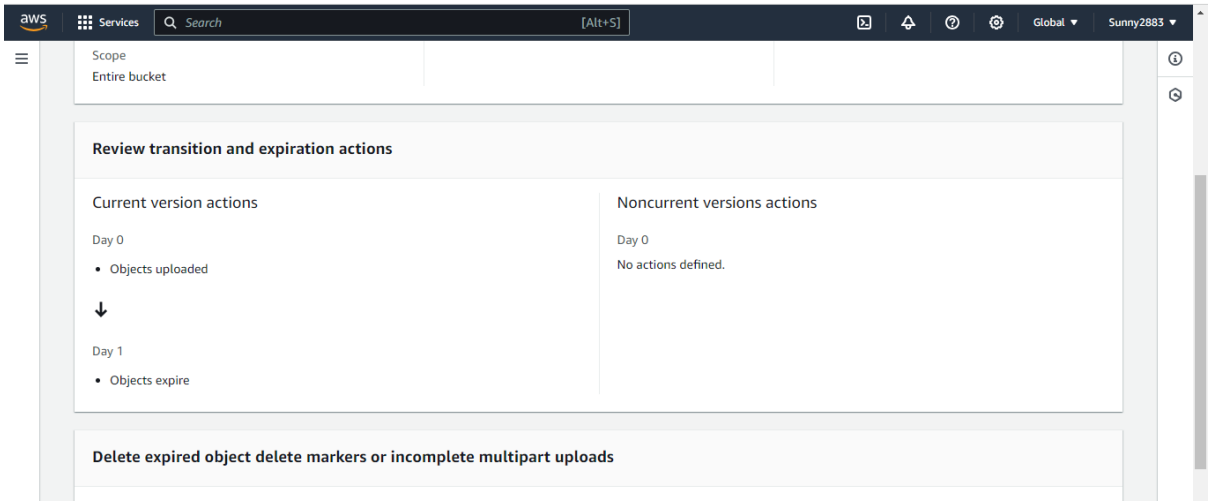
Step13: Run the command for the retention rule;

aws s3api put-bucket-lifecycle-configuration --bucket mybackupbucketsunny --lifecycle-configuration file://backup3.json

```
[ec2-user@ip-192-168-2-59 ~]$ aws s3api put-bucket-lifecycle-configuration --bucket mybackupbucketsunny --lifecycle-configuration file://backup3.jso
n
```

Retention rule for 1 day applied:





Step14: Open the crontab file for editing Add a cron job for every hour :

```
[ec2-user@ip-192-168-2-59 ~]$ crontab -e
crontab: installing new crontab
```

```
0 * * * * /home/ec2-user/backup_script3.sh
~
~
~
```

The backup process is set to occur automatically every hour, ensuring that backups are performed consistently and at regular intervals.