In [63]:
```
import boto3
import botocore
from botocore.config import Config
import getpass
import snowflake.connector
import pandas as pd
import os
from io import StringIO
import csv
```

In [64]:
```
os.getcwd()
```

Out[64]:
```
'C:\\Users\\Anand Jha\\Downloads'
```

In [75]:
```
# Set up AWS credentials manually (only for testing)
aws_access_key_id = 'AKIAZUFV73CQGGRUVW5W'
aws_secret_access_key = 'VhjVDY2JFAVx0vR8f1Hypd2FCzqLMp8d9eNh1ngh'
region_name = 'us-east-1'  # Replace with your region
```

In [76]:
```
# Create a session using the manual credentials
session = boto3.Session(
    aws_access_key_id=aws_access_key_id,
    aws_secret_access_key=aws_secret_access_key,
    region_name=region_name
)
```

In [77]:
```
# Create an S3 client
s3 = session.client('s3')
```

Listing All Buckets

In [78]:
```
# Now you can use the S3 client to perform operations
response = s3.list_buckets()
print(response)
```

```
{'ResponseMetadata': {'RequestId': '9YJ4JRG6JV6GXTQW', 'HostId': 'q751VE5Ma6EDNaZ
QpQjDGqiiyoYkcvMiDGmtU/os7/CHxLsVpIJOgqGym0wphu6KU6dPVhxPJjg=', 'HTTPStatusCode':
200, 'HTTPHeaders': {'x-amz-id-2': 'q751VE5Ma6EDNaZQpQjDGqiiyoYkcvMiDGmtU/os7/CHx
LsVpIJOgqGym0wphu6KU6dPVhxPJjg=', 'x-amz-request-id': '9YJ4JRG6JV6GXTQW', 'date':
'Tue, 10 Dec 2024 10:24:32 GMT', 'content-type': 'application/xml', 'transfer-enc
oding': 'chunked', 'server': 'AmazonS3'}, 'RetryAttempts': 0}, 'Buckets': [{'Nam
e': 'aj-calender', 'CreationDate': datetime.datetime(2024, 7, 10, 21, 17, 33, tzi
nfo=tzutc())}, {'Name': 'aj-iotv2', 'CreationDate': datetime.datetime(2024, 2, 1
1, 21, 56, 10, tzinfo=tzutc())}, {'Name': 'aj-northwind-data', 'CreationDate': da
tetime.datetime(2023, 10, 28, 19, 57, 19, tzinfo=tzutc())}, {'Name': 'czec-bankin
g', 'CreationDate': datetime.datetime(2024, 2, 18, 12, 56, 39, tzinfo=tzutc())},
{'Name': 'mat-pi', 'CreationDate': datetime.datetime(2024, 7, 28, 16, 9, 52, tzin
fo=tzutc())}, {'Name': 'matillionclass', 'CreationDate': datetime.datetime(2024,
12, 4, 9, 0, 29, tzinfo=tzutc())}, {'Name': 'matillionprac', 'CreationDate': date
time.datetime(2024, 7, 23, 11, 8, 19, tzinfo=tzutc())}, {'Name': 'parsing-xml-fil
e', 'CreationDate': datetime.datetime(2024, 5, 30, 10, 9, 16, tzinfo=tzutc())},
{'Name': 'restaurant2024', 'CreationDate': datetime.datetime(2024, 8, 12, 11, 29,
25, tzinfo=tzutc())}, {'Name': 'sarangmybucket', 'CreationDate': datetime.datetim
e(2024, 10, 31, 9, 31, 38, tzinfo=tzutc())}, {'Name': 'sdggdggssdgag', 'CreationD
ate': datetime.datetime(2024, 2, 18, 16, 20, 3, tzinfo=tzutc())}, {'Name': 'ss-pr
actice-matillion-revision', 'CreationDate': datetime.datetime(2024, 11, 1, 1, 54,
54, tzinfo=tzutc())}, {'Name': 'ssbucketdemo', 'CreationDate': datetime.datetime
(2024, 10, 5, 6, 57, 27, tzinfo=tzutc())}, {'Name': 'videosummarizer', 'CreationD
ate': datetime.datetime(2024, 3, 11, 19, 23, 4, tzinfo=tzutc())}], 'Owner': {'Dis
playName': 'info', 'ID': '10954629e3dfaabc0680ca7e878fc30c5affd71bd871a8f63bdb28f
675a874d9'}}
```

In [ ]:

Creating a bucket

In [80]:
```python
s3.create_bucket(Bucket='s3demobucketpy')
```

Out[80]:
```
{'ResponseMetadata': {'RequestId': 'P5SAPGMBYSZTQGB5',
  'HostId': 'C1P2I7P7B8qxSdBrzOy4A+uzYqNZeKhTNK4C4uiNrUULYSd4RwN4b4ludXhtsZiYrm
40s+FZn49Xnb+p+KN1AOHnsFf3mZo9',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amz-id-2': 'C1P2I7P7B8qxSdBrzOy4A+uzYqNZeKhTNK4C4uiNrUULYS
d4RwN4b4ludXhtsZiYrm40s+FZn49Xnb+p+KN1AOHnsFf3mZo9',
   'x-amz-request-id': 'P5SAPGMBYSZTQGB5',
   'date': 'Tue, 10 Dec 2024 10:25:01 GMT',
   'location': '/s3demobucketpy',
   'content-length': '0',
   'server': 'AmazonS3'},
  'RetryAttempts': 0},
 'Location': '/s3demobucketpy'}
```

Accessing a specific bucket

In [82]:
```python
# Specify the name of your S3 bucket
bucket_name = 'ssbucketdemo'

# List all objects in the specific S3 bucket
response = s3.list_objects_v2(Bucket=bucket_name)

# Print object keys (file names)
```

```
if 'Contents' in response:
    for obj in response['Contents']:
        print(f"Object Key: {obj['Key']}")
else:
    print("No objects found in the bucket.")
```

```
Object Key: NETFLIX/
Object Key: NETFLIX/netflix_titles.csv
Object Key: NHANES/
Object Key: NHANES/NHANES.csv
Object Key: bank_trnx/
Object Key: bank_trnx/tranx_new_csv.csv
Object Key: bank_trnx/trnx_16.csv
Object Key: bank_trnx/trnx_17.csv
Object Key: bank_trnx/trnx_18.csv
Object Key: bank_trnx/trnx_19_NEW.csv
Object Key: bank_trnx/trnx_20_NEW.csv
Object Key: bank_trnx/trnx_21_NEW.csv
Object Key: transactions/
Object Key: transactions/trnx177_353.csv
Object Key: transactions/trnx1_176.csv
Object Key: transactions/trnx354_530.csv
Object Key: transactions/trnx531_711.csv
```

Upload file in a specific bucket

In [85]:
```
# Specify the name of your S3 bucket and the file to upload
bucket_name = 'ssbucketdemo'
file_name = 'C:/Users/Anand Jha/Downloads/sales_data_df2.csv'
s3_object_name = 'NHANES/sales_data_df2.csv'  # This is the key in S3 or specifi

# Upload the file to the specified bucket
s3.upload_file(file_name, bucket_name, s3_object_name)

print(f"File '{file_name}' uploaded to S3 bucket '{bucket_name}' as '{s3_object_
```

```
File 'C:/Users/Anand Jha/Downloads/sales_data_df2.csv' uploaded to S3 bucket 'ssb
ucketdemo' as 'NHANES/sales_data_df2.csv'.
```

Download file from a specific bucket

In [92]:
```
# Specify the name of your S3 bucket and the file to download
bucket_name = 'ssbucketdemo'
s3_object_name = 'NHANES/sales_data_df2.csv'  # The key in S3
download_path = 'E:/Tiger_Analytics/BOTOS3'  # Local path to save the downloaded

# Download the file from the S3 bucket
s3.download_file(bucket_name, s3_object_name, download_path)

print(f"File '{s3_object_name}' downloaded from S3 bucket '{bucket_name}' to '{d
---------------------------------------------------------------------------
PermissionError                           Traceback (most recent call last)
Cell In[92], line 7
      4 download_path = 'E:/Tiger_Analytics/BOTOS3'  # Local path to save the dow
nloaded file
      6 # Download the file from the S3 bucket
----> 7 s3.download_file(bucket_name, s3_object_name, download_path)
      9 print(f"File '{s3_object_name}' downloaded from S3 bucket '{bucket_name}'
to '{download_path}'.")

File ~\anaconda3\Lib\site-packages\boto3\s3\inject.py:190, in download_file(self,
Bucket, Key, Filename, ExtraArgs, Callback, Config)
    155 """Download an S3 object to a file.
    156
    157 Usage::
   (...)
    187     transfer.
    188 """
    189 with S3Transfer(self, Config) as transfer:
--> 190     return transfer.download_file(
    191         bucket=Bucket,
    192         key=Key,
    193         filename=Filename,
    194         extra_args=ExtraArgs,
    195         callback=Callback,
    196     )

File ~\anaconda3\Lib\site-packages\boto3\s3\transfer.py:320, in S3Transfer.downlo
ad_file(self, bucket, key, filename, extra_args, callback)
    316 future = self._manager.download(
    317     bucket, key, filename, extra_args, subscribers
    318 )
    319 try:
--> 320     future.result()
    321 # This is for backwards compatibility where when retries are
    322 # exceeded we need to throw the same error from boto3 instead of
    323 # s3transfer's built in RetriesExceededError as current users are
    324 # catching the boto3 one instead of the s3transfer exception to do
    325 # their own retries.
    326 except S3TransferRetriesExceededError as e:

File ~\anaconda3\Lib\site-packages\s3transfer\futures.py:103, in TransferFuture.r
esult(self)
     98 def result(self):
     99     try:
    100         # Usually the result() method blocks until the transfer is done,
    101         # however if a KeyboardInterrupt is raised we want want to exit
    102         # out of this and propagate the exception.
--> 103         return self._coordinator.result()
    104     except KeyboardInterrupt as e:
    105         self.cancel()

File ~\anaconda3\Lib\site-packages\s3transfer\futures.py:266, in TransferCoordina
tor.result(self)
    263 # Once done waiting, raise an exception if present or return the
    264 # final result.
    265 if self._exception:
--> 266     raise self._exception
    267 return self._result
```

```
File ~\anaconda3\Lib\site-packages\s3transfer\tasks.py:139, in Task.__call__(sel
f)
    135     # If the task is not done (really only if some other related
    136     # task to the TransferFuture had failed) then execute the task's
    137     # main() method.
    138     if not self._transfer_coordinator.done():
--> 139         return self._execute_main(kwargs)
    140 except Exception as e:
    141     self._log_and_set_exception(e)

File ~\anaconda3\Lib\site-packages\s3transfer\tasks.py:162, in Task._execute_main
(self, kwargs)
    159 # Log what is about to be executed.
    160 logger.debug(f"Executing task {self} with kwargs {kwargs_to_display}")
--> 162 return_value = self._main(**kwargs)
    163 # If the task is the final task, then set the TransferFuture's
    164 # value to the return value from main().
    165 if self._is_final:

File ~\anaconda3\Lib\site-packages\s3transfer\download.py:673, in IORenameFileTas
k._main(self, fileobj, final_filename, osutil)
    671 def _main(self, fileobj, final_filename, osutil):
    672     fileobj.close()
--> 673     osutil.rename_file(fileobj.name, final_filename)

File ~\anaconda3\Lib\site-packages\s3transfer\utils.py:284, in OSUtils.rename_fil
e(self, current_filename, new_filename)
    283 def rename_file(self, current_filename, new_filename):
--> 284     rename_file(current_filename, new_filename)

File ~\anaconda3\Lib\site-packages\s3transfer\compat.py:24, in rename_file(curren
t_filename, new_filename)
     22 def rename_file(current_filename, new_filename):
     23     try:
---> 24         os.remove(new_filename)
     25     except OSError as e:
     26         if not e.errno == errno.ENOENT:
     27             # We only want to a ignore trying to remove
     28             # a file that does not exist.  If it fails
     29             # for any other reason we should be propagating
     30             # that exception.

PermissionError: [WinError 5] Access is denied: 'E:/Tiger_Analytics/BOTOS3'
```

Deleting a file from a specific bucket

In [93]:

```python
# Specify the name of your S3 bucket and the object to delete
bucket_name = 'ssbucketdemo'
s3_object_name = 'NHANES/sales_data_df2.csv'  # The key of the object to delete

# Delete the object from the S3 bucket
s3.delete_object(Bucket=bucket_name, Key=s3_object_name)

print(f"Object '{s3_object_name}' deleted from S3 bucket '{bucket_name}'.")
```

```
Object 'NHANES/sales_data_df2.csv' deleted from S3 bucket 'ssbucketdemo'.
```

Creating a Zero-Byte File in S3:

In [94]:
```
# to create a zero-byte file (empty file) in S3:

# Define the bucket and file name
bucket_name = 'ssbucketdemo'
file_name = 'your-zero-byte-file.txt'

# Create a zero-byte file
s3.put_object(Bucket=bucket_name, Key=file_name, Body=b'')
print(f"Zero-byte file '{file_name}' created in bucket '{bucket_name}'.")
```

```
Zero-byte file 'your-zero-byte-file.txt' created in bucket 'ssbucketdemo'.
```

In [17]:
```
# Establish the connection
conn = snowflake.connector.connect(
    account= 'prrgexw-hb87719',
    user='ANANDTIGERANALYTICS',
    password = getpass.getpass('Your Snowflake Password: '),
    warehouse='DEMO_WAREHOUSE',
    database='DEMO_DATABASE',
    schema='DEMO_SCHEMA',
    role='ACCOUNTADMIN'
)

# Test the connection
cursor = conn.cursor()
cursor.execute("SELECT CURRENT_VERSION()")
print(cursor.fetchone())
```

```
Your Snowflake Password: ········
('8.45.1',)
```

In [23]:
```
sales_query = 'select * from DEMO_DATABASE.DEMO_SCHEMA.SALES'
```

In [26]:
```
data = conn.cursor().execute(sales_query)
```

In [27]:
```
sales_data = data.fetch_pandas_all()
```

In [28]:
```
sales_data
```

Out[28]:

	DATE	REGION	PRODUCTID	SALESAMOUNT	CUSTOMERID
0	2022-10-20	North	218	39816.0	1348
1	2022-08-09	West	392	77266.0	1817
2	2022-02-04	East	326	16569.0	1222
3	2023-07-27	West	223	45714.0	1621
4	2024-01-19	West	362	27900.0	1005
...	...	...	...	...	...
99995	2021-12-13	East	112	92349.0	1775
99996	2023-01-14	East	225	49705.0	1817
99997	2022-10-31	North	94	87987.0	1566
99998	2023-01-16	South	109	84441.0	1446
99999	2023-11-26	West	307	70879.0	1009

100000 rows × 5 columns

In [34]:
```
sdf = pd.DataFrame(sales_data)
```

In [35]:
```
sdf
```

Out[35]:

	DATE	REGION	PRODUCTID	SALESAMOUNT	CUSTOMERID
0	2022-10-20	North	218	39816.0	1348
1	2022-08-09	West	392	77266.0	1817
2	2022-02-04	East	326	16569.0	1222
3	2023-07-27	West	223	45714.0	1621
4	2024-01-19	West	362	27900.0	1005
...	...	...	...	...	...
99995	2021-12-13	East	112	92349.0	1775
99996	2023-01-14	East	225	49705.0	1817
99997	2022-10-31	North	94	87987.0	1566
99998	2023-01-16	South	109	84441.0	1446
99999	2023-11-26	West	307	70879.0	1009

100000 rows × 5 columns

In [18]:
```
# Create cursor object
cursor = conn.cursor()

# Execute SQL statement
cursor.execute('SELECT * FROM SALES LIMIT 100')
```

```
Out[18]: <snowflake.connector.cursor.SnowflakeCursor at 0x1b7390ec210>
```

```
In [20]: # Fetch results
         results = cursor.fetchall()
```

```
In [ ]: # Close the connection
        conn.close()
```

```
In [ ]:
```

```
In [ ]: def fetch_aws_credentials_from_snowflake():
                try:
                    # Establish Snowflake connection
                    conn = snowflake.connector.connect
                    (
                    account= 'prrgexw-hb87719',
                    user='ANANDTIGERANALYTICS',
                    password = getpass.getpass('Your Snowflake Password: '),
                    warehouse='DEMO_WAREHOUSE',
                    database='DEMO_DATABASE',
                    schema='DEMO_SCHEMA',
                    role='ACCOUNTADMIN'
                    )

                    # Query AWS credentials from a Snowflake table
                    query = "SELECT aws_access_key, aws_secret_key, aws_session_token FR
                    cursor = conn.cursor()
                    cursor.execute(query)
                    row = cursor.fetchone()
                    if row:
                        return {
                            "aws_access_key": row[0],
                            "aws_secret_key": row[1],
                            "aws_session_token": row[2] if len(row) > 2 else None
                        }
                    else:
                        raise ValueError("No credentials found in Snowflake.")
                except Exception as e:
                    raise Exception(f"Error fetching credentials from Snowflake: {str(e)
                finally:
                    cursor.close()
                    conn.close()

            def create_zero_byte_file_in_s3(aws_access_key, aws_secret_key, aws_session_
                """
                Creates a zero-byte file in the specified AWS S3 bucket using boto3.

                Parameters:
                aws_access_key (str): AWS access key.
                aws_secret_key (str): AWS secret access key.
                aws_session_token (str, optional): AWS session token (for temporary cred

                Returns:
                str: Success or error message.
                """
                # Set up the boto3 session
                session = boto3.Session(
                    aws_access_key_id=aws_access_key,
                    aws_secret_access_key=aws_secret_key,
                    aws_session_token=aws_session_token
                )
                s3 = session.client('s3')

                try:
                    # Create a zero-byte object
                    s3.put_object(Bucket=bucket_name, Key=file_name, Body=b'')
                    return f"Zero-byte file '{file_name}' successfully created in bucket
                except Exception as e:
                    return f"Error creating zero-byte file: {str(e)}"

            # Main procedure logic
            try:
                # Fetch AWS credentials from Snowflake
                aws_credentials = fetch_aws_credentials_from_snowflake()

                # Create a zero-byte file in S3
                result = create_zero_byte_file_in_s3(
                    aws_credentials["aws_access_key"],
                    aws_credentials["aws_secret_key"],
                    aws_credentials.get("aws_session_token")
                )
                return result
            except Exception as e:
                return f"Error in stored procedure: {str(e)}"

        # Example usage
        if __name__ == "__main__":
            # Snowflake connection details
            sf_user = "your_snowflake_user"
            sf_password = "your_snowflake_password"
            sf_account = "your_snowflake_account"
            sf_database = "your_snowflake_database"
            sf_schema = "your_snowflake_schema"
            sf_warehouse = "your_snowflake_warehouse"
            sf_role = "your_snowflake_role"

            # AWS S3 bucket and file details
            bucket_name = "your-s3-bucket-name"
            file_name = "your-zero-byte-file.txt"

            # Call the stored procedure
            result = create_zero_byte_file(
                bucket_name, file_name,
                sf_user, sf_password, sf_account,
                sf_database, sf_schema, sf_warehouse, sf_role
            )
            print(result)
```