In [ ]:
```
!pip install boto3 -

import boto3
import logging
from botocore.exceptions import ClientError
import json

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s
logger = logging.getLogger(__name__)

print("Libraries imported successfully!")
```

In [ ]:
```
# AWS Configuration
AWS_REGION = 'us-west-1'
BUCKET_NAME = 'your-bucket-processed'
DATABASE_NAME = 'your-database-name'
CRAWLER_NAME = 'your-crawler-name'
IAM_ROLE = 'your-iam-role-name'
S3_PATH = f's3://{BUCKET_NAME}/'

# Display configuration
print("Current Configuration:")
print(f"Region: {AWS_REGION}")
print(f"Bucket: {BUCKET_NAME}")
print(f"Database: {DATABASE_NAME}")
print(f"Crawler: {CRAWLER_NAME}")
print(f"IAM Role: {IAM_ROLE}")
```

In [ ]:
```
def create_s3_bucket(bucket_name, region=AWS_REGION):
    """
    Create an S3 bucket in a specified region

    :param bucket_name: Bucket to create
    :param region: String region to create bucket in
    :return: True if bucket was created, else False
    """
    try:
        s3_client = boto3.client('s3', region_name=region)

        if region == 'us-east-1':
            s3_client.create_bucket(Bucket=bucket_name)
        else:
            location = {'LocationConstraint': region}
            s3_client.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration=location
            )

        logger.info(f"Bucket '{bucket_name}' created successfully in region
        return True

    except ClientError as e:
        error_code = e.response['Error']['Code']
        if error_code == 'BucketAlreadyExists':
            logger.error(f"Bucket '{bucket_name}' already exists (owned by s
        elif error_code == 'BucketAlreadyOwnedByYou':
            logger.warning(f"Bucket '{bucket_name}' already owned by you")
            return True
        else:
            logger.error(f"Error creating bucket: {e}")
        return False


print("Creating S3 bucket...")
if create_s3_bucket(BUCKET_NAME, AWS_REGION):
    print(f"Successfully created/verified bucket: {BUCKET_NAME}")
    S3_PATH = f's3://{BUCKET_NAME}/'
else:
    print(f"Failed to create bucket: {BUCKET_NAME}")
```

In [ ]:

```
def create_glue_database(database_name=DATABASE_NAME, region=AWS_REGION):
    """
    Create an AWS Glue database

    :param database_name: Name of the database to create
    :param region: AWS region
    :return: True if successful, False otherwise
    """
    try:
        glue_client = boto3.client('glue', region_name=region)

        response = glue_client.create_database(
            DatabaseInput={
                'Name': database_name,
                'Description': 'Simple data lake database for analytics'
            }
        )

        logger.info(f"Database '{database_name}' created successfully!")
        return True

    except ClientError as e:
        error_code = e.response['Error']['Code']
        if error_code == 'AlreadyExistsException':
            logger.warning(f"Database '{database_name}' already exists")
            return True
        else:
            logger.error(f"Error creating database: {e}")
            return False

print("Creating Glue database...")
if create_glue_database():
    print(f"Successfully created/verified database: {DATABASE_NAME}")
else:
    print(f"Failed to create database: {DATABASE_NAME}")
```

In [ ]:

```
def create_glue_crawler(crawler_name=CRAWLER_NAME, s3_path=S3_PATH,
                       database_name=DATABASE_NAME, iam_role=IAM_ROLE):
    """
    Create a simple AWS Glue crawler for S3

    :param crawler_name: Name of the crawler
    :param s3_path: S3 path to crawl
    :param database_name: Target database name
    :param iam_role: IAM role for the crawler
    :return: Response from create_crawler API call
    """
    try:
        glue_client = boto3.client('glue', region_name=AWS_REGION)

        response = glue_client.create_crawler(
            Name=crawler_name,
            Role=iam_role,
            DatabaseName=database_name,
            Targets={
                'S3Targets': [
                    {
                        'Path': s3_path
                    }
                ]
            },
            Description='Simple S3 crawler for data lake',
            Configuration=json.dumps({
                "Version": 1,
                "CreatePartitionIndex": True,
                "Grouping": {
                    "TableGroupingPolicy": "CombineCompatibleSchemas"
                }
            }),
            SchemaChangePolicy={
                'UpdateBehavior': 'UPDATE_IN_DATABASE',
                'DeleteBehavior': 'DEPRECATE_IN_DATABASE'
            }
        )

        logger.info(f"Crawler '{crawler_name}' created successfully")
        return response

    except ClientError as e:
        error_code = e.response['Error']['Code']
        if error_code == 'AlreadyExistsException':
            logger.warning(f"Crawler '{crawler_name}' already exists")
            return True
        else:
            logger.error(f"Error creating crawler: {e}")
            return False

print("Creating Glue crawler...")
crawler_response = create_glue_crawler()
if crawler_response:
    print(f"Successfully created/verified crawler: {CRAWLER_NAME}")
else:
    print(f"Failed to create crawler: {CRAWLER_NAME}")
```

In [ ]:
```python
def verify_setup():
    """Verify that all resources were created successfully"""

    print("\n Verifying setup...")

    # Check S3 bucket
    try:
        s3_client = boto3.client('s3', region_name=AWS_REGION)
        s3_client.head_bucket(Bucket=BUCKET_NAME)
        print(f"S3 bucket '{BUCKET_NAME}' exists")
    except:
        print(f"S3 bucket '{BUCKET_NAME}' not found")

    # Check Glue database
    try:
        glue_client = boto3.client('glue', region_name=AWS_REGION)
        glue_client.get_database(Name=DATABASE_NAME)
        print(f"Glue database '{DATABASE_NAME}' exists")
    except:
        print(f"Glue database '{DATABASE_NAME}' not found")

    # Check Glue crawler
    try:
        glue_client.get_crawler(Name=CRAWLER_NAME)
        print(f"Glue crawler '{CRAWLER_NAME}' exists")
    except:
        print(f"Glue crawler '{CRAWLER_NAME}' not found")

# Run verification
verify_setup()

print("\n Next Steps:")
print("1. Upload data files to your S3 bucket")
print("2. Run the Glue crawler to discover schemas")
print("3. Query your data using Amazon Athena or other analytics tools")
print(f"4. Access your data at: s3://{BUCKET_NAME}/")
```

In [ ]:
```python
def run_crawler(crawler_name=CRAWLER_NAME):
    """Start the Glue crawler"""
    try:
        glue_client = boto3.client('glue', region_name=AWS_REGION)
        response = glue_client.start_crawler(Name=crawler_name)
        print(f"Crawler '{crawler_name}' started successfully")
        return response
    except ClientError as e:
        print(f"Error starting crawler: {e}")
        return False

def check_crawler_status(crawler_name=CRAWLER_NAME):
    """Check the status of the Glue crawler"""
    try:
        glue_client = boto3.client('glue', region_name=AWS_REGION)
        response = glue_client.get_crawler(Name=crawler_name)
        status = response['Crawler']['State']
        print(f"Crawler '{crawler_name}' status: {status}")
        return status
    except ClientError as e:
        print(f"Error checking crawler status: {e}")
        return None

def list_tables_in_database(database_name=DATABASE_NAME):
    """List all tables in the Glue database"""
    try:
        glue_client = boto3.client('glue', region_name=AWS_REGION)
        response = glue_client.get_tables(DatabaseName=database_name)
        tables = [table['Name'] for table in response['TableList']]
        print(f"Tables in database '{database_name}': {tables}")
        return tables
    except ClientError as e:
        print(f"❌ Error listing tables: {e}")
        return []

print("Helper functions loaded!")
print("Available functions:")
print("- run_crawler(): Start the crawler")
print("- check_crawler_status(): Check crawler status")
print("- list_tables_in_database(): List discovered tables")
```

In [ ]:

```
# Example usage:

#run_crawler()
#check_crawler_status()
#list_tables_in_database()

print("Uncomment the functions above to use them!")
print("AWS Data Lake setup complete!")
```