

# CIS 602: Big Data – Homework 3

Pradyoth Singenahalli Prabhu - 02071847

## Task 1: Using an AWS Glue crawler with the GHCN-D dataset.

Adding tables using crawler called **Weather** :

The screenshot shows the AWS Glue 'Add crawler' wizard at Step 2: Choose data sources and classifiers. On the left sidebar, under 'Data Catalog', 'Tables' is selected. The main panel displays the 'Edit data source' configuration. In the 'Data source' dropdown, 'S3' is chosen. Below it, the 'Network connection - optional' section is visible. Under 'S3 path', the URL 's3://noaa-ghcn-pds/csv/by\_year/' is entered. The 'Subsequent crawler runs' section contains several options: 'Crawl all sub-folders' (selected), 'Crawl new sub-folders only', 'Crawl based on events', 'Sample only a subset of files', and 'Exclude files matching pattern'. At the bottom right of the modal is the 'Update S3 data source' button.

The screenshot shows the AWS Glue 'Add crawler' wizard at Step 3: Set crawler properties. The left sidebar remains the same. The main panel now displays the 'Data source configuration' section. It asks if the data is already mapped to Glue tables, with 'Not yet' selected. Below this, the 'Data sources (1) Info' section shows a single entry: 'Type: S3' and 'Data source: s3://noaa-ghcn-pds/csv/by\_year/'. The 'Custom classifiers - optional' section is present but empty. At the bottom right of the modal is the 'Next' button.

## Choosing **LabRole** as IAM role to configure settings

AWS Glue > Crawlers > Add crawler

**Configure security settings**

IAM role **LabRole**

Existing IAM role **LabRole**

Create new IAM role Update chosen IAM role

Lake Formation configuration - optional

Security configuration - optional

Cancel Previous Next

Creating database called **weatherdata**.

AWS Glue > Databases > Add database

Create a database

Database details

Name **weatherdata**

Location - optional

Description - optional

Create database

Setting output and scheduling

## Reviewing and creating crawler:

Crawler ran successfully:

**Crawler successfully starting**  
The following crawler is now starting: "Weather"

**Weather**

**Crawler properties**

Name Weather	IAM role LabRole	Database weatherdata	State READY
Description -	Security configuration -	Lake Formation configuration -	Table prefix -
Maximum table threshold -			

**Crawler runs (1)**  
The list of crawler runs for this crawler.

Start time (UTC)	End time (UTC)	Current/last duration	Status	DPU hours	Table changes
October 10, 2023 at 16:40:56	October 10, 2023 at 16:41:52	56 s	Completed	-	-

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

## Reviewing the metadata that AWS Glue created:

**The following crawler is now starting: "Weather"**

**by\_year**

**Table overview** | Data quality New

**Table details** | Advanced properties

Name by_year	Description -	Database weatherdata	Classification CSV
Location <a href="s3://noaa-ghcn-pds/csv/by_year/">s3://noaa-ghcn-pds/csv/by_year/</a>	Connection -	Deprecated -	Last updated October 10, 2023 at 16:41:52

**Schema (8)**  
View and manage the table schema.

#	Column name	Data type	Partition key	Comment
1	id	string	-	-
2	date	bigint	-	-
3	element	string	-	-
4	data_value	bigint	-	-
5	m_flag	string	-	-
6	q_flag	string	-	-
7	s_flag	string	-	-
8	obs_time	bigint	-	-

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

## Updating the schema:

The following crawler is now starting: "Weather"

AWS Glue > Tables > by\_year

**by\_year**

Last updated (UTC) October 10, 2023 at 17:00:08 Version 1 (Current version) Actions

**Table overview** Data quality New

**Table details** Advanced properties

Name: by\_year Description: - Database: weatherdata Classification: CSV

Location: s3://noaa-ghcn-pds/csv/by\_year/ Connection: - Deprecated: - Last updated: October 10, 2023 at 17:00:08

Input format: org.apache.hadoop.mapred.TextInputFormat Output format: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe

**Schemas** | Partitions | Indexes

**Schema (8)** View and manage the table schema.

#	Column name	Data type	Partition key	Comment
1	station	string	-	-
2	date	bigint	-	-
3	type	string	-	-
4	observation	bigint	-	-
5	mflag	string	-	-
6	qflag	string	-	-
7	sflag	string	-	-
8	time	bigint	-	-

Edit schema as JSON | Edit schema

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

## Task 2: Querying a table by using Athena

Configuring an S3 bucket to store Athena query results:

Athena now supports typeahead code suggestions to speed up SQL query development. Typeahead suggestions are turned on by default. You can change this setting in query editor preferences. Edit preferences

Query 11 : X | Query 12 : X | Query 13 : X

1 | SELECT \* FROM "weatherdata"."by\_year" limit 10;

SQL Ln 1, Col 1 Run again Explain Cancel Clear Create + Reuse query results up to 60 minutes ago

Query results | Query stats

Completed Time in queue: 158 ms Run time: 873 ms Data scanned: 552.70 KB

Results (10)

#	station	date	type	observation	mflag	qflag	sflag	time
1	EZE00100082	17950101	TMAX	-65			E	
2	EZE00100082	17950101	TMIN	-140			E	
3	GM000010962	17950101	PRCP	10			E	
4	ITE00100554	17950101	TMAX	19			E	
5	ITE00100554	17950101	TMIN	0	I		E	
6	EZE00100082	17950102	TMAX	-130			E	
7	EZE00100082	17950102	TMIN	-216			E	
8	GM000010962	17950102	PRCP	0			E	
9	ITE00100554	17950102	TMAX	-15	I		E	
10	ITE00100554	17950102	TMIN	-28			E	

Copy Download results

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

Creating a table for data after 1950:

```
CREATE table weatherdata.late20th
WITH (
    format='PARQUET', external_location='s3://ade-my-bucket-48f0e7e0/lab3'
```

```
) AS SELECT date, type, observation FROM by_year
WHERE date/10000 between 1950 and 2015;
```

The screenshot shows the AWS Athena Query Editor interface. The top navigation bar includes 'Services' and 'Search' with '(Option+S)' and a user icon. The top right shows 'N. Virginia' and 'vocabs/user2486983=psingenhallprabhu@umassd.edu @ 4284-441...'. Below the navigation is a breadcrumb trail: 'Amazon Athena > Query editor'. The main area has tabs for 'Editor', 'Recent queries', 'Saved queries', and 'Settings'. A 'Workgroup' dropdown is set to 'primary'. A message box at the top says 'Athena now supports typeahead code suggestions to speed up SQL query development. Typeahead suggestions are turned on by default. You can change this setting in query editor preferences.' with a 'Edit preferences' button. The central workspace shows a query editor with three tabs: 'Query 11', 'Query 12', and 'Query 13'. The 'Query 13' tab contains the following SQL code:

```
1 CREATE table weatherdata.late20th
2 - WITH (
3   format='PARQUET', external_location='s3://ade-my-bucket-48f0e7e0/lab3'
4 ) AS SELECT date, type, observation FROM by_year
5 WHERE date/10000 between 1950 and 2015;
```

Below the code are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. To the right is a 'Reuse query results' link. The 'Query results' tab shows a green status bar with 'Completed' and performance metrics: 'Time in queue: 163 ms', 'Run time: 1 min 42.611 sec', and 'Data scanned: 100.58 GB'. The 'Query stats' tab is also visible. On the left sidebar, under 'Data', there are sections for 'Data source' (set to 'AwsDataCatalog'), 'Database' (set to 'weatherdata'), and 'Tables and views' (listing 'Tables (2)': 'by\_year' and 'late20th', and 'Views (0)'). At the bottom of the editor are 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create' buttons. The bottom right of the editor area shows 'CloudShell' and 'Feedback' links.

**Time in queue:** 163 ms

**Run time:** 1 min 42.611 sec

**Data scanned:** 100.58 GB

## Previewing the table late20th

The screenshot shows the AWS Athena Query Editor interface, similar to the previous one but with a different query. The top navigation bar and user information are the same. The main area shows a query editor with four tabs: 'Query 11', 'Query 12', 'Query 13', and 'Query 14'. The 'Query 14' tab contains the following SQL code:

```
1 SELECT * FROM "weatherdata"."late20th" limit 10;
```

Below the code are buttons for 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create'. To the right is a 'Reuse query results' link. The 'Query results' tab shows a green status bar with 'Completed' and performance metrics: 'Time in queue: 174 ms', 'Run time: 1.21 sec', and 'Data scanned: 82.54 MB'. The 'Query stats' tab is also visible. The 'Results (10)' section displays a table with 10 rows of data:

#	date	type	observation
1	19520115	TMIN	-180
2	19520115	PRCP	0
3	19520115	TAVG	-120
4	19520115	TMAX	-80
5	19520115	TMIN	-181
6	19520115	PRCP	1
7	19520115	TAVG	-129
8	19520115	PRCP	1
9	19520115	PRCP	4
10	19520115	TAVG	-158

At the bottom of the editor are 'Run again', 'Explain', 'Cancel', 'Clear', and 'Create' buttons. The bottom right of the editor area shows 'CloudShell' and 'Feedback' links.

## Creating a view that only includes the maximum temperature reading, or TMAX, value:

```
CREATE VIEW TMAX AS
SELECT date, observation, type
FROM late20th
WHERE type = 'TMAX'
```

The screenshot shows the AWS Glue Data Catalog interface. On the left, the sidebar displays the 'Data' section with 'Tables (2)' and 'Views (1)'. Under 'Tables', there are entries for 'by\_year' and 'late20th'. Under 'Views', there is one entry for 'tmax'. The main area shows a query editor with multiple tabs (Query 11 to Query 16) and a selected tab for 'Query 16' containing the SQL command: `1 | SELECT * FROM "weatherdata"."tmax" limit 10;`. Below the editor is a results pane titled 'Results (10)' showing a table with columns: #, date, observation, and type. The data is as follows:

#	date	observation	type
1	19530101	117	TMAX
2	19530101	129	TMAX
3	19530101	132	TMAX
4	19530101	154	TMAX
5	19530101	150	TMAX
6	19530101	179	TMAX
7	19530101	146	TMAX
8	19530101	-43	TMAX
9	19530101	62	TMAX
10	19530101	44	TMAX

**Time in queue:** 148 ms

**Run time:** 2.164 sec

**Data scanned:** 40.94 MB

## Running query to calculate the average maximum temperature for each year in the dataset:

```
SELECT date/10000 as Year, avg(observation)/10 as Max
FROM tmax
GROUP BY date/10000 ORDER BY date/10000;
```

The screenshot shows the AWS Glue Data Catalog Query Editor interface. On the left, the sidebar displays the Data source (AwsDataCatalog) and Database (weatherdata). Under Tables and views, there are two tables: by\_year and late20th, and one view: tmax. The main area shows a query editor with the following SQL code:

```

1 SELECT date/10000 as Year, avg(observation)/10 as Max
2 FROM tmax
3 GROUP BY date/10000 ORDER BY date/10000;

```

Below the query editor, the results pane shows the output of the query. The results table has three columns: #, Year, and Max. The data is as follows:

#	Year	Max
1	1950	16.75775918412137
2	1951	16.837634449250636
3	1952	17.314945621006597
4	1953	17.950618416461015
5	1954	17.515388428455587
6	1955	17.138707766551114
7	1956	17.191869074624613
8	1957	17.437853715079818
9	1958	17.3350013111580267
10	1959	17.404585407210014
11	1960	16.963348095682107

At the bottom of the results pane, it says "Time in queue: 164 ms Run time: 29.201 sec Data scanned: 2.25 GB".

**Time in queue:** 164 ms

**Run time:** 29.201 sec

**Data scanned:** 2.25 GB

## Task 3: Creating a CloudFormation template for an AWS Glue crawler

Creating a new CloudFormation template called `gluecrawler.cf.yml` :

```

AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  CFNCrawlerName:
    Type: String
    Default: cfn-crawler-weather
  CFNDatabaseName:
    Type: String
    Default: cfn-database-weather
  CFNTablePrefixName:
    Type: String
    Default: cfn_sample_1-weather
# Resources section defines metadata for the Data Catalog
Resources:
# Create a database to contain tables created by the crawler
  CFNDatabaseWeather:
    Type: AWS::Glue::Database
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        Name: !Ref CFNDatabaseName
        Description: "AWS Glue container to hold metadata tables for the weather crawler"
#Create a crawler to crawl the weather data on a public S3 bucket
  CFNCrawlerWeather:
    Type: AWS::Glue::Crawler
    Properties:
      Name: !Ref CFNCrawlerName
      Role: arn:aws:iam::428444136833:role/LabRole
      #Classifiers: none, use the default classifier
      Description: AWS Glue crawler to crawl weather data
      #Schedule: none, use default run-on-demand
      DatabaseName: !Ref CFNDatabaseName
    Targets:
      S3Targets:
        # Public S3 bucket with the weather data
        - Path: "s3://noaa-ghcn-pds/csv/by_year/"
  TablePrefix: !Ref CFNTablePrefixName

```

```

SchemaChangePolicy:
  UpdateBehavior: "UPDATE_IN_DATABASE"
  DeleteBehavior: "LOG"
Configuration: "{\"Version\":1.0,\"CrawlerOutput\":{\"Partitions\":{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":{\"AddOrUpdateBehavior\":\"MergeNewColumns\"}}}"

```

**To validate the CloudFormation template, running the following command in the AWS Cloud9 terminal:**

```
aws cloudformation validate-template --template-body file://gluecrawler.cf.yml
```

Got the following output:

```
{
  "Parameters": [
    {
      "ParameterKey": "CFNCrawlerName",
      "DefaultValue": "cfn-crawler-weather",
      "NoEcho": false
    },
    {
      "ParameterKey": "CFNTablePrefixName",
      "DefaultValue": "cfn_sample_1-weather",
      "NoEcho": false
    },
    {
      "ParameterKey": "CFNDatabaseName",
      "DefaultValue": "cfn-database-weather",
      "NoEcho": false
    }
  ]
}
```

The screenshot shows the AWS Cloud9 IDE interface. On the left, there's a file tree for a project named 'cloud9\_athena'. It contains several files: 'athenaquery.cf.yml', 'creat\_iam\_role\_glue.yml', 'gluecrawler.cf.yml', 'iam-role.yml', and 'README.md'. The 'gluecrawler.cf.yml' file is open in the main editor area. It defines a CloudFormation stack with three parameters: 'CFNCrawlerName' (value: 'cfn-crawler-weather'), 'CFNTablePrefixName' (value: 'cfn\_sample\_1-weather'), and 'CFNDatabaseName' (value: 'cfn-database-weather'). The stack also includes a 'CloudWatchLogsLogGroup' resource with a log stream named 'CloudWatchLogsLogStream'. Below the editor, there's an 'aws - \*' terminal window showing the validation command output.

```

aws - *ip-172-31-33-75.ec2.internal aws cloudformation validate-template --template-body file://gluecrawler.cf.yml

```

**To create the CloudFormation stack, running the following command:**

```
aws cloudformation create-stack --stack-name gluecrawler --template-body file://gluecrawler.cf.yml --capabilities CAPABILITY_NAMED_IAM
```

Got this as a response output:

```
{
    "StackId": "arn:aws:cloudformation:us-east-1:428444136833:stack/gluecrawler/05476a30-67b2-11ee-bdaa-0ef627c8827f"
}
```

The screenshot shows the AWS CloudFormation Stack Editor interface. On the left, there's a sidebar with navigation links like 'File', 'Edit', 'Find', 'View', 'Run', 'Tools', 'Window', 'Support', 'Preview', and 'Run'. Below the sidebar, there's a tree view of the stack resources:

- cloudSt\_athena - #**
  - athenaquery.cfn.yml
  - creat\_iam\_role\_glue.yml
  - gluecrawler.cfn.yml
  - iam-role.yaml
  - READEME.md

The main area displays the `gluecrawler.cfn.yml` template:

```

10 type: AWS::Glue::Database
Properties:
  CatalogId: !Ref AWS::AccountId
  DatabaseInput:
    CrawlerSchemaName: !Ref CNDatabaseName
    Description: "AWS Glue container to hold metadata tables for the weather crawler"
    #Create a crawler to crawl the weather data on a public S3 bucket
23 #CFNWeatherCrawler:
24   Type: AWS::Glue::Crawler
25   Properties:
26     Name: !Ref CFNcrawlerName
27     Role: !Ref IAMRole
28     #Classifier: none, use the default classifier
29     #Description: AWS Glue crawler to crawl weather data
30     #LastCrawl: none, use default run-on-demand
31     DatabaseName: !Ref CNDatabaseName
32     Targets:
33       S3Targets:
34         # Public S3 bucket with the weather data
35         Path: "s3://noaa-ghcn-pds/csv/by_year/"
36       TablePrefix: !Ref CNTablePrefixName
37     SchemaChangePolicy:
38       UpdateBehavior: "UPDATE_IN_DATABASE"
39       DeleteBehavior: "LOG"
40     Configuration: "{\"Version\":1.0,\"CrawlerOutput\":{\"Partitions\":{\"InheritFromTable\":true},\"Tables\":{\"AddOrUpdateBehavior\":\"MergeNewColumns\"}}}"
41
  
```

Below the code editor is an AWS Lambda function configuration panel with tabs for 'Configuration', 'Code', 'Logs', and 'Test'. The 'Code' tab shows the Lambda function code:

```

aws -p:172-31-33-76.ecx | Immediate
{
  "ParameterKey": "CFNWeatherCrawler",
  "DefaultValue": "cfn-crawler-weather",
  "NoEcho": false
},
{
  "ParameterKey": "CNTablePrefixName",
  "DefaultValue": "cfn_sample_1-weather",
  "NoEcho": false
},
{
  "ParameterKey": "CNDatabaseName",
  "DefaultValue": "cfn-database-weather",
  "NoEcho": false
}
vocabs:~/environment $ aws cloudformation create-stack --stack-name gluecrawler --template-body file://gluecrawler.cfn.yml --capabilities CAPABILITY_NAMED_IAM
{
  "StackId": "arn:aws:cloudformation:us-east-1:428444136833:stack/gluecrawler/05476a30-67b2-11ee-bdaa-0ef627c8827f"
}
vocabs:~/environment $ 
  
```

## To verify that the AWS Glue database was created in the stack:

```
aws glue get-databases
```

```
{
  "DatabaseList": [
    {
      "Name": "cfn-database-weather",
      "Description": "AWS Glue container to hold metadata tables for the weather crawler",
      "Parameters": {},
      "CreateTime": "2023-10-10T21:14:42+00:00",
      "CreateTableDefaultPermissions": [
        {
          "Principal": {
            "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
          },
          "Permissions": [
            "ALL"
          ]
        }
      ],
      "CatalogId": "428444136833"
    },
    {
      "Name": "default",
      "CreateTime": "2023-09-27T21:21:11+00:00",
      "CreateTableDefaultPermissions": [
        {
          "Principal": {
            "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
          },
          "Permissions": [
            "ALL"
          ]
        }
      ],
      "CatalogId": "428444136833"
    }
  ]
}
```

```

{
    "Name": "taxidata",
    "CreateTime": "2023-09-27T21:21:12+00:00",
    "CreateTableDefaultPermissions": [
        {
            "Principal": {
                "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
            },
            "Permissions": [
                "ALL"
            ]
        }
    ],
    "CatalogId": "428444136833"
},
{
    "Name": "weatherdata",
    "CreateTime": "2023-10-10T16:37:11+00:00",
    "CreateTableDefaultPermissions": [
        {
            "Principal": {
                "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
            },
            "Permissions": [
                "ALL"
            ]
        }
    ],
    "CatalogId": "428444136833"
}
]
}

```

The screenshot shows the AWS Cloud9 IDE interface. The left sidebar displays the file structure: `aws-lambda-function.zip` containing `awsquery.yaml`, `create_lambda_glue.yaml`, `gluecrawler.yaml`, `lambda.yaml`, and `README.md`. The main editor window shows the `aws-lambda-function.zip` file with the following AWS Lambda function code:

```

function handler(event, context) {
    const AWS = require('aws-sdk');
    const glueClient = new AWS.Glue();
    const dynamoDBClient = new AWS.DynamoDB.DocumentClient();

    const databaseList = [
        {
            "Name": "cfn-database-weather",
            "Description": "AWS Glue container to hold metadata tables for the weather crawler",
            "CreateTime": "2023-10-18T21:14:42+00:00",
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "428444136833"
        },
        {
            "Name": "default",
            "CreateTime": "2023-09-27T21:21:11+00:00",
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "428444136833"
        },
        {
            "Name": "taxidata",
            "CreateTime": "2023-09-27T21:21:12+00:00",
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "428444136833"
        }
    ];
    const databaseNames = databaseList.map(db => db.Name);
    const catalogId = databaseList[0].CatalogId;

    const databaseParams = {
        DatabaseName: databaseNames[0],
        CatalogId: catalogId
    };

    const tableParams = {
        DatabaseName: databaseNames[0],
        CatalogId: catalogId
    };

    const crawlerParams = {
        Name: "weatherdata",
        CatalogId: catalogId,
        Role: "arn:aws:iam::123456789012:role/service-role/AWSGlueServiceRole"
    };

    const crawlerConfig = {
        CrawlerParameters: {
            Parameters: {
                "partitionKeys": "[]",
                "partitionKeysType": "None"
            }
        },
        DatabaseName: databaseNames[0],
        Role: "arn:aws:iam::123456789012:role/service-role/AWSGlueServiceRole"
    };

    const crawler = new AWS.Lambda();
    const database = new AWS.DynamoDB();
    const glue = new AWS.Glue();
    const s3 = new AWS.S3();

    const createDatabasePromise = glue.createDatabase(databaseParams).promise();
    const createTablePromise = database.createTable(tableParams).promise();
    const startCrawlerPromise = crawler.start(crawlerParams).promise();
    const updateCrawlerPromise = crawler.update(crawlerConfig).promise();

    Promise.all([createDatabasePromise, createTablePromise, startCrawlerPromise, updateCrawlerPromise]).then(() =>
        console.log("Database and table created, crawler started, and crawler configuration updated successfully.");
    ).catch(error =>
        console.error(`An error occurred: ${error}`);
    );
}

```

### Verifying that the crawler was created in the stack:

```
aws glue list-crawlers
```

```

awsquery.yaml | lamlock.yaml | glueawsecr.yaml
File Edit Find View Go Run Tools Window Support Preview Run
Go to Anything (F5)
dynamodb: &gt;
  <awsquery.yaml>
  <lamlock.yaml>
  <glueawsecr.yaml>
  <aws>
    README.md
  </aws>
  <aws>
    <awsquery.yaml>
      <aws>
        <awsquery.yaml>
          <aws>
            <aws>
              <aws>
                <aws>
                  <aws>
                    <aws>
                      <aws>
                        <aws>
                          <aws>
                            <aws>
                              <aws>
                                <aws>
                                  <aws>
                                    <aws>
                                      <aws>
                                        <aws>
                                          <aws>
                                            <aws>
                                              <aws>
                                                <aws>
                                                  <aws>
                                                    <aws>
                                                      <aws>
                                                        <aws>
                                                          <aws>
                                                            <aws>
                                                              <aws>
                                                                <aws>
                                                                  <aws>
                                                                    <aws>
                                                                      <aws>
                                                                        <aws>
                                                                          <aws>
                                                                            <aws>
                                                                              <aws>
                                                                                <aws>
                                                                                  <aws>
                                                                                    <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
                                                                                      <aws>
................................................................

```

```

bash-3.1-31-75.e: ~ /aws-glue/cfn-crawler-weather > Immediate > x
vectabs:~/environment $ aws glue list-crawlers
{
  "CrawlerNames": [
    "Weather",
    "cfn-crawler-weather"
  ]
}
vectabs:~/environment $ 

```

To retrieve the details of the crawler, running the following command:

```
aws glue get-crawler --name cfn-crawler-weather
```

```
{
  "Crawler": {
    "Name": "cfn-crawler-weather",
    "Role": "LabRole",
    "Targets": {
      "S3Targets": [
        {
          "Path": "s3://noaa-ghcn-pds/csv/by_year/",
          "Exclusions": []
        }
      ],
      "JdbcTargets": [],
      "MongoDBTargets": [],
      "DynamoDBTargets": [],
      "CatalogTargets": [],
      "DeltaTargets": [],
      "IcebergTargets": [],
      "HudiTargets": []
    },
    "DatabaseName": "cfn-database-weather",
    "Description": "AWS Glue crawler to crawl weather data",
    "Classifiers": [],
    "RecrawlPolicy": {
      "RecrawlBehavior": "CRAWL_EVERYTHING"
    },
    "SchemaChangePolicy": {
      "UpdateBehavior": "UPDATE_IN_DATABASE",
      "DeleteBehavior": "LOG"
    },
    "LineageConfiguration": {
      "CrawlerLineageSettings": "DISABLE"
    },
    "State": "READY",
    "TablePrefix": "cfn_sample_1-weather",
    "CrawlElapsedTime": 0,
    "CreationTime": "2023-10-10T21:14:42+00:00",
    "LastUpdated": "2023-10-10T21:14:42+00:00",
    "Version": 1,
    "Configuration": "{\"Version\":1.0,\"CrawlerOutput\":{\"Partitions\":{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":{\"LakeFormationConfiguration\":{},\"UseLakeFormationCredentials\":false}}}"
  }
}
```

```

    "AccountId": ""
  }
}

```

```

Go to Anything (F1) File Edit Find View Go Run Tools Window Support Preview Run
awsquery.yml imrole.yml gluecrawler.yml
Search: cloudformation > gluemetering
cloudformation
gluecrawler.yml
gluecrawler
imrole.yml
imrole
README.md
AWS Go to Anything (F1)
awsquery.yml
imrole.yml
gluecrawler.yml
Preview Run
1 type: AWS::CloudWatchMetricsInsights
2 DefaultCfnDatabase: weather
3 DefaultCfnDatabaseName: weather
4 Type: String
5 Default: cfnsample_1-weather
6 #Description: AWS CloudWatch Metrics Insights metadata for the Data Catalog
7 Resources:
8   CfnDatabaseMetrics: AWS::CloudWatchMetricsInsights::Database
9     Type: AWS::CloudWatchMetricsInsights::Database
10    Properties:
11      CatalogId: !Ref AWS::AccountId
12      DatabaseName: CfnDatabaseName
13      Description: AWS Glue container to hold metadata tables for the weather crawler
14      #Creates a crawler to crawl the weather data on a public S3 bucket
15      CfnDatabaseMetrics:
16        Type: AWS::Glue::Database
17        Properties:
18          Name: cfnsample_1-weather
19          CatalogId: !Ref AWS::AccountId
20          DatabaseType: CfnDatabaseName
21          Description: AWS Glue crawler to crawl weather data on a public S3 bucket
22          #Creates a crawler to crawl the weather data on a public S3 bucket
23          Type: AWS::Glue::Crawler
24          Properties:
25            Role: !Ref CFNRoleLambda
26            Targets:
27              LambdaTargets:
28                Path: !Ref LambdaTargetPath
29                Description: AWS Glue crawler to crawl weather data
30                Schedule: none, use default run-on-demand
31                DatabaseName: !Ref CfnDatabaseName
32                Targets:
33                  S3Targets:
33:18 YAML Spaces: 1
aws -D T2-31-03-70.ecx Immediate
{
  "Crawler": {
    "Name": "cfnsample_1-weather",
    "Role": "Lambda",
    "Targets": {
      "S3Targets": [
        {
          "Path": "s3://noaa-ghcn-pds/csv/by_year/",
          "Exclusions": []
        }
      ],
      "MongoDBTargets": [],
      "MySQLTargets": [],
      "CatalogTargets": [],
      "DeltaTargets": [],
      "RedisTargets": [],
      "HudiTargets": []
    },
    "DatabaseName": "cfn-database-weather",
    "Description": "AWS Glue crawler to crawl weather data",
    "Classifiers": [],
    "RecrawlBehavior": {
      "RecrawlBehavior": "CRAWL_EVERYTHING"
    },
    "SchemaChangePolicy": {
      "UpdateBehavior": "UPDATE_IN_DATABASE",
      "DeleteBehavior": "LOG"
    },
    "LineageConfiguration": {
      "CrawlingLineageSettings": "DISABLE"
    },
    "State": "READY",
    "TablePrefix": "cfnsample_1-weather",
    "CreateLastUpdate": 0,
    "CreationTime": "2022-10-10T21:14:42+00:00",
    "LastUpdate": "2022-10-10T21:14:42+00:00",
    "Version": 1,
    "Configuration": "{\"Values\":{}},{\"CrawlerOutput\":{\"Partitions\":{},\"InheritFromTable\":true},\"Tables\":{},\"AddOrUpdateBehavior\":{\"MergeNewColumns\":true}},\"UseLakeFormationCredentials\":false,\"AccountId\": \""
  }
}
[END]

```

## Conclusion Remarks:

In these tasks, we adeptly harnessed a suite of AWS services to enhance data processing and analysis. AWS Glue played a pivotal role in automating the discovery and cataloging of data stored in an S3 bucket. This markedly streamlined the traditionally labor-intensive process of inferring schemas and crafting databases from Amazon S3 data, significantly boosting data management efficiency.

Subsequently, we strategically deployed Athena to query tables generated by the AWS Glue crawler. By optimizing queries with the Apache Parquet format, we expedited query execution while concurrently achieving cost savings. This was exemplified in the calculation of average maximum temperatures by year, underlining the effectiveness of integrating AWS Glue and Athena in deriving actionable data insights.

A distinctive highlight was the integration of an AWS Glue crawler into a CloudFormation template, allowing standardized deployment across multiple AWS accounts. This aligned with best practices for managing discrete development, testing, and production environments.

These endeavors collectively underscore our profound proficiency in AWS data services and our ability to streamline data processing, a valuable skillset poised to greatly benefit our objectives in the U.S. tech sector. For further guidance or questions related to your report, please feel free to reach out.

Furthermore, to summarize the "Concluding Remarks," it's essential to note that our approach ensured data accessibility and cataloging through AWS Glue, adapted the dataset's schema to optimize it, streamlined data retrieval and storage using Athena, effectively segmented and archived data, and conducted vital analytics and insights to support decision-making across various domains.