Scenario

The data science team has asked you to create a series of proofs of concept (POCs) to use AWS services to address many of the data engineering and data analysis needs of the university. Mary, one of the data science team members, has seen what Athena can do to create tables that have defined schemas and is impressed. She asks you if it's possible to infer the columns and data types automatically. Defining the schema takes much of her time when she deals with large amounts of varied data. You want to develop a POC to use AWS Glue, which is designed for use cases that are similar to this one.

To develop a POC, Mary suggests that you use a publicly available dataset, the Global Historical Climatology Network Daily [GHCN-D] dataset, contains daily weather summaries from ground-based stations, going back to 1763. The dataset is publicly available in an S3 bucket.

Mary explains that the most common recorded parameters in the dataset are daily temperatures, rainfall, and snowfall. These parameters are useful to assess risks for drought, flooding, and extreme weather. The data definitions used in this lab are available on the NOAA Global Historical Climatology Network Daily (GHCN-D) Dataset page.

Note: As of October 2022, the dataset has been split into sub-datasets, **by_year** and **by_station**. Throughout this exercise you will be using **by_year** and it can be found at **s3://noaa-ghcn-pds/csv/by_year/**.

Tasks

Numbered Task	Detail
1	You will create and use an AWS Glue crawler named *weather* to access the GHCN-D dataset, which is in a public S3 bucket. The crawler will populate the *weatherdata* database in the AWS Glue Data Catalog.
2	You will also use Athena to transform the database by modifying its schema.

Accessing the AWS Management Console

1. At the top of these instructions, choose **Start Lab**.
    - The lab session starts.
    - A timer displays at the top of the page and shows the time remaining in the session.

      Tip: To refresh the session length at any time, choose **Start Lab** again before the timer reaches 00:00.

    - Before you continue, wait until the circle icon to the right of the AWS link in the upper-left corner turns green.

2. To connect to the AWS Management Console, choose the **AWS** link in the upper-left corner.
    - A new browser tab opens and connects you to the console.

      Tip: If a new browser tab does not open, a banner or icon is usually at the top of your browser with the message that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and then choose **Allow pop-ups**.

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

As a data engineer or analyst, you might not always know the schema of the data that you need to analyze. AWS Glue is designed for this situation. You can direct AWS Glue to data that is stored on AWS, and the service will discover your data. AWS Glue will then store the associated metadata (for example, the table definition and schema) in the AWS Glue Data Catalog. You accomplish this by creating a crawler, which inspects the data source and infers a schema based on the data.

In this task, you will work with data that is publicly available in an S3 bucket to do the following:

- Configure and create an AWS Glue crawler.
- Run the crawler to extract, transform, and load data into an AWS Glue database.
- Review the metadata of a table that the crawler created.
- Edit the schema of the table.

First, you will configure and create a crawler to discover the schema for the GHCN-D dataset and extract the data from it.

3. Configure and create the AWS Glue crawler.
    - In the AWS Management Console, in the search box next to **Services**, search for and choose **AWS Glue** to open the AWS Glue console.
    - In the navigation pane, under **Databases**, choose **Tables**.
    - Choose **Add tables using crawler**.
    - For **Name**, enter `Weather`
    - Expand the **Tags (optional)** section.

      Notice that this is where you could add tags or extra security configurations. Keep the default settings.

    - Choose **Next** at the bottom of the page.
    - Choose **Add a data source** and configure the following:
        - **Data source:** Choose **S3**.
        - **Location of S3 data:** Choose **In a different account**.

        - **S3 path:** Enter the following S3 bucket location for the publicly available dataset:

          ```
          s3://noaa-ghcn-pds/csv/by_year/
          ```

        - **Subsequent crawler runs:** Choose **Crawl all sub-folders**.
        - Choose **Add an S3 data source**.
    - Choose **Next**.
    - For **Existing IAM role**, choose **LabRole**.
4. Choose **Next**.
5. In the **Output configuration** section, choose **Add database**.

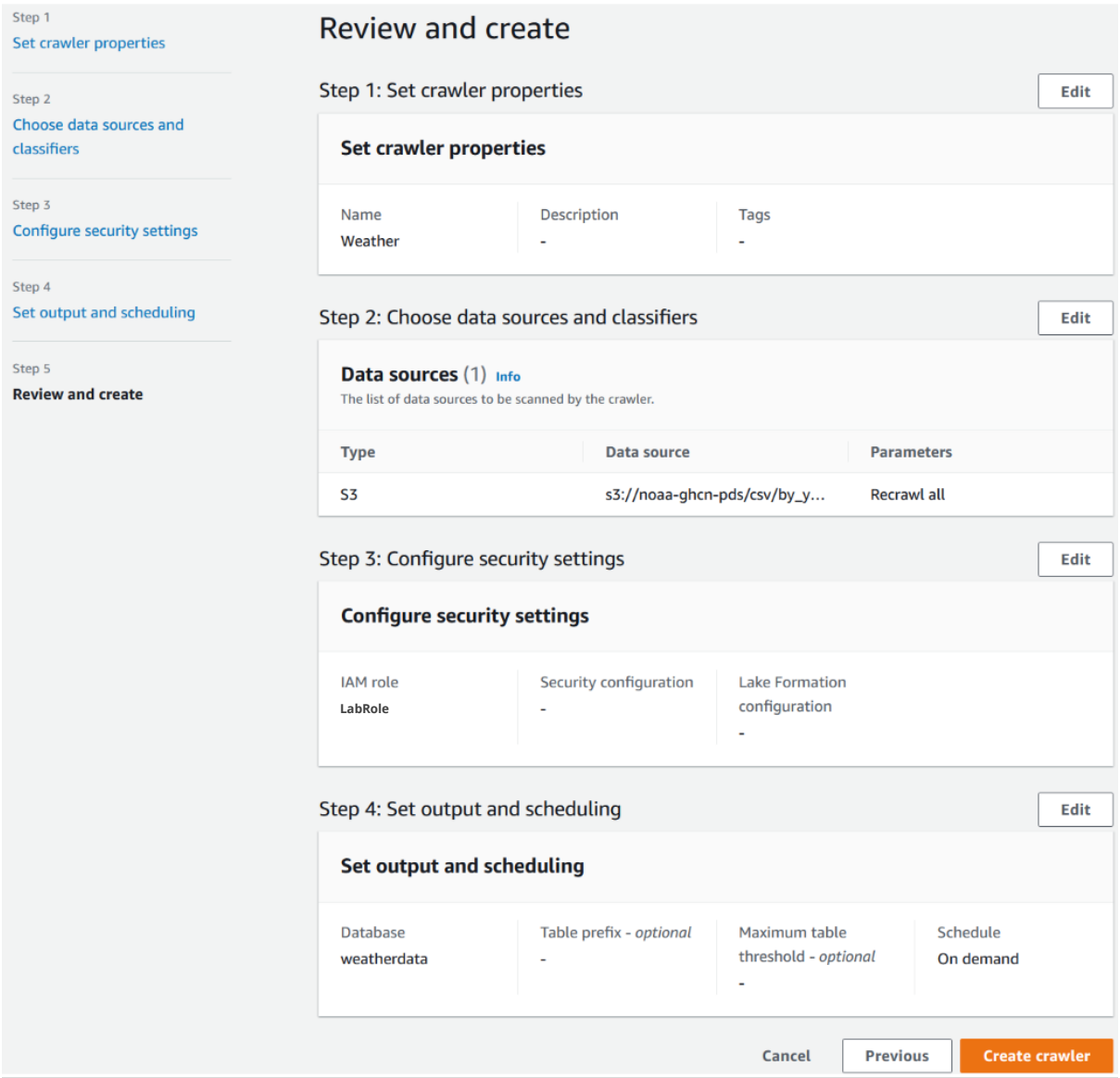
   A new browser tab opens.

6. For **Name**, enter `weatherdata`
7. Choose **Create database**.
8. Return to the browser tab that is open to the **Set output and scheduling** page in the AWS Glue console.
9. For **Target database**, choose the **weatherdata** database that you just created.

   Tip: To refresh the list of available databases, choose the refresh icon to the right of the dropdown list.

10. In the **Crawler schedule** section, for **Frequency**, keep the default **On demand**.
11. Choose **Next**.

12. Confirm your crawler configuration is similar to the following.



13. Choose **Create crawler**. To perform the extract and load steps of the ETL process, you will now run the crawler. You can create AWS Glue crawlers to either run on demand or on a set schedule. Because you created your crawler to run on demand, you must *run* the crawler to build the database and generate the metadata.

14. Run the crawler.

    - On the **Crawlers** page, select the **Weather** crawler that you just created.
    - Choose **Run**.

The crawler state changes to *Running*.

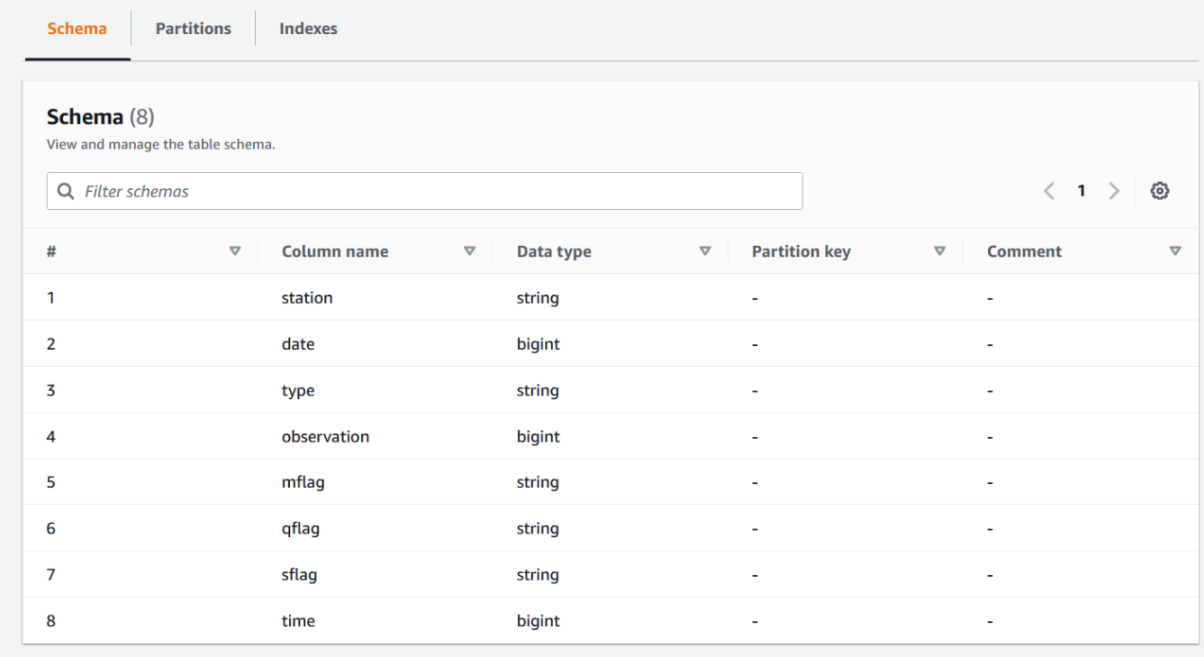
Important: Wait for the status to change to *Ready* before moving to the next step. This will take about 3 minutes.

AWS Glue creates a table to store metadata about the GHCN-D dataset. Next, you will inspect the data that AWS Glue captured about the data source.

Review the metadata that AWS Glue created.

- In the navigation pane to the left, under Data Catalog, choose **Databases**.
- Choose the link for the **weatherdata** database.
- In the **Tables** section, choose the **by_year** link.

You should see 8 columns. Not every column has an intuitive name. In the next step, we will use Athena to change the column names and convert the schema to the following:



#	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	-	-

Task 1 summary

In this task, you used the console to create a crawler in AWS Glue. You directed the crawler to data that is stored in an S3 bucket, and the crawler discovered the data. Then, the crawler stored the associated metadata (the table definition and

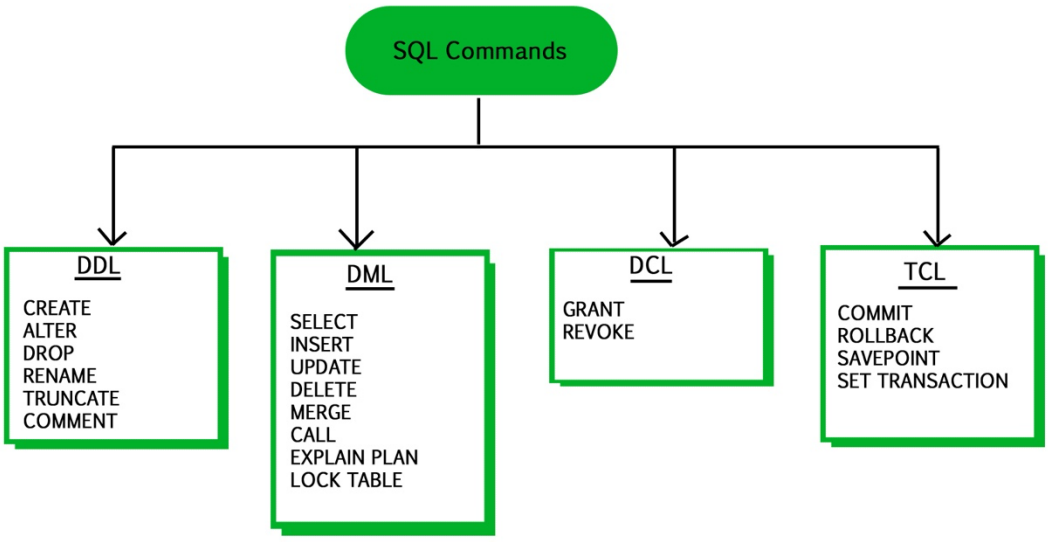
schema) in a Data Catalog. By using a crawler in AWS Glue, you can inspect a data source and infer its schema. You have now learnt that you can direct a Crawler to a public S3 bucket but also to a bucket on your own account.

The team can now use crawlers to inspect data sources quickly and reduce the manual steps to create database schemas from data sources in Amazon S3. You share the results of this POC with Mary, and she is happy with the new functionality.

Next, she wants to be able to do more analysis on the data in the AWS Glue Data Catalog.

Task 2: Querying a table by using Athena

Now that you created the Data Catalog, you can use the metadata to query the data even further by using Athena. You can also use Athena to modify the schema, create new views or temporary tables.



In this task, you will complete the following steps:

- Configure an S3 bucket to store Athena query results.
- Create an S3 bucket to store a converted version of your results.
- Preview a database table in Athena.
- Update the schema using DDL commands.
- Create a table for data after 1950.
- Run a query on selected data.

7. If you never run Athena this step is for you. If you have used Athena, you have already set this up.
    - Configure an S3 bucket to store Athena query results.
        1. In AWS Glue using the navigation pane, under **Databases**, choose **Tables**.
        2. Choose the link for the **by_year** table.
        3. Choose **Actions** > **View data**.
        4. When the pop-up appears to warn you that you will be taken to the Athena console, choose **Proceed**.

        The Athena console opens. At the bottom of the page, if this is the first time you are using Athena, you receive an error message that indicates that an output location was not provided. Before you run a query in Athena, you need to specify an S3 bucket to hold query results.

            1. Use the top search bar to search for S3
            2. Click on S3
            3. Go to Buckets
            4. In the buckets list, click on create bucket
            5. Select General Purpose, and in bucket name type: athena-results-yourfirstandlastnamelowercaseNoSpaces
            6. Click on Create Bucket
            7. Return to Athena
            8. Choose the **Settings** tab.
            9. Choose **Manage**.
            10. To the right of **Location of query result**, choose **Browse S3**.
            11. Choose the bucket name that is similar to the following: athena-results-yourfirstandlastnamelowercaseNoSpaces
            12. Select **Choose**.
            13. Keep the default settings for the other options, and choose **Save**.
8. Preview a table in Athena.
    - Choose the **Editor** tab.
    - In the **Data** panel on the left, notice that the **Data source** is **AwsDataCatalog**.
    - For **Database**, choose **weatherdata**.
    - In the **Tables** section, choose the ellipsis (three dot) icon for the **by_year** table, and then choose **Preview Table**.

Tip: To view the column names and their data types in this table, choose the icon to the left of the table name.

If you remember, we want to update the schema.

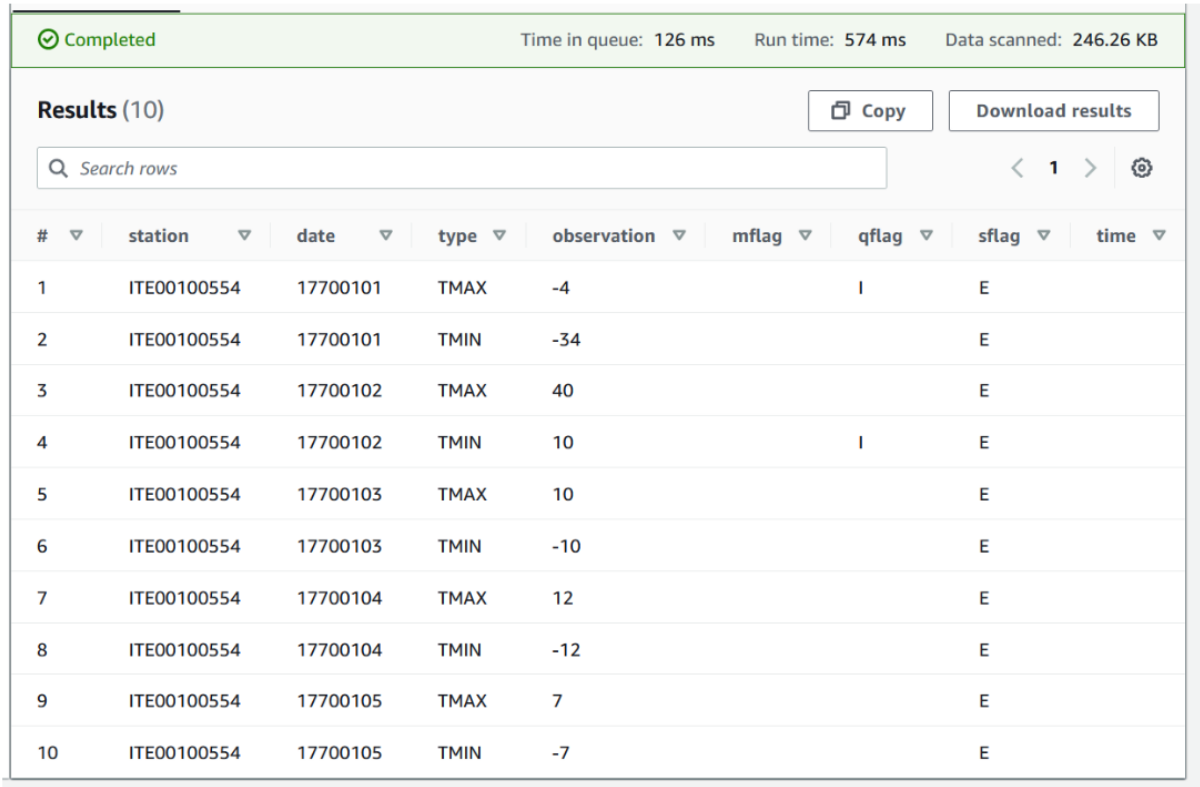
Previous Name	New Name
id	station
date	date
element	type
data_value	observation
m_flag	mflag
q_flag	qflag
s_flag	sflag
obs_time	time

You can use DDL Statements in Athena:

```
ALTER table by_year
REPLACE COLUMNS (station string, date bigint, type string, observation bigint,
mflag string, qflag string, sflag string, time bigint);
```

Now preview the table again:

The first 10 records from the *weatherdata* table display, similar to the following screenshot:



Completed | Time in queue: 126 ms | Run time: 574 ms | Data scanned: 246.26 KB

Results (10) | Copy | Download results

Search rows | 1

#	station	date	type	observation	mflag	qflag	sflag	time
1	ITE00100554	17700101	TMAX	-4		I	E	
2	ITE00100554	17700101	TMIN	-34			E	
3	ITE00100554	17700102	TMAX	40			E	
4	ITE00100554	17700102	TMIN	10		I	E	
5	ITE00100554	17700103	TMAX	10			E	
6	ITE00100554	17700103	TMIN	-10			E	
7	ITE00100554	17700104	TMAX	12			E	
8	ITE00100554	17700104	TMIN	-12			E	
9	ITE00100554	17700105	TMAX	7			E	
10	ITE00100554	17700105	TMIN	-7			E	

Notice the run time and amount of data that was scanned for the query. As you develop more complex applications, it is important to minimize resource consumption to optimize costs. You will see examples of how to optimize cost for Athena queries later in this task.

In the next step, you will create an AWS Glue database table that only includes data since 1950. To optimize your use of Athena, you will store data in the Apache Parquet format. Apache Parquet is an open-source columnar data format that is optimized for performance and storage.

First, let's create a S3 bucket where to store the data after 1950 in Apache Parquet format.

9. Create a table for data after 1950.

   First, you need to create a bucket for you to store this data.

   - In the search box next to **Services**, search for and choose **S3**.
   - In the **Buckets** list, click on create bucket.
   - Select General Purpose, and in bucket name type: **glue-1950-bucket-yourfirstnameandlastnameLowerCaseNoSpaces**

- Click on Create bucket
- Return to the Athena query editor.
- Copy and paste the following query into a query tab in the editor:

```
CREATE table weatherdata.late20th
WITH (
  format='PARQUET', external_location='s3://<bucket_name_you_just_created/'
) AS SELECT date, type, observation  FROM by_year
WHERE date/10000 between 1950 and 2015;
```

- Replace the S3 bucket name with the name you just created

10. **Important:** Don't choose the same bucket name that you set up in Athena Settings.

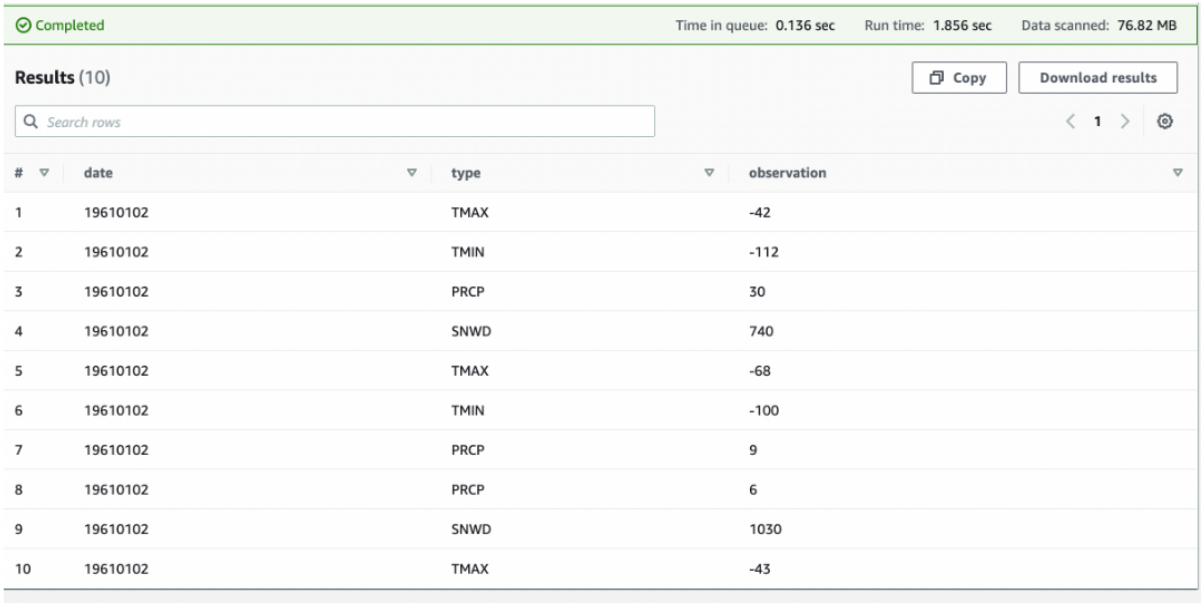
- Choose **Run**.

  After the query runs, the run time and data scanned values are similar to the following:

```
Time in queue: 128 ms
Run time: 1 min 8.324 sec
Data scanned: 98.44 GB
```

- To preview the results, in the **Tables** section, to the right of the *late20th* table, choose the ellipsis icon, and then choose **Preview Table**.

  The results are similar to the following screenshot.



#	date	type	observation
1	19610102	TMAX	-42
2	19610102	TMIN	-112
3	19610102	PRCP	30
4	19610102	SNWD	740
5	19610102	TMAX	-68
6	19610102	TMIN	-100
7	19610102	PRCP	9
8	19610102	PRCP	6
9	19610102	SNWD	1030
10	19610102	TMAX	-43

11. Now that you have isolated the data that you are interested in, you can write queries for further analysis.

12. Run a query on the new table.

    First, create a view that only includes the maximum temperature reading, or TMAX, value.

    - Run the following query in a new query tab:

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

    - To preview the results, in the **Views** section, to the right of the *tmax* view, choose the ellipsis icon, and then choose **Preview View**.

      The results are similar to the following screenshot:

Run the following query in a new query tab.

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

The purpose of this query is to calculate the average maximum temperature for each year in the dataset.

After the query runs, the run time and data scanned values are similar to the following:

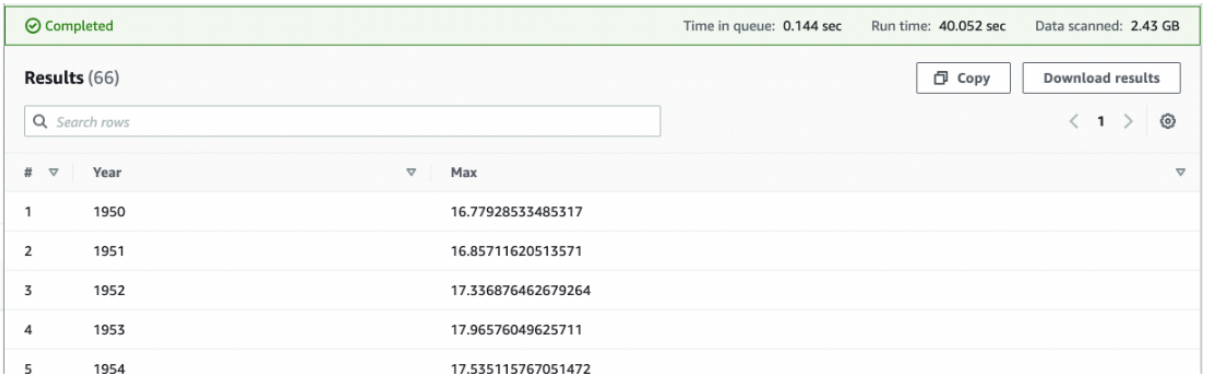
```
Time in queue: 0.211 sec
Run time: 25.109 sec
Data scanned: 2.45 GB
```

The results display the average maximum temperature for each year from 1950 to 2015. The following screenshot displays an example:

Remember that when you create queries with Athena, the results of the query must be stored back in Amazon S3. In a previous step, you specified the location in Amazon S3 where your queries are stored.

When using AWS services, you generally pay for what you use. However, because you reduced your query to only three columns of temperature data from 1950 through 2015, you reduced your costs for storage. Also, because you arranged the query data in columnar format with Apache Parquet, the time that it took to perform the queries in this task was reduced, resulting in less cost as you used fewer computational resources in Athena.

You show Mary that she can speed up her process by using AWS Glue in combination with Athena and still use views. She is delighted.



Now you can use Quicksight to create some useful visualizations to support Mary's analysis. Athena stores all the results of your queries in a CSV in an S3 bucket. You can download the CSV and upload it in Quicksight. You can experiment also on how to connect Quicksight directly to Athena by setting up Quicksight to pull the CSV from the result S3 bucket that Athena creates with the results of your query.

Task 2 summary

In this task, you learned how to use Athena to query tables in a database that an AWS Glue crawler created. You built a table for all data after 1950 from the original dataset. You used the Apache Parquet format to optimize your Athena queries, which reduced the time that it took to complete each query, resulting in less cost. After isolating this data, you created a view that calculated the average maximum temperature for each year.

AWS Glue integrates with original datasets stored in Amazon S3. AWS Glue can create a crawler to ingest the original dataset into a database and infer the appropriate schema. Then, you can quickly shift to Athena to develop queries and better understand your data. This integration reduces the time that it takes to derive insights from your data and apply these insights to make better decisions.