

AWS Data Exchange and Teradata Vantage

*Many Teradata customers are interested in integrating Teradata Vantage with Amazon Web Services (AWS) First Party Services. This guide will help you connect Teradata Vantage with the AWS Data Exchange service.*

*The procedure offered in this guide has been implemented and tested by Teradata. However, it is offered on an as-is basis. Neither AWS nor Teradata provide validation of Teradata Vantage with AWS Data Exchange.*

*This guide includes content from both AWS and Teradata product documentation.*

# Overview

This post describes the process to subscribe to a dataset in AWS Data Exchange, export it to Amazon S3, and then query it with Teradata Vantage, either leveraging Native Object Store (NOS) capability or the Teradata Parallel Transporter (TPT) utility.

## About AWS Data Exchange

AWS Data Exchange makes it easy to find, subscribe to, and use third-party data in the cloud. Today, AWS Data Exchange contains data products from a broad range of domains, including financial services, healthcare and life sciences, geospatial, weather, and mapping. Once subscribed to a data product, you can use the AWS Data Exchange API to export data into Amazon S3, and then use Teradata Vantage to explore and analyze the data.

## About Teradata Vantage

[Teradata Vantage](https://www.teradata.com/Vantage) is the cloud data analytics platform that unifies everything—data lakes, data warehouses, analytics, and new data sources and types. Leading the way with multi-cloud environments and priced for flexibility, Vantage leverages 100 percent of a business’s data, regardless of scale, volume, or complexity.

Vantage combines descriptive, predictive, prescriptive analytics, autonomous decision-making, machine learning (ML) functions, and visualization tools into a unified, integrated platform that uncovers real-time business intelligence at scale, no matter where the data resides.

Vantage enables companies to start small and elastically scale compute or storage, paying only for what they use, harnessing low-cost object stores and integrating their analytic workloads. Vantage supports R, Python, Teradata Studio, and any other SQL-based tools. You can deploy Vantage across public clouds, on-premises, on optimized or commodity infrastructure, or as-a-service.

Teradata Vantage Native Object Store (NOS) can be used to explore data in external object stores, like Amazon S3, using standard SQL. No special object storage-side compute infrastructure is required to use NOS. You can explore data located in an Amazon S3 bucket by simply creating a NOS table definition that points to your bucket. With NOS, you can quickly import data from Amazon S3 or even join it other tables in the database.

Alternatively, the Teradata Parallel Transporter (TPT) utility can be used to import data from Amazon S3 to Teradata Vantage in bulk fashion. Once loaded, data can be efficiently queried within Vantage.

# Prerequisites

You are expected to be familiar with the AWS Data Exchange service, Teradata Vantage, and (optionally) Teradata Tools and Utilities (TTU).

You will need the following accounts, and systems:

* An AWS account. You can start with a [free account](https://aws.amazon.com/getting-started/).
* An [AWS Access Key](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html#Using_CreateAccessKey) and Secret Access Key.
* An [Amazon S3 bucket](https://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-configure-bucket.html) to store the dataset

If you will be using NOS:

* A Teradata Vantage 2.0 instance (version 17.0) or later

If you will not be using NOS:

* A Teradata Vantage 1.1 instance (version 16.20) or later
* An account on the Teradata Developer website
* A client system to run Teradata Parallel Transporter scripts

# Procedure

Once you have met the prerequisites, follow these steps:

1. Create an Amazon S3 bucket
2. Subscribe to a Data Product on AWS Data Exchange
3. Export a dataset to your S3 bucket

If you have NOS capability, you may follow these steps:

1. Configure NOS access to S3
2. Query the dataset in S3
3. Load data from S3 into Vantage (optional)

If you do not have NOS capability or do not want to use NOS, follow these steps:

1. Configure Teradata Parallel Transporter
2. Load data from S3 into Vantage using TPT
3. Query the dataset in Vantage

# Create an Amazon S3 bucket

Open the [Amazon S3 console](https://s3.console.aws.amazon.com) in a browser. Chrome, Firefox, and Safari work well.

|  |
| --- |
| Amazon S3 is region sensitive. Ensure that your console reflects the region where you want to access a dataset.  AWS Data Exchange is also region sensitive. You will want to coordinate and collocate your Amazon S3 bucket and Data Exchange data in the same region. |

You may use an existing bucket. We will create a new bucket.

|  |
| --- |
| AWS is enhancing the Amazon S3 console. These screenshots might not match what you see exactly. |

Click the **Create bucket** button.

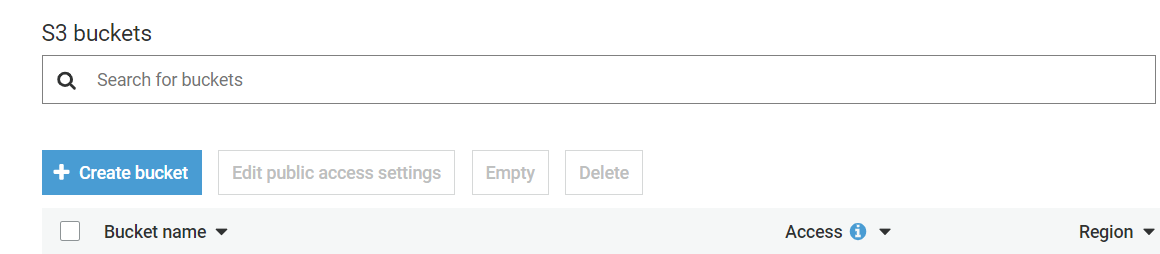


Figure 1 Create Amazon S3 bucket button

Enter a bucket name. We will use *dataexchange-dataset* in this article.

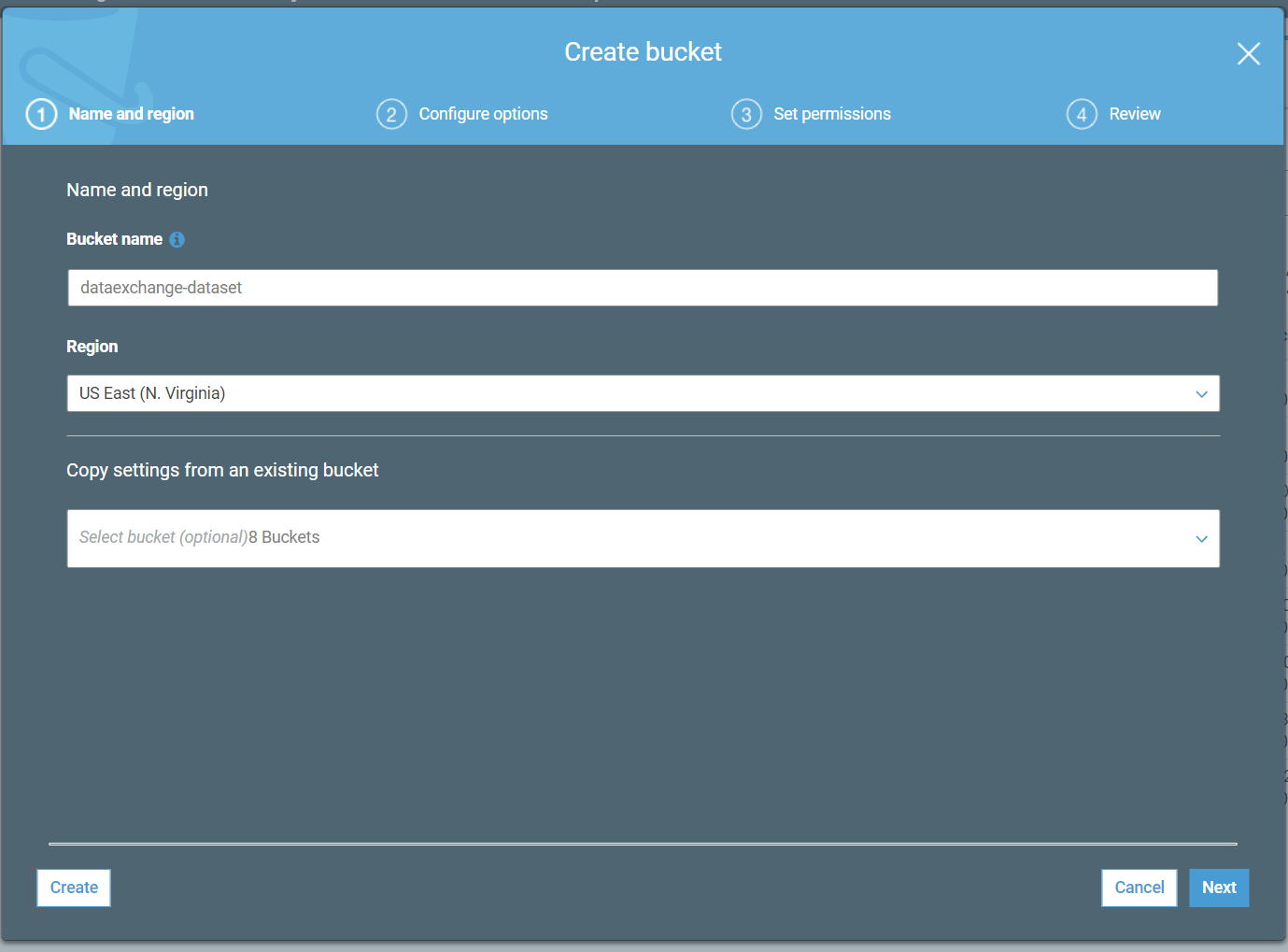


Figure 2 Create Amazon S3 bucket dialog

|  |
| --- |
| Amazon S3 bucket names must be globally unique. You will not be able to use the same bucket name as in this article. Instead, you will need to create a different bucket name and use it in place of our bucket name throughout this article. |

Click the **Create** button.

|  |
| --- |
| It is beyond the scope of this article to discuss permissions for the Amazon S3 bucket needed by Vantage. |

# Subscribe to a Data Product on AWS Data Exchange

Open the [AWS Data Exchange console](https://console.aws.amazon.com/dataexchange) in a browser.

Click on **Explore available data products** to search for a dataset.

Enter “*historical weather data*” and click the **Search** button.

Scroll down to the “*OnPoint Historical Weather Data*” dataset. Ensure that the price is **Free**.

|  |
| --- |
| Datasets are products, many with recurring fees. Some can be quite expensive (multiple thousands of dollars).  Ensure that you are selecting a free dataset for this article to avoid being charged. |

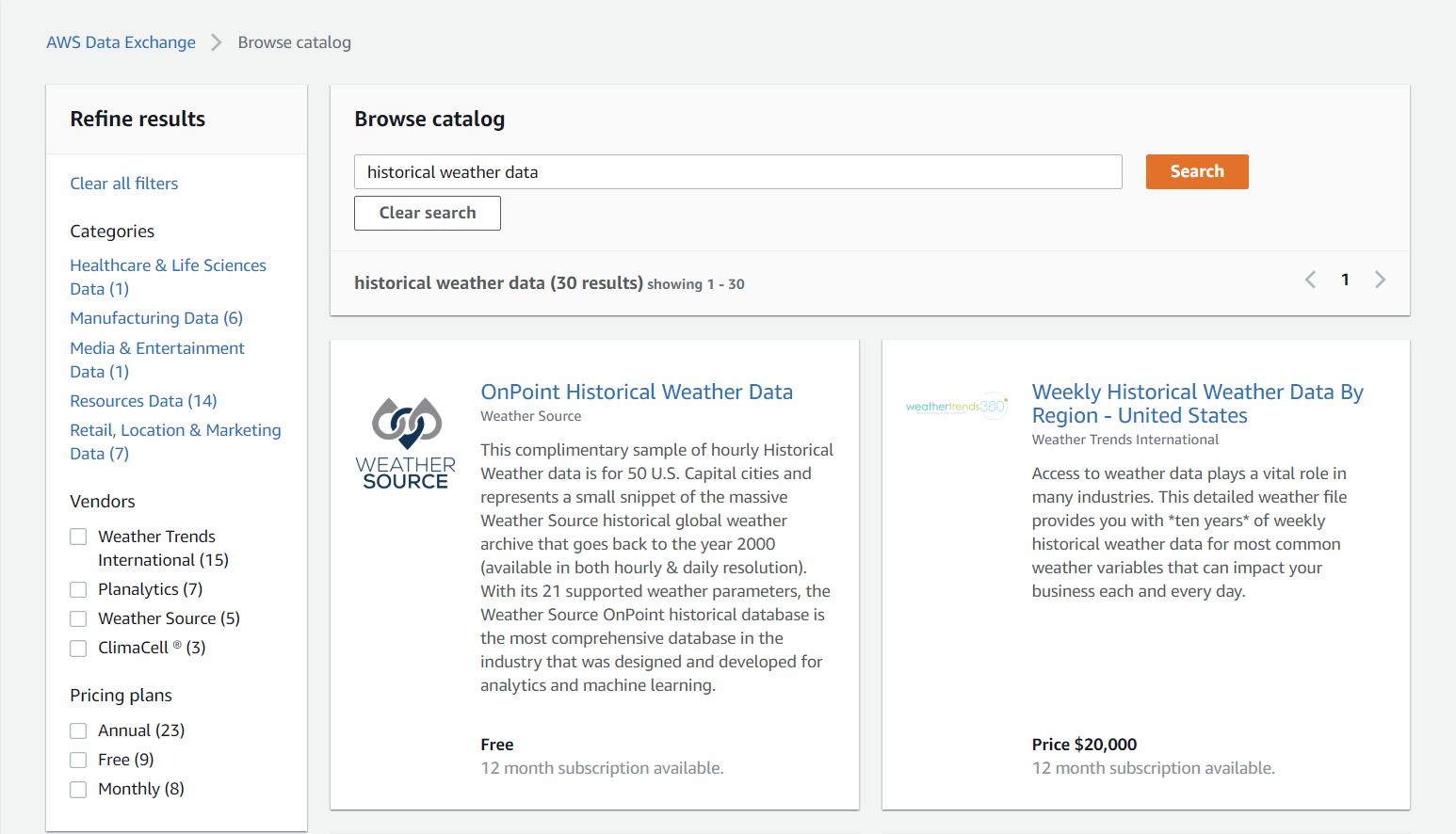


Figure 3 OnPoint Historical Weather Data dataset in AWS Data Exchange

Click on the **OnPoint Historical Weather Data** dataset.

Click on **Continue to subscribe**.

Ensure that the price is zero.

Under *Renewal settings*, click on **No** to avoid being charged in the future.

­­A screenshot of a social media post

Description automatically generated

Figure 4 Dataset pricing and renewal settings

Read the *Data subscription agreement*.

Scroll down and click on **Subscribe**. The request will take a few minutes.

You will see a message when your subscription has been successfully processed.

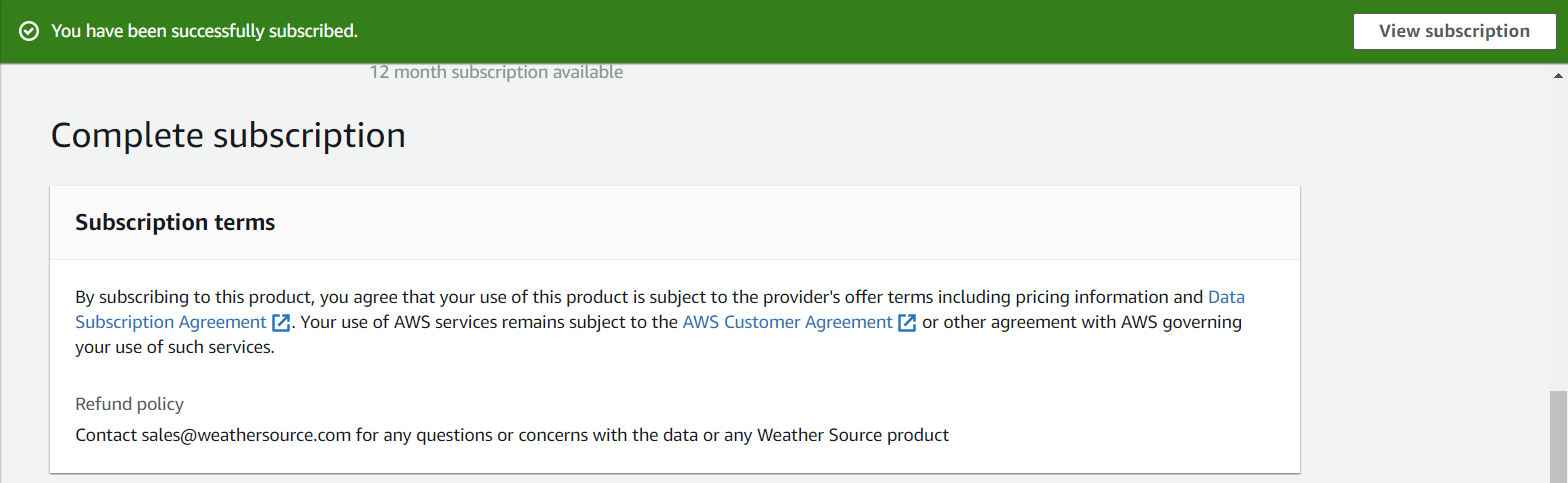


Figure 5 Successful subscription

# Export Dataset to Amazon S3

In the AWS Data Exchange console, click on **My Subscriptions** > **Subscriptions**.

You will see the *OnPoint Historical Weather Data* subscription (plus any others that you have subscribed to).

Click on the **OnPoint Historical Weather Data** subscription.

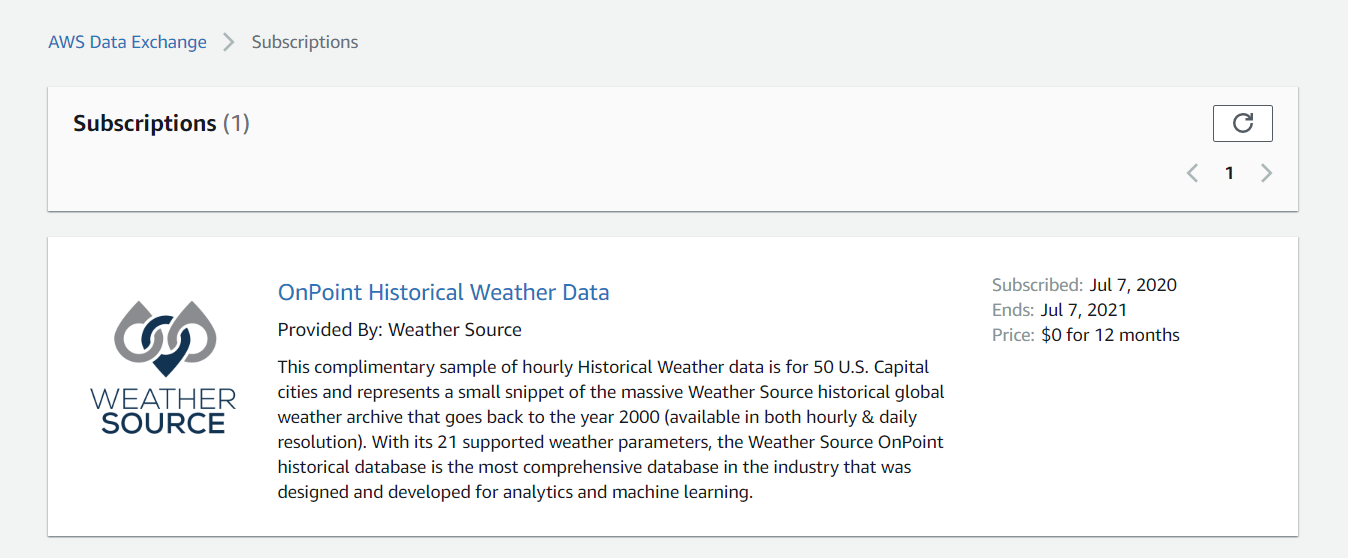


Figure 6 OnPoint Historical Weather Data subscription

Scroll down to *Entitled data sets*.

Click on **2018 Historical Hourly Weather for 50 U.S. Capital Cities** dataset.

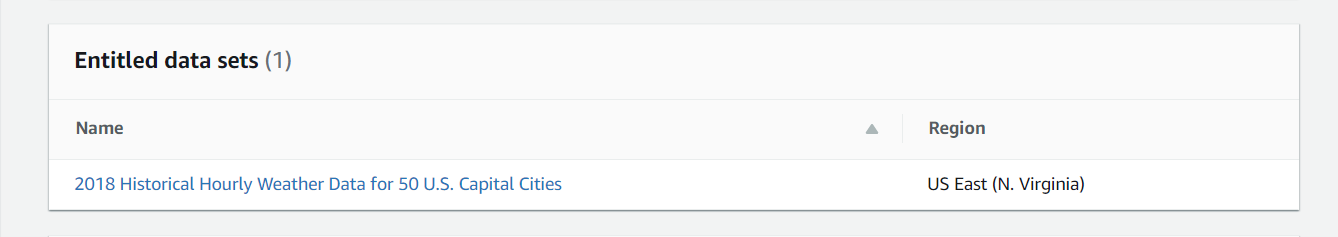


Figure 7 Entitled dataset

Click on the *Nov 5, 2019* revision.

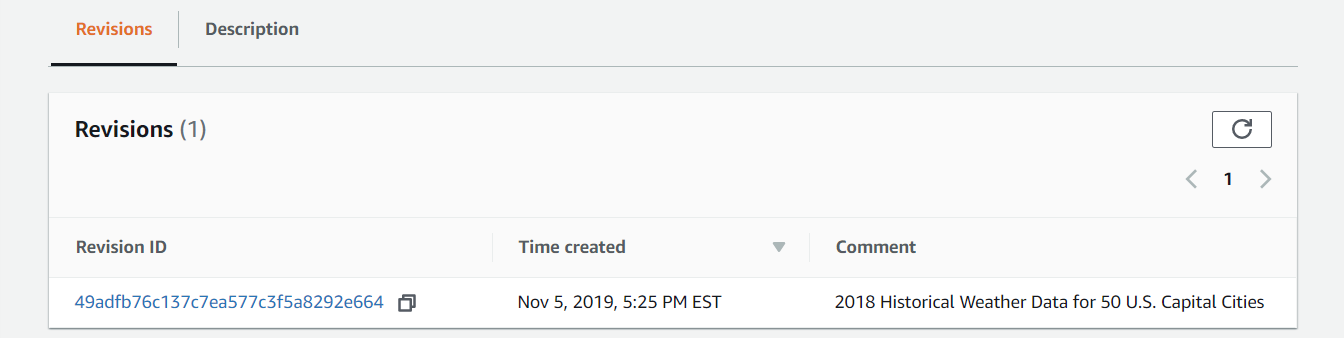


Figure 8 Nov 5, 2019 dataset revision

Select the checkbox next to the *Nov 5, 2019* asset.

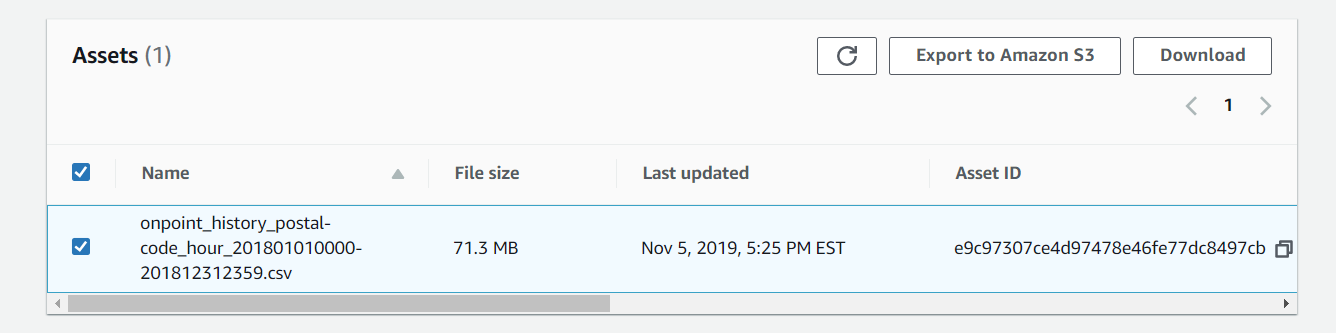


Figure 9 Nov 5, 2019 asset

Click on the **Export to Amazon S3** button.

Select the Amazon S3 bucket you created. You can leave the encryption set to *None*.

Click the **Export** button.

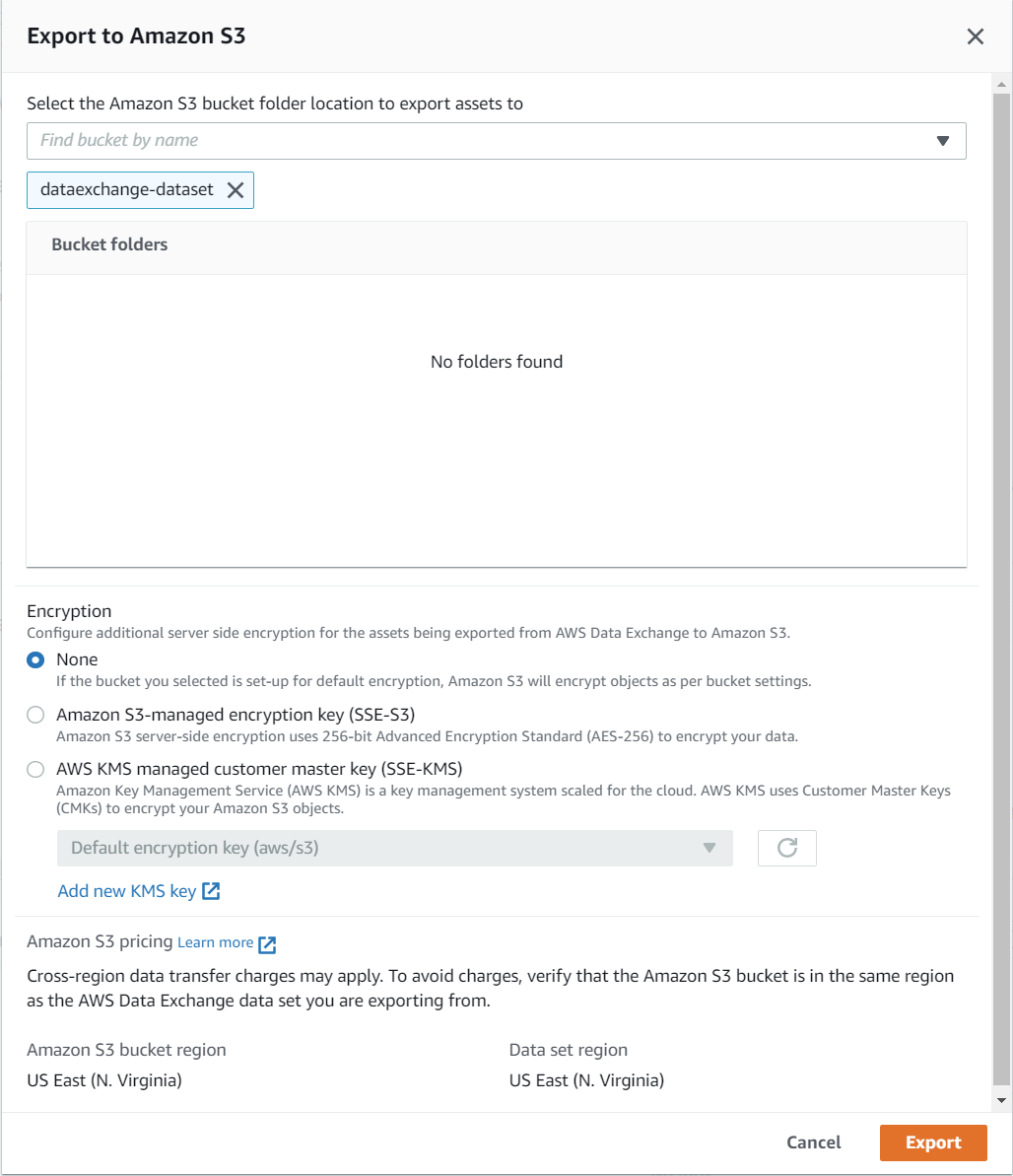


Figure 10 Export to Amazon S3

The export should take less than a minute. You will see a success message when the export is completed.

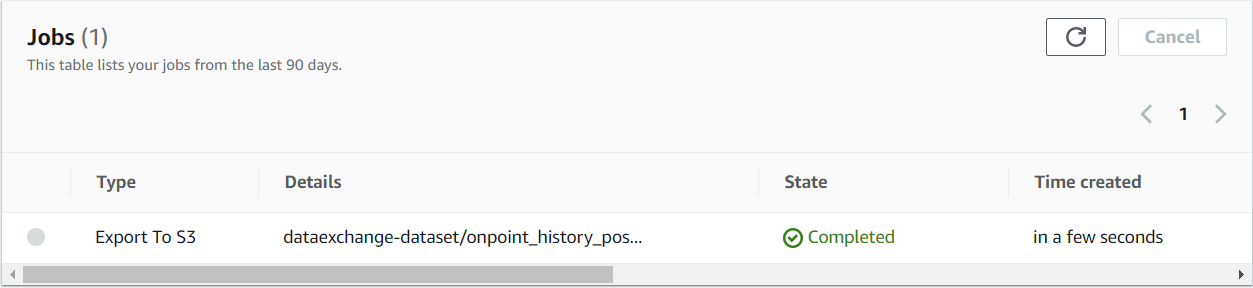


Figure 11 Export to S3 success

You can view the CSV object in your Amazon S3 bucket.

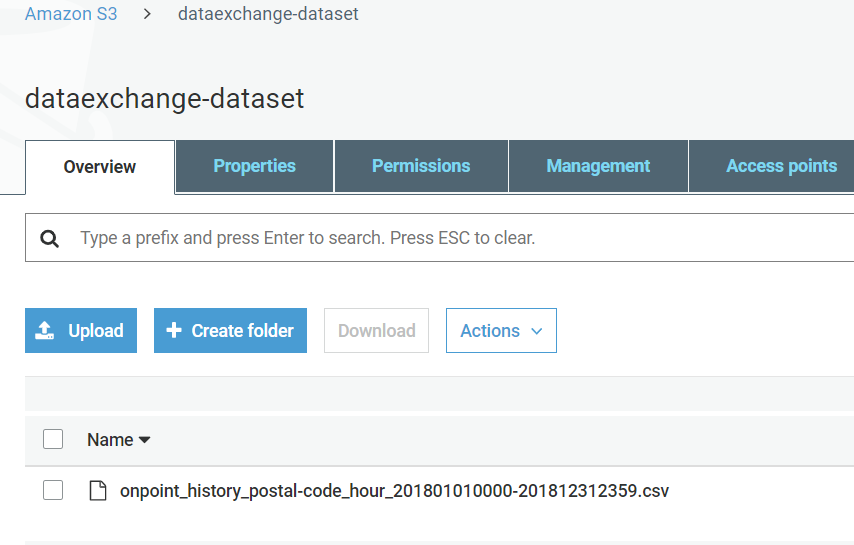


Figure 12 CSV object stored in Amazon S3 bucket

# Configure NOS access to Amazon S3

|  |
| --- |
| You can choose to use Native Object Store (NOS) to directly read objects on Amazon S3, or you can load the data from Amazon S3 into Vantage with the Teradata Parallel Transporter utility.  NOS is only available with Vantage 2.0 (version 17.0) or later. Teradata Parallel Transporter (TPT) is available with all available versions of Vantage.  If you are using TPT, please skip to step 7. |

Native Object Store (NOS) can directly read data in Amazon S3, which allows you to explore and analyze data in Amazon S3 without explicitly loading the data.

## Create a foreign table definition

A foreign table definition allows data in Amazon S3 to be easily referenced within the Advanced SQL Engine and makes the data available in a structured, relational format.

|  |
| --- |
| NOS supports data in CSV, JSON, and Parquet formats. |

Login to your Vantage system with Teradata Studio.

Create an AUTHORIZATION object to access your Amazon S3 bucket with the following SQL command.

|  |
| --- |
| CREATE AUTHORIZATION DefAuth\_S3 |
| AS DEFINER TRUSTED |
| USER 'A\*\*\*\*\*' /\* AccessKeyId \*/ |
| PASSWORD '\*\*\*\*\*' /\* SecretAccessKey \*/ |
| ; |

Replace the string for *USER* with your AWS access key.

Replace the string for *PASSWORD* with your AWS secret access key.

Create a foreign table definition for the CSV file on Amazon S3 with the following SQL command.

|  |
| --- |
| CREATE MULTISET FOREIGN TABLE WeatherData, |
| EXTERNAL SECURITY DEFINER TRUSTED DefAuth\_S3 |
| ( |
| Location VARCHAR(2048) CHARACTER SET UNICODE CASESPECIFIC, |
| Payload DATASET INLINE LENGTH 64000 STORAGE FORMAT CSV |
| ) |
| USING |
| ( |
| LOCATION ('/S3/s3.amazonaws.com/dataexchange-dataset/') |
| ) |
| ; |

At a minimum, the foreign table definition must include a table name (WeatherData) and a location clause, which points to the object store data.

The *LOCATION* requires a top-level name, which is the bucket name. This is highlighted above in yellow. You will need to replace this with your own bucket name.

If the object doesn’t have a standard extension (e.g. “.json”, “.csv”, “.parquet”), then the *Location…Payload* columns definition phrase is also needed, and the LOCATION phase need to include the file name. For example: LOCATiON (‘S3/s3.amazonaws.com/dataexchange-dataset/samplefile’).

Foreign tables are always defined as No Primary Index (NoPI) tables.

# Query the Dataset in Amazon S3

Run the following SQL command to query the dataset.

|  |
| --- |
| SELECT \* FROM WeatherData SAMPLE 10; |

The foreign table only contains two columns: Location and Payload. Location is the address in the object store system. The data itself is represented in the payload column, with the payload value within each record in the foreign table representing a single CSV row.

A screen shot of a person

Description automatically generated

Figure 13 WeatherData table

Run the following SQL command to focus on the data in the object.

|  |
| --- |
| SELECT payload..\* FROM WeatherData SAMPLE 10; |

A picture containing building, front, large, person

Description automatically generated

Figure 14 WeatherData table payload

## Create a view

Views can simplify the names associated with the payload attributes, can make it easier to code SQL against the object data, and can hide the Location references in the foreign table.

|  |
| --- |
| Vantage foreign tables use the “..” (double dot or double period) operator to separate the object name from the column name. |

Run the following SQL command to create a view.

|  |
| --- |
| REPLACE VIEW WeatherData\_view AS ( |
| SELECT |
| CAST(payload..postal\_code AS VARCHAR(10)) Postal\_code, |
| CAST(payload..country AS CHAR(2)) Country, |
| CAST(payload..time\_valid\_utc AS TIMESTAMP(0) FORMAT 'YYYY-MM-DDBHH:MI:SS') Time\_Valid\_UTC, |
| CAST(payload..doy\_utc AS INTEGER) DOY\_UTC, |
| CAST(payload..hour\_utc AS INTEGER) Hour\_UTC, |
| CAST(payload..time\_valid\_lcl AS TIMESTAMP(0) FORMAT 'YYYY-MM-DDBHH:MI:SS') Time\_Valid\_LCL, |
| CAST(payload..dst\_offset\_minutes AS INTEGER) DST\_Offset\_Minutes, |
| CAST(payload..temperature\_air\_2m\_f AS DECIMAL(4,1)) Temperature\_Air\_2M\_F, |
| CAST(payload..temperature\_wetbulb\_2m\_f AS DECIMAL(3,1)) Temperature\_Wetbulb\_2M\_F, |
| CAST(payload..temperature\_dewpoint\_2m\_f AS DECIMAL(3,1)) Temperature\_Dewpoint\_2M\_F, |
| CAST(payload..temperature\_feelslike\_2m\_f AS DECIMAL(4,1)) Temperature\_Feelslike\_2M\_F, |
| CAST(payload..temperature\_windchill\_2m\_f AS DECIMAL(4,1)) Temperature\_Windchill\_2M\_F, |
| CAST(payload..temperature\_heatindex\_2m\_f AS DECIMAL(4,1)) Temperature\_Heatindex\_2M\_F, |
| CAST(payload..humidity\_relative\_2m\_pct AS DECIMAL(3,1)) Humidity\_Relative\_2M\_Pct, |
| CAST(payload..humidity\_specific\_2m\_gpkg AS DECIMAL(3,1)) Humdity\_Specific\_2M\_GPKG, |
| CAST(payload..pressure\_2m\_mb AS DECIMAL(5,1)) Pressure\_2M\_Mb, |
| CAST(payload..pressure\_tendency\_2m\_mb AS DECIMAL(2,1)) Pressure\_Tendency\_2M\_Mb, |
| CAST(payload..pressure\_mean\_sea\_level\_mb AS DECIMAL(5,1)) Pressure\_Mean\_Sea\_Level\_Mb, |
| CAST(payload..wind\_speed\_10m\_mph AS DECIMAL(3,1)) Wind\_Speed\_10M\_MPH, |
| CAST(payload..wind\_direction\_10m\_deg AS DECIMAL(4,1)) Wind\_Direction\_10M\_Deg, |
| CAST(payload..wind\_speed\_80m\_mph AS DECIMAL(3,1)) Wind\_Speed\_80M\_MPH, |
| CAST(payload..wind\_direction\_80m\_deg AS DECIMAL(4,1)) Wind\_Direction\_80M\_Deg, |
| CAST(payload..wind\_speed\_100m\_mph AS DECIMAL(3,1)) Wind\_Speed\_100M\_MPH, |
| CAST(payload..wind\_direction\_100m\_deg AS DECIMAL(4,1)) Wind\_Direction\_100M\_Deg, |
| CAST(payload..precipitation\_in AS DECIMAL(3,2)) Precipitation\_in, |
| CAST(payload..snowfall\_in AS DECIMAL(3,2)) Snowfall\_in, |
| CAST(payload..cloud\_cover\_pct AS INTEGER) Cloud\_Cover\_Pct, |
| CAST(payload..radiation\_solar\_total\_wpm2 AS DECIMAL(5,1)) Radiation\_Solar\_Total\_WPM2 |
| FROM WeatherData |
| ) |
| ; |

Run the following SQL command to validate the view.

|  |
| --- |
| SELECT \* FROM WeatherData\_view SAMPLE 10; |

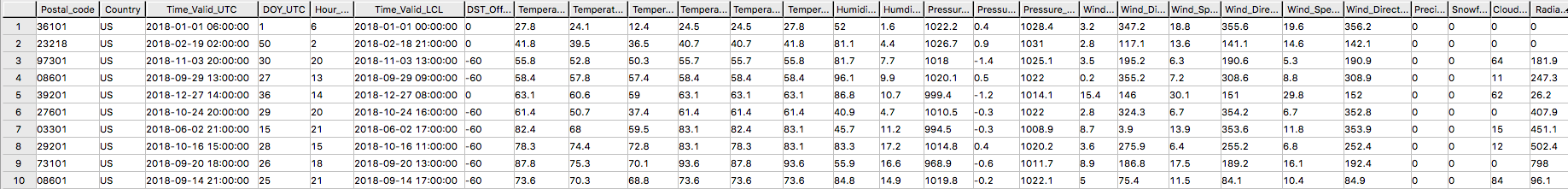


Figure 15 WeatherData\_view

Now that you have created a view, you can easily reference the object store data in a query and combine it with other tables, both relational tables in Vantage as well as foreign tables in an object store. This allows you to leverage the full analytic capabilities of Vantage on 100% of the data, no matter where the data is located.

# Load Data from Amazon S3 into Vantage (optional)

Having a persistent copy of the Amazon S3 data can be useful when repetitive access of the same data is expected. NOS does not automatically make a persistent copy of the Amazon S3 data. Each time you reference a foreign table, Vantage will fetch the data from Amazon S3. (Some data may be cached, but this depends on the size of the data in Amazon S3 and other active workloads in Vantage.)

In addition, you may be charged network fees for data transferred from Amazon S3. If you will be referencing the data in Amazon S3 multiple times, you may reduce your cost by loading it into Vantage, even temporarily.

You can select among the approaches below to load the data into Vantage.

## Create the table and load the data in a single statement

You can use a single statement to both create the table and load the data. You can choose the desired attributes from the foreign table payload and what they will be called in the relational table.

A **CREATE TABLE AS … WITH DATA** statement can be used with the foreign table definition as the source table.

Run the following SQL command to create the relational table and load the data.

|  |
| --- |
| CREATE MULTISET TABLE WeatherData\_temp AS ( |
| SELECT |
| CAST(payload..postal\_code AS VARCHAR(10)) Postal\_code, |
| CAST(payload..country AS CHAR(2)) Country, |
| CAST(payload..time\_valid\_utc AS TIMESTAMP(0) FORMAT 'YYYY-MM-DDBHH:MI:SS') Time\_Valid\_UTC, |
| CAST(payload..doy\_utc AS INTEGER) DOY\_UTC, |
| CAST(payload..hour\_utc AS INTEGER) Hour\_UTC, |
| CAST(payload..time\_valid\_lcl AS TIMESTAMP(0) FORMAT 'YYYY-MM-DDBHH:MI:SS') Time\_Valid\_LCL, |
| CAST(payload..dst\_offset\_minutes AS INTEGER) DST\_Offset\_Minutes, |
| CAST(payload..temperature\_air\_2m\_f AS DECIMAL(4,1)) Temperature\_Air\_2M\_F, |
| CAST(payload..temperature\_wetbulb\_2m\_f AS DECIMAL(3,1)) Temperature\_Wetbulb\_2M\_F, |
| CAST(payload..temperature\_dewpoint\_2m\_f AS DECIMAL(3,1)) Temperature\_Dewpoint\_2M\_F, |
| CAST(payload..temperature\_feelslike\_2m\_f AS DECIMAL(4,1)) Temperature\_Feelslike\_2M\_F, |
| CAST(payload..temperature\_windchill\_2m\_f AS DECIMAL(4,1)) Temperature\_Windchill\_2M\_F, |
| CAST(payload..temperature\_heatindex\_2m\_f AS DECIMAL(4,1)) Temperature\_Heatindex\_2M\_F, |
| CAST(payload..humidity\_relative\_2m\_pct AS DECIMAL(3,1)) Humidity\_Relative\_2M\_Pct, |
| CAST(payload..humidity\_specific\_2m\_gpkg AS DECIMAL(3,1)) Humdity\_Specific\_2M\_GPKG, |
| CAST(payload..pressure\_2m\_mb AS DECIMAL(5,1)) Pressure\_2M\_Mb, |
| CAST(payload..pressure\_tendency\_2m\_mb AS DECIMAL(2,1)) Pressure\_Tendency\_2M\_Mb, |
| CAST(payload..pressure\_mean\_sea\_level\_mb AS DECIMAL(5,1)) Pressure\_Mean\_Sea\_Level\_Mb, |
| CAST(payload..wind\_speed\_10m\_mph AS DECIMAL(3,1)) Wind\_Speed\_10M\_MPH, |
| CAST(payload..wind\_direction\_10m\_deg AS DECIMAL(4,1)) Wind\_Direction\_10M\_Deg, |
| CAST(payload..wind\_speed\_80m\_mph AS DECIMAL(3,1)) Wind\_Speed\_80M\_MPH, |
| CAST(payload..wind\_direction\_80m\_deg AS DECIMAL(4,1)) Wind\_Direction\_80M\_Deg, |
| CAST(payload..wind\_speed\_100m\_mph AS DECIMAL(3,1)) Wind\_Speed\_100M\_MPH, |
| CAST(payload..wind\_direction\_100m\_deg AS DECIMAL(4,1)) Wind\_Direction\_100M\_Deg, |
| CAST(payload..precipitation\_in AS DECIMAL(3,2)) Precipitation\_in, |
| CAST(payload..snowfall\_in AS DECIMAL(3,2)) Snowfall\_in, |
| CAST(payload..cloud\_cover\_pct AS INTEGER) Cloud\_Cover\_Pct, |
| CAST(payload..radiation\_solar\_total\_wpm2 AS DECIMAL(5,1)) Radiation\_Solar\_Total\_WPM2 |
| FROM WeatherData |
| Where Postal\_Code = '36101' |
| ) |
| WITH DATA |
| NO PRIMARY INDEX |
| ; |

Run the following SQL command to validate the contents of the table.

|  |
| --- |
| SELECT \* FROM WeatherData\_temp SAMPLE 10; |

A close up of a piece of paper

Description automatically generated

Figure 16 WeatherData\_temp

## Create the table and load the data in multiple statements

You can also use multiple statements to first create the relational table and then load the data. An advantage of this choice is that you can perform multiple loads, possibly selecting different data or loading in smaller increments if the object is very large.

Run the following SQL command to create the relational table.

|  |
| --- |
| CREATE MULTISET TABLE WeatherData\_temp ( |
| Postal\_code VARCHAR(10), |
| Country CHAR(2), |
| Time\_Valid\_UTC TIMESTAMP(0) FORMAT ‘YYYY-MM-DDBHH:MI:SS’, |
| DOY\_UTC INTEGER, |
| Hour\_UTC INTEGER, |
| Time\_Valid\_LCL TIMESTAMP(0) FORMAT ‘YYYY-MM-DDBHH:MI:SS’, |
| DST\_Offset\_Minutes INTEGER, |
| Temperature\_Air\_2M\_F DECIMAL(4,1), |
| Temperature\_Wetbulb\_2M\_F DECIMAL(3,1), |
| Temperature\_Dewpoint\_2M\_F DECIMAL(3,1), |
| Temperature\_Feelslike\_2M\_F DECIMAL(4,1), |
| Temperature\_Windchill\_2M\_F DECIMAL(4,1), |
| Temperature\_Heatindex\_2M\_F DECIMAL(4,1), |
| Humidity\_Relative\_2M\_Pct DECIMAL(3,1), |
| Humdity\_Specific\_2M\_GPKG DECIMAL(3,1), |
| Pressure\_2M\_Mb DECIMAL(5,1), |
| Pressure\_Tendency\_2M\_Mb DECIMAL(2,1), |
| Pressure\_Mean\_Sea\_Level\_Mb DECIMAL(5,1), |
| Wind\_Speed\_10M\_MPH DECIMAL(3,1), |
| Wind\_Direction\_10M\_Deg DECIMAL(4,1), |
| Wind\_Speed\_80M\_MPH DECIMAL(3,1), |
| Wind\_Direction\_80M\_Deg DECIMAL(4,1), |
| Wind\_Speed\_100M\_MPH DECIMAL(3,1), |
| Wind\_Direction\_100M\_Deg DECIMAL(4,1), |
| Precipitation\_in DECIMAL(3,2), |
| Snowfall\_in DECIMAL(3,2), |
| Cloud\_Cover\_Pct INTEGER, |
| Radiation\_Solar\_Total\_WPM2 DECIMAL(5,1) |
| ) |
| UNIQUE PRIMARY INDEX ( Postal\_Code, Time\_Valid\_UTC ) |
| ; |

Run the following SQL to load the data into the table.

|  |
| --- |
| INSERT INTO WeatherData\_temp |
| SELECT |
| CAST(payload..postal\_code AS VARCHAR(10)) Postal\_code, |
| CAST(payload..country AS CHAR(2)) Country, |
| CAST(payload..time\_valid\_utc AS TIMESTAMP(0) FORMAT 'YYYY-MM-DDBHH:MI:SS') Time\_Valid\_UTC, |
| CAST(payload..doy\_utc AS INTEGER) DOY\_UTC, |
| CAST(payload..hour\_utc AS INTEGER) Hour\_UTC, |
| CAST(payload..time\_valid\_lcl AS TIMESTAMP(0) FORMAT 'YYYY-MM-DDBHH:MI:SS') Time\_Valid\_LCL, |
| CAST(payload..dst\_offset\_minutes AS INTEGER) DST\_Offset\_Minutes, |
| CAST(payload..temperature\_air\_2m\_f AS DECIMAL (4,1)) Temperature\_Air\_2M\_F, |
| CAST(payload..temperature\_wetbulb\_2m\_f AS DECIMAL(3,1)) Temperature\_Wetbulb\_2M\_F, |
| CAST(payload..temperature\_dewpoint\_2m\_f AS DECIMAL(3,1)) Temperature\_Dewpoint\_2M\_F, |
| CAST(payload..temperature\_feelslike\_2m\_f AS DECIMAL(4,1)) Temperature\_Feelslike\_2M\_F, |
| CAST(payload..temperature\_windchill\_2m\_f AS DECIMAL(4,1)) Temperature\_Windchill\_2M\_F, |
| CAST(payload..temperature\_heatindex\_2m\_f AS DECIMAL(4,1)) Temperature\_Heatindex\_2M\_F, |
| CAST(payload..humidity\_relative\_2m\_pct AS DECIMAL(3,1)) Humidity\_Relative\_2M\_Pct, |
| CAST(payload..humidity\_specific\_2m\_gpkg AS DECIMAL(3,1)) Humdity\_Specific\_2M\_GPKG, |
| CAST(payload..pressure\_2m\_mb AS DECIMAL(5,1)) Pressure\_2M\_Mb, |
| CAST(payload..pressure\_tendency\_2m\_mb AS DECIMAL(2,1)) Pressure\_Tendency\_2M\_Mb, |
| CAST(payload..pressure\_mean\_sea\_level\_mb AS DECIMAL(5,1)) Pressure\_Mean\_Sea\_Level\_Mb, |
| CAST(payload..wind\_speed\_10m\_mph AS DECIMAL(3,1)) Wind\_Speed\_10M\_MPH, |
| CAST(payload..wind\_direction\_10m\_deg AS DECIMAL(4,1)) Wind\_Direction\_10M\_Deg, |
| CAST(payload..wind\_speed\_80m\_mph AS DECIMAL(3,1)) Wind\_Speed\_80M\_MPH, |
| CAST(payload..wind\_direction\_80m\_deg AS DECIMAL(4,1)) Wind\_Direction\_80M\_Deg, |
| CAST(payload..wind\_speed\_100m\_mph AS DECIMAL(3,1)) Wind\_Speed\_100M\_MPH, |
| CAST(payload..wind\_direction\_100m\_deg AS DECIMAL(4,1)) Wind\_Direction\_100M\_Deg, |
| CAST(payload..precipitation\_in AS DECIMAL(3,2)) Precipitation\_in, |
| CAST(payload..snowfall\_in AS DECIMAL(3,2)) Snowfall\_in, |
| CAST(payload..cloud\_cover\_pct AS INTEGER) Cloud\_Cover\_Pct, |
| CAST(payload..radiation\_solar\_total\_wpm2 AS DECIMAL(5,1)) Radiation\_Solar\_Total\_WPM2 |
| FROM WeatherData |
| WHERE Postal\_Code = '30301' |
| ; |

Run the following SQL command to validate the contents of the table.

|  |
| --- |
| SELECT \* FROM WeatherData\_temp SAMPLE 10; |

A picture containing large

Description automatically generated

Figure 17 WeatherData\_temp

## READ\_NOS – An alternative method for foreign tables

An alternative to defining a foreign table is to use the READ\_NOS table operator. This table operator allows you to access data directly from an object store without first creating a foreign table, or to view a list of the keys associated with all the objects specified by a Location clause.

You can use the READ\_NOS table operator to explore the data in an object.

Run the following command to explore the data in an object.

|  |
| --- |
| SELECT TOP 5 payload..\* |
| FROM READ\_NOS ( |
| ON ( |
| SELECT CAST( NULL AS DATASET STORAGE FORMAT CSV)) |
| USING |
| LOCATION ('/S3/s3.amazonaws.com/dataexchange-dataset/') |
| ACCESS\_ID('A\*\*\*\*\*') |
| ACCESS\_KEY('\*\*\*\*\*') |
| ) AS THE\_TABLE |
| ORDER BY 1 |
| ; |

The *LOCATION* requires a top-level name, which is the bucket name. This is highlighted above in yellow. You will need to replace this with your own bucket name.

Replace the string for *ACCESS\_ID* with your AWS access key.

Replace the string for *ACCESS\_KEY* with your AWS secret access key.

A screenshot of a cell phone

Description automatically generated

Figure 18 READ\_NOS

You can also leverage the READ\_NOS table operator to get the length (size) of the object.

Run the following SQL command to view the size of the object.

|  |
| --- |
| SELECT location(CHAR(120)), ObjectLength |
| FROM READ\_NOS ( |
| ON ( |
| SELECT CAST( NULL AS DATASET STORAGE FORMAT CSV)) |
| USING |
| LOCATION ('/S3/s3.amazonaws.com/dataexchange-dataset/') |
| ACCESS\_ID('A\*\*\*\*\*') |
| ACCESS\_KEY('\*\*\*\*\*') |
| RETURNTYPE('NOSREAD\_KEYS') |
| ) AS THE\_TABLE |
| ORDER BY 1 |
| ; |

As above, replace the values for *LOCATION*, *ACCESS\_ID*, and *ACCESS\_KEY*.



Figure 19 READ\_NOS object length

You can substitute the NOS\_READ table operator for a foreign table definition in the above section for loading the data into a relational table.

|  |
| --- |
| CREATE MULTISET TABLE WeatherData\_temp AS ( |
| SELECT |
| CAST(payload..postal\_code AS VARCHAR(10)) Postal\_code, |
| CAST(payload..country AS CHAR(2)) Country, |
| CAST(payload..time\_valid\_utc AS TIMESTAMP(0) FORMAT 'YYYY-MM-DDBHH:MI:SS') Time\_Valid\_UTC, |
| CAST(payload..doy\_utc AS INTEGER) DOY\_UTC, |
| CAST(payload..hour\_utc AS INTEGER) Hour\_UTC, |
| CAST(payload..time\_valid\_lcl AS TIMESTAMP(0) FORMAT 'YYYY-MM-DDBHH:MI:SS') Time\_Valid\_LCL, |
| CAST(payload..dst\_offset\_minutes AS INTEGER) DST\_Offset\_Minutes, |
| CAST(payload..temperature\_air\_2m\_f AS DECIMAL (4,1)) Temperature\_Air\_2M\_F, |
| CAST(payload..temperature\_wetbulb\_2m\_f AS DECIMAL(3,1)) Temperature\_Wetbulb\_2M\_F, |
| CAST(payload..temperature\_dewpoint\_2m\_f AS DECIMAL(3,1)) Temperature\_Dewpoint\_2M\_F, |
| CAST(payload..temperature\_feelslike\_2m\_f AS DECIMAL(4,1)) Temperature\_Feelslike\_2M\_F, |
| CAST(payload..temperature\_windchill\_2m\_f AS DECIMAL(4,1)) Temperature\_Windchill\_2M\_F, |
| CAST(payload..temperature\_heatindex\_2m\_f AS DECIMAL(4,1)) Temperature\_Heatindex\_2M\_F, |
| CAST(payload..humidity\_relative\_2m\_pct AS DECIMAL(3,1)) Humidity\_Relative\_2M\_Pct, |
| CAST(payload..humidity\_specific\_2m\_gpkg AS DECIMAL(3,1)) Humdity\_Specific\_2M\_GPKG, |
| CAST(payload..pressure\_2m\_mb AS DECIMAL(5,1)) Pressure\_2M\_Mb, |
| CAST(payload..pressure\_tendency\_2m\_mb AS DECIMAL(2,1)) Pressure\_Tendency\_2M\_Mb, |
| CAST(payload..pressure\_mean\_sea\_level\_mb AS DECIMAL(5,1)) Pressure\_Mean\_Sea\_Level\_Mb, |
| CAST(payload..wind\_speed\_10m\_mph AS DECIMAL(3,1)) Wind\_Speed\_10M\_MPH, |
| CAST(payload..wind\_direction\_10m\_deg AS DECIMAL(4,1)) Wind\_Direction\_10M\_Deg, |
| CAST(payload..wind\_speed\_80m\_mph AS DECIMAL(3,1)) Wind\_Speed\_80M\_MPH, |
| CAST(payload..wind\_direction\_80m\_deg AS DECIMAL(4,1)) Wind\_Direction\_80M\_Deg, |
| CAST(payload..wind\_speed\_100m\_mph AS DECIMAL(3,1)) Wind\_Speed\_100M\_MPH, |
| CAST(payload..wind\_direction\_100m\_deg AS DECIMAL(4,1)) Wind\_Direction\_100M\_Deg, |
| CAST(payload..precipitation\_in AS DECIMAL(3,2)) Precipitation\_in, |
| CAST(payload..snowfall\_in AS DECIMAL(3,2)) Snowfall\_in, |
| CAST(payload..cloud\_cover\_pct AS INTEGER) Cloud\_Cover\_Pct, |
| CAST(payload..radiation\_solar\_total\_wpm2 AS DECIMAL(5,1)) Radiation\_Solar\_Total\_WPM2 |
| FROM     READ\_NOS ( |
| ON ( |
| SELECT CAST( NULL AS DATASET STORAGE FORMAT CSV)) |
| USING |
| LOCATION ('/S3/s3.amazonaws.com/dataexchange-dataset/') |
| ACCESS\_ID('\*\*\*\*\*\*\*\*\*\*') |
| ACCESS\_KEY('\*\*\*\*\*\*\*\*\*\*\*’) |
| ) AS THE\_TABLE |
| Where Postal\_Code = '36101' |
| ) |
| WITH DATA |
| ; |

# Configure Teradata Parallel Transporter

If you do not have access to Native Object Store (NOS) or do not want to use NOS, you can use the Teradata Parallel Transporter (TPT) utility to import data into Vantage. Using this method means that you must load the data into Vantage first before performing analysis on the data.

|  |
| --- |
| TPT works on Windows, Mac, and Linux platforms. This article demonstrates TPT on Windows but the instructions are applicable to the Mac and Linux platforms. |

## Install Teradata Tools and Utilities

If already have Teradata Tools and Utilities (TTU) installed, you can skip to the next section, [*Setup AWS Configuration Files*](#_Setup_AWS_configuration).

Download [Teradata Tools and Utilities - Windows](https://downloads.teradata.com/download/tools/teradata-tools-and-utilities-windows-installation-package) to your Windows system. While this will download the current version of TTU, it will work with previous versions of Vantage.

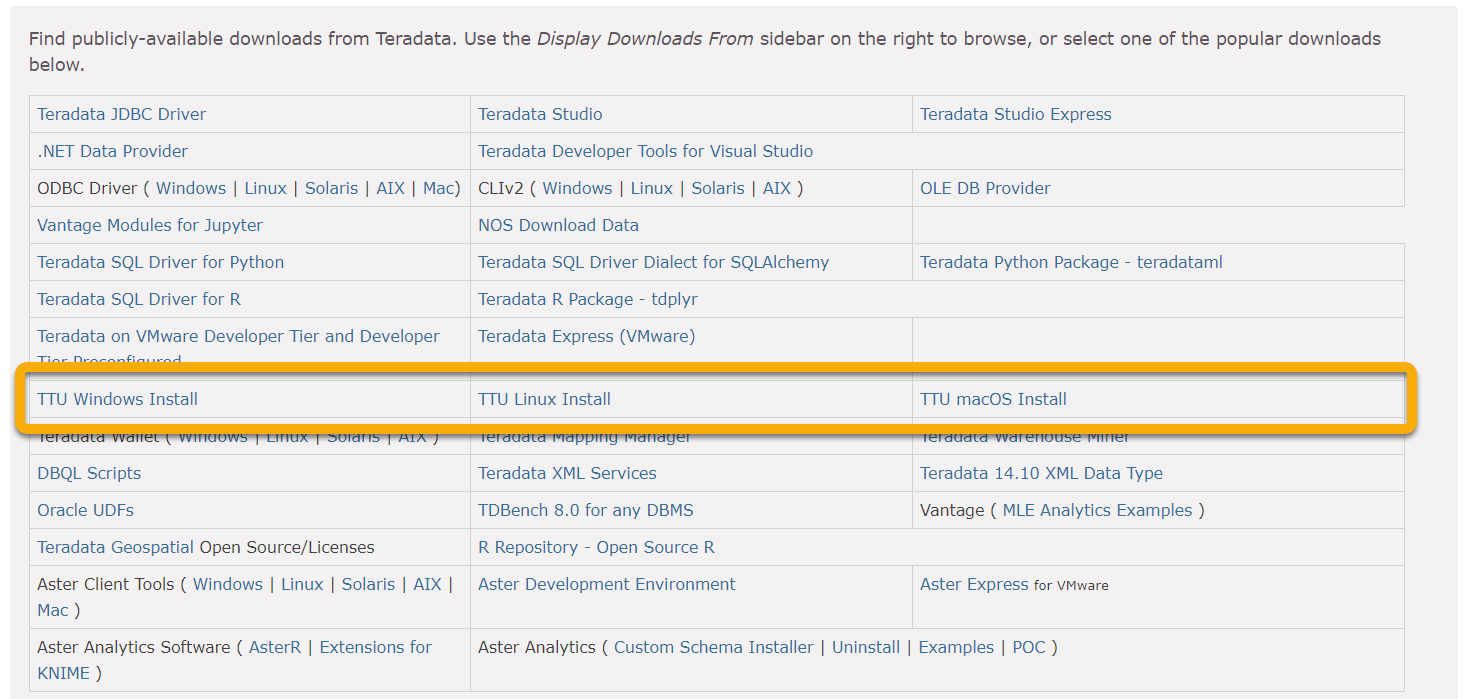


Figure 20 Download Teradata Tools and Utilities (TTU)

Unzip the download file.

Run **Setup.bat**.

At the minimum, install the following packages:

* Teradata Parallel Transporter Base
* Teradata Access Module for Amazon S3

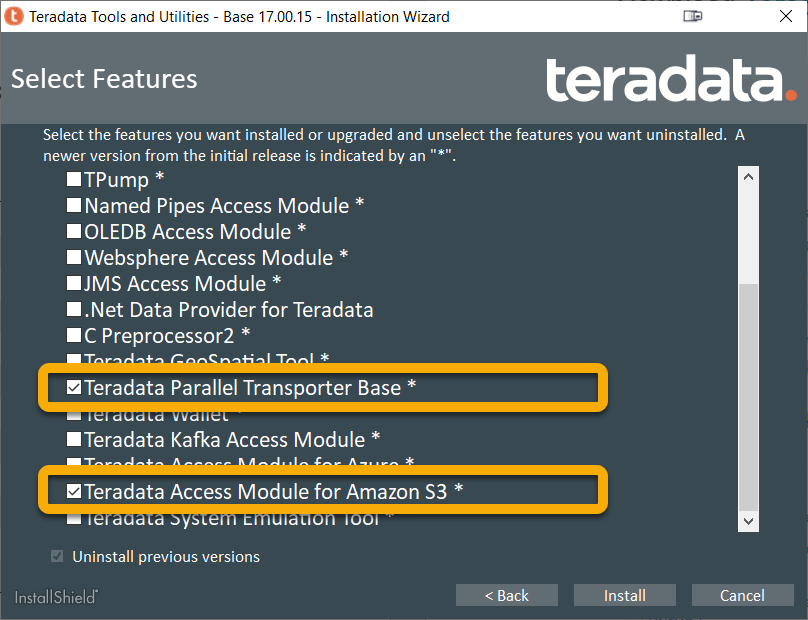


Figure 21 Select TPT Base and Access Module for Amazon S3

|  |
| --- |
| The installer will add any necessary dependent packages. You may find it useful to add the following features:   * + - ODBC Driver for Teradata     - Teradata GSS Administration Package     - BTEQ     - .Net Data Provider for Teradata     - Teradata Access Module for Azure |
| The installer will automatically *uninstall* previous versions of any features that you have selected. If you have any third-party dependencies that require a specific version that you already have installed, you should de-select the *Uninstall previous versions* checkbox. |

## Setup AWS configuration files

The Access Module for Amazon S3 uses the AWS configuration files for credentials and the region.

If you have previously installed the [AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html), you can use the **aws configure** command at the command prompt to both view and edit your default access credentials and region.

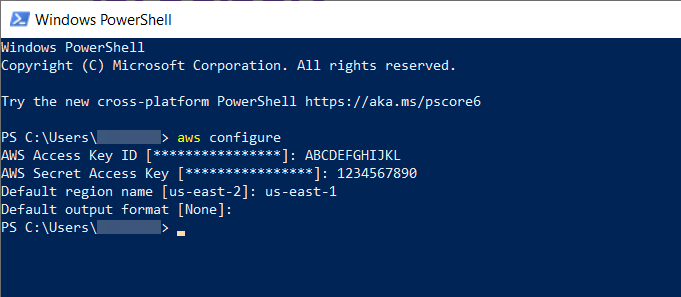


Figure 22 Configure AWS CLI

By default, the two configuration files are located in your *%USERPROFILE%\.aws* directory. (Note that there is a period in front of “aws”.)

You can also create the files manually, which is necessary if you do not have the AWS CLI installed.

The *config* file holds the default region for actions on AWS.

|  |
| --- |
| [default] |
| Region = us-east-1 |

The *credentials* file holds the default access keys for actions on AWS.

|  |
| --- |
| [default] |
| aws\_access\_key\_id = ABCDEFGHIJKL |
| aws\_secret\_access\_key = 1234567890 |

The region and [access keys specified](https://docs.amazonaws.cn/en_us/IAM/latest/UserGuide/id_credentials_access-keys.html) in these two files must be able to access the S3 bucket you created.

The Access Module for Amazon S3 needs the location of the AWS configuration files. It uses the *HOME* environment variable for this value. You will need to create it manually.

Open the *System* control panel applet by going into *Control Panel* > *System and Security* > *System*.

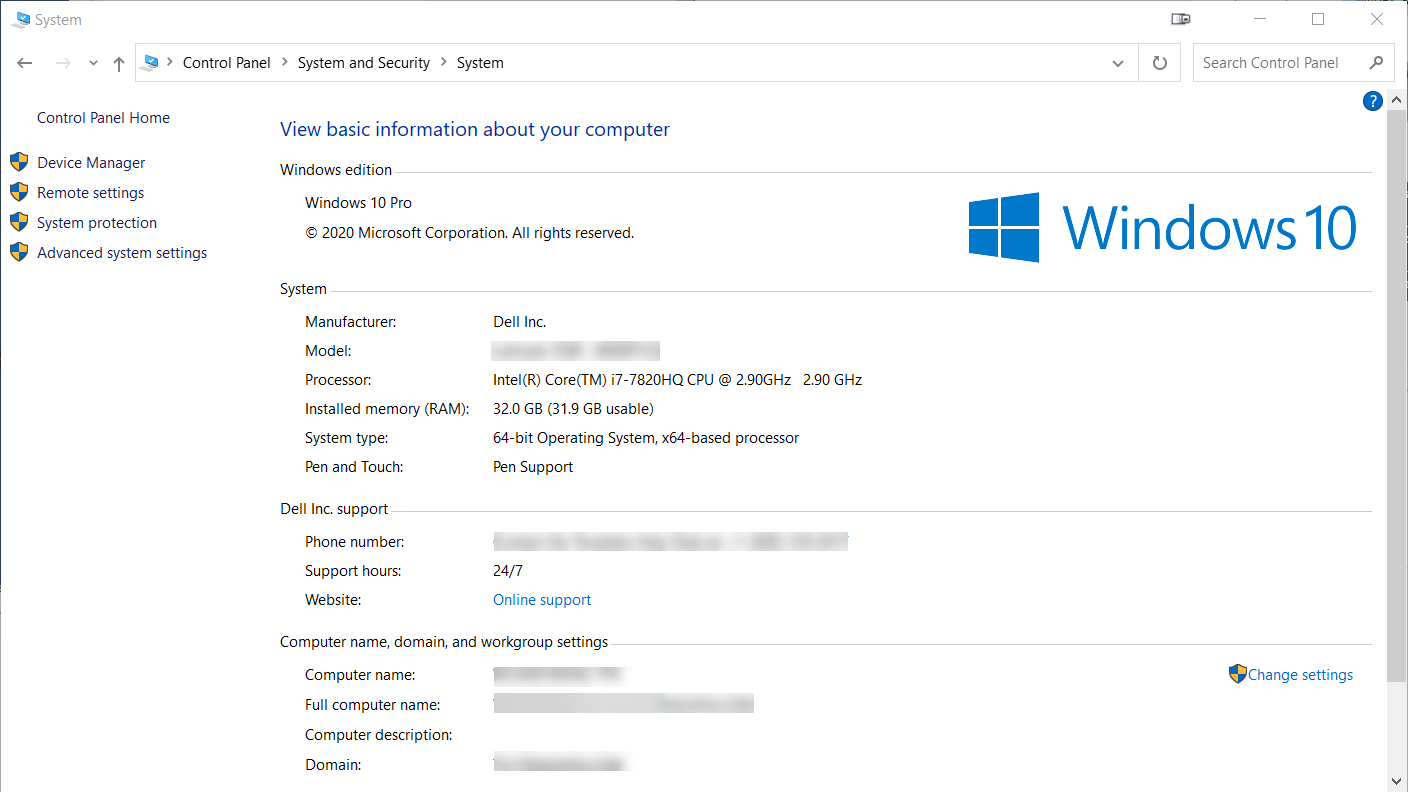


Figure 23 System control panel

Click on **Advanced system settings**.

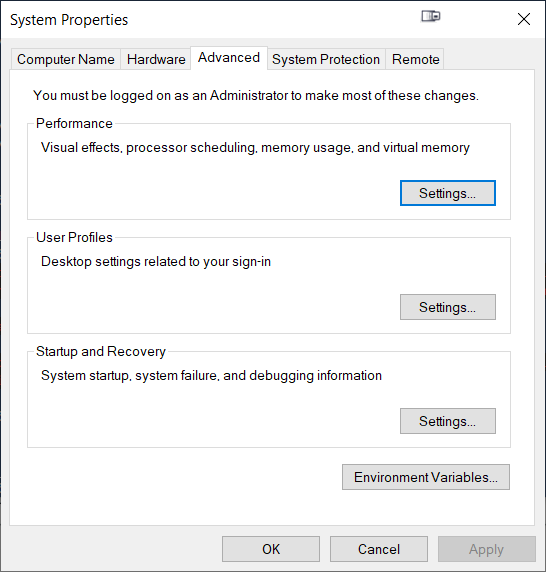


Figure 24 System properties

Click on **Environment Variables…**.

Click on **New…** in the *User variables* section.

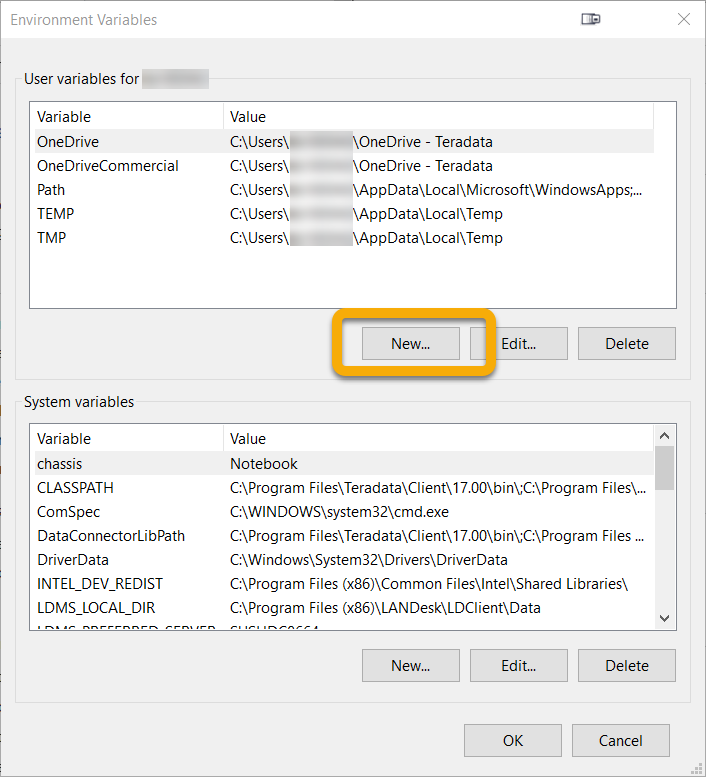


Figure 25 User environment variables

Enter *HOME* for the *Variable name*.

Enter the directory where your AWS configurations files are located for the *Variable value*. You may use the configuration files created by the AWS CLI, or you may create a separate directory that holds configuration files that you created (or copied from the AWS CLI).

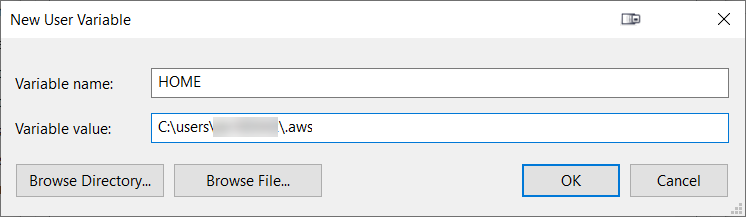


Figure 26 User HOME environment variable

# Load data from Amazon S3 into Vantage using TPT

TPT uses the following script files to load data into Vantage:

* Job variables
* Create table
* Load data

## Create the job variables file

The job variables file defines the values for variables used by TPT when running jobs. Using a job variables file allows the variables to be defined once and then used in multiple jobs.

Create the following *jobVars* script file on your Windows system.

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| /\* TPT attributes - Common for all Samples \*/ |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| TargetUserName = '<*TargetUserName*>' |
| ,TargetUserPassword = '<*TargetUserPassword*>' |
| ,TargetTdpId = '*<VantageSystemName>*' |
| ,TargetErrorList = [ '3706','3803','3807' ] |
| ,DDLPrivateLogName = 'DDL\_OPERATOR\_LOG' |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| /\* TPT LOAD Operator attributes \*/ |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| ,LoadPrivateLogName = 'LOAD\_OPERATOR\_LOG' |
| ,LoadTargetTable = 'onpoint' |
| ,LoadLogTable = 'onpoint\_log' |
| ,LoadErrorTable1 = 'onpoint\_e1' |
| ,LoadErrorTable2 = 'onpoint\_e2' |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| /\* TPT DataConnector Producer Operator \*/ |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| ,FileReaderFormat = 'Delimited' |
| ,FileReaderTextDelimiter = ',' |
| ,FileReaderPrivateLogName = 'onpointP2\_1' |
| ,FileReaderFileName = 'onpointDT' |
| ,FileReaderDirectoryPath = '.' |
| ,FileReaderOpenMode = 'Read' |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| /\* APPLY STATEMENT parameters \*/ |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| ,LoadInstances = 1 |
| ,FileReaderInstances = 1 |

Replace *<TargetUserName>*, *<TargetUserPassword>*, and *<VantageSystemName>* with appropriate values. (Remember to remove the brackets, too.)

## Create the CREATE TABLE script file

The CREATE TABLE script file cleans up the temporary load tables (if any) and creates the target table in Vantage. You only need to run this script file if the target table needs to be created.

Create the following *CreateTable* script file on your Windows system.

|  |
| --- |
| DEFINE JOB CREATE\_TABLE |
| DESCRIPTION 'Clean up and create target table' |
| ( |
| STEP CLEANUP\_CREATE\_TABLE\_STEP |
| ( |
| APPLY |
| (' DROP TABLE WeatherData '), |
| (' DROP TABLE WeatherData\_e1 '), |
| (' DROP TABLE WeatherData\_e2 '), |
| (' DROP TABLE WeatherData\_log '), |
| ('CREATE MULTISET TABLE WeatherData |
| ( |
| Postal\_code VARCHAR(5), |
| Country VARCHAR(3), |
| Time\_Valid\_UTC VARCHAR(20), |
| DOY\_UTC VARCHAR(2), |
| Hour\_UTC VARCHAR(3), |
| Time\_Valid\_LCL VARCHAR(20), |
| DST\_Offset\_Minutes VARCHAR(5), |
| Temperature\_Air\_2M\_F VARCHAR(5), |
| Temperature\_Wetbulb\_2M\_F VARCHAR(5), |
| Temperature\_Dewpoint\_2M\_F VARCHAR(5), |
| Temperature\_Feelslike\_2M\_F VARCHAR(5), |
| Temperature\_Windchill\_2M\_F VARCHAR(5), |
| Temperature\_Heatindex\_2M\_F VARCHAR(5), |
| Humidity\_Relative\_2M\_Pct VARCHAR(5), |
| Humdity\_Specific\_2M\_GPKG VARCHAR(5), |
| Pressure\_2M\_Mb VARCHAR(8), |
| Pressure\_Tendency\_2M\_Mb VARCHAR(5), |
| Pressure\_Mean\_Sea\_Level\_Mb VARCHAR(8), |
| Wind\_Speed\_10M\_MPH VARCHAR(5), |
| Wind\_Direction\_10M\_Deg VARCHAR(8), |
| Wind\_Speed\_80M\_MPH VARCHAR(5), |
| Wind\_Direction\_80M\_Deg VARCHAR(8), |
| Wind\_Speed\_100M\_MPH VARCHAR(5), |
| Wind\_Direction\_100M\_Deg VARCHAR(8), |
| Precipitation\_in VARCHAR(5), |
| Snowfall\_in VARCHAR(5), |
| Cloud\_Cover\_Pct VARCHAR(5), |
| Radiation\_Solar\_Total\_WPM2 VARCHAR(5) |
| ); |
| ') |
| TO OPERATOR ($DDL); |
| ); |
| ); |

## Create the load job script file

The load job script file performs that actual loading of data from the Amazon S3 bucket into Vantage.

Create the following *LoadDataFromS3* script file on your Windows system.

|  |
| --- |
| DEFINE JOB LOAD\_DATA\_FROM\_S3\_TO\_VANTAGE |
| DESCRIPTION 'Load data from Amazon S3 bucket into Vantage' |
| ( |
| SET TargetTable = 'WeatherData'; |
| STEP IMPORT\_THE\_DATA |
| ( |
| APPLY $INSERT TO OPERATOR ($LOAD) |
| SELECT \* FROM OPERATOR ($FILE\_READER () |
| ATTR |
| ( |
| SkipRows = 1, |
| AccessModuleName = 'libs3axsmod.dll', |
| AccessModuleInitStr = 'S3Bucket=<*S3BucketName*> S3Prefix="/" S3Object=onpoint\_history\_postal-code\_hour\_201801010000-201812312359.csv S3SinglePartFile=True' |
| ) |
| ); |
| ); |
| ); |

The *SkipRows* parameter determines how many rows will be ignored at the start of the file. Typically, this is one (1) because the first row has the column headers, not data.

Replace the *S3BucketName* parameter with the name of your bucket. (We are using *dataexchange‑dataset* in this article. Your bucket name will be different.) Remember to remove the brackets.

If the dataset object is at the “root” of your bucket, you may leave the *S3Prefix* parameter as *“/”*. If you have created “directories”, then you will need to replace the “directory” path, ensuring that it ends as a forward slash. (Technically, Amazon S3 does not support directories, but it acts as if it does.)

The *S3Object* parameter defines the name of the object to be loaded. *onpoint\_history\_postal‑code\_hour\_201801010000‑201812312359.csv* is the name of the object used in this article.

## Run the scripts

Ensure that all of the script files are in the same directory.

In your command prompt, change to the directory with the script files.

Run the following command to create the target table.

|  |
| --- |
| tbuild -f CreateTable -v jobVars -j createTable |

Run the following command to load the target table.

|  |
| --- |
| tbuild -f LoadDataFromS3 -v jobVars -j loadData |

# Query the Data Set in Vantage

Login to your Vantage system with Teradata Studio.

Run the following SQL command to validate the contents of the table.

|  |
| --- |
| SELECT \* FROM WeatherData SAMPLE 10; |

A picture containing large

Description automatically generated

17095 Via Del Campo, San Diego, CA 92127      [Teradata.com](http://www.teradata.com)

Teradata and the Teradata logo are registered trademarks of Teradata Corporation and/or its affiliates in the U.S. and worldwide. Teradata continually improves products as new technologies and components become available. Teradata, therefore, reserves the right to change specifications without prior notice. All features, functions, and operations described herein may not be marketed in all parts of the world. Consult your Teradata representative or Teradata.com for more information.

Copyright © 2021 by Teradata Corporation    All Rights Reserved.    Produced in U.S.A.

07.21 EBxxxx