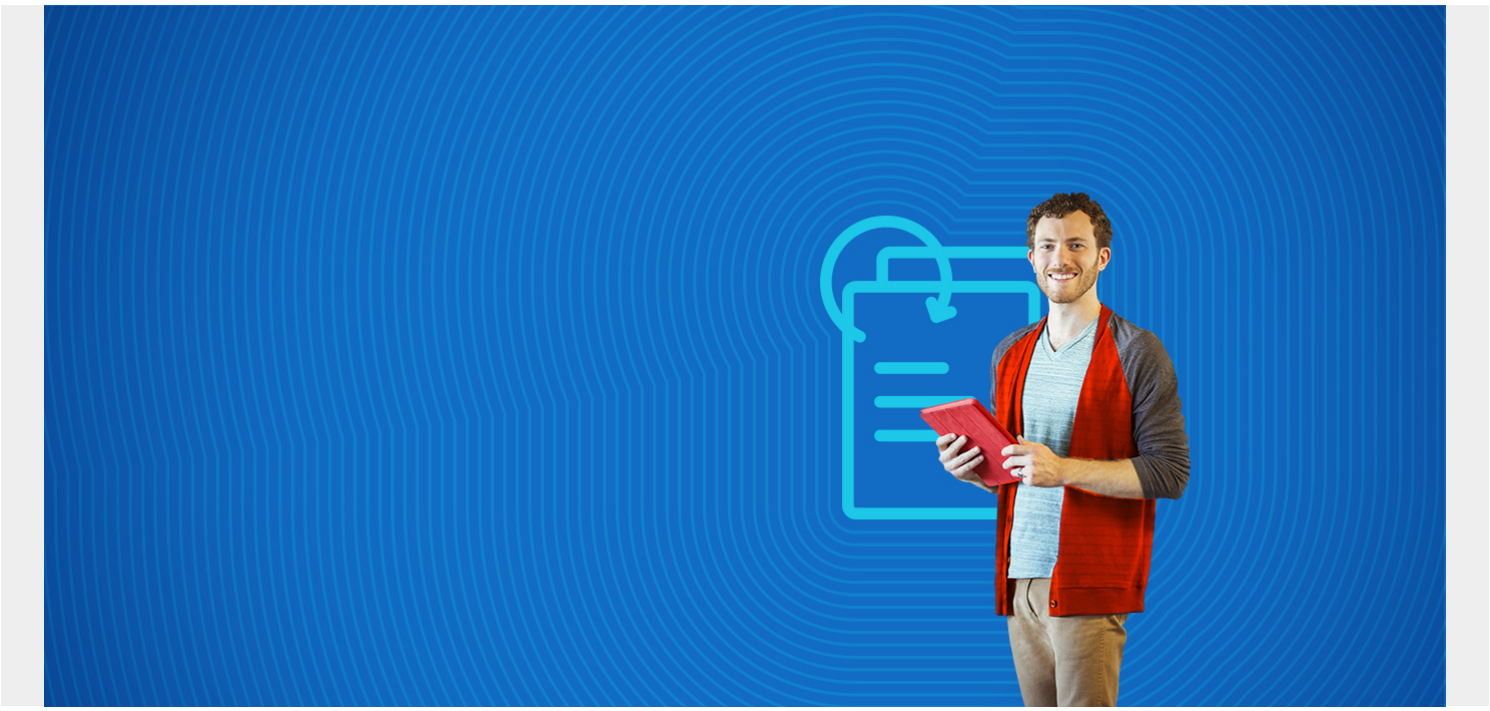
LOADING CSV FILES FROM S3 TO SNOWFLAKE



In this tutorials, we show how to load a CSV file from Amazon S3 to a Snowflake table.

We've also covered how to load JSON files to Snowflake.

(This article is part of our Snowflake Guide. Use the right-hand menu to navigate.)

Sample data

You need some data to work through this example. Download 1,000 weather records from here. (We purchased 20 years of weather data for Paphos, Cyprus, from OpenWeather. This is just a small subset of 1,000 records converted to CSV format.)

Upload this data to Amazon S3 like this:

```
aws s3 cp paphosWeather.csv s3://gluebmcwalkerrowe/paphosWeather.csv
```

Create table in Snowflake

Unfortunately, Snowflake does not read the header record and create the table for you. (Presumably that is because they would prefer that you define the column data types, number precision, and size.)

```
create table weather(
dt integer,
temp decimal(6,2),
temp_min decimal(6,2),
```

```
temp_max decimal(6,2),
pressure int,
humidity int,
speed decimal(6,2),
deg int,
main varchar(50),
description varchar(50))
```

Copy data file to Snowflake stage area

Then create a Snowflake stage area like this.

It does not matter what your snowflake credentials are. Put your Amazon AWS credentials. In fact, there is no place to look up your Snowflake credentials since they are the same as your Amazon IAM credentials—even if you are running it on a Snowflake instance, i.e., you did not install Snowflake locally on your EC2 servers.

```
create or replace stage paphosweather
url='s3://gluebmcwalkerrowe/paphosWeather.csv'
credentials=(aws_key_id='xxxxxxx' aws_secret_key='xxxxxxx')
```

Now copy the data into the table you created above.

```
copy into weather
  from 's3://gluebmcwalkerrowe/paphosWeather.csv'
  credentials=(aws_key_id='xxxxxxx' aws_secret_key='xxxxxxxx')
  file_format = (type = csv field_delimiter = ',' skip_header = 1);
```

Convert the epoch time to readable format

The **dt** column is epoch time, which is the number of seconds since January 1, 1970. You can convert it to readable format (e.g., 2000-01-01 01:00:00.000) like this.

```
select to_timestamp(dt) from weather
```

Snowflake date and time formats

Snowflake seems to have some limits on importing date and time values. The data I had included the epoch time () as well as the time in this format:

2020-09-12 23:00:00 +0000 UTC

which I converted to this format, were the +00.00 means the time zone offset in hh:mm:

2020-09-12 23:00:00+00:00

This format is also available, which drops the time zone altogether:

2020-09-12 23:00:00

Because I could not get any of those values to load, I used the epoch time. (If you can make it work write to us at blogs@bmc.com.)

2020-09-12 23:00:00+00:00 should match this formatting statement, which you set using the **alter session timestamp** statement prior to loading the csv file:

```
alter session set TIMESTAMP_INPUT_FORMAT = 'yyyy-mm-dd HH24:MI:SS+TZH:TZM'
```

But Snowflake threw an error saying they did not match. That makes no sense since this statement, which tests that, worked with no problem:

```
select to_timestamp('2000-01-01 00:00:00+00:00', 'yyyy-mm-dd
HH24:MI:SS+TZH:TZM');
```

Looking further, I saw in the documentation that the Snowflake datetime format does not support milliseconds or time zone offset. But I still got an error when I dropped that part of the date. The timestamp format did not work either. (Again, write to us if you do get this to work at blogs@bmc.com.)

Additional resources

For more tutorials like this, explore these resources:

- BMC Machine Learning & Big Data Blog
- Snowflake SQL Aggregate Functions & Table Joins
- Snowflake Window Functions: Partition By and Order By
- Snowflake Lag Function and Moving Averages
- AWS Guide, with 15 articles and tutorials
- Amazon Braket Quantum Computing: How To Get Started