



HANDS-ON LAB GUIDE FOR MACHINE LEARNING WITH SNOWFLAKE AND AMAZON SAGEMAKER

To be used with the Snowflake free 30-day trial at:
<https://trial.snowflake.com>

Snowflake Enterprise Edition preferred on **AWS** - US West, US East (Ohio or N. Virginia) regions recommended

AWS Account - Select region - US-WEST-2 (Oregon), US-EAST-2 (Ohio) and US-EAST-1(N. Virginia) are good choices

[Create AWS Account](#)

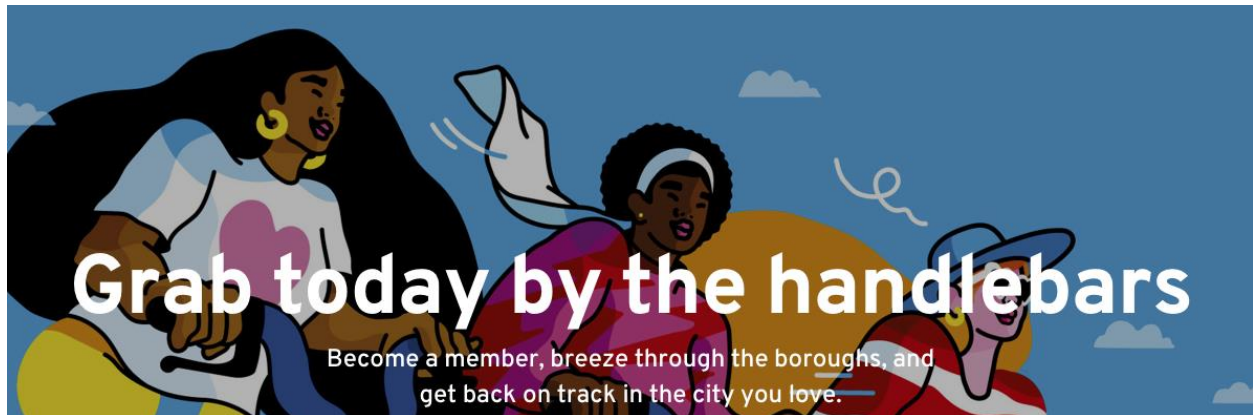
Approximate duration: 90 minutes.

Table of Contents

1	<i>Lab Overview</i>	<i>3</i>
1.1	Target Audience.....	3
1.2	What you'll learn.....	3
1.3	Pre-requisites	3
2	<i>Prepare Your Lab Environment.....</i>	<i>5</i>
2.1	Steps to Prepare Your Lab Environment.....	5
2.1.1	Register for a Snowflake free 30-day trial.....	5
2.1.2	Create AWS Account.....	5
2.1.3	Download the Snowflake and Notebook files to your local machine.....	6
2.2	Prepare your SageMaker Notebook.....	7
2.2.1	Create a Notebook Instance.....	7
2.2.2	Upload Jupyter files	8
2.2.3	Run the first cell to load required libraries.....	8
3	<i>The Snowflake User Interface.....</i>	<i>10</i>
3.1	Logging Into the Snowflake User Interface (UI).....	10
3.2	Close any Welcome Boxes and Tutorials.....	10
3.3	Navigating the Snowflake UI	11
4	<i>Preparing to Load Data & Loading Data</i>	<i>16</i>
4.1	Start using Worksheets and Create a Virtual Warehouse.....	16
4.1.1	Prepare the worksheet.....	16
4.1.2	Create a warehouse for Sagemaker	18
4.1.3	Create a Role and User	19
4.1.4	Create Citibike_ML Database.....	20
4.1.5	Create the Trips table	21
4.1.6	Create an External Stage.....	21
4.1.7	Create a File Format	22
4.1.8	Load the data into the Trips table.....	23
4.1.9	Verify data loading.....	23
4.1.10	Create a table for predictions.....	24
5	<i>Using Snowflake Market Place.....</i>	<i>25</i>
6	<i>Machine Learning in SageMaker Notebook.....</i>	<i>31</i>
7	<i>Summary & Next Steps.....</i>	<i>44</i>
8	<i>Resetting Your Snowflake Environment</i>	<i>45</i>
9	<i>Resetting Your AWS & SageMaker Environment.....</i>	<i>46</i>

1 Lab Overview

In this lab, you'll learn the basics of creating an advanced analytics solution. We'll train a time series model from data in Snowflake using a SageMaker Notebook instance. This model will predict the number of bike trips for a day. First, we will load data from [Citibike](#) trips into Snowflake to provide basic analysis and prediction. We will see how to enhance the model adding new features for weather conditions. Using Snowflake Data Market place, an enhanced data set will be used with immediate access to data. Snowflake will provide a single source of truth for Data Scientists using the Snowflake Data Marketplace capabilities.



1.1 Target Audience

Database and Data Warehouse Administrators and Architects. Developers looking to extend Data Apps with Machine Learning. ML Practitioners looking to incorporate Snowflake data in their ML workflow.

1.2 What you'll learn

The exercises in this lab will walk you through the steps to:

- Create stages, databases, tables, user, role and warehouses
- Load data into a table within your Snowflake account
- Launch a SageMaker Notebook instance
- Connect to your Snowflake instance and pull data into a Pandas dataframe
- Use Snowflake as a data repository for ML models

1.3 Pre-requisites

Make sure you comply with the following pre-prequisites

- Use of the Snowflake free 30-day trial environment
- Use of an AWS Account

- Basic knowledge of SQL, and database concepts and objects
- Familiarity with CSV comma-delimited files
- Basic Jupyter notebook and Python knowledge

2 Prepare Your Lab Environment

2.1 Steps to Prepare Your Lab Environment

2.1.1 Register for a Snowflake free 30-day trial

If not yet done, register for a Snowflake free 30-day trial at <https://trial.snowflake.com>

- The Snowflake Enterprise Edition on AWS and US West, US East (Ohio or N. Virginia) regions recommended. Or we suggest you select the region which is physically closest to you.
- After registering, you will receive an email with an activation link and your Snowflake account URL. Bookmark this URL for easy, future access. After activation, you will create a username and password. Write down these credentials.

2.1.2 Create AWS Account

If you do not already have a AWS account please create a new account using this link - [Create AWS Account](#). Make sure you have the permissions to launch a SageMaker Notebook instance. Once logged in to your account select an AWS region closest to your Snowflake account, US-WEST-2 (Oregon), US-EAST-2 (Ohio) and US-EAST-1(N. Virginia) are good choices.

You can try Amazon SageMaker for free. Here are the notes from the Amazon [SageMaker Pricing](#):

“As part of the [AWS Free Tier](#), you can get started with Amazon SageMaker for free. If you have never used Amazon SageMaker before, for the first two months, you are offered a monthly free tier of 250 hours of t2.medium or t3.medium notebook usage with on-demand notebook instances or t3.medium instances with SageMaker Studio notebooks for building your models, plus 50 hours of m4.xlarge or m5.xlarge for training your models, plus 125 hours of m4.xlarge or m5.xlarge for deploying your machine learning models for real-time inference and batch transform with Amazon SageMaker. The free tier does not cover the storage volume usage. Your free tier starts from the first month when you create your first SageMaker resource.”

Resize your browser windows so you can view this lab guide PDF and your web browser side-by-side to more easily follow the lab instructions. If possible, even better is to use a secondary display dedicated to the lab guide. It is also advisable to open a second browser window so you are able to view the Snowflake UI and AWS console side by side

2.1.3 Download the Snowflake and Notebook files to your local machine.

Download the following files:

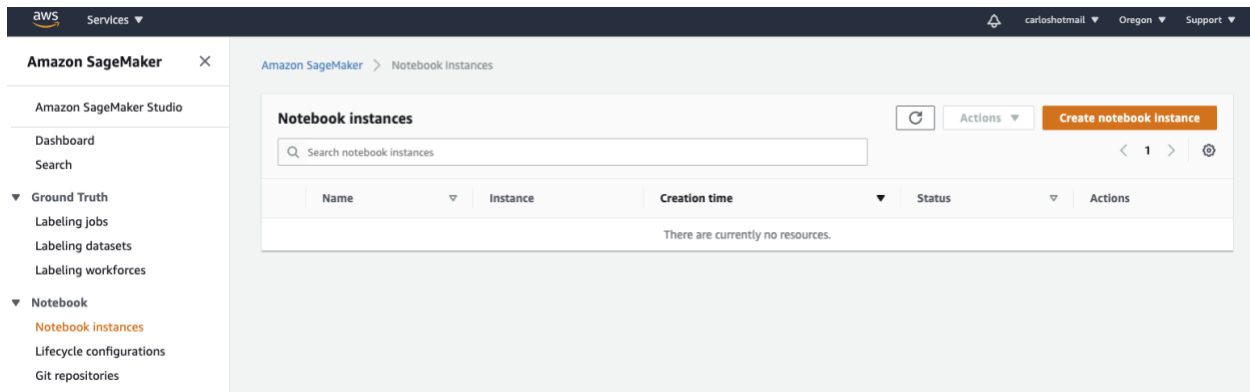
- Jupyter Notebook [citibike-trips-prediction-sagemaker.iypnb](#) that will be the Notebook we will be running to create our models:
- JSON file [creds.json](#) where credentials will be stored
- SQL file [load_citibike.sql](#) with all the sequence to create the role, user, database and load data
- SQL file [query_citibike.sql](#) with some queries
- SQL file [cleanup_citibike.sql](#) that will remove all the previous content from your account

2.2 Prepare your SageMaker Notebook

This first step will take longer time as we have to load some libraries in the Notebook. We recommend to start with this step first.

2.2.1 Create a Notebook Instance

Login into your AWS Account and select an AWS region closest to your Snowflake account. Select Notebook instances and click on Create notebook instance.



Type a name for your instance and select or create a new IAM role. Make sure you use a **ml.t3.medium** instance

Amazon SageMaker > Notebook Instances > Create notebook instance

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

Notebook instance settings

Notebook instance name

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

Elastic Inference [Learn more](#)

► Additional configuration

Permissions and encryption

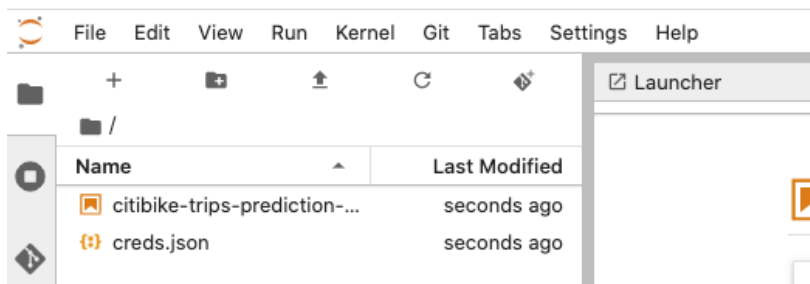
IAM role
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

Root access - optional
☒ Enable - Give users root access to the notebook
☐ Disable - Don't give users root access to the notebook
Lifecycle configurations always have root access

Encryption key - optional
Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

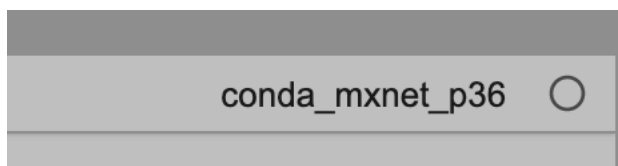
2.2.2 Upload Jupyter files

Once the instance has been created, open it by clicking in Open Jupyter Notebook. Now upload the Jupyter notebook and the json file with the credentials previously downloaded.



2.2.3 Run the first cell to load required libraries.

Open the Jupyter Notebook and make sure kernelconda_mxnet_p36 is selected. There are no specific reasons for that, but this lab has been tested with it.



Run the first cell that will take a while to complete.

To install the dependencies, select the cell below, change its type to 'code' from the menu above and then run it.

```
[ ]: !pip install --upgrade pip && pip install --upgrade numpy
!pip install --upgrade snowflake-connector-python[pandas]
!conda install -c conda-forge matplotlib --yes
!conda install -c plotly plotly==3.10.0 --yes
!conda install -c conda-forge fbprophet --yes
!pip install --upgrade pystan
!pip install --upgrade fbprophet
```

IMPORTANT: Make sure to restart the kernel before you move forward

This cell will take around 30 minutes to run. This is only needed the first time. Meanwhile we are going to get all the data we need into Snowflake.

3 The Snowflake User Interface



About the screen captures, sample code, and environment

Screen captures in this lab depict examples and results that may slightly vary from what you may see when you complete the exercises.

3.1 Logging Into the Snowflake User Interface (UI)

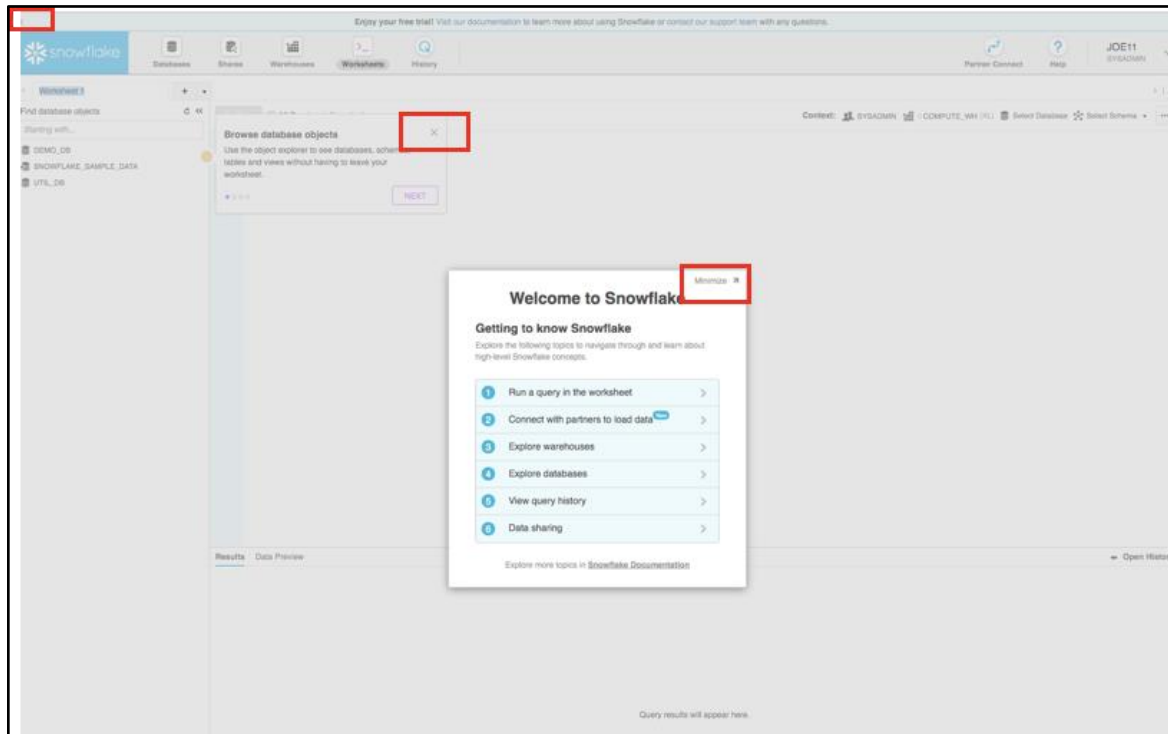
Open a browser window and enter the URL of your Snowflake 30-day trial environment.

You should see the login screen below. Enter your unique credentials to log in.

The screenshot shows the Snowflake login page. On the left, the Snowflake logo (a stylized snowflake) and the word 'snowflake' are displayed in white on a blue background. On the right, there is a white rectangular box with a light gray border. Inside this box, the text 'Log in to Snowflake' is centered at the top. Below this, there are two input fields: 'User Name' and 'Password'. To the right of the 'Password' field is a blue link that says '[Forgot password]'. At the bottom of the box is a gray button with the text 'Log In'.

3.2 Close any Welcome Boxes and Tutorials

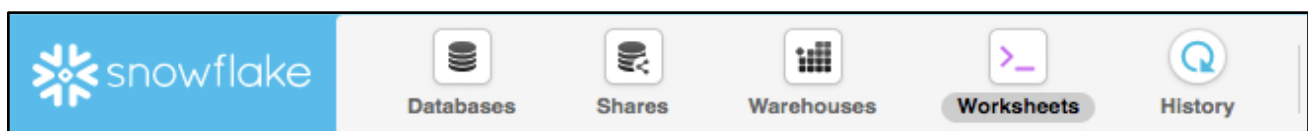
You may see “welcome” and “helper” boxes in the UI when you log in for the first time. Also a “Enjoy your free trial...” ribbon at the top of the UI. Minimize and close them by clicking on the items in the red boxes on screenshot below.



3.3 Navigating the Snowflake UI

First let's get you acquainted with Snowflake! This section covers the basic components of the user interface to help you orient yourself. We will move left to right in the top of the UI.

The top menu allows you to switch between the different areas of Snowflake:



The **Databases** tab shows information about the databases you have created or have privileges to access. You can create, clone, drop, or transfer ownership of databases as well as load data (limited) in the UI. Notice several databases already exist in your environment. However, we will not be using these in this lab.

Database	Origin	Creation Time ▼	Owner	Comment
SNOWFLAKE_SAMPLE_DATA	SFC_SAMPLES.SA...	6/27/19 11:52:44 PM	ACCOUNTADMIN	TPC-H, OpenWeatherMap, etc
DEMO_DB		6/27/19 11:52:42 PM	SYSADMIN	demo database
UTIL_DB		6/27/19 11:52:30 PM	SYSADMIN	utility database

The **Shares** tab is where data sharing can be configured to easily and securely share Snowflake table(s) among separate Snowflake accounts or external users, without having to create a second copy of the table data. We are going to use the Market Place to get weather data and enhance the prediction of our model.

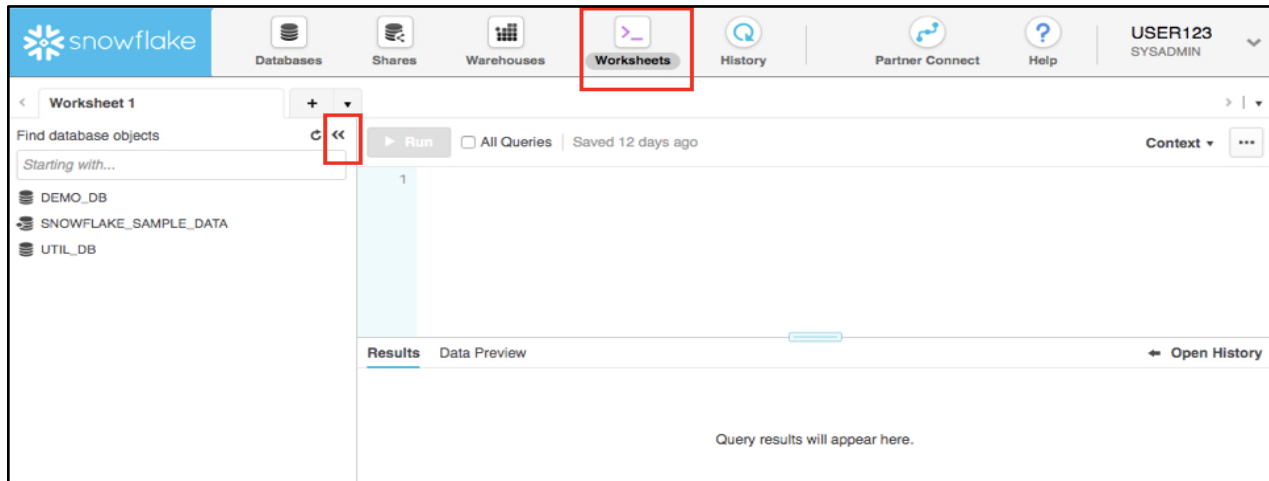
The **Warehouses** tab is where you set up and manage compute resources (virtual warehouses) to load or query data in Snowflake. Note a warehouse called “COMPUTE_WH (XL)” already exists in your environment.

Status	Warehouse Name	Size	Run...	Que...	Auto Suspend	Auto Resume	Created On ▼	Resumed On	Owner
Suspended	COMPUTE_WH	X-Large	0	0	10 minutes	Yes	6/28/19 12:00:24 AM	6/28/19 12:00:24 AM	SYSADMIN

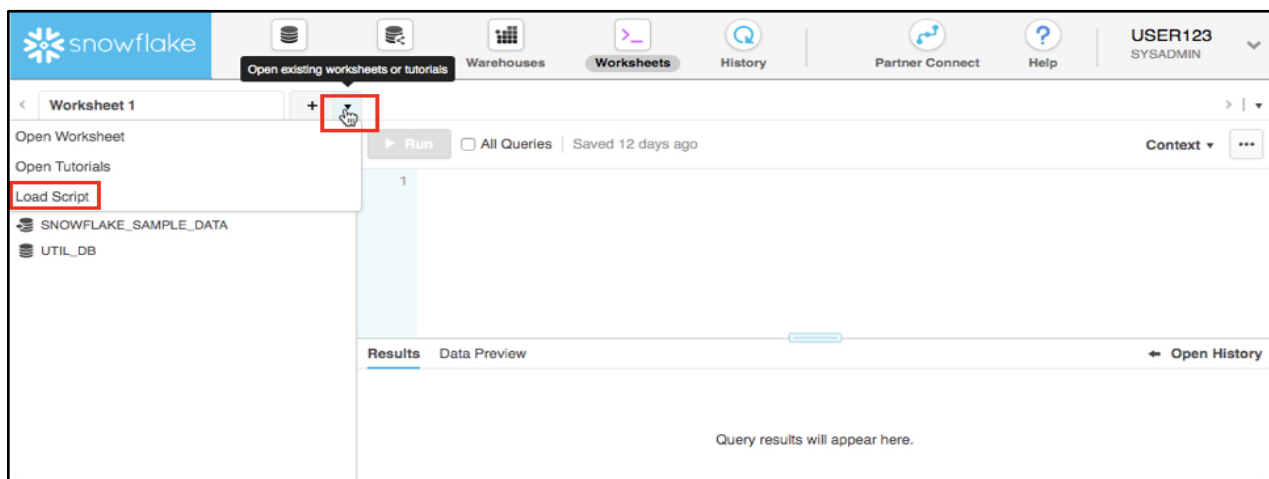
The **Worksheets** tab provides an interface for submitting SQL queries, performing DDL and DML operations and viewing results as your queries/operations complete. The default “Worksheet 1” appears.

In the left pane is the database objects browser which enables users to explore all databases, schemas, tables, and views accessible by the role selected for a worksheet. The bottom pane shows results of queries and operations.

The various windows on this page can be resized by moving the small sliders on them. And if during the lab you need more room to work in the worksheet, collapse the database objects browser in the left pane. Many of the screenshots in this guide will have this database objects browser closed.



At the top left of the default “Worksheet 1,” just to the right of the worksheet tab, click on the small, downward facing arrow, select “Load Script”, then browse to the [load_citibike.sql](#) file you downloaded in the prior module and select “Open”. All of the SQL commands you need to run for the remainder of this lab will now appear on the new worksheet. Do not run any of the SQL commands yet. We will come back to them later in the lab and execute them one at a time.



Warning - Do Not Copy/Paste SQL From This PDF to a Worksheet



Copy-pasting the SQL code from this PDF into a Snowflake worksheet will result in formatting errors and the SQL will not run correctly. Make sure to use the “Load Script” method just covered.

On older or locked-down browsers, this “load script” step may not work as the browser will prevent you from opening the .sql file. If this is the

case, open the .sql file with a text editor and then copy/paste all the text from the .sql file to the “Worksheet 1”

Worksheets vs the UI



Most of the configurations in this lab will be executed via this pre-written SQL in the Worksheet in order to save time. These configurations could also be done via the UI in a less technical manner but would take more time.

The **History** tab allows you to view the details of all queries executed in the last 14 days in the Snowflake account (click on a Query ID to drill into the query for more detail).

Status	Query ID	SQL Text	User	Warehouse	Size	Session ID	Start Time	End Time	Total Duration
✓	018d6a78-...	SHOW GRANTS TO ...	USER123	COMPUTE_...		225259525	1:08:12 PM	1:08:12 PM	134ms

If you click on the top right of the UI where your username appears, you will see that here you can do things like change your password, roles, or preferences.

Snowflake has several system defined roles. You are currently in the default role of SYSADMIN.

Status	Query ID	SQL Text	User	Warehouse	Size	Session ID	Start Time	End Time	Total Duration
✓	018d6a78-...	SHOW GRANTS TO ...	USER123	COMPUTE_...		225259525	1:08:12 PM	1:08:12 PM	134ms

SYSADMIN



For most this lab you will be in the SYSADMIN (aka System Administrator) role which has privileges to create warehouses and databases and other objects in an account.

In a real-world environment, you would use different roles for the tasks in this lab, and assign the roles to your users.

<https://docs.snowflake.net/manuals/user-guide/security-access-control.html>

4 Preparing to Load Data & Loading Data

Let's start by preparing to load the structured data on Citibike into Snowflake.

This module will walk you through the steps to:

- Create a virtual warehouse
- Create a role and user
- Granting of a role to a user and privileges to a role
- Create a database and tables
- Create external stages
- Create a file format for the data
- Load data into a table and querying the table

Getting Data into Snowflake



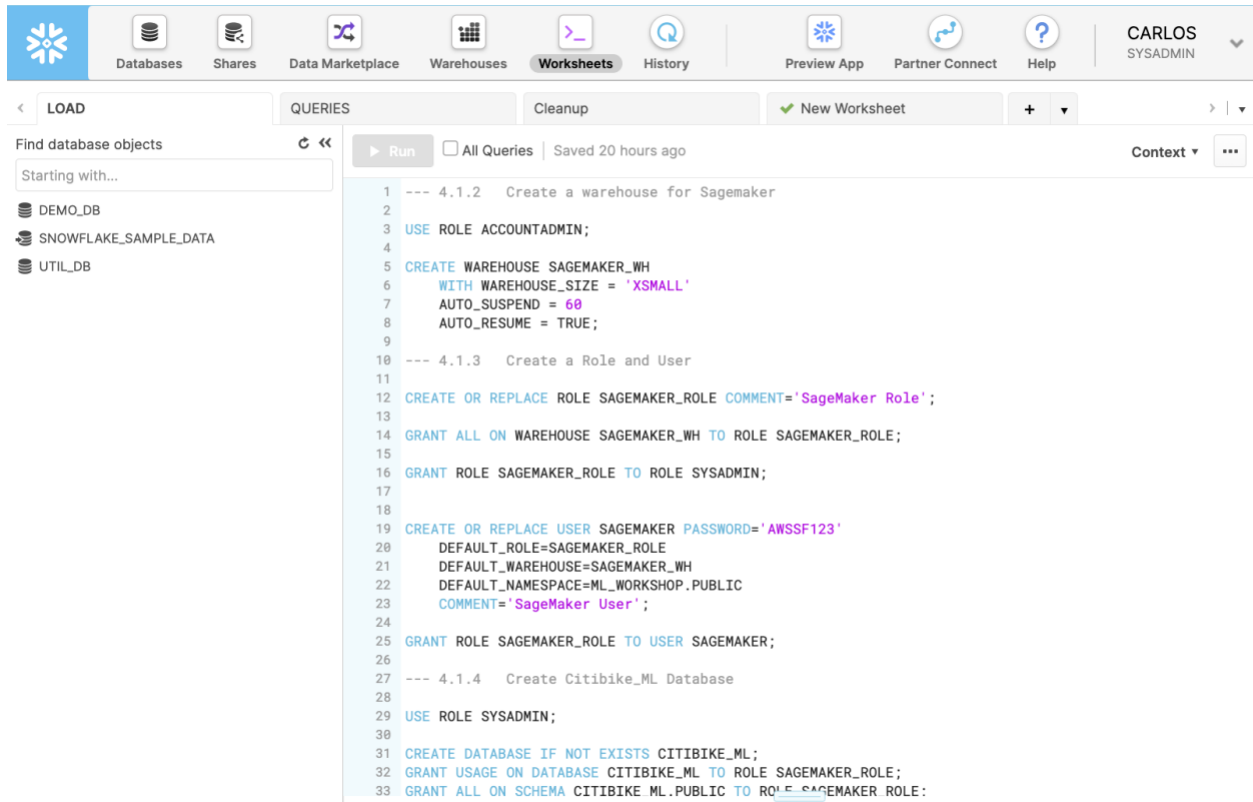
There are many ways to get data into Snowflake from many locations including the COPY command, Snowpipe auto-ingestion, an external connector, or a third-party ETL/ELT product. More information on getting data into Snowflake, see <https://docs.snowflake.net/manuals/user-guide-data-load.html>

We are using the COPY command and S3 storage for this module in a manual process so you can see and learn from the steps involved. In the real-world, a customer would likely use an automated process or ETL product to make the data loading process fully automated and much easier.

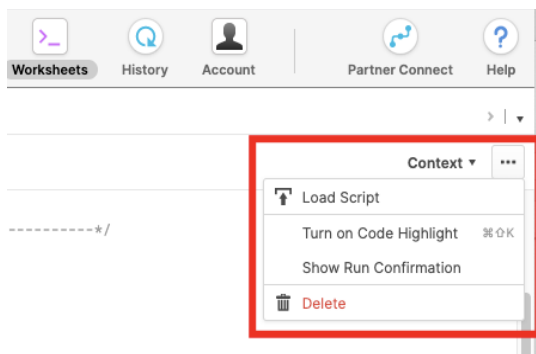
4.1 Start using Worksheets and Create a Virtual Warehouse

4.1.1 Prepare the worksheet

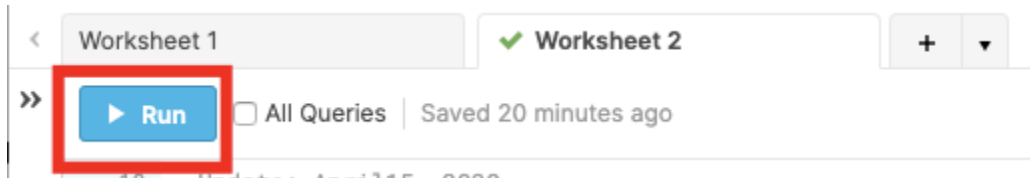
At the top of the Snowflake UI, click the Worksheets tab. You should see the worksheet with all the SQL we loaded in a prior step.



Before we start using SQL in Worksheets we will turn on Code Highlight by clicking on the 3 dots on the top right hand corner of the worksheet, and then clicking on Turn on Code Highlight. This will make it easier to identify the SQL that will be executed.



To execute a SQL command or commands, click or select multiple commands with your mouse. The SQL command(s) will be highlighted in BLUE. You can now either press COMMAND & RETURN on a Mac or CONTROL & ENTER on Windows; or you can click the RUN button towards the top left hand side of the Worksheet.



Warning

In this lab, never check the “All Queries” box at the top of the worksheet. We want to run SQL queries one at a time in a specific order; not all at once.

4.1.2 Create a warehouse for Sagemaker

Next we will briefly switch roles to the ACCOUNTADMIN role primarily to allow us to create a specific role for this workshop. Execute the SQL command shown below.

```
USE ROLE ACCOUNTADMIN;
```

Let's create a Warehouse called SAGEMAKER_WH.

Note: that the Warehouse is configured to auto suspend and resume, this prevents the unnecessary use of credits if the Warehouse is not being used with the convenience that it will automatically resume when needed

```
CREATE WAREHOUSE SAGEMAKER_WH  
  WITH WAREHOUSE_SIZE = 'XSMALL'  
  AUTO_SUSPEND = 60  
  AUTO_RESUME = TRUE;
```

Note this could also be done using the UI. Click on Warehouses and select Create and you will have the following window.

Create Warehouse

Name*

Size

X-Large (16 credits / hour)

▼

Learn more about virtual warehouse sizes [here](#)

Maximum Clusters

2

▼

Multi-cluster warehouses improve the query throughput for high concurrency workloads.

Minimum Clusters

1

▼

The number of active clusters will vary between the specified minimum and maximum values, based on number of concurrent users/queries.

Scaling Policy

Standard

▼

The policy used to automatically start up and shut down clusters.

Auto Suspend

10 minutes

▼

The maximum idle time before the warehouse will be automatically suspended.

☒ Auto Resume

?

Comment

Show SQL

Cancel

Finish

4.1.3 Create a Role and User

Now we will create a role named `SAGEMAKER_ROLE` that will be used to control access to objects in Snowflake. We will also `GRANT` all privileges on the `SAGEMAKER_WH` Warehouse to this role. We will also grant the `SAGEMAKER_ROLE` to the `SYSADMIN` role to allow `SYSADMIN` to have all the `SAGEMAKER_ROLE` privileges.

```
CREATE OR REPLACE ROLE SAGEMAKER_ROLE COMMENT='SageMaker Role';
GRANT ALL ON WAREHOUSE SAGEMAKER_WH TO ROLE SAGEMAKER_ROLE;
GRANT ROLE SAGEMAKER_ROLE TO ROLE SYSADMIN;
```

The next step is to create a user that will be used by external services to connect to the Snowflake account. The user `SAGEMAKER` will be created and assigned a default role, warehouse and database to establish the default context when connected to the Snowflake account.

Note that the SQL statement has a password “`AWSSF123`”, you can change the password to one that you prefer. Do note the password you use if you do change it as it will be required for the next portion of the lab when Amazon SageMaker will connect to Snowflake.

We will also grant the `SAGEMAKER_ROLE` to the `SAGEMAKER` user.

Finally, we will switch to the `SYSADMIN` role for the next steps.

```
CREATE OR REPLACE USER SAGEMAKER PASSWORD='AWSSF123'
  DEFAULT_ROLE=SAGEMAKER_ROLE
  DEFAULT_WAREHOUSE=SAGEMAKER_WH
  DEFAULT_NAMESPACE=ML_WORKSHOP.PUBLIC
  COMMENT='SageMaker User';
GRANT ROLE SAGEMAKER_ROLE TO USER SAGEMAKER;
```

4.1.4 Create Citibike_ML Database

First, let's create a database called CITIBIKE_ML that will be used for loading the Citibike trips. Then assign the usage on the database to the SAGEMAKER_ROLE. We will also grant all privileges on the default schema, PUBLIC, in the database to the SAGEMAKER_ROLE.

```
USE ROLE SYSADMIN;
CREATE DATABASE IF NOT EXISTS CITIBIKE_ML;
GRANT USAGE ON DATABASE CITIBIKE_ML TO ROLE SAGEMAKER_ROLE;
GRANT ALL ON SCHEMA CITIBIKE_ML.PUBLIC TO ROLE SAGEMAKER_ROLE;
```

Let's now switch context to the CITIBIKE_ML database and PUBLIC schema. As well as switch to use the SAGEMAKER_WH warehouse. The context for executing SQL commands is shown in the top right hand corner of the worksheets.

```
USE CITIBIKE_ML.PUBLIC;
USE WAREHOUSE SAGEMAKER_WH;
```



4.1.5 Create the Trips table

Next we will create the TRIPS table to load Citibike trips from a S3 bucket. All privileges will be granted to the SAGEMAKER_ROLE as well.

```
CREATE OR REPLACE TABLE TRIPS
(tripduration integer,
 starttime timestamp,
 stoptime timestamp,
 start_station_id integer,
 start_station_name string,
 start_station_latitude float,
 start_station_longitude float,
 end_station_id integer,
 end_station_name string,
 end_station_latitude float,
 end_station_longitude float,
 bikeid integer,
 membership_type string,
 usertype string,
 birth_year integer,
 gender integer);

GRANT ALL ON TABLE TRIPS TO ROLE SAGEMAKER_ROLE;
```

4.1.6 Create an External Stage

We are working with structured, comma-delimited data that has already been staged in a public, external S3 bucket. Before we can use this data, we first need to create an External Stage that specifies the location of our external bucket. We also need to create a File Format for the comma-delimited data.

NOTE - For this lab we are using an AWS-West bucket. In the real-world, to prevent data egress/transfer costs, you would want to select a staging location from the same region that your Snowflake environment is in.

Create an External Stage to an existing Amazon S3 Location. The data is located in a public S3 bucket and does not require credentials for ease of use with the lab. In practice you will need to configure secure access to Amazon S3. For more information see the Snowflake documentation

<https://docs.snowflake.com/en/user-guide/data-load-s3.html>

```
CREATE or replace STAGE CITIBIKE_ML.PUBLIC.citibike_trips URL =  
's3://snowflake-workshop-lab/citibike-trips';
```

We will also grant the necessary privileges to the SAGEMAKER_ROLE to the external stage.

```
GRANT USAGE ON STAGE CITIBIKE_ML.PUBLIC.citibike_trips TO ROLE  
SAGEMAKER_ROLE;
```

And verify you have access to the files that will be loaded

```
list @citibike_trips;
```

We can see there are 376 files that will be ingested into Snowflake.
















4.1.7 Create a File Format

Create the File Format that will be used. The CSV data does not have a header. We will also grant privileges on the File Format to the SAGEMAKER_ROLE.

```
create or replace FILE FORMAT CITIBIKE_ML.PUBLIC.CSV  
  COMPRESSION = 'AUTO'  
  FIELD_DELIMITER = ','  
  RECORD_DELIMITER = '\n'  
  SKIP_HEADER = 0  
  FIELD_OPTIONALLY_ENCLOSED_BY = '\042'  
  TRIM_SPACE = FALSE  
  ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE  
  ESCAPE = 'NONE'  
  ESCAPE_UNENCLOSED_FIELD = '\134'  
  DATE_FORMAT = 'AUTO'  
  TIMESTAMP_FORMAT = 'AUTO'  
  NULL_IF = ('');
```

```
GRANT USAGE ON FILE FORMAT CITIBIKE_ML.PUBLIC.CSV TO ROLE  
SAGEMAKER_ROLE;
```

Clicking in Databases at the top, and selecting CITIBIKE_ML we can see Tables, Views, Schemas, Stages, File Formats, Sequences and Pipes.

									
Databases > CITIBIKE_ML Last refreshed									
Tables	Views	Schemas	Stages	File Formats	Sequences	Pipes			
 Create...  Clone...  Edit...  Drop...  Transfer Ownership									
File Format	Schema	Type	Creation Time ▼	Owner	Comment				
CSV	PUBLIC	CSV	12:40:24 PM	SYSADMIN					

4.1.8 Load the data into the Trips table






Finally let's load the Trips data from the S3 bucket into Snowflake. First we switch to the SAGEMAKER_ROLE.

```
USE ROLE SAGEMAKER_ROLE;
```

Now we are going to use one of the Snowflake features that allow us to dynamically increase the size of the warehouse and decrease it later. Before running the load, we are going to increase from XS to L size so the load will be faster.

```
alter warehouse SAGEMAKER_WH set WAREHOUSE_SIZE = 'LARGE';
```

Note the size has change to (L):

 SAGEMAKER_R...	 SAGEMAKER_WH (L)	 CITIBIKE_ML	 PUBLIC ▼	
--	--	---	--	---





Then we will load the data using the COPY command.

```
copy into trips from @citibike_trips file_format=CSV;
```

Set the warehouse size back to xsmall:

```
alter warehouse SAGEMAKER_WH set WAREHOUSE_SIZE = 'XSMALL';
```

Verify it is now (XS):

 SAGEMAKER_R...	 SAGEMAKER_WH (XS)	 CITIBIKE_ML	 PUBLIC ▼
--	---	--	--

4.1.9 Verify data loading

Let's look at the data by running a SELECT command.


```
select * from trips limit 10;
```

The results are displayed in the frame below the Worksheet.

Row	TRIPDURATION	STARTTIME	STOPTIME	START_STATION	START_STATION	START_STATION	START_STATION	END_STATION_I	END_STATION_P	END_STATION_L
1	592	2013-11-11 0...	2013-11-11 0...	466	W 25 St & 6 ...	40.74395411	-73.99144871	459	W 20 St & 11...	40.746745
2	311	2013-11-11 0...	2013-11-11 0...	316	Fulton St & ...	40.70955958	-74.00653609	327	Vesey Pl & R...	40.7153379
3	1278	2013-11-11 0...	2013-11-11 0...	251	Mott St & Pri...	40.72317958	-73.99480012	488	W 39 St & 9 ...	40.75645824
4	532	2013-11-11 0...	2013-11-11 0...	545	E 23 St & 1 A...	40.736502	-73.97809472	345	W 13 St & 6 ...	40.73649403

4.1.10 Create a table for predictions

We are going to create a table to store results for one of our predictions.

```
CREATE OR REPLACE TABLE TRIPS_FORECAST
(ds date,
 yhat float,
 start_station_name string
);
```

```
GRANT ALL ON TABLE TRIPS_FORECAST TO ROLE SAGEMAKER_ROLE;
```

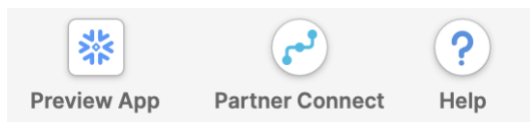
Also let's create a view to get the number of trips per day and station.

```
create or replace view trips_vw as
  select date_trunc('day', starttime) ds, start_station_name,
 count(*) trips
  from trips
 group by 1, 2
 order by 1 asc;
```

5 Using Snowflake Market Place

In order to build a better model to predict the number of trips for Citibike the Data Scientist would like to get access to more features. An important factor that could affect the rent of bikes is the weather condition. Using Snowflake Data Marketplace it is easy to get access to that data. With a few clicks we are going to make available a complete database with weather information to the Data Scientist.

First click on the “Preview App” icon.



Select Discover Data and click on Data Marketplace.

NAME ↑	SOURCE	CREATED
CITIBIKE_ML	Local	7 hours ago
DEMO_DB	Local	23 hours ago
SNOWFLAKE	Share	23 hours ago
SNOWFLAKE_SAMPLE_DATA	Share	23 hours ago
UTIL_DB	Local	23 hours ago

Snowflake Data Marketplace

Recent Listings

- SNL Bank Regulatory S&P Global Market Intelligence
- Textual Data Analytics: Sentiment Scores and Behavioral Metrics S&P Global Market Intelligence

Click on Weather and select Global Weather & and Climate Data for BI.

🏠 Home

Categories

📁 Business

👤 Demographics

⚡ Energy

💰 Financial

🏛️ Government

🏠 Health

📍 Local

📈 Marketing

📺 Media

🔒 Security

🚚 Transportation

✈️ Travel

⚡ Weather

Filter

☐ Ready to Query ⓘ

🔍 My Requests

S&P Global Market Intelligence

Weather Source

A curated continuum of analytics-grade historical, present, forecast and climatology data.

🔍 Personalized



AccuWeather

Actionable Weather Forecasts (Demo Product)

Hyper-localized forecasts to help businesses make the best weather-impacted decisions.



AccuWeather

Actionable Weather Forecasts

Hyper-localized forecasts to help businesses make the best weather-impacted decisions.

🔍 Personalized



Weather Source

Premier Weather Data for Analytics

A curated continuum of hyper-local past, present, forecast, and climatology weather data.

🔍 Personalized



Weather Source

Global Weather & Climate Data for BI

Daily and hourly past, forecast & climatology weather data for select cities

Click on Get Data:



Global Weather & Climate Data for BI

Weather Source Weather Daily

Get Data

Global Weather & Climate Data for BI

Weather Source data, products, and solutions were built for analytics and machine learning. Since 2015, Weather Source has empowered companies to quantify the impact of weather on their operations and increase return on investment, fine-tune logistics, strengthen supply chains, optimize marketing, improve resource planning, and more. Seamlessly integrate weather and climate data with your internal data for insightful analytics and actionable business intelligence.

Name your database **WEATHER**. Grant access to the SAGEMAKER_ROLE and SYSADMIN role.

Get Data

Create a database to query data from the listing. This database will not take up any storage space in your account.

Database name

WEATHER

Which roles, in addition to ACCOUNTADMIN, can access this database?

SAGEMAKER_ROLE, SYSADMIN

☒ I accept [Snowflake's consumer terms](#) and [Weather Source's terms of use](#).

Snowflake processes the personal information you provide us in accordance with our [Privacy Notice](#).

Cancel

Create Database

If you go to your database objects and refresh the view, the WEATHER database is available and you can browse all their objects.

Find database objects

Starting with...

- CITIBIKE_ML
- DEMO_DB
- SNOWFLAKE_SAMPLE_DATA
- UTIL_DB
- WEATHER**
 - INFORMATION_SCHEMA
 - POSTCODE
 - PUBLIC
 - No Tables in this Schema
 - Views
 - CLIMATOLOGY_DAY
 - FORECAST_DAY
 - FORECAST_DAY_COVID_19
 - HISTORY_DAY
 - HISTORY_DAY_COVID_19

Run All Queries Saved 0 seconds ago

```

1  -- Create the ware house
2
3  USE ROLE ACCOUNTADMIN;
4
5  CREATE WAREHOUSE SAGEMAKER_WH
6
7
8
9
10
11
12 CREATE OR REPLACE ROLE SAGEMAKER_ROLE COMMENT='Sa
13
14 GRANT ALL ON WAREHOUSE SAGEMAKER_WH TO ROLE SAGEM.
15
16 GRANT ROLE SAGEMAKER_ROLE TO ROLE SYSADMIN;
17
18 /* Create one specific user for Sagemaker */
19
20 CREATE OR REPLACE USER SAGEMAKER PASSWORD='AWSSF1:
21     DEFAULT_ROLE=SAGEMAKER_ROLE
22     DEFAULT_WAREHOUSE=SAGEMAKER_WH
23     DEFAULT_NAMESPACE=ML_WORKSHOP.PUBLIC
24     COMMENT='SageMaker User';
25

```

Database: WEATHER (share)
 Created on: 10/24/2020, 6:22:04 PM
 Owner: ACCOUNTADMIN
 Origin: WEATHERSOURCE_PARTNER.WS_ONPOINT_WEATHER_DATA_SHARE

Please note that no data has been copied or moved into our account. The data is at the provider and we use our compute resources (warehouses) to query it. The database is immediately available for us without any need to transfer data. Also, as soon as the provider adds more data, it will also be available for us.

Grant imported privileges on the database to SAGEMAKER_ROLE.

```

USE ROLE ACCOUNTADMIN;

grant imported privileges on database weather to role
SAGEMAKER_ROLE;

```

Now we have data for Trips and Weather available for our Data Scientists. Here some queries to have a first impression of the data we have available.

Let's take a look to some of the data from WEATHER database:

```

USE ROLE SAGEMAKER_ROLE;

USE CITIBIKE_ML.PUBLIC;

USE WAREHOUSE SAGEMAKER_WH;

select * from WEATHER.PUBLIC.HISTORY_DAY
      where POSTAL_CODE = '06101'

```

```
limit 10;
```

Results Data Preview Open History

✓ Query ID SQL 1.36s 10 rows

Filter result... Download Copy Columns ▾

Row	COUNTRY	POSTAL_CODE	DATE_VALID_STD	DOY_STD	MIN_TEMPERAT	AVG_TEMPERAT	MAX_TEMPERAT	MIN_TEM
1	US	06101	2013-02-01	32	21.2	26.8	31.3	
2	US	06101	2013-01-27	27	10.2	20.8	32.2	
3	US	06101	2013-02-04	35	21.3	25.4	29.6	
4	US	06101	2013-02-17	48	15.7	23.3	28.0	
5	US	06101	2013-02-16	47	28.5	34.2	40.1	

We have a total of 61M+ trips we can use for insights:

```
select count(*) num_trips from trips;
```

✓ Query ID SQL 47ms 1 rows

Filter result... Download Copy Columns ▾

Row	NUM_TRIPS
1	61468359

I can join the data from TRIPS, that I uploaded using CSV files with the data from the WEATHER database. This query will be used by our Data Scientist to know the number of trips per day and the average temperature of that day.

```
select date_trunc('day', STARTTIME) day,
avg(w.MAX_TEMPERATURE_FEELS LIKE_2M_F) temp, count(*) trips
from trips t, WEATHER.PUBLIC.HISTORY_DAY w
where day = w.DATE_VALID_STD and w.POSTAL_CODE = '06101'
group by 1
order by 1;
```

We are going to have 1848 days to train our time series model in order to make predictions:

✓ Query ID SQL

11.2s

1,848 rows

Filter result...



Copy

Columns ▾

Row	DAY	TEMP	TRIPS
1	2013-06-01 00:00:00.000	95.7000000	8722
2	2013-06-02 00:00:00.000	90.4000000	15971
3	2013-06-03 00:00:00.000	81.6000000	7598
4	2013-06-04 00:00:00.000	72.7000000	15782
5	2013-06-05 00:00:00.000	76.6000000	15690
6	2013-06-06 00:00:00.000	72.8000000	12420

6 Machine Learning in SageMaker Notebook

In this module we are going to review the SageMaker Notebook that will be run. This Notebook will use the known Prophet library to make simple predictions for number of trips. All data to train our model is available at Snowflake, so the Data Scientists have a single source of truth for their models. We have seen in the previous module how to use the Snowflake Data Marketplace to get access to weather data.

The Notebook first build a model using only the number of trips per day information. Later, we add the weather temperature for each day and feed the model with that information. All that data is available with a simple SQL query, reducing the time spent in data preparation for the Data Scientist.

The Notebook is well explained with comments and we encourage you to carefully read through each section.

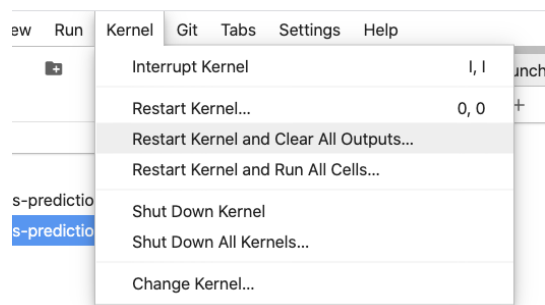
6.1 Preparation

6.1.1 Libraries to be installed

Verify that the first cell has completed the execution, so we have all the libraries needed to run this Notebook. Remember that when `[*]` appears in the cell that means it is still running. Once the cell has been completed there will be a number. This cell was executed at the beginning of this lab in preparation. Double check it has completed.

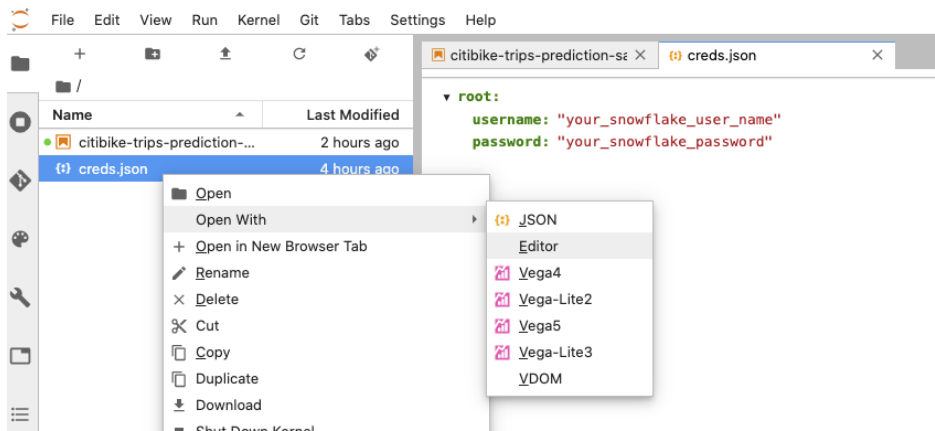
```
[1]: !pip install --upgrade pip && pip install --upgrade numpy
!pip install --upgrade snowflake-connector-python[pandas]
!conda install -c conda-forge matplotlib --yes
!conda install -c plotly plotly==3.10.0 --yes
!conda install -c conda-forge fbprophet --yes
!pip install --upgrade pystan
!pip install --upgrade fbprophet
```

Then restart the Kernel and Clear All Outputs:



6.1.2 Set Credentials

Set your credentials for the user we created previously within Snowflake. The user should be SAGEMAKER and set the password you created previously. Edit the [creds.json](#) file you have downloaded and uploaded into the Notebook.



Run the cell that will set the variables for the credentials.

```
import json

with open('creds.json') as f:
    data = json.load(f)
    username = data['username']
    password = data['password']
```

6.1.3 Prepare connection

Next we import the libraries we are going to be using. The Snowflake connector in order to query and write results into Snowflake, Pandas to manage results and Prophet which is the ML model we will be training.

```
import snowflake.connector
import pandas as pd
from fbprophet import Prophet
```

In the next cell, modify SF_ACCOUNT to point to your own Snowflake account. The rest of the parameters should be ok if you have followed the instructions so far about creating roles and the database.

```

SF_ACCOUNT = 'tnaxxxxxxx'
SF_WH = 'SAGEMAKER_WH'

# Connecting to Snowflake using the default authenticator
ctx = snowflake.connector.connect(
    user=username,
    password=password,
    account=SF_ACCOUNT,
    warehouse=SF_WH,
    role='SAGEMAKER_ROLE',
    database='CITIBIKE_ML',
    schema='public'
)

cur=ctx.cursor()

```

6.2 Forecast number of trips per day

6.2.1 Basic model

Let's bring the first data set for our model. Using SQL we can easily get the number of trips per day. We can see we have 1847 days.

```

# Lets aggregate the trips data up to the day level
sql = "select date_trunc('day', STARTTIME), count(*) from trips" + \
      " group by date_trunc('day', STARTTIME)" + \
      " order by date_trunc('day', STARTTIME)"
cur.execute(sql)

# Fetch the result set from the cursor and deliver it as the Pandas DataFrame.
df = cur.fetch_pandas_all()
# The FB Prophet library expects the data to be in two columns: 'ds' for timestamp and 'y' for the value
df.columns=['ds', 'y']
df.tail()

```

	ds	y
1843	2018-06-26	80646
1844	2018-06-27	67698
1845	2018-06-28	50432
1846	2018-06-29	69859
1847	2018-06-30	54721

Define a Prophet model and fit it using the data from the query.

```

model = Prophet()
model.fit(df)

```

Prophet can provide metrics to validate the precision of the model. Let's use it with one horizon of 180 days and we will take a look to the RMSE (root-mean-square deviation) as a value to see if we can improve our model.

```

from fbprophet.diagnostics import cross_validation, performance_metrics

cutoffs = pd.to_datetime(['2018-01-01'])
df_cv = cross_validation(model, cutoffs=cutoffs, horizon = '180 days')

df_p = performance_metrics(df_cv, rolling_window=1)
print(df_p)

```

	horizon	mse	rmse	mae	mape	mdape	\
0	180 days	1.562338e+08	12499.352575	10069.51157	0.549224	0.185693	

	coverage
0	0.566667

That means our prediction is failing on average 12499 trips.

Let's predict a longer period and we will look how it looks like in a graph.

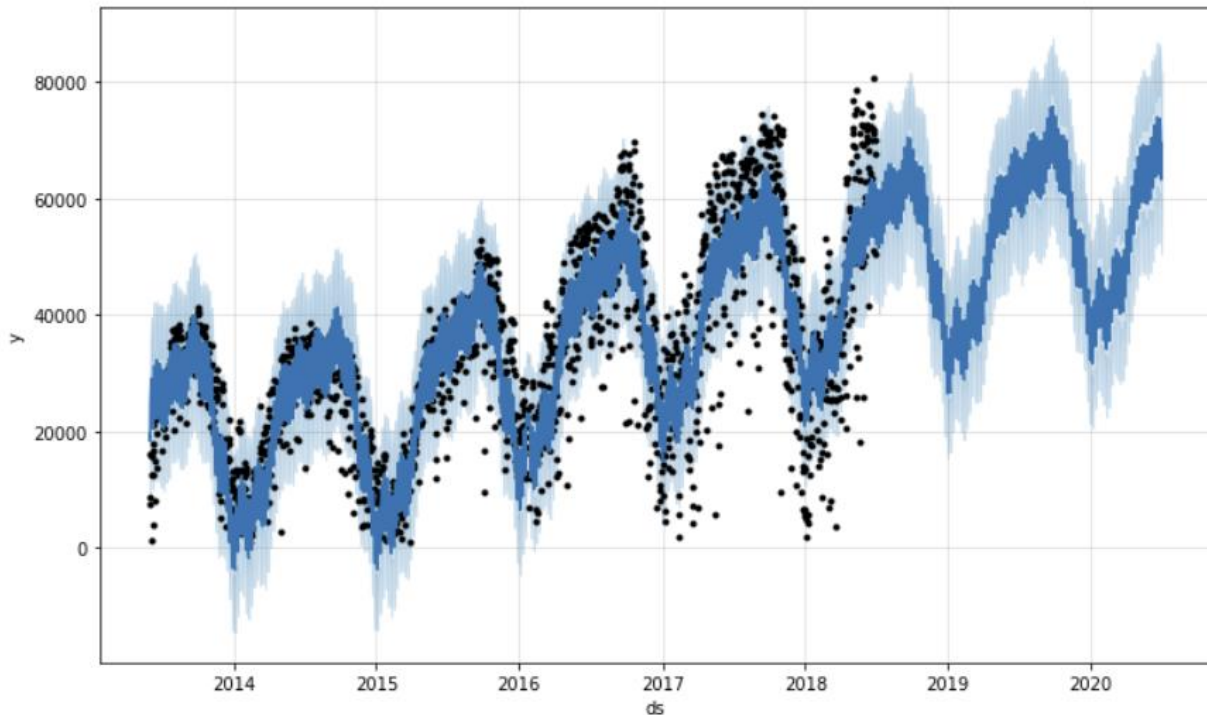
```

future = model.make_future_dataframe(periods=730)
forecast = model.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()

```

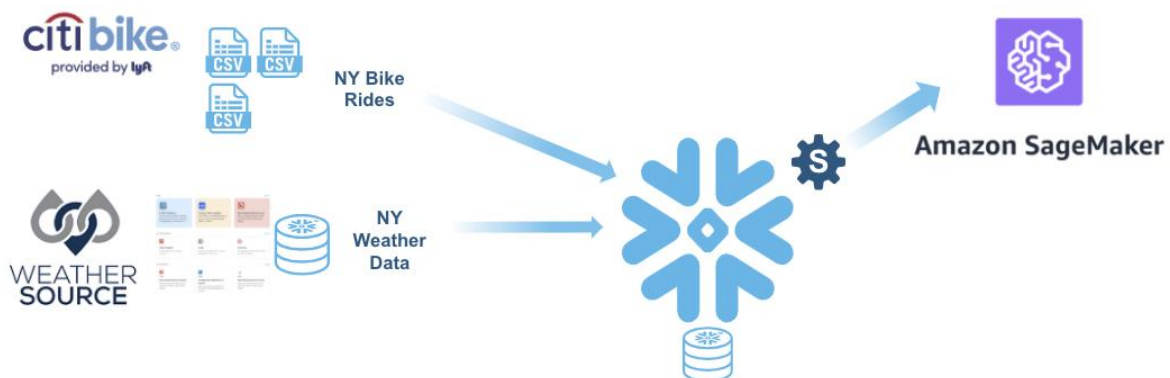
	ds	yhat	yhat_lower	yhat_upper
2573	2020-06-25	72717.174994	59903.940462	85673.830777
2574	2020-06-26	70899.602596	57958.463195	83037.485872
2575	2020-06-27	64803.908157	52299.409699	77946.511745
2576	2020-06-28	63206.267027	50496.471468	76380.383143
2577	2020-06-29	69406.132985	56933.468527	81735.297008

```
fig1 = model.plot(forecast)
```



6.2.2 Enhancing the model accuracy by enriching data

Our next step is to see if we can enrich the data set and build a more precise model by using Snowflake Data Marketplace and Data Sharing. In a previous module we made the weather database available for the Data Scientist. By doing that, now we can run queries where the trips data can be joined with weather data. Our goal is to introduce in the model what is the daily temperature along with the number of trips per day.



This can be done with a simple query where we join trips and weather data. Now we have our 1847 days that include number of trips and the average temperature for that day.

```
cur=ctx.cursor()

sql = "select date_trunc('day', STARTTIME) d, avg(w.AVG_TEMPERATURE_FEELS LIKE_2M_F), count(*) \
from trips t, WEATHER.PUBLIC.HISTORY_DAY w \
where d = w.DATE_VALID_STD and w.POSTAL_CODE = '06101' \
group by 1 \
order by 1;"

cur.execute(sql)

# Fetch the result set from the cursor and deliver it as the Pandas DataFrame.
df = cur.fetch_pandas_all()
df.columns=['ds', 't', 'y']
df.tail()
```

```
INFO:snowflake.connector.cursor:query: [select date_trunc('day', STARTTIME) d, avg(w.AVG_TEMPERA
TURE_FEELS LIKE_2M_F), co...]
INFO:snowflake.connector.cursor:query execution done
INFO:snowflake.connector.arrow_result:fetching data into pandas dataframe done
```

	ds	t	y
1843	2018-06-26	66.2000000	80646
1844	2018-06-27	68.1000000	67698
1845	2018-06-28	74.2000000	50432
1846	2018-06-29	78.5000000	69859
1847	2018-06-30	81.4000000	54721

We create another model where we add the temperature as a new parameter to our model and then fit it.

```
enhanced_model = Prophet()
enhanced_model.add_regressor('t')
enhanced_model.fit(df)
```

And let's run the same validation we did before now using the new model.

```
cutoffs = pd.to_datetime(['2018-01-01'])
df_cv = cross_validation(enhanced_model, cutoffs=cutoffs, horizon = '180 days')

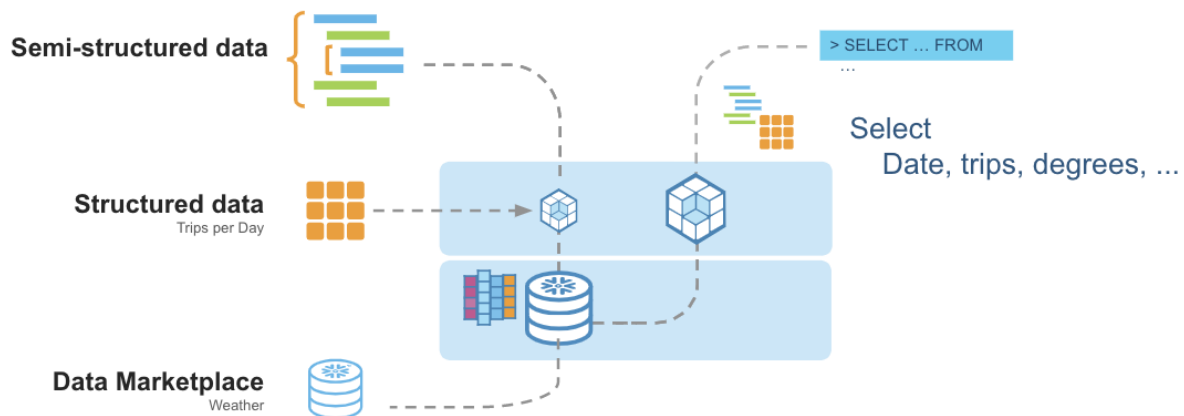
df_p = performance_metrics(df_cv, rolling_window=1)
print(df_p)
```

```
HBox(children=(FloatProgress(value=0.0, max=1.0), HTML(value='')))
```

	horizon	mse	rmse	mae	mape	mdape
0	180 days	1.486967e+08	12194.12404	9801.086912	0.509621	0.190557

	coverage
0	1.0

Here we can see how the RMSE has been reduced from the previous model. We could add more features in a very simple way like rain to see if that could make an effect to our model. This would be quite easy for the Data Scientist as all the data manipulation is done at Snowflake. We only have to ask what information we want. Snowflake can also use semi-structured data like JSON, XML, Parquet, etc.. All that information will be at the service of the Data Scientist to build a better model.



6.3 Predicting usage at the stations

6.3.1 Basic model

In the same way, we can easily create new models to predict how many bikes should be placed at some stations. All the information needed to build the model is within Snowflake and it is simple to query it. First, we can create a model just using number of trips starting from one specific station.

```
cur=ctx.cursor()

sql = "select start_station_name, date_trunc('day', STARTTIME) d, count(*) \
from CITIBIKE_ML.public.trips \
where start_station_name like 'E 17 St & Broadway' \
group by 1, 2 \
order by 2"

cur.execute(sql)

# Fetch the result set from the cursor and deliver it as the Pandas DataFrame.
df = cur.fetch_pandas_all()
df.columns=['station', 'ds', 'y']
df.tail()
```

	station	ds	y
1841	E 17 St & Broadway	2018-06-26	478
1842	E 17 St & Broadway	2018-06-27	307
1843	E 17 St & Broadway	2018-06-28	299
1844	E 17 St & Broadway	2018-06-29	436
1845	E 17 St & Broadway	2018-06-30	300

And create and fit the model for that station.

```
station_model = Prophet(daily_seasonality=True)
station_model.add_country_holidays('US')
station_model.fit(df)
```

As we did before, let's get performance metrics for the model.

```
cutoffs = pd.to_datetime(['2018-01-01'])
df_cv = cross_validation(station_model, cutoffs=cutoffs, horizon = '180 days')
df_p = performance_metrics(df_cv, rolling_window=1)
print(df_p)
```

```
HBox(children=(FloatProgress(value=0.0, max=1.0), HTML(value='')))
```

	horizon	mse	rmse	mae	mape	mdape	coverage
0	180 days	6414.005765	80.087488	58.588887	0.774781	0.15403	0.783333

Here the RMSE is 80 bikes per station.

6.3.2 Enhancing the model accuracy by enriching data

Let's use the temperature from the weather database to see if we can improve the model. We again join trips and weather tables and get the number of trips and temperature for one specific station.

```
cur=ctx.cursor()

sql = "select start_station_name, date_trunc('day', STARTTIME) d, avg(w.AVG_TEMPERATURE_FEELS LIKE_2M_F), count(*) \
from trips t, WEATHER.PUBLIC.HISTORY_DAY w \
where start_station_name like 'E 17 St & Broadway' \
and d = w.DATE_VALID_STD and w.POSTAL_CODE = '06101' \
group by 1, 2 \
order by 2;"

cur.execute(sql)

# Fetch the result set from the cursor and deliver it as the Pandas DataFrame.
df = cur.fetch_pandas_all()
df.columns=['station', 'ds', 't', 'y']
df.tail()
```

```
INFO:snowflake.connector.cursor:query: [select start_station_name, date_trunc('day', STARTTIME) d, avg(w.AVG_TEMPERA
TURE...)]
INFO:snowflake.connector.cursor:query execution done
INFO:snowflake.connector.arrow_result:fetching data into pandas dataframe done
```

	station	ds	t	y
1841	E 17 St & Broadway	2018-06-26	66.2000000	478
1842	E 17 St & Broadway	2018-06-27	68.1000000	307
1843	E 17 St & Broadway	2018-06-28	74.2000000	299
1844	E 17 St & Broadway	2018-06-29	78.5000000	436
1845	E 17 St & Broadway	2018-06-30	81.4000000	300

Create the model adding the new regressor for the temperature and fit it.

```

enhanced_station_model = Prophet(daily_seasonality=True)
# add a regressor for the temperature feature
enhanced_station_model.add_country_holidays('US')
enhanced_station_model.add_regressor('t')
enhanced_station_model.fit(df)

```

And run the evaluation for the new model.

```

cutoffs = pd.to_datetime(['2018-01-01'])
df_cv = cross_validation(enhanced_station_model, cutoffs=cutoffs, horizon = '180 days')
df_p = performance_metrics(df_cv, rolling_window=1)
print(df_p)

```

```
HBox(children=(FloatProgress(value=0.0, max=1.0), HTML(value='')))
```

	horizon	mse	rmse	mae	mape	mdape	coverage
0	180 days	5840.401284	76.422518	57.556376	0.715965	0.170776	1.0

Here for this model we also have a better prediction as we have reduced RMSE from 80 to 76.

6.3.3 Predict next days for one station

Final step will be to run some predictions and store it back into Snowflake. First, we are going to go to the weather database and get the temperature for the next 15 days after our last day in the database.

```

cur=ctx.cursor()

last_day = str(df['ds'].iloc[-1])

sql = "select DATE_VALID_STD ds, AVG_TEMPERATURE_FEELS LIKE_2M_F t from WEATHER.public.HISTORY_DAY \
      where POSTAL_CODE like '06101' \
      and DATE_VALID_STD < dateadd(day, 15, to_date('' + last_day + '')) \
      order by DATE_VALID_STD;"

cur.execute(sql)

# Fetch the result set from the cursor and deliver it as the Pandas DataFrame.
df_future = cur.fetch_pandas_all()
df_future.columns=['ds', 't']
df_future.tail()

```

```

INFO:snowflake.connector.cursor:query: [select DATE_VALID_STD ds, AVG_TEMPERATURE_FEELS LIKE_2M_F t from WEATHER.publ
ic.H...]
INFO:snowflake.connector.cursor:query execution done
INFO:snowflake.connector.arrow_result:fetching data into pandas dataframe done

```

	ds	t
2016	2018-07-10	80.3
2017	2018-07-11	75.8
2018	2018-07-12	72.7
2019	2018-07-13	75.5
2020	2018-07-14	76.1

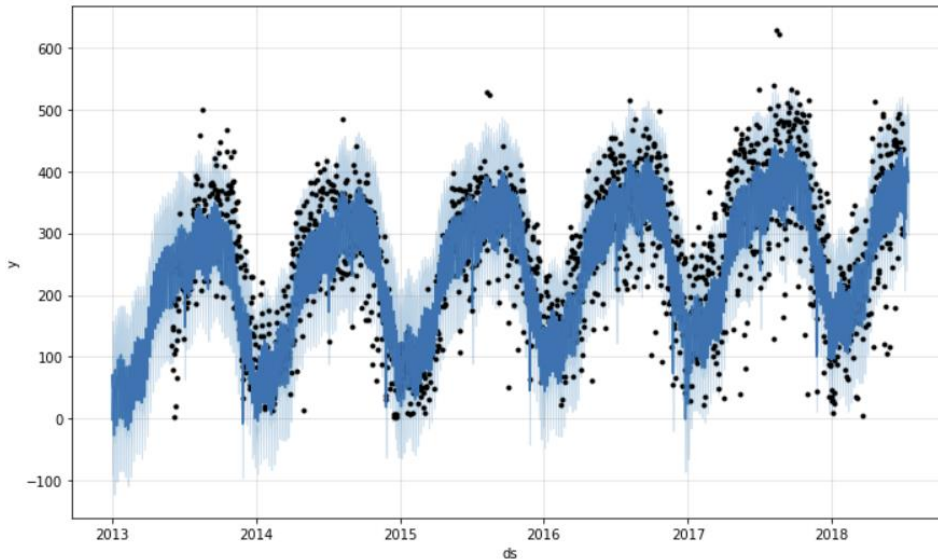
And now we can predict the model with that data frame.


```
forecast = enhanced_station_model.predict(df_future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

	ds	yhat	yhat_lower	yhat_upper
2016	2018-07-10	394.641750	303.025606	480.675428
2017	2018-07-11	421.473189	333.772761	509.367601
2018	2018-07-12	400.494089	315.818231	496.890183
2019	2018-07-13	400.083569	309.811508	484.361737
2020	2018-07-14	382.498070	292.414414	469.873214

We can plot the model for the specific station.

```
fig2 = enhanced_station_model.plot(forecast)
```



6.3.4 Write prediction back into Snowflake

Snowflake can be used to store the results for all ML activities. Not only that, using Zero-Copy Snapshots, a copy of the data used for training could also be maintained without any additional extra capacity.

Here we are going to get from the forecast just the values for date and the prediction made (yhat).

```
forecast = forecast[['ds', 'yhat']]
forecast.columns = ['DS', 'YHAT']
forecast['START_STATION_NAME'] = 'E 17 St & Broadway'
```

We create a function that will be used to store the prediction in the TRIPS_FORECAST table.

```
def insert_into_trips_forecast (predictions):

    cur=ctx.cursor()

    sql = "INSERT INTO TRIPS_FORECAST VALUES"

    count = 0
    for index, row in predictions.iterrows():
        count += 1
        sql += "(" + str(row['DS']) + ", " + str(float(row['YHAT'])) + ", " + str(row['START_STATION_NAME']) + ")"
        if (count % 10000 == 0):
            ctx.cursor().execute(sql)
            print("Flushing 10000 records: ", count)
            sql = "INSERT INTO TRIPS_FORECAST VALUES"
        else:
            if count < len(predictions):
                sql += ","

    ctx.cursor().execute(sql)

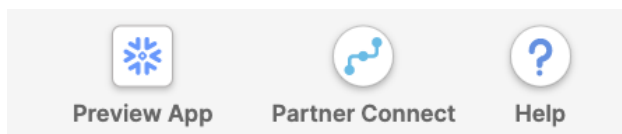
    print("Total records inserted: ", count)
```

Call that function with the forecast so it will be stored back in Snowflake.

```
insert_into_trips_forecast (forecast)
```

6.4 Review the prediction using Snowsight

We are going to go back to the Snowflake UI to review the TRIPS_FORECAST table for the last prediction performed. Once in the UI, click on Preview App to open Snowsight.



Click on Worksheet to add a new one.



Here we can run several queries to visualize the results we have got. First make sure we are using our CITIBIKE_ML database:

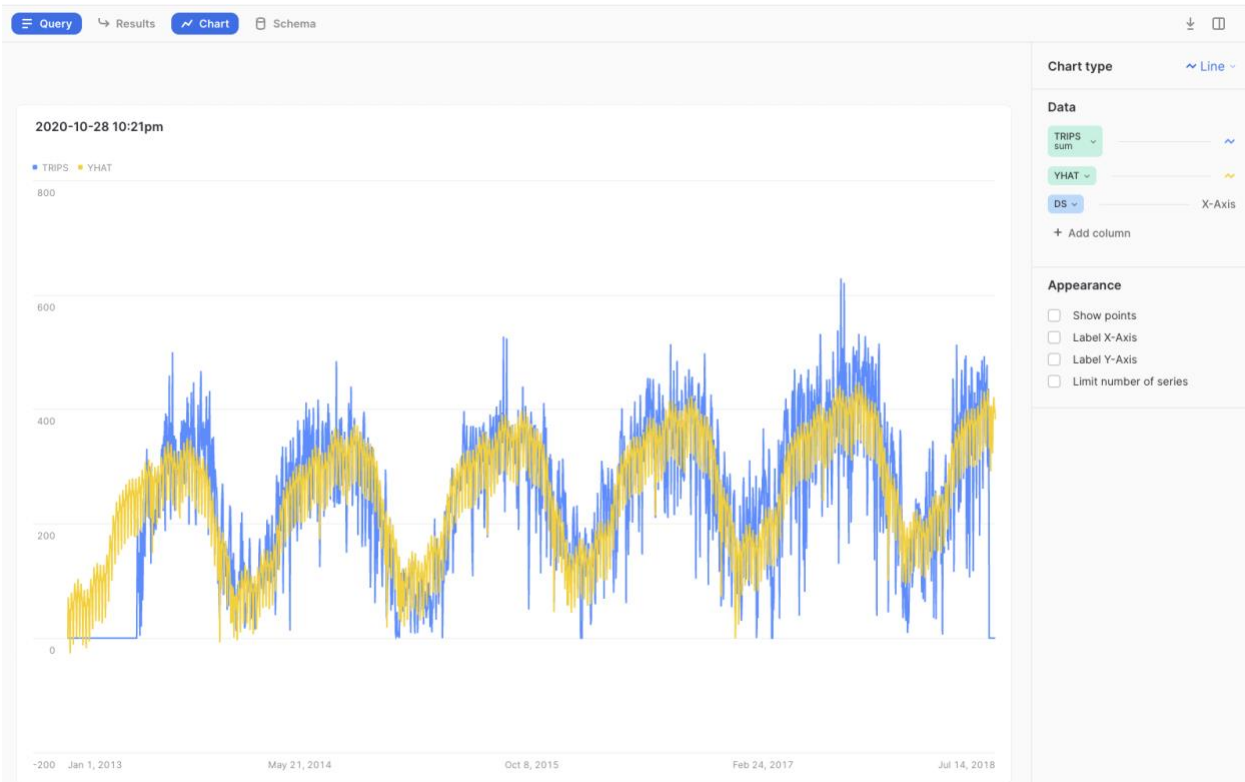
```
use database citibike_ml;
```

The following join query will give us a view of the real number of trips and the predicted one, including the last 15 days where we only had predictions.

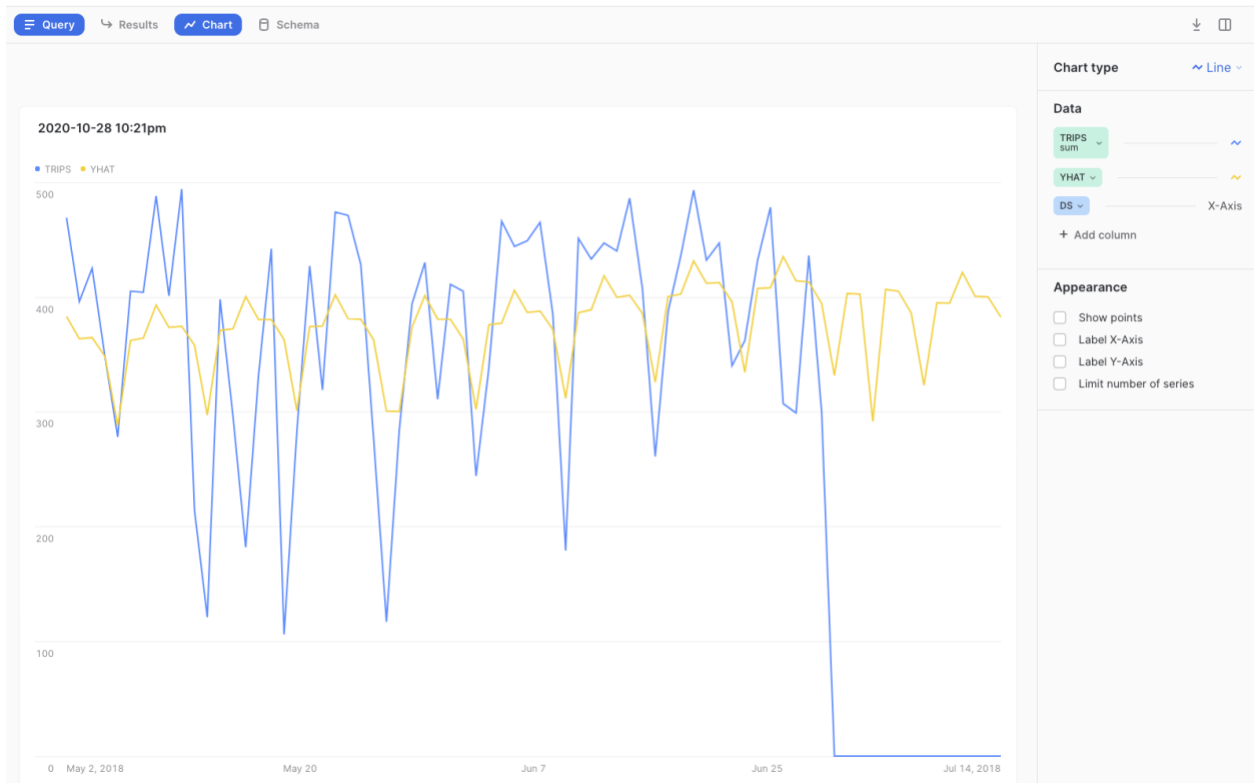
```
select f.ds, trips, yhat
from trips_forecast f left join trips_vw t
```

```
on f.ds = t.ds
and f.start_station_name = t.start_station_name
order by f.ds asc;
```

Select chart. In the chart type, select line, get DS for the X-Axis and add YHAT and TRIPS for the Y-Axis.



And run the following query to just visualize the latest months with the prediction.



7 Summary & Next Steps

This lab was designed as a hands-on introduction to Snowflake and SageMaker Notebook to simultaneously teach you how to use it, while showcasing some of its key capabilities.

We encourage you to continue with your free Snowflake trial by loading in your own sample or production data and by using some of the more advanced capabilities of Snowflake not covered in this lab. There are several ways Snowflake can help you with this:

At the very top of the UI click on the “Partner Connect” icon to get access to trial/free ETL and BI tools to help you get more data into Snowflake and then analyze it

Read the “Definitive Guide to Maximizing Your Free Trial” document at: <https://www.snowflake.com/test-driving-snowflake-the-definitive-guide-to-maximizing-your-free-trial/>

Attend a Snowflake virtual or in-person event to learn more about our capabilities and how customers use us <https://www.snowflake.com/about/events/>

Contact Sales to learn more <https://www.snowflake.com/free-trial-contact-sales/>

We also encourage you to continue to explore the capabilities of Amazon SageMaker. For other SageMaker Machine Learning examples visit: <https://docs.aws.amazon.com/sagemaker/latest/dg/howitworks-nbexamples.html>

8 Resetting Your Snowflake Environment

You can reset your Snowflake environment by running the SQL commands in the Worksheet:

```
USE ROLE ACCOUNTADMIN;
```

```
drop warehouse SAGEMAKER_WH;
```

```
drop user SAGEMAKER;
```

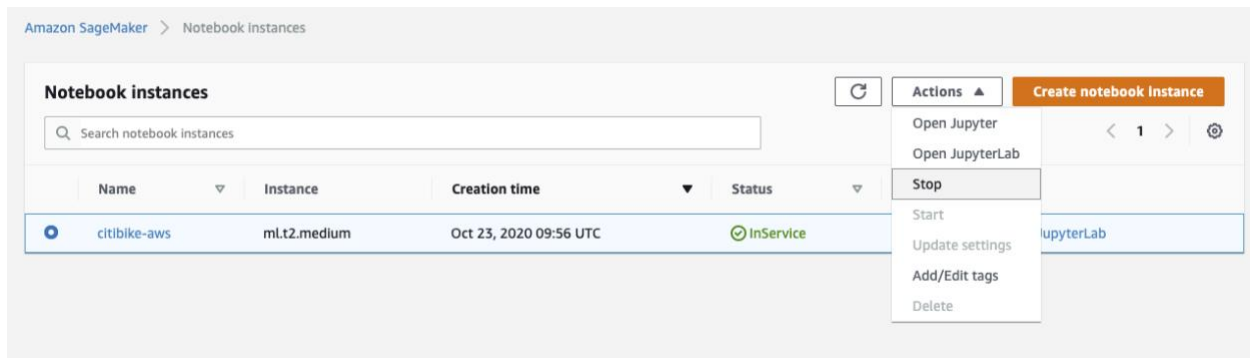
```
drop role SAGEMAKER_ROLE;
```

```
drop database CITIBIKE_ML;
```

9 Resetting Your AWS & SageMaker Environment

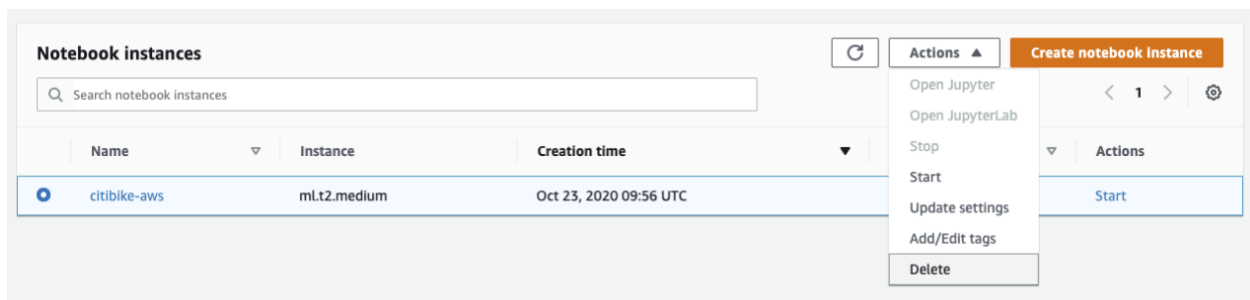
To avoid incurring charge for the AWS Services that were deployed for the lab you will need to remove the services following these steps.

Go to the Notebook instances within Sagemaker. Select your instance, Actions and Stop.



The screenshot shows the Amazon SageMaker Notebook instances page. At the top, there's a breadcrumb 'Amazon SageMaker > Notebook instances'. Below it, a search bar is labeled 'Search notebook instances'. A table lists the instances with columns: Name, Instance, Creation time, and Status. One instance is listed: 'citibike-aws' with instance type 'ml.t2.medium' and creation time 'Oct 23, 2020 09:56 UTC'. Its status is 'InService'. To the right of the table, an 'Actions' dropdown menu is open, showing options: 'Open Jupyter', 'Open JupyterLab', 'Stop', 'Start', 'Update settings', 'Add/Edit tags', and 'Delete'. A 'Create notebook instance' button is also visible at the top right.

Select the instance again and click on Delete.



This screenshot is similar to the previous one, showing the same SageMaker Notebook instances page. The 'Actions' dropdown menu is open, and the 'Delete' option is highlighted at the bottom of the list. The table and other UI elements remain the same.