



# 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

[Lab Overview](#)

[Module 1: Prepare Your Lab Environment](#)

[Module 2: The Snowflake User Interface](#)

[Module 3: Preparing to Load Data & Loading Data](#)

[Module 4: Deploy an Amazon SageMaker Notebook](#)

[Module 5: Machine Learning Workflow in SageMaker](#)

[Module 6: Upload the Churn Scores to Snowflake](#)

[Summary & Next Steps](#)

## Lab Overview

In this lab, you'll learn the basics of creating an advanced analytics solution. We'll train a customer churn prediction model from data in Snowflake using SageMaker and load the churn risk scores back into Snowflake for analysis.

## 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.

## 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
- Visualize the data and perform basic feature engineering
- Unload a dataset into S3 and use it to train a machine learning model
- Run a batch of data through your model and load the results back into Snowflake

## Prerequisites

- Use of the Snowflake free 30-day trial environment
- Use of an AWS Account with the ability to launch a CloudFormation template, create a S3 bucket and SageMaker Instance
- Basic knowledge of SQL, and database concepts and objects
- Familiarity with CSV comma-delimited files
- Basic Jupyter notebook and Python knowledge

# Module 1: Prepare Your Lab Environment

## 1.1 Steps to Prepare Your Lab Environment

- 1.1.1** 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 user name and password. Write down these credentials.
- 1.1.2** 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 use a Cloudformation Template, create a S3 bucket and 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.
- 1.1.3** 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.
- 1.1.4** Download the Snowflake and SageMaker files to your local machine.
- Click on [https://snowflake-corp-se-workshop.s3-us-west-1.amazonaws.com/sagemaker-snowflake-devdays-v1/notebooks/sagemaker\\_workshop\\_v1.4.sql](https://snowflake-corp-se-workshop.s3-us-west-1.amazonaws.com/sagemaker-snowflake-devdays-v1/notebooks/sagemaker_workshop_v1.4.sql) and download the file. This file contains pre-written SQL commands and we will use this file later in the lab.
  - Click on <https://snowflake-corp-se-workshop.s3-us-west-1.amazonaws.com/sagemaker-snowflake-devdays-v1/notebooks/workshop-snowflake-sagemaker-v1.4.ipynb> and download the file. This file contains the SageMaker Notebook and we will use this file later in the lab.

## Module 2: The Snowflake User Interface

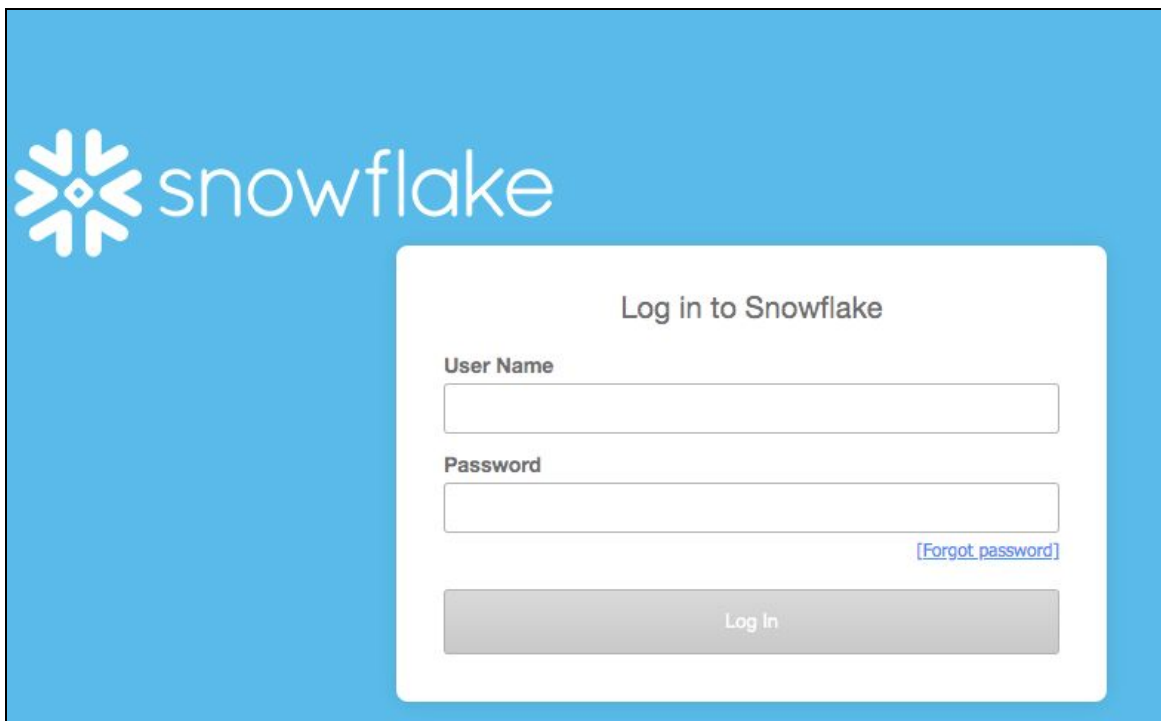


### About the screen captures, sample code, and environment

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

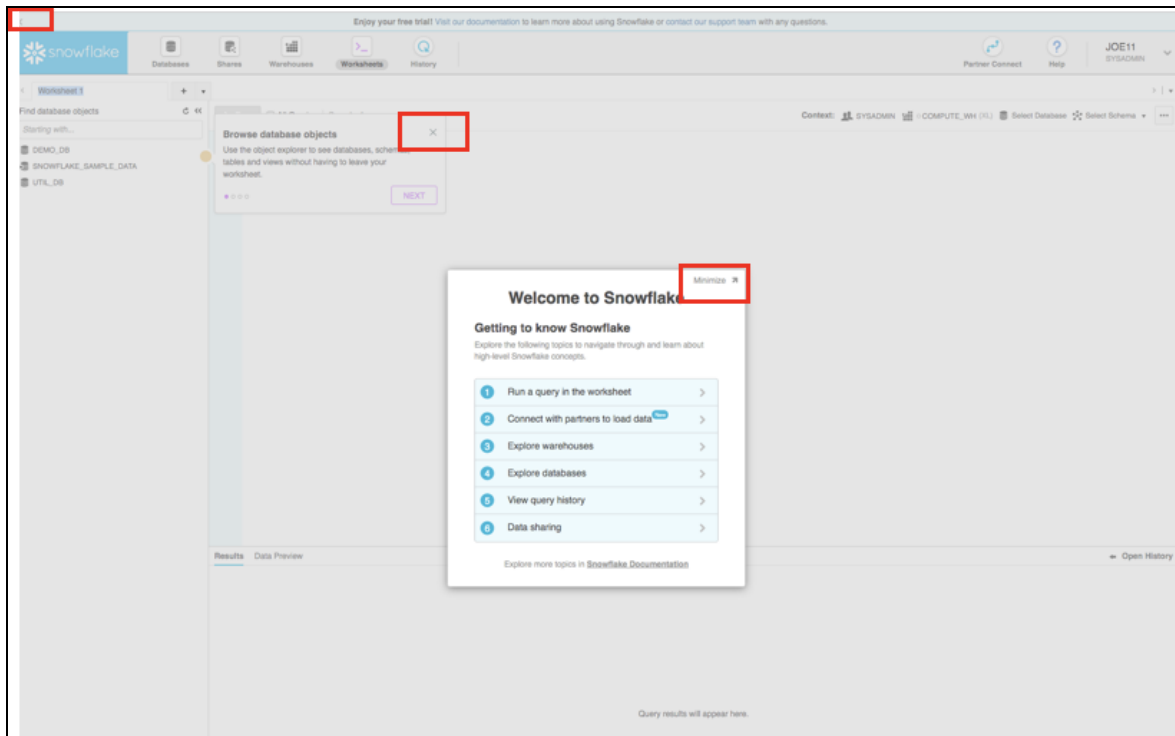
## 2.1 Logging Into the Snowflake User Interface (UI)

- 2.1.1 Open a browser window and enter the URL of your the Snowflake 30-day trial environment.
- 2.1.2 You should see the login screen below. Enter your unique credentials to log in.



## 2.2 Close any Welcome Boxes and Tutorials

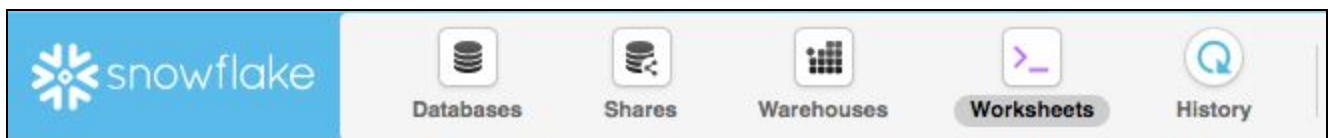
- 2.2.1 You may see “welcome” and “helper” boxes in the UI when you log in for the first time. Also a “Enjoy your free trial...” ribbon at the top of the UI. Minimize and close them by clicking on the items in the red boxes on screenshot below.



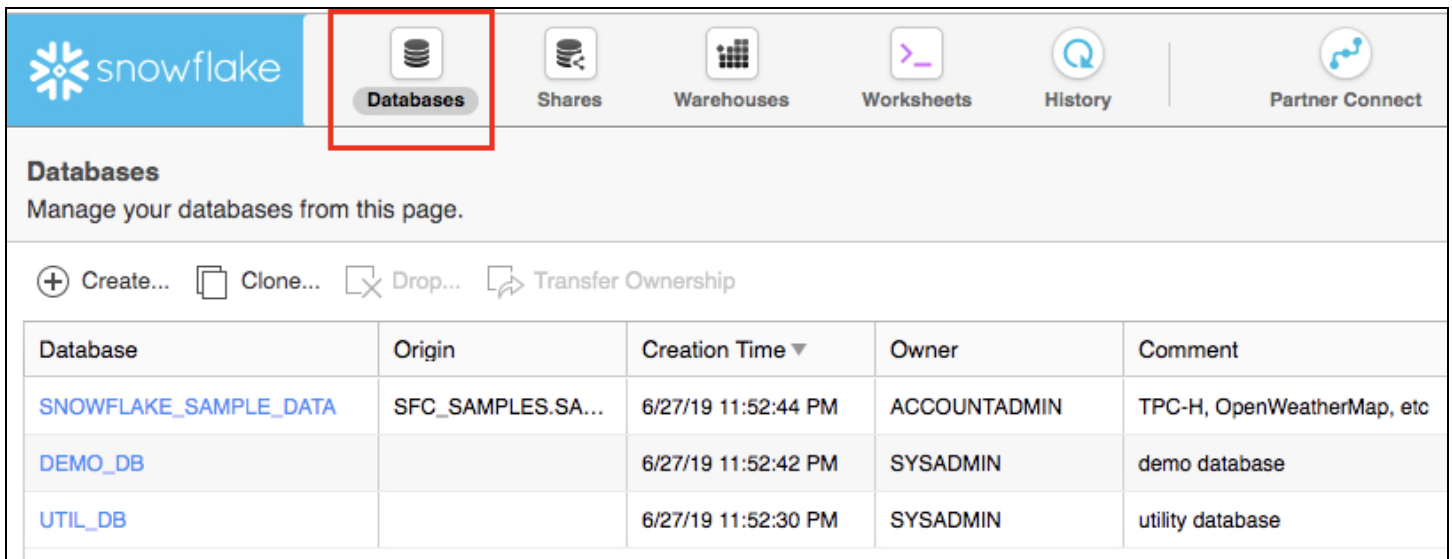
## 2.3 Navigating the Snowflake UI

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.

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



2.3.2 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.



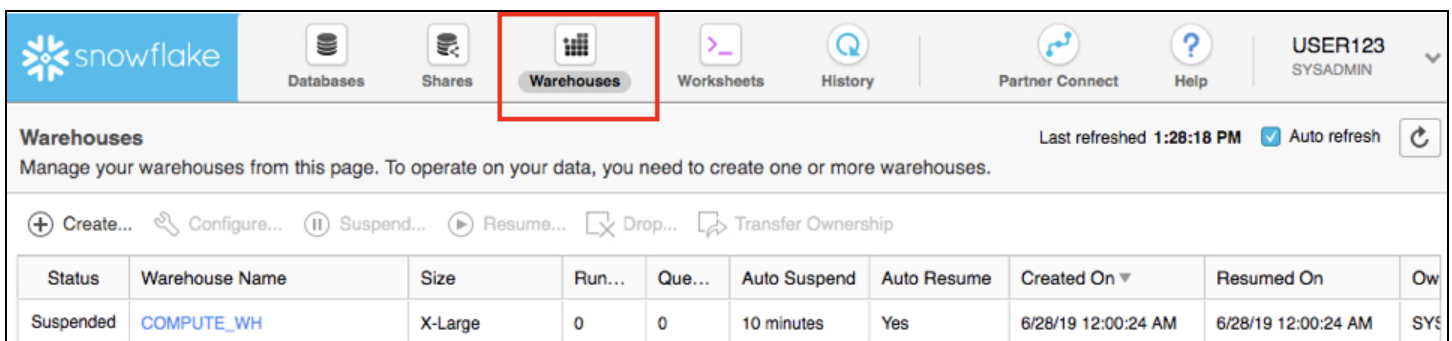
**Databases**  
Manage your databases from this page.

(+) Create... 
 (Clone) Clone... 
 (Drop) Drop... 
 (Transfer Ownership) Transfer Ownership

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

2.3.3 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. At the end of this lab is a module on data sharing.

2.3.4 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.



**Warehouses**  
Manage your warehouses from this page. To operate on your data, you need to create one or more warehouses.

(+) Create... 
 (Configure) Configure... 
 (Suspend) Suspend... 
 (Resume) Resume... 
 (Drop) Drop... 
 (Transfer Ownership) Transfer Ownership

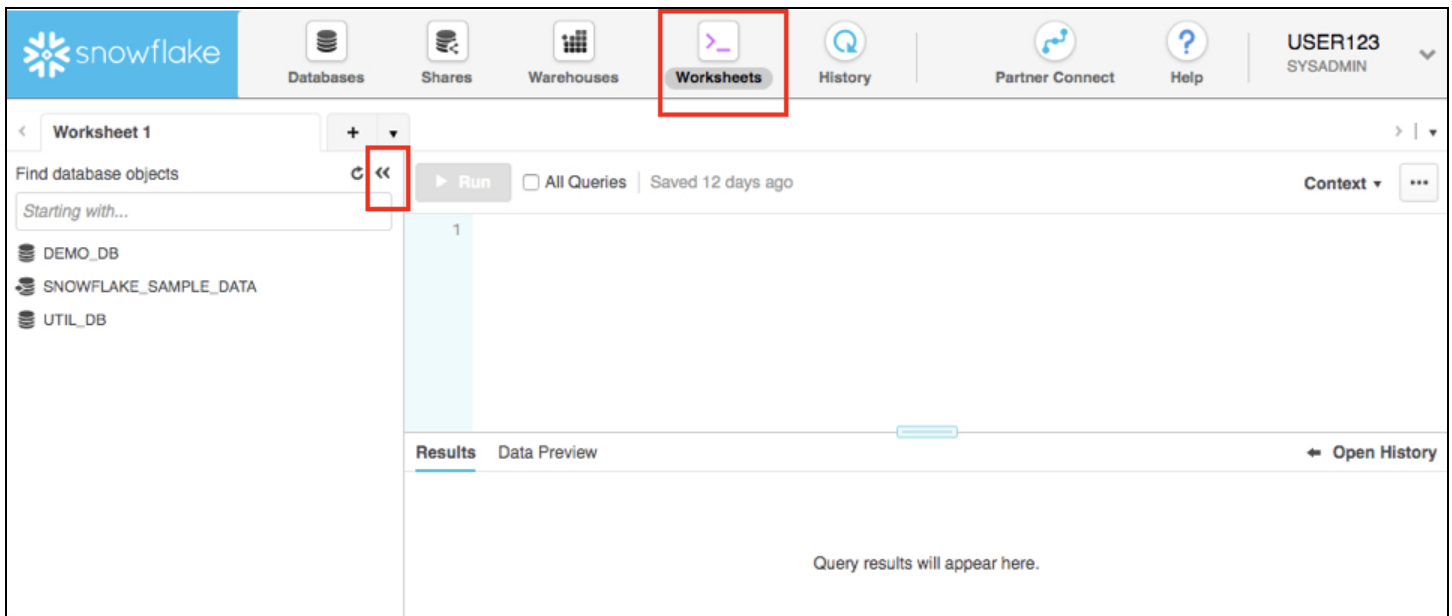
Last refreshed 1:28:18 PM ☒ Auto refresh

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

2.3.5 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.

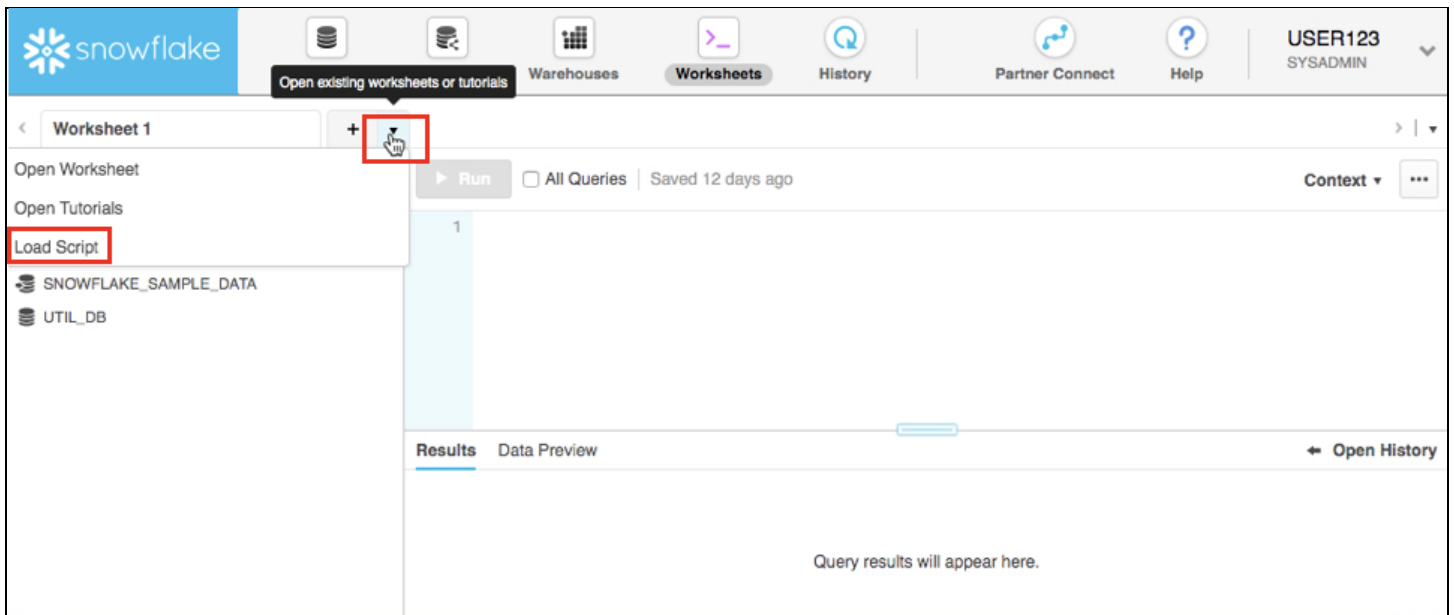
In the left pane is the database objects browser which enables users to explore all databases, schemas, tables, and views accessible by the role selected for a worksheet. The bottom pane shows results of queries and operations.

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.





- 2.3.6 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 “sagemaker\_workshop\_v1.4.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**



Much 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.

- 2.3.7 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).

History

Last refreshed 1:43:13 PM ☐ Auto refresh

Hide Filters View SQL Abort...

Display queries that meet all of the following criteria:

Status  is Select query status

☐ Include client-generated statements

☐ Include queries executed by user tasks

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

Search stopped at 1:08:12 PM

Continue search...

2.3.8 If you click on the top right of the UI where your user name 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.

History

Last refreshed 1:43:13 PM

Hide Filters View SQL Abort...

Display queries that meet all of the following criteria:

USER123 SYSADMIN

- Change Password
- Switch Role
- Preferences
- Log Out

Clear filters

## **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>

## Module 3: Preparing to Load Data & Loading Data

Let's start by preparing to load the structured data on Customer Churn 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 and internal 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.

The dataset we use is publicly available and was mentioned in the book [Discovering Knowledge in Data](#) by Daniel T. Larose. It is attributed by the author to the University of California Irvine Repository of Machine Learning Datasets. The data has been exported and pre-staged for you in an Amazon AWS S3 bucket in the US-WEST region. The data consists of information from mobile operators historical records on which customers ultimately ended up churning and which continued using the service.

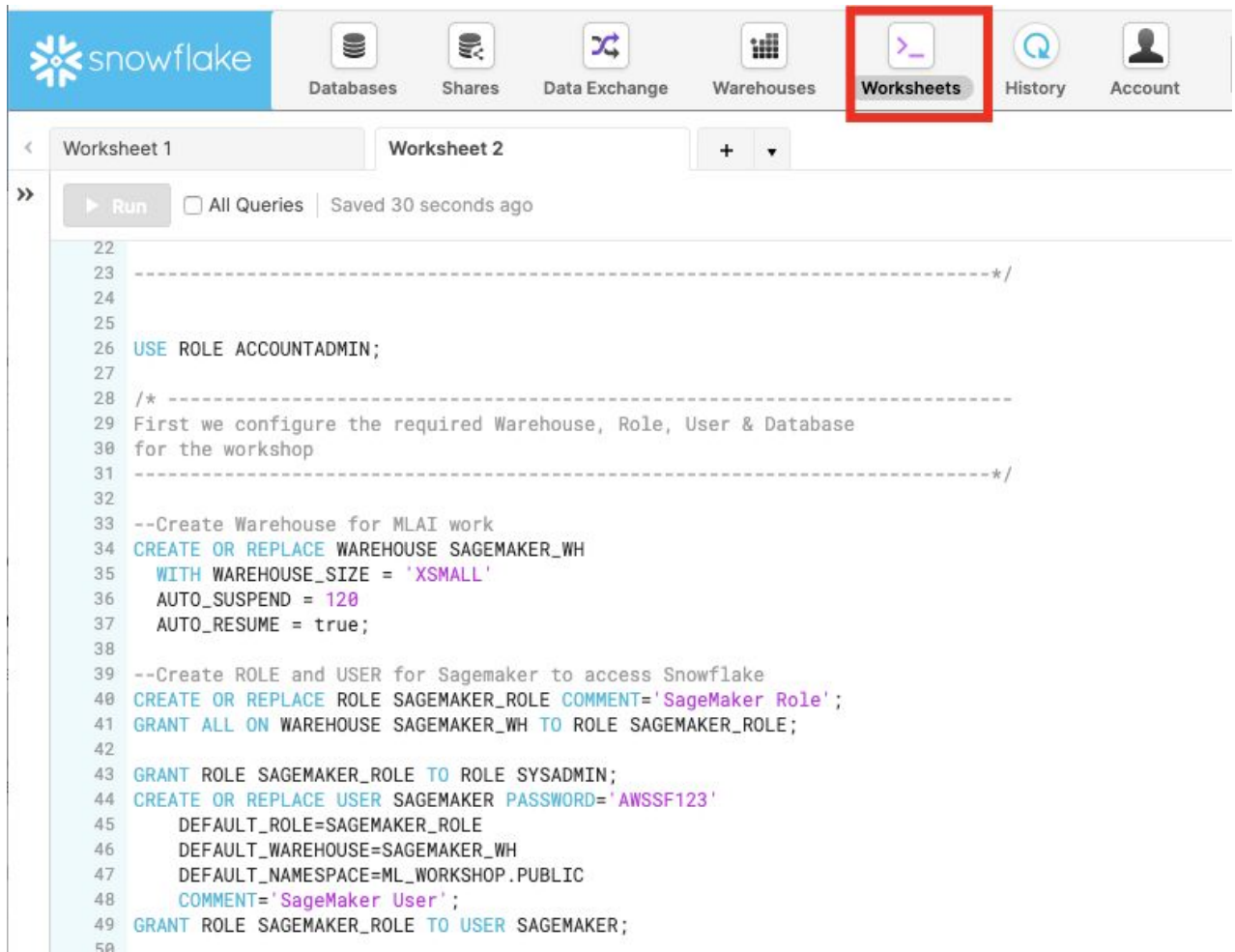
Below is a snippet from one of the Customer Churn CSV data files:

```
1,KS,128,415,382-4657,no,yes,
25,265.1,110,45.07,197.4,99,16.78,244.7,91,11.01,10,3,2.7,1,False.
2,OH,107,415,371-7191,no,yes,
26,161.6,123,27.47,195.5,103,16.62,254.4,103,11.45,13,7,3,3.7,1,False.
3,NJ,137,415,358-1921,no,no,
0,243.4,114,41.38,121.2,110,10.3,162.6,104,7.32,12.2,5,3.29,0,False.
4,OH,84,408,375-9999,yes,no,0,299.4,71,50.9,61.9,88,5.26,196.9,89,8.86,6.6,7,1.78,2,False.
5,MA,121,510,355-9993,no,yes,
24,218.2,88,37.09,348.5,108,29.62,212.6,118,9.57,7.5,7,2.03,3,False.
6,MO,147,415,329-9001,yes,no,
0,157,79,26.69,103.1,94,8.76,211.8,96,9.53,7.1,6,1.92,0,False.
```

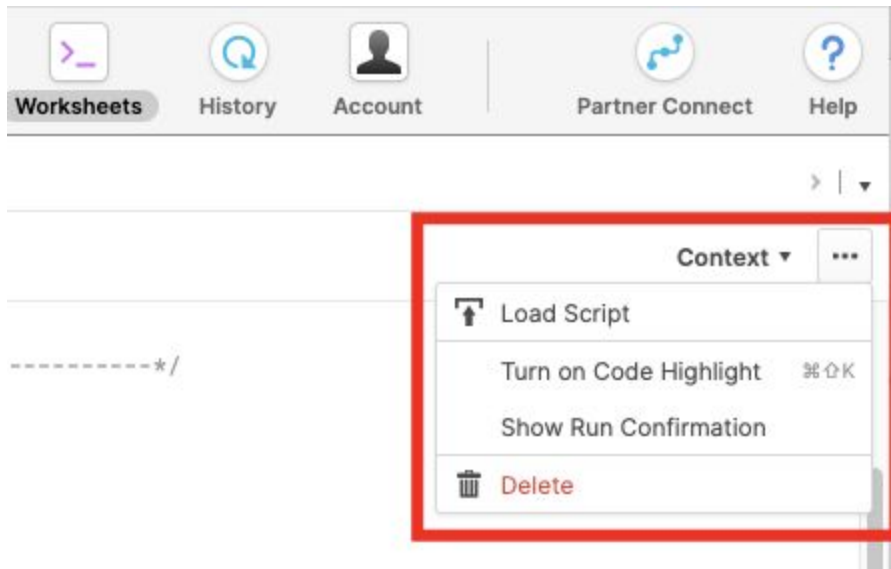
It is in comma-delimited format with no header line. This will come into play later in this module as we configure the Snowflake table which will store this data.

## 3.1 Start using Worksheets and Create a Virtual Warehouse

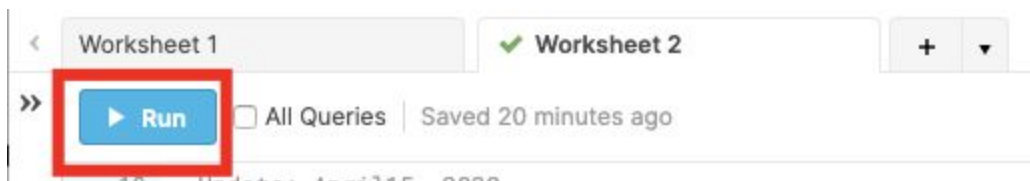
3.1.1 At the top of the Snowflake UI, click the Worksheets tab. You should see the worksheet with all the SQL we loaded in a prior step.



3.1.2 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.



- 3.1.3 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.

- 3.1.4 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;**

- 3.1.5 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 OR REPLACE WAREHOUSE SAGEMAKER\_WH  
WITH WAREHOUSE\_SIZE = 'XSMALL'  
AUTO\_SUSPEND = 120  
AUTO\_RESUME = true;**

## 3.2 Create a Role and User

- 3.2.1 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;
```

- 3.2.2 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;

USE ROLE SYSADMIN;
```

### 3.3 Create a Database and Tables

- 3.3.1 First, let's create a database called ML\_WORKSHOP that will be used for loading the customer churn data. 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.

```
CREATE DATABASE IF NOT EXISTS ML_WORKSHOP;
```

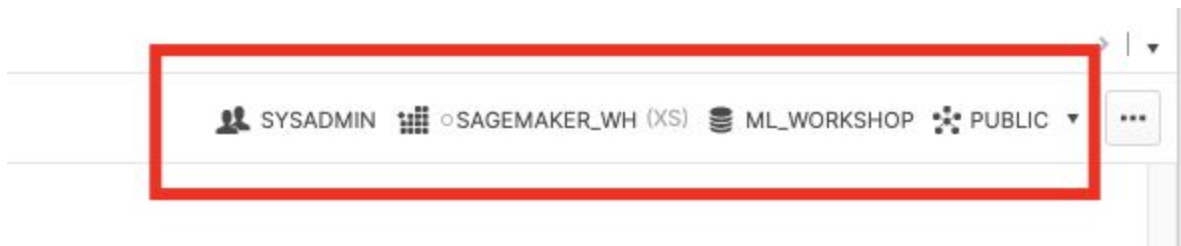
```
GRANT USAGE ON DATABASE ML_WORKSHOP TO ROLE SAGEMAKER_ROLE;
```

```
GRANT ALL ON SCHEMA ML_WORKSHOP.PUBLIC TO ROLE SAGEMAKER_ROLE;
```

- 3.3.2 Let's now switch context to the ML\_WOKRSHOP 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 ML_WORKSHOP.PUBLIC;
```

```
USE WAREHOUSE SAGEMAKER_WH;
```



3.3.3 Next we will create the CUSTOMER\_CHURN table to load customer churn data from a S3 bucket. We will also create a table, ML\_RESULTS, to load the predictions from SageMaker at a later stage back into Snowflake. For both tables all privileges will be granted to the SAGEMAKER\_ROLE as well.

```
CREATE OR REPLACE TABLE CUSTOMER_CHURN (  
    Cust_ID INT,  
    State varchar(10),  
    Account_Length INT,  
    Area_Code INT,  
    Phone varchar(10),  
    Intl_Plan varchar(10),  
    VMail_Plan varchar(10),  
    VMail_Message INT,  
    Day_Mins FLOAT,  
    Day_Calls INT,  
    Day_Charge FLOAT,  
    Eve_Mins FLOAT,  
    Eve_Calls INT,  
    Eve_Charge FLOAT,  
    Night_Mins FLOAT,  
    Night_Calls INT,  
    Night_Charge FLOAT,  
    Intl_Mins FLOAT,  
    Intl_Calls INT,  
    Intl_Charge FLOAT,  
    CustServ_Calls INT,  
    Churn varchar(10)  
);
```

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

```
CREATE OR REPLACE TABLE ML_RESULTS (  
    Churn_IN INT,  
    Cust_ID INT,  
    Churn_Score REAL  
);
```

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





#### Many Options to Run Commands.

SQL commands can be executed through the UI (limited), via the Worksheets tab, using our SnowSQL command line tool, a SQL editor of your choice via ODBC/JDBC, or through our Python or Spark connectors.

As mentioned earlier, in this lab we will run operations via pre-written SQL in the worksheet (as opposed to using the UI) to save time.

### 3.4 Create a File Format, an External Stage and Internal 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.

We will also create an Internal Stage that will be used to load the SageMaker predictions back to a SNOWflake table for final analysis.

- 3.4.1 First we will 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 CSVHEADER
  TYPE = 'CSV'
  FIELD_DELIMITER = ','
  SKIP_HEADER = 0;
```

```
GRANT USAGE ON FILE FORMAT CSVHEADER TO ROLE SAGEMAKER_ROLE;
```

- 3.4.2 Next we will 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 CHURN_DATA
  url='s3://snowflake-corp-se-workshop/sagemaker-snowflake-devdays-v1/sourcedata/';
```

3.4.3 Let's also create the Internal Stage for convenient data loading of the SageMaker predictions later in the lab. We will also grant the necessary privileges to the SAGEMAKER\_ROLE to both the internal and external stages, note the difference in privileges granted between the internal and external stages.

```
CREATE OR REPLACE STAGE ML_RESULTS  
FILE_FORMAT = (TYPE = CSV);
```

```
GRANT READ, WRITE ON STAGE ML_RESULTS TO ROLE SAGEMAKER_ROLE;
```

```
GRANT USAGE ON STAGE CHURN_DATA TO ROLE SAGEMAKER_ROLE;
```

NOTE - You can look at the data in an external stage before loading it. It is not part of this lab, but you can look and run the SQL commands that are commented out in the SQL Script file.

3.4.4 Finally let's load the Customer Churn data from the S3 bucket into Snowflake.

First we switch to the SAGEMAKER\_ROLE.

```
USE ROLE SAGEMAKER_ROLE;
```

Then we will load the data using the COPY command.

```
COPY INTO CUSTOMER_CHURN FROM @CHURN_DATA/ FILE_FORMAT =  
(FORMAT_NAME = CSVHEADER);
```

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

```
SELECT * FROM CUSTOMER_CHURN LIMIT 10;
```

The results are displayed in the frame below the Worksheet.

Results

Data Preview

Open History

Query ID

SQL

1.21s

10 rows

Filter result...

Copy

Columns

Row	CUST_ID	STATE	ACCOUNT_LENGTH	AREA_CODE	PHONE	INTL_PLAN	VMAIL_PLAN	VMAIL_MESSAGES	DAY_MINS
1	1	KS	128	415	382-4657	no	yes	25	265.1
2	2	OH	107	415	371-7191	no	yes	26	161.6
3	3	NJ	137	415	358-1921	no	no	0	243.4
4	4	OH	84	408	375-9999	yes	no	0	299.4
5	5	MA	121	510	355-9993	no	yes	24	218.2

We can also do some quick analysis on the data in the table by querying the average voicemail messages by state for customers with a voicemail plan.

```
SELECT STATE, INTL_PLAN, AVG(VMAIL_MESSAGE)
FROM CUSTOMER_CHURN
WHERE VMAIL_PLAN = 'yes'
GROUP BY 1,2 ORDER BY AVG(VMAIL_MESSAGE) DESC;
```

Results

Data Preview

Open History

✓

Query ID

SQL

872ms

94 rows

Filter result...

Download

Copy

Columns ▾

Row	STATE	INTL_PLAN	AVG(VMAIL_MESSAGE)
1	AZ	yes	45.000000
2	SC	yes	39.000000
3	DC	yes	39.000000
4	AR	yes	38.000000
5	VT	yes	37.000000
6	NC	yes	37.000000

## Module 4: Deploy an Amazon SageMaker Notebook

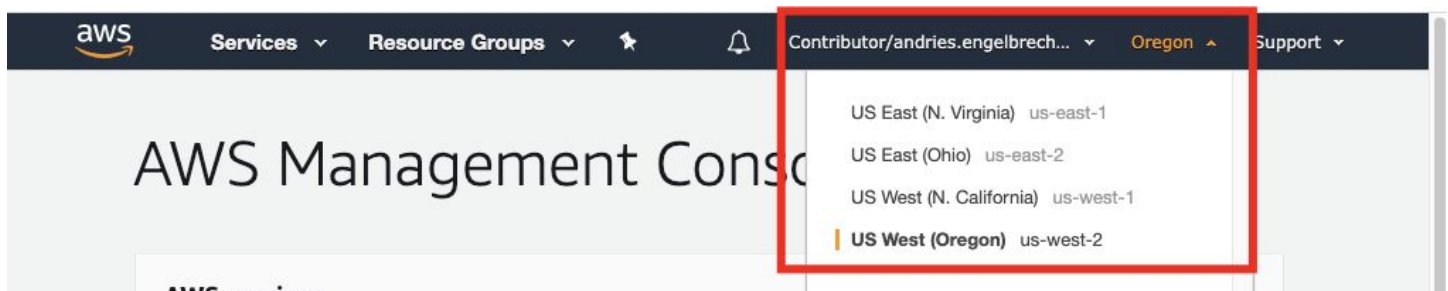
For this module, we will login to the AWS Management Console and deploy a SageMaker Notebook using a CloudFormation Template. We will then upload the workshop Notebook used for this lab.

### 4.1 Deploy the CloudFormation Template

The CloudFormation Template (CFT) will create a SageMaker Notebook Instance and also create a S3 bucket that will be used for the lab. The CFT will also install the Snowflake Python Connector on the Instance for our convenience.

Open another window in your browser and log in to the AWS Management Console.

- 4.1.1 In the Management Console select the region, preferably the same region as the Snowflake account you are using. In this case us-west-2 (Oregon) region is used.



- 4.1.2 The CloudFormation Template (CFT) can now be deployed by clicking on the link below (or right click on the URL below, copy link and then paste it in a new browser tab).

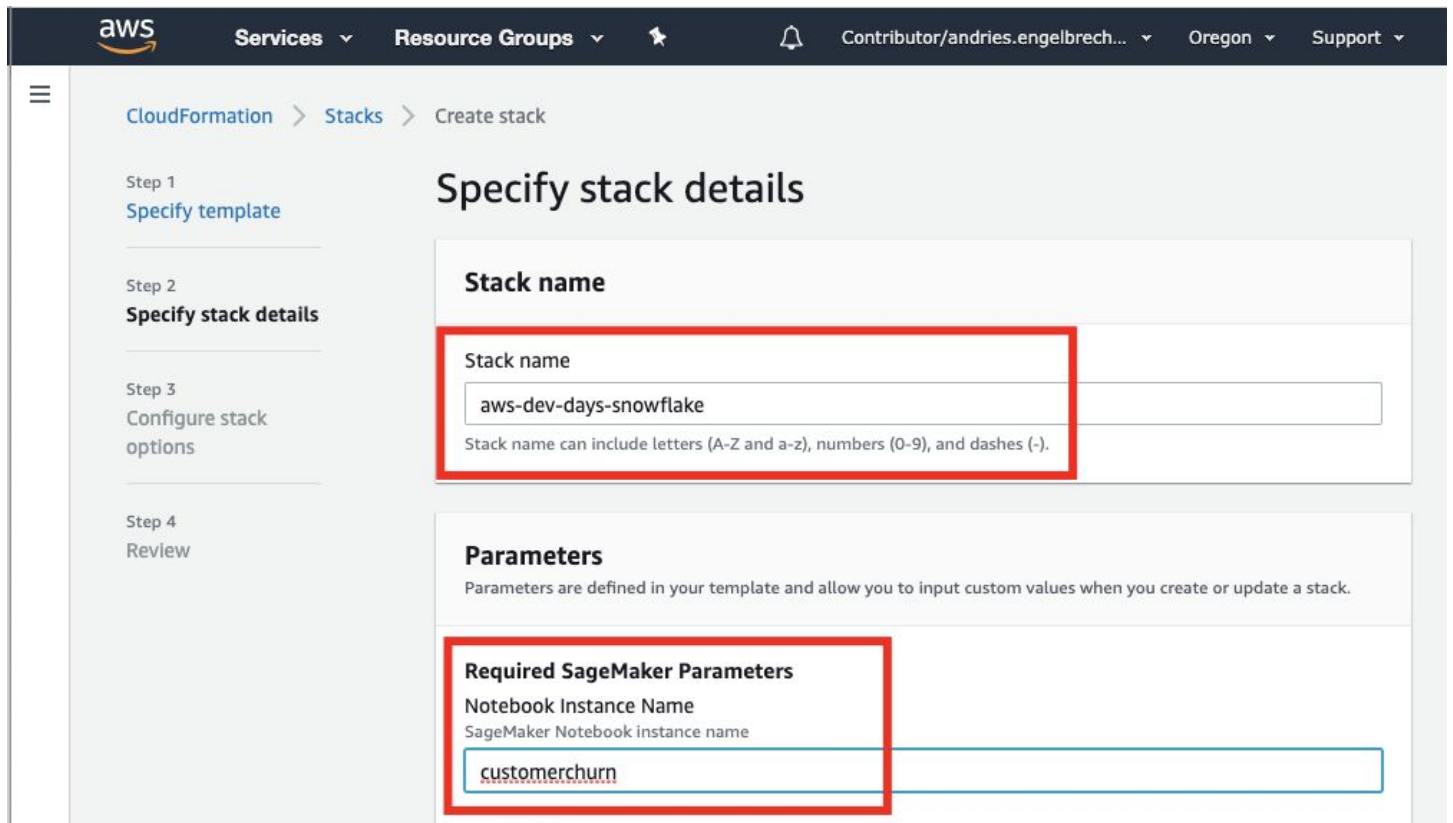
#### [Deploy SageMaker Notebook](https://console.aws.amazon.com/cloudformation/home?#/stacks/new?stackName=snowflake-notebook&templateURL=https://snowflake-corp-se-workshop.s3-us-west-1.amazonaws.com/sagemaker-snowflake-devdays-v1/sagemaker/snowflake-sagemaker-notebook-v1.1.yaml)

<https://console.aws.amazon.com/cloudformation/home?#/stacks/new?stackName=snowflake-notebook&templateURL=https://snowflake-corp-se-workshop.s3-us-west-1.amazonaws.com/sagemaker-snowflake-devdays-v1/sagemaker/snowflake-sagemaker-notebook-v1.1.yaml>

- 4.1.3 By clicking the link a new tab on your browser should open and you should see the Create Stack. Before proceeding verify the AWS region by looking at the top right hand bar, similar to the step above.

The screenshot shows the AWS CloudFormation console's 'Create stack' wizard. The top navigation bar includes the AWS logo, 'Services', 'Resource Groups', a star icon, a notification bell, the user 'Contributor/andries.engelbrech...', the region 'Oregon', and a 'Support' link. The breadcrumb trail is 'CloudFormation > Stacks > Create stack'. On the left, a sidebar lists four steps: 'Step 1 Specify template' (active), 'Step 2 Specify stack details', 'Step 3 Configure stack options', and 'Step 4 Review'. The main content area is titled 'Create stack' and is divided into two sections. The first section, 'Prerequisite - Prepare template', explains that every stack is based on a template (JSON or YAML) and provides three radio button options: 'Template is ready' (selected), 'Use a sample template', and 'Create template in Designer'. The second section, 'Specify template', explains that a template is a JSON or YAML file describing stack resources and properties. It has a 'Template source' section with two radio button options: 'Amazon S3 URL' (selected) and 'Upload a template file'. Below this is a text input field for the 'Amazon S3 URL' containing the URL 'https://snowflake-corp-se-workshop.s3-us-west-1.amazonaws.com/sagemaker-snowflake-devdays-'. A label 'Amazon S3 template URL' is positioned below the input field. At the bottom of this section, the full 'S3 URL' is displayed: 'https://snowflake-corp-se-workshop.s3-us-west-1.amazonaws.com/sagemaker-snowflake-devdays-v1/sagemaker/snowflake-sagemaker-notebook-v1.1.yaml', with a 'View in Designer' button to its right. At the bottom right of the entire form, there are 'Cancel' and 'Next' buttons.

- 4.1.4 Click the NEXT button on the bottom right. On the following page you will need to assign a unique name for the CFT Stack Name and the Notebook Instance Name. All the other parameters can be left to the default settings for this lab. Scroll down and click the NEXT button again.



aws Services Resource Groups Contributor/andries.engelbrech... Oregon Support

CloudFormation > Stacks > Create stack

Step 1  
Specify template

Step 2  
Specify stack details

Step 3  
Configure stack options

Step 4  
Review

## Specify stack details

**Stack name**

Stack name

aws-dev-days-snowflake

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

**Parameters**

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

**Required SageMaker Parameters**

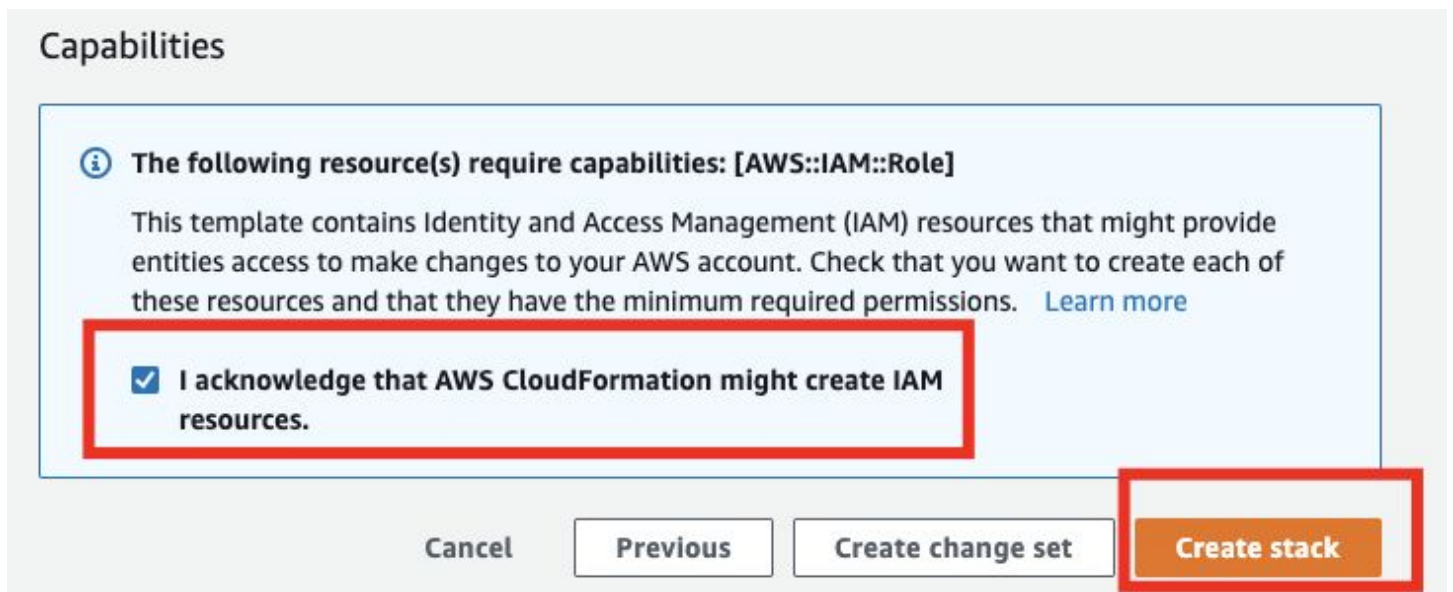
Notebook Instance Name

SageMaker Notebook instance name

customerchurn

4.1.5 For steps 3 you can leave the defaults in place and just click NEXT. Step 4 will review all the settings. Scroll down and verify Notebook Instance Name. At the bottom of the page check the acknowledgement that IAM resources may be created and click on the Create stack button.

Note: It can take a few minutes to deploy the SageMaker Notebook.



## Capabilities

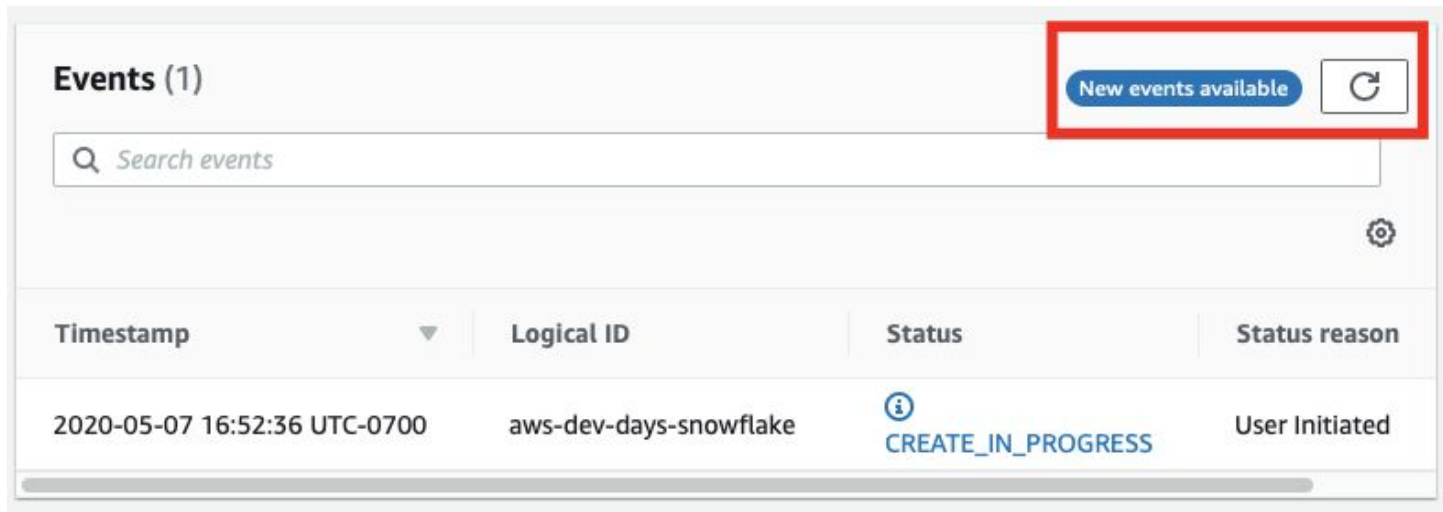
**The following resource(s) require capabilities: [AWS::IAM::Role]**

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. [Learn more](#)

☒ I acknowledge that AWS CloudFormation might create IAM resources.

Cancel Previous Create change set **Create stack**

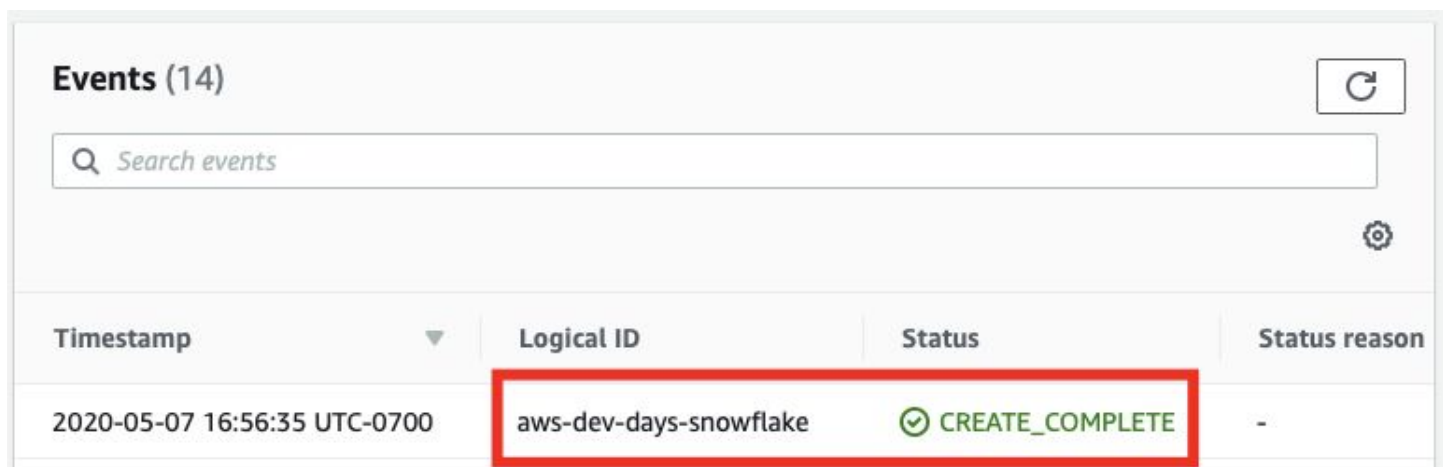
4.1.6 You can refresh the Events page by clicking the circular arrow on the top right hand side.



The screenshot shows the AWS CloudFormation 'Events' page. At the top right, a red box highlights a 'New events available' button and a circular refresh icon. Below this is a search bar labeled 'Search events'. The main content is a table with the following data:

Timestamp	Logical ID	Status	Status reason
2020-05-07 16:52:36 UTC-0700	aws-dev-days-snowflake	CREATE_IN_PROGRESS	User Initiated

4.1.7 Once the Stack has completed you will see the Stack Name and a Status of CREATE\_COMPLETE.



The screenshot shows the AWS CloudFormation 'Events' page with 14 events. A red box highlights the following row in the table:

Timestamp	Logical ID	Status	Status reason
2020-05-07 16:56:35 UTC-0700	aws-dev-days-snowflake	CREATE_COMPLETE	-

4.1.8 In the CloudFormation page click on the Outputs tab. It will list the SageMaker Notebook ARN value as well as the S3 Bucket's name that was created. Note the S3 bucket name and the SageMaker Notebook Instance name for later in the lab.

# aws-dev-days-snowflake

DeleteUpdateStack actions ▼Create stack ▼

[Stack info](#)[Events](#)[Resources](#)[Outputs](#)[Parameters](#)[Template](#)[Change s](#)

## Outputs (2)

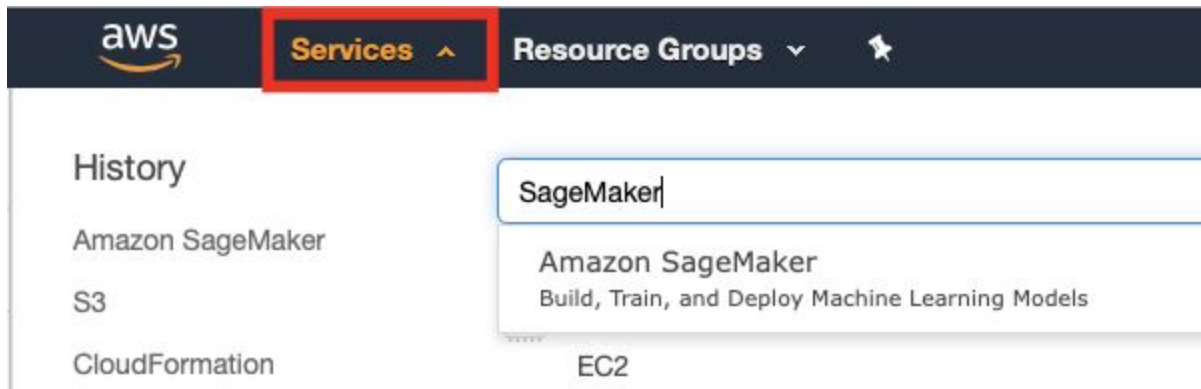
Key ▲	Value ▼	Description ▼
NotebookARN	arn:aws:sagemaker:us-west-2: [REDACTED]:notebook-instance/customerchurn	Snowflake SageMaker Notebook ARN
S3BucketName	snowflake-sagemaker-workshop- [REDACTED]	Name of your S3 bucket to use for the workshop

## 4.2 Upload the Notebook to the SageMaker instance

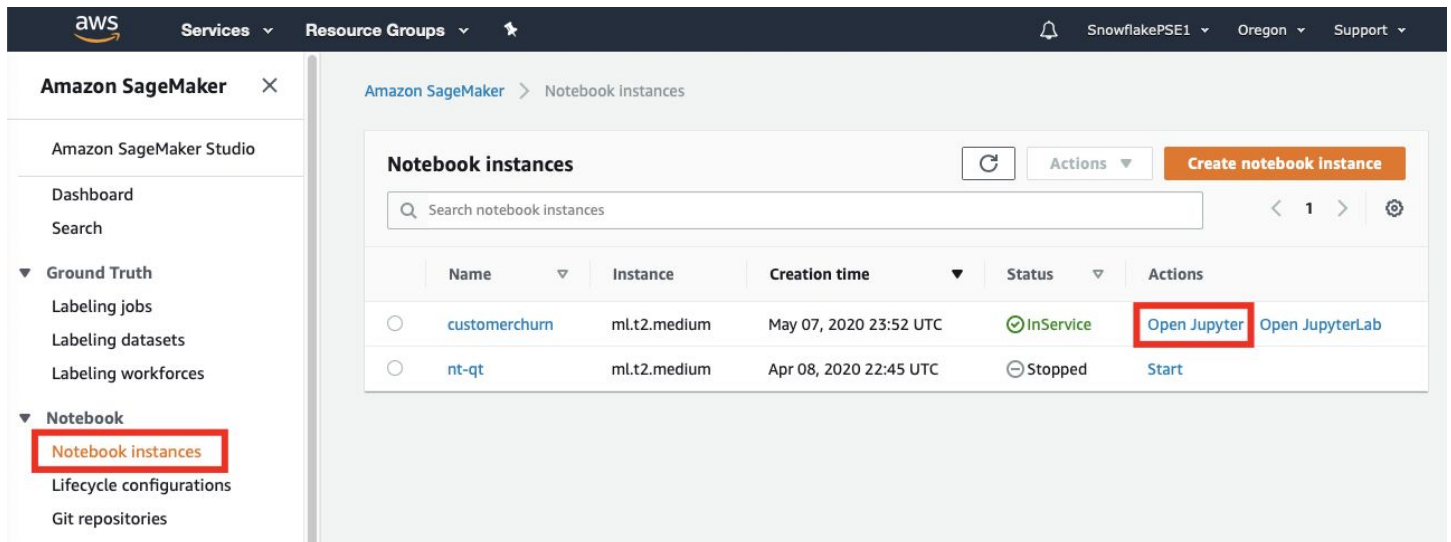
Now we can upload the notebook that was downloaded for the lab to SageMaker.

- 4.2.1 In the AWS console navigate to the SageMaker service. The easiest way to do it is by clicking on the Services menu at the top left and then typing SageMaker in the search bar. Click on Amazon SageMaker once it appears below the search bar.

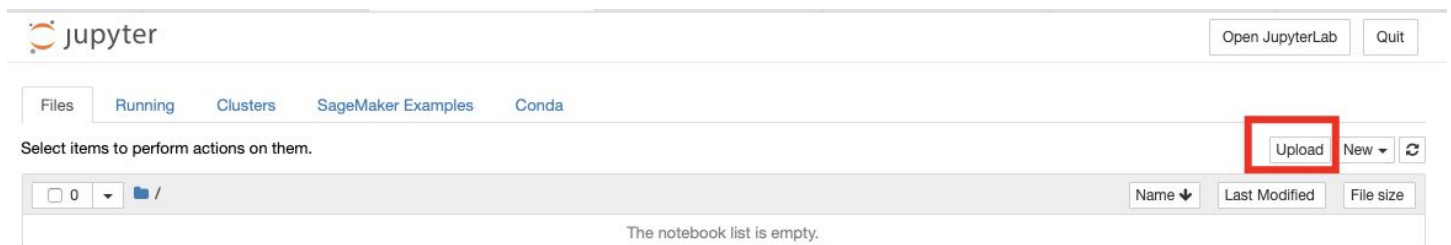


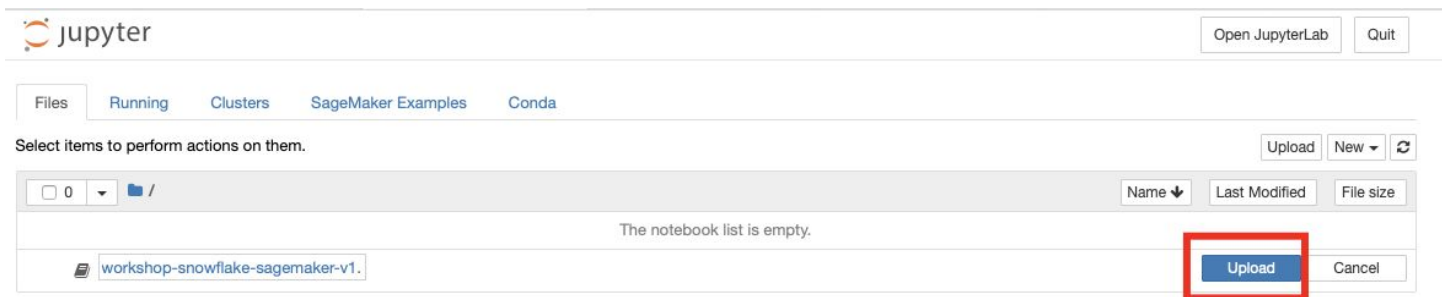


4.2.2 Once in the SageMaker console, click on the Notebook Instance menu on the left hand side. Then click on the Open Jupyter link next to the Notebook Instance you specified in the CFT.



4.2.3 This will open a new tab in your browser. On the top right hand corner you will see an Upload button. Click it and select the notebook file (.ipynb) that you downloaded to your computer at the start of the lab. To complete uploading it click the blue highlighted upload button next to the filename.





4.2.4 The notebook is now uploaded and ready to use.

## Module 5: Machine Learning Workflow in SageMaker

In this module we will perform a basic Machine Learning workflow in a SageMaker Notebook. We will be performing the following steps:

- Query data from Snowflake
- Explore and visualize the data
- Do feature selection
- Train a model
- Compile a model
- Run batch inference using SageMaker using the model
- Upload the model's Churn Score predictions to Snowflake

**Note:** The steps and process is very well documented in the Notebook itself with various comments. We encourage you to carefully read through each section of the Notebook.

### 5.1 Open the Notebook

5.1.1 After we uploaded the notebook file to SageMaker we can now open it by simply clicking on the file name.




5.1.2 Once you click on the notebook link it will open a new tab in your browser and provide you with Jupyter notebook interface seen below.

Jupyter











workshop-snowflake-sagemaker-v1.1

Last Checkpoint: 14 minutes ago (autosaved)





FileEditViewInsertCellKernelWidgetsHelp

Trustedconda\_python3



Markdown



## Churn Predictive Analytics using Amazon SageMaker and Snowflake

### Background

The purpose of this lab is to demonstrate the basics of building an advanced analytics solution using Amazon SageMaker on data stored in Snowflake. In this notebook we will create a customer churn analytics solution by training an XGBoost churn model, and batching churn prediction scores into a data warehouse.

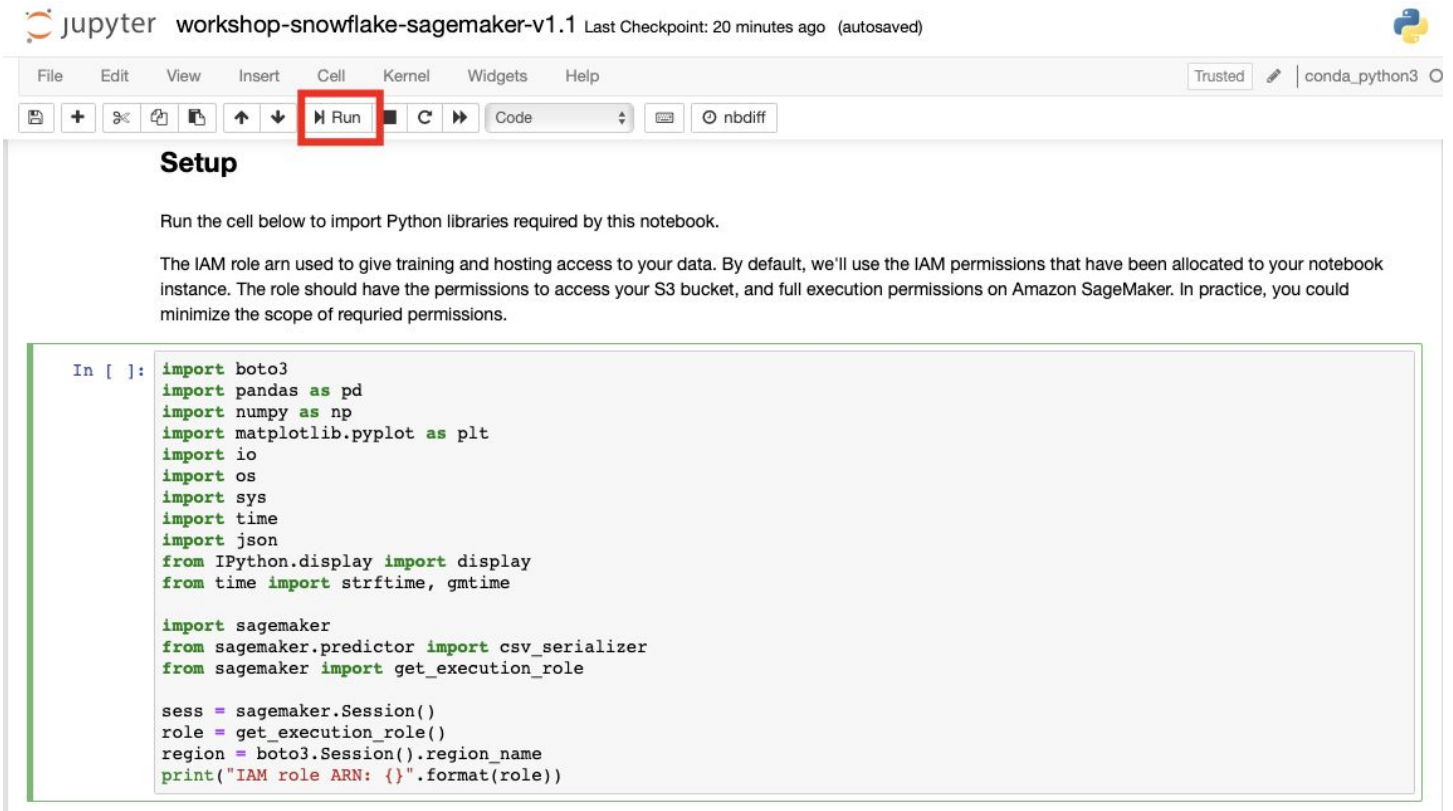
(Need to update) This notebook extends one of the example tutorial notebooks: [Customer Churn Prediction with XGBoost](#). The extended learning objectives are highlighted in bold below.

## 5.2 Execute the Machine Learning workflow in the Notebook

The notebook consists of comment sections and code sections. To execute a code section click on it to highlight it and then click on the Run button on the toolbar. *An alternative to clicking the Run button is to use Shift+Return or Enter keyboard shortcuts.*

We will execute one code block at a time and make sure to work through the comments to describe what each step is doing. Do not click the run button yet, but get familiar with the interface.

### 5.2.1 Read through the comments and scroll down to the first code block below the **Setup** section. Click on the code block to highlight it and then click on Run.



**Setup**

Run the cell below to import Python libraries required by this notebook.

The IAM role arn used to give training and hosting access to your data. By default, we'll use the IAM permissions that have been allocated to your notebook instance. The role should have the permissions to access your S3 bucket, and full execution permissions on Amazon SageMaker. In practice, you could minimize the scope of required permissions.

```
In [ ]: import boto3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import io
import os
import sys
import time
import json
from IPython.display import display
from time import strftime, gmtime

import sagemaker
from sagemaker.predictor import csv_serializer
from sagemaker import get_execution_role

sess = sagemaker.Session()
role = get_execution_role()
region = boto3.Session().region_name
print("IAM role ARN: {}".format(role))
```

The setup code block loads a number of libraries that will be used, as well as checks the IAM role being used.

Once a code block has executed it will put a number in the square brackets next to it and an output below it, as seen in the example below.

```
In [1]: import boto3
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import io
import os
import sys
import time
import json
from IPython.display import display
from time import strftime, gmtime

import sagemaker
from sagemaker.predictor import csv_serializer
from sagemaker import get_execution_role

sess = sagemaker.Session()
role = get_execution_role()
region = boto3.Session().region_name
print("IAM role ARN: {}".format(role))

IAM role ARN: arn:aws:iam::[REDACTED]:role/service-role/aws-dev-days-snowflake-ExecutionRole-PDO3Z1KANZP5
```

5.2.2 Next, let's configure the notebook to use the S3 bucket that was created by the CFT and the name that was captured from step 4.1.8. Replace the <REPLACE WITH YOUR BUCKET NAME> text with the name of your S3 bucket. **Do note that your bucket name will be different from the one listed in the examples.** Also remember to click the Run button after selecting the code box.

Now let's set the S3 bucket and prefix that you want to use for training and model data. This bucket should be created within the same region as the Notebook Instance, training, and hosting.

- Replace <<'REPLACE WITH YOUR BUCKET NAME'>> with the name of your bucket.

```
In [ ]: bucket = 'snowflake-sagemaker-workshop-[REDACTED]'
prefix = 'churn-analytics-lab'
```

5.2.3 After reading the description of the data set in the Data comment section we will configure the connection to Snowflake in the Notebook.

For the user and password you have to use the username and password you configured in your Snowflake account earlier in section 3.2.2. There is no need to change if you used the defaults provided in the SQL script.

The **Snowflake Account Name** can be derived from the URL, in your browser address bar, you use to login to your Snowflake account i.e.

<https://ab12345.snowflakecomputing.com> then your account name will be **ab12345** and that is all you will need to identify it.

Snowflake is deployed in various AWS regions. For the US West region you only need the account name itself. For other regions you will see a region identifier in the Snowflake URL, as an example in the US East region the URL i.e.

<https://cd123456.us-east-1.snowflakecomputing.com> . In this case you will need to identify your account with the account name and region identifier - [cd123456.us-east-1](#)

For this example we will use a fictitious account identifier in the US East region. Replace <ACCOUNT> with your SNOWflake account name and region identifier.

Provide the connection and credentials required to connect to your Snowflake account. You'll need to modify the cell below with the appropriate **ACCOUNT** for your Snowflake trial. If you followed the lab guide instructions, the username and password below will work.

**NOTE:** For Snowflake accounts in regions other than US WEST add the Region ID after a period . i.e. XYZ123456.US-EAST-1.

In practice, security standards might prohibit you from providing credentials in clear text. As a best practice in production, you should utilize a service like [AWS Secrets Manager](#) to manage your database credentials.

```
In [ ]: import snowflake.connector
        # Connecting to Snowflake using the default authenticator
        ctx = snowflake.connector.connect(
            user='sagemaker',
            password='AWSSF123',
            account='cd123456.us-east-1',
            warehouse='SAGEMAKER_WH',
            database='ML_WORKSHOP',
            schema='PUBLIC'
        )
```

5.2.4 We will work through each of the comment and code blocks using the detailed comments in the Notebook. There are a number of well documented steps and not all of them will be covered in detail in the lab guide.

The following steps will be performed:

- Query the data in Snowflake and loading it into a pandas dataframe
- Exploring and visualization of the data
- Feature selection
- Splitting the data set into training, testing and validation data sets
- The S3 bucket will be used to store the data sets



**5.2.5 Training** the model using XGBoost. In this step we'll specify the locations of the XGBoost algorithm container, the S3 locations of the training and validation data sets. Then we will deploy a training instance using SageMaker to train the model. This step typically will take a few minutes to deploy the instance and train the model, but is fully managed by SageMaker. The model is saved in the S3 location specified in the code block.

```
In [44]: xgb = sagemaker.estimator.Estimator(xgb_training_container,
                                             role,
                                             train_instance_count=1,
                                             train_instance_type='ml.m5.xlarge',
                                             output_path='s3://{}/{}/output'.format(bucket, prefix),
                                             sagemaker_session=sess)

xgb.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        silent=0,
                        objective='binary:logistic',
                        num_round=100)

xgb.fit({'train': s3_input_train, 'validation': s3_input_validation})
```

```
[86]#011train-error:0.030433#011validation-error:0.057057
[87]#011train-error:0.029147#011validation-error:0.055556
[88]#011train-error:0.029147#011validation-error:0.055556
[89]#011train-error:0.029147#011validation-error:0.055556
[90]#011train-error:0.029147#011validation-error:0.055556
[91]#011train-error:0.029147#011validation-error:0.055556
[92]#011train-error:0.028718#011validation-error:0.054054
[93]#011train-error:0.028718#011validation-error:0.054054
[94]#011train-error:0.028718#011validation-error:0.054054
[95]#011train-error:0.028718#011validation-error:0.054054
[96]#011train-error:0.028718#011validation-error:0.054054
[97]#011train-error:0.028718#011validation-error:0.054054
[98]#011train-error:0.028718#011validation-error:0.054054
[99]#011train-error:0.028718#011validation-error:0.054054

2020-05-08 18:53:16 Uploading - Uploading generated training model
2020-05-08 18:53:16 Completed - Training job completed
Training seconds: 48
Billable seconds: 48
```

**5.2.6** Next we will **Compile** the model for **SageMaker Neo** for better optimization to run up to twice as fast. This step takes a few seconds to complete.

### Compile

[Amazon SageMaker Neo](#) optimizes models to run up to twice as fast, with no loss in accuracy. When calling `compile_model()` function, we specify the target instance family (c5) as well as the S3 bucket to which the compiled model would be stored.

```
In [45]: compiled_model = xgb
try:
    xgb.create_model()._neo_image_account(boto3.Session().region_name)
except:
    print('Neo is not currently supported in', boto3.Session().region_name)
else:
    output_path = ''.join(xgb.output_path.split('/')[:-1])
    compiled_model = xgb.compile_model(target_instance_family='ml_c5',
                                      input_shape={'data':[1, 69]},
                                      role=role,
                                      framework='xgboost',
                                      framework_version='0.7',
                                      output_path=output_path)
    compiled_model.name = 'deployed-xgboost-customer-churn-c5'
    compiled_model.image = get_image_uri(sess.boto_region_name, 'xgboost-neo', repo_version='latest')

?...!
```



5.2.7 Now we will use the optimized model to run a **Batch Inference** job. To simplify the lab we will leverage the existing data sets that we have available. First we specify the input and output S3 locations.

## Batch Inference

Next we're going to evaluate our model by using a Batch Transform to generate churn scores in batch from our `model_data`.

First, we upload the model data to S3. SageMaker Batch Transform is designed to run asynchronously and ingest input data from S3. This differs from SageMaker's real-time inference endpoints, which receive input data from synchronous HTTP requests.

For large scale deployments the data set will be retrieved from Snowflake using SQL and an External Stage to S3.

Batch Transform is often the ideal option for advanced analytics use case for several reasons:

- Batch Transform is better optimized for throughput in comparison with real-time inference endpoints. Thus, Batch Transform is ideal for processing large volumes of data for analytics.
- Offline asynchronous processing is acceptable for most analytics use cases.
- Batch Transform is more cost efficient when real-time inference isn't necessary. You only need to pay for resources used during batch processing. There is no need to pay for ongoing resources like a hosted endpoint for real-time inference.

```
In [46]: batch_input = model_data.iloc[:,1:]
batch_input.to_csv('model.csv', header=False, index=False)
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'model/model.csv')).upload_file('model.csv')

s3uri_batch_input = 's3://{}/{}/model'.format(bucket, prefix)
print('Batch Transform input S3 uri: {}'.format(s3uri_batch_input))

s3uri_batch_output = 's3://{}/{}/out'.format(bucket, prefix)
print('Batch Transform output S3 uri: {}'.format(s3uri_batch_output))

Batch Transform input S3 uri: s3://snowflake-sagemaker-workshop-[REDACTED]/churn-analytics-lab/model
Batch Transform output S3 uri: s3://snowflake-sagemaker-workshop-[REDACTED]/churn-analytics-lab/out
```

Then we run the batch transform job. Do note that the batch transform jobs run asynchronously and non blocking by default. Run the transformer wait command in the next code block “**transformer.wait()**” to notify us when the batch job is complete. This operation can take a few minutes to complete.

```
In [47]: from sagemaker.transformer import Transformer
BATCH_INSTANCE_TYPE = 'ml.c5.xlarge'

transformer = compiled_model.transformer(instance_count=1,
                                         strategy='SingleRecord',
                                         assemble_with='Line',
                                         instance_type=BATCH_INSTANCE_TYPE,
                                         accept = 'text/csv',
                                         output_path=s3uri_batch_output)

transformer.transform(s3uri_batch_input,
                     split_type='Line',
                     content_type='text/csv',
                     input_filter = "${1:}",
                     join_source = "Input",
                     output_filter = "${0,-1,-2}")

WARNING:sagemaker:Using already existing model: deployed-xgboost-customer-churn-c5
```

Batch transform jobs run asynchronously, and are non-blocking by default. Run the command below to block until the batch job completes.

```
In [*]: transformer.wait()

....
```

You may also see an error where a line failed to parse. For the purposes of the lab, this error can be ignored.

[illegible]

```
In [21]: batched_churn_scores = pd.read_csv(s3uri_batch_output+'/model.csv.out', usecols=[0,1], names=['id', 'scores'])
gt_df = pd.DataFrame(model_data['Churn? True.']).reset_index(drop=True)
results_df = pd.concat([gt_df, batched_churn_scores], axis=1)

pd.crosstab(index=results_df['Churn? True.'], columns=np.round(results_df['scores']), rownames=['actual'], colnames=['p'])
```

actual \ predictions	0.0	1.0
	0	1
0	2827	23
1	96	387

## Module 6: Upload the Churn Scores to Snowflake

To allow multiple business users to access the output of the SageMaker Batch Inference job we will upload the churn scores to Snowflake.

We will use the Snowflake Internal Stage we created earlier and the Snowflake Python connector to upload the churn scores in the required table.

- 6.1.1 First we need to copy the churn scores to a CSV file in SageMaker Notebook Instance and then we can PUT the file in the Internal Stage on SNOWflake. To do this we execute the last code block in the SageMaker Notebook.

### Upload Churn Score to Snowflake

To be able to allow multiple business users and dashboards simple access to the churn scores we will upload it to Snowflake by using a Snowflake internal stage.

```
In [50]: results_df.to_csv('results.csv', header=False, index=False)
         cs.execute("PUT file://results.csv @ml_results")

Out[50]: <snowflake.connector.cursor.SnowflakeCursor at 0x7f9f78928d68>
```

- 6.1.2 We can now switch back to our Snowflake Worksheet in the browser tab or window we were using earlier. Scroll down in the SQL till you see the comment lines listed below.

```
/*-----  
Get the ML Results returned from Sagemaker and perform some SQL Reporting  
-----*/
```

- 6.1.3 First we will copy the data from the Internal Stage to the Snowflake table using the COPY command. The data will now be in the ML\_RESULTS table.

```
COPY INTO ML_RESULTS FROM @ML_RESULTS;
```

- 6.1.4 Now we are able to join the Churn Scores output in the ML\_RESULTS table from the SageMaker batch inference with the customer data.

Below is a simple analytical query that joins the ML\_RESULTS table with the CUSTOMER\_CHURN table that includes the customer information. This query identifies and ranks states that have 10 or more customers with a churn score above 0.75.

```
SELECT C.STATE, COUNT(DISTINCT(C.CUST_ID))  
FROM ML_RESULTS M INNER JOIN CUSTOMER_CHURN C on M.CUST_ID =  
C.CUST_ID  
WHERE M.CHURN_SCORE >= 0.75  
GROUP BY C.STATE  
HAVING COUNT(DISTINCT(C.CUST_ID)) >= 10  
ORDER BY COUNT(C.CUST_ID) DESC;
```

Results Data Preview			Open History
✓	Query ID	SQL	472ms 6 rows
Filter result...		Download	Copy
Row	STATE	COUNT(DISTINCT(C.CUST_ID))	
1	MD	12	
2	NY	12	
3	WA	12	
4	SC	10	
5	MT	10	
6	NV	10	

## Summary & Next Steps

This lab was designed as a hands-on introduction to Snowflake and SageMaker 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>

## Resetting Your Snowflake Environment

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

```
USE ROLE ACCOUNTADMIN;  
DROP WAREHOUSE IF EXISTS SAGEMAKER_WH;  
DROP DATABASE IF EXISTS ML_WORKSHOP;  
DROP ROLE IF EXISTS SAGEMAKER_ROLE;  
DROP USER IF EXISTS SAGEMAKER;
```

## 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:

- First empty the S3 bucket that was created for the lab
  - Go to the AWS Console and select the CloudFormation Service under Services.
  - Select the S3 bucket and click on Empty
  - Follow the steps to Empty the bucket



**Amazon S3**

**Buckets (2)** Copy ARN Empty Delete Create bucket

Buckets are the fundamental container in Amazon S3 for data storage. For others to access the objects in your buckets, you'll need to explicitly grant them permissions. [Learn more](#)

Find bucket by name

Name	Region	Access	Bucket created
sfsagemaker-test	US West (Oregon) us-west-2	Not public	2019-10-15T18:30:37.000Z
snowflake-sagemaker-workshop- [REDACTED]	US West (Oregon) us-west-2	Objects can be public	2020-05-07T23:52:41.000Z

- Finally delete the AWS CloudFormation Stack that was created in Module 4
  - Go to the AWS Console and select the CloudFormation Service under Services.
  - Select the Stack name that was created for this lab
  - Click the Delete button (see picture below)
  - Follow the steps to complete the Stack deletion

**CloudFormation > Stacks > aws-dev-days-snowflake**

**Stacks (2)** Filter by stack name Active View nested

**aws-dev-days-snowflake** Delete Update Stack actions Create stack

**Stack info** Events Resources Outputs Parameters Template Change sets

**Overview**

Stack ID	Description
arn:aws:cloudformation:us-west-2:[REDACTED]:stack/aws-dev-days-snowflake/d3f292d0-90bd-11ea-8764-0251676acdffa	Snowflake SageMaker Notebook
Status	Status reason
CREATE_COMPLETE	-
Root stack	Parent stack
-	-