SNOWFLAKE

フリートライアル

ハンズオンラボガイド

以下からフリートライアル版をお申し込みください。

https://trial.snowflake.com

Snowflake のどのエディションまたはどのクラウドプロバイダにも対応しています。

ラボを完了するおおよその時間は 90 分ほどです。約 7 クレジットが使用されます。

目次

ラボの概要

モジュール 1： ラボ環境の準備

モジュール 2： Snowflake ユーザーインターフェイスとラボシナリオ

モジュール 3： データをロードする準備

モジュール 4： データの読み込み

モジュール 5： 分析クエリ、結果キャッシュ、クローニング

モジュール 6： 半構造化データ、VIEW、JOIN 操作

モジュール 7： タイムトラベルの使用

モジュール 8： ロールベースのアクセス制御とアカウント管理

モジュール 9： データシェア

まとめと次のステップ

ラボの概要

このエントリーレベルのラボは、Snowflake のユーザーインターフェイスと基本的な機能をご紹介し、効率的に 30 日間のフリートライアル版（https://trial.snowflake.com）をお使い頂けるように設計されております。ラボ作業の完了後、ご利用者の独自データを Snowflake にロードし、より高度な機能を学習する準備ができていることを目指します。

対象読者

データベースおよびデータウェアハウスの管理者あるいはアーキテクト

学習内容

このラボの演習では、次の手順を説明します。

- ステージ、データベース、テーブル、ビュー、ウェアハウスの作成
- 構造化および半構造化データのロード
- テーブル間の結合を含むクエリ
- オブジェクトのクローン
- ユーザーエラーの UNDO
- ロールとユーザを作成し、それらに権限を付与
- 他のアカウントと安全かつ簡単にデータを共有

前提条件

- Snowflake の 30 日間フリートライアル環境
- SQL の基本的な知識、およびデータベースの概念とデータベースオブジェクトの知識と経験
- CSV カンマ区切りファイルと JSON 半構造化データの知識と経験

モジュール 1：ラボ環境の準備

1.1 ラボ環境を準備する手順

1.1.1、まだお済みでない場合は Snowflake 30 日間フリートライアルにご登録ください。

- Snowflake エディション（Standard、Premier、Enterprise など）、クラウドプロバイダ（AWS、Azure など）、およびリージョン（米国東部、EU など）は、このラボでは*依存*しません。物理的に最も近いリージョンを選択することをお勧めします。また、Enterprise エディションを選択すると、下位エディションでは利用できないマルチクラスタなどの高度な機能を活用できます。

- フリートライアル登録後、Snowflake アカウント URL へのアクティベーションリンクが記載されたメールが届きます。この URL をブックマークしておくと、後で簡単にアクセスできます。アクティベーション後、ユーザ名とパスワードを作成し、書き留めておきます。

1.1.2 ブラウザウィンドウのサイズを変更し、このラボガイド PDF と Web ブラウザを並べて表示し、ラボの手順を簡単に実行できるようにします。可能であれば、ラボガイド専用のセカンダリディスプレイを使用することをお勧めします。

1.1.3 https://s3.amazonaws.com/snowflake-workshop-lab/lab_scripts_free_trial.sql をクリックし、ローカルマシンに「lab_scripts_free_trial.sql」ファイルをダウンロードします。このファイルには事前に作成された SQL コマンドが含まれており、ラボの中で使用します。

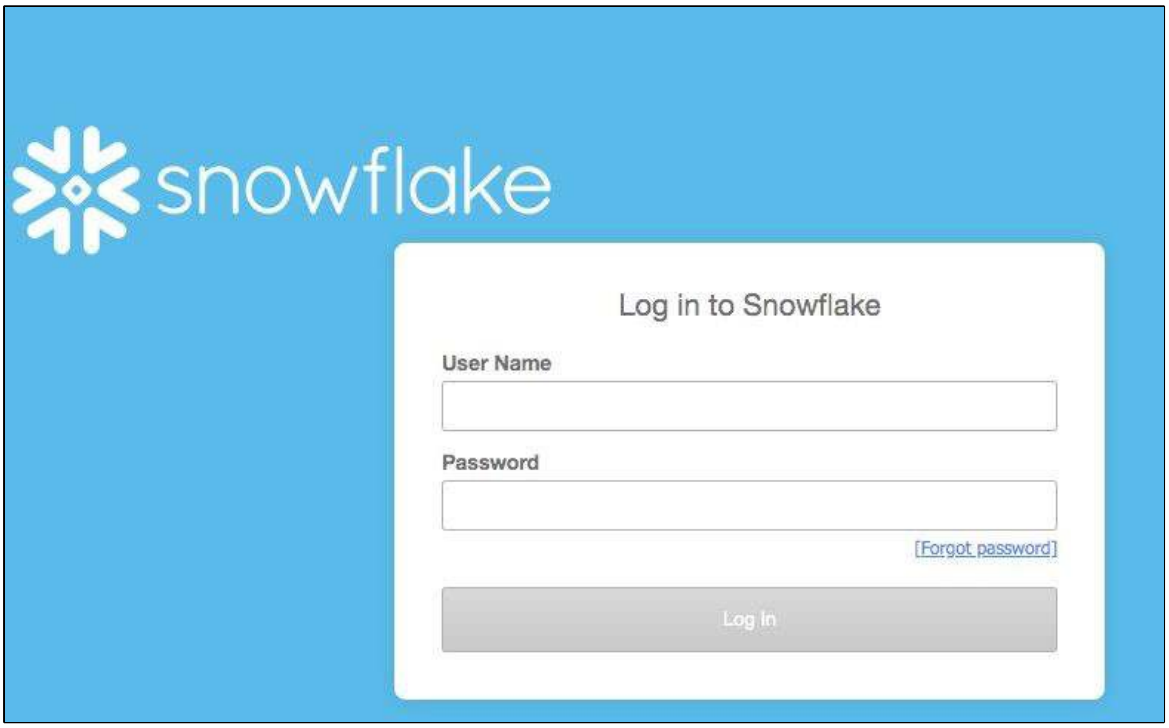
モジュール 2：Snowflake ユーザインターフェイスとラボシナリオ

スクリーンキャプチャ、サンプルコード、および環境について
このラボのスクリーンキャプチャは、演習を完了したときに表示されるものとは若干異なる場合があることをご了承ください。

2.1 Snowflake ユーザーインターフェイス（UI）へのログイン

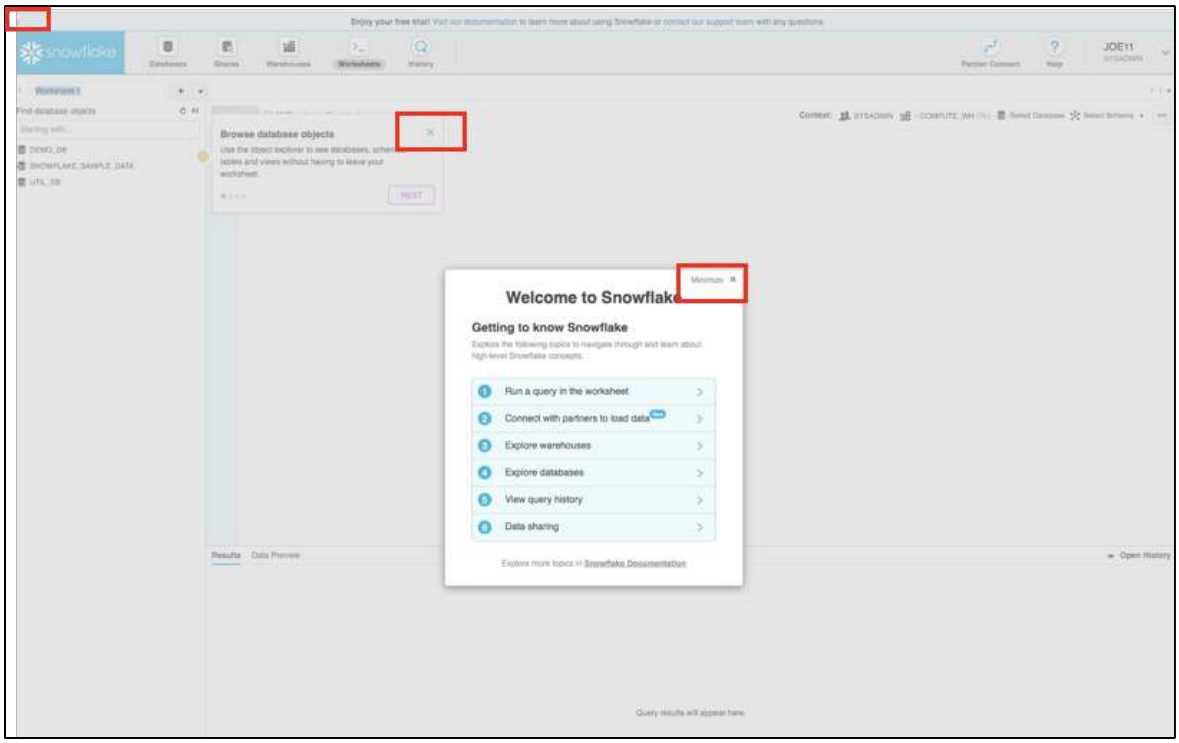
2.1.1 ブラウザウィンドウを開き、Snowflake 30 日間フリートライアル環境の URL を入力します。

2.1.2 以下のログイン画面が表示されるはずです。資格情報（ユーザ名、パスワード）を入力してログインします。



2.2 Welcome ボックスとチュートリアルを閉じます

2.2.1 初めてログインすると、UI に「Welcome」および「ヘルパー」ボックスが表示されます。また、UI の上部に「Enjoy your free trial...」リボンが表示されます。下のスクリーンショットの赤いボックスのアイテムをクリックして、これらを閉じます。

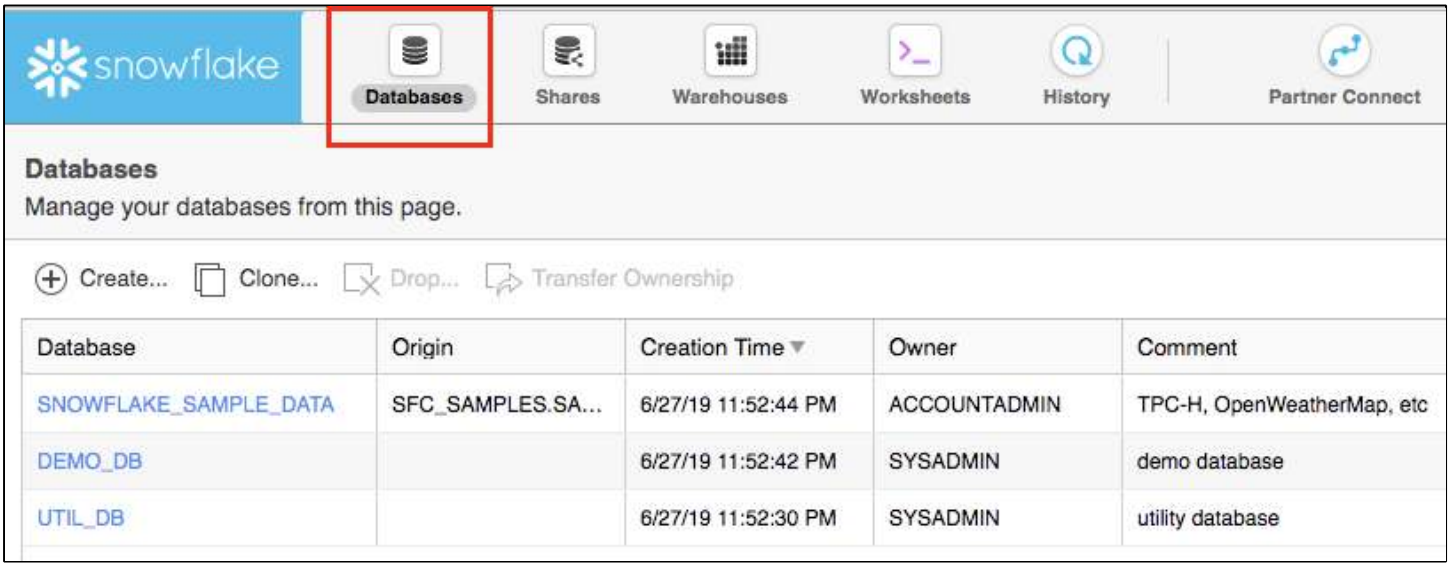


2.3 Snowflake UI をナビゲート

まず、Snowflake を理解するために、ユーザインターフェイスの基本的なコンポーネントについて説明します。UI 上部の左から右に説明します。

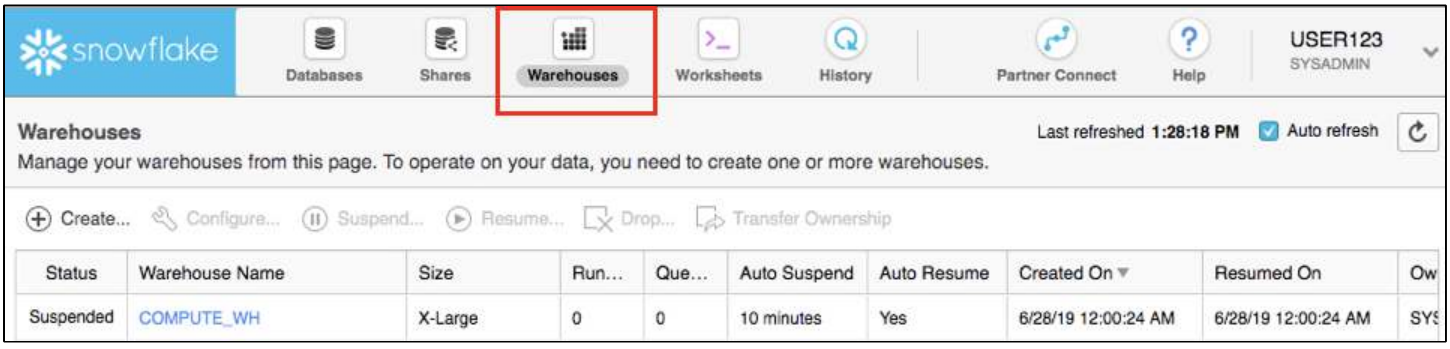
2.3.1 トップメニューでは、Snowflake のさまざまな領域を切り替えることができます。

2.3.2 Databases タブを選択すると、ユーザが作成した、またはアクセス権限を持っているデータベースに関する情報が表示されます。UI から、データベースの作成、クローン、ドロップ、またはオーナーシップの移行をしたり、データをロードできます。フリートライアル環境にいくつかのデータベースが既に存在します。ただし、このラボではこれらを使用しません。



2.3.3 Shares タブを選択すると、データのコピーを作成することなく、別 Snowflake アカウント間または外部ユーザ間で容易かつ確実に Snowflake テーブル（単数または複数）のシェアを構成することができます。このハンズオンラボの最後にデータシェアに関するモジュールがあります。

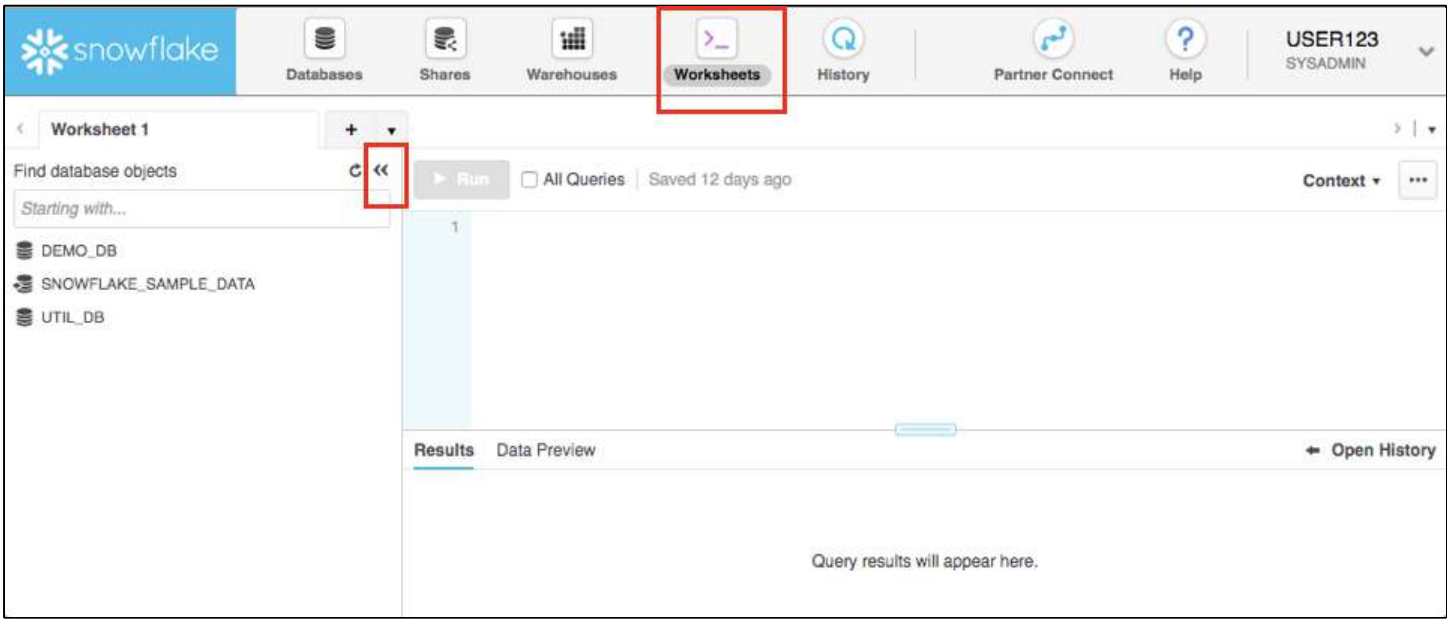
2.3.4 Warehouses タブを選択すると、Snowflake でデータのロードまたはクエリをするためのコンピューティングリソース（ウェアハウス）を管理することができます。「COMPUTE_WH（XL）」というウェアハウスがすでに存在していることを確認します。



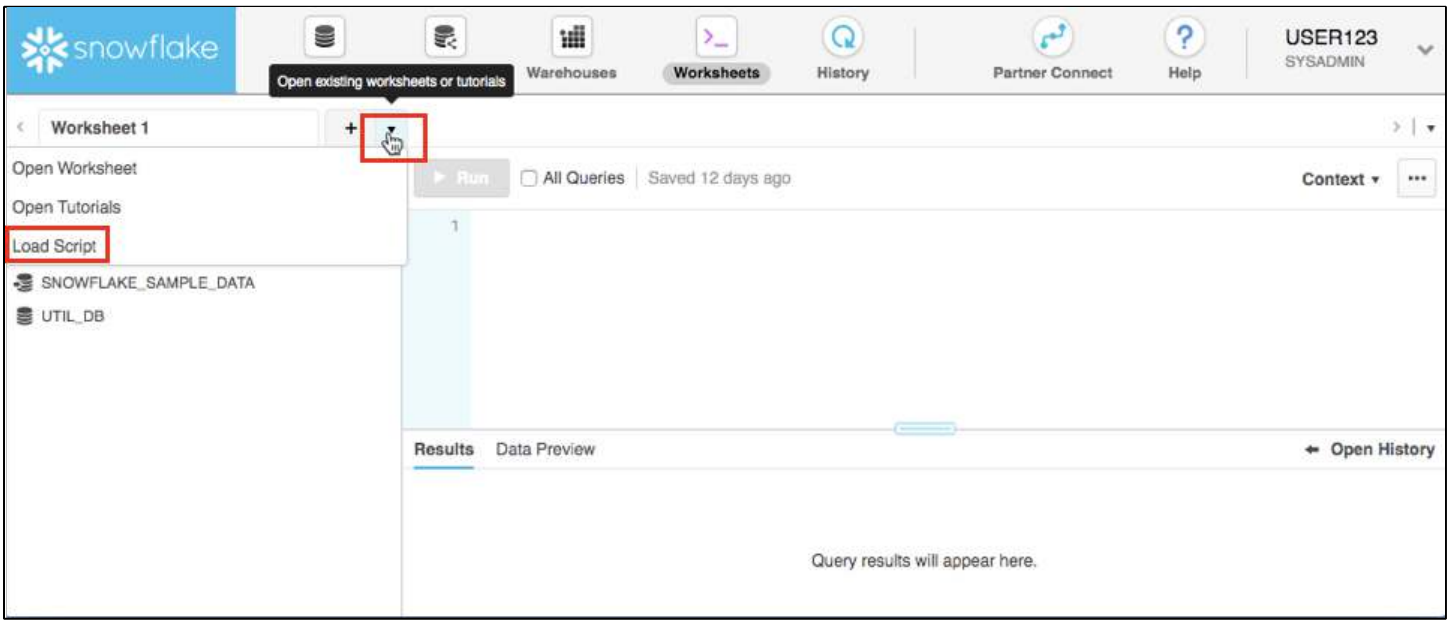
2.3.5 Worksheets タブを選択すると、SQL インタフェースが表示され、DDL や DML 操作を実行できます。クエリと操作が完了すると結果セットを表示します。デフォルトの「Worksheet1」が表示されます。

左側のペインには、データベースオブジェクトブラウザーがあります。このブラウザーを使用すると、ユーザーは、ワークシート用に選択したロールでアクセスできるすべてのデータベース、スキーマ、テーブル、およびビューを確認できます。下側のペインには、クエリと操作の結果が表示されます。

このページのさまざまなウィンドウのサイズを変更するには、ペインの枠をドラッグして調整します。また、ハンズオンラボ中にワークシートで作業をするスペースが必要な場合は、左側ペインで、「<<」をクリックし、データベースオブジェクトブラウザを折りたたみます。このガイドのスクリーンショットの多くでは、このデータベースオブジェクトブラウザが閉じられています。



2.3.6 デフォルトの「Worksheet1」ワークシートタブのすぐ右にある小さな下向きの矢印をクリックし、「Load Script」を選択し、事前にダウンロードした「lab_scripts.sql」ファイルを選択しロードします。このハンズオンラボで実行する必要がある SQL コマンドがすべて新しいワークシートに表示されます。まだこれらの SQL コマンドを実行しないでください。後のラボで、一度に一つずつ実行します。



警告 - この PDF から SQL をワークシートにコピー/貼り付けしないでください。

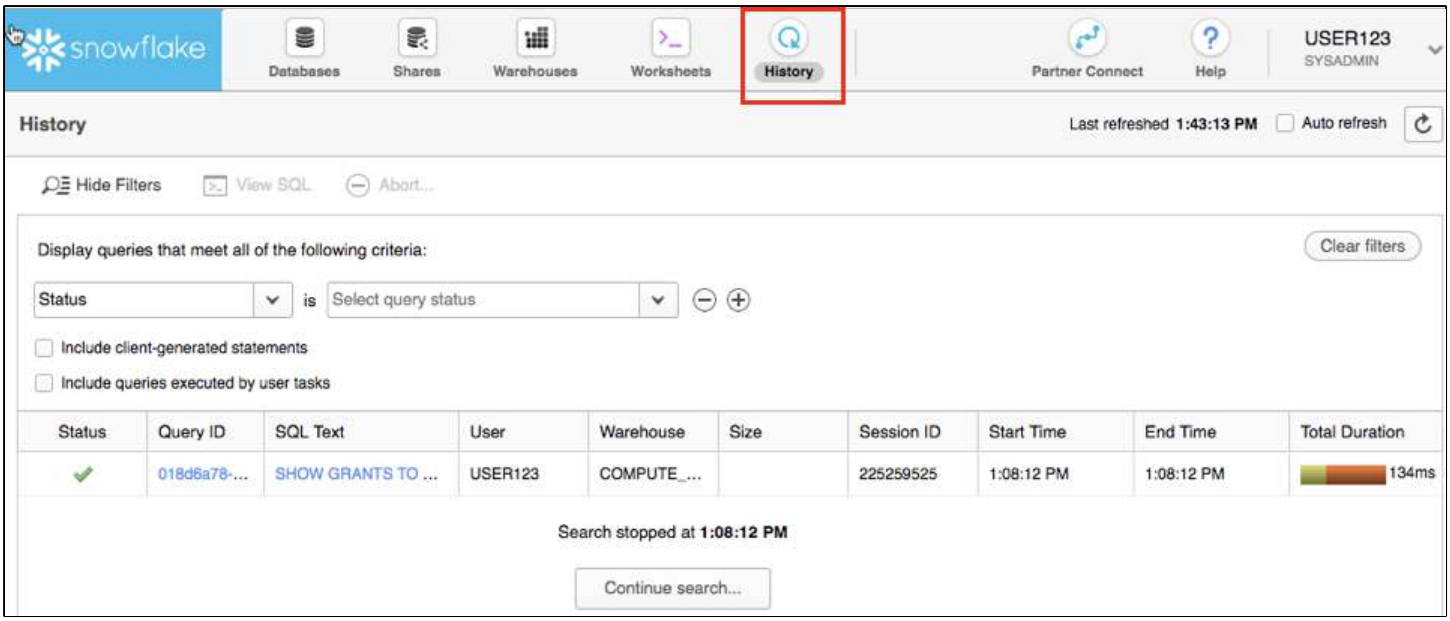
この PDF から SQL コードを Snowflake ワークシートにコピーアンドペーストすると、フォーマットエラーが発生し、SQL が正しく実行されません。先ほど説明した「Load Script」の手順をご利用ください。

古いブラウザまたはセキュリティの厳しい設定がされたブラウザでは、ブラウザが.sql ファイルを開くことができないため、この「Load Script」が機能しない場合があります。その場合、テキストエディタで.sql ファイルを開き、すべてのテキストを.sql ファイルから「Worksheet1」にコピーアンドペーストします。

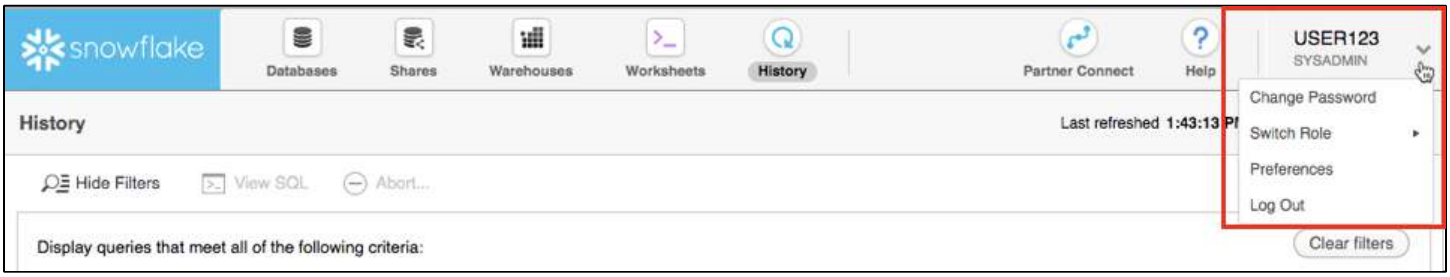
ワークシートと UI

このハンズオンラボの多くの設定は、時間を節約するために、ワークシート内に読み込まれた事前定義済み SQL を介して実行されます。これらの設定は、SQL を使わず UI を介して行うことも可能ですが、より時間がかかります。

2.3.7 History タブでは、Snowflake アカウント上で最後の 14 日間に実行されたすべてのクエリの詳細を表示することができます（詳細については、クエリ ID をクリックし、ドリルダウンします）。



2.3.8 ユーザ名が表示されている UI の右上下矢印をクリックすると、パスワード、ロール、または設定を変更することができます。Snowflake には、事前にシステムが定義したロールがいくつかあります。現在、SYSADMIN がデフォルトのロールになっており、このラボの終わりまでこのロールを使います。



SYSADMIN

このラボでは、アカウントにウェアハウスとデータベースおよびその他のオブジェクトを作成する権限を持つ SYSADMIN（システム管理者）のロールを使います。

実際の環境では、このラボのタスクごとにロールを使用し、ユーザーにロールを割り当てます。

Snowflake のアクセス制御の詳細はこのラボの終わりのモジュールで説明します、また以下から詳細を確認できます。

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

2.4 ラボシナリオ

2.4.1 この Snowflake ラボでは、実世界の「シナリオ」を想定することで、ラボでの操作をする理由と順序を理解しやすくします。

このラボの「シナリオ」は、米国ニューヨーク市にある本物の公共自転車共有システムである Citi Bike の分析チームに基づいています。このチームは、データの分析を実行して、ライダーに最適なサービスを提供する方法をよりよく理解したいと考えています。

私たちは、まずライダーのトランザクションデータを CSV 構造化データとして Snowflake にロードします。これは、Citi Bike の内部トランザクションシステムから抽出されます。その後、私たちは Snowflake に、オープンソースである半構造化 JSON の気象データをロードし、自転車に乗る回数と天候に相関関係があるかどうかを分析します。

モジュール 3：データをロードする準備

Citi Bike ライダーの構造化トランザクションデータを Snowflake にロードする準備から始めましょう。

このモジュールでは、次の手順を説明します。

- データベースとテーブルを作成する
- 外部ステージを作成する
- データのファイルフォーマットを作成する

Snowflake へのデータロード

COPY コマンド、Snowpipe による自動取り込み、外部コネクタ、またはサードパーティ ETL / ELT 製品など、Snowflake にデータをロードする方法は多数あります。以下から、Snowflake へのデータロードに関する詳細情報をご参照ください。

https://docs.snowflake.net/manuals/user-guide-data-load.html

このモジュールでは、COPY コマンドを使い、S3 ストレージにあるデータを手動でロードする手順を確認します。現実的には、ユーザは自動化されたデータのロード処理を行うために自動化されたプロセスや ETL 製品を使用しているでしょう。

私たちが使用するデータは、Citi Bike NYC によって公開されているデータです。データはエクスポートされ、US-EAST リージョンの Amazon AWS S3 バケットに事前にステージングされています。データは、トラベル時間、場所、ユーザータイプ、性別、ライダーの年齢などに関する情報で構成されています。AWS S3 では、データは 61,500,000 行、377 個のオブジェクト、合計 1.9GB の圧縮サイズを表示します。

以下は、1 つの Citi Bike CSV データファイルからの抜粋です

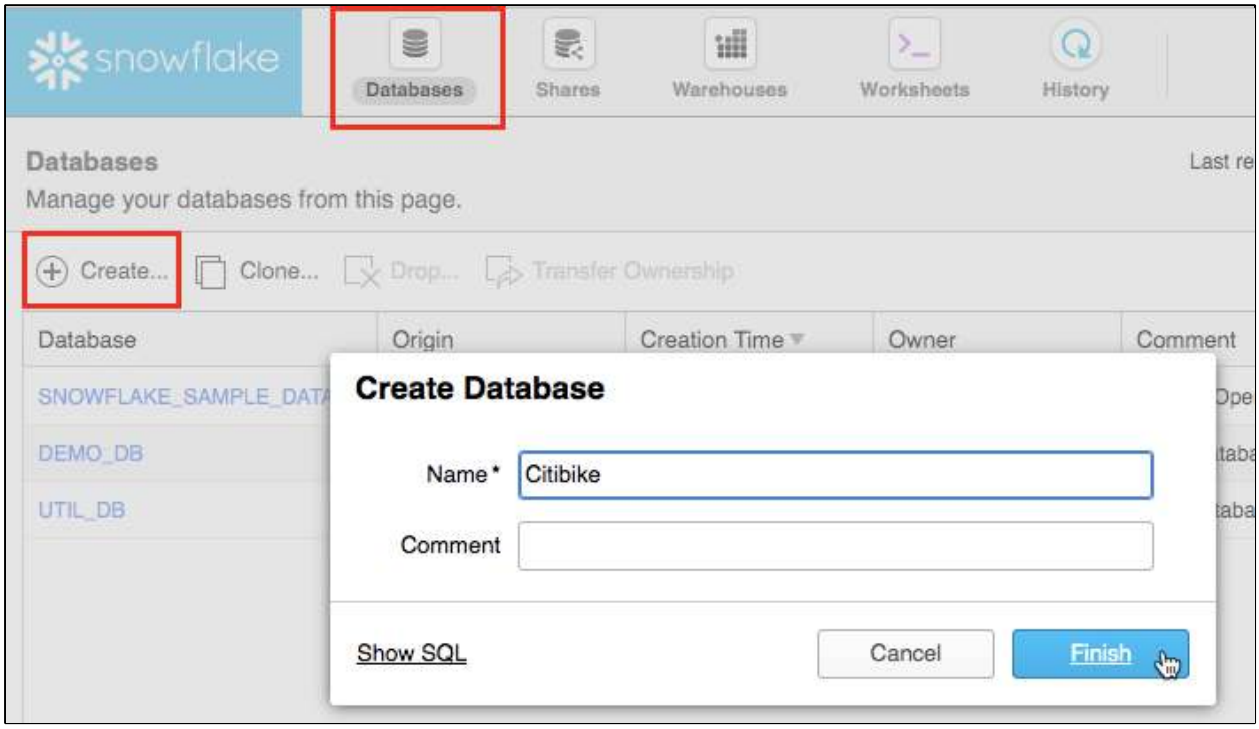
```
"tripduration","starttime","stoptime","start station id","start station name","start
station latitude","start station longitude","end station id","end station name","end
station latitude","end station
longitude","bikeid","name_localizedValue0","usertype","birth year","gender"
196,"2018-01-01 00:01:51","2018-01-01 00:05:07",315,"South St & Gouverneur Ln",
40.70355377,-74.00670227,259,"South St & Whitehall St",
40.70122128,-74.01234218,18534,"Annual Membership","Subscriber",1997,1
207,"2018-01-01 00:02:44","2018-01-01 00:06:11",3224,"W 13 St & Hudson St",
40.73997354103409,-74.00513872504234,470,"W 20 St & 8 Ave",
40.74345335,-74.00004031,19651,"Annual Membership","Subscriber",1978,1
613,"2018-01-01 00:03:15","2018-01-01 00:13:28",386,"Centre St & Worth St",
40.71494807,-74.00234482,2008,"Little West St & 1 Pl",
40.70569254,-74.01677685,21678,"Annual Membership","Subscriber",1982,1
```

二重引用符で囲まれたヘッダー行が 1 つあるコンマ区切り形式です。この情報は、このモジュールの後半で、このデータを保存する Snowflake テーブルを構成する際に役立ちます。

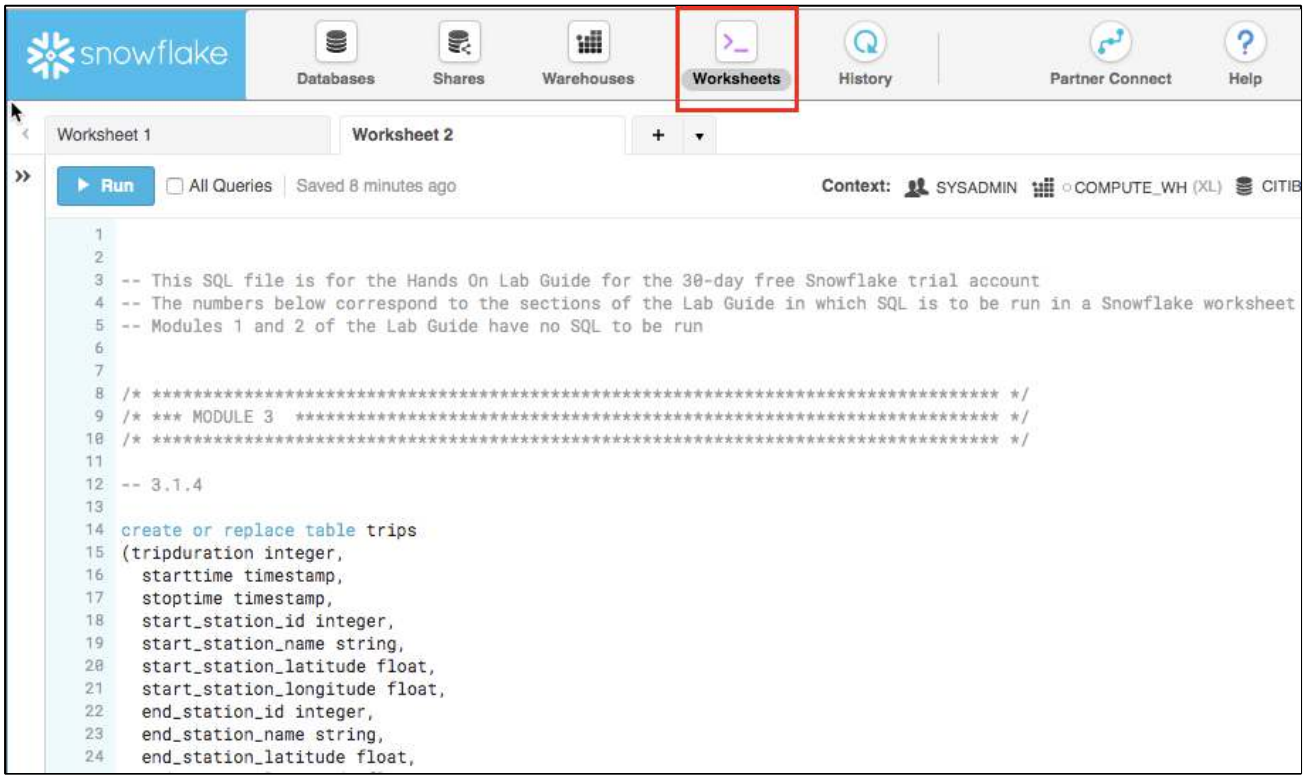
3.1 データベースとテーブルを作成する

3.1.1 構造化データのロードに使用される CITIBIKE というデータベースを作成しましょう。

UI の上部で[Databases]タブを選択します。次に、「Create」をクリックして、データベースに「CITIBIKE」という名前を付け、「Finish」をクリックします。



3.1.2 Snowflake UI の上部で、[Worksheets]タブをクリックします。前の手順で読み込まれたすべての SQL を含むワークシートが表示されます。



3.1.3 ワークシート内でコンテキストを適切に設定する必要があります。右上の「Context」セクションの横にあるドロップダウン矢印をクリックし、ワークシートのコンテキストメニューを表示します。ユーザが各ワークシートから表示および実行できる要素を制御します。ここでは、コンテキストを設定するために UI を使用します。ラボの後半で、ワークシートの SQL コマンドを介してワークシートコンテキストを設定します。

必要に応じて、下向き矢印を選択、クリックしてコンテキストを表示します。

ロール：SYSADMIN
ウェアハウス：COMPUTE_WH（XL）
データベース：CITIBIKE
スキーマ：PUBLIC

DDL 操作は無料です！

これまでに行ったすべての DDL 操作はウェアハウスリソースを必要としないため、すべてのオブジェクトは無料で作成できます。

3.1.4 ここで、カンマ区切りのデータをロードするために使用される TRIPS というテーブルを作成しましょう。Snowflake UI の[Worksheets]タブを使用して、DDL（データ定義言語）を実行し、テーブルを作成します。前の手順に基づいてロードした以下 SQL テキスト部分をワークシートで実行します。

```
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);
```

コマンドを実行するための多くのオプション。

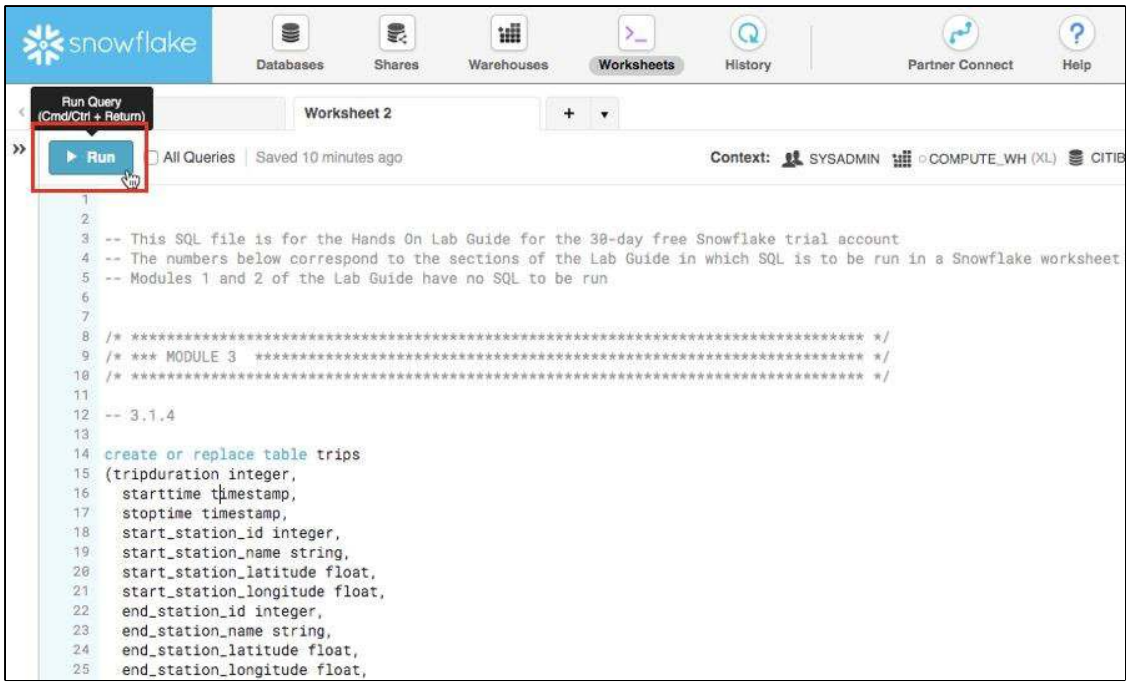
SQL コマンドは、UI、[Worksheets]タブ、SnowSQL コマンドラインツール、ODBC / JDBC 経由で接続した SQL エディター、または Python や Spark コネクタを使用して実行できます。

前述したように、このラボでは、時間を節約するために、ワークシート内に事前にロードした SQL を介して操作を実行します。

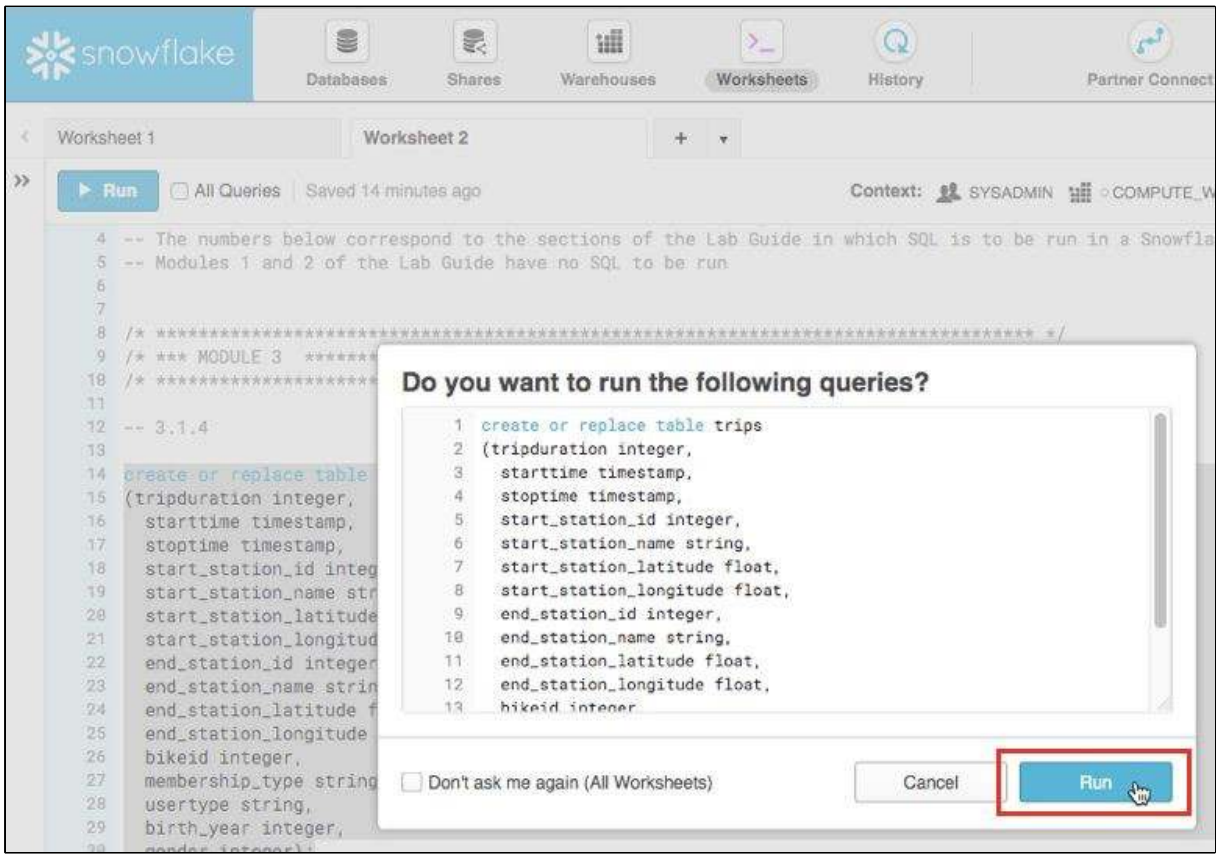
3.1.5 コマンド内の任意の場所にカーソルを置き、ページ上部の青い「Run」ボタンをクリックするか、キーボードの Ctrl / Cmd + Enter を押して、クエリを実行します。

警告

このハンズオンラボでは、ワークシートの上部にある[All Queries]ボックスをオンにしないでください。特定の順序で一度に一つずつ SQL クエリを実行します。一度にすべてではありません。



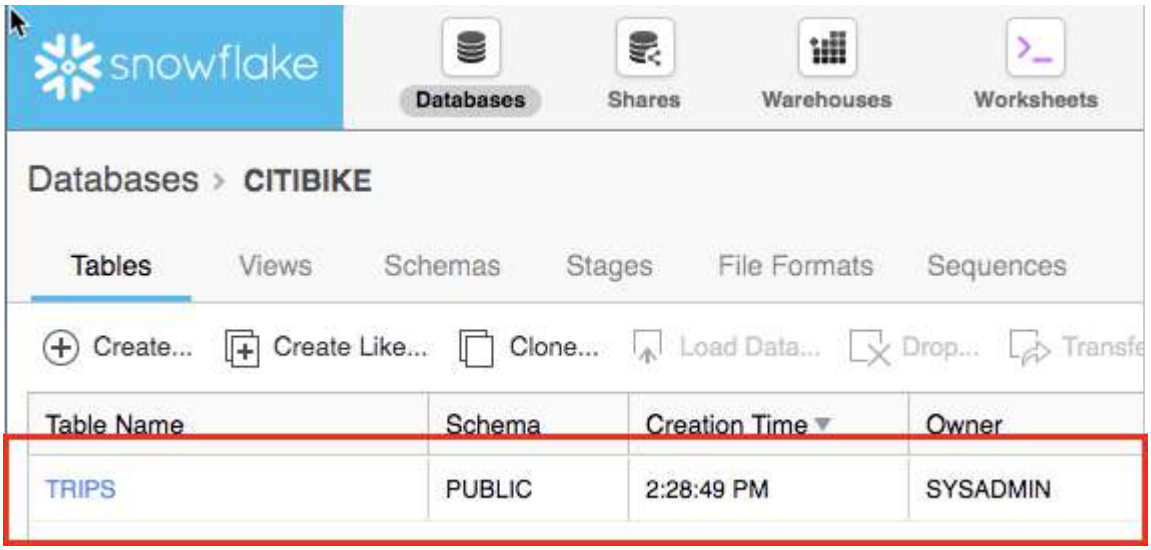
3.1.6 SQL のコマンドテキスト全体を強調表示して（コマンドにカーソルを置いただけではありません）実行した場合、「Do you want to run the following queries?」という確認ボックスが表示されます。ボックス内の青い「Run」ボタンをクリックします。今後、SQL を実行する度に、この確認ボックスの[Run]ボタンをクリックし続けるか、このボックスの[Don't ask me again（All Worksheets）]オプションをチェックし、確認ボックスを非表示にできます。



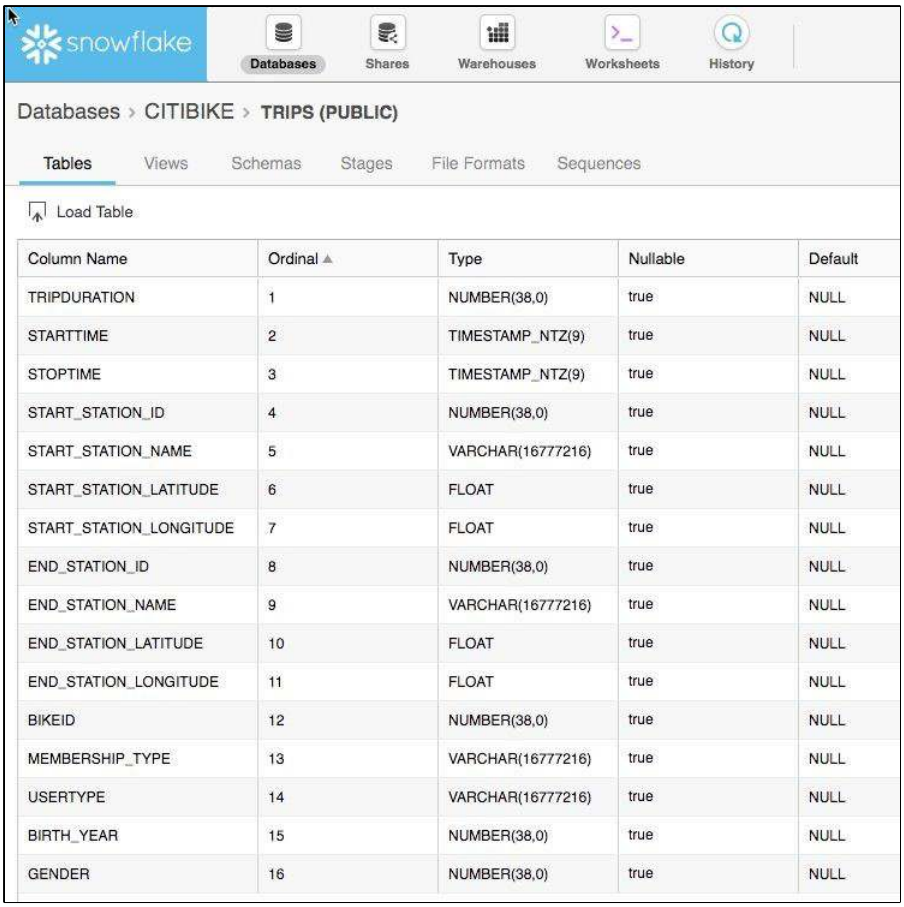
3.1.7 テーブル TRIPS が作成されたことを確認します。ワークシートの下部に、「Table TRIPS successfully created」という「Results」セクションが表示されます。

3.1.8 ページの上部で、[Databases]タブに移動し、[CITIBIKE]データベースリンクをクリックします。新しく作成された TRIPS テーブルが表示されます。

重要：データベースが表示されない場合は、ブラウザで非表示になっている可能性があるため、ブラウザを展開します



3.1.9「TRIPS」ハイパーリンクをクリックして、設定したテーブル定義を表示します。



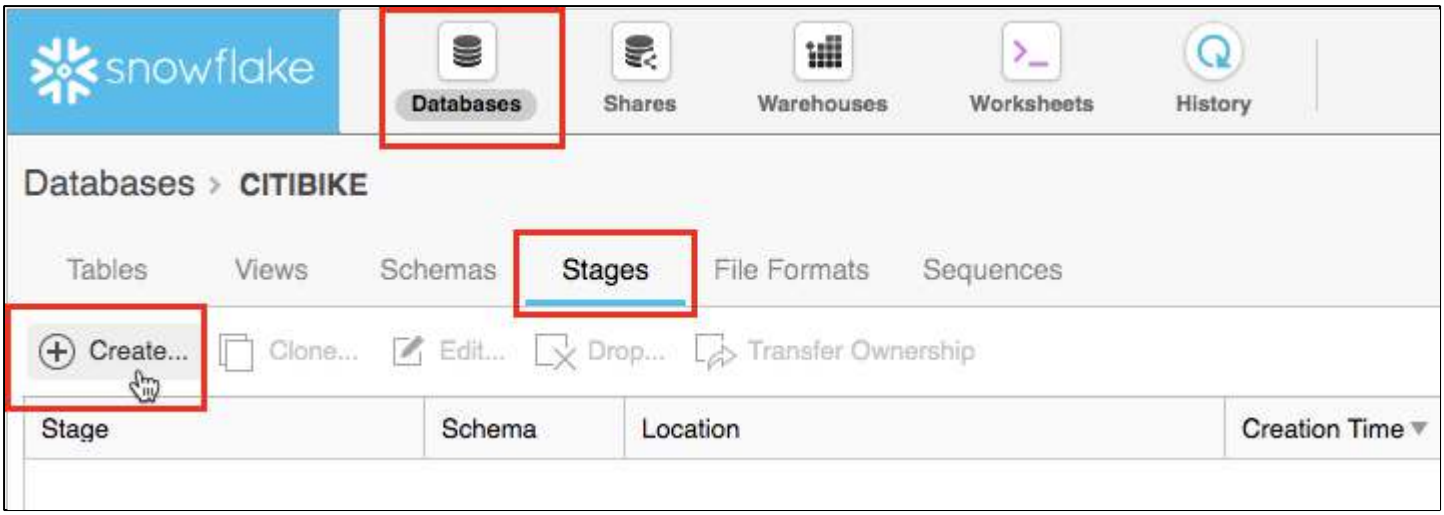
Column Name	Ordinal ▲	Type	Nullable	Default
TRIPDURATION	1	NUMBER(38,0)	true	NULL
STARTTIME	2	TIMESTAMP_NTZ(9)	true	NULL
STOPTIME	3	TIMESTAMP_NTZ(9)	true	NULL
START_STATION_ID	4	NUMBER(38,0)	true	NULL
START_STATION_NAME	5	VARCHAR(16777216)	true	NULL
START_STATION_LATITUDE	6	FLOAT	true	NULL
START_STATION_LONGITUDE	7	FLOAT	true	NULL
END_STATION_ID	8	NUMBER(38,0)	true	NULL
END_STATION_NAME	9	VARCHAR(16777216)	true	NULL
END_STATION_LATITUDE	10	FLOAT	true	NULL
END_STATION_LONGITUDE	11	FLOAT	true	NULL
BIKEID	12	NUMBER(38,0)	true	NULL
MEMBERSHIP_TYPE	13	VARCHAR(16777216)	true	NULL
USERTYPE	14	VARCHAR(16777216)	true	NULL
BIRTH_YEAR	15	NUMBER(38,0)	true	NULL
GENDER	16	NUMBER(38,0)	true	NULL

3.2 外部ステージを作成する

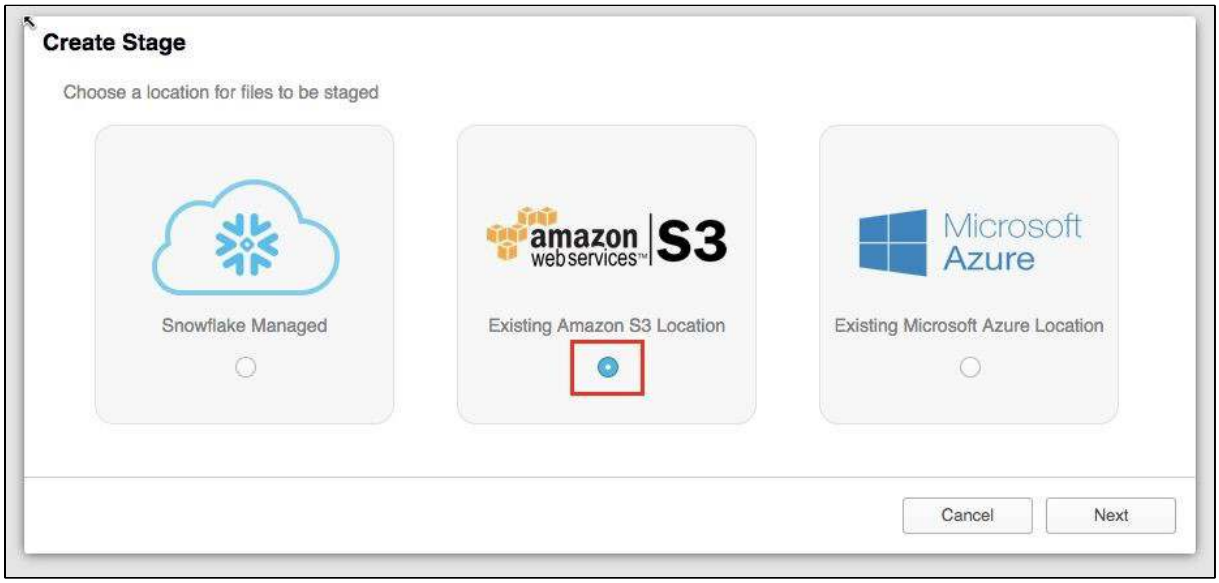
パブリックに公開された外部用 S3 バケットに、既にステージングされた構造化コンマ区切りデータを使用します。このデータを使用する前に、まず外部 S3 バケットの場所を指定するステージを作成する必要があります。

注 - このラボでは、AWS-East リージョンにあるバケットを使用しています。現実的には、データ出力/転送コストを防ぐために、ユーザが利用するクラウドプロバイダーとユーザの Snowflake 環境が同じリージョン内にあるステージ場所を選択するとよいでしょう。

3.2.1 「Databases」タブから「CITIBIKE」データベースをクリックし、「Stages」をクリックして「Create…」をクリックします。



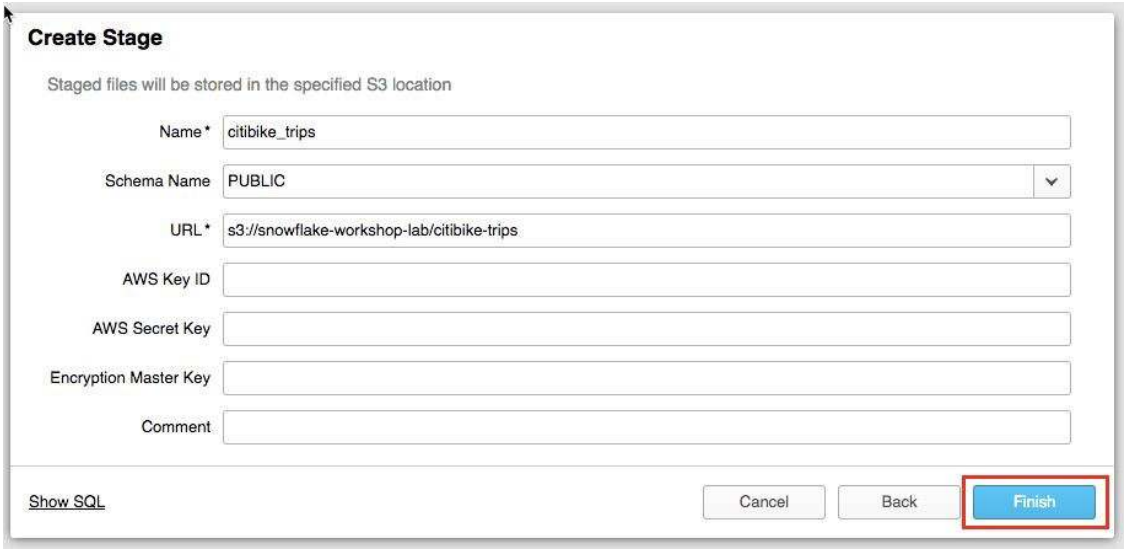
3.2.2「Existing Amazon S3 Location」のオプションを選択し、「Next」をクリックします



3.2.3 表示される「Create Stage」ボックスで、次の設定を入力/選択し、「Finish」をクリックします。

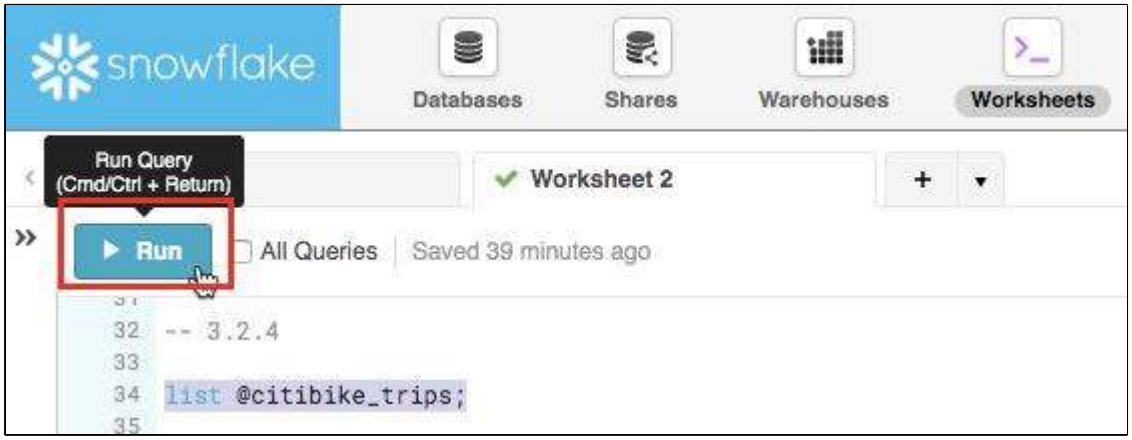
Name	citibike_trips
Schema Name	PUBLIC
URL *	s3://snowflake-workshop-lab/citibike-trips

注 - このラボの S3 バケットはパブリックに公開されているため、キーフィールドを空のままにしておくことができます。現実的には、バケットにアクセスするにはこれらのキー情報が必要になるでしょう。



3.2.4 では、cititibike_trips ステージの内容を見てみましょう。ページの上部にある[Worksheets]タブをクリックします。次に以下のステートメントを実行します。

```
list @citibike_trips;
```



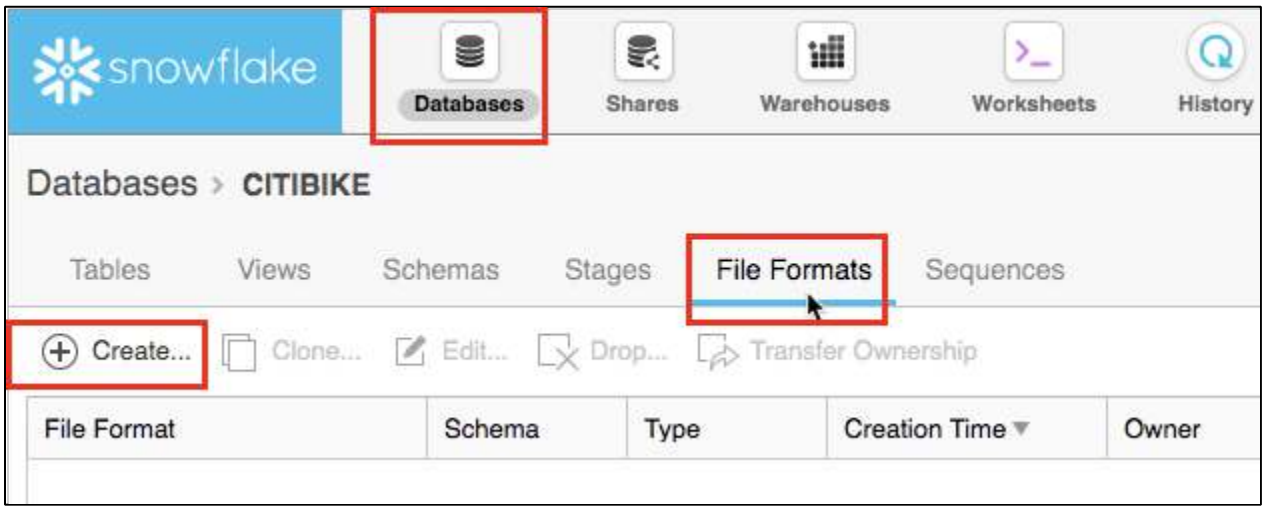
下のペインにある[Results]ウィンドウに出力が表示されます。

Row	name	size	md5	last_modified
1	s3://snowflake-workshop-lab/citibike-tri…	3072073	cfc69e04228a94d1337ab383a3af7472	Wed, 12 Jun 2019 16:30:58 GMT
2	s3://snowflake-workshop-lab/citibike-tri…	2877852	92a1c064a3c632f338b57d5c6531e97d	Wed, 12 Jun 2019 16:30:58 GMT
3	s3://snowflake-workshop-lab/citibike-tri…	3174598	39faac098802c1b29f2d4d99f31378be	Wed, 12 Jun 2019 16:30:58 GMT
4	s3://snowflake-workshop-lab/citibike-tri…	3031012	cd0dca1dcfa309c0bb4bd40d1265c3a9	Wed, 12 Jun 2019 16:30:58 GMT
5	s3://snowflake-workshop-lab/citibike-tri…	3005838	fb24c0cc5fb6ee54d2aa4d50265792c7	Wed, 12 Jun 2019 16:30:58 GMT
6	s3://snowflake-workshop-lab/citibike-tri…	3099881	441efe806352c57a50c4f31afcccb2e3	Wed, 12 Jun 2019 16:30:58 GMT
7	s3://snowflake-workshop-lab/citibike-tri…	3060952	372765a1ca713eeb1d56f79e0956e48f	Wed, 12 Jun 2019 16:30:59 GMT

3.3 ファイルフォーマットの作成

データを Snowflake に読み込む前に、データ構造に一致するファイルフォーマットを作成する必要があります。

3.3.1 [Databases]タブで、[CITIBIKE] データベースのハイパーリンクをクリックします。次に、「File Formats」をクリックします。次に、「Create…」をクリックします。

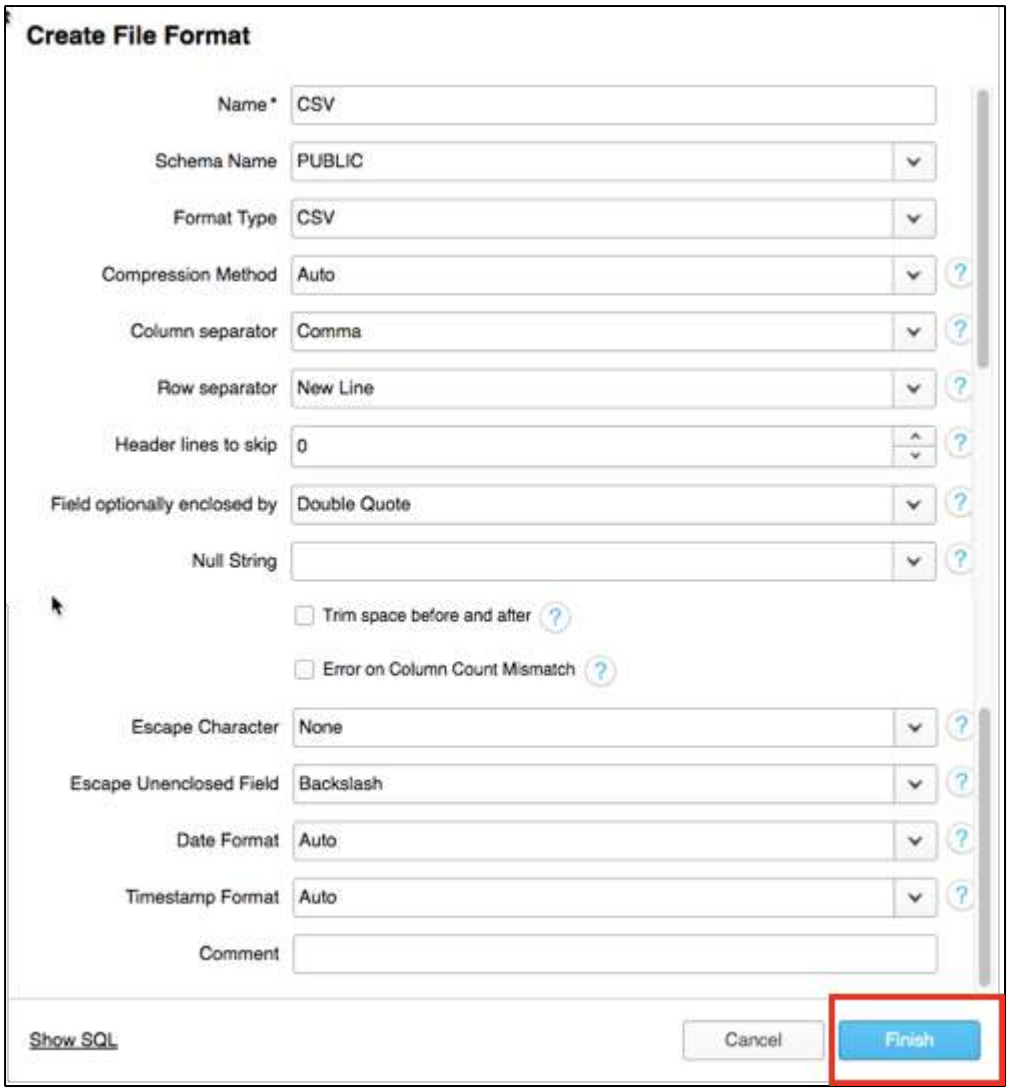


3.3.2 出力されたダイアログボックスで、ファイルフォーマットを作成します。表示されているデフォルト設定をそのままにして、以下の変更を行います。

Name ： CSV
Field optionally enclosed by：Double Quote
NULL String ： <このフィールドの既存テキストを削除し空にします>
Error on Column Count Mismatch：<このチェックボックスのチェックを外します>

重要：[Error on Column Count Mismatch]チェックボックスが表示されない場合は、ダイアログボックスを下にスクロールします

完了すると、ダイアログボックスの設定は次のようになります。



次に、「Finish」ボタンをクリックしファイルフォーマットを作成します。

モジュール 4：データのロード

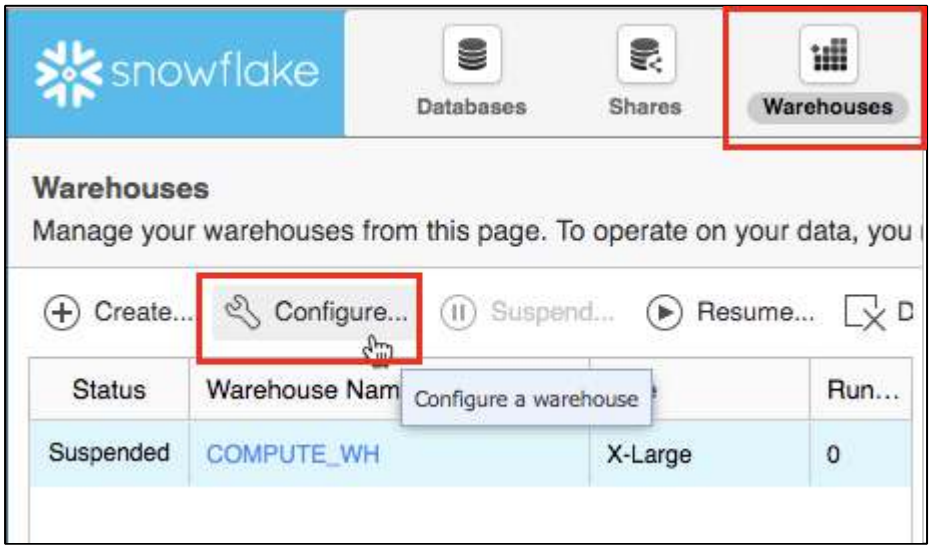
このモジュールでは、データウェアハウスと COPY コマンドを使用し、作成した Snowflake テーブルへの構造化データの一括読み込みを開始します。

4.1 データロード時にウェアハウスサイズを変更して使用する

データをロードするには、計算能力が必要です。Snowflake のコンピュートノードはウェアハウスと呼ばれ、データの読み込み、クエリの実行、DML 操作の実行など、ワークロードに応じて動的にスケールアウトまたはスケールインできます。また、各ワークロードに独自のデータウェアハウスを持たせることができるため、リソースの競合が起こりません。

4.1.1 [Warehouses]タブに移動します。上部の[Create...]オプションを使い、新しいウェアハウスをすばやく作成できます。ここでは、30 日間のフリートライアル環境にデフォルトで存在する既存のウェアハウス「COMPUTE_WH」を使用します。

この「COMPUTE_WH」ウェアハウスの行（「COMPUTE_WH」という青色のハイパーリンクではありません）をクリックして、行全体を強調表示します。次に、その上の「Configure...」オプションをクリックして、「COMPUTE_WH」の構成の詳細を確認します。このウェアハウスを使用して、AWS S3 からデータをロードします。



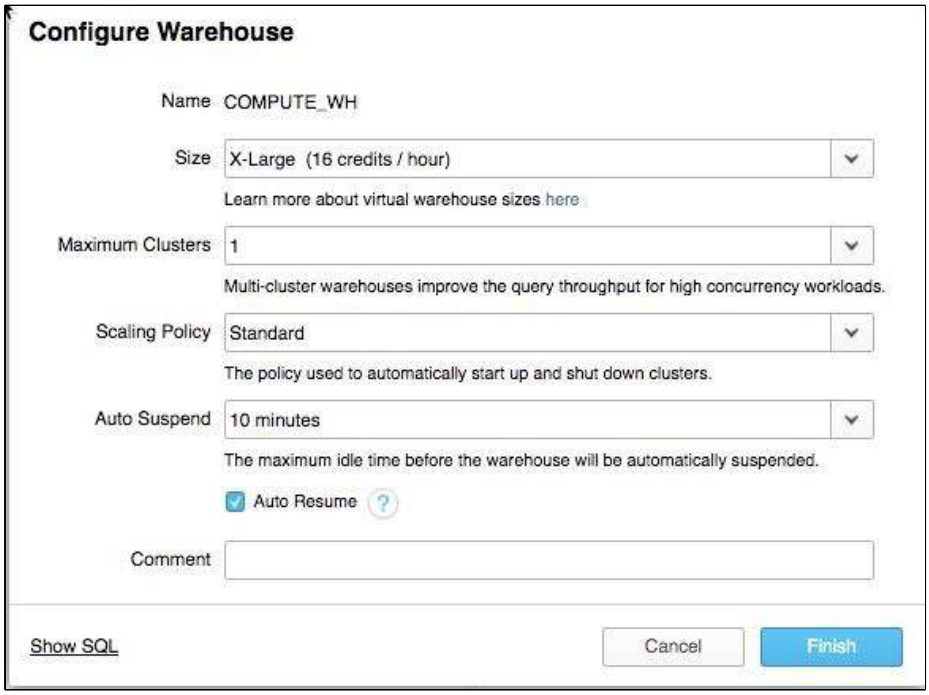
4.1.2 ここには多くの機能があり、その多くは他のデータウェアハウスと比較して Snowflake 固有機能となります。では、それら特徴的なウェアハウスの設定を見ていきましょう。

注 - Enterprise 以上の Snowflake エディションがない場合、下のスクリーンショットの「Maximum Clusters」または「Scaling Policy」の構成は表示されません。マルチクラスタリングはこのラボでは使用しませんが、Snowflake の重要な機能であるため、これについては引き続き説明します。

- 「Size」ドロップダウンリストは、ウェアハウスのサイズを選択するオプションです。大規模なデータ読み込み操作または複雑な計算クエリが必要とされる場合、より大きなウェアハウスが必要になります。T シャツのサイズは、基になる AWS EC2 または Azure Virtual Machines の計算ノードに変換されます。T シャツのサイズが大きいほど、クラウドプロバイダからのコンピューティングリソースが多くそのウェアハウスに割り当てられます。例として、4- XL オプションは 128 ノードを割り当てます。また、このサイズ設定は、ウェアハウスが作成された後も自由に変更できます。

- Snowflake Enterprise エディション以降を使用している場合、最大クラスタセクションが表示されます。ここで、単一のウェアハウスをセットアップし、最大 10 個のマルチクラスタにすることができます。例として、先ほど言及した 4-XL ウェアハウスに最大クラスタサイズ 10 が割り当てられた場合、そのウェアハウスを構成する 1280（128 * 10）AWS EC2 または Azure VM マシンモードにスケールアップできます…しかも、これを数秒で行います！マルチクラスタは、同じウェアハウスを使用しさまざまなクエリを同時に実行する多くのビジネスユーザがいるなど、同時実行シナリオに最適です。このシナリオでは、複数のクラスタにさまざまなクエリを割り当て、それらが高速に実行されるようにすることができます。

- 最後の「Auto Suspend」リストボックスでは、不必要にクレジットが消費されないように使われていないウェアハウスを自動的に一時停止できます。また、中断されたウェアハウスを自動的に再開（開始）するオプションもあるため、新しいワークロードが割り当てられた場合、自動的にウェアハウスが再開されます。この機能により、Snowflake の公正な「使用に応じて支払う」コンピューティング価格設定モデルが実現され、ユーザはデータウェアハウスのコストを最小限に抑えることができます。



Snowflake Compute と他のデータウェアハウス

先ほど説明したウェアハウス/コンピューティング機能の多くは、Snowflake で簡単に作成でき、数秒で実行できるものであり、ウェアハウスの作成、スケールアップ、アウト、自動停止/再開などが可能です。しかし、オンプレミスのデータウェアハウスでは、これらの機能を実行するのは非常に困難（または不可能）です。

物理ハードウェア、ワークロードの急増に対するハードウェアのオーバープロビジョニング、大幅な構成作業、およびより多くの準備が必要です。他のクラウドデータウェアハウスでも、設定作業と時間を大幅に増やすことなく、Snowflake のようにスケールアップおよびアウトすることはできません。

警告 - 支出にご注意ください！

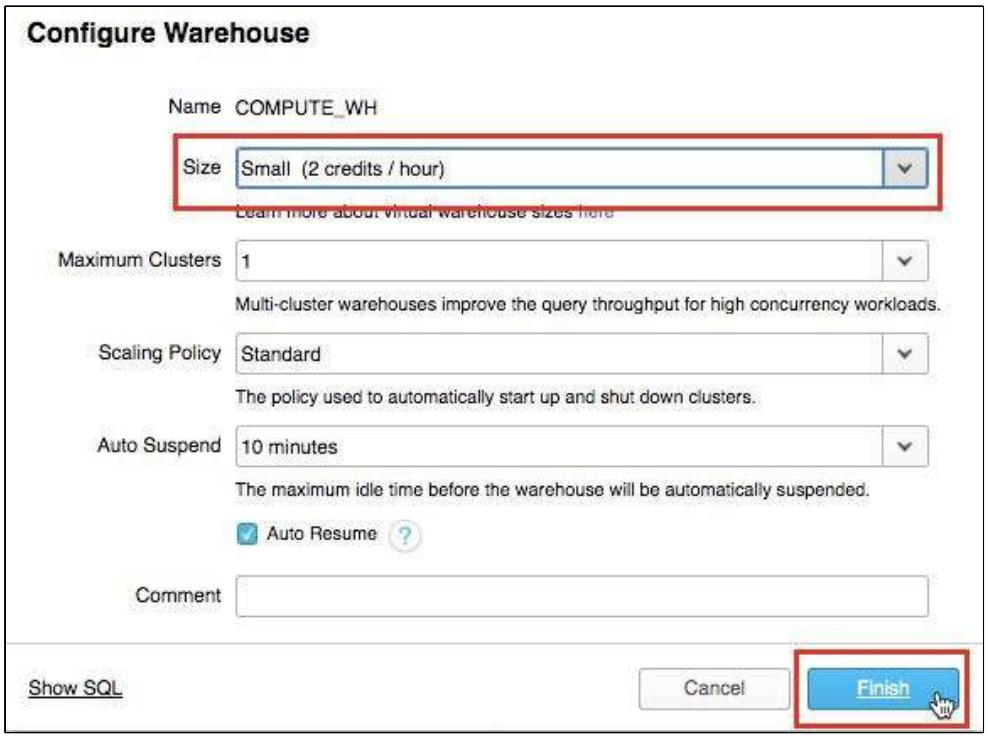
このハンズオン中またはハンズオン後には、次のことを実施しないことをお薦めします。400 ドルの無料クレジットが必要以上に早く消費されてしまいます。

- 自動サスペンドを無効にする。自動サスペンドが無効になっている場合、ウェアハウスは使用されていなくても実行し続け、クレジットを消費します。

● ワークロードが必要とするリソース以上の過剰なウェアハウスサイズを使用する。ウェアハウスが大きいほど、より多くのクレジットが消費されます。

4.1.3 作成されたデータウェアハウスを使用し、構造化データを Snowflake にロードします。ただし、最初はウェアハウスのサイズを縮小し、ウェアハウスに含まれる計算能力を削減します。今は、このロードにかかる時間を記録し、次回、大きなウェアハウスで同じロード操作を再実行します。大きなウェアハウスで性能がどれだけ速くなるかを観察します。

データウェアハウスのサイズを X-Large から Small に縮小します。次に、「Finish」ボタンをクリックします。

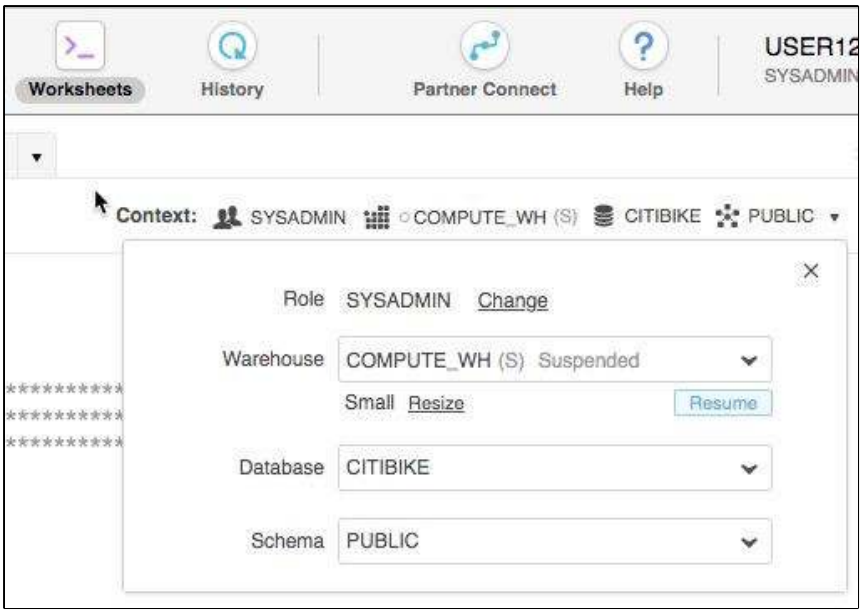


4.2 データを読み込む

これで、COPY コマンドを実行して、前に作成した TRIPS テーブルにデータをロードできます。

4.2.1 UI の上部から、「Worksheets」タブをクリックします。ワークシートの右上からコンテキストダイアログボックスを表示し、コンテキストが正しいことを確認します。

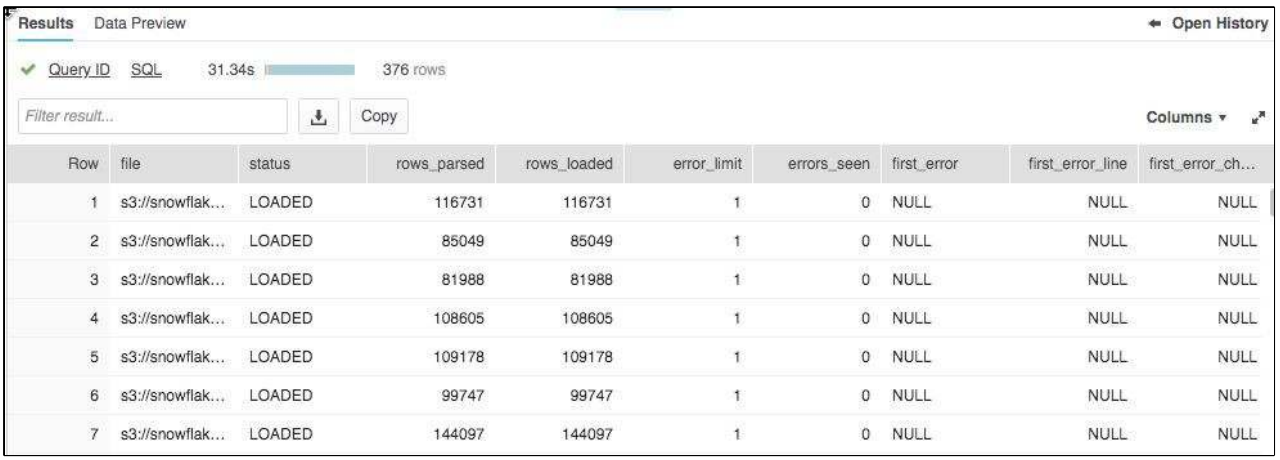
Role ： SYSADMIN
Warehouse ： COMPUTE_WH（S）
Database ： CITIBIKE
Schema：PUBLIC



4.2.2 ワークシートで次のステートメントを実行し、ステージングされたデータをテーブルにロードします。これには最大 30 秒かかる場合があります。

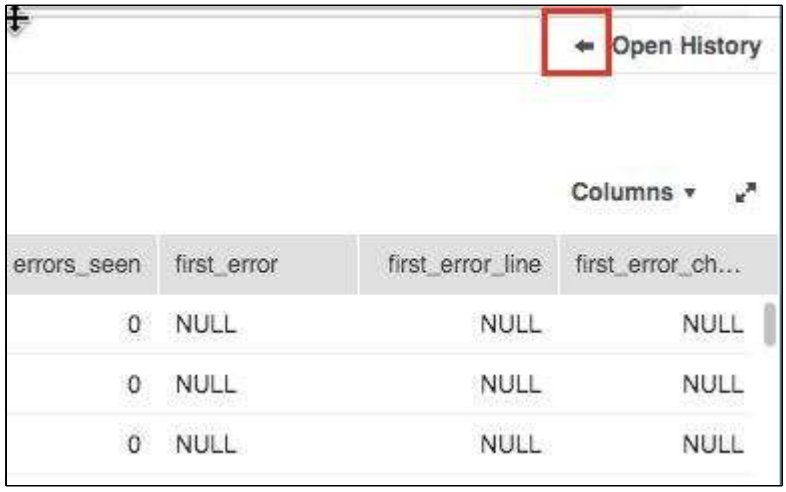
```
copy into trips from @citibike_trips
file_format=CSV;
```

[Results]ウィンドウに、ロードの状況が表示されます。



Row	file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error	first_error_line	first_error_ch...
1	s3://snowflak...	LOADED	116731	116731	1	0	NULL	NULL	NULL
2	s3://snowflak...	LOADED	85049	85049	1	0	NULL	NULL	NULL
3	s3://snowflak...	LOADED	81988	81988	1	0	NULL	NULL	NULL
4	s3://snowflak...	LOADED	108605	108605	1	0	NULL	NULL	NULL
5	s3://snowflak...	LOADED	109178	109178	1	0	NULL	NULL	NULL
6	s3://snowflak...	LOADED	99747	99747	1	0	NULL	NULL	NULL
7	s3://snowflak...	LOADED	144097	144097	1	0	NULL	NULL	NULL

4.2.3 ロードが完了したら、ワークシートの右下で、「Open History」テキストの横にある小さな矢印をクリックし、特定のワークシートで実行された Snowflake 操作の履歴を表示します。

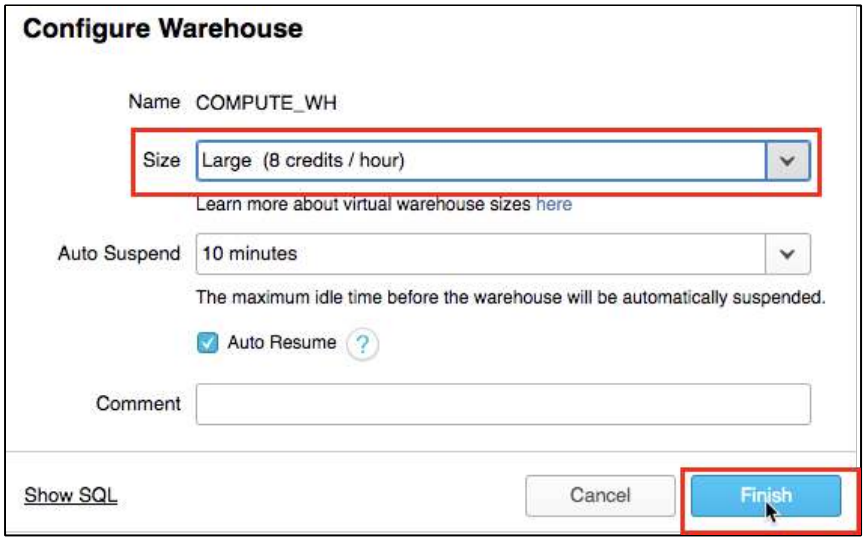


4.2.4 [履歴]ウィンドウで、「copy into trips from @ citibike_trips file_format = CSV;」を参照してください。実行した SQL クエリ、実行時間、スキャンされたバイト数、行を確認します。必要に応じてペインの左側にあるスライダを使用し、ペインを展開しすべての詳細を表示します。

4.2.5 ワークシートに戻り、すべてのテーブルデータおよびロードのメタデータをクリアするために、TRUNCATE TABLE コマンドを実行します。

```
truncate table trips;
```

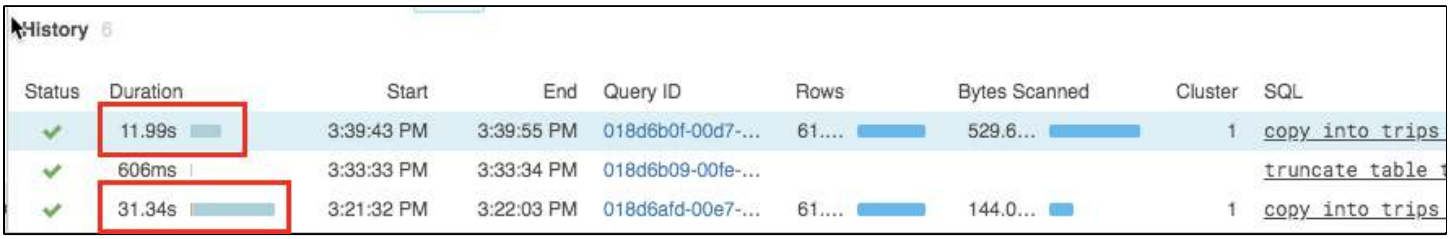
4.2.6 UI の上部にある[Warehouses]タブに移動し、[Configure...]をクリックします。Warehouse のサイズを[Large]に変更し、[Finish]をクリックします。この Warehouse は、Small サイズの 4 倍です。



4.2.7 UI の上部にある[Worksheets]タブに戻り、クリックします。ワークシートで次のステートメントを実行し、同じデータを再度ロードします。

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

4.2.8 ロードが完了したら、ワークシートの下部にある履歴ウィンドウの 2 つのロード間の実行時間を比較します。Large サイズの Warehouse が非常に高速な処理をしたことがわかります。



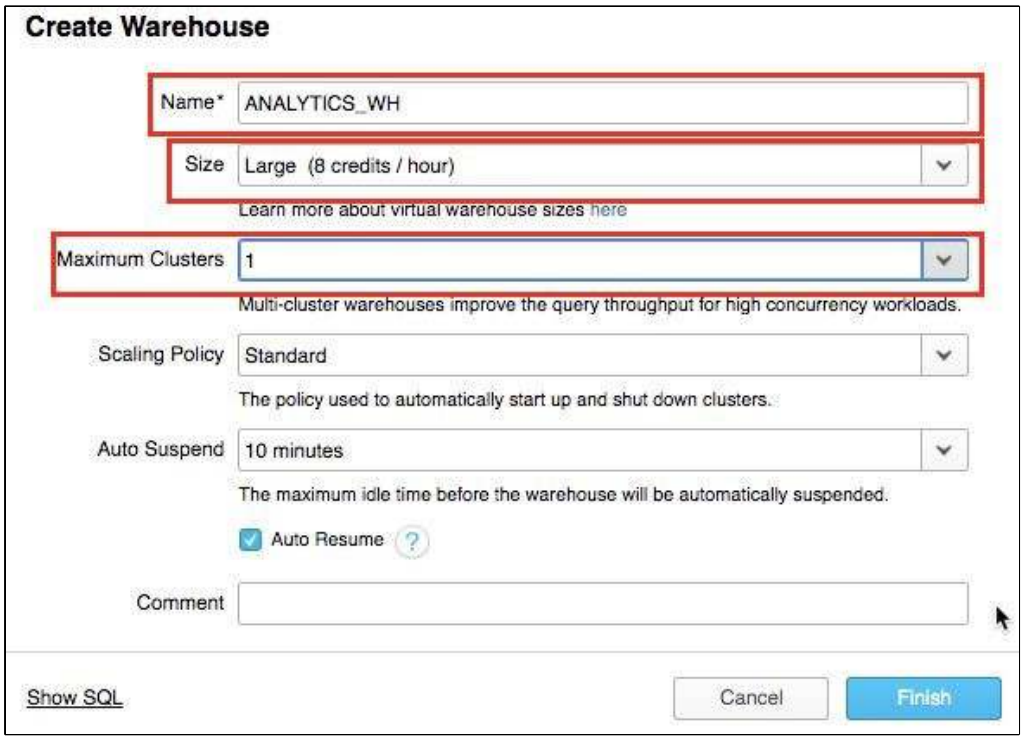
4.3 データ分析用の新しいウェアハウスを作成する

ラボのストーリーに戻ります。Citi Bike チームは、データロード/ ETL ワークロードと BI ツールを使用するときに Snowflake をクエリする分析エンドユーザとの間でリソースの競合がないことを確認したいと思っています。前述のように、Snowflake は、適切なサイズの異なるウェアハウスをさまざまなワークロードに割り当てることで、これを実現できま

す。Citi Bike はすでにデータをロードするためのウェアハウスを持っているので、分析を実行するエンドユーザ用の新しいウェアハウスを作成しましょう。次のモジュールで、分析を実行するために、このウェアハウスを使用します。

4.3.1 UI の上部にある[Warehouses]タブをクリックし、[Create...]をクリックします。サイズを「Large」にして「ANALYTICS_WH」という名前を付けます。Snowflake エディションの Enterprise 以上を使用している場合は、「Maximum Clusters」の設定が表示されます。これを「1」に設定します。

他の設定はデフォルト設定のままにします。設定は次のようになります。



次に、「Finish」ボタンをクリックしてウェアハウスを作成します。

モジュール 5：分析クエリ、結果キャッシュ、クローニング

前のハンズオンでは、Snowflake のバルクローダ（COPY コマンド）とウェアハウス COMPUTE_WH を使用し、テーブルにデータをロードしました。ここでは、2 番目のウェアハウス ANALYTICS_WH を使用し、これらのテーブルのデータをクエリする必要がある Citi Bike の分析ユーザの仕事を確認します。

現実世界のロールとクエリ

「現実世界」では、分析ユーザはおそらく SYSADMIN とは異なる役割を持ちます。ラボをシンプルに保つため、このモジュールでは SYSADMIN ロールを使用します。

また、「現実世界」では、ビジネスユーザは Tableau、Looker、PowerBI などの BI 製品を使いクエリをするでしょう。あるいは、Spark や R などのより高度な分析、データ・サイエンスの製品を使い Snowflake をクエリするでしょう。基本的に、JDBC / ODBC を活用するテクノロジはすべて、Snowflake のデータに対してクエリを実行できます。このラボでは、シンプルにするために、すべてのクエリは Snowflake ワークシートを介して実行されています。

5.1 SELECT ステートメントと結果キャッシュの実行

5.1.1 「Worksheets」タブに移動します。ワークシート内で、コンテキストを適切に設定してください。

Role ： SYSADMIN
Warehouse ： ANALYTICS_WH（L）
Database ： CITIBIKE
Schema：PUBLIC

5.1.2 以下のクエリを実行して、トラベルデータのサンプルを表示します。

```
select * from trips limit 20;
```

Results Data Preview

Query ID SQL 4.05s 20 rows

Filter result... Copy

Row	TRIPDURATION	STARTTIME	STOPTIME	START_STATION...	START_STATION_NAME	START_STATION_LATITUDE
1	518	2016-09-18 ...	2016-09-18 ...	3108	Nassau Ave & Russell St	40.72557
14	1898	2016-09-18 ...	2016-09-18 ...	3345	Madison Ave & E 99 St	40.789485416
13	1231	2016-09-18 ...	2016-09-18 ...	3168	Central Park West & W 8...	40.78472675
8	510	2016-09-18 ...	2016-09-18 ...	3160	Central Park West & W 7...	40.77896784
17	847	2016-09-18 ...	2016-09-18 ...	3423	West Drive & Prospect P...	40.661063372

5.1.3 最初に、Citi Bike の利用に関する基本的な 1 時間ごとの統計を見てみましょう。ワークシートで以下のクエリを実行します。1 時間ごとの、トラベル数、平均トラベル時間、および平均トラベル距離が表示されます。

```
select date_trunc('hour', starttime) as "date",
count(*) as "num trips",
avg(tripduration)/60 as "avg duration
(mins)",
avg(haversine(start_station_latitude, start_station_longitude,
                    end_station_latitude, end_station_longitude)) as "avg distance
                    (km)"
from trips
 group by 1 order by 1;
```

Results Data Preview Open History

Query ID SQL 984ms 44,295 rows

Filter result... Copy Columns

Row	date	num trips	avg duration (mins)	avg distance (km)
1	2013-06-01 00:00:00.000	152	56.058442983333	2.127971476
2	2013-06-01 01:00:00.000	102	26.525163400000	2.067906273
3	2013-06-01 02:00:00.000	67	36.119900500000	2.31784827
4	2013-06-01 03:00:00.000	41	44.485365850000	2.349126632
5	2013-06-01 04:00:00.000	16	23.278125000000	1.840026007

5.1.4 Snowflake には、過去 24 時間に実行されたすべてのクエリの結果を保持する結果キャッシュの仕組みがあります。これらはウェアハウス全体で利用できるため、1 人のユーザーに返されたクエリ結果は、同じクエリを実行するシステム上の他のユーザーが利用でき、基礎なるデータは変更されません。同じクエリは非常に高速に返されるだけでなく、計算クレジットも消費されません。

まったく同じクエリを再度実行し、実行中の結果キャッシュを見てみましょう。

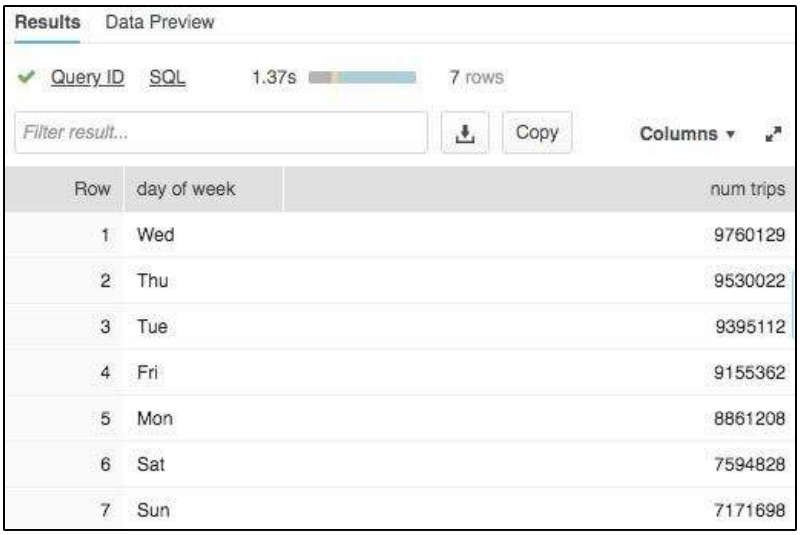
```
select date_trunc('hour', starttime) as "date",
count(*) as "num trips",
avg(tripduration)/60 as "avg duration
(mins)",
avg(haversine(start_station_latitude, start_station_longitude,
                    end_station_latitude, end_station_longitude)) as "avg distance
                    (km)"
from trips
group by 1 order by 1;
```

履歴ウィンドウで、結果がキャッシュされているため、クエリの実行速度が大幅に向上していることに注目ください。

5.1.5 次のクエリを実行し、最も忙しい曜日を確認します。

```
select
    dayname(starttime) as "day of week",
    count(*) as "num trips"
from trips
```

```
group by 1 order by 2 desc;
```



Results | Data Preview

Query ID | SQL | 1.37s | 7 rows

Filter result... | Copy | Columns

Row	day of week	num trips
1	Wed	9760129
2	Thu	9530022
3	Tue	9395112
4	Fri	9155362
5	Mon	8861208
6	Sat	7594828
7	Sun	7171698

5.2 テーブルのクローン

Snowflake を使用すると、テーブル、スキーマ、データベースの「ゼロコピークローン」とも呼ばれるクローンを数秒で作成できます。ソースオブジェクトに存在するデータのスナップショットは、クローンの作成時に取得され、クローンオブジェクトとして使用できるようになります。クローンされたオブジェクトは書き込み可能であり、ソースオブジェクトから独立しています。つまり、ソースオブジェクトまたはクローンオブジェクトのいずれかに加えられた変更は、他方に影響を与えません。

ゼロコピークローニングの一般的な使用例は、（1）本番環境に影響を与えず、（2）本番環境と開発テスト環境の 2 つのセットアップと管理の必要性を排除、することです。

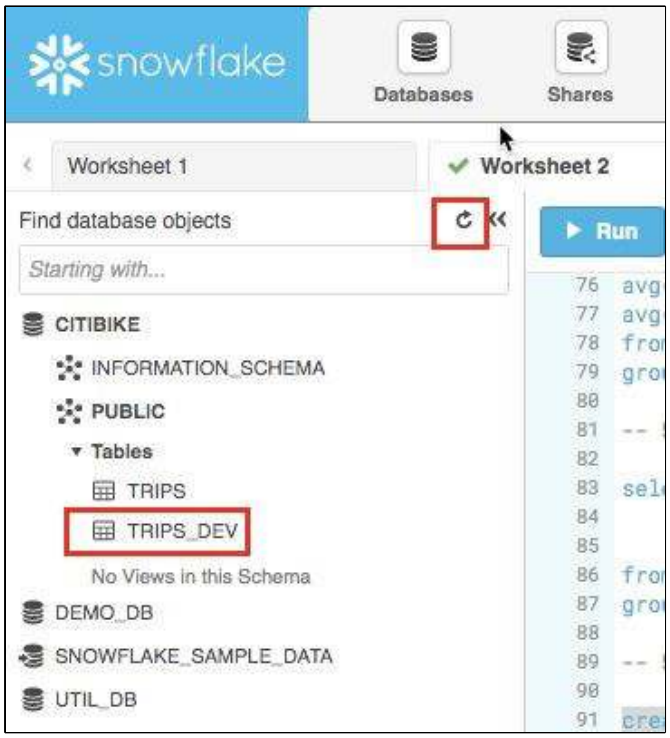
ゼロコピークローニング

大きな利点は、ソースとなるデータがコピーされないことです。ソースになるデータへのメタデータ/ポインターのみが変更されます。したがって、データのクローン作成時にストレージ要件が 2 倍になることはありません。ほとんどのデータウェアハウスではこれを行えません。Snowflake の場合は簡単にできます！

5.2.1 ワークシートで次のコマンドを実行し、開発（dev）用のテーブルを作成します

```
create table trips_dev clone trips
```

5.2.2 閉じている場合は、ワークシートの左側にあるデータベースオブジェクトブラウザを展開します。左側のパネルにある小さな[更新]ボタンをクリックし、CITIBIKE データベースの下のオブジェクトツリーを展開します。CITIBIKE データベースの下に TRIPS_DEV という名前の新しいテーブルが表示されることを確認します。開発チームは、TRIPS テーブルや他のオブジェクトに影響を与えることなく、このテーブルを使用して、削除することも含め、必要なことを行うことができます。



モジュール 6：半構造化データ、ビュー、JOIN の操作

注 - ここでの最初の手順は、以前のモジュール 3（データをロードする準備）および 4（データのロード）と似ていますが、時間を節約するために、UI ではなくワークシートの SQL を使用して行います。

ラボの「ストーリ」に戻ると、Citi Bike の分析チームは、天候が乗車回数に与える影響を確認したいと考えています。これを行うには、このモジュールで次のことを行います。

- S3 バケットに保持されている JSON 形式の気象データをロードします
- ビューを作成し、SQL ドット表記を使用して半構造化データをクエリします
- JSON データをこのガイドの前のモジュールの TRIPS データに結合するクエリを実行します
- 天気がトラベルカウントに与える影響を確認します

気象情報で構成された JSON データは OpenWeatherMap によって提供され、2016 年 7 月 5 日から 2019 年 6 月 25 日までのニューヨーク市の気象情報を提供します。AWS S3 にステージングされた気象情報は、57,900 行、61 個のオブジェクト及び圧縮された合計 2.5 メガバイトサイズとなります。

GZ ファイル内の JSON をテキストエディタで見ると次のようになります。


```
{"city":{"coord":{"lat":43.000351,"lon":-75.499901},"country":"US","findname":"NEW YORK","id":5128638,"name":"New York","zoom":
1},"clouds":{"all":90},"main":{"humidity":93,"pressure":1008,"temp":293.47,"temp_max":295.37,"temp_min":292.04},"time":
1561467737,"weather":[{"description":"moderate rain","icon":"10d","id":501,"main":"Rain"}],"wind":{"deg":170,"speed":4.1}}
{"city":{"coord":{"lat":40.714272,"lon":-74.005966},"country":"US","findname":"NEW YORK","id":5128581,"name":"New York","zoom":
1},"clouds":{"all":90},"main":{"humidity":94,"pressure":1010,"temp":295.16,"temp_max":296.15,"temp_min":294.15},"time":
1561467737,"weather":[{"description":"light rain","icon":"10d","id":500,"main":"Rain"},{"description":"mist","icon":"50d","id":
701,"main":"Mist"}],"wind":{"deg":0,"speed":2.1}}
{"city":{"coord":{"lat":43.000351,"lon":-75.499901},"country":"US","findname":"NEW YORK","id":5128638,"name":"New York","zoom":
1},"clouds":{"all":90},"main":{"humidity":94,"pressure":1008,"temp":294.58,"temp_max":297.04,"temp_min":292.04},"time":
1561471336,"weather":[{"description":"overcast clouds","icon":"04d","id":804,"main":"Clouds"}],"wind":{"deg":270,"speed":3.1}}
{"city":{"coord":{"lat":40.714272,"lon":-74.005966},"country":"US","findname":"NEW YORK","id":5128581,"name":"New York","zoom":
1},"clouds":{"all":90},"main":{"humidity":100,"pressure":1010,"temp":295.37,"temp_max":296.48,"temp_min":294.26},"time":
1561471336,"weather":[{"description":"mist","icon":"50d","id":701,"main":"Mist"}],"wind":{"deg":170.797,"speed":0.4}}
```


半構造化データ

Snowflake は、事前処理することなく、JSON、Parquet、Avro、ORC、XML などの半構造化データをそのままロードして参照できます。これは重要です。なぜなら、現在生成されるビジネス関連の半構造化データの量は増え続けているからです。

多くの従来型データウェアハウスは、半構造化データを簡単にロードおよび参照できません。Snowflake は半構造化データのキーバリューを解析し、カラム形式で最適に圧縮保存するため、高速なクエリを実現します。

6.1 データベースとテーブルの作成

6.1.1 ワークシートを使用して、半構造化データを保存する WEATHER というデータベースを作成します。

```
create database weather;
```

6.1.2 ワークシート内でコンテキストを適切に設定します。

```
use role sysadmin; use
warehouse compute_wh;
use database weather;
use schema public;
```

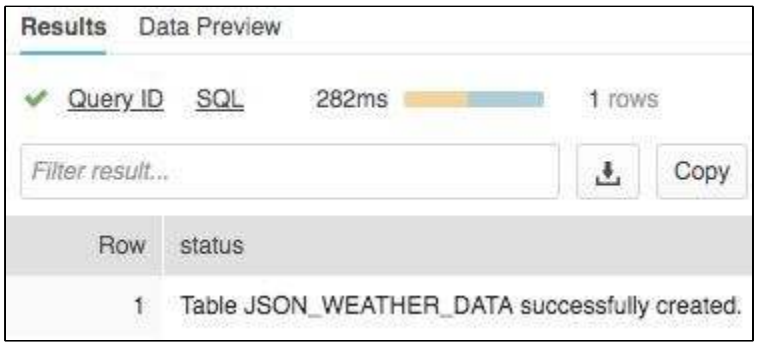
6.1.3 ワークシートを使用して、JSON データのロードに使用される JSON_WEATHER_DATA というテーブルを作成しましょう。ワークシートで、以下の SQL テキストを実行します。Snowflake には VARIANT と呼ばれる特別な列タイプがあり、JSON オブジェクト全体を保存し、直接クエリをすることができます。

```
create table json_weather_data (v variant);
```

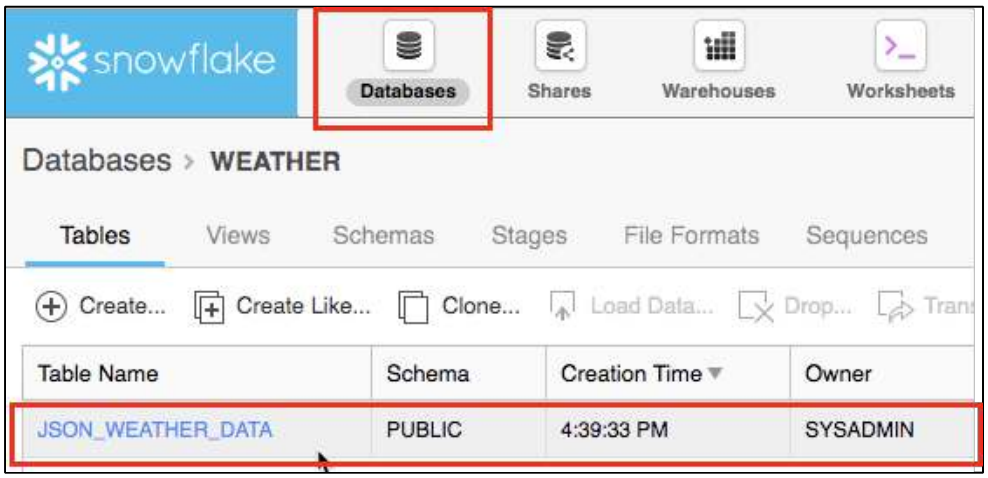
半構造化データをそのまま取り込める秘密

Snowflake の VARIANT データ型は、Snowflake が半構造化用スキーマを事前定義することなくデータを格納できます。

6.1.4 テーブル JSON_WEATHER_DATA が作成されたことを確認します。ワークシートの下部に、「Table JSON_WEATHER_DATA successfully created」という「Results」セクションが表示されます



6.1.5 ページの上部で、[Databases]タブに移動し、[Weather] データベースリンクをクリックします。新しく作成された JSON_WEATHER_DATA テーブルが表示されます。



6.2 外部ステージを作成する

6.2.1 ワークシートを介して、半構造化データが保存される AWS S3 ステージを作成します。

```
create stage nyc_weather
url = 's3://snowflake-workshop-lab/weather-nyc';
```

6.2.2 つぎに、nyc_weather ステージの内容を見てみましょう。ページの上部にある「Worksheets」タブをクリックします。ワークシートで、LIST コマンドを使用し次のステートメントを実行し、ファイルのリストを表示します。

```
list @nyc_weather;
```

S3 内の複数 gz ファイルを示す出力が下部ペインの結果ウィンドウに表示されるはずです。

Row	name	size	md5	last_modified
1	s3://snowflake-workshop-lab/weather...	40905	79638b8890d72e7d9bae14e3db6d28...	Tue, 25 Jun 2019 21:40:55 GMT
2	s3://snowflake-workshop-lab/weather...	42150	76fcd16208b4ca385fbbafe6fedff0b5	Tue, 25 Jun 2019 21:40:55 GMT
3	s3://snowflake-workshop-lab/weather...	43130	57ee5eae2ff354f9fffe3e3740f9b2c3	Tue, 25 Jun 2019 21:40:55 GMT
4	s3://snowflake-workshop-lab/weather...	42132	cfc162be5002f95f71999b287608fb72	Tue, 25 Jun 2019 21:40:55 GMT
5	s3://snowflake-workshop-lab/weather...	80842	ea4b3e62c759f610c9f4650520191 0f1	Tue, 25 Jun 2019 21:40:56 GMT

6.3 非構造化データのロードと検証

このセクションでは、ウェアハウスを使用し、作成した Snowflake テーブルに S3 バケットからデータをロードします。

6.3.1 ワークシートを介して、COPY コマンドを実行して、前に作成した JSON_WEATHER_DATA テーブルにデータをロードします。

ここの SQL で、FILE FORMAT オブジェクトをインラインで指定する方法を利用します。構造化データをロードした以前のモジュールでは、ファイルフォーマットの詳細を定義する必要がありました。ただし、この JSON データは適切にフォーマットされているため、デフォルト設定を使用し、JSON タイプを指定します。

```
copy into json_weather_data
from @nyc_weather
file_format = (type=json);
```

6.3.2 ロードされたデータを確認します。

```
select * from json_weather_data limit 10;
```

6.3.3 値のいずれかをクリックします。JSON 形式で生データがどのように保存されるているのかを確認します。完了したら「Done」をクリックします。

6.4 ビューの作成と半構造化データのクエリ

Snowflake を使用しビューを作成し、さらに SQL を使用し JSON データを直接クエリする方法を見てみましょう。

ビューとマテリアライズドビュー

ビューを使用すると、クエリの結果をテーブルのようにアクセスできます。ビューはデータをよりきれいなフォーマットでエンドユーザに提供します（このラボのように JSON フォーマットをカラムナ形式で提供します）、また、エンドユーザがプライバシー/セキュリティ上の理由によりソーステーブルから表示できるものを制限します。また、モジュール化された SQL を書くためにも利用します。

SQL の結果が保存されているマテリアライズドビューもあります。ほとんどの場合、結果はテーブルであるかのように振る舞います。これにより、アクセスが高速になりますが、ストレージスペースが必要となります。マテリアライズドビューを使うには、Snowflake Enterprise Edition 以上が必要です。

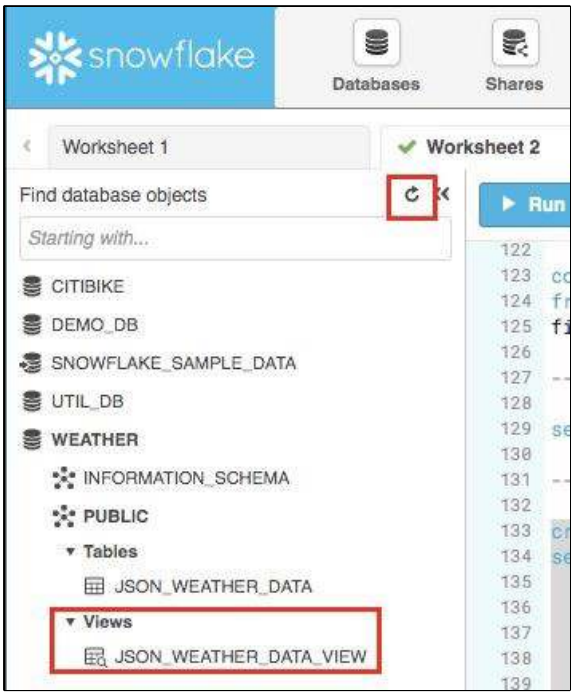
6.4.1 [Worksheets]タブから、ワークシートに移動し、次のコマンドを実行します。カラムナビューで非構造化 JSON 気象データのビューを作成するため、アナリストが理解し、クエリしやすくなります。ニューヨーク市の都市 ID5128638 を使います。

```
create view json_weather_data_view as
select
v:time::timestamp as observation_time,
v:city.id::int as city_id,
```

```
v:city.name::string as city_name,
v:city.country::string as country,
v:city.coord.lat::float as city_lat,
v:city.coord.lon::float as city_lon,
v:clouds.all::int as clouds,
(v:main.temp::float)-273.15 as temp_avg,
(v:main.temp_min::float)-273.15 as temp_min,
(v:main.temp_max::float)-273.15 as temp_max,
v:weather[0].main::string as weather,
v:weather[0].description::string as weather_desc,
v:weather[0].icon::string as weather_icon,
v:wind.deg::float as wind_dir,
v:wind.speed::float as wind_speed
from json_weather_data
where city_id = 5128638;
```

6.4.2 前のステップでは、SQL ドット表記（v.city.coo rd.lat ）方法を使い JSON 階層のより低いレベルでの値を引き出しました。これにより、各フィールドをリレーショナルテーブルの列であるかのように扱うことができます。

6.4.3 json_weather_data テーブルのすぐ下に新しいビューが表示されるのを確認します。表示するには、データベースオブジェクトブラウザを展開および/または更新する必要があります。



6.4.4 ワークシートを使用し、次のクエリでビューを確認します。通常の構造化データソースのように結果が見えることに注目ください（注 - 結果セットの observation_time 値が異なる場合があります。）

```
select * from json_weather_data_view
where date_trunc('month',observation_time) = '2018-01-01'  limit 20;
```

Results　Data Preview

Query ID　SQL　352ms　20 rows

Filter result...　Copy

Row	OBSERVATION_TIME	CITY_ID	CITY_NAME	COUNTRY	CITY_LAT	CITY_LON	CLOUDS
1	2018-01-03 01:02:38.000	5128638	New York	US	43.000351	-75.499901	90
2	2018-01-03 02:05:20.000	5128638	New York	US	43.000351	-75.499901	90
3	2018-01-15 13:04:58.000	5128638	New York	US	43.000351	-75.499901	1
4	2018-01-15 14:03:24.000	5128638	New York	US	43.000351	-75.499901	1
5	2018-01-14 04:02:46.000	5128638	New York	US	43.000351	-75.499901	1
6	2018-01-14 05:04:41.000	5128638	New York	US	43.000351	-75.499901	1

6.5 JOIN 操作を使用しデータセットを相関させる

ここで、JSON 天気データを CITIBIKE.PUBLIC.TRIPS データと結合して、天気が乗車回数にどのように作用するかという質問に対する答えを確認します。

6.5.1 以下のコマンドを実行し、WEATHER と TRIPS を JOIN させ、特定の気象条件に関連するトラベル数をカウントします。

注　-　まだワークシートにデフォルトとして WEATHER データベースを使用しているため、データベースとスキーマ名を指定し、完全に修飾された TRIPS テーブル名を参照します。

```
select weather as conditions ,count(*) as num_trips from
 citibike.public.trips
 left outer join json_weather_data_view
    on date_trunc('hour', observation_time) = date_trunc('hour', starttime)
 where conditions is not null
 group by 1 order by 2 desc;
```

Results | Data Preview

Query ID | SQL | 435ms | 10 rows

Filter result... | Copy | Columns

Row	CONDITIONS	NUM_TRIPS
1	Clear	9249531
2	Clouds	8934964
3	Rain	3206720
4	Snow	1380418
5	Mist	798487
6	Fog	362496
7	Thunderstorm	230539
8	Drizzle	98779
9	Haze	40724
10	Smoke	2871

6.5.2 Citi Bike チームの当初目標は、ライダーと気象の両方のデータを分析することにより、自転車のトラベル回数と天候の間に相関関係があるかどうかを確認することでした。上記の結果に従うと、明確な答えが出たことになります。ご想像のとおり、天気が良いときはトラベル回数が大幅に増えています！

残りのラボでは、Snowflake の他機能について見ていきましょう！

モジュール 7：タイムトラベルの使用

Snowflake のタイムトラベル機能を使用すると、事前設定可能な期間内の任意の時点で履歴データにアクセスできます。デフォルトの期間は 24 時間で、Snowflake Enterprise Edition では最大 90 日間です。ほとんどの他データウェアハウスでは、この機能を利用できません。Snowflake を使えば簡単に実現できます！

タイムトラベルを活用する有用なケースは次のとおりです。

- 誤ってまたは意図的に削除された可能性のあるデータ関連オブジェクト（テーブル、スキーマ、およびデータベース）の復元
- 過去の時点からのデータの複製とバックアップ
- 指定された期間にわたるデータ使用量/操作の分析

7.1 テーブルの DROP と UNDROP

まず、誤ってまたは意図的に削除されたデータオブジェクトを復元する方法を見てみましょう。

7.1.1 ワークシートから、json_weather_data テーブルをドロップ（削除）する次のコマンドを実行します。

```
drop table json_weather_data;
```

7.1.2 次に、json_weather_data テーブルで SELECT ステートメントを実行します。「Results」ペインには、ベースとなるテーブルが削除されたため、エラーが表示されます。

```
select * from json_weather_data limit 10;
```

7.1.3 テーブルを復元します。

```
undrop table json_weather_data;
```

7.1.4 json_weather_data テーブルが復元されました。

7.2 テーブルのロールバック

次に、テーブルを以前の状態にロールバックし、Citibike データベースの TRIPS テーブルにあるすべてのステーション名を「oops」という単語に置き換える意図しない DML エラーを修正します。

7.2.1 最初に、ワークシートが適切なコンテキストであることを確認します。

```
use role sysadmin; use
warehouse compute_wh;
use database citibike; use
schema public;
```

7.2.2 次に、テーブル内のすべてのステーション名を「oops」という単語に置き換える次のコマンドを実行します

```
update trips set start_station_name = 'oops';
```

7.2.3 ここで、乗車回数で上位 20 駅を返すクエリを実行します。駅名が間違っているため、1 行しか表示されません。

```
select
start_station_name as "station",
count(*) as "rides"
from trips
group by 1
order by 2 desc
```

```
limit 20;
```

7.2.4 このようなケースに遭遇した場合、通常、私たちは緊急対応する必要があり、バックアップがあることを望みます。ただし、Snowflake では、コマンドを実行するだけで、最後の UPDATE コマンドのクエリ ID を見つけ、$ QUERY_ID という変数に格納できます…

```
set query_id =
(select query_id from
table(information_schema.query_history_by_session(result_limit=>5))
where query_text like 'update%' order by start_time limit 1);
```

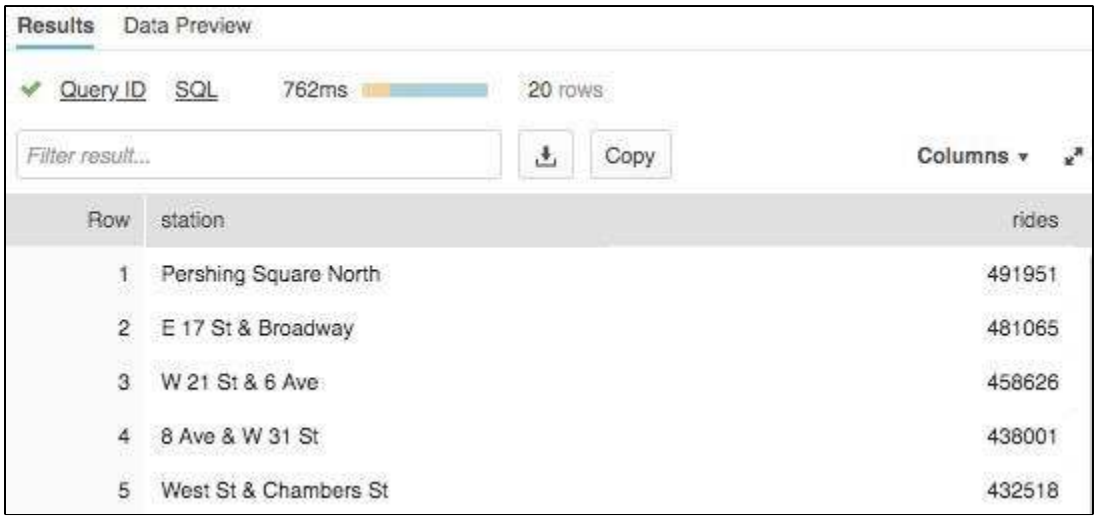
7.2.5 次に、更新前時点のテーブルを再作成します。

```
create or replace table trips as
(select * from trips before (statement => $query_id));
```

7.2.6 SELECT ステートメントを再度実行して、ステーション名が復元されたことを確認します。

```
select
start_station_name as "station",
count(*) as "rides"
from trips
group by 1
order by 2 desc
```

```
limit 20;
```



Results Data Preview

Query ID SQL 762ms 20 rows

Filter result... Copy Columns

Row	station	rides
1	Pershing Square North	491951
2	E 17 St & Broadway	481065
3	W 21 St & 6 Ave	458626
4	8 Ave & W 31 St	438001
5	West St & Chambers St	432518

モジュール 8：ロールベースアクセス制御とアカウント管理

このモジュールでは、新しいロールの作成や特定のアクセス許可の付与など、Snowflake ロールベースのアクセス制御（RBAC）のいくつかの側面を見て行きます。また、アカウント管理者ロール、別名 ACCOUNTADMIN についても確認します。

Citi Bike のストーリを続けます。新米 DBA が Citi Bike チームに参加したため、システム定義のデフォルトロール SYS ADMIN よりも少ない権限の新しいロールを作成し、付与します。

役割ベースのアクセス制御（RBAC）

Snowflake は、ロールまたはユーザがアクセスできるオブジェクトと機能、およびアクセスのレベルを制御を可能にする RBAC を提供します。

詳細については、https：//docs.snowflake.net/ma nuals / user-guide / security-access-control.html のドキュメントを参照してください。

8.1 新しいロールを作成しユーザーを追加する

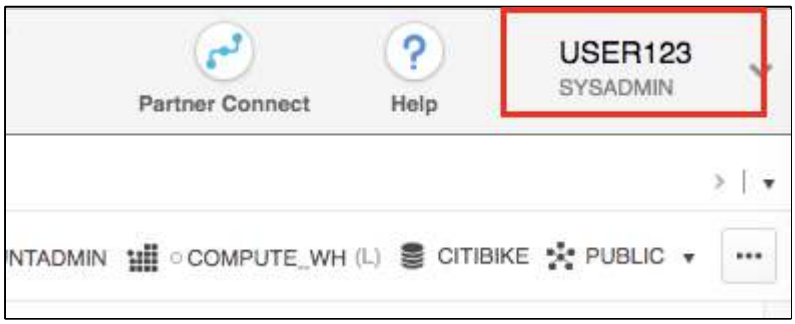
8.1.1 ワークシートで、ACCOUNTADMIN ロールに切り替えて新しいロールを作成しましょう。この役割は、SYSADMIN および SECURITYADMIN システム定義の役割を併せ持ちます。これはシステムの最上位の役割であり、アカウント内の限られた/制御された数のユーザにのみ付与する必要があります。ワークシートで、次を実行します。

```
use role accountadmin;
```

完了したら、ワークシートの右上にあるワークシートコンテキストが変更されているため、ロールは ACCOUNTADMIN になります。

8.1.2 ロールが機能するためには、少なくとも 1 人のユーザーが割り当てられている必要があります。「junior_dba」という新しいロールを作成し、ユーザ名を割り当てましょう。これは、30 日間のフリートライアル版 Snowflake アカウントを最初

に接続したときに作成したユーザー名です。この名前は、UI の右上にも表示されます。以下のスクリーンショットでは、「USER123」です。あなたが作成したユーザ名を書き留めます。



8.1.3 ここで、ロールを作成し、一意のユーザー名でユーザを追加しましょう。

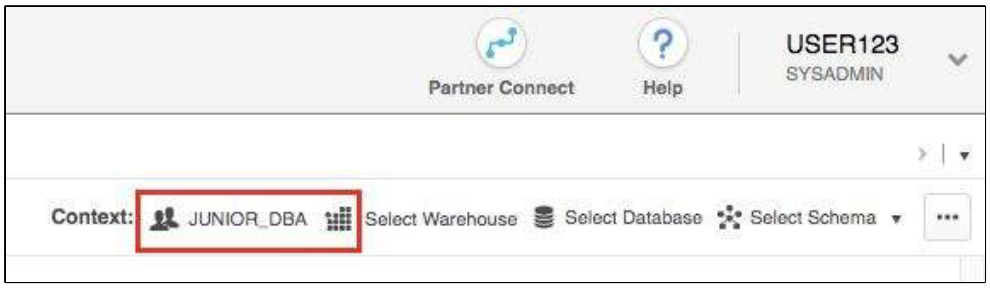
```
create role junior_dba;
grant role junior_dba to user YOUR_USER_NAME_GOES HERE;
```

注 - SYSADMIN などのロールでこの操作を実行しようとすると、デフォルトでは SYSADMIN ロールは新しいロールまたはユーザーを作成できず権限が不十分なため失敗します。

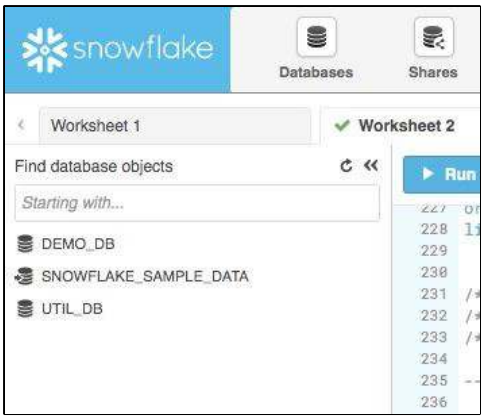
8.1.4 ワークシートのコンテキストを新しい junior_dba ロールに変更する

```
use role
junior_dba;
```

ワークシートの右上で、junior_dba ロールを反映するようにコンテキストが変更されていることにご注意ください



8.1.5 データベースオブジェクトブラウザペインの UI の左側に、Citibike データベースと Weather データベースの両方が表示されていないことにご注意ください。これは、junior_dba ロールにはそれらを表示するアクセス権がないためです。



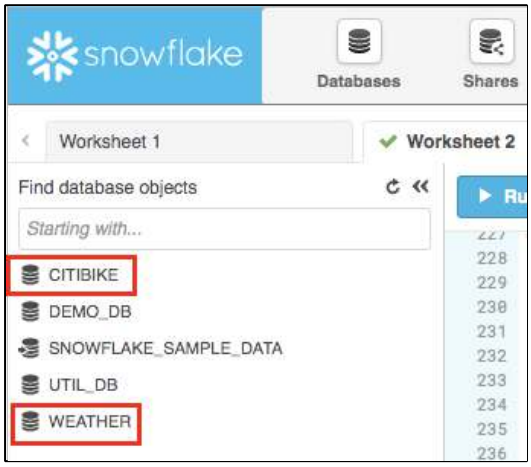
8.1.6 ACCOUNTADMIN ロールに戻り、junior_dba に CITIBIKE および WEATHER データベースを表示および使用する権限を付与しましょう

```
use role accountadmin;

grant usage on database citibike to role junior_dba;
grant usage on database weather to role junior_dba;
```

8.1.7 junior_dba ロールに切り替え、データベースオブジェクトブラウザの左側で、Citibike および Weather データベースが表示されることをご確認ください。表示されない場合は、更新アイコンをクリックします。

```
use role junior_dba;
```



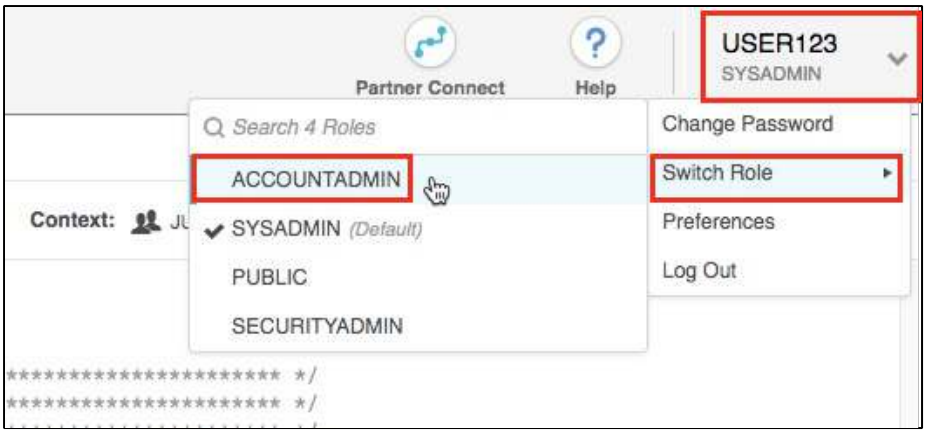
8.2 アカウント管理者ビュー

セッションのセキュリティロールを ACCOUNTADMIN に変更し、このロールのみが見ることができる UI の他の部分を見てみましょう。

ユーザー設定の役割とワークシートの役割

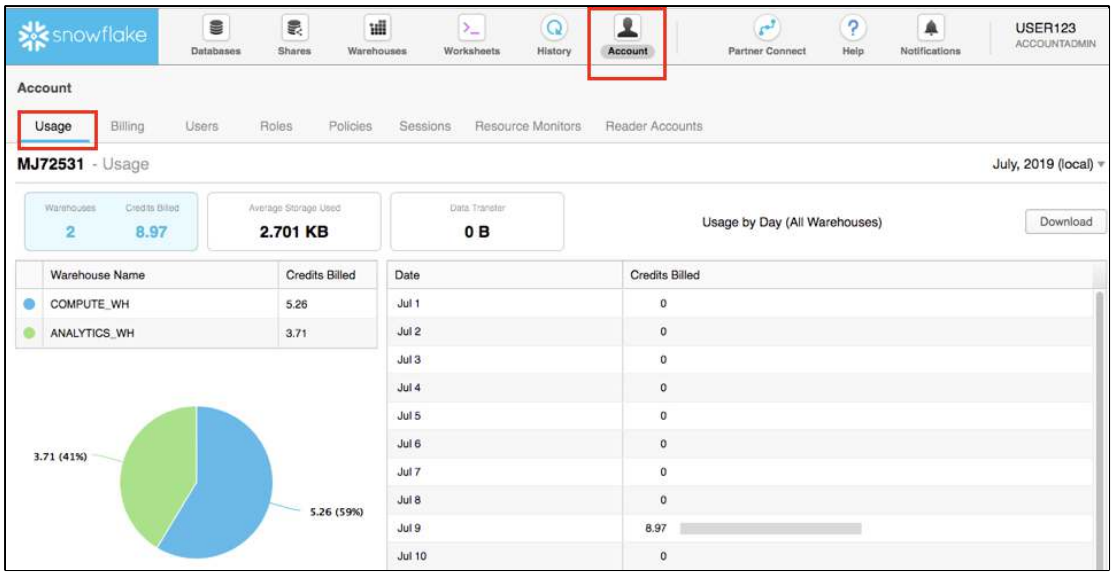
UI の右上にあるユーザ設定メニューで、セッションのセキュリティロールを変更しました。これにより、UI に表示されるものが変わります。これは異なるロールを割り当てるワークシートのコンテキストメニューとは異なります。ワークシートのロールは特定のワークシートで実行されるコマンドに適用されます。また、セッションセキュリティロールは、ワークシートで使用されるロールとは異なる場合があります。

8.2.1 UI の右上隅で、ユーザ名をクリックし[ユーザ設定]メニューを表示します。次に、[Switch Role]に移動し、ACCOUNTADMIN ロールを選択します。



.2.2 UI の最上部に、「Account」という 6 番目のタブが表示されます。このタブは、ACCOUNTADMIN ロールおよび SECURITYADMIN ロールでのみ表示できます。

[アカウント]タブをクリックします。次に、このページの上部に向かって、デフォルトですでに表示されている「Usage」をクリックします。ここでは、クレジット、ストレージ、および毎日の使用量の詳細が表示されます。



8.2.3 「Usage」の右側には「Billing」があり、このフリートライアルで利用できる無料分 400 ドル相当のクレジットを超えて利用したい場合は、クレジットカードを追加します。さらに右側には、ユーザ、ロール、およびリソースモニタに関する情報があります。リソースモニタは、アカウントのクレジット消費に制限を設定するため、クレジット消費を適切に監視および管理できます。

注 - 次のモジュールのために ACCOUNTADMIN ロールにとどまります。

モジュール 9：データシェア

Snowflake を介してアカウント間のデータシェアが可能になります。データプロバイダがシェアを作成し、プロバイダの Snowflake アカウントを介し、またはプロビジョニングされた Snowflake リーダアカウントのいずれかの方法でデータコンシューマにデータを提供します。コンシューマは、外部のエンティティ/パートナ、または独自の Snowflake アカウントを持っている別の内部ビジネスユニットである可能性があります。

データシェアの利点

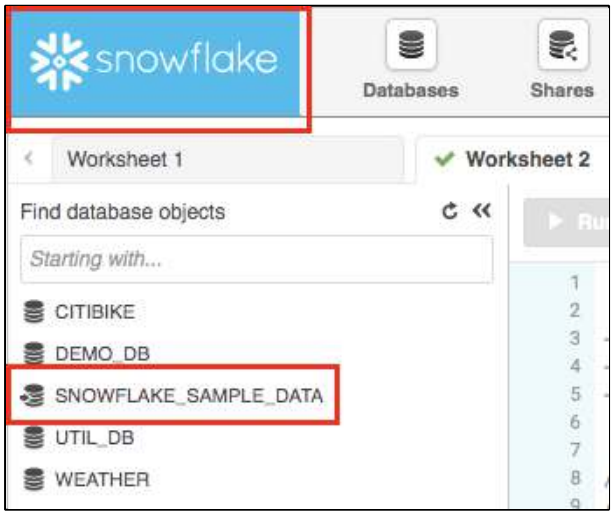
- データのコピーは 1 つのみであり、データプロバイダのアカウントに存在します
- 共有データは常にリアルタイムで、コンシューマがすぐに利用できます
- プロバイダは、シェアに対する取り消しを含めたきめ細かいアクセス許可を確立できます。
- データシェアは簡単で安全です。特に、データを共有する従来の方法と比較すると分かります。例えば、インターネット経由で大きな.csv ファイルを転送する必要がある。

注 - 現在、データシェアは Snowflake プロバイダと同じリージョンのアカウント間でのみサポートされています。

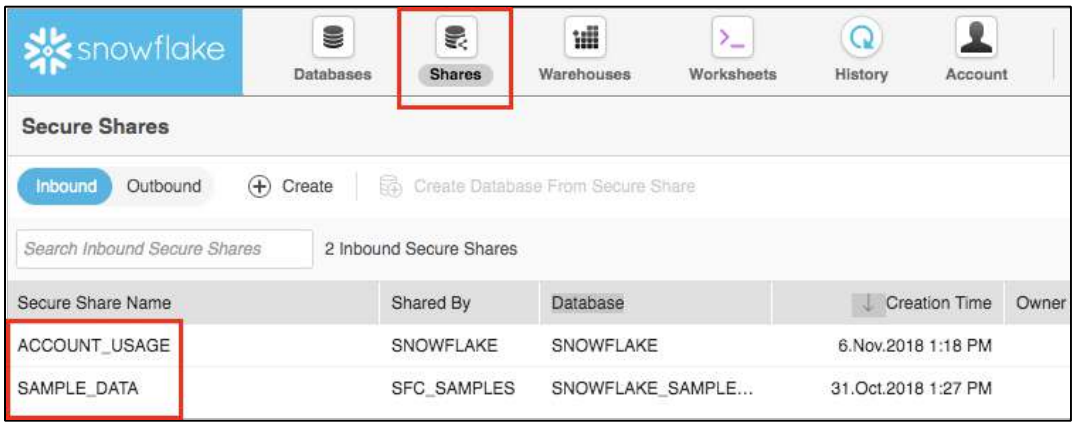
データシェアの例として、Snowflake が安全なデータシェアを使用することで、アカウント利用データとサンプルデータセットをすべての Snowflake アカウントと共有しています。Snowflake はデータのプロバイダとして機能し、他のすべてのアカウントはコンシューマとして機能します。Snowflake 環境で、これを簡単に確認できます。これについては、次のセクションで説明します。

9.1 既存のシェアを見る

9.1.1 UI の左上にある青い Snowflake ロゴをクリックします。データベースオブジェクトブラウザの UI の左側で、データベース「SNOWFLAKE_SAMPLE_DATA」をご確認ください。データベースアイコンの小さな矢印は、これが共有であることを示しています。



9.1.2 UI の右上で、ACCOUNTADMIN ロールに指定されていることを確認します。次に、UI の上部にある[Shares]タブをクリックします。このページでは、Inbound Secure Shares を表示しています。Snowflake がこのアカウントと共有している 2 つのシェアがあります。1 つにはアカウントの利用状況が含まれ、もう 1 つには利用できるサンプルデータが含まれています。これが実際のデータシェアです – プロビジョニングされたすべての Snowflake アカウントは、Snowflake が共有/提供するデータのコンシューマです！



9.2 アウトバウンドシェアを作成する

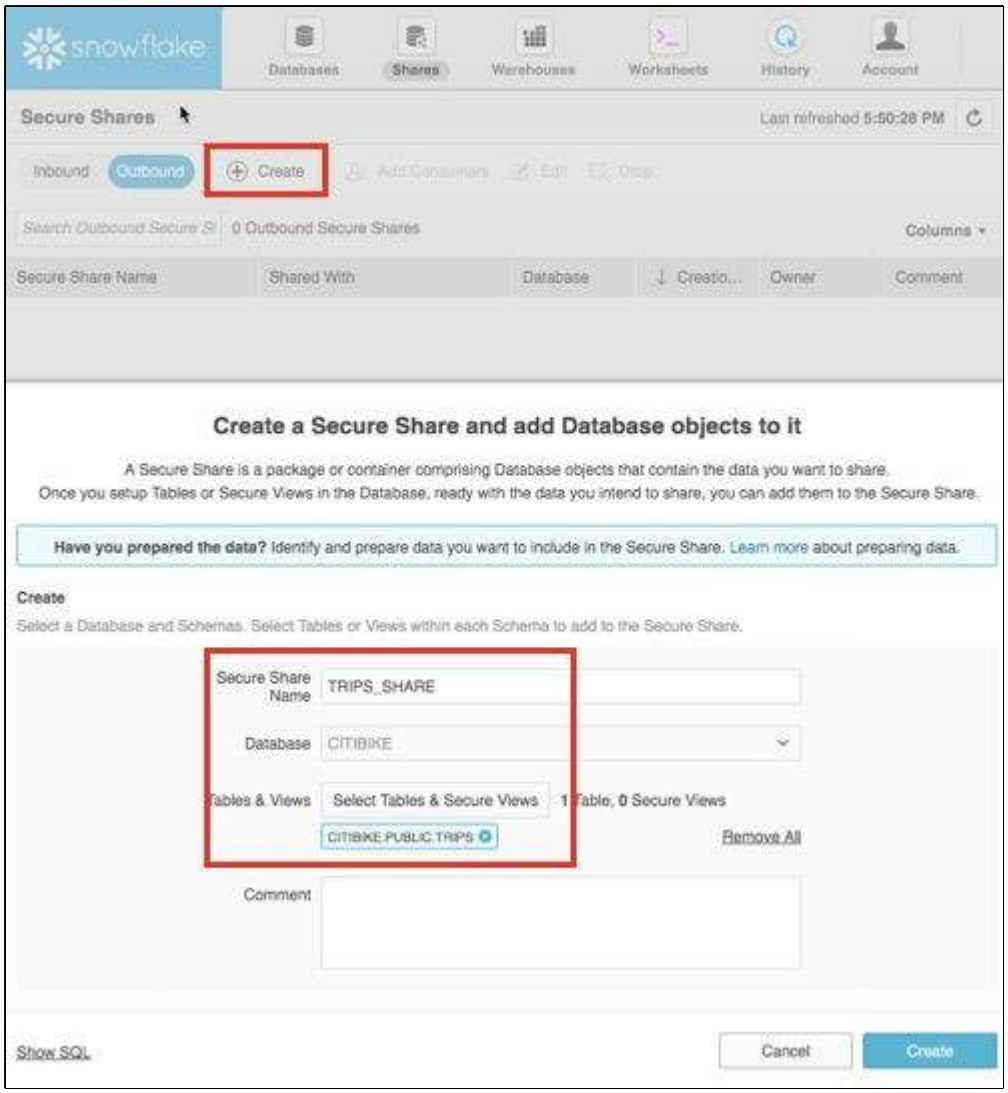
9.2.1 Citi Bike のストーリに戻り、私たちが Citi Bike の Snowflake アカウント管理者であると仮定します。我々の信頼出来るパートナが TRIPS データベース内のデータにリアルタイムでアクセスし、データサイセンスの分析をしたいと仮定します。このパートナは、当社と同じリージョンに独自の Snowflake アカウントも持っています。それでは、Snowflake データシェア機能を使用しこのデータを共有し、分析できるようにします。

UI 上部にある[Shares]タブをクリックします。次に、ページのさらに下にある[Outbound]ボタンをクリックします。

9.2.2 「Create」ボタンをクリックし、表示されるフィールドで、以下に示すように入力します。

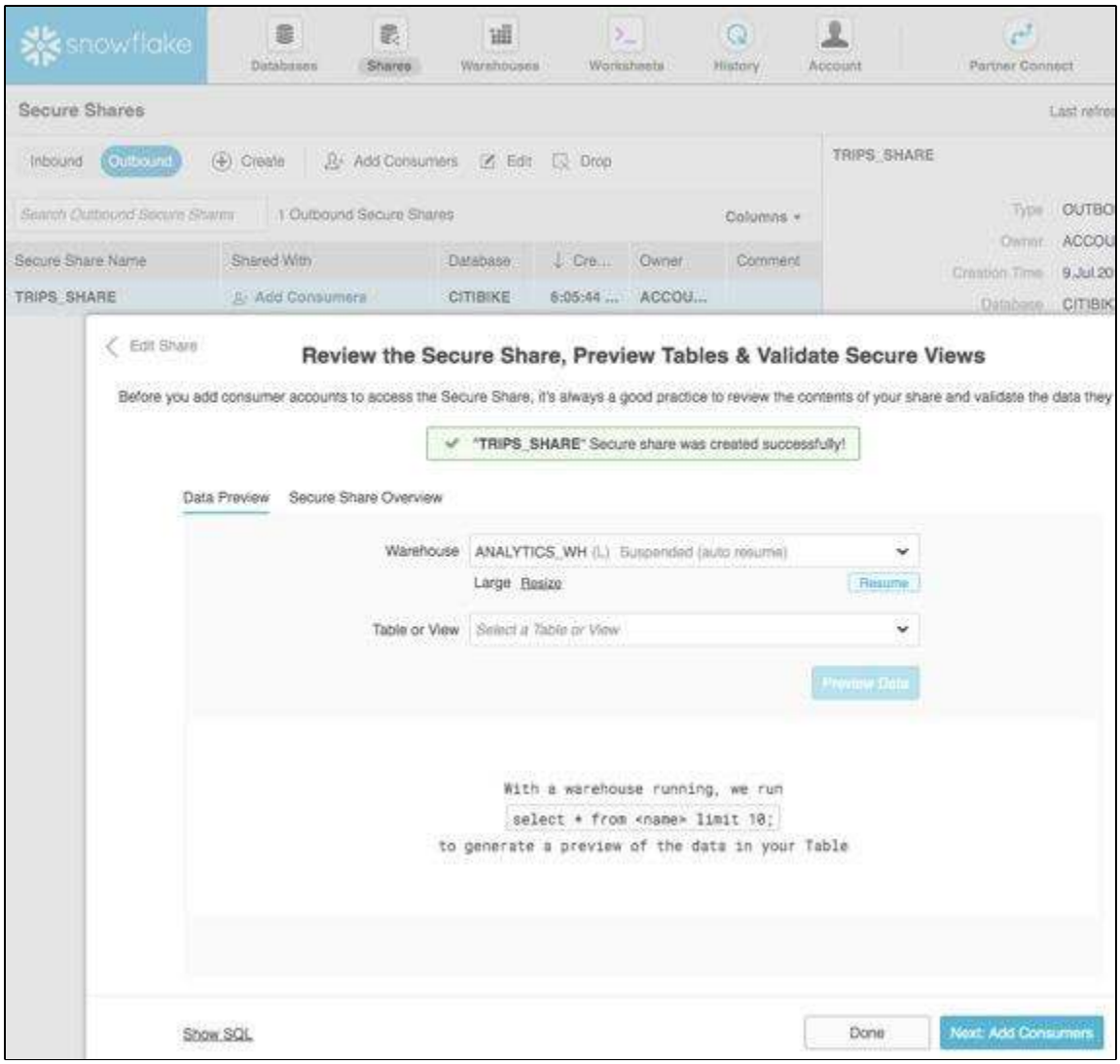
- Secure Share Name : 「TRIPS_SHARE」と入力します
- Database : ドロップダウンを使用し「CITIBIKE」を選択します
- Tables & Views : データベースオブジェクトブラウザを使用し、CITIBIKE> PUBLIC> TRIPS を参照します。

青い[Apply]ボタンをクリックします



9.2.3 ボックスの下部にある青い「Continue」ボタンをクリックします。

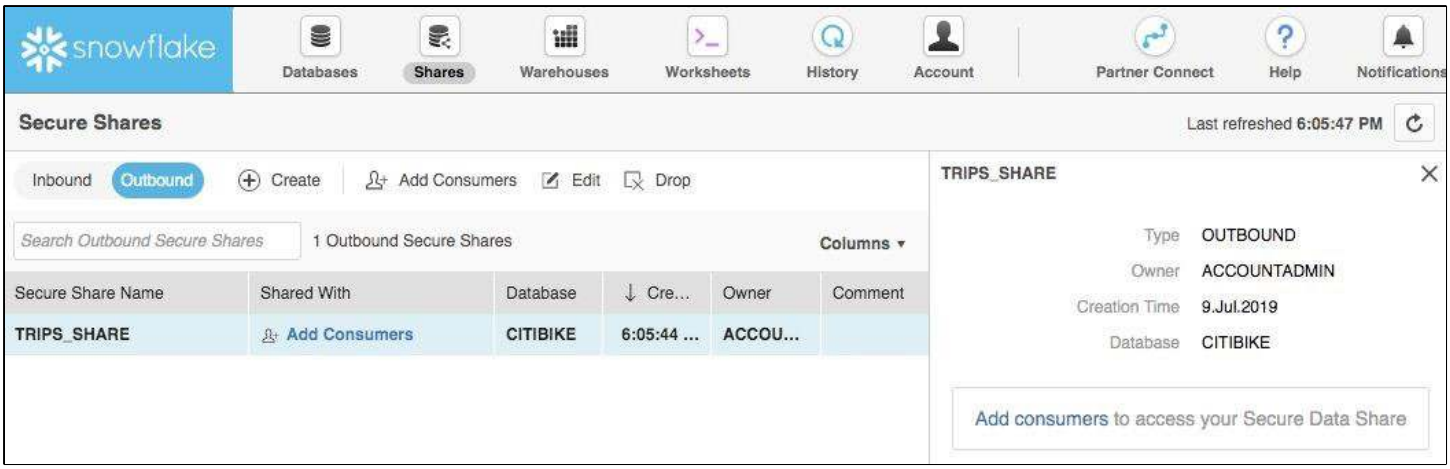
このウィンドウは、セキュアシェアが正常に作成されたことを示しています。



実際には、Citi Bike アカウント管理者は「Next Add Consumers」ボタンをクリックし、パートナーの Snowflake アカウント名などの情報を追加します。ただし、ラボでは自分のアカウントを使用しているだけなので、ここで停止します。

9.2.4 ボックスの下部にある「Done」ボタンをクリックします。

このページに、「TRIPS_SHARE」セキュアシェアが表示されるようになりました。データのコピーを作成する必要がないため、Snowflake のデータに他のアカウントが安全にアクセスできるようになるのに数秒しかかかりませんでした。



最後に、Snowflake には、機密性を損なうことなくデータを安全に共有するためのいくつかの方法が用意されています。テーブルとビューだけでなく、セキュアビュー、セキュア UDF（ユーザー定義関数）、およびセキュアジョインも共有できます。機密情報へのアクセスを防止しながら、データを共有するためのこれらの詳細については、Snowflake のマニュアルをご参照ください。

おめでとうございます、これでこのハンズオンラボは終了です！次の最後のセクションでまとめを見て行きましょう。

まとめと次のステップ

このラボでは、Snowflake の実践的なシナリオをもとに、Snowflake の使用方法を体験しながら、その主要な機能と差別化要因の一部を紹介しました。具体的には、UI のナビゲート、データベースとウェアハウスの作成、構造化および半構造化データのロードとクエリ、ゼロコピークローンの実行、ユーザエラーの取り消し、RBAC、データシェアの方法について説明しました。

より実データに近いサンプルまたは実稼働しているデータをロードし、このラボでは取り上げていない Snowflake の高度な機能を使用し、フリートライアルを継続してご利用されることをお勧めします。Snowflake からは以下の方法で支援可能です。

- UI の一番上にある[Partner Connect]アイコンをクリックし、パートナが提供するトライアル/無料の ETL および BI ツールに簡単に統合し、Snowflake を使い大量データを取得して分析できるようにします
- 次のサイトの「Definitive Guide to Maximizing Your Free Trial」ドキュメントをお読みください。https://www.snowflake.com/test-driving-snowflake-the-definitive-guide-to-maximizing-your-free trial /
- 当社の機能についての詳細を学ぶために Snowflake が提供するイベントにご参加頂き、Snowflake がどのようにお客様を支援できるかご確認ください。https://www.snowflake.com/about/events
- その他、弊社営業にお問い合わせください。https://www.snowflake.com/free-trial-contact-sales/

Snowflake 環境のリセット

最後に、この実習ラボの一部として作成されたすべてのオブジェクトを削除して環境をリセットする場合は、ワークシートで次の SQL を実行します。

この SQL を実行し、ワークシートのコンテキストを設定します。

```
use role accountadmin;
use warehouse
compute_wh; use
database weather;
use schema public;
```

次に、この SQL を実行して、ラボで作成したすべてのオブジェクトを削除します。

```
drop share if exists trips_share;
drop database if exists citibike;
drop database if exists weather;
drop warehouse if exists
analytics_wh; drop role if exists
junior_dba;
```