

URL : <https://bit.ly/48pn3vA>

Snowflake Partner Boot Camp

Hands-on Workbook

Snowflake Korea

본 워크북은 파트너사의 Snowflake 데이터 플랫폼 설계자, 관리자를 위해 핵심적인 기능들을 직접 실행시켜 보면서 중요한 컨셉과 피처를 이해하도록 하는 가이드입니다.

Features Deep Dive 세션을 통해서 Snowflake의 핵심적인 컨셉과 기능을 이해한 다음에 직접 핸즈온을 진행하면서 해당 토픽에 대한 이해도를 높이는 것을 목표로 하고 있습니다.

[Snowflake 30일 무료 평가판에 등록](#)해서 실습이 진행되며, 핸즈온이 진행되는 동안 [Snowflake Documentation\(한국어\)](#)을 참조하면서 각 토픽에 대한 이해도를 높이도록 권장 드립니다.

1. Course Setup

Trial 계정 등록 및 랩 환경 준비

[Snowflake 30일 무료 평가판](#)에서 Trial 계정을 등록합니다.

워크샵이 진행되는 동안 [Snowflake Documentation\(한국어\)](#)을 함께 참조하도록 합니다.

- Email : 비즈니스 이메일 계정을 등록하기 바랍니다.
- 클라우드 공급자, 리전 및 에디션: AWS의 Seoul 리전과 Enterprise 에디션을 선택하도록 권장합니다.

등록 후, 활성화 링크와 Snowflake 계정 URL이 담긴 이메일을 받게 되므로 스팸 메일로 분류되지 않았는지 확인합니다.

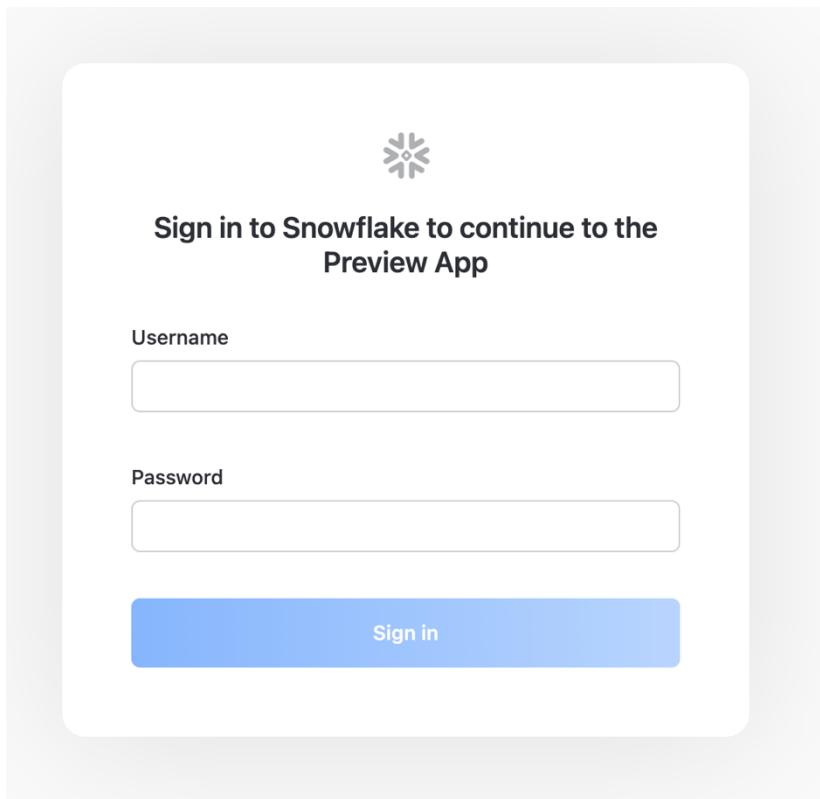
(3 ~ 5분 후에 링크가 활성화되는 경우도 있으므로 이 경우에는 조금 기다린 후에 활성화 링크를 클릭하고 로그인하시기 바랍니다.)

2. Snowflake User Interface (Snowsight)

Snowflake 사용자 인터페이스(UI)에 로그인

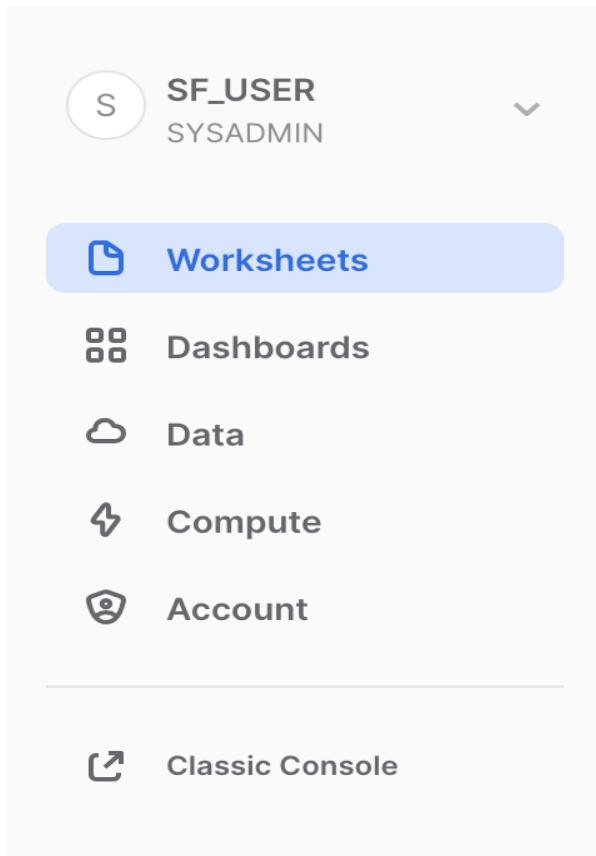
브라우저 창을 열고 등록 이메일에서 받은 Snowflake 30일 평가판 환경의 URL을 입력합니다.

아래 로그인 화면이 나타납니다. 등록에 사용한 사용자 이름 및 암호를 입력하세요.



Snowflake UI 확인

로그인 이후에 사용자 인터페이스를 확인합니다. 왼쪽의 대메뉴부터 위에서 아래로 이동하면서 알아 보도록 하겠습니다.

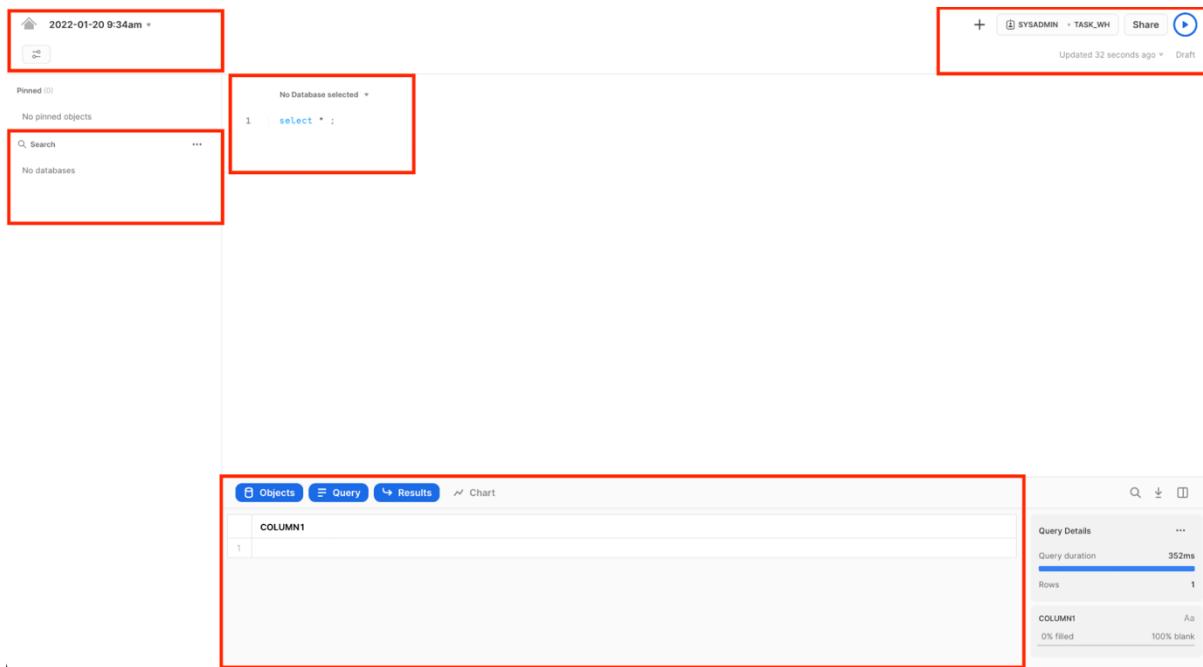


왼쪽의 대메뉴를 통해서 각 구성요소들을 살펴보도록 하겠습니다.

Worksheets

A screenshot of the 'Worksheets' page. The left sidebar is identical to the one above. The main area has a title 'Worksheets' and a sub-header 'Welcome to Worksheets'. It includes a search bar, a 'New folder' button, and a prominent blue '+ Worksheet' button. A red box highlights the '+ Worksheet' button. Below the header, there is a large blue button with a white '!' icon and the text 'Welcome to Worksheets' followed by a brief description of the features.

Worksheets 탭은 SQL 쿼리 실행, DDL 및 DML 작업 수행 그리고 쿼리 또는 작업 완료 시 결과 확인을 위한 인터페이스를 제공합니다. + **Worksheet** 버튼을 클릭하여 새로운 워크시트를 생성 할 수 있습니다.



왼쪽 상단 모서리에는 다음이 포함됩니다.

- **Home** 아이콘: 기본 콘솔로 돌아가거나 워크시트를 닫을 때 사용합니다.
- **Worksheet 이름** 드롭다운: 기본 이름은 워크시트가 생성된 타임스탬프입니다. 타임스탬프를 클릭하여 워크시트 이름을 편집합니다. 드롭다운에는 워크시트에 대해 수행할 수 있는 추가 작업도 표시됩니다.
- **필터 관리자** 버튼: 맞춤 필터는 하위 쿼리 또는 값 목록으로 해석되는 특수 키워드입니다.

오른쪽 상단 모서리에는 다음이 포함됩니다.

- + 버튼: 새 워크시트를 생성합니다.
- **Context** 상자: 해당 세션 동안 사용할 역할과 웨어하우스를 선택합니다. UI 또는 SQL 명령을 통해 변경할 수 있습니다.
- **Share** 버튼: 공유 메뉴를 열어 다른 사용자와 공유하거나 워크시트 링크를 복사합니다.
- **Play/Run** 버튼: 현재 커서가 있는 SQL 문 또는 선택한 여러 문을 실행합니다.

중간 창에는 다음이 포함됩니다.

- 워크시트에 대한 데이터베이스/스키마/개체 컨텍스트를 설정하기 위한 상단의 드롭다운.

- 쿼리 및 기타 SQL 문을 입력하고 실행하는 일반 작업 영역입니다.

가운데 왼쪽 패널에는 현재 워크시트에 사용 중인 역할이 액세스할 수 있는 모든 데 이터베이스, 스키마, 테이블 및 보기를 탐색할 수 있는 데이터베이스 개체 브라우저가 있습니다.

아래쪽 창에는 쿼리 및 기타 작업의 결과가 표시됩니다. 또한 UI에서 해당 패널을 열고 닫는 4가지 옵션(**Object, Query, Result, Chart**)이 포함됩니다. **Chart**는 반환된 결과에 대한 시각화 패널을 열 수 있습니다.

Dashboards

Dashboards 탭을 사용하면 하나 이상의 차트(재배열할 수 있는 타일 형태의 그래프)를 유연하게 표시할 수 있습니다. 대시보드에 표현되는 각 타일 및 위젯은 워크시트에 결과를 반환하는 SQL 쿼리를 실행하여 생성됩니다.

Databases

The screenshot shows the Snowflake UI with the left sidebar expanded. Under the 'Data' section, the 'Databases' tab is selected. The main area displays a table titled 'Databases' with one entry: 'SNOWFLAKE'. The table includes columns for NAME, SOURCE, OWNER, and CREATED. A search bar and filter buttons are at the top right.

Data 아래의 **Database** 탭에는 사용자가 생성했거나 액세스 권한이 있는 데이터베이스에 대한 정보가 표시됩니다. 데이터베이스 생성, 복제, 삭제 또는 소유권(Ownership)을 이전하고 UI에서 데이터를 로드할 수 있습니다.

Shared Data

The screenshot shows the Snowflake UI with the left sidebar expanded. Under the 'Data' section, the 'Shared Data' tab is selected. The main area displays a table titled 'Shared With Me' with one entry: 'SFC_SAMPLES'. The table includes columns for SOURCE, OWNER, and LAST UPDATED. A search bar and filter buttons are at the top right. Below this, there is a section titled 'Snowflake Demo Resources' showing four demo resources: 'CITIBIKE Admins - Snowflake Demo Resources', 'SAP ERP Employee, Customer, Material', 'Snowhealth', and 'Citibike COVID-19 - Email Campaign Performance'. Each resource card includes a thumbnail, title, description, and publish date.

또한 Data 아래의 **Shared Data** 탭에서는 데이터 복사본을 만들지 않고도 별도의 Snowflake 계정 또는 외부 사용자 간에 Snowflake 테이블을 쉽고 안전하게 공유하도록 데이터 공유를 구성할 수 있습니다.

Marketplace

The screenshot shows the Snowflake Marketplace interface. On the left, there's a sidebar with navigation links: Worksheets, Dashboards, Data (Databases, Shared Data, Marketplace), Compute, Account, and Classic Console. The 'Marketplace' link is highlighted. At the top right, there's a search bar labeled 'Search Snowflake Marketplace', dropdown menus for 'Categories', 'Business Needs', 'Providers', and 'My Requests', and a 'New' button with the subtext 'Buy data products seamlessly on Snowflake. Sign up for Private Preview'. Below the search bar are several category tabs: Ready to Query, Free, Weather, Financial, 360-Degree Customer View, Advertising Optimization, and Health and Life Sciences. A section titled 'New Marketplace Capabilities' features four cards: 'Total Consumer Insights' by Infutor Data Solutions, 'IP to Geolocation' by IPInfo, 'SafeGraph Core Places - US...' by SafeGraph, and 'Weather Data for the United States - West' by Weather Source, LLC. The 'Featured Providers' section displays logos and names for Funnel, Dun & Bradstreet, Equifax, and ADP, Inc. The 'Most Recent' section shows four provider cards: CARTO (Analytics Toolbox), Monte Carlo (Data Observability Insights), Rystad Energy (RENEWableCube - A Global Plant-Level Solar and Wind Database), and Mintec Ltd (Commodity Pricing Data & Forecasts). Each card includes a brief description and a 'More' link.

Data 아래의 마지막 탭인 **Marketplace**는 모든 Snowflake 고객이 공급자가 제공한 데이터 세트를 검색하고 사용할 수 있는 곳입니다. 공유 데이터에는 공개 및 개인화의 두 가지 유형이 있습니다. 공개 데이터는 즉시 쿼리할 수 있는 무료 데이터 세트입니다. 개인화된 데이터는 데이터 공유 승인을 위해 데이터 제공자에게 연락해야 합니다.

History

The screenshot shows the Snowflake History page. On the left, there's a sidebar with navigation links: Worksheets, Dashboards, Data, Compute (with History selected), Warehouses, Resource Monitors, and Account. Below the sidebar is a 'Classic Console' link. The main area has a header 'History' with tabs for 'Queries' (selected) and 'Copies'. It displays '0 Queries' and includes filters for Status (All), User (ADMIN), Filters, Columns, and a search bar. A message at the bottom says 'No Queries' and 'There are no queries matching your filters.'

Compute 아래의 History 탭에 다음이 표시됩니다.

- **Queries**는 결과(사용자, 웨어하우스, 상태, 쿼리 태그 등)를 다듬는 데 사용할 수 있는 필터와 함께 이전 쿼리가 표시되는 곳입니다. Snowflake 계정에서 지난 14일 동안 실행된 모든 쿼리의 세부 정보를 봅니다. 자세한 내용을 보려면 쿼리 ID를 클릭하십시오.
- **Copies**은 Snowflake로 데이터를 수집하기 위해 실행되는 복사 명령의 상태를 보여줍니다.

Warehouses

The screenshot shows the 'Warehouses' page in the Snowflake interface. On the left, a sidebar menu includes 'Worksheets', 'Dashboards', 'Data', 'Compute' (with 'History' and 'Warehouses' sub-options), 'Resource Monitors' (which is selected and highlighted in blue), and 'Account'. The main content area is titled 'Warehouses' and shows a table with one row:

NAME	STATUS	SIZE	CLUSTERS	RUNNING	QUEUED	OWNER	CREATED
COMPUTE_WH	Suspended	X-Large	1 - 1	0	0	SYSADMIN	just now

At the top right of the main area, there is a blue button labeled '+ Warehouse'. Below the table, there are search and filter options: 'Search' (with a magnifying glass icon), 'Status All', 'Size All', and a clear search button (a small 'C' icon). A 'PREVIEW APP' button is visible at the bottom left.

또한 Compute 아래의 Warehouses 탭은 Snowflake에서 데이터를 로드하거나 쿼리하기 위해 가상 웨어하우스로 알려진 컴퓨팅 리소스를 설정하고 관리하는 곳입니다. COMPUTE_WH(XS)라는 웨어하우스가 이미 사용자 환경에 있습니다.

Resource Monitors

The screenshot shows the 'Resource Monitors' page in the Snowflake interface. The sidebar menu is identical to the previous screenshot, with 'Resource Monitors' selected. The main content area is titled 'Resource Monitors' and shows a table with zero rows:

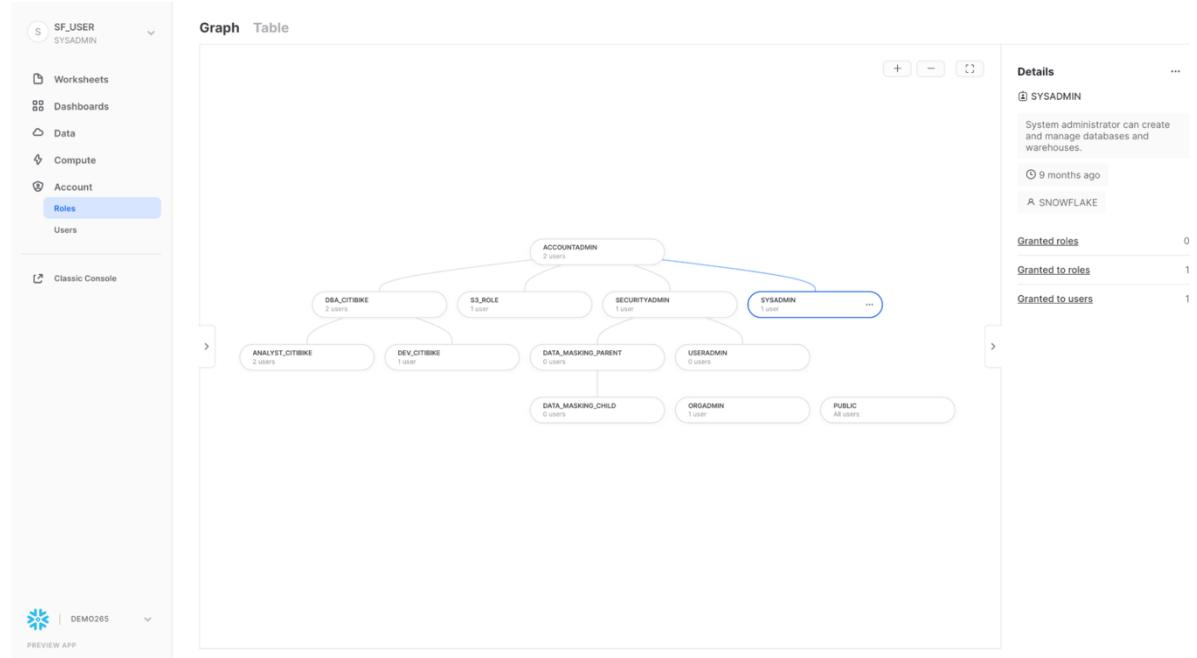
NAME	TYPE	DESCRIPTION

Below the table, a message states: 'No Resource Monitors'. It includes a small icon of a document with a checkmark. The text continues: 'There are no resource monitors associated with this role. Switch roles to view resource monitors available to that role.'

At the bottom left, there is a 'PREVIEW APP' button.

Compute 아래의 마지막 탭인 **Resource Monitors**에는 가상 웨어하우스(Virtual Warehouse)가 소비하는 크레딧(Credit)를 제어하기 위해 생성된 모든 리소스 모니터가 표시됩니다. 각 리소스 모니터에 대해 크레딧 할당량, 모니터링 유형, 일정 및 가상 웨어하우스가 크레딧 한도에 도달했을 때 수행되는 조치를 보여줍니다.

Roles



Account 아래의 **Roles** 탭에는 역할 목록과 계층(Hierarchy)이 표시됩니다. 이 탭에서 역할을 생성, 재구성 및 사용자에게 부여할 수 있습니다. 역할은 페이지 상단의 **Table**을 클릭하여 표/목록 형식으로 표시할 수도 있습니다.

Users

The screenshot shows the Salesforce classic interface. On the left, there's a sidebar with a user icon and the text 'SF_USER' and 'SYSADMIN'. Below the sidebar are links for 'Worksheets', 'Dashboards', 'Data', 'Compute', 'Account', 'Roles', and 'Users', with 'Users' being the active tab. At the bottom of the sidebar is a link to 'Classic Console'. The main content area has a title 'Users' and a message '0 Users'. Below this is a small icon of a person with a lightbulb above their head. A message 'Data not found' follows, stating 'Either this data doesn't exist or your role may not have access to it.' At the top right of the main area are buttons for 'Search', 'Owner All', 'Status All', and a refresh icon.

또한 **Account** 탭 아래의 **Users** 탭에는 계정의 사용자 목록, 기본 역할 및 사용자 소유자가 표시됩니다. 새 계정의 경우 추가 역할이 생성되지 않았기 때문에 레코드가 표시되지 않습니다. 탭에서 사용할 수 있는 모든 정보를 보려면 역할을 ACCOUNTADMIN으로 전환(Switch Role)하세요.

The screenshot shows the Salesforce classic interface with a user profile dropdown menu. The top bar shows 'SF_USER' and 'SYSADMIN'. The dropdown menu contains the following items: 'Switch Role' (with 'SYSADMIN' selected), 'Profile', 'Partner Connect', 'Documentation', and 'Sign Out'. At the bottom of the menu is a link to 'Classic Console'.

UI 오른쪽 상단의 사용자 이름을 클릭하면 비밀번호, 역할 및 기본 설정을 변경할 수 있습니다. Snowflake에는 여러 시스템 정의 역할(System Defined Roles)이 있습니다. 현재 기본 역할인 **SYSADMIN**을 사용하고 있으며 대부분의 실습에서 이 역할을 유지하겠습니다.

SYSADMIN **SYSADMIN**(시스템 관리자라고도 함) 역할은 웨어하우스, 데이터베이스 및 기타 객체를 계정에 생성할 수 있는 권한을 가집니다. 실제 환경에서는 서로 다른 역할에 따라서 적합한 Role을 할당하고 사용자에게 이 Role을 할당하는 형태로 액세스 제어가 이루어 집니다.

- 액세스 제어의 개요: <https://docs.snowflake.com/ko/user-guide/security-access-control-overview.html>
- 액세스 제어 구성하기: <https://docs.snowflake.com/ko/user-guide/security-access-control-configure.html>
- 사용자 지정 역할 만들기: <https://docs.snowflake.com/ko/user-guide/security-access-control-configure.html#creating-custom-roles>

3. Staging and Loading Data

샘플 트랜잭션 정형 데이터를 Snowflake에 로드할 준비부터 시작하겠습니다.

이 섹션은 다음과 같은 단계로 진행됩니다.

- 데이터베이스 및 테이블 생성
- 외부 스테이지(External Stage) 생성
- 데이터에 대한 파일 형식(File Format) 생성

Snowflake 데이터 로딩하는 방법은 아래 매뉴얼에서 확인하기 바랍니다.

Snowflake에 데이터 로딩하기: <https://docs.snowflake.com/ko/user-guide-data-load.html>

Snowflake로 데이터 가져오기

사용할 데이터는 자전거 공유 서비스의 트랜잭션 데이터입니다. 특정 정거장에서 자전거를 빌려서 특정 장소에 반납하면 하나의 트랜잭션(한 번의 Trip)이 완결되고 하나의 레코드로 저장됩니다. 이 데이터는 미국 동부 지역의 Amazon AWS S3 버킷에 미리 저장되어 있으며, 이동 시간, 위치, 사용자 유형, 성별, 나이 등에 관한 관련 정보로 구성됩니다. AWS S3에서 이 데이터는 6,150만 행, 376개의 객체로 표현되고 1.8GB로 압축되어 있습니다.

아래는 Citi Bike CSV 데이터 파일 중 하나의 한 조각입니다.

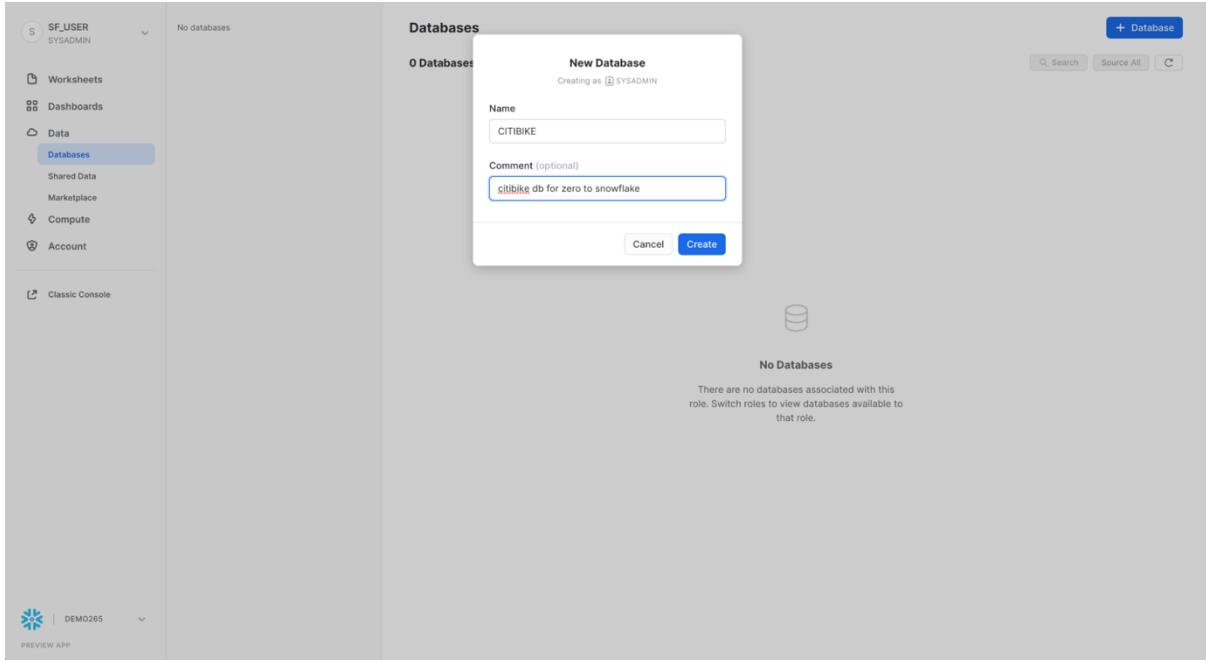
```
"tripduration","starttime","stoptime","start station id","start station name","start station latitude","start station longitude","end station id","end station name","end station latitude","end station longitude","bikeid","name_localizedValue0","usertype","birth year","gender"196,"2018-01-01 00:01:51","2018-01-01 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",1997,1207,"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",1978,1613,"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
```

맨 위에 헤더가 한 줄이 있고, 각 컬럼은 큰 따옴표로 둘러싸여 있으며 쉼표로 구분된 형식입니다. 이 형식은 이 섹션의 뒷부분에서 이 데이터를 저장할 Snowflake 테이블을 구성할 때 적용됩니다.

데이터베이스 및 테이블 생성

먼저, 정형 데이터를 로딩하는 데 사용할 CITIBIKE라는 이름의 데이터베이스를 생성합니다.

Databases 탭으로 이동합니다. 만들기를 클릭하고, 데이터베이스 이름을 CITIBIKE로 지정한 뒤, 마침을 클릭합니다.



이제 Worksheets 탭으로 이동합니다.

A screenshot of the Snowflake Worksheet interface. At the top, it shows the date and time: '2022-01-20 9:34am'. On the right, it shows the user 'SYSADMIN - TASK_WH' and a 'Share' button. Below the header, there's a search bar and a 'Pinned' section which is currently empty. The main workspace has a query editor with the following content:

```
No Database selected
1 | select col from table
```

The 'Query' tab is selected at the bottom. To the right of the query editor, there's a 'Query Details' panel showing performance metrics: 'Query duration 790ms', 'Rows 1', and 'COLUMN1 0% filled 100% blank'. The bottom right corner of the interface has a small icon of a document with a chart.

워크시트 내에서 컨텍스트를 적절하게 설정해야 합니다. 워크시트의 오른쪽 상단에 + 옆에 있는 상자를 클릭하여 상황에 맞는 메뉴를 표시합니다. 여기에서 각 워크 시트에서 보고 실행할 수 있는 요소를 제어합니다. 여기서 UI를 사용하여 컨텍스트를 설정합니다.

다음과 같이 Role과 Warehouse의 컨텍스트 설정을 선택합니다.

Role: SYSADMIN Warehouse: COMPUTE_WH

The screenshot shows the Snowflake UI interface. On the left, there's a sidebar with pinned objects and a search bar. The main area has a query editor with the following text:

```
1 select col from table where created = :daterange
```

To the right of the query editor, there are two dropdown menus for setting context:

- Roles**: Shows options like SYSADMIN, ANALYST_CITIBIKE, DBA_CITIBIKE, DEV_CITIBIKE, and PUBLIC. SYSADMIN is selected.
- Warehouses**: Shows options like COMPUTE_WH. COMPUTE_WH is selected.

Below these context menus, there's a preview pane showing the results of the query. The results table has one row with the value 'null' under the column 'COLUMN1'. To the right of the preview pane, there's a 'Query Details' panel showing metrics: Query duration (790ms), Rows (1), and COLUMN1 (100% blank).

다음으로 데이터베이스 드롭다운에서 Database와 Schema의 컨텍스트 설정을 선택합니다.

Database: CITIBIKE Schema = PUBLIC

The screenshot shows the Snowflake UI interface. On the left, there's a sidebar with pinned objects and a search bar. The main area has a query editor with the following text:

```
1 null
```

To the right of the query editor, there are two dropdown menus for setting context:

- Databases**: Shows options like CITIBIKE and SNOWFLAKE_SAMPLE_DATA. CITIBIKE is selected.
- Schemas**: Shows options like PUBLIC. PUBLIC is selected.

Below these context menus, there's a preview pane showing the results of the query. The results table has one row with the value 'null' under the column 'COLUMN1'. To the right of the preview pane, there's a 'Query Details' panel showing metrics: Query duration (56ms), Rows (1), and COLUMN1 (100% null).

SQL로 데이터베이스 생성 및 컨텍스트 설정

앞에서 UI로 진행했던 과정은 SQL을 통해서도 진행할 수 있습니다.

```
/* 먼저 어떤 Role과 Warehouse를 사용할지 컨텍스트를 지정합니다.*/
use role sysadmin;
use warehouse compute_wh;

/* citibike 데이터베이스를 생성합니다.*/
create or replace database citibike;

/* 테이블을 생성할 데이터베이스와 스키마를 컨텍스트로 지정합니다.*/
use database citibike;
use schema public;
```

다음으로 샘플 트랜잭션 정형 데이터를 로드하는 데 사용할 TRIPS라는 테이블을 만듭니다. UI를 사용하는 대신 워크시트를 사용하여 테이블을 생성하는 DDL을 실행합니다. 다음 SQL 텍스트를 워크시트에 복사합니다.

```
/* 데이터를 로딩할 테이블을 해당 칼럼으로 생성합니다.*/
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);

/* 생성된 테이블과 설정된 parameter를 확인합니다.*/
show tables like 'tri%' in citibike.public;

/* alter table 명령으로 parameter를 설정합니다. 예를 들어, Change_tracking을
true로 설정하면 테이블에 일어난 모든 change를 활용해서 CDC 작업을 수행할 수
있습니다.*/
alter table trips set change_tracking = true;

/* 테이블의 각 컬럼 정보를 확인합니다.*/
desc table trips;

```

- 테이블 관리: <https://docs.snowflake.com/ko/sql-reference/ddl-table.html#table-management>

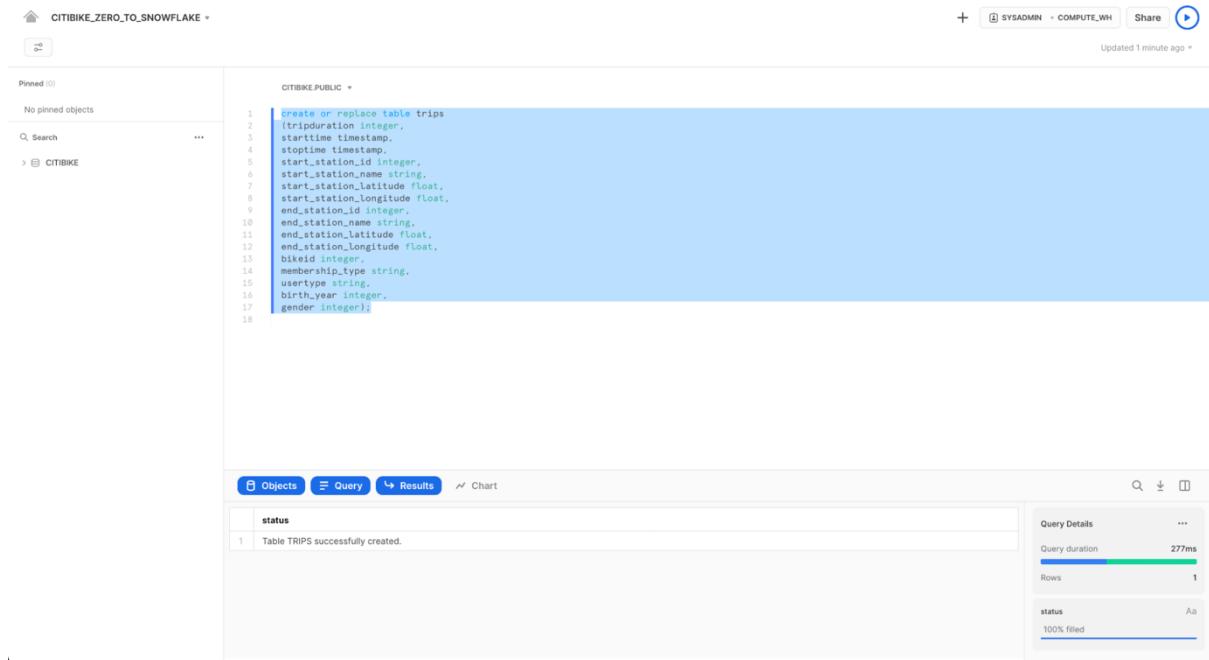
명령을 실행하는 다양한 옵션.

SQL 명령은 UI를 통해서나, Worksheets 탭을 통해, SnowSQL 명령행 도구를 사용해서, ODBC/JDBC를 통해 선택한 SQL 편집기를 이용해서 또는 Python이나 Spark 커넥터를 통해 실행할 수 있습니다.

- SnowSQL(CLI 클라이언트) : <https://docs.snowflake.com/ko/user-guide/snowsql.html>
- ODBC 드라이버: <https://docs.snowflake.com/ko/user-guide/odbc.html>

커서를 명령 내 어디든 두고 페이지 상단의 파란색 실행 버튼을 클릭하여 쿼리를 실행하십시오. 또는 바로 가기 키 [Ctrl]/[Cmd]+[Enter]를 이용하십시오.

TRIPS 테이블이 생성되었는지 확인합니다. 워크시트 하단에 "테이블 TRIPS가 성공적으로 생성됨" 메시지를 표시하는 결과 섹션이 표시되어야 합니다.

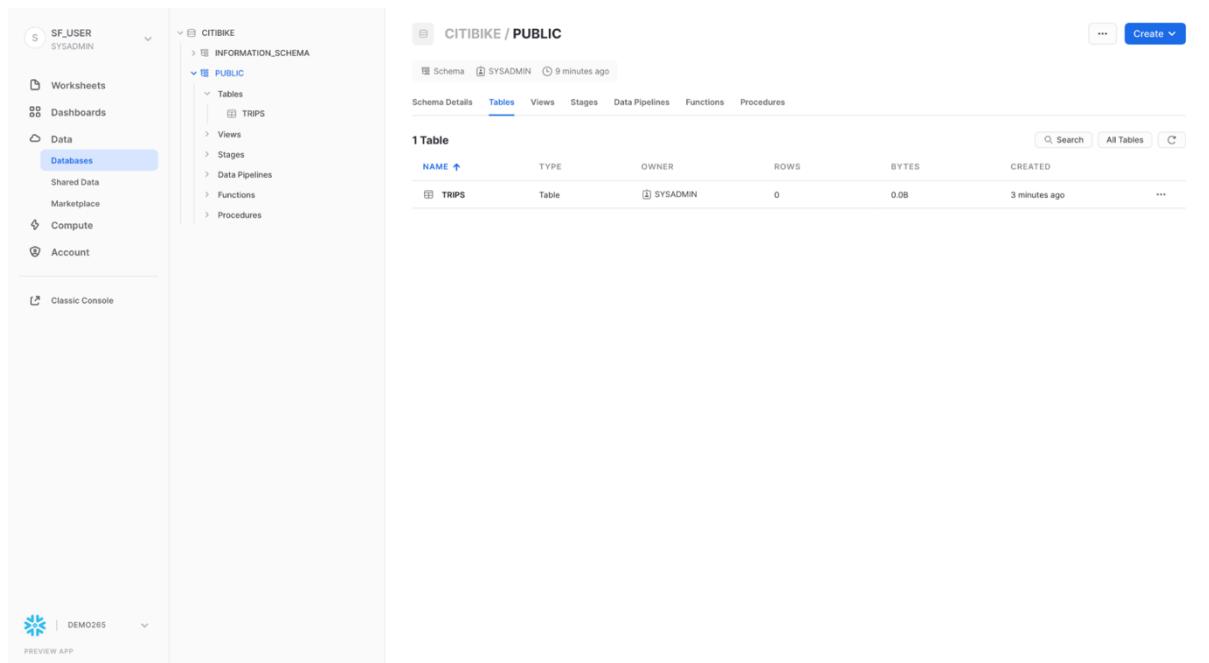


The screenshot shows a Snowflake worksheet titled "CITIBIKE_ZERO_TO_SNOWFLAKE". On the left, there's a sidebar with pinned objects, a search bar, and a connection dropdown. The main area contains a code editor with the following SQL script:

```
1 create or replace table trips
2   (tripduration integer,
3    starttime timestamp,
4    stoptime timestamp,
5    start_station_id integer,
6    start_station_name string,
7    start_station_latitude float,
8    start_station_longitude float,
9    end_station_id integer,
10   end_station_name string,
11   end_station_latitude float,
12   end_station_longitude float,
13   bikeid integer,
14   membership_type string,
15   user type string,
16   birth_year integer,
17   gender integer);
```

Below the code editor, there's a results section with tabs for Objects, Query, Results, and Chart. The Results tab shows a status message: "Table TRIPS successfully created." To the right, there's a "Query Details" panel with metrics like Query duration (277ms), Rows (1), and status (100% filled).

워크시트의 왼쪽 상단에 있는 **HOME** 아이콘을 클릭하여 **Databases** 탭으로 이동합니다. 그런 다음 **Data > Databases**를 클릭합니다. 데이터베이스 목록에서 **CITIBIKE > PUBLIC > TABLES**를 클릭하여 새로 생성된 **TRIPS** 테이블을 확인합니다.



The screenshot shows the Snowflake UI with the navigation bar on the left. Under the "Data" section, the "Databases" tab is selected. In the main content area, the path "CITIBIKE / PUBLIC / TABLES" is selected. The table list shows one entry: "TRIPS". The table details page is shown on the right, displaying the following information:

NAME	TYPE	OWNER	ROWS	BYTES	CREATED
TRIPS	Table	SYSADMIN	0	0.0B	3 minutes ago

방금 생성한 테이블 구조를 보려면 TRIPS 및 Columns 탭을 클릭합니다.

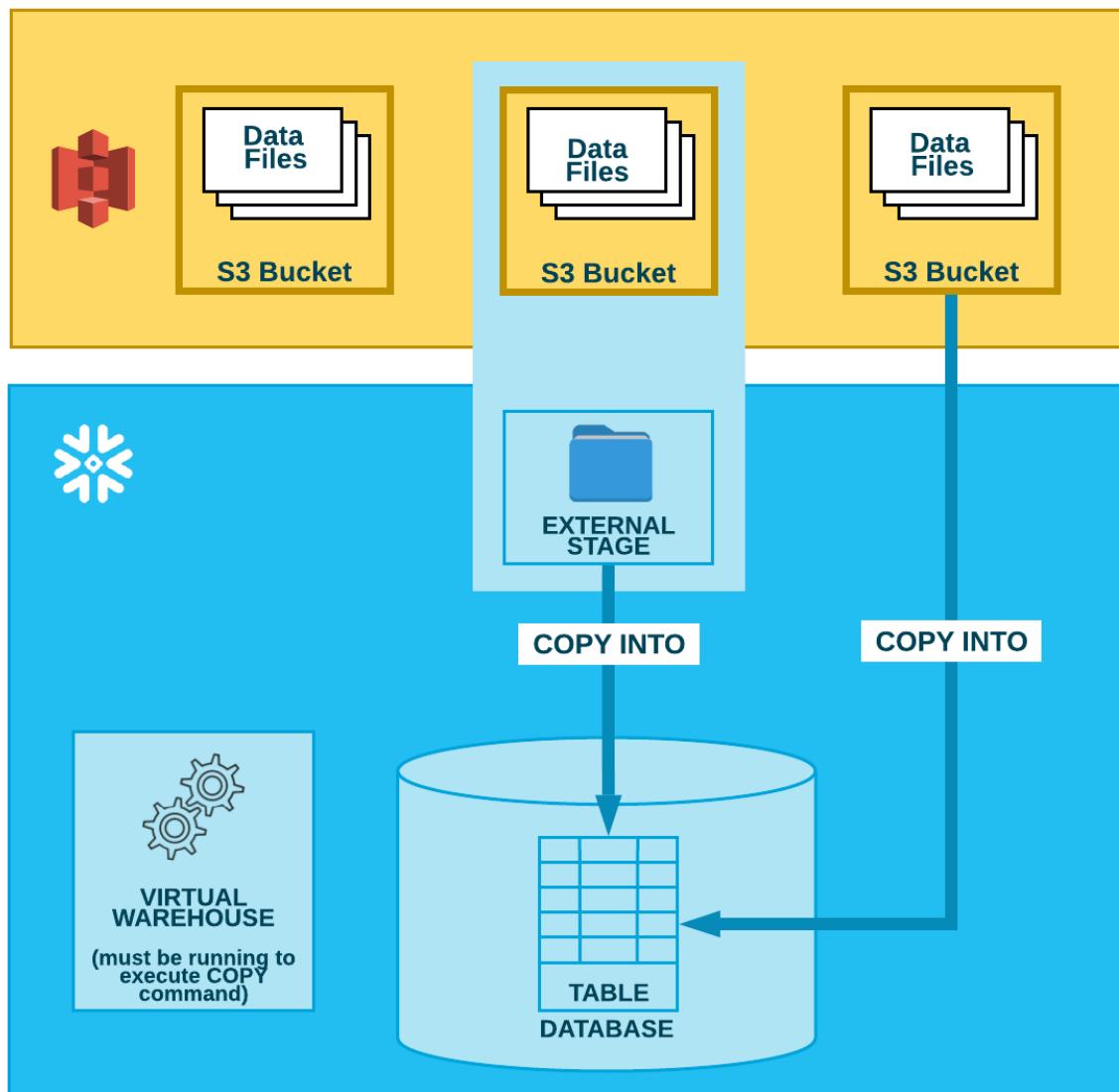
The screenshot shows the Snowflake Data Studio interface. On the left, the sidebar navigation includes: SF_USER (SYSADMIN), Worksheets, Dashboards, Data (selected), Databases (selected), Shared Data, Marketplace, Compute, Account, and Classic Console. At the bottom left is a preview app icon labeled DEMO265 and PREVIEW APP. The main content area displays the 'CITIBIKE / PUBLIC / TRIPS' table. It shows 'Table Details' (Table, SYSADMIN, 3 minutes ago, 0 rows, 0.08 MB) and 'Columns' (16 columns). The 'Columns' tab is selected, showing the following schema:

NAME	TYPE	NULLABLE	DEFAULT
BIKEID	NUMBER(38,0)	Yes	NULL
BIRTH_YEAR	NUMBER(38,0)	Yes	NULL
END_STATION_ID	NUMBER(38,0)	Yes	NULL
END_STATION_LATITUDE	FLOAT	Yes	NULL
END_STATION_LONGITUDE	FLOAT	Yes	NULL
END_STATION_NAME	VARCHAR(16777216)	Yes	NULL
GENDER	NUMBER(38,0)	Yes	NULL
MEMBERSHIP_TYPE	VARCHAR(16777216)	Yes	NULL
STARTTIME	TIMESTAMP_NTZ(9)	Yes	NULL
START_STATION_ID	NUMBER(38,0)	Yes	NULL
START_STATION_LATITUDE	FLOAT	Yes	NULL
START_STATION_LONGITUDE	FLOAT	Yes	NULL
START_STATION_NAME	VARCHAR(16777216)	Yes	NULL
STOPTIME	TIMESTAMP_NTZ(9)	Yes	NULL
TRIPDURATION	NUMBER(38,0)	Yes	NULL
USERTYPE	VARCHAR(16777216)	Yes	NULL

Create an External Stage

데이터베이스 테이블로 읽을 데이터는 외부 S3 버킷에 준비되어 있으므로, 이 데이터를 사용하기 전에 먼저 외부 버킷의 위치를 지정하는 단계를 생성해야 합니다.

- 아마존 S3에서 대량 로드: <https://docs.snowflake.com/ko/user-guide/data-load-s3.html>



데이터 송신/전송 비용을 방지하려면 Snowflake 계정과 동일한 클라우드 제공업체 및 지역에서 스테이징 위치를 선택해야 합니다.

Databases 탭에서 CITIBIKE 데이터베이스와 PUBLIC 스키마를 클릭합니다. **Stage** 탭에서 **Create** 버튼을 클릭한 다음 **Stages > Amazon S3**를 클릭합니다.

The screenshot shows the Snowflake UI interface. On the left, the navigation sidebar is visible with options like Worksheets, Dashboards, Data (selected), Databases (selected), Shared Data, Marketplace, Compute, and Account. Under Data, the Databases section is expanded, showing CITIBIKE, INFORMATION_SCHEMA, and PUBLIC. The PUBLIC schema is selected, revealing Tables, Views, Stages, Data Pipelines, Functions, and Procedures. The Stages section is currently active. In the center, the 'CITIBIKE / PUBLIC' schema details page is shown, with the Stages tab selected. A modal window titled 'Create' is open, showing a dropdown menu with options: Table, View, Stage (selected), Pipe, Stream, Task, Function, and Procedure. Under Stage, there are sub-options: External Stage, Amazon S3, Microsoft Azure, Google Cloud Platform, Storage Integration, Amazon S3, Microsoft Azure, and Google Cloud Platform. The 'Amazon S3' option is highlighted. Below the modal, a message says 'No stages' and 'This schema contains no stages.' At the bottom left, there's a preview app icon with 'DEMO265' and 'PREVIEW APP' text.

열리는 "Create Securable Object" 대화 상자에서 SQL 문에서 다음 값을 바꿉니다.

stage_name: citibike_trips

url: s3://snowflake-workshop-lab/citibike-trips-csv/

참고: URL 끝에 마지막 슬래시(/)를 포함해야 합니다. 그렇지 않으면 나중에 버킷에서 데이터를 로드할 때 오류가 발생합니다.

이 실습의 S3 버킷은 공개되어 있으므로 자격 증명 옵션을 비워 둘 수 있습니다. 실제 시나리오에서 외부 단계에 사용되는 버킷에는 주요 정보가 필요할 수 있습니다.



SQL로 External Stage 생성하기

앞에서 UI로 진행했던 과정은 SQL을 통해서도 진행할 수 있습니다.

```
/* 외부 스테이지를 생성합니다.
```

```
AWS S3의 버킷을 그대로 데이터 파일을 저장하는 외부 스테이지로 활용할 수 있습니다. */
```

```
create or replace stage citibike_trips
url='s3://snowflake-workshop-lab/citibike-trips-csv/';
```

- Create Stage: <https://docs.snowflake.com/ko/sql-reference/sql/create-stage.html>
- Storage Integration: <https://docs.snowflake.com/ko/sql-reference/sql/create-storage-integration.html>

실환경에서는 보안 Credentials의 자격 증명 옵션들이 필요하며 이 옵션들은 Create Stage에서 직접 지정을 하거나 Storage Integration에서 설정을 해 두고 활용할 수도 있습니다.

이제 `citibike_trips` 스테이지를 살펴보도록 하겠습니다.

```
list @citibike_trips;
```



하단 창의 결과에서 스테이지의 파일 목록을 확인해야 합니다.

name	size	md5	last_modified
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/03/data_01a19496-4	219,733	5ce4f42198b692a8cea379ccdd094aca	Wed, 12 Jan 2022 13:10:38 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/04/data_01a19496-4	240,490	9207a4132df8da7cd84b4fd1f465d8	Wed, 12 Jan 2022 13:10:37 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/05/data_01a19496-4	227,228	d982d1a2e3692a0e5191f734e9325	Wed, 12 Jan 2022 13:10:34 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/06/data_01a19496-4	235,713	1cc34af2eeb9c3eab0771f3ae193167	Wed, 12 Jan 2022 13:10:34 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/07/data_01a19496-4	224,639	82fe12194abeb60dc2e46579cef3dd3	Wed, 12 Jan 2022 13:10:39 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/08/data_01a19496-4	236,398	2d6cc2d8b20245127bcfb65beb1938	Wed, 12 Jan 2022 13:10:35 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/09/data_01a19496-4	227,747	4a0b6bcc6d93b0379e23da4a466fc1a8c	Wed, 12 Jan 2022 13:10:40 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/10/data_01a19496-4	249,432	fe216a83f84d7df525199886194cf6	Wed, 12 Jan 2022 13:10:33 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/11/data_01a19496-4	285,569	c057bbb681dd842d0ea42d8aa24777d8	Wed, 12 Jan 2022 13:10:40 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/12/data_01a19496-4	261,702	91cb698909124c23ff70e75aa2d71c	Wed, 12 Jan 2022 13:10:32 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/13/data_01a19496-4	269,365	1bd678c7148d8995ac76e85276e1d2a2	Wed, 12 Jan 2022 13:10:40 GMT
s3://snowflake-workshop-lab/citibike-trips-parquet/2013/06/14/data_01a19496-4	266,097	d8572a36ea0df3e7f5be1b725863b0	Wed, 12 Jan 2022 13:10:38 GMT

citibike_trips 스테이지에 저장된 데이터 파일의 현황을 검토합니다.

/* 마지막으로 실행시킨 쿼리의 결과를 통해서 전체 사이즈, 평균 파일 크기, 파일 갯수를 확인합니다. */

```
select
    floor(sum($2) / power(1024, 3), 1) total_compressed_storage_gb,
    floor(avg($2) / power(1024, 2), 1) avg_file_size_mb,
    count(*) as num_files
from
    table(result_scan(last_query_id()));
```

Result Scan은 결과가 테이블인 것처럼 이전 명령의 결과 세트를 반환합니다.

- Result Scan: https://docs.snowflake.com/ko/sql-reference/functions/result_scan.html

Create File Format

Snowflake로 데이터를 로드하려면, 먼저 데이터 구조와 일치하는 파일 형식(File Format)을 생성합니다.

워크시트에서 다음 명령을 실행하여 파일 형식을 만듭니다.create file format

```
/* 데이터 파일에 저장한 데이터의 구조를 반영하는 File Format을 생성합니다. */

create or replace file format csv type='csv'
    compression = 'auto' field_delimiter = ',' record_delimiter = '\n'
    skip_header = 0 field Optionally_enclosed_by = '\"' trim_space = false
    error_on_column_count_mismatch = false escape = 'none'
    escape_unenclosed_field = '\"'
    date_format = 'auto' timestamp_format = 'auto' null_if = ('') comment = 'file
format for ingesting data to snowflake';
```

- Create File Format: <https://docs.snowflake.com/ko/sql-reference/sql/create-file-format.html>

다음 명령을 실행하여 파일 형식이 올바른 설정으로 생성되었는지 확인합니다.

```
/* 파일 포맷이 생성되었는지 확인합니다. */

show file formats in database citibike;

/* 메타데이터를 통해서 각 파일의 경로 및 이름과 레코드 카운트에 대한 정보를 확인합니다. */

select metadata$filename, metadata$file_row_number
from @citibike_trips (file_format => csv);
```

생성된 파일 형식은 결과에 나열되어야 합니다.

The screenshot shows the Snowflake interface. In the top navigation bar, it says 'CITIBIKE_ZERO_TO_SNOWFLAKE'. On the left sidebar, there's a 'Pinned' section with 'No pinned objects' and a search bar. Below that is a tree view with 'CITIBIKE' expanded. The main area contains a code editor with the following SQL script:

```
8   start_station_name string,
9   start_station_latitude float,
10  start_station_longitude float,
11  end_station_id integer,
12  end_station_name string,
13  end_station_latitude float,
14  end_station_longitude float,
15  bikeid integer,
16  member_casual string,
17  usertype string,
18  birth_year integer,
19  gender integer
20 );
21 --created external stage. List files.
22 list @citibike_trips;
23
24 --create file format
25 CREATE_FILE_FORMAT "CITIBIKE"."PUBLIC".CSV TYPE = 'CSV' COMPRESSION = 'AUTO' FIELD_DELIMITER = ',' RECORD_DELIMITER = '\n' SKIP_HEADER = 0
FIELD_OPTIONALLY_ENCLOSED_BY = '\042' TRIM_SPACE = FALSE ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE ESCAPE = 'NONE' ESCAPE_UNENCLOSED_FIELD = '\134' DATE_FORMAT = 'AUTO'
TIMESTAMP_FORMAT = 'AUTO' NULL_IF = ('') COMMENT = 'creation of file format for zero to snowflake';
26
27 --verify file format is created
28 | show file formats in database citibike;
```

Below the code editor is a results table:

	created_on	name	database_name	schema_name	type	owner	comment	format
1	2022-01-20 11:12:27.666 -0800	CSV	CITIBIKE	PUBLIC	CSV	SYSADMIN	creation of file format for zero to snowflake	{T}

On the right side, there's a 'Query Details' panel showing:

- Query duration: 97ms
- Rows: 1
- created_on: 100% filled
- name: 100% filled
- database_name: 100% filled

Copy Into <Table>

이 섹션에서는 데이터 웨어하우스와 COPY 명령을 사용하여 방금 생성한 Snowflake 테이블에 정형 데이터 대량 로드 (bulk loading)를 진행합니다.

- Copy Into <Table>: <https://docs.snowflake.com/ko/sql-reference/sql/copy-into-table.html>

데이터 로드를 위한 웨어하우스 크기 조정

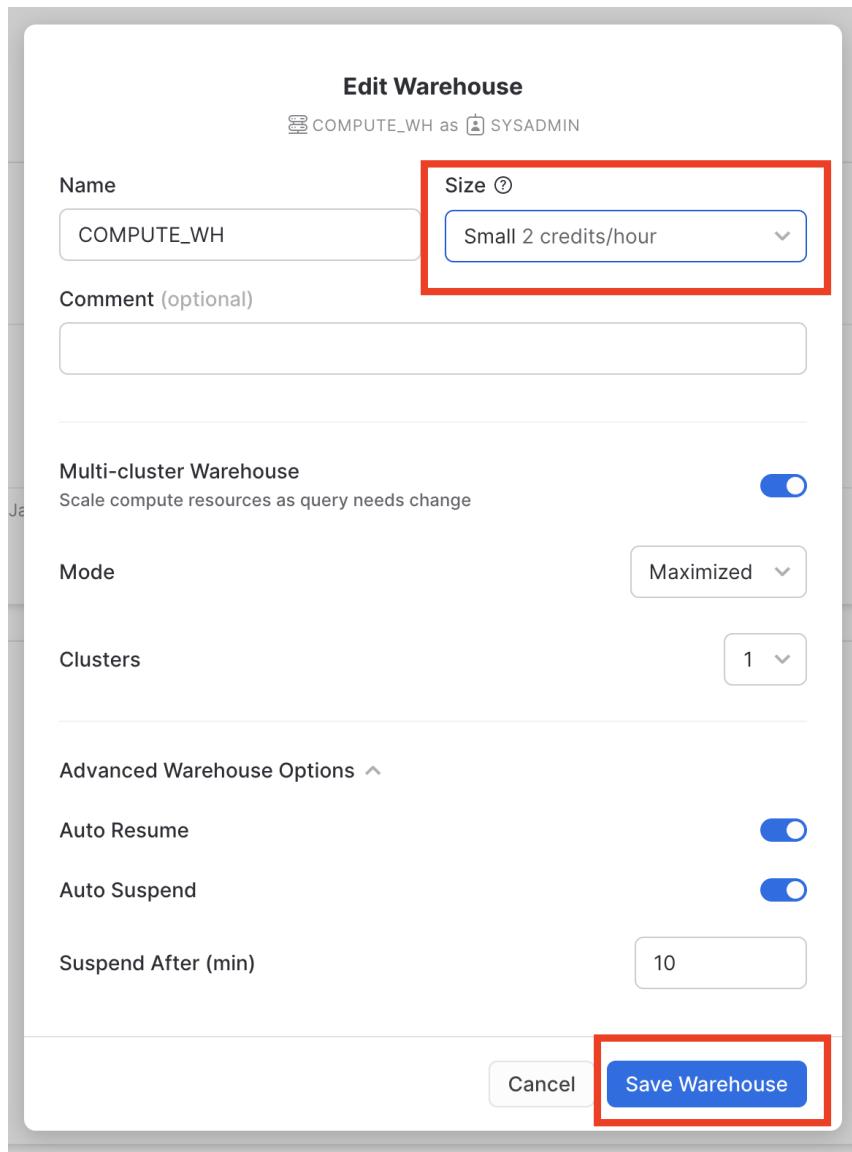
데이터 로딩은 많은 컴퓨팅 리소스를 사용할 수도 있어서 데이터 로딩을 하기 전에 가상 웨어하우스의 성능을 높이도록 하겠습니다.

Snowflake의 컴퓨팅 노드는 가상 웨어하우스 (Virtual Warehouse)라고 하며 워크로드가 데이터 로드, 쿼리 실행 또는 DML 작업을 수행하는지 여부에 따라 워크로드에 맞춰 동적으로 크기를 늘리거나 줄일 수 있습니다. 각 워크로드는 자체 데이터 웨어하우스를 보유할 수 있으므로 리소스 경합이 없습니다.

Home->Admin->Warehouses 탭으로 이동합니다. 여기에서 기존 웨어하우스를 모두 보고 사용 추세를 분석할 수 있습니다.

상단 오른쪽 상단 모서리에 있는 **+ Warehouse** 옵션을 확인하세요. 여기에서 새 웨어하우스를 빠르게 추가할 수 있습니다. 단, 30일 체험판 환경에 포함된 기존 웨어하우스 COMPUTE_WH를 사용합니다.

COMPUTE_WH 웨어하우스 행을 클릭합니다. 그런 다음 위의 오른쪽 상단 모서리에 있는 ...(점 점)을 클릭하고 **Edit**을 선택해서 Size를 조정할 수 있습니다.



- **Size** 드롭다운은 웨어하우스의 용량을 선택하는 곳입니다. 더 큰 데이터 로드 작업이나 컴퓨팅 집약적인 쿼리의 경우 더 큰 웨어하우스가 권장됩니다. 이 데이터 웨어하우스의 **Size**를 X-Small에서 Small로 변경합니다. **Save Warehouse** 버튼을 클릭합니다.

데이터 로드

이제 COPY 명령을 실행하여 데이터를 앞서 생성한 TRIPS 테이블로 로드할 수 있습니다.

Worksheet 탭에서 컨텍스트가 올바르게 설정되었는지 확인합니다.

Role: SYSADMIN Warehouse: COMPUTE_WH Database: CITIBIKE Schema = PUBLIC

The screenshot shows the Snowflake Worksheet interface. At the top, it displays the current role as 'SYSADMIN', warehouse as 'COMPUTE_WH', database as 'CITIBIKE', and schema as 'PUBLIC'. Below this, there are two dropdown menus: 'Roles' on the left containing options like 'ANALYST_CITIBIKE', 'DBA_CITIBIKE', 'DEV_CITIBIKE', and 'PUBLIC', with 'SYSADMIN' selected; and 'Warehouses' on the right containing 'COMPUTE_WH', with it also selected. A timestamp '15 seconds ago' is shown at the bottom right of the sidebar area.

워크시트에서 다음의 문을 실행하여 구성한 데이터를 테이블로 로드하고 실행 시간을 체크합니다.

```
/* citibike_trips 외부 스테이지에 있는 데이터 파일을 csv 파일 포맷에 맞춰서 trips 테이블에 로딩합니다. */
```

```
copy into trips from @citibike_trips file_format=csv pattern='.*CSV.*';
```

```
/* 테이블에 로딩된 데이터를 확인합니다. */
```

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

결과 창에서 로드된 각 파일의 상태를 확인해야 합니다. 로드가 완료되면 오른쪽 하단의 **Query Details** 창에서 마지막으로 실행된 명령문에 대한 다양한 상태, 오류 통계 및 시각화를 스크롤할 수 있습니다.

```

--create external stage via SQL
23 create or replace stage citibike_trips url = 's3://snowflake-workshop-lab/citibike-trips/';
24
25 --created external stage. List files.
26 list @citibike_trips;
27
28 --create or replace file format
29 create or replace file format csv type='csv'
30 compression = 'auto' field_delimiter = ',' record_delimiter = '\n'
31 skip_header = 0 field Optionally_enclosed_by = '\042' trim_space = false
32 error_on_column_count_mismatch = false escape = 'none' escape_ unenclosed_field = '\i34'
33 date_format = 'auto' timestamp_format = 'auto' null_if = ('') comment = 'file format for ingesting data for zero to snowflake';
34
35 --verify file format is created
36 show file format in database citibike;
37
38 --copy data from stage into target table
39 copy into trips from @citibike_trips file_format=csv;
40
41
42
43
44
45
46
47
48
49

```

file	status	rows_parsed	rows_loaded	error_limit	errors_seen	first_error
s3://snowflake-workshop-lab/citibike-trips/trips_2013_1_0.csv.gz	LOADED	112,123	112,123	1	0	
s3://snowflake-workshop-lab/citibike-trips/trips_2013_2_7_0.csv.gz	LOADED	93,143	93,143	1	0	
s3://snowflake-workshop-lab/citibike-trips/trips_2013_5_7_0.csv.gz	LOADED	111,042	111,042	1	0	
s3://snowflake-workshop-lab/citibike-trips/trips_2014_0_7_0.csv.gz	LOADED	112,334	112,334	1	0	
s3://snowflake-workshop-lab/citibike-trips/trips_2014_3_0_0.csv.gz	LOADED	126,769	126,769	1	0	
s3://snowflake-workshop-lab/citibike-trips/trips_2014_4_7_0.csv.gz	LOADED	109,635	109,635	1	0	
s3://snowflake-workshop-lab/citibike-trips/trips_2014_6_7_0.csv.gz	LOADED	106,608	106,608	1	0	
s3://snowflake-workshop-lab/citibike-trips/trips_2015_0_2_0.csv.gz	LOADED	125,062	125,062	1	0	
s3://snowflake-workshop-lab/citibike-trips/trips_2015_1_2_0.csv.gz	LOADED	154,940	154,940	1	0	

그런 다음 **Home** 아이콘을 클릭한 다음 **Activity > Query History**를 클릭 합니다. 목록 맨 위에서 마지막으로 실행된 COPY INTO 문을 선택합니다. 쿼리가 실행하기 위해 취한 단계, 쿼리 세부 정보, 가장 비싼 노드 및 추가 통계를 살펴 볼 수 있습니다.



이제 더 큰 웨어하우스로 TRIPS 테이블을 다시 로드하여 추가 컴퓨팅 리소스가 로드 시간에 미치는 영향을 살펴보겠습니다.

워크시트로 돌아가서 TRUNCATE TABLE 명령을 사용하여 모든 데이터와 메타데이터를 지웁니다.

```
/* trips 테이블의 데이터와 메타데이터를 지웁니다. */
truncate table trips;
```

결과에 "Query produced no results"(쿼리에서 결과가 생성되지 않음)이 표시되어야 합니다.

다음 ALTER WAREHOUSE를 사용하여 웨어하우스 크기를 large으로 변경합니다.

```
/* 명령어를 통해서 직접 가상 웨어하우스의 크기를 large로 변경합니다. */
alter warehouse compute_wh set warehouse_size='large';
```

다음 SHOW WAREHOUSES를 사용하여 변경 사항을 확인하십시오.

```
/* 웨어하우스의 변경 사항을 확인 */
show warehouses;
```

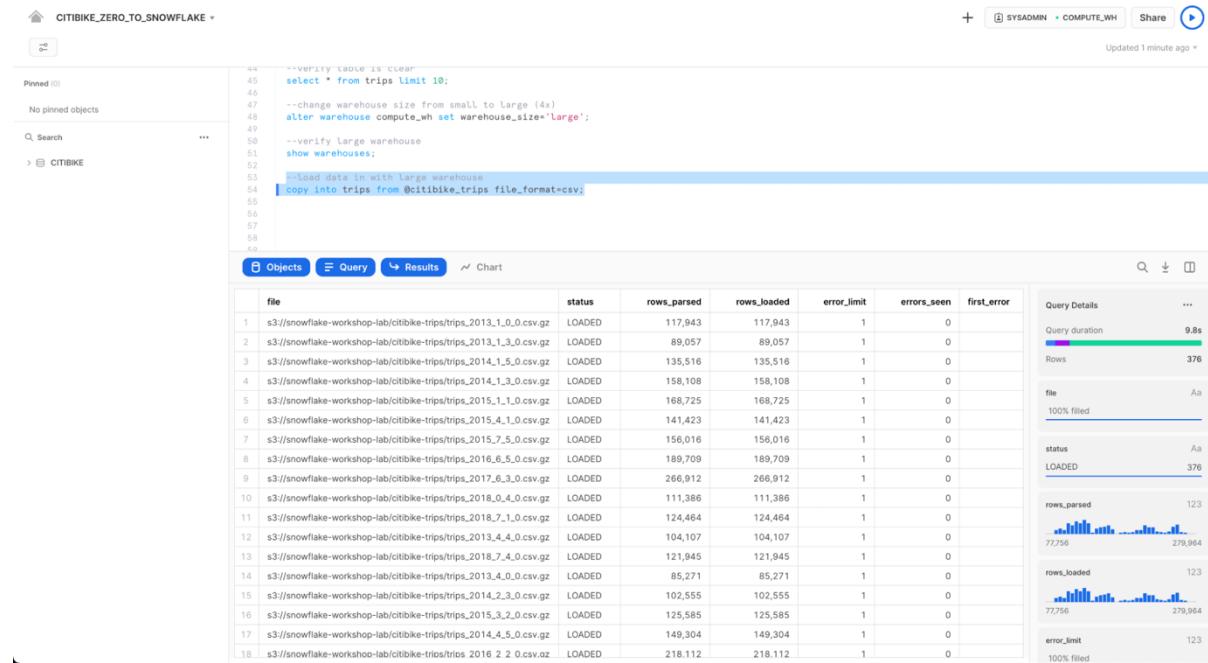
이제 성능의 향상된 가상 웨어하우스를 활용해서 다시 데이터를 로딩합니다

```
/* 데이터를 로딩하기 전에 validation_mode로 데이터 로딩 시 발생할 에러를 미리
체크할 수 있습니다. */
copy into trips from @citibike_trips file_format=csv pattern='.*csv.*'
validation_mode=return_all_errors;

/* citibike_trips 외부 스테이지에 있는 데이터 파일을 csv 파일 포맷에 맞춰서 trips
테이블에 로딩합니다. */
copy into trips from @citibike_trips file_format=csv pattern='.*CSV.*' ;
```

```
/* 테이블에 로딩된 데이터를 확인합니다.*/
select * from trips sample (50 rows);
```

- Validating Staged Files: <https://docs.snowflake.com/en/sql-reference/sql/copy-into-table.html#validating-staged-files>



로드가 완료되면 **Queries** 페이지로 다시 이동합니다(**Home** 아이콘 > **Activity** > **Query History**). 두 COPY INTO 명령의 시간을 비교하십시오.

Transforming Data During a Load

Copy into <Table> 명령은 Copy Into <Table> From (SELECT 구문)를 통해서 데이터 파일에서 데이터를 로드하기 전에 변환 작업을 진행할 수 있습니다.

```
/* 데이터 파일에서 일부 데이터를 선별적으로 테이블을 만들기 위해서 새로운 테이블을 생성합니다. 원본 데이터에 없는 tripid라는 새로운 컬럼도 구성합니다. */
create or replace table trips_agg
(tripid number autoincrement,
tripduration integer,
start_station_name string,
end_station_name string,
bikeid integer);

/* 테이블의 각 컬럼 정보를 확인합니다. */
desc table trips_agg;

/* 데이터 파일에서 필요한 컬럼만 추출해서 테이블로 로딩하고 tripid는 자동으로 생성되도록 합니다. */
copy into trips_agg(tripduration, start_station_name, end_station_name, bikeid)
from (select t.$1, t.$5, t.$9, t.$12 from @citibike_trips t)
file_format=csv pattern='.*csv.*';

select * from trips_agg limit 20;

/* 데이터 파일에서 일부 데이터를 전처리해서 로딩하기 위해서 새로운 테이블을 생성합니다. 원본 데이터에 없는 tripid라는 새로운 컬럼도 구성하고 bikeid는 스트링 타입으로 저장하고 membership에는 NULL 값이 없도록 저장하고자 합니다. */
create or replace table trips_cust
(tripid number autoincrement,
tripduration integer,
start_station_name string,
```

```
end_station_name string,  
bikeid string,  
membership_type string);  
  
desc table trips_cust;  
  
/* 데이터 파일에서 필요한 컬럼만 추출하고 해당하는 function을 통해서 데이터 로  
딩 전에 변환 작업을 수행합니다. */  
copy into trips_cust (tripduration, start_station_name, end_station_name, bikeid,  
membership_type)  
from (select t.$1, t.$5, t.$9, to_varchar(t.$12), ifnull(t.$13,'Free Membership')  
from @citibike_trips t)  
file_format=csv pattern='.*csv.*';  
  
select * from trips_cust limit 20;
```

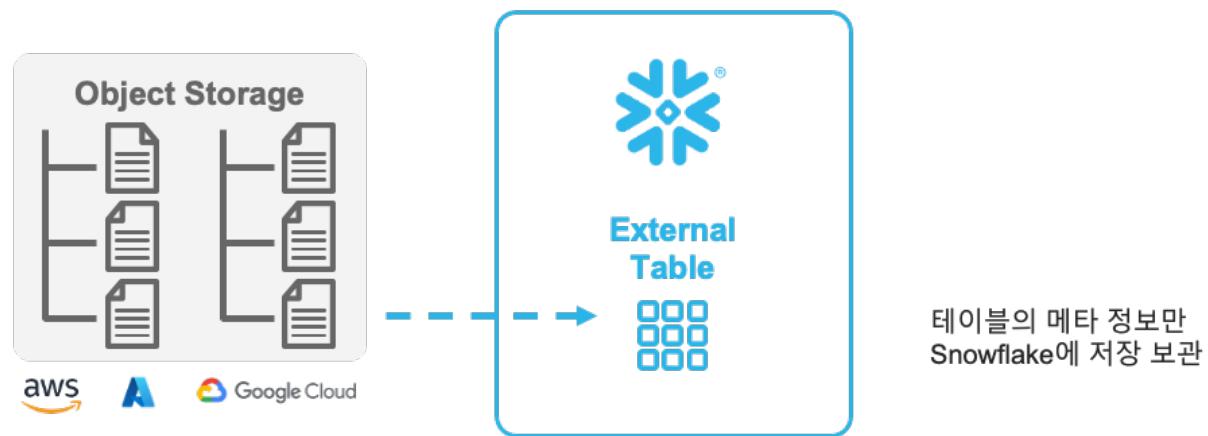
- 로드 중 데이터 변환하기: <https://docs.snowflake.com/ko/user-guide/data-load-transform.html#transforming-csv-data>

Create External Table

외부 테이블은 데이터 레이크를 Snowflake와 통합하기 위한 효과적인 방법입니다. 일반적인 테이블에서 데이터는 데이터베이스에 저장되지만, 외부 테이블에서 데이터는 외부 스테이지(External Stage)의 파일에 저장됩니다. 외부 테이블은 파일 이름, 버전 식별자 및 관련 속성과 같은 데이터 파일에 대한 파일 수준 메타데이터를 저장합니다. 이를 통해 데이터베이스 내부에 있는 것처럼 외부 스테이지에서 파일에 저장된 데이터를 쿼리할 수 있습니다. 외부 테이블은 `COPY INTO <테이블>` 문에서 지원하는 모든 형식으로 저장된 데이터에 액세스할 수 있습니다.

외부 테이블은 읽기 전용이므로 DML 작업을 수행할 수 없지만, 쿼리 및 조인 작업에는 외부 테이블을 사용할 수 있습니다. 외부 테이블에 대해 뷰를 생성할 수 있습니다.

데이터베이스 외부에 저장된 데이터를 쿼리하는 것은 기본 데이터베이스 테이블을 쿼리하는 것보다 느릴 수 있지만, 외부 테이블을 기반으로 하는 Materialized View는 쿼리 성능을 향상시킬 수 있습니다.



- 외부 테이블 소개: <https://docs.snowflake.com/ko/user-guide/tables-external-intro.html>

```
/* 외부 스테이지에 있는 경로와 파일 이름을 확인합니다. */
select metadata$filename from @citibike_trips;

/* 앞에서 지정한 file_format에 따라 데이터 파일의 컬럼을 해석해서 외부 테이블에
대한 메타데이터만 저장합니다. 기본적으로 데이터는 variant 타입의 value 컬럼에
저장이 되므로 여기서 컬럼을 순서대로 추출해서 형변환을 하는 형태로 외부 테이블
의 컬럼을 정의합니다. */

create or replace external table trips_ext
(tripduration integer as (value:c1::integer),
starttime timestamp as (value:c2::timestamp),
stoptime timestamp as (value:c3::timestamp),
start_station_id integer as (value:c4::integer),
start_station_name string as (value:c5::string),
start_station_latitude float as (value:c6::float),
start_station_longitude float as (value:c7::float),
end_station_id integer as (value:c8::integer),
end_station_name string as (value:c9::string),
end_station_latitude float as (value:c10::float),
end_station_longitude float as (value:c11::float),
bikeid integer as (value:c12::integer),
membership_type string as (value:c13::string),
usertype string as (value:c14::string),
birth_year integer as (value:c15::integer),
gender integer as (value:c16::integer))
location=@citibike_trips/
auto_refresh=false
file_format=csv;

/* 수동으로 refresh를 실행해서 최신 업데이트를 반영합니다. */
alter external table trips_ext refresh;
```

```
/* 외부 테이블에 쿼리를 바로 실행해서 결과를 확인합니다. 먼저 value 컬럼에 데이터가 어떻게 들어 있는지 확인합니다. */
select value from trips_ext limit 10;

/* 일반 컬럼들을 확인합니다. */
select tripduration, start_station_name, end_station_name from trips_ext limit 10;

/* 성능을 향상시키기 위해서 쿼리 결과를 저장하는 Materialized View를 생성합니다. */
create materialized view trips_mat as
select tripduration, start_station_name, end_station_name, bikeid from trips_ext
where tripduration > 10;

/* trips_mat view에 쿼리를 실행해서 결과를 확인합니다. */
select * from trips_mat limit 100;
```

- Create Materialized View: <https://docs.snowflake.com/ko/sql-reference/sql/create-materialized-view.html>

Continuous Load with Snowpipe

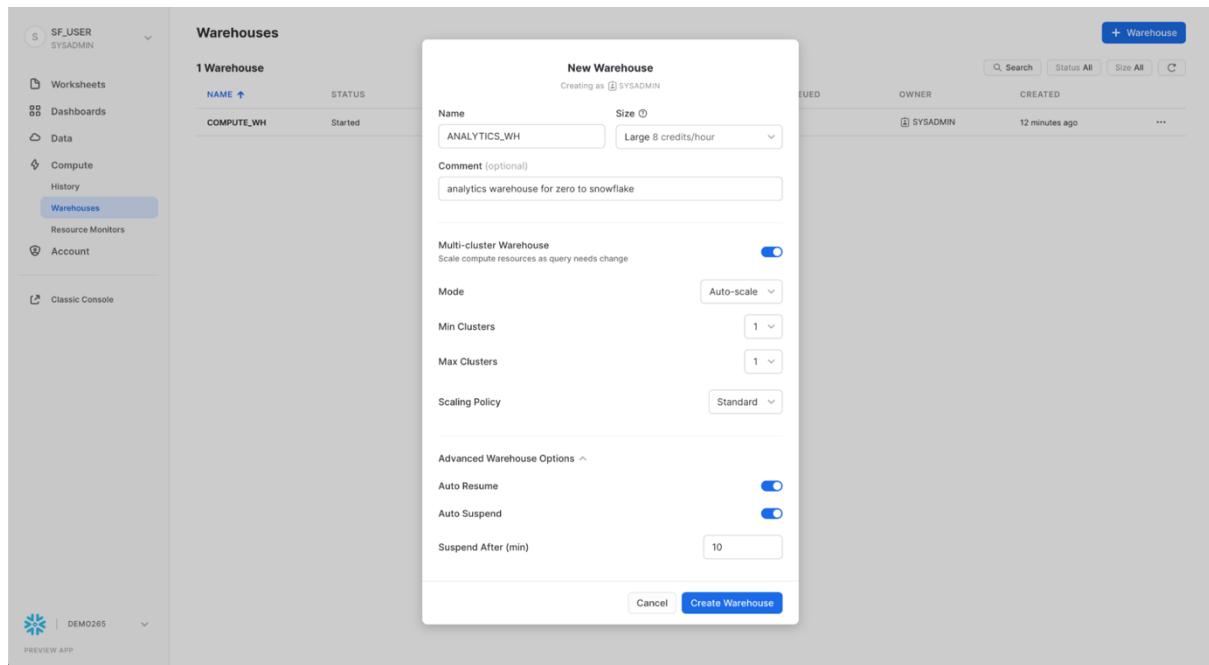
- Snowpipe를 이용한 연속 로드: <https://docs.snowflake.com/ko/user-guide/data-load-snowpipe.html>
- Snowpipe 시작하기 핸즈온 코스:
https://quickstarts.snowflake.com/guide/getting_started_with_snowpipe/index.html#0
- Kafka Connector로 데이터 로딩하기: <https://docs.snowflake.com/ko/user-guide/kafka-connector-overview.html>

4. Database Objects and Working with Queries

데이터 분석을 위한 가상 웨어하우스 생성 및 워크로드 격리

데이터 로드를 위한 가상 웨어하우스를 통해서 데이터 로딩 작업은 진행하고 있기 때문에, 데이터 분석을 실행하기 위한 새로운 가상 웨어하우스를 생성해서 분석 작업의 워크로드를 격리합니다.

Compute > Warehouses 탭으로 이동하여 + Warehouse를 클릭하고 새 웨어하우스의 이름을 ANALYTICS_WH로 지정하고 크기를 Large로 설정합니다.



SQL로 분석용 가상 웨어하우스 생성 및 설정

```
/* 데이터 분석용으로 large 크기의 새로운 가상 웨어하우스를 생성합니다. 5분 이상  
쿼리 요청이 없으면 자동으로 suspend되도록 설정을 하고 쿼리 요청이 들어 오면  
깨어 나서 요청을 처리하는 형태로 비용을 최적화합니다. */
```

```
create or replace warehouse analytics_wh  
warehouse_size='large'  
auto_suspend=300  
auto_resume=true;
```

- Create Warehouse: <https://docs.snowflake.com/ko/sql-reference/sql/create-warehouse.html>

데이터 분석하기

실제로는 분석 사용자가 SYSADMIN이 아닌 다른 역할을 수행할 수도 있습니다. 뒤 세션에서 RBAC(Role Based Access Control)에 대해서 더 알아 보도록 하겠습니다. 그리고 일반적으로 Tableau, Looker, PowerBI 등과 같은 비즈니스 인텔리전스 제품을 통해서 많은 분석이 이루어 집니다. 고급 분석을 위해 Datarobot, Dataiku, AWS Sagemaker와 같은 데이터 과학 도구나 기타 광범위한 파트너 에코시스템의 솔루션들이 Snowflake를 네이티브하게 지원합니다.

JDBC/ODBC, Spark 또는 Python을 활용하는 모든 기술이 Snowflake의 데이터에 대한 분석을 실행할 수 있습니다.

- 커넥터 & 드라이버: <https://docs.snowflake.com/ko/user-guide/conns-drivers.html>

작업 중인 워크시트로 이동하여 마지막 섹션에서 생성한 새 웨어하우스를 사용하도록 웨어하우스를 변경합니다. 워크시트 컨텍스트는 다음과 같아야 합니다.

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

= PUBLIC

The screenshot shows the Snowflake interface with the following details:

- Role: SYSADMIN (highlighted with a red box)
- Warehouse: ANALYTICS_WH (highlighted with a red box)
- Schema: PUBLIC (highlighted with a red box)

A code editor window on the left contains the following SQL query:1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
CITIBIKE.PUBLIC

1 create or replace table trips (2 tripduration integer,3 starttime timestamp,4 stoptime timestamp,5 start_station_id integer,6 start_station_name string,7 start_station_latitude float,8 start_station_longitude float,9 end_station_id integer,10 end_station_name string,11 and station name string);

아래의 쿼리를 실행하여 trips 데이터 샘플을 확인합니다.

```
/* 데이터 분석용 컨텍스트를 확인합니다. 웨어하우스는 분석용 웨어하우스를 사용하도록 하겠습니다. */
```

```
use role sysadmin;  
use warehouse analytics_wh;  
use database citibike;  
use schema public;  
  
select * from trips limit 20;
```

```

40
41 --clear table and metadata on table trips for next test
42
43
44
45 select * from trips limit 10;
46
47 --change warehouse size from small to large (4x)
48 alter warehouse compute_wh set warehouse_size='large';
49
50 --verify large warehouse
51 show warehouses;
52
53 --load data in with large warehouse
54 copy into trips from @citibike_trips file_format=csv;
55
56 --preview trips data
57 select * from trips limit 20;
58
59
60

```

	TRIPDURATION	STARTTIME	STOPTIME	START_STATION_ID	START_STATION_NAME	START_STATION_LATLNG
1	258	2018-04-03 18:31:11.000	2018-04-03 18:35:30.000	3,664	North Moore St & Greenwich St	40.720195;
2	659	2018-04-03 18:31:12.000	2018-04-03 18:42:12.000	127	Barrow St & Hudson St	40.731724;
3	1,628	2018-04-03 18:31:13.000	2018-04-03 18:58:22.000	3,002	South End Ave & Liberty St	40.711!
4	286	2018-04-03 18:31:18.000	2018-04-03 18:36:05.000	465	Broadway & W 41 St	40.75513!
5	1,097	2018-04-03 18:31:19.000	2018-04-03 18:49:36.000	305	E 58 St & 3 Ave	40.76095;
6	346	2018-04-03 18:31:19.000	2018-04-03 18:37:06.000	526	E 33 St & 5 Ave	40.74765!
7	313	2018-04-03 18:31:22.000	2018-04-03 18:36:35.000	519	Pershing Square North	40.751!
8	421	2018-04-03 18:31:24.000	2018-04-03 18:38:26.000	3,258	W 27 St & 10 Ave	40.750181!
9	1,185	2018-04-03 18:31:25.000	2018-04-03 18:51:10.000	3,263	Cooper Square & Astor Pl	40.729514!
10	272	2018-04-03 18:31:26.000	2018-04-03 18:35:59.000	3,140	1 Ave & E 78 St	40.77140!
11	509	2018-04-03 18:31:27.000	2018-04-03 18:39:56.000	528	2 Ave & E 31 St	40.74290!
12	454	2018-04-03 18:31:29.000	2018-04-03 18:39:04.000	248	Laight St & Hudson St	40.72185!
13	2,266	2018-04-03 18:31:34.000	2018-04-03 19:09:20.000	334	W 20 St & 7 Ave	40.74238!
14	358	2018-04-03 18:31:34.000	2018-04-03 18:37:32.000	418	Front St & Gold St	40.70;

Snowflake는 기본적으로 표준 SQL 기반의 쿼리를 제공하므로 Snowflake SQL을 통해서 데이터 분석을 진행할 수 있습니다.

- Snowflake SQL 쿼리 구문: <https://docs.snowflake.com/ko/sql-reference/constructs.html>
- Snowflake SQL 명령 요약: <https://docs.snowflake.com/ko/sql-reference/intro-summary-sql.html>

시간대 별로 이용 현황, 평균 이동 시간 및 평균 이동 거리 등 기본적인 집계는 기존 SQL 구문의 group by를 활용해서 진행할 수 있으며, 다른 분석 쿼리들도 익숙한 SQL 구문에 따라서 다양하게 테스트해 볼 수 있습니다.

```

/* 자전거를 대여한 시간에서 시간대를 추출해서 각 시간대 별로 이용 현황을 집계 합니다. 데이터 분석용 컨텍스트를 확인합니다.

아래는 하버사인 삼각함수를 통해서 위도 경도 데이터로 대략적인 이동 거리를 계산 했지만 그 외에 다양한 지리 정보 함수를 함께 제공합니다. */

```

```

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;
```

/* 자전거 트립이 많고 이동 거리가 길었던 시간대를 집계합니다. 여기서는 지리 공간 함수를 사용해서 평균 이동 거리를 계산했습니다. */

```
select date_trunc('hour', starttime) as "date",  
count(*) as "num trips",  
avg(tripduration)/60 as "avg duration (mins)",  
avg(st_distance(st_makepoint(start_station_latitude, start_station_longitude),  
st_makepoint(end_station_latitude, end_station_longitude)) / 1000) as "avg  
distance (km)"  
from trips  
group by 1  
having "num trips" > 100 and "avg distance (km)" > 1.5  
order by 1;
```

/* 2013년 아래로 가장 인기 있는 루트를 집계합니다. */

```
select  
start_station_name,  
end_station_name,  
count(*) as "num trips"  
from trips  
where starttime > '2013'  
group by 1, 2 order by 3 desc;
```

- Snowflake집계 함수: <https://docs.snowflake.com/ko/sql-reference/functions-aggregation.html>

- 지리 공간 함수: <https://docs.snowflake.com/ko/sql-reference/functions-geospatial.html>

Result Cache

Snowflake에는 지난 24시간 동안 실행된 모든 쿼리의 결과를 보유하고 있는 결과 캐시가 있습니다. 이는 웨어하우스 전반에 걸쳐 사용할 수 있으므로 기본 데이터가 변경되지 않았다면, 한 사용자에게 반환된 쿼리 결과를 동일한 쿼리를 실행하는 해당 시스템의 다른 사용자가 사용할 수 있습니다. 이러한 반복 쿼리는 매우 빠르게 반환될 뿐만 아니라 컴퓨팅 크레딧도 전혀 사용하지 않습니다.

```
/* 처음 실행되는 쿼리는 시간과 동일한 쿼리를 결과 캐쉬에서 제공하는 시간을 비교해서 확인합니다. */
```

```
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;
```

결과가 캐시되었기 때문에 두 번째 쿼리가 훨씬 더 빠르게 실행되었음을 오른쪽 **Query Details** 창에서 확인합니다.

```

Pinned 0
No pinned objects
Q Search ...
> CITIBIKE
61 count(*) as "num trips"
62 avg(tripduration)/60 as "avg duration (mins)",
63 avg(haversine(start_station_latitude, start_station_longitude, end_station_latitude, end_station_longitude)) as "avg distance (km)"
64
65 from trips
66 group by 1 order by 1;
67
68 --run same query again to see performance improvement due to cache
69 select date, sum(num_trips) as "num trips",
70 count(*) as "num trips",
71 avg(tripduration)/60 as "avg duration (mins)",
72 avg(haversine(start_station_latitude, start_station_longitude, end_station_latitude, end_station_longitude)) as "avg distance (km)"
73 from trips
74 group by 1 order by 1;
75
76
77
78
79
80

```

Objects Query Results Chart

	date	num trips	...	avg duration (mins)	avg distance (km)
1	2013-06-01 00:00:00	152		56.058442983333	2.127971476
2	2013-06-01 01:00:00	102		26.5251634	2.067906273
3	2013-06-01 02:00:00	67		36.1199005	2.31784827
4	2013-06-01 03:00:00	41		44.48536585	2.349126632
5	2013-06-01 04:00:00	16		23.278125	1.840026007
6	2013-06-01 05:00:00	13		34.584615383333	3.337844489
7	2013-06-01 06:00:00	40		22.3975	2.832927896
8	2013-06-01 07:00:00	93		66.397849466667	2.691774983
9	2013-06-01 08:00:00	177		18.530225983333	2.244399992
10	2013-06-01 09:00:00	280		40.2610119	2.292639556
11	2013-06-01 10:00:00	375		28.673466866667	2.22578826
12	2013-06-01 11:00:00	473		35.63520085	2.26058226
13	2013-06-01 12:00:00	604		33.8763521	2.177910979
14	2013-06-01 13:00:00	638		48.623093	2.270395192
15	2013-06-01 14:00:00	688		34.7400436	2.271981468

Zero-Copy Clone

Snowflake를 사용하면 "제로 카피 클론"이라고도 하는 테이블, 스키마 및 데이터베이스의 클론을 몇 초 안에 생성할 수 있습니다. 클론을 생성할 때 원본 객체에 있는 데이터의 스냅샷을 찍으며 복제된 객체에서 이를 사용할 수 있습니다. 복제된 객체는 쓰기 가능하고 클론 원본과는 독립적입니다. 따라서 원본 객체 또는 클론 객체 중 하나에 이뤄진 변경은 다른 객체에는 포함되지 않습니다.

제로 카피 클론 생성의 일반적인 사용 사례는 개발 및 테스팅이 사용하는 운영 환경을 복제하여 운영 환경에 부정적인 영향을 미치지 않게 두 개의 별도 환경을 설정하여 관리할 필요가 없도록 테스트하고 실험하는 것입니다.

워크시트에서 다음 명령을 실행하여 `trips` 테이블의 개발(dev) 테이블 복제본을 만듭니다.

```
/* 운영 환경의 trips 테이블을 클로닝해서 데이터의 복사 없이 새로운 개발 테이블을 생성합니다. */
```

```
create table trips_dev clone trips
```

CITIBIKE 데이터베이스 아래의 개체 트리를 확장하고 trips_dev라는 새 테이블이 표시되는지 확인합니다. 이제 개발 팀은 trips 테이블이나 다른 개체에 영향을 주지 않고 이 테이블을 사용하여 업데이트 또는 삭제를 포함하여 원하는 모든 작업을 수행할 수 있습니다.

The screenshot shows the Snowflake UI interface. On the left, there is a sidebar with pinned objects, a search bar, and a tree view of the database schema. The schema tree shows the CITIBIKE database, which contains INFORMATION_SCHEMA and PUBLIC schemas. The PUBLIC schema has a TABLES node, which further contains TRIPS and TRIPS_DEV. The TRIPS_DEV table is selected. On the right, the main area displays the SQL code for creating the TRIPS_DEV table:

```
67      -- same query arguments
68      select date_trunc('hour', starttime) as hour,
69      count(*) as "num trips",
70      avg(tripduration)/60 as "avg duration",
71      avg(haversine(start_station_longitude,
72      group by 1 order by 1;
73
74      --find busiest months
75      select
76          monthname(starttime) as "month",
77          count(*) as "num trips"
78      from trips
79      group by 1 order by 2 desc;
80
81      --create dev table
82
```

A context menu is open over the TRIPS_DEV table, with the 'Clone' option highlighted. A tooltip for the 'Clone' button is visible, showing the following information:

Number of rows	61.5M
Size	1.9GB
Cluster Key	—
Owner	SYSADMIN
Created	1 minute ago

- Create Clone: <https://docs.snowflake.com/ko/sql-reference/sql/create-clone.html>

Semi-Structured Data and Working with Queries

날씨가 자전거 이용 횟수에 어떻게 영향을 미치는지 확인하고자 하기 위해서 JSON 형식으로 저장된 반정형 데이터를 로딩 합니다.

- 공개된 S3 버킷에 보관된 JSON 형식의 날씨 데이터 로드
- 뷰 생성 및 SQL 점 표기법 (dot notation)을 사용해 반정형 데이터를 쿼리
- JSON 데이터를 이전에 로드된 TRIPS 데이터에 조인하는 쿼리를 실행
- 날씨 및 자전거 이용 횟수 데이터를 분석하여 관계 파악
-

JSON 데이터는 57,900행, 61개 객체 및 2.5MB 압축으로 이루어 진 압축된 JSON 문서로 AWS S3에 저장되어 있습니다.

```
{"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}}
```

반정형 데이터 처리하기

Snowflake는 변환 없이 JSON, Parquet, Avro, ORC, XML과 같은 반정형 데이터를 쉽게 로드하고 쿼리할 수 있습니다.

- 반정형 데이터 소개: <https://docs.snowflake.com/ko/user-guide/semistructured-intro.html>
- 반정형 데이터에 지원하는 형식: <https://docs.snowflake.com/ko/user-guide/semistructured-data-formats.html>
- 반정형 데이터 타입: <https://docs.snowflake.com/ko/sql-reference/data-types-semistructured.html>

데이터베이스 및 테이블 생성

먼저, 워크시트를 통해, 반정형 데이터를 저장하는 데 사용할 WEATHER라는 이름의 데이터베이스를 생성합니다.

```
/* 새 데이터베이스 생성하기 */  
create or replace database weather;
```

워크시트 내에 컨텍스트를 적절하게 설정합니다.

```
/* 테이블을 생성하기 위한 컨텍스트를 설정합니다. 새로 생성한 weather 데이터베이스의 public 스키마에서 작업을 합니다. */
```

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

JSON 데이터를 로딩하는 데 사용할 JSON_WEATHER_DATA라는 이름의 테이블을 생성합니다. Snowflake에는 VARIANT라는 반정형 데이터 타입을 네이티브하게 지원하는 컬럼 유형이 있어 이를 통해 전체 JSON 객체를 저장하고 바로 쿼리할 수 있습니다.

```
/* JSON 데이터를 로딩하려고 하기 때문에 반정형 데이터 타입을 네이티브하게 지원하는 variant 타입의 컬럼을 가진 테이블을 생성합니다. */
```

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

- Variant 타입: <https://docs.snowflake.com/ko/sql-reference/data-types-semistructured.html#variant>

VARIANT 데이터 유형을 사용하면 Snowflake가 스키마를 미리 정의하지 않고도 반정형 데이터를 수집할 수 있습니다.

워크시트 하단의 결과 창에서 JSON_WEATHER_DATA 테이블이 생성되었는지 확인합니다.

The screenshot shows the Snowflake Worksheet interface with the following details:

- Left Panel (Object Navigator):** Shows the database structure under the schema CITIBIKE. It includes objects like INFORMATION_SCHEMA, PUBLIC, and WEATHER, with the WEATHER schema expanded to show a single table named JSON_WEATHER_DATA.
- Top Bar:** Displays the current role (SYSADMIN), warehouse (COMPUTE_WH), and session information. It also shows the last update time: "Updated 31 seconds ago".
- Code Editor:** Contains the SQL command used to create the table:

```
--create database for unstructured weather data
create database weather;
--set context for module 7
use role sysadmin;
use warehouse compute_wh;
use database weather;
use schema public;

--create table for weather data using variant to handle semi structured data
create table json_weather_data (v variant);
```
- Results Tab:** Active tab, showing the status of the query execution.

status	
1	Table JSON_WEATHER_DATA successfully created.
- Right Panel (Query Details):** Provides performance metrics for the query.

Query Details	
Query duration	127ms
Rows	1
status	Aa
100% filled	

External Stage 생성

워크시트에서 다음 명령을 사용하여 AWS S3에서 반정형 JSON 데이터가 저장되는 버킷을 가리키는 외부 스테이지를 설정합니다.

```
/* public s3의 버킷을 가리키는 외부 스테이지를 생성합니다. */

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

LIST 명령을 통해서 파일 목록을 살펴보도록 하겠습니다.

압축된 JSON 파일들이 저장되어 있는 것을 확인할 수 있습니다.

```
list @nyc_weather;
/* 이 결과에서 바로 전체 사이즈와 평균 파일 사이즈를 알아 보도록 하겠습니다. */

select
    floor(sum($2) / power(1024, 2), 1) total_compressed_storage_mb,
    floor(avg($2) / power(1024, 1), 1) avg_file_size_kb,
    count(*) as num_files
from
    table(result_scan(last_query_id()));
```

테이블로 로딩하기 전에 실제로 데이터 파일로 저장되어 있는 데이터의 내용도 바로 살펴 보도록 하겠습니다.

```
/* 외부 스테이지에 있는 데이터 파일에 대해서 바로 쿼리를 실행해서 내용을 살펴  
볼 수 있습니다. JSON 타입으로 해석하는 File Format을 먼저 생성한 다음에 select  
문에서 설정을 해 줍니다. */  
create or replace file format my_json_format type = 'json';  
  
select $1  
From @nyc_weather/ (file_format => my_json_format)  
Limit 10;  
  
/* 메타 데이터도 함께 결과를 확인해 보겠습니다. Parse_json은 문자열을 Variant 형  
식으로 변환하는 유용한 함수입니다. */  
select  
    metadata$filename, metadata$file_row_number, parse_json($1)  
from  
    @nyc_weather/ (file_format => my_json_format);
```

이 섹션에서는 웨어하우스를 사용하여 S3 버킷의 데이터를 이전에 생성한 JSON_WEATHER_DATA 테이블로 로드합니다.

워크시트에서 아래 COPY 명령을 실행하여 데이터를 로드합니다.
SQL 명령에서도 FILE FORMAT 객체를 인라인으로 지정할 수 있는 방법을 사용할 수도 있습니다.

정형 데이터를 로드했던 이전 섹션에서 파일 형식을 자세하게 정의해야 했습니다.
여기에는 JSON 데이터는 형식이 잘 지정되어 있기 때문에 기본 설정을 사용해 간단하게 JSON 유형을 로딩하고 결과를 살펴 보겠습니다.

```
/* 먼저 Variant 타입의 컬럼에 JSON 데이터를 로딩하겠습니다. */
```

```

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

select * from json_weather_data limit 20;

```

결과에서 한 행을 클릭하여 오른쪽 패널에 형식 JSON을 표시해서 어떤 형태의 데이터인지 살펴 보겠습니다.

The screenshot shows the Snowflake UI interface. On the left, the sidebar displays the schema structure under the 'CITIBIKE' database, specifically the 'WEATHER' schema which contains a single table named 'JSON_WEATHER_DATA'. A query is run on this table:

```

create stage nyc_weather
url='s3://snowflake-workshop-Lab/weather-nyc';
--List out files in the stage
list @nyc_weather;
--load json data into table
copy into json_weather_data
from @nyc_weather
file_format=(type=json);
--view weather data
select * from json_weather_data limit 10;

```

The results of the query are displayed in a table on the left, showing 10 rows of JSON data. Each row represents a city record with fields like 'city', 'coord', 'country', and 'findname'. The right panel shows the detailed JSON structure for the first few rows. For example, the first row is:

```

{
  "city": {
    "coord": {
      "lat": 40.714272,
      "lon": -74.005966
    },
    "country": "US",
    "findname": "NEW YORK",
    "id": 5128
  }
}

```

This JSON structure represents a city record with latitude and longitude coordinates, country, findname, and an ID.

View 및 Materialized View

원본 테이블은 JSON 형식의 컬럼이기 때문에 뷰를 사용하면 쿼리 결과에 테이블처럼 액세스할 수 있습니다. Snowflake는 여러 가지 뷰의 형태를 제공하는 데 그 중에 Materialized View는 쿼리의 결과를 저장하기 때문에 저장 용량을 필요로 하지만 성능을 향상 시킬 수 있습니다. Snowflake 엔터프라이즈 이상에서 Materialized View를 사용할 수 있습니다.

- View의 개요: <https://docs.snowflake.com/ko/user-guide/views-introduction.html>
- Materialized View: <https://docs.snowflake.com/ko/user-guide/views-materialized.html>

먼저 반정형 JSON 날씨 데이터에 대한 뷰를 테이블 형식으로 구성해서 분석가들이 익숙한 형태로 쿼리를 할 수 있도록 하겠습니다.

city_id 5128638은 뉴욕시에 해당합니다.

```
/* 원본 테이블이 Variant 타입의 JSON 구조체이므로 분석가들에게 익숙한 테이블 형태의 뷰를 생성해서 분석을 용이하게 합니다.
```

```
JSON 구조체에서 계층에 따라서 .(닷)으로 원하는 컬럼을 구성하고 ::으로 형변환을 해 주는 형식으로 구성합니다. */
```

```
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;
```

예를 들어서, v:city.coord.lat은 이 명령에서 JSON 계층 구조 내 더 낮은 수준의 값을 가져오는 데 사용됩니다. 이는 각 필드를 관계형 테이블의 열인 것처럼 취급할 수 있도록 합니다.

새로운 뷰가 UI 왼쪽 WEATHER > PUBLIC > Views에 json_weather_data로 나타나도록 브라우저를 새로 고침을 합니다.

```

119 create view json_weather_data_view as
120 select
121   v:time::timestamp as observation_time,
122   v:city_id::int as city_id,
123   v:city_name::string as city_name,
124   v:city_country::string as country,
125   v:city.coord.lat::float as city_lat,
126   v:city.coord.lon::float as city_lon,
127   v:clouds.all::int as clouds,
128   (v:main.temp::float)-273.15 as temp_avg,
129   (v:main.temp_min::float)-273.15 as temp_min,
130   (v:main.temp_max::float)-273.15 as temp_max,
131   v:weather[0].main::string as weather_main,
132   v:weather[0].description::string as weather_desc,
133   v:weather[0].icon::string as weather_icon,
134   v:wind.deg::float as wind_dir,
135   v:wind.speed::float as wind_speed
136   from json_weather_data
137   where city_id = 512863;

```

JSON_WEATHER_DATA_VIEW

Type: View
Number of columns: 15
Owner: ASYSADMIN
Created: just now

successfully created.

Query Details
Query duration: 256ms
Rows: 1
status: 100% filled

/* 뷰의 내용을 다른 테이블 형식의 데이터처럼 확인을 합니다. */

```

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

```

```

176   v:city.coord.lon::float as city_lon,
177   v:clouds.all::int as clouds,
178   (v:main.temp::float)-273.15 as temp_avg,
179   (v:main.temp_min::float)-273.15 as temp_min,
180   (v:main.temp_max::float)-273.15 as temp_max,
181   v:weather[0].main::string as weather_desc,
182   v:weather[0].icon::string as weather_icon,
183   v:wind.deg::float as wind_dir,
184   v:wind.speed::float as wind_speed
185   from json_weather_data
186   where city_id = 5128638;
187
188   -- validate the view created
189   select * from json_weather_data_view
190   where date_trunc('month',observation_time) = '2018-01-01'
191   limit 20;
192
193
194

```

Objects **Query** **Results** **Chart**

	OBSERVATION_TIME	CITY_ID	CITY_NAME	COUNTRY	CITY_LAT	CITY_LON	LOUDS	TEMP_AVG	TEMP
1	2018-01-01 03:05:19.000	5,128,638	New York	US	43.000351	-75.499901	1	-20.85	
2	2018-01-01 04:02:30.000	5,128,638	New York	US	43.000351	-75.499901	1	-21.38	
3	2018-01-08 18:02:41.000	5,128,638	New York	US	43.000351	-75.499901	90	0.1	
4	2018-01-08 19:05:05.000	5,128,638	New York	US	43.000351	-75.499901	90	1.05	
5	2018-01-12 20:03:23.000	5,128,638	New York	US	43.000351	-75.499901	90	14.57	
6	2018-01-12 21:05:14.000	5,128,638	New York	US	43.000351	-75.499901	90	8.85	
7	2018-01-16 01:02:38.000	5,128,638	New York	US	43.000351	-75.499901	90	-8.52	
8	2018-01-16 02:05:09.000	5,128,638	New York	US	43.000351	-75.499901	90	-8	
9	2018-01-25 14:05:13.000	5,128,638	New York	US	43.000351	-75.499901	1	-11.95	
10	2018-01-25 15:02:40.000	5,128,638	New York	US	43.000351	-75.499901	1	-10.95	
11	2018-01-04 15:02:41.000	5,128,638	New York	US	43.000351	-75.499901	90	-9.48	
12	2018-01-04 16:05:25.000	5,128,638	New York	US	43.000351	-75.499901	90	-8.95	
13	2018-01-03 03:04:00.000	5,128,638	New York	US	43.000351	-75.499901	90	-8.53	
14	2018-01-03 04:03:27.000	5,128,638	New York	US	43.000351	-75.499901	90	-8	
15	2018-01-04 23:05:18.000	5,128,638	New York	US	43.000351	-75.499901	90	-7.47	

Query Details

- Query duration: 1.1s
- Rows: 20

OBSERVATION_TIME

CITY_ID: 123 (100% filled)

CITY_NAME: New York (20)

COUNTRY: US (20)

JOIN 연산

이제 JSON 날씨 데이터를 CITIBIKE.PUBLIC.TRIPS 데이터에 조인하여 원래 질문인 날씨가 자전거 이용 횟수에 어떻게 영향을 미치는지에 대해 살펴 보겠습니다.

아직 워크시트에 있기 때문에 WEATHER 데이터베이스가 기본값입니다. 데이터베이스와 스키마 이름을 제공하여 TRIPS 테이블에 대해 완전히 참조할 수 있도록 합니다.

```
/* trips 테이블에 weather view를 left outer join으로 시간대 별로 조인해서 날씨 조  
건 별로 전체 트립의 개수를 집계하면 각 날씨에 따른 이용 횟수의 관계를 알아 볼  
수 있습니다. */
```

```
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;
```

```
/* 이 연산의 결과를 Secure view로 생성해서 안전하게 공유하는 데 활용할 수도 있  
습니다. Secure view는 어떤 SQL 구문을 통해서 이 결과가 나왔는지에 대한 DDL 정  
보를 권한이 있는 사용자(예, 소유자)만 열람할 수 있습니다. */
```

```
create or replace secure view weather_trips_sv as  
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;  
  
select * from weather_trips_sv;
```

- Secure View 관련 작업하기: <https://docs.snowflake.com/ko/user-guide/views-secure.html>

```

138 --validate the view created
139 select * from json_weather_data_view
140 where date_trunc('month',observation_time) = '2018-01-01'
141 limit 20;
142
143
144 --join weather data and trips data
145 select weather.conditions
146 ,count(*) as num_trips
147 from citibike.public.trips
148 left outer join json_weather_data_view
149 on date_trunc('hour', observation_time) = date_trunc('hour', starttime)
150 where conditions is not null
151 group by 1 order by 2 desc;
152
153
154
155
156
157

```

CONDITIONS	NUM_TRIPS
1 Clear	9,249,531
2 Clouds	8,934,964
3 Rain	3,206,720
4 Snow	1,380,418
5 Mist	798,487
6 Fog	362,496
7 Thunderstorm	230,539
8 Drizzle	98,779
9 Haze	40,724
10 Smoke	2,871

Query Details

- Query duration: 1.3s
- Rows: 10

CONDITIONS

100% filled

NUM_TRIPS

123

2,871 9,249,531

5. Data Protection, RBAC and Managing Account

Time Travel

타임 트래블 기능으로 사전 구성 가능한 기간 내 어느 시점이든 데이터에 액세스할 수 있습니다. 기본 기간은 24시간이며 Snowflake 엔터프라이즈 에디션으로는 90일 까지 가능합니다.

몇 가지 유용한 적용례는 다음과 같습니다.

- 삭제되었을 수도 있는 테이블, 스키마 및 데이터베이스 같은 데이터 관련 객체를 복구
- 과거의 주요 시점으로부터 데이터를 복제하고 백업
- 데이터 사용을 분석하고 특정 기간에 대해 히스토리 분석

Continuous Data Protection Lifecycle



먼저 실수로 또는 의도적으로 삭제한 데이터 객체를 어떻게 복구할 수 있는지 살펴보겠습니다.

워크시트에서 다음의 DROP 명령을 실행하여 `JSON_WEATHER_DATA` 테이블을 제거합니다.

```
drop table json_weather_data;
```

`json_weather_data` 테이블에서 `SELECT` 문을 실행합니다. 기본 테이블이 삭제되었기 때문에 결과 창에 오류가 나타나야 합니다.

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

The screenshot shows the Snowflake UI interface. On the left, the sidebar displays the database structure under 'CITIBIKE_ZERO_TO_SNOWFLAKE'. A query editor window is open with the following code:

```
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
```

The error message 'Object 'JSON_WEATHER_DATA' does not exist or not authorized.' is displayed below the query editor.

이제 이 테이블을 다음과 같이 복구합니다.

```
undrop table json_weather_data;
```

json_weather_data 테이블이 복구된 것을 확인합니다.

```
select * from json_weather_data_view limit 10;
```

The screenshot shows the Snowflake UI interface. A query editor window is open with the following code:

```
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
275
276
277
278
279
279
280
281
282
283
284
284
285
286
287
287
288
289
289
290
290
291
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345

```

테이블을 이전 상태로 룰백하여 CITIBIKE 데이터베이스의 TRIPS 테이블에 있는 모든 스테이션 이름을 "mistake"라는 단어로 대체하는 의도하지 않은 DML 오류를 수정하겠습니다.

먼저 워크시트의 컨텍스트가 적절한지 다음과 같이 확인합니다.

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

다음의 명령을 실행하여 테이블의 모든 스테이션 이름을 "mistake"라는 단어로 대체 합니다.

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

이제 자전거 이용 횟수별로 상위 20개 스테이션을 반환하는 쿼리를 실행합니다. 스테이션 이름 결과는 단 하나의 행으로 나오는 것을 확인할 수 있습니다.

```
select  
start_station_name as "station",  
count(*) as "rides"  
from trips  
group by 1  
order by 2 desc  
limit 20;
```

```

167 select * from json_weather_data_view limit 10;
168 --set context
169 use role sysadmin;
170 use warehouse compute_wh;
171 use database citibike;
172 use schema public;
173
174
175 --make a mistake by changing the column name
176 update trips set start_station_name='oops';
177
178 --query for top 20 stations by total number of rides
179 select
180   start_station_name as "station",
181   count(*) as "rides"
182   from trips
183   group by 1
184   order by 2 desc
185   limit 20;
186

```

station	rides
oops	61,468,359

Query Details

- Query duration: 315ms
- Rows: 1

station

- 100% filled

rides

- 123
- 100% filled

백업을 하지 않은 상황이라도 Snowflake에서는 아래 처럼 마지막UPDATE 명령의 쿼리 ID를 찾아 \$QUERY_ID라는 변수에 저장하면 됩니다.

```

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);

```

올바른 스테이션 이름을 가진 테이블을 다음과 같이 다시 만듭니다.

```

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

```

다음과 같이 SELECT 문을 다시 실행하여 스테이션 이름이 복구되었는지 확인합니다.

```

select
start_station_name as "station",
count(*) as "rides"
from trips
group by 1
order by 2 desc
limit 20;

```

```

187 --find the query id that was the last update to the table
188 set query_id
189 (select query_id from table(information_schema.query_history_by_session (result_limit>5)
190 where query_text like 'update%' order by start_time limit 1);
191
192 --recreate the table with proper station names
193 create or replace table trips as
194 (select * from trips before (statement => $Query_id));
195
196 select from the restored table to verify
197 select
198 start_station_name as "station",
199 count(*) as "rides"
200 from trips
201 group by 1
202 order by 2 desc
203
204 limit 20;
205

```

station	rides
Pershing Square North	491,951
E 17 St & Broadway	481,065
W 21 St & 6 Ave	458,626
8 Ave & W 31 St	438,001
West St & Chambers St	432,518
Broadway & E 22 St	421,812
Lafayette St & E 8 St	397,724
Broadway & E 14 St	384,995
8 Ave & W 33 St	379,843
W 41 St & 6 Ave	359,838
Cleveland Pk & Spring St	358,485
W 20 St & 11 Ave	352,099
Carmine St & 6 Ave	348,158
University Pl & E 14 St	332,803
Greenwich Ave & 8 Ave	329,347

- 타임 트래블 이해 및 사용하기: <https://docs.snowflake.com/ko/user-guide/data-time-travel.html>
- Fail-Safe 이해 및 보기: <https://docs.snowflake.com/ko/user-guide/data-failsafe.html>

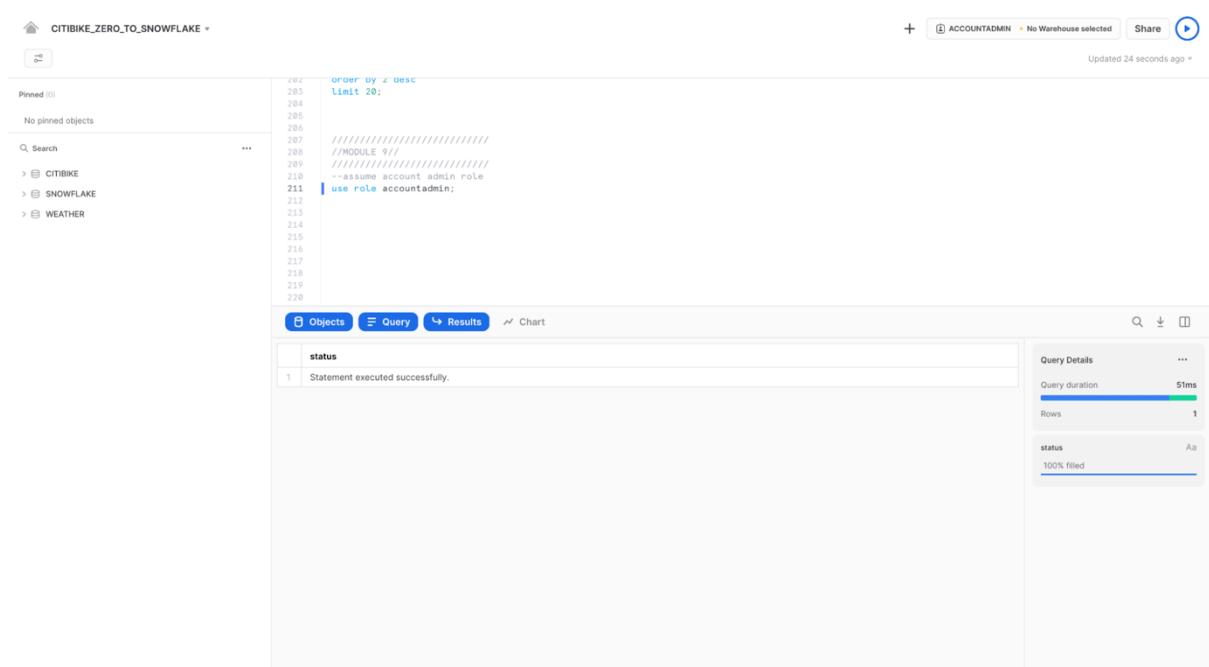
Role Based Access Control

이 섹션에서는 새로운 역할 생성 및 특정 권한 부여와 같은 Snowflake의 역할 기반 액세스 제어(RBAC) 측면을 살펴보고자 합니다. 또한 ACCOUNTADMIN(계정 관리자) 역할도 다뤄볼 것입니다.

주니어 DBA가 Citi Bike에 합류하여 시스템에서 정의한 기본 역할인 SYSADMIN보다 적은 권한을 지닌 새로운 역할을 만들고 싶다고 가정해 보겠습니다.

워크시트에서 ACCOUNTADMIN 역할을 새로운 역할로 전환합니다. ACCOUNTADMIN 은 SYSADMIN 및 SECURITYADMIN 시스템 정의 역할을 함께 가지고 있는 시스템의 최상위 역할이므로 계정에서 제한된 수의 사용자에게만 부여되어야 합니다. 워크시트에서 다음을 실행합니다.

```
use role accountadmin;
```



The screenshot shows the Snowflake UI interface. On the left, there's a sidebar with pinned objects and a search bar. The main area is a code editor with the following content:

```
order by z desc
limit 20;
//MODULE 9//
--assume account admin role
use role accountadmin;
```

Below the code editor, there are tabs for Objects, Query, Results, and Chart. The Results tab is selected, showing the message "Statement executed successfully." In the bottom right corner, there's a "Query Details" panel displaying the following information:

Query Details	
Query duration	51ms
Rows	1
status	Aa
100% filled	

역할을 액세스 제어에 사용하려면 최소한 한 명의 사용자가 역할에 할당되어야 합니다. 이제 JUNIOR_DBAs라는 새 역할을 만들고 Snowflake 사용자에게 할당해 보겠습니다. 이 작업을 완료하려면 UI에 로그인할 때 사용한 이름인 사용자 이름을 알아야 합니다.

다음 명령을 사용하여 역할을 생성하고 할당합니다. GRANT ROLE 명령을 실행하기 전에 <user>를 사용자 이름으로 바꿉니다.

```
create role junior_dba;
grant role junior_dba to user <user>;
```

SYSADMIN과 같은 역할로 이 작업을 수행하려고 하면, 권한이 부족하여 실패할 것입니다. 기본적으로 SYSADMIN 역할은 새로운 역할이나 사용자를 생성할 수 없습니다.

워크시트 컨텍스트를 다음과 같이 JUNIOR_DBA 역할로 변경합니다.

```
use role junior_dba;
```

The screenshot shows the Snowflake UI with a query editor. The database is set to CITIBIKE_ZERO_TO_SNOWFLAKE. The role is set to JUNIOR_DBA. The code in the editor is:

```
206 ///////////////////////////////////////////////////////////////////
207 //MODULE 9//
208 ///////////////////////////////////////////////////////////////////
209 --create account admin role
210 use role accountadmin;
211
212 --create new role and grant role to current user
213 create role junior_dba;
214 grant role junior_dba to user SF_USER;
215
216 --assume junior_dba role
217 use role junior_dba;
```

The line `use role junior_dba;` is highlighted with a blue background. A tooltip at the bottom right of the editor says "JUNIOR_DBA required to view results".

또한 새로 생성된 역할에는 웨어하우스에 대한 사용 권한이 없기 때문에 웨어하우스가 선택되지 않습니다. ADMIN 역할로 다시 전환하여 수정하고 COMPUTE_WH 웨어하우스에 사용 권한을 부여해 보겠습니다.

```
use role accountadmin;
```

```
grant usage on warehouse compute_wh to role junior_dba;
```

JUNIOR_DBA 역할로 다시 전환합니다. 이제 COMPUTE_WH를 사용할 수 있습니다.

```
use role junior_dba;  
  
use warehouse compute_wh;
```

마지막으로 왼쪽의 데이터베이스 개체 브라우저 패널에서 CITIBIKE 및 WEATHER 데이터베이스가 더 이상 나타나지 않는 것을 확인할 수 있습니다. 이는 JUNIOR_DBA 역할에 액세스 권한이 없기 때문입니다.

ACCOUNTADMIN 역할로 다시 전환하고 CITIBIKE 및 WEATHER 데이터베이스를 보고 사용하는 데 필요한 USAGE 권한을 JUNIOR_DBA에 부여합니다.

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

JUNIOR_DBA 역할로 다음과 같이 전환합니다.

```
use role junior_dba;
```

이제 CITIBIKE 및 WEATHER 데이터베이스가 나타나는지 확인하십시오. 나타나지 않는다면 새로 고침 아이콘을 클릭하여 시도하십시오.

CITIBIKE_ZERO_TO_SNOWFLAKE

JUNIOR_DBA • No Warehouse selected Share Run as... Updated 34 seconds ago

Search ...

Objects unavailable. Select a different role.

CITIBIKE WEATHER

```
use role junior_dba;
--change role to give permissions to junior_dba role
use role accountadmin;
grant usage on database citibike to junior_dba;
grant usage on database weather to junior_dba;
--switch back to junior_dba
use role junior_dba;
```

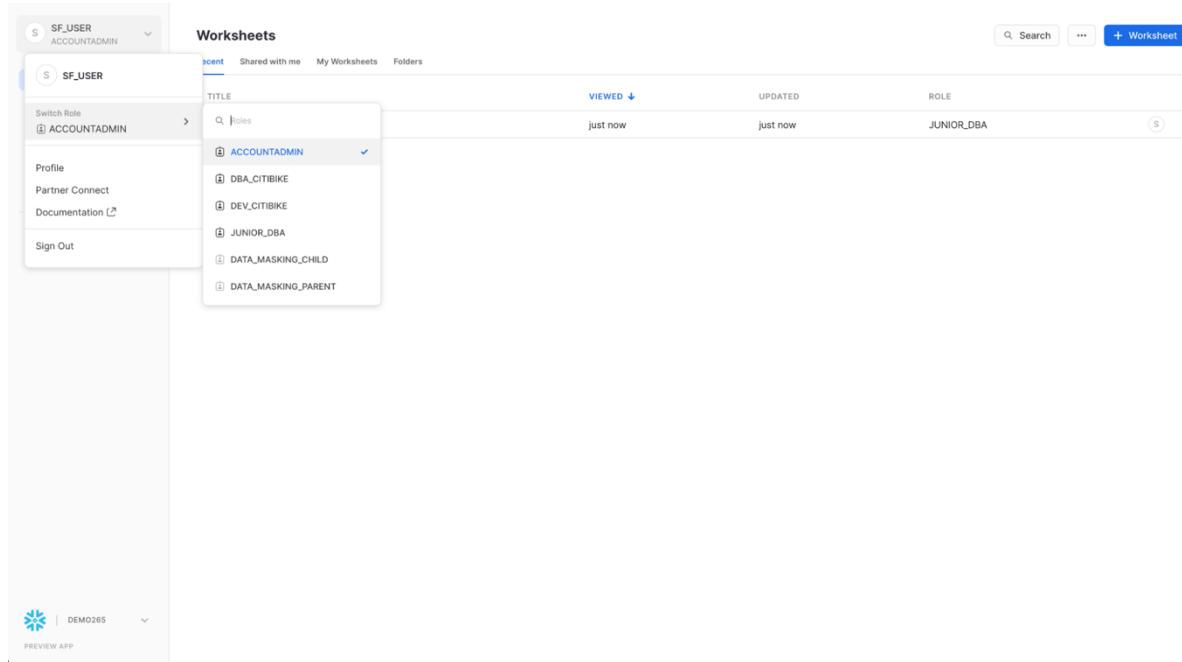
Objects Query Results Chart

JUNIOR_DBA required to view results

ACCOUNTADMIN

이 역할만 액세스할 수 있는 UI의 다른 영역을 보려면 액세스 제어 역할을 다시 ACCOUNTADMIN으로 변경합니다. 그러나 이 작업을 수행하려면 워크시트 대신 UI를 사용하십시오.

먼저 워크시트의 왼쪽 상단 모서리에 있는 **Home** 아이콘을 클릭합니다. 그런 다음 UI의 왼쪽 상단에서 이름을 클릭하여 사용자 기본 설정 메뉴를 표시합니다. 메뉴에서 **Switch Role**으로 이동하여 ACCOUNTADMIN을 선택합니다.

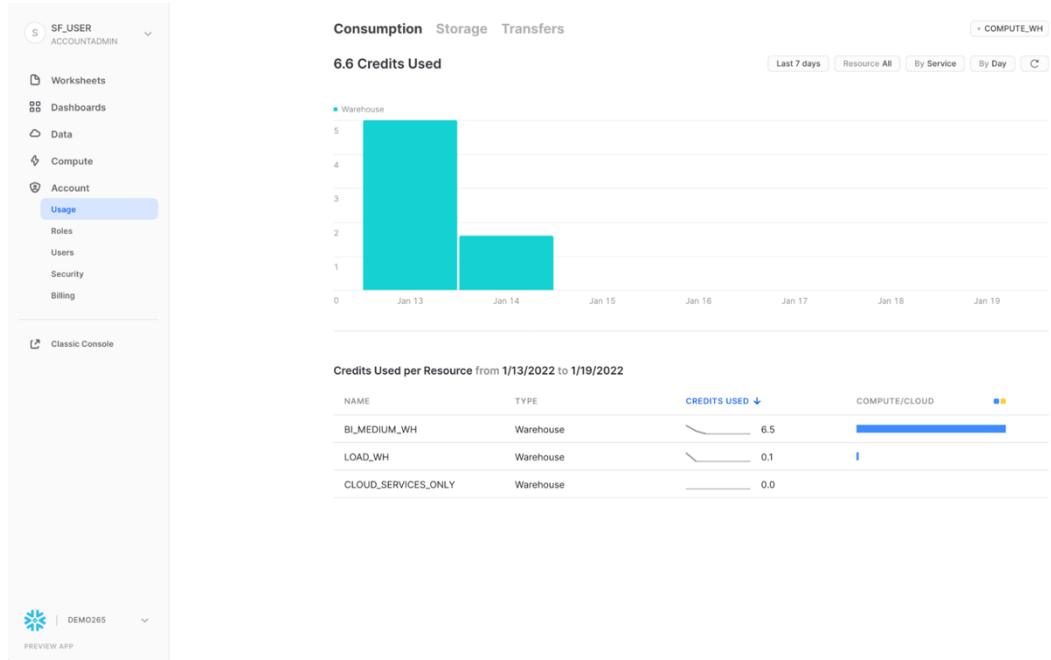


User Preference 설정의 역할 vs 워크시트

UI 세션과 각 워크시트에는 고유한 역할이 있습니다. UI 세션 역할은 UI에서 보고 액세스할 수 있는 요소를 제어하는 반면 워크시트 역할은 역할 내에서 액세스할 수 있는 개체 및 작업만 제어합니다.

UI 세션을 ACCOUNTADMIN 역할로 전환하면 **Account**에서 새 탭을 사용할 수 있습니다.

Usage



Usage 탭에는 각각 고유한 페이지가 있는 다음이 표시됩니다.

- **Organization:** 조직의 모든 계정에서 사용된 크레딧입니다.
- **Consumption:** 현재 계정의 가상 웨어하우스에서 소비한 크레딧입니다.
- **Storage:** 지난 달 현재 계정의 모든 데이터베이스, 내부 단계 및 Snowflake Failsafe에 저장된 평균 데이터 양입니다.
- **Transfers:** 지난 한 달 동안 해당 지역(현재 계정의 경우)에서 다른 지역으로 전송된 평균 데이터 양입니다.

각 페이지의 오른쪽 상단 모서리에 있는 필터를 사용하여 사용량/소비량 등을 분류 할 수 있습니다.

Security

The screenshot shows the Snowflake Security interface. On the left, a sidebar menu includes options like Worksheets, Dashboards, Data, Compute, Account, Usage, Roles, Users, and Security (which is selected and highlighted in blue). Below the sidebar is a "Classic Console" link. The main content area is titled "Network Policies" and "Sessions". At the top right, there is a blue button labeled "+ Network Policy". The "Network Policies" section displays a message: "No network policies. Restrict or allow access to your Snowflake account based on IP address. Learn More". The "Sessions" section shows a list of active sessions, with one entry for "DEMO265" from "PREVIEW APP".

Security 탭에는 Snowflake 계정에 대해 생성된 네트워크 정책이 포함되어 있습니다. 페이지 오른쪽 상단의 "+ 네트워크 정책"을 선택하여 새 네트워크 정책을 만들 수 있습니다.

Billing

The screenshot shows the Snowflake Billing interface. The left sidebar menu includes Worksheets, Dashboards, Data, Compute, Account, Usage, Roles, Users, Security, and Billing (which is selected and highlighted in blue). Below the sidebar is a "Classic Console" link. The main content area is titled "Billing" and contains a "Payment Method" section. It includes a sub-section for "Credit Card" and a note: "Add a credit card to continue using Snowflake once your free trial ends." At the top right, there is a blue button labeled "+ Credit Card". The bottom of the screen shows a preview of the "DEMO265" application.

Billing 탭에는 계정에 대한 결제 수단이 포함되어 있습니다.

- Snowflake 계약 고객인 경우 탭에 계약 정보와 연결된 이름이 표시됩니다.
- 주문형 Snowflake 고객인 경우 탭에 월별 결제에 사용된 신용 카드가 표시됩니다(입력된 경우). 등록된 신용 카드가 없는 경우 평가판이 종료될 때 Snowflake를 계속 사용하려면 신용 카드를 추가할 수 있습니다.

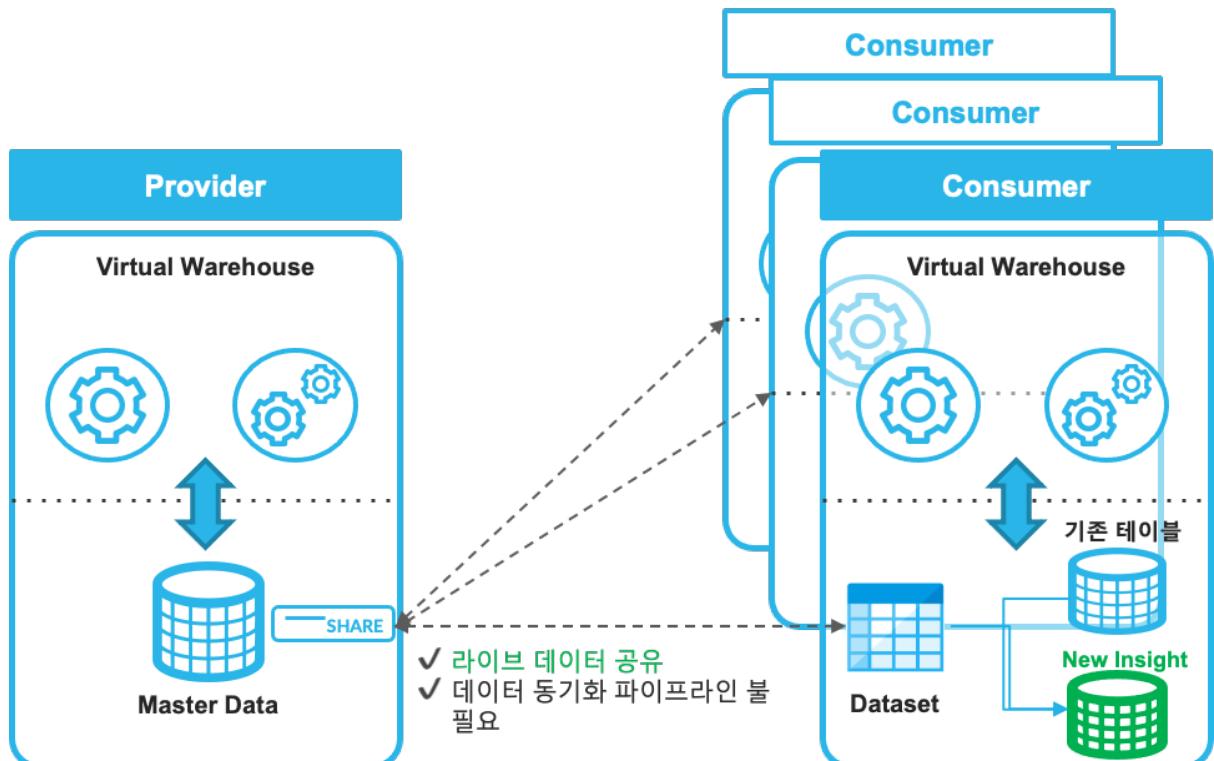
다음 섹션에서도 공유에 대한 중요한 권한 때문에 ACCOUNTADMIN 역할을 유지합니다.

6. Secure Data Sharing and Snowflake Marketplace

Snowflake는 공유를 통해 계정 간 데이터 액세스를 가능하게 합니다. 공유는 데이터 공급자가 생성하고 데이터 소비자가 자신의 Snowflake 계정 또는 프로비저닝된 Snowflake 읽기 전용 계정을 통해 가져옵니다.

다음을 통해 안전하게 데이터를 공유합니다.

- 데이터 사본은 하나뿐이며 데이터 제공자의 계정에 있음
- 공유 데이터는 항상 활성화되어 있고, 실시간이며 소비자가 즉시 사용할 수 있음
- 공급자가 공유에 대한 취소 가능하고 세분화된 액세스 권한을 설정할 수 있음
- 데이터 공유는 특히 인터넷을 통한 대용량 .csv 파일 전송을 포함하는 수동의 안전하지 않은 이전 데이터 공유 방법과 비교하여 간단하고 안전함



- Secure Data Sharing 소개: <https://docs.snowflake.com/ko/user-guide/data-sharing-intro.html>

기존 공유 보기

홈페이지에서 **Data > Databases**로 이동합니다. 데이터베이스 목록에서 **SOURCE** 열을 확인합니다. 열에 Local이 있는 두 개의 데이터베이스가 표시되어야 합니다. 이들은 이전에 실습에서 만든 두 개의 데이터베이스입니다. 다른 데이터베이스인 SNOWFLAKE는 열에 Share를 표시하여 공급자로부터 공유되었음을 나타냅니다.

The screenshot shows the Snowflake Data Catalog interface. On the left, there is a sidebar with navigation links: Worksheets, Dashboards, Data (with Databases selected), Databases (Shared Data, Marketplace, Compute, Account), and Classic Console. The main area displays the 'Databases' page with the following details:

NAME	SOURCE	OWNER	CREATED	Actions
CITIBIKE	Local	SYSADMIN	13 hours ago	...
SNOWFLAKE	Share	—	9 months ago	...
WEATHER	Local	SYSADMIN	53 minutes ago	...

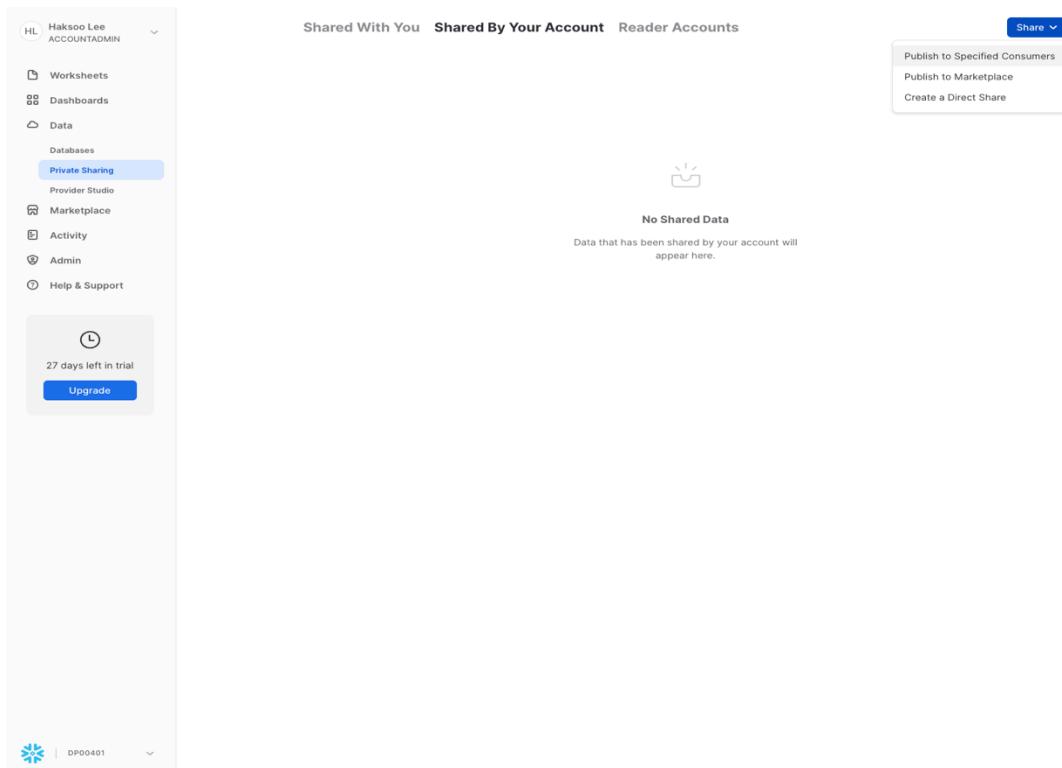
At the bottom left, there is a preview app section with the text 'DEMO265' and 'PREVIEW APP'.

아웃바운드 공유 생성

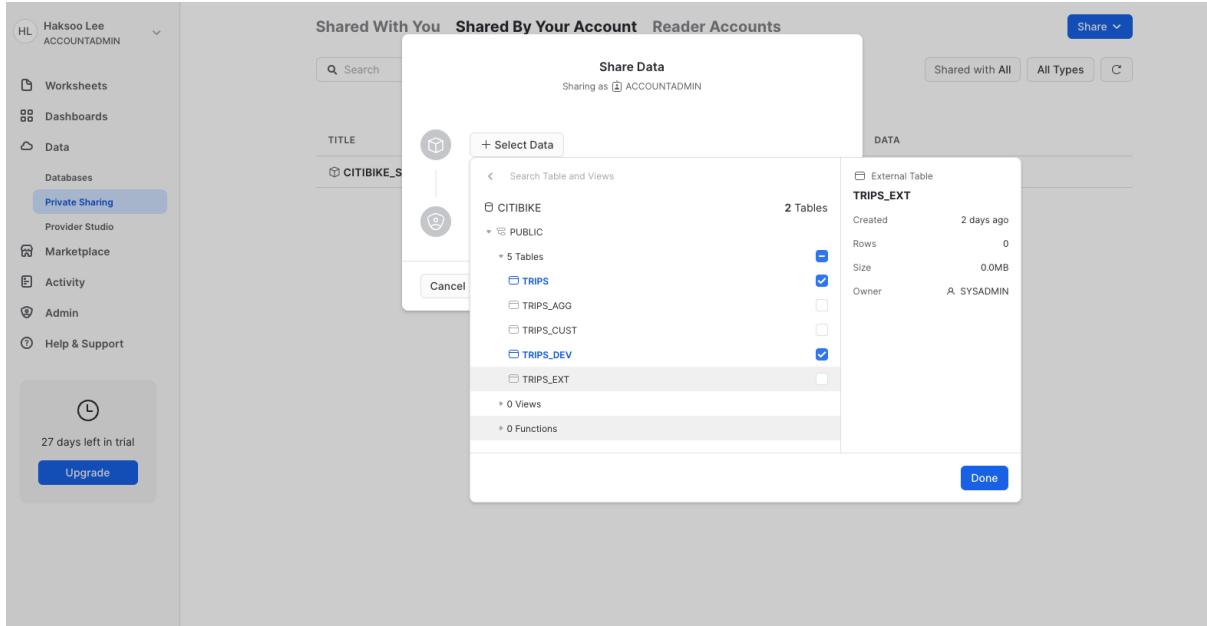
Snowflake 계정 관리자라고 가정하겠습니다. TRIPS 데이터베이스의 데이터를 거의 실시간으로 분석하길 원하는 신뢰할 수 있는 파트너가 있습니다. 이 파트너는 또한 우리 지역(Region)에 속한 고유한 Snowflake 계정도 갖고 있습니다.

따라서 Snowflake 데이터 공유를 이용하여 그들이 이 정보에 액세스할 수 있도록 해 보겠습니다.

Data > Private Sharing으로 이동한 다음 탭 상단의 **Shared by Your Account**를 클릭합니다. 오른쪽 상단의 **Share** 버튼을 클릭하고 **Create a Direct Share**를 선택합니다.

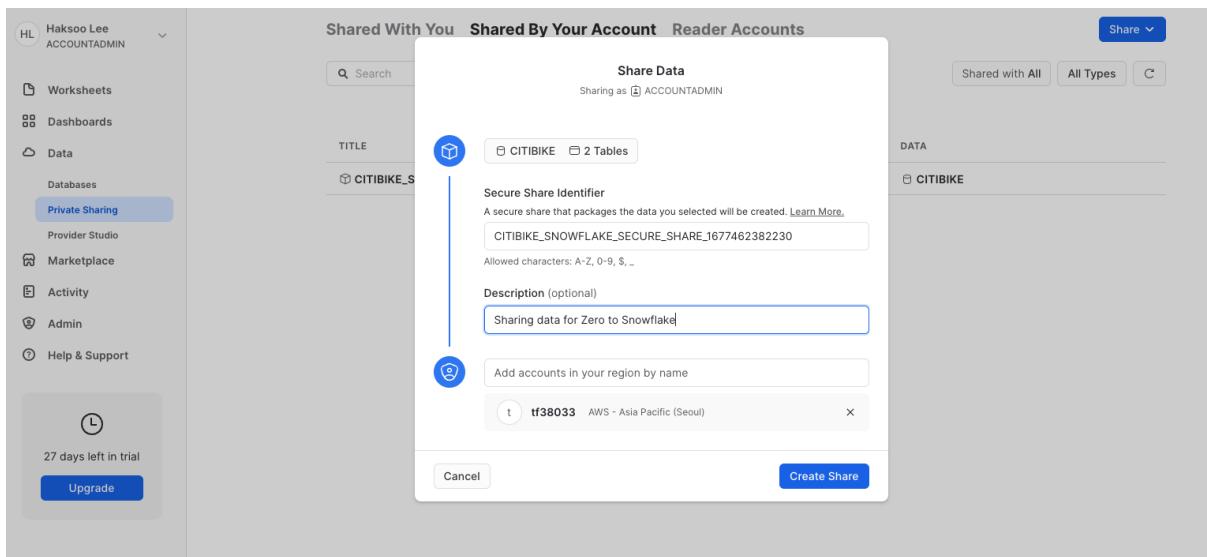


+ **Data**를 클릭하고 CITIBIKE 데이터베이스 및 PUBLIC 스키마로 이동합니다. 스키마에서 생성한 2개의 테이블을 선택하고 **Done** 버튼을 클릭합니다.



공유의 기본 이름은 임의의 숫자 값이 추가된 일반 이름입니다. 나중에 공유를 식별하는 데 도움이 되도록 기본 이름을 수정합니다.

대화 상자 하단의 **Create Share** 버튼을 클릭합니다.



대화 상자가 닫히고 페이지에 생성한 Secure Share가 표시됩니다.

The screenshot shows the 'Shared With' section of a secure share named 'CITIBIKE_SNOWFLAKE_SECURE_SHARE_1677462382230'. The page includes a header with the share name, a creation timestamp, and location. It features sections for 'Shared With', 'Description', and 'Data'. The 'Shared With' section lists one account: 'SFSEAPAC.KR_DEMO16' (Account type). The 'Description' section contains a placeholder for a summary. The 'Data' section shows two tables: 'TRIPS' and 'TRIPS_DEV', both under the 'PUBLIC' schema.

언제든지 소비자를 추가하고 설명을 추가/변경하고 공유의 개체를 편집할 수 있습니다. 페이지에서 공유 이름 옆에 있는 < 버튼을 클릭하여 **Share with Other Accounts** 페이지로 돌아갑니다.

The screenshot shows the 'Shared With You' page. It lists a single shared item: 'CITIBIKE_SNOWFLAKE_SECURE_SHARE_1677462382230' shared by 'SFSEAPAC.KR_DEMO16' 23 minutes ago. The page includes a search bar, filters for 'Shared with All' and 'All Types', and a 'Share' dropdown menu.

Snowflake는 기밀성을 손상시키지 않고 데이터를 안전하게 공유하는 여러 방법을 제공합니다. 테이블 외에도 Secure View, Secure UDF(사용자 정의 함수) 및 기타 보안 개체를 공유할 수 있습니다.

데이터 수신

The screenshot shows the 'Shared With You' section of the Snowflake interface. A modal window titled 'Get Data' is open, prompting the user to create a database to query the shared data. The database name is set to 'ZBGJAGW_DP00401_CITIBIKE_SNOWFLAKE_SECURE_SHARE_1677462382230'. The roles assigned are 'ANALYST_CITIBIKE, HOL_ROLE, PUBLIC'. To the right, there's a sidebar for 'Knoema' featuring the 'COVID-19 Data Atlas' with 30+ datasets from WHO, JHU, ECDC, and CDC.

Snowflake Marketplace

ACCOUNTADMIN 역할을 사용 중인지 확인하고 Data 아래에서 Marketplace 탭으로 이동합니다.

The screenshot shows the 'Marketplace' tab within the Snowflake interface. It features a search bar and filters for categories like Business Needs, Providers, and Requests. Key sections include 'New Marketplace Capabilities' (Total Consumer Insights, IP to Geolocation, SafeGraph Core Places - US..., Weather Data for the United States - West), 'Featured Providers' (Funnel, Dun & Bradstreet, Equifax, ADP, Inc.), and 'Most Recent' products (SEC Reporting Analytics, CARTO Analytics Toolbox, Monte Carlo Data Observability Insights, Rystad Energy REnewableCube). A preview app icon is visible at the bottom left.

Listing 찾기

상단의 검색 상자를 사용하여 목록을 검색할 수 있습니다. 검색 상자 오른쪽에 있는 드롭다운 목록을 사용하면 공급자, 비즈니스 요구 사항 및 범주별로 데이터 목록을 필터링할 수 있습니다.

검색창에 COVID를 입력하고 결과를 스크롤한 후 **COVID-19 Epidemiological Data**(Starschema 제공)를 선택합니다.

The screenshot shows a search interface with a search bar containing "COVID". Below the search bar, there are several navigation links: "Browse Providers", "Business Needs", "Financial", "Weather", "Commerce", "Health and Life Sciences", "Demographics", and "More Categories". The main area displays "44 results for COVID" in bold. The results are presented in a grid of cards:

- Demand Data**: UK Covid Cases. Description: COVID Cases and population in the UK by Local Authority including Shape Files.
- State of California**: California COVID-19 Datasets. Description: COVID-19 confirmed counts and county information.
- Knoema**: COVID-19 Data Atlas. Description: 30+ public COVID-19 pandemic-related datasets from the WHO, JHU, ECDC, CDC, and other authoritative...
- Accern**: AI Powered COVID-19 Insights & Analytics. Description: Discover trends and insights where COVID-19 intersects with company issues.
- Traject**: COVID-19 Dataset. Description: A collection of SERP data related to COVID-19 search queries across a variety of US states and metro areas.
- ThinkData Works**: Covid-19 Canadian Outbreak. Description: A daily refreshed feed of Covid-19 case counts by Province and Regional Health Boundaries.
- DemystData**: Zip Code level COVID-19. Description: US COVID-19 cases and death counts by county and ZIP Code.
- Starschema**: COVID-19 Epidemiological Data. Description: Analytics-ready data on COVID-19: cases, vaccinations, healthcare resource availability and...
- Wunderman Thompson Data**: AmeriLINK Insights - Premium. Description: Attitudinal, Buying Behavior, Demographic, Health Related, Lifestyle and Transactional Data.
- Wunderman Thompson Data**: AmeriLINK Insights. Description: Marketing information on U.S. consumers.
- Lifesight**: Mobility Data - Custom. Description: High fidelity SDK-based mobility data collected from location-aware apps.
- Prevedere**: COVID19 - US Economy Leading Indicators. Description: Leading signals for the economic recovery following this economic downturn.

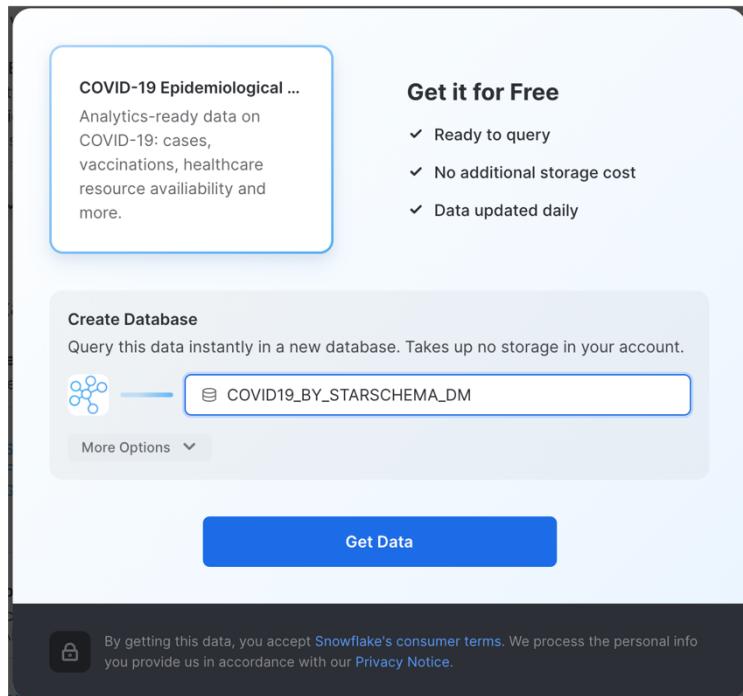
COVID-19 Epidemiological Data 페이지에서 데이터세트에 대해 자세히 알아보고 사용 예시 쿼리를 볼 수 있습니다. 준비가 되면 **Get Data** 버튼을 클릭하여 Snowflake 계정에서 이 정보를 사용할 수 있도록 합니다.

The screenshot shows the 'COVID-19 Epidemiological Data' page from the Snowflake Data Marketplace. At the top right, there is a 'Free' offer box with 'Unlimited queries' and a prominent blue 'Get Data' button, which is highlighted by a large red arrow. To the left of the offer box, there is a section titled 'COVID-19 Epidemiological Data' by Starschema, Health and Life Sciences, Daily. It includes a brief description of the dataset, example use cases, and a 'Show More' link. Below this, there is a 'Usage Examples' section with a code snippet for getting total case counts by country:

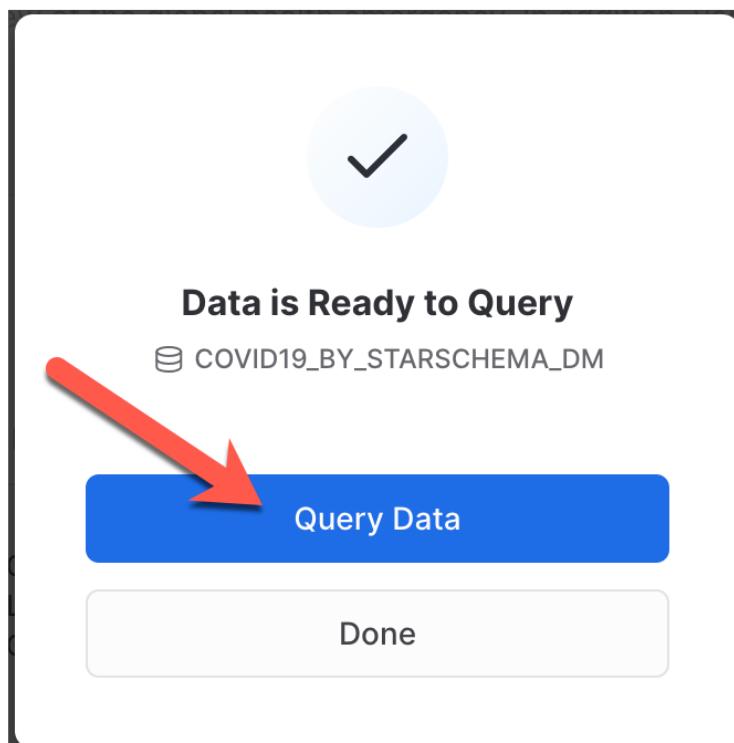
```
1  SELECT      COUNTRY_REGION, SUM(CASES) AS Cases
2  FROM        ECDC_GLOBAL
3  GROUP BY    COUNTRY_REGION;
```

At the bottom right of the page, there is a 'Starschema' sidebar with links to 'Contact', 'Documentation', and 'Terms of Service'.

대화 상자의 정보를 검토하고 **Get Data**를 다시 클릭합니다.



이제 **Done**를 클릭하거나 Starschema에서 제공하는 샘플 쿼리를 실행하도록 선택할 수 있습니다.



Query Data를 선택한 경우 새 브라우저 탭/창에서 새 워크사이트가 열립니다.

1. 실행할 쿼리를 선택하거나 쿼리 텍스트에 커서를 놓습니다.
2. **Play/Run** 버튼을 클릭합니다(또는 키보드 단축키 사용).
3. 하단 창에서 데이터 결과를 볼 수 있습니다.
4. 샘플 쿼리 실행이 완료되면 왼쪽 상단의 **Home** 아이콘을 클릭합니다.

The screenshot shows the Snowflake web interface. On the left, there's a sidebar with pinned objects and a list of databases. A red arrow labeled '4' points to the 'Home' icon at the top of the sidebar. In the center, a query editor window is open with a timestamp '2021-10-19 5:36pm'. The query itself is highlighted with a red box and contains a line number '1'. A red circle labeled '1' is placed over the first line of the query. To the right of the query editor is a results table showing data from the COVID19_BY_STARSCHHEMA_DM.PUBLIC schema. A red circle labeled '3' is placed over the fourth row of the results table. On the far right, there's a 'Query Details' panel with metrics like duration and a histogram. A red circle labeled '2' is placed over the 'Share' button in the top right corner of the interface.

```

1 // Get total case count by country
2 // Calculates the total number of cases by country, aggregated over time.
3 SELECT COUNTRY_REGION, SUM(CASES) AS Cases
4 FROM ECDC_GLOBAL
5 GROUP BY COUNTRY_REGION;
6
7 // Change in mobility in over time
8 // Displays the change in visits to places like grocery stores and parks by
9 // date, location and location type for a sub-region (Alexandria) of a state
10 // (Virginia) of a country (United States).
11
12
13
14
15
16
17
18
19
20
21
22

```

COUNTRY_REGION	CASES
Afghanistan	49,273
Albania	48,530
Algeria	92,102
Andorra	7,338
Angola	16,188
Anguilla	10
Antigua and Barbuda	148
Argentina	1,498,160
Armenia	148,682
Aruba	5,049

다음:

1. **Data > Databases**를 클릭합니다.
2. 'COVID19_BY_STARSCHHEMA_DM' 데이터베이스를 클릭합니다.
3. 쿼리에 사용할 수 있는 스키마, 테이블 및 뷰에 대한 세부 정보를 볼 수 있습니다.

The screenshot shows the Snowflake Data Marketplace interface. On the left, the user's profile (JW jason ACCOUNTADMIN) and navigation menu are visible. The 'Data' section is selected, with 'Databases' highlighted (marked with a red circle containing '1'). In the main pane, a database named 'COVID19_BY_STARSCHEMA_DM' is listed. This database was created by 'ACCOUNTADMIN' 15 hours ago and is available in the 'Snowflake Data Marketplace'. The 'Database Details' tab is selected, showing the 'Source details' and 'Privileges' sections. The 'Source details' section includes information about the provider ('Starschema') and the title ('COVID-19 Epidemiological Data'). The 'Privileges' section shows that 'ACCOUNTADMIN' has ownership of the database. A red circle containing '2' is positioned over the database listing, and another red circle containing '3' is positioned over the 'Source details' section.

이제 글로벌 COVID 데이터로 매일 업데이트되는 Starschema의 COVID-19 데이터 세트를 성공적으로 구독했습니다. 데이터베이스, 테이블, 뷰 또는 ETL 프로세스를 생성할 필요 없이 라이브 업데이트되는 데이터 세트를 분석에 활용할 수 있습니다.