

Content

Introduction	2
Create a Snowflake Account	3 – 7
Create Database	8
Create Schema	9
Create TABLE and COLUMNS	9 – 11
Types of Snowflake Tables	11 – 12
SHOW TABLES	13 – 15
SnowSQL Installation	15 – 17
Load CSV data from the local to Snowflake	18 – 21
SnowSight	22 – 23
Micro-partitions	24 – 25
Data Clustering	26 – 29
Sequence	29 – 32
Monitor Usage and Storage	33 – 34

Introduction

Snowflake is a cloud-based advanced data platform system, provided as Software-as-a-Service (SaaS). Snowflake provides features of data storage from AWS S3, Azure, and Google Cloud, processing complex queries and different analytic solutions. The analytic solutions provided by Snowflake are faster, easy to use, and more flexible than traditional databases and their analytics features. Snowflake stores and provides data near time not in actual real time.

Snowflake is an advanced solution for OLAP (Online Analytical Processing) technology. OLAP is also known as an online data retrieving and data analysis system using historical data. It processes complex and aggregated queries with a low number of transactions. For Ex: Getting the number of orders, sales amount in the last month for a company, number of new users listed in the company in the last quarter, etc. Snowflake is not used as an OLTP (Online Transactional Processing) database. OLTP databases usually contain real-time data with a high volume of small data transactions. For Ex: Inserting customer's order details, registering a new customer, tracking order delivery status, etc.

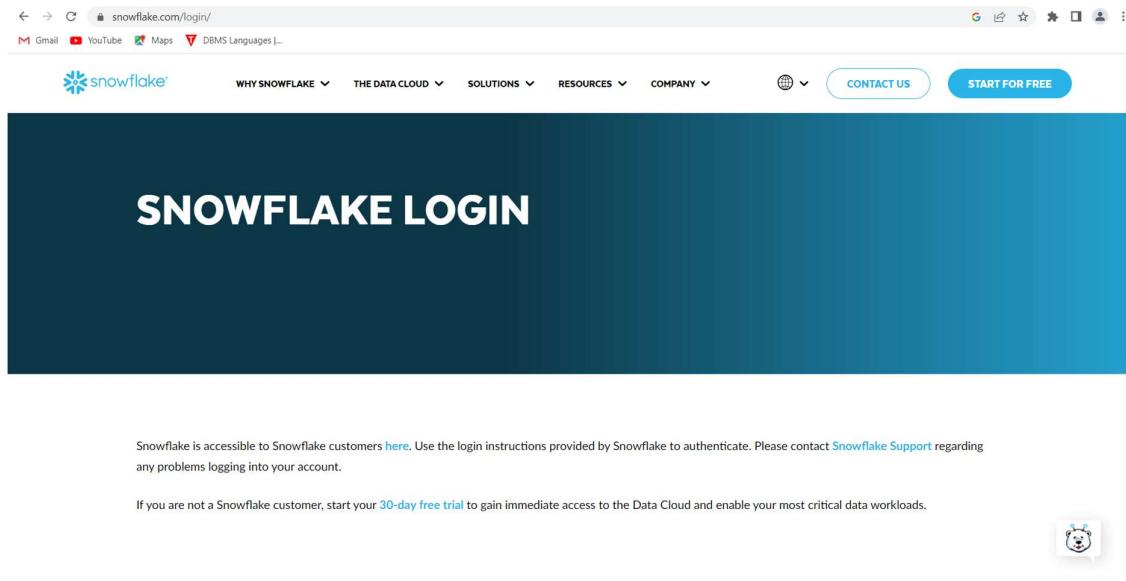
Why Use Snowflake?

Snowflake provides a Data Platform as a Cloud Service.

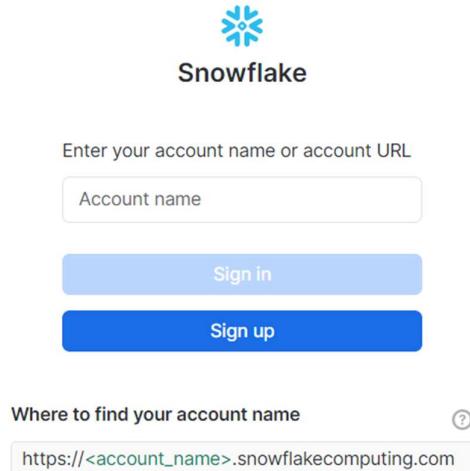
- There is no hardware neither virtual nor physical to select, install, configure, or manage from the client side.
- There is no software to install, configure, or manage to access it.
- All ongoing maintenance, management, upgrades, and patching are owned by Snowflake itself.

Create a Snowflake Account

1. To create a Snowflake account please click on this link
<https://www.snowflake.com/login/>
2. Next you will see a window like below, click on the Snowflake customer here option as shown below.



3. Next click on the signup option as shown below.

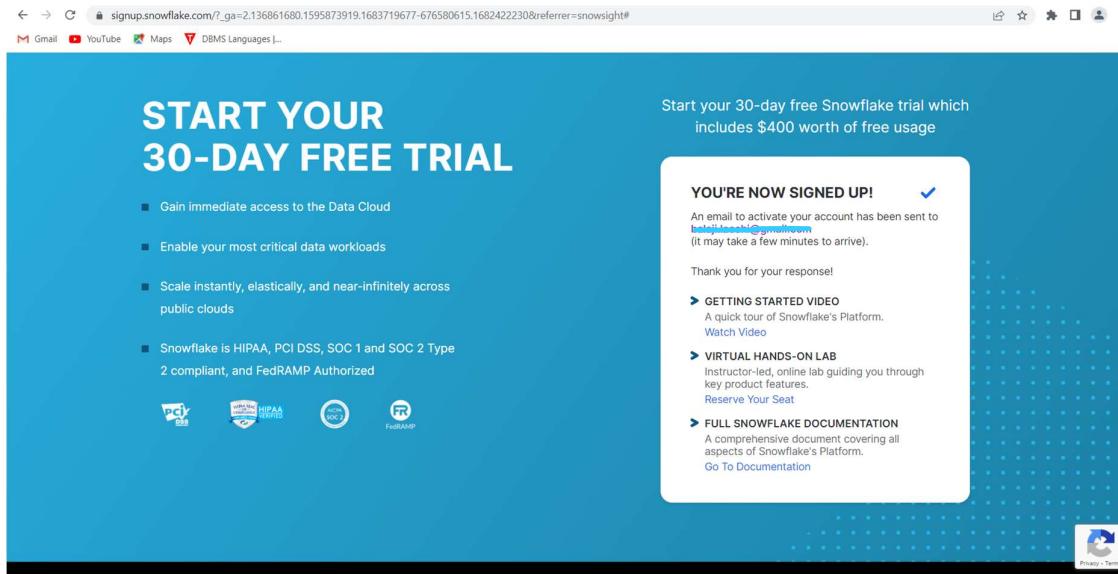


4. Next fill up the following details and click on continue as shown below.

5. Next check the enterprise edition click on Microsoft Azure give central India(Pune) and click on Get Started as shown below.

6. Skip the conditions as shown below.

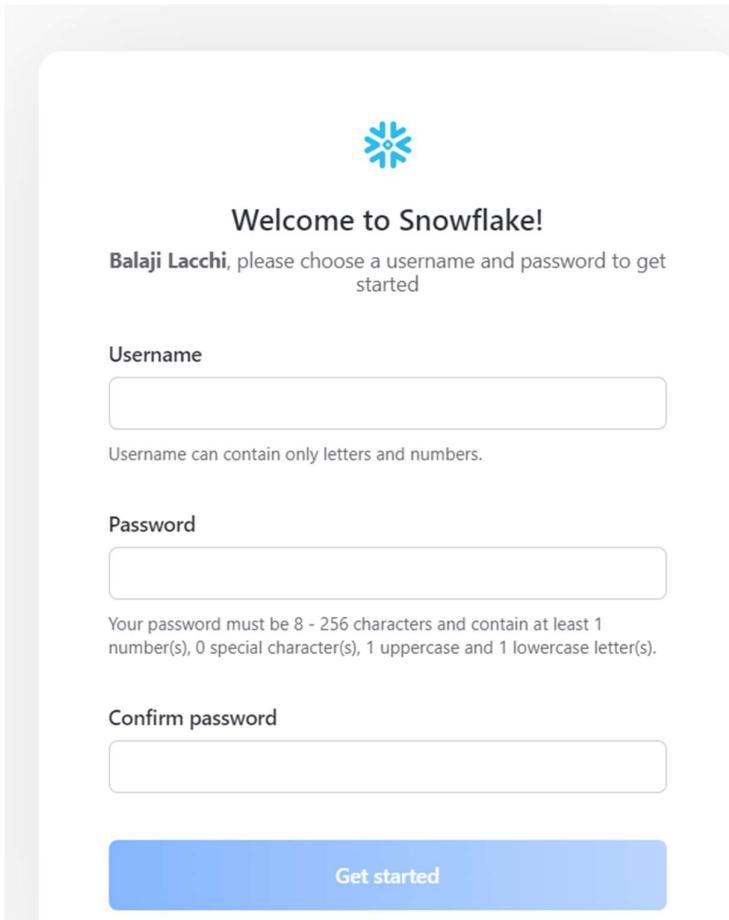
7. Next you will see a message like this with your mail ID as shown below.



8. Next you will receive one mail from Snowflake below click on the Click to Activate option as shown below.

A screenshot of an email inbox showing an activation email from "Snowflake Computing <no-reply@snowflake.net>". The subject line is "Activate your Snowflake account". The email body contains the Snowflake logo and a message: "Congratulations on getting started with Snowflake! Click the button below to activate your account." Below this is a blue "CLICK TO ACTIVATE" button. A note states: "This activation link is temporary and will expire in 72 hours." There is also a "Save this for later" link with the URL "https://vxnvxz-db97690.snowflakecomputing.com/console/login". The email was sent at 4:45PM (0 minutes ago).

9. Now give the username and password then confirm the password as shown below.
10. Next click on Get Started.

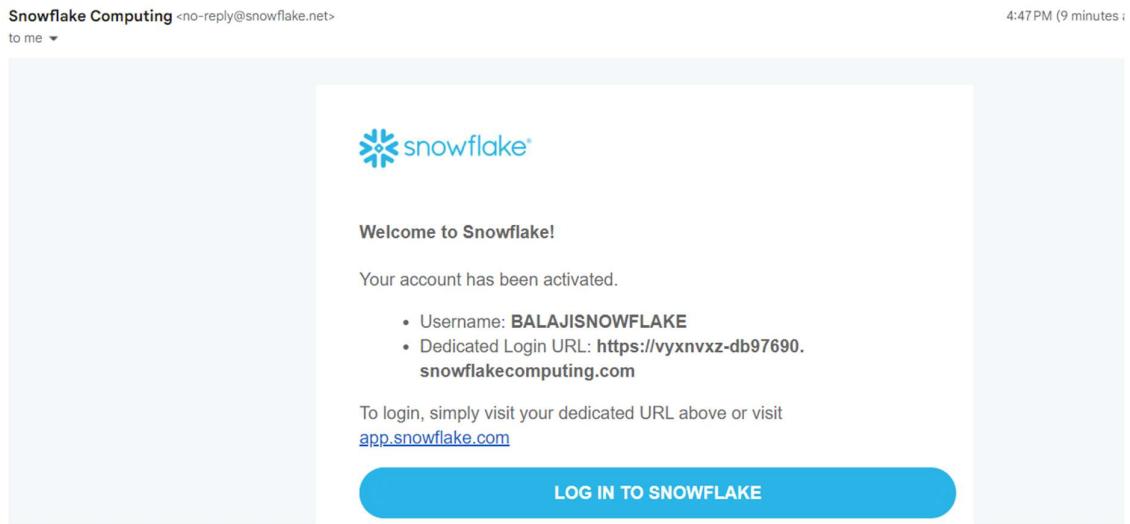


11. It will take you to the snowflake home window as shown below.

The screenshot shows the Snowflake home page with the "Worksheets" section selected. On the left is a sidebar with navigation links: "+ Create", "Search", "Projects", "Worksheets" (which is highlighted in blue), "Notebooks", "Streamlit", "Dashboards", "App Packages", and "Data". The main area is titled "Worksheets" and displays a list of recent worksheets. The table has columns for TITLE, TYPE, VIEWED (with a downward arrow), and UPDATED. The data in the table is as follows:

TITLE	TYPE	VIEWED ↓	UPDATED
Load sample data with SQL from S3 bu...	SQL	—	2 hours ago
Load sample data with Python from S3 b...	Python	—	2 hours ago
[Template] Adding a user and granting r...	SQL	—	2 hours ago
Getting Started Tutorials	Folder	—	2 hours ago
Sample queries on TPC-H data	SQL	—	2 hours ago
Sample queries on TPC-DS data	SQL	—	2 hours ago
TPC-DS 10TB Complete Query Test	SQL	—	2 hours ago

12. In case you don't see the above window. You will receive another mail from Snowflake like below.
13. Just click on login then give Username and password.



Create Database

1. Write the following query to create a database "SAMPLE_SNOW"

Query: Create database SAMPLE_SNOW

```
1 Create DATABASE "SAMPLE_SNOW"
```

The screenshot shows a database workspace interface. At the top, there are two buttons: 'Results' (highlighted in blue) and 'Chart'. Below this is a table with one row. The table has two columns: 'status' and '1'. The 'status' column contains the text 'Database SAMPLE_SNOW successfully created.'

	status
1	Database SAMPLE_SNOW successfully created.

2. As you can see on the left side Our database has been created.

The screenshot shows the left sidebar of a database application. The top navigation bar has tabs for 'Databases' (which is selected and highlighted in blue) and 'Worksheets'. Below this, there's a section titled 'Pinned (0)' followed by 'No pinned objects'. A search bar labeled 'All Objects' is present. The main list contains several databases: IICS, SAMPLE_SNOW (which is selected and highlighted with a blue square icon), SNOWFLAKE, SNOWFLAKE_SAMPLE_DATA, and TEST.

3. Now on the top of the workspace select the Database as shown below.

The screenshot shows the top navigation bar of a workspace. It includes a dropdown menu labeled '"SAMPLE_SNOW"."Snowflake"' with a downward arrow, a 'Settings' button with a dropdown arrow, and a search bar labeled 'Databases'. Below the search bar, a list shows three databases: SAMPLE_SNOW (which is selected and highlighted with a blue checkmark icon), IICS, and Sample Snow.

Create Schema

1. Write the following query to create a schema Snowflake under the database SAMPLE_SNOW.

Query: CREATE SCHEMA "SAMPLE_SNOW"."Snowflake"

```
"SAMPLE_SNOW"."Snowflake" ▾  
1 | Create schema "SAMPLE_SNOW"."Snowflake"
```

Results	
	status
1	Schema Snowflake successfully created.

2. As you can see under the SAMPLE_SNOW database Snowflake schema has been created.

Databases Worksheets
Pinned (0)
No pinned objects
All Objects ...
> IICS
< SAMPLE_SNOW
 > INFORMATION_SCHEMA
 > PUBLIC
 > Snowflake
 > SNOWFLAKE
 > SNOWFLAKE_SAMPLE_DATA

Create TABLE and COLUMNS

1. Use the following query to create a table and columns under the database SAMPLE_SNOW and schema Snowflake.

Query: create table Employee_Details(id int,
Name varchar(30),
age int,
Salary decimal,
Gender varchar(30),
contact_number int,
Email_id varchar(30)
)

```

"SAMPLE_SNOW"."Snowflake" ▾

1  create table Employee_Details(
2    id int,
3    Name varchar(30),
4    age int,
5    Salary decimal,
6    Gender varchar(30),
7    contact_number int,
8    Email_id varchar(30)
9  )

```

The screenshot shows a database interface with a 'Results' tab selected. The output is a single row of text: 'Table EMPLOYEE_DETAILS successfully created.'

	status
1	Table EMPLOYEE_DETAILS successfully created.

- Now Insert some sample data into our table using the Insert command as shown below.

Query: insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values (1,'sagar',24,120000,'male',89942345,'sagar@gmail.com');

insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values (2,'ratan',22,130000,'male',80052345,'ratan@gmail.com');

insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values (3,'sakshi',23,125000,'female',72983456,'sakshi@gmail.com');

insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values (4,'sanskriti',24,145000,'female',78453478,'sanskriti@gmail.com');

insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values (5,'roshi',25,155000,'female',93287456,'roshi@gmail.com');

insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values (6,'rebanta',26,160000,'male',83465873,'rebanta@gmail.com');

- Copy the query and paste it in the workspace then click on Run.

```

"SAMPLE_SNOW"."Snowflake" ▾

1  insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values
2    (1,'sagar',24,120000,'male',89942345,'sagar@gmail.com');
3  insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values
4    (2,'ratan',22,130000,'male',80052345,'ratan@gmail.com');
5  insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values
6    (3,'sakshi',23,125000,'female',72983456,'sakshi@gmail.com');
7  insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values
8    (4,'sanskriti',24,145000,'female',78453478,'sanskriti@gmail.com');
9  insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values
10   (5,'roshi',25,155000,'female',93287456,'roshi@gmail.com');
11  insert into Employee_Details(id, Name, age, Salary, Gender, contact_number, Email_id) values
12   (6,'rebanta',26,160000,'male',83465873,'rebanta@gmail.com');

```

The screenshot shows a database interface with a 'Results' tab selected. The output is a single row of text: '1' under the 'number of rows inserted' column.

	number of rows inserted
...	1

4. Use to check the table use SELECT command.

Query: Select * from Employee_Details

The screenshot shows a Snowflake query editor interface. At the top, there is a dropdown menu labeled "SAMPLE_SNOW"."Snowflake". Below it, a code editor window contains the SQL command: "Select * from Employee_Details". The results tab is selected, showing a table titled "Employee_Details" with six rows of data. The columns are ID, NAME, AGE, SALARY, GENDER, CONTACT_NUMBER, EMAIL_ID, and ... (ellipsis). The data is as follows:

ID	NAME	AGE	SALARY	GENDER	CONTACT_NUMBER	EMAIL_ID	...
1	sagar	24	120,000	male	89,942,345	sagar@gmail.com	
2	ratan	22	130,000	male	80,052,345	ratan@gmail.com	
3	sakshi	23	125,000	female	72,983,456	sakshi@gmail.com	
4	sanskriti	24	145,000	female	78,453,478	sanskriti@gmail.com	
5	roshi	25	155,000	female	93,287,456	roshi@gmail.com	
6	rebanta	26	160,000	male	83,465,873	rebanta@gmail.com	

Types of Snowflake Tables

Tables are database objects logically structured as a collection of rows and columns. All data in Snowflake is stored in database tables. Apart from standard database tables, Snowflake supports other table types that are especially useful for storing data that does not need to be maintained for extended periods of time.

Snowflake supports three types of tables

- Permanent table
- Transient table
- Temporary table

Permanent Table

These are the standard, regular database tables. Permanent tables are the default table type in Snowflake and do not need any additional syntax while creating to make them permanent.

1. The data stored in permanent tables consumes space and contributes to the storage charges that Snowflake bills your account.
2. It also comes with additional features like Time-Travel and Fail-Safe which helps in data availability and recovery.
3. To create a Permanent table in Snowflake.

Query: create table employee (id number, name varchar(50));

The screenshot shows a Snowflake query editor interface. At the top, there is a dropdown menu labeled "SAMPLE_SNOW"."Snowflake" and a settings icon. Below it, a code editor window contains the SQL command: "create table employee (id number, name varchar(50));". The results tab is selected, showing a table titled "status" with one row of data. The data is as follows:

status
Table EMPLOYEE successfully created.

Transient Table

Transient tables in Snowflake are similar to permanent tables except that they do not have a Fail-safe period and only have a very limited Time-Travel period. These are best suited in scenarios where the data in your table is not critical and can be recovered from external means if required.

1. To create a Transient table in Snowflake, You need to mention transient in the create table syntax.

Query: `create transient table employee_transient (id number, name varchar(50));`

The screenshot shows the Snowflake UI interface. At the top, there is a navigation bar with the database name "SAMPLE_SNOW"."Snowflake" and a "Settings" dropdown. Below the navigation bar, a code editor window contains the SQL command: `create transient table employee_transient (id number, name varchar(50));`. The code is highlighted in blue. At the bottom of the code editor, there is a "Results" tab which is selected, and a "status" section below it. The status section displays the message: "Table EMPLOYEE_TRANSIENT successfully created."

Temporary Table

Temporary Tables in Snowflake exist only within the session in which they were created and are available only for the remainder of the session. They are not visible to other users or sessions. Once the session ends, data stored in the table is dropped completely and is not recoverable.

2. To create a Temporary table in Snowflake, you need to mention temporary in the create table syntax.

Query: `create temporary table employee (id number, name varchar(50));`

The screenshot shows the Snowflake UI interface. At the top, there is a navigation bar with the database name "SAMPLE_SNOW"."Snowflake" and a "Settings" dropdown. Below the navigation bar, a code editor window contains the SQL command: `create temporary table employee (id number, name varchar(50));`. The code is highlighted in blue. At the bottom of the code editor, there is a "Results" tab which is selected, and a "status" section below it. The status section displays the message: "Table EMPLOYEE_TEMPORARY successfully created."

SHOW TABLES

1. Lists the tables for which you have access privileges, including dropped tables that are still within the Time Travel retention period and, therefore, can be undropped.
2. Syntax to execute show tables in Snowflake is as below.

Query: SHOW TABLES;

The screenshot shows the Snowflake UI interface. At the top, there is a dropdown menu set to "'SAMPLE_SNOW'.'Snowflake'" and a 'Settings' button. Below this, a code editor window contains the query '1 SHOW TABLES;'. The results are displayed in a table titled 'Results'. The table has columns: created_on, name, database_name, schema_name, ..., and kind. There are three rows of data:

	created_on	name	database_name	schema_name	...	kind
1	2024-06-13 23:52:57.503 -0700	EMPLOYEE	SAMPLE_SNOW	Snowflake		TEMPORARY
2	2024-06-13 23:51:07.508 -0700	EMPLOYEE_DETAILS	SAMPLE_SNOW	Snowflake		TABLE
3	2024-06-13 23:52:46.639 -0700	EMPLOYEE_TRANSIENT	SAMPLE_SNOW	Snowflake		TRANSIENT

On the right side of the interface, there are three vertical tabs: 'Query D', 'Query di', and 'Rows', with 'Rows' being the active tab.

3. To get table details from a particular database and Schema.

Query: SHOW TABLES IN SAMPLE_SNOW."Snowflake";

The screenshot shows the Snowflake UI interface. At the top, there is a dropdown menu set to "'SAMPLE_SNOW'.'Snowflake'" and a 'Settings' button. Below this, a code editor window contains the query '1 SHOW TABLES IN SAMPLE_SNOW."Snowflake"';'. The results are displayed in a table titled 'Results'. The table has columns: created_on, name, database_name, schema_name, ..., and kind. There are three rows of data:

	created_on	name	database_name	schema_name	...	kind
1	2024-06-13 23:52:57.503 -0700	EMPLOYEE	SAMPLE_SNOW	Snowflake		TEMPORARY
2	2024-06-13 23:51:07.508 -0700	EMPLOYEE_DETAILS	SAMPLE_SNOW	Snowflake		TABLE
3	2024-06-13 23:52:46.639 -0700	EMPLOYEE_TRANSIENT	SAMPLE_SNOW	Snowflake		TRANSIENT

On the right side of the interface, there are three vertical tabs: 'Query D', 'Query di', and 'Rows', with 'Rows' being the active tab.

GET_DDL

4. Returns a DDL statement that can be used to recreate the specified object. The DDL statement contains the type of the Snowflake table.
5. Syntax to get DDL of a table.

Query: select get_ddl('table','employee');

The screenshot shows a Snowflake query editor interface. At the top, it says "SAMPLE_SNOW"."Snowflake" and "Settings". Below the editor area, there's a code snippet:

```
1 | select get_ddl('table','employee');
```

Underneath the code, there are two tabs: "Results" (which is selected) and "Chart". The results table has one row:

GET_DDL('TABLE','EMPLOYEE')	
1	create or replace TABLE EMPLOYEE (ID NUMBER(38,0), NAME VARCHAR(50));

Create a transient database

1. To create a Transient database use the below query.

Query: CREATE TRANSIENT DATABASE mytransientdb;

The screenshot shows a Snowflake query editor interface. At the top, it says "No Database selected" and "Settings". Below the editor area, there's a code snippet:

```
| CREATE TRANSIENT DATABASE mytransientdb;
```

Underneath the code, there are two tabs: "Results" (which is selected) and "Chart". The results table has one row:

status	
Database MYTRANSIENTDB successfully created.	

2. Show the database use below query.

Query: SHOW DATABASES LIKE 'my%';

```
MYTRANSIENTDB.PUBLIC ▾ Settings ▾
1 | SHOW DATABASES LIKE 'my%';

Results ▾ Chart | Query Details | Query duration
```

	created_on	name	is_default	is_current	origin	owner
1	2024-06-13 23:54:38.986 -0700	MYTRANSIENTDB	N	Y		ACCOUNTADMIN

SnowSQL Installation

1. Click on the link to download the SnowSQL file <https://developers.snowflake.com/snowsql/>.
2. Scroll down a bit and choose the windows then click on latest version File Name to download.

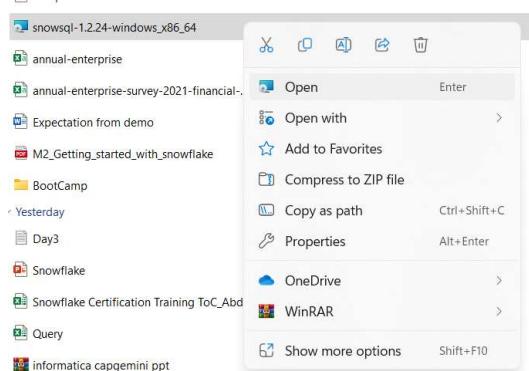
DEVELOPERS DOCS COMMUNITY MEDIUM BLOG YOUTUBE CHANNEL OPEN SOURCE DOWNLOADS ▾ START FOR FR

Download All Versions

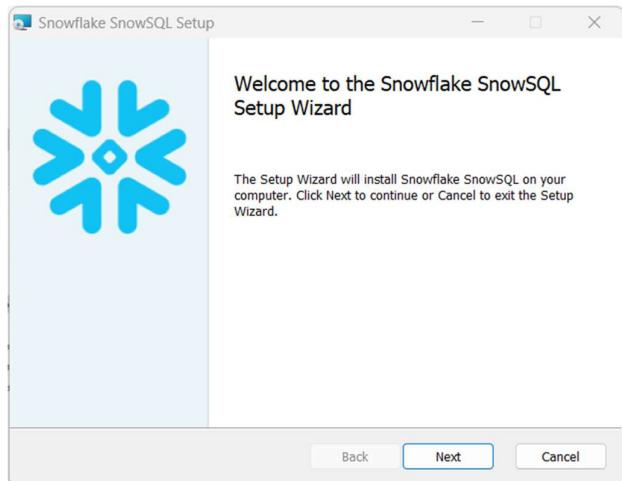
LINUX MACOS WINDOWS

Version	File Name	Architecture	Size	Release Date	SHA256 Checksum
1.2.24	snowsql-1.2.24-windows_x86_64.msi	windows_x86_64	34607 kB	2022-10-20T20:57:27	81d88b0b8780b45e6a13496447fde f204be359c8278b5a147459ca422c5 3d1e2
1.2.23	snowsql-1.2.23-windows_x86_64.msi	windows_x86_64	34570 kB	2022-07-28T22:07:47	307ba45ff251e3d51a3c20f8adc672 e69c999f9960c4b402e7e005a9f669 11bb
1.2.22	snowsql-1.2.22-	windows x86 64	34512	2022-06-	7abc1cf8f8dde0c45fb7e4210cc03d 28a60aee2e7164ef7425d024563a6

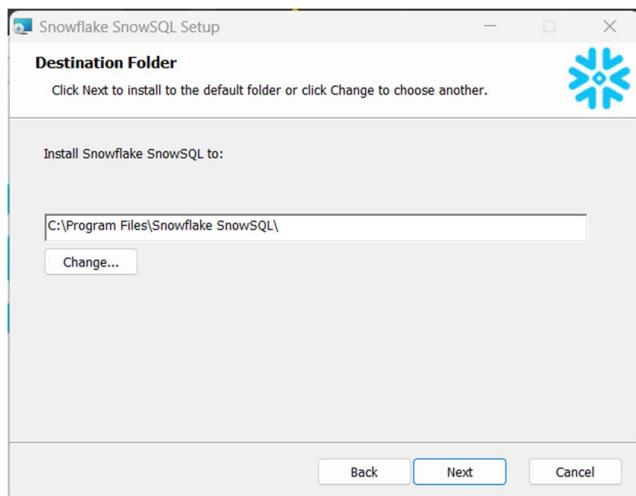
3. After downloading complete open the file as shown below.



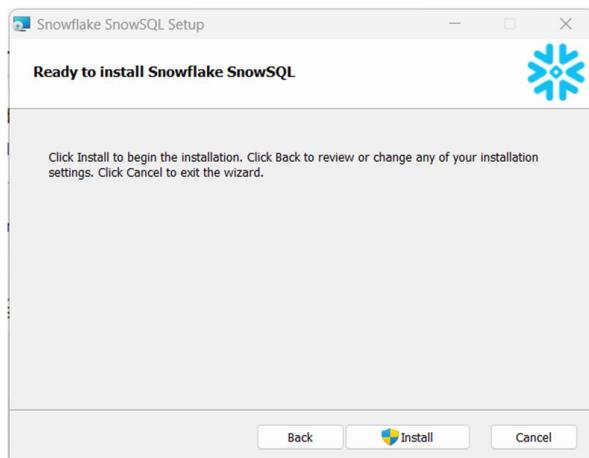
4. Click on Next.



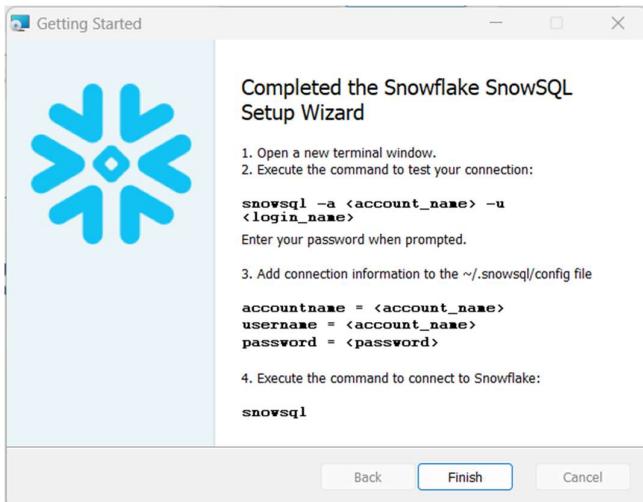
5. Now choose the location then click on next.



6. Click on Install.

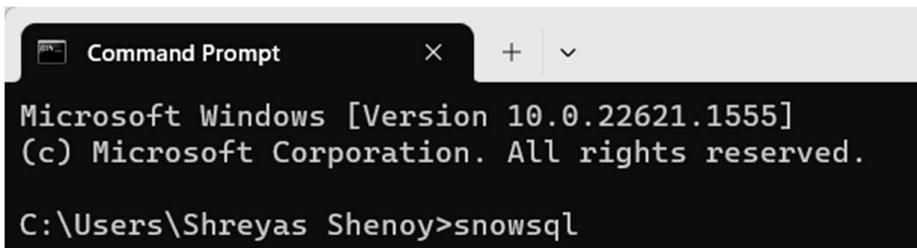


7. After some time you will get this.

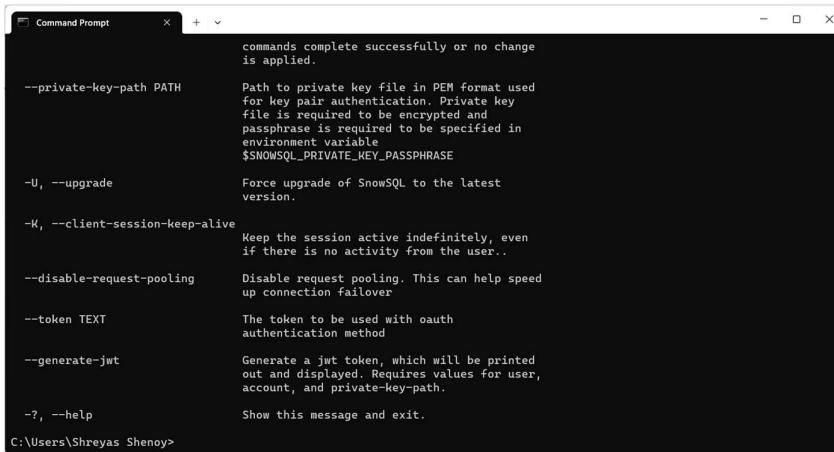


8. Open the command prompt and give the following command as shown below.

Command: `snowsql`



9. Click Enter then you will see a message as shown below.



Load CSV data from the local to Snowflake

For this **SnowSQL** should be installed in your system.

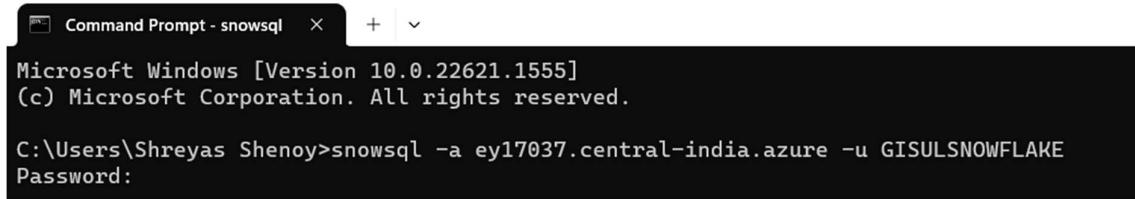
In this example we are going load this CSV data into snowflake.

1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
2	Year	Industry_a	Industry_c	Industry_n	Units	Variable_c	Variable_n	Variable_c	Value		Industry_code_ANZSIC06									
3	2021	Level1		99999	All Industri	Dollars	(m)	H01	Total incor	Financial p	7,57,504	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
4	2021	Level1		99999	All Industri	Dollars	(m)	H04	Sales, gove	Financial p	6,74,890	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
5	2021	Level1		99999	All Industri	Dollars	(m)	H05	Interest, di	Financial p	49,593	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
6	2021	Level1		99999	All Industri	Dollars	(m)	H07	Non-oper	Financial p	33,020	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
7	2021	Level1		99999	All Industri	Dollars	(m)	H09	Total expe	Financial p	6,54,404	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
8	2021	Level1		99999	All Industri	Dollars	(m)	H10	Interest ar	Financial p	26,138	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
9	2021	Level1		99999	All Industri	Dollars	(m)	H10	Indirect ta	Financial p	6,991	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
10	2021	Level1		99999	All Industri	Dollars	(m)	H11	Depreciat	Financial p	27,801	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
11	2021	Level1		99999	All Industri	Dollars	(m)	H12	Salaries an	Financial p	1,23,620	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
12	2021	Level1		99999	All Industri	Dollars	(m)	H13	Redundan	Financial p	275	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
13	2021	Level1		99999	All Industri	Dollars	(m)	H14	Salaries an	Financial p	2,085	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
14	2021	Level1		99999	All Industri	Dollars	(m)	H20	Purchases	Financial p	4,52,963	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
15	2021	Level1		99999	All Industri	Dollars	(m)	H21	Non-oper	Financial p	14,806	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
16	2021	Level1		99999	All Industri	Dollars	(m)	H22	Opening st	Financial p	68,896	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
17	2021	Level1		99999	All Industri	Dollars	(m)	H23	Closing st	Financial p	69,127	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
18	2021	Level1		99999	All Industri	Dollars	(m)	H24	Surplus be	Financial p	1,03,330	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
19	2021	Level1		99999	All Industri	Dollars	(m)	H25	Total asse	Financial p	25,12,677	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
20	2021	Level1		99999	All Industri	Dollars	(m)	H26	Current as	Financial p	7,30,587	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
21	2021	Level1		99999	All Industri	Dollars	(m)	H29	Fixed tang	Financial p	5,91,351	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
22	2021	Level1		99999	All Industri	Dollars	(m)	H30	Other asse	Financial p	11,90,739	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
23	2021	Level1		99999	All Industri	Dollars	(m)	H31	Total equi	Financial p	25,12,677	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
24	2021	Level1		99999	All Industri	Dollars	(m)	H32	Sharehold	Financial p	8,13,949	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
25	2021	Level1		99999	All Industri	Dollars	(m)	H33	Current lia	Financial p	9,33,093	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
26	2021	Level1		99999	All Industri	Dollars		H34	Other labi	Financial p	7,65,635	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						
									Total incor	Financial r:	4,00,900	ANZSIC06	divisions A-S	(excluding classes K6330, L6711, 07552, 0760, 0771, 0772, 09540, S9601, S9602, and S9603)						

1. Open command prompt and give the following command as shown below.

Command syntax: `snowsql -a <Account> -u <Username>`

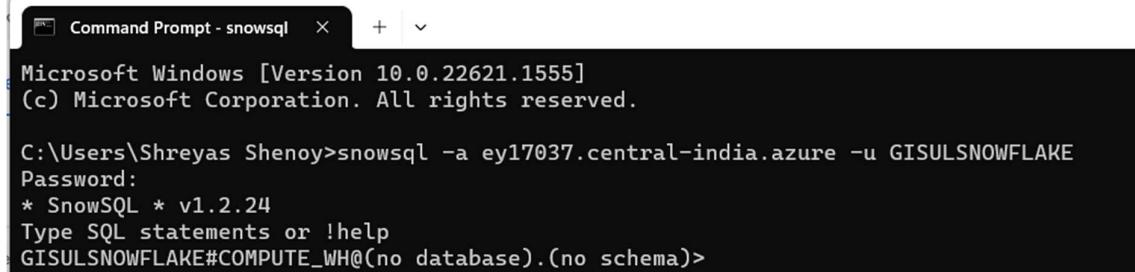
Command: `snowsql -a ey17037.central-india.azure -u GISULSNOWFLAKE`



```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shreyas Shenoy>snowsql -a ey17037.central-india.azure -u GISULSNOWFLAKE
Password:
```

2. Give the account Password and click enter.



```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shreyas Shenoy>snowsql -a ey17037.central-india.azure -u GISULSNOWFLAKE
Password:
* SnowSQL * v1.2.24
Type SQL statements or !help
GISULSNOWFLAKE#COMPUTE_WH@(no database).(no schema)>
```

3. Now to connect with the Warehouse give the below command.

Command: use warehouse COMPUTE_WH;

4. I am using my Warehouse.

```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shreyas Shenoy>snowsql -a ey17037.central-india.azure -u GISULSNOWFLAKE
Password:
* SnowSQL * v1.2.24
Type SQL statements or !help
GISULSNOWFLAKE#COMPUTE_WH@(no database).(no schema)>use warehouse COMPUTE_WH;
+-----+
| status |
+-----|
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 0.204s
GISULSNOWFLAKE#COMPUTE_WH@(no database).(no schema)>
```

5. Now to connect with database give below command.

Command: Use database SAMPLE_SNOW;

```
1 Row(s) produced. Time Elapsed: 0.204s
GISULSNOWFLAKE#COMPUTE_WH@(no database).(no schema)>use database SAMPLE_SNOW;
+-----+
| status |
+-----|
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 0.124s
GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.PUBLIC>
```

6. Now to connect with the schema give the below command.

Command: Use schema "Sample";

```
1 Row(s) produced. Time Elapsed: 0.124s
GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.PUBLIC>use schema "Sample";
+-----+
| status |
+-----|
| Statement executed successfully. |
+-----+
1 Row(s) produced. Time Elapsed: 0.126s
GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.Sample>
```

7. Create a table with the create command as shown below.

Query: create or Replace table ENTERPRICE(
YEAR VARCHAR(250),
INDUSTRY_AGGREGATION VARCHAR(250),
INDUSTRY_CODE VARCHAR(25),
INDUSTRY_NAME VARCHAR(250),
UNITS VARCHAR(150),
VARIABLE_CODE VARCHAR(250),
VARIABLE_NAME VARCHAR(250),
VARIABLE_CATEGORY VARCHAR(25),

```

    VALUE VARCHAR(250),
INDUSTRY_CODE_ANZSIC06 VARCHAR(250)
);

```

```

GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.Sample>create or Replace table ENTERPRICE (
    YEAR VARCHAR(250),
    INDUSTRY_AGGREGATION VARCHAR(250),
    INDUSTRY_CODE VARCHAR(25),
    INDUSTRY_NAME VARCHAR(250),
    UNITS VARCHAR(150),
    VARIABLE_CODE VARCHAR(250),
    VARIABLE_NAME VARCHAR(250),
    VARIABLE_CATEGORY VARCHAR(25),
    VALUE VARCHAR(250),
    INDUSTRY_CODE_ANZSIC06 VARCHAR(250)
);

+-----+
| status
|-----|
| Table ENTERPRICE successfully created. |
+-----+
1 Row(s) produced. Time Elapsed: 0.437s
GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.Sample>

```

8. We will load the CSV data file from your local system to the staging of the Snowflake as shown below.

Command: put file://C:\Users\Public\AnnualEnterprise.csv
 @"SAMPLE_SNOW"."Sample".%Enterprise;

```

1 Row(s) produced. Time Elapsed: 0.437s
GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.Sample>put file://C:\Users\Public\AnnualEnterprise.csv @"SAMPLE_SNOW"."Sample".%Enterprise;
+-----+
| source          | target           | source_size | target_size | source_compression | target_compression | status      | message
+-----+
| AnnualEnterprise.csv | AnnualEnterprise.csv.gz | 6618545 | 310256 | NONE | GZIP | UPLOADED |
+-----+
1 Row(s) produced. Time Elapsed: 2.072s
GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.Sample>

```

9. To check the load details use the below command.

Command: list @"SAMPLE_SNOW"."Sample".%Enterprise;

```

1 Row(s) produced. Time Elapsed: 2.072s
GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.Sample>list @"SAMPLE_SNOW"."Sample".%Enterprise;
+-----+
| name        | size | md5          | last_modified
+-----+
| AnnualEnterprise.csv.gz | 310256 | 17b9709e2f9456d06bd4dfd0d93c15c0 | Wed, 26 Apr 2023 08:23:28 GMT
+-----+
1 Row(s) produced. Time Elapsed: 0.200s
GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.Sample>

```

10. Copy the data into the Target Table using the below command.

Command: copy into Enterprise from @"%Enterprise file_format = (type = csv skip_header = 1 field_optionally_enclosed_by=""") pattern = '*.AnnualEnterprise.csv.gz' on_error = 'skip_file';

```

1 Row(s) produced. Time Elapsed: 0.159s
GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.Sample>copy into Enterprise from @"%Enterprise file_format = (type = csv skip_header = 1 field_optionally_enclosed_by=""
") pattern = '*.AnnualEnterprise.csv.gz' on_error = 'skip_file';
+-----+
| file          | status | rows_parsed | rows_loaded | error_limit | errors_seen | first_error | first_error_line | first_error_character | first_error_column_name
+-----+
| AnnualEnterprise.csv.gz | LOADED | 41715 | 41715 | 1 | 0 | NULL | NULL | NULL | NULL
+-----+
1 Row(s) produced. Time Elapsed: 1.500s
GISULSNOWFLAKE#COMPUTE_WH@SAMPLE_SNOW.Sample>

```

11. Now our csv data is loaded into the target table.

12. Go to Snowflake and refresh the database then check whether our table is created or not.

The screenshot shows the Snowflake interface under the 'Databases' tab. A new database named 'SAMPLE_SNOW' has been created. Inside 'SAMPLE_SNOW', a schema named 'PUBLIC' contains a table named 'Sample'. This table has a single row named 'ENTERPRISE'. Below the table is another table named 'SAMPLE_TBL'. The 'Tables' section also lists 'Snowflake' and 'SNOWFLAKE'.

13. Now to display the records use the select command in the worksheet and give a limit of 100 as shown below.

Query: select * from Enterprise limit 100;

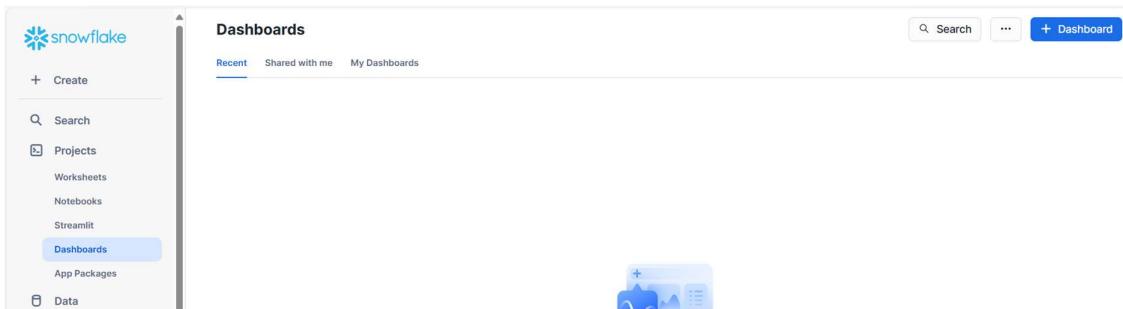
The screenshot shows the Snowflake interface under the 'Worksheets' tab. A query is being run: 'Select * from ENTERPRISE limit 100;'. The results show 12 rows of data. The columns are: YEAR, INDUSTRY_AGGREGATION, INDUSTRY_CODE, INDUSTRY_NAME, and UNITS. The data is as follows:

	YEAR	INDUSTRY_AGGREGATION	INDUSTRY_CODE	INDUSTRY_NAME	UNITS
1	2021	Level 1	99999	All industries	Dollars (millions)
2	2021	Level 1	99999	All industries	Dollars (millions)
3	2021	Level 1	99999	All industries	Dollars (millions)
4	2021	Level 1	99999	All industries	Dollars (millions)
5	2021	Level 1	99999	All industries	Dollars (millions)
6	2021	Level 1	99999	All industries	Dollars (millions)
7	2021	Level 1	99999	All industries	Dollars (millions)
8	2021	Level 1	99999	All industries	Dollars (millions)
9	2021	Level 1	99999	All industries	Dollars (millions)
10	2021	Level 1	99999	All industries	Dollars (millions)
11	2021	Level 1	99999	All industries	Dollars (millions)
12	2021	Level 1	99999	All industries	Dollars (millions)

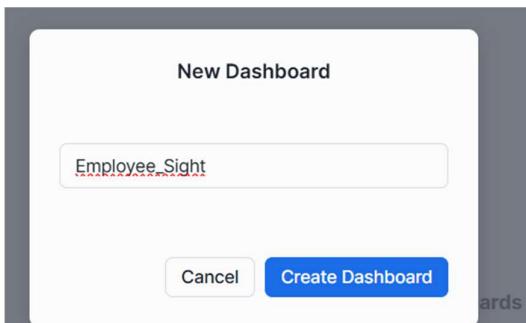
Snowsight

Snowflake Snowsight is a web-based interface that lets you query, visualize, and share data in Snowflake. Snowsight simplifies data tasks for users of all data roles, by providing an intuitive graphical interface, built-in data visualization capabilities, simplified data access and security controls—and collaboration features. Snowsight is accessible via any modern browser and can help you perform advanced data analysis and reporting more efficiently and effectively, without requiring extensive coding skills or complex/lengthy setup processes.

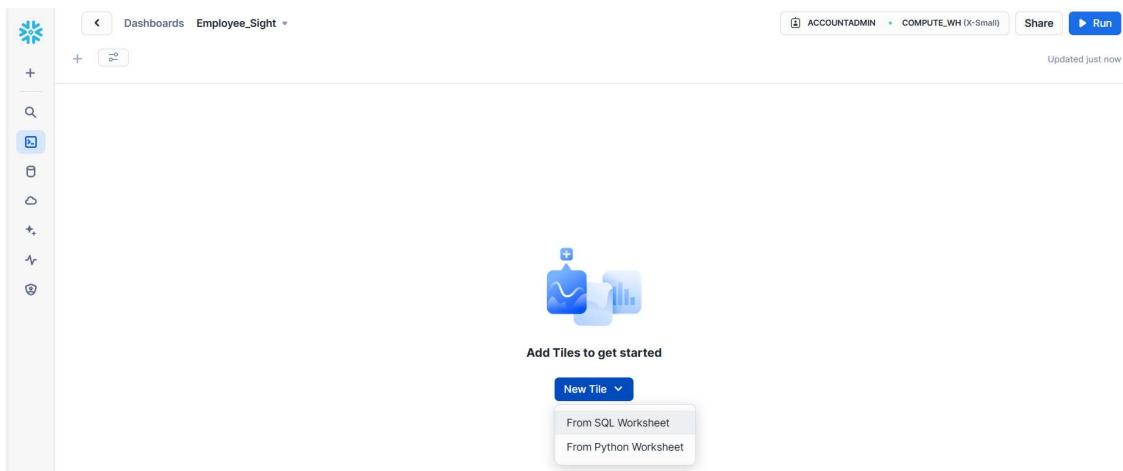
1. In this example we are using the Employee Details table that we created before.
2. Go to the Home page and go to Dashboard click on the plus symbol.



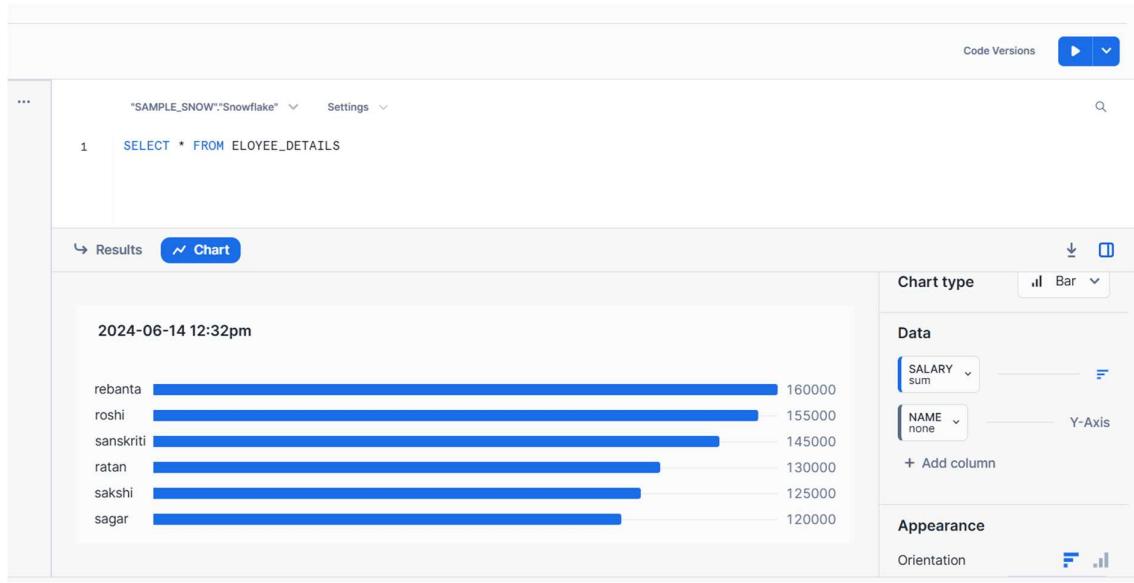
3. Give a name and click on Create.



4. Select the warehouse and click on Select from SQL Worksheet as shown below.



5. Give a query create a column chart with the below properties and click on Run.



6. Click on Return to Employee sight and you will see a chart like below.



Micro-partitions

Micro-partitions are the foundational unit of data storage in Snowflake. Each micro-partition holds between 50 MB and 500 MB of compressed data in a columnar format, enabling efficient storage and rapid query performance. When data is loaded into a Snowflake table, it is automatically divided into micro-partitions based on the natural order of the incoming data. Each micro-partition includes metadata, such as min and max values for each column, null counts, and the number of distinct values, which aids in optimizing query performance by reducing the amount of data scanned during queries.

1. Create a table and load data by using the below query.

Query: CREATE OR REPLACE TABLE sales (

```
sale_id INTEGER,  
sale_date DATE,  
customer_id INTEGER,  
product_id INTEGER,  
amount DECIMAL(10,2)  
);
```

-- Insert sample data

```
INSERT INTO sales (sale_id, sale_date, customer_id, product_id, amount) VALUES  
(1, '2023-01-01', 1001, 2001, 150.00),  
(2, '2023-01-02', 1002, 2002, 200.00),  
(3, '2023-01-03', 1003, 2003, 250.00),  
(4, '2023-01-04', 1004, 2004, 300.00),  
(5, '2023-01-05', 1005, 2005, 350.00);
```

The screenshot shows the Snowflake UI interface. At the top, there's a navigation bar with 'ACCOUNTADMIN' and 'COMPUTE_WH (X-Small)' and a 'Share' button. Below the navigation is a code editor area with syntax highlighting for SQL. The code in the editor is:

```
CREATE OR REPLACE TABLE sales (  
    sale_id INTEGER,  
    sale_date DATE,  
    customer_id INTEGER,  
    product_id INTEGER,  
    amount DECIMAL(10,2)  
);  
-- Insert sample data  
INSERT INTO sales (sale_id, sale_date, customer_id, product_id, amount) VALUES  
(1, '2023-01-01', 1001, 2001, 150.00),  
(2, '2023-01-02', 1002, 2002, 200.00),  
(3, '2023-01-03', 1003, 2003, 250.00),  
(4, '2023-01-04', 1004, 2004, 300.00),  
(5, '2023-01-05', 1005, 2005, 350.00);
```

Below the code editor is a results table with one row of data. The table has columns for 'number of rows inserted' (5) and 'Query Details' (Query duration: 2.4s). The bottom right corner of the UI shows the page number 'Page | 24'.

2. Let's run a query to retrieve sales data for a specific date range. Snowflake will use the metadata stored in the micro-partitions to efficiently scan only the relevant partitions.

Query: `SELECT sale_id, sale_date, customer_id, product_id, amount`

`FROM sales`

`WHERE sale_date BETWEEN '2023-01-02' AND '2023-01-04';`

```
"SAMPLE_SNOW"."Snowflake" ▾ Settings ▾
Code Versions Q
1 | SELECT sale_id, sale_date, customer_id, product_id, amount
2 | FROM sales
3 | WHERE sale_date BETWEEN '2023-01-02' AND '2023-01-04';
4

↳ Results ⚡ Chart
Query Details ...
Query duration 1.2s
Rows 3
```

	SALE_ID	SALE_DATE	CUSTOMER_ID	...	PRODUCT_ID	AMOUNT
1	2	2023-01-02	1002		2002	200.00
2	3	2023-01-03	1003		2003	250.00
3	4	2023-01-04	1004		2004	300.00

How Micro-partitions Optimize the Query:

- Metadata Pruning:** Snowflake examines the metadata of each micro-partition to determine if the sale_date range falls within the min and maximum values of the partition. Only the relevant micro-partitions are scanned.
- Columnar Access:** Since the data is stored in a columnar format, Snowflake only reads the columns specified in the query (sale_id, sale_date, customer_id, product_id, amount), reducing I/O and improving performance.
- Compression:** The compressed data within the micro-partitions is decompressed on the fly, minimizing storage requirements and speeding up data retrieval.

Data Clustering

Clustering is a technique used to organize data storage to better accommodate anticipated queries. The key objective is to increase query performance while lowering the amount of system resources needed to run queries.

Create Clustered Tables

1. A clustering key can be defined when a table is created by appending a CLUSTER BY clause to CREATE TABLE as shown below.

Query: CREATE TABLE MY_TABLE (

```
    type NUMBER  
    , name VARCHAR(50)  
    , country VARCHAR(50)  
    , date DATE  
)
```

```
CLUSTER BY (date) ;
```

"SAMPLE_SNOW"."Snowflake" ▾ Settings ▾

```
1   CREATE TABLE MY_TABLE (  
2       type NUMBER  
3       , name VARCHAR(50)  
4       , country VARCHAR(50)  
5       , date DATE  
6   )  
7   CLUSTER BY (date) ;
```

→ Results ▾ Chart

status

Table MY_TABLE successfully created.

2. In the above example we have created a table with clustered key.
3. However you can also cluster a table by specifying a CLUSTER KEY at a later point in time using ALTER TABLE as shown below.

Query: ALTER TABLE MY_TABLE CLUSTER BY (date) ;

"SAMPLE_SNOW"."Snowflake" ▾ Settings ▾

```
ALTER TABLE MY_TABLE  
CLUSTER BY (date) ;
```

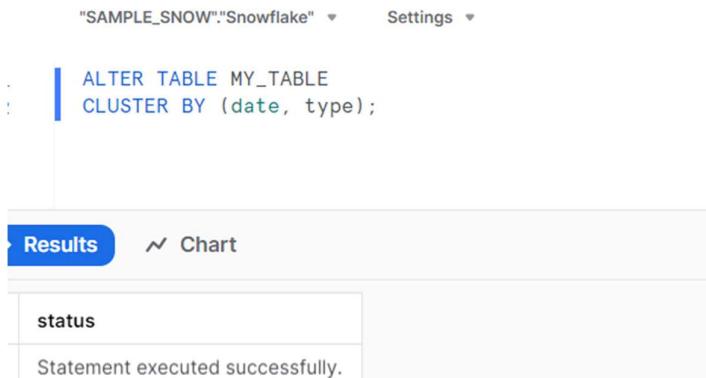
Results ▾ Chart

status

Statement executed successfully.

4. Although we have just used one field as a clustering key in our example, this is not the only choice.
5. A clustering key can be defined using many fields if desired as shown below.

Query: ALTER TABLE MY_TABLE CLUSTER BY (date, type) ;



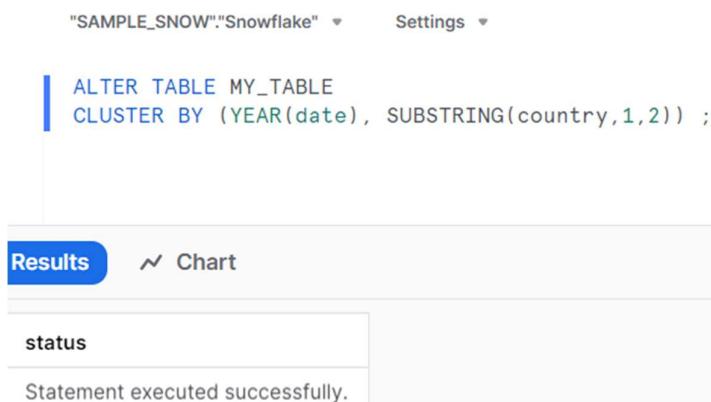
The screenshot shows a Snowflake query editor interface. At the top, it says "SAMPLE_SNOW"."Snowflake" and "Settings". Below that is a code editor with the following SQL statement:

```
ALTER TABLE MY_TABLE
CLUSTER BY (date, type);
```

Below the code editor is a results pane with tabs for "Results" (which is selected) and "Chart". The results pane displays the word "status" and the message "Statement executed successfully.".

6. Another advantage is that Snowflake also supports expressions on fields in a cluster key.
7. The below example shows the table is clustered using the year of the date field and the first two letters of the country field.

Query: ALTER TABLE MY_TABLE CLUSTER BY (YEAR(date), SUBSTRING(country,1,2)) ;



The screenshot shows a Snowflake query editor interface. At the top, it says "SAMPLE_SNOW"."Snowflake" and "Settings". Below that is a code editor with the following SQL statement:

```
ALTER TABLE MY_TABLE
CLUSTER BY (YEAR(date), SUBSTRING(country,1,2)) ;
```

Below the code editor is a results pane with tabs for "Results" (selected) and "Chart". The results pane displays the word "status" and the message "Statement executed successfully.".

Automatic Clustering

Automatic Clustering is the Snowflake service that seamlessly and continually manages all reclustering, as needed, of clustered tables. Note that Automatic Clustering consumes Snowflake credits, but does not require you to provide a virtual warehouse. Instead, Snowflake internally manages and achieves efficient resource utilization for reclustering the tables.

1. To suspend Automatic Clustering for a table, use the ALTER TABLE command with a SUSPEND RECLUSTER clause.

Query: ALTER TABLE my_table SUSPEND RECLUSTER ;

```
"SAMPLE_SNOW"."Snowflake" ▾      Settings ▾  
- :   ALTER TABLE my_table  
:     SUSPEND RECLUSTER ;
```

Results ↗ Chart

status
Statement executed successfully.

2. To resume Automatic Clustering for a clustered table, use the ALTER TABLE command with a RESUME RECLUSTER clause.

Query: ALTER TABLE my_table RESUME RECLUSTER ;

```
"SAMPLE_SNOW"."Snowflake" ▾      Settings ▾  
- :   ALTER TABLE my_table  
:     RESUME RECLUSTER ;
```

Results ↗ Chart

status
Statement executed successfully.

3. Verify if Automatic Clustering is enabled for a table using the SHOW TABLES command.

Query: SHOW TABLES LIKE 'my_table';

The screenshot shows the Snowflake WebUI interface. At the top, it says "SAMPLE_SNOW"."Snowflake" and "Settings". Below that is a code editor with the query: "SHOW TABLES LIKE 'my_table';". Underneath the code editor is a results table with the following columns: created_on, name, database_name, schema_name, kind, and comment. The data in the table is:

created_on	name	database_name	schema_name	kind	comment
2023-04-27 22:58:14.883 -0700	MY_TABLE	SAMPLE_SNOW	Snowflake	TABLE	

Sequence

Sequences are a common way to generate unique, sequential numbers. They can be used to allocate primary keys, auto-increment values, or other types of unique identifiers.

Sequences in Snowflake are also used to generate unique numbers with auto-increment functionality based on a defined interval. They guarantee uniqueness across multiple statements and sessions, ensuring that each execution of the sequence produces a distinct value.

1. We need to create a database to house the objects for this sequence generation.
2. Execute the following commands either in Snowflake WebUI or SnowSQL command-line client.

Query: CREATE DATABASE my_awesome_db;

CREATE SEQUENCE my_sequence;

The screenshot shows the Snowflake WebUI interface. At the top, it says "MY_AWESOME_DB.PUBLIC" and "Settings". Below that is a code editor with the following queries:
1. CREATE DATABASE my_awesome_db;
2. CREATE SEQUENCE my_sequence;.

Underneath the code editor is a results table with the following columns: status. The data in the table is:

status
1 Sequence MY_SEQUENCE successfully created.

On the right side of the interface, there are two panels: "Query Details" and "Query duration".

3. Now, let's retrieve a value from the newly created sequence. You can now use the NEXTVAL function with sequence to generate unique numbers.
4. For example, the following statement will return the next value in the my_sequence sequence.

Query: SELECT my_sequence.NEXTVAL;

The screenshot shows a database interface with a query editor and a results table. The query editor contains the SQL statement:

```
1 | SELECT my_sequence.NEXTVAL;
```

The results table has one row with the value 1 in the NEXTVAL column. A sidebar on the right shows 'Query Details' and 'Query duration'.

	NEXTVAL
1	1

5. After the sequence has been incremented, running the same SELECT statement again will provide the next value in the sequence.

Query: SELECT my_sequence.NEXTVAL;

The screenshot shows a database interface with a query editor and a results table. The query editor contains the SQL statement:

```
1 | SELECT my_sequence.NEXTVAL;
```

The results table has one row with the value 2 in the NEXTVAL column. A sidebar on the right shows 'Query Details' and 'Query duration'.

	NEXTVAL
1	2

6. This time, the query will return the next value, which should be 2.
7. To validate that the sequence always returns unique values, even within the same statement, execute the following SQL.

Query: SELECT my_sequence.NEXTVAL, my_sequence.NEXTVAL, my_sequence.NEXTVAL, my_sequence.NEXTVAL, my_sequence.NEXTVAL, my_sequence.NEXTVAL;

The screenshot shows a database interface with a query editor and a results table. The query editor contains the SQL statement:

```
1 | SELECT my_sequence.NEXTVAL, my_sequence.NEXTVAL, my_sequence.NEXTVAL, my_sequence.NEXTVAL, my_sequence.NEXTVAL, my_sequence.NEXTVAL;
```

The results table has one row with values 3, 4, 5, 6, 7, and 8 in the NEXTVAL columns respectively. A sidebar on the right shows 'Query Details' and 'Query duration'.

	NEXTVAL	NEXTVAL	NEXTVAL	NEXTVAL	NEXTVAL	...	NEXTVAL
1	3	4	5	6	7		8

8. Sequences can also be used to populate columns in a table.
9. Let's create a custom sequence with custom start and increment values.
10. The following example shows how to create a sequence & table and populate a column with values from a sequence.

Query: CREATE SEQUENCE MY_CUSTOM_SEQUENCE

START WITH 5

INCREMENT BY 5;

```
MY_AWESOME_DB.PUBLIC <--> Settings
CREATE SEQUENCE MY_CUSTOM_SEQUENCE
START WITH 5
INCREMENT BY 5;
```

Results

status
Sequence MY_CUSTOM_SEQUENCE successfully created.

Query

11. To test the sequence with the custom start and increment values, execute the following SQL.

Query: SELECT MY_CUSTOM_SEQUENCE.NEXTVAL, MY_CUSTOM_SEQUENCE.NEXTVAL,
MY_CUSTOM_SEQUENCE.NEXTVAL, MY_CUSTOM_SEQUENCE.NEXTVAL;

```
MY_AWESOME_DB.PUBLIC <--> Settings <--> Code Versions
SELECT MY_CUSTOM_SEQUENCE.NEXTVAL, MY_CUSTOM_SEQUENCE.NEXTVAL, MY_CUSTOM_SEQUENCE.NEXTVAL, MY_CUSTOM_SEQUENCE.NEXTVAL;
```

Results

NEXTVAL	NEXTVAL	...	NEXTVAL	NEXTVAL
5	10		15	20

Query Details

Query duration: 216ms

12. Sequences can be used to populate incremental values in table columns. Let's create a new table and insert data into one of its columns using a sequence.
13. Execute the following SQL statements.

Query: CREATE TABLE MY_TABLE (
ID INT,
NAME STRING
);

```
MY_AWESOME_DB.PUBLIC <--> Settings <--> Code Versions
CREATE TABLE MY_TABLE (
ID INT,
NAME STRING
);
```

Results

status
Table MY_TABLE successfully created.

Query Details

Query duration: 1

14. Let's write a command that inserts some rows into the MY_TABLE table.
15. We will populate the ID column using the MY_CUSTOM_SEQUENCE sequence, which ensures that each row receives a unique and incremental identifier.
16. Also, we will fill the NAME column with custom string values. You can either add your own custom strings or use random strings if desired.

Query: `INSERT INTO MY_TABLE (ID, NAME)`

```
SELECT MY_CUSTOM_SEQUENCE.NEXTVAL, t.NAME
FROM (VALUES ('Apple'), ('Banana'), ('Cherry'), ('Potato'), ('Mango'), ('Tomato')) AS
t(NAME);
```

```
MY_AWESOME_DB.PUBLIC ▾ Settings ▾ Code Versions ▾
1 | INSERT INTO MY_TABLE (ID, NAME)
2 | SELECT MY_CUSTOM_SEQUENCE.NEXTVAL, t.NAME
3 | FROM (VALUES ('Apple'), ('Banana'), ('Cherry'), ('Potato'), ('Mango'), ('Tomato')) AS t(NAME);

↳ Results ▾ Chart
number of rows inserted
6
Query Details ...
```

17. To view the data inserted into the table, execute the following SQL.

Query: `SELECT * FROM MY_TABLE;`

```
MY_AWESOME_DB.PUBLIC ▾ Settings ▾ Code Versions ▾
1 | SELECT * FROM MY_TABLE;

↳ Results ▾ Chart
ID | NAME
1  | Apple
2  | Banana
3  | Cherry
4  | Potato
5  | Mango
6  | Tomato
Query Details ...
Query duration 214ms
Rows 6
Query ID 01b5003a-0001-2513...
ID #
```

18. As you can see, the output displays the inserted data, with the ID column showing auto-incremented values.

Monitor Usage and Storage

1. The user can check how many rows have individual tables as well as data size. If the user has access to the table, he/she can view these details by just selecting a table.
2. In the left-down panel, the user can see the Table Name then number of rows, and the data storage size. After that, it shows the column's definition of the table.

The screenshot shows the Snowflake interface. On the left, there's a sidebar with icons for database, schema, table, and history. The main area shows the PUBLIC schema with a Snowflake database selected. Under Tables, the EMPLOYEE schema is expanded, and the EMPLOYEE_DETAILS table is selected, highlighted with a blue background. To the right, a detailed view of the EMPLOYEE_DETAILS table is shown, including its 6 rows and a list of columns with their data types:

Column	Type
ID	NUMBER(38,0)
NAME	VARCHAR(30)
AGE	NUMBER(38,0)
SALARY	NUMBER(38,0)
GENDER	VARCHAR(30)
CONTACT_NUMBER	NUMBER(38,0)
EMAIL_ID	VARCHAR(30)

3. In this section, users can check their activities in a snowflake like what queries they are using, the current status of the query, how much time it took to run, etc.
4. To view the history, click the History tab present at the bottom Ribbon. It will show the user's history.

The screenshot shows the History tab in the Snowflake interface. At the top, there are filter options for 'Status' (set to 'All'), 'Time Range' (set to 'Last 24 hours'), and a search bar. Below this, a table lists 25 recent queries:

Time	Duration	Query
3:00:30 PM	214ms	SELECT * FROM MY_TAB...
2:59:35 PM	447ms	INSERT INTO MY_TABLE...
2:58:18 PM	128ms	CREATE TABLE MY_TABL...
2:57:27 PM	216ms	SELECT MY_CUSTOM_SEQ...
2:56:32 PM	54ms	CREATE SEQUENCE MY_C...
2:55:38 PM	203ms	SELECT my_sequence.N...
2:54:39 PM	178ms	SELECT my_sequence.N...
2:53:40 PM	203ms	SELECT my_sequence.N...
2:52:40 PM	61ms	CREATE SEQUENCE my_s...
2:52:35 PM	80ms	CREATE DATABASE my_a...
2:48:05 PM	26ms	SELECT SYSTEM\$CLUSTE...

- To perform account-level monitoring, the user must be logged in as an ACCOUNTADMIN role.
- It will show Account Usage by default. Users can see the number of warehouses created, how much credit has been used, average storage used (it means that how much data we scanned while running the query vs overall storage), and how much data transferred.

The screenshot shows the 'Cost Management' section of the Snowflake interface. On the left, a sidebar lists various administrative tasks like 'Create', 'Search', 'Projects', 'Data', etc., with 'Cost Management' selected. The main area displays 'Account spend for KC65683 from Jun 7 - Jun 14'. It shows metrics: Spend in currency (\$0.35), Spend in credits (0.35), Compute price/credit (0.04), and Average daily cost (0.04). Below this, a chart titled 'Top warehouses by cost' shows two entries: 'COMPUTE_WH' with a blue bar indicating a cost of 0.35, and 'CLOUD_SERVICES_ONLY' with a bar at 0.00. To the right, a note says 'Results unavailable' with a small icon.

- Click the tab USERS. It displays the names of all the users present in the account.

The screenshot shows the 'Users & Roles' section. The sidebar has 'Users & Roles' selected. The main table lists '2 Users' with columns: NAME, DISPLAY NAME, STATUS, LAST LOGIN, MFA, and OWNER. The first user is 'SNOWFLAKE' (Temporary, last login 5 hours ago, No MFA, no owner). The second user is 'SNOWFLAKE97098' (Enabled, last login just now, No MFA, OWNER: ACCOUNTADMIN). A 'User' button is visible at the top right.

- A new role can be created here by clicking the Create button present at the top of the role list. Selecting a role, gives the option to enable or delete the role as well, by clicking the Edit button and Drop respectively.

The screenshot shows the 'Users & Roles' section again, but this time the 'Roles' tab is selected. The sidebar has 'Cost Management' selected. The main table lists '6 Roles' with columns: NAME, GRANTED ROLES, GRANTED TO ROLES, USERS, CREATED, and OWNER. The roles listed are: ACCOUNTADMIN (2 granted roles, 0 granted to, 1 user, created 5 hours ago, no owner), ORGADMIN (0 granted roles, 0 granted to, 1 user, created 5 hours ago, no owner), PUBLIC (0 granted roles, 0 granted to, 0 users, created 5 hours ago, no owner), SECURITYADMIN (1 granted role, 1 granted to, 0 users, created 5 hours ago, no owner), SYSADMIN (0 granted roles, 1 granted to, 0 users, created 5 hours ago, no owner), and USERADMIN (0 granted roles, 1 granted to, 0 users, created 5 hours ago, no owner). A 'Role' button is visible at the top right.