Target Audience: This session is ideal for individuals with varying levels of experience, from beginners with no prior knowledge of Snowflake to those with some SQL experience looking to explore Snowflake's functionalities.

Session Objectives:

- Understand the core concepts of Snowflake, including its architecture and key features.
- Gain hands-on experience setting up a Snowflake account and navigating the web interface.
- Learn how to load and manage data within Snowflake.
- Write basic and complex SQL queries to analyze data in Snowflake.
- Explore advanced features like views, materialized views, and Time Travel

Session Agenda

Module 1: Introduction to Snowflake

1. **What is Snowflake?**
    - **Definition**: Snowflake is a cloud-based data warehouse and data lakehouse solution.
    - **Benefits**:
        - Scalability: Seamlessly scales up and down based on demand.
        - Security: Offers robust security features including data encryption and role-based access control.
        - Ease of Use: Simplified management and intuitive interface.
2. **Key Snowflake Concepts**
    - **Cloud Storage**: Snowflake leverages cloud storage to provide a scalable and flexible data storage solution.
    - **Virtual Warehouses**: Compute resources in Snowflake that perform data processing tasks.
    - **Users and Roles**: Define different users and roles for access control and security within the Snowflake platform.

Module 2: Setting Up a Snowflake Account

1. **Instructions**: Step-by-step guide to setting up a free trial Snowflake account.
2. **Web Interface Tour**:
    - Navigation: Main sections of the interface including Databases, Warehouses, Worksheets, and History.

- Key Functionalities: How to run queries, manage resources, and monitor activities.

Module 3: Data Loading and Management

1. **Supported Data Formats**: Overview of data formats supported by Snowflake (CSV, Parquet, JSON, etc.).
2. **Loading Data with COPY Command**: Demonstration of the COPY command for loading data from external sources.
3. **Data Management Practices**:
   - Best practices for organizing and managing data within Snowflake.
   - Data partitioning and clustering techniques to enhance performance.

Module 4: SQL Queries in Snowflake

Introduction to SQL: Brief overview of SQL concepts for beginners (SELECT, FROM, WHERE, etc.).

1. **Writing Basic and Complex Queries**: How to write SQL queries to retrieve and manipulate data.
2. **Using Functions and Aggregations**: Common SQL functions (SUM, COUNT, AVG) and aggregation techniques.
3. **Joining Tables and Subqueries**: Techniques for joining multiple tables and using subqueries for complex data retrieval.

Module 5: Advanced Snowflake Features

1. **Views and Materialized Views**:
   - Definition and use cases for views and materialized views.
   - Benefits for optimized querying performance.
2. **Snowflake's Time Travel Feature**: Allows you to query, clone, and restore data to a previous state.
3. **Snowflake's Zero-Copy Cloning Feature**: Enables creating a copy of your data without additional storage costs.

Module 1: Introduction to Snowflake

Solution	Description	Purpose	Data Structure	Scalability	Integration
Database	Real-time, transactional data management with structured data. Limited scalability and integration capabilities.	Transactional data management	Structured data	Limited	Limited
Data Warehouse	Optimized for analytical queries and reporting with structured and semi-structured data. Expensive and moderately scalable.	Analytical queries and reporting	Structured and semi-structured data	Moderate	Requires ETL processes
Data Lake	Suitable for storing vast amounts of raw data in various formats. High scalability but low data integrity without additional frameworks.	Storage of raw data	Raw, unstructured, semi-structured, structured	High	High (requires additional frameworks for data integrity)
Delta Lake	Enhances data lakes with ACID transactions, ensuring reliability and	Enhancing data lakes	Structured, unstructured, semi-structured	High	High with existing data lakes

Solution	Description	Purpose	Data Structure	Scalability	Integration
	enabling unified batch and stream processing. High scalability and data integrity.				
OneLake	Provides a unified and centralized data storage solution, integrating data from multiple sources and supporting comprehensive data management and analytics. High scalability and integration capabilities.	Centralized data storage and management	Structured, unstructured, semi-structured	High	Very high with various data sources and tools
Data Lakehouse	Combines the scalability and flexibility of data lakes with the reliability and performance of data warehouses. Supports both analytical and transactional workloads with ACID transactions. High scalability, performance, and data integrity.	Unified platform for analytical and transactional workloads	Structured, unstructured, semi-structured	High	High with ACID transactions and support for both workloads

This table provides a clear and concise comparison of the different data storage solutions, highlighting their key features, purposes, data structures, scalability, and integration capabilities.

1. What is Snowflake?

Definition:

Snowflake is a cloud-based data warehouse and data lakehouse solution. It allows organizations to consolidate data from various sources into a single, scalable, and secure platform, enabling comprehensive data analysis and insights.

Benefits:

Scalability:

- **Seamless Scaling**: Snowflake's architecture allows it to automatically scale up or down based on the workload. This means it can handle small data sets as well as extremely large ones without any performance issues.
- **Concurrency Handling**: Supports multiple concurrent users and queries without degradation in performance, making it ideal for large organizations and collaborative environments.

Security:

- **Data Encryption**: Snowflake provides end-to-end encryption for data at rest and in transit, ensuring data privacy and protection.
- **Role-Based Access Control (RBAC)**: Allows administrators to define roles and permissions, ensuring that only authorized users can access sensitive data.
- **Compliance**: Snowflake complies with various industry standards and regulations such as GDPR, HIPAA, and SOC 2, providing confidence to enterprises regarding data security.

Ease of Use:

- **Simplified Management**: Snowflake eliminates the need for traditional database management tasks such as indexing, partitioning, and vacuuming. Its cloud-native architecture handles these automatically.
- **Intuitive Interface**: The web-based user interface is designed to be user-friendly, allowing users to easily navigate and manage their data without extensive training.
- **Quick Setup**: Setting up a Snowflake account is straightforward and can be done in minutes, allowing users to start analyzing data quickly.

- **Cross-Cloud Compatibility**: Snowflake operates across multiple cloud platforms (AWS, Azure, Google Cloud), providing flexibility and reducing vendor lock-in.

2. Key Snowflake Concepts

Cloud Storage

- **Definition**: Snowflake leverages cloud storage to provide a scalable and flexible data storage solution.
- **Benefits**:
  - **Scalability**: Storage scales automatically as data volumes grow, ensuring you never run out of space.
  - **Flexibility**: Supports a wide range of data formats and types, including structured and semi-structured data (e.g., JSON, Avro, Parquet).
  - **Separation of Storage and Compute**: Storage and compute resources are decoupled, allowing each to scale independently. This leads to cost savings and performance optimization as you pay only for the storage you use and the compute resources you need.

Virtual Warehouses

- **Definition**: Virtual warehouses are compute resources in Snowflake that perform data processing tasks.
- **Characteristics**:
  - **Independent Scaling**: Virtual warehouses can be resized (up or down) based on the workload requirements without affecting other operations.
  - **Concurrency**: Multiple virtual warehouses can operate concurrently, handling different workloads without contention.
  - **Auto-Suspend and Auto-Resume**: Virtual warehouses can be configured to automatically suspend when not in use and resume when needed, optimizing resource usage and cost.
- **Usage**:
  - **Query Execution**: Execute SQL queries for data analysis and reporting.
  - **Data Loading**: Perform data loading operations from various sources into Snowflake.
  - **Data Transformation**: Process and transform data to prepare it for analysis or further processing.

Users and Roles

- **Definition**: Snowflake employs a role-based access control (RBAC) system to define different users and roles, ensuring secure and controlled access to data and resources.
- **Key Concepts**:

- **Users**: Individual accounts that access Snowflake. Each user can be assigned one or more roles.
  - **Roles**: Define a set of permissions that determine what actions a user can perform. Roles can be hierarchical, where a role can inherit permissions from another role.
  - **Permissions**: Specific access rights granted to roles, such as the ability to read, write, or modify data and resources.
- **Benefits**:
  - **Enhanced Security**: Ensures that users have only the permissions necessary for their tasks, minimizing security risks.
  - **Simplified Management**: Roles and permissions can be easily managed and updated as organizational needs change.
  - **Compliance**: Helps meet regulatory requirements by controlling and auditing access to sensitive data.

Summary:

Snowflake stands out as a versatile, scalable, and secure cloud data platform that simplifies data management and analysis, making it an excellent choice for organizations of all sizes looking to leverage their data more effectively.

Module 2: Setting Up a Snowflake Account

1. Instructions: Step-by-Step Guide to Setting Up a Free Trial Snowflake Account

Step 1: Visit Snowflake's Website

- Go to the Snowflake website: Snowflake Free Trial.

Step 2: Sign Up for a Free Trial

- Click on the "Start for Free" or "Get Started" button.
- Fill in the required information, including:
  - **Name**
  - **Email Address**
  - **Company Name**
  - **Country**
- Click on the "Continue" button.

Step 3: Account Setup

- Choose your preferred Snowflake edition and cloud provider (AWS, Azure, Google Cloud).
- Select the region closest to your location for optimal performance.
- Create your account by setting a username and password.

Step 4: Email Verification

- Check your email for a verification message from Snowflake.
- Click on the verification link in the email to activate your account.

Step 5: Login to Your Snowflake Account

- Once your account is verified, return to the Snowflake login page.
- Enter your username and password to log in.

Step 6: Initial Setup

- Upon logging in, you may be prompted to complete additional setup steps, such as configuring your organization's account settings.
- Follow the on-screen instructions to complete the setup process.

2. Web Interface Tour

Navigation: Main Sections of the Interface

1. **Databases**
   - **Overview**: Lists all databases accessible to the user.

- **Functions**: Create, manage, and query databases. Import and export data as needed.

2. **Warehouses**
   - **Overview**: Displays virtual warehouses available for processing data.
   - **Functions**: Create, start, stop, and manage virtual warehouses. Configure settings such as auto-suspend and auto-resume.
3. **Worksheets**
   - **Overview**: Interactive SQL editor where users can write and execute SQL queries.
   - **Functions**: Save and organize SQL scripts. Share worksheets with other users for collaboration.
4. **History**
   - **Overview**: Provides a log of all executed queries and operations.
   - **Functions**: View details of past queries, including execution time and status. Re-run or troubleshoot queries as needed.

Key Functionalities

1. **Running Queries**
   - **How-To**:
     - Navigate to the Worksheets section.
     - Enter your SQL query in the editor.
     - Click the "Run" button to execute the query.
   - **Tips**: Utilize Snowflake's SQL autocomplete and syntax highlighting features for efficient query writing.
2. **Managing Resources**
   - **Warehouses**:
     - Start or stop warehouses as needed based on workload.
     - Adjust size and concurrency settings to optimize performance and cost.
   - **Databases**:
     - Create new databases to organize your data.
     - Import data from various sources using the available tools and commands.
3. **Monitoring Activities**
   - **Query History**:
     - Access the History section to view details of all executed queries.
     - Monitor query performance and identify any issues or bottlenecks.
   - **Usage Metrics**:
     - Track resource usage and costs associated with virtual warehouses and storage.
     - Use this information to make informed decisions about resource allocation and budgeting.

Module 3: Data Loading and Management

1. Supported Data Formats

Overview of Supported Data Formats:

- **CSV (Comma-Separated Values)**: Commonly used for tabular data.
- **Parquet**: Columnar storage file format optimized for performance.
- **JSON (JavaScript Object Notation)**: Widely used for representing structured data.
- **Other Formats**: Avro, ORC, XML, and more.

2. Loading Data with the Web UI

Steps to Load Data Using the Snowflake Web UI:

1. **Login to Snowflake**:
   - Navigate to the Snowflake web interface and log in using your credentials.
2. **Create a Database and Schema**:
   - In the Databases section, create a new database if not already present.
   - Within the database, create a schema to organize your tables.
   - Navigate to the Databases section and click on the "+" icon to create a new database.
   - Enter the name of the database and click "Create".
   - Select the database and click on the "Create" button next to Schemas to add a new schema.
3. **Create a Stage**:
   - Go to the "Data" tab and select "Stages".
   - Click on "Create Stage" and specify the stage name and cloud storage location (e.g., AWS S3, Azure Blob Storage, Google Cloud Storage).
4. **Upload Data to the Stage**:
   - Use the web interface or a cloud storage tool to upload data files (CSV, Parquet) to the created stage.
5. **Create a Table**:
   - In the Worksheets section, create a table where the data will be loaded.
   - Example SQL:

     ```
     CREATE OR REPLACE TABLE my_table (
       id INT,
       name STRING,
       value DOUBLE
     );
     ```

6. **Load Data into the Table**:
   - Use the web UI to execute the COPY INTO command.
   - Example SQL:

     ```
     COPY INTO my_table
     FROM @my_stage/sample.csv
     FILE_FORMAT = (type = 'CSV', field_delimiter = ',', skip_header = 1);
     ```

3. Data Management Practices

Best Practices for Organizing and Managing Data:

- **Schema Design**: Use schemas to logically group related tables and objects.
- **Staging Areas**: Utilize staging areas to temporarily store data before loading into target tables.
- **Data Quality**: Implement data validation checks during the loading process to ensure data integrity.

Data Partitioning and Clustering Techniques:

- **Partitioning**: Use date or other relevant columns to partition data for better performance.
- **Clustering**: Organize data based on one or more columns to enhance query efficiency.

4. Handling Large Datasets

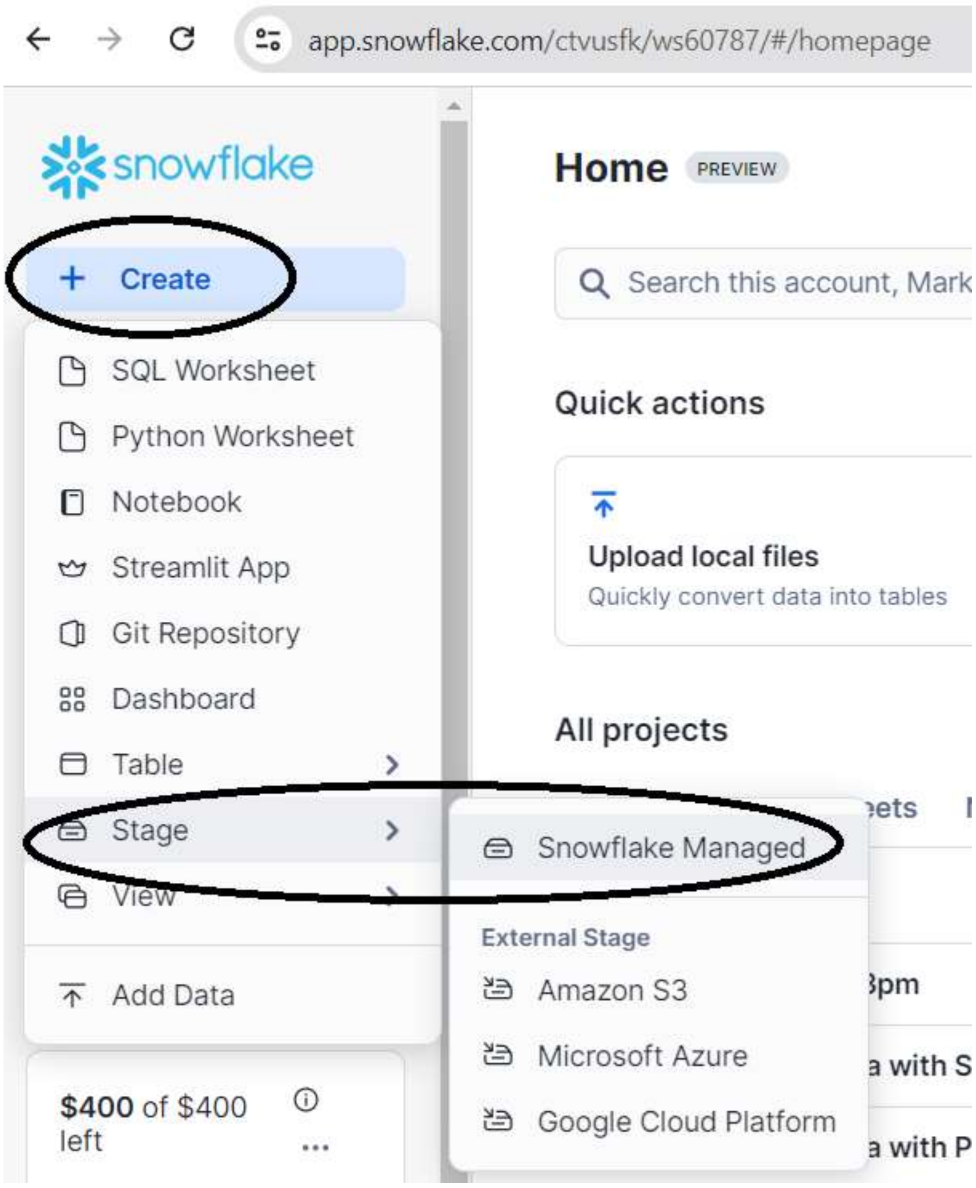
Introduction to Snowpipe for Continuous Data Ingestion:

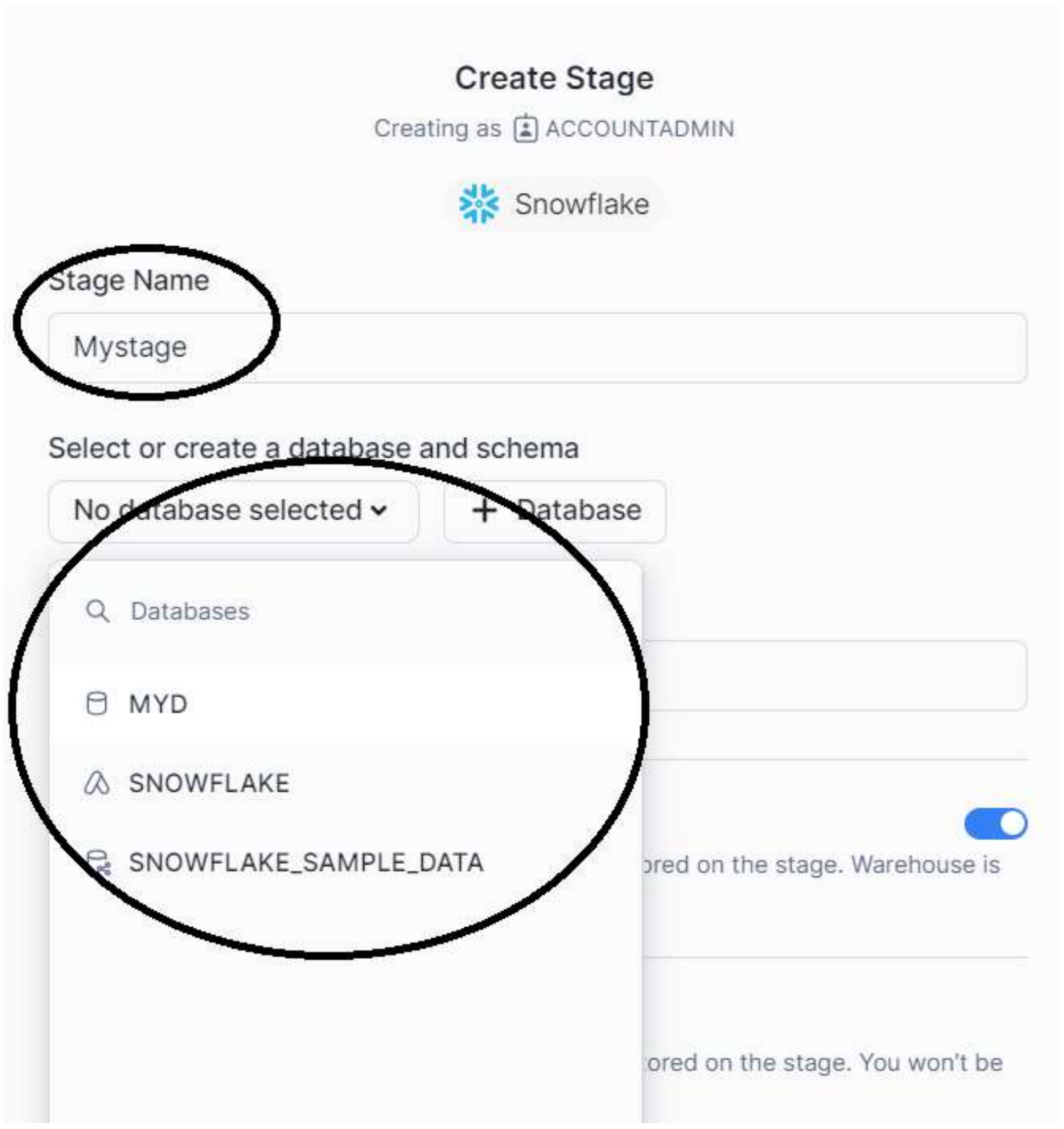
- **Automated Data Loading**: Snowpipe allows for continuous data ingestion.
- **Configuration**: Set up Snowpipe using the web UI to automatically load data as it arrives in cloud storage.

Hands-on Lab

1. Hands-on Practice: Using the Web UI to Load Data

- **Objective**: Load sample data in various formats (CSV, Parquet) into Snowflake using the web UI.
- **Steps**:
  1. **Create a Stage**:
     - Navigate to the "Data" tab and select "Stages".
     - Click "Create Stage" and specify the details for the stage.


app.snowflake.com/ctvusfk/ws60787/#/homepage


Create Stage

Creating as ACCOUNTADMIN

Snowflake

Stage Name

Mystage

Select or create a database and schema

No database selected

+ Database

Databases

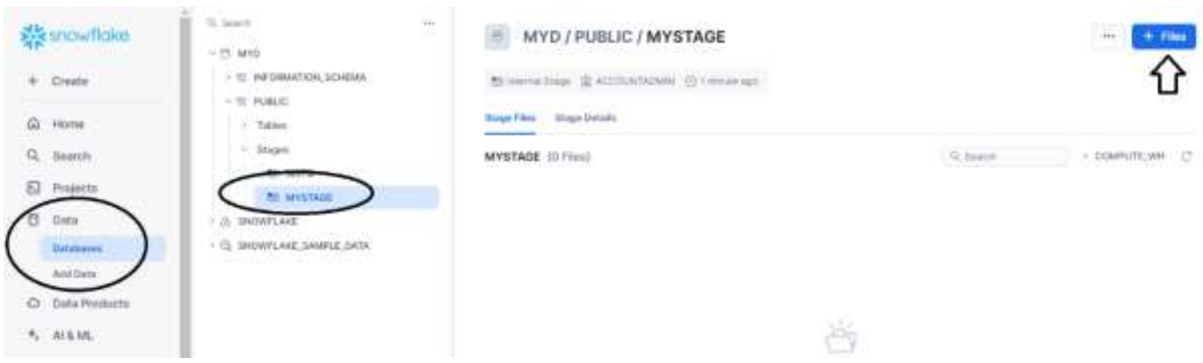
MYD

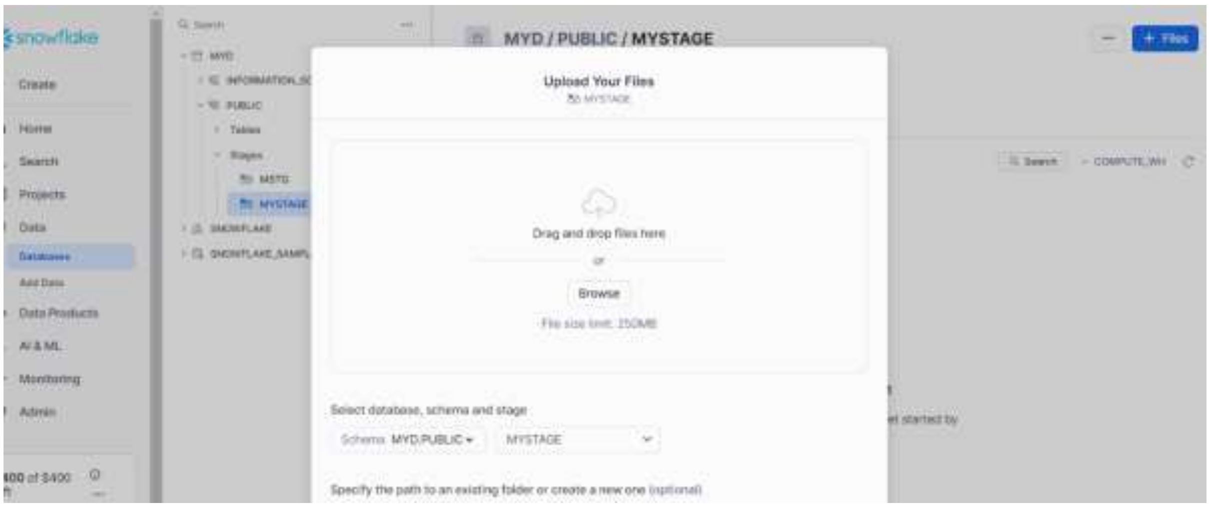
SNOWFLAKE

SNOWFLAKE_SAMPLE_DATA

ored on the stage. Warehouse is

ored on the stage. You won't be





2. **Upload Sample Data**:
   - Use the web UI or cloud storage tools to upload sample data files to the stage.
3. **Load Data into a Table**:
   - Navigate to the "Worksheets" section.
   - Use the COPY INTO command in the worksheet to load data into the table.

Sample Dataset:customer.csv

custid	name	Age	profession
c001	Ravi	23	Engineer
c008	Sonu	34	Actor
c003	Anand	28	Trainer
c004	Rajesh	35	Team Lead
c007	Sudheer	34	PM
c005	Amrut	40	HR Head

Example SQL:

create database mydb

use mydb

```
CREATE OR REPLACE TABLE customer (
  custid STRING,
  name STRING,age STRING,Prof STRING
  );
COPY INTO customer
FROM @mystage/customer.csv
FILE_FORMAT = (type = 'CSV', field_delimiter = ',', skip_header = 1);
select * from customer
```

Module 4: SQL Queries in Snowflake

Introduction to SQL

Snowflake supports most SQL data types:

Category	Type	Notes
Numeric Data Types	NUMBER	Default precision and scale are (38,0).
	DECIMAL, NUMERIC	Synonymous with NUMBER.
	INT, INTEGER, BIGINT, SMALLINT, TINYINT, BYTEINT	Synonymous with NUMBER except precision and scale cannot be specified.
	FLOAT, FLOAT4, FLOAT8	[1]
	DOUBLE, DOUBLE PRECISION, REAL	Synonymous with FLOAT. [1]
String & binary data types	VARCHAR	Default (and maximum) is 16,777,216 bytes.
	CHAR, CHARACTER	Synonymous with VARCHAR except default length is VARCHAR(1).
	STRING	Synonymous with VARCHAR.
	TEXT	Synonymous with VARCHAR.
	BINARY	
	VARBINARY	Synonymous with BINARY.
Logical Data Types	BOOLEAN	Currently only supported for accounts provisioned after January 25, 2016.
Date & Time Data Types	DATE	
	DATETIME	Alias for TIMESTAMP_NTZ
	TIME	
	TIMESTAMP	Alias for one of the TIMESTAMP variations (TIMESTAMP_NTZ by default).
	TIMESTAMP_LTZ	TIMESTAMP with local time zone; time zone, if provided, is not stored.
	TIMESTAMP_NTZ	TIMESTAMP with no time zone; time zone, if provided, is not stored.
	TIMESTAMP_TZ	TIMESTAMP with time zone.
	VARIANT	
	OBJECT	

Category	Type	Notes
Semi-structured Data Types	ARRAY	
Geospatial Data Types	GEOGRAPHY	
	GEOMETRY	
Vector Data Types	VECTOR	

Brief Overview of SQL Concepts for Beginners:

- **SELECT**: Used to select data from a database.
  - Example:

    ```
    SELECT column1, column2
    FROM table_name;
    ```

- **FROM**: Specifies the table from which to retrieve data.
  - Example:

    ```
    SELECT *
    FROM employees;
    ```

- **WHERE**: Filters records based on specified conditions.
  - Example:

    ```
    SELECT *
    FROM employee
    WHERE department_id = '1';
    ```

1. Writing Basic and Complex Queries

How to Write SQL Queries to Retrieve and Manipulate Data:

- **Basic Queries**:
  - **Retrieve Specific Columns**:

    ```
    SELECT first_name, last_name
    FROM employee;
    ```

  - **Filter Records**:

    ```
    SELECT *
    FROM employee
    WHERE salary > 50000;
    ```

- **Complex Queries**:
  - **Using Multiple Conditions**:

    ```
    SELECT first_name, last_name, salary
    FROM employee
    WHERE department_id = '2' AND salary > 60000;
    ```

    Sorting Results:
    ```
    SELECT first_name, last_name, salary
    FROM employee
    WHERE department_id = '2'
    ORDER BY salary DESC;
    ```

  - **Limiting Results**:

    ```
    SELECT first_name, last_name, salary
    FROM employee
    WHERE department_id = '1'
    ORDER BY salary DESC
    LIMIT 2;
    ```

2. Using Functions and Aggregations

Common SQL Functions and Aggregation Techniques:

- **SUM**: Calculates the total sum of a numeric column.
  - Example:

    ```
    SELECT SUM(salary)
    FROM employee
    WHERE department_id = '2';
    ```

- **COUNT**: Returns the number of rows that match a specified condition.
  - Example:

    ```
    SELECT COUNT(*)
    FROM employee
    WHERE department_id = '3';
    ```

- **AVG**: Calculates the average value of a numeric column.
  - Example:

    ```
    SELECT AVG(salary)
    FROM employee
    WHERE department_id = '2';
    ```

- **MAX**: Retrieves the maximum value in a column.
  - Example:

    ```
    SELECT MAX(salary)
    FROM employee;
    ```

- **MIN**: Retrieves the minimum value in a column.
  - Example:

    ```
    SELECT MIN(salary)
    FROM employee;
    ```

3. Joining Tables and Subqueries

Techniques for Joining Multiple Tables and Using Subqueries for Complex Data Retrieval:

Joining Tables:

- **INNER JOIN**: Selects records with matching values in both tables.
  - Example:

    ```
    SELECT employee.first_name, employee.last_name,
    departments.department_name
    FROM employee
    INNER JOIN departments
    ON employee.department_id = departments.department_id;
    ```

- **LEFT JOIN**: Selects all records from the left table, and matched records from the right table.
  - Example:

    ```
    SELECT employee.first_name, employee.last_name,
    departments.department_name
    FROM employee
    LEFT JOIN departments
    ON employee.department_id = departments.department_id;
    ```

- **RIGHT JOIN**: Selects all records from the right table, and matched records from the left table.
  - Example:

    ```
    SELECT employee.first_name, employee.last_name,
    departments.department_name
    FROM employee
    ```

```
RIGHT JOIN departments
ON employee.department_id = departments.department_id;
```

- **FULL JOIN**: Selects all records when there is a match in either left or right table.
  - Example:

```
SELECT employee.first_name, employee.last_name,
departments.department_name
FROM employee
FULL JOIN departments
ON employee.department_id = departments.department_id;
```

Subqueries:

  - Example:
- SELECT e.employee_id, e.first_name, e.last_name, d.department_name FROM employee e JOIN departments d ON e.department_id = d.department_id WHERE d.department_name = (SELECT department_name FROM departments WHERE department_id = 1);
-

Summary

By the end of this module, participants will understand the basic and advanced SQL concepts necessary to:

- Write basic and complex queries to retrieve and manipulate data.
- Use common SQL functions and aggregation techniques.
- Join multiple tables and use subqueries to perform complex data retrieval.

This knowledge will enable participants to efficiently analyze and work with data in Snowflake, preparing them for more advanced data operations.

Sample Datasets for "employees", "departments", and "sales_data".

Dataset 1: Employees

employee_id	first_name	last_name	department_id	salary	hire_date
1	John	Smith	1	50000	2021-01-15
2	Jane	Doe	2	60000	2020-03-22
3	Mary	Johnson	1	55000	2019-07-11
4	James	Brown	3	75000	2022-05-30
5	Patricia	Davis	2	62000	2021-10-02

employee_id	first_name	last_name	department_id	salary	hire_date
6	Robert	Martinez	3	68000	2020-11-12
7	Linda	Garcia	4	70000	2018-06-25
8	Michael	Rodriguez	4	72000	2019-09-17
9	Barbara	Wilson	1	48000	2020-12-01
10	William	Moore	2	53000	2019-02-14
11	Elizabeth	Taylor	3	75000	2021-08-21
12	David	Anderson	4	80000	2022-04-10
13	Susan	Thomas	1	51000	2017-11-23
14	Charles	Jackson	2	59000	2019-05-08
15	Karen	White	3	74000	2020-07-31
16	Joseph	Harris	4	82000	2021-03-19
17	Nancy	Martin	1	46000	2018-10-05
18	Thomas	Thompson	2	61000	2019-08-15
19	Lisa	Martinez	3	73000	2020-06-24
20	Christopher	Robinson	4	81000	2021-12-12

Dataset 2: Departments

department_id	department_name
1	Sales
2	HR
3	IT
4	Finance

Dataset 3: Sales_Data

sale_id	order_id	order_date	amount
1	101	2023-01-01	1000.50
2	102	2023-01-02	1500.00
3	103	2023-01-03	800.75
4	104	2023-01-04	1200.00
5	105	2023-01-05	950.25
6	106	2023-01-06	1100.00
7	107	2023-01-07	1450.50
8	108	2023-01-08	600.00
9	109	2023-01-09	850.00
10	110	2023-01-10	1300.75

sale_id	order_id	order_date	amount
11	111	2023-01-11	900.50
12	112	2023-01-12	1150.00
13	113	2023-01-13	1400.00
14	114	2023-01-14	700.00
15	115	2023-01-15	1050.25
16	116	2023-01-16	1250.75
17	117	2023-01-17	1350.00
18	118	2023-01-18	800.50
19	119	2023-01-19	950.75
20	120	2023-01-20	1500.50

Module 5: Advanced Snowflake Features

1. Views and Materialized Views

Definition and Use Cases:

- **Views**:
  - **Definition**: A view is a saved SQL query that you can treat as a table. It doesn't store data itself but dynamically retrieves data from underlying tables.
  - **Use Cases**: Simplifying complex queries, reusing query logic, abstracting underlying table structures.
- **Materialized Views**:
  - **Definition**: A materialized view is similar to a view, but it stores the result set of the query. It can be refreshed to keep data up-to-date.
  - **Use Cases**: Improving query performance for frequently accessed data, pre-computing complex aggregations.
- ☐ Static: Stores the results of the query at the time the view is created or refreshed.

- ☐ Performance: Querying a materialized view is faster as it avoids re-executing the underlying query.
- ☐ Use Case: Suitable for scenarios where query performance is critical and the data does not need to be real-time.

Benefits for Optimized Querying Performance:

- **Views**: Simplify complex SQL and provide a level of abstraction.
- **Materialized Views**: Improve performance by storing precomputed results, reducing the need to recompute for every query.

```
CREATE OR REPLACE TABLE sale(sale_id string,order_id string,amount int)

insert into sale values(1,101,1000.5);

insert into sale values(2,102,1500);

insert into sale values(3,103,800.75);

insert into sale values(4,104,1200);

insert into sale values(5,105,800)

select * from sale

drop view sales_view

CREATE VIEW sales_view AS

SELECT sale_id, SUM(amount) AS total_sales

FROM sale

GROUP BY sale_id;


select * from sales_view


CREATE VIEW sales_mview AS

SELECT sale_id,SUM(amount) AS total_sales

FROM sale
```

GROUP BY sale_id;

select * from sales_view

select * from sales_mview

Snowflake's Time Travel Feature

Allows You to Query, Clone, and Restore Data to a Previous State:

- **Definition**: Time Travel allows you to access historical data (i.e., data that has been changed or deleted) at any point within a defined retention period.
- **Use Cases**: Recovering from accidental data changes or deletions, auditing data changes, analyzing data at different points in time.

Snowflake's Zero-Copy Cloning Feature

Enables Creating a Copy of Your Data Without Additional Storage Costs:

- **Definition**: Zero-Copy Cloning allows you to create a clone of databases, schemas, and tables without copying the data, thus avoiding additional storage costs.
- **Use Cases**: Creating development and testing environments, creating backups, experimenting with data without affecting the original dataset.
- **Example**:
  - Cloning a table:

    CREATE CLONE employees_clone AS employees;

Summary

By the end of this module, participants will have a deep understanding of:

- Creating and using views and materialized views for efficient data retrieval.
- Writing and using UDFs to encapsulate custom logic.
- Implementing stored procedures to automate repetitive tasks.
- Utilizing Snowflake's Time Travel feature to access and restore historical data.
- Leveraging Snowflake's Zero-Copy Cloning feature to create data copies without additional storage costs.

This module will equip participants with advanced Snowflake features, enhancing their ability to manage and manipulate data efficiently.

Further Learning:

Get Certified: Free Test With Certificate on Scoring 80% and above

https://www.skillupguru.com/courses/Snowflake-Basic-Understanding-Quiz-65fced822578b770022a1a8a

Note: Suggested learning before test **The Ultimate Guide To Mastering Snowflake(Free Download Link Below)**

Complimentary learning link

https://www.skillupguru.com/courses/Snowflake--The-Native-Data-Warehouse-of-Cloud-661e7976ef9f7303cb1264b6

Download Link Free Book On Snowflake by Mukund Kumar Mishra

The Ultimate Guide To Mastering Snowflake

https://www.skillupguru.com/products/The-Ultimate-Guide-To-Mastering-Snowflake-667176bd33abdc6db5111201?dgps_s=pbl&dgps_u=c&dgps_uid=6597f5a5e4b058dd0cc6bd79&dgps_t=cp_m