IST Project - Data Pipeline

Students :

- Barros Fernandes Gabriel
- Bouzourène Ryad
- Nguyen Thomas
- Rothenbühler Lei

What is Snowflake ?

Snowflake is a fully managed SaaS (software as a service) that provides a single platform for data warehousing, data lakes, data engineering, data science, data application development, and secure sharing and consumption of real-time / shared data. Snowflake features out-of-the-box features like separation of storage and compute, on-the-fly scalable compute, data sharing, data cloning, and third-party tools support in order to handle the demanding needs of growing enterprises.

Why it's interesting ?

Snowflake has several advantages :

1. Scalability and performance

Snowflake allows independent scaling of compute and storage, optimizing costs and performance. Resources adjust dynamically based on workload demand, ensuring efficient query execution without manual intervention. Additionally, Snowflake leverages Massively Parallel Processing (MPP), enabling fast execution of complex queries across large datasets.

2. Multi-Cloud Flexibility

Snowflake is multi-cloud compatible, running with no interruptions on AWS, Azure, and Google Cloud. It supports cross-cloud data sharing, allowing businesses to exchange and access data across different cloud providers without migration overhead.

3. Ease of use and Management

Snowflake provides a fully managed infrastructure, eliminating the need for manual tuning, maintenance or complex configurations. Engineers can interact with Snowflake using standard SQL, making it easy to adopt. The platform also automates performance optimization, ensuring efficient query

execution and storage management without requiring dedicated database administrators.

4. Cost Efficency

Snowflake adopted the pay-as-you-go plan, which is mean that you only pay for the compute and storage ressources used, avoiding unecessary costs. Additionally, data compression and deduplication minimize storage expenses, further improving cost efficiency compared to traditional data warehouses.

How to get started

Requirements :

- Python 3.10 or higher
- Makefile

⚠ *Note: This setup uses a Makefile and works best on Unix/Linux systems.*

1. Clone the repository

```
git clone git@github.com:lei-rth/IST-Project-Snowflake.git
cd IST-Project-Snowflake
```

2. Clone the OpenSky API repository for getting the planes data

```
git clone git@github.com:openskynetwork/opensky-api.git
```

3. Create a stage on Snowflake. setup environment variables using the command bellow and fill it with your credentials.

```
make dotenv
```

4. Create a python virtual environment

```
make venv
source venv/bin/activate
```

5. Install dependencies

```
make install
```

6. Try to download data.

```
make download
```

7. Verify data in `data/` folder. The data are contained into two files : `flight_data.csv` and `flights_from_zurich.csv`
8. Upload data to Snowflake (It should try to redownload the data before uploading)

```
make upload
```

9. Verify on Snowflake that the data are in the `flight_data` and `flights_from_zurich` tables
10. Configure the Snowflake integration with the S3 bucket - Follow the official documentation
11. Unload data to S3 directly from the interface with

```
COPY INTO @my_s3_stage
FROM your_table_name
FILE_FORMAT = (TYPE = CSV FIELD_OPTIONALLY_ENCLOSED_BY = '"' COMPRES
HEADER = TRUE
OVERWRITE = TRUE;
```

12. Verify that the data are in your S3 bucket

How to use it with the most important commands/operations

Basic SQL Operations in Snowflake

Create a table:

```
CREATE TABLE flight_data (
    icao24 STRING,
    callsign STRING,
    origin_country STRING,
    ...
);
```

Insert data into a table: it is usually done via `COPY INTO` (better for performance) command or Python `write_pandas`

Query data:

```
SELECT * FROM flight_data;
```

Find flights over Switzerland:

```
SELECT *
FROM flight_data
WHERE latitude BETWEEN 45.8179 AND 47.8085
  AND longitude BETWEEN 5.9559 AND 10.4920;
```

List files in a stage:

```
LIST @my_stage;
```

Data loading and unloading

Upload (PUT) a local file to Stage:

```
PUT file://path/to/file.csv @your_stage;
```

Copy data into a table:

```
COPY INTO your_table
FROM @your_stage/file.csv
FILE_FORMAT = (TYPE = 'CSV' FIELD_OPTIONALLY_ENCLOSED_BY = '"')
SKIP_HEADER = 1;
```

Unload data to S3:

```
COPY INTO @my_s3_stage
FROM your_table
FILE_FORMAT = (TYPE = CSV FIELD_OPTIONALLY_ENCLOSED_BY = '"' COMPRESSION =
HEADER = TRUE
OVERWRITE = TRUE;
```

Resources

- Snowflake website: https://www.snowflake.com
- Snowflake Documentation: https://docs.snowflake.com/
- Python Connector Documentation: https://docs.snowflake.com/en/developer-guide/python-connector/python-connector
- OpenSky API Documentation: https://openskynetwork.github.io/opensky-api/