EXTERNAL LINK ACCESS FROM SNOWFLAKE

Atanu Das
Atanu.das@gds.ey.com

Table of Contents

1. Introduction:

Snowpark, with its libraries and runtimes, empowers data engineers, data scientists, and developers to create robust data engineering pipelines, ML workflows, and data applications in Python, Java, and Scala. In certain scenarios, accessing external network locations becomes essential, prompting the introduction of External Access. This feature, currently in public preview on Amazon Web Services (AWS) regions, facilitates seamless and secure connectivity to external endpoints from Snowpark code.

2. Benefits of External Access:

External Access eliminates the need for additional infrastructure setup when compared to External Functions. Users can connect to external network locations directly from Snowpark code, maintaining high security and governance over their data. Unlike External Functions, External Access provides flexibility in defining data transfer parameters such as batch size, retry behaviour, and exception handling.

3. Use Cases:

Generative AI and LLM Services: Accessing leading LLMs, such as GPT-4, for text, image, audio, or code generation.

Data Enrichment: Utilizing external APIs, like Google Maps, for location data to optimize supply chain routes.

Ingesting Data from Various Systems: Ingesting data from sources like Twitter, Google Sheets, MySQL, or other public internet data sources.

ML Tooling: Connecting to external endpoints for ML artifacts or leveraging external ML Flow servers with Snowpark.

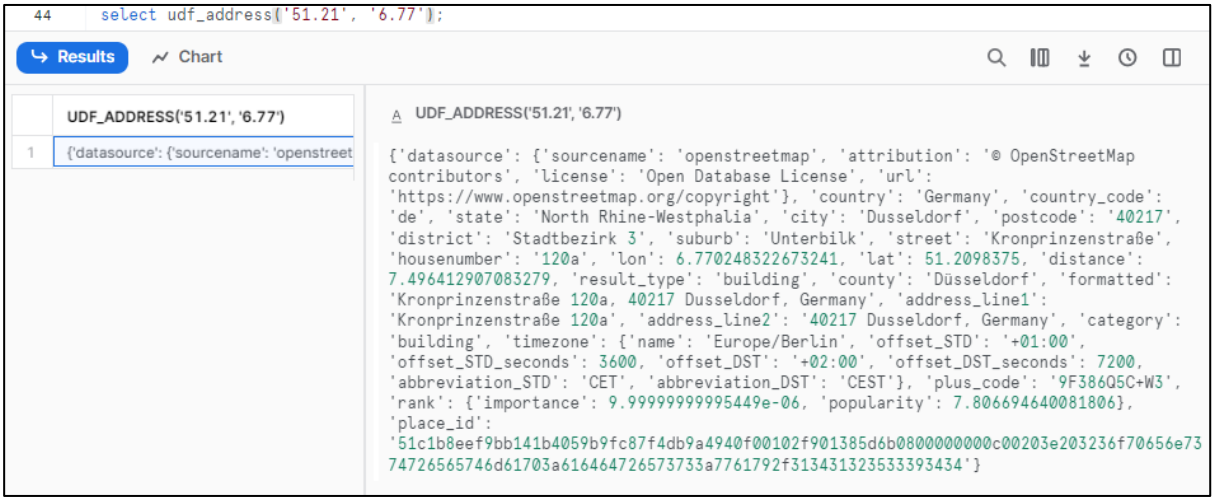
Reverse ETL: Copying data from Snowflake to operational systems and SaaS tools, enhancing customer experiences.

Partner Integrations: Partners can leverage External Access to seamlessly connect to solutions outside of Snowflake, ensuring scalability and secure end-to-end solutions.

Maintaining Security and Governance: External Access allows user code in the Snowpark sandbox to connect to external endpoints while maintaining high security and governance.

4. Connecting to External Network Locations:

Here we will use an address provider API to fetch the address details based on input coordinates.



Steps to connect to external API from a Python UDF:

1. **Create a network rule to define external network locations:**

```
create or replace network rule geo_network_rule
mode = egress type = host_port value_list = ('api.geoapify.com');
```

Note: geoapify.com is the API provider for our use case.

2. **Create a secret to store credentials:**

```
create or replace secret geo_api_key type = generic_string
secret_string = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';
```

3. **Create an external access integration object:**

```
create or replace external access integration geo_integration
allowed_network_rules = (geo_network_rule)
allowed_authentication_secrets = (geo_api_key)
enabled = true;
```

4. **Grant usage privilege on the integration and READ privilege on the secret to the UDF required role:**

```
grant usage on integration geo_integration to role sysadmin;
grant read on secret geo_api_key to role sysadmin;
```

5. **Create the UDF to call the API:**

```
Create or replace function udf_address(lattitude string, longitude
string)
returns string
language python
runtime_version = 3.9
handler = 'get_address'
external_access_integrations = (geo_integration)
packages = ('requests')
secrets = ('cred' = geo_api_key)
as
$$
import _snowflake
import requests
from requests.structures import CaseInsensitiveDict

token = _snowflake.get_generic_secret_string('cred')

def get_address(lattitude,longitude):

    url = "https://api.geoapify.com/v1/geocode/reverse?lat=" +
lattitude + "&lon=" + longitude + "&apiKey="+ token
    headers = CaseInsensitiveDict()
    headers["Accept"] = "application/json"

    resp = requests.get(url, headers=headers)
    api_response = resp.json()
    properties = api_response['features'][0]['properties']
    return properties
$$;
```

6. **Use the UDF directly in select statement or call the procedure:**

```
select udf_address('51.21', '6.77');
```

5. Conclusion:

External Access in Snowpark opens numerous possibilities for developers on the Snowflake Data Cloud. With a focus on security, flexibility, and scalability, this feature is pivotal in enhancing data integration workflows within the Snowflake environment.

6. References:

1. External network access examples | Snowflake Documentation
2. Maps, APIs and components | Geoapify Location Platform