Snowpark Performance Scaling Test (Azure + Snowflake)

Introduction

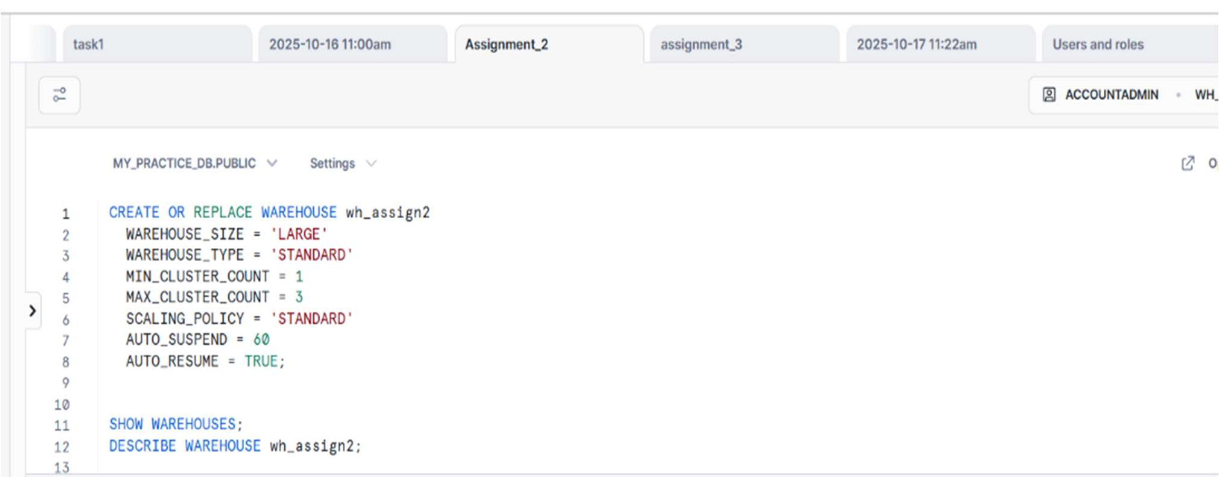
In this performance-based assignment, the objective was to evaluate how **scaling Snowflake Virtual Warehouses** affects query execution time when processing large datasets using **Snowpark for Python** in **Azure**.

The test was carried out in a controlled environment using the database **MY_PRACTICE_DB**, schema **PUBLIC**, and a performance warehouse **WH_ASSIGN2**. The process covered warehouse configuration, dataset generation, Snowpark connection from Azure, performance testing, and monitoring through query history.

Step 1 — Create and Configure Virtual Warehouse

A dedicated warehouse named **WH_ASSIGN2** was created exclusively for the Snowpark performance test.

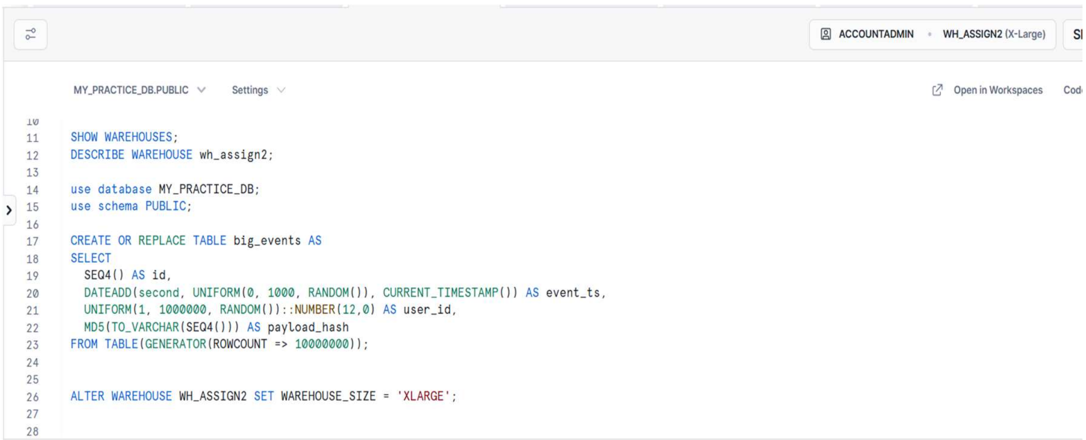
- The warehouse was configured as **STANDARD type** with **auto-suspend (60 seconds)** and **auto-resume enabled**.
- **Multi-cluster scaling** was set with a minimum of 1 and a maximum of 3 clusters to allow elasticity during heavy processing.
- After creation, the configuration was verified using *SHOW WAREHOUSES* and *DESCRIBE WAREHOUSE WH_ASSIGN2*.
- The database context was switched to **MY_PRACTICE_DB** and schema to **PUBLIC** for further operations.



Step 2 — Create Database, Schema, and Large Sample Table

To simulate a real-time data environment, a large event dataset was generated.

- A new table named **BIG_EVENTS** was created under **MY_PRACTICE_DB.PUBLIC**.
- This table was populated using Snowflake's internal generator function, producing **10 million records** with random event timestamps, user IDs, and unique payload hash values.
- The warehouse **WH_ASSIGN2** was later scaled up to **XLARGE** size to efficiently handle large data volume operations.
- Verification confirmed that the data was successfully generated and stored in the table.




```sql
SHOW WAREHOUSES;
DESCRIBE WAREHOUSE wh_assign2;

use database MY_PRACTICE_DB;
use schema PUBLIC;

CREATE OR REPLACE TABLE big_events AS
SELECT
  SEQ4() AS id,
  DATEADD(second, UNIFORM(0, 1000, RANDOM()), CURRENT_TIMESTAMP()) AS event_ts,
  UNIFORM(1, 1000000, RANDOM())::NUMBER(12,0) AS user_id,
  MD5(TO_VARCHAR(SEQ4())) AS payload_hash
FROM TABLE(GENERATOR(ROWCOUNT => 10000000));


ALTER WAREHOUSE WH_ASSIGN2 SET WAREHOUSE_SIZE = 'XLARGE';
```

Step 3 — Connect from Azure using Snowpark and Execute Aggregation Job

Using **Snowpark for Python**, a connection was established from **Azure** to **Snowflake**.

- The Snowpark session was configured with account credentials, warehouse (**WH_ASSIGN2**), and the **MY_PRACTICE_DB.PUBLIC** schema.
- A Snowpark DataFrame was created to read data from the **BIG_EVENTS** table.
- An aggregation operation was performed to count the number of events per user and store the output into a new table named **BIG_EVENTS_AGG**.
- The total execution time was recorded to measure job performance.
- Initial runtime on a smaller warehouse size served as a baseline for comparison.

Untitled Notebook 2025-10-22 16:49:20 | Untitled Notebook 2025-10-22 16:00:02 | Assignment_2

File Edit View Run Help Python Tabs: ON Last edit was 7 days ago Run all Terminated Schedule Share

Oct 16, 2025 (18s) 1 Python

```
%pip install "snowflake-snowpark-python[pandas]"
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.6/4.6 MB 59.1 MB/s eta 0:00:00
Downloading sortedcontainers-2.4.0-py2.py3-none-any.whl (29 kB)
Downloading tomlkit-0.13.3-py3-none-any.whl (38 kB)
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/347.8 kB ? eta -:--:--
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 347.8/347.8 kB 21.3 MB/s eta 0:00:00
Installing collected packages: sortedcontainers, asn1crypto, tzlocal, tzdata, tomlkit, pandas, cryptography, pyOpenSSL, snowflake-connector-python,
snowflake-snowpark-python
  Attempting uninstall: pandas
    Found existing installation: pandas 1.5.3
    Not uninstalling pandas at /databricks/python3/lib/python3.12/site-packages, outside environment /local_disk0/.ephemeral_nfs/envs/pythonEnv-2dc
704bb-c99f-4b45-8f9f-681ac65c17ab
    Can't uninstall 'pandas'. No files were found to uninstall.
  Attempting uninstall: cryptography
    Found existing installation: cryptography 42.0.5
    Not uninstalling cryptography at /databricks/python3/lib/python3.12/site-packages, outside environment /local_disk0/.ephemeral_nfs/envs/pythonE
nv-2dc704bb-c99f-4b45-8f9f-681ac65c17ab
    Can't uninstall 'cryptography'. No files were found to uninstall.
Successfully installed asn1crypto-1.5.1 cryptography-46.0.0 pandas-2.3.3 pyOpenSSL-25.3.0 snowflake-connector-python-3.18.0 snowflake-snowpark-pyth
on-1.40.0 sortedcontainers-2.4.0 tomlkit-0.13.3 tzdata-2025.2 tzlocal-5.3.1
Note: you may need to restart the kernel using %restart_python or dbutils.library.restartPython() to use updated packages.
```

Oct 16, 2025 (1s) 3 Python

```python
from snowflake.snowpark import Session
import time

# --- Snowflake connection configuration ---
connection_parameters = {
    "account":"tprtvog-tg33465",
    "user": "shivashankari",
    "password": "Shivashankari_04",
    "role": "ACCOUNTADMIN",                   # or another valid role
    "warehouse": "wh_assign2",              # your test warehouse
    "database": "MY_PRACTICE_DB",
    "schema": "PUBLIC"
}

# --- Create a Snowpark session ---
session = Session.builder.configs(connection_parameters).create()
print("Connected to Snowflake!")
```

```
Connected to Snowflake!
```

Oct 16, 2025 (1s) 4 Python

```python
from snowflake.snowpark.functions import col, count

start = time.time()

# Load the big table
df = session.table("BIG_EVENTS")

# Group by USER_ID and count events
agg_df = df.group_by(col("USER_ID")).agg(count("*").alias("EVENT_COUNT"))

# Write the results into a new table
agg_df.write.save_as_table("BIG_EVENTS_AGG", mode="overwrite")

end = time.time()
print(f"Snowpark job completed in {end - start:.2f} seconds")
```

```
Snowpark job completed in 0.78 seconds
```

```
Oct 16, 2025 (1s)                6                Python

start = time.time()

df = session.table("BIG_EVENTS")
agg = df.group_by(col("USER_ID")).agg(count("*").alias("EVENT_COUNT"))
agg.write.save_as_table("BIG_EVENTS_AGG", mode="overwrite")

end = time.time()
print(f"Job finished in {end - start:.2f} seconds")

session.table("BIG_EVENTS_AGG").show(5)
```

```
Job finished in 0.64 seconds
----------------------------
|"USER_ID"  |"EVENT_COUNT"  |
----------------------------
|368522     |1              |
|332204     |1              |
|811189     |2              |
|404799     |2              |
|504308     |3              |
----------------------------
```

Step 4 — Scale Warehouse and Re-run Aggregation

To observe the performance impact, the same Snowpark job was re-executed after scaling the warehouse.

- The **WH_ASSIGN2** warehouse was resized to **LARGE** and then resumed for computation.
- The same aggregation logic was executed again using the same dataset and environment.
- The runtime was measured and compared across different warehouse sizes.

WH_ASSIGN2

Standard warehouse   ACCOUNTADMIN   6 days ago

Details

Type Standard	Status Suspended	Running 0 queries
Queued 0 queries	Size X-Large	Max Clusters 3
Min Clusters 1	Scaling Policy STANDARD	Auto Suspend 60 seconds
Auto Resume Enabled	Resumed On 6 days ago	Query Acceleration Disabled
Resource Constraint STANDARD_GEN_1		

Step 5 — Monitor Query Performance using Snowflake Query History

The **Snowflake Query History** feature was used to analyze and confirm performance changes.

- Queries containing BIG_EVENTS were filtered to review runtime, warehouse name, and execution start time.
- The metrics verified that queries executed under **larger warehouse sizes** completed much faster than smaller configurations.
- The improvement in performance directly reflected the benefits of Snowflake's **scaling architecture**.

Before Scaling:

	SQL TEXT	QUERY ID	STATUS	USER	WAREHOUSE	DURATION	STARTED
29	SELECT query_id, warehouse_name, total_el	01bfc077-0001-66bb-000c-5b020002453a	Success	—	WH_ASSIGN2	312ms	10/16/2025, 9:53:24 I
30	SELECT * FROM BIG_EVENTS_AGG LIMIT 5	01bfc076-0001-6698-000c-5b020002542a	Success	—	WH_ASSIGN2	135ms	10/16/2025, 9:52:55 I
31	CREATE OR REPLACE TABLE BIG_EVENTS_AGG("US	01bfc076-0001-66bb-000c-5b0200024532	Success	—	WH_ASSIGN2	601ms	10/16/2025, 9:52:54 I
32	SELECT * FROM BIG_EVENTS_AGG LIMIT 5	01bfc076-0001-66bb-000c-5b020002452e	Success	—	WH_ASSIGN2	205ms	10/16/2025, 9:52:51 I
33	CREATE OR REPLACE TABLE BIG_EVENTS_AGG("US	01bfc076-0001-6698-000c-5b020002541e	Success	—	WH_ASSIGN2	729ms	10/16/2025, 9:52:30 I
34	SELECT * FROM BIG_EVENTS_AGG LIMIT 5	01bfc076-0001-66bb-000c-5b0200024522	Success	—	WH_ASSIGN2	120ms	10/16/2025, 9:52:17 I
35	CREATE OR REPLACE TABLE BIG_EVENTS_AGG("US	01bfc076-0001-6698-000c-5b020002541a	Success	—	WH_ASSIGN2	776ms	10/16/2025, 9:52:16 I
36	SELECT * FROM BIG_EVENTS_AGG LIMIT 5	01bfc075-0001-6698-000c-5b0200025412	Success	—	WH_ASSIGN2	304ms	10/16/2025, 9:51:45 I

After Scaling:

40	SELECT * FROM BIG_EVENTS_AGG LIMIT 5	01bfc072-0001-6698-000c-5b02000253d6	Success	—	WH_ASSIGN2	167ms	10/16/2025, 9:48:05 I
41	CREATE OR REPLACE TABLE BIG_EVENTS_AGG("US	01bfc071-0001-6698-000c-5b02000253d2	Success	—	WH_ASSIGN2	3.4s	10/16/2025, 9:47:56 I
42	CREATE OR REPLACE TABLE big_events AS SELE	01bfc05f-0001-66bb-000c-5b020002448a	Success	—	WH_ASSIGN2	10s	10/16/2025, 9:29:35 I
43	CREATE OR REPLACE TABLE big_events AS SELE	01bfc05d-0001-66bb-000c-5b020002447e	Failed	—	WH_ASSIGN2	57ms	10/16/2025, 9:27:53 I
44	use schema PUBLIC;	01bfc05d-0001-66bb-000c-5b0200024476	Success	—	WH_ASSIGN2	28ms	10/16/2025, 9:27:47 I
45	use database MY_PRACTICE_DB;	01bfc05d-0001-66bb-000c-5b020002446a	Success	—	WH_ASSIGN2	36ms	10/16/2025, 9:27:32 I

Conclusion

The experiment successfully demonstrated how **warehouse scaling** in Snowflake enhances performance when handling large datasets with Snowpark.

Through this case study, we:

- Created and configured **WH_ASSIGN2** warehouse for testing.
- Generated a large dataset (BIG_EVENTS) under **MY_PRACTICE_DB.PUBLIC**.
- Connected from **Azure** using **Snowpark for Python**.
- Executed and re-ran aggregation jobs at different warehouse sizes.
- Verified faster execution through **query history analysis**.

This exercise confirmed that increasing warehouse size in Snowflake provides significant performance improvement for large-scale Snowpark data transformations executed from Azure.