

# Snowflake Artifact Extraction Prerequisites

# Contents

- 1. Introduction .....3
- 2. Artifact Extraction.....3
  - 2.1 Bulk Export – DDL Scripts .....3
  - 2.2 Bulk Export – Query Execution Logs.....3
    - 2.2.1 Bulk Export – Using COPY INTO Command .....3
    - 2.2.2 Bulk Export – Using Snowflake UI (Snowsight or Classic UI) .....6
  - 2.1 Bulk Export – Warehouse Artifacts .....8
  - 2.2 Other Scripts/Artifacts .....9
- 3. Getting Help.....10

## 1. Introduction

LeapLogic's Assessment profiles existing inventory, identifies complexity, performs dependency analysis, and provides recommendations for migration to modern data platform.

## 2. Artifact Extraction

LeapLogic requires certain artifacts to perform an assessment. As a prerequisite, you need to have an ACCOUNTADMIN privilege to get data. You can copy them from your GIT instance or the Snowflake repository where all artifacts are stored. LeapLogic needs the following artifacts in the form of files such as .sql.

- DDL scripts
- Snowflake Stored Procedures
- Functions
- Query execution logs
- DML scripts
- Snowpark scripts
- Snowflake Scripting files

### 2.1 Bulk Export – DDL Scripts

To export the required DDLs, refer to [this](#) script. Change the following parameters as applicable.

- Provide a comma-separated list of database names to extract the required DDLs. Keep it empty to extract DDLs from all the databases.
- Provide a path where you want to store the exported DDLs.
- Provide the Snowflake connection details to establish a connection to the environment.

This script exports all the database objects such as databases, external tables, file formats, schemas, sequences, stored procedures, tables, tags, tasks, UDFs, external functions, views etc.

### 2.2 Bulk Export – Query Execution Logs

There are two options available for extracting query execution logs. The first option involves utilizing the COPY INTO command, while the second option involves using the Snowflake UI (Snowsight or Classic UI). It is recommended to use the approach of COPY INTO command, especially if the query log size is substantial.

#### 2.2.1 Bulk Export – Using COPY INTO Command

To utilize the COPY INTO command, it is necessary to create a named stage to store the output of the COPY command.

Refer to the following command to create a stage in your database.

```
CREATE OR REPLACE STAGE database_name.schema_name.querylog_stage;
```

where:

- database\_name –represents the name of the database where the stage object is to be created.
- schema\_name – represents the name of the schema where the stage object is to be created.

To export the necessary query execution logs, you can use the following COPY INTO command as a reference. Change the start date `au.START_TIME <= current_date - 30` as required. The query's value of `-30` will retrieve records from the past 30 days.

```
COPY INTO @querylog_stage
FROM (SELECT
      'IDWWM' || '~~' ||
au.USER_NAME || '~~' ||
s.CLIENT_APPLICATION_ID || '~~' ||
'client_u' || '~~' ||
START_TIME || '~~' ||
(QUEUED_PROVISIONING_TIME + QUEUED_REPAIR_TIME + QUEUED_OVERLOAD_TIME)::varchar
|| '~~' ||
(BYTES_SCANNED + BYTES_WRITTEN + BYTES_WRITTEN_TO_RESULT +
BYTES_READ_FROM_RESULT)::varchar || '~~' ||
COMPILATION_TIME::varchar || '~~' ||
START_TIME || '~~' ||
START_TIME || '~~' ||
0::varchar || '~~' ||
au.QUERY_ID || '~~' ||
(max(QUEUED_PROVISIONING_TIME + QUEUED_REPAIR_TIME + QUEUED_OVERLOAD_TIME)
OVER (
  ORDER BY NULL))::varchar || '~~' ||
(max(BYTES_SCANNED + BYTES_WRITTEN + BYTES_WRITTEN_TO_RESULT +
BYTES_READ_FROM_RESULT) OVER (
  ORDER BY NULL))::varchar || '~~' ||
(sum(QUEUED_PROVISIONING_TIME) OVER (
  ORDER BY NULL) + sum(QUEUED_REPAIR_TIME) OVER (
  ORDER BY NULL)
+ sum(QUEUED_OVERLOAD_TIME) OVER (
  ORDER BY NULL))::varchar || '~~' ||
(sum(BYTES_SCANNED) OVER (
  ORDER BY NULL) + sum(BYTES_WRITTEN) OVER (
  ORDER BY NULL) +
sum(BYTES_WRITTEN_TO_RESULT) OVER (
  ORDER BY NULL) + sum(BYTES_READ_FROM_RESULT) OVER (
  ORDER BY NULL))::varchar || '~~' ||
(TOTAL_ELAPSED_TIME / 1000)::varchar || '~~' ||
COALESCE(DATABASE_NAME,
  'Not Present') || '~~' ||
0::varchar || '~~' ||
QUERY_TYPE || '~~' ||
l.REPORTED_CLIENT_TYPE || '~~' ||
' ' || '~~' ||
QUERY_TEXT || '~~' ||
au.execution_time/(1000*60*60)*wh.credits_per_hour || '~~' ||
au.WAREHOUSE_SIZE || '~~' ||
au.WAREHOUSE_TYPE || '~~' ||
nvl(au.warehouse_name, 'NA') || '~~' ||
nvl(au.warehouse_id, 0)
FROM
```

```

        SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY AS AU
JOIN snowflake.account_usage.sessions s ON
        AU.session_id = s.session_id
JOIN snowflake.account_usage.login_history l ON
        l.event_id = s.login_event_id
JOIN (
        SELECT 'X-Small' AS warehouse_size, 1 AS credits_per_hour UNION ALL
        SELECT 'Small' AS warehouse_size, 2 AS credits_per_hour UNION ALL
        SELECT 'Medium' AS warehouse_size, 4 AS credits_per_hour UNION ALL
        SELECT 'Large' AS warehouse_size, 8 AS credits_per_hour UNION ALL
        SELECT 'X-Large' AS warehouse_size, 16 AS credits_per_hour UNION ALL
        SELECT '2X-Large' AS warehouse_size, 32 AS credits_per_hour UNION ALL
        SELECT '3X-Large' AS warehouse_size, 64 AS credits_per_hour UNION ALL
        SELECT '4X-Large' AS warehouse_size, 128 AS credits_per_hour
) AS wh
        ON AU.warehouse_size=wh.warehouse_size
WHERE
        au.START_TIME between '2023-01-01' AND '2023-12-29')
FILE_FORMAT = ( TYPE = CSV)
OVERWRITE = TRUE
max file size=4900000000;

```

Keep in mind that the COPY INTO command operates in parallel, which means it will generate multiple files in the stage.

### 1. Download Query Log Files from Named Stage

There are two options available for downloading query execution log files from the name stage. The first option to use Snowflake UI (Snowsight), while the second option involves using the Snowflake SnowSQL CLI (command Line Interface).

### 2. Download Query Log Files using Snowsight

Follow the steps below to download query log files from the named stage using the Snowflake Snowsight UI:

- a) Go to the 'Data' section in the Snowsight left panel.
- b) Choose 'Databases' under the Data section.
- c) Search for the Database and Schema that were used to create the Stage object.
- d) Select 'Stages' under the chosen Schema.
- e) Click on the stage name, e.g., QUERYLOG\_STAGE.
- f) (Optional) If prompted, 'Enable Directory Table.'
- g) (Optional) Select the warehouse name from the dropdown next to the search option if not already selected.
- h) Click on the three dots (...) on the file row and choose the download option.

### 3. Download Query Log Files using snowsql CLI.

To download files from the named stage using snowsql, it is necessary to install the SnowSQL CLI on your system.

Proceed with the following steps to download query log files from the named stage using the Snowflake snowsql CLI.

a) Access the Snowflake by Logging in using Snowsql.

```
snowsql -a account.us-east-1 -u username -d database -s schema -w warehouse  
-r role;
```

Here is the key information for the parameters:

- -a: Snowflake account name
- -u: Snowflake username
- -d: Snowflake database used to create the stage object
- -s: Snowflake Schema used to create the stage object
- -w: Warehouse name to use
- -r: Role used to create the stage object.

b) Verify the content of stage object.

```
list @QUERYLOG_STAGE;
```

c) Use the GET command to download the query log files.

Windows:

```
GET @QUERYLOG_STAGE file://C:\path\tmp\data;
```

The command will download all files in the stage for the QUERYLOG\_STAGE table to the C:\path\tmp\data local windows directory.

Linux:

```
GET @QUERYLOG_STAGE file:///path/tmp/data/;
```

The command will download all files in the stage for the QUERYLOG\_STAGE table to the /path/tmp/data/ local Linux directory.

## 2.2.2 Bulk Export – Using Snowflake UI (Snowsight or Classic UI)

To export the required query execution logs, refer to the below script. Change the start date `au.START_TIME <= current_date - 30` as required. The query's value of **-30** will retrieve records from the past 30 days.

The recommended time frame is six months. Next, execute the given script below on the Snowflake UI console and download the output after execution.

```
SELECT  
    'IDWWM' || '~~' ||  
au.USER_NAME || '~~' ||  
s.CLIENT_APPLICATION_ID || '~~' ||  
'client_u' || '~~' ||  
START TIME || '~~' ||
```

```

(QUEUED_PROVISIONING_TIME + QUEUED_REPAIR_TIME + QUEUED_OVERLOAD_TIME)::varchar
|| '~~' ||
(BYTES_SCANNED + BYTES_WRITTEN + BYTES_WRITTEN_TO_RESULT +
BYTES_READ_FROM_RESULT)::varchar || '~~' ||
COMPILATION_TIME::varchar || '~~' ||
START_TIME || '~~' ||
START_TIME || '~~' ||
0::varchar || '~~' ||
au.QUERY_ID || '~~' ||
(max(QUEUED_PROVISIONING_TIME + QUEUED_REPAIR_TIME + QUEUED_OVERLOAD_TIME)
OVER(
ORDER BY NULL))::varchar || '~~' ||
(max(BYTES_SCANNED + BYTES_WRITTEN + BYTES_WRITTEN_TO_RESULT +
BYTES_READ_FROM_RESULT) OVER(
ORDER BY NULL))::varchar || '~~' ||
(sum(QUEUED_PROVISIONING_TIME) OVER (
ORDER BY NULL) + sum(QUEUED_REPAIR_TIME)OVER (
ORDER BY NULL)
+ sum(QUEUED_OVERLOAD_TIME) OVER(
ORDER BY NULL))::varchar || '~~' ||
(sum(BYTES_SCANNED) OVER(
ORDER BY NULL) + sum(BYTES_WRITTEN) OVER (
ORDER BY NULL) +
sum(BYTES_WRITTEN_TO_RESULT) OVER(
ORDER BY NULL) + sum(BYTES_READ_FROM_RESULT) OVER(
ORDER BY NULL))::varchar || '~~' ||
(TOTAL_ELAPSED_TIME / 1000)::varchar || '~~' ||
COALESCE(DATABASE_NAME,
'Not Present') || '~~' ||
0::varchar || '~~' ||
QUERY_TYPE || '~~' ||
1.REPORTED_CLIENT_TYPE || '~~' ||
' ' || '~~' ||
QUERY_TEXT || '~~' ||
au.execution_time/(1000*60*60)*wh.credits_per_hour || '~~' ||
au.WAREHOUSE_SIZE || '~~' ||
au.WAREHOUSE_TYPE || '~~' ||
nvl(au.warehouse_name, 'NA') || '~~' ||
nvl(au.warehouse_id, 0)
FROM
SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY AS AU
JOIN snowflake.account_usage.sessions s ON
AU.session_id = s.session_id
JOIN snowflake.account_usage.login_history l ON
l.event_id = s.login_event_id
JOIN (
SELECT 'X-Small' AS warehouse_size, 1 AS credits_per_hour UNION ALL
SELECT 'Small' AS warehouse_size, 2 AS credits_per_hour UNION ALL
SELECT 'Medium' AS warehouse_size, 4 AS credits_per_hour UNION ALL
SELECT 'Large' AS warehouse_size, 8 AS credits_per_hour UNION ALL
SELECT 'X-Large' AS warehouse_size, 16 AS credits_per_hour UNION ALL
SELECT '2X-Large' AS warehouse_size, 32 AS credits_per_hour UNION ALL
SELECT '3X-Large' AS warehouse_size, 64 AS credits_per_hour UNION ALL
SELECT '4X-Large' AS warehouse_size, 128 AS credits_per_hour
) AS wh
ON AU.warehouse_size=wh.warehouse_size
WHERE
au.START_TIME between '2023-01-01' AND '2023-12-29'
;

```

## 2.1 Bulk Export – Warehouse Artifacts

To export the query results as separated delimited files, refer to the below script. Additional artifacts are needed for better assessment.

```
-- Warehouse with credit used till date
SELECT WAREHOUSE_NAME, sum(CREDITS_USED) AS TOTAL_CREDIT_USED
FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
GROUP BY 1
ORDER BY 2 DESC;

-- Warehouse load
SELECT WAREHOUSE_NAME, sum(AVG_RUNNING+AVG_QUEUED_LOAD) Avg_warehouse_load
FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_LOAD_HISTORY
GROUP BY 1
ORDER BY 2 DESC;

-- Warehouse Usage Month over month
SELECT WAREHOUSE_NAME, to_char(START_TIME, 'MON-YYYY') AS "MONTH",
sum(CREDITS_USED) AS TOTAL_CREDIT_USED
FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
GROUP BY 1, 2
ORDER BY 1,3 DESC;

-- 2.4 Database Objects
SELECT
    TABLE_CATALOG AS databasename,
    CASE
        WHEN IS_TRANSIENT = 'YES' THEN 'TRANSIENT'
        ELSE TABLE_TYPE
    END tablekind,
    COUNT(TABLE_NAME)
FROM
    SNOWFLAKE.ACCOUNT_USAGE."TABLES"
GROUP BY 1,2
ORDER BY 1;

-- 2.5 Database Volume
select TABLE_CATALOG AS databasename, cast (sum(ACTIVE_BYTES)/1024/1024/1024 as
decimal(18,2))
from SNOWFLAKE.ACCOUNT_USAGE.TABLE_STORAGE_METRICS
GROUP BY 1
HAVING cast (sum(ACTIVE_BYTES)/1024/1024/1024 as decimal(18,2)) > 10;

-- 2.6 High Data Volume Tables, last DML/COPY INTO tables
SELECT t1.TABLE_CATALOG AS databasename,
    t1.TABLE_NAME AS tablename,
    t1.row_count AS num_rows,
    cast (ACTIVE_BYTES/1024/1024/1024 as decimal(18,2)) AS
currentsize_gb,
    NULL AS ConstraintType,
    NULL AS ConstraintText,
    TO_CHAR(TO_TIMESTAMP(GET(GREATEST(ARRAY_CONSTRUCT(T1.LAST_ALTERED),
    ARRAY_CONSTRUCT(L1.LAST_LOAD_TIME)),0)), 'YYYY-MON-DD') AS
LAST_DML_COPY
FROM snowflake.account_usage."TABLES" AS t1
JOIN SNOWFLAKE.ACCOUNT_USAGE.TABLE_STORAGE_METRICS AS t2
ON (t1.TABLE_ID = t2.ID)
LEFT JOIN SNOWFLAKE.ACCOUNT_USAGE.LOAD_HISTORY L1
ON (T1.TABLE_ID = L1.TABLE_ID)
WHERE table_type = 'BASE TABLE'
```



```

AND T1.TABLE_CATALOG IN ('NEWTPCDS', 'TEST_DB') -- Provide DATABASE names
ORDER BY 3 DESC ;

-- 2.7      Databases and Users
SELECT DATABASE_NAME, count(DISTINCT USER_NAME)
FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
GROUP BY DATABASE_NAME;

-- 2.8 Warehouse details
SELECT
WAREHOUSE_NAME, to_char("TIMESTAMP", 'MON-YYYY') AS "MONTH",
SUM(CASE WHEN EVENT_NAME = 'SPINUP_CLUSTER' THEN 1 ELSE 0 END ) AS
NO_TIMES_SPINUP,
SUM(CASE WHEN EVENT_NAME = 'RESIZE_CLUSTER' THEN 1 ELSE 0 END ) AS
NO_TIMES_RESIZED,
SUM(CASE WHEN EVENT_NAME = 'RESUME_CLUSTER' THEN 1 ELSE 0 END ) AS
NO_TIMES_RESUMED
FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_EVENTS_HISTORY
WHERE EVENT_STATE LIKE '%COMPLETED'
GROUP BY 1,2
ORDER BY NO_TIMES_RESUMED DESC
;

```

## 2.2 Other Scripts/Artifacts

Copy any other scripts such as DML scripts, Snowpark scripts, SnowScript files etc. from your GIT instance or the Snowflake repository and provide these artifacts to LeapLogic Team.

### 3. Getting Help

Contact LeapLogic technical support at [info@leaplogic.io](mailto:info@leaplogic.io)