Lab 7: Resource Optimization: Billing Metrics

Overview

This resource optimization guide represents one module of the four contained in the series. These guides are meant to help customers better monitor and manage their credit consumption. Helping our customers build confidence that their credits are being used efficiently is key to an ongoing successful partnership. In addition to this set of Snowflake Quickstarts for Resource Optimization, Snowflake also offers community support as well as Training and Professional Services offerings.

Billing Metrics

Billing queries are responsible for identifying total costs associated with the high level functions of the Snowflake Cloud Data Platform, which includes warehouse compute, snowpipe compute, and storage costs. If costs are noticeably higher in one category versus the others, you may want to evaluate what might be causing that.

These metrics also seek to identify those queries that are consuming the most amount of credits. From there, each of these queries can be analyze for their importance (do they need to be run as frequently, if at all) and explore if additional controls need to be in place to prevent excessive consumption (i.e. resource monitors, statement timeouts, etc.).

What You'll Learn

- how to identify and analyze Snowflake consumption across all services
- how to analyze most resource-intensive queries
- how to analyze serverless consumption

What You'll Need

- A Snowflake Account
- Access to view [Account Usage Data Share]

Related Materials

- Resource Optimization: Setup & Configuration
- Resource Optimization: Usage Monitoring
- Resource Optimization: Performance

Query Tiers

Each query within the Resource Optimization Snowflake Quickstarts will have a tier designation just to the right of its name as "(T*)". The following tier descriptions should help to better understand those designations.

Tier 1 Queries

At its core, Tier 1 queries are essential to Resource Optimization at Snowflake and should be used by each customer to help with their consumption monitoring - regardless of size, industry, location, etc.

Tier 2 Queries

Tier 2 queries, while still playing a vital role in the process, offer an extra level of depth around Resource Optimization and while they may not be essential to all customers and their workloads, it can offer further explanation as to any additional areas in which over-consumption may be identified.

Tier 3 Queries

Finally, Tier 3 queries are designed to be used by customers that are looking to leave no stone unturned when it comes to optimizing their consumption of Snowflake. While these queries are still very helpful in this process, they are not as critical as the queries in Tier 1 & 2.

Billing Metrics (T1)

TIER 1

Description:

Identify key metrics as it pertains to total compute costs from warehouses, serverless features, and total storage costs.

How to Interpret Results:

Where are we seeing most of our costs coming from (compute, serverless, storage)? Are seeing excessive costs in any of those categories that are above expectations?

Primary Schema:

Account_Usage

SQL

```
/* These queries can be used to measure where costs have been incurred by
   the different cost vectors within a Snowflake account including:
   1) Warehouse Costs
   2) Serverless Costs
   3) Storage Costs

   To accurately report the dollar amounts, make changes to the variables
   defined on lines 17 to 20 to properly reflect your credit price, the initial
   capacity purchased, when your contract started and the term (default 12 months)

   If unsure, ask your Sales Engineer or Account Executive
*/

USE DATABASE SNOWFLAKE;
USE SCHEMA ACCOUNT_USAGE;

SET CREDIT_PRICE = 4.00; --edit this number to reflect credit price
SET TERM_LENGTH = 12; --integer value in months
SET TERM_START_DATE = '2019-01-01';
SET TERM_AMOUNT = 100000.00; --number(10,2) value in dollars


WITH CONTRACT_VALUES AS (

      SELECT
                $CREDIT_PRICE::decimal(10,2) as CREDIT_PRICE
               ,$TERM_AMOUNT::decimal(38,0) as TOTAL_CONTRACT_VALUE
               ,$TERM_START_DATE::timestamp as CONTRACT_START_DATE
               ,DATEADD(month,$TERM_LENGTH,$TERM_START_DATE)::timestamp as
```

```
CONTRACT_END_DATE

),
PROJECTED_USAGE AS (

      SELECT
                  CREDIT_PRICE
                 ,TOTAL_CONTRACT_VALUE
                 ,CONTRACT_START_DATE
                 ,CONTRACT_END_DATE
                 ,(TOTAL_CONTRACT_VALUE)
                     /
                     DATEDIFF(day,CONTRACT_START_DATE,CONTRACT_END_DATE)  AS
DOLLARS_PER_DAY
                 , (TOTAL_CONTRACT_VALUE/CREDIT_PRICE)
                     /
                 DATEDIFF(day,CONTRACT_START_DATE,CONTRACT_END_DATE) AS CREDITS_PER_DAY
      FROM       CONTRACT_VALUES

)

--COMPUTE FROM WAREHOUSES
SELECT
          'WH Compute' as WAREHOUSE_GROUP_NAME
         ,WMH.WAREHOUSE_NAME
         ,NULL AS GROUP_CONTACT
         ,NULL AS GROUP_COST_CENTER
         ,NULL AS GROUP_COMMENT
         ,WMH.START_TIME
         ,WMH.END_TIME
         ,WMH.CREDITS_USED
         ,$CREDIT_PRICE
         ,($CREDIT_PRICE*WMH.CREDITS_USED) AS DOLLARS_USED
         ,'ACTUAL COMPUTE' AS MEASURE_TYPE
from     SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY WMH

UNION ALL

--COMPUTE FROM SNOWPIPE
SELECT
          'Snowpipe' AS WAREHOUSE_GROUP_NAME
         ,PUH.PIPE_NAME AS WAREHOUSE_NAME
         ,NULL AS GROUP_CONTACT
         ,NULL AS GROUP_COST_CENTER
         ,NULL AS GROUP_COMMENT
         ,PUH.START_TIME
         ,PUH.END_TIME
         ,PUH.CREDITS_USED
         ,$CREDIT_PRICE
         ,($CREDIT_PRICE*PUH.CREDITS_USED) AS DOLLARS_USED
         ,'ACTUAL COMPUTE' AS MEASURE_TYPE
from     SNOWFLAKE.ACCOUNT_USAGE.PIPE_USAGE_HISTORY PUH
```

```
UNION ALL

--COMPUTE FROM CLUSTERING
SELECT
         'Auto Clustering' AS WAREHOUSE_GROUP_NAME
        ,DATABASE_NAME || '.' || SCHEMA_NAME || '.' || TABLE_NAME AS WAREHOUSE_NAME
        ,NULL AS GROUP_CONTACT
        ,NULL AS GROUP_COST_CENTER
        ,NULL AS GROUP_COMMENT
        ,ACH.START_TIME
        ,ACH.END_TIME
        ,ACH.CREDITS_USED
        ,$CREDIT_PRICE
        ,($CREDIT_PRICE*ACH.CREDITS_USED) AS DOLLARS_USED
        ,'ACTUAL COMPUTE' AS MEASURE_TYPE
from    SNOWFLAKE.ACCOUNT_USAGE.AUTOMATIC_CLUSTERING_HISTORY ACH

UNION ALL

--COMPUTE FROM MATERIALIZED VIEWS
SELECT
         'Materialized Views' AS WAREHOUSE_GROUP_NAME
        ,DATABASE_NAME || '.' || SCHEMA_NAME || '.' || TABLE_NAME AS WAREHOUSE_NAME
        ,NULL AS GROUP_CONTACT
        ,NULL AS GROUP_COST_CENTER
        ,NULL AS GROUP_COMMENT
        ,MVH.START_TIME
        ,MVH.END_TIME
        ,MVH.CREDITS_USED
        ,$CREDIT_PRICE
        ,($CREDIT_PRICE*MVH.CREDITS_USED) AS DOLLARS_USED
        ,'ACTUAL COMPUTE' AS MEASURE_TYPE
from    SNOWFLAKE.ACCOUNT_USAGE.MATERIALIZED_VIEW_REFRESH_HISTORY MVH

UNION ALL

--COMPUTE FROM SEARCH OPTIMIZATION
SELECT
         'Search Optimization' AS WAREHOUSE_GROUP_NAME
        ,DATABASE_NAME || '.' || SCHEMA_NAME || '.' || TABLE_NAME AS WAREHOUSE_NAME
        ,NULL AS GROUP_CONTACT
        ,NULL AS GROUP_COST_CENTER
        ,NULL AS GROUP_COMMENT
        ,SOH.START_TIME
        ,SOH.END_TIME
        ,SOH.CREDITS_USED
        ,$CREDIT_PRICE
        ,($CREDIT_PRICE*SOH.CREDITS_USED) AS DOLLARS_USED
        ,'ACTUAL COMPUTE' AS MEASURE_TYPE
from    SNOWFLAKE.ACCOUNT_USAGE.SEARCH_OPTIMIZATION_HISTORY SOH
```

```
UNION ALL

--COMPUTE FROM REPLICATION
SELECT
         'Replication' AS WAREHOUSE_GROUP_NAME
        ,DATABASE_NAME AS WAREHOUSE_NAME
        ,NULL AS GROUP_CONTACT
        ,NULL AS GROUP_COST_CENTER
        ,NULL AS GROUP_COMMENT
        ,RUH.START_TIME
        ,RUH.END_TIME
        ,RUH.CREDITS_USED
        ,$CREDIT_PRICE
        ,($CREDIT_PRICE*RUH.CREDITS_USED) AS DOLLARS_USED
        ,'ACTUAL COMPUTE' AS MEASURE_TYPE
from    SNOWFLAKE.ACCOUNT_USAGE.REPLICATION_USAGE_HISTORY RUH

UNION ALL

--STORAGE COSTS
SELECT
         'Storage' AS WAREHOUSE_GROUP_NAME
        ,'Storage' AS WAREHOUSE_NAME
        ,NULL AS GROUP_CONTACT
        ,NULL AS GROUP_COST_CENTER
        ,NULL AS GROUP_COMMENT
        ,SU.USAGE_DATE
        ,SU.USAGE_DATE
        ,NULL AS CREDITS_USED
        ,$CREDIT_PRICE
        ,((STORAGE_BYTES + STAGE_BYTES +
FAILSAFE_BYTES)/(1024*1024*1024*1024)*23)/DA.DAYS_IN_MONTH AS DOLLARS_USED
        ,'ACTUAL COMPUTE' AS MEASURE_TYPE
from    SNOWFLAKE.ACCOUNT_USAGE.STORAGE_USAGE SU
JOIN    (SELECT COUNT(*) AS DAYS_IN_MONTH,TO_DATE(DATE_PART('year',D_DATE)||'-
'||DATE_PART('month',D_DATE)||'-01') as DATE_MONTH FROM
SNOWFLAKE_SAMPLE_DATA.TPCDS_SF10TCL.DATE_DIM GROUP BY
TO_DATE(DATE_PART('year',D_DATE)||'-'||DATE_PART('month',D_DATE)||'-01')) DA ON
DA.DATE_MONTH = TO_DATE(DATE_PART('year',USAGE_DATE)||'-
'||DATE_PART('month',USAGE_DATE)||'-01')

UNION ALL


SELECT
         NULL as WAREHOUSE_GROUP_NAME
        ,NULL as WAREHOUSE_NAME
        ,NULL as GROUP_CONTACT
        ,NULL as GROUP_COST_CENTER
        ,NULL as GROUP_COMMENT
        ,DA.D_DATE::timestamp as START_TIME
        ,DA.D_DATE::timestamp as END_TIME
```

```
        ,PU.CREDITS_PER_DAY AS CREDITS_USED
        ,PU.CREDIT_PRICE
        ,PU.DOLLARS_PER_DAY AS DOLLARS_USED
        ,'PROJECTED COMPUTE' AS MEASURE_TYPE
FROM    PROJECTED_USAGE PU
JOIN    SNOWFLAKE_SAMPLE_DATA.TPCDS_SF10TCL.DATE_DIM DA ON DA.D_DATE BETWEEN
PU.CONTRACT_START_DATE AND PU.CONTRACT_END_DATE

UNION ALL


SELECT
         NULL as WAREHOUSE_GROUP_NAME
        ,NULL as WAREHOUSE_NAME
        ,NULL as GROUP_CONTACT
        ,NULL as GROUP_COST_CENTER
        ,NULL as GROUP_COMMENT
        ,NULL as START_TIME
        ,NULL as END_TIME
        ,NULL AS CREDITS_USED
        ,PU.CREDIT_PRICE
        ,PU.TOTAL_CONTRACT_VALUE AS DOLLARS_USED
        ,'CONTRACT VALUES' AS MEASURE_TYPE
FROM    PROJECTED_USAGE PU
;
```

Screenshot

	WAREHOUSE_GROUP_NAME	WAREHOUSE_NAME	GROUP_CONTACT	GROUP_COST_CENTER	GROUP_COMMENT	START_TIME	END_TIME	CREDITS_USED	$CREDIT_PRICE	DOLLARS_USED	MEASURE_TYPE
1	WH Compute	COMPUTE_WH				2021-07-09 13:00:00.000 -0700	2021-07-09 14:00:00.000 -0700	0.399620556	4	1.598482224	ACTUAL COMPUTE
2	WH Compute	RESET_WH				2021-07-09 12:00:00.000 -0700	2021-07-09 13:00:00.000 -0700	0.000061389	4	0.000245556	ACTUAL COMPUTE
3	WH Compute	RESET_WH				2021-07-29 06:00:00.000 -0700	2021-07-29 07:00:00.000 -0700	2.752222222	4	11.008888888	ACTUAL COMPUTE
4	Storage	Storage				2021-03-21 00:00:00.000 -0700	2021-03-21 00:00:00.000 -0700		4	0.625538534603	ACTUAL COMPUTE
5	Storage	Storage				2020-10-10 00:00:00.000 -0700	2020-10-10 00:00:00.000 -0700		4	0.625538534601	ACTUAL COMPUTE
6	Storage	Storage				2020-08-25 00:00:00.000 -0700	2020-08-25 00:00:00.000 -0700		4	0.625538534603	ACTUAL COMPUTE
7	Storage	Storage				2020-11-10 00:00:00.000 -0800	2020-11-10 00:00:00.000 -0800		4	0.646389819087	ACTUAL COMPUTE
8	Storage	Storage				2021-07-04 00:00:00.000 -0700	2021-07-04 00:00:00.000 -0700		4	0.625538534829	ACTUAL COMPUTE
9	Storage	Storage				2021-05-19 00:00:00.000 -0700	2021-05-19 00:00:00.000 -0700		4	0.625538534829	ACTUAL COMPUTE
10	Storage	Storage				2020-12-26 00:00:00.000 -0800	2020-12-26 00:00:00.000 -0800		4	0.6255385346	ACTUAL COMPUTE
11	Storage	Storage				2021-08-06 00:00:00.000 -0700	2021-08-06 00:00:00.000 -0700		4	0.625539930966	ACTUAL COMPUTE
12	Storage	Storage				2021-02-12 00:00:00.000 -0800	2021-02-12 00:00:00.000 -0800		4	0.69256052045	ACTUAL COMPUTE
13	Storage	Storage				2021-07-20 00:00:00.000 -0700	2021-07-20 00:00:00.000 -0700		4	0.625539522069	ACTUAL COMPUTE

Most Expensive Queries (T2)

TIER 2

Description:

This query orders the most expensive queries from the last 30 days. It takes into account the warehouse size, assuming that a 1 minute query on larger warehouse is more expensive than a 1 minute query on a smaller warehouse

How to Interpret Results:

This is an opportunity to evaluate expensive queries and take some action. The admin could:

-look at the query profile

-contact the user who executed the query

-take action to optimize these queries

Primary Schema:

Account_Usage

SQL

```
WITH WAREHOUSE_SIZE AS
(
     SELECT WAREHOUSE_SIZE, NODES
       FROM (
              SELECT 'XSMALL' AS WAREHOUSE_SIZE, 1 AS NODES
              UNION ALL
              SELECT 'SMALL' AS WAREHOUSE_SIZE, 2 AS NODES
              UNION ALL
              SELECT 'MEDIUM' AS WAREHOUSE_SIZE, 4 AS NODES
              UNION ALL
              SELECT 'LARGE' AS WAREHOUSE_SIZE, 8 AS NODES
              UNION ALL
              SELECT 'XLARGE' AS WAREHOUSE_SIZE, 16 AS NODES
              UNION ALL
              SELECT '2XLARGE' AS WAREHOUSE_SIZE, 32 AS NODES
              UNION ALL
              SELECT '3XLARGE' AS WAREHOUSE_SIZE, 64 AS NODES
              UNION ALL
              SELECT '4XLARGE' AS WAREHOUSE_SIZE, 128 AS NODES
            )
),
QUERY_HISTORY AS
(
     SELECT QH.QUERY_ID
           ,QH.QUERY_TEXT
           ,QH.USER_NAME
           ,QH.ROLE_NAME
           ,QH.EXECUTION_TIME
           ,QH.WAREHOUSE_SIZE
      FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY QH
     WHERE START_TIME > DATEADD(month,-2,CURRENT_TIMESTAMP())
)

SELECT QH.QUERY_ID
      ,'https://' || current_account() ||
'.snowflakecomputing.com/console#/monitoring/queries/detail?queryId='||QH.QUERY_ID AS
QU
      ,QH.QUERY_TEXT
      ,QH.USER_NAME
      ,QH.ROLE_NAME
      ,QH.EXECUTION_TIME as EXECUTION_TIME_MILLISECONDS
      ,(QH.EXECUTION_TIME/(1000)) as EXECUTION_TIME_SECONDS
      ,(QH.EXECUTION_TIME/(1000*60)) AS EXECUTION_TIME_MINUTES
      ,(QH.EXECUTION_TIME/(1000*60*60)) AS EXECUTION_TIME_HOURS
      ,WS.WAREHOUSE_SIZE
      ,WS.NODES
      ,(QH.EXECUTION_TIME/(1000*60*60))*WS.NODES as RELATIVE_PERFORMANCE_COST
```

```
FROM QUERY_HISTORY QH
JOIN WAREHOUSE_SIZE WS ON WS.WAREHOUSE_SIZE = upper(QH.WAREHOUSE_SIZE)
ORDER BY RELATIVE_PERFORMANCE_COST DESC
LIMIT 200
;
```

Average Cost per Query by Warehouse (T2)

TIER 2

Description:

This summarize the query activity and credit consumption per warehouse over the last month. The query also includes the ratio of queries executed to credits consumed on the warehouse

How to Interpret Results:

Highlights any scenarios where warehouse consumption is significantly out of line with the number of queries executed. Maybe auto-suspend needs to be adjusted or warehouses need to be consolidated.

Primary Schema:

Account_Usage

SQL

```
set credit_price = 4;  --edit this value to reflect your credit price

SELECT
    COALESCE(WC.WAREHOUSE_NAME,QC.WAREHOUSE_NAME) AS WAREHOUSE_NAME
    ,QC.QUERY_COUNT_LAST_MONTH
    ,WC.CREDITS_USED_LAST_MONTH
    ,WC.CREDIT_COST_LAST_MONTH
    ,CAST((WC.CREDIT_COST_LAST_MONTH / QC.QUERY_COUNT_LAST_MONTH) AS decimal(10,2) )
AS COST_PER_QUERY

FROM (
    SELECT
       WAREHOUSE_NAME
      ,COUNT(QUERY_ID) as QUERY_COUNT_LAST_MONTH
    FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
    WHERE TO_DATE(START_TIME) >= TO_DATE(DATEADD(month,-1,CURRENT_TIMESTAMP()))
    GROUP BY WAREHOUSE_NAME
      ) QC
JOIN (

    SELECT
        WAREHOUSE_NAME
        ,SUM(CREDITS_USED) as CREDITS_USED_LAST_MONTH
        ,SUM(CREDITS_USED)*($CREDIT_PRICE) as CREDIT_COST_LAST_MONTH
    FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
    WHERE TO_DATE(START_TIME) >= TO_DATE(DATEADD(month,-1,CURRENT_TIMESTAMP()))
    GROUP BY WAREHOUSE_NAME
  ) WC
```

```
    ON WC.WAREHOUSE_NAME = QC.WAREHOUSE_NAME

ORDER BY COST_PER_QUERY DESC
;
```

Screenshot

WAREHOUSE_NAME	QUERY_COUNT_LAST_MONTH	CREDITS_USED_LAST_MONTH	CREDIT_COST_LAST_MONTH	COST_PER_QUERY
DATALAKE_WH	53	8.392294444	33.569177776	0.63
COMPUTE_WH	234	10.023920556	40.095682224	0.17
RESET_WH	109	3.836485278	15.345941112	0.14
ANALYTICS_WH	189	2.526540556	10.106162224	0.05
LOAD_WH	219	2.752667501	11.010670004	0.05
TASK_WH	302	1.077893611	4.311574444	0.01
BI_MEDIUM_WH	2	0.000060000	0.000240000	0.00
BI_LARGE_WH	16734	0.481809443	1.927237772	0.00

AutoClustering Cost History (by Day by Object) (T3)

TIER 3

Description:

Full list of tables with auto-clustering and the volume of credits consumed via the service over the last 30 days, broken out by day.

How to Interpret Results:

Look for irregularities in the credit consumption or consistently high consumption

Primary Schema:

Account_Usage

SQL

```
SELECT TO_DATE(START_TIME) as DATE
,DATABASE_NAME
,SCHEMA_NAME
,TABLE_NAME
,SUM(CREDITS_USED) as CREDITS_USED


FROM "SNOWFLAKE"."ACCOUNT_USAGE"."AUTOMATIC_CLUSTERING_HISTORY"

WHERE START_TIME >= dateadd(month,-1,current_timestamp())
GROUP BY 1,2,3,4
ORDER BY 5 DESC
;
```

Screenshot

CLUSTERING_DATE	DATABASE_NAME	SCHEMA_NAME	TABLE_NAME	CREDITS_USED
2020-02-19	ASHISH_TEST_DB	PUBLIC	GENOTYPES_COPY	266.442864197
2020-04-28	PACIONE_TEST	PUBLIC	COLLAT_AMT_FACT_AC	118.290533383
2020-10-13	CARLINENG	TPCDS_UNCLUSTERED	AUTOCLUSTER_SOLD_DATE_ITEM	78.132147290
2020-10-13	CARLINENG	TPCDS_UNCLUSTERED	AUTOCLUSTER_TICKET_NUMBER	67.328296503
2020-10-13	CARLINENG	TPCDS_UNCLUSTERED	AUTOCLUSTER_SOLD_DATE	57.013410098
2020-01-08	GENOMES_STAGE	GENOME1K	GENOTYPES	29.637784249
2019-12-18	JCLARKE_DB	PUBLIC	SNOW121900_CBD	25.405727086
2020-06-18	SKIPSANDBOX	PUBLIC	WEB_SALES	18.709104655
2019-12-18	JCLARKE_DB	PUBLIC	SNOW121900_CBTS	11.821695786
2020-06-18	SKIPSANDBOX	PUBLIC	TEST_CLUSTER_IMPACT_ON_CARD_EST	9.956016226
2020-07-01	GEOSPATIAL	NASANEX	NEX_GDDP_ALL	8.324776052
2020-06-25	JCLARKE_DB	PUBLIC	C_TEST	4.867021840
2020-07-30	SKIPSANDBOX	PUBLIC	SEARCH OPTIMIZATION ON TABLE_ID:2778292	4.259085124
2020-04-28	NTIERNEY_TEST_DB	CLUSTER_TEST	V_ORDERS_CUSTOMER	3.167348136
2020-06-22	CARLOS_TEST_DB	PUBLIC	LINEITEM	2.060566449
2020-06-26	JCLARKE_DB	PUBLIC	C_TEST	1.202380758
2020-07-02	GEOSPATIAL	NASANEX	NEX_GDDP_ALL	0.698955203
2020-04-28	NTIERNEY_TEST_DB	CLUSTER_TEST	ORDERS	0.575332238
2020-06-23	CARLOS_TEST_DB	PUBLIC	LINEITEM	0.363553380
2020-09-23	CARLINENG	PUBLIC	ORDERS	0.212436370

Materialized Views Cost History (by Day by Object) (T3)

TIER 3

Description:

Full list of materialized views and the volume of credits consumed via the service over the last 30 days, broken out by day.

How to Interpret Results:

Look for irregularities in the credit consumption or consistently high consumption

Primary Schema:

Account_Usage

SQL

```
SELECT

TO_DATE(START_TIME) as DATE
,DATABASE_NAME
,SCHEMA_NAME
,TABLE_NAME
,SUM(CREDITS_USED) as CREDITS_USED

FROM "SNOWFLAKE"."ACCOUNT_USAGE"."MATERIALIZED_VIEW_REFRESH_HISTORY"

WHERE START_TIME >= dateadd(month,-1,current_timestamp())
GROUP BY 1,2,3,4
ORDER BY 5 DESC
;
```

Search Optimization Cost History (by Day by Object) (T3)

TIER 3

Description:

Full list of tables with search optimization and the volume of credits consumed via the service over the last 30 days, broken out by day.

How to Interpret Results:

Look for irregularities in the credit consumption or consistently high consumption

Primary Schema:

Account_Usage

SQL

```
TSELECT

TO_DATE(START_TIME) as DATE
,DATABASE_NAME
,SCHEMA_NAME
,TABLE_NAME
,SUM(CREDITS_USED) as CREDITS_USED

FROM "SNOWFLAKE"."ACCOUNT_USAGE"."SEARCH_OPTIMIZATION_HISTORY"

WHERE START_TIME >= dateadd(month,-1,current_timestamp())
GROUP BY 1,2,3,4
ORDER BY 5 DESC
;
```

Snowpipe Cost History (by Day by Object) (T3)

TIER 3

Description:

Full list of pipes and the volume of credits consumed via the service over the last 30 days, broken out by day.

How to Interpret Results:

Look for irregularities in the credit consumption or consistently high consumption

Primary Schema:

Account_Usage

SQL

```
SELECT

TO_DATE(START_TIME) as DATE
,PIPE_NAME
,SUM(CREDITS_USED) as CREDITS_USED

FROM "SNOWFLAKE"."ACCOUNT_USAGE"."PIPE_USAGE_HISTORY"

WHERE START_TIME >= dateadd(month,-1,current_timestamp())
GROUP BY 1,2
```

```
ORDER BY 3 DESC
;
```

Replication Cost History (by Day by Object) (T3)

TIER 3

Description:

Full list of replicated databases and the volume of credits consumed via the replication service over the last 30 days, broken out by day.

How to Interpret Results:

Look for irregularities in the credit consumption or consistently high consumption

Primary Schema:

Account_Usage

SQL

```
SELECT
TO_DATE(START_TIME) as DATE
,DATABASE_NAME
,SUM(CREDITS_USED) as CREDITS_USED
FROM "SNOWFLAKE"."ACCOUNT_USAGE"."REPLICATION_USAGE_HISTORY"
WHERE START_TIME >= dateadd(month,-1,current_timestamp())
GROUP BY 1,2
ORDER BY 3 DESC
;
```