Resource Optimization: Usage Monitoring

Usage Monitoring

Usage Monitoring queries are designed to identify the warehouses, queries, tools, and users that are responsible for consuming the most credits over a specified period of time. These queries can be used to determine which of those resources are consuming more credits than anticipated and take the necessary steps to reduce their consumption.

What You'll Learn

- how to analyze consumption trends and patterns
- how to identify consumption anomolies
- how to analyze partner tool consumption metrics

What You'll Need

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

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.

Credit Consumption by Warehouse (T1)

TIER 1

Description:

Shows the total credit consumption for each warehouse over a specific time period.

How to Interpret Results:

Are there specific warehouses that are consuming more credits than the others? Should they be? Are there specific warehouses that are consuming more credits than anticipated for that warehouse?

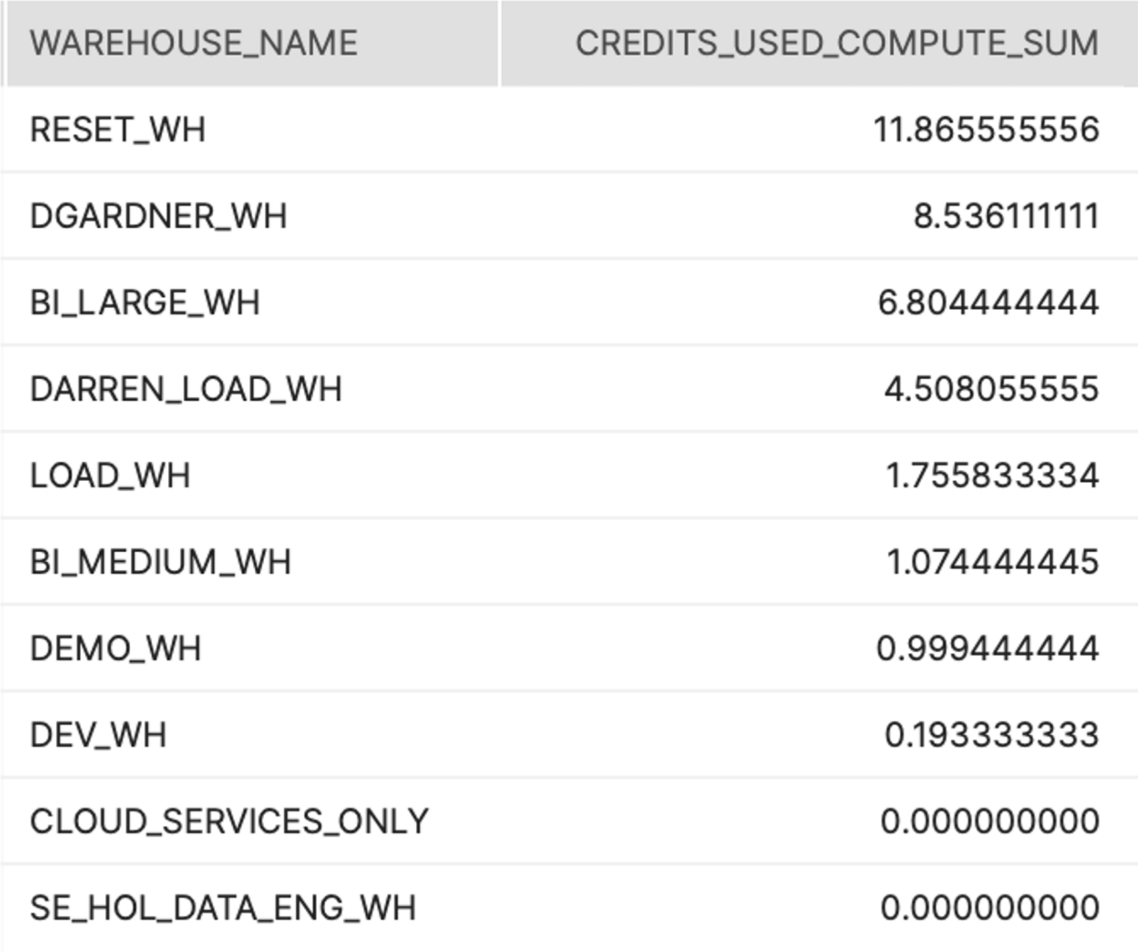
Primary Schema:

Account_Usage

SQL

```
// Credits used (all time = past year)
SELECT WAREHOUSE_NAME
      ,SUM(CREDITS_USED_COMPUTE) AS CREDITS_USED_COMPUTE_SUM
  FROM ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
 GROUP BY 1
 ORDER BY 2 DESC
;

// Credits used (past N days/weeks/months)
SELECT WAREHOUSE_NAME
      ,SUM(CREDITS_USED_COMPUTE) AS CREDITS_USED_COMPUTE_SUM
  FROM ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
 WHERE START_TIME >= DATEADD(DAY, -7, CURRENT_TIMESTAMP())  // Past 7 days
 GROUP BY 1
 ORDER BY 2 DESC
;
```

Screenshot



WAREHOUSE_NAME	CREDITS_USED_COMPUTE_SUM
RESET_WH	11.865555556
DGARDNER_WH	8.536111111
BI_LARGE_WH	6.804444444
DARREN_LOAD_WH	4.508055555
LOAD_WH	1.755833334
BI_MEDIUM_WH	1.074444445
DEMO_WH	0.999444444
DEV_WH	0.193333333
CLOUD_SERVICES_ONLY	0.000000000
SE_HOL_DATA_ENG_WH	0.000000000

Average Hour-by-Hour Consumption Over the Past 7 Days (T1)

TIER 1

Description:

Shows the total credit consumption on an hourly basis to help understand consumption trends (peaks, valleys) over the past 7 days.

How to Interpret Results:

At which points of the day are we seeing spikes in our consumption? Is that expected?

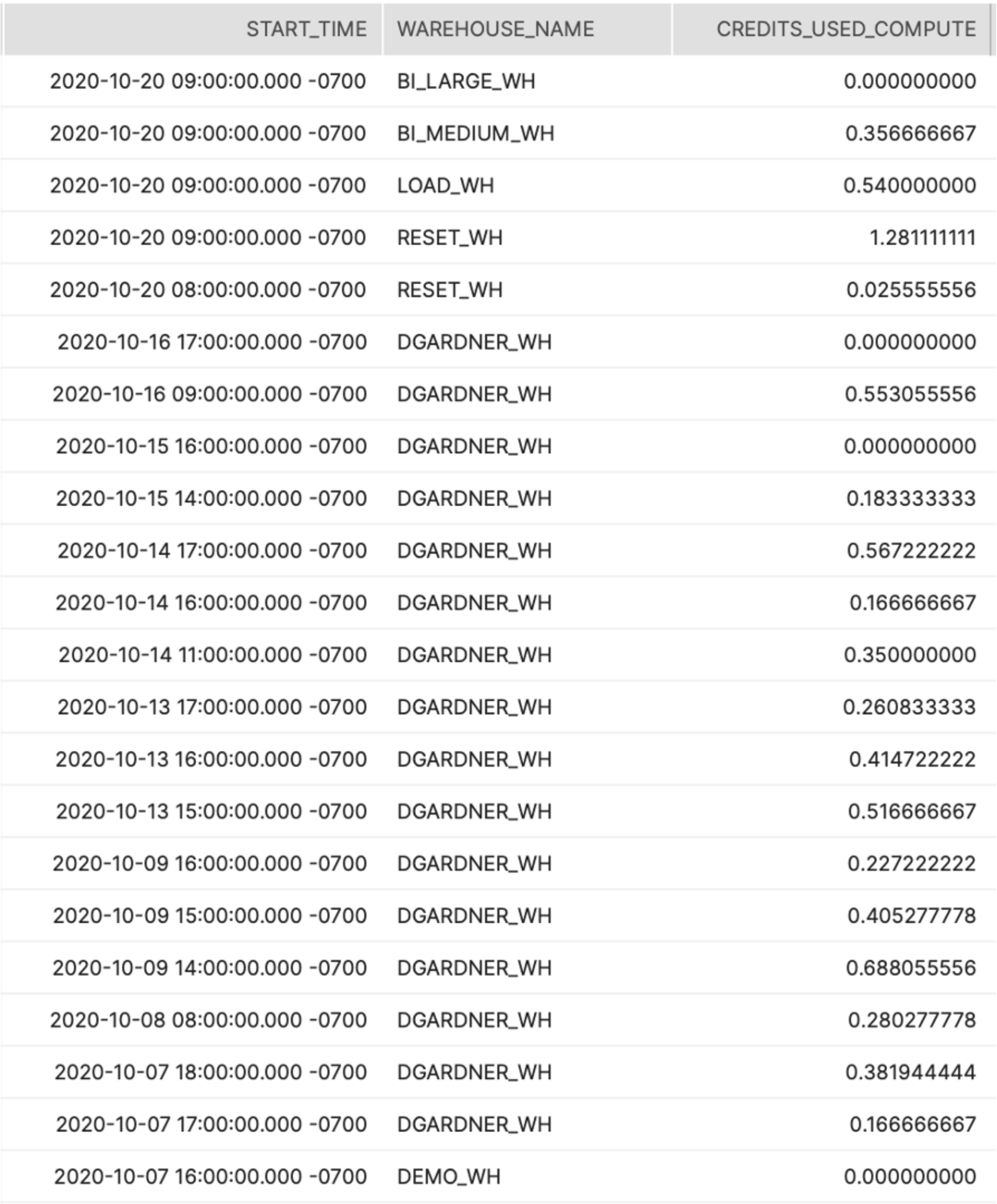
Primary Schema:

Account_Usage

SQL (by hour, warehouse)

```sql
// Credits used by [hour, warehouse] (past 7 days)
SELECT START_TIME
      ,WAREHOUSE_NAME
      ,CREDITS_USED_COMPUTE
  FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
 WHERE START_TIME >= DATEADD(DAY, -7, CURRENT_TIMESTAMP())
   AND WAREHOUSE_ID > 0  // Skip pseudo-VWs such as "CLOUD_SERVICES_ONLY"
 ORDER BY 1 DESC,2
;
```

Screenshot

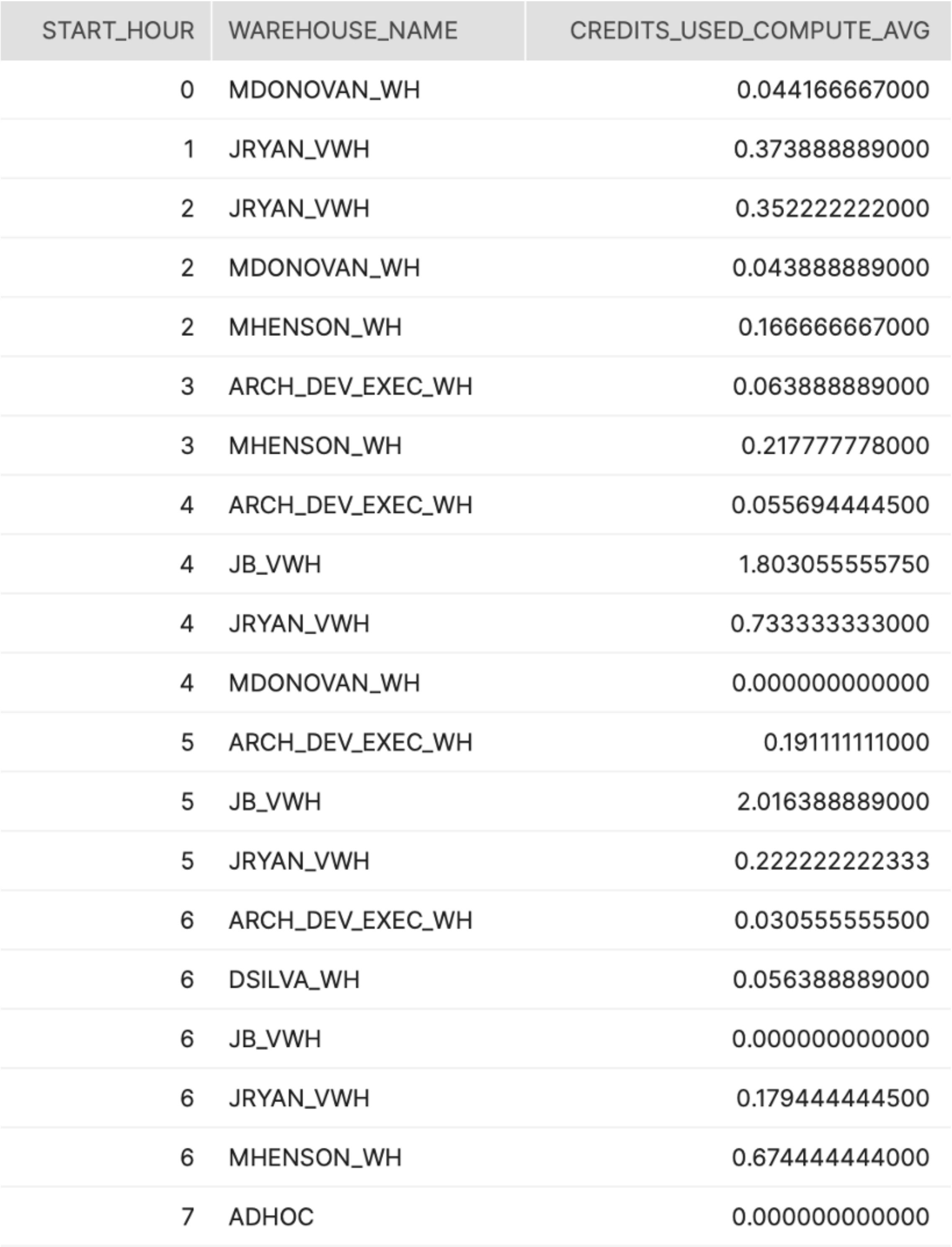


START_TIME	WAREHOUSE_NAME	CREDITS_USED_COMPUTE
2020-10-20 09:00:00.000 -0700	BI_LARGE_WH	0.000000000
2020-10-20 09:00:00.000 -0700	BI_MEDIUM_WH	0.356666667
2020-10-20 09:00:00.000 -0700	LOAD_WH	0.540000000
2020-10-20 09:00:00.000 -0700	RESET_WH	1.281111111
2020-10-20 08:00:00.000 -0700	RESET_WH	0.025555556
2020-10-16 17:00:00.000 -0700	DGARDNER_WH	0.000000000
2020-10-16 09:00:00.000 -0700	DGARDNER_WH	0.553055556
2020-10-15 16:00:00.000 -0700	DGARDNER_WH	0.000000000
2020-10-15 14:00:00.000 -0700	DGARDNER_WH	0.183333333
2020-10-14 17:00:00.000 -0700	DGARDNER_WH	0.567222222
2020-10-14 16:00:00.000 -0700	DGARDNER_WH	0.166666667
2020-10-14 11:00:00.000 -0700	DGARDNER_WH	0.350000000
2020-10-13 17:00:00.000 -0700	DGARDNER_WH	0.260833333
2020-10-13 16:00:00.000 -0700	DGARDNER_WH	0.414722222
2020-10-13 15:00:00.000 -0700	DGARDNER_WH	0.516666667
2020-10-09 16:00:00.000 -0700	DGARDNER_WH	0.227222222
2020-10-09 15:00:00.000 -0700	DGARDNER_WH	0.405277778
2020-10-09 14:00:00.000 -0700	DGARDNER_WH	0.688055556
2020-10-08 08:00:00.000 -0700	DGARDNER_WH	0.280277778
2020-10-07 18:00:00.000 -0700	DGARDNER_WH	0.381944444
2020-10-07 17:00:00.000 -0700	DGARDNER_WH	0.166666667
2020-10-07 16:00:00.000 -0700	DEMO_WH	0.000000000

####SQL (by hour)

```
SELECT DATE_PART('HOUR', START_TIME) AS START_HOUR
      ,WAREHOUSE_NAME
      ,AVG(CREDITS_USED_COMPUTE) AS CREDITS_USED_COMPUTE_AVG
  FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
 WHERE START_TIME >= DATEADD(DAY, -7, CURRENT_TIMESTAMP())
   AND WAREHOUSE_ID > 0  // Skip pseudo-VWs such as "CLOUD_SERVICES_ONLY"
 GROUP BY 1, 2
 ORDER BY 1, 2
;
```

Screenshot



START_HOUR	WAREHOUSE_NAME	CREDITS_USED_COMPUTE_AVG
0	MDONOVAN_WH	0.044166667000
1	JRYAN_VWH	0.373888889000
2	JRYAN_VWH	0.352222222000
2	MDONOVAN_WH	0.043888889000
2	MHENSON_WH	0.166666667000
3	ARCH_DEV_EXEC_WH	0.063888889000
3	MHENSON_WH	0.217777778000
4	ARCH_DEV_EXEC_WH	0.055694444500
4	JB_VWH	1.803055555750
4	JRYAN_VWH	0.733333333000
4	MDONOVAN_WH	0.000000000000
5	ARCH_DEV_EXEC_WH	0.191111111000
5	JB_VWH	2.016388889000
5	JRYAN_VWH	0.222222222333
6	ARCH_DEV_EXEC_WH	0.030555555500
6	DSILVA_WH	0.056388889000
6	JB_VWH	0.000000000000
6	JRYAN_VWH	0.179444444500
6	MHENSON_WH	0.674444444000
7	ADHOC	0.000000000000

Average Query Volume by Hour (Past 7 Days) (T1)

TIER 1

Description:

Shows average number of queries run on an hourly basis to help better understand typical query activity.

How to Interpret Results:

How many queries are being run on an hourly basis? Is this more or less than we anticipated? What could be causing this?

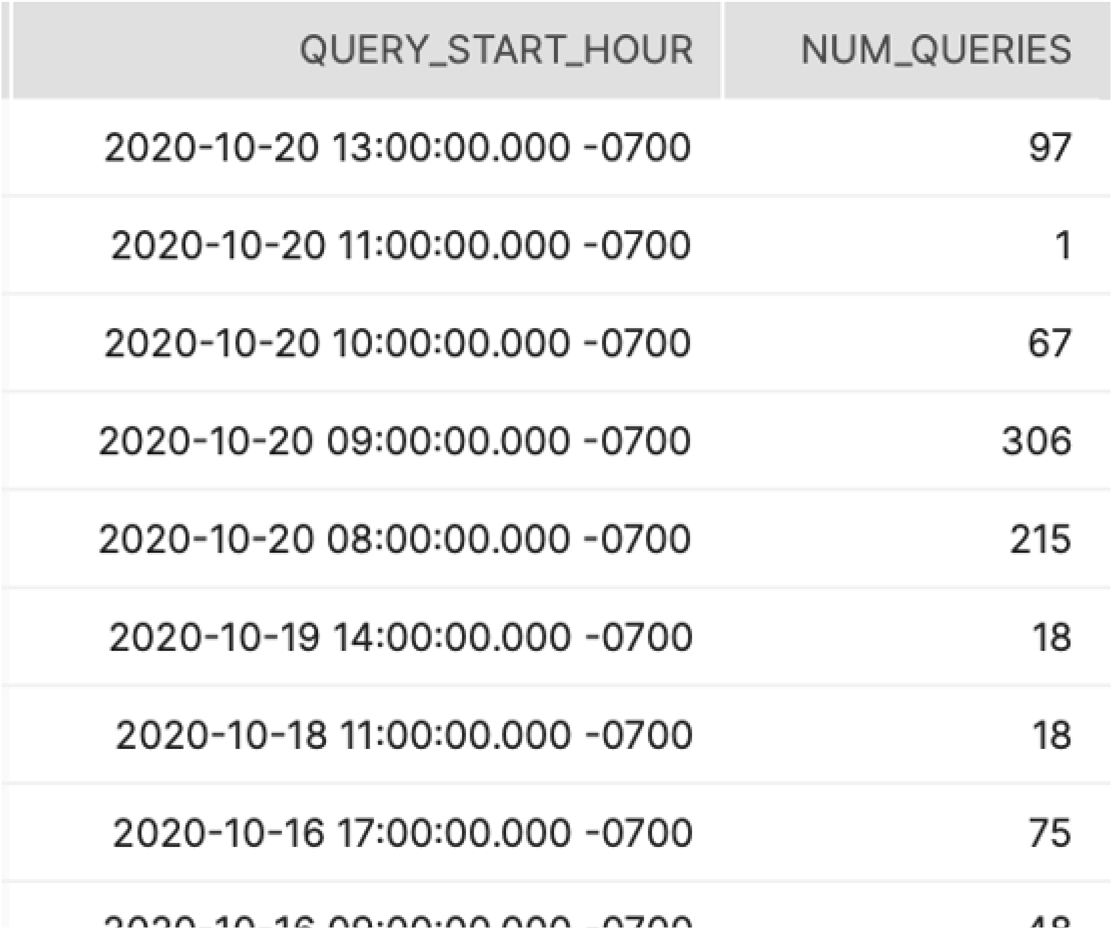
Primary Schema:

Account_Usage

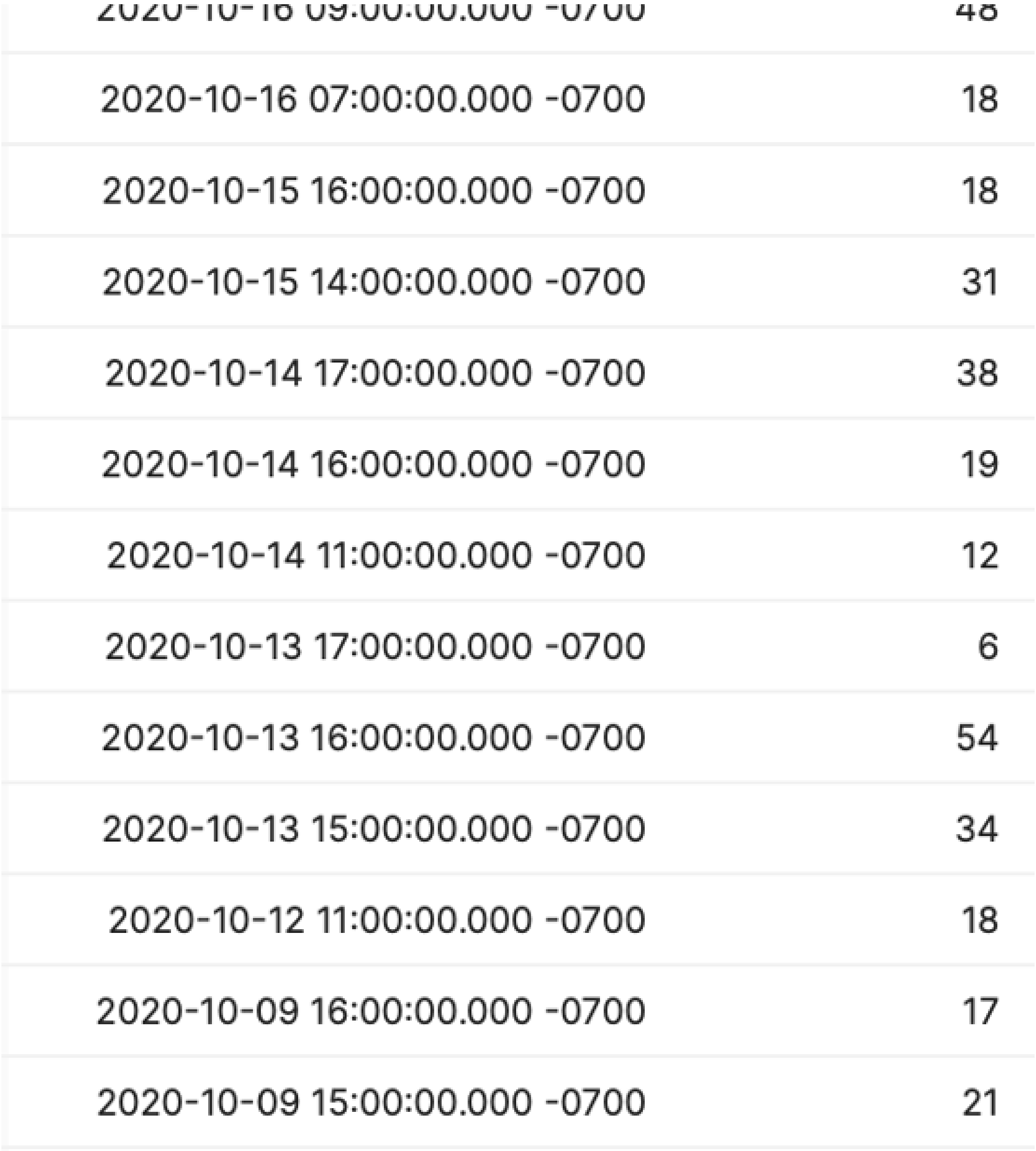
SQL

```
SELECT DATE_TRUNC('HOUR', START_TIME) AS QUERY_START_HOUR
      ,WAREHOUSE_NAME
      ,COUNT(*) AS NUM_QUERIES
  FROM SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY
 WHERE START_TIME >= DATEADD(DAY, -7, CURRENT_TIMESTAMP())  // Past 7 days
 GROUP BY 1, 2
 ORDER BY 1 DESC, 2
;
```

Screenshot



QUERY_START_HOUR	NUM_QUERIES
2020-10-20 13:00:00.000 -0700	97
2020-10-20 11:00:00.000 -0700	1
2020-10-20 10:00:00.000 -0700	67
2020-10-20 09:00:00.000 -0700	306
2020-10-20 08:00:00.000 -0700	215
2020-10-19 14:00:00.000 -0700	18
2020-10-18 11:00:00.000 -0700	18
2020-10-16 17:00:00.000 -0700	75
2020-10-16 09:00:00.000 -0700	48



2020-10-16 09:00:00.000 -0700	48
2020-10-16 07:00:00.000 -0700	18
2020-10-15 16:00:00.000 -0700	18
2020-10-15 14:00:00.000 -0700	31
2020-10-14 17:00:00.000 -0700	38
2020-10-14 16:00:00.000 -0700	19
2020-10-14 11:00:00.000 -0700	12
2020-10-13 17:00:00.000 -0700	6
2020-10-13 16:00:00.000 -0700	54
2020-10-13 15:00:00.000 -0700	34
2020-10-12 11:00:00.000 -0700	18
2020-10-09 16:00:00.000 -0700	17
2020-10-09 15:00:00.000 -0700	21

Warehouse Utilization Over 7 Day Average (T1)

TIER 1

Description:

This query returns the daily average of credit consumption grouped by week and warehouse.

How to Interpret Results:

Use this to identify anomolies in credit consumption for warehouses across weeks from the past year.

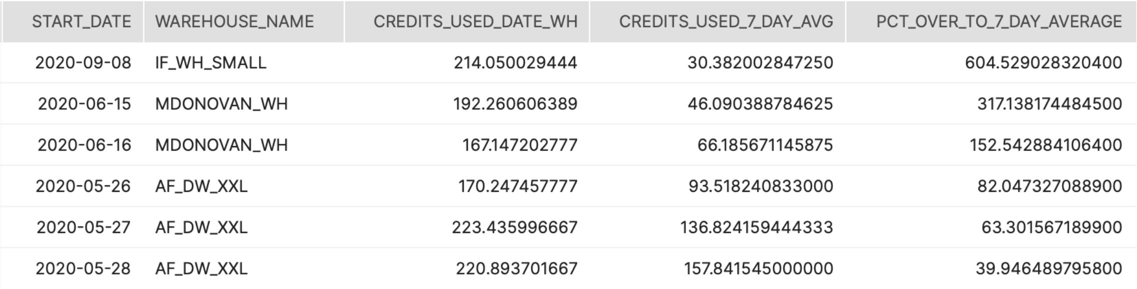
Primary Schema:

Account_Usage

SQL

```
WITH CTE_DATE_WH AS(
  SELECT TO_DATE(START_TIME) AS START_DATE
        ,WAREHOUSE_NAME
        ,SUM(CREDITS_USED) AS CREDITS_USED_DATE_WH
    FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY
   GROUP BY START_DATE
           ,WAREHOUSE_NAME
)
SELECT START_DATE
      ,WAREHOUSE_NAME
      ,CREDITS_USED_DATE_WH
      ,AVG(CREDITS_USED_DATE_WH) OVER (PARTITION BY WAREHOUSE_NAME ORDER BY START_DATE
ROWS 7 PRECEDING) AS CREDITS_USED_7_DAY_AVG
      ,100.0*((CREDITS_USED_DATE_WH / CREDITS_USED_7_DAY_AVG) - 1) AS
PCT_OVER_TO_7_DAY_AVERAGE
  FROM CTE_DATE_WH
QUALIFY CREDITS_USED_DATE_WH > 100  // Minimum N=100 credits
    AND PCT_OVER_TO_7_DAY_AVERAGE >= 0.5  // Minimum 50% increase over past 7 day
average
 ORDER BY PCT_OVER_TO_7_DAY_AVERAGE DESC
;
```

Screenshot



START_DATE	WAREHOUSE_NAME	CREDITS_USED_DATE_WH	CREDITS_USED_7_DAY_AVG	PCT_OVER_TO_7_DAY_AVERAGE
2020-09-08	IF_WH_SMALL	214.050029444	30.382002847250	604.529028320400
2020-06-15	MDONOVAN_WH	192.260606389	46.090388784625	317.138174484500
2020-06-16	MDONOVAN_WH	167.147202777	66.185671145875	152.542884106400
2020-05-26	AF_DW_XXL	170.247457777	93.518240833000	82.047327088900
2020-05-27	AF_DW_XXL	223.435996667	136.824159444333	63.301567189900
2020-05-28	AF_DW_XXL	220.893701667	157.841545000000	39.946489795800

Forecasting Usage/Billing (T1)

TIER 1

Description:

This query provides three distinct consumption metrics for each day of the contract term. (1) the contracted consumption is the dollar amount consumed if usage was flat for the entire term. (2) the actual consumption pulls from the various usage views and aggregates dollars at a day level. (3) the forecasted consumption creates a straight line regression from the actuals to project go-forward consumption.

How to Interpret Results:

This data should be mapped as line graphs with a running total calculation to estimate future forecast against the contract amount.

Primary Schema:

Account_Usage

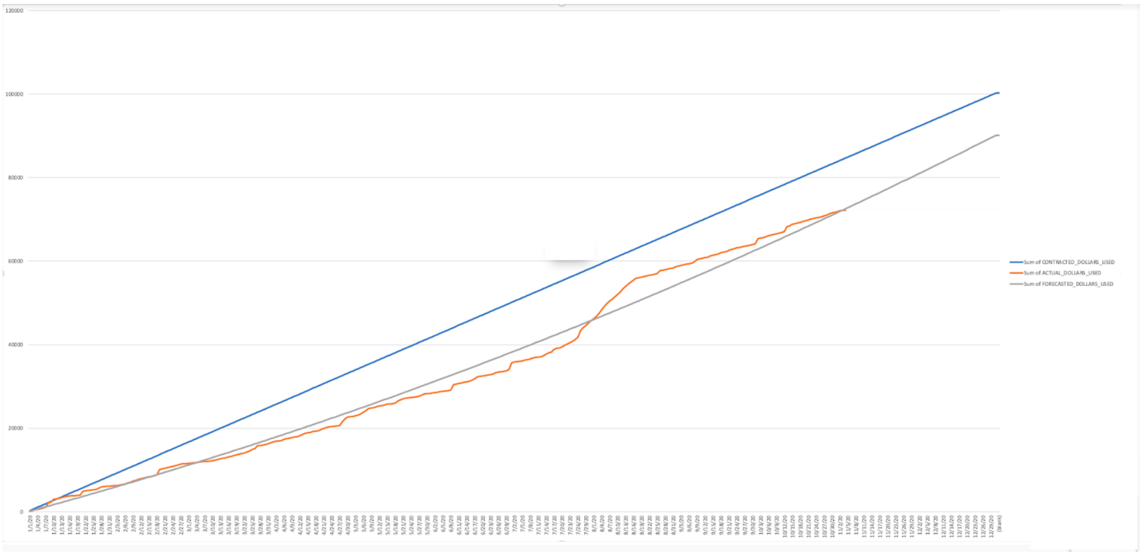
SQL

```sql
SET CREDIT_PRICE = 4.00; --edit this number to reflect credit price
SET TERM_LENGTH = 12; --integer value in months
SET TERM_START_DATE = '2020-01-01';
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
             ,$TERM_START_DATE::timestamp as CONTRACT_START_DATE
             ,DATEADD(day,-1,DATEADD(month,$TERM_LENGTH,$TERM_START_DATE))::timestamp
as CONTRACT_END_DATE
),
PROJECTED_USAGE AS (
     SELECT
                CREDIT_PRICE
               ,TOTAL_CONTRACT_VALUE
               ,CONTRACT_START_DATE
               ,CONTRACT_END_DATE
               ,(TOTAL_CONTRACT_VALUE)
                   /
                   DATEDIFF(day,CONTRACT_START_DATE,CONTRACT_END_DATE)  AS
DOLLARS_PER_DAY
               , (TOTAL_CONTRACT_VALUE/CREDIT_PRICE)
                   /
               DATEDIFF(day,CONTRACT_START_DATE,CONTRACT_END_DATE) AS CREDITS_PER_DAY
     FROM      CONTRACT_VALUES
),
ACTUAL_USAGE AS (
 SELECT TO_DATE(START_TIME) AS CONSUMPTION_DATE
   ,SUM(DOLLARS_USED) as ACTUAL_DOLLARS_USED
 FROM (
   --COMPUTE FROM WAREHOUSES
   SELECT
            'WH Compute' as WAREHOUSE_GROUP_NAME
           ,WMH.WAREHOUSE_NAME
           ,NULL AS GROUP_CONTACT
           ,NULL AS GROUP_COST_CENTER
           ,NULL AS GROUP_COMMENT
           ,WMH.START_TIME
           ,WMH.END_TIME
           ,WMH.CREDITS_USED
           ,$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')
) A
 group by 1
),
FORECASTED_USAGE_SLOPE_INTERCEPT as (
 SELECT

REGR_SLOPE(AU.ACTUAL_DOLLARS_USED,DATEDIFF(day,CONTRACT_START_DATE,AU.CONSUMPTION_DATE))
 as SLOPE

,REGR_INTERCEPT(AU.ACTUAL_DOLLARS_USED,DATEDIFF(day,CONTRACT_START_DATE,AU.CONSUMPTION_I
 as INTERCEPT
 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
 LEFT JOIN   ACTUAL_USAGE AU ON AU.CONSUMPTION_DATE = TO_DATE(DA.D_DATE)
)
SELECT
        DA.D_DATE::date as CONSUMPTION_DATE
       ,PU.DOLLARS_PER_DAY AS CONTRACTED_DOLLARS_USED
       ,AU.ACTUAL_DOLLARS_USED
       --the below is the mx+b equation to get the forecasted linear slope
       ,DATEDIFF(day,CONTRACT_START_DATE,DA.D_DATE)*FU.SLOPE + FU.INTERCEPT AS
FORECASTED_DOLLARS_USED
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
LEFT JOIN   ACTUAL_USAGE                                    AU ON AU.CONSUMPTION_DATE
= TO_DATE(DA.D_DATE)
JOIN        FORECASTED_USAGE_SLOPE_INTERCEPT                FU ON 1 = 1
;
```

Screenshot



Partner Tools Consuming Credits (T1)

TIER 1

Description:

Identifies which of Snowflake's partner tools/solutions (BI, ETL, etc.) are consuming the most credits.

How to Interpret Results:

Are there certain partner solutions that are consuming more credits than anticipated? What is the reasoning for this?

Primary Schema:

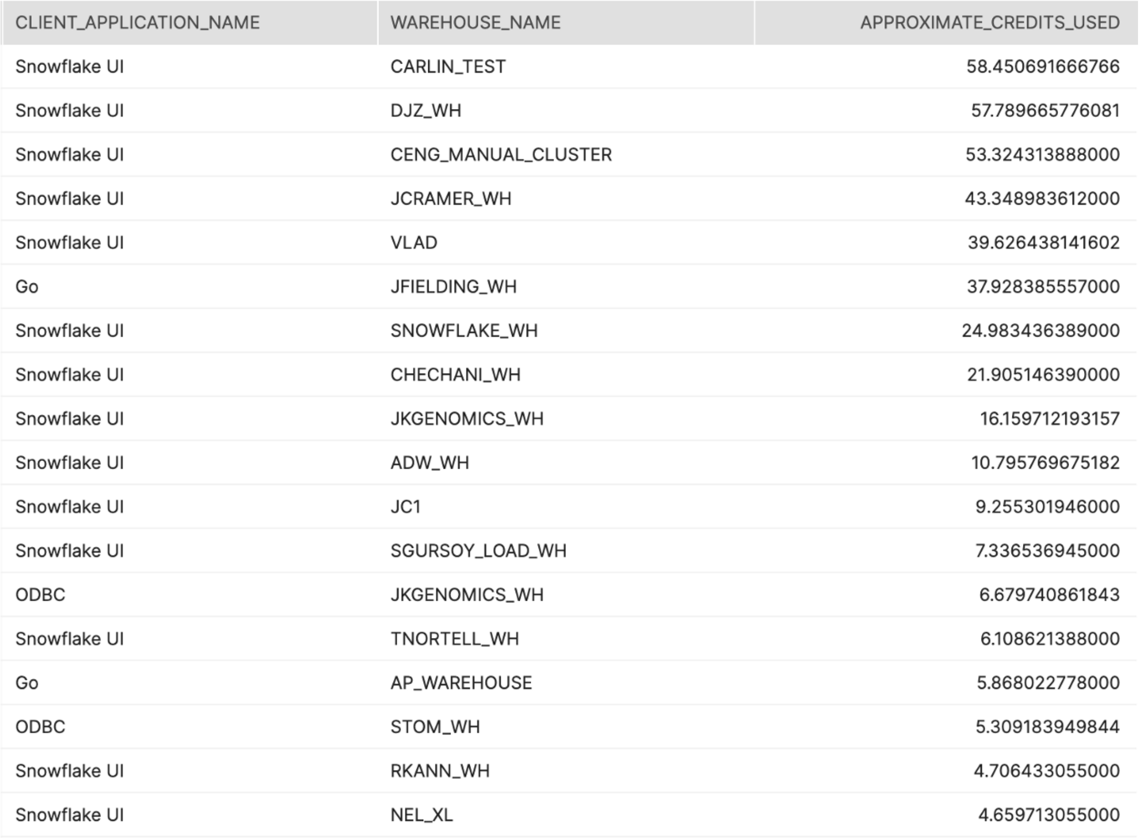
Account_Usage

SQL

```
--THIS IS APPROXIMATE CREDIT CONSUMPTION BY CLIENT APPLICATION
WITH CLIENT_HOUR_EXECUTION_CTE AS (
    SELECT  CASE
         WHEN CLIENT_APPLICATION_ID LIKE 'Go %' THEN 'Go'
         WHEN CLIENT_APPLICATION_ID LIKE 'Snowflake UI %' THEN 'Snowflake UI'
         WHEN CLIENT_APPLICATION_ID LIKE 'SnowSQL %' THEN 'SnowSQL'
         WHEN CLIENT_APPLICATION_ID LIKE 'JDBC %' THEN 'JDBC'
         WHEN CLIENT_APPLICATION_ID LIKE 'PythonConnector %' THEN 'Python'
         WHEN CLIENT_APPLICATION_ID LIKE 'ODBC %' THEN 'ODBC'
         ELSE 'NOT YET MAPPED: ' || CLIENT_APPLICATION_ID
       END AS CLIENT_APPLICATION_NAME
    ,WAREHOUSE_NAME
    ,DATE_TRUNC('hour',START_TIME) as START_TIME_HOUR
    ,SUM(EXECUTION_TIME)  as CLIENT_HOUR_EXECUTION_TIME
    FROM "SNOWFLAKE"."ACCOUNT_USAGE"."QUERY_HISTORY" QH
    JOIN "SNOWFLAKE"."ACCOUNT_USAGE"."SESSIONS" SE ON SE.SESSION_ID = QH.SESSION_ID
    WHERE WAREHOUSE_NAME IS NOT NULL
    AND EXECUTION_TIME > 0

 --Change the below filter if you want to look at a longer range than the last 1 month
    AND START_TIME > DATEADD(Month,-1,CURRENT_TIMESTAMP())
    group by 1,2,3
    )
, HOUR_EXECUTION_CTE AS (
    SELECT  START_TIME_HOUR
    ,WAREHOUSE_NAME
    ,SUM(CLIENT_HOUR_EXECUTION_TIME) AS HOUR_EXECUTION_TIME
    FROM CLIENT_HOUR_EXECUTION_CTE
    group by 1,2
)
, APPROXIMATE_CREDITS AS (
    SELECT
    A.CLIENT_APPLICATION_NAME
    ,C.WAREHOUSE_NAME
    ,(A.CLIENT_HOUR_EXECUTION_TIME/B.HOUR_EXECUTION_TIME)*C.CREDITS_USED AS
APPROXIMATE_CREDITS_USED

    FROM CLIENT_HOUR_EXECUTION_CTE A
    JOIN HOUR_EXECUTION_CTE B  ON A.START_TIME_HOUR = B.START_TIME_HOUR and
B.WAREHOUSE_NAME = A.WAREHOUSE_NAME
    JOIN "SNOWFLAKE"."ACCOUNT_USAGE"."WAREHOUSE_METERING_HISTORY" C ON
C.WAREHOUSE_NAME = A.WAREHOUSE_NAME AND C.START_TIME = A.START_TIME_HOUR
)

SELECT
 CLIENT_APPLICATION_NAME
,WAREHOUSE_NAME
,SUM(APPROXIMATE_CREDITS_USED) AS APPROXIMATE_CREDITS_USED
FROM APPROXIMATE_CREDITS
GROUP BY 1,2
```

```
ORDER BY 3 DESC
;
```

Screenshot



CLIENT_APPLICATION_NAME	WAREHOUSE_NAME	APPROXIMATE_CREDITS_USED
Snowflake UI	CARLIN_TEST	58.450691666766
Snowflake UI	DJZ_WH	57.789665776081
Snowflake UI	CENG_MANUAL_CLUSTER	53.324313888000
Snowflake UI	JCRAMER_WH	43.348983612000
Snowflake UI	VLAD	39.626438141602
Go	JFIELDING_WH	37.928385557000
Snowflake UI	SNOWFLAKE_WH	24.983436389000
Snowflake UI	CHECHANI_WH	21.905146390000
Snowflake UI	JKGENOMICS_WH	16.159712193157
Snowflake UI	ADW_WH	10.795769675182
Snowflake UI	JC1	9.255301946000
Snowflake UI	SGURSOY_LOAD_WH	7.336536945000
ODBC	JKGENOMICS_WH	6.679740861843
Snowflake UI	TNORTELL_WH	6.108621388000
Go	AP_WAREHOUSE	5.868022778000
ODBC	STOM_WH	5.309183949844
Snowflake UI	RKANN_WH	4.706433055000
Snowflake UI	NEL_XL	4.659713055000

Credit Consumption by User (T1)

TIER 1

Description:

Identifies which users are consuming the most credits within your Snowflake environment.

How to Interpret Results:

Are there certain users that are consuming more credits than they should? What is the purpose behind this additional usage?

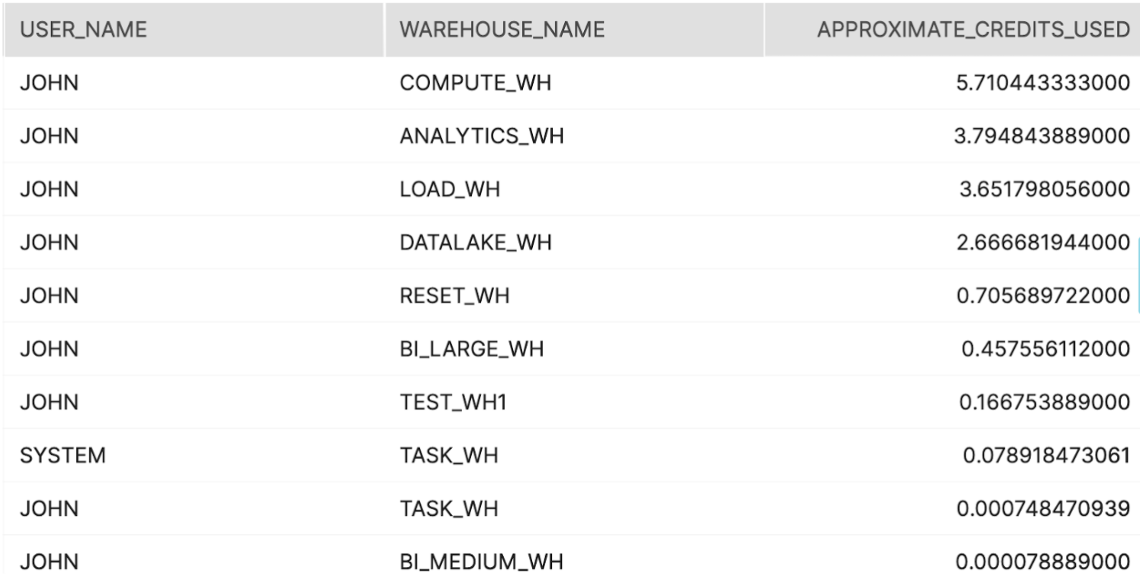
Primary Schema:

Account_Usage

SQL

```
--THIS IS APPROXIMATE CREDIT CONSUMPTION BY USER
WITH USER_HOUR_EXECUTION_CTE AS (
    SELECT  USER_NAME
    ,WAREHOUSE_NAME
    ,DATE_TRUNC('hour',START_TIME) as START_TIME_HOUR
    ,SUM(EXECUTION_TIME)  as USER_HOUR_EXECUTION_TIME
    FROM "SNOWFLAKE"."ACCOUNT_USAGE"."QUERY_HISTORY"
    WHERE WAREHOUSE_NAME IS NOT NULL
    AND EXECUTION_TIME > 0

 --Change the below filter if you want to look at a longer range than the last 1 month
    AND START_TIME > DATEADD(Month,-1,CURRENT_TIMESTAMP())
    group by 1,2,3
    )
, HOUR_EXECUTION_CTE AS (
    SELECT  START_TIME_HOUR
    ,WAREHOUSE_NAME
    ,SUM(USER_HOUR_EXECUTION_TIME) AS HOUR_EXECUTION_TIME
    FROM USER_HOUR_EXECUTION_CTE
    group by 1,2
)
, APPROXIMATE_CREDITS AS (
    SELECT
    A.USER_NAME
    ,C.WAREHOUSE_NAME
    ,(A.USER_HOUR_EXECUTION_TIME/B.HOUR_EXECUTION_TIME)*C.CREDITS_USED AS
APPROXIMATE_CREDITS_USED

    FROM USER_HOUR_EXECUTION_CTE A
    JOIN HOUR_EXECUTION_CTE B  ON A.START_TIME_HOUR = B.START_TIME_HOUR and
B.WAREHOUSE_NAME = A.WAREHOUSE_NAME
    JOIN "SNOWFLAKE"."ACCOUNT_USAGE"."WAREHOUSE_METERING_HISTORY" C ON
C.WAREHOUSE_NAME = A.WAREHOUSE_NAME AND C.START_TIME = A.START_TIME_HOUR
)

SELECT
 USER_NAME
,WAREHOUSE_NAME
,SUM(APPROXIMATE_CREDITS_USED) AS APPROXIMATE_CREDITS_USED
FROM APPROXIMATE_CREDITS
GROUP BY 1,2
ORDER BY 3 DESC
;
```

Screenshot



USER_NAME	WAREHOUSE_NAME	APPROXIMATE_CREDITS_USED
JOHN	COMPUTE_WH	5.710443333000
JOHN	ANALYTICS_WH	3.794843889000
JOHN	LOAD_WH	3.651798056000
JOHN	DATALAKE_WH	2.666681944000
JOHN	RESET_WH	0.705689722000
JOHN	BI_LARGE_WH	0.457556112000
JOHN	TEST_WH1	0.166753889000
SYSTEM	TASK_WH	0.078918473061
JOHN	TASK_WH	0.000748470939
JOHN	BI_MEDIUM_WH	0.000078889000

Queries by # of Times Executed and Execution Time (T2)

TIER 2

Description:

Are there any queries that get executed a ton?? how much execution time do they take up?

How to Interpret Results:

Opportunity to materialize the result set as a table?

Primary Schema:

Account_Usage

SQL

```
SELECT
QUERY_TEXT
,count(*) as number_of_queries
,sum(TOTAL_ELAPSED_TIME)/1000 as execution_seconds
,sum(TOTAL_ELAPSED_TIME)/(1000*60) as execution_minutes
,sum(TOTAL_ELAPSED_TIME)/(1000*60*60) as execution_hours

  from SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY Q
  where 1=1
  and TO_DATE(Q.START_TIME) >     DATEADD(month,-1,TO_DATE(CURRENT_TIMESTAMP()))
 and TOTAL_ELAPSED_TIME > 0 --only get queries that actually used compute
  group by 1
  having count(*) >= 10 --configurable/minimal threshold
  order by 2 desc
  limit 100 --configurable upper bound threshold
  ;
```

Top 50 Longest Running Queries (T2)

TIER 2

Description:

Looks at the top 50 longest running queries to see if there are patterns

How to Interpret Results:

Is there an opportunity to optimize with clustering or upsize the warehouse?

Primary Schema:

Account_Usage

SQL

```sql
select

          QUERY_ID
         --reconfigure the url if your account is not in AWS US-West

,'https://'||CURRENT_ACCOUNT()||'.snowflakecomputing.com/console#/monitoring/queries/det
queryId='||Q.QUERY_ID as QUERY_PROFILE_URL
         ,ROW_NUMBER() OVER(ORDER BY PARTITIONS_SCANNED DESC) as QUERY_ID_INT
         ,QUERY_TEXT
         ,TOTAL_ELAPSED_TIME/1000 AS QUERY_EXECUTION_TIME_SECONDS
         ,PARTITIONS_SCANNED
         ,PARTITIONS_TOTAL

from SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY Q
 where 1=1
  and TO_DATE(Q.START_TIME) >     DATEADD(month,-1,TO_DATE(CURRENT_TIMESTAMP()))
    and TOTAL_ELAPSED_TIME > 0 --only get queries that actually used compute
    and ERROR_CODE iS NULL
    and PARTITIONS_SCANNED is not null

  order by  TOTAL_ELAPSED_TIME desc

   LIMIT 50
   ;
```

Top 50 Queries that Scanned the Most Data (T2)

TIER 2

Description:

Looks at the top 50 queries that scan the largest number of micro partitions

How to Interpret Results:

Is there an opportunity to optimize with clustering or upsize the warehouse?

Primary Schema:

Account_Usage

SQL

```
select

          QUERY_ID
          --reconfigure the url if your account is not in AWS US-West

,'https://'||CURRENT_ACCOUNT()||'.snowflakecomputing.com/console#/monitoring/queries/det
queryId='||Q.QUERY_ID as QUERY_PROFILE_URL
         ,ROW_NUMBER() OVER(ORDER BY PARTITIONS_SCANNED DESC) as QUERY_ID_INT
         ,QUERY_TEXT
         ,TOTAL_ELAPSED_TIME/1000 AS QUERY_EXECUTION_TIME_SECONDS
         ,PARTITIONS_SCANNED
         ,PARTITIONS_TOTAL

from SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY Q
 where 1=1
  and TO_DATE(Q.START_TIME) >     DATEADD(month,-1,TO_DATE(CURRENT_TIMESTAMP()))
    and TOTAL_ELAPSED_TIME > 0 --only get queries that actually used compute
    and ERROR_CODE iS NULL
    and PARTITIONS_SCANNED is not null

  order by  PARTITIONS_SCANNED desc

   LIMIT 50
   ;
```

Queries by Execution Buckets over the Past 7 Days (T2)

TIER 2

Description:

Group the queries for a given warehouse by execution time buckets

How to Interpret Results:

This is an opportunity to identify query SLA trends and make a decision to downsize a warehouse, upsize a warehouse, or separate out some queries to another warehouse

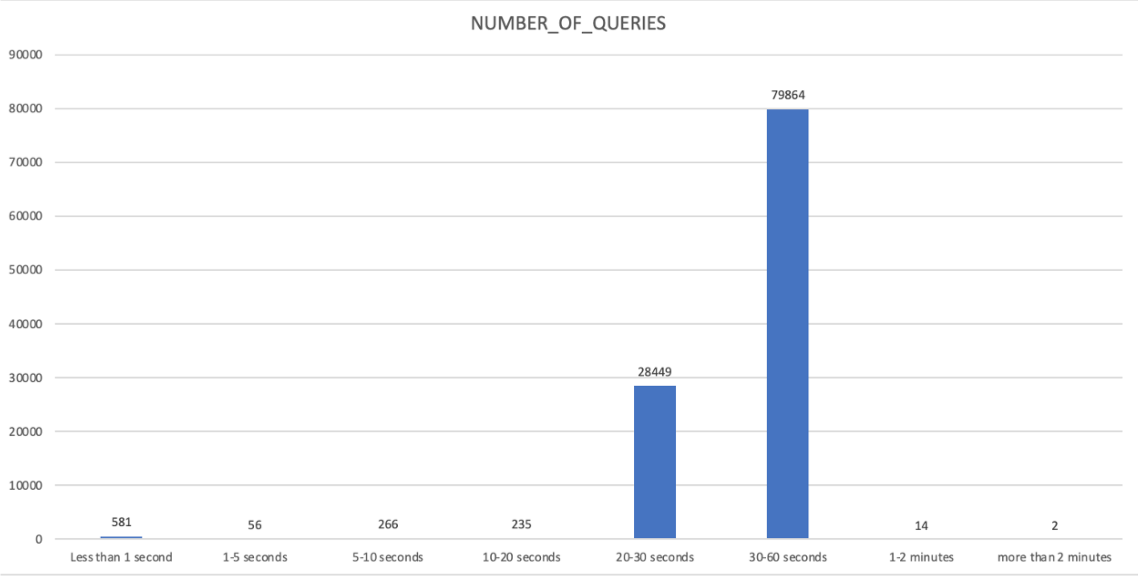
Primary Schema:

Account_Usage

SQL

```
WITH BUCKETS AS (

SELECT 'Less than 1 second' as execution_time_bucket, 0 as execution_time_lower_bound,
1000 as execution_time_upper_bound
UNION ALL
SELECT '1-5 seconds' as execution_time_bucket, 1000 as execution_time_lower_bound,
5000 as execution_time_upper_bound
UNION ALL
SELECT '5-10 seconds' as execution_time_bucket, 5000 as execution_time_lower_bound,
10000 as execution_time_upper_bound
UNION ALL
SELECT '10-20 seconds' as execution_time_bucket, 10000 as execution_time_lower_bound,
20000 as execution_time_upper_bound
UNION ALL
SELECT '20-30 seconds' as execution_time_bucket, 20000 as execution_time_lower_bound,
30000 as execution_time_upper_bound
UNION ALL
SELECT '30-60 seconds' as execution_time_bucket, 30000 as execution_time_lower_bound,
60000 as execution_time_upper_bound
UNION ALL
SELECT '1-2 minutes' as execution_time_bucket, 60000 as execution_time_lower_bound,
120000 as execution_time_upper_bound
UNION ALL
SELECT 'more than 2 minutes' as execution_time_bucket, 120000 as
execution_time_lower_bound, NULL as execution_time_upper_bound
)

SELECT
 COALESCE(execution_time_bucket,'more than 2 minutes')
,count(Query_ID) as number_of_queries

from SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY Q
FULL OUTER JOIN BUCKETS B ON (Q.TOTAL_ELAPSED_TIME) >= B.execution_time_lower_bound
and (Q.TOTAL_ELAPSED_TIME) < B.execution_time_upper_bound
where Q.Query_ID is null
OR (
TO_DATE(Q.START_TIME) >= DATEADD(week,-1,TO_DATE(CURRENT_TIMESTAMP()))
and warehouse_name = <WAREHOUSE_NAME>
and TOTAL_ELAPSED_TIME > 0
  )
group by 1,COALESCE(b.execution_time_lower_bound,120000)
order by COALESCE(b.execution_time_lower_bound,120000)
  ;
```

Screenshot



Warehouses with High Cloud Services Usage (T2)

TIER 2

Description:

Shows the warehouses that are not using enough compute to cover the cloud services portion of compute, ordered by the ratio of cloud services to total compute

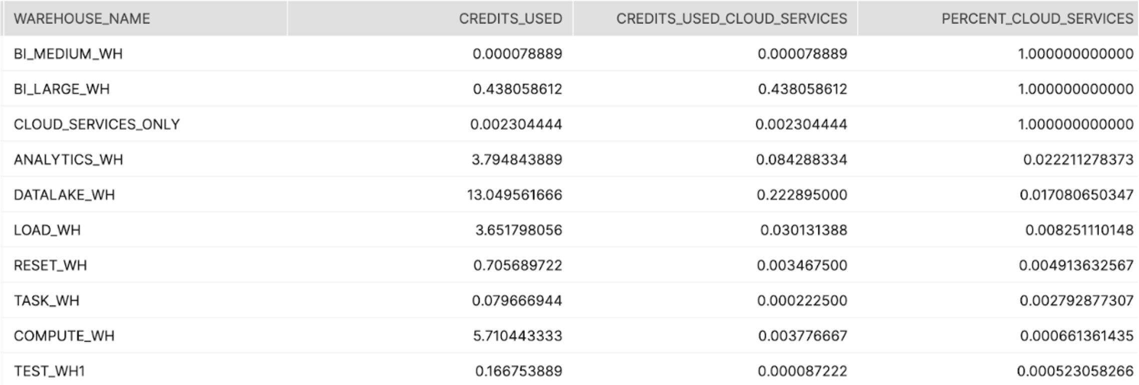
How to Interpret Results:

Focus on Warehouses that are using a high volume and ratio of cloud services compute. Investigate why this is the case to reduce overall cost (might be cloning, listing files in S3, partner tools setting session parameters, etc.). The goal to reduce cloud services credit consumption is to aim for cloud services credit to be less than 10% of overall credits. ####Primary Schema: Account_Usage

SQL

```
select
    WAREHOUSE_NAME
    ,SUM(CREDITS_USED) as CREDITS_USED
    ,SUM(CREDITS_USED_CLOUD_SERVICES) as CREDITS_USED_CLOUD_SERVICES
    ,SUM(CREDITS_USED_CLOUD_SERVICES)/SUM(CREDITS_USED) as PERCENT_CLOUD_SERVICES
from "SNOWFLAKE"."ACCOUNT_USAGE"."WAREHOUSE_METERING_HISTORY"
where TO_DATE(START_TIME) >= DATEADD(month,-1,CURRENT_TIMESTAMP())
and CREDITS_USED_CLOUD_SERVICES > 0
group by 1
order by 4 desc
;
```

Screenshot



WAREHOUSE_NAME	CREDITS_USED	CREDITS_USED_CLOUD_SERVICES	PERCENT_CLOUD_SERVICES
BI_MEDIUM_WH	0.000078889	0.000078889	1.000000000000
BI_LARGE_WH	0.438058612	0.438058612	1.000000000000
CLOUD_SERVICES_ONLY	0.002304444	0.002304444	1.000000000000
ANALYTICS_WH	3.794843889	0.084288334	0.022211278373
DATALAKE_WH	13.049561666	0.222895000	0.017080650347
LOAD_WH	3.651798056	0.030131388	0.008251110148
RESET_WH	0.705689722	0.003467500	0.004913632567
TASK_WH	0.079666944	0.000222500	0.002792877307
COMPUTE_WH	5.710443333	0.003776667	0.000661361435
TEST_WH1	0.166753889	0.000087222	0.000523058266

Warehouse Utilization (T2)

TIER 2

Description:

This query is designed to give a rough idea of how busy Warehouses are compared to the credit consumption per hour. It will show the end user the number of credits consumed, the number of queries executed and the total execution time of those queries in each hour window.

How to Interpret Results:

This data can be used to draw correlations between credit consumption and the #/duration of query executions. The more queries or higher query duration for the fewest number of credits may help drive more value per credit.

Primary Schema:

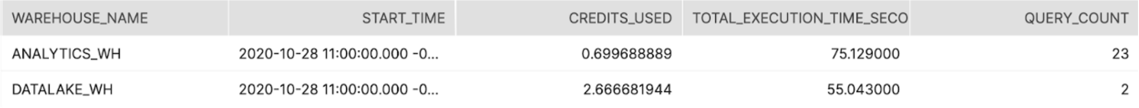
Account_Usage

SQL

```
SELECT
       WMH.WAREHOUSE_NAME
      ,WMH.START_TIME
      ,WMH.CREDITS_USED
      ,SUM(COALESCE(B.EXECUTION_TIME_SECONDS,0)) as TOTAL_EXECUTION_TIME_SECONDS
      ,SUM(COALESCE(QUERY_COUNT,0)) AS QUERY_COUNT

FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY WMH
LEFT JOIN (

      --QUERIES FULLY EXECUTED WITHIN THE HOUR
      SELECT
         WMH.WAREHOUSE_NAME
        ,WMH.START_TIME
        ,SUM(COALESCE(QH.EXECUTION_TIME,0))/(1000) AS EXECUTION_TIME_SECONDS
        ,COUNT(DISTINCT QH.QUERY_ID) AS QUERY_COUNT
      FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY     WMH
      JOIN SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY             QH ON QH.WAREHOUSE_NAME =
WMH.WAREHOUSE_NAME
                                                                          AND
```

```
QH.START_TIME BETWEEN WMH.START_TIME AND WMH.END_TIME
                                                                          AND
QH.END_TIME BETWEEN WMH.START_TIME AND WMH.END_TIME
      WHERE TO_DATE(WMH.START_TIME) >= DATEADD(week,-1,CURRENT_TIMESTAMP())
      AND TO_DATE(QH.START_TIME) >= DATEADD(week,-1,CURRENT_TIMESTAMP())
      GROUP BY
      WMH.WAREHOUSE_NAME
      ,WMH.START_TIME

      UNION ALL

      --FRONT part OF QUERIES Executed longer than 1 Hour
      SELECT
         WMH.WAREHOUSE_NAME
        ,WMH.START_TIME
        ,SUM(COALESCE(DATEDIFF(seconds,QH.START_TIME,WMH.END_TIME),0)) AS
EXECUTION_TIME_SECONDS
        ,COUNT(DISTINCT QUERY_ID) AS QUERY_COUNT
      FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY     WMH
      JOIN SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY             QH ON QH.WAREHOUSE_NAME =
WMH.WAREHOUSE_NAME
                                                                          AND
QH.START_TIME BETWEEN WMH.START_TIME AND WMH.END_TIME
                                                                          AND
QH.END_TIME > WMH.END_TIME
      WHERE TO_DATE(WMH.START_TIME) >= DATEADD(week,-1,CURRENT_TIMESTAMP())
      AND TO_DATE(QH.START_TIME) >= DATEADD(week,-1,CURRENT_TIMESTAMP())
      GROUP BY
      WMH.WAREHOUSE_NAME
      ,WMH.START_TIME

      UNION ALL

      --Back part OF QUERIES Executed longer than 1 Hour
      SELECT
         WMH.WAREHOUSE_NAME
        ,WMH.START_TIME
        ,SUM(COALESCE(DATEDIFF(seconds,WMH.START_TIME,QH.END_TIME),0)) AS
EXECUTION_TIME_SECONDS
        ,COUNT(DISTINCT QUERY_ID) AS QUERY_COUNT
      FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY     WMH
      JOIN SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY             QH ON QH.WAREHOUSE_NAME =
WMH.WAREHOUSE_NAME
                                                                          AND
QH.END_TIME BETWEEN WMH.START_TIME AND WMH.END_TIME
                                                                          AND
QH.START_TIME < WMH.START_TIME
      WHERE TO_DATE(WMH.START_TIME) >= DATEADD(week,-1,CURRENT_TIMESTAMP())
      AND TO_DATE(QH.START_TIME) >= DATEADD(week,-1,CURRENT_TIMESTAMP())
      GROUP BY
      WMH.WAREHOUSE_NAME
      ,WMH.START_TIME
      UNION ALL

      --Middle part OF QUERIES Executed longer than 1 Hour
      SELECT
         WMH.WAREHOUSE_NAME
        ,WMH.START_TIME
        ,SUM(COALESCE(DATEDIFF(seconds,WMH.START_TIME,WMH.END_TIME),0)) AS
EXECUTION_TIME_SECONDS
        ,COUNT(DISTINCT QUERY_ID) AS QUERY_COUNT
      FROM SNOWFLAKE.ACCOUNT_USAGE.WAREHOUSE_METERING_HISTORY     WMH
      JOIN SNOWFLAKE.ACCOUNT_USAGE.QUERY_HISTORY             QH ON QH.WAREHOUSE_NAME =
WMH.WAREHOUSE_NAME
                                                                          AND
WMH.START_TIME > QH.START_TIME
                                                                          AND
WMH.END_TIME < QH.END_TIME
      WHERE TO_DATE(WMH.START_TIME) >= DATEADD(week,-1,CURRENT_TIMESTAMP())
      AND TO_DATE(QH.START_TIME) >= DATEADD(week,-1,CURRENT_TIMESTAMP())
      GROUP BY
      WMH.WAREHOUSE_NAME
      ,WMH.START_TIME

) B ON B.WAREHOUSE_NAME = WMH.WAREHOUSE_NAME AND B.START_TIME = WMH.START_TIME

WHERE TO_DATE(WMH.START_TIME) >= DATEADD(week,-1,CURRENT_TIMESTAMP())
GROUP BY

      WMH.WAREHOUSE_NAME
      ,WMH.START_TIME
      ,WMH.CREDITS_USED
;
```

Screenshot



WAREHOUSE_NAME	START_TIME	CREDITS_USED	TOTAL_EXECUTION_TIME_SECO	QUERY_COUNT
ANALYTICS_WH	2020-10-28 11:00:00.000 -0...	0.699688889	75.129000	23
DATALAKE_WH	2020-10-28 11:00:00.000 -0...	2.666681944	55.043000	2