# **Compute Monitor Queries**

**Note : Flow of execution :**

Step 1 : Execute **Snowflake Prerequisite** Queries.

Step 2 : Execute **Table Creation** Queries.

Step 3 : Execute **Stored Procedures and tasks to update above table** Queries.

## **Snowflake Prerequisite Query**

Please refer **Snowflake Prerequisite Query** and execute those queries if you have not already executed them before you move on to the next steps

## **List of Tables Used**

| **Table Name** | **Dashboards** | **Latency** |
| --- | --- | --- |
| TABLE\_ROLE\_MONITOR | Detailed Compute Credit Analysis | Up to 45 minutes |
| TABLE\_USER\_MONITOR | Detailed Compute Credit Analysis | Up to 45 minutes |
| TABLE\_WAREHOUSE\_MONITOR | Compute Credit Overview,  Compute Credit Forecasting | Up To 180 minutes |
| TABLE\_PIPE\_MONITOR | Detailed Compute Credit Analysis | Up To 180 minutes |
| TABLE\_SERVERLESSTASK\_MONITOR | Detailed Compute Credit Analysis | Up To 180 minutes |
| TABLE\_REPLICATION\_MONITOR | Detailed Compute Credit Analysis | Up To 180 minutes |
| TABLE\_AUTOMATICCLUSTERING\_MONITOR | Detailed Compute Credit Analysis | Up To 180 minutes |
| TABLE\_MATERIALIZEDVIEWREFRESH\_MONITOR | Detailed Compute Credit Analysis | Up To 180 minutes |
| TABLE\_SEARCHOPTIMIZATION\_MONITOR | Detailed Compute Credit Analysis | Up To 180 minutes |

## 

## 

## **Table Creation**

**Note :** Run the below queries to make sure that you use the appropriate role, warehouse, database and schema before you create the tables and run the queries.

**QUERY :**

USE role MONITOR\_ADMIN;

USE warehouse MONITOR\_WH;

USE database MONITOR\_DB;

CREATE schema COMPUTE\_CREDIT\_MONITOR\_SCHEMA;

Use schema COMPUTE\_CREDIT\_MONITOR\_SCHEMA;

### **TABLE\_ROLE\_MONITOR**

**Note** : This table uses the view “snowflake”.”account\_usage”.”query\_history” and as per the snowflake documentation, Latency for the view may be up to 45 minutes.

Refresh Rate depends upon the scheduled time for the task credit\_role\_task which calls the stored procedure for inserting the latest data to the table.

create or replace TRANSIENT TABLE TABLE\_ROLE\_MONITOR (

DATE DATETIME,

ROLE\_NAME VARCHAR(16777216),

USER\_NAME VARCHAR(16777216),

WAREHOUSE\_NAME VARCHAR(16777216),

STMT\_CNT NUMBER(18,0),

ESTIMATED\_CREDITS NUMBER(38,12));

### **TABLE\_USER\_MONITOR**

**Note** : This table uses the view “snowflake”.”account\_usage”.”query\_history” and as per the snowflake documentation, Latency for the view may be up to 45 minutes. Refresh Rate depends upon the scheduled time for the task credit\_user\_task which calls the stored procedure for inserting the latest data to the table.

create or replace TRANSIENT TABLE TABLE\_USER\_MONITOR (

DATE DATETIME,

USER\_NAME VARCHAR(16777216),

STMT\_CNT NUMBER(18,0),

ESTIMATED\_CREDITS NUMBER(38,12));

### **TABLE\_WAREHOUSE\_MONITOR**

**Note :** This table uses the view “snowflake”.”account\_usage”.”warehouse\_metering\_history” and as per the snowflake documentation, Latency for the view may be up to 180 minutes (3 hours), except for the CREDITS\_USED\_CLOUD\_SERVICES column. Latency for CREDITS\_USED\_CLOUD\_SERVICES may be up to 6 hours.Refresh Rate depends upon the scheduled time for the task credit\_warehouse\_task which calls the stored procedure for inserting the latest data to the table.

create or replace TRANSIENT TABLE TABLE\_WAREHOUSE\_MONITOR (

WAREHOUSE\_NAME VARCHAR(16777216),

START\_DATE DATETIME,

DATE DATE,

DAILY\_CREDITS\_USED NUMBER(38,9),

DAILY\_CREDITS\_USED\_COMPUTE NUMBER(38,9),

DAILY\_CREDITS\_USED\_CLOUD NUMBER(38,9)

);

### 

### **TABLE\_PIPE\_MONITOR**

**Note :** This table uses the view “snowflake”.”account\_usage”.”pipe\_usage\_history” and as per the snowflake documentation, Latency for the view may be up to 180 minutes (3 hours). Refresh Rate depends upon the scheduled time for the task credit\_pipe\_task which calls the stored procedure for inserting the latest data to the table.

create or replace TRANSIENT TABLE TABLE\_PIPE\_MONITOR (

DATE DATETIME,

PIPE\_NAME VARCHAR(16777216),

DAILY\_CREDITS\_USED NUMBER(38,9)

);

### **TABLE\_SERVERLESSTASK\_MONITOR**

**Note :** This table uses the view “snowflake”.”account\_usage”.”serverless\_task\_history” and as per the snowflake documentation, Latency for the view may be up to 180 minutes (3 hours). Refresh Rate depends upon the scheduled time for the task credit\_stask\_task which calls the stored procedure for inserting the latest data to the table.

create or replace TRANSIENT TABLE TABLE\_SERVERLESSTASK\_MONITOR (

DATE DATETIME,

TASK\_NAME VARCHAR(16777216),

DAILY\_CREDITS\_USED NUMBER(38,9)

);

### **TABLE\_REPLICATION\_MONITOR**

**Note :** This table uses the view “snowflake”.”account\_usage”.”replication\_usage\_history” and as per the snowflake documentation, Latency for the view may be up to 180 minutes (3 hours). Refresh Rate depends upon the scheduled time for the task credit\_replication\_task which calls the stored procedure for inserting the latest data to the table.

create or replace TRANSIENT TABLE TABLE\_REPLICATION\_MONITOR (

DATE DATETIME,

DATABASE\_NAME VARCHAR(16777216),

DAILY\_CREDITS\_USED NUMBER(38,9)

);

### **TABLE\_AUTOMATICCLUSTERING\_MONITOR**

**Note :** This table uses the view “snowflake”.”account\_usage”.”automatic\_clustering\_history” and as per the snowflake documentation, Latency for the view may be up to 180 minutes (3 hours). Refresh Rate depends upon the scheduled time for the task credit\_autocluster\_task which calls the stored procedure for inserting the latest data to the table.

create or replace TRANSIENT TABLE TABLE\_AUTOMATICCLUSTERING\_MONITOR (

DATE DATETIME,

DATABASE\_NAME VARCHAR(16777216),

TABLE\_NAME VARCHAR(16777216),

DAILY\_CREDITS\_USED NUMBER(38,9)

);

### **TABLE\_MATERIALIZEDVIEWREFRESH\_MONITOR**

**Note :** This table uses the view “snowflake”.”account\_usage”.”materialized\_view\_refresh\_history” and as per the snowflake documentation, Latency for the view may be up to 180 minutes (3 hours). Refresh Rate depends upon the scheduled time for the task credit\_mview\_task which calls the stored procedure for inserting the latest data to the table.

create or replace TRANSIENT TABLE TABLE\_MATERIALIZEDVIEWREFRESH\_MONITOR (

DATE DATETIME,

DATABASE\_NAME VARCHAR(16777216),

TABLE\_NAME VARCHAR(16777216),

DAILY\_CREDITS\_USED NUMBER(38,9)

);

### **TABLE\_SEARCHOPTIMIZATION\_MONITOR**

**Note :** This table uses the view “snowflake”.”account\_usage”.”search\_optimization\_history” and as per the snowflake documentation, Latency for the view may be up to 180 minutes (3 hours). Refresh Rate depends upon the scheduled time for the task credit\_SEARCHOPTIMIZATION\_task which calls the stored procedure for inserting the latest data to the table.

create or replace TRANSIENT TABLE TABLE\_SEARCHOPTIMIZATION\_MONITOR (

DATE DATETIME,

DATABASE\_NAME VARCHAR(16777216),

TABLE\_NAME VARCHAR(16777216),

DAILY\_CREDITS\_USED NUMBER(38,9)

);

## **STORED PROCEDURES and TASKS to update above TABLES**

Note: All these stored procedures are used for incremental load. Inorder to insert all previous(historical) records you can run the following insert statements written in section **3.4.1**

### **To Load all the previous records :**

Run the below statements to insert all the previous records to the custom tables.

#### **Table\_User\_Monitor :**

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA".TABLE\_USER\_MONITOR (DATE ,

USER\_NAME ,

STMT\_CNT ,

ESTIMATED\_CREDITS)

Select convert\_timezone('UTC', START\_TIME)::datetime as date,

user\_name,

count(\*) as stmt\_cnt,

sum(execution\_time/1000 \*

case warehouse\_size

when 'X-Small' then 1/60/60

when 'Small' then 2/60/60

when 'Medium' then 4/60/60

when 'Large' then 8/60/60

when 'X-Large' then 16/60/60

when '2X-Large' then 32/60/60

when '3X-Large' then 64/60/60

when '4X-Large' then 128/60/60

else 0

end) as estimated\_credits

from snowflake.account\_usage.query\_history

group by 1,2

order by 1 desc,4 desc,2 ;

#### **Table\_Role\_Monitor :**

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA".TABLE\_ROLE\_MONITOR(DATE ,ROLE\_NAME ,USER\_NAME ,WAREHOUSE\_NAME ,STMT\_CNT ,ESTIMATED\_CREDITS)

select convert\_timezone('UTC', start\_time)::datetime as date,role\_name,user\_name,warehouse\_name ,count(\*) as stmt\_cnt,

sum(execution\_time/1000 \*

case warehouse\_size

when 'X-Small' then 1/60/60

when 'Small' then 2/60/60

when 'Medium' then 4/60/60

when 'Large' then 8/60/60

when 'X-Large' then 16/60/60

when '2X-Large' then 32/60/60

when '3X-Large' then 64/60/60

when '4X-Large' then 128/60/60

else 0

end) as estimated\_credits

from snowflake.account\_usage.query\_history

group by 1,2,3,4

order by 1 desc,4 desc,2;

#### **Table\_Warehouse\_Monitor :**

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA".TABLE\_WAREHOUSE\_MONITOR

(WAREHOUSE\_NAME,START\_DATE,DATE,DAILY\_CREDITS\_USED, DAILY\_CREDITS\_USED\_COMPUTE, DAILY\_CREDITS\_USED\_CLOUD)

SELECT WAREHOUSE\_NAME,

START\_TIME as START\_DATE,

convert\_timezone('UTC', a.start\_time)::datetime as date,

SUM(CREDITS\_USED) AS DAILY\_CREDITS\_USED,

SUM(CREDITS\_USED\_COMPUTE) AS DAILY\_CREDITS\_USED\_COMPUTE,

SUM(CREDITS\_USED\_CLOUD\_SERVICES) AS DAILY\_CREDITS\_USED\_CLOUD

FROM SNOWFLAKE.ACCOUNT\_USAGE.WAREHOUSE\_METERING\_HISTORY AS a

GROUP BY a.WAREHOUSE\_NAME, a.START\_TIME;

#### **Table\_Pipe\_Monitor :**

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA".TABLE\_PIPE\_MONITOR(DATE, PIPE\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

pipe\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."PIPE\_USAGE\_HISTORY"

group by 1,2

order by 1 desc,3 desc,2;

#### **Table\_Serverlesstask\_Monitor :**

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA"."TABLE\_SERVERLESSTASK\_MONITOR"(DATE, TASK\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

task\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."SERVERLESS\_TASK\_HISTORY"

group by 1,2

order by 1 desc,3 desc,2;

#### **Table\_Replication\_Monitor :**

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA"."TABLE\_REPLICATION\_MONITOR"(DATE, DATABASE\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

database\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."REPLICATION\_USAGE\_HISTORY"

group by 1,2

order by 1 desc,3 desc,2;

#### **Table\_Materializedviewrefresh\_Monitor:**

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA"."TABLE\_MATERIALIZEDVIEWREFRESH\_MONITOR"(DATE, DATABASE\_NAME,TABLE\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

database\_name,

table\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."MATERIALIZED\_VIEW\_REFRESH\_HISTORY"

group by 1,2,3

order by 1 desc,3 desc,2;

#### **Table\_AutomaticClustering\_Monitor :**

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA"."TABLE\_AUTOMATICCLUSTERING\_MONITOR"(DATE, DATABASE\_NAME,TABLE\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

database\_name,

table\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."AUTOMATIC\_CLUSTERING\_HISTORY"

group by 1,2,3

order by 1 desc,3 desc,2;

#### **Table\_Searchoptimization\_Monitor :**

#### 

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA"."TABLE\_SEARCHOPTIMIZATION\_MONITOR"(DATE, DATABASE\_NAME,TABLE\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

database\_name,

table\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."SEARCH\_OPTIMIZATION\_HISTORY"

group by 1,2,3

order by 1 desc,3 desc,2;

**Note** : Run the stored procedures for performing incremental load into the custom tables.

### **Stored procedure - SP\_USER\_MONITOR()**

--Stored procedure to update custom table table\_user\_monitor

CREATE OR REPLACE PROCEDURE SP\_USER\_MONITOR()

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

snowflake.execute( {sqlText: `

delete from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_USER\_MONITOR

where date = (Select max(date)

from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_USER\_MONITOR);`} );

var sql\_command = `

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA".TABLE\_USER\_MONITOR (DATE ,

USER\_NAME ,

STMT\_CNT ,

ESTIMATED\_CREDITS)

Select convert\_timezone('UTC', START\_TIME)::datetime as date,

user\_name,

count(\*) as stmt\_cnt,

sum(execution\_time/1000 \*

case warehouse\_size

when 'X-Small' then 1/60/60

when 'Small' then 2/60/60

when 'Medium' then 4/60/60

when 'Large' then 8/60/60

when 'X-Large' then 16/60/60

when '2X-Large' then 32/60/60

when '3X-Large' then 64/60/60

when '4X-Large' then 128/60/60

else 0

end) as estimated\_credits

from snowflake.account\_usage.query\_history

WHERE convert\_timezone('UTC', DATE) > (select max(convert\_timezone('UTC', DATE)) from monitor\_db.compute\_credit\_monitor\_schema.table\_user\_monitor)

group by 1,2

order by 1 desc,4 desc,2;`;

try {

snowflake.execute({sqlText: sql\_command});

return "Success";

}

catch (err) {

return "Failed" + err;

}

$$

;

call SP\_USER\_MONITOR();

**Note :** Alter the task credit\_user\_task in order to schedule the calling of the stored procedure SP\_USER\_MONITOR() as per the user requirements.

--Task to call SP\_USER\_MONITOR()

create or replace task credit\_user\_task

warehouse = MONITOR\_WH

schedule = 'USING CRON 0 3 \* \* \* UTC'

as

call SP\_USER\_MONITOR();

ALTER TASK credit\_user\_task resume;

show tasks;

### **Stored procedure - SP\_ROLE\_MONITOR()**

--Stored procedure to update custom table table\_role\_monitor

CREATE OR REPLACE PROCEDURE SP\_ROLE\_MONITOR()

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

snowflake.execute( {sqlText: `

delete from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_ROLE\_MONITOR

where date = (Select max(date)

from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_ROLE\_MONITOR);`} );

var sql\_command = `

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA".TABLE\_ROLE\_MONITOR(DATE ,ROLE\_NAME ,USER\_NAME ,WAREHOUSE\_NAME ,STMT\_CNT ,ESTIMATED\_CREDITS)

select convert\_timezone('UTC', start\_time)::datetime as date,ROLE\_NAME,user\_name,warehouse\_name ,count(\*) as stmt\_cnt,

sum(execution\_time/1000 \*

case warehouse\_size

when 'X-Small' then 1/60/60

when 'Small' then 2/60/60

when 'Medium' then 4/60/60

when 'Large' then 8/60/60

when 'X-Large' then 16/60/60

when '2X-Large' then 32/60/60

when '3X-Large' then 64/60/60

when '4X-Large' then 128/60/60

else 0

end) as estimated\_credits

from snowflake.account\_usage.query\_history

WHERE convert\_timezone('UTC', DATE) > (select max(convert\_timezone('UTC', DATE)) from monitor\_db.compute\_credit\_monitor\_schema.table\_role\_monitor)

group by 1,2,3,4

order by 1 desc,4 desc,2;`;

try {

snowflake.execute({sqlText: sql\_command});

return "Success";

}

catch (err) {

return "Failed" + err;

}

$$

;

call SP\_ROLE\_MONITOR();

**Note :** Alter the task credit\_role\_task in order to schedule the calling of the stored procedure SP\_ROLE\_MONITOR() in order to change the Refresh Rate of the tables as per the user requirements

--Task to call SP\_ROLE\_MONITOR()

create or replace task credit\_role\_task

warehouse = MONITOR\_WH

schedule = 'USING CRON 0 3 \* \* \* UTC'

as

call SP\_ROLE\_MONITOR();

ALTER TASK credit\_role\_task resume;

### **Stored procedure - SP\_WAREHOUSE\_MONITOR()**

--Stored procedure to update custom table table\_warehouse\_monitor

CREATE OR REPLACE PROCEDURE SP\_WAREHOUSE\_MONITOR()

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

snowflake.execute( {sqlText: `

delete from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_WAREHOUSE\_MONITOR

where start\_date = (Select max(start\_date)

from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_WAREHOUSE\_MONITOR);`} );

var sql\_command = `

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA".TABLE\_WAREHOUSE\_MONITOR

(WAREHOUSE\_NAME,START\_DATE,DATE,DAILY\_CREDITS\_USED, DAILY\_CREDITS\_USED\_COMPUTE, DAILY\_CREDITS\_USED\_CLOUD)

SELECT WAREHOUSE\_NAME,

START\_TIME as START\_DATE,

convert\_timezone('UTC', a.start\_time)::datetime as date,

SUM(CREDITS\_USED) AS DAILY\_CREDITS\_USED,

SUM(CREDITS\_USED\_COMPUTE) AS DAILY\_CREDITS\_USED\_COMPUTE,

SUM(CREDITS\_USED\_CLOUD\_SERVICES) AS DAILY\_CREDITS\_USED\_CLOUD

FROM SNOWFLAKE.ACCOUNT\_USAGE.WAREHOUSE\_METERING\_HISTORY AS a

WHERE convert\_timezone('UTC', start\_date) > (select max(convert\_timezone('UTC', start\_date)) from monitor\_db.compute\_credit\_monitor\_schema.table\_warehouse\_monitor)

GROUP BY a.WAREHOUSE\_NAME, a.START\_TIME;`;

try {

snowflake.execute({sqlText: sql\_command});

return "Success";

}

catch (err) {

return "Failed" + err;

}

$$

;

call SP\_WAREHOUSE\_MONITOR();

**Note :** Alter the task credit\_warehouse\_task in order to schedule the calling of the stored procedure SP\_WAREHOUSE\_MONITOR() as per the user requirements.

--Task to call SP\_WAREHOUSE\_MONITOR()

create or replace task credit\_warehouse\_task

warehouse = MONITOR\_WH

schedule = 'USING CRON 0 3 \* \* \* UTC'

as

call SP\_WAREHOUSE\_MONITOR();

ALTER TASK credit\_warehouse\_task resume;

### 

### **Stored procedure - SP\_PIPE\_MONITOR()**

--Stored procedure to update custom table table\_pipe\_monitor

CREATE OR REPLACE PROCEDURE SP\_PIPE\_MONITOR()

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

snowflake.execute( {sqlText: `

delete from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_PIPE\_MONITOR

where date = (Select max(date)

from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_PIPE\_MONITOR);`} );

var sql\_command = `

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA".TABLE\_PIPE\_MONITOR(DATE, PIPE\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

pipe\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."PIPE\_USAGE\_HISTORY"

WHERE convert\_timezone('UTC', DATE) > (select max(convert\_timezone('UTC', DATE)) from monitor\_db.compute\_credit\_monitor\_schema.table\_pipe\_monitor)

group by 1,2

order by 1 desc,3 desc,2;

`;

try {

snowflake.execute({sqlText: sql\_command});

return "Success";

}

catch (err) {

return "Failed" + err;

}

$$

;

call SP\_PIPE\_MONITOR();

**Note :** Alter the task credit\_pipe\_task in order to schedule the calling of the stored procedure SP\_PIPE\_MONITOR() as per the user requirements.

--Task to call SP\_PIPE\_MONITOR()

create or replace task credit\_pipe\_task

warehouse = MONITOR\_WH

schedule = 'USING CRON 0 3 \* \* \* UTC'

as

call SP\_PIPE\_MONITOR();

ALTER TASK credit\_pipe\_task resume;

### **Stored procedure - SP\_STASK\_MONITOR()**

--Stored procedure to update custom table table\_serverlesstask\_monitor

CREATE OR REPLACE PROCEDURE SP\_STASK\_MONITOR()

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

snowflake.execute( {sqlText: `

delete from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_SERVERLESSTASK\_MONITOR

where date = (Select max(date)

from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_SERVERLESSTASK\_MONITOR);`} );

var sql\_command = `

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA"."TABLE\_SERVERLESSTASK\_MONITOR"(DATE, TASK\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

task\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."SERVERLESS\_TASK\_HISTORY"

WHERE convert\_timezone('UTC', DATE) > (select max(convert\_timezone('UTC', DATE)) from monitor\_db.compute\_credit\_monitor\_schema.table\_serverlesstask\_monitor)

group by 1,2

order by 1 desc,3 desc,2;

`;

try {

snowflake.execute({sqlText: sql\_command});

return "Success";

}

catch (err) {

return "Failed" + err;

}

$$

;

call SP\_STASK\_MONITOR();

**Note :** Alter the task credit\_stask\_task in order to schedule the calling of the stored procedure SP\_STASK\_MONITOR() as per the user requirements.

--Task to call SP\_STASK\_MONITOR()

create or replace task credit\_stask\_task

warehouse = MONITOR\_WH

schedule = 'USING CRON 0 3 \* \* \* UTC'

as

call SP\_STASK\_MONITOR();

ALTER TASK credit\_stask\_task resume;

### **Stored procedure - SP\_REPLICATION\_MONITOR()**

--Stored procedure to update custom table\_replication\_monitor

CREATE OR REPLACE PROCEDURE SP\_REPLICATION\_MONITOR()

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

snowflake.execute( {sqlText: `

delete from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_REPLICATION\_MONITOR

where date = (Select max(date)

from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_REPLICATION\_MONITOR);`} );

var sql\_command = `

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA"."TABLE\_REPLICATION\_MONITOR"(DATE, DATABASE\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

database\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."REPLICATION\_USAGE\_HISTORY"

WHERE convert\_timezone('UTC', DATE) > (select max(convert\_timezone('UTC', DATE)) from monitor\_db.compute\_credit\_monitor\_schema.table\_replication\_monitor)

group by 1,2

order by 1 desc,3 desc,2;

`;

try {

snowflake.execute({sqlText: sql\_command});

return "Success";

}

catch (err) {

return "Failed" + err;

}

$$;

call SP\_REPLICATION\_MONITOR();

**Note :** Alter the task credit\_replication\_task in order to schedule the calling of the stored procedure SP\_REPLICATION\_MONITOR() as per the user requirements.

--Task to call SP\_REPLICATION\_MONITOR();

create or replace task credit\_replication\_task

warehouse = MONITOR\_WH

schedule = 'USING CRON 0 3 \* \* \* UTC'

as

call SP\_REPLICATION\_MONITOR();

ALTER TASK credit\_replication\_task resume;

### **Stored procedure - SP\_MVIEW\_MONITOR()**

--Stored procedure to update custom table table\_materializedviewrefresh\_monitor

CREATE OR REPLACE PROCEDURE SP\_MVIEW\_MONITOR()

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

snowflake.execute( {sqlText: `

delete from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_MATERIALIZEDVIEWREFRESH\_MONITOR

where date = (Select max(date)

from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_MATERIALIZEDVIEWREFRESH\_MONITOR);`} );

var sql\_command = `

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA"."TABLE\_MATERIALIZEDVIEWREFRESH\_MONITOR"(DATE, DATABASE\_NAME,TABLE\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

database\_name,

table\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."MATERIALIZED\_VIEW\_REFRESH\_HISTORY"

WHERE convert\_timezone('UTC', DATE) > (select max(convert\_timezone('UTC', DATE)) from monitor\_db.compute\_credit\_monitor\_schema.table\_materializedviewrefresh\_monitor)

group by 1,2,3

order by 1 desc,3 desc,2;

`;

try {

snowflake.execute({sqlText: sql\_command});

return "Success";

}

catch (err) {

return "Failed" + err;

}

$$

;

call SP\_MVIEW\_MONITOR();

**Note :** Alter the task credit\_mview\_task in order to schedule the calling of the stored procedure SP\_MVIEW\_MONITOR() as per the user requirements.

--Task to call SP\_MVIEW\_MONITOR()

create or replace task credit\_mview\_task

warehouse = MONITOR\_WH

schedule = 'USING CRON 0 3 \* \* \* UTC'

as

call SP\_MVIEW\_MONITOR();

ALTER TASK credit\_mview\_task resume;

### **Stored procedure - SP\_AUTOCLUSTER\_MONITOR()**

--Stored procedure to update custom table table\_automaticclustering\_monitor

CREATE OR REPLACE PROCEDURE SP\_AUTOCLUSTER\_MONITOR()

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

snowflake.execute( {sqlText: `

delete from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_AUTOMATICCLUSTERING\_MONITOR

where date = (Select max(date)

from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_AUTOMATICCLUSTERING\_MONITOR);`} );

var sql\_command = `

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA"."TABLE\_AUTOMATICCLUSTERING\_MONITOR"(DATE, DATABASE\_NAME,TABLE\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

database\_name,

table\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."AUTOMATIC\_CLUSTERING\_HISTORY"

WHERE convert\_timezone('UTC', DATE) > (select max(convert\_timezone('UTC', DATE)) from monitor\_db.compute\_credit\_monitor\_schema.table\_automaticclustering\_monitor)

group by 1,2,3

order by 1 desc,3 desc,2;

`;

try {

snowflake.execute({sqlText: sql\_command});

return "Success";

}

catch (err) {

return "Failed" + err;

}

$$

;

call SP\_AUTOCLUSTER\_MONITOR();

**Note :** Alter the task credit\_autocluster\_task in order to schedule the calling of the stored procedure SP\_autocluster\_MONITOR() as per the user requirements.

--Task to call SP\_autocluster\_MONITOR()

create or replace task credit\_autocluster\_task

warehouse = MONITOR\_WH

schedule = 'USING CRON 0 3 \* \* \* UTC'

as

call SP\_AUTOCLUSTER\_MONITOR();

ALTER TASK credit\_autocluster\_task resume;

### **Stored procedure - SP\_SEARCHOPTIMIZATION\_MONITOR()**

--Stored procedure to update custom table table\_searchoptimization\_monitor

CREATE OR REPLACE PROCEDURE SP\_SEARCHOPTIMIZATION\_MONITOR()

RETURNS STRING

LANGUAGE JAVASCRIPT

AS

$$

snowflake.execute( {sqlText: `

delete from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_SEARCHOPTIMIZATION\_MONITOR

where date = (Select max(date)

from MONITOR\_DB.COMPUTE\_CREDIT\_MONITOR\_SCHEMA.TABLE\_SEARCHOPTIMIZATION\_MONITOR);`} );

var sql\_command = `

insert into "MONITOR\_DB"."COMPUTE\_CREDIT\_MONITOR\_SCHEMA"."TABLE\_SEARCHOPTIMIZATION\_MONITOR"(DATE, DATABASE\_NAME,TABLE\_NAME, DAILY\_CREDITS\_USED)

select convert\_timezone('UTC', start\_time)::datetime as date,

database\_name,

table\_name,

sum(credits\_used) as DAILY\_CREDITS\_USED

from "SNOWFLAKE"."ACCOUNT\_USAGE"."SEARCH\_OPTIMIZATION\_HISTORY"

WHERE convert\_timezone('UTC', DATE) > (select max(convert\_timezone('UTC', DATE)) from monitor\_db.compute\_credit\_monitor\_schema.table\_searchoptimization\_monitor)

group by 1,2,3

order by 1 desc,3 desc,2;

`;

try {

snowflake.execute({sqlText: sql\_command});

return "Success";

}

catch (err) {

return "Failed" + err;

}

$$

;

call SP\_SEARCHOPTIMIZATION\_MONITOR();

**Note :** Alter the task credit\_SEARCHOPTIMIZATION\_task in order to schedule the calling of the stored procedure SP\_SEARCHOPTIMIZATION\_MONITOR() as per the user requirements.

--Task to call SP\_SEARCHOPTIMIZATION\_MONITOR()

create or replace task credit\_SEARCHOPTIMIZATION\_task

warehouse = MONITOR\_WH

schedule = 'USING CRON 0 3 \* \* \* UTC'

as

call SP\_SEARCHOPTIMIZATION\_MONITOR();

ALTER TASK credit\_SEARCHOPTIMIZATION\_task resume;

# **Best Practices :-**

* It is mandatory to execute all the insert statements for historical load wherever required before executing SPs for incremental load of data.
* Always run historical load commands only once. If in any case you want to run it again then first truncate the respective custom table.
* You can always schedule the task as per your requirement . Reference for setting up task using cron expression - <https://docs.snowflake.com/en/sql-reference/sql/create-task.html#examples>
* If you want to customize bad query table results then always start by replacing the old table and do historical and incremental load again.