use role accountadmin;

--Initialize tag\_admin variables ------------------------------------------------------

set tag\_admin\_warehouse = 'ADMIN\_WH';

set tag\_admin\_db = 'ADMIN\_DB';

set tag\_admin\_schema = $tag\_admin\_db||'.TAG\_MASTER';

--set tag\_admin\_schema = 'ADMIN\_DB.TAG\_MASTER';

set tag\_admin\_role\_name = 'TAG\_ADMIN';

set tag\_admin\_user\_name = 'TAG\_ADMIN\_USER';

set tag\_admin\_user\_pwd = 'P@$$w0&d';

------------------------------------------------------------------------------

--Initialize app1 variables ------------------------------------------------------

set app1\_warehouse = 'READ\_WH';

set app1\_db = 'APP1\_DB'; --name that will be given to new clone for testing

set app1\_admin\_schema = $app1\_db||'.PUBLIC';

--set app1\_admin\_schema = 'APP1\_DB.PUBLIC';

set app1\_admin\_role\_name = $app1\_db||'\_ADMIN';

--set app1\_admin\_role\_name = 'APP1\_DB\_ADMIN';

set app1\_admin\_user\_name = $app1\_db||'\_ADMIN\_USER';

--set app1\_admin\_user\_name = 'APP1\_DB\_ADMIN\_USER';

set app1\_admin\_user\_pwd = 'P@$$w0&d';

set app1\_user\_role\_name = $app1\_db||'\_USER';

--set app1\_user\_role\_name = 'APP1\_DB\_USER';

set app1\_user\_name = $app1\_db||'\_USER1';

--set app1\_user\_name = 'APP1\_DB\_USER1';

set app1\_user\_pwd = 'P@$$w0&d';

------------------------------------------------------------------------------

--Initialize app2 variables ------------------------------------------------------

set app2\_warehouse = 'READ\_WH';

set app2\_db = 'APP2\_DB'; --name that will be given to new clone for testing

set app2\_admin\_schema = $app2\_db||'.PUBLIC';

--set app2\_admin\_schema = 'APP2\_DB.PUBLIC';

set app2\_admin\_role\_name = $app2\_db||'\_ADMIN';

--set app2\_admin\_role\_name = 'APP2\_DB\_ADMIN';

set app2\_admin\_user\_name = $app2\_db||'\_ADMIN\_USER';

--set app2\_admin\_user\_name = 'APP2\_DB\_ADMIN\_USER';

set app2\_admin\_user\_pwd = 'P@$$w0&d';

set app2\_user\_role\_name = $app2\_db||'\_USER';

--set app2\_user\_role\_name = 'APP2\_DB\_USER';

set app2\_user\_name = $app2\_db||'\_USER1';

--set app2\_user\_name = 'APP2\_DB\_USER1';

set app2\_user\_pwd = 'P@$$w0&d';

-------------------------------------------------------------------------------

/\* clean up from prior run \*/

DROP DATABASE IF EXISTS identifier($tag\_admin\_db);

--DROP DATABASE IF EXISTS ADMIN\_DB;

DROP DATABASE IF EXISTS identifier($app1\_db);

--APP1\_DB

DROP DATABASE IF EXISTS identifier($app2\_db);

--APP2\_DB

DROP ROLE IF EXISTS identifier($tag\_admin\_role\_name);

--TAG\_ADMIN

DROP ROLE IF EXISTS identifier($app1\_admin\_role\_name);

--APP1\_DB\_ADMIN

DROP ROLE IF EXISTS identifier($app2\_admin\_role\_name);

--APP2\_DB\_ADMIN

DROP USER IF EXISTS identifier($tag\_admin\_user\_name);

--TAG\_ADMIN\_USER

DROP USER IF EXISTS identifier($app1\_admin\_user\_name);

--APP1\_DB\_ADMIN\_USER

DROP USER IF EXISTS identifier($app2\_admin\_user\_name);

--APP2\_DB\_ADMIN\_USER

DROP USER IF EXISTS identifier($app1\_user\_name);

--APP1\_DB\_USER1

DROP USER IF EXISTS identifier($app2\_user\_name);

--APP2\_DB\_USER1

------------------------------------------------------------------------------

--Create Warehouses

CREATE OR REPLACE WAREHOUSE identifier($tag\_admin\_warehouse) WITH WAREHOUSE\_SIZE = 'XSMALL' WAREHOUSE\_TYPE = 'STANDARD' AUTO\_SUSPEND = 300 AUTO\_RESUME = TRUE MIN\_CLUSTER\_COUNT = 1 MAX\_CLUSTER\_COUNT = 2 SCALING\_POLICY = 'STANDARD' COMMENT = 'Warehouse for DB1 users.';

--ADMIN\_WH

CREATE OR REPLACE WAREHOUSE identifier($app1\_warehouse) WITH WAREHOUSE\_SIZE = 'SMALL' WAREHOUSE\_TYPE = 'STANDARD' AUTO\_SUSPEND = 300 AUTO\_RESUME = TRUE MIN\_CLUSTER\_COUNT = 1 MAX\_CLUSTER\_COUNT = 2 SCALING\_POLICY = 'STANDARD' COMMENT = 'Warehouse for DB1 users.';

--READ\_WH

CREATE OR REPLACE WAREHOUSE identifier($app2\_warehouse) WITH WAREHOUSE\_SIZE = 'SMALL' WAREHOUSE\_TYPE = 'STANDARD' AUTO\_SUSPEND = 300 AUTO\_RESUME = TRUE MIN\_CLUSTER\_COUNT = 1 MAX\_CLUSTER\_COUNT = 2 SCALING\_POLICY = 'STANDARD' COMMENT = 'Warehouse for DB1 users.';

--READ\_WH

------------------------------------------------------------------------------

--Create Databases and Schemas

CREATE OR REPLACE DATABASE identifier($tag\_admin\_db) COMMENT = 'Admin DB';

--ADMIN\_DB

CREATE SCHEMA if not exists identifier($tag\_admin\_schema) COMMENT = 'Tag admin schema';

--TAG\_MASTER

CREATE DATABASE if not exists identifier($app1\_db) COMMENT = 'Application1 DB';

--APP1\_DB

CREATE DATABASE if not exists identifier($app2\_db) COMMENT = 'Application2 DB';

--APP2\_DB

---------------------------------------------------------------------------------

--Create Admin Roles and apply Grants

---------------------------------------------------------------------------------

--Tag Admin

CREATE or REPLACE ROLE identifier($tag\_admin\_role\_name) COMMENT = 'Tag Admin for account';

--TAG\_ADMIN

GRANT ROLE identifier($tag\_admin\_role\_name) TO ROLE "SYSADMIN";

--TAG\_ADMIN

grant imported privileges on database snowflake to role identifier($tag\_admin\_role\_name);

--TAG\_ADMIN

--App1 Admin roles to own DBs

CREATE or REPLACE ROLE identifier($app1\_admin\_role\_name) COMMENT = 'App1 Admin';

--APP1\_DB\_ADMIN

GRANT ROLE identifier($app1\_admin\_role\_name) TO ROLE "SYSADMIN";

--APP1\_DB\_ADMIN

grant imported privileges on database snowflake to role identifier($app1\_admin\_role\_name);

--APP1\_DB\_ADMIN

grant imported privileges on database SNOWFLAKE\_SAMPLE\_DATA to role identifier($app1\_admin\_role\_name);

--APP1\_DB\_ADMIN

--App2 Admin roles to own DBs

CREATE or REPLACE ROLE identifier($app2\_admin\_role\_name) COMMENT = 'App2 Admin';

--APP2\_DB\_ADMIN

GRANT ROLE identifier($app2\_admin\_role\_name) TO ROLE "SYSADMIN";

--APP2\_DB\_ADMIN

grant imported privileges on database snowflake to role identifier($app2\_admin\_role\_name);

--APP2\_DB\_ADMIN

grant imported privileges on database SNOWFLAKE\_SAMPLE\_DATA to role identifier($app2\_admin\_role\_name);

--APP2\_DB\_ADMIN

---------------------------------------------------------------------------------

--Create Admin Users and apply Grants

---------------------------------------------------------------------------------

--Create App Admin users

create or replace user identifier($app1\_admin\_user\_name) password=$app1\_admin\_user\_pwd default\_role = $app1\_admin\_role\_name default\_secondary\_roles = ('ALL') must\_change\_password = false;

--APP1\_DB\_ADMIN\_USER

create or replace user identifier($app2\_admin\_user\_name) password=$app2\_admin\_user\_pwd default\_role = $app2\_admin\_role\_name default\_secondary\_roles = ('ALL') must\_change\_password = false;

--APP2\_DB\_ADMIN\_USER

--Grant Admin Users to Admin Roles

GRANT ROLE identifier($app1\_admin\_role\_name) TO USER identifier($app1\_admin\_user\_name);

--GRANT ROLE APP1\_DB\_ADMIN to USER APP1\_DB\_ADMIN\_USER;

GRANT ROLE identifier($app2\_admin\_role\_name) TO USER identifier($app2\_admin\_user\_name);

--GRANT ROLE APP2\_DB\_ADMIN to USER APP2\_DB\_ADMIN\_USER;

--Grant ownership of DBs to Admin Roles

GRANT OWNERSHIP ON DATABASE identifier($app1\_db) TO ROLE identifier($app1\_admin\_role\_name);

--GRANT OWNERSHIP ON DATABASE APP1\_DB TO ROLE APP1\_DB\_ADMIN;

GRANT OWNERSHIP ON DATABASE identifier($app2\_db) TO ROLE identifier($app2\_admin\_role\_name);

--GRANT OWNERSHIP ON DATABASE APP2\_DB TO ROLE APP2\_DB\_ADMIN;

--Grant USAGE of all schemas in DBs to Admin Roles

GRANT OWNERSHIP ON ALL SCHEMAS IN DATABASE identifier($app1\_db) TO ROLE identifier($app1\_admin\_role\_name);

GRANT OWNERSHIP ON ALL SCHEMAS IN DATABASE identifier($app2\_db) TO ROLE identifier($app2\_admin\_role\_name);

--Grant ALL of DBs tables to Admin Roles

GRANT ALL ON ALL TABLES IN DATABASE identifier($app1\_db) TO ROLE identifier($app1\_admin\_role\_name);

GRANT ALL ON ALL TABLES IN DATABASE identifier($app2\_db) TO ROLE identifier($app2\_admin\_role\_name);

--Grant ALL of DBs future tables to Admin Roles

GRANT ALL ON FUTURE TABLES IN DATABASE identifier($app1\_db) TO ROLE identifier($app1\_admin\_role\_name);

GRANT ALL ON FUTURE TABLES IN DATABASE identifier($app2\_db) TO ROLE identifier($app2\_admin\_role\_name);

--Grant ALL of DBs views to Admin Roles

GRANT ALL ON ALL VIEWS IN DATABASE identifier($app1\_db) TO ROLE identifier($app1\_admin\_role\_name);

GRANT ALL ON ALL VIEWS IN DATABASE identifier($app2\_db) TO ROLE identifier($app2\_admin\_role\_name);

--Grant ALL of DBs future views to Admin Roles

GRANT ALL ON FUTURE VIEWS IN DATABASE identifier($app1\_db) TO ROLE identifier($app1\_admin\_role\_name);

GRANT ALL ON FUTURE VIEWS IN DATABASE identifier($app2\_db) TO ROLE identifier($app2\_admin\_role\_name);

--Grant USAGE on WHs to Admin Roles

GRANT USAGE ON WAREHOUSE identifier($app1\_warehouse) to ROLE identifier($app1\_admin\_role\_name);

GRANT USAGE ON WAREHOUSE identifier($app2\_warehouse) to ROLE identifier($app2\_admin\_role\_name);

--grant tag privileges needed for creating and applying tags

grant create tag on schema identifier($app1\_admin\_schema) to role identifier($app1\_admin\_role\_name);

grant create tag on schema identifier($app2\_admin\_schema) to role identifier($app2\_admin\_role\_name);

---------------------------------------------------------------------------------

--Create Tag Admin and apply Grants

---------------------------------------------------------------------------------

--Create Tag Admin user

create or replace user identifier($tag\_admin\_user\_name) password=$tag\_admin\_user\_pwd default\_role = $tag\_admin\_role\_name default\_secondary\_roles = ('ALL') must\_change\_password = false;

--TAG\_ADMIN\_USER

GRANT ROLE identifier($tag\_admin\_role\_name) TO USER identifier($tag\_admin\_user\_name); --TAG\_ADMIN granted to TAG\_ADMIN\_USER

--grant the tag privileges needed for creating and applying tags

grant create tag on schema identifier($tag\_admin\_schema) to role identifier($tag\_admin\_role\_name);

grant apply tag on account to role identifier($tag\_admin\_role\_name);

--This needs to happen after ownership changes above

--Grant USAGE of DB objects to Tag\_Admin

GRANT USAGE ON DATABASE identifier($tag\_admin\_db) TO ROLE identifier($tag\_admin\_role\_name);

GRANT USAGE ON DATABASE identifier($app1\_db) TO ROLE identifier($tag\_admin\_role\_name);

GRANT USAGE ON DATABASE identifier($app2\_db) TO ROLE identifier($tag\_admin\_role\_name);

--Grant USAGE of tag admin schema to tag admin Roles

GRANT USAGE ON SCHEMA identifier($tag\_admin\_schema) TO ROLE identifier($tag\_admin\_role\_name);

GRANT USAGE ON ALL SCHEMAS IN DATABASE identifier($app1\_db) TO ROLE identifier($tag\_admin\_role\_name);

GRANT USAGE ON ALL SCHEMAS IN DATABASE identifier($app2\_db) TO ROLE identifier($tag\_admin\_role\_name);

--Grant USAGE on WHs to tag admin

GRANT USAGE ON WAREHOUSE identifier($tag\_admin\_warehouse) to ROLE identifier($tag\_admin\_role\_name);

---------------------------------------------------------------------------------

--Create App users and apply Grants

---------------------------------------------------------------------------------

--Create Read-only User roles

CREATE or REPLACE ROLE identifier($app1\_user\_role\_name) COMMENT = 'App1 DB Users'; --APP1\_DB\_USER

CREATE or REPLACE ROLE identifier($app2\_user\_role\_name) COMMENT = 'App2 DB Users'; --APP2\_DB\_USER

--Grant users to roles

GRANT ROLE identifier($app1\_user\_role\_name) TO ROLE identifier($app1\_admin\_role\_name);

GRANT ROLE identifier($app2\_user\_role\_name) TO ROLE identifier($app2\_admin\_role\_name);

--Create test users

create or replace user identifier($app1\_user\_name) password = $app1\_user\_pwd default\_role = $app1\_user\_role\_name must\_change\_password = false;

--APP1\_DB\_USER1

create or replace user identifier($app2\_user\_name) password = $app2\_user\_pwd default\_role = $app2\_user\_role\_name must\_change\_password = false;

--APP2\_DB\_USER1

--Grant User Roles to Users

GRANT ROLE identifier($app1\_user\_role\_name) TO USER identifier($app1\_user\_name);

GRANT ROLE identifier($app2\_user\_role\_name) TO USER identifier($app2\_user\_name);

--Grant USAGE of DBs to User Roles

GRANT USAGE ON DATABASE identifier($app1\_db) TO ROLE identifier($app1\_user\_role\_name);

GRANT USAGE ON DATABASE identifier($app2\_db) TO ROLE identifier($app2\_user\_role\_name);

--Grant USAGE of all schemas in DBs to User Roles

GRANT USAGE ON ALL SCHEMAS IN DATABASE identifier($app1\_db) TO ROLE identifier($app1\_user\_role\_name);

GRANT USAGE ON ALL SCHEMAS IN DATABASE identifier($app2\_db) TO ROLE identifier($app2\_user\_role\_name);

--Grant SELECT of DBs tables to User Roles

GRANT SELECT ON ALL TABLES IN DATABASE identifier($app1\_db) TO ROLE identifier($app1\_user\_role\_name);

GRANT SELECT ON ALL TABLES IN DATABASE identifier($app2\_db) TO ROLE identifier($app2\_user\_role\_name);

--Grant SELECT of DBs views to User Roles

GRANT SELECT ON ALL VIEWS IN DATABASE identifier($app1\_db) TO ROLE identifier($app1\_user\_role\_name);

GRANT SELECT ON ALL VIEWS IN DATABASE identifier($app2\_db) TO ROLE identifier($app2\_user\_role\_name);

--Grant SELECT of DBs future tables to User Roles

GRANT SELECT ON FUTURE TABLES IN DATABASE identifier($app1\_db) TO ROLE identifier($app1\_user\_role\_name);

GRANT SELECT ON FUTURE TABLES IN DATABASE identifier($app2\_db) TO ROLE identifier($app2\_user\_role\_name);

--Grant SELECT of DBs future views to User Roles

GRANT SELECT ON FUTURE VIEWS IN DATABASE identifier($app1\_db) TO ROLE identifier($app1\_user\_role\_name);

GRANT SELECT ON FUTURE VIEWS IN DATABASE identifier($app2\_db) TO ROLE identifier($app2\_user\_role\_name);

--Grant USAGE on WHs to User Roles

GRANT USAGE ON WAREHOUSE identifier($app1\_warehouse) to ROLE identifier($app1\_user\_role\_name);

GRANT USAGE ON WAREHOUSE identifier($app2\_warehouse) to ROLE identifier($app2\_user\_role\_name);

--------------------------------------------------------------------------------------------------------------------------------------

--Create and set App1 tags

--------------------------------------------------------------------------------------------------------------------------------------

use role identifier($app1\_admin\_role\_name);

use warehouse identifier($app1\_warehouse);

use schema identifier($app1\_admin\_schema);

CREATE TAG IF NOT EXISTS DATA\_SENSITIVITY ALLOWED\_VALUES 'HR', 'PII', 'OPEN';

ALTER SCHEMA identifier($app1\_admin\_schema) SET TAG DATA\_SENSITIVITY='OPEN';

show tags in account;

select \* from table(information\_schema.tag\_references($app1\_admin\_schema, 'schema'));

--------------------------------------------------------------------------------------------------------------------------------------

--Create and set App2 tags

--------------------------------------------------------------------------------------------------------------------------------------

use role identifier($app2\_admin\_role\_name);

use warehouse identifier($app2\_warehouse);

use schema identifier($app2\_admin\_schema);

CREATE TAG IF NOT EXISTS DATA\_SENSITIVITY ALLOWED\_VALUES 'HR', 'PII', 'OPEN';

ALTER SCHEMA identifier($app2\_admin\_schema) SET TAG DATA\_SENSITIVITY='PII';

--DROP TAG DATA\_SENSITIVITY;

show tags in account;

select \* from table(information\_schema.tag\_references($app2\_admin\_schema, 'schema'));

--------------------------------------------------------------------------------------------------------------------------------------

--Create and set master tags

--------------------------------------------------------------------------------------------------------------------------------------

use role identifier($tag\_admin\_role\_name); --TAG\_ADMIN

use warehouse identifier($tag\_admin\_warehouse); --TAG\_ADMIN

use schema identifier($tag\_admin\_schema); --ADMIB\_DB.TAG\_MASTER

--Show that tag admin has limited privileges to operate on DBs.

--ALTER DATABASE identifier($app1\_db) SET DATA\_RETENTION\_TIME\_IN\_DAYS=7;

--Create master tags

CREATE TAG IF NOT EXISTS COST\_CENTER ALLOWED\_VALUES 'ADMIN', 'APP1', 'APP2';

CREATE TAG IF NOT EXISTS ENVIRONMENT ALLOWED\_VALUES 'DEV', 'TST', 'PRD';

--SET DATABASE ENVIRONMENT TAG

ALTER DATABASE identifier($tag\_admin\_db) SET TAG ENVIRONMENT='DEV';

ALTER DATABASE identifier($app1\_db) SET TAG ENVIRONMENT='DEV';

ALTER DATABASE identifier($app2\_db) SET TAG ENVIRONMENT='DEV';

--SET DATABASE COST\_CENTER TAG

ALTER DATABASE identifier($tag\_admin\_db) SET TAG COST\_CENTER='ADMIN';

ALTER DATABASE identifier($app1\_db) SET TAG COST\_CENTER='APP1';

ALTER DATABASE identifier($app2\_db) SET TAG COST\_CENTER='APP2';

--SET ROLE COST\_CENTER TAG

ALTER ROLE identifier($tag\_admin\_role\_name) SET TAG COST\_CENTER='ADMIN';

ALTER ROLE identifier($app1\_admin\_role\_name) SET TAG COST\_CENTER='APP1';

ALTER ROLE identifier($app2\_admin\_role\_name) SET TAG COST\_CENTER='APP2';

ALTER ROLE identifier($app1\_user\_role\_name) SET TAG COST\_CENTER='APP1';

ALTER ROLE identifier($app2\_user\_role\_name) SET TAG COST\_CENTER='APP2';

--View App Tags as Tag Admin

show tags in account;

select \* from table(app1\_db.information\_schema.tag\_references($app1\_admin\_schema, 'schema'));

select \* from table(app2\_db.information\_schema.tag\_references($app2\_admin\_schema, 'schema'));

--select \* from snowflake.account\_usage.tag\_references; NOTE: this can work but has a lag behind info schema

--------------------------------------------------------------------------------------------------------------------------------------

--Create App1 data

--------------------------------------------------------------------------------------------------------------------------------------

use role identifier($app1\_admin\_role\_name); --APP1\_DB\_ADMIN

use warehouse identifier($app1\_warehouse); --READ\_WH

use schema identifier($app1\_admin\_schema); --APP1\_DB.PUBLIC

create or replace table part as select \* from "SNOWFLAKE\_SAMPLE\_DATA"."TPCH\_SF1"."PART";

--------------------------------------------------------------------------------------------------------------------------------------

--Create App2 data

--------------------------------------------------------------------------------------------------------------------------------------

use role identifier($app2\_admin\_role\_name); --APP2\_DB\_ADMIN

use warehouse identifier($app2\_warehouse); -- READ\_WH

use schema identifier($app2\_admin\_schema); -- APP2\_DB.PUBLIC

create or replace table part as select \* from "SNOWFLAKE\_SAMPLE\_DATA"."TPCH\_SF10"."PART";

--------------------------------------------------------------------------------------------------------------------------------------

--Query App1 data

--------------------------------------------------------------------------------------------------------------------------------------

use role identifier($app1\_user\_role\_name); --APP1\_DB\_USER

use warehouse identifier($app1\_warehouse); --READ\_WH

use schema identifier($app1\_admin\_schema); --APP1\_DB.PUBLIC

select \* from PART order by P\_RETAILPRICE desc limit 100;

--------------------------------------------------------------------------------------------------------------------------------------

--Query App2 data

--------------------------------------------------------------------------------------------------------------------------------------

use role identifier($app2\_user\_role\_name); --APP2\_DB\_USER

use warehouse identifier($app2\_warehouse); --READ\_WH

use schema identifier($app2\_admin\_schema); --APP2\_DB.PUBLIC

select \* from PART order by P\_RETAILPRICE desc limit 100;

--------------------------------------------------------------------------------------------------------------------------------------

--View Usage Stats

--------------------------------------------------------------------------------------------------------------------------------------

use role accountadmin;

show tags in account;

use schema identifier($tag\_admin\_schema); --ADMIB\_DB.TAG\_MASTER

--WH credits used is here

select \* from table(app1\_db.information\_schema.warehouse\_metering\_history(dateadd('days',-1,current\_date())));

--Calculate credit usage per query

--Note: time spent on activities not associated with user credit usage is taken out

//check this query in a day due to latency

WITH CTE\_QH AS (

SELECT QH.START\_TIME

,QH.END\_TIME

,QH.TOTAL\_ELAPSED\_TIME - (QH.LIST\_EXTERNAL\_FILES\_TIME+QH.COMPILATION\_TIME+QH.QUEUED\_PROVISIONING\_TIME+QH.QUEUED\_REPAIR\_TIME+QH.QUEUED\_OVERLOAD\_TIME+QH.TRANSACTION\_BLOCKED\_TIME) AS WH\_TIME

,DATEADD(MILLISECOND, -WH\_TIME, QH.END\_TIME) AS START\_TIME\_WH

,QH.WAREHOUSE\_ID

,QH.WAREHOUSE\_NAME

,QH.WAREHOUSE\_SIZE

,QH.WAREHOUSE\_TYPE

,QH.CLUSTER\_NUMBER

,QH.QUERY\_ID

,QH.QUERY\_TAG

,QH.SESSION\_ID

,QH.USER\_NAME

,NVL(TR.TAG\_VALUE,'ADMIN') as COST\_CENTER

,QH.ROLE\_NAME

,QH.DATABASE\_ID

,QH.DATABASE\_NAME

,QH.SCHEMA\_ID

,QH.SCHEMA\_NAME

,QH.QUERY\_TYPE

,QH.EXECUTION\_STATUS

,QH.ERROR\_CODE

,QH.ERROR\_MESSAGE

FROM SNOWFLAKE.ACCOUNT\_USAGE.QUERY\_HISTORY QH

LEFT JOIN SNOWFLAKE.ACCOUNT\_USAGE.TAG\_REFERENCES TR on tr.object\_name = qh.role\_name

and tr.TAG\_NAME='COST\_CENTER'

and tr.DOMAIN='ROLE'

WHERE 1=1

and QH.CLUSTER\_NUMBER IS NOT NULL

--AND START\_TIME = :daterange

and QH.WAREHOUSE\_NAME NOT IN ('COMPUTE\_SERVICE\_WH', 'CLOUD\_SERVICES\_ONLY')

)

,CTE\_WMH AS (

SELECT START\_TIME

,END\_TIME

,WAREHOUSE\_ID

,WAREHOUSE\_NAME

,CREDITS\_USED\_COMPUTE

,sum(CREDITS\_USED\_COMPUTE) over (partition by warehouse\_id, warehouse\_name) TOTAL\_CREDITS\_USED\_COMPUTE\_WMH

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

WHERE 1=1

--and START\_TIME = :daterange

and WAREHOUSE\_NAME NOT IN ('COMPUTE\_SERVICE\_WH', 'CLOUD\_SERVICES\_ONLY')

)

,CTE\_OVERLAP AS (

SELECT WMH.WAREHOUSE\_ID

,WMH.WAREHOUSE\_NAME

,GREATEST(WMH.START\_TIME, IFNULL(QH.START\_TIME\_WH, WMH.START\_TIME)) AS START\_TIME\_SEGMENT

,LEAST(WMH.END\_TIME, IFNULL(QH.END\_TIME, WMH.END\_TIME)) AS END\_TIME\_SEGMENT

,WMH.CREDITS\_USED\_COMPUTE \*

(

DATEDIFF(MILLISECOND, START\_TIME\_SEGMENT, END\_TIME\_SEGMENT) /

SUM(DATEDIFF(MILLISECOND, START\_TIME\_SEGMENT, END\_TIME\_SEGMENT)) OVER (PARTITION BY WMH.WAREHOUSE\_ID, WMH.START\_TIME)

) AS SEGMENT\_CREDITS

,QH.QUERY\_ID

,QH.WAREHOUSE\_SIZE

,QH.WAREHOUSE\_TYPE

,QH.CLUSTER\_NUMBER

,QH.QUERY\_TAG

,QH.SESSION\_ID

,QH.USER\_NAME

,QH.ROLE\_NAME

,QH.COST\_CENTER

,QH.DATABASE\_ID

,QH.DATABASE\_NAME

,QH.SCHEMA\_ID

,QH.SCHEMA\_NAME

,QH.QUERY\_TYPE

,QH.EXECUTION\_STATUS

,QH.ERROR\_CODE

,QH.ERROR\_MESSAGE

,WMH.CREDITS\_USED\_COMPUTE CREDITS\_USED\_COMPUTE\_WMH

,WMH.TOTAL\_CREDITS\_USED\_COMPUTE\_WMH

FROM CTE\_WMH WMH

LEFT JOIN CTE\_QH QH

ON WMH.WAREHOUSE\_ID = QH.WAREHOUSE\_ID

AND WMH.START\_TIME < QH.END\_TIME

AND QH.START\_TIME\_WH < WMH.END\_TIME

),

CTE\_AGG\_COST as (

SELECT

--:datebucket(OV.START\_TIME\_SEGMENT) usage\_date,

OV.START\_TIME\_SEGMENT::date as usage\_date,

OV.COST\_CENTER,

OV.warehouse\_id,

OV.warehouse\_name,

OV.user\_name,

OV.role\_name,

count(OV.QUERY\_ID) queries,

sum(OV.SEGMENT\_CREDITS) CREDITS\_USED

FROM CTE\_OVERLAP OV

group by usage\_date, cost\_center, warehouse\_id, warehouse\_name, user\_name, role\_name, TOTAL\_CREDITS\_USED\_COMPUTE\_WMH

)

SELECT

\*

FROM CTE\_AGG\_COST

WHERE COST\_CENTER != 'ADMIN' --this include services that would be better tracked separately

HAVING QUERIES > 0

ORDER BY USAGE\_DATE DESC, COST\_CENTER

;