# CONTROL\_TABLE

create or replace TRANSIENT TABLE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.CONTROL\_TABLE (

KEY VARCHAR(250),

VALUE TIMESTAMP\_NTZ(9)

)

COMMENT='stores key info used by the query analyzer report procedure' ;



# DURATION\_FILTER

create or replace TRANSIENT TABLE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.DURATION\_FILTER (

DURATION NUMBER(38,0)

)

COMMENT='custom table to implement duration filter'

;



# MISSING\_QIDS

create or replace TRANSIENT TABLE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.MISSING\_QIDS (

QUERY\_ID VARCHAR(16777216)

)

;



# QUERY\_HISTORY

create or replace TRANSIENT TABLE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY (

QUERY\_ID VARCHAR(16777216),

QUERY\_TEXT VARCHAR(16777216),

DATABASE\_NAME VARCHAR(16777216),

SCHEMA\_NAME VARCHAR(16777216),

QUERY\_TYPE VARCHAR(16777216),

SESSION\_ID NUMBER(38,0),

USER\_NAME VARCHAR(16777216),

ROLE\_NAME VARCHAR(16777216),

WAREHOUSE\_NAME VARCHAR(16777216),

WAREHOUSE\_SIZE VARCHAR(16777216),

WAREHOUSE\_TYPE VARCHAR(16777216),

CLUSTER\_NUMBER NUMBER(38,0),

QUERY\_TAG VARCHAR(16777216),

EXECUTION\_STATUS VARCHAR(16777216),

ERROR\_CODE VARCHAR(16777216),

ERROR\_MESSAGE VARCHAR(16777216),

START\_TIME TIMESTAMP\_LTZ(6),

END\_TIME TIMESTAMP\_LTZ(6),

TOTAL\_ELAPSED\_TIME NUMBER(38,0),

BYTES\_SCANNED NUMBER(38,0),

PERCENTAGE\_SCANNED\_FROM\_CACHE FLOAT,

BYTES\_WRITTEN NUMBER(38,0),

BYTES\_WRITTEN\_TO\_RESULT NUMBER(38,0),

BYTES\_READ\_FROM\_RESULT NUMBER(38,0),

ROWS\_PRODUCED NUMBER(38,0),

ROWS\_INSERTED NUMBER(38,0),

ROWS\_UPDATED NUMBER(38,0),

ROWS\_DELETED NUMBER(38,0),

ROWS\_UNLOADED NUMBER(38,0),

BYTES\_DELETED NUMBER(38,0),

PARTITIONS\_SCANNED NUMBER(38,0),

PARTITIONS\_TOTAL NUMBER(38,0), BYTES\_SPILLED\_TO\_LOCAL\_STORAGE NUMBER(38,0),

BYTES\_SPILLED\_TO\_REMOTE\_STORAGE NUMBER(38,0),

BYTES\_SENT\_OVER\_THE\_NETWORK NUMBER(38,0),

COMPILATION\_TIME NUMBER(38,0),

EXECUTION\_TIME NUMBER(38,0),

QUEUED\_PROVISIONING\_TIME NUMBER(38,0),

QUEUED\_REPAIR\_TIME NUMBER(38,0),

QUEUED\_OVERLOAD\_TIME NUMBER(38,0),

TRANSACTION\_BLOCKED\_TIME NUMBER(38,0),

LIST\_EXTERNAL\_FILES\_TIME NUMBER(38,0),

CREDITS\_USED\_CLOUD\_SERVICES FLOAT,

QUERY\_LOAD\_PERCENT NUMBER(38,0), QUERY\_ACCELERATION\_BYTES\_SCANNED NUMBER(38,0), QUERY\_ACCELERATION\_PARTITIONS\_SCANNED NUMBER(38,0), QUERY\_ACCELERATION\_UPPER\_LIMIT\_SCALE\_FACTOR NUMBER(38,0), TRANSACTION\_ID NUMBER(38,0),

CHILD\_QUERIES\_WAIT\_TIME NUMBER(38,0)

)

;



# QUERY\_HISTORY\_SP()

CREATE OR REPLACE PROCEDURE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY\_SP()

RETURNS VARCHAR(16777216)

LANGUAGE JAVASCRIPT

COMMENT='Updates Query History Table'

EXECUTE AS OWNER

AS

$$

var sql\_command = `insert into MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY (

QUERY\_ID

, QUERY\_TEXT

, DATABASE\_NAME

, SCHEMA\_NAME

, QUERY\_TYPE

, SESSION\_ID

, USER\_NAME

, ROLE\_NAME

, WAREHOUSE\_NAME

, WAREHOUSE\_SIZE

, WAREHOUSE\_TYPE

, CLUSTER\_NUMBER

, QUERY\_TAG

, EXECUTION\_STATUS

, ERROR\_CODE

, ERROR\_MESSAGE

, START\_TIME

, END\_TIME

, TOTAL\_ELAPSED\_TIME

, BYTES\_SCANNED

, PERCENTAGE\_SCANNED\_FROM\_CACHE

, BYTES\_WRITTEN

, BYTES\_WRITTEN\_TO\_RESULT

, BYTES\_READ\_FROM\_RESULT

, ROWS\_PRODUCED

, ROWS\_INSERTED

, ROWS\_UPDATED

, ROWS\_DELETED

, ROWS\_UNLOADED

, BYTES\_DELETED

, PARTITIONS\_SCANNED

, PARTITIONS\_TOTAL

, BYTES\_SPILLED\_TO\_LOCAL\_STORAGE

, BYTES\_SPILLED\_TO\_REMOTE\_STORAGE

, BYTES\_SENT\_OVER\_THE\_NETWORK

, COMPILATION\_TIME

, EXECUTION\_TIME

, QUEUED\_PROVISIONING\_TIME

, QUEUED\_REPAIR\_TIME

, QUEUED\_OVERLOAD\_TIME

, TRANSACTION\_BLOCKED\_TIME

, LIST\_EXTERNAL\_FILES\_TIME

, CREDITS\_USED\_CLOUD\_SERVICES

, QUERY\_LOAD\_PERCENT

, QUERY\_ACCELERATION\_BYTES\_SCANNED

, QUERY\_ACCELERATION\_PARTITIONS\_SCANNED

, QUERY\_ACCELERATION\_UPPER\_LIMIT\_SCALE\_FACTOR

, TRANSACTION\_ID,CHILD\_QUERIES\_WAIT\_TIME

)

select QUERY\_ID

, QUERY\_TEXT

, DATABASE\_NAME

, SCHEMA\_NAME

, QUERY\_TYPE

, SESSION\_ID

, USER\_NAME

, ROLE\_NAME

, WAREHOUSE\_NAME

, WAREHOUSE\_SIZE

, WAREHOUSE\_TYPE

, CLUSTER\_NUMBER

, QUERY\_TAG

, EXECUTION\_STATUS

, ERROR\_CODE

, ERROR\_MESSAGE

, START\_TIME

, END\_TIME

, TOTAL\_ELAPSED\_TIME

, BYTES\_SCANNED

, PERCENTAGE\_SCANNED\_FROM\_CACHE

, BYTES\_WRITTEN

, BYTES\_WRITTEN\_TO\_RESULT

, BYTES\_READ\_FROM\_RESULT

, ROWS\_PRODUCED

, ROWS\_INSERTED

, ROWS\_UPDATED

, ROWS\_DELETED

, ROWS\_UNLOADED

, BYTES\_DELETED

, PARTITIONS\_SCANNED

, PARTITIONS\_TOTAL

, BYTES\_SPILLED\_TO\_LOCAL\_STORAGE

, BYTES\_SPILLED\_TO\_REMOTE\_STORAGE

, BYTES\_SENT\_OVER\_THE\_NETWORK

, COMPILATION\_TIME

, EXECUTION\_TIME

, QUEUED\_PROVISIONING\_TIME

, QUEUED\_REPAIR\_TIME

, QUEUED\_OVERLOAD\_TIME

, TRANSACTION\_BLOCKED\_TIME

, LIST\_EXTERNAL\_FILES\_TIME

, CREDITS\_USED\_CLOUD\_SERVICES

, QUERY\_LOAD\_PERCENT

, QUERY\_ACCELERATION\_BYTES\_SCANNED

, QUERY\_ACCELERATION\_PARTITIONS\_SCANNED

, QUERY\_ACCELERATION\_UPPER\_LIMIT\_SCALE\_FACTOR

, TRANSACTION\_ID,CHILD\_QUERIES\_WAIT\_TIME

from snowflake.account\_usage.query\_history

WHERE START\_TIME > (Select max(START\_TIME) from MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY)

`;

var del\_command =`delete from MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY

where start\_time <= current\_date() - 730`;

try {

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

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

return "Success";

}

catch (err) {

return "Failed" + err;

}

$$

;

# 

# QUERY\_HISTORY\_TASK

create or replace task MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY\_TASK

warehouse=MONITOR\_IRB\_WH

schedule='USING CRON 10 3 \* \* \* UTC'

as

call MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY\_SP();



# QUERY\_PERFORMANCE\_KPI\_ALL

create or replace TABLE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_ALL (

QUERY\_ID VARCHAR(16777216),

PARAMETERS VARCHAR(16777216),

VALUE NUMBER(38,0),

RECOMMENDATION VARCHAR(16777216),

INSERT\_TIME TIMESTAMP\_NTZ(9)

)

;



# QUERY\_PERFORMANCE\_KPI\_UNIQUE

create or replace TABLE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_UNIQUE (

QUERY\_ID VARCHAR(16777216),

PARAMETERS VARCHAR(16777216),

VALUE NUMBER(38,0),

RECOMMENDATION VARCHAR(16777216),

INSERT\_TIME TIMESTAMP\_NTZ(9)

)

;

# QUERY\_ANALYZER SP

CREATE OR REPLACE PROCEDURE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER("QID" VARCHAR(16777216))

RETURNS TABLE ( "QUERY\_ID" VARCHAR(784584)

, "PARAMETERS" VARCHAR(784584)

, "VALUE" NUMBER(30,2)

, "RECOMMENDATION" VARCHAR(784584))

LANGUAGE SQL

COMMENT='Function to analyse query\_id on 14 metrics '

EXECUTE AS OWNER

AS

$$

DECLARE

result RESULTSET;

test varchar;

s varchar;

n number;

t timestamp;

comment varchar;

qid2 varchar;

query1 varchar;

query2 varchar;

similarity number;

c number;

TOTAL\_ELAPSED\_TIME NUMBER(38,0);

EXECUTION\_TIME NUMBER(38,0);

QUERY\_LOAD\_PERCENT NUMBER(38,0);

PARTITIONS\_SCANNED NUMBER(38,0);

PARTITIONS\_TOTAL NUMBER(38,0);

ERROR\_CODE NUMBER(38,0);

ERROR\_MESSAGE VARCHAR(16777216);

QUEUED\_OVERLOAD\_TIME NUMBER(38,0);

TRANSACTION\_BLOCKED\_TIME NUMBER(38,0);

BYTES\_SCANNED NUMBER(38,0);

ROLE\_NAME VARCHAR(16777216);

USER\_NAME VARCHAR(16777216);

WAREHOUSE\_NAME VARCHAR(16777216);

WAREHOUSE\_SIZE VARCHAR(16777216);

BYTES\_SPILLED\_TO\_REMOTE\_STORAGE NUMBER(38,0);

START\_TIME TIMESTAMP\_LTZ(6);

END\_TIME TIMESTAMP\_LTZ(6);

MOST\_EXPENSIVE\_NODES VARCHAR(16777216);

BEGIN

SELECT coalesce(total\_elapsed\_time,0) as TOTAL\_ELAPSED\_TIME

, coalesce(execution\_time,0) as EXECUTION\_TIME

, coalesce(query\_load\_percent,0) as query\_load\_percent

, coalesce(partitions\_scanned,0) as partitions\_scanned

, coalesce(partitions\_total,0) as partitions\_total

, coalesce(ERROR\_CODE,0) as ERROR\_CODE

, coalesce(ERROR\_MESSAGE,'') as ERROR\_MESSAGE

, coalesce(QUEUED\_OVERLOAD\_TIME,0) as QUEUED\_OVERLOAD\_TIME

, coalesce(TRANSACTION\_BLOCKED\_TIME,0) as TRANSACTION\_BLOCKED\_TIME

, coalesce(BYTES\_SCANNED,0) as BYTES\_SCANNED

, coalesce(ROLE\_NAME,'') as ROLE\_NAME

, coalesce(USER\_NAME,'') as USER\_NAME

, COALESCE(WAREHOUSE\_NAME,'') AS WAREHOUSE\_NAME

, coalesce(WAREHOUSE\_SIZE,'') as WAREHOUSE\_SIZE

, coalesce(BYTES\_SPILLED\_TO\_REMOTE\_STORAGE,0) as BYTES\_SPILLED\_TO\_REMOTE\_STORAGE

, START\_TIME

, END\_TIME

INTO :TOTAL\_ELAPSED\_TIME

, :EXECUTION\_TIME

, :QUERY\_LOAD\_PERCENT

, :PARTITIONS\_SCANNED

, :PARTITIONS\_TOTAL

, :ERROR\_CODE

, :ERROR\_MESSAGE

, :QUEUED\_OVERLOAD\_TIME

, :TRANSACTION\_BLOCKED\_TIME

, :BYTES\_SCANNED

, :ROLE\_NAME

, :USER\_NAME

, :WAREHOUSE\_NAME

, :WAREHOUSE\_SIZE

, :BYTES\_SPILLED\_TO\_REMOTE\_STORAGE

, :START\_TIME

, :END\_TIME

FROM MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY

WHERE query\_id = :qid ;

SELECT LISTAGG(effective\_info,', ')

INTO :MOST\_EXPENSIVE\_NODES

FROM (SELECT operator\_type||' ['||step\_id||'.'||eff\_operator\_id||'] '||round(pct,1)||'%' as effective\_info

FROM (SELECT operator\_type, step\_id, (ROW\_NUMBER() OVER (ORDER BY step\_id asc, operator\_id asc) - 1) as eff\_operator\_id

, round(execution\_time\_breakdown:overall\_percentage\*100,2) as pct

FROM table(get\_query\_operator\_stats(:qid))

ORDER BY 3 DESC )

ORDER BY pct desc

LIMIT 3 ) ;

CREATE OR REPLACE TEMPORARY TABLE res\_table(

QUERY\_ID varchar(784584)

, PARAMETERS varchar(784584)

, VALUE number(30,2)

, RECOMMENDATION varchar(784584)

);

CREATE OR REPLACE TEMPORARY TABLE sim\_table(

QUERY\_ID1 VARCHAR(784584)

, QUERY1 VARCHAR(784584)

, QUERY\_ID2 VARCHAR(784584)

, QUERY2 VARCHAR(784584)

, SIMILARITY\_MEASURE NUMBER(30,2)

);

let query\_cursor CURSOR FOR (

WITH role\_info(query\_id,role\_name) AS (

SELECT query\_id

,recommendation

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS='ROLE\_NAME'

and recommendation = ?

), user\_info(query\_id,user\_name) AS (

SELECT query\_id

, recommendation

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS ='USER\_NAME'

and recommendation = ?

), warehouse\_info(query\_id,warehouse\_name) AS (

SELECT query\_id

, recommendation

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS='WAREHOUSE\_NAME'

and recommendation = ?

)

select r.query\_id as query\_id

from role\_info r

inner join user\_info u on r.query\_id = u.query\_id

inner join warehouse\_info w on u.query\_id = w.query\_id

);

open query\_cursor using (:ROLE\_NAME, :USER\_NAME, :WAREHOUSE\_NAME);

fetch query\_cursor INTO qid2;

FOR row\_var IN query\_cursor DO

qid2 := row\_var.query\_id;

query1 := (select query\_text

from MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY

WHERE QUERY\_ID = :qid);

query2 := (select query\_text

from MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY

WHERE QUERY\_ID = :qid2);

similarity := JAROWINKLER\_SIMILARITY(:query1,:query2);

if(similarity >= 95) then

insert into sim\_table(QUERY\_ID1, QUERY1, QUERY\_ID2, QUERY2, SIMILARITY\_MEASURE)

values (:qid, :query1, :qid2, :query2, :similarity);

END IF;

END FOR;

delete from monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where query\_id IN ( SELECT QUERY\_ID2 FROM sim\_table where query\_id2 <> :qid);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'TOP 3 MOST EXPENSIVE NODES', -1, :MOST\_EXPENSIVE\_NODES);

IF (EXECUTION\_TIME / TOTAL\_ELAPSED\_TIME <= 0.4) THEN

comment := 'Time spent on overhead activites is too high : '|| ROUND((100 - 100 \* EXECUTION\_TIME / TOTAL\_ELAPSED\_TIME),2)||'%';

ELSEIF (TOTAL\_ELAPSED\_TIME > 10800000) THEN

comment := 'Long Running Query';

ELSE comment := ' '; END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'TOTAL\_ELAPSED\_TIME(ms)', :TOTAL\_ELAPSED\_TIME, :comment);

IF (QUERY\_LOAD\_PERCENT <= 50) THEN

comment := 'If query load percentage is frequently less than 50 percent, adjust warehouse size to a smaller size';

ELSE

comment := 'OK';

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'QUERY\_LOAD\_PERCENT', :QUERY\_LOAD\_PERCENT, :comment);

n := PARTITIONS\_SCANNED / (PARTITIONS\_TOTAL + 1) \* 100.0;

IF (PARTITIONS\_TOTAL > 0 AND PARTITIONS\_SCANNED > (0.80 \* PARTITIONS\_TOTAL)) THEN

comment := 'Optimization is required - Clustering/ Precision in where condition';

ELSE

comment := 'OK';

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'PARTITIONS\_SCANNED / PARTITIONS\_TOTAL', :n, :comment);

IF (BYTES\_SCANNED > 0 AND BYTES\_SPILLED\_TO\_REMOTE\_STORAGE >= 5 \* BYTES\_SCANNED) THEN

comment := 'Excessive remote spillage observed';

ELSE

comment := 'OK';

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'BYTES\_SPILLED\_TO\_REMOTE\_STORAGE', :BYTES\_SPILLED\_TO\_REMOTE\_STORAGE, :comment);

IF (TRANSACTION\_BLOCKED\_TIME > 0) THEN

comment := 'If this repeats for the query frequently,take action accordingly';

ELSE

comment := 'OK';

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'TRANSACTION\_BLOCKED\_TIME', :TRANSACTION\_BLOCKED\_TIME, :comment);

IF (QUEUED\_OVERLOAD\_TIME > 1000) THEN

comment := 'Increase number of clusters on the warehouse to optimize performance(Helps in performance, not cost reduction)';

ELSE

comment := 'OK';

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'QUEUED\_OVERLOAD\_TIME', :QUEUED\_OVERLOAD\_TIME, :comment);

IF (ERROR\_MESSAGE = ' ') THEN

comment := 'OK';

ELSE

comment := ERROR\_MESSAGE;

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'ERROR\_CODE', :ERROR\_CODE, :comment);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'ROLE\_NAME', -1, :ROLE\_NAME);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'USER\_NAME', -1, :USER\_NAME);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'WAREHOUSE\_NAME', -1, :WAREHOUSE\_NAME);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'WAREHOUSE\_SIZE', -1, :WAREHOUSE\_SIZE);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'START\_TIME', -1, to\_varchar(:START\_TIME));

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'END\_TIME', -1, to\_varchar(:END\_TIME));

result := (SELECT QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION FROM res\_table);

RETURN TABLE(result);

END;

$$

;



# QUERY\_ANALYZER\_REPORT()

CREATE OR REPLACE PROCEDURE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_REPORT()

RETURNS NUMBER(38,0)

LANGUAGE SQL

EXECUTE AS OWNER

AS

$$

DECLARE

query\_list RESULTSET DEFAULT (SELECT query\_id, start\_time

FROM MONITOR\_DB.PERFORMANCE\_MONITOR\_SCHEMA.BAD\_QUERY

WHERE performance = 'LONG'

AND START\_TIME > (SELECT VALUE

FROM MONITOR\_DB.COMMON\_TABLE\_SCHEMA.CONTROL\_TABLE

WHERE KEY = 'LAST\_QUERY\_START\_TIME')

AND START\_TIME >= DATEADD(DAY, -14, CURRENT\_TIMESTAMP())

ORDER BY start\_time asc);

query\_cursor CURSOR FOR query\_list;

start\_time timestamp;

query\_res RESULTSET;

query varchar;

rowcount integer default 0;

BEGIN

DELETE FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique kpi

using ( select query\_id, parameters, count(\*), max(insert\_time) insert\_time

from monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

group by query\_id, parameters

having count(\*) > 1 ) as CTE

WHERE CTE.query\_id = kpi.query\_id

and cte.insert\_time <> kpi.insert\_time;

FOR row\_var IN query\_cursor DO

query := 'CALL QUERY\_ANALYZER('''||row\_var.query\_id||''')';

query\_res := (EXECUTE IMMEDIATE :query);

INSERT INTO MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_UNIQUE(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, INSERT\_TIME)

SELECT QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, CURRENT\_TIMESTAMP()

FROM MONITOR\_DB.COMMON\_TABLE\_SCHEMA.RES\_TABLE;

INSERT INTO MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_ALL(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, INSERT\_TIME)

SELECT QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, CURRENT\_TIMESTAMP()

FROM MONITOR\_DB.COMMON\_TABLE\_SCHEMA.RES\_TABLE;

rowcount := rowcount + SQLROWCOUNT;

start\_time := row\_var.start\_time;

END FOR;

Delete from MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_UNIQUE

where insert\_time <= current\_date() - 183;

query := 'UPDATE CONTROL\_TABLE SET VALUE = '''||start\_time||'WHERE KEY = ''LAST\_QUERY\_START\_TIME''';

query\_res := (EXECUTE IMMEDIATE :query);

return rowcount;

END;

$$

;



# QUERY\_ANALYZER\_TASK

create or replace task MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_TASK

warehouse=MONITOR\_IRB\_WH

schedule='USING CRON 30 3 \* \* \* UTC'

USER\_TASK\_TIMEOUT\_MS=18000000

as

call MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_REPORT();



# EMAIL\_ALERT() SP

CREATE OR REPLACE PROCEDURE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.EMAIL\_ALERT()

RETURNS VARCHAR(16777216)

LANGUAGE SQL

COMMENT='Fetches from kpi\_unique table and then sends mail for top 50 queries daily'

EXECUTE AS OWNER

AS

$$

BEGIN

CREATE OR replace temporary table email\_body as

select '<table>'||listagg(header,'\\n\\n')||'</table>' as body

from ( select '<tr><th>QUERY\_ID</th>

<th>TOTAL\_ELAPSED\_TIME</th>

<th>TRANSACTION\_BLOCKED\_TIME</th>

<th>PARTITION\_SCAN\_RATIO</th>

<th>REMOTE\_SPILLAGE</th>

<th>QUERY\_LOAD\_PERCENT</th>

<th>QUEUE\_OVERLOAD\_TIME</th>

<th>USER\_NAME</th>

<th>WAREHOUSE\_NAME</th>

<th>WAREHOUSE\_SIZE</th>

<th>ROLE\_NAME</th>

<th>QUERY\_START\_TIME</th>

<th>QUERY\_END\_TIME</th>

<th>TOP\_3\_MOST\_EXPENSIVE\_NODES</th>

<th>STATUS</th></tr>' as header

union all

select '<tr><td>'||query\_id||'</td><td>'

||SPLIT\_PART(full\_str,'$',19)||'ms.'||SPLIT\_PART(full\_str,'$',20)||'</td><td>'

|| CASE WHEN SPLIT\_PART(full\_str,'$',22) = 'OK' THEN 'OK '

ELSE SPLIT\_PART(full\_str,'$',21)||'ms. '||SPLIT\_PART(full\_str,'$',22)

END||'</td><td>'

|| CASE WHEN SPLIT\_PART(full\_str,'$',8) = 'OK' THEN 'OK'

ELSE 'Partition scan ratio is '||SPLIT\_PART(full\_str,'$',7)||'%. '||SPLIT\_PART(full\_str,'$',8)

END||'</td><td>'

|| CASE WHEN SPLIT\_PART(full\_str,'$',2) = 'OK' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',1)||' bytes'

END||'</td><td>'

|| CASE WHEN SPLIT\_PART(full\_str,'$',10) = 'OK' THEN 'OK'

ELSE 'Load Percent = '||SPLIT\_PART(full\_str,'$',9)||'. '||SPLIT\_PART(full\_str,'$',10)

END||'</td><td>'

|| CASE WHEN SPLIT\_PART(full\_str,'$',12) = 'OK' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',11)||'ms. '||SPLIT\_PART(full\_str,'$',12)

END||'</td><td>'

|| SPLIT\_PART(full\_str,'$',24)||'</td><td>'

|| SPLIT\_PART(full\_str,'$',26)||'</td><td>'

|| SPLIT\_PART(full\_str,'$',28)||'</td><td>'

|| SPLIT\_PART(full\_str,'$',14)||'</td><td>'

|| SPLIT\_PART(full\_str,'$',16)||'</td><td>'

|| SPLIT\_PART(full\_str,'$',4)||'</td><td>'

|| SPLIT\_PART(full\_str,'$',18)||'</td><td>'

|| CASE WHEN SPLIT\_PART(full\_str,'$',5) = 0 THEN 'Success'

ELSE 'Query failed due to error code - '||SPLIT\_PART(full\_str,'$',5)||'('||SPLIT\_PART(full\_str,'$',6)||')'

END||'</td></tr>' as row\_line

from ( select main.query\_id, full\_str

from ( select query\_id

, listagg(coalesce(VALUE,'0')||'$'||recommendation, '$') within group (order by parameters) as full\_str

from monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where insert\_time > dateadd('HOUR',-23,current\_timestamp())

group by QUERY\_ID ) MAIN

inner join

( SELECT KPI.query\_id

, round(value/1000/60,0) as elapsed\_time\_in\_mins

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique KPI

where PARAMETERS = 'TOTAL\_ELAPSED\_TIME(ms)' ) TET

ON TET.query\_id = main.query\_id

inner join

( SELECT KPI.query\_id

, count(\*) recom\_cnt

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique KPI

where PARAMETERS IN('QUERY\_LOAD\_PERCENT', 'QUEUED\_OVERLOAD\_TIME', 'TRANSACTION\_BLOCKED\_TIME', 'TOTAL\_ELAPSED\_TIME(ms)', 'BYTES\_SPILLED\_TO\_REMOTE\_STORAGE', 'PARTITIONS\_SCANNED / PARTITIONS\_TOTAL')

and recommendation not in ('OK', ' ', '')

group by KPI.query\_id ) RECOM\_CNT

on RECOM\_CNT.query\_id = main.query\_id

inner join ( SELECT query\_id,recommendation warehouse\_name

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS='WAREHOUSE\_NAME'

AND RECOMMENDATION LIKE '%PROD%' ) wh

on wh.query\_id = main.query\_id

order by recom\_cnt.recom\_cnt desc, elapsed\_time\_in\_mins desc

Limit 50

) data

);

CALL SYSTEM$SEND\_EMAIL('query\_analysis\_mail'

,'rohit.jagannath@inspirebrands.com,simran.jain@inspirebrands.com,asokhi@inspirebrands.com'

, (SELECT 'Query Analysis Report : '||current\_timestamp())

, (select body from email\_body)

, 'text/html'

);

RETURN 'Success';

END;

$$

;



# QUERY\_ANALYZER\_EMAIL\_ALERT

create or replace task MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_EMAIL\_ALERT

warehouse=MONITOR\_IRB\_WH

after MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_TASK as

call MONITOR\_DB.COMMON\_TABLE\_SCHEMA.EMAIL\_ALERT();

# QUERY\_ANALYZER\_REPORT\_MISSING()

CREATE OR REPLACE PROCEDURE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_REPORT\_MISSING()

RETURNS NUMBER(38,0)

LANGUAGE SQL

COMMENT='Picks Query\_id from missing\_qids table , processes using Query\_analyser() and updates both KPI (all and unique ) tables)'

EXECUTE AS OWNER

AS

$$

DECLARE

query\_list RESULTSET DEFAULT (select query\_id

from MONITOR\_DB.COMMON\_TABLE\_SCHEMA.MISSING\_QIDS);

query\_cursor CURSOR FOR query\_list;

query\_res RESULTSET;

query varchar;

rowcount integer default 0;

BEGIN

FOR row\_var IN query\_cursor DO

query := 'CALL QUERY\_ANALYZER('''||row\_var.query\_id||''')';

query\_res := (EXECUTE IMMEDIATE :query);

INSERT INTO MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_ALL(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, INSERT\_TIME)

SELECT QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, CURRENT\_TIMESTAMP()

FROM MONITOR\_DB.COMMON\_TABLE\_SCHEMA.RES\_TABLE;

INSERT INTO MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_UNIQUE(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, INSERT\_TIME)

SELECT QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, CURRENT\_TIMESTAMP()

FROM MONITOR\_DB.COMMON\_TABLE\_SCHEMA.RES\_TABLE;

rowcount := rowcount + SQLROWCOUNT;

END FOR;

return rowcount;

END;

$$

;



# OPTIMIZATION\_UTILITY.QUERY\_ANALYZER()

CREATE OR REPLACE PROCEDURE MONITOR\_DB.OPTIMIZATION\_UTILITY.QUERY\_ANALYZER("QID" VARCHAR(16777216))

RETURNS TABLE ( "QUERY\_ID" VARCHAR(784584)

, "PARAMETERS" VARCHAR(784584)

, "VALUE" NUMBER(30,2)

, "RECOMMENDATION" VARCHAR(784584))

LANGUAGE SQL

COMMENT='Could be used for single QID analysis.Uses Query history(Account usage) table and does not have Similarity Logic.'

EXECUTE AS OWNER

AS

$$

DECLARE

result RESULTSET;

test varchar;

s varchar;

n number;

t timestamp;

comment varchar;

qid2 varchar;

query1 varchar;

query2 varchar;

similarity number;

c number;

TOTAL\_ELAPSED\_TIME NUMBER(38,0);

EXECUTION\_TIME NUMBER(38,0);

QUERY\_LOAD\_PERCENT NUMBER(38,0);

PARTITIONS\_SCANNED NUMBER(38,0);

PARTITIONS\_TOTAL NUMBER(38,0);

ERROR\_CODE NUMBER(38,0);

ERROR\_MESSAGE VARCHAR(16777216);

QUEUED\_OVERLOAD\_TIME NUMBER(38,0);

TRANSACTION\_BLOCKED\_TIME NUMBER(38,0);

BYTES\_SCANNED NUMBER(38,0);

ROLE\_NAME VARCHAR(16777216);

USER\_NAME VARCHAR(16777216);

WAREHOUSE\_NAME VARCHAR(16777216);

WAREHOUSE\_SIZE VARCHAR(16777216);

BYTES\_SPILLED\_TO\_REMOTE\_STORAGE NUMBER(38,0);

START\_TIME TIMESTAMP\_LTZ(6);

END\_TIME TIMESTAMP\_LTZ(6);

MOST\_EXPENSIVE\_NODES VARCHAR(16777216);

BEGIN

SELECT coalesce(total\_elapsed\_time,0) as TOTAL\_ELAPSED\_TIME

, coalesce(execution\_time,0) as EXECUTION\_TIME

, coalesce(query\_load\_percent,0) as query\_load\_percent

, coalesce(partitions\_scanned,0) as partitions\_scanned

, coalesce(partitions\_total,0) as partitions\_total

, coalesce(ERROR\_CODE,0) as ERROR\_CODE

, coalesce(ERROR\_MESSAGE,'') as ERROR\_MESSAGE

, coalesce(QUEUED\_OVERLOAD\_TIME,0) as QUEUED\_OVERLOAD\_TIME

, coalesce(TRANSACTION\_BLOCKED\_TIME,0) as TRANSACTION\_BLOCKED\_TIME

, coalesce(BYTES\_SCANNED,0) as BYTES\_SCANNED

, coalesce(ROLE\_NAME,'') as ROLE\_NAME

, coalesce(USER\_NAME,'') as USER\_NAME

, COALESCE(WAREHOUSE\_NAME,'') AS WAREHOUSE\_NAME

, coalesce(WAREHOUSE\_SIZE,'') as WAREHOUSE\_SIZE

, coalesce(BYTES\_SPILLED\_TO\_REMOTE\_STORAGE,0) as BYTES\_SPILLED\_TO\_REMOTE\_STORAGE

, START\_TIME

, END\_TIME

INTO :TOTAL\_ELAPSED\_TIME

, :EXECUTION\_TIME

, :QUERY\_LOAD\_PERCENT

, :PARTITIONS\_SCANNED

, :PARTITIONS\_TOTAL

, :ERROR\_CODE

, :ERROR\_MESSAGE

, :QUEUED\_OVERLOAD\_TIME

, :TRANSACTION\_BLOCKED\_TIME

, :BYTES\_SCANNED

, :ROLE\_NAME

, :USER\_NAME

, :WAREHOUSE\_NAME

, :WAREHOUSE\_SIZE

, :BYTES\_SPILLED\_TO\_REMOTE\_STORAGE

, :START\_TIME

, :END\_TIME

FROM SNOWFLAKE.ACCOUNT\_USAGE.QUERY\_HISTORY

WHERE query\_id = :qid ;

SELECT LISTAGG(effective\_info,', ')

INTO :MOST\_EXPENSIVE\_NODES

FROM (SELECT operator\_type||' ['||step\_id||'.'||eff\_operator\_id||'] '||round(pct,1)||'%' as effective\_info

FROM (SELECT operator\_type, step\_id, (ROW\_NUMBER() OVER (ORDER BY step\_id asc, operator\_id asc) - 1) as eff\_operator\_id

, round(execution\_time\_breakdown:overall\_percentage\*100,2) as pct

FROM table(get\_query\_operator\_stats(:qid))

ORDER BY 3 DESC )

ORDER BY pct desc

LIMIT 3 ) ;

CREATE OR REPLACE TEMPORARY TABLE res\_table(

QUERY\_ID varchar(784584)

, PARAMETERS varchar(784584)

, VALUE number(30,2)

, RECOMMENDATION varchar(784584)

);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'TOP 3 MOST EXPENSIVE NODES', -1, :MOST\_EXPENSIVE\_NODES);

IF (EXECUTION\_TIME / TOTAL\_ELAPSED\_TIME <= 0.4) THEN

comment := 'Time spent on overhead activites is too high : '|| ROUND((100 - 100 \* EXECUTION\_TIME / TOTAL\_ELAPSED\_TIME),2)||'%';

ELSEIF (TOTAL\_ELAPSED\_TIME > 10800000) THEN

comment := 'Long Running Query';

ELSE comment := ' '; END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'TOTAL\_ELAPSED\_TIME(ms)', :TOTAL\_ELAPSED\_TIME, :comment);

IF (QUERY\_LOAD\_PERCENT <= 50) THEN

comment := 'If query load percentage is frequently less than 50 percent, adjust warehouse size to a smaller size';

ELSE

comment := 'OK';

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'QUERY\_LOAD\_PERCENT', :QUERY\_LOAD\_PERCENT, :comment);

n := PARTITIONS\_SCANNED / (PARTITIONS\_TOTAL + 1) \* 100.0;

IF (PARTITIONS\_TOTAL > 0 AND PARTITIONS\_SCANNED > (0.80 \* PARTITIONS\_TOTAL)) THEN

comment := 'Optimization is required - Clustering/ Precision in where condition';

ELSE

comment := 'OK';

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'PARTITIONS\_SCANNED / PARTITIONS\_TOTAL', :n, :comment);

IF (BYTES\_SCANNED > 0 AND BYTES\_SPILLED\_TO\_REMOTE\_STORAGE >= 5 \* BYTES\_SCANNED) THEN

comment := 'Excessive remote spillage observed';

ELSE

comment := 'OK';

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'BYTES\_SPILLED\_TO\_REMOTE\_STORAGE', :BYTES\_SPILLED\_TO\_REMOTE\_STORAGE, :comment);

IF (TRANSACTION\_BLOCKED\_TIME > 0) THEN

comment := 'If this repeats for the query frequently,take action accordingly';

ELSE

comment := 'OK';

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'TRANSACTION\_BLOCKED\_TIME', :TRANSACTION\_BLOCKED\_TIME, :comment);

IF (QUEUED\_OVERLOAD\_TIME > 1000) THEN

comment := 'Increase number of clusters on the warehouse to optimize performance(Helps in performance, not cost reduction)';

ELSE

comment := 'OK';

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'QUEUED\_OVERLOAD\_TIME', :QUEUED\_OVERLOAD\_TIME, :comment);

IF (ERROR\_MESSAGE = ' ') THEN

comment := 'OK';

ELSE

comment := ERROR\_MESSAGE;

END IF;

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'ERROR\_CODE', :ERROR\_CODE, :comment);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'ROLE\_NAME', -1, :ROLE\_NAME);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'USER\_NAME', -1, :USER\_NAME);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'WAREHOUSE\_NAME', -1, :WAREHOUSE\_NAME);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'WAREHOUSE\_SIZE', -1, :WAREHOUSE\_SIZE);

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'START\_TIME', -1, to\_varchar(:START\_TIME));

INSERT INTO res\_table(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION) VALUES(:qid, 'END\_TIME', -1, to\_varchar(:END\_TIME));

result := (SELECT QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION FROM res\_table);

RETURN TABLE(result);

END;

$$

;



# DASHBOARD QUERIES

## QUERY\_ID vs METRICS(PROD)

WITH filter AS (

SELECT coalesce(min(duration),0) as min\_tet

from monitor\_db.common\_table\_schema.duration\_filter

where DURATION = :elapsedtime

)

,QIDS (query\_id, start\_time) AS (

SELECT query\_id, recommendation

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS='START\_TIME'

AND RECOMMENDATION = :daterange)

, TET AS (

SELECT KPI.query\_id, round(value/1000/60,0) as elapsed\_time\_in\_mins

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique KPI

inner join QIDS ON QIDS.QUERY\_ID = KPI.QUERY\_ID

where PARAMETERS = 'TOTAL\_ELAPSED\_TIME(ms)')

, RECOM\_CNT AS (

SELECT KPI.query\_id, count(\*) recom\_cnt

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique KPI

inner join QIDS ON QIDS.QUERY\_ID = KPI.QUERY\_ID

where PARAMETERS IN('QUERY\_LOAD\_PERCENT', 'QUEUED\_OVERLOAD\_TIME', 'TRANSACTION\_BLOCKED\_TIME', 'TOTAL\_ELAPSED\_TIME(ms)',

'BYTES\_SPILLED\_TO\_REMOTE\_STORAGE', 'PARTITIONS\_SCANNED / PARTITIONS\_TOTAL')

and recommendation not in ('OK', ' ', '')

group by KPI.query\_id)

, RECOM\_AGG AS (

SELECT kpi.query\_id, LISTAGG(value, ' \n') within group(order by value asc) OBSERVATIONS

FROM (SELECT query\_id, (parameters||' (' ||value||') -> '||recommendation) value

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS IN('QUERY\_LOAD\_PERCENT', 'QUEUED\_OVERLOAD\_TIME', 'TRANSACTION\_BLOCKED\_TIME', 'TOTAL\_ELAPSED\_TIME(ms)',

'BYTES\_SPILLED\_TO\_REMOTE\_STORAGE', 'PARTITIONS\_SCANNED / PARTITIONS\_TOTAL', 'TOP 3 MOST EXPENSIVE NODES',

'WAREHOUSE\_NAME', 'ROLE\_NAME')

and recommendation not in ('OK', ' ', '')

) KPI

GROUP BY kpi.query\_id)

, warehouse\_info(query\_id,warehouse\_name) as(

SELECT query\_id,recommendation

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS='WAREHOUSE\_NAME'

AND RECOMMENDATION LIKE '%PROD%'

)

SELECT RECOM\_CNT.QUERY\_ID, qids.start\_time as query\_start\_time, TET.elapsed\_time\_in\_mins, RECOM\_CNT.RECOM\_CNT, RECOM\_AGG.OBSERVATIONS,warehouse\_info.warehouse\_name

FROM RECOM\_CNT

INNER JOIN RECOM\_AGG ON RECOM\_CNT.QUERY\_ID = RECOM\_AGG.QUERY\_ID

INNER JOIN warehouse\_info ON RECOM\_CNT.QUERY\_ID = warehouse\_info.QUERY\_ID

INNER JOIN TET ON RECOM\_CNT.QUERY\_ID = TET.QUERY\_ID

INNER JOIN QIDS ON RECOM\_CNT.QUERY\_ID = QIDS.QUERY\_ID

where TET.elapsed\_time\_in\_mins >= (select min\_tet from filter)

ORDER BY RECOM\_CNT.RECOM\_CNT DESC, TET.elapsed\_time\_in\_mins DESC

;



## QUEUE OVERLOAD TIME BY WAREHOUSE

with query\_wh as (

select query\_id, recommendation as wh\_name

from monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS = 'WAREHOUSE\_NAME'

and insert\_time = :daterange

),

query\_qlp as (

select query\_id, round(value/1000,0) as overload\_time\_in\_secs

from monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where insert\_time = :daterange

and value > 0 -- to eliminate the non-KPI rows

and recommendation not in ('OK', ' ', '') -- to eliminate rows with no recommendations

and parameters = 'QUEUED\_OVERLOAD\_TIME'

)

select wh.query\_id, wh.wh\_name, qlp.overload\_time\_in\_secs

from query\_qlp qlp

inner join query\_wh wh on qlp.query\_id = wh.query\_id

;



## WAREHOUSE\_WISE - QUERY\_ID METRIC LIST



WITH QIDS (query\_id) AS (

SELECT query\_id

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS='START\_TIME'

AND RECOMMENDATION = :daterange)

, WH AS (

SELECT WH.query\_id, recommendation as warehouse\_name

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique WH

INNER JOIN QIDS ON QIDS.QUERY\_ID = WH.QUERY\_ID

where PARAMETERS='WAREHOUSE\_NAME')

, RECOM\_CNT AS (

SELECT KPI.query\_id, count(\*) recom\_cnt

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique KPI

inner join QIDS ON QIDS.QUERY\_ID = KPI.QUERY\_ID

where PARAMETERS IN('QUERY\_LOAD\_PERCENT', 'QUEUED\_OVERLOAD\_TIME', 'TRANSACTION\_BLOCKED\_TIME',

'BYTES\_SPILLED\_TO\_REMOTE\_STORAGE', 'PARTITIONS\_SCANNED / PARTITIONS\_TOTAL', 'TOP 3 MOST EXPENSIVE NODES')

and recommendation not in ('OK', ' ', '')

group by KPI.query\_id)

, RECOM\_AGG AS (

SELECT kpi.query\_id, LISTAGG(value, ' \n') within group(order by value asc) OBSERVATIONS

FROM (SELECT query\_id, (parameters||' (' ||value||') -> '||recommendation) value

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS IN('QUERY\_LOAD\_PERCENT', 'QUEUED\_OVERLOAD\_TIME', 'TRANSACTION\_BLOCKED\_TIME',

'BYTES\_SPILLED\_TO\_REMOTE\_STORAGE', 'PARTITIONS\_SCANNED / PARTITIONS\_TOTAL', 'TOP 3 MOST EXPENSIVE NODES',

'WAREHOUSE\_NAME', 'ROLE\_NAME')

and recommendation not in ('OK', ' ', '')

) KPI

GROUP BY kpi.query\_id

)

SELECT RECOM\_CNT.QUERY\_ID, RECOM\_AGG.OBSERVATIONS, RECOM\_CNT.RECOM\_CNT, WH.warehouse\_name

FROM RECOM\_CNT

INNER JOIN RECOM\_AGG ON RECOM\_CNT.QUERY\_ID = RECOM\_AGG.QUERY\_ID

INNER JOIN WH ON WH.QUERY\_ID = RECOM\_CNT.QUERY\_ID

;



## RECOMMENDATION COUNT per WAREHOUSE IN PROD



WITH QIDS (query\_id) AS (

SELECT query\_id

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS='START\_TIME'

AND RECOMMENDATION = :daterange)

, RECOM\_CNT AS (

SELECT KPI.query\_id, count(\*) recom\_cnt

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique KPI

inner join QIDS ON QIDS.QUERY\_ID = KPI.QUERY\_ID

where PARAMETERS IN('QUERY\_LOAD\_PERCENT', 'QUEUED\_OVERLOAD\_TIME', 'TRANSACTION\_BLOCKED\_TIME',

'BYTES\_SPILLED\_TO\_REMOTE\_STORAGE', 'PARTITIONS\_SCANNED / PARTITIONS\_TOTAL', 'TOP 3 MOST EXPENSIVE NODES')

and recommendation not in ('OK', ' ', '')

group by KPI.query\_id)

, RECOM\_AGG AS (

SELECT kpi.query\_id, LISTAGG(value, ' \n') within group(order by value asc) OBSERVATIONS

FROM (SELECT query\_id, (parameters||' (' ||value||') -> '||recommendation) value

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS IN('QUERY\_LOAD\_PERCENT', 'QUEUED\_OVERLOAD\_TIME', 'TRANSACTION\_BLOCKED\_TIME',

'BYTES\_SPILLED\_TO\_REMOTE\_STORAGE', 'PARTITIONS\_SCANNED / PARTITIONS\_TOTAL', 'TOP 3 MOST EXPENSIVE NODES',

'WAREHOUSE\_NAME', 'ROLE\_NAME')

and recommendation not in ('OK', ' ', '')

) KPI

GROUP BY kpi.query\_id

)

,warehouse\_info(query\_id,warehouse\_name) as(

SELECT query\_id,recommendation

FROM monitor\_db.common\_table\_schema.query\_performance\_kpi\_unique

where PARAMETERS='WAREHOUSE\_NAME'

AND RECOMMENDATION LIKE '%PROD%'

)

SELECT RECOM\_CNT.QUERY\_ID, RECOM\_CNT.RECOM\_CNT, RECOM\_AGG.OBSERVATIONS,warehouse\_info.warehouse\_name

FROM RECOM\_CNT

INNER JOIN RECOM\_AGG ON RECOM\_CNT.QUERY\_ID = RECOM\_AGG.QUERY\_ID

INNER JOIN warehouse\_info ON RECOM\_CNT.QUERY\_ID = warehouse\_info.QUERY\_ID

ORDER BY RECOM\_CNT.RECOM\_CNT DESC;



# RANDOM QUERIES

## QUERY\_ANALYZER\_REPORT TABLE

create or replace transient table QUERY\_ANALYZER\_REPORT\_ALL AS(

select k.query\_id as query\_id,query\_text,database\_name

, CASE WHEN SPLIT\_PART(full\_str,'$',20) = ' ' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',19)||'ms.'||SPLIT\_PART(full\_str,'$',20)

END as TOTAL\_ELAPSED\_TIME\_IN\_MILLISECONDS,

CASE WHEN SPLIT\_PART(full\_str,'$',8) = 'OK' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',7)||'%. '||SPLIT\_PART(full\_str,'$',8)

END as PARTITION\_SCAN\_RATIO,

CASE WHEN SPLIT\_PART(full\_str,'$',2) = 'OK' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',1)||' bytes'

END as SPILLAGE,

CASE WHEN SPLIT\_PART(full\_str,'$',10) = 'OK' THEN 'OK'

ELSE 'Load Percent = '||SPLIT\_PART(full\_str,'$',9)||'. '||SPLIT\_PART(full\_str,'$',10)

END as QUERY\_LOAD\_PERCENTAGE,

CASE WHEN SPLIT\_PART(full\_str,'$',22) = 'OK' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',21)||'ms. '||SPLIT\_PART(full\_str,'$',22)

END as TRANSACTION\_BLOCKED\_TIME,

(SPLIT\_PART(full\_str,'$',19)/3600000)\*(SPLIT\_PART(full\_str,'$',9)/100)\*

CASE WHEN SPLIT\_PART(full\_str,'$',28) ='X-Small' THEN 1

WHEN SPLIT\_PART(full\_str,'$',28) ='Small' THEN 2

WHEN SPLIT\_PART(full\_str,'$',28) ='Medium' THEN 4

WHEN SPLIT\_PART(full\_str,'$',28) ='Large' THEN 8

WHEN SPLIT\_PART(full\_str,'$',28) ='X-Large' THEN 16

WHEN SPLIT\_PART(full\_str,'$',28) ='2X-Large' THEN 32

WHEN SPLIT\_PART(full\_str,'$',28) ='3X-Large' THEN 64

WHEN SPLIT\_PART(full\_str,'$',28) ='4X-Large' THEN 128

WHEN SPLIT\_PART(full\_str,'$',28) ='5X-Large' THEN 256

WHEN SPLIT\_PART(full\_str,'$',28) ='6X-Large' THEN 512

ELSE 0

END as CREDITS,

SPLIT\_PART(full\_str,'$',24) as USER\_NAME,

SPLIT\_PART(full\_str,'$',26) as WAREHOUSE\_NAME,

SPLIT\_PART(full\_str,'$',28) as WAREHOUSE\_SIZE,

SPLIT\_PART(full\_str,'$',14) as ROLE\_NAME,

SPLIT\_PART(full\_str,'$',16)as START\_TIME,

SPLIT\_PART(full\_str,'$',4) as END\_TIME,

SPLIT\_PART(full\_str,'$',18) as MOST\_EXPENSIVE\_NODES,

CASE WHEN SPLIT\_PART(full\_str,'$',5) = '0' THEN 'Success'

ELSE 'Query failed due to error code - '||SPLIT\_PART(full\_str,'$',5)||'('||SPLIT\_PART(full\_str,'$',6)||')'

END as STATUS

from

(

select query\_id,listagg(coalesce(VALUE,0)||'$'||coalesce(recommendation,''), '$') within group (order by parameters) as full\_str

from monitor\_db.common\_table\_schema.query\_performance\_kpi\_clone

group by QUERY\_ID

) k

left join monitor\_db.common\_table\_schema.query\_history q

on k.query\_id = q.query\_id

);

## ALTER TASK

ALTER TASK MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY\_TASK SUSPEND;

alter task MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_TASK SET

SCHEDULE = 'USING CRON 30 3 \* \* \* UTC'

;

ALTER TASK MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_TASK RESUME;

ALTER TASK MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_TASK SUSPEND;

//We have data starting from 2023-07-06 04:03:16.537

select \* from snowflake.account\_usage.query\_history where QUERY\_ID ='01adcee5-0b04-c8d2-0000-5f214cb5a96e' ;

select count(distinct query\_id) from monitor\_db.common\_table\_schema.query\_performance\_kpi\_clone;

select query\_id from monitor\_db.common\_table\_schema.query\_performance\_kpi\_clone where parameters ='WAREHOUSE\_NAME'and recommendation='';

select \* from QUERY\_ANALYZER\_REPORT\_WITH\_DUPLICATES WHERE WAREHOUSE\_NAME =' ';

select \* from monitor\_db.common\_table\_schema.query\_performance\_kpi where parameters LIKE 'TOTAL\_ELAPSED\_TIME%' and value IS NULL;

//We have data starting from 2023-07-06 04:03:16.537

select \* from snowflake.account\_usage.query\_history where QUERY\_ID ='01ad8d40-0c04-bb15-0000-5f214b07f9ba' ;

select count(distinct query\_id) from monitor\_db.common\_table\_schema.query\_analyzer\_report;

select query\_id from monitor\_db.common\_table\_schema.query\_performance\_kpi\_clone where parameters ='WAREHOUSE\_NAME'and recommendation='';

select \* from QUERY\_ANALYZER\_REPORT\_WITH\_DUPLICATES WHERE WAREHOUSE\_NAME =' ';

## QUERY\_EMAIL\_ALERT

select '<table>'||listagg(header,'\n\n')||'</table>' as body

from

(

select '<tr><th>QUERY\_ID</th> <th>TOTAL\_ELAPSED\_TIME</th> <th>TRANSACTION\_BLOCKED\_TIME</th> <th>PARTITION\_SCAN\_RATIO</th> <th>REMOTE\_SPILLAGE</th> <th>QUERY\_LOAD\_PERCENT</th> <th>QUEUE\_OVERLOAD\_TIME</th> <th>USER\_NAME</th> <th>WAREHOUSE\_NAME</th> <th>WAREHOUSE\_SIZE</th> <th>ROLE\_NAME</th> <th>QUERY\_START\_TIME</th> <th>QUERY\_END\_TIME</th> <th>TOP\_3\_MOST\_EXPENSIVE\_NODES</th> <th>STATUS</th></tr>' as header

union all

select '<tr><td>'||query\_id||'</td><td>'

||SPLIT\_PART(full\_str,'$',19)||'ms.'||SPLIT\_PART(full\_str,'$',20)||'</td><td>'||

CASE WHEN SPLIT\_PART(full\_str,'$',22) = 'OK' THEN 'OK '

ELSE SPLIT\_PART(full\_str,'$',21)||'ms. '||SPLIT\_PART(full\_str,'$',22)

END||'</td><td>'||

CASE WHEN SPLIT\_PART(full\_str,'$',8) = 'OK' THEN 'OK'

ELSE 'Partition scan ratio is '||SPLIT\_PART(full\_str,'$',7)||'%. '||SPLIT\_PART(full\_str,'$',8)

END||'</td><td>'||

CASE WHEN SPLIT\_PART(full\_str,'$',2) = 'OK' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',1)||' bytes'

END||'</td><td>'||

CASE WHEN SPLIT\_PART(full\_str,'$',10) = 'OK' THEN 'OK'

ELSE 'Load Percent = '||SPLIT\_PART(full\_str,'$',9)||'. '||SPLIT\_PART(full\_str,'$',10)

END||'</td><td>'||

CASE WHEN SPLIT\_PART(full\_str,'$',12) = 'OK' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',11)||'ms. '||SPLIT\_PART(full\_str,'$',12)

END||'</td><td>'||

SPLIT\_PART(full\_str,'$',24)||'</td><td>'

||SPLIT\_PART(full\_str,'$',26)||'</td><td>'

||SPLIT\_PART(full\_str,'$',28)||'</td><td>'

||SPLIT\_PART(full\_str,'$',14)||'</td><td>'||

SPLIT\_PART(full\_str,'$',16)||'</td><td>'

||SPLIT\_PART(full\_str,'$',4)||'</td><td>'

||SPLIT\_PART(full\_str,'$',18)||'</td><td>'||

CASE WHEN SPLIT\_PART(full\_str,'$',5) = 0 THEN 'Success'

ELSE 'Query failed due to error code - '||SPLIT\_PART(full\_str,'$',5)||'('||SPLIT\_PART(full\_str,'$',6)||')'

END||'</td></tr>'

as row\_line

/\* ,SPLIT\_PART(full\_str,'$',19)||'ms. '||SPLIT\_PART(full\_str,'$',20) as TOTAL\_ELAPSED\_TIME

,CASE WHEN SPLIT\_PART(full\_str,'$',22) = 'OK' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',21)||'ms. '||SPLIT\_PART(full\_str,'$',22)

END as TRANSACTION\_BLOCKED\_TIME

,CASE WHEN SPLIT\_PART(full\_str,'$',8) = 'OK' THEN 'OK'

ELSE 'Partition scan ratio is '||SPLIT\_PART(full\_str,'$',7)||'%. '||SPLIT\_PART(full\_str,'$',8)

END as PARTITION\_SCAN\_RATIO

,CASE WHEN SPLIT\_PART(full\_str,'$',2) = 'OK' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',1)||' bytes'

END as REMOTE\_SPILLAGE

,CASE WHEN SPLIT\_PART(full\_str,'$',10) = 'OK' THEN 'OK'

ELSE 'Load Percent = '||SPLIT\_PART(full\_str,'$',9)||'. '||SPLIT\_PART(full\_str,'$',10)

END as QUERY\_LOAD\_PERCENT

,CASE WHEN SPLIT\_PART(full\_str,'$',12) = 'OK' THEN 'OK'

ELSE SPLIT\_PART(full\_str,'$',11)||'ms. '||SPLIT\_PART(full\_str,'$',12)

END as QUEUE\_OVERLOAD\_TIME

,SPLIT\_PART(full\_str,'$',24) as USER\_NAME

,SPLIT\_PART(full\_str,'$',26) as WAREHOUSE\_NAME

,SPLIT\_PART(full\_str,'$',28) as WAREHOUSE\_SIZE

,SPLIT\_PART(full\_str,'$',14) as ROLE\_NAME

,SPLIT\_PART(full\_str,'$',16) as QUERY\_START\_TIME

,SPLIT\_PART(full\_str,'$',4) as QUERY\_END\_TIME

,SPLIT\_PART(full\_str,'$',18) as TOP\_3\_MOST\_EXPENSIVE\_NODES

,CASE WHEN SPLIT\_PART(full\_str,'$',5) = 0 THEN 'Success'

ELSE 'Query failed due to error code - '||SPLIT\_PART(full\_str,'$',5)||'('||SPLIT\_PART(full\_str,'$',6)||')'

END as STATUS \*/

from

(

select query\_id, listagg(coalesce(VALUE,'0')||'$'||recommendation, '$') within group (order by parameters) as full\_str

from monitor\_db.common\_table\_schema.query\_performance\_kpi

where insert\_time > to\_timestamp('2023-07-17 20:15:09', 'YYYY-MM-DD HH24:MI:SS')

group by QUERY\_ID

Limit 25

)

)

;

CALL SYSTEM$SEND\_EMAIL(

'query\_analysis\_mail',

'rohit.jagannath@inspirebrands.com,simran.jain@inspirebrands.com',

(SELECT 'Query Analysis Report : '||current\_timestamp()),

(SELECT BODY FROM TABLE(RESULT\_SCAN('01adc111-0b04-bf44-0000-5f214c49417a'))),

'text/html'

);

## CLONE TABLE

CREATE OR REPLACE PROCEDURE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_REPORT2()

RETURNS NUMBER(38,0)

LANGUAGE SQL

EXECUTE AS OWNER

AS '

DECLARE

query\_list RESULTSET DEFAULT (SELECT query\_id, start\_time FROM MONITOR\_DB.PERFORMANCE\_MONITOR\_SCHEMA.BAD\_QUERY WHERE performance = ''LONG'' AND START\_TIME > (SELECT MAX(RECOMMENDATION) FROM MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_CLONE WHERE PARAMETERS =''START\_TIME''

) ORDER BY start\_time asc);

query\_cursor CURSOR FOR query\_list;

start\_time timestamp;

query\_res RESULTSET;

query varchar;

rowcount integer default 0;

BEGIN

FOR row\_var IN query\_cursor DO

query := ''CALL QUERY\_ANALYZER2(''''''||row\_var.query\_id||'''''')'';

query\_res := (EXECUTE IMMEDIATE :query);

INSERT INTO MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_CLONE(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, INSERT\_TIME)

SELECT QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, CURRENT\_TIMESTAMP() FROM MONITOR\_DB.COMMON\_TABLE\_SCHEMA.RES\_TABLE;

rowcount := rowcount + SQLROWCOUNT;

start\_time := row\_var.start\_time;

END FOR;

return rowcount;

END;

';

select \* from COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_CLONE ;

//60382- before + 12,796

call MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_REPORT2();

select \* from QUERY\_ANALYZER\_REPORT;

select \* from query\_performance\_kpi where query\_id = '01adbe05-0b04-c793-0000-5f214c33a9da';

select \* from query\_history where query\_id ='01adbe05-0b04-c793-0000-5f214c33a9da';

## MISSING RECORDS

insert into monitor\_db.common\_table\_schema.yappa(

select query\_id from monitor\_db.common\_table\_schema.query\_performance\_kpi\_clone where parameters LIKE 'TOTAL\_ELAPSED\_TIME%' and value IS NULL

);

truncate table yappa;

delete from monitor\_db.common\_table\_schema.query\_performance\_kpi\_clone

where query\_id in (

select query\_id

from monitor\_db.common\_table\_schema.query\_performance\_kpi\_clone

where parameters like 'TOTAL\_ELAPSED\_TIME%'

and recommendation is null

)

;

CREATE OR REPLACE PROCEDURE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_REPORT\_MISSING()

RETURNS NUMBER(38,0)

LANGUAGE SQL

EXECUTE AS OWNER

AS '

DECLARE

query\_list RESULTSET DEFAULT (select query\_id from MONITOR\_DB.COMMON\_TABLE\_SCHEMA.YAPPA);

query\_cursor CURSOR FOR query\_list;

query\_res RESULTSET;

query varchar;

rowcount integer default 0;

BEGIN

FOR row\_var IN query\_cursor DO

query := ''CALL QUERY\_ANALYZER2(''''''||row\_var.query\_id||'''''')'';

query\_res := (EXECUTE IMMEDIATE :query);

INSERT INTO MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_CLONE(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, INSERT\_TIME)

SELECT QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, CURRENT\_TIMESTAMP() FROM MONITOR\_DB.COMMON\_TABLE\_SCHEMA.RES\_TABLE;

rowcount := rowcount + SQLROWCOUNT;

END FOR;

return rowcount;

END;

';

call MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_REPORT\_MISSING();

SELECT \* FROM QUERY\_HISTORY WHERE QUERY\_ID IN (SELECT QUERY\_ID FROM YAPPA);

## QUERY\_ANALYZER\_REPORT\_MISSING()

CREATE OR REPLACE PROCEDURE MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_REPORT\_MISSING()

RETURNS NUMBER(38,0)

LANGUAGE SQL

EXECUTE AS OWNER

AS '

DECLARE

query\_list RESULTSET DEFAULT (select query\_id from MONITOR\_DB.COMMON\_TABLE\_SCHEMA.YAPPA);

query\_cursor CURSOR FOR query\_list;

query\_res RESULTSET;

query varchar;

rowcount integer default 0;

BEGIN

FOR row\_var IN query\_cursor DO

query := ''CALL QUERY\_ANALYZER2(''''''||row\_var.query\_id||'''''')'';

query\_res := (EXECUTE IMMEDIATE :query);

INSERT INTO MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_CLONE(QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, INSERT\_TIME)

SELECT QUERY\_ID, PARAMETERS, VALUE, RECOMMENDATION, CURRENT\_TIMESTAMP() FROM MONITOR\_DB.COMMON\_TABLE\_SCHEMA.RES\_TABLE;

rowcount := rowcount + SQLROWCOUNT;

END FOR;

return rowcount;

END;

';

call MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_REPORT\_MISSING();

## List of Artifacts:

1. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.CONTROL\_TABLE
2. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.DURATION\_FILTER(created for dashboard)
3. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY
4. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI
5. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_SIMILARITY\_COUNT
6. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_TASK
7. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY\_TASK
8. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER("QID" VARCHAR(16777216))
9. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_ANALYZER\_REPORT()
10. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_HISTORY\_SP()
11. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.QUERY\_PERFORMANCE\_KPI\_CLONE
12. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.SIMILARITY\_CHECK("QID" VARCHAR(16777216))
13. MONITOR\_DB.COMMON\_TABLE\_SCHEMA.SIMILARITY\_CHECK2("QID" VARCHAR(16777216))

## KPI

--Inactive Users

select

count(NAME) as Inactive\_users

from

"SNOWFLAKE"."ACCOUNT\_USAGE"."USERS"

where

last\_success\_login < current\_date() -30

or last\_success\_login is null

and name <> 'SNOWFLAKE'

and deleted\_on is null;

-- Total Queries run by database

select

database\_name,

count(distinct query\_id) as total\_queries

from

snowflake.account\_usage.query\_history

where

start\_time = :daterange

group by

1

order by

2 desc;

--

select role\_name,

count(distinct query\_id), max(END\_TIME) as total\_queries

from

snowflake.account\_usage.query\_history

group by

1

order by

3 asc;