Badge 5 Data Engineering

Monday, December 19, 2022 10:48 PM

```
select current_user();

alter user FORRESTLI2011 set default_role = 'SYSADMIN';
alter user FORRESTLI2011 set default_warehouse = 'COMPUTE_WH';
alter user FORRESTLI2011 set default_namespace = 'DEMO_DB.PUBLIC';

use role accountadmin;

select demo_db.public.grader(step, (actual = expected), actual, expected, description) as graded_results from
(SELECT
 'DORA_IS_WORKING' as step
 ,(select 123 ) as actual
 ,123 as expected
 ,'Dora is working!' as description
);

select current_account();

create database AGS_GAME_AUDIENCE;
drop schema AGS_GAME_AUDIENCE.public;
create schema AGS_GAME_AUDIENCE.raw;

create or replace TABLE AGS_GAME_AUDIENCE.RAW.GAME_LOGS (
        RAW_LOG VARIANT
);

create or replace stage uni_kishore
  url='s3://uni-kishore';

list@uni_kishore/kickoff;

create or replace file format FF_JSON_LOGS
  type='JSON'
  strip_outer_array=true;

select $1 from @uni_kishore/kickoff (file_format=>ff_json_logs);

copy into GAME_LOGS
  from @uni_kishore/kickoff
  file_format=(format_name=ff_json_logs);
select * from game_logs;

create or replace view LOGS as
select raw_log:agent::text as agent,
      raw_log:user_login::text as user_login,
      raw_log:user_event::text as user_event,
      raw_log:datetime_iso8601::timestamp_ntz as datetime_iso8601,
      raw_log
   from game_logs;

select * from logs;

--
-- DO NOT EDIT THIS CODE
select GRADER(step, (actual = expected), actual, expected, description) as graded_results from
(
 SELECT
 'DNGW01' as step
  ,(
     select count(*)
     from ags_game_audience.raw.logs
     where is_timestamp_ntz(to_variant(datetime_iso8601))= TRUE
   ) as actual
```

```
, 250 as expected
, 'Project DB and Log File Set Up Correctly' as description
);

select current_timestamp();


--what time zone is your account(and/or session) currently set to? Is it -0700?
select current_timestamp();

--worksheets are sometimes called sessions -- we'll be changing the worksheet time zone
alter session set timezone = 'UTC';
select current_timestamp();

--how did the time differ after changing the time zone for the worksheet?
alter session set timezone = 'Africa/Nairobi';
select current_timestamp();

alter session set timezone = 'Pacific/Funafuti';
select current_timestamp();

alter session set timezone = 'Asia/Shanghai';
select current_timestamp();

--show the account parameter called timezone
show parameters like 'timezone';


--
list @uni_kishore/updated_feed;
select $1 from @uni_kishore/updated_feed (file_format=>ff_json_logs);

copy into GAME_LOGS
  from @uni_kishore/updated_feed
  file_format=(format_name=ff_json_logs);
select * from game_logs;

select raw_log:agent:: text, $1:ip_address::text from game_logs;

create or replace view LOGS as
select raw_log:ip_address::text as ip_address,
      raw_log:agent::text as agent,
      raw_log:user_login::text as user_login,
      raw_log:user_event::text as user_event,
      raw_log:datetime_iso8601::timestamp_ntz as datetime_iso8601,
      raw_log
   from game_logs;

select * from logs;

select * from logs where ip_address <> '';
select * from logs where ip_address is not null;


create or replace view LOGS as
  select raw_log:ip_address::text as ip_address,
      raw_log:user_login::text as user_login,
      raw_log:user_event::text as user_event,
      raw_log:datetime_iso8601::timestamp_ntz as datetime_iso8601,
      raw_log
   from game_logs
   where ip_address is not null;
select * from logs;

select * from logs where user_login like '%Kish%';

--
select GRADER(step, (actual = expected), actual, expected, description) as graded_results from
(
```

```
SELECT
  'DNGW02' as step
  ,( select sum(tally) from(
      select (count(*) * -1) as tally
      from ags_game_audience.raw.logs
      union all
      select count(*) as tally
      from ags_game_audience.raw.game_logs)
    ) as actual
  ,250 as expected
  ,'View is filtered' as description
);

--

select * from logs;

select parse_ip('107.217.231.17','inet');
select parse_ip('107.217.231.17','inet'):host;
select parse_ip('107.217.231.17','inet'):family;

create schema enhanced;

desc view IPINFO_GEOLOC.demo.location;
--Look up Kishore's Time Zone in the IPInfo share using his IP Address with the PARSE_IP function.
select start_ip, end_ip, start_ip_int, end_ip_int, city, region, country, timezone
from IPINFO_GEOLOC.demo.location
where parse_ip('63.235.11.128', 'inet'):ipv4 --Kishore's IP Address
BETWEEN start_ip_int AND end_ip_int;


--Join the log and location tables to add time zone to each row using the PARSE_IP function.
select logs.*
      , loc.city
      , loc.region
      , loc.country
      , loc.timezone
from AGS_GAME_AUDIENCE.RAW.LOGS logs
join IPINFO_GEOLOC.demo.location loc
where parse_ip(logs.ip_address, 'inet'):ipv4
BETWEEN start_ip_int AND end_ip_int;

--Use two functions supplied by IPShare to help with an efficient IP Lookup Process!
SELECT logs.ip_address
, logs.user_login
, logs.user_event
, logs.datetime_iso8601
, city
, region
, country
, timezone
from AGS_GAME_AUDIENCE.RAW.LOGS logs
JOIN IPINFO_GEOLOC.demo.location loc
ON IPINFO_GEOLOC.public.TO_JOIN_KEY(logs.ip_address) = loc.join_key
AND IPINFO_GEOLOC.public.TO_INT(logs.ip_address)
BETWEEN start_ip_int AND end_ip_int;


SELECT logs.ip_address
, logs.user_login
, logs.user_event
, logs.datetime_iso8601
, city
, region
, country
, timezone
, convert_timezone('UTC', timezone, logs.datetime_iso8601)  as game_event_ltz
, dayname(game_event_ltz) as dow_name
from AGS_GAME_AUDIENCE.RAW.LOGS logs
```

```
JOIN IPINFO_GEOLOC.demo.location loc
ON IPINFO_GEOLOC.public.TO_JOIN_KEY(logs.ip_address) = loc.join_key
AND IPINFO_GEOLOC.public.TO_INT(logs.ip_address)
BETWEEN start_ip_int AND end_ip_int;


--a Look Up table to convert from hour number to "time of day name"
create table ags_game_audience.raw.time_of_day_lu
(  hour number
  ,tod_name varchar(25)
);

--insert statement to add all 24 rows to the table
insert into time_of_day_lu
values
(6,'Early morning'),
(7,'Early morning'),
(8,'Early morning'),
(9,'Mid-morning'),
(10,'Mid-morning'),
(11,'Late morning'),
(12,'Late morning'),
(13,'Early afternoon'),
(14,'Early afternoon'),
(15,'Mid-afternoon'),
(16,'Mid-afternoon'),
(17,'Late afternoon'),
(18,'Late afternoon'),
(19,'Early evening'),
(20,'Early evening'),
(21,'Late evening'),
(22,'Late evening'),
(23,'Late evening'),
(0,'Late at night'),
(1,'Late at night'),
(2,'Late at night'),
(3,'Toward morning'),
(4,'Toward morning'),
(5,'Toward morning');

select tod_name, listagg(hour,',')
from time_of_day_lu
group by tod_name;

--
create table ags_game_audience.enhanced.logs_enhanced as(

SELECT logs.ip_address
, logs.user_login as gamer_name
, logs.user_event as game_event_name
, logs.datetime_iso8601 as game_event_UTC
, city
, region
, country
, timezone as gamer_LTZ_name
, convert_timezone('UTC', timezone, logs.datetime_iso8601)  as game_event_ltz
, dayname(game_event_ltz) as dow_name
, tod_name
from AGS_GAME_AUDIENCE.RAW.LOGS logs
JOIN IPINFO_GEOLOC.demo.location loc
ON IPINFO_GEOLOC.public.TO_JOIN_KEY(logs.ip_address) = loc.join_key
AND IPINFO_GEOLOC.public.TO_INT(logs.ip_address)
BETWEEN start_ip_int AND end_ip_int
join ags_game_audience.raw.time_of_day_lu
on hour(game_event_ltz)=hour
);

select * from ags_game_audience.enhanced.logs_enhanced;
```

```
--
select GRADER(step, (actual = expected), actual, expected, description) as graded_results from
(
 SELECT
  'DNGW03' as step
  ,( select count(*)
    from ags_game_audience.enhanced.logs_enhancedf
    where dow_name = 'Sat'
    and tod_name = 'Early evening'
    and gamer_name like '%prajina'
   ) as actual
  ,2 as expected
  ,'Playing the game on a Saturday evening' as description
);


create task ags_game_audience.raw.load_logs_enhanced
   warehouse = 'compute_wh'
   schedule = '5 minute'
   as
    select 'hello';
show tasks in account;

execute task ags_game_audience.raw.load_logs_enhanced;

--You have to run this grant or you won't be able to test your tasks while in SYSADMIN role
--this is true even if SYSADMIN owns the task!!
use role accountadmin;
grant execute task on account to role SYSADMIN;

--Now you should be able to run the task, even if your role is set to SYSADMIN
use role sysadmin;
execute task AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED;

--the SHOW command might come in handy to look at the task
show tasks in account;

--you can also look at any task more in depth using DESCRIBE
describe task AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED;


create or replace  task ags_game_audience.raw.load_logs_enhanced
   warehouse = 'compute_wh'
   schedule = '5 minute'
   as SELECT logs.ip_address
, logs.user_login as gamer_name
, logs.user_event as game_event_name
, logs.datetime_iso8601 as game_event_UTC
, city
, region
, country
, timezone as gamer_LTZ_name
, convert_timezone('UTC', timezone, logs.datetime_iso8601)  as game_event_ltz
, dayname(game_event_ltz) as dow_name
, tod_name
from AGS_GAME_AUDIENCE.RAW.LOGS logs
JOIN IPINFO_GEOLOC.demo.location loc
ON IPINFO_GEOLOC.public.TO_JOIN_KEY(logs.ip_address) = loc.join_key
AND IPINFO_GEOLOC.public.TO_INT(logs.ip_address)
BETWEEN start_ip_int AND end_ip_int
join ags_game_audience.raw.time_of_day_lu
on hour(game_event_ltz)=hour;

--make a note of how many rows you have in the table
select count(*)
from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

--Run the task to load more rows
execute task AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED;
```

```
--check to see how many rows were added
select count(*)
from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

--
--first we dump all the rows out of the table
truncate table ags_game_audience.enhanced.LOGS_ENHANCED;

--then we put them all back in
INSERT INTO ags_game_audience.enhanced.LOGS_ENHANCED (
SELECT logs.ip_address
, logs.user_login as GAMER_NAME
, logs.user_event as GAME_EVENT_NAME
, logs.datetime_iso8601 as GAME_EVENT_UTC
, city
, region
, country
, timezone as GAMER_LTZ_NAME
, CONVERT_TIMEZONE( 'UTC',timezone,logs.datetime_iso8601) as game_event_ltz
, DAYNAME(game_event_ltz) as DOW_NAME
, TOD_NAME
from ags_game_audience.raw.LOGS logs
JOIN ipinfo_geoloc.demo.location loc
ON ipinfo_geoloc.public.TO_JOIN_KEY(logs.ip_address) = loc.join_key
AND ipinfo_geoloc.public.TO_INT(logs.ip_address)
BETWEEN start_ip_int AND end_ip_int
JOIN ags_game_audience.raw.TIME_OF_DAY_LU tod
ON HOUR(game_event_ltz) = tod.hour);

create table ags_game_audience.enhanced.LOGS_ENHANCED_UF
clone ags_game_audience.enhanced.LOGS_ENHANCED;


MERGE INTO ENHANCED.LOGS_ENHANCED e
USING RAW.LOGS r
ON r.user_login = e.GAMER_NAME
WHEN MATCHED THEN
UPDATE SET IP_ADDRESS = 'Hey I updated matching rows!';

select * from RAW.LOGS;
select * from ENHANCED.LOGS_ENHANCED;

MERGE INTO ENHANCED.LOGS_ENHANCED e
USING RAW.LOGS r
ON r.user_login = e.GAMER_NAME
and r.datetime_iso8601 = e.game_event_UTC
and r.user_event = e.game_event_name
WHEN MATCHED THEN
UPDATE SET IP_ADDRESS = 'Hey I updated matching rows!';


truncate table AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

merge into AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED e
  using (
    SELECT logs.ip_address
    , logs.user_login as GAMER_NAME
    , logs.user_event as GAME_EVENT_NAME
    , logs.datetime_iso8601 as GAME_EVENT_UTC
    , city
    , region
    , country
    , timezone as GAMER_LTZ_NAME
    , CONVERT_TIMEZONE( 'UTC',timezone,logs.datetime_iso8601) as game_event_ltz
    , DAYNAME(game_event_ltz) as DOW_NAME
    , TOD_NAME
    from ags_game_audience.raw.LOGS logs
    JOIN ipinfo_geoloc.demo.location loc
```

```
      ON ipinfo_geoloc.public.TO_JOIN_KEY(logs.ip_address) = loc.join_key
      AND ipinfo_geoloc.public.TO_INT(logs.ip_address)
      BETWEEN start_ip_int AND end_ip_int
      JOIN ags_game_audience.raw.TIME_OF_DAY_LU tod
      ON HOUR(game_event_ltz) = tod.hour
   ) r
ON r.GAMER_NAME = e.GAMER_NAME
and r.GAME_EVENT_UTC = e.game_event_UTC
and r.GAME_EVENT_NAME = e.game_event_name
WHEN NOT MATCHED THEN
insert (IP_address, gamer_name, game_event_name
     , game_event_UTC, city, region
     , country, gamer_ltz_name, game_event_ltz
     , dow_name, tod_name)
     values
     (IP_address, gamer_name, game_event_name
     , game_event_UTC, city, region
     , country, gamer_ltz_name, game_event_ltz
     , dow_name, tod_name);



create or replace task ags_game_audience.raw.load_logs_enhanced
   warehouse = 'compute_wh'
   schedule = '5 minute'
   as
merge into AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED e
   using (
      SELECT logs.ip_address
      , logs.user_login as GAMER_NAME
      , logs.user_event as GAME_EVENT_NAME
      , logs.datetime_iso8601 as GAME_EVENT_UTC
      , city
      , region
      , country
      , timezone as GAMER_LTZ_NAME
      , CONVERT_TIMEZONE( 'UTC',timezone,logs.datetime_iso8601) as game_event_ltz
      , DAYNAME(game_event_ltz) as DOW_NAME
      , TOD_NAME
      from ags_game_audience.raw.LOGS logs
      JOIN ipinfo_geoloc.demo.location loc
      ON ipinfo_geoloc.public.TO_JOIN_KEY(logs.ip_address) = loc.join_key
      AND ipinfo_geoloc.public.TO_INT(logs.ip_address)
      BETWEEN start_ip_int AND end_ip_int
      JOIN ags_game_audience.raw.TIME_OF_DAY_LU tod
      ON HOUR(game_event_ltz) = tod.hour
   ) r
ON r.GAMER_NAME = e.GAMER_NAME
and r.GAME_EVENT_UTC = e.game_event_UTC
and r.GAME_EVENT_NAME = e.game_event_name
WHEN NOT MATCHED THEN
insert (IP_address, gamer_name, game_event_name
     , game_event_UTC, city, region
     , country, gamer_ltz_name, game_event_ltz
     , dow_name, tod_name)
     values
     (IP_address, gamer_name, game_event_name
     , game_event_UTC, city, region
     , country, gamer_ltz_name, game_event_ltz
     , dow_name, tod_name);

EXECUTE TASK AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED;

--Testing cycle for MERGE. Use these commands to make sure the Merge works as expected

--Write down the number of records in your table
select * from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

--Run the Merge a few times. No new rows should be added at this time
```

```
EXECUTE TASK AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED;

--Check to see if your row count changed
select * from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

--Insert a test record into your Raw Table
--You can change the user_event field each time to create "new" records
--editing the ip_address or datetime_iso8601 can complicate things more than they need to
--editing the user_login will make it harder to remove the fake records after you finish testing
INSERT INTO ags_game_audience.raw.game_logs
select PARSE_JSON('{"datetime_iso8601":"2025-01-01 00:00:00.000", "ip_address":"196.197.196.255", "user_event":"fake event", "user_login":"fake user"}');

select * from ags_game_audience.raw.game_logs;
--After inserting a new row, run the Merge again
EXECUTE TASK AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED;

--Check to see if any rows were added
select * from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

--When you are confident your merge is working, you can delete the raw records
delete from ags_game_audience.raw.game_logs where raw_log like '%fake user%';

--You should also delete the fake rows from the enhanced table
delete from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED
where gamer_name = 'fake user';

--Row count should be back to what it was in the beginning
select * from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

--
select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as graded_results from
(
SELECT
'DNGW04' as step
 ,(select count(*)/count(*)
  from table(ags_game_audience.information_schema.task_history
           (task_name=>'LOAD_LOGS_ENHANCED'))) as actual
 ,1 as expected
 ,'Task exists and has been run at least once' as description
 );


-- data pipeline
create or replace stage raw.UNI_KISHORE_PIPELINE
   url='s3://uni-kishore-pipeline';
list @raw.UNI_KISHORE_PIPELINE;
create or replace table raw.PIPELINE_LOGS (
   raw_log variant
);
copy into raw.PIPELINE_LOGS
  from @raw.UNI_KISHORE_PIPELINE
  file_format=(format_name=ff_json_logs);
select * from PIPELINE_LOGS;

create or replace view AGS_GAME_AUDIENCE.RAW.PL_LOGS(
         IP_ADDRESS,
         USER_LOGIN,
         USER_EVENT,
         DATETIME_ISO8601,
         RAW_LOG
) as
  select raw_log:ip_address::text as ip_address,
      raw_log:user_login::text as user_login,
      raw_log:user_event::text as user_event,
      raw_log:datetime_iso8601::timestamp_ntz as datetime_iso8601,
      raw_log
   from PIPELINE_LOGS
   where ip_address is not null;
```

```
select * from PL_LOGS;

-- task
create or replace task ags_game_audience.raw.load_logs_enhanced
   warehouse = 'compute_wh'
   schedule = '5 minute'
   as
merge into AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED e
   using (
      SELECT logs.ip_address
      , logs.user_login as GAMER_NAME
      , logs.user_event as GAME_EVENT_NAME
      , logs.datetime_iso8601 as GAME_EVENT_UTC
      , city
      , region
      , country
      , timezone as GAMER_LTZ_NAME
      , CONVERT_TIMEZONE( 'UTC',timezone,logs.datetime_iso8601) as game_event_ltz
      , DAYNAME(game_event_ltz) as DOW_NAME
      , TOD_NAME
      from ags_game_audience.raw.PL_LOGS logs
      JOIN ipinfo_geoloc.demo.location loc
      ON ipinfo_geoloc.public.TO_JOIN_KEY(logs.ip_address) = loc.join_key
      AND ipinfo_geoloc.public.TO_INT(logs.ip_address)
      BETWEEN start_ip_int AND end_ip_int
      JOIN ags_game_audience.raw.TIME_OF_DAY_LU tod
      ON HOUR(game_event_ltz) = tod.hour
   ) r
ON r.GAMER_NAME = e.GAMER_NAME
and r.GAME_EVENT_UTC = e.game_event_UTC
and r.GAME_EVENT_NAME = e.game_event_name
WHEN NOT MATCHED THEN
insert (IP_address, gamer_name, game_event_name
      , game_event_UTC, city, region
      , country, gamer_ltz_name, game_event_ltz
      , dow_name, tod_name)
      values
      (IP_address, gamer_name, game_event_name
      , game_event_UTC, city, region
      , country, gamer_ltz_name, game_event_ltz
      , dow_name, tod_name);


create or replace task raw.get_new_files
   warehouse = compute_wh
   schedule = '5 minute'
   as
   copy into raw.PIPELINE_LOGS
    from @raw.UNI_KISHORE_PIPELINE
    file_format=(format_name=ff_json_logs);

select * from PIPELINE_LOGS;
execute task raw.get_new_files;

--
select * from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;
truncate table AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;
show tasks;
--Turning on a task is done with a RESUME command
alter task AGS_GAME_AUDIENCE.RAW.GET_NEW_FILES resume;
alter task AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED resume;

select * from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;
select count(*) from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

--Keep this code handy for shutting down the tasks each day
alter task AGS_GAME_AUDIENCE.RAW.GET_NEW_FILES suspend;
alter task AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED suspend;
```

```
list @UNI_KISHORE_PIPELINE;
select count(*) from PIPELINE_LOGS;
select count(*) from PL_LOGS;
select count(*) from ags_game_audience.enhanced.LOGS_ENHANCED;

--
-- tasks run every 5 minutes that spin up WH to run the task.
-- go to serverless task
use role accountadmin;
grant EXECUTE MANAGED TASK on account to SYSADMIN;

--switch back to sysadmin
use role sysadmin;

create or replace task raw.get_new_files
   USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
   schedule = '5 Minutes'
   as
   copy into raw.PIPELINE_LOGS
    from @raw.UNI_KISHORE_PIPELINE
    file_format=(format_name=ff_json_logs);

create or replace task ags_game_audience.raw.load_logs_enhanced
   USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
   after ags_game_audience.raw.get_new_files
   as
merge into AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED e
   using (
      SELECT logs.ip_address
      , logs.user_login as GAMER_NAME
      , logs.user_event as GAME_EVENT_NAME
      , logs.datetime_iso8601 as GAME_EVENT_UTC
      , city
      , region
      , country
      , timezone as GAMER_LTZ_NAME
      , CONVERT_TIMEZONE( 'UTC',timezone,logs.datetime_iso8601) as game_event_ltz
      , DAYNAME(game_event_ltz) as DOW_NAME
      , TOD_NAME
      from ags_game_audience.raw.PL_LOGS logs
      JOIN ipinfo_geoloc.demo.location loc
      ON ipinfo_geoloc.public.TO_JOIN_KEY(logs.ip_address) = loc.join_key
      AND ipinfo_geoloc.public.TO_INT(logs.ip_address)
      BETWEEN start_ip_int AND end_ip_int
      JOIN ags_game_audience.raw.TIME_OF_DAY_LU tod
      ON HOUR(game_event_ltz) = tod.hour
   ) r
ON r.GAMER_NAME = e.GAMER_NAME
and r.GAME_EVENT_UTC = e.game_event_UTC
and r.GAME_EVENT_NAME = e.game_event_name
WHEN NOT MATCHED THEN
insert (IP_address, gamer_name, game_event_name
      , game_event_UTC, city, region
      , country, gamer_ltz_name, game_event_ltz
      , dow_name, tod_name)
      values
      (IP_address, gamer_name, game_event_name
      , game_event_UTC, city, region
      , country, gamer_ltz_name, game_event_ltz
      , dow_name, tod_name);


alter task AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED resume;
alter task AGS_GAME_AUDIENCE.RAW.GET_NEW_FILES resume;


alter task AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED suspend;
alter task AGS_GAME_AUDIENCE.RAW.GET_NEW_FILES suspend;
show tasks;
```

```
--
select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as graded_results from
(
SELECT
'DNGW05' as step
 ,(
  select max(tally) from (
     select CASE WHEN SCHEDULED_FROM = 'SCHEDULE'
                 and STATE= 'SUCCEEDED'
          THEN 1 ELSE 0 END as tally
  from table(ags_game_audience.information_schema.task_history (task_name=>'GET_NEW_FILES')))
 ) as actual
,1 as expected
,'Task succeeds from schedule' as description
 );

 ----
 select METADATA$FILENAME as log_file_name
  FROM @AGS_GAME_AUDIENCE.RAW.UNI_KISHORE_PIPELINE
 (file_format => 'ff_json_logs');


SELECT
    METADATA$FILENAME as log_file_name --new metadata column
  , METADATA$FILE_ROW_NUMBER as log_file_row_id --new metadata column
  , current_timestamp(0) as load_ltz --new local time of load
  , get($1,'datetime_iso8601')::timestamp_ntz as DATETIME_ISO8601
  , get($1,'user_event')::text as USER_EVENT
  , get($1,'user_login')::text as USER_LOGIN
  , get($1,'ip_address')::text as IP_ADDRESS
  FROM @AGS_GAME_AUDIENCE.RAW.UNI_KISHORE_PIPELINE
  (file_format => 'ff_json_logs');


create or replace table ED_PIPELINE_LOGS as
 SELECT
    METADATA$FILENAME as log_file_name --new metadata column
  , METADATA$FILE_ROW_NUMBER as log_file_row_id --new metadata column
  , current_timestamp(0) as load_ltz --new local time of load
  , get($1,'datetime_iso8601')::timestamp_ntz as DATETIME_ISO8601
  , get($1,'user_event')::text as USER_EVENT
  , get($1,'user_login')::text as USER_LOGIN
  , get($1,'ip_address')::text as IP_ADDRESS
  FROM @AGS_GAME_AUDIENCE.RAW.UNI_KISHORE_PIPELINE
  (file_format => 'ff_json_logs');
 select * from ED_PIPELINE_LOGS;

 --truncate the table rows that were input during the CTAS
truncate table ED_PIPELINE_LOGS;

--reload the table using your COPY INTO <<<<<<<<<< copy into table from a SELECT reulsts which runs on un-loaded json files.
COPY INTO ED_PIPELINE_LOGS
FROM (
    SELECT
    METADATA$FILENAME as log_file_name
  , METADATA$FILE_ROW_NUMBER as log_file_row_id
  , current_timestamp(0) as load_ltz
  , get($1,'datetime_iso8601')::timestamp_ntz as DATETIME_ISO8601
  , get($1,'user_event')::text as USER_EVENT
  , get($1,'user_login')::text as USER_LOGIN
  , get($1,'ip_address')::text as IP_ADDRESS
  FROM @AGS_GAME_AUDIENCE.RAW.UNI_KISHORE_PIPELINE
)
file_format = (format_name = ff_json_logs);


-- snowpipe
CREATE OR REPLACE PIPE GET_NEW_FILES
```

```
auto_ingest=true
aws_sns_topic='arn:aws:sns:us-west-2:321463406630:dngw_topic'
AS
COPY INTO ED_PIPELINE_LOGS
FROM (
   SELECT
   METADATA$FILENAME as log_file_name
  , METADATA$FILE_ROW_NUMBER as log_file_row_id
  , current_timestamp(0) as load_ltz
  , get($1,'datetime_iso8601')::timestamp_ntz as DATETIME_ISO8601
  , get($1,'user_event')::text as USER_EVENT
  , get($1,'user_login')::text as USER_LOGIN
  , get($1,'ip_address')::text as IP_ADDRESS
  FROM @AGS_GAME_AUDIENCE.RAW.UNI_KISHORE_PIPELINE
)
file_format = (format_name = ff_json_logs);

create or replace task ags_game_audience.raw.load_logs_enhanced
   USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
   schedule = '5 minutes'
   as
merge into AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED e
   using (
      SELECT logs.ip_address
      , logs.user_login as GAMER_NAME
      , logs.user_event as GAME_EVENT_NAME
      , logs.datetime_iso8601 as GAME_EVENT_UTC
      , city
      , region
      , country
      , timezone as GAMER_LTZ_NAME
      , CONVERT_TIMEZONE( 'UTC',timezone,logs.datetime_iso8601) as game_event_ltz
      , DAYNAME(game_event_ltz) as DOW_NAME
      , TOD_NAME
      from ags_game_audience.raw.ED_PIPELINE_LOGS logs
      JOIN ipinfo_geoloc.demo.location loc
      ON ipinfo_geoloc.public.TO_JOIN_KEY(logs.ip_address) = loc.join_key
      AND ipinfo_geoloc.public.TO_INT(logs.ip_address)
      BETWEEN start_ip_int AND end_ip_int
      JOIN ags_game_audience.raw.TIME_OF_DAY_LU tod
      ON HOUR(game_event_ltz) = tod.hour
   ) r
ON r.GAMER_NAME = e.GAMER_NAME
and r.GAME_EVENT_UTC = e.game_event_UTC
and r.GAME_EVENT_NAME = e.game_event_name
WHEN NOT MATCHED THEN
insert (IP_address, gamer_name, game_event_name
      , game_event_UTC, city, region
      , country, gamer_ltz_name, game_event_ltz
      , dow_name, tod_name)
      values
      (IP_address, gamer_name, game_event_name
      , game_event_UTC, city, region
      , country, gamer_ltz_name, game_event_ltz
      , dow_name, tod_name);


alter task AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED resume;


select count(*) from ED_PIPELINE_LOGS;
select count(*) from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

--create a stream that will keep track of changes to the table
create or replace stream ags_game_audience.raw.ed_cdc_stream
on table AGS_GAME_AUDIENCE.RAW.ED_PIPELINE_LOGS;

--look at the stream you created
show streams;
```

```
--check to see if any changes are pending
select system$stream_has_data('ed_cdc_stream');

--query the stream
select *
from ags_game_audience.raw.ed_cdc_stream;

--check to see if any changes are pending
select system$stream_has_data('ed_cdc_stream');

--if your stream remains empty for more than 10 minutes, make sure your PIPE is running
select SYSTEM$PIPE_STATUS('GET_NEW_FILES');

--if you need to pause or unpause your pipe
--alter pipe GET_NEW_FILES set pipe_execution_paused = true;
--alter pipe GET_NEW_FILES set pipe_execution_paused = false;



--make a note of how many rows are in the stream
select *
from ags_game_audience.raw.ed_cdc_stream;


--process the stream by using the rows in a merge
MERGE INTO AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED e
USING (
      SELECT cdc.ip_address
      , cdc.user_login as GAMER_NAME
      , cdc.user_event as GAME_EVENT_NAME
      , cdc.datetime_iso8601 as GAME_EVENT_UTC
      , city
      , region
      , country
      , timezone as GAMER_LTZ_NAME
      , CONVERT_TIMEZONE( 'UTC',timezone,cdc.datetime_iso8601) as game_event_ltz
      , DAYNAME(game_event_ltz) as DOW_NAME
      , TOD_NAME
      from ags_game_audience.raw.ed_cdc_stream cdc
      JOIN ipinfo_geoloc.demo.location loc
      ON ipinfo_geoloc.public.TO_JOIN_KEY(cdc.ip_address) = loc.join_key
      AND ipinfo_geoloc.public.TO_INT(cdc.ip_address)
      BETWEEN start_ip_int AND end_ip_int
      JOIN AGS_GAME_AUDIENCE.RAW.TIME_OF_DAY_LU tod
      ON HOUR(game_event_ltz) = tod.hour
    ) r
ON r.GAMER_NAME = e.GAMER_NAME
AND r.GAME_EVENT_UTC = e.GAME_EVENT_UTC
AND r.GAME_EVENT_NAME = e.GAME_EVENT_NAME
WHEN NOT MATCHED THEN
INSERT (IP_ADDRESS, GAMER_NAME, GAME_EVENT_NAME
      , GAME_EVENT_UTC, CITY, REGION
      , COUNTRY, GAMER_LTZ_NAME, GAME_EVENT_LTZ
      , DOW_NAME, TOD_NAME)
      VALUES
      (IP_ADDRESS, GAMER_NAME, GAME_EVENT_NAME
      , GAME_EVENT_UTC, CITY, REGION
      , COUNTRY, GAMER_LTZ_NAME, GAME_EVENT_LTZ
      , DOW_NAME, TOD_NAME);

--Did all the rows from the stream disappear?
select *
from ags_game_audience.raw.ed_cdc_stream;


--
--turn off the other task (we won't need it anymore)
alter task AGS_GAME_AUDIENCE.RAW.LOAD_LOGS_ENHANCED suspend;
```

```
--Create a new task that uses the MERGE you just tested
create or replace task AGS_GAME_AUDIENCE.RAW.CDC_LOAD_LOGS_ENHANCED
        USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE='XSMALL'
        SCHEDULE = '5 minutes'
   when
      system$stream_has_data('ed_cdc_stream')
        as
MERGE INTO AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED e
USING (
      SELECT cdc.ip_address
      , cdc.user_login as GAMER_NAME
      , cdc.user_event as GAME_EVENT_NAME
      , cdc.datetime_iso8601 as GAME_EVENT_UTC
      , city
      , region
      , country
      , timezone as GAMER_LTZ_NAME
      , CONVERT_TIMEZONE( 'UTC',timezone,cdc.datetime_iso8601) as game_event_ltz
      , DAYNAME(game_event_ltz) as DOW_NAME
      , TOD_NAME
      from ags_game_audience.raw.ed_cdc_stream cdc
      JOIN ipinfo_geoloc.demo.location loc
      ON ipinfo_geoloc.public.TO_JOIN_KEY(cdc.ip_address) = loc.join_key
      AND ipinfo_geoloc.public.TO_INT(cdc.ip_address)
      BETWEEN start_ip_int AND end_ip_int
      JOIN AGS_GAME_AUDIENCE.RAW.TIME_OF_DAY_LU tod
      ON HOUR(game_event_ltz) = tod.hour
    ) r
ON r.GAMER_NAME = e.GAMER_NAME
AND r.GAME_EVENT_UTC = e.GAME_EVENT_UTC
AND r.GAME_EVENT_NAME = e.GAME_EVENT_NAME
WHEN NOT MATCHED THEN
INSERT (IP_ADDRESS, GAMER_NAME, GAME_EVENT_NAME
      , GAME_EVENT_UTC, CITY, REGION
      , COUNTRY, GAMER_LTZ_NAME, GAME_EVENT_LTZ
      , DOW_NAME, TOD_NAME)
      VALUES
      (IP_ADDRESS, GAMER_NAME, GAME_EVENT_NAME
      , GAME_EVENT_UTC, CITY, REGION
      , COUNTRY, GAMER_LTZ_NAME, GAME_EVENT_LTZ
      , DOW_NAME, TOD_NAME);

--Resume the task so it is running
alter task AGS_GAME_AUDIENCE.RAW.CDC_LOAD_LOGS_ENHANCED suspend;
alter task AGS_GAME_AUDIENCE.RAW.CDC_LOAD_LOGS_ENHANCED resume;

select count(*) from AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

truncate table AGS_GAME_AUDIENCE.ENHANCED.LOGS_ENHANCED;

alter pipe GET_NEW_FILES set pipe_execution_paused = true;

--
select DEMO_DB.PUBLIC.GRADER(step, (actual = expected), actual, expected, description) as graded_results from
(
SELECT
'DNGW06' as step
 ,(
  select CASE WHEN pipe_status:executionState::text = 'RUNNING' THEN 1 ELSE 0 END
  from(
  select parse_json(SYSTEM$PIPE_STATUS( 'ags_game_audience.raw.GET_NEW_FILES' )) as pipe_status)
 ) as actual
 ,1 as expected
 ,'Pipe exists and is RUNNING' as description
 );

create schema ags_game_audience.curated;
```