NOMENCLATURE

DATABASE:
DIGITAL_1BRONZE
DIGITAL_2SILVER
DIGITAL_3GOLD

SCHEMA:
DIGITAL_BSCHEMA
DIGITAL_SSCHEMA
DIGITAL_GSCHEMA

TABLES:
MONTHLY_DIGITAL_BTABLE
MONTHLY_DIGITAL_STABLE
MONTHLY_DIGITAL_GTABLE

STAGE:
MONTHLY_DIGITAL_STAGE

BRONZE TABLE

```
create or replace TABLE MONTHLY_DIGITAL_BTABLE (
        DISTRIBUTION STRING,
        LAST_PING_TIMESTAMP TIMESTAMP,
        HOST STRING,
        COOKIE_ID STRING,
        PAGE_SESSION_ID STRING,
        DOMAIN STRING,
        PATH STRING,
        NEW_USER BOOLEAN,
        DEVICE STRING,
        ENGAGED_TIME_ON_PAGE_SECONDS NUMBER(38,0),
        PAGE_WIDTH NUMBER(38,0),
        PAGE_HEIGHT NUMBER(38,0),
        MAX_SCROLL_POSITION_TOP NUMBER(38,0),
        WINDOW_HEIGHT NUMBER(38,0),
        EXTERNAL_REFERRER STRING,
        NO_CLIENT_STORAGE BOOLEAN,
```

```
        CITY_NAME STRING,
        COUNTRY_CODE STRING,
        COUNTRY_NAME STRING,
        CONTINENT_NAME STRING,
        DMA_CODE NUMBER(38,0),
        UTC_OFFSET_MINUTES NUMBER(38,0),
        USER_AGENT VARCHAR(16777216),
        RECENCY NUMBER(38,0),
        FREQUENCY NUMBER(38,0),
    INTERNAL_REFERRER STRING,
        AUTHOR STRING,
        SECTION STRING,
        CONTENT_TYPE STRING,
        SPONSOR STRING,
        UTM_CAMPAIGN STRING,
        UTM_MEDIUM STRING,
        UTM_SOURCE STRING,
        UTM_CONTENT STRING,
        UTM_TERM STRING,
        ACCOUNT_ID NUMBER(38,0),
        PAGE_TITLE STRING,
        VIRTUAL_PAGE BOOLEAN,
        SCROLLDEPTH NUMBER(38,0),
        TOTAL_TIME_ON_PAGE_SECONDS NUMBER(38,0),
        GA_CLIENT_ID STRING,
        LOGIN_ID STRING,
        ID_SYNC VARCHAR(16777216),
        SUBSCRIBER_ACCT NUMBER(38,0),
        PAGE_LOAD_TIME NUMBER(38,0)
);
```

COPY INTO COMMAND

```
/copy into command
copy into MONTHLY_DIGITAL_BTABLE
from @MONTHLY_DIGITAL_STAGE
file_format = (
      TYPE=CSV,
      SKIP_HEADER=1,
      FIELD_DELIMITER=',',
      TRIM_SPACE=false,
```

```
      FIELD_OPTIONALLY_ENCLOSED_BY='"',
      REPLACE_INVALID_CHARACTERS=TRUE,
      DATE_FORMAT='DD-MM-YYYY',
      TIME_FORMAT=AUTO,
      TIMESTAMP_FORMAT=AUTO
    );
```

PROCEDURE : STAGE TO TABLE BRONZE - WEDNESDAY 2AM

```
CREATE OR REPLACE PROCEDURE stage_to_table_digital()
RETURNS STRING
LANGUAGE JAVASCRIPT
EXECUTE AS CALLER
AS
$$
var task_name = "stage_to_table_task";
var stage_name = "MONTHLY_DIGITAL_STAGE";
var final_table = "MONTHYL_DIGITAL_BTABLE";

// Define the SQL command to copy data from the stage to the table
var copy_sql = `
COPY INTO ${final_table}
FROM @${stage_name}
FILE_FORMAT = (
   TYPE = 'CSV',
   SKIP_HEADER = 1,
   FIELD_DELIMITER = ',',
   TRIM_SPACE = FALSE,
   FIELD_OPTIONALLY_ENCLOSED_BY = '"',
   REPLACE_INVALID_CHARACTERS = TRUE,
   DATE_FORMAT = 'DD-MM-YYYY',
   TIME_FORMAT = 'AUTO'
)
ON_ERROR = 'CONTINUE';
`;

// Define the SQL command to create the task
var task_sql = `
   CREATE OR REPLACE TASK ${task_name}
   WAREHOUSE = COMPUTE_WH
   SCHEDULE = 'USING CRON 0 2 * * 3 UTC'
   COMMENT = 'Push new data from stage to BRONZE TABLE'
```

```
    AS
    ${copy_sql}
`;

// Execute the task creation SQL
try {
    var stmt = snowflake.createStatement({ sqlText: task_sql });
    stmt.execute();
    return "Task created successfully.";
} catch (err) {
    return "Error creating the task: " + err;
}
$$;

-- Execute the stored procedure
CALL stage_to_table_digital();
```

```
create or replace procedure stg_to_bronze()
returns string
language javascript
execute as caller
as
$$
  var final_table = 'MONTHLY_DIGITAL_BTABLE';
  var stage_name = 'MONTHLY_DIGITAL_STAGE';
  var task_name = "stage_to_bronze";

  var task_sql = `
    create or replace task ${task_name}
    warehouse = compute_wh
    schedule = 'using cron 0 2 * * 3 utc'
    comment = 'push new data into table'
    as
    begin
      copy into ${final_table}
      from @${stage_name}
      file_format = (
        TYPE=CSV,
        SKIP_HEADER=1,
        FIELD_DELIMITER=',',
        TRIM_SPACE=false,
        FIELD_OPTIONALLY_ENCLOSED_BY="",
```

```
      REPLACE_INVALID_CHARACTERS=TRUE,
      DATE_FORMAT='DD-MM-YYYY',
      TIME_FORMAT=AUTO,
      TIMESTAMP_FORMAT=AUTO
    );
  end;
`;

 try{
    snowflake.execute({sqlText:task_sql});
    return "task created successfully";
  }catch(err){
    return "error creating the task: "+err;
  }
$$;
```

Finding percentage of missing values in each column - using snowpark

```
import snowflake.snowpark as snowpark
from snowflake.snowpark.functions import col, count, when

def main(session: snowpark.Session):
    table_name = 'MONTHLY_DIGITAL_BTABLE'
    dataframe = session.table(table_name)

    # Get the total number of rows
    total_rows = dataframe.count()

    # Calculate the percentage of missing values for each column
    missing_value_percentage = {}
    for column in dataframe.columns:
        missing_count = dataframe.select(count(when(col(column).is_null(), 1))).collect()[0][0]
        missing_percentage = (missing_count / total_rows) * 100 if total_rows > 0 else 0
        missing_value_percentage[column] = missing_percentage

    # Convert the results to a DataFrame for better readability
    results_df = session.create_dataframe(
```

```
        [(col_name, missing_value_percentage[col_name]) for col_name in
missing_value_percentage.keys()],
        schema=["Column_Name", "Missing_Percentage"]
    )

    # Return the results DataFrame
    return results_df
```

Result:

COLUMN_NAME	MISSING_PERCENTAGE
DISTRIBUTION	0
LAST_PING_TIMESTAMP	0
HOST	0
COOKIE_ID	0
PAGE_SESSION_ID	0
DOMAIN	0
PATH	0
NEW_USER	0
DEVICE	0
ENGAGED_TIME_ON_PAGE_SECONDS	0
PAGE_WIDTH	50.310315739
PAGE_HEIGHT	0
MAX_SCROLL_POSITION_TOP	0
WINDOW_HEIGHT	0
EXTERNAL_REFERRER	29.592472082
NO_CLIENT_STORAGE	0
CITY_NAME	13.21201896
COUNTRY_CODE	0.09239174098
COUNTRY_NAME	0.09239174098

CONTINENT_NAME	0.02912348357
DMA_CODE	0
UTC_OFFSET_MINUTES	0
USER_AGENT	0
RECENCY	0
FREQUENCY	0
INTERNAL_REFERRER	85.53366273
AUTHOR	0
SECTION	0
CONTENT_TYPE	0.01807664497
SPONSOR	100
UTM_CAMPAIGN	99.900578453
UTM_MEDIUM	99.894552904
UTM_SOURCE	99.89154013
UTM_CONTENT	99.998995742
UTM_TERM	100
ACCOUNT_ID	0
PAGE_TITLE	6.011488712
VIRTUAL_PAGE	0
SCROLLDEPTH	0
TOTAL_TIME_ON_PAGE_SECONDS	50.310315739
GA_CLIENT_ID	100
LOGIN_ID	100
ID_SYNC	100
SUBSCRIBER_ACCT	53.535992609

PAGE_LOAD_TIME 0

DATA RETENTION (TIME TRAVEL FEATURE)

```
## Task to be run at 1AM every wednesday

CREATE OR REPLACE PROCEDURE create_retention_task()
RETURNS STRING
LANGUAGE JAVASCRIPT
EXECUTE AS CALLER
AS
$$
var task_name = "data_retention_task";
var final_table = "MONTHLY_DIGITAL_BTABLE";

// Calculate the current timestamp and format it
var now = new Date(); // Use the current time
var current_timestamp = now.toISOString().slice(0, 19).replace('T', '_').replace(/[-:]/g, ''); //
Format the timestamp

// Define the snapshot table name
var snapshot_name = `${final_table}_SNAPSHOT_${current_timestamp}`;

var task_sql = `
CREATE OR REPLACE TASK ${task_name}
WAREHOUSE = COMPUTE_WH  -- Specify your warehouse
SCHEDULE = 'USING CRON 0 1 * * 3 UTC' -- Run at 1 AM UTC every Wednesday
COMMENT = 'Create a snapshot of the table for data retention'
AS
CREATE TABLE ${snapshot_name} CLONE ${final_table};
`;

try {
  var stmt = snowflake.createStatement({ sqlText: task_sql });
  stmt.execute();

  return "Retention task created successfully";
} catch (err) {
  return "Error creating the retention task: " + err;
}
$$;
```

- Call the procedure to create the task
CALL create_retention_task();

—----------

SILVER TABLE

```
create or replace TABLE
DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE (
        DISTRIBUTION STRING,
        LAST_PING_TIMESTAMP TIMESTAMP,
        COOKIE_ID STRING,
        DEVICE STRING,
        ENGAGED_TIME_ON_PAGE_SECONDS NUMBER(38,0),
        COUNTRY_NAME STRING,
        FREQUENCY NUMBER(38,0),
        AUTHOR STRING,
        SECTION STRING,
        CONTENT_TYPE STRING,
        SPONSOR STRING,
        ACCOUNT_ID NUMBER(38,0),
        PAGE_TITLE STRING,
        SCROLLDEPTH NUMBER(38,0),
        TOTAL_TIME_ON_PAGE_SECONDS NUMBER(38,0),
        SUBSCRIBER_ACCT NUMBER(38,0),
        PAGE_LOAD_TIME NUMBER(38,0)
);
```

BRONZE TO SILVER PROCEDURE

```
CREATE OR REPLACE PROCEDURE bronze_to_silver_digital()
RETURNS STRING
LANGUAGE JAVASCRIPT
EXECUTE AS CALLER
AS
$$
   var sourceTable =
"DIGITAL_1BRONZE.DIGITAL_BSCHEMA.MONTHLY_DIGITAL_BTABLE";
   var targetTable = "DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE";
```

```
    var task_name = "bronze_to_silver_digital_task";

    var insertSQL = `
        INSERT INTO ${targetTable}(DISTRIBUTION, LAST_PING_TIMESTAMP, COOKIE_ID,
DEVICE, ENGAGED_TIME_ON_PAGE_SECONDS, COUNTRY_NAME,
        FREQUENCY, AUTHOR, SECTION, CONTENT_TYPE, SPONSOR, ACCOUNT_ID,
PAGE_TITLE, SCROLLDEPTH, TOTAL_TIME_ON_PAGE_SECONDS, SUBSCRIBER_ACCT,
PAGE_LOAD_TIME)
        SELECT DISTRIBUTION, LAST_PING_TIMESTAMP, COOKIE_ID, DEVICE,
ENGAGED_TIME_ON_PAGE_SECONDS, COUNTRY_NAME,
        FREQUENCY, AUTHOR, SECTION, CONTENT_TYPE, SPONSOR, ACCOUNT_ID,
PAGE_TITLE, SCROLLDEPTH, TOTAL_TIME_ON_PAGE_SECONDS, SUBSCRIBER_ACCT,
PAGE_LOAD_TIME
        FROM ${sourceTable};
    `;

    var task_sql = `
        CREATE OR REPLACE TASK ${task_name}
        WAREHOUSE = COMPUTE_WH
        SCHEDULE = 'USING CRON 16 8 * * * UTC'
        AS
        ${insertSQL}
    `;

    try {
        var stmt = snowflake.createStatement({sqlText: task_sql});
        stmt.execute();
        return "Task created successfully";
    } catch (err) {
        return "Error while creating task: " + err;
    }
$$;

-- Call the procedure to create the task
CALL bronze_to_silver_digital();

ALTER TASK bronze_to_silver_digital_task RESUME;
```

DATA TRANSFORMATION IN SILVER

CATEGORIES COULMN

```
ALTER TABLE MONTHLY_DIGITAL_STABLE ADD COLUMN CATEGORIES VARCHAR;

CREATE OR REPLACE PROCEDURE create_categories()
RETURNS STRING
LANGUAGE JAVASCRIPT
EXECUTE AS CALLER
AS
$$
   var sourceTable = "DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE";
   var task_name = "create_categories_task";


    var updateTableSQL = `
      UPDATE ${sourceTable}
      SET CATEGORIES = CASE
         WHEN SECTION LIKE 'news%' THEN 'News'
         WHEN SECTION LIKE 'features%' THEN 'Features'
         WHEN SECTION LIKE 'education%' THEN 'Education'
         WHEN SECTION LIKE 'movies & music%' THEN 'Movies & Music'
         WHEN SECTION LIKE 'homepage' THEN 'Homepage'
         WHEN SECTION LIKE 'special pages%' THEN 'Special Pages'
         WHEN SECTION LIKE 'sports%' THEN 'Sports'
         WHEN SECTION LIKE 'agriculture%' THEN 'Agriculture'
         WHEN SECTION LIKE 'health%' THEN 'Health'
         WHEN SECTION LIKE 'technology%' THEN 'Technology'
         WHEN SECTION LIKE 'books%' THEN 'Books'
         WHEN SECTION LIKE 'travel%' THEN 'Travel'
         WHEN SECTION LIKE 'columns%' THEN 'Columns'
         WHEN SECTION LIKE 'multimedia%' THEN 'Multimedia'
         WHEN SECTION LIKE 'money%' THEN 'Money'
         WHEN SECTION LIKE 'election%' THEN 'Election'
         WHEN SECTION LIKE 'topics%' OR SECTION = 'discover' THEN 'Discover/Topics'
         WHEN SECTION = 'english homepage' OR SECTION = 'samachar homepage' THEN
'English Homepage'
         ELSE 'Other'
      END;
   `;

   var taskSQL = `
      CREATE OR REPLACE TASK ${task_name}
      WAREHOUSE = COMPUTE_WH
      SCHEDULE='USING CRON 53 13 * * * UTC'
      AS
      ${updateTableSQL}
```

```
    `;

    try {

          var stmt1 = snowflake.createStatement({sqlText: updateTableSQL});
        stmt1.execute();
        var stmt2 = snowflake.createStatement({sqlText: taskSQL});
        stmt2.execute();
    } catch (err) {
        return "Error while creating task: " + err;
    }
$$;




CALL create_categories();
ALTER TASK create_categories_task RESUME;
```

DATE AND HOUR Column

```
ALTER TABLE DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE
ADD COLUMN DATE DATE;
ALTER TABLE DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE
ADD COLUMN HOUR INT;

CREATE OR REPLACE PROCEDURE create_date_hour()
RETURNS STRING
LANGUAGE JAVASCRIPT
EXECUTE AS CALLER
AS
$$
    var sourceTable = "DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE";
    var task_name = "create_date_hour_task";



     var updateTableSQL = `
        UPDATE ${sourceTable}
        SET DATE = TO_DATE(TO_TIMESTAMP(LAST_PING_TIMESTAMP)),
        HOUR = EXTRACT(HOUR FROM TO_TIMESTAMP(LAST_PING_TIMESTAMP));
```

```
    `;

    var taskSQL = `
        CREATE OR REPLACE TASK ${task_name}
        WAREHOUSE = COMPUTE_WH
        SCHEDULE='USING CRON 56 13 * * * UTC'
        AS
        ${updateTableSQL}
    `;

    try {

          var stmt1 = snowflake.createStatement({sqlText: updateTableSQL});
        stmt1.execute();
        var stmt2 = snowflake.createStatement({sqlText: taskSQL});
        stmt2.execute();
    } catch (err) {
        return "Error while creating task: " + err;
    }
$$;

CALL create_date_hour();
ALTER TASK create_date_hour_task resume;
```

ENGAGEMENT_SCORE Column

```
ALTER TABLE DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE
ADD COLUMN ENGAGEMENT_SCORE DECIMAL(5, 2);


CREATE OR REPLACE PROCEDURE calculate_engagement_score()
RETURNS STRING
LANGUAGE JAVASCRIPT
EXECUTE AS CALLER
AS
$$
    var sourceTable = "DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE";
    var task_name = "calculate_engagement_score_task";
```

```
    var updateTableSQL = `
      WITH MaxValues AS (
   SELECT
      MAX(ENGAGED_TIME_ON_PAGE_SECONDS) AS MAX_ENGAGED_TIME,
      MAX(SCROLLDEPTH) AS MAX_SCROLL_DEPTH,
      MAX(PAGE_LOAD_TIME) AS MAX_PAGE_LOAD_TIME
   FROM MONTHLY_DIGITAL_BTABLE
)

SELECT mb.*,
   (0.4 * (mb.ENGAGED_TIME_ON_PAGE_SECONDS / mv.MAX_ENGAGED_TIME) * 100 +
    0.3 * (mb.SCROLLDEPTH / mv.MAX_SCROLL_DEPTH) * 100 +
    0.3 * (mb.PAGE_LOAD_TIME / mv.MAX_PAGE_LOAD_TIME) * 100) AS
ENGAGEMENT_SCORE
FROM MONTHLY_DIGITAL_BTABLE mb, MaxValues mv;
   `;

   var taskSQL = `
      CREATE OR REPLACE TASK ${task_name}
      WAREHOUSE = COMPUTE_WH
      SCHEDULE='USING CRON 50 8 * * * UTC'
      AS
      ${updateTableSQL}
   `;

   try {

       var stmt1 = snowflake.createStatement({sqlText: updateTableSQL});
      stmt1.execute();
      var stmt2 = snowflake.createStatement({sqlText: taskSQL});
      stmt2.execute();
   } catch (err) {
      return "Error while creating task: " + err;
   }
$$;

CALL calculate_engagement_score();
ALTER TASK calculate_engagement_score_task RESUME;
```

Creating the Path_Journey Column

```
ALTER TABLE monthly_digital_stable ADD COLUMN USER_JOURNEY_PATH VARCHAR;

CREATE OR REPLACE PROCEDURE calculate_user_journey_path()
RETURNS VARCHAR
LANGUAGE JAVASCRIPT
execute as caller
AS
$$

var sourceTable = "DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE";
var task_name = "calculate_user_journey_path_task";

var taskSQL = `
      CREATE OR REPLACE TASK ${task_name}
      WAREHOUSE = COMPUTE_WH
      SCHEDULE='USING CRON 45 14 * * * UTC'
      AS
      begin

       CREATE OR REPLACE TEMPORARY TABLE TEMP_USER_JOURNEY_PATH AS
        SELECT
           PAGE_SESSION_ID,
           LISTAGG(page_title, ' -> ') WITHIN GROUP (ORDER BY LAST_PING_TIMESTAMP)
AS USER_JOURNEY_PATH
        FROM
           ${sourceTable}
        GROUP BY
           PAGE_SESSION_ID;


        MERGE INTO ${sourceTable} AS target
        USING TEMP_USER_JOURNEY_PATH AS source
        ON target.PAGE_SESSION_ID = source.PAGE_SESSION_ID
        WHEN MATCHED THEN
           UPDATE SET USER_JOURNEY_PATH = source.USER_JOURNEY_PATH;

        -- Drop the temporary table
        DROP TABLE IF EXISTS TEMP_USER_JOURNEY_PATH;
```

```
        END;
    `;


try {
        var stmt1 = snowflake.createStatement({sqlText: taskSQL});
        stmt1.execute();
    } catch (err) {
        return "Error while creating task: " + err;
    }
$$;

call calculate_user_journey_path()

alter task calculate_user_journey_path_task RESUME;
```

GOLD TABLES

Creating aggregated columns

MONTHLY ENGAGEMENT METRICS TABLE

```
create or replace procedure monthly_engagement_metrics()
returns string
language javascript
execute as caller
as
$$
   var new_table =
"DIGITAL_3GOLD.DIGITAL_GSCHEMA.MONTHLY_ENGAGEMENT_METRICS";
   var silver_table = "DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE";
   var task_name = "MONTHLY_ENGAGEMENT_METRIC_TASK";
   var task_sql = `
      create or replace task ${task_name}
      warehouse = compute_wh
      schedule = 'USING CRON 20 17 * * * UTC'
      as
      begin
        CREATE OR  REPLACE table ${new_table} AS
```

```
        SELECT
        TO_CHAR(TO_TIMESTAMP(LAST_PING_TIMESTAMP), 'Month') as Months,
        COUNT(distinct COOKIE_ID) AS UNIQUE_USERS,
        sum(SCROLLDEPTH) AS total_SCROLLDEPTH,
        sum(engaged_time_on_page_seconds) AS total_engaged_time_on_page,
        sum(total_time_on_page_seconds) as total_time_on_page
      FROM ${silver_table}
      GROUP BY Months;
    end;
  `;
  try{
    snowflake.execute({sqlText:task_sql});
    return "task created successfully";
  }catch(err){
    return "error creating the task: "+err;
  }
$$;

call monthly_engagement_metrics()
alter task MONTHLY_ENGAGEMENT_METRIC_TASK resume;
```

DAILY ENGAGEMENT METRICS TABLE

```
create or replace procedure daily_engagement_metrics()
returns string
language javascript
execute as caller
as
$$
  var new_table =
"DIGITAL_3GOLD.DIGITAL_GSCHEMA.DAILY_ENGAGEMENT_METRICS";
  var silver_table = "DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE";
  var task_name = "DAILY_ENGAGEMENT_METRIC_TASK";

  var task_sql = `
    create or replace task ${task_name}
    warehouse = compute_wh
    schedule = 'USING CRON 15 17 * * * UTC'
    as
    begin
      CREATE OR  REPLACE table ${new_table} AS
       SELECT
       TO_CHAR(TO_TIMESTAMP(LAST_PING_TIMESTAMP), 'DD Month') as DAY,
```

```
        COUNT(distinct COOKIE_ID) AS UNIQUE_USERS,
        sum(SCROLLDEPTH) AS total_SCROLLDEPTH,
        sum(engaged_time_on_page_seconds) AS total_engaged_time_on_page,
        sum(total_time_on_page_seconds) as total_time_on_page
      FROM ${silver_table}
      GROUP BY DAY;
    end;
  `;

  try{
    snowflake.execute({sqlText:task_sql});
    return "task created successfully";
  }catch(err){
    return "error creating the task: "+err;
  }
$$;

call daily_engagement_metrics()

alter task DAILY_ENGAGEMENT_METRIC_TASK resume;
```

ENGAGEMENT_GROUPS TABLE

```
CREATE OR REPLACE TABLE
DIGITAL_3GOLD.DIGITAL_GSCHEMA.ENGAGEMENT_GROUPS
(
   COOKIE_ID VARCHAR,
   ENGAGEMENT_SCORE DECIMAL,
   ENGAGEMENT_GROUP VARCHAR

);


CREATE OR REPLACE PROCEDURE create_engagement_groups()
RETURNS STRING
LANGUAGE JAVASCRIPT
EXECUTE AS CALLER
AS
$$
   var new_table = "DIGITAL_3GOLD.DIGITAL_GSCHEMA.ENGAGEMENT_GROUPS";
   var original_table="DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE";
   var task_name = "create_engagement_groups_task";
```

```
   var create_SQL = `
      INSERT INTO ${new_table} (COOKIE_ID, ENGAGEMENT_SCORE,
ENGAGEMENT_GROUP)
      SELECT COOKIE_ID, ENGAGEMENT_SCORE,
         CASE
            WHEN ENGAGEMENT_SCORE >= 0 AND ENGAGEMENT_SCORE < 0.5 THEN
'Very Low Engagement'
            WHEN ENGAGEMENT_SCORE >= 0.5 AND ENGAGEMENT_SCORE < 1 THEN
'Low Engagement'
            WHEN ENGAGEMENT_SCORE >= 1 AND ENGAGEMENT_SCORE < 1.5 THEN
'Moderate Engagement'
            WHEN ENGAGEMENT_SCORE >= 1.5 AND ENGAGEMENT_SCORE < 2 THEN
'Above Average Engagement'
            WHEN ENGAGEMENT_SCORE >= 2 AND ENGAGEMENT_SCORE < 2.5 THEN
'High Engagement'
            WHEN ENGAGEMENT_SCORE >= 2.5 AND ENGAGEMENT_SCORE < 3 THEN
'Very High Engagement'
            WHEN ENGAGEMENT_SCORE >= 3 AND ENGAGEMENT_SCORE < 3.5 THEN
'Extremely High Engagement'
            WHEN ENGAGEMENT_SCORE >= 3.5 AND ENGAGEMENT_SCORE < 4 THEN
'Outstanding Engagement'
            WHEN ENGAGEMENT_SCORE >= 4 AND ENGAGEMENT_SCORE < 4.5 THEN
'Exceptional Engagement'
            WHEN ENGAGEMENT_SCORE >= 4.5 AND ENGAGEMENT_SCORE < 5 THEN
'Top Tier Engagement'
            ELSE 'Elite Engagement'
         END
      FROM ${original_table};
   `;

   var task_sql = `
      CREATE OR REPLACE TASK ${task_name}
      WAREHOUSE = COMPUTE_WH
      SCHEDULE='USING CRON 28 17 * * * UTC'
      AS
      ${create_SQL};
   `;

   try {
      var stmt = snowflake.createStatement({sqlText: create_SQL});
      stmt.execute();

      stmt = snowflake.createStatement({sqlText: task_sql});
      stmt.execute();
```

```
        return "Task created successfully";
    } catch (err) {
        return "error while creating task: " + err;
    }
$$;


CALL create_engagement_groups();
ALTER TASK create_engagement_groups_task resume;
```

CONTENT PERFORMANCE_METRICS TABLE

```
CREATE OR REPLACE PROCEDURE create_content_performance_metrics()
RETURNS STRING
LANGUAGE JAVASCRIPT
EXECUTE AS CALLER
AS
$$
    var new_table =
"DIGITAL_3GOLD.DIGITAL_GSCHEMA.CONTENT_PERFORMANCE_METRICS";
    var engagement_groups_table =
"DIGITAL_3GOLD.DIGITAL_GSCHEMA.ENGAGEMENT_GROUPS";
    var original_table =
"DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE";
    var task_name = "create_content_performance_metrics_task";

    var create_table_sql = `
        CREATE OR REPLACE TABLE ${new_table} AS
        SELECT
            a.AUTHOR,
            a.SECTION,
            b.ENGAGEMENT_GROUP,
            COUNT(*) AS TOTAL_ARTICLES,
            AVG(a.ENGAGEMENT_SCORE) AS AVG_ENGAGEMENT_SCORE
        FROM
            ${original_table} a
        JOIN
            ${engagement_groups_table} b
```

```
        ON
            a.COOKIE_ID = b.COOKIE_ID
        GROUP BY
            a.AUTHOR, a.SECTION, b.ENGAGEMENT_GROUP;
    `;

    var task_sql = `
        CREATE OR REPLACE TASK ${task_name}
        WAREHOUSE = COMPUTE_WH
        SCHEDULE='USING CRON 35 17 * * * UTC'
        AS
        ${create_table_sql};
    `;
    try {
        var stmt = snowflake.createStatement({sqlText: create_table_sql});
        stmt.execute();

        stmt = snowflake.createStatement({sqlText: task_sql});
        stmt.execute();

        return "Task created successfully";
    } catch (err) {
        return "error while creating task: " + err;
    }
$$;

CALL create_content_performance_metrics();
ALTER TASK create_content_performance_metrics_task RESUME;
```

PUSHING CLEANED SILVER TABLE TO GOLD

```
create or replace procedure silver_to_gold_digital()
RETURNS STRING
LANGUAGE JAVASCRIPT
EXECUTE AS CALLER
as
$$

  var new_table = "DIGITAL_3GOLD.DIGITAL_GSCHEMA.MONTHLY_DIGITAL_GTABLE";
  var silver_table = "DIGITAL_2SILVER.DIGITAL_SSCHEMA.MONTHLY_DIGITAL_STABLE";
  var task_name = "silver_to_gold_digital_task"
```

```
  var task_sql= `

      create or replace task ${task_name}
      warehouse = compute_wh
      schedule = 'using cron 45 06 * * * utc'
      as
      begin
       create or replace table ${new_table} as select
DISTRIBUTION,COOKIE_ID,DEVICE,SECTION,CONTENT_TYPE,AUTHOR,
       COUNTRY_NAME,SPONSOR,PAGE_TITLE,ENGAGED_TIME_ON_PAGE_SECONDS,

LAST_PING_TIMESTAMP,FREQUENCY,ACCOUNT_ID,SCROLLDEPTH,TOTAL_TIME_ON_P
AGE_SECONDS,

PAGE_LOAD_TIME,SUBSCRIBER_ACCT,PAGE_SESSION_ID,CATEGORIES,DATE,HOUR,U
SER_JOURNEY_PATH,ENGAGEMENT_SCORE
       from ${silver_table};
    end;

   `;
   try{
      snowflake.execute({sqlText:task_sql});
      return "task created successfully";
   }catch(err){
      return "error creating the task: "+err;
   }

$$;

call silver_to_gold_digital()

alter task silver_to_gold_digital_task RESUME;
```

ROLES GRANT

```
CREATE ROLE DATA_ENGINEER
CREATE ROLE DATA_SCIENTIST
CREATE ROLE MARKET_ANALYST

GRANT ROLE DATA_ENGINEER TO USER ADITISAXENA2002;

GRANT ROLE DATA_SCIENTIST TO USER PRATHAMH;
```

```
GRANT ROLE DATA_SCIENTIST TO USER GHANMOHAN;
GRANT ROLE DATA_SCIENTIST TO USER ANJALI303;

GRANT ROLE MARKET_ANALYST TO USER PRATHAMH;
GRANT ROLE MARKET_ANALYST TO USER GHANMOHAN;
GRANT ROLE MARKET_ANALYST TO USER ANJALI303;
GRANT ROLE MARKET_ANALYST TO USER MANJUSHAKOLLI;

--FOR DATA ENGINEER
GRANT ALL PRIVILEGES ON WAREHOUSE COMPUTE_WH TO ROLE data_engineer;
GRANT ALL PRIVILEGES ON DATABASE DIGITAL_1BRONZE TO ROLE data_engineer;
GRANT ALL PRIVILEGES ON SCHEMA DIGITAL_1BRONZE.DIGITAL_BSCHEMA TO ROLE
data_engineer;

--FOR DATA SCIENTIST
GRANT USAGE ON DATABASE DIGITAL_1BRONZE TO ROLE data_scientist;
GRANT USAGE ON ALL SCHEMAS IN DATABASE DIGITAL_1BRONZE TO ROLE
data_scientist;
GRANT CREATE SCHEMA ON DATABASE DIGITAL_1BRONZE TO ROLE data_scientist;
GRANT CREATE TABLE, CREATE VIEW, CREATE STAGE, CREATE FILE FORMAT, CREATE
SEQUENCE, CREATE FUNCTION ON ALL SCHEMAS IN DATABASE DIGITAL_1BRONZE TO
ROLE data_scientist;
GRANT SELECT ON ALL TABLES IN DATABASE DIGITAL_1BRONZE TO ROLE
data_scientist;

--FOR MARKET ANALYST
GRANT USAGE ON DATABASE DIGITAL_1BRONZE TO ROLE market_analyst;
GRANT USAGE ON ALL SCHEMAS IN DATABASE DIGITAL_1BRONZE TO ROLE
market_analyst;
GRANT SELECT ON ALL TABLES IN SCHEMA DIGITAL_1BRONZE.DIGITAL_BSCHEMA TO
ROLE market_analyst;
GRANT SELECT ON ALL VIEWS IN SCHEMA DIGITAL_1BRONZE.DIGITAL_BSCHEMA TO
ROLE market_analyst;

GRANT ROLE data_scientist TO ROLE data_engineer;
GRANT ROLE market_analyst TO ROLE data_scientist;
```

MASKING POLICY

```
USE DATABASE DIGITAL_1BRONZE;

--MASKING POLICY 1
CREATE OR REPLACE MASKING POLICY MASK_CITY_NAME AS (VAL VARCHAR)
RETURNS VARCHAR->
```

```
CASE
WHEN CURRENT_ROLE() IN ('DATA_ENGINEER') THEN VAL ELSE NULL
END;

--MASKING POLICY 2
CREATE OR REPLACE MASKING POLICY MASK_COUNTRY_CODE AS (VAL VARCHAR)
RETURNS VARCHAR->
CASE
WHEN CURRENT_ROLE() IN ('DATA_ENGINEER , DATA_SCIENTIST') THEN VAL ELSE '****'
END;

--APPLYING MASKING POLICIES
ALTER TABLE DIGITAL_1BRONZE.DIGITAL_BSCHEMA.MONTHLY_DIGITAL_BTABLE
MODIFY COLUMN CITY_NAME SET MASKING POLICY MASK_CITY_NAME;
ALTER TABLE DIGITAL_1BRONZE.DIGITAL_BSCHEMA.MONTHLY_DIGITAL_BTABLE
MODIFY COLUMN COUNTRY_CODE SET MASKING POLICY MASK_COUNTRY_CODE;
```

Dashboarding

Engagement score distribution by content type

```
SELECT
   CONTENT_TYPE,
   AVG(ENGAGEMENT_SCORE) AS Average_Engagement_Score,
   COUNT(*) AS Content_Type_Count
FROM
   MONTHLY_DIGITAL_GTABLE
GROUP BY
   CONTENT_TYPE
ORDER BY
   Average_Engagement_Score DESC;
```

Engagement by Content Type and Device

```
SELECT
   CONTENT_TYPE,
   DEVICE,
   AVG(ENGAGEMENT_SCORE) AS Average_Engagement_Score
FROM
   MONTHLY_DIGITAL_GTABLE
GROUP BY
```

```
    CONTENT_TYPE, DEVICE
ORDER BY
    Average_Engagement_Score DESC;
```

Engagement Score vs. Page Load Time

```
SELECT
    FLOOR(PAGE_LOAD_TIME / 10) * 10 AS PAGE_LOAD_TIME_BIN,
    AVG(ENGAGEMENT_SCORE) AS Average_Engagement_Score
FROM
    MONTHLY_DIGITAL_GTABLE
GROUP BY
    PAGE_LOAD_TIME_BIN
ORDER BY
    PAGE_LOAD_TIME_BIN;
```

peak hours for each content category.

```
SELECT
    CATEGORIES,
    HOUR,
    SUM(TOTAL_TIME_ON_PAGE_SECONDS) AS Total_Time_On_Page
FROM
    MONTHLY_DIGITAL_GTABLE
GROUP BY
    CATEGORIES, HOUR
ORDER BY
    Total_Time_On_Page DESC;
```

Marketing Campaigns

```
SELECT
    UTM_CAMPAIGN,
    UTM_MEDIUM,
    UTM_SOURCE,
    COUNT(*) AS Interaction_Count
FROM
    DIGITAL_2SILVER.DIGITAL_SSCHEMA.MARKETING_CAMPAIGNS_SILVER
WHERE
    UTM_CAMPAIGN IS NOT NULL
GROUP BY
    UTM_CAMPAIGN, UTM_MEDIUM, UTM_SOURCE;
```

Deciding threshold to define BOUNCE - engagement time below which 75% of your visits fall

```
WITH Percentile_CTE AS (
   SELECT
      PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY
ENGAGED_TIME_ON_PAGE_SECONDS) AS Percentile_75
   FROM
      MONTHLY_DIGITAL_GTABLE
)
SELECT
   Percentile_75
FROM
   Percentile_CTE;
```

Subsequently calculating BOUNCE RATE

```
WITH Percentile_CTE AS (
   SELECT
      PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY
ENGAGED_TIME_ON_PAGE_SECONDS) AS Percentile_75
   FROM
      MONTHLY_DIGITAL_GTABLE
)
SELECT
   COUNT(*) AS Total_Visits,
   SUM(CASE WHEN ENGAGED_TIME_ON_PAGE_SECONDS < p.Percentile_75 THEN 1
ELSE 0 END) AS Bounces,
   (SUM(CASE WHEN ENGAGED_TIME_ON_PAGE_SECONDS < p.Percentile_75 THEN 1
ELSE 0 END) / COUNT(*)) * 100 AS Bounce_Rate
FROM
   MONTHLY_DIGITAL_GTABLE,
   Percentile_CTE p
GROUP BY
   p.Percentile_75
ORDER BY
   Bounce_Rate DESC;
```

Bounce Rate by Content Type

```
SELECT
   CONTENT_TYPE,
   COUNT(*) AS Total_Visits,
   SUM(CASE WHEN ENGAGED_TIME_ON_PAGE_SECONDS < 35 THEN 1 ELSE 0 END)
AS Bounces,
```

```
    (SUM(CASE WHEN ENGAGED_TIME_ON_PAGE_SECONDS < 35 THEN 1 ELSE 0 END) /
COUNT(*)) * 100 AS Bounce_Rate
FROM
    MONTHLY_DIGITAL_GTABLE
GROUP BY
    CONTENT_TYPE
ORDER BY
    Bounce_Rate DESC;
```

Bounce Rate by Content Type

```
SELECT
    CONTENT_TYPE,
    COUNT(*) AS Total_Visits,
    SUM(CASE WHEN ENGAGED_TIME_ON_PAGE_SECONDS < 35 THEN 1 ELSE 0 END)
AS Bounces,
    (SUM(CASE WHEN ENGAGED_TIME_ON_PAGE_SECONDS < 35 THEN 1 ELSE 0 END) /
COUNT(*)) * 100 AS Bounce_Rate
FROM
    MONTHLY_DIGITAL_GTABLE
GROUP BY
    CONTENT_TYPE
ORDER BY
    Bounce_Rate DESC;
```

Subscriber vs. Non-Subscriber Engagement

```
SELECT
    CASE
        WHEN SUBSCRIBER_ACCT IS NOT NULL THEN 'Subscriber'
        ELSE 'Non-Subscriber'
    END AS User_Type,
    AVG(ENGAGEMENT_SCORE) AS Average_Engagement_Score
FROM
    MONTHLY_DIGITAL_GTABLE
GROUP BY
    User_Type
ORDER BY
    Average_Engagement_Score DESC;
```

Engagement by Path Journey - TOP5

```
SELECT
    PATH_JOURNEY,
```

```
    AVG(ENGAGEMENT_SCORE) AS Average_Engagement_Score
FROM
    MONTHLY_DIGITAL_GTABLE
GROUP BY
    PATH_JOURNEY
ORDER BY
    Average_Engagement_Score DESC
LIMIT 5;
```

TREND ANALYSIS

```
WITH monthly_engagement AS (
    SELECT
        YEAR(DATE) AS year,
        MONTH(DATE) AS month,
        CATEGORIES,
        AVG(ENGAGEMENT_SCORE) AS avg_engagement_score
    FROM
        MONTHLY_DIGITAL_GTABLE
    GROUP BY
        YEAR(DATE), MONTH(DATE), CATEGORIES
),
trend_analysis AS (
    SELECT
        year,
        month,
        CATEGORIES,
        avg_engagement_score,
        LAG(avg_engagement_score) OVER (PARTITION BY CATEGORIES ORDER BY year,
month) AS prev_month_engagement,
        avg_engagement_score - LAG(avg_engagement_score) OVER (PARTITION BY
CATEGORIES ORDER BY year, month) AS change_in_engagement
    FROM
        monthly_engagement
)
SELECT
    year,
    month,
    CATEGORIES,
    avg_engagement_score,
    prev_month_engagement,
    change_in_engagement
FROM
    trend_analysis
```

```
ORDER BY
    year, month, CATEGORIES;
```

Scroll Depth for each category

```
WITH scrolldepth_agg AS (
  SELECT
    CONCAT(FLOOR(scrolldepth / 5000) * 5000, '-', FLOOR(scrolldepth / 5000) * 5000 + 4999)
AS scrolldepth_range,
    CATEGORIES,
    COOKIE_ID
  FROM MONTHLY_DIGITAL_GTABLE
)
SELECT
  scrolldepth_range,
  CATEGORIES AS category,
  COUNT(DISTINCT COOKIE_ID) AS user_count
FROM scrolldepth_agg
GROUP BY 1, 2;
```

top 5 countries having highest engagement score

```
select COUNTRY_NAME,avg(ENGAGEMENT_SCORE) as avg_engagment_score from
monthly_digital_gtable group by country_name
order by avg_engagment_score
desc limit 5;
```

Journey path vs frequency

```
WITH last_two_words_cte AS (
  SELECT SPLIT_PART(path_journey, ' ', -2) || ' ' || SPLIT_PART(path_journey, ' ', -1) AS
last_two_words
  FROM monthly_digital_gtable
)
SELECT last_two_words, COUNT(*) AS frequency
FROM last_two_words_cte
GROUP BY last_two_words
HAVING COUNT(*) > 1
ORDER BY frequency DESC;
```

number of new users per month

```
SELECT
```

```
    country_name,
    TO_CHAR(TO_TIMESTAMP(LAST_PING_TIMESTAMP), 'Month') as MY_month,
    COUNT(CASE WHEN FREQUENCY = 0 THEN 1 END) AS new_user_count
FROM monthly_digital_gtable
GROUP BY MY_month,country_name
ORDER BY new_user_count desc;
```

Month-over-Month Growth in Unique Users

```
SELECT months,
       unique_users,
       LAG(unique_users) OVER (ORDER BY months) AS prev_unique_users,
       (unique_users - LAG(unique_users) OVER (ORDER BY months)) /
NULLIF(LAG(unique_users) OVER (ORDER BY months), 0) * 100 AS mom_growth_rate
FROM DIGITAL_3GOLD.DIGITAL_GSCHEMA.MONTHLY_ENGAGEMENT_METRICS
ORDER BY months;
```

Cumulative Total Time on Page

```
SELECT months,
       SUM(total_time_on_page) OVER (ORDER BY months ROWS BETWEEN UNBOUNDED
PRECEDING AND CURRENT ROW) AS cumulative_time_on_page
FROM DIGITAL_3GOLD.DIGITAL_GSCHEMA.MONTHLY_ENGAGEMENT_METRICS
ORDER BY months;
```

devices which have the highest engagement time

```
select device,avg(engaged_time_on_page_seconds) from monthly_digital_gtable group by
device;
```

Total Articles and Average Engagement Score by Engagement Group

```
SELECT engagement_group, SUM(total_articles) AS total_articles,
AVG(avg_engagement_score) AS avg_engagement
FROM DIGITAL_3GOLD.DIGITAL_GSCHEMA.CONTENT_PERFORMANCE_METRICS
GROUP BY engagement_group
ORDER BY total_articles DESC;
```

Section Performance: Total Articles and Engagement Scores

```
SELECT section, SUM(total_articles) AS total_articles, AVG(avg_engagement_score) AS
avg_engagement
FROM DIGITAL_3GOLD.DIGITAL_GSCHEMA.CONTENT_PERFORMANCE_METRICS
```

```
GROUP BY section
ORDER BY total_articles DESC;
```

Engagement Efficiency (Engaged Time vs. Total Time on Page)

```
SELECT
    day,
    (SUM(total_engaged_time_on_page) / NULLIF(SUM(total_time_on_page), 0)) * 100 AS
engagement_efficiency
FROM
    DIGITAL_3GOLD.DIGITAL_GSCHEMA.DAILY_ENGAGEMENT_METRICS
GROUP BY
    day
ORDER BY
  engagement_efficiency DESC
 LIMIT 5;
```

Section Vs Author

```
SELECT
    AUTHOR,
    SECTION,
    ENGAGEMENT_GROUP,
    AVG(AVG_ENGAGEMENT_SCORE) AS Average_Engagement_Score,
    SUM(TOTAL_ARTICLES) AS Total_Articles
FROM
    CONTENT_PERFORMANCE_METRICS
GROUP BY
    AUTHOR, SECTION, ENGAGEMENT_GROUP
ORDER BY
    AUTHOR, SECTION, ENGAGEMENT_GROUP;
```