-- Setting variables for regular expression based analyses

DECLARE

TIMESTAMP\_REGEX STRING DEFAULT r'^\d{4}-\d{1,2}-\d{1,2}[T ]\d{1,2}:\d{1,2}:\d{1,2}(\.\d{1,6})? \*(([+-]\d{1,2}(:\d{1,2})?)|Z|UTC)?$';

DECLARE

DATE\_REGEX STRING DEFAULT r'^\d{4}-(?:[1-9]|0[1-9]|1[012])-(?:[1-9]|0[1-9]|[12][0-9]|3[01])$';

DECLARE

TIME\_REGEX STRING DEFAULT r'^\d{1,2}:\d{1,2}:\d{1,2}(\.\d{1,6})?$';

-- Setting variables for time of day/ day of week analyses

DECLARE

MORNING\_START,

MORNING\_END,

AFTERNOON\_END,

EVENING\_END INT64;

-- Set the times for the times of the day

SET

MORNING\_START = 6;

SET

MORNING\_END = 12;

SET

AFTERNOON\_END = 18;

SET

EVENING\_END = 21;

-- Check to see which column names are shared across tables

SELECT

column\_name,

COUNT(table\_name)

FROM

`data\_analytics\_cert.fitbit.INFORMATION\_SCHEMA.COLUMNS`

GROUP BY

1;

-- We found that Id was a common column, let's make sure that it is in every table we have

SELECT

table\_name,

SUM(CASE

WHEN column\_name = "Id" THEN 1

ELSE

0

END

) AS has\_id\_column

FROM

`data\_analytics\_cert.fitbit.INFORMATION\_SCHEMA.COLUMNS`

GROUP BY

1

ORDER BY

1 ASC;

-- This query checks to make sure that each table has a column of a date or time related type

-- If your column types were detected properly prior to upload this table should be empty

SELECT

table\_name,

SUM(CASE

WHEN data\_type IN ("TIMESTAMP", "DATETIME", "TIME", "DATE") THEN 1

ELSE

0

END

) AS has\_time\_info

FROM

`data\_analytics\_cert.fitbit.INFORMATION\_SCHEMA.COLUMNS`

WHERE

data\_type IN ("TIMESTAMP",

"DATETIME",

"DATE")

GROUP BY

1

HAVING

has\_time\_info = 0;

-- If we found that we have columns of the type DATETIME, TIMESTAMP, or DATE we can use this query to check for their names

SELECT

CONCAT(table\_catalog,".",table\_schema,".",table\_name) AS table\_path,

table\_name,

column\_name

FROM

`data\_analytics\_cert.fitbit.INFORMATION\_SCHEMA.COLUMNS`

WHERE

data\_type IN ("TIMESTAMP",

"DATETIME",

"DATE");

-- We now know that every table has an "Id" column but we don't know how to join the dates

-- If we find that not every table has a DATETIME, TIMESTAMP, or DATE column we use their names to check for what might be date-related

-- Here we check to see if the column name has any of the keywords below:

-- date, minute, daily, hourly, day, seconds

SELECT

table\_name,

column\_name

FROM

`data\_analytics\_cert.fitbit.INFORMATION\_SCHEMA.COLUMNS`

WHERE

REGEXP\_CONTAINS(LOWER(column\_name), "date|minute|daily|hourly|day|seconds");

-- ADVANCED

-- In the dailyActivity\_merged table we saw that there is a column called ActivityDate, let's check to see what it looks like

-- One way to check if something follows a particular pattern is to use a regular expression.

-- In this case we use the regular expression for a timestamp format to check if the column follows that pattern.

-- The is\_timestamp column demonstrates that this column is a valid timestamp column

SELECT

ActivityDate,

REGEXP\_CONTAINS(STRING(ActivityDate), TIMESTAMP\_REGEX) AS is\_timestamp

FROM

`data\_analytics\_cert.fitbit.dailyActivity\_merged`

LIMIT

5;

-- To quickly check if all columns follow the timestamp pattern we can take the minimum value of the boolean expression across the entire table

SELECT

CASE

WHEN MIN(REGEXP\_CONTAINS(STRING(ActivityDate), TIMESTAMP\_REGEX)) = TRUE THEN "Valid"

ELSE

"Not Valid"

END

AS valid\_test

FROM

`data\_analytics\_cert.fitbit.dailyActivity\_merged`;

-- Say we want to do an analysis based upon daily data, this could help us to find tables that might be at the day level

SELECT

DISTINCT table\_name

FROM

`data\_analytics\_cert.fitbit.INFORMATION\_SCHEMA.COLUMNS`

WHERE

REGEXP\_CONTAINS(LOWER(table\_name),"day|daily");

-- Now that we have a list of tables we should look at the columns that are shared among the tables

SELECT

column\_name,

data\_type,

COUNT(table\_name) AS table\_count

FROM

`data\_analytics\_cert.fitbit.INFORMATION\_SCHEMA.COLUMNS`

WHERE

REGEXP\_CONTAINS(LOWER(table\_name),"day|daily")

GROUP BY

1,

2;

-- Now that we have a list of tables we should look at the columns that are shared among the tables

-- We should also make certain that the data types align between tables

SELECT

column\_name,

table\_name,

data\_type

FROM

`data\_analytics\_cert.fitbit.INFORMATION\_SCHEMA.COLUMNS`

WHERE

REGEXP\_CONTAINS(LOWER(table\_name),"day|daily")

AND column\_name IN (

SELECT

column\_name

FROM

`data\_analytics\_cert.fitbit.INFORMATION\_SCHEMA.COLUMNS`

WHERE

REGEXP\_CONTAINS(LOWER(table\_name),"day|daily")

GROUP BY

1

HAVING

COUNT(table\_name) >=2)

ORDER BY

1;

SELECT

A.Id,

A.Calories,

\* EXCEPT(Id,

Calories,

ActivityDay,

SleepDay,

SedentaryMinutes,

LightlyActiveMinutes,

FairlyActiveMinutes,

VeryActiveMinutes,

SedentaryActiveDistance,

LightActiveDistance,

ModeratelyActiveDistance,

VeryActiveDistance),

I.SedentaryMinutes,

I.LightlyActiveMinutes,

I.FairlyActiveMinutes,

I.VeryActiveMinutes,

I.SedentaryActiveDistance,

I.LightActiveDistance,

I.ModeratelyActiveDistance,

I.VeryActiveDistance

FROM

`data\_analytics\_cert.fitbit.dailyActivity\_merged` A

LEFT JOIN

`data\_analytics\_cert.fitbit.dailyCalories\_merged` C

ON

A.Id = C.Id

AND A.ActivityDate=C.ActivityDay

AND A.Calories = C.Calories

LEFT JOIN

`data\_analytics\_cert.fitbit.dailyIntensities\_merged` I

ON

A.Id = I.Id

AND A.ActivityDate=I.ActivityDay

AND A.FairlyActiveMinutes = I.FairlyActiveMinutes

AND A.LightActiveDistance = I.LightActiveDistance

AND A.LightlyActiveMinutes = I.LightlyActiveMinutes

AND A.ModeratelyActiveDistance = I.ModeratelyActiveDistance

AND A.SedentaryActiveDistance = I.SedentaryActiveDistance

AND A.SedentaryMinutes = I.SedentaryMinutes

AND A.VeryActiveDistance = I.VeryActiveDistance

AND A.VeryActiveMinutes = I.VeryActiveMinutes

LEFT JOIN

`data\_analytics\_cert.fitbit.dailySteps\_merged` S

ON

A.Id = S.Id

AND A.ActivityDate=S.ActivityDay

LEFT JOIN

`data\_analytics\_cert.fitbit.sleepDay\_merged` Sl

ON

A.Id = Sl.Id

AND A.ActivityDate=Sl.SleepDay;

-- Say we are considering sleep related products as a possibility, let's take a moment to see if/ how people nap during the day

-- To do this we are assuming that a nap is any time someone sleeps but goes to sleep and wakes up on the same day

SELECT

Id,

sleep\_start AS sleep\_date,

COUNT(logId) AS number\_naps,

SUM(EXTRACT(HOUR

FROM

time\_sleeping)) AS total\_time\_sleeping

FROM (

SELECT

Id,

logId,

MIN(DATE(date)) AS sleep\_start,

MAX(DATE(date)) AS sleep\_end,

TIME( TIMESTAMP\_DIFF(MAX(date),MIN(date),HOUR),

MOD(TIMESTAMP\_DIFF(MAX(date),MIN(date),MINUTE),60),

MOD(MOD(TIMESTAMP\_DIFF(MAX(date),MIN(date),SECOND),3600),60) ) AS time\_sleeping

FROM

`data\_analytics\_cert.fitbit.minuteSleep\_merged`

WHERE

value=1

GROUP BY

1,

2)

WHERE

sleep\_start=sleep\_end

GROUP BY

1,

2

ORDER BY

3 DESC;

-- Suppose we would like to do an analysis based upon the time of day and day of the week

-- We will do this at a person level such that we smooth over anomalous days for an individual

WITH

user\_dow\_summary AS (

SELECT

Id,

FORMAT\_TIMESTAMP("%w", ActivityHour) AS dow\_number,

FORMAT\_TIMESTAMP("%A", ActivityHour) AS day\_of\_week,

CASE

WHEN FORMAT\_TIMESTAMP("%A", ActivityHour) IN ("Sunday", "Saturday") THEN "Weekend"

WHEN FORMAT\_TIMESTAMP("%A", ActivityHour) NOT IN ("Sunday",

"Saturday") THEN "Weekday"

ELSE

"ERROR"

END

AS part\_of\_week,

CASE

WHEN TIME(ActivityHour) BETWEEN TIME(MORNING\_START, 0, 0) AND TIME(MORNING\_END, 0, 0) THEN "Morning"

WHEN TIME(ActivityHour) BETWEEN TIME(MORNING\_END,

0,

0)

AND TIME(AFTERNOON\_END,

0,

0) THEN "Afternoon"

WHEN TIME(ActivityHour) BETWEEN TIME(AFTERNOON\_END, 0, 0) AND TIME(EVENING\_END, 0, 0) THEN "Evening"

WHEN TIME(ActivityHour) >= TIME(EVENING\_END,

0,

0)

OR TIME(TIMESTAMP\_TRUNC(ActivityHour, MINUTE)) <= TIME(MORNING\_START,

0,

0) THEN "Night"

ELSE

"ERROR"

END

AS time\_of\_day,

SUM(TotalIntensity) AS total\_intensity,

SUM(AverageIntensity) AS total\_average\_intensity,

AVG(AverageIntensity) AS average\_intensity,

MAX(AverageIntensity) AS max\_intensity,

MIN(AverageIntensity) AS min\_intensity

FROM

`data\_analytics\_cert.fitbit.hourlyIntensities\_merged`

GROUP BY

1,

2,

3,

4,

5),

intensity\_deciles AS (

SELECT

DISTINCT dow\_number,

part\_of\_week,

day\_of\_week,

time\_of\_day,

ROUND(PERCENTILE\_CONT(total\_intensity,

0.1) OVER (PARTITION BY dow\_number, part\_of\_week, day\_of\_week, time\_of\_day),4) AS total\_intensity\_first\_decile,

ROUND(PERCENTILE\_CONT(total\_intensity,

0.2) OVER (PARTITION BY dow\_number, part\_of\_week, day\_of\_week, time\_of\_day),4) AS total\_intensity\_second\_decile,

ROUND(PERCENTILE\_CONT(total\_intensity,

0.3) OVER (PARTITION BY dow\_number, part\_of\_week, day\_of\_week, time\_of\_day),4) AS total\_intensity\_third\_decile,

ROUND(PERCENTILE\_CONT(total\_intensity,

0.4) OVER (PARTITION BY dow\_number, part\_of\_week, day\_of\_week, time\_of\_day),4) AS total\_intensity\_fourth\_decile,

ROUND(PERCENTILE\_CONT(total\_intensity,

0.6) OVER (PARTITION BY dow\_number, part\_of\_week, day\_of\_week, time\_of\_day),4) AS total\_intensity\_sixth\_decile,

ROUND(PERCENTILE\_CONT(total\_intensity,

0.7) OVER (PARTITION BY dow\_number, part\_of\_week, day\_of\_week, time\_of\_day),4) AS total\_intensity\_seventh\_decile,

ROUND(PERCENTILE\_CONT(total\_intensity,

0.8) OVER (PARTITION BY dow\_number, part\_of\_week, day\_of\_week, time\_of\_day),4) AS total\_intensity\_eigth\_decile,

ROUND(PERCENTILE\_CONT(total\_intensity,

0.9) OVER (PARTITION BY dow\_number, part\_of\_week, day\_of\_week, time\_of\_day),4) AS total\_intensity\_ninth\_decile

FROM

user\_dow\_summary ),

basic\_summary AS (

SELECT

part\_of\_week,

day\_of\_week,

time\_of\_day,

SUM(total\_intensity) AS total\_total\_intensity,

AVG(total\_intensity) AS average\_total\_intensity,

SUM(total\_average\_intensity) AS total\_total\_average\_intensity,

AVG(total\_average\_intensity) AS average\_total\_average\_intensity,

SUM(average\_intensity) AS total\_average\_intensity,

AVG(average\_intensity) AS average\_average\_intensity,

AVG(max\_intensity) AS average\_max\_intensity,

AVG(min\_intensity) AS average\_min\_intensity

FROM

user\_dow\_summary

GROUP BY

1,

dow\_number,

2,

3)

SELECT

\*

FROM

basic\_summary

LEFT JOIN

intensity\_deciles

USING

(part\_of\_week,

day\_of\_week,

time\_of\_day)

ORDER BY

1,

dow\_number,

2,

CASE

WHEN time\_of\_day = "Morning" THEN 0

WHEN time\_of\_day = "Afternoon" THEN 1

WHEN time\_of\_day = "Evening" THEN 2

WHEN time\_of\_day = "Night" THEN 3

END

;