```
 -- 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");
```

```sql
 -- 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`;
```

```sql
 -- 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
 ;
```