Strava Fitness Data Analysis Case Study

About the Strava Fitness App

Strava is a GPS-based workout tracker and training log app designed for fitness enthusiasts. The app records key details such as route, pace, duration, elevation, and other metrics during various physical activities. It supports 50 activity types including running, cycling, walking, swimming, and even gym workouts like strength training and yoga. Strava does not have direct heartrate and sleep monitoring features. However, to dive deeper into these, you can sync Strava with fitness trackers like apple watch, Garmin, etc. Strava has officially teamed up with the Oura ring which records heart rates during workouts.

Business Requirement

This case study aims to analyse data from the app to gain insights into user behaviour, app/tracker usage patterns. The goal is to identify trends that can inform high level marketing and business strategies in order to enhance user engagement and retention. By doing this, the stakeholders can position Strava more effectively within the competitive digital fitness landscape.

Description of Data Sources Provided

A total of 18 data sources were provided, each capturing different aspects of user activity, health metrics, and tracking data. The datasets are:

- dailyActivity_merged.csv
- dailyCalories_merged.csv
- dailyIntensities_merged.csv
- dailySteps_merged.csv
- heartrate_seconds_merged.csv
- hourlyCalories_merged.csv
- hourlyIntensities_merged.csv
- hourlySteps_merged.csv
- minuteCaloriesNarrow_merged.csv
- minuteCaloriesWide_merged.csv
- minuteIntensitiesNarrow_merged.csv
- minuteIntensitiesWide_merged.csv
- minuteMETsNarrow_merged.csv
- minuteSleep_merged.csv
- minuteStepsNarrow_merged.csv
- minuteStepsWide_merged.csv
- sleepDay_merged.csv
- weightLogInfo_merged.csv

Among these, dailyActivity_merged.csv is the most important one which consolidates all daily activity details into a single dataset including metrics such as dates, total steps, total distance, active distances and calories. The other daily datasets are repetitions, and hence those are excluded from the analysis. All three hourly datasets are considered for the tasks. For minute datasets, both narrow and wide formats are available, containing the same data in different formats. The narrow format is chosen as it is ideal for analysis and visualization tasks. There are two datasets, sleepDay_merged and minuteSleep_merged for sleep data at daily and minute level. The heartrate_seconds_merged contains continuous heart rate measurements which will be useful too. The weightLogInfo_merged holds weight logs, but the dataset doesn't have sufficient data to conduct meaningful analysis.

In summary, eleven datasets were selected from the original 18, to focus on the analysis.

SQL Analysis and Insights

The SQL analysis was conducted using **MySQL Workbench**. A database **strava_analysis** was created for the case study. The datasets were imported into the workbench using the **Table Data Import Wizard**. This helped with seamless loading of CSV files into respective tables within the database.

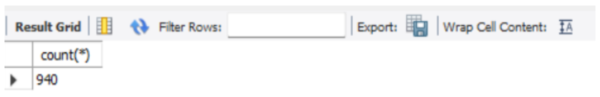
Data Preprocessing Using SQL

After importing the data was thoroughly verified to ensure accuracy and completeness. This included confirming row counts, verifying datatypes, checking for negative and null values, and resolving any discrepancies.

For preprocessing, we have considered the dailyactived_merged CSV, the hourly data CSVs, and the SleepDay_merged CSVs. The minutes seconds data were too large to work on, and hence they were added to the python tasks. During preprocessing, each table was taken and the necessary steps for cleaning and analysis were performed. Below are some steps done for the preprocessing of dailyactivity_merged:

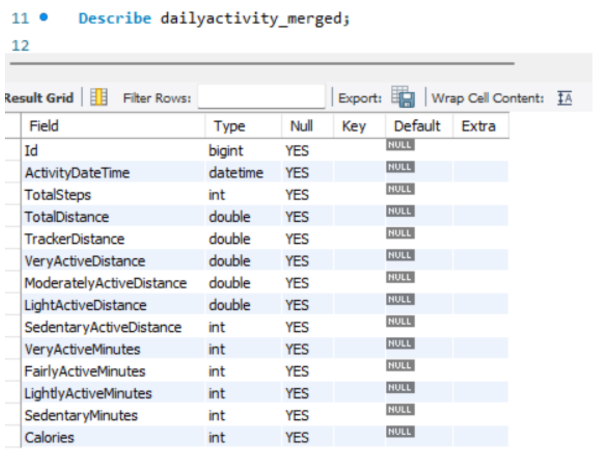
- **Row Count verification**

```sql
Select count(*) from dailyactivity_merged;
```



```
count(*)
940
```

- **Verification of datatypes using describe**



```sql
Describe dailyactivity_merged;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Id	bigint	YES		NULL	
ActivityDateTime	datetime	YES		NULL	
TotalSteps	int	YES		NULL	
TotalDistance	double	YES		NULL	
TrackerDistance	double	YES		NULL	
VeryActiveDistance	double	YES		NULL	
ModeratelyActiveDistance	double	YES		NULL	
LightActiveDistance	double	YES		NULL	
SedentaryActiveDistance	int	YES		NULL	
VeryActiveMinutes	int	YES		NULL	
FairlyActiveMinutes	int	YES		NULL	
LightlyActiveMinutes	int	YES		NULL	
SedentaryMinutes	int	YES		NULL	
Calories	int	YES		NULL	

- **Negative values verification**

```sql
-- Checking for negative values
Select min(Id), min(ActivityDate), min(TotalSteps),min(TotalDistance), min(TrackerDistance), min(VeryActiveDistance),
 min(ModeratelyActiveDistance), min(LightActiveDistance), min(SedentaryActiveDistance), min(VeryActiveMinutes),
 min(FairlyActiveMinutes), min(LightlyActiveMinutes), min(SedentaryMinutes), min(Calories) from dailyactivity_merged;
-- No negative values present
```

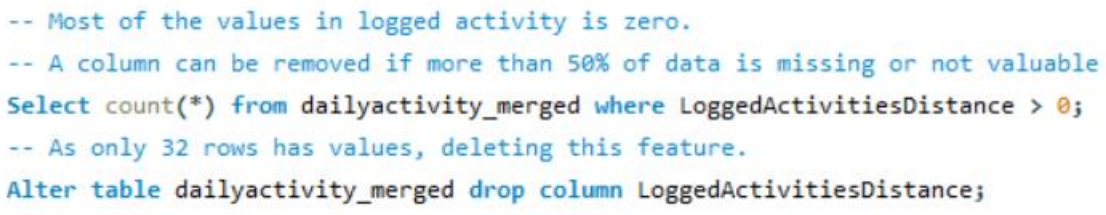
- **Null Values verification**

```sql
-- Checking for null values in the columns
Select
  sum(case when TotalSteps is null then 1 else 0 end) as Null_TotalSteps,
  sum(case when TotalDistance is null then 1 else 0 end) as Null_TotalDistance,
  sum(case when TrackerDistance is null then 1 else 0 end) as Null_TrackerDistance,
  sum(case when LoggedActivitiesDistance is null then 1 else 0 end) as Null_LoggedActivitiesDistance,
  sum(case when VeryActiveDistance is null then 1 else 0 end) as Null_VeryActiveDistance,
  sum(case when ModeratelyActiveDistance is null then 1 else 0 end) as Null_ModeratelyActiveDistance,
  sum(case when SedentaryActiveDistance is null then 1 else 0 end) as Null_SedentaryActiveDistance,
  sum(case when VeryActiveMinutes is null then 1 else 0 end) as Null_VeryActiveMinutes,
  sum(case when FairlyActiveMinutes is null then 1 else 0 end) as Null_FairlyActiveMinutes,
  sum(case when LightlyActiveMinutes is null then 1 else 0 end) as Null_LightlyActiveMinutes,
  sum(case when SedentaryMinutes is null then 1 else 0 end) as Null_SedentaryMinutes,
  sum(case WHEN Calories is null then 1 else 0 end) as Null_Calories
from dailyactivity_merged;
```

- **The date column 'ActivityDate' was in text format. The was converted to datetime format.**

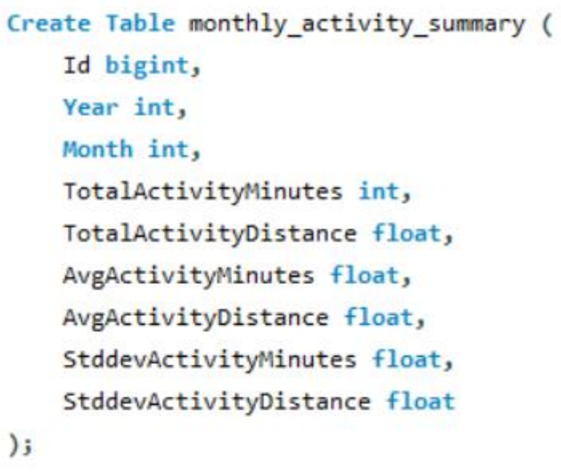
```sql
-- The date colum has text values. Let's convert it to date format
-- In the excel, we can see different formats of date. Let us update it all to the standard date format first

Update dailyactivity_merged
set ActivityDate = case
    when ActivityDate like '%/%/%' then STR_TO_DATE(ActivityDate, '%c/%e/%Y')
    when ActivityDate like '%-%-%' then STR_TO_DATE(ActivityDate, '%m-%d-%Y')
    else ActivityDate
end
where ActivityDate like '%/%/%' or ActivityDate like '%-%-%';
-- Verify if all are in the same format
Select distinct ActivityDate from dailyactivity_merged;
-- Alter the datatype to date
Alter table dailyactivity_merged modify column ActivityDate date;
-- Verify if the datatype has changed
Describe dailyactivity_merged;
```

- **Some features like LoggedActivityDistance had mostly zeros. If a column has more than 50% of null or invalid values, they can be removed. So, removal of unnecessary columns are done as below.**

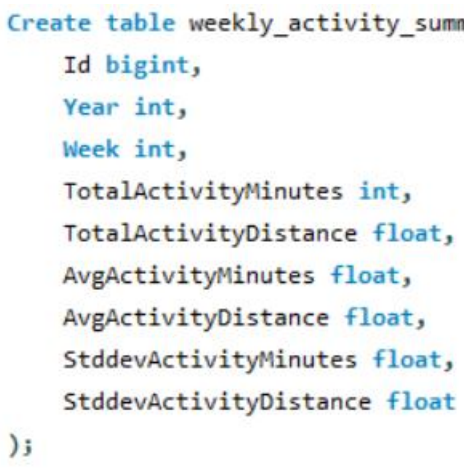


```sql
-- Most of the values in logged activity is zero.
-- A column can be removed if more than 50% of data is missing or not valuable
Select count(*) from dailyactivity_merged where LoggedActivitiesDistance > 0;
-- As only 32 rows has values, deleting this feature.
Alter table dailyactivity_merged drop column LoggedActivitiesDistance;
```

- **Created new tables for future analysis and visualization purposes. Two table showing weekly and monthly data were made using the daily activity data.**

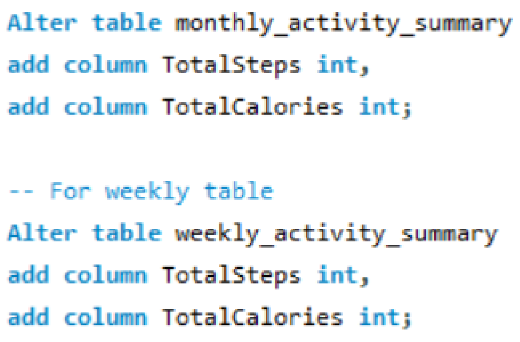


```sql
Create Table monthly_activity_summary (
    Id bigint,
    Year int,
    Month int,
    TotalActivityMinutes int,
    TotalActivityDistance float,
    AvgActivityMinutes float,
    AvgActivityDistance float,
    StddevActivityMinutes float,
    StddevActivityDistance float
);
```



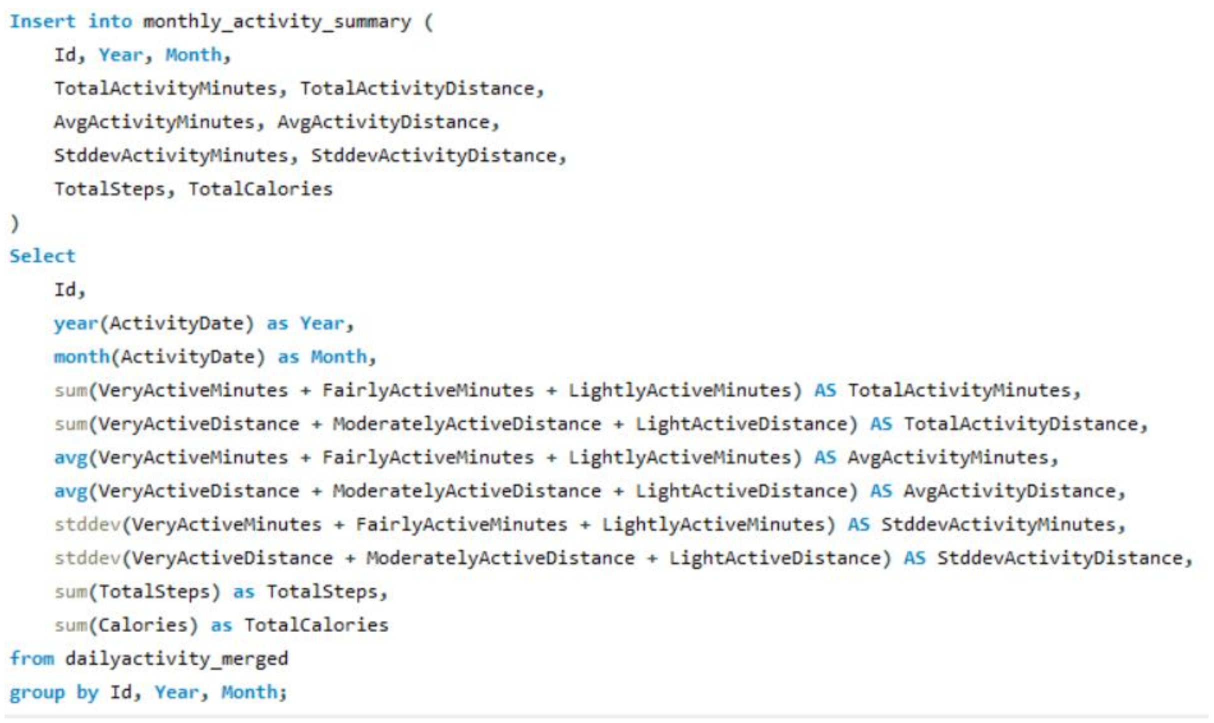
```sql
Create table weekly_activity_summary (
    Id bigint,
    Year int,
    Week int,
    TotalActivityMinutes int,
    TotalActivityDistance float,
    AvgActivityMinutes float,
    AvgActivityDistance float,
    StddevActivityMinutes float,
    StddevActivityDistance float
);
```

Later decided to add two more columns to the tables. Hence altered them.



```sql
Alter table monthly_activity_summary
add column TotalSteps int,
add column TotalCalories int;

-- For weekly table
Alter table weekly_activity_summary
add column TotalSteps int,
add column TotalCalories int;
```

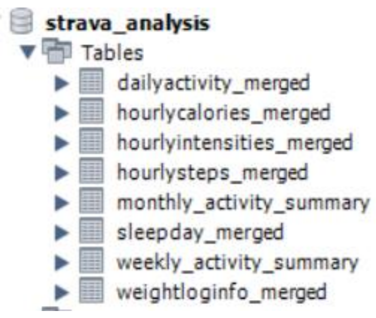
The tables were then populated with necessary values extracted from the daily activity table.



```sql
Insert into monthly_activity_summary (
    Id, Year, Month,
    TotalActivityMinutes, TotalActivityDistance,
    AvgActivityMinutes, AvgActivityDistance,
    StddevActivityMinutes, StddevActivityDistance,
    TotalSteps, TotalCalories
)
Select
    Id,
    year(ActivityDate) as Year,
    month(ActivityDate) as Month,
    sum(VeryActiveMinutes + FairlyActiveMinutes + LightlyActiveMinutes) AS TotalActivityMinutes,
    sum(VeryActiveDistance + ModeratelyActiveDistance + LightActiveDistance) AS TotalActivityDistance,
    avg(VeryActiveMinutes + FairlyActiveMinutes + LightlyActiveMinutes) AS AvgActivityMinutes,
    avg(VeryActiveDistance + ModeratelyActiveDistance + LightActiveDistance) AS AvgActivityDistance,
    stddev(VeryActiveMinutes + FairlyActiveMinutes + LightlyActiveMinutes) AS StddevActivityMinutes,
    stddev(VeryActiveDistance + ModeratelyActiveDistance + LightActiveDistance) AS StddevActivityDistance,
    sum(TotalSteps) as TotalSteps,
    sum(Calories) as TotalCalories
from dailyactivity_merged
group by Id, Year, Month;
```

```
Insert into weekly_activity_summary (
    Id, Year, Week,
    TotalActivityMinutes, TotalActivityDistance,
    AvgActivityMinutes, AvgActivityDistance,
    StddevActivityMinutes, StddevActivityDistance,
    TotalSteps, TotalCalories
)
SELECT
    Id,
    year(ActivityDate) as Year,
    week(ActivityDate) as Week,
    sum(VeryActiveMinutes + FairlyActiveMinutes + LightlyActiveMinutes) AS TotalActivityMinutes,
    sum(VeryActiveDistance + ModeratelyActiveDistance + LightActiveDistance) AS TotalActivityDistance,
    avg(VeryActiveMinutes + FairlyActiveMinutes + LightlyActiveMinutes) AS AvgActivityMinutes,
    avg(VeryActiveDistance + ModeratelyActiveDistance + LightActiveDistance) AS AvgActivityDistance,
    stddev(VeryActiveMinutes + FairlyActiveMinutes + LightlyActiveMinutes) AS StddevActivityMinutes,
    stddev(VeryActiveDistance + ModeratelyActiveDistance + LightActiveDistance) AS StddevActivityDistance,
    sum(TotalSteps) as TotalSteps,
    sum(Calories) as TotalCalories
from dailyactivity_merged
group by Id, Year, Week;
```

The same procedure was repeated for all the chosen tables, and the data was made ready for the analysis. After the preprocessing, the following tables were there in the database.



strava_analysis
- Tables
  - dailyactivity_merged
  - hourlycalories_merged
  - hourlyintensities_merged
  - hourlysteps_merged
  - monthly_activity_summary
  - sleepday_merged
  - weekly_activity_summary
  - weightloginfo_merged

Statistical Data Analysis Using SQL

After preprocessing, a statistical analysis was performed on the data to derive valuable insights.

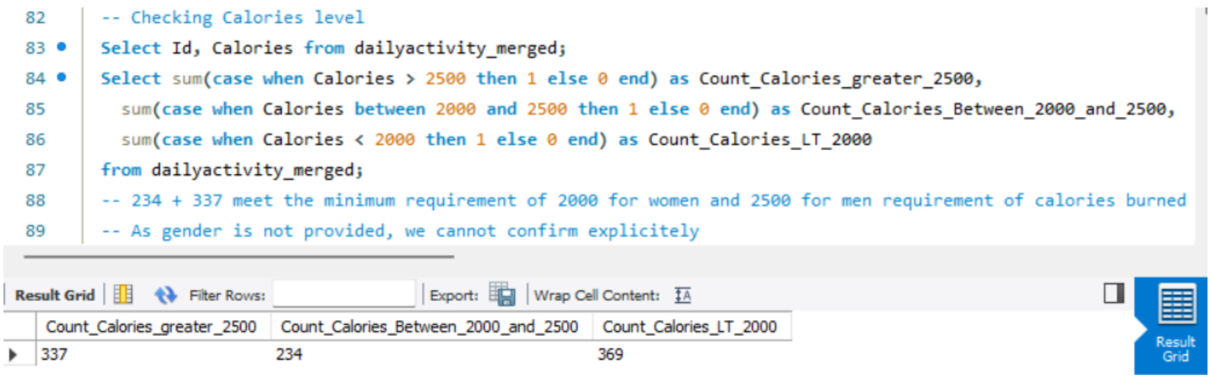
- **Distinct user IDs were counted to learn how many users' data are there in the dataset.**

```
Select count(distinct(Id)) from dailyactivity_merged; -- we have data of 33 users
```

- **The range of ActivityDate was taken to understand the duration of the data we have. In addition to this, the count for each user was taken to check if the number of days of data is consistent across all users.**

```sql
-- Find the date range
Select min(ActivityDate), max(ActivityDate) from dailyactivity_merged ;
-- Results show min as 12-04-2016 and max as 12-05-2016 which indicates that we have 1 month's data
Select id, min(ActivityDate), max(ActivityDate) from dailyactivity_merged group by id;
Select count(ActivityDate) from dailyactivity_merged group by id;
-- Howerever, the data is not consistent for all Ids. Some have data for fewer dates
-- most have 30 or 31 days. Some have 18, 19 and even 4 rows of data
```

- **The calories burned by the users was verified by putting them into three categories to make sure the users are meeting the daily requirement.**



```sql
-- Checking Calories level
Select Id, Calories from dailyactivity_merged;
Select sum(case when Calories > 2500 then 1 else 0 end) as Count_Calories_greater_2500,
  sum(case when Calories between 2000 and 2500 then 1 else 0 end) as Count_Calories_Between_2000_and_2500,
  sum(case when Calories < 2000 then 1 else 0 end) as Count_Calories_LT_2000
from dailyactivity_merged;
-- 234 + 337 meet the minimum requirement of 2000 for women and 2500 for men requirement of calories burned
-- As gender is not provided, we cannot confirm explicitely
```

Count_Calories_greater_2500	Count_Calories_Between_2000_and_2500	Count_Calories_LT_2000
337	234	369

- **The minimum, maximum and average values for various metrics were analysed to get a better understanding of the data.**

```sql
-- Min, Max and Average steps
Select Id, min(TotalSteps), max(TotalSteps), avg(TotalSteps) from dailyactivity_merged group by Id;

-- Min, Max and Average calories burned
Select Id, min(Calories), max(Calories), avg(Calories) from dailyactivity_merged group by Id;

-- Min, Max and Average TrackerDistance
Select Id, min(TrackerDistance), max(TrackerDistance), avg(TrackerDistance) from dailyactivity_merged group by Id;
```

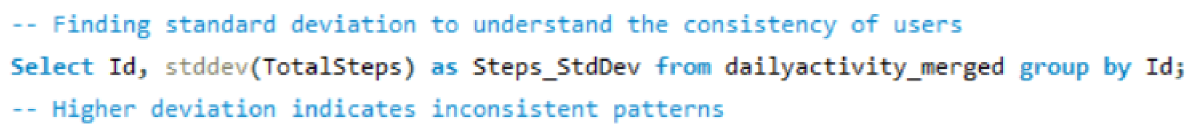
- **The total distance was compared to the tracker distance to verify if they match to make sure the tracker is performing well.**

```sql
-- Min, Max and Average TotalDistance
Select Id, min(TotalDistance), max(TotalDistance), avg(TotalDistance) from dailyactivity_merged group by Id;
set sql_mode = (select replace(@@sql_mode, 'ONLY_FULL_GROUP_BY', ''));
Select Id, TrackerDistance, TotalDistance  from dailyactivity_merged group by Id;
Select * from dailyactivity_merged where TrackerDistance != TotalDistance;
-- The tracker and total distances are the same or very close indicating a good performance of the tracker
```

- **The average active minutes for users were calculated by combing various levels of active minutes.**

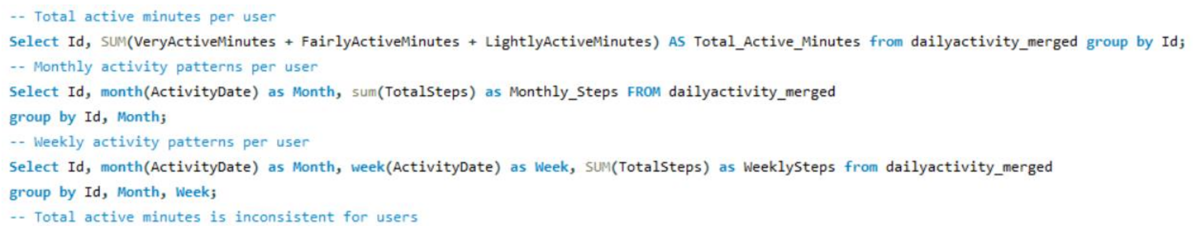
```sql
-- Finding the average active minutes for users
Select Id, avg(VeryActiveMinutes + FairlyActiveMinutes + LightlyActiveMinutes) as Avg_Active_Minutes
from dailyactivity_merged group by Id;
```

- **The standard deviation of steps was calculated in order to check the consistency of the usage.**



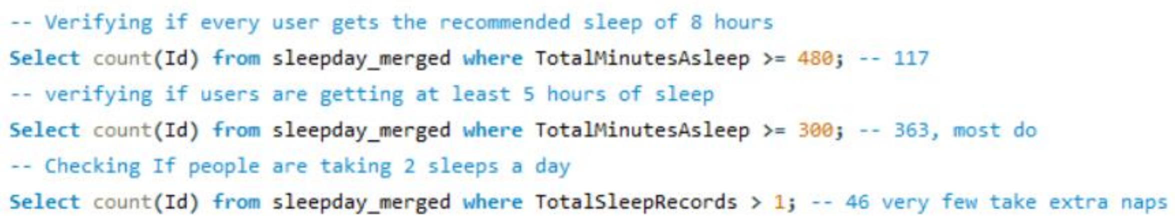
```
-- Finding standard deviation to understand the consistency of users
Select Id, stddev(TotalSteps) as Steps_StdDev from dailyactivity_merged group by Id;
-- Higher deviation indicates inconsistent patterns
```

- **Total active time was verified for different time frames to verify consistency of users.**



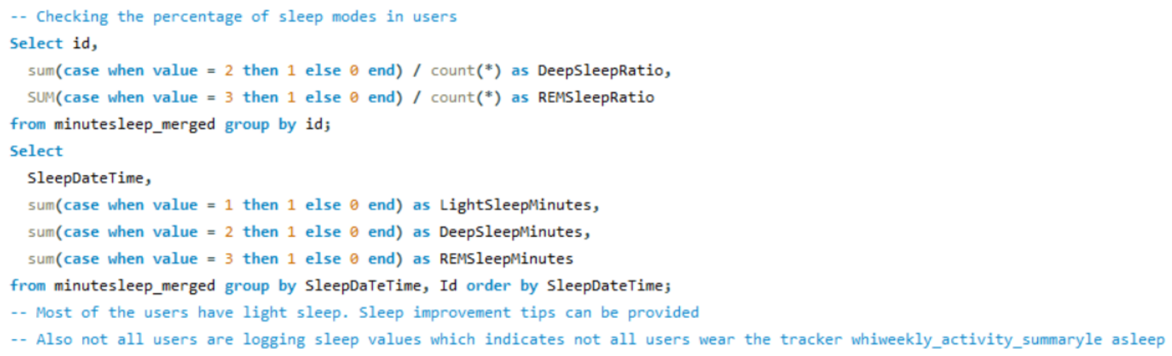
```
-- Total active minutes per user
Select Id, SUM(VeryActiveMinutes + FairlyActiveMinutes + LightlyActiveMinutes) AS Total_Active_Minutes from dailyactivity_merged group by Id;
-- Monthly activity patterns per user
Select Id, month(ActivityDate) as Month, sum(TotalSteps) as Monthly_Steps FROM dailyactivity_merged
group by Id, Month;
-- Weekly activity patterns per user
Select Id, month(ActivityDate) as Month, week(ActivityDate) as Week, SUM(TotalSteps) as WeeklySteps from dailyactivity_merged
group by Id, Month, Week;
-- Total active minutes is inconsistent for users
```

- **The sleep data was verified to see if every user gets the recommended 8 hrs of sleep.**



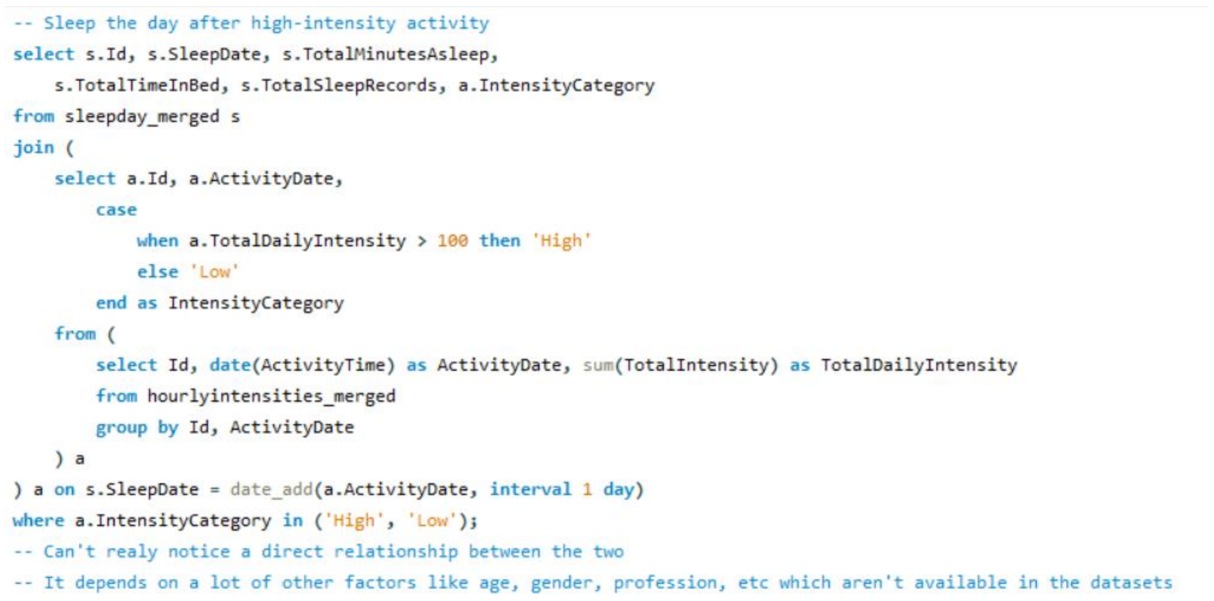
```
-- Verifying if every user gets the recommended sleep of 8 hours
Select count(Id) from sleepday_merged where TotalMinutesAsleep >= 480; -- 117
-- verifying if users are getting at least 5 hours of sleep
Select count(Id) from sleepday_merged where TotalMinutesAsleep >= 300; -- 363, most do
-- Checking If people are taking 2 sleeps a day
Select count(Id) from sleepday_merged where TotalSleepRecords > 1; -- 46 very few take extra naps
```

- **Further analysis was done to check the percentage of sleep modes in various users.**



```
-- Checking the percentage of sleep modes in users
Select id,
  sum(case when value = 2 then 1 else 0 end) / count(*) as DeepSleepRatio,
  SUM(case when value = 3 then 1 else 0 end) / count(*) as REMSleepRatio
from minutesleep_merged group by id;
Select
  SleepDateTime,
  sum(case when value = 1 then 1 else 0 end) as LightSleepMinutes,
  sum(case when value = 2 then 1 else 0 end) as DeepSleepMinutes,
  sum(case when value = 3 then 1 else 0 end) as REMSleepMinutes
from minutesleep_merged group by SleepDaTeTime, Id order by SleepDateTime;
-- Most of the users have light sleep. Sleep improvement tips can be provided
-- Also not all users are logging sleep values which indicates not all users wear the tracker whiweekly_activity_summaryle asleep
```

- **The sleep and work out intensity were compared to see if high intensity workouts helped with better sleep.**



```sql
-- Sleep the day after high-intensity activity
select s.Id, s.SleepDate, s.TotalMinutesAsleep,
    s.TotalTimeInBed, s.TotalSleepRecords, a.IntensityCategory
from sleepday_merged s
join (
    select a.Id, a.ActivityDate,
        case
            when a.TotalDailyIntensity > 100 then 'High'
            else 'Low'
        end as IntensityCategory
    from (
        select Id, date(ActivityTime) as ActivityDate, sum(TotalIntensity) as TotalDailyIntensity
        from hourlyintensities_merged
        group by Id, ActivityDate
    ) a
) a on s.SleepDate = date_add(a.ActivityDate, interval 1 day)
where a.IntensityCategory in ('High', 'Low');
-- Can't realy notice a direct relationship between the two
-- It depends on a lot of other factors like age, gender, profession, etc which aren't available in the datasets
```

Insights from the SQL Analysis

Data Completeness: The dataset includes activity data for 33 users over approximately one month, with most users having 30-31 days of data, though some have incomplete records. There is one user with only 4 records, and a few others with 18 or 19 records. So, there are inconsistencies in the data. Also, the data is from 2016 which is quite outdated.

Data Quality and Integrity: No negative values or null entries were found in key metrics. Date formats were inconsistent initially but were standardized successfully. The data verification ensured accuracy for subsequent analysis.

Activity Patterns and Behaviour:

- A significant portion of the users (approx. 515 out of 930 records) has high sedentary activity (more than 1000 minutes), indicating prolonged inactivity or non-wear periods.
- A small subset (approx. 79 users) shows complete inactivity, possibly because they aren't wearing the tracker of using the app for most part of the day.
- Users burn between 2000 to over 2500 calories daily, aligning with general activity guidelines of 2000 calories for women and 2500 calories for men. However, the details about age and gender are not provided in the dataset for further analysis.

User Engagement and Consistency:

- The average daily steps and active minutes vary across users, with high deviations indicating inconsistent activity patterns among users.
- Weekly and monthly activity summaries reveal fluctuations and provide a basis for identifying active versus inactive periods.

Activity Intensity and Sleep Relationship: High-intensity activity days do not show a strong direct correlation with subsequent day's sleep quality, suggesting that other factors (age, health status, profession, etc.) may influence sleep more significantly. Many users experience around 8 hours of sleep (approx. 480 minutes), with some sleeping less (<5 hours), indicating variability in sleep patterns.

Key Takeaways from SQL Analysis

The suggestion for the team to improve user engagement and overall usage and experience of the app are below:

- Additional demographic data (age, gender, profession) would enrich analysis but is absent from current datasets.
- Strava has many activities which are of different intensities. Including the details of the activity along with the intensity and active minutes in the dataset would provide deeper insights on the effect on users' activity patterns. This will also help in suggesting activities that the user has genuine interest in.
- Send timely reminders or motivational messages to users with prolonged sedentary periods, nudging them toward short activity breaks.
- Implement rewards for consistent activity, weekly goals, or sleep improvements to foster motivation and habitual usage. Monitor the activity to observe if it's working for users (Strava already have this and have a social aspect to motivate people. But the data was not present).
- Encourage users to log weight, sleep, and other health metrics regularly. Implement a system for this, or nudge the user time to time.
- Utilize activity and sleep patterns to tailor personalized notifications encouraging users to meet daily step goals or sleep targets.