```sql
SELECT DISTINCT(Id)
FROM weightLogInfo_merged
--checking how many Id's there are in daily activity table. 8

SELECT DISTINCT(Id)
FROM sleepDay_merged
--checking how many Id's there are in sleep table. 23

SELECT DISTINCT(Id)
FROM dailyActivity_merged
--checking how many Id's there are in weight table. 33

SELECT DISTINCT(da.Id) DailyActivity_Id, wli.Id WeightLog_Id, sd.Id SleepDay_Id
FROM dailyActivity_merged da
RIGHT JOIN weightLogInfo_merged wli
ON da.Id = wli.Id
LEFT JOIN sleepDay_merged sd
ON da.Id = sd.Id
--Unnecessary query that verifies the existence of all Id's in the weight log table,
corresponding to Id's in the daily activity and sleep table
--A left join here would show every Id existing in the activity table and their
correspondants in the other 2 tables if they exist
--an inner join would show if the Id exists in all 3 tables and truncate those that do
not have rows in all tables

--This query reveals that all id’s in daily activity overlap with the id’s in the other
two tables, so there are no more than 33 unique id’s.

ALTER TABLE weightLogInfo_merged
ALTER COLUMN Date date NOT NULL

ALTER TABLE sleepDay_merged
ALTER COLUMN SleepDay date NOT NULL
--adjusting the datetime data type of the dates in sleep and weight as hours are not
necessary, only the date

SELECT MIN(ActivityDate) Activity_Date_Start, MAX(ActivityDate) AS Activity_Date_End,
MIN(SleepDay) Sleep_Day_Start, MAX(SleepDay) Sleep_Day_End, MIN(Date) Weight_Date_Start,
MAX(Date) Weight_Date_End
FROM dailyActivity_merged da, sleepDay_merged sd, weightLogInfo_merged wli
--Find the period of time used for data recording in each table. This is to check for
consistant data between tables. The start and end dates are from 04-12-2016 to 05-12-
2016.

SELECT Id, ActivityDate, COUNT(*) AS ocurrences
FROM dailyActivity_merged
GROUP BY Id, ActivityDate
HAVING COUNT(*) > 1
--Checking for duplicate entries based on the entry date. Date criteria is only one entry
a day, so any more than one would be a dupelicate.
--returns no duplicates
SELECT Id, SleepDay, COUNT(*) AS ocurrences
FROM sleepDay_merged
GROUP BY Id, SleepDay
HAVING COUNT(*) > 1
--returns 3 duplicates

SELECT Id, Date, COUNT(*) AS ocurrences
FROM weightLogInfo_merged
```

```sql
GROUP BY Id, Date
HAVING COUNT(*) > 1
--returns no duplicates

SELECT DISTINCT *
INTO sleepDay
FROM sleepDay_merged
--Purging duplicate rows and appending resulting rows into a new table

SELECT Id, ActivityDate, TotalSteps, TrackerDistance, VeryActiveDistance,
ModeratelyActiveDistance, LightActiveDistance,
VeryActiveMinutes, FairlyActiveMinutes, LightlyActiveMinutes, SedentaryMinutes, Calories
INTO dailyActivity
FROM dailyActivity_merged
WHERE TotalSteps > 100 AND SedentaryMinutes != 1440 AND Calories > 1000
--3 columns truncated in the new table as these columns were redundant/irelevant or did
not have data.
--filters have been placed on TotalSteps, Sedentary minutes and calories to remove any
--inconsistent, incomplete and inaccurate data, i.e. people not wearing their fitbits yet
still having records.
--total steps filter was guestimated based on movement upon getting out of bed/moving
around house (bare minimum)
--anything with 24 hours worth of sendentary activity was removed as this means the
fitbit was not worn or used
--calorie filter is based on average calories burnt in a sedentary state. having looked
at the data
--i have determined that there is too much variability in calories burnt, so i have opted
to use
--a baseline of greater than 1000 calories burnt a day, as an average adult woman will
burn around
--1600-1800 calories in a purely sedentary state. this is not reflected properly in the
data,


DELETE
FROM sleepDay
WHERE TotalTimeInBed = 961
--4 rows with identical time spent in bed. data may have been recorded incorrectly
--as this is the most common anomoly occurence in the table.

SELECT Id, Date, WeightKg, WeightPounds, BMI, IsManualReport
INTO weightLog
FROM weightLogInfo_merged
--truncating fat column as there is not enough entries

ALTER TABLE weightLog
ALTER COLUMN IsManualReport VARCHAR(3) NOT NULL;
--Type casting a column to prepare it for row updating

UPDATE weightLog
SET IsManualReport =
CASE
WHEN IsManualReport = '1' THEN 'Yes'
WHEN IsManualReport = '0' THEN 'No'
ELSE IsManualReport
END
--making the column ismanualreport slightly more human readable.

UPDATE dailyActivity
```

```
SET
TrackerDistance = ROUND(TrackerDistance, 2),
VeryActiveDistance = ROUND(VeryActiveDistance, 2),
ModeratelyActiveDistance = ROUND(ModeratelyActiveDistance, 2),
LightActiveDistance = ROUND(LightActiveDistance, 2)
--Rounding each recorded distance two the nearest 2 decimal place

UPDATE weightLog
SET
WeightKg = ROUND(WeightKg, 2),
WeightPounds = ROUND(WeightPounds, 2),
BMI = ROUND(BMI, 2)
--rounding recorded weights and bmi's to nearest 2 decimal place

EXEC sp_rename 'sleepDay.TotalMinutesAsleep', 'HoursAsleep';
EXEC sp_rename 'sleepDay.TotalTimeInBed', 'HoursInBed';
--renaming columns to properly reflect values

ALTER TABLE dailyActivity ALTER COLUMN VeryActiveMinutes float;
ALTER TABLE dailyActivity ALTER COLUMN FairlyActiveMinutes float;
ALTER TABLE dailyActivity ALTER COLUMN LightlyActiveMinutes float;
ALTER TABLE dailyActivity ALTER COLUMN SedentaryMinutes float;
--type casting in preparation for converting minute values to hours

UPDATE dailyActivity
SET
VeryActiveMinutes = ROUND(VeryActiveMinutes/60, 2),
FairlyActiveMinutes = ROUND(FairlyActiveMinutes/60, 2),
LightlyActiveMinutes = ROUND(LightlyActiveMinutes/60, 2),
SedentaryMinutes = ROUND(SedentaryMinutes/60, 2)
--converting minutes to hours

EXEC sp_rename 'dailyActivity.TrackerDistance', 'TrackerKM';
EXEC sp_rename 'dailyActivity.VeryActiveDistance', 'VeryActiveKM';
EXEC sp_rename 'dailyActivity.ModeratelyActiveDistance', 'ModerateActiveKM';
EXEC sp_rename 'dailyActivity.LightActiveDistance', 'LiteActiveKM';
EXEC sp_rename 'dailyActivity.VeryActiveMinutes', 'VeryActiveHR';
EXEC sp_rename 'dailyActivity.FairlyActiveMinutes', 'ModerateActiveHR';
EXEC sp_rename 'dailyActivity.LightlyActiveMinutes', 'LiteActiveHR';
EXEC sp_rename 'dailyActivity.SedentaryMinutes', 'SedentaryHR';
--renaming columns to properly reflect values

ALTER TABLE dailyActivity
ADD dayOfWeek char(9)

UPDATE dailyActivity
SET dayOfWeek = DATENAME(weekday, ActivityDate)
--adding a new column to reflect the dates as the weekday. Repeated for every table

END OF DATA CLEANING
```

Exporting cleaned tables into CSV's and then transferring to Tableau/Sheets for analysis and visualisation

```
SELECT *
FROM dailyActivity
ORDER BY Id, ActivityDate

SELECT *
```

```
FROM sleepDay
ORDER BY Id, SleepDay

SELECT *
FROM weightLog
ORDER BY Id, Date
```

All exported into CSV formats and ready to be analyzed

```
SELECT dayOfWeek WeekDay, SUM(TotalSteps) TotalSteps, SUM(Calories) TotalCaloriesBurnt,
SUM(LiteActiveKM) TotalLiteActiveKM,
SUM(ModerateActiveKM) TotalModeratelyActiveKM, SUM(VeryActiveKM) TotalVeryActiveKM,
SUM(LiteActiveHR) TotalLiteActiveHours, SUM(ModerateActiveHR) TotalModeratelyActiveHours,
SUM(VeryActiveHR) TotalVeryActiveHours, SUM(SedentaryHR) TotalSedentaryHours,
AVG(TotalSteps) MeanSteps, AVG(Calories) MeanCaloriesBurnt,
ROUND(AVG(LiteActiveKM), 2) MeanLiteKM, ROUND(AVG(ModerateActiveKM), 2) MeanModerateKM,
ROUND(AVG(VeryActiveKM), 2) MeanVeryActiveKM,
ROUND(AVG(LiteActiveHR), 2) MeanLiteHours, ROUND(AVG(ModerateActiveHR), 2)
MeanModerateHours, ROUND(AVG(VeryActiveHR), 2) MeanVeryActiveHours,
ROUND(AVG(SedentaryHR), 2) MeanSedentaryHours
FROM dailyActivity
GROUP BY dayOfWeek
--Query to pull summary data from the dailyActivity table.
--This newly generated table will be used to create the visualisations and point out key
points in the data


SELECT dayOfWeek, SUM(HoursAsleep) TotalAsleepHours, SUM(HoursInBed) TotalBedHours,
ROUND(AVG(HoursAsleep), 2) MeanHoursAsleep, ROUND(AVG(HoursInBed), 2) MeanBedHours
FROM sleepDay
GROUP BY dayOfWeek


SELECT da.Id DailyActivity_Id, ActivityDate, da.dayOfWeek, TotalSteps, LiteActiveKM,
ModerateActiveKM, VeryActiveKM, LiteActiveHR, ModerateActiveHR, VeryActiveHR,
SedentaryHR, Calories, HoursAsleep, HoursInBed
FROM dailyActivity da
INNER JOIN sleepDay sd
ON da.Id = sd.Id AND da.ActivityDate = sd.SleepDay
--Merging the tables for analysis
```

Records from everything have been exported in a CSV file. The summary data will be put into a pivot table and then made into graphs with google sheets. The cleaned tables will be loaded into tableau for visualisation.