In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Google Data Analytics Capstone Project

Case Study 2: How Can a Wellness Technology Company Play It Smart?

In order to help a fictional, high-tech manufacturer of health-focused products, Bellabeat, data from fitbit users was analyzed in order to identify trends in the data that could be used to inform the fictional company Bellabeat of their market, and to give them marketing strategies for a specific product that Bellabeat makes.

The first step in the analysis process was to load the data into python:

In [3]:
```
dailyActivity_merged = pd.read_csv('data/dailyActivity_merged.csv')

ID = dailyActivity_merged["Id"]
ActivityDate = dailyActivity_merged["ActivityDate"]
TotalSteps = dailyActivity_merged["TotalSteps"]
TotalDistance = dailyActivity_merged["TotalDistance"]
TrackerDistance = dailyActivity_merged["TrackerDistance"]
LoggedActivitiesDistance = dailyActivity_merged["LoggedActivitiesDistance"]
VeryActiveDistance = dailyActivity_merged["VeryActiveDistance"]
ModeratelyActiveDistance = dailyActivity_merged["ModeratelyActiveDistance"]
LightActiveDistance = dailyActivity_merged["LightActiveDistance"]
SedentaryActiveDistance = dailyActivity_merged["SedentaryActiveDistance"]
VeryActiveMinutes = dailyActivity_merged["VeryActiveMinutes"]
FairlyActiveMinutes = dailyActivity_merged["FairlyActiveMinutes"]
SedentaryMinutes = dailyActivity_merged["SedentaryMinutes"]
Calories = dailyActivity_merged["Calories"]
```

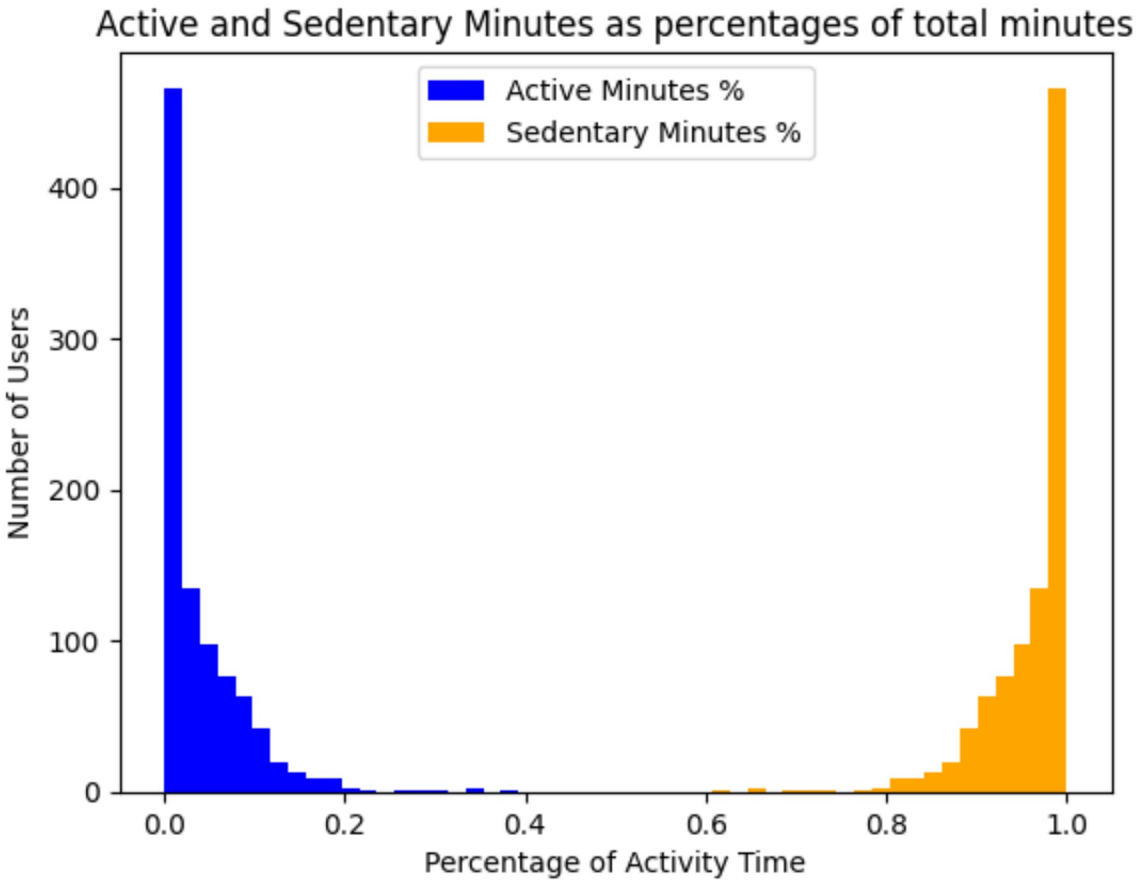
Next new columns of data were calculated using the existing data.

In [26]:
```
TotalMinutes = VeryActiveMinutes + FairlyActiveMinutes + SedentaryMinutes
ActiveMinutes_Perc = (VeryActiveMinutes + FairlyActiveMinutes)/TotalMinutes
SedentaryMinutes_Perc = SedentaryMinutes/TotalMinutes
ActiveDistance_Perc = (VeryActiveDistance + ModeratelyActiveDistance + LightActiveD
SedentaryDistance_Perc = SedentaryActiveDistance/TotalDistance
VeryActiveDistance_Perc = VeryActiveDistance/TotalDistance
ModeratelyActiveDistance_Perc = ModeratelyActiveDistance/TotalDistance
LightActiveDistance_Perc = LightActiveDistance/TotalDistance


print(np.mean(SedentaryMinutes_Perc))
```

```
0.9625196630751879
```

In order to get an idea on what percentage of users are more active or less active, a histogram was plotted for the percentage of active and sedentary minutes.

In [5]:
```
#Visualize the percentage of active vs sedentary activity

plt.hist(ActiveMinutes_Perc,bins=20,color="blue")
plt.hist(SedentaryMinutes_Perc,bins=20,color="orange")
plt.legend(['Active Minutes %', 'Sedentary Minutes %'], loc = "upper center")
plt.title("Active and Sedentary Minutes as percentages of total minutes")
plt.xlabel('Percentage of Activity Time')
plt.ylabel('Number of Users')
plt.show()
```



From this graph, we can see that most people are more sedentary than active.

The next question to answer was what activity contributes the most to this activity. To do this, data was imported for data on steps and intensities.

In [6]:
```
hourlySteps_merged = pd.read_csv('data/hourlySteps_merged.csv')
ID_steps = hourlySteps_merged["Id"]
ActivityHour = hourlySteps_merged["ActivityHour"]
StepTotal = hourlySteps_merged["StepTotal"]
```

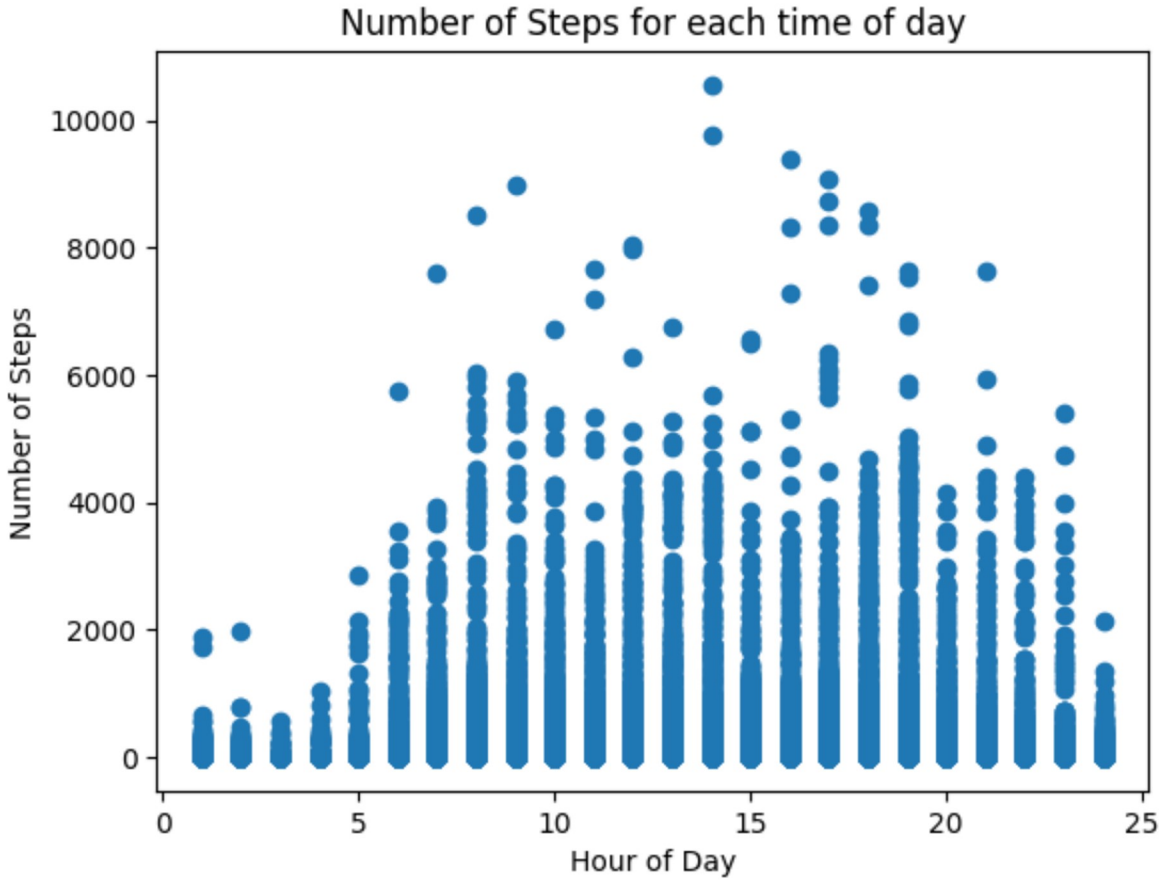
Next the data had to be cleaned in order to ensure the data could be used in the analysis process.

In [7]:
```
#Data Cleaning

#This for-loop searches the variable ActivityHour for the time of day (AM or PM) an
#The array is split into individual components in the for-loop and information is e
#After this, the extracted elements are put back into an array for use later in ana


TIMEList = []
HOURList_step = []
MONTHList = []
YEARList = []
DAYList = []
for i in range(len(ActivityHour)):
   ActHourInd = ActivityHour[i]
   IndLen = len(ActHourInd)
   ActMonth = ActHourInd[0]
   ActTIME = ActHourInd[-2:]
   TIMEList.append(ActTIME)
   MONTHList.append(ActMonth)

   for j in range(IndLen):
      ActHour = ActHourInd[j]
      if ActHour == '/':
         for k in range(1,9):
            if ActHourInd[j+k] == '/':
               ActDay = ActHourInd[j+1:j+k]
               ActYear = ActHourInd[j+k+1:j+k+1+4]
               DAYList.append(ActDay)
               YEARList.append(ActYear)

      if ActHour == ':' and ActHourInd[j+3] == ':' and ActHourInd[j-2] != ' ':
         ActHour1 = ActHourInd[j-2:j]
         HOURList_step.append(ActHour1)
      if ActHour == ':' and ActHourInd[j+3] == ':' and ActHourInd[j-2] == ' ':
         ActHour1 = ActHourInd[j-1]
         HOURList_step.append(ActHour1)



for i in range(len(HOURList_step)):
   HOURList_stepINT = int(HOURList_step[i])
   HOURList_step[i] = HOURList_stepINT
   #HOURListINT_step.append(HOURListINT_i)

HOURListINT_step = np.array(HOURList_step)




for i in range(len(HOURListINT_step)):
   if HOURListINT_step[i] < 12 and TIMEList[i] == 'PM':
      HOURListINT_step[i] = HOURListINT_step[i]+12

   if HOURListINT_step[i] == 12 and TIMEList[i] == 'AM':
      HOURListINT_step[i] = HOURListINT_step[i]+12
```

```
#plt.hist(HOURListINT,bins=24)
#plt.show()



StepTotalList = []
for i in range(len(StepTotal)):
   StepTotalInd = StepTotal[i]
   StepTotalList.append(StepTotalInd)
```

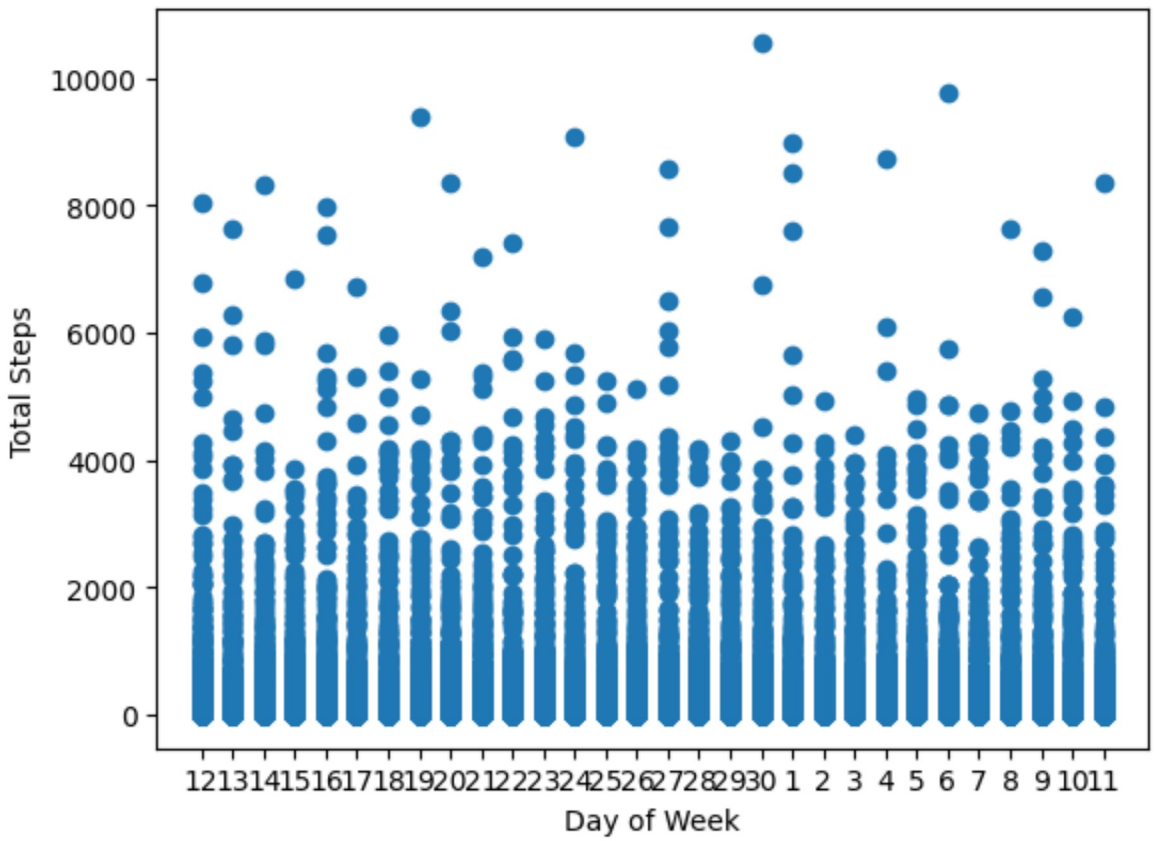
After cleaning the data, the next step was to visualize the number of steps people took throughout the day.

In [9]:
```
plt.scatter(HOURListINT_step,StepTotal)
#plt.xlim(0,24)
plt.xlabel('Hour of Day')
plt.ylabel('Number of Steps')
plt.title('Number of Steps for each time of day')
plt.show()


print('Average Number of Steps = ', round(np.mean(StepTotal),2))
print('Standard Deviation of Number of Steps = ', round(np.std(StepTotal),2))



plt.scatter(DAYList,StepTotal)
plt.xlabel('Day of Week')
plt.ylabel('Total Steps')
```




```
Average Number of Steps =  320.17
Standard Deviation of Number of Steps =  690.37
```

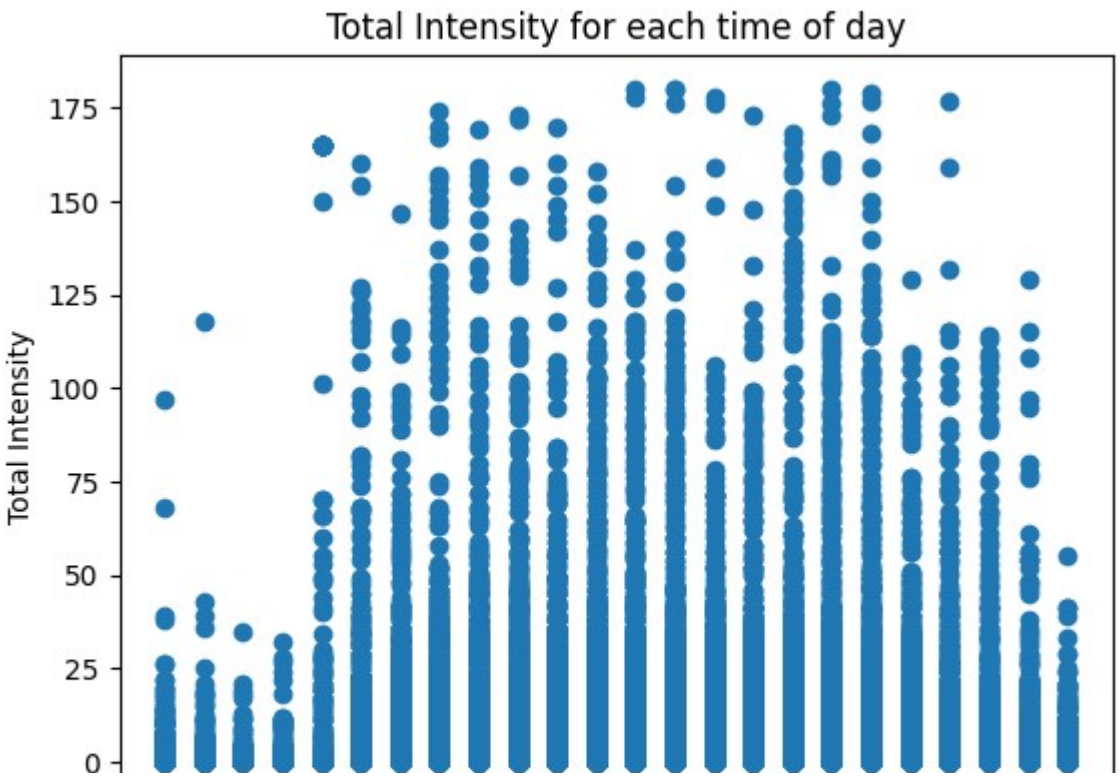
Out[9]: Text(0, 0.5, 'Total Steps')

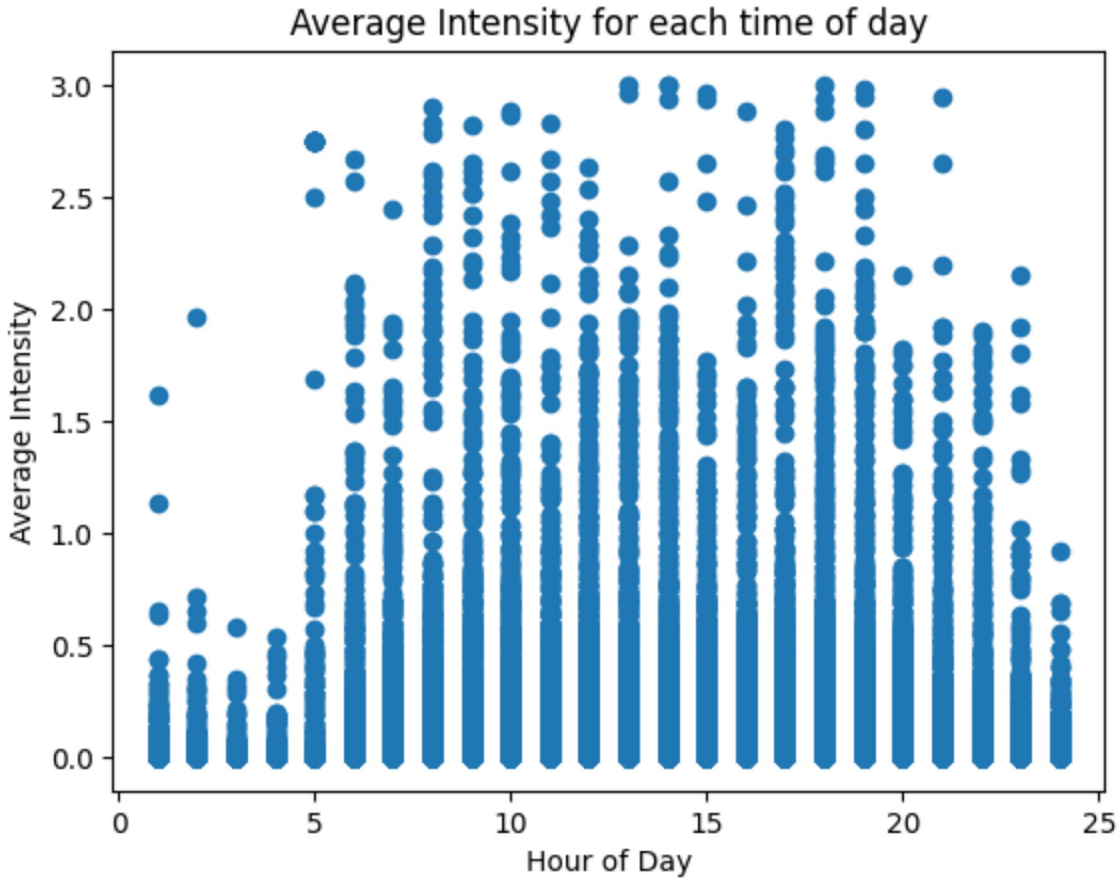


This process was repeated for the intensities data.

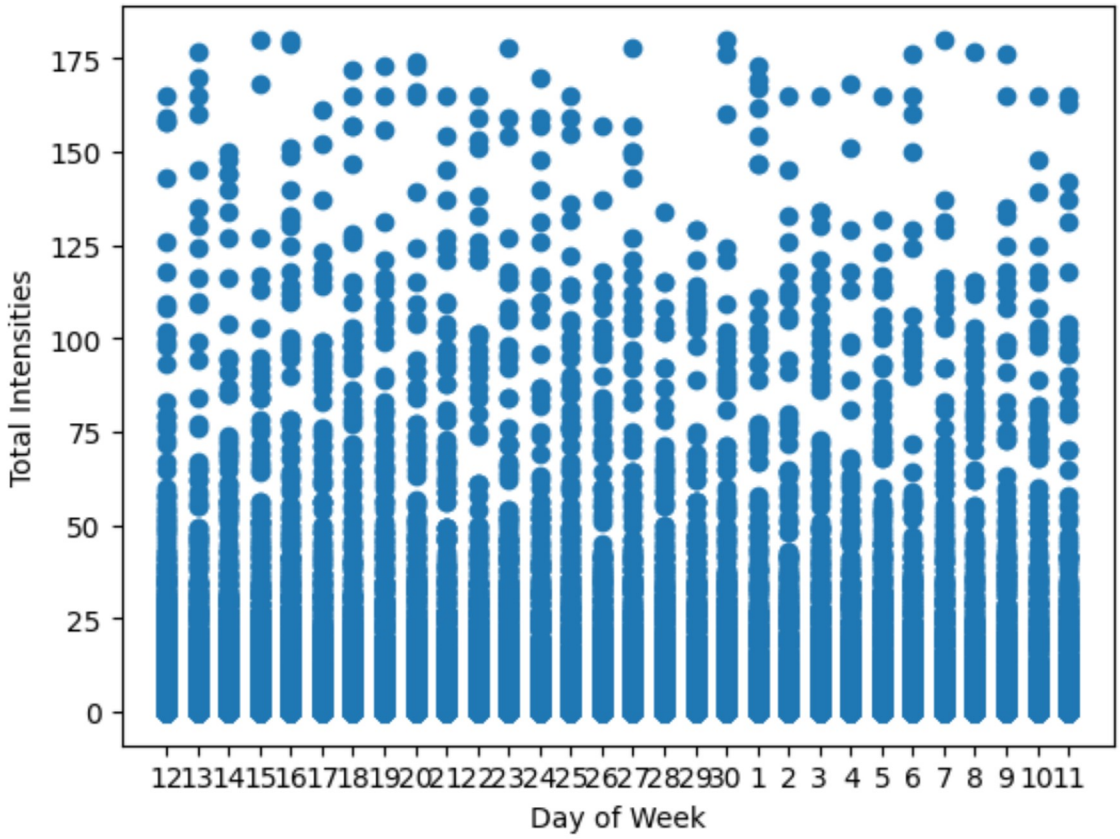
In [14]:
```
hourlyIntensities_merged = pd.read_csv('data/hourlyIntensities_merged.csv')

ID_intense = hourlyIntensities_merged["Id"]
ActivityHour_Intense = hourlyIntensities_merged["ActivityHour"]
TotalIntensity = hourlyIntensities_merged["TotalIntensity"]
AverageIntensity = hourlyIntensities_merged["AverageIntensity"]




TIMEList_Intense = []
HOURList_Intense = []
MONTHList_Intense = []
YEARList_Intense = []
DAYList_Intense = []
for i in range(len(ActivityHour_Intense)):
   ActHourInd_Intense = ActivityHour_Intense[i]
   IndLen_Intense = len(ActHourInd_Intense)
   ActMonth_Intense = ActHourInd_Intense[0]
   ActTIME_Intense = ActHourInd_Intense[-2:]
   TIMEList_Intense.append(ActTIME_Intense)
   MONTHList_Intense.append(ActMonth_Intense)

   for j in range(IndLen_Intense):
      ActHour_Intense = ActHourInd_Intense[j]
      if ActHour_Intense == '/':
         for k in range(1,9):
            if ActHourInd_Intense[j+k] == '/':
               ActDay_Intense = ActHourInd_Intense[j+1:j+k]
               ActYear_Intense = ActHourInd_Intense[j+k+1:j+k+1+4]
               DAYList_Intense.append(ActDay_Intense)
               YEARList_Intense.append(ActYear_Intense)

      if ActHour_Intense == ':' and ActHourInd_Intense[j+3] == ':' and ActHourInd_I
         ActHour1_Intense = ActHourInd_Intense[j-2:j]
         HOURList_Intense.append(ActHour1_Intense)
      if ActHour_Intense == ':' and ActHourInd_Intense[j+3] == ':' and ActHourInd_I
         ActHour1_Intense = ActHourInd_Intense[j-1]
         HOURList_Intense.append(ActHour1_Intense)


#HOURList_Intense = np.asarray(HOURList_Intense)
#HOURListINT_Intnese = []
for i in range(len(HOURList_Intense)):
    HOURList_Intense[i] = int(HOURList_Intense[i])
   #HOURListINT_Intense_i = int(HOURList_Intense[i])
   #HOURList_Intense[i] = HOURListINT_Intense_i
   #HOURListINT_Intnese.append(HOURListINT_Intense_i)


HOURListINT_Intense = np.array(HOURList_Intense)
#print(HOURList_Intense)

#print(HOURList_Intense)
```

```
for i in range(len(HOURList_Intense)):
   if HOURListINT_Intense[i] < 12 and TIMEList_Intense[i] == 'PM':
      HOURListINT_Intense[i] = HOURListINT_Intense[i]+12

   if HOURListINT_Intense[i] == 12 and TIMEList_Intense[i] == 'AM':
      HOURListINT_Intense[i] = HOURListINT_Intense[i]+12



plt.scatter(HOURListINT_Intense,TotalIntensity)
#plt.xlim(0,24)
plt.xlabel('Hour of Day')
plt.ylabel('Total Intensity')
plt.title('Total Intensity for each time of day')
plt.show()



plt.scatter(HOURListINT_Intense,AverageIntensity)
#plt.xlim(0,24)
plt.xlabel('Hour of Day')
plt.ylabel('Average Intensity')
plt.title('Average Intensity for each time of day')
plt.show()



plt.scatter(DAYList,TotalIntensity)
plt.xlabel('Day of Week')
plt.ylabel('Total Intensities')
```





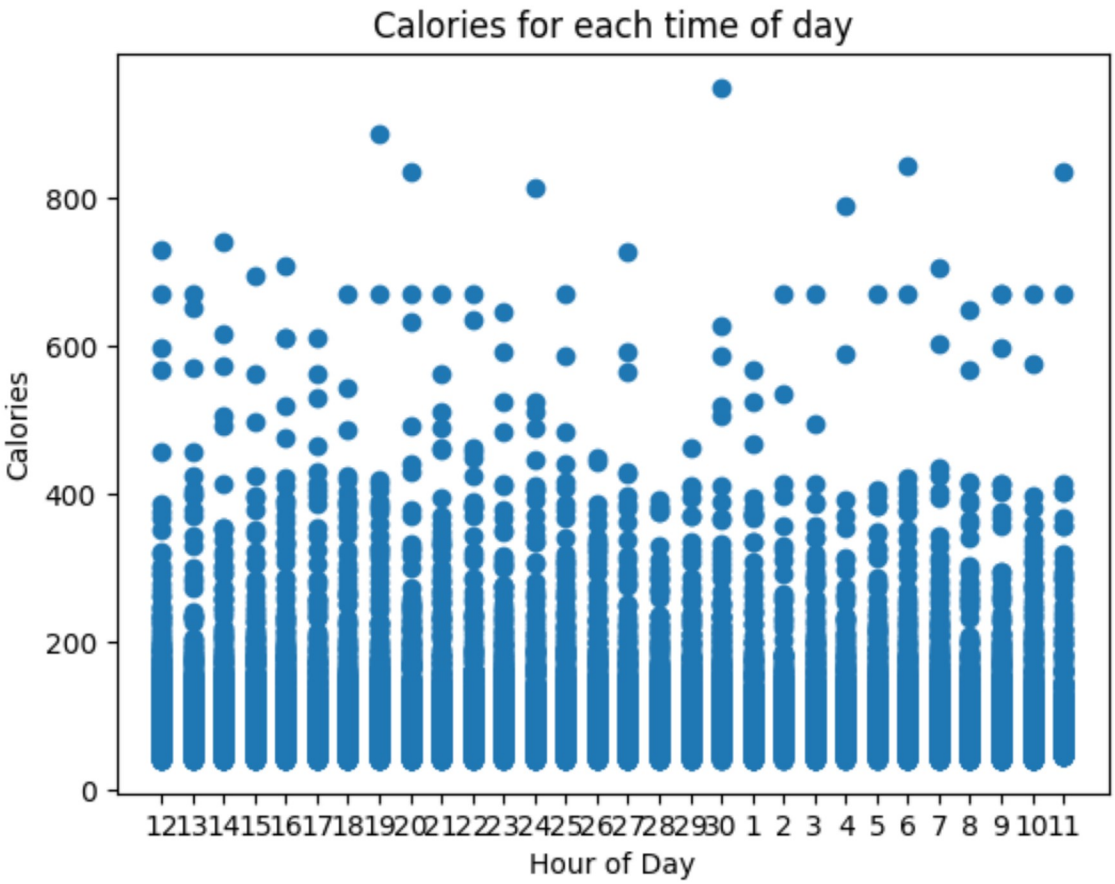
Out[14]: Text(0, 0.5, 'Total Intensities')



Next, the calories data was loaded in.

In [20]:
```
hourlyCalories_merged = pd.read_csv("data/hourlyCalories_merged.csv")

ID_cal = hourlyCalories_merged["Id"]
ActivityHour_Cal = hourlyCalories_merged["ActivityHour"]
Calories = hourlyCalories_merged["Calories"]




TIMEList_cal = []
HOURList_cal = []
MONTHList_cal = []
YEARList_cal = []
DAYList_cal = []
for i in range(len(ActivityHour_Cal)):
   ActHourInd_Cal = ActivityHour_Cal[i]
   IndLen_Cal = len(ActHourInd_Cal)
   ActMonth_Cal = ActHourInd_Cal[0]
   ActTIME_Cal = ActHourInd_Cal[-2:]
   TIMEList_cal.append(ActTIME_Cal)
   MONTHList_cal.append(ActMonth_Cal)

   for j in range(IndLen_Cal):
      ActHour_Cal = ActHourInd_Cal[j]
      if ActHour_Cal == '/':
         for k in range(1,9):
            if ActHourInd_Cal[j+k] == '/':
               ActDay_Cal = ActHourInd_Cal[j+1:j+k]
               ActYear_Cal = ActHourInd_Cal[j+k+1:j+k+1+4]
               DAYList_cal.append(ActDay_Cal)
               YEARList_cal.append(ActYear_Cal)

      if ActHour_Cal == ':' and ActHourInd_Cal[j+3] == ':' and ActHourInd_Cal[j-2]
         ActHour1_Cal = ActHourInd_Cal[j-2:j]
         HOURList_cal.append(ActHour1_Cal)
      if ActHour_Cal == ':' and ActHourInd_Cal[j+3] == ':' and ActHourInd_Cal[j-2]
         ActHour1_Cal = ActHourInd_Cal[j-1]
         HOURList_cal.append(ActHour1_Cal)


HOURList_cal = np.asarray(HOURList_cal)
HOURListINT_cal = []
for i in range(len(HOURList_cal)):
   HOURListINT_cal_i = int(HOURList_cal[i])
   HOURListINT_cal.append(HOURListINT_cal_i)

HOURListINT_cal = np.array(HOURListINT_cal)




for i in range(len(HOURListINT_cal)):
   if HOURListINT_cal[i] < 12 and TIMEList_cal[i] == 'PM':
      HOURListINT_cal[i] = HOURListINT_cal[i]+12
```

```
    if HOURListINT_cal[i] == 12 and TIMEList_cal[i] == 'AM':
        HOURListINT_cal[i] = HOURListINT_cal[i]+12



plt.scatter(HOURListINT_cal,Calories)
#plt.xlim(0,24)
plt.xlabel('Hour of Day')
plt.ylabel('Calories')
plt.title('Calories for each time of day')
plt.show()



print('Average Calories per hour = ', round(np.mean(Calories),2))
```




```
Average Calories per hour =  97.39
```

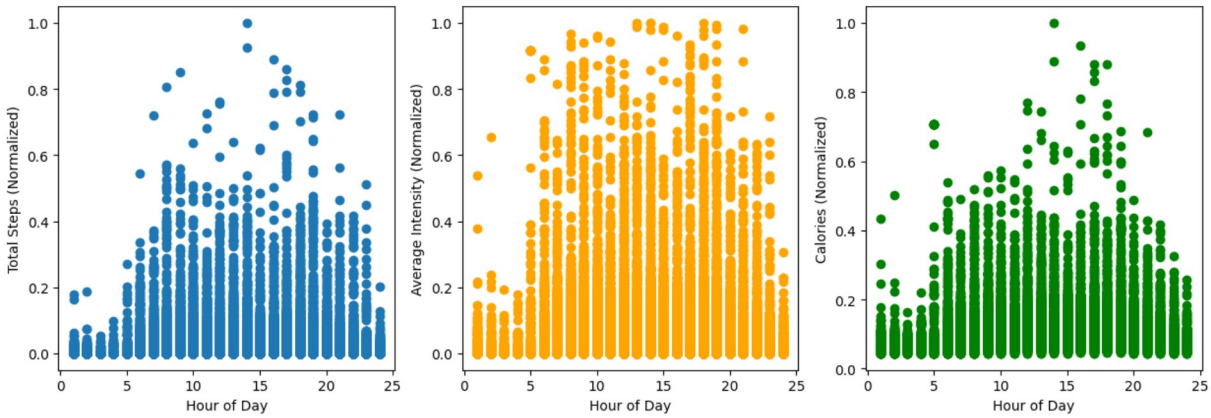
After loading in and cleaning all of this data, the coorelation between calories and steps, and calories and intensities could be determined.

In [17]:
```
Calories_Step_CorrCoef = np.corrcoef(Calories, StepTotal)
Calories_Intensity_CorrCoef = np.corrcoef(Calories, TotalIntensity)

print('Correlation between Calories and Steps ', Calories_Step_CorrCoef[0,1])
print('Correlation between Calories and Intensities', Calories_Intensity_CorrCoef[0
```

```
Correlation between Calories and Steps  0.8149679692954465
Correlation between Calories and Intensities 0.8966160631899995
```

In [19]:
```
fig, (ax1, ax2, ax3) = plt.subplots(1, 3)
fig.set_figwidth(15)


ax1.scatter(HOURListINT_step,StepTotal/max(StepTotal))
ax1.set_xlabel('Hour of Day')
ax1.set_ylabel('Total Steps (Normalized)')

ax2.scatter(HOURListINT_Intense,AverageIntensity/max(AverageIntensity), color = 'or
ax2.set_xlabel('Hour of Day')
ax2.set_ylabel('Average Intensity (Normalized)')

#ax3.scatter(HOURListINT_intense,TotalIntensity/max(TotalIntensity), color = 'orang
#ax3.set_xlabel('Hour of Day')
#ax3.set_ylabel('Total Intensity (Normalized)')

ax3.scatter(HOURListINT_cal,Calories/max(Calories), color='green')
ax3.set_xlabel('Hour of Day')
ax3.set_ylabel('Calories (Normalized)')
```

Out[19]:
```
Text(0, 0.5, 'Calories (Normalized)')
```



In [ ]:
```
StepTotalNorm = StepTtotal/max(StepTotal)
AverageIntensityNorm = AverageIntensity/max(AverageIntensity)
TotalIntensityNorm = TotalIntensity/max(TotalIntensity)
CaloriesNorm = Calories/max(Calories)


fig, (ax1, ax2) = plt.subplots(2,1)

ax1.scatter(HOURListINT_step, StepTotalNorm-CaloriesNorm)

ax2.scatter(HOURListINT_Intense, AverageIntensityNorm-CaloriesNorm)
```

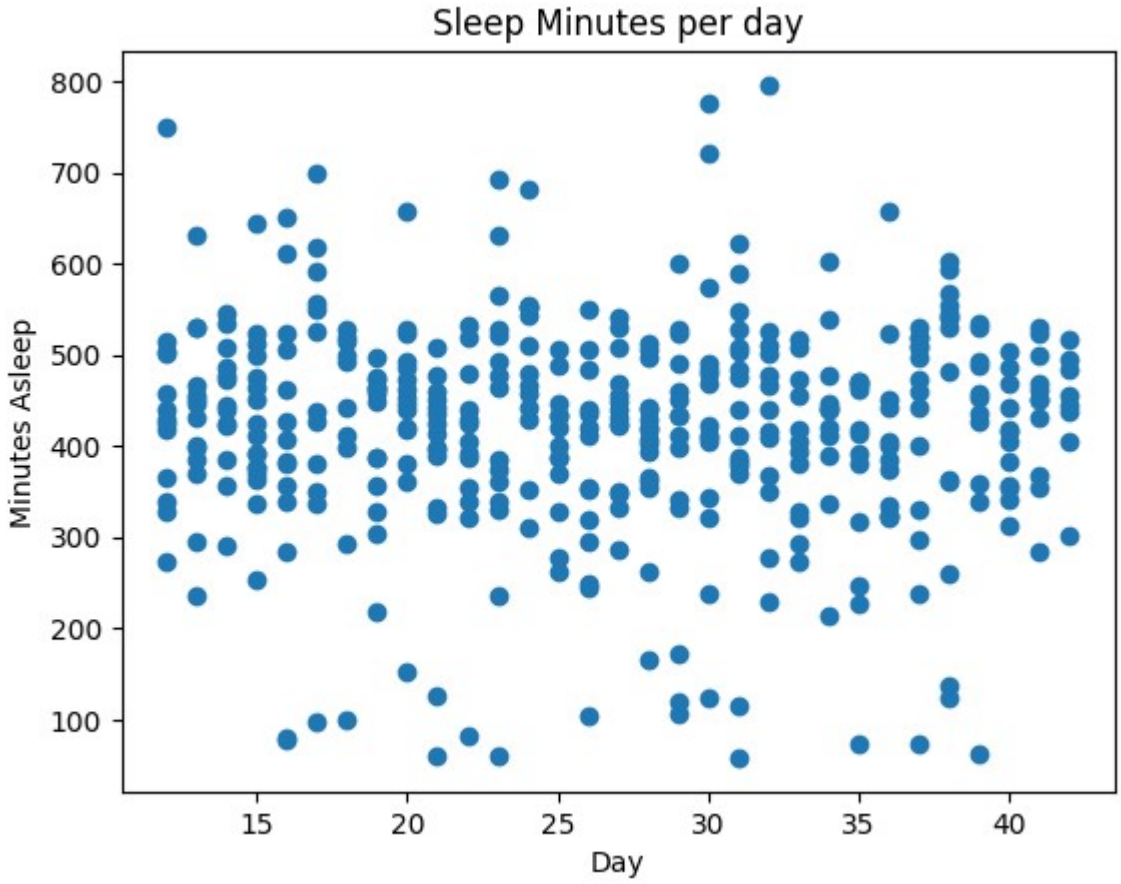
After this, data on sleep was cleaned and visualized.

In [27]:
```
sleepDay_merged = pd.read_csv("data/sleepDay_merged.csv")

ID = sleepDay_merged["Id"]
SleepDay = sleepDay_merged["SleepDay"]
TotalSleepRecords = sleepDay_merged["TotalSleepRecords"]
TotalMinutesAsleep = sleepDay_merged["TotalMinutesAsleep"]
TotalTimeInBed = sleepDay_merged["TotalTimeInBed"]


TIMEList = []
HOURList_sleep = []
MONTHList = []
YEARList = []
DAYList = []
for i in range(len(SleepDay)):
   SleepDayInd = SleepDay[i]
   IndLen = len(SleepDayInd)
   ActMonth = SleepDayInd[0]
   ActTIME = SleepDayInd[-2:]
   TIMEList.append(ActTIME)
   MONTHList.append(ActMonth)

   for j in range(IndLen):
      SleepHour = SleepDayInd[j]
      if SleepHour == '/':
         for k in range(1,9):
            if SleepDayInd[j+k] == '/':
               ActDay = SleepDayInd[j+1:j+k]
               ActYear = SleepDayInd[j+k+1:j+k+1+4]
               DAYList.append(ActDay)
               YEARList.append(ActYear)

      if SleepHour == ':' and SleepDayInd[j+3] == ':' and SleepDayInd[j-2] != ' ':
         ActHour1 = SleepDayInd[j-2:j]
         HOURList_sleep.append(ActHour1)
      if SleepHour == ':' and SleepDayInd[j+3] == ':' and SleepDayInd[j-2] == ' ':
         ActHour1 = SleepDayInd[j-1]
         HOURList_sleep.append(ActHour1)



#HOURList_sleep = np.asarray(HOURList_sleep)
for i in range(len(HOURList_sleep)):
   HOURList_sleep[i] = int(HOURList_sleep[i])

HOURList_sleep = np.array(HOURList_sleep)
HOURListINT_sleep = HOURList_sleep


for i in range(len(HOURListINT_sleep)):
   if HOURListINT_sleep[i] < 12 and TIMEList_sleep[i] == 'PM':
      HOURListINT_sleep[i] = HOURListINT_sleep[i]+12

   if HOURListINT_sleep[i] == 12 and TIMEList[i] == 'AM':
      HOURListINT_sleep[i] = HOURListINT_sleep[i]+12
```

```
MONTHList = np.asarray(MONTHList)
MONTHListINT = []
for i in range(len(MONTHList)):
   MONTHListINT_i = int(MONTHList[i])
   MONTHListINT.append(MONTHListINT_i)

MONTHListINT = np.array(MONTHListINT)
```

```
DAYList = np.asarray(DAYList)
DAYListINT = []
for i in range(len(DAYList)):
   DAYListINT_i = int(DAYList[i])
   DAYListINT.append(DAYListINT_i)

DAYListINT = np.array(DAYListINT)
```

```
DAYListINT_chronological = []
DAYListINT_c = DAYListINT_chronological
for i in range(len(MONTHListINT)):
    if MONTHListINT[i] == 4:
        DAYListINT_ind = DAYListINT[i]
        DAYListINT_c.append(DAYListINT_ind)
    if MONTHListINT[i] == 5:
        DAYListINT_ind = DAYListINT[i]+30
        DAYListINT_c.append(DAYListINT_ind)
    if MONTHListINT[i] == 6:
        DAYListINT_ind = DAYListINT[i]+30+31
        DAYListINT_c.append(DAYListINT_ind)


DAYListINT_c = np.array(DAYListINT_c)

plt.scatter(DAYListINT_c,TotalMinutesAsleep)
plt.xlabel('Day')
plt.ylabel('Minutes Asleep')
plt.title('Sleep Minutes per day')
plt.show()


AvgSleepHour = TotalMinutesAsleep/60


print('Average Minutes Asleep = ', round(np.mean(TotalMinutesAsleep),2))
print('Average Hours Asleep = ', round(np.mean(AvgSleepHour),2))
print('Variance in Sleep Hours = ', round(np.var(AvgSleepHour),2))
print('Standard Deviation of Sleep Hours = ', round(np.std(AvgSleepHour),2))
```



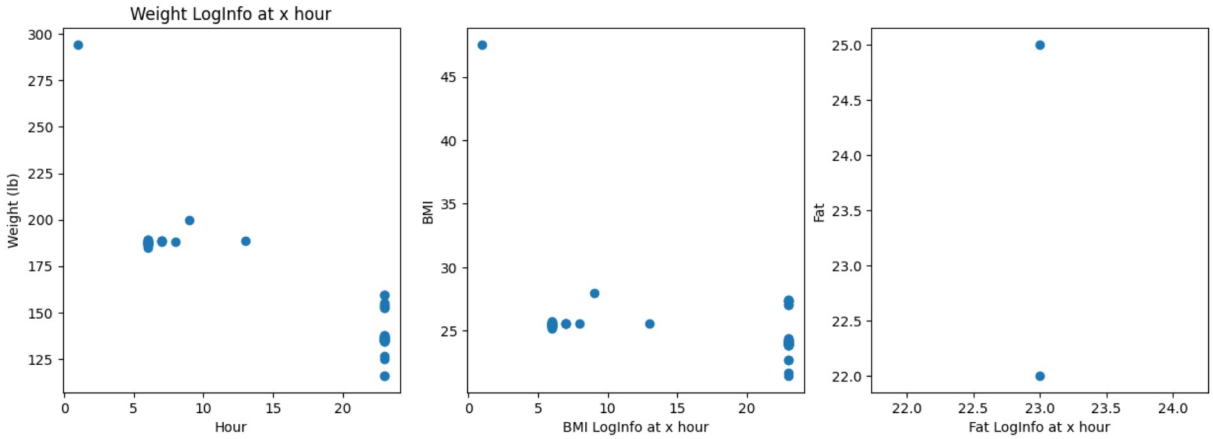

```
Average Minutes Asleep =  419.47
Average Hours Asleep =  6.99
Variance in Sleep Hours =  3.88
Standard Deviation of Sleep Hours =  1.97
```

Next, data on weight was loaded in and cleaned.

In [13]:
```
weightLogInfo_merged = pd.read_csv("data/weightLogInfo_merged.csv")

ID_w = weightLogInfo_merged["Id"]
Date = weightLogInfo_merged["Date"]
WeightKg = weightLogInfo_merged["WeightKg"]
WeightLb = weightLogInfo_merged["WeightPounds"]
Fat = weightLogInfo_merged["Fat"]
BMI = weightLogInfo_merged["BMI"]
IsManualReport = weightLogInfo_merged["IsManualReport"]
LogId = weightLogInfo_merged["LogId"]


TIMEList_w = []
HOURList_w = []
MONTHList_w = []
YEARList_w = []
DAYList_w = []
for i in range(len(Date)):
   DateInd = Date[i]
   IndLen = len(DateInd)
   DateMonth = DateInd[0]
   DateTIME = DateInd[-2:]
   TIMEList_w.append(DateTIME)
   MONTHList_w.append(DateMonth)

   for j in range(IndLen):
      DateHour = DateInd[j]
      if DateHour == '/':
         for k in range(1,9):
            if DateInd[j+k] == '/':
               DateDay = DateInd[j+1:j+k]
               DateYear = DateInd[j+k+1:j+k+1+4]
               DAYList_w.append(DateDay)
               YEARList_w.append(DateYear)

      if DateHour == ':' and DateInd[j+3] == ':' and DateInd[j-2] != ' ':
         DateHour1 = DateInd[j-2:j]
         HOURList_w.append(DateHour1)
      if DateHour == ':' and DateInd[j+3] == ':' and DateInd[j-2] == ' ':
         DateHour1 = DateInd[j-1]
         HOURList_w.append(DateHour1)


HOURList_w = np.asarray(HOURList_w)
HOURListINT_w = []
for i in range(len(HOURList_w)):
   HOURListINT_w_i = int(HOURList_w[i])
   HOURListINT_w.append(HOURListINT_w_i)

HOURListINT_w = np.array(HOURListINT_w)




for i in range(len(HOURListINT w)):
```

```
    if HOURListINT_w[i] < 12 and TIMEList_w[i] == 'PM':
        HOURListINT_w[i] = HOURListINT_w[i]+12

    if HOURListINT_w[i] == 12 and TIMEList_w[i] == 'AM':
        HOURListINT_w[i] = HOURListINT_w[i]+12


fig, (ax1, ax2, ax3) = plt.subplots(1,3)
fig.set_figwidth(15)

ax1.scatter(HOURListINT_w, WeightLb)
ax1.set_xlabel("Hour")
ax1.set_ylabel("Weight (lb)")
ax1.set_title("Weight LogInfo at x hour")
#plt.show()


ax2.scatter(HOURListINT_w, BMI)
ax2.set_xlabel("BMI LogInfo at x hour")
ax2.set_ylabel("BMI")
#plt.show()


ax3.scatter(HOURListINT_w, Fat)
ax3.set_xlabel("Fat LogInfo at x hour")
ax3.set_ylabel("Fat")
#plt.show()
```

Out[13]: Text(0, 0.5, 'Fat')

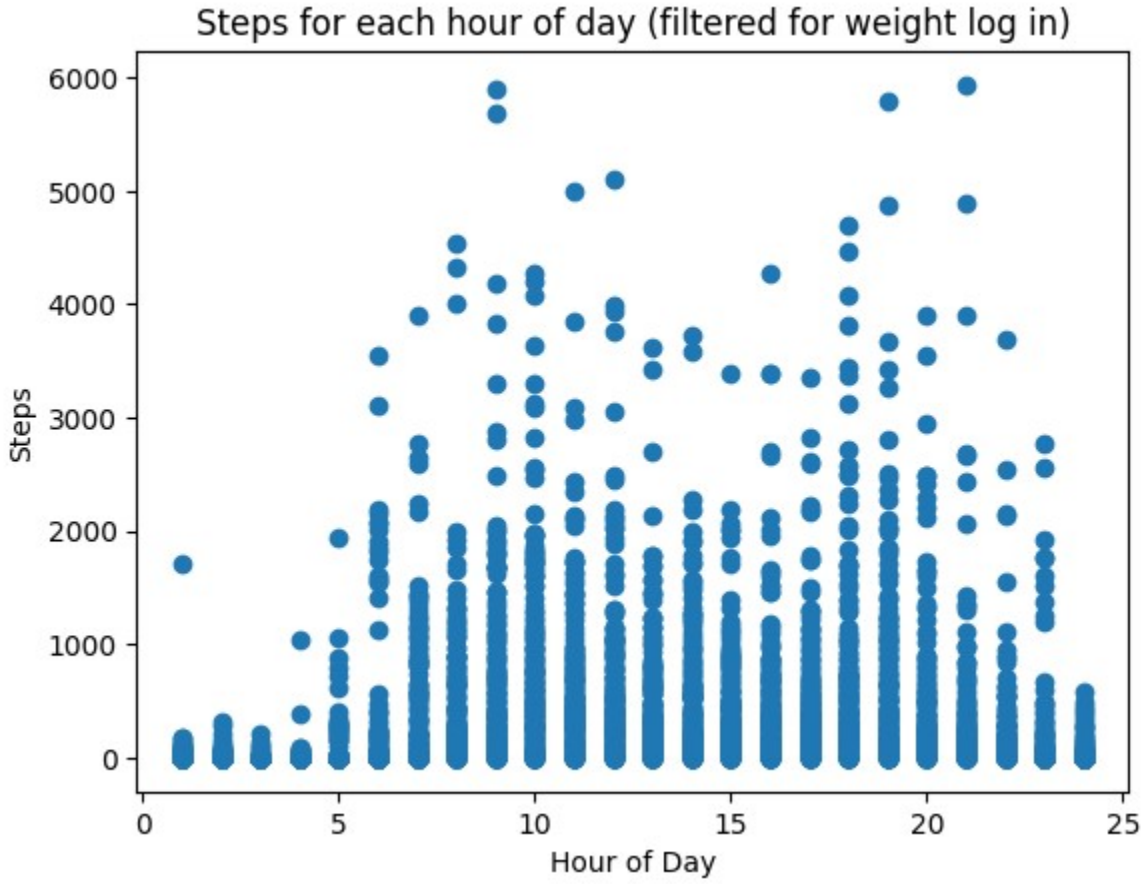


In [28]:
```
ID_steps_w = []
for i in range(len(StepTotal)):
   for j in range(len(ID_w)):
      if ID_steps[i] == ID_w[j]:
         ID_steps_i = ID_steps[i]
         ID_steps_w.append(ID_steps_i)

ID_steps_w1 = []
for i in range(len(ID_steps_w)-1):
   if ID_steps_w[i] != ID_steps_w[i+1]:
      ID_steps_w1.append(ID_steps_w[i])

#print(ID_steps_w1)


StepTotal_w = []
for i in range(len(StepTotal)):
    for j in range(len(ID_steps_w1)):
        if ID_steps[i] == ID_steps_w1[j]:
            StepTotal_w.append(StepTotal[i])
```

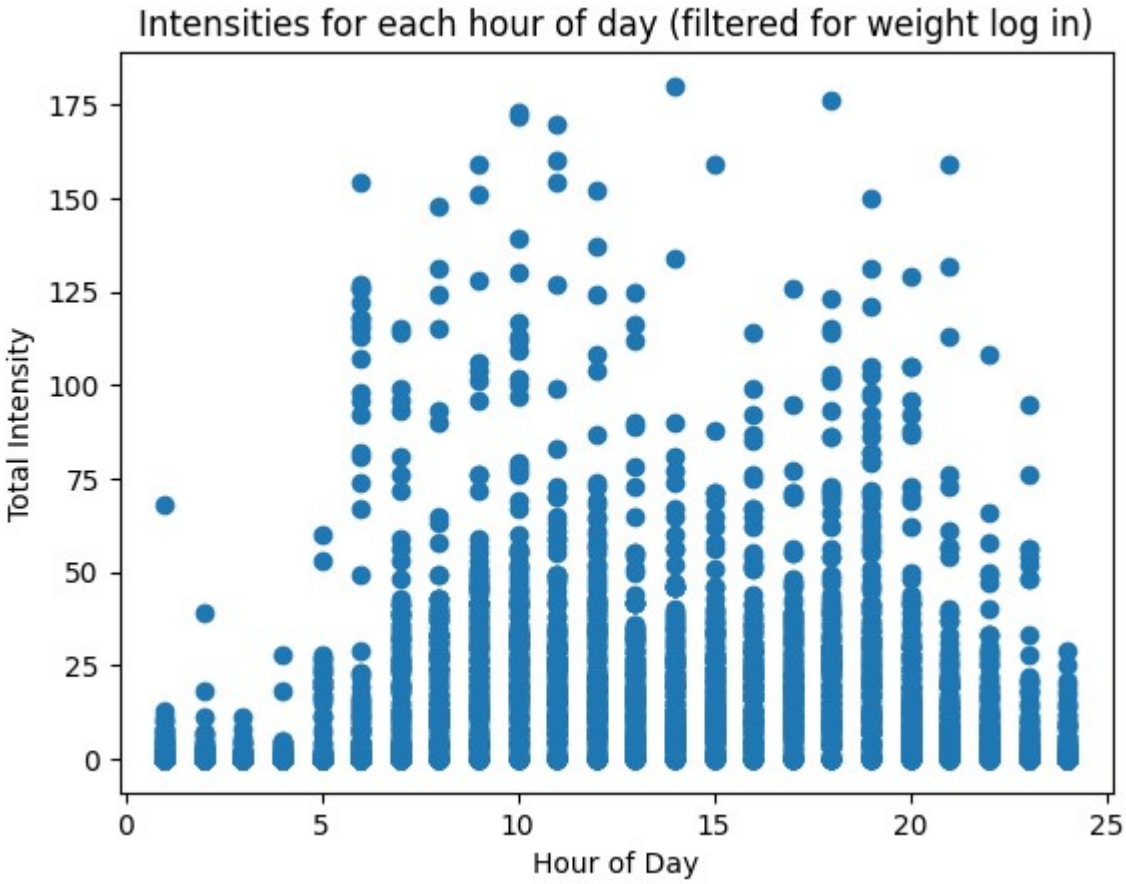
In [54]:
```
ID_steps_w = []
for i in range(len(StepTotal)):
   for j in range(len(ID_w)):
      if ID_steps[i] == ID_w[j]:
         ID_steps_i = ID_steps[i]
         ID_steps_w.append(ID_steps_i)

ID_steps_w1 = []
for i in range(len(ID_steps_w)-1):
   if ID_steps_w[i] != ID_steps_w[i+1]:
      ID_steps_w1.append(ID_steps_w[i])

print(ID_steps_w1)

StepTotal_w = []
HOURListstep_w = []
for i in range(len(StepTotal)):
   for j in range(len(ID_steps_w1)):
      if ID_steps[i] == ID_steps_w1[j]:
         StepTotal_w.append(StepTotal[i])
         HOURListstep_w.append(HOURListINT_step[i])


print(len(StepTotal_w))

plt.scatter(HOURListstep_w,StepTotal_w)
plt.title('Steps for each hour of day (filtered for weight log in)')
plt.xlabel('Hour of Day')
plt.ylabel('Steps')
plt.show()









############## Filter Intensities for IDs in which weight is recorded #############




ID_Intense_w = []
for i in range(len(TotalIntensity)):
   for j in range(len(ID_w)):
      if ID_intense[i] == ID_w[j]:
         ID_Intense_i = ID_intense[i]
         ID_Intense_w.append(ID_Intense_i)

print(len(ID_Intense_w))

ID_Intense_w1 = []
for i in range(len(ID_Intense_w)-1):
   if ID_Intense_w[i] != ID_Intense_w[i+1]:
```

```
        ID_Intense_w1.append(ID_Intense_w[i])


TotalIntensity_w = []
HOURList_Intense_w = []
for i in range(len(TotalIntensity)):
   for j in range(len(ID_Intense_w1)):
      if ID_intense[i] == ID_Intense_w1[j]:
         TotalIntensity_w.append(TotalIntensity[i])
         HOURList_Intense_w.append(HOURListINT_Intense[i])



plt.scatter(HOURList_Intense_w,TotalIntensity_w)
plt.title('Intensities for each hour of day (filtered for weight log in)')
plt.xlabel('Hour of Day')
plt.ylabel('Total Intensity')
plt.show()




fig, (ax1, ax2, ax3) = plt.subplots(1,3)

fig.set_figwidth(15)

ax1.scatter(HOURListstep_w,StepTotal_w)
ax1.set_title('Steps for each hour of day (filtered for weight log in)')
ax1.set_xlabel('Hour of Day')
ax1.set_ylabel('Steps')



ax2.scatter(HOURList_Intense_w,TotalIntensity_w)
ax2.set_title('Intensities for each hour of day (filtered for weight log in)')
ax2.set_xlabel('Hour of Day')
ax2.set_ylabel('Total Intensity')


ax3.scatter(HOURListINT_w, WeightLb)
ax3.set_xlabel("Hour")
ax3.set_ylabel("Weight (lb)")
ax3.set_title("Weight LogInfo at x hour")
```

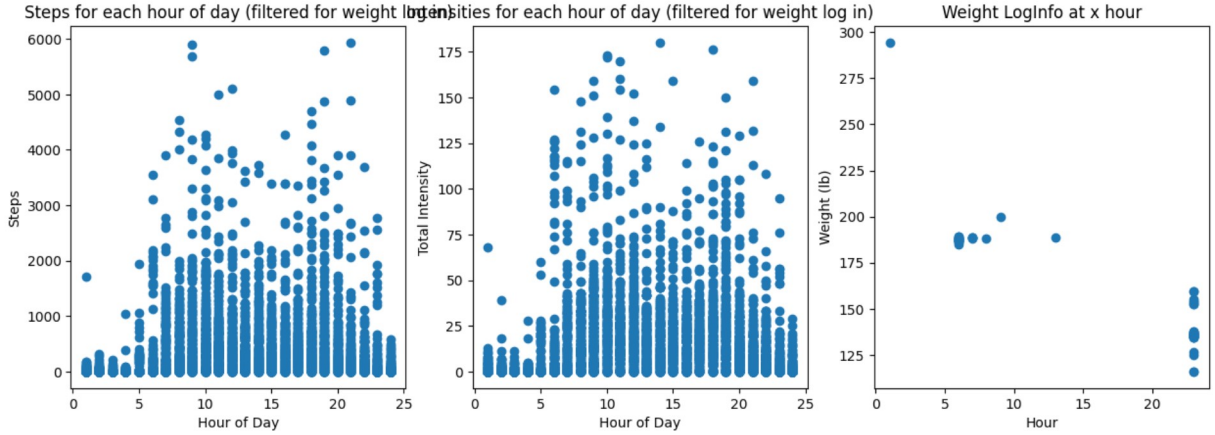
```
[1503960366, 1927972279, 2873212765, 4319703577, 4558609924, 5577150313, 6962181067]
5089
```


Steps for each hour of day (filtered for weight log in)
Steps
Hour of Day
0
1000
2000
3000
4000
5000
6000
0
5
10
15
20
25

```
49078
```


Intensities for each hour of day (filtered for weight log in)
Total Intensity
Hour of Day
0
25
50
75
100
125
150
175
0
5
10
15
20
25

Out[54]: Text(0.5, 1.0, 'Weight LogInfo at x hour')



Insights and Conclusions

From the graphs on intensities and steps vs the hour of day, it seems that most people do not exercise much or take that many steps in a day. It does appear that for those people who do exercise, most exercise in the daytime, at and after 6 A.M. This trend is also present in the steps data. The amount of intensities and steps seems to be roughly constant over the days of the weeks that data was recorded for. The correlation coefficient was found to be 0.81 between calories and the total steps, and 0.89 between calories and intensities. So both intensities and steps are correlated to the calories, which is to be expected. The vast majority of users did not have data recorded for weight, fat, or BMI, so it seems that most users are not very health conscious. For the few who did have weight data recorder, most of them were either overweight or at a healthy weight. One data point showed a BMI of above 45, which would be extremely overweight. The cooresponding weigth is roughly 300lbs, however the limitations of BMI should be mentioned here. If there were a very healthy person with a lot of muscles, this person would be considered by the BMI scale to be very obese. This is because BMI does not differentiate between muscle mass and fat mass. So this outlier data point could either be an extremely unhealthy or an extremely healthy person.

Given the trends in this data, the smartwatch market appears to be average people who are not extremely health conscious. Most people take 6,000 or less steps per day and burn a bit over 400 calories a day. Most people sleep for 6.99 hours with a variance of 3.88. It is generally recommended to get 8 hours of sleep a day. Seven hours of sleep may be fine for most people, however more sleep is always better. The percentage of time spent being sedentary for all the users in the daily activity dataset is 96%.

From these insights, I suggest that Bellabeat implement some reminders within the Bellabeat app to inform its users of the importance of sleep and not being sedentary. I would also suggest that the Bellabeat devices track sleep quality. Some helpful tips could be given in the app that go beyond just a buzz on the wrist to remind people to not be sedentary. For instance, an explanation could be given on why it is important to not be sedentary. In terms of the Bellabeat product itself, since most people from the data do not appear to be very health conscious, perhaps the Bellabeat products should focus more on the aesthetic features of their devices.