```
1 # Análisis de datos - Caso Práctico Bellabeat - Python
```

Escenario

Bellabeat, fabricante de productos de alta tecnología orientados a la salud de la mujer. Bellabeat es una empresa pequeña exitosa, pero tiene el potencial para convertirse en un actor más grande en el mercado global de dispositivos inteligentes. Urška Sršen, cofundadora y directora creativa de Bellabeat, cree que analizar los datos de actividad física de los dispositivos inteligentes podría desplegar nuevas oportunidades de negocio para la empresa. Sršen sabe que el análisis de los datos de consumo disponibles de Bellabeat revelaría nuevas oportunidades de crecimiento. Ella le pidió al equipo de análisis computacional de datos de marketing que se concentrara en un producto Bellabeat y analizara los datos de uso de dispositivos inteligentes para conocer cómo las personas están usando sus dispositivos inteligentes. Después, con esta información, le gustaría recibir recomendaciones de alto nivel sobre cómo estas tendencias pueden colaborar en la estrategia de marketing de Bellabeat.

Personajes y productos

Personajes ¶

- Urška Sršen: Cofundadora y directora creativa de Bellabeat
- Sando Mur: Matemático y cofundador de Bellabeat, miembro clave del equipo ejecutivo de Bellabeat.
- Equipo de análisis computacional de datos de marketing de Bellabeat: Un equipo de analistas de datos que se encarga de recopilar, analizar e informar datos que ayudan a conducir la estrategia de marketing de Bellabeat.

Productos

- Aplicación Bellabeat: La aplicación Bellabeat proporciona a los usuarios datos de salud relacionados con su actividad física, sueño, estrés, ciclo menstrual y hábitos de conciencia plena. Estos datos pueden ayudar a los usuarios a comprender sus hábitos actuales y adoptar decisiones saludables. La aplicación Bellabeat se conecta a su línea de productos de bienestar inteligentes.
- Leaf: Dispositivo de seguimiento clásico de bienestar de Bellabeat que se puede usar como pulsera, collar o clip. El dispositivo Leaf se conecta a la aplicación Bellabeat para hacer un seguimiento de la actividad física, el sueño y el estrés.
- Time: Este reloj de bienestar combina el aspecto intemporal de un reloj clásico con la tecnología inteligente para hacer el seguimiento de la actividad física, el sueño y el estrés del usuario. El reloj Time se conecta a la aplicación Bellabeat para proporcionar información sobre el bienestar diario.
- Spring: Es una botella de agua que hace el seguimiento diario del consumo de agua mediante el uso de tecnología inteligente para garantizar la hidratación adecuada a lo largo del día. La botella Spring se conecta a la aplicación Bellabeat para hacer el seguimiento de los niveles de hidratación.
- Membresía de Bellabeat: Bellabeat también ofrece a los usuarios un programa de membresía mediante suscripción. La membresía brinda a los usuarios un acceso 24/7 a una orientación totalmente personalizada sobre nutrición, actividad física, sueño, salud y belleza y conciencia plena según el estilo de vida y las metas del usuario.

Acerca de la empresa

Urška Sršen y Sando Mur fundaron Bellabeat, una empresa de alta tecnología que fabrica productos inteligentes focalizados en el cuidado de la salud. Sršen usó su experiencia como artista para desarrollar una tecnología con un bonito diseño que informará e inspirará a las mujeres de todo el mundo. Recopilar datos sobre la actividad

física, el sueño, el estrés y la salud reproductiva le ha permitido a Bellabeat proporcionar a las mujeres conocimientos sobre su propia salud y sus hábitos. Desde su fundación, en 2013, Bellabeat creció a un ritmo vertiginoso y rápidamente se posicionó como empresa de bienestar impulsada por la tecnología para las mujeres.

En 2016, Bellabeat ya había inaugurado oficinas en todo el mundo y lanzado múltiples productos. Los productos Bellabeat pasaron a estar disponibles en línea a través de un creciente número de comerciantes minoristas además del canal de comercio electrónico propio de Bellabeat en su sitio web (https://bellabeat.com/ (https://bellabeat.com/)). La empresa invirtió en medios publicitarios tradicionales, como radio, cartelería en la vía pública, prensa gráfica y televisión, pero se centra mayormente en el marketing digital. Bellabeat invierte todo el año en Google Search, mantiene activas las páginas de Facebook e Instagram e interactúa de manera constante con los consumidores en Twitter. A su vez, Bellabeat publica anuncios por video en YouTube y avisos publicitarios en Red de Display de Google para apoyar las campañas en fechas de marketing claves.

Datos

Datos de seguimiento de actividad física de Fitbit https://www.kaggle.com/datasets/arashnic/fitbit (https://www.kaggle.com/datasets/arashnic/fitbit) (CC0: Dominio público, conjunto de datos disponibles a través de Mobius): Este conjunto de datos de Kaggle contiene el seguimiento de la actividad física personal en treinta usuarios de Fitbit. Treinta usuarios elegibles de Fitbit prestaron su consentimiento para el envío de datos personales de seguimiento que incluyen rendimiento de la actividad física en minutos, ritmo cardíaco y monitoreo del sueño. Incluye información sobre la actividad diaria, pasos y ritmo cardíaco que se puede usar para explorar los hábitos de los usuarios.

Este conjunto de datos generado por los encuestados de una encuesta distribuida a través de Amazon Mechanical Turk entre el 12.03.2016 y el 12.05.2016. Treinta usuarios elegibles de Fitbit dieron su consentimiento para el envío de datos de seguimiento personal, incluida la salida a nivel de minutos para la actividad física, la frecuencia cardíaca y el control del sueño. Los informes individuales se pueden analizar por ID de sesión de exportación (columna A) o marca de tiempo (columna B). La variación entre la salida representa el uso de diferentes tipos de rastreadores Fitbit y comportamientos/preferencias de seguimiento individuales.

Los datos se encuentran organizados en 18 archivos formato csv, su formato es largo con una columna de identificación por cada usuario y las otra columnas contienen información.

Análisis de sesgo o credibilidad de los datos (ROCCC)?

- R:Reliable(Confiabilidad): La muestra es de una empresa reconocida en el rubro, pero es de 30 usuarios, por lo que se considera una muestra pequeña, poco representativa para asumir tendencias y patrones. Baja
- Original(Origen de datos): Amazon Mechanical Turk. Media
- Comprensive(Datos coherentes para el análisis): Si, contienen datos coherentes, aunque no suficientes. Media
- Current (Vigencia de datos): No se encuentran vigentes, ya que fueron tomados en el año 2016 y la fecha del presente análisis es 03/2023. Baja.
- Cited (Cita origen de los datos-> Medio): Amazon Mechanical Turk. Media

Los datos se consideran poco confiables porque no es una muestra representativa, ya que corresponde a una encuenta de 30 usuarios. Son de un proveedor externo, aunque es una empresa reconocida. Se pueden considerar integrales ya que concuerdan con el objetivo empresarial, pero no estan actualizados ya que fueron tomados en 2016. Se desconocen otros aspectos de los usuarios encuestados como alimentación, edad, situación económica, entre otros aspectos que pueden influir directamente.

Ciclo de Análisis de Datos:

Para responder a las preguntas clave de la empresa, seguiremos los pasos del proceso de análisis de datos:

- Preguntar,

- Preparar,
- Procesar,
- Analizar,
- Compartir,
- Actuar.

1.- PREGUNTAR

Estas preguntas orientarán el análisis:

1. ¿Cuáles son algunas tendencias de uso de los dispositivos inteligentes?
2. ¿Cómo se podrían aplicar estas tendencias a los clientes de Bellabeat?
3. ¿Cómo podrían ayudar estas tendencias a influir en la estrategia de marketing de Bellabeat?

Tarea Empresarial(Objetivo del análisis):

Analizar los datos de uso de dispositivos inteligentes para saber cómo usan los consumidores los dispositivos inteligentes que no son de Bellabeat. Después centrarse en un producto Bellabeat y aplicar esos conocimientos. Por último entregar recomendaciones de alto nivel sobre cómo estas tendencias pueden colaborar en la estrategia de marketing de Bellabeat.

2.- PREPARAR

2.1.- Importar librerías

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

2.2.- Importamos los 18 archivos

A continuación se importarán todos los archivos para realizar una exploración por cada uno y seleccionar cuales serán analizados. Es importante considerar que es una encuenta de 30 ususarios, por lo que se debe verificar que exista esa cantidad de usuarios en cada archivo.

In [2]:
```
dailyActivity = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\dail
dailyCalories = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\dail
dailyIntensities = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\d
dailySteps = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\dailySt
heartrate_seconds = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\
hourlyCalories = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\hou
hourlyIntensities = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\
hourlySteps = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\hourly
minuteCaloriesNarrow =pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGL
minuteCaloriesWide = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE
minuteIntensitiesNarrow = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_C
minuteIntensitiesWide = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOO
minuteMETsNarrow = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\m
minuteSleep = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\minute
minuteStepsNarrow = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\
minuteStepsWide = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\mi
sleepDay = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\sleepDay_
weightLogInfo = pd.read_csv(r'G:\Mi unidad\Analisis de Datos_PRACTICA\Caso_2_GOOGLE\weig
```

2.2.1.- dailyActivity

In [3]:
```
dailyActivity.head()
```

Out[3]:

	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryActiveDistan
0	1503960366	4/12/2016	13162	8.50	8.50	0.0	1.
1	1503960366	4/13/2016	10735	6.97	6.97	0.0	1.
2	1503960366	4/14/2016	10460	6.74	6.74	0.0	2.
3	1503960366	4/15/2016	9762	6.28	6.28	0.0	2.
4	1503960366	4/16/2016	12669	8.16	8.16	0.0	2.

In [4]:
```
dailyActivity['Id'].nunique()
```

Out[4]:
```
33
```

In [5]:
```
dailyActivity['Id'].unique()
```

Out[5]:
```
array([1503960366, 1624580081, 1644430081, 1844505072, 1927972279,
       2022484408, 2026352035, 2320127002, 2347167796, 2873212765,
       3372868164, 3977333714, 4020332650, 4057192912, 4319703577,
       4388161847, 4445114986, 4558609924, 4702921684, 5553957443,
       5577150313, 6117666160, 6290855005, 6775888955, 6962181067,
       7007744171, 7086361926, 8053475328, 8253242879, 8378563200,
       8583815059, 8792009665, 8877689391], dtype=int64)
```

Observación:

- Aunque el proveedor indicó que era una encuesta de 30 usuarios se comprobó que existen 33 "Id" en este caso.

2.2.2.- dailyCalories

In [6]:
```
1 dailyCalories.head()
```

Out[6]:

	Id	ActivityDay	Calories
0	1503960366	4/12/2016	1985
1	1503960366	4/13/2016	1797
2	1503960366	4/14/2016	1776
3	1503960366	4/15/2016	1745
4	1503960366	4/16/2016	1863

Observación:

- Estos datos no se van a considerar para el presente análisis porque ya se encuentran incluidos en el archivo "dailyActivity".

2.2.3.- dailyIntensities

In [7]:
```
1 dailyIntensities.head()
```

Out[7]:

	Id	ActivityDay	SedentaryMinutes	LightlyActiveMinutes	FairlyActiveMinutes	VeryActiveMinutes	Seden
0	1503960366	4/12/2016	728	328	13	25	
1	1503960366	4/13/2016	776	217	19	21	
2	1503960366	4/14/2016	1218	181	11	30	
3	1503960366	4/15/2016	726	209	34	29	
4	1503960366	4/16/2016	773	221	10	36	

Observación:

- Estos datos no se van a considerar para el presente análisis porque ya se encuentran incluidos en el archivo "dailyActivity".

2.2.4.- dailySteps

In [8]:
```
1 dailySteps.head()
```

Out[8]:

	Id	ActivityDay	StepTotal
0	1503960366	4/12/2016	13162
1	1503960366	4/13/2016	10735
2	1503960366	4/14/2016	10460
3	1503960366	4/15/2016	9762
4	1503960366	4/16/2016	12669

Observación:

- Estos datos no se van a considerar para el presente análisis porque ya se encuentran incluidos en el archivo "dailyActivity"

2.2.5 .- heartrate_seconds

In [9]:
```
heartrate_seconds.head()
```

Out[9]:

	Id	Time	Value
0	2022484408	4/12/2016 7:21:00 AM	97
1	2022484408	4/12/2016 7:21:05 AM	102
2	2022484408	4/12/2016 7:21:10 AM	105
3	2022484408	4/12/2016 7:21:20 AM	103
4	2022484408	4/12/2016 7:21:25 AM	101

In [10]:
```
heartrate_seconds['Id'].nunique()
```

Out[10]: 14

Observación:

- Este archivo tiene datos de 14 usuarios, los cuales en este caso representan el 42%. Al ser un % tan alto de datos faltantes no serán considerados.

2.2.6.- hourlyCalories

In [11]:
```
hourlyCalories.head()
```

Out[11]:

	Id	ActivityHour	Calories
0	1503960366	4/12/2016 12:00:00 AM	81
1	1503960366	4/12/2016 1:00:00 AM	61
2	1503960366	4/12/2016 2:00:00 AM	59
3	1503960366	4/12/2016 3:00:00 AM	47
4	1503960366	4/12/2016 4:00:00 AM	48

In [12]:
```
hourlyCalories['Id'].nunique()
```

Out[12]: 33

In [13]:
```
hourlyCalories.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22099 entries, 0 to 22098
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Id            22099 non-null  int64 
 1   ActivityHour  22099 non-null  object
 2   Calories      22099 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 518.1+ KB
```

Observación:

- Se tomarán estos datos para su posterior análisis ya que no presenta valores NA y tiene información referente a 33 usuarios.

2.2.7.- hourlyIntensities

In [14]:
```
1 hourlyIntensities.head()
```

Out[14]:

	Id	ActivityHour	TotalIntensity	AverageIntensity
0	1503960366	4/12/2016 12:00:00 AM	20	0.333333
1	1503960366	4/12/2016 1:00:00 AM	8	0.133333
2	1503960366	4/12/2016 2:00:00 AM	7	0.116667
3	1503960366	4/12/2016 3:00:00 AM	0	0.000000
4	1503960366	4/12/2016 4:00:00 AM	0	0.000000

Observación:

- No se considerará este archivo para el análisis.

2.2.8 .- hourlySteps

In [15]:
```
1 hourlySteps.head()
```

Out[15]:

	Id	ActivityHour	StepTotal
0	1503960366	4/12/2016 12:00:00 AM	373
1	1503960366	4/12/2016 1:00:00 AM	160
2	1503960366	4/12/2016 2:00:00 AM	151
3	1503960366	4/12/2016 3:00:00 AM	0
4	1503960366	4/12/2016 4:00:00 AM	0

In [16]:
```
1 hourlySteps['Id'].nunique()
```

Out[16]: 33

Observación:

- Se tomarán estos datos para su posterior análisis ya que no presenta valores NA y tiene información referente a 33 usuarios.

2.2.9.- minuteCaloriesNarrow

In [17]:
```
1 minuteCaloriesNarrow.head()
```

Out[17]:

	Id	ActivityMinute	Calories
0	1503960366	4/12/2016 12:00:00 AM	0.7865
1	1503960366	4/12/2016 12:01:00 AM	0.7865
2	1503960366	4/12/2016 12:02:00 AM	0.7865
3	1503960366	4/12/2016 12:03:00 AM	0.7865
4	1503960366	4/12/2016 12:04:00 AM	0.7865

Observación:

- No se considerará este archivo para el análisis.

2.2.10.- minuteCaloriesWide

In [18]:
```
1 minuteCaloriesWide.head()
```

Out[18]:

	Id	ActivityHour	Calories00	Calories01	Calories02	Calories03	Calories04	Calories05	Calories06	Ca
0	1503960366	4/13/2016 12:00:00 AM	1.8876	2.2022	0.9438	0.9438	0.9438	2.0449	0.9438	
1	1503960366	4/13/2016 1:00:00 AM	0.7865	0.7865	0.7865	0.7865	0.9438	0.9438	0.9438	
2	1503960366	4/13/2016 2:00:00 AM	0.7865	0.7865	0.7865	0.7865	0.7865	0.7865	0.7865	
3	1503960366	4/13/2016 3:00:00 AM	0.7865	0.7865	0.7865	0.7865	0.7865	0.7865	0.7865	
4	1503960366	4/13/2016 4:00:00 AM	0.7865	0.7865	0.7865	0.7865	0.7865	0.7865	0.7865	

5 rows × 62 columns

Observación:

- No se considerará este archivo para el análisis.

2.2.11.- minuteIntensitiesNarrow

In [19]:
```
1 minuteIntensitiesNarrow.head()
```

Out[19]:

	Id	ActivityMinute	Intensity
0	1503960366	4/12/2016 12:00:00 AM	0
1	1503960366	4/12/2016 12:01:00 AM	0
2	1503960366	4/12/2016 12:02:00 AM	0
3	1503960366	4/12/2016 12:03:00 AM	0
4	1503960366	4/12/2016 12:04:00 AM	0

Observación:

- No se considerará este archivo para el análisis.

2.2.12.- minuteIntensitiesWide

In [20]:
```
1 minuteIntensitiesWide.head()
```

Out[20]:

	Id	ActivityHour	Intensity00	Intensity01	Intensity02	Intensity03	Intensity04	Intensity05	Intensity06
0	1503960366	4/13/2016 12:00:00 AM	1	1	0	0	0	1	0
1	1503960366	4/13/2016 1:00:00 AM	0	0	0	0	0	0	0
2	1503960366	4/13/2016 2:00:00 AM	0	0	0	0	0	0	0
3	1503960366	4/13/2016 3:00:00 AM	0	0	0	0	0	0	0
4	1503960366	4/13/2016 4:00:00 AM	0	0	0	0	0	0	0

5 rows × 62 columns

Observación:

- No se considerará este archivo para el análisis.

2.2.13.-minuteMETsNarrow

In [21]:
```
1 minuteMETsNarrow.head()
```

Out[21]:

	Id	ActivityMinute	METs
0	1503960366	4/12/2016 12:00:00 AM	10
1	1503960366	4/12/2016 12:01:00 AM	10
2	1503960366	4/12/2016 12:02:00 AM	10
3	1503960366	4/12/2016 12:03:00 AM	10
4	1503960366	4/12/2016 12:04:00 AM	10

Observación:

- No se considerará este archivo para el análisis.

2.2.14.- minuteSleep

In [22]:
```
1 minuteSleep.head()
```

Out[22]:

	Id	date	value	logId
0	1503960366	4/12/2016 2:47:30 AM	3	11380564589
1	1503960366	4/12/2016 2:48:30 AM	2	11380564589
2	1503960366	4/12/2016 2:49:30 AM	1	11380564589
3	1503960366	4/12/2016 2:50:30 AM	1	11380564589
4	1503960366	4/12/2016 2:51:30 AM	1	11380564589

Observación:

- No se considerará este archivo para el análisis.

2.2.15.- minuteStepsNarrow

In [23]:
```
1 minuteStepsNarrow.head()
```

Out[23]:

	Id	ActivityMinute	Steps
0	1503960366	4/12/2016 12:00:00 AM	0
1	1503960366	4/12/2016 12:01:00 AM	0
2	1503960366	4/12/2016 12:02:00 AM	0
3	1503960366	4/12/2016 12:03:00 AM	0
4	1503960366	4/12/2016 12:04:00 AM	0

Observación:

- No se considerará este archivo para el análisis.

2.2.16.- minuteStepsWide

In [24]:
```
1 minuteStepsWide.head()
```

Out[24]:

	Id	ActivityHour	Steps00	Steps01	Steps02	Steps03	Steps04	Steps05	Steps06	Steps07	...	Steps
0	1503960366	4/13/2016 12:00:00 AM	4	16	0	0	0	9	0	17	...	
1	1503960366	4/13/2016 1:00:00 AM	0	0	0	0	0	0	0	0	...	
2	1503960366	4/13/2016 2:00:00 AM	0	0	0	0	0	0	0	0	...	
3	1503960366	4/13/2016 3:00:00 AM	0	0	0	0	0	0	0	0	...	
4	1503960366	4/13/2016 4:00:00 AM	0	0	0	0	0	0	0	0	...	

5 rows × 62 columns

Observación:

- No se considerará este archivo para el análisis.

2.2.17.- sleepDay

In [25]:
```
1 sleepDay.head()
```

Out[25]:

	Id	SleepDay	TotalSleepRecords	TotalMinutesAsleep	TotalTimeInBed
0	1503960366	4/12/2016 12:00:00 AM	1	327	346
1	1503960366	4/13/2016 12:00:00 AM	2	384	407
2	1503960366	4/15/2016 12:00:00 AM	1	412	442
3	1503960366	4/16/2016 12:00:00 AM	2	340	367
4	1503960366	4/17/2016 12:00:00 AM	1	700	712

In [26]:
```
1 sleepDay['Id'].nunique()
```

Out[26]: 24

In [27]:
```
1 sleepDay.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 413 entries, 0 to 412
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Id                  413 non-null    int64 
 1   SleepDay            413 non-null    object
 2   TotalSleepRecords   413 non-null    int64 
 3   TotalMinutesAsleep  413 non-null    int64 
 4   TotalTimeInBed      413 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 16.3+ KB
```

Observación:

- Estos datos se van a considerar para el análisis, a pesar de que cuenta con información de 24 usuarios y no de 33 como los archivos anteriores. A priori los consideraremos para más exploración y evaluación.

2.2.18 weightLogInfo

In [28]:
```
1 weightLogInfo.head()
```

Out[28]:

	Id	Date	WeightKg	WeightPounds	Fat	BMI	IsManualReport	LogId
0	1503960366	5/2/2016 11:59:59 PM	52.599998	115.963147	22.0	22.650000	True	1462233599000
1	1503960366	5/3/2016 11:59:59 PM	52.599998	115.963147	NaN	22.650000	True	1462319999000
2	1927972279	4/13/2016 1:08:52 AM	133.500000	294.317120	NaN	47.540001	False	1460509732000
3	2873212765	4/21/2016 11:59:59 PM	56.700001	125.002104	NaN	21.450001	True	1461283199000
4	2873212765	5/12/2016 11:59:59 PM	57.299999	126.324875	NaN	21.690001	True	1463097599000

In [29]:
```
1 weightLogInfo['Id'].nunique()
```

Out[29]: 8

Observación:

- No se va a considerar este archivo puesto que solo cuenta con información de 8 usuarios.

Datos seleccionados

Se va a centralizar toda la exploración y evaluación en los siguientes datos:

- dailyActivity,
- hourlyCalories,
- hourlySteps,
- sleepDay.

3.- PROCESAR

En esta etapa de procesamiento vamos a desarrollar los siguientes pasos:

- 3.1.- Eliminar filas duplicadas
- 3.2.- Evaluar valores nulos
- 3.3.- Cambiar el formato de datos - Crear columnas nuevas
- 3.4.- Evaluar valores atípicos

3.1.- Eliminar duplicados

In [30]:
```
1 print(f'El tamaño del set antes de eliminar duplicados es: {dailyActivity.shape}')
2 dailyActivity.drop_duplicates(inplace=True)
3 print(f'El tamaño del set después de eliminar duplicados es: {dailyActivity.shape}')
```

```
El tamaño del set antes de eliminar duplicados es: (940, 15)
El tamaño del set después de eliminar duplicados es: (940, 15)
```

In [31]:
```
print(f'El tamaño del set antes de eliminar duplicados es: {hourlyCalories.shape}')
hourlyCalories.drop_duplicates(inplace=True)
print(f'El tamaño del set después de eliminar duplicados es: {hourlyCalories.shape}')
```

```
El tamaño del set antes de eliminar duplicados es: (22099, 3)
El tamaño del set después de eliminar duplicados es: (22099, 3)
```

In [32]:
```
print(f'El tamaño del set antes de eliminar duplicados es: {hourlySteps.shape}')
hourlySteps.drop_duplicates(inplace=True)
print(f'El tamaño del set después de eliminar duplicados es: {hourlySteps.shape}')
```

```
El tamaño del set antes de eliminar duplicados es: (22099, 3)
El tamaño del set después de eliminar duplicados es: (22099, 3)
```

In [33]:
```
print(f'El tamaño del set antes de eliminar duplicados es: {sleepDay.shape}')
sleepDay.drop_duplicates(inplace=True)
print(f'El tamaño del set después de eliminar duplicados es: {sleepDay.shape}')
```

```
El tamaño del set antes de eliminar duplicados es: (413, 5)
El tamaño del set después de eliminar duplicados es: (410, 5)
```

Observación: Solo habían datos duplicados en sleepDay y fueron borrados.

3.2.- Evaluar valores nulos

Para detectar los valores faltantes se usará .isna() y luego para sumar .sum()

In [34]:
```
dailyActivity.isna().sum()
```

Out[34]:
```
Id                          0
ActivityDate                0
TotalSteps                  0
TotalDistance               0
TrackerDistance             0
LoggedActivitiesDistance    0
VeryActiveDistance          0
ModeratelyActiveDistance    0
LightActiveDistance         0
SedentaryActiveDistance     0
VeryActiveMinutes           0
FairlyActiveMinutes         0
LightlyActiveMinutes        0
SedentaryMinutes            0
Calories                    0
dtype: int64
```

In [35]:
```
hourlyCalories.isna().sum()
```

Out[35]:
```
Id              0
ActivityHour    0
Calories        0
dtype: int64
```

In [36]:
```
1 hourlySteps.isna().sum()
```

Out[36]:
```
Id             0
ActivityHour   0
StepTotal      0
dtype: int64
```

In [37]:
```
1 sleepDay.isna().sum()
```

Out[37]:
```
Id                  0
SleepDay            0
TotalSleepRecords   0
TotalMinutesAsleep  0
TotalTimeInBed      0
dtype: int64
```

Observación:

- Se ha verificado que los datos no cuentan con valores nulos.

3.3.- Cambiar el formato de datos - Crear variable nuevas

Cambios en dailyActivity

- Pasaremos la columna 'ActivityDate' a formato 'datetime64' para posteriormente realizar cálculos de tiempo.
- Cambiaremos el nombre de dicha columna

In [38]:
```
1 dailyActivity = dailyActivity.astype({'ActivityDate': 'datetime64'})
2 dailyActivity.rename(columns={'ActivityDate': 'Date'}, inplace=True)
```

A partir de la columna 'Date' vamos a crear columnas nuevas:

- Year
- Month
- Day_week

In [39]:
```
1  dailyActivity['Year'] = dailyActivity['Date'].dt.year
2
3  meses = ['January', 'February', 'March', 'April', 'May', 'June',
4           'July', 'August', 'September', 'October', 'November', 'December']
5  dailyActivity['Month'] = dailyActivity['Date'].dt.month_name()
6  dailyActivity['Month'] = dailyActivity['Month'].astype(pd.api.types.CategoricalDtype(cat
7
8  dias = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
9  dailyActivity['Day_week'] = dailyActivity['Date'].dt.day_name()
10 dailyActivity['Day_week'] = dailyActivity['Day_week'].astype(pd.api.types.CategoricalDty
```

Cambios en hourlyCalories

- Pasaremos la columna 'ActivityHour' a formato 'datetime64' para posteriormente realizar cálculos de tiempo.
- Cambiaremos el nombre de dicha columna

In [40]:
```
hourlyCalories = hourlyCalories.astype({'ActivityHour': 'datetime64'})
hourlyCalories.rename(columns={'ActivityHour': 'Datehour'}, inplace=True)
```

A partir de la columna 'Datehour' vamos a crear columnas nuevas:

- Year
- Month
- Day_week
- Hour

In [41]:
```
hourlyCalories['Year'] = hourlyCalories['Datehour'].dt.year

meses = ['January', 'February', 'March', 'April', 'May', 'June',
         'July', 'August', 'September', 'October', 'November', 'December']
hourlyCalories['Month'] = hourlyCalories['Datehour'].dt.month_name()
hourlyCalories['Month'] = hourlyCalories['Month'].astype(pd.api.types.CategoricalDtype(c

dias = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
hourlyCalories['Day_week'] = hourlyCalories['Datehour'].dt.day_name()
hourlyCalories['Day_week'] = hourlyCalories['Day_week'].astype(pd.api.types.CategoricalD

hourlyCalories['Hour'] = hourlyCalories['Datehour'].dt.hour
```

Cambios en hourlySteps

- Pasaremos la columna 'ActivityHour' a formato 'datetime64' para posteriormente realizar cálculos de tiempo.
- Cambiaremos el nombre de dicha columna

In [42]:
```
hourlySteps = hourlySteps.astype({'ActivityHour': 'datetime64'})
hourlySteps.rename(columns={'ActivityHour': 'Datehour'}, inplace=True)
```

A partir de la columna 'Datehour' vamos a crear columnas nuevas:

- Year
- Month
- Day_week
- Hour

In [43]:
```
hourlySteps['Year'] = hourlySteps['Datehour'].dt.year

meses = ['January', 'February', 'March', 'April', 'May', 'June',
         'July', 'August', 'September', 'October', 'November', 'December']
hourlySteps['Month'] = hourlySteps['Datehour'].dt.month_name()
hourlySteps['Month'] = hourlySteps['Month'].astype(pd.api.types.CategoricalDtype(categor

dias = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
hourlySteps['Day_week'] = hourlySteps['Datehour'].dt.day_name()
hourlySteps['Day_week'] = hourlySteps['Day_week'].astype(pd.api.types.CategoricalDtype(c

hourlySteps['Hour'] = hourlySteps['Datehour'].dt.hour
```

Cambios en sleepDay

- Pasaremos la columna 'SleepDay' a formato 'datetime64' para posteriormente realizar cálculos de tiempo.
- Cambiaremos el nombre de dicha columna

In [44]:
```
sleepDay = sleepDay.astype({'SleepDay': 'datetime64'})
sleepDay.rename(columns={'SleepDay': 'Date'}, inplace=True)
```

In [45]:
```
sleepDay.head()
```

Out[45]:

	Id	Date	TotalSleepRecords	TotalMinutesAsleep	TotalTimeInBed
0	1503960366	2016-04-12	1	327	346
1	1503960366	2016-04-13	2	384	407
2	1503960366	2016-04-15	1	412	442
3	1503960366	2016-04-16	2	340	367
4	1503960366	2016-04-17	1	700	712

A partir de la columna 'Datehour' vamos a crear columnas nuevas:

- Year
- Month
- Day_week

In [46]:
```
sleepDay['Year'] = sleepDay['Date'].dt.year

meses = ['January', 'February', 'March', 'April', 'May', 'June',
         'July', 'August', 'September', 'October', 'November', 'December']
sleepDay['Month'] = sleepDay['Date'].dt.month_name()
sleepDay['Month'] = sleepDay['Month'].astype(pd.api.types.CategoricalDtype(categories=me

dias = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']
sleepDay['Day_week'] = sleepDay['Date'].dt.day_name()
sleepDay['Day_week'] = sleepDay['Day_week'].astype(pd.api.types.CategoricalDtype(categor
```

3.4.- Evaluar valores atípicos

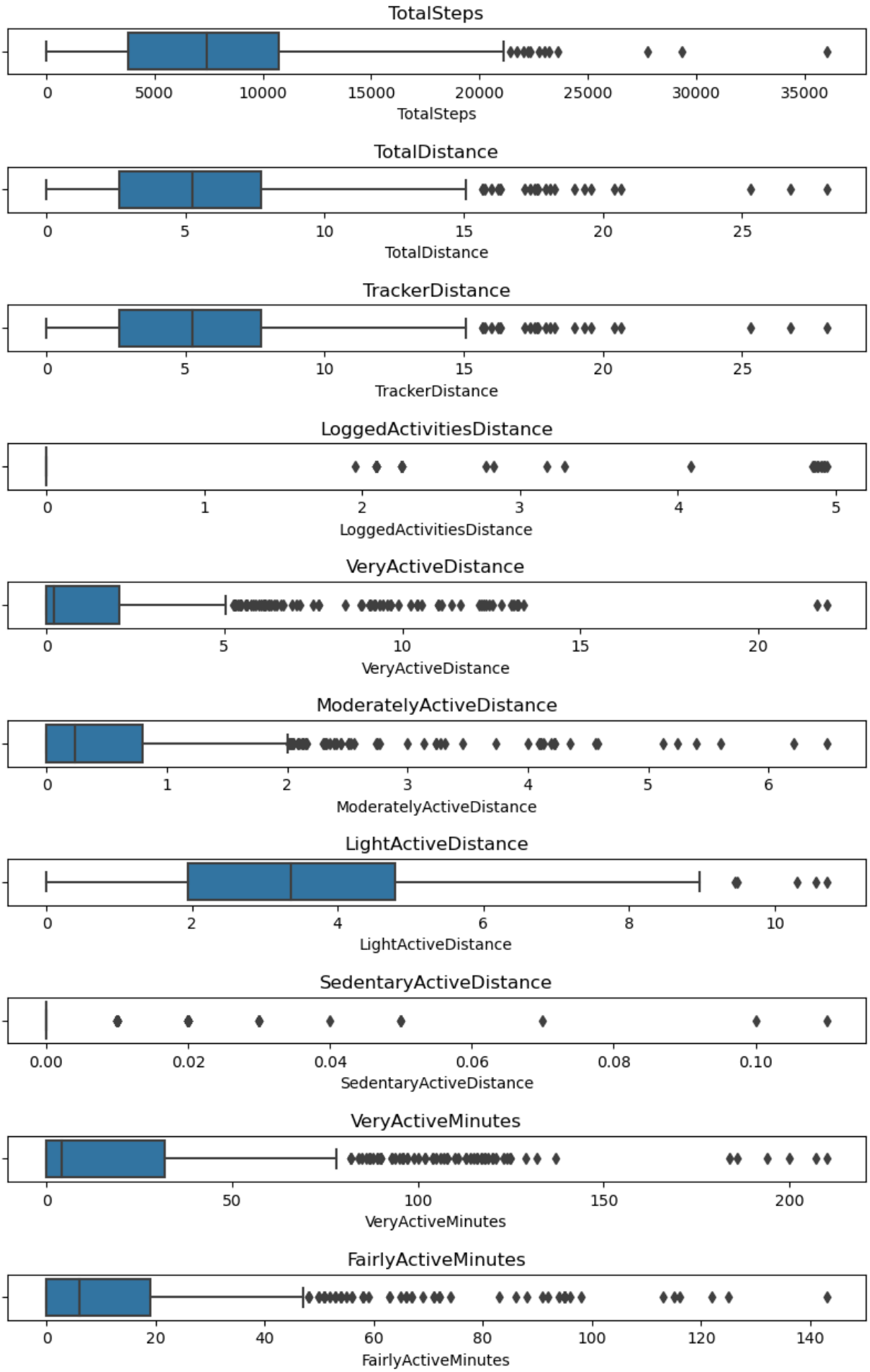
dailyActivity

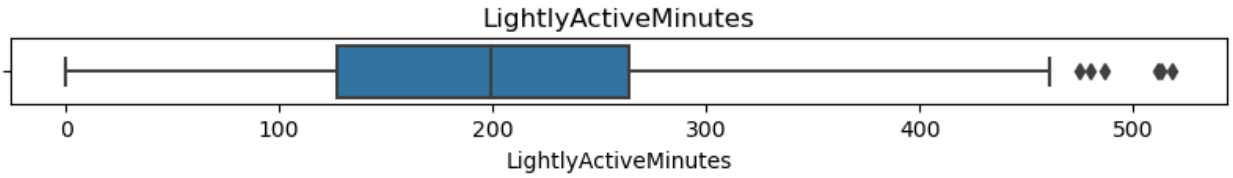
In [47]:
```
1 dailyActivity.info()
```

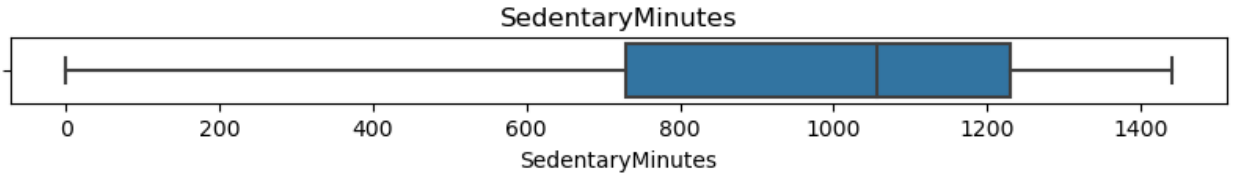
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 940 entries, 0 to 939
Data columns (total 18 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Id                        940 non-null    int64
 1   Date                      940 non-null    datetime64[ns]
 2   TotalSteps                940 non-null    int64
 3   TotalDistance             940 non-null    float64
 4   TrackerDistance           940 non-null    float64
 5   LoggedActivitiesDistance  940 non-null    float64
 6   VeryActiveDistance        940 non-null    float64
 7   ModeratelyActiveDistance  940 non-null    float64
 8   LightActiveDistance       940 non-null    float64
 9   SedentaryActiveDistance   940 non-null    float64
 10  VeryActiveMinutes         940 non-null    int64
 11  FairlyActiveMinutes       940 non-null    int64
 12  LightlyActiveMinutes      940 non-null    int64
 13  SedentaryMinutes          940 non-null    int64
 14  Calories                  940 non-null    int64
 15  Year                      940 non-null    int64
 16  Month                     940 non-null    category
 17  Day_week                  940 non-null    category
dtypes: category(2), datetime64[ns](1), float64(7), int64(8)
memory usage: 127.4 KB
```

In [48]:

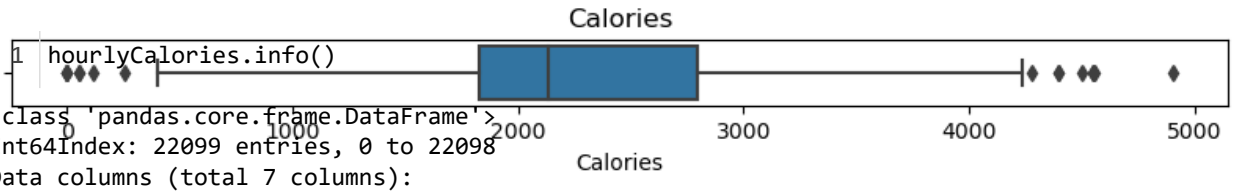
```
#Seleccionamos las columnas númericas para revisar los valores atípicos

cols_num = ['TotalSteps', 'TotalDistance', 'TrackerDistance',
       'LoggedActivitiesDistance', 'VeryActiveDistance',
       'ModeratelyActiveDistance', 'LightActiveDistance',
       'SedentaryActiveDistance', 'VeryActiveMinutes', 'FairlyActiveMinutes',
       'LightlyActiveMinutes', 'SedentaryMinutes', 'Calories']

fig, ax = plt.subplots(nrows=13, ncols=1, figsize=(10,20))
fig.subplots_adjust(hspace=2)

for i, col in enumerate(cols_num):
    sns.boxplot(x=col, data=dailyActivity, ax=ax[i])
    ax[i].set_title(col)
```


TotalSteps
0
5000
10000
15000
20000
25000
30000
35000
TotalSteps
TotalDistance
0
5
10
15
20
25
TotalDistance
TrackerDistance
0
5
10
15
20
25
TrackerDistance
LoggedActivitiesDistance
0
1
2
3
4
5
LoggedActivitiesDistance
VeryActiveDistance
0
5
10
15
20
VeryActiveDistance
ModeratelyActiveDistance
0
1
2
3
4
5
6
ModeratelyActiveDistance
LightActiveDistance
0
2
4
6
8
10
LightActiveDistance
SedentaryActiveDistance
0.00
0.02
0.04
0.06
0.08
0.10
SedentaryActiveDistance
VeryActiveMinutes
0
50
100
150
200
VeryActiveMinutes
FairlyActiveMinutes
0
20
40
60
80
100
120
140
FairlyActiveMinutes



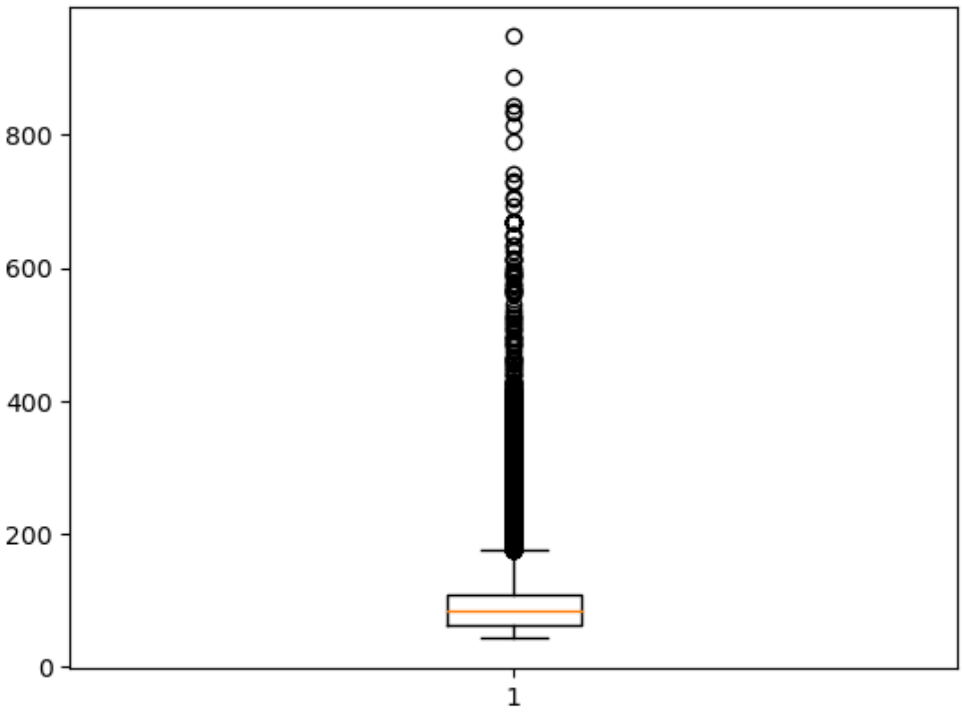


In [49]:
```
1 hourlyCalories.info()
```




```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22099 entries, 0 to 22098
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype         
---  ------    --------------  -----         
 0   Id        22099 non-null  int64         
 1   Datehour  22099 non-null  datetime64[ns]
 2   Calories  22099 non-null  int64         
 3   Year      22099 non-null  int64         
 4   Month     22099 non-null  category      
 5   Day_week  22099 non-null  category      
 6   Hour      22099 non-null  int64         
dtypes: category(2), datetime64[ns](1), int64(4)
memory usage: 1.1 MB
```

In [50]:
```
1 fig1_diag_de_cajas_cal = plt.boxplot(hourlyCalories['Calories'])
2 fig1_diag_de_cajas_cal;
```



Observación:

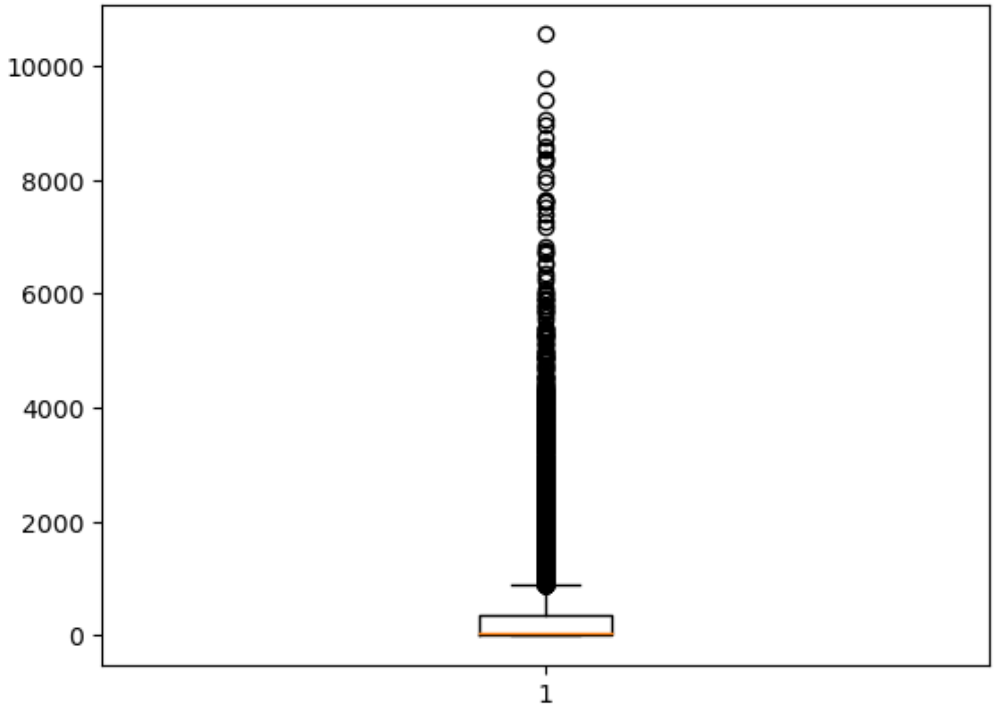
- Se detectan valores elevados pero podrían ser valores que se ajustan a determinadas actividades de alta

hourlySteps

In [51]:
```
1 hourlySteps.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22099 entries, 0 to 22098
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Id         22099 non-null  int64
 1   Datehour   22099 non-null  datetime64[ns]
 2   StepTotal  22099 non-null  int64
 3   Year       22099 non-null  int64
 4   Month      22099 non-null  category
 5   Day_week   22099 non-null  category
 6   Hour       22099 non-null  int64
dtypes: category(2), datetime64[ns](1), int64(4)
memory usage: 1.1 MB
```

In [52]:
```
1 fig2_diag_de_cajas_Steptotal = plt.boxplot(hourlySteps['StepTotal'])
2 fig2_diag_de_cajas_Steptotal;
```



Observación:

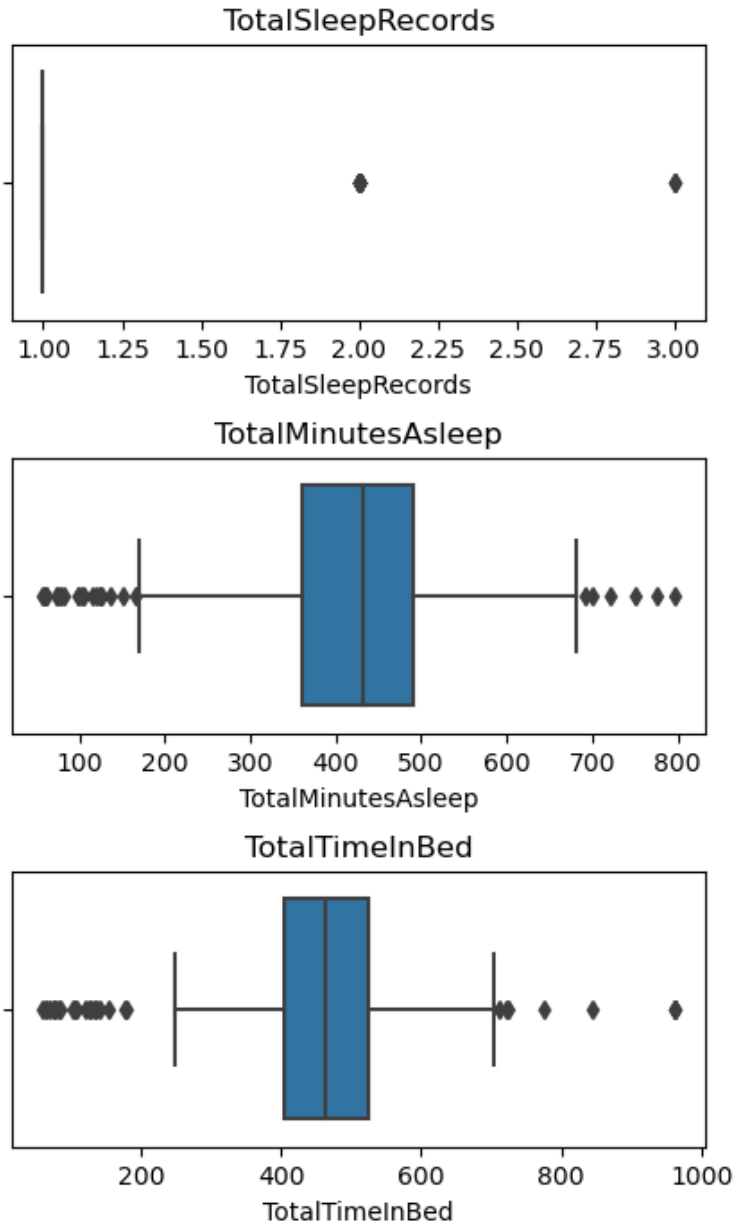
- Se deciden dejar los datos ya que los valores maximos podrían ser posibles para una persona que se ejercita o realiza actividades donde debe movilizarse con mucha frecuencia.

sleepDay

In [53]:
```
1 sleepDay.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 410 entries, 0 to 412
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype         
---  ------              --------------  -----         
 0   Id                  410 non-null    int64         
 1   Date                410 non-null    datetime64[ns]
 2   TotalSleepRecords   410 non-null    int64         
 3   TotalMinutesAsleep  410 non-null    int64         
 4   TotalTimeInBed      410 non-null    int64         
 5   Year                410 non-null    int64         
 6   Month               410 non-null    category      
 7   Day_week            410 non-null    category      
dtypes: category(2), datetime64[ns](1), int64(5)
memory usage: 24.0 KB
```

In [54]:
```
cols_num = ['TotalSleepRecords', 'TotalMinutesAsleep',
       'TotalTimeInBed']

fig, ax = plt.subplots(nrows=3, ncols=1, figsize=(5,8))
fig.subplots_adjust(hspace=0.5)

for i, col in enumerate(cols_num):
    sns.boxplot(x=col, data=sleepDay, ax=ax[i])
    ax[i].set_title(col)
```



Observaciones:

- El número de sueños al día podría ser 3, lo dejaremos porque es completamente razonable.
- Los tiempos en la cama y dormido pueden ser posibles y no existe ningún tiempo negativo por lo que no los eliminaremos de momento.

4.- ANALIZAR

"dailyActivity"

- 1.- Análisis de variables categóricas y númericas
- 2.- Análisis univariado y bivariado

Vamos a verificar cuantos usuarios hay en cada archivo.

In [55]:
```
1 dailyActivity['Id'].nunique()
```

Out[55]: 33

Observación:

- Este DF cuenta con información de 33 usuarios. El proveedor de los datos fue impreciso al indicar que era una encuesta a 30 usuarios.

Variables númericas

In [56]:
```
1 dailyActivity.describe()
```

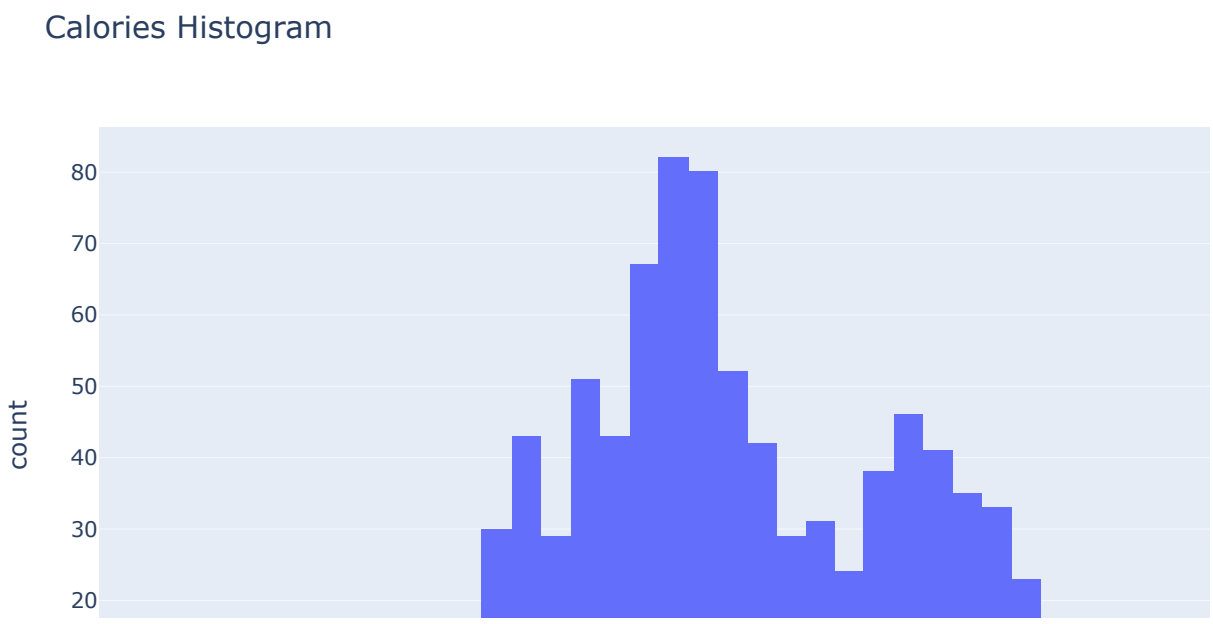
Out[56]:

	Id	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryActiveDistance	I
count	9.400000e+02	940.000000	940.000000	940.000000	940.000000	940.000000	
mean	4.855407e+09	7637.910638	5.489702	5.475351	0.108171	1.502681	
std	2.424805e+09	5087.150742	3.924606	3.907276	0.619897	2.658941	
min	1.503960e+09	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2.320127e+09	3789.750000	2.620000	2.620000	0.000000	0.000000	
50%	4.445115e+09	7405.500000	5.245000	5.245000	0.000000	0.210000	
75%	6.962181e+09	10727.000000	7.712500	7.710000	0.000000	2.052500	
max	8.877689e+09	36019.000000	28.030001	28.030001	4.942142	21.920000	

Observaciones:

- Existen registros de "Calories" igual a 0, se va a investigar un poco más al respecto. Puede ser que el dispositivo se uso por un tiempo muy breve o que no se uso durante todo el día.
- También se observa que el registro maximo de "Calories" es de 4900, esto puede ser gasto calórico de un atleta. Investiguemos un poco más al respecto para evaluar.

In [57]:
```
fig3_hist_calories = px.histogram(dailyActivity,
                                  x='Calories',
                                  labels = {'Calories': 'Calories'},
                                  title = 'Calories Histogram'
                                  #text_auto=True
                                 )
fig3_hist_calories.show()
```



Interpretación:

- Histograma bimodal con sesgo a la derecha.
- Existen observaciones por debajo de las 1000 calorías, que puede reflejar poco uso del dispositivo durante todo el día.
- El 1er pico esta cercano a 2000 calorías, donde se encuentra la mediana. Estas observaciones están relacionadas a las actividades de baja intensidad o actividades del día a día.
- El segundo pico esta orientado hacia las 2800 calorías. Este grupo pertenece a los usuarios con registro de alguna actividad diaria.
- Luego existe una cola que va desde los 3400 hasta 4900 calorías. Este grupo puede ser de usuarios que registren actividades de mayor intensidad o con mayor duración.

Observaciones:

- Según la web de alimentación Natursane se estima que podemos quemar entre 0,9 y 1,02 Kcal por kilo de nuestro peso por hora. Si multiplicas tu peso por 8 horas te sale el resultado de lo que quema tu cuerpo aproximadamente en reposo. Esto no es exacto ya que cada persona tiene una constitución diferente, pero se estima que se queman entre 400-600 Kcal.

- Se estima que en una hora de actividad sentado, por ejemplo leyendo, en el transporte, trabajando…etc., podemos quemar alrededor de 60 Kcal, si multiplicamos 60 por 16 horas del día que no pasamos durmiendo serían 960Kcal.
- Entendiendo que una persona sedentaria quema entre 1600 - 1800 calorías aprox, se puede decir que una persona no puede quemar 0 calorías en el día, por lo que se puede intuir que estos valores esten relacionados al poco tiempo de uso de los dispositivos durante el día.
- Se van a contabilizar los valores inferiores a 1000 calorías para decidir si se eliminan o no del análisis

In [58]:
```
print(f'Tamaño del set antes de eliminar los valores de calorías = 0: {dailyActivity.sha
dailyActivity = dailyActivity[dailyActivity['Calories']>999]
print(f'Tamaño del set después de eliminar los valores de calorías = 0: {dailyActivity.s
```

```
Tamaño del set antes de eliminar los valores de calorías = 0: (940, 18)
Tamaño del set después de eliminar los valores de calorías = 0: (928, 18)
```

Se decide eliminar valores de calorías <= 999, es decir, 12 filas que representarían 1,2% de los datos para el posterior análisis.

In [59]:
```
dailyActivity.describe()
```

Out[59]:

	Id	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryActiveDistance	I
count	9.280000e+02	928.000000	928.000000	928.000000	928.000000	928.000000	
mean	4.847679e+09	7728.580819	5.555356	5.540819	0.109570	1.521088	
std	2.422645e+09	5054.652504	3.905802	3.888408	0.623773	2.671046	
min	1.503960e+09	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2.320127e+09	3966.000000	2.680000	2.680000	0.000000	0.000000	
50%	4.445115e+09	7537.500000	5.305000	5.305000	0.000000	0.225000	
75%	6.962181e+09	10747.000000	7.730000	7.722500	0.000000	2.102500	
max	8.877689e+09	36019.000000	28.030001	28.030001	4.942142	21.920000	

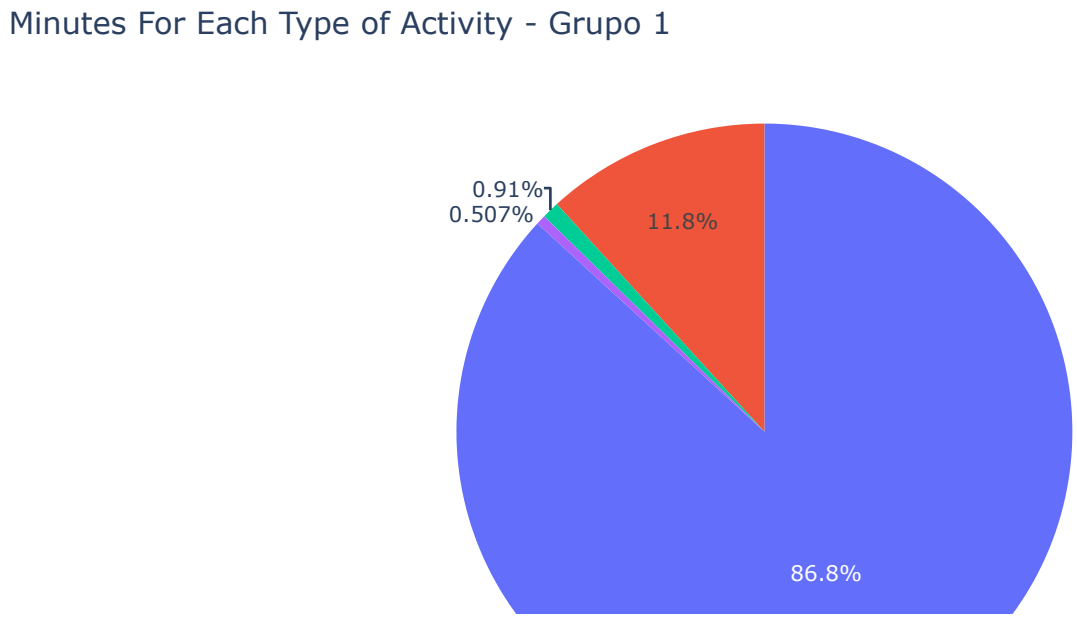
Vamos a segmentar nuestro análisis en 3 grupos de acuerdo al histograma bimodal y a los percentiles estadísticos para la variable "Calories":

- grupo1 : (1000 - 1841) Se podría considentar usuarios sedentarios para este análisis. Se considera valor del percentil 25%.
- grupo2 : (1842-2797) Incluye desde el 1er pico hasta el 2do pico del histograma. Desde percentil 50%-75%. Usuarios que realizan alguna actividad física durante el día.
- grupo3: (>2797) Mayores al percentil 75% que relacionaremos con las actividades de mayor intensidad o mayor duración.

In [60]:
```
grupo1 = dailyActivity[(dailyActivity['Calories']>999) & (dailyActivity['Calories']<=184
grupo2 = dailyActivity[(dailyActivity['Calories']>1842) & (dailyActivity['Calories']<=27
grupo3 = dailyActivity[(dailyActivity['Calories']>2798)]
```

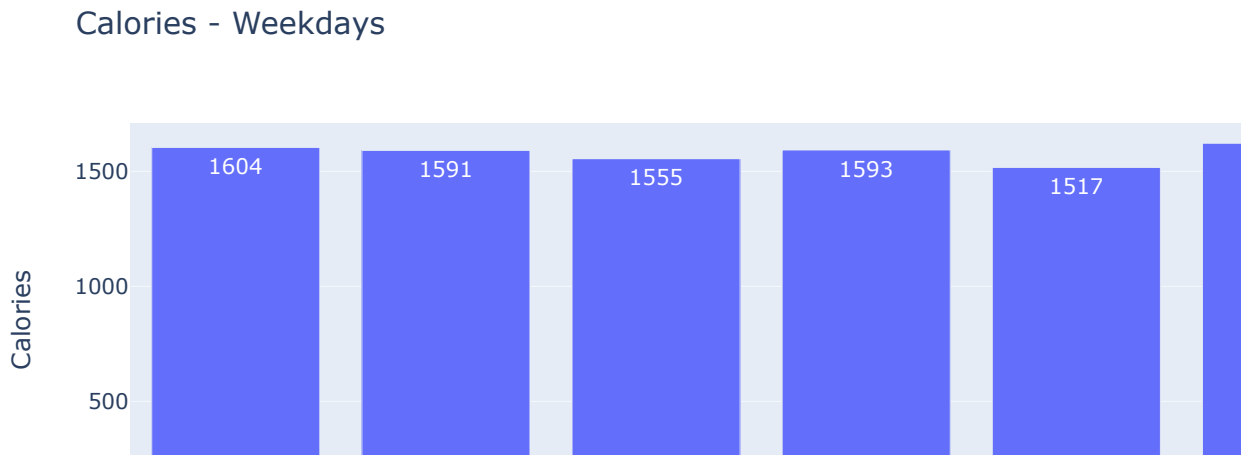
Grupo 1

In [61]:
```
VeryActiveMinutes1 =grupo1['VeryActiveMinutes'].sum()
FairlyActiveMinutes1 =grupo1['FairlyActiveMinutes'].sum()
LightlyActiveMinutes1 =grupo1['LightlyActiveMinutes'].sum()
SedentaryMinutes1 =grupo1['SedentaryMinutes'].sum()

variables = [VeryActiveMinutes1, FairlyActiveMinutes1, LightlyActiveMinutes1, SedentaryM
labels= ['VeryActiveMinutes', 'FairlyActiveMinutes', 'LightlyActiveMinutes', 'SedentaryM

fig4_pie_grupo1 = px.pie(values = variables,
                         names= labels,
                         title = 'Minutes For Each Type of Activity - Grupo 1'
                        )
fig4_pie_grupo1.show()
```



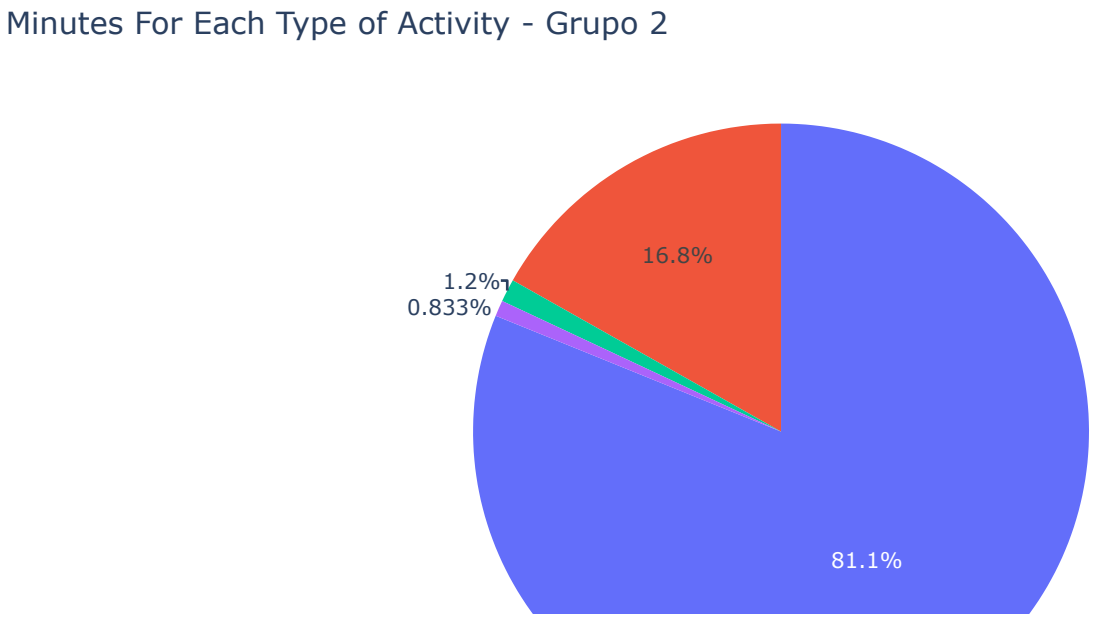
In [62]:

```
dias_semana = round(grupo1.groupby(['Day_week'], as_index=False).mean(), 0)

fig5_bar_grupo1 =px.bar(dias_semana, x ='Day_week', y = 'Calories',
       #color='TotalSteps',
       barmode='group', #Para colocar una barra al lado de lotra
       text ='Calories',
       title='Calories - Weekdays',
       labels = {'Day_week': 'Weekdays', 'Calories': 'Calories'}, height=400,
       hover_name = 'Calories',
       hover_data = {'Day_week': False, 'Calories': True},
        color_discrete_map = {'casual': '#0b5394', 'member': '#3fe2d6'})
fig5_bar_grupo1.show()
```



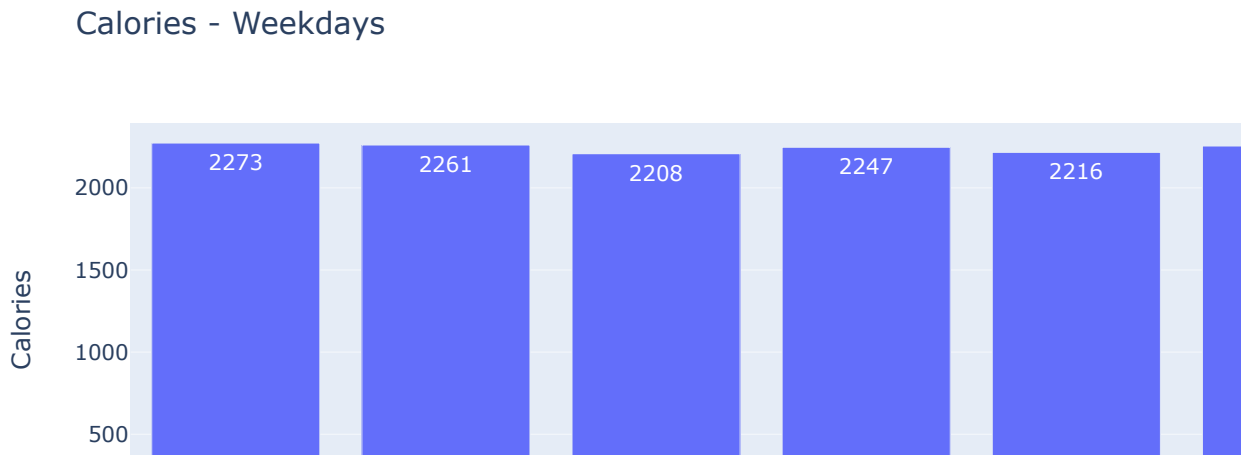
Grupo2

In [63]:
```
VeryActiveMinutes2 =grupo2['VeryActiveMinutes'].sum()
FairlyActiveMinutes2 =grupo2['FairlyActiveMinutes'].sum()
LightlyActiveMinutes2 =grupo2['LightlyActiveMinutes'].sum()
SedentaryMinutes2 =grupo2['SedentaryMinutes'].sum()

variables = [VeryActiveMinutes2, FairlyActiveMinutes2, LightlyActiveMinutes2, SedentaryM
labels= ['VeryActiveMinutes', 'FairlyActiveMinutes', 'LightlyActiveMinutes', 'SedentaryM

fig6_pie_grupo2 = px.pie(values = variables,
                         names= labels,
                         title = 'Minutes For Each Type of Activity - Grupo 2'
                        )
fig6_pie_grupo2.show()
```



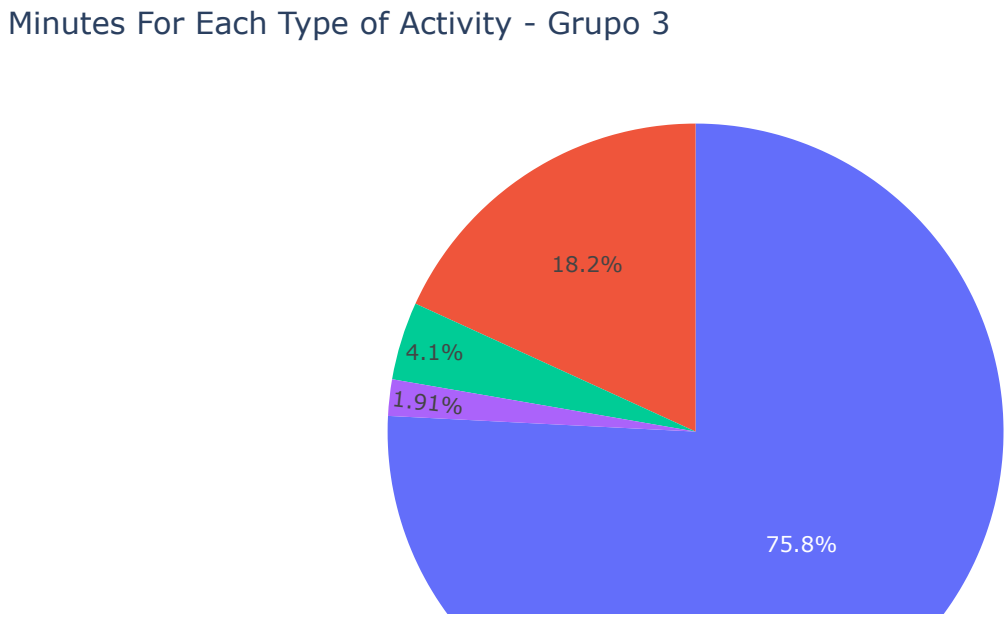
In [64]:

```
dias_semana2 = round(grupo2.groupby(['Day_week'], as_index=False).mean(), 0)

fig7_bar_grupo2 = px.bar(dias_semana2, x ='Day_week', y = 'Calories',
       #color='TotalSteps',
       barmode='group', #Para colocar una barra al lado de lotra
       text ='Calories',
       title='Calories - Weekdays',
       labels = {'Day_week': 'Weekdays', 'Calories': 'Calories'}, height=400,
       hover_name = 'Calories',
       hover_data = {'Day_week': False, 'Calories': True},
        color_discrete_map = {'casual': '#0b5394', 'member': '#3fe2d6'})
fig7_bar_grupo2.show()
```

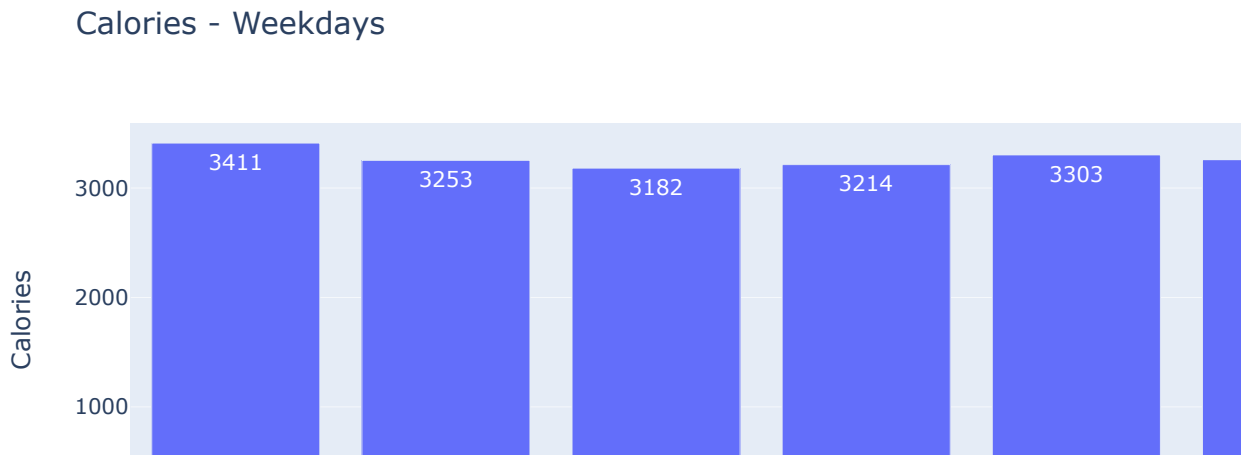


Grupo 3

In [65]:
```
VeryActiveMinutes3 =grupo3['VeryActiveMinutes'].sum()
FairlyActiveMinutes3 =grupo3['FairlyActiveMinutes'].sum()
LightlyActiveMinutes3 =grupo3['LightlyActiveMinutes'].sum()
SedentaryMinutes3 =grupo3['SedentaryMinutes'].sum()

variables = [VeryActiveMinutes3, FairlyActiveMinutes3, LightlyActiveMinutes3, SedentaryM
labels= ['VeryActiveMinutes', 'FairlyActiveMinutes', 'LightlyActiveMinutes', 'SedentaryM

fig8_bpie_grupo3 = px.pie(values = variables,
                          names= labels,
                          title = 'Minutes For Each Type of Activity - Grupo 3'
                         )
fig8_bpie_grupo3.show()
```



In [66]:
```
dias_semana3 = round(grupo3.groupby(['Day_week'], as_index=False).mean(), 0)

fig8_bar_grupo3 =px.bar(dias_semana3, x ='Day_week', y = 'Calories',
       #color='TotalSteps',
       barmode='group', #Para colocar una barra al lado de lotra
       text ='Calories',
       title='Calories - Weekdays',
       labels = {'Day_week': 'Weekdays', 'Calories': 'Calories'}, height=400,
       hover_name = 'Calories',
       hover_data = {'Day_week': False, 'Calories': True},
        color_discrete_map = {'casual': '#0b5394', 'member': '#3fe2d6'})
fig8_bar_grupo3.show()
```

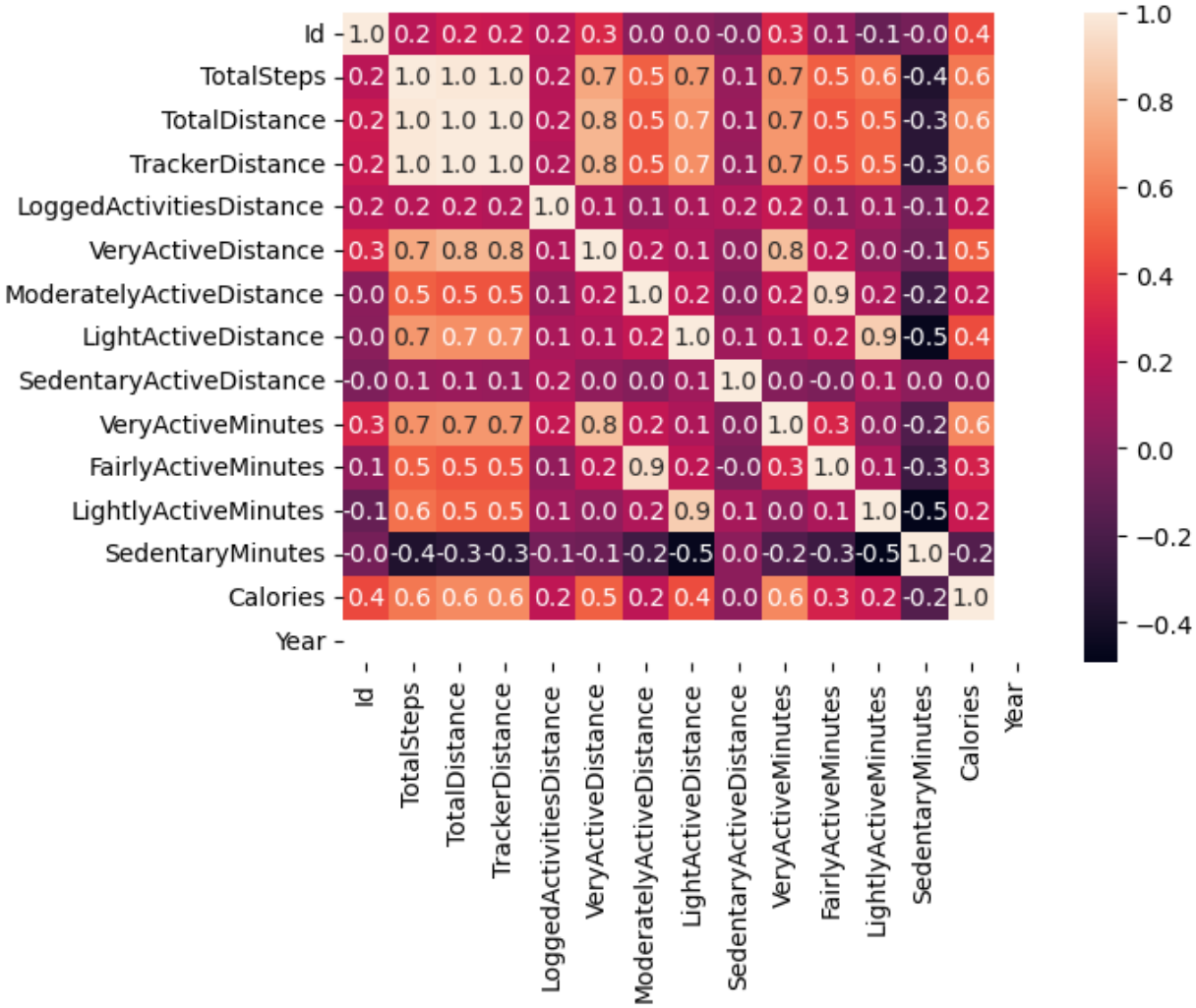


Observaciones:

- El grupo1 tiene un 86.8% de minutos sedentarios, es el grupo con el porcentaje más alto en este aspecto. Mientras que el grupo2 tiene 81.1% y el grupo3 75.8%
- El grupo3 tiene un 18,2% de minutos activos, es el grupo con el porcentaje más alto en este aspecto. Mientras que el grupo2 tiene 16.8% siendo mayor al porcentaje del grupo1 con 11.8%. El mismo comportamiento se refleja para los minutos muy activos y bastante activos.
- El usuario que gasta menor cantidad de calorías tiende a contabilizar mayor cantidad de minutos sedentarios.
- El usuario que gasta mayor cantidad de calorías tiende a contabilizar menor cantidad de minutos sedentarios y mayor cantidad de minutos ligeramente activos, bastante activos y muy activos.
- El usuario usuario que realiza alguna actividad (contabiliza minutos ligeros, activos y/o muy activos) es el que utiliza mayor cantidad de calorías.
- El grupo1 utiliza más cantidad de calorías los días viernes, sabados y domingos. Mientras que los días Jueves es el día que menos calorías utiliza.
- El grupo2 es casi constante en el uso de las calorías todos los días.
- El grupo3 usa mayor cantidada de calorías los días Sabados y domingo. Mientras que los días Martes es el día que usa menos calorías.
- Recordar que el 1,2% de las mediciones (calorías<999) se considera como mediciones erroneas por poco tiempo de uso de los dispositivos.

Evaluemos la correlación en dailyActivity

In [67]:
```
grafico_dailyActivity = dailyActivity.corr()
sns.heatmap(grafico_dailyActivity,annot=True, fmt=".1f" );
```



Interpretación:

- Las variables "TotalDistance" y "TrackerDistance" arrojan los mismos datos por lo que para análisis tomaremos "TotalDistance" y eliminaremos "TrackerDistance".
- Importante correlación positiva entre "totalSteps" y "Total Distance", lo que implica que a medida que aumentan los pasos por los usuarios aumenta la distancia.
- El "totalSteps" y "TotalDistance" estan fuertemente relacionadas con "VeryActiveDistance" y "VeryActiveMinutes", con 0.8 y 0.7 respectivamente. Lo que nos indica que al incrementarse el total de los pasos y el total de la distancia se incrementan los minutos muy activos y la distancia muy activa.
- También se abserva un correlación positiva que nos incia que a medica que se incrementa "totalSteps" se incrementa "Calories".
- "LoggedActivitiesDistance" tiene poca relación con otras variables y será eliminada para el posterior análisis.
- "ModeratelyActiveDistance tiene un fuerte correlacion positiva de 0.9 con "FairlyActiveMinutes", lo que tiene sentido, el mismo comportamiento se observa entre "LightlyActiveMinutes" y "LightActiveDistance".
- La variable "SedentaryMinutes" tiene una correlación negativa con las siguientes "TotalSteps", "TotalDistance", "Moderately Active Distance", "LightActiveDistance", "Very Active Minutes", "FairlyActiveMinutes", "LightlyActiveMinutes", "Calories". Lo que nos indica que a medida que aumentan las

calorías y las actividades independientemente de su intensidad disminuyen los minutos de sedentarismo, este comportamiento tmabién se observo el en análisis de los grupos.

In [68]:
```
dailyActivity = dailyActivity.drop(['TrackerDistance',
       'LoggedActivitiesDistance'], axis=1)
```

Ahora evaluemos relaciones entre "sleepDay" y "dailyActivity"

Empezaremos explorando "sleepDay". Observando sus variables númericas y la cantidad de usuarios.

In [69]:
```
sleepDay.describe()
```

Out[69]:

	Id	TotalSleepRecords	TotalMinutesAsleep	TotalTimeInBed	Year
count	4.100000e+02	410.000000	410.000000	410.000000	410.0
mean	4.994963e+09	1.119512	419.173171	458.482927	2016.0
std	2.060863e+09	0.346636	118.635918	127.455140	0.0
min	1.503960e+09	1.000000	58.000000	61.000000	2016.0
25%	3.977334e+09	1.000000	361.000000	403.750000	2016.0
50%	4.702922e+09	1.000000	432.500000	463.000000	2016.0
75%	6.962181e+09	1.000000	490.000000	526.000000	2016.0
max	8.792010e+09	3.000000	796.000000	961.000000	2016.0

In [70]:
```
sleepDay['Id'].nunique()
```

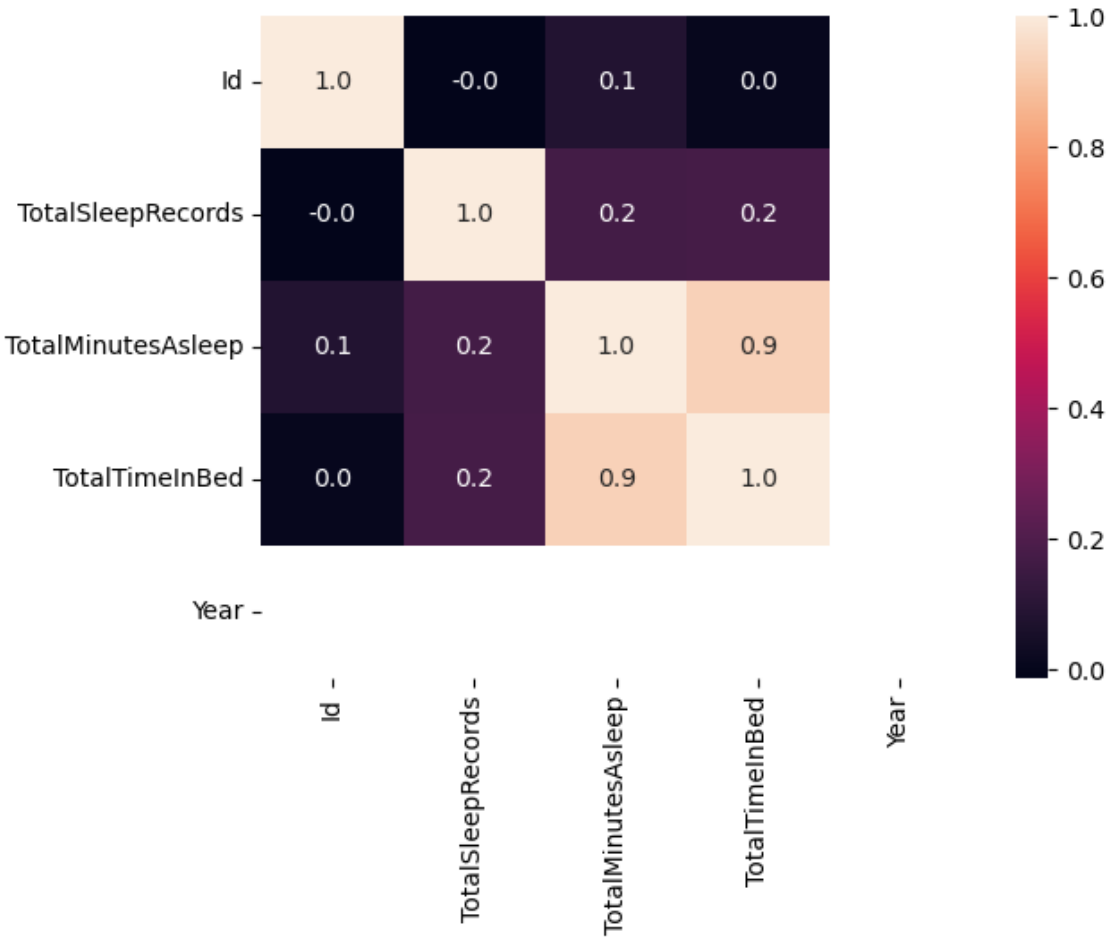
Out[70]: 24

Observación:

- Este DF cuenta con información de 24 usuarios a diferencia del DF "dailyActivity". A pesar de dicha característica será considerado para el análisis.

Visualicemos como están relacionadas las variables de dicho DF:

In [71]:
```
grafico_sleepDay_corr = sleepDay.corr()
sns.heatmap(grafico_sleepDay_corr,annot=True, fmt=".1f" );
```



Interpretación:

- Cuenta con una correlación positiva de 0.9 lo que indica que a medida que aumenta "TotalMinutesAsleep" también se incrementa "TotalTimeInBed". Aunque la correlación no significva causalidad.

Visualicemos la correlación de estas dos variables a continuación:

In [72]:
```
fig10_corr_timesleep_timeinbed = px.scatter(x = sleepDay['TotalMinutesAsleep'],
                          y = sleepDay['TotalTimeInBed'],
                          title = "Total minutes of sleep vs Total time in bed",
                          labels = {'x': 'Total minutes of sleep', 'y':'Total time in be
fig10_corr_timesleep_timeinbed.show()
```

Interpretación:

- El gráfico muestra una fuerte asociación lineal positiva entre las dos variables, por lo que a medida que aumenta el tiempo en la cama se incrementan los minutos de sueño.
- Los usuarios se van a la cama para dormir casi todo el tiempo, sin embargo, puede observarse algunos casos que pasan mas tiempo en la cama sin dormir, esto puede deberse a los fines de semana o minutos para conversar, leer, ver redes sociales antes de dormir o al despertar y quedarse en la cama los días de descanso.

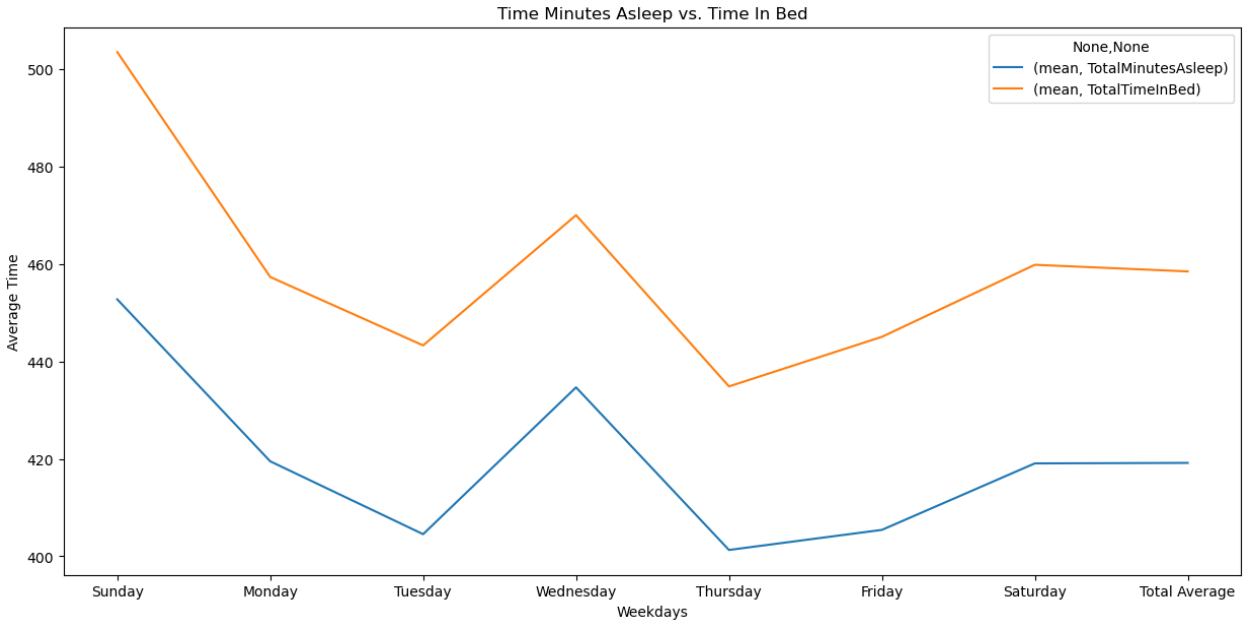
Visualicemos el comportamiento de estas variables por cada día de la semana.

In [73]:
```
fig11_line_timesleep_timeinbed_weekdays= pd.pivot_table(sleepDay,
              index = 'Day_week',
               values = ['TotalMinutesAsleep', 'TotalTimeInBed'],
               aggfunc = ['mean'],
               margins = True,
               margins_name = 'Total Average')
fig11_line_timesleep_timeinbed_weekdays
```

Out[73]:

	mean	
	TotalMinutesAsleep	TotalTimeInBed
Day_week		
Sunday	452.745455	503.509091
Monday	419.500000	457.347826
Tuesday	404.538462	443.292308
Wednesday	434.681818	470.030303
Thursday	401.296875	434.875000
Friday	405.421053	445.052632
Saturday	419.070175	459.842105
Total Average	419.173171	458.482927

In [74]:
```
fig11_line_timesleep_timeinbed_weekdays.plot(kind = 'line',
          figsize = [15,7],
          xlabel = 'Weekdays',
         ylabel= 'Average Time',
         title = 'Time Minutes Asleep vs. Time In Bed',
          legend = True
         );

```



Interpretación:

- El tiempo en la cama y el tiempo de sueño tienen una relación positiva por lo que si una se incrementa la otra variable también y viceversa.

- Los días que los usuarios suelen pasar más tiempo en la cama son los días domingos, tiendo a pensar que es el día de descanso de la mayoría de los usuarios.
- El segundo día que más tiempo pasan los usuarios en la cama son los días miércoles, la mitad de la semana.
- Los días martes y jueves son los días que menos tiempo se quedan en la cama los usuarios, lo que llama mi atención es que corresponde a un día antes y un día después del 2do día que más duermen (miércoles).
- El tiempo promedio de sueño sería 419.17 minutos(7 horas aprox.) y el tiempo promedio en cama 458.48 (7,6 horas aprox.)
- Los usuarios pasan en promedio 39,31 minmutos en la cama sin dormir, siendo el día domingo el día que más tiempo pasan en la cama con 51 minutos.

Vamos a unir "sleepDay" con "dailyActivity" para hacer un análisis exploratorio

In [75]:
```
sleepDay_dailyActivity = pd.merge(dailyActivity, sleepDay, on =['Id', 'Date', 'Year', 'M
sleepDay_dailyActivity.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 932 entries, 0 to 931
Data columns (total 19 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Id                        932 non-null    int64
 1   Date                      932 non-null    datetime64[ns]
 2   TotalSteps                928 non-null    float64
 3   TotalDistance             928 non-null    float64
 4   VeryActiveDistance        928 non-null    float64
 5   ModeratelyActiveDistance  928 non-null    float64
 6   LightActiveDistance       928 non-null    float64
 7   SedentaryActiveDistance   928 non-null    float64
 8   VeryActiveMinutes         928 non-null    float64
 9   FairlyActiveMinutes       928 non-null    float64
 10  LightlyActiveMinutes      928 non-null    float64
 11  SedentaryMinutes          928 non-null    float64
 12  Calories                  928 non-null    float64
 13  Year                      932 non-null    int64
 14  Month                     932 non-null    category
 15  Day_week                  932 non-null    category
 16  TotalSleepRecords         410 non-null    float64
 17  TotalMinutesAsleep        410 non-null    float64
 18  TotalTimeInBed            410 non-null    float64
dtypes: category(2), datetime64[ns](1), float64(14), int64(2)
memory usage: 133.6 KB
```

Se observan NA que se generaron por la diferencia de "Id" entre los DF, por lo que vamos a realizar lo siguiente:

- Eliminar columnas poco relevantes para el objetivo empresarial de este análisis.
- Calcular la mediana para las variables númericas con NA
- Calcular la moda para las variables categóricas con NA
- Imputar la mediana y la moda.

In [76]:
```
# Eliminamos las columnas poco relevantes con drop()
sleepDay_dailyActivity = sleepDay_dailyActivity.drop (['TotalSleepRecords', 'VeryActiveD
       'ModeratelyActiveDistance', 'LightActiveDistance'], axis =1)
```

In [77]:
```
1 sleepDay_dailyActivity['TotalMinutesAsleep'].median()
```

Out[77]: 432.5

In [78]:
```
1 sleepDay_dailyActivity['TotalTimeInBed'].median()
```

Out[78]: 463.0

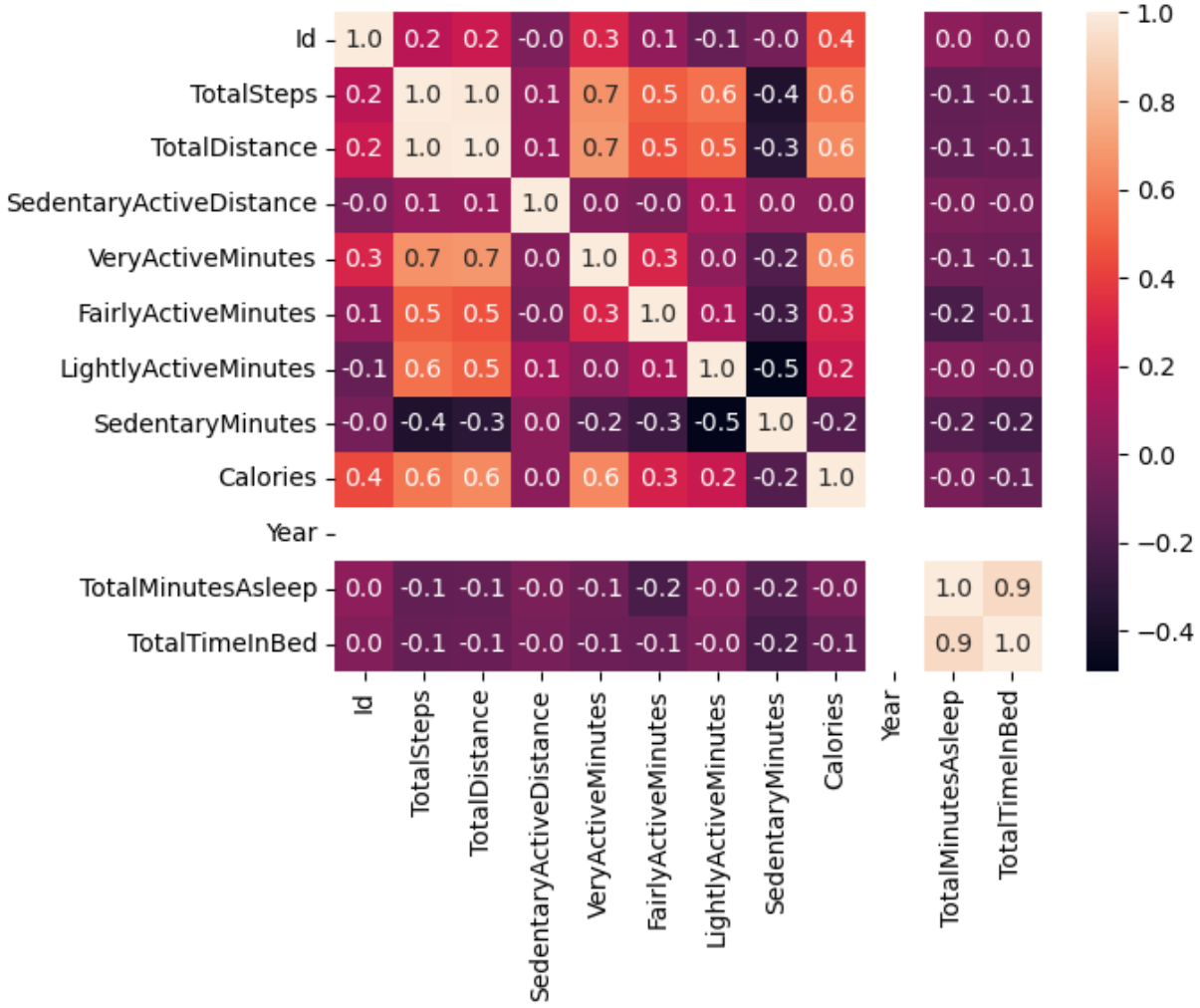
In [79]:
```
1 sleepDay_dailyActivity['Day_week'].mode()
```

Out[79]:
```
0    Tuesday
Name: Day_week, dtype: category
Categories (7, object): ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday']
```

In [80]:
```
1 #Imputamos las medianas y la moda:
2 sleepDay_dailyActivity['TotalMinutesAsleep'] = sleepDay_dailyActivity['TotalMinutesAslee
3 sleepDay_dailyActivity['TotalTimeInBed']=sleepDay_dailyActivity['TotalTimeInBed'].fillna
4 sleepDay_dailyActivity['Day_week'] = sleepDay_dailyActivity['Day_week'].fillna('Wednesda
```

Visualicemos la correlación del DF creado:

In [81]:
```
1 grafico_sleepDay_dailyActivity_corr = sleepDay_dailyActivity.corr()
2 sns.heatmap(grafico_sleepDay_dailyActivity_corr,annot=True, fmt=".1f" );
```



Observación:

- Los minutos de sueño mantienen una correlación negativa con los minutos de sedentarismo.
- No se visualiza relación positiva entre las variables de ambos DF y las variables se analizarán por separado, por lo que seguiremos con el análisis de DF hourlyCalories.

Evaluemos ahora "hourlyCalories" de manera individual

In [82]:
```
1 hourlyCalories.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22099 entries, 0 to 22098
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Id        22099 non-null  int64
 1   Datehour  22099 non-null  datetime64[ns]
 2   Calories  22099 non-null  int64
 3   Year      22099 non-null  int64
 4   Month     22099 non-null  category
 5   Day_week  22099 non-null  category
 6   Hour      22099 non-null  int64
dtypes: category(2), datetime64[ns](1), int64(4)
memory usage: 1.1 MB
```

In [83]:
```
1 hourlyCalories.describe()
```

Out[83]:

	Id	Calories	Year	Hour
count	2.209900e+04	22099.000000	22099.0	22099.000000
mean	4.848235e+09	97.386760	2016.0	11.415765
std	2.422500e+09	60.702622	0.0	6.915140
min	1.503960e+09	42.000000	2016.0	0.000000
25%	2.320127e+09	63.000000	2016.0	5.000000
50%	4.445115e+09	83.000000	2016.0	11.000000
75%	6.962181e+09	108.000000	2016.0	17.000000
max	8.877689e+09	948.000000	2016.0	23.000000

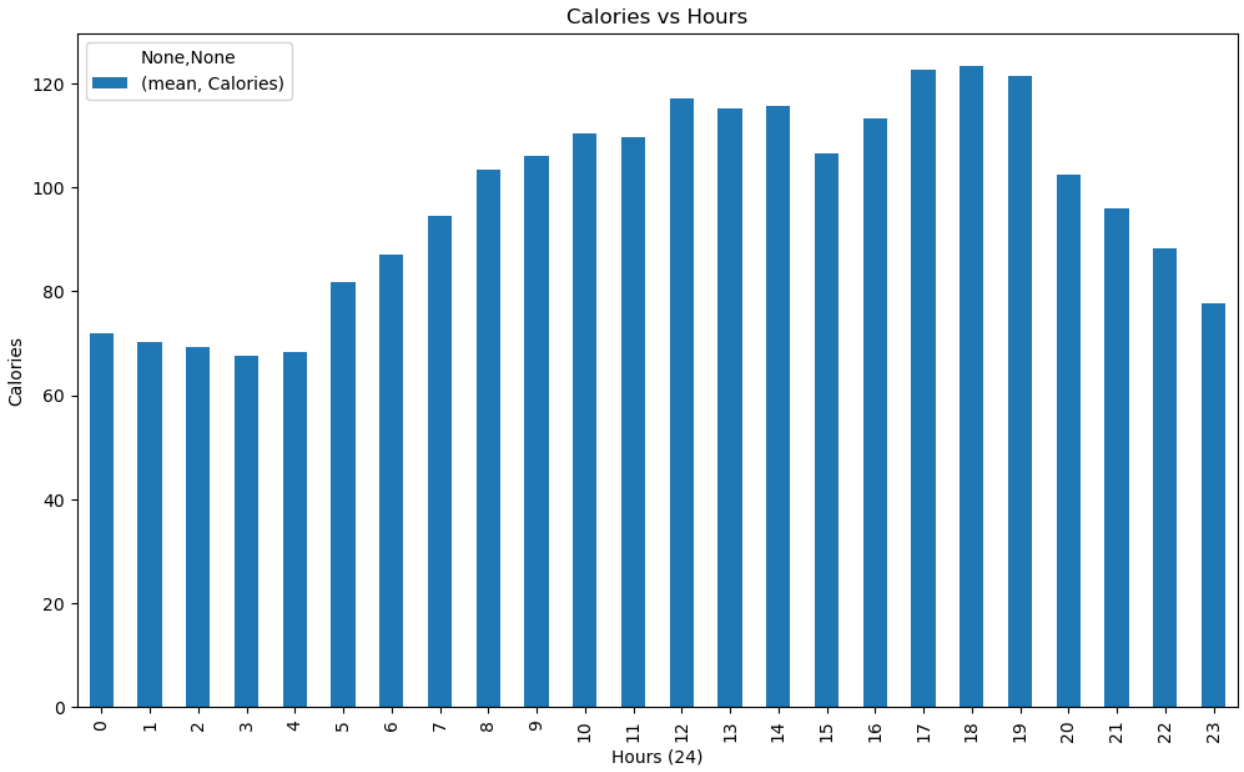
Visualicemos el gasto calórico por horas y días

In [84]:

```
fig12_bar_calories_hours = pd.pivot_table(hourlyCalories,
                                          index=['Hour'],
                                          values ='Calories',
                                          aggfunc = ['mean'],
                                          #margins = True,
                                           #margins_name = 'Average Calories vs Hours'
                                          )
fig12_bar_calories_hours.plot(kind = 'bar',
          figsize = [12,7],
          xlabel = 'Hours (24)',
         ylabel= 'Calories',
         title = 'Calories vs Hours',
          legend = True, #'Calories', 'Hour', title:'MEs'),
          xticks = range(24)
        #legend('Calories':'Calories', 'Hour':'Hours' title ='MEs')
                          );

fig12_bar_calories_hours
```

Out[84]:

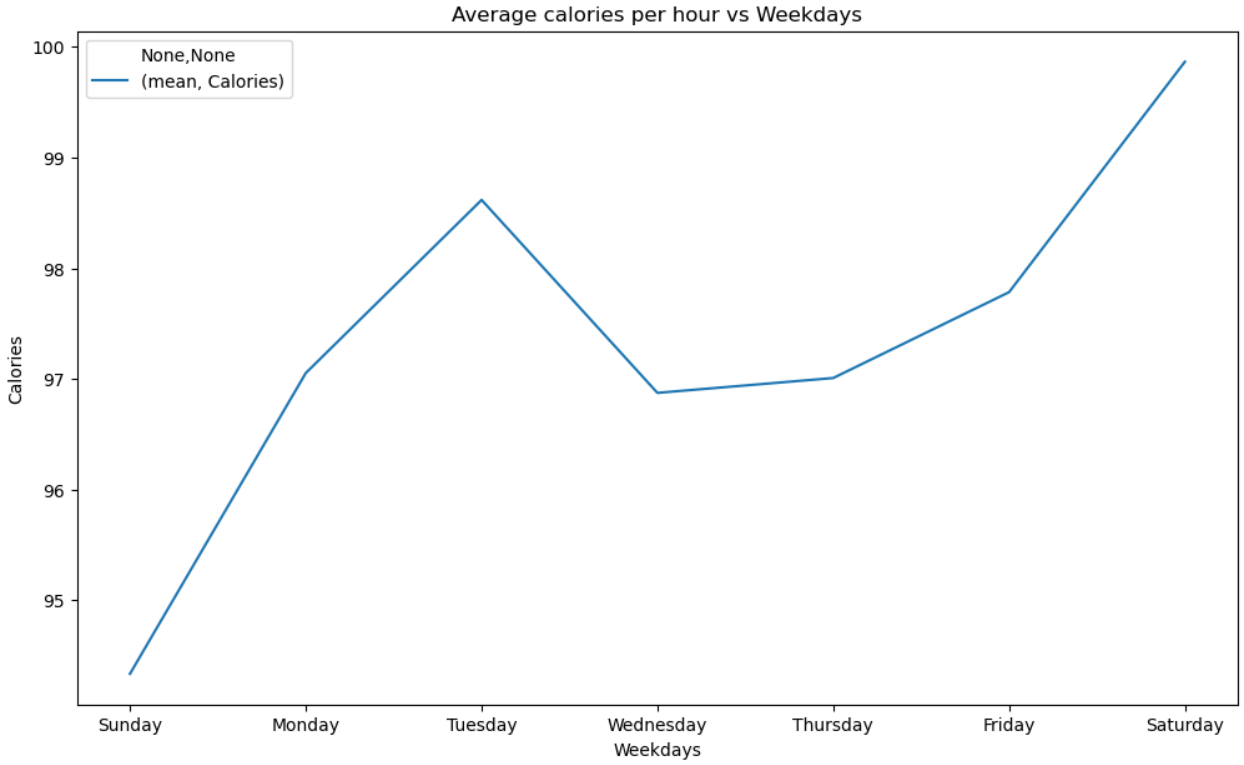
	mean
	Calories
Hour	
0	71.805139
1	70.165059
2	69.186495
3	67.538049
4	68.261803
5	81.708155
6	86.996778
7	94.477981
8	103.337272
9	106.142857
10	110.460710
11	109.806904
12	117.197397
13	115.309446
14	115.732899
15	106.637158
16	113.327453
17	122.752759
18	123.492274
19	121.484547
20	102.357616
21	96.056354
22	88.265487
23	77.593577


Calories vs Hours
None,None
(mean, Calories)
Calories
0
20
40
60
80
100
120
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Hours (24)

In [85]:
```
fig13_line_calories_weekdays = pd.pivot_table(hourlyCalories,
                                      index=['Day_week'],
                                      values ='Calories',
                                      aggfunc = ['mean'],
                                      #margins = True,
                                       #margins_name = 'Average Calories vs Weekdays'
                                      )
fig13_line_calories_weekdays.plot(kind = 'line',
          figsize = [12,7],
          xlabel = 'Weekdays',
         ylabel= 'Calories',
         title = 'Average calories per hour vs Weekdays',
          legend = True
                     );
fig13_line_calories_weekdays
```

Out[85]:

	mean
	Calories
Day_week	
Sunday	94.335981
Monday	97.053478
Tuesday	98.617500
Wednesday	96.874260
Thursday	97.008529
Friday	97.784117
Saturday	99.865866



Interpretación:

- Se observa un incremento en el gasto de calorías desde las 5 de la mañana.
- El mayor gasto se genera desde las 17 horas hasta las 19 horas. Esto podría ser por la salida del trabajo y el retorno al hogar.
- Entre las 12 horas y las 14 horas se genera el segundo período con mayor gasto calorico, se tiende a pensar que corresponde a la hora del almuerzo.
- Entre los rangos más activos ocurre una disminución de consumo calórico, esto ocurre a las 15 horas.
- Desde las 20 horas existe una tendencia de bajo consumo calórico hasta las 5 horas que empieza a aumentar.
- Observamos que lod días Domingos y Lunes son los días con menor gasto calórico.
- El día martes es el día de mayor consumo de calorías y sigue bajando respectivamente hasta el día sabado.

Evaluemos ahora "hourlySteps" de manera individual

In [86]:
```
1 hourlySteps.head()
```

Out[86]:

	Id	Datehour	StepTotal	Year	Month	Day_week	Hour
0	1503960366	2016-04-12 00:00:00	373	2016	April	Tuesday	0
1	1503960366	2016-04-12 01:00:00	160	2016	April	Tuesday	1
2	1503960366	2016-04-12 02:00:00	151	2016	April	Tuesday	2
3	1503960366	2016-04-12 03:00:00	0	2016	April	Tuesday	3
4	1503960366	2016-04-12 04:00:00	0	2016	April	Tuesday	4

In [87]:
```
1 hourlySteps.describe()
```

Out[87]:

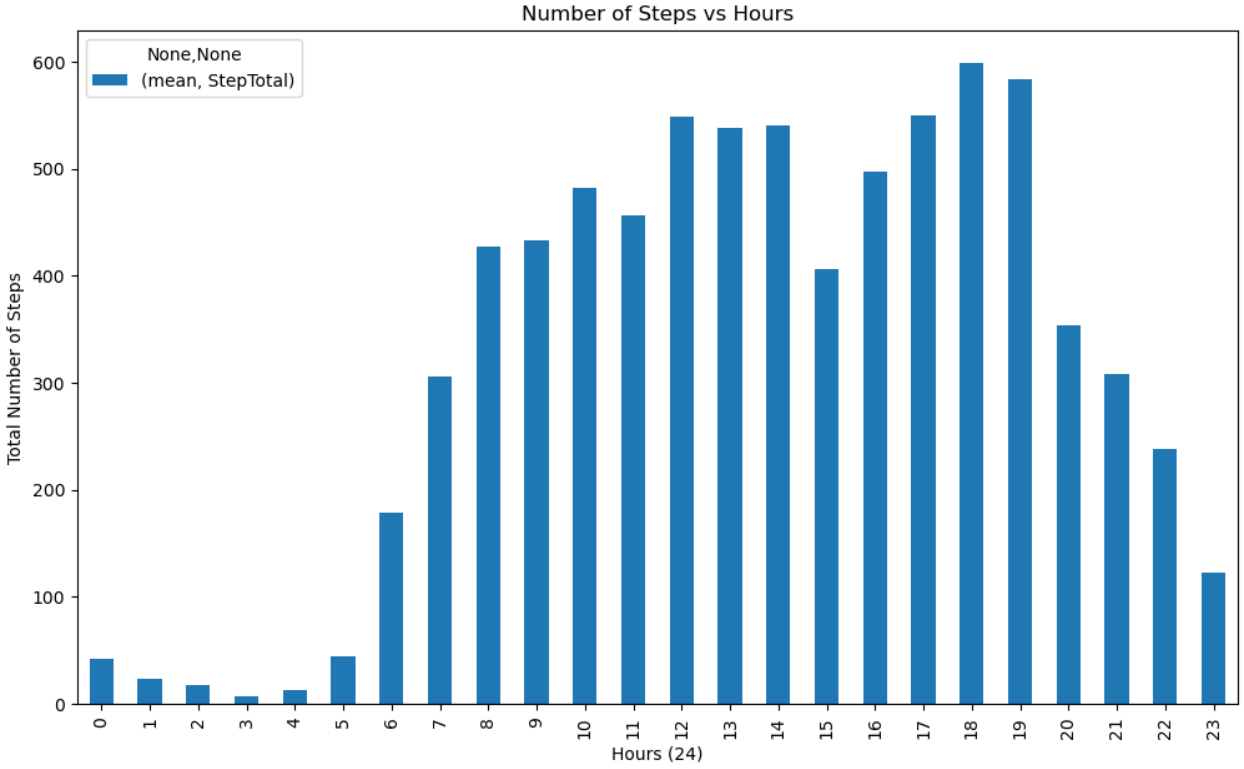
	Id	StepTotal	Year	Hour
count	2.209900e+04	22099.000000	22099.0	22099.000000
mean	4.848235e+09	320.166342	2016.0	11.415765
std	2.422500e+09	690.384228	0.0	6.915140
min	1.503960e+09	0.000000	2016.0	0.000000
25%	2.320127e+09	0.000000	2016.0	5.000000
50%	4.445115e+09	40.000000	2016.0	11.000000
75%	6.962181e+09	357.000000	2016.0	17.000000
max	8.877689e+09	10554.000000	2016.0	23.000000

In [88]:

```
fig14_bar_steptotal_hours = pd.pivot_table(hourlySteps,
                                           index=['Hour'],
                                           values ='StepTotal',
                                           aggfunc = ['mean'],
                                           #margins = True,
                                            #margins_name = 'Average Calories vs Hours'
                                           )

fig14_bar_steptotal_hours.plot(kind = 'bar',
          figsize = [12,7],
          xlabel = 'Hours (24)',
         ylabel= 'Total Number of Steps',
         title = 'Number of Steps vs Hours',
          legend = True,xticks = range(24)
                     );
fig14_bar_steptotal_hours
```

Out[88]:

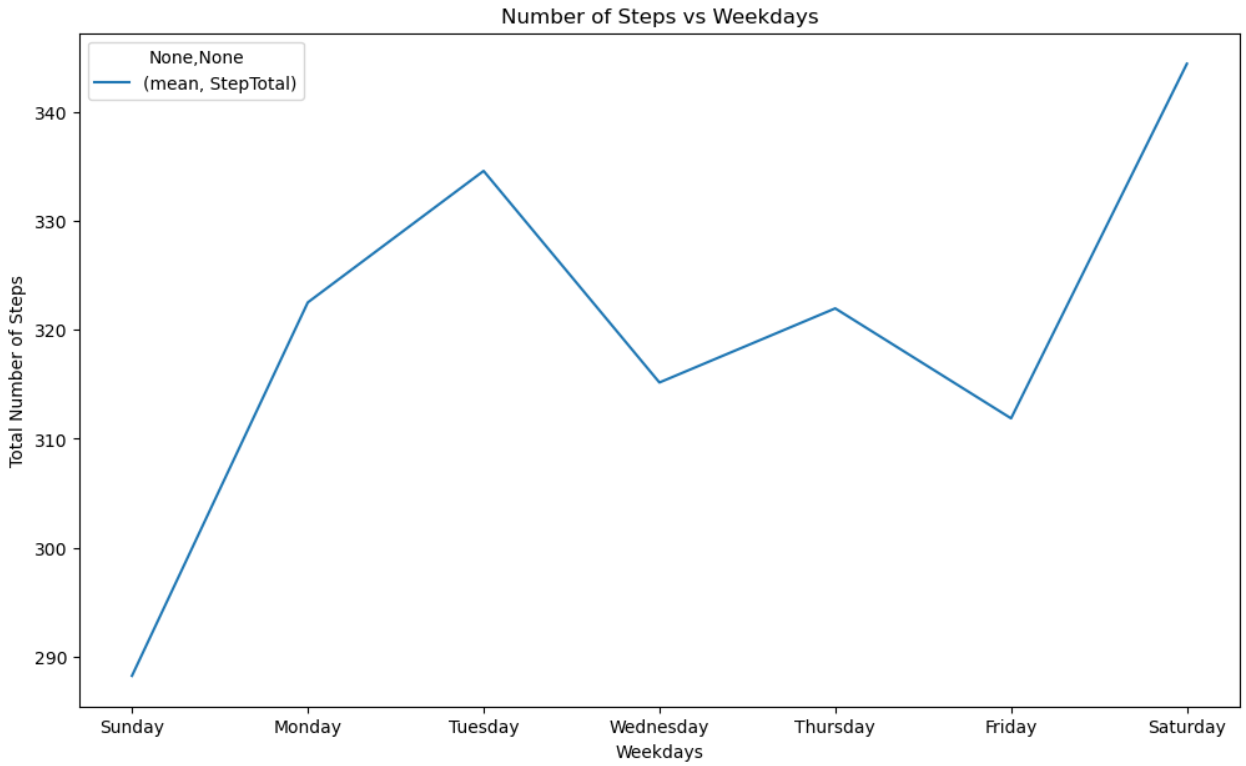
	mean
	StepTotal
Hour	
0	42.188437
1	23.102894
2	17.110397
3	6.426581
4	12.699571
5	43.869099
6	178.508056
7	306.049409
8	427.544576
9	433.301826
10	481.665231
11	456.886731
12	548.642082
13	537.698154
14	540.513572
15	406.319126
16	496.845645
17	550.232892
18	599.169978
19	583.390728
20	353.905077
21	308.138122
22	237.987832
23	122.132890


Number of Steps vs Hours
None,None
(mean, StepTotal)
Total Number of Steps
0
100
200
300
400
500
600
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Hours (24)

In [89]:
```
imagen_hourlySteps_weekdays = pd.pivot_table(hourlySteps,
                                     index=['Day_week'],
                                     values ='StepTotal',
                                     aggfunc = ['mean'],
                                     #margins = True,
                                      #margins_name = 'Average Calories vs Hours'
                                     )

imagen_hourlySteps_weekdays.plot(kind = 'line',
          figsize = [12,7],
          xlabel = 'Weekdays',
         ylabel= 'Total Number of Steps',
         title = 'Number of Steps vs Weekdays',
          legend = True,
                      );

imagen_hourlySteps_weekdays
```

Out[89]:

	mean
	StepTotal
Day_week	
Sunday	288.256215
Monday	322.503321
Tuesday	334.564444
Wednesday	315.160417
Thursday	321.956442
Friday	311.872206
Saturday	344.395883



Interpretación:

- El mayor número de pasos se genera entre las 18 horas hasta las 19 horas. Esto podría ser por la salida del trabajo y el retorno al hogar.
- Entre las 12 horas y las 14 horas se genera el segundo período con mayor número de pasos, se tiende a pensar que corresponde a la hora del almuerzo.
- Entre los rangos más activos ocurre una disminución de pasos, esto ocurre cerca de las 15 horas.
- Desde las 20 horas existe una tendencia de bajada hasta las 3 horas que empieza a subir nuevamente.
- Observamos que los días Domingos son los días con menor número de pasos.
- Los días sabados son los días con mayor número de pasos.
- El día martes es el segundo día con mayor número de pasos.

Ahora evaluemos la unión de hourlySteps_hourlyCalories

In [90]:
```
hourlySteps_hourlyCalories = pd.merge(hourlySteps, hourlyCalories, on =['Id','Hour', 'Da
hourlySteps_hourlyCalories.head()
```

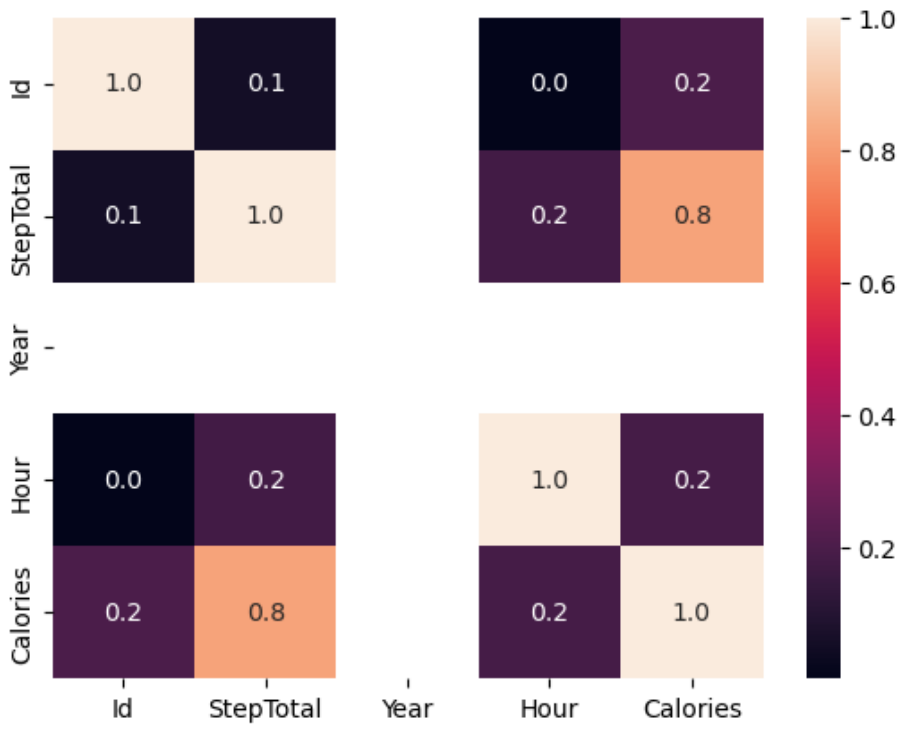
Out[90]:

	Id	Datehour	StepTotal	Year	Month	Day_week	Hour	Calories
0	1503960366	2016-04-12 00:00:00	373	2016	April	Tuesday	0	81
1	1503960366	2016-04-12 01:00:00	160	2016	April	Tuesday	1	61
2	1503960366	2016-04-12 02:00:00	151	2016	April	Tuesday	2	59
3	1503960366	2016-04-12 03:00:00	0	2016	April	Tuesday	3	47
4	1503960366	2016-04-12 04:00:00	0	2016	April	Tuesday	4	48

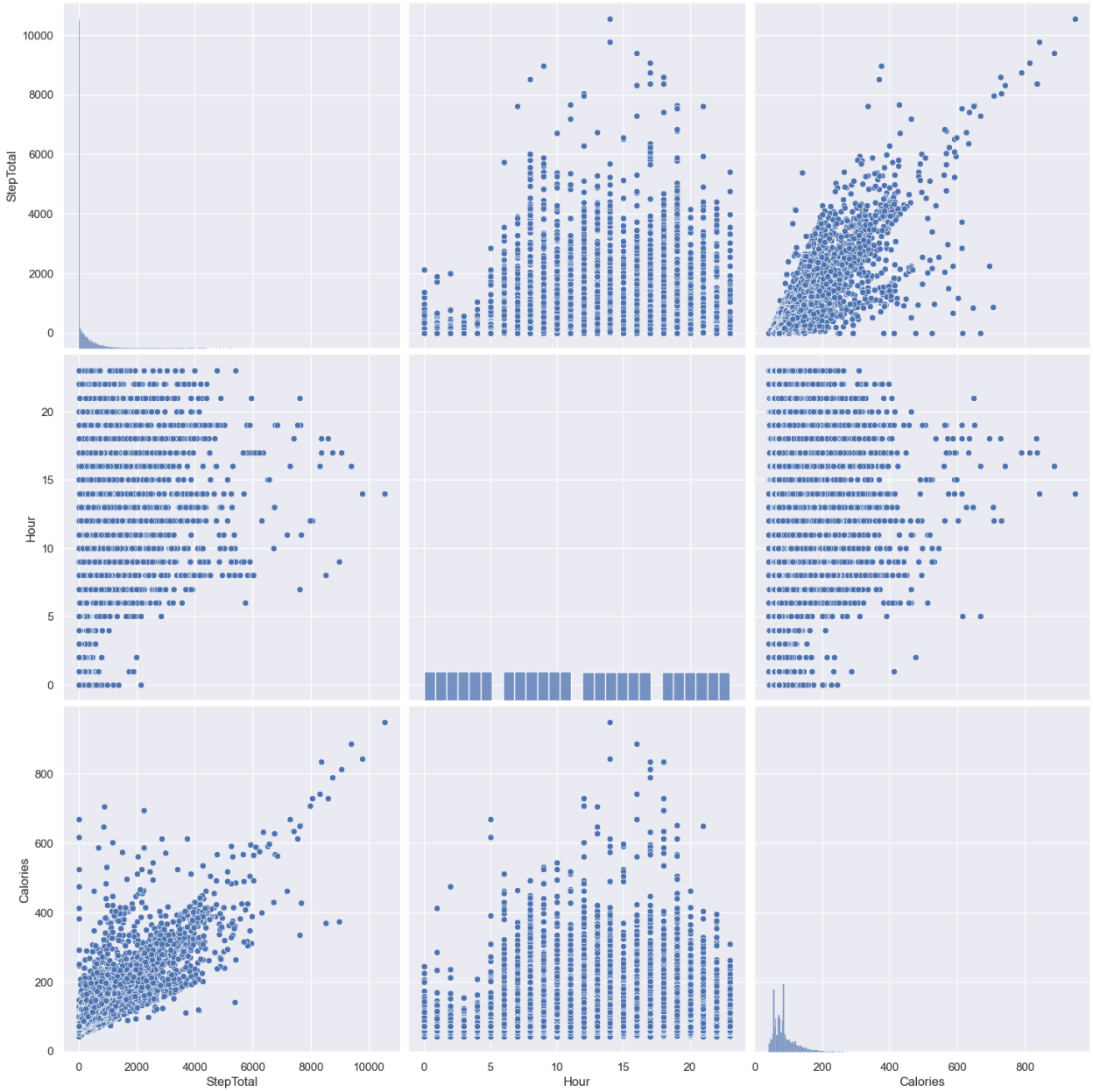
In [91]:
```
hourlySteps_hourlyCalories.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22099 entries, 0 to 22098
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Id         22099 non-null  int64
 1   Datehour   22099 non-null  datetime64[ns]
 2   StepTotal  22099 non-null  int64
 3   Year       22099 non-null  int64
 4   Month      22099 non-null  category
 5   Day_week   22099 non-null  category
 6   Hour       22099 non-null  int64
 7   Calories   22099 non-null  int64
dtypes: category(2), datetime64[ns](1), int64(5)
memory usage: 1.2 MB
```

In [92]:
```
#Evaluemos la correlación entre sus variables

grafico_hourlySteps_hourlyCalories_corr = hourlySteps_hourlyCalories.corr()
sns.heatmap(grafico_hourlySteps_hourlyCalories_corr,annot=True, fmt=".1f" );
```



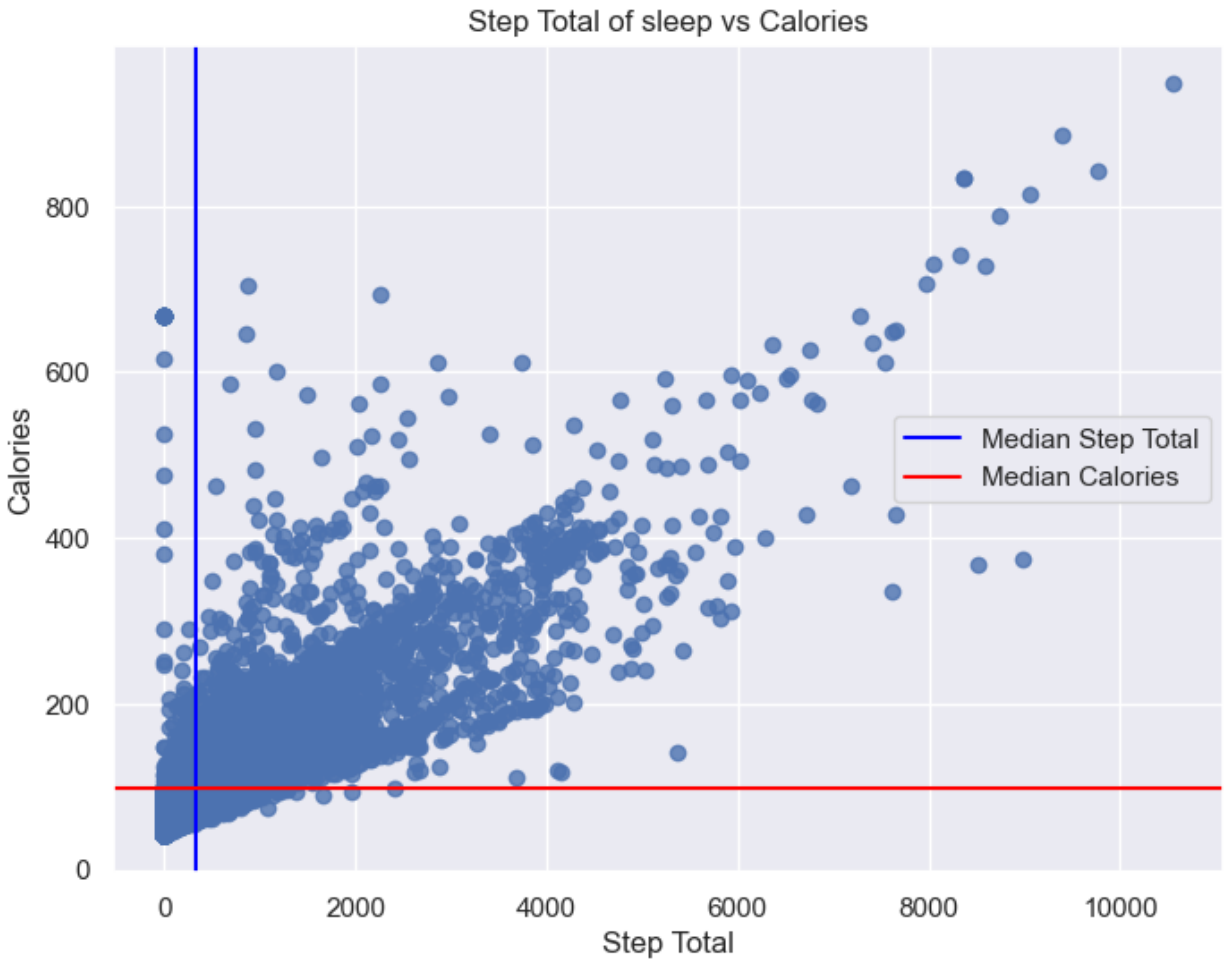
In [93]:
```
#Evaluemos la correlación entre sus variables

sns.set()
cols = ['StepTotal', 'Day_week', 'Hour',
       'Calories']
sns.pairplot(hourlySteps_hourlyCalories[cols], height = 5)
plt.show();
```



In [94]:

```
plt.figure(figsize=(8,6)) # specify size of the chart
plt.scatter(hourlySteps_hourlyCalories['StepTotal'], hourlySteps_hourlyCalories['Calorie
            alpha = 0.8,
            cmap = "Spectral")

mean_StepTotal = hourlySteps_hourlyCalories['StepTotal'].mean()
print(f"Media de pasos = {mean_StepTotal}")
mean_Calories = hourlySteps_hourlyCalories['Calories'].mean()
print(f"Media de pasos = {mean_Calories}")

plt.axvline(mean_StepTotal, color = "Blue", label = "Median Step Total")
plt.axhline(mean_Calories, color = "Red", label = "Median Calories")
plt.xlabel("Step Total")
plt.ylabel("Calories")
plt.title("Step Total of sleep vs Calories")
plt.legend()
plt.grid(True)
```

```
Media de pasos = 320.16634236843294
Media de pasos = 97.38675958188153
```



Interpretación:

- Se visualiza una correlación lineal positiva fuerte de 0.9 entre la cantidad total de pasos y las calorías, lo que significa que a medida que aumenta la cantidad total de pasos también aumentan las calorías.

5.- CONCLUSIONES

- Los usuarios que realizan actividades con mayor gasto calórico por la intensidad de sus actividades reflejan menor cantidad de munutos sedentarios.
- Los usuarios que consumen menor cantidad de calorías tiende a contabilizar mayor cantidad de minutos sedentarios.
- El grupo más sedentarios (consumo de calorías hasta 1841 diarias) utiliza mayor cantidad de calorías los días viernes, sábados y domingos. Mientras que los días Jueves es el día que menos calorías utiliza.
- El grupo intermedio (consumo de calorías entre 1842-2797 diarias) es el grupo más constante, es decir, varía poco la cantidad de calorías por día.
- El grupo más activo (consumo de calorías mayor a 2797 diarias) usa mayor cantidad de calorías los días Sábados y Domingo. Mientras que los días Martes es el día que gasta menos calorías.
- A medida que aumenta el tiempo en la cama aumenta el tiempo de sueño de los usuarios.
- Los días en que los usuarios suelen pasar más tiempo en la cama son los días domingos.
- El segundo día que más tiempo pasan los usuarios en la cama son los días miércoles, la mitad de la semana.
- Los días martes y jueves son los días que menos tiempo se quedan en la cama, lo que llama mi atención es que corresponde a un día antes y un día después del 2do día que más duermen (miércoles).
- El tiempo promedio de sueño sería de 419 minutos(7 horas aprox.) y el tiempo promedio en cama 458 (7,6 horas aprox.).
- Los usuarios pasan en promedio 39,31 minutos en la cama sin dormir, siendo el día domingo el día que más tiempo pasan en la cama con 51 minutos.
- Se observa un incremento en el gasto de calorías desde las 5 de la mañana, por lo que tiendo a pesar que a esa hora inicai la jornada de las actividades diarias.
- El mayor gasto se genera desde las 17 horas hasta las 19 horas. Esto podría ser por la salida del trabajo y el retorno al hogar.
- Entre las 12 horas y las 14 horas se genera el segundo período con mayor gasto calórico, se tiende a pensar que corresponde a la hora del almuerzo.
- Entre los rangos más activos ocurre una disminución de consumo calórico, esto ocurre a las 15 horas.
- Desde las 20 horas existe una tendencia de baja en el consumo calórico hasta las 5 horas que empieza a aumentar nuevamente.
- Observamos que los días Domingos y Lunes son los días con menor gasto calórico.
- El día sabado es el día de mayor consumo de calorías seguido del día martes.
- Existe un fuerte correlación entre el total de los pasos y el consumo de calorías, es decir, a medida que aumenta el número de pasos también aumentan las calorías.
- El número de pasos en promedio es de 7637.91 diarios se encuentra por debajo de 10.000 diarios que es lo recomendado.

6.- RECOMENDACIONES

Después de realizar un análisis detallado se presentan las siguientes recomendaciones de alto valor a considerar para la campaña de marketing de la empresa Bellabeat:

- Enviar como estrategía de las campañas de marketing los beneficios de utilizar los productos de Bellabeat alrededor de las 15:00 horas ya que es el período más tranquilo del día y las personas están más propensas a leer y estar revisando las redes sociales.
- Enviar mensajes acerca de los beneficios de moverse y de estar hidratados poco tiempo antes del inicio de los períodos más activos, es decir, a antes de las 12 y a las 17 horas.
- Incentivar a los usuarios por todos los canales a realizar actividad física durante todos los días, ya que observamos que tanto el grupo sedentarios como los que realizan alguna actividad intensa prefieren moverse más los fines de semana.

- Enviar recordatorios diarios del conteo de pasos y de las calorías para incentivar a los usuarios a moverse.
- Enviar felicitaciones a los usuarios cuando cumplen con las metas de pasos diarios y entrenamientos diarios. También enviar felicitaciones por los días consecutivos de cumplimiento de metas.
- Enviar información acerca de la importancia de la recolección de datos para la mejora de los dispositivos, incentivar a los usuarios a entregar sus estadísticas de actividad, sueño y monitoreo en general. Pueden realizar énfasis en que se tomarán los datos de manera anónima para su respectivo análisis y así lograr que las personas se sientan más confiadas en compartir sus datos. Tambiéin pueden ofrecer el uso de algún producto gratis a modo de prueba.
- Ofrecer el beneficio de metas y consumo de calorías personalizados y acceso a información personalizada si se comparten los datos con la app.
- El día martes es el segundo día más activo y uno de los días en que menos duermen los usuarios, por lo que se recomienda que ese día se publiquen y/o envíen información relacionada a tecnicas de respiración, meditación o cualquier alternativa para mejorar la calidad y tiempo de sueño.
- Ofrecer beneficios y/o dedscuentos para las personas que realizan actividades los fines de semana, ya que estos días la moyoría quema más calorías y podrían convertirse en clientes.
- También enviar promociones los días lunes que es un día propenso a empezar a realizar actividades físicas.
- Enviar informes períodicos acerca de la actividad de cada usuario, para mantenerlos motivados.
- Promover el uso en modo de prueba gratis junto con información relacionada a los beneficios de moverse los días Lunes.
- Enviar promociones a partir de las 20 horas que es cuando baja el consumo de calorías y es otro período con menor moviento, las personas empiezan a preparase para dormir y pasan minutos en la cama, por lo que están propensas a recibir esta información.

Ing. Norwys León