In [2]:
```
# Import of libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
```

In [3]:
```
df = pd.read_csv('dailyActivity_smartwatch.csv')
df.head()
```

Out[3]:

	Id	ActivityDate	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryActiveDistance	ModeratelyActiveDistance	LightActiveDista
0	1503960366	4/12/2016	13162	8.50	8.50	0.0	1.88	0.55	
1	1503960366	4/13/2016	10735	6.97	6.97	0.0	1.57	0.69	
2	1503960366	4/14/2016	10460	6.74	6.74	0.0	2.44	0.40	
3	1503960366	4/15/2016	9762	6.28	6.28	0.0	2.14	1.26	
4	1503960366	4/16/2016	12669	8.16	8.16	0.0	2.71	0.41	

In [4]:
```
#checking null values
df.isnull().sum()
```

Out[4]:
```
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
```

There is no null values in the dataset

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 940 entries, 0 to 939
Data columns (total 15 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Id                         940 non-null    int64
 1   ActivityDate               940 non-null    object
 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
dtypes: float64(7), int64(7), object(1)
memory usage: 110.3+ KB
```

From the information, we can see that the 'ActivityDate' is a object datatype. We need to change it into date type for further analysis.

In [6]:
```
# Convert ActivityDate to date data type
df['ActivityDate'] = pd.to_datetime(df['ActivityDate'], format = '%m/%d/%Y')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 940 entries, 0 to 939
Data columns (total 15 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Id                         940 non-null    int64
 1   ActivityDate               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
dtypes: datetime64[ns](1), float64(7), int64(7)
memory usage: 110.3 KB
```

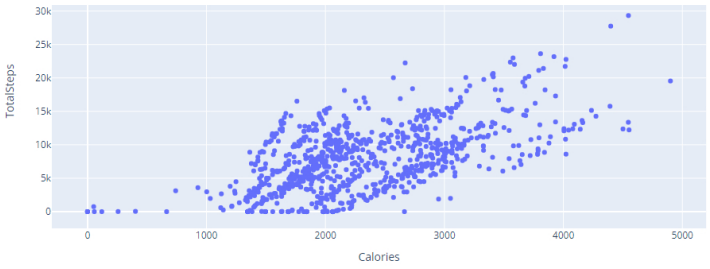
Now we can see that the ActivityDate is well formated

In [7]:
```
# Let us have a look at the descriptive statitical view of the dataset
df.describe()
```

Out[7]:

	Id	TotalSteps	TotalDistance	TrackerDistance	LoggedActivitiesDistance	VeryActiveDistance	ModeratelyActiveDistance	LightActiveDistance
count	9.400000e+02	940.000000	940.000000	940.000000	940.000000	940.000000	940.000000	940.000000
mean	4.855407e+09	7637.910638	5.489702	5.475351	0.108171	1.502681	0.567543	3.340819
std	2.424805e+09	5087.150742	3.924606	3.907276	0.619897	2.658941	0.883580	2.040655
min	1.503960e+09	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2.320127e+09	3789.750000	2.620000	2.620000	0.000000	0.000000	0.000000	1.945000
50%	4.445115e+09	7405.500000	5.245000	5.245000	0.000000	0.210000	0.240000	3.365000
75%	6.962181e+09	10727.000000	7.712500	7.710000	0.000000	2.052500	0.800000	4.782500
max	8.877689e+09	36019.000000	28.030001	28.030001	4.942142	21.920000	6.480000	10.710000

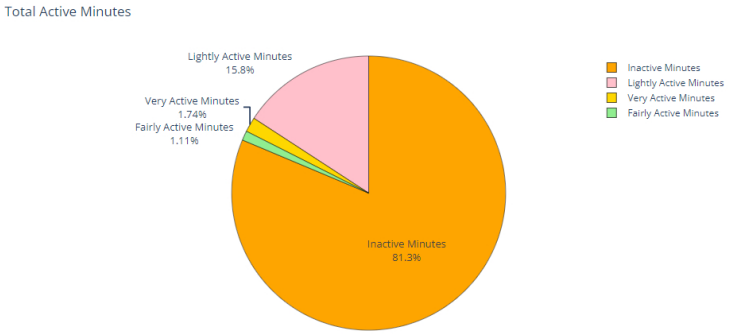
In [8]:
```
# Here we are checking the raltionship between calories burned and the total steps walked in a day.
figure = px.scatter(data_frame = df, x="Calories",
                    y="TotalSteps",
                    title="Relationship between Calories & Total Steps")
figure.show()
```



It is clearly visible that there is a linear relationship between these two variables 'Total Steps' & 'Calories'

In [9]:
```
label = ["Very Active Minutes", "Fairly Active Minutes",
         "Lightly Active Minutes", "Inactive Minutes"]
counts = df[["VeryActiveMinutes", "FairlyActiveMinutes",
             "LightlyActiveMinutes", "SedentaryMinutes"]].mean()
colors = ['gold','lightgreen', "pink", "orange"]

fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Total Active Minutes')
fig.update_traces(hoverinfo='value', textinfo='label+percent', textfont_size=13,
                  marker=dict(colors=colors, line=dict(color='black', width=.5)))
fig.show()
```



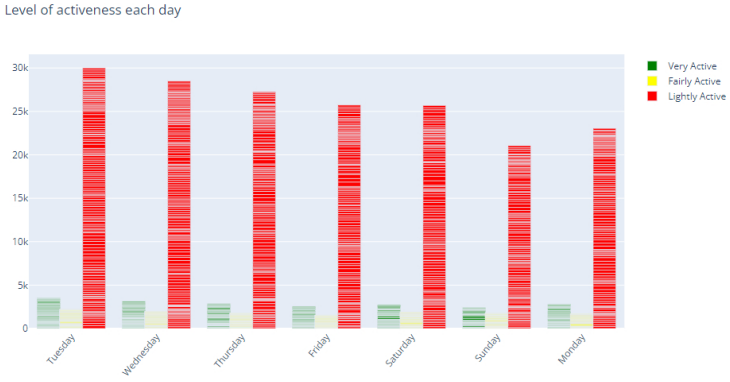
It is clear that a lions portion of a day remains inactive, whereas 1.74% remains effectively active.

In [10]:
```
#Let us have a look the day in a week
df['Day'] = df['ActivityDate'].dt.day_name()
df['Day'].head()
```

Out[10]:
```
0      Tuesday
1    Wednesday
2     Thursday
3       Friday
4     Saturday
Name: Day, dtype: object
```

In [11]:
```
# here we are checking the different levels of activeness in each day.
```

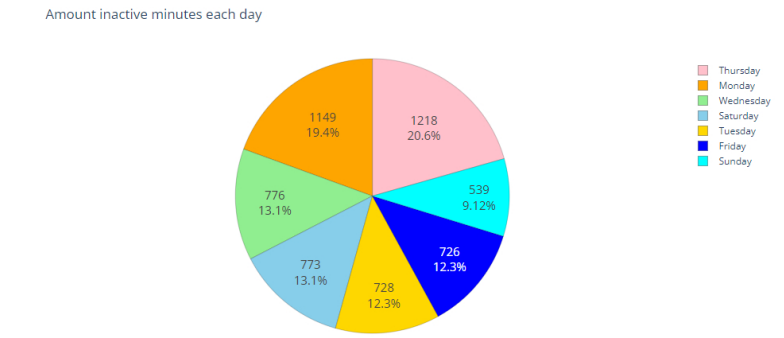
In [12]:
```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=df["Day"],
    y=df["VeryActiveMinutes"],
    name='Very Active',
    marker_color='green'
))
fig.add_trace(go.Bar(
    x=df["Day"],
    y=df["FairlyActiveMinutes"],
    name='Fairly Active',
    marker_color='yellow'
))
fig.add_trace(go.Bar(
    x=df["Day"],
    y=df["LightlyActiveMinutes"],
    name='Lightly Active',
    marker_color='red'
))
fig.update_layout(title_text = 'Level of activeness each day', barmode='group', xaxis_tickangle=-50)
fig.show()
```



In [13]:
```
# Amount of inactive minutes in a day
day = df["Day"].value_counts()
label = day.index
counts = df["SedentaryMinutes"]
colors = ['gold','lightgreen', "pink", "blue", "skyblue", "cyan", "orange"]

fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Amount inactive minutes each day')
fig.update_traces(hoverinfo='label', textinfo='value+percent', textfont_size=15
```

```
                  marker=dict(colors=colors, line=dict(color='black', width=.2)))
fig.show()
```



Thursday and Monday are the most inactive days. It might imply that the person is busy in doing some other jobs in these two days. However, other days are comparatively closer to each other regarding inactiveness.

Findings:

From the above analysis we can conclude the following:

- It is possible that people misses a large portion of their time to take care of their health.
- We have found out that almost more than 80% of their time remains inactive, meaning that people use less time to care about their health. There might be various reason behind this inactiveness.
- Other than sunday and monday, people use to remain inactive most of the day of week.
- It is also important to mention that 'Calories' burnt and 'TotalSteps' are corelated to each other and Calories burn increase with the increase of TotalStep.