Assignment 2 - Individual Checkpoint 2

Author: Hanchen Wang

Group ID: CC08-3

Driving problem: Do they avoid inactivity in at least 10 hours a day?

Personal Planning and Progress Github wiki page: https://github.sydney.edu.au/jlin4572/2024-DATA3406-CC08-Group03/wiki/Hanchen-Wang's--Page (https://github.sydney.edu.au/jlin4572/2024-DATA3406-CC08-Group03/wiki/Hanchen-Wang's--Page)

Overview of the work:

IDs explored in Checkpoint 1

- *1503960366*
- *1624580081*
- *1644430081*

ID report for Checkpoint 2

- The participant with id *1503960366*
- By analysing csv dataset *hourlySteps_merged* and focusing on the selected id, I will analyse and visualise numerous aspects of data while varying the threshold of inactivity. Finally, the discussion will focus on the impact of different threshold on result to the driving problem.
- While presenting visual analysis of results for participant with id *1503960366*, Literate Programming is strictly applied to this Jupyter Notebook.

Assumptions & Predictions:

Week8

- The participants in three datasets are consistent. (Has been confirmed in data analysis.)
- There are independence between different participants in all datasets. (Depends on the data collection process, we assume this is true.)

Week9, 10

- The data records for the selected person is complete and accurate for conducting a reliable visual analysis. (Depends on the data collection process, we assume this is true.)
- As agreed with team members, a daily step count of 5000 is an active day.

Load Dataset

start_date: 22/9/2024 end_date: 22/9/2024

Three datasets are provided and they are stored in the **src** folder.

Rationale: Considering my driving problem, only the dataset that relates to hourly step count will be used.

We use the **read_csv** function in hourly step count to read **hourlySteps_merged.csv** as dataframe.

In [15]:
```
# start_date: 22/9/2024
# end_date: 22/9/2024

# Import all necessary packages
import pandas as pd
from plotnine import ggplot, aes, geom_bar, labs, theme_classic, theme
import matplotlib.pyplot as plt
import plotly.express as px
import warnings

# Suppress all warnings
warnings.filterwarnings("ignore")

# Read CSV file from the 'src' folder as dataframe
df_hour = pd.read_csv('src/hourlySteps_merged.csv')
```

The hourly step count dataset has been successfully loaded and we can access it using **df_hour**. Let's have an overview of those datasets by printing their size and column names.

In [16]:
```
# start_date: 22/9/2024
# end_date: 22/9/2024

# Access the size of data set using .shape[0] and .shape[1]
print("The size of hourly dataset:", df_hour.shape[0], "rows and", df_hour.shape[1], "columns." )

# As .columns return a list of column name, use ',' to join them as a string
print("The column names of hourly dataset:", ", ".join(df_hour.columns))
```

```
The size of hourly dataset: 22099 rows and 3 columns.
The column names of hourly dataset: Id, ActivityHour, StepTotal
```

As this dataset have 22099 rows representing records for numerous participants, our first step in data analysis is to filter the dataframe to have only selected person: *1503960366*.

Check Missing Values

start_date: 22/9/2024 end_date: 22/9/2024

Rationale: Before doing any data analysis, we need to make sure we handle any missing values properly.

Prediction: There should not exist any missing values, since this has been checked in checkpoint 1.

In [17]:
```
# start_date: 22/9/2024
# end_date: 22/9/2024

df_hour.isnull().any()
```

Out[17]:
```
Id              False
ActivityHour    False
StepTotal       False
dtype: bool
```

Since results for all columns are **False**, there are **no missing values** across the hourly step count dataset.

Data Analysis

Filter Dataframe

start_date: 22/9/2024 end_date: 22/9/2024

Rationale: To ensure we only focus on the selected person, the dataframe needs to be filtered.

Prediction: The size of dataframe should decrease significantly in number of rows after filtering.

In [18]:
```
# start_date: 22/9/2024
# end_date: 22/9/2024

# Declare the id of selected person for code reusability
id_selected = 1503960366

# Filter the rows in df_hour with id_selected in 'Id' column using isin()
df_id_selected = df_hour[df_hour['Id'].isin([id_selected])]

# Convert ActivityHour to datetime format for proper time series plotting
df_id_selected.loc[:, 'ActivityHour'] = pd.to_datetime(df_id_selected['ActivityHour'], format='%m/%d/%Y %I:%M:%S %p')

# Observe the filtered dataframe
df_id_selected
```

Out[18]:

	Id	ActivityHour	StepTotal
0	1503960366	2016-04-12 00:00:00	373
1	1503960366	2016-04-12 01:00:00	160
2	1503960366	2016-04-12 02:00:00	151
3	1503960366	2016-04-12 03:00:00	0
4	1503960366	2016-04-12 04:00:00	0
...	...	...	...
712	1503960366	2016-05-11 16:00:00	289
713	1503960366	2016-05-11 17:00:00	245
714	1503960366	2016-05-11 18:00:00	3449
715	1503960366	2016-05-11 19:00:00	293
716	1503960366	2016-05-11 20:00:00	1209

717 rows × 3 columns

The number of rows in the dataframe decreases to 717 rows as expected, while keeping all three columns and *1503960366* in **Id** column.

Define Inactivity Threshold

start_date: 23/9/2024 end_date: 26/9/2024

Rationale: To answer the driving problem "Do they avoid inactivity in at least 10 hours a day?", we have to define what is **inactivity**.

As agreed with other team members, a **daily step count of 5000** is an active day.

Let's set three threshold for catarizing if the participant is active in an hour and visualize acorss a time series for them.

1. Does not account for sleeping hours.
2. Assume the sleeping time to be 6 hours.
3. Assume the sleeping time to be 8 hours.

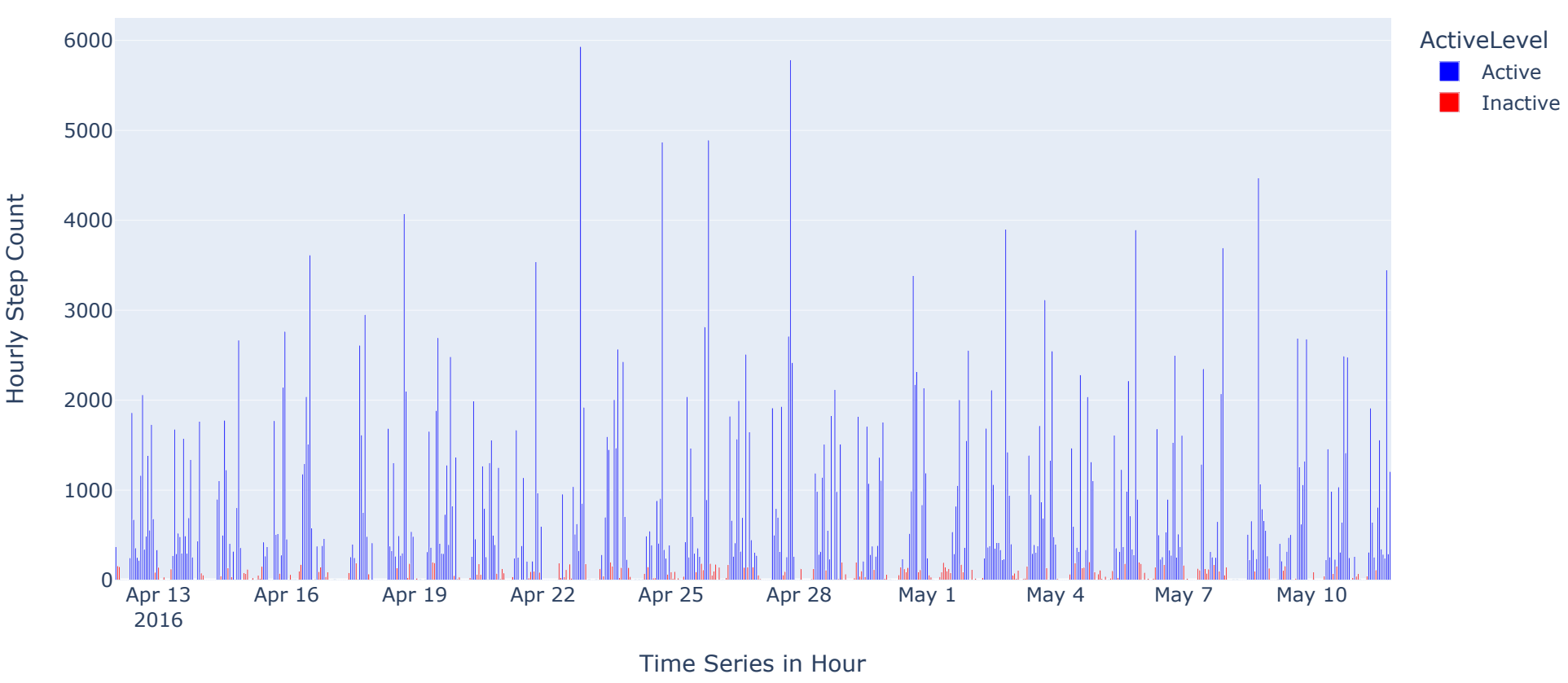
Prediction:

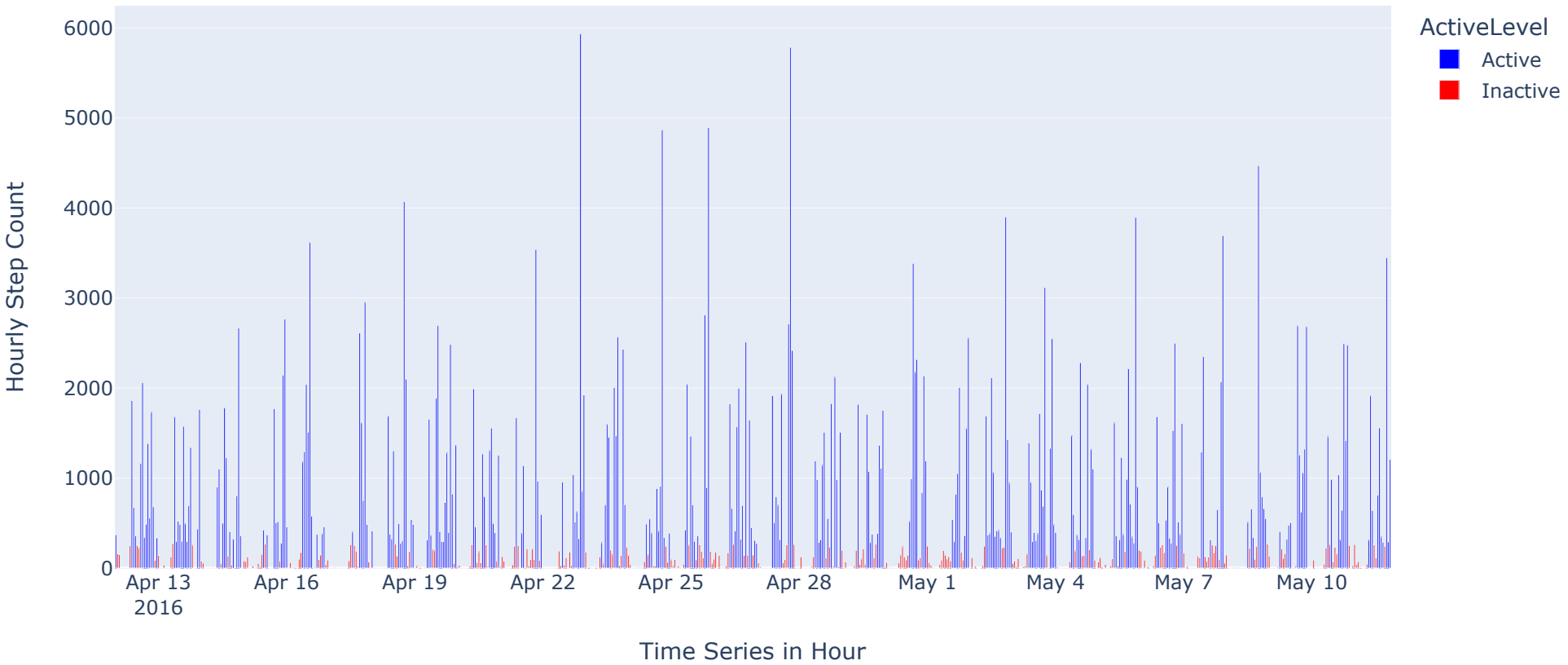
- There might be obvious differences between the plot of three thresholds due to varying value of hourly step count required to reach an active hour.
- The plot showing a time series of hourly step count could have a large amount of bars due to massive number of dates and hours. Therefore, an interative plot that allows zooming in and out is necessary for clarity.

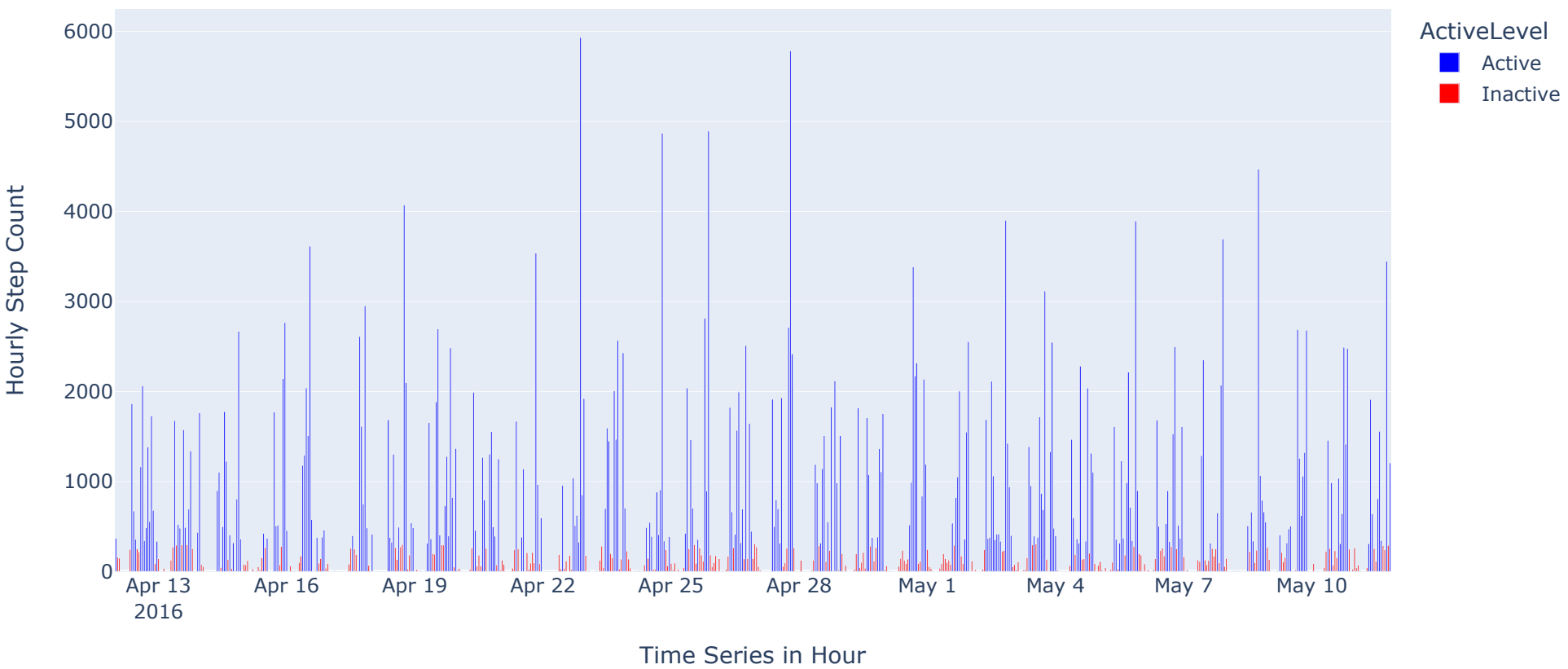
In [19]:
```
# start_date: 23/9/2024
# end_date: 26/9/2024

# Define three thresholds of inactivity for comparison:
active_day_constant = 5000
threshold1 = int(active_day_constant/24) # Does not account for sleeping hours
threshold2 = int(active_day_constant/(24-6)) # 6 sleeping hours
threshold3 = int(active_day_constant/(24-8)) # 8 sleeping hours


# Create copies of dataframes for three thresholds
df_t1 = df_id_selected.copy()
df_t2 = df_id_selected.copy()
df_t3 = df_id_selected.copy()


# Function that return active status
def determine_active_level(steps, threshold):
    # Less than threshold -> Inactive (Red). Otherwise Active (Blue)
    if steps < threshold:
        return 'Inactive'
    else:
        return 'Active'

# Run function for three dataframes that adds a new column showing activity level
df_t1['ActiveLevel'] = df_t1['StepTotal'].apply(determine_active_level, threshold=threshold1)
df_t2['ActiveLevel'] = df_t2['StepTotal'].apply(determine_active_level, threshold=threshold2)
df_t3['ActiveLevel'] = df_t3['StepTotal'].apply(determine_active_level, threshold=threshold3)
```

```
# Function that create interactive bar plots for given dataframe
def plot_overall_activity_level(dataframe, threshold, fig_name):
    fig =px.bar(dataframe,
             x='ActivityHour',
             y='StepTotal',
             labels={'ActivityHour': 'Time Series in Hour', 'StepTotal': 'Hourly Step Count'},
             title=f'{fig_name}: Overview of Hourly-activity-level for id {id_selected} (Threshold: {threshold})',
             color=f'ActiveLevel',
             color_discrete_map={'Inactive': 'red', 'Active': 'blue'})
    return fig


# Run function for three dataframes
fig1a = plot_overall_activity_level(df_t1, threshold1, "Fig1a")
fig1b = plot_overall_activity_level(df_t2, threshold2, "Fig1b")
fig1c = plot_overall_activity_level(df_t3, threshold3, "Fig1c")


# Show all plots
fig1a.show()
fig1b.show()
fig1c.show()
```

Fig1a: Overview of Hourly-activity-level for id 1503960366 (Threshold: 208)







After interacting with the plots *Fig1a*, *Fig1b* and *Fig1c*, we discover:

- While the majority of bars show blue color representing an activity hour, there is **a small number of red bars** representing inactivity after zooming in the plot.
- There are quite a few time points with no bars, caused by its step count of 0. This may represent either an **inactive hour** or **sleeping time**.
- The actual result is **opposed to my prediction**, as there are no obvious differences between three thresholds based on quick observation to three plots above.

Hence, more detailed computations and visualizations are required to find out the impact of different thresholds on answering our driving problem.

Visualize Active/Inactive Hours

start_date: 26/9/2024 end_date: 27/9/2024

Rationale: For the ease of in-depth visualizations, we need to count the number of active hours in each day and save into a dataframe.

In [20]:
```
# start_date: 26/9/2024
# end_date: 27/9/2024

# Function that restructures the dataframe for counting active hours per day
def count_active_hours(dataframe):
    # Extract the date from ActivityHour
    dataframe['Date'] = pd.to_datetime(dataframe['ActivityHour']).dt.date

    # Keep rows where ActiveLevel is "Active"
    dataframe = dataframe[dataframe['ActiveLevel'] == 'Active']

    # Use size() to count the number of "active"s grouping by date
    dataframe = dataframe.groupby('Date').size().reset_index()

    # Rename the column names for plotting
    dataframe.columns = ['Date', 'ActiveHours']

    return dataframe

# Run function for three dataframes
df_t1 = count_active_hours(df_t1)
df_t2 = count_active_hours(df_t2)
df_t3 = count_active_hours(df_t3)

# Add threshold column to each dataframe for identification
df_t1['Threshold'] = 'Threshold1 (5000/24)' # Does not account for sleep
df_t2['Threshold'] = 'Threshold2 (5000/18)' # 6 hours of sleep
df_t3['Threshold'] = 'Threshold3 (5000/16)' # 8 hours of sleep

# Concatenate the three dataframes into one for comparison
df_active_level = pd.concat([df_t1, df_t2, df_t3])

# Glimpse to dataframe for visualization
df_active_level
```

Out[20]:

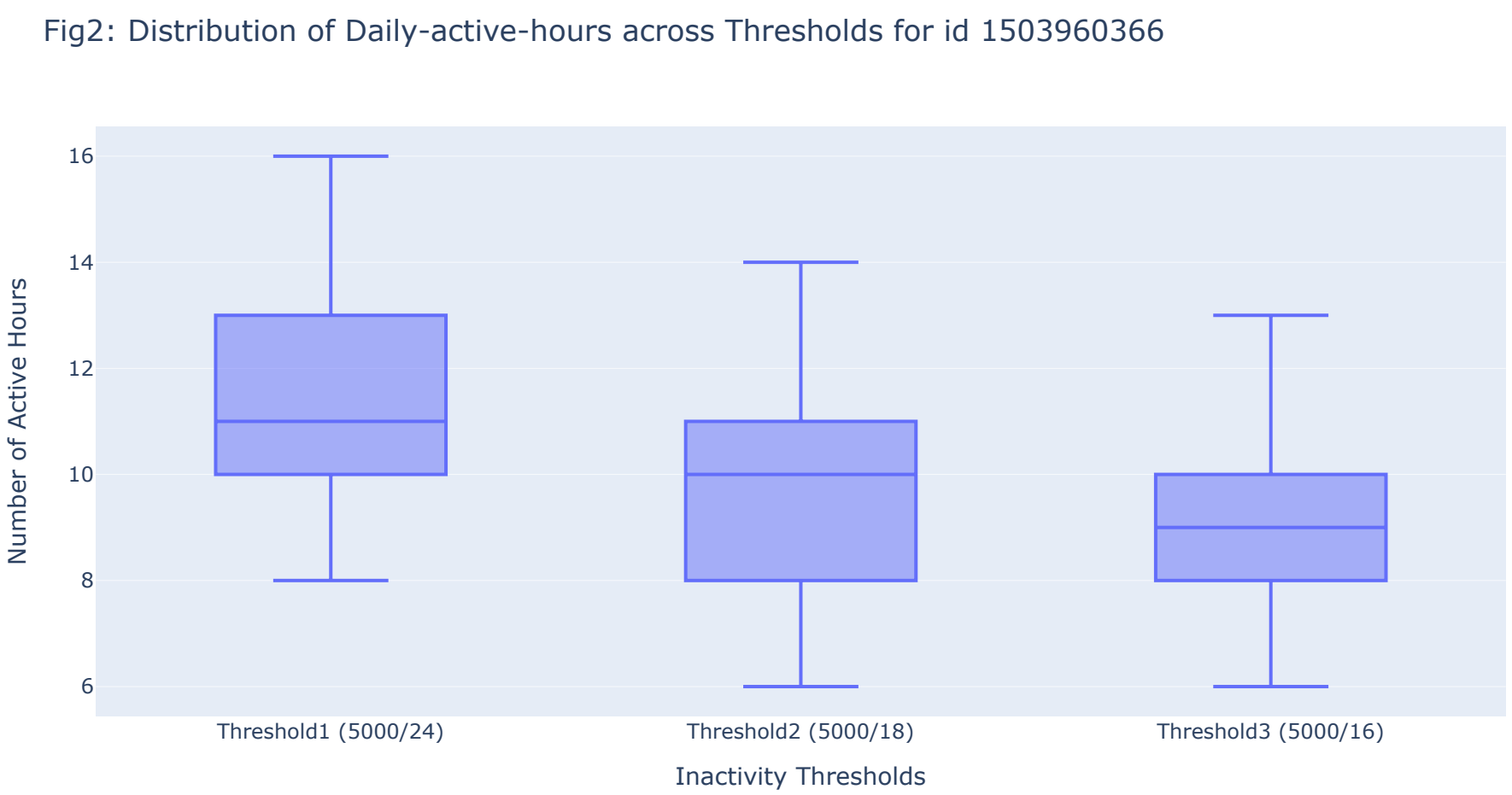
	Date	ActiveHours	Threshold
0	2016-04-12	16	Threshold1 (5000/24)
1	2016-04-13	14	Threshold1 (5000/24)
2	2016-04-14	10	Threshold1 (5000/24)
3	2016-04-15	9	Threshold1 (5000/24)
4	2016-04-16	10	Threshold1 (5000/24)
...	...	...	...
25	2016-05-07	6	Threshold3 (5000/16)
26	2016-05-08	8	Threshold3 (5000/16)
27	2016-05-09	10	Threshold3 (5000/16)
28	2016-05-10	8	Threshold3 (5000/16)
29	2016-05-11	8	Threshold3 (5000/16)

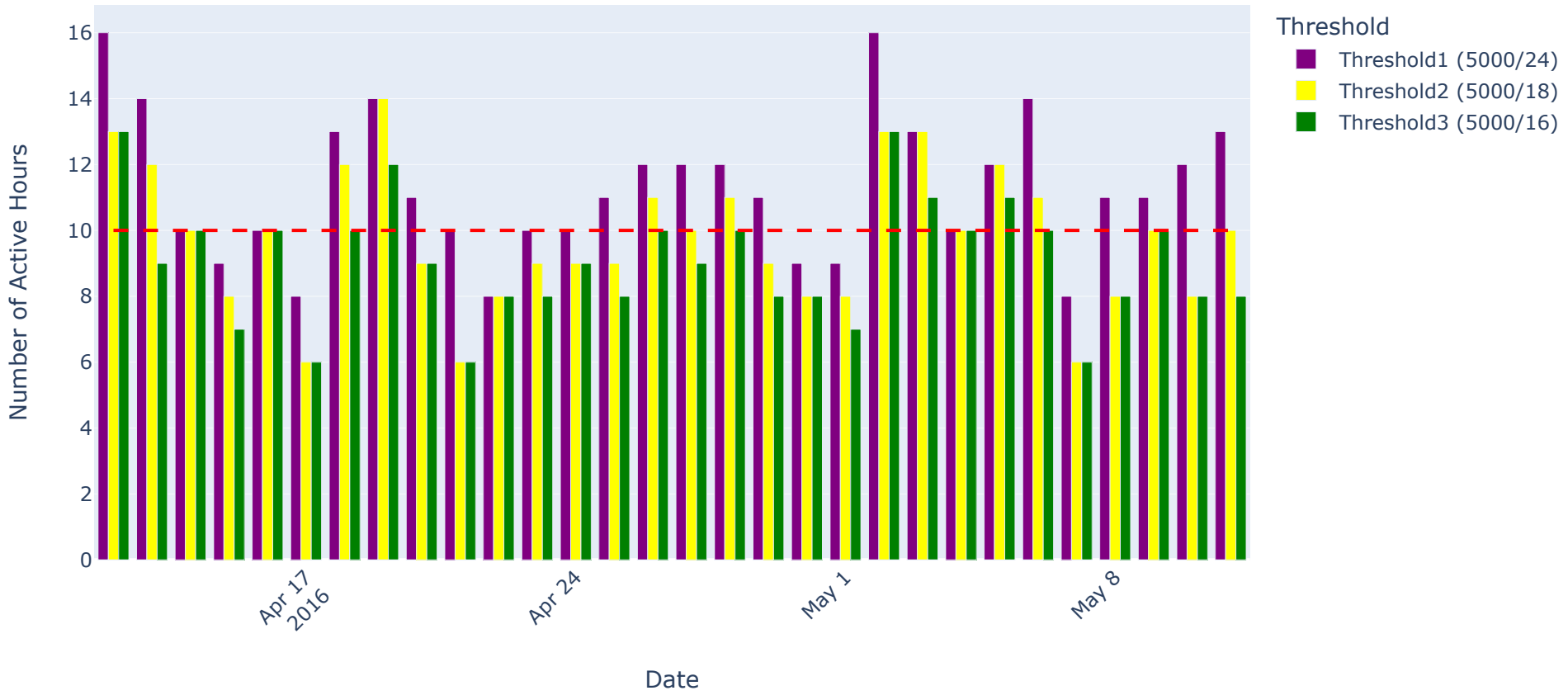
90 rows × 3 columns

Rationale: To visualize the count of daily active hours based on three thresholds for id *1503960366*, box chart and bar chart can be utilized effectively for clear comparison.

Prediction: A lower threshold would lead to higher number of active hours.

In [21]:
```
# start_date: 27/9/2024
# end_date: 27/9/2024

# Create an interactive box chart for checking the distribution of active hours
fig2 = px.box(df_active_level,
              x='Threshold',  # Add Threshold to the x-axis for comparison
              y='ActiveHours',
              title=f'Fig2: Distribution of Daily-active-hours across Thresholds for id {id_selected}',
              labels={'ActiveHours': 'Number of Active Hours', 'Threshold': 'Inactivity Thresholds'}
)


# Create an interactive grouped bar chart for comparing across the dates by thresholds
fig3 = px.bar(df_active_level,
              x='Date',
              y='ActiveHours',
              color='Threshold',  # Use Threshold to differentiate bars
              barmode='group',  # Group bars side by side for comparison
              title=f'Fig3: Comparison of Daily-active-hours across Thresholds for id {id_selected} ',
              labels={'Date': 'Date', 'ActiveHours': 'Number of Active Hours'},
              color_discrete_map={'Threshold1 (5000/24)': 'purple',
                                  'Threshold2 (5000/18)': 'yellow',
                                  'Threshold3 (5000/16)': 'green'}
)

# Customize the x-axis to show all dates clearly (rotate if necessary)
fig3.update_layout(
    xaxis_tickangle=-45,  # Rotate x-axis labels for better readability
    xaxis_title='Date',
    yaxis_title='Number of Active Hours',
    legend_title='Threshold'
)

# Add a red horizontal line at y=10 to indicate the 10-hour inactivity threshold
fig3.add_shape(
    type="line",
    x0=df_active_level['Date'].min(), y0=10, # Set start from the minimum date
    x1=df_active_level['Date'].max(), y1=10, # Set end at the maximum date
    line=dict(color="red", width=2, dash="dash"),  # Add red line
)


# Show plots
fig2.show()
fig3.show()
```





According to the plot *Fig2*, we discover that the value of threshold has a **significant impact** to conclude whether the participant avoids inactivity in at least 10 hours.

- The daily active hours based on Threshold 1 mainly ranges from 10 to 13 with a median of 11 after excluding outliers, which passes the requirement of at least 10 active hours per day.
- The daily active hours based on Threshold 2 mainly ranges from 8 to 11 with a median of 10 after excluding outliers, which is rejected by the requirements as the participants did not avoid inactivity in at least 10 hours for most days of data.
- The daily active hours based on Threshold 3 mainly ranges from 8 to 10 with a median of 9 after excluding outliers, which definitely fails the requirement.

According to the plot *Fig3*, we can observe that:

- The majority of purple bars can exceed the 10-active-hour requirement represented by the red dashed line. There are only six days of exceptions.
- Most yellow and green bars cannot reach the red dashed line, indicating the domination of inactivity using the thresholds considering 6 hours or 8 hours of sleep time.

Overall, we can conclude that the participant with id *1503960366* does not avoid inactivity in at least 10 hours a day considering results of all three thresholds.

Conclusion

start_date: 27/9/2024 end_date: 27/9/2024

In this **checkpoint 2**, we performed an in-depth analysis of the hourly step count data focusing on id *1503960366* by:

1. Filtering the dataframe for the selected person.
2. Setting three levels of thresholds to define an inactive hour.
3. Visualizing the impact of varying thresholds on answering the driving problem.

As a **final statement**, I learnt how to implement a thorough visualization of any person in the dataset by promoting code reusability and literate programming in my notebook, while discovering the significant impact of thresholds on answering our driving problem.

My implementation highly relates to the driving problem "Do they avoid inactivity in at least 10 hours a day?" by creating the following data visualizations:

- Fig 1a, 1b, 1c: Three interactive bar charts providing an **overview of hourly step count data**. This visualization highlights the activity levels at each hour, helping to identify inactive periods across the entire timeline.
- Fig 2: A box chart showing the **distribution of daily active hours** considering three thresholds. This plot offers insights into the variability of the participant's daily active periods, emphasizing how often they meet or exceed the 10-hour activity threshold.
- Fig 3: A grouped bar chart depicting the comparison of **number of active hours across different days** for three thresholds. This chart reveals trends in their daily activity, showing whether they consistently avoid inactivity for at least 10 hours each day.

Hence, all these visualizations help answering the driving problem for the selected person.