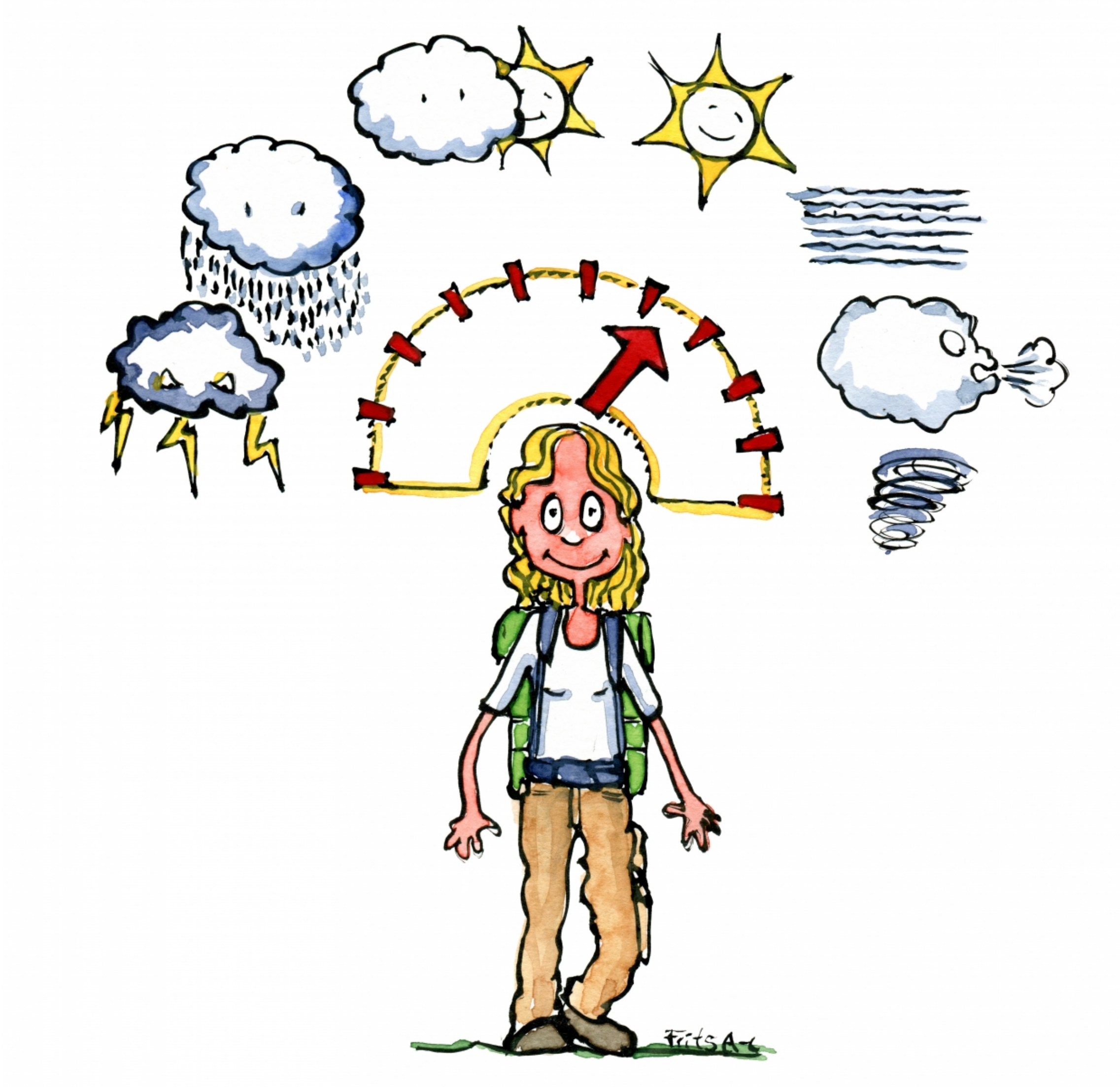
Statistic Method Analysis Notebook focus on weather.



Importing nessary Python Library for Analysis.

Data Analysis of ring data and weather interpretation. This EDA to analyze the Mood Data with Weather Data, show the interpretive data and help researcher figure the impact between different feature relate with only Weather.

In [1]:
```
# import the correct library
import pandas as pd
import seaborn as sns
```

```python
import matplotlib.pyplot as plt
import plotly.express as px
import numpy as np

from knmy import knmy

# Comment this if the data visualisations doesn't work on your side
%matplotlib inline

plt.style.use('bmh')
pd.set_option('display.max_rows', 50)
pd.set_option('display.max_columns', None)

%matplotlib inline
sns.set(color_codes=True)
```

Import MoodMetric Rings Data

Here we have some rings that actually being weared and it's record the people data. However it alway has some missing leading and wrong data identity. Therefore it should being imported and skim the overview be data analyst.

In [3]:
```python
# import the data into the Jupiter notebook.
df_F1 = pd.read_excel('F1_ID2049_moodmetric data_NoASD.xlsx', sheet_name='Blad1', i
df_F2 = pd.read_excel('F2_ID2175_MoodmetricData2022.xlsx', sheet_name='Csv file', i
df_M2 = pd.read_excel('M2_ID253_MyData_NoASD.xlsx', index_col=0)
df_M3 = pd.read_excel('M3_ID2551_DataNoASD.xlsx', index_col=0)
df_U1 = pd.read_excel('U1_ID2542_MyData.xlsx', index_col=0)
```

In [4]:
```python
# This part will support to see the insight of data from weekly hour.
df_F2_week_hour = pd.read_excel('F2_ID2175_MoodmetricData2022.xlsx', sheet_name='Re
```

In [5]:
```python
# Checking the collumn and Shape of the dataset to see how long of the dataset.
df_F1.shape
df_F2.shape
df_M2.shape
df_M3.shape
```

Out[5]:
```
(17758, 9)
```

In [6]:
```python
# Calculated the nessary row.
df_F1.count()
df_F2.count()
df_M2.count()
df_M3.count()
```

```
Out[6]: Device_ID      17758
        Ring_ID        17758
        MM level       17758
        SCR/min        17758
        SCL            17757
        Step count     17758
        aa             17758
        Time_ISO       17758
        Time_UNIX      17758
        dtype: int64
```

Tranforming and cleaning data.

The process of data transformation can also be referred to as extract/transform/load (ETL). The extraction phase involves identifying and pulling data from the various source systems that create data and then moving the data to a single repository. Next, the raw data is cleansed, if needed. It's then transformed into a target format that can be fed into operational systems or into a data warehouse, a date lake or another repository for use in business intelligence and analytics applications. The transformation may involve converting data types, removing duplicate data and enriching the source data.

```
In [7]: # Dropping irrelevant columns
df_M2 = df_M2.rename(columns={"aa":"Calibration_Value","MM level":"MM_level","Step
df_M2
```

Out[7]:

User_ID	Device_ID	Ring_ID	MM_level	SCR/min	SCL	Step_count	Calibrati
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	48.863636	0	0.473485	1	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	70.689655	11	0.363372	7	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	70.689655	0	1.736111	0	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	70.689655	0	0.217014	0	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	100.000000	0	0.070701	7	
...	...	...	...	...	...	...	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	60.655738	3	0.919118	11	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	100.000000	5	0.919118	8	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	63.934426	2	0.300481	0	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	55.737705	3	0.355114	0	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	55.737705	3	0.558036	0	

6836 rows × 9 columns

In [8]:

```
df_F2 = df_F2.rename(columns={"aa":"Calibration_Value","MM level":"MM_level","Step
df_F2
```

Out[8]:

User_ID	Device_ID	Ring_ID	MM_level	SCR/min	SCL	Step_count	Calibration_Valu
2175	B8E6E335	C5:C7:9C:58:95:27	54.982818	5	0.710227	0	2
2175	B8E6E335	C5:C7:9C:58:95:27	38.277512	0	7.812500	4	2
2175	B8E6E335	C5:C7:9C:58:95:27	36.363636	5	7.812500	9	2
2175	B8E6E335	C5:C7:9C:58:95:27	37.320574	0	7.812500	11	2
2175	B8E6E335	C5:C7:9C:58:95:27	41.148325	1	5.208333	7	2
...	...	...	...	...	...	...	
2175	F8BC4330	E0:76:33:4B:B2:3B	69.182390	0	NaN	2	1
2175	F8BC4330	E0:76:33:4B:B2:3B	80.503145	1	15.625000	11	1
2175	F8BC4330	E0:76:33:4B:B2:3B	100.000000	0	5.208333	0	1
2175	F8BC4330	E0:76:33:4B:B2:3B	100.000000	2	1.302083	0	1
2175	F8BC4330	E0:76:33:4B:B2:3B	100.000000	1	0.976562	2	1

23969 rows × 9 columns

In [9]:

```
df_F1 = df_F1.rename(columns={"MM level":"MM_level","Step count":"Step_count","all
df_F1
```

Out[9]:

User_ID	Device_ID	Ring_ID	MM_level	SCR/min	SCL	Step_count	Calibrati
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	41.340782	0	2.232143	5	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	39.106145	6	1.953125	8	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	48.044693	1	2.232143	4	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	53.631285	2	2.604167	14	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	51.396648	0	2.604167	2	
...	...	...	...	...	...	...	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	71.140940	5	3.906250	2	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	72.483221	0	2.604167	4	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	73.825503	5	3.125000	7	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	64.429530	2	3.906250	14	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	52.348993	0	3.125000	8	

22079 rows × 9 columns

In [10]:

```
df_M3 =  df_M3.rename(columns={"MM level":"MM_level","Step count":"Step_count","aa"
df_M3
```

Out[10]:

User_ID	Device_ID	Ring_ID	MM_level	SCR/min	SCL	Step_count	Calibratic
2551	22b77d7829c68dcd	E2:1F:7B:35:CC:08	52.073733	8	0.679348	7	
2551	22b77d7829c68dcd	E2:1F:7B:35:CC:08	71.895425	8	0.236742	0	
2551	22b77d7829c68dcd	E2:1F:7B:35:CC:08	79.738562	1	0.220070	1	
2551	22b77d7829c68dcd	E2:1F:7B:35:CC:08	82.352941	4	0.264831	0	
2551	22b77d7829c68dcd	E2:1F:7B:35:CC:08	98.039216	7	0.312500	0	
...	...	...	...	...	...	...	
2551	22b77d7829c68dcd	E2:1F:7B:35:CC:08	68.292683	11	0.434028	22	
2551	22b77d7829c68dcd	E2:1F:7B:35:CC:08	46.689895	3	0.422297	4	
2551	22b77d7829c68dcd	E2:1F:7B:35:CC:08	38.327526	3	0.434028	15	
2551	22b77d7829c68dcd	E2:1F:7B:35:CC:08	75.958188	8	0.229779	0	
2551	22b77d7829c68dcd	E2:1F:7B:35:CC:08	81.533101	12	0.318878	5	

17758 rows × 9 columns

In [11]:
```
df_F1_1 = pd.read_excel('New_F1_Time.xlsx', index_col=0)
```

In [12]:
```
data_ring_insight = df_F1.copy()
```

In [13]:
```
data_ring_insight
```

Out[13]:

	Device_ID	Ring_ID	MM_level	SCR/min	SCL	Step_count	Calibrati
User_ID							
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	41.340782	0	2.232143	5	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	39.106145	6	1.953125	8	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	48.044693	1	2.232143	4	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	53.631285	2	2.604167	14	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	51.396648	0	2.604167	2	
...	...	...	...	...	...	...	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	71.140940	5	3.906250	2	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	72.483221	0	2.604167	4	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	73.825503	5	3.125000	7	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	64.429530	2	3.906250	14	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	52.348993	0	3.125000	8	

22079 rows × 9 columns

Data Generation Part with technique

This part will add a new condition of data belong different location. The assumption that i have made here with certain amount arrange period of the day. Our user will going to wear the ring and they will located at specific place of the area. The asumption will focus only the area in Eindhoven, therefore we will able to explore more detail of how stress level will be falsity.

In [14]:

```
# data_ring_insight = data_ring_insight.drop(['Time','Specific Time','Time_UNIX','T
data_ring_insight['Time_New'] = data_ring_insight['Time_ISO'].astype(str).str[11:16
data_ring_insight['Time_New'] = data_ring_insight['Time_New'].str.replace(':','.')
data_ring_insight['Time_New'] = data_ring_insight['Time_New'].astype(float)
data_ring_insight
```

Out[14]:

	Device_ID	Ring_ID	MM_level	SCR/min	SCL	Step_count	Calibrati
User_ID							
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	41.340782	0	2.232143	5	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	39.106145	6	1.953125	8	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	48.044693	1	2.232143	4	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	53.631285	2	2.604167	14	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	51.396648	0	2.604167	2	
...	...	...	...	...	...	...	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	71.140940	5	3.906250	2	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	72.483221	0	2.604167	4	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	73.825503	5	3.125000	7	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	64.429530	2	3.906250	14	
2049	57dee7ab79b1d20a	C6:03:16:2D:1A:33	52.348993	0	3.125000	8	

22079 rows × 10 columns

In [15]:

```
# data_ring_insight_filter = data_ring_insight.loc[(data_ring_insight['Time_New'] >
# data_ring_insight_filter

discard = ["1970"]

# drop rows that contain the partial string "Sci"
df_M2[~df_M2.Time_ISO.str.contains('|'.join(discard))]
```

Out[15]:

User_ID	Device_ID	Ring_ID	MM_level	SCR/min	SCL	Step_count	Calibrati
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	50.000000	0	1.302083	2	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	41.379310	0	1.116071	0	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	41.379310	1	1.302083	0	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	48.275862	1	0.919118	0	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	41.379310	1	1.420455	0	
...	...	...	...	...	...	...	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	60.655738	3	0.919118	11	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	100.000000	5	0.919118	8	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	63.934426	2	0.300481	0	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	55.737705	3	0.355114	0	
2536	f2ca147e382a66e9	C6:03:16:2D:1A:33	55.737705	3	0.558036	0	

6829 rows × 9 columns

Add Fabric DataSet

To determine the time period associated with a certain place or the potential location of the user at a given hour, a mock dataset was added.

In [16]:

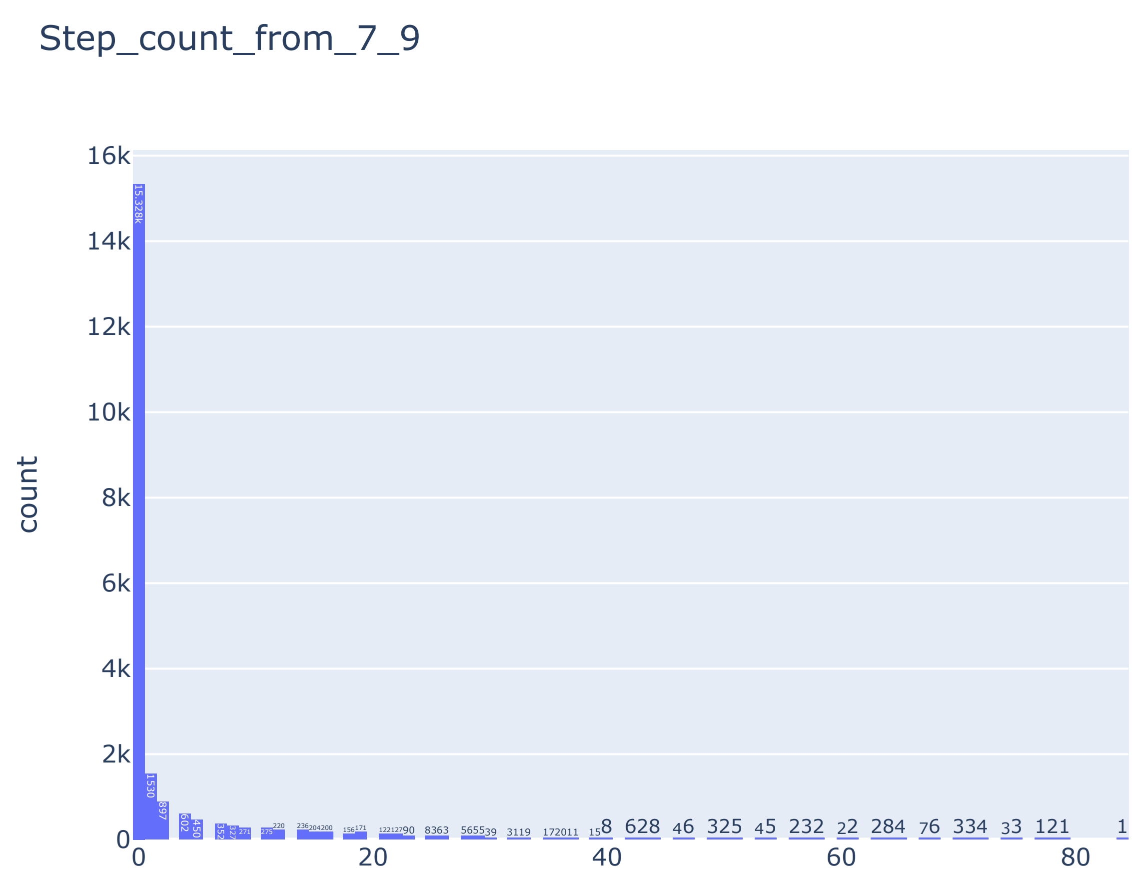
```
conditions = [
    (data_ring_insight['Time_New'] <= 7),
    (data_ring_insight['Time_New'] > 7) & (data_ring_insight['Time_New'] <= 9),
    (data_ring_insight['Time_New'] > 9) & (data_ring_insight['Time_New'] <= 16),
    (data_ring_insight['Time_New'] > 16)
]

# Create a list of the values we want to assign for each condition
location_area_point = ['Part_time','Outside','University','Home']

data_ring_insight['location'] = np.select(conditions, location_area_point)
```

Add user_ID from index to become a collumns

In [18]:
```python
df_F1.reset_index(inplace=True, level=['User_ID'])
df_M2.reset_index(inplace=True, level=['User_ID'])
df_F2.reset_index(inplace=True, level=['User_ID'])
df_M3.reset_index(inplace=True, level=['User_ID'])
df_U1.reset_index(inplace=True, level=['User_ID'])
```

Read the Data Trend of Step Count.

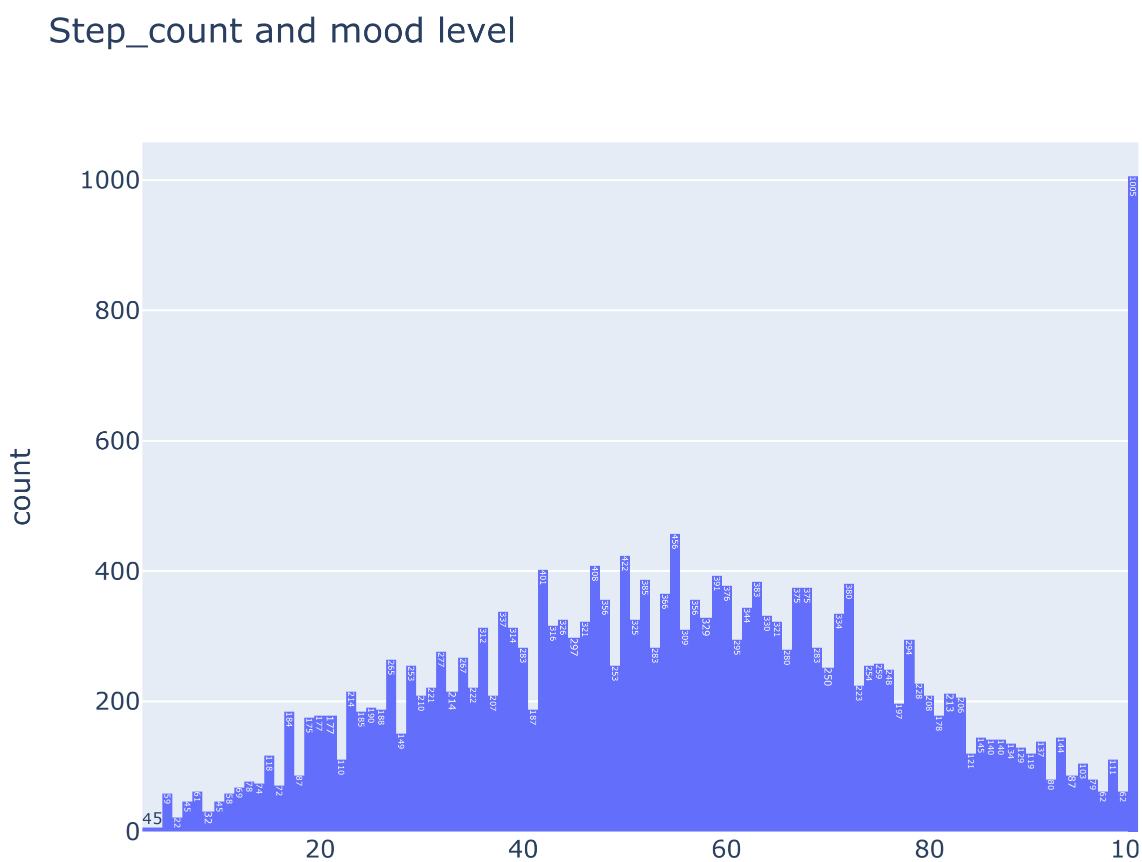
In [19]:
```python
import plotly.express as px

fig = px.histogram(data_ring_insight, x="Step_count", title='Step_count_from_7_9',
fig.show()

# Adjust the easy step_count for this graph.
```



Detail on Step_Count and Mood_Level

In [20]:
```python
import plotly.express as px

# N
df = px.data.tips()
```

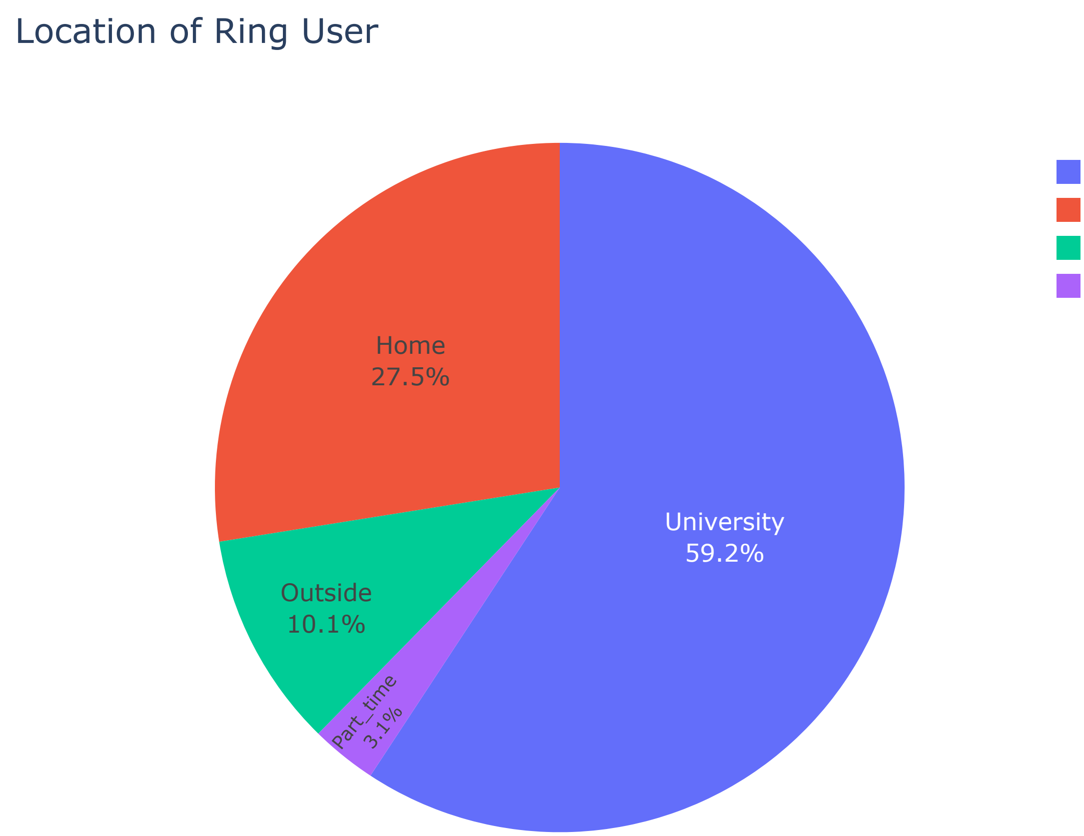
```
fig = px.histogram(data_ring_insight, x="MM_level", text_auto=True, title="Step_cou
fig.show()

# make the clear graph - intention, pattern note
```



Percentage of use location

In [21]:

```
# Analysis to add 2 to 4 collumns.
# Add social media consumption time.
fig = px.pie(data_ring_insight, values='Step_count', names='location',
             title='Location of Ring User',
             hover_data=['Step_count'], labels={'lifeExp':'life expectancy'})
fig.update_traces(textposition='inside', textinfo='percent+label')
fig.show()
```



Number Outlook

In [23]:
```
## Step Count
total_step_F1 = data_ring_insight.groupby(by='Device_ID', as_index=False)['Step_cou
total_step_F1.columns = ['Device_ID', 'step_count_df1_F1']
total_step_F1.describe()
```

Out[23]:

	step_count_df1_F1
count	1.0
mean	62615.0
std	NaN
min	62615.0
25%	62615.0
50%	62615.0
75%	62615.0
max	62615.0

Histogram

- How many times each value appears in dataset.
- This description is called the distribution of variable
- Most common way to represent distribution of varible is histogram that is graph which shows frequency of each value.
- Frequency = number of times each value appears
- Example: [1,1,1,1,2,2,2]. Frequency of 1 is four and frequency of 2 is three.

In [24]:

```
df = data_ring_insight.copy()
```

We are having a specicial so high MM_level in this graph.

Feature relationship

Comprarison between different feature SCR/Min, SCL, Step Count.

We took the importance of step count into the main core of analysis. Due to the interesting of the client. Step-count seem tobe the most corellation level in this coorellation chart.

As we can see they are the most correlation and Not show many meaning level. Therefore we have to dive deep into different way to understand data interpretation.

Explore the Weather Data.

Weather effective to mental health

https://www.verywellmind.com/how-weather-changes-can-affect-your-mental-health-5222029

In [25]:
```
from knmy import knmy

disclaimer, stations, variables, data_wt_ring = knmy.get_hourly_data(stations=[370]
                                                            inseason=True, variabl

data_wt_ring = data_wt_ring.rename(columns={"STN":"Station_Code","H":"Time_Strip",

data_wt_ring = data_wt_ring[["Station_Code","Datetime","Wind_Speed","Tempurature","

data_wt_ring["Tempurature"] = data_wt_ring["Tempurature"] / 10

data_wt_ring
# data_wt_ring
```

Out[25]:

	Station_Code	Datetime	Wind_Speed	Tempurature	Time_Strip	Precitipation
0	370	20210101	10	2.0	1	0
1	370	20210101	10	1.8	2	0
2	370	20210101	10	1.1	3	0
3	370	20210101	20	1.0	4	0
4	370	20210101	20	0.3	5	0
...	...	...	...	...	...	...
13771	370	20221014	10	11.8	20	0
13772	370	20221014	10	10.8	21	0
13773	370	20221014	10	11.4	22	0
13774	370	20221014	10	11.4	23	0
13775	370	20221014	10	10.0	24	0

13776 rows × 6 columns

Apply the weather API from KMNI Service. We are collected the range of the data from 2021 - 2022. With the aim to merge the dataset weather to normal ring data. Therefore the output will be the data + weather data, it support the progress to figure the right insight which is the main key factor from weather might give the significant influence to the user Mood Level.

Data Problem + Solution:

There is an unsual of tempureture key data which hard to explain, now we decided to bring the back to normal celcious tempt. Which will make the right sense of logic by divided the actually tempt collumn to 10. Then, we have the right temperature data column in Celcious degree.

In [26]:
```
# # Return dataframe with hourly wind and temperature data for station 370 (Eindhov
# # til 1th of January of the years 2021 til 2022 for the 18th til 24th hour of the
disclaimer, stations, variables, data = knmy.get_hourly_data(stations=[370], start=
                                                             inseason=True, variabl

weather_data_all = data.rename(columns={"STN":"Station_Code","H":"Time_Strip", "YYY

weather_data_all = weather_data_all[["Station_Code","Datetime","Wind_Speed","Tempur
weather_data_all["Tempurature"] = weather_data_all["Tempurature"] / 10
```

In [27]:
```
frames = [df_F1, df_F2, df_M2, df_M3, df_U1]

result_all = pd.concat(frames)

result_all['Time_New'] = result_all['Time_ISO'].astype(str).str[11:16]
result_all['Time_New'] = result_all['Time_New'].str.replace(':','.')
result_all['Time_New'] = result_all['Time_New'].astype(float)

result_all['Datetime'] = result_all['Time_ISO'].astype(str).str[0:10]
result_all['Time_Strip'] = result_all['Time_New'].astype(int)
result_all['Datetime'] = result_all['Datetime'].str.replace("-", "")
result_all[['Datetime','Time_Strip']] = result_all[['Datetime','Time_Strip']].astyp

result_all = pd.merge(weather_data_all, result_all, how="right", on=['Datetime','Ti

result_all
```

Out[27]:

	Station_Code	Datetime	Wind_Speed	Tempurature	Time_Strip	Precitipation	User_ID	
0	370.0	20210430	30.0	11.0	11	0.0	2049	57c
1	370.0	20210430	30.0	11.0	11	0.0	2049	57c
2	370.0	20210430	30.0	11.0	11	0.0	2049	57c
3	370.0	20210430	30.0	11.0	11	0.0	2049	57c
4	370.0	20210430	30.0	11.0	11	0.0	2049	57c
...	...	...	...	...	...	...	...	
78442	370.0	20220601	40.0	15.3	9	0.0	2542	
78443	370.0	20220601	40.0	15.3	9	0.0	2542	
78444	370.0	20220601	40.0	15.3	9	0.0	2542	
78445	370.0	20220601	50.0	16.5	10	0.0	2542	
78446	370.0	20220601	50.0	16.5	10	0.0	2542	

78447 rows × 17 columns

After receive the weather data, some of feature of Ring Data that need to be shift to feature engineering part to have the right key collumn which able to merge as the [right join] with the weather dataset.

In [28]:

```
data_ring_insight['Datetime'] = data_ring_insight['Time_ISO'].astype(str).str[0:10]
data_ring_insight['Time_Strip'] = data_ring_insight['Time_New'].astype(int)
data_ring_insight['Datetime'] = data_ring_insight['Datetime'].str.replace("-", "")
data_ring_insight[['Datetime','Time_Strip']] = data_ring_insight[['Datetime','Time_

result = pd.merge(data_wt_ring, data_ring_insight, how="right", on=['Datetime','Tim
result
```

Out[28]:

	Station_Code	Datetime	Wind_Speed	Tempurature	Time_Strip	Precitipation	Dev
0	370.0	20210430	30.0	11.0	11	0.0	57dee7ab79b
1	370.0	20210430	30.0	11.0	11	0.0	57dee7ab79b
2	370.0	20210430	30.0	11.0	11	0.0	57dee7ab79b
3	370.0	20210430	30.0	11.0	11	0.0	57dee7ab79b
4	370.0	20210430	30.0	11.0	11	0.0	57dee7ab79b
...	...	...	...	...	...	...	
22074	370.0	20210516	40.0	12.4	14	6.0	57dee7ab79b
22075	370.0	20210516	40.0	12.4	14	6.0	57dee7ab79b
22076	370.0	20210516	40.0	12.4	14	6.0	57dee7ab79b
22077	370.0	20210516	40.0	12.4	14	6.0	57dee7ab79b
22078	370.0	20210516	40.0	12.4	14	6.0	57dee7ab79b

22079 rows × 17 columns

Visulization in Advanced.

Feature Analysis

In [29]:

```
df = result.copy()
df.dtypes
```

Out[29]:
```
Station_Code         float64
Datetime               int64
Wind_Speed           float64
Tempuratur e         float64
Time_Strip             int64
Precitipation        float64
Device_ID             object
Ring_ID               object
MM_level             float64
SCR/min                int64
SCL                  float64
Step_count             int64
Calibration_Value      int64
Time_ISO              object
Time_UNIX              int64
Time_New             float64
location              object
dtype: object
```

Linear regession to analysis

Transfer the right feature and sort them out to have the right logic for correlation.

In [30]:
```
df_num = df.select_dtypes(include = ['float64', 'int64'])

df_num_corr = df_num.corr()['MM_level'][:-1] # -1 because the latest row is SalePri
golden_features_list = df_num_corr[abs(df_num_corr) > 0.1].sort_values(ascending=Fa
df_num
```

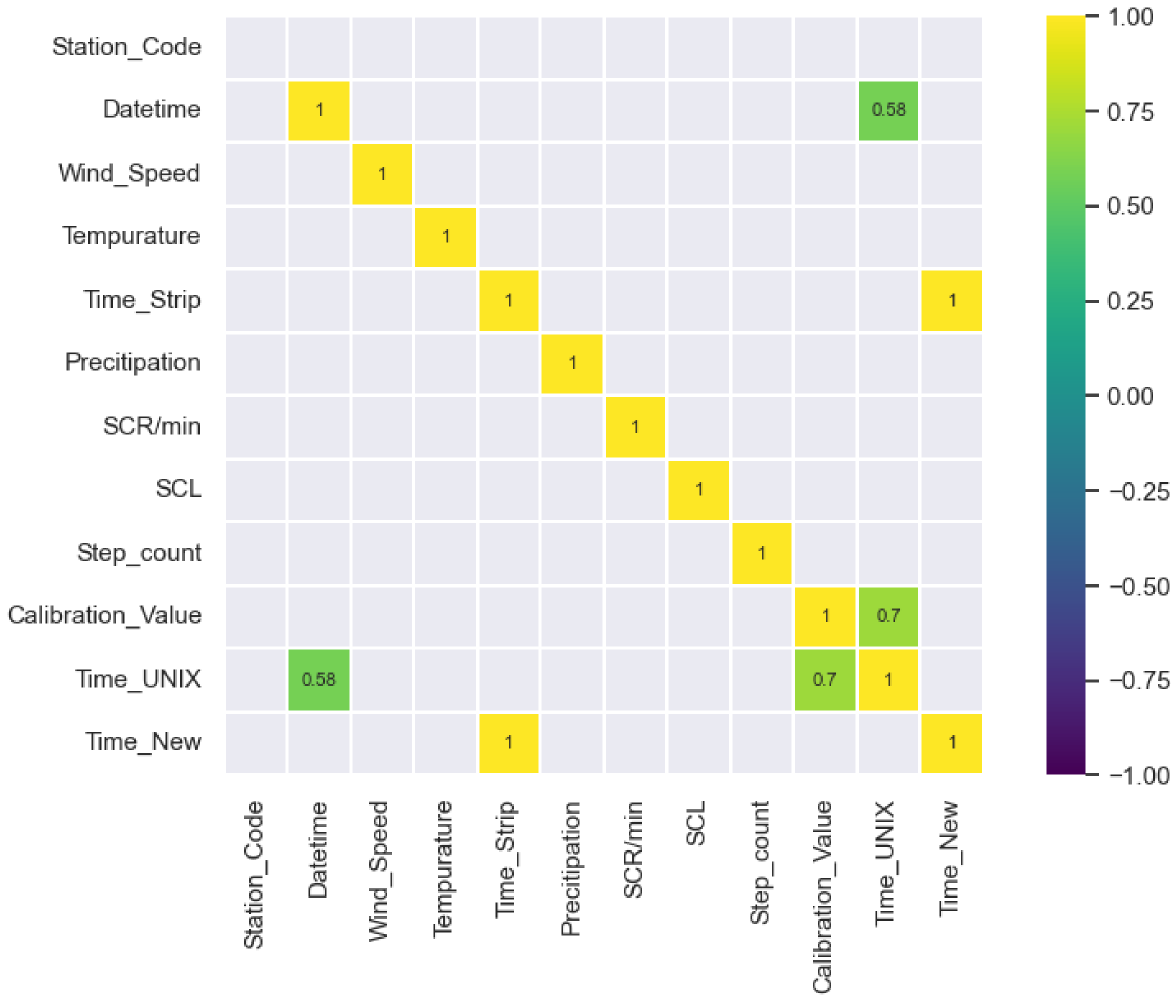
Out[30]:

	Station_Code	Datetime	Wind_Speed	Tempurature	Time_Strip	Precitipation	MM_level	S
0	370.0	20210430	30.0	11.0	11	0.0	41.340782	
1	370.0	20210430	30.0	11.0	11	0.0	39.106145	
2	370.0	20210430	30.0	11.0	11	0.0	48.044693	
3	370.0	20210430	30.0	11.0	11	0.0	53.631285	
4	370.0	20210430	30.0	11.0	11	0.0	51.396648	
...	...	...	...	...	...	...	...	
22074	370.0	20210516	40.0	12.4	14	6.0	71.140940	
22075	370.0	20210516	40.0	12.4	14	6.0	72.483221	
22076	370.0	20210516	40.0	12.4	14	6.0	73.825503	
22077	370.0	20210516	40.0	12.4	14	6.0	64.429530	
22078	370.0	20210516	40.0	12.4	14	6.0	52.348993	

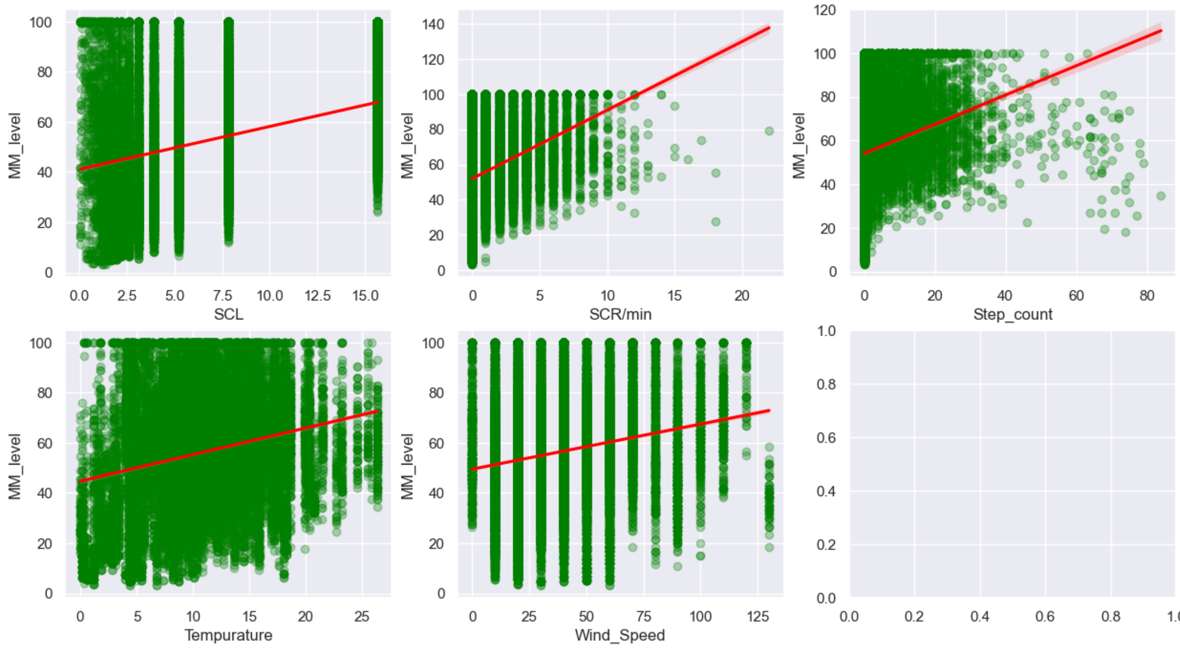
22079 rows × 13 columns

In [31]:
```
corr = df_num.drop('MM_level', axis=1).corr() # We already examined SalePrice corre
plt.figure(figsize=(10, 6))
```

```python
sns.heatmap(corr[(corr >= 0.5) | (corr <= -0.4)],
            cmap='viridis', vmax=1.0, vmin=-1.0, linewidths=0.1,
            annot=True, annot_kws={"size": 8}, square=True);

# Turn to more advance the comment detail of the correlation.
```



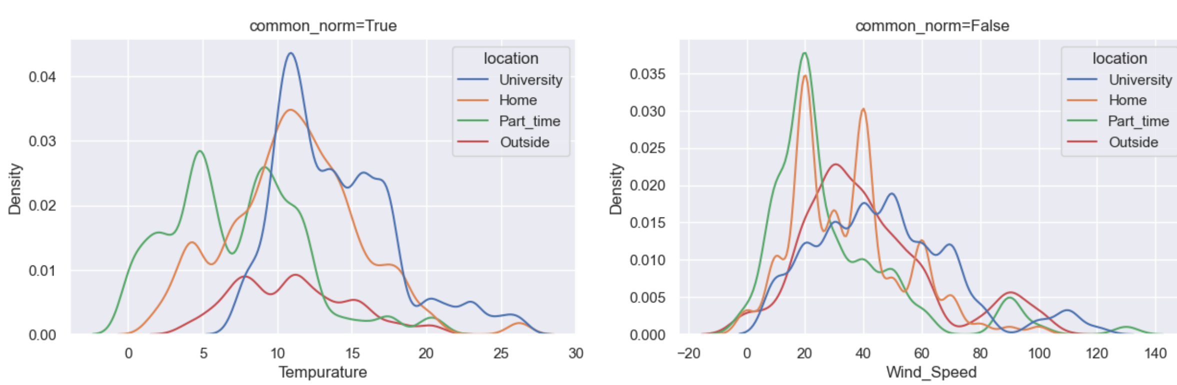
In [32]:
```python
quantitative_features_list = ['SCL', 'SCR/min', 'Step_count','Calibration_Value','T
df_quantitative_values = df[quantitative_features_list]
df_quantitative_values.head(100)
# Create the analytic for the Chart following the MM Level comparison.

features_to_analyse = [x for x in quantitative_features_list if x in golden_feature
features_to_analyse.append('MM_level')
features_to_analyse


fig, ax = plt.subplots(round(len(features_to_analyse) / 3), 3, figsize = (15, 8))

for i, ax in enumerate(fig.axes):
    if i < len(features_to_analyse) - 1:
        sns.regplot(x=features_to_analyse[i],y='MM_level',scatter_kws={"color": "gr
```



Detail Analysis With Weather Data

Kdeplot() is a Kernel Distribution Estimation Plot which depicts the probability density function of the continuous or non-parametric data variables i.e. we can plot for the univariate or multiple variables altogether. Using the Python Seaborn module, we can build the Kdeplot with various functionality added to it.

In [33]:
```
fig, ax = plt.subplots(1, 2, figsize=(15, 4))

sns.kdeplot(data=df, x='Tempurature', hue='location', shade=False,
            ax=ax[0])
ax[0].set_title(f"common_norm=True")
sns.kdeplot(data=df, x='Wind_Speed', hue='location', shade=False,
            common_norm=False, ax=ax[1])
ax[1].set_title(f"common_norm=False");
```

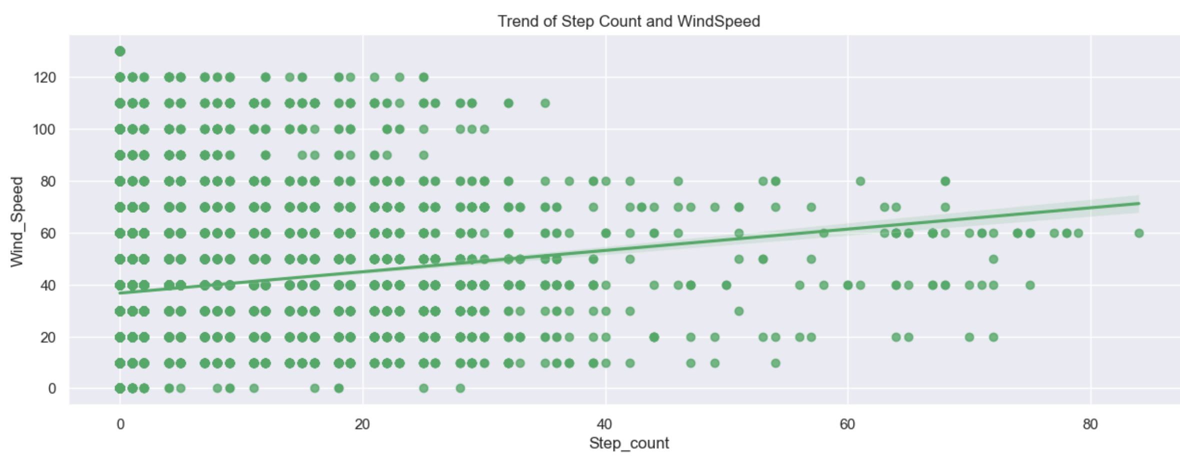


This plot show the different range of temperature and windspend level happend in multiple different location that the user are accutually locate. We discover there is a strong impact of tempurature in part-time job it generate around 2 spike and university. Proparly working in the university is crucial key impact to people. However these temperature are thing happend from the outsite, not indoor internally.

Analysis Regplot

Regplot() : This method is used to plot data and a **Linear Regression** model fit. There are a number of mutually exclusive options for estimating the regression model. For more information click here

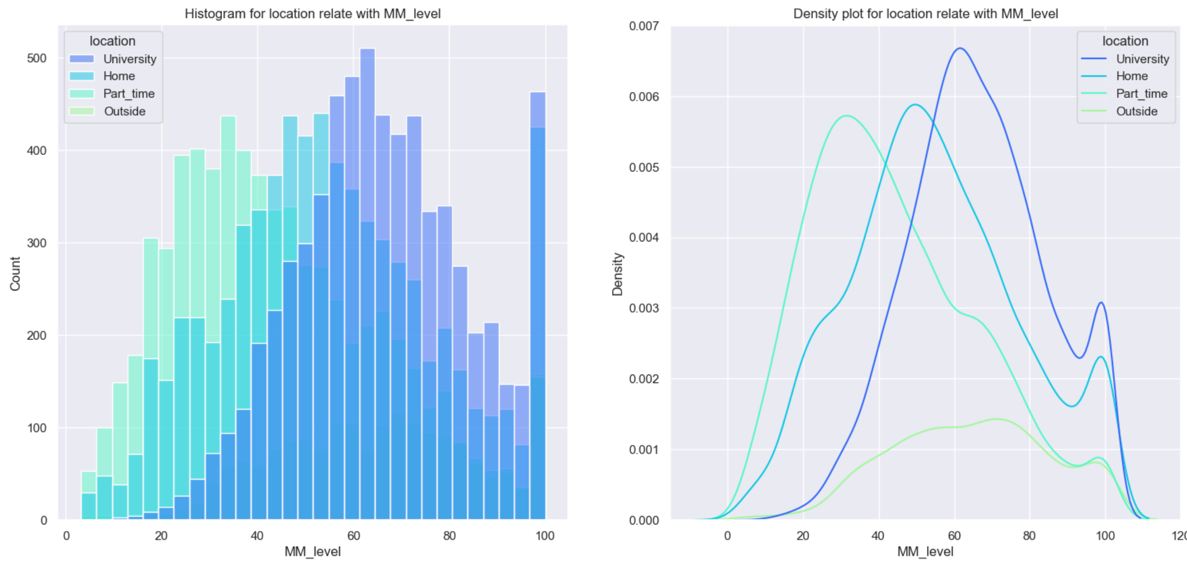
In [34]:
```
plt.figure(figsize=(15,5))

sns.set_palette('rainbow')

sns.regplot(x=df["Step_count"],y=df.Wind_Speed,color="g")

plt.title("Trend of Step Count and WindSpeed")
plt.show()
```



Recognized the linear line and demonstrated the independence between step count and wind speed. This diagram shows how the variables Wind spped and Step count relate to one another. From the starting position to the upper point, I can see the positive trend continuing. The linear relaxation increases after the step count. Which suggests that more people may have stepped into genuine generator activity.

In [35]:
```
numerical = list(result.select_dtypes('number').columns)


fig, ax = plt.subplots(1, 2, figsize=(18, 8))
sns.histplot(result, x='MM_level', hue='location', bins=30, ax=ax[0])
ax[0].set_title(f"Histogram for location relate with MM_level")
sns.kdeplot(data=result, x='MM_level', hue='location', fill=False,
            common_norm=True, ax=ax[1])
ax[1].set_title(f"Density plot for location relate with MM_level")
```

Out[35]:
```
Text(0.5, 1.0, 'Density plot for location relate with MM_level')
```

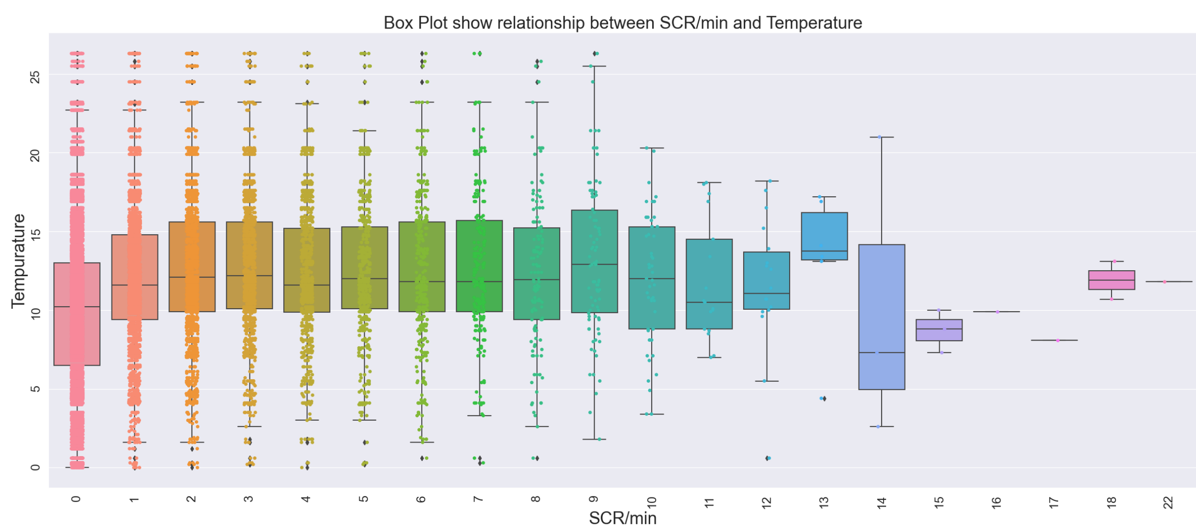


Histogram Plot Advanced

Anayze the trend of SCR/min and Temperature with Histogram Technique

In [36]:
```
fig, ax = plt.subplots(figsize=(30, 12))
ax = sns.boxplot(x="SCR/min", y = "Tempurature", data=result)
ax.tick_params(rotation=90, labelsize=18)
ax = sns.stripplot(x = "SCR/min", y = "Tempurature", data=result)
ax.set_title('Box Plot show relationship between SCR/min and Temperature'"", fontsi
ax.set_xlabel('SCR/min', fontsize=25)
ax.set_ylabel('Tempurature', fontsize=25)
```

Out[36]:
```
Text(0, 0.5, 'Tempurature')
```

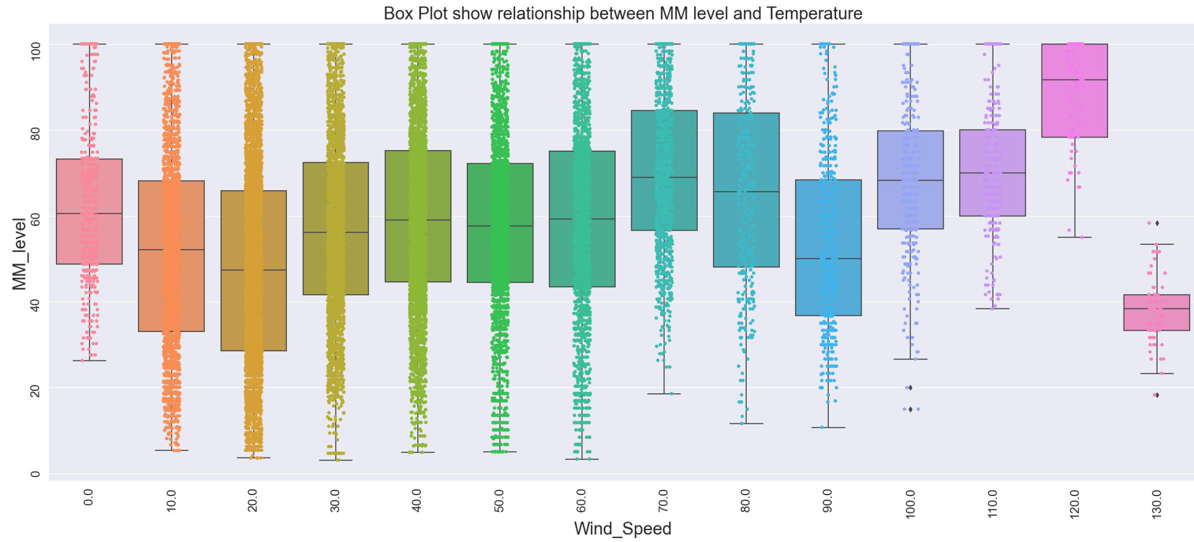


Anayze the trend of Wind_Speed and Temperature with Histogram Technique

In [37]:
```
fig, ax = plt.subplots(figsize=(30, 12))
ax = sns.boxplot(x="Wind_Speed", y = "MM_level", data=result)
ax.tick_params(rotation=90, labelsize=18)
ax = sns.stripplot(x = "Wind_Speed", y = "MM_level", data=result)
```

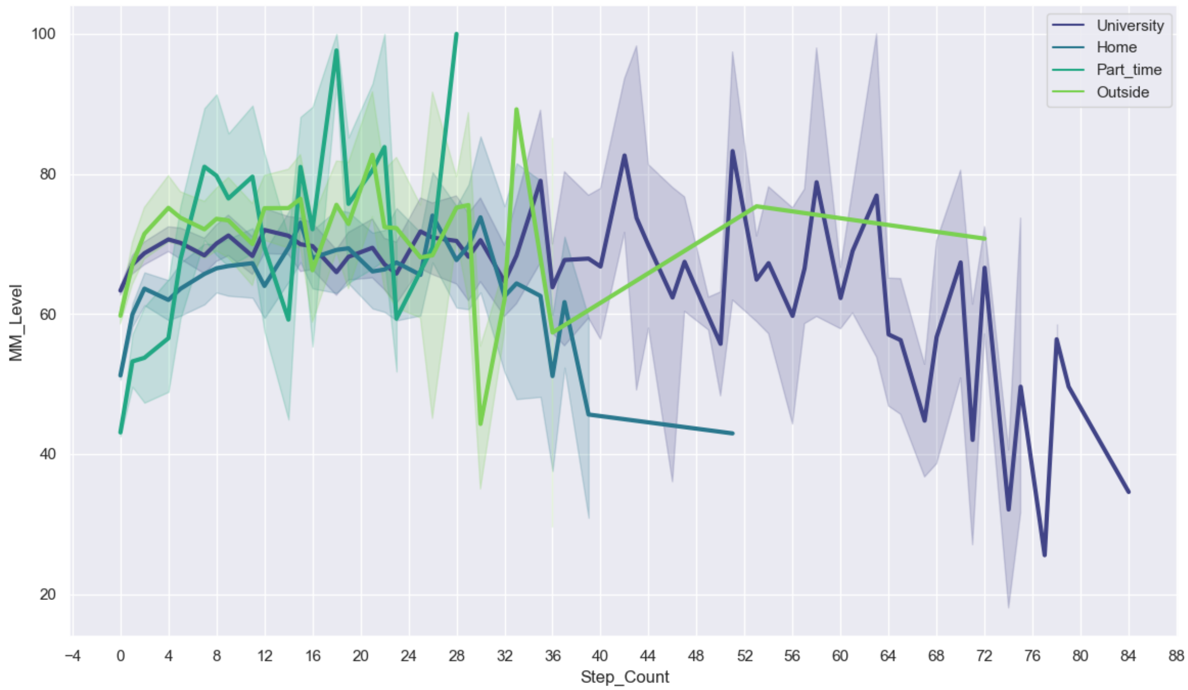
```
ax.set_title('Box Plot show relationship between MM level and Temperature', fontsiz
ax.set_xlabel('Wind_Speed', fontsize=25)
ax.set_ylabel('MM_level', fontsize=25)
```

Out[37]: Text(0, 0.5, 'MM_level')



At the highest point of "Wind speed" we can see. There will be additional tension at level, for example, a bump at 120. However, there are more rows that appear between 10 and 50. As a result, the Win Speed scale, which ranges from 20 to 60, is an excellent way to gauge the density of stress.

In [38]:
```
import matplotlib.ticker as ticker

sns.set_style('darkgrid')
sns.set(rc={'figure.figsize':(14,8)})

ax = sns.lineplot(data=result, x ='Step_count', y = 'MM_level',
                  hue='location', palette='viridis',
                  legend='full', lw=3)

ax.xaxis.set_major_locator(ticker.MultipleLocator(4))
plt.legend(bbox_to_anchor=(1, 1))
plt.ylabel('MM_Level')
plt.xlabel('Step_Count')
plt.show()
```



From the "**step-count**", we can see that the peak downtrend is somewhat declining as university period of time. In order to make it clear to the reader why there aren't as many step-count as total in this graph, it's important to note that step-count is the number of a single row and that they won't show the total sum of "**step-count**" in this area.

We can proceed directly while doing a part-time job and having one high peak on a 100 rapidly. It indicates that the user might engage in some activities that cause significant levels of stress at a "**part-time**" job. There is no point where something is normal and maintains its stability. It implies that emotional swings in mood are constant, and some tension was evident in the external surroundings such as "**out-site**" environment.

Create the Pivot Dataframe.

In [40]:

```
# Just recopy the dataframe to see the time strip
df = result.copy()
df['Time_Strip'] = df['Time_Strip'].replace(0,24)

data_df_heat_map = result.pivot_table(
    index='Step_count').fillna(0)

# total_step_F1.columns = ['Device_ID', 'step_count_df1_F1']
# Select columns to use, optionally subset or use relative numbers
#data_df_heat_map['total'] = data_df_heat_map[data_df_heat_map.columns[0:24]].sum(a
data_df_heat_map = data_df_heat_map[["Wind_Speed","Tempurature","Precitipation","SC

# setting on the relatie growth numbers
# data_df_heat_map = data_df_heat_map / data_df_heat_map.shift()

data_df_heat_map.tail(5).loc[::-1].transpose()
# Show the tail of the data.
```

```
data_df_heat_map

# swip pivot the data.
```

```
C:\Users\Mark-Nguyen\AppData\Local\Temp\ipykernel_32504\3590564755.py:5: FutureWar
ning:

pivot_table dropped a column because it failed to aggregate. This behavior is depr
ecated and will raise in a future version of pandas. Select only the columns that
can be aggregated.
```

Out[40]:

	Wind_Speed	Tempurature	Precitipation	SCR/min	MM_level
Step_count					
0	34.722396	10.092030	0.577166	0.551540	50.867637
1	42.534653	12.390561	0.519472	1.601307	63.362456
2	45.319149	13.031131	0.419933	1.832776	66.446238
4	46.461794	12.796512	0.641196	1.978405	67.850023
5	46.331096	12.700224	0.684564	1.984444	68.864179
...	...	...	...	...	...
75	53.333333	16.066667	0.000000	0.000000	49.624060
77	60.000000	18.100000	0.000000	0.000000	25.563910
78	60.000000	19.150000	0.000000	1.500000	56.390977
79	60.000000	20.200000	0.000000	0.000000	49.624060
84	60.000000	20.200000	0.000000	0.000000	34.586466

59 rows × 5 columns

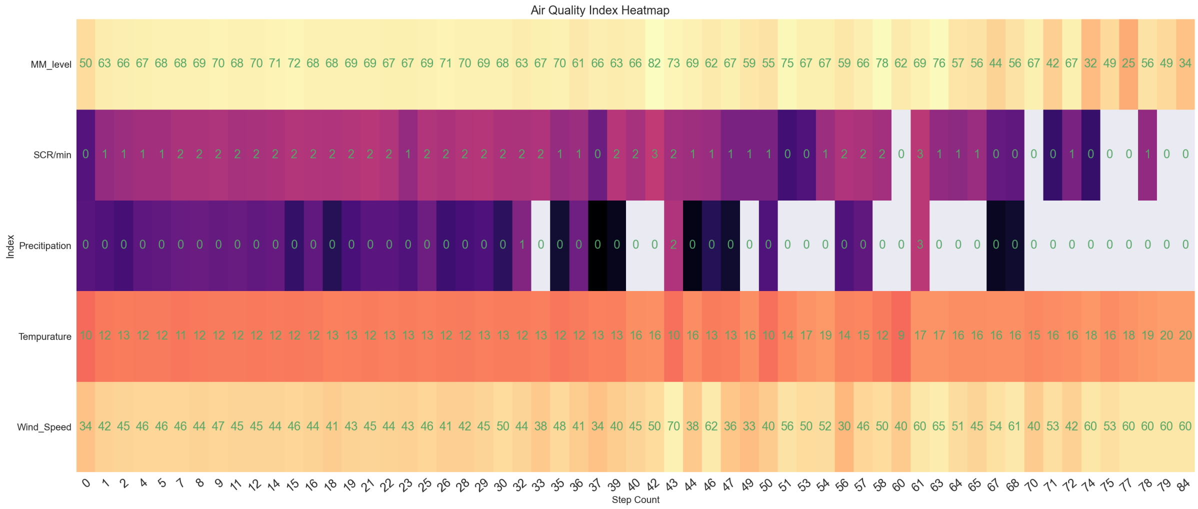
Advanced Heatmap

A heatmap is a graphical representation of data that uses a system of color-coding to represent different values. Heatmaps are used in various forms of analytics but are most commonly used to show user behavior on specific webpages or webpage templates.

In [41]:

```
import matplotlib

## Define array of row and columns headers
durationsperday = data_df_heat_map.index
air_index = data_df_heat_map.columns

plt.rcParams['axes.grid'] = False
## Output size to modified with data size and length
fig, ax = plt.subplots(figsize=(35,15))

heatmap = plt.imshow(
```

```
    np.log(data_df_heat_map[data_df_heat_map > 0].loc[:].transpose()),
    cmap='magma',
    interpolation='None',
    aspect='auto',
    origin='lower')

# Value add to be axis tick label
ax.set_xticks(np.arange(len(durationsperday)))
ax.set_yticks(np.arange(len(air_index)))
# set the lable for duration per day, and air_index.
ax.set_xticklabels(durationsperday)
ax.set_yticklabels(air_index)

# X labels diagonally
plt.setp(
    ax.get_xticklabels(),
    rotation=40,
    ha="right",
    rotation_mode="anchor",
    size=25)

# Y labels diagonally
plt.setp(
    ax.get_yticklabels(),
    size=20)

# Convert dataframe to numpy dataframe
np_heat = data_df_heat_map.to_numpy()

# Set numbers as text labeles
for i in range(len(durationsperday)):
    for j in range(len(air_index)):
        text = ax.text(
            i,
            j,
            int(np_heat[i, j]),
            ha="center",
            va="center",
            color="g",
            size=25)

# ax.set_title("Positive tests weekly, on sex and age group")
fig.suptitle('Air Quality Index Heatmap', fontsize=25)
plt.xlabel('Step Count', fontsize=20)
plt.ylabel('Index', fontsize=21)
fig.tight_layout()
plt.show()
```



The final graph displays the effects of the various features. As we can see, the precitation has the least effect of all. We would say that temperature has the greatest influence on data features, as it affects all different ratios, including SCL. Calibration will also change whether the present temperature is kept constant or drastically modified, and Win-Speed is the second factor that influences other data columns.

Last Words in Interpretive MoodRing Data

Data may not always be accurate, especially for various "genotypes". Investigate the details, discern the ideal location, and develop the ability to discern how others perceive situations. Highly recommend caring for those who are truly willing to labor for a long time when working with rings. Activities are a significant additional factor that may have an impact on the feat, in addition to the weather. All of these graphs show some statistical data that we believe may be useful to future experts in helping other autistic children. The normal folks we selected as users will differ from some specific users when they have children.

However, expressing the opinion on how to improve data is exponentially committed to a future project. This is our accomplishment during the project.