```
import pandas as pd
import numpy as np
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df = pd.read_csv("/content/drive/MyDrive/Timehacks/Copy_Timehacks_synchronize2")
```

```
# df.resample('5T')
df['timestamp']=pd.to_datetime(df['timestamp'], format='%Y-%m-%d %H:%M:%S %Z')
```

```
df
```

	timestamp	device_id	latitude	longitude	temperature	humidity	pm10	pm2_5
0	2023-01-15 06:18:00+00:00	3424	20.239200	85.76710	28.60	73.10	59.00	44.00
1	2023-12-02 06:55:00+00:00	3424	20.239200	85.76710	33.06	40.61	135.00	128.00
2	2023-12-02 14:38:00+00:00	3424	20.239200	85.76710	30.53	49.78	127.00	114.00
3	2023-01-12 19:40:00+00:00	3424	20.239200	85.76710	21.50	99.90	89.00	66.00
4	2023-01-12 20:22:00+00:00	3424	20.239200	85.76710	21.40	99.90	76.00	57.00
...	...	...	...	...	...	...	...	...
7239869	2023-11-24 15:24:00+00:00	11905	20.295448	85.83895	25.63	47.95	229.10	215.25
7239870	2023-11-24 01:31:00+00:00	11905	20.295448	85.83895	21.77	59.92	266.57	244.60
7239871	2023-11-24 08:02:00+00:00	11905	20.295448	85.83895	28.54	35.08	151.73	142.93
7239872	2023-12-18 02:09:00+00:00	11905	20.295448	85.83895	18.83	50.08	250.33	229.39
7239873	2023-12-19 22:05:00+00:00	11905	20.295448	85.83895	19.06	51.55	290.75	265.20

7239874 rows × 8 columns

```
df_copy = df
```

```
df_copy
```

	timestamp	device_id	latitude	longitude	temperature	humidity	pm10	pm2_5
0	2023-01-15 06:18:00+00:00	3424	20.239200	85.76710	28.60	73.10	59.00	44.00
1	2023-12-02 06:55:00+00:00	3424	20.239200	85.76710	33.06	40.61	135.00	128.00
2	2023-12-02 14:38:00+00:00	3424	20.239200	85.76710	30.53	49.78	127.00	114.00
3	2023-01-12 19:40:00+00:00	3424	20.239200	85.76710	21.50	99.90	89.00	66.00
4	2023-01-12 20:22:00+00:00	3424	20.239200	85.76710	21.40	99.90	76.00	57.00
...	...	...	...	...	...	...	...	...
7239869	2023-11-24 15:24:00+00:00	11905	20.295448	85.83895	25.63	47.95	229.10	215.25
7239870	2023-11-24 01:31:00+00:00	11905	20.295448	85.83895	21.77	59.92	266.57	244.60
7239871	2023-11-24 08:02:00+00:00	11905	20.295448	85.83895	28.54	35.08	151.73	142.93
7239872	2023-12-18 02:09:00+00:00	11905	20.295448	85.83895	18.83	50.08	250.33	229.39
7239873	2023-12-19 22:05:00+00:00	11905	20.295448	85.83895	19.06	51.55	290.75	265.20

7239874 rows × 8 columns

```
df_temp = df.drop(['humidity','pm10','pm2_5'],axis=1)
```

```
df_humid = df.drop(['temperature','pm10','pm2_5'],axis=1)
```

```
df_pm10 = df.drop(['temperature','humidity','pm2_5'],axis=1)
```

```
df_pm2_5= df.drop(['temperature','humidity','pm10'],axis=1)
```

```
df_sorted = df_copy.sort_values('timestamp')
```

```
df_sorted
```

	timestamp	device_id	latitude	longitude	temperature	humidity	pm10	pm2_5	time	date
307449	2023-01-01 00:00:00+00:00	3425	20.283834	85.766691	26.60	99.90	138.00	103.00	00:00:00	2023-01-01
2691731	2023-01-01 00:00:00+00:00	3413	20.204940	85.800600	22.10	83.80	126.00	93.00	00:00:00	2023-01-01
2927841	2023-01-01 00:00:00+00:00	3405	20.287117	85.859258	22.10	99.90	187.00	138.00	00:00:00	2023-01-01
2289848	2023-01-01 00:00:00+00:00	3422	20.256570	85.838950	24.20	99.90	131.00	96.00	00:00:00	2023-01-01
3610998	2023-01-01 00:00:00+00:00	3412	20.292080	85.742040	21.60	90.10	239.00	147.00	00:00:00	2023-01-01
...	...	...	...	...	...	...	...	...	...	...
6439980	2023-12-31 23:59:00+00:00	3414	20.319892	85.827658	26.20	48.05	203.93	183.57	23:59:00	2023-12-31
5811566	2023-12-31 23:59:00+00:00	3410	20.336274	85.804951	23.96	55.31	186.68	168.36	23:59:00	2023-12-31
2232287	2023-12-31 23:59:00+00:00	3424	20.239200	85.767100	23.88	55.97	175.42	163.92	23:59:00	2023-12-31
3044841	2023-12-31 23:59:00+00:00	3409	20.243168	85.786551	23.59	58.53	213.83	203.33	23:59:00	2023-12-31
4733718	2023-12-31 23:59:00+00:00	3419	20.363750	85.797380	17.73	78.30	70.12	52.50	23:59:00	2023-12-31

7239874 rows × 10 columns

```
df_sorted['time']=pd.to_datetime(df['timestamp']).dt.time
df_sorted['date']=pd.to_datetime(df['timestamp']).dt.date
```

```
df_temp['time']=pd.to_datetime(df['timestamp']).dt.time
df_temp['date']=pd.to_datetime(df['timestamp']).dt.date
```

```
df_temp=df_temp.sort_values('timestamp')

```

```
df_temp.head()
```

	timestamp	device_id	latitude	longitude	temperature	temp_change	time	date
307449	2023-01-01 00:00:00+00:00	3425	20.283834	85.766691	26.6	1	00:00:00	2023-01-01
2691731	2023-01-01 00:00:00+00:00	3413	20.204940	85.800600	22.1	2	00:00:00	2023-01-01
2927841	2023-01-01 00:00:00+00:00	3405	20.287117	85.859258	22.1	2	00:00:00	2023-01-01
2289848	2023-01-01 00:00:00+00:00	3422	20.256570	85.838950	24.2	3	00:00:00	2023-01-01
3610998	2023-01-01 00:00:00+00:00	3412	20.292080	85.742040	21.6	4	00:00:00	2023-01-01

```
df_3409=df[df['device_id']==3409].sort_values('timestamp')
df_3409
```

	timestamp	device_id	latitude	longitude	temperature	humidity	pm10	pm2_5
2132995	2023-01-01 16:32:00+00:00	3409	20.243168	85.786551	23.70	99.90	183.00	135.00
5783900	2023-01-01 16:33:00+00:00	3409	20.243168	85.786551	23.70	99.90	181.00	133.00
2141761	2023-01-01 16:34:00+00:00	3409	20.243168	85.786551	23.70	99.90	178.00	131.00
629090	2023-01-01 16:35:00+00:00	3409	20.243168	85.786551	23.70	99.90	176.00	130.00
2120705	2023-01-01 16:36:00+00:00	3409	20.243168	85.786551	23.70	99.90	173.00	128.00
...	...	...	...	...	...	...	...	...
4440167	2023-12-31 23:55:00+00:00	3409	20.243168	85.786551	23.59	58.66	215.94	203.06
613118	2023-12-31 23:56:00+00:00	3409	20.243168	85.786551	23.60	58.77	216.28	203.83
1582130	2023-12-31 23:57:00+00:00	3409	20.243168	85.786551	23.61	58.74	215.75	203.75
4444661	2023-12-31 23:58:00+00:00	3409	20.243168	85.786551	23.51	59.21	215.14	203.23
3044841	2023-12-31 23:59:00+00:00	3409	20.243168	85.786551	23.59	58.53	213.83	203.33

483107 rows × 8 columns

```
df_3405=df_temp[df_temp['device_id']==3405]
df_3405
```

	timestamp	device_id	latitude	longitude	temperature	temp_change	time	date
2927841	2023-01-01 00:00:00+00:00	3405	20.287117	85.859258	22.10	2907679	00:00:00	2023-01-01
3376524	2023-01-01 00:01:00+00:00	3405	20.287117	85.859258	22.10	3353352	00:01:00	2023-01-01
2425929	2023-01-01 00:02:00+00:00	3405	20.287117	85.859258	22.10	2409015	00:02:00	2023-01-01
504050	2023-01-01 00:03:00+00:00	3405	20.287117	85.859258	22.20	500730	00:03:00	2023-01-01
1025353	2023-01-01 00:04:00+00:00	3405	20.287117	85.859258	22.20	1018734	00:04:00	2023-01-01
...	...	...	...	...	...	...	...	...
2923694	2023-12-31 23:55:00+00:00	3405	20.287117	85.859258	22.80	2903541	23:55:00	2023-12-31
2926234	2023-12-31 23:56:00+00:00	3405	20.287117	85.859258	22.83	2906075	23:56:00	2023-12-31
999437	2023-12-31 23:57:00+00:00	3405	20.287117	85.859258	22.80	992868	23:57:00	2023-12-31
6141049	2023-12-31 23:58:00+00:00	3405	20.287117	85.859258	22.81	6098686	23:58:00	2023-12-31
2441283	2023-12-31 23:59:00+00:00	3405	20.287117	85.859258	22.81	2424346	23:59:00	2023-12-31

493378 rows × 8 columns

```
df_humid=df_humid.sort_values('timestamp')

```

```
df_3405=df_sorted[df_sorted['device_id']==3405]
df_3405
```

	timestamp	device_id	latitude	longitude	temperature	humidity	pm10	pm2_5	time	date
2927841	2023-01-01 00:00:00+00:00	3405	20.287117	85.859258	22.10	99.90	187.00	138.00	00:00:00	2023-01-01
3376524	2023-01-01 00:01:00+00:00	3405	20.287117	85.859258	22.10	99.90	186.00	138.00	00:01:00	2023-01-01
2425929	2023-01-01 00:02:00+00:00	3405	20.287117	85.859258	22.10	99.90	185.00	137.00	00:02:00	2023-01-01
504050	2023-01-01 00:03:00+00:00	3405	20.287117	85.859258	22.20	99.90	185.00	136.00	00:03:00	2023-01-01
1025353	2023-01-01 00:04:00+00:00	3405	20.287117	85.859258	22.20	99.90	184.00	136.00	00:04:00	2023-01-01
...	...	...	...	...	...	...	...	...	...	...
2923694	2023-12-31 23:55:00+00:00	3405	20.287117	85.859258	22.80	62.35	293.14	276.79	23:55:00	2023-12-31
2926234	2023-12-31 23:56:00+00:00	3405	20.287117	85.859258	22.83	62.23	294.19	277.81	23:56:00	2023-12-31
999437	2023-12-31 23:57:00+00:00	3405	20.287117	85.859258	22.80	62.35	295.06	278.50	23:57:00	2023-12-31
6141049	2023-12-31 23:58:00+00:00	3405	20.287117	85.859258	22.81	62.30	294.85	279.05	23:58:00	2023-12-31
2441283	2023-12-31 23:59:00+00:00	3405	20.287117	85.859258	22.81	62.20	295.50	280.23	23:59:00	2023-12-31

493378 rows × 10 columns

```
df_3405=df_pm10[df_pm10['device_id']==3405]
df_3405
```

```
import pandas as pd

# Assuming df_only_device is already defined and contains 'temperature' and 'date' columns
# df_only_temp=df_only_device.drop(['humidity','pm10','pm2_5'],axis=1)
# Calculate temp_change to identify consecutive temperature readings that are the same
df_3405['temp_change'] = df_3405['temperature'].diff().ne(0).cumsum()
df_3405['humid_change'] = df_3405['humidity'].diff().ne(0).cumsum()
df_3405['pm10_change'] = df_3405['pm10'].diff().ne(0).cumsum()

# Define a function to replace temperatures that are constant for more than 5 minutes with 0
def replace_constant_temps(group):
    # For each group, count the occurrences
    counts = group.groupby('temp_change').size()
    # Find which temp_change groups have counts > 5 (constant for more than 5 minutes)
    to_replace = counts[counts > 5].index.tolist()
    # Replace the temperature values with 0 for those groups
    group.loc[group['temp_change'].isin(to_replace), 'temperature'] = 0
    return group
def replace_constant_humid(group):
    # For each group, count the occurrences
    counts = group.groupby('humid_change').size()
    # Find which temp_change groups have counts > 5 (constant for more than 5 minutes)
    to_replace = counts[counts > 5].index.tolist()
    # Replace the temperature values with 0 for those groups
    group.loc[group['humid_change'].isin(to_replace), 'humidity'] = 0
    return group
def replace_constant_pm10(group):
    # For each group, count the occurrences
    counts = group.groupby('pm10_change').size()
    # Find which temp_change groups have counts > 5 (constant for more than 5 minutes)
    to_replace = counts[counts > 5].index.tolist()
    # Replace the temperature values with 0 for those groups
    group.loc[group['pm10_change'].isin(to_replace), 'temperature'] = 0
    return group
# Apply the function to each date group
modified_data = df_3405.groupby('date').apply(replace_constant_temps)
modified_data = df_3405.groupby('date').apply(replace_constant_humid)
modified_data = df_3405.groupby('date').apply(replace_constant_pm10)

# Optionally, you can drop the 'temp_change' column if it's no longer needed
modified_data.drop(['temp_change'],['humid_change'],['pm10_change'], axis=1, inplace=True)
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_3405['humid_change'] = df_3405['humidity'].diff().ne(0).cumsum()
<ipython-input-52-e06cebe14050>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_3405['pm10_change'] = df_3405['pm10'].diff().ne(0).cumsum()
<ipython-input-52-e06cebe14050>:36: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardle
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  modified_data = df_3405.groupby('date').apply(replace_constant_temps)
<ipython-input-52-e06cebe14050>:37: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardle
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  modified_data = df_3405.groupby('date').apply(replace_constant_humid)
<ipython-input-52-e06cebe14050>:38: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardle
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  modified_data = df_3405.groupby('date').apply(replace_constant_pm10)
<ipython-input-52-e06cebe14050>:41: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  modified_data.drop(['temp_change'],['humid_change'],['pm10_change'], axis=1, inplace=True)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-52-e06cebe14050> in <cell line: 41>()
     39
     40 # Optionally, you can drop the 'temp_change' column if it's no longer needed
---> 41 modified_data.drop(['temp_change'],['humid_change'],['pm10_change'], axis=1, inplace=True)

/usr/local/lib/python3.10/dist-packages/pandas/util/_decorators.py in wrapper(*args, **kwargs)
    329                     stacklevel=find_stack_level(),
    330                 )
--> 331             return func(*args, **kwargs)
    332
    333         # error: "Callable[[VarArg(Any), KwArg(Any)], Any]" has no

TypeError: DataFrame.drop() got multiple values for argument 'axis'
```

SEARCH STACK OVERFLOW

```
import pandas as pd

# Assuming df_3405 is your DataFrame and it contains 'temperature', 'humidity', 'pm10', and 'date' columns

# Calculate change flags for temperature, humidity, and PM10
df_3405['temp_change'] = df_3405['temperature'].diff().ne(0).cumsum()
df_3405['humid_change'] = df_3405['humidity'].diff().ne(0).cumsum()
df_3405['pm10_change'] = df_3405['pm10'].diff().ne(0).cumsum()

# Define functions to replace constant values with 0 for temperature, humidity, and PM10
def replace_constant_temps(group):
    counts = group.groupby('temp_change').size()
    to_replace = counts[counts > 5].index.tolist()
    group.loc[group['temp_change'].isin(to_replace), 'temperature'] = 0
    return group

def replace_constant_humid(group):
    counts = group.groupby('humid_change').size()
    to_replace = counts[counts > 5].index.tolist()
    group.loc[group['humid_change'].isin(to_replace), 'humidity'] = 0
    return group

def replace_constant_pm10(group):
    counts = group.groupby('pm10_change').size()
    to_replace = counts[counts > 5].index.tolist()
    group.loc[group['pm10_change'].isin(to_replace), 'pm10'] = 0
    return group

# Apply the functions to each date group
df_3405 = df_3405.groupby('date').apply(replace_constant_temps)
df_3405 = df_3405.groupby('date').apply(replace_constant_humid)
df_3405 = df_3405.groupby('date').apply(replace_constant_pm10)

# Drop the 'temp_change', 'humid_change', and 'pm10_change' columns
df_3405.drop(['temp_change', 'humid_change', 'pm10_change'], axis=1, inplace=True)

# Now, modified_data correctly reflects the application of all three functions.
modified_data = df_3405

```

```
<ipython-input-55-9e48a5d33dde>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_3405['temp_change'] = df_3405['temperature'].diff().ne(0).cumsum()
<ipython-input-55-9e48a5d33dde>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_3405['humid_change'] = df_3405['humidity'].diff().ne(0).cumsum()
<ipython-input-55-9e48a5d33dde>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_3405['pm10_change'] = df_3405['pm10'].diff().ne(0).cumsum()
<ipython-input-55-9e48a5d33dde>:30: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardless
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  df_3405 = df_3405.groupby('date').apply(replace_constant_temps)
<ipython-input-55-9e48a5d33dde>:31: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardless
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  df_3405 = df_3405.groupby('date').apply(replace_constant_humid)
<ipython-input-55-9e48a5d33dde>:32: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardless
To preserve the previous behavior, use

        >>> .groupby(..., group_keys=False)

To adopt the future behavior and silence this warning, use

        >>> .groupby(..., group_keys=True)
  df_3405 = df_3405.groupby('date').apply(replace_constant_pm10)
```

```
modified_data.head()
```

	timestamp	device_id	latitude	longitude	temperature	humidity	pm10	pm2_5	time	date
2927841	2023-01-01 00:00:00+00:00	3405	20.287117	85.859258	22.1	0.0	187.0	138.0	00:00:00	2023-01-01
3376524	2023-01-01 00:01:00+00:00	3405	20.287117	85.859258	22.1	0.0	186.0	138.0	00:01:00	2023-01-01
2425929	2023-01-01 00:02:00+00:00	3405	20.287117	85.859258	22.1	0.0	185.0	137.0	00:02:00	2023-01-01
504050	2023-01-01 00:03:00+00:00	3405	20.287117	85.859258	22.2	0.0	185.0	136.0	00:03:00	2023-01-01
1025353	2023-01-01 00:04:00+00:00	3405	20.287117	85.859258	22.2	0.0	184.0	136.0	00:04:00	2023-01-01

```
import pandas as pd

# Assuming df_3405 is your DataFrame and it contains 'temperature', 'humidity', 'pm10', 'date', and 'device_id' columns

# Function to apply all the changes for a single device_id
def process_device_data(df):
    df.sort_values(by='timestamp', inplace=True)
    df['temp_change'] = df['temperature'].diff().ne(0).cumsum()
    df['humid_change'] = df['humidity'].diff().ne(0).cumsum()
    df['pm10_change'] = df['pm10'].diff().ne(0).cumsum()
    df['pm2_5_change'] = df['pm2_5'].diff().ne(0).cumsum()
    def replace_constant_temps(group):
      counts = group.groupby('temp_change').size()
      to_replace = counts[counts > 5].index.tolist()
      group.loc[group['temp_change'].isin(to_replace), 'temperature'] = 0
      return group
    def replace_constant_humid(group):
      counts = group.groupby('humid_change').size()
      to_replace = counts[counts > 5].index.tolist()
      group.loc[group['humid_change'].isin(to_replace), 'humidity'] = 0
      return group
    def replace_constant_pm10(group):
      counts = group.groupby('pm10_change').size()
      to_replace = counts[counts > 5].index.tolist()
      group.loc[group['pm10_change'].isin(to_replace), 'pm10'] = 0
      return group
    def replace_constant_pm2_5(group):
      counts = group.groupby('pm2_5_change').size()
      to_replace = counts[counts > 5].index.tolist()
      group.loc[group['pm2_5_change'].isin(to_replace), 'pm2_5'] = 0
      return group

    df = df.groupby('date').apply(replace_constant_temps)
    df = df.groupby('date').apply(replace_constant_humid)
    df = df.groupby('date').apply(replace_constant_pm10)
    df = df.groupby('date').apply(replace_constant_pm2_5)

    # Drop the change tracking columns
    df.drop(['temp_change', 'humid_change', 'pm10_change','pm2_5_change'], axis=1, inplace=True)
    return df

# Separate the DataFrame by device_id and process each separately
list_processed_dfs = []
for device_id, group in df_sorted.groupby('device_id'):
    processed_df = process_device_data(group.copy())
    list_processed_dfs.append(processed_df)

# Concatenate all the processed DataFrames back together
final_df = pd.concat(list_processed_dfs).reset_index(drop=True)

# final_df now contains the processed data for all devices

```

```
      >>> .groupby(..., group_keys=True)
 df = df.groupby('date').apply(replace_constant_humid)
ipython-input-62-c4f4dcf9916d>:35: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardles
o preserve the previous behavior, use

      >>> .groupby(..., group_keys=False)

o adopt the future behavior and silence this warning, use

      >>> .groupby(..., group_keys=True)
 df = df.groupby('date').apply(replace_constant_pm10)
ipython-input-62-c4f4dcf9916d>:36: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardles
o preserve the previous behavior, use

      >>> .groupby(..., group_keys=False)

o adopt the future behavior and silence this warning, use

      >>> .groupby(..., group_keys=True)
 df = df.groupby('date').apply(replace_constant_pm2_5)
ipython-input-62-c4f4dcf9916d>:33: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardles
o preserve the previous behavior, use

      >>> .groupby(..., group_keys=False)

o adopt the future behavior and silence this warning, use

      >>> .groupby(..., group_keys=True)
 df = df.groupby('date').apply(replace_constant_temps)
ipython-input-62-c4f4dcf9916d>:34: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardles
o preserve the previous behavior, use

      >>> .groupby(..., group_keys=False)

o adopt the future behavior and silence this warning, use

      >>> .groupby(..., group_keys=True)
 df = df.groupby('date').apply(replace_constant_humid)
ipython-input-62-c4f4dcf9916d>:35: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardles
o preserve the previous behavior, use

      >>> .groupby(..., group_keys=False)

o adopt the future behavior and silence this warning, use

      >>> .groupby(..., group_keys=True)
 df = df.groupby('date').apply(replace_constant_pm10)
ipython-input-62-c4f4dcf9916d>:36: FutureWarning: Not prepending group keys to the result index of transform-like apply. In the future, the group keys will be included in the index, regardles
o preserve the previous behavior, use

      >>> .groupby(..., group_keys=False)

o adopt the future behavior and silence this warning, use

      >>> .groupby(..., group_keys=True)
 df = df.groupby('date').apply(replace_constant_pm2_5)
```

```
final_df
```

	timestamp	device_id	latitude	longitude	temperature	humidity	pm10	pm2_5	time	date
0	2023-01-01 00:00:00+00:00	3405	20.287117	85.859258	22.10	0.00	187.00	138.00	00:00:00	2023-01-01
1	2023-01-01 00:01:00+00:00	3405	20.287117	85.859258	22.10	0.00	186.00	138.00	00:01:00	2023-01-01
2	2023-01-01 00:02:00+00:00	3405	20.287117	85.859258	22.10	0.00	185.00	137.00	00:02:00	2023-01-01
3	2023-01-01 00:03:00+00:00	3405	20.287117	85.859258	22.20	0.00	185.00	136.00	00:03:00	2023-01-01
4	2023-01-01 00:04:00+00:00	3405	20.287117	85.859258	22.20	0.00	184.00	136.00	00:04:00	2023-01-01
...	...	...	...	...	...	...	...	...	...	...
7239869	2023-12-21 04:53:00+00:00	11905	20.295448	85.838950	20.94	40.68	160.08	146.83	04:53:00	2023-12-21
7239870	2023-12-21 04:54:00+00:00	11905	20.295448	85.838950	21.07	40.51	158.64	144.86	04:54:00	2023-12-21
7239871	2023-12-21 04:55:00+00:00	11905	20.295448	85.838950	21.16	40.81	157.69	143.50	04:55:00	2023-12-21
7239872	2023-12-21 04:56:00+00:00	11905	20.295448	85.838950	21.50	40.10	157.61	143.28	04:56:00	2023-12-21
7239873	2023-12-21 04:57:00+00:00	11905	20.295448	85.838950	21.68	39.36	158.55	143.25	04:57:00	2023-12-21

7239874 rows × 10 columns

```
final_df.to_csv('final.csv')
```

```
import
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import numpy as np

# Load your dataset
data = pd.read_csv('/content/final.csv')  # Replace with your actual file path

```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-521ebfdcad16> in <cell line: 6>()
      4
      5 # Load your dataset
----> 6 data = pd.read_csv('/content/final.csv')  # Replace with your actual file path

NameError: name 'pd' is not defined
```

SEARCH STACK OVERFLOW

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.