In [1]:
```
%pip install pandas
%pip install openpyxl
import pandas as pd
import openpyxl
```

```
Requirement already satisfied: pandas in c:\users\maste\appdata\local\programs\py
thon\python312\lib\site-packages (2.2.3)
Requirement already satisfied: numpy>=1.26.0 in c:\users\maste\appdata\local\prog
rams\python\python312\lib\site-packages (from pandas) (2.1.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\maste\appdata\r
oaming\python\python312\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\maste\appdata\local\progr
ams\python\python312\lib\site-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\maste\appdata\local\pro
grams\python\python312\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\maste\appdata\roaming\python
\python312\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: openpyxl in c:\users\maste\appdata\local\programs
\python\python312\lib\site-packages (3.1.5)
Requirement already satisfied: et-xmlfile in c:\users\maste\appdata\local\program
s\python\python312\lib\site-packages (from openpyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
bikes = pd.read_csv('london_merged.csv')
```

In [3]:
```
bikes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17414 entries, 0 to 17413
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   timestamp     17414 non-null  object
 1   cnt           17414 non-null  int64
 2   t1            17414 non-null  float64
 3   t2            17414 non-null  float64
 4   hum           17414 non-null  float64
 5   wind_speed    17414 non-null  float64
 6   weather_code  17414 non-null  float64
 7   is_holiday    17414 non-null  float64
 8   is_weekend    17414 non-null  float64
 9   season        17414 non-null  float64
dtypes: float64(8), int64(1), object(1)
memory usage: 1.3+ MB
```

In [4]:
```
bikes.shape
```

Out[4]:
```
(17414, 10)
```

In [5]:
```
bikes
```

Out[5]:

	timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weel
0	2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	
1	2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	
2	2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	
3	2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	
4	2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	
...	...	...	...	...	...	...	...	...	
17409	2017-01-03 19:00:00	1042	5.0	1.0	81.0	19.0	3.0	0.0	
17410	2017-01-03 20:00:00	541	5.0	1.0	81.0	21.0	4.0	0.0	
17411	2017-01-03 21:00:00	337	5.5	1.5	78.5	24.0	4.0	0.0	
17412	2017-01-03 22:00:00	224	5.5	1.5	76.0	23.0	4.0	0.0	
17413	2017-01-03 23:00:00	139	5.0	1.0	76.0	22.0	2.0	0.0	

17414 rows × 10 columns

Counting unique values in the weather_code column.

In [6]:
```
bikes.weather_code.value_counts()
```

Out[6]:
```
weather_code
1.0     6150
2.0     4034
3.0     3551
7.0     2141
4.0     1464
26.0      60
10.0      14
Name: count, dtype: int64
```

Counting the unique values in the season column.

In [7]:
```
bikes.season.value_counts()
```

Out[7]:
```
season
0.0    4394
1.0    4387
3.0    4330
2.0    4303
Name: count, dtype: int64
```

Specifying new column names I want to use.

In [8]:
```
new_cols_dict ={
    'timestamp':'time',
    'cnt':'count',
    't1':'temp_real_C',
    't2':'temp_feels_like_C',
    'hum':'humidity_percent',
    'wind_speed':'wind_speed_kph',
    'weather_code':'weather',
    'is_holiday':'is_holiday',
    'is_weekend':'is_weekend',
    'season':'season'
}
bikes.rename(new_cols_dict, axis=1, inplace=True)
```

Changing humidity values to percentage i.e. a value between 0 and 1.

In [9]:
```
bikes.humidity_percent = bikes.humidity_percent / 100
```

Creating a season dictionary so I can map integers 0-3 to the actual written values.

In [10]:
```
season_dict = {
    '0.0':'spring',
    '1.0':'summer',
    '2.0':'autumn',
    '3.0':'winter'
}
```

Creating a weather dictionary so that I can map the integers to the actual written values.

In [11]:
```
weather_dict = {
    '1.0':'Clear',
    '2.0':'Scattered clouds',
    '3.0':'Broken clouds',
    '4.0':'Cloudy',
    '7.0':'Rain',
    '10.0':'Rain with thunderstorm',
    '26.0':'Snowfall'
}
```

Changing seasons and weather columns data type to string and mapping the values to the actual written seasons and weathers.

In [12]:
```
bikes.season = bikes.season.astype('str')
bikes.season = bikes.season.map(season_dict)
bikes.weather = bikes.weather.astype('str')
bikes.weather = bikes.weather.map(weather_dict)
```

In [13]:
```
bikes.head()
```

Out[13]:

	time	count	temp_real_C	temp_feels_like_C	humidity_percent	wind_speed_kph
0	2015-01-04 00:00:00	182	3.0	2.0	0.930	6.0
1	2015-01-04 01:00:00	138	3.0	2.5	0.930	5.0
2	2015-01-04 02:00:00	134	2.5	2.5	0.965	0.0
3	2015-01-04 03:00:00	72	2.0	2.0	1.000	0.0
4	2015-01-04 04:00:00	47	2.0	0.0	0.930	6.5

Writing the final dataframe to an excel file to use in Tableau for visualisation.

In [14]:
```
bikes.to_excel('london_bikes_final.xlsx', sheet_name='Data')
```