Visualization of logged data

Station history enlightens dublin bike users about average availability of bikes through day and over a week.

- Dynamic data scraped from JCDAUX API is downloaded as dBikeD.csv
- This data is operated on to know about weekly and hourly distribution of bike station availability

In [1]:
```
# Import pandas to make and operate on dataframe
import pandas as pd
```

In [2]:
```
def prepareData(para = 'day', dt = False):
    try:

        bikeDynamic = pd.read_csv('dBikeD.csv')
        bikeDynamic = bikeDynamic.drop(columns=['bike_stands','last_update', 'status', 'id_Entry'])

        #Cast data into Datetime format
        bikeDynamic['data_entry_timestamp'] = pd.to_datetime(bikeDynamic['data_entry_timestamp'])

        # limit resolution of datetime objects to minutes
        bikeDynamic['data_entry_timestamp'] = bikeDynamic['data_entry_timestamp'].dt.strftime("%Y-%m-%d %H:%M:00")
        bikeDynamic['data_entry_timestamp'] = pd.to_datetime(bikeDynamic['data_entry_timestamp'])

        # Parameter all ensures that both hour and name of day are taken into a seprate column of dataframe 'bikeDynamic'
        if para == 'all' or para == 'hr':
            bikeDynamic['hour'] = bikeDynamic.data_entry_timestamp.dt.hour
            bikeDynamic['hour'] = bikeDynamic['hour'].astype('int')
        if para == 'all' or para == 'day':
            bikeDynamic['dayOfWeek'] = bikeDynamic.data_entry_timestamp.dt.day_name()

        # flag dt is assigned for future use.
        # If it is True,then user needs data_entry_timestamp in the returned dataframe for further processing hence not dropped
        if not dt:
            bikeDynamic.drop(['data_entry_timestamp'], axis=1)

        return bikeDynamic

    except Exception as e:
        print("Error in prepareData:", e)
```

In [3]:
```
# Support function which returns average bike station avialability of each day
def getWeeklyCount( station = 42, datetimeIn = None):
    try:

        #Retuns a dataframe consisting a column 'dayOfWeek'
        bikeDynamic = prepareData(para = 'day')

        availability = ['available_bikes','available_bike_stands']
        days_name = ['Monday','Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
        result = []

        station_mask = bikeDynamic['number'] == station

        # Generate list of lists consisting name of day and average availability parameters for those weekdays
        for available in availability:
            stationWeeklyCount = pd.DataFrame()
            result.append(days_name)

            for day in days_name:
                day_mask = bikeDynamic['dayOfWeek'] == str(day)
                stationWeeklyCount[day] = bikeDynamic.loc[station_mask & day_mask, available].reset_index(drop=True)

            stationWeeklyCount = stationWeeklyCount.fillna(0)
            averageCount = [int(d) for d in (list(stationWeeklyCount.mean(axis=0)))]
            result.append(averageCount)
        return result

    except Exception as e:
        print("Error in getWeeklyCount:", e)
```

In [4]:
```
# Support function which returns average bike station avialability of each hour
def getHourlyCount( station = 42, datetimeIn = None):
    try:

        #Retuns a dataframe consisting a column 'hour'
        bikeDynamic = prepareData(para = 'hr')

        availability = ['available_bikes','available_bike_stands']
        hours = list(range(0,24))
        result = []

        station_mask = bikeDynamic['number'] == station

        # Generate list of lists consisting name of day and average availability parameters for those weekdays
        for available in availability:
            stationHourlyCount = pd.DataFrame()
            result.append(hours)

            for hr in hours:
                hr_mask = bikeDynamic['hour'] == int(hr)
                stationHourlyCount[hr] = bikeDynamic.loc[station_mask & hr_mask, available].reset_index(drop=True)

            stationHourlyCount = stationHourlyCount.fillna(0)
            averageCount = [int(d) for d in (list(stationHourlyCount.median(axis=0)))]
            result.append(averageCount)

        return result

    except Exception as e:
        print("Error in getHourlyCount:", e)
```

getCount()

Return weekly and hourly averages of bike and stand availability at a station

In [5]:
```
def getCount( station = 42, datetimeIn = None):
    try:

        # Retuns a dataframe consisting a column 'hour' amd 'dayOfWeek'

        bikeDynamic = prepareData(para = 'all')

        availability = ['available_bikes','available_bike_stands']

        days_name = ['Monday','Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
        hours_name = ['12:00 AM', '01:00 AM','02:00 AM','03:00 AM','04:00 AM','05:00 AM','06:00 AM',\
                      '07:00 AM','08:00 AM','09:00 AM','10:00 AM','11:00 AM',\
                    '12:00 PM.','01:00 PM.','02:00 PM.','03:00 PM.','04:00 PM.','05:00 PM.','06:00 PM.',\
                    '07:00 PM.','08:00 PM.','09:00 PM.','10:00 PM.','11:00 PM.']

        hours = list(range(0,24))
        result = []

        station_mask = bikeDynamic['number'] == station

        # Generate list of lists consisting :
        # name of day and average availability parameters for those weekdays
        # name of hour and averape availability parameters for those hours
        for available in availability:

            # Weekdays
            stationWeeklyCount = pd.DataFrame()
            result.append(days_name)

            for day in days_name:
                day_mask = bikeDynamic['dayOfWeek'] == str(day)
                stationWeeklyCount[day] = bikeDynamic.loc[station_mask & day_mask, available].reset_index(drop=True)

            stationWeeklyCount = stationWeeklyCount.fillna(0)
            averageCount = [int(d) for d in (list(stationWeeklyCount.mean(axis=0)))]
            result.append(averageCount)

            # Hours
            stationHourlyCount = pd.DataFrame()
            result.append(hours_name)

            for hr in hours:
                hr_mask = bikeDynamic['hour'] == int(hr)
                stationHourlyCount[hr] = bikeDynamic.loc[station_mask & hr_mask, available].reset_index(drop=True)

            stationHourlyCount = stationHourlyCount.fillna(0)
            averageCount = [int(d) for d in (list(stationHourlyCount.mean(axis=0)))]
            result.append(averageCount)

        return result

    except Exception as e:
        print("Error in getCount:", e)
```

In [6]:
```
getCount()
```

Out[6]:
```
[['Monday',
  'Tuesday',
  'Wednesday',
  'Thursday',
  'Friday',
  'Saturday',
  'Sunday'],
 [17, 16, 16, 14, 17, 17, 22],
 ['12:00 AM',
  '01:00 AM',
  '02:00 AM',
  '03:00 AM',
  '04:00 AM',
  '05:00 AM',
  '06:00 AM',
  '07:00 AM',
  '08:00 AM',
  '09:00 AM',
  '10:00 AM',
  '11:00 AM',
  '12:00 PM.',
  '01:00 PM.',
  '02:00 PM.',
  '03:00 PM.',
  '04:00 PM.',
  '05:00 PM.',
  '06:00 PM.',
  '07:00 PM.',
  '08:00 PM.',
  '09:00 PM.',
  '10:00 PM.',
  '11:00 PM.'],
 [24,
  23,
  24,
  24,
  24,
  23,
  23,
  18,
  14,
  13,
  13,
  12,
  12,
  12,
  13,
  13,
  14,
  16,
  21,
  23,
  24,
  25,
  24,
  23],
 ['Monday',
  'Tuesday',
  'Wednesday',
  'Thursday',
  'Friday',
  'Saturday',
  'Sunday'],
 [12, 9, 7, 9, 9, 6, 7],
 ['12:00 AM',
  '01:00 AM',
  '02:00 AM',
  '03:00 AM',
  '04:00 AM',
  '05:00 AM',
  '06:00 AM',
  '07:00 AM',
  '08:00 AM',
  '09:00 AM',
  '10:00 AM',
  '11:00 AM',
  '12:00 PM.',
  '01:00 PM.',
  '02:00 PM.',
  '03:00 PM.',
  '04:00 PM.',
  '05:00 PM.',
  '06:00 PM.',
  '07:00 PM.',
  '08:00 PM.',
  '09:00 PM.',
  '10:00 PM.',
  '11:00 PM.'],
 [5,
  5,
  5,
  5,
  5,
  5,
  6,
  11,
  14,
  16,
  16,
  17,
  17,
  17,
  16,
  16,
  16,
  15,
  13,
  8,
  6,
  5,
  4,
  5,
  6]]
```

In [ ]:
```

```