In [1]:

```
import pandas as pd
import os, sqlite3
import re
import json
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

1. Create a SQLite database called `my_sqlite.db`

In [2]:

```
db_name = 'my_sqlite.db'

try:
    os.remove(db_name)
except:
    pass
```

In [3]:

```
# start db connection
db_conn = sqlite3.connect(db_name)
c = db_conn.cursor()
```

Reading CSV Files

2. Write SQL to create a table called `time_series` with columns date, device_id, series_type and value which will store the numerical data.

In [4]:

```
c.execute('CREATE TABLE IF NOT EXISTS time_series (date TEXT, device_id INTEGER,series_
db_conn.commit()
```

3. Insert the network traffic series as is into the `time_series` table for each IoT device (you can use any viable series_type descriptor)

In [5]:

```
# code here

path = 'data/'
all_files = os.listdir(path)

csv_files = list(filter(lambda f: f.endswith('.csv'), all_files))

regex = re.compile(r'\d+')

time_series_list = []

for file in csv_files:
    device_id = [int(x) for x in regex.findall(file)]

    df_temp = pd.read_csv(path + "\\" + file)
    df_temp['device_id'] = device_id[0]
    df_temp['series_type'] = 'Network Traffic'

    time_series_list.append(df_temp)
```

```python
time_series = pd.concat(time_series_list,axis=0,ignore_index=True)
time_series = time_series.sort_values(by='device_id').set_index('date')
```

In [6]:

```python
time_series.to_sql('time_series', db_conn, if_exists='replace', index = True)
```

In [7]:

```python
c.execute('''
SELECT * FROM time_series
LIMIT 10
          ''')

for row in c.fetchall():
    print (row)
```

```
('2020-01-02', 99.32285114119422, 4, 'Network Traffic')
('2020-09-07', 105.5295672832182, 4, 'Network Traffic')
('2020-09-06', 105.54748365091464, 4, 'Network Traffic')
('2020-09-05', 101.27449224834427, 4, 'Network Traffic')
('2020-09-04', 101.12942654918751, 4, 'Network Traffic')
('2020-09-03', 103.5393410213001, 4, 'Network Traffic')
('2020-09-02', 101.1135567522263, 4, 'Network Traffic')
('2020-09-01', 101.29715582313536, 4, 'Network Traffic')
('2020-08-31', 107.8811560076593, 4, 'Network Traffic')
('2020-08-30', 95.12485829238466, 4, 'Network Traffic')
```

Reading meta file

4. Write SQL to create a table called `meta` with columns id and name

In [8]:

```python
c.execute('CREATE TABLE IF NOT EXISTS meta (id INTEGER,name TEXT)')
db_conn.commit()
```

5. Parse the meta data file and insert the data into the `meta` table

In [9]:

```python
txt_file = list(filter(lambda f: f.endswith('.txt'), all_files))[0]
with open(path + "\\" + txt_file) as f:
    data = f.read()

js = json.loads(data)
meta_data = pd.DataFrame(js).sort_values(by=['id']).reset_index(drop=True)
```

In [10]:

```python
meta_data.to_sql('meta', db_conn, if_exists='replace', index = False)
```

In [11]:

```python
c.execute('''
SELECT * FROM meta
LIMIT 10
          ''')

for row in c.fetchall():
    print (row)
```

```
(4, 'running_gopher')
(11, 'sleeping_beaver')
(12, 'laughing_ferret')
(13, 'hiding_eagle')
(14, 'sneaking_chicken')
(21, 'jumping_buffalo')
(39, 'hiding_kangaroo')
(51, 'crawling_gopher')
(65, 'playing_horse')
(68, 'crawling_hawk')
```

Join Data

Insert the data into the `meta` table

6. Write SQL to select all the records from the `meta` table and store it in a Pandas DataFrame called `df_meta`

In [12]:
```
df_meta = pd.read_sql('SELECT * FROM meta',db_conn)
df_time_series = pd.read_sql('SELECT * FROM time_series', db_conn)
```

7. Write SQL to find the total network traffic on the first day of the year across all the devices

In [13]:
```
total_network_traffic = pd.read_sql('SELECT SUM(trafFic) AS begining_total_network_traf
print(total_network_traffic)
```

```
   begining_total_network_traffic
0                    99991.502007
```

8. Write SQL to find the top 5 devices total network traffic by device name

Getting the Top 5 using SQL will require

-- Table Join between time series data and meta data using the device_id

-- To get the SUM will GROUP BY device name

-- Will have to get the TOP 5 of the SUM

*** To get result will have to nest SQL ***

I decided to do all of this using a single line with pandas - this way can also be parameterised and used as a function

In [14]:
```
df_meta.merge(df_time_series, left_on='id', right_on='device_id', how='inner').groupby(
```

Out[14]:

name	traffic
sneaking_catfish	38483.085738
hiding_orca	36897.428177
running_kangaroo	36844.243984
lurking_buzzard	36841.165169
sleeping_dolphin	36840.047211

In [15]:

```
# finally close connection
db_conn.close()
```

Task 2

1. Identify the IoT devices that exhibit odd behaviour relative to the subset.

In [16]:

```
df = df_meta.merge(df_time_series, left_on='id', right_on='device_id', how='inner')
df = df.drop(columns=['series_type','id']).set_index('date')
```

In [17]:

```
df.describe()['traffic']
```

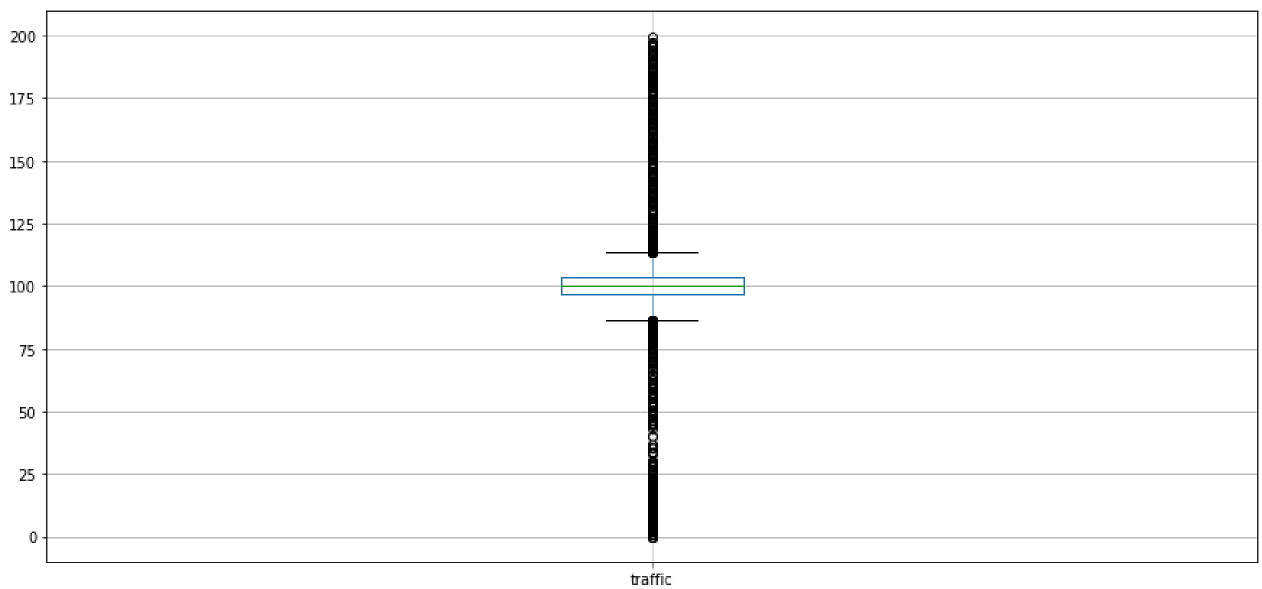
Out[17]:

```
count    366000.000000
mean         99.915448
std           5.936745
min           0.000000
25%          96.617741
50%          99.979066
75%         103.373141
max         199.703226
Name: traffic, dtype: float64
```

As we can see the traffic has outliers. For instance, max traffic is 199.7 while its mean is 99.9. The mean is sensitive to outliers, but the fact that the mean is small compared to the max value idicates the max value is an outlier. Similarly, the min traffic is 0 while the mean is 99.9 with a small standard deviation of 5.9. We will explore using additional methods...

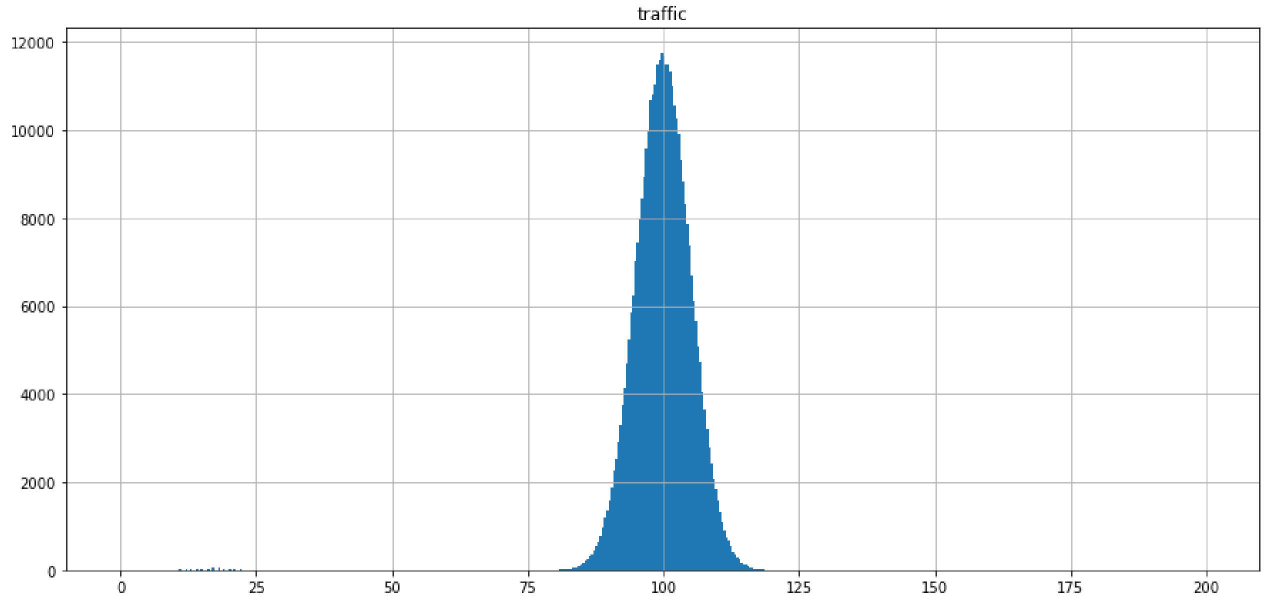
In [18]:

```
box = df.boxplot(column=['traffic'], figsize=(15,7))
```



In [19]:

```
hist = df.hist(column=['traffic'], figsize=(15,7), bins=500)
```



Using visuals we are able to identify that the are some outliers - to investigate further we use statistical method.

Finding outliers using statistical methods

We will calculate the outlier data points using the statistical method called interquartile range (IQR).

Using the IQR, the outliers data points are the ones falling below Q1 - 1.5 IQR or above Q3 + 1.5 IQR. The Q1 is the 25th percentile and Q3 is the 75th percentile of the dataset, and IQR represents the inter quartile range calculate by Q3 minus Q1 (Q3 - Q1),

In [20]:

```
def find_outliers_IQR(data):

    q1 = data['traffic'].quantile(0.25)
    q3 = data['traffic'].quantile(0.75)

    IQR = q3 - q1

    outliers = data[((data['traffic']<(q1-1.5*IQR)) | (data['traffic']>(q3+1.5*IQR)))]

    return outliers
```

In [21]:

```
outliers = find_outliers_IQR(df)


print('Number of outliers: ' + str(len(outliers)))
print('Total records: ' + str(len(df)))
print('Percent of outliers: {} % '.format(int(len(outliers))/int(len(df)) * 100) )
print('max outlier value: ' + str(outliers.traffic.max()))
print('min outlier value: ' + str(outliers.traffic.min()))
```

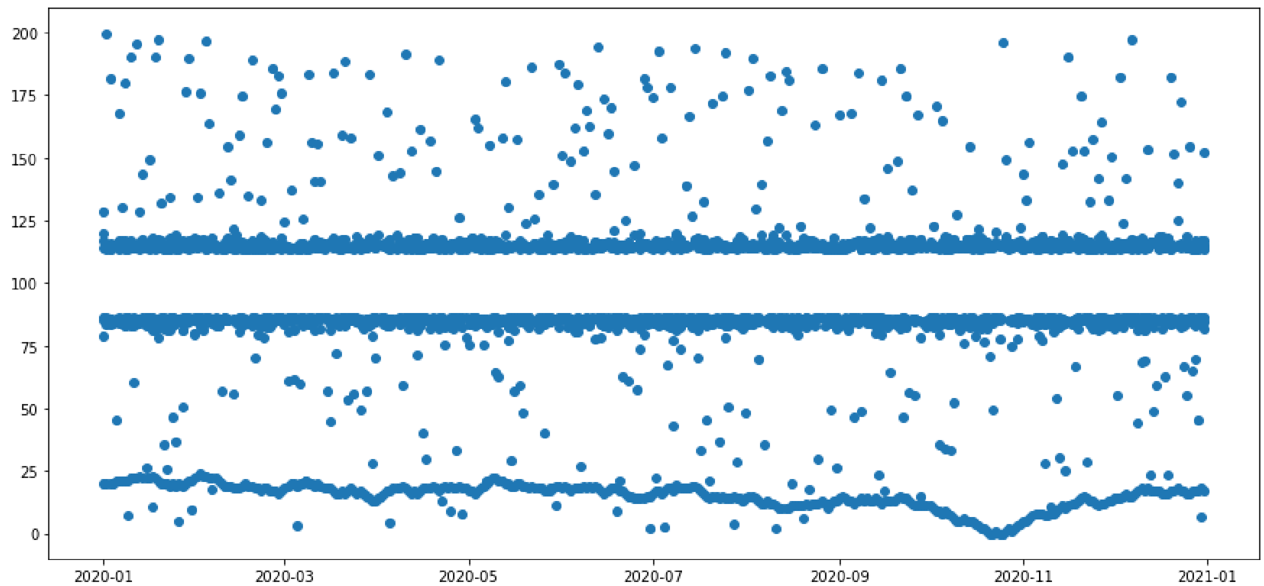
```
Number of outliers: 3210
Total records: 366000
Percent of outliers: 0.8770491803278688 %
```

```
max outlier value: 199.70322616167655
min outlier value: 0.0
```

2. Plot the daily network traffic of the odd behaved IoT devices

In [22]:
```
outliers.index = pd.to_datetime(outliers.index)
fig, ax = plt.subplots(figsize=(15, 7))
year_month_formatter = mdates.DateFormatter("%Y-%m")
ax.xaxis.set_major_formatter(year_month_formatter)
ax.scatter(outliers.index, outliers['traffic'])
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x27dd82a8bb0>
```



3. What can you deduce about these oddities?

Using IQR method, we find (3210) ~ 0.9 % percent of the data is outliers, these points fall outside of the interquartile range. The minimum outlier is 0 and the maximum outlier is ~ 199.7. This agrees with the data description method.

4. Output the odd behaved devices ID and name to a JSON formatted file called `oddities.json`

In [24]:
```
oddities = outliers.reset_index().to_json(orient='records')
```

In [25]:
```
with open('oddities.json', 'w', encoding='utf-8') as f:
    json.dump(oddities, f)
```

In [ ]:

In [ ]: