In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import datetime
from datetime import date
```

In [2]:
```
donations = pd.read_csv('data/donations.csv')
```

First, we'll review the dataset to see what it looks like and if it needs cleaning

In [3]:
```
donations.shape
```

Out[3]: (4687884, 3)

In [4]:
```
donations.head()
```

Out[4]:

	id	created_at	amount
0	00000ce845c00cbf0686c992fc369df4	2013-12-17 21:47:14	50.00
1	00002783bc5d108510f3f9666c8b1edd	2016-02-02 18:34:27	99.00
2	00002d44003ed46b066607c5455a999a	2016-10-25 20:15:11	10.00
3	00002d44003ed46b066607c5455a999a	2017-01-16 01:11:20	15.51
4	00002d44003ed46b066607c5455a999a	2017-01-16 14:20:10	100.00

In [5]:
```
donations.describe()
```

Out[5]:

	amount
count	4.687884e+06
mean	6.066879e+01
std	1.668996e+02
min	1.000000e-02
25%	1.482000e+01
50%	2.500000e+01
75%	5.000000e+01
max	6.000000e+04

Looking at the amount, 0.0 is listed. Let's look more closely. Could be errors in the data, the non-profit could be using their system to track free promotions etc etc

In [6]:
```
donations.sort_values('amount', ascending=True).head(10)
```

Out[6]:

	id	created_at	amount
3249451	af33360a708078c4e9ab9d5db05f37b1	2016-12-22 23:41:57	0.01
4521101	f7025786469e9df71c4e492aadfb7426	2015-02-05 17:59:17	0.01
2756016	95553d45a82f46cca385492d8dc2e1b2	2018-04-05 00:17:10	0.01
2103028	710cbea177231cfd0762786a60e886be	2017-08-18 16:21:28	0.01
4501370	f5d33f583486444a58154b5050440a4d	2016-09-14 18:03:42	0.01
3249448	af33360a708078c4e9ab9d5db05f37b1	2016-12-22 23:18:02	0.01
4145355	e1e4e887455b7cf2396116721f06d42c	2015-02-23 14:30:38	0.01
3249449	af33360a708078c4e9ab9d5db05f37b1	2016-12-22 23:23:39	0.01
472900	19df6adb35764e63470b5bfd13aff13b	2018-03-01 15:45:58	0.01
3249450	af33360a708078c4e9ab9d5db05f37b1	2016-12-22 23:33:08	0.01

In [7]:
```
donations_0 = donations[(donations.amount == 0)]
donations_0.shape
```

Out[7]:
```
(0, 3)
```

As there are only 350 items that have a 0 amount out of 4.6mm items, we will drop them for now

In [8]:
```
donations_minus0 = donations[donations.amount != 0]
donations_minus0.shape
```

Out[8]:
```
(4687884, 3)
```

Lets see the type of data we have.

In [9]:
```
donations.dtypes
```

Out[9]:
```
id             object
created_at     object
amount        float64
dtype: object
```

Any missing values?

In [10]:
```
donations.isnull().sum()
```

Out[10]:
```
id            0
created_at    0
amount        0
dtype: int64
```

No missing values which is good. If there was, we would have to decide how to handle them. Some possibilities:

- How much data was missing? Was it multiple data fields per row or just one data field?
- If say a few amounts were missing, one way is to sort by ID and if the the missing amount was by an ID that had other donations, we would look to add the missing value as an average of the amount before and after it

Updating the date/time string

Updating the created_at to a date/time field and adding Year and Month columns will allow us to better filter out date periods for analysis

In [11]:
```
donations['created_at'] = pd.to_datetime(donations.created_at)
donations['year'] = donations.created_at.dt.year
donations['month'] = donations.created_at.dt.month
```

Updating amount

In [12]:
```
donations['amount'] = donations['amount'].astype(np.int64)
```

In [13]:
```
donations.dtypes
```

Out[13]:
```
id                      object
created_at    datetime64[ns]
amount                 int64
year                   int64
month                  int64
dtype: object
```

Updating column names

I usually update the column names for various reasons:

- so they all follow the same format
- so they explain the purpose of the column better
- so they are easier to code with

In [14]:
```
donations.columns
```

Out[14]:
```
Index(['id', 'created_at', 'amount', 'year', 'month'], dtype='object')
```

In [15]:
```
donations.rename(columns = {'created_at' : 'date'}, inplace=True)
donations.columns
```

Out[15]:
```
Index(['id', 'date', 'amount', 'year', 'month'], dtype='object')
```

Saving cleaned dataset as new csv

Doing this allows us to now start analysis with a cleaned and fresh csv file

In [16]:
```
donations.to_csv('data/donations_clean.csv', index=False)
```