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

%matplotlib inline
```

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

Grouping year/month for further analysis

Grouping by year to further see if any data may be incomplete. Looking at 2012 and 2018, the counts are much lower than the other years so they may be incomplete.

In [3]:
```
donations_clean.groupby('year').count()
```

Out[3]:

year	id	date	amount	month
2012	149	149	149	149
2013	573983	573983	573983	573983
2014	746608	746608	746608	746608
2015	783362	783362	783362	783362
2016	957265	957265	957265	957265
2017	1190542	1190542	1190542	1190542
2018	435975	435975	435975	435975

As we thought, 2012 and 2018 have missing months so we need to decide what to do with them, if anything.

In [4]:
```
donations_clean.groupby(['year', 'month']).count()
```

Out[4]:

year	month	id	date	amount
2012	**10**	74	74	74
	11	8	8	8
	12	67	67	67
2013	**1**	22141	22141	22141
	2	32268	32268	32268
	3	32892	32892	32892
	4	28792	28792	28792
	5	33390	33390	33390
	6	28686	28686	28686
	7	35328	35328	35328
	8	62967	62967	62967
	9	93778	93778	93778
	10	65304	65304	65304
	11	60374	60374	60374
	12	78063	78063	78063
2014	**1**	50732	50732	50732
	2	55472	55472	55472
	3	68004	68004	68004
	4	45856	45856	45856
	5	38218	38218	38218
	6	34222	34222	34222
	7	47501	47501	47501
	8	103559	103559	103559
	9	89033	89033	89033
	10	73007	73007	73007
	11	54694	54694	54694
	12	86310	86310	86310
2015	**1**	67746	67746	67746
	2	61536	61536	61536
	3	63558	63558	63558
	...	...	...	...
	12	94473	94473	94473
2016	**1**	59069	59069	59069
	2	65706	65706	65706

year	month	id	date	amount
	3	89520	89520	89520
	4	48296	48296	48296
	5	46409	46409	46409
	6	40176	40176	40176
	7	65972	65972	65972
	8	140603	140603	140603
	9	116665	116665	116665
	10	81104	81104	81104
	11	106781	106781	106781
	12	96964	96964	96964
2017	1	95324	95324	95324
	2	94443	94443	94443
	3	115851	115851	115851
	4	54822	54822	54822
	5	59425	59425	59425
	6	56363	56363	56363
	7	95112	95112	95112
	8	169609	169609	169609
	9	120183	120183	120183
	10	114897	114897	114897
	11	115538	115538	115538
	12	98975	98975	98975
2018	1	116930	116930	116930
	2	87280	87280	87280
	3	95386	95386	95386
	4	111018	111018	111018
	5	25361	25361	25361

68 rows × 3 columns

For the next part of this analysis, lets drop 2012 and 2018 so we ony have complete years to work with

In [5]:
```
donations_gb = donations_clean[(donations_clean.year != 2012) & (donations_cle
an.year != 2018)]
donations_gb.groupby('year').count()
```

Out[5]:

year	id	date	amount	month
2013	573983	573983	573983	573983
2014	746608	746608	746608	746608
2015	783362	783362	783362	783362
2016	957265	957265	957265	957265
2017	1190542	1190542	1190542	1190542

Data Analysis: What to look for?

While this is a rather large dataset, it is also very limited. Some things we can look for that would be interesting to a non-profit organization in regards to their dontations are:

- When should we focus our donation drive?
- How should we focus our donation drive?
- Do we target all donators the same way?

First, lets do some grouping & plotting by year to see if the dollar amount of dontations has increased/decreased over period of time

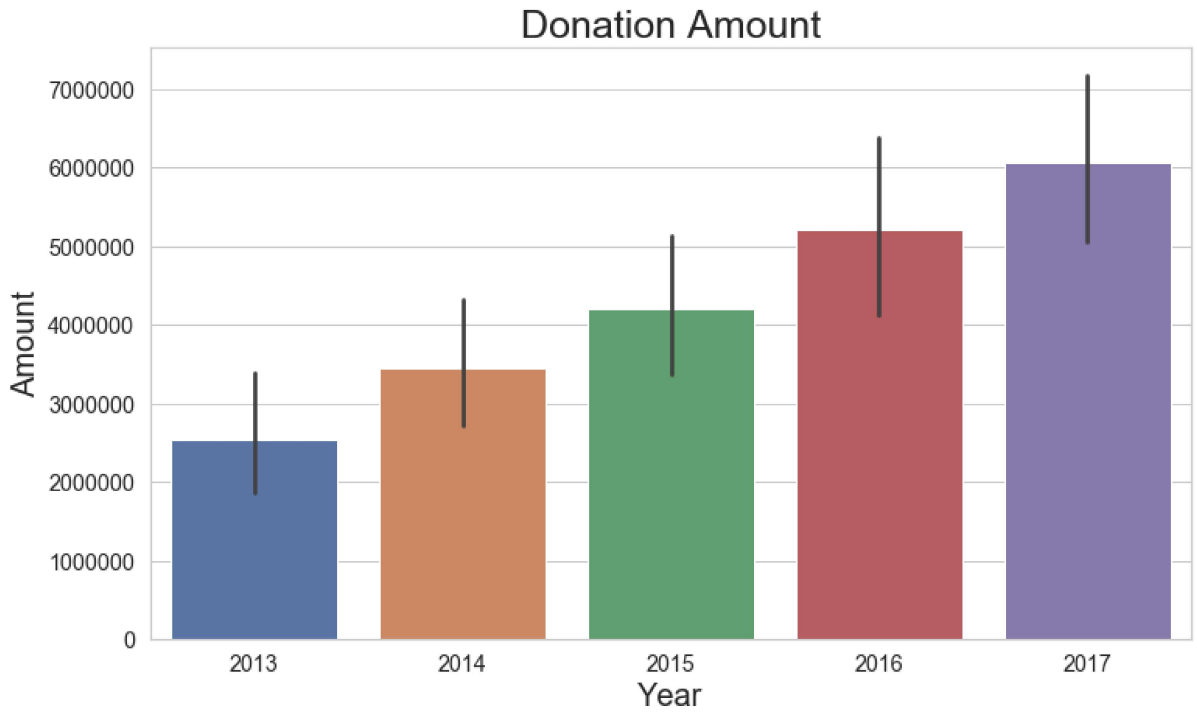
In [6]:
```
donations_gb = donations_gb.groupby(['year', 'month']).sum().reset_index()
donations_gb.head()
```

Out[6]:

	year	month	amount
0	2013	1	834960
1	2013	2	1502144
2	2013	3	1503283
3	2013	4	1456045
4	2013	5	1877268

In [7]:
```
plt.figure(figsize=(12, 7))
sns.set(style='whitegrid', font_scale=1.3)
ax = sns.barplot(x='year', y='amount', data=donations_gb)
plt.title('Donation Amount', fontsize=25)
plt.xlabel('Year', fontsize=20)
plt.ylabel('Amount', fontsize=20)
```

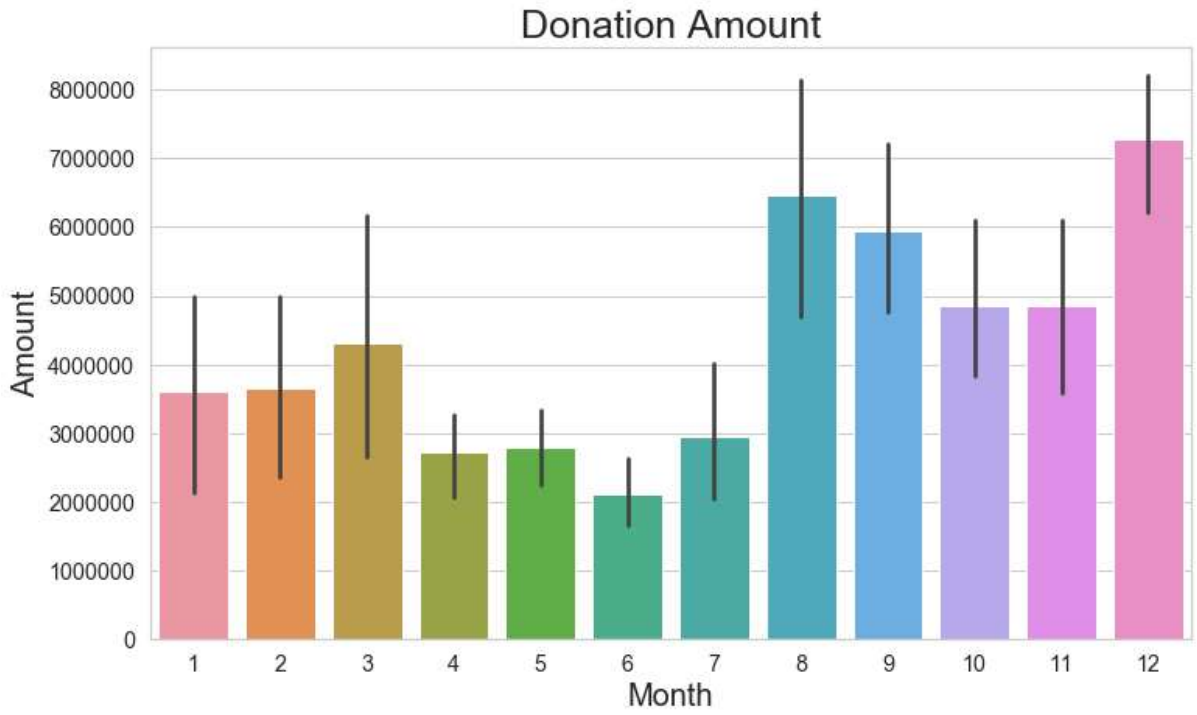
Out[7]:
```
Text(0, 0.5, 'Amount')
```



Steady increases in the donation amounts over the years. Good sign that the nonprofit group is healthy and thriving

In [8]:
```
plt.figure(figsize=(12, 7))
sns.set(style='whitegrid', font_scale=1.3)
ax = sns.barplot(x='month', y='amount', data=donations_gb)
plt.title('Donation Amount', fontsize=25)
plt.xlabel('Month', fontsize=20)
plt.ylabel('Amount', fontsize=20)
```

Out[8]:
```
Text(0, 0.5, 'Amount')
```



Looking by months, we see the donation amounts are much higher toward the end of the year. As this is a nonprofit organization, this could be caused by end of the year donation drives followed by holiday parties/fund raising events. This is something we could confirm with the group. If this is the case, they could think about delaying their fall campaign by a month and do a small fundraising push in the spring, give them 2 opportunites in the year for big donation drives.

Who to target and for how much?

Next I want to bring back in the original data set and only show transactions from 2018 where money was actually donated. Then I can target those donators and key in on how much they donated and using a sliding scale, come up with a suggested "new amount" that can be used in 2019 fundraising campaigns. For example, "We thank you for your last donation of $100 but if you increase your 2019 donation to \$130, we can add these services to our program"

In [9]:
```
donations_clean2018 = donations_clean[(donations_clean.amount != 0) & (donatio
ns_clean.year == 2018)]
donations_clean2018.sort_values(by=['amount'], ascending=True)
```

Out[9]:

	id	date	amount	year	month
4178841	e3e72e525e1038c838161bde8a00efff	2018-03-08 10:30:13	1	2018	3
708387	24ecca49933c30a0beb83090591720c0	2018-04-15 15:44:54	1	2018	4
708386	24ecca49933c30a0beb83090591720c0	2018-04-15 15:44:54	1	2018	4
708385	24ecca49933c30a0beb83090591720c0	2018-04-15 15:44:54	1	2018	4
708384	24ecca49933c30a0beb83090591720c0	2018-04-15 15:44:54	1	2018	4
1957456	69a8e18204a562b9b7901bddf8ab076d	2018-03-31 12:50:45	1	2018	3
1957458	69a8e18204a562b9b7901bddf8ab076d	2018-03-31 13:16:23	1	2018	3
1374609	4942150e351df37d02b1b1008032d795	2018-04-04 13:24:45	1	2018	4
708382	24ecca49933c30a0beb83090591720c0	2018-04-15 13:11:41	1	2018	4
1957461	69a8e18204a562b9b7901bddf8ab076d	2018-03-31 19:18:56	1	2018	3
1957463	69a8e18204a562b9b7901bddf8ab076d	2018-04-01 13:02:24	1	2018	4
201124	0b676f4f19188c233d7fbdda3ff2d002	2018-04-03 16:57:59	1	2018	4
1957464	69a8e18204a562b9b7901bddf8ab076d	2018-04-01 13:28:51	1	2018	4
1957466	69a8e18204a562b9b7901bddf8ab076d	2018-04-01 14:40:51	1	2018	4
708381	24ecca49933c30a0beb83090591720c0	2018-04-15 13:11:40	1	2018	4
708380	24ecca49933c30a0beb83090591720c0	2018-04-15 13:11:40	1	2018	4
708379	24ecca49933c30a0beb83090591720c0	2018-04-15 13:11:40	1	2018	4
1374608	4942150e351df37d02b1b1008032d795	2018-04-04 13:24:45	1	2018	4
1374607	4942150e351df37d02b1b1008032d795	2018-04-04 13:24:45	1	2018	4
708378	24ecca49933c30a0beb83090591720c0	2018-04-15 13:11:40	1	2018	4
708377	24ecca49933c30a0beb83090591720c0	2018-04-15 13:11:40	1	2018	4
1957480	69a8e18204a562b9b7901bddf8ab076d	2018-04-07 13:55:48	1	2018	4
1957481	69a8e18204a562b9b7901bddf8ab076d	2018-04-07 13:57:35	1	2018	4
1957465	69a8e18204a562b9b7901bddf8ab076d	2018-04-01 14:22:06	1	2018	4
1374606	4942150e351df37d02b1b1008032d795	2018-04-04 13:24:45	1	2018	4
1957451	69a8e18204a562b9b7901bddf8ab076d	2018-03-29 18:04:33	1	2018	3
201127	0b676f4f19188c233d7fbdda3ff2d002	2018-04-04 00:01:30	1	2018	4
1374617	4942150e351df37d02b1b1008032d795	2018-04-04 13:24:46	1	2018	4
201150	0b676f4f19188c233d7fbdda3ff2d002	2018-04-27 19:17:00	1	2018	4
201149	0b676f4f19188c233d7fbdda3ff2d002	2018-04-26 18:01:21	1	2018	4
...	...	...	...	...	...
1321764	4684a039c5e3217897a2094b3a14786b	2018-01-04 18:41:52	4354	2018	1
2217435	7705e9d9b7f95f7e768ac5501b64d2aa	2018-02-26 10:38:10	4397	2018	2
2213647	76cf4359ab09ecc85fc211f23d0d3663	2018-05-07 17:45:59	4414	2018	5
2043820	6e0c4f2497e8f42829c7f2d7986f7c47	2018-04-16 11:52:56	4648	2018	4

	id	date	amount	year	month
389118	1553ec82e1747171e492539a61211eeb	2018-03-01 20:39:54	4713	2018	3
2240653	7830793ea961d726ea62807babf2c9ce	2018-01-25 06:47:42	4737	2018	1
179807	0a24c21c9ce08be104a66b987174222e	2018-04-23 16:50:57	4785	2018	4
3223219	adaa89fac174b8ddca01214df765ff2a	2018-04-26 15:14:37	4803	2018	4
633977	21d6b25e35b4806577e9007b753313fd	2018-04-21 17:49:35	4854	2018	4
1446838	4d13712af0373671e0557f8f34ada412	2018-04-13 14:06:42	4929	2018	4
4100200	df5347111d6a231595cbfeb240cad0f4	2018-03-21 14:53:39	4991	2018	3
2755845	955245660b96c4c410c17fee65cb1c82	2018-04-27 14:27:34	5000	2018	4
558658	1df0793a1862177bce24d6272380d381	2018-02-02 10:29:45	5000	2018	2
4384357	ef4ef1fb5b90e9680a83e7a5a8bcf66e	2018-01-25 23:18:53	5032	2018	1
3727604	ca2d0036ccbc5ff03edd0f70304cc4c1	2018-01-25 21:35:33	5085	2018	1
3727603	ca2d0036ccbc5ff03edd0f70304cc4c1	2018-01-25 20:56:09	5085	2018	1
3443117	b9fc3c0f511fd84c1ca631f366a9cb41	2018-01-10 18:18:17	5402	2018	1
60540	036b437648dd2a825c9caf653eb01475	2018-02-10 19:07:32	5569	2018	2
1599364	55aa4bb22cda065e8658acb371c10135	2018-01-25 12:23:05	5700	2018	1
1486401	4f574681fb8805c553bcde9578e9661c	2018-01-04 13:59:18	6617	2018	1
2783478	96c4f21513cd8962acb147ab384e6434	2018-03-13 21:41:26	6729	2018	3
976385	345470c95caa44fd8c5c129075367669	2018-02-05 11:58:53	7890	2018	2
1348730	47fddc46e990cf6e13cfeee2185971d6	2018-05-03 18:10:24	8740	2018	5
485260	1a4447c8d8c82db6afb1db5a1aa19e1e	2018-04-16 12:18:57	9448	2018	4
4316037	ebb56c8b81859e95c817c941377a98f1	2018-01-06 10:54:59	9990	2018	1
1696952	5b4a72dff03a9acec082fc4d252e53b5	2018-03-01 17:42:09	12335	2018	3
221791	0c796f4db2c7adc93db6969153616cdc	2018-03-09 23:49:47	14732	2018	3
221792	0c796f4db2c7adc93db6969153616cdc	2018-04-03 17:30:21	14732	2018	4
221790	0c796f4db2c7adc93db6969153616cdc	2018-02-24 12:34:33	17777	2018	2
2226872	7779c0b3af936b7f8f6953f06882c094	2018-01-25 06:41:08	20000	2018	1

435903 rows × 5 columns

In [10]:
```
donations_clean2018 = donations_clean2018.assign(suggested_amount=0)
donations_clean2018.head()
```

Out[10]:

	id	date	amount	year	month	suggested_amount
17	00002eb25d60a09c318efbd0797bffb5	2018-01-16 15:32:41	50	2018	1	0
19	00004c31ce07c22148ee37acd0f814b9	2018-04-28 02:45:55	25	2018	4	0
39	00006084c3d92d904a22e0a70f5c119a	2018-01-04 17:48:50	5	2018	1	0
40	00006084c3d92d904a22e0a70f5c119a	2018-01-04 17:54:14	5	2018	1	0
41	00006084c3d92d904a22e0a70f5c119a	2018-03-31 08:41:32	10	2018	3	0

In [11]:
```
def suggested_amount(amount):
  if amount < 50:
    return (amount * .50) + amount
  elif amount < 100:
    return (amount * .40) + amount
  elif amount < 500:
    return (amount * .30) + amount
  elif amount < 1000:
    return (amount * .20) + amount
  elif amount < 10000:
    return (amount * .10) + amount
  else:
    return (amount * .05) + amount
```

In [12]:
```
donations_clean2018['suggested_amount'] = round(donations_clean2018.amount.app
ly(suggested_amount), 0)
```

In [13]:
```
donations_clean2018.head(10)
```

Out[13]:

	id	date	amount	year	month	suggested_amount
17	00002eb25d60a09c318efbd0797bffb5	2018-01-16 15:32:41	50	2018	1	70.0
19	00004c31ce07c22148ee37acd0f814b9	2018-04-28 02:45:55	25	2018	4	38.0
39	00006084c3d92d904a22e0a70f5c119a	2018-01-04 17:48:50	5	2018	1	8.0
40	00006084c3d92d904a22e0a70f5c119a	2018-01-04 17:54:14	5	2018	1	8.0
41	00006084c3d92d904a22e0a70f5c119a	2018-03-31 08:41:32	10	2018	3	15.0
42	00006084c3d92d904a22e0a70f5c119a	2018-04-02 20:44:23	10	2018	4	15.0
43	00006084c3d92d904a22e0a70f5c119a	2018-04-14 22:36:22	10	2018	4	15.0
44	00006084c3d92d904a22e0a70f5c119a	2018-04-14 23:04:42	5	2018	4	8.0
46	0000812bd5629117f8909f73acbe8b7d	2018-04-23 16:10:13	50	2018	4	70.0
50	0000a1288b8ccdeaaf716a2480d7b06a	2018-02-12 20:07:32	50	2018	2	70.0

Now with this new suggested amount, we are able to create a fund raising campaign and target every donator in 2018 with a suggested donation amount that is taylored to their last donation. Asking someone to increase their donation by 50\% from a previous $5 donation is very different then asking someone to increase their donation by 50% from a prevoius \$1000 or higher donation. This sliding scale gives us a higher success rate of increaseing our donation amounts for 2019.

In [ ]:
```

```