Snapchat Political Ads

This project uses political ads data from Snapchat, a popular social media app. Interesting questions to consider include:

- What are the most prevalent organizations, advertisers, and ballot candidates in the data? Do you recognize any?
- What are the characteristics of ads with a large reach, i.e., many views? What may a campaign consider when maximizing an ad's reach?
- What are the characteristics of ads with a smaller reach, i.e., less views? Aside from funding constraints, why might a campaign want to produce an ad with a smaller but more targeted reach?
- What are the characteristics of the most expensive ads? If a campaign is limited on advertising funds, what type of ad may the campaign consider?
- What groups or regions are targeted frequently? (For example, for single-gender campaigns, are men or women targeted more frequently?) What groups or regions are targeted less frequently? Why? Does this depend on the type of campaign?
- Have the characteristics of ads changed over time (e.g. over the past year)?
- When is the most common local time of day for an ad's start date? What about the most common day of week? (Make sure to account for time zones for both questions.)

Getting the Data

The data and its corresponding data dictionary is downloadable [here (https://www.snap.com/en-US/political-ads/)](https://www.snap.com/en-US/political-ads/). Download both the 2018 CSV and the 2019 CSV.

The CSVs have the same filename; rename the CSVs as needed.

Note that the CSVs have the exact same columns and the exact same data dictionaries (`readme.txt`).

Cleaning and EDA

- Concatenate the 2018 CSV and the 2019 CSV into one DataFrame so that we have data from both years.
- Clean the data.
  - Convert `StartDate` and `EndDate` into datetime. Make sure the datetimes are in the correct time zone. You can use whatever timezone (e.g. UTC) you want as long as you are consistent. However, if you want to answer a question like "When is the most common local time of day for an ad's start date," you will need to convert timezones as needed. See Hint 2 below for more information.
- Understand the data in ways relevant to your question using univariate and bivariate analysis of the data as well as aggregations.

Hint 1: What is the "Z" at the end of each timestamp?

Hint 2: `pd.to_datetime` *will be useful here.* `Series.dt.tz_convert` *will be useful if a change in time zone is needed.*

Tip: To visualize geospatial data, consider [Folium (https://python-visualization.github.io/folium/)](https://python-visualization.github.io/folium/) or another geospatial plotting library.

Assessment of Missingness

Many columns which have `NaN` values may not actually have missing data. How come? In some cases, a null or empty value corresponds to an actual, meaningful value. For example, `readme.txt` states the following about `Gender`:

> Gender - Gender targeting criteria used in the Ad. If empty, then it is targeting all genders

In this scenario, an empty `Gender` value (which is read in as `NaN` in pandas) corresponds to "all genders".

- Refer to the data dictionary to determine which columns do **not** belong to the scenario above. Assess the missingness of one of these columns.

Hypothesis Test / Permutation Test

Find a hypothesis test or permutation test to perform. You can use the questions at the top of the notebook for inspiration.

Summary of Findings

Introduction

Our dataset is about political ads on Snapchat, including descriptions on the date they were published, the amount of money a company invested into the ad, the number of people that viewed the ad, and more. Given Snapchat political data from 2018 and 2019, we want to visualize our data and test to see if the ad creators specifically choose a certain day to publish their ads. It is important for these creators to publish their ad on the right day in order to maximize the number of views. We measure the importance of a day in the week by the amount of money spent on the ads that were published on that given day.

Cleaning and EDA

First, we needed to select only the columns that we needed. From all the columns, I chose: 'ADID', 'Spend', 'StartDate', 'EndDate', 'CountryCode'. 'Spend' was our measure of how much money a third party put into creating the ad and 'StartDate' and 'EndDate' gave us some reference as to when the ads were published and taken back. Although we did not use 'ADID' in our analysis, it served as a way of distinguishing between ads so that we did not get confused.

After extracting our necessary columns, we checked to see the percentage of null values in our data. All the columns had no null values except for the EndDate. As such, we decided to focus on when the ad was published as our indication of importance. It also makes more sense because the release date of an ad is far more important than the end date because the life of an ad is dictacted by how well it does in the beginning.

by how well it does in the beginning.

The data that was provided to us made it difficult to pinpoint the exact origin of the political ad. This was an obstacle because we were focusing on the day that the ad was published according to its local time. We were not provided the timezone or area of the ad's creation so we made the executive decision to focus solely on ads created in the United States, which was indicated by 'CountryCode'. As a result, we got rid of around 47% of the data so that we only had ads from the U.S. We converted the times to Pacific Standard Time, another abstraction we were forced to do because we could not account for all the different timezones.

Now that our data has been cleaned completely, we started our Exploratory Data Analysis. We plotted the number of ads by weekday to see how it was distributed amongst the seven days and then categorized all the ads by the time of week they were released (Weekday vs Weekend). We noticed that there were far more political ads on the weekdays than on the weekends and that the money spent on ads was significantly higher on the weekdays. We looked at the distributions of expenditures of the ads on weekdays vs weekends and noticed that they both were gathered around zero and spread mostly to around 5000. However, these distributions were strongly skewed to the right because there were many outliers, companies and creators that invested hundreds of thousands of dollars into the ads.

In order to properly visualize the distribution and eradicate strong biases in the data, we decided to also work with the data but without outliers. We approached this by calculating the z-score of the expenditures and getting rid of the z-scores that were greater than 3, which were 16 data points. When plotting the amount of money spent on the ads on weekdays vs weekends, the bars drastically changed and were more similar than when it included the outliers. The distributions of the expenditures were more similar as well without having the outliers drag out the data points and the box plots showed a better representation of the data, although there still being outliers.

Despite looking at the data with and without outliers, we need to understand that these outliers are valid points. There were some ads that had a lot of money invested into it such as ads by General Mills. However, if we were to look from a company's standpoint, we would want to see the ads that did relatively well and managed their cost, which we saw that most fell between 0-5000 dollars. We might not have the budget to afford ads like General Mills so we need to look at a distribution that does not include these expensive ads. We included both outliers and no outliers distributions and visualizations to show the difference between the two and depending on our viewpoint, it is appropriate to choose one over the other.

Finally, because we were also curious, we created pivot tables that not only indicating how much money on average was spent on an ad on that particular day, but we can see that for every month. It is interesting to see how our distributions change over the months, as some days are prioritized more than others when comparing months.

Assessment of Missingness

We believe the data of End Date is NMAR; the missingness of enddate is likely to be predicted by how much money was spent on the advertising, as well as Organization or Country. If it is NMAR, we will not use it in analysis. Based on the permutation tests assessing missingness in the 'End Date' column, we were able to calculate p-values that were less than 0.05 that indicated that the distribution is significantly different than if it were randomized, and the missigness of End Date is dependent on Spend but not dependent on StartMonth.

Permutation Test

We conducted a permutation test to test the following question:

Is there a difference between the amount of money spent on ads on the weekends vs the weekdays?

Null hypothesis: There is no significant difference between the amount of money spent on ads shown on weekends and weekdays.

Alternate hypothesis: There is a significant difference between the amount of money spent on ads shown on weekends and weekdays.

Test Statistic: Absolute difference in means

We had a 95% significance level and got a p-value of 0.162. As a result: "We cannot reject the null hypothesis that there is no significant difference between the amount of money spent on ads shown on weekdays and weekends"

However, we decided to run the permutation test again on our data that did not include outliers. We observed that the outliers mostly on Weekdays. After running our test again, we got 0.419, and we cannot reject our null hypothesis. It is possible that there is no significant difference between the weekday and weekend in

Code

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns
from scipy import stats
%matplotlib inline
%config InlineBackend.figure_format = 'retina'  # Higher resolution figures
```

Cleaning and EDA

In [2]:
```
# Reading in the CSVs

pol18 = pd.read_csv('2018PoliticalAds.csv')
pol19 = pd.read_csv('2019PoliticalAds.csv')
```

In [3]:
```
# First, I check to see if all the columns match between the CSVs

pol19.columns == pol18.columns
```

Out[3]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

In [4]:
```
# I check to see if the concatenation of the CSVs was successful

pol_comb = pd.concat([pol18, pol19], ignore_index = True)
pol_comb.columns == pol19.columns
```

Out[4]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

In [5]:
```
# I chose the columns that were relevant to my question

useful_cols = ['ADID', 'Spend', 'StartDate', 'EndDate', 'CountryCode']
pol_comb = pol_comb[useful_cols]
```

In [6]:
```
# I create a table that contains the percentage of null values in each cate

s_type = pol_comb.dtypes
s_null = pol_comb.isnull().mean().sort_values(ascending = False)
type_null = pd.concat([s_type, s_null], axis = 1)
type_null.columns = ['type', 'null %']
type_null.sort_values(by = 'null %', ascending = False)
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: Futur
eWarning: Sorting because non-concatenation axis is not aligned. A future
version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  """
```

Out[6]:

	type	null %
EndDate	object	0.182052
ADID	object	0.000000
CountryCode	object	0.000000
Spend	int64	0.000000
StartDate	object	0.000000

In [7]:
```
# First we convert the date to DateTime Objects

pol_comb["StartDate"] = pd.to_datetime(pol_comb["StartDate"])
pol_comb["EndDate"] = pd.to_datetime(pol_comb["EndDate"])
```

In [8]:
```
# Given the data, it is difficult to assess the origin of the political ad,
# For the purposes of this project, I identified the country origin of wher

pol_comb['CountryCode'].value_counts(normalize = True)
```

Out[8]:
```
united states           0.535848
united kingdom          0.122306
norway                  0.105201
canada                  0.093486
denmark                 0.024602
netherlands             0.018510
france                  0.016870
austria                 0.008669
sweden                  0.008669
australia               0.008435
finland                 0.008201
kuwait                  0.008201
switzerland             0.007263
belgium                 0.006560
ireland                 0.006092
india                   0.004217
poland                  0.003280
germany                 0.003046
south africa            0.002343
nigeria                 0.002109
united arab emirates    0.001640
argentina               0.001406
turkey                  0.000937
lithuania               0.000703
puerto rico             0.000469
chile                   0.000234
iraq                    0.000234
new zealand             0.000234
brazil                  0.000234
Name: CountryCode, dtype: float64
```

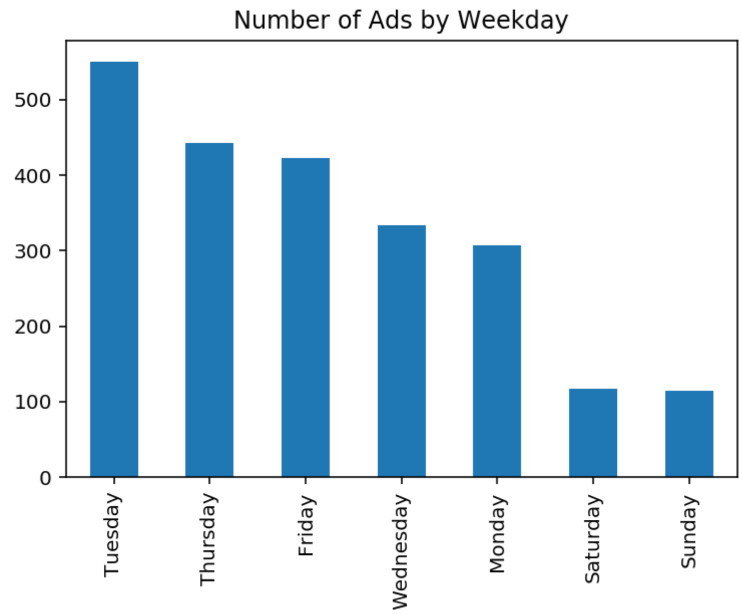
In [9]:
```
# Because the majority of ads come from the US (53.58%), I decided to focus

us_pol_comb = pol_comb[pol_comb['CountryCode'] == 'united states'].reset_in
```

In [10]:
```
# Although there are still different time zones in the US, I decided to set
# As a result, ads that were published after 9pm PST could potentially be a

us_pol_comb.loc[:, "StartDate"] = us_pol_comb.loc[:, "StartDate"].dt.tz_con
us_pol_comb.loc[:, "EndDate"] = us_pol_comb.loc[:, "EndDate"].dt.tz_convert
```

In [11]:
```
# We can extract the month, day of week, and hour of when the ads are relea

us_pol_comb['StartDOW'] = us_pol_comb['StartDate'].apply(lambda x: x.weekda
us_pol_comb['StartMonth'] = us_pol_comb['StartDate'].apply(lambda x: x.mont
```

In [12]:
```
# Distribution of Weekdays - It is interesting to note that the ads usually
# This is an issue that we want to focus on

dayDict = {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Fr
us_pol_comb['StartDOW'].replace(dayDict, inplace = True)
us_pol_comb['StartDOW'].value_counts().plot(kind = 'bar', title = 'Number c
```

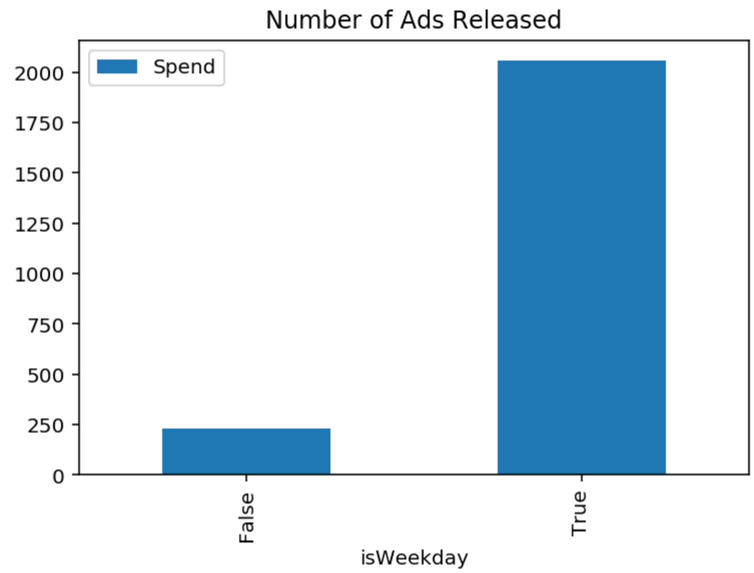
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1a210bd0d0>



In [13]:
```
# In order to focus on our question, it is important we now group the ads i

us_pol_comb['isWeekday'] = us_pol_comb['StartDOW'].apply(lambda x: True if
```

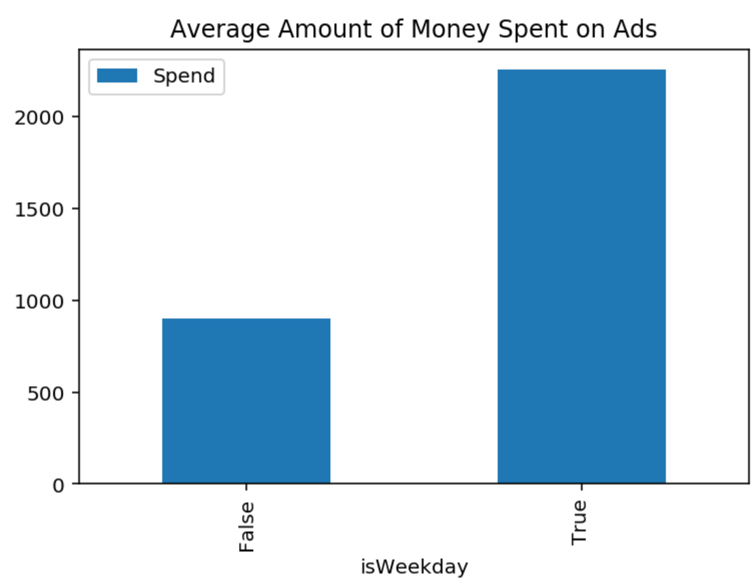
In [14]:
```
# We can now visualize a bar chart of the number of ads aggregated by the p
# We observe a high number of ads released on a weekday compared to the wee

dow = us_pol_comb[['isWeekday', 'Spend']]
dow_counts = dow.groupby('isWeekday').count()
dow_counts.plot.bar(title = 'Number of Ads Released')
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x1a20031590>



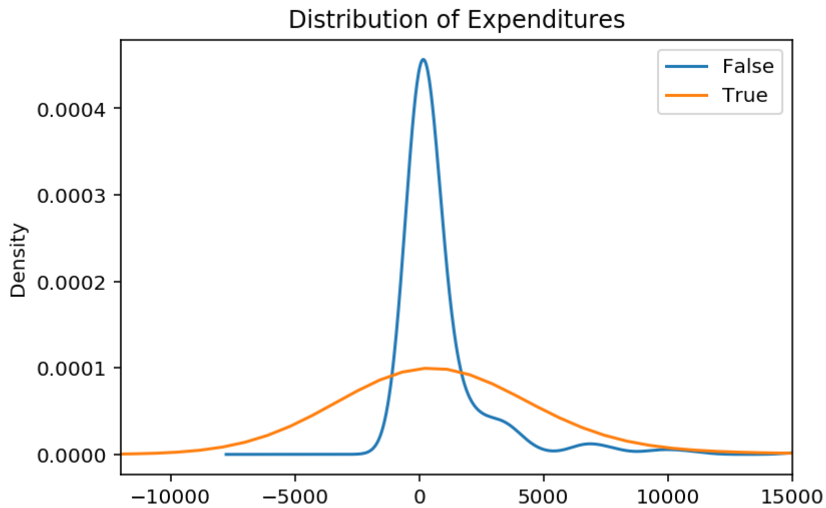
In [15]:
```
# Similarly, we can see that more money was spent on average on ads release
# However, this visualization can be biased due to outliers in the data

dow_median_spend = dow.groupby('isWeekday').mean()
dow_median_spend.plot.bar(title = 'Average Amount of Money Spent on Ads')
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1a202bc890>



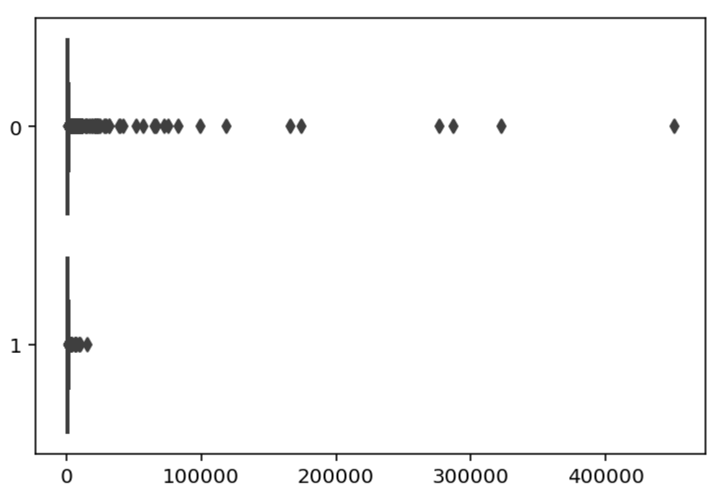
In [16]:
```
# We visualized the distributions of the expenditures on the weekday vs the
# The visualization is wide because it is being drawn out by outliers

us_pol_comb.groupby('isWeekday')['Spend'].plot(kind='kde', legend=True, tit
plt.xlim(-12000, 15000)
```

Out[16]: (-12000, 15000)



In [17]:
```
# To emphasize the influence of the outliers, we created a box plot to see
# The box itself is squished on the far left because there are so many data

weekday = dow[dow['isWeekday'] == True]
weekend = dow[dow['isWeekday'] == False]
sns.boxplot(data=[weekday['Spend'], weekend['Spend']], orient='h')
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x1a20a77150>



In [18]:
```
us_pol_comb.shape[0]
```

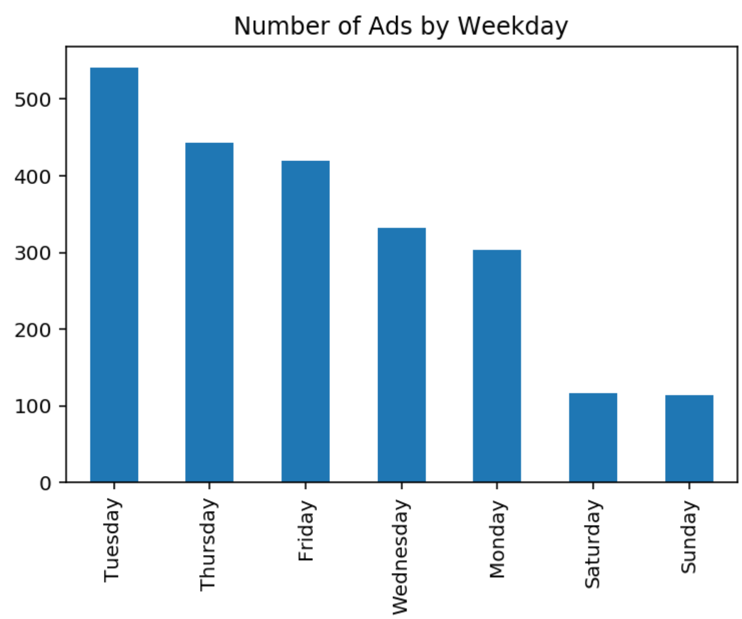
Out[18]: 2287

In [19]:
```
# To visualize our data without the outliers, we decided to calculate the z
# We got rid of the data points that had a z-score greater than 3, which we


no_out = us_pol_comb.copy()
z = np.abs(stats.zscore(no_out['Spend']))
print(np.where(z > 3))
no_out = no_out[z < 3]
```

```
(array([ 675,  739,  774,  910, 1252, 1292, 1355, 1396, 1416, 1430, 1441,
       1639, 1931, 1934, 1962, 2153]),)
```

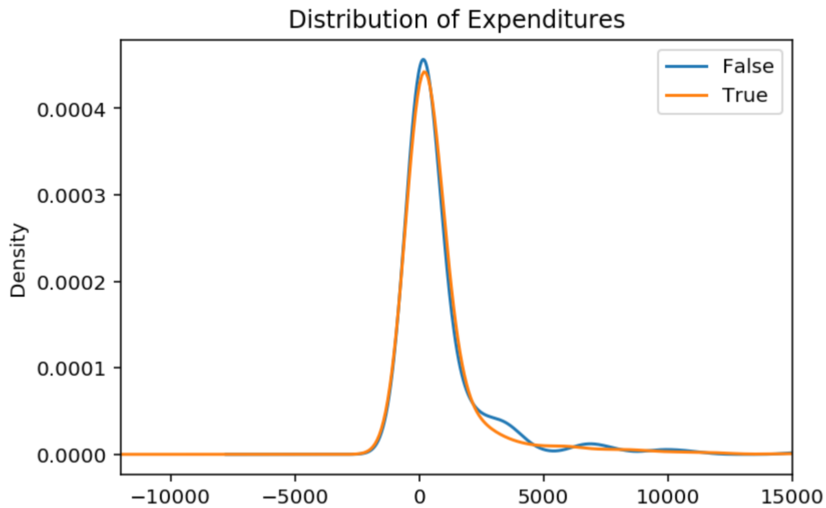
In [20]:
```
# Because we took out only 16 data points, we do not expect the distributio

dayDict = {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Fr
no_out['StartDOW'].replace(dayDict, inplace = True)
no_out['StartDOW'].value_counts().plot(kind = 'bar', title = 'Number of Ads
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x1a20dd9710>



In [21]:
```
# The distribution around zero

no_out.groupby('isWeekday')['Spend'].plot(kind='kde', legend=True, title='D
plt.xlim(-12000, 15000)
```

Out[21]: (-12000, 15000)



In [22]:
```
# Now we can go back to categorizing the ads by time of week without incluc

dow_no_out = no_out[['isWeekday', 'Spend']]
dow_no_out
```

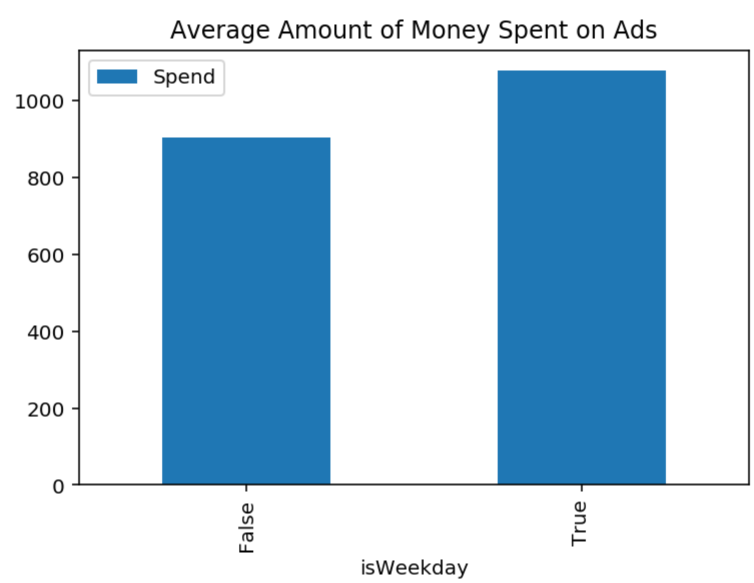
Out[22]:

	isWeekday	Spend
0	True	35
1	True	56
2	True	2048
3	True	196
4	True	655
...	...	...
2282	True	49
2283	True	12
2284	True	28
2285	True	2
2286	True	19

2271 rows × 2 columns

In [23]:
```
# By taking out the outliers, we can drastically see a change in the averag
# This is because all the outliers were in weekdays, which means that the c

dow_median_spend_no_out = dow_no_out.groupby('isWeekday').mean()
dow_median_spend_no_out.plot.bar(title = 'Average Amount of Money Spent on
```

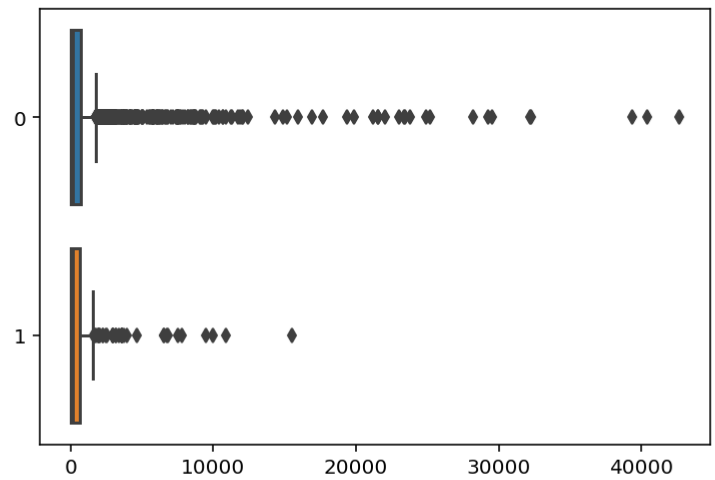
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1a210bd210>



In [24]:
```
weekday_outno = dow_no_out[dow_no_out['isWeekday'] == True]
weekend_outno = dow_no_out[dow_no_out['isWeekday'] == False]
```

In [25]:
```
# Although not as visual as we would want, the data is distributed as such
# There are several data points that are still outliers (not as extreme) ar

sns.boxplot(data=[weekday_outno['Spend'], weekend_outno['Spend']], orient='
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2148f690>



In [26]:
```
# We also made a pivot table to look at the average amount of money spent o
# This includes the outliers

dayDict = {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Fr
us_pol_comb['StartDOW'].replace(dayDict, inplace = True)
monthDict= {1:'Jan', 2:'Feb', 3:'Mar', 4:'Apr', 5:'May', 6:'Jun', 7:'Jul',
us_pol_comb['StartMonth'].replace(monthDict, inplace = True)

pd.pivot_table(us_pol_comb, values = 'Spend', index = 'StartDOW', columns =
```

Out[26]:

StartMonth	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	No
StartDOW										
Friday	4772.50	150.08	4120.98	903.17	1013.00	1020.75	1654.93	2326.27	1337.20	355.0
Monday	1714.00	1070.62	97.23	437.50	0.00	10612.23	120.25	299.50	831.83	1359.9
Saturday	230.00	1703.50	93.33	0.00	107.67	0.00	0.00	1712.17	77.00	294.1
Sunday	0.00	150.67	2324.00	0.00	5357.00	0.00	976.00	352.00	298.00	315.1
Thursday	324.80	1510.17	254.27	0.00	1405.00	2133.86	270.75	416.50	753.84	387.5
Tuesday	7765.75	1712.33	3031.45	100.00	1817.00	456.45	682.77	378.00	240.60	922.3
Wednesday	0.00	1473.00	3522.29	0.00	212.00	1090.40	254.33	836.57	121.43	1347.1

In [27]:
```
e also made a pivot table to look at the average amount of money spent on a
his does not include outliers and we can see some differences like Tuesdays
e want to test

Dict = {0: 'Monday', 1: 'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Friday
out['StartDOW'].replace(dayDict, inplace = True)
thDict= {1:'Jan', 2:'Feb', 3:'Mar', 4:'Apr', 5:'May', 6:'Jun', 7:'Jul', 8:'A
out['StartMonth'].replace(monthDict, inplace = True)

pivot_table(no_out, values = 'Spend', index = 'StartDOW', columns = 'StartMo
```

Out[27]:

StartMonth	Apr	Aug	Dec	Feb	Jan	Jul	Jun	Mar	May	Nov
StartDOW										
Friday	4772.50	150.08	23.39	903.17	1013.00	1020.75	1654.93	2326.27	1337.20	355.03
Monday	1714.00	1070.62	97.23	437.50	0.00	8192.14	120.25	299.50	831.83	1359.97
Saturday	230.00	1703.50	93.33	0.00	107.67	0.00	0.00	1712.17	77.00	294.17
Sunday	0.00	150.67	2324.00	0.00	5357.00	0.00	976.00	352.00	298.00	315.17
Thursday	324.80	1510.17	254.27	0.00	1405.00	2133.86	270.75	416.50	753.84	387.50
Tuesday	2471.73	1712.33	3031.45	100.00	1817.00	456.45	682.77	378.00	240.60	922.37
Wednesday	0.00	1473.00	3522.29	0.00	212.00	1090.40	254.33	836.57	121.43	1347.11

Assessment of Missingness

Analyzing the Missingness of End Date

In [28]:
```
missing_table = us_pol_comb.assign(end_date_isnull=us_pol_comb['EndDate'].i
missing_table.head()
```

Out[28]:

	ADID	Spend	StartDate	EndDate	CountryCode
0	6f6f6abb25d183bc3f0a2df46d18c65a18f6e1cac73416...	35	2018-11-06 10:21:20-08:00	2018-11-06 17:19:08-08:00	united states
1	64d906646b616c034c91b69b9e7851944844eb456dd203...	56	2018-09-28 16:10:14-07:00	2018-10-16 19:00:00-07:00	united states
2	45d7697e2522ccdd56b699e832792b9b659f7159e180a2...	2048	2018-09-28 12:00:00-07:00	2018-10-26 20:59:00-07:00	united states
3	46d8326f706f56296fa29f51b5127c67190807ccc08534...	196	2018-10-26 10:58:01-07:00	2018-11-06 14:59:59-08:00	united states
4	3af2a0894b7d969aed065b1c1d0a399882df677209dfe5...	655	2018-10-01 14:08:10-07:00	NaT	united states

Is missingness dependent on Spend?

In [29]:
```
#only need Spend and end_date_isnull columns
df = missing_table[['Spend','end_date_isnull']]
df.head()
```

Out[29]:

	Spend	end_date_isnull
0	35	False
1	56	False
2	2048	False
3	196	False
4	655	True

In [30]:
```
df_means = df.groupby('end_date_isnull').mean()
df
```

Out[30]:

	Spend	end_date_isnull
0	35	False
1	56	False
2	2048	False
3	196	False
4	655	True
...	...	...
2282	49	False
2283	12	False
2284	28	False
2285	2	False
2286	19	True

2287 rows × 2 columns

In [31]:
```
#observed abs diff in means
observed_diff_means = abs(df_means.diff().iloc[-1,0])
observed_diff_means
```

Out[31]: 2255.1096904324604

In [32]:
```
#permutation test
N = 1000
results = []

for _ in range(N):
    #create shuffled dataframe
    s = df['end_date_isnull'].sample(frac=1, replace=False).reset_index(drc
    shuffled = df.assign(shuffled_null=s)

    #calculate difference of means and add to results array
    shuff_means_table = shuffled.groupby('shuffled_null').mean()
    results.append(abs(shuff_means_table.diff().iloc[-1,0]))

diffs_of_means_shuff = pd.Series(results)
```

In [33]:
```
pval = (diffs_of_means_shuff>=observed_diff_means).sum()/N
pval
```

Out[33]: 0.002

Since the p-value is less than 0.05, the distribution is significantly different than if it were randomized, and the missingness of End Date is dependent on Spend

Is missingness dependent on StartMonth?

In [34]:
```
#only need StartMonth and end_date_isnull columns
df = missing_table[['StartMonth','end_date_isnull']]
df.head()
```

Out[34]:

	StartMonth	end_date_isnull
0	Nov	False
1	Sep	False
2	Sep	False
3	Oct	False
4	Oct	True

In [35]:
```
df_means = df.groupby('StartMonth').mean()
df_means
```

Out[35]:

	end_date_isnull
StartMonth	
Apr	0.214286
Aug	0.069307
Dec	0.157895
Feb	0.000000
Jan	0.100000
Jul	0.259615
Jun	0.186047
Mar	0.111111
May	0.053435
Nov	0.277966
Oct	0.101852
Sep	0.556686

In [36]:
```
#observed abs diff in means
observed_tvd = sum(df_means['end_date_isnull'] - (df_means['end_date_isnull
observed_tvd
```

Out[36]:
```
-4.440892098500626e-16
```

In [37]:
```
#permutation test
N = 1000
results = []

for _ in range(N):
    #create shuffled dataframe
    s = df['end_date_isnull'].sample(frac=1, replace=False).reset_index(drc
    shuffled = df.assign(shuffled_null=s)

    #calculate tvd and add to results array
    shuff_means_table = shuffled.groupby('shuffled_null').mean()
    results.append(sum(shuff_means_table['end_date_isnull'] - (shuff_means_

tvds_shuff = pd.Series(results)
```

In [38]:
```
pval = (tvds_shuff>=observed_tvd).sum()/N
pval
```

Out[38]: 1.0

The p-value is greater than 0.05, so missingness of end date is NOT DEPENDENT on start month.

Hypothesis Test

In [39]:
```
#we only need StartDOW, which represents the day of week on which the ad wa
#of money spent in USD
DOW_and_spend = us_pol_comb[['StartDOW', 'Spend']]
DOW_and_spend.head()
```

Out[39]:

	StartDOW	Spend
0	Tuesday	35
1	Friday	56
2	Friday	2048
3	Friday	196
4	Monday	655

In [40]:
```
dow.head()
```

Out[40]:

	isWeekday	Spend
0	True	35
1	True	56
2	True	2048
3	True	196
4	True	655

In [41]:
```
#separating ads into weekend and weekday; we no longer need StartDOW
weekday = DOW_and_spend['StartDOW'].apply(lambda x: True if x not in ['Satu
weekday_and_spend = DOW_and_spend.assign(Weekday = weekday).drop('StartDOW'
weekday_and_spend.head()
```

Out[41]:

	Spend	Weekday
0	35	True
1	56	True
2	2048	True
3	196	True
4	655	True

Permutation Test - Testing by Simulation

- **Null hypothesis**: There is no significant difference between the amount of money spent on ads shown on weekends and weekdays.
- **Alternate hypothesis**: There is a significant difference between the amount of money spent on ads shown on weekends and weekdays.
- **Test Statistic**: Absolute difference in means

set a significance level of 0.05

In [42]:
```
#observed means
means_table = dow.groupby('isWeekday').mean()
means_table
```

Out[42]:

isWeekday	Spend
False	903.160173
True	2254.464494

In [43]:
```
#observed test statistic
observed_difference = means_table.diff().iloc[-1,0]
observed_difference
```

Out[43]:
```
1351.304321003251
```

In [44]:
```
#simulation

N = 1000
results = []

for _ in range(N):
    #create shuffled dataframe
    s = weekday_and_spend['Weekday'].sample(frac=1, replace=False).reset_in
    shuffled = weekday_and_spend.assign(weekday=s)

    #calculate difference of means and add to results array
    shuff_means_table = shuffled.groupby('weekday').mean()
    results.append(abs(shuff_means_table.diff().iloc[-1,0]))

diffs_of_means = pd.Series(results)
```

In [45]:
```
diffs_of_means
```

Out[45]:
```
0      1381.937467
1      3582.934680
2      1231.093034
3        52.000181
4       442.802434
          ...
995    2393.832977
996      97.890806
997     586.922112
998     455.707668
999     320.036514
Length: 1000, dtype: float64
```

In [46]:
```
pval = (diffs_of_means >= observed_difference).sum() / N
pval
```

Out[46]:
```
0.179
```

Conclusion

- We cannot reject the null hypothesis that there is no significant difference between the amount of money spent on ads shown on weekdays and weekends

However

Our exploratory data analysis showed clear outliers - what would happen if these were removed?

In [47]:
```
z = np.abs(stats.zscore(weekday_and_spend['Spend']))
weekday_and_spend_clean = weekday_and_spend[z<3]
weekday_and_spend_clean.head()
```

Out[47]:

	Spend	Weekday
0	35	True
1	56	True
2	2048	True
3	196	True
4	655	True

In [48]:
```
#observed means
means_table_clean = dow_no_out.groupby('isWeekday').mean()
means_table_clean
```

Out[48]:

	Spend
isWeekday	
False	903.160173
True	1077.363235

In [49]:
```
#observed test statistic
observed_difference_clean = means_table_clean.diff().iloc[-1,0]
observed_difference_clean
```

Out[49]: 174.2030621339445

In [50]:
```
#simulation

N = 1000
results = []

for _ in range(N):
    #create shuffled dataframe
    s = weekday_and_spend_clean['Weekday'].sample(frac=1, replace=False).re
    shuffled = weekday_and_spend_clean.assign(weekday=s)

    #calculate difference of means and add to results array
    shuff_means_table = shuffled.groupby('weekday').mean()
    results.append(abs(shuff_means_table.diff().iloc[-1,0]))

diffs_of_means_clean = pd.Series(results)
```

In [51]:
```
pval = (diffs_of_means_clean >= observed_difference_clean).sum() / N
pval
```

Out[51]: 0.426

Conclusion Without Outliers

- with a p-value of less than 0.05, we reject the null hypothesis that there is no significant difference between the amount of money spent on ads shown on weekdays and weekends
- we accept the alternate hypothesis - we have found a **significant difference** between the observed distribution and one created by random chance
- the outliers have had a significant effect on the outcome of the test - the outliers themselves merit more analysis