```
In [ ]: # Initialize Otter
        import otter
        grader = otter.Notebook("hw03.ipynb")
```

1 Homework 3: Text Analysis Using Twitter

1.1 Cleaning and Exploring Twitter Data using REGEX

1.2 Due Date: Tuesday, July 5, 11:59 PM

1.3 Collaboration Policy

Data science is a collaborative activity. While you may talk with others about the homework, we ask that you **write your solutions individually**. If you do discuss the assignments with others please **include their names** below.

Collaborators: *list collaborators here*

1.4 This Assignment

Welcome to the third homework assignment of Data 100! In this assignment, we will be exploring tweets from several high profile Twitter users.

In this assignment you will gain practice with: * Conducting Data Cleaning and EDA on a text-based dataset. * Manipulating data in pandas with the datetime and string accessors. * Writing regular expressions and using pandas regex methods. * Performing sentiment analysis on social media using VADER.

```
In [1]: # Run this cell to set up your notebook
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import re

        from ds100_utils import *

        # Ensure that Pandas shows at least 280 characters in columns, so we can see full tweets
        pd.set_option('max_colwidth', 280)
        plt.style.use('fivethirtyeight')
        sns.set()
```

```python
sns.set_context("talk")

def horiz_concat_df(dict_of_df, head=None):
    """
    Horizontally concatenante multiple DataFrames for easier visualization.
    Each DataFrame must have the same columns.
    """
    df = pd.concat([df.reset_index(drop=True) for df in dict_of_df.values()], axis=1, keys=dict_
    if head is None:
        return df
    return df.head(head)
```

1.4.1 Score Breakdown

Question	Points
1a	1
1b	1
1c	3
1d	1
2a	2
2b	2
2c	2
2d	2
2e	2
2f	1
3a	1
3b	1
3c	1
4a	1
4b	1
4ci	1
4cii	1
4d	1
4e	2
4f	2
4g	2
5a	2
5b	2
Total	**35**

1.5 Question 1: Importing the Data

The data for this assignment was obtained using the Twitter APIs. To ensure that everyone has the same data and to eliminate the need for every student to apply for a Twitter developer account, we have collected a sample of tweets from several high-profile public figures. The data is stored in the folder `data`. Run the following cell to list the contents of the directory:

```
In [2]: # just run this cell
        from os import listdir
        for f in listdir("data"):
            print(f)
```

```
AOC_recent_tweets.txt
BernieSanders_recent_tweets.txt
BillGates_recent_tweets.txt
Cristiano_recent_tweets.txt
EmmanuelMacron_recent_tweets.txt
elonmusk_recent_tweets.txt
```

1.5.1 Question 1a

Let's examine the contents of one of these files. Using the `open` function and `read` operation on a python file object, read the first 1000 **characters** in `data/BernieSanders_recent_tweets.txt` and store your result in the variable `q1a`. Then display the result so you can read it.

Caution: Viewing the contents of large files in a Jupyter notebook could crash your browser. Be careful not to print the entire contents of the file.

Hint: You might want to try to use `with`:

```
with open("filename", "r") as f:
    f.read(2)
```

```
In [3]: q1a = ...
        # BEGIN SOLUTION NO PROMPT
        with open("data/BernieSanders_recent_tweets.txt", 'r') as f:
            q1a = f.read(1000)
        print(q1a)
        # END SOLUTION
```


[{"created_at": "Sat Feb 06 22:43:03 +0000 2021", "id": 1358184460794163202, "id_str": "135818446079416


```
In [ ]: grader.check("q1a")
```

1.5.2 Question 1b

What format is the data in? Answer this question by entering the letter corresponding to the right format in the variable `q1b` below.

A. CSV **B.** HTML **C.** JavaScript Object Notation (JSON) **D.** Excel XML

Answer in the following cell. Your answer should be a string, either `"A"`, `"B"`, `"C"`, or `"D"`.

In [115]:
```
q1b = "..."
q1b = 'C' # SOLUTION NO PROMPT
```

In [ ]:
```
grader.check("q1b")
```

1.5.3 Question 1c

Pandas has built-in readers for many different file formats including the file format used here to store tweets. To learn more about these, check out the documentation for `pd.read_csv`, `pd.read_html`, `pd.read_json`, and `pd.read_excel`.

1. Use one of these functions to populate the `tweets` dictionary with the tweets for: `AOC`, `Cristiano`, and `elonmusk`. The keys of `tweets` should be the handles of the users, which we have provided in the cell below, and the values should be the DataFrames.
2. Set the index of each DataFrame to correspond to the `id` of each tweet.

Hint: You might want to first try loading one of the DataFrames before trying to complete the entire question.

In [11]:
```
tweets = {
    "AOC": ...,
    "Cristiano": ...,
    "elonmusk": ...,
}
# BEGIN SOLUTION NO PROMPT
data_paths =  {
    "AOC": "data/AOC_recent_tweets.txt",
    "Cristiano": "data/Cristiano_recent_tweets.txt",
    "elonmusk": "data/elonmusk_recent_tweets.txt",
```

```
    }
    tweets = {
        name: pd.read_json(data_paths[name], orient="records").set_index("id")
        for name in data_paths
    }
    # END SOLUTION
```

```
In [ ]: grader.check("q1c")
```

If you did everything correctly, the following cells will show you the first 5 tweets for Elon Musk (and a lot of information about those tweets).

```
In [20]: # just run this cell
         tweets["elonmusk"].head()
```

```
Out[20]:                                    created_at               id_str  \
         id
         1357991946082418690 2021-02-06 09:58:04+00:00  1357991946082418688
         1357973565413367808 2021-02-06 08:45:02+00:00  1357973565413367808
         1357972904663687173 2021-02-06 08:42:25+00:00  1357972904663687168
         1357970517165182979 2021-02-06 08:32:55+00:00  1357970517165182976
         1357964347813687296 2021-02-06 08:08:24+00:00  1357964347813687296

                                                                         full_text  \
         id
         1357991946082418690        The Second Last Kingdom https://t.co/Je4EI88HmV
         1357973565413367808  @DumDin7 @Grimezsz Haven’ t heard that name in years …
         1357972904663687173                                     @Grimezsz Dogecake
         1357970517165182979                         YOLT\n\nhttps://t.co/cnOf9yjpF1
         1357964347813687296                             @Kristennetten That’ s Damian

                              truncated display_text_range  \
         id
         1357991946082418690      False            [0, 23]
         1357973565413367808      False           [19, 53]
         1357972904663687173      False           [10, 18]
         1357970517165182979      False            [0, 29]
         1357964347813687296      False           [15, 28]


         id
         1357991946082418690  {'hashtags': [], 'symbols': [], 'user_mentions': [], 'urls': [], 'media':
         1357973565413367808  {'hashtags': [], 'symbols': [], 'user_mentions': [{'screen_name': 'DumDin
         1357972904663687173
         1357970517165182979                                                        {'hashtags': [], 'syml
         1357964347813687296


         id
```

```
1357991946082418690  {'media': [{'id': 1357991942471094275, 'id_str': '1357991942471094275', '
1357973565413367808
1357972904663687173
1357970517165182979
1357964347813687296


id
1357991946082418690  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for i
1357973565413367808  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for i
1357972904663687173  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for i
1357970517165182979  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for i
1357964347813687296  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for i

                     in_reply_to_status_id  in_reply_to_status_id_str  …  \
id                                                                     …
1357991946082418690                    NaN                        NaN  …
1357973565413367808           1.357973e+18               1.357973e+18  …
1357972904663687173           1.357835e+18               1.357835e+18  …
1357970517165182979                    NaN                        NaN  …
1357964347813687296           1.357964e+18               1.357964e+18  …

                     favorite_count  favorited retweeted possibly_sensitive  \
id
1357991946082418690          352096      False     False                0.0
1357973565413367808            2155      False     False                NaN
1357972904663687173            5373      False     False                NaN
1357970517165182979           62717      False     False                0.0
1357964347813687296            5726      False     False                NaN

                     lang  retweeted_status  quoted_status_id  \
id
1357991946082418690    en               NaN               NaN
1357973565413367808    en               NaN               NaN
1357972904663687173    en               NaN               NaN
1357970517165182979    en               NaN               NaN
1357964347813687296    en               NaN               NaN

                     quoted_status_id_str  quoted_status_permalink  \
id
1357991946082418690                   NaN                      NaN
1357973565413367808                   NaN                      NaN
1357972904663687173                   NaN                      NaN
1357970517165182979                   NaN                      NaN
1357964347813687296                   NaN                      NaN

                     quoted_status
id
1357991946082418690            NaN
1357973565413367808            NaN
1357972904663687173            NaN
1357970517165182979            NaN
1357964347813687296            NaN
```

```
[5 rows x 30 columns]
```

1.5.4 Question 1d

There are many ways we could choose to read tweets. Why might someone be interested in doing data analysis on tweets? Name a kind of person or institution which might be interested in this kind of analysis. Then, give two reasons why a data analysis of tweets might be interesting or useful for them. Answer in 2-3 sentences.

Type your answer here, replacing this text.

SOLUTION: Any answer with thoughtful analysis should receive full credit. An example solution is:

"Many influential figures are using Twitter as a main form of communication, so tweets are an important source of data. Financial analysts are one group of people who may be interested in analyzing tweets in order to predict changes in stock price or understand policy changes that impact the economy."

1.6 Question 2: Source Analysis

In some cases, the Twitter feed of a public figure may be partially managed by a public relations firm. In these cases, the device used to post the tweet may help reveal whether it was the individual (e.g., from an iPhone) or a public relations firm (e.g., TweetDeck). The tweets we have collected contain the source information but it is formatted strangely :(

In [21]:
```python
# just run this cell
tweets["Cristiano"][["source"]]
```

Out[21]:
```
id
1358137564587319299  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iI
1357379984399212545  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iI
1356733030962987008  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iI
1355924395064233986  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iI
1355599316300292097  <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iI
…
32514882561638401                                   <a href="http://www.whosay.com" rel="nofollow">WI
32513604662071296                                   <a href="http://www.whosay.com" rel="nofollow">WI
32511823722840064                                   <a href="http://www.whosay.com" rel="nofollow">WI
32510294081146881                                   <a href="http://www.whosay.com" rel="nofollow">WI
32508748819857410                                   <a href="http://www.whosay.com" rel="nofollow">WI
```

```
[3198 rows x 1 columns]
```

In this question we will use a regular expression to convert this messy HTML snippet into something more readable. For example: `<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>` should be `Twitter for iPhone`.

1.6.1 Question 2a

We will first use the Python `re` library to cleanup the above test string. In the cell below, write a regular expression that will match the **HTML tag** and assign it to the variable `q2a_pattern`. We then use the `re.sub` function to substitute anything that matches the pattern with an empty string `""`.

An HTML tag is defined as a `<` character followed by zero or more non-`>` characters, followed by a `>` character. That is, `<a>` and `</a>` are both considered *separate* HTML tags.

```
In [22]: q2a_pattern = r"..."
         # BEGIN SOLUTION NO PROMPT
         q2a_pattern = r"<[^>]*>"
         # END SOLUTION
         test_str = '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
         re.sub(q2a_pattern, "", test_str)
```

```
Out[22]: 'Twitter for iPhone'
```

```
In [ ]: grader.check("q2a")
```

1.6.2 Question 2b

Rather than writing a regular expression to detect and remove the HTML tags we could instead write a regular expression to **capture** the device name between the angle brackets. Here we will use **capturing groups** by placing parenthesis around the part of the regular expression we want to return. For example, to capture the `21` in the string `08/21/83` we could use the pattern `r"08/(..)/83"`.

Hint: The output of the following cell should be `['Twitter for iPhone']`.

```
In [28]: q2b_pattern = r"..."
         # BEGIN SOLUTION NO PROMPT
         q2b_pattern = r">([^<]*)<"
         # END SOLUTION
         test_str = '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a>
         re.findall(q2b_pattern, test_str)
```

```
Out[28]: ['Twitter for iPhone']
```

```
In [ ]: grader.check("q2b")
```

1.6.3 Question 2c

Using either of the two regular expressions you just created and `Series.str.replace` or `Series.str.extract`, add a new column called `"device"` to **all** of the DataFrames in `tweets` containing just the text describing the device (without the HTML tags).

```
In [33]: """ # BEGIN PROMPT
         tweets = ...
         """ # END PROMPT
         # BEGIN SOLUTION NO PROMPT
         tweets = {
             name: tweets[name].assign(device = tweets[name]['source'].str.extract(r">([^<]*)<"))
             for name in tweets
         }
         # END SOLUTION
```

```
In [ ]: grader.check("q2c")
```

1.6.4 Question 2d

To examine the most frequently used devices by each individual, implement the `most_freq` function that takes in a `Series` and returns a new `Series` containing the `k` most commonly occurring entries in the first series, where the values are the counts of the entries and the indices are the entries themselves.

For example:

```
most_freq(pd.Series(["A", "B", "A", "C", "B", "A"]), k=2)
```

would return:

```
A    3
B    2
dtype: int64
```

Hint: Consider using `value_counts`, `sort_values`, `head`, and/or `nlargest` (for the last one, read the documentation here). Think of what might be the most efficient implementation.

```
In [42]: def most_freq(series, k = 5):
             ...
             # BEGIN SOLUTION NO PROMPT
             return series.value_counts().nlargest(k)
             # END SOLUTION

         most_freq(tweets["Cristiano"]['device'])
```

```
Out[42]: Twitter for iPhone     1183
         Twitter Web Client      959
         WhoSay                  453
         MobioINsider.com        144
         Twitter for Android     108
         Name: device, dtype: int64
```

```
In [ ]: grader.check("q2d")
```

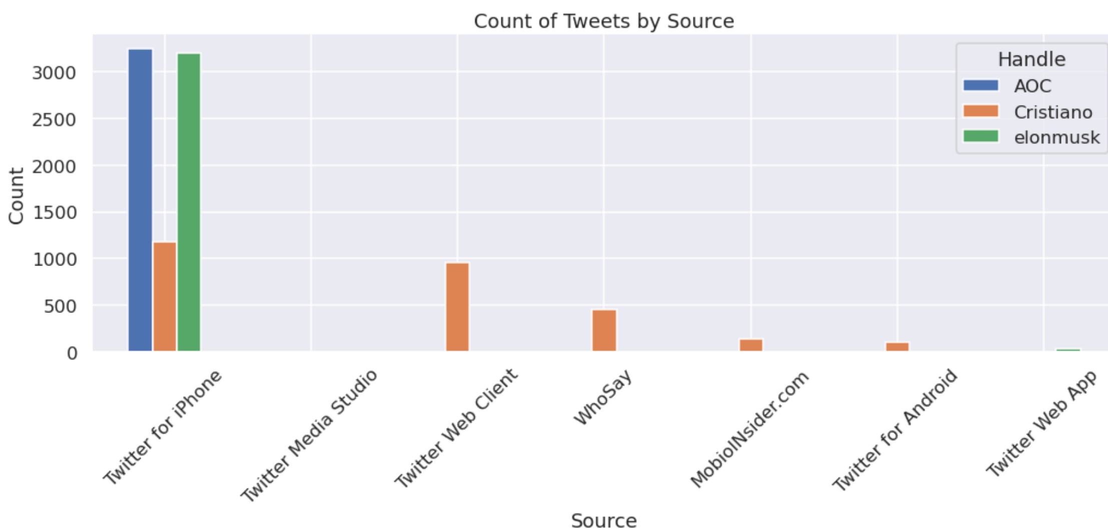
Run the following two cells to compute a table and plot describing the top 5 most commonly used devices for each user.

```
In [47]: # just run this cell
         device_counts = pd.DataFrame(
             [most_freq(tweets[name]['device']).rename(name)
              for name in tweets]
         ).fillna(0)
         device_counts
```

```
Out[47]:            Twitter for iPhone  Twitter Media Studio  Twitter Web Client  \
         AOC                    3245.0                   2.0                 0.0
         Cristiano              1183.0                   0.0               959.0
```

```
elonmusk                 3202.0                    0.0                 0.0

             WhoSay  MobioINsider.com  Twitter for Android  Twitter Web App
AOC             0.0               0.0                  0.0              0.0
Cristiano     453.0             144.0                108.0              0.0
elonmusk        0.0               0.0                  0.0             37.0
```

In [48]:
```python
# just run this cell
make_bar_plot(device_counts.T, title="Count of Tweets by Source",
               xlabel="Source", ylabel="Count")
plt.xticks(rotation=45)
plt.legend(title="Handle");
```



1.6.5 Question 2e

What might we want to investigate further? Write a few sentences below.

Type your answer here, replacing this text.

SOLUTION: Some correct answers include exploring the few tweets written on Twitter Media Studio by AOC or the 37 tweets Elon Musk used the Twitter Web App to write. Also what is going on with Cristiano, why does he use so many devices?

1.6.6 Question 2f

We just looked at the top 5 most commonly used devices for each user. However, we used the number of tweets as a measure, when it might be better to compare these distributions by comparing *proportions* of tweets. Why might proportions of tweets be better measures than numbers of tweets?

Type your answer here, replacing this text.

SOLUTION: Proportion of tweets provides a fairer comparison than number of tweets because some users will tend to tweet more often than the others. Since our interest is comparing the amount of tweets per source across different users, we need a measure that can scale them to the same level.

1.7 Question 3: When?

Now that we've explored the sources of each of the tweets, we will perform some time series analysis. A look into the temporal aspect of the data could reveal insights about how a user spends their day, when they eat and sleep, etc. In this question, we will focus on the time at which each tweet was posted.

1.7.1 Question 3a

Complete the following function `add_hour` that takes in a tweets dataframe `df`, and two column names `time_col` and `result_col`. Your function should use the timestamps in the `time_col` column to store in a new column `result_col` the computed hour of the day as a floating point number according to the formula:

$$\text{hour} + \frac{\text{minute}}{60} + \frac{\text{second}}{60^2}$$

Note: The below code calls your `add_hour` function and updates each tweets dataframe by using the `created_at` timestamp column to calculate and store the `hour` column.

Hint: See the following link for an example of working with timestamps using the `dt` accessors.

```
In [49]: def add_hour(df, time_col, result_col):
             ...
```

```
    # BEGIN SOLUTION NO PROMPT
    df[result_col] = df[time_col].dt.hour + df[time_col].dt.minute / 60 + df[time_col].dt.secon
    # END SOLUTION
    return df

# do not modify the below code
tweets = {handle: add_hour(df, "created_at", "hour") for handle, df in tweets.items()}
tweets["AOC"]["hour"].head()
```

Out[49]:
```
id
1358149122264563712    20.377222
1358147616400408576    20.277500
1358145332316667909    20.126389
1358145218407759875    20.118611
1358144207333036040    20.051667
Name: hour, dtype: float64
```

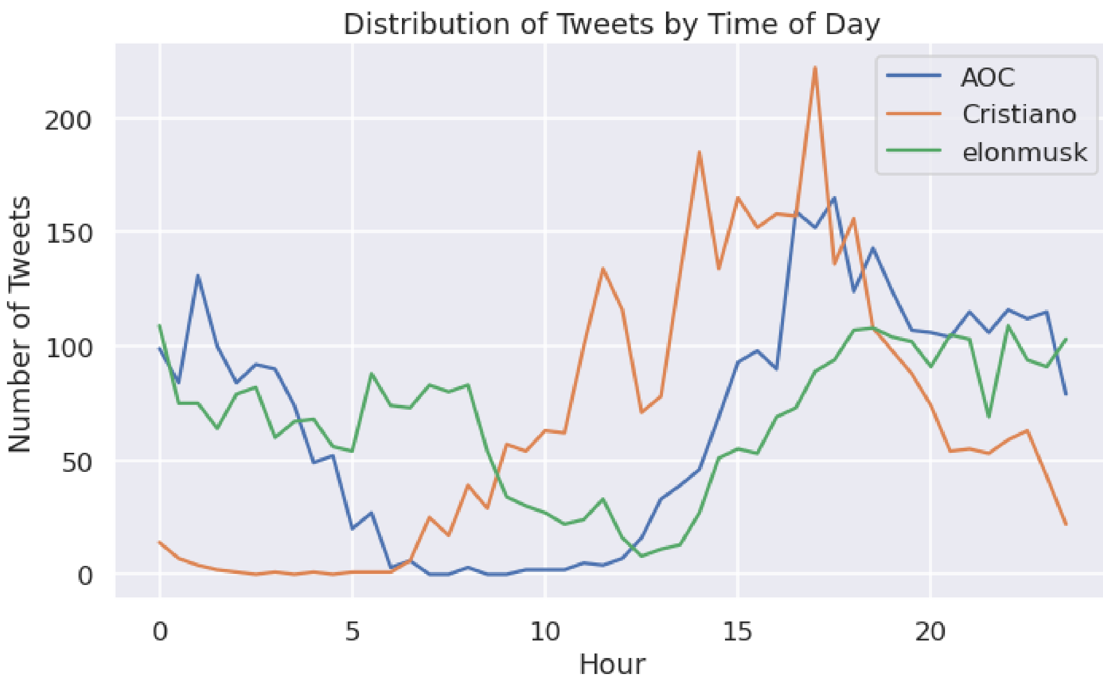
In [ ]:
```
grader.check("q3a")
```

With our new **hour** column, let's take a look at the distribution of tweets for each user by time of day. The following cell helps create a density plot on the number of tweets based on the hour they are posted.

The function `bin_df` takes in a dataframe, an array of bins, and a column name; it bins the the values in the specified column, returning a dataframe with the bin lower bound and the number of elements in the bin. This function uses `pd.cut`, a pandas utility for binning numerical values that you may find helpful in the distant future.

Run the cell and answer the following question about the plot.

In [55]:
```
# just run this cell
def bin_df(df, bins, colname):
    binned = pd.cut(df[colname], bins).value_counts().sort_index()
    return pd.DataFrame({"counts": binned, "bin": bins[:-1]})

hour_bins = np.arange(0, 24.5, .5)
binned_hours = {handle: bin_df(df, hour_bins, "hour") for handle, df in tweets.items()}

make_line_plot(binned_hours, "bin", "counts", title="Distribution of Tweets by Time of Day",
               xlabel="Hour", ylabel="Number of Tweets")
```



1.7.2 Question 3b

Compare Cristiano's distribution with those of AOC and Elon Musk. In particular, compare the distributions before and after Hour 6. What differences did you notice? What might be a possible cause of that? Do the data plotted above seem reasonable?

Type your answer here, replacing this text.

SOLUTION: As we can see, before hour 6, AOC and Elon Musk tweet more than Cristiano. After hour 6, there is an increase in the number of Cristinao's tweets, while AOC and Elon Musk has lesser tweets in comparison. This could be caused by the unaccounted timezone difference between Cristiano (Europe), AOC, and Elon Musk (US).

1.7.3 Question 3c

To account for different locations of each user in our analysis, we will next adjust the `created_at` timestamp for each tweet to the respective timezone of each user. Complete the following function `convert_timezone` that takes in a tweets dataframe `df` and a timezone `new_tz` and adds a new column `converted_time` that has the adjusted `created_at` timestamp for each tweet. The timezone for each user is provided in `timezones`.

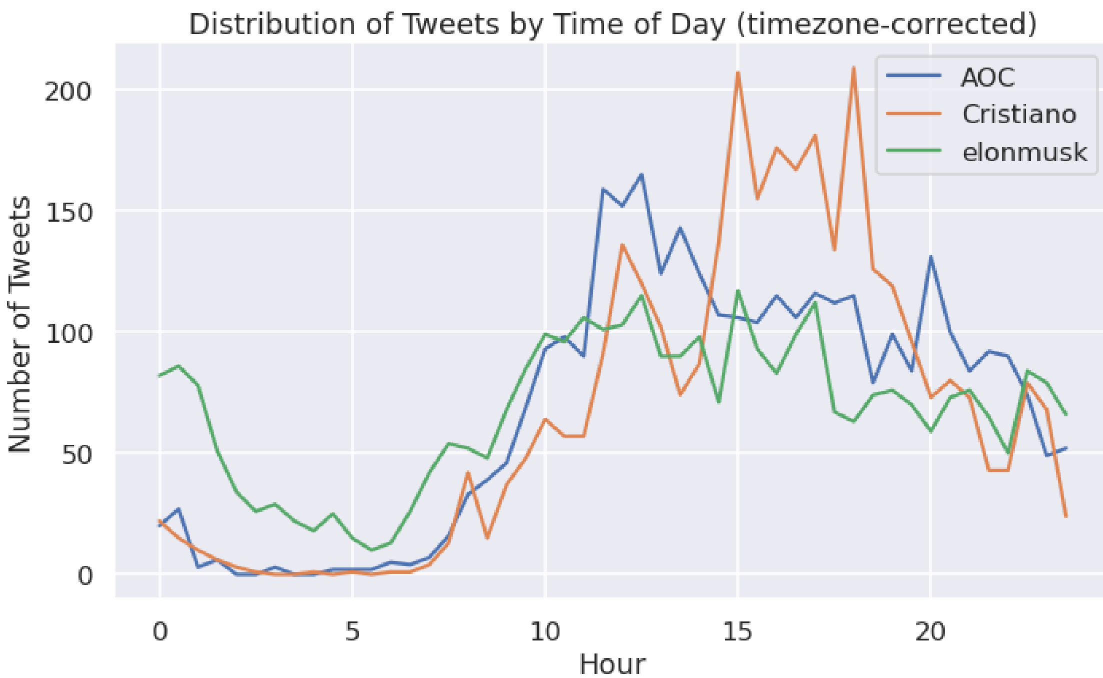
Hint: Again, please see the following link for an example of working with `dt` accessors.

In [56]:
```
def convert_timezone(df, new_tz):
    ...
    # BEGIN SOLUTION NO PROMPT
    df["converted_time"] = df["created_at"].dt.tz_convert(new_tz)
    # END SOLUTION
    return df

timezones = {"AOC": "EST", "Cristiano": "Europe/Lisbon", "elonmusk": "America/Los_Angeles"}

tweets = {handle: convert_timezone(df, tz) for (handle, df), tz in zip(tweets.items(), timezon
```

In [ ]:
```
grader.check("q3c")
```

With our adjusted timestamps for each user based on their timezone, let's take a look again at the distribution of tweets by time of day.

In [61]:
```
# just run this cell
tweets = {handle: add_hour(df, "converted_time", "converted_hour") for handle, df in tweets.it
binned_hours = {handle: bin_df(df, hour_bins, "converted_hour") for handle, df in tweets.items

make_line_plot(binned_hours, "bin", "counts", title="Distribution of Tweets by Time of Day (ti
               xlabel="Hour", ylabel="Number of Tweets")
```



1.8 Question 4: Sentiment

In the past few questions, we have explored the sources of the tweets and when they are posted. Although on their own, they might not seem particularly intricate, combined with the power of regular expressions, they could actually help us infer a lot about the users. In this section, we will continue building on our past analysis and specifically look at the sentiment of each tweet – this would lead us to a much more direct and detailed understanding of how the users view certain subjects and people.

How do we actually measure the sentiment of each tweet? In our case, we can use the words in the text of a tweet for our calculation! For example, the word "love" within the sentence "I love America!" has a positive sentiment, whereas the word "hate" within the sentence "I hate taxes!" has a negative sentiment. In addition, some words have stronger positive / negative sentiment than others: "I love America." is more positive than "I like America."

We will use the VADER (Valence Aware Dictionary and sEntiment Reasoner) lexicon to analyze the sentiment of AOC's tweets. VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media which is great for our usage.

The VADER lexicon gives the sentiment of individual words. Run the following cell to show the first few rows of the lexicon:

```
In [62]: # just run this cell
         print(''.join(open("vader_lexicon.txt").readlines()[:10]))
```

```
$:        -1.5       0.80623        [-1, -1, -1, -1, -3, -1, -3, -1, -2, -1]
%)        -0.4       1.0198        [-1, 0, -1, 0, 0, -2, -1, 2, -1, 0]
%-)        -1.5       1.43178        [-2, 0, -2, -2, -1, 2, -2, -3, -2, -3]
&-:        -0.4       1.42829        [-3, -1, 0, 0, -1, -1, -1, 2, -1, 2]
&:        -0.7       0.64031        [0, -1, -1, -1, 1, -1, -1, -1, -1, -1]
( '}{' )        1.6       0.66332        [1, 2, 2, 1, 1, 2, 2, 1, 3, 1]
(%        -0.9       0.9434        [0, 0, 1, -1, -1, -1, -2, -2, -1, -2]
('-:        2.2       1.16619        [4, 1, 4, 3, 1, 2, 3, 1, 2, 1]
(':        2.3       0.9        [1, 3, 3, 2, 2, 4, 2, 3, 1, 2]
((-:        2.1       0.53852        [2, 2, 2, 1, 2, 3, 2, 2, 3, 2]
```

As you can see, the lexicon contains emojis too! Each row contains a word and the *polarity* of that word, measuring how positive or negative the word is.

1.8.1 VADER Sentiment Analysis

VADER is a tool that can quantitatively describe the polarity or "sentiment" of a word.

VADER doesn't "read" sentences, but works by parsing sentences into words, assigning a preset generalized score from their testing sets to each word separately.

VADER relies on humans to stabilize its scoring. The creators use Amazon Mechanical Turk, a crowdsourcing survey platform, to train its model. Its training data consists of a small corpus of tweets, New York Times editorials and news articles, Rotten Tomatoes reviews, and Amazon product reviews, tokenized using the natural language toolkit (NLTK). Each word in each dataset was reviewed and rated by at least 20 trained individuals who had signed up to work on these tasks through Mechanical Turk.

1.8.2 Question 4a

Please score the sentiment of one of the following words, using your own personal interpretation. No code is required for this question!

- police
- order

- Democrat
- Republican
- gun
- dog
- technology
- TikTok
- security
- face-mask
- science
- climate change
- vaccine

What score did you give it and why? Can you think of a situation in which this word would carry the opposite sentiment to the one you've just assigned?

Type your answer here, replacing this text.

SOLUTION: Any response which identifies a score, states whether it is negative or positive and gives an example of an opposite scenario, indicating that students have a grasp of the way in which sentiment of a word is context-dependent. Answers should be within 1-2 sentences, but no credit taken away for longer responses. Example responses:

- Gun: -0.8: in most cases the word Gun has negative connotations and hence would receive a score closer to the extreme negative. This word can convey the opposite sentiment if someone is referring to a water gun or toy gun, or if someone is speaking about guns as a supporter of the 2nd Amendment.
- Dog: 0.8: in most cases the word Dog has positive connotations as they are common pets loved by many and therefore receives a score close to the extreme positive. This word can convey the opposite sentiment if someone is being called a dog as an insult.

Optional (ungraded): Are there circumstances (e.g. certain kinds of language or data) when you might not want to use VADER? What features of human speech might VADER misrepresent or fail to capture?

1.8.3 Question 4b

Let's first load in the data containing all the sentiments. Read `vader_lexicon.txt` into a dataframe called `sent`. The index of the dataframe should be the words in the lexicon and should be named `token`. `sent` should have one column named `polarity`, storing the polarity of each word.

Hint: The `pd.read_csv` function may help here. Since the file is tab-separated, be sure to set `sep='\t'` in your call to `pd.read_csv`. The first token will be `$:`.

In [63]:
```
sent = ...
# BEGIN SOLUTION NO PROMPT
sent = pd.read_csv('vader_lexicon.txt', sep='\t',
                   usecols=[0, 1], header=None, names=['token', 'polarity'],
                   index_col='token')
# END SOLUTION
sent.head()
```

Out[63]:
```
       polarity
token
$:         -1.5
%)         -0.4
%-)        -1.5
&-:        -0.4
&:         -0.7
```

In [ ]:
```
grader.check("q4b")
```

1.8.4 Question 4c

Before further analysis, we will need some more tools that can help us extract the necessary information and clean our data.

Complete the following regular expressions that will help us match part of a tweet that we either (i) want to remove or (ii) are interested in learning more about.

Question 4c Part (i) Assign a regular expression to a new variable `punct_re` that captures all of the punctuations within a tweet. We consider punctuation to be any non-word, non-whitespace character.

Note: A word character is any character that is alphanumeric or an underscore. A whitespace character is any character that is a space, a tab, a new line, or a carriage return.

In [72]:
```
punct_re = r'...'
punct_re = r'[^\w\s]' # SOLUTION NO PROMPT

re.sub(punct_re, " ", tweets["AOC"].iloc[0]["full_text"])
```

Out[72]:
```
'RT  RepEscobar  Our country has the moral obligation and responsibility to reunite every singl
```

```
In [ ]: grader.check("q4ci")
```

Question 4c Part (ii) Assign a regular expression to a new variable `mentions_re` that matches any mention in a tweet. Your regular expression should use a capturing group to extract the user's username in a mention.

Hint: a user mention within a tweet always starts with the `@` symbol and is followed by a series of word characters (with no space in between). For more explanations on what a word character is, check out the **Note** section in Part 1.

```
In [77]: mentions_re = r'...'
         mentions_re = r'@([^\W]+)' # SOLUTION NO PROMPT

         re.findall(mentions_re, tweets["AOC"].iloc[0]["full_text"])
```

```
Out[77]: ['RepEscobar']
```

```
In [ ]: grader.check("q4cii")
```

1.8.5 Tweet Sentiments and User Mentions

As you have seen in the previous part of this question, there are actually a lot of interesting components that we can extract out of a tweet for further analysis! For the rest of this question though, we will focus on one particular case: the sentiment of each tweet in relation to the users mentioned within it.

To calculate the sentiments for a sentence, we will follow this procedure:

1. Remove the punctuation from each tweet so we can analyze the words.
2. For each tweet, find the sentiment of each word.
3. Calculate the sentiment of each tweet by taking the sum of the sentiments of its words.

1.8.6 Question 4d

Let's use our `punct_re` regular expression from the previous part to clean up the text a bit more! The goal here is to remove all of the punctuations to ensure words can be properly matched with those from VADER to actually calculate the full sentiment score.

Complete the following function `sanitize_texts` that takes in a table `df` and adds a new column `clean_text` by converting all characters in its original `full_text` column to lower case and replace all instances of punctuations with a space character.

```
In [82]: def sanitize_texts(df):
             df["clean_text"] = ...
             # BEGIN SOLUTION NO PROMPT
             df["clean_text"] = df["full_text"].str.lower().str.replace(punct_re, ' ', regex=True)
             # END SOLUTION
             return df

         tweets = {handle: sanitize_texts(df) for handle, df in tweets.items()}
         tweets["AOC"]["clean_text"].head()
```

```
Out[82]: id
         1358149122264563712
         1358147616400408576
         1358145332316667909
         1358145218407759875                                                      joe cunningham pledged to r
         1358144207333036040    what s even more gross is that mace takes corporate pac money \n\nshe s
         Name: clean_text, dtype: object
```

```
In [ ]: grader.check("q4d")
```

1.8.7 Question 4e

With the texts sanitized, we can now extract all the user mentions from tweets.

Complete the following function `extract_mentions` that takes in the **`full_text`** (not `clean_text`!) column from a tweets dataframe and uses `mentions_re` to extract all the mentions in a dataframe. The returned dataframe is: * single-indexed by the IDs of the tweets * has one row for each mention * has one column named `mentions`, which contains each mention in all lower-cased characters

Hint: There are several ways to approach this problem. Here is documentation for potentially useful functions: `str.extractall` (link) and `str.findall` (link), `dropna` (link), and `explode` (link).

```
In [87]: def extract_mentions(full_texts):
             mentions = ...
             # BEGIN SOLUTION NO PROMPT
             # solution 1 with str.extractall
             mentions1 = (full_texts
```

```
                    .str.lower()
                    .str.extractall(mentions_re)[0]
                    .rename("mentions")
                    .reset_index(level=1)
                   )
    # solution 2 with str.findall + explode
    mentions2 = (full_texts
                    .str.lower()
                    .str.findall(mentions_re)
                    .explode()
                    .dropna()      # findall returns [] --> NaN
                    .rename("mentions")
                    .to_frame()
                   )
    mentions = mentions1
    # END SOLUTION
    return mentions[["mentions"]]

# uncomment this line to help you debug
display(extract_mentions(tweets["AOC"]["full_text"]).head())

# do not modify the below code
mentions = {handle: extract_mentions(df["full_text"]) for handle, df in tweets.items()}
horiz_concat_df(mentions).head()
```

```
                         mentions
id
1358149122264563712    repescobar
1358147616400408576      rokhanna
1358130063963811840    jaketapper
1358130063963811840  repnancymace
1358130063963811840           aoc
```

Out[87]:

	AOC	Cristiano	elonmusk
	mentions	mentions	mentions
0	repescobar	sixpadhomegym	dumdin7
1	rokhanna	globe_soccer	grimezsz
2	jaketapper	pestanacr7	grimezsz
3	repnancymace	goldenfootofficial	kristennetten
4	aoc	herbalife	kristennetten

```
In [ ]: grader.check("q4e")
```

1.8.8 Tidying Up the Data

Now, let's convert the tweets into what's called a *tidy format* to make the sentiments easier to calculate. The `to_tidy_format` function implemented for you uses the `clean_text` column of each tweets dataframe to create a tidy table, which is:

- single-indexed by the IDs of the tweets, for every word in the tweet.
- has one column named `word`, which contains the individual words of each tweet.

Run the following cell to convert the table into the tidy format. Take a look at the first 5 rows from the "tidied" tweets dataframe for AOC and see if you can find out how the structure has changed.

Note: Although there is no work needed on your part, we have referenced a few more advanced pandas methods you might have not seen before – you should definitely look them up in the documentation when you have a chance, as they are quite powerful in restructuring a dataframe into a useful intermediate state!

```
In [93]: # just run this cell
         def to_tidy_format(df):
             tidy = (
                 df["clean_text"]
                 .str.split()
                 .explode()
                 .to_frame()
                 .rename(columns={"clean_text": "word"})
             )
             return tidy

         tidy_tweets = {handle: to_tidy_format(df) for handle, df in tweets.items()}
         tidy_tweets["AOC"].head()
```

```
Out[93]:                             word
         id
         1358149122264563712           rt
         1358149122264563712  repescobar
         1358149122264563712          our
         1358149122264563712      country
         1358149122264563712          has
```

1.8.9 Adding in the Polarity Score

Now that we have this table in the tidy format, it becomes much easier to find the sentiment of each tweet: we can join the table with the lexicon table.

The following `add_polarity` function adds a new `polarity` column to the `df` table. The `polarity` column contains the sum of the sentiment polarity of each word in the text of the tweet.

Note: Again, though there is no work needed on your part, it is important for you to go through how we set up this method and actually understand what each method is doing. In particular, see how we deal with missing data.

```
In [94]: # just run this cell
```

```
def add_polarity(df, tidy_df):
    df["polarity"] = (
        tidy_df
        .merge(sent, how='left', left_on='word', right_index=True)
        .reset_index()
        .loc[:, ['id', 'polarity']]
        .fillna(0)
        .groupby('id')
        .sum()
    )
    return df

tweets = {handle: add_polarity(df, tidy_df) for (handle, df), tidy_df in \
          zip(tweets.items(), tidy_tweets.values())}
tweets["AOC"][["clean_text", "polarity"]].head()
```

Out[94]:

```
id
1358149122264563712
1358147616400408576
1358145332316667909
1358145218407759875                                                    joe cunningham pledged to ne
1358144207333036040  what s even more gross is that mace takes corporate pac money \n\nshe s a

                     polarity
id
1358149122264563712       0.0
1358147616400408576       1.0
1358145332316667909       0.0
1358145218407759875       0.0
1358144207333036040      -6.4
```

1.8.10 Question 4f

Finally, with our polarity column in place, we can finally explore how the sentiment of each tweet relates to the user(s) mentioned in it.

Complete the following function `mention_polarity` that takes in a mentions dataframe `mentions` and the original tweets dataframe `df` and returns a series where the mentioned users are the index and the corresponding mean sentiment scores of the tweets mentioning them are the values.

Hint: You should consider joining tables together in this question.

In [95]:
```
def mention_polarity(df, mention_df):
```

```
    ...
    # BEGIN SOLUTION NO PROMPT
    return (
        pd.merge(mention_df, df["polarity"], left_index=True, right_index=True)
        .dropna()
        .groupby("mentions")['polarity'].mean()
    )
    # END SOLUTION

aoc_mention_polarity = mention_polarity(tweets["AOC"],mentions["AOC"]).sort_values(ascending=Fa
aoc_mention_polarity
```

```
Out[95]: mentions
         booker4ky          15.4
         texasaflcio        12.8
         davidscottjaffe    12.6
         teamwarren         12.6
         padmalakshmi       12.3
                            …
         meggiebaer         -8.6
         manhattanda       -10.8
         scotthech         -10.8
         repmarktakano     -10.8
         repchuygarcia     -10.8
         Name: polarity, Length: 1182, dtype: float64
```

```
In [ ]: grader.check("q4f")
```

1.8.11 Question 4g

When grouping by mentions and aggregating the polarity of the tweets, what aggregation function should we use? What might be one drawback of using the mean?

Type your answer here, replacing this text.

SOLUTION: We should use median (or randomly divide all the polarity scores into k groups, calculate the mean of each group, and then take the median of those means) as our aggregation function here. One particular drawbacks of using the mean is it would be sensitive to outliers within the data. If the user posted a few extremely positive/negative tweets, that could end up affecting our interpretation of the overall sentiment of the user's tweets.

1.9 Question 5: You Do EDA!

Congratulations! You have finished all of the preliminary analysis on AOC, Cristiano, and Elon Musk's recent tweets.

As you might have recognized, there is still far more to explore within the data and build upon what we have uncovered so far. In this open-ended question, we want you to come up with a new perspective that can expand upon our analysis of the sentiment of each tweet.

For this question, you will perform some text analysis on our `tweets` dataset. Your analysis should have two parts:

1. a piece of code that manipulates `tweets` in some way and produces informative output (e.g. a dataframe, series, or plot)
2. a short (4-5 sentence) description of the findings of your analysis: what were you looking for? What did you find? How did you go about answering your question?

Your work should involve text analysis in some way, whether that's using regular expressions or some other form.

To aid you in creating plots, we provide the plotting helper functions in the table below. These are same helpers we have used throughout this notebook, and all accept dictionaries with a similar structure to `tweets`. That being said, if you know how to make plots, please do so! Very soon in this class, you'll learn how to use the matplotlib and seaborn libraries that we use to write these the helpers.

Helper	Description
`make_bar_plot`	Plot side-by-side bar plots of data like `plt.bar`
`make_histogram`	Plot overlaid histograms of data like `plt.hist`
`make_line_plot`	Plot overlaid line plots of data like `plt.plot`
`make_scatter_plot`	Plot overlaid scatter plots of data like `plt.scatter`

Each of the provided helpers is in `ds100_utils.py` and has a comprehensive docstring. You can read the docstring by calling `help` on the plotting function:

```
In [101]: help(make_line_plot)
```

```
Help on function make_line_plot in module ds100_utils:

make_line_plot(df_dict, x_col, y_col, include=None, title=None, xlabel=None, ylabel=None, legend=True)
    Plot a line plot of two columns for each dataframe in `df_dict`.

    Uses `sns.lineplot` to plot a line plot of two columns for each
```

```
dataframe in `df_dict`. The keys of `df_dict` are used as entries in
the legend when `legend` is `True`.

Parameters
----------
    df_dict: dict[str: pd.DataFrame]
        a dictionary mapping handles to dataframes with the data to plot
    x_col: str
        the name of a column in each dataframe in `df_dict` to plot on
        the x-axis
    y_col: str
        the name of a column in each dataframe in `df_dict` to plot on
        the y-axis
    include: list[str], optional
        a list of handles to include in the plot; all keys in `df_dict` not
        present in `include`, if specified, will *not* be included in the plot
    title: str, optional
        a title for the plot
    xlabel: str, optional
        a label for the x-axis; if unspecified, `x_col` is used
    ylabel: str, optional
        a label for the y-axis; if unspecified, `y_col` is used
    legend: bool, optional
        whether to include a legend with each key in `df_dict`
```

To assist you in getting started, here are a few ideas for this you can analyze for this question:

- dig deeper into when devices were used
- how sentiment varies with time of tweet
- expand on regexes from 4b to perform additional analysis (e.g. hashtags)
- examine sentiment of tweets over time

In general, try to combine the analyses from earlier questions or create new analysis based on the scaffolding we have provided.

This question is worth 4 points and will be graded based on this rubric:

	2 points	1 point	0 points
Code	Produces a mostly informative plot or pandas output that addresses the question posed in the student's description and uses at least one of the following pandas DataFrame/Series methods: `groupby`, `agg`, `merge`, `pivot_table`, `str`, `apply`	Attempts to produce a plot or manipulate data but the output is unrelated to the proposed question, or doesn't utilize at least one of the listed methods	No attempt at writing code
Description	Describes the analysis question and procedure comprehensively and summarizes results correctly	Attempts to describe analysis and results but description of results is incorrect or analysis of results is disconnected from the student's original question	No attempt at writing a description

---

1.9.1 Question 5a

Use this space to put your EDA code.

```
In [102]: # perform your text analysis here
```

1.9.2 Question 5b

Use this space to put your EDA description.

Write your description here.

SOLUTION: See rubric above.

1.10 Congratulations! You have finished Homework 3!

To double-check your work, the cell below will rerun all of the autograder tests.

```
In [ ]: grader.check_all()
```

1.11 Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

```
In [ ]: # Save your notebook first, then run this cell to export your submission.
        grader.export()
```