Exploring Hacker News Posts

In this project, I will be working with a dataset of submissions to the popular technology site, Hacker News. The site was started by Y Combinator, a startup incubator, where user-submitted stories or posts receive votes and comments. Posts that make it to the top of the Hacker News listings can receive hundreds of thousands of visitors, making it an extremely popular platform in the technology and startup communities. The dataset contains approximately 20,000 rows and includes columns such as id, title, url, num_points, num_comments, author, and created_at. I am specifically interested in posts with titles that begin with Ask HN or Show HN, as these are users submitting questions or showcasing projects. My goal is to compare these two types of posts to determine which receive more comments on average, and to analyze whether posts created at a certain time receive more comments on average.

Now, I will open the dataset file and use to create a list of lists:

In [17]:
```
opened_file = open('hacker_news.csv')
from csv import reader
read_file = reader(opened_file)
hn = list(read_file)
```

Taking a look at the first five rows of the dataset:

In [18]:
```
hn[:4]
```

Out[18]:
```
[['id', 'title', 'url', 'num_points', 'num_comments', 'author', 'created_at'],
 ['12224879',
  'Interactive Dynamic Video',
  'http://www.interactivedynamicvideo.com/',
  '386',
  '52',
  'ne0phyte',
  '8/4/2016 11:52'],
 ['10975351',
  'How to Use Open Source and Shut the Fuck Up at the Same Time',
  'http://hueniverse.com/2016/01/26/how-to-use-open-source-and-shut-the-fuck-up-at-the-same-time/',
  '39',
  '10',
  'joeyp2',
  '1/26/2016 19:30'],
 ['11964716',
  'Florida DJs May Face Felony for April Fools' Water Joke',
  'http://www.thewire.com/entertainment/2013/04/florida-djs-april-fools-water-joke/63798/',
  '2',
  '1',
  'vezycash',
  '6/23/2016 22:20']]
```

Removing the header row to analyze the data:

In [19]:
```
headers = hn[0]
hn = hn[1:]
print(headers)
hn[:4]
```

```
['id', 'title', 'url', 'num_points', 'num_comments', 'author', 'created_at']
```

Out[19]:
```
[['12224879',
  'Interactive Dynamic Video',
  'http://www.interactivedynamicvideo.com/',
  '386',
  '52',
  'ne0phyte',
  '8/4/2016 11:52'],
 ['10975351',
  'How to Use Open Source and Shut the Fuck Up at the Same Time',
  'http://hueniverse.com/2016/01/26/how-to-use-open-source-and-shut-the-fuck-up-at-the-same-time/',
  '39',
  '10',
  'joeyp2',
  '1/26/2016 19:30'],
 ['11964716',
  'Florida DJs May Face Felony for April Fools' Water Joke',
  'http://www.thewire.com/entertainment/2013/04/florida-djs-april-fools-water-joke/63798/',
  '2',
  '1',
  'vezycash',
  '6/23/2016 22:20'],
 ['11919867',
  'Technology ventures: From Idea to Enterprise',
  'https://www.amazon.com/Technology-Ventures-Enterprise-Thomas-Byers/dp/0073523429',
  '3',
  '1',
  'hswarna',
  '6/17/2016 0:01']]
```

Extracting `Ask HN` and `Show HN` posts:

In [20]:
```
ask_posts = []
show_posts = []
other_posts = []
```

In [21]:
```
for row in hn:
    title = row[1]
    ask_prefix = "ask hn"
    show_prefix = "show hn"
    if title.lower().startswith(ask_prefix.lower()):
        ask_posts.append(row)
    elif title.lower().startswith(show_prefix.lower()):
        show_posts.append(row)
    else: other_posts.append(row)
```

In [22]:
```
print(len(ask_posts))
print(len(show_posts))
print(len(other_posts))
```

```
1744
1162
17194
```

The code above is iterating over each row of the dataset hn using a for loop. In each iteration, it extracts the second column of the row, which represents the title of a post. It then checks whether the title starts with either "ask hn" or "show hn" using the startswith() method. If the title starts with "ask hn", it adds the title to a list called `ask_post`. If it starts with "show hn", it adds the title to a list called `show_post`. If the title doesn't start with either prefix, it adds it to a list called `other_post`.

After each iteration, the code prints the current length of the `ask_post` list. The code also prints the final length of the `show_post` and `other_post` lists, which gives us the total number of posts that started with "show hn" and the total number of posts that didn't start with either "ask hn" or "show hn".

Calculating the Average Number of Comments for Ask HN and Show HN Posts

Now that I have separated ask posts and show posts into different lists, I will calculate the average number of comments each type of post receives.

In [23]:
```
total_ask_comments = 0

for row in ask_posts:
    num_comments = int(row[4])
    total_ask_comments += num_comments

avg_ask_comments = total_ask_comments / len(ask_posts)
print("Average number of comments for Ask HN Posts = ", round(avg_ask_comments))
```

```
Average number of comments for Ask HN Posts =  14
```

In [24]:
```
total_show_comments = 0

for row in show_posts:
    num_comments = int(row[4])
    total_show_comments += num_comments

avg_show_comments = total_show_comments / len(show_posts)
print("Average number of comments for Show HN Posts = ", round(avg_show_comments))
```

```
Average number of comments for Show HN Posts =  10
```

In this sample, ask posts receive an average of about 14 comments, while show posts receive about 10 comments. Given that ask posts tend to receive more comments, I will narrow my analysis to focus on this type of post.

Finding the Amount of Ask Posts and Comments by Hour Created

Next, I will determine if ask posts created at a certain time are more likely to attract comments. I will use the following steps to perform this analysis:

-Calculate the number of ask posts created in each hour of the day, along with the number of comments received.

-Calculate the average number of comments ask posts receive by hour created.

In [25]:
```
import datetime as dt

result_list = []

for post in ask_posts:
    result_list.append([post[6], int(post[4])])

counts_by_hour = {}
comments_by_hour = {}

for row in result_list:
    date_string = row[0]
    date_object = dt.datetime.strptime(date_string, "%m/%d/%Y %H:%M")
    hour = date_object.hour

    comments = int(row[1])

    if hour not in counts_by_hour:
        counts_by_hour[hour] = 1
        comments_by_hour[hour] = comments
    else:
        counts_by_hour[hour] += 1
        comments_by_hour[hour] += comments

comments_by_hour
```

Out[25]:
```
{9: 251,
 13: 1253,
 10: 793,
 14: 1416,
 16: 1814,
 23: 543,
 12: 687,
 17: 1146,
 15: 4477,
 21: 1745,
 20: 1722,
 2: 1381,
 18: 1439,
 3: 421,
 5: 464,
 19: 1188,
 1: 683,
 22: 479,
 8: 492,
 4: 337,
 0: 447,
 6: 397,
 7: 267,
 11: 641}
```

First, I imported the `datetime` module and gave it an alias of `dt`. Then, I created an empty list called `result_list`.

Using a for loop, I iterated over the `ask_posts` list and appended to `result_list` a sublist containing two elements: the date and time the post was created `(post[6])`, and the number of comments it received `(int(post[4]))`.

After that, I created two empty dictionaries called `counts_by_hour` and `comments_by_hour`.

I then looped through each sublist in `result_list` and extracted the hour from the date and time using the `strptime()` method from the `datetime` module. I stored the hour in a variable called `hour`.

I also extracted the number of comments from the sublist and stored it in a variable called `comments`.

Next, I used conditional statements to populate `counts_by_hour` and `comments_by_hour` dictionaries. If the hour was not in `counts_by_hour`, I added the hour as a key and set its value to 1 in `counts_by_hour`, and set the number of comments as the value for the corresponding hour in `comments_by_hour`. If the hour was already in the `counts_by_hour`, I incremented the value of that hour in `counts_by_hour` and added the number of comments to the existing value in `comments_by_hour`.

Finally, I printed out the `comments_by_hour` dictionary.

Calculating the Average Number of Comments for Ask HN Posts by Hour

In [26]:
```
avg_by_hour = []

for hr in comments_by_hour:
    avg_by_hour.append([hr, comments_by_hour[hr] / counts_by_hour[hr]])

avg_by_hour
```

Out[26]:
```
[[9, 5.577777777777775],
 [13, 14.741176470588234],
 [10, 13.440677966101696],
 [14, 13.233644859813085],
 [16, 16.796296296296298],
 [23, 7.985294117647059],
 [12, 9.410958904109589],
 [17, 11.46],
 [15, 38.594852758620689],
 [21, 16.009174311926607],
 [20, 21.525],
 [2, 21.810344827586206],
 [18, 13.201834862638532],
 [3, 7.7962962962962971],
 [5, 10.086956521739131],
 [19, 10.8],
 [1, 11.383333333333333],
 [22, 6.7464788732394437],
 [8, 10.25],
 [4, 7.1702127659574471],
 [0, 8.1272727272727271],
 [6, 9.0227272727272731],
 [7, 7.8529411764705881],
 [11, 11.051724137931034]]
```

In [27]:
```
swap_avg_by_hour = []

for row in avg_by_hour:
    swap_avg_by_hour.append([row[1], row[0]])

print(swap_avg_by_hour)
```

```
[[5.577777777777775, 9], [14.741176470588234, 13], [13.440677966101696, 10], [13.233644859813085, 14], [16.796296296296298, 16], [7.985294117647059, 23], [9.410958904109589, 12], [11.46, 17], [38.594827586206069, 15], [16.009174311926607, 21], [21.525, 20], [21.810344827586206, 2], [13.201834862638532, 18], [7.7962962962962971, 3], [10.086956521739131, 5], [10.8, 19], [11.383333333333333, 1], [6.7464788732394437, 22], [10.25, 8], [7.1702127659574471, 4], [8.1272727272727271, 0], [9.0227272727272731, 6], [7.8529411764705881, 7], [11.051724137931034, 11]]
```

In [32]:
```
sorted_swap = sorted(swap_avg_by_hour, reverse = True)
print('Top 5 Hours for Ask Post Comments')

for avg, hr in sorted_swap[:4]:
    print("{}: {:.2f} average comments per post".format(
        dt.datetime.strptime(str(hr), "%H").strftime("%H:%M"), avg))
```

```
Top 5 Hours for Ask Post Comments
15:00: 38.59 average comments per post
02:00: 23.81 average comments per post
20:00: 21.52 average comments per post
16:00: 16.80 average comments per post
```

In this section, I created two lists, `avg_by_hour` and `swap_avg_by_hour` using the `comments_by_hour` and `counts_by_hour lists`. The `avg_by_hour` list contains the average number of comments per post for each hour of the day, while the `swap_avg_by_hour` list is a modified version of `avg_by_hour` where the average number of comments and the corresponding hour have been swapped.

After printing the `swap_avg_by_hour` list, I sorted it in descending order based on the average number of comments per post using the sorted() function. Finally, I printed the top 5 hours for ask post comments and their corresponding average number of comments per post using string formatting and the `datetime` module to format the hour as HH:MM.

On average, the hour that attracts the highest number of comments per post is at 15:00, with an average of 38.59 comments per post. Additionally, there's a notable increase of about 60% in the number of comments between the hours with the highest and second-highest average number of comments.

Conclusion

In this project, I analyzed ask posts and show posts to determine which type of post and time receive the most comments on average. Based on my analysis, to maximize the amount of comments a post receives, I'd recommend the post be categorized as ask post and created between 15:00 and 16:00 (3:00 pm est - 4:00 pm est).

However, it's important to note that the data set I analyzed excluded posts without any comments. Therefore, it's more accurate to say that of the posts that received comments, ask posts received more comments on average and ask posts created between 15:00 and 16:00 (3:00 pm est - 4:00 pm est) received the most comments on average.