Hacker News Posts Analytics

In this project my interest is to explore and analyze Hacker News Posts to determine which of these two types of posts Ask HN and Show HN received more comments on the average and to also show if post created at certain time of the day received more comments than others on the average.

Exploring data to find if Ask HN or Show HN receives more comments on the average:

First I will read the hacker_news.csv file, assign the result to the variable HN and display the first five rows of HN.

In [1]:
```
from csv import reader

Opened_file = open('hacker_news.csv')
Read_file = reader(Opened_file)
HN = list(Read_file)
HN[:5]
```

Out[1]:
```
[['id', 'title', 'url', 'num_points', 'num_comments', 'author', 'created_a
t'],
 ['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',
  'josep2',
  '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/0073
523429',
  '3',
  '1',
  'hswarna',
  '6/17/2016 0:01']]
```

I will proceed to extract the first row of the data and name it Headers. I will also display the first five rows of HN after extracting the headers.

In [2]:
```
Headers = HN[0]
HN = HN[1:]
print(Headers)
print(HN[:5])
```

```
['id', 'title', 'url', 'num_points', 'num_comments', 'author', 'created_at']
[['12224879', 'Interactive Dynamic Video', 'http://www.interactivedynamicvide
o.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/20
16/01/26/how-to-use-open-source-and-shut-the-fuck-up-at-the-same-time/', '3
9', '10', 'josep2', '1/26/2016 19:30'], ['11964716', "Florida DJs May Face Fe
lony for April Fools' Water Joke", 'http://www.thewire.com/entertainment/201
3/04/florida-djs-april-fools-water-joke/63798/', '2', '1', 'vezycash', '6/23/
2016 22:20'], ['11919867', 'Technology ventures: From Idea to Enterprise', 'h
ttps://www.amazon.com/Technology-Ventures-Enterprise-Thomas-Byers/dp/00735234
29', '3', '1', 'hswarna', '6/17/2016 0:01'], ['10301696', 'Note by Note: The
Making of Steinway L1037 (2007)', 'http://www.nytimes.com/2007/11/07/movies/0
7stein.html?_r=0', '8', '2', 'walterbell', '9/30/2015 4:12']]
```

Looking at the data above, we can see that it contains the title which we can use to separate any title beginning with Ask HN and Show HN, the number of comments and the time post was created.

Let's extract Ask HN and Show HN Posts as below:

In [3]:
```
Ask_HN_posts = []
Show_HN_posts =[]
All_other_posts = []
for post in HN:
    title = post[1]
    if title.lower().startswith("ask hn"):
        Ask_HN_posts.append(post)
    elif title.lower().startswith("show hn"):
        Show_HN_posts.append(post)
    else:
        All_other_posts.append(post)
print(len(Ask_HN_posts))
print(len(Show_HN_posts))
print(len(All_other_posts))
```

```
1744
1162
17194
```

Total number of Ask HN posts = 1744 and that of Show HN posts = 1162. With these, we can calculate the average number of comments each of these posts receive:

In [5]:
```
total_Ask_HN_comments = 0
for post in Ask_HN_posts:
    total_Ask_HN_comments += int(post[4]) #Total number of comments in Ask HN
 posts
avg_Ask_HN_comments = total_Ask_HN_comments / len(Ask_HN_posts)
print(avg_Ask_HN_comments)
```

```
14.038417431192661
```

Also for Show HN posts, average number of comments is calucualted as below:

In [6]:
```
total_Show_HN_comments = 0

for post in Show_HN_posts:
    total_Show_HN_comments += int(post[4])
avg_Show_HN_comments = total_Show_HN_comments / len(Show_HN_posts)
print(avg_Show_HN_comments)
```

```
10.31669535283993
```

From the analysis, Ask HN posts receives an average of 14 comments per post while Show HN receives an average of 10 comments for each post.

Determining what time of the day are posts created to receive more comments:

I will base this analysis on Ask HN posts since we already know it receives more comments on the average. Now I want to find what time am I likely to create a post that will receive the most comments on the average:

I will proceed to determine the number of Ask posts created at each hour of the day and the number of comments received.

In [9]:
```
import datetime as dt
result_list = []
for post in Ask_HN_posts:
    result_list.append([post[6], int(post[4])])

comments_per_hour = {}
counts_per_hour = {}
date_format = "%m/%d/%Y %H:%M"

for the_row in result_list:
    date = the_row[0]
    comment = the_row[1]
    time = dt.datetime.strptime(date, date_format).strftime("%H")
    if time in counts_per_hour:
        comments_per_hour[time] += comment
        counts_per_hour[time] += 1
    else:
        comments_per_hour[time] = comment
        counts_per_hour[time] = 1
comments_per_hour
```

Out[9]:
```
{'00': 447,
 '01': 683,
 '02': 1381,
 '03': 421,
 '04': 337,
 '05': 464,
 '06': 397,
 '07': 267,
 '08': 492,
 '09': 251,
 '10': 793,
 '11': 641,
 '12': 687,
 '13': 1253,
 '14': 1416,
 '15': 4477,
 '16': 1814,
 '17': 1146,
 '18': 1439,
 '19': 1188,
 '20': 1722,
 '21': 1745,
 '22': 479,
 '23': 543}
```

Now we have the number of comments for each hour of the day for Ask HN posts.

Now I will calculate the average number of comments that Ask HN Posts received per hour:

In [10]:
```
avg_per_hour = []
for hr in comments_per_hour:
    avg_per_hour.append([hr, comments_per_hour[hr] / counts_per_hour[hr]])
avg_per_hour
```

Out[10]:
```
[['06', 9.022727272727273],
 ['02', 23.810344827586206],
 ['00', 8.127272727272727],
 ['17', 11.46],
 ['01', 11.383333333333333],
 ['18', 13.20183486238532],
 ['11', 11.051724137931034],
 ['21', 16.009174311926607],
 ['08', 10.25],
 ['14', 13.233644859813085],
 ['19', 10.8],
 ['10', 13.440677966101696],
 ['20', 21.525],
 ['23', 7.985294117647059],
 ['09', 5.5777777777777775],
 ['07', 7.852941176470588],
 ['15', 38.5948275862069],
 ['05', 10.08695652173913],
 ['13', 14.741176470588234],
 ['22', 6.746478873239437],
 ['03', 7.796296296296297],
 ['16', 16.796296296296298],
 ['12', 9.41095890410959],
 ['04', 7.170212765957447]]
```

Now let's tidy this up and arrange it in descending order:

In [11]:
```
order_avg_per_hour = []
for row in avg_per_hour:
    order_avg_per_hour.append([row[1], row[0]])
print(order_avg_per_hour)

sorted_order = sorted(order_avg_per_hour, reverse=True)
sorted_order
```

```
[[9.022727272727273, '06'], [23.810344827586206, '02'], [8.127272727272727,
'00'], [11.46, '17'], [11.383333333333333, '01'], [13.20183486238532, '18'],
[11.051724137931034, '11'], [16.009174311926607, '21'], [10.25, '08'], [13.23
3644859813085, '14'], [10.8, '19'], [13.440677966101696, '10'], [21.525, '2
0'], [7.985294117647059, '23'], [5.5777777777777775, '09'], [7.85294117647058
8, '07'], [38.5948275862069, '15'], [10.08695652173913, '05'], [14.7411764705
88234, '13'], [6.746478873239437, '22'], [7.796296296296297, '03'], [16.79629
6296296298, '16'], [9.41095890410959, '12'], [7.170212765957447, '04']]
```

Out[11]:
```
[[38.5948275862069, '15'],
 [23.810344827586206, '02'],
 [21.525, '20'],
 [16.796296296296298, '16'],
 [16.009174311926607, '21'],
 [14.741176470588234, '13'],
 [13.440677966101696, '10'],
 [13.233644859813085, '14'],
 [13.20183486238532, '18'],
 [11.46, '17'],
 [11.383333333333333, '01'],
 [11.051724137931034, '11'],
 [10.8, '19'],
 [10.25, '08'],
 [10.08695652173913, '05'],
 [9.41095890410959, '12'],
 [9.022727272727273, '06'],
 [8.127272727272727, '00'],
 [7.985294117647059, '23'],
 [7.852941176470588, '07'],
 [7.796296296296297, '03'],
 [7.170212765957447, '04'],
 [6.746478873239437, '22'],
 [5.5777777777777775, '09']]
```

Now I will print top three hours you should create a post during to have a higher chance of receiving comments:

In [12]:
```
print("Top 3 Hours for 'Ask HN' Comments")
for avg, hr in sorted_order[:3]:
    print("{}: {:.2f} average comments per post".format(dt.datetime.strptime(h
r, "%H").strftime("%H:%M"),avg))
```

```
Top 3 Hours for 'Ask HN' 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
```

Conclusion:

Hacker News Posts were analysed and the following conclusions arrived at:

1. Ask HN Posts received more comments on the average than Show HN posts.
2. Best time to create an Ask HN post to ensure higher chnace of receiving comments is 1pm MDT (Documentation for the data set shows that the time zone is Easter Time in the US).