Photography Stack Exchange Data Dump

- Fiona LU
- fiona0304.lu@gmail.com

INTRODUCTION

As a question and answer site, **Photography Stack Exchange** is actively used by enthusiast photographers who are either professional or amateur, with over 25,000 questions and 65,000 answers currently. This report is going to conduct several analysis by exploring a recent data dump from the site.

METHODOLOGY

By using Python, this study will firstly import and prepare the following data tables from *Stack Exchange(2023)* which was uploaded in Dec 2023 for a further analysis:

These tables are converted from XML to CSV:

- **Badges**, loaded dataframe please refer to Figure 1.

```
import pandas as pd
import xml.etree.ElementTree as ET
import numpy as np
import csv
pd.set_option("display.notebook_repr_html", False)

# PARSE XML
xml = ET.parse("/Users/jinglu/Desktop/Study/DEAKIN/SIT731 Data Wraggling/Asses"+
               "sment/HD8/photo.stackexchange.com/Badges.xml")
root = xml.getroot()

# CREATE CSV FILE
csvfile = open("Badges.csv",'w')
```

```
csvfile_writer = csv.writer(csvfile)

# ADD THE HEADER TO CSV FILE
csvfile_writer.writerow(["Id","UserID","Name","Date","Class","TagBased"])

for item in root.findall(".//row"):
    csv_line = [item.get("Id", ""),
                item.get("UserID", ""),
                item.get("Name", ""),
                item.get("Date", ""),
                item.get("Class", ""),
                item.get("TagBased", "")]
    csvfile_writer.writerow(csv_line)

csvfile.close()
```

- **Comments**, loaded dataframe please refer to Figure 2.

```
# PARSE XML
xml = ET.parse("/Users/jinglu/Desktop/Study/DEAKIN/SIT731 Data Wraggling/Asses"+
               "sment/HD8/photo.stackexchange.com/Comments.xml")
root = xml.getroot()

# CREATE CSV FILE
csvfile = open("Comments.csv",'w')
csvfile_writer = csv.writer(csvfile)

# ADD THE HEADER TO CSV FILE
csvfile_writer.writerow(["Id","PostID","Score","Text","CreationDate","UserDispl"+
                         "ayName","UserID","ContentLicense"])

for item in root.findall(".//row"):
    csv_line = [item.get("Id", ""),
                item.get("PostId", ""),
                item.get("Score", ""),
                item.get("Text", ""),
                item.get("CreationDate", ""),
                item.get("UserDisplayName", ""),
                item.get("UserId", ""),
                item.get("ContentLicense", "")]
    csvfile_writer.writerow(csv_line)
```

```
csvfile.close()
```

- **Posts**, loaded dataframe please refer to Figure 3.

```
# PARSE XML
xml = ET.parse("/Users/jinglu/Desktop/Study/DEAKIN/SIT731 Data Wraggling/Asses"+
               "sment/HD8/photo.stackexchange.com/Posts.xml")
root = xml.getroot()

# CREATE CSV FILE
csvfile = open("Posts.csv",'w')
csvfile_writer = csv.writer(csvfile)

# ADD THE HEADER TO CSV FILE
csvfile_writer.writerow(["Id","PostTypeId","AcceptedAnswerId ","ParentId",
                         "CreationDate","DeletionDate","Score","ViewCount",
                         "Body ","OwnerUserId","OwnerDisplayName",
                         "LastEditorUserId","LastEditorDisplayName",
                         "LastEditDate","LastActivityDate","Title", "Tags",
                         "AnswerCount","CommentCount","FavoriteCount",
                         "ClosedDate","CommunityOwnedDate","ContentLicense"])

for item in root.findall(".//row"):
    csv_line = [item.get("Id", ""),
                item.get("PostTypeId", ""),
                item.get("AcceptedAnswerId", ""),
                item.get("ParentId", ""),
                item.get("CreationDate", ""),
                item.get("DeletionDate", ""),
                item.get("Score", ""),
                item.get("ViewCount", ""),
                item.get("Body", ""),
                item.get("OwnerUserId", ""),
                item.get("OwnerDisplayName", ""),
                item.get("LastEditorUserId", ""),
                item.get("LastEditorDisplayName", ""),
                item.get("LastEditDate", ""),
                item.get("LastActivityDate", ""),
                item.get("Title", ""),
                item.get("Tags", ""),
                item.get("AnswerCount", ""),
                item.get("CommentCount", ""),
```

```
                item.get("FavoriteCount", ""),
                item.get("ClosedDate", ""),
                item.get("CommunityOwnedDate", ""),
                item.get("ContentLicense", "")]
    csvfile_writer.writerow(csv_line)

csvfile.close()
```

- **PostHistory**, loaded dataframe please refer to Figure 4.

```
# PARSE XML
xml = ET.parse("/Users/jinglu/Desktop/Study/DEAKIN/SIT731 Data Wraggling/Asses"+
               "sment/HD8/photo.stackexchange.com/PostHistory.xml")
root = xml.getroot()

# CREATE CSV FILE
csvfile = open("PostHistory.csv",'w')
csvfile_writer = csv.writer(csvfile)

# ADD THE HEADER TO CSV FILE
csvfile_writer.writerow(["Id","PostHistoryTypeId","PostId","RevisionGUID",
                         "CreationDate","UserId","UserDisplayName","Comment",
                         "Text","ContentLicense"])

for item in root.findall(".//row"):
    csv_line = [item.get("Id", ""),
                item.get("PostHistoryTypeId", ""),
                item.get("PostId", ""),
                item.get("RevisionGUID", ""),
                item.get("CreationDate", ""),
                item.get("UserId", ""),
                item.get("UserDisplayName", ""),
                item.get("Comment", ""),
                item.get("Text", ""),
                item.get("ContentLicense", "")]
    csvfile_writer.writerow(csv_line)

csvfile.close()
```

- **PostLinks**, loaded dataframe please refer to Figure 5.

```
# PARSE XML
xml = ET.parse("/Users/jinglu/Desktop/Study/DEAKIN/SIT731 Data Wraggling/Asses"+
               "sment/HD8/photo.stackexchange.com/PostLinks.xml")
root = xml.getroot()

# CREATE CSV FILE
csvfile = open("PostLinks.csv",'w')
csvfile_writer = csv.writer(csvfile)

# ADD THE HEADER TO CSV FILE
csvfile_writer.writerow(["Id","CreationDate","PostId ","RelatedPostId ","LinkTypeId"])

for item in root.findall(".//row"):
    csv_line = [item.get("Id", ""),
                item.get("CreationDate", ""),
                item.get("PostId", ""),
                item.get("RelatedPostId", ""),
                item.get("LinkTypeId", "")]
    csvfile_writer.writerow(csv_line)

csvfile.close()
```

- **Tags**, loaded dataframe please refer to Figure 6.

```
# PARSE XML
xml = ET.parse("/Users/jinglu/Desktop/Study/DEAKIN/SIT731 Data Wraggling/Asses"\
               "sment/HD8/photo.stackexchange.com/Tags.xml")
root = xml.getroot()

# CREATE CSV FILE
csvfile = open("Tags.csv",'w')
csvfile_writer = csv.writer(csvfile)

# ADD THE HEADER TO CSV FILE
csvfile_writer.writerow(["Id","TagName","Count","ExcerptPostId","WikiPostId",
                         "IsModeratorOnly","IsRequired"])

for item in root.findall(".//row"):
    csv_line = [item.get("Id", ""),
                item.get("TagName", ""),
                item.get("Count", ""),
                item.get("ExcerptPostId", ""),
```

```
                item.get("WikiPostId", ""),
                item.get("IsModeratorOnly", ""),
                item.get("IsRequired", "")]
    csvfile_writer.writerow(csv_line)

csvfile.close()
```

- **Users**, loaded dataframe please refer to Figure 7.

'EmailHash' in the *Users* table is a good evidence showing data privacy and ethics. According to the schema discriptions, the 'EmailHash' column is now always blank to protect users information to avoid disclosure risks.

```
# PARSE XML
xml = ET.parse("/Users/jinglu/Desktop/Study/DEAKIN/SIT731 Data Wraggling/Asses"+
               "sment/HD8/photo.stackexchange.com/Users.xml")
root = xml.getroot()

# CREATE CSV FILE
csvfile = open("Users.csv",'w')
csvfile_writer = csv.writer(csvfile)

# ADD THE HEADER TO CSV FILE
csvfile_writer.writerow(["Id","Reputation","CreationDate ","DisplayName",
                         "LastAccessDate","WebsiteUrl","Location","AboutMe",
                         "Views ","UpVotes","DownVotes", "ProfileImageUrl",
                         "EmailHash","AccountId"])

for item in root.findall(".//row"):
    csv_line = [item.get("Id", ""),
                item.get("Reputation", ""),
                item.get("CreationDate", ""),
                item.get("DisplayName", ""),
                item.get("LastAccessDate", ""),
                item.get("WebsiteUrl", ""),
                item.get("Location", ""),
                item.get("AboutMe", ""),
                item.get("Views", ""),
                item.get("UpVotes", ""),
                item.get("DownVotes", ""),
                item.get("ProfileImageUrl", ""),
                item.get("EmailHash", ""),
                item.get("AccountId", "")]
```

```
    csvfile_writer.writerow(csv_line)

csvfile.close()
```

- **Votes**, loaded dataframe please refer to Figure 8.

```
# PARSE XML
xml = ET.parse("/Users/jinglu/Desktop/Study/DEAKIN/SIT731 Data Wraggling/Asses"\
               "sment/HD8/photo.stackexchange.com/Votes.xml")
root = xml.getroot()

# CREATE CSV FILE
csvfile = open("Votes.csv",'w')
csvfile_writer = csv.writer(csvfile)

# ADD THE HEADER TO CSV FILE
csvfile_writer.writerow(["Id","PostId","VoteTypeId ","UserId","CreationDate",
                         "BountyAmount"])

for item in root.findall(".//row"):
    csv_line = [item.get("Id", ""),
                item.get("PostId", ""),
                item.get("VoteTypeId", ""),
                item.get("UserId", ""),
                item.get("CreationDate", ""),
                item.get("BountyAmount", "")]
    csvfile_writer.writerow(csv_line)

csvfile.close()
```

The CSV files are loaded as Pandas dataframes:

- **Badges**

```
Badges = pd.read_csv('Badges.csv', comment="#")
Badges.head(3)
```

- **Comments**

```
Comments = pd.read_csv('Comments.csv', comment="#")
Comments.head(3)
```

```
   Id  UserID            Name                      Date  Class  TagBased
0   1     NaN  Autobiographer  2010-07-15T19:05:50.707      3     False
1   2     NaN  Autobiographer  2010-07-15T19:05:50.707      3     False
2   3     NaN  Autobiographer  2010-07-15T19:05:50.707      3     False
```

Figure 1: Badge table - loaded Pandas dataframe

```
   Id  PostID  Score                                              Text  \
0   1       2      0  Are you asking for how you can simulate this e...
1   2      10      0  I own a couple Gorillapods (SLR & SLR-Zoom) an...
2   3       8      3  I wish I could upvote this a thousand times.  ...

              CreationDate UserDisplayName  UserID ContentLicense
0  2010-07-15T19:24:02.130             NaN    23.0   CC BY-SA 2.5
1  2010-07-15T19:28:43.197             NaN    28.0   CC BY-SA 2.5
2  2010-07-15T19:30:32.753          Jessie     NaN   CC BY-SA 2.5
```

Figure 2: Comments table - loaded Pandas dataframe

- **Posts**

```
Posts = pd.read_csv('Posts.csv', comment="#")
Posts.head(3)
```

- **PostsHistory**

```
PostHistory = pd.read_csv('PostHistory.csv', on_bad_lines='skip')
PostHistory.head(3)
```

- **PostLinks**

```
PostLinks = pd.read_csv('PostLinks.csv', comment="#")
PostLinks.head(3)
```

- **Tags**

```
Tags = pd.read_csv('Tags.csv', comment="#")
Tags.head(3)
```

```
   Id  PostTypeId  AcceptedAnswerId   ParentId            CreationDate  \
0   1           1            9984.0       NaN  2010-07-15T19:19:59.877
1   2           1               NaN       NaN  2010-07-15T19:20:38.567
2   3           1              89.0       NaN  2010-07-15T19:21:55.490

   DeletionDate  Score  ViewCount  \
0           NaN     57    24293.0
1           NaN     48    17261.0
2           NaN     33     6871.0

                                                Body   OwnerUserId  ...  \
0  <p>I have a Canon 7D with a 50mm f/1.4 lens, a...          6.0  ...
1  <p>I know that this effect occurs when there's...          9.0  ...
2  <p>All my attempts at HDR come out looking rem...         21.0  ...

             LastEditDate         LastActivityDate  \
0  2019-08-07T07:25:42.660  2019-08-07T07:25:42.660
1  2013-06-03T15:10:12.677  2022-01-11T20:46:23.567
2  2012-08-06T15:31:00.220  2019-09-05T15:12:33.887

                                                Title  \
0  What methods can be used to micro-adjust autof...
1  How can I maximize the "blurry background, sha...
2       How can I stop my HDR shots looking so fake?

                                                Tags AnswerCount CommentCount  \
0  <autofocus><back-focus><tests><focus-adjust><f...         5.0          2.0
1  <depth-of-field><blur><bokeh><shooting-techniq...        20.0          6.0
2                <technique><hdr><artifacts><halos>         7.0          5.0

  FavoriteCount  ClosedDate  CommunityOwnedDate  ContentLicense
0           NaN         NaN                 NaN    CC BY-SA 4.0
1           NaN         NaN                 NaN    CC BY-SA 3.0
2           NaN         NaN                 NaN    CC BY-SA 2.5

[3 rows x 23 columns]
```

Figure 3: Posts table - loaded Pandas dataframe

```
   Id  PostHistoryTypeId  PostId                              RevisionGUID  \
0   1                  2       1  8504513c-6719-455d-8ddb-cbb9a367df23
1   2                  1       1  8504513c-6719-455d-8ddb-cbb9a367df23
2   3                  3       1  8504513c-6719-455d-8ddb-cbb9a367df23

              CreationDate  UserId UserDisplayName Comment  \
0  2010-07-15T19:19:59.877     6.0             NaN     NaN
1  2010-07-15T19:19:59.877     6.0             NaN     NaN
2  2010-07-15T19:19:59.877     6.0             NaN     NaN

                                                 Text ContentLicense
0  I have a Canon 7D with a 50mm F1.4 and I think...   CC BY-SA 2.5
1      What is the best way to micro-adjust a lens?   CC BY-SA 2.5
2                                 <canon><7d><50mm>   CC BY-SA 2.5
```

Figure 4: PostsHistory table - loaded Pandas dataframe

```
   Id             CreationDate  PostId   RelatedPostId   LinkTypeId
0  21  2010-07-15T21:58:57.253      278             230           1
1  40  2010-07-16T03:57:55.867      269             258           1
2  89  2010-07-17T08:34:46.287      719             667           1
```

Figure 5: PostLink table - loaded Pandas dataframe

```
   Id         TagName  Count  ExcerptPostId  WikiPostId  IsModeratorOnly  \
0   2           canon   2610         2992.0      2938.0              NaN
1   5  depth-of-field    337         4472.0      4471.0              NaN
2   6       technique    318         7710.0      7709.0              NaN

   IsRequired
0         NaN
1         NaN
2         NaN
```

Figure 6: Tags table - loaded Pandas dataframe

- **Users**

```
Users = pd.read_csv('Users.csv', on_bad_lines='skip')
Users.head(3)
```

```
   Id  Reputation           CreationDate      DisplayName  \
0  -1           1  2010-07-15T17:44:52.020       Community
1   1         593  2010-07-15T18:15:20.407    Geoff Dalgas
2   2         101  2010-07-15T18:16:40.217  Kevin Montrose

            LastAccessDate                      WebsiteUrl  \
0  2010-07-15T17:44:52.020  http://meta.stackexchange.com/
1  2020-05-05T17:01:00.853        http://stackoverflow.com
2  2013-09-26T18:44:57.720       https://kevinmontrose.com

                     Location  \
0           on the server farm
1                Corvallis, OR
2  New York, NY, United States

                                              AboutMe  Views   UpVotes  \
0  <p>Hi, I'm not really a person.</p>\n\n<p>I'm ...     352     4834
1  <p>Dev #2 who helped create Stack Overflow cur...    2334       21
2  <p><a href="http://blog.stackoverflow.com/2010...      16       13

   DownVotes  ProfileImageUrl  EmailHash  AccountId
0       8097              NaN        NaN       -1.0
1          0              NaN        NaN        2.0
2          1              NaN        NaN    29738.0
```

Figure 7: Users table - loaded Pandas dataframe

- **Votes**

```
Votes = pd.read_csv('Votes.csv', comment="#")
Votes.head(2)
```

	Id	PostId	VoteTypeId	UserId	CreationDate	BountyAmount
0	3	10	2	NaN	2010-07-15T00:00:00.000	NaN
1	4	5	2	NaN	2010-07-15T00:00:00.000	NaN

Figure 8: Votes table - loaded Pandas dataframe

FINDINGS

Result 1 - Where are PHOTO users from?

The first result is about which country the Photography users are from. 'Location' in the *Users* table shows the location information of the users. However, when users fill in the information, their formats are not always aligned, some may fill in addresses with or without details, some may have abbreviations, and some may have typos. In order to know which country has the most users in Photography, several packages are introduced.

A **world map** will be applied to visualise the result, see Figure 9

```
import re
#extract the names and codes info
from country_named_entity_recognition import find_countries
country = pd.Series(Users['Location']).dropna().reset_index()

countries = country['Location'].apply(find_countries, is_ignore_case=True)

def name(x):
    if not x or not x[0]:
        return None
    return x[0][0].name

def code(x):
    if not x or not x[0]:
        return None
    return x[0][0].alpha_3

country['names'] = countries.apply(name)
country['codes'] = countries.apply(code)
count = country.groupby(['names','codes']).size().reset_index(name = 'count').sort_values('c

count.head(3)
```

	names	codes	count

```
150   United States   USA   4355
64            India   IND   3084
149  United Kingdom   GBR   2337
```

```
import geopandas as gpd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

# Load world map and merge with count
world = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))
world_count = world.merge(count, how='left', left_on='iso_a3', right_on='codes')

# initialize a new figure
fig, ax = plt.subplots(1, 1)

# plot a map of the countries
world_count.plot(column='count',
                 ax=ax,
                 legend=True,
                 legend_kwds={'label': 'Number of Site Users',
                              'orientation': "horizontal"}
)

# turn off axis ticks
ax.set_xticks([])
ax.set_yticks([])

plt.title('Number Of PHOTOGRAPHY Users In The World')
plt.show()
```

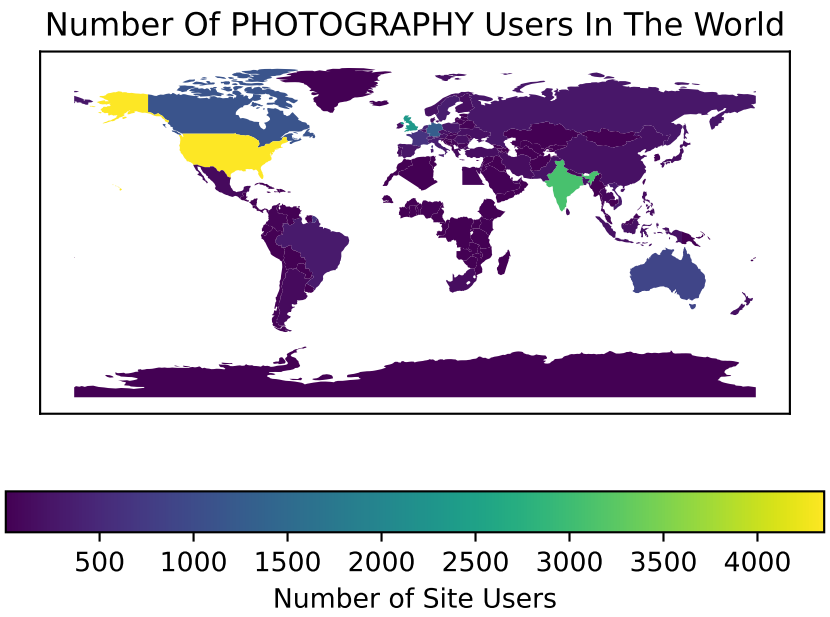

Figure 9: a world map showing where are the site users from

From the plot, we can see that USA has the most users in PHOTO, which is over 4,000. India has the second largest number of users, which is about 3,000. Overall, the PHOTO site has users globally.

Result 2 - What are the most frequent tags?

The second result is about the frequency of tags being used. It will start from extracting the tags information for each post in the *Posts* table.

A **word cloud** will be applied to visualise the result, see Figure 10

```
tags = pd.Series(Posts['Tags']).dropna()
tags = tags.str.findall(r'<(.*?)>')

tag = ' '.join([' '.join(tag_list) for tag_list in tags])
```

```
import os

from os import path
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

```
wordcloud = WordCloud(max_font_size=50).generate(tag)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

Figure 10: a word cloud plot on the frequency of tags being used

We can see that 'equipment' and 'recommendtion' take the most parts of the word cloud, which probably means that users are more interested in the photography equipment or seeking advice for a recommendation of relevant equipment. Furthermore, we can also see some popular brands, such as Canon, Nikon, and Sony.

Result 3 - Which brand is more frequently mentioned in the posts?

The third result compares the frequency of five popular camera brand names being mentioned in the posts. It will start from extracting the title texts for each post in the *Posts* table, and match the brand names, returning the frequency of the names. Besides the three brands found in **Result2(Figure 10)**, two more brands Fujifilm and Panasonic are added for comparison.

A **bar chart** will be applied to visualise the result, see Figure 11

```
post = pd.Series(Posts['Title'])

Sony = post.str.count(r"\bsony\b",re.IGNORECASE).sum()
Panasonic = post.str.count(r"\bpanasonic\b",re.IGNORECASE).sum()
Nikon = post.str.count(r"\bnikon\b",re.IGNORECASE).sum()
```

```
#in case there are typos or abbreviations
Canon = post.str.count(r"\bcan+on\b",re.IGNORECASE).sum()
Fujifilm = post.str.count(r"\bfujifilm|fuji\b",re.IGNORECASE).sum()

brand = ['Sony', 'Panasonic', 'Nikon', 'Canon', 'Fujifilm']
freq  = [Sony, Panasonic, Nikon, Canon, Fujifilm]
```

```
import matplotlib.pyplot as plt
import seaborn as sns

fig, ax = plt.subplots(figsize=(6, 5))
sns.barplot(x=brand, y=freq, ax=ax)

ax.bar_label(ax.containers[0], label_type='edge')

plt.xlabel("Brands")
plt.ylabel("Frequency in Titles")
plt.title("How Many Times The Camera Brands Being Mentioned In Posts Titles?")

plt.show()
```

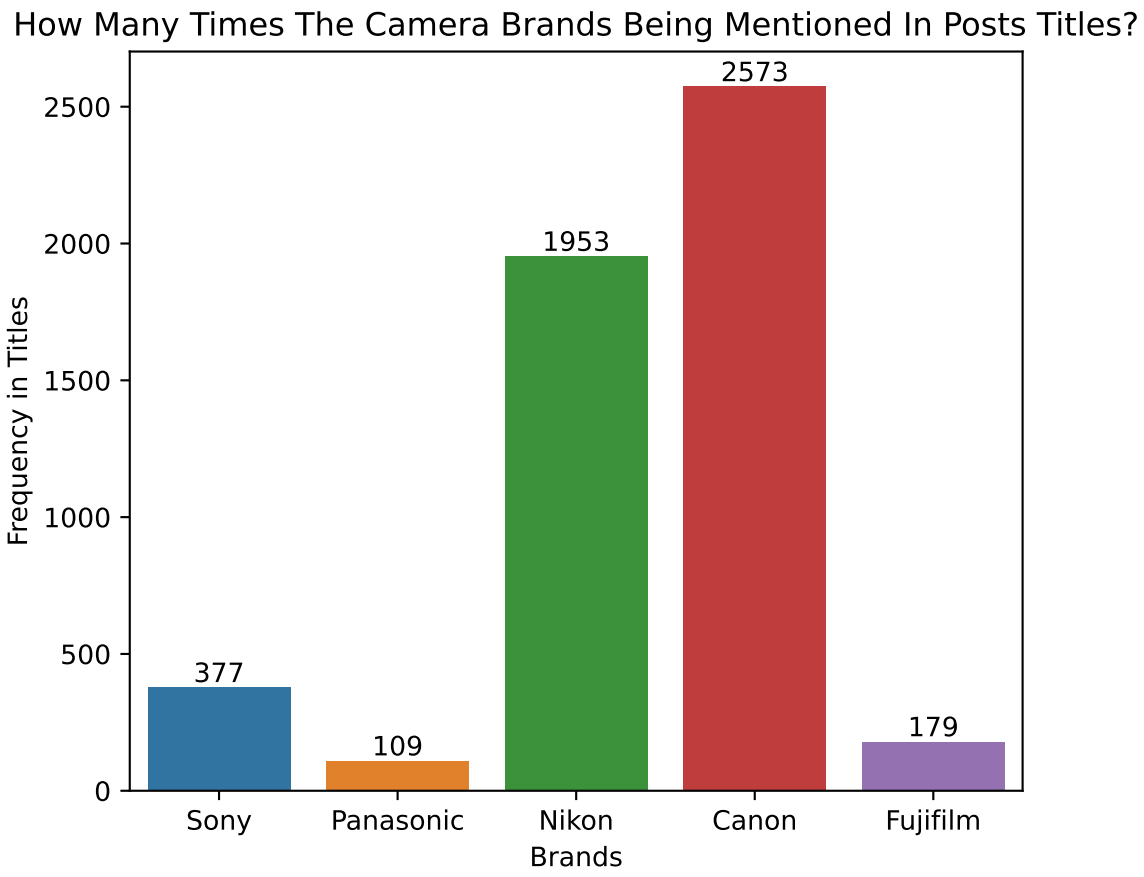

Figure 11: a bar chart showing the frequency of brand names being mentioned in the post titles

From the plot, we can see that Canon is the most frequently mentioned name among the five, whereas Nikon and Sony has the second and third frequency. This result is aligned with the word cloud plot in Result 2.

CONCLUSION

To sum up, the PHOTO site has users globally, and USA has the most users. Users are more interested in the photography equipment or seeking advice for a recommendation of relevant equipment, popular brands on the site are Canon and Nikon.

REFERENCE

- Stack Exchange, (2023) Stack Exchange, Stack Exchange Data Dump. URL: https://archive.org/details/stackexchange#reviews