In [261…
```
import pandas as pd
import ast
import collections
import operator
import numpy as np
```

In [262…
```
!ls -lh -S main_product/ | head -5
```

```
total 18064
-rwxr-xr-x@ 1 kushthaker  staff   694K 12 Nov 14:49 B06ZZ65855.csv
-rwxr-xr-x@ 1 kushthaker  staff   665K 12 Nov 11:16 B010ESCLHW.csv
-rwxr-xr-x@ 1 kushthaker  staff   663K 12 Nov 11:53 B00I14HLLS.csv
-rwxr-xr-x@ 1 kushthaker  staff   558K 12 Nov 15:05 B00OH5MIPO.csv
```

In [263…
```
df1 = pd.read_csv('main_product/B001NZO85O.csv')
df2 = pd.read_csv('main_product/B00OH5MIPO.csv')
```

In [264…
```
df1.columns == df2.columns
```

Out[264…
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True])
```

In [265…
```
!ls main_product/ > lights.txt
```

In [266…
```
with open('lights.txt','r') as l:
    csvs = [f.strip() for f in l.read().split('\n') if f != '']

df = pd.read_csv('main_product/' + csvs[0])

for csv in csvs[1:]:
    df = pd.concat([df, pd.read_csv('main_product/' + csv)])
```

In [267…
```
import sys; print(sys.getsizeof(df) * 1.e-6, 'MB')
```

```
18.43407 MB
```

list of helper functions for `df_products`

In [268…
```
def get_mass(dim_string):
    if dim_string == 0:
        return 0.00
    else:
        grams = np.array(dim_string.split(';')[1].split(' ')[1]).astype(np.float
        grams = np.prod(grams)
        return np.around(grams,2)

def get_volume(dim_string):
    if dim_string == 0:
        return 0.00
```

```
    # get density of flashlight in grams / m^3
    else:
        dims = [dim.strip() for dim in dim_string.split(';')[0].split('x')]
        dims[-1] = dims[-1][:-3]
        dims = np.array(dims).astype(np.float)
        volume_m = np.prod(dims) / 100 # convert metres
        return np.around(volume_m,2)

def convert_meta_to_dict(row):
    row['meta'] = ast.literal_eval(row.meta)
    return row

def add_mass(row):
    row['mass'] = get_mass(row.meta.get('product dimensions',0))
    return row

def add_volume(row):
    row['volume'] = get_volume(row.meta.get('product dimensions',0))
    return row

def get_price(row):
    row['price'] = df.loc[df.product_id == row.id, ['price']].values[0][0].astyp
    return row

def get_rating(row):
    row['rating'] = df.loc[df.product_id == row.id, ['average_rating']].values[0
    return row

def get_image_url(row):
    row['image_url'] = df.loc[df.product_id == row.id, ['product_image_url']].va
    return row
```

In [269…

```
df_products = pd.DataFrame({'id':df.product_id.unique(),
                            'product':df.product_name.unique(),
                            'meta':[meta for meta in df.meta_data.unique() if ty
                           })
```

In [270…

```
df_products = df_products.apply(get_price, axis=1)
df_products = df_products.apply(get_rating, axis=1)
df_products = df_products.apply(get_image_url, axis=1)
df_products = df_products.apply(convert_meta_to_dict, axis=1)
df_products = df_products.apply(add_mass, axis=1)
df_products = df_products.apply(add_volume, axis=1)
```

In [271…

```
df_products
```

Out[271…

| | id | product | meta | price | rating | image_url | mass | volu |
|---|---|---|---|---|---|---|---|---|
| **0** | B001NZO85O | Fenix Flashlight Headband (Fits Lights with 18... | {'product dimensions': '10 x 10 x 10 cm; 68.04... | 36.47 | 4.2 | https://images-na.ssl-images-amazon.com/images... | 68.04 | 1C |

| | id | product | meta | price | rating | image_url | mass | volu |
|---|---|---|---|---|---|---|---|---|
| **1** | B005CWRB44 | Fenix Compact 140 Lumen Flashlight | {'item weight': '0.81 Ounces', 'number of piec... | 81.71 | 4.0 | https://images-na.ssl-images-amazon.com/images... | 0.00 | C |
| **2** | B0062PVSGW | Fenix Pd32-R5 Cree Xp-G Led Flashlight | {'product dimensions': '2.4 x 12.7 x 2.4 cm; 6... | 129.65 | 4.5 | https://images-na.ssl-images-amazon.com/images... | 60.95 | C |
| **3** | B0091TRPVI | Fenix E25 Flashlight-187 Lumens | {'product dimensions': '2.4 x 14.6 x 2.4 cm; 7... | 118.45 | 3.9 | https://images-na.ssl-images-amazon.com/images... | 73.71 | C |
| **4** | B00937X7G0 | Nitecore MT2A CREE XP-G R5 LED 280 Lumen Multi... | {'manufacturer': 'Nitecore', 'part number': 'M... | 59.94 | 4.2 | https://images-na.ssl-images-amazon.com/images... | 66.90 | C |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **85** | B0841RSDCR | Nitecore E4K 4400 Lumen high powered Flashligh... | {'manufacturer': 'Nitecore Flashlights', 'parc... | 110.00 | 4.5 | https://images-na.ssl-images-amazon.com/images... | 0.00 | C |
| **86** | B086PW9TTP | ACEBEAM E10 LED Flashlight, 760 Lumens, Long T... | {'manufacturer': 'ZENBON', 'part number': 'E10... | 73.55 | 4.5 | https://images-na.ssl-images-amazon.com/images... | 148.00 | 1 |
| **87** | B087CG1YW6 | Fenix PD40R v2 3000 Lumen Mechanical Rotary Sw... | {'manufacturer': 'Fenix Flashlights', 'part nu... | NaN | 4.5 | https://images-na.ssl-images-amazon.com/images... | 117.08 | |
| **88** | B08BTQ2T4C | Fenix E03R 260 Lumen Rechargeable EDC Keychain... | {'manufacturer': 'Fenix Flashlights', 'part nu... | NaN | 4.9 | https://images-na.ssl-images-amazon.com/images... | 22.11 | C |
| **89** | B08DCSF6ZX | ACEBEAM L17 1400 Lumens 802m Long Range Throw ... | {'manufacturer': 'Zenbon', 'part number': 'L17... | 90.00 | 4.7 | https://images-na.ssl-images-amazon.com/images... | 272.00 | 1 |

90 rows × 8 columns

next steps get manual df

In [293…

```
df_manual = pd.read_csv('j2-urls.csv')
df_manual = df_manual[['Size','Max Throw (yards)','Lumen','Brand','Product','Lin
df_manual = df_manual.loc[df_manual['Link'].isna() == False]
```

In [294…
```
import re

pattern = re.compile(r'/([a-zA-Z0-9]{10})(?:[/?]|$)')

def get_asin(url):
    asin = ([m.group(0) for m in pattern.finditer(url)][0]).replace('/','')
    return asin

def add_asin(row):
    row['id'] = get_asin(row.Link)
    return row
```

In [295…
```
df_manual = df_manual.apply(add_asin, axis=1)
print('len df_manual: ', len(df_manual))
print('len df_products: ', len(df_products))
```

```
len df_manual:  83
len df_products:  90
```

In [296…
```
df_final = pd.merge(df_products, df_manual, how='left', on='id')
print('len merged: ', len(df_final))
df_final = df_final.dropna()
print('len w dropna(): ', len(df_final))
```

```
len merged:  92
len w dropna():  63
```

In [297…
```
df_final = df_final.rename(columns={'Size':'size',
                                    'Lumen':'lumens',
                                    'Max Throw (yards)':'throw_yards',
                                    'Brand':'brand',
                                    'Product':'model',
                                    'Link':'page_url'})
df_final.columns
```

Out[297…
```
Index(['id', 'product', 'meta', 'price', 'rating', 'image_url', 'mass',
       'volume', 'size', 'throw_yards', 'lumens', 'brand', 'model',
       'page_url'],
      dtype='object')
```

In [298…
```
def get_lumens(row):
#     if '?' in row.lumens:
#         row.lumens = row.lumens.replace('?','')
    row.lumens = np.array(row.lumens).astype(np.float)
    return row

def get_throw_yards(row):
#     if '?' in row.throw_yards:
#         row.throw_yards = row.throw_yards.replace('?','')
    row.throw_yards = np.array(row.throw_yards).astype(np.float)
    return row

def get_label(row):
    row['label'] = row['brand'] + ' ' + row['model']
    return row
```

In [299…

```
df_final = df_final.apply(get_lumens, axis=1)
df_final = df_final.apply(get_throw_yards, axis=1)
df_final = df_final.apply(get_label, axis=1)
```

In [300…

```
df_final = df_final[['id',
                     'brand',
                     'model',
                     'label',
                     'product',
                     'meta',
                     'price',
                     'lumens',
                     'throw_yards',
                     'rating',
                     'size',
                     'mass',
                     'volume',
                     'image_url',
                     'page_url'
                    ]]
```

In [301…

```
df_final.at
```

Out[301…

| | id | brand | model | label | product | meta | price | lumens | throw_y |
|---|---|---|---|---|---|---|---|---|---|
| **1** | B005CWRB44 | Fenix | E15 | Fenix E15 | Fenix Compact 140 Lumen Flashlight | {'item weight': '0.81 Ounces', 'number of piec... | 81.71 | 170.0 | |
| **2** | B0062PVSGW | Fenix | Pd32-R5 | Fenix Pd32-R5 | Fenix Pd32-R5 Cree Xp-G Led Flashlight | {'product dimensions': '2.4 x 12.7 x 2.4 cm; 6... | 129.65 | 315.0 | |
| **3** | B0091TRPVI | Fenix | E25 | Fenix E25 | Fenix E25 Flashlight-187 Lumens | {'product dimensions': '2.4 x 14.6 x 2.4 cm; 7... | 118.45 | 187.0 | |
| **4** | B00937X7G0 | Nitecore | MT1A | Nitecore MT1A | Nitecore MT2A CREE XP-G R5 LED 280 Lumen Multi... | {'manufacturer': 'Nitecore', 'part number': 'M... | 59.94 | 345.0 | |
| **5** | B00937YE6C | Nitecore | MT1C | Nitecore MT1C | NiteCore CREE XP-G R5 MT1C Multitask LED Flash... | {'product dimensions': '8.79 x 2.34 x 2.34 cm;... | 53.99 | 280.0 | |

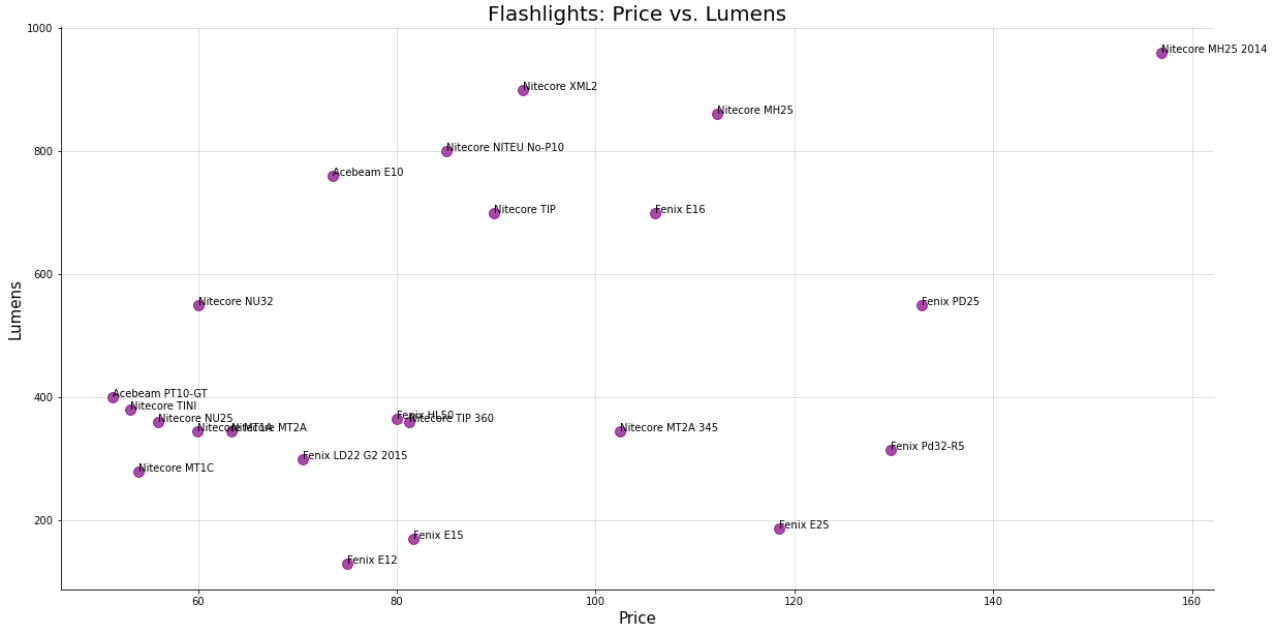
| | id | brand | model | label | product | meta | price | lumens | throw_y |
|---|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **81** | B07YXRYCH7 | Acebeam | PT10-GT | Acebeam PT10-GT | ACEBEAM PT10-GT Pen Light Flashlight, SAMSUNG ... | {'manufacturer': 'ZENBON', 'part number': 'PT1... | 51.46 | 400.0 | |
| **86** | B0841RSDCR | Nitecore | E4K | Nitecore E4K | Nitecore E4K 4400 Lumen high powered Flashligh... | {'manufacturer': 'Nitecore Flashlights', 'parc... | 110.00 | 4400.0 | 1 |
| **87** | B0841RSDCR | Nitecore | E4K | Nitecore E4K | Nitecore E4K 4400 Lumen high powered Flashligh... | {'manufacturer': 'Nitecore Flashlights', 'parc... | 110.00 | 4400.0 | 1 |
| **88** | B086PW9TTP | Acebeam | E10 | Acebeam E10 | ACEBEAM E10 LED Flashlight, 760 Lumens, Long T... | {'manufacturer': 'ZENBON', 'part number': 'E10... | 73.55 | 760.0 | 6 |
| **91** | B08DCSF6ZX | Acebeam | L17 | Acebeam L17 | ACEBEAM L17 1400 Lumens 802m Long Range Throw ... | {'manufacturer': 'Zenbon', 'part number': 'L17... | 90.00 | 1400.0 | |

63 rows × 15 columns

In [335…

```
import matplotlib.pyplot as plt
import matplotlib.cm as cm

df_final = df_final.loc[((df_final.price > 50) & (df_final.price < 175) & (df_fi
ax = df_final.plot(kind='scatter',x='price',y='lumens', s=100, c='purple', alpha
ax.set_title('Flashlights: Price vs. Lumens', fontsize=20)
ax.set_xlabel('Price',fontsize=15)
ax.set_ylabel('Lumens',fontsize=15)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.grid(color='grey', linestyle='-', linewidth=0.5, alpha=0.5)
df_final[['price','lumens','label']].apply(lambda row: ax.text(*row),axis=1);
```




In [352…

```
sent_df = pd.read_csv('some_sentiment.csv')
sent_df = sent_df.drop('Unnamed: 0', axis=1)
sent_df = sent_df.rename(columns={'product':'id',
                                  'size':'size_sentiment',
                                  'quality':'quality_sentiment',
                                  'battery':'battery_sentiment',
                                  'design':'design_sentiment',
                                  'beam':'beam_sentiment',
                                  'price':'price'})
sent_df
```

Out[352…

| | id | size | quality | battery | design | beam | price |
|---|---|---|---|---|---|---|---|
| **0** | B001NZO85O | 0.162880 | 0.256529 | 0.146292 | 0.210179 | 0.215273 | 0.243250 |
| **1** | B005CWRB44 | 0.276473 | 0.228552 | 0.178218 | 0.166431 | 0.295173 | 0.277621 |
| **2** | B0062PVSGW | 0.271035 | 0.246449 | 0.175576 | 0.208615 | 0.275401 | 0.241623 |
| **3** | B0091TRPVI | 0.288658 | 0.290800 | 0.194815 | 0.181698 | 0.259889 | 0.295250 |
| **4** | B00937X7G0 | 0.334429 | 0.264286 | 0.202900 | 0.226259 | 0.326632 | 0.373400 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **85** | B0841RSDCR | 0.238538 | 0.236000 | 0.167667 | 0.136273 | 0.297864 | 0.142000 |
| **86** | B086PW9TTP | 0.368500 | 0.105500 | 0.229500 | 0.179000 | 0.252500 | 0.104000 |
| **87** | B087CG1YW6 | 0.359333 | 0.272818 | 0.210200 | 0.184333 | 0.373156 | 0.217778 |
| **88** | B08BTQ2T4C | 0.304944 | 0.193600 | 0.227417 | 0.203429 | 0.289783 | 0.204667 |
| **89** | B08DCSF6ZX | 0.175200 | 0.217000 | 0.201778 | 0.233700 | 0.252063 | 0.178750 |

90 rows × 7 columns

In [350…

```
df_final = pd.merge(df_final, sent_df, how='left', on='id')
```

In [351…

```
df_final
```

Out[351…

| | id | brand | model | label | product | meta | price_x | lumens | thr |
|---|---|---|---|---|---|---|---|---|---|
| **0** | B005CWRB44 | Fenix | E15 | Fenix E15 | Fenix Compact 140 Lumen Flashlight | {'item weight': '0.81 Ounces', 'number of piec... | 81.71 | 170.0 | |
| **1** | B0062PVSGW | Fenix | Pd32-R5 | Fenix Pd32-R5 | Fenix Pd32-R5 Cree Xp-G Led Flashlight | {'product dimensions': '2.4 x 12.7 x 2.4 cm; 6... | 129.65 | 315.0 | |
| **2** | B0091TRPVI | Fenix | E25 | Fenix E25 | Fenix E25 Flashlight-187 Lumens | {'product dimensions': '2.4 x 14.6 x 2.4 cm; 7... | 118.45 | 187.0 | |
| **3** | B00937X7G0 | Nitecore | MT1A | Nitecore MT1A | Nitecore MT2A CREE XP-G R5 LED 280 Lumen Multi... | {'manufacturer': 'Nitecore', 'part number': 'M... | 59.94 | 345.0 | |
| **4** | B00937YE6C | Nitecore | MT1C | Nitecore MT1C | NiteCore CREE XP-G R5 MT1C Multitask LED Flash... | {'product dimensions': '8.79 x 2.34 x 2.34 cm;... | 53.99 | 280.0 | |
| **5** | B0093804JM | Nitecore | MT2A | Nitecore MT2A | NiteCore MT1A Multitask CREE XP-G R5 LED Flash... | {'product dimensions': '10.49 x 2.34 x 2.34 cm... | 63.38 | 345.0 | |
| **6** | B00GZA1V02 | Nitecore | MH25 | Nitecore MH25 | NiteCore MH25 CREE XM-L U2 LED 960 Lumens USB ... | {'product dimensions': '16 x 3.99 x 3.99 cm; 1... | 112.24 | 860.0 | |
| **7** | B00I14HLLS | Fenix | E12 | Fenix E12 | Fenix E12 Flashlight 130 Lumens | {'product dimensions': '8.89 x 1.27 x 1.27 cm;... | 75.01 | 130.0 | |
| **8** | B00KVR27QM | Fenix | HL50 | Fenix HL50 | Fenix Flashlights HL50 365 Lumens Headlamp, Black | {'product dimensions': '6.1 x 5.08 x 5.08 cm; ... | 79.98 | 365.0 | |
| **9** | B00M94WSVY | Nitecore | NITEU No-P10 | Nitecore NITEU No-P10 | Nitecore NITEU No-P10 Cree XM-L2 T6 800-Lumen ... | {'product dimensions': '13.41 x 2.54 x 2.54 cm... | 84.99 | 800.0 | |

| | id | brand | model | label | product | meta | price_x | lumens | thr |
|---|---|---|---|---|---|---|---|---|---|
| **10** | B00NLL7PDA | Nitecore | MH25 2014 | Nitecore MH25 2014 | Nitecore MH25 2014 Edition 960 Lumens 340 Yard... | {'manufacturer': 'NITECORE', 'part number': 'M... | 156.90 | 960.0 | |
| **11** | B00NQNZWIS | Nitecore | MT2A 345 | Nitecore MT2A 345 | Nitecore MT2A 345 Lumens LED Flashlight w/Bonu... | {'manufacturer': 'NiteCore', 'part number': 'M... | 102.41 | 345.0 | |
| **12** | B00VVV7W9Y | Nitecore | XML2 | Nitecore XML2 | NiteCore CREE XML2 900 lm LED Flashlight Secon... | {'product dimensions': '7.49 x 2.54 x 2.54 cm;... | 92.70 | 900.0 | |
| **13** | B00Y18AF6Q | Fenix | PD25 | Fenix PD25 | Fenix PD25 550 Lumens CREE XP-L LED Pocket Fla... | {'manufacturer': 'Fenix', 'part number': 'PD25... | 132.81 | 550.0 | |
| **14** | B01418RAZY | Fenix | LD22 G2 2015 | Fenix LD22 G2 2015 | Fenix LD22 G2 2015 EDT 300 Lumens LED Flashlight | {'product dimensions': '15.5 x 2.15 x 2.15 cm;... | 70.56 | 300.0 | |
| **15** | B06XXLK75C | Nitecore | TIP 360 | Nitecore TIP 360 | Nitecore TIP 360 Lumen USB Rechargeable Keycha... | {'manufacturer': 'Nitecore', 'part number': 'T... | 81.24 | 360.0 | |
| **16** | B076QJMK4X | Nitecore | TINI | Nitecore TINI | Nitecore TINI 380 Lumens USB Rechargeable Keyc... | {'product dimensions': '4.32 x 2.54 x 1.14 cm;... | 53.19 | 380.0 | |
| **17** | B078PH9DCB | Nitecore | NU25 | Nitecore NU25 | NITECORE NU25 360 LM Rechargeable Headlamp | {'product dimensions': '5.56 x 3.45 x 0.48 cm;... | 56.02 | 360.0 | |
| **18** | B07DQMXLCX | Nitecore | TIP | Nitecore TIP | Nitecore Tip SE Black 700 Lumen USB-C Recharge... | {'manufacturer': 'Nitecore', 'part number': 'T... | 89.74 | 700.0 | |
| **19** | B07HM6GM36 | Fenix | E16 | Fenix E16 | Fenix E16 700 Lumen High Performance Keychain ... | {'manufacturer': 'FENIX', 'part number': 'E16+... | 105.95 | 700.0 | |
| **20** | B07K1XQ8M2 | Nitecore | NU32 | Nitecore NU32 | NITECORE NU32 550 Lumen LED Rechargeable Headl... | {'product dimensions': '6.3 x 4.34 x 4.34 cm; ... | 59.99 | 550.0 | |

| | id | brand | model | label | product | meta | price_x | lumens | thr |
|---|---|---|---|---|---|---|---|---|---|
| **21** | B07YXRYCH7 | Acebeam | PT10-GT | Acebeam PT10-GT | ACEBEAM PT10-GT Pen Light Flashlight, SAMSUNG ... | {'manufacturer': 'ZENBON', 'part number': 'PT1... | 51.46 | 400.0 | |
| **22** | B086PW9TTP | Acebeam | E10 | Acebeam E10 | ACEBEAM E10 LED Flashlight, 760 Lumens, Long T... | {'manufacturer': 'ZENBON', 'part number': 'E10... | 73.55 | 760.0 | |

23 rows × 21 columns

In [ ]:

## Get spreadsheet names

In [22]:
```
!ls main_product/ > lights.txt
```

## Import Libraries

In [45]:
```
import nltk
nltk.download('vader_lexicon')
nltk.download('punkt')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/kushthaker/nltk_data...
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/kushthaker/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[45]: True

In [46]:
```
import pandas as pd
import numpy as np
import time


from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()

from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
```

## Requirement file

In [31]:
```
# !pip install -r requirements.txt
# !pip freeze > requirements.txt
```

In [32]:
```
with open('lights.txt','r') as l:
    csvs = [f.strip() for f in l.read().split('\n') if f != '']

df = pd.read_csv('main_product/' + csvs[0])

for csv in csvs[1:]:
    df = pd.concat([df, pd.read_csv('main_product/' + csv)])
```

## Read a spreadsheet

In [33]:
```
df.columns
```

Out[33]:
```
Index(['Unnamed: 0', 'author_id', 'verified_purchase', 'review_title',
       'reviews', 'country', 'date', 'reviewer_name', 'ratings',
       'people_find_helpful', 'start_time', 'end_time', 'product_name',
       'average_rating', 'price', 'total_reviews', 'product_category',
```

```
       'product_id', 'meta_data', 'rank', 'product_image_url'],
      dtype='object')
```

In [36]:

```
# Keep column reviews only
df = df[['product_id','reviews']].dropna()
print(f"New shape: {df.shape}")
df.head(3)
```

```
New shape: (11434, 2)
```

Out[36]:

| | product_id | reviews |
|---|---|---|
| 0 | B001NZO85O | Happy with the quality & make. Surpassed my ex... |
| 1 | B001NZO85O | Used it with 2 Fenix Flashlights E12 130-Lumen... |
| 2 | B001NZO85O | You won't be disappointed. This thing is aweso... |

## Partition reviews into sentences

In [ ]:

```
reviews_corpus = list(df['reviews'])
# Partition into sentences
reviews_in_sentences = [sent_tokenize(review.lower()) for review in reviews_corp
reviews_length = [len(review) for review in reviews_in_sentences]
print(f"Number of reviews: {len(reviews_in_sentences)}")
print(f"Number of sentences in each review: {reviews_length}")
```

# !!! Need extra work on this!!!

## Use Fliashlight corpus for matching sentences into aspects

https://docs.google.com/document/d/1ZiQVK4czqH0UGWZM1XaEhribQUT0xkI71Xh_aCGtw1E/edit

In [38]:

```
size_keywords_string = "Size, small, tiny, petite, slim, compact, large, big, gi
huge, enormous, gigantic, bulky, colossal, massive, sizable, weight, heavy, ligh

quality_keywords_string = "Build, built, quality,durability, sturdy, sturdiness,
coating, solid, cheaply, aluminum, steel, titanium, brass, copper,\
material, metal, rubber, plastic, nylon, bent, broke, faulty, shatter,\
waterproof, dustproof, corrosion, ingress, drop, shock, impact, resistance,\
screws, threads, knurling, anodized, flicker, housing, indestructible,\
wet, temperature, hot, heat, overheat, cold,  well"

battery_keywords_string = "Battery, batteries, rechargeable, charge, charging,\
recharge, USB, solar, runtime, hours, lifetime, dies, died, dead"

design_keywords_string = "Features, design, setting, settings, mode, modes,\
interface, programmable, memory, dim, roll, design, roll, upright, stand, tailst
strobe, sos, float, warranty, grip, rotate, rotating, head, hang, lantern, eco,
zoom, clip, lanyard, holster, indicator, easy to use, versatile, switch, twist,\
activation, clicky, click, magnetize, accessories, bezel"

beam_keywords_string = "Power, project, projects, far, illuminate, shine,\
focus, distance, range, feet, meters, beam, distance, visibility, throw, \
flood, lumens, bright, brightness, lens, optics, frosted, reflector, mule, LED,
colour, color, hotspot, spill, corona, lux, candelas, intensity, lights"
```

```
price_keywords_string = "Price, cost, costly, pricey, pricy, expensive, overpric
unreasonable, value, affordable, cheap, $, bargain, budget, cash, discount, mone


size_keywords = size_keywords_string.lower().replace(" ","").split(",")
quality_keywords = quality_keywords_string.lower().replace(" ","").split(",")
battery_keywords = battery_keywords_string.lower().replace(" ","").split(",")
design_keywords = design_keywords_string.lower().replace(" ","").split(",")
beam_keywords = beam_keywords_string.lower().replace(" ","").split(",")
price_keywords = price_keywords_string.lower().replace(" ","").split(",")
```

## Helper functions

In [39]:

```
# checkPresence takes in:
# a sentence: represented by a string
# keywords: a list of keywords
# returns True if the sentence contains any of the keywords
def checkPresence(sentence, keywords):
    for keyword in keywords:
        if keyword in word_tokenize(sentence):
            return True
    return False

# checkPresence takes in:
# reviews_in_sentences: a list of list of sentences
#  (A review is represented by a list of sentences)
# keywords: a list of keywords
# returns: a list of filtered review which contains the keywords.
#         an empty string for a review that contains no keyword.
def filteredReview(reviews_in_sentences, keywords):
    ret = []
    for sentences in reviews_in_sentences:
        filtered = ''
        for sentence in sentences:
            if checkPresence(sentence, keywords):
                filtered += sentence
        ret.append(filtered)
    return ret
```

## Match sentences into aspects

Took 4 seconds to process 80 reviews. (Need 250 seconds for 5000 reviews)

In [40]:

```
start = time.time()
df['size'] = filteredReview(reviews_in_sentences, size_keywords)
df['quality'] = filteredReview(reviews_in_sentences, quality_keywords)
df['battery'] = filteredReview(reviews_in_sentences, battery_keywords)
df['design'] = filteredReview(reviews_in_sentences, design_keywords)
df['beam'] = filteredReview(reviews_in_sentences, beam_keywords)
df['price'] = filteredReview(reviews_in_sentences, price_keywords)
end = time.time()
print(f"Took {end - start} seconds to match sentences into aspects.")
```

```
Took 962.9222347736359 seconds to match sentences into aspects.
```

In [41]:
```
df.head()
```

Out[41]:

| | product_id | reviews | size | quality | battery | design | beam | price |
|---|---|---|---|---|---|---|---|---|
| **0** | B001NZO85O | Happy with the quality & make. Surpassed my ex... | | happy with the quality & make.both fit well an... | | both fit well and even accommodated the clip o... | excellent adjustability if you need a close-up... | |
| **1** | B001NZO85O | Used it with 2 Fenix Flashlights E12 130-Lumen... | | | | | used it with 2 fenix flashlights e12 130-lumen... | |
| **2** | B001NZO85O | You won't be disappointed. This thing is aweso... | | | | | | |
| **3** | B001NZO85O | Works well with my Fenix LD22 flashlight as de... | | works well with my fenix ld22 flashlight as de... | | | | |
| **4** | B001NZO85O | old one was worn and stretched very good | | | | | | |

In [78]:
```
df.to_csv('sentence_partition.csv')
```

---

# Sentiment Part

In [79]:
```
df_sent = pd.read_csv('sentence_partition.csv', encoding='utf8')
```

In [80]:
```
def find_sentiment(sentence):
    return sid.polarity_scores(sentence)

df.iloc[:,3:] = df.iloc[:,3:].applymap(find_sentiment)
```

In [83]:
```
df.to_csv('sentence_sentiment.csv')
```

In [136…
```
def get_size_scores(row):
    if type(row['size']) == dict:
        if row['size'].get('neg',0) and row['size'].get('pos',0):
            row['size'] = np.max((row['size'].get('neg',0), row['size'].get('pos
        elif row['size'].get('pos',0):
```

```
            row['size'] = row['size'].get('pos',0)
        elif row['size'].get('neg',0):
            row['size'] = row['size'].get('neg',0)
        else:
            row['size'] = 0

        return row
    return row
```

In [137…

```
def get_quality_scores(row):
    if type(row['quality']) == dict:
        if row['quality'].get('neg',0) and row['quality'].get('pos',0):
            row['quality'] = np.max((row['quality'].get('neg',0), row['quality']
        elif row['quality'].get('pos',0):
            row['quality'] = row['quality'].get('pos',0)
        elif row['quality'].get('neg',0):
            row['quality'] = row['quality'].get('neg',0)
        else:
            row['quality'] = 0

        return row
    return row
```

In [138…

```
def get_battery_scores(row):
    if type(row['battery']) == dict:
        if row['battery'].get('neg',0) and row['battery'].get('pos',0):
            row['battery'] = np.max((row['battery'].get('neg',0), row['battery']
        elif row['battery'].get('pos',0):
            row['battery'] = row['battery'].get('pos',0)
        elif row['battery'].get('neg',0):
            row['battery'] = row['battery'].get('neg',0)
        else:
            row['battery'] = 0

        return row
    return row
```

In [139…

```
def get_design_scores(row):
    if type(row['design']) == dict:
        if row['design'].get('neg',0) and row['design'].get('pos',0):
            row['design'] = np.max((row['design'].get('neg',0), row['design'].ge
        elif row['design'].get('pos',0):
            row['design'] = row['design'].get('pos',0)
        elif row['design'].get('neg',0):
            row['design'] = row['design'].get('neg',0)
        else:
            row['design'] = 0

        return row
    return row
```

In [140…

```
def get_beam_scores(row):
```

```
    if type(row['beam']) == dict:
        if row['beam'].get('neg',0) and row['beam'].get('pos',0):
            row['beam'] = np.max((row['beam'].get('neg',0), row['beam'].get('pos
        elif row['beam'].get('pos',0):
            row['beam'] = row['beam'].get('pos',0)
        elif row['beam'].get('neg',0):
            row['beam'] = row['beam'].get('neg',0)
        else:
            row['beam'] = 0

        return row
    return row
```

In [141…

```
def get_price_scores(row):
    if type(row['price']) == dict:
        if row['price'].get('neg',0) and row['price'].get('pos',0):
            row['price'] = np.max((row['price'].get('neg',0), row['price'].get('
        elif row['price'].get('pos',0):
            row['price'] = row['price'].get('pos',0)
        elif row['price'].get('neg',0):
            row['price'] = row['price'].get('neg',0)
        else:
            row['price'] = 0

        return row
    return row
```

In [146…

```
df = df.apply(get_size_scores,axis=1)
df = df.apply(get_quality_scores,axis=1)
df = df.apply(get_battery_scores,axis=1)
df = df.apply(get_design_scores,axis=1)
df = df.apply(get_beam_scores,axis=1)
df = df.apply(get_price_scores,axis=1)
```

In [147…

```
df.head()
```

Out[147…

| | product_id | reviews | size | quality | battery | design | beam | price |
|---|---|---|---|---|---|---|---|---|
| **0** | B001NZO85O | Happy with the quality & make. Surpassed my ex... | 0.0 | 0.409 | 0.0 | 0.338 | 0.316 | 0.0 |
| **1** | B001NZO85O | Used it with 2 Fenix Flashlights E12 130-Lumen... | 0.0 | 0.000 | 0.0 | 0.000 | 0.000 | 0.0 |
| **2** | B001NZO85O | You won't be disappointed. This thing is aweso... | 0.0 | 0.000 | 0.0 | 0.000 | 0.000 | 0.0 |
| **3** | B001NZO85O | Works well with my Fenix LD22 flashlight as de... | 0.0 | 0.208 | 0.0 | 0.000 | 0.000 | 0.0 |
| **4** | B001NZO85O | old one was worn and stretched very good | 0.0 | 0.000 | 0.0 | 0.000 | 0.000 | 0.0 |

In [149…

```
products = df.product_id.unique()
```

```
sent_df_final = pd.DataFrame({'product':products,
                              'size':np.zeros(len(products)),
                              'quality':np.zeros(len(products)),
                              'battery':np.zeros(len(products)),
                              'design':np.zeros(len(products)),
                              'beam':np.zeros(len(products)),
                              'price':np.zeros(len(products))
                             })
```

Out[149…

| | product | size | quality | battery | design | beam | price |
|---|---|---|---|---|---|---|---|
| **0** | B001NZO85O | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | B005CWRB44 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | B0062PVSGW | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | B0091TRPVI | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | B00937X7G0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **85** | B0841RSDCR | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **86** | B086PW9TTP | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **87** | B087CG1YW6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **88** | B08BTQ2T4C | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **89** | B08DCSF6ZX | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

90 rows × 7 columns

In [195…

```
# sent_df_final.loc[sent_df_final['product'] == 'B08DCSF6ZX']['size']
```

Out[195…

| | product | size | quality | battery | design | beam | price |
|---|---|---|---|---|---|---|---|
| **0** | B001NZO85O | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | B005CWRB44 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | B0062PVSGW | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | B0091TRPVI | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | B00937X7G0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **85** | B0841RSDCR | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **86** | B086PW9TTP | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **87** | B087CG1YW6 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **88** | B08BTQ2T4C | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **89** | B08DCSF6ZX | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

90 rows × 7 columns

In [198…

```
size_df = df.loc[df['size'] > 0]
quality_df = df.loc[df['quality'] > 0]
battery_df = df.loc[df['battery'] > 0]
design_df = df.loc[df['design'] > 0]
beam_df = df.loc[df['beam'] > 0]
price_df = df.loc[df['price'] > 0]

for i,product in enumerate(products):
    sent_df_final.loc[i,'size'] = np.mean(size_df.loc[size_df.product_id == prod
    sent_df_final.loc[i,'quality'] = np.mean(quality_df.loc[quality_df.product_i
    sent_df_final.loc[i,'battery'] = np.mean(battery_df.loc[battery_df.product_i
    sent_df_final.loc[i,'design'] = np.mean(design_df.loc[design_df.product_id =
    sent_df_final.loc[i,'beam'] = np.mean(beam_df.loc[beam_df.product_id == prod
    sent_df_final.loc[i,'price'] = np.mean(price_df.loc[price_df.product_id == p
```

In [199…

```
sent_df_final
```

Out[199…

| | product | size | quality | battery | design | beam | price |
|---|---|---|---|---|---|---|---|
| **0** | B001NZO85O | 0.162880 | 0.256529 | 0.146292 | 0.210179 | 0.215273 | 0.243250 |
| **1** | B005CWRB44 | 0.276473 | 0.228552 | 0.178218 | 0.166431 | 0.295173 | 0.277621 |
| **2** | B0062PVSGW | 0.271035 | 0.246449 | 0.175576 | 0.208615 | 0.275401 | 0.241623 |
| **3** | B0091TRPVI | 0.288658 | 0.290800 | 0.194815 | 0.181698 | 0.259889 | 0.295250 |
| **4** | B00937X7G0 | 0.334429 | 0.264286 | 0.202900 | 0.226259 | 0.326632 | 0.373400 |
| **…** | … | … | … | … | … | … | … |
| **85** | B0841RSDCR | 0.238538 | 0.236000 | 0.167667 | 0.136273 | 0.297864 | 0.142000 |
| **86** | B086PW9TTP | 0.368500 | 0.105500 | 0.229500 | 0.179000 | 0.252500 | 0.104000 |
| **87** | B087CG1YW6 | 0.359333 | 0.272818 | 0.210200 | 0.184333 | 0.373156 | 0.217778 |
| **88** | B08BTQ2T4C | 0.304944 | 0.193600 | 0.227417 | 0.203429 | 0.289783 | 0.204667 |
| **89** | B08DCSF6ZX | 0.175200 | 0.217000 | 0.201778 | 0.233700 | 0.252063 | 0.178750 |

90 rows × 7 columns

In [200…

```
sent_df_final.to_csv('some_sentiment.csv')
```

In [ ]: