1. Read the .csv file using Pandas. Take a look at the top few records.

In [2]:
```
for each in os.listdir():
    print(each)
```

```
.ipynb_checkpoints
1580822492_1570782847_proj1
1580822492_1570782847_proj1.zip
Amazon Reviews LDA Topic Modelling Project-20231124T194448Z-001
Amazon Reviews LDA Topic Modelling Project-20231124T194448Z-001.zip
K8 Reviews v0.2.csv
Untitled.ipynb
```

In [6]:
```
df = pd.read_csv("K8 Reviews v0.2.csv")
df.head()
```

Out[6]:

	sentiment	review
0	1	Good but need updates and improvements
1	0	Worst mobile i have bought ever, Battery is dr...
2	1	when I will get my 10% cash back.... its alrea...
3	1	Good
4	0	The worst phone everThey have changed the last...

2. Normalize casings for the review text and extract the text into a list for easier manipulation.

```
df_lower = [sent.lower() for sent in df.review.values]
df_lower[0]
```

```
'good but need updates and improvements'
```

3. Tokenize the reviews using NLTKs word_tokenize function.

```
df_token = [word_tokenize(sent) for sent in df_lower]
df_token[0]
```

```
['good', 'but', 'need', 'updates', 'and', 'improvements']
```

4. Perform parts-of-speech tagging on each sentence using the NLTK POS tagger.

```
nltk.pos_tag(df_token[0])
```

```
[('good', 'JJ'),
 ('but', 'CC'),
 ('need', 'VBP'),
 ('updates', 'NNS'),
 ('and', 'CC'),
 ('improvements', 'NNS')]
```

```
df_tagged = [nltk.pos_tag(tokens) for tokens in df_token]
df_tagged[0]
```

```
[('good', 'JJ'),
 ('but', 'CC'),
 ('need', 'VBP'),
 ('updates', 'NNS'),
 ('and', 'CC'),
 ('improvements', 'NNS')]
```

5. For the topic model, we should want to include only nouns.
    a. Find out all the POS tags that correspond to nouns.
    b. Limit the data to only terms with these tags.

```
nltk.help.upenn_tagset()
```

```
...
```

```
df_noun=[]
for sent in df_tagged:
    df_noun.append([token for token in sent if re.search("NN.*", token[1])])
df_noun[0]
```

```
[('updates', 'NNS'), ('improvements', 'NNS')]
```

6. Lemmatize.
    a. Different forms of the terms need to be treated as one.
    b. No need to provide POS tag to lemmatizer for now.

```
In [34]: lemm = WordNetLemmatizer()
         df_lemm=[]
         for sent in df_noun:
             df_lemm.append([lemm.lemmatize(word[0]) for word in sent])
```

```
In [35]: df_lemm[0]
Out[35]: ['update', 'improvement']
```

7. Remove stopwords and punctuation (if there are any).

```
In [45]: from string import punctuation
         from nltk.corpus import stopwords

         stop_nltk = stopwords.words("english")

         [nltk_data] Downloading package stopwords to
         [nltk_data]     C:\Users\david\AppData\Roaming\nltk_data...
         [nltk_data]   Unzipping corpora\stopwords.zip.
```

```
In [47]: stop_updated = stop_nltk + list(punctuation) + ["..."] + [".."]
         reviews_sw_removed=[]
         for sent in df_lemm:
             reviews_sw_removed.append([term for term in sent if term not in stop_updated])
```

```
In [50]: reviews_sw_removed[1]

Out[50]: ['mobile',
          'battery',
          'hell',
          'backup',
          'hour',
          'us',
          'idle',
          'discharged.this',
          'lie',
          'amazon',
          'lenove',
          'battery',
          'charger',
          'hour']
```

8. Create a topic model using LDA on the cleaned up data with 12 topics.
    a. Print out the top terms for each topic.
    b. What is the coherence of the model with the c_v metric?

```
In [52]: id2word = corpora.Dictionary(reviews_sw_removed)
         texts = reviews_sw_removed
         corpus = [id2word.doc2bow(text) for text in texts]
```

```
In [54]: print(corpus[200])

         [(36, 1), (143, 1), (314, 1), (415, 1), (416, 1)]
```

```
In [55]: lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                                    id2word=id2word,
                                                    num_topics=12,
                                                    random_state=42,
                                                    passes=10,
                                                    per_word_topics=True)
```

```
In [56]: pprint(lda_model.print_topics())

         [(0,
           '0.167*"mobile" + 0.049*"screen" + 0.034*"call" + 0.028*"option" + '
           '0.028*"video" + 0.025*"feature" + 0.019*"music" + 0.018*"app" + '
           '0.017*"cast" + 0.016*"sensor"'),
          (1,
           '0.066*"delivery" + 0.050*"superb" + 0.050*"glass" + 0.048*"h" + '
           '0.031*"device" + 0.030*"thanks" + 0.027*"super" + 0.026*"slot" + '
           '0.026*"gorilla" + 0.024*"card"'),
          (2,
           '0.151*"note" + 0.094*"lenovo" + 0.078*"k8" + 0.017*"device" + 0.015*"model" '
           '+ 0.015*"system" + 0.012*"atmos" + 0.011*"version" + 0.010*"power" + '
           '0.010*"k4"'),
          (3,
```

```
In [57]: coherence_model_lda = CoherenceModel(model=lda_model, texts=reviews_sw_removed, dictionary=id2word, coherence='c_v')
         coherence_lda = coherence_model_lda.get_coherence()
         print('\nCoherence Score: ', coherence_lda)

         Coherence Score:  0.5572093987253456
```

9. Analyze the topics through the business lens.
    a. Determine which of the topics can be combined.

In [65]:
```
res = Counter(term_list)
res.most_common(100)
```

Out[65]:
```
[('phone', 7007),
 ('camera', 3273),
 ('battery', 3143),
 ('product', 2261),
 ('problem', 1565),
 ('mobile', 1517),
 ('issue', 1490),
 ('quality', 1387),
 ('note', 1163),
 ('lenovo', 1003),
 ('time', 1003),
 ('performance', 952),
 ('price', 924),
 ('day', 897),
 ('feature', 841),
 ('backup', 661),
 ('money', 642),
 ('k8', 619),
 ('....', 618),
 ('amazon', 582),
 ('heating', 570),
 ('screen', 549),
 ('network', 515),
 ('hour', 506),
 ('month', 506),
 ('service', 506),
 ('call', 480),
 ('charger', 462),
 ('device', 446),
 ('option', 390),
```

In [60]:
```
pprint(lda_model.print_topics())
```

```
[(0,
  '0.167*"mobile" + 0.049*"screen" + 0.034*"call" + 0.028*"option" + '
  '0.028*"video" + 0.025*"feature" + 0.019*"music" + 0.018*"app" + '
  '0.017*"cast" + 0.016*"sensor"'),
 (1,
  '0.066*"delivery" + 0.050*"superb" + 0.050*"glass" + 0.048*"h" + '
  '0.031*"device" + 0.030*"thanks" + 0.027*"super" + 0.026*"slot" + '
  '0.026*"gorilla" + 0.024*"card"'),
 (2,
  '0.151*"note" + 0.094*"lenovo" + 0.078*"k8" + 0.017*"device" + 0.015*"model" '
  '+ 0.015*"system" + 0.012*"atmos" + 0.011*"version" + 0.010*"power" + '
  '0.010*"k4"'),
 (3,
  '0.230*"problem" + 0.117*"...." + 0.107*"heating" + 0.097*"performance" + '
  '0.088*"battery" + 0.049*"....." + 0.022*"issue" + 0.016*"hang" + '
  '0.013*"awesome" + 0.011*"cell"'),
 (4,
  '0.188*"battery" + 0.077*"phone" + 0.046*"charger" + 0.044*"hour" + '
  '0.036*"backup" + 0.035*"heat" + 0.035*"day" + 0.034*"life" + 0.031*"charge" '
  '+ 0.023*"hai"'),
 (5,
  '0.122*"price" + 0.104*"money" + 0.062*"value" + 0.058*"handset" + '
  '0.045*"range" + 0.043*"feature" + 0.034*"mobile" + 0.028*"please" + '
  '0.021*"pls" + 0.018*"experience"'),
 (6,
  '0.098*"speaker" + 0.074*"sound" + 0.071*"display" + 0.040*"work" + '
  '0.028*"month" + 0.025*"set" + 0.024*"volume" + 0.020*"class" + '
```

10. Create topic model using LDA with what you think is the optimal number of topics
    a. What is the coherence of the model?

In [94]:
```
# Build LDA model
lda_model9 = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                             id2word=id2word,
                                             num_topics=9,
                                             random_state=42,
                                             passes=10,
                                             per_word_topics=True)

#Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model9, texts=reviews_sw_removed, dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Coherence Score:  0.5746914918351291
```

In [ ]:
```
# The coherence is now 0.57 which is a significant increase from 0.55 previously.
```

11. The business should be able to interpret the topics.
    a. Name each of the identified topics.
    b. Create a table with the topic name and the top 10 terms in each to present to the business.

In [97]:
```
x = lda_model9.show_topics(formatted=False)
topics_words = [(tp[0], [wd[0] for wd in tp[1]]) for tp in x]
```

In [98]:
```
for topic,words in topics_words:
    print(str(topic)+ "::"+ str(words))
print()
```

```
0::['mobile', 'feature', 'screen', 'call', 'option', 'video', 'app', 'music', 'apps', 'cast']
1::['delivery', 'return', 'glass', 'h', 'device', 'amazon', 'policy', 'super', 'gorilla', 'volta']
2::['phone', 'note', 'lenovo', 'k8', 'time', 'issue', 'service', 'day', 'problem', 'network']
3::['problem', 'issue', 'battery', 'phone', 'heating', 'performance', 'camera', 'network', 'update', 'drain']
4::['battery', 'charger', 'hour', 'backup', 'heat', 'charge', 'phone', 'hai', 'charging', 'turbo']
5::['product', 'money', 'waste', 'value', 'handset', 'price', 'amazon', 'experience', 'lenovo', 'plz']
6::['speaker', 'superb', '.....', 'display', 'mobile', 'sound', 'work', '......', 'set', 'item']
7::['phone', 'camera', 'price', 'battery', 'quality', 'performance', 'feature', 'range', 'mode', 'processor']
8::['camera', 'quality', '....', 'battery', 'everything', 'clarity', 'expectation', 'headphone', 'speed', 'mark']
```

In [ ]:
```
#possible topics from terms present

#topic0 = call and video features
#Topic1 = amazon
#Topic2 = service related issues
#Topic3 = battery related issues
#Topic4 = product accessories
#Topic5 = pricing
#Topic6 = sound features
#Topic7 = overall general phone features
#Topic8 = phone performance
```