In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv("training.tsv", sep='\t')
```

In [3]:

```
df.shape
```

Out[3]:

```
(1200000, 3)
```

In [4]:

```
df.head(10)
```

Out[4]:

	title	description	category
0	ZicZac // Black + Red (Euro: 44)	Clothing & related products (B2C) - Shoes and ...	R
1	9X9 RESISTA/484938	Publishing/Printing - Printing Services	S
2	Halle Pant - Short Inseam 013049561D0010001_ 02	Clothing & related products (B2C) - General	R
3	Harry Houser Travel Expenses - Meals	Security - personnel	S
4	Tee Time: 740078609 : Greens Fee - Composite	Admissions - Green Fees for Privately Owned Go...	R
5	Flat Rate (5-7 Business Days) Shipping line: 4...	Shipping Only - common carrier - FOB destination	R
6	Travel to Water Batteries Plant 1 During regul...	Repair (other) - Performed on TPP (labor only)	S
7	F5 Networks Consulting Services Standard Hourl...	Installation - associated with the sale of TPP...	S
8	Network Time and Materials Services - May 2019...	Consulting - Systems	S
9	2c92a0ad707bb947017095aa4a973307	Cloud Services-Platform as a Service (PaaS)	S

In [5]:

```
df['description']
```

Out[5]:

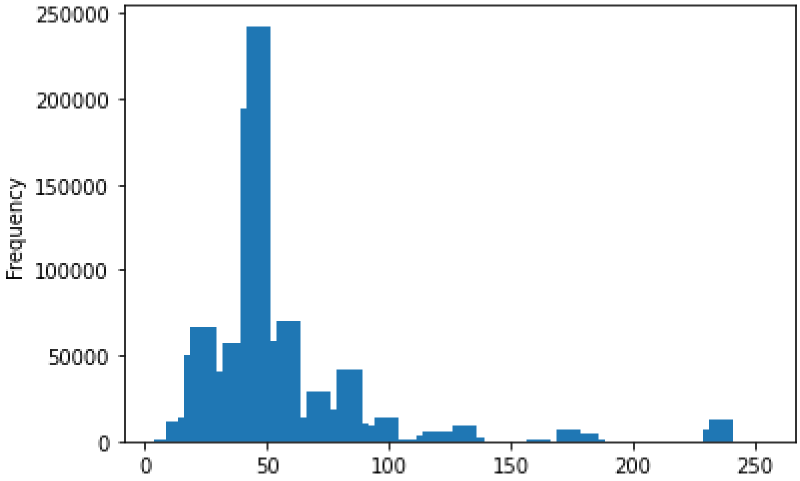
```
0          Clothing & related products (B2C) - Shoes and ...
1                    Publishing/Printing - Printing Services
2                Clothing & related products (B2C) - General
3                                       Security - personnel
4          Admissions - Green Fees for Privately Owned Go...
                                 ...                        
1199995             Cloud Services - SaaS - Service Agreement
1199996              Videos - Streaming / electronic download
1199997    Clothing & related products (Business-To-Custo...
1199998              Movies - Streaming / electronic download
1199999    Optional maintenance agreements related to the...
Name: description, Length: 1200000, dtype: object
```

In [6]:

```
df['descp_len'] = df['description'].astype(str).apply(len)
```

In [7]:

```
ax = df['descp_len'].plot.hist(bins=100, alpha=1.0, width=10)
```

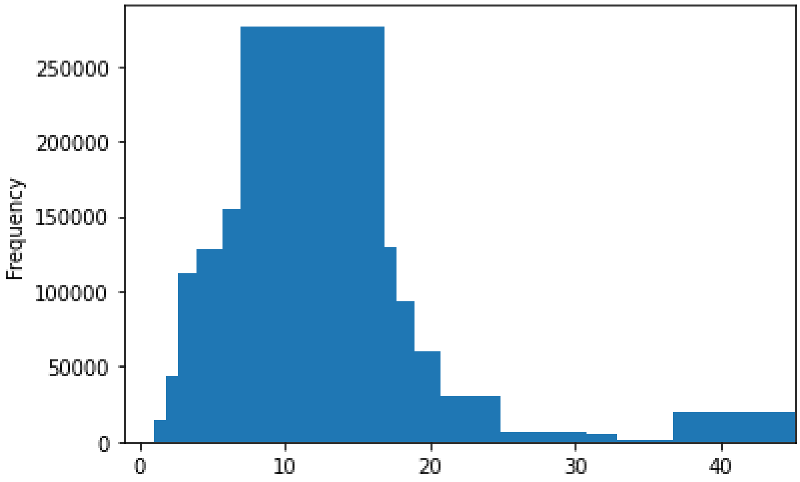


In [8]:

```
df['word_count'] = df['description'].apply(lambda x: len(str(x).split()))
```

In [9]:

```
ax1 = df['word_count'].plot.hist(bins=100, alpha=1.0, width=10)
```



In [10]:

```
df['class'] = df['category'].replace(to_replace ="S",
                 value =1)
```

In [11]:

```
df['class'] = df['class'].replace(to_replace ="R",
                 value =0)
```

In [12]:

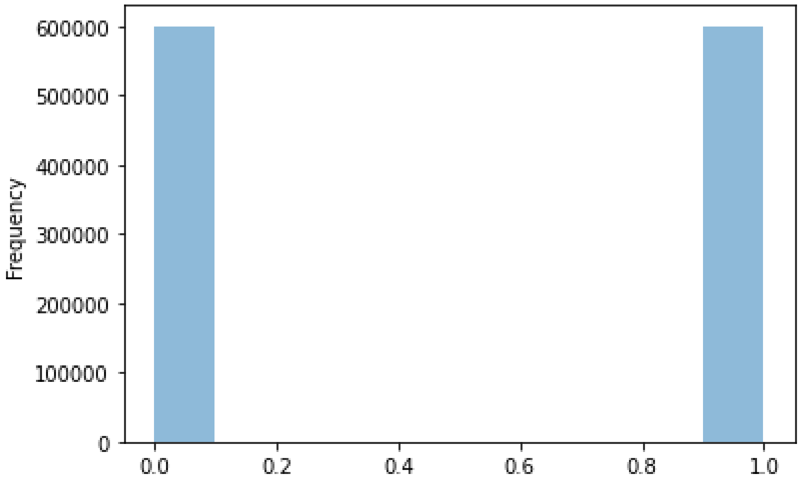
```
df['class']
```

Out[12]:

```
0          0
1          1
2          0
3          1
4          0
          ..
1199995    1
1199996    0
1199997    0
1199998    0
1199999    1
Name: class, Length: 1200000, dtype: int64
```

In [13]:

```
ax1 = df['class'].plot.hist(bins=10, alpha=0.5)
```



In [15]:

```
# almost equal number of samples for each class. Balanced Data.
```

In [14]:

```
df[~df['description'].isna()]['description']
```

Out[14]:

```
0          Clothing & related products (B2C) - Shoes and ...
1                    Publishing/Printing - Printing Services
2                Clothing & related products (B2C) - General
3                                        Security - personnel
4          Admissions - Green Fees for Privately Owned Go...
                                 ...                        
1199995             Cloud Services - SaaS - Service Agreement
1199996              Videos - Streaming / electronic download
1199997    Clothing & related products (Business-To-Custo...
1199998              Movies - Streaming / electronic download
1199999    Optional maintenance agreements related to the...
Name: description, Length: 1200000, dtype: object
```

In [18]:

```
from sklearn.feature_extraction.text import CountVectorizer
def get_top_n_words(corpus, n=None):

    vec = CountVectorizer().fit(corpus.astype('U'))
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq = sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

corpus = df[~df['description'].isna()]

corpus_title = corpus['description']

corpus_service = corpus[corpus['category']=='S']['description']
corpus_product = corpus[corpus['category']=='R']['description']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_words(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_words(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newdescp' , 'count'])
df_s = df_s.groupby('newdescp').sum()['count'].sort_values(ascending=False)

df_p = pd.DataFrame(common_words_product, columns = ['newdescp' , 'count'])
df_p = df_p.groupby('newdescp').sum()['count'].sort_values(ascending=False)
```

```
(600178,)
(599822,)
```

In [19]:

```
df_s  # top 20 service words
```

Out[19]:

```
newdescp
software       21122
services       20852
website        11324
in             10524
of              9948
not             9720
hosted          8664
server          8519
state           8519
asp             8093
cloud           7483
agreement       7276
only            7059
saas            6904
hosting         6654
service         6570
optional        6378
maintenance     6342
tpp             6209
information     6150
Name: count, dtype: int64
```

In [20]:

```
df_p   # top 20 product words.
```

Out[20]:

```
newdescp
products       26061
clothing       24812
related        24802
b2c            24013
general        14717
electronic     10843
streaming      10842
download       10842
movies          8493
food            8314
tpp             6010
personal        5936
property        5935
tangible        5931
and             5853
ingredients     3955
admissions      3648
supplements     3537
for             3396
golf            2866
Name: count, dtype: int64
```

In [21]:

```
# from the above top 20 words we can see that for services mostly the date and t
ime are mentioned.
# Whereas for products we can see that those date and time are not common words.
# So Date and Time related words are crucial for distinguishing between them.
```

In [22]:

```
def get_top_n_words(corpus, n=None):
    vec = CountVectorizer(stop_words = 'english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.ite
ms()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]


corpus = df[~df['description'].isna()]

corpus_title = corpus['description']

corpus_service = corpus[corpus['category']=='S']['description']
corpus_product = corpus[corpus['category']=='R']['description']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_words(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_words(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newdescription' , 'count'
])
df_s = df_s.groupby('newdescription').sum()['count'].sort_values(ascending=False
)

df_p = pd.DataFrame(common_words_product, columns = ['newdescription' , 'count'
])
df_p = df_p.groupby('newdescription').sum()['count'].sort_values(ascending=False
)
```

```
(600178,)
(599822,)
```

In [23]:

```
# top 20 words in products after removing stop words.
df_p
```

Out[23]:

```
newdescription
products       26066
clothing       24787
related        24733
b2c            23915
general        14803
electronic     10821
streaming      10821
download       10821
movies          8532
food            8146
tpp             6216
property        6135
personal        6131
tangible        6130
ingredients     3895
admissions      3632
supplements     3474
golf            2867
fees            2852
green           2840
Name: count, dtype: int64
```

In [24]:

```
# top 20 words after removing stop words for service. words like 'for' and 'of'
 are eliminated.
df_s
```

Out[24]:

```
newdescription
services       21100
software       20766
website        11335
hosted          8366
state           8236
server          8236
asp             7792
cloud           7555
agreement       7328
saas            6970
hosting         6712
service         6561
optional        6441
maintenance     6408
tpp             6309
information     6183
code            6166
sale            6128
downloaded      6007
computer        5971
Name: count, dtype: int64
```

In [25]:

```
def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

corpus = df[~df['description'].isna()]

corpus_title = corpus['description']

corpus_service = corpus[corpus['category']=='S']['description']
corpus_product = corpus[corpus['category']=='R']['description']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_bigram(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_bigram(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newdescription' , 'count' ])
df_s_bigram = df_s.groupby('newdescription').sum()['count'].sort_values(ascending=False)

df_p = pd.DataFrame(common_words_product, columns = ['newdescription' , 'count' ])
df_p_bigram = df_p.groupby('newdescription').sum()['count'].sort_values(ascending=False)
```

```
(600178,)
(599822,)
```

In [26]:

```
df_p_bigram
```

Out[26]:

```
newdescription
related products        24510
clothing related        24282
products b2c            23637
b2c general             13624
electronic download     10951
streaming electronic    10951
movies streaming         8626
tangible personal        6014
personal property        6014
property tpp             5275
food ingredients         4040
food food                3915
fees for                 2787
golf course              2787
for privately            2787
green fees               2787
owned golf               2787
privately owned          2787
admissions green         2787
dietary supplements      1896
Name: count, dtype: int64
```

In [27]:

```
df_s_bigram
```

Out[27]:

```
newdescription
hosted software              8439
software server              8307
server not                   8307
not in                       8307
asp hosted                   7879
in state                     7879
cloud services               7659
services saas                7050
website hosting              6676
computer software            6110
service agreement            5479
saas service                 5479
sale of                      5462
prewritten software          5326
the sale                     5150
electronically downloaded    4505
see additional               4088
avatax system                4088
system tax                   4088
additional avatax            4088
Name: count, dtype: int64
```

In [28]:

```
def get_top_n_bigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(2, 2), stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

corpus = df[~df['description'].isna()]

corpus_title = corpus['description']

corpus_service = corpus[corpus['category']=='S']['description']
corpus_product = corpus[corpus['category']=='R']['description']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_bigram(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_bigram(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newdescription' , 'count'])
df_s_bigram = df_s.groupby('newdescription').sum()['count'].sort_values(ascending=False)

df_p = pd.DataFrame(common_words_product, columns = ['newdescription' , 'count'])
df_p_bigram = df_p.groupby('newdescription').sum()['count'].sort_values(ascending=False)
```

```
(600178,)
(599822,)
```

In [29]:

```
df_p_bigram # after stop word removal.
```

Out[29]:

```
newdescription
related products        24419
clothing related        24338
products b2c            23575
b2c general             13727
electronic download     11132
streaming electronic    11132
movies streaming         8770
tangible personal        6070
personal property        6070
property tpp             5321
food ingredients         3900
food food                3896
fees privately           2721
green fees               2721
golf course              2721
owned golf               2721
privately owned          2721
admissions green         2721
dietary supplements      1859
ingredients dietary      1842
Name: count, dtype: int64
```

In [30]:

```
df_s_bigram # after stop word removal.
```

Out[30]:

```
newdescription
hosted software              8412
software server              8280
server state                 8280
asp hosted                   7859
cloud services               7788
services saas                7175
website hosting              6674
computer software            6054
service agreement            5550
saas service                 5550
prewritten software          5253
electronically downloaded    4461
code information             4235
avatax tax                   4235
tax code                     4235
additional avatax            4235
software maintenance         3347
performed tpp                3243
repair performed             3243
website domain               3172
Name: count, dtype: int64
```

In [31]:

```
def get_top_n_trigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3, 3)).fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

corpus = df[~df['description'].isna()]

corpus_title = corpus['description']

corpus_service = corpus[corpus['category']=='S']['description']
corpus_product = corpus[corpus['category']=='R']['description']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_trigram(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_trigram(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newdescription' , 'count' ])
df_s_trigram = df_s.groupby('newdescription').sum()['count'].sort_values(ascending=False)

df_p = pd.DataFrame(common_words_product, columns = ['newdescription' , 'count' ])
df_p_trigram = df_p.groupby('newdescription').sum()['count'].sort_values(ascending=False)
```

```
(600178,)
(599822,)
```

In [32]:

```
df_s_trigram
```

Out[32]:

```
newdescription
hosted software server            8351
software server not               8351
server not in                     8351
asp hosted software               7924
not in state                      7924
cloud services saas               7023
services saas service             5457
saas service agreement            5457
the sale of                       5164
additional avatax system          4090
tax code information              4090
system tax code                   4090
see additional avatax             4090
avatax system tax                 4090
repair other performed            3355
other performed on                3355
performed on tpp                  3355
computer software maintenance     3209
website domain registration       3173
tpp equipment parts               3011
Name: count, dtype: int64
```

In [33]:

```
df_p_trigram
```

Out[33]:

```
newdescription
clothing related products           24269
related products b2c                23632
products b2c general                13769
streaming electronic download       11008
movies streaming electronic          8601
tangible personal property           6143
personal property tpp                5381
food food ingredients                3763
green fees for                       2831
fees for privately                   2831
for privately owned                  2831
admissions green fees                2831
owned golf course                    2831
privately owned golf                 2831
food ingredients dietary             1915
ingredients dietary supplements      1914
videos streaming electronic          1836
socks and stockings                  1826
products b2c socks                   1819
b2c socks and                        1819
Name: count, dtype: int64
```

In [34]:

```
def get_top_n_trigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3, 3), stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

corpus = df[~df['description'].isna()]

corpus_title = corpus['description']

corpus_service = corpus[corpus['category']=='S']['description']
corpus_product = corpus[corpus['category']=='R']['description']

print(corpus_service.shape)
print(corpus_product.shape)

common_words_service = get_top_n_trigram(corpus_service.sample(frac=0.1), 20)
common_words_product = get_top_n_trigram(corpus_product.sample(frac=0.1), 20)


df_s = pd.DataFrame(common_words_service, columns = ['newdescription' , 'count'])
df_s_trigram = df_s.groupby('newdescription').sum()['count'].sort_values(ascending=False)

df_p = pd.DataFrame(common_words_product, columns = ['newdescription' , 'count'])
df_p_trigram = df_p.groupby('newdescription').sum()['count'].sort_values(ascending=False)
```

```
(600178,)
(599822,)
```

In [35]:

```
df_s_trigram # after removing stop words.
```

Out[35]:

```
newdescription
hosted software server                8291
software server state                 8291
asp hosted software                   7856
cloud services saas                   7046
services saas service                 5476
saas service agreement                5476
additional avatax tax                 4102
tax code information                  4102
avatax tax code                       4102
repair performed tpp                  3396
computer software maintenance         3224
website domain registration           3204
labor separately stated               3058
equipment parts labor                 3058
tpp equipment parts                   3058
parts labor separately                3058
associated sale tpp                   2663
software electronically downloaded    2514
tangible personal property            2431
maintenance agreements related        2428
Name: count, dtype: int64
```

In [36]:

```
df_p_trigram  # after removing stop words.
```

Out[36]:

```
newdescription
clothing related products          24484
related products b2c               23705
products b2c general               13661
streaming electronic download      11019
movies streaming electronic         8658
tangible personal property          5995
personal property tpp               5260
food food ingredients               3899
owned golf course                   2818
fees privately owned                2818
green fees privately                2818
admissions green fees               2818
privately owned golf                2818
products b2c socks                  1851
b2c socks stockings                 1851
food ingredients dietary            1846
ingredients dietary supplements     1843
videos streaming electronic         1829
products b2c cosmetics              1720
shipping common carrier             1538
Name: count, dtype: int64
```

In [37]:

```
df['description'].shape
```

Out[37]:

```
(1200000,)
```

In [38]:

```
from textblob import TextBlob
import nltk
nltk.download('averaged_perceptron_tagger')

all = [str(i) for i in list(df['description'].sample(frac=0.05).values)]
all = ' '.join(all)
blob = TextBlob(all)
#print(blob.tags)
pos_df = pd.DataFrame(blob.tags, columns = ['word' , 'pos'])
print(pos_df['pos'])
pos_df = pos_df.pos.value_counts()
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/rnsandeep/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

```
0         NNS
1         NNP
2         VBG
3          JJ
4          JJ
         ... 
404173     NN
404174     JJ
404175    NNP
404176    NNP
404177    NNP
Name: pos, Length: 404178, dtype: object
```

In [39]:

```
pos_df # part of speech.
```

Out[39]:

```
NNP      153833
NN        47681
JJ        36831
NNS       35578
VBN       24154
CC        22452
IN        22109
RB        19650
VBG       10738
NNPS       8425
VBD        6057
DT         5626
CD         3522
VBZ        2242
TO         2097
FW         1074
RP          980
VB          639
VBP         208
POS         134
JJR          57
PRP$         42
SYM          28
WRB          11
PRP           4
WDT           4
JJS           1
MD            1
Name: pos, dtype: int64
```

In [ ]:

```

```