In [17]:

```
%matplotlib inline
```

In [24]:

```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

Read reviews data

In [192]:

```
reviews_file = '/Users/jiaweihe/Apps/spark-1.5.2/data/yelp_dataset_challenge_ac
ademic_dataset/yelp_academic_dataset_review.json'
reviews = sqlContext.read.json(reviews_file)
reviews.first()
```

Out[192]:

```
Row(business_id=u'vcNAWiLM4dR7D2nwwJ7nCA', date=u'2007-05-17', revie
w_id=u'15SdjuK7DmYqUAj6rjGowg', stars=5, text=u"dr. goldberg offers
everything i look for in a general practitioner.  he's nice and easy
 to talk to without being patronizing; he's always on time in seeing
 his patients; he's affiliated with a top-notch hospital (nyu) which
 my parents have explained to me is very important in case something
 happens and you need surgery; and you can get referrals to see spec
ialists without having to see him first.  really, what more do you n
eed?  i'm sitting here trying to think of any complaints i have abou
t him, but i'm really drawing a blank.", type=u'review', user_id=u'X
qd0DzHaiyRqVH3WRG7hzg', votes=Row(cool=1, funny=0, useful=2))
```

Rating counts

In [10]:

```
reviews_stars = reviews.map(lambda x: x[3])
```

In [22]:

```
min_rating = reviews_stars.reduce(lambda x,y : min(x,y))
max_rating = reviews_stars.reduce(lambda x,y : max(x,y))
```

In [23]:

```
print 'Min rating: {}'.format(min_rating)
print 'Max rating: {}'.format(max_rating)
```

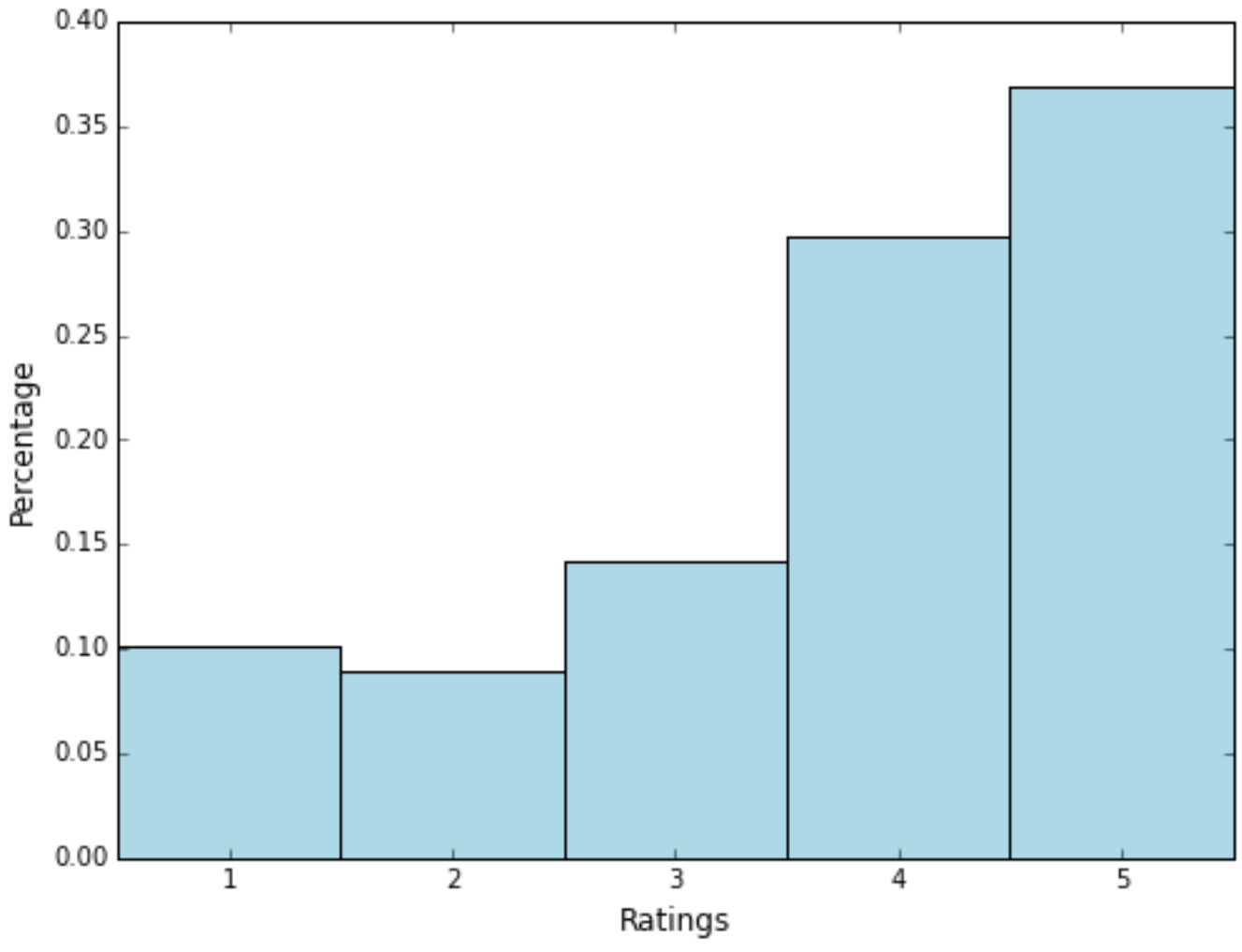
```
Min rating: 1
Max rating: 5
```

In [ ]:

```
count_by_rating = reviews_stars.countByValue()
```

In [181]:

```
x_axis = np.array(count_by_rating.keys())
y_axis = np.array([float(c) for c in count_by_rating.values()])
# we normalize the y-axis here to percentages
y_axis_normed = y_axis / y_axis.sum()
pos = np.arange(len(x_axis))
width = 1.0
ax = plt.axes()
ax.set_xticks(pos + (width / 2))
ax.set_xticklabels(x_axis)
plt.bar(pos, y_axis_normed, width, color='lightblue')
plt.xticks()
plt.xlabel('Ratings')
plt.ylabel('Percentage')
fig = plt.gcf()
fig.set_size_inches(8, 6)
```



Word Counts

In [131]:

```
import nltk.stem
stem = nltk.stem.SnowballStemmer('english')
from nltk.corpus import stopwords
stopWords = stopwords.words("english")
```

Out[131]:

```
u'graphic'
```

In [180]:

```
stem.stem("Dancing")
```

Out[180]:

```
u'danc'
```

In [196]:

```
stopWords[:20]
```

Out[196]:

```
[u'i',
 u'me',
 u'my',
 u'myself',
 u'we',
 u'our',
 u'ours',
 u'ourselves',
 u'you',
 u'your',
 u'yours',
 u'yourself',
 u'yourselves',
 u'he',
 u'him',
 u'his',
 u'himself',
 u'she',
 u'her',
 u'hers']
```

In [143]:

```
def get_words(x):
    ### get text words from reviews table
    try:
        s = str(x[4].strip())
        ### get rid of punctuation
        stripped_s = s.translate(string.maketrans("",""), string.punctuation)
        ### remove stop words, "\n", leading and trailing whitespace
        return [word for word in stripped_s.replace("\n"," ").strip().split(" "
) if word not in stopWords]
    except:
        ### UnicodeEncodeError: 'ascii' codec can't encode character u'\xed' in
 position 1003: ordinal not in range(128)
        return ""
```

In [144]:

```
rdd_texts = reviews.flatMap(lambda x: get_words(x))
```

In [145]:

```
rdd_texts.first()
```

Out[145]:

```
'dr'
```

In [148]:

```
### not working: ImportError: No module named nltk.stem.snowball
rdd_stemmed_texts = rdd_texts.map(lambda x : stem.stem(x))
```

In [151]:

```
wordCounts = rdd_texts.countByValue()
```

In [169]:

```
wordCounts_sorted = sorted(wordCounts.items(), key=lambda x: x[1], reverse=True
)
```

In [194]:

```
wordCounts_sorted[:30]
```

Out[194]:

```
[('I', 4843867),
 ('The', 1684681),
 ('place', 860426),
 ('good', 839233),
 ('food', 779140),
 ('like', 644014),
 ('great', 583999),
 ('get', 568243),
 ('time', 533053),
 ('We', 532472),
 ('one', 517918),
 ('would', 456766),
 ('back', 449195),
 ('service', 448360),
 ('go', 443821),
 ('really', 437825),
 ('It', 399264),
 ('They', 381095),
 ('This', 350447),
 ('us', 318242),
 ('got', 306747),
 ('My', 306103),
 ('nice', 301650),
 ('dont', 300502),
 ('also', 299930),
 ('Im', 293104),
 ('even', 280272),
 ('little', 270659),
 ('Ive', 268060),
 ('well', 259919)]
```

In [176]:

```
### eliminate upper_class stopWords
for k,v in wordCounts.items():
    try:
        if stem.stem(k) in stopWords:
            del wordCounts[k]
    ##AttributeError: 'int' object has no attribute 'lower'
    except:
        pass
```

In [177]:

```
wordCounts_sorted_stemmed = sorted(wordCounts.items(), key=lambda x: x[1], reve
rse=True)
```

In [195]:

```
wordCounts_sorted_stemmed[:30]
```

Out[195]:

```
[('place', 860426),
 ('good', 839233),
 ('food', 779140),
 ('like', 644014),
 ('great', 583999),
 ('get', 568243),
 ('time', 533053),
 ('one', 517918),
 ('would', 456766),
 ('back', 449195),
 ('service', 448360),
 ('go', 443821),
 ('really', 437825),
 ('us', 318242),
 ('got', 306747),
 ('nice', 301650),
 ('dont', 300502),
 ('also', 299930),
 ('Im', 293104),
 ('even', 280272),
 ('little', 270659),
 ('Ive', 268060),
 ('well', 259919),
 ('didnt', 254223),
 ('best', 251523),
 ('much', 247143),
 ('always', 245083),
 ('ordered', 239653),
 ('people', 237330),
 ('restaurant', 231180)]
```

In [ ]:

```

```

Review ages (review years from now)

In [31]:

```
def get_year(x):
    return int(x[:4])

rdd_years = reviews.map(lambda x: int(x[1][:4]))
```

In [33]:

```
max_year = rdd_years.reduce(lambda x,y: max(x,y))
min_year = rdd_years.reduce(lambda x,y: min(x,y))
```

In [34]:

```
max_year
```

Out[34]:

```
2015
```

In [35]:

```
min_year
```

Out[35]:

```
2004
```

In [36]:

```
rating_ages = rdd_years.map(lambda yr: 2016 - yr).countByValue()
```

In [39]:

```
rating_ages
```

Out[39]:

```
defaultdict(<type 'int'>, {1: 14665, 2: 486306, 3: 336273, 4: 244106
, 5: 209429, 6: 137764, 7: 72948, 8: 45117, 9: 17724, 10: 4239, 11:
680, 12: 13})
```

In [66]:

```
plt.bar(rating_ages.keys(), rating_ages.values(), width, color='g')
plt.xlabel('Rating Ages')
plt.ylabel('Counts')
x_axis = np.array(rating_ages.keys())
pos = np.array(rating_ages.keys())
width = 1.0
ax = plt.axes()
ax.set_xticks(pos + (width / 2))
ax.set_xticklabels(x_axis)
fig = plt.gcf()
fig.set_size_inches(8, 6)
```