```
import pandas as pd
import numpy as np
from collections import Counter
import re


np.random.seed(42)
sms_data = pd.DataFrame({
    'label': np.random.choice(['ham', 'spam'], 1000, p=[0.86, 0.14]),
    'message': np.random.choice([
        "Free entry in 2 a wkly comp to win FA Cup",
        "Nah I don't think he goes to usf, he lives around here though",
        "Win a brand new car just text WIN to 80082",
        "Hey there, what are you doing today?",
        "URGENT! Your mobile No. was awarded a £2,000 prize",
        "Ok lar... Joking wif u oni...",
        "As per your request your subscription has been renewed",
        "You have 1 unread message. Click to read",
        "Sorry, I missed your call. Can we talk later?",
        "Congratulations! You've won a $1,000 Walmart gift card"
    ], 1000)
})

# Add additional columns
sms_data['message_length'] = sms_data['message'].str.len()
sms_data['word_count'] = sms_data['message'].str.split().str.len()
```

1. Count total number of messages

```
print("Total messages:", len(sms_data))
```

```
Total messages: 1000
```

2. Count number of spam and ham messages

```
print("Spam messages:", (sms_data['label'] == 'spam').sum())
print("Ham messages:", (sms_data['label'] == 'ham').sum())
```

```
Spam messages: 146
Ham messages: 854
```

3. Calculate the percentage of spam messages

```
print("Spam percentage:", round((sms_data['label'] == 'spam').mean() * 100, 2), "%")
```

```
Spam percentage: 14.6 %
```

4. Find the average length of messages

```
print("Average message length:", sms_data['message_length'].mean())
```

```
Average message length: 45.565
```

5. Find the shortest and longest messages

```
print("Longest message:\n", sms_data.loc[sms_data['message_length'].idxmax()])
print("Shortest message:\n", sms_data.loc[sms_data['message_length'].idxmin()])
```

```
Longest message:
 label                                                    spam
message           Nah I don't think he goes to usf, he lives aro...
message_length                                                61
word_count                                                    13
Name: 11, dtype: object
Shortest message:
 label                                  spam
message           Ok lar... Joking wif u oni...
message_length                              29
word_count                                   6
Name: 7, dtype: object
```

6. Compare average message length for spam vs. ham

```
print("Avg message length by label:\n", sms_data.groupby('label')['message_length'].mean())
```

```
Avg message length by label:
 label
ham     45.721311
spam    44.650685
Name: message_length, dtype: float64
```

7. Count how many messages contain the word “free”

```
print("Messages with 'free':", sms_data['message'].str.contains('free', case=False).sum())
```

```
Messages with 'free': 98
```

8. Count how many messages contain digits

```
print("Messages with digits:", sms_data['message'].str.contains(r'\d').sum())
```

```
Messages with digits: 488
```

9. Find the number of messages with more than 10 words

```
print("Messages >10 words:", (sms_data['word_count'] > 10).sum())
```

```
Messages >10 words: 216
```

10. Find how many messages are exact duplicates

```
print("Duplicate messages:", sms_data.duplicated('message').sum())
```

```
Duplicate messages: 990
```

11. Find the most common word in spam messages

```
spam_words = ' '.join(sms_data[sms_data['label'] == 'spam']['message']).lower().split()
print("Most common word in spam:", Counter(spam_words).most_common(1))
```

```
Most common word in spam: [('to', 67)]
```

12. Get the top 5 most frequent words in all messages

```
all_words = ' '.join(sms_data['message']).lower().split()
print("Top 5 most common words:", Counter(all_words).most_common(5))
```

```
Top 5 most common words: [('to', 408), ('your', 402), ('a', 389), ('win', 284), ('he', 236)]
```

13. Find how many messages contain a URL (basic pattern matching for 'http' or 'www')

```
print("Messages with URLs:", sms_data['message'].str.contains(r'http|www', case=False).sum())
```

```
Messages with URLs: 0
```

14. Calculate standard deviation and median of message lengths

```
print("Message length standard deviation:", sms_data['message_length'].std())
print("Median message length:", sms_data['message_length'].median())
```

```
Message length standard deviation: 9.255130060265625
Median message length: 45.0
```

15. Identify messages that match typical spam features (like words: ‘win’, ‘prize’, ‘congratulations’, ‘urgent’)

```
spam_keywords = ['win', 'prize', 'congratulations', 'urgent']
keyword_pattern = '|'.join(spam_keywords)
spammy_msgs = sms_data['message'].str.contains(keyword_pattern, case=False, na=False)
print("Messages with spammy keywords:", spammy_msgs.sum())
```

```
Messages with spammy keywords: 389
```

16. Generate a new column: 'is_long' = True if message length > average

```
sms_data['message_length'] = sms_data['message'].str.len()
average_length = sms_data['message_length'].mean()
sms_data['is_long'] = sms_data['message_length'] > average_length
print(sms_data[['message', 'message_length', 'is_long']].head())
```

```
                                             message  message_length  is_long
0           You have 1 unread message. Click to read              40    False
1         Win a brand new car just text WIN to 80082              42    False
2           You have 1 unread message. Click to read              40    False
3  URGENT! Your mobile No. was awarded a £2,000 p...              50     True
4          Free entry in 2 a wkly comp to win FA Cup              41    False
```

17. Find average number of words per message in spam vs ham

```
sms_data['word_count'] = sms_data['message'].str.split().str.len()
avg_words = sms_data.groupby('label')['word_count'].mean()
print("Average word count per label:\n", avg_words)
```

```
Average word count per label:
 label
ham     9.078454
spam    9.109589
Name: word_count, dtype: float64
```

18. Create word clouds for spam and ham messages (requires wordcloud)

Double-click (or enter) to edit

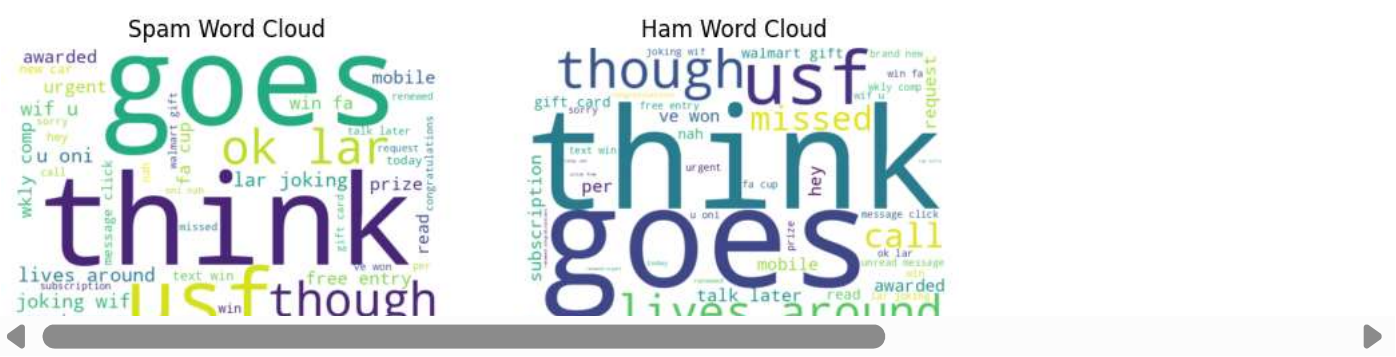
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Combine all messages by label
spam_text = " ".join(sms_data[sms_data['label'] == 'spam']['message']).lower()
ham_text = " ".join(sms_data[sms_data['label'] == 'ham']['message']).lower()

# Generate word clouds
spam_wc = WordCloud(width=600, height=400, background_color='white').generate(spam_text)
ham_wc = WordCloud(width=600, height=400, background_color='white').generate(ham_text)

# Display
plt.figure(figsize=(9, 6))
plt.subplot(1, 2, 1)
plt.imshow(spam_wc, interpolation='bilinear')
plt.axis('off')
plt.title("Spam Word Cloud")

plt.subplot(1, 2, 2)
plt.imshow(ham_wc, interpolation='bilinear')
plt.axis('off')
plt.title("Ham Word Cloud")
plt.show()
```



19. Tokenize messages and calculate average token count for spam vs ham

```
sms_data['tokens'] = sms_data['message'].str.findall(r'\b\w+\b')  # Tokenize words
sms_data['token_count'] = sms_data['tokens'].str.len()
avg_tokens = sms_data.groupby('label')['token_count'].mean()
print("Average token count per label:\n", avg_tokens)
```

```
Average token count per label:
 label
ham     9.492974
spam    9.500000
Name: token_count, dtype: float64
```

20. Use regex to extract all monetary values (e.g., "$100", "£500")

```
money_regex = r'(\£\d+[,.]?\d*|\$\d+[,.]?\d*)'
money_found = sms_data['message'].str.extractall(money_regex)
print("Extracted monetary values:\n", money_found.dropna())
```

```
Extracted monetary values:
                  0
      match
3   0      £2,000
6   0      £2,000
13  0      $1,000
15  0      $1,000
16  0      $1,000
...           ...
971 0      $1,000
973 0      £2,000
984 0      £2,000
987 0      £2,000
998 0      £2,000

[198 rows x 1 columns]
```