NAME : Vedant Pawar
DIVISION : ET-2
Roll No. : ET2-69
PRN : 202401070212
BATCH : E24

DATA SHEET LINK :
https://www.kaggle.com/datasets/wcukierski/enron-email-dataset

```python
import pandas as pd
import numpy as np

try:
    df = pd.read_csv('/kaggle/input/enron-email-dataset/emails.csv')
    print("Dataset loaded successfully.\n")
except Exception as e:
    print(f"Error loading dataset: {e}")
    exit()

print("\nAvailable columns in the dataset:")
print(df.columns.tolist())

try:
    def extract_field(message, field):
        try:
            lines = message.split('\n')
            for line in lines:
                if line.startswith(field):
                    return line.split(':', 1)[1].strip()
        except:
            return np.nan
        return np.nan

    df['sender'] = df['message'].apply(lambda x: extract_field(x, 'From'))
    df['recipient'] = df['message'].apply(lambda x: extract_field(x, 'To'))
    df['subject'] = df['message'].apply(lambda x: extract_field(x, 'Subject'))
    df['date'] = df['message'].apply(lambda x: extract_field(x, 'Date'))
```

```
    def extract_body(message):
        try:
            parts = message.split('\n\n', 1)
            if len(parts) > 1:
                return parts[1]
        except:
            return np.nan
        return np.nan

    df['body'] = df['message'].apply(extract_body)

    print("\nFields extracted successfully!\n")
except Exception as e:
    print(f"Error extracting fields: {e}")
```

```
Dataset loaded successfully.


Available columns in the dataset:
['file', 'message']

Fields extracted successfully!
```

```python
# 1. Display first 5 rows
try:
    print("\n1. First 5 rows:")
    print(df.head())
except Exception as e:
    print(f"Error in Step 1: {e}")
```

```
1. First 5 rows:
                        file  \
0     allen-p/_sent_mail/1.
1    allen-p/_sent_mail/10.
2   allen-p/_sent_mail/100.
3  allen-p/_sent_mail/1000.
4  allen-p/_sent_mail/1001.

                                                 message                    sender  \
0  Message-ID: <18782981.1075855378110.JavaMail.e...  phillip.allen@enron.com
1  Message-ID: <15464986.1075855378456.JavaMail.e...  phillip.allen@enron.com
2  Message-ID: <24216240.1075855687451.JavaMail.e...  phillip.allen@enron.com
3  Message-ID: <13505866.1075863688222.JavaMail.e...  phillip.allen@enron.com
4  Message-ID: <30922949.1075863688243.JavaMail.e...  phillip.allen@enron.com

                 recipient    subject                                 date  \
0    tim.belden@enron.com              Mon, 14 May 2001 16:39:00 -0700 (PDT)
1  john.lavorato@enron.com        Re:   Fri, 4 May 2001 13:51:00 -0700 (PDT)
2   leah.arsdall@enron.com   Re: test  Wed, 18 Oct 2000 03:00:00 -0700 (PDT)
3    randall.gay@enron.com              Mon, 23 Oct 2000 06:13:00 -0700 (PDT)
4     greg.piper@enron.com  Re: Hello  Thu, 31 Aug 2000 05:07:00 -0700 (PDT)

                                                 body
0                           Here is our forecast\n\n
1  Traveling to have a business meeting takes the...
2                      test successful.  way to go!!!
3  Randy,\n\n Can you send me a schedule of the s...
4                 Let's shoot for Tuesday at 11:45.
```

```python
# 2. Total number of emails
try:
    print("\n2. Total number of emails:")
    print(len(df))
except Exception as e:
    print(f"Error in Step 2: {e}")
```

```
2. Total number of emails:
517401
```

```python
# 3. Unique senders
try:
    print("\n3. Unique senders:")
    print(df['sender'].nunique())
except Exception as e:
    print(f"Error in Step 3: {e}")
```

```
3. Unique senders:
20328
```

```python
# 4. Unique recipients
try:
    print("\n4. Unique recipients:")
    print(df['recipient'].nunique())
except Exception as e:
    print(f"Error in Step 4: {e}")
```

```
4. Unique recipients:
51743
```

```python
# 5. Top 5 most active senders
try:
    print("\n5. Top 5 senders:")
    print(df['sender'].value_counts().head(5))
except Exception as e:
    print(f"Error in Step 5: {e}")
```

```
5. Top 5 senders:
sender
kay.mann@enron.com           16735
vince.kaminski@enron.com     14368
jeff.dasovich@enron.com      11411
pete.davis@enron.com          9149
chris.germany@enron.com       8801
Name: count, dtype: int64
```

```python
# 6. Top 5 most emailed recipients
try:
    print("\n6. Top 5 recipients:")
    print(df['recipient'].value_counts().head(5))
except Exception as e:
    print(f"Error in Step 6: {e}")
```

```
6. Top 5 recipients:
recipient
pete.davis@enron.com          9155
tana.jones@enron.com          5677
sara.shackleton@enron.com     4974
vkaminski@aol.com             4870
jeff.dasovich@enron.com       4350
Name: count, dtype: int64
```

```python
# 7. Missing subjects
try:
    print("\n7. Missing subjects:")
    print(df['subject'].isnull().sum())
except Exception as e:
    print(f"Error in Step 7: {e}")
```

```
7. Missing subjects:
0
```

```python
# 8. Average length of email bodies
try:
    print("\n8. Average body length:")
    df['body_length'] = df['body'].astype(str).apply(len)
    print(df['body_length'].mean())
except Exception as e:
    print(f"Error in Step 8: {e}")
```

```
8. Average body length:
1845.3546572194487
```

```python
# 9. Maximum length of an email body
try:
    print("\n9. Maximum body length:")
    print(df['body_length'].max())
except Exception as e:
    print(f"Error in Step 9: {e}")
```

```
9. Maximum body length:
2011422
```

```python
# 10. Most frequent word used in email bodies
try:
    print("\n10. Most frequent word used in email bodies:")
    all_words = ' '.join(df['body'].dropna()).lower().split()
    words_series = pd.Series(all_words)
    print(words_series.value_counts().idxmax())
except Exception as e:
    print(f"Error in Step 10: {e}")
```

```
10. Most frequent word used in email bodies:
the
```

```python
# 11. Number of emails without recipient
try:
    print("\n11. Number of emails with missing recipient:")
    print(df['recipient'].isnull().sum())
except Exception as e:
    print(f"Error in Step 11: {e}")
```

```
11. Number of emails with missing recipient:
19308
```

```python
try:
    print("\n12. Number of emails with empty bodies:")
    empty_bodies = df['body'].isnull().sum() + (df['body'].astype(str).str.strip() == '').sum()
    print(empty_bodies)
except Exception as e:
    print(f"Error in Step 12: {e}")
```

```
12. Number of emails with empty bodies:
0
```

```python
# 13. Most common sender domain (like enron.com)
try:
    print("\n13. Most common sender domain:")
    df['sender_domain'] = df['sender'].astype(str).apply(lambda x: x.split('@')[-1] if '@' in x else np.
    print(df['sender_domain'].value_counts().idxmax())
except Exception as e:
    print(f"Error in Step 13: {e}")
```

```
13. Most common sender domain:
enron.com
```

```python
# 14. Duplicate emails
try:
    print("\n14. Duplicate emails:")
    print(df.duplicated(subset=['subject', 'body']).sum())
except Exception as e:
    print(f"Error in Step 14: {e}")
```

```
14. Duplicate emails:
267213
```

```python
# 15. Emails mentioning 'energy'
try:
    print("\n15. Emails mentioning 'energy':")
    print(df['body'].str.contains('energy', case=False, na=False).sum())
except Exception as e:
    print(f"Error in Step 15: {e}")
```

```
15. Emails mentioning 'energy':
69283
```

```python
# 16. Top 5 common words in subjects
try:
    print("\n16. Top 5 common words in subjects:")
    subject_words = ' '.join(df['subject'].dropna()).lower().split()
    words_series = pd.Series(subject_words)
    print(words_series.value_counts().head(5))
except Exception as e:
    print(f"Error in Step 16: {e}")
```

```
16. Top 5 common words in subjects:
re:    155198
-       46042
for     40956
fw:     39043
of      25917
Name: count, dtype: int64
```

```python
# 17. Percentage of emails to multiple recipients
try:
    print("\n17. Percentage of emails to multiple recipients:")
    multiple = df['recipient'].astype(str).str.contains(';').sum()
    print((multiple / len(df)) * 100)
except Exception as e:
    print(f"Error in Step 17: {e}")
```

```
17. Percentage of emails to multiple recipients:
0.026478495402985306
```

```python
# 18. Pivot table of emails by sender per month
try:
    print("\n18. Pivot table (sender vs month):")
    df['month'] = df['date'].dt.to_period('M')
    pivot = pd.pivot_table(df, index='sender', columns='month', values='subject', aggfunc='count', fill_
    print(pivot)
except Exception as e:
    print(f"Error in Step 18: {e}")
```

```
18. Pivot table (sender vs month):
Error in Step 18: Can only use .dt accessor with datetimelike values
```

```python
# 19. Standard deviation of body lengths
try:
    print("\n19. Standard deviation of body lengths:")
    print(np.std(df['body_length']))
except Exception as e:
    print(f"Error in Step 19: {e}")
```

```
19. Standard deviation of body lengths:
8181.156134536513
```

```python
# 20. Average number of words per sender
try:
    print("\n20. Average words per email per sender:")
    df['word_count'] = df['body'].astype(str).apply(lambda x: len(x.split()))
    print(df.groupby('sender')['word_count'].mean())
except Exception as e:
    print(f"Error in Step 20: {e}")
```

```
20. Average words per email per sender:
sender
'todd'.delahoussaye@enron.com          314.333333
--migrated--bmishkin@ercot.com          21.000000
--migrated--dodle@ercot.com             85.000000
-nikole@excite.com                     582.125000
-persson@ricemail.ricefinancial.com    406.200000
                                           ...
zufferli@enron.com                      11.000000
zulie.flores@enron.com                 100.166667
zvo2z17d0@untappedmarkets.com          152.000000
zwharton@dawray.com                    176.407407
zzmacmac@aol.com                        60.000000
Name: word_count, Length: 20328, dtype: float64
```