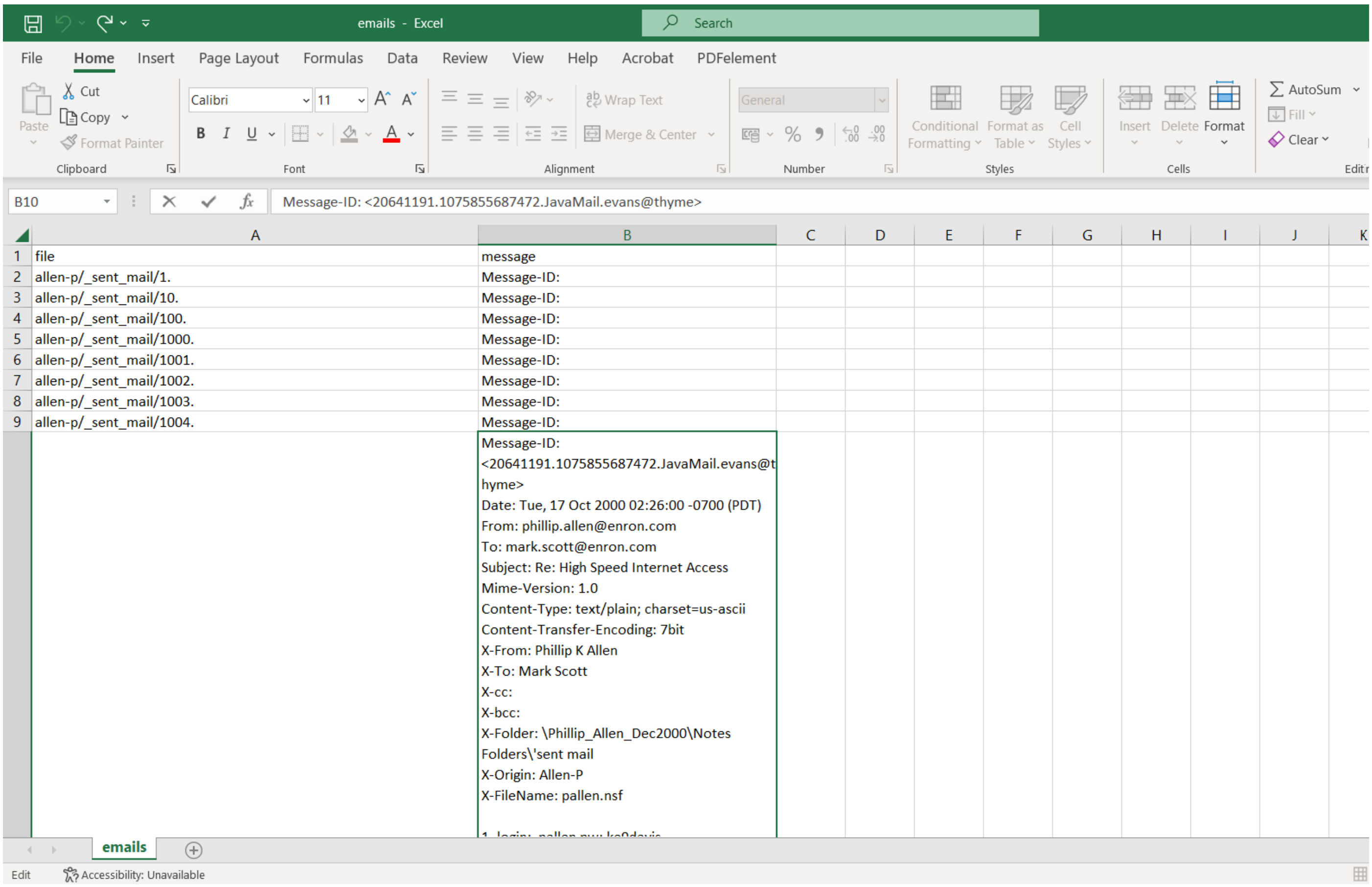
EDS THEORY ACTIVITY NO-1

NAME- POORVA SUNIL MALI

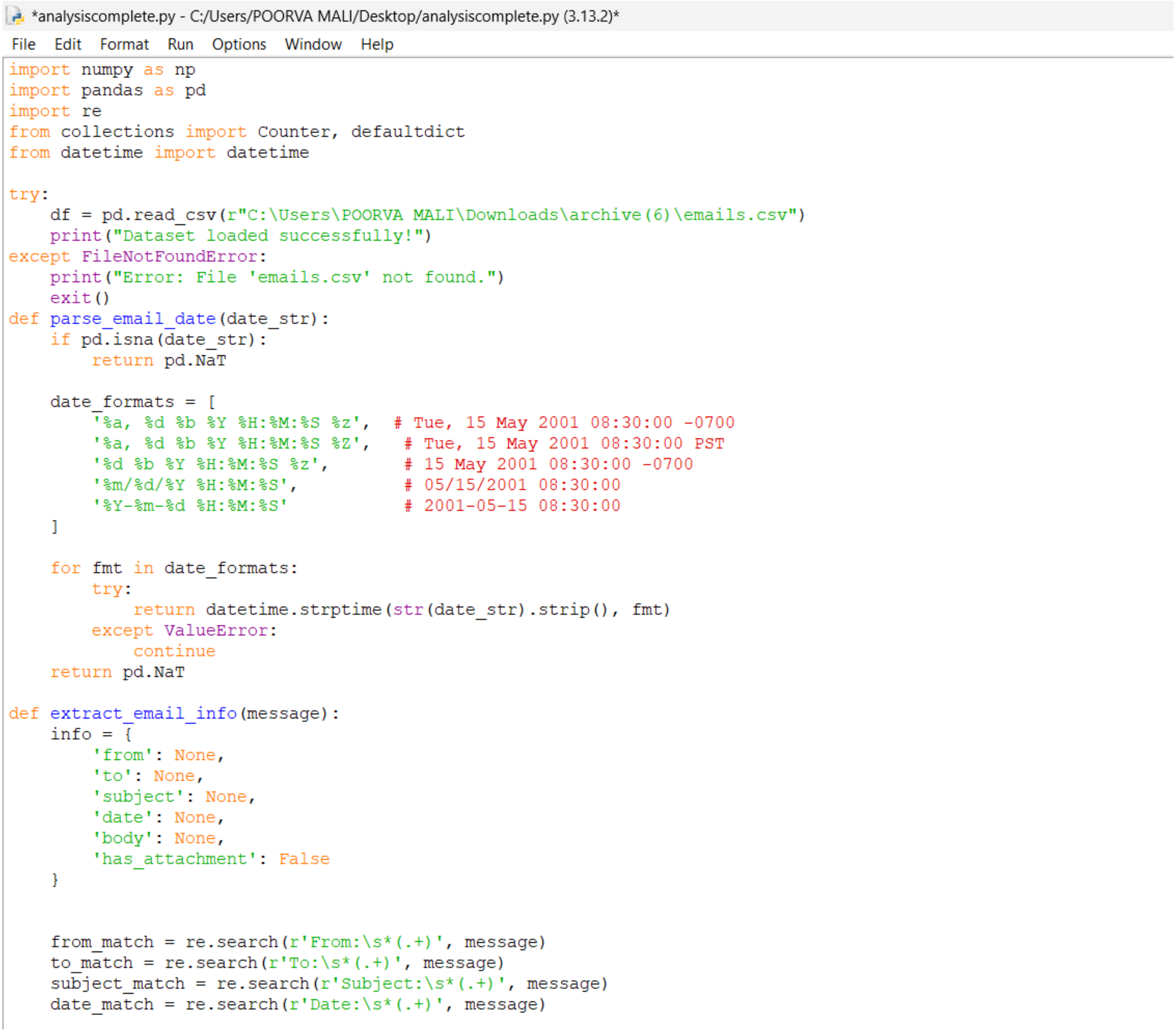
ROLL NO – CS8-27

PRN NO – 202401120027

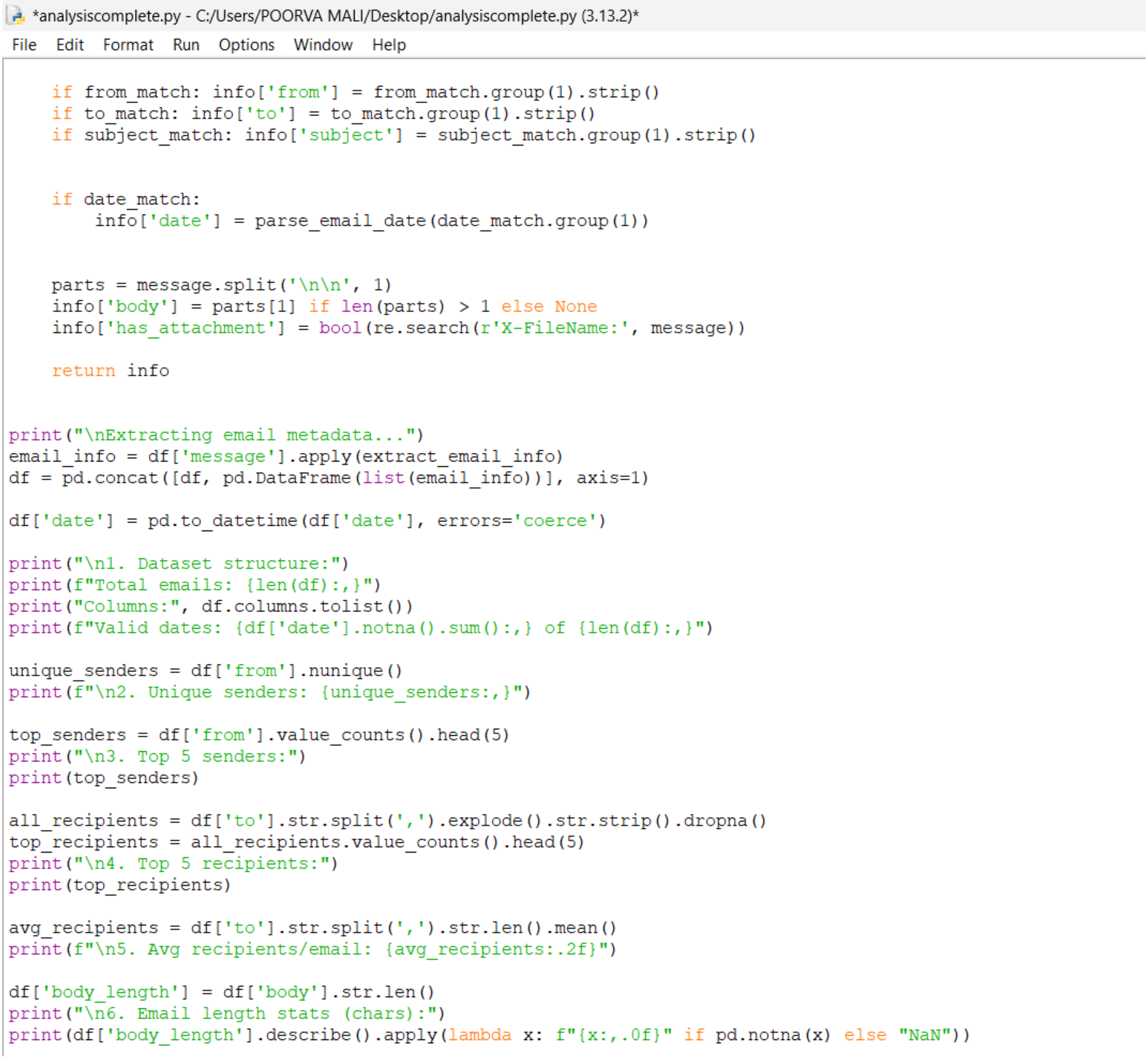


B10 | Message-ID: <20641191.1075855687472.JavaMail.evans@thyme>

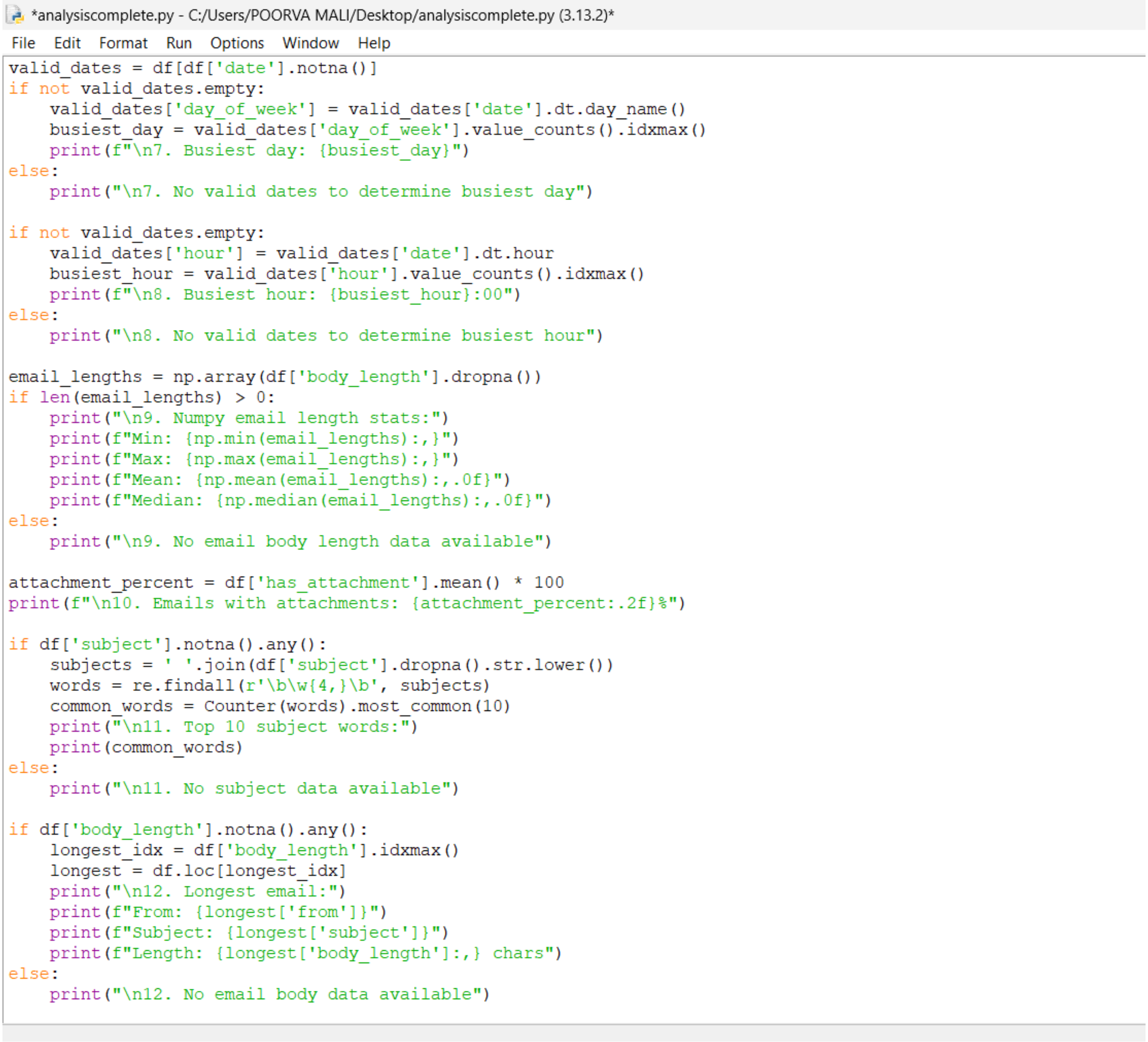
	A	B
1	file	message
2	allen-p/_sent_mail/1.	Message-ID:
3	allen-p/_sent_mail/10.	Message-ID:
4	allen-p/_sent_mail/100.	Message-ID:
5	allen-p/_sent_mail/1000.	Message-ID:
6	allen-p/_sent_mail/1001.	Message-ID:
7	allen-p/_sent_mail/1002.	Message-ID:
8	allen-p/_sent_mail/1003.	Message-ID:
9	allen-p/_sent_mail/1004.	Message-ID:
		Message-ID: <20641191.1075855687472.JavaMail.evans@t hyme> Date: Tue, 17 Oct 2000 02:26:00 -0700 (PDT) From: phillip.allen@enron.com To: mark.scott@enron.com Subject: Re: High Speed Internet Access Mime-Version: 1.0 Content-Type: text/plain; charset=us-ascii Content-Transfer-Encoding: 7bit X-From: Phillip K Allen X-To: Mark Scott X-cc: X-bcc: X-Folder: \Phillip_Allen_Dec2000\Notes Folders\'sent mail X-Origin: Allen-P X-FileName: pallen.nsf 1. login: pallen pw: ke9davis


analysiscomplete.py - C:/Users/POORVA MALI/Desktop/analysiscomplete.py (3.13.2)
File
Edit
Format
Run
Options
Window
Help

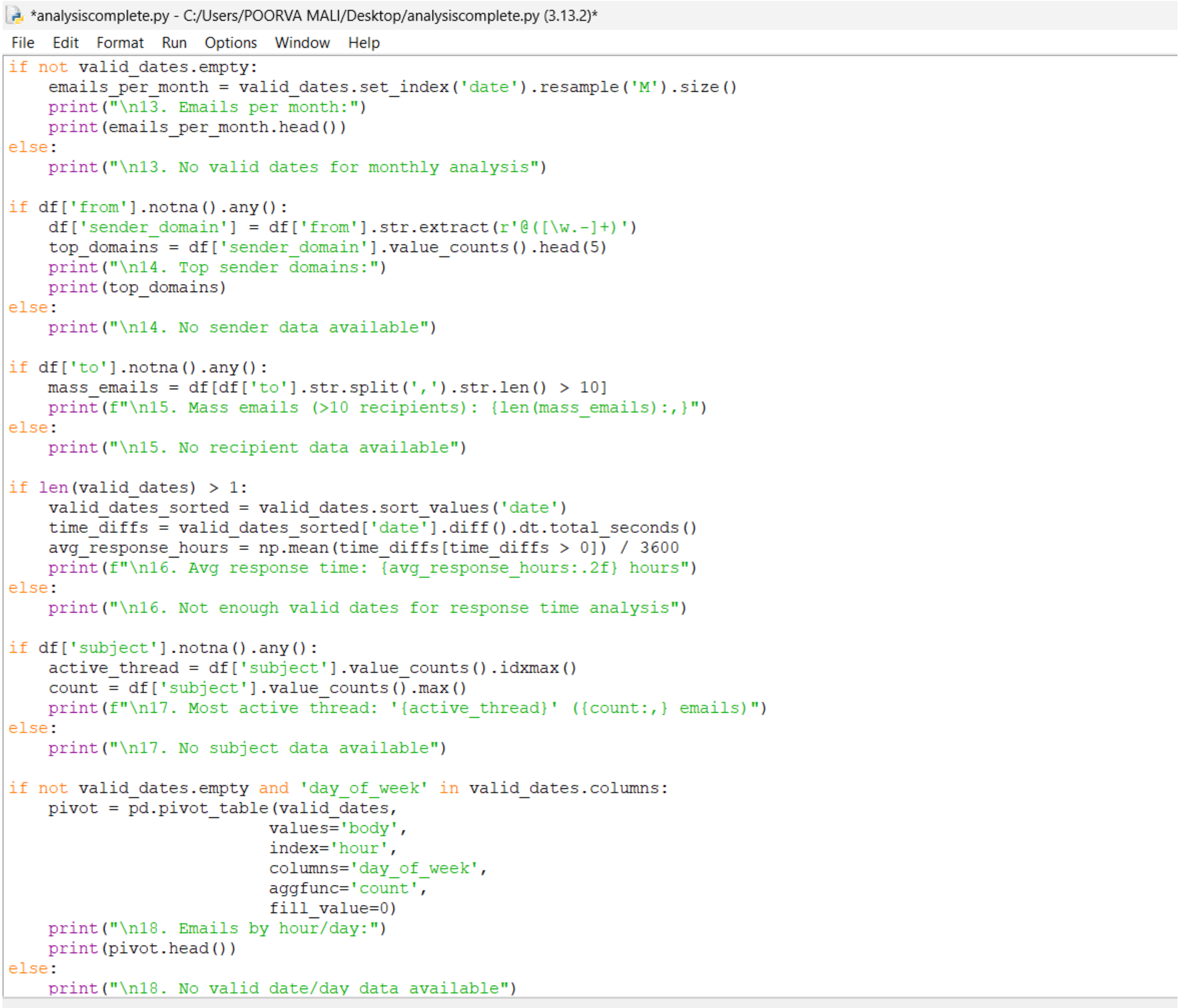
```
import numpy as np
import pandas as pd
import re
from collections import Counter, defaultdict
from datetime import datetime

try:
    df = pd.read_csv(r"C:\Users\POORVA MALI\Downloads\archive(6)\emails.csv")
    print("Dataset loaded successfully!")
except FileNotFoundError:
    print("Error: File 'emails.csv' not found.")
    exit()
def parse_email_date(date_str):
    if pd.isna(date_str):
        return pd.NaT

    date_formats = [
        '%a, %d %b %Y %H:%M:%S %z',  # Tue, 15 May 2001 08:30:00 -0700
        '%a, %d %b %Y %H:%M:%S %Z',   # Tue, 15 May 2001 08:30:00 PST
        '%d %b %Y %H:%M:%S %z',       # 15 May 2001 08:30:00 -0700
        '%m/%d/%Y %H:%M:%S',          # 05/15/2001 08:30:00
        '%Y-%m-%d %H:%M:%S'           # 2001-05-15 08:30:00
    ]

    for fmt in date_formats:
        try:
            return datetime.strptime(str(date_str).strip(), fmt)
        except ValueError:
            continue
    return pd.NaT

def extract_email_info(message):
    info = {
        'from': None,
        'to': None,
        'subject': None,
        'date': None,
        'body': None,
        'has_attachment': False
    }


    from_match = re.search(r'From:\s*(.+)', message)
    to_match = re.search(r'To:\s*(.+)', message)
    subject_match = re.search(r'Subject:\s*(.+)', message)
    date_match = re.search(r'Date:\s*(.+)', message)
```


analysiscomplete.py - C:/Users/POORVA MALI/Desktop/analysiscomplete.py (3.13.2)
File
Edit
Format
Run
Options
Window
Help

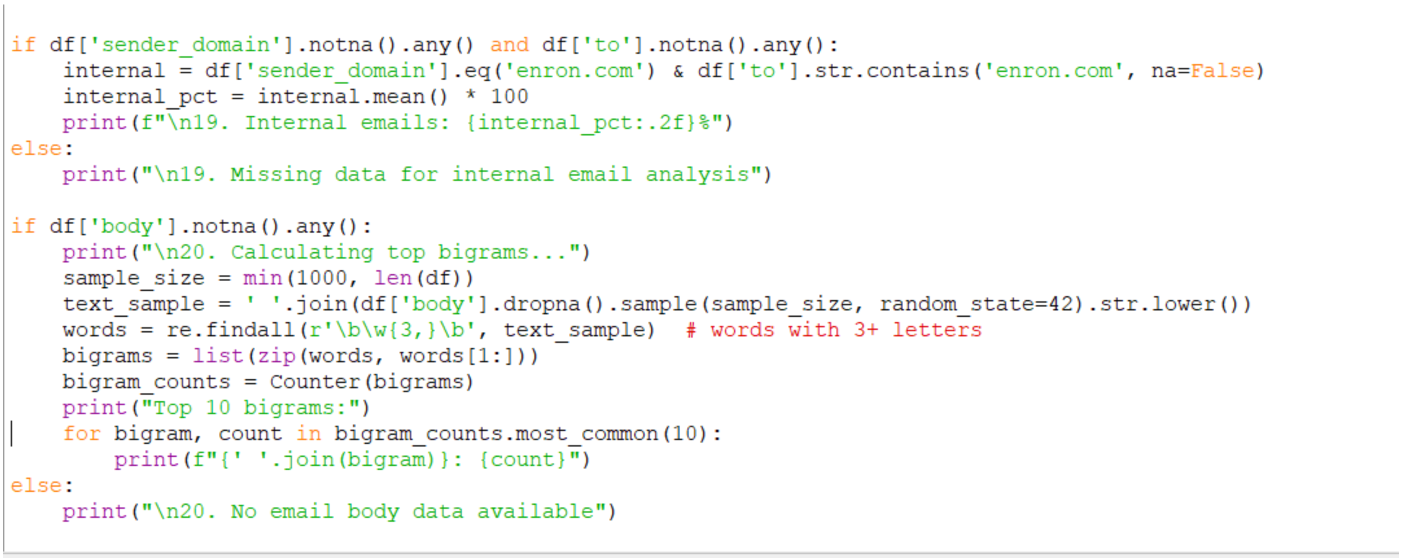
```
    if from_match: info['from'] = from_match.group(1).strip()
    if to_match: info['to'] = to_match.group(1).strip()
    if subject_match: info['subject'] = subject_match.group(1).strip()


    if date_match:
        info['date'] = parse_email_date(date_match.group(1))


    parts = message.split('\n\n', 1)
    info['body'] = parts[1] if len(parts) > 1 else None
    info['has_attachment'] = bool(re.search(r'X-FileName:', message))

    return info


print("\nExtracting email metadata...")
email_info = df['message'].apply(extract_email_info)
df = pd.concat([df, pd.DataFrame(list(email_info))], axis=1)

df['date'] = pd.to_datetime(df['date'], errors='coerce')

print("\n1. Dataset structure:")
print(f"Total emails: {len(df):,}")
print("Columns:", df.columns.tolist())
print(f"Valid dates: {df['date'].notna().sum():,} of {len(df):,}")

unique_senders = df['from'].nunique()
print(f"\n2. Unique senders: {unique_senders:,}")

top_senders = df['from'].value_counts().head(5)
print("\n3. Top 5 senders:")
print(top_senders)

all_recipients = df['to'].str.split(',').explode().str.strip().dropna()
top_recipients = all_recipients.value_counts().head(5)
print("\n4. Top 5 recipients:")
print(top_recipients)

avg_recipients = df['to'].str.split(',').str.len().mean()
print(f"\n5. Avg recipients/email: {avg_recipients:.2f}")

df['body_length'] = df['body'].str.len()
print("\n6. Email length stats (chars):")
print(df['body_length'].describe().apply(lambda x: f"{x:,.0f}" if pd.notna(x) else "NaN"))
```


analysiscomplete.py - C:/Users/POORVA MALI/Desktop/analysiscomplete.py (3.13.2)
File
Edit
Format
Run
Options
Window
Help

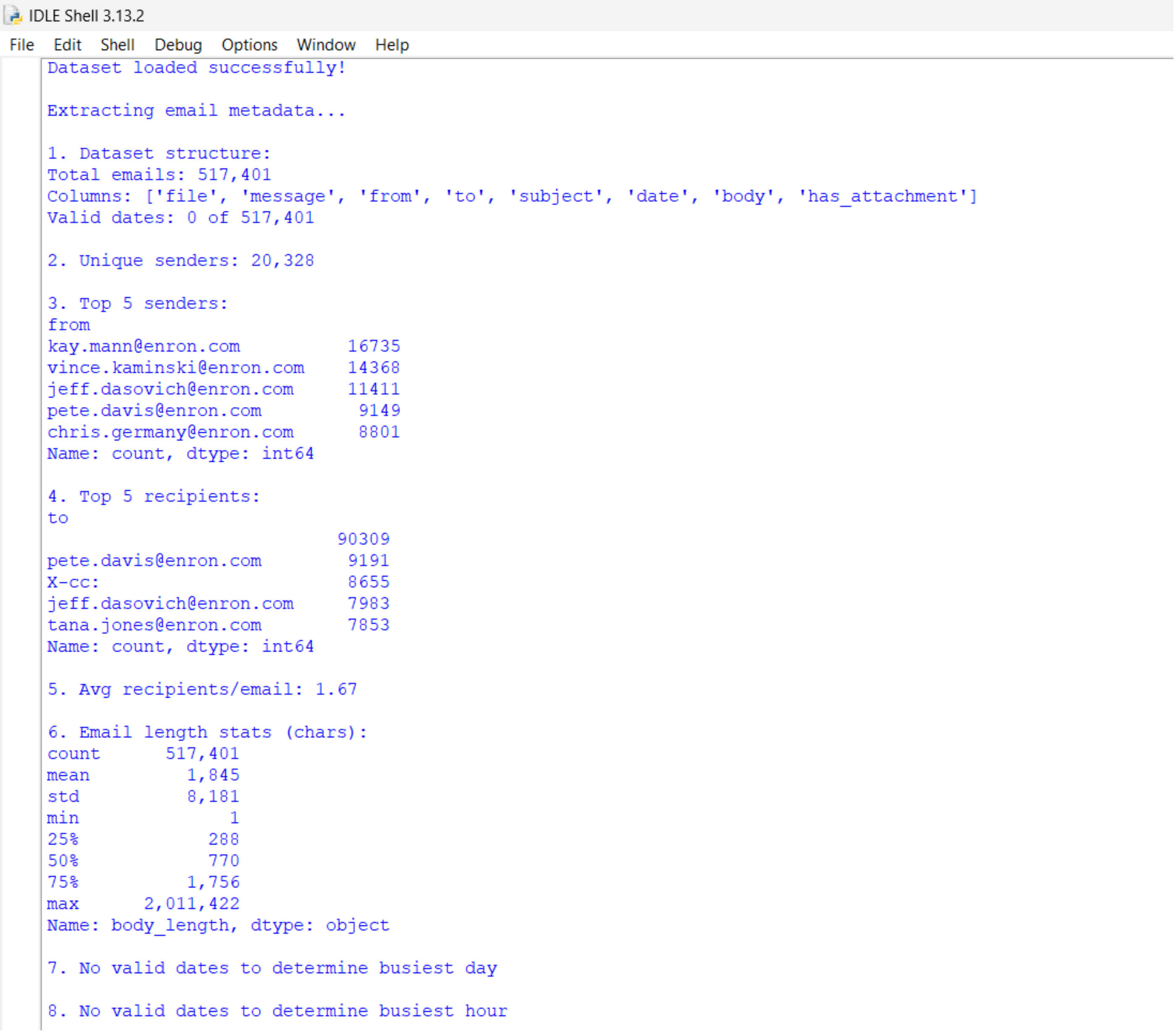
```
valid_dates = df[df['date'].notna()]
if not valid_dates.empty:
    valid_dates['day_of_week'] = valid_dates['date'].dt.day_name()
    busiest_day = valid_dates['day_of_week'].value_counts().idxmax()
    print(f"\n7. Busiest day: {busiest_day}")
else:
    print("\n7. No valid dates to determine busiest day")

if not valid_dates.empty:
    valid_dates['hour'] = valid_dates['date'].dt.hour
    busiest_hour = valid_dates['hour'].value_counts().idxmax()
    print(f"\n8. Busiest hour: {busiest_hour}:00")
else:
    print("\n8. No valid dates to determine busiest hour")

email_lengths = np.array(df['body_length'].dropna())
if len(email_lengths) > 0:
    print("\n9. Numpy email length stats:")
    print(f"Min: {np.min(email_lengths):,}")
    print(f"Max: {np.max(email_lengths):,}")
    print(f"Mean: {np.mean(email_lengths):,.0f}")
    print(f"Median: {np.median(email_lengths):,.0f}")
else:
    print("\n9. No email body length data available")

attachment_percent = df['has_attachment'].mean() * 100
print(f"\n10. Emails with attachments: {attachment_percent:.2f}%")

if df['subject'].notna().any():
    subjects = ' '.join(df['subject'].dropna().str.lower())
    words = re.findall(r'\b\w{4,}\b', subjects)
    common_words = Counter(words).most_common(10)
    print("\n11. Top 10 subject words:")
    print(common_words)
else:
    print("\n11. No subject data available")

if df['body_length'].notna().any():
    longest_idx = df['body_length'].idxmax()
    longest = df.loc[longest_idx]
    print("\n12. Longest email:")
    print(f"From: {longest['from']}")
    print(f"Subject: {longest['subject']}")
    print(f"Length: {longest['body_length']:,} chars")
else:
    print("\n12. No email body data available")
```


analysiscomplete.py - C:/Users/POORVA MALI/Desktop/analysiscomplete.py (3.13.2)
File
Edit
Format
Run
Options
Window
Help

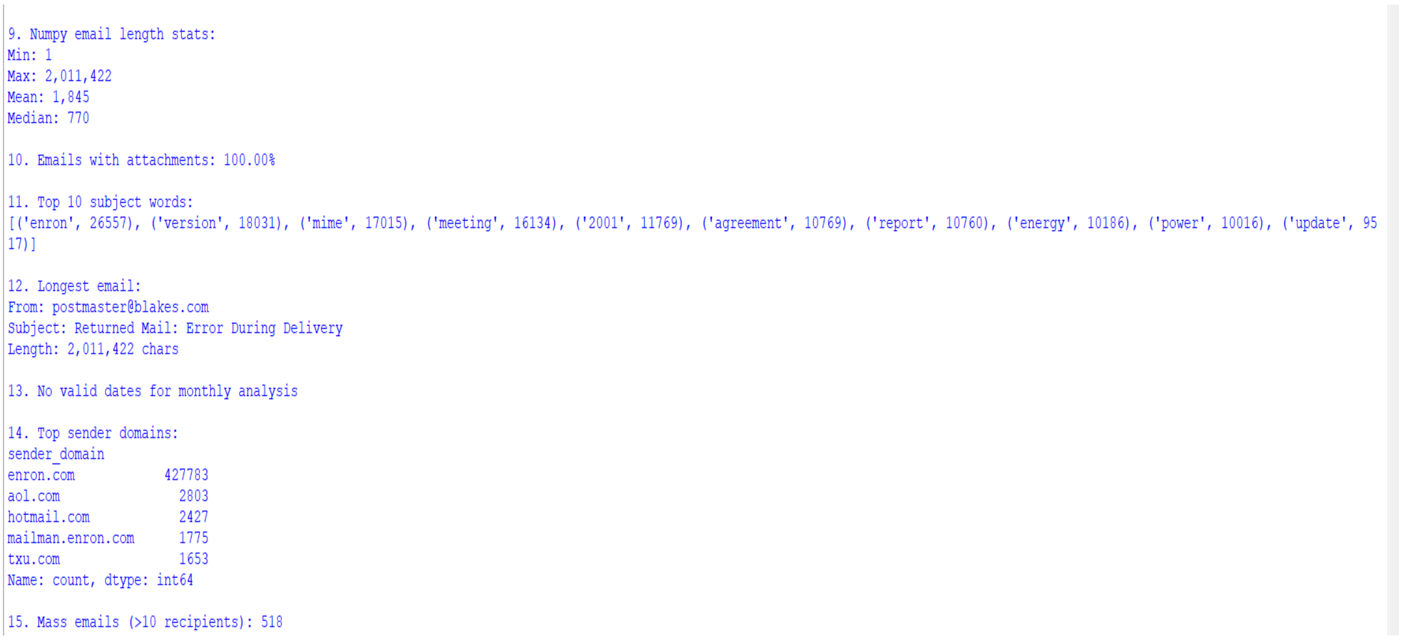
```python
if not valid_dates.empty:
    emails_per_month = valid_dates.set_index('date').resample('M').size()
    print("\n13. Emails per month:")
    print(emails_per_month.head())
else:
    print("\n13. No valid dates for monthly analysis")

if df['from'].notna().any():
    df['sender_domain'] = df['from'].str.extract(r'@([\w.-]+)')
    top_domains = df['sender_domain'].value_counts().head(5)
    print("\n14. Top sender domains:")
    print(top_domains)
else:
    print("\n14. No sender data available")

if df['to'].notna().any():
    mass_emails = df[df['to'].str.split(',').str.len() > 10]
    print(f"\n15. Mass emails (>10 recipients): {len(mass_emails):,}")
else:
    print("\n15. No recipient data available")

if len(valid_dates) > 1:
    valid_dates_sorted = valid_dates.sort_values('date')
    time_diffs = valid_dates_sorted['date'].diff().dt.total_seconds()
    avg_response_hours = np.mean(time_diffs[time_diffs > 0]) / 3600
    print(f"\n16. Avg response time: {avg_response_hours:.2f} hours")
else:
    print("\n16. Not enough valid dates for response time analysis")

if df['subject'].notna().any():
    active_thread = df['subject'].value_counts().idxmax()
    count = df['subject'].value_counts().max()
    print(f"\n17. Most active thread: '{active_thread}' ({count:,} emails)")
else:
    print("\n17. No subject data available")

if not valid_dates.empty and 'day_of_week' in valid_dates.columns:
    pivot = pd.pivot_table(valid_dates,
                           values='body',
                           index='hour',
                           columns='day_of_week',
                           aggfunc='count',
                           fill_value=0)
    print("\n18. Emails by hour/day:")
    print(pivot.head())
else:
    print("\n18. No valid date/day data available")
```



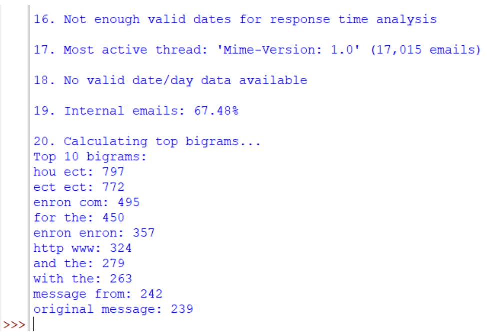
```python
if df['sender_domain'].notna().any() and df['to'].notna().any():
    internal = df['sender_domain'].eq('enron.com') & df['to'].str.contains('enron.com', na=False)
    internal_pct = internal.mean() * 100
    print(f"\n19. Internal emails: {internal_pct:.2f}%")
else:
    print("\n19. Missing data for internal email analysis")

if df['body'].notna().any():
    print("\n20. Calculating top bigrams...")
    sample_size = min(1000, len(df))
    text_sample = ' '.join(df['body'].dropna().sample(sample_size, random_state=42).str.lower())
    words = re.findall(r'\b\w{3,}\b', text_sample)  # words with 3+ letters
    bigrams = list(zip(words, words[1:]))
    bigram_counts = Counter(bigrams)
    print("Top 10 bigrams:")
    for bigram, count in bigram_counts.most_common(10):
        print(f"{' '.join(bigram)}: {count}")
else:
    print("\n20. No email body data available")
```



```
Dataset loaded successfully!

Extracting email metadata...

1. Dataset structure:
Total emails: 517,401
Columns: ['file', 'message', 'from', 'to', 'subject', 'date', 'body', 'has_attachment']
Valid dates: 0 of 517,401

2. Unique senders: 20,328

3. Top 5 senders:
from
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

4. Top 5 recipients:
to
                            90309
pete.davis@enron.com         9191
X-cc:                        8655
jeff.dasovich@enron.com      7983
tana.jones@enron.com         7853
Name: count, dtype: int64

5. Avg recipients/email: 1.67

6. Email length stats (chars):
count      517,401
mean         1,845
std          8,181
min              1
25%            288
50%            770
75%          1,756
max      2,011,422
Name: body_length, dtype: object

7. No valid dates to determine busiest day

8. No valid dates to determine busiest hour
```



```
9. Numpy email length stats:
Min: 1
Max: 2,011,422
Mean: 1,845
Median: 770

10. Emails with attachments: 100.00%

11. Top 10 subject words:
[('enron', 26557), ('version', 18031), ('mime', 17015), ('meeting', 16134), ('2001', 11769), ('agreement', 10769), ('report', 10760), ('energy', 10186), ('power', 10016), ('update', 95
17)]

12. Longest email:
From: postmaster@blakes.com
Subject: Returned Mail: Error During Delivery
Length: 2,011,422 chars

13. No valid dates for monthly analysis

14. Top sender domains:
sender_domain
enron.com           427783
aol.com               2803
hotmail.com           2427
mailman.enron.com     1775
txu.com               1653
Name: count, dtype: int64

15. Mass emails (>10 recipients): 518
```



```
16. Not enough valid dates for response time analysis

17. Most active thread: 'Mime-Version: 1.0' (17,015 emails)

18. No valid date/day data available

19. Internal emails: 67.48%

20. Calculating top bigrams...
Top 10 bigrams:
hou ect: 797
ect ect: 772
enron com: 495
for the: 450
enron enron: 357
http www: 324
and the: 279
with the: 263
message from: 242
original message: 239
>>>
```