Real-Life Problem Statements and Solutions

1. Find how many total emails were exchanged during the Enron scandal period.

```
import pandas as pd



df = pd.read_csv('enron_emails.csv')

total_emails = len(df)

print(total_emails)
```

2. Identify how many unique employees were involved in email communication.

```
unique_senders = df['From'].nunique()


unique_receivers = df['To'].nunique()

total_people = unique_senders + unique_receivers

print(total_people)
```

3. Find the top 5 employees who sent the most number of emails.

```
top5_senders = df['From'].value_counts().head(5)


print(top5_senders)
```

4. Detect the employee who received the maximum number of emails.

```
top_receiver = df['To'].value_counts().idxmax()


print(top_receiver)
```

5. Find the most common subject line keywords during the scandal period.

```
from collections import Counter



subject_words = ' '.join(df['Subject'].dropna()).split()

common_words = Counter(subject_words).most_common(10)

print(common_words)
```

6. Calculate the average length of an email body to understand communication detail.

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

average_words = df['Word_Count'].mean()

print(average_words)
```

7. Find the total number of emails with no subject (possibly internal informal communication).

```
no_subject_emails = df['Subject'].isnull().sum()

print(no_subject_emails)
```

8. Find emails where the subject contains sensitive words like 'confidential' or 'secret'.

```
sensitive_emails = df[df['Subject'].str.contains('confidential|secret', case=False, na=False)]

print(sensitive_emails)
```

9. Find all emails that mention 'stock', 'deal', or 'energy' in the message body.

```
stock_related_emails = df[df['Message'].str.contains('stock|deal|energy', case=False, na=False)]

print(stock_related_emails)
```

10. Calculate how many emails were sent per month to analyze peak communication periods.

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

df['Month'] = df['Date'].dt.month

emails_per_month = df['Month'].value_counts()

print(emails_per_month)
```

11. Identify the month with maximum internal email traffic.

```
peak_month = df['Month'].value_counts().idxmax()

print(peak_month)
```

12. Check how many emails were sent during working hours (9 AM to 6 PM).

```
df['Hour'] = df['Date'].dt.hour
```

```
working_hours_emails = df[(df['Hour'] >= 9) & (df['Hour'] <= 18)]

print(len(working_hours_emails))
```

13. Find emails sent on weekends (Saturday and Sunday).

```
df['DayOfWeek'] = df['Date'].dt.day_name()

weekend_emails = df[df['DayOfWeek'].isin(['Saturday', 'Sunday'])]

print(len(weekend_emails))
```

14. Find the email with the longest message content (could indicate important communication).

```
longest_email_index = df['Word_Count'].idxmax()

print(df.loc[longest_email_index])
```

15. Identify communication between top executives (e.g., emails sent to/from CEO).

```
ceo_emails = df[df['From'].str.contains('ceo', case=False, na=False) | df['To'].str.contains('ceo', case=False,

na=False)]

print(ceo_emails)
```

16. Find how many emails have empty message bodies (could indicate spam or corrupted entries).

```
empty_messages = df[df['Message'].isnull() | (df['Message'].str.strip() == '')]

print(len(empty_messages))
```

17. Calculate total number of missing values in the dataset to estimate data cleaning effort.

```
total_missing = df.isnull().sum().sum()

print(total_missing)
```

18. Analyze top email domains used internally (like @enron.com vs external domains).

```
df['From_Domain'] = df['From'].apply(lambda x: str(x).split('@')[-1])

top_domains = df['From_Domain'].value_counts().head(5)
```

```
print(top_domains)
```

19. Find emails where the subject suggests urgency ('urgent', 'asap', 'important').

```
urgent_emails = df[df['Subject'].str.contains('urgent|asap|important', case=False, na=False)]

print(urgent_emails)
```

20. Check the correlation between number of words in an email and the year it was sent.

```
df['Email_Year'] = df['Date'].dt.year

correlation = df['Email_Year'].corr(df['Word_Count'])

print(correlation)
```

Conclusion

Through the exploration of the Enron email dataset using NumPy and Pandas, we have extracted valuable insights regarding communication patterns, sender/receiver behavior,
content length, time-based activities, and special keywords in emails.
This analysis showcases the practical application of Python libraries in solving real-world
data problems effectively.

NAME: Sanmesh Mane
ROLL NO: ET1-27
PRN NO:202401070038

THANK YOU