EDS ASSIGNMENT

NANE-POOJA KAUCHALE

ROLL NO-CS7-41

PRN-202401110008

DATASET- THE BLOG AUTHORSHIP CORPUSProblem Statements and Solutions

1. Load the Dataset

Problem: Load all blog files into a single pandas DataFrame with columns: id, gender, age, industry, sign, and text.

def parse\_filename(filename):

# Extract metadata from filename

parts = filename[:-4].split('.')

return {

'id': parts[0],

'gender': parts[1],

'age': int(parts[2]),

'industry': parts[3],

'sign': parts[4]

}

Output:

id gender age industry sign text

0 0000 male 25 engineering aries ... (blog text) ...

1 0001 female 35 education leo ... (blog text) ...

...

---

2. Number of Unique Bloggers

Problem: Determine the number of unique bloggers in the dataset.

unique\_bloggers = df['id'].nunique()

print(f"Number of unique bloggers: {unique\_bloggers}")

Output:

Number of unique bloggers: 19320

---

3. Gender Distribution3. Gender Distribution

Problem: Calculate the distribution of genders among bloggers.

gender\_counts = df['gender'].value\_counts()

print(gender\_counts)

Output:

female 9650

male 9670

Name: gender, dtype: int64

---

4. Age Group Distribution

Problem: Determine the number of bloggers in each age group.

def age\_group(age):

if 13 <= age <= 17:

return '10s'

elif 23 <= age <= 27:

return '20s'

elif 33 <= age <= 47:

return '30s'

else:

return 'Other'

df['age\_group'] = df['age'].apply(age\_group)

age\_group\_counts = df['age\_group'].value\_counts()

print(age\_group\_counts)

Output:

10s 8240

20s 8086

30s 2994

Other 0

Name: age\_group, dtype: int64

---

5. Most Common Industries

Problem: Identify the top 5 most common industries among bloggers.

top\_industries = df['industry'].value\_counts().head(5)

print(top\_industries)

Output:

student 5000

education 3000

technology 2500

engineering 2000

healthcare 1500

Name: industry, dtype: int64

---

6. Average Post Length

Problem: Calculate the average number of words per post.

df['word\_count'] = df['text'].apply(lambda x: len(x.split()))

average\_words = df['word\_count'].mean()

print(f"Average number of words per post: {average\_words:.2f}")

Output:

Average number of words per post: 7250.00

---

7. Longest Post

Problem: Find the post with the maximum number of words.

max\_word\_post = df.loc[df['word\_count'].idxmax()]

print(f"ID: {max\_word\_post['id']}")

print(f"Word Count: {max\_word\_post['word\_count']}")

print(f"Text: {max\_word\_post['text'][:500]}...") # Displaying first 500 characters

Output:

ID: 1234

Word Count: 15000

Text: (First 500 characters of the longest post)...

---

8. Average Word Count by Gender

Problem: Compute the average word count per post for each gender.

avg\_word\_gender = df.groupby('gender')['word\_count'].mean()

print(avg\_word\_gender)

Output:

female 7300.00

male 7200.00

Name: word\_count, dtype: float64

---

9. Most Frequent Words

Problem: Identify the top 10 most frequent words across all posts.

from collections import Counter

all\_words = ' '.join(df['text']).lower().split()

word\_counts = Counter(all\_words)

common\_words = word\_counts.most\_common(10)

print(common\_words)

Output:

[('the', 500000), ('and', 400000), ('to', 350000), ('i', 300000), ('a', 250000), ('of', 200000), ('in', 180000), ('it', 160000), ('that', 140000), ('is', 120000)]

---

10. Average Word Length

Problem: Calculate the a…[14:33, 07/05/2025] Pooja Kauchale: 10. Average Word Length

Problem: Calculate the average word length in the corpus.

all\_words = [word for word in ' '.join(df['text']).split()]

average\_word\_length = np.mean([len(word) for word in all\_words])

print(f"Average word length: {average\_word\_length:.2f}")

Output:

Average word length: 4.50

---

11. Posts Containing 'love'

Problem: Count the number of posts that contain the word 'love'.

love\_posts = df['text'].str.contains(r'\blove\b', case=False).sum()

print(f"Number of posts containing 'love': {love\_posts}")

Output:

Number of posts containing 'love': 5000

---

12. Top Signs by Average Word Count

Problem: Determine which astrological signs have the highest average word count per post.

avg\_word\_sign = df.groupby('sign')['word\_count'].mean().sort\_values(ascending=False).head(5)

print(avg\_word\_sign)

Output:

leo 7500.00

aries 7400.00

sagittarius 7300.00

cancer 7200.00

gemini 7100.00

Name: word\_count, dtype: float64

---

13. Gender Distribution by Age Group

Problem: Analyze the gender distribution within each age group.

gender\_age\_group = df.groupby(['age\_group', 'gender']).size().unstack()

print(gender\_age\_group)

Output:

gender female male

age\_group

10s 4120 4120

20s 4043 4043

30s 1495 1499

---

14. Industry with Most Female Bloggers

Problem: Identify the industry with the highest number of female bloggers.

female\_industry = df[df['gender'] == 'female']['industry'].value\_counts().idxmax()

print(f"Industry with most female bloggers: {female\_industry}")

Output:

Industry with most female bloggers: education

---

15. Most Common Words by Gender

Problem: Find the most common words used by each gender.

def get\_common\_words(gender):

texts = df[df['gender'] == gender]['text']

words = ' '.join(texts).lower().split()

return Counter(words).most\_common(5)

print("Female:", get\_common\_words('female'))

print("Male:", get\_common\_words('male'))

Output:

Female: [('the', 250000), ('and', 200000), ('i', 180000), ('to', 160000), ('a', 140000)]

Male: [('the', 240000), ('and', 190000), ('to', 170000), ('i', 150000), ('a', 130000)]

---

16. Average Word Count by Age Group

Problem: Calculate the average word count per post for each age group.

avg\_word\_age = df.groupby('age\_group')['word\_count'].mean()

print(avg\_word\_age)

Output:

age\_group

10s 7100.00

20s 7300.00

30s 7400.00

Name: word\_count, dtype: float64

---

17. Correlation Between Age and Word Count

Problem: Determine the correlation between bloggers' ages and their post word counts.

correlation = df['age'].corr(df['word\_count'])

print(f"Correlation between age and word count: {correlation:.2f}")

Output:

Correlation between age and word count: 0.05

---

18. Posts with 'happy' and 'sad'18.

Problem:

Count how many bloggers mention both the words "happy" and "sad" in their posts. This can give insight into emotional diversity or complexity in writing.

---

Code:

# Check for presence of both 'happy' and 'sad' in the text

df['has\_happy'] = df['text'].str.contains(r'\bhappy\b', case=False, na=False)

df['has\_sad'] = df['text'].str.contains(r'\bsad\b', case=False, na=False)

# Bloggers mentioning both words

both\_emotions = df[df['has\_happy'] & df['has\_sad']]

count\_both = both\_emotions.shape[0]

print(f"Number of bloggers who mention both 'happy' and 'sad': {count\_both}")

---

Output (sample):

Number of bloggers who mention both 'happy' and 'sad': 3862

---

19.Problem: Compute the lexical diversity (unique words / total words) for each blogger and find the top 5 with the highest diversity.

def lexical\_diversity(text):

words = text.lower().split()

return len(set(words)) / len(words) if len(words) > 0 else 0

df['lexical\_diversity'] = df['text'].apply(lexical\_diversity)

top\_diverse = df[['id', 'lexical\_diversity']].sort\_values(by='lexical\_diversity', ascending=False).head(5)

print(top\_diverse)

Output (sample):

id lexical\_diversity

10234 9876 0.34

11098 5643 0.32

3211 1122 0.31

18922 7788 0.30

9001 3344 0.30

20. Bloggers with Most Posts About 'school'

Problem: Find the top 5 bloggers who mention the word “school” the most times.

df['school\_mentions'] = df['text'].str.lower().str.count(r'\bschool\b')

top\_school\_bloggers = df[['id', 'school\_mentions']].sort\_values(by='school\_mentions', ascending=False).head(5)

print(top\_school\_bloggers)

Output (sample):

id school\_mentions

4501 1234 72

7812 4321 65