** Reading the Data from Database**

In [ ]:
```
import sqlite3
import pandas as pd
```

Step 1 - Reading the Tables from Database file ¶

In [ ]:
```
cd C:\Users\DELL\OneDrive\Desktop\data-20240407T052706Z-001\data
```

In [ ]:
```
# Read the code below and write your observation in the next cell

conn = sqlite3.connect("eng_subtitles_database.db")
cursor = conn.cursor()
cursor.execute("SELECT name FROM sqlite_master WHERE type='table'")
print(cursor.fetchall())
```

In the above cell, I am able to read the table inside the database. As mentioned earlier, table name is `zipfiles` . We also know from README.txt that this table contains three columns: 'num', 'name' and 'content'.

Step 2 - Reading the columns of Table

In [ ]:
```
cursor.execute("PRAGMA table_info('zipfiles')")
cols = cursor.fetchall()
for col in cols:
    print(col[1])
```

The above code helps in checking the column names in the database table.

Let's now use `SELECT * FROM zipfiles` to read all the data into a `df` variable.

Step 3 - Loading the Database Table inside a Pandas DataFrame

In [ ]:
```
df = pd.read_sql_query("""SELECT * FROM zipfiles""", conn)
df.head()
```

In [ ]:
```
df.info()
```

Looks like the `content` column donot contain the subtitles text. Instead as mentioned in README.txt, it might be latin-1 encoded.

Step 4 - Printing `content` of 0th Row

In [ ]:
```
b_data = df.iloc[0, 2]

# here 2 represent the index of content column
# 0 represents the row number
```

In [ ]:
```
print(b_data)
```

From the content, it appears to start with the bytes "PK\x03.....", which suggests that it might be a ZIP archive file. How do I know it? Experience! I have worked with something similar earlier.

Step 5 - Unzipping the content of 385th row and decoding using `latin-1`

In [ ]:
```
import zipfile
import io

# Assuming 'content' is the binary data from your database
binary_data = df.iloc[385, 2]

# Decompress the binary data using the zipfile module
with io.BytesIO(binary_data) as f:
    with zipfile.ZipFile(f, 'r') as zip_file:
        # Reading only one file in the ZIP archive
        subtitle_content = zip_file.read(zip_file.namelist()[0])

# Now 'subtitle_content' should contain the extracted subtitle content
print(subtitle_content.decode('latin-1'))  # Assuming the content is latin-1 e
```

Look's like it worked.

Step 6 - Applying the above Function on the Entire Data

In [ ]:
```
import zipfile
import io

count = 0

def decode_method(binary_data):
    global count
    # Decompress the binary data using the zipfile module
    # print(count, end=" ")
    count += 1
    with io.BytesIO(binary_data) as f:
        with zipfile.ZipFile(f, 'r') as zip_file:
            # Assuming there's only one file in the ZIP archive
            subtitle_content = zip_file.read(zip_file.namelist()[0])

    # Now 'subtitle_content' should contain the extracted subtitle content
    return subtitle_content.decode('latin-1')  # Assuming the content is UTF-8
```

In [ ]:
```
df['file_content'] = df['content'].apply(decode_method)

df.head()
```

In [ ]:
```
df.info()
```

In [ ]:
```
df.tail()
```

In [ ]:
```
df.drop(columns="content",inplace=True)
```

In [ ]:
```
df
```

In [ ]:
```
df.head()
```

In [ ]:
```
data=df
```

In [ ]:
```
data.head()
```

In [ ]:
```
data['name']=data['name'].str.replace('.eng.1cd', '')
```

In [ ]:
```
data.head()
```

In [ ]:
```
import nltk
nltk.download('stopwords')
```

In [ ]:
```
nltk.download('wordnet')
```

In [ ]:
```
nltk.download('punkt')
```

In [ ]:
```
import string
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from bs4 import BeautifulSoup
import unicodedata
from nltk.stem import WordNetLemmatizer
import re

def clean_text(sentence):
    # Remove timestamps
    clean_sentence = re.sub(r'\d+:\d+:\d+,?\d* --> \d+:\d+:\d+,?\d*', '', sent

    # Remove special characters and extra spaces
    clean_sentence = re.sub(r'[^a-zA-Z0-9\s]', '', clean_sentence)

    # Convert text to lowercase
    clean_sentence = clean_sentence.lower()

    # Remove leading and trailing whitespace
    clean_sentence = clean_sentence.strip()

    # Removing HTML tags
    clean_sentence = BeautifulSoup(clean_sentence, 'html.parser').get_text()

    # Removing URLs
    clean_sentence = ' '.join([word for word in clean_sentence.split() if not

    # Removing punctuation
    clean_sentence = ''.join([char for char in clean_sentence if char not in s

    # Removing numbers
    clean_sentence = ''.join([i for i in clean_sentence if not i.isdigit()])

    # Removing stopwords
    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(clean_sentence)
    clean_sentence = ' '.join([word for word in word_tokens if word.lower() no

    # Handling special characters
    clean_sentence = unicodedata.normalize('NFKD', clean_sentence).encode('asc

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = word_tokenize(clean_sentence)
    clean_sentence = ' '.join([lemmatizer.lemmatize(word) for word in tokens])

    return clean_sentence
```

In [ ]:
```
data.head()
```

In [ ]:
```
data.to_csv("eng_subtitles_database.csv.csv", index=False, escapechar='\\')
```

In [ ]:
```
data['clean_file_content']=data['file_content'].apply(clean_text)
```

In [ ]:
```
data
```

In [ ]:
```
data.drop(columns='file_content',inplace=True)
```

In [ ]:
```
import pandas as pd
import os

# Example directory path
directory = 'D:\internship\Data science\TASKS\search engine\data'

# Check if the directory has write permission
if not os.access(directory, os.W_OK):
    print(f"No write permission in directory: {directory}")
    # You may choose to exit the script or handle this situation differently
else:
    # Assuming 'data' is your DataFrame
    try:
        data.to_csv(os.path.join(directory, 'eng_subtitles_database.csv'), ind
        print("Data successfully saved to CSV.")
    except PermissionError as e:
        print(f"PermissionError: {e}")
```

In [ ]:
```
#END
```