In [2]:
```python
import sqlite3
import pandas as pd
```

Step 1 - Reading the Tables from Database file

In [3]:
```python
import sqlite3

# Connect to the SQLite database
conn = sqlite3.connect('eng_subtitles_database.db')
cursor = conn.cursor()

# Execute the query to fetch table names
cursor.execute("SELECT name FROM sqlite_master WHERE type='table'")

# Fetch all table names from the database
table_names = cursor.fetchall()

# Print the table names
print(table_names)
```

```
[('zipfiles',)]
```

Step 2 - Reading the columns of Table

In [4]:
```python
cursor.execute("PRAGMA table_info('zipfiles')")
cols = cursor.fetchall()
for col in cols:
    print(col[1])
```

```
num
name
content
```

Step 3 - Loading the Database Table inside a Pandas DataFrame

In [5]:
```python
df = pd.read_sql_query("""SELECT * FROM zipfiles""", conn)
df.head()
```

Out[5]:

	num	name	content
0	9180533	the.message.(1976).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x1c\xa9\x...
1	9180583	here.comes.the.grump.s01.e09.joltin.jack.in.bo...	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x17\xb9\x...
2	9180592	yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00L\xb9\x99V...
3	9180594	yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00U\xa9\x99V...
4	9180600	broker.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x001\xa9\x99V...

In [6]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 82498 entries, 0 to 82497
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   num      82498 non-null  int64
 1   name     82498 non-null  object
 2   content  82498 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.9+ MB
```

Unzipping the content of row and decoding using latin-1

In [5]:
```python
import random
import zipfile
import io

# Function to decode binary data from ZIP archives
def decode_method(binary_data):
    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 encoded text

# Calculate the number of rows to select
num_rows_to_select = int(0.3 * len(df))

# Randomly select rows without replacement
selected_rows_indices = random.sample(range(len(df)), num_rows_to_select)

# Create a new column to store the selected rows
df['file content'] = ''

# Iterate over selected rows and populate the "file content" column
for idx in selected_rows_indices:
    df.at[idx, 'file content'] = decode_method(df.at[idx, 'content'])

# Now df contains the original data along with the decoded content for approximately 30% of the rows
```

In [7]:
```python
import random
import zipfile
import io

# Function to decode binary data from ZIP archives
def decode_method(binary_data):
    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 encoded text

# Get unique file names from the DataFrame
unique_file_names = df['name'].unique().tolist()

# Calculate the number of files to select
num_files_to_select = int(0.3 * len(unique_file_names))

# Randomly select files without replacement
selected_file_names = random.sample(unique_file_names, num_files_to_select)

# Create a new column to store the selected files' content
df['file content'] = ''

# Iterate over selected files, decode their content, and populate the "file content" column
for file_name in selected_file_names:
    file_rows = df[df['name'] == file_name]
    for _, row in file_rows.iterrows():
        decoded_content = decode_method(row['content'])
        df.at[row.name, 'file content'] = decoded_content

# Now df contains the original data along with the decoded content for approximately 30% of the files
```

In [8]:
```python
df
```

Out[8]:

	num	name	content	file content
0	9180533	the.message.(1976).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x1c\xa9\x...	
1	9180583	here.comes.the.grump.s01.e09.joltin.jack.in.bo...	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x17\xb9\x...	1\r\n00:00:29,359 --> 00:00:32,048\r\nAh! Ther...
2	9180592	yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00L\xb9\x99V...	1\r\n00:00:53,200 --> 00:00:56,030\r\n<i>Yumi'...
3	9180594	yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00U\xa9\x99V...	1\r\n00:00:06,000 --> 00:00:12,074\r\nWatch an...
4	9180600	broker.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x001\xa9\x99V...	
...	...	...	...	...
82493	9521935	the.prophets.game.(2000).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\xb8\xa6\x...	
82494	9521937	west.beirut.(1998).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x13\x97\x...	1\r\n00:00:06,000 --> 00:00:12,074\r\napi.Open...
82495	9521938	frankenstein.the.true.story.(1973).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00$\x97\x9aV...	
82496	9521940	frankenstein.the.true.story.(1973).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x00\x97\x...	
82497	9521941	zombie.island.massacre.(1984).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00,\x97\x9aV...	

82498 rows × 4 columns

In [9]:
```python
import pandas as pd
import zipfile
import io
import random

# Define a function to extract content from ZIP files
def extract_zip_content(binary_data):
    try:
        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
                file_content = zip_file.read(zip_file.namelist()[0])
        return file_content.decode('latin-1')  # Assuming the content is UTF-8 encoded text
    except Exception as e:
        print("Error:", e)
        return None

# Define a function to randomly select 30% of files and extract their content
def randomly_select_and_extract(df):
    # Get unique file names from the DataFrame
    unique_file_names = df['name'].unique().tolist()

    # Calculate the number of files to select
    num_files_to_select = int(0.3 * len(unique_file_names))

    # Randomly select files without replacement
    selected_file_names = random.sample(unique_file_names, num_files_to_select)

    # Create a new column to store the selected files' content
    df['file content'] = ''

    # Iterate over selected files, decode their content, and populate the "file content" column
    for file_name in selected_file_names:
        file_rows = df[df['name'] == file_name]
        for _, row in file_rows.iterrows():
            decoded_content = extract_zip_content(row['content'])
            df.at[row.name, 'file content'] = decoded_content

    return df

# Read the data into a pandas DataFrame (replace "conn" with your database connection)
df = pd.read_sql_query("""SELECT * FROM zipfiles""", conn)

# Apply the random selection and extraction function
df_with_random_selection = randomly_select_and_extract(df)

# Now df_with_random_selection contains the original data along with the decoded content for approximately 30% of the files
```

In [10]:
```python
df_with_random_selection
```

Out[10]:

	num	name	content	file content
0	9180533	the.message.(1976).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x1c\xa9\x...	1\r\n00:00:06,000 --> 00:00:12,074\r\nWatch an...
1	9180583	here.comes.the.grump.s01.e09.joltin.jack.in.bo...	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x17\xb9\x...	
2	9180592	yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00L\xb9\x99V...	1\r\n00:00:53,200 --> 00:00:56,030\r\n<i>Yumi'...
3	9180594	yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00U\xa9\x99V...	
4	9180600	broker.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x001\xa9\x99V...	
...	...	...	...	...
82493	9521935	the.prophets.game.(2000).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\xb8\xa6\x...	
82494	9521937	west.beirut.(1998).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x13\x97\x...	
82495	9521938	frankenstein.the.true.story.(1973).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00$\x97\x9aV...	
82496	9521940	frankenstein.the.true.story.(1973).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x00\x97\x...	
82497	9521941	zombie.island.massacre.(1984).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00,\x97\x9aV...	1\r\n00:00:01,919 --> 00:00:03,253\r\n(Sharp w...

82498 rows × 4 columns

In [11]:
```python
df_with_random_selection['content']
```

Out[11]:
```
0        b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x1c\xa9\x...
1        b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x17\xb9\x...
2        b'PK\x03\x04\x14\x00\x00\x00\x08\x00L\xb9\x99V...
3        b'PK\x03\x04\x14\x00\x00\x00\x08\x00U\xa9\x99V...
4        b'PK\x03\x04\x14\x00\x00\x00\x08\x001\xa9\x99V...
                               ...
82493    b'PK\x03\x04\x14\x00\x00\x00\x08\x00\xb8\xa6\x...
82494    b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x13\x97\x...
82495    b'PK\x03\x04\x14\x00\x00\x00\x08\x00$\x97\x9aV...
82496    b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x00\x97\x...
82497    b'PK\x03\x04\x14\x00\x00\x00\x08\x00,\x97\x9aV...
Name: content, Length: 82498, dtype: object
```

Encoding the content column

In [12]:
```python
# Define a function to decode file content from bytes to string
def decode_content(content):
    try:
        decoded_content = content.decode('latin-1').encode('utf-8', errors='ignore').decode('utf-8')
        return decoded_content
    except Exception as e:
        print("Error decoding content:", e)
        return None

# Decode the content in the 'content' column and create a new column for the decoded content
df_with_random_selection['decoded_content'] = df_with_random_selection['content'].apply(decode_content)

# Display the DataFrame with the decoded content column
print(df_with_random_selection.head())
```

```
       num                                             name  \
0  9180533                          the.message.(1976).eng.1cd
1  9180583  here.comes.the.grump.s01.e09.joltin.jack.in.bo...
2  9180592    yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd
3  9180594    yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd
4  9180600                               broker.(2022).eng.1cd

                                             content  \
0  b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x1c\xa9\x...
1  b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x17\xb9\x...
2  b'PK\x03\x04\x14\x00\x00\x00\x08\x00L\xb9\x99V...
3  b'PK\x03\x04\x14\x00\x00\x00\x08\x00U\xa9\x99V...
4  b'PK\x03\x04\x14\x00\x00\x00\x08\x001\xa9\x99V...

                                        file content  \
0  1\r\n00:00:06,000 --> 00:00:12,074\r\nWatch an...
1
2  1\r\n00:00:53,200 --> 00:00:56,030\r\n<i>Yumi'...
3
4

                                     decoded_content
0  PK©Vxð;The.Message.1976...
1  PK¹Vo°ÂgZ$8Here Comes the G...
2  PKL¹V¿[|èN$á6Yumi's.Cells.S02...
3  PKU©V'R\ãYY6Yumi's.Cells.S02...
4  PK1©Vþw¨6Ä+Broker 2022 720p...
```

In [13]:
```python
df_with_random_selection
```

Out[13]:

	num	name	content	file content	decoded_content
0	9180533	the.message.(1976).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x1c\xa9\x...	1\r\n00:00:06,000 --> 00:00:12,074\r\nWatch an...	PK©Vxð;The.Message.1976...
1	9180583	here.comes.the.grump.s01.e09.joltin.jack.in.bo...	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x17\xb9\x...		PK¹Vo°ÂgZ$8Here Comes the G...
2	9180592	yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00L\xb9\x99V...	1\r\n00:00:53,200 --> 00:00:56,030\r\n<i>Yumi'...	PKL¹V¿[|èN$á6Yumi's.Cells.S02...
3	9180594	yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00U\xa9\x99V...		PKU©V'R\ãYY6Yumi's.Cells.S02...
4	9180600	broker.(2022).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x001\xa9\x99V...		PK1©Vþw¨6Ä+Broker 2022 720p...
...	...	...	...	...	...
82493	9521935	the.prophets.game.(2000).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\xb8\xa6\x...		PK¸¦V\njÇ±øìIThe.Prophet's.G...
82494	9521937	west.beirut.(1998).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x13\x97\x...		PKV¹Ù"ðvT!West.Beirut.WEBR...
82495	9521938	frankenstein.the.true.story.(1973).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00$\x97\x9aV...		PK$V áäª?Frankenstein.The...
82496	9521940	frankenstein.the.true.story.(1973).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00\x00\x97\x...		PKV,yèXÁß<Frankenstein.The...
82497	9521941	zombie.island.massacre.(1984).eng.1cd	b'PK\x03\x04\x14\x00\x00\x00\x08\x00,\x97\x9aV...	1\r\n00:00:01,919 --> 00:00:03,253\r\n(Sharp w...	PK,V#øI?U!:Zombie.Island.Ma...

82498 rows × 5 columns

In [14]:
```python
df_with_random_selection['decoded_content']
```

Out[14]:
```
0        PK
©Vxð;The.Message.1976...
1        PK¹Vo°ÂgZ$8Here Comes the G...
2        PKL¹V¿[|èN$á6Yumi's.Cells.S02...
3        PKU©V'R\ãYY6Yumi's.Cells.S02...
4        PK1©Vþ
w¨6Ä+Broker 2022 720p...
                               ...
82493    PK,¦V\nj
Ç±øìIThe.Prophet's.G...
82494    PKV¹Ù"ðvT!West.Beirut.WEBR...
82495    PK$V á
äª?Frankenstein.The...
82496    PKV,yèXÁß
<Frankenstein.The...
82497    PK,V#øI?U!:Zombie.Island.Ma...
Name: decoded_content, Length: 82498, dtype: object
```

In [18]:
```python
# Define a function to decode file content from bytes to string
def decode_content(content):
    try:
        decoded_content = content.decode('latin-1').encode('utf-8', errors='ignore').decode('utf-8')
        return decoded_content
    except Exception as e:
        print("Error decoding content:", e)
        return None

# Decode the content in the 'content' column and create a new column for the decoded content
df_with_random_selection['decoded_content'] = df_with_random_selection['content'].apply(decode_content)

# Convert binary data in 'decoded_content' column to readable text
df_with_random_selection['decoded_content'] = df_with_random_selection['decoded_content'].apply(lambda x: x if isinstance(x, str) else '')

# Display the DataFrame with the decoded content column
print(df_with_random_selection['decoded_content'].head())
```

```
0    PK©Vxð;The.Message.1976...
1    PK¹Vo°ÂgZ$8Here Comes the G...
2    PKL¹V¿[|èN$á6Yumi's.Cells.S02...
3    PKU©V'R\ãYY6Yumi's.Cells.S02...
4    PK1©Vþw¨6Ä+Broker 2022 720p...
Name: decoded_content, dtype: object
```

In [15]:
```python
# Define a function to format the decoded content with subtitle numbers
def format_decoded_content(row):
    subtitle_number = row['num']
    decoded_content = row['decoded_content']
    formatted_content = f"Subtitle {subtitle_number} ({row['name']}):\n\n{decoded_content}"
    return formatted_content

# Apply the formatting function to create a new column with formatted content
df_with_random_selection['formatted_content'] = df_with_random_selection.apply(format_decoded_content, axis=1)

# Display the DataFrame with the formatted content
print(df_with_random_selection['formatted_content'].head())
```

```
0    Subtitle 9180533 (the.message.(1976).eng.1cd):...
1    Subtitle 9180583 (here.comes.the.grump.s01.e09...
2    Subtitle 9180592 (yumis.cells.s02.e13.episode....
3    Subtitle 9180594 (yumis.cells.s02.e14.episode....
4    Subtitle 9180600 (broker.(2022).eng.1cd):\n\nP...
Name: formatted_content, dtype: object
```

In [16]:
```python
df_with_random_selection['formatted_content']
```

Out[16]:
```
0        Subtitle 9180533 (the.message.(1976).eng.1cd):...
1        Subtitle 9180583 (here.comes.the.grump.s01.e09...
2        Subtitle 9180592 (yumis.cells.s02.e13.episode....
3        Subtitle 9180594 (yumis.cells.s02.e14.episode....
4        Subtitle 9180600 (broker.(2022).eng.1cd):\n\nP...
                               ...
82493    Subtitle 9521935 (the.prophets.game.(2000).eng...
82494    Subtitle 9521937 (west.beirut.(1998).eng.1cd):...
82495    Subtitle 9521938 (frankenstein.the.true.story....
82496    Subtitle 9521940 (frankenstein.the.true.story....
82497    Subtitle 9521941 (zombie.island.massacre.(1984...
Name: formatted_content, Length: 82498, dtype: object
```

Extractind data which present inside the content column

In [17]:
```python
import re

# Function to extract content inside subtitles
def extract_content_inside(formatted_content):
    # Define a pattern to extract content inside the subtitles
    pattern = r'\d+\n(.*?)(?=\d+\n|$)'
    # Use regular expression to find matches
    matches = re.findall(pattern, formatted_content, flags=re.DOTALL)
    # Join the matches with newline separator
    content_inside = '\n'.join(matches)
    return content_inside

# Apply the function to extract content inside each row
df_with_random_selection['content_inside'] = df_with_random_selection['formatted_content'].apply(extract_content_inside)

# Display the DataFrame with the newly added content_inside column
print(df_with_random_selection[['name', 'content_inside']].head())
```

```
                                                name  \
0                         the.message.(1976).eng.1cd
1  here.comes.the.grump.s01.e09.joltin.jack.in.bo...
2    yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd
3    yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd
4                              broker.(2022).eng.1cd

                                      content_inside
0  OçÕÌù¤r¹¶ºÍ¦s´c%w¤ðÔ¿ÿ.Ü¹zøoÏÂßÝOòå...
1
2  Á¢XÔ}±ëdÛñÐ¯zdA±4rjíÉÊEXÆäÕcIs5è<çMÜ...
3  ÐJ÷¼ìÃhìçöÙx¶¶ÙÃÜ¼f¦²-XÐ}V³ÄÄót_...
4    +6®vD+±UÝC>|ÑA  ëQ`d)ÂØ>¬ÊF\rX¨¯6¾>Ò¸á^...
```

In [18]:
```python
df_with_random_selection['content_inside']
```

Out[18]:
```
0
OçÕÌù¤r¹¶ºÍ¦s´c%w¤ðÔ¿ÿ.Ü¹z
øoÏÂßÝOòå...
1
2        Á
¢XÔ}±ëdÛñÐ¯zdA±4rjíÉÊEXÆäÕcIs5è<çMÜ...
3        ÐJ÷¼ìÃhìçöÙx¶¶ÙÃÜ¼f¦²-XÐ}V³ÄÄót_...
4          +6®
vD+±UÝC>|Ñ
A  ëQ`d)ÂØ>¬ÊF\rX¨¯6¾>Ò¸á^...
                               ...
82493    X1¡Í{^¸<æÍÊ
íäz  ÐaènÂÑ$k  ñ~ã«f<a  ~¿ÊÈ.ê5...
82494    Ò ß'°ág§þiÝ8ÁæÂd*ÍKItÃÏÕoXE½þ±¿íð\°
...
82495    \rwxíÓ¢\r!îd2qen=óe1ÀÂiîãûñµÀ¸K³õÞUdF...
82496    ¤í×È7!rÕC_Í,÷w*ªL-´³¬ÞlÂ§RDsUÎMÆÔ...
82497    ]ÊægR(zÉ®HSÏ3¤ºç_s#ÍEÇh
Í¤%zqE|u³$X...
Name: content_inside, Length: 82498, dtype: object
```

In [13]:
```python
import pandas as pd
import codecs

# Assuming df_with_random_selection is your DataFrame containing the 'content_inside' column

# Define a function to decode the content inside the 'content_inside' column
def decode_content(content):
    try:
        decoded_content = codecs.decode(content.encode('utf-8'), 'utf-8')
        return decoded_content
    except Exception as e:
        print("Error decoding content:", e)
        return None

# Apply the decode_content function to decode the 'content_inside' column
df_with_random_selection['decoded_content_inside'] = df_with_random_selection['content_inside'].apply(decode_content)

# Display the DataFrame with the newly added 'decoded_content_inside' column
print(df_with_random_selection[['name', 'decoded_content_inside']].head())
```

```
                                                name  \
0                         the.message.(1976).eng.1cd
1  here.comes.the.grump.s01.e09.joltin.jack.in.bo...
2    yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd
3    yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd
4                              broker.(2022).eng.1cd

                              decoded_content_inside
0  OçÕÌù¤r¹¶ºÍ¦s´c%w¤ðÔ¿ÿ.Ü¹zøoÏÂßÝOòå...
1
2  Á¢XÔ}±ëdÛñÐ¯zdA±4rjíÉÊEXÆäÕcIs5è<çMÜ...
3  ÐJ÷¼ìÃhìçöÙx¶¶ÙÃÜ¼f¦²-XÐ}V³ÄÄót_...
4    +6®vD+±UÝC>|ÑA  ëQ`d)ÂØ>¬ÊF\rX¨¯6¾>Ò¸á^...
```

In [24]:
```python
df_with_random_selection['decoded_content_inside']
```

Out[24]:
```
0
OçÕÌù¤r¹¶ºÍ¦s´c%w¤ðÔ¿ÿ.Ü¹z
øoÏÂßÝOòå...
1
2        Á
¢XÔ}±ëdÛñÐ¯zdA±4rjíÉÊEXÆäÕcIs5è<çMÜ...
3        ÐJ÷¼ìÃhìçöÙx¶¶ÙÃÜ¼f¦²-XÐ}V³ÄÄót_...
4          +6®
vD+±UÝC>|Ñ
A  ëQ`d)ÂØ>¬ÊF\rX¨¯6¾>Ò¸á^...
                               ...
82493    X1¡Í{^¸<æÍÊ
íäz  ÐaènÂÑ$k  ñ~ã«f<a  ~¿ÊÈ.ê5...
82494    Ò ß'°ág§þiÝ8ÁæÂd*ÍKItÃÏÕoXE½þ±¿íð\°
...
82495    \rwxíÓ¢\r!îd2qen=óe1ÀÂiîãûñµÀ¸K³õÞUdF...
82496    ¤í×È7!rÕC_Í,÷w*ªL-´³¬ÞlÂ§RDsUÎMÆÔ...
82497    ]ÊægR(zÉ®HSÏ3¤ºç_s#ÍEÇh
Í¤%zqE|u³$X...
Name: decoded_content_inside, Length: 82498, dtype: object
```

In [19]:
```python
import sqlite3
import zipfile
import io
import pandas as pd

# Connect to the SQLite database
conn = sqlite3.connect('eng_subtitles_database.db')
cursor = conn.cursor()

# Query the 'zipfiles' table to fetch rows in batches
batch_size = 1000
offset = 0
rows = []
while True:
    cursor.execute(f'SELECT * FROM zipfiles LIMIT {batch_size} OFFSET {offset}')
    batch_rows = cursor.fetchall()
    if not batch_rows:
        break
    rows.extend(batch_rows)
    offset += batch_size

# Close the cursor and connection to the database
cursor.close()
conn.close()

# Calculate 30% of the total number of rows
num_rows_to_process = int(len(rows) * 0.3)

# Initialize an empty list to store the data
data = []

# Define a function to extract dialogues from content
def extract_dialogues(content):
    dialogue_pattern = r'(\d{2}:\d{2}:\d{2},\d{3} --> \d{2}:\d{2}:\d{2},\d{3})\r?\n([^\r\n]+)'
    dialogues = re.findall(dialogue_pattern, content)
    extracted_content = '\n'.join([f'{timestamp}\n{dialogue}' for timestamp, dialogue in dialogues])
    return extracted_content

# Example of processing subtitle files
for row in rows[:num_rows_to_process]:  # Processing only 30% of the subtitle files
    num, name, content = row
    # Read the ZIP file content from the database
    zip_file = io.BytesIO(content)
    # Extract the contents of the ZIP file
    with zipfile.ZipFile(zip_file, 'r') as zip_ref:
        # Assuming there is only one file in each ZIP archive
        extracted_file = zip_ref.open(zip_ref.namelist()[0])
        # Read and decode the content of the extracted file with fallback encoding
        try:
            decoded_content = extracted_file.read().decode('utf-8')
        except UnicodeDecodeError:
            decoded_content = extracted_file.read().decode('latin-1')
    # Extract dialogues from the decoded content
    separated_content = extract_dialogues(decoded_content)
    # Append data to the list
    data.append((num, name, separated_content))

# Convert the list of tuples to a DataFrame
df = pd.DataFrame(data, columns=['Subtitle Number', 'Name', 'Separated Content'])

# Display the DataFrame
print(df)
```

```
       Subtitle Number                                             Name  \
0              9180533                         the.message.(1976).eng.1cd
1              9180583  here.comes.the.grump.s01.e09.joltin.jack.in.bo...
2              9180592    yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd
3              9180594    yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd
4              9180600                              broker.(2022).eng.1cd
...                ...                                                ...
24744          9279492       scorpion.s04.e15.wave.goodbye.(2018).eng.1cd
24745          9279493     scorpion.s04.e16.nerd.wind.fire.(2018).eng.1cd
24746          9279494      scorpion.s04.e17.dumbster.fire.(2018).eng.1cd
24747          9279495  scorpion.s04.e18.dork.day.afternoon.(2018).eng...
24748          9279496         scorpion.s04.e19.gator.done.(2018).eng.1cd

                                       Separated Content
0      00:00:06,000 --> 00:00:12,074\nWatch any video...
1      00:00:29,359 --> 00:00:32,048\nAh! There's Pri...
2      00:00:53,200 --> 00:00:56,030\n<i>Yumi's Cells...
3      00:00:06,000 --> 00:00:12,074\nWatch any video...
4      00:00:06,000 --> 00:00:12,074\nWatch any video...
...                                                  ...
24744  00:00:01,101 --> 00:00:03,798\n- WALTER:<i>Pre...
24745  00:00:01,101 --> 00:00:04,367\n- WALTER:<i>Pre...
24746  00:00:01,101 --> 00:00:04,071\n- WALTER:<i>Pre...
24747  00:00:02,002 --> 00:00:05,029\n- Kid, we'll ru...
24748  00:00:01,101 --> 00:00:02,399\nWALTER:<i>Previ...

[24749 rows x 3 columns]
```

In [18]:
```python
import sqlite3
import zipfile
import io
import pandas as pd
import re

# Connect to the SQLite database
conn = sqlite3.connect('eng_subtitles_database.db')
cursor = conn.cursor()

# Query the 'zipfiles' table to fetch all rows
cursor.execute('SELECT * FROM zipfiles')

# Fetch all rows from the table
rows = cursor.fetchall()

# Close the cursor and connection to the database
cursor.close()
conn.close()

# Define a function to extract dialogues from content
def extract_dialogues(content):
    dialogue_pattern = r'\d{2}:\d{2}:\d{2},\d{3} --> \d{2}:\d{2}:\d{2},\d{3}\r?\n([^\r\n]+)'
    dialogues = re.findall(dialogue_pattern, content)
    extracted_content = '\n'.join(dialogues)
    return extracted_content

# Initialize an empty list to store the data
data = []

# Example of processing subtitle files
for row in rows:
    num, name, content = row
    # Read the ZIP file content from the database
    zip_file = io.BytesIO(content)
    # Extract the contents of the ZIP file
    with zipfile.ZipFile(zip_file, 'r') as zip_ref:
        # Assuming there is only one file in each ZIP archive
        extracted_file = zip_ref.open(zip_ref.namelist()[0])
        try:
            # Read and decode the content of the extracted file
            decoded_content = extracted_file.read().decode('utf-8')
        except UnicodeDecodeError:
            try:
                # Try decoding with a different encoding
                decoded_content = extracted_file.read().decode('latin1')
            except UnicodeDecodeError:
                # Handle the decoding error gracefully
                print(f"Error decoding file: {name}. Skipping...")
                continue
    # Extract dialogues from the decoded content
    separated_content = extract_dialogues(decoded_content)
    # Append data to the list
    data.append((num, name, separated_content))

# Convert the list of tuples to a DataFrame
df = pd.DataFrame(data, columns=['Subtitle Number', 'Name', 'Separated Content'])

# Print the DataFrame
print(df)
```

```
       Subtitle Number                                             Name  \
0              9180533                         the.message.(1976).eng.1cd
1              9180583  here.comes.the.grump.s01.e09.joltin.jack.in.bo...
2              9180592    yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd
3              9180594    yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd
4              9180600                              broker.(2022).eng.1cd
...                ...                                                ...
82493          9521935                   the.prophets.game.(2000).eng.1cd
82494          9521937                         west.beirut.(1998).eng.1cd
82495          9521938         frankenstein.the.true.story.(1973).eng.1cd
82496          9521940         frankenstein.the.true.story.(1973).eng.1cd
82497          9521941              zombie.island.massacre.(1984).eng.1cd

                                       Separated Content
0      Watch any video online with Open-SUBTITLES\nIn...
1      Ah! There's Princess\nBlooney Looney soldiers\...
2      <i>Yumi's Cells 2</i>\n<i>Episode 36\n<i>Yumi....
3      Watch any video online with Open-SUBTITLES\n<i...
4      Watch any video online with Open-SUBTITLES\nIf...
...                                                  ...
82493  God,\n"With red on\nWith green on his\nWho is ...
82494  api.OpenSubtitles.org is deprecated, please\nI...
82495  (Dramatic orchestral music)\nAdvertise your pr...
82496  Advertise your product or brand here\nElizabet...
82497  (Sharp whistling)\n- [Man] Hey, wait, wait!\n(...

[82498 rows x 3 columns]
```

In [20]:
```python
df
```

Out[20]:

	Subtitle Number	Name	Separated Content
0	9180533	the.message.(1976).eng.1cd	00:00:06,000 --> 00:00:12,074\nWatch any video...
1	9180583	here.comes.the.grump.s01.e09.joltin.jack.in.bo...	00:00:29,359 --> 00:00:32,048\nAh! There's Pri...
2	9180592	yumis.cells.s02.e13.episode.2.13.(2022).eng.1cd	00:00:53,200 --> 00:00:56,030\n<i>Yumi's Cells...
3	9180594	yumis.cells.s02.e14.episode.2.14.(2022).eng.1cd	00:00:06,000 --> 00:00:12,074\nWatch any video...
4	9180600	broker.(2022).eng.1cd	00:00:06,000 --> 00:00:12,074\nWatch any video...
...	...	...	...
24744	9279492	scorpion.s04.e15.wave.goodbye.(2018).eng.1cd	00:00:01,101 --> 00:00:03,798\n- WALTER:<i>Pre...
24745	9279493	scorpion.s04.e16.nerd.wind.fire.(2018).eng.1cd	00:00:01,101 --> 00:00:04,367\n- WALTER:<i>Pre...
24746	9279494	scorpion.s04.e17.dumbster.fire.(2018).eng.1cd	00:00:01,101 --> 00:00:04,071\n- WALTER:<i>Pre...
24747	9279495	scorpion.s04.e18.dork.day.afternoon.(2018).eng...	00:00:02,002 --> 00:00:05,029\n- Kid, we'll ru...
24748	9279496	scorpion.s04.e19.gator.done.(2018).eng.1cd	00:00:01,101 --> 00:00:02,399\nWALTER:<i>Previ...

24749 rows × 3 columns

In [21]:
```python
import pandas as pd

# Assuming df contains your DataFrame

# Save DataFrame to CSV with tab delimiter
df.to_csv('subtitles_data.csv', sep='\t', index=False)
```

After successfully extracting data stored each file in folder

In [24]:
```python
import os

# Create a folder named 'data_subtitle' if it doesn't exist
folder_name = 'data_subtitle1'
if not os.path.exists(folder_name):
    os.makedirs(folder_name)

# Iterate through each row in the DataFrame and save the data to separate files
for index, row in df.iterrows():
    subtitle_id = row['Subtitle Number']
    name = row['Name']
    data = row['Separated Content']
    file_name = f"{subtitle_id}_{name}.txt"
    file_path = os.path.join(folder_name, file_name)
    with open(file_path, 'w', encoding='utf-8') as file:
        file.write(data)

print("Subtitle data saved to individual files in 'data_subtitle' folder.")
```

Subscribe Data saved to individual files in "data_subfolder" folder.