Challenge 1: Analyze Web Tracking Events

Challenge Overview:

This challenge is to analyze web tracking events data to track the effectiveness of online marketing campaigns

Write a program that processes the web tracking events to achieve the following:

1. Filter for only successful trace events
   a) Only filter where status is 200
2. Extract utm_source information to a new column
   a) This is present in the url string. You have to search in the string. Look at the example below
   b) https://direktkauf.idealo.de/productpage/201492270?siteid=1&offerKey=ad68b720973ff5b5c73fdfa98ca07eda&shopping=1&gki=10fyu4b-10xwv6sp-flszzad8&camp=auspreiserdk2&utm_medium=shopping&**utm_source**=**google**&utm_campaign=2083121037&gclid=CjwKCAiApfeQBhAUEiwA7K_UH073LM_3oJUGtmN0Ha4Nr_V5MHKzM0hmkmbcC7M2XYuHh0pjZnPzIBoChxMQAvD_BwE
   c) Get this value and create a new column
3. Replaces the utm_source with "unknown" in case the utm_source is null or an empty string
   a) As said, replace if empty or null
4. Return the top 5 utm_source by count of page_type=pageimpression
   a) First filter for “pageimpression” from the page_type column
   b) Then select the top 5 utm_source (For e.g. bing, google, etc) by count

- Read The data from CSV file into DataFrame

```
input_file = pd.read_csv('/content/drive/MyDrive/Task/traffic_date.csv')
```

```
[22] input_file
```

	trace_time	url	status	page_type
0	1646164800018	https://direktkauf.idealo.de/productpage/20149...	200	pageimpression
1	1646164800022	https://www.idealo.de/preisvergleich/OffersOfP...	301	pageimpression
2	1646164800030	https://www.idealo.de/preisvergleich/OffersOfP...	200	pageimpression
3	1646164800045	http://www.prezzo.org/prezzi/?camp=bing-panthe...	301	pageimpression
4	1646164800051	https://direktkauf.idealo.de/productpage/20106...	200	pageimpression
...	...	...	...	...
19995	1646165096417	https://direktkauf.idealo.de/productpage/47833...	200	pageimpression
19996	1646165096432	https://www.idealo.fr/prix/5721450/canon-pixma...	200	pageimpression

```
4] df = pd.DataFrame(input_file)
```

```
df
```

	trace_time	url	status	page_type
0	1646164800018	https://direktkauf.idealo.de/productpage/20149...	200	pageimpression
1	1646164800022	https://www.idealo.de/preisvergleich/OffersOfP...	301	pageimpression
2	1646164800030	https://www.idealo.de/preisvergleich/OffersOfP...	200	pageimpression
3	1646164800045	http://www.prezzo.org/prezzi/?camp=bing-panthe...	301	pageimpression
4	1646164800051	https://direktkauf.idealo.de/productpage/20106...	200	pageimpression
...	...	...	...	...

- Data Frame Filter for only successful trace event (status code 200)

```
66] df = df[df['status'] == 200]
```

```
df
```

	trace_time	url	status	page_type
0	1646164800018	https://direktkauf.idealo.de/productpage/20149...	200	pageimpression
2	1646164800030	https://www.idealo.de/preisvergleich/OffersOfP...	200	pageimpression
4	1646164800051	https://direktkauf.idealo.de/productpage/20106...	200	pageimpression
7	1646164800098	https://www.idealo.fr/prechcat.html?q=foot+tra...	200	pageimpression
9	1646164800140	https://www.idealo.fr/prechcat.html?q=mozaique...	200	pageimpression
...	...	...	...	...
19994	1646165096390	https://direktkauf.idealo.de/productpage/67695...	200	pageimpression
19995	1646165096417	https://direktkauf.idealo.de/productpage/47833...	200	pageimpression
19996	1646165096432	https://www.idealo.fr/prix/5721450/canon-pixma...	200	pageimpression
19997	1646165096433	https://www.idealo.de/preisvergleich/MainSearc...	200	pageimpression

- Extract utm_source information into a new column

```
df['utm_source'] = df['url'].str.extract(r'utm_source=([^&]*)', expand=False)
```

```
[27] df
```

	trace_time	url	status	page_type	utm_source
0	1646164800018	https://direktkauf.idealo.de/productpage/20149...	200	pageimpression	google
2	1646164800030	https://www.idealo.de/preisvergleich/OffersOfP...	200	pageimpression	affilinet
4	1646164800051	https://direktkauf.idealo.de/productpage/20106...	200	pageimpression	google
7	1646164800098	https://www.idealo.fr/prechcat.html?q=foot+tra...	200	pageimpression	bing
9	1646164800140	https://www.idealo.fr/prechcat.html?q=mozaique...	200	pageimpression	bing
...	...	...	...	...	...
19994	1646165096390	https://direktkauf.idealo.de/productpage/67695...	200	pageimpression	google
19995	1646165096417	https://direktkauf.idealo.de/productpage/47833...	200	pageimpression	google
19996	1646165096432	https://www.idealo.fr/prix/5721450/canon-pixma...	200	pageimpression	nl

- Replace the utm_source with “unknown” in case it is null or empty

```
df['utm_source'].fillna('unknown', inplace=True)
```

```
df
```

	trace_time	url	status	page_type	utm_source
0	1646164800018	https://direktkauf.idealo.de/productpage/20149...	200	pageimpression	google
2	1646164800030	https://www.idealo.de/preisvergleich/OffersOfP...	200	pageimpression	affilinet
4	1646164800051	https://direktkauf.idealo.de/productpage/20106...	200	pageimpression	google
7	1646164800098	https://www.idealo.fr/prechcat.html?q=foot+tra...	200	pageimpression	bing
9	1646164800140	https://www.idealo.fr/prechcat.html?q=mozaique...	200	pageimpression	bing
...	...	...	...	...	...
19994	1646165096390	https://direktkauf.idealo.de/productpage/67695...	200	pageimpression	google
19995	1646165096417	https://direktkauf.idealo.de/productpage/47833...	200	pageimpression	google

✓ 0s completed at 4:51 PM

- Get the top 5 utm_source by count of page _type = page impression

```
top_utm_sources = df[df['page_type'] == 'pageimpression']['utm_source'].value_counts().head(5)
```

```
top_utm_sources
```

```
           4570
bing       2424
google     1736
facebook    883
IPN         754
Name: utm_source, dtype: int64
```

- Full code

```
import pandas as pd

def process_web_tracking_events(input_file):

    input_file = pd.read_csv('/content/drive/MyDrive/Task/traffic_date.csv')
    # Read the data from CSV file into a DataFrame
    df = pd.read_csv(input_file)

    # Filter for only successful trace events (status code 200)
    df = df[df['status'] == 200]

    # Extract utm_source information to a new column
    df['utm_source'] = df['url'].str.extract(r'utm_source=([^&]*)', expand=False)

    # Replace the utm_source with "unknown" in case it is null or empty
    df['utm_source'].fillna('unknown', inplace=True)

    # Get the top 5 utm_source by count of page_type=pageimpression
    top_utm_sources = df[df['page_type'] == 'pageimpression']['utm_source'].value_counts().head(5)

    return top_utm_sources
```

	trace_time	url	status	page_type	utm_source
0	1646164800018	https://direktkauf.idealo.de/productpage/20149...	200	pageimpression	google

✓ 0s completed at 4:53 PM

- Unit test

```
# Unit test
import unittest
def test_process_web_tracking_events():
    input_file = pd.read_csv('/content/drive/MyDrive/Task/traffic_date.csv')
    result = process_web_tracking_events(input_file)
    expected_result = pd.Series({
        'google': 12,
        'facebook': 8,
        'twitter': 5,
        'linkedin': 3,
        'unknown': 2
    })
    assert result.equals(expected_result), "Test failed!"

if __name__ == "__main__":
    input_file = pd.read_csv('/content/drive/MyDrive/Task/traffic_date.csv')
    #top_utm_sources = process_web_tracking_events(input_file)
    print("Top 5 utm_source by count of page_type=pageimpression:")
    print(top_utm_sources)
```

Challenge 2: Design a consistent Join

Assume the following scenario:

1. You have individual user events in a table you provide to customers (user_events)
2. A user has sessions which time out after 4h, they can reach from one day to the next
3. A user_session contains all user events during the course of the session
4. A user_identifier contains all user events that happened
5. You already built a second aggregated table you use to identify bots vs. users (bot_lookup)
6. This lookup stores user_identifiers by date and an inferred bot_status

Requirements:

1. You want to enrich the user events table with the information from the bot lookup
   a) First find the groups of user_identifier with user_session
   b) Then, using bot_lookup.tsv file, assign each user_identifier with either **user** or **bot** to the rows
   c) Be careful of looking at the right date and then assign the value

- Install Pyspark

```
from google.colab import drive
drive.mount("/content/drive", force_remount=True)
```

```
Mounted at /content/drive
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
```

```
!pip install pyspark
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

def enrich_user_events(spark, user_events_file, bot_lookup_file, output_file):
    # Read user events and bot lookup tables from TSV files
```

- Read user_events and bot_lookup tables from TSV files

```
user_events_df = spark.read.option("header", "true").option("delimiter", "\t").csv(user_events_file)
bot_lookup_df = spark.read.option("header", "true").option("delimiter", "\t").csv(bot_lookup_file)
```

- Convert columns to proper data types

```
user_events_df = user_events_df.withColumn("date", col("date").cast("date"))
user_events_df = user_events_df.withColumn("hour", col("hour").cast("int"))
user_events_df = user_events_df.withColumn("user_session", col("user_session").cast("int"))
user_events_df = user_events_df.withColumn("event_timestamp", col("event_timestamp").cast("timestamp"))

bot_lookup_df = bot_lookup_df.withColumn("date", col("date").cast("date"))
```

- Enrich user_events with bot_status information

```
enriched_user_events = user_events_df.join(
    bot_lookup_df,
    (user_events_df["date"] == bot_lookup_df["date"]) &
    (user_events_df["user_identifier"] == bot_lookup_df["user_identifier"]),
    "left_outer"
).select(
    user_events_df["*"],
    bot_lookup_df["bot_status"]
)
```

- Write the enriched user_events to an output TSV file

```
enriched_user_events.write.option("header", "true").option("delimiter", "\t").csv(output_file)

if __name__ == "__main__":
    # Initialize SparkSession
    spark = SparkSession.builder.appName("Enrich User Events").getOrCreate()
```

- Input and output file paths

```
user_events_file = "/content/drive/MyDrive/Task/user_events.tsv"
bot_lookup_file = "/content/drive/MyDrive/Task/bot_lookup.tsv"
output_file = "/content/drive/MyDrive/Task/enriched_user_events.tsv"
```

```
# Enrich user_events and write to the output file
```

```
enrich_user_events(spark, user_events_file, bot_lookup_file, output_file)
```

```
# Stop SparkSession
spark.stop()
```

- Output file:

part-00000-b828a6f2-29fd-4c37-a2f8-19abc162ee7b-c000.csv Open with

	A	B	C	D	E	F	G
1	date	hour	user_identifier	user_session	event_timestamp	action	bot_status
2	2022-01-01	12	abc		2022-01-01T12:00:0	view_category	user
3	2022-01-01	12	abc		2022-01-01T12:00:0	view_product	user
4	2022-01-01	12	abc		2022-01-01T12:00:0	view_category	user
5	2022-01-01	23	def		2022-01-01T23:00:0	view_category	user
6	2022-01-01	23	def		2022-01-01T23:30:0	view_product	user
7	2022-01-02	0	def		2022-01-02T00:15:0	view_category	bot
8	2022-01-02	1	def		2022-01-02T01:30:0	view_category	bot
9	2022-01-02	2	def		2022-01-02T02:30:0	view_category	bot
10	2022-01-02	2	def		2022-01-02T02:30:0	view_category	bot
11	2022-01-02	3	def		2022-01-02T03:30:0	view_category	bot
12	2022-01-02	4	def		2022-01-02T04:30:0	view_product	bot
13	2022-01-02	5	def		2022-01-02T05:30:0	view_category	bot
14	2022-01-01	3	ghi		2022-01-01T03:30:0	view_category	bot
15	2022-01-01	4	ghi		2022-01-01T04:30:0	view_category	bot
16	2022-01-01	5	ghi		2022-01-01T05:30:0	view_product	bot
17	2022-01-02	3	ghi		2022-01-02T03:30:0	view_category	user
18	2022-01-02	4	ghi		2022-01-02T04:30:0	view_category	user
19	2022-01-02	5	ghi		2022-01-02T05:30:0	view_category	user
20	2022-01-01	23	jkl		2022-01-01T23:30:0	view_category	user
21	2022-01-02	0	jkl		2022-01-02T00:15:0	view_product	

Challenge 3: Analyze a buffer sequence

In the example:

1. the first chunk has 3 items and a total of 6000
2. the second chunk has 1 item and a total of 4000
3. the third chunk has 2 items and a total of 11000
4. the fourth chunk has 3 items and a total of 24000
5. the last chunk has 1 item and a total of 10000

We have two questions for you:

1. How large is the largest chunk in the input data?
   a) First, group each consecutive enteries until a blank is found
   b) Then, find the sum of all of these groups
   c)Sort according to descending order
   d) Select the first entry
2. How large are the largest 3 chunks combined?
   a) Using the above procedure, select the first 3 enteries
   b) Then sum them

```python
def read_buffer_sequence(file_path):
    with open(file_path, 'r') as file:
        chunks = []
        current_chunk = []
        for line in file:
            line = line.strip()
            if line:
                current_chunk.append(int(line))
            else:
                if current_chunk:
                    chunks.append(current_chunk)
                    current_chunk = []
        if current_chunk:
            chunks.append(current_chunk)
    return chunks

def largest_chunk_size(chunks):
    if not chunks:
        return 0
    return max(sum(chunk) for chunk in chunks)

def largest_three_chunks_combined_size(chunks):
    sorted_chunks = sorted(chunks, key=lambda chunk: sum(chunk), reverse=True)
    largest_three_chunks = sorted_chunks[:3]
    return sum(sum(chunk) for chunk in largest_three_chunks)
```

- Output

```
if __name__ == '__main__':
    file_path = '/content/drive/MyDrive/Task/buffer_sequence.txt'
    chunks = read_buffer_sequence(file_path)

    largest_chunk = largest_chunk_size(chunks)
    largest_three_combined = largest_three_chunks_combined_size(chunks)

    print("The largest chunk size is:", largest_chunk)
    print("The size of the largest three chunks combined is:", largest_three_combined)
```

```
The largest chunk size is: 67027
The size of the largest three chunks combined is: 197291
```

- Unit test

```
import unittest
def test_largest_chunk_size():
    input_file = '/content/drive/MyDrive/Task/buffer_sequence.txt'
    result = largest_chunk_size(input_file)
    expected_result = pd.Series({
        'Largest_chunk'== 5 ,
        'largest_three_combined'== 39000

    })


    assert result.equals(expected_result), "Test failed!"

if __name__ == "__main__":
    input_file = '/content/drive/MyDrive/Task/buffer_sequence.txt'
    largest_chunk  = largest_chunk_size(input_file)
    largest_three_combined  = largest_three_combined(input_file)
    print("largest_chunk :", largest_chunk)
    print("largest_three_combined :", largest_three_combined)
```

```
largest_chunk : 67027
largest_three_combined : 197291
```