Question 1: Redpanda version

Now let's find out the version of redpandas.

For that, check the output of the command `rpk help` *inside the container*. The name of the container is `redpanda-1`.

Find out what you need to execute based on the `help` output.

What's the version, based on the output of the command you executed? (copy the entire version

Answer:

```
docker exec -it redpanda-1 rpk version
```

```
PS C:\Users\wongs\OneDrive\Desktop\Homework6> docker exec -it redpanda-1 rpk version
v22.3.5 (rev 28b2443)
```

v22.3.5 (rev 28b2443)

Question 2. Creating a topic

Before we can send data to the redpanda server, we need to create a topic. We do it also with the `rpk` command we used previously for figuring out the version of redpandas.

Read the output of `help` and based on it, create a topic with name `test-topic`

What's the output of the command for creating a topic? Include the entire output in your answer.

Answer:

```
PS C:\Users\wongs\OneDrive\Desktop\Homework6> docker exec -it redpanda-1 rpk topic create test-topic
TOPIC       STATUS
```

Question 3. Connecting to the Kafka server

We need to make sure we can connect to the server, so later we can send some data to its topics

First, let's install the kafka connector (up to you if you want to have a separate virtual environment for that)

```
pip install kafka-python
```

You can start a jupyter notebook in your solution folder or create a script

Let's try to connect to our server:

```
import json
import time

from kafka import KafkaProducer

def json_serializer(data):
    return json.dumps(data).encode('utf-8')

server = 'localhost:9092'

producer = KafkaProducer(
    bootstrap_servers=[server],
    value_serializer=json_serializer
)

producer.bootstrap_connected()
```

Provided that you can connect to the server, what's the output of the last command?

Answer:

```python
import json
import time

from kafka import KafkaProducer

def json_serializer(data):
    return json.dumps(data).encode('utf-8')

server = 'localhost:9092'

producer = KafkaProducer(
    bootstrap_servers=[server],
    value_serializer=json_serializer
)

producer.bootstrap_connected()
```

✓ 1.9s

```
True
```

True.

Question 4. Sending data to the stream

Now we're ready to send some test data:

```
t0 = time.time()

topic_name = 'test-topic'

for i in range(10):
    message = {'number': i}
    producer.send(topic_name, value=message)
    print(f"Sent: {message}")
    time.sleep(0.05)

producer.flush()
```

```
t1 = time.time()
print(f'took {(t1 - t0):.2f} seconds')
```

How much time did it take? Where did it spend most of the time?

- Sending the messages
- Flushing
- **Both took approximately the same amount of time**

(Don't remove `time.sleep` when answering this question)

```python
t0 = time.time()

topic_name = 'test-topic'

for i in range(10):
    message = {'number': i}
    producer.send(topic_name, value=message)
    print(f"Sent: {message}")
    time.sleep(0.05)

producer.flush()

t1 = time.time()
print(f'took {(t1 - t0):.2f} seconds')
```

✓ 0.5s

```
Sent: {'number': 0}
Sent: {'number': 1}
Sent: {'number': 2}
Sent: {'number': 3}
Sent: {'number': 4}
Sent: {'number': 5}
Sent: {'number': 6}
Sent: {'number': 7}
Sent: {'number': 8}
Sent: {'number': 9}
took 0.53 seconds
```

Creating the PySpark consumer

- Create a topic `green-trips` and send the data there

```
PS C:\Users\wongs\OneDrive\Desktop\Homework6> docker exec -it redpanda-1 rpk topic create green-trips
TOPIC        STATUS
green-trips  OK
```

- How much time in seconds did it take? (You can round it to a whole number)
- Make sure you don't include sleeps in your code

```python
import json
import time

from kafka import KafkaProducer

def json_serializer(data):
    return json.dumps(data).encode('utf-8')

server = 'localhost:9092'

producer = KafkaProducer(
    bootstrap_servers=[server],
    value_serializer=json_serializer
)

producer.bootstrap_connected()

t0 = time.time()

topic_name = 'green-trips'

for i in range(len(df_green)):
    message = {'number': i}
    producer.send(topic_name, value=message)
    #print(f"Sent: {message}")
    #time.sleep(0.05)

producer.flush()

t1 = time.time()
print(f'took {(t1 - t0):.2f} seconds')
```

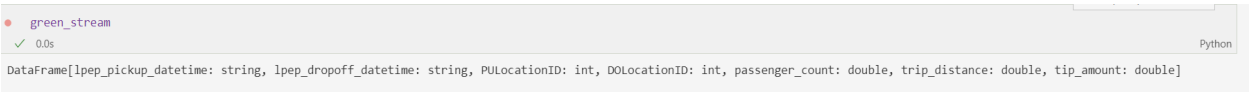
✓ 31.5s

```
took 30.82 seconds
```

30.82 seconds

Question 6. Parsing the data

How does the record look after parsing? Copy the output.
Answer:



DataFrame[lpep_pickup_datetime: string, lpep_dropoff_datetime: string, PULocationID: int, DOLocationID: int, passenger_count: double, trip_distance: double, tip_amount: double]

Question 7: Most popular destination

Now let's finally do some streaming analytics. We will see what's the most popular destination currently based on our stream of data (which ideally we should have sent with delays like we did in workshop 2)

This is how you can do it:

- Add a column "timestamp" using the `current_timestamp` function
- Group by:
    - 5 minutes window based on the timestamp column (`F.window(col("timestamp"), "5 minutes")`)
    - `"DOLocationID"`
- Order by count

You can print the output to the console using this code

```
query = popular_destinations \
    .writeStream \
    .outputMode("complete") \
    .format("console") \
    .option("truncate", "false") \
    .start()

query.awaitTermination()
```

Write the most popular destination, your answer should be *either* the zone ID or the zone name of this destination. (You will need to re-send the data for this to work)

```
from pyspark.sql.functions import current_timestamp

popular_destinations  = green_stream \
    .withColumn("timestamp", current_timestamp()) \
    .groupBy(F.window(F.col("timestamp"), "5 minutes"), 'DOLocationID') \
    .count() \
    .orderBy("count") \

popular_destinations
```

✓ 0.5s

```
DataFrame[window: struct<start:timestamp,end:timestamp>, DOLocationID: int, count: bigint]
```

Answer:

But my kafka consumer failed. I use python pandas to get the data not based in stream of data.

```
df.sort_values(by=['count'], ascending=False).head()
```

✓ 0.0s

DOLocationID	count
74	17741
42	15942
41	14061
75	12840
129	11930

DOLocationID = 74