```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import random
import datetime


#Creo el df
events = pd.DataFrame(columns = ['uid', 'sessionid','event', 'value','datetime'], dtype = int)

event = ['ecommerce.view-product','ecommerce.checkout','ecommerce.conversion','evento4','evento5','evento6','evento7','evento8','even
datetime = ['2020-06-06 06:06:06','2019-09-09 09:09:09', '2020-02-02 02:02:02', '2019-04-04 04:04:04','2020-01-01 01:01:01','2011-02-

start = pd.to_datetime('2015-01-01')
end = pd.to_datetime('2020-10-10')

for i in range(500):
    events = events.append({
        'uid': (i + 4),
        'sessionid': (i + 100),
        'event': random.choice(event),
        'value': i,
        'datetime' : random.choice(datetime)
    }, ignore_index = True)
```

Pandas 2020 2C 1 Enunciado Una importante compañía de e-commerce acumula información de las interacciones de sus usuarios con su plataforma mediante un motor de analytics. Esta información está disponible en un data frame para que sea analizada por nuestro equipo. El data frame llamado `events.csv` posee las siguientes columnas:

- uid: identificador único que representa a un usuario que visita la plataforma.
- sessionid: identificador único de la sesión que generó el usuario.
- event: nombre del evento que se generó.

- value: información adicional que se guarda con el evento.
- datetime: timedate64 que contiene fecha y hora del evento. Para poder realizar un análisis es necesario tener las siguientes consideraciones:

● Cada vez que un usuario visita la plataforma, el sistema le genera una nueva sesión con un `sessionid` asociado. Esto implica que un usuario puede realizar múltiples sesiones.

● Los eventos son interacciones que realiza el usuario con el sistema. Una sesión está compuesta por al menos un evento.

Se le pide a nuestro equipo las siguientes tareas utilizando pandas:

a) Listar aquellos usuarios que tengan un total de sesiones mayor al promedio de sesiones por usuario del sistema. Ej: si el promedio de los usuarios cuenta con 20 sesiones, se debe listar los usuarios que tengan más de 20 sesiones asociadas.

b) Calcular el valor promedio de los eventos 'ecommerce.view-product', 'ecommerce.checkout', 'ecommerce.conversion' para aquellos usuarios que hayan realizado más de 25 sesiones. Los resultados deben ser presentados en un dataframe con las siguientes columnas ('uid','ecommerce_view_product_mean', 'ecommerce_checkout_mean', 'ecommerce_conversion_mean'). Nota: Es posible realizar el cálculo de los promedios en una única operación sobre un dataframe, por lo que soluciones que realizan el cálculo del promedio de cada métrica por separado serán consideradas como menos eficientes.

Punto A

```
#Primero leo el csv
#events = pd.read_csv('events.csv')
events.head()
```

```
        uid  sessionid                event  value          datetime
```

```
#Filtro por cantidad de sesiones por usuario y calculo el promedio
events['cantidad_sesiones_usuario'] = events.groupby('uid').agg({'sessionid': 'count'})
events['promedio_sesiones_usuario'] = events['cantidad_sesiones_usuario'].mean()
events = events.dropna()
events.head()
```

	uid	sessionid	event	value	datetime	cantidad_sesiones_usuario	promedio_sesiones_usuario
4	8	104	ecommerce.view-product	4	2020-01-01 01:01:01	1.0	1.0
5	9	105	evento8	5	2019-09-09 09:09:09	1.0	1.0
6	10	106	ecommerce.checkout	6	2019-04-04	1.0	1.0

```
#Separo a los usuarios con sesiones mayor al promedio y los listo
usuarios_con_sesiones_mayor_al_promedio = events[events['cantidad_sesiones_usuario'] >= events['promedio_sesiones_usuario']]
usuarios_con_sesiones_mayor_al_promedio['uid']
```

```
4        8
5        9
6       10
7       11
8       12
      ...
495    499
496    500
497    501
498    502
499    503
Name: uid, Length: 496, dtype: int64
```

Punto B

b) Calcular el valor promedio de los eventos 'ecommerce.view-product', 'ecommerce.checkout', 'ecommerce.conversion' para aquellos usuarios que hayan realizado más de 25 sesiones. Los resultados deben ser presentados en un dataframe con las siguientes columnas ('uid','ecommerce_view_product_mean', 'ecommerce_checkout_mean', 'ecommerce_conversion_mean').

Nota: Es posible realizar el cálculo de los promedios en una única operación sobre un dataframe, por lo que soluciones que realizan el cálculo del promedio de cada métrica por separado serán consideradas como menos eficientes.

```
#Separo a los usuarios con más de 25 sesiones
usuarios_con_mas_de_25_sesiones = events[events['cantidad_sesiones_usuario'] >= 25]
usuarios_con_mas_de_25_sesiones['uid']
```

```
4        8
5        9
6       10
7       11
8       12
      ...
495    499
496    500
497    501
498    502
499    503
Name: uid, Length: 496, dtype: int64
```

```
#Separo los eventos pedidos
eventos_pedidos = events[(events['event'] == 'ecommerce.view-product') | (events['event'] == 'ecommerce.checkout') | (events['event']
eventos_pedidos.head()
```

	uid	sessionid	event	value	datetime	cantidad_sesiones_usuario	promedio_sesiones_usuario
			ecommerce.view-		2020-01-		

```
#Junto los dos dataframe
Eventos_con_25_sesiones = usuarios_con_mas_de_25_sesiones.merge(eventos_pedidos)
Eventos_con_25_sesiones.head()
```

	uid	sessionid	event	value	datetime	cantidad_sesiones_usuario	promedio_sesiones_usuario
0	8	104	ecommerce.view-product	4	2020-01-01 01:01:01	1.0	1.0
1	10	106	ecommerce.checkout	6	2019-04-04 04:04:04	1.0	1.0
2	12	108	ecommerce.checkout	8	2010-05-02	1.0	1.0

```
#Pivoteo con los eventos
Eventos_con_25_sesiones_filtrado =Eventos_con_25_sesiones[['uid','event','value']]
Eventos_con_25_sesiones_pivoteado = Eventos_con_25_sesiones_filtrado.pivot_table(index = 'uid', columns = 'event', values = 'value',
Eventos_con_25_sesiones_pivoteado.reset_index().rename_axis(None, axis=1)
```

	uid	ecommerce.checkout	ecommerce.conversion	ecommerce.view-product
0	8	NaN	NaN	4.0
1	10	6.0	NaN	NaN
2	12	8.0	NaN	NaN
3	16	NaN	12.0	NaN

```
#renombro las columnas
df_final = Eventos_con_25_sesiones_pivoteado.rename(columns = {'ecommerce.checkout' :'ecommerce.checkout_mean', 'ecommerce.conversion
df_final.reset_index().rename_axis(None, axis=1)
```

	uid	ecommerce.checkout_mean	ecommerce.conversion_mean	ecommerce.view-product_mean
0	8	NaN	NaN	4.0
1	10	6.0	NaN	NaN
2	12	8.0	NaN	NaN
3	16	NaN	12.0	NaN
4	17	NaN	13.0	NaN
...	...	...	...	...
145	488	NaN	NaN	484.0
146	491	487.0	NaN	NaN
147	497	NaN	493.0	NaN
148	500	NaN	496.0	NaN
149	502	498.0	NaN	NaN

150 rows × 4 columns