```python
#Importo librerias necesarias
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import random


#Creo dataframes para usar como ejemplo
events = pd.DataFrame(columns = ['uid', 'sessionid','event', 'value', 'datetime'], dtype =
datetime = ['2020-12-06','2019-09-06', '2020-04-20', '2019-12-06', '2019-12-06', '2019-12-

id = [101, 102, 104, 118, 140, 109, 108]
value = [20, 25, 28, 10, 11, 14]
precio = [200, 2000, 2800, 100, 100, 2500, 25]
event = ['ecommerce.view-product', 'ecommerce.checkout', 'ecommerce.conversion', 'ecommerc
for i in range(300):
    events = events.append({
        'uid': random.choice(id),
        'sessionid': i+2,
        'event': random.choice(event),
        'value': random.choice(value),
        'datetime': random.choice(datetime)
    }, ignore_index = True)


events = events.dropna()
events['datetime'] = pd.to_datetime(events['datetime'], errors = 'coerce')
events
```

	uid	sessionid	event	value	datetime
0	109	2	ecommerce.conversion	28	2020-04-20
1	102	3	ecommerce.view	20	2019-12-06
2	109	4	ecommerce.view	28	2019-12-15
3	118	5	ecommerce.view	25	2020-04-20
4	118	6	ecommerce.conversion	20	2020-04-20
...	...	...	...	...	...
295	109	297	ecommerce.session	14	2020-04-20
296	140	298	ecommerce.conversion	14	2019-12-06
297	104	299	ecommerce.conversion	11	2019-05-06
298	104	300	ecommerce.view	10	2019-12-15
299	140	301	ecommerce.session	20	2019-09-06

300 rows × 5 columns

A)

```python
events_total = events.groupby('uid').agg({'sessionid':'count'})
events_total =events_total.rename(columns = {'sessionid':'total_sesiones'})
events_total.reset_index()
```

	uid	total_sesiones
0	101	41
1	102	51
2	104	34
3	108	43
4	109	47
5	118	34
6	140	50

```python
promedio_de_sesiones = events_total['total_sesiones'].mean()
promedio_de_sesiones
```

```
42.857142857142854
```

```python
events_final = events_total[events_total['total_sesiones'] > promedio_de_sesiones]
events_final.reset_index()
```

	uid	total_sesiones
0	102	51
1	108	43
2	109	47
3	140	50

B)

```python
events = events[(events['event'] == 'ecommerce.view-product') | (events['event'] == 'ecomm
events
```

	uid	sessionid	event	value	datetime
0	109	2	ecommerce.conversion	28	2020-04-20
4	118	6	ecommerce.conversion	20	2020-04-20
5	118	7	ecommerce.checkout	25	2019-12-15
6	102	8	ecommerce.view-product	11	2020-04-20
8	109	10	ecommerce.conversion	28	2019-12-06
...	...	...	...	...	...
293	104	294	ecommerce.checkout	14	2019-12-06

```
events_filtro['total_sesiones'] = events.groupby('uid')['sessionid'].transform('count')
events_filtro
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarnin
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
  """Entry point for launching an IPython kernel.
```

	uid	sessionid	event	value	datetime	total_sesiones
1	104	3	ecommerce.view-product	11	2019-12-06	NaN
2	108	4	ecommerce.view-product	20	2019-12-06	NaN
3	108	5	ecommerce.view-product	28	2020-12-06	NaN
5	118	7	ecommerce.view-product	28	2019-05-06	17.0
6	101	8	ecommerce.conversion	20	2019-05-06	27.0
...	...	...	...	...	...	...
494	102	496	ecommerce.view-product	14	2019-12-15	NaN
495	140	497	ecommerce.checkout	10	2019-12-06	NaN
496	109	498	ecommerce.conversion	11	2019-12-15	NaN
498	108	500	ecommerce.view-product	10	2019-12-15	NaN
499	101	501	ecommerce.checkout	14	2019-12-06	NaN

311 rows × 6 columns

```
events_25 = events_filtro[events_filtro['total_sesiones'] > 25]
events_25
```

⤷

94	102	96	ecommerce.conversion	25	2019-09-06	33.0
96	140	98	ecommerce.conversion	14	2019-09-06	33.0
102	102	104	ecommerce.conversion	20	2019-12-06	33.0
125	101	127	ecommerce.view-product	14	2020-04-20	26.0
136	140	138	ecommerce.view-product	11	2019-12-15	33.0
139	101	141	ecommerce.checkout	10	2019-05-06	27.0
141	101	143	ecommerce.checkout	25	2019-09-06	27.0
148	108	150	ecommerce.view-product	20	2020-04-20	33.0
154	102	156	ecommerce.conversion	20	2020-12-06	27.0
155	109	157	ecommerce.view-product	28	2019-12-15	33.0
162	102	164	ecommerce.view-product	11	2019-12-06	33.0
167	104	169	ecommerce.view-product	20	2019-09-06	27.0
168	109	170	ecommerce.checkout	28	2019-12-06	26.0
170	109	172	ecommerce.view-product	25	2019-05-06	27.0
175	102	177	ecommerce.conversion	25	2019-12-06	26.0
178	118	180	ecommerce.checkout	11	2019-09-06	33.0
187	109	189	ecommerce.checkout	28	2019-05-06	27.0
191	109	193	ecommerce.view-product	20	2019-12-06	33.0
192	104	194	ecommerce.view-product	11	2019-09-06	26.0
197	101	199	ecommerce.checkout	10	2020-04-20	27.0
218	140	220	ecommerce.checkout	25	2019-12-15	27.0
219	104	221	ecommerce.view-product	14	2019-05-06	33.0
220	108	222	ecommerce.checkout	20	2020-04-20	26.0
233	118	235	ecommerce.checkout	14	2019-12-06	27.0
235	102	237	ecommerce.checkout	25	2019-05-06	33.0
240	109	242	ecommerce.view-product	20	2019-09-06	33.0
257	140	259	ecommerce.checkout	20	2019-12-06	33.0
261	109	263	ecommerce.view-product	28	2020-12-06	33.0
277	118	279	ecommerce.checkout	11	2019-12-06	33.0
284	140	286	ecommerce.checkout	11	2020-12-06	27.0
293	101	295	ecommerce.view-product	20	2019-12-06	33.0
296	102	298	ecommerce.checkout	20	2019-05-06	33.0

```
events_25_promedio = events_25.pivot_table(index = 'uid', columns ='event', values = 'valu
events_25_promedio = events_25_promedio.reset_index().rename_axis(None, axis=1)
events_25_promedio = events_25_promedio.rename(columns = {'ecommerce.checkout': 'ecommerce
                                                          'ecommerce.conversion': 'ecommer
                                                          'ecommerce.view-product': 'ecomm
events_25_promedio
```

	uid	ecommerce_checkout_mean	ecommerce_conversion_mean	ecommerce_view_product_m
0	101	16.250000	15.0	1
1	102	20.000000	22.5	1
2	104	NaN	NaN	1
3	108	22.500000	25.0	2
4	109	23.750000	25.0	2
5	118	15.600000	NaN	l
6	140	18.666667	14.0	1