지그재그의 데이터분석 팀에 오신 것을 환영합니다!

안녕하세요! 지그재그 데이터분석팀에 오신 것을 환영합니다.

지그재그는 여성 패션 쇼핑몰들의 상품 정보를 통합적으로 제공하는 플랫폼으로 개인 취향 알고리즘을 통해 적절한 제품을 추천하여 유저 경험을 높이고 있습니다. 지그재그는 2015년 6월 '내 스타일의 쇼핑몰을 편하게 북마크 해보자'라는 작은 아이디어에서 시작했습니다. 서비스가 출시되고 3년이 지난 지금, 지그재그에 입점한 2,700여 개의 온라인 쇼핑몰들에서는 하루에 10,000여 개의 패션 아이템이 업데이트되며 한 달에 200만 명이 넘는 사용자들이 방문해 자신이 좋아하는 상품을 실시간으로 발견하고 있습니다. 런칭 2년 반 만에 1,000만 앱 다운로드를 돌파하며 대한민국 여성의 쇼핑 필수 앱으로 자리 잡았습니다.

오늘 이 쥬피터 노트북을 받은 수강생분들께서는 지그재그의 일일 데이터 분석가(Data Analyst)로서 일을 할 것입니다. 지그재그의 하루 동안의 로그 데이터를 바탕으로 매출 분석 등 기본적인 데이터 탐색을 할 예정입니다. 일반적으로 매출 또는 핵심 지표에 영향을 미치는 요인에 대한 가설을 세우고 이를 데이터로 검증하는 일이 일반적인 데이터 분석이지만, 때로는 가설 없이 데이터 그 자체를 이해하는 것부터 출발하여 역으로 핵심 지표 또는 매출 등을 개선하기 위한 아이디어를 얻을 수도 있습니다. 이러한 데이터 자체를 이해하는 과정을 탐색적 자료 분석이라고 합니다. 여러분들은 6월 11일에 활동한 고객 정보, 쇼핑몰 정보, 거래 정보, 상품 정보, 그리고 고객의 행동 정보를 이해하고 분석하여 다른 팀들과 정보를 공유하게 될 것입니다.

---

*set options

In [43]:
```
# 데이터를 다루는 library인 padas를 import합니다.
```

```
import pandas as pd
```

In [44]:
```
# 화면에 출력하는 데이터 프레임의 최대 row 수를 500으로 설정합니다.
pd.set_option('display.max_rows', 500)

# 화면에 출력하는 데이터 프레임의 최대 column 수를 500으로 설정합니다.
pd.set_option('display.max_columns', 500)
```

1. data 폴더의 zigzag_DB.db에 연결한 뒤 데이터베이스 스키마를 출력해주세요. 그 다음, order 테이블을 불러와주세요.

zigzag 데이터 테이블들을 모두 zigzag_DB.db 파일에 저장해두었습니다. 저번 주 수업자료를 참고해 database를 조회하고 그 중, order 테이블을 불러와주세요.

로드한 테이블이 다음과 같은 모양이어야 합니다.

	timestamp	user_id	goods_id	shop_id	price
0	2018-06-11 00:00:43.032	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	1414	38	45000
1	2018-06-11 00:02:33.763	smDmRnykg61KajpxXKzQ0oNkrh2nuSBj	1351	12	9500
2	2018-06-11 00:04:06.364	EyGjKYtSqZgqJ1ddKCtH5XwGirTyOH2P	646	14	22000
3	2018-06-11 00:04:17.258	KQBGi33Zxh5Dgu0WEkOkjN0YqTT_wxC3	5901	46	29800
4	2018-06-11 00:05:26.010	lq1Je3voA3a0MouSFba3629lKCvweI24	5572	89	29000

In [45]:
```
# python에서 DB를 다루는 library인 sqlite3을 import 합니다.
import sqlite3
```

In [46]:
```
# connect
connect = sqlite3.connect('data/zigzag_DB.db')
connect
```

Out[46]:
```
<sqlite3.Connection at 0x1a1cd6c030>
```

In [47]:
```
query = "SELECT * FROM sqlite_master"

schema = pd.read_sql(query, connect)

for i in schema['sql']:
    print(i)
```

```
CREATE TABLE "order" (
"timestamp" TEXT,
  "user_id" TEXT,
  "goods_id" INTEGER,
  "shop_id" INTEGER,
  "price" INTEGER
)
CREATE TABLE "good" (
"goods_id" INTEGER,
  "timestamp" TEXT,
  "shop_id" INTEGER,
  "category" TEXT,
  "price" INTEGER,
  "image_type" TEXT,
  "image_width" INTEGER,
  "image_height" INTEGER
)
CREATE TABLE "shop" (
"shop_id" INTEGER,
  "name" TEXT,
  "category" TEXT,
  "age" TEXT,
  "style" TEXT
)
CREATE TABLE "log" (
"timestamp" TEXT,
  "user_id" TEXT,
  "event_origin" TEXT,
  "event_name" TEXT,
  "event_goods_id" REAL,
  "event_shop_id" REAL
)
CREATE TABLE "user" (
"user_id" TEXT,
```

```
  "os" TEXT,
  "age" INTEGER
)
```

In [48]:

```
# 원본
query = "SELECT * FROM 'order'"

order = pd.read_sql(query, connect)
order.head()
```

Out[48]:

	timestamp	user_id	goods_id	shop_id	price
0	2018-06-11 00:00:43.032	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	1414	38	45000
1	2018-06-11 00:02:33.763	smDmRnykg61KajpxXKzQ0oNkrh2nuSBj	1351	12	9500
2	2018-06-11 00:04:06.364	EyGjKYtSqZgqJ1ddKCtH5XwGirTyOH2P	646	14	22000
3	2018-06-11 00:04:17.258	KQBGi33Zxh5Dgu0WEkOkjN0YqTT_wxC3	5901	46	29800
4	2018-06-11 00:05:26.010	lq1Je3voA3a0MouSFba3629lKCvwel24	5572	89	29000

order 테이블의 각 행은 주문이 일어난 로그를 나타내며, 2018년 6월 11일 하루치 데이터가 주어졌습니다.

timestamp는 주문시각, user_id는 주문을 한 유저의 고유 아이디, goods_id는 상품의 id, shop_id는 쇼핑몰의 id, price는 상품의 가격을 나타냅니다.

2. order 테이블을 이용해 지그재그의 당일 매출 상위 10개 쇼핑몰을 구해주세요.

order 테이블을 이용해 지그재그를 통해 이루어진 6월 11일의 쇼핑몰별 매출 정보를 피벗 테이블을 이용하여 구해주세요. 결과는 다음과 같이 나와야 합니다.

shop_id	price
22	1365200
14	872000
63	710700
32	707900
126	669400
6	655900
11	653000
60	558300
19	518400
12	446900

In [49]:

```
table = pd.pivot_table(order,
                                    values = 'price',
                                    index = 'shop_id',
                                    aggfunc = 'sum')

table.sort_values(( 'price'), ascending=False).head(10)
```

Out[49]:

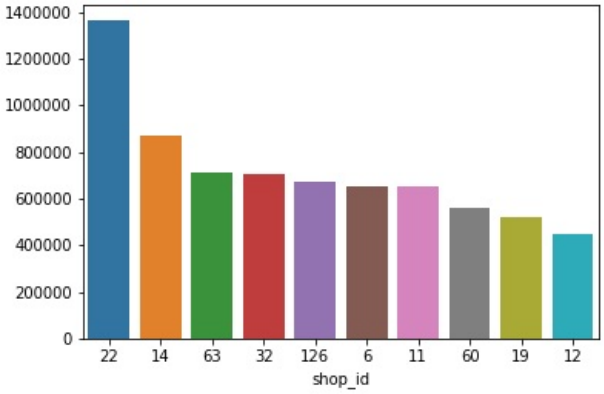
shop_id	price
22	1365200
14	872000
63	710700
32	707900
126	669400
6	655900
11	653000
60	558300
19	518400
12	446900

3. 판매 건수를 포함하여 피벗테이블을 만들어주세요. 또한, 상위 10개 쇼핑몰의 매출을 막대그래프로 보여주세요.

같은 횟수의 거래라도 제품의 가격이 비싸면 매출이 크게 나오게 되어있습니다. 상대적으로 저렴한 물품을 파는 쇼핑몰들은 결제 건수에 비해 매출이 적을 수 있습니다.

매출과 결제 건수를 함께 볼 수 있는 피벗테이블을 작성해주세요. 그리고, 그 결과를 막대그래프로 시각화하여 보여주세요. 결과는 다음과 같이 나오게됩니다.

shop_id	sum	count
22	1365200	99
14	872000	30
63	710700	27
32	707900	37
126	669400	39
6	655900	24
11	653000	19
60	558300	23
19	518400	19
12	446900	42



In [50]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

In [51]:
```
table = pd.pivot_table(order,
                                    values = 'price',
                                    index = 'shop_id',
                                    aggfunc = ['sum', 'count'])

table.columns = ['sum', 'count']
table = table.sort_values('sum', ascending=False)

table_top10 = table.head(10)
table_top10
```

Out[51]:

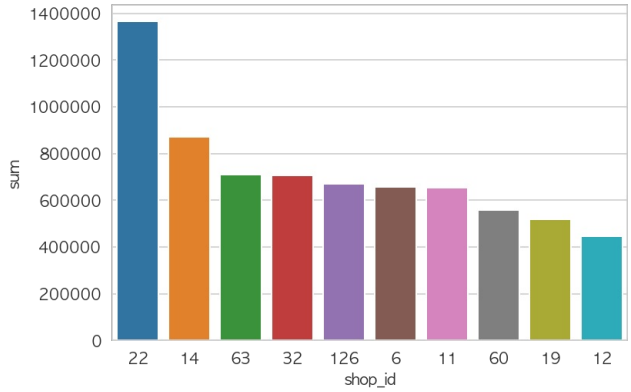
shop_id	sum	count
22	1365200	99
14	872000	30
63	710700	27
32	707900	37
126	669400	39
6	655900	24
11	653000	19
60	558300	23
19	518400	19

12	446900	42

In [53]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [54]:
```
sns.barplot(data=table_top10, x=table_top10.index, y='sum', order=table_top10.index)
```

Out[54]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1b19e710>
```

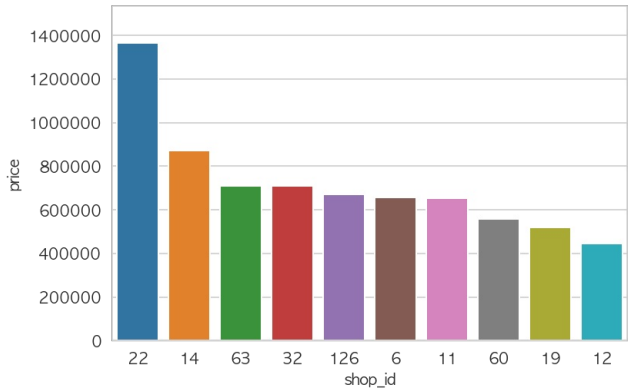


또는, 다음과 같은 방법으로도 시각화를 할 수 있습니다.

In [55]:
```
top10_index = table_top10.index
```

In [56]:
```
sns.barplot(data=order, x='shop_id', y='price',estimator=sum, order=table_top10.index, errwidth=0)
```

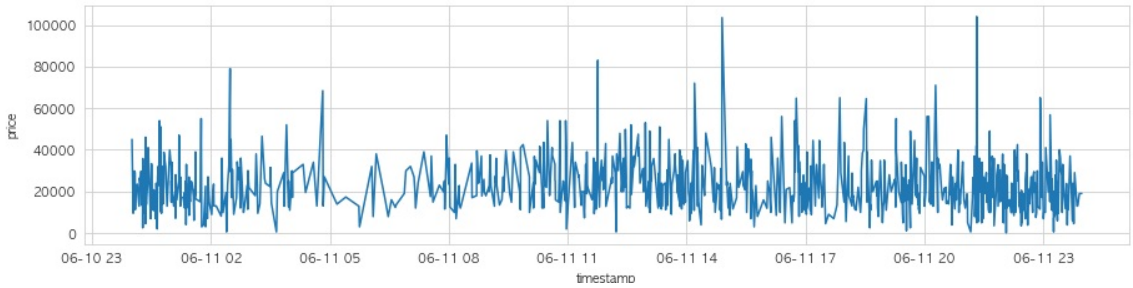
Out[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1cb7cc18>
```



4. 시간대별 지그재그 매출을 구하려고 합니다. lineplot을 이용하여 6월 11일의 시간대별 매출을 시각화 해주세요.

지그재그 이용자들의 구매 패턴을 알아보려고 합니다. timestamp를 to_datetime을 이용하여 datetime 자료형으로 만든 뒤, 이를 이용하여 시간대별 총 매출량을 구해주세요.

x를 timestamp, y를 price로 놓는 경우 다음과 같은 그래프가 나오게 됩니다.



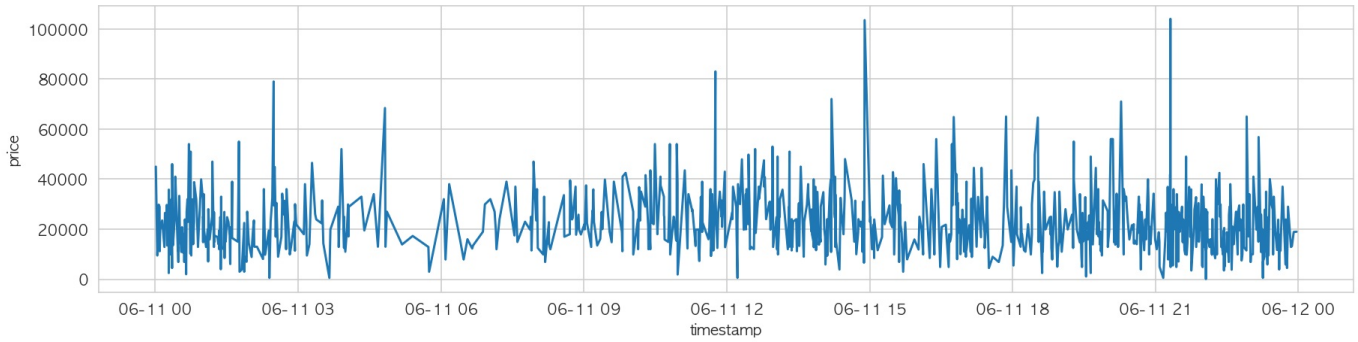
In [57]:
```
import matplotlib as mpl

sns.set_style('whitegrid') #스타일은 원하는 것을 사용하세요.

mpl.rc('font', family='AppleGothic') # Mac의 경우는 AppleGothic, 윈도우의 경우는 Malgun Gothic을 사용하면 됩니다 :)
mpl.rc('axes', unicode_minus=False)

from IPython.display import set_matplotlib_formats
set_matplotlib_formats('retina')
```

In [58]:
```
order['timestamp'] = pd.to_datetime(order['timestamp'])
order.dtypes
```

Out[58]:
```
timestamp    datetime64[ns]
user_id              object
goods_id              int64
shop_id               int64
price                 int64
dtype: object
```

In [59]:
```
plt.figure(figsize=[15,3.5])
sns.lineplot(x='timestamp', y='price', data=order)
```

Out[59]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1cc257b8>
```

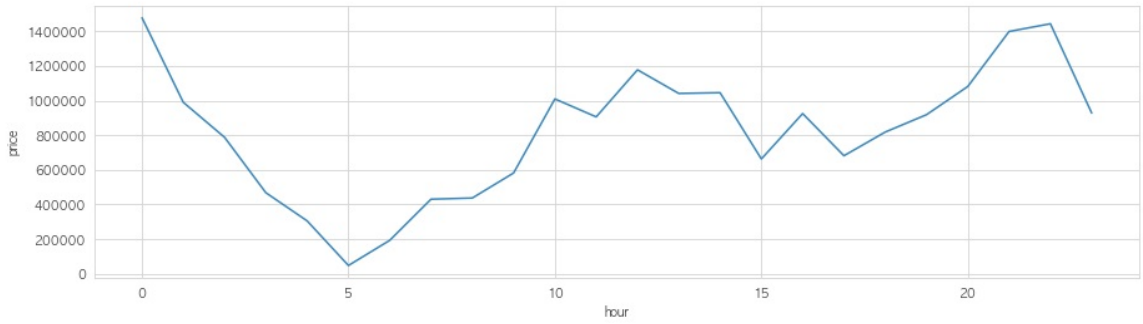


5. 위의 시각화를 구간화(binning) 작업을 거쳐 보기 좋은 형태로 만들어주세요.

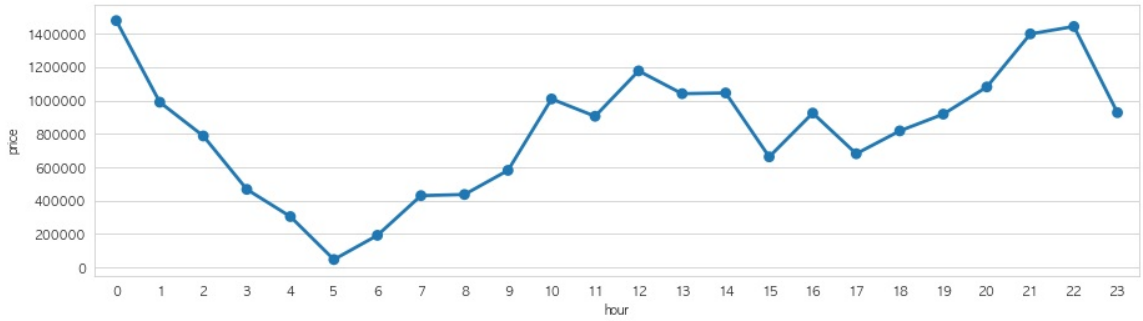
위의 시각화는 적절하지 못한 시각화입니다. 정확하게는 x시 00분 ~ x시 59분 까지의 매출을 시간대별로 모두 합하는 binning 과정을 거친 뒤 시각화를 진행해야 원하는 결과를 얻을 수 있습니다.

order 테이블에 로그의 발생 시간을 나타내는 hour 칼럼을 추가한 뒤, pivot_table을 이용하여 시간대별 매출을 구해주세요. 그리고, 이를 이용해 아래와 같은 그래프를 만들어 주세요.

lineplot을 사용하는 경우



pointplot을 사용하는 경우



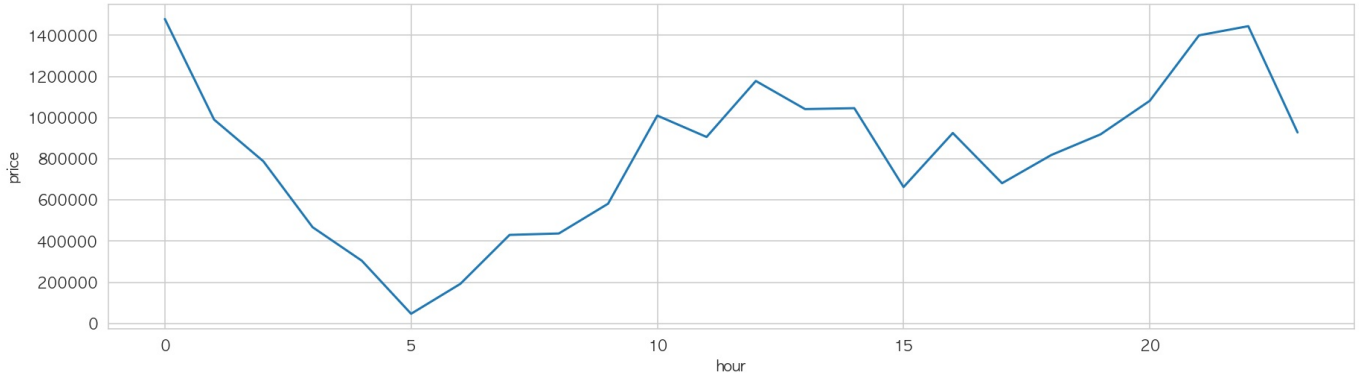
In [60]:
```
order['hour'] = order['timestamp'].dt.hour
```

In [61]:
```
table = order.pivot_table(values='price',
                                        index='hour',
                                        aggfunc='sum')

table.head()
```

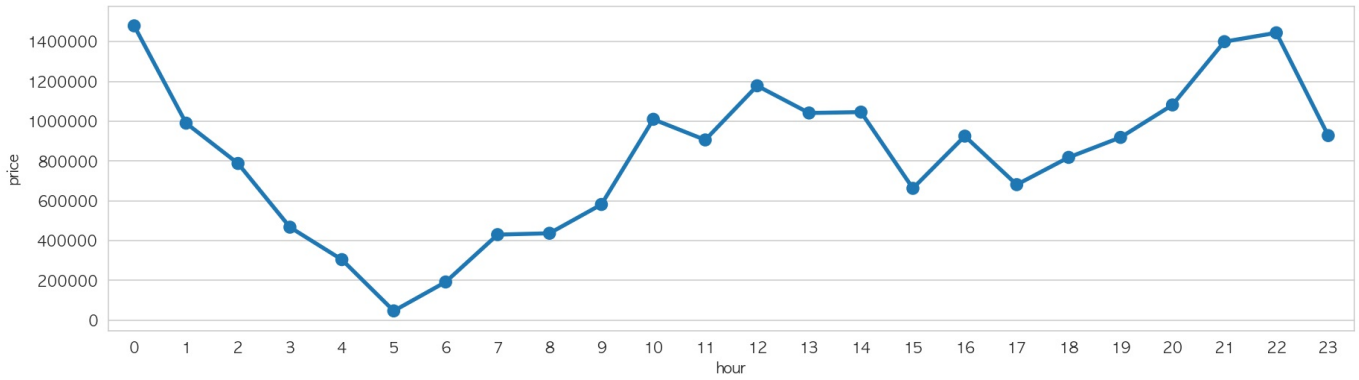
Out[61]:

hour	price
0	1479210
1	990300
2	787830
3	467650
4	304800

In [62]:
```
plt.figure(figsize=[15,4])
sns.lineplot(data=table, x=table.index, y='price')
plt.savefig('image3.png')
```



In [63]:
```
plt.figure(figsize=[15,4])
sns.pointplot(data=table, x=table.index, y='price')
plt.savefig('image4.png')
```



6. user 테이블을 분리와 order 테이블과 병합해주세요

6 : user 테이블을 불러와 order 테이블과 병합해주세요.

user 테이블에는 고객의 고유 아이디를 나타내는 user_id, 그리고 접속 기기정보를 나타내는 os, 그리고 나이정보 age가 있습니다. 이를 이용해 top 10 쇼핑몰 매출이 어떤 연령층에서 발생했는지 분석하고자 합니다. 데이터베이스에서 user 테이블을 불러온 뒤, order 테이블과 병합해주세요.

병합 결과는 다음과 같이 나오게 됩니다.

	timestamp	user_id	goods_id	shop_id	price	hour	os	age
0	2018-06-11 00:00:43.032	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	1414	38	45000	0	iOS	39
1	2018-06-11 00:02:33.763	smDmRnykg61KajpxXKzQ0oNkrh2nuSBj	1351	12	9500	0	And	17
2	2018-06-11 00:04:06.364	EyGjKYtSqZgqJ1ddKCtH5XwGirTyOH2P	646	14	22000	0	And	-1
3	2018-06-11 00:04:17.258	KQBGi33Zxh5Dgu0WEkOkjN0YqTT_wxC3	5901	46	29800	0	And	34
4	2018-06-11 00:05:26.010	lq1Je3voA3a0MouSFba3629lKCvwel24	5572	89	29000	0	And	17

In [64]:
```
query = "SELECT * FROM 'user'"
user = pd.read_sql(query, connect)
print(user.shape)
user.head()
```

```
(10000, 3)
```

Out[64]:

	user_id	os	age
0	--PYPMX8QWg0ioT5zfORmU-S5LIn0lot	And	41
1	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	iOS	31
2	-1de9sT-MLwVVvnC0ncCLnqEqpSi3XSN	iOS	16
3	-3A3L2jnM55B_Q1bRXMjZ6sPnINlj-Y1	And	41
4	-3bhcSgPOIdQAPkPNcchxvECGqGQQ78k	And	42

In [65]:
```
merged = order.merge(user, on='user_id')
merged.head()
```

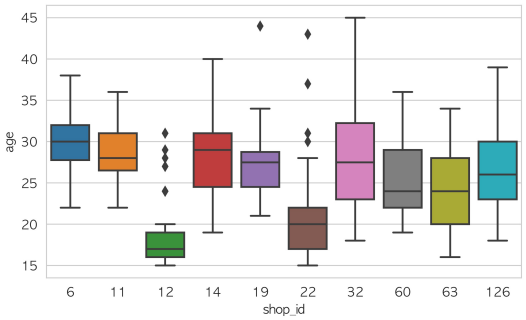
Out[65]:

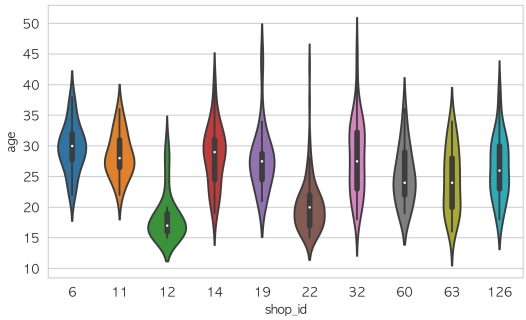
	timestamp	user_id	goods_id	shop_id	price	hour	os	age
0	2018-06-11 00:00:43.032	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	1414	38	45000	0	iOS	39
1	2018-06-11 00:02:33.763	smDmRnykg61KajpxXKzQ0oNkrh2nuSBj	1351	12	9500	0	And	17
2	2018-06-11 00:04:06.364	EyGjKYtSqZgqJ1ddKCtH5XwGirTyOH2P	646	14	22000	0	And	-1
3	2018-06-11 00:04:17.258	KQBGi33Zxh5Dgu0WEkOkjN0YqTT_wxC3	5901	46	29800	0	And	34
4	2018-06-11 00:05:26.010	lq1Je3voA3a0MouSFba3629lKCvwel24	5572	89	29000	0	And	17

7. 매출 Top 10 쇼핑몰 구매자들의 연령대를 쇼핑몰별로 시각화하여 보여주세요.

위의 병합된 테이블을 이용하여, 당일 매출 Top 10 쇼핑몰에서 구매를 한 고객들의 연령대 분포를 시각화로 표현하고자 합니다. 이를 이용해 쇼핑몰이 설정한 타겟 연령대와 실제 구매층이 일치하는지를 비교해보고자 합니다.

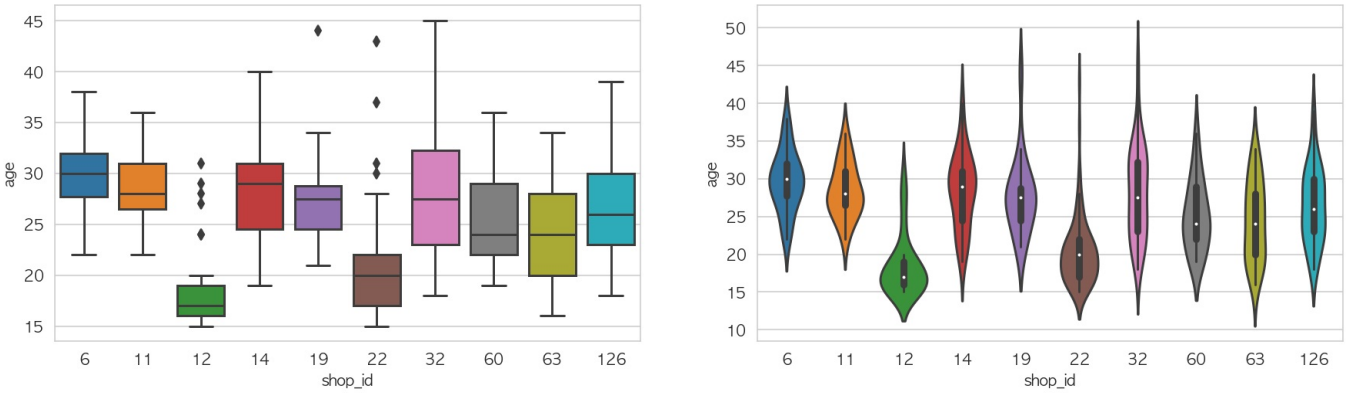
나이 정보가 없는 경우는 -1이 입력되어 있기 때문에 이를 처리한 다음 시각화를 해야 합니다. 시각화 결과가 아래의 둘 중 하나가 나오게 해주세요.





In [66]:
```
merged_top10 = merged[(merged['shop_id'].isin(top10_index)) & (merged['age'] != -1)]
```

In [67]:
```
fig, (ax1, ax2) = plt.subplots(ncols=2, nrows=1)
fig.set_size_inches([15,4])

sns.boxplot(data = merged_top10, x='shop_id', y='age', ax=ax1)
sns.violinplot(data = merged_top10, x='shop_id', y='age', ax=ax2)

fig.savefig('figure.png', dpi=400)
```



8. user 테이블에 연령대를 나타내는 칼럼을 만들어주세요. 그리고 쇼핑몰이 설정한 타겟 연령대와 실제로 구매를 한 고객의 연령과 일치하는지를 검증해주세요.

지그재그의 쇼핑몰들은 아래의 필터에서 보여지는 것과 같이 타겟 연령대를 가지고 있습니다. 하지만, 실제 구매가 설정되어 있는 타겟 연령대에 맞게 이루어지는지 꾸준히 검증이 이루어져야 합니다. 유저에게 더 적합한 제품이나 쇼핑몰을 추천해주어 유저 경험 (UX)를 증진시키는 것은 추천 플랫폼에게 매우 중요한 요소이기 때문입니다.

수행해야 할 작업은 총 3단계입니다.

1. 실제 나이를 바탕으로 user 테이블에 연령대 칼럼을 만들기
2. shop 테이블을 불러와 user, order 테이블과 병합하기
3. 쇼핑몰의 타겟 연령대와 해당 쇼핑몰에서의 결제를 한 고객의 연령대를 비교하기

아래의 함수를 이용해 user 테이블에 연령대를 만들어주세요.

In [68]:
```
def make_generation(age):
    if age == -1:
        return '미입력'
    elif age // 10 >= 4:
        return "30대 후반"
    elif age // 10 == 1:
        return "10대"
```

```
    elif age % 10 < 3:
        return str(age // 10 * 10) + f"대 초반"
    elif age % 10 <= 6:
        return str(age // 10 * 10) + f"대 중반"
    else:
        return str(age // 10 * 10) + f"대 후반"


print(make_generation(10))
print(make_generation(23))
print(make_generation(29))
print(make_generation(32))
print(make_generation(35))
print(make_generation(40))
```

```
10대
20대 중반
20대 후반
30대 초반
30대 중반
30대 후반
```

In [69]:

```
user["연령대"] = user['age'].map(make_generation)
user.head()
```

Out[69]:

	user_id	os	age	연령대
0	--PYPMX8QWg0ioT5zfORmU-S5Lln0lot	And	41	30대 후반
1	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	iOS	31	30대 초반
2	-1de9sT-MLwVVvnC0ncCLnqEqpSi3XSN	iOS	16	10대
3	-3A3L2jnM55B_Q1bRXMjZ6sPnINlj-Y1	And	41	30대 후반
4	-3bhcSgPOIdQAPkPNcchxvECGqGQQ78k	And	42	30대 후반

shop 테이블을 DB에서 불러와 주세요. 그 다음, user, order 테이블과 병합해주세요.

병합 결과는 다음과 같게 됩니다.

	timestamp	user_id	goods_id	shop_id	price	hour	os	age_x	연령대	name	category	age_y	style
0	2018-06-11 00:00:43.032	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	1414	38	45000	0	iOS	39	30대 후반	Mabel	의류	20대 후반/30대 초반/30대 중반	모던 시크/페미닌
1	2018-06-11 07:33:39.823	ni3NQK35j-YaSxli-C_Sz7ZmQqOwMljL	2278	38	37000	7	And	32	30대 초반	Mabel	의류	20대 후반/30대 초반/30대 중반	모던 시크/페미닌
2	2018-06-11 12:56:27.867	MnvhmV0tA89bN9TLXgRTbLza689bTkT9	5513	38	31000	12	And	37	30대 후반	Mabel	의류	20대 후반/30대 초반/30대 중반	모던 시크/페미닌
3	2018-06-11 22:57:11.582	3Vo9NP0qU_176pgbqk6Cu-CY7kpJ2-WB	7026	38	17100	22	iOS	34	30대 중반	Mabel	의류	20대 후반/30대 초반/30대 중반	모던 시크/페미닌
4	2018-06-11 00:02:33.763	smDmRnykg61KajpxXKzQ0oNkrh2nuSBj	1351	12	9500	0	And	17	10대 후반	Rachel	의류	10대/20대 초반	러블리/심플베이직

In [70]:

```
query = "SELECT * FROM 'shop'"
```

In [71]:

```
shop = pd.read_sql(query, connect, index_col='shop_id')
```

```
print(shop.shape)
shop.head()
```

```
(200, 4)
```

Out[71]:

shop_id	name	category	age	style
1	Edna	의류	20대 중반/20대 후반/30대 초반	모던시크/러블리
2	Pam	의류	20대 중반/20대 후반/30대 초반	러블리/심플베이직
3	Carolyn	의류	20대 중반/20대 후반/30대 초반	모던시크/심플베이직
4	Joan	의류	30대 초반/30대 중반	미시스타일/유니크
5	Florene	의류	20대 중반/20대 후반/30대 초반	심플베이직/헐리웃스타일

In [72]:

```
merged_table = (
order.merge(user, on='user_id')
    .merge(shop, on='shop_id')
)

merged_table.head()
```

Out[72]:

	timestamp	user_id	goods_id	shop_id	price	hour	os	age_x	연령대	name	category	age_y	style
0	2018-06-11 00:00:43.032	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	1414	38	45000	0	iOS	39	30대 후반	Mabel	의류	20대 후반/30대 초반/30대 중반	모던시크/페미닌
1	2018-06-11 07:33:39.823	ni3NQK35j-YaSxli-C_Sz7ZmQqOwMljL	2278	38	37000	7	And	32	30대 초반	Mabel	의류	20대 후반/30대 초반/30대 중반	모던시크/페미닌
2	2018-06-11 12:56:27.867	MnvhmV0tA89bN9TLXgRTbLza689bTkT9	5513	38	31000	12	And	37	30대 후반	Mabel	의류	20대 후반/30대 초반/30대 중반	모던시크/페미닌
3	2018-06-11 22:57:11.582	3Vo9NP0qU_176pgbqk6Cu-CY7kpJ2-WB	7026	38	17100	22	iOS	34	30대 중반	Mabel	의류	20대 후반/30대 초반/30대 중반	모던시크/페미닌
4	2018-06-11 00:02:33.763	smDmRnykg61KajpxXKzQ0oNkrh2nuSBj	1351	12	9500	0	And	17	10대	Rachel	의류	10대/20대 초반	러블리/심플베이직

병합한 테이블을 이용하여 '거래연령 일치여부' 칼럼을 아래의 칼럼을 이용하여 만들어 주세요. 각 열이 아닌 각 행에 함수를 적용할 때는 `apply(function, axis=1)` 을 이용해야 합니다.

In [73]:

```
def check_generation(row):
    if row['category'] == '의류' and row['연령대'] == '미입력':
        return True
    else:
        return row['연령대'] in str(row['age_y'])
```

피벗 테이블을 이용한 결과가 다음과 같이 나오게 됩니다.

	mean	count
	거래연령 일치여부	거래연령 일치여부
shop_id		
1	0.666667	3
2	0.937500	16

3	0.400000	5
4	1.000000	1
5	0.000000	1

In [74]:
```
merged_table['거래연령 일치여부'] = merged_table.apply(check_generation, axis=1)
merged_table.head(2)
```

Out[74]:

	timestamp	user_id	goods_id	shop_id	price	hour	os	age_x	연령대	name	category	age_y	style	거래연령 일치여부
0	2018-06-11 00:00:43.032	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	1414	38	45000	0	iOS	39	30대 후반	Mabel	의류	20대 후반/30대 초반/30대 중반	모던 시크/ 페미닌	False
1	2018-06-11 07:33:39.823	ni3NQK35j-YaSxli-C_Sz7ZmQqOwMljL	2278	38	37000	7	And	32	30대 초반	Mabel	의류	20대 후반/30대 초반/30대 중반	모던 시크/ 페미닌	True

In [75]:
```
table = merged_table.pivot_table(values='거래연령 일치여부',
                                 index='shop_id',
                                 aggfunc=['mean', 'count'])

table.head()
```

Out[75]:

	mean	count
	거래연령 일치여부	거래연령 일치여부
shop_id		
1	0.666667	3
2	0.937500	16
3	0.400000	5
4	1.000000	1
5	0.000000	1

위의 정보를 Top 10 쇼핑몰에 대해 뽑아보면 다음과 같은 결과가 나옵니다.

`table[table.index.isin(top10_index)]`

	mean	count
	거래연령 일치여부	거래연령 일치여부
shop_id		
6	0.750000	24
11	0.684211	19
12	0.857143	42
14	0.566667	30
19	0.789474	19
22	0.929293	99
32	0.540541	37
60	0.695652	23
63	0.000000	27
126	0.000000	39

의류이외의 제품을 파는 쇼핑몰은 타겟 연령층이 없기 때문에 일치여부가 0이 나옵니다. 일치여부가 낮은 쇼핑몰의 경우는 더 긴 기간의 로그를 모니터링 한 다음, 태그 수정을 제안하여 타겟 적합도를 높일 수 있습니다.

9. 쇼핑몰의 스타일 태그를 정리해주세요.

쇼핑몰별로 제품군의 스타일을 나타낼 수 있는 태그를 가지고 있습니다. 이 태그는 다음의 리스트에 정리되어 있습니다.

In [76]:
```
style_list = ['페미닌', '모던시크', '심플베이직', '러블리', '유니크', '미시스타일', '캠퍼스룩', '빈티지', '섹시글램', '스쿨룩
```

```
                '럭셔리', '헐리웃스타일', '심플시크', '키치', '펑키', '큐티', '볼드&에스닉' ]
```

위의 스타일을 정리하여 shop 테이블을 전처리해 아래와 같은 테이블을 만들어주세요.

shop_id	name	category	age	style	페미닌	모던시크	심플베이직	러블리	유니크	미시스타일	캠퍼스룩	빈티지	섹시글램	스쿨룩	로맨틱	오피스룩	럭셔리	헐리웃스타일	심플시크	키치
1	Edna	의류	20대 중반/20대 후반/30대 초반	모던시크/러블리	False	True	False	True	False	False	False	False	False	False	False	False	False	False	False	Fal
2	Pam	의류	20대 중반/20대 후반/30대 초반	러블리/심플베이직	False	False	True	True	False	False	False	False	False	False	False	False	False	False	False	Fal
3	Carolyn	의류	20대 중반/20대 후반/30대 초반	모던시크/심플베이직	False	True	True	False	False	False	False	False	False	False	False	False	False	False	False	Fal

예를 들어 shop_id가 1인 쇼핑몰의 스타일이 `모던시크/러블리` 인 경우 모던시크 칼럼과 러블리 칼럼은 True 값을 가지고 나머지 칼럼은 False의 값을 가집니다.

In [77]:
```
for style in style_list:
    shop[f"{style}"] = shop['style'].str.contains(style)
```

10. 스타일별 실제 구매 기록을 바탕으로 가장 구매가 많이 일어난 스타일 키워드를 찾아주세요. 또한, 매출이 가장 많은 3가지 스타일의 구매 연령대 분포를 그려주세요.

위에서 전처리한 스타일을 이용하기 위해 order, shop, user 테이블을 다시 테이블을 병합하도록 하겠습니다.

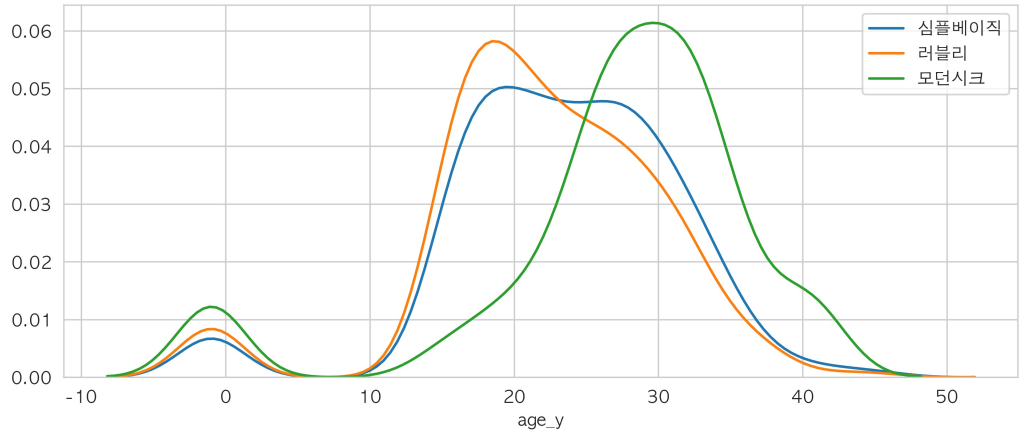
In [78]:
```
merged = (
    order.merge(shop, on='shop_id')
             .merge(user, on='user_id')
)
print(merged.shape)
merged.head(3)
```

```
(867, 32)
```

Out[78]:

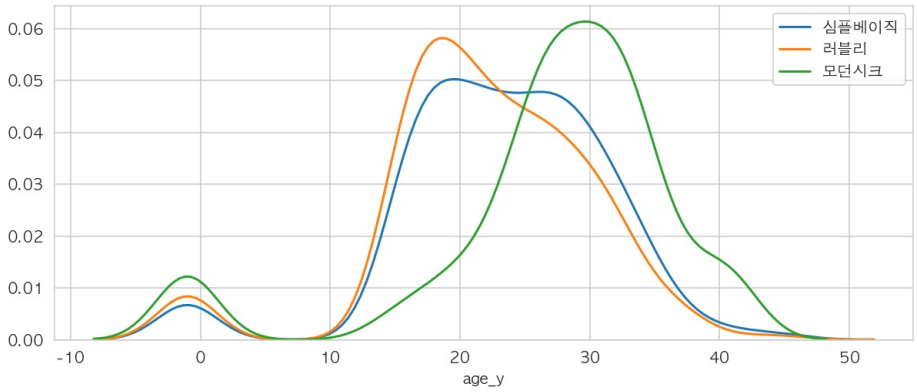
	timestamp	user_id	goods_id	shop_id	price	hour	name	category	age_x	style	페미닌	모던시크	심플베이직	러블리
0	2018-06-11 00:00:43.032	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	1414	38	45000	0	Mabel	의류	20대 후반/30대 초반/30대 중반	모던시크/페미닌	True	True	False	False
1	2018-06-11 07:33:39.823	ni3NQK35j-YaSxli-C_Sz7ZmQqOwMljL	2278	38	37000	7	Mabel	의류	20대 후반/30대 초반/30대 중반	모던시크/페미닌	True	True	False	False
2	2018-06-11 12:56:27.867	MnvhmV0tA89bN9TLXgRTbLza689bTkT9	5513	38	31000	12	Mabel	의류	20대 후반/30대 초반/30대 중반	모던시크/페미닌	True	True	False	False

위의 merged table을 이용하여 다음과 같은 그래프가 나오게 해주세요.



In [79]:

```
plt.figure(figsize=[10,4])

# sns.distplot(merged.loc[merged['심플베이직'], 'age_y'], label = '심플베이직', hist=False)
# sns.distplot(merged.loc[merged['러블리'], 'age_y'], label = '러블리', hist=False)
# sns.distplot(merged.loc[merged['모던시크'], 'age_y'], label = '모던시크', hist=False)

sns.distplot(merged.loc[merged['심플베이직'] == True, 'age_y'], label = '심플베이직', hist=False)
sns.distplot(merged.loc[merged['러블리'] == True, 'age_y'], label = '러블리', hist=False)
sns.distplot(merged.loc[merged['모던시크'] == True, 'age_y'], label = '모던시크', hist=False)

plt.savefig('dist.png', dpi=400)
```



수고하셨습니다!

여기서 부터는 로그 데이터분석에 관한 내용입니다.

아래의 내용은 로그 데이터분석의 핵심 수치들을 구하는 것과 이를 위한 전처리로 이루어져 있습니다. 핵심 수치들은 다음과 같습니다.

1. page duration
   - 사용자가 앱의 한 page당 체류하는 시간입니다.
2. session
   - 앱을 실행하는 단위로서, 세션은 사용자가 앱을 실행한 후부터 그 실행을 마칠 때까지의 일련의 과정을 포함합니다.
3. 체류 시간
   - 사용자가 page, session 혹은 특정 기준동안 머무르는 시간입니다.

`[assignment]` 에서는 각 용어를 "log duration", "cycle", "잔존 시간"이란 표현을 사용하였지만, 앞으로는 로그 분석에서 보편적으로 사용하는 용어인 위의 용어들로 사용하겠습니다.

또한 앞으론 pandas의 groupby()를 매우 중요하게 다루게 됩니다. 아래의 링크에서 groupby() 메서드에 대해서 참고하시면 도움이 될 것 입니다.

[https://pandas.pydata.org/pandas-docs/stable/groupby.html]

10. DB에서 로그 데이터를 불러와주신 다음 timestamp컬럼을 datetime 형식으로 변환해 주세요. 그리고 user id를 보기 쉽게 간단한 자연수 형태로 변환해주세요.

In [80]:
```
query = "SELECT * FROM 'log'"

data_logs = pd.read_sql(query, connect)
print(data_logs.shape)
data_logs.head()
```

```
(105815, 6)
```

Out[80]:

	timestamp	user_id	event_origin	event_name	event_goods_id	event_shop_id
0	2018-06-11 00:00:00.213	K1d8_t3-QIskaSkrx32oAFu856D8JmLo	shops_ranking	app_page_view	NaN	NaN
1	2018-06-11 00:00:00.810	lwFZ77v_ygk0uU40t1ud3l30EZ6sE2R3	shops_bookmark	app_page_view	NaN	NaN
2	2018-06-11 00:00:00.956	mR-bO6hC9g-m8ERXMRQZaRwJFvzNNdd8	goods_search_result/로브	app_page_view	NaN	NaN
3	2018-06-11 00:00:01.084	K1d8_t3-QIskaSkrx32oAFu856D8JmLo	shops_bookmark	app_page_view	NaN	NaN
4	2018-06-11 00:00:01.561	Yjny5AchUWLiuv4kdeq50COF-S8OFXPd	shops_bookmark	app_page_view	NaN	NaN

In [81]:
```
# timestamp 컬럼를 datetime 타입으로 변환해주세요.

data_logs['timestamp'] = pd.to_datetime(data_logs['timestamp'])
```

지그재그 로그 데이터의 명세는 다음과 같습니다.

- 컬럼 별 명세

```
1. timestamp : 이벤트 발생 시간 (한국 시간 기준)
2. user_id : 이용자 고유 식별자
3. event_origin : 이벤트가 발생한 앱 위치
    - event_origin 값 별 의미
        a. goods_search_result : 특정 검색어의 상품 검색 결과
            (Ex: goods_search_result/반팔티)
        b. shops_ranking : '쇼핑몰 랭킹' 영역
        c. shops_bookmark : '즐겨찾기' 영역
        d. category_search_result : 카테고리 검색 결과
            (Ex:category_search_result/상의)
        e. my_goods : '내 상품' 영역

4. event_name : 발생한 이벤트 명
    - event_name 값 별 의미
        a. app_page_view : 앱 내 화면 이동
        b. enter_browser : 앱 내 클릭을 통해, 특정 웹페이지로 진입
        c. add_bookmark : 특정 쇼핑몰을 즐겨찾기 추가
        d. remove_bookmark : 특정 쇼핑몰을 즐겨찾기 제거
        e. add_my_goods : 특정 상품을 내 상품 추가
        f. remove_my_goods : 특정 상품을 내 상품 제거

5. event_goods_id : 이벤트가 발생한 상품 고유 식별자
     - 상품 관련 이벤트가 아닌 경우, 공백

6. event_shop_id : 이벤트가 발생한 쇼핑몰 고유 식별자
     - 쇼핑몰 관련 이벤트가 아닌 경우, 공백
```

user_id는 아래에서 확인할 수 있듯이 매우 복잡한 형태로 되어 있어 한눈에 파악하기 어렵습니다.

	user_id	os	age
0	--PYPMX8QWg0ioT5zfORmU-S5LIn0lot	And	41
1	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	iOS	31
2	-1de9sT-MLwVVvnC0ncCLnqEqpSi3XSN	iOS	16
3	-3A3L2jnM55B_Q1bRXMjZ6sPnINIj-Y1	And	41
4	-3bhcSgPOIdQAPkPNcchxvECGqGQQ78k	And	42

앞으로의 분석을 용이하게 하기위하여 user_id을 간단하게 0, 1, 2, 3 ...과 같이 연속된 정수 형태로 변환하여 아래와 같이 만들겠습니다.

	user_id	n_user_id
0	K1d8_t3-QIskaSkrx32oAFu856D8JmLo	3314
1	lwFZ77v_ygk0uU40t1ud3l30EZ6sE2R3	7844

2	mR-bO6hC9g-m8ERXMRQZaRwJFvzNNdd8	7920
3	K1d8_t3-QIskaSkrx32oAFu856D8JmLo	3314
4	Yjny5AchUWLiuv4kdeq50COF-S8OFXPd	5608
5	LZZ0ktGq6hW685TFAQfcGNhsKVUEceHI	3548
6	TUoAGIbbNds5cYLZLnz-R5VlkG5L8RuZ	4790
7	B9F_BHH9F3b6MW329go9jDr71Uunx629	1902
8	e_xrTZ9fHVodxxadLx688qUKMWCdL8bW	6663
9	aA9S7LxnFm6ym6IUEa-4SSxJa-iL5m2J	5976

기존의 고객 아이디와 새롭게 만들 고객 아이디를 짝지어 딕셔너리로 만들고 이를 기존의 고객 아이디에 mapping하여 진행을 할 것입니다.

In [82]:
```
# 판다스의 unique() 기능을 이용하여 유저 아이디를 user_id라는 변수에 저장합니다.

user_id = user['user_id'].unique()

user_id
```

Out[82]:
```
array(['--PYPMX8QWg0ioT5zfORmU-S5Lln0lot',
       '-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv',
       '-1de9sT-MLwVVvnC0ncCLnqEqpSi3XSN', ...,
       'zz-aNy7UWfvyrZxO4Fs4K5ewmqZVaMOs',
       'zznj-LHhddVvuzZmbZpw6MSylLO64982',
       'zzxBQ7i7mttX0cv1GqFuuMstg7keEkdV'], dtype=object)
```

In [83]:
```
#새로운 user_id는 연속된 자연수들로 지정합니다. range()를 사용하여 user_id의 개수만큼의 연속된 정수를 만듭니다.

n_user_id = range(user['user_id'].size)

n_user_id
```

Out[83]:
```
range(0, 10000)
```

In [84]:
```
#python 내장 함수인 zip()을 이용하여 기존의 id와 새로운 id를 짝지어 묶습니다.
id_zip = zip(user_id, n_user_id)

#id_zip을 출력하면 아래 결과와 같이 zip object이 출력됩니다.
print(id_zip)
```

```
<zip object at 0x11847dcc8>
```

In [85]:
```
#zip()의 결과를 구체적으로 보기 위하여 list로 변환하여 print하겠습니다. 상위 5개만 출력합니다.
list(zip(user_id, n_user_id))[:5]
```

Out[85]:
```
[('--PYPMX8QWg0ioT5zfORmU-S5Lln0lot', 0),
 ('-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv', 1),
 ('-1de9sT-MLwVVvnC0ncCLnqEqpSi3XSN', 2),
 ('-3A3L2jnM55B_Q1bRXMjZ6sPnINIj-Y1', 3),
 ('-3bhcSgPOIdQAPkPNcchxvECGqGQQ78k', 4)]
```

In [86]:
```
#새로운 id와 기존의 id가 대응된 딕셔너리 타입 변수를 생성합니다.
id_dict = dict(id_zip)

id_dict
```

Out[86]:
```
{'--PYPMX8QWg0ioT5zfORmU-S5Lln0lot': 0,
 '-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv': 1,
 '-1de9sT-MLwVVvnC0ncCLnqEqpSi3XSN': 2,
 '-3A3L2jnM55B_Q1bRXMjZ6sPnINIj-Y1': 3,
 '-3bhcSgPOIdQAPkPNcchxvECGqGQQ78k': 4,
 '-3fmY1WsLkYJwN_8lZQMmxZd6zJTAcT1': 5,
 '-3q-oynqxFEgSHUwX802hpmi1louyQNv': 6,
 '-428TMckUlhn6ptxN7gR2FGaSyXjSnaD': 7,
 '-4O8WnD8dT6nWho-4KbIm6TvnK4BmjX_': 8,
 '-4ltLPS55n6J2wSUCLxEZwxYdeW37cK5': 9,
 '-5BA0EwkyhGLCC8FxzvvDgyrZWYJM33I': 10,
 '-5Cwn2Fcx9j16QSM2-SLiaLMm0sS4E2I': 11,
```

```
'-5o3lkvJctT3uURb5JWPVxe1VjqhyzAi': 12,
'-622WUNWBtjX5VGKx8UnOtn2NVHD_NaB': 13,
'-62U2A3KHjNZ2XXmOgQTSWEfPg1RRWWy': 14,
'-63J8veARgGL3ulnRKblm4xhhwkvjKzG': 15,
'-6UZWGgl3AAI7Df2sVWLX6oT6zP43zo0': 16,
'-6jxyh56lSivkbLm3WNGRCmdyrdsBmNW': 17,
'-71z4lG_D-eKnOmDCJlUaNvVcwd808yw': 18,
'-75tFsDSoUwapUvwCUHTZiGTGkaSDleQ': 19,
'-7SKUZkBmbG2ZMvJ0E0jmMDcd8PgmARb': 20,
'-7uBbvfy4gff6mHV9XotjVO2YlCY2r8v': 21,
'-8htVW7UIA8qRupSdCx-6PzIXLI_vk2p': 22,
'-9qbSavSdufdw9JwmiWX1_URT2E2QxFZ': 23,
'-Ae6T8G5uAZldwUEOTMR-KzG8C8G6SYn': 24,
'-BvO7h0cREi0zszH5EhugLlvfnXmK7le': 25,
'-CHArPAojNa8sfK1Ni0WCyQvtF9rlujb': 26,
'-CSzbGJ06nTB7UJ8CSsU-IVtn4apPUwX': 27,
'-DD4QE4Kp4i7tYalqeGSmJyAtfWLcYY7': 28,
'-DM7pBgq8_K3F5ttDYENsUzf8EMYx1kR': 29,
'-DhBA7c5OP3gFvBkng5b2jHdNYGDF3at': 30,
'-DmelscKWaGOgODpbk5udtReEUvRmY25': 31,
'-Dt4wPzvVjzno0dgjUAhs_BtbniR8fyU': 32,
'-Dtg2_c38T2WqcxvzxNoQOKjwV_WXW3U': 33,
'-EBvKufhJMmB-uL4Gb8up_dZubfazQrN': 34,
'-EpdIEklTRF8V0wFNcJ1hTriLDKGgY7h': 35,
'-EzeqHwqCE5w2DcivhEZT6OcjzIe-BnO': 36,
'-G8e4-zH6pTty3OzEgDWpMDxzaQw0Inq': 37,
'-GCtw39Q5SUAkgnbzaVsWxxYUyvXYESf': 38,
'-GN_i-M2FVR_OC7ERqsbuXsHT5WHT_19': 39,
'-HrAJSQTzq7A2a7if0CARcpR-MPQNtjM': 40,
'-IP3FwhEXrqR1OUDjicLNuZD0pXNhAwh': 41,
'-IeEef_WmYqN9g87TP-XY_cBDExtMnYM': 42,
'-IfjJ89djHFZi9rTYHcE_AJqfwdJbqYP': 43,
'-Ii7NyvhizIwHlbCvi4ovYoB3WOxVR6e': 44,
'-J1-YI15bmfHqz706XL-t2DELNae3BK4': 45,
'-JrptE-k2qGZTgqoWgKQ2CkDI3e2i--i': 46,
'-K76uxrcXqUDULcH-OKfoWxfWrH7-bYc': 47,
'-KAPzobWDxGY7bCfCSfgeWBH8uzDXS8n': 48,
'-L2Awbp23c9b1o1R_do--BZEtPivAUua': 49,
'-LF9kd_LkqLCKsyRECBwDnToEIoZpT4F': 50,
'-LHdmkfa3uKwlpRTaOpARhyNHqcWAIFI': 51,
'-LJW8D3rOJs-cTcSsspvuSUh4pKDUdh_': 52,
'-MhXFZWekBnvZ-Bj4lbt_Q5rDbwOAJIc': 53,
'-N19jslUgVdZgFhTsbt-Qa8-2avlWjJQ': 54,
'-NHyDXqV106lyjbh4e3e5JXoIqEYzxpZ': 55,
'-O78CyLDmPILHvfRINWC7GtxZ6C8XEFd': 56,
'-O8132_YYdBFHlNUbLWQpeIKGMdX_mjI': 57,
'-OAaalRdrKcy2JnQuUktMTivUkhzhGqL': 58,
'-OjNBoKeSseQqxcl7hC7LofbZAxkM7h7': 59,
'-Q2zMYOSfxJzAT3bCmpb-mm2pBCL9_Hd': 60,
'-Qhp1QtAa66pHomNmhUl1qkksPW-Jmin': 61,
'-QmN2AFoHc97kjxWy28HF9flCsfce9hg': 62,
'-QmYIj-H47FsR_I8SD0qIxC7RACgcpFU': 63,
'-RX5mi3RbwWxrZX05aE9Oqujjc8v0sDI': 64,
'-RkcyYiat4mnNUK48f_0xL1rWJC1e5Sy': 65,
'-SXxoXdABrYMQnFPoS7RUV4grtN_-20j': 66,
'-SnTp0qfMPhA5YoODFQZHTCI2m1BJATt': 67,
'-Sp5JstbFxVFSRTFEZ00pVyADBzt6h8d': 68,
'-TorTFJx8gRSr7xTxVwumR0Rk0ohtsoK': 69,
'-U-JT5YB3QE68MYwRy0BYeJMT6A2AUTs': 70,
'-UGZrYZ4hoGNyUEawyoI0Q4UZXW9TNe2': 71,
'-UiAEis3zGS38VgdUczHp3tGrypLp83k': 72,
'-UnJPVKthwrXlYyXwvZe8B_eISA4ONLK': 73,
'-VHehO1XVrDe_wHR1QEuOEVeeG9PgDH7': 74,
'-VI_fTFPd4n1Iu6LAcF1uKDdhYKgqK44': 75,
'-WxYaD63mK1pn_T0A_EjiTs4f5VYsLoM': 76,
'-XRBHrHjtKKB8pubuJ309ZTe8AjLQcaU': 77,
'-ZsgqTZl3KSd_oXnl8N5XNKNk60JWPJa': 78,
'-ZxtGvoUVaNpSPcSf-2FfDGJTcmnceK_': 79,
'-_8SvCI-GlKGtaoS2E0PNSQT3ampaHdr': 80,
'-__V4fapuZu9JrEXk3S5B0V5Uftme2so': 81,
'-ayuG0oBDd5HxFZ6g_59IN6s8jc5JoLR': 82,
'-b0w_nXB4TLZQzp0u7vR7eoxvp0qreVt': 83,
'-b3xSGJo8dxPFgkGoP1fF4OKQe47WTQ3': 84,
'-bAmsDSUs_GwVs2sBRRiFtDpclgqnTWp': 85,
'-bW8vR_yKhs40MmASelnYoyY3daCIMkW': 86,
'-bpKDs9gKdOVd33-yIwyia68QJXlQA0J': 87,
'-by1c_0QC-R8GN0ZhZeV8697f3Awzagi': 88,
'-c59nvb2SpJP-fVBqMkV5nOCI3ySQDWs': 89,
'-cNzffxv2Egd3c3COdtI2mDrp8ypWv0z': 90,
'-cckGqSmzK_SQvzueqkrnQzJ6OgD6IAZ': 91,
'-cm3QsSxiloyb5kpVV1gka38-dTWzzSp': 92,
'-cmO1MWB0jSPkf836eBLRGmRWQAO-QtQ': 93,
'-czlIBWDQG0fOX-m6gPh9Rl82FzqOu_d': 94,
```

```
'-d0OtLmJDVgGxab_DNkvvQZgVG0X1Xux': 95,
'-dmc0sa0gCvJ5KmqkVKZT8Jb5tfRpkjw': 96,
'-eGH0rie-miqE6oRMLTjyCkZhUWJh_QM': 97,
'-eRC4daHqu5n662d-Y3a8qjtVx1pOttI': 98,
'-ed4eHabeosfJaBWnsZxgXRrBAOETXYK': 99,
'-egWwe3isl7rtKJDxP8ozKC_dzZUhumA': 100,
'-ekX42BJpQaWE8_19gyjU0of98E5_N9B': 101,
'-etFrkS-ghB79NyWTUCL92dCvEGXCD-6': 102,
'-fK48I4C6gjWI3F1xBffm4303nL_-VOU': 103,
'-fNm3fnrElUu7nF-OwtVxA7fY8ijU0aJ': 104,
'-fSOxMY6izO8LrY3s5oyaHVmhk7hpprL': 105,
'-fp540bwSD_DJp-xbwjn395KoX66v6YT': 106,
'-gACDk5ovMXy6DclIVCPAdjQOqjGq_1l': 107,
'-gycbv-w_Yfm3QmnQJUOB-KRQJdXhan1': 108,
'-h8JdR6KGdu_BQeAvIxDtqNCRqG3Zkk4': 109,
'-hXRmy_ILzSgXkpzCYKfB5GFDPOtV7ys': 110,
'-icMk-sfTgtBE1aExGorSxYo2mntelyj': 111,
'-iezn-WitSN0k6W55Lss8SiLcqxZ-PRD': 112,
'-if4yvtFMDPrtC578OMJ5gta-JNM-VaL': 113,
'-iyXTao7hdCDWujpJ_lqoBW0K6-JXCRL': 114,
'-k4EuKu1x29GF9gqHMrAazkd9rrTfZxu': 115,
'-lEUMUFkTARejo2wQaqRWVnm24Jg98rA': 116,
'-mZEEW22aqNU6B794YSve6VB0UOFGCwA': 117,
'-m_o_fjLt2dumaTA3hGFfWDWc5XVFYbh': 118,
'-mfvXk0bpG9FF9BxhtsPO2oIPbLzpPtM': 119,
'-mzE8kJNOw_XrQ73bQDSa0wWQRGZEDIg': 120,
'-nE3YSLqNuTm_mtrZ5L9BwrbypQdXVcf': 121,
'-nUpZjqKLOzZyBDfw5JEa3YV1X9I9CGH': 122,
'-ntZw9gYagAhaZJ6jDNcFLorwZ4bzTUr': 123,
'-o7kzi5qmIZfaIIEd5NGcAzZHV_9QRgH': 124,
'-oQDz92wu03AZ2-D1irCumssCXQpN-5X': 125,
'-or848YnAnSJGWOVYW1G68RlyL5F4MnZ': 126,
'-q1k5EBWQpvt3doDD_yPffzNHxAc4ZOE': 127,
'-q4JWL7qLn0v4G7sF2zu2qk6oKZX3q8-': 128,
'-qJDt4SYPRFW4VJ4Vkicp6Uq8Maex4s1': 129,
'-qSFJnPE73ecwpMi6IEMY-h8JvC3l6-g': 130,
'-qi0FQUqpRT_HTyqUJkh6xOX8s_nJ2Ig': 131,
'-rX2fqH11nFsbjhz85EE97M1TDBPt6L0': 132,
'-rXiq_hr4k7Ahc08xOeY0pBBZiowt0A6': 133,
'-rca_qeVKg_LY2-qzo9cSd_zJGWlpDSj': 134,
'-s5PfPAiXs-afbkx18hh5W5PgO0qBXGj': 135,
'-sMkdXHIJZNnQcDVLmXgtvViedasPi1x': 136,
'-sgzu6doO3Y_3K6NRTbBxKBrea12eboi': 137,
'-t2VveCySpPTSKUSl9-pMzo3u62Ggit-': 138,
'-tW5GAv8KvgKrFo2-Z19DWlPcLk8TTTF': 139,
'-tzA6cOZvW5Efu0xpECy40CowqfkPwO8': 140,
'-tzXKYtM-zPSNz8FuSe3S00plYhVss2s': 141,
'-uP3hBXAvdArCk1kddfJEMJpLDB-7wkg': 142,
'-uWVJ7qhriUBwmcXPp1J6hiaPnJgzo-k': 143,
'-vi0zV05ls8IREioBZFot7kEfW07mzzZ': 144,
'-w6Hn7to-3UNAYeCgfZz65QY4BG1Qpm3': 145,
'-wcrIwgwRiN9I748jpQUPFcSLq183rx_': 146,
'-x35t_xG2_8rLJydgJ7uog3IlE1__M6U': 147,
'-x5yC2fNbmZ1y1TZUQG8-BiqZao_Q_UN': 148,
'-x6YjF3A7Tm_cOG2S689FT3vgjWVbJaO': 149,
'-xUJVucG0yeNyx0jos2Vpt7vjR4tVG_D': 150,
'-xY1NMPEmDc7HPv7Ar7-HhCV6rkFNbn5': 151,
'-yHKGC8ESwJbi6ysS4cMoBCAjhmvymvV': 152,
'-yYxdTxSPvod6MHBdrpRf-4eUFVLy80Q': 153,
'-yh-CtwaVROnUvKvhEV4-pwmxkU1dYUb': 154,
'-ytNEbQLZTfvkGpycZ0_j1t4Y8Glgv3H': 155,
'-zU17fYsoGBmwr3198kSvz0Qc4K3Ht61': 156,
'-zYXYlWkMTgpQOQYuk4TpXpKqUKilB9n': 157,
'0-gltdZTK2hQMBKsXc1uzlWe3SsWFgma': 158,
'00Eh8XFntJ9gWn4xFj8U_tDbk6kO0eI0': 159,
'00X9QSX5rgLtmdgNSPhg6OSO25l53OnS': 160,
'00gZAUnFn0Ask_plQ9OG6eN0Ze2EEoOL': 161,
'014pMyWVzj4zAYL4OlGGroRL5tdKNPhv': 162,
'01KUOJqNtZUrItHzjf1pgo5UFMswmFg-': 163,
'01Nj5u5R5nIhoUZVSG2WA8hy5dM2pG3p': 164,
'01XiiHWG6ex33qwoaUO6m3OLdcxI52OA': 165,
'01grFSp_ulX9ezy6CfeWesfYrj3sm7v0': 166,
'01m8FY2yeGk9raQDP_FKPXnqWCrZY0Y5': 167,
'01oEO-_tOKpAiMkUAJwEreqD3qTgSU3U': 168,
'024d801eT2x_sk-1hS8tqUIZJQgoffxL': 169,
'02D_k1dBXOaRksPQzwx4WOviMRsTnaVu': 170,
'02H7ZMqxlxiWdrj3CMNvgxIwkD8nqmXJ': 171,
'02iSxgOe4uNFpOBo4xBYQ3ZnkYk9c9iL': 172,
'031IrKuZ-JWtE242Z5V4kk9xC0gs089g': 173,
'034N8bUP-LDPaIrgDyzHLLhrtyn4QrYD': 174,
'03NMvIIU-VSut44TktPL1eQ2U2GEQ2vT': 175,
'03ZPQuOqobYFqRdE5LZgcDrAOV9b7BCa': 176,
'04-4OZ0ymZ_-_FAVcOTAZCfs2rdvnVoS': 177,
```

```
'04iVDKAyu75QU3KlclcU2sFBtIkjK4Y_': 178,
'05Nnjm26REx1v0ZiA9Ope9BMk9FbVwFL': 179,
'05hXopf36nfEF7E76ga3BTZ5rv0YsoCQ': 180,
'069VNqpO_kh7D_Cd9qww-B5CalJzLpGL': 181,
'06KK25MxvBiv3SkYlX_UqVA7TWS-OwSM': 182,
'06OK0BrZPe9kam_jNNs91aU5D5GVOTxu': 183,
'06PPERX_ghAqv2sBqzuu3f0EGQvxH485': 184,
'07i5CvcIsrFGamyYUQn_AvP0tX_AENHT': 185,
'07spgxiiUETUmIocoG-8ufYW-hEx7uFz': 186,
'07ulvZ4sDElozBKCRxrY2hSE9NmK4XQf': 187,
'08KSaST5YdgsbKomkja6LKzO4_upevtO': 188,
'08Kdc6Yynsn4cYY8CfZmeHUimCt4BpVW': 189,
'08Vj8RCzTySLKA9xchuy14Dwt6-c74mh': 190,
'0A8OQkB3vtVZfrR_va9GuohGwnOC3bz8': 191,
'0APXGmH7Ok52R80Dk6Pmvt8Wj_YGQ0AB': 192,
'0AXScEXtOR6BZ2LoCJPflPR1EFBXCwwx': 193,
'0AczcUWa3arslb53eXOTJPUYzE7-xi2N': 194,
'0BDgEfQJwF0cudqtYkzIKniWA3D9-iwU': 195,
'0BDwr2zgQfJiNYehRGhPH7Whpu1tVCYE': 196,
'0C7ChjFwTnR9xHyIAR1bhdCDzd6CbLHZ': 197,
'0DMjVnSXpASdqQQaQwopEkGaIReiLCUa': 198,
'0DQvdfu6eAQUja9cBpLE7zKdjdtmAm62': 199,
'0DT8_c1omY8k-rkbEto46XXpCyh1JYT2': 200,
'0DXsYrJ-ffcI8ApKXHb03ibFQmqVKYhv': 201,
'0DmK3YpGHiIqjGLGeLcuPxuFVUFNBXNm': 202,
'0FYscHJu7mSgwCQVTsFTLUHnlIK4TQWh': 203,
'0FlD3jD1b9mDW5c0Rbes3LPWVgPZyvH_': 204,
'0FnBTJU0QJFDAOj9TGsKtea3Zkplzia3': 205,
'0Fx5LbfOeqs9wtsb8HQJhJgrGcnDlAf8': 206,
'0Gb0YtIWqOts5NJrVduPw-QDukGcF0zg': 207,
'0GmiHxmWRI9yWtGgZNDsZfYGWVXlvkJL': 208,
'0GuqiwNd8_1TBrw8ig0LzqY6jO61lP9C': 209,
'0HxrF9vnFhT-1HREcMRGTfgKb0gWFzxo': 210,
'0IVkO10P9oorsf_yUHFrXt6fPcrqiwYe': 211,
'0IWhGUJc8g7XK3MtzeqcRHTIot4EwO1Y': 212,
'0IczEJlRXPQ7rWVs6SU9wS10k41Etn2t': 213,
'0Is1QmxVRf6SzJi6Qm6H5-21QIlC6Ek5': 214,
'0JKhLtIx-NkFM440Adbg1wVi4JXzCylu': 215,
'0JXPzy9rbTMO2974aUNNxjfHyVYRSQTz': 216,
'0KU_nrrjmsoYBkDKTxMwb35DmdVKqUQd': 217,
'0L8x9JBvv76aaw6vfpu2pF4zpv_ysRp-': 218,
'0MLy1VJJ9QwY3mqZaSQKOBugwAnbJZVH': 219,
'0Mo4DHeFEi9ox7cENQBE8v90K4gZ-pjS': 220,
'0NaqAt2fjN-JhPRX79Btu0V-P4UhtwEq': 221,
'0NunOZYHrDHIQBSTzia2t2ujljWeFIdU': 222,
'0O5UkP2PjckTrtmiI5T_vq9xpjgbCKXG': 223,
'0OlMbb3hrJw8eCgJYkVDYrEYIezR9JbF': 224,
'0Q4ALyHqMT9h5UyAghB-GGLKoPSWlIdX': 225,
'0QDaEav42T34tlNXPni3kzqr3Jy8R-ac': 226,
'0RMN5gHlIFemGGvgXesN1DWfxXj4ce6B': 227,
'0Rh5zykltVmn_tRl6QmxBgbRcDgsPJxW': 228,
'0Rk8GxhkCGSiOy8erVRvEITaXkkRgXlq': 229,
'0RsDTwKEYNt8tlWoNk1kJi8H7yFpK8ll': 230,
'0SGJPS4QooPt1MV-CIp9aKYlUIwNIqnW': 231,
'0Shc3mYNv6HapWlAL8C5nb_O8XEZy-lI': 232,
'0SrUbI4m_nOPyTzC19KHNFn_jRTeGHXh': 233,
'0TFp43Z5GtmPIvFAjJlJ4VBYRyU-XV_k': 234,
'0Ui4C1Al9g4w_RWzUaNMqxljuQnWqrQ-': 235,
'0V97gBVq8-sVO1sfVQuVwhYduz56fABA': 236,
'0VQoPKljij0vNzMRxiyiRsiH_H0YBmAs': 237,
'0VqIpVCjNZTUKf4diGRj0daT8YVdOBQk': 238,
'0W43jLHSMhsaH2ggCfrp5KmWEP4ZlqiD': 239,
'0WE-8x36fShcC3VpQ3FmeI7W-Tu11F4U': 240,
'0XDOdDmxiNj2f-EDRtinS_vXNXC8aYPp': 241,
'0XFToWYIDpGw813B_7xv6OhrSALXxxdy': 242,
'0XGmnerLJmMrXpCiBE_tjLU8cfc4dctz': 243,
'0XPWqMdoFqWKHcQhnB7NaO9k5zvmCNvx': 244,
'0YNZe4EjjOPQkcRl0_aXgUsalLFPkrCh': 245,
'0Z0KgIwqfdku8UsKE_ou2gjEyhBR7fQ1': 246,
'0Z6h_ap4-9HiXGx9pqzQu8BL75BRDuzH': 247,
'0ZECVOXqREDC_GkM26ZI4UlTVqNrvW3q': 248,
'0Zf0GUoHH_jUkTzi3lYUzmIL1DsjuMsm': 249,
'0ZpjflYqN2U5OXM_wLF_1tUYz2uJDcya': 250,
'0_Cn3oC0Kh-iNUpa1tPOFPnQE4_sKNir': 251,
'0_bjfcGmMkwQAa5OOJlZclavXm96UL4v': 252,
'0a05D4dY0tbwmtnuuOE5jeuIpoMpDyvQ': 253,
'0aotjYOyR7ko18AswfRHroY-zdf8pv_n': 254,
'0bHdH2xtnsZG_OWAazx6qpEY7GCVLGHw': 255,
'0cCZgeF8UnJZuG2-B4RfLS98Nrq2lmyp': 256,
'0cr6hffVAdoCRVHw055FGOAgcgwZdus3': 257,
'0dS3hsANFXxP6_gu1UxK77Nw8oa9XJsg': 258,
'0dr-gYBcJ3l27jiq8hexxkZEHGIZx3yM': 259,
'0eHNCO93JYE9haSP9xga_BqFDs1hFZHU': 260,
```

```
'0eXufTpwvtTJw_LCsI-d1eCQwOYenX-j': 261,
'0fOZKkH3DyccewemxeiUX2PVlodRrw0E': 262,
'0fT221loKrCKHBjnuqHy3C5E16P6Sati': 263,
'0fb3gjWcIFEZhmXxppsgceIfQPCyBRll': 264,
'0h770WXoLOIoR35Q4fICQHfn77GJK0vq': 265,
'0hLk22zcupwQQNQeOCNtaZ3Mp6L1DcHY': 266,
'0i5UOrJ_Czfntj933oFJZvXjcwiohrSK': 267,
'0i7yulFZRdYYDkaXjacFPBxmHWzQA2ci': 268,
'0iB3Q2iNsefqk0WbFCgLn6SsuDRA7fKk': 269,
'0iGJ6o5tisw1QIvxL8EvpPqkCfRzqJpS': 270,
'0iksS9880WhuXnPhaxIxTzdGhOk9bSke': 271,
'0irHusHHH_OFwwZ-o0a8CV8-0dgtfa7I': 272,
'0iwz9g--onT-efA6XkkHUZMZuS_CJzfK': 273,
'0jDJEHkM5qzpJp1dH8f_vl1pt3OagluS': 274,
'0kGYO7tlwhIt-L14QDcHKbTIg4UfqJCD': 275,
'0kQZr5aGOvPtUXhy8CdCw6K_-ZN0JQR6': 276,
'0kT_MaV5iDk0x0HhSx4O3sMC1BWgvP9c': 277,
'0kWYTXVESpS19o-n0orNiH80G2YeAm4f': 278,
'0kyZPjj-YgjoAWzoWA66Zbas1KwjWX1V': 279,
'0lBszkbz9kovzQT9kFqxmdF3dZgBySl6': 280,
'0lqhixdLVBd-OwnAmDXYYFUTuYXbHrNX': 281,
'0lskkqdtaMhWq6eRzo6C_sXCNUL8yYIg': 282,
'0m93Pf39llcsraknvNOydN7i1ZDaussh': 283,
'0nijm7Rm2CT19_2w03b64vv4sDgURQgC': 284,
'0oW2TKuLD52MVnHOp59TkfI0UrJ6fyXa': 285,
'0pqtZNif9M7bG3JlTrZFeZqxZ68-2265': 286,
'0qI6xdz-eW3YdbWKU40O_W9iwrlDDiHZ': 287,
'0qkSOZNXgBIsTk-OicgAoPAHAFoS9ewi': 288,
'0re14M5dlXLX96c0nppbi8lgBysDXAX9': 289,
'0rsUT9-6YmSYMHwet2JDXZTDm2NGTIuC': 290,
'0smySpfFcVkfvqNjp2wX1M3tWjKjiZWp': 291,
'0ssxm5cQEJgNRBRIH4qAsqRc0NnsP_oN': 292,
'0t99FIhUsy_ZUc0jFVIrSvB2sFCTDeM0': 293,
'0unthUZRKC3pnIOPKi1iNUAB40cjJxzR': 294,
'0vA6pPSSVurO_ngTNEWag3Y46kbNB2uB': 295,
'0vLItdbtHdwAzdkq0Cx-iFhwu9-HoWgI': 296,
'0vUUJAAk4ajw0yDPqp-CCyKYfDQIhZOq': 297,
'0vYBY_pYEf0M6-ZLliAidPi7RNUzoaqk': 298,
'0vbOpEGgMq35ZRiic3EmiJg9nhvcHf85': 299,
'0wYpjA95SAaJBo1qxpUrCjmuGtXNV9JT': 300,
'0wh_rlOMGI9bIUwE3opajvEY1rx2CdRA': 301,
'0wiOOlG4jSCZA6f1Xfp9MBB5RsAqbGku': 302,
'0wtRkLBCYD5mxdj4WhN1-DBDIwApp8JZ': 303,
'0xVTAc_xhemQ6Nm5_22pQzCm5RMal4VP': 304,
'0yLFmLjRqLr5sHysNpshgTJLI6dPkgSB': 305,
'0ybXl-z_oeHprYjYQN8RhEWgEBHB5qRs': 306,
'0zG5WqRlSV8cTpBluRcykdjLvljV4RRH': 307,
'0zH9odveWsdGzqcLAQq8jXILtbQAeYM3': 308,
'0zRbO-8SB59HNnbnnWukr2peecVXjB11': 309,
'104M4u84kiL5aF64NLY0vL9qXmfSTaub': 310,
'10DY5fJQXOx7KalUWWQCfGwkbUF4_tSj': 311,
'10V-ulgTHS2DLolh4KW5tp85XO9bCx4m': 312,
'10fNHxo7I0UeslZ_jWO0u6Ks-gheaRM_': 313,
'10iAIDcSdelLbl727vZlrD0by-uCTME0': 314,
'112p8Eubmt0BGipihoMYmBYKcss29t5s': 315,
'11NVzbZdGxYQE5_-gctsEfldK6wdbtXK': 316,
'11VVVSXb7c1ckNPv7_Vk6rKqO-s4RieM': 317,
'11XECLB3yN9wqqFLg6hW4e2N-eamqWuE': 318,
'12FCBLx8K7kBgFDwiOvJ_IeyC9bMNhip': 319,
'12bzBfTJ6oCjrn0D4rH3_KZpTimQ2gHg': 320,
'13Z3ivsDVWRZsbm-_FGIgEv9DbEZzROQ': 321,
'13a0jS7wPrQHu1RXaU37k_mFy1Zcuxru': 322,
'14m1Xuuji5NKN-W_r6gcv6aalPQKapiT': 323,
'14rNnVtNC8putPL8uG00qwvfvlbqrhgI': 324,
'17i5A1oJK4pgHuLplyCYQtaavxVS-sxs': 325,
'17vFMhpucWeA7YPxhVCcbSk0sXXYvasG': 326,
'18N29EQm4FrLiq_O-e6jaK0o06yQxnM8': 327,
'18ydVrQEd5cHpg8yfQa2aFdHdOcqfDOn': 328,
'19CzwF8XzU1ot754JLbJx3Fpf8VevfkM': 329,
'19aRDkVkz0c0jRjkuPT7el0ExIB6CTIO': 330,
'19qyPUeyHtky7DtEF-z5jWIpx5kcqvRO': 331,
'19vOH1W4-cFanhVEGQd_eJeo6YofJZiP': 332,
'1A4QseB1hB802AE8hX3yOmN_NUeL_dz9': 333,
'1AYyk6F9YhbM0VfBanxosdrV7UZxlIqf': 334,
'1B03YqTegi5rWzLl3fGyv-C2CD6_We6Q': 335,
'1B8S7rUTZyaH6bTUft9NXFvcGdpWMvrK': 336,
'1BMz17rMd7z9tT1ro-D7sQpt0R34qQX3': 337,
'1Bd4UXH5we3OcmHlc7R4jYEux9Pc6So-': 338,
'1DIV8bDQkNGrU7jN3o4r5UVceBROpLrO': 339,
'1DTMQE_BODMnM91lzy2ueuWtSi3zRpWI': 340,
'1DUYh7L-d_IW6f6AKvqcE4mOGb78X_Xv': 341,
'1DZPhpJ2aaxpVGcSBJWjsa7kIG-FMpYG': 342,
'1De_f5uGduK-EuRMJ9bCQdEJQ9okt5Eq': 343,
```

```
'1E7_sCfoczuOsgwEOuHHTqzZoExkYhn2': 344,
'1EQDT6V6rDh5ggghF1AzaxQYqp5dYRkt': 345,
'1Ew3cVtiFPG7suOuRwoNYyd2gWzu9LX1': 346,
'1FBkQtWUeOW-aNZTiVr9o0_nu23C7DLC': 347,
'1GHNmC7S_bX62J8MI5MZ7Py0C_7UcSPn': 348,
'1GU6Ooc3qpY4OQyNc24ufGk7Y7jP-n4A': 349,
'1HI4dPEZ-y21jYKhjfQE6gBGPEm877px': 350,
'1HSzg5OwHx_szDwoLRTrk-AcTdQEB7hT': 351,
'1HYpmK191903zM4bePZXe4asdhwuZViN': 352,
'1IbeQqYXFyQQECqQQLtDIfTcMyioCGhI': 353,
'1ImVVVl3_ksamJz7GbAo4lyCZNqo7MZt': 354,
'1IygVyx527vpKAqhUjaVE2nQNnuluOmR': 355,
'1JcrkjyqVsQPp6WFNAa29HYQt5sBhjC4': 356,
'1Jf9FuYyyx6j6pabPWQB7CvCmShz3fbe': 357,
'1Jg4yMAvdCvI3WXcGeqvvtrdrdqd3b-l': 358,
'1K7_rSGjsLKDKWd9VOpHQQq06qUM54Wt': 359,
'1KnoOIPFzuMsD_PItAtDnZdLiAYihrKz': 360,
'1L5xgu5SKZCpxuhCKjVS4SSfPvG_v_zI': 361,
'1L_JXMkZFtQ3r9_hQJxfBw92aHqc2zOu': 362,
'1M4ut3jN5QclQfsst-YzP_m9BMogaFwx': 363,
'1MGnWMFJeWFB3QX8uxM6TLBmq_Fhw3tO': 364,
'1NSgBL12iqrfwjkovKFN-Ur3EMbqwo7t': 365,
'1NrUWXwi1AlWjpLZruXGmk53sbDkfj-9': 366,
'1OINw1ZOt_ORW1oBMsc-KdLl02cskDl9': 367,
'1Obc9B7eSwQewlujrRkWq7kUJfcw5J2b': 368,
'1Oip4wEMJv2ng1Aqe83PAjx56HGiMTtX': 369,
'1Oy4KGGOlh1c8wUUUdO4PZKBLrpBpdIr': 370,
'1PBPsoBPqFWHRfT9fvDEm0zPNVGDGlOW': 371,
'1PRNbvTl51TTGQ5zV0nRkCkmJMMiG4ne': 372,
'1PwSmg5OXG0sKu9InlOINBRfh7QtHFgz': 373,
'1QFtOeiyU_agnNAPHRILYsan4VkA-U9J': 374,
'1QR_F1PTejVat5774nIa9UTX9d7S471v': 375,
'1QuPn7hnGAwz7X6H1CWzwB20DGziBJ_S': 376,
'1R50flueeATWMQ9q4qE1TsNKBO39b4d8': 377,
'1REpaSmBGCDilAggy3M0B9LA6dq3C8pC': 378,
'1SOB8K3767HRXQibM8yM6CURzq4qpBO3': 379,
'1SZp7sMFWAfcLfGOZOWKLyDgfaIw2jT7': 380,
'1Sc5Z0wfPctbwkk4YCUcc1L6nChEDZq_': 381,
'1T57BY7FCh7Jw_baywaGT4EG-GJWzjmj': 382,
'1U9nVLKX2MvRPmaWCJXYU4Yn0kdxYYfr': 383,
'1UIY5zj1VUAlk2qFS_5MWBJxXrdIaS4w': 384,
'1Ud-h_HARSSP4Tjjhc0-HUDVd6soarT2': 385,
'1UvF3NrkZf24wENUp8It0ZGF8QWKnKVE': 386,
'1V2qyn-5WTsZm47-jPLkIR3bNPZ6IudQ': 387,
'1VjLsA1Ra4tcPqZNUemDZzil5lsIJkcF': 388,
'1VuGX5MFEMjI1pfLdyaJvp5BpKWv_UEY': 389,
'1W9k1jIcgnZl8tajA-W3EtcROTSkfk8g': 390,
'1WF9sIGXdgCPXFGqfB6ad_YwDNYKcmUr': 391,
'1WMzJ7fXNLeJY9mp3SmVBYEkNO-qBkSU': 392,
'1WNWYzlw1kc_ZYVHayEIYSzScUXMWYB2': 393,
'1WXqBxcqRoHZR-kny3GcwkyFlSX6jyXQ': 394,
'1Waq690jmuPnR0T9njwXqk-KsHP-QRQo': 395,
'1WgvWDWtK3O0NnLkjHcZxjTod-au8wSe': 396,
'1WvmJ2K50lVupBdV8NDihwOabfqF9oTz': 397,
'1XBQ-o8v2YHnJ4RzjzbkwlK6MYTdkawz': 398,
'1XQdBeA3dgcGo79Q7BcsNX4nb7WtrV30': 399,
'1Y10GrORCJp09R855KIEDMNNi9mM93Of': 400,
'1YQoX0msz86oNuQm7GktS3r0bwTngnmE': 401,
'1Z2jLqgMt2oPkCC-nDIoPsk6LItvLFVY': 402,
'1ZgnLSzuuP5JZnM_Ccv_j3fVnuDaVi2U': 403,
'1ZjA72_d_tFGEggXa9Q3KIk7jC6IzWCh': 404,
'1Zw9IYLc_1dTPlThgdhjz_wpvsZvvrQM': 405,
'1aRvFZ_x433FpXLyzCY-pxDAN3DmTyir': 406,
'1aXpa4_iXlGZ0SC3ARONEEf3zPGwA8HY': 407,
'1aijUXlIyIeC-qDJkIiz4LttS5_39fqs': 408,
'1aj7uuV4ZRxGdoaVMdY0dpJ58ZeCSK9B': 409,
'1anVuyXldQUTrsFmgHCD7L_IGIiTD9Nm': 410,
'1bJ1R8cPBMomopERf8vK_WrS3mT-YYKz': 411,
'1bRed8uTlCZAHvHm3vqcZc9GEdaKn2ht': 412,
'1bSW6d7k9gKOK4K1rOxeoAUJx3o_Xyf8': 413,
'1bZNv2012OVPN8FtxFMnqypwyVZD7DBC': 414,
'1bvgRyI1X05R4t2WSrk5WlplTGoxdLbQ': 415,
'1cDxUS2U8yKjmtXAA95BMehgCKW4lKBz': 416,
'1cIJJkvsvJpsroO6v-A9JUwABvC7pgR6': 417,
'1cLQRI-6caCbkqVHqp2HHAdvBt2TnobX': 418,
'1cNiEz6Kd_JZWOvmN55JD7tGOpscbljl': 419,
'1cSJdNSbLpGJegymzm8yT7jmJjz4Z2rX': 420,
'1ckAhGAmbHPpzWUZQIqPtTLQAXDohnfJ': 421,
'1cv67BXJdByOsAnhgK8KnzhhX17iDjzd': 422,
'1dBeTGoClhinOrIwXA2ATL_3CLZfTcKy': 423,
'1dEG0k_XvkuMTO-hTEc8YfGoAJoUsGw5': 424,
'1e-j-oyqCiTZ0l261_paxr31kUDi2hc2': 425,
'1eHqv0_f3wlGaezxnznWxMmktV3CX3gD': 426,
```

```
'1f2DSllXtn-izegn_AGH9i6hFhQcmbZ8': 427,
'1f8GcvWC5nogX58P0yRQBUZH0mQEMZ8Y': 428,
'1f8cay9_mVE6kf2yq_9ImDy7UPpYwHOU': 429,
'1feFupTY-OBxN89dhIJsra1MzKf78Ibj': 430,
'1ftSQEKFEre0nH8-rr-gsrdC6EoWKWN4': 431,
'1g44MQYprRumF9F7nVvGuZrPh2MLAKBL': 432,
'1gRedqoFomYyPP551id0qbQH9Qesybgu': 433,
'1hlnlle6X8caPWChTTqTv4qcPhV34wKl': 434,
'1htAKqqU-G6Cqwhl1AQZdrcqm9i-xhn7': 435,
'1iA0pr_QrCOKhjsc5fU3ejyyKEIpsMcE': 436,
'1ii_9Nn_ux--ITv9abIW9cRPrhZmLNSk': 437,
'1iruKpK-kBlKGyDXkbj9Chxbqp_LL5_6': 438,
'1itdNpamXOtd8mi0q2ZjCEIvqx1N7xr6': 439,
'1jdrNSJeVkzEltc_K7Mk2-IsrEyWfbur': 440,
'1jf3uD0ZFVqE9iQYiYOO-VE6odm7OaDM': 441,
'1kAS3sdh1P0gJ3U_DQmyJ0DqmaPSzUoG': 442,
'1kE9OOfeZ7_F6OQeFUMWt_5-uq05Vizl': 443,
'1kYCSEcDW_-nwTrBQPsCoUfXGHM8MvN6': 444,
'1kqumLwCPfPRBTZJm0tEKVAzbeHtUPXN': 445,
'1kt1VtiQM5FdJ-y3uQeQIjTT9q5_SZnG': 446,
'1lZcLyrNae8pdIqX58G667YoVJ5Zbqpw': 447,
'1la4E6OyV36DsnisYxeJMQJ1qc9g4p3z': 448,
'1ltpqGcpXDCsBIfXqZVYYs1RoEJ2pNuM': 449,
'1m1KoQlANgUhQ1cLTUHU6QbvfoCfYqDI': 450,
'1mlBvhGsVy1MZCR_Cdy_-fFEO97lIWe9': 451,
'1n47E4hpnfNyr6QJTEK8CmkFW0w5P5in': 452,
'1nDNb65XLzS6z3iHDsN03UeOnzMNyiQx': 453,
'1oI1E_inhroGuubv10Zxmiavustn-xma': 454,
'1oxvx6QU3Fu-8FsPvaWYm_uTVeuEQcQj': 455,
'1pA5JWMDttZTv37wV7iaFj9y79MSn6nE': 456,
'1pKa7AahA287dRDVJoZ3M05rbVcpunM0': 457,
'1qFzIIp9eUzXnZZL0ZH-pMAU0jwWashr': 458,
'1rSpRFWvYdCxqttrl-2I4zjXLspTxCZ_': 459,
'1s2IljhWFsisB7B3XT06G6pYlx2y0ijU': 460,
'1sFJ-TanktdMCx4TrnXhZJFtKJPGLnpV': 461,
'1sQ7u_Mnu59Z9jdS0zAZX_GgFLk3kXTJ': 462,
'1sYHGf76K4vYBlUzN7rymJCNmrNf9fPk': 463,
'1skzYa7AZfFS7smFkEjBhiw6X_c3iRjS': 464,
'1svSF84pb8dJ0GHWhMjo10SpIbEvoIgG': 465,
'1t61qKKjelgtm36MSBk_C_2rNs8nfZYI': 466,
'1tMp-b0zm9iEuTHKcbfT1kLMe8hRMEUg': 467,
'1tffPgyd8F6VAo1L6dcuKvmg76JWg4JY': 468,
'1tpd9HwRSnfBRfVH4CvpCyJvnqGcBecI': 469,
'1tyxzqtsqIjOw7x89shkTEIhJYfKUt9y': 470,
'1uJDKqRfdhmuiCTX__v37Bjix4Cfw1gJ': 471,
'1ujUd1Y_M_AgHP_ZMdg7mxNu3TmmjLeS': 472,
'1upuGPVh_TF-bCVblux0jaf9YqGw0ihI': 473,
'1vCU4q2AgWuScMV5RyH8SS8cswOFPUxz': 474,
'1vieClDxMdBPSeLhZPj1rnr5dP8OATEg': 475,
'1vqtN6J2VexAfykY9ayzbtSn-0CZjES2': 476,
'1vtVnw2WCi0LSAbAuL5-8Tax24ogWCM7': 477,
'1wGcB92Cw_GI9rIDVazzOPVoPJ_JeG-K': 478,
'1yXRH_ONSgDxTptIVvrHt7QEJSauwh3X': 479,
'1yaOztkceY3_RaCa_Usj4ju6jlYCWkie': 480,
'1ycd5yWwd2XW98qjws_ci7bOMgXcSlcA': 481,
'1zLIXtKrAx0c_CVhitmzpRBguBJbDKSP': 482,
'1zXOhoedJcbb5BDCRgDfv4lNFXn4PeRv': 483,
'1zXcV0MchLwux-27envVuBM0vSQZEANu': 484,
'2-Mj5qJtC_hAbyhe_HPi9XScqJ8WcFIT': 485,
'2-WZgciWGVInPQ-d11FHfsIApiZ6zsW0': 486,
'21d3xGeFyRYR94vDx0mnaRdStmi-HTF_': 487,
'21gQZTSokkiF5Q3JLIcbywYGbLPNsxdm': 488,
'21odkmLfwvSoM6f1WzVF5Jy5Jx57pgK-': 489,
'22NVuUJYPus1Qg8NR96muz13ZZUtz0-W': 490,
'23uStLFgUsRXqLv9o_IxgH_zELB6bWdX': 491,
'24uCYBvWTg5IGFvWiHb1LNlfXVHm5GxR': 492,
'25OY31Z51hpksR1rsQSpAoArscDmk_eO': 493,
'25fGrJaNXpkC8KFNG6RmnaRZXjkCG7K6': 494,
'25fpUJtAqIv_lliHZNFvCoz0-pAGrhqM': 495,
'264bPKxM4mbHcfDsRUia8KPKMflxcpoF': 496,
'266M4mTwfBgTsxyMHw6qF_aFDJ20gcQh': 497,
'27xcM228lHAI3XIOmoMbcgfujDdj6JFK': 498,
'28E1m4kX2_weQHMm-D3sjcz4X-LDbg-H': 499,
'28i8bwqceXDwV4RYNvLbtCSODe9MDB4O': 500,
'28zTBqGuDcmPacyvUNRzHRN3emCa1pRy': 501,
'292Fi2ccIYPOAp9ldEDb0lhLF0weGgxB': 502,
'299hJLbX-1VHPQgPzjRoMdaEMhWmMVyc': 503,
'29NM6JRlHEmA4dOnne7tLDgj4uKViN_k': 504,
'29b6bz1eLgTq7oEMU_QjK8vUp1a57G1C': 505,
'29bmJswm39lTCKm5-dqrSLoMOm5nNRnh': 506,
'29wBqtPBkIUm2f4dY_eiK62Dqa9t4cHi': 507,
'29xPZAacKpc_dOVhbRhQRjA7vgktsRXP': 508,
'2A7C43DhTs0_HsA99uM3Udy2JV-iFs3a': 509,
```

```
'2AFyNp0ZyH_cHbZRpW7Pm3l3uDrlVTpP': 510,
'2AXZm5I7yNLRVjDPyR5VxGzxLL8ip1Uh': 511,
'2AZlrEAdcyT6WBY5bzf5HTlfba5d-RBe': 512,
'2BFqNazfbmcvz1ZEsMNmv1s3Uk88koSb': 513,
'2CC97FzvDQQSFWdrSbsJ3_oWmhwz_Uf8': 514,
'2EB-AwdM5WOLhHghDSpC7BQz19FPlUnC': 515,
'2EGIJtn9LrHlyOvkYNxhY4Umo8RnoIXj': 516,
'2ELWzGOLvrDaLeh8f95PNfpCqURa8xbG': 517,
'2ExiBngP70__xW9v7X4HPr_6hxuiZcDj': 518,
'2FctdBzG8mlIN3-TDipag_GW3l4o6cj3': 519,
'2Fj-wtut6ib3GxItwIFRFVsJPWso7QKt': 520,
'2G2HSRx6MJ0a5-Hil7BZxOnPJPYmX3lb': 521,
'2GbSB47wNXtCYeY7jVJ9PYCoBo730Fw2': 522,
'2GdCrYmp7IVnMbA7R3i5tt5mXCyWFw-r': 523,
'2H-j8azYlCU-LAAOL7VZePPbU3t65Kln': 524,
'2H5cAmytxgWUv2zMX_UMMceiNuPovHyd': 525,
'2H9mwGqGh-0gn5NUdXUoS9taRdYHMJKr': 526,
'2I3AnSEvEirZML-a1XhKQcKGU8rlDmru': 527,
'2I9YYAtdqh6avO3ZlNZQ7Z-e_arvibhU': 528,
'2IJVrPbZbQ0A3ob0UXjmHjZsl7ccuXjd': 529,
'2Ioz7gi6uYgKZyry9iVbH_xd7KdxO_2u': 530,
'2JBqQ_xvoIu_sL6oSZRJIaKcC4548UgH': 531,
'2JIzmfXZwOgejGJtkQ3Y_pUPnTmKsuH2': 532,
'2JK-C5xpf8lcXZLrUoEnEzrCBNwxh202': 533,
'2JfJuOhSuVix0556r6eCu_VgXF-vTde9': 534,
'2JiOmsyREeLf1KlAeyDIPWf-03uWHior': 535,
'2KRPHX0XgibwxEGTOrMNXGZ4dKGGEn7H': 536,
'2Kw4DHzKYH_SU2lRBILCFQ9KnmIGGCxH': 537,
'2LryHAcV_frn9qIKtT7BsOpDLu5zJBaS': 538,
'2M0nsWQK2CLbzipyQh8wlLb1P0kqAl4Z': 539,
'2M7MBfPERGTMcefEI8789_Bo84PTj6FT': 540,
'2MMk5U1d3cULX_xTUNMQfoMEX-aPOBfG': 541,
'2MqT0iMIA-EKjFVlI0oyLnF6P9VtHsqi': 542,
'2MtgHxSZ-CUi6BIiahW_1MUhUmy9h8cg': 543,
'2Nr6AMsPzfv8mPmsadmoRDOW_kO_d0Fe': 544,
'2NxQ3XRmdy32nAkg7ovE0c0hJ_NV1bdH': 545,
'2OErx_5KYxVeJ0NNLpGJY42rLf7B5dXq': 546,
'2OOxe7-9cfnE-2qStynzYHpNg2kVrURY': 547,
'2OTRL4Vk7cvcnVUSO2ZFSfDvmIY1E834': 548,
'2OnqVvr8NTnGm-KuR25AU1sJ-mC0O4C6': 549,
'2PFbeRSsfeXxOwm02rLWlXBmU8mu3yCP': 550,
'2PUIvK3CqLF_pHULYmRVBQreM58VtV6-': 551,
'2QIG_Csz7oS_gb9vBM0AZwKgmXvNeAWi': 552,
'2QXAECqExTyMajnU46SuUXl9OrIsONcb': 553,
'2QaV2zjB4-of5tj1G-ZpIaJe_mWvkrOW': 554,
'2R4ghmq5-U07M1cEmRfvzgey8qTJs0kp': 555,
'2REA4LsVawbVf0dL2i5Qg8pH7UVmvf1F': 556,
'2RHgUATllvsnw6bFMKts2BHDGFNVXpF7': 557,
'2SDgCCJsgHBN4K6ni09rgtWvuNff5LM2': 558,
'2SUHPhoMfJgdCByqaucka-aZIl2X-D09': 559,
'2TC1iazRqLu_QuNnp12qMO565lpvGNk-': 560,
'2TGjDwtrDsk3LkXrwLOmbsFQp7urFjzI': 561,
'2TGtXoTC58l8lDPcHbQYgLeUoF_NKMXL': 562,
'2UXzXPtwePTtqK8HvdjbAXeqtDwxX5Iz': 563,
'2Uxv3lsyi7OOZmbUMda9vJUwOJkxPrIR': 564,
'2UzMB-RWy6u7ZjkJQTDY3WeGeg3enfKu': 565,
'2VMr0CPWDCYFD1fJvPVNlUxX0r1uj-e0': 566,
'2VVVrT9ygtul0pI53z58umLWJCJ3475f': 567,
'2Vs8sjAukjr5uzwbFzB3V565183w7ovT': 568,
'2VuCVDTz_M-NzQC-FQlBOakekG91-TYJ': 569,
'2VzMj2uPU-SXbs-0Zn8lZLY32PusfEnv': 570,
'2W8mtYP2jJXkNUjSXLSX1pPwFFwS3hL3': 571,
'2WeQWS5NFcvyHZkeIgeCxuRE32l2Na0e': 572,
'2Wlubn7CNV7iITy8MOo6XLbwhyDkaOlg': 573,
'2Y0hTCPTowUyGWZ2SLMzFPtbxtV5CDrP': 574,
'2Y5loh8asrmXtYhHEu6VC7Cs_6R-IVnl': 575,
'2YXVnV1GxU2fp25TL26ZYmGVvaU3BP0L': 576,
'2YfiPJVIhdnNQOirxLyKhQBBj76MGELH': 577,
'2ZSLHCSWR9F2fC9x-KHWry9cXSukSV95': 578,
'2ZYSZpfKxp9IdlSOGiptNVAC6CmGIPNK': 579,
'2ZpXG8kvBU0qCqrAIwTRKccDUkJlHuuD': 580,
'2_JF--K2bp4W-gx7Bnttgi5LLqJWihML': 581,
'2_KDqwjd93fch60jpkCdLNwwOtDbdpyg': 582,
'2_qoyEhX8Y4qaxI2CYC_73rbocB2QdXy': 583,
'2_yUwNH7bRZMBHRL4gy-e3SIJnaJ7bO7': 584,
'2a2UB8-haR0iaIwOy5F69nbeb5mhJrD7': 585,
'2aK-rPd-fC40QlV0yusu55GpUphqqB7y': 586,
'2ahVwk0WHFlGBzUxzvEMGqThn5wdP1Wd': 587,
'2bhK1rYQONmkKiL376U4dVdyT-dUHTLL': 588,
'2c1Lmy2yaxWQeDw32OvwbL6zJWdsx6x4': 589,
'2dfQ1jFUyyyBfzW4S6Yi4ERFMdo-Ngq3': 590,
'2dkaK7LvUgo2IJsx4ChvEDst2PJaGQTb': 591,
'2ea2af65-9jDtBIh3JECCxI662fFZ1qa': 592,
```

```
'2eb5qysY4MRSlB4wy9N6ZsfNffmvyoKY': 593,
'2enQoMRP2kb3o1-SLnWB-phfvA8DmyD5': 594,
'2fQ0rTt3cyqRm9wl_mPO_KvddIp_XBdr': 595,
'2fSp8nmBID6t3RnmCockQTLi407GfDGG': 596,
'2favKmuB3TLTE_1BkLja9dXl-xD-9yeC': 597,
'2gIV2EwclvGNbVMuuW9RX2DidcCyvJ0U': 598,
'2gN0DmFaz3BGu0cdEUfMINkM7ZsVqVhG': 599,
'2gYosQ5_VBzIpLHT7ZmWNloB57_MosB6': 600,
'2gdgivtU9VWV28dVpD-5Sjn_1i14D36V': 601,
'2hE74HZImMQXRiF27d9vMPOyEYdrTXqI': 602,
'2hFlOG0cSG9ykl5MbRa_xvUUyeuHxP8_': 603,
'2hHSvS7nDc7JIRDZuYbENfTh_qBOIpum': 604,
'2hi3PvtPcUjlMPCs7RxG0IQ_kFPev_uL': 605,
'2iUmmQu8Y1736ji_r2n3R0uKXvt2l1T_': 606,
'2iWVemdgaEpfFFmCTd6RUDO2jPbfXLn3': 607,
'2iq5de96l9r-ItqEVzsghZPiUWWxcg9a': 608,
'2jEUGPzofqTCF1NWsKgeB4UJn69ycPH-': 609,
'2kmmdwoQXrHv0Jh4Mg-wQySTA1E_mRNj': 610,
'2kuPZouZ8Ov5N-4MSTcF2pcU5HtXaaHh': 611,
'2lBLLfhnWrRAoLc-7vCaUKQHf99FCPLY': 612,
'2lQgKJA00NWx_MIugn85gE4opJKo0P_q': 613,
'2lUgcVYWqZX_iSxyEwdx451Lfz7sN0FC': 614,
'2ldLYl9Qs1czIeVpwl9ssKRIepCWL0F6': 615,
'2ljQt40V1O2NWXv0sy1DJyyVOm-qMofl': 616,
'2luwce2q-sVdmJGAcjuGF5HVCnv0Fl4h': 617,
'2m1CSRFfIjPCY0p6GwuJTuQ3rojkAxbH': 618,
'2mZCQNY2YSMPULzXNe6A2cBD36TtnQ9-': 619,
'2nP3DqdT40-TdoWfXJUoelg_PrXgZTjS': 620,
'2o5-TTPIOe6ogyf4YuoRyRotY29-Ummn': 621,
'2orWV9cLujBLSdKLg68-oZkWXu7fE6Q6': 622,
'2ouPGDAm6CYGD68YTkaZXm53ES-55E2G': 623,
'2oxNxEGKiEKQCm76hu_9V4BW1IE3rYYD': 624,
'2pFG7Gl6XzZ0H4jU9EdEh5kwA14xR8E8': 625,
'2pkwTo5_DQZFqK04TFdvgLOddG95xruG': 626,
'2pomFjuSpbZU5NW6aKiosPR8gwemOKZY': 627,
'2pufrxUcaS3NBfNofd1xGREVwRJwWJ01': 628,
'2qWFM9hUZ1zLT07ZsPmk4rvvAYcQvQuf': 629,
'2qbHSofNWZwaDs9tzFAH7drK5afqlGqF': 630,
'2qxVWoFe0VnIqTNANButMUYuMKebNMEh': 631,
'2r8mYDfL1MZu4zp2ZFEmOC9WHDaEYses': 632,
'2rmJ2nm8GzL5XB8Rf594DIcu1LpSKOUf': 633,
'2rxVwFjqHwOqMAbqwwI2BVgQ7fbnk2vZ': 634,
'2tYPcKfoUsoROKxUwFZrAjDrmjuoWivx': 635,
'2tdWjdAzZ-WNBnnzMCsHGiaPtOZifBDH': 636,
'2tjF3yHPG8JSuzlBcH2Xyux9URR7gioo': 637,
'2tqCbmKZtkXiZcF02I4mQV-t-Ul5Yna7': 638,
'2vEgWEYekOD30tlh33DBNwb3g_jWsTkb': 639,
'2viJZBbDu6HY_wtPpiFqZ3P550h47Ej_': 640,
'2vrTCA_UgqQUlcXKoTwSlvw70PYfswFa': 641,
'2w3VVowSzWReKKVgghC6SiiACTOlDhrv': 642,
'2w4GtDaxdU61SGCRRBzVKvc-_CPmj7rY': 643,
'2w8rMMYEwX15FCvCSq-sCgxP1wBz6ADn': 644,
'2wtWQ8KRCLrJOJKIeKSuQtFFDajlIq4g': 645,
'2x7pd7gmbExGtOAKA7R3ZI5ERxqaU3o1': 646,
'2x8cobwQWg8aMoXQ_w0FydK4NMaM_TNT': 647,
'2y31YniV8lFkOg8j9kcGEe58W1HL42uT': 648,
'2yIF5RMXrr2rhTPo77YOUcX1-nW3DCPp': 649,
'2yoE_ceHn4fZdo4Btojp2C8x4soikMwx': 650,
'2yu9_VBUc6LBHzhPSQnunXg_lgtDan9g': 651,
'2zcR-ykdM128rEhBRIqKG_rXscM4sPL4': 652,
'3-BHm7Qsxv_j5QbmS_6IytvAkzLNLCmo': 653,
'3-jcMi9X-TugvI5JPCMatCy0ApL9GKFt': 654,
'3-w2NggoP7AxnkRaKmQ94XWobjbRV8_F': 655,
'316TRyGB0N4gUii2dxIsUat1TftKWEpE': 656,
'31DCyqwnJrTq27KA2ZFU833Aabe1Bm2M': 657,
'31DFBHjxHj9et3f0J9c294OdVDSwOnvP': 658,
'32-IQL8Q_4qwebVR9HmUbs5gl2hKv3kw': 659,
'32dth4wVVKpeBOvikQinyFfKc5UwcMA4': 660,
'32xHxoa4WZTZPA9K9M8KLCSSrvbnkRw8': 661,
'33CVrEnmILJAdpH29TAOSIlw5lLs5Nn9': 662,
'33s5lsr7Q9Ix9sV-xxv4jnC7kG4yIlaS': 663,
'34F5UccouPRMKTn5omdSUGGJWA3A6bwA': 664,
'34M4AbuZ3QE3YxPb9YN0Tyo4KJlKjDV6': 665,
'352YlQC72eian835oYihpJPW8VLoWcLk': 666,
'35F0U1b9b0GzGB9Y9n0lWLlAoKaYbSWb': 667,
'35R3TUjOBqRkYNIRKYbOdyc2Khp6m3zw': 668,
'35bmKe7pBhzlJeUZuk4FlvxNC9mwhtX3': 669,
'361BbLGl2CMLrXQdCtAidSx9TVBaajEM': 670,
'36RiZ2e7bDsh7qawrng1GlqdqoOwmfRP': 671,
'36juXs9XAQYhY88n5ZuduRdx-MLtfc1O': 672,
'371vunWHQ3nMh2byzu--4mvvHVwIGVKj': 673,
'372e0sBx6sXsQ2xk-bSlaFsg3FyKJdQx': 674,
'37Xml_-NJW3Tuq_JVuOCP5GrA7u0M_eU': 675,
```

```
'37hQiBsELk7x5g2wAPXRZm-LNYJUE3pF': 676,
'37wHri0d6UhHI_5LpZTeLDgxIFl2a1yD': 677,
'38aTKFXVaQ1PYIXa1iUvCsxHuUeqw0m_': 678,
'38b1Spz-mgWoI8m4UhsYf0eDfUaDPkmO': 679,
'39M26oMzfZ9Ri7slPZ9pR9GZpqZxUDtU': 680,
'39p3YEg8a1lcpEiFpxbiEHzPJRqRKJWM': 681,
'3AB_C0gkyMJ7WnpuSzeeLsPmeTBtAidW': 682,
'3BzX1B03T8JOF59LFARoaeKYTXbq_0HB': 683,
'3CPa-xat3t70sYLQSCRl-h5PpK3SDtmS': 684,
'3DBE2emw_yVx0qiHt3fBsuO7koD9zTdR': 685,
'3EnU_oBgFEmpPjZdbiNliZu2cmIvzqXl': 686,
'3GFCAZr8KsUy1vyfLQ2boJaSknP-K_88': 687,
'3GMDpRkDIIqcpgMRXkayojQS5mZU-5Dd': 688,
'3H35JLJDv9FwaMaTcfAaikuJ8JM51g6u': 689,
'3Hvxi52Z_gsUoxGBL7t-DOH1zdlDed5s': 690,
'3J88s73PvWGtKMze3NGp-aRgxkFSjVdQ': 691,
'3J9-leIj3z6aoHUqkyGyyv9SXzXZDiJA': 692,
'3L6oKHFP6GnbDTMMEE4P3tiTSKCDuDNb': 693,
'3LGSst6B7iam6ua5uoPG_D43HkkIGY0v': 694,
'3LffzcWX3Za2XP6ukSgRXpoMEc4NjNWF': 695,
'3M2YxbbPrRjUUR5pfDECU4wqVv8Hbttb': 696,
'3NgErGjxiT-PrbBOmxVPK5Ymvka0bXFi': 697,
'3Nw8kTCy5JnEmWqJNZuav493mo7NT-PF': 698,
'3O0kKszroAIQiiti_rAvhysjMv20MgQ4': 699,
'3OVSzuRPnQa517I79gpAdaDL-Bnu0ZgD': 700,
'3PaGN_YNmd-s3Tc-kVQ_gmnz1qJfAUHl': 701,
'3Q0d12url4VzO7IxjwPK0OoWDKQd5GnH': 702,
'3Q1kVD94ZqyPZ4u0KM2twUKHTtCln9X6': 703,
'3Qu6qWI3mFexpzbBxn_J5CDGz35k64nW': 704,
'3RXZB5WjHD3amDdDK66AjQa9ECiCxY92': 705,
'3R_4GV5KOCoarNCGAzSXgpMVjekTYMkT': 706,
'3Rd_uagxkHDztjgtQ8pU-DeOyGcdnDmV': 707,
'3T4FRN6-LkJFjSVE15AMkSkmiSl5fKWU': 708,
'3T6SC7kHyQb_ILUr2wuYJPc613L0Gwor': 709,
'3T8VgqPH8lrsTgg-u5rFVQ84VuJpO8zf': 710,
'3TOEvIQ7xFc6YBf9dFLUq7tq5xCbERMz': 711,
'3UE6lkTPPZV4wxA7vTJT14B7bzauNQ-c': 712,
'3UyfxlJNKpr--Hq5nZaYG1Y-qNJE0Y76': 713,
'3VJUKZtHaarZVvBY7gTOeX8Amd6wz4--': 714,
'3VPycn4-dAI1dWHGHpYpMAwlePldxdNi': 715,
'3Vo9NP0qU_176pgbqk6Cu-CY7kpJ2-WB': 716,
'3W-rYXJxKPrTPczKPbKgXiJCbC7Xgt_N': 717,
'3W5YLccdAzcdrAGU1_unkngnq8qZiQlD': 718,
'3W7qP4zf89YXHwkiY7AHujibneg0lRjt': 719,
'3W9R6BtTpDwz4O3sHCGHw3QXiEHfQity': 720,
'3WSe9LFJSINJHQQKUEltApbt5F20rtDb': 721,
'3W__YolXb87BhzqosWyJczgoeRyPv8BA': 722,
'3Xm8Y8UrBAmeMj_ZI96LcyykZaoj5Vri': 723,
'3YkQkXI0wuv2Bxem3ICNWU7HCxpogvsk': 724,
'3ZgmyImLc4IDgqlk-gyRz2pcTCtrZs_r': 725,
'3ZpuX9NlIP9JqEmVz_IR_7xm0qSz9zQz': 726,
'3_A8lFgxrTsBnj8P8YR7tPY7_yTXkJrt': 727,
'3_IMM_g250HQWsDOXnlgjP_cY9mzJUtX': 728,
'3_gu5K88nuLfw1lzTkkuGugPHEd37wYA': 729,
'3_tEnHCfJV2PII55sh5B3nmkzuZtahRF': 730,
'3a_0HrRwwRjt3jcXXl2H3UZTLWM4DOy_': 731,
'3b8Zo_g7sLWXj3LyJ4fS6DgWS16ERx44': 732,
'3c2CeiDThxLymOoy0xeu7LKlLf5l28LX': 733,
'3cOwMHUWT32LKeYzehnmpG48CEny5uRz': 734,
'3dodNXUor8NvetwHL6vySIjDhfGE7vXn': 735,
'3f1F8S_XEv0TO1FSh8I2aFK4LKYCE3fe': 736,
'3fUfX6lL73972EwX79drMHvvFrnWNMcc': 737,
'3fZ8geGvtIE8mAGBFcOV3_IbC0M7EAzl': 738,
'3fq3zjmANg_F9vxKMgLBdqphejeVwdgC': 739,
'3gEMrWgy-4lu8oaW_HLOx_KP18B4PzVZ': 740,
'3gUxZJ_KOSrv3yCax6Xitc43vSG-AqtN': 741,
'3g_ZHBajeE5KwRMRH-puiptkr0NAKoi3': 742,
'3hG1b8nUkp74cZ3wUpsIH9qtFeB7u67I': 743,
'3hGicNFMVSSWDDezaDWXqxSUwP6J9FnW': 744,
'3hjQ-jt6bpoSCXan_s2KwdOmjLmTQLhr': 745,
'3iW3Di79SqENzSg9sT-79tjNeGgEfvBE': 746,
'3j6Ak3rtgM-nZnF4rYHnjoPhKV-EpleH': 747,
'3jTjkgfkaZJx17_FTq3KO-lFidV2EcTk': 748,
'3kfCGwUvHInXf36I5EFZAjcp_JGZd8dD': 749,
'3ktPVxdkkfvPxXT7KrpJ7ut2APwWmn4u': 750,
'3lFYyNo__o-utiLDmby3mZ00SfcVCJwV': 751,
'3loXwsb-wc-mR7ili3a3B8EZWLWvvFKF': 752,
'3nP4RUw4sZiF2d1WlaqHnv7QsnvK73-l': 753,
'3nc8Djl0rBHGAEPlFGtkyHLgXTwL_8SX': 754,
'3ntrLPHn8Xu0IsVQ1z00T2g1FnaJG6jf': 755,
'3oGkieXgdcyJFEBmNqvlaAqPn5nrzcbt': 756,
'3oodir1dwph3mJipgwJzs8PiV0ziJBcg': 757,
'3pqMiLsGQ7juDnCzhxw-jMO7UDcP4ZhC': 758,
```

```
'3qciWxd4O1k219qIsSpvIBcVQe0O4w7g': 759,
'3rG0EKehjCLcyFdoXHa9ln4X4TTfsDOo': 760,
'3rzk_Zb93hHxE86bG-sxlfovl6SQtTz-': 761,
'3s3x0UtTcSGNmBqwy97QNSBFw48GM2KB': 762,
'3s4DZXiZwmXlV-52Y6-AwZOX88-kMHPI': 763,
'3s6b2IuNRNScyEiNb2Bck1smA3URxC1j': 764,
'3sxn1odNdJVQIQYt_O2Ygm4KRxVuME7h': 765,
'3v81OtXvwH1B8oMaKMGjKlBh1k301M42': 766,
'3vgd0qkhIGG-ToibESLfVCPfNBLjhVnF': 767,
'3vspF5hfYJFs7Cs_C9p32fWsg_tgPJBm': 768,
'3vvFty7e4E3kYopxslUqoyA4yDkKQXkg': 769,
'3wMRUYhhEiX_96V4QZ4hAZaEq0jm835x': 770,
'3wtiTl-L_dBoerMnuISR7L5Q9Xf8Q0pa': 771,
'3wwWTYOrvkZ2FGHWdlX5RVuFJDOxwOGo': 772,
'3x-tIETNPoGD2XvkzSx7wYgQkptZXAmW': 773,
'3yBJYtWhXCO4ROSsGp5ACT7ZdN5lS-GL': 774,
'3yby3wMAAKFnwYi-C2I3k0ufxRuUHdht': 775,
'3ypKKB64CBEAuiwUm4oeCtR9EY9Eyypw': 776,
'3ysSTp_xwu0yT25Ku1ka0zaT4zLv8xbS': 777,
'3z-shimOO1skpclrkA9gyx1AV3-MAn15': 778,
'3zBH7k9Y-3spUaJi1xY9tK6HvL8Ye_at': 779,
'4-9aTr7FNWVfE4--8b-x4xBNLdke8bQl': 780,
'4-V2QJQy21qkQ2hS8RM390lX5jq0Yb2j': 781,
'4-eAo4BJ5euk_kbJEvZTgNZq1tMoVOru': 782,
'406QLnJ_jtj8Y3e5FLRtavScRWivR5Aa': 783,
'40UCgZgOsQnTIoj-1HtvVZBvd-UiNK33': 784,
'40jdGtb3w3sPr0JkgX_zMqNKii4kvo8m': 785,
'411WQJm9kG89-PwVLid1cKMoyfeBR4CL': 786,
'41G373B8slNieX38YyjIvHXYhrwz06Nq': 787,
'41RhApWr22-mwo6QUIoWenKHQ8qkDFuN': 788,
'41gAzCmN4A_EvseecpqRtEBznlxGeqMN': 789,
'41jLpf_LoZ-ALU19HtAKQXytM2pF_l1t': 790,
'41xSKhntNnqgYlIfXrd5t0sj642h9hdB': 791,
'42nJzdAmAd_PovRI2OZcunWLfs6VWyUI': 792,
'43ExCgywmeoDQiGQD_sqZkYhSmqDKRSn': 793,
'43ZC89EOSHJ5ofjS5FbzYCCidhvkclrU': 794,
'4410ZbvChIi9KX5MsoEbsmqtJxrxETYD': 795,
'44291OEfL34muF6cL-t4qJospOkTbeht': 796,
'44zvseG9fhDvBdogQQDVAwfesSBg8cTT': 797,
'45EbYhshOB6X7SuZW2PbKX553gDTt0wJ': 798,
'45TmDQuhC0bjgTIsSStEj6ttHdgF_jOr': 799,
'45sggATS-5O3dT4TmIgHw232Gsj9BXsP': 800,
'46auWJOjQBU7ip_5jYUK0GGjXXDrGYfn': 801,
'4706gg7MI0NDzeY6J9T9wr4l6FA0URKX': 802,
'477026M-WwBGX6H7R7uAy9jYiNPSyKyU': 803,
'47BUyMUoiWRREVbUYTUwcrqYK65hr3sh': 804,
'47Zsi7hdYWc0mkC8c-8fNJbjRm4TFuVA': 805,
'47g3zrBWMIU0YfTEE5tck6ssCgN-Rca4': 806,
'47gaY9jckiZr7XvGh_vYbNI2R867PZwo': 807,
'485Je17jo7HjucnukyxdOkXxVCVBY6BQ': 808,
'486yS1MK7xd3KLAQkgTJeusUlFbumoyb': 809,
'489HH9DWdnmZqy_TLs8kvigvyxax0rld': 810,
'48ZzjnZCUG3tum9h6Yz57iubRvcT8taQ': 811,
'49JWS9E7KuSnvv1B7uqMnsiT82ZmtRlO': 812,
'49NWlNgyXEbVxExxD_v92qTImx45RKjt': 813,
'49ZNRunJz2CinBpJLEgX0AQzldgXez4c': 814,
'49lX2Tkx_OzK1LY7cepz0kqXUDV1y2Yy': 815,
'49sNrj7V3OO2hh90a_Z6R5voQiG-H2Ha': 816,
'4Ac-6oopEzUPxCOl-BOUe8nHjrUI3wt0': 817,
'4BIazM5UOqam4tIDbAJHXqjAsZEb7Q_5': 818,
'4DK_28SteVujsg--lDvYSOyVl3WlrUHn': 819,
'4DTQ9-W7ccSJjV7pvkjB8dexMg5cA4zQ': 820,
'4DX7s9Jnv7uLmTU0AlWLsQhoa0JBb4pA': 821,
'4D_m1AKzT03busl3SDou2PVVAo82fYgp': 822,
'4D_rdRYXzIWNGIguAbgnYoXGMRTiWWch': 823,
'4EIkvOcQAR--nhBRSpeCK2BOgLwHUcIJ': 824,
'4EKN_ISnJHTTasizExhENltDvKuhOQ51': 825,
'4EQyfCc5-mow_U212x-IPwq341cc5afx': 826,
'4ETqZkMHxAXmD2ZzaZXjjmVtPsv6Sbq2': 827,
'4FVYWrii59MME-1803XxvgI2n1eduyrp': 828,
'4FXjrlMpVAz6bumFBfsufywpJgCcpwvE': 829,
'4G0oWBgzbOfgzVE9E7qrKHeLZBPGj4_q': 830,
'4GW7RbedBpz9OracRhs2AxlRD0jpnmu4': 831,
'4GqwIngq5--ERXc3NdXDK44fe_UUmlQD': 832,
'4GrQDVKFuBwFaqNr4VXrSWqaQiWCNjWR': 833,
'4H0T8PMefCG17EOGR12gRWrQ74Iqe2XG': 834,
'4H432QIngXL5nhtDcDwqYdNKynbXv-dZ': 835,
'4HbrpffP0EdKro4EoR0vQINl73taIOXR': 836,
'4J-s_U9as7_B1qPwthRJKNTWAztt8rSE': 837,
'4JZfHSGAa50K2jfJ-Z0f07HzjcZxo9TG': 838,
'4K2IZxhqGtKjw5QuS3KUHaztFOS--8kW': 839,
'4KALSbzAOjKiGEwvcMJwLeH5k1eVfwMa': 840,
'4L4Kb5iLFYt2-ZwRbjpwIIDq2KrTawQ0': 841,
```

```
'4MXyfHYDV5OO-j40vluEUrrWL1pb7rus': 842,
'4Mfke4gSYVsrCuzRtPQnUd4QU6jU8sNs': 843,
'4MhZS_r8TatbtD2ehVobUwcJHPHsBWDi': 844,
'4N3rulgOoee_JAsy44eKeizeiF1L23q5': 845,
'4NUdMNYzHaeVX4nixYGv5j9vHPgzaVqc': 846,
'4NhRKZOH6DVCBEeXZJfy30evRbhUsof8': 847,
'4O5iLrDE9hpz45LO-MQ2STT1BKwsbfWn': 848,
'4P6mpYdZR3RwHSz2Z5PCK4J_bcwOF_HU': 849,
'4Q8B7XBnR1DJYny3x1iXOK63AvewYWOh': 850,
'4R-pR78UU17lcW6mMoUfjBkKVpoRAfBa': 851,
'4RQTO7CQcCfdfDuGUca0gRo0jIFTsN_6': 852,
'4RQx92741iUZfCPCWkeetO208lRb_Gyb': 853,
'4RR1vbngnybZjd5hjz6V09cd-fibje5F': 854,
'4Rls-AL03umGHf8-D26FS3ONsJtD3PYi': 855,
'4RvyPujjyDQmzMps3IrBC4m-Zr1L2uqW': 856,
'4T7XHqft6zF9pcETZnOBc5XehLQ6mDZ8': 857,
'4T91pid_BvqpJJVk6hrGw8C7BvhaJLAJ': 858,
'4T95OfGup6ZLZn5rViQSTUeS7JI8HNQw': 859,
'4T9buseRYH9VBwK6tDS_Z3xZN7sYk1ik': 860,
'4UXinMnG51liZqI9CK01s3bLFx_r95Di': 861,
'4Umtzmfvc2J34sKhezWRGoQ1rQxVpIUi': 862,
'4UpIZ3aJcQonB-UYIcdYjsvbVWFBezw2': 863,
'4Uz9teUhN4tZrmzd9AD25S3YpdljIN7L': 864,
'4V93J414hLyJOqTUK1fxLOaXa8vAo5lo': 865,
'4Von9wJS4JJ9HrIHgs8C3dnazulK2jZp': 866,
'4WjShZUBaJm6lWQF9Cnzp6PVfZz2c2EH': 867,
'4WwnZaRbD5EyAr8U8ILTI-d7QZ-Mfrka': 868,
'4X9XjHD6DAuF2uTvw4yssa-aIQzSbKh7': 869,
'4XCC-OR3FNun2agRRQ2q19Xs1h7IGLHs': 870,
'4Xy7E07nLANaW6HRlqdPdsboV4GKcmAl': 871,
'4YIJBxdUpYXqIs_4YSKFp2gRCSGkm9Om': 872,
'4YSIXrnQjqYjTgDBnQWT_S7cB-NkMT0-': 873,
'4YiYgw9bY1OQzQb3MX5aBTgrZCq0H7Z-': 874,
'4YlmRBKM7ZZfaLqT_KfLiCQcXXFUHb4b': 875,
'4ZfCT4b_tVpo5KLi7j2bS4lT9oG9ePvj': 876,
'4ZzqZO8UeDbtSETYM7WFwuwjEZONI_Xt': 877,
'4_1LWJ4QkXSzsfTuznPKfBg8X2G8_VO6': 878,
'4_VLhHoSA7ntO1WhnJAfyIkCjMUTEvSc': 879,
'4aSys6KOMGiSfqpbQ3s7xMYP890owDcL': 880,
'4brfUkxRTHLzbKVWip6Q_w4tXVIuBYng': 881,
'4c3lDZh1c5LI09HhZQCSXDlbWNWqUrl0': 882,
'4cVMkHoqFwu0taFXSzbHG9ud05zQYbmX': 883,
'4dIAPjIFU5FJUP15v4RJY-95gtDKCo7v': 884,
'4dJdE0l6aVjfxpfiUjdfMPUNiA_EU2h2': 885,
'4dSMHqbsJn9CElpXms9r7FZwuyW6ssyd': 886,
'4eS7iyQhUqa17v0cFsUvCWTA-dPyN7-n': 887,
'4eV1w1ONY7N6PfEETef9a5Ic5SLZoeGX': 888,
'4enQvE2kUbKYV70_O_Ul6Dd5DaG9TmAN': 889,
'4fPdsDyCWs27EfBGMhN5roiKFPrRK6VH': 890,
'4fcT5XmJdG6Xp8N112jeBX970MCruFHn': 891,
'4g2p-C8OF1nKwIJuNLPmVg-Uwmfu5tDT': 892,
'4hQnv-dNj-NtnnhyPHQw1z2fi9pstyhX': 893,
'4iGSV3mGmlUF9v-LxYQDMpCcN4NYPCDQ': 894,
'4jtvevgAk-iHvq5zZhgJYcjAJKrQ5lVg': 895,
'4jvauHnKo6_j7OpGaj0DOwpe3dNd-EtA': 896,
'4keRojyj0zJcchjR6u-ZmQvdpRkWKnod': 897,
'4lBVgtvAv9YQvFL7-WaQae8A4oriVPLQ': 898,
'4mAlLixhUN2om4-wa3XDTKM9OYofNDaK': 899,
'4mFTk0wk3AkEdUlvM7ZI_aHxeibRqE8-': 900,
'4mn16q5dJemBuA4JdHVWIoFDbR6qLoGE': 901,
'4msp4tay4ftI4L5byUoJS9YGnRZw-u6m': 902,
'4n7hN6ydVRSukiwXBmcYyno4bL9OiLE_': 903,
'4nd8eMyLU0zxrbPo1oe7WYdt_v6VPY6-': 904,
'4nqg988GFxQp4LLbW5onMflj54WlJvra': 905,
'4oVj6y0dwROdfyFzoA5M2t9S4YjDQ5F8': 906,
'4ob9y7xOPZrNFobZJurSWzDZkAbJKp-3': 907,
'4orSn6XfWeHsnaMfqds6x5DsaSXU1BBq': 908,
'4ouNdQMW47T5j8fCKnHd7OZSJHrmWnrb': 909,
'4p6NoLm-aQfp0UeFiEzR9ALrrqkdoehd': 910,
'4pQFFwITd_fgus8vMt0nshxCeWOyRlyl': 911,
'4qRNG7_zqvIQ08dQVNRuNdRmeQUEhDf8': 912,
'4qscDhyNbUEUcNJJR6-UqBzLUfvX98wr': 913,
'4rPsfkmZ718_-SUdXMZvtsjDeU8F5Cwh': 914,
'4rarA0c5f0yv4H9viqzcbakoBqzJxL3G': 915,
'4s3zbVPU1Ccy3VuTKMemyGbUBB6QpfmK': 916,
'4sUXzRaMk_86WF_uPiLinQnILSjztlbk': 917,
'4szxQc-VF4RyfmamkhQlANd0vFCiW2TA': 918,
'4tJTRLZCH8jdin0Y2ZrTidpelFdNk3ZA': 919,
'4tPLsLE7Mtkco7v6EK2ofc4Zcsqg3joF': 920,
'4tVqWHkxOLpFuBvmjm29nbxGkgzeNpJn': 921,
'4u3hslcMKVTuRa29LqQExPGPJL1SmFD0': 922,
'4u3okc8uvwi6wdUJzbQtEmMFRFjCJwxf': 923,
'4uzbfZ1IcKLg5P2weKyjM6HKDe6VXRjy': 924,
```

```
 '4vTm4XXm6uTq5iRKbjDT5fSBEn7ujPM0': 925,
 '4vjmCaFj17JYxKxLOCcGF2ms1EB8iFtm': 926,
 '4wXI6Hj7OeiarwANmTrDS3339DEJWwd6': 927,
 '4x4MeTEpmYuyUPRTFeVkvsGtW71DaAWX': 928,
 '4xlN7cPvaYk8YmFTZJyE5UlnB6MR6exb': 929,
 '4yNyJhQnTOFubOj69U5DyQqjxV7PcZpF': 930,
 '4zB0armWSKlrCz7_2KpiQFq37Or0D14o': 931,
 '4zEC2VzsSUrpr6QH-aeGcLza2_CEwNdk': 932,
 '5-FpnyPWSH6EXeLmI_wjovtPCJn3bM6e': 933,
 '5-YSXnfqfR6c4BzKp7uHFpsgp7laAAEU': 934,
 '50JllO7U2pwhEQgNCme5TxaINqIOfWWK': 935,
 '50_dgYsSYnTeKdYc4Sy0D5qo6UxvM4Jv': 936,
 '511zFStHNuPlWv5yx-4-0ToPmHOI-DA8': 937,
 '51331WZvvqbvKJ2d8pKrUldH1V1CDDA8': 938,
 '51AJgn1VmCb_KtFs0-JoMNi32X7z3ue2': 939,
 '51VkvvbjdcOwwaCVT5HKuUL1Uyb83_ED': 940,
 '51a9_iBXNLujPshssUN1Lv5dQ6ZR3rwU': 941,
 '51cggs-aKxLP24D655IX2BjzA_8NvjP8': 942,
 '51oPQSdGNKGFa4XTtO_LHimY3CVOoH1N': 943,
 '53IXlwt_bDzeZ5rp7Uycf7yXN9KDqRVc': 944,
 '53ZeorDUP_zEbxtnDiMCLf4Jdn-9A1dM': 945,
 '53lpINI37q982LFHKl6E7HgvRf8i0eBx': 946,
 '53rF94jhIooroRf7h7rz3Oo-9X9Fbvv6': 947,
 '54bVWbFzC3pbj9qYJvyXC4dc63zF_5JJ': 948,
 '54qtceLF9yYnRP83_W58h9X0ankIBUk6': 949,
 '55l8n4HWKEd4ZYM3TFwkqu1CLqDoct0j': 950,
 '56EU0TVyi8UorpOFhJmTLseCA8TsCmrm': 951,
 '56Gw1PWzOqLRQBNvcsRQcRZ27NS0CRqu': 952,
 '56e47a3fKvBURcnj_zx4H9k2yysZxcbY': 953,
 '57AdF8Pg0f9Gj20DYItyJ_0H0TdHQKvr': 954,
 '57KScQlbv8eUq3ckpgA-rthyOQ63bKl-': 955,
 '57_l338OuPuWPvyagNZ98A2GQ_EGUMNJ': 956,
 '57cohfYdQB5XWa5xwj62CcqQw-izhfg1': 957,
 '58elvaD1Pg0lFC5Zgjng7dPtHfGM_jkP': 958,
 '59Itqhq-72tvCYVsPXC1INALPCCQOvu1': 959,
 '59a7r1bxIMRypvOPIwzetvXsaCgH6nhZ': 960,
 '59a7vrM26nCNWj3SYBTK3h9EgbW_TX_l': 961,
 '59lr29fu8SFzIL3DOOzEeMpsvAYRZ7Zq': 962,
 '59o7LehMGOfL0cGEvDERUaim9zYWR6mN': 963,
 '59p_2YUIYlDGEMghymRBJaue0MRh-_aI': 964,
 '59t7MYv9D0lYO3OwL1ZnTEkffNEu4p5o': 965,
 '5A9mcgtTvHdndhHHQbn4eFiEHLO2Vi1L': 966,
 '5ATcNS9tsSrlEvloWZjGf4sS1gda2bsw': 967,
 '5BDR0qbmRNOXeWtn8YqyN8TbRYaZMptE': 968,
 '5BcrYUl9iTrWRYpI-mEEjV4Ls5NS_x_v': 969,
 '5BdGEHdJcjVujQLaXwVTlf0Gz7QBbtQC': 970,
 '5C05g_vxGqZYZwOZv8zuIsdy1cGUhdB7': 971,
 '5DlFMgnNQD5e_avzgoZzOwc8tQIDgFgM': 972,
 '5EA1bVBvgYoVG5vQAExlx6ArxfyZlHYu': 973,
 '5EL8eUTWOl02kHw93bM1Dh20PEaWjgwj': 974,
 '5G7MgsUHGjFQIEqAk__6XyHEp2-GMcv9': 975,
 '5GV6S7mh_SiBDS54_IDbDAf4G28YGWWD': 976,
 '5H8y5BBRUDvdqT7m2U1B4QplqI1FK-gH': 977,
 '5HHdaECJ_UsYQYlMiVhkqpKeMW0Z5_V9': 978,
 '5HMZLo2KWysOJUHXP4vWhVYtG_0pqTIa': 979,
 '5HY38_iZNN8N4nTX3i0nd_xezzi07Yhe': 980,
 '5HeynkzvXh7fP4mCLEKFYI_LUcMbS7iu': 981,
 '5HxsLNw_8w4CLChr0YUfl5QVDyIIpW4h': 982,
 '5Io3YXB7dYm2k7HYJJ_nWVC57H1EE9kG': 983,
 '5J0wWFkDiJAYR3FJY2O82gNXYaQFBoFJ': 984,
 '5KuMjIfOgUU42zP3zNzU0xFgla5AZRnM': 985,
 '5LKLimaj94Tt85pbhhW-Uf69o8pPpO8-': 986,
 '5M7aMeqZNL-WZKziTN1zNNtWIN1s4VMB': 987,
 '5MnFx33ZnloO0g9NBkXfp6mt209ERN5l': 988,
 '5MrvgHtigUo8W2wxKPgAcMpo9lc9jgDo': 989,
 '5NDuk1HB6dypgi4eS_jTA8ptiyTwSqsg': 990,
 '5NSbOHD6_x5RJtJTxXnnhVw2X2YuXU4F': 991,
 '5NjzUGBt-0pJKjaomqeIR2EjkAPQiQTP': 992,
 '5O7FFwQlpmu7hXAOHExPAtodHAcRp6Xj': 993,
 '5OKziFBotwzKH9zyyvLtCmm3yCZAtJUA': 994,
 '5OQEexCzgG08BEWnDrBOAu4bjA6mrfqG': 995,
 '5Obj4WDYp72wxVmyHOD_h3Ml11JK3asu': 996,
 '5P8TDWQLIIOrlBRw0Y7GZIPN8IelX0Gg': 997,
 '5QkEjzSnMlXfnnF044bYN4MRjQmpjM5V': 998,
 '5QzvuS_olXP4UVqjQSTU8WhdwUKRWamn': 999,
 ...}
```

이제 위에서 구한 id_dict를 기존의 고객 아이디에 mapping을 하여 새로운 고객 아이디를 만들겠습니다.

기존에 map() 메서드를 이용하여 함수를 컬럼에 적용시키는 것을 해보셨을 겁니다. map() 메서드는 함수 말고도 딕셔너리 타입 변수를 인자로

받아서 컬럼에 적용시킬 수 있습니다.

딕셔너리 타입 변수를 컬럼에 mapping하게 되면 딕셔너리에서 컬럼의 값을 key로 갖는 value를 반환합니다.

Hint) mapping 예시

```
data['column'].map(dict or def)
```

In [87]:
```
# data_logs, order, user에 id_idct를 mapping하여 각 데이터 프레임에 새로운 고객 id컬럼인 n_user_id를 만듭니다.

data_logs["n_user_id"] = data_logs['user_id'].map(id_dict)
order["n_user_id"] = order['user_id'].map(id_dict)
user["n_user_id"] = user['user_id'].map(id_dict)

# 결과 확인을 위해 data_logs에서 user_id, n_user_id 컬럼의 상위 5 rows만 출력해주세요.

data_logs[['user_id', 'n_user_id']].head()
```

Out[87]:

	user_id	n_user_id
0	K1d8_t3-QIskaSkrx32oAFu856D8JmLo	3314
1	lwFZ77v_ygk0uU40t1ud3l30EZ6sE2R3	7844
2	mR-bO6hC9g-m8ERXMRQZaRwJFvzNNdd8	7920
3	K1d8_t3-QIskaSkrx32oAFu856D8JmLo	3314
4	Yjny5AchUWLiuv4kdeq50COF-S8OFXPd	5608

12. 주문 데이터, 로그 데이터를 concat해주세요

주문기록은 user_event_log에 기록되지 않습니다. 이는 바로 아래에서 확인할 수 있습니다. 이를 확인하고 해당 데이터들을 concat해야 되는 이유를 살펴보겠습니다.

In [88]:
```
#주문 데이터의 첫번째 row를 출력합니다.
order.iloc[0]
```

Out[88]:
```
timestamp          2018-06-11 00:00:43.032000
user_id      bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx
goods_id                                1414
shop_id                                   38
price                                  45000
hour                                       0
n_user_id                               6241
Name: 0, dtype: object
```

위의 user_id에 해당하는 고객의 log기록을 가져와주세요. 결과는 아래와 같습니다.

	timestamp	user_id	event_origin	event_name	event_goods_id	event_shop_id	n_user_id
878	2018-06-11 00:06:45.357	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨	app_page_view	NaN	NaN	6241
901	2018-06-11 00:06:54.034	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	app_page_view	NaN	NaN	6241
1062	2018-06-11 00:08:00.579	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	enter_browser	2048.0	46.0	6241
1259	2018-06-11 00:09:38.881	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	app_page_view	NaN	NaN	6241
1439	2018-06-11 00:11:04.446	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	enter_browser	3486.0	38.0	6241
1473	2018-06-11 00:11:20.354	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	app_page_view	NaN	NaN	6241
1526	2018-06-11 00:11:48.284	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	enter_browser	4006.0	24.0	6241
2423	2018-06-11 00:18:21.906	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	app_page_view	NaN	NaN	6241
2529	2018-06-11 00:19:01.928	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨	app_page_view	NaN	NaN	6241
2758	2018-06-11 00:20:30.432	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	shops_bookmark	app_page_view	NaN	NaN	6241
4502	2018-06-11 00:32:29.738	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	shops_bookmark	app_page_view	NaN	NaN	6241

5156	2018-06-11 00:37:22.757	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	shops_bookmark	app_page_view	NaN	NaN	6241

In [89]:
```
data_logs.loc[data_logs["user_id"] == "bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx"]
```

Out[89]:

	timestamp	user_id	event_origin	event_name	event_goods_id	event_shop_id	n_user_id
878	2018-06-11 00:06:45.357	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨	app_page_view	NaN	NaN	6241
901	2018-06-11 00:06:54.034	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	app_page_view	NaN	NaN	6241
1062	2018-06-11 00:08:00.579	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	enter_browser	2048.0	46.0	6241
1259	2018-06-11 00:09:38.881	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	app_page_view	NaN	NaN	6241
1439	2018-06-11 00:11:04.446	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	enter_browser	3486.0	38.0	6241
1473	2018-06-11 00:11:20.354	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	app_page_view	NaN	NaN	6241
1526	2018-06-11 00:11:48.284	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	enter_browser	4006.0	24.0	6241
2423	2018-06-11 00:18:21.906	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨 바지	app_page_view	NaN	NaN	6241
2529	2018-06-11 00:19:01.928	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	goods_search_result/린넨	app_page_view	NaN	NaN	6241
2758	2018-06-11 00:20:30.432	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	shops_bookmark	app_page_view	NaN	NaN	6241
4502	2018-06-11 00:32:29.738	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	shops_bookmark	app_page_view	NaN	NaN	6241
5156	2018-06-11 00:37:22.757	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	shops_bookmark	app_page_view	NaN	NaN	6241

로그 데이터에서 00시 37분에 마지막 log가 기록되어 있는 것을 확인할 수 있습니다. 또한 주문 데이터에서 해당 고객이 00시 43분에 구매를 한 것을 확인 할 수 있습니다. 이 기록을 바탕으로 해당 고객은 쇼핑몰 즐겨찾기 목록에서 특정 쇼핑몰을 클릭하여 들어간다음 43분에 린넨바지를 구매한 것을 유추할 수 있습니다.

이 기록들은 구매 고객을 tracking할 때 매우 중요하지만 각 각 나누어 기록되어 있습니다. 따라서 해당 로그데이터와 주문데이터를 concat 해야됩니다.

하지만 log기록엔 없는 구매 고객들이 있습니다. 이들은 전날 기록이 넘어 온 것 고객들로 유추됩니다. 이들을 확인한 다음 제외하고 concat 하도록 하겠습니다.

로그 데이터와 주문 데이터에 동시에 기록된 고객수를 확인하는데에는 집합 타입 변수(set)의 교집합 연산(&)을 이용하겠습니다.

데이터를 집합으로 변환하면 중복된 원소들은 1개만 남고 반환됩니다. unique() 메서드의 결과와 같다고 생각하시면 됩니다.

코드예시는 아래와 같습니다.

```
#집합 타입으로 변환하는 코드입니다.
set(data['column'])

#집합의 원소의 개수를 반환하는 코드입니다.
len(set 타입 변수)

#두 집합의 공통된 원소들의 집합을 반환하는 코드입니다.
(set 타입 변수) & (set 타입 변수)
```

In [90]:
```
# 주문한 고객의 수를 user_number_order란 변수에 저장해주세요.
user_set = set(order['user_id'])
user_number_order = len(user_set)

# log에 기록된 고객의 수를 user_number_log란 변수에 저장해주세요.
log_set = set(data_logs['user_id'])
user_number_log = len(log_set)

# 중복된 고객의 수를 user_duplicated란 변수에 저장해주세요.
user_duplicated = user_set & log_set

# 결과를 출력합니다.
print('해당 날짜에 구매한 총 고객수 입니다 :',user_number_order, end ='명\n')
print('해당 날짜 log데이터에 기록된 총 고객수입니다 :',user_number_log,  end ='명\n')
print('중복되는 고객수입니다 :', len(user_duplicated) , end ='명\n')
```

```
해당 날짜에 구매한 총 고객수 입니다 : 832명
해당 날짜 log데이터에 기록된 총 고객수입니다 : 9909명
중복되는 고객수입니다 : 742명
```

In [91]:
```
# 판다스의 merge를 이용할 수도 있습니다.
log_order_user = pd.merge(data_logs, order, on = 'user_id', how = 'inner')

user_duplicated = log_order_user['user_id'].unique()

print('중복되는 고객수입니다 :', len(user_duplicated) , end ='명\n')
```

```
중복되는 고객수입니다 : 742명
```

중복되는 고객의 목록을 구하였으므로 이제는 주문데이터와 로그데이터를 concat하기 위해 column명을 동일하게 맞추겠습니다.

우선 order, data_logs의 column을 확인하겠습니다.

In [92]:
```
print(order.columns)
print(data_logs.columns)
```

```
Index(['timestamp', 'user_id', 'goods_id', 'shop_id', 'price', 'hour',
       'n_user_id'],
      dtype='object')
Index(['timestamp', 'user_id', 'event_origin', 'event_name', 'event_goods_id',
       'event_shop_id', 'n_user_id'],
      dtype='object')
```

data_logs의 columns을 기준으로 병합할 것입니다. 따라서 order의 columns을 data_logs에 맞춰 변형합니다.

In [93]:
```
# order 원본을 변형하지 않기 위해 order을 copy하여 사용합니다.
order_copy = order.copy()
```

In [94]:
```
# 겹치는 유저만으로 data를 indexing합니다. user_duplicated 이용합니다.
# isin을 이용한 indexing은 뒤의 문제에서도 많이 나오기 때문에 유심히 봐두시면 유용합니다.
order_copy = order_copy[order_copy['user_id'].isin(user_duplicated)]

#  event_origin 컬럼에는 shop_id 컬럼을 저장합니다.
order_copy['event_origin'] = order_copy['shop_id']

#  event_name 컬럼에는 'purchase'를 저장합니다.
order_copy['event_name'] = 'purchase'

# event_goods_id 컬럼에는 good_id 컬럼를 저장합니다.
order_copy['event_goods_id'] = order_copy['goods_id']

# 사용할 columns를 설정합니다.
order_copy = order_copy[['timestamp', 'n_user_id', 'user_id','event_origin',
         'event_name', 'event_goods_id', 'price']]

order_copy.head()
```

Out[94]:

	timestamp	n_user_id	user_id	event_origin	event_name	event_goods_id	price
0	2018-06-11 00:00:43.032	6241	bvu0aLTqiFDoU-963xnr5nzQWTNLUMjx	38	purchase	1414	45000
1	2018-06-11 00:02:33.763	8899	smDmRnykg61KajpxXKzQ0oNkrh2nuSBj	12	purchase	1351	9500
4	2018-06-11 00:05:26.010	7832	lq1Je3voA3a0MouSFba3629lKCvweI24	89	purchase	5572	29000
5	2018-06-11 00:05:35.182	2745	GM0-EsJPHjkpteIpAQIwaCdUjU81IhW1	22	purchase	55	11200
6	2018-06-11 00:06:14.314	7800	lgvWxrv7r5RGklXSJqM2x6NUBZ5H-RQZ	22	purchase	2451	19800

In [95]:
```
# data_logs, order_copy를 concat 하고 log_order에 저장해주세요.

log_order = pd.concat([data_logs, order_copy], sort = False)

log_order.shape
```

Out[95]: (106587, 8)

앞으로의 분석은 log_order을 사용하여 진행할 것입니다. 따라서 별도의 설명이 없으면 log_order를 사용해주세요.

---

log_order에 구매기록여부 컬럼인 purchase 컬럼을 만들어주세요.

price컬럼을 이용해주세요.

출력 결과는 아래와 같습니다.

	event_goods_id	event_name	event_origin	event_shop_id	n_user_id	price	timestamp	user_id	purcha
0	NaN	app_page_view	shops_ranking	NaN	3314	NaN	2018-06-11 00:00:00.213	K1d8_t3-QIskaSkrx32oAFu856D8JmLo	False
1	NaN	app_page_view	shops_bookmark	NaN	7844	NaN	2018-06-11 00:00:00.810	lwFZ77v_ygk0uU40t1ud3l30EZ6sE2R3	False
2	NaN	app_page_view	goods_search_result/로브	NaN	7920	NaN	2018-06-11 00:00:00.956	mR-bO6hC9g-m8ERXMRQZaRwJFvzNNdd8	False
3	NaN	app_page_view	shops_bookmark	NaN	3314	NaN	2018-06-11 00:00:01.084	K1d8_t3-QIskaSkrx32oAFu856D8JmLo	False
4	NaN	app_page_view	shops_bookmark	NaN	5608	NaN	2018-06-11 00:00:01.561	Yjny5AchUWLiuv4kdeq50COF-S8OFXPd	False

---

In [96]:
```
log_order['purchase'] = log_order["price"].notnull()

log_order.head()
```

Out[96]:

	timestamp	user_id	event_origin	event_name	event_goods_id	event_shop_id	n_user_id	price	purch
0	2018-06-11 00:00:00.213	K1d8_t3-QIskaSkrx32oAFu856D8JmLo	shops_ranking	app_page_view	NaN	NaN	3314	NaN	F
1	2018-06-11 00:00:00.810	lwFZ77v_ygk0uU40t1ud3l30EZ6sE2R3	shops_bookmark	app_page_view	NaN	NaN	7844	NaN	F
2	2018-06-11 00:00:00.956	mR-bO6hC9g-m8ERXMRQZaRwJFvzNNdd8	goods_search_result/로브	app_page_view	NaN	NaN	7920	NaN	F
3	2018-06-11 00:00:01.084	K1d8_t3-QIskaSkrx32oAFu856D8JmLo	shops_bookmark	app_page_view	NaN	NaN	3314	NaN	F
4	2018-06-11 00:00:01.561	Yjny5AchUWLiuv4kdeq50COF-S8OFXPd	shops_bookmark	app_page_view	NaN	NaN	5608	NaN	F

데이터를 파악하기 쉽게 하기 위해 user_id, timestamp 컬럼을 기준으로 log_order를 정렬해주세요.

sort_values()를 사용해주시고 reset_index()를 통해 index를 정리해주세요.

출력 결과는 아래와 같습니다.

	event_goods_id	event_name	event_origin	event_shop_id	n_user_id	price	timestamp	user_id	purchase
0	NaN	app_page_view	shops_bookmark	NaN	0	NaN	2018-06-11 15:57:10.615	--PYPMX8QWg0ioT5zfORmU-S5Ln0lot	False
1	NaN	app_page_view	shops_bookmark	NaN	0	NaN	2018-06-11 15:59:05.505	--PYPMX8QWg0ioT5zfORmU-S5Ln0lot	False
2	NaN	app_page_view	my_goods	NaN	1	NaN	2018-06-11 00:55:37.309	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	False
3	2506.0	enter_browser	my_goods	40.0	1	NaN	2018-06-11 00:55:44.430	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	False
4	NaN	app_page_view	my_goods	NaN	1	NaN	2018-06-11 01:00:33.295	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	False

In [97]:
```
log_order = log_order.sort_values(by = ['user_id', 'timestamp']).reset_index(drop =True)

log_order.head()
```

Out[97]:

	timestamp	user_id	event_origin	event_name	event_goods_id	event_shop_id	n_user_id	price	purchase
0	2018-06-11 15:57:10.615	--PYPMX8QWg0ioT5zfORmU-S5Ln0lot	shops_bookmark	app_page_view	NaN	NaN	0	NaN	False
1	2018-06-11 15:59:05.505	--PYPMX8QWg0ioT5zfORmU-S5Ln0lot	shops_bookmark	app_page_view	NaN	NaN	0	NaN	False

2	2018-06-11 00:55:37.309	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	my_goods	app_page_view	NaN	NaN	1	NaN	False
3	2018-06-11 00:55:44.430	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	my_goods	enter_browser	2506.0	40.0	1	NaN	False
4	2018-06-11 01:00:33.295	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	my_goods	app_page_view	NaN	NaN	1	NaN	False

13. page duration 을 구해주세요.

page duration은 사용자가 앱의 한 page당 체류하는 시간입니다. 이는 동일한 사용자에 대한 연속한 로그들 사이의 시간 간격을 뜻합니다.

이를 구하기 위하여, 로그별로 연속된 다음 로그의 timestamp를 저장하는timestamp_after라는 컬럼을 만들고 timestamp와의 차이를 계산할 것입니다.

현재 제공된 데이터에서는 고객이 앱을 종료하는 기록이 없기 때문에 마지막 log의 page_duration은 0이라고 가정을 합니다.

---

timestamp_after 컬럼을 만들어줍니다.

고객별로 groupby()한 뒤 shift(-1)을 적용한 결과를 timestamp_after에 저장합니다.

groupby()를 하고 shift()를 하면 고객의 당일 마지막 로그의 timestamp_after은 NaT(Not a Time)값을 갖게 됩니다.

결과는 아래와 같습니다.

	event_goods_id	event_name	event_origin	event_shop_id	n_user_id	price	timestamp	user_id	purchase	ti
0	NaN	app_page_view	shops_bookmark	NaN	0	NaN	2018-06-11 15:57:10.615	--PYPMX8QWg0ioT5zfORmU-S5LIn0lot	False	2(1:
1	NaN	app_page_view	shops_bookmark	NaN	0	NaN	2018-06-11 15:59:05.505	--PYPMX8QWg0ioT5zfORmU-S5LIn0lot	False	N
2	NaN	app_page_view	my_goods	NaN	1	NaN	2018-06-11 00:55:37.309	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	False	2(0(
3	2506.0	enter_browser	my_goods	40.0	1	NaN	2018-06-11 00:55:44.430	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	False	2(0
4	NaN	app_page_view	my_goods	NaN	1	NaN	2018-06-11 01:00:33.295	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	False	2(0

In [98]:
```
# timestamp_after 컬럼을 다음과 같이 만듭니다.

log_order['timestamp_after'] = log_order.groupby(['n_user_id'])['timestamp'].shift(-1)

log_order.head()
```

Out[98]:

	timestamp	user_id	event_origin	event_name	event_goods_id	event_shop_id	n_user_id	price	purchase	t
0	2018-06-11 15:57:10.615	--PYPMX8QWg0ioT5zfORmU-S5LIn0lot	shops_bookmark	app_page_view	NaN	NaN	0	NaN	False	
1	2018-06-11 15:59:05.505	--PYPMX8QWg0ioT5zfORmU-S5LIn0lot	shops_bookmark	app_page_view	NaN	NaN	0	NaN	False	
2	2018-06-11 00:55:37.309	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	my_goods	app_page_view	NaN	NaN	1	NaN	False	
3	2018-06-11 00:55:44.430	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	my_goods	enter_browser	2506.0	40.0	1	NaN	False	
4	2018-06-11 01:00:33.295	-16-xXbeDcvkZJtTpRwMi57Yo2ZQpORv	my_goods	app_page_view	NaN	NaN	1	NaN	False	

이제 timestamp_after과 timestamp의 차이를 계산하여 page_duration을 구한후 . NaT 값은 연산시 NaN값이 됩니다. 이후에 NaN값을 0으로 채워주시면 됩니다.

출력 결과는 아래와 같습니다.

	timestamp	timestamp_after	page_duration
0	2018-06-11 15:57:10.615	2018-06-11 15:59:05.505	00:01:54.890000
1	2018-06-11 15:59:05.505	NaT	00:00:00
2	2018-06-11 00:55:37.309	2018-06-11 00:55:44.430	00:00:07.121000
3	2018-06-11 00:55:44.430	2018-06-11 01:00:33.295	00:04:48.865000

4	2018-06-11 01:00:33.295	2018-06-11 01:11:03.608	00:10:30.313000

In [99]:
```
log_order['page_duration'] = (log_order['timestamp_after'] - log_order['timestamp']).fillna(0)

log_order[['timestamp','timestamp_after','page_duration']].head()
```

Out[99]:

	timestamp	timestamp_after	page_duration
0	2018-06-11 15:57:10.615	2018-06-11 15:59:05.505	00:01:54.890000
1	2018-06-11 15:59:05.505	NaT	00:00:00
2	2018-06-11 00:55:37.309	2018-06-11 00:55:44.430	00:00:07.121000
3	2018-06-11 00:55:44.430	2018-06-11 01:00:33.295	00:04:48.865000
4	2018-06-11 01:00:33.295	2018-06-11 01:11:03.608	00:10:30.313000

분석의 편의를 위하여 map()과 total_seconds() 메서드를 이용하여 page_duration을 초로 환산해주세요.

마지막으로, 이 후의 계산을 위해 astype() 메서드를 이용하여 page_duration 컬럼을 float으로 변환해주세요.

출력 결과는 아래와 같습니다.

	n_user_id	timestamp	timestamp_after	page_duration
0	0	2018-06-11 15:57:10.615	2018-06-11 15:59:05.505	114.890
1	0	2018-06-11 15:59:05.505	NaT	0.000
2	1	2018-06-11 00:55:37.309	2018-06-11 00:55:44.430	7.121
3	1	2018-06-11 00:55:44.430	2018-06-11 01:00:33.295	288.865
4	1	2018-06-11 01:00:33.295	2018-06-11 01:11:03.608	630.313

In [100…
```
log_order['page_duration'] = log_order['page_duration'].map(lambda x: x.total_seconds()).astype('float')
#log_order['page_duration'].dt.total_seconds()

log_order[['n_user_id','timestamp', 'timestamp_after', 'page_duration']].head()
```

Out[100…

	n_user_id	timestamp	timestamp_after	page_duration
0	0	2018-06-11 15:57:10.615	2018-06-11 15:59:05.505	114.890
1	0	2018-06-11 15:59:05.505	NaT	0.000
2	1	2018-06-11 00:55:37.309	2018-06-11 00:55:44.430	7.121
3	1	2018-06-11 00:55:44.430	2018-06-11 01:00:33.295	288.865
4	1	2018-06-11 01:00:33.295	2018-06-11 01:11:03.608	630.313

14. session을 구해주세요.

사용자가 앱을 실행하는 단위를 세션(session)이라고 정의합니다. 세션은 사용자가 앱을 실행한 후부터 그 실행을 마칠 때까지의 일련의 과정을 포함합니다.

session을 구하기 위해서 고객이 session을 종료하고 앱을 나갔는지 여부(boolean)인 is_out이라는 컬럼을 만듭니다.

is_out의 조건은 아래와 같이 2가지가 있습니다.

```
조건 1) page_duration이 0 이면 고객의 당일 마지막 log이기 때문에 고객이 session을 종료하고 앱을 나갔다고 가정합니다. (page_duration 설명 부분에서 가정하였습니다.)

조건 2) page_duration이 40분이상이면 고객이 한 session을 종료한 것이라고 가정하겠습니다.
```

따라서 위의 조건중 적어도 1개를 만족시키키면 True되도록 is_out 컬럼을 생성해주세요.

출력 결과는 아래와 같습니다.

	n_user_id	timestamp	timestamp_after	page_duration	is_out
0	0	2018-06-11 15:57:10.615	2018-06-11 15:59:05.505	114.890	False

1	0	2018-06-11 15:59:05.505	NaT	0.000	True
2	1	2018-06-11 00:55:37.309	2018-06-11 00:55:44.430	7.121	False
3	1	2018-06-11 00:55:44.430	2018-06-11 01:00:33.295	288.865	False
4	1	2018-06-11 01:00:33.295	2018-06-11 01:11:03.608	630.313	False

In [101…

```
# 30분을 초로 환산해주셔야 합니다.

log_order["is_out"] = (log_order["page_duration"] > 2400) | (log_order["page_duration"] == 0 )

log_order[['n_user_id','timestamp', 'timestamp_after', 'page_duration', 'is_out']].head()
# data_logs.loc[data_logs['page_duration'] ==0, "is_out"] = True
```

Out[101…

	n_user_id	timestamp	timestamp_after	page_duration	is_out
0	0	2018-06-11 15:57:10.615	2018-06-11 15:59:05.505	114.890	False
1	0	2018-06-11 15:59:05.505	NaT	0.000	True
2	1	2018-06-11 00:55:37.309	2018-06-11 00:55:44.430	7.121	False
3	1	2018-06-11 00:55:44.430	2018-06-11 01:00:33.295	288.865	False
4	1	2018-06-11 01:00:33.295	2018-06-11 01:11:03.608	630.313	False

page_duration이 2400(40분)이상 또는 0인 log들은 session의 마지막 log입니다. 예를 들어, page_duration이 36000인 log가 있으면 이는 10시간 뒤에 다시 app에 접속한다는 것을 의미하는 것이지 실제 10시간 동안 앱을 사용했다는 의미가 아닙니다. 따라서 이 사람에 대한 해당 log는 해당 session의 마지막 log가 되고 다음 log는 10시간 뒤인 다음 session의 첫 log가 됩니다.

이제 is_out 컬럼을 만들었으니 is_out이 True인 log들, 즉 session의 마지막 log들의 page_duration은 0이 되도록 변환해주어야 합니다.

is_out 컬럼이 True인 page_duration을 0으로 지정해주세요.

In [102…

```
log_order.loc[log_order['is_out'], 'page_duration'] = 0

log_order[['n_user_id','timestamp', 'timestamp_after', 'page_duration', 'is_out']].head(20)
```

Out[102…

	n_user_id	timestamp	timestamp_after	page_duration	is_out
0	0	2018-06-11 15:57:10.615	2018-06-11 15:59:05.505	114.890	False
1	0	2018-06-11 15:59:05.505	NaT	0.000	True
2	1	2018-06-11 00:55:37.309	2018-06-11 00:55:44.430	7.121	False
3	1	2018-06-11 00:55:44.430	2018-06-11 01:00:33.295	288.865	False
4	1	2018-06-11 01:00:33.295	2018-06-11 01:11:03.608	630.313	False
5	1	2018-06-11 01:11:03.608	2018-06-11 01:11:05.713	2.105	False
6	1	2018-06-11 01:11:05.713	2018-06-11 01:12:09.565	63.852	False
7	1	2018-06-11 01:12:09.565	2018-06-11 01:12:28.850	19.285	False
8	1	2018-06-11 01:12:28.850	NaT	0.000	True
9	2	2018-06-11 02:21:04.848	2018-06-11 02:21:18.719	13.871	False
10	2	2018-06-11 02:21:18.719	2018-06-11 02:27:23.809	365.090	False
11	2	2018-06-11 02:27:23.809	2018-06-11 02:28:42.663	78.854	False
12	2	2018-06-11 02:28:42.663	2018-06-11 02:30:24.883	102.220	False
13	2	2018-06-11 02:30:24.883	2018-06-11 02:31:26.044	61.161	False
14	2	2018-06-11 02:31:26.044	2018-06-11 02:31:26.867	0.823	False
15	2	2018-06-11 02:31:26.867	2018-06-11 02:34:30.616	183.749	False
16	2	2018-06-11 02:34:30.616	2018-06-11 02:34:45.644	15.028	False
17	2	2018-06-11 02:34:45.644	2018-06-11 02:35:08.966	23.322	False
18	2	2018-06-11 02:35:08.966	2018-06-11 02:35:39.187	30.221	False
19	2	2018-06-11 02:35:39.187	2018-06-11 02:35:54.009	14.822	False

---

이제 session을 구하여 번호를 부여할 것입니다.

번호는 session별 고유 번호(session_idx_unique)와 일(day)마다 0부터 시작하는 daily session 번호(session_idx_daily)를 부여합니다.

보다 쉽은 이해를 위하여 각 메서드를 적용하는 단계마다 column을 추가하고 전체 결과를 출력하였습니다.

In [103…

```
log_order['is_out-cumsum()'] = log_order['is_out'].cumsum()
log_order["is_out-cumsum()-shift(1)"] = log_order['is_out-cumsum()'].shift(1)
log_order["is_out-cumsum()-shift(1)-fillna(0)"] = log_order["is_out-cumsum()-shift(1)"].fillna(0)
log_order["is_out-cumsum()-shift(1)-fillna(0)-astype(int)"] = log_order["is_out-cumsum()-shift(1)-fillna(0)"].ast

log_order['session_idx_unique'] = (log_order['is_out']
                                   .cumsum() # 컬럼의 누적 합계를 반환합니다.
                                   .shift(1)
                                   .fillna(0)
                                   .astype(int)
                                  )

log_order[['n_user_id','timestamp', 'timestamp_after', 'page_duration', 'is_out','is_out','is_out-cumsum()','is_
```

Out[103…

	n_user_id	timestamp	timestamp_after	page_duration	is_out	is_out	is_out-cumsum()	is_out-cumsum()-shift(1)	is_out-cumsum()-shift(1)-fillna(0)	is_out-cumsum()-shift(1)-fillna(0)-astype(int)	session_idx_unique
0	0	2018-06-11 15:57:10.615	2018-06-11 15:59:05.505	114.890	False	False	0	NaN	0.0	0	0
1	0	2018-06-11 15:59:05.505	NaT	0.000	True	True	1	0.0	0.0	0	0
2	1	2018-06-11 00:55:37.309	2018-06-11 00:55:44.430	7.121	False	False	1	1.0	1.0	1	1
3	1	2018-06-11 00:55:44.430	2018-06-11 01:00:33.295	288.865	False	False	1	1.0	1.0	1	1
4	1	2018-06-11 01:00:33.295	2018-06-11 01:11:03.608	630.313	False	False	1	1.0	1.0	1	1
5	1	2018-06-11 01:11:03.608	2018-06-11 01:11:05.713	2.105	False	False	1	1.0	1.0	1	1
6	1	2018-06-11 01:11:05.713	2018-06-11 01:12:09.565	63.852	False	False	1	1.0	1.0	1	1
7	1	2018-06-11 01:12:09.565	2018-06-11 01:12:28.850	19.285	False	False	1	1.0	1.0	1	1
8	1	2018-06-11 01:12:28.850	NaT	0.000	True	True	2	1.0	1.0	1	1
9	2	2018-06-11 02:21:04.848	2018-06-11 02:21:18.719	13.871	False	False	2	2.0	2.0	2	2
10	2	2018-06-11 02:21:18.719	2018-06-11 02:27:23.809	365.090	False	False	2	2.0	2.0	2	2
11	2	2018-06-11 02:27:23.809	2018-06-11 02:28:42.663	78.854	False	False	2	2.0	2.0	2	2
12	2	2018-06-11 02:28:42.663	2018-06-11 02:30:24.883	102.220	False	False	2	2.0	2.0	2	2
13	2	2018-06-11 02:30:24.883	2018-06-11 02:31:26.044	61.161	False	False	2	2.0	2.0	2	2
14	2	2018-06-11 02:31:26.044	2018-06-11 02:31:26.867	0.823	False	False	2	2.0	2.0	2	2
15	2	2018-06-11 02:31:26.867	2018-06-11 02:34:30.616	183.749	False	False	2	2.0	2.0	2	2
16	2	2018-06-11 02:34:30.616	2018-06-11 02:34:45.644	15.028	False	False	2	2.0	2.0	2	2
17	2	2018-06-11 02:34:45.644	2018-06-11 02:35:08.966	23.322	False	False	2	2.0	2.0	2	2
18	2	2018-06-11 02:35:08.966	2018-06-11 02:35:39.187	30.221	False	False	2	2.0	2.0	2	2
19	2	2018-06-11 02:35:39.187	2018-06-11 02:35:54.009	14.822	False	False	2	2.0	2.0	2	2
20	2	2018-06-11 02:35:54.009	2018-06-11 02:35:58.590	4.581	False	False	2	2.0	2.0	2	2
21	2	2018-06-11 02:35:58.590	2018-06-11 16:15:23.793	0.000	True	True	3	2.0	2.0	2	2
22	2	2018-06-11 16:15:23.793	2018-06-11 16:15:36.161	12.368	False	False	3	3.0	3.0	3	3
23	2	2018-06-11 16:15:36.161	2018-06-11 16:16:17.091	40.930	False	False	3	3.0	3.0	3	3
24	2	2018-06-11 16:16:17.091	2018-06-11 16:16:33.411	16.320	False	False	3	3.0	3.0	3	3
25	2	2018-06-11 16:16:33.411	2018-06-11 22:27:56.017	0.000	True	True	4	3.0	3.0	3	3
26	2	2018-06-11	2018-06-11	2.903	False	False	4	4.0	4.0	4	4

		22:27:56.017	22:27:58.920								
27	2	2018-06-11 22:27:58.920	2018-06-11 22:27:59.632	0.712	False	False	4	4.0	4.0	4	4
28	2	2018-06-11 22:27:59.632	2018-06-11 22:28:04.500	4.868	False	False	4	4.0	4.0	4	4
29	2	2018-06-11 22:28:04.500	2018-06-11 22:28:10.860	6.360	False	False	4	4.0	4.0	4	4

이번엔 daily session 번호를 부여하겠습니다. 위의 고유 번호와는 다르게

groupby() 메서드를 사용해야 합니다. 이 역시 보다 쉬운 이해를 위해 위와 같이 컬럼을 만들도록 하겠습니다.

In [104…

```
log_order.groupby('n_user_id')['is_out'].shift(1)
```

Out[104…

```
0        NaN
1      False
2        NaN
3      False
4      False
5      False
6      False
7      False
8      False
9        NaN
10     False
11     False
12     False
13     False
14     False
15     False
16     False
17     False
18     False
19     False
20     False
21     False
22      True
23     False
24     False
25     False
26      True
27     False
28     False
29     False
30     False
31     False
32     False
33     False
34     False
35     False
36     False
37     False
38     False
39     False
40     False
41     False
42     False
43     False
44     False
45     False
46     False
47     False
48     False
49     False
50     False
51     False
52     False
53     False
54     False
55     False
56     False
57       NaN
58      True
59     False
60     False
61     False
62     False
63     False
```

64 True
65 False
66 False
67 True
68 False
69 False
70 False
71 False
72 False
73 False
74 False
75 False
76 False
77 False
78 False
79 False
80 False
81 False
82 False
83 False
84 False
85 False
86 False
87 False
88 NaN
89 False
90 NaN
91 True
92 NaN
93 False
94 False
95 False
96 False
97 False
98 False
99 False
100 False
101 NaN
102 False
103 False
104 NaN
105 False
106 False
107 False
108 False
109 False
110 False
111 False
112 False
113 False
114 False
115 False
116 True
117 False
118 False
119 False
120 False
121 False
122 False
123 False
124 False
125 False
126 False
127 False
128 False
129 False
130 False
131 False
132 False
133 False
134 False
135 False
136 False
137 True
138 False
139 False
140 False
141 False
142 False
143 False
144 False
145 False
146 False

```
147       False
148       False
149       False
150       False
151       False
152       False
153       False
154       False
155       False
156       False
157       False
158       False
159       False
160       False
161       False
162        True
163       False
164       False
165       False
166       False
167       False
168       False
169       False
170       False
171       False
172       False
173       False
174       False
175       False
176       False
177       False
178       False
179        True
180       False
181       False
182       False
183       False
184       False
185       False
186       False
187       False
188       False
189       False
190       False
191       False
192       False
193       False
194       False
195       False
196       False
197       False
198       False
199       False
200       False
201       False
202       False
203         NaN
204       False
205       False
206       False
207       False
208       False
209       False
210       False
211       False
212       False
213       False
214       False
215       False
216       False
217         NaN
218       False
219       False
220         NaN
221       False
222       False
223       False
224       False
225         NaN
226       False
227       False
228       False
229         NaN
```

```
230       False
231       False
232         NaN
233        True
234       False
235        True
236         NaN
237       False
238       False
239       False
240       False
241       False
242       False
243       False
244       False
245       False
246       False
247       False
248         NaN
249       False
          ...
106337      NaN
106338    False
106339    False
106340    False
106341    False
106342    False
106343    False
106344      NaN
106345    False
106346      NaN
106347      NaN
106348    False
106349    False
106350      NaN
106351    False
106352    False
106353     True
106354    False
106355    False
106356    False
106357    False
106358    False
106359    False
106360      NaN
106361      NaN
106362    False
106363    False
106364    False
106365     True
106366    False
106367    False
106368    False
106369     True
106370    False
106371    False
106372    False
106373    False
106374    False
106375    False
106376    False
106377    False
106378    False
106379    False
106380    False
106381    False
106382    False
106383    False
106384    False
106385    False
106386    False
106387    False
106388    False
106389    False
106390     True
106391    False
106392    False
106393    False
106394    False
106395    False
106396     True
106397    False
106398    False
```

```
106399    False
106400    False
106401    False
106402    False
106403    False
106404    False
106405    False
106406    False
106407    False
106408    False
106409    False
106410    False
106411    False
106412    False
106413    False
106414    False
106415     True
106416    False
106417    False
106418     True
106419    False
106420    False
106421    False
106422    False
106423    False
106424    False
106425     True
106426    False
106427    False
106428    False
106429    False
106430    False
106431    False
106432    False
106433      NaN
106434    False
106435     True
106436    False
106437    False
106438    False
106439    False
106440    False
106441    False
106442    False
106443    False
106444    False
106445    False
106446    False
106447      NaN
106448    False
106449    False
106450    False
106451    False
106452    False
106453    False
106454    False
106455    False
106456    False
106457    False
106458    False
106459     True
106460    False
106461      NaN
106462      NaN
106463      NaN
106464    False
106465    False
106466    False
106467    False
106468      NaN
106469    False
106470    False
106471    False
106472    False
106473    False
106474    False
106475    False
106476    False
106477    False
106478    False
106479    False
106480    False
106481    False
```

```
106482    False
106483    False
106484    False
106485    False
106486    False
106487      NaN
106488    False
106489    False
106490    False
106491    False
106492    False
106493    False
106494    False
106495    False
106496    False
106497     True
106498    False
106499    False
106500    False
106501    False
106502    False
106503      NaN
106504    False
106505    False
106506      NaN
106507    False
106508    False
106509    False
106510    False
106511    False
106512    False
106513    False
106514      NaN
106515    False
106516    False
106517      NaN
106518    False
106519    False
106520    False
106521    False
106522    False
106523    False
106524    False
106525    False
106526    False
106527    False
106528    False
106529    False
106530    False
106531    False
106532    False
106533    False
106534    False
106535    False
106536    False
106537    False
106538    False
106539    False
106540    False
106541    False
106542    False
106543    False
106544    False
106545    False
106546    False
106547    False
106548    False
106549    False
106550    False
106551    False
106552    False
106553    False
106554    False
106555    False
106556    False
106557    False
106558      NaN
106559    False
106560    False
106561    False
106562    False
106563      NaN
106564    False
```

```
106565    False
106566    False
106567    False
106568    False
106569    False
106570    False
106571    False
106572    False
106573    False
106574    False
106575    False
106576    False
106577    False
106578     True
106579    False
106580    False
106581    False
106582    False
106583    False
106584    False
106585    False
106586    False
Name: is_out, Length: 106587, dtype: object
```

In [105…

```
log_order['is_out-cumsum()'] = log_order.groupby('n_user_id')['is_out'].cumsum()
log_order["is_out-cumsum()-shift(1)"] = log_order.groupby('n_user_id')['is_out'].cumsum().shift(1)
log_order["is_out-cumsum()-shift(1)-fillna(0)"] = log_order.groupby('n_user_id')['is_out'].cumsum().shift(1).fill
log_order["is_out-cumsum()-shift(1)-fillna(0)-astype(int)"] = log_order.groupby('n_user_id')['is_out'].cumsum().s



log_order['session_idx_daily'] = (log_order.groupby('n_user_id')['is_out']
                                           .cumsum()
                                           .shift(1)
                                           .fillna(0)
                                           .astype(int)
                                  )

# user마다 첫 session_idx_daily가 1로 되어있기 때문에 이를 0으로 바꿔줍니다.
head_index = log_order.groupby('n_user_id')['session_idx_daily'].head(1).index
log_order.loc[head_index, 'session_idx_daily'] = 0

log_order[['n_user_id','timestamp', 'timestamp_after', 'page_duration','is_out','is_out-cumsum()','is_out-cumsum(
```

Out[105…

	n_user_id	timestamp	timestamp_after	page_duration	is_out	is_out-cumsum()	is_out-cumsum()-shift(1)	is_out-cumsum()-shift(1)-fillna(0)	is_out-cumsum()-shift(1)-fillna(0)-astype(int)	session_idx_daily
0	0	2018-06-11 15:57:10.615	2018-06-11 15:59:05.505	114.890	False	0.0	NaN	0.0	0	0
1	0	2018-06-11 15:59:05.505	NaT	0.000	True	1.0	0.0	0.0	0	0
2	1	2018-06-11 00:55:37.309	2018-06-11 00:55:44.430	7.121	False	0.0	1.0	1.0	1	0
3	1	2018-06-11 00:55:44.430	2018-06-11 01:00:33.295	288.865	False	0.0	0.0	0.0	0	0
4	1	2018-06-11 01:00:33.295	2018-06-11 01:11:03.608	630.313	False	0.0	0.0	0.0	0	0
5	1	2018-06-11 01:11:03.608	2018-06-11 01:11:05.713	2.105	False	0.0	0.0	0.0	0	0
6	1	2018-06-11 01:11:05.713	2018-06-11 01:12:09.565	63.852	False	0.0	0.0	0.0	0	0
7	1	2018-06-11 01:12:09.565	2018-06-11 01:12:28.850	19.285	False	0.0	0.0	0.0	0	0
8	1	2018-06-11 01:12:28.850	NaT	0.000	True	1.0	0.0	0.0	0	0
9	2	2018-06-11 02:21:04.848	2018-06-11 02:21:18.719	13.871	False	0.0	1.0	1.0	1	0
10	2	2018-06-11 02:21:18.719	2018-06-11 02:27:23.809	365.090	False	0.0	0.0	0.0	0	0
11	2	2018-06-11 02:27:23.809	2018-06-11 02:28:42.663	78.854	False	0.0	0.0	0.0	0	0
12	2	2018-06-11 02:28:42.663	2018-06-11 02:30:24.883	102.220	False	0.0	0.0	0.0	0	0
13	2	2018-06-11 02:30:24.883	2018-06-11 02:31:26.044	61.161	False	0.0	0.0	0.0	0	0

14	2	2018-06-11 02:31:26.044	2018-06-11 02:31:26.867	0.823	False	0.0	0.0	0.0	0	0
15	2	2018-06-11 02:31:26.867	2018-06-11 02:34:30.616	183.749	False	0.0	0.0	0.0	0	0
16	2	2018-06-11 02:34:30.616	2018-06-11 02:34:45.644	15.028	False	0.0	0.0	0.0	0	0
17	2	2018-06-11 02:34:45.644	2018-06-11 02:35:08.966	23.322	False	0.0	0.0	0.0	0	0
18	2	2018-06-11 02:35:08.966	2018-06-11 02:35:39.187	30.221	False	0.0	0.0	0.0	0	0
19	2	2018-06-11 02:35:39.187	2018-06-11 02:35:54.009	14.822	False	0.0	0.0	0.0	0	0
20	2	2018-06-11 02:35:54.009	2018-06-11 02:35:58.590	4.581	False	0.0	0.0	0.0	0	0
21	2	2018-06-11 02:35:58.590	2018-06-11 16:15:23.793	0.000	True	1.0	0.0	0.0	0	0
22	2	2018-06-11 16:15:23.793	2018-06-11 16:15:36.161	12.368	False	1.0	1.0	1.0	1	1
23	2	2018-06-11 16:15:36.161	2018-06-11 16:16:17.091	40.930	False	1.0	1.0	1.0	1	1
24	2	2018-06-11 16:16:17.091	2018-06-11 16:16:33.411	16.320	False	1.0	1.0	1.0	1	1
25	2	2018-06-11 16:16:33.411	2018-06-11 22:27:56.017	0.000	True	2.0	1.0	1.0	1	1
26	2	2018-06-11 22:27:56.017	2018-06-11 22:27:58.920	2.903	False	2.0	2.0	2.0	2	2
27	2	2018-06-11 22:27:58.920	2018-06-11 22:27:59.632	0.712	False	2.0	2.0	2.0	2	2
28	2	2018-06-11 22:27:59.632	2018-06-11 22:28:04.500	4.868	False	2.0	2.0	2.0	2	2
29	2	2018-06-11 22:28:04.500	2018-06-11 22:28:10.860	6.360	False	2.0	2.0	2.0	2	2

이제 session을 이용한 분석을 해보겠습니다.

session별 log 수(접속별 활동 개수), **user별 session당 평균 log수(고객별 접속당 평균 활동수)**을 구하겠습니다.

session별 log 수를 구하는 코드는 아래와 같습니다.

이를 활용하여 `14.2)user별 session당 평균 log수` 를 구해주세요.

14.1) session별 log 수(접속별 활동 개수)

In [106…

```
session_log_count = (log_order
                     .groupby(['n_user_id', 'session_idx_daily'])
                     .size()  # 그룹별 속한 row 수(log 수)를 반환합니다.
                     .reset_index()  # index를 초기하여 groupby object가 아닌 일반적인 data frame형태로 바꿉니다.
                     .rename(columns = { 0 : "log_count"} )   # 컬럼이름을 log_count로 변경합니다.
                    )



session_log_count.head()
```

Out[106…

	n_user_id	session_idx_daily	log_count
0	0	0	2
1	1	0	7
2	2	0	13
3	2	1	4
4	2	2	31

14.2) user별 session당 log수(고객별 접속당 평균 활동수)의 평균

In [107…

```
session_user_log_count = (
                          session_log_count
                          .groupby('n_user_id')['log_count']
                          .mean()
```

```
                        .reset_index()
                        .rename(columns = {'log_count' : 'log_count_mean'})
                    )

session_user_log_count.head(20)
```

Out[107…

	n_user_id	log_count_mean
0	0	2.000000
1	1	7.000000
2	2	16.000000
3	3	7.750000
4	4	2.000000
5	5	1.000000
6	6	9.000000
7	7	3.000000
8	8	19.800000
9	9	14.000000
10	10	3.000000
11	11	5.000000
12	12	4.000000
13	13	3.000000
14	14	1.333333
15	15	12.000000
16	16	6.000000
17	17	3.333333
18	18	9.000000
19	19	4.000000

In [116…

```
# 접속별 평균 활동수가 가장 많은 상위 5명의 user를 구합니다.
session_user_log_count.sort_values(by = 'log_count_mean').tail()
```

Out[116…

	n_user_id	log_count_mean
3643	3675	76.0
4181	4215	81.0
6085	6140	81.0
1421	1435	97.0
1123	1135	151.0

14.3) 하루동안 가장 많은 session을 갖는(가장 많이 활동한) 상위 5명의 user를 구해주세요.

`Hint) groupby(), nunique(), max() 메서드`

In [108…

```
(log_order
 .groupby(["n_user_id"])['session_idx_daily']
 .nunique()
 .sort_values()
 .tail()
)
```

Out[108…

```
n_user_id
6234     9
2424     9
2249    10
6010    10
5847    11
Name: session_idx_daily, dtype: int64
```

15. 기준별 체류 시간을 구해주세요.

체류시간이란 고객이 앱에서 머문 시간을 뜻합니다.

체류시간이 높다는 것은 사이트 운영의 청신호라고 생각할 수 있습니다. 일단 방문 목적과 랜딩페이지에서 제공되는 컨텐츠가 부합한다는 뜻이며, 웹사이트의 컨텐츠에 흥미를 느낀 방문자들이 계속 머물고 있다는 뜻이기도 합니다. 방문자들이 웹사이트에 오랜 시간 머물게 되면 웹사이트에서 제공하는 다양한 장치들을 접할 기회가 많아지고 전환에 도달할 확률이 더욱 높아지기 때문에 체류시간은 전환에 있어 매우 중요한 요소입니다. 광고를 고객을 통해 app으로의 유입을 성공해다면 그 다음 목표는 방문자를 계속 머물게 하여 전환으로 이어질 수 있도록 하는 것입니다.

해당 수업의 zigzag 데이터는 짧은 시간의 제한된 데이터라 체류시간을 이용하여 분석할 수 있는게 많지 않지만 현업에서는 체류시간을 이용하여 통해 어떤 채널을 이용한 고객 또는 어떤 광고를 통해 유입된 고객이 웹사이트/app에 오래 머물고 제품을 구매하는지에 대한 분석 또는 시간대/요일별 노출전략을 세우는 등 다양한 insight를 얻을 수 있습니다.

두가지 기준으로 체류시간을 구할 것입니다.

```
1) user별
2) 구매 user/ 비구매 유저
```

15.1) user별 체류시간을 구해주세요.

이는 두 단계로 구분됩니다.

1. session별 체류시간 구하기.
2. session별 체류시간을 바탕으로 user별 체류시간 구하기

먼저 session별 체류시간을 다음과 같이 구합니다. session 번호별 page_duration들의 합을 구하면 됩니다.

In [109…

```
duration_session = (log_order
                        .groupby(['n_user_id', 'session_idx_daily'])['page_duration']
                        .sum()
                        .reset_index()
                        .rename(columns = {'page_duration' : 'duration'}))

duration_session.head(10)
```

Out[109…

	n_user_id	session_idx_daily	duration
0	0	0	114.890
1	1	0	1011.541
2	2	0	893.742
3	2	1	69.618
4	2	2	3075.422
5	3	0	0.000
6	3	1	45.911
7	3	2	85.183
8	3	3	1711.031
9	4	0	49.130

이제 duration_session을 이용하여 user별 체류시간을 구해주세요.

`Hint) groupby(), mean() 메서드`

결과는 아래와 같습니다.

	duration
n_user_id	
0	114.89000
1	1011.54100
2	463.03625
3	460.53125
4	49.13000

In [110…

```
duration_user = duration_session.groupby("n_user_id")['duration'].mean()

duration_user.to_frame().head()
```

Out[110…

	duration

	n_user_id
0	114.890000
1	1011.541000
2	1346.260667
3	460.531250
4	49.130000

잔존 시간이 0인 고객들은 app에 들어와서 아무것도 안하고 나간 고객들입니다.

제외하고 계산할 수도 있습니다.

15.2) 구매/비구매 session별 평균 체류시간 구하기

구매 기록이 있는 session은 체류시간이 길 것이라고 예상할 수 있습니다. 이를 확인하기 위하여 구매/비구매 session별 평균 체류시간을 구하겠습니다.

먼저 구매기록이 있는 session list를 만들어 주세요.

session_purchse 변수에 구매기록이 있는 session들을 저장해주시면 됩니다.

`Hint) purchase 컬럼, indexing, unique() 메서드`

In [111…

```
session_purchase = log_order.loc[log_order['purchase']==True, 'session_idx_unique'].unique()

session_purchase
```

Out[111…

```
array([    4,    15,    16,    19,    88,    92,   123,   168,   192,
        232,   234,   337,   393,   397,   416,   429,   437,   491,
        500,   516,   554,   566,   592,   618,   626,   631,   682,
        707,   708,   727,   742,   756,   783,   792,   801,   819,
        832,   853,   860,   872,   943,  1008,  1020,  1053,  1088,
       1093,  1135,  1144,  1184,  1210,  1230,  1261,  1283,  1291,
       1324,  1330,  1384,  1386,  1404,  1413,  1422,  1455,  1461,
       1537,  1539,  1595,  1624,  1655,  1719,  1735,  1756,  1763,
       1764,  1771,  1799,  1801,  1802,  1805,  1807,  1833,  1915,
       1919,  1930,  1952,  1963,  2026,  2028,  2037,  2063,  2072,
       2093,  2104,  2112,  2160,  2195,  2212,  2242,  2254,  2266,
       2282,  2290,  2295,  2313,  2322,  2343,  2344,  2405,  2431,
       2441,  2466,  2468,  2510,  2547,  2551,  2565,  2575,  2585,
       2591,  2664,  2700,  2711,  2716,  2724,  2736,  2748,  2753,
       2844,  2876,  2881,  2907,  2934,  3006,  3017,  3062,  3067,
       3101,  3233,  3238,  3253,  3268,  3275,  3287,  3310,  3321,
       3326,  3376,  3410,  3417,  3457,  3464,  3503,  3528,  3545,
       3577,  3609,  3639,  3648,  3685,  3704,  3722,  3723,  3755,
       3772,  3774,  3785,  3798,  3828,  3846,  3889,  3891,  3897,
       3908,  3912,  3914,  3949,  3982,  4013,  4026,  4033,  4080,
       4099,  4157,  4162,  4172,  4189,  4212,  4228,  4234,  4261,
       4273,  4291,  4304,  4323,  4332,  4335,  4354,  4366,  4372,
       4381,  4400,  4461,  4466,  4487,  4603,  4605,  4649,  4716,
       4755,  4759,  4766,  4770,  4779,  4786,  4884,  4893,  4907,
       4939,  4955,  4961,  4994,  5001,  5020,  5028,  5040,  5078,
       5091,  5106,  5147,  5154,  5170,  5179,  5188,  5240,  5256,
       5317,  5322,  5344,  5353,  5378,  5387,  5453,  5454,  5478,
       5482,  5486,  5522,  5554,  5559,  5595,  5623,  5626,  5656,
       5674,  5687,  5709,  5718,  5754,  5805,  5816,  5834,  5836,
       5883,  5900,  5943,  5945,  5956,  5965,  5969,  6042,  6055,
       6082,  6093,  6134,  6195,  6208,  6210,  6223,  6253,  6270,
       6323,  6329,  6335,  6359,  6426,  6466,  6473,  6519,  6530,
       6536,  6556,  6564,  6575,  6605,  6659,  6675,  6711,  6730,
       6765,  6782,  6807,  6823,  6849,  6869,  6919,  6927,  6933,
       6950,  6963,  6966,  6991,  7021,  7041,  7106,  7125,  7158,
       7174,  7176,  7177,  7203,  7256,  7271,  7302,  7323,  7335,
       7348,  7353,  7355,  7369,  7382,  7388,  7393,  7457,  7464,
       7497,  7515,  7516,  7537,  7567,  7580,  7601,  7657,  7666,
       7683,  7720,  7750,  7790,  7795,  7834,  7835,  7843,  7848,
       7858,  7874,  7881,  7905,  7928,  7933,  7964,  7978,  8013,
       8035,  8065,  8072,  8083,  8119,  8170,  8172,  8183,  8228,
       8296,  8308,  8319,  8348,  8364,  8393,  8430,  8434,  8437,
       8440,  8456,  8487,  8496,  8505,  8574,  8584,  8598,  8607,
       8610,  8667,  8679,  8693,  8718,  8738,  8761,  8771,  8776,
       8779,  8799,  8803,  8867,  8870,  8871,  8907,  8935,  8947,
       8960,  8962,  8992,  9005,  9039,  9055,  9065,  9138,  9153,
       9155,  9172,  9177,  9190,  9194,  9198,  9247,  9265,  9285,
       9320,  9324,  9351,  9371,  9410,  9421,  9424,  9425,  9435,
       9442,  9450,  9453,  9472,  9479,  9509,  9513,  9540,  9554,
```

```
 9568,  9591,  9627,  9632,  9641,  9665,  9720,  9753,  9811,
 9869,  9871,  9930,  9936,  9937, 10063, 10089, 10091, 10120,
10135, 10226, 10248, 10258, 10265, 10268, 10290, 10298, 10306,
10378, 10449, 10466, 10507, 10526, 10550, 10559, 10596, 10604,
10620, 10635, 10641, 10662, 10666, 10710, 10726, 10747, 10757,
10758, 10801, 10825, 10840, 10852, 10892, 10918, 10938, 10966,
11003, 11012, 11111, 11131, 11143, 11189, 11190, 11204, 11216,
11232, 11235, 11280, 11295, 11302, 11356, 11398, 11418, 11422,
11439, 11441, 11460, 11466, 11490, 11492, 11510, 11533, 11581,
11599, 11611, 11615, 11635, 11708, 11716, 11743, 11751, 11759,
11782, 11784, 11816, 11863, 11900, 11953, 12026, 12038, 12048,
12066, 12075, 12087, 12095, 12096, 12151, 12190, 12204, 12238,
12303, 12314, 12331, 12334, 12338, 12403, 12412, 12437, 12462,
12488, 12528, 12558, 12586, 12591, 12627, 12667, 12686, 12739,
12742, 12775, 12783, 12831, 12857, 12859, 12889, 12948, 12953,
12963, 13037, 13076, 13098, 13121, 13139, 13149, 13177, 13244,
13258, 13295, 13301, 13379, 13396, 13416, 13417, 13428, 13439,
13442, 13471, 13535, 13560, 13563, 13571, 13589, 13607, 13670,
13713, 13748, 13783, 13826, 13870, 13874, 13914, 13920, 13926,
13928, 13932, 13935, 13975, 14068, 14075, 14077, 14105, 14107,
14140, 14167, 14177, 14192, 14197, 14212, 14246, 14249, 14251,
14262, 14268, 14314, 14321, 14329, 14351, 14391, 14406, 14418,
14431, 14433, 14491, 14500, 14522, 14542, 14582, 14588, 14607,
14635, 14653, 14755, 14758, 14766, 14802, 14815, 14818, 14848,
14858, 14873, 14924, 14936, 14941, 14972, 14986, 15004, 15005,
15033, 15078, 15111, 15112, 15118, 15133, 15152, 15178, 15212,
15235, 15281, 15287, 15330, 15416, 15446, 15468, 15477, 15479,
15544, 15555, 15570, 15607, 15610, 15616, 15671, 15676, 15752,
15755, 15779, 15790, 15821, 15846, 15851, 15892, 15898, 15916,
15920, 15928, 15934, 15970, 15988, 16037, 16039, 16062, 16074,
16129, 16152, 16173, 16227, 16244, 16255, 16257, 16278, 16295,
16309, 16356, 16360, 16375, 16389, 16431, 16441, 16486, 16497,
16500, 16537, 16560, 16576, 16608, 16615, 16640, 16689, 16720,
16760, 16795, 16825, 16857, 16860, 16886, 16953, 16966, 17000,
17032, 17034, 17036])
```

이번에는 data_purchase 변수에는 구매 기록이 있는 session들의 데이터를,

data_npurchase 변수에는 구매 기록이 없는 session들의 데이터를 저장해주세요.

`Hint) inin()메서드를 이용한 indexing`

In [112…

```
data_purchase = log_order.loc[log_order['session_idx_unique'].isin(session_purchase)]
data_npurchase = log_order.loc[~log_order['session_idx_unique'].isin(session_purchase)]
```

이제 data_purchase를 이용하여 구매 session의 체류시간을 구해주세요.

`Hint) User별 체류시간을 구하는 방법과 매우 유사합니다. session_idx_unique를 이용해주세요.`

In [113…

```
purchase_session_duration = (
                    data_purchase
                    .groupby(['n_user_id', 'session_idx_unique'])['page_duration']
                    .sum()
                    .reset_index()
                    .rename(columns = {"page_duration" : 'session_duration'})
                )

purchase_session_duration.head(10)
```

Out[113…

	n_user_id	session_idx_unique	session_duration
0	2	4	3075.422
1	8	15	412.025
2	8	16	1791.231
3	9	19	1657.393
4	47	88	3116.367
5	49	92	2615.611
6	65	123	3093.858
7	86	168	1906.721
8	97	192	3031.770
9	117	232	1191.110

마찬가지로 비구매 session의 체류시간을 구해주세요.

In [114…
```
npurchase_session_duration = (
                    data_npurchase
                    .groupby(['n_user_id', 'session_idx_unique'])['page_duration']
                    .sum()
                    .reset_index()
                    .rename(columns = {"page_duration" : 'session_duration'})
                )

npurchase_session_duration.head(10)
```

Out[114…

	n_user_id	session_idx_unique	session_duration
0	0	0	114.890
1	1	1	1011.541
2	2	2	893.742
3	2	3	69.618
4	3	5	0.000
5	3	6	45.911
6	3	7	85.183
7	3	8	1711.031
8	4	9	49.130
9	5	10	0.000

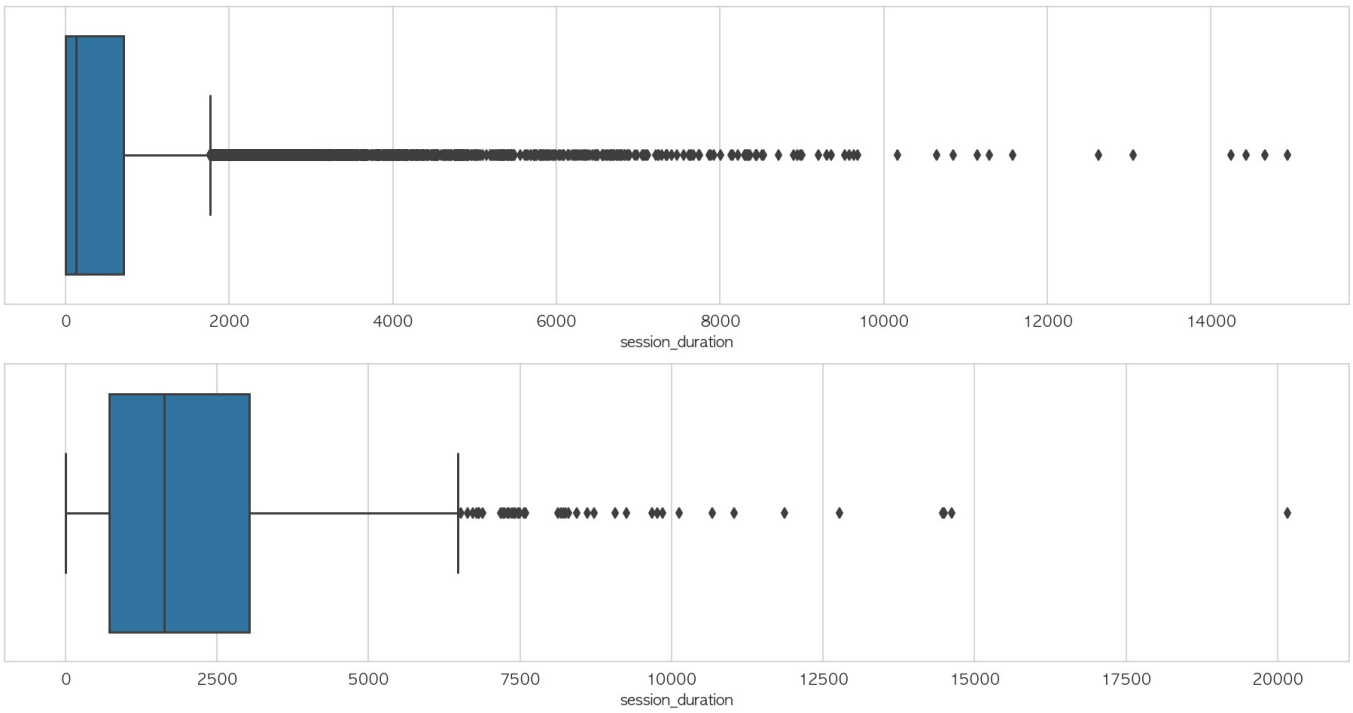
마지막으로 구매/비구매 cycle의 잔존 시간으로 boxplot을 그려주세요. 그리고 구매 cycle 잔존 시간 평균과 비구매 cycle 잔존 시간 평균을 구해주세요.

결과는 아래와 같습니다.

```
구매 session 잔존 시간 평균: 2280.0800799999997
비구매 session 잔존 시간 평균: 611.9902027010454
```

In [124…
```
figure, (ax1, ax2) = plt.subplots(nrows = 2, ncols = 1)

figure.set_size_inches(16,8)

sns.boxplot(data = npurchase_session_duration, x = "session_duration", ax = ax1)
sns.boxplot(data = purchase_session_duration, x = "session_duration", ax = ax2)
```

Out[124…
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a27898b00>
```



In [115…

```
purchase_session_mean = purchase_session_duration['session_duration'].mean()
npurchase_session_mean = npurchase_session_duration['session_duration'].mean()

print("구매 session 잔존 시간 평균:",purchase_session_mean)
print("비구매 session 잔존 시간 평균:",npurchase_session_mean)
```

```
구매 session 잔존 시간 평균: 2280.0800799999997
비구매 session 잔존 시간 평균: 611.9902027010454
```

제출

과제를 다 끝내셨으면 http://bit.ly/ds-assignment 에서 안내에 따라 과제를 제출하여 주세요! 과제를 제출해주시면 솔루션과 검토 결과를 드립니다. 오프라인 수업의 경우 과제를 제출하지 않으시더라도 솔루션은 다음 수업 시간에 제공해드립니다.

수업이나 과제 관련 질문, 수료증 문의 등은 담당 튜터(조교)에게 문의 주세요. 영수증 발급 등의 문의는 support@dsschool.co.kr 로 메일 주시면 담당자분이 응대해주실 겁니다. 기타 궁금한 사항은 슬랙으로 문의 주세요!

In [ ]: