남지원

2021.04.22

In [1]:

```
import pandas as pd
```

out.csv 읽기

In [3]:

```
out_data = pd.read_csv("E:/BigData/8ju/out.csv")
out_data.head()
```

Out[3]:

	purchase_date	item_name	item_price	customer_name
0	2019-06-13 18:02	상품A	100.0	김가온
1	2019-07-13 13:05	상 품 S	NaN	김우찬
2	2019-05-11 19:42	상 품 a	NaN	김유찬
3	2019-02-12 23:40	상품Z	2600.0	김재현
4	2019-04-22 3:09	상품a	NaN	김강현

cust.xlsx 읽기

In [7]:

```
cust_data = pd.read_excel("E:/BigData/8ju/cust.xlsx")
cust_data.head()
```

Out[7]:

	고객이름	지역	등록일
0	김 현성	H시	2018-01-04 00:00:00
1	김 도윤	E시	42782
2	김 지한	A시	2018-01-07 00:00:00
3	김 하윤	F시	42872
4	김 시온	E시	43127

In [8]:

```
cust_data.tail()
```

Out[8]:

	고객이름	지역	등록일
195	김 재희	G시	2017-06-20 00:00:00
196	김 도영	E시	2018-06-20 00:00:00
197	김 이안	F시	2017-04-29 00:00:00
198	김 시현	H시	2019-04-19 00:00:00
199	김 서우	D시	2019-04-23 00:00:00

In [9]:

```
out_data.tail()
```

Out[9]:

	purchase_date	item_name	item_price	customer_name
2994	2019-02-15 2:56	상품Y	2500.0	김정민
2995	2019-06-22 4:03	상품M	1300.0	김재원
2996	2019-03-29 11:14	상품Q	NaN	김지율
2997	2019-07-14 12:56	상품H	NaN	김승주
2998	2019-07-21 0:31	상품D	400.0	정준기

전처리 과정의 필요성을 보여준다

In [11]:

```
out_data["purchase_date"] = pd.to_datetime(out_data["purchase_date"])
out_data["purchase_month"] = out_data["purchase_date"].dt.strftime("%Y%m")
res = out_data.pivot_table(index="purchase_month", columns="item_name", aggfunc="size", fill_val
ue=0)
res
```

Out[11]:

item_name	상품n	상품E	상품M	상품P	상품S	상품W	상품X	상품W	상품O	상품Q	...	상품k	상품l	상품o	상품p	상품r	상품s	상품t	상품v	상품x
purchase_month																				
201901	1	0	0	0	0	0	0	0	0	0	...	1	1	1	0	0	0	0	0	0
201902	0	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	1	1	1	0
201903	0	1	1	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0
201904	0	0	0	0	0	0	0	1	0	1	...	0	0	0	0	0	1	0	0	0
201905	0	0	0	0	1	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0
201906	0	0	0	0	0	1	0	0	0	0	...	0	0	0	1	0	0	0	0	1
201907	0	0	0	0	0	0	0	0	1	0	...	0	0	1	0	2	0	0	0	0

7 rows × 99 columns

In [12]:

```
res = out_data.pivot_table(index="purchase_month",
                           columns="item_name",
                           values="item_price",
                           aggfunc="sum",
                           fill_value=0)
res
```

Out[12]:

item_name	상품n	상품E	상품M	상품P	상품S	상품W	상품X	상품W	상품O	상품Q	...	상품k	상품l
purchase_month													
201901	1400	0	0	0	0	0	0	0	0	0	...	1100	1200
201902	0	0	0	0	0	0	2400	0	0	0	...	0	0
201903	0	500	1300	1600	0	0	0	0	0	0	...	0	0
201904	0	0	0	0	0	0	0	2300	0	1700	...	0	0
201905	0	0	0	0	1900	0	0	0	0	0	...	0	1200
201906	0	0	0	0	0	2300	0	0	0	0	...	0	0
201907	0	0	0	0	0	0	0	0	0	0	...	0	0

7 rows × 99 columns

오류 수정

In [14]:

```
# 상품명 개수 확인

print(len(pd.unique(out_data["item_name"])))
```

99

In [18]:

```
# 수정

out_data["item_name"] = out_data["item_name"].str.upper() # 대문자로 수정
out_data["item_name"] = out_data["item_name"].str.replace("  ", "") # 공백을 삭제
out_data["item_name"] = out_data["item_name"].str.replace(" ", "") # 공백을 삭제
out_data.sort_values(by=["item_name"], ascending=True) # 정렬(item_name 기준으로 오름차순)

print(len(pd.unique(out_data["item_name"])))
```

26

In [19]:

```
# 금액의 결측치 수정
# isnull : 결측치 확인
# any : 하나라도 결측이라면 true반환

out_data.isnull().any()
```

Out[19]:

```
purchase_date     False
item_name         False
item_price         True
customer_name     False
purchase_month    False
dtype: bool
```

In [23]:

```
# 결측치가 있는 상품명 리스트 작성 : fig_is_null 이용

flg_is_null = out_data["item_price"].isnull()
flg_is_null
```

Out[23]:

```
0       False
1        True
2        True
3       False
4        True
        ...
2994    False
2995    False
2996     True
2997     True
2998    False
Name: item_price, Length: 2999, dtype: bool
```

In [24]:

```
# loc를 사용해 조건에 일치하는 데이터 추출 => 중복 제거 => 리스트로 생성

out_data.loc[flg_is_null, "item_name"]
```

Out[24]:

```
1       상품S
2       상품A
4       상품A
6       상품A
14      상품A
       ...
2987    상품K
2990    상품O
2992    상품C
2996    상품Q
2997    상품H
Name: item_name, Length: 387, dtype: object
```

In [26]:

```
# z가 결측치가 없음

pd.unique(out_data.loc[flg_is_null, "item_name"])
```

Out[26]:

```
array(['상품S', '상품A', '상품P', '상품N', '상품W', '상품R', '상품I', '상품L', '상
품F',
       '상품O', '상품B', '상품C', '상품V', '상품Q', '상품U', '상품K', '상품T', '상
품X',
       '상품E', '상품M', '상품G', '상품J', '상품D', '상품H', '상품Y'], dtype=objec
t)
```

In [29]:

```
for trg in list(out_data.loc[flg_is_null, "item_name"].unique()):
    # 금액을 가져옴
    price = out_data.loc[(~flg_is_null) & (out_data["item_name"] == trg), "item_price"].max()
    # 해당 금액을 넣어줌
    out_data["item_price"].loc[(flg_is_null) & (out_data["item_name"]==trg)]

out_data.head()
```

Out[29]:

	purchase_date	item_name	item_price	customer_name	purchase_month
0	2019-06-13 18:02:00	상품A	100.0	김가온	201906
1	2019-07-13 13:05:00	상품S	1900.0	김우찬	201907
2	2019-05-11 19:42:00	상품A	100.0	김유찬	201905
3	2019-02-12 23:40:00	상품Z	2600.0	김재현	201902
4	2019-04-22 03:09:00	상품A	100.0	김강현	201904

In [30]:

```
# 결측 확인

out_data.isnull().any()
```

Out[30]:

```
purchase_date     False
item_name         False
item_price        False
customer_name     False
purchase_month    False
dtype: bool
```

In [31]:

```
# 각 상품의 금액이 정상적으로 수정되었는지 확인

for trg in list(out_data["item_name"].sort_values().unique()):
    print(trg + "의최고가 : " + str(out_data.loc[out_data["item_name"]==trg]["item_price"].max())
          + "의최저가 : " + str(out_data.loc[out_data["item_name"]==trg]["item_price"].min(skipna
=False)))
```

```
상품A의최고가 : 100.0의최저가 : 100.0
상품B의최고가 : 200.0의최저가 : 200.0
상품C의최고가 : 300.0의최저가 : 300.0
상품D의최고가 : 400.0의최저가 : 400.0
상품E의최고가 : 500.0의최저가 : 500.0
상품F의최고가 : 600.0의최저가 : 600.0
상품G의최고가 : 700.0의최저가 : 700.0
상품H의최고가 : 800.0의최저가 : 800.0
상품I의최고가 : 900.0의최저가 : 900.0
상품J의최고가 : 1000.0의최저가 : 1000.0
상품K의최고가 : 1100.0의최저가 : 1100.0
상품L의최고가 : 1200.0의최저가 : 1200.0
상품M의최고가 : 1300.0의최저가 : 1300.0
상품N의최고가 : 1400.0의최저가 : 1400.0
상품O의최고가 : 1500.0의최저가 : 1500.0
상품P의최고가 : 1600.0의최저가 : 1600.0
상품Q의최고가 : 1700.0의최저가 : 1700.0
상품R의최고가 : 1800.0의최저가 : 1800.0
상품S의최고가 : 1900.0의최저가 : 1900.0
상품T의최고가 : 2000.0의최저가 : 2000.0
상품U의최고가 : 2100.0의최저가 : 2100.0
상품V의최고가 : 2200.0의최저가 : 2200.0
상품W의최고가 : 2300.0의최저가 : 2300.0
상품X의최고가 : 2400.0의최저가 : 2400.0
상품Y의최고가 : 2500.0의최저가 : 2500.0
상품Z의최고가 : 2600.0의최저가 : 2600.0
```

고객이름 통일

In [32]:

```
cust_data["고객이름"].head()
```

Out[32]:

```
0    김 현성
1    김 도윤
2    김 지한
3    김 하윤
4    김 시온
Name: 고객이름, dtype: object
```

In [33]:

```
out_data["customer_name"].head()
```

Out[33]:

```
0    김가온
1    김우찬
2    김유찬
3    김재현
4    김강현
Name: customer_name, dtype: object
```

In [34]:

```
cust_data["고객이름"] = cust_data["고객이름"].str.replace("  ", "")
cust_data["고객이름"] = cust_data["고객이름"].str.replace(" ", "")
cust_data["고객이름"].head()
```

Out[34]:

```
0    김현성
1    김도윤
2    김지한
3    김하윤
4    김시온
Name: 고객이름, dtype: object
```

In [35]:

```
flg_is_serial = cust_data["등록일"].astype("str").str.isdigit()
flg_is_serial.sum()
```

Out[35]:

```
22
```

In [36]:

```
# timedelta : 숫자를 날짜로 변경하는 함수
# 엑셀과 2일 차이가 난다.
# 그래서 -2하는거임

fromSerial = pd.to_timedelta(cust_data.loc[flg_is_serial, "등록일"].astype("float")-2, unit="D")
+ pd.to_datetime("1900/01/01")
fromSerial
```

Out[36]:

```
1      2017-02-16
3      2017-05-17
4      2018-01-27
21     2017-07-04
27     2017-06-15
47     2017-01-06
49     2017-07-13
53     2017-04-08
76     2018-03-29
80     2018-01-10
99     2017-05-30
114    2018-06-03
118    2018-01-29
122    2018-04-16
139    2017-05-25
143    2017-03-24
155    2017-01-19
172    2018-03-22
179    2017-01-08
183    2017-07-24
186    2018-07-13
192    2018-06-08
Name: 등록일, dtype: datetime64[ns]
```

In [37]:

```
# string을 날짜형으로 변경

fromString = pd.to_datetime(cust_data.loc[~flg_is_serial, "등록일"])
fromString
```

Out[37]:

```
0      2018-01-04
2      2018-01-07
5      2017-06-20
6      2018-06-11
7      2017-05-19
          ...
195    2017-06-20
196    2018-06-20
197    2017-04-29
198    2019-04-19
199    2019-04-23
Name: 등록일, Length: 178, dtype: datetime64[ns]
```

In [38]:

```
# 등록일을 세로로 합치기

cust_data["등록일"] = pd.concat([fromSerial, fromString])
cust_data
```

Out[38]:

	고객이름	지역	등록일
0	김현성	H시	2018-01-04
1	김도윤	E시	2017-02-16
2	김지한	A시	2018-01-07
3	김하윤	F시	2017-05-17
4	김시온	E시	2018-01-27
...	...	...	...
195	김재희	G시	2017-06-20
196	김도영	E시	2018-06-20
197	김이안	F시	2017-04-29
198	김시현	H시	2019-04-19
199	김서우	D시	2019-04-23

200 rows × 3 columns

In [39]:

```
cust_data["등록연월"] = cust_data["등록일"].dt.strftime("%Y%m")
rslt = cust_data.groupby("등록연월").count()["고객이름"]
print(rslt)
print(len(cust_data))
```

```
등록연월
201701    15
201702    11
201703    14
201704    15
201705    14
201706    13
201707    17
201801    13
201802    15
201803    17
201804     5
201805    19
201806    13
201807    17
201904     2
Name: 고객이름, dtype: int64
200
```

In [40]:

```
# 숫자 데이터가 아직 남아있는지 확인

flg_is_serial = cust_data["등록일"].astype("str").str.isdigit()
flg_is_serial.sum()
```

Out[40]:

0

피봇테이블로도 만들 수 있다.

In [89]:

```
pdf1 = pd.pivot_table(cust_data,                # 피벗할 데이터프레임
                      index = '등록연월',    # 행 위치에 들어갈 열
                      values = '등록일',     # 데이터로 사용할 열
                      aggfunc = 'count')   # 데이터 집계함수
pdf1
```

Out[89]:

등록연월	등록일
201701	15
201702	11
201703	14
201704	15
201705	14
201706	13
201707	17
201801	13
201802	15
201803	17
201804	5
201805	19
201806	13
201807	17
201904	2

데이터 가로 병합

In [51]:

```
join_data = pd.merge(out_data,
                     cust_data,
                     left_on="customer_name",
                     right_on="고객이름",
                     how="left")
join_data
```

Out[51]:

	purchase_date	item_name	item_price	customer_name	purchase_month	고객이름	지역	등록일
0	2019-06-13 18:02:00	상품A	100.0	김가온	201906	김가온	C시	2017-01-26
1	2019-07-13 13:05:00	상품S	1900.0	김우찬	201907	김우찬	C시	2018-04-07
2	2019-05-11 19:42:00	상품A	100.0	김유찬	201905	김유찬	A시	2018-06-19
3	2019-02-12 23:40:00	상품Z	2600.0	김재현	201902	김재현	D시	2018-07-22
4	2019-04-22 03:09:00	상품A	100.0	김강현	201904	김강현	D시	2017-06-07
...	...	...	...	...	...	...	...	...
2994	2019-02-15 02:56:00	상품Y	2500.0	김정민	201902	김정민	B시	2017-07-01
2995	2019-06-22 04:03:00	상품M	1300.0	김재원	201906	김재원	E시	2018-03-31
2996	2019-03-29 11:14:00	상품Q	1700.0	김지율	201903	김지율	B시	2017-03-15
2997	2019-07-14 12:56:00	상품H	800.0	김승주	201907	김승주	E시	2018-07-13
2998	2019-07-21 00:31:00	상품D	400.0	정준기	201907	정준기	B시	2017-02-05

2999 rows × 9 columns

In [52]:

```
# 이름이 두번 나오기 때문에 customer_name 을 삭제

join_data = join_data.drop("customer_name", axis=1)
join_data
```

Out[52]:

	purchase_date	item_name	item_price	purchase_month	고객이름	지역	등록일	등록연월
0	2019-06-13 18:02:00	상품A	100.0	201906	김가온	C시	2017-01-26	201701
1	2019-07-13 13:05:00	상품S	1900.0	201907	김우찬	C시	2018-04-07	201804
2	2019-05-11 19:42:00	상품A	100.0	201905	김유찬	A시	2018-06-19	201806
3	2019-02-12 23:40:00	상품Z	2600.0	201902	김재현	D시	2018-07-22	201807
4	2019-04-22 03:09:00	상품A	100.0	201904	김강현	D시	2017-06-07	201706
...	...	...	...	...	...	...	...	...
2994	2019-02-15 02:56:00	상품Y	2500.0	201902	김정민	B시	2017-07-01	201707
2995	2019-06-22 04:03:00	상품M	1300.0	201906	김재원	E시	2018-03-31	201803
2996	2019-03-29 11:14:00	상품Q	1700.0	201903	김지율	B시	2017-03-15	201703
2997	2019-07-14 12:56:00	상품H	800.0	201907	김승주	E시	2018-07-13	201807
2998	2019-07-21 00:31:00	상품D	400.0	201907	정준기	B시	2017-02-05	201702

2999 rows × 8 columns

파일로 덤프

In [53]:

```
# 컬럼 순서 변경해서 join

dump_data = join_data[["purchase_date", "purchase_month", "item_name", "item_price", "고객이름", "지역", "등록일"]]
dump_data
```

Out[53]:

	purchase_date	purchase_month	item_name	item_price	고객이름	지역	등록일
0	2019-06-13 18:02:00	201906	상품A	100.0	김가온	C시	2017-01-26
1	2019-07-13 13:05:00	201907	상품S	1900.0	김우찬	C시	2018-04-07
2	2019-05-11 19:42:00	201905	상품A	100.0	김유찬	A시	2018-06-19
3	2019-02-12 23:40:00	201902	상품Z	2600.0	김재현	D시	2018-07-22
4	2019-04-22 03:09:00	201904	상품A	100.0	김강현	D시	2017-06-07
...	...	...	...	...	...	...	...
2994	2019-02-15 02:56:00	201902	상품Y	2500.0	김정민	B시	2017-07-01
2995	2019-06-22 04:03:00	201906	상품M	1300.0	김재원	E시	2018-03-31
2996	2019-03-29 11:14:00	201903	상품Q	1700.0	김지율	B시	2017-03-15
2997	2019-07-14 12:56:00	201907	상품H	800.0	김승주	E시	2018-07-13
2998	2019-07-21 00:31:00	201907	상품D	400.0	정준기	B시	2017-02-05

2999 rows × 7 columns

In [61]:

```
dump_data.to_csv("E:/bigdata/8ju/dump_data.csv", index=False)
```

실기 테스트

In [62]:

```
menu_list = pd.read_csv('E:/bigdata/8ju/menus.csv', encoding='cp949')
```

In [63]:

```
meal_list = pd.read_excel('E:/bigdata/8ju/meals.xlsx', header=0)
```

In [68]:

```
# inner_join

# meal_list와 menu_list를 left에는 menuNo, right에는 menuID, 로 inner join
inner_join = pd.merge(meal_list,menu_list,left_on = 'menuNo',right_on = 'menuID',how = 'inner')

inner_join
```

Out[68]:

	menuNo	selDate	amount	menuID	menuName
0	1	2011-05-01	4000	1	백반
1	1	2011-05-03	4000	1	백반
2	4	2011-05-01	3800	4	짬뽕
3	4	2011-05-05	3800	4	짬뽕
4	2	2011-05-02	4000	2	김치찌개

In [78]:

```
re_data = inner_join[["menuNo", "menuID", "selDate", "amount"," menuName"]]
re_data
```

Out[78]:

	menuNo	menuID	selDate	amount	menuName
0	1	1	2011-05-01	4000	백반
1	1	1	2011-05-03	4000	백반
2	4	4	2011-05-01	3800	짬뽕
3	4	4	2011-05-05	3800	짬뽕
4	2	2	2011-05-02	4000	김치찌개

In [80]:

```
outer_join = pd.merge(meal_list,menu_list,left_on = 'menuNo',right_on = 'menuID',how = 'outer')
outer_join
```

Out[80]:

	menuNo	selDate	amount	menuID	menuName
0	1.0	2011-05-01	4000.0	1.0	백반
1	1.0	2011-05-03	4000.0	1.0	백반
2	4.0	2011-05-01	3800.0	4.0	짬뽕
3	4.0	2011-05-05	3800.0	4.0	짬뽕
4	2.0	2011-05-02	4000.0	2.0	김치찌개
5	9.0	2011-05-05	2500.0	NaN	NaN
6	NaN	NaT	NaN	3.0	순두부

In [86]:

```
outer_join = pd.merge(meal_list,menu_list,left_on = 'menuNo',right_on = 'menuID',how = 'right')
outer_join
```

Out[86]:

	menuNo	selDate	amount	menuID	menuName
0	1.0	2011-05-01	4000.0	1	백반
1	1.0	2011-05-03	4000.0	1	백반
2	4.0	2011-05-01	3800.0	4	짬뽕
3	4.0	2011-05-05	3800.0	4	짬뽕
4	2.0	2011-05-02	4000.0	2	김치찌개
5	NaN	NaT	NaN	3	순두부

In [ ]:

```

```