データの加工

In [25]:
```python
import pandas as pd
```

売上および顧客データの読み込み

In [26]:
```python
uriage_data = pd.read_csv('./data/data2/uriage.csv')
uriage_data.head()
```

Out[26]:

	purchase_date	item_name	item_price	customer_name
0	2019-06-13 18:02:34	商品A	100.0	深井菜々美
1	2019-07-13 13:05:29	商 品 S	NaN	浅田賢二
2	2019-05-11 19:42:07	商 品 a	NaN	南部慶二
3	2019-02-12 23:40:45	商品Z	2600.0	麻生莉緒
4	2019-04-22 03:09:35	商品a	NaN	平田鉄二

In [27]:
```python
kokyaku_data = pd.read_excel('./data/data2/kokyaku_daicho.xlsx')
kokyaku_data.head()
```

Out[27]:

	顧客名	かな	地域	メールアドレス	登録日
0	須賀ひとみ	すが ひとみ	H市	suga_hitomi@example.com	2018/01/04
1	岡田　敏也	おかだ としや	E市	okada_toshiya@example.com	42782
2	芳賀 希	はが のぞみ	A市	haga_nozomi@example.com	2018/01/07
3	荻野 愛	おぎの あい	F市	ogino_ai@example.com	42872
4	栗田 憲一	くりた けんいち	E市	kurita_kenichi@example.com	43127

In [28]:
```python
# 表記が統一されていないデータの確認
uriage_data[['item_name']].head()
```

Out[28]:

	item_name
0	商品A
1	商 品 S
2	商 品 a
3	商品Z
4	商品a

データの前処理を行う前にデータを確認

In [29]:
```python
uriage_data['purchase_date'] = pd.to_datetime(uriage_data['purchase_date'])
uriage_data['purchase_month'] = uriage_data['purchase_date'].dt.strftime('%Y%m')
result = uriage_data.pivot_table(index = 'purchase_month', columns = 'item_name', aggfunc = 'size', fill_value = 0)
result
```

Out[29]:

item_name	商品W	商 品 n	商品E	商品M	商品P	商品S	商品W	商品X	商 品O	商 品Q	...	商品k	商品l	商品o	商品p	商品r	商品s	商品t	商品v	商品x	商品y
purchase_month																					
201901	0	1	0	0	0	0	0	0	0	0	...	1	1	1	0	0	0	0	0	0	0
201902	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	1	1	1	0	0
201903	0	0	1	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
201904	1	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	1	0	0	0	0
201905	0	0	0	0	0	1	0	0	0	0	...	0	1	0	0	0	0	0	0	0	1
201906	0	0	0	0	0	0	1	0	0	0	...	0	0	0	1	0	0	0	0	1	0
201907	0	0	0	0	0	0	0	0	1	0	...	0	0	1	0	2	0	0	0	0	0

7 rows × 99 columns

商品名の統一処理

In [30]:
```python
len(pd.unique(uriage_data['item_name'])) # 処理前の商品名のユニークデータ数
```

Out[30]:
```
99
```

In [31]:
```python
uriage_data['item_name'] = uriage_data['item_name'].str.upper() # 小文字を大文字に変換
uriage_data['item_name'] = uriage_data['item_name'].str.replace('　', '') # 全角スペースを削除
uriage_data['item_name'] = uriage_data['item_name'].str.replace(' ', '') # 半角スペースを削除
uriage_data.sort_values(by = ['item_name'], ascending = True) # item_nameで昇順にソート
```

Out[31]:

	purchase_date	item_name	item_price	customer_name	purchase_month
0	2019-06-13 18:02:34	商品A	100.0	深井菜々美	201906
1748	2019-05-19 20:22:22	商品A	100.0	松川綾女	201905
223	2019-06-25 08:13:20	商品A	100.0	板橋隆	201906
1742	2019-06-13 16:03:17	商品A	100.0	小平陽子	201906
1738	2019-02-10 00:28:43	商品A	100.0	松田浩正	201902
...	...	...	...	...	...
2880	2019-04-22 00:36:52	商品Y	NaN	田辺光洋	201904
2881	2019-04-30 14:21:09	商品Y	NaN	高原充則	201904
1525	2019-01-24 10:27:23	商品Y	2500.0	五十嵐春樹	201901
1361	2019-05-28 13:45:32	商品Y	2500.0	大崎ヒカル	201905
3	2019-02-12 23:40:45	商品Z	2600.0	麻生莉緒	201902

2999 rows × 5 columns

In [32]:
```python
# 処理の確認
print(len(pd.unique(uriage_data['item_name']))) # 商品の種類の数
print(pd.unique(uriage_data['item_name'])) # 商品全体のリストを表示
```

```
26
['商品A' '商品S' '商品Z' '商品V' '商品O' '商品U' '商品L' '商品C' '商品I' '商品R' '商品X' '商品G'
 '商品P' '商品Q' '商品Y' '商品N' '商品W' '商品E' '商品K' '商品B' '商品F' '商品D' '商品M' '商品H'
 '商品T' '商品J']
```

欠損値の処理

In [33]:
```python
uriage_data.isnull().any() # いずれかの列で欠損値を含むか否か
```

Out[33]:
```
purchase_date     False
item_name         False
item_price         True
customer_name     False
purchase_month    False
dtype: bool
```

In [34]:
```python
uriage_data['item_price'].isnull().sum() # 欠損値のあった列において欠損値は何個存在するか
```

Out[34]:
```
387
```

In [35]:
```python
# 欠損値補完
flg_is_null = uriage_data['item_price'].isnull()
for trg in list(uriage_data.loc[flg_is_null, 'item_name'].unique()):
    price = uriage_data.loc[(~flg_is_null) & (uriage_data['item_name'] == trg), 'item_price'].max()
    uriage_data['item_price'].loc[(flg_is_null) & (uriage_data['item_name'] == trg)] = price
uriage_data.head()
```

```
/Users/rtmakbook2018/opt/anaconda3/lib/python3.8/site-packages/pandas/core/indexing.py:1637: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

Out[35]:

	purchase_date	item_name	item_price	customer_name	purchase_month
0	2019-06-13 18:02:34	商品A	100.0	深井菜々美	201906
1	2019-07-13 13:05:29	商品S	1900.0	浅田賢二	201907
2	2019-05-11 19:42:07	商品A	100.0	南部慶二	201905
3	2019-02-12 23:40:45	商品Z	2600.0	麻生莉緒	201902
4	2019-04-22 03:09:35	商品A	100.0	平田鉄二	201904

In [36]:
```python
uriage_data['item_price'].isnull().sum() # 欠損値補完の確認
```

Out[36]:
```
0
```

顧客名の記載方法を統一

In [37]:
```python
kokyaku_data.head()
```

Out[37]:

	顧客名	かな	地域	メールアドレス	登録日
0	須賀ひとみ	すが ひとみ	H市	suga_hitomi@example.com	2018/01/04
1	岡田　敏也	おかだ としや	E市	okada_toshiya@example.com	42782
2	芳賀 希	はが のぞみ	A市	haga_nozomi@example.com	2018/01/07
3	荻野 愛	おぎの あい	F市	ogino_ai@example.com	42872
4	栗田 憲一	くりた けんいち	E市	kurita_kenichi@example.com	43127

In [38]:
```python
# 全角・半角スペースの削除処理
kokyaku_data['顧客名'] = kokyaku_data['顧客名'].str.replace('　', '')
kokyaku_data['顧客名'] = kokyaku_data['顧客名'].str.replace(' ', '')
kokyaku_data.head()
```

Out[38]:

	顧客名	かな	地域	メールアドレス	登録日
0	須賀ひとみ	すが ひとみ	H市	suga_hitomi@example.com	2018/01/04
1	岡田敏也	おかだ としや	E市	okada_toshiya@example.com	42782
2	芳賀希	はが のぞみ	A市	haga_nozomi@example.com	2018/01/07
3	荻野愛	おぎの あい	F市	ogino_ai@example.com	42872
4	栗田憲一	くりた けんいち	E市	kurita_kenichi@example.com	43127

登録日の修正

In [39]:
```python
# 登録日が日付でない（数値となっている）数の算出
flg_is_serial = kokyaku_data['登録日'].astype('str').str.isdigit()
flg_is_serial.sum()
```

Out[39]:
```
22
```

In [40]:
```python
fromSerial = pd.to_timedelta(kokyaku_data.loc[flg_is_serial, '登録日'].astype(float), unit = 'D') + pd.to_datetime('1900/01/01')  # loc関数はbool型で
fromSerial
print(fromSerial.count())
```

```
22
```

In [41]:
```python
# 登録日の数値データが正常に変換されたことを確認
flg_is_serial = kokyaku_data['登録日'].astype('str').str.isdigit()
flg_is_serial.sum()
```

Out[41]:
```
22
```

In [42]:
```python
fromString = pd.to_datetime(kokyaku_data.loc[~flg_is_serial, '登録日']) # Falseのデータをpandasのデータタイム型へ変換
fromString
print(fromString.count())
```

```
178
```

In [43]:
```python
# データの型と表記を統一した各登録日のデータをconcatで統合する
kokyaku_data['登録日'] = pd.concat([fromSerial, fromString])
kokyaku_data
```

Out[43]:

	顧客名	かな	地域	メールアドレス	登録日
0	須賀ひとみ	すが ひとみ	H市	suga_hitomi@example.com	2018-01-04
1	岡田敏也	おかだ としや	E市	okada_toshiya@example.com	2017-02-18
2	芳賀希	はが のぞみ	A市	haga_nozomi@example.com	2018-01-07
3	荻野愛	おぎの あい	F市	ogino_ai@example.com	2017-05-19
4	栗田憲一	くりた けんいち	E市	kurita_kenichi@example.com	2018-01-29
...	...	...	...	...	...
195	川上りえ	かわかみ りえ	G市	kawakami_rie@example.com	2017-06-20
196	小松季衣	こまつ としえ	E市	komatsu_toshie@example.com	2018-06-20
197	白鳥りえ	しらとり りえ	F市	shiratori_rie@example.com	2017-04-29
198	大西隆之介	おおにし りゅうのすけ	H市	oonishi_ryuunosuke@example.com	2019-04-19
199	福井美希	ふくい みき	D市	fukui_miki1@example.com	2019-04-23

200 rows × 5 columns

In [44]:
```python
kokyaku_data['登録年月'] = kokyaku_data['登録日'].dt.strftime('%Y%m')
rslt = kokyaku_data.groupby('登録年月').count()['顧客名']
rslt
```

Out[44]:
```
登録年月
201701    15
201702    11
201703    14
201704    15
201705    13
201706    14
201707    17
201801    13
201802    15
201803    17
201804     5
201805    19
201806    13
201807    17
201904     2
Name: 顧客名, dtype: int64
```

In [45]:
```python
len(kokyaku_data) # データの確認
```

Out[45]:
```
200
```

データの結合

In [46]:
```python
# 顧客名をキーにして結合
join_data = pd.merge(uriage_data, kokyaku_data, left_on = 'customer_name', right_on = '顧客名', how = 'left')
join_data = join_data.drop('customer_name', axis = 1) # 顧客名が重複しているので削除
join_data
```

Out[46]:

	purchase_date	item_name	item_price	purchase_month	顧客名	かな	地域	メールアドレス	登録日	登録年月
0	2019-06-13 18:02:34	商品A	100.0	201906	深井菜々美	ふかい ななみ	C市	fukai_nanami@example.com	2017-01-26	201701
1	2019-07-13 13:05:29	商品S	1900.0	201907	浅田賢二	あさだ けんじ	C市	asada_kenji@example.com	2018-04-07	201804
2	2019-05-11 19:42:07	商品A	100.0	201905	南部慶二	なんぶ けいじ	A市	nannbu_keiji@example.com	2018-06-19	201806
3	2019-02-12 23:40:45	商品Z	2600.0	201902	麻生莉緒	あそう りお	D市	asou_rio@example.com	2018-07-22	201807
4	2019-04-22 03:09:35	商品A	100.0	201904	平田鉄二	ひらた てつじ	D市	hirata_tetsuji@example.com	2017-06-07	201706
...	...	...	...	...	...	...	...	...	...	...
2994	2019-02-15 02:56:39	商品Y	2500.0	201902	福島友也	ふくしま ともや	B市	fukushima_tomoya@example.com	2017-07-01	201707
2995	2019-06-22 04:03:43	商品M	1300.0	201906	大倉晃司	おおくら こうじ	E市	ookura_kouji@example.com	2018-03-31	201803
2996	2019-03-29 11:14:05	商品Q	1700.0	201903	尾形小雁	おがた こがん	B市	ogata_kogan@example.com	2017-03-15	201703
2997	2019-07-14 12:56:49	商品H	800.0	201907	芦田博之	あしだ ひろゆき	E市	ashida_hiroyuki@example.com	2018-07-15	201807
2998	2019-07-21 00:31:36	商品D	400.0	201907	石田郁恵	いしだ いくえ	B市	ishida_ikue@example.com	2017-02-05	201702

2999 rows × 10 columns

データの最終的な整形（列の並びをわかりやすく）

In [47]:
```python
# 列の並びを指定
dump_data = join_data[['purchase_date', 'purchase_month', 'item_name', 'item_price', '顧客名', 'かな', '地域', 'メールアドレス', '登録日']]
dump_data
```

Out[47]:

	purchase_date	purchase_month	item_name	item_price	顧客名	かな	地域	メールアドレス	登録日
0	2019-06-13 18:02:34	201906	商品A	100.0	深井菜々美	ふかい ななみ	C市	fukai_nanami@example.com	2017-01-26
1	2019-07-13 13:05:29	201907	商品S	1900.0	浅田賢二	あさだ けんじ	C市	asada_kenji@example.com	2018-04-07
2	2019-05-11 19:42:07	201905	商品A	100.0	南部慶二	なんぶ けいじ	A市	nannbu_keiji@example.com	2018-06-19
3	2019-02-12 23:40:45	201902	商品Z	2600.0	麻生莉緒	あそう りお	D市	asou_rio@example.com	2018-07-22
4	2019-04-22 03:09:35	201904	商品A	100.0	平田鉄二	ひらた てつじ	D市	hirata_tetsuji@example.com	2017-06-07
...	...	...	...	...	...	...	...	...	...
2994	2019-02-15 02:56:39	201902	商品Y	2500.0	福島友也	ふくしま ともや	B市	fukushima_tomoya@example.com	2017-07-01
2995	2019-06-22 04:03:43	201906	商品M	1300.0	大倉晃司	おおくら こうじ	E市	ookura_kouji@example.com	2018-03-31
2996	2019-03-29 11:14:05	201903	商品Q	1700.0	尾形小雁	おがた こがん	B市	ogata_kogan@example.com	2017-03-15
2997	2019-07-14 12:56:49	201907	商品H	800.0	芦田博之	あしだ ひろゆき	E市	ashida_hiroyuki@example.com	2018-07-15
2998	2019-07-21 00:31:36	201907	商品D	400.0	石田郁恵	いしだ いくえ	B市	ishida_ikue@example.com	2017-02-05

2999 rows × 9 columns

整形したデータの出力（csv）

In [48]:
```python
# .pyファイルと同一の階層にcsvファイルの出力（インデックスは含めず）
dump_data.to_csv('dump_data.csv', index = False)
```

データの集計

In [49]:
```python
import_data = pd.read_csv('./dump_data.csv')
import_data
```

Out[49]:

	purchase_date	purchase_month	item_name	item_price	顧客名	かな	地域	メールアドレス	登録日
0	2019-06-13 18:02:34	201906	商品A	100.0	深井菜々美	ふかい ななみ	C市	fukai_nanami@example.com	2017-01-26 00:00:00
1	2019-07-13 13:05:29	201907	商品S	1900.0	浅田賢二	あさだ けんじ	C市	asada_kenji@example.com	2018-04-07 00:00:00
2	2019-05-11 19:42:07	201905	商品A	100.0	南部慶二	なんぶ けいじ	A市	nannbu_keiji@example.com	2018-06-19 00:00:00
3	2019-02-12 23:40:45	201902	商品Z	2600.0	麻生莉緒	あそう りお	D市	asou_rio@example.com	2018-07-22 00:00:00
4	2019-04-22 03:09:35	201904	商品A	100.0	平田鉄二	ひらた てつじ	D市	hirata_tetsuji@example.com	2017-06-07 00:00:00
...	...	...	...	...	...	...	...	...	...
2994	2019-02-15 02:56:39	201902	商品Y	2500.0	福島友也	ふくしま ともや	B市	fukushima_tomoya@example.com	2017-07-01 00:00:00
2995	2019-06-22 04:03:43	201906	商品M	1300.0	大倉晃司	おおくら こうじ	E市	ookura_kouji@example.com	2018-03-31 00:00:00
2996	2019-03-29 11:14:05	201903	商品Q	1700.0	尾形小雁	おがた こがん	B市	ogata_kogan@example.com	2017-03-15 00:00:00
2997	2019-07-14 12:56:49	201907	商品H	800.0	芦田博之	あしだ ひろゆき	E市	ashida_hiroyuki@example.com	2018-07-15 00:00:00
2998	2019-07-21 00:31:36	201907	商品D	400.0	石田郁恵	いしだ いくえ	B市	ishida_ikue@example.com	2017-02-05 00:00:00

2999 rows × 9 columns

In [50]:
```python
# 購入年月と商品の集計
byItem = import_data.pivot_table(index = 'purchase_month', columns = 'item_name', aggfunc = 'size', fill_value = 0)
byItem
```

Out[50]:

item_name	商品A	商品B	商品C	商品D	商品E	商品F	商品G	商品H	商品I	商品J	...	商品Q	商品R	商品S	商品T	商品U	商品V	商品W	商品X	商品Y	商品Z
purchase_month																					
201901	18	13	19	17	18	15	11	16	18	17	...	17	21	20	17	7	22	13	14	10	0
201902	19	14	26	21	16	14	14	17	12	14	...	22	22	22	23	19	22	24	16	11	1
201903	17	21	20	17	9	27	14	18	12	16	...	23	16	20	12	23	18	16	21	16	0
201904	17	19	24	20	18	17	14	11	18	13	...	20	20	16	16	11	15	14	16	20	0
201905	24	14	16	14	19	18	23	15	16	11	...	13	22	18	16	16	9	21	16	20	0
201906	24	12	11	19	13	18	15	13	19	22	...	15	16	21	12	18	20	17	15	13	0
201907	20	20	17	17	12	17	19	19	19	23	...	15	19	23	21	13	28	16	18	12	0

7 rows × 26 columns

In [51]:
```python
# 購入年月と売上金額の集計
byPrice = import_data.pivot_table(index = 'purchase_month', columns = 'item_name', values = 'item_price', aggfunc = 'sum', fill_value = 0)
byPrice
```

Out[51]:

item_name	商品A	商品B	商品C	商品D	商品E	商品F	商品G	商品H	商品I	商品J	...	商品Q	商品R	商品S	商品T	商品U	商品V	商品W	商品X	商品Y	商品Z
purchase_month																					
201901	1800	2600	5700	6800	9000	9000	7700	12800	16200	17000	...	28900	37800	38000	34000	14700	48400	29900	33600	25000	0
201902	1900	2800	7800	8400	8000	8400	9800	13600	10800	14000	...	37400	39600	41800	46000	39900	48400	55200	38400	27500	2600
201903	1700	4200	6000	6800	4500	16200	9800	14400	10800	16000	...	39100	28800	38000	24000	48300	39600	36800	50400	40000	0
201904	1700	3800	7200	8000	9000	10200	9800	8800	16200	13000	...	34000	36000	30400	32000	23100	33000	32200	38400	50000	0
201905	2400	2800	4800	5600	9500	10800	16100	12000	14400	11000	...	22100	39600	34200	32000	33600	19800	48300	38400	50000	0
201906	2400	2400	3300	7600	6500	10800	10500	10400	17100	22000	...	25500	28800	39900	24000	37800	44000	39100	36000	32500	0
201907	2000	4000	5100	6800	6000	10200	13300	15200	17100	23000	...	25500	34200	43700	42000	27300	61600	36800	43200	30000	0

7 rows × 26 columns

In [52]:
```python
# 購入年月と各顧客における購入数の集計
byCustomer = import_data.pivot_table(index = 'purchase_month', columns = '顧客名', aggfunc = 'size', fill_value = 0)
byCustomer
```

Out[52]:

顧客名	さだ千佳子	中仁晶	中田美智子	丸山光臣	久保田倫子	亀井一徳	五十嵐春樹	井上桃子	井口寛治	井川真悠子	...	香椎優一	高原充則	高梨結衣	高沢美咲	高田さんま	鳥居広司	鶴岡薫	麻生莉緒	黄川田博之	黒谷長利
purchase_month																					
201901	3	1	4	2	2	0	5	3	3	1	...	0	1	1	1	5	2	0	2	2	5
201902	9	1	2	2	1	4	2	1	0	4	...	4	0	3	2	0	1	2	4	0	1
201903	1	2	1	6	1	4	3	3	2	2	...	3	1	6	2	4	2	4	2	2	1
201904	0	3	1	2	0	2	2	0	3	2	...	2	4	2	3	4	3	2	1	2	0
201905	3	2	5	2	4	1	2	1	3	3	...	1	1	1	0	2	2	3	4	4	1
201906	1	3	0	4	1	1	1	2	2	3	...	7	3	0	2	1	0	2	1	2	4
201907	3	0	3	2	5	3	5	2	5	5	...	2	4	4	2	0	2	4	3	4	1

7 rows × 199 columns

In [53]:
```python
# 購入年月と地域別販売数の集計
byRegion = import_data.pivot_table(index = 'purchase_month', columns = '地域', aggfunc = 'size', fill_value = 0)
byRegion
```

Out[53]:

地域	A市	B市	C市	D市	E市	F市	G市	H市
purchase_month								
201901	59	55	72	34	49	57	49	42
201902	71	46	65	48	61	52	43	63
201903	64	52	57	43	52	59	51	59
201904	64	48	54	45	48	58	40	52
201905	57	52	68	48	59	65	35	43
201906	53	47	61	30	51	51	58	58
201907	76	53	61	42	54	64	47	54