ウェブからの注文数を分析

1. データの読み込み

In [1]:
```python
from numpy.core.defchararray import join
import pandas as pd
```

In [2]:
```python
# csvファイルの読み込みとデータの先頭5列の表示
customer_master = pd.read_csv('./data/data1/customer_master.csv')
customer_master.head()
```

Out[2]:

	customer_id	customer_name	registration_date	customer_name_kana	email	gender	age	birth	pref
0	IK152942	平田 裕次郎	2019-01-01 00:25:33	ひらた ゆうじろう	hirata_yuujirou@example.com	M	29	1990/6/10	石川県
1	TS808488	田村 詩織	2019-01-01 01:13:45	たむら しおり	tamura_shiori@example.com	F	33	1986/5/20	東京都
2	AS834628	久野 由樹	2019-01-01 02:00:14	ひさの ゆき	hisano_yuki@example.com	F	63	1956/1/2	茨城県
3	AS345469	鶴岡 薫	2019-01-01 04:48:22	つるおか かおる	tsuruoka_kaoru@example.com	M	74	1945/3/25	東京都
4	GD892565	大内 高史	2019-01-01 04:54:51	おおうち たかし	oouchi_takashi@example.com	M	54	1965/8/5	千葉県

In [3]:
```python
# csvファイルの読み込み
item_master = pd.read_csv('./data/data1/item_master.csv')
item_master.head()
```

Out[3]:

	item_id	item_name	item_price
0	S001	PC-A	50000
1	S002	PC-B	85000
2	S003	PC-C	120000
3	S004	PC-D	180000
4	S005	PC-E	210000

In [4]:
```python
# csvファイルの読み込み
transaction_1 = pd.read_csv('./data/data1/transaction_1.csv')
transaction_1.head()
```

Out[4]:

	transaction_id	price	payment_date	customer_id
0	T0000000113	210000	2019-02-01 01:36:57	PL563502
1	T0000000114	50000	2019-02-01 01:37:23	HD678019
2	T0000000115	120000	2019-02-01 02:34:19	HD298120
3	T0000000116	210000	2019-02-01 02:47:23	IK452215
4	T0000000117	170000	2019-02-01 04:33:46	PL542865

In [5]:
```python
# csvファイルの読み込み
transaction_detail_1 = pd.read_csv('./data/data1/transaction_detail_1.csv')
transaction_1.head()
```

Out[5]:

	transaction_id	price	payment_date	customer_id
0	T0000000113	210000	2019-02-01 01:36:57	PL563502
1	T0000000114	50000	2019-02-01 01:37:23	HD678019
2	T0000000115	120000	2019-02-01 02:34:19	HD298120
3	T0000000116	210000	2019-02-01 02:47:23	IK452215
4	T0000000117	170000	2019-02-01 04:33:46	PL542865

2. データの結合（ユニオン）

今回のデータ結合ではデータを縦方向に結合（行を追加）

In [6]:
```python
# csvファイルの読み込み
transaction_2 = pd.read_csv('./data/data1/transaction_2.csv')
transaction_2.head()
```

Out[6]:

	transaction_id	price	payment_date	customer_id
0	T0000005113	295000	2019-06-15 07:20:27	TS169261
1	T0000005114	50000	2019-06-15 07:35:47	HI599892
2	T0000005115	85000	2019-06-15 07:56:36	HI421757
3	T0000005116	50000	2019-06-15 08:40:55	OA386378
4	T0000005117	120000	2019-06-15 08:44:23	TS506913

In [7]:
```python
# concat関数で2つのデータフレームを結合（インデックスは含めず結合）
transaction = pd.concat([transaction_1, transaction_2], ignore_index = True)
```

In [8]:
```python
# 結合の確認
print(f'{len(transaction_1)} + {len(transaction_2)} = {len(transaction)}')
```

```
5000 + 1786 = 6786
```

In [9]:
```python
# transaction_detail_1と2のデータを結合
transaction_detail_2 = pd.read_csv('./data/data1/transaction_detail_2.csv')
transaction_detail_2.head()
```

Out[9]:

	detail_id	transaction_id	item_id	quantity
0	5000	T0000004870	S002	3
1	5001	T0000004871	S003	1
2	5002	T0000004872	S001	2
3	5003	T0000004873	S004	1
4	5004	T0000004874	S003	2

In [10]:
```python
transaction_detail = pd.concat([transaction_detail_1, transaction_detail_2], ignore_index = True)
print(f'transaction_detail_1：{transaction_detail_1.shape} \ntransaction_detail_2：{transaction_detail_2.shape}')
transaction_detail.shape
```

```
transaction_detail_1：(5000, 4)
transaction_detail_2：(2144, 4)
```

Out[10]:
```
(7144, 4)
```

売上データ同士を結合（ジョイン）

In [11]:
```python
join_data = pd.merge(transaction_detail, transaction[['transaction_id', 'payment_date', 'customer_id']], on = 'transaction_id', how = 'left')
join_data.head()
```

Out[11]:

	detail_id	transaction_id	item_id	quantity	payment_date	customer_id
0	0	T0000000113	S005	1	2019-02-01 01:36:57	PL563502
1	1	T0000000114	S001	1	2019-02-01 01:37:23	HD678019
2	2	T0000000115	S003	1	2019-02-01 02:34:19	HD298120
3	3	T0000000116	S005	1	2019-02-01 02:47:23	IK452215
4	4	T0000000117	S002	2	2019-02-01 04:33:46	PL542865

マスターデータの結合（ジョイン）

In [12]:
```python
join_data = pd.merge(join_data, customer_master, on = 'customer_id', how = 'left')
join_data = pd.merge(join_data, item_master, on = 'item_id', how = 'left')
join_data.head()
```

Out[12]:

	detail_id	transaction_id	item_id	quantity	payment_date	customer_id	customer_name	registration_date	customer_name_kana	email	gender	age
0	0	T0000000113	S005	1	2019-02-01 01:36:57	PL563502	井本 芳正	2019-01-07 14:34:35	いもと よしまさ	imoto_yoshimasa@example.com	M	30
1	1	T0000000114	S001	1	2019-02-01 01:37:23	HD678019	三船 六郎	2019-01-27 18:00:11	みふね ろくろう	mifune_rokurou@example.com	M	73
2	2	T0000000115	S003	1	2019-02-01 02:34:19	HD298120	山根 小雁	2019-01-11 08:16:02	やまね こがん	yamane_kogan@example.com	M	42
3	3	T0000000116	S005	1	2019-02-01 02:47:23	IK452215	池田 菜摘	2019-01-10 05:07:38	いけだ なつみ	ikeda_natsumi@example.com	F	47
4	4	T0000000117	S002	2	2019-02-01 04:33:46	PL542865	栗田 憲一	2019-01-25 06:46:05	くりた けんいち	kurita_kenichi@example.com	M	74

In [13]:
```python
# joinデータにデータを追加
join_data['price'] = join_data['quantity'] * join_data['item_price']
join_data[['price']].tail()
```

Out[13]:

	price
7139	180000
7140	85000
7141	100000
7142	85000
7143	85000

In [14]:
```python
# 確認
print(join_data['price'].sum() == transaction['price'].sum())
```

```
True
```

記述統計を行う

In [15]:
```python
join_data.describe()
print(join_data['payment_date'].min())
print(join_data['payment_date'].max())
```

```
2019-02-01 01:36:57
2019-07-31 23:41:38
```

In [16]:
```python
# 各列における欠損値の合計
join_data.isnull().sum()
```

Out[16]:
```
detail_id             0
transaction_id        0
item_id               0
quantity              0
payment_date          0
customer_id           0
customer_name         0
registration_date     0
customer_name_kana    0
email                 0
gender                0
age                   0
birth                 0
pref                  0
item_name             0
item_price            0
price                 0
dtype: int64
```

In [17]:
```python
# データフレームにひとつでも欠損値が含まれるか確認
join_data.isna().any().any()
```

Out[17]:
```
False
```

データフレームの各データ型を確認・変換する

In [18]:
```python
# 全体の一覧
join_data.dtypes
```

Out[18]:
```
detail_id              int64
transaction_id        object
item_id               object
quantity               int64
payment_date          object
customer_id           object
customer_name         object
registration_date     object
customer_name_kana    object
email                 object
gender                object
age                    int64
birth                 object
pref                  object
item_name             object
item_price             int64
price                  int64
dtype: object
```

In [19]:
```python
# 個別で確認
join_data.dtypes[['payment_date']]
```

Out[19]:
```
payment_date    object
dtype: object
```

In [20]:
```python
# データ型の変換
join_data['payment_date'] = pd.to_datetime(join_data['payment_date'])
join_data['payment_month'] = join_data['payment_date'].dt.strftime('%Y%m') # dt以下の部分で列全体に任意の時刻表示（文字型）を適用
join_data[['payment_date', 'payment_month']]
```

Out[20]:

	payment_date	payment_month
0	2019-02-01 01:36:57	201902
1	2019-02-01 01:37:23	201902
2	2019-02-01 02:34:19	201902
3	2019-02-01 02:47:23	201902
4	2019-02-01 04:33:46	201902
...	...	...
7139	2019-07-31 21:20:44	201907
7140	2019-07-31 21:52:48	201907
7141	2019-07-31 23:35:25	201907
7142	2019-07-31 23:39:35	201907
7143	2019-07-31 23:41:38	201907

7144 rows × 2 columns

月別の集計（groupby）

In [21]:
```python
# 月別の販売数と価格
join1 = join_data.groupby('payment_month').sum()[['quantity', 'price']]
```

In [22]:
```python
join_data.groupby(['payment_month', 'item_name']).sum()[['price', 'quantity']]
```

Out[22]:

payment_month	item_name	price	quantity
201902	PC-A	24150000	483
	PC-B	25245000	297
	PC-C	19800000	165
	PC-D	31140000	173
	PC-E	59850000	285
201903	PC-A	26000000	520
	PC-B	25500000	300
	PC-C	19080000	159
	PC-D	25740000	143
	PC-E	64050000	305
201904	PC-A	25900000	518
	PC-B	23460000	276
	PC-C	21960000	183
	PC-D	24300000	135
	PC-E	64890000	309
201905	PC-A	24850000	497
	PC-B	25330000	298
	PC-C	20520000	171
	PC-D	25920000	144
	PC-E	58800000	280
201906	PC-A	26000000	520
	PC-B	23970000	282
	PC-C	21840000	182
	PC-D	28800000	160
	PC-E	63420000	302
201907	PC-A	25250000	505
	PC-B	28220000	332
	PC-C	19440000	162
	PC-D	26100000	145
	PC-E	71610000	341

ピボットテーブルによる集計

In [23]:
```python
pd.pivot_table(join_data, index = 'item_name', columns = 'payment_month', values = ['price', 'quantity'], aggfunc = 'sum')
```

Out[23]:

	price						quantity					
payment_month	201902	201903	201904	201905	201906	201907	201902	201903	201904	201905	201906	201907
item_name												
PC-A	24150000	26000000	25900000	24850000	26000000	25250000	483	520	518	497	520	505
PC-B	25245000	25500000	23460000	25330000	23970000	28220000	297	300	276	298	282	332
PC-C	19800000	19080000	21960000	20520000	21840000	19440000	165	159	183	171	182	162
PC-D	31140000	25740000	24300000	25920000	28800000	26100000	173	143	135	144	160	145
PC-E	59850000	64050000	64890000	58800000	63420000	71610000	285	305	309	280	302	341

データの可視化

In [24]:
```python
import matplotlib.pyplot as plt

# データの整形
graph_data = pd.pivot_table(join_data, index = 'payment_month', columns = 'item_name', values = 'price', aggfunc = 'sum')

# 図の表示
plt.plot(list(graph_data.index), graph_data['PC-A'], label = 'PC-A')
plt.plot(list(graph_data.index), graph_data['PC-B'], label = 'PC-B')
plt.plot(list(graph_data.index), graph_data['PC-C'], label = 'PC-C')
plt.plot(list(graph_data.index), graph_data['PC-D'], label = 'PC-D')
plt.plot(list(graph_data.index), graph_data['PC-E'], label = 'PC-E')

plt.legend()
plt.show()
```