出したりするなど、様々なことが可能です。今回は、strftimeを使用し、文字列として年月を作成しました。

それでは、集計していきましょう。

```
join_data.groupby("payment_month").sum()["price"]
```

■図1-10：月別売上の集計結果

```
In [25]: join_data.groupby("payment_month").sum()["price"]

Out[25]: payment_month
         201902   160185000
         201903   160370000
         201904   160510000
         201905   155420000
         201906   164030000
         201907   170620000
         Name: price, dtype: int64
```

実行すると、月別の売上が表示されます。

groupbyは、まとめたい列(payment_month)と、集計方法(sum)を記述します。また、priceのみを表示させるために、最後にprice列を指定しています。

この結果を見ると、5月は少し売上が下がりましたが、6月、7月と回復してきており、7月が半年間で最も売上が高いです。およそ、1億6千万円程度の単月売上が出ており、年間で20億円くらいの売上は期待できそうです。

では、どの商品が売れ筋なのでしょうか。

月別かつ商品別に集計してみましょう。

ノック9：月別、商品別でデータを集計してみよう

月別かつ商品別に、売上の合計値、数量を表示してみましょう。

先ほどと同様にgroupbyを使用して集計します。

```
join_data.groupby(["payment_month","item_name"]).sum()[["price", "quantit
y"]]
```


■図1-11：月別、商品別の集計

ノック9：月別、商品別でデータを集計してみよう

```
In [35]: join_data.groupby(["payment_month","item_name"]).sum() ["price", "quantity"]
```

Out[35]:

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

実行すると、月別、商品別のprice、quantityの集計結果が表示されます。

今回のようにまとめたい列が複数ある場合、groupbyでは、リスト型で指定することができます。

少し表示が直感的に理解しにくいので、pivot_tableを使用して集計してみましょう。

```
pd.pivot_table(join_data, index='item_name', columns='payment_month', valu
es=['price', 'quantity'], aggfunc='sum')
```