挙げだしたらキリがありませんが、この統計量だけからでも多くの情報を得ることができ、この先の分析に繋げることができます。

今回用いたdescribe()は数値データの集計を行なってくれます。追加で見ておく必要があるとすると、データの期間範囲です。大抵の場合は、データを受領する際にヒアリングをして把握しているケースが多いですが、確認のために見ておくことをお勧めします。

```
print(join_data["payment_date"].min())
print(join_data["payment_date"].max())
```

実行すると、2019年2月1日から2019年7月31日までのデータ範囲であることがわかります。

ノック8：月別でデータを集計してみよう

全体の数字感を把握したところで、まずは時系列で状況を見てみましょう。今回のケースのように半年程度のデータであれば影響は出ることはあまりありませんが、過去数年のデータ等を扱うとなると、ビジネスモデルの変化等により一纏めに分析すると見誤るケースがあります。その場合、データ範囲を絞るケースもあります。

また、全体的に売上が伸びているのか、落ちているのかを把握するのは、分析の第一歩と言えるでしょう。

まずは、月別に集計して一覧表示してみましょう。

流れとしては、購入日であるpayment_dateから年月の列を作成した後、年月列単位でpriceを集計し、表示します。

まずは、payment_dateのデータ型を確認しましょう。

```
join_data.dtypes
```


図1-8：データ型の確認

ノック8：月別でデータを集計してみよう

```
In [17]: join_data.dtypes

Out[17]: 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
```

実行すると、列毎にデータ型を確認できます。今回加工したいデータは、payment_dateでobject型となっています。このまま文字列として扱うこともできますが、今後も踏まえてdatetime型に変更して、年月列の作成を行いましょう。

```
join_data["payment_date"] = pd.to_datetime(join_data["payment_date"])
join_data["payment_month"] = join_data["payment_date"].dt.strftime("%Y%m")
join_data[["payment_date", "payment_month"]].head()
```

図1-9：年月列の作成

```
In [22]: join_data["payment_date"] = pd.to_datetime(join_data["payment_date"])
         join_data["payment_month"] = join_data["payment_date"].dt.strftime("%Y%m")
         join_data[["payment_date", "payment_month"]].head()

Out[22]:
```

	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

1行目でdatetime型に変換し、2行目で新たな列payment_monthを年月単位で作成しています。pandasのdatetime型は、dtを使うことで、年のみを抽