数値から日付に補正したデータ(fromSerial)と、書式を変更したデータ(fromString)を.concatで結合し、元の「登録日」に代入して更新しています。
結果を見ると、登録日が奇麗に補正された事が確認できます。

登録日から登録月を算出し、集計をしてみましょう。

```
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"]
print(rslt)
print(len(kokyaku_data))
```

図2-22：登録月の集計結果

```
In [22]: 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"]
         print(rslt)
         print(len(kokyaku_data))

         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    18
         201806    13
         201807    17
         201904     2
         Name: 顧客名, dtype: int64
         200
```

補正した日付から「登録年月」を作成し、groupby()で集計します。
groupbyを行った後、.count()を行う事でデータの件数を集計できます。集計した結果から「顧客名」列の集計結果を画面上に表示しています。
また、検証用にlen(kokyaku_data)で顧客台帳の総データ件数を表示します。

年月単位での集計結果を足し算すると、200となりますので、日付の補正が正しく行われた事も併せて確認する事ができました。

最初に実施した処理と同じく、登録日列に数値データが残っていないか確認しましょう。

```
flg_is_serial = kokyaku_data["登録日"].astype("str").str.isdigit()
flg_is_serial.sum()
```

図2-23：数値項目の有無

```
In [23]: flg_is_serial = kokyaku_data["登録日"].astype("str").str.isdigit()
         flg_is_serial.sum()

Out[23]: 0
```

最初は「22件」だった結果が「0件」となり、すべての数値データが日付に補正された事が確認できました。

ノック18：顧客名をキーに2つのデータを結合(ジョイン)しよう

売上履歴と顧客台帳を結合し、集計のベースとなるデータを作成しましょう。
第1章は整合性のあるデータだったため、ID等の共通するキーで結合する事ができましたが、今回は2つのデータの別の列を指定して結合を実施します。

```
join_data = pd.merge(uriage_data, kokyaku_data, left_on="customer_name", r
ight_on="顧客名", how="left")
join_data = join_data.drop("customer_name", axis=1)
join_data
```