■図2-11：金額欠損値の補完処理結果

```
In [12]: 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()
```

Out[12]:

	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	南郷慶二	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

1行目でitem_priceの中で欠損値のある個所を特定します。この処理を実行する事でflg_is_null変数にどの行に欠損値が存在するかが保持されます。

2行目でループ処理を行いますが、ループ条件としてlist(uriage_data.loc[flg_is_null, “item_name”].unique())というデータを用いています。

これは、先ほど生成したflg_is_nullによりデータが欠損している商品名の一覧を作成する処理になっています。1行に複数の処理が記載されていますので、分割して説明していきます。

まずは一番大枠のlist()は変数の値をリスト形式に変換する処理になります。

次にuriage_data.loc[flg_is_null, “item_name”]ですが、.loc関数は条件を付与し、それに合致するデータを抽出する事ができます。今回の条件とは「金額が欠損している」となるため、先ほど生成したflg_is_nullを渡すことで条件付けしています。2番目のitem_nameは条件に合致したデータの、どの列を取得するかを指定します。今回は欠損値の存在する商品名を抽出しています。

最後のunique()は抽出した商品名の重複をなくし、一意にしています。無駄なループ処理をなくすために行っています。

続いて、ループ処理内のprice = uriage_data.loc[(~flg_is_null) & (uriage_data["item_name"] == trg), "item_price"].max()について説明します。こちらはループ変数である「欠損値がある商品名」を用いて、同じ商品で金額が正しく記載されている行を.locで探し、その金額を取得しています。.loc()の条件には「(~flg_is_null) & (uriage_data["item_name"] == trg)」のように複数の条件を指定する事が可能です。~flg_is_nullの「~」は否定演算子といい、

「flg_is_null == False」と同義です。これにより、欠損値がある商品と同じ商品データから金額を取得する事ができました。

次に取得した金額でデータを補完していきます。

uriage_data["item_price"].loc[(flg_is_null) & (uriage_data["item_name"]==trg)] = priceでは、売上履歴のitem_price列に対して.locを行い、欠損を起こしている対象データを抽出し、先ほど生成したpriceを欠損値に代入しています。

ループ後、補完後の売上履歴の先頭5行を表示しています。

商品名と金額の揺れが処理され、だいぶ綺麗なデータに見えてきましたが、油断せずに検証を実施します。

```
uriage_data.isnull().any(axis=0)
```

■図2-12：欠損値チェック結果(補完後)

```
In [13]: uriage_data.isnull().any(axis=0)
Out[13]: purchase_date     False
         item_name         False
         item_price        False
         customer_name     False
         purchase_month    False
         dtype: bool
```

先ほどと同じ処理のuriage_data.isnull().any(axis=0)を実行し、結果を確認します。

先ほどと異なり、item_price Falseとなっている事から、無事item_priceの金額欠損がなくなった事が確認できます。

次に、各商品の金額が正しく補完されたか確認してみましょう。

```
for trg in list(uriage_data["item_name"].sort_values().unique()):
    print(trg + "の最大額：" + str(uriage_data.loc[uriage_data["item_name"]==trg]["item_price"].max()) + "の最小額：" + str(uriage_data.loc[uriage_data["item_name"]==trg]["item_price"].min(skipna=False)))
```