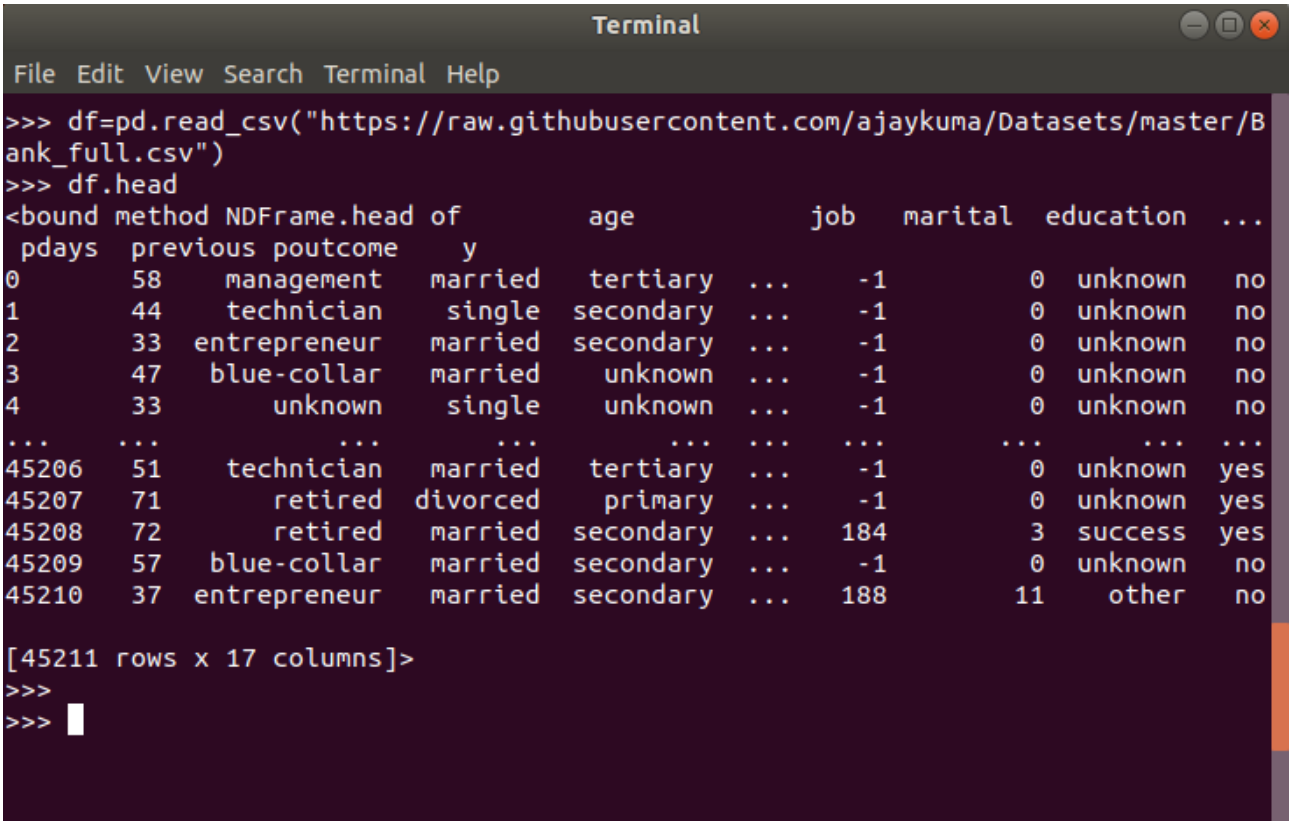
Assignment

NAME-TUSHAR RAJ VERMA

1.
Create a DF using python pandas n numpy using a dataset from
:
https://github.com/ajaykuma/datasets
For ex: Bank_full.csv .This dataset contains data of a marketing campaign that was conducted to find out if customers would be interested in availing a deposit facility and consider depositing money in bank for long term.
Write the code that answers these questions:
a.
Success and failure rate of the campaign. (Hint: The last column 'Y' shows response of customers)
b.
If Balance of a customer is deciding factor in hi
s/her decision.
c.
If Marital status, age are deciding factors for their decision.
d.
Create a function that groups age into categories/bins [3 groups] and using this function add a new column to existing DF or create a new DF. Check which age group has inclinat
ion towards availing this facility.

Ans-1
setting the dataset



a. Success and failure rate of the campaign.

```
Terminal
File  Edit  View  Search  Terminal  Help
>>> count=pd.DataFrame(df['y'].value_counts())
>>> count=(count.div(count['y'].sum()))*100
>>> print("The failure rate is:- {}".format(count.loc['no'].values))
The failure rate is:- [88.30151954]
>>> print("The success rate is:- {}".format(count.loc['yes'].values))
The success rate is:- [11.69848046]
>>>
```

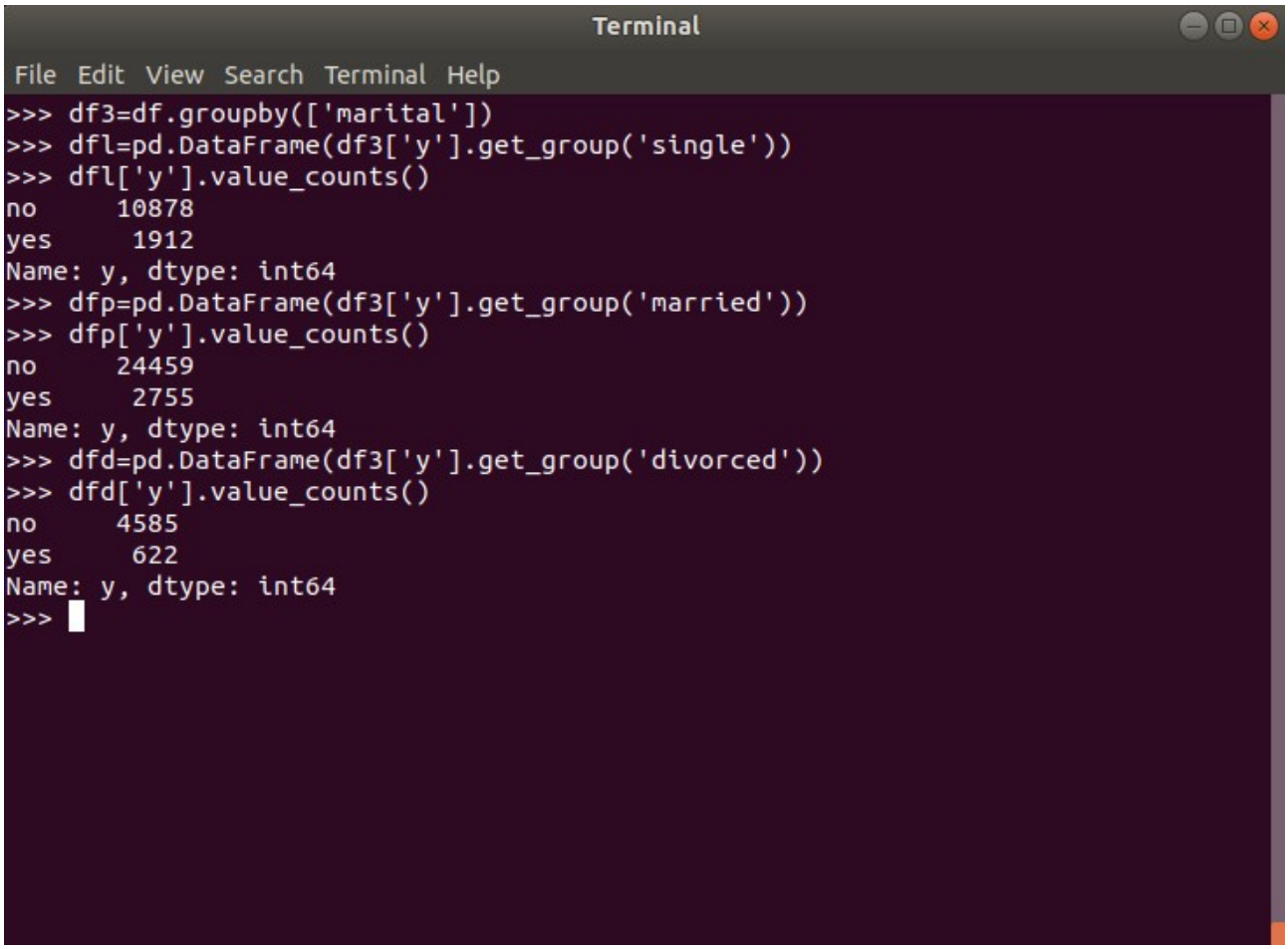
b.Balance is affecting as every group having more count of no

```
>>> li=[]
>>> for x in df['balance']:
...     if(x>df['balance'].mean()):
...             li.append('high')
...     else:
...             li.append('low')
...

>>>
>>> df['balance']=li
```

```
>>> df3=df.groupby(['balance'])
>>> dfb=pd.DataFrame(df3['y'].get_group('high'))
>>> dfb['y'].value_counts()
no     9869
yes    1879
Name: y, dtype: int64
>>> dfb=pd.DataFrame(df3['y'].get_group('low'))
>>> dfb['y'].value_counts()
no     30053
yes     3410
Name: y, dtype: int64
>>>
```

c.Marital status affects the ouput as every group counts more towards no



d.Groping of age and adding that column

function declare

```
>>> li=[]
>>> def conver(d):
...     if d<=25:
...             li.append('Youth')
...     elif d>25 and d<=50:
...             li.append('MidAge')
...     else:
...             li.append('Senior citizen')
...
```

passing the dataframe[‘age’] to function

```
NameError: name 'x' is not defined
>>> for x in df['age']:
...     conver(x)
...
>>> df
       age           job   marital  education  ... pdays  previous poutcome    y
0       58    management   married   tertiary  ...    -1         0  unknown   no
1       44    technician    single  secondary  ...    -1         0  unknown   no
2       33  entrepreneur   married  secondary  ...    -1         0  unknown   no
3       47   blue-collar   married    unknown  ...    -1         0  unknown   no
4       33       unknown    single    unknown  ...    -1         0  unknown   no
...    ...           ...       ...        ...  ...   ...       ...      ...  ...
45206   51    technician   married   tertiary  ...    -1         0  unknown  yes
45207   71       retired  divorced    primary  ...    -1         0  unknown  yes
45208   72       retired   married  secondary  ...   184         3  success  yes
45209   57   blue-collar   married  secondary  ...    -1         0  unknown   no
45210   37  entrepreneur   married  secondary  ...   188        11    other   no

[45211 rows x 17 columns]
>>> df['AgeCategory']=li
>>> df
       age           job   marital  ... poutcome    y     AgeCategory
```

NEW COLUMN ADDED

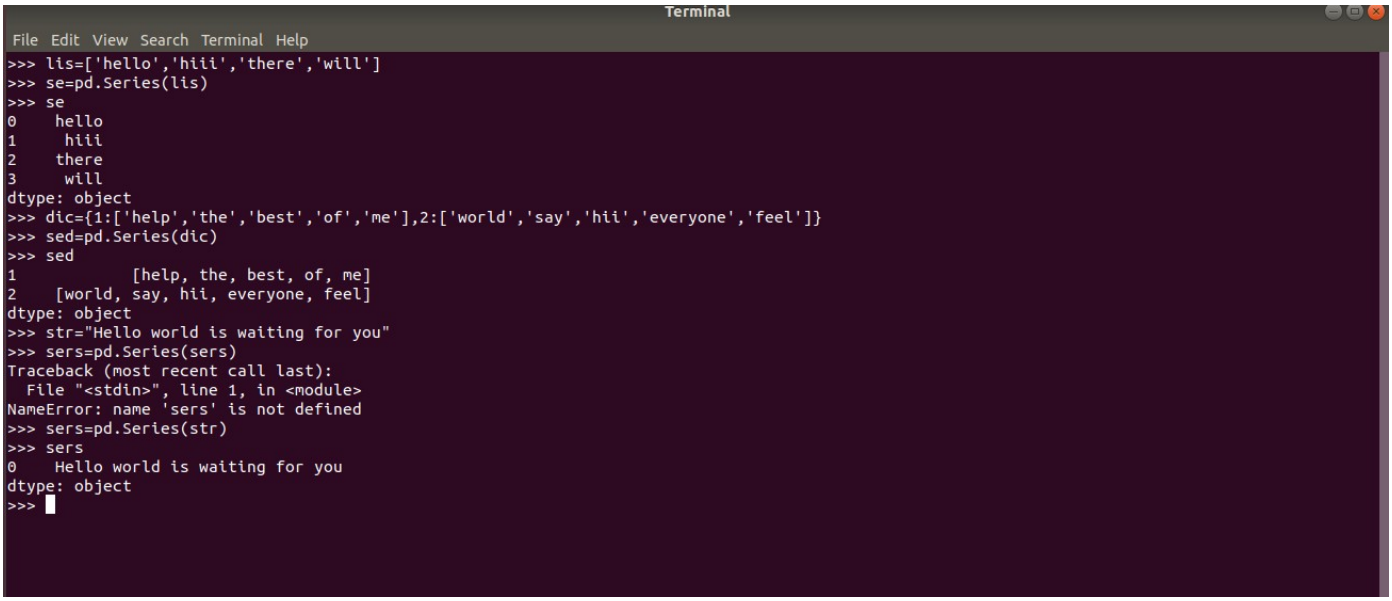
```
                                Terminal
File  Edit  View  Search  Terminal  Help
45207   71       retired  divorced    primary  ...    -1         0  unknown  yes
45208   72       retired   married  secondary  ...   184         3  success  yes
45209   57   blue-collar   married  secondary  ...    -1         0  unknown   no
45210   37  entrepreneur   married  secondary  ...   188        11    other   no

[45211 rows x 17 columns]
>>> df['AgeCategory']=li
>>> df
       age           job   marital  ... poutcome    y     AgeCategory
0       58    management   married  ...  unknown   no  Senior citizen
1       44    technician    single  ...  unknown   no          MidAge
2       33  entrepreneur   married  ...  unknown   no          MidAge
3       47   blue-collar   married  ...  unknown   no          MidAge
4       33       unknown    single  ...  unknown   no          MidAge
...    ...           ...       ...  ...      ...  ...             ...
45206   51    technician   married  ...  unknown  yes  Senior citizen
45207   71       retired  divorced  ...  unknown  yes  Senior citizen
45208   72       retired   married  ...  success  yes  Senior citizen
45209   57   blue-collar   married  ...  unknown   no  Senior citizen
45210   37  entrepreneur   married  ...    other   no          MidAge

[45211 rows x 18 columns]
>>>
```

Affecting the output as ‘no’ counts are greater for each group

2.Demonstrate creation of Series from collections: List, Dictionary, String.
Ans-2



3.Demonstrate creation of DF from collections : Lists, Dictionary, Series.
Ans-3

```
File  Edit  View  Search  Terminal  Help
>>> lis=['hello','hiii','there','will']
>>> dfl=pd.DataFrame(lis)
>>> dfl
       0
0  hello
1   hiii
2  there
3   will
>>> dic={1:['help','the','best','of','me'],2:['world','say','hii','everyone','feel']}
>>> dfd=pd.DataFrame(dic)
>>> dfd
      1         2
0  help     world
1   the       say
2  best       hii
3    of  everyone
4    me      feel
>>> str="Hello world is waiting for you"
>>> dfs=pd.DataFrame(str)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/tushar/.local/lib/python3.6/site-packages/pandas/core/frame.py", line 509, in __init__
    raise ValueError("DataFrame constructor not properly called!")
ValueError: DataFrame constructor not properly called!
>>> sers
0    Hello world is waiting for you
dtype: object
>>> dfs=pd.DataFrame(sers)
>>> dfs
                             0
0  Hello world is waiting for you
>>> type(dfs)
<class 'pandas.core.frame.DataFrame'>
>>>
```

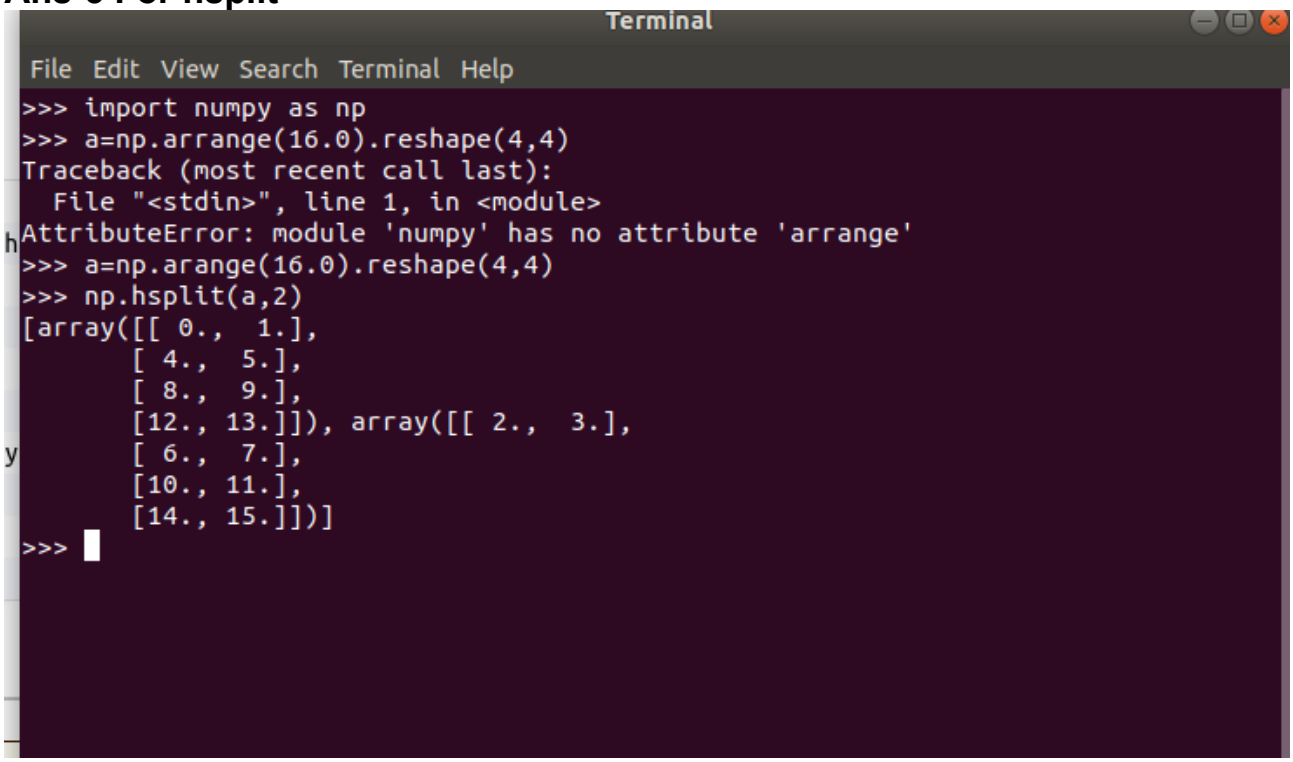
4.Create a series from list of numbers which may have duplicates and demonstrate usage of pandas to extract duplicates from this list

Ans-4

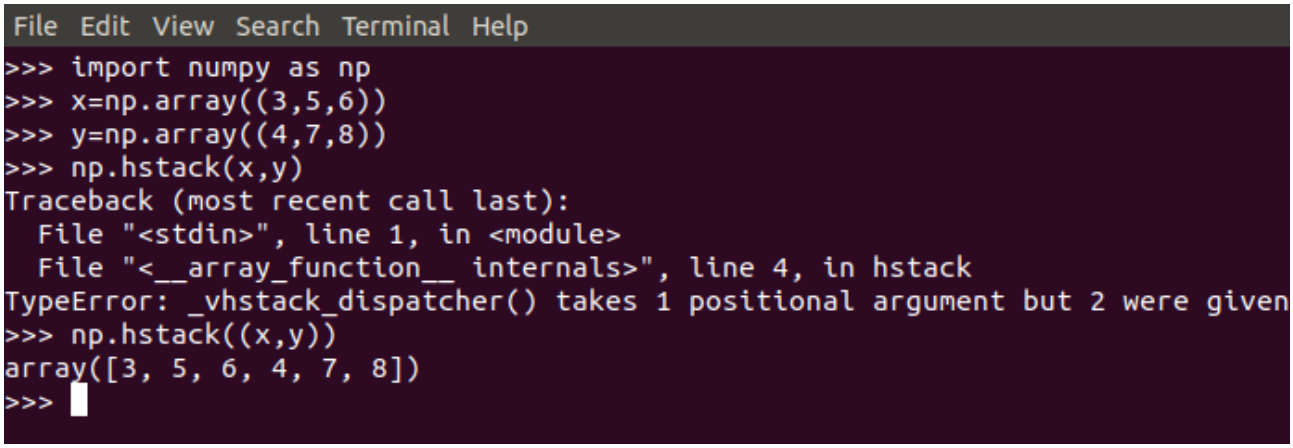
```
Terminal
File  Edit  View  Search  Terminal  Help
tushar@tushar:~ $ python3
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> lis=[1,2,3,4,5,1,2,3,6,7,8,2,3]
>>> ser=pd.Series(lis)
>>> ser.drop_duplicates()
0     1
1     2
2     3
3     4
4     5
8     6
9     7
10    8
dtype: int64
>>> list(ser)
[1, 2, 3, 4, 5, 1, 2, 3, 6, 7, 8, 2, 3]
>>> ser=ser.drop_duplicates()
>>> list(ser)
[1, 2, 3, 4, 5, 6, 7, 8]
>>>
```

5.Demonstrate examples to show usage of “hsplit” and “hstack” on a numpy series.

Ans-5 For hsplit



for hstack



6.Demonstrate your knowledge with one example each for:
a.Hierarchical indexing
b.Grouping data into bins
c.Using stack and unstack functions to reshape DF with hierarchical indexes and back to DF.
d.Using merge to do a “inner”, “outer”, “left” join and using suffixes.

Ans-6
a.Hierarchical indexing

```
>>> frame=pd.DataFrame(np.arange(10).reshape((5,2)),index=[['a','b','c','d','e'][1,2,>>> frame=pd.D
=[['a','b','c','d','e'],[1,2,3,4,5]],columns=[['germany','canada'],['India','USA']])
>>>
>>> frame
    germany canada
      India    USA
a 1       0      1
b 2       2      3
c 3       4      5
d 4       6      7
e 5       8      9
>>>
```

b.Grouping data into bins

```
>>> pd.cut(np.array([1, 7, 5, 4, 6, 3]),
... 3, labels=["bad", "medium", "good"])
[bad, good, medium, medium, good, bad]
Categories (3, object): [bad < medium < good]
>>>
```

c.Using stack and unstack functions to reshape DF with hierarchical indexes
and back to DF.

```
>>> frame=frame.stack()
>>> frame
          canada  germany
a 1 India     NaN      0.0
    USA       1.0      NaN
b 2 India     NaN      2.0
    USA       3.0      NaN
c 3 India     NaN      4.0
    USA       5.0      NaN
d 4 India     NaN      6.0
    USA       7.0      NaN
e 5 India     NaN      8.0
    USA       9.0      NaN
>>> frame=frame.unstack()
>>> frame
    canada      germany
     India  USA   India USA
a 1    NaN  1.0     0.0 NaN
b 2    NaN  3.0     2.0 NaN
c 3    NaN  5.0     4.0 NaN
d 4    NaN  7.0     6.0 NaN
e 5    NaN  9.0     8.0 NaN
>>>
```

d.Using merge to do a “inner”, “right”, “left” join and using suffixes.

```
>>> # data frame 1
... d1 = {'Customer_id':pd.Series([1,2,3,4,5,6]),
...   'Product':pd.Series(['Oven','Oven','Oven','Television','Television','Television'])}
>>> df1 = pd.DataFrame(d1)
>>> # data frame 2
... d2 = {'Customer_id':pd.Series([2,4,6]),
...     'State':pd.Series(['California','California','Texas'])}
>>> df2 = pd.DataFrame(d2)
>>> pd.merge(df1, df2, on='Customer_id', how='inner')
   Customer_id     Product       State
0            2        Oven  California
1            4  Television  California
2            6  Television       Texas
>>> pd.merge(df1, df2, on='Customer_id', how='left')
   Customer_id     Product       State
0            1        Oven         NaN
1            2        Oven  California
2            3        Oven         NaN
3            4  Television  California
4            5  Television         NaN
5            6  Television       Texas
>>> pd.merge(df1, df2, on='Customer_id', how='right')
   Customer_id     Product       State
0            2        Oven  California
1            4  Television  California
2            6  Television       Texas
>>>
```