Categorical Data

In [3]:
```
import pandas as pd
import numpy as np
```

In [5]:
```
fruit=pd.Series(['apple', 'orange','apple','apple','banana']*2)
```

In [6]:
```
fruit
```

Out[6]:
```
0     apple
1    orange
2     apple
3     apple
4    banana
5     apple
6    orange
7     apple
8     apple
9    banana
dtype: object
```

In [7]:
```
pd.unique(fruit)
```

Out[7]:
```
array(['apple', 'orange', 'banana'], dtype=object)
```

In [8]:
```
pd.value_counts(fruit)
```

Out[8]:
```
apple     6
banana    2
orange    2
dtype: int64
```

In [6]:
```
value=pd.Series([0,1,0,3,0]*2)
```

In [7]:
```
dim=pd.Series(['apple','orange','banana','abcd'])
```

In [67]:
```
dim.take(value)
```

Out[67]:
```
0     apple
1    orange
0     apple
3      abcd
0     apple
0     apple
1    orange
0     apple
3      abcd
0     apple
dtype: object
```

In [60]:
```
n=len(fruit)
```

In [10]:
```
n
```

Out[10]:
```
8
```

In [68]:
```
df=pd.DataFrame({'fruit':fruit,'b_id':np.arange(n),'count':np.random.randint(3,15,
```

In [69]:
```
df
```

Out[69]:

	b_id	fruit	count	weight
0	0	apple	10	0.737974
1	1	orange	7	2.817180
2	2	apple	9	2.406291
3	3	apple	9	2.528047
4	4	banana	13	0.372604
5	5	apple	8	3.338335
6	6	orange	4	1.730122
7	7	apple	12	1.088215
8	8	apple	5	3.235450
9	9	banana	6	2.921038

In [70]:
```
fruit_cat=df['fruit'].astype('category')
```

In [71]:
```
fruit_cat
```

Out[71]:
```
0     apple
1    orange
2     apple
3     apple
4    banana
5     apple
6    orange
7     apple
8     apple
9    banana
Name: fruit, dtype: category
Categories (3, object): [apple, banana, orange]
```

In [72]:
```
c=fruit_cat.values
```

In [64]:
```
type(c)
```

Out[64]:
```
pandas.core.arrays.categorical.Categorical
```

In [73]:
```
c.categories
```

Out[73]:
```
Index(['apple', 'banana', 'orange'], dtype='object')
```

In [74]:
```
c.codes
```

Out[74]:
```
array([0, 2, 0, 0, 1, 0, 2, 0, 0, 1], dtype=int8)
```

covert dataframe column to categorical

In [75]:
```
df['fruit']=df['fruit'].astype('category')
```

In [76]:
```
df.fruit
```

Out[76]:
```
0     apple
1    orange
2     apple
3     apple
4    banana
5     apple
6    orange
7     apple
8     apple
9    banana
Name: fruit, dtype: category
Categories (3, object): [apple, banana, orange]
```

In [21]:
```
my_categories=pd.Categorical(['foo','bar','baz','foo','bar'])
```

In [22]:
```
my_categories
```

Out[22]:
```
[foo, bar, baz, foo, bar]
Categories (3, object): [bar, baz, foo]
```

In [23]:
```
catgo=['foo','bar','baz']
```

In [24]:
```
code=[0,1,2,0,0,1]
```

In [25]:
```
catego1=pd.Categorical.from_codes(code,catgo)
```

In [26]:
```
catego1
```

Out[26]:
```
[foo, bar, baz, foo, foo, bar]
Categories (3, object): [foo, bar, baz]
```

In [27]:
```
ordered_catgo=pd.Categorical.from_codes(code,catgo,ordered=True)
```

In [28]:
```
ordered_catgo
```

Out[28]:
```
[foo, bar, baz, foo, foo, bar]
Categories (3, object): [foo < bar < baz]
```

In [29]:
```
catego1
```

Out[29]:
```
[foo, bar, baz, foo, foo, bar]
Categories (3, object): [foo, bar, baz]
```

In [30]:
```
catego1.as_ordered()  #adding ordering
```

Out[30]:
```
[foo, bar, baz, foo, foo, bar]
Categories (3, object): [foo < bar < baz]
```

performance with categoricals

In [32]:
```
n=10000
```

In [33]:
```
dra=pd.Series(np.random.randn(n))
```

In [34]:
```
dra.size
```

Out[34]:
```
10000
```

In [35]:
```
label=pd.Series(['foo','bar','baz','qux']*(n//4))
```

In [36]:
```
catgo=label.astype('category')
```

In [37]:
```
label.memory_usage()
```

Out[37]:
```
80128
```

In [38]:
```
catgo.memory_usage()
```

Out[38]:
```
10320
```

In [41]:
```
%time _ =label.astype('category')
```

```
Wall time: 2.99 ms
```

In [42]:
```
t=pd.Series(['a','b','c','d']*2)
```

In [43]:
```
cat_t=t.astype('category')
```

In [44]:
```
cat_t
```

Out[44]:
```
0    a
1    b
2    c
3    d
4    a
5    b
6    c
7    d
dtype: category
Categories (4, object): [a, b, c, d]
```

In [45]:
```
cat_t.cat.codes #cat method to provide access to categorical methods
```

Out[45]:
```
0    0
1    1
2    2
3    3
4    0
5    1
6    2
7    3
dtype: int8
```

In [46]:
```
cat_t.cat.categories
```

Out[46]:
```
Index(['a', 'b', 'c', 'd'], dtype='object')
```

In [47]:
```
actual_cat = ['a', 'b', 'c', 'd', 'e']
```

In [48]:
```
cat_t1=cat_t.cat.set_categories(actual_cat)
```

In [49]:
```
cat_t1
```

Out[49]:
```
0    a
1    b
2    c
3    d
4    a
5    b
6    c
7    d
dtype: category
Categories (5, object): [a, b, c, d, e]
```

In [50]:
```
cat_t.value_counts()
```

Out[50]:
```
d    2
c    2
b    2
a    2
dtype: int64
```

In [51]:
```
cat_t1.value_counts()
```

Out[51]:
```
d    2
c    2
b    2
a    2
e    0
dtype: int64
```

In [52]:
```
cat_t3 = cat_t[cat_t.isin(['a', 'b'])]
```

In [53]:
```
cat_t3
```

Out[53]:
```
0    a
1    b
4    a
5    b
dtype: category
Categories (4, object): [a, b, c, d]
```

In [54]:
```
cat_t3.cat.remove_unused_categories()
```

Out[54]:
```
0    a
1    b
4    a
5    b
dtype: category
Categories (2, object): [a, b]
```

In [55]:
```
cat_s = pd.Series(['a', 'b', 'c', 'd'] * 2, dtype='category')
```

In [56]:
```
pd.get_dummies(cat_s)
```

Out[56]:

	a	b	c	d
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0
5	0	1	0	0
6	0	0	1	0
7	0	0	0	1

group by

In [86]:
```
df=pd.DataFrame({'key':['a','b','d']*4,'value':np.arange(12)})
```

In [78]:
```
df
```

Out[78]:

	key	value
0	a	0
1	b	1
2	c	2
3	d	3
4	a	4
5	b	5
6	c	6
7	d	7
8	a	8
9	b	9
10	c	10
11	d	11
12	a	12
13	b	13
14	c	14
15	d	15

In [87]:
```
grp=df.groupby('key').value
```

In [80]:
```
grp
```

Out[80]:
```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x0000024313AD02B0>
```

In [88]:
```
grp.mean()
```

Out[88]:
```
key
a    4.5
b    5.5
d    6.5
Name: value, dtype: float64
```

In [89]:
```
grp.transform(lambda x:x.mean())
```

Out[89]:
```
0     4.5
1     5.5
2     6.5
3     4.5
4     5.5
5     6.5
6     4.5
7     5.5
8     6.5
9     4.5
10    5.5
11    6.5
Name: value, dtype: float64
```

In [90]:
```
grp.transform('mean')
```

Out[90]:
```
0     4.5
1     5.5
2     6.5
3     4.5
4     5.5
5     6.5
6     4.5
7     5.5
8     6.5
9     4.5
10    5.5
11    6.5
Name: value, dtype: float64
```

In [84]:
```
grp.transform(lambda x:x*3)
```

Out[84]:
```
0      0
1      3
2      6
3      9
4     12
5     15
6     18
7     21
8     24
9     27
10    30
11    33
12    36
13    39
14    42
15    45
Name: value, dtype: int32
```

In [91]:
```
grp.transform(lambda x:x.rank(ascending=False))
```

Out[91]:
```
0     4
1     4
2     4
3     3
4     3
5     3
6     2
7     2
8     2
9     1
10    1
11    1
Name: value, dtype: int32
```

In [92]:
```
def normalize(x):
    return(x-x.mean())/x.std()
```

In [93]:
```
grp.transform(normalize)
```

Out[93]:
```
0    -1.161895
1    -1.161895
2    -1.161895
3    -0.387298
4    -0.387298
5    -0.387298
6     0.387298
7     0.387298
8     0.387298
9     1.161895
10    1.161895
11    1.161895
Name: value, dtype: float64
```

In [94]:
```
grp.apply(normalize)
```

Out[94]:
```
0    -1.161895
1    -1.161895
2    -1.161895
3    -0.387298
4    -0.387298
5    -0.387298
6     0.387298
7     0.387298
8     0.387298
9     1.161895
10    1.161895
11    1.161895
Name: value, dtype: float64
```

In [95]:
```
norm_df=(df['value']-grp.transform('mean'))/grp.transform('std')
```

In [96]:
```
norm_df
```

Out[96]:
```
0    -1.161895
1    -1.161895
2    -1.161895
3    -0.387298
4    -0.387298
5    -0.387298
6     0.387298
7     0.387298
8     0.387298
9     1.161895
10    1.161895
11    1.161895
Name: value, dtype: float64
```

method chaining

In [102…
```
df=pd.read_csv('Documents/ex1.csv')
```

In [103…
```
df
```

Out[103]:

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

In [105…
```
df1=df[df['b']>5]
```

In [107…
```
df1
```

Out[107]:

	a	b	c	d	message
1	5	6	7	8	world
2	9	10	11	12	foo

In [111…
```
df1['col1_chan'] = df1['a']- df1['a'].mean()
```

```
<ipython-input-111-102f91df6fbe>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['col1_chan'] = df1['a']- df1['a'].mean()
```

In [110…
```
df1['a']- df1['a'].mean()
```

Out[110]:
```
1   -2.0
2    2.0
Name: a, dtype: float64
```

In [112…
```
df1=df.copy()
```

In [ ]:
```

```

In [113…
```
df1['a1']=df['a'].mean()
```

In [114…]
```
df1
```

Out[114]:

	a	b	c	d	message	a1
0	1	2	3	4	hello	5.0
1	5	6	7	8	world	5.0
2	9	10	11	12	foo	5.0

In [118…]
```
df1=df.assign('a1'= df['a'].mean())
```

```
  File "<ipython-input-118-37491e28d86f>", line 1
    df1=df.assign('a1'= df['a'].mean())
                  ^
SyntaxError: expression cannot contain assignment, perhaps you meant "=="?
```

In [119…]
```
result=(pd.read_csv('Documents/ex1.csv')
        lambda x: x.'b'>5])
```

```
  File "<ipython-input-119-d5bc696220aa>", line 2
    lambda x: x.'b'>5])
    ^
SyntaxError: invalid syntax
```

In [127…]
```
import pandas as pd
bins=[0, 3,5, 7]
labels=['Short', 'medium', 'long']
res = (
  pd.read_csv('documents/iris.csv')
    .query('species == "setosa"')
    .assign(petal_length = lambda df: pd.cut(df['sepal_length'], bins=bins, labels=
)
res.head()
```

Out[127]:

	sepal_length	sepal_width	petal_length	petal_width	species	Unnamed: 5	Unnamed: 6	Unnamed 7
0	5.1	3.5	long	0.2	setosa	NaN	NaN	NaN
1	4.9	3.0	medium	0.2	setosa	NaN	NaN	NaN
2	4.7	3.2	medium	0.2	setosa	NaN	NaN	NaN
3	4.6	3.1	medium	0.2	setosa	NaN	NaN	NaN
4	5.0	3.6	medium	0.2	setosa	NaN	5.2	4.1

In [ ]:
```
(df.dropna(subset=['dep_time', 'unique_carrier'])
   .loc[df['unique_carrier']
       .isin(df['unique_carrier'].value_counts().index[:5])]
   .set_index('dep_time')
   # TimeGrouper to resample & groupby at once
   .groupby(['unique_carrier', pd.TimeGrouper("H")])
   .fl_num.count()
   .unstack(0)
   .fillna(0)
   .rolling(24)
   .sum()
   .rename_axis("Flights per Day", axis=1)
   .plot()
)
```

In [128…

```
pd.read_csv('documents/iris.csv')
```

Out[128]:

	sepal_length	sepal_width	petal_length	petal_width	species	Unnamed: 5	Unnamed: 6	Unnan
0	5.1	3.5	1.4	0.2	setosa	NaN	NaN	I
1	4.9	3.0	1.4	0.2	setosa	NaN	NaN	I
2	4.7	3.2	1.3	0.2	setosa	NaN	NaN	I
3	4.6	3.1	1.5	0.2	setosa	NaN	NaN	I
4	5.0	3.6	1.4	0.2	setosa	NaN	5.2	
...	...	...	...	...	...	...	...	
145	6.7	3.0	5.2	2.3	virginica	NaN	NaN	I
146	6.3	2.5	5.0	1.9	virginica	NaN	NaN	I
147	6.5	3.0	5.2	2.0	virginica	NaN	NaN	I
148	6.2	3.4	5.4	2.3	virginica	NaN	NaN	I
149	5.9	3.0	5.1	1.8	virginica	NaN	NaN	I

150 rows × 10 columns

In [ ]: