Categorical Data

In [86]:

```
import pandas as pd
import numpy as np
```

In [87]:

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

In [88]:

```
fruit
```

Out[88]:

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

In [91]:

```
pd.unique(fruit)
```

Out[91]:

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

In [92]:

```
pd.value_counts(fruit)
```

Out[92]:

```
apple     6
orange    2
banana    2
dtype: int64
```

In [95]:

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

In [96]:

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

In [97]:

```
dim.take(value)
```

Out[97]:

```
0     apple
1    orange
0     apple
3      abcd
0     apple
0     apple
1    orange
0     apple
3      abcd
0     apple
dtype: object
```

In [100...

```
n=len(fruit)
```

In [101...

```
n
```

Out[101]:

```
10
```

In [102...

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

In [12]:

```
df
```

Out[12]:

	b_id	fruit	count	weight
0	0	apple	4	2.132555
1	1	orange	12	2.131443
2	2	apple	6	2.836932
3	3	apple	6	0.087985
4	4	banana	3	3.649417
5	5	apple	13	1.285766
6	6	orange	12	3.707616
7	7	apple	11	1.552540
8	8	apple	9	3.865476
9	9	banana	10	2.270199

In [103...

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

In [104...

```
fruit_cat
```

Out[104]:

```
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 [106...

```
c=fruit_cat.values
```

In [107...

```
type(c)
```

Out[107]:

```
pandas.core.arrays.categorical.Categorical
```

In [108...

```
c.categories
```

Out[108]:

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

In [109...

```
c.codes
```

Out[109]:

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

covert dataframe column to categorical

In [110...

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

In [111...

```
df.fruit
```

```
Out[111]: 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 [113…  my_categories=pd.Categorical(['foo','bar','baz','foo','bar'])
```

```
In [114…  my_categories
```

```
Out[114]: ['foo', 'bar', 'baz', 'foo', 'bar']
          Categories (3, object): ['bar', 'baz', 'foo']
```

```
In [115…  catgo=['foo','bar','baz']
```

```
In [116…  code=[0,1,2,0,0,1]
```

```
In [117…  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 [118…  catego1
```

```
Out[118]: ['foo', 'bar', 'baz', 'foo', 'foo', 'bar']
          Categories (3, object): ['foo', 'bar', 'baz']
```

```
In [119…  catego1.as_ordered()  #adding ordering
```

```
Out[119]: ['foo', 'bar', 'baz', 'foo', 'foo', 'bar']
          Categories (3, object): ['foo' < 'bar' < 'baz']
```

performance with categoricals

```
In [120…  n=10000
```

```
In [121…  dra=pd.Series(np.random.randn(n))
```

```
In [122…  dra.size
```

```
Out[122]: 10000
```

```
In [123…  label=pd.Series(['foo','bar','baz','qux']*(n//4))
In [124…  catgo=label.astype('category')
```

```
In [125…  label.memory_usage()
```

```
Out[125]: 80128
```

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

```
Out[37]:  10332
```

```
In [38]:  %time _ =label.astype('category')
```

```
CPU times: total: 0 ns
Wall time: 1.97 ms
```

```
In [39]:  t=pd.Series(['a','b','c','d']*2)
```

```
In [40]:  cat_t=t.astype('category')
```

```
In [41]:  cat_t
```

```
Out[41]:  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 [42]:  cat_t.cat.codes #cat method to provide access to categorical methods
```

```
Out[42]:  0    0
          1    1
          2    2
          3    3
          4    0
          5    1
          6    2
          7    3
          dtype: int8
```

```
In [43]:  cat_t.cat.categories
```

```
Out[43]:  Index(['a', 'b', 'c', 'd'], dtype='object')
```

```
In [44]:  actual_cat = ['a', 'b', 'c', 'd', 'e']
```

```
In [45]:  cat_t1=cat_t.cat.set_categories(actual_cat)
```

```
In [46]:  cat_t1
```

```
Out[46]: 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 [47]: cat_t.value_counts()
```

```
Out[47]: a    2
         b    2
         c    2
         d    2
         dtype: int64
```

```
In [48]: cat_t1.value_counts()
```

```
Out[48]: a    2
         b    2
         c    2
         d    2
         e    0
         dtype: int64
```

```
In [49]: cat_t3 = cat_t[cat_t.isin(['a', 'b'])]
```

```
In [50]: cat_t3
```

```
Out[50]: 0    a
         1    b
         4    a
         5    b
         dtype: category
         Categories (4, object): ['a', 'b', 'c', 'd']
```

```
In [51]: cat_t3.cat.remove_unused_categories()
```

```
Out[51]: 0    a
         1    b
         4    a
         5    b
         dtype: category
         Categories (2, object): ['a', 'b']
```

```
In [52]: cat_s = pd.Series(['a', 'b', 'c', 'd'] * 2, dtype='category')
```

```
In [126…: pd.get_dummies(cat_s)
```

Out[126]:

	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 [127...

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

In [128...

```
df
```

Out[128]:

	key	value
0	a	0
1	b	1
2	d	2
3	a	3
4	b	4
5	d	5
6	a	6
7	b	7
8	d	8
9	a	9
10	b	10
11	d	11

In [129...

```
grp=df.groupby('key').value
```

In [130...

```
grp
```

Out[130]:

```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x00000283540042E0>
```

In [131...

```
grp.mean()
```

Out[131]:

```
key
a    4.5
b    5.5
d    6.5
Name: value, dtype: float64
```

In [132…

```
grp.transform(lambda x:x.mean())
```

Out[132]:

```
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 [133…

```
grp.transform('mean')
```

Out[133]:

```
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 [134…

```
grp.transform(lambda x:x*3)
```

Out[134]:

```
0      0
1      3
2      6
3      9
4     12
5     15
6     18
7     21
8     24
9     27
10    30
11    33
Name: value, dtype: int32
```

In [135…

```
grp.transform(lambda x:x.rank(ascending=False))
```

Out[135]:

```
0     4.0
1     4.0
2     4.0
3     3.0
4     3.0
5     3.0
6     2.0
7     2.0
8     2.0
9     1.0
10    1.0
11    1.0
Name: value, dtype: float64
```

In [136…

```
def normalize(x):
    return(x-x.mean())/x.std()
```

In [64]:
```
grp.transform(normalize)
```

Out[64]:
```
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 [65]:
```
grp.apply(normalize)
```

Out[65]:
```
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 [66]:
```
norm_df=(df['value']-grp.transform('mean'))/grp.transform('std')
```

In [67]:
```
norm_df
```

Out[67]:
```
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 [69]:
```
df=pd.read_csv('ex1.csv')
```

In [70]:
```
df
```

Out[70]:

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

In [71]:

```
df1=df[df['b']>5]
```

In [72]:

```
df1
```

Out[72]:

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

In [73]:

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

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_23076\3268691799.py:1: SettingWithCop
yWarning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['col1_chan'] = df1['a']- df1['a'].mean()
```

In [74]:

```
df1['a']- df1['a'].mean()
```

Out[74]:

```
1   -2.0
2    2.0
Name: a, dtype: float64
```

In [75]:

```
df1=df.copy()
```

In [ ]:

In [76]:

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

In [77]:

```
df1
```

Out[77]:

	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 [78]:

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

```
  Input In [78]
    df1=df.assign('a1'= df['a'].mean())
                  ^
SyntaxError: expression cannot contain assignment, perhaps you meant "=="?
```

In [82]:

```
result=(pd.read_csv('ex1.csv'), lambda x: x.'b'>5])
```

```
  Input In [82]
    result=(pd.read_csv('ex1.csv'), lambda x: x.'b'>5])
                                                ^
SyntaxError: invalid syntax
```

In [84]:

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

Out[84]:

sepal_length	sepal_width	petal_length	petal_width	species

In [85]:

```python
(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()
)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [85], in <cell line: 1>()
----> 1 (df.dropna(subset=['dep_time', 'unique_carrier'])
      2    .loc[df['unique_carrier']
      3        .isin(df['unique_carrier'].value_counts().index[:5])]
      4    .set_index('dep_time')
      5    # TimeGrouper to resample & groupby at once
      6    .groupby(['unique_carrier', pd.TimeGrouper("H")])
      7    .fl_num.count()
      8    .unstack(0)
      9    .fillna(0)
     10    .rolling(24)
     11    .sum()
     12    .rename_axis("Flights per Day", axis=1)
     13    .plot()
     14 )

File ~\anaconda3\lib\site-packages\pandas\util\_decorators.py:311, in deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:6002, in DataFrame.dropna(self, axis, how, thresh, subset, inplace)
   6000     check = indices == -1
   6001     if check.any():
-> 6002         raise KeyError(np.array(subset)[check].tolist())
   6003     agg_obj = self.take(indices, axis=agg_axis)
   6005 if thresh is not None:

KeyError: ['dep_time', 'unique_carrier']
```

```
In [ ]: pd.read_csv('documents/iris.csv')
```

```
In [ ]:
```