Lab-4: Analyzing the data cleaning.

- Generate similar (not same) code cells as given below.
- You can use the same or similar logic for the cells.
- You can group multiple cells into a single one if they fall under the same logic.
- Explain the main operation of each or multiple (same logic) cells in just one line.

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

```
string_data = pd.Series(['aardvark', 'artichoke', np.nan, 'avocado'])
print (string_data)
print (string_data.isnull())
```

```
string_data[0] = None
string_data.isnull()
```

```
from numpy import nan as NA
data = pd.Series([1, NA, 3.5, NA, 7])
data.dropna()
```

```
data[data.notnull()]
```

```
data = pd.DataFrame([[1., 6.5, 3.], [1., NA, NA],
                     [NA, NA, NA], [NA, 6.5, 3.]])
cleaned = data.dropna()
print (data)
print (cleaned)
```

```
data.dropna(how='all')
```

```
data[4] = NA
print (data)
print (data.dropna(axis=1, how='all'))
```

```
df = pd.DataFrame(np.random.randn(7, 3))
df.iloc[:4, 1] = NA
```

```
df.iloc[:2, 2] = NA
print (df)
print (df.dropna())
print (df.dropna(thresh=2))



df.fillna(0)



df.fillna({1: 0.5, 2: 0})



df = pd.DataFrame(np.random.randn(6, 3))
df.iloc[2:, 1] = NA
df.iloc[4:, 2] = NA
print (df)
print (df.fillna(method='ffill'))
print (df.fillna(method='ffill', limit=2))



data = pd.Series([1., NA, 3.5, NA, 7])
data.fillna(data.mean())



data = pd.DataFrame({'k1': ['one', 'two'] * 3 + ['two'],
                     'k2': [1, 1, 2, 3, 3, 4, 4]})



data.duplicated()



data.drop_duplicates()



data['v1'] = range(7)
data.drop_duplicates(['k1'])



data = pd.DataFrame({'food': ['bacon', 'pulled pork', 'bacon',
                              'Pastrami', 'corned beef', 'Bacon',
                              'pastrami', 'honey ham', 'nova lox'],
                     'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6]})
```

```
meat_to_animal = {
  'bacon': 'pig',
  'pulled pork': 'pig',
  'pastrami': 'cow',
  'corned beef': 'cow',
  'honey ham': 'pig',
  'nova lox': 'salmon'
}


lowercased = data['food'].str.lower()
print (lowercased)
data['animal'] = lowercased.map(meat_to_animal)
print (data)


data['food'].map(lambda x: meat_to_animal[x.lower()])


data = pd.Series([1., -999., 2., -999., -1000., 3.])


data.replace(-999, np.nan)


data.replace([-999, -1000], np.nan)


data.replace([-999, -1000], [np.nan, 0])


data.replace({-999: np.nan, -1000: 0})


data = pd.DataFrame(np.arange(12).reshape((3, 4)),
                    index=['Ohio', 'Colorado', 'New York'],
                    columns=['one', 'two', 'three', 'four'])


transform = lambda x: x[:4].upper()
data.index.map(transform)


data.index = data.index.map(transform)
```

```
data.rename(index=str.title, columns=str.upper)


data.rename(index={'OHIO': 'INDIANA'},
            columns={'three': 'peekaboo'})


data.rename(index={'OHIO': 'INDIANA'}, inplace=True)


ages = [20, 22, 25, 27, 21, 23, 37, 31, 61, 45, 41, 32]


bins = [18, 25, 35, 60, 100]
cats = pd.cut(ages, bins)


print (cats.codes)
print (cats.categories)
print (pd.value_counts(cats))


pd.cut(ages, [18, 26, 36, 61, 100], right=False)


group_names = ['Youth', 'YoungAdult', 'MiddleAged', 'Senior']
pd.cut(ages, bins, labels=group_names)


data = np.random.rand(20)
pd.cut(data, 4, precision=2)


data = np.random.randn(1000)
cats = pd.qcut(data, 4)
pd.value_counts(cats)


pd.qcut(data, [0, 0.1, 0.5, 0.9, 1.])
```

```
data = pd.DataFrame(np.random.randn(1000, 4))
data.describe()
```

```
col = data[2]
col[np.abs(col) > 3]
```

```
data[(np.abs(data) > 3).any(1)]
```

```
np.sign(data).head()
```

```
df = pd.DataFrame(np.arange(5 * 4).reshape((5, 4)))
sampler = np.random.permutation(5)
```

```
df.take(sampler)
```

```
df.sample(n=3)
```

```
choices = pd.Series([5, 7, -1, 6, 4])
draws = choices.sample(n=10, replace=True)
```

```
val = 'a,b,  guido'
val.split(',')
```

```
pieces = [x.strip() for x in val.split(',')]
```

```
first, second, third = pieces
first + '::' + second + '::' + third
```

```
'::'.join(pieces)
```

```
'guido' in val
val.index(',')
val.find(':')


val.count(',')


val.replace(',', '::')
val.replace(',', '')


import re
text = "foo    bar\t baz  \tqux"
re.split('\s+', text)


regex = re.compile('\s+')
regex.split(text)


regex.findall(text)


text = """Dave dave@google.com
Steve steve@gmail.com
Rob rob@gmail.com
Ryan ryan@yahoo.com
"""
pattern = r'[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}'

# re.IGNORECASE makes the regex case-insensitive
regex = re.compile(pattern, flags=re.IGNORECASE)


regex.findall(text)


m = regex.search(text)
text[m.start():m.end()]


print(regex.match(text))
```

```
print(regex.sub('REDACTED', text))
```

```
pattern = r'([A-Z0-9._%+-]+)@([A-Z0-9.-]+)\.([A-Z]{2,4})'
regex = re.compile(pattern, flags=re.IGNORECASE)
```

```
m = regex.match('wesm@bright.net')
m.groups()
```

```
regex.findall(text)
```

```
print(regex.sub(r'Username: \1, Domain: \2, Suffix: \3', text))
```

```
data = {'Dave': 'dave@google.com', 'Steve': 'steve@gmail.com',
        'Rob': 'rob@gmail.com', 'Wes': np.nan}
data = pd.Series(data)
data
data.isnull()
```

```
data.str.contains('gmail')
```

```
data.str.findall(pattern, flags=re.IGNORECASE)
```

```
matches = data.str.match(pattern, flags=re.IGNORECASE)
```

```
data.str[:5]
```