Part_5_Miscellaneous_Operations

October 31, 2022

[1]:
```
import pandas as pd
import numpy as np
```

1 Concatenation:

- Combining data from two or more data frames
- If both sources are in same format, then a concatenation through pd.concat( ) in enough
- Pandas will automatically fills NaN where necessary

[2]:
```
data_one = {'A': ['A0', 'A1', 'A2', 'A3'], 'B': ['B0', 'B1', 'B2', 'B3']}
data_two = {'C': ['C0', 'C1', 'C2', 'C3'], 'D': ['D0', 'D1', 'D2', 'D3']}

df_one = pd.DataFrame(data_one)
df_two = pd.DataFrame(data_two)
df_one
```

[2]:
```
    A   B
0  A0  B0
1  A1  B1
2  A2  B2
3  A3  B3
```

[9]:
```
df_two
```

[9]:
```
    A   B
0  C0  D0
1  C1  D1
2  C2  D2
3  C3  D3
```

[10]:
```
# concatenation along columns

concatenated_dfs = pd.concat([df_one, df_two], axis = 1)
concatenated_dfs
```

```
[10]:     A   B   A   B
      0  A0  B0  C0  D0
      1  A1  B1  C1  D1
      2  A2  B2  C2  D2
      3  A3  B3  C3  D3
```

```
[11]: # concatenation along rows

      concatenated_dfs = pd.concat([df_one, df_two])
      concatenated_dfs
```

```
[11]:     A   B
      0  A0  B0
      1  A1  B1
      2  A2  B2
      3  A3  B3
      0  C0  D0
      1  C1  D1
      2  C2  D2
      3  C3  D3
```

```
[12]: # to handle concat along rows

      df_two.columns = df_one.columns
      print(df_two.columns)
```

```
      Index(['A', 'B'], dtype='object')
```

```
[14]: new_concat = pd.concat([df_two, df_one])
      new_concat
```

```
[14]:     A   B
      0  C0  D0
      1  C1  D1
      2  C2  D2
      3  C3  D3
      0  A0  B0
      1  A1  B1
      2  A2  B2
      3  A3  B3
```

```
[16]: new_concat.index = range(len(new_concat))
      new_concat
```

```
[16]:     A   B
      0  C0  D0
      1  C1  D1
```

```
2  C2  D2
3  C3  D3
4  A0  B0
5  A1  B1
6  A2  B2
7  A3  B3
```

2 Merge:

[17]:
```
register = {'Reg_id': [1,2,3,4], 'Name': ['Andrew', 'Bob', 'Charlie', 'David']}
logins = {'log_id': [1,2,3,4], 'Name': ['Xavier', 'Andrew', 'Yolanda', 'Bob']}
```

[63]:
```
registerations = pd.DataFrame(register)
registerations
```

[63]:
```
   Reg_id     Name
0       1   Andrew
1       2      Bob
2       3  Charlie
3       4    David
```

[22]:
```
log_in = pd.DataFrame(logins)
log_in
```

[22]:
```
   log_id     Name
0       1   Xavier
1       2   Andrew
2       3  Yolanda
3       4      Bob
```

[36]:
```
pd.merge(registerations, log_in, how='inner', on = 'Name')
```

[36]:
```
   Reg_id    Name  log_id
0       1  Andrew       2
1       2     Bob       4
```

[58]:
```
registerations = registerations.set_index('Name')
```

[50]:
```
registerations
```

[50]:
```
         Reg_id
Name
Andrew        1
Bob           2
Charlie       3
```

```
David         4
```

[51]:

```
pd.merge(registerations, log_in, how='inner', left_index = True, right_on =␣
 ↪'Name')
```

[51]:

```
   Reg_id  log_id    Name
1       1       2  Andrew
3       2       4     Bob
```

[37]:

```
pd.merge(registerations, log_in, how='left', on = 'Name')
```

[37]:

```
   Reg_id     Name  log_id
0       1   Andrew     2.0
1       2      Bob     4.0
2       3  Charlie     NaN
3       4    David     NaN
```

[39]:

```
pd.merge(registerations, log_in, how='outer', on = 'Name')
```

[39]:

```
   Reg_id     Name  log_id
0     1.0   Andrew     2.0
1     2.0      Bob     4.0
2     3.0  Charlie     NaN
3     4.0    David     NaN
4     NaN   Xavier     1.0
5     NaN  Yolanda     3.0
```

[59]:

```
registerations.reset_index(inplace = True)
```

[60]:

```
registerations
```

[60]:

```
      Name  Reg_id
0   Andrew       1
1      Bob       2
2  Charlie       3
3    David       4
```

[64]:

```
registerations.columns = ['ID', 'Name']
log_in.columns = ['ID', 'Name']
```

[65]:

```
registerations
```

[65]:

```
   ID     Name
0   1   Andrew
1   2      Bob
2   3  Charlie
3   4    David
```

```
[66]: log_in
```

```
[66]:    ID     Name
      0   1   Xavier
      1   2   Andrew
      2   3  Yolanda
      3   4      Bob
```

```
[67]: pd.merge(registerations, log_in, how='inner', on = 'Name')
```

```
[67]:    ID_x     Name  ID_y
      0     1   Andrew     2
      1     2      Bob     4
```

```
[68]: pd.merge(registerations, log_in, how='inner', on = 'Name', suffixes = ('-reg',␣
       ↪'-log'))
```

```
[68]:    ID-reg     Name  ID-log
      0       1   Andrew       2
      1       2      Bob       4
```

3 Text Methods for String data:

```
[69]: email = 'aliza@gmail.com'
      email.split('@')
```

```
[69]: ['aliza', 'gmail.com']
```

```
[73]: email.isdigit()
```

```
[73]: False
```

```
[75]: '5'.isdigit()
```

```
[75]: True
```

```
[70]: names = pd.Series(['Andrew', 'Bob', '4'])
```

```
[71]: names
```

```
[71]: 0    Andrew
      1       Bob
      2         4
      dtype: object
```

[72]:
```
names.str.upper()
```

[72]:
```
0    ANDREW
1       BOB
2         4
dtype: object
```

[78]:
```
names.str.isdigit()
```

[78]:
```
0    False
1    False
2     True
dtype: bool
```

[84]:
```
tech_com = ['Google,Apple,AMAZON', 'JPM,BAC,GS']
```

[85]:
```
tech_ser = pd.Series(tech_com)
```

[86]:
```
tech_ser
```

[86]:
```
0    Google,Apple,AMAZON
1             JPM,BAC,GS
dtype: object
```

[87]:
```
tech_ser.str.split(',', expand = True)
```

[87]:
```
        0      1       2
0  Google  Apple  AMAZON
1     JPM    BAC      GS
```

[89]:
```
messy_names = pd.Series(['Andrew    ', 'bob;bob', "   claire"])
messy_names
```

[89]:
```
0    Andrew    
1       bob;bob
2        claire
dtype: object
```

[94]:
```
temp1 = messy_names.str.replace(';', ' ')
```

[96]:
```
temp2 = temp1.str.strip()
temp2
```

[96]:
```
0     Andrew
1    bob bob
2     claire
dtype: object
```

[98]:

```python
temp3 = temp2.str.capitalize()
temp3
```

[98]:

```
0      Andrew
1     Bob bob
2      Claire
dtype: object
```

[107]:

```python
def handle_func(names):
  return names.replace(';', ' ').strip().capitalize()
```

[108]:

```python
messy_names.apply(handle_func)
```

[108]:

```
0      Andrew
1     Bob bob
2      Claire
dtype: object
```

[111]:

```python
import timeit

setup = '''
import pandas as pd
import numpy as np


messy_names = pd.Series(['Andrew    ', 'bob;bob', "   claire"])


def handle_func(names):
  return names.replace(';', ' ').strip().capitalize()

'''

stmt_one = '''
messy_names.str.replace(';', ' ').str.strip().str.capitalize()
'''

stmt_two = '''
messy_names.apply(handle_func)
'''

stmt_three = '''

np.vectorize(handle_func)(messy_names)
'''

timeit.timeit(setup=setup, stmt = stmt_one, number=1000)
```

```
[111]: 1.4535369449999962
```

```
[112]: timeit.timeit(setup=setup, stmt = stmt_two, number=1000)
```

```
[112]: 0.23004439899978024
```

```
[113]: timeit.timeit(setup=setup, stmt = stmt_three, number=1000)
```

```
[113]: 0.06301108899970131
```

4 Time Methods for Date & Time Data:

```
[114]: from datetime import datetime
```

```
[116]: myyear = 1995
        mymonth = 3
        myday = 28
        myhour = 12
        myminutes = 5
        myseconds = 00
```

```
[118]: mybrday = datetime(myyear, mymonth, myday, myhour, myminutes, myseconds)
```

```
[119]: mybrday.year
```

```
[119]: 1995
```

```
[122]: mybrday.minute
```

```
[122]: 5
```

```
[123]: myser = pd.Series(['NOV 3, 1990', '2000-01-01', None])
        myser
```

```
[123]: 0    NOV 3, 1990
        1     2000-01-01
        2           None
        dtype: object
```

```
[126]: timeseries = pd.to_datetime(myser)
```

```
[127]: timeseries[0]
```

```
[127]: Timestamp('1990-11-03 00:00:00')
```

[128]:

```
timeseries[0].year
```

[128]:

```
1990
```

[130]:

```
my_euro_date = '31-10-2022'
pd.to_datetime(my_euro_date)
```

[130]:

```
Timestamp('2022-10-31 00:00:00')
```

[132]:

```
euro_date = '10-12-2022'
pd.to_datetime(euro_date)
```

[132]:

```
Timestamp('2022-10-12 00:00:00')
```

[133]:

```
pd.to_datetime(euro_date, dayfirst = True)
```

[133]:

```
Timestamp('2022-12-10 00:00:00')
```

[137]:

```
stylish_date = '12 -- December -- 2022'
pd.to_datetime(stylish_date, format = '%d -- %B -- %Y')
```

[137]:

```
Timestamp('2022-12-12 00:00:00')
```

[138]:

```
custom_date = '12th of December 2022'
pd.to_datetime(custom_date)
```

[138]:

```
Timestamp('2022-12-12 00:00:00')
```

[147]:

```
mydata = {'DATE': ['2020-01-01', '2021-02-03', '2023-02-08'], 'MRT':␣
 ↪[123,124,567]}
```

[161]:

```
myser = pd.DataFrame(mydata)
myser
```

[161]:

```
         DATE  MRT
0  2020-01-01  123
1  2021-02-03  124
2  2023-02-08  567
```

[162]:

```
myser['DATE']
```

[162]:

```
0    2020-01-01
1    2021-02-03
2    2023-02-08
Name: DATE, dtype: object
```

[163]:

```
myser['DATE'] = pd.to_datetime(myser['DATE'])
```

```
[164]: myser['DATE']
```

```
[164]: 0   2020-01-01
       1   2021-02-03
       2   2023-02-08
       Name: DATE, dtype: datetime64[ns]
```

```
[ ]: myser = pd.read_csv(mydata, parse_dates = [0])
```

```
[165]: myser['DATE'].dt.year
```

```
[165]: 0    2020
       1    2021
       2    2023
       Name: DATE, dtype: int64
```

```
[166]: myser['DATE'].dt.is_leap_year
```

```
[166]: 0     True
       1    False
       2    False
       Name: DATE, dtype: bool
```

```
[155]: myser = myser.set_index('DATE')
```

```
[156]: myser
```

```
[156]:             MRT
       DATE
       2020-01-01  123
       2021-02-03  124
       2023-02-08  567
```

```
[157]: myser.resample(rule = 'A')
```

```
[157]: <pandas.core.resample.DatetimeIndexResampler object at 0x7f532ca5e890>
```

```
[158]: myser.resample(rule = 'A').mean()
```

```
[158]:               MRT
       DATE
       2020-12-31  123.0
       2021-12-31  124.0
       2022-12-31    NaN
       2023-12-31  567.0
```

```
[159]: myser.resample(rule = 'B').mean()
```

```
[159]:                MRT
       DATE
       2020-01-01  123.0
       2020-01-02    NaN
       2020-01-03    NaN
       2020-01-06    NaN
       2020-01-07    NaN
       …             …
       2023-02-02    NaN
       2023-02-03    NaN
       2023-02-06    NaN
       2023-02-07    NaN
       2023-02-08  567.0

       [811 rows x 1 columns]
```

5 Input & Output - CSV Files:

```
[167]: pwd
```

```
[167]: '/content'
```

```
[168]: import os
       os.getcwd()
```

```
[168]: '/content'
```

```
[169]: df = pd.read_csv('/content/sample_data/example.csv')
```

```
[170]: df.head()
```

```
[170]:     a   b   c   d
       0   0   1   2   3
       1   4   5   6   7
       2   8   9  10  11
       3  12  13  14  15
```

```
[171]: df = pd.read_csv('/content/sample_data/example.csv', header = None)
       df
```

```
[171]:     0   1   2   3
       0   a   b   c   d
       1   0   1   2   3
       2   4   5   6   7
       3   8   9  10  11
```

```
4  12  13  14  15
```

```
[173]: df = pd.read_csv('/content/sample_data/example.csv', index_col = 0)
       df
```

```
[173]:      b   c   d
       a
       0    1   2   3
       4    5   6   7
       8    9  10  11
       12  13  14  15
```

```
[174]: df.to_csv('new_file.csv', index = True)
```

```
[175]: new_df = pd.read_csv('/content/new_file.csv')
       new_df
```

```
[175]:     a   b   c   d
       0   0   1   2   3
       1   4   5   6   7
       2   8   9  10  11
       3  12  13  14  15
```

6 Input & Output - HTML Tables:

```
[176]: !pip install lxml
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages
(4.9.1)
```

```
[177]: url = "https://en.wikipedia.org/wiki/World_population"
       tables = pd.read_html(url)
```

```
[179]: len(tables)
```

```
[179]: 25
```

```
[180]: tables[0]
```

```
[180]:   World population milestones in billions[3] (Worldometers estimates)        \
                                                                  Population     1
       0                                                Year                   1804
       1                                       Years elapsed                      -
```

```
      2     3     4     5     6     7     8     9    10
0  1927  1960  1974  1987  1999  2011  2022  2037  2057
1   123    33    14    13    12    12    11    15    20
```

```
[183]: tables[0].columns
```

```
[183]: MultiIndex([('World population milestones in billions[3] (Worldometers
estimates)', …),
            ('World population milestones in billions[3] (Worldometers
estimates)', …),
            ('World population milestones in billions[3] (Worldometers
estimates)', …),
            ('World population milestones in billions[3] (Worldometers
estimates)', …),
            ('World population milestones in billions[3] (Worldometers
estimates)', …),
            ('World population milestones in billions[3] (Worldometers
estimates)', …),
            ('World population milestones in billions[3] (Worldometers
estimates)', …),
            ('World population milestones in billions[3] (Worldometers
estimates)', …),
            ('World population milestones in billions[3] (Worldometers
estimates)', …),
            ('World population milestones in billions[3] (Worldometers
estimates)', …),
            ('World population milestones in billions[3] (Worldometers
estimates)', …)],
           )
```

```
[185]: tab = tables[0]
       tab['World population milestones in billions[3] (Worldometers estimates)']
```

```
[185]:      Population     1     2     3     4     5     6     7     8     9    10
       0          Year  1804  1927  1960  1974  1987  1999  2011  2022  2037  2057
       1 Years elapsed     -   123    33    14    13    12    12    11    15    20
```

```
[187]: tab = tab.drop(0, axis = 0)
```

```
[188]: tab
```

```
[188]:   World population milestones in billions[3] (Worldometers estimates)          \
                                                           Population  1    2
       1                                    Years elapsed               -  123
```

```
    3   4   5   6   7   8   9  10
1  33  14  13  12  12  11  15  20
```

```
[189]: tab.to_html('mytable.html', index=False)
```

7 Input & Output - Excel Files:

```
[3]: df = pd.read_excel('/content/sample_data/my_excel_file.xlsx', sheet_name =␣
      ↪'First_Sheet')
```

```
[8]: df
```

```
[8]:     a   b   c   d
     0   0   1   2   3
     1   4   5   6   7
     2   8   9  10  11
     3  12  13  14  15
```

```
[15]: new_df = pd.read_excel('/content/sample_data/my_excel_file.xlsx', sheet_name =␣
       ↪None)
      new_df
```

```
[15]: {'First_Sheet':      a   b   c   d
       0   0   1   2   3
       1   4   5   6   7
       2   8   9  10  11
       3  12  13  14  15}
```

```
[16]: new_df.keys()
```

```
[16]: dict_keys(['First_Sheet'])
```

```
[17]: new_df['First_Sheet']
```

```
[17]:     a   b   c   d
      0   0   1   2   3
      1   4   5   6   7
      2   8   9  10  11
      3  12  13  14  15
```

```
[5]: wb = pd.ExcelFile('/content/sample_data/my_excel_file.xlsx')
```

```
[10]: wb
```

```
[10]: <pandas.io.excel._base.ExcelFile at 0x7f97041b7b50>
```

```
[11]: wb.sheet_names
```

```
[11]: ['First_Sheet']
```

```
[13]: type(wb)
```

```
[13]: pandas.io.excel._base.ExcelFile
```

```
[24]: df.to_excel('mywork.xlsx', sheet_name = 'My_Sheet', index = False)
```

8 Input & Output - SQL Database:

```
[25]: from sqlalchemy import create_engine
```

```
[26]: temp_db = create_engine('sqlite:///:memory:')
```

```
[28]: df = pd.DataFrame(data = np.random.randint(0,100,(4,4)), columns = ['A', 'B',␣
       ↪'C', 'D'])
      df
```

```
[28]:     A   B   C   D
      0  92  61  70  22
      1  69   7  35  14
      2  25  33  96  46
      3  16  65  69  83
```

```
[34]: df.to_sql(name = 'My_table', con = temp_db)
```

```
[35]: pd.read_sql(sql = 'My_table', con = temp_db)
```

```
[35]:    index   A   B   C   D
      0      0  92  61  70  22
      1      1  69   7  35  14
      2      2  25  33  96  46
      3      3  16  65  69  83
```

```
[36]: pd.read_sql_query(sql = 'SELECT A, C FROM My_table', con = temp_db)
```

```
[36]:     A   C
      0  92  70
      1  69  35
      2  25  96
      3  16  69
```

9 Pandas Pivot Tables:

```
[38]: mydata = {'foo': ['one', 'one', 'one', 'two', 'two', 'two'], 'bar': ['A', 'B',␣
       ↪'C', 'A', 'B', 'C'], 'baz': [1,2,3,4,5,6], 'zoo': ['x', 'y', 'z', 'q', 'w',␣
       ↪'t']}

      df = pd.DataFrame(mydata)
      df
```

```
[38]:   foo bar  baz zoo
      0  one   A    1   x
      1  one   B    2   y
      2  one   C    3   z
      3  two   A    4   q
      4  two   B    5   w
      5  two   C    6   t
```

```
[39]: df.pivot(index = 'foo',
               columns = 'bar',
               values = 'baz')
```

```
[39]: bar  A  B  C
      foo
      one  1  2  3
      two  4  5  6
```

```
[43]: df = pd.read_csv('/content/sample_data/Sales_Funnel_CRM.csv')
      df.head()
```

```
[43]:   Account Number  Company      Contact Account Manager      Product  Licenses  \
      0        2123398   Google  Larry Pager    Edward Thorp    Analytics       150
      1        2123398   Google  Larry Pager    Edward Thorp   Prediction       150
      2        2123398   Google  Larry Pager    Edward Thorp     Tracking       300
      3        2192650     BOBO  Larry Pager    Edward Thorp    Analytics       150
      4         420496     IKEA    Elon Tusk    Edward Thorp    Analytics       300

         Sale Price        Status
      0     2100000     Presented
      1      700000     Presented
      2      350000  Under Review
      3     2450000          Lost
      4     4550000           Won
```

```
[46]: licenses = df[['Company', 'Product', 'Licenses']]
      licenses.head()
```

```
[46]:    Company     Product  Licenses
      0   Google   Analytics       150
      1   Google  Prediction       150
      2   Google    Tracking       300
      3     BOBO   Analytics       150
      4     IKEA   Analytics       300
```

```
[47]: pd.pivot(data = licenses, index = 'Company', columns = 'Product', values =␣
       ↪'Licenses')
```

```
[47]: Product        Analytics  GPS Positioning  Prediction  Tracking
      Company
       Google            150.0              NaN       150.0     300.0
      ATT                  NaN              NaN       150.0     150.0
      Apple              300.0              NaN         NaN       NaN
      BOBO               150.0              NaN         NaN       NaN
      CVS Health           NaN              NaN         NaN     450.0
      Cisco              300.0            300.0         NaN       NaN
      Exxon Mobile       150.0              NaN         NaN       NaN
      IKEA               300.0              NaN         NaN       NaN
      Microsoft            NaN              NaN         NaN     300.0
      Salesforce         750.0              NaN         NaN       NaN
      Tesla Inc.         300.0              NaN       150.0       NaN
      Walmart            150.0              NaN         NaN       NaN
```

```
[50]: pd.pivot_table(df, index = 'Company', aggfunc = 'sum')
```

```
[50]:               Account Number  Licenses  Sale Price
      Company
       Google             6370194       600     3150000
      ATT                 1396064       300     1050000
      Apple                405886       300     4550000
      BOBO                2192650       150     2450000
      CVS Health           902797       450      490000
      Cisco               4338998       600     4900000
      Exxon Mobile         470248       150     2100000
      IKEA                 420496       300     4550000
      Microsoft           1216870       300      350000
      Salesforce          2046943       750     7000000
      Tesla Inc.          1273370       450     3500000
      Walmart             2200450       150     2450000
```

```
[51]: df.groupby('Company').sum()
```

```
[51]:               Account Number  Licenses  Sale Price
      Company
       Google             6370194       600     3150000
```

```
ATT                    1396064        300      1050000
Apple                   405886        300      4550000
BOBO                   2192650        150      2450000
CVS Health              902797        450       490000
Cisco                  4338998        600      4900000
Exxon Mobile            470248        150      2100000
IKEA                    420496        300      4550000
Microsoft              1216870        300       350000
Salesforce             2046943        750      7000000
Tesla Inc.             1273370        450      3500000
Walmart                2200450        150      2450000
```

```
[52]: pd.pivot_table(df, index = 'Company', aggfunc = 'sum', values = ['Licenses',␣
       ↪'Sale Price'])
```

```
[52]:               Licenses  Sale Price
      Company
       Google             600     3150000
      ATT                 300     1050000
      Apple               300     4550000
      BOBO                150     2450000
      CVS Health          450      490000
      Cisco               600     4900000
      Exxon Mobile        150     2100000
      IKEA                300     4550000
      Microsoft           300      350000
      Salesforce          750     7000000
      Tesla Inc.          450     3500000
      Walmart             150     2450000
```

```
[55]: pd.pivot_table(df, index = ['Account Manager', 'Contact'], aggfunc = 'sum',␣
       ↪values = ['Licenses', 'Sale Price'], fill_value = 0, margins = True)
```

```
[55]:                              Licenses  Sale Price
      Account Manager Contact
      Claude Shannon  Cindy Phoner       750     7700000
                      Emma Gordian      1800    12390000
      Edward Thorp    Elon Tusk          750     8050000
                      Larry Pager        750     5600000
                      Will Grates        450     2800000
      All                               4500    36540000
```