datascience_project1_data_analysis

July 15, 2018

1 Datascience Project-1 - Data Analysis

1.0.1 Brief Description of Project

- In this project, URLs for few datasets (data-text.csv, berlin_weather_oldest.csv, users.csv, products.csv, session.csv, transactions.csv) are provided which will be used for analysis
- These datasets are first loaded using read_csv from the given URL
- Next these datasets are analyzed, processed including analyzing missing data to find answers for a set of given questions.
- Each question is answered by giving brief description of solution steps, code in python and output
- For analying mainly python packages pandas, numpy are mainly used

1.1 Load Given Dataset

1.2 Solution Steps:

- Import python packages numpy, pandas
- Load dataset data-text.csv from the URL provided using pandas read_csv method and assign to dataframe df
- Load dataset berlin_weather_oldest.csv from the URL provided using pandas read_csv method and assign to dataframe df1

```
In [187]: import numpy as np
          import pandas as pd
```

```
In [188]: # Load dataset data-text.csv from the URL provided using pandas read_csv method and
          df = pd.read_csv('https://raw.githubusercontent.com/jackiekazil/data-wrangling/maste
```

```
In [189]: # Display two records of dataframe df by calling head method on df
          df.head(2)
```

```
Out[189]:                          Indicator PUBLISH STATES  Year WHO region  \
          0  Life expectancy at birth (years)      Published  1990     Europe
          1  Life expectancy at birth (years)      Published  2000     Europe

            World Bank income group  Country         Sex  Display Value  Numeric  Low  \
          0             High-income  Andorra  Both sexes             77     77.0  NaN
```

```
1             High-income  Andorra  Both sexes             80     80.0  NaN

   High  Comments
0   NaN       NaN
1   NaN       NaN
```

```
In [190]: # Load dataset berlin_weather_oldest.csv from the URL provided using pandas read_csv
          df1 = pd.read_csv('https://raw.githubusercontent.com/kjam/data-wrangling-pycon/maste
          df1.head(2)
```

```
Out[190]:              STATION          STATION_NAME      DATE  PRCP  SNWD  SNOW  TMAX  \
          0  GHCND:GME00111445  BERLIN TEMPELHOF GM  19310101    46 -9999 -9999 -9999
          1  GHCND:GME00111445  BERLIN TEMPELHOF GM  19310102   107 -9999 -9999    50

             TMIN  WDFG  PGTM  ...   WT09  WT07  WT01  WT06  WT05  WT04  WT16  WT08  \
          0   -11 -9999 -9999  ...  -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
          1    11 -9999 -9999  ...  -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999

             WT18  WT03
          0 -9999 -9999
          1 -9999 -9999

          [2 rows x 21 columns]
```

1.3 1. Get the Metadata from the above files.

1.4 Solution Steps:

- Invoke method info on df with verbose=True, which will give metadata for dataframe df
- Invoke method info on df1 with verbose=True, which will give metadata for dataframe df1

```
In [191]: # Invoke method info on df with verbose=True will give metadata for dataframe df
          df.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4656 entries, 0 to 4655
Data columns (total 12 columns):
Indicator                   4656 non-null object
PUBLISH STATES              4656 non-null object
Year                        4656 non-null int64
WHO region                  4656 non-null object
World Bank income group     4656 non-null object
Country                     4656 non-null object
Sex                         4656 non-null object
Display Value               4656 non-null int64
Numeric                     4656 non-null float64
Low                         0 non-null float64
High                        0 non-null float64
Comments                    0 non-null float64
```

```
dtypes: float64(4), int64(2), object(6)
memory usage: 436.6+ KB
```

```
In [192]: # Invoke method info on df1 with verbose=True will give metadata for dataframe df1
          df1.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 117208 entries, 0 to 117207
Data columns (total 21 columns):
STATION         117208 non-null object
STATION_NAME    117208 non-null object
DATE            117208 non-null int64
PRCP            117208 non-null int64
SNWD            117208 non-null int64
SNOW            117208 non-null int64
TMAX            117208 non-null int64
TMIN            117208 non-null int64
WDFG            117208 non-null int64
PGTM            117208 non-null int64
WSFG            117208 non-null int64
WT09            117208 non-null int64
WT07            117208 non-null int64
WT01            117208 non-null int64
WT06            117208 non-null int64
WT05            117208 non-null int64
WT04            117208 non-null int64
WT16            117208 non-null int64
WT08            117208 non-null int64
WT18            117208 non-null int64
WT03            117208 non-null int64
dtypes: int64(19), object(2)
memory usage: 18.8+ MB
```

1.5 2. Get the row names from the above files.

1.6 Solution Steps:

- Create np.array using df.index to get row names for dataframe df
- Create np.array using df1.index to get row names for dataframe df1

```
In [193]: # Create np.array using df.index to get row names for dataframe df
          np.array(df.index)

Out[193]: array([   0,    1,    2, ..., 4653, 4654, 4655], dtype=int64)

In [195]: # Create np.array using df.index to get row names for dataframe df
          np.array(df1.index)

Out[195]: array([     0,      1,      2, ..., 117205, 117206, 117207], dtype=int64)
```

1.7 3. Change the column name from any of the above file.

1.8 Solution Steps:

- Invoke rename method on dataframe df with with a dictionary with entry having key as current Column Name "Indicator" and value as changed column name "indicator_id" and store to a temporary dataframe df_temp
- Display first two records of df_temp by calling head method with argument 2

```
In [196]: # Invoke rename method on dataframe df with with a dictionary with entry having
          # key as current Column Name "Indicator" and value as changed column name "indicator
          # store to a temporary dataframe df_temp
          df_temp = df.rename(columns= {"Indicator":"indicator_id"})

          # Display first two records
          df_temp.head(2)
```

```
Out[196]:                        indicator_id PUBLISH STATES  Year WHO region  \
          0  Life expectancy at birth (years)      Published  1990     Europe
          1  Life expectancy at birth (years)      Published  2000     Europe

            World Bank income group  Country         Sex  Display Value  Numeric  Low  \
          0             High-income  Andorra  Both sexes             77     77.0  NaN
          1             High-income  Andorra  Both sexes             80     80.0  NaN

             High  Comments
          0   NaN       NaN
          1   NaN       NaN
```

1.9 4. Change the column name from any of the above file and store the changes made permanently.

- Invoke rename method on dataframe df with a dictionary with entry having key as current Column Name "Indicator" and value as changed column name "indicator_id"
- To persist permanently pass inplace=True while invoking rename method on df

```
In [197]: # Invoke rename method on dataframe df with with a dictionary columns with entry hav
          # current Column Name "Indicator" and value as changed column name "indicator_id"
          # To persist permanently pass inplace=True while invoking rename method on df
          df.rename(columns= {"Indicator":"indicator_id"}, inplace=True)
```

```
In [198]: # Display first two records
          df.head(2)
```

```
Out[198]:                        indicator_id PUBLISH STATES  Year WHO region  \
          0  Life expectancy at birth (years)      Published  1990     Europe
          1  Life expectancy at birth (years)      Published  2000     Europe

            World Bank income group  Country         Sex  Display Value  Numeric  Low  \
          0             High-income  Andorra  Both sexes             77     77.0  NaN
```

```
1             High-income  Andorra  Both sexes             80     80.0  NaN

   High  Comments
0   NaN       NaN
1   NaN       NaN
```

1.10 5. Change the names of multiple columns.

1.11 Solution Steps:

- Invoke rename method on dataframe df with with a dictionary columns with two entries, first entry having key as existing Column Name "PUBLISH STATES" and value as changed column name "Publication Status", second entry having key as existing Column Name "WHO region" and value as changed Column Name "WHO Region"
- To persist permanently pass inplace=True while invoking rename method on df

```
In [199]: # Invoke rename method on dataframe df with with a dictionary columns with two entri
          # first entry having key as existing Column Name "PUBLISH STATES" and value as chang
          # "Publication Status", second entry having key as existing Column Name "WHO region"
          # changed Column Name "WHO Region"
          df.rename(columns= {"PUBLISH STATES":"Publication Status", "WHO region": "WHO Region
```

```
In [201]: df.head(2)
```

```
Out[201]:                      indicator_id Publication Status  Year WHO Region  \
          0  Life expectancy at birth (years)          Published  1990     Europe
          1  Life expectancy at birth (years)          Published  2000     Europe

            World Bank income group  Country         Sex  Display Value  Numeric  Low  \
          0             High-income  Andorra  Both sexes             77     77.0  NaN
          1             High-income  Andorra  Both sexes             80     80.0  NaN

             High  Comments
          0   NaN       NaN
          1   NaN       NaN
```

1.12 6. Arrange values of a particular column in ascending order.

1.13 Solution Steps:

- Invoke sort_values on dataframe df passing 'Year' as column to sort and ascending=True
- Call head method to display first 5 records

```
In [202]: # Invoke sort_values on dataframe df passing 'Year' as column to sort and ascending=
          # Call head method to display first 5 records
          df.sort_values(by='Year', ascending=True).head()
```

```
Out[202]:                            indicator_id Publication Status  Year WHO Region  \
          0       Life expectancy at birth (years)          Published  1990     Europe
          1270    Life expectancy at birth (years)          Published  1990     Europe
```

```
3193   Life expectancy at birth (years)          Published  1990     Europe
3194   Life expectancy at birth (years)          Published  1990     Europe
3197  Life expectancy at age 60 (years)          Published  1990     Europe

     World Bank income group              Country         Sex  Display Value  \
0                High-income              Andorra  Both sexes             77
1270             High-income              Germany        Male             72
3193     Lower-middle-income  Republic of Moldova        Male             65
3194     Lower-middle-income  Republic of Moldova  Both sexes             68
3197     Lower-middle-income  Republic of Moldova        Male             15

      Numeric  Low  High  Comments
0        77.0  NaN   NaN       NaN
1270     72.0  NaN   NaN       NaN
3193     65.0  NaN   NaN       NaN
3194     68.0  NaN   NaN       NaN
3197     15.0  NaN   NaN       NaN
```

1.14 7. Arrange multiple column values in ascending order.

1.15 Solution Steps:

- Invoke sort_values on dataframe df passing list of columns 'indicator_id','Country','Year', 'WHO Region', 'Publication Status' as column to sort and ascending=False for 'indicator_id' for rest of columns ascending=True and assign to dataframe df_temp
- Display only list of columns 'indicator_id','Country','Year', 'WHO Region', 'Publication Status'
- Call head method to display first 3 records on df_temp
- NOTE: There are few differences between output in this project and given output in project requirement, as if we make 'indicator_id' as ascending it will never match with Life expectancy, so I made with ascending=False, while rest of columns I have made ascending=True but still fields like 'Country', 'Year', 'WHO Region' will not match but I tried to keep as close output as possible

In [203]:
```
# Invoke sort_values on dataframe df passing list of columns 'indicator_id','Country
# 'WHO Region', 'Publication Status' as column to sort and ascending=False for 'indi
# rest of columns ascending=True and assign to dataframe df_temp
df_temp = df.sort_values(by=['indicator_id','Country','Year', 'WHO Region', 'Publica

# Display only list of columns 'indicator_id','Country','Year', 'WHO Region', 'Publi
# Call head method to display first 3 records on df_temp
df_temp[['indicator_id','Country','Year', 'WHO Region', 'Publication Status']].head(
```

Out[203]:
```
                          indicator_id      Country  Year  \
554   Life expectancy at birth (years)  Afghanistan  1990
965   Life expectancy at birth (years)  Afghanistan  1990
1792  Life expectancy at birth (years)  Afghanistan  1990
```

```
                 WHO Region Publication Status
554   Eastern Mediterranean          Published
965   Eastern Mediterranean          Published
1792  Eastern Mediterranean          Published
```

1.16 8. Make country as the first column of the dataframe.

1.17 Steps:

- Get list of column values from dataframe df and store into current_column_list
- Remove "Country" column from the current_column_list
- Insert "Country" as first column into the current_column_list
- Get first few records of df with columns as current_column_list

```
In [204]: # Get list of column values from dataframe df and store into current_column_list
          current_column_list = df.columns.values.tolist()

          #  Remove "Country" column from the current_column_list
          current_column_list.remove("Country")

          # Insert "Country" as first column into the current_column_list
          current_column_list.insert(0, "Country")

          # Get first few records of df with columns as current_column_list
          df[current_column_list].head()
```

```
Out[204]:                 Country                      indicator_id Publication Status  \
          0               Andorra   Life expectancy at birth (years)          Published
          1               Andorra   Life expectancy at birth (years)          Published
          2               Andorra  Life expectancy at age 60 (years)          Published
          3               Andorra  Life expectancy at age 60 (years)          Published
          4  United Arab Emirates   Life expectancy at birth (years)          Published

             Year             WHO Region World Bank income group         Sex  \
          0  1990                 Europe             High-income  Both sexes
          1  2000                 Europe             High-income  Both sexes
          2  2012                 Europe             High-income      Female
          3  2000                 Europe             High-income  Both sexes
          4  2012  Eastern Mediterranean             High-income      Female

             Display Value  Numeric  Low  High  Comments
          0             77     77.0  NaN   NaN       NaN
          1             80     80.0  NaN   NaN       NaN
          2             28     28.0  NaN   NaN       NaN
          3             23     23.0  NaN   NaN       NaN
          4             78     78.0  NaN   NaN       NaN
```

1.18 9. Get the column array using a variable

1.19 Solution Steps:

- Create a np.array on only column 'WHO Region' of dataframe df

```
In [205]: # Create a np.array on only column 'WHO Region' of dataframe df
          np.array(df['WHO Region'])
```

```
Out[205]: array(['Europe', 'Europe', 'Europe', ..., 'Africa', 'Africa', 'Africa'],
                dtype=object)
```

1.20 10. Get the subset rows 11, 24, 37

1.21 Solution Steps:

- Call iloc on dataframe df passign list of indexes 11, 24, 37

```
In [206]: # Call iloc on dataframe df passign list of indexes 11, 24, 37
          df.iloc[[11, 24, 37]]
```

```
Out[206]:                          indicator_id Publication Status  Year  \
          11   Life expectancy at birth (years)          Published  2012
          24  Life expectancy at age 60 (years)          Published  2012
          37  Life expectancy at age 60 (years)          Published  2012

                   WHO Region World Bank income group            Country     Sex  \
          11           Europe             High-income            Austria  Female
          24  Western Pacific             High-income  Brunei Darussalam  Female
          37           Europe             High-income             Cyprus  Female

              Display Value  Numeric  Low  High  Comments
          11             83     83.0  NaN   NaN       NaN
          24             21     21.0  NaN   NaN       NaN
          37             26     26.0  NaN   NaN       NaN
```

1.22 11. Get the subset rows excluding 5, 12, 23, and 56

1.23 Solution Steps:

- Exclude indexes 5,12,23,56 by calling drop method on dataframe df and passing these indexes

```
In [207]: # Exclude indexes 5,12,23,56 by calling drop method on dataframe df and passing thes
          df.drop(df.index[[5, 12, 23, 56]])
```

```
Out[207]:                                       indicator_id Publication Status  \
          0                    Life expectancy at birth (years)          Published
          1                    Life expectancy at birth (years)          Published
          2                   Life expectancy at age 60 (years)          Published
          3                   Life expectancy at age 60 (years)          Published
```

```
4                    Life expectancy at birth (years)          Published
6                   Life expectancy at age 60 (years)          Published
7                   Life expectancy at age 60 (years)          Published
8                    Life expectancy at birth (years)          Published
9                    Life expectancy at birth (years)          Published
10                   Life expectancy at birth (years)          Published
11                   Life expectancy at birth (years)          Published
13                   Life expectancy at birth (years)          Published
14                   Life expectancy at birth (years)          Published
15                   Life expectancy at birth (years)          Published
16                  Life expectancy at age 60 (years)          Published
17                   Life expectancy at birth (years)          Published
18                  Life expectancy at age 60 (years)          Published
19                   Life expectancy at birth (years)          Published
20                  Life expectancy at age 60 (years)          Published
21                  Life expectancy at age 60 (years)          Published
22                  Life expectancy at age 60 (years)          Published
24                  Life expectancy at age 60 (years)          Published
25                   Life expectancy at birth (years)          Published
26                  Life expectancy at age 60 (years)          Published
27                  Life expectancy at age 60 (years)          Published
28                   Life expectancy at birth (years)          Published
29                   Life expectancy at birth (years)          Published
30                  Life expectancy at age 60 (years)          Published
31                   Life expectancy at birth (years)          Published
32                  Life expectancy at age 60 (years)          Published
...                                                ...                ...
4626  Healthy life expectancy (HALE) at birth (years)          Published
4627  Healthy life expectancy (HALE) at birth (years)          Published
4628  Healthy life expectancy (HALE) at birth (years)          Published
4629  Healthy life expectancy (HALE) at birth (years)          Published
4630  Healthy life expectancy (HALE) at birth (years)          Published
4631  Healthy life expectancy (HALE) at birth (years)          Published
4632  Healthy life expectancy (HALE) at birth (years)          Published
4633  Healthy life expectancy (HALE) at birth (years)          Published
4634  Healthy life expectancy (HALE) at birth (years)          Published
4635  Healthy life expectancy (HALE) at birth (years)          Published
4636  Healthy life expectancy (HALE) at birth (years)          Published
4637  Healthy life expectancy (HALE) at birth (years)          Published
4638  Healthy life expectancy (HALE) at birth (years)          Published
4639  Healthy life expectancy (HALE) at birth (years)          Published
4640  Healthy life expectancy (HALE) at birth (years)          Published
4641  Healthy life expectancy (HALE) at birth (years)          Published
4642  Healthy life expectancy (HALE) at birth (years)          Published
4643  Healthy life expectancy (HALE) at birth (years)          Published
4644  Healthy life expectancy (HALE) at birth (years)          Published
4645  Healthy life expectancy (HALE) at birth (years)          Published
4646  Healthy life expectancy (HALE) at birth (years)          Published
```

```
4647  Healthy life expectancy (HALE) at birth (years)          Published
4648  Healthy life expectancy (HALE) at birth (years)          Published
4649  Healthy life expectancy (HALE) at birth (years)          Published
4650  Healthy life expectancy (HALE) at birth (years)          Published
4651  Healthy life expectancy (HALE) at birth (years)          Published
4652  Healthy life expectancy (HALE) at birth (years)          Published
4653  Healthy life expectancy (HALE) at birth (years)          Published
4654  Healthy life expectancy (HALE) at birth (years)          Published
4655  Healthy life expectancy (HALE) at birth (years)          Published

      Year             WHO Region World Bank income group  \
0     1990                 Europe             High-income
1     2000                 Europe             High-income
2     2012                 Europe             High-income
3     2000                 Europe             High-income
4     2012  Eastern Mediterranean             High-income
6     1990               Americas             High-income
7     2012               Americas             High-income
8     2012        Western Pacific             High-income
9     2000        Western Pacific             High-income
10    2012        Western Pacific             High-income
11    2012                 Europe             High-income
13    2012                 Europe             High-income
14    2000  Eastern Mediterranean             High-income
15    1990  Eastern Mediterranean             High-income
16    1990  Eastern Mediterranean             High-income
17    2012               Americas             High-income
18    2000               Americas             High-income
19    1990               Americas             High-income
20    2012               Americas             High-income
21    2012               Americas             High-income
22    1990        Western Pacific             High-income
24    2012        Western Pacific             High-income
25    2000               Americas             High-income
26    2000               Americas             High-income
27    1990               Americas             High-income
28    1990                 Europe             High-income
29    2012                 Europe             High-income
30    2000                 Europe             High-income
31    2012        Western Pacific             High-income
32    2012        Western Pacific             High-income
...    ...                    ...                     ...
4626  2012                 Europe     Upper-middle-income
4627  2012               Americas     Upper-middle-income
4628  2012                 Europe             High-income
4629  2012                 Africa     Lower-middle-income
4630  2000                 Africa     Upper-middle-income
4631  2000  Eastern Mediterranean     Lower-middle-income
```

```
4632  2012                 Africa              Low-income
4633  2000        South-East Asia     Lower-middle-income
4634  2000        South-East Asia     Lower-middle-income
4635  2000                 Europe              Low-income
4636  2012                 Europe              Low-income
4637  2012        Western Pacific     Lower-middle-income
4638  2012               Americas             High-income
4639  2012               Americas             High-income
4640  2000  Eastern Mediterranean     Lower-middle-income
4641  2012        Western Pacific     Upper-middle-income
4642  2000                 Africa              Low-income
4643  2000                 Europe     Lower-middle-income
4644  2012               Americas     Upper-middle-income
4645  2012               Americas     Upper-middle-income
4646  2012               Americas     Upper-middle-income
4647  2000               Americas     Upper-middle-income
4648  2012               Americas     Upper-middle-income
4649  2000        Western Pacific     Lower-middle-income
4650  2012        Western Pacific     Lower-middle-income
4651  2012        Western Pacific     Lower-middle-income
4652  2012  Eastern Mediterranean              Low-income
4653  2000                 Africa     Upper-middle-income
4654  2000                 Africa              Low-income
4655  2012                 Africa              Low-income

                                 Country         Sex  Display Value  Numeric  \
0                                Andorra  Both sexes             77     77.0
1                                Andorra  Both sexes             80     80.0
2                                Andorra      Female             28     28.0
3                                Andorra  Both sexes             23     23.0
4                   United Arab Emirates      Female             78     78.0
6                    Antigua and Barbuda        Male             17     17.0
7                    Antigua and Barbuda  Both sexes             22     22.0
8                              Australia        Male             81     81.0
9                              Australia  Both sexes             80     80.0
10                             Australia  Both sexes             83     83.0
11                               Austria      Female             83     83.0
13                               Belgium      Female             83     83.0
14                               Bahrain        Male             73     73.0
15                               Bahrain      Female             74     74.0
16                               Bahrain        Male             17     17.0
17                               Bahamas        Male             72     72.0
18                               Bahamas  Both sexes             21     21.0
19                              Barbados        Male             71     71.0
20                              Barbados      Female             25     25.0
21                              Barbados  Both sexes             23     23.0
22                     Brunei Darussalam      Female             20     20.0
24                     Brunei Darussalam      Female             21     21.0
```

```
25                                Canada      Female             82     82.0
26                                Canada        Male             21     21.0
27                                Canada      Female             24     24.0
28                           Switzerland        Male             74     74.0
29                           Switzerland  Both sexes             83     83.0
30                           Switzerland  Both sexes             23     23.0
31                          Cook Islands  Both sexes             76     76.0
32                          Cook Islands      Female             22     22.0
...                                  ...         ...            ...      ...
4626                              Serbia      Female             67     67.0
4627                            Suriname  Both sexes             66     66.0
4628                              Sweden  Both sexes             72     72.0
4629                           Swaziland      Female             47     47.0
4630                          Seychelles        Male             61     61.0
4631                Syrian Arab Republic      Female             64     64.0
4632                                Chad      Female             44     44.0
4633                            Thailand        Male             59     59.0
4634                            Thailand      Female             65     65.0
4635                          Tajikistan  Both sexes             56     56.0
4636                          Tajikistan      Female             60     60.0
4637                               Tonga      Female             61     61.0
4638                 Trinidad and Tobago      Female             64     64.0
4639                 Trinidad and Tobago  Both sexes             61     61.0
4640                             Tunisia        Male             63     63.0
4641                              Tuvalu        Male             57     57.0
4642                              Uganda      Female             40     40.0
4643                             Ukraine  Both sexes             60     60.0
4644                             Uruguay        Male             65     65.0
4645                             Uruguay      Female             70     70.0
4646                             Uruguay  Both sexes             68     68.0
4647    Saint Vincent and the Grenadines  Both sexes             61     61.0
4648  Venezuela (Bolivarian Republic of)  Both sexes             66     66.0
4649                             Vanuatu        Male             59     59.0
4650                               Samoa        Male             62     62.0
4651                               Samoa      Female             66     66.0
4652                               Yemen  Both sexes             54     54.0
4653                        South Africa        Male             49     49.0
4654                              Zambia  Both sexes             36     36.0
4655                            Zimbabwe      Female             51     51.0

      Low  High  Comments
0     NaN   NaN       NaN
1     NaN   NaN       NaN
2     NaN   NaN       NaN
3     NaN   NaN       NaN
4     NaN   NaN       NaN
6     NaN   NaN       NaN
7     NaN   NaN       NaN
```

```
8     NaN   NaN       NaN
9     NaN   NaN       NaN
10    NaN   NaN       NaN
11    NaN   NaN       NaN
13    NaN   NaN       NaN
14    NaN   NaN       NaN
15    NaN   NaN       NaN
16    NaN   NaN       NaN
17    NaN   NaN       NaN
18    NaN   NaN       NaN
19    NaN   NaN       NaN
20    NaN   NaN       NaN
21    NaN   NaN       NaN
22    NaN   NaN       NaN
24    NaN   NaN       NaN
25    NaN   NaN       NaN
26    NaN   NaN       NaN
27    NaN   NaN       NaN
28    NaN   NaN       NaN
29    NaN   NaN       NaN
30    NaN   NaN       NaN
31    NaN   NaN       NaN
32    NaN   NaN       NaN
...   ...   ...       ...
4626  NaN   NaN       NaN
4627  NaN   NaN       NaN
4628  NaN   NaN       NaN
4629  NaN   NaN       NaN
4630  NaN   NaN       NaN
4631  NaN   NaN       NaN
4632  NaN   NaN       NaN
4633  NaN   NaN       NaN
4634  NaN   NaN       NaN
4635  NaN   NaN       NaN
4636  NaN   NaN       NaN
4637  NaN   NaN       NaN
4638  NaN   NaN       NaN
4639  NaN   NaN       NaN
4640  NaN   NaN       NaN
4641  NaN   NaN       NaN
4642  NaN   NaN       NaN
4643  NaN   NaN       NaN
4644  NaN   NaN       NaN
4645  NaN   NaN       NaN
4646  NaN   NaN       NaN
4647  NaN   NaN       NaN
4648  NaN   NaN       NaN
4649  NaN   NaN       NaN
```

```
4650  NaN   NaN        NaN
4651  NaN   NaN        NaN
4652  NaN   NaN        NaN
4653  NaN   NaN        NaN
4654  NaN   NaN        NaN
4655  NaN   NaN        NaN

[4652 rows x 12 columns]
```

1.24 Load datasets from CSV

1.25 Solution steps

- Load users dataframe by calling pandas_csv method and calling the URL provided
- Load sessions dataframe by calling pandas_csv method and calling the URL provided
- Load products dataframe by calling pandas_csv method and calling the URL provided
- Load transactions dataframe by calling pandas_csv method and calling the URL provided

```
In [208]: # Load users dataframe by calling pandas_csv method and calling the URL provided
          users = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Da
```

```
In [209]: # Load sessions dataframe by calling pandas_csv method and calling the URL provided
          sessions = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/maste
```

```
In [210]: # Load products dataframe by calling pandas_csv method and calling the URL provided
          products = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/maste
```

```
In [211]: # Load transactions dataframe by calling pandas_csv method and calling the URL provi
          transactions = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/ma
```

```
In [212]: # Display first 5 records of users by calling head method on users
          users.head()
```

```
Out[212]:    UserID      User  Gender  Registered   Cancelled
          0       1   Charles    male  2012-12-21         NaN
          1       2     Pedro    male  2010-08-01  2010-08-08
          2       3  Caroline  female  2012-10-23  2016-06-07
          3       4   Brielle  female  2013-07-17         NaN
          4       5  Benjamin    male  2010-11-25         NaN
```

```
In [213]: # Display first 5 records of users by calling head method on sessions
          sessions.head()
```

```
Out[213]:    SessionID SessionDate  UserID
          0          1  2010-01-05       2
          1          2  2010-08-01       2
          2          3  2010-11-25       2
          3          4  2011-09-21       5
          4          5  2011-10-19       4
```

```
In [214]: # Display first 5 records of users by calling head method on transactions
          transactions.head()

Out[214]:    TransactionID TransactionDate  UserID  ProductID  Quantity
          0              1      2010-08-21     7.0          2         1
          1              2      2011-05-26     3.0          4         1
          2              3      2011-06-16     3.0          3         1
          3              4      2012-08-26     1.0          2         3
          4              5      2013-06-06     2.0          4         1
```

```
In [215]: # Display first 5 records of users by calling head method on products
          products.head()

Out[215]:    ProductID Product  Price
          0          1       A  14.16
          1          2       B  33.04
          2          3       C  10.65
          3          4       D  10.02
          4          5       E  29.66
```

1.26 12. Join users to transactions, keeping all rows from transactions and only matching rows from users (left join)

1.27 Solution:

- Perform left join on transactions dataframe and users dataframe so that all all rows from transactions are kept and only matching rows from users using pandas merge method

```
In [216]: # Perform left join on transactions dataframe and  users dataframe  so that all all
          # from transactions are kept and only matching rows from users using pandas merge me
          pd.merge(transactions, users,  how='left')

Out[216]:    TransactionID TransactionDate UserID  ProductID  Quantity      User  \
          0              1      2010-08-21      7          2         1       NaN
          1              2      2011-05-26      3          4         1  Caroline
          2              3      2011-06-16      3          3         1  Caroline
          3              4      2012-08-26      1          2         3   Charles
          4              5      2013-06-06      2          4         1     Pedro
          5              6      2013-12-23      2          5         6     Pedro
          6              7      2013-12-30      3          4         1  Caroline
          7              8      2014-04-24    NaN          2         3       NaN
          8              9      2015-04-24      7          4         3       NaN
          9             10      2016-05-08      3          4         4  Caroline

             Gender  Registered   Cancelled
          0     NaN         NaN         NaN
          1  female  2012-10-23  2016-06-07
          2  female  2012-10-23  2016-06-07
          3    male  2012-12-21         NaN
```

```
4     male  2010-08-01  2010-08-08
5     male  2010-08-01  2010-08-08
6   female  2012-10-23  2016-06-07
7      NaN         NaN         NaN
8      NaN         NaN         NaN
9   female  2012-10-23  2016-06-07
```

1.28 13. Which transactions have a UserID not in users?

1.29 Solution Steps:

- Apply a filter on transactions dataframe, where column 'UserID' in transactions dataframe is not in column 'UserID' in users dataframe

```
In [217]: # Apply a filter on transactions dataframe, here column 'UserID' in transactions dat
          # is not in column 'UserID' in users dataframe
          transactions[~transactions['UserID'].isin(users['UserID']) ]
```

```
Out[217]:    TransactionID TransactionDate  UserID  ProductID  Quantity
          0              1      2010-08-21     7.0          2         1
          7              8      2014-04-24     NaN          2         3
          8              9      2015-04-24     7.0          4         3
```

2 14. Join users to transactions, keeping only rows from transactions and users that match via UserID (inner join)

2.1 Solution Steps:

- Using pandas merge method perform inner join between dataframes transactions and users with 'UserID' as common field beween these dataframes

```
In [218]: # Using pandas merge method perform inner join between dataframes transactions and u
          # with 'UserID' as common field beween these dataframes
          pd.merge(transactions, users,  how='inner', on=['UserID'])
```

```
Out[218]:    TransactionID TransactionDate UserID  ProductID  Quantity      User  \
          0              2      2011-05-26      3          4         1  Caroline
          1              3      2011-06-16      3          3         1  Caroline
          2              7      2013-12-30      3          4         1  Caroline
          3             10      2016-05-08      3          4         4  Caroline
          4              4      2012-08-26      1          2         3   Charles
          5              5      2013-06-06      2          4         1     Pedro
          6              6      2013-12-23      2          5         6     Pedro

             Gender  Registered   Cancelled
          0  female  2012-10-23  2016-06-07
          1  female  2012-10-23  2016-06-07
          2  female  2012-10-23  2016-06-07
          3  female  2012-10-23  2016-06-07
```

```
4    male  2012-12-21         NaN
5    male  2010-08-01  2010-08-08
6    male  2010-08-01  2010-08-08
```

2.2 15. Join users to transactions, displaying all matching rows AND all non-matching rows (full outer join)

2.3 Steps:

- Using pandas merge method perform full outer join between transactions and users dataframe

```
In [219]: # Using pandas merge method perform full outer join between transactions and users d
          pd.merge(transactions, users, how='outer')
```

```
Out[219]:     TransactionID TransactionDate  UserID  ProductID  Quantity      User  \
          0             1.0      2010-08-21     7.0        2.0       1.0       NaN
          1             9.0      2015-04-24     7.0        4.0       3.0       NaN
          2             2.0      2011-05-26     3.0        4.0       1.0  Caroline
          3             3.0      2011-06-16     3.0        3.0       1.0  Caroline
          4             7.0      2013-12-30     3.0        4.0       1.0  Caroline
          5            10.0      2016-05-08     3.0        4.0       4.0  Caroline
          6             4.0      2012-08-26     1.0        2.0       3.0   Charles
          7             5.0      2013-06-06     2.0        4.0       1.0     Pedro
          8             6.0      2013-12-23     2.0        5.0       6.0     Pedro
          9             8.0      2014-04-24     NaN        2.0       3.0       NaN
          10            NaN             NaN     4.0        NaN       NaN   Brielle
          11            NaN             NaN     5.0        NaN       NaN  Benjamin

              Gender  Registered   Cancelled
          0      NaN         NaN         NaN
          1      NaN         NaN         NaN
          2   female  2012-10-23  2016-06-07
          3   female  2012-10-23  2016-06-07
          4   female  2012-10-23  2016-06-07
          5   female  2012-10-23  2016-06-07
          6     male  2012-12-21         NaN
          7     male  2010-08-01  2010-08-08
          8     male  2010-08-01  2010-08-08
          9      NaN         NaN         NaN
          10  female  2013-07-17         NaN
          11    male  2010-11-25         NaN
```

2.4 16. Determine which sessions occurred on the same day each user registered

2.5 Solution Steps:

- Perform inner join on dataframes users, sessions by joining on common field UserID and 'Registered' fiedl on users dataframe with 'SessionDate' on sessions dataframe using pandas merge method

```
In [220]: # Perform inner join on dataframes users, sessions by joining on common field 'UserI
          # and 'Registered' fiedl on users dataframe with 'SessionDate' on sessions dataframe
          # using pandas merge method
          pd.merge(users, sessions, how='inner', left_on=['UserID', 'Registered'], right_on=['U
```

```
Out[220]:    UserID     User  Gender  Registered   Cancelled  SessionID SessionDate
          0       2    Pedro    male  2010-08-01  2010-08-08          2  2010-08-01
          1       4  Brielle  female  2013-07-17         NaN          9  2013-07-17
```

2.6 17. Build a dataset with every possible (UserID, ProductID) pair (cross join)

3 Solution Steps:

- Create a new pandas dataframe users_temp_df from the 'UserID' column of users dataframe
- Create a new pandas dataframe products_temp_df from the 'ProductID' column of products dataframe
- To perform outer join a dummy common column 'key' is added to both the dataframes users_temp_df, products_temp_df
- Add a dummy column key to products_temp_df and initialize its value to 0
- Add a dummy column key to products_temp_df and initialize its value to 0
- Do full outer join on dataframes users_temp_df, products_temp_df using padas merge and take only columns "UserID", "ProductID" and assign the result to dataframe user_id_product_id_df

```
In [221]: # Create a new pandas dataframe users_temp_df from the 'UserID' column of users data
          users_temp_df = pd.DataFrame(users['UserID'])

          # Create a new pandas dataframe products_temp_df from the 'ProductID' column of prod
          products_temp_df = pd.DataFrame(products['ProductID'])

          # To perform outer join a dummy common column 'key' is added to both the dataframes
          # Add a dummy column key to products_temp_df and initialize its value to 0
          # Add a dummy column key to products_temp_df and initialize its value to 0
          users_temp_df['key'] = 0
          products_temp_df['key'] = 0

          # Do full outer join on dataframes users_temp_df, products_temp_df using padas merge
          # take only columns "UserID", "ProductID" and assign the result to dataframe user_id
          user_id_product_id_df = pd.merge(users_temp_df, products_temp_df, how='outer')[["Use
          user_id_product_id_df
```

```
Out[221]:     UserID  ProductID
          0        1          1
          1        1          2
          2        1          3
          3        1          4
          4        1          5
          5        2          1
```

```
6          2            2
7          2            3
8          2            4
9          2            5
10         3            1
11         3            2
12         3            3
13         3            4
14         3            5
15         4            1
16         4            2
17         4            3
18         4            4
19         4            5
20         5            1
21         5            2
22         5            3
23         5            4
24         5            5
```

3.1 18. Determine how much quantity of each product was purchased by each user

3.2 Solution Steps:

- Perform left join on user_id_product_id_df, transactions dataframe using fields 'UserID', 'ProductID' using pandas merge method
- Perform sum on Quatity column of user_product_transation_df group by 'UserID', 'ProductID'

```
In [222]: # Perform left join on user_id_product_id_df, transactions dataframe using fields 'U
          user_product_transation_df = pd.merge(user_id_product_id_df, transactions,  how='lef

          #  Perform sum on Quatity column of user_product_transation_df group by 'UserID', 'P
          user_product_transation_df.groupby(['UserID', 'ProductID'], as_index=False).Quantity
```

```
Out[222]:     UserID  ProductID  Quantity
          0        1          1       0.0
          1        1          2       3.0
          2        1          3       0.0
          3        1          4       0.0
          4        1          5       0.0
          5        2          1       0.0
          6        2          2       0.0
          7        2          3       0.0
          8        2          4       1.0
          9        2          5       6.0
          10       3          1       0.0
          11       3          2       0.0
          12       3          3       1.0
```

```
13        3           4        6.0
14        3           5        0.0
15        4           1        0.0
16        4           2        0.0
17        4           3        0.0
18        4           4        0.0
19        4           5        0.0
20        5           1        0.0
21        5           2        0.0
22        5           3        0.0
23        5           4        0.0
24        5           5        0.0
```

3.3 19. For each user, get each possible pair of pair transactions (TransactionID1,TransacationID2)

3.4 Solution Steps:

- Perform full outer self join on same transactions dataframe twice on column 'UserID' using merge method of pandas

```
In [223]: # Perform full outer self join on same transactions dataframe twice on column 'UserI
          pd.merge(transactions, transactions,  how='outer', on=['UserID'])
```

```
Out[223]:     TransactionID_x TransactionDate_x  UserID  ProductID_x  Quantity_x  \
          0                 1        2010-08-21     7.0            2           1
          1                 1        2010-08-21     7.0            2           1
          2                 9        2015-04-24     7.0            4           3
          3                 9        2015-04-24     7.0            4           3
          4                 2        2011-05-26     3.0            4           1
          5                 2        2011-05-26     3.0            4           1
          6                 2        2011-05-26     3.0            4           1
          7                 2        2011-05-26     3.0            4           1
          8                 3        2011-06-16     3.0            3           1
          9                 3        2011-06-16     3.0            3           1
          10                3        2011-06-16     3.0            3           1
          11                3        2011-06-16     3.0            3           1
          12                7        2013-12-30     3.0            4           1
          13                7        2013-12-30     3.0            4           1
          14                7        2013-12-30     3.0            4           1
          15                7        2013-12-30     3.0            4           1
          16               10        2016-05-08     3.0            4           4
          17               10        2016-05-08     3.0            4           4
          18               10        2016-05-08     3.0            4           4
          19               10        2016-05-08     3.0            4           4
          20                4        2012-08-26     1.0            2           3
          21                5        2013-06-06     2.0            4           1
          22                5        2013-06-06     2.0            4           1
          23                6        2013-12-23     2.0            5           6
```

```
24                6        2013-12-23     2.0             5           6
25                8        2014-04-24     NaN             2           3

    TransactionID_y TransactionDate_y  ProductID_y  Quantity_y
0                 1        2010-08-21            2           1
1                 9        2015-04-24            4           3
2                 1        2010-08-21            2           1
3                 9        2015-04-24            4           3
4                 2        2011-05-26            4           1
5                 3        2011-06-16            3           1
6                 7        2013-12-30            4           1
7                10        2016-05-08            4           4
8                 2        2011-05-26            4           1
9                 3        2011-06-16            3           1
10                7        2013-12-30            4           1
11               10        2016-05-08            4           4
12                2        2011-05-26            4           1
13                3        2011-06-16            3           1
14                7        2013-12-30            4           1
15               10        2016-05-08            4           4
16                2        2011-05-26            4           1
17                3        2011-06-16            3           1
18                7        2013-12-30            4           1
19               10        2016-05-08            4           4
20                4        2012-08-26            2           3
21                5        2013-06-06            4           1
22                6        2013-12-23            5           6
23                5        2013-06-06            4           1
24                6        2013-12-23            5           6
25                8        2014-04-24            2           3
```

3.5 20. Join each user to his/her first occuring transaction in the transactions table

3.6 Solution Steps:

- Get the list of indexes from minimum value of TransactionID by Group transactions dataframe based on UserID column
- Get the records for the transaction for list of indexes found above and assign to dataframe first_transacton_df
- For the UserID column first_transacton_df apply lambda expression of converting to integer. This is needed while joining with dataframe users as by default UserID column in first_transacton_df becomes float type
- Left join users and first_transacton_df dataframes based on common column 'UserID' using pandas merge method

```
In [167]: # Get the list of indexes from minimum value of TransactionID by Group transactions
          # Get the records for the transaction for list of indexes found above and assign to
          first_transacton_df = transactions.loc[transactions.groupby(["UserID"], as_index=Fal
          # For the UserID column first_transacton_df apply lambda expression of converting to
          # This is needed while joining with dataframe users, as by default values in UserID
          # in first_transacton_df becomes float type
          first_transacton_df["UserID"] = first_transacton_df["UserID"].apply(lambda x: int(x)

          # Display first_transacton_df
          first_transacton_df
```

```
Out[167]:    TransactionID TransactionDate  UserID  ProductID  Quantity
          3              4      2012-08-26       1          2         3
          4              5      2013-06-06       2          4         1
          1              2      2011-05-26       3          4         1
          0              1      2010-08-21       7          2         1
```

```
In [165]: # Left join users and first_transacton_df dataframes based on common column 'UserID'
          # using pandas merge method
          # Note: UserID 7 on first_transacton_df does not show up in the data as there is no
          # UserID in users table, this is likely to be a invalid data
          pd.merge(users, first_transacton_df, how='left', on='UserID')
```

```
Out[165]:    UserID      User  Gender  Registered   Cancelled  TransactionID  \
          0       1   Charles    male  2012-12-21         NaN            4.0
          1       2     Pedro    male  2010-08-01  2010-08-08            5.0
          2       3  Caroline  female  2012-10-23  2016-06-07            2.0
          3       4   Brielle  female  2013-07-17         NaN            NaN
          4       5  Benjamin    male  2010-11-25         NaN            NaN

            TransactionDate  ProductID  Quantity
          0      2012-08-26        2.0       3.0
          1      2013-06-06        4.0       1.0
          2      2011-05-26        4.0       1.0
          3             NaN        NaN       NaN
          4             NaN        NaN       NaN
```

3.7 21. Test to see if we can drop columns

3.8 Solution Steps:

- Left join users and transactions dataframes using pands merge method and assign the result to dataframe data
- Get list of columns of data and display them
- Set threshold to drop NAs in data
- Find and display columns in data which has missing values
- Find and Display number of missing values in each column
- Find and Display percentage of missing values in each column

```
In [224]: # Left join users and transactions dataframes using pands merge method and
          # assign the result to dataframe data
          data = pd.merge(users, transactions, how='left')
```

```
         # Display data
         data
```

```
Out[224]:   UserID      User  Gender  Registered   Cancelled  TransactionID  \
          0       1   Charles    male  2012-12-21         NaN            4.0
          1       2     Pedro    male  2010-08-01  2010-08-08            5.0
          2       2     Pedro    male  2010-08-01  2010-08-08            6.0
          3       3  Caroline  female  2012-10-23  2016-06-07            2.0
          4       3  Caroline  female  2012-10-23  2016-06-07            3.0
          5       3  Caroline  female  2012-10-23  2016-06-07            7.0
          6       3  Caroline  female  2012-10-23  2016-06-07           10.0
          7       4   Brielle  female  2013-07-17         NaN            NaN
          8       5  Benjamin    male  2010-11-25         NaN            NaN

            TransactionDate  ProductID  Quantity
          0      2012-08-26        2.0       3.0
          1      2013-06-06        4.0       1.0
          2      2013-12-23        5.0       6.0
          3      2011-05-26        4.0       1.0
          4      2011-06-16        3.0       1.0
          5      2013-12-30        4.0       1.0
          6      2016-05-08        4.0       4.0
          7             NaN        NaN       NaN
          8             NaN        NaN       NaN
```

```
In [158]: # Get list of columns of data and display them
          my_columns = list(data.columns)
          my_columns
```

```
Out[158]: ['UserID',
           'User',
           'Gender',
           'Registered',
           'Cancelled',
           'TransactionID',
           'TransactionDate',
           'ProductID',
           'Quantity']
```

```
In [225]: #set threshold to drop NAs
          list(data.dropna(thresh=int(data.shape[0] * .9), axis=1).columns)
```

```
Out[225]: ['UserID', 'User', 'Gender', 'Registered']
```

```
In [226]: # Find and display columns in data which has missing values
          missing_info = list(data.columns[data.isnull().any()])
          missing_info
```

```
Out[226]: ['Cancelled', 'TransactionID', 'TransactionDate', 'ProductID', 'Quantity']
```

```
In [161]: # Find and Display number of missing values in each column
          for col in missing_info:
              num_missing = data[data[col].isnull() == True].shape[0]
              print('number missing for column {}: {}'.format(col, num_missing))
```

```
number missing for column Cancelled: 3
number missing for column TransactionID: 2
number missing for column TransactionDate: 2
number missing for column ProductID: 2
number missing for column Quantity: 2
```

```
In [227]: # Find and Display percentage of missing values in each column
          for col in missing_info:
              percent_missing = data[data[col].isnull() == True].shape[0] / data.shape[0] * 100
              print('percent missing for column {}: {:.2f}%'.format(col, round(percent_missing
```

```
percent missing for column Cancelled: 33.33%
percent missing for column TransactionID: 22.22%
percent missing for column TransactionDate: 22.22%
percent missing for column ProductID: 22.22%
percent missing for column Quantity: 22.22%
```