day5

April 8, 2022

[2]:
```
# Import libraries
import pandas as pd
```

[5]:
```
file_url = 'https://github.com/fenago/MLEssentials/blob/main/datasets/
↪Online%20Retail.xlsx?raw=true'
```

[6]:
```
df = pd.read_excel(file_url)
df.head()
```

[6]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

[4]:
```
uk_holidays_2010 = pd.read_csv('https://date.nager.at/PublicHoliday/Country/GB/
↪2010/CSV')
```

[5]:
```
uk_holidays_2010.shape
```

[5]:
```
(13, 9)
```

[6]:
```
uk_holidays_2010.head()
```

[6]:

	Date	LocalName	Name	CountryCode	Fixed
0	2010-01-01	New Year's Day	New Year's Day	GB	False
1	2010-01-04	New Year's Day	New Year's Day	GB	False
2	2010-03-17	Saint Patrick's Day	Saint Patrick's Day	GB	True
3	2010-04-02	Good Friday	Good Friday	GB	False

```
4  2010-04-05        Easter Monday        Easter Monday          GB  False

   Global  LaunchYear    Type               Counties
0    True         NaN  Public                    NaN
1   False         NaN  Public                 GB-SCT
2   False         NaN  Public                 GB-NIR
3    True         NaN  Public                    NaN
4   False         NaN  Public  GB-ENG,GB-WLS,GB-NIR
```

[7]:
```
uk_holidays_2011 = pd.read_csv('https://date.nager.at/PublicHoliday/Country/GB/
↪2011/CSV')
```

[8]:
```
uk_holidays_2011.shape
```

[8]:
```
(15, 9)
```

[9]:
```
uk_holidays_2011.head()
```

[9]:
```
         Date            LocalName                 Name CountryCode  Fixed  \
0  2011-01-01       New Year's Day       New Year's Day          GB  False
1  2011-01-03       New Year's Day       New Year's Day          GB  False
2  2011-01-03       New Year's Day       New Year's Day          GB  False
3  2011-01-04       New Year's Day       New Year's Day          GB  False
4  2011-03-17  Saint Patrick's Day  Saint Patrick's Day          GB   True

   Global  LaunchYear    Type       Counties
0   False         NaN  Public         GB-NIR
1   False         NaN  Public  GB-ENG,GB-WLS
2   False         NaN  Public         GB-SCT
3   False         NaN  Public         GB-SCT
4   False         NaN  Public         GB-NIR
```

[10]:
```
uk_holidays = uk_holidays_2010.append(uk_holidays_2011)
```

[11]:
```
uk_holidays.shape
```

[11]:
```
(28, 9)
```

[12]:
```
uk_holidays.head(28)
```

[12]:
```
          Date               LocalName                    Name CountryCode  \
0   2010-01-01          New Year's Day          New Year's Day          GB
1   2010-01-04          New Year's Day          New Year's Day          GB
2   2010-03-17     Saint Patrick's Day     Saint Patrick's Day          GB
3   2010-04-02            Good Friday             Good Friday          GB
4   2010-04-05          Easter Monday           Easter Monday          GB
5   2010-05-03  Early May Bank Holiday  Early May Bank Holiday          GB
```

6	2010-05-31	Spring Bank Holiday	Spring Bank Holiday	GB
7	2010-07-12	Battle of the Boyne	Battle of the Boyne	GB
8	2010-08-02	Summer Bank Holiday	Summer Bank Holiday	GB
9	2010-08-30	Summer Bank Holiday	Summer Bank Holiday	GB
10	2010-11-30	Saint Andrew's Day	Saint Andrew's Day	GB
11	2010-12-27	Christmas Day	Christmas Day	GB
12	2010-12-28	Boxing Day	St. Stephen's Day	GB
0	2011-01-01	New Year's Day	New Year's Day	GB
1	2011-01-03	New Year's Day	New Year's Day	GB
2	2011-01-03	New Year's Day	New Year's Day	GB
3	2011-01-04	New Year's Day	New Year's Day	GB
4	2011-03-17	Saint Patrick's Day	Saint Patrick's Day	GB
5	2011-04-22	Good Friday	Good Friday	GB
6	2011-04-25	Easter Monday	Easter Monday	GB
7	2011-05-02	Early May Bank Holiday	Early May Bank Holiday	GB
8	2011-05-30	Spring Bank Holiday	Spring Bank Holiday	GB
9	2011-07-12	Battle of the Boyne	Battle of the Boyne	GB
10	2011-08-01	Summer Bank Holiday	Summer Bank Holiday	GB
11	2011-08-29	Summer Bank Holiday	Summer Bank Holiday	GB
12	2011-11-30	Saint Andrew's Day	Saint Andrew's Day	GB
13	2011-12-26	Boxing Day	St. Stephen's Day	GB
14	2011-12-27	Christmas Day	Christmas Day	GB

	Fixed	Global	LaunchYear	Type	Counties
0	False	True	NaN	Public	NaN
1	False	False	NaN	Public	GB-SCT
2	True	False	NaN	Public	GB-NIR
3	False	True	NaN	Public	NaN
4	False	False	NaN	Public	GB-ENG,GB-WLS,GB-NIR
5	False	True	1978.0	Public	NaN
6	False	True	1971.0	Public	NaN
7	True	False	NaN	Public	GB-NIR
8	False	False	1971.0	Public	GB-SCT
9	False	False	1971.0	Public	GB-ENG,GB-WLS,GB-NIR
10	True	False	NaN	Public	GB-SCT
11	False	True	NaN	Public	NaN
12	False	True	NaN	Public	NaN
0	False	False	NaN	Public	GB-NIR
1	False	False	NaN	Public	GB-ENG,GB-WLS
2	False	False	NaN	Public	GB-SCT
3	False	False	NaN	Public	GB-SCT
4	True	False	NaN	Public	GB-NIR
5	False	True	NaN	Public	NaN
6	False	False	NaN	Public	GB-ENG,GB-WLS,GB-NIR
7	False	True	1978.0	Public	NaN
8	False	True	1971.0	Public	NaN
9	True	False	NaN	Public	GB-NIR

```
10  False   False      1971.0  Public                 GB-SCT
11  False   False      1971.0  Public  GB-ENG,GB-WLS,GB-NIR
12   True   False         NaN  Public                 GB-SCT
13  False    True         NaN  Public                    NaN
14  False    True         NaN  Public                    NaN
```

[13]:
```
uk_holidays.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28 entries, 0 to 14
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         28 non-null     object
 1   LocalName    28 non-null     object
 2   Name         28 non-null     object
 3   CountryCode  28 non-null     object
 4   Fixed        28 non-null     bool
 5   Global       28 non-null     bool
 6   LaunchYear   8 non-null      float64
 7   Type         28 non-null     object
 8   Counties     17 non-null     object
dtypes: bool(2), float64(1), object(6)
memory usage: 1.8+ KB
```

[14]:
```
# create new col and reformat invoice date
df['InvoiceDay'] = df['InvoiceDate'].astype(str).str.slice(stop=10)
df.head()
```

[14]:
```
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

          InvoiceDate  UnitPrice  CustomerID         Country  InvoiceDay
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom  2010-12-01
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom  2010-12-01
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom  2010-12-01
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom  2010-12-01
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom  2010-12-01
```

[15]:
```
# Preserve all records in the left data set and match them with the records on␣
 ↪the right data set
df_left = pd.merge(df, uk_holidays, left_on='InvoiceDay', right_on='Date',␣
 ↪how='left')
```

```
df_left.shape
```

[15]:
```
(541909, 18)
```

[16]:
```
df_left.head()
```

[16]:
```
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

          InvoiceDate  UnitPrice  CustomerID         Country  InvoiceDay Date  \
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom  2010-12-01  NaN
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom  2010-12-01  NaN
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom  2010-12-01  NaN
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom  2010-12-01  NaN
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom  2010-12-01  NaN

  LocalName Name CountryCode Fixed Global  LaunchYear Type Counties
0       NaN  NaN         NaN   NaN    NaN         NaN  NaN      NaN
1       NaN  NaN         NaN   NaN    NaN         NaN  NaN      NaN
2       NaN  NaN         NaN   NaN    NaN         NaN  NaN      NaN
3       NaN  NaN         NaN   NaN    NaN         NaN  NaN      NaN
4       NaN  NaN         NaN   NaN    NaN         NaN  NaN      NaN
```

[17]:
```
df_right = df.merge(uk_holidays, left_on='InvoiceDay', right_on='Date',␣
↪how='right')
```

[18]:
```
df_right.shape
```

[18]:
```
(9602, 18)
```

[19]:
```
# with Left Join we will know which of the invice days are holidyas. while with␣
↪right join, we will know what are invoices on holidays.
```

[20]:
```
df_inner = df.merge(uk_holidays, left_on='InvoiceDay', right_on='Date',␣
↪how='inner')
df_inner.shape
```

[20]:
```
(9579, 18)
```

[21]:
```
df_outer = df.merge(uk_holidays, left_on='InvoiceDay', right_on='Date',␣
↪how='outer')
df_outer.shape
```

[21]:
```
(541932, 18)
```

[ ]:
```
# Binning
```

[22]:
```
df['Country'].unique()
```

[22]:
```
array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany',
       'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
       'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland',
       'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria',
       'Israel', 'Finland', 'Bahrain', 'Greece', 'Hong Kong', 'Singapore',
       'Lebanon', 'United Arab Emirates', 'Saudi Arabia',
       'Czech Republic', 'Canada', 'Unspecified', 'Brazil', 'USA',
       'European Community', 'Malta', 'RSA'], dtype=object)
```

[23]:
```
df['Country_bin'] = df['Country']
```

[27]:
```
asian_countries = ['Japan', 'Hong Kong', 'Singapore']
```

[29]:
```
# change all values that are in asian countries to "Asia"
# i.e. all values equal to Japan, Hong Kong, or Singapore are now categorized␣
 ↪as "Asia"

# commonly used for ages (categorizing into certain deciles)
df.loc[df['Country'].isin(asian_countries), 'Country_bin'] = 'Asia'
df['Country_bin'].unique()
```

[29]:
```
array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany',
       'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
       'Italy', 'Belgium', 'Lithuania', 'Asia', 'Iceland',
       'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria',
       'Middle East', 'Finland', 'Greece', 'Czech Republic', 'Canada',
       'Unspecified', 'Brazil', 'USA', 'European Community', 'Malta',
       'RSA'], dtype=object)
```

[30]:
```
m_east_countries = ['Israel', 'Bahrain', 'Lebanon', 'United Arab Emirates',␣
 ↪'Saudi Arabia']
df.loc[df['Country'].isin(m_east_countries), 'Country_bin'] = 'Middle East'
df['Country_bin'].unique()
```

[30]:
```
array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany',
       'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
       'Italy', 'Belgium', 'Lithuania', 'Asia', 'Iceland',
       'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria',
       'Middle East', 'Finland', 'Greece', 'Czech Republic', 'Canada',
       'Unspecified', 'Brazil', 'USA', 'European Community', 'Malta',
       'RSA'], dtype=object)
```

[31]:
```
american_countries = ['Canada', 'Brazil', 'USA']
df.loc[df['Country'].isin(american_countries), 'Country_bin'] = 'America'
df['Country_bin'].unique()
```

[31]:
```
array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany',
       'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
       'Italy', 'Belgium', 'Lithuania', 'Asia', 'Iceland',
       'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria',
       'Middle East', 'Finland', 'Greece', 'Czech Republic', 'America',
       'Unspecified', 'European Community', 'Malta', 'RSA'], dtype=object)
```

[32]:
```
df['Country_bin'].nunique()
```

[32]:
```
30
```

[33]:
```
# Dates
```

[34]:
```
df.dtypes
```

[34]:
```
InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID            float64
Country                object
InvoiceDay             object
Country_bin            object
dtype: object
```

[10]:
```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])
```

[8]:
```
df['InvoiceDate'].dt.dayofweek
```

[8]:
```
0         2
1         2
2         2
3         2
4         2
         ..
541904    4
541905    4
541906    4
541907    4
541908    4
Name: InvoiceDate, Length: 541909, dtype: int64
```

[9]:
```
df['InvoiceDate'].dt.year
```

[9]:
```
0         2010
1         2010
2         2010
3         2010
4         2010
          …
541904    2011
541905    2011
541906    2011
541907    2011
541908    2011
Name: InvoiceDate, Length: 541909, dtype: int64
```

[11]:
```
df['InvoiceDate'] + pd.tseries.offsets.BusinessDay(-1)
```

[11]:
```
0        2010-11-30 08:26:00
1        2010-11-30 08:26:00
2        2010-11-30 08:26:00
3        2010-11-30 08:26:00
4        2010-11-30 08:26:00
                 …
541904   2011-12-08 12:50:00
541905   2011-12-08 12:50:00
541906   2011-12-08 12:50:00
541907   2011-12-08 12:50:00
541908   2011-12-08 12:50:00
Name: InvoiceDate, Length: 541909, dtype: datetime64[ns]
```

[12]:
```
df['InvoiceDate'] + pd.tseries.offsets.Day(3)
```

[12]:
```
0        2010-12-04 08:26:00
1        2010-12-04 08:26:00
2        2010-12-04 08:26:00
3        2010-12-04 08:26:00
4        2010-12-04 08:26:00
                 …
541904   2011-12-12 12:50:00
541905   2011-12-12 12:50:00
541906   2011-12-12 12:50:00
541907   2011-12-12 12:50:00
541908   2011-12-12 12:50:00
Name: InvoiceDate, Length: 541909, dtype: datetime64[ns]
```

[13]:
```
df['InvoiceDate'] + pd.Timedelta(1, unit='MS')
```

[13]:
```
0        2010-12-01 08:26:00.001
1        2010-12-01 08:26:00.001
2        2010-12-01 08:26:00.001
3        2010-12-01 08:26:00.001
4        2010-12-01 08:26:00.001
                   …
541904   2011-12-09 12:50:00.001
541905   2011-12-09 12:50:00.001
541906   2011-12-09 12:50:00.001
541907   2011-12-09 12:50:00.001
541908   2011-12-09 12:50:00.001
Name: InvoiceDate, Length: 541909, dtype: datetime64[ns]
```

[ ]:
```
# Data Aggregation
# 2 important parts: group by method amd aggregation method
```

[15]:
```
# aggregate quantity and do a sum by country
# total volume of items sold for every country
df.groupby('Country').agg({'Quantity': 'sum'})
```

[15]:

Country	Quantity
Australia	83653
Austria	4827
Bahrain	260
Belgium	23152
Brazil	356
Canada	2763
Channel Islands	9479
Cyprus	6317
Czech Republic	592
Denmark	8188
EIRE	142637
European Community	497
Finland	10666
France	110480
Germany	117448
Greece	1556
Hong Kong	4769
Iceland	2458
Israel	4353
Italy	7999
Japan	25218
Lebanon	386
Lithuania	652
Malta	944
Netherlands	200128

```
Norway                    19247
Poland                     3653
Portugal                  16180
RSA                         352
Saudi Arabia                 75
Singapore                  5234
Spain                     26824
Sweden                    35637
Switzerland               30325
USA                        1034
United Arab Emirates        982
United Kingdom          4263829
Unspecified                3300
```

[16]:
```
df.groupby(['Country', 'StockCode']).agg({'Quantity': 'sum'})
```

[16]:
```
                        Quantity
Country     StockCode
Australia   15036            600
            20665              6
            20675            216
            20676            216
            20677            216
…                            …
Unspecified 85049A             1
            85179A             1
            85179C             1
            85180A             2
            85180B             1

[19839 rows x 1 columns]
```

[ ]:
```
# items sold for every country
```

[17]:
```
# create new feature called "Invoice Date"
df['Invoice_Date'] = df['InvoiceDate'].dt.date
df.groupby(['Country', 'StockCode', 'Invoice_Date']).agg({'Quantity': 'sum'})
```

[17]:
```
                                    Quantity
Country     StockCode Invoice_Date
Australia   15036     2011-05-17         600
            20665     2011-03-24           6
            20675     2011-01-06          72
                      2011-03-03         144
            20676     2011-01-06          72
…                                        …
Unspecified 85049A    2011-07-28           1
```

```
              85179A    2011-07-28           1
              85179C    2011-07-28           1
              85180A    2011-07-28           2
              85180B    2011-07-28           1

[310015 rows x 1 columns]
```

[18]:
```
df_agg = df.groupby(['Country', 'StockCode', 'Invoice_Date']).agg({'Quantity':␣
↪'sum'}).reset_index()
df_agg.head()
```

[18]:

	Country	StockCode	Invoice_Date	Quantity
0	Australia	15036	2011-05-17	600
1	Australia	20665	2011-03-24	6
2	Australia	20675	2011-01-06	72
3	Australia	20675	2011-03-03	144
4	Australia	20676	2011-01-06	72

[19]:
```
#merge back into original data set
df_merged = pd.merge(df, df_agg, how='left', on = ['Country', 'StockCode',␣
↪'Invoice_Date'])
df_merged
```

[19]:

	InvoiceNo	StockCode	Description	Quantity_x	\
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	
1	536365	71053	WHITE METAL LANTERN	6	
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	
…	…	…	…	…	
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	

	InvoiceDate	UnitPrice	CustomerID	Country	\
0	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	
1	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	
2	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	
3	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	
4	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	
…	…	…	…	…	
541904	2011-12-09 12:50:00	0.85	12680.0	France	
541905	2011-12-09 12:50:00	2.10	12680.0	France	
541906	2011-12-09 12:50:00	4.15	12680.0	France	
541907	2011-12-09 12:50:00	4.15	12680.0	France	

```
541908 2011-12-09 12:50:00       4.95     12680.0          France

       Invoice_Date  Quantity_y
0        2010-12-01         454
1        2010-12-01          33
2        2010-12-01          40
3        2010-12-01          59
4        2010-12-01         551
…               …           …
541904   2011-12-09          12
541905   2011-12-09           6
541906   2011-12-09           4
541907   2011-12-09           4
541908   2011-12-09           3

[541909 rows x 10 columns]
```

[ ]: