Analyze

November 6, 2022

1 Import software libraries

[1]:
```
# Import required libraries.
import sys                          # Read system parameters.
import numpy as np                  # Work with multi-dimensional arrays.
import pandas as pd                 # Manipulate and analyze data.
import matplotlib                   # Create and format charts.
import matplotlib.pyplot as plt
import seaborn as sns               # Make charting easier.
import category_encoders as ce

# Summarize software libraries used.
print('Libraries used in this project:')
print('- NumPy {}'.format(np.__version__))
print('- Python {}'.format(sys.version))
print('- pandas {}'.format(pd.__version__))
print('- Matplotlib {}'.format(matplotlib.__version__))
print('- Seaborn {}'.format(sns.__version__))
```

```
Libraries used in this project:
- NumPy 1.19.2
- Python 3.7.6 | packaged by conda-forge | (default, Mar 23 2020, 23:03:20)
[GCC 7.3.0]
- pandas 1.1.3
- Matplotlib 3.3.2
- Seaborn 0.11.0
```

2 Read and examine the data

[2]:
```
# Read the data that was put through the ETL process in Course 2 of the CDSP␣
↪Specialization.

data_1 = pd.read_pickle('data/online_history_cleaned.pickle')

# Preview the first five rows of the data.
```

```
data_1.head()
```

[2]:

	Invoice	StockCode	Quantity	InvoiceDate	Price	CustomerID	Country	TotalAmount	Description
0	536365	85123A	6	2010-12-01 08:26:00	2.55	u1785	United Kingdom	15.30	CREAM HANGING HEART T-LIGHT HOLDER
1	536367	84879	32	2010-12-01 08:34:00	1.69	u13047	United Kingdom	54.08	ASSORTED COLOUR BIRD ORNAMENT
2	536373	85123A	6	2010-12-01 09:02:00	2.55	u1785	United Kingdom	15.30	CREAM HANGING HEART T-LIGHT HOLDER
3	536375	85123A	6	2010-12-01 09:32:00	2.55	u1785	United Kingdom	15.30	CREAM HANGING HEART T-LIGHT HOLDER
4	536378	20725	10	2010-12-01 09:37:00	1.65	u14688	United Kingdom	16.50	LUNCH BAG RED RETROSPOT

[3]:
```
# Get the shape of the data.

data_1.shape
```

[3]: (15206, 9)

[4]:
```
# Get the data types for every column in the DataFrame.

data_1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15206 entries, 0 to 17031
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Invoice      15206 non-null  object
 1   StockCode    15206 non-null  object
 2   Quantity     15206 non-null  int64
 3   InvoiceDate  15206 non-null  datetime64[ns]
 4   Price        15194 non-null  float64
 5   CustomerID   12435 non-null  object
 6   Country      15206 non-null  object
 7   TotalAmount  15194 non-null  float64
 8   Description  15206 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(5)
memory usage: 1.2+ MB
```

3 Generate summary statistics for all of the data

[5]:
```
# Get a DataFrame of summary statistics that describe the data, including mean,␣
 ↪median, standard deviation, etc.
# Be sure to include all variables, including categorical ones.

data_1.describe(datetime_is_numeric= True, include='all')
```

[5]:

	Invoice	StockCode	Quantity	InvoiceDate
count	15206	15206	15206.000000	15206
unique	8315	10	NaN	NaN
top	536876	85123A	NaN	NaN
freq	10	2163	NaN	NaN
mean	NaN	NaN	16.775483	2011-06-19 06:03:05.279503872
min	NaN	NaN	1.000000	2010-12-01 08:26:00
25%	NaN	NaN	2.000000	2011-03-22 15:31:00
50%	NaN	NaN	6.000000	2011-06-20 12:32:00
75%	NaN	NaN	12.000000	2011-09-23 12:57:30
max	NaN	NaN	4300.000000	2011-12-09 12:31:00
std	NaN	NaN	79.496270	NaN

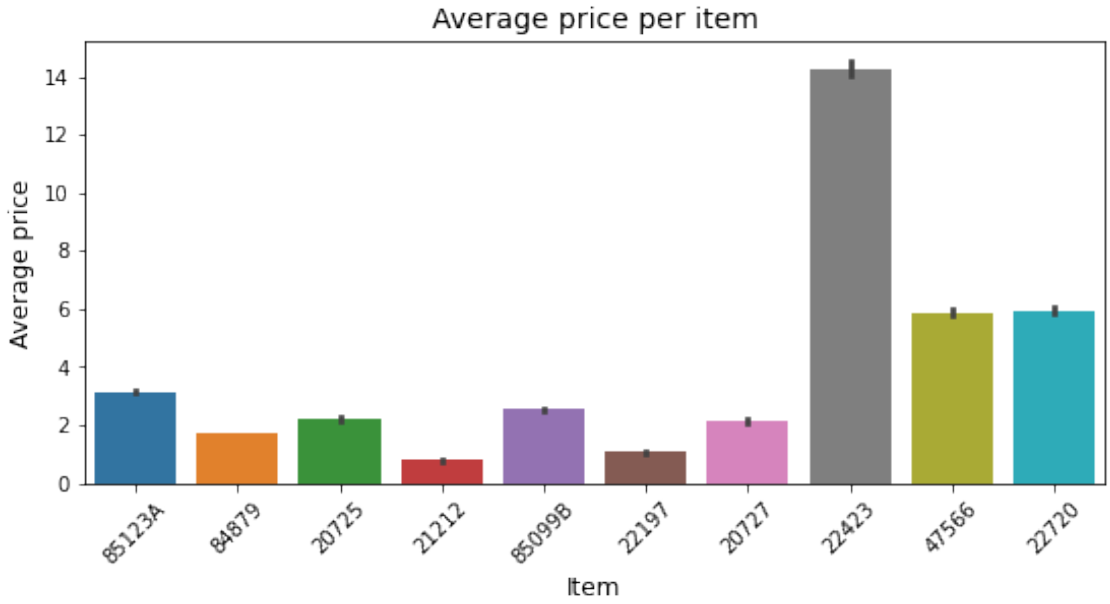
	Price	CustomerID	Country	TotalAmount
count	15194.000000	12435	15206	15194.000000
unique	NaN	2473	1	NaN
top	NaN	u17841	United Kingdom	NaN
freq	NaN	171	15206	NaN
mean	4.164267	NaN	NaN	40.705153
min	0.400000	NaN	NaN	0.550000
25%	1.650000	NaN	NaN	8.850000
50%	2.550000	NaN	NaN	16.500000
75%	4.950000	NaN	NaN	30.360000
max	32.040000	NaN	NaN	4921.500000
std	4.377605	NaN	NaN	132.142503

	Description
count	15206
unique	10
top	CREAM HANGING HEART T-LIGHT HOLDER
freq	2163
mean	NaN
min	NaN
25%	NaN
50%	NaN
75%	NaN
max	NaN
std	NaN

4 Plot a bar chart for the average price per item

[6]:
```
# Plot the average price per item using a bar chart.
# Make sure the average price is on one axis, and each distinct item␣
 ↪description is on the other axis.


plt.figure(figsize=(9,4))
sns.barplot(data=data_1, x='StockCode', y='Price')
plt.xticks(rotation=45)
plt.title('Average price per item', fontsize=14)
plt.xlabel('Item', fontsize=12)
plt.ylabel('Average price', fontsize=12)
```

[6]:
```
Text(0, 0.5, 'Average price')
```



5 Explore the distribution of the numeric variable `Price`

[7]:
```
# Get a DataFrame of summary statistics for numeric variables only.

data_1.describe()
```

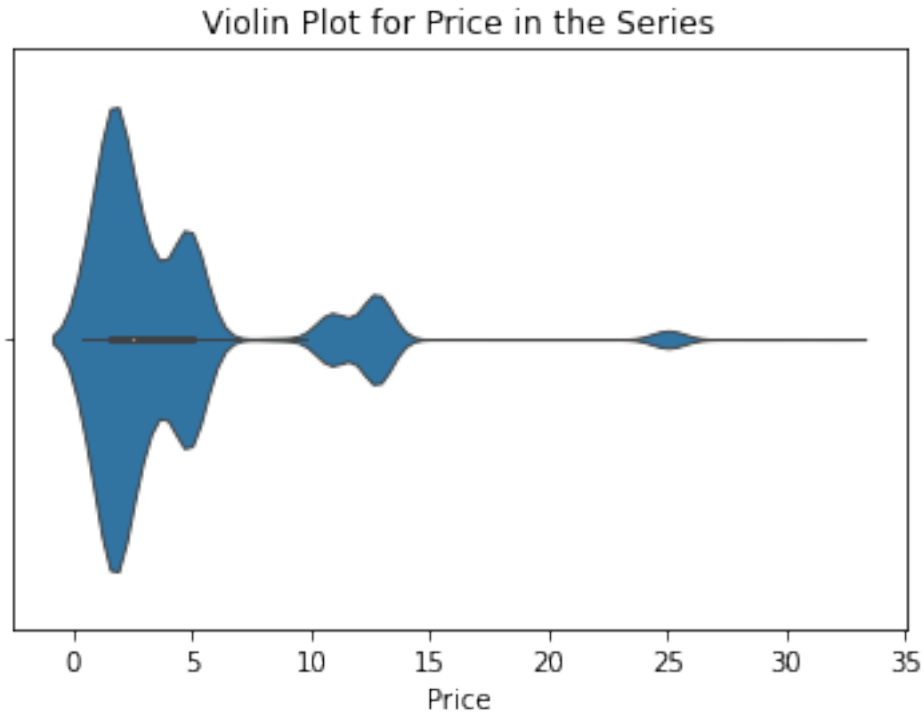
[7]:

	Quantity	Price	TotalAmount
count	15206.000000	15194.000000	15194.000000

mean	16.775483	4.164267	40.705153
std	79.496270	4.377605	132.142503
min	1.000000	0.400000	0.550000
25%	2.000000	1.650000	8.850000
50%	6.000000	2.550000	16.500000
75%	12.000000	4.950000	30.360000
max	4300.000000	32.040000	4921.500000

[8]:
```
# Generate a violin plot for the "Price" variable.
# Decorate and style the plot however you think is best.

sns.violinplot(x=data_1['Price'], linewidth = 0.9);
plt.title('Violin Plot for Price in the Series')
```

[8]: Text(0.5, 1.0, 'Violin Plot for Price in the Series')



6 Visualize correlations between numeric variables

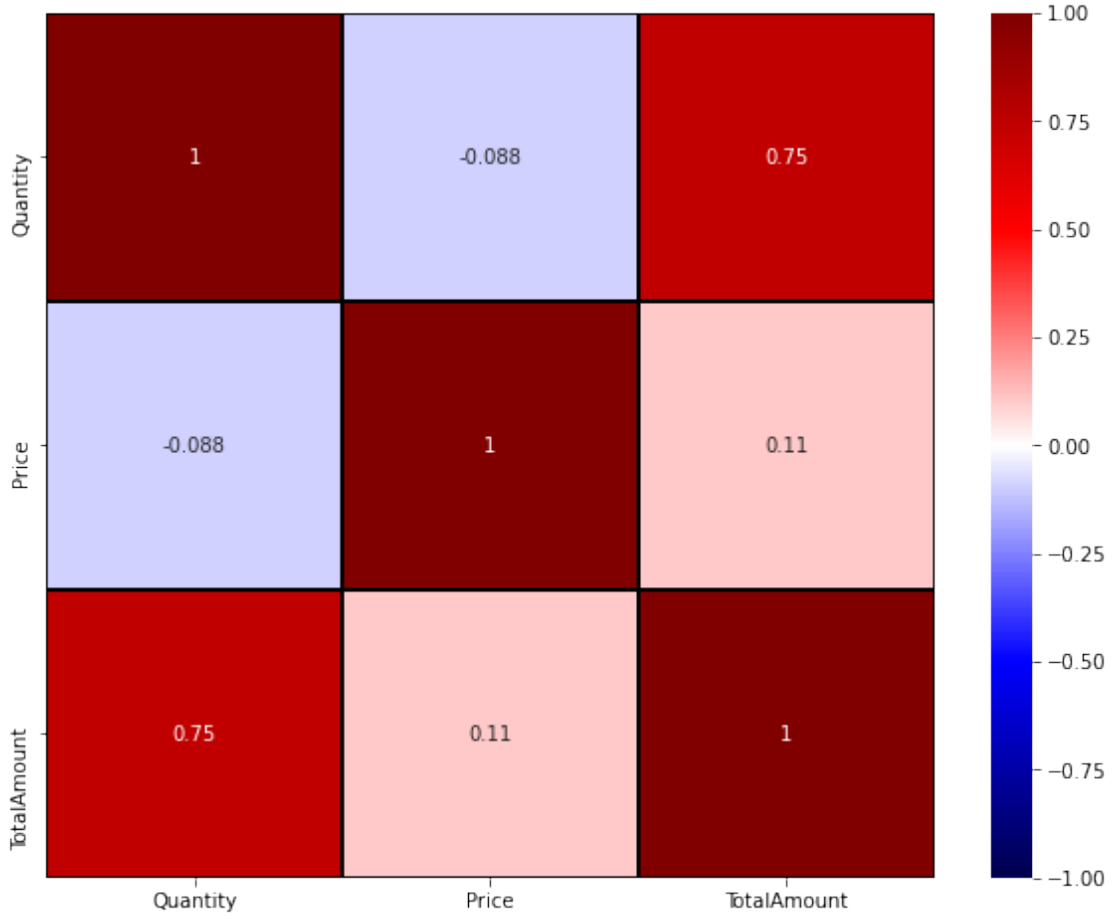
[9]:
```
# Generate a correlation matrix between all numeric variables.

data_corr = data_1.corr()
```

```
data_corr
```

[9]:

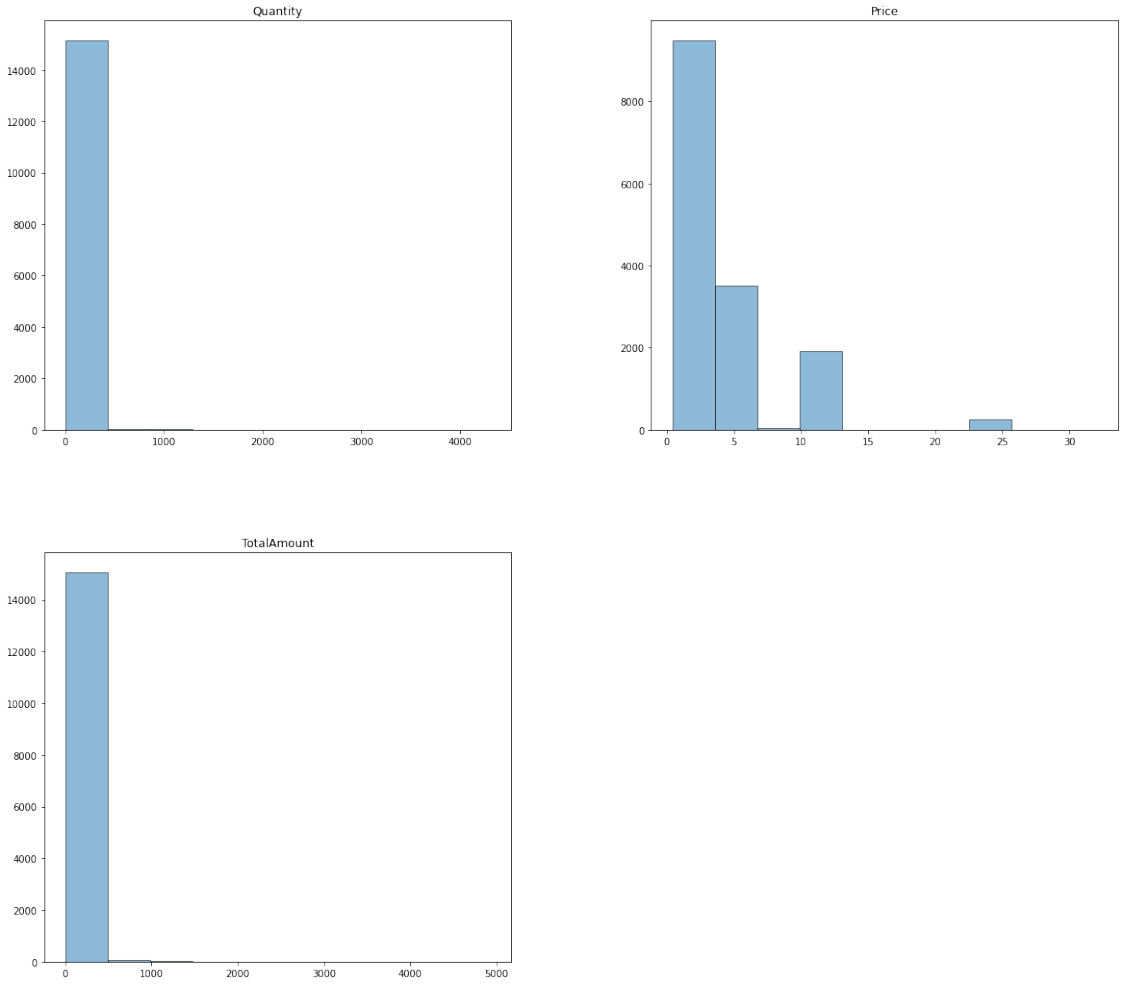
	Quantity	Price	TotalAmount
Quantity	1.000000	-0.088356	0.745641
Price	-0.088356	1.000000	0.109054
TotalAmount	0.745641	0.109054	1.000000

[10]:
```
# Visualize the correlations with a heatmap.

fig = plt.figure(figsize = (10,8))

sns.heatmap(data_corr,
            cmap = 'seismic',
            linewidth = 0.75,
            linecolor = 'black',
            cbar = True,
            vmin = -1,
            vmax = 1,
            annot = True);
```

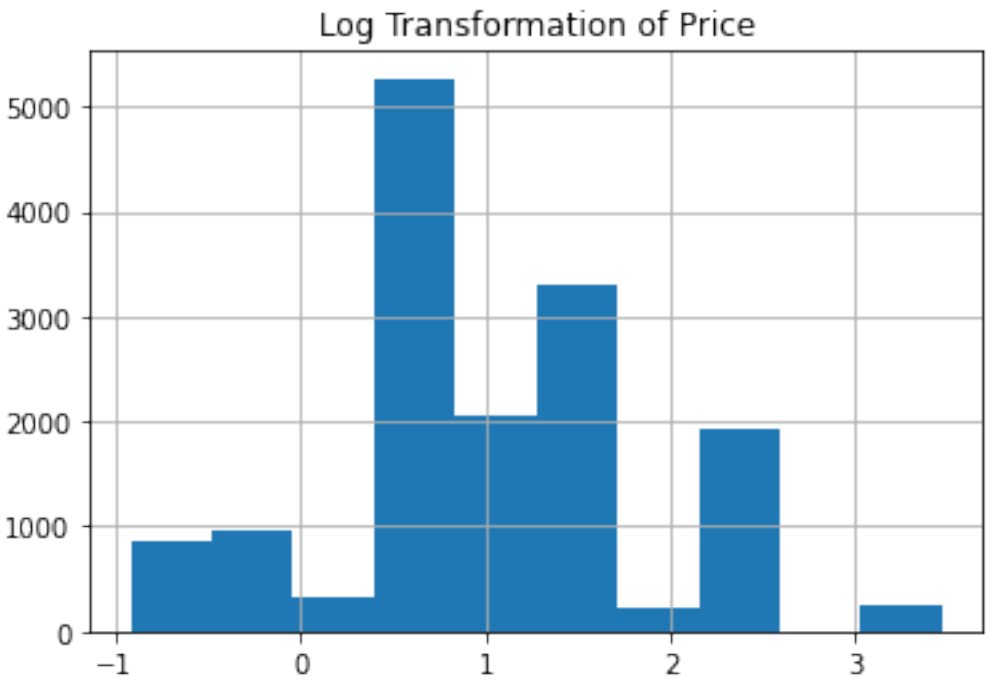


7 Transform skewed variables

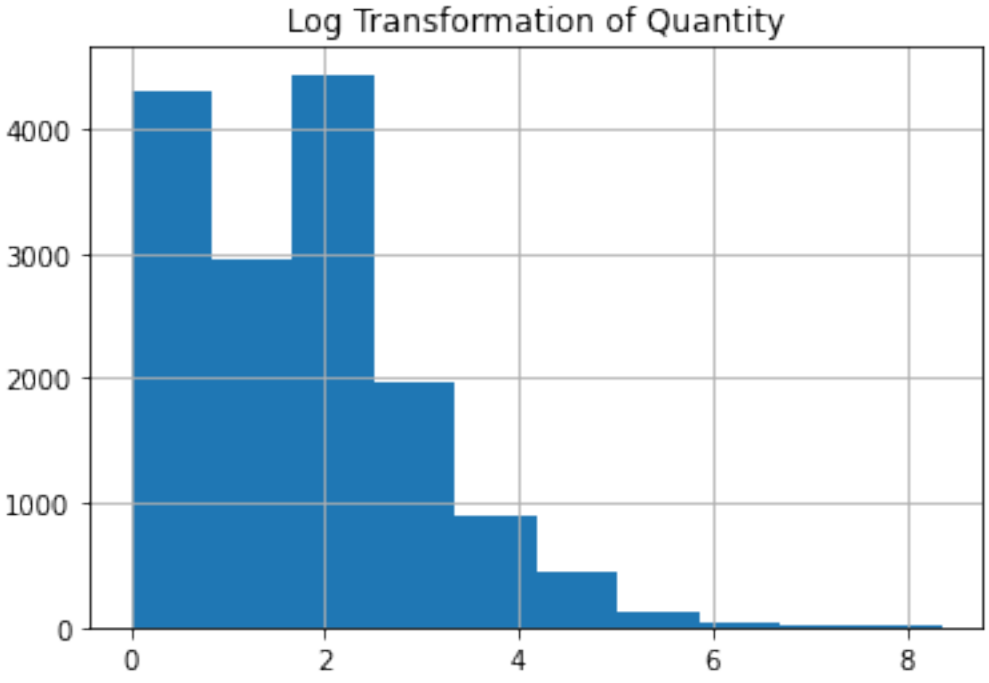
[11]:
```
# Plot histograms for the original distributions of all numeric variables.

data_1.hist(figsize = (20,18), alpha = 0.5,
            edgecolor = 'black', grid = False);
```



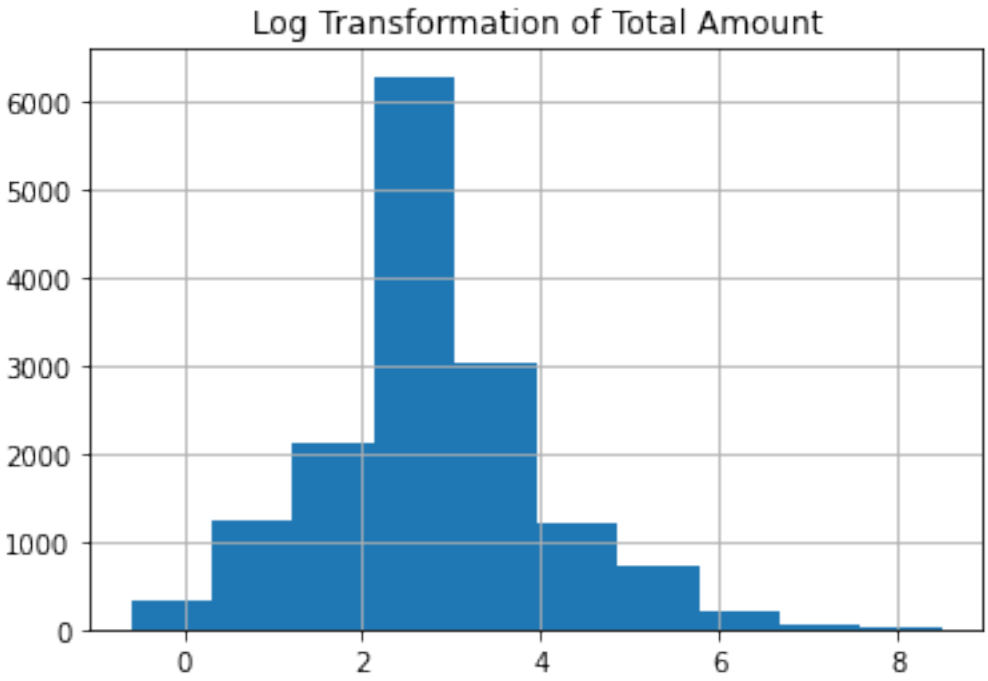
[12]:
```
# Plot the log transformation of "Price".

np.log(data_1['Price']).hist()
plt.title('Log Transformation of Price');
```



[13]:
```
# Plot the log transformation of "Quantity".

np.log(data_1['Quantity']).hist()
plt.title('Log Transformation of Quantity');
```



[14]:
```
# Plot the log transformation of "TotalAmount".

np.log(data_1['TotalAmount']).hist()
plt.title('Log Transformation of Total Amount');
```



8 Analyze time series data

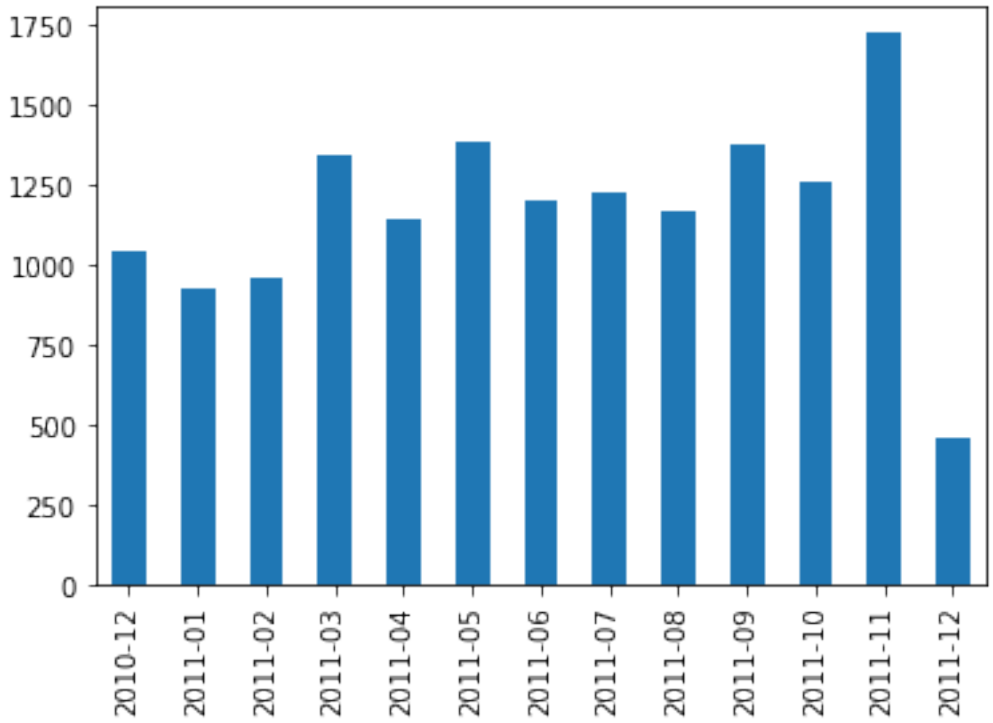
[15]:
```
# Obtain the number of invoices by month.

time_series = data_1.InvoiceDate.dt.strftime('%Y-%m').value_counts().
 ↪sort_index()

#list_month.sort([])
time_series
```

[15]:
```
2010-12    1042
2011-01     923
2011-02     956
2011-03    1345
2011-04    1140
```

```
2011-05    1384
2011-06    1203
2011-07    1227
2011-08    1169
2011-09    1378
2011-10    1257
2011-11    1726
2011-12     456
Name: InvoiceDate, dtype: int64
```

[16]:
```
# Use a bar chart to plot the number of invoices by month.

time_series.plot(kind='bar')
```

[16]: <AxesSubplot:>



9 Identify and handle missing data

[17]:
```
# Identify any missing data for all variables.

data_1.isnull().sum()
```

[17]:
```
Invoice           0
StockCode         0
Quantity          0
InvoiceDate       0
Price            12
CustomerID     2771
Country           0
TotalAmount      12
Description       0
dtype: int64
```

[18]:
```
# Print the current shape of the data.

print('Shape of data: {}'.format(data_1.shape))

# Remove rows of data where "CustomerID" is unknown.

data_clean_id = data_1.dropna(subset=['CustomerID'])

# Print the new shape of the data.

print('Shape new data: {}'.format(data_clean_id.shape))
```

```
Shape of data: (15206, 9)
Shape new data: (12435, 9)
```

[19]:
```
# Fill in N/A values for "Price" and "TotalAmount" with 0.

data_clean_id['Price'].fillna(0, inplace=True)
data_clean_id['TotalAmount'].fillna(0, inplace=True)

# Confirm there are no longer any missing values.
#print('Shape new data: {}'.format(data_clean.shape))

data_clean_id.isnull().sum()
```

[19]:
```
Invoice        0
StockCode      0
Quantity       0
InvoiceDate    0
Price          0
CustomerID     0
Country        0
TotalAmount    0
Description    0
dtype: int64
```

10 One-hot encode the `Description` variable

[20]:
```
data_clean_id.Description.value_counts()
encoder = ce.OneHotEncoder( cols = 'Description',
                           return_df = True,
                           use_cat_names = True)

# Concatenate the new encoded columns with the main DataFrame.

#data_encoded_final = pd.concat([data_clean_id, data_encoded], axis = 1)
data_encoded_final = encoder.fit_transform(data_clean_id)

# Drop the original "Description" variable.

#data_encoded_final.drop(['Description'], axis = 1, inplace = True)
```

[21]:
```
# One-hot encode the "Description" variable with dummy variables for each␣
 ↪unique description.
# Prefix each dummy variable name with "Description".

data_encoded = pd.get_dummies(data=data_clean_id['Description'], drop_first =␣
 ↪True)

# Preview the first five rows of the DataFrame.
data_encoded.head(n=5)
```

[21]:

	CREAM HANGING HEART T-LIGHT HOLDER	JUMBO BAG RED RETROSPOT	\
0	1	0	
1	0	0	
2	1	0	
3	1	0	
4	0	0	

	LUNCH BAG	BLACK SKULL.	LUNCH BAG RED RETROSPOT	\
0	0	0		
1	0	0		
2	0	0		
3	0	0		
4	0	1		

	PACK OF 72 RETROSPOT CAKE CASES	PARTY BUNTING	POPCORN HOLDER	\
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	0	
4	0	0	0	

```
   REGENCY CAKESTAND 3 TIER  SET OF 3 CAKE TINS PANTRY DESIGN
0                         0                                 0
1                         0                                 0
2                         0                                 0
3                         0                                 0
4                         0                                 0
```

[22]:
```
# Preview the first five rows of the data.

data_encoded_final.head(n=5)
```

[22]:
```
  Invoice StockCode  Quantity         InvoiceDate  Price CustomerID  \
0  536365    85123A         6 2010-12-01 08:26:00   2.55      u1785
1  536367     84879        32 2010-12-01 08:34:00   1.69     u13047
2  536373    85123A         6 2010-12-01 09:02:00   2.55      u1785
3  536375    85123A         6 2010-12-01 09:32:00   2.55      u1785
4  536378     20725        10 2010-12-01 09:37:00   1.65     u14688

         Country  TotalAmount  \
0  United Kingdom        15.30
1  United Kingdom        54.08
2  United Kingdom        15.30
3  United Kingdom        15.30
4  United Kingdom        16.50

   Description_CREAM HANGING HEART T-LIGHT HOLDER  \
0                                               1
1                                               0
2                                               1
3                                               1
4                                               0

   Description_ASSORTED COLOUR BIRD ORNAMENT  \
0                                          0
1                                          1
2                                          0
3                                          0
4                                          0

   Description_LUNCH BAG RED RETROSPOT  \
0                                    0
1                                    0
2                                    0
3                                    0
4                                    1
```

```
   Description_PACK OF 72 RETROSPOT CAKE CASES  \
0                                            0
1                                            0
2                                            0
3                                            0
4                                            0

   Description_JUMBO BAG RED RETROSPOT  Description_POPCORN HOLDER  \
0                                    0                           0
1                                    0                           0
2                                    0                           0
3                                    0                           0
4                                    0                           0

   Description_LUNCH BAG  BLACK SKULL.  Description_REGENCY CAKESTAND 3 TIER  \
0                                    0                                     0
1                                    0                                     0
2                                    0                                     0
3                                    0                                     0
4                                    0                                     0

   Description_PARTY BUNTING  Description_SET OF 3 CAKE TINS PANTRY DESIGN
0                          0                                             0
1                          0                                             0
2                          0                                             0
3                          0                                             0
4                          0                                             0
```

11 Identify and remove columns with low variance

[23]:
```
# Obtain the standard deviation of each variable.
data_encoded_final.std()
```

[23]:
```
Quantity                                            77.436253
Price                                                3.506381
TotalAmount                                        132.092738
Description_CREAM HANGING HEART T-LIGHT HOLDER       0.362575
Description_ASSORTED COLOUR BIRD ORNAMENT            0.308044
Description_LUNCH BAG RED RETROSPOT                  0.288011
Description_PACK OF 72 RETROSPOT CAKE CASES          0.254417
Description_JUMBO BAG RED RETROSPOT                  0.322017
Description_POPCORN HOLDER                           0.267557
Description_LUNCH BAG  BLACK SKULL.                  0.274661
Description_REGENCY CAKESTAND 3 TIER                 0.317766
Description_PARTY BUNTING                            0.306081
```

```
Description_SET OF 3 CAKE TINS PANTRY DESIGN        0.273557
dtype: float64
```

[24]:
```
# Define a standard deviation threshold of 0.26.

threshold = 0.26

# Identify any columns that are lower than the threshold.

cols_to_drop = list(data_encoded_final.std()[data_encoded_final.std() <␣
↪threshold].index.values)

# Print the column(s) that will be dropped.

print('Cols will be dropped: ', cols_to_drop)
```

```
Cols will be dropped:  ['Description_PACK OF 72 RETROSPOT CAKE CASES']
```

[25]:
```
# Drop the column(s) that have low standard deviation from the main dataset.

data_encoded_droped = data_encoded_final.drop(cols_to_drop, axis = 1)

# Preview the first five rows of data.

data_encoded_droped.head(n=5)
```

[25]:

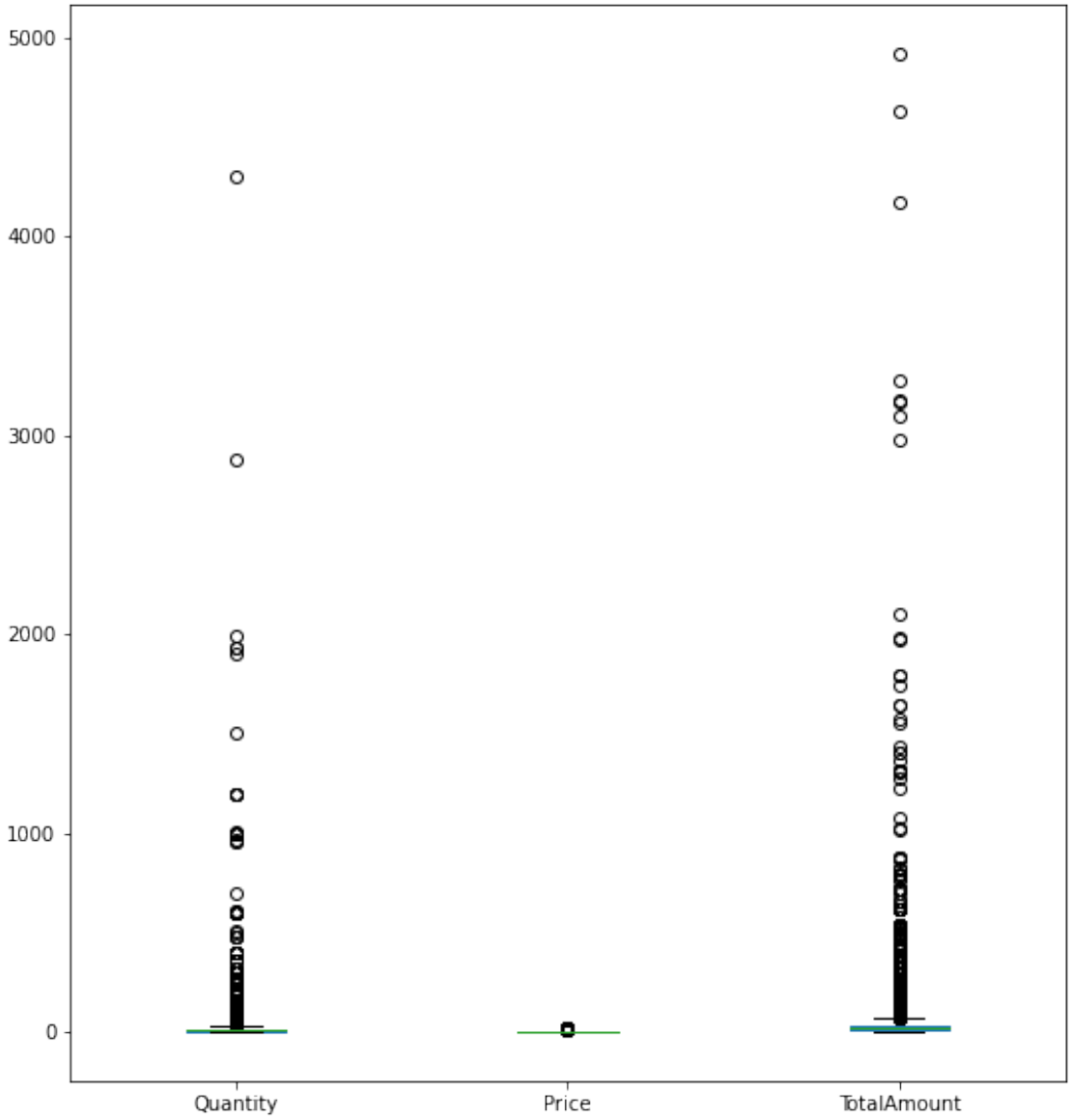
	Invoice	StockCode	Quantity	InvoiceDate	Price	CustomerID	Country	TotalAmount	Description_CREAM HANGING HEART T-LIGHT HOLDER
0	536365	85123A	6	2010-12-01 08:26:00	2.55	u1785	United Kingdom	15.30	1
1	536367	84879	32	2010-12-01 08:34:00	1.69	u13047	United Kingdom	54.08	0
2	536373	85123A	6	2010-12-01 09:02:00	2.55	u1785	United Kingdom	15.30	1
3	536375	85123A	6	2010-12-01 09:32:00	2.55	u1785	United Kingdom	15.30	1
4	536378	20725	10	2010-12-01 09:37:00	1.65	u14688	United Kingdom	16.50	0

```
   Description_ASSORTED COLOUR BIRD ORNAMENT  \
0                                          0
1                                          1
2                                          0
3                                          0
4                                          0

   Description_LUNCH BAG RED RETROSPOT  Description_JUMBO BAG RED RETROSPOT  \
0                                    0                                    0
1                                    0                                    0
2                                    0                                    0
3                                    0                                    0
4                                    1                                    0

   Description_POPCORN HOLDER  Description_LUNCH BAG  BLACK SKULL.  \
0                           0                                    0
1                           0                                    0
2                           0                                    0
3                           0                                    0
4                           0                                    0

   Description_REGENCY CAKESTAND 3 TIER  Description_PARTY BUNTING  \
0                                     0                           0
1                                     0                           0
2                                     0                           0
3                                     0                           0
4                                     0                           0

   Description_SET OF 3 CAKE TINS PANTRY DESIGN
0                                             0
1                                             0
2                                             0
3                                             0
4                                             0
```

12 Generate box plots for each numeric variable

[26]:
```
# Draw box plots for each numeric variable.

cols = ['Quantity', 'Price', 'TotalAmount']

data_encoded_droped[cols].plot(kind = 'box', figsize = (9,10))
```

[26]: <AxesSubplot:>



13 Identify and remove outliers

[27]:
```
# This function returns the lower and upper bounds of a numeric input variable.

def calc_outliers(var):
    q3 = np.percentile(var, 75)
    q1 = np.percentile(var, 25)

    iqr = 1.5 * (q3 - q1)
```

```
    lb = q1 - iqr
    ub = q3 + iqr

    print('Lower bound of outliers:', round(lb, 2), '\nUpper bound of outliers:
 ↪', round(ub, 2))

    return lb, ub
```

[28]:
```
# Identify the shape of the data before removing outliers.

data_encoded_droped.shape
```

[28]:
```
(12435, 17)
```

[29]:
```
# Call the calc_outliers() function iteratively for each numeric variable.
# For each variable:
#     Remove the outliers that are higher than the upper bounds.
#     Remove the variables that are lower than the lower bounds.
# As you iterate through each variable, print the shape of the data after the␣
 ↪outliers for that variable are removed.


for col in cols:

    print(f'Column = {col}')
    lb, ub = calc_outliers(data_encoded_droped[col])

    data_final = data_encoded_droped[(data_encoded_droped[col]>lb) &␣
 ↪(data_encoded_droped[col]<ub)]

    print(f'After removing outliers data shape = {data_final.shape}')
```

```
Column = Quantity
Lower bound of outliers: -15.0
Upper bound of outliers: 33.0
After removing outliers data shape = (11479, 17)
Column = Price
Lower bound of outliers: -3.3
Upper bound of outliers: 9.9
After removing outliers data shape = (11010, 17)
Column = TotalAmount
Lower bound of outliers: -23.8
Upper bound of outliers: 66.07
After removing outliers data shape = (10936, 17)
```

14 Save the final dataset as a pickle file

[30]:
```
# Save the final dataset as a pickle file named online_history_cleaned_final.
↪pickle.
data_final.to_pickle('online_history_cleaned_final.pickle')
```

[ ]:
```

```