S_NITHISHKUMAR_EDA

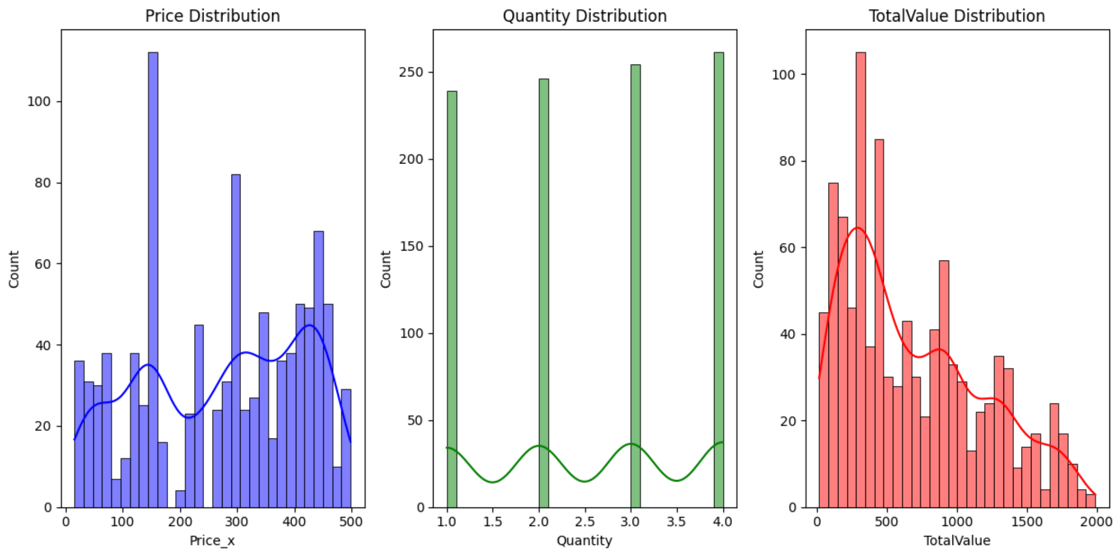
January 27, 2025

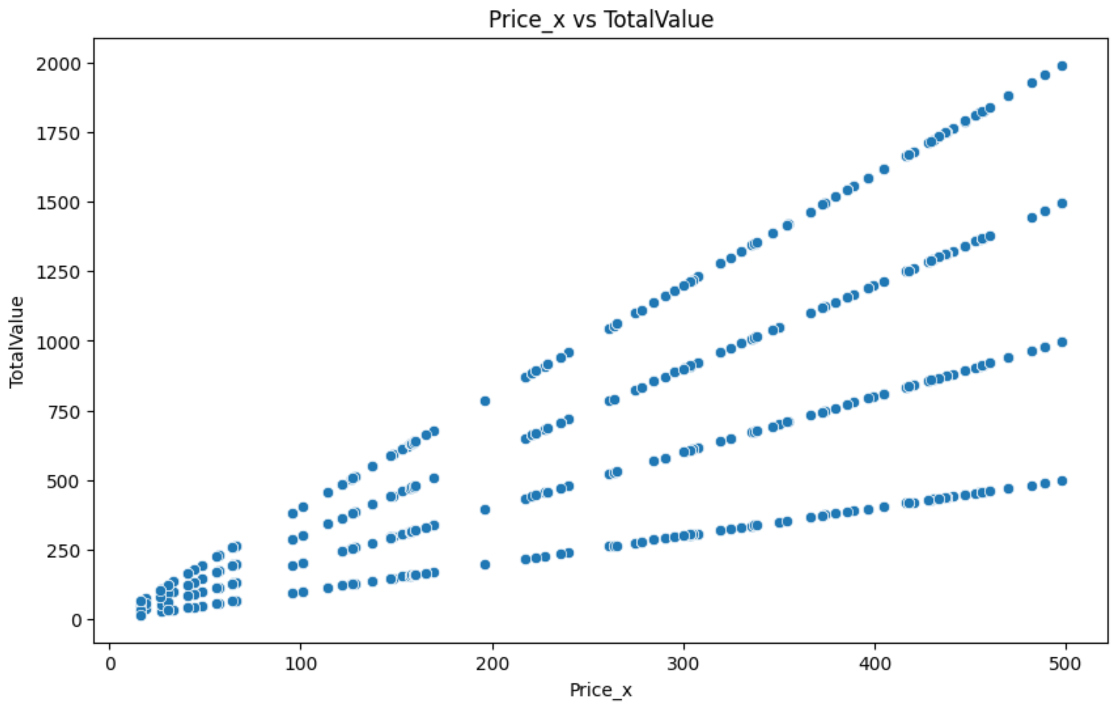
[1]:
```python
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load datasets
customers = pd.read_csv("/content/drive/My Drive/Zeotap/Customers.csv")
products = pd.read_csv("/content/drive/My Drive/Zeotap/Products.csv")
transactions = pd.read_csv("/content/drive/My Drive/Zeotap/Transactions.csv")

# Convert date columns to datetime
customers['SignupDate'] = pd.to_datetime(customers['SignupDate'],␣
 ↪format='%d-%m-%Y', errors='coerce')
transactions['TransactionDate'] = pd.
 ↪to_datetime(transactions['TransactionDate'], format='%d-%m-%Y %H:%M',␣
 ↪errors='coerce')

# Merge datasets on relevant columns
data = transactions.merge(customers, on='CustomerID', how='left').
 ↪merge(products, on='ProductID', how='left')

# 1. Data Overview
print("Data Info:")
print(data.info())  # Check for missing values and data types
print("\nSummary Statistics:")
print(data.describe())  # Summary statistics for numerical columns

# 2. Univariate Analysis: Plot distributions for numerical features
plt.figure(figsize=(12, 6))

# Distribution of Price_x
plt.subplot(1, 3, 1)
sns.histplot(data['Price_x'], kde=True, bins=30, color='blue')
plt.title('Price Distribution')

# Distribution of Quantity
plt.subplot(1, 3, 2)
```

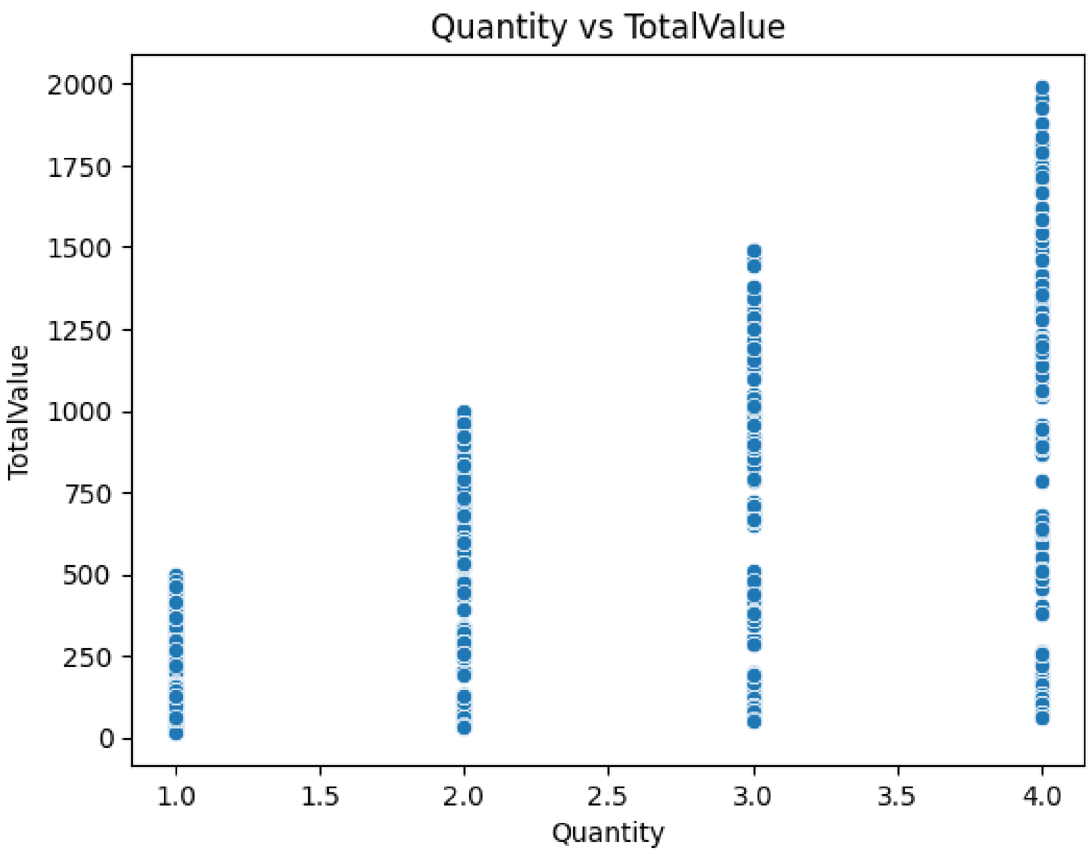
```
sns.histplot(data['Quantity'], kde=True, bins=30, color='green')
plt.title('Quantity Distribution')

# Distribution of TotalValue
plt.subplot(1, 3, 3)
sns.histplot(data['TotalValue'], kde=True, bins=30, color='red')
plt.title('TotalValue Distribution')

plt.tight_layout()
plt.show()

# 3. Bivariate Analysis: Price vs Quantity, TotalValue vs Price, etc.
plt.figure(figsize=(10, 6))

# Scatter plot of Price_x vs TotalValue
sns.scatterplot(x='Price_x', y='TotalValue', data=data)
plt.title('Price_x vs TotalValue')
plt.show()

# Scatter plot of Quantity vs TotalValue
sns.scatterplot(x='Quantity', y='TotalValue', data=data)
plt.title('Quantity vs TotalValue')
plt.show()

# 4. Correlation Heatmap between numerical features
correlation_matrix = data[['Price_x', 'Quantity', 'TotalValue']].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()

# 5. Outlier Detection: Boxplots
plt.figure(figsize=(12, 6))

# Boxplot of Price_x
plt.subplot(1, 3, 1)
sns.boxplot(x=data['Price_x'])
plt.title('Price_x Outliers')

# Boxplot of Quantity
plt.subplot(1, 3, 2)
sns.boxplot(x=data['Quantity'])
plt.title('Quantity Outliers')

# Boxplot of TotalValue
plt.subplot(1, 3, 3)
sns.boxplot(x=data['TotalValue'])
```

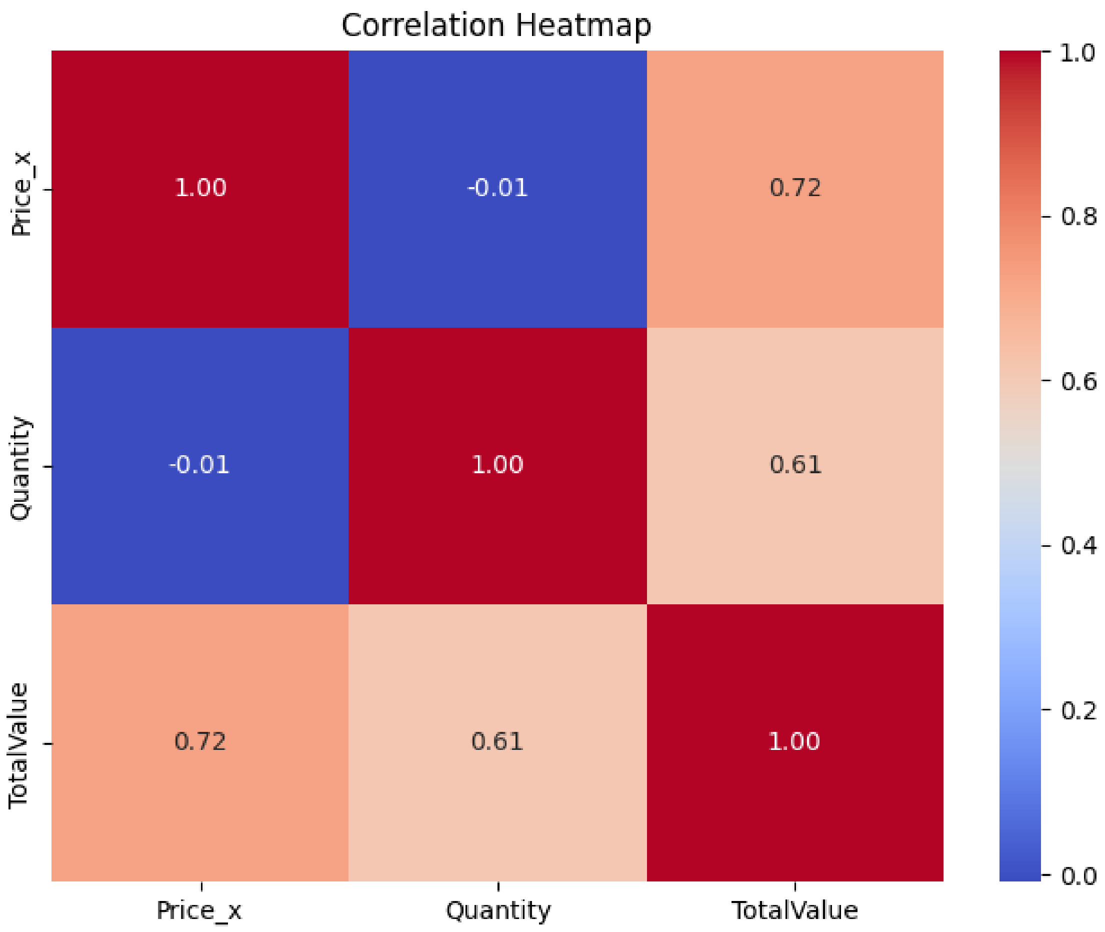
```python
plt.title('TotalValue Outliers')

plt.tight_layout()
plt.show()

# 6. Categorical Features Analysis: Region and Category
plt.figure(figsize=(10, 5))

# Countplot of Regions
sns.countplot(x='Region', data=data)
plt.title('Distribution of Transactions by Region')
plt.xticks(rotation=45)
plt.show()

# Countplot of Categories
sns.countplot(x='Category', data=data)
plt.title('Distribution of Transactions by Category')
plt.xticks(rotation=45)
plt.show()

# 7. Create TransactionMonth column for Peak Months for Transactions
data['TransactionMonth'] = data['TransactionDate'].dt.month

# Peak Months for Transactions
sns.countplot(x='TransactionMonth', data=data)
plt.title('Distribution of Transactions by Month')
plt.xlabel('Month')
plt.ylabel('Transaction Count')
plt.show()

# 8. Insights on Relationship Between TotalValue and Features
sns.barplot(x='Region', y='TotalValue', data=data)
plt.title('TotalValue by Region')
plt.show()

sns.barplot(x='Category', y='TotalValue', data=data)
plt.title('TotalValue by Category')
plt.xticks(rotation=45)
plt.show()
```

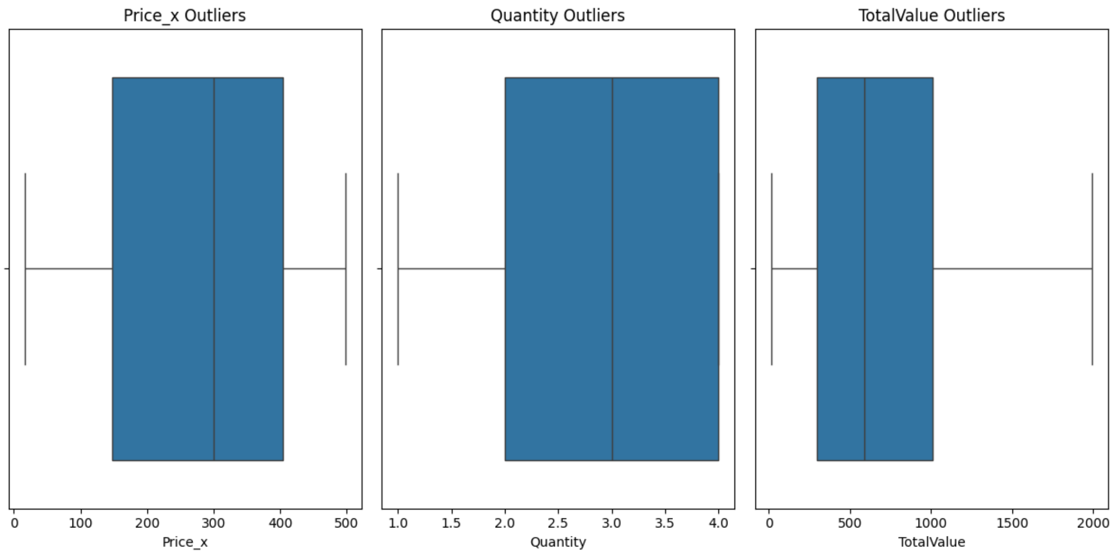
```
Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   TransactionID     1000 non-null   object
 1   CustomerID       1000 non-null   object
 2   ProductID        1000 non-null   object
 3   TransactionDate  0 non-null      datetime64[ns]
 4   Quantity         1000 non-null   int64
 5   TotalValue       1000 non-null   float64
 6   Price_x          1000 non-null   float64
 7   CustomerName     1000 non-null   object
 8   Region           1000 non-null   object
 9   SignupDate       1000 non-null   datetime64[ns]
 10  ProductName      1000 non-null   object
 11  Category         1000 non-null   object
 12  Price_y          1000 non-null   float64
dtypes: datetime64[ns](2), float64(3), int64(1), object(7)
memory usage: 101.7+ KB
None

Summary Statistics:
      TransactionDate     Quantity   TotalValue     Price_x  \
count               0  1000.000000  1000.000000  1000.00000
mean              NaT     2.537000   689.995560   272.55407
min               NaT     1.000000    16.080000    16.08000
25%               NaT     2.000000   295.295000   147.95000
50%               NaT     3.000000   588.880000   299.93000
75%               NaT     4.000000  1011.660000   404.40000
max               NaT     4.000000  1991.040000   497.76000
std               NaN     1.117981   493.144478   140.73639

                          SignupDate     Price_y
count                           1000  1000.00000
mean   2023-07-09 02:49:55.199999744   272.55407
min              2022-01-22 00:00:00    16.08000
25%              2022-09-17 12:00:00   147.95000
50%              2023-07-23 00:00:00   299.93000
75%              2024-04-12 00:00:00   404.40000
max              2024-12-28 00:00:00   497.76000
std                              NaN   140.73639
```

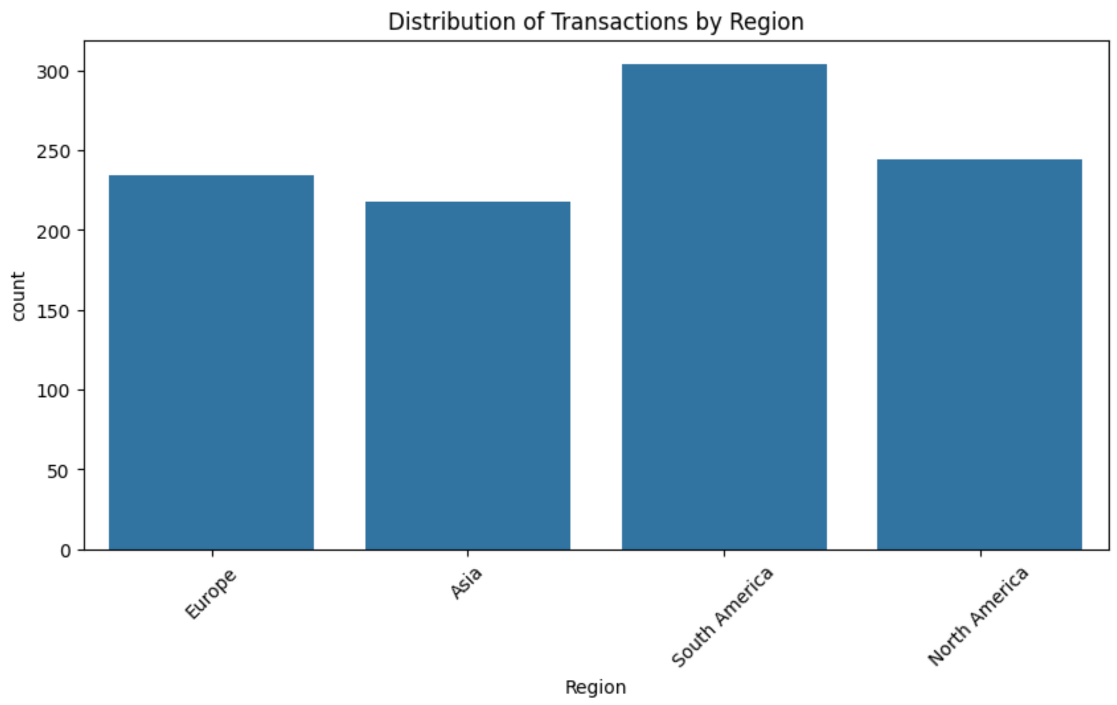

Price Distribution
Quantity Distribution
TotalValue Distribution
Count
Price_x
Quantity
TotalValue

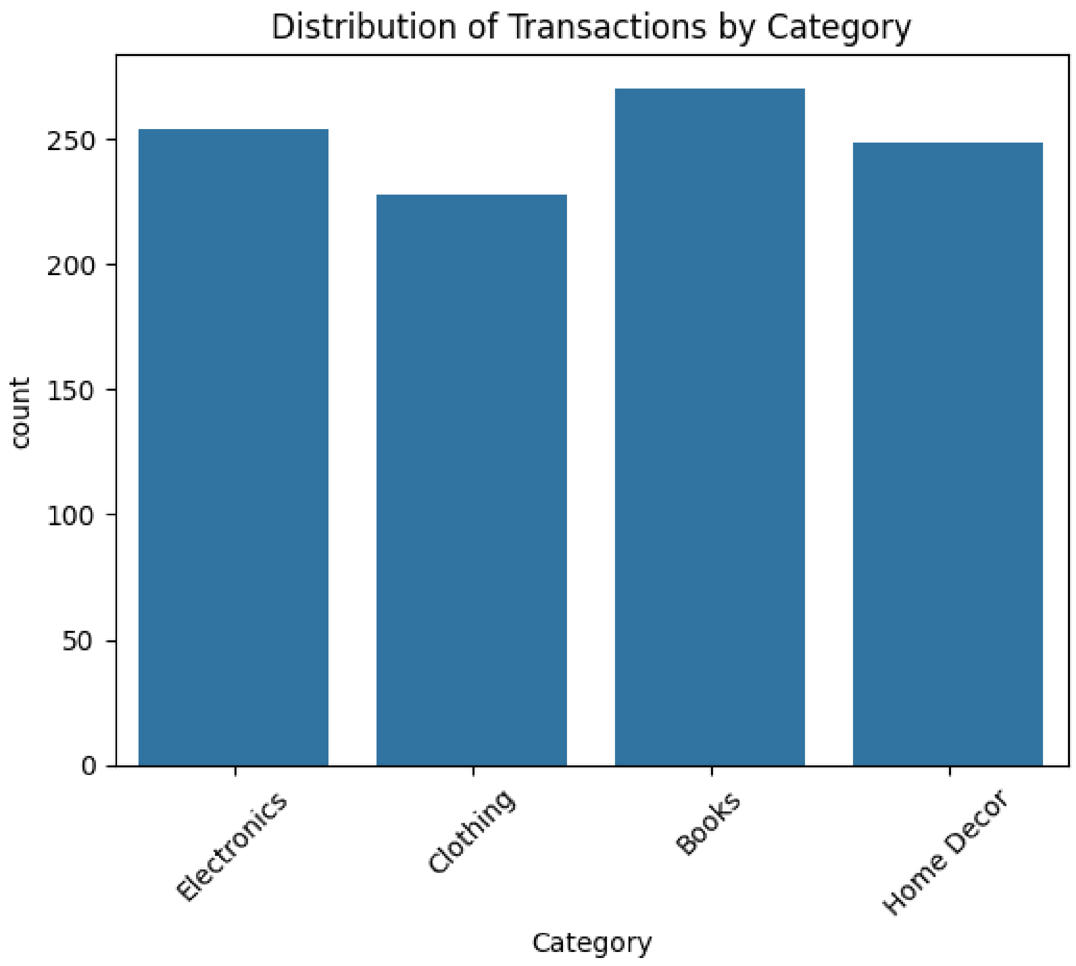

Price_x vs TotalValue
TotalValue
Price_x

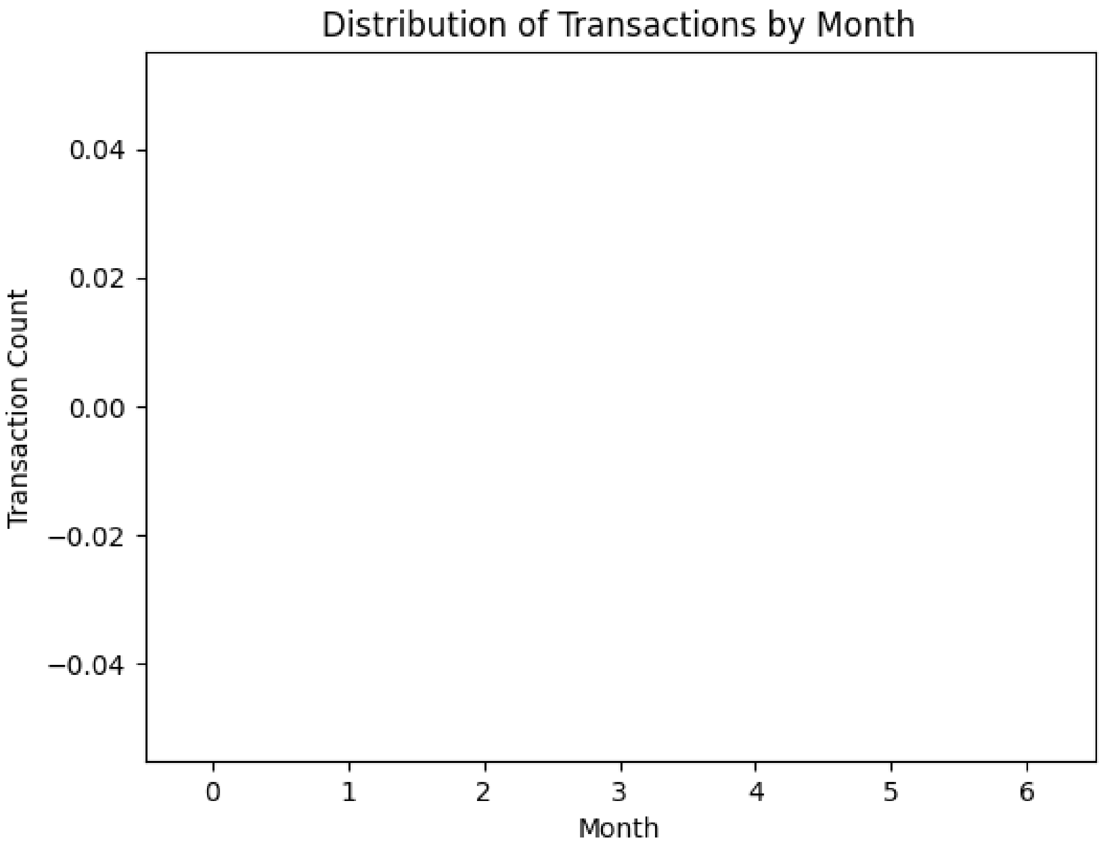

Quantity vs TotalValue
TotalValue
Quantity
2000
1750
1500
1250
1000
750
500
250
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0

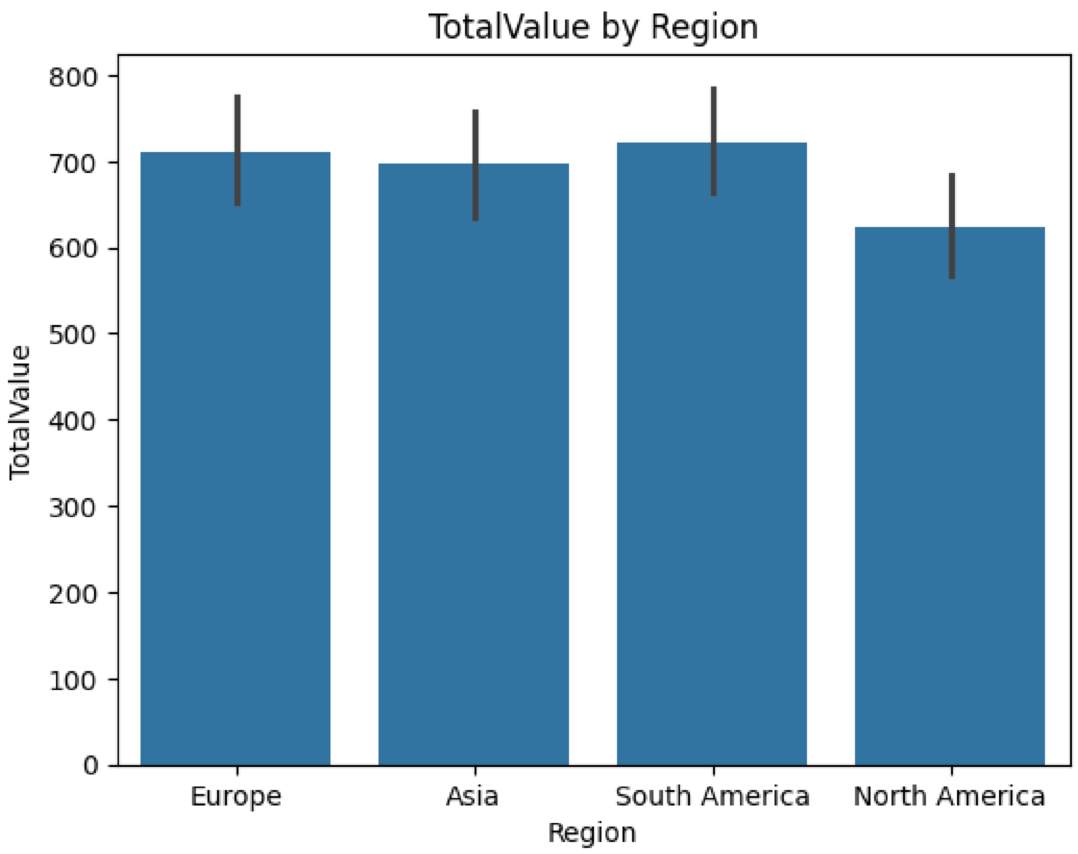

Correlation Heatmap
Price_x
Quantity
TotalValue
1.00
-0.01
0.72
-0.01
1.00
0.61
0.72
0.61
1.00
1.0
0.8
0.6
0.4
0.2
0.0

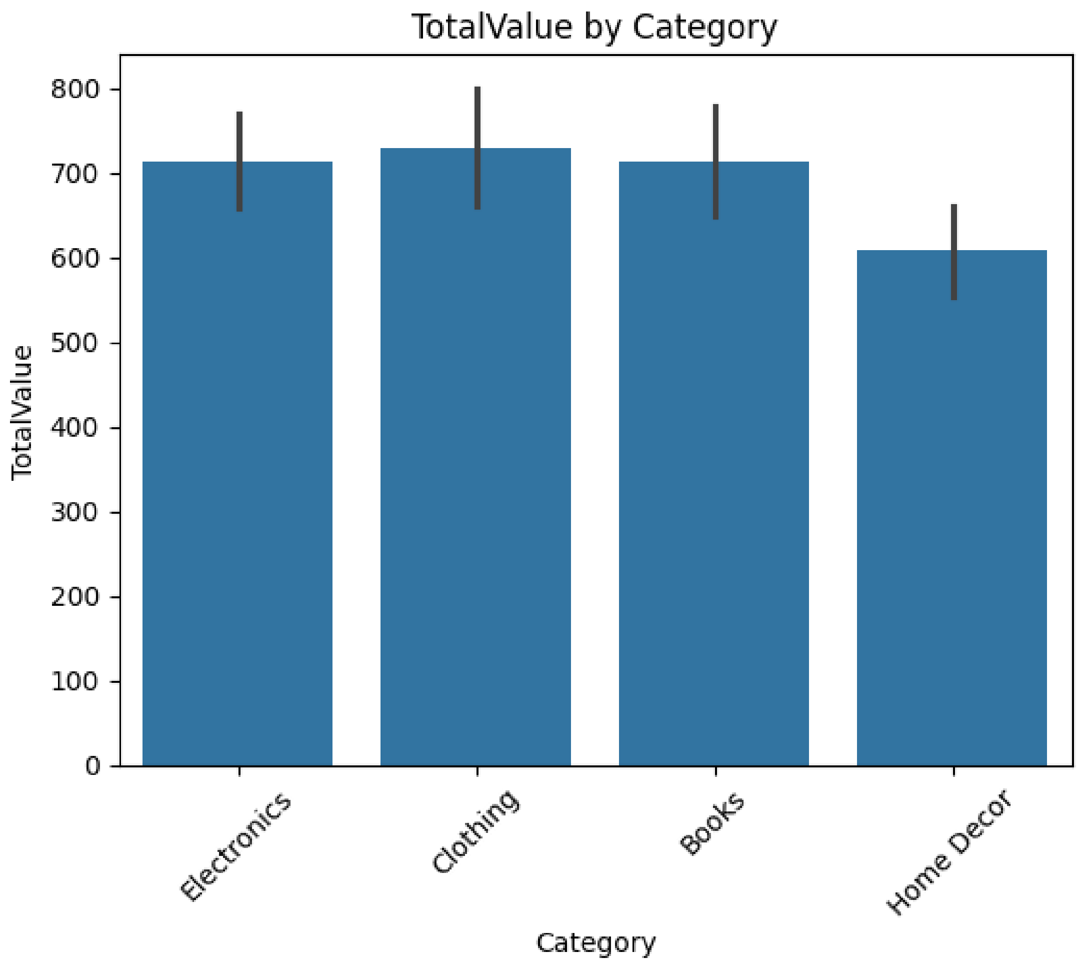

Price_x Outliers
Quantity Outliers
TotalValue Outliers
0
100
200
300
400
500
Price_x
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Quantity
0
500
1000
1500
2000
TotalValue


Distribution of Transactions by Region
count
300
250
200
150
100
50
0
Europe
Asia
South America
North America
Region


Distribution of Transactions by Category
count
250
200
150
100
50
0
Electronics
Clothing
Books
Home Decor
Category


Distribution of Transactions by Month
Transaction Count
0.04
0.02
0.00
−0.02
−0.04
0
1
2
3
4
5
6
Month


TotalValue by Region
TotalValue
Region
Europe
Asia
South America
North America
0
100
200
300
400
500
600
700
800



[2]:
```
!jupyter nbconvert "/content/drive/My Drive/Zeotap/S_NITHISHKUMAR_EDA.ipynb"␣
↪--to pdf
```

```
[NbConvertApp] WARNING | pattern '/content/drive/My
Drive/Zeotap/S_NITHISHKUMAR_EDA.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all
```

```
--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and
include the error message in the cell output (the default behaviour is to abort
conversion). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with
default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
            relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
            overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--ClearOutputPreprocessor.enabled=True]
--coalesce-streams
    Coalesce consecutive stdout and stderr outputs into one stream (within each
cell).
    Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--CoalesceStreamsPreprocessor.enabled=True]
--no-prompt
```

```
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True
--TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
            This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True
--TemplateExporter.exclude_input=True
--TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
    Whether to allow downloading chromium if no suitable version is found on the
system.
    Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
    Disable chromium security sandbox when converting to PDF..
    Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
    Shows code input. This flag is only useful for dejavu users.
    Equivalent to: [--TemplateExporter.exclude_input=False]
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful
for the HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
            ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook',
'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides', 'webpdf']
            or a dotted object name that represents the import path for an
            ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
```

```
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed
    as prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This
    should be set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                    results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                    results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    Overwrite base name use for output files.
                Supports pattern replacements '{notebook_name}'.
    Default: '{notebook_name}'
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                  to output to the directory of each notebook.
To recover
                                  previous default behaviour (outputting to the
current
                                  working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a
copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local
            copy of reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the
            current directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-
html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb --to html

            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown',
'notebook', 'pdf', 'python', 'qtpdf', 'qtpng', 'rst', 'script', 'slides',
'webpdf'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX
includes
            'base', 'article' and 'report'.  HTML includes 'basic', 'lab' and
            'classic'. You can specify the flavor of the format used.

            > jupyter nbconvert --to html --template lab mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing::

                c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```

[ ]:

```

```