UK Online Store Retail Transactions

Dataset Variable Information:

1. InvoiceNo: Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.
2. StockCode: Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
3. Description: Product (item) name. Nominal.
4. Quantity: The quantities of each product (item) per transaction. Numeric.
5. InvoiceDate: Invoice Date and time. Numeric, the day and time when each transaction was generated.
6. UnitPrice: Unit price. Numeric, Product price per unit in sterling.
7. CustomerID: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.
8. Country: Country name. Nominal, the name of the country where each customer resides.

Establishing Python Library Packages

Show code

Dataset Overview

Show code

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Ui King
1	536365	71053	WHITE METAL	6	2010-12-01	3.39	17850.0	Ui

Dataset Summary Overview

Show code

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

// Observations

Show code

Observation

- Dataset has 8 columns
- Max Row numbers: 541,909
- "Description" and "CustomerID" have lesser row count; possibly null values
- Description = 540,455 total rows
- CustomerID = 406,829 total rows
- "CustomerID" datatype is float64; convert into str object

.

Show code

CLEANING | Null Values

Count nulls

Show code

	0
InvoiceNo	0
StockCode	0
Description	1454
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	135080
Country	0

dtype: int64

.describe(): 'Description' overview

Show code

	InvoiceNo	StockCode	Description	Country
count	541909	541909	540455	541909
unique	25900	4070	4223	38
top	573585	85123A	WHITE HANGING HEART T-LIGHT HOLDER	United Kingdom
freq	1114	2313	2369	495478

// Observations

Show code

Observation

where using 'StockCode' as identifier:

- 4070 unique rows on 'StockCode'
- 1454 null values on 'StockCode'

- 'StockCode' = 541,909 total row
- 'Description' = 540,455 total rows
- // most likely, 1,454 'StockCode' rows have no corresponding 'Description', (541,909 - 540,455)

> .

Show code

> [Description] Nulls

Show code

> ~~ Investigate: 'Description' Null Values

Show code

// Objective: generate a dataframe with 3 columns:

1. 'StockCode' = lists out unique rows
2. 'Count' = shows the number of occurrences of each unique 'StockCode'
3. 'Description' = provides the corresponding description for each 'StockCode'

// Method: create specific dataframes then concatenate ON unique 'StockCode'

.

> Create Dataframe: unique 'StockCode' & corresponding counts

Show code

> Create Dataframe: excluding 'Description' nulls on 'raw'

Show code

```
<class 'pandas.core.frame.DataFrame'>
Index: 540455 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
```

```
---  ------       --------------  -----
 0   InvoiceNo    540455 non-null  object
 1   StockCode    540455 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     540455 non-null  int64
 4   InvoiceDate  540455 non-null  datetime64[ns]
 5   UnitPrice    540455 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      540455 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 37.1+ MB
```

// Observations

Show code

Observation

- 540,455 max rows as per excluding NAs (original 541,909 max rows)
- 'StockCode' = 540,455 total rows (previously 541,909)
- 1,454 rows are 'Description' nulls as per calculation and section: counting nulls

Create Dataframe: unique 'StockCode' & corresponding 'Description'

Show code

StockCode	Description
10002	INFLATABLE POLITICAL GLOBE
10080	GROOVY CACTUS INFLATABLE
10120	DOGGY RUBBER
10123C	HEARTS WRAPPING TAPE
10124A	SPOTS ON RED BOOKCOVER TAPE

dtype: object

Concatenate Dataframe: unique StockCode + Counts + 'Description'

Show code

	StockCode	Count	Description
3536	85123A	2313	WHITE HANGING HEART T-LIGHT HOLDER
1348	22423	2203	REGENCY CAKESTAND 3 TIER
3515	85099B	2159	JUMBO BAG RED RETROSPOT
2733	47566	1727	PARTY BUNTING
180	20725	1639	LUNCH BAG RED RETROSPOT
...	...	...	...
885	21854	0	NaN
886	21858	0	NaN
2786	62095B	0	NaN
937	21923	0	NaN
2593	35951	0	NaN

4070 rows × 3 columns

// Observations

Show code

Observation

- 4070 unique 'StockCode' values (consistent with section:.describe(): 'Description' overview)
- highest count at 2,313 = 'StockCode' 85123A, WHITE HANGING HEART T-LIGHT HOLDER

Merge DataFrames: 'unique_stocks' and 'raw'

Show code

```
<class 'pandas.core.frame.DataFrame'>
Index: 541909 entries, 160128 to 40383
Data columns (total 10 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   InvoiceNo      541909 non-null  object
 1   StockCode      541909 non-null  object
 2   Description_x  540455 non-null  object
 3   Quantity       541909 non-null  int64
 4   InvoiceDate    541909 non-null  datetime64[ns]
 5   UnitPrice      541909 non-null  float64
 6   CustomerID     406829 non-null  float64
```

```
 7   Country        541909 non-null  object
 8   Count          541909 non-null  int64
 9   Description_y  541797 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(2), object(5)
memory usage: 45.5+ MB
```

// Observations

Show code

Observation

- Description_x = 540,455 total rows (from 'raw)
- Description_y = 541, 797 total rows (from 'unique_stocks')
- CustomerID = 406, 829 total rows
- CustomerID datatype = float64 (must be converted into 'object')
- 541, 909 max total rows

```
# Refining updated dataframe
'''
> dropping 'Description_x' from 'raw'
> dropping 'Count' from 'unique_stocks'
> renaming 'Description_y
'''
merged_data = merged_data.drop(['Description_x', 'Count'], axis=1).rename(columns={'Descrip
```

```
# Renaming dataframe back as 'raw'
raw = merged_data.copy()
raw.head()
```

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Descrip
0	536365	85123A	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	WI HANG HEAF LI HOL
1	536365	71053	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	WI ME

Count remaining nulls

Show code

	0
InvoiceNo	0
StockCode	0
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	135080
Country	0
Description	112

dtype: int64

// Observations

Show code

Observation

- There are still 112 nulls on 'Description'
- 135,080 nulls on 'CustomerID'

~~Investigate: 'Description' Remaining Null Values

Show code

// Objective: examine nature of nulls on [Description]; specifically, those that could pose as irrelevant rows for the sales transaction analysis

.

// Method: identify nature of 'Description' nulls accounting corresponding values on the following: (1) 'UnitPrice' (2) 'Quantity' (3) 'CustomerID'

```
# Create a dataframe to examine nulls
'''
a separate dataframe to examine nulls without affecting the updated working dataframe 'raw'
'''

zero_unitprice = raw[raw['UnitPrice'] == 0][['UnitPrice','Description', 'Quantity', 'Custom
```

```
null_zero_unitprice = zero_unitprice[zero_unitprice['Description'].isna()].sort_values(by =
null_zero_unitprice.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 112 entries, 1259 to 14
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   index        112 non-null    int64  
 1   UnitPrice    112 non-null    float64
 2   Description  0 non-null      object 
 3   Quantity     112 non-null    int64  
 4   CustomerID   0 non-null      float64
dtypes: float64(2), int64(2), object(1)
memory usage: 5.2+ KB
```

```
null_zero_unitprice.describe()
```

	index	UnitPrice	Quantity	CustomerID
count	112.000000	112.0	112.000000	0.0
mean	129823.839286	0.0	-8.196429	NaN
std	83493.428445	0.0	16.003288	NaN
min	1970.000000	0.0	-102.000000	NaN
25%	75228.500000	0.0	-11.000000	NaN
50%	143303.500000	0.0	-4.000000	NaN
75%	171576.500000	0.0	-1.000000	NaN
max	497301.000000	0.0	57.000000	NaN

```
raw[raw['Description'].isna()].describe()
```

	Quantity	InvoiceDate	UnitPrice	CustomerID
count	112.000000	112	112.0	0.0
mean	-8.196429	2011-03-19 12:59:55.178571520	0.0	NaN
min	-102.000000	2010-12-01 14:32:00	0.0	NaN
25%	-11.000000	2011-01-28 14:48:15	0.0	NaN
50%	-4.000000	2011-04-01 16:40:30	0.0	NaN
75%	-1.000000	2011-04-28 15:06:15	0.0	NaN
max	57.000000	2011-11-24 10:36:00	0.0	NaN
std	16.003288	NaN	0.0	NaN

// Observations

Show code

Observation

- 112 rows are zero in 'UnitPrice' and null in both 'Description' and 'CustomerID'
- Considering these 112 have insufficient information, they will be considered as irrelevent rows hence be removed

Remove null rows

Show code

```
print('Reviewing null count on working dataset')
# Working dataset overview
raw.isna().sum()
```

Reviewing null count on working dataset

	0
InvoiceNo	0
StockCode	0
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	135080
Country	0
Description	112

dtype: int64

```
# Drop rows that have missing values in the 'item_name' column
raw = raw.dropna(subset=['Description'])

# Updated working dataset overview
print('Updated dataset: reviewing null count')
raw.isna().sum()
```

```
Updated dataset: reviewing null count
```

	0
InvoiceNo	0
StockCode	0
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	134968
Country	0
Description	0

dtype: int64

// Observations

Show code

Observations

- 'Description' has now zero nulls
- 'CustomerID' has 134,968 nulls

.

Show code

[CustomerID] Nulls

Show code

// Objective: remaining nulls on CustomerID are consired relevant rows hence be kept

// Method:

1. rename those nulls with 'NA'

2. convert datatype float64 into int64 (to remove decimals), then str 'object'

Replace null values with 'NA'

Show code

	0
InvoiceNo	0
StockCode	0
Quantity	0
InvoiceDate	0
UnitPrice	0
CustomerID	0
Country	0
Description	0

dtype: int64

```
raw.describe(include='object')
```

	InvoiceNo	StockCode	CustomerID	Country	Description
count	541797	541797	541797	541797	541797
unique	25788	3958	4373	38	3823
top	573585	85123A	NA	United Kingdom	WHITE HANGING HEART T-LIGHT HOLDER
freq	1114	2313	134968	495366	2380

NULL-CLEAN Working Dataset

Show code

```
<class 'pandas.core.frame.DataFrame'>
Index: 541797 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541797 non-null  object
 1   StockCode    541797 non-null  object
 2   Quantity     541797 non-null  int64
 3   InvoiceDate  541797 non-null  datetime64[ns]
```

```
 4   UnitPrice    541797 non-null  float64
 5   CustomerID   541797 non-null  object
 6   Country      541797 non-null  object
 7   Description  541797 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 37.2+ MB
```

.

Show code

CLEANING | Duplicate Rows

[ ] ↳ *7 cells hidden*

COLUMNS | Examine Nature

.describe() numberic values

Show code

	Quantity	InvoiceDate	UnitPrice
count	536527.000000	536527	536527.000000
mean	9.623219	2011-07-04 09:28:59.156911360	4.633627
min	-80995.000000	2010-12-01 08:26:00	-11062.060000
25%	1.000000	2011-03-28 11:34:00	1.250000
50%	3.000000	2011-07-19 14:29:00	2.080000
75%	10.000000	2011-10-18 17:05:00	4.130000
max	80995.000000	2011-12-09 12:50:00	38970.000000
std	219.152804	NaN	97.243424

// Observation

Show code

Observation

- 'Quantity' = -80,995.00 extreme min value
- 'UnitPrice' = -11062.06 extreme min value
- 'InvoiceDate' = December 2010 to 2011 transaction range of dataset

> .

Show code

> [UnitPrice] Extreme Values

Show code

> ~~Investigate: 'UnitPrice' extreme values

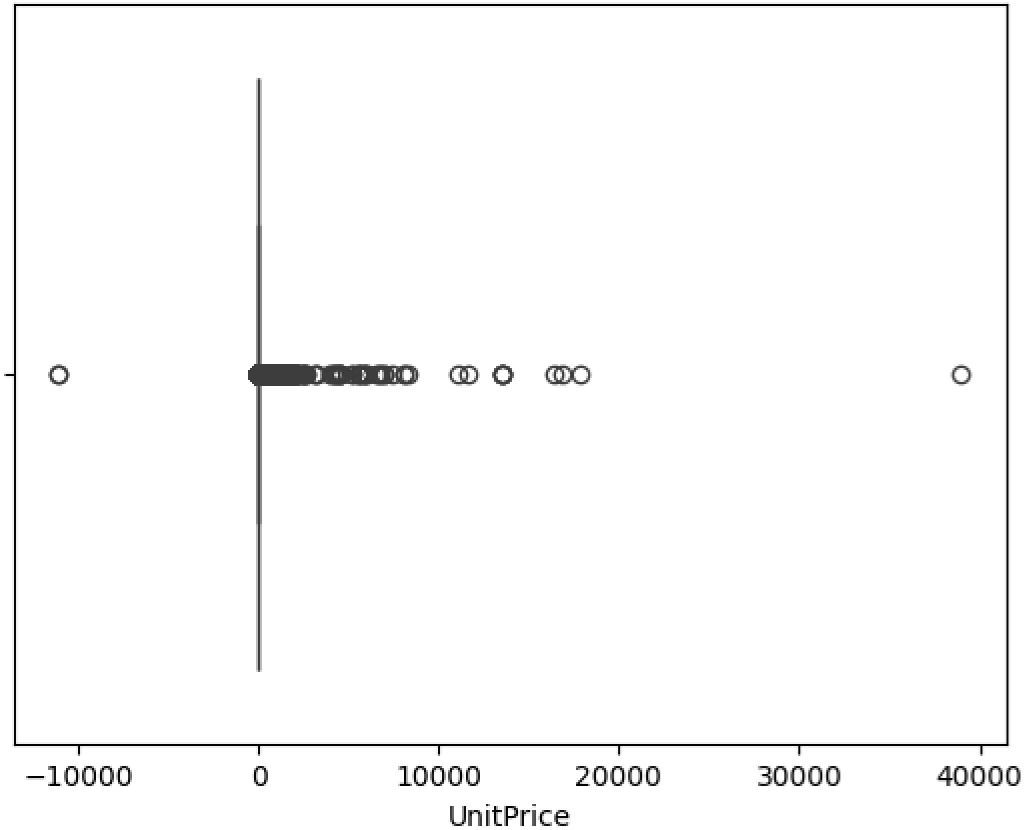
Show code

```
<class 'pandas.core.frame.DataFrame'>
Index: 536527 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    536527 non-null  object
 1   StockCode    536527 non-null  object
 2   Quantity     536527 non-null  int64
 3   InvoiceDate  536527 non-null  datetime64[ns]
 4   UnitPrice    536527 non-null  float64
 5   CustomerID   536527 non-null  object
 6   Country      536527 non-null  object
 7   Description  536527 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 36.8+ MB
```

> Check Outlier: boxplot 'UnitPrice'

Show code

<Axes: xlabel='UnitPrice'>



Check Outlier: isolate values

Show code

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
15016	C537630	AMAZONFEE	-1	2010-12-07 15:04:00	13541.33	NA	United Kingdom
15017	537632	AMAZONFEE	1	2010-12-07 15:08:00	13541.33	NA	United Kingdom
16232	C537644	AMAZONFEE	-1	2010-12-07 15:34:00	13474.79	NA	United Kingdom
16356	C537651	AMAZONFEE	-1	2010-12-07 15:49:00	13541.33	NA	United Kingdom
43702	C540117	AMAZONFEE	-1	2011-01-05 09:55:00	16888.02	NA	United Kingdom
43703	C540118	AMAZONFEE	-1	2011-01-05 09:57:00	16453.71	NA	United Kingdom

// Observation

Show code

Observation

- 'UnitPrice' values > 10000 have alphameric 'StockCode' values {instead of alphanumeric}
- Hence, investigate nature of extreme values accounting columns (1) UnitPrice, (2) StockCode, (3) Quantity, (4) Description

.

Show code

Examine 'StockCode' Alphamerics

Show code

// Objective: find patterns on 'StockCode' related to the extreme values found on 'UnitPrice'

// Method:

1. create a dataframe isolating only alphameric values on 'StockCode'
2. create a dataframe of unique alphameric 'StockCode'& corresponding counts
3. create dataframe printing out the following:

- (1) unique alphameric 'StockCode'
- (2) each corresponding 'Description'
- (3) each corresponding count of 'StockCode' occurences
- (4) each corresponding most reoccurring value on 'Quantity' and 'UnitPrice'

Create Dataframe: .info() alphameric 'StockCode'

Show code

```
<class 'pandas.core.frame.DataFrame'>
Index: 2790 entries, 45 to 541768
Data columns (total 4 columns):
```

```
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   StockCode    2790 non-null   object 
 1   Description  2790 non-null   object 
 2   Quantity     2790 non-null   int64  
 3   UnitPrice    2790 non-null   float64
dtypes: float64(1), int64(1), object(2)
memory usage: 109.0+ KB
```

> Create Dataframe: unique alphameric 'StockCode' & corresponding counts

Show code

> Create Dataframe: unique alphameric 'StockCode' + Description + Count + Quantity + UnitPrice

Show code

```
Total Count Rows containing Alphameric StockCode Values = 2790
```

	StockCode	Description	Count	Max_Quantity	Max_UnitPrice
0	AMAZONFEE	AMAZON FEE	34	-1	13541.330
1	B	Adjust bad debt	3	1	-11062.060
2	BANK CHARGES	Bank Charges	37	-1	15.000
3	CRUK	CRUK Commission	16	-1	1.600
4	D	Discount	77	-1	11.840
5	DCGSSBOY	BOYS PARTY BAG	11	1	3.290
6	DCGSSGIRL	GIRLS PARTY BAG	13	2	3.290
7	DOT	DOTCOM POSTAGE	710	1	3.290
8	M	Manual	566	-1	1.250
9	PADS	PADS TO MATCH ALL CUSHIONS	4	1	0.001
10	POST	POSTAGE	1256	1	18.000
11	S	SAMPLES	62	-1	33.050

> // Observations

Show code

Observation:

- 2,790 alphameric 'StockCode' rows
- 13 unique alphameric 'StockCode' values
- Most identified alphameric 'StockCode' are not relevant to the sales transaction analysis; all shall be removed except:

1. DCGSSBOY
2. DCGSSGIRL

Remove Certain Alphameric Values on 'StockCode'

Show code

Updated Working Dataframe

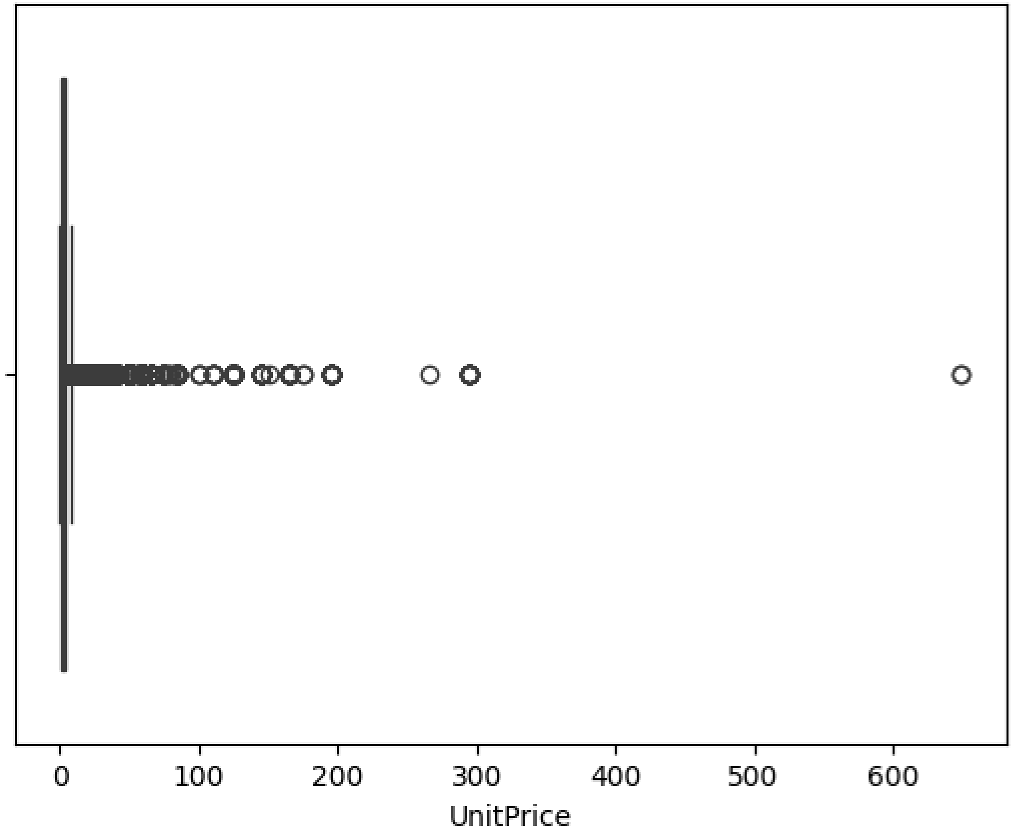
Show code

```
<class 'pandas.core.frame.DataFrame'>
Index: 533761 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    533761 non-null  object
 1   StockCode    533761 non-null  object
 2   Quantity     533761 non-null  int64
 3   InvoiceDate  533761 non-null  datetime64[ns]
 4   UnitPrice    533761 non-null  float64
 5   CustomerID   533761 non-null  object
 6   Country      533761 non-null  object
 7   Description  533761 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 36.7+ MB
```

Check Outlier: boxplot 'Unitprice'

Show code

<Axes: xlabel='UnitPrice'>



// Observations

Show code

Observation

- removal of particular rows improved the distribution on 'UnitPrice'
- 2, 766 rows were removed
- 533, 761 rows on updated working dataset

.

Show code

[Quantity] Extreme Values

Show code

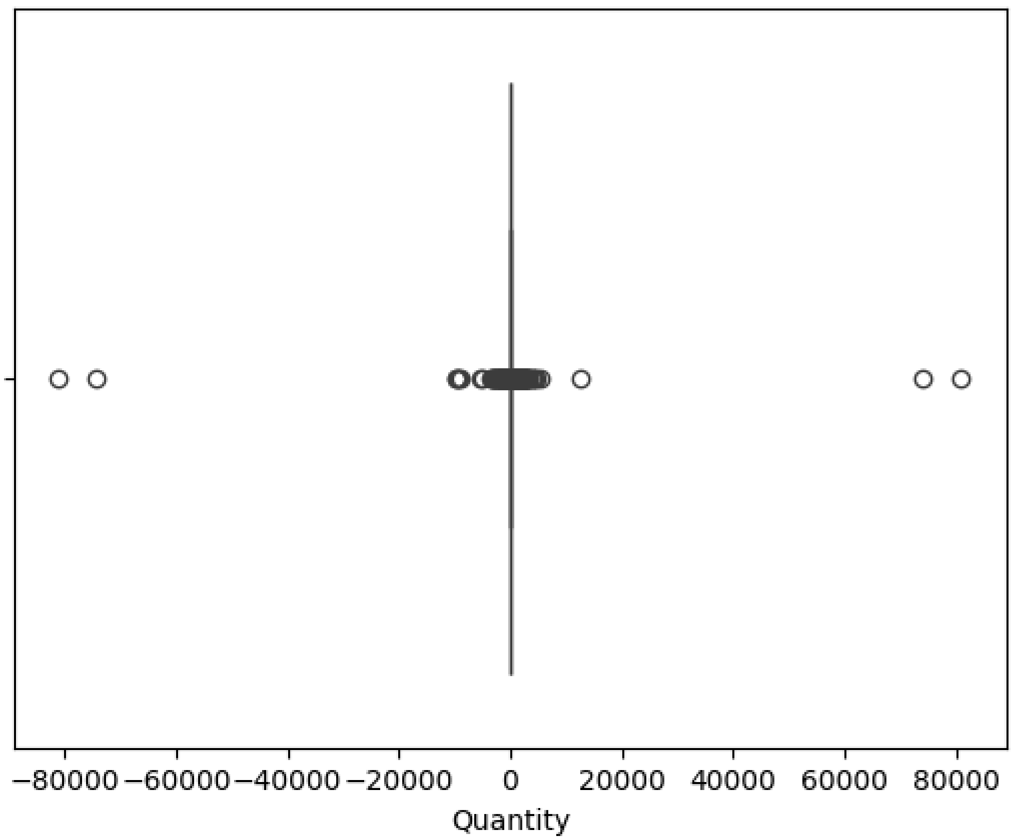
~~Investigate: 'Quantity' extreme values

Show code

Check Outlier: boxplot 'Quantity'

Show code

```
<Axes: xlabel='Quantity'>
```



Check Outlier: isolate values

Show code

```
Rows having extreme values (>= 15000 and <= -15000) on 'Quantity'
```

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	De
61619	541431	23166	74215	2011-01-18 10:01:00	1.04	12346	United Kingdom	

// Observations

Show code

Observations:

- 'Quantity' Negative values could have a corresponding transaction having a (+) value on 'Quantity'

Remove Extreme Values

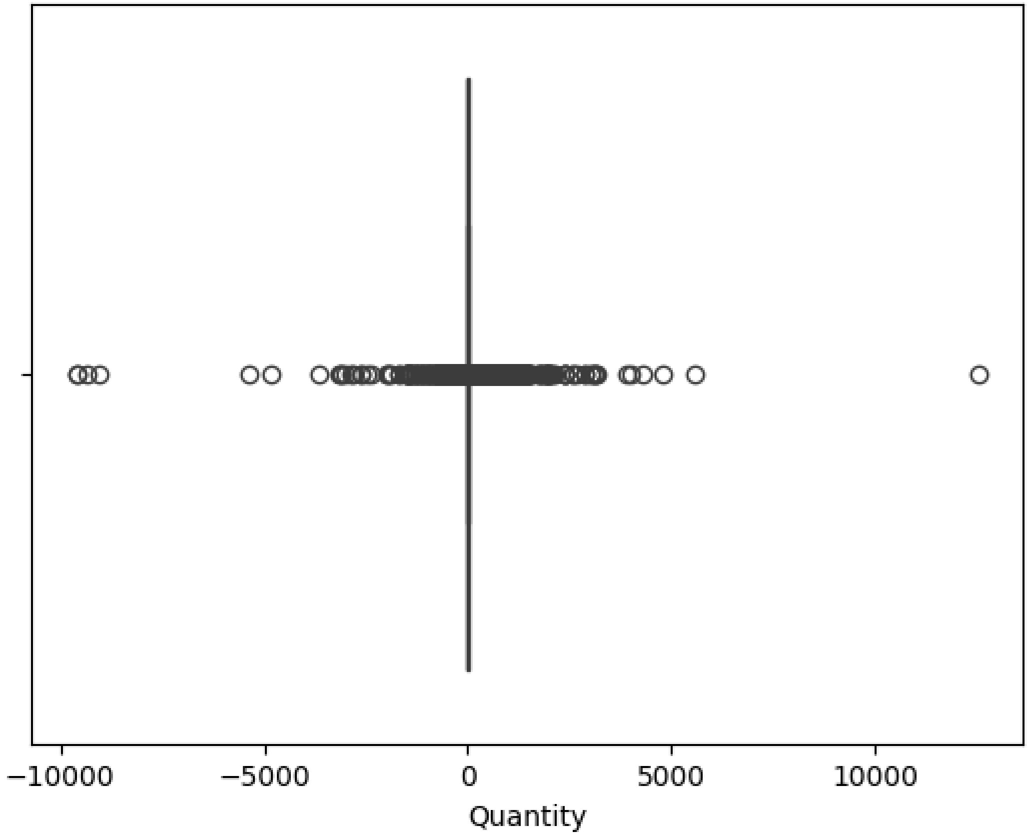
Show code

```
<class 'pandas.core.frame.DataFrame'>
Index: 533757 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    533757 non-null  object
 1   StockCode    533757 non-null  object
 2   Quantity     533757 non-null  int64
 3   InvoiceDate  533757 non-null  datetime64[ns]
 4   UnitPrice    533757 non-null  float64
 5   CustomerID   533757 non-null  object
 6   Country      533757 non-null  object
 7   Description  533757 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 36.7+ MB
```

Check Outlier: boxplot 'Quantity'

Show code

<Axes: xlabel='Quantity'>



// Observations

Show code

Observations:

- 4 rows were considered outliers hence removed
- 533, 757 rows on updated working dataset
- **NOTE:** 'Quantity' Negative values could have a corresponding transaction having a (+) value on 'Quantity'
- Several values on 'Quantity' are negative: **examine**

Check negatives on 'Quantity'

Show code

~~Investigate: 'Quantity' negative values

Show code

// Objective: examine nature of negative valued 'Quatity' rows; specifically:

1. those that could have a corresponding transaction having a (+) value on 'Quantity'
2. those that could pose as irrelevant rows for the sales transaction analysis

// Method:

- identify cancelled transactions that have corresponding (+) values as orders prior cancellation; accounting the following columns:

-> Exact Values on (1) StockCode (2) Quantity (3) CustomerID (4) Description (5) Country (6) InvoiceDate (7) UnitPrice*

-> Slight variation values on (6) InvoiceNo (7)

- remove those identified rows that are considered irrelevant cancelled transactions; to then further identify other factors outside cancelled transactions

- identify the nature accounting corresponding values on the following: (1) 'InvoiceNo' (2) 'Description' (3) 'Quantity', (4) 'UnitPrice'

```
# Columns to check for exact matching
exact_match_columns = ['StockCode', 'Quantity', 'CustomerID', 'Description', 'Country']

# Extract numeric part from InvoiceNo using vectorized operations
raw['NumericInvoiceNo'] = raw['InvoiceNo'].str.extract(r'(\d+)$', expand=False)
raw.sort_values(by='InvoiceNo', ascending=False)
```

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	De
541716	C581569	84978	-1	2011-12-09 11:58:00	1.25	17315	United Kingdom	H
541717	C581569	20979	-5	2011-12-09 11:58:00	1.25	17315	United Kingdom	3(RE
541715	C581568	21258	-5	2011-12-09 11:57:00	10.95	15311	United Kingdom	V B(
540448	C581490	22178	-12	2011-12-09 09:57:00	1.95	14397	United Kingdom	V H/
540449	C581490	23144	-11	2011-12-09 09:57:00	0.83	14397	United Kingdom	
...	...	...	...	...	...	...	...	
5	536365	22752	2	2010-12-01 08:26:00	7.65	17850	United Kingdom	E

```
# Filter rows where each group has more than one row (indicating duplicates)
filtered_df = raw[raw.groupby(exact_match_columns + ['NumericInvoiceNo']).transform('size')

print("Filtered rows from DataFrame X:")
(filtered_df)
```

```
Filtered rows from DataFrame X:
```

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	De
6086	536876	84879	8	2010-12-03 11:36:00	3.19	NA	United Kingdom	A OI
6186	536876	84879	8	2010-12-03 11:36:00	1.69	NA	United Kingdom	A OI
21049	538071	22585	1	2010-12-09 14:09:00	0.00	NA	United Kingdom	F B
21050	538071	22585	1	2010-12-09 14:09:00	2.51	NA	United Kingdom	F B
24401	538349	21524	1	2010-12-10 14:59:00	7.95	NA	United Kingdom	I

```
filtered_df.describe(include='object')
```

	InvoiceNo	StockCode	CustomerID	Country	Description
count	825	825	825	825	825
unique	518	252	159	11	252
top	C545677	22423	17511	United Kingdom	REGENCY CAKESTAND 3 TIER
freq	9	47	48	694	47

+~+`+ I LEFT OFF HERE!!!

View Cancelled Transactions 'InvoiceNo'

Show code

	Quantity	InvoiceDate	UnitPrice
count	8668.000000	8668	8668.000000
mean	-13.245155	2011-06-26 08:55:37.801107712	4.423229
min	-9360.000000	2010-12-01 09:49:00	0.030000
25%	-6.000000	2011-03-21 16:15:00	1.450000
50%	-2.000000	2011-07-08 13:37:00	2.550000
75%	-1.000000	2011-10-06 20:36:00	4.950000
max	-1.000000	2011-12-09 11:58:00	295.000000
std	120.595003	NaN	9.145608

// Observations

Show code

Observation

- 8,668 rows were cancelled transactions; to remove

Remove Cancelled Transactions 'InvoiceNo'

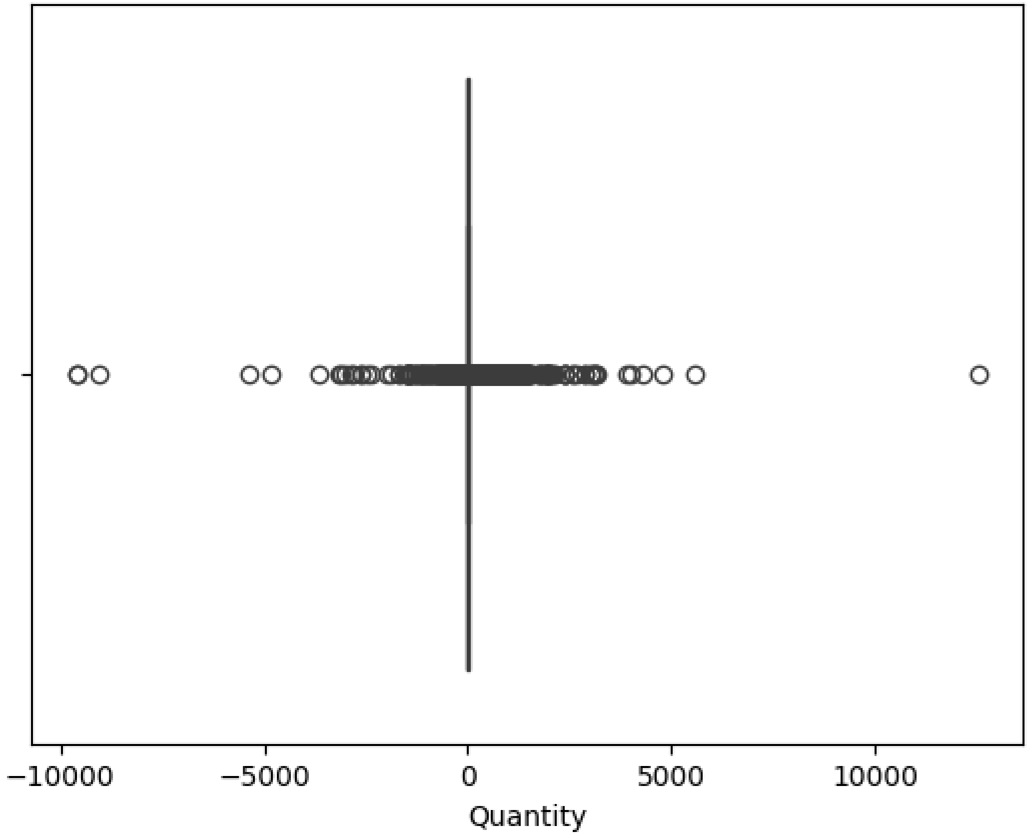
Show code

```
<class 'pandas.core.frame.DataFrame'>
Index: 525089 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    525089 non-null  object
 1   StockCode    525089 non-null  object
 2   Quantity     525089 non-null  int64
 3   InvoiceDate  525089 non-null  datetime64[ns]
 4   UnitPrice    525089 non-null  float64
 5   CustomerID   525089 non-null  object
 6   Country      525089 non-null  object
 7   Description  525089 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 36.1+ MB
```

Check Outlier: boxplot 'Quantity'

Show code

<Axes: xlabel='Quantity'>




```
raw2
```

Check negatives on 'Quantity'

Show code

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	De
115818	546152	72140F	-5368	2011-03-09 17:25:00	0.0	NA	United Kingdom	t
225528	556687	23003	-9058	2011-06-14 10:36:00	0.0	NA	United Kingdom	

```
raw2[raw2['Quantity'] <= -1000].sort_values(by='Quantity', ascending = False).describe()
```

	Quantity	InvoiceDate	UnitPrice
count	47.000000	47	47.0
mean	-2357.446809	2011-07-09 06:43:29.361702400	0.0
min	-9600.000000	2011-01-10 10:36:00	0.0
25%	-2609.000000	2011-04-24 13:36:00	0.0
50%	-1440.000000	2011-06-21 11:34:00	0.0
75%	-1280.500000	2011-10-16 01:39:30	0.0

```
raw2[raw2['Quantity'] <= 0].sort_values(by='Quantity', ascending = False).describe()
```

	Quantity	InvoiceDate	UnitPrice
count	1239.000000	1239	1239.0
mean	-166.213075	2011-06-21 15:29:55.351089664	0.0
min	-9600.000000	2010-12-01 16:50:00	0.0
25%	-93.500000	2011-04-01 13:53:00	0.0
50%	-33.000000	2011-06-10 10:50:00	0.0
75%	-10.000000	2011-10-03 14:28:30	0.0
max	-1.000000	2011-12-08 15:24:00	0.0
std	609.447074	NaN	0.0

Observations

Show code

Double-click (or enter) to edit