

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	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

> 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: investigate the nature of null values on column 'Description' by identifying significant null rows

// Method:

- 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'
- Create specific dataframes then concatenate ON unique 'StockCode'

.

> Create Dataframe: unique 'StockCode' & corresponding counts

[Show code](#)



Description

StockCode	
10002	71
10080	23
10120	30
10123C	3
10123G	0
...	...
gift_0001_20	10
gift_0001_30	7
gift_0001_40	3
gift_0001_50	4
m	1

4070 rows × 1 columns

dtype: int64

> 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



Description	
StockCode	
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

Next
steps:

Generate code
with unique_stocks



View recommended
plots

New interactive
sheet

> // 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

> Refine generated dataframe

[Show code](#)



	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
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 DataFrame: examine nulls

[Show code](#)

```

↗ <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'

Since these 112 rows have insufficient information, they are considered irrelevant for the transaction analysis and should be removed.

> Review null row count on working dataset

[Show code](#)

 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

> Remove: null rows on 'Description'

[Show code](#)

↔ 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](#)

The remaining nulls on CustomerID are consired relevant rows hence be kept.

// Objective: refine column 'CustomerID'

// 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](#)

↔ Updated dataset: reviewing null count

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

dtype: int64

> .describe(): dataset overview 'object'

[Show code](#)

	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

> Count Duplicate Rows

Show code

 5270

> Remove: duplicate rows

Show code

> .shape: updated dataset

Show code

 (536527, 8)

> // Observations

Show code

Observation

- Updated dataset = 536,527 max total rows (previously 541,797)
- Removed 5,270 duplicate rows

> .




Show code

✓ INVESTIGATE COLUMNS OF DATASET

✓ COLUMNS | Examine Nature of numeric values

> .describe() numeric 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](#)

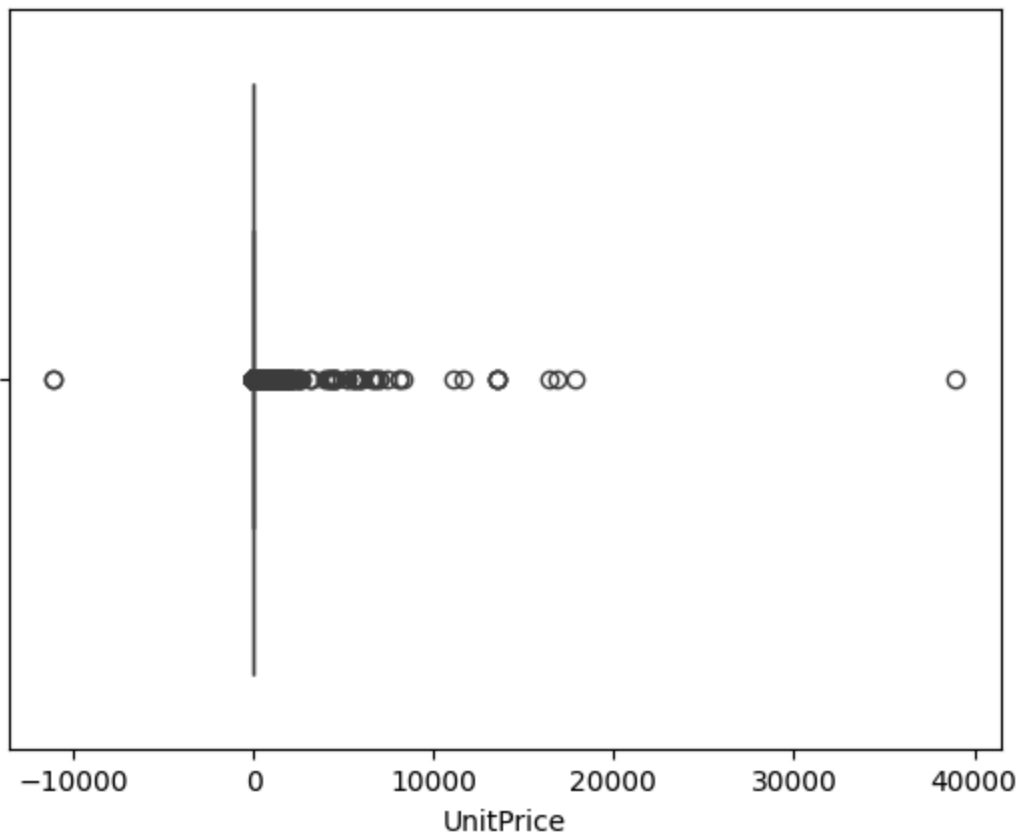
⇌ Overview of Current Working Dataset

```
<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 >10,000 'UnitPrice'

[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

> 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 alphanumeric 'StockCode'
- (2) each corresponding 'Description'
- (3) each corresponding count of 'StockCode' occurrences
- (4) each corresponding most reoccurring value on 'Quantity' and 'UnitPrice'

> Create Dataframe: .info() alphanumeric '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 alphanumeric 'StockCode' & corresponding counts

Show code

> Create Dataframe: unique alphanumeric '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	

Next
steps:

Generate code
with combined_df



View recommended
plots

New interactive
sheet

> // 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 = BOYS PARTY BAG
 2. DCGSSGIRL = GIRLS PARTY BAG
 3. D = Discount

> List: alphameric StockCode values to exclude

Show code





```
[ 'POST',  
  'DOT',  
  'M',  
  'BANK CHARGES',  
  'S',  
  'AMAZONFEE',  
  'm',  
  'PADS',  
  'B',  
  'CRUK' ]
```

> Remove: alphameric StockCode values on Working Dataframe

Show code



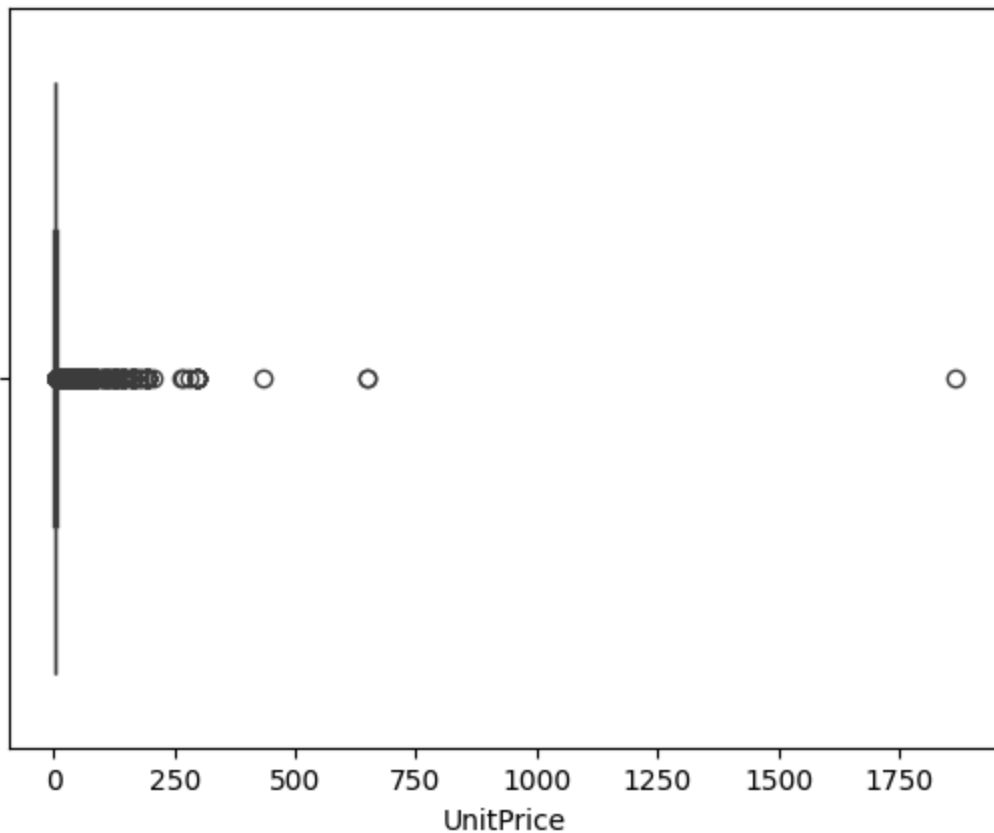
Previewing details of the updated Working DataFrame

	Quantity	InvoiceDate	UnitPrice	
count	533838.000000	533838	533838.000000	
mean	9.651321	2011-07-04 10:50:14.597461760	3.301370	
min	-80995.000000	2010-12-01 08:26:00	0.000000	
25%	1.000000	2011-03-28 11:34:00	1.250000	
50%	3.000000	2011-07-19 15:23:00	2.080000	
75%	10.000000	2011-10-18 17:10:00	4.130000	
max	80995.000000	2011-12-09 12:50:00	1867.860000	
std	219.652203	NaN	5.336549	

> Updated: boxplot 'UnitPrice'

Show code

↔ <Axes: xlabel='UnitPrice'>



> // Observations

[Show code](#)

Observations

- 533,838 total rows of the updated working dataframe
- 2,689 rows with alphanumeric StockCodes were removed after being identified as postage and bad debt records. While some of these rows were also recognized as outliers, all were deemed not relevant to the transaction analysis of retail products.
- Their removal improved the distribution of 'UnitPrice' by reducing extreme values.

Updated 'UnitPrice' status:

1. minimum value = 0 (previously -11,062.06)
2. maximum value = 1867.86 (previously 38,970)

> Examine 'StockCode' AlphaNumerics

[Show code](#)

> Count: alphanumerics on StockCode

[Show code](#)

 Listing Alphanumerics on StockCode at index 0

	count
StockCode	
2	433968
8	62423
4	11368
1	7574
7	7142
3	5691
9	4633
5	633
C	144
D	116
6	112
g	34

dtype: int64

> Check: details of 'C' StockCode

[Show code](#)



	InvoiceNo	StockCode	CustomerID	Country	Description
count	144	144	144	144	144
unique	144	1	30	4	1
top	536540	C2	14911	EIRE	CARRIAGE
freq	1	144	85	108	144

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
1423	536540	C2	1	2010-12-01 14:05:00	50.0	14911	EIRE	CARRIAGE
12119	537368	C2	1	2010-12-06 12:40:00	50.0	14911	EIRE	CARRIAGE
12452	537378	C2	1	2010-12-06 13:06:00	50.0	14911	EIRE	CARRIAGE
19975	537963	C2	1	2010-12-09 11:30:00	50.0	13369	United Kingdom	CARRIAGE
20016	538002	C2	1	2010-12-09 11:48:00	50.0	14932	Channel Islands	CARRIAGE
...
515000	579768	C2	1	2011-11-30 15:08:00	50.0	14911	EIRE	CARRIAGE
516484	579910	C2	1	2011-12-01 08:52:00	50.0	14911	EIRE	CARRIAGE

> Check: details of 'D' StockCode

Show code



	count
Description	
Discount	77
GIRLS PARTY BAG	13
BOYS PARTY BAG	11
BOXED GLASS ASHTRAY	5
ebay	3
SUNJAR LED NIGHT NIGHT LIGHT	2
CAMOUFLAGE DOG COLLAR	2
OOH LA LA DOGS COLLAR	2
HAYNES CAMPER SHOULDER BAG	1

dtype: int64

> Check: details of 'g' StockCode

[Show code](#)



InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Descr
539492	gift_0001_40	1	2010-12-20 10:14:00	34.04	NA	United Kingdom	Dotcom Gift Vo £40.
562933	gift_0001_30	1	2011-08-10 16:51:00	25.00	NA	United Kingdom	Dotcom Gift Vo £30.
558066	gift_0001_50	1	2011-06-24 15:45:00	41.67	NA	United Kingdom	Dotcom Gift Vo £50.
558068	gift_0001_20	1	2011-06-24 15:51:00	16.67	NA	United Kingdom	Dotcom Gift Vo £20.
558614	gift_0001_10	1	2011-06-30 15:56:00	8.33	NA	United Kingdom	Dotcom Gift Vo £10.
	gift_0001_50	1	2011-06-30 15:56:00	41.67	NA	United Kingdom	Dotcom Gift Vo £50.
561513	gift_0001_40	1	2011-07-27 15:12:00	33.33	NA	United Kingdom	Dotcom Gift Vo £40.
562420	gift_0001_20	1	2011-08-04 16:38:00	16.67	NA	United Kingdom	Dotcom Gift Vo £20.
564760	gift_0001_10	1	2011-08-30 10:47:00	8.33	NA	United Kingdom	Dotcom Gift Vo £10.
539958	gift_0001_50	1	2010-12-23 13:26:00	42.55	NA	United Kingdom	Dotcom Gift Vo £50.
564760	gift_0001_30	1	2011-08-30 10:47:00	25.00	NA	United Kingdom	Dotcom Gift Vo £30.
564761	gift_0001_30	30	2011-08-30 10:48:00	0.00	NA	United Kingdom	Dotcom Gift Vo £30.
564762	gift_0001_10	30	2011-08-30 10:48:00	0.00	NA	United Kingdom	Dotcom Gift Vo £10.
564974	gift_0001_10	2	2011-08-31 15:32:00	8.33	NA	United Kingdom	Dotcom Gift Vo £10.

565231	gift_0001_30	1	2011-09-02 09:26:00	25.00	NA	United Kingdom	Dotcom Gift Vo £30.
573585	gift_0001_20	1	2011-10-31 14:41:00	16.67	NA	United Kingdom	Dotcom Gift Vo £20.
557500	gift_0001_20	1	2011-06-20 17:07:00	16.67	NA	United Kingdom	Dotcom Gift Vo £20.

> // Observations

Show code

Observations

- 144 rows starting with 'C' = CARRIAGE; remove since these are not sales transactions
- 39 rows starting with 'D' = has several descriptions; but remove 'ebay' records since these are not sales transactions
- 34 rows starting with 'g' = gift vouchers; since no further details were found, these will be assumed as purchased vouchers since the values on 'Quantity' are non-negatives

> Remove: alphanumeric 'StockCode' on Working DataFrame

Show code

```

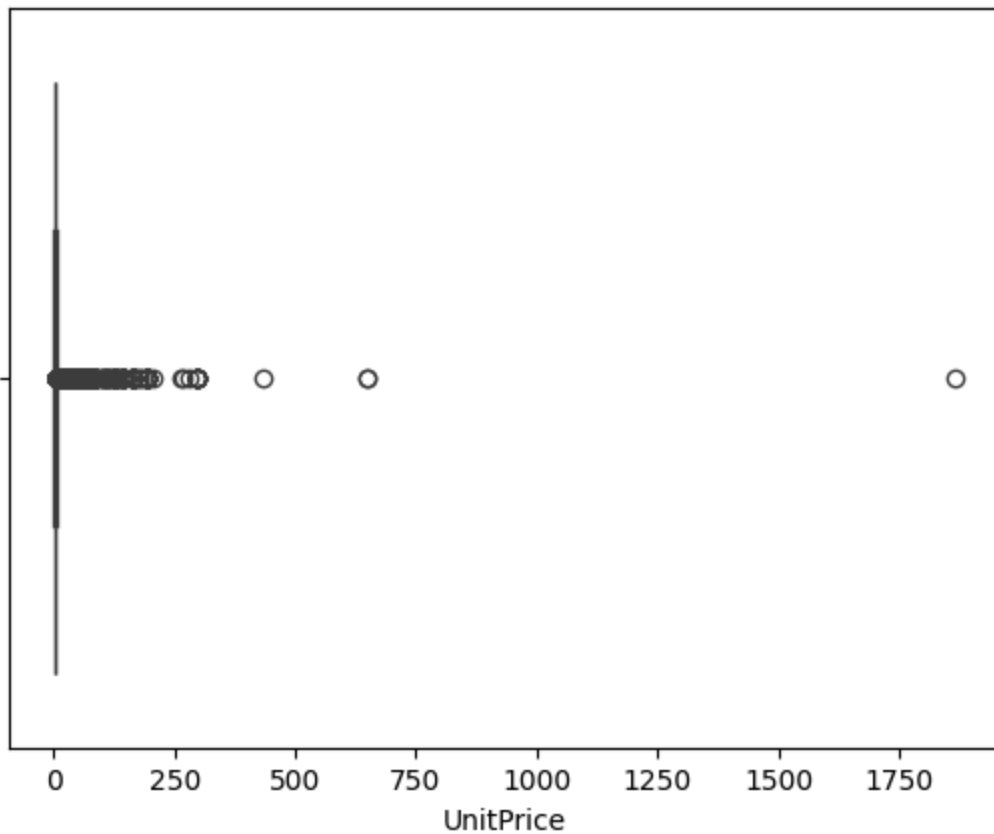
<class 'pandas.core.frame.DataFrame'>
Index: 533691 entries, 0 to 541908
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   InvoiceNo        533691 non-null object
1   StockCode       533691 non-null object
2   Quantity        533691 non-null int64
3   InvoiceDate      533691 non-null datetime64[ns]
4   UnitPrice       533691 non-null float64
5   CustomerID      533691 non-null object
6   Country         533691 non-null object
7   Description      533691 non-null object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 36.6+ MB

```

> Check Outlier: boxplot 'Unitprice'

Show code

 <Axes: xlabel='UnitPrice'>



> // Observations



Show code

Observations

- 533,691 total rows of the updated working dataframe
- 147 rows with alphaNumeric StockCodes were removed after being identified as postages (CARRIAGE and ebay), as deemed not relevant to the transaction analysis of retail products.

> Updated Working Dataframe

Show code

	Quantity	InvoiceDate	UnitPrice	
count	533614.000000	533614	533614.000000	
mean	9.657114	2011-07-04 10:48:47.551675648	3.278995	
min	-80995.000000	2010-12-01 08:26:00	0.000000	
25%	1.000000	2011-03-28 11:34:00	1.250000	
50%	3.000000	2011-07-19 15:23:00	2.080000	
75%	10.000000	2011-10-18 17:10:00	4.130000	
max	80995.000000	2011-12-09 12:50:00	649.500000	
std	219.695530	NaN	4.510962	

> // Observations

[Show code](#)

Observation

- 'UnitPrice' = 0 (minimum value)

| Examine

> [UnitPrice] Zero Values

[Show code](#)

➡ Previewing rows with zero values on 'UnitPrice'

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description
622	536414	22139	56	2010-12-01 11:52:00	0.0	NA	United Kingdom	RETURN CLOTHES
1971	536546	22145	1	2010-12-01 14:33:00	0.0	NA	United Kingdom	CORPSE
1972	536547	37509	1	2010-12-01 14:33:00	0.0	NA	United Kingdom	MILK
2025	536553	37461	3	2010-12-01 14:35:00	0.0	NA	United Kingdom	
2406	536589	21777	-10	2010-12-01 16:50:00	0.0	NA	United Kingdom	RETURN WASH

> // Observations

[Show code](#)

Observation

- 2,380 rows have zero values on 'UnitPrice'
- There are positive (+) and negative (-) values on 'Quantity'

█ Examine those that have negative values

> Create DataFrame: zero unitprice & negative quantity

[Show code](#)

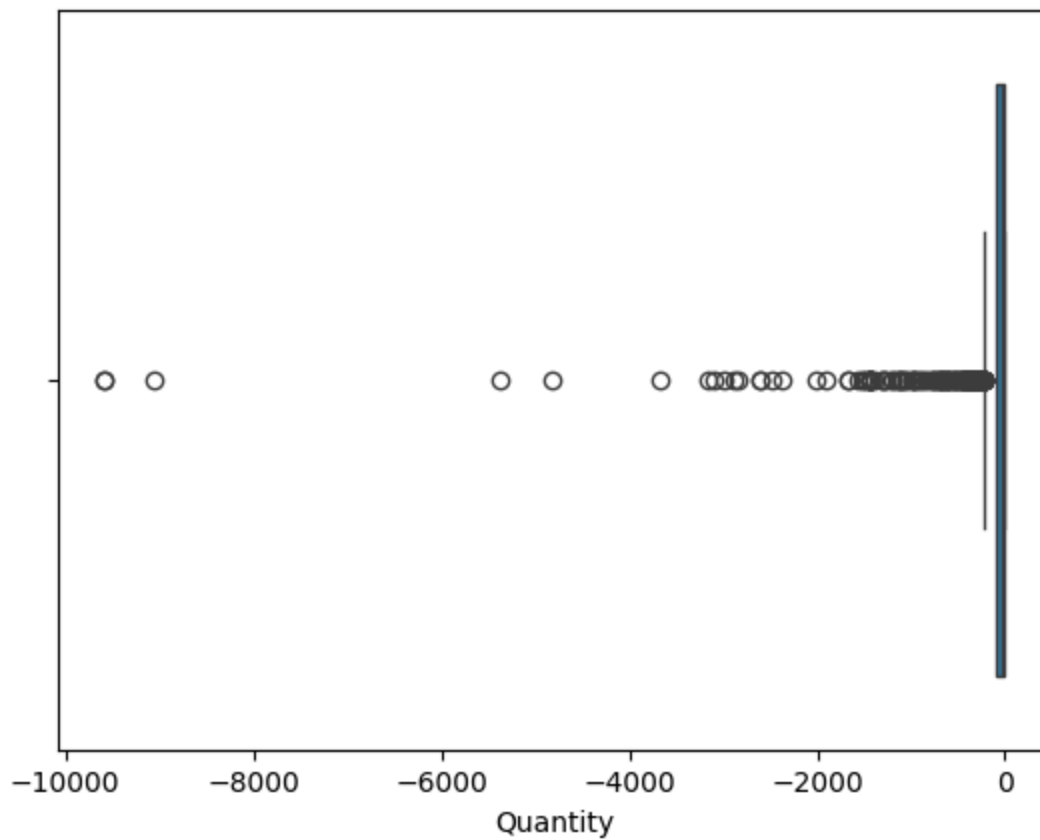


	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	De
	225529	556690	23005	-9600	2011-06-14 10:37:00	0.0	NA	United Kingdom
	225530	556691	23005	-9600	2011-06-14 10:37:00	0.0	NA	United Kingdom
	225528	556687	23003	-9058	2011-06-14 10:36:00	0.0	NA	United Kingdom
	115818	546152	72140F	-5368	2011-03-09 17:25:00	0.0	NA	United Kingdom

> Create boxplot: zero unitprice & negative quantity

Show code

 <Axes: xlabel='Quantity'>



> Check Outlier: isolate values

[Show code](#)

 Rows having extreme values (≤ -4000) on 'Quantity'

InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	De:
225529	556690	23005	2011-06-14 10:37:00	0.0	NA	United Kingdom	

2011-06-14	United Kingdom
------------	----------------

Next steps:

[code](#) [excluded_zeros_n_negatives](#)

☒ [recommended](#)

[New interactive sheet](#)

> // Observations

[Show code](#)

Observations

- 5 rows are extreme values that is zero on 'UnitPrice' and negative in value on 'Quantity'; considered as outliers, these will be removed.
- A "throw away" product description has been found

Examine more of those "throw away" rows on the main dataframe; all of these will be removed as they are not relevant to the transaction analysis of products being sold by the retailer

```
raw.info()
```

```
↔ <class 'pandas.core.frame.DataFrame'>
Index: 533614 entries, 0 to 541908
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   InvoiceNo        533614 non-null object
1   StockCode       533614 non-null object
2   Quantity        533614 non-null int64
3   InvoiceDate      533614 non-null datetime64[ns]
4   UnitPrice       533614 non-null float64
5   CustomerID      533614 non-null object
6   Country         533614 non-null object
7   Description      533614 non-null object
dtypes: datetime64[ns](1), float64(1), int64(1), object(5)
memory usage: 36.6+ MB
```

> Remove DataFrame outliers: with zero unitprice & negative quantity

[Show code](#)



	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
532142	C581484	23843	-80995	2011-12-09 09:27:00	2.08	16446	United Kingdom	I
60711	C541433	23166	-74215	2011-01-18 10:17:00	1.04	12346	United Kingdom	
4212	C536757	84347	-9360	2010-12-02 14:23:00	0.03	15838	United Kingdom	S
336814	566768	16045	-3667	2011-09-14 17:53:00	0.00	NA	United Kingdom	
318919	565304	16259	-3167	2011-09-02 12:18:00	0.00	NA	United Kingdom	P ST
...	
415575	572008	84077	4800	2011-10-27	0.24	12004	United	C

› Identify Rows: remaining 'throw away' product description



Show code



InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Description

› Updated Working Dataframe on 'UnitPrice'

Show code

	Quantity	InvoiceDate	UnitPrice	
count	533609.000000	533609	533609.000000	
mean	9.729272	2011-07-04 10:48:56.872204288	3.279025	
min	-80995.000000	2010-12-01 08:26:00	0.000000	
25%	1.000000	2011-03-28 11:34:00	1.250000	
50%	3.000000	2011-07-19 15:23:00	2.080000	
75%	10.000000	2011-10-18 17:10:00	4.130000	
max	80995.000000	2011-12-09 12:50:00	649.500000	
std	218.330663	NaN	4.510972	

➤ Create DataFrame: zero unitprice & positive quantity

[Show code](#)

↔ Total rows with zero unitprice & positive quantity = 1144

Previewing the first 20 rows, sorted by 'Quantity' in descending order

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
494507	578841	84826	12540	2011-11-25 15:57:00	0.0	13256	United Kingdom	A
73512	542504	37413	5568	2011-01-28 12:03:00	0.0	NA	United Kingdom	
217593	556231	85123A	4000	2011-06-09 15:04:00	0.0	NA	United Kingdom	W
260071	560040	23343	3100	2011-07-14 14:28:00	0.0	NA	United Kingdom	
114091	546139	84988	3000	2011-03-09 16:35:00	0.0	NA	United Kingdom	
73513	542505	79063D	2560	2011-01-28 12:04:00	0.0	NA	United Kingdom	
200750	554550	47566B	1300	2011-05-25 09:57:00	0.0	NA	United Kingdom	T
158100	550460	47556B	1300	2011-04-18 13:18:00	0.0	NA	United Kingdom	
81557	543258	84611B	1287	2011-02-04 16:06:00	0.0	NA	United Kingdom	
416674	573114	20713	1000	2011-10-27 15:36:00	0.0	NA	United Kingdom	
79455	543051	79062D	960	2011-02-03 10:15:00	0.0	NA	United Kingdom	AS
375377	569830	23343	800	2011-10-06 12:38:00	0.0	NA	United Kingdom	
37596	539494	21479	752	2010-12-20	0.0	NA	United Kingdom	V

Next
steps:

[code](#) [excluded_zeros_n_positives](#)

☒ recommended

[New interactive sheet](#)

> // Observations

[Show code](#)

- A "thrown away" product description has been found

All of these will be removed from the main dataset as they are not relevant to the transaction analysis of products being sold by the retailer

> Remove Rows: 'thrown away' product description

Show code



	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	
532142	C581484	23843	-80995	2011-12-09 09:27:00	2.08	16446	United Kingdom	I
60711	C541433	23166	-74215	2011-01-18 10:17:00	1.04	12346	United Kingdom	
4212	C536757	84347	-9360	2010-12-02 14:23:00	0.03	15838	United Kingdom	S
336814	566768	16045	-3667	2011-09-14 17:53:00	0.00	NA	United Kingdom	
318919	565304	16259	-3167	2011-09-02 12:18:00	0.00	NA	United Kingdom	P ST
...	
415575	572008	84077	4800	2011-10-27	0.24	12004	United Kingdom	C

> Count Rows: remaining rows

Show code



```
Total Rows of Updated Working DataFrame = 533605
Remaining rows with zero unit price = 2371
Remaining rows with zero unitprice & negative quantity = 1228
Remaining rows with zero unitprice & positive quantity = 1143
```

> // Observations

Show code

Observations

- 4 rows were considered outliers hence removed; these have zero values on 'UnitPrice' and extreme negative values on 'Quantity'
 - The identified "throw away" product description consists of only one record, which was classified as an outlier and has been removed No additional records were found after eliminating the outlier rows
 - 4 rows with the 'thrown away' product description have been removed
 - 533,605 total rows on updated working dataframe
 - 2,371 rows remaining with zero values in 'UnitPrice'. Although these rows have zero values, they will not be removed, as many represent valid transactions, including placement orders and canceled transactions
-

> [Quantity] Extreme Values


[Show code](#)

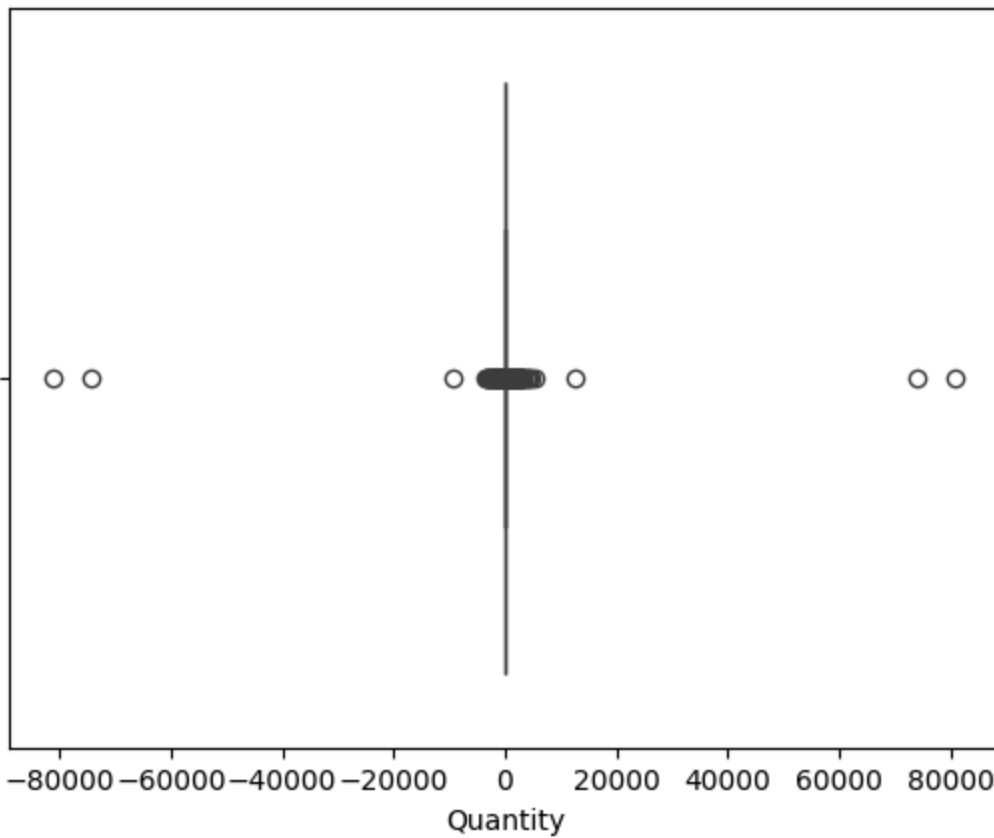
> ~~Investigate: 'Quantity' extreme values

[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:
60706	541431	23166	74215	2011-01-18 10:01:00	1.04	12346	United Kingdom



Next
steps:

[Generate
code with](#) `exclude_quantity`



[View recommended
plots](#)

[New interactive
sheet](#)

> // Observations

[Show code](#)