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

```
# @title Dataset Overview
# Importing csv dataset file
raw = pd.read_csv('/Online Retail_python_csv.csv')
raw.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
			KNITTED UNION FLAG HOT WATER		12/1/2010			United

Dataset Summary Overview

```
# @title Dataset Summary Overview
raw.info()
```

```
<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  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

OBSERVATIONS

- Dataset has 8 columns

- Max Row numbers: 541909
- Column Observations:
    1. Description = lesser rows, at 540455
    2. CustomerID = lesser rows, at 406829
- Hence, existence of nulls

Null Values: counting and replacing

```
#@title Null Values: counting and replacing

# Counting null values
print("Counting Null Values")
nan_count = raw.isna().sum()
print(nan_count)
print()

# Description: replacing null values with 'NA'
print("[Description]: replacing null values with 'NA'")
raw['Description'] = raw['Description'].fillna("NA")
print(raw['Description'].head())
print()

# CustomerID: replacing null values with 'NA'
print("[CustomerID]: replacing null values with 'NA'")
raw['CustomerID'] = raw['CustomerID'].fillna("NA").astype('str') #converting datatype into str
print(raw['CustomerID'].head())
print()

# Updated Dataset Overview
print("UPDATED DATASET OVERVIEW")
print(raw.info())
```

```
Counting Null Values
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64

[Description]: replacing null values with 'NA'
0     WHITE HANGING HEART T-LIGHT HOLDER
1                    WHITE METAL LANTERN
2         CREAM CUPID HEARTS COAT HANGER
3    KNITTED UNION FLAG HOT WATER BOTTLE
4         RED WOOLLY HOTTIE WHITE HEART.
Name: Description, dtype: object

[CustomerID]: replacing null values with 'NA'
0    17850.0
1    17850.0
2    17850.0
3    17850.0
4    17850.0
Name: CustomerID, dtype: object

UPDATED DATASET OVERVIEW
<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  541909 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   541909 non-null  object
 7   Country      541909 non-null  object
dtypes: float64(1), int64(1), object(6)
memory usage: 33.1+ MB
None
```

Notes: [Description] and [CustomerID] have null values: 1454, 135080 respectively

Overview: Datset Columns with 'object' type

Show code

	InvoiceNo	StockCode	Description	InvoiceDate	CustomerID	Country
count	541909	541909	541909	541909	541909	541909
unique	25900	4070	4224	23260	4373	38
top	573585	85123A	WHITE HANGING HEART T-LIGHT HOLDER	10/31/2011 14:41	NA	United Kingdom
freq	1114	2313	2369	1114	135080	495478

Overview: Quant Columns of Dataset

Show code

	Quantity	UnitPrice
count	541909.000000	541909.000000
mean	9.552250	4.611114
std	218.081158	96.759853
min	-80995.000000	-11062.060000
25%	1.000000	1.250000
50%	3.000000	2.080000
75%	10.000000	4.130000
max	80995.000000	38970.000000

Notes:

1. [Unit Price] Why a negative (-) , at -11062.06 as min value?
2. [Quantity] Why a negative (-) , at -80995.00 as min value?
3. [InvoiceNo] unique values at 25900 ~ invoice numbers are duplicated; there are transactions having the same invoice numbers
4. [StockCode] unique values at 4070,
5. [Description] unique values at 4224 -> indicates that there are stockcodes with varying derscriptions
6. [CustomerID] unique values at 4373 ~ distinct customer identity; having total rows of 541,909, there are repeat customer including the 'null' 135,080 customers

~nature investigation of [StockCode]

Show code

Checking Duplicate Rows across the working table

```
#@title Checking Duplicate Rows across the working table
print(f"Duplicated Rows Count = {raw.duplicated().sum()}")
print()

# Displaying Duplicate Table
print('TABLE: List of Duplicated Rows and Their Frequency')
raw_duplicates = raw[raw.duplicated(keep=False)] # working DataFrame

# Count frequency of each duplicated rows
freq_duplicates = raw_duplicates.groupby(list(raw.columns)).size().reset_index()
freq_duplicates.columns = list(raw.columns) + ['Frequency']
distinct_duplicates = freq_duplicates.sort_values(by=list(raw.columns))
print(distinct_duplicates)

print(f"Total Frequency of All Duplicate Rows = {distinct_duplicates['Frequency'].sum()}")
```

```
Duplicated Rows Count = 5268

TABLE: List of Duplicated Rows and Their Frequency
     InvoiceNo StockCode                          Description  Quantity  \
0       536409     21866           UNION JACK FLAG LUGGAGE TAG         1
1       536409     22111          SCOTTIE DOG HOT WATER BOTTLE         1
2       536409     22866         HAND WARMER SCOTTY DOG DESIGN         1
3       536409     22900       SET 2 TEA TOWELS I LOVE LONDON          1
4       536412     21448             12 DAISY PEGS IN WOOD BOX         1
...        ...       ...                                   ...       ...
4874   C572226     85066           CREAM SWEETHEART MINI CHEST        -1
4875   C574095     22326  ROUND SNACK BOXES SET OF4 WOODLAND         -1
4876   C574510     22360      GLASS JAR ENGLISH CONFECTIONERY        -1
4877   C575940     23309  SET OF 60 I LOVE LONDON CAKE CASES        -24
4878   C580764     22667                RECIPE BOX RETROSPOT        -12

           InvoiceDate  UnitPrice CustomerID         Country  Frequency
0      12/1/2010 11:45       1.25    17908.0  United Kingdom          2
1      12/1/2010 11:45       4.95    17908.0  United Kingdom          2
2      12/1/2010 11:45       2.10    17908.0  United Kingdom          2
3      12/1/2010 11:45       2.95    17908.0  United Kingdom          2
4      12/1/2010 11:49       1.65    17920.0  United Kingdom          2
...                ...        ...        ...             ...        ...
4874  10/21/2011 13:58      12.75    15321.0  United Kingdom          2
4875    11/3/2011 9:54       2.95    12674.0          France          2
4876   11/4/2011 13:25       2.95    15110.0  United Kingdom          2
4877  11/13/2011 11:38       0.55    17838.0  United Kingdom          2
4878   12/6/2011 10:38       2.95    14562.0  United Kingdom          2

[4879 rows x 9 columns]
Total Frequency of All Duplicate Rows = 10147
```

OBSERVATIONS:

1. 10147 rows are duplicates of distinct rows
2. 4879 distinct rows are duplicated nth times
3. 5268 rows are exact duplicates of the 4879 distinct rows; **HENCE**, be removed.

~ where 5268 = 10147 - 4879

SUMMARY

- There are 8 Columns: InvoiceNo, StockCode, Description, Quantity, InvoiveDate, UnitPrice, CustomerID, Country
- Raw Max Rows = 541, 909
- Raw: Nulls on 'Description' = 1454
- Raw: Nulls on 'CustomerID' = 135, 080
- [Description] and [CustomerID] Nulls were replaced "NA"

- [Unit Price] negative (-) , at -11062.06 as min value
- [Quantity] negative (-) , at -80995.00 as min value
- [InvoiceNo] unique values at 25900 ~ invoice numbers are duplicated; there are transactions having the same invoice numbers
- [StockCode] unique values at 4070,
- [Description] unique values at 4224 -> indicates that there are stockcodes with varying derscriptions
- [CustomerID] unique values at 4373 ~ distinct customer identity; having total rows of 541,909, there are repeat customer including the 'null' 135,080 customers
- 10147 rows are duplicates of distinct rows
- 4879 distinct rows are duplicated nth times
- 5268 rows are exact duplicates of the 4879 distinct rows; HENCE, be removed. ~ where 5268 = 10147 - 4879

Removing Duplicate Rows

Show code

```
<class 'pandas.core.frame.DataFrame'>
Index: 536641 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    536641 non-null  object
 1   StockCode    536641 non-null  object
 2   Description  536641 non-null  object
 3   Quantity     536641 non-null  int64
 4   InvoiceDate  536641 non-null  object
 5   UnitPrice    536641 non-null  float64
 6   CustomerID   536641 non-null  object
 7   Country      536641 non-null  object
dtypes: float64(1), int64(1), object(6)
memory usage: 36.8+ MB
```

UPDATED Dataset Overview: Total Rows and Columns

Show code

```
(536641, 8)
```

UPDATED Dataset Overview: Columns with 'object'

```
#@title UPDATED Dataset Overview: Columns with 'object'
raw.describe(include='object')
```

	InvoiceNo	StockCode	Description	InvoiceDate	CustomerID	Country
count	536641	536641	536641	536641	536641	536641
unique	25900	4070	4224	23260	4373	38
top	573585	85123A	WHITE HANGING HEART T-LIGHT HOLDER	10/31/2011 14:41	NA	United Kingdom
freq	1114	2301	2357	1114	135037	490300

UPDATED Dataset Overview: Columns with values

```
#@title UPDATED Dataset Overview: Columns with values
raw.describe()
```

	Quantity	UnitPrice
count	536641.000000	536641.000000
mean	9.620029	4.632656
std	219.130156	97.233118
min	-80995.000000	-11062.060000
25%	1.000000	1.250000
50%	3.000000	2.080000
75%	10.000000	4.130000
max	80995.000000	38970.000000

// COLUMN [InvoiceNo]

Column [InvoiceNo]: Examining Column

```
#@title Column [InvoiceNo]: Examining Column
print(raw['InvoiceNo'].str[0].value_counts())
print()
```

```
InvoiceNo
5    527387
C      9251
A         3
Name: count, dtype: int64
```

[InvoiceNo] A's: checking rows starting with A's

```
#@title [InvoiceNo] A's: checking rows starting with A's
raw[raw['InvoiceNo'].str[0] == 'A']
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
299982	A563185	B	Adjust bad debt	1	8/12/2011 14:50	11062.06	NA	United Kingdom
299983	A563186	B	Adjust bad debt	1	8/12/2011 14:51	-11062.06	NA	United Kingdom
299984	A563187	B	Adjust bad debt	1	8/12/2011 14:52	-11062.06	NA	United Kingdom

OBSERVATION: These 3 transactions having InvoiceNos starting with A are bad debts, therefore irrelevant for the sales analysis

[InvoiceNo] A's: removing rows begining with A's

```
#@title [InvoiceNo] A's: removing rows begining with A's
raw = raw[~(raw['InvoiceNo'].str[0] == 'A')]
raw
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	12/9/2011 12:50	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	12/9/2011 12:50	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	12/9/2011 12:50	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	12/9/2011 12:50	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	12/9/2011 12:50	4.95	12680.0	France

536638 rows × 8 columns

UPDATED Dataset Overview: Total Rows and Columns

```
#@title UPDATED Dataset Overview: Total Rows and Columns
raw.shape
```

```
(536638, 8)
```

[InvoiceNo] C's: checking rows starting with C's

```
#@title [InvoiceNo] C's: checking rows starting with C's
raw[raw['InvoiceNo'].str[0] == 'C']
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
141	C536379	D	Discount	-1	12/1/2010 9:41	27.50	14527.0	United Kingdom
154	C536383	35004C	SET OF 3 COLOURED FLYING DUCKS	-1	12/1/2010 9:49	4.65	15311.0	United Kingdom
235	C536391	22556	PLASTERS IN TIN CIRCUS PARADE	-12	12/1/2010 10:24	1.65	17548.0	United Kingdom
236	C536391	21984	PACK OF 12 PINK PAISLEY TISSUES	-24	12/1/2010 10:24	0.29	17548.0	United Kingdom
237	C536391	21983	PACK OF 12 BLUE PAISLEY TISSUES	-24	12/1/2010 10:24	0.29	17548.0	United Kingdom
...	...	...	...	...	...	...	...	...
540449	C581490	23144	ZINC T-LIGHT HOLDER STARS SMALL	-11	12/9/2011 9:57	0.83	14397.0	United Kingdom
541541	C581499	M	Manual	-1	12/9/2011 10:28	224.69	15498.0	United Kingdom

OBSERVATION

As established from the beginning, InvoiceNo starting in 'c's indicates ***Cancellation***, hence:

9251 rows were Cancelled Transactions

// COLUMN [StockCode]

Column [StockCode]: Examining Column

```
#@title Column [StockCode]: Examining Column
raw['StockCode'].str[0].value_counts()
```

StockCode	count
2	434001
8	62460
4	11373
1	7578
7	7151
3	5705
9	4638
P	1260
D	832
5	633
M	566
C	160
6	113
S	62
B	37
A	34
g	34
m	1

dtype: int64

[StockCode] P's: checking rows starting with P's

```
#@title [StockCode] P's: checking rows starting with P's
raw[raw['StockCode'].str[0] == 'P'].head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
45	536370	POST	POSTAGE	3	12/1/2010 8:45	18.0	12583.0	France
386	536403	POST	POSTAGE	1	12/1/2010 11:27	15.0	12791.0	Netherlands
1123	536527	POST	POSTAGE	1	12/1/2010 13:04	18.0	12662.0	Germany
5073	536840	POST	POSTAGE	1	12/2/2010 18:27	18.0	12738.0	Germany
5258	536852	POST	POSTAGE	1	12/3/2010 9:51	18.0	12686.0	France

[StockCode] D's: checking rows starting with D's

```
#@title [StockCode] D's: checking rows starting with D's
raw[raw['StockCode'].str[0] == 'D']
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
141	C536379	D	Discount	-1	12/1/2010 9:41	27.50	14527.0	United Kingdom
1814	536544	DOT	DOTCOM POSTAGE	1	12/1/2010 14:32	569.77	NA	United Kingdom
3041	536592	DOT	DOTCOM POSTAGE	1	12/1/2010 17:06	607.49	NA	United Kingdom
5450	536862	DOT	DOTCOM POSTAGE	1	12/3/2010 11:13	254.43	NA	United Kingdom
5545	536864	DOT	DOTCOM POSTAGE	1	12/3/2010 11:27	121.06	NA	United Kingdom
...	...	...	...	...	...	...	...	...
536834	581219	DOT	DOTCOM POSTAGE	1	12/8/2011 9:28	1008.96	NA	United Kingdom
537254	581238	DOT	DOTCOM POSTAGE	1	12/8/2011 10:53	1683.75	NA	United Kingdom
539368	581439	DOT	DOTCOM POSTAGE	1	12/8/2011 16:30	938.59	NA	United Kingdom
540908	581492	DOT	DOTCOM POSTAGE	1	12/9/2011 10:03	933.17	NA	United Kingdom
541540	581498	DOT	DOTCOM POSTAGE	1	12/9/2011 10:26	1714.17	NA	United Kingdom

832 rows × 8 columns

[StockCode] M's: checking rows starting with M's

```
#@title [StockCode] M's: checking rows starting with M's
raw[raw['StockCode'].str[0] == 'M'].head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
2239	536569	M	Manual	1	12/1/2010 15:35	1.25	16274.0	United Kingdom
2250	536569	M	Manual	1	12/1/2010 15:35	18.95	16274.0	United Kingdom
5684	536865	M	Manual	1	12/3/2010 11:28	2.55	NA	United Kingdom
6798	536981	M	Manual	2	12/3/2010 14:26	0.85	14723.0	United Kingdom
7976	537077	M	Manual	12	12/5/2010 11:59	0.42	17062.0	United Kingdom

[StockCode] C's: checking rows starting with C's

```
#@title [StockCode] C's: checking rows starting with C's
raw[raw['StockCode'].str[0] == 'C'].head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1423	536540	C2	CARRIAGE	1	12/1/2010 14:05	50.0	14911.0	EIRE
12119	537368	C2	CARRIAGE	1	12/6/2010 12:40	50.0	14911.0	EIRE
12452	537378	C2	CARRIAGE	1	12/6/2010 13:06	50.0	14911.0	EIRE
19975	537963	C2	CARRIAGE	1	12/9/2010 11:30	50.0	13369.0	United Kingdom
20016	538002	C2	CARRIAGE	1	12/9/2010 11:48	50.0	14932.0	Channel Islands

[StockCode] S's: checking rows starting with S's

```
#@title [StockCode] S's: checking rows starting with S's
raw[raw['StockCode'].str[0] == 'S'].head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
14436	C537581	S	SAMPLES	-1	12/7/2010 12:03	12.95	NA	United Kingdom
14437	C537581	S	SAMPLES	-1	12/7/2010 12:03	52.00	NA	United Kingdom
96680	C544580	S	SAMPLES	-1	2/21/2011 14:25	5.74	NA	United Kingdom
96681	C544580	S	SAMPLES	-1	2/21/2011 14:25	11.08	NA	United Kingdom
96682	C544580	S	SAMPLES	-1	2/21/2011 14:25	5.79	NA	United Kingdom

[StockCode] B's: checking rows starting with B's

```
#@title [StockCode] B's: checking rows starting with B's
raw[raw['StockCode'].str[0] == 'B'].head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
4406	536779	BANK CHARGES	Bank Charges	1	12/2/2010 15:08	15.00	15823.0	United Kingdom
14435	C537572	BANK CHARGES	Bank Charges	-1	12/7/2010 12:00	95.38	NA	United Kingdom
28992	C538680	BANK CHARGES	Bank Charges	-1	12/13/2010 17:10	966.92	NA	United Kingdom
62508	541505	BANK CHARGES	Bank Charges	1	1/18/2011 15:58	15.00	15939.0	United Kingdom
64573	C541653	BANK CHARGES	Bank Charges	-1	1/20/2011 11:50	1050.15	NA	United Kingdom

[StockCode] A's: checking rows starting with A's

```
#@title [StockCode] A's: checking rows starting with A's
raw[raw['StockCode'].str[0] == 'A'].head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
14514	C537600	AMAZONFEE	AMAZON FEE	-1	12/7/2010 12:41	1.00	NA	United Kingdom
15016	C537630	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:04	13541.33	NA	United Kingdom
15017	537632	AMAZONFEE	AMAZON FEE	1	12/7/2010 15:08	13541.33	NA	United Kingdom
16232	C537644	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:34	13474.79	NA	United Kingdom
16313	C537647	AMAZONFEE	AMAZON FEE	-1	12/7/2010 15:41	5519.25	NA	United Kingdom

[StockCode] g's: checking rows starting with g's

```
#@title [StockCode] g's: checking rows starting with g's
raw[raw['StockCode'].str[0] == 'g'].head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
38248	539492	gift_0001_40	Dotcomgiftshop Gift Voucher £40.00	1	12/20/2010 10:14	34.04	NA	United Kingdom
42057	539958	gift_0001_50	Dotcomgiftshop Gift Voucher £50.00	1	12/23/2010 13:26	42.55	NA	United Kingdom
44725	540238	gift_0001_30	Dotcomgiftshop Gift Voucher £30.00	1	1/5/2011 14:44	25.53	NA	United Kingdom
			Dotcomgiftshop Gift Voucher					United

[StockCode] m's: checking rows starting with m's

```
#@title [StockCode] m's: checking rows starting with m's
raw[raw['StockCode'].str[0] == 'm'].head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
40383	539736	m	Manual	1	12/21/2010 15:18	2.55	NA	United Kingdom

NOTES: Not being sales transactions, 9 Alphameric StockCode values shall be removed:

P - Postage D - Dotcom postage M - Manual C - Carriage, CRUK Commission S - Samples B - Bank Charges A - Amazon fees g - gift vouchers m - Manual

[StockCode]: removing rows beginning with 'P', 'D' ++

```
#@title [StockCode]: removing rows beginning with 'P', 'D' ++
exclude_prefix = ['P', 'D', 'M', 'C', 'S', 'B', 'A', 'g', 'm']
raw = raw[~raw['StockCode'].str[0].isin(exclude_prefix)]
```

UPDATED Dataset Overview: Total Rows and Columns

```
#@title UPDATED Dataset Overview: Total Rows and Columns
raw.shape
```

```
(533652, 8)
```

UPDATED Dataset Overview: Columns with 'object'

```
#@title UPDATED Dataset Overview: Columns with 'object'
raw.describe(include='object')
```

	InvoiceNo	StockCode	Description	InvoiceDate	CustomerID	Country
count	533652	533652	533652	533652	533652	533652
unique	25284	4037	4199	22765	4364	38
top	573585	85123A	WHITE HANGING HEART T-LIGHT HOLDER	10/31/2011 14:41	NA	United Kingdom
freq	1112	2301	2357	1112	133963	488632

OBSERVATION

- 533,652 total rows
- 4,037 [StockCode] unique rows
- 25,284 [InvoiceNo] unique rows

- (-) negative value -80,995.00 [Quantity] as minimum value: **INVESTIGATE**

UPDATED Dataset Overview: Columns with values

```
#@title UPDATED Dataset Overview: Columns with values
raw.describe()
```

	Quantity	UnitPrice
count	533652.000000	533652.000000
mean	9.654949	3.277232
std	219.688312	4.507985
min	-80995.000000	0.000000
25%	1.000000	1.250000
50%	3.000000	2.080000
75%	10.000000	4.130000
max	80995.000000	649.500000

```
raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 533652 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    533652 non-null  object
 1   StockCode    533652 non-null  object
 2   Description  533652 non-null  object
 3   Quantity     533652 non-null  int64
 4   InvoiceDate  533652 non-null  object
 5   UnitPrice    533652 non-null  float64
 6   CustomerID   533652 non-null  object
 7   Country      533652 non-null  object
dtypes: float64(1), int64(1), object(6)
memory usage: 36.6+ MB
```

COLUMN [UnitPrice]

[UnitPrice]: counting "0" values

```
#@title [UnitPrice]: counting "0" values
raw[raw['UnitPrice'] == 0][['UnitPrice']].count()
```

	0
UnitPrice	2480

dtype: int64

OBSERVATION

- 2,480 rows have "0" [UnitPrice]
- being irrelevant to the sales transaction analysis, these will be removed

[UnitPrice]: removing "0" values

```
#@title [UnitPrice]: removing "0" values
raw = raw[raw['UnitPrice'] != 0]
```

```
raw.describe()
```

	Quantity	UnitPrice
count	531172.000000	531172.000000
mean	9.961487	3.292533
std	217.014514	4.512918
min	-80995.000000	0.030000
25%	1.000000	1.250000
50%	3.000000	2.080000
75%	10.000000	4.130000
max	80995.000000	649.500000

```
raw.shape
```

```
(531172, 8)
```

[UnitPrice]: checking price-level accounting*

```
#@title [UnitPrice]: checking price-level accounting*
'''Price-Level Accounting = take into account changes in price levels, thus avoiding some of the criticisms of historical-cost

unitprice_level = raw.groupby(['StockCode'])['UnitPrice'].nunique().reset_index()
unitprice_level[unitprice_level['UnitPrice'] > 1].sort_values(by='UnitPrice')
```

	StockCode	UnitPrice
3619	90039A	2
3629	90043	2
3632	90049	2
3633	90050	2
3636	90053	2
...	...	...
1034	22111	13
413	21175	13
407	21166	14
2851	79321	15
2675	47566	15

3361 rows × 2 columns

OBSERVATION

- Price-Level Accounting = taking into account changes in price levels (oxfordreference.com)
- 3,361 [StockCode] units have more than 1 unitprice value

[UnitPrice]: previewing price-level accounts for [StockCode] "90039A"

```
#@title [UnitPrice]: previewing price-level accounts for [StockCode] "90039A"
raw[raw['StockCode'] == '90039A'][['StockCode','UnitPrice']]
```

	StockCode	UnitPrice
55693	90039A	3.73
187814	90039A	3.75
528765	90039A	3.73

[UnitPrice]: previewing price-level accounts for [StockCode] "90053"

```
#@title [UnitPrice]: previewing price-level accounts for [StockCode] "90053"
raw[raw['StockCode'] == '90053'][['StockCode','UnitPrice']]
```

	StockCode	UnitPrice
7277	90053	2.55
209019	90053	2.48
243730	90053	2.48
385196	90053	2.48

```
unitprice_level['UnitPrice'].sort_values().value_counts()
```

UnitPrice	count
4	866
3	822
1	553
5	531
2	526
6	292
7	158
8	71
9	36
10	23
11	20
12	10
13	3
15	2
14	1

dtype: int64

OBSERVATIONS

- Only 533 [StockCode] have 1 unitprice value; the rest has varying unitprice values

COLUMN [Quantity]

[Quantity]: counting negative values

```
#@title [Quantity]: counting negative values
raw[raw['Quantity'] < 0][['Quantity']].count()
```

	0
Quantity	9992

dtype: int64

[Quantity]: counting positive values

```
#@title [Quantity]: counting positive values
raw[raw['Quantity'] > 0][['Quantity']].count()
```

	0
Quantity	523660

dtype: int64

OBSERVATIONS

- 9,992 [Quantity] rows have negative value
- There were negative [Quantity] values observed in [InvoiceNo] beginning with 'C' denoting Cancelled Transactions (from previous notes)
- **INVESTIGATE** link between negative [Quantity] and cancelled transactions labelled in [InvoiceNo]

[Quantity]: reviewing [InvoiceNo] with 'C' in the current updated datset

```
#@title [Quantity]: reviewing [InvoiceNo] with 'C' in the current updated datset
print(raw['InvoiceNo'].str[0].value_counts())
print()
```

```
InvoiceNo
5    524984
C      8668
Name: count, dtype: int64
```

OBSERVATION

- 8,668 rows are Cancelled Transactions in the current updated dataset
- 9,992 rows are Negative [Quantity] values
- Hence, there are about 1,324 rows that are (-) in qunatity but NOT having [InvoiceNo] values starting with 'C'; **INVESTIGATE** these

```
##@title [Quantity]: identifying negative [Qunatity] with non 'C' [InvoiceNo]
filtered_negative_non_c = raw[(raw['Quantity'] < 0) & (raw['InvoiceNo'].str[0] != 'C')][['Quantity','InvoiceNo', 'Description
filtered_negative_non_c
```

	Quantity	InvoiceNo	Description
2406	-10	536589	NA
4347	-38	536764	NA
7188	-20	536996	NA

Column [StockCode]: Examining Column

```
#@title Column [StockCode]: Examining Column
filtered_negative_non_c['Description'].value_counts()
```

Description	count
NA	855
check	120
damages	45
damaged	42
?	41
sold as set on dotcom	20
Damaged	14
Unsaleable, destroyed.	9
thrown away	9
??	7
wet damaged	5
damages?	5
smashed	4
CHECK	3
missing	3
wet pallet	3
mixed up	2
?missing	2
sold as 1	2
incorrect stock entry.	2