Ahmed_Hasan_Askar_techno

November 29, 2023

1 Pharmaceutical Sales Analysis Notebook

This notebook is dedicated for the description of *pharmaceutical-sales.csv* dataset, which is available at:

https://www.kaggle.com/datasets/bakasas/pharmaceutical-sales/

1.1 1. Importing needed libraries and the dateset

[99]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[100]:
```
data = pd.read_csv('pharmaceutical-sales.csv', sep = ';',␣
 ↪index_col=['product_code', 'check_no'])
data.head()
```

[100]:

product_code	check_no	date	customer_id	customer_gender	customer_age	chain	store_code	store_name	category_1	category_2	category_3	qty	sale
13006	177911	24.10.2021	70478	Woman	40	A	8	Name 8	Medication	1	27	0,15	9,75
8593	40144	13.09.2021	87806	Woman	41	A	0	Name 0	NonMedication	4			
11705	271220	20.11.2021	61579	Woman	28	B	7	Name 7	NonMedication	12			
10068	279977	21.11.2021	102389	Woman	48	B	5	Name 5	NonMedication	3			
6395	170982	22.10.2021	3396	Woman	40	A	3	Name 3	NonMedication	13			

8593	40144	56	0,032	133,65	
11705	271220	172	1	2 784,38	
10068	279977	49	1	1 386,00	
6395	170982	183	10	371,25	

1.2 2. Dataset overview

[101]:
```
data.shape
```

[101]:
```
(116686, 12)
```

[102]:
```
data.describe()
```

[102]:

	customer_id	customer_age	store_code	category_2	category_3
count	116686.000000	116686.000000	116686.000000	116686.000000	116686.000000
mean	65892.503205	43.883525	3.920505	3.717370	60.743063
std	37722.358749	14.587512	2.557146	4.882389	65.231528
min	1.000000	3.000000	0.000000	0.000000	0.000000
25%	32541.250000	32.000000	2.000000	1.000000	18.000000
50%	67308.000000	41.000000	4.000000	1.000000	32.000000
75%	97478.000000	54.000000	6.000000	4.000000	81.000000
max	130938.000000	97.000000	8.000000	18.000000	240.000000

[103]:
```
data.head()
```

[103]:

product_code	check_no	date	customer_id	customer_gender	customer_age	chain	store_code	store_name	category_1	category_2
13006	177911	24.10.2021	70478	Woman	40	A	8	Name 8	Medication	1
8593	40144	13.09.2021	87806	Woman	41					
11705	271220	20.11.2021	61579	Woman	28					
10068	279977	21.11.2021	102389	Woman	48					
6395	170982	22.10.2021	3396	Woman	40					

```
8593          40144         A             0      Name 0  NonMedication            4
11705         271220        B             7      Name 7  NonMedication           12
10068         279977        B             5      Name 5  NonMedication            3
6395          170982        A             3      Name 3  NonMedication           13
```

product_code	check_no	category_3	qty	sale
13006	177911	27	0,15	9,75
8593	40144	56	0,032	133,65
11705	271220	172	1	2 784,38
10068	279977	49	1	1 386,00
6395	170982	183	10	371,25

[104]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 116686 entries, (13006, 177911) to (6895, 192794)
Data columns (total 12 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   date             116686 non-null  object
 1   customer_id      116686 non-null  int64
 2   customer_gender  116686 non-null  object
 3   customer_age     116686 non-null  int64
 4   chain            116686 non-null  object
 5   store_code       116686 non-null  int64
 6   store_name       116686 non-null  object
 7   category_1       116686 non-null  object
 8   category_2       116686 non-null  int64
 9   category_3       116686 non-null  int64
 10  qty              116686 non-null  object
 11  sale             116686 non-null  object
dtypes: int64(5), object(7)
memory usage: 14.4+ MB
```

Its clear than there are no null values. However,**'date'**, **'qty'** and **'sale'** columns are not of the right data type.

1.3 3. Dataset cleansing

###

Adjusting columns data types

[105]:
```
data2 = pd.read_csv('pharmaceutical-sales.csv', sep = ';',␣
 ↪index_col=['product_code', 'check_no'])
```

Its preferable to copy data to another dataframe to keep original data intact. The columns **'qty'** and **'sale'** should be proberly adjusted for sake of changing their data type to float.

[106]:
```
data2['qty'] = data2['qty'].replace(',', '.', regex = True)
data2['qty'] = data2['qty'].replace(' ', '', regex = True)
data2['sale'] = data2['sale'].replace(',', '.', regex = True)
data2['sale'] = data2['sale'].replace(' ', '', regex = True)

data2.head()
```

[106]:

product_code	check_no	date	customer_id	customer_gender	customer_age	chain	store_code	store_name	category_1	category_2	category_3	qty	sale
13006	177911	24.10.2021	70478	Woman	40	A	8	Name 8	Medication	1	27	0.15	9.75
8593	40144	13.09.2021	87806	Woman	41	A	0	Name 0	NonMedication	4	56	0.032	133.65
11705	271220	20.11.2021	61579	Woman	28	B	7	Name 7	NonMedication	12	172	1	2784.38
10068	279977	21.11.2021	102389	Woman	48	B	5	Name 5	NonMedication	3	49	1	1386.00
6395	170982	22.10.2021	3396	Woman	40	A	3	Name 3	NonMedication	13	183	10	371.25

[107]:
```
data2['qty'] = pd.to_numeric(data2['qty'], errors='coerce')
data2['sale'] = pd.to_numeric(data2['sale'], errors='coerce')
```

'date' column also should be adjusted.

[108]:
```
data2.date = pd.to_datetime(data2.date, format = "%d.%m.%Y")
```

[109]:
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 116686 entries, (13006, 177911) to (6895, 192794)
Data columns (total 12 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   date             116686 non-null  datetime64[ns]
```

```
 1   customer_id      116686 non-null  int64
 2   customer_gender  116686 non-null  object
 3   customer_age     116686 non-null  int64
 4   chain            116686 non-null  object
 5   store_code       116686 non-null  int64
 6   store_name       116686 non-null  object
 7   category_1       116686 non-null  object
 8   category_2       116686 non-null  int64
 9   category_3       116686 non-null  int64
 10  qty              116686 non-null  float64
 11  sale             116007 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(5), object(4)
memory usage: 14.4+ MB
```

###

The next step is to remove null values.

[110]:
```
data2.isna().sum()
```

[110]:
```
date                0
customer_id         0
customer_gender     0
customer_age        0
chain               0
store_code          0
store_name          0
category_1          0
category_2          0
category_3          0
qty                 0
sale              679
dtype: int64
```

[111]:
```
data2.dropna(subset=['sale'], inplace = True)
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 116007 entries, (13006, 177911) to (6895, 192794)
Data columns (total 12 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   date             116007 non-null  datetime64[ns]
 1   customer_id      116007 non-null  int64
 2   customer_gender  116007 non-null  object
 3   customer_age     116007 non-null  int64
 4   chain            116007 non-null  object
 5   store_code       116007 non-null  int64
```

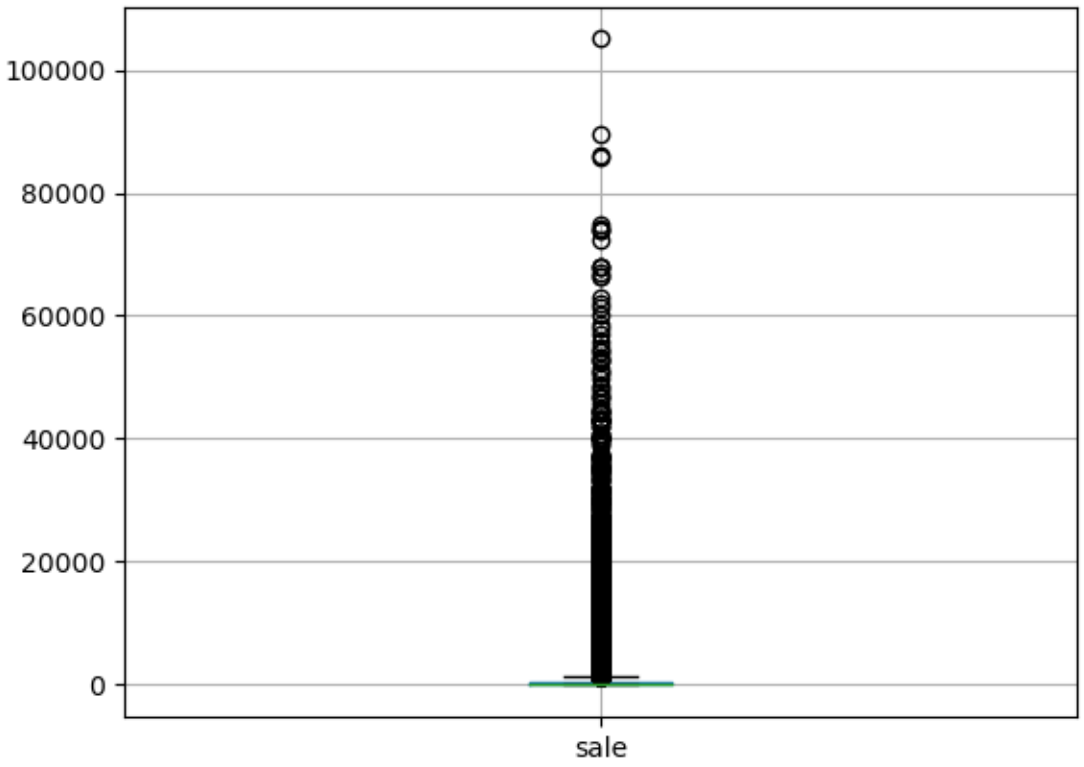
```
 6   store_name       116007 non-null  object
 7   category_1       116007 non-null  object
 8   category_2       116007 non-null  int64
 9   category_3       116007 non-null  int64
 10  qty              116007 non-null  float64
 11  sale             116007 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(5), object(4)
memory usage: 14.3+ MB
```

All null values removed!

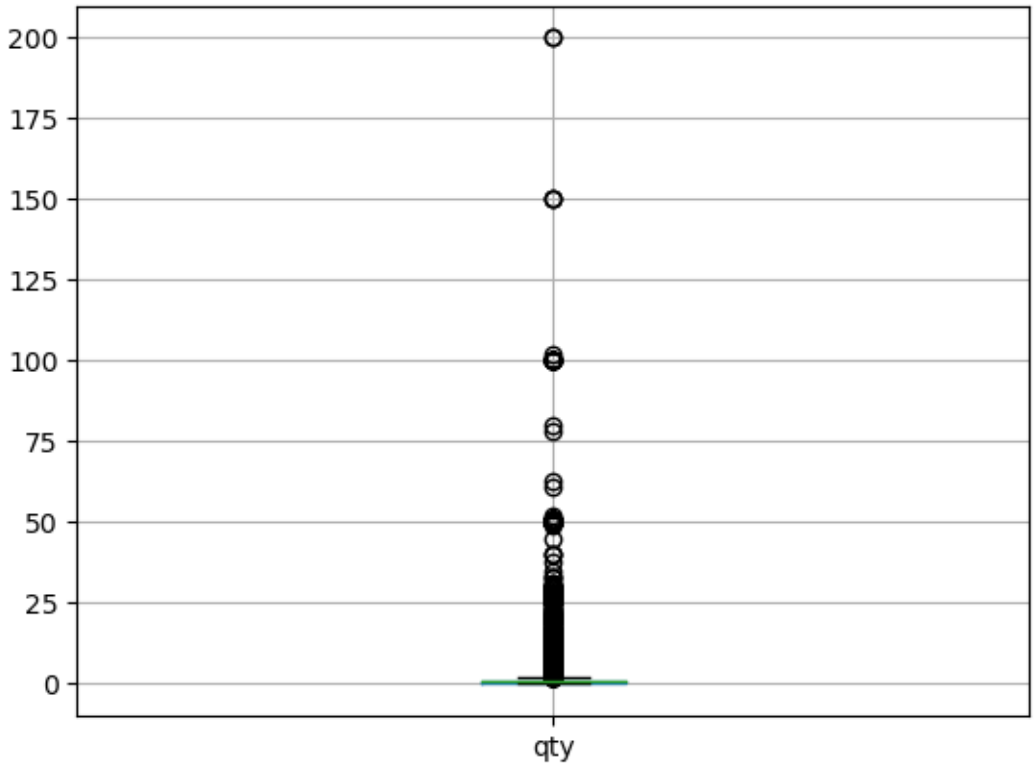
###

Removing Outliers

[112]:
```
boxplot = data2.boxplot( column = ['sale'])
```



[113]:
```
boxplot = data2.boxplot( column = ['qty'])
```



The outlies data should be assessed and counted to determine the effect of their removal on data integrity.

[114]:
```
from scipy import stats
```

[115]:
```
qty_z = np.abs(stats.zscore(data2['qty']))
sale_z = np.abs(stats.zscore(data2['sale']))
```

[116]:
```
print('Number of rows in "qty" column with z-score more than 2: {}'.
  ↪format((qty_z > 2).sum()))
print('Number of rows in "sale" column with z-score more than 2: {}'.
  ↪format((sale_z > 2).sum()))
print('Number of total rows that should be deleted: {}'.format(len(data2.
  ↪index[np.where((qty_z > 2) | (sale_z > 2))])))
```

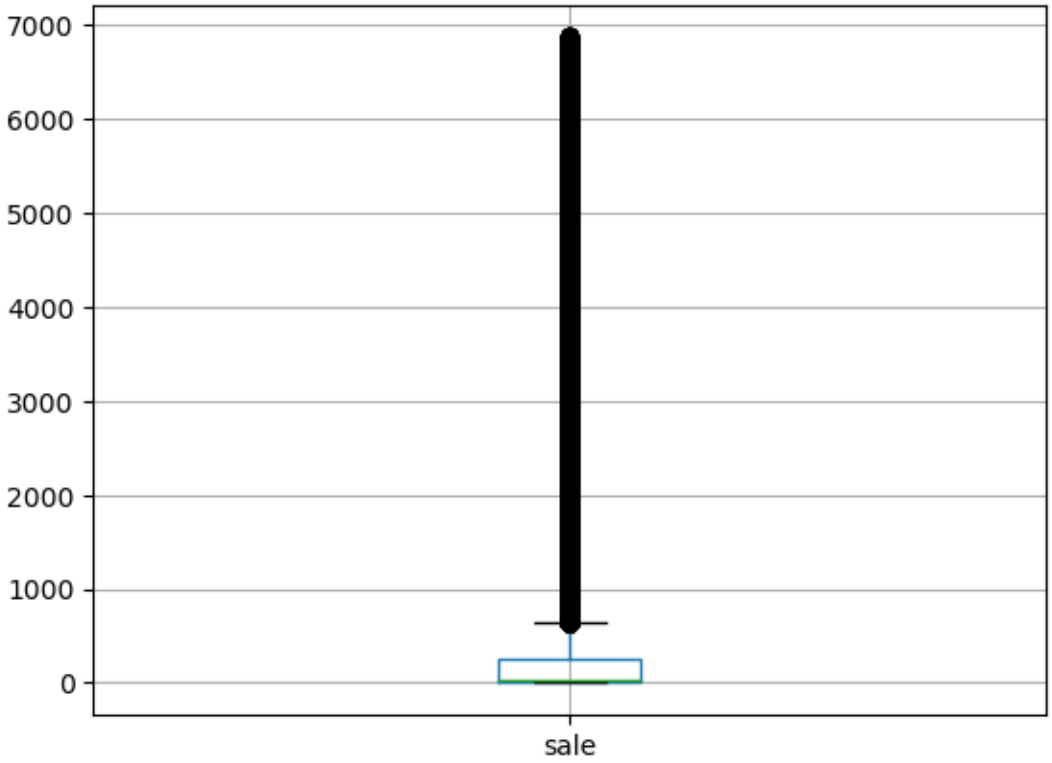
```
Number of rows in "qty" column with z-score more than 2: 1676
Number of rows in "sale" column with z-score more than 2: 3965
Number of total rows that should be deleted: 5615
```

Those rows could be removed, relatively to the data size.

[117]:
```
data2.drop(data2.index[np.where((qty_z > 2) | (sale_z > 2))], inplace = True)
```

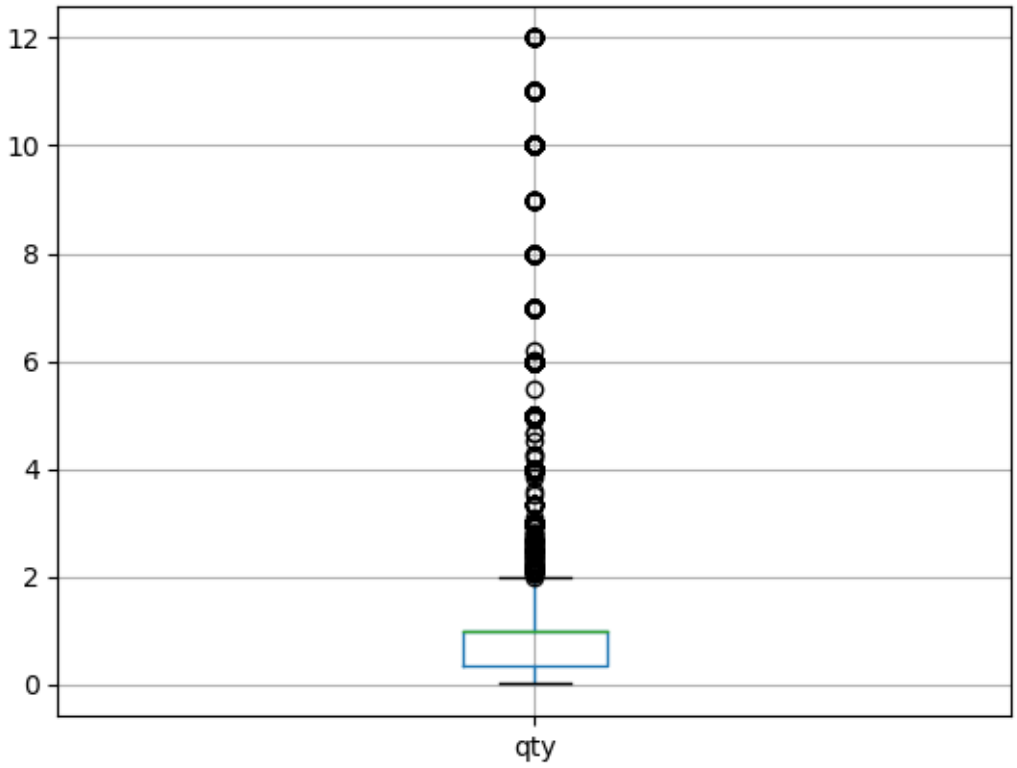
[118]:
```
data2.boxplot( column = ['sale'])
```

[118]: <AxesSubplot:>



[119]:
```
data2.boxplot( column = ['qty'])
```

[119]: <AxesSubplot:>



Still there are outliers in the data, which declares the necessity of further investigation of these outliers.

The outliers lie approximately above 2 in the 'qty' column and above 800 in the 'sale' column. These rows shall be removed before proceeding to next steps.

[120]:
```
data3 = data2.copy()
```

[121]:
```
data3.drop(data3.query(' qty > 2 | sale > 800').index, inplace = True)
data3.shape
```

[121]: (83266, 12)

###

Removing Duplicates

[122]:
```
data3.drop_duplicates(inplace = True)
data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
MultiIndex: 83170 entries, (13006, 177911) to (1369, 221204)
Data columns (total 12 columns):
```

```
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   date             83170 non-null  datetime64[ns]
 1   customer_id      83170 non-null  int64
 2   customer_gender  83170 non-null  object
 3   customer_age     83170 non-null  int64
 4   chain            83170 non-null  object
 5   store_code       83170 non-null  int64
 6   store_name       83170 non-null  object
 7   category_1       83170 non-null  object
 8   category_2       83170 non-null  int64
 9   category_3       83170 non-null  int64
 10  qty              83170 non-null  float64
 11  sale             83170 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(5), object(4)
memory usage: 11.2+ MB
```

Fortunately, the data didn't include many duplicates.

1.4 4. Calculation of summary statistics

Average sales amount and quantity of sold items per check:

[123]:
```
print('Average sales amount per check::{}'.format(round(data3.
  ↪groupby('check_no').sale.sum().mean(),2)))
print('Average quantity of sold items per check::{}'.format(round(data3.
  ↪groupby('check_no').qty.count().mean(),2)))
```

```
Average sales amount per check::74.02
Average quantity of sold items per check::1.17
```

Median, mode and standard deviation of the check amounts:

[124]:
```
print('Median of check amount:{}'.format(data3.groupby('check_no').sale.sum().
  ↪median()))
print('Mode of check amount:{}'.format(data3.groupby('check_no').sale.sum().
  ↪mode()[0]))
print('Standard deviation of check amount:{}'.format(data3.groupby('check_no').
  ↪sale.sum().std()))
```

```
Median of check amount:24.07
Mode of check amount:33.41
Standard deviation of check amount:143.08535358235576
```

[125]:
```
print('Percent of medication sales: {}%'.format(round((data3.
  ↪groupby('category_1')['sale'].sum()/data3.sale.sum()*100)[0],2)))
print('Percent of medication Quantity sold: {}%'.format(round((data3.
  ↪groupby('category_1')['qty'].sum()/data3.qty.sum()*100)[0],2)))
```

```
print('Percent of non-medication sales: {}%'.format(round((data3.
  ↪groupby('category_1')['sale'].sum()/data3.sale.sum()*100)[1],2)))
print('Percent of non-medication quantity sold: {}%'.format(round((data3.
  ↪groupby('category_1')['qty'].sum()/data3.qty.sum()*100)[1],2)))
```

```
Percent of medication sales: 45.23%
Percent of medication Quantity sold: 83.01%
Percent of non-medication sales: 54.77%
Percent of non-medication quantity sold: 16.99%
```

Non-medication sales represent nearly 55% of sales, although 17% only of sold items.

[ ]: