

# Stores Sales Preprocessing and Analysis

By/ Shimaa Elsayed



# The Database

The screenshot shows the Kaggle website interface. On the left is a sidebar with navigation links: Home, Competitions, Datasets (highlighted), Code, Discussions, Courses, and More. Below these are 'Your Work' and 'RECENTLY VIEWED' sections, with the first item being 'Sales data for a chain...'. The main content area displays the dataset 'Sales data for a chain of Brazilian stores' by user 'marcio486', updated 2 years ago (Version 2). It features a search bar at the top, a 'Dataset' header, and a list of tabs: Data (selected), Code (3), Discussion, Activity, and Metadata. Action buttons for 'Download (4 GB)' and 'New Notebook' are present. Below the tabs, the dataset has a 'Usability' score of 4.7 and a 'Tags' section with 'business'. A 'Description' section at the bottom explains that the data contains actual sales data for a chain of Brazilian stores, with names modified for privacy, and is made available for analysis.

www.kaggle.com/marcio486/sales-data-for-a-chain-of-brazilian-stores

**Kaggle**

Create

Home

Competitions

**Datasets**

Code

Discussions

Courses

More

Your Work

RECENTLY VIEWED

Sales data for a chain...

View Active Events

Search

Dataset

21

**Sales data for a chain of Brazilian stores**

marcio486 • updated 2 years ago (Version 2)

Data Code (3) Discussion Activity Metadata

Download (4 GB) New Notebook

Usability 4.7 Tags business

Description

This data set contains actual sales data for a chain of Brazilian stores. I modified the names of products, customers, and employees to preserve their identity. I am making this data available so that they can help me get the most out of it, analysis such as:

# The Database

	Company Code	Order Number	Employee	Product	Product Category	Client	Client City	Sale Date Time	Product Cost	Discount Amount	Amount	Total	Form of payment
0	39000	12	Stacy Day	Special Gasoline	Fuel	Customer not informed	No City	2017-03-31 04:10:00	3.050	0.0	5.642	20.02	Money
1	39000	21	Olive Stevenson	Special Diesel	Fuel	Customer not informed	No City	2017-03-31 04:13:00	2.510	0.0	125.045	350.00	Debit Card
2	39000	38	Stacy Day	Special Diesel	Fuel	Customer not informed	No City	2017-03-31 04:25:00	2.510	0.0	35.699	99.92	Money
3	39000	39	Olive Stevenson	Lubricant 1108	Lubricant	Customer not informed	No City	2017-03-31 04:26:00	7.409	0.0	1.000	13.00	Money
4	39000	39	Olive Stevenson	Diesel Auto Clean	Fuel	Customer not informed	No City	2017-03-31 04:26:00	2.560	0.0	42.162	120.96	Money
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1048570	39000	312686	Terri Massey	Special Gasoline	Fuel	Customer not informed	No City	2017-09-05 21:04:00	3.187	0.0	35.499	131.31	Money
1048571	1420000	312214	Gerardo Moore	Special Gasoline	Fuel	Customer not informed	No City	2017-09-05 21:04:00	3.215	0.0	27.035	100.00	Credit Card
1048572	1312000	312191	Phillip Fletcher	Gasoline	Fuel	Penny Burns	Krafdale	2017-09-05 21:04:00	3.163	0.0	11.692	45.00	Money
1048573	39000	312687	Rodney Patton	Gasoline	Fuel	Customer not informed	No City	2017-09-05 21:05:00	3.177	0.0	30.160	111.56	Money
1048574	11848000	312058	Robin Johnston	Special Gasoline	Fuel	Customer not informed	No City	2017-09-05 21:05:00	3.227	0.0	27.405	100.00	Money

1048575 rows × 13 columns

# Index:

## 1- data Preprocessing and cleaning.

- # null values
- # datatypes readability
- # finding outliers and removing them

## 2- sales statistics

- # max and min products sold and product prices

## 3- data visualization

- # numerical and categorical
- # product cost, amount, total, form of payment, company code



# 1- Data Preprocessing and cleaning

# **a- Checking for Null Values**



# By Checking for Null values there were no null values found

Company Code	0
Order Number	0
Employee	0
Product	0
Product Category	0
Client	0
Client City	0
Sale Date Time	0
Product Cost	0
Discount Amount	0
Amount	0
Total	0
Form of payment	0
dtype: int64	

# b- checking Data Types





# By checking data types the time and date was not recognized therefore it's recognized by code as shown

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Company Code          1048575 non-null int64
1   Order Number          1048575 non-null int64
2   Employee              1048575 non-null object
3   Product               1048575 non-null object
4   Product Category      1048575 non-null object
5   Client                1048575 non-null object
6   Client City           1048575 non-null object
7   Sale Date Time        1048575 non-null object
8   Product Cost          1048575 non-null float64
9   Discount Amount       1048575 non-null float64
10  Amount                1048575 non-null float64
11  Total                 1048575 non-null float64
12  Form of payment       1048575 non-null object
dtypes: float64(4), int64(2), object(7)
memory usage: 104.0+ MB
```



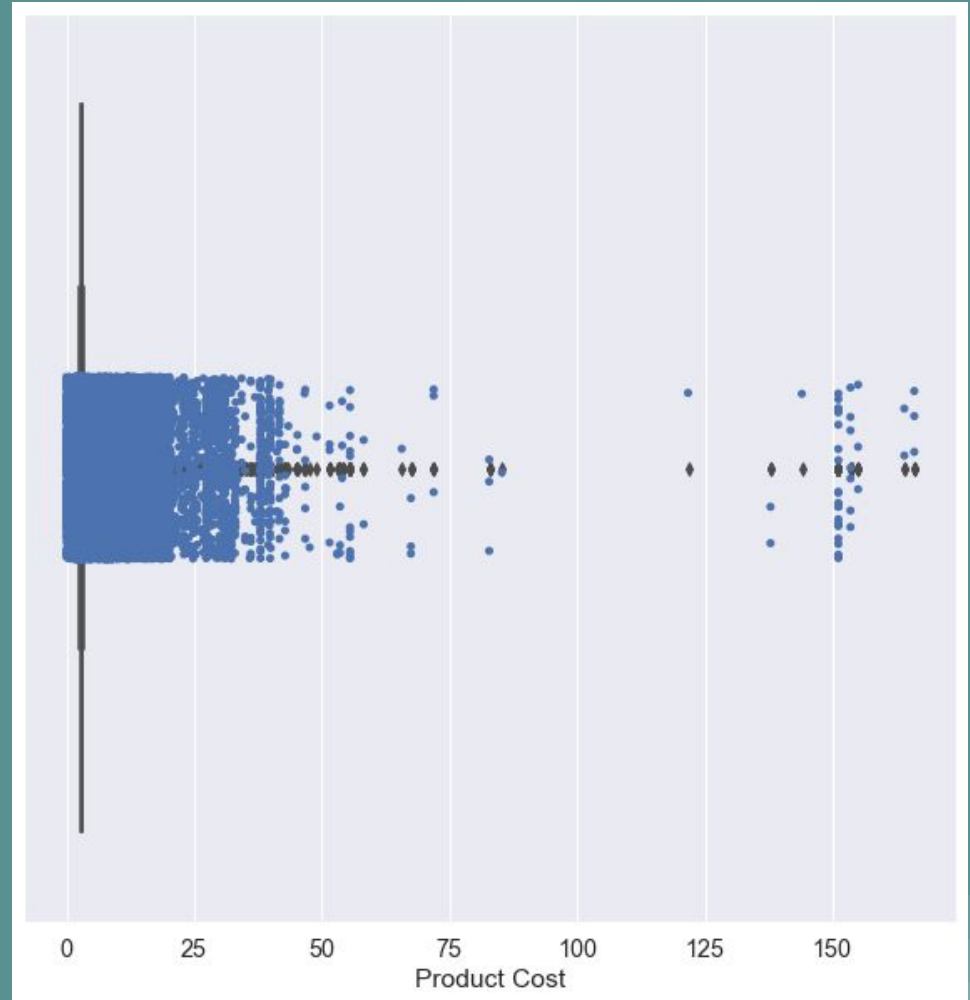
```
stores_df['Sale Date Time']=pd.to_datetime(stores_df['Sale Date Time'], format='%Y/%m/%d %H:%M:%S')
stores_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 13 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Company Code          1048575 non-null int64
1   Order Number          1048575 non-null int64
2   Employee              1048575 non-null object
3   Product               1048575 non-null object
4   Product Category      1048575 non-null object
5   Client                1048575 non-null object
6   Client City           1048575 non-null object
7   Sale Date Time        1048575 non-null datetime64[ns]
8   Product Cost          1048575 non-null float64
9   Discount Amount       1048575 non-null float64
10  Amount                1048575 non-null float64
11  Total                 1048575 non-null float64
12  Form of payment       1048575 non-null object
dtypes: datetime64[ns](1), float64(4), int64(2), object(6)
memory usage: 104.0+ MB
```

**c-checking for  
outliers and  
removing them**



The box plot for the  
“Product cost” feature  
showing outliers



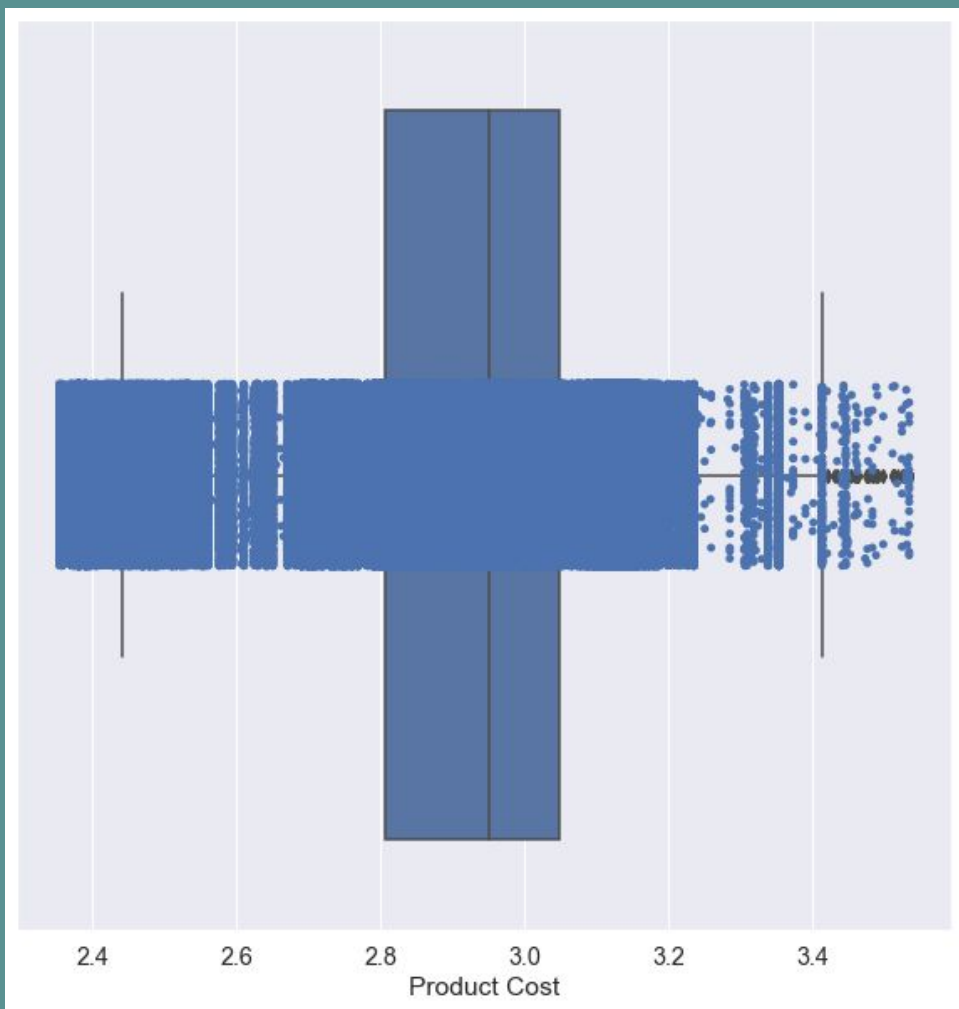
# After Removing outliers from "Product Cost" feature

```
# Replacing outliers in Product Cost
from datascist.structdata import detect_outliers

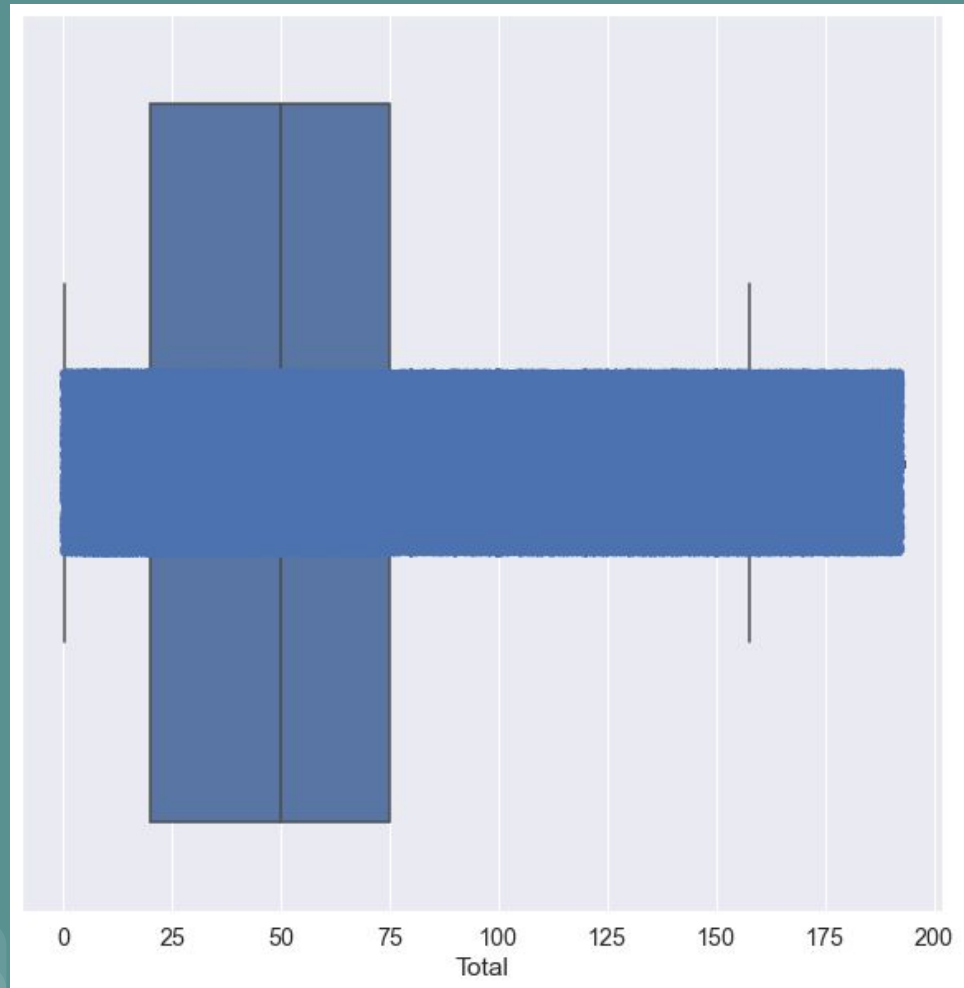
#detecting outliers in "Total Calories" column
outlier_indices=detect_outliers(stores_df,0,['Product Cost'])

#removing outliers and setting them to the median
stores_df.loc[outlier_indices,['Product Cost']]= stores_df['Product Cost'].median()

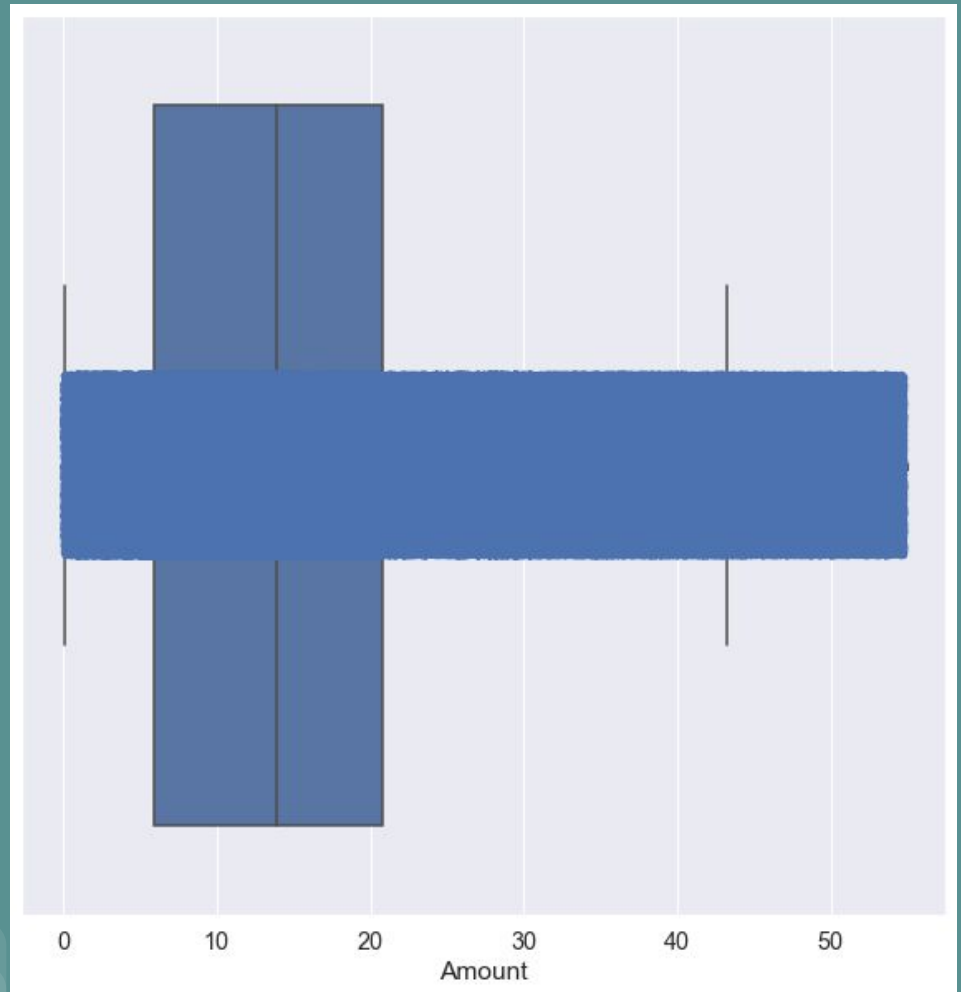
#rechecking ...
sns.boxplot(x='Product Cost',data=stores_df)
sns.stripplot(x='Product Cost',data=stores_df)
```



**After Removing outliers  
from “Total” feature**



**After Removing outliers  
from “Amount” feature**



**Replacing outliers in  
'Discount amount' would  
ruin the data since most of  
it is zero therefore the real  
values are the outliers**

Discount Amount
0.0
0.0
0.0
0.0
0.0
...
0.0
0.0
0.0
0.0

# 2- Checking the Sales Statistics





# Using the describe function

```
stores_df[['Product Cost', 'Discount Amount', 'Amount', 'Total']].describe()
```

	Product Cost	Discount Amount	Amount	Total
<b>count</b>	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
<b>mean</b>	2.917386e+00	2.935126e-04	1.569439e+01	5.557508e+01
<b>std</b>	1.802976e-01	3.516798e-02	1.239711e+01	4.284051e+01
<b>min</b>	2.354000e+00	0.000000e+00	3.800000e-02	1.300000e-01
<b>25%</b>	2.806000e+00	0.000000e+00	5.885000e+00	2.001000e+01
<b>50%</b>	2.950000e+00	0.000000e+00	1.389300e+01	5.000000e+01
<b>75%</b>	3.049000e+00	0.000000e+00	2.084000e+01	7.502000e+01
<b>max</b>	3.534000e+00	2.249000e+01	5.475000e+01	1.925100e+02

# Showing most sold products

```
stores_df.loc[stores_df['Total'] == 192.51]['Product'].unique()  
array(['Special Gasoline', 'Diesel Auto Clean'], dtype=object)
```

# Showing least sold products

```
stores_df.loc[stores_df['Total'] == .13]['Product'].unique()  
array(['Gasoline'], dtype=object)
```

# Showing most expensive products

```
stores_df.loc[stores_df['Product Cost'] == 3.534]['Product'].unique()  
array(['Several 351'], dtype=object)
```

# Showing cheapest products

```
stores_df.loc[stores_df['Product Cost'] == 2.354]['Product'].unique()  
array(['Special Diesel'], dtype=object)
```

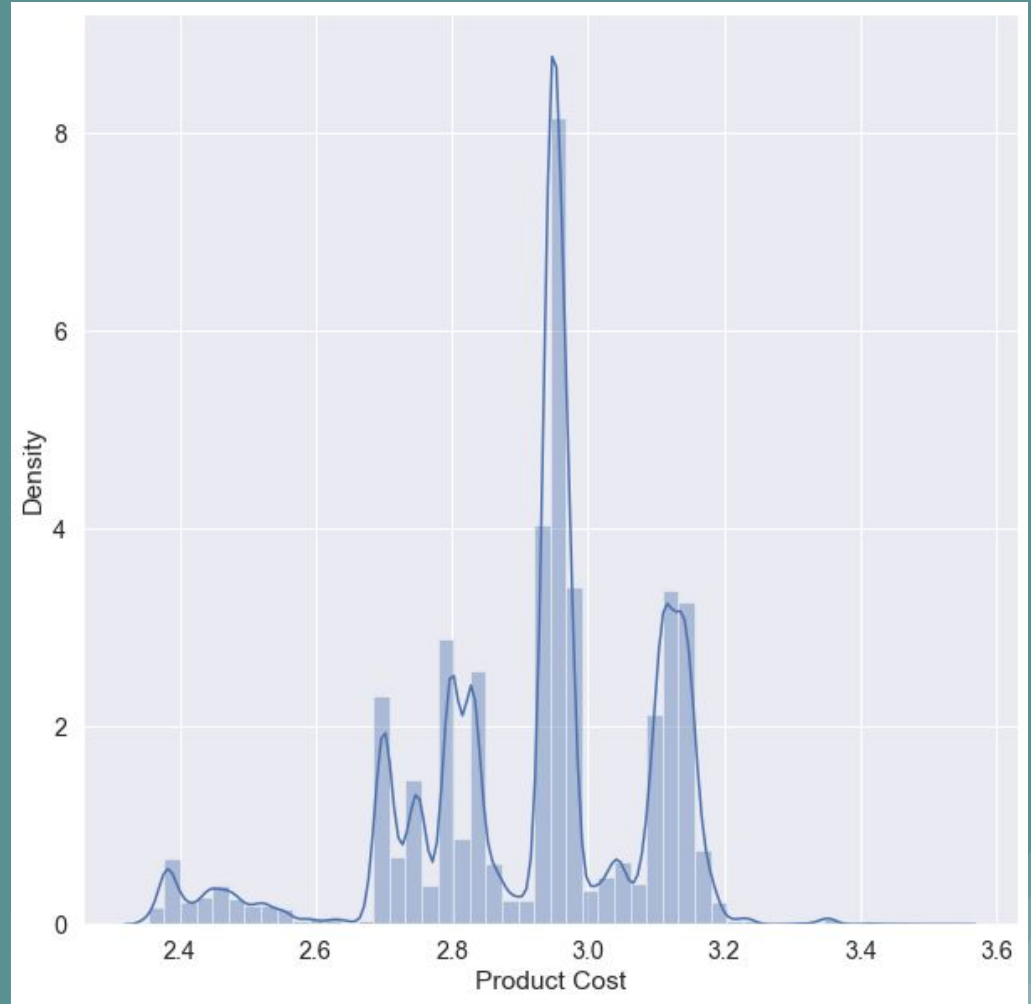
# 3- Data Visualization



# **a- Numerical Data Visualization**

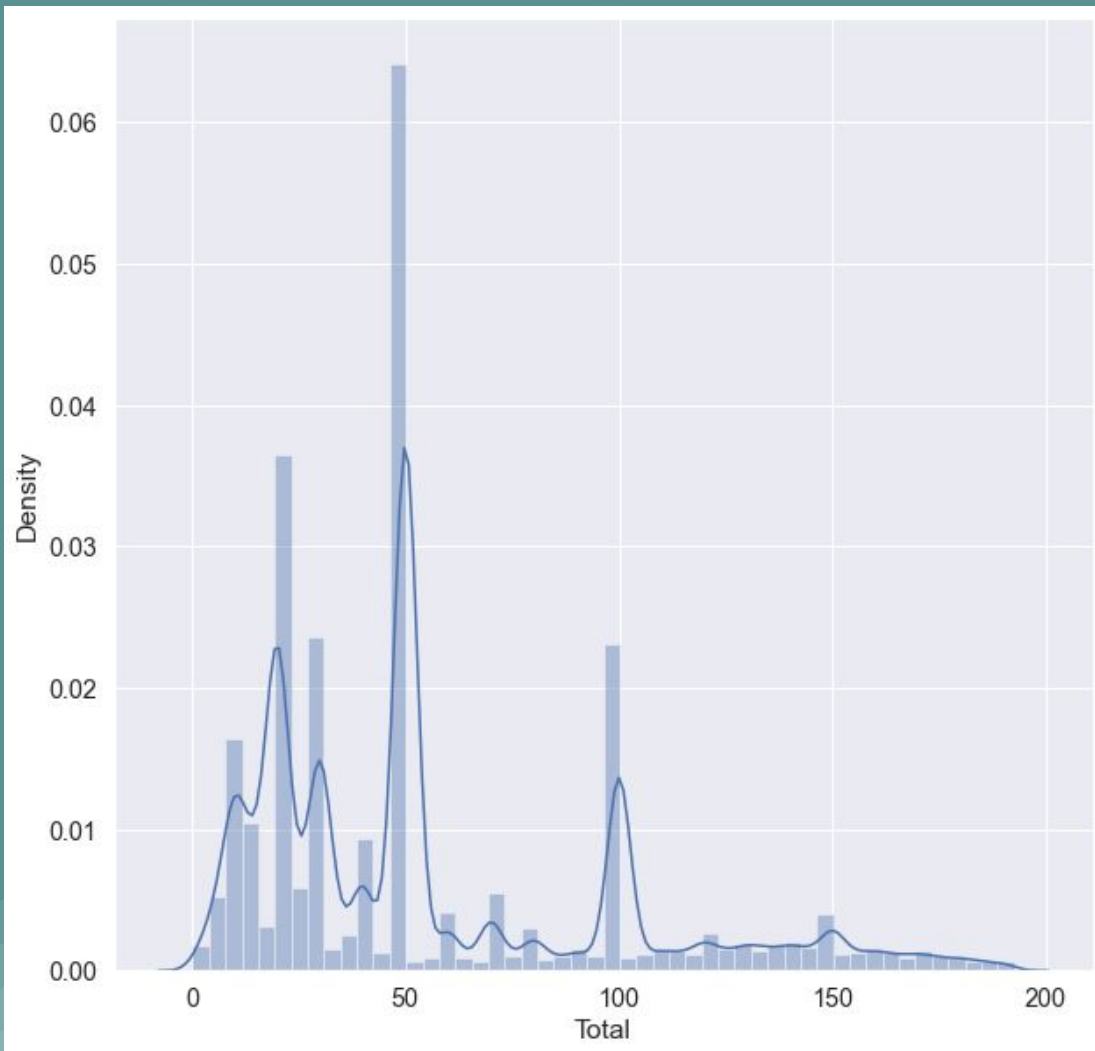


# Product Cost

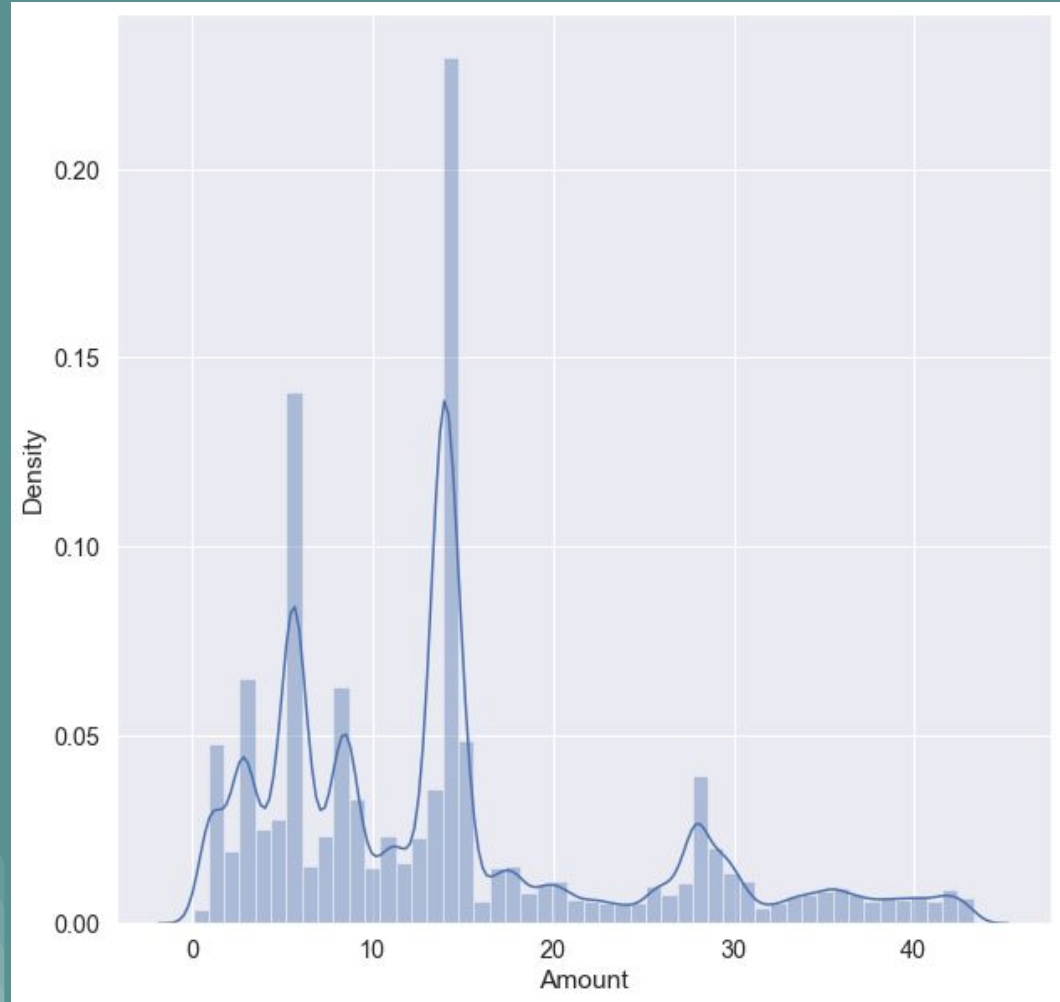




# Total Cost

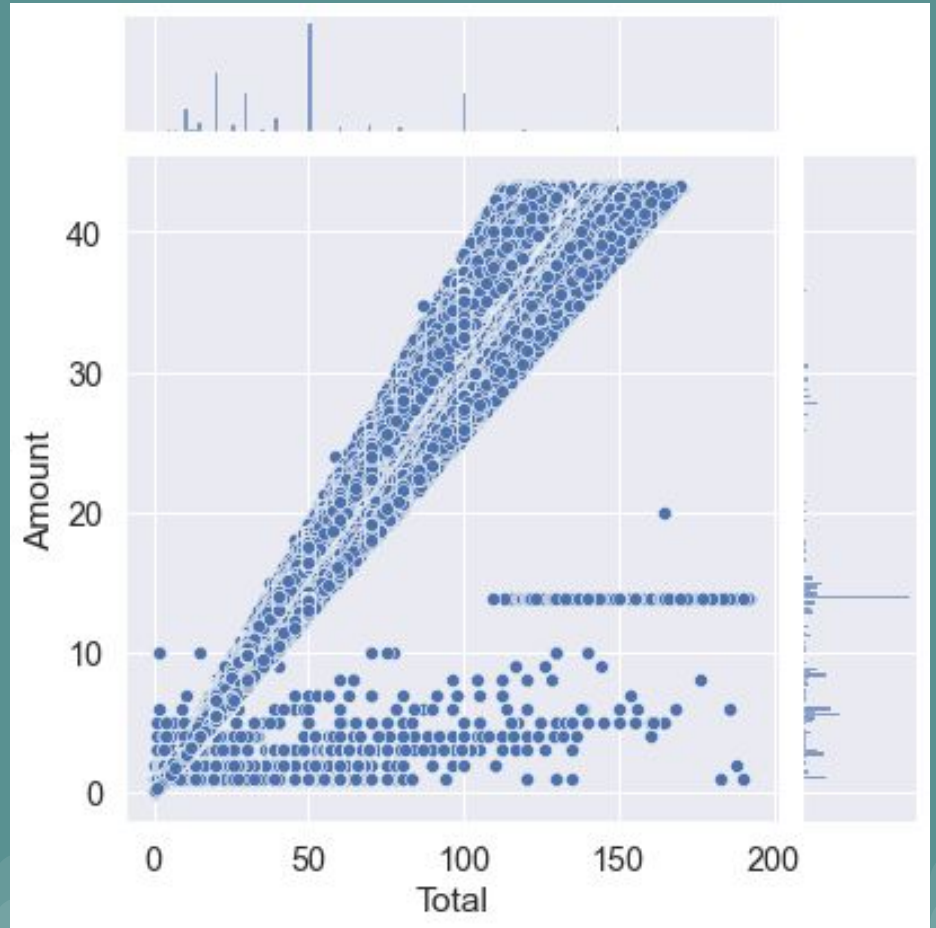


# Amount sold



# Total sold vs Amount sold

Most data is showing a directly proportional relationship as expected



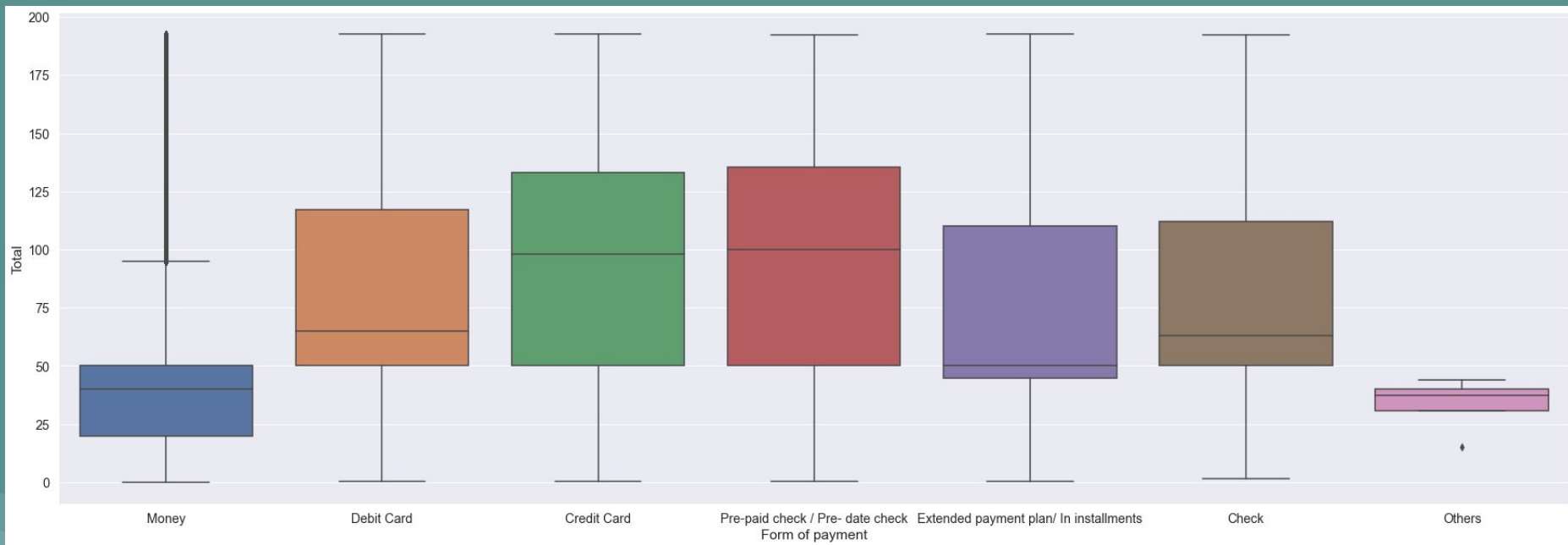
# **b- Categorical analysis**



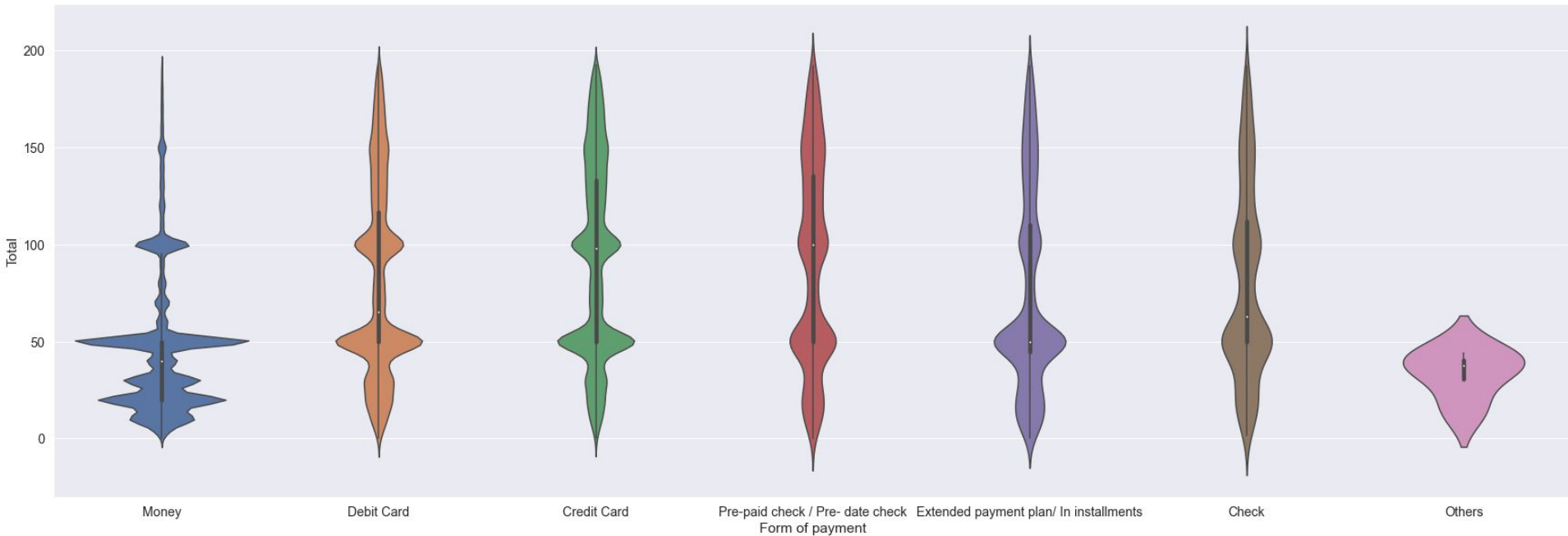
# frequency of usage of each form of payment



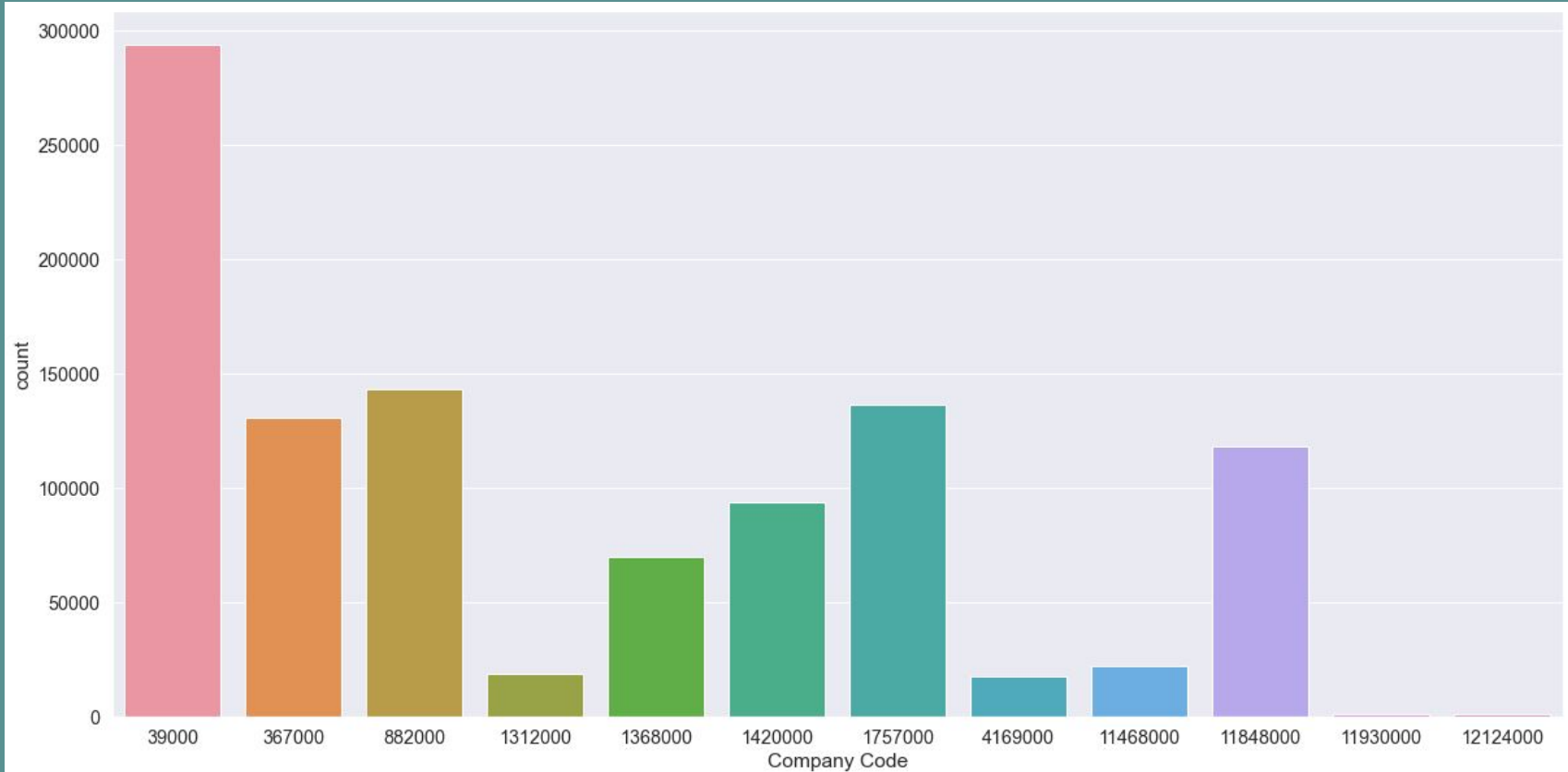
# The 5 number theory with the forms of payment with the total cost of products (box plot)



# The 5 number theory with the forms of payment with the total cost of products with distribution (violin plot)

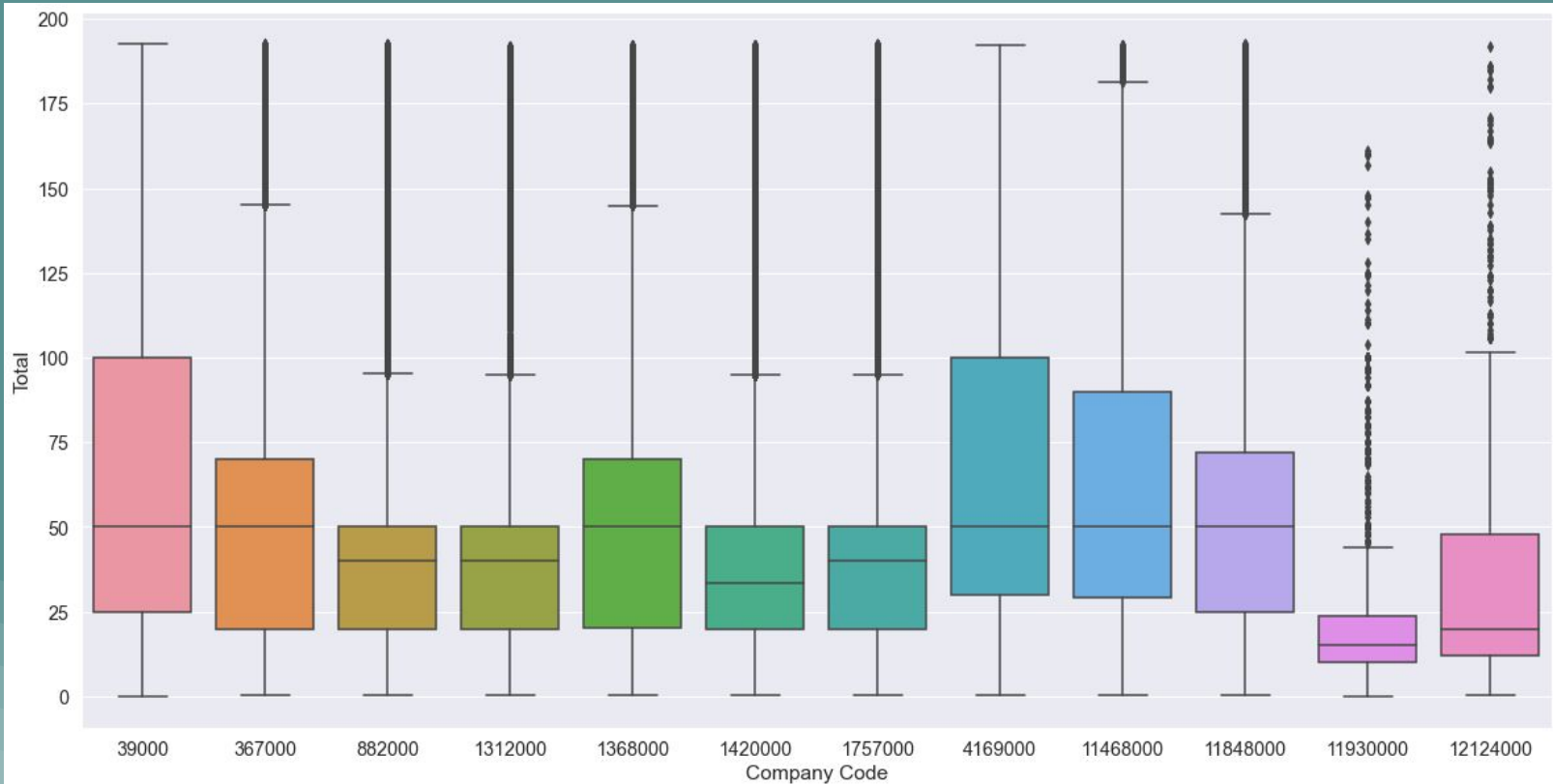


# frequency of usage of each company code

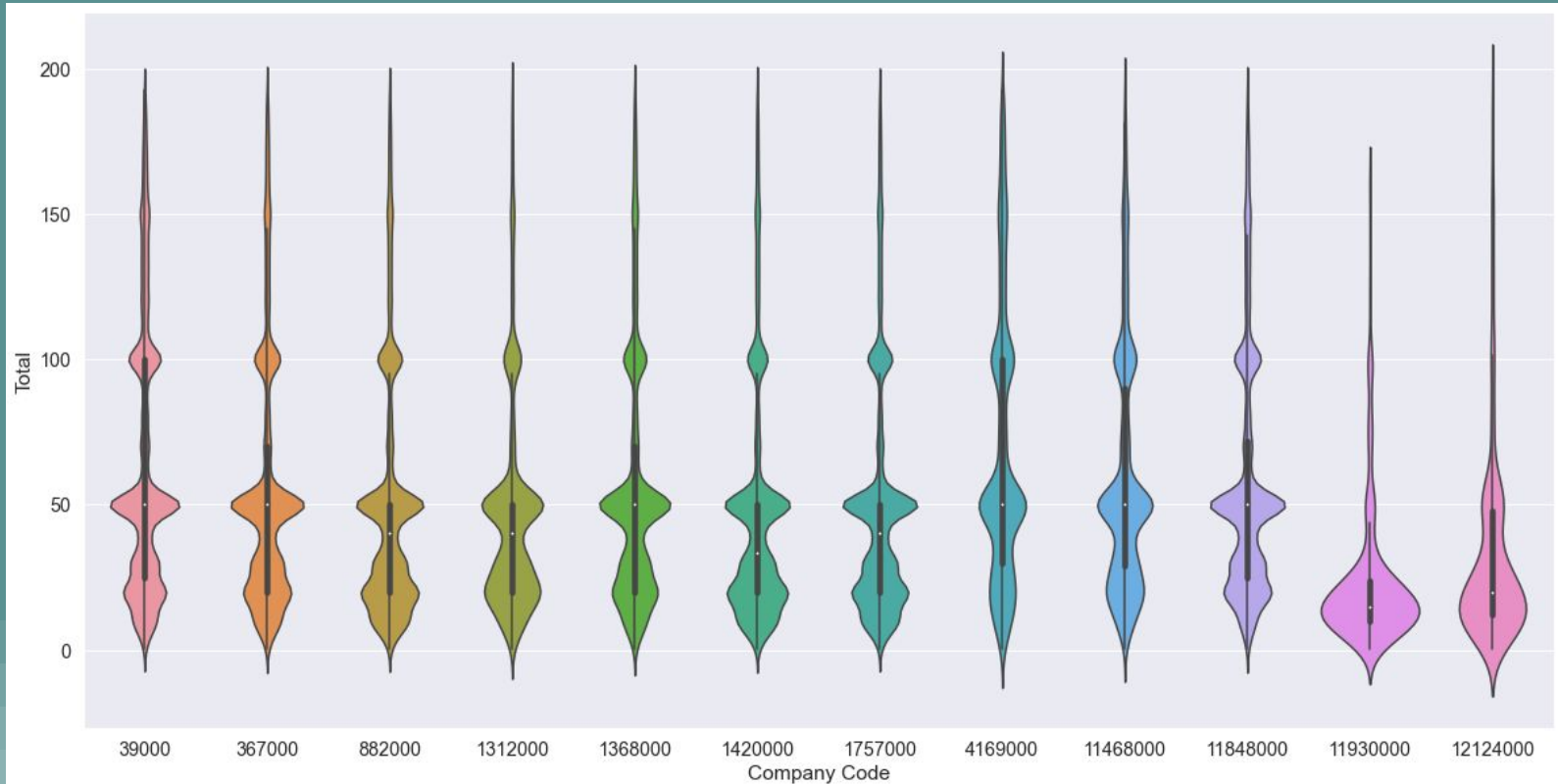




# The 5 number theory with the company code with the total cost of products (box plot)



# The 5 number theory with the company code with the total cost of products with distribution (violin plot)



# Recommendation:

Recommending companies with company codes: 1312000, 4169000, 11930000, 12124000 to increase their sellings and look into products and average cost.