Week1_task_eda_SatwikSaurav

October 11, 2023

Notebook for Week 1 task for Cognizant. The task is to perform Exploatory Data Analysis on the dataset provided and gain insights into the data and summarise the findings in a concise and business-friendly manner within an email to the Data Science team leader.

```
[1]: import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import numpy as np
     import os
```

```
[2]: # Visualization Prefrences.
     %matplotlib inline
     sns.set_style("whitegrid")
     plt.style.use("fivethirtyeight")
```

DATA LOADING

```
[3]: data=pd.read_csv("4.1 sample_sales_data.csv")
```

DATA DESCRIPTION

```
[4]: data.head(8).T
```

```
[4]:                                                     0  \
     Unnamed: 0                                          0
     transaction_id  a1c82654-c52c-45b3-8ce8-4c2a1efe63ed
     timestamp                        2022-03-02 09:51:38
     product_id      3bc6c1ea-0198-46de-9ffd-514ae3338713
     category                                       fruit
     customer_type                                   gold
     unit_price                                      3.99
     quantity                                           2
     total                                           7.98
     payment_type                                e-wallet

                                                        1  \
     Unnamed: 0                                          1
     transaction_id  931ad550-09e8-4da6-beaa-8c9d17be9c60
     timestamp                        2022-03-06 10:33:59
```

```
product_id      ad81b46c-bf38-41cf-9b54-5fe7f5eba93e
category                                       fruit
customer_type                               standard
unit_price                                      3.99
quantity                                           1
total                                           3.99
payment_type                                e-wallet

                                                   2  \
Unnamed: 0                                         2
transaction_id  ae133534-6f61-4cd6-b6b8-d1c1d8d90aea
timestamp                        2022-03-04 17:20:21
product_id      7c55cbd4-f306-4c04-a030-628cbe7867c1
category                                       fruit
customer_type                                premium
unit_price                                      0.19
quantity                                           2
total                                           0.38
payment_type                                e-wallet

                                                   3  \
Unnamed: 0                                         3
transaction_id  157cebd9-aaf0-475d-8a11-7c8e0f5b76e4
timestamp                        2022-03-02 17:23:58
product_id      80da8348-1707-403f-8be7-9e6deeccc883
category                                       fruit
customer_type                                   gold
unit_price                                      0.19
quantity                                           4
total                                           0.76
payment_type                                e-wallet

                                                   4  \
Unnamed: 0                                         4
transaction_id  a81a6cd3-5e0c-44a2-826c-aea43e46c514
timestamp                        2022-03-05 14:32:43
product_id      7f5e86e6-f06f-45f6-bf44-27b095c9ad1d
category                                       fruit
customer_type                                  basic
unit_price                                      4.49
quantity                                           2
total                                           8.98
payment_type                              debit card

                                                   5  \
Unnamed: 0                                         5
transaction_id  b5b3c8b9-f496-484d-aa30-4f2efb5ed56c
```

```
timestamp                        2022-03-07 17:59:47
product_id      3bc6c1ea-0198-46de-9ffd-514ae3338713
category                                       fruit
customer_type                               standard
unit_price                                      3.99
quantity                                           4
total                                          15.96
payment_type                                    cash

                                                   6  \
Unnamed: 0                                         6
transaction_id  4997b1ae-f5aa-4b9f-8fc8-22ad8f19837c
timestamp                        2022-03-07 19:36:57
product_id      14736243-d346-438f-9535-d80fcb9f3882
category                                       fruit
customer_type                               standard
unit_price                                      1.49
quantity                                           4
total                                           5.96
payment_type                                e-wallet

                                                   7
Unnamed: 0                                         7
transaction_id  bfffee68-0736-42af-bd3e-4ca77541b0d6
timestamp                        2022-03-07 19:03:20
product_id      0ddc2379-adba-4fb0-aa97-19fcafc738a1
category                                       fruit
customer_type                                  basic
unit_price                                      3.99
quantity                                           4
total                                          15.96
payment_type                             credit card
```

[5]:
```
data.head()
```

[5]:
```
   Unnamed: 0                        transaction_id            timestamp  \
0           0  a1c82654-c52c-45b3-8ce8-4c2a1efe63ed  2022-03-02 09:51:38
1           1  931ad550-09e8-4da6-beaa-8c9d17be9c60  2022-03-06 10:33:59
2           2  ae133534-6f61-4cd6-b6b8-d1c1d8d90aea  2022-03-04 17:20:21
3           3  157cebd9-aaf0-475d-8a11-7c8e0f5b76e4  2022-03-02 17:23:58
4           4  a81a6cd3-5e0c-44a2-826c-aea43e46c514  2022-03-05 14:32:43

                             product_id category customer_type  unit_price  \
0  3bc6c1ea-0198-46de-9ffd-514ae3338713    fruit          gold        3.99
1  ad81b46c-bf38-41cf-9b54-5fe7f5eba93e    fruit      standard        3.99
2  7c55cbd4-f306-4c04-a030-628cbe7867c1    fruit       premium        0.19
3  80da8348-1707-403f-8be7-9e6deeccc883    fruit          gold        0.19
```

```
4  7f5e86e6-f06f-45f6-bf44-27b095c9ad1d    fruit          basic         4.49

   quantity  total payment_type
0         2   7.98     e-wallet
1         1   3.99     e-wallet
2         2   0.38     e-wallet
3         4   0.76     e-wallet
4         2   8.98   debit card
```

[6]:
```
data.shape
```

[6]:
```
(7829, 10)
```

[7]:
```
# Extract Descriptive Data.
pd.set_option("display.float", "{:.2f}".format)
data.describe().T
```

[7]:
```
             count    mean     std  min      25%     50%     75%     max
Unnamed: 0 7829.00 3914.00 2260.18 0.00 1957.00 3914.00 5871.00 7828.00
unit_price 7829.00    7.82    5.39 0.19    3.99    7.19   11.19   23.99
quantity   7829.00    2.50    1.12 1.00    1.00    3.00    4.00    4.00
total      7829.00   19.71   17.45 0.19    6.57   14.97   28.47   95.96
```

[8]:
```
data.describe()
```

[8]:
```
       Unnamed: 0  unit_price  quantity   total
count     7829.00     7829.00   7829.00 7829.00
mean      3914.00        7.82      2.50   19.71
std       2260.18        5.39      1.12   17.45
min          0.00        0.19      1.00    0.19
25%       1957.00        3.99      1.00    6.57
50%       3914.00        7.19      3.00   14.97
75%       5871.00       11.19      4.00   28.47
max       7828.00       23.99      4.00   95.96
```

[9]:
```
data.drop(columns=["Unnamed: 0"], inplace=True, errors='ignore')
data.tail(8)
```

[9]:
```
                           transaction_id            timestamp  \
7821  a8109d22-e192-41d4-911d-84c772a68013  2022-03-02 10:42:44
7822  6857feab-f2b1-4de7-bd4e-14a838591411  2022-03-04 11:06:33
7823  60524862-cd12-47e8-aaa6-9a15e3f2c74d  2022-03-07 12:44:43
7824  6c19b9fc-f86d-4526-9dfe-d8027a4d13ee  2022-03-03 18:22:09
7825  1c69824b-e399-4b79-a5e7-04a3a7db0681  2022-03-04 19:14:46
7826  79aee7d6-1405-4345-9a15-92541e9e1e74  2022-03-03 14:00:09
7827  e5cc4f88-e5b7-4ad5-bc1b-12a828a14f55  2022-03-04 15:11:38
7828  afd70b4f-ee21-402d-8d8f-0d9e13c2bea6  2022-03-06 13:50:36
```

```
                                product_id           category customer_type  \
7821  6c8d0a2a-576a-432f-a090-c123dee91aaa  cleaning products          gold
7822  364035ab-945a-4c34-9734-5167b787ae5c  cleaning products      standard
7823  bc6187a9-d508-482b-9ca6-590d1cc7524f  cleaning products         basic
7824  bc6187a9-d508-482b-9ca6-590d1cc7524f  cleaning products         basic
7825  707e4237-191c-4cc9-85af-383a6c1cb2ab  cleaning products      standard
7826  a9325c1a-2715-41df-b7f4-3078fa5ecd97  cleaning products         basic
7827  707e4237-191c-4cc9-85af-383a6c1cb2ab  cleaning products         basic
7828  d6ccd088-11be-4c25-aa1f-ea87c01a04db  cleaning products    non-member

      unit_price  quantity  total payment_type
7821       15.49         4  61.96  credit card
7822        8.99         3  26.97   debit card
7823       14.19         2  28.38  credit card
7824       14.19         2  28.38     e-wallet
7825       16.99         1  16.99  credit card
7826       14.19         2  28.38  credit card
7827       16.99         4  67.96         cash
7828       14.99         4  59.96   debit card
```

1 EXPLORATORY DATA ANALYSIS

[10]:

```
# Check for Null Values
data.isna().sum()
```

[10]:

```
transaction_id    0
timestamp         0
product_id        0
category          0
customer_type     0
unit_price        0
quantity          0
total             0
payment_type      0
dtype: int64
```

no null values

[11]:

```
data.dtypes
```

[11]:

```
transaction_id     object
timestamp          object
product_id         object
category           object
customer_type      object
unit_price        float64
quantity            int64
```

```
total              float64
payment_type        object
dtype: object
```

- transaction_id = this is a unique ID that is assigned to each transaction
- timestamp = this is the datetime at which the transaction was made
- product_id = this is an ID that is assigned to the product that was sold. Each product has a unique ID
- category = this is the category that the product is contained within
- customer_type = this is the type of customer that made the transaction
- unit_price = the price that 1 unit of this item sells for
- quantity = the number of units sold for this product within this transaction
- total = the total amount payable by the customer
- payment_type = the payment method used by the customer

[12]:
```
data["total"].median()
```

[12]:
```
14.97
```

[13]:
```
data["total"].mean()
```

[13]:
```
19.70990547962791
```

[14]:
```
data["total"].count()
```

[14]:
```
7829
```

[15]:
```
data["unit_price"].median()
```

[15]:
```
7.19
```

[16]:
```
data["unit_price"].mean()
```

[16]:
```
7.819480137948519
```

[17]:
```
data["quantity"].mean()
```

[17]:
```
2.501596627921829
```

[18]:
```
data["quantity"].median()
```

[18]:
```
3.0
```

[19]:
```
def plot_continuous_distribution(data: pd.DataFrame = None, column: str = None,␣
 ↪height: int = 8):
  _ = sns.displot(data, x=column, kde=True, height=height, aspect=height/5).
 ↪set(title=f'Distribution of {column}');

def get_unique_values(data, column):
  num_unique_values = len(data[column].unique())
  value_counts = data[column].value_counts()
  print(f"Column: {column} has {num_unique_values} unique values\n")
  print(value_counts)

def plot_categorical_distribution(data: pd.DataFrame = None, column: str =␣
 ↪None, height: int = 8, aspect: int = 2):
  _ = sns.catplot(data=data, x=column, kind='count', height=height,␣
 ↪aspect=aspect).set(title=f'Distribution of {column}');
```

[20]:

```
get_unique_values(data, 'category')
```

```
Column: category has 22 unique values

fruit                    998
vegetables               846
packaged foods           507
baked goods              443
canned foods             431
refrigerated items       425
kitchen                  382
meat                     382
dairy                    375
beverages                301
cheese                   293
cleaning products        292
baking                   264
snacks                   263
frozen                   263
seafood                  253
medicine                 243
baby products            224
condiments and sauces    181
personal care            177
pets                     161
spices and herbs         125
Name: category, dtype: int64
```

[21]:

```
get_unique_values(data, 'customer_type')
```

```
Column: customer_type has 5 unique values

non-member    1601
standard      1595
premium       1590
basic         1526
gold          1517
Name: customer_type, dtype: int64
```

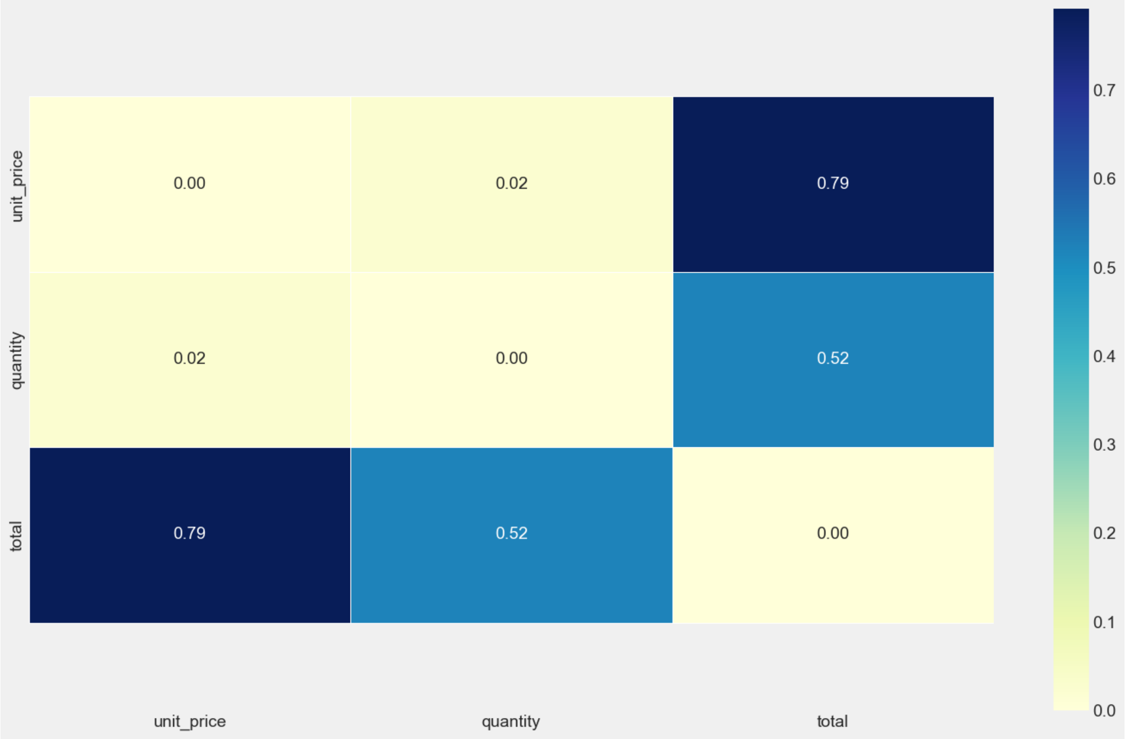
2 EXPLORATORY DATA ANALYSIS USING VISUALISATION

[22]:

```
corr_matrix = data.corr()
for x in range(corr_matrix.shape[0]):
    corr_matrix.iloc[x,x] = 0.0

fig, ax = plt.subplots(figsize=(16, 10))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```

[22]:

```
(3.5, -0.5)
```



[23]:

```
# The correlation matrix
corr_mat = data.corr()

# Strip out the diagonal values for the next step
for x in range(corr_mat.shape[0]):
    corr_mat.iloc[x,x] = 0.0

corr_mat
```

[23]:

```
             unit_price  quantity  total
unit_price         0.00      0.02   0.79
quantity           0.02      0.00   0.52
total              0.79      0.52   0.00
```

[24]:

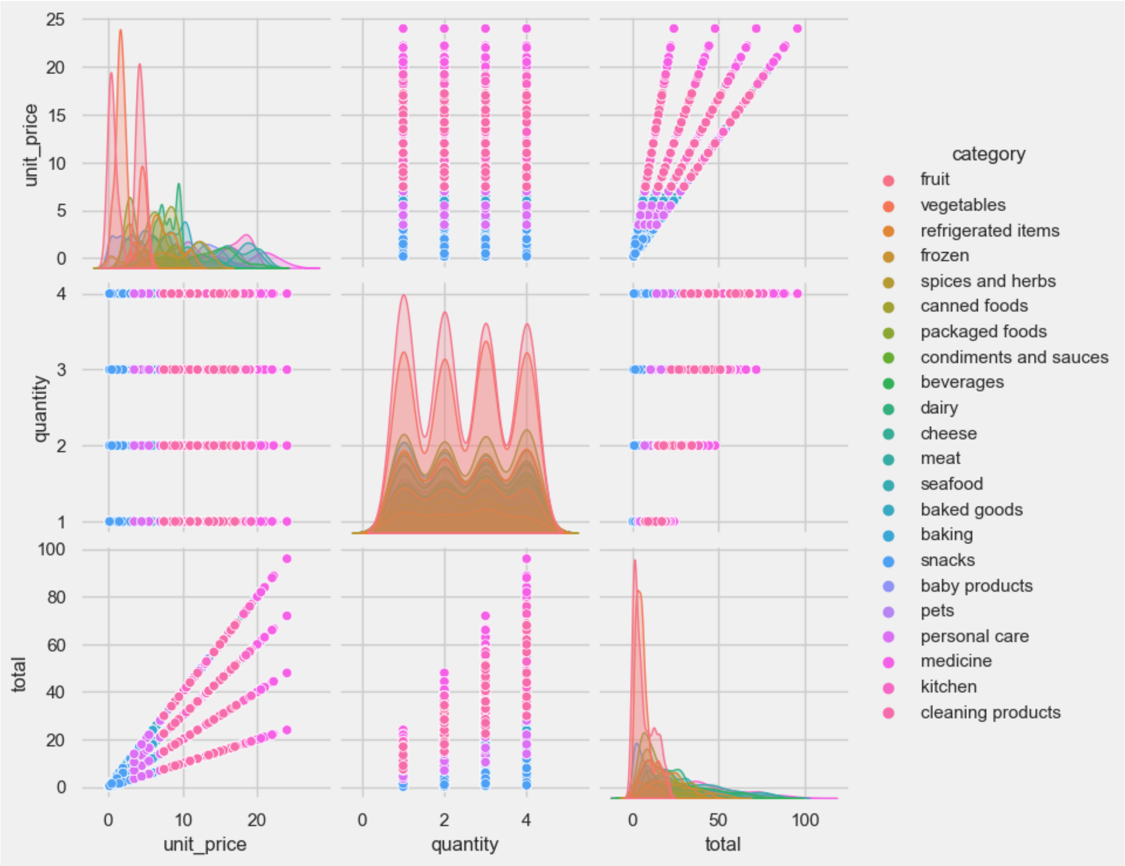
```
corr_max  = corr_mat.abs().max().to_frame()
corr_id_max = corr_mat.abs().idxmax().to_frame()

# dataframe aggrigation and processing
pair_features_corr = pd.merge(corr_id_max, corr_max, on = corr_max.index)
pair_features_corr = pair_features_corr.rename(columns = {'key_0':
 ↪'Feature_one', '0_x':'Feature_two', '0_y':'correlation'})\
                                                .sort_values('correlation',␣
 ↪ascending=False)\
                                                .reset_index().drop('index',␣
 ↪axis=1)
pair_features_corr
```

[24]:

```
  Feature_one Feature_two  correlation
0  unit_price       total         0.79
1       total  unit_price         0.79
2    quantity       total         0.52
```

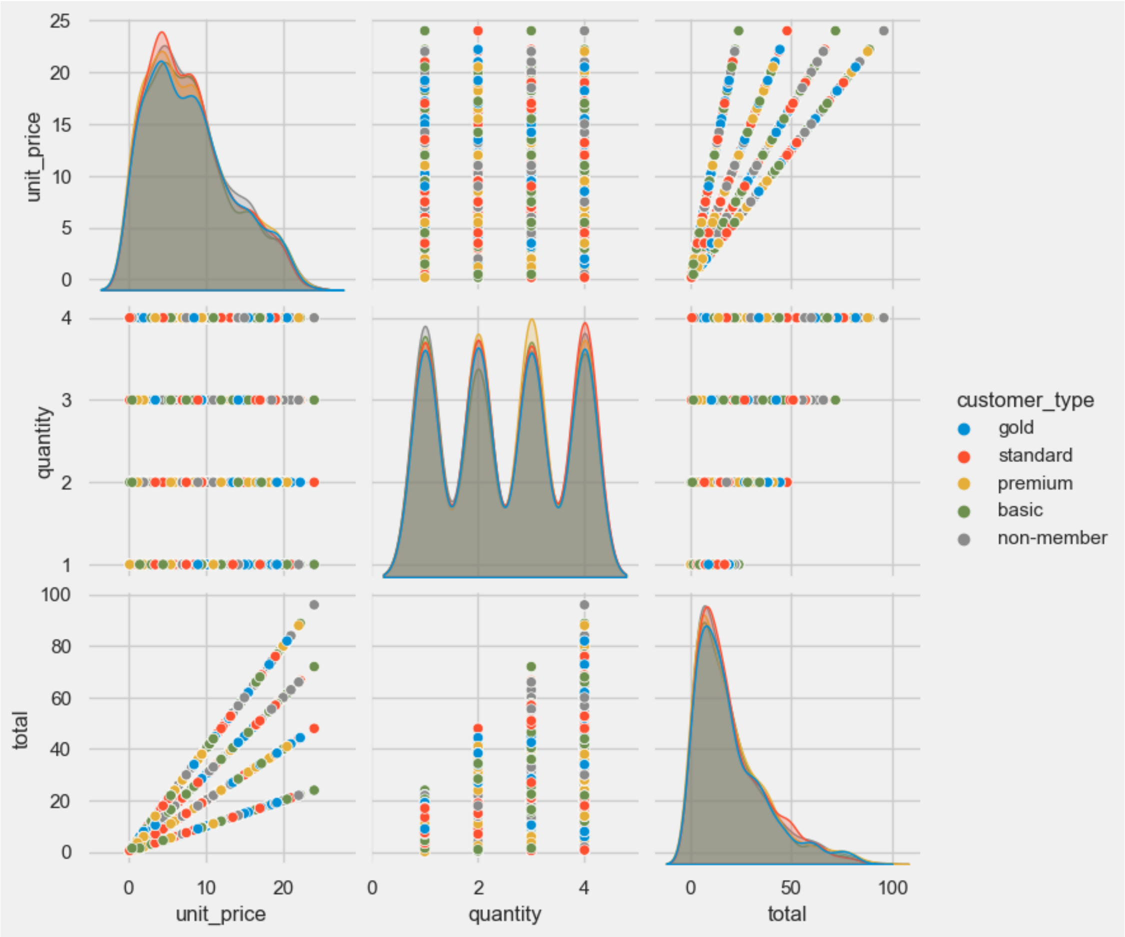
[25]:

```
float_columns = [col for col in data.columns if col != 'category']

sns.set_context('notebook')
sns.pairplot(data[float_columns + ['category']],
             hue='category'
             );
```



[26]:

```
float_columns = [col for col in data.columns if col != 'customer_type']

sns.set_context('notebook')
sns.pairplot(data[float_columns + ['customer_type']],
             hue='customer_type'
             );
```



[27]:

```
skew_trans_columns = (data
                    .skew()
                    .sort_values(ascending=False)).to_frame("skewness_value")
skew_trans_columns
```

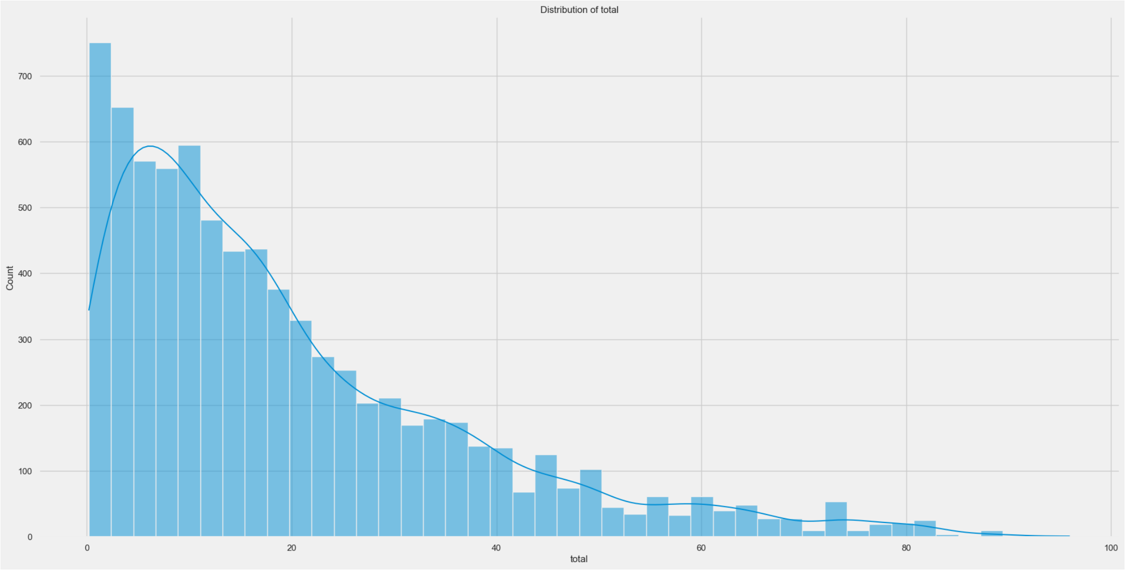
```
C:\Users\simmy\AppData\Local\Temp\ipykernel_22284\560499137.py:1: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
  skew_trans_columns = (data
```

[27]:

```
               skewness_value
total                    1.35
unit_price               0.65
quantity                -0.00
```

[28]:

```
plot_continuous_distribution(data, 'total', 10)
```



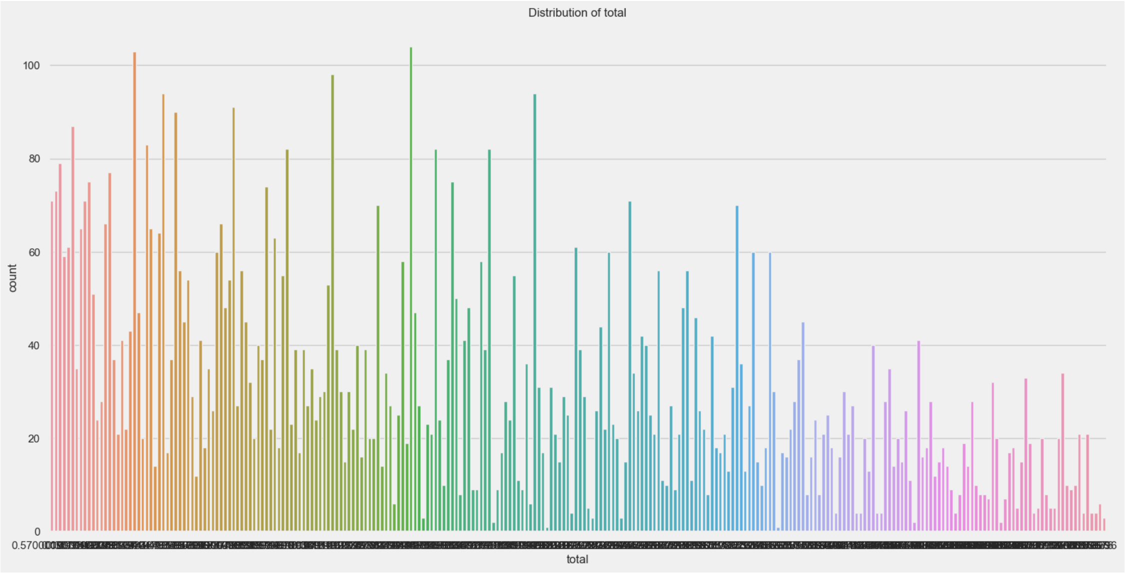
[29]:

```
get_unique_values(data,'total')
```

```
Column: total has 256 unique values

14.97    104
3.99     103
11.97     98
4.99      94
19.96     94
        …
60.57      2
47.98      2
17.99      2
20.19      1
35.98      1
Name: total, Length: 256, dtype: int64
```

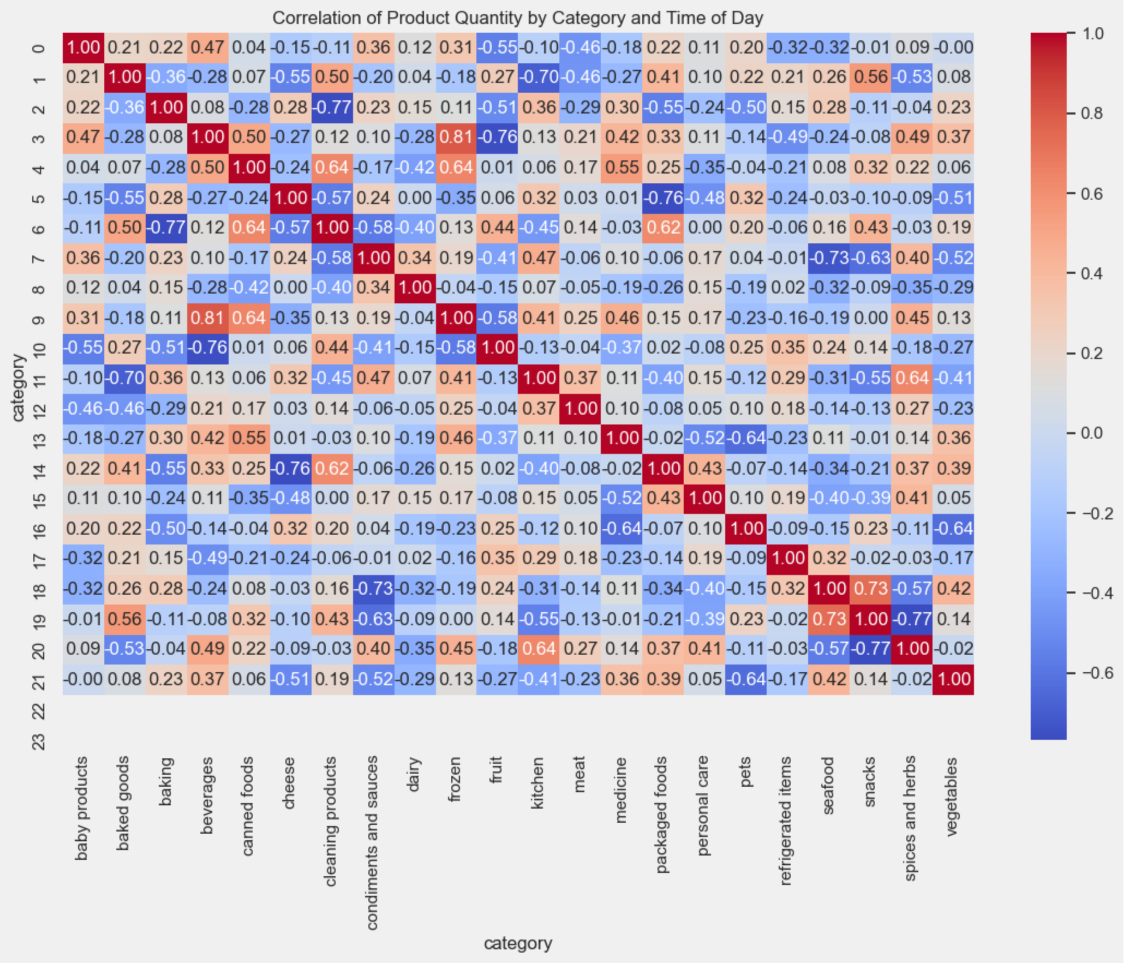
[30]:

```
plot_categorical_distribution(data, 'total')
```



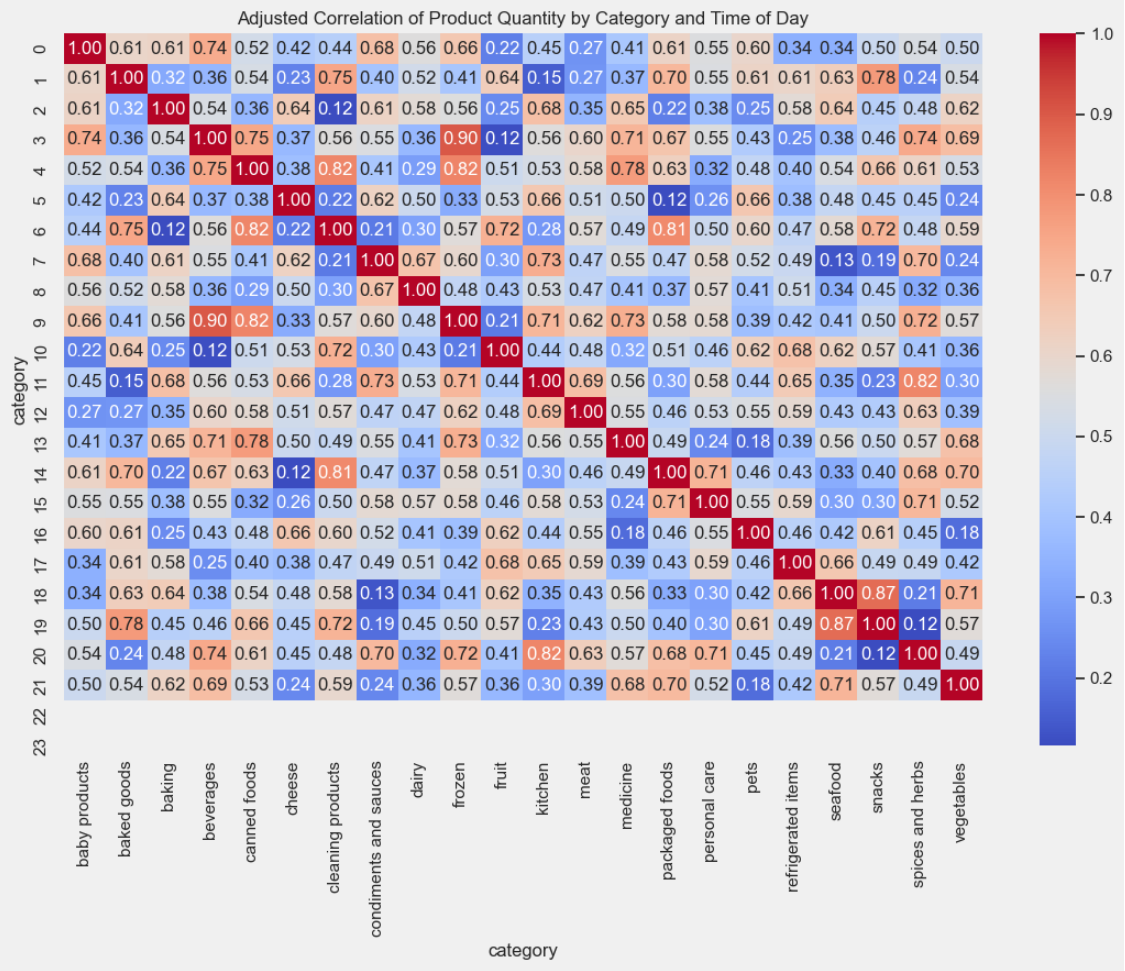
[31]:

```python
data['timestamp'] = pd.to_datetime(data['timestamp'])
# Extract the hour of the day
data['hour'] = data['timestamp'].dt.hour

# Group the data by 'hour' and 'category' and calculate the sum of 'quantity' sold
grouped_data = data.groupby(['hour', 'category'])['quantity'].sum().reset_index()

# pivot table
pivot_data = grouped_data.pivot('hour', 'category', 'quantity')

# Set the order of hours and labels for the x-axis
hour_order = range(24)  # Assuming 24 hours
hour_labels = [str(hour) for hour in hour_order]

# heatmap with x-axis labels showing the time of the day
plt.figure(figsize=(12, 8))
sns.heatmap(pivot_data.corr(), annot=True, cmap='coolwarm', fmt='.2f', xticklabels=pivot_data.columns, yticklabels=hour_labels)
plt.title('Correlation of Product Quantity by Category and Time of Day')
plt.show()
```



[32]:

```python
pivot_data = grouped_data.pivot('hour', 'category', 'quantity')

# Set the order of hours and labels for the x-axis
hour_order = range(24)
hour_labels = [str(hour) for hour in hour_order]

correlation_matrix = pivot_data.corr()

# Ensure all values are positive and adjusted
adjusted_correlation_matrix = (correlation_matrix + 1) / 2
plt.figure(figsize=(12, 8))
sns.heatmap(adjusted_correlation_matrix, annot=True, cmap='coolwarm', fmt='.
 ↪2f', xticklabels=pivot_data.columns, yticklabels=hour_labels)
plt.title('Adjusted Correlation of Product Quantity by Category and Time of␣
 ↪Day')
plt.show()
```



	baby products	baked goods	baking	beverages	canned foods	cheese	cleaning products	condiments and sauces	dairy	frozen	fruit	kitchen	meat	medicine	packaged foods	personal care	pets	refrigerated items	seafood	snacks	spices and herbs	vegetables
0	1.00	0.61	0.61	0.74	0.52	0.42	0.44	0.68	0.56	0.66	0.22	0.45	0.27	0.41	0.61	0.55	0.60	0.34	0.34	0.50	0.54	0.50
1	0.61	1.00	0.32	0.36	0.54	0.23	0.75	0.40	0.52	0.41	0.64	0.15	0.27	0.37	0.70	0.55	0.61	0.61	0.63	0.78	0.24	0.54
2	0.61	0.32	1.00	0.54	0.36	0.64	0.12	0.61	0.58	0.56	0.25	0.68	0.35	0.65	0.22	0.38	0.25	0.58	0.64	0.45	0.48	0.62
3	0.74	0.36	0.54	1.00	0.75	0.37	0.56	0.55	0.36	0.90	0.12	0.56	0.60	0.71	0.67	0.55	0.43	0.25	0.38	0.46	0.74	0.69
4	0.52	0.54	0.36	0.75	1.00	0.38	0.82	0.41	0.29	0.82	0.51	0.53	0.58	0.78	0.63	0.32	0.48	0.40	0.54	0.66	0.61	0.53
5	0.42	0.23	0.64	0.37	0.38	1.00	0.22	0.62	0.50	0.33	0.53	0.66	0.51	0.50	0.12	0.26	0.66	0.38	0.48	0.45	0.45	0.24
6	0.44	0.75	0.12	0.56	0.82	0.22	1.00	0.21	0.30	0.57	0.72	0.28	0.57	0.49	0.81	0.50	0.60	0.47	0.58	0.72	0.48	0.59
7	0.68	0.40	0.61	0.55	0.41	0.62	0.21	1.00	0.67	0.60	0.30	0.73	0.47	0.55	0.47	0.58	0.52	0.49	0.13	0.19	0.70	0.24
8	0.56	0.52	0.58	0.36	0.29	0.50	0.30	0.67	1.00	0.48	0.43	0.53	0.47	0.41	0.37	0.57	0.41	0.51	0.34	0.45	0.32	0.36
9	0.66	0.41	0.56	0.90	0.82	0.33	0.57	0.60	0.48	1.00	0.21	0.71	0.62	0.73	0.58	0.58	0.39	0.42	0.41	0.50	0.72	0.57
10	0.22	0.64	0.25	0.12	0.51	0.53	0.72	0.30	0.43	0.21	1.00	0.44	0.48	0.32	0.51	0.46	0.62	0.68	0.62	0.57	0.41	0.36
11	0.45	0.15	0.68	0.56	0.53	0.66	0.28	0.73	0.53	0.71	0.44	1.00	0.69	0.56	0.30	0.58	0.44	0.65	0.35	0.23	0.82	0.30
12	0.27	0.27	0.35	0.60	0.58	0.51	0.57	0.47	0.47	0.62	0.48	0.69	1.00	0.55	0.46	0.53	0.55	0.59	0.43	0.43	0.63	0.39
13	0.41	0.37	0.65	0.71	0.78	0.50	0.49	0.55	0.41	0.73	0.32	0.56	0.55	1.00	0.49	0.24	0.18	0.39	0.56	0.50	0.57	0.68
14	0.61	0.70	0.22	0.67	0.63	0.12	0.81	0.47	0.37	0.58	0.51	0.30	0.46	0.49	1.00	0.71	0.46	0.43	0.33	0.40	0.68	0.70
15	0.55	0.55	0.38	0.55	0.32	0.26	0.50	0.58	0.57	0.58	0.46	0.58	0.53	0.24	0.71	1.00	0.55	0.59	0.30	0.30	0.71	0.52
16	0.60	0.61	0.25	0.43	0.48	0.66	0.60	0.52	0.41	0.39	0.62	0.44	0.55	0.18	0.46	0.55	1.00	0.46	0.42	0.61	0.45	0.18
17	0.34	0.61	0.58	0.25	0.40	0.38	0.47	0.49	0.51	0.42	0.68	0.65	0.59	0.39	0.43	0.59	0.46	1.00	0.66	0.49	0.49	0.42
18	0.34	0.63	0.64	0.38	0.54	0.48	0.58	0.13	0.34	0.41	0.62	0.35	0.43	0.56	0.33	0.30	0.42	0.66	1.00	0.87	0.21	0.71
19	0.50	0.78	0.45	0.46	0.66	0.45	0.72	0.19	0.45	0.50	0.57	0.23	0.43	0.50	0.40	0.30	0.61	0.49	0.87	1.00	0.12	0.57
20	0.54	0.24	0.48	0.74	0.61	0.45	0.48	0.70	0.32	0.72	0.41	0.82	0.63	0.57	0.68	0.71	0.45	0.49	0.21	0.12	1.00	0.49
21	0.50	0.54	0.62	0.69	0.53	0.24	0.59	0.24	0.36	0.57	0.36	0.30	0.39	0.68	0.70	0.52	0.18	0.42	0.71	0.57	0.49	1.00
22																						
23																						

Using means, medians, correlation matrix, heatmaps, pairplots feature analysis to analyse the data, we have no distinct conclusion. More data is required for any usable conclusions.

Even with analysis of product category, quantity and time of the data is not sufficient data to derive a definite conclusion. However, we have some results for what type of product has more demand at what time fo the day.

3 SUMMARY

From this dataset, it is impossible to answer that question. In order to make the next step on this project with the client, it is clear that:

- We need more rows of data. The current sample is only from 1 store and 1 week worth of data
- We need to frame the specific problem statement that we want to solve. The current business problem is too broad, we should narrow down the focus in order to deliver a valuable end product
- We need more features. Based on the problem statement that we move forward with, we need

more columns (features) that may help us to understand the outcome that we're solving for