EDA Commerce-e marketplace in *Pakistan*

about the data :

> Geographic Location: Pakistan
>
> Time Period: March 2016 - August 2018
>
> Data Source: Commerce-e marketplace in Pakistan
>
> Data Set: Detailed information on half a million online trade orders in Pakistan,including item details, shipping method, payment methods, product categories,order date, SKU (Product ID), price, quantity, total amount, and customer ID.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objs as go
import statsmodels.api as sm
from scipy import stats

data = pd.read_csv('/content/drive/MyDrive/Pakistan Largest Ecommerce Dataset.csv')
data.head()

<ipython-input-2-3602f242e9ed>:1: DtypeWarning: Columns (1,2,3,7,8,9,11,12,13,14,17,18,19) have mixed types. Specify dtype option on import or set low_memory=False.
  data = pd.read_csv('/content/drive/MyDrive/Pakistan Largest Ecommerce Dataset.csv')
```

{"type":"dataframe","variable_name":"data"}

check the size of data

```
print(data.shape)

(1048575, 26)
```

we found empty columns (Unnamed: 21 - Unnamed: 25) then we will delete the empty columns

```
data = data.dropna(axis=1, how='all')
data.head()
```

{"type":"dataframe","variable_name":"data"}

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 21 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   item_id                584524 non-null  float64
 1   status                 584509 non-null  object 
 2   created_at             584524 non-null  object 
 3   sku                    584504 non-null  object 
 4   price                  584524 non-null  float64
 5   qty_ordered            584524 non-null  float64
 6   grand_total            584524 non-null  float64
 7   increment_id           584524 non-null  object 
 8   category_name_1        584360 non-null  object 
 9   sales_commission_code  447346 non-null  object 
 10  discount_amount        584524 non-null  float64
 11  payment_method         584524 non-null  object 
 12  Working Date           584524 non-null  object 
 13  BI Status              584524 non-null  object 
 14   MV                    584524 non-null  object 
 15  Year                   584524 non-null  float64
 16  Month                  584524 non-null  float64
 17  Customer Since         584513 non-null  object 
 18  M-Y                    584524 non-null  object 
 19  FY                     584524 non-null  object 
 20  Customer ID            584513 non-null  float64
dtypes: float64(8), object(13)
memory usage: 168.0+ MB
```

Details about the data :

```
print(f"There are {data.shape[0]} instances.")
print(f"There are {data.shape[1]} dataframe columns/attributes.")

num_attribs = data.select_dtypes(include = ['float64', 'int64'])
cat_attribs = [data.columns[i] for i in range(len(data.columns)) if
data.columns[i] not in num_attribs]

print(f"\nThere are {len(cat_attribs)} categorical attributes: ")
for i in range(len(cat_attribs)):
    print(f"{i+1}. {cat_attribs[i]}")
```

```
There are 1048575 instances.
There are 21 dataframe columns/attributes.

There are 13 categorical attributes: 
1. status
2. created_at
3. sku
```

```
4. increment_id
5. category_name_1
6. sales_commission_code
7. payment_method
8. Working Date
9. BI Status
10.  MV
11. Customer Since
12. M-Y
13. FY
```

check how many null values we have

```
data.isna().sum()
```

```
item_id                    464051
status                     464066
created_at                 464051
sku                        464071
price                      464051
qty_ordered                464051
grand_total                464051
increment_id               464051
category_name_1            464215
sales_commission_code      601229
discount_amount            464051
payment_method             464051
Working Date               464051
BI Status                  464051
 MV                        464051
Year                       464051
Month                      464051
Customer Since             464062
M-Y                        464051
FY                         464051
Customer ID                464062
dtype: int64
```

take a look to the tail

```
data.tail()
```

{"type":"dataframe"}

delete null rows

```
data.dropna(how='all', inplace = True)
```

check if it is done

```
data.tail()
```

{"type":"dataframe"}

null value

```
data.isna().sum()
```

```
item_id                        0
status                        15
created_at                     0
sku                           20
price                          0
qty_ordered                    0
grand_total                    0
increment_id                   0
category_name_1              164
sales_commission_code     137178
discount_amount                0
payment_method                 0
Working Date                   0
BI Status                      0
 MV                            0
Year                           0
Month                          0
Customer Since                11
M-Y                            0
FY                             0
Customer ID                   11
dtype: int64
```

find how many duplicated values

```
data.duplicated().sum()
```

```
0
```

number of items in the data

```
print("we have",data['item_id'].nunique(),"items in the data")
```

```
we have 584524 items in the data
```

orders status for all items

```
statusfilt=data.groupby('status')
['item_id'].nunique().sort_values(ascending=False)
statusfilt
```

```
status
complete          233685
canceled          201249
received           77290
order_refunded     59529
refund              8050
cod                 2859
paid                1159
closed               494
payment_review        57
pending               48
processing            33
holded                31
fraud                 10
pending_paypal         7
\N                     4
exchange               4
Name: item_id, dtype: int64
```

the status for every payment method

```
status_payment_pivot = pd.pivot_table(data,
                                      index='status',
                                      columns='payment_method',
                                      values='item_id',
                                      aggfunc=pd.Series.nunique)
status_payment_pivot
```


```
{"summary":"{\n  \"name\": \"status_payment_pivot\",\n  \"rows\": 16,\
n  \"fields\": [\n    {\n      \"column\": \"status\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 16,\n        \"samples\": [\n          \"\\\\
N\",\n          \"canceled\",\n          \"exchange\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Easypay\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
18088.39449560329,\n        \"min\": 2.0,\n        \"max\": 52040.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          16.0,\n
241.0,\n          52040.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Easypay_MA\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3558.5542569981985,\n
\"min\": 35.0,\n        \"max\": 9210.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          9210.0,\n
3116.0,\n          144.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Payaxis\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 19468.99057732578,\n        \"min\":
4.0,\n        \"max\": 61267.0,\n        \"num_unique_values\": 10,\n
```

```
\"samples\": [\n          8709.0,\n          71.0,\n          3813.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"apg\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 535.3791180089115,\n
\"min\": 3.0,\n        \"max\": 1361.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          1361.0,\n
27.0,\n          3.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"bankalfalah\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 5124.717470803201,\n        \"min\":
5.0,\n        \"max\": 16184.0,\n        \"num_unique_values\": 10,\n
\"samples\": [\n          5166.0,\n          5.0,\n          183.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"cashatdoorstep\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
295.1208227150365,\n        \"min\": 5.0,\n        \"max\": 674.0,\n
\"num_unique_values\": 4,\n        \"samples\": [\n          674.0,\n
18.0,\n          5.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"cod\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 43305.382871024936,\n        \"min\": 4.0,\n        \"max\":
148039.0,\n        \"num_unique_values\": 11,\n        \"samples\": [\
n          6.0,\n          4.0,\n          44555.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"customercredit\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1453.8685279035967,\n        \"min\": 3.0,\n        \"max\": 4151.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n          24.0,\n
3.0,\n          47.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"easypay_voucher\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 6437.554016416928,\n        \"min\":
2.0,\n        \"max\": 16066.0,\n        \"num_unique_values\": 8,\n
\"samples\": [\n          2.0,\n          33.0,\n          12189.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"financesettlement\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3.774917217635375,\n        \"min\": 1.0,\n        \"max\": 9.0,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          1.0,\n
9.0,\n          4.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"internetbanking\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 124.44597221284424,\n        \"min\":
4.0,\n        \"max\": 286.0,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          4.0,\n          10.0,\n          156.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"jazzvoucher\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3087.226561055362,\n        \"min\": 2.0,\n        \"max\": 8472.0,\n
```

\"num_unique_values\": 8,\n   \"samples\": [\n   2.0,\n   12.0,\n   8472.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"jazzwallet\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 6836.794266268784,\n   \"min\": 1.0,\n   \"max\": 16933.0,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   3.0,\n   1.0,\n   16933.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"marketingexpense\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 23.388031127053,\n   \"min\": 1.0,\n   \"max\": 42.0,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   42.0,\n   1.0,\n   2.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"mcblite\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 153.79434319896166,\n   \"min\": 1.0,\n   \"max\": 393.0,\n   \"num_unique_values\": 6,\n   \"samples\": [\n   179.0,\n   1.0,\n   11.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"mygateway\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 371.56560658919983,\n   \"min\": 3.0,\n   \"max\": 652.0,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   652.0,\n   14.0,\n   3.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"productcredit\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 32.85574531189332,\n   \"min\": 2.0,\n   \"max\": 83.0,\n   \"num_unique_values\": 4,\n   \"samples\": [\n   2.0,\n   15.0,\n   10.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"ublcreditcard\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 244.3187808308372,\n   \"min\": 1.0,\n   \"max\": 660.0,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   660.0,\n   4.0,\n   6.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"status_payment_pivot"}

Number of Customer

Here we looked at the number of customers in each month of the year

```
pd.crosstab(index = data['Customer Since'],
columns='count',values=data[~(data['Customer ID'].duplicated())]
['Customer ID'],aggfunc='count').sort_values(by = 'count')
```

{"summary":"{\n \"name\": \"pd\",\n \"rows\": 26,\n \"fields\": [\n   {\n   \"column\": \"Customer Since\",\n   \"properties\": {\n

\"dtype\": \"object\",\n   \"num_unique_values\": 26,\n
\"samples\": [\n   \"2018-4\",\n   \"2017-7\",\n
\"2018-8\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 3894,\n   \"min\": 1530,\n   \"max\": 16719,\n
\"num_unique_values\": 26,\n   \"samples\": [\n   2486,\n
3395,\n   1530\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

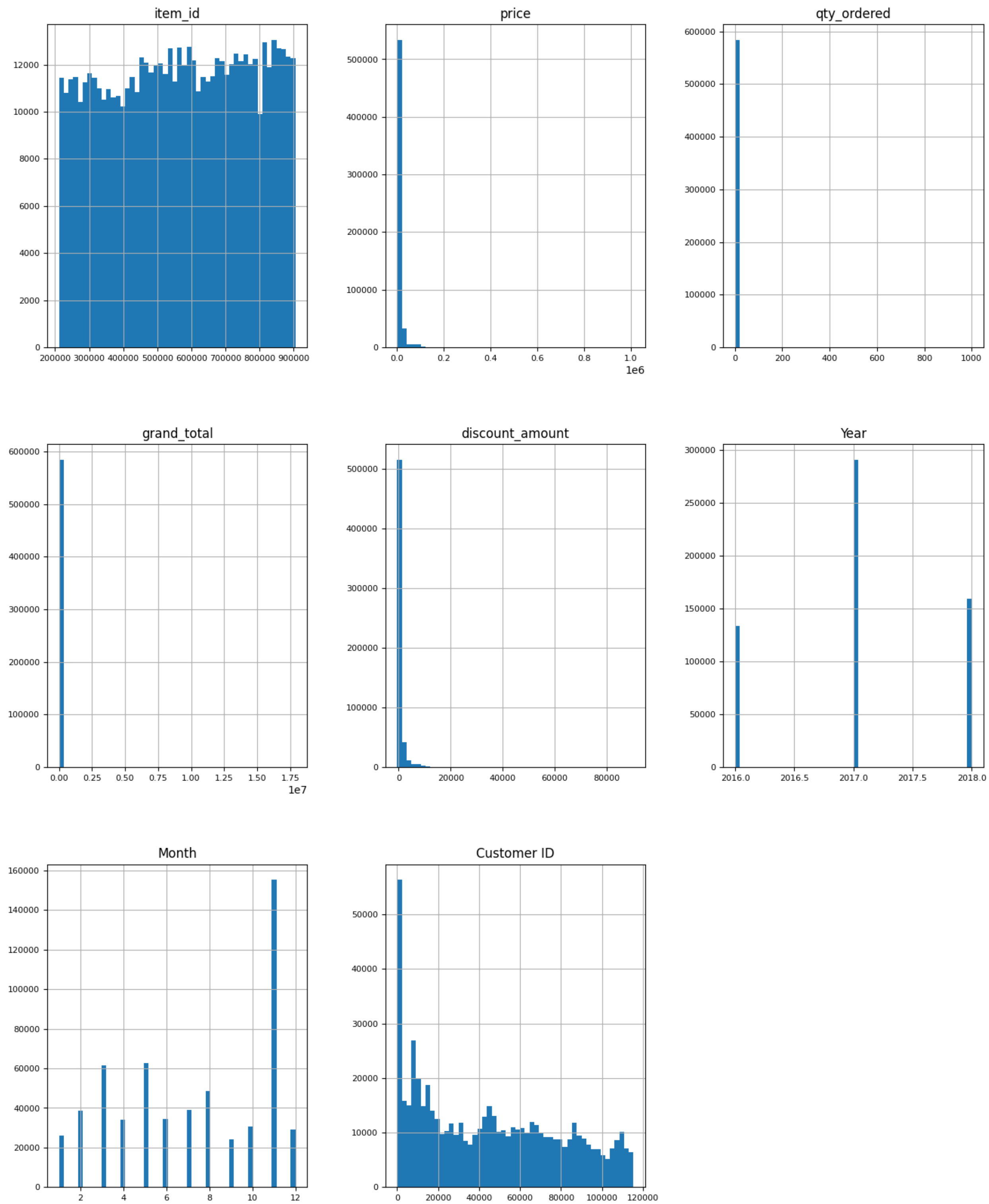
get the numerical data

```
df_num = data.select_dtypes(include = ['float64', 'int64'])
df_num.head()
```


{"type":"dataframe","variable_name":"df_num"}

plot them all in one graph:

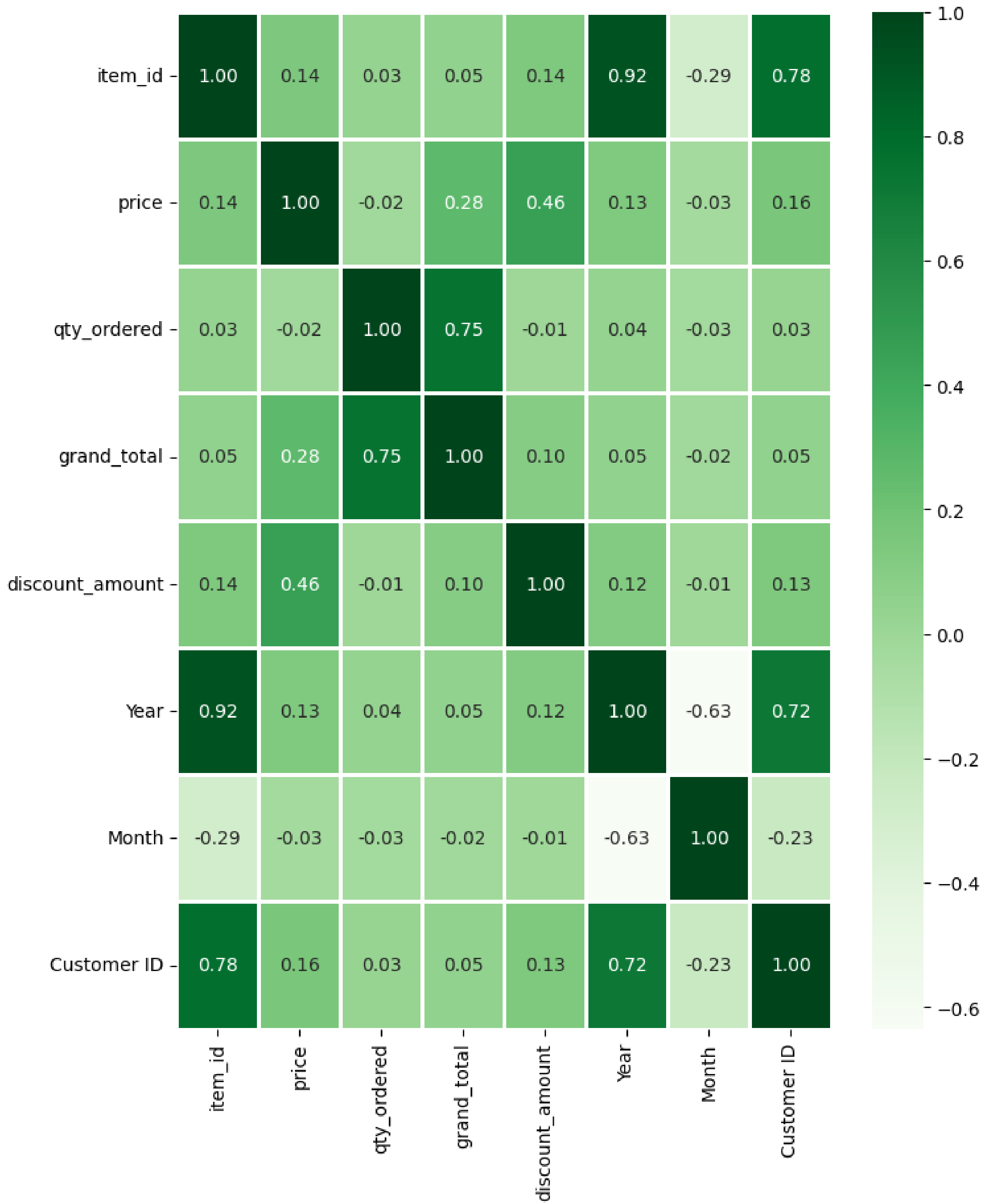
```
df_num.hist(figsize=(16, 20), bins=50, xlabelsize=8, ylabelsize=8);
```



Heat map for the numerical values

Python, and its libraries, make lots of things easy. For example, once the correlation matrix is defined , it can be passed to Seaborn's heatmap() method to create a heatmap (or headgrid). The basic idea of heatmaps is that they replace numbers with colors of varying shades, as indicated by the scale on the right. Cells that are lighter have higher values of r. This type of visualization can make it much easier to spot linear relationships between variables than a table of numbers. For example, if I focus on the “Strength” column, I immediately see that “Cement” and “FlyAsh” have the largest positive correlations whereas “Slag” has the large negative correlation.

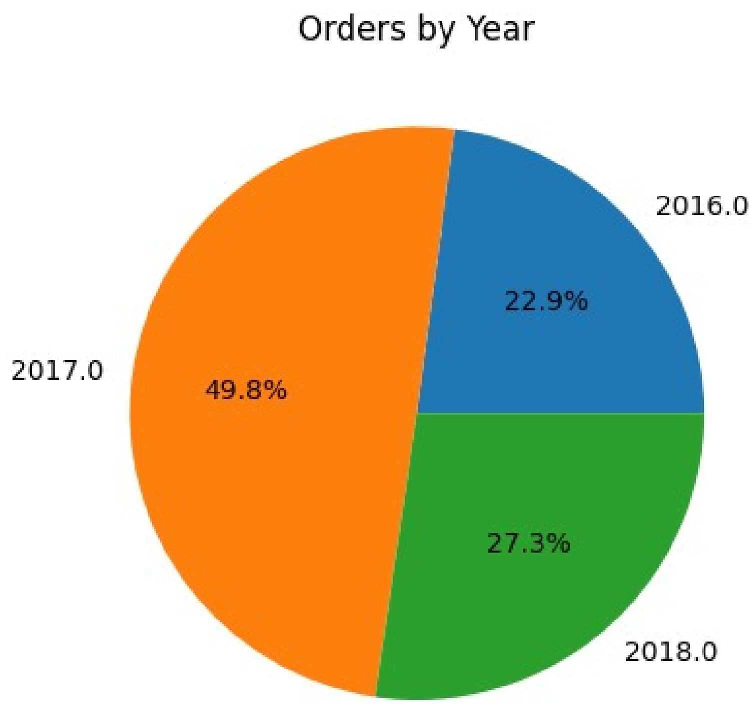
```
plt.figure(figsize = (8, 10))
sns.heatmap(df_num.corr(), annot = True, fmt = '0.2f', annot_kws =
{'size' : 10}, linewidth = 2, linecolor = 'white', cmap="Greens")
plt.show()
```



orders count percent by year

```
monthly_counts = data.groupby('Year')['item_id'].count()
plt.pie(monthly_counts, labels=monthly_counts.index, autopct='%1.1f%
```

```
%')
plt.title('Orders by Year')
plt.show()
```



Sales in each year by the grand total

In this drawing we can see all sales over the given three years, as it shows them in detail by month

```
def data_filter(year, xaxis, yaxis):
    df_year_filter = data[data['Year'] == year]
    df_cat_sales = df_year_filter.groupby(xaxis)
[yaxis].sum().reset_index()
    df_cat_sort = df_cat_sales.sort_values([xaxis], ascending=True)
    return df_cat_sort

data_2016 = data_filter(2016, "created_at", "grand_total")
data_2017 = data_filter(2017, "created_at", "grand_total")
data_2018 = data_filter(2018, "created_at", "grand_total")

trace_2016 = go.Scatter(x=data_2016['created_at'],
y=data_2016['grand_total'], mode='lines', name='2016')
trace_2017 = go.Scatter(x=data_2017['created_at'],
y=data_2017['grand_total'], mode='lines', name='2017')
trace_2018 = go.Scatter(x=data_2018['created_at'],
```

```
y=data_2018['grand_total'], mode='lines', name='2018')

layout = go.Layout(title='Sales by Year',
                   xaxis=dict(title='Date'),
                   yaxis=dict(title='Total Sales'),
                   )
fig = go.Figure(data=[trace_2016, trace_2017, trace_2018],
layout=layout)
fig.show()

ax = sns.scatterplot(x="qty_ordered", y="price", data=data)
ax.set_title("qty_ordered vs price")
ax.set_xlabel("qty_ordered");
```

find the Top selling category in all years

```
tops = data.groupby('category_name_1')
topsprice= tops['price'].agg(np.sum)
topsqty= tops['qty_ordered'].agg(np.sum)

print("Top selling category\n", topsqty)
```

```
Top selling category
 category_name_1
Appliances                 58203.0
Beauty & Grooming          53790.0
Books                       2641.0
Computing                  17251.0
Entertainment              27419.0
Health & Sports            21420.0
Home & Living              30065.0
Kids & Baby                18565.0
Men's Fashion             101424.0
Mobiles & Tablets         132695.0
Others                     84916.0
School & Education          4136.0
Soghaat                    47418.0
Superstore                 82542.0
Women's Fashion            64216.0
\N                          9647.0
Name: qty_ordered, dtype: float64
```

```
cat_total = data.groupby(['category_name_1'])
['grand_total'].sum().sort_values(ascending=False)
fig = px.bar(cat_total, x=cat_total.index, y=cat_total.values,
text='grand_total')
fig.update_traces(texttemplate='%{text:.2s}',textposition='outside',
marker_color='green')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
fig.update_layout(yaxis_title='Total payments')
fig.show()
```

we notes that we have 18 unique value of payment method

```
data['payment_method'] = data['payment_method'].astype('category')
data.describe(include='category')
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 4,\n  \"fields\": [\n    {\n      \"column\": \"payment_method\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          18,\n          \"271960\",\n          \"584524\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Month\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 276244.9183694004,\n        \"min\": 11.0,\n        \"max\": 584524.0,\n        \"num_unique_values\": 4,\n        \"samples\": [\n          12.0,\n          155456.0,\n          584524.0\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


```
data['payment_method'].unique()
```

```
['cod', 'ublcreditcard', 'mygateway', 'customercredit',
'cashatdoorstep', ..., 'Easypay', 'Easypay_MA', 'easypay_voucher',
'bankalfalah', 'apg']
Length: 18
Categories (18, object): ['Easypay', 'Easypay_MA', 'Payaxis',
'apg', ..., 'mcblite', 'mygateway',
                          'productcredit', 'ublcreditcard']
```

Top payment method in all years by total price

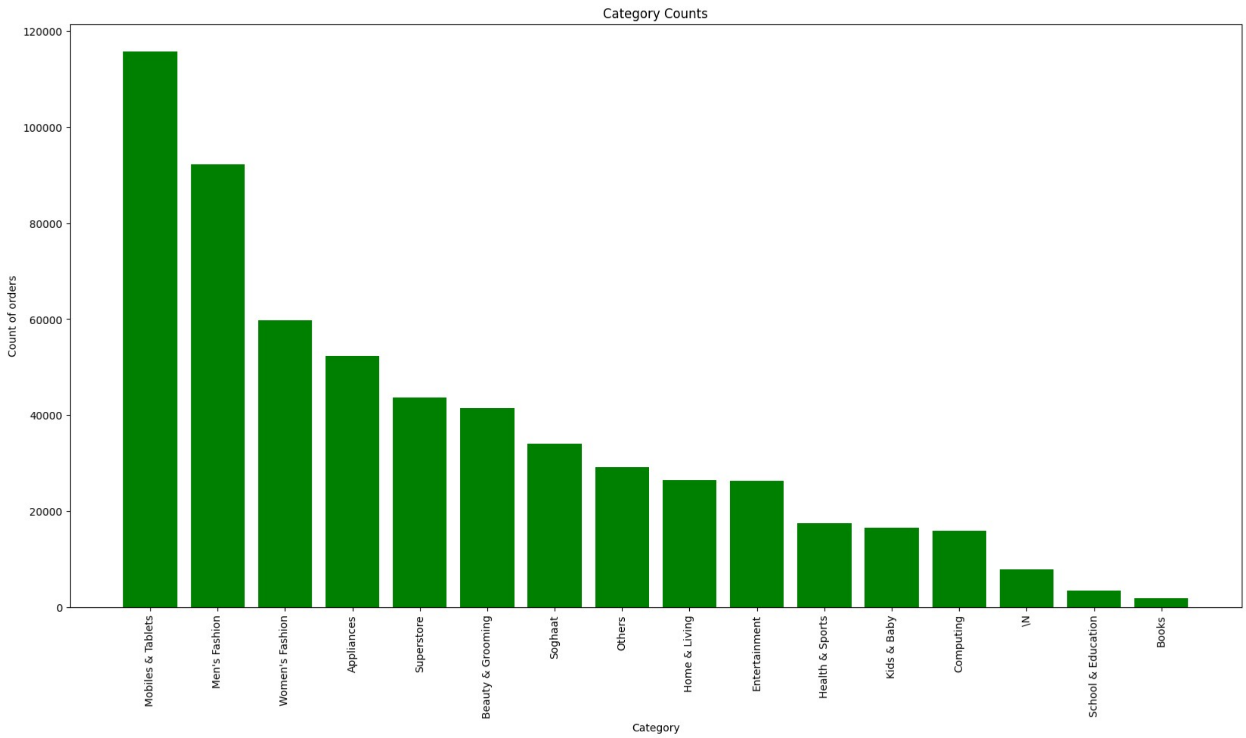
```
pay_met = data.groupby(['payment_method'])
['grand_total'].sum().sort_values(ascending=False)
fig = px.bar(pay_met, x=pay_met.index, y=pay_met.values,
text='grand_total')
fig.update_traces(texttemplate='%{text:.2s}',textposition='outside')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
fig.update_layout(yaxis_title='Total payments')
fig.show()
```

order status in all years

```
pay_met = data.groupby(['status'])
['grand_total'].sum().sort_values(ascending=False)
fig = px.bar(pay_met, x=pay_met.index, y=pay_met.values,
text='grand_total')
fig.update_traces(texttemplate='%{text:.2s}',textposition='outside')
fig.update_layout(uniformtext_minsize=8, uniformtext_mode='hide')
fig.update_layout(yaxis_title='Total orders')
fig.show()
```

Number of orders for every Category

```
cat_counts = data['category_name_1'].value_counts()
plt.figure(figsize=(20,10))
plt.bar(cat_counts.index, cat_counts, color='green')
plt.title('Category Counts')
plt.xlabel('Category')
plt.ylabel('Count of orders')
plt.xticks(ticks=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15], rotation=90)
plt.show()
```



top10 products ordered products in all years

```
top10_products = data['sku'].value_counts().nlargest(10).to_frame()
top10_products
```


{"summary":"{\n \"name\": \"top10_products\",\n \"rows\": 10,\n \"fields\": [\n   {\n   \"column\": \"sku\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 10,\n   \"samples\": [\n   \"unilever_Deal-6\",\n   \"Al Muhafiz Sohan Halwa Almond\",\n   \"emart_00-1\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"count\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 794,\n   \"min\": 1173,\n   \"max\": 3775,\n   \"num_unique_values\": 10,\n   \"samples\": [\n   1213,\n   2258,\n   1391\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"top10_products"}

To see the dashboard on tableau click here

Conclusion:

```
• The top category sales is mobile & tablets and its have the top
count of discount amount.
• The most payment method used is payaxis and then cod.
• In 2017, it was the best seller according to the data provided.
```

• The top 10 orders products was from the category mobiles & tablets.
• In november 2016 & 2017 was the most order products and in august was the most
orders in all years.