Exploratory_Data_Analysis

May 18, 2025

Table of Contents

F. Romaric Berger

[2]:
```
from IPython.display import Image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sweetviz as sv
from autoviz.AutoViz_Class import AutoViz_Class
%matplotlib inline
```

```
Imported v0.1.905. Please call AutoViz in this sequence:
    AV = AutoViz_Class()
    %matplotlib inline
    dfte = AV.AutoViz(filename, sep=',', depVar='', dfte=None, header=0,
verbose=1, lowess=False,
                chart_format='svg',max_rows_analyzed=150000,max_cols_analyzed=30,
save_plot_dir=None)
```

To understand the Olist dataset, we will focus on the data that are at the core of any e-commerce organisation business model. We will focus on orders, product purchased, customers and their reviews. - The order gives us information about who buys what and when. - The product purchased tells us what drives the revenue, it allows us to see bestselling products, poor working product, allows category level analysis and can be connected to review, returns etc. - The customers data allows segmentation and retention analysis - The reviews allows us to perform sentiment analysis, show satisfaction and dissatisfaction. The **Seller** workflow

The seller:

1. joins Olist
2. uploads their product catalogues
    - (Olist) displays these catalogues to existing marketplaces (Amazon, Bahia, Walmart, …)
3. gets notified whenever a product is sold
4. hands over the ordered items to third-party logistic carriers

Note: Multiple sellers can be involved in one customer's order!

The **Customer** workflow

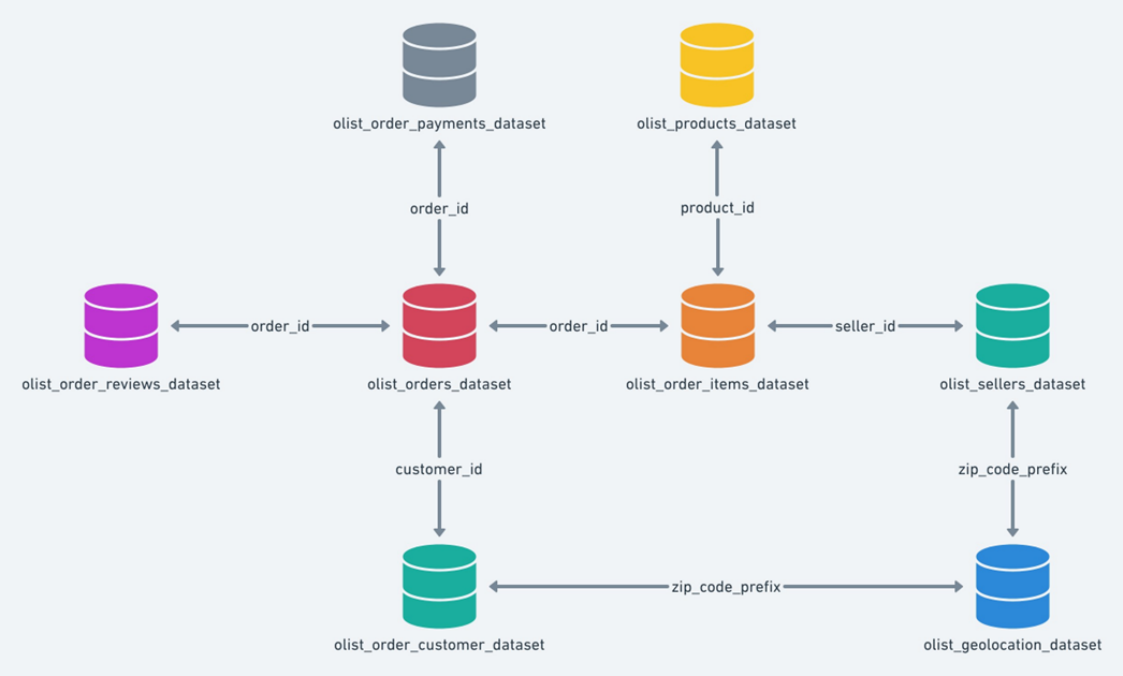
The customer:

1. browses products on marketplaces (Amazon, Bahia, Walmart, …)
2. purchases products listed via store
3. gets an expected date for delivery
    - *ETA = Estimated Time of Arrival (of the orders)*
4. receives the order(s)
5. leaves a review

Note: Between 2016 and mid-2018, a review could be left as soon as the order was sent, meaning that a customer could potentially leave a review for a product they hadn't received yet! It is showing the whole customer journey, from browsing to placing an order, receiving the product(s) he purchased to leaving a review.

[2]:
```
Image("Image/olist_erd.png")
```

[2]:

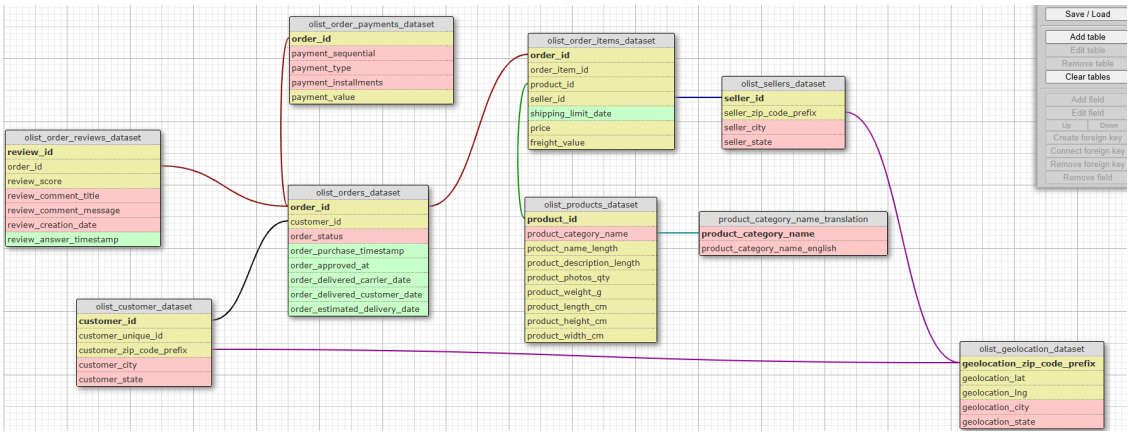


1 Objectives

Understand basic information about the company, and simple reflection on which products, categories, regions they should target or avoid.

2 Data preparation

[3]:
```
Image('Image/olist_erd_details.png')
```

[3]:


[3]:
```
# Load CSVs
orders = pd.read_csv('data/olist_orders_dataset.csv')
customers = pd.read_csv('data/olist_customers_dataset.csv')
reviews = pd.read_csv('data/olist_order_reviews_dataset.csv')
order_items = pd.read_csv('data/olist_order_items_dataset.csv')
products = pd.read_csv('data/olist_products_dataset.csv')
translation = pd.read_csv('data/product_category_name_translation.csv')
```

[4]:
```
# Merge datasets
data = orders.merge(customers, on='customer_id', how='left') \
    .merge(order_items, on='order_id', how='left') \
    .merge(reviews, on='order_id', how='left') \
    .merge(products, on='product_id', how='left') \
    .merge(translation, on='product_category_name', how='left')

data.head()
```

[4]:

	order_id	customer_id	\
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	

	order_status	order_purchase_timestamp	order_approved_at	\
0	delivered	2017-10-02 10:56:33	2017-10-02 11:07:15	
1	delivered	2018-07-24 20:41:37	2018-07-26 03:24:27	
2	delivered	2018-08-08 08:38:49	2018-08-08 08:55:23	
3	delivered	2017-11-18 19:28:06	2017-11-18 19:45:59	
4	delivered	2018-02-13 21:18:39	2018-02-13 22:20:29	

	order_delivered_carrier_date	order_delivered_customer_date	\
0	2017-10-04 19:55:00	2017-10-10 21:25:13	
1	2018-07-26 14:31:00	2018-08-07 15:27:45	
2	2018-08-08 13:50:00	2018-08-17 18:06:29	
3	2017-11-22 13:39:59	2017-12-02 00:28:42	
4	2018-02-14 19:46:34	2018-02-16 18:17:02	

	order_estimated_delivery_date	customer_unique_id	\
0	2017-10-18 00:00:00	7c396fd4830fd04220f754e42b4e5bff	
1	2018-08-13 00:00:00	af07308b275d755c9edb36a90c618231	
2	2018-09-04 00:00:00	3a653a41f6f9fc3d2a113cf8398680e8	
3	2017-12-15 00:00:00	7c142cf63193a1473d2e66489a9ae977	
4	2018-02-26 00:00:00	72632f0f9dd73dfee390c9b22eb56dd6	

	customer_zip_code_prefix	customer_city	customer_state	\
0	3149	sao paulo	SP	

```
1            47813                          barreiras       BA
2            75265                         vianopolis       GO
3            59296           sao goncalo do amarante       RN
4             9195                         santo andre       SP

   order_item_id             product_id             \
0            1.0  87285b34884572647811a353c7ac498a
1            1.0  595fac2a385ac33a80bd5114aec74eb8
2            1.0  aa4383b373c6aca5d8797843e5594415
3            1.0  d0b61bfb1de832b15ba9d266ca96e5b0
4            1.0  65266b2da20d04dbe00c5c2d3bb7859e

                 seller_id             shipping_limit_date    price  \
0  3504c0cb71d7fa48d967e0e4c94d59d9  2017-10-06 11:07:15   29.99
1  289cdb325fb7e7f891c38608bf9e0962  2018-07-30 03:24:27  118.70
2  4869f7a5dfa277a7dca6462dcf3b52b2  2018-08-13 08:55:23  159.90
3  66922902710d126a0e7d26b0e3805106  2017-11-23 19:45:59   45.00
4  2c9e548be18521d1c43cde1c582c6de8  2018-02-19 20:31:37   19.90

   freight_value             review_id              review_score  \
0           8.72  a54f0611adc9ed256b57ede6b6eb5114           4.0
1          22.76  8d5266042046a06655c8db133d120ba5           4.0
2          19.22  e73b67b67587f7644d5bd1a52deb1b01           5.0
3          27.20  359d03e676b3c069f62cadba8dd3f6e8           5.0
4           8.72  e50934924e227544ba8246aeb3770dd4           5.0

  review_comment_title  \
0                  NaN
1   Muito boa a loja
2                  NaN
3                  NaN
4                  NaN

review_comment_message
\
0  Não testei o produto ainda, mas ele veio correto e em boas condições. Apenas
a caixa que veio bem amassada e danificada, o que ficará chato, pois se trata de
um presente.
1
Muito bom o produto.
2
NaN
3                                                                    O produto
foi exatamente o que eu esperava e estava descrito no site e chegou bem antes da
data prevista.
4
NaN
```

```
  review_creation_date review_answer_timestamp product_category_name   \
0  2017-10-11 00:00:00    2017-10-12 03:43:48   utilidades_domesticas
1  2018-08-08 00:00:00    2018-08-08 18:37:50              perfumaria
2  2018-08-18 00:00:00    2018-08-22 19:07:58              automotivo
3  2017-12-03 00:00:00    2017-12-05 19:21:58               pet_shop
4  2018-02-17 00:00:00    2018-02-18 13:02:51              papelaria

   product_name_lenght  product_description_lenght  product_photos_qty  \
0                 40.0                       268.0                 4.0
1                 29.0                       178.0                 1.0
2                 46.0                       232.0                 1.0
3                 59.0                       468.0                 3.0
4                 38.0                       316.0                 4.0

   product_weight_g  product_length_cm  product_height_cm  product_width_cm  \
0             500.0               19.0                8.0              13.0
1             400.0               19.0               13.0              19.0
2             420.0               24.0               19.0              21.0
3             450.0               30.0               10.0              20.0
4             250.0               51.0               15.0              15.0

  product_category_name_english
0                    housewares
1                     perfumery
2                          auto
3                     pet_shop
4                    stationery
```

2.1 We start by looking at the basic information form the data

[4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114092 entries, 0 to 114091
Data columns (total 33 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_id                       114092 non-null  object
 1   customer_id                    114092 non-null  object
 2   order_status                   114092 non-null  object
 3   order_purchase_timestamp       114092 non-null  object
 4   order_approved_at              113930 non-null  object
 5   order_delivered_carrier_date   112112 non-null  object
 6   order_delivered_customer_date  110839 non-null  object
 7   order_estimated_delivery_date  114092 non-null  object
 8   customer_unique_id             114092 non-null  object
```

```
 9   customer_zip_code_prefix       114092 non-null  int64
 10  customer_city                  114092 non-null  object
 11  customer_state                 114092 non-null  object
 12  order_item_id                  113314 non-null  float64
 13  product_id                     113314 non-null  object
 14  seller_id                      113314 non-null  object
 15  shipping_limit_date            113314 non-null  object
 16  price                          113314 non-null  float64
 17  freight_value                  113314 non-null  float64
 18  review_id                      113131 non-null  object
 19  review_score                   113131 non-null  float64
 20  review_comment_title           13523 non-null   object
 21  review_comment_message         48166 non-null   object
 22  review_creation_date           113131 non-null  object
 23  review_answer_timestamp        113131 non-null  object
 24  product_category_name          111702 non-null  object
 25  product_name_lenght            111702 non-null  float64
 26  product_description_lenght     111702 non-null  float64
 27  product_photos_qty             111702 non-null  float64
 28  product_weight_g               113296 non-null  float64
 29  product_length_cm              113296 non-null  float64
 30  product_height_cm              113296 non-null  float64
 31  product_width_cm               113296 non-null  float64
 32  product_category_name_english  111678 non-null  object
dtypes: float64(11), int64(1), object(21)
memory usage: 28.7+ MB
```

[6]:
```
data.describe()
```

[6]:

	customer_zip_code_prefix	order_item_id	price	freight_value	\
count	114092.000000	113314.000000	113314.000000	113314.000000	
mean	35105.227308	1.198528	120.478701	19.979428	
std	29868.300916	0.707016	183.279678	15.783227	
min	1003.000000	1.000000	0.850000	0.000000	
25%	11250.000000	1.000000	39.900000	13.080000	
50%	24320.000000	1.000000	74.900000	16.260000	
75%	59022.000000	1.000000	134.900000	21.150000	
max	99990.000000	21.000000	6735.000000	409.680000	

	review_score	product_name_lenght	product_description_lenght	\
count	113131.000000	111702.000000	111702.000000	
mean	4.016998	48.777560	786.899250	
std	1.400074	10.024616	651.758866	
min	1.000000	5.000000	4.000000	
25%	4.000000	42.000000	348.000000	
50%	5.000000	52.000000	601.000000	
75%	5.000000	57.000000	985.000000	

```
max         5.000000         76.000000              3992.000000

       product_photos_qty  product_weight_g  product_length_cm  \
count       111702.000000     113296.000000      113296.000000
mean             2.206908       2091.915037          30.162495
std              1.719500       3749.804597          16.151737
min              1.000000          0.000000           7.000000
25%              1.000000        300.000000          18.000000
50%              1.000000        700.000000          25.000000
75%              3.000000       1800.000000          38.000000
max             20.000000      40425.000000         105.000000

       product_height_cm  product_width_cm
count      113296.000000     113296.000000
mean           16.584513         23.003539
std            13.439206         11.708481
min             2.000000          6.000000
25%             8.000000         15.000000
50%            13.000000         20.000000
75%            20.000000         30.000000
max           105.000000        118.000000
```

[6]:
```
# Looking for columns with too many missing rows.
data.isna().sum()
```

[6]:
```
order_id                            0
customer_id                         0
order_status                        0
order_purchase_timestamp            0
order_approved_at                 162
order_delivered_carrier_date     1980
order_delivered_customer_date    3253
order_estimated_delivery_date       0
customer_unique_id                  0
customer_zip_code_prefix            0
customer_city                       0
customer_state                      0
order_item_id                     778
product_id                        778
seller_id                         778
shipping_limit_date               778
price                             778
freight_value                     778
review_id                         961
review_score                      961
review_comment_title           100569
review_comment_message          65926
```

```
review_creation_date              961
review_answer_timestamp           961
product_category_name            2390
product_name_lenght              2390
product_description_lenght       2390
product_photos_qty               2390
product_weight_g                  796
product_length_cm                 796
product_height_cm                 796
product_width_cm                  796
product_category_name_english    2414
dtype: int64
```

2.2 Preparing and cleaning the data

[7]:
```
# transform the date related data to the right format
data['order_purchase_timestamp'] = pd.
 ↪to_datetime(data['order_purchase_timestamp'])
data['order_approved_at'] = pd.to_datetime(data['order_approved_at'])
data['order_delivered_carrier_date'] = pd.
 ↪to_datetime(data['order_delivered_carrier_date'])
data['order_delivered_customer_date'] = pd.
 ↪to_datetime(data['order_delivered_customer_date'])
data['order_estimated_delivery_date'] = pd.
 ↪to_datetime(data['order_estimated_delivery_date'])
```

[8]:
```
# To enhance the understanding of customers satisfaction, we calculate if␣
 ↪deliveries were late or early
data["order_reception_delay"] = data["order_estimated_delivery_date"] -␣
 ↪data["order_delivered_customer_date"]
```

[9]:
```
# Cleaning
# Drop columns not inherently related to the objectives, especially in the␣
 ↪context of EDA
# drop columns with too many missing values

data = data.drop(columns=['order_delivered_carrier_date','order_approved_at',␣
 ↪'review_id','review_comment_title',
                           'review_comment_message', 'product_name_lenght',␣
 ↪'product_description_lenght',
                           'product_photos_qty','product_weight_g',␣
 ↪'product_length_cm', 'product_height_cm','product_width_cm',
                           'product_category_name', 'customer_id'])
```

[10]:
```
data.columns
```

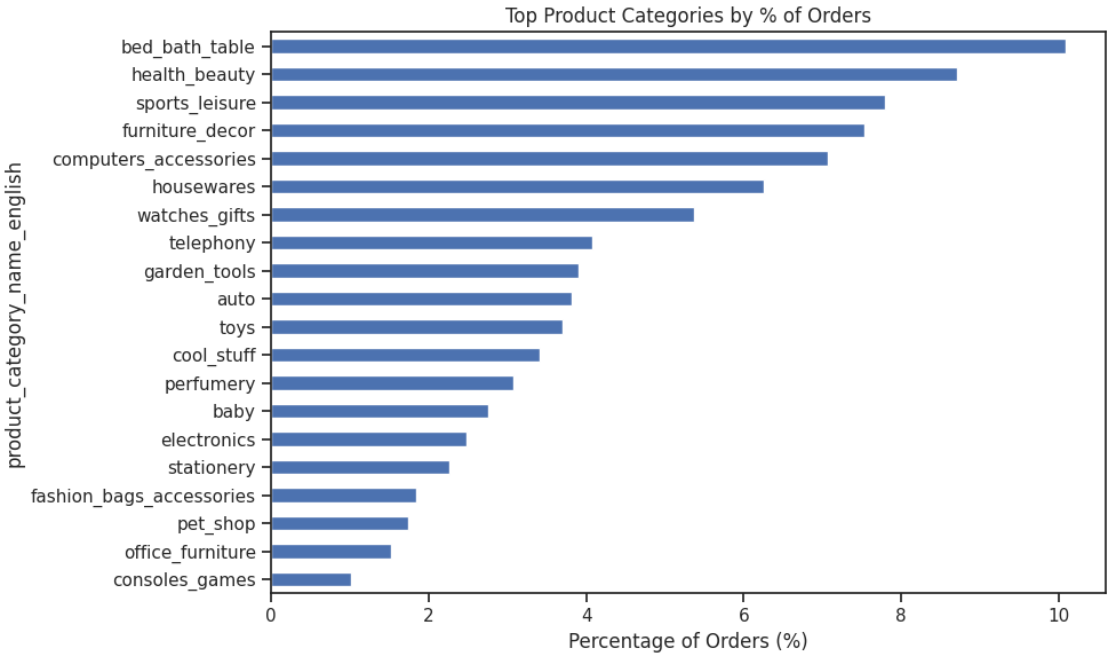
[10]:
```
Index(['order_id', 'order_status', 'order_purchase_timestamp',
       'order_delivered_customer_date', 'order_estimated_delivery_date',
       'customer_unique_id', 'customer_zip_code_prefix', 'customer_city',
       'customer_state', 'order_item_id', 'product_id', 'seller_id',
       'shipping_limit_date', 'price', 'freight_value', 'review_score',
       'review_creation_date', 'review_answer_timestamp',
       'product_category_name_english', 'order_reception_delay'],
      dtype='object')
```

3 Categories and products_Exploration

[10]:
```
cat_number = data['product_category_name_english'].nunique()
print(f'There is {cat_number} unique category.')
```

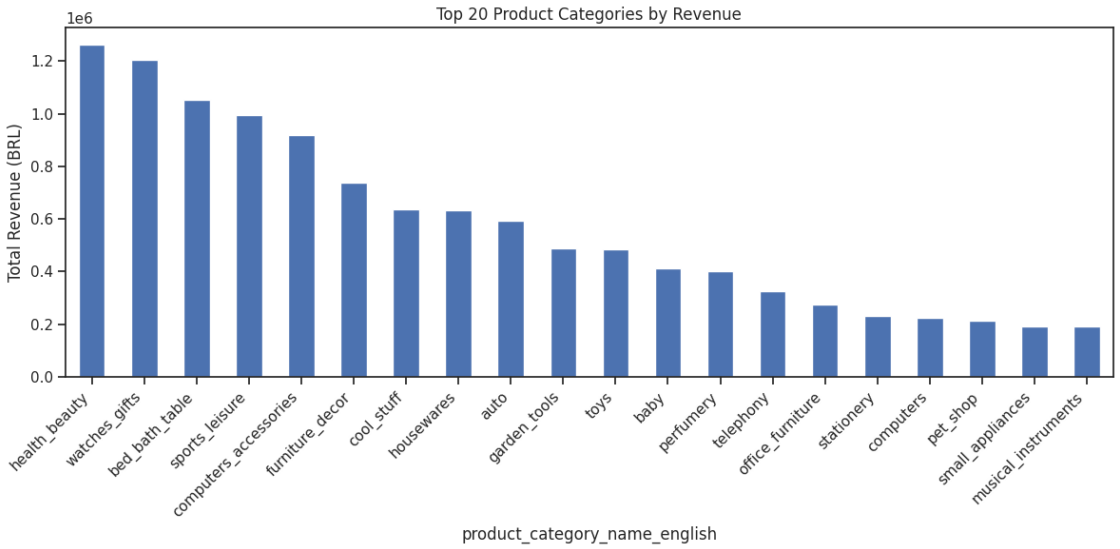
```
There is 71 unique category.
```

[12]:
```
# Let's classify them in order of volume of sales

category_distribution = data['product_category_name_english'].
 ↪value_counts(normalize=True, dropna=True) * 100


plt.figure(figsize=(10, 6))
category_distribution.head(20).sort_values().plot(kind='barh')
plt.xlabel('Percentage of Orders (%)')
plt.title('Top Product Categories by % of Orders')
plt.tight_layout()
plt.show()
```



We can see here the categories that are the most sold, but are they the ones that brings in the most revenue?

[13]:
```
# Drop rows with missing prices or categories
filtered_data = data[['product_category_name_english', 'price']].dropna()

# Group by category and sum the revenue
revenue_by_category = filtered_data.
 ↪groupby('product_category_name_english')['price'].sum()
revenue_by_category = revenue_by_category.sort_values(ascending=False)

plt.figure(figsize=(12, 6))
revenue_by_category.head(20).plot(kind='bar')
plt.ylabel('Total Revenue (BRL)')
plt.title('Top 20 Product Categories by Revenue')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



We have 2 important informative plots - Sales Volume (%) per Category (from value_counts(normalize=True)) - Total Revenue (BRL) per Category (from groupby('category')['price'].sum())

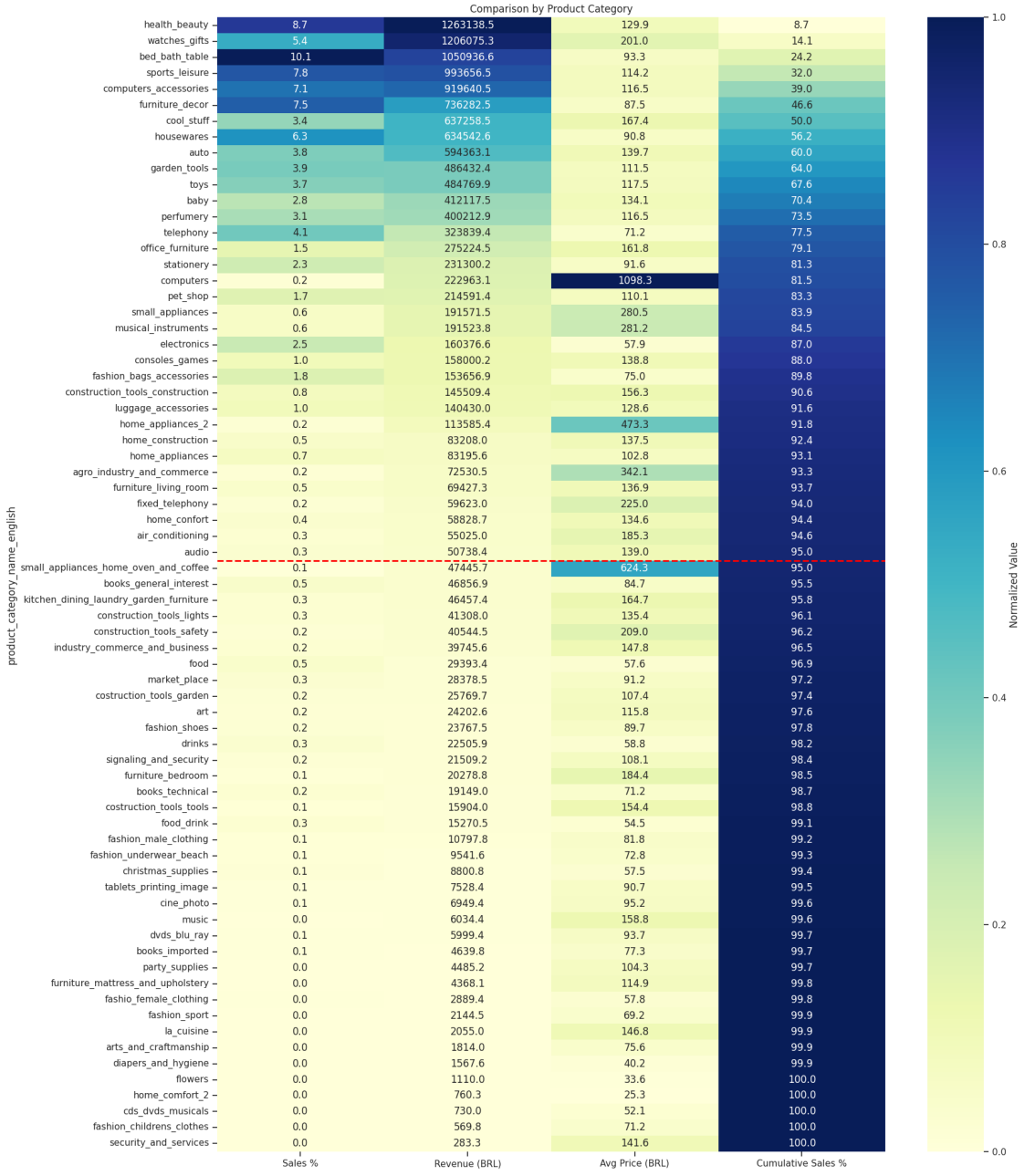
Let's visualize the comparison of sales percentage vs revenue with a heatmap! (And add the average price per category)

[14]:
```
top_categories = category_distribution.index

# Filter & align both metrics
sales_pct = category_distribution[top_categories]
revenue = revenue_by_category[top_categories]

# Add average price
avg_price = data.groupby('product_category_name_english')['price'].mean()

cat_heatmap_df = pd.DataFrame({
    'Sales %': sales_pct,
    'Revenue (BRL)': revenue,
    'Avg Price (BRL)': avg_price[top_categories]
})
```

[15]:
```
# Sort the heatmap by revenue
cat_heatmap_df = cat_heatmap_df.sort_values(by='Revenue (BRL)', ascending=False)

# Add cumulative Sales %
cat_heatmap_df['Cumulative Sales %'] = cat_heatmap_df['Sales %'].cumsum()

#Find the index (row) where cumulative sales exceed 95%
```

```
cutoff_index = cat_heatmap_df['Cumulative Sales %'].searchsorted(95)

# We normalize so that all metrics are comparable on the same color scale for␣
 ↪the heatmap(only!!)
cat_normalized_df = (cat_heatmap_df - cat_heatmap_df.min()) / (cat_heatmap_df.
 ↪max() - cat_heatmap_df.min())
```

[16]:
```
plt.figure(figsize=(18, 20))
sns.heatmap(
    cat_normalized_df,
    annot=cat_heatmap_df.round(1),
    fmt='',
    cmap='YlGnBu',
    cbar_kws={'label': 'Normalized Value'}
)
plt.axhline(cutoff_index, color='red', linewidth=2, linestyle='--')
plt.title('Comparison by Product Category')
plt.tight_layout()
plt.show()
```



The color intensity tells you how high or low that metric is compared to others. The annotations show actual values for clarity.

We can observe that out of the 71 categories, 95% of the revenue is made by:

[17]:
```
print(f" {cutoff_index + 1} categories only!")
```

```
 35 categories only!
```

How many product would it save in the inventory if we were not selling the 36 categories that do

not sell well? Could it reduce significantly the cost?

[11]:
```
# How many product do we have?
product_count = data['product_id'].nunique()
print(f'Olist sellers sell {product_count} products.')
```

```
Olist sellers sell 32951 products.
```

[20]:
```
# How many left after filtering to keep only products from the top 35␣
 ↪categories that generate revenue.

top_cat = cat_heatmap_df.head(35).index.tolist()

top_data = data[data['product_category_name_english'].isin(top_cat)]

top_cat_product_count = top_data['product_id'].nunique()

print(f'There would be {top_cat_product_count} products left, which means the␣
 ↪last 36 categories countain only {product_count - top_cat_product_count}␣
 ↪products.')
```

```
There would be 30288 products left, which means the last 36 categories countain
only 2663 products.
```

[22]:
```
pct_bad_items = (product_count - top_cat_product_count)/product_count * 100
print(f'The percentage of products not in the top categories is {pct_bad_items:.
 ↪2f}%.')
```

```
The percentage of products not in the top categories is 8.08%.
```

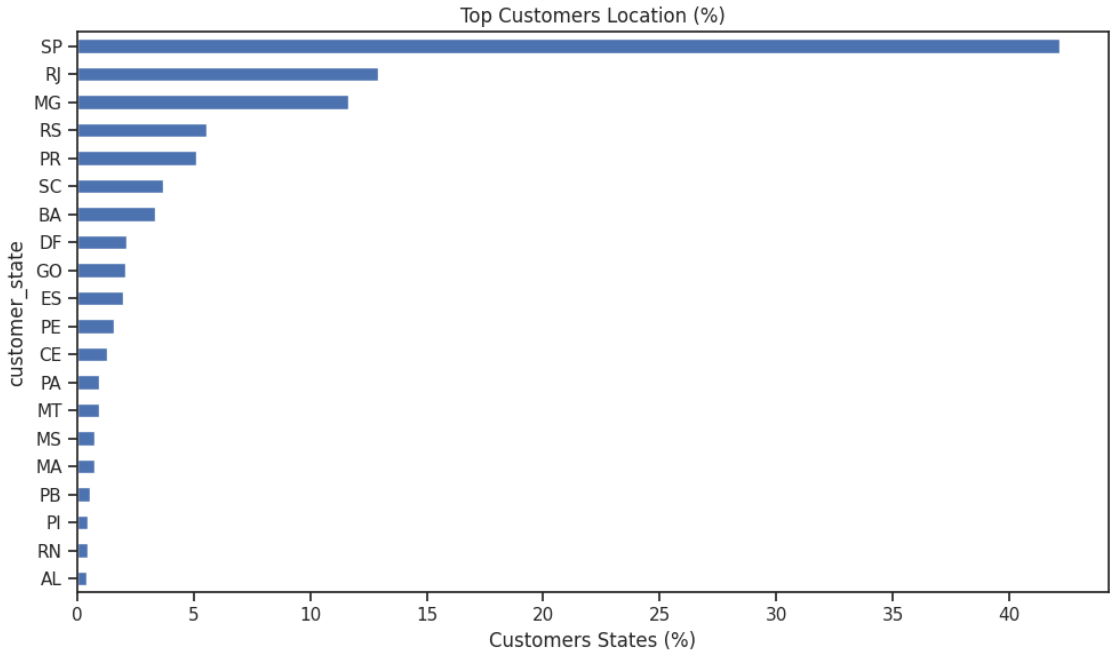
4 Regions_Exploration

[13]:
```
brl_states = data['customer_state'].nunique()
print(f'Olist operates in {brl_states} states in Brasil.')
```

```
Olist operates in 27 states in Brasil.
```

[20]:
```
# Where are the customers located?

customers_distribution = data['customer_state'].value_counts(normalize=True,␣
 ↪dropna=True) * 100


plt.figure(figsize=(10, 6))
customers_distribution.head(20).sort_values().plot(kind='barh')
plt.xlabel('Customers States (%)')
plt.title('Top Customers Location (%)')
```

```
plt.tight_layout()
plt.show()
```



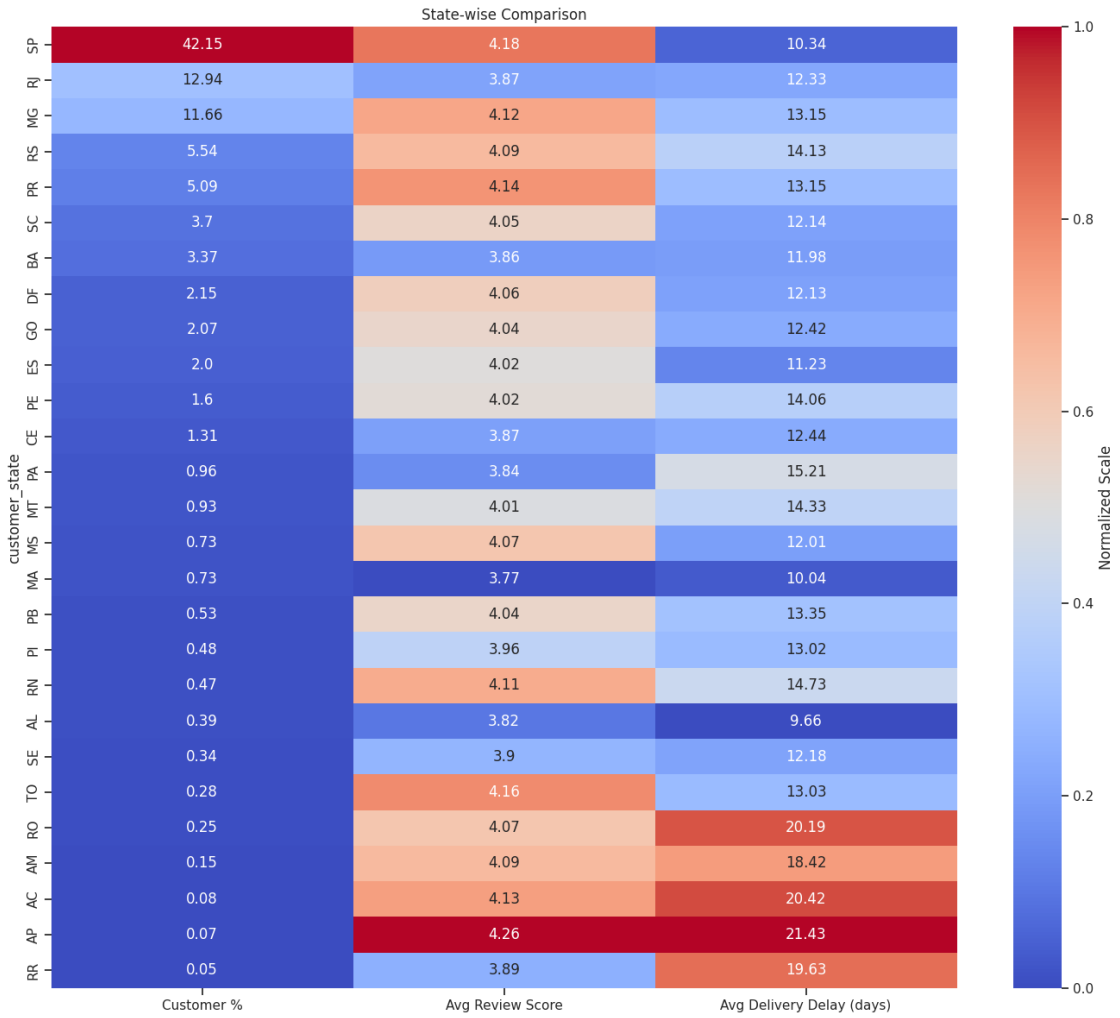
It would be interesting to see if the reviews score is correlated to states, as well as the delay, it could show a distribution problem that has to be fixed.

[21]:
```
# Drop rows with missing prices or categories
filtered_data_loc = data[['review_score', 'customer_state',␣
 ↪'order_reception_delay']].dropna()

# Group reviews and delay by states
states_review_score = filtered_data_loc.
 ↪groupby('customer_state')['review_score'].mean().sort_values(ascending=False)
states_delay = filtered_data_loc.
 ↪groupby('customer_state')['order_reception_delay'].median().
 ↪sort_values(ascending=False)
```

Once Again, we'll use a heatmap to show the relationship between these 3 information.

[22]:
```
# Ensure values are aligned by index
states = customers_distribution.index
review = states_review_score.reindex(states)

# Convert timedelta to float (in days)
delay_days = states_delay.reindex(states).dt.total_seconds() / 86400
```

```
# Combine and convert to numeric (just in case)
states_df = pd.DataFrame({
    'Customer %': customers_distribution[states],
    'Avg Review Score': review,
    'Avg Delivery Delay (days)': delay_days
}).astype(float).dropna()

# Normalize numeric values for heatmap coloring
normalized_states_df = (states_df - states_df.min()) / (states_df.max() -␣
 ↪states_df.min())

# Plot
plt.figure(figsize=(14, 12))
sns.heatmap(
    normalized_states_df,
    annot=states_df.round(2),
    fmt='',
    cmap='coolwarm',
    cbar_kws={'label': 'Normalized Scale'}
)
plt.title('State-wise Comparison')
plt.tight_layout()
plt.show()
```



4.1 Quick analysis

Potential Insight

Customer Satisfaction: States with higher review scores might indicate higher customer satisfaction.

Delivery Efficiency: States with lower delivery delays might have more efficient logistics or better infrastructure.

Market Focus: States with higher customer percentages might be key markets.

4.2 Effective Observation about review scores and delay.

- There is no clear relation between the review score and the delivery delay.

- There is an obvious problem with the order_estimated_delivery_date calculation that does not reflect the reality at all.

4.3 How to explain the repartition of customers and the delay from factual information.

São Paulo (SP) is by far the most populated state in Brazil.

As of the latest estimates:

São Paulo has ~46 million residents, about 20–22% of the country's total population.

It is also Brazil's main economic and logistical hub, which explains its dominant role in e-commerce.

States with high delivery delays: RO (Rondônia), AM (Amazonas), AC (Acre), AP (Amapá), RR (Roraima):

Geography: These states are vast, sparsely populated, and covered with dense forests and rivers (Rain forest). Roads are limited, and many areas are only accessible by boat or small aircraft. <= *Remote*

Infrastructure: Fewer major highways, fewer distribution centers, and less advanced logistical networks compared to more industrialized regions like the Southeast. <= *Underdeveloped*

Long transport distances: Most e-commerce products ship from Southeastern hubs (especially São Paulo), which are thousands of kilometers away from the North. <= *Far from economical hubs*

Weather and seasonal issues: Rainy seasons often flood roads and delay river transportation, which affects delivery schedules. <= *Very dependent from the season, which can explain the resilience of customers that gives good review regardless of delay*

5 Generated EDA visualization

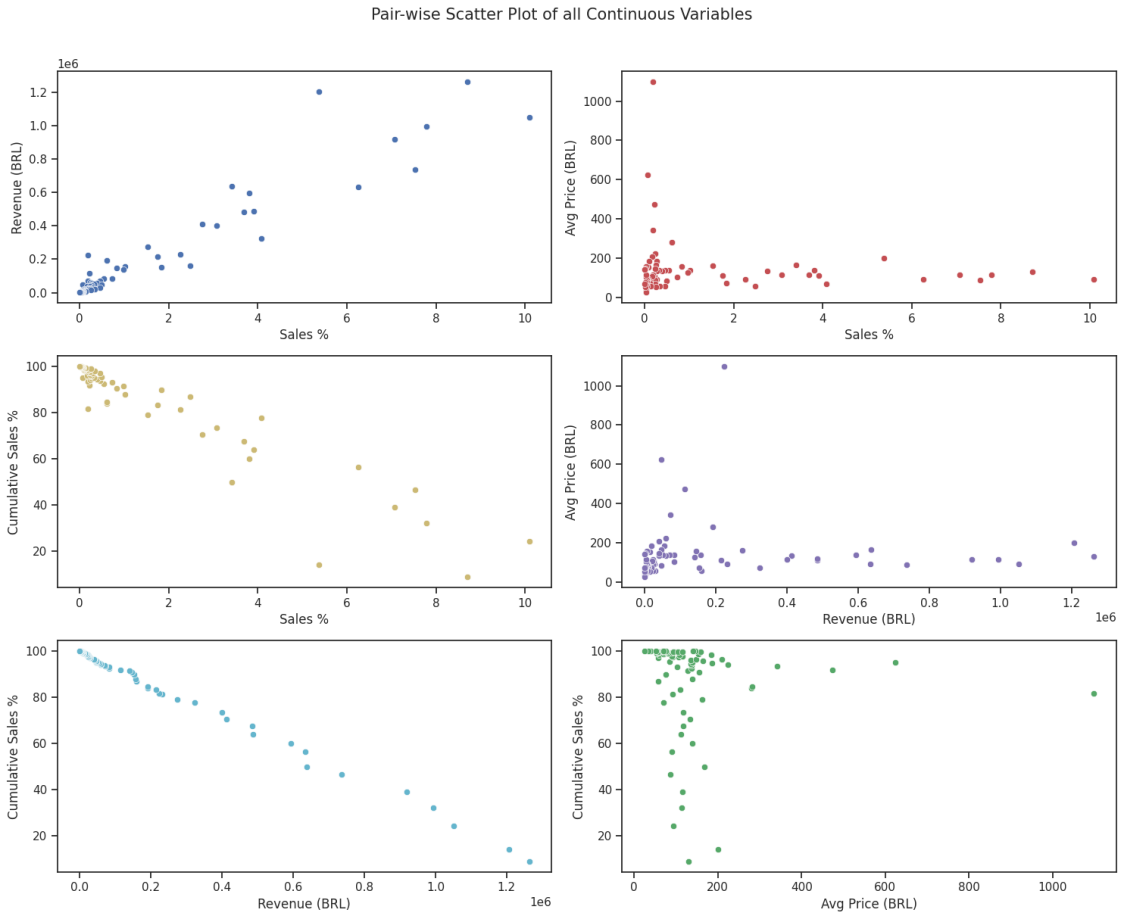
5.1 Categories reports

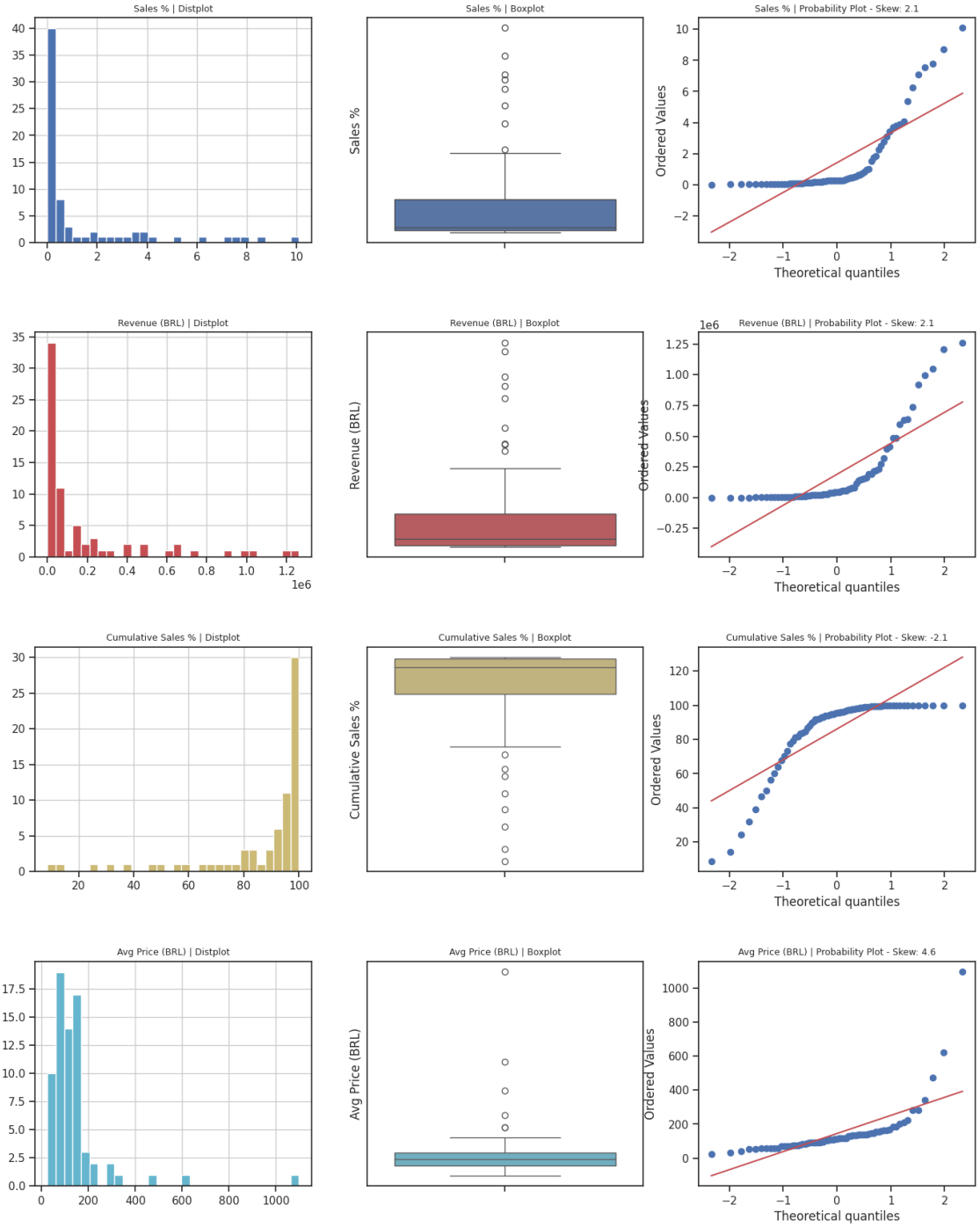
[23]:
```
AV = AutoViz_Class()
```

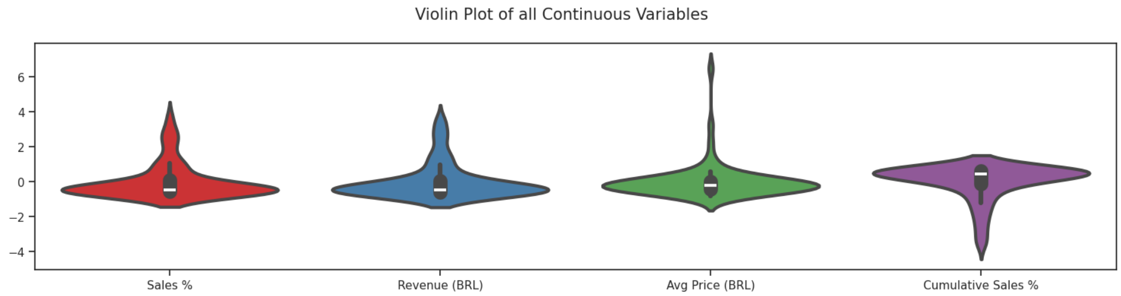
[24]:
```
report = AV.AutoViz(cat_heatmap_df)
```

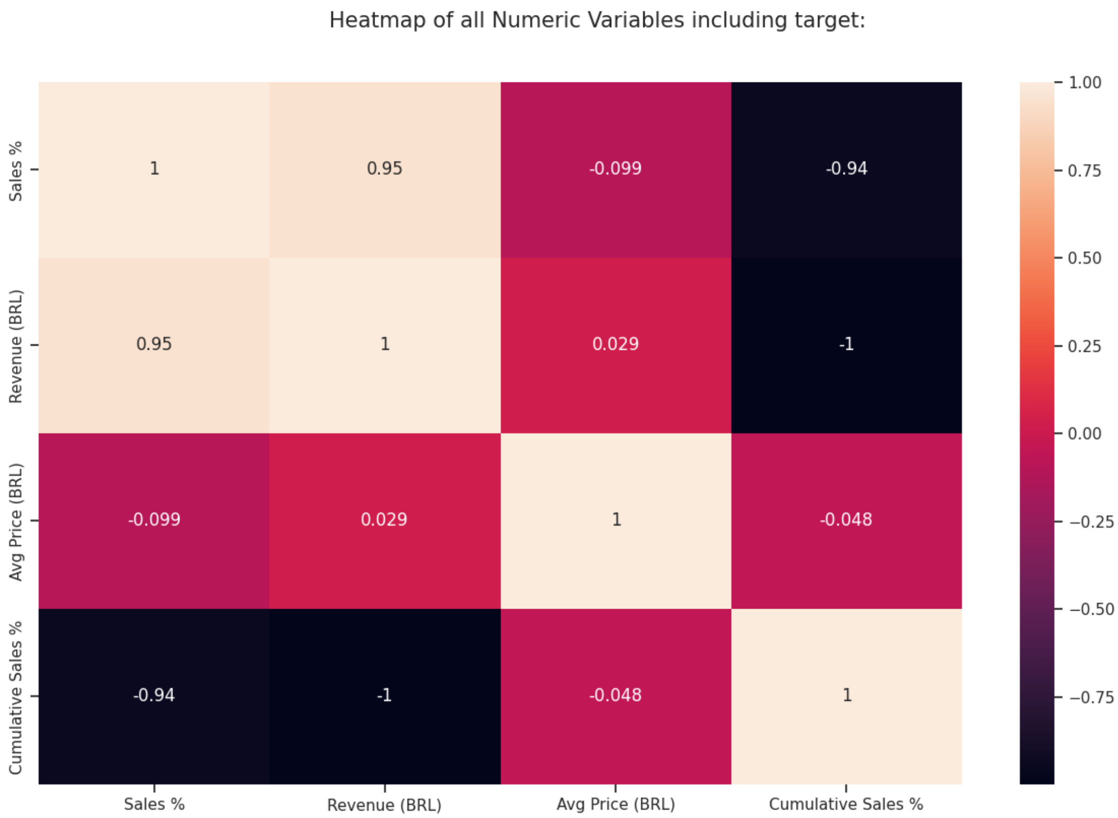
```
Shape of your Data Set loaded: (71, 4)
############################################################################
#######
######################### C L A S S I F Y I N G  V A R I A B L E S
####################
############################################################################
#######
Classifying variables in data set…
    Number of Numeric Columns =  4
    Number of Integer-Categorical Columns =  0
    Number of String-Categorical Columns =  0
    Number of Factor-Categorical Columns =  0
    Number of String-Boolean Columns =  0
    Number of Numeric-Boolean Columns =  0
    Number of Discrete String Columns =  0
    Number of NLP String Columns =  0
```

```
    Number of Date Time Columns =  0
    Number of ID Columns =  0
    Number of Columns to Delete =  0
    4 Predictors classified…
        No variables removed since no ID or low-information variables found in
data set
To fix these data quality issues in the dataset, import FixDQ from autoviz…
    All variables classified into correct types.

<pandas.io.formats.style.Styler at 0x7f63b4201630>

Number of All Scatter Plots = 10
```




Sales % | Distplot
Sales % | Boxplot
Sales %
Sales % | Probability Plot - Skew: 2.1
Ordered Values
Theoretical quantiles
Revenue (BRL) | Distplot
1e6
Revenue (BRL) | Boxplot
Revenue (BRL)
Revenue (BRL) | Probability Plot - Skew: 2.1
1e6
Ordered Values
Theoretical quantiles
Cumulative Sales % | Distplot
Cumulative Sales % | Boxplot
Cumulative Sales %
Cumulative Sales % | Probability Plot - Skew: -2.1
Ordered Values
Theoretical quantiles
Avg Price (BRL) | Distplot
Avg Price (BRL) | Boxplot
Avg Price (BRL)
Avg Price (BRL) | Probability Plot - Skew: 4.6
Ordered Values
Theoretical quantiles






```
All Plots done
Time to run AutoViz = 3 seconds

 ###################### AUTO VISUALIZATION Completed ########################
```

[25]:
```
cat_eda_report = sv.analyze(cat_heatmap_df)
```

```
                                        |
                   | [  0%]   00:…
```

[26]:
```
cat_eda_report.show_html("cat_eda_report.html")
```

```
Report cat_eda_report.html was generated! NOTEBOOK/COLAB USERS: the web browser
MAY not pop up, regardless, the report IS saved in your notebook/colab files.
```

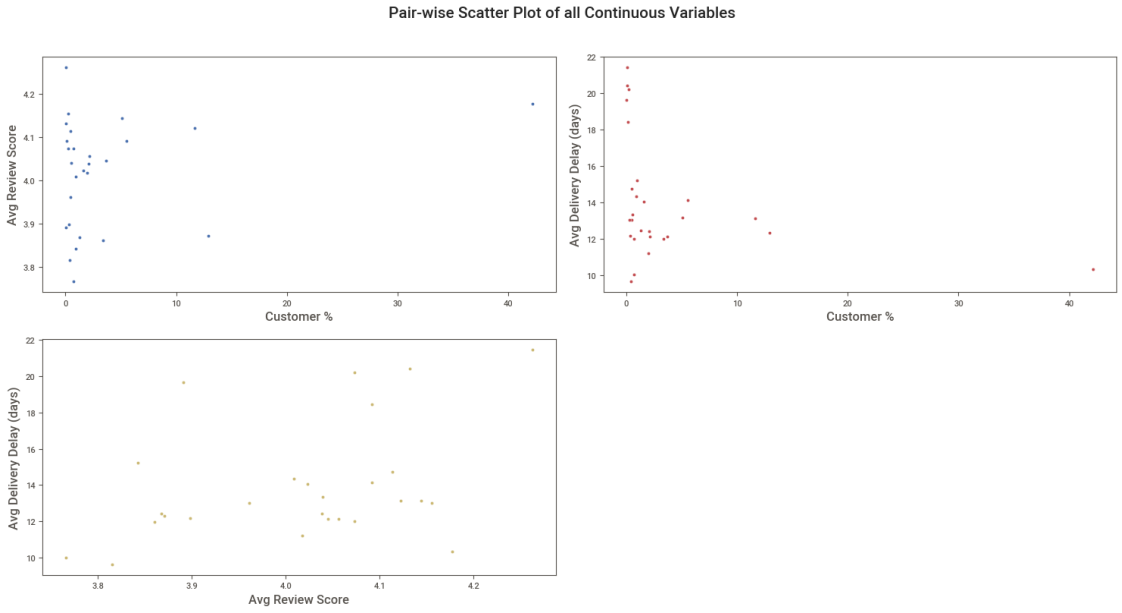
5.2 States reports

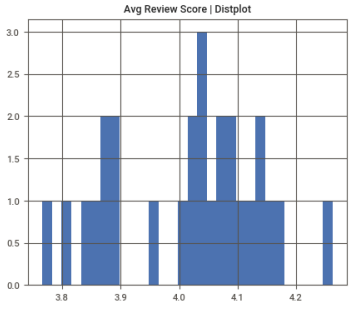
[27]:
```
states_report =  AV.AutoViz(states_df)
```

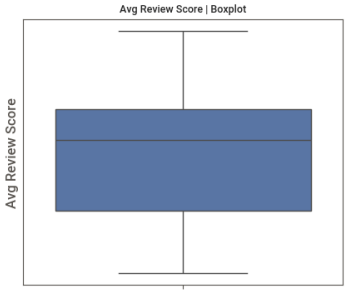
```
Shape of your Data Set loaded: (27, 3)
############################################################################################
#######
######################## C L A S S I F Y I N G  V A R I A B L E S
####################
############################################################################################
#######
Classifying variables in data set…
    Number of Numeric Columns =  3
    Number of Integer-Categorical Columns =  0
    Number of String-Categorical Columns =  0
    Number of Factor-Categorical Columns =  0
    Number of String-Boolean Columns =  0
    Number of Numeric-Boolean Columns =  0
    Number of Discrete String Columns =  0
    Number of NLP String Columns =  0
    Number of Date Time Columns =  0
    Number of ID Columns =  0
    Number of Columns to Delete =  0
    3 Predictors classified…
        No variables removed since no ID or low-information variables found in
data set
To fix these data quality issues in the dataset, import FixDQ from autoviz…
    All variables classified into correct types.
```

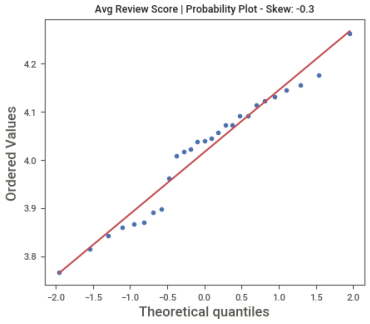
```
<pandas.io.formats.style.Styler at 0x7f63afa6b490>
```

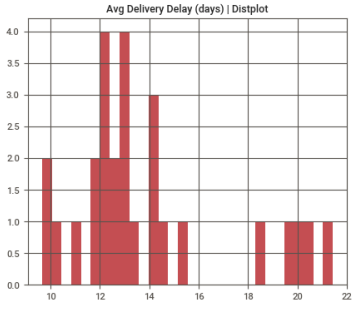
```
Number of All Scatter Plots = 6
```

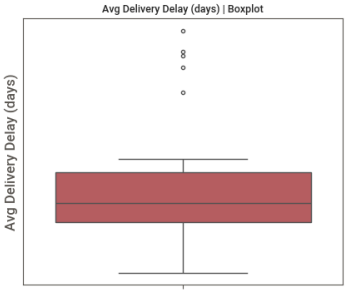

Pair-wise Scatter Plot of all Continuous Variables
Avg Review Score
Customer %
Avg Delivery Delay (days)
Customer %
Avg Delivery Delay (days)
Avg Review Score

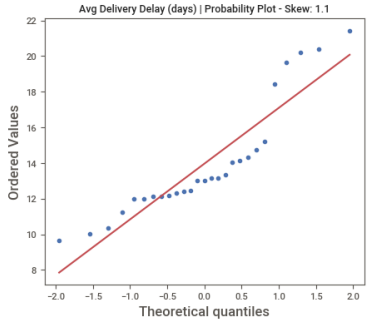

Avg Review Score | Distplot

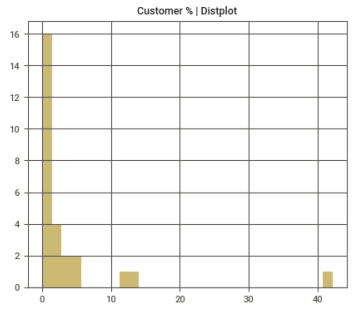

Avg Review Score | Boxplot
Avg Review Score


Avg Review Score | Probability Plot - Skew: -0.3
Ordered Values
Theoretical quantiles

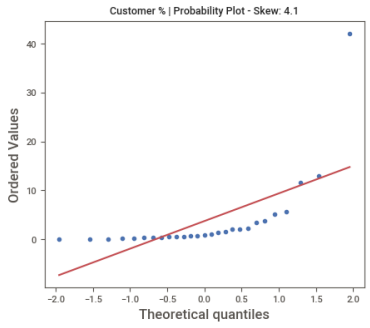

Avg Delivery Delay (days) | Distplot

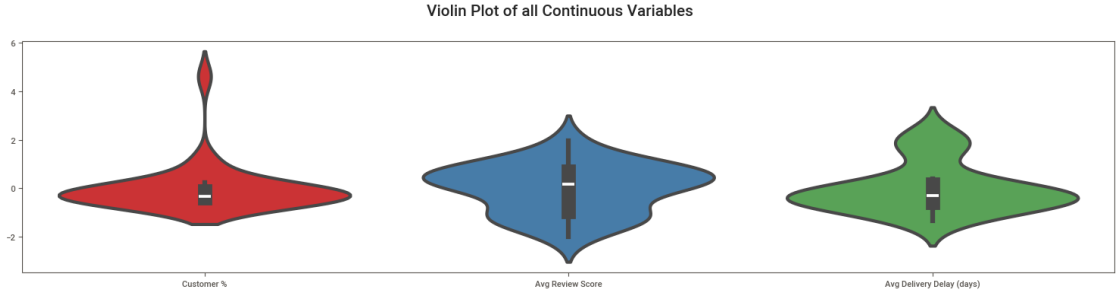

Avg Delivery Delay (days) | Boxplot
Avg Delivery Delay (days)

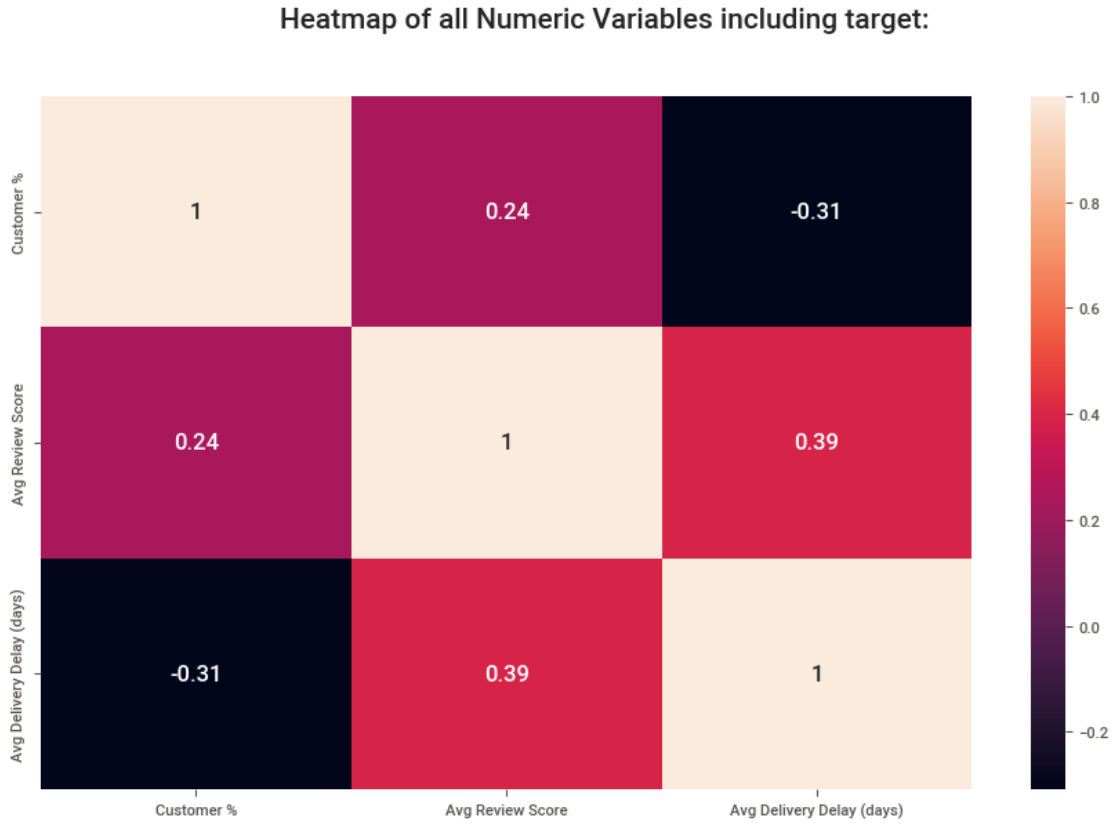

Avg Delivery Delay (days) | Probability Plot - Skew: 1.1
Ordered Values
Theoretical quantiles


Customer % | Distplot


Customer % | Boxplot
Customer %


Customer % | Probability Plot - Skew: 4.1
Ordered Values
Theoretical quantiles


Violin Plot of all Continuous Variables
Customer %
Avg Review Score
Avg Delivery Delay (days)




```
All Plots done
Time to run AutoViz = 2 seconds

 ###################### AUTO VISUALIZATION Completed ########################
```

[28]:
```
states_eda_report = sv.analyze(states_df)
states_eda_report.show_html("states_eda_report.html")
```

```
                                                    |                                                  ␣
↪                  | [  0%]   00:…
Report states_eda_report.html was generated! NOTEBOOK/COLAB USERS: the web
browser MAY not pop up, regardless, the report IS saved in your notebook/colab
files.
```