Table of Contents

F. Romaric Berger

In [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
```

executed in 5.94s, finished 13:24:18 2025-05-18

```
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 ⚙

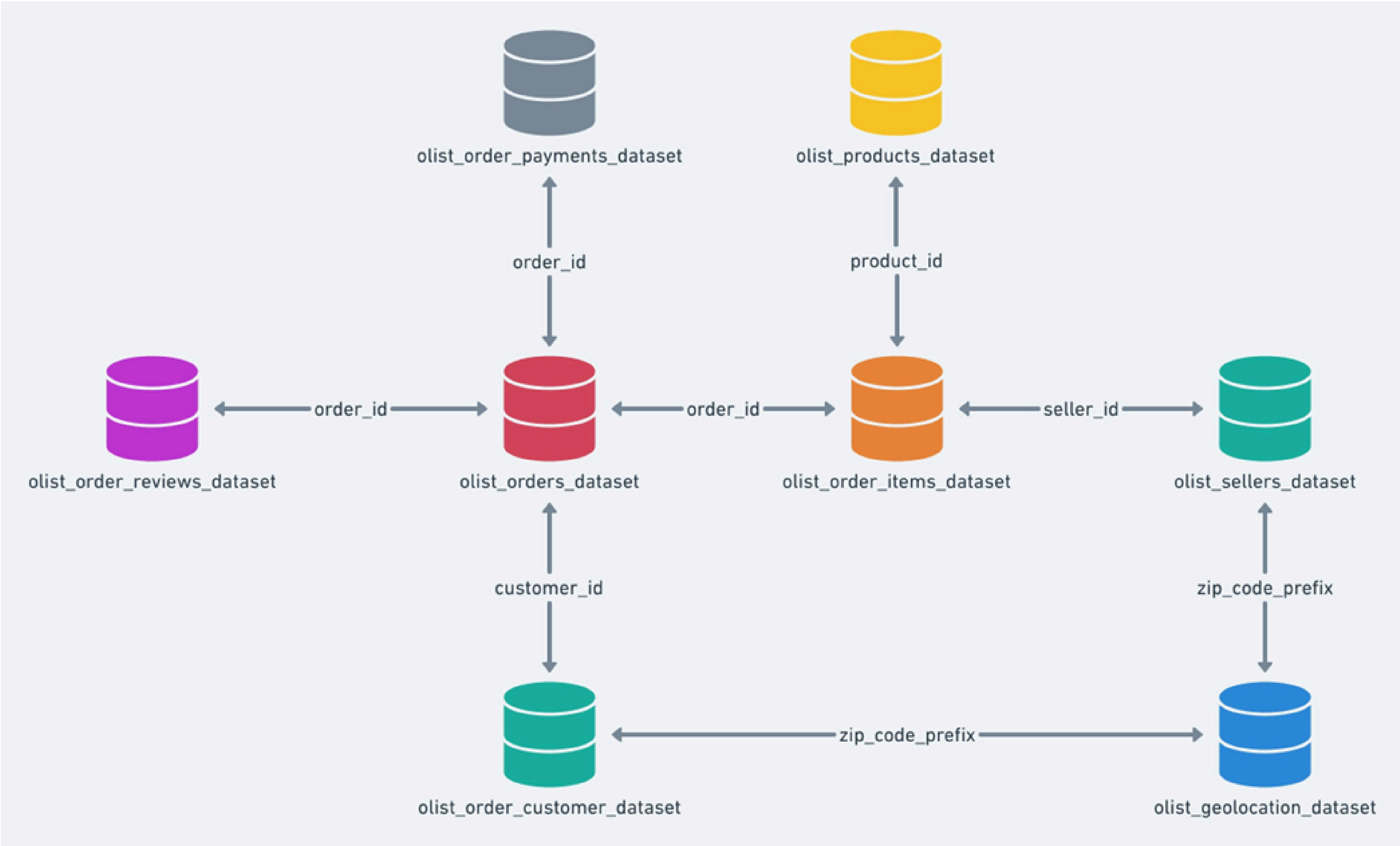
The customer:

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.

In [2]:
```python
Image("Image/olist_erd.png")
```
executed in 119ms, finished 15:42:01 2025-05-09

Out[2]:



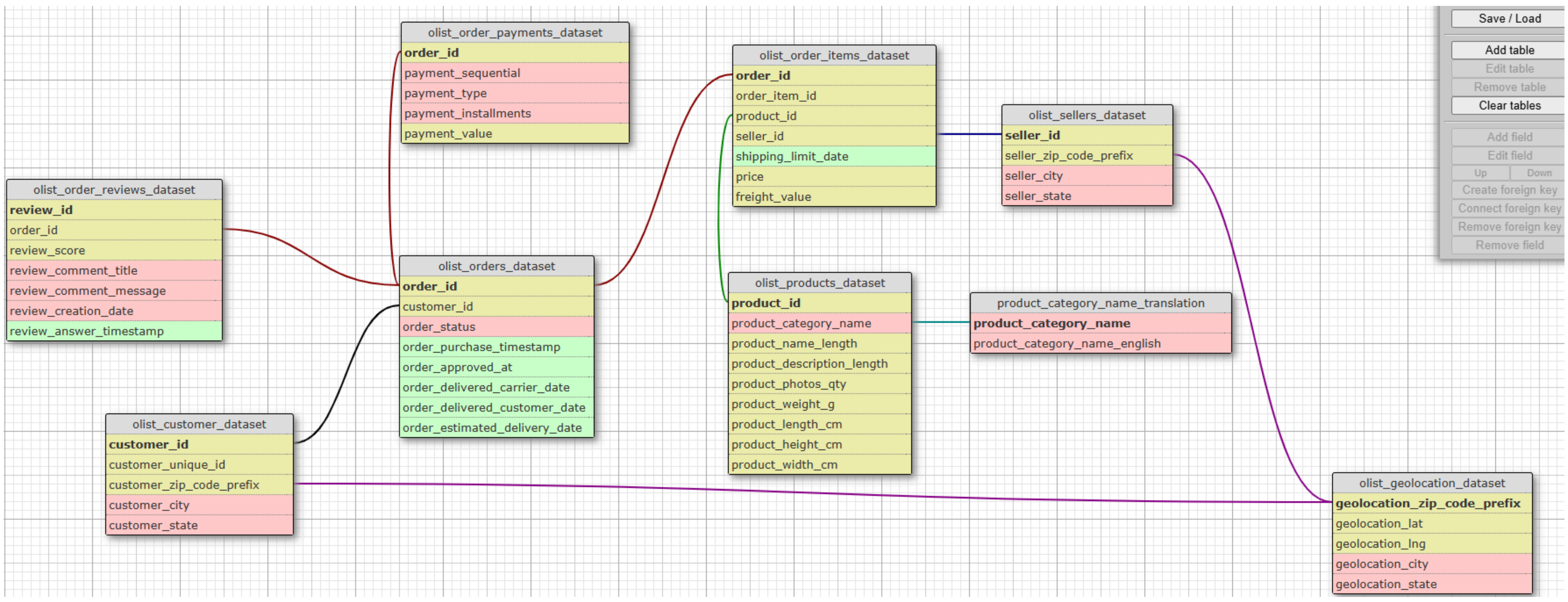
1 Objectives

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

2 Data preparation

In [3]:
```python
Image('Image/olist_erd_details.png')
```
executed in 55ms, finished 15:42:01 2025-05-09

Out[3]:



In [3]:
```python
# 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')
```
executed in 1.26s, finished 13:24:19 2025-05-18

In [4]:
```python
# 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()
```
executed in 930ms, finished 13:24:20 2025-05-18

...

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

In [4]:
```python
data.info()
```
executed in 160ms, finished 15:24:29 2025-05-15

```
<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
```

In [6]:
```python
data.describe()
```
executed in 97ms, finished 13:26:29 2025-05-18

Out[6]:

	customer_zip_code_prefix	order_item_id	price	freight_value	review_score	product_name_lenght	product_description_lenght	product_photos_c
count	114092.000000	113314.000000	113314.000000	113314.000000	113131.000000	111702.000000	111702.000000	111702.0000
mean	35105.227308	1.198528	120.478701	19.979428	4.016998	48.777560	786.899250	2.2069
std	29868.300916	0.707016	183.279678	15.783227	1.400074	10.024616	651.758866	1.7195
min	1003.000000	1.000000	0.850000	0.000000	1.000000	5.000000	4.000000	1.0000
25%	11250.000000	1.000000	39.900000	13.080000	4.000000	42.000000	348.000000	1.0000
50%	24320.000000	1.000000	74.900000	16.260000	5.000000	52.000000	601.000000	1.0000
75%	59022.000000	1.000000	134.900000	21.150000	5.000000	57.000000	985.000000	3.0000
max	99990.000000	21.000000	6735.000000	409.680000	5.000000	76.000000	3992.000000	20.0000

In [6]:
```python
# Looking for columns with too many missing rows.
data.isna().sum()
```
executed in 180ms, finished 15:24:30 2025-05-15

Out[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

In [7]:
```python
# 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'])
```
executed in 173ms, finished 15:24:30 2025-05-15

In [8]:
```python
# 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"]
```
executed in 8ms, finished 15:24:30 2025-05-15

In [9]:
```python
# 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'])
```
executed in 43ms, finished 15:24:30 2025-05-15

In [10]:
```python
data.columns
```
executed in 8ms, finished 15:24:30 2025-05-15

Out[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

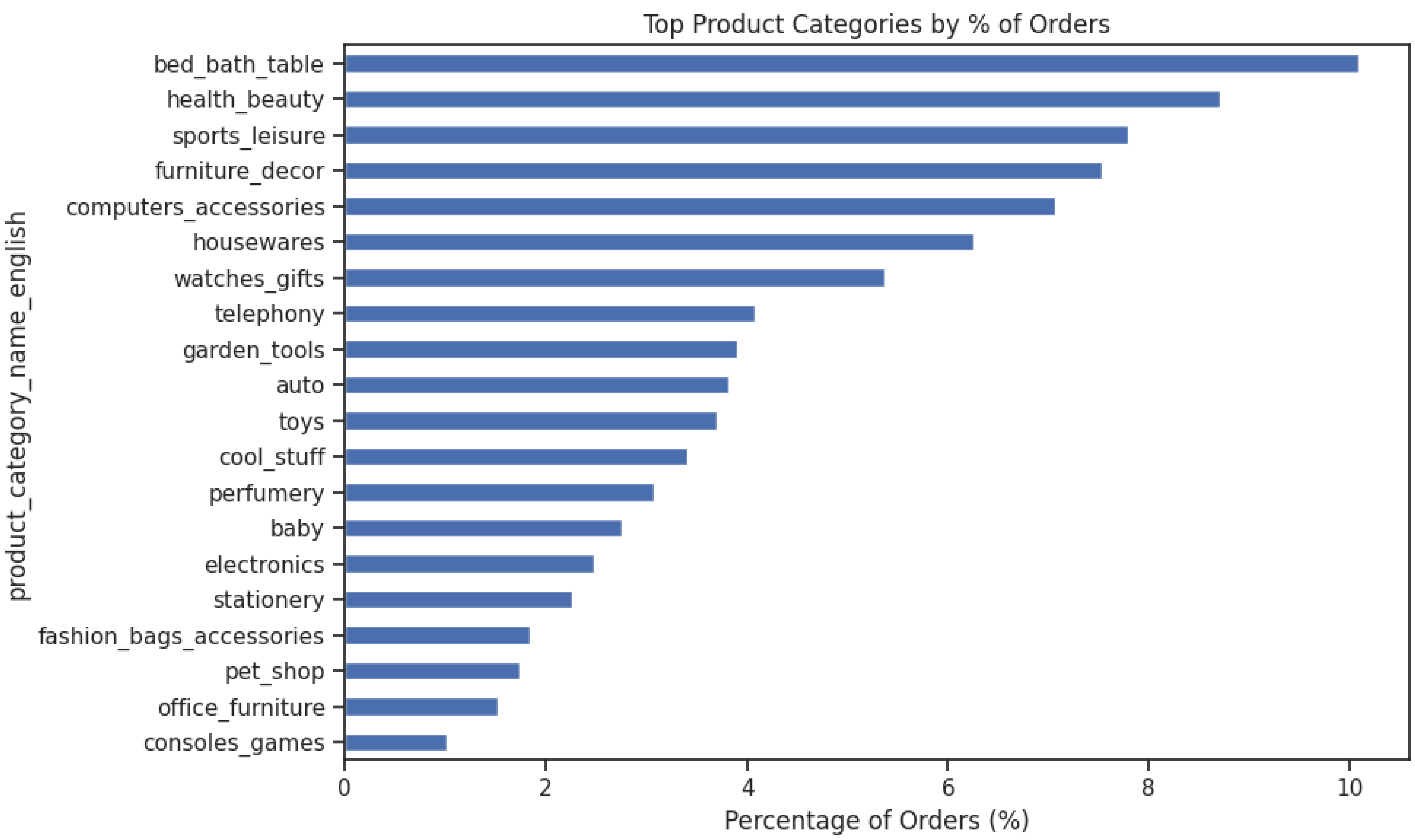
In [10]:
```python
cat_number = data['product_category_name_english'].nunique()
print(f'There is {cat_number} unique category.')
```
executed in 14ms, finished 13:30:02 2025-05-18

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

In [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()
```

executed in 397ms, finished 15:24:31 2025-05-15

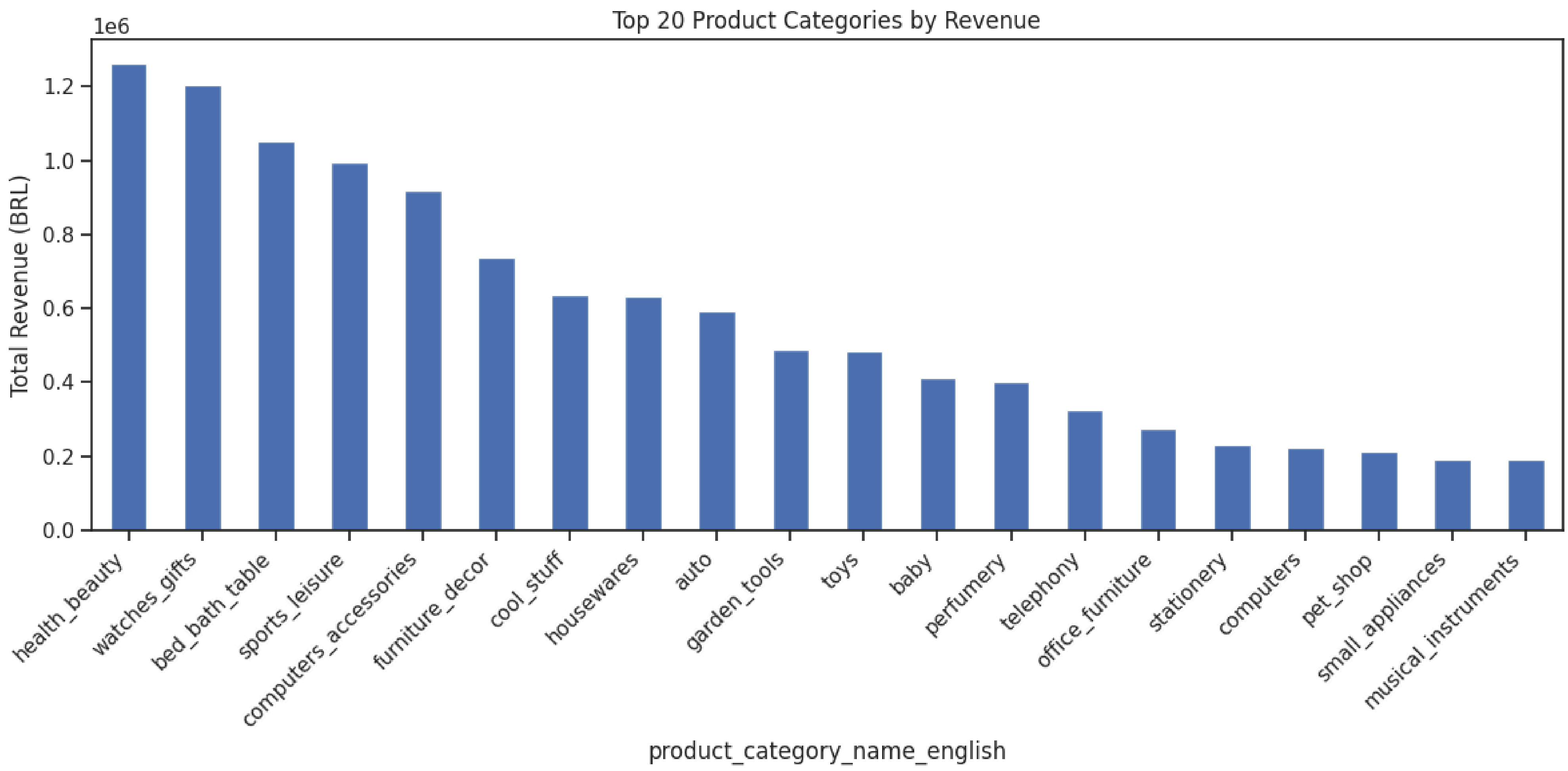


We can see here the categories that are the most sold, but are they the ones that brings in the most revenue?

In [13]:

```python
# 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()
```

executed in 312ms, finished 15:24:33 2025-05-15



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)

In [14]:

```python
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]
})
```

executed in 27ms, finished 15:24:33 2025-05-15

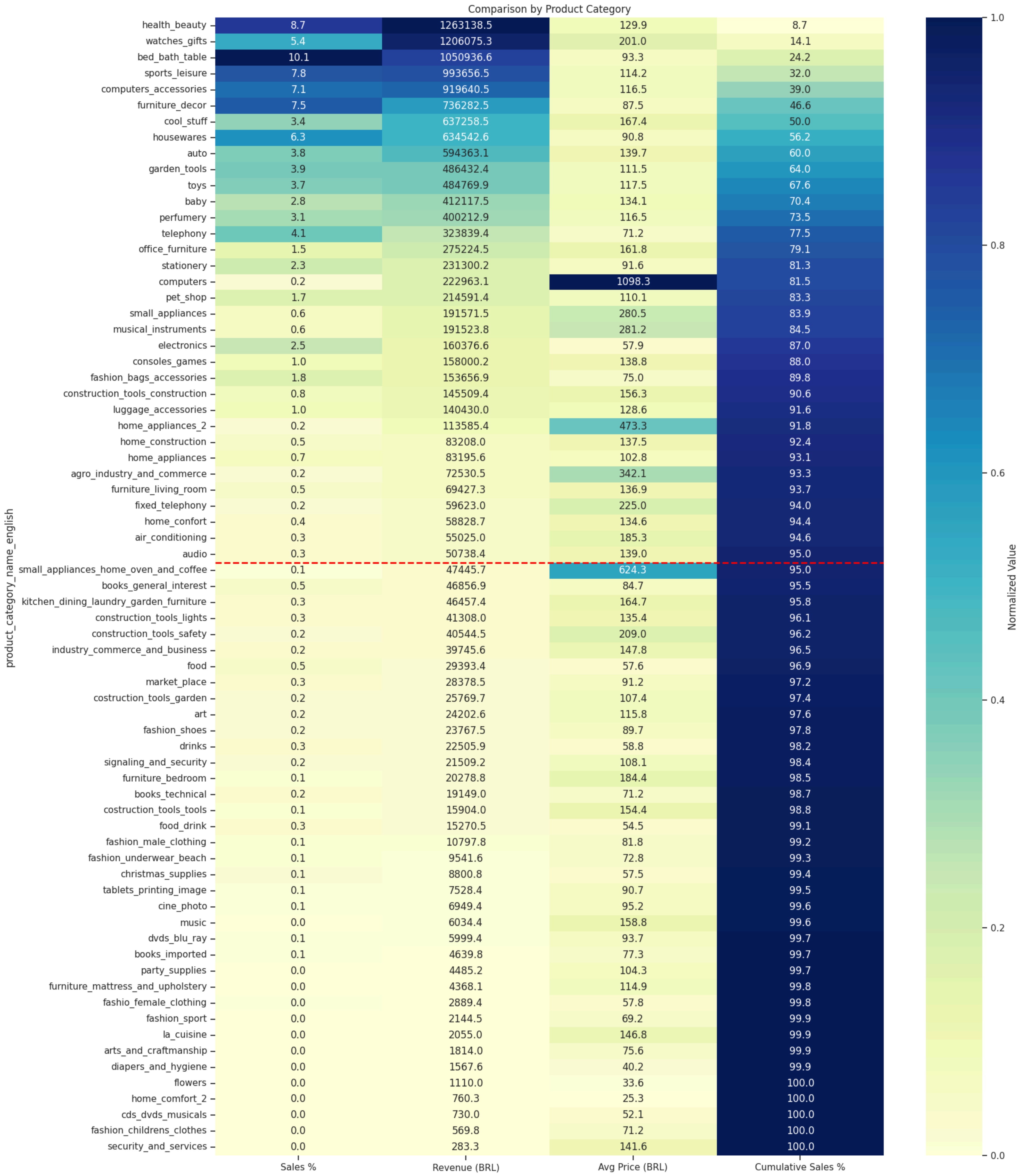
In [15]:

```python
# 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())
```

executed in 14ms, finished 15:24:34 2025-05-15

In [16]:
```python
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()
```

executed in 1.62s, finished 15:24:36 2025-05-15



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:

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

executed in 8ms, finished 15:24:37 2025-05-15

```
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?

In [11]:
```
# How many product do we have?
product_count = data['product_id'].nunique()
print(f'Olist sellers sell {product_count} products.')
```
executed in 33ms, finished 13:32:25 2025-05-18

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

In [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 - 
```
executed in 75ms, finished 15:25:24 2025-05-15

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

In [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}%.')
```
executed in 11ms, finished 15:30:27 2025-05-15

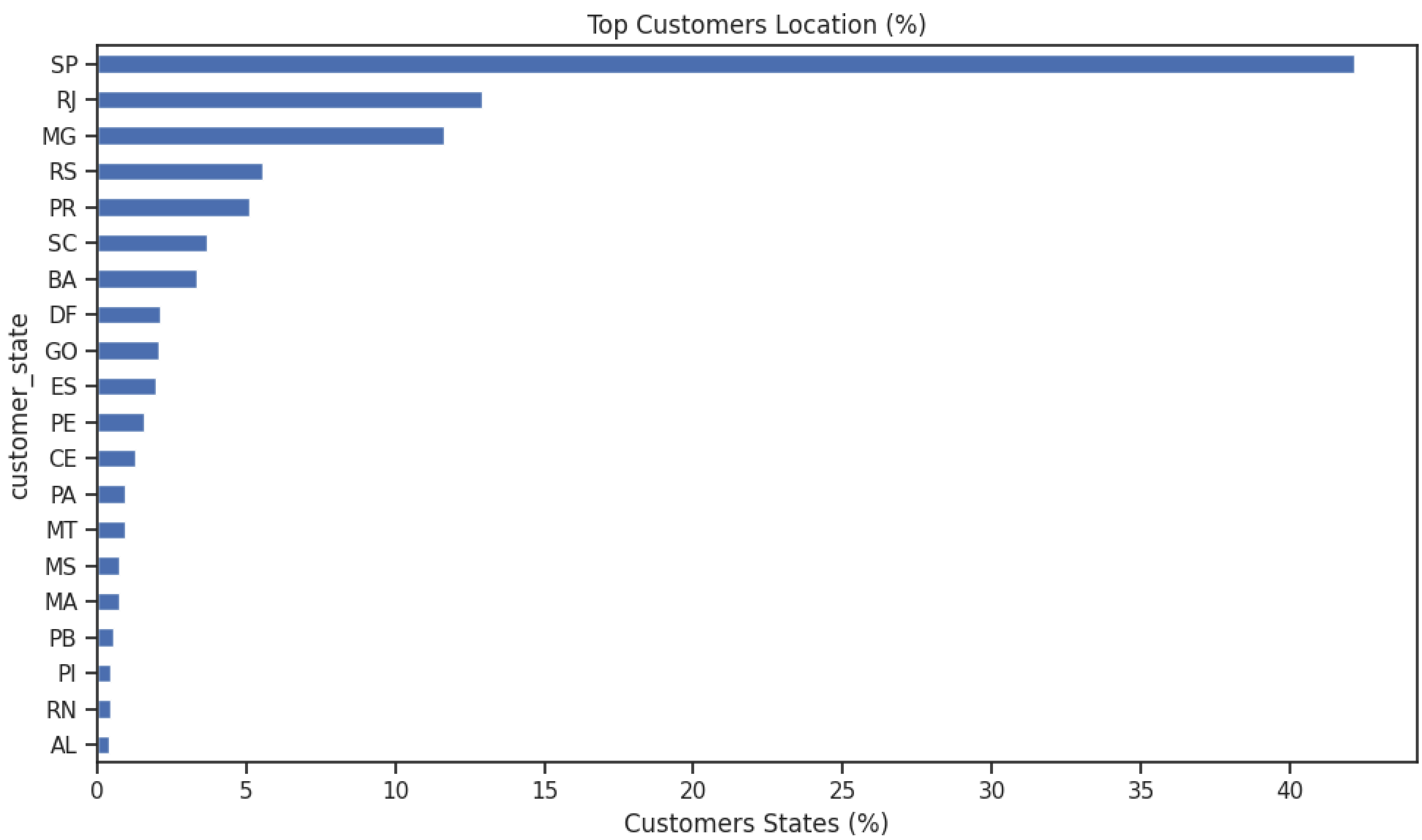
```
The percentage of products not in the top categories is 8.08%.
```

4 Regions_Exploration

In [13]:
```
brl_states = data['customer_state'].nunique()
print(f'Olist operates in {brl_states} states in Brasil.')
```
executed in 20ms, finished 13:33:54 2025-05-18

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

In [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()
```
executed in 328ms, finished 11:29:16 2025-05-11



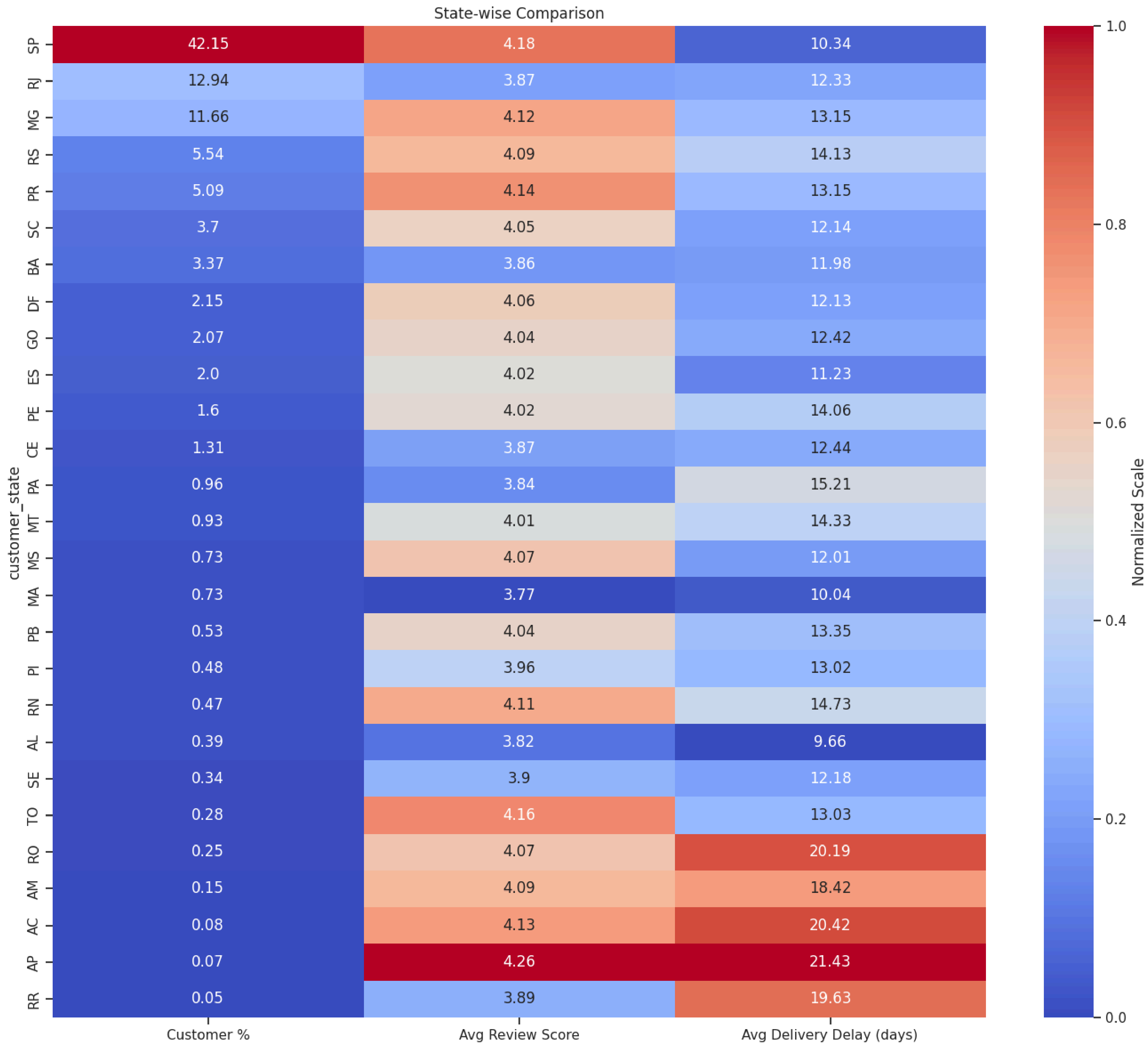
It would be interesting to see if the reviews score is correlated to states, as well as the delay, it could show a distribution problem that has to be fixed.

In [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)
```
executed in 43ms, finished 11:29:18 2025-05-11

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

In [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()
```

executed in 547ms, finished 11:29:20 2025-05-11



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

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

executed in 7ms, finished 11:29:33 2025-05-11

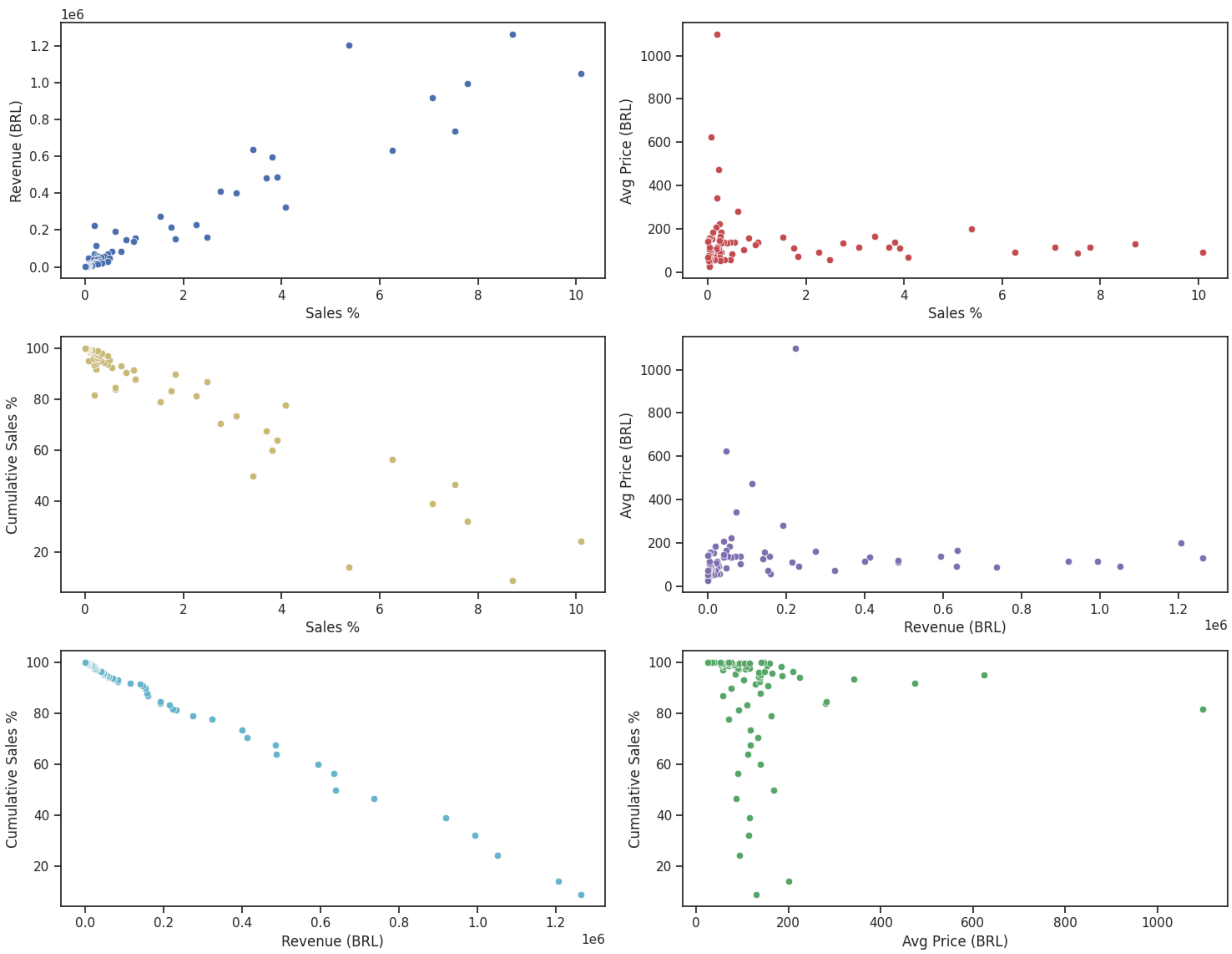
In [24]:
```
report = AV.AutoViz(cat_heatmap_df)
```
executed in 3.21s, finished 11:29:37 2025-05-11

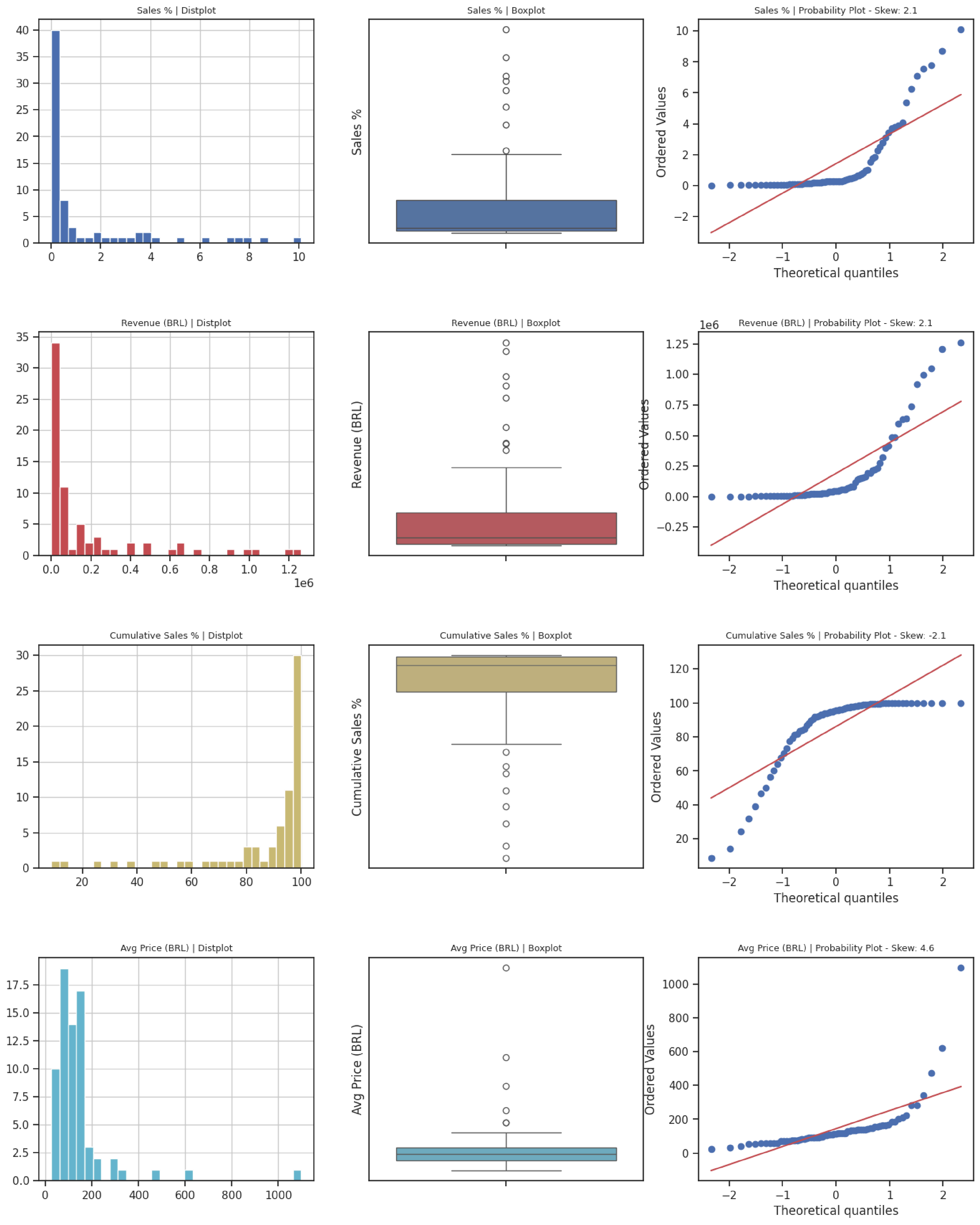
```
Shape of your Data Set loaded: (71, 4)
#######################################################################################
######################## C L A S S I F Y I N G  V A R I A B L E S  ####################
#######################################################################################
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.
```

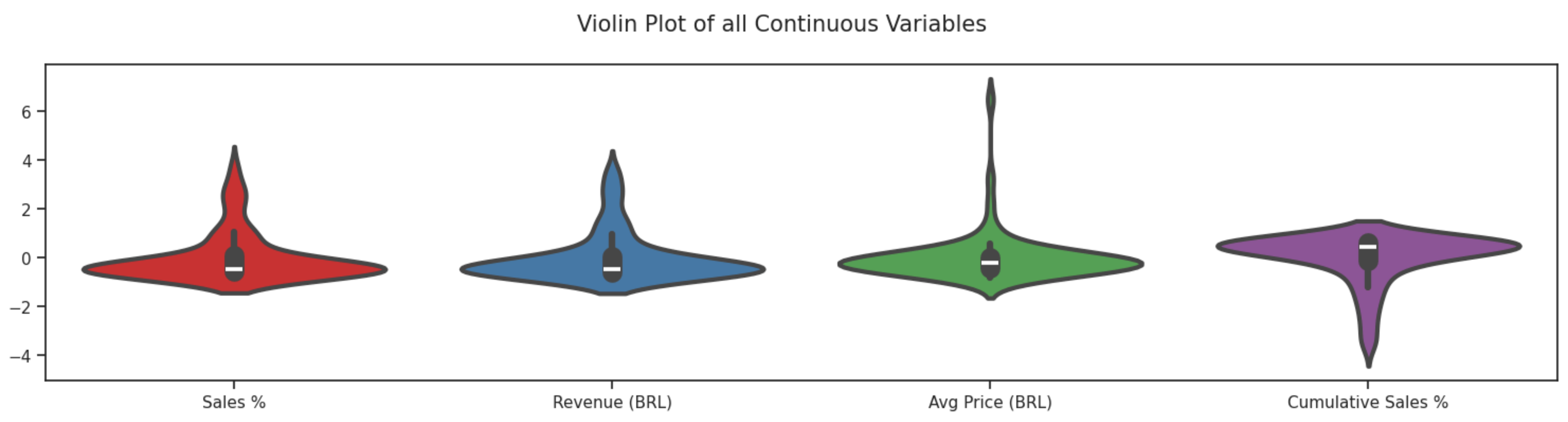
	Data Type	Missing Values%	Unique Values%	Minimum Value	Maximum Value	DQ Issue
Sales %	float64	0.000000	NA	0.001791	10.091513	Column has 8 outliers greater than upper bound (3.96) or lower than lower bound(-2.24). Cap them or remove them.
Revenue (BRL)	float64	0.000000	NA	283.290000	1263138.540000	Column has 9 outliers greater than upper bound (493946.84) or lower than lower bound(-281694.21). Cap them or remove them., Column has a high correlation with ['Sales %']. Consider dropping one of them.
Avg Price (BRL)	float64	0.000000	NA	25.342333	1098.340542	Column has 6 outliers greater than upper bound (248.82) or lower than lower bound(-21.99). Cap them or remove them.
Cumulative Sales %	float64	0.000000	NA	8.709862	100.000000	Column has 8 outliers greater than upper bound (123.04) or lower than lower bound(59.88). Cap them or remove them., Column has a high correlation with ['Sales %', 'Revenue (BRL)']. Consider dropping one of them.

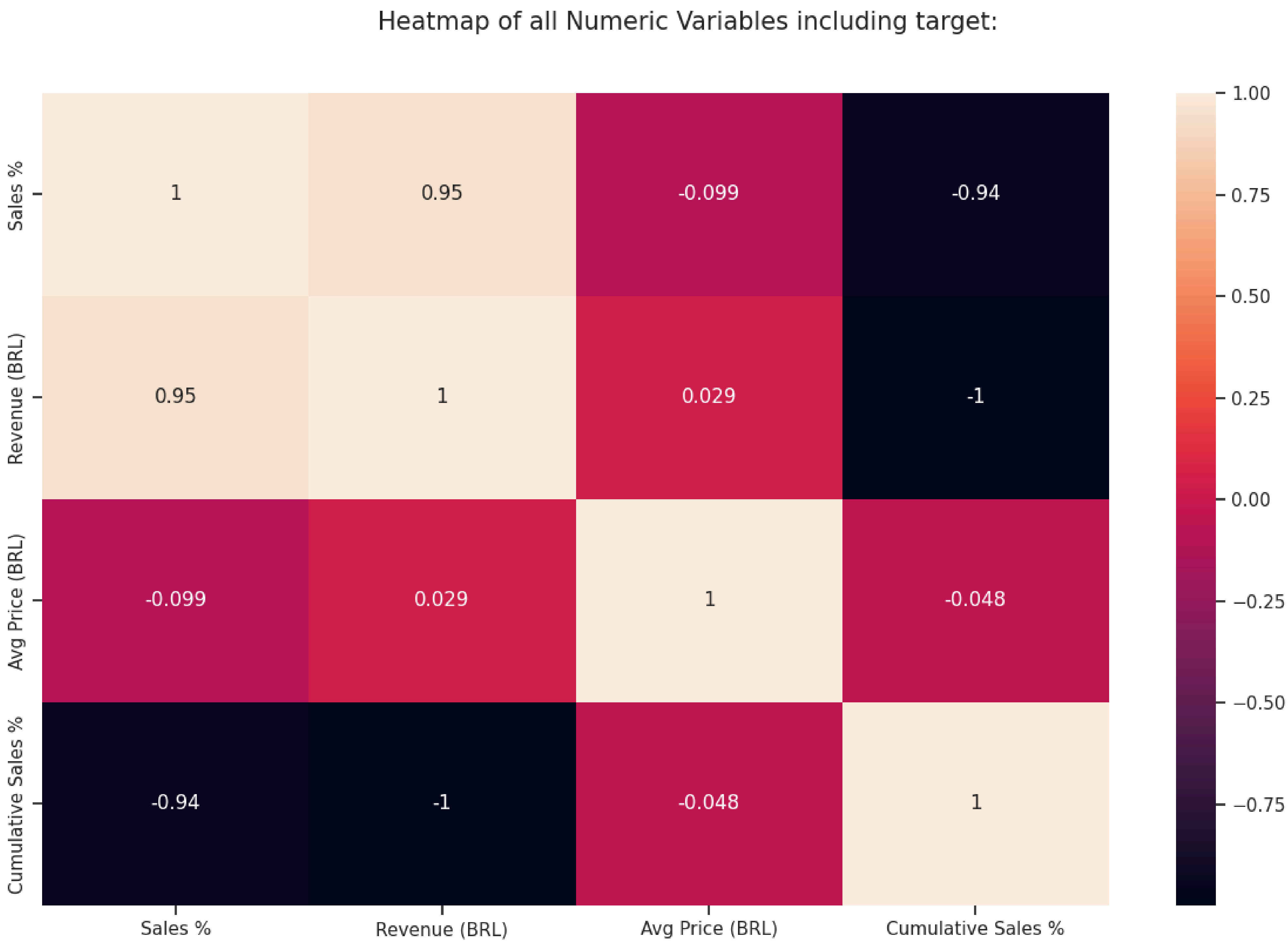
```
Number of All Scatter Plots = 10
```

Pair-wise Scatter Plot of all Continuous Variables




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


Violin Plot of all Continuous Variables
Sales %
Revenue (BRL)
Avg Price (BRL)
Cumulative Sales %




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

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

In [25]:
```
cat_eda_report = sv.analyze(cat_heatmap_df)
```
executed in 2.56s, finished 11:30:54 2025-05-11

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

In [26]:
```
cat_eda_report.show_html("cat_eda_report.html")
```
executed in 1.66s, finished 11:30:56 2025-05-11

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

5.2 States reports

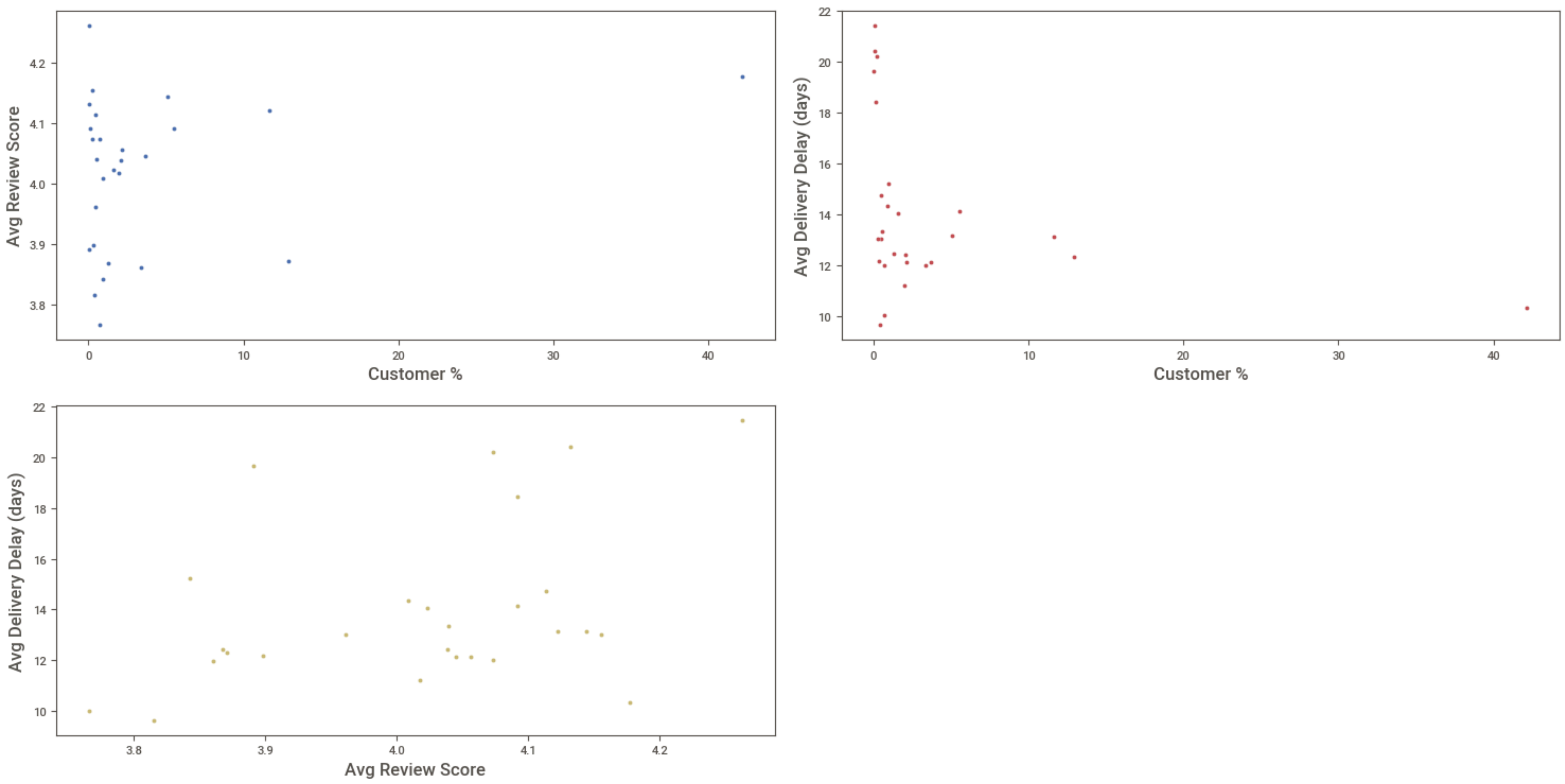
In [27]:
```
states_report =  AV.AutoViz(states_df)
```
executed in 2.41s, finished 11:33:41 2025-05-11

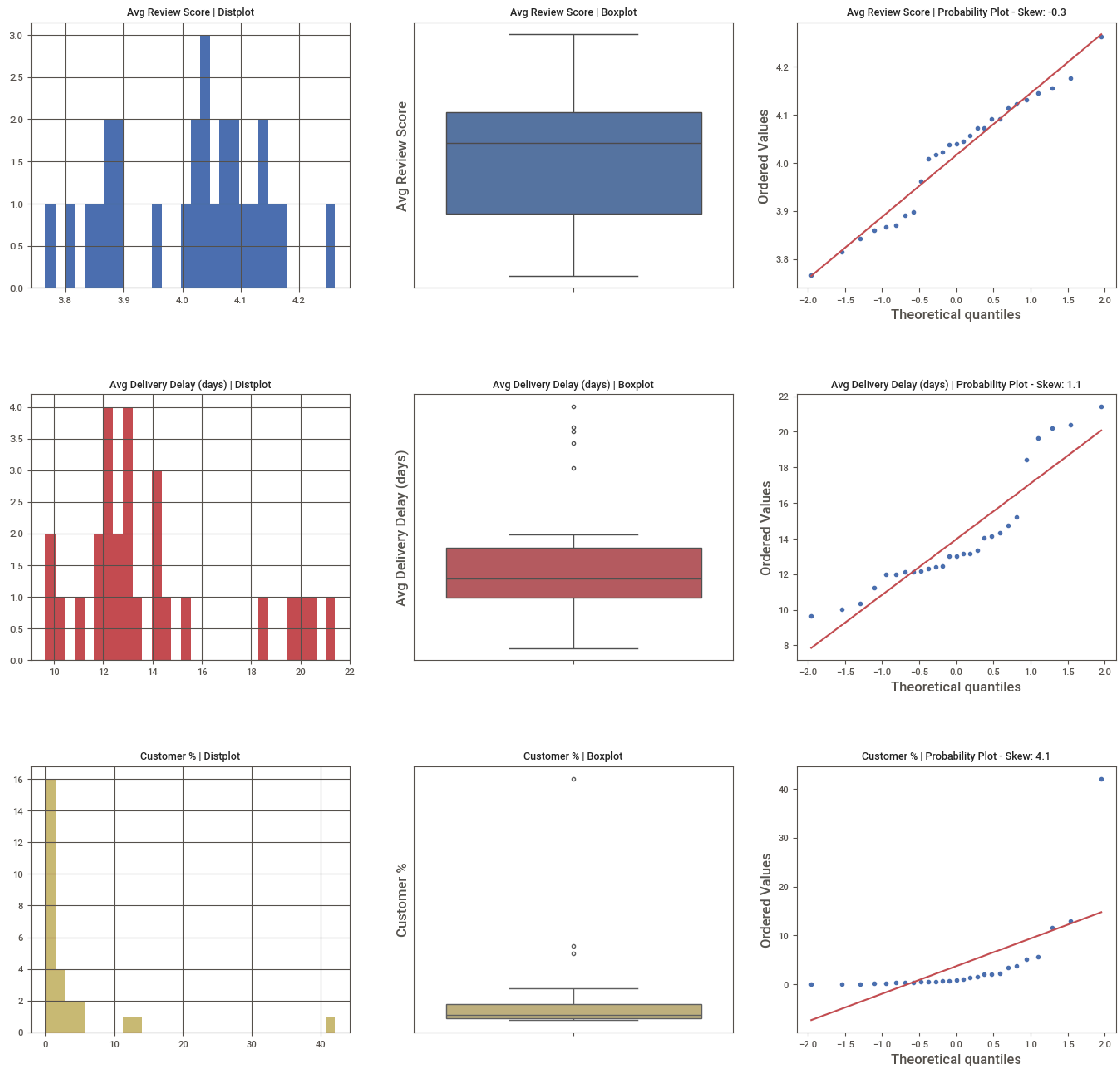
```
Shape of your Data Set loaded: (27, 3)
#######################################################################################
######################## C L A S S I F Y I N G   V A R I A B L E S  ####################
#######################################################################################
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.
```

	Data Type	Missing Values%	Unique Values%	Minimum Value	Maximum Value	DQ Issue
Customer %	float64	0.000000	NA	0.045577	42.152824	Column has 3 outliers greater than upper bound (6.34) or lower than lower bound(-3.21). Cap them or remove them.
Avg Review Score	float64	0.000000	NA	3.765957	4.262500	No issue
Avg Delivery Delay (days)	float64	0.000000	NA	9.661777	21.429097	Column has 5 outliers greater than upper bound (18.13) or lower than lower bound(8.53). Cap them or remove them.

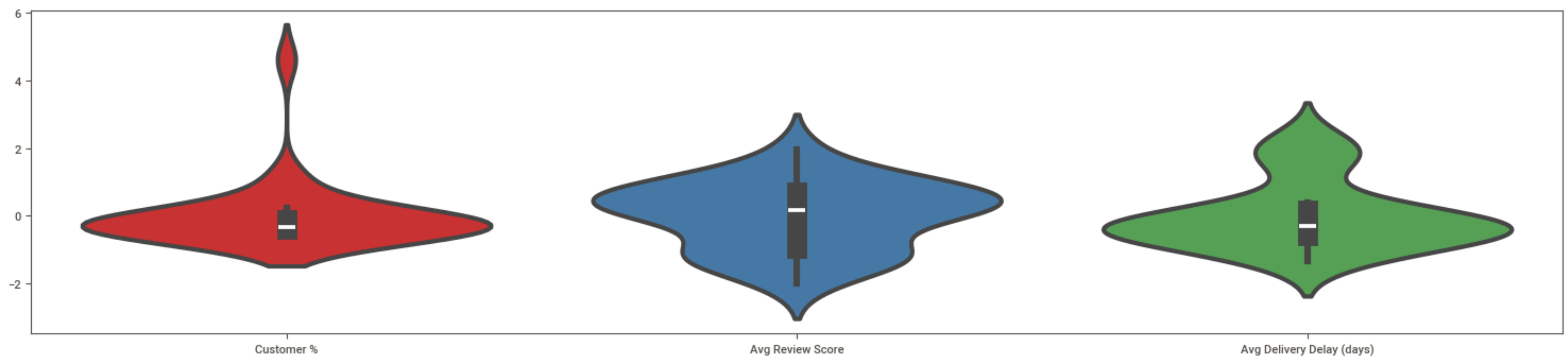
```
Number of All Scatter Plots = 6
```

Pair-wise Scatter Plot of all Continuous Variables

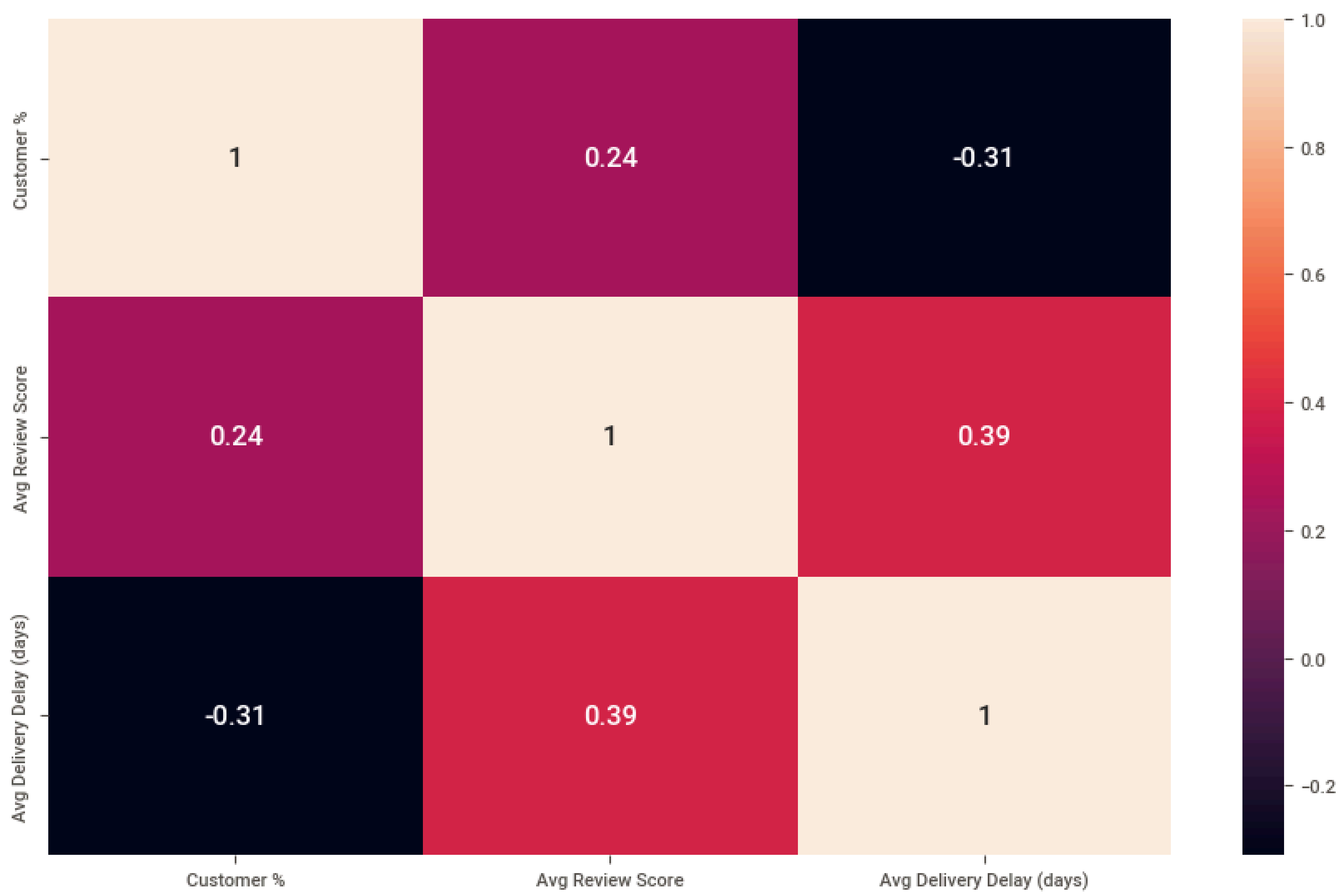




Violin Plot of all Continuous Variables



Heatmap of all Numeric Variables including target:




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

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

In [28]:

```python
states_eda_report = sv.analyze(states_df)
states_eda_report.show_html("states_eda_report.html")
```

executed in 2.08s, finished 11:35:47 2025-05-11

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

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