Imports

Importing all raw datasets and necessary python libraries

In [1]:
```
# vscode import
!pip install folium
```

```
Requirement already satisfied: folium in c:\users\cryp1\anaconda3\lib\site-packages (0.19.6)
Requirement already satisfied: branca>=0.6.0 in c:\users\cryp1\anaconda3\lib\site-packages (fr
om folium) (0.8.1)
Requirement already satisfied: jinja2>=2.9 in c:\users\cryp1\anaconda3\lib\site-packages (from
folium) (3.1.4)
Requirement already satisfied: numpy in c:\users\cryp1\anaconda3\lib\site-packages (from foliu
m) (1.26.4)
Requirement already satisfied: requests in c:\users\cryp1\anaconda3\lib\site-packages (from fo
lium) (2.32.3)
Requirement already satisfied: xyzservices in c:\users\cryp1\anaconda3\lib\site-packages (from
folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\cryp1\anaconda3\lib\site-packages
(from jinja2>=2.9->folium) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\cryp1\anaconda3\lib\site-p
ackages (from requests->folium) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\cryp1\anaconda3\lib\site-packages (fro
m requests->folium) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\cryp1\anaconda3\lib\site-package
s (from requests->folium) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\cryp1\anaconda3\lib\site-package
s (from requests->folium) (2024.8.30)
```

In [2]:
```
# jupyter import
!python3 -m pip install folium
```

```
Python was not found; run without arguments to install from the Microsoft Store, or disable th
is shortcut from Settings > Apps > Advanced app settings > App execution aliases.
```

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import folium
from folium.plugins import HeatMap
from math import radians, sin, cos, sqrt, atan2

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)

# set style
sns.set(style='whitegrid', palette='pastel')
pd.set_option('display.max_colwidth', None)
```

In [4]:
```
# importing and assigning simplified names
try:
    customers = pd.read_csv("data/01_raw/olist_customers_dataset.csv")
    orders = pd.read_csv("data/01_raw/olist_orders_dataset.csv")
    order_items = pd.read_csv("data/01_raw/olist_order_items_dataset.csv")
    payments = pd.read_csv("data/01_raw/olist_order_payments_dataset.csv")
    reviews = pd.read_csv("data/01_raw/olist_order_reviews_dataset.csv")
    products = pd.read_csv("data/01_raw/olist_products_dataset.csv")
    sellers = pd.read_csv("data/01_raw/olist_sellers_dataset.csv")
    geolocation = pd.read_csv("data/01_raw/olist_geolocation_dataset.csv")
    product_translation = pd.read_csv("data/01_raw/product_category_name_translation.csv")
except FileNotFoundError:
    print("Try to change the file directories/names")
```

Initial EDA

Understanding the data cahracteristics within the raw datasets.

In [5]:
```
print("customers:")
print(customers.head().to_string())
print("\n-------------------------\norders:")
print(orders.head().to_string())
print("\n-------------------------\norder_items:")
print(order_items.head().to_string())
print("\n-------------------------\npayments:")
print(payments.head().to_string())
print("\n-------------------------\nreviews:")
print(reviews.head().to_string())
print("\n-------------------------\nproducts:")
print(products.head().to_string())
print("\n-------------------------\nsellers:")
print(sellers.head().to_string())
print("\n-------------------------\ngeolocation:")
print(geolocation.head().to_string())
print("\n-------------------------\nproduct_translation:")
print(product_translation.head().to_string())
```

```
customers:
                        customer_id                customer_unique_id  customer_zip_code_prefi
x          customer_city customer_state
0  06b8999e2fba1a1fbc88172c00ba8bc7  861eff4711a542e4b93843c6dd7febb0                     1440
9                 franca             SP
1  18955e83d337fd6b2def6b18a428ac77  290c77bc529b7ac935b93aa66c333dc3                      979
0  sao bernardo do campo             SP
2  4e7b3e00288586ebd08712fdd0374a03  060e732b5b29e8181a18229c7b0b2b5e                      115
1             sao paulo              SP
3  b2b6027bc5c5109e529d4dc6358b12c3  259dac757896d24d7702b9acbbff3f3c                      877
5       mogi das cruzes              SP
4  4f2d8ab171c80ec8364f7c12e35b23ad  345ecd01c38d18a9036ed96c73b8d066                     1305
6               campinas             SP

-------------------------
orders:
                           order_id                       customer_id order_status order_purch
ase_timestamp    order_approved_at order_delivered_carrier_date order_delivered_customer_date
order_estimated_delivery_date
0  e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d    delivered      2017-1
0-02 10:56:33  2017-10-02 11:07:15          2017-10-04 19:55:00           2017-10-10 21:25:13
2017-10-18 00:00:00
1  53cdb2fc8bc7dce0b6741e2150273451  b0830fb4747a6c6d20dea0b8c802d7ef    delivered      2018-0
7-24 20:41:37  2018-07-26 03:24:27          2018-07-26 14:31:00           2018-08-07 15:27:45
2018-08-13 00:00:00
2  47770eb9100c2d0c44946d9cf07ec65d  41ce2a54c0b03bf3443c3d931a367089    delivered      2018-0
8-08 08:38:49  2018-08-08 08:55:23          2018-08-08 13:50:00           2018-08-17 18:06:29
2018-09-04 00:00:00
3  949d5b44dbf5de918fe9c16f97b45f8a  f88197465ea7920adcdbec7375364d82    delivered      2017-1
1-18 19:28:06  2017-11-18 19:45:59          2017-11-22 13:39:59           2017-12-02 00:28:42
2017-12-15 00:00:00
4  ad21c59c0840e6cb83a9ceb5573f8159  8ab97904e6daea8866dbdbc4fb7aad2c    delivered      2018-0
2-13 21:18:39  2018-02-13 22:20:29          2018-02-14 19:46:34           2018-02-16 18:17:02
2018-02-26 00:00:00

-------------------------
order_items:
                           order_id  order_item_id                        product_id
seller_id  shipping_limit_date   price  freight_value
0  00010242fe8c5a6d1ba2dd792cb16214              1  4244733e06e7ecb4970a6e2683c13e61  48436dad
e18ac8b2bce089ec2a041202  2017-09-19 09:45:35   58.90          13.29
1  00018f77f2f0320c557190d7a144bdd3              1  e5f2d52b802189ee658865ca93d83a8f  dd7ddc04
e1b6c2c614352b383efe2d36  2017-05-03 11:05:13  239.90          19.93
2  000229ec398224ef6ca0657da4fc703e              1  c777355d18b72b67abbeef9df44fd0fd  5b51032e
ddd242adc84c38acab88f23d  2018-01-18 14:48:30  199.00          17.87
3  00024acbcdf0a6daa1e931b038114c75              1  7634da152a4610f1595efa32f14722fc  9d7a1d34
a5052409006425275ba1c2b4  2018-08-15 10:10:18   12.99          12.79
4  00042b26cf59d7ce69dfabb4e55b4fd9              1  ac6c3623068f30de03045865e4e10089  df560393
f3a51e74553ab94004ba5c87  2017-02-13 13:57:51  199.90          18.14

-------------------------
payments:
                           order_id  payment_sequential payment_type  payment_installments  pa
yment_value
0  b81ef226f3fe1789b1e8b2acac839d17                   1  credit_card                     8
99.33
1  a9810da82917af2d9aefd1278f1dcfa0                   1  credit_card                     1
24.39
2  25e8ea4e93396b6fa0d3dd708e76c1bd                   1  credit_card                     1
65.71
3  ba78997921bbcdc1373bb41e913ab953                   1  credit_card                     8
107.78
4  42fdf880ba16b47b59251dd489d4441a                   1  credit_card                     2
128.45

-------------------------
```

reviews:

	review_id	order_id	review_score	review_comment_title	review_comment_message	review_creation_date	review_answer_timestamp
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	NaN	NaN	2018-01-18 00:00:00	2018-01-18 21:46:59
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	NaN	NaN	2018-03-10 00:00:00	2018-03-11 03:05:13
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	NaN	NaN	2018-02-17 00:00:00	2018-02-18 14:36:24
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	NaN	Recebi bem antes do prazo estipulado.	2017-04-21 00:00:00	2017-04-21 22:02:06
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	NaN	Parabéns lojas lannister adorei comprar pela Internet seguro e prático Parabéns a todos feliz Páscoa	2018-03-01 00:00:00	2018-03-02 10:26:53

---

products:

	product_id	product_category_name	product_name_lenght	product_description_lenght	product_photos_qty	product_weight_g	product_length_cm	product_height_cm	product_width_cm
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	287.0	1.0	225.0	16.0	10.0	14.0
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	276.0	1.0	1000.0	30.0	18.0	20.0
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	250.0	1.0	154.0	18.0	9.0	15.0
3	cef67bcfe19066a932b7673e239eb23d	bebes	27.0	261.0	1.0	371.0	26.0	4.0	26.0
4	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	37.0	402.0	4.0	625.0	20.0	17.0	13.0

---

sellers:

	seller_id	seller_zip_code_prefix	seller_city	seller_state
0	3442f8959a84dea7ee197c632cb2df15	13023	campinas	SP
1	d1b65fc7debc3361ea86b5f14c68d2e2	13844	mogi guacu	SP
2	ce3ad9de960102d0677a81f5d0bb7b2d	20031	rio de janeiro	RJ
3	c0f3eea2e14555b6faeea3dd58c1b1c3	4195	sao paulo	SP
4	51a04a8a6bdcb23deccc82b0b80742cf	12914	braganca paulista	SP

---

geolocation:

	geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
0	1037	-23.545621	-46.639292	sao paulo	SP
1	1046	-23.546081	-46.644820	sao paulo	SP
2	1046	-23.546129	-46.642951	sao paulo	SP
3	1041	-23.544392	-46.639499	sao paulo	SP
4	1035	-23.541578	-46.641607	sao paulo	SP

```
product_translation:
    product_category_name product_category_name_english
0            beleza_saude                  health_beauty
1  informatica_acessorios          computers_accessories
2              automotivo                          auto
3         cama_mesa_banho                bed_bath_table
4        moveis_decoracao                furniture_decor
```

In [6]:

```
# get dtypes
print("customers:")
print(customers.info())
print("\n------------------------\norders:")
print(orders.info())
print("\n------------------------\norder_items:")
print(order_items.info())
print("\n------------------------\npayments:")
print(payments.info())
print("\n------------------------\nreviews:")
print(reviews.info())
print("\n------------------------\nproducts:")
print(products.info())
print("\n------------------------\nsellers:")
print(sellers.info())
print("\n------------------------\ngeolocation:")
print(geolocation.info())
print("\n------------------------\nproduct_translation:")
print(product_translation.info())
```

```
customers:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 5 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   customer_id               99441 non-null  object
 1   customer_unique_id        99441 non-null  object
 2   customer_zip_code_prefix  99441 non-null  int64 
 3   customer_city             99441 non-null  object
 4   customer_state            99441 non-null  object
dtypes: int64(1), object(4)
memory usage: 3.8+ MB
None

-------------------------
orders:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   order_id                       99441 non-null  object
 1   customer_id                    99441 non-null  object
 2   order_status                   99441 non-null  object
 3   order_purchase_timestamp       99441 non-null  object
 4   order_approved_at              99281 non-null  object
 5   order_delivered_carrier_date   97658 non-null  object
 6   order_delivered_customer_date  96476 non-null  object
 7   order_estimated_delivery_date  99441 non-null  object
dtypes: object(8)
memory usage: 6.1+ MB
None

-------------------------
order_items:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112650 entries, 0 to 112649
Data columns (total 7 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   order_id             112650 non-null  object 
 1   order_item_id        112650 non-null  int64  
 2   product_id           112650 non-null  object 
 3   seller_id            112650 non-null  object 
 4   shipping_limit_date  112650 non-null  object 
 5   price                112650 non-null  float64
 6   freight_value        112650 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 6.0+ MB
None

-------------------------
payments:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103886 entries, 0 to 103885
Data columns (total 5 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   order_id              103886 non-null  object 
 1   payment_sequential    103886 non-null  int64  
 2   payment_type          103886 non-null  object 
 3   payment_installments  103886 non-null  int64  
 4   payment_value         103886 non-null  float64
dtypes: float64(1), int64(2), object(2)
memory usage: 4.0+ MB
```

```
None

-------------------------
reviews:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99224 entries, 0 to 99223
Data columns (total 7 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   review_id                99224 non-null  object
 1   order_id                 99224 non-null  object
 2   review_score             99224 non-null  int64 
 3   review_comment_title     11568 non-null  object
 4   review_comment_message   40977 non-null  object
 5   review_creation_date     99224 non-null  object
 6   review_answer_timestamp  99224 non-null  object
dtypes: int64(1), object(6)
memory usage: 5.3+ MB
None

-------------------------
products:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32951 entries, 0 to 32950
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   product_id                  32951 non-null  object 
 1   product_category_name       32341 non-null  object 
 2   product_name_lenght         32341 non-null  float64
 3   product_description_lenght  32341 non-null  float64
 4   product_photos_qty          32341 non-null  float64
 5   product_weight_g            32949 non-null  float64
 6   product_length_cm           32949 non-null  float64
 7   product_height_cm           32949 non-null  float64
 8   product_width_cm            32949 non-null  float64
dtypes: float64(7), object(2)
memory usage: 2.3+ MB
None

-------------------------
sellers:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3095 entries, 0 to 3094
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   seller_id               3095 non-null   object
 1   seller_zip_code_prefix  3095 non-null   int64 
 2   seller_city             3095 non-null   object
 3   seller_state            3095 non-null   object
dtypes: int64(1), object(3)
memory usage: 96.8+ KB
None

-------------------------
geolocation:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000163 entries, 0 to 1000162
Data columns (total 5 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   geolocation_zip_code_prefix  1000163 non-null  int64  
 1   geolocation_lat              1000163 non-null  float64
 2   geolocation_lng              1000163 non-null  float64
 3   geolocation_city             1000163 non-null  object 
```

```
 4   geolocation_state            1000163 non-null  object
dtypes: float64(2), int64(1), object(2)
memory usage: 38.2+ MB
None

-------------------------
product_translation:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 71 entries, 0 to 70
Data columns (total 2 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   product_category_name          71 non-null     object
 1   product_category_name_english  71 non-null     object
dtypes: object(2)
memory usage: 1.2+ KB
None
```

In [7]:

```python
# num cols info
print("customers:")
print(customers.describe().to_string())
print("\n-------------------------\norders:")
print(orders.describe().to_string())
print("\n-------------------------\norder_items:")
print(order_items.describe().to_string())
print("\n-------------------------\npayments:")
print(payments.describe().to_string())
print("\n-------------------------\nreviews:")
print(reviews.describe().to_string())
print("\n-------------------------\nproducts:")
print(products.describe().to_string())
print("\n-------------------------\nsellers:")
print(sellers.describe().to_string())
print("\n-------------------------\ngeolocation:")
print(geolocation.describe().to_string())
print("\n-------------------------\nproduct_translation:")
print(product_translation.describe().to_string())
```

```
customers:
       customer_zip_code_prefix
count              99441.000000
mean               35137.474583
std                29797.938996
min                 1003.000000
25%                11347.000000
50%                24416.000000
75%                58900.000000
max                99990.000000

-------------------------
orders:
                               order_id                         customer_id order_status order_
purchase_timestamp    order_approved_at order_delivered_carrier_date order_delivered_customer_
date order_estimated_delivery_date
count                           99441                             99441        99441
99441               99281                         97658                         96476
99441
unique                          99441                             99441            8
98875               90733                         81018                         95664
459
top     e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d    delivered      2
018-04-11 10:48:14  2018-02-27 04:31:10          2018-05-09 15:48:00           2018-05-08 23:3
8:46           2017-12-20 00:00:00
freq                                1                                 1        96478
3                   9                           47                            3
522

-------------------------
order_items:
       order_item_id          price  freight_value
count  112650.000000  112650.000000  112650.000000
mean        1.197834     120.653739      19.990320
std         0.705124     183.633928      15.806405
min         1.000000       0.850000       0.000000
25%         1.000000      39.900000      13.080000
50%         1.000000      74.990000      16.260000
75%         1.000000     134.900000      21.150000
max        21.000000    6735.000000     409.680000

-------------------------
payments:
       payment_sequential  payment_installments  payment_value
count       103886.000000         103886.000000  103886.000000
mean             1.092679              2.853349     154.100380
std              0.706584              2.687051     217.494064
min              1.000000              0.000000       0.000000
25%              1.000000              1.000000      56.790000
50%              1.000000              1.000000     100.000000
75%              1.000000              4.000000     171.837500
max             29.000000             24.000000   13664.080000

-------------------------
reviews:
       review_score
count  99224.000000
mean       4.086421
std        1.347579
min        1.000000
25%        4.000000
50%        5.000000
75%        5.000000
max        5.000000

-------------------------
```

```
products:
       product_name_lenght  product_description_lenght  product_photos_qty  product_weight_g
product_length_cm  product_height_cm  product_width_cm
count         32341.000000                32341.000000        32341.000000      32949.000000
32949.000000       32949.000000      32949.000000
mean             48.476949                  771.495285            2.188986       2276.472488
30.815078          16.937661         23.196728
std              10.245741                  635.115225            1.736766       4282.038731
16.914458          13.637554         12.079047
min               5.000000                    4.000000            1.000000          0.000000
7.000000           2.000000          6.000000
25%              42.000000                  339.000000            1.000000        300.000000
18.000000           8.000000         15.000000
50%              51.000000                  595.000000            1.000000        700.000000
25.000000          13.000000         20.000000
75%              57.000000                  972.000000            3.000000       1900.000000
38.000000          21.000000         30.000000
max              76.000000                 3992.000000           20.000000      40425.000000
105.000000         105.000000        118.000000

-------------------------
sellers:
       seller_zip_code_prefix
count             3095.000000
mean             32291.059451
std              32713.453830
min               1001.000000
25%               7093.500000
50%              14940.000000
75%              64552.500000
max              99730.000000

-------------------------
geolocation:
       geolocation_zip_code_prefix  geolocation_lat  geolocation_lng
count                 1.000163e+06     1.000163e+06     1.000163e+06
mean                  3.657417e+04    -2.117615e+01    -4.639054e+01
std                   3.054934e+04     5.715866e+00     4.269748e+00
min                   1.001000e+03    -3.660537e+01    -1.014668e+02
25%                   1.107500e+04    -2.360355e+01    -4.857317e+01
50%                   2.653000e+04    -2.291938e+01    -4.663788e+01
75%                   6.350400e+04    -1.997962e+01    -4.376771e+01
max                   9.999000e+04     4.506593e+01     1.211054e+02

-------------------------
product_translation:
       product_category_name product_category_name_english
count                     71                           71
unique                    71                           71
top             beleza_saude                health_beauty
freq                       1                            1
```

In [8]:

```python
# full duplicate rows
for name, df_ in zip(
    ['customers', 'orders', 'order_items', 'payments', 'reviews', 'products', 'sellers', 'geol
    [customers, orders, order_items, payments, reviews, products, sellers, geolocation]):
    print(f"\n{df_.shape[0]}")
    print(f"{name}: {df_.value_counts().sum()} unique rows")
    print(f"{name}: {df_.duplicated().sum()} duplicate rows")
    print(f"{name} null values:")
    print(df_.isnull().sum())


# check duplicate IDs
print(f"\nUnique ids {customers['customer_unique_id'].duplicated().sum()}")
print(f"ids: {customers['customer_id'].duplicated().sum()}")
```

```
print(f"order_ids: {orders['order_id'].duplicated().sum()}")         # Should be 0
print(f"product_ids: {products['product_id'].duplicated().sum()}")   # Should be 0
print(f"seller_ids: {sellers['seller_id'].duplicated().sum()}")      # Should be 0
```

```
99441
customers: 99441 unique rows
customers: 0 duplicate rows
customers null values:
customer_id                 0
customer_unique_id          0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64

99441
orders: 96461 unique rows
orders: 0 duplicate rows
orders null values:
order_id                            0
customer_id                         0
order_status                        0
order_purchase_timestamp            0
order_approved_at                 160
order_delivered_carrier_date     1783
order_delivered_customer_date    2965
order_estimated_delivery_date       0
dtype: int64

112650
order_items: 112650 unique rows
order_items: 0 duplicate rows
order_items null values:
order_id               0
order_item_id          0
product_id             0
seller_id              0
shipping_limit_date    0
price                  0
freight_value          0
dtype: int64

103886
payments: 103886 unique rows
payments: 0 duplicate rows
payments null values:
order_id                0
payment_sequential      0
payment_type            0
payment_installments    0
payment_value           0
dtype: int64

99224
reviews: 9839 unique rows
reviews: 0 duplicate rows
reviews null values:
review_id                       0
order_id                        0
review_score                    0
review_comment_title        87656
review_comment_message      58247
review_creation_date            0
review_answer_timestamp         0
dtype: int64

32951
products: 32340 unique rows
products: 0 duplicate rows
products null values:
```

```
product_id                      0
product_category_name         610
product_name_lenght           610
product_description_lenght    610
product_photos_qty            610
product_weight_g                2
product_length_cm               2
product_height_cm               2
product_width_cm                2
dtype: int64

3095
sellers: 3095 unique rows
sellers: 0 duplicate rows
sellers null values:
seller_id                 0
seller_zip_code_prefix    0
seller_city               0
seller_state              0
dtype: int64

1000163
geolocation: 1000163 unique rows
geolocation: 261831 duplicate rows
geolocation null values:
geolocation_zip_code_prefix    0
geolocation_lat                0
geolocation_lng                0
geolocation_city               0
geolocation_state              0
dtype: int64

Unique ids 3345
ids: 0
order_ids: 0
product_ids: 0
seller_ids: 0
```

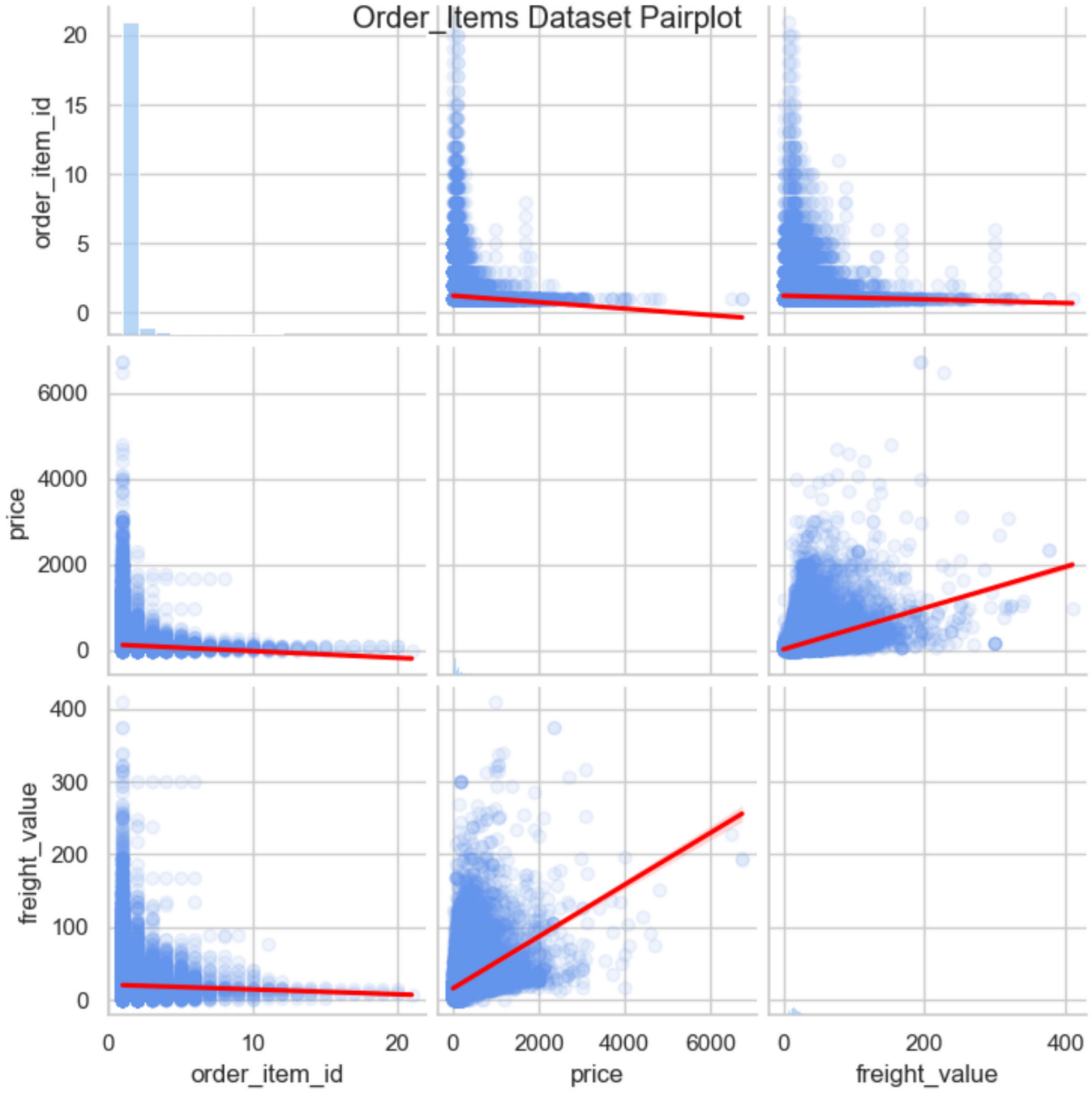
Dataset Pairplots

Visualize and explore potential relationships and patterns between numerical features within raw datasets. For general EDA and understanding of data. May be referred during later stages of EDA or model building.

In [9]:
```python
def pair_plotter(df: pd.DataFrame, title: str, figsize: tuple=None):
    """ creates pairplots of datasets

    Parameters:
        figsize: plot size
        title: name of dataset
        dataset: dataframe/dataset
    Returns:
        dataset pairplot
    """
    plt.figure(figsize=figsize)
    sns.pairplot(df, kind='reg', plot_kws={'line_kws':{'color':'red'}, 'scatter_kws': {'alpha
    plt.show()
```

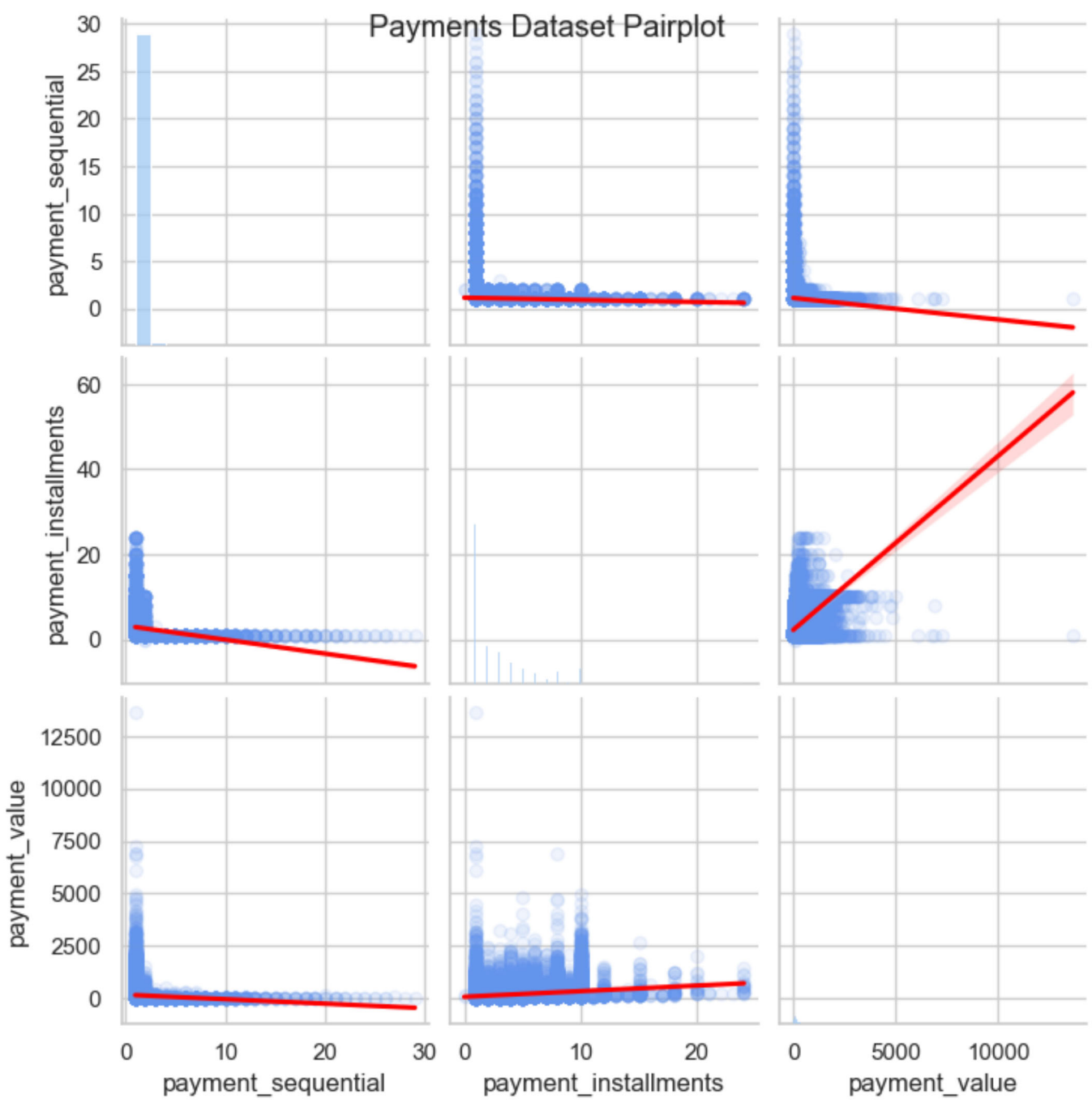
In [10]:
```python
pair_plotter(order_items, "Order_Items", (8, 6))
```

```
<Figure size 800x600 with 0 Axes>
```



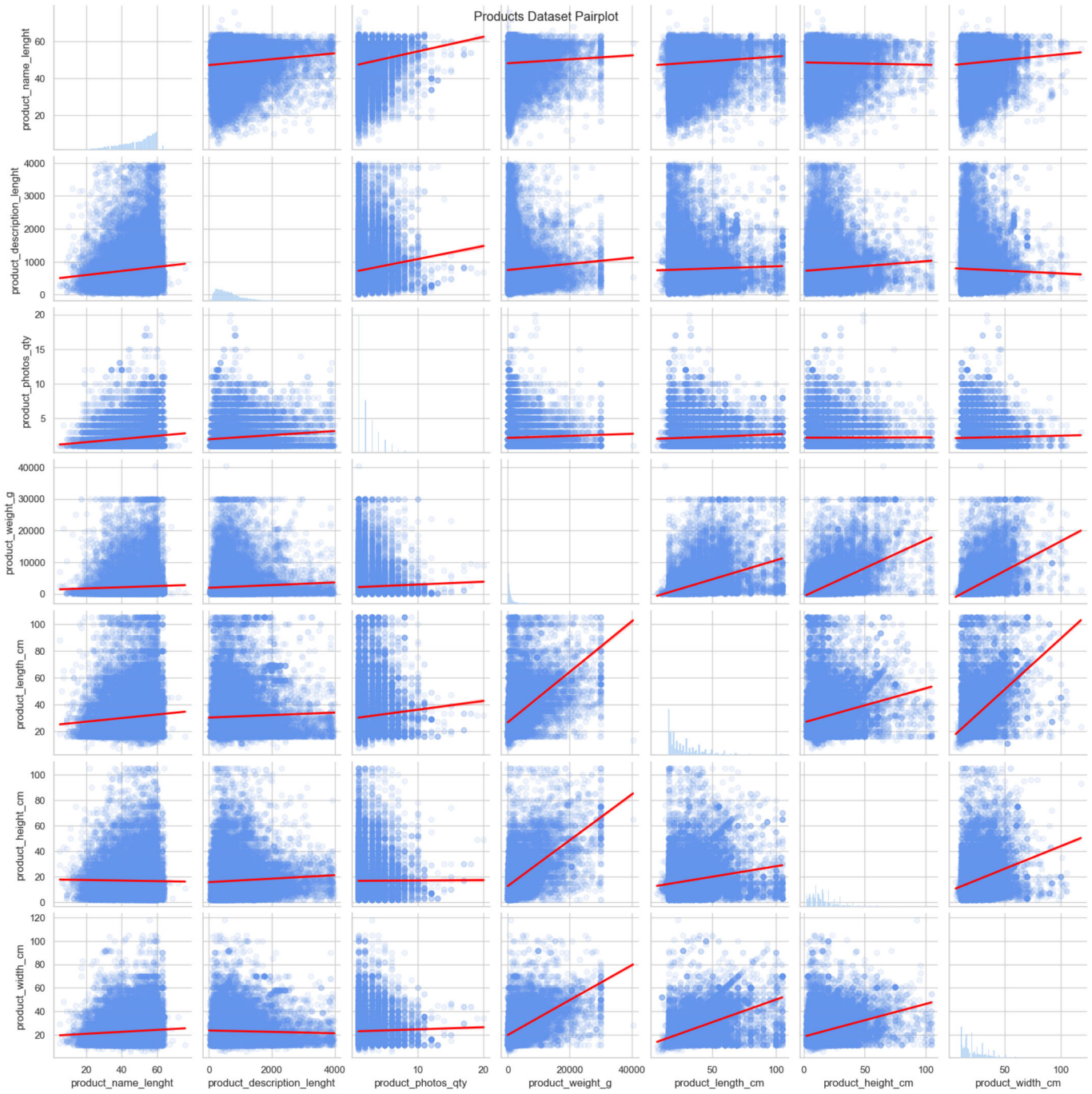
In [11]:
```
pair_plotter(payments, "Payments",  (8, 6))
```

```
<Figure size 800x600 with 0 Axes>
```



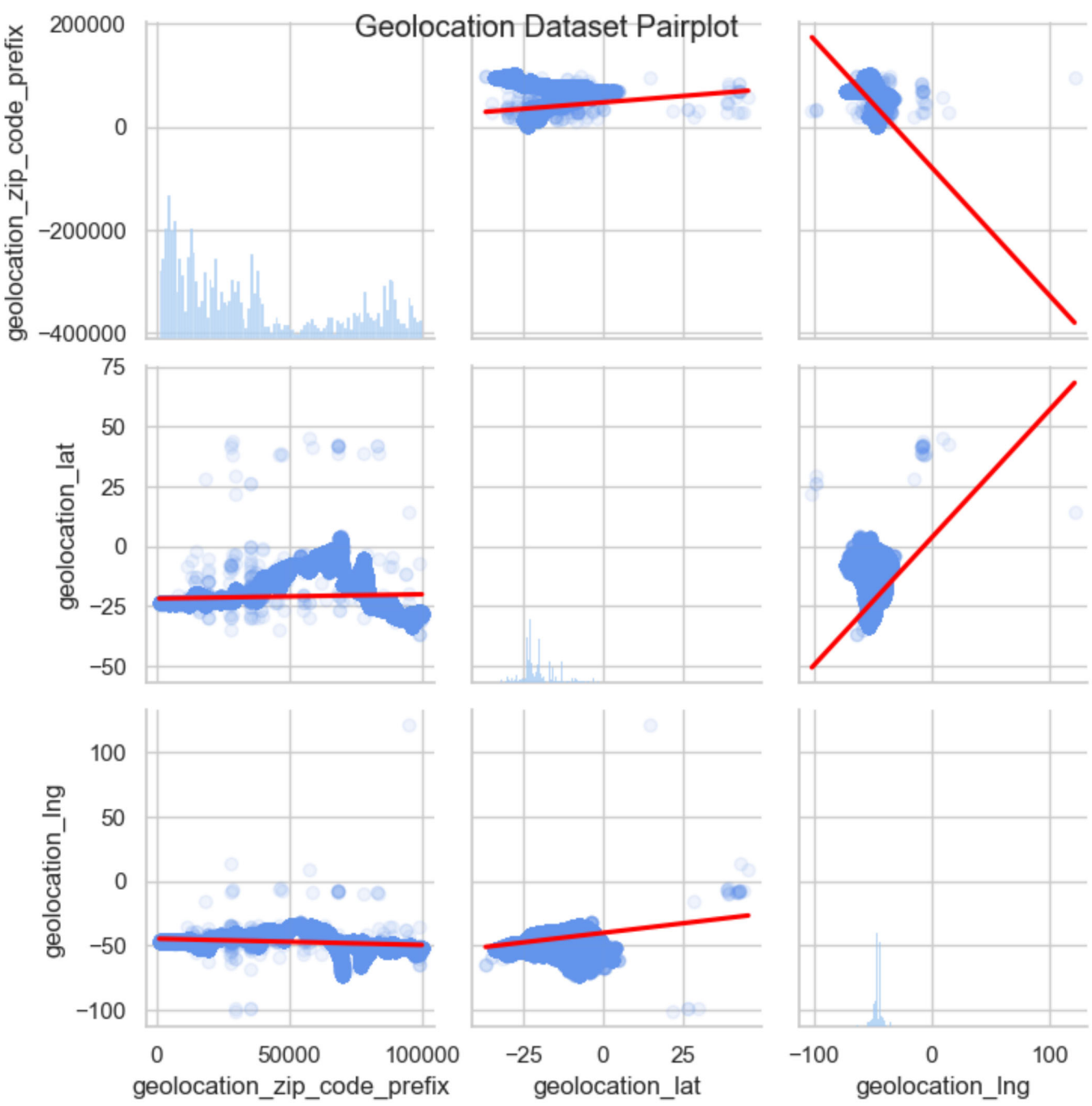
In [12]:
```
pair_plotter(products, "Products")
```

```
<Figure size 640x480 with 0 Axes>
```



In [13]:
```
pair_plotter(geolocation, "Geolocation", (8, 6))
```

```
<Figure size 800x600 with 0 Axes>
```



Orders EDA Visualization

Visualizing the monthly trend of orders as well as weekly and daily distributions. For general EDA and understanding of data. May be referred during later stages of EDA or model building.

In [14]:
```
orders['order_status'].value_counts()
```

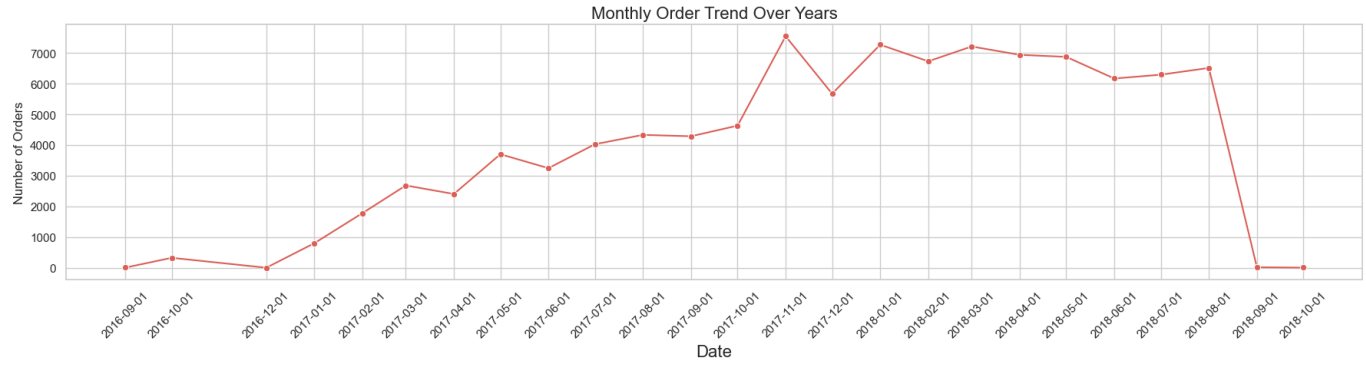
Out[14]:
```
order_status
delivered      96478
shipped         1107
canceled         625
unavailable      609
invoiced         314
processing       301
created            5
approved           2
Name: count, dtype: int64
```

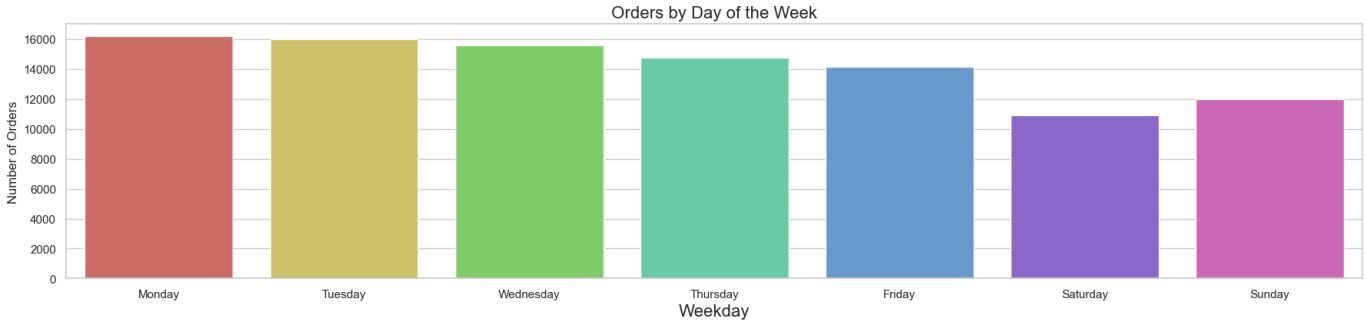
In [15]:
```
# convert to datetime
orders['order_purchase_timestamp'] = pd.to_datetime(orders['order_purchase_timestamp'])

# extract dt features
```

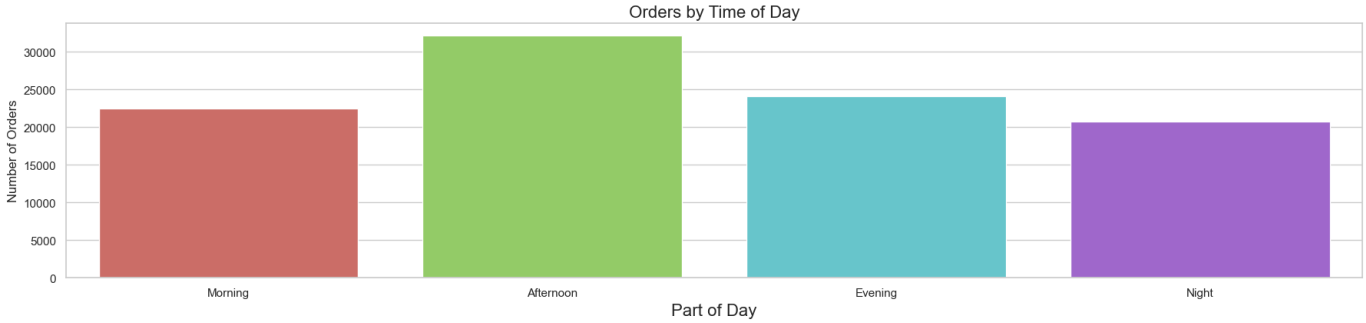
```
orders['year'] = orders['order_purchase_timestamp'].dt.year
orders['month'] = orders['order_purchase_timestamp'].dt.month
orders['weekday'] = orders['order_purchase_timestamp'].dt.day_name()
orders['hour'] = orders['order_purchase_timestamp'].dt.hour
```

In [16]:
```
# categorize time of day
def get_part_of_day(hour):
    """ Creates time of day categories from hour """
    if 5 <= hour < 12:
        return 'Morning'
    elif 12 <= hour < 17:
        return 'Afternoon'
    elif 17 <= hour < 21:
        return 'Evening'
    else:
        return 'Night'

orders['part_of_day'] = orders['hour'].apply(get_part_of_day)
```

In [17]:
```
sns.set_theme(style="whitegrid", palette="hls")
plt.figure(figsize=(18, 14))

# 1. monthly order trend over the years
monthly_orders = orders.groupby(['year', 'month']).size().reset_index(name='order_count')
monthly_orders['year_month'] = pd.to_datetime(monthly_orders[['year', 'month']].assign(day=1))

plt.subplot(311)
sns.lineplot(data=monthly_orders, x='year_month', y='order_count', marker='o')
plt.title('Monthly Order Trend Over Years', fontsize=16)
plt.xlabel('Date', fontsize=16)
plt.ylabel('Number of Orders')
plt.xticks(monthly_orders['year_month'], rotation=45)

# 2. orders by day of week
weekday_order = orders['weekday'].value_counts().reindex(['Monday', 'Tuesday', 'Wednesday', '

plt.subplot(312)
sns.barplot(x=weekday_order.index, y=weekday_order.values, palette="hls", hue=weekday_order.in
plt.ylim(0)
plt.title('Orders by Day of the Week', fontsize=16)
plt.xlabel('Weekday', fontsize=16)
plt.ylabel('Number of Orders')
plt.xticks()

# 3. orders by time of Day
plt.subplot(313)
part_of_day_order = orders['part_of_day'].value_counts().reindex(['Morning', 'Afternoon', 'Eve
sns.barplot(x=part_of_day_order.index, y=part_of_day_order.values, palette="hls", hue=part_of_
plt.ylim(0)
plt.title('Orders by Time of Day', fontsize=16)
plt.xlabel('Part of Day', fontsize=16)
plt.ylabel('Number of Orders')

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

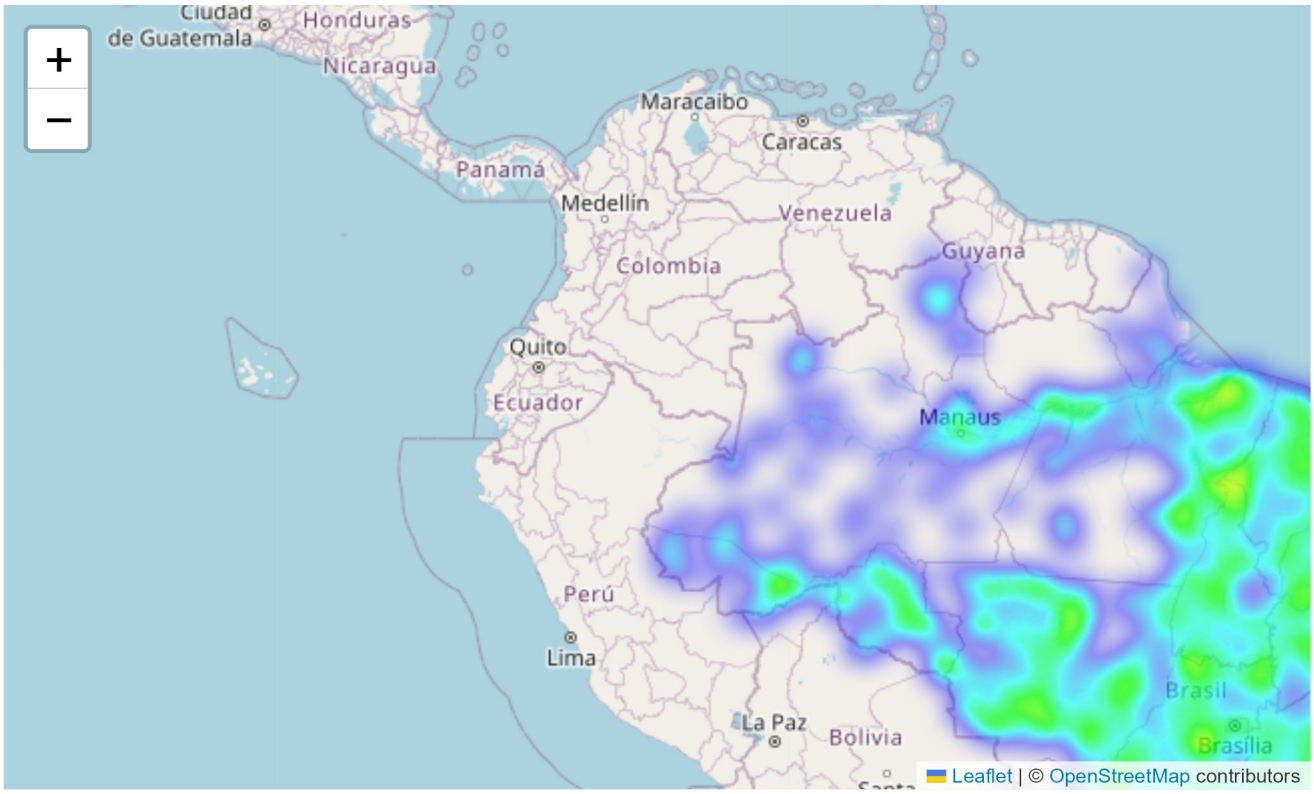

Monthly Order Trend Over Years
Number of Orders
0
1000
2000
3000
4000
5000
6000
7000
2016-09-01
2016-10-01
2016-12-01
2017-01-01
2017-02-01
2017-03-01
2017-04-01
2017-05-01
2017-06-01
2017-07-01
2017-08-01
2017-09-01
2017-10-01
2017-11-01
2017-12-01
2018-01-01
2018-02-01
2018-03-01
2018-04-01
2018-05-01
2018-06-01
2018-07-01
2018-08-01
2018-09-01
2018-10-01
Date


Orders by Day of the Week
Number of Orders
0
2000
4000
6000
8000
10000
12000
14000
16000
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Weekday


Orders by Time of Day
Number of Orders
0
5000
10000
15000
20000
25000
30000
Morning
Afternoon
Evening
Night
Part of Day

In [18]:
```
# create map centered around Brazil
brazil_center = [-14.2350, -51.9253]
m = folium.Map(location=brazil_center, zoom_start=4)

# prepare data for heatmap and add to layer
heat_data = geolocation[['geolocation_lat', 'geolocation_lng']].dropna().values.tolist()

HeatMap(heat_data, radius=8, blur=12, max_zoom=10).add_to(m)

m # plot
```

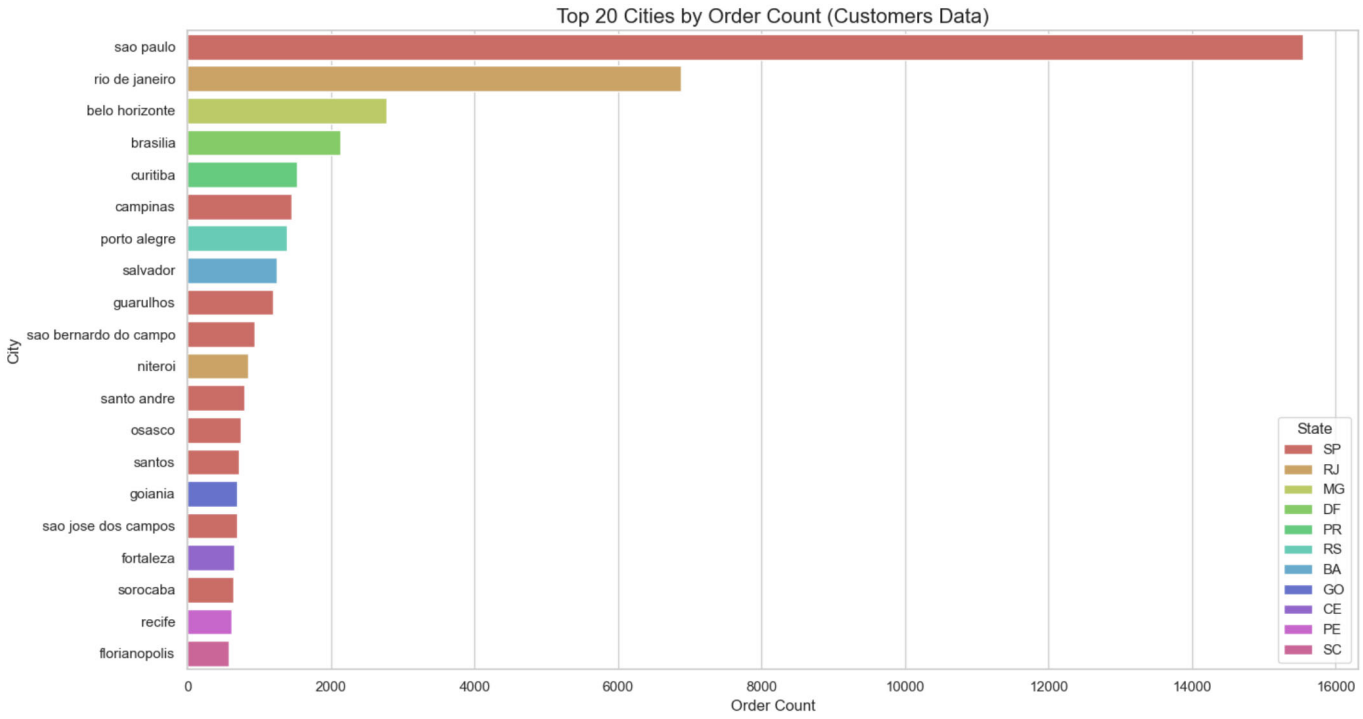
Out[18]:



In [19]:
```
# counts by customer's city and state
city_state_counts = customers.groupby(['customer_city', 'customer_state'])['customer_id'].cou
city_state_counts.rename(columns={'customer_id': 'order_count'}, inplace=True)
print(city_state_counts.head())
print("\n---------------------\nAmount of cities:", city_state_counts['customer_city'].nunique
print("Amount of state:", city_state_counts['customer_state'].nunique())
```

```
         customer_city customer_state  order_count
0  abadia dos dourados             MG            3
1            abadiania             GO            1
2               abaete             MG           12
3           abaetetuba             PA           11
4              abaiara             CE            2

---------------------
Amount of cities: 4119
Amount of state: 27
```

In [20]:
```
top_cities = city_state_counts.sort_values(by=['order_count'], ascending=False).head(20)

# city: text | state: colour/hue
plt.figure(figsize=(15, 8))
sns.barplot(x='order_count', y='customer_city', hue='customer_state', data=top_cities, palette
plt.title('Top 20 Cities by Order Count (Customers Data)', fontsize=16)
plt.xlabel('Order Count')
plt.ylabel('City')
plt.legend(title='State', loc='lower right')
plt.tight_layout()
plt.show()
```



Merge datasets

Creating a single dataset for EDA, visualization and modelling.

In [21]:
```
orders_delivered = orders[orders['order_status'] == 'delivered']
orders_delivered = orders_delivered.dropna()
orders_delivered['order_delivered_customer_date'] = pd.to_datetime(orders_delivered['order_del

df = pd.merge(orders_delivered, order_items, on='order_id')
df = pd.merge(df, payments, on='order_id')
df = pd.merge(df, reviews, on='order_id')
df = pd.merge(df, products, on='product_id')

df = pd.merge(df, customers, on='customer_id')

# count unique orders per customer from orders_delivered
order_counts = df.groupby('customer_unique_id')['order_id'].nunique().reset_index(name='order_

df = df.drop(['order_id', 'order_status', 'order_approved_at', 'order_delivered_carrier_date',
              'order_item_id', 'seller_id', 'review_id', 'review_comment_title', 'payment_sequ
              'product_name_lenght', 'product_description_lenght', 'review_comment_message',

geo_clean = geolocation.groupby('geolocation_zip_code_prefix')[['geolocation_lat', 'geolocatio

df = pd.merge(df, geo_clean, left_on='customer_zip_code_prefix', right_on='geolocation_zip_cod

df = df.drop(['customer_zip_code_prefix', 'customer_id'], axis=1)

print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114835 entries, 0 to 114834
Data columns (total 25 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_purchase_timestamp       114835 non-null  datetime64[ns]
 1   order_delivered_customer_date  114835 non-null  datetime64[ns]
 2   year                           114835 non-null  int32
 3   month                          114835 non-null  int32
 4   weekday                        114835 non-null  object
 5   hour                           114835 non-null  int32
 6   part_of_day                    114835 non-null  object
 7   shipping_limit_date            114835 non-null  object
 8   price                          114835 non-null  float64
 9   freight_value                  114835 non-null  float64
 10  payment_type                   114835 non-null  object
 11  payment_installments           114835 non-null  int64
 12  payment_value                  114835 non-null  float64
 13  review_score                   114835 non-null  int64
 14  product_category_name          113210 non-null  object
 15  product_photos_qty             113210 non-null  float64
 16  product_weight_g               114815 non-null  float64
 17  product_length_cm              114815 non-null  float64
 18  product_height_cm              114815 non-null  float64
 19  product_width_cm               114815 non-null  float64
 20  customer_unique_id             114835 non-null  object
 21  customer_city                  114835 non-null  object
 22  customer_state                 114835 non-null  object
 23  geolocation_lat                114533 non-null  float64
 24  geolocation_lng                114533 non-null  float64
dtypes: datetime64[ns](2), float64(10), int32(3), int64(2), object(8)
memory usage: 20.6+ MB
None
```

Feature Engineering 1

Creating more features for EDA and model learning.

In [22]:

```python
# using brazil (southern hemisphere), get seasons
def month_to_season(month):
    """ Create time of year categories from month """
    if month in [12, 1, 2]:
        return 'Summer'
    elif month in [3, 4, 5]:
        return 'Autumn'
    elif month in [6, 7, 8]:
        return 'Winter'
    else:  # [9, 10, 11]
        return 'Spring'

df['season'] = df['month'].apply(month_to_season)

# get last purchase date per customer
last_purchase = df.groupby('customer_unique_id')['order_purchase_timestamp'].max().reset_index

# get latest date in entire dataset
max_date = df['order_purchase_timestamp'].max()

# get months inactive
last_purchase['months_inactive'] = ((max_date - last_purchase['last_purchase_date']) / pd.Time

# get delivery time
df['delivery_time'] = (df['order_delivered_customer_date'] - df['order_purchase_timestamp']).
```

```
# merge and drop cols
distinct_products_per_customer = df.groupby('customer_unique_id')['product_category_name'].nun
df = df.drop(['order_purchase_timestamp', 'order_delivered_customer_date', 'month', 'year', 'p
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114835 entries, 0 to 114834
Data columns (total 22 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   weekday                114835 non-null  object
 1   hour                   114835 non-null  int32
 2   shipping_limit_date    114835 non-null  object
 3   price                  114835 non-null  float64
 4   freight_value          114835 non-null  float64
 5   payment_type           114835 non-null  object
 6   payment_installments   114835 non-null  int64
 7   payment_value          114835 non-null  float64
 8   review_score           114835 non-null  int64
 9   product_category_name  113210 non-null  object
 10  product_photos_qty     113210 non-null  float64
 11  product_weight_g       114815 non-null  float64
 12  product_length_cm      114815 non-null  float64
 13  product_height_cm      114815 non-null  float64
 14  product_width_cm       114815 non-null  float64
 15  customer_unique_id     114835 non-null  object
 16  customer_city          114835 non-null  object
 17  customer_state         114835 non-null  object
 18  geolocation_lat        114533 non-null  float64
 19  geolocation_lng        114533 non-null  float64
 20  season                 114835 non-null  object
 21  delivery_time          114835 non-null  int64
dtypes: float64(10), int32(1), int64(3), object(8)
memory usage: 18.8+ MB
None
```

Aggregating Merged Dataset

Aggregating features in the merged dataset to obtain singular customer rows.

In [23]:
```
# picking most common value (mode) in cat cols
# if multiple equally common values, get the first one.
# if empty, return None
agg_df = df.groupby('customer_unique_id').agg({
    'weekday': lambda x: x.mode().iloc[0] if not x.mode().empty else None,
    'hour': 'median',
    'price': ['sum', 'mean'],
    'freight_value': 'mean',
    'payment_type': lambda x: x.mode().iloc[0] if not x.mode().empty else None,
    'payment_installments': 'mean',
    'payment_value': ['sum', 'mean'],
    'review_score': 'mean',
    'product_category_name': lambda x: x.mode().iloc[0] if not x.mode().empty else None,
    'product_photos_qty': 'mean',
    'product_weight_g': 'mean',
    'product_length_cm': 'mean',
    'product_height_cm': 'mean',
    'product_width_cm': 'mean',
    'customer_city': lambda x: x.mode().iloc[0] if not x.mode().empty else None,
    'customer_state': lambda x: x.mode().iloc[0] if not x.mode().empty else None,
    'geolocation_lat': 'mean',
    'geolocation_lng': 'mean',
    'season': lambda x: x.mode().iloc[0] if not x.mode().empty else None,
```

```
    'delivery_time': 'mean'
}).reset_index()
```

In [24]:
```
# flatten multi index columns
agg_df.columns = ['_'.join(col).strip('_') if isinstance(col, tuple) else col for col in agg_c

# translating product name to english
agg_df = pd.merge(agg_df, product_translation, left_on='product_category_name_<lambda>', right
agg_df = agg_df.drop(['product_category_name', 'product_category_name_<lambda>'], axis=1) # Dr

# merge engineered features
agg_df = pd.merge(agg_df, order_counts, on='customer_unique_id', how='left')

# define repeat buyer
agg_df['repeat_buyer'] = agg_df['order_count'].apply(lambda x: 1 if x > 1 else 0)
agg_df = pd.merge(agg_df, distinct_products_per_customer, on='customer_unique_id')
agg_df = pd.merge(agg_df, last_purchase[['customer_unique_id', 'months_inactive']], on='custor
```

In [25]:
```
print(agg_df.head().to_string())
```

```
                customer_unique_id weekday_<lambda>  hour_median  price_sum  price_mean  frei
ght_value_mean payment_type_<lambda>  payment_installments_mean  payment_value_sum  payment_va
lue_mean  review_score_mean  product_photos_qty_mean  product_weight_g_mean  product_length_cm
_mean  product_height_cm_mean  product_width_cm_mean customer_city_<lambda> customer_state_<la
mbda>  geolocation_lat_mean  geolocation_lng_mean season_<lambda>  delivery_time_mean product_
category_name_english  order_count  repeat_buyer  distinct_product_categories  months_inactive
0  0000366f3b9a7992bf8c76cfdf3221e2         Thursday         10.0     129.90      129.90
12.00           credit_card                       8.0             141.90              141.90
5.0                     1.0                  1500.0                  34.0
7.0                  32.0                cajamar                       SP            -23.34023
5            -46.830140          Autumn                6.0                bed_bath_table
1             0                              1               4
1  0000b849f77a49e4a4ce2b2a4ca5be3f           Monday         11.0      18.90       18.90
8.29           credit_card                       1.0             27.19               27.19
4.0                     1.0                  375.0                  26.0
11.0                 18.0                osasco                       SP            -23.5591
15            -46.787626          Autumn                3.0                 health_beauty
1             0                              1               4
2  0000f46a3911fa3c0805444483337064           Friday         21.0      69.00       69.00
17.22           credit_card                       8.0             86.22               86.22
3.0                     3.0                  1500.0                  25.0
50.0                 35.0                sao jose                      SC            -27.5428
80            -48.633426          Autumn               25.0                    stationery
1             0                              1              18
3  0000f6ccb0745a6a4b88665a16c9f078         Thursday         20.0      25.99       25.99
17.63           credit_card                       4.0             43.62               43.62
4.0                     5.0                  150.0                  19.0
5.0                  11.0                belem                       PA            -1.31221
4            -48.483159          Spring               20.0                    telephony
1             0                              1              11
4  0004aac84e0df4da2b147fca70cf8255           Tuesday         19.0     180.00      180.00
16.89           credit_card                       6.0             196.89              196.89
5.0                     3.0                  6050.0                  16.0
3.0                  11.0              sorocaba                       SP            -23.50554
8            -47.469705          Spring               13.0                    telephony
1             0                              1              10
```

In [26]:
```
print(agg_df['payment_type_<lambda>'].value_counts())
print(agg_df['season_<lambda>'].value_counts())
print(agg_df['distinct_product_categories'].value_counts())
```

```
payment_type_<lambda>
credit_card    71031
boleto         18548
voucher         1732
debit_card      1421
Name: count, dtype: int64
season_<lambda>
Winter    28512
Autumn    28121
Summer    20561
Spring    15538
Name: count, dtype: int64
distinct_product_categories
1    89340
2     2018
0     1253
3      105
4       11
5        5
Name: count, dtype: int64
```

In [27]:
```
print(agg_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92732 entries, 0 to 92731
Data columns (total 27 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   customer_unique_id              92732 non-null  object
 1   weekday_<lambda>                92732 non-null  object
 2   hour_median                     92732 non-null  float64
 3   price_sum                       92732 non-null  float64
 4   price_mean                      92732 non-null  float64
 5   freight_value_mean              92732 non-null  float64
 6   payment_type_<lambda>           92732 non-null  object
 7   payment_installments_mean       92732 non-null  float64
 8   payment_value_sum               92732 non-null  float64
 9   payment_value_mean              92732 non-null  float64
 10  review_score_mean               92732 non-null  float64
 11  product_photos_qty_mean         91479 non-null  float64
 12  product_weight_g_mean           92719 non-null  float64
 13  product_length_cm_mean          92719 non-null  float64
 14  product_height_cm_mean          92719 non-null  float64
 15  product_width_cm_mean           92719 non-null  float64
 16  customer_city_<lambda>          92732 non-null  object
 17  customer_state_<lambda>         92732 non-null  object
 18  geolocation_lat_mean            92479 non-null  float64
 19  geolocation_lng_mean            92479 non-null  float64
 20  season_<lambda>                 92732 non-null  object
 21  delivery_time_mean              92732 non-null  float64
 22  product_category_name_english   91460 non-null  object
 23  order_count                     92732 non-null  int64
 24  repeat_buyer                    92732 non-null  int64
 25  distinct_product_categories     92732 non-null  int64
 26  months_inactive                 92732 non-null  int32
dtypes: float64(16), int32(1), int64(3), object(7)
memory usage: 18.7+ MB
None
```

Feature Engineering 2

Creating more features for EDA, visualization and model learning.

In [28]:
```
# reduce geolocation to one latitude/longitude per zip code prefix
# (take average of all points sharing same prefix)
geo_prefix = geolocation.groupby('geolocation_zip_code_prefix').agg({
    'geolocation_lat': 'mean',   # average latitude
    'geolocation_lng': 'mean'    # average longitude
}).reset_index()

# merge averaged coords into customers df
customers_loc = (
    customers
    .merge(
        geo_prefix,
        left_on='customer_zip_code_prefix',         # match customer prefix
        right_on='geolocation_zip_code_prefix',     # to averaged geolocation
        how='left'                                  # keep all customers
    )
    .rename(columns={
        'geolocation_lat': 'customer_lat',
        'geolocation_lng': 'customer_lng'
    })
    .drop('geolocation_zip_code_prefix', axis=1)  # drop redundant column
)

# merge averaged coords into sellers df
sellers_loc = (
    sellers
    .merge(
        geo_prefix,
        left_on='seller_zip_code_prefix',
        right_on='geolocation_zip_code_prefix',
        how='left'
    )
    .rename(columns={
        'geolocation_lat': 'seller_lat',
        'geolocation_lng': 'seller_lng'
    })
    .drop('geolocation_zip_code_prefix', axis=1)
)

# build one dataframe linking orders to customers to sellers
order_merged = (
    orders[['order_id', 'customer_id']]                    # start with orders and customer
    .merge(
        customers[['customer_id', 'customer_unique_id']],   # attach customer_unique_id from
        on='customer_id'
    )
    .merge(
        order_items[['order_id', 'seller_id']],             # attach seller IDs via order_iter
        on='order_id'
    )
    .merge(
        customers_loc[['customer_id', 'customer_lat', 'customer_lng']],
        on='customer_id',                                   # attach customer coords
        how='left'
    )
    .merge(
        sellers_loc[['seller_id', 'seller_lat', 'seller_lng']],
        on='seller_id',                                     # attach seller coords
        how='left'
    )
)
```

In [29]:
```
# calculate distances on round earth surface using lattitude + longitude: Haversine forumla
def haversine(lat1, lon1, lat2, lon2):
    """ Get great circle distance using Haversine Formula
```

```
    Parameters:
        lat1: customer lattitude
        lon1: customer longitude
        lat2: seller lattitude
        lon2: seller longitude
    Returns:
        distance in km
    """
    R = 6371.0  # Earth's radius in km

    # convert degrees → radians
    phi1, phi2 = radians(lat1), radians(lat2)
    dphi = radians(lat2 - lat1)
    dlambda = radians(lon2 - lon1)

    # Haversine calculation
    a = sin(dphi / 2)**2 + cos(phi1) * cos(phi2) * sin(dlambda / 2)**2
    return 2 * R * atan2(sqrt(a), sqrt(1 - a))

order_merged['distance_km'] = order_merged.apply(
    lambda row: haversine(
        row['customer_lat'], row['customer_lng'],
        row['seller_lat'],   row['seller_lng']
    )
    if pd.notnull(row['customer_lat']) and pd.notnull(row['seller_lat'])
    else None,  # skip if any coordinate is missing
    axis=1
)

order_merged.head()
```

Out[29]:

	order_id	customer_id	customer_uni
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	7c396fd4830fd04220f754e42
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	af07308b275d755c9edb36a90c
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	3a653a41f6f9fc3d2a113cf839
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	7c142cf63193a1473d2e66489a
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	72632f0f9dd73dfee390c9b22e

In [30]:
```
# agg avg distance per customer
avg_distance_per_customer = (
    order_merged
    .groupby('customer_unique_id')['distance_km']
    .mean()
    .reset_index()
    .rename(columns={'distance_km': 'avg_distance_km'})
)

# merge back into agg_df
agg_df = pd.merge(agg_df, avg_distance_per_customer, on='customer_unique_id', how='left')
```

In [31]:
```
print(agg_df.head().to_string())
```

```
                customer_unique_id weekday_<lambda>  hour_median  price_sum  price_mean  frei
ght_value_mean payment_type_<lambda>  payment_installments_mean  payment_value_sum  payment_va
lue_mean  review_score_mean  product_photos_qty_mean  product_weight_g_mean  product_length_cm
_mean  product_height_cm_mean  product_width_cm_mean customer_city_<lambda> customer_state_<la
mbda>  geolocation_lat_mean  geolocation_lng_mean season_<lambda>  delivery_time_mean product_
category_name_english  order_count  repeat_buyer  distinct_product_categories  months_inactive
avg_distance_km
0  0000366f3b9a7992bf8c76cfdf3221e2         Thursday         10.0     129.90      129.90
12.00           credit_card                       8.0             141.90              141.90
5.0                     1.0                1500.0                    34.0
7.0                   32.0                cajamar                      SP            -23.34023
5            -46.830140          Autumn                6.0                bed_bath_table
1             0                          1                4       110.568636
1  0000b849f77a49e4a4ce2b2a4ca5be3f           Monday         11.0      18.90       18.90
8.29           credit_card                       1.0             27.19               27.19
4.0                     1.0                 375.0                    26.0
11.0                   18.0                osasco                      SP            -23.5591
15            -46.787626          Autumn                3.0                health_beauty
1             0                          1                4        22.168333
2  0000f46a3911fa3c0805444483337064           Friday         21.0      69.00       69.00
17.22           credit_card                       8.0             86.22               86.22
3.0                     3.0                1500.0                    25.0
50.0                   35.0               sao jose                     SC            -27.5428
80            -48.633426          Autumn               25.0                     stationery
1             0                          1               18       516.938836
3  0000f6ccb0745a6a4b88665a16c9f078         Thursday         20.0      25.99       25.99
17.63           credit_card                       4.0             43.62               43.62
4.0                     5.0                 150.0                    19.0
5.0                   11.0                belem                      PA            -1.31221
4            -48.483159          Spring               20.0                     telephony
1             0                          1               11      2481.287188
4  0004aac84e0df4da2b147fca70cf8255          Tuesday         19.0     180.00      180.00
16.89           credit_card                       6.0             196.89              196.89
5.0                     3.0                6050.0                    16.0
3.0                   11.0               sorocaba                     SP            -23.50554
8            -47.469705          Spring               13.0                     telephony
1             0                          1               10       154.507887
```

In [32]:
```
print(agg_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 92732 entries, 0 to 92731
Data columns (total 28 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   customer_unique_id              92732 non-null  object
 1   weekday_<lambda>                92732 non-null  object
 2   hour_median                     92732 non-null  float64
 3   price_sum                       92732 non-null  float64
 4   price_mean                      92732 non-null  float64
 5   freight_value_mean              92732 non-null  float64
 6   payment_type_<lambda>           92732 non-null  object
 7   payment_installments_mean       92732 non-null  float64
 8   payment_value_sum               92732 non-null  float64
 9   payment_value_mean              92732 non-null  float64
 10  review_score_mean               92732 non-null  float64
 11  product_photos_qty_mean         91479 non-null  float64
 12  product_weight_g_mean           92719 non-null  float64
 13  product_length_cm_mean          92719 non-null  float64
 14  product_height_cm_mean          92719 non-null  float64
 15  product_width_cm_mean           92719 non-null  float64
 16  customer_city_<lambda>          92732 non-null  object
 17  customer_state_<lambda>         92732 non-null  object
 18  geolocation_lat_mean            92479 non-null  float64
 19  geolocation_lng_mean            92479 non-null  float64
 20  season_<lambda>                 92732 non-null  object
 21  delivery_time_mean              92732 non-null  float64
 22  product_category_name_english   91460 non-null  object
 23  order_count                     92732 non-null  int64
 24  repeat_buyer                    92732 non-null  int64
 25  distinct_product_categories     92732 non-null  int64
 26  months_inactive                 92732 non-null  int32
 27  avg_distance_km                 92280 non-null  float64
dtypes: float64(17), int32(1), int64(3), object(7)
memory usage: 19.5+ MB
None
```

In [33]:
```
print(agg_df[['product_length_cm_mean', 'product_height_cm_mean', 'product_width_cm_mean', 'pr
```

```
       product_length_cm_mean  product_height_cm_mean  product_width_cm_mean  product_weight_g
_mean
count            92719.000000            92719.000000           92719.000000           92719.0
00000
mean                30.059017               16.468115              22.988345            2098.7
20480
std                 15.948927               13.211282              11.613216            3726.1
93197
min                  7.000000                2.000000               6.000000               0.0
00000
25%                 18.000000                8.000000              15.000000             300.0
00000
50%                 25.000000               13.000000              20.000000             700.0
00000
75%                 38.000000               20.000000              30.000000            1825.0
00000
max                105.000000              105.000000             118.000000           40425.0
00000
```

In [34]:
```
# creating volume mean for models
agg_df['volume_mean'] = agg_df['product_length_cm_mean'] * agg_df['product_width_cm_mean'] * a
print(agg_df[['volume_mean', 'product_length_cm_mean', 'product_width_cm_mean', 'product_heigh
```

	volume_mean	product_length_cm_mean	product_width_cm_mean	product_height_cm_mean
0	7616.0	34.0	32.0	7.0
1	5148.0	26.0	18.0	11.0
2	43750.0	25.0	35.0	50.0
3	1045.0	19.0	11.0	5.0
4	528.0	16.0	11.0	3.0

In [35]:
```
# replace 0g weights to 0.1g weights to avoid division by zero error
agg_df['product_weight_g_mean'] = agg_df['product_weight_g_mean'].replace(0, 0.1)

agg_df['cost_volume'] = agg_df['price_mean'] / agg_df['volume_mean']
agg_df['density_mean'] = agg_df['product_weight_g_mean'] / agg_df['volume_mean']
agg_df['cost_weight'] = agg_df['price_mean'] / agg_df['product_weight_g_mean']
agg_df['lh_ratio'] = agg_df['product_length_cm_mean'] / agg_df['product_height_cm_mean']
agg_df['lw_ratio'] = agg_df['product_length_cm_mean'] / agg_df['product_width_cm_mean']
agg_df['hw_ratio'] = agg_df['product_height_cm_mean'] / agg_df['product_width_cm_mean']

print(agg_df[['cost_volume', 'density_mean', 'cost_weight', 'lh_ratio', 'lw_ratio', 'hw_ratio
```

	cost_volume	density_mean	cost_weight	lh_ratio	lw_ratio	hw_ratio
0	0.017056	0.196954	0.086600	4.857143	1.062500	0.218750
1	0.003671	0.072844	0.050400	2.363636	1.444444	0.611111
2	0.001577	0.034286	0.046000	0.500000	0.714286	1.428571
3	0.024871	0.143541	0.173267	3.800000	1.727273	0.454545
4	0.340909	11.458333	0.029752	5.333333	1.454545	0.272727

In [36]:
```
# confirm no duplicates or nulls and drop for modelling
print(agg_df.duplicated().sum())
print(agg_df.isnull().sum().sum())
```

```
0
3626
```

In [37]:
```
agg_df.dropna(inplace=True)
print(agg_df.isnull().sum().sum())
```

```
0
```

Saving Final Dataset

For convenient use in model building stage.

In [38]:
```
agg_df.to_csv('data/03_primary/olist_final.csv', index=False)
```

Checking Saved Dataset

In [39]:
```
final_df = pd.read_csv('data/03_primary/olist_final.csv')
```

In [40]:
```
print(final_df.shape, '\n--------------------')
print(final_df.head().to_string())
```

```
(91013, 35)
--------------------
                 customer_unique_id weekday_<lambda>  hour_median  price_sum  price_mean  frei
ght_value_mean payment_type_<lambda>  payment_installments_mean  payment_value_sum  payment_va
lue_mean  review_score_mean  product_photos_qty_mean  product_weight_g_mean  product_length_cm
_mean  product_height_cm_mean  product_width_cm_mean customer_city_<lambda> customer_state_<la
mbda>  geolocation_lat_mean  geolocation_lng_mean season_<lambda>  delivery_time_mean product_
category_name_english  order_count  repeat_buyer  distinct_product_categories  months_inactive
avg_distance_km  volume_mean  cost_volume  density_mean  cost_weight  lh_ratio  lw_ratio  hw_r
atio
0  0000366f3b9a7992bf8c76cfdf3221e2         Thursday         10.0     129.90      129.90
12.00           credit_card                        8.0             141.90              141.90
5.0                     1.0                 1500.0                  34.0
7.0                  32.0                cajamar                    SP            -23.34023
5            -46.830140          Autumn                6.0                bed_bath_table
1             0                             1               4       110.568636       7616.0
0.017056      0.196954     0.086600  4.857143  1.062500  0.218750
1  0000b849f77a49e4a4ce2b2a4ca5be3f           Monday         11.0      18.90       18.90
8.29           credit_card                       1.0             27.19               27.19
4.0                     1.0                 375.0                  26.0
11.0                 18.0                osasco                    SP            -23.5591
15            -46.787626          Autumn                3.0                health_beauty
1             0                             1               4        22.168333       5148.0
0.003671      0.072844     0.050400  2.363636  1.444444  0.611111
2  0000f46a3911fa3c0805444483337064           Friday         21.0      69.00       69.00
17.22           credit_card                       8.0             86.22               86.22
3.0                     3.0                 1500.0                  25.0
50.0                 35.0               sao jose                    SC            -27.5428
80            -48.633426          Autumn               25.0                    stationery
1             0                             1              18       516.938836      43750.0
0.001577      0.034286     0.046000  0.500000  0.714286  1.428571
3  0000f6ccb0745a6a4b88665a16c9f078         Thursday         20.0      25.99       25.99
17.63           credit_card                       4.0             43.62               43.62
4.0                     5.0                  150.0                  19.0
5.0                  11.0                belem                     PA            -1.31221
4            -48.483159          Spring               20.0                    telephony
1             0                             1              11      2481.287188       1045.0
0.024871      0.143541     0.173267  3.800000  1.727273  0.454545
4  0004aac84e0df4da2b147fca70cf8255          Tuesday         19.0     180.00      180.00
16.89           credit_card                       6.0             196.89              196.89
5.0                     3.0                 6050.0                  16.0
3.0                  11.0              sorocaba                    SP            -23.50554
8            -47.469705          Spring               13.0                    telephony
1             0                             1              10       154.507887        528.0
0.340909     11.458333     0.029752  5.333333  1.454545  0.272727
```

Visualization Plot Functions

Defining various plot functions for target features: repeat_buyer, freight_value_mean, delivery_time_mean

In [41]:
```python
def plot_feature_target_distribution(df: pd.DataFrame, feature: str, target: str):
    """
    Creates grouped bar chart to show percentage distribution of a given feature
    within each repeat_buyer group (Repeat Buyer vs One-time Buyer).

    Parameters:
        feature: column name to analyze
        target: target name
        df: dataframe
    Returns:
        grouped bar graph plot
    """
    if feature not in df.columns:
```

```
        raise ValueError(f"'{feature}' not found in the DataFrame.")
    if target not in df.columns:
        raise ValueError(f"'{target}' not found in the DataFrame.")

    # prepare and count
    plot_df = df.groupby([target, feature]).size().reset_index(name='count')
    plot_df[target] = plot_df[target].map({1: 'Repeat Buyer', 0: 'One-time Buyer'})

    # total count per group (repeat vs one-time)
    group_totals = plot_df.groupby(target)['count'].transform('sum')
    plot_df['percentage'] = (plot_df['count'] / group_totals * 100).round(2)

    # sort feature values if numeric
    if pd.api.types.is_numeric_dtype(df[feature]):
        plot_df = plot_df.sort_values(by=feature)

    # Plot
    sns.set_theme(style="whitegrid")
    plt.figure(figsize=(12, 6))
    sns.barplot(
        data=plot_df,
        x=feature,
        y='percentage',
        hue=target,
        palette='Set2'
    )

    plt.title(f'{feature.replace("_", " ").title()} (% within Repeat Buyer Groups)', fontsize=
    plt.xlabel(feature.replace("_", " ").title())
    plt.ylabel('Percentage of Customers (%)')
    plt.legend(title='Customer Type')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()
```

In [42]:

```
def plot_kde_by_class_target(df: pd.DataFrame, feature: str, target: str, labels: list, clip_u
    """
    Plots KDE distribution for a num feature, comparing target classes.

    Parameters:
        df: dataframe with feature and target
        feature: numeric column to visualize.
        target: target column
        clip_upper (float, optional): upper limit to clip outliers for cleaner visualization.
        bandwidth_adjust (float, optional): KDE smoothing.
    Returns:
        KDE plot
    """
    if feature not in df.columns or target not in df.columns:
        raise ValueError(f"'{feature}' or '{target}' not found in DataFrame.")

    plot_df = df.copy()

    # optional clipping to handle outliers
    if clip_upper is not None:
        clipped_col = f"{feature}_clipped"
        plot_df[clipped_col] = plot_df[feature].clip(upper=clip_upper)
        num_clipped = (plot_df[feature] > clip_upper).sum()
        print(f"Clipped {num_clipped} values > {clip_upper} in '{feature}'.")
        feature = clipped_col

    plt.figure(figsize=(10, 6))
    sns.kdeplot(
        data=plot_df[plot_df[target] == 0],
        x=feature,
```

```
        label=labels[0],
        fill=True,
        color='salmon',
        alpha=0.5,
        bw_adjust=bandwidth_adjust
    )
    sns.kdeplot(
        data=plot_df[plot_df[target] == 1],
        x=feature,
        label=labels[1],
        fill=True,
        color='seagreen',
        alpha=0.5,
        bw_adjust=bandwidth_adjust
    )

    plt.title(f'Distribution of {feature.replace("_", " ").title()}', fontsize=16)
    plt.xlabel(feature.replace("_", " ").title())
    plt.ylabel('Density')
    plt.legend(title='Customer Type')
    plt.tight_layout()
    plt.show()
```

In [43]:
```
def plot_boxplot_by_target(df: pd.DataFrame, feature: str, target: str, clip_upper_quantile=0.
    """
    Draws a boxplot comparing a categorical feature across buyer types.
    Clipping is applied to reduce the impact of extreme outliers.

    Parameters:
        df: dataframe with feature and target
        feature: numeric column to visualize.
        target: target column
        clip_upper_quantile (float): Quantile to clip the feature at (default is 95th percent
    Returns:
        Box Plot
    """
    if feature not in df.columns or target not in df.columns:
        raise ValueError(f"'{feature}' or '{target}' not found in DataFrame.")

    df_plot = df.copy()

    # determine clipping threshold
    upper_threshold = df_plot[feature].quantile(clip_upper_quantile)
    num_clipped = (df_plot[feature] > upper_threshold).sum()
    df_plot[feature] = df_plot[feature].clip(upper=upper_threshold)

    # print note
    if num_clipped > 0:
        print(f"Note: {num_clipped} values clipped at {clip_upper_quantile*100:.0f}th percent

    sns.set_theme(style="whitegrid")
    plt.figure(figsize=(6, 4))
    sns.boxplot(data=df_plot, x=target, y=feature)
    plt.title(f'{feature.replace("_", " ").title()} by {target.replace("_", " ").title()} (Cl
    plt.xlabel(target.replace("_", " ").title())
    plt.ylabel(feature.replace("_", " ").title())

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

In [44]:
```
def plot_histogram_by_feature(df: pd.DataFrame, feature: str, target: str, bins: int=30, clip_
    """
    Draws a histograme comparing a categorical feature across buyer types.
    Clipping is applied to reduce the impact of extreme outliers.
```

```
    Parameters:
        df: dataframe with feature and target
        feature: numeric column to visualize.
        target: target column
        bins: range of values for a bin to capture
        clip_upper (float, optional): upper limit to clip outliers for cleaner visualization.
    Returns:
        Histogram Plot
    """
    if feature not in df.columns or target not in df.columns:
        raise ValueError(f"'{feature}' or '{target}' not found in DataFrame.")

    df_plot = df.copy()
    unique_classes = df_plot[feature].unique()
    palette = sns.color_palette("hls", n_colors=len(unique_classes))

    if clip_upper is not None:
        df_plot[feature] = df_plot[feature].clip(upper=clip_upper)

    plt.figure(figsize=(10, 6))
    if len(unique_classes) < 12:
        sns.histplot(data=df_plot, x=target, bins=bins, palette=palette, hue=feature,
                     element='step', stat='density', common_norm=False)
    else:
        print("Too many labels: ", len(unique_classes))
        sns.histplot(data=df_plot, x=target, bins=bins, palette=palette, hue=feature,
                     element='step', stat='density', common_norm=False, legend=False)
    plt.title(f'Histogram of {target.replace("_", " ").title()} by {feature.replace("_", " ")
    plt.xlabel(feature.replace("_", " ").title())
    plt.ylabel('Density')
    plt.tight_layout()
    plt.show()
```

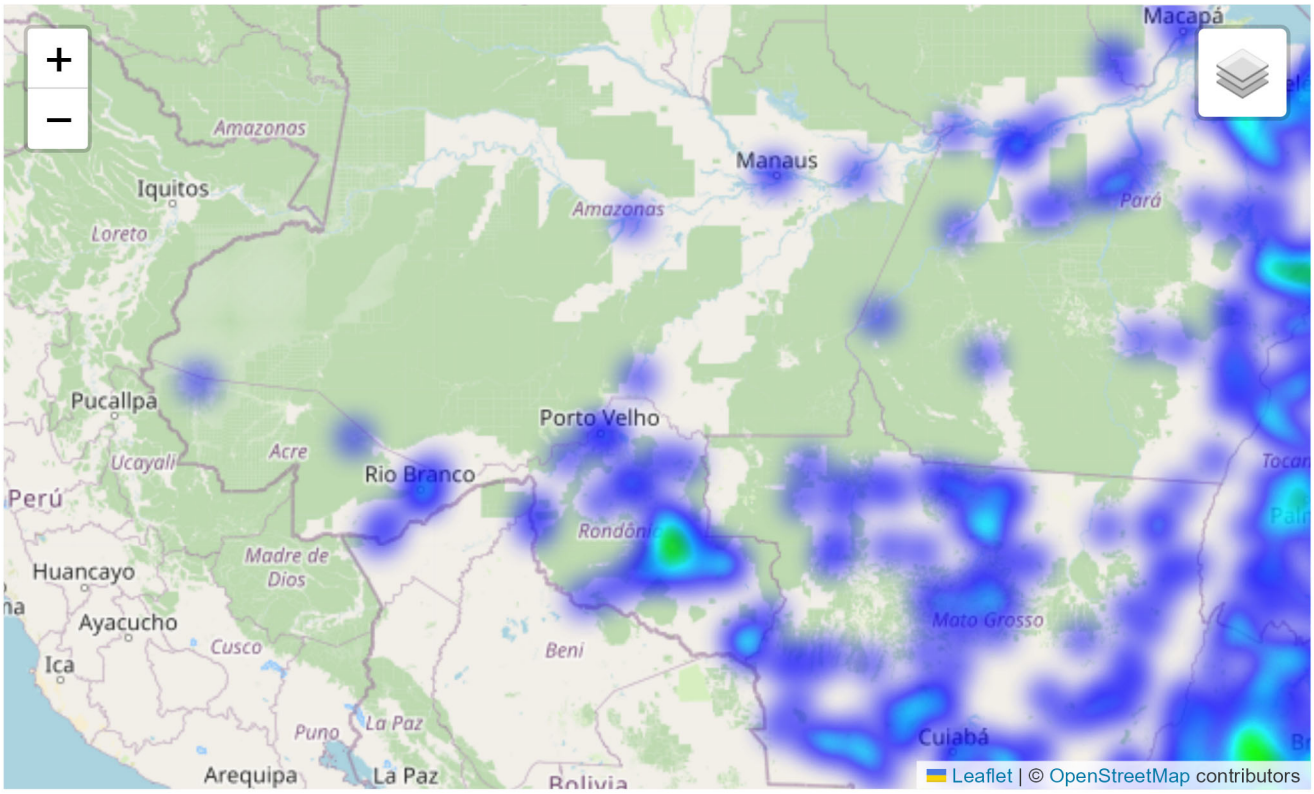
EDA: Repeat Buyer Target

Visualizing various data on Repeat Buyer Target for model building and to gather insights.

In [45]:
```
repeat_coords = agg_df[agg_df['repeat_buyer'] == 1][['geolocation_lat_mean','geolocation_lng_
non_repeat_coords = agg_df[agg_df['repeat_buyer'] == 0][['geolocation_lat_mean','geolocation_
standard_coords = agg_df[['geolocation_lat_mean', 'geolocation_lng_mean']].dropna().values.to

m = folium.Map(location=brazil_center, zoom_start=5)

# all buyers
HeatMap(
    standard_coords,
    radius=8,
    blur=12,
    max_zoom=10,
    name='All Buyers'
).add_to(m)

# non-repeat buyers
HeatMap(
    non_repeat_coords,
    radius=8,
    blur=12,
    max_zoom=10,
    name='Non-Repeat Buyers'
).add_to(m)

# repeat buyers
HeatMap(
```

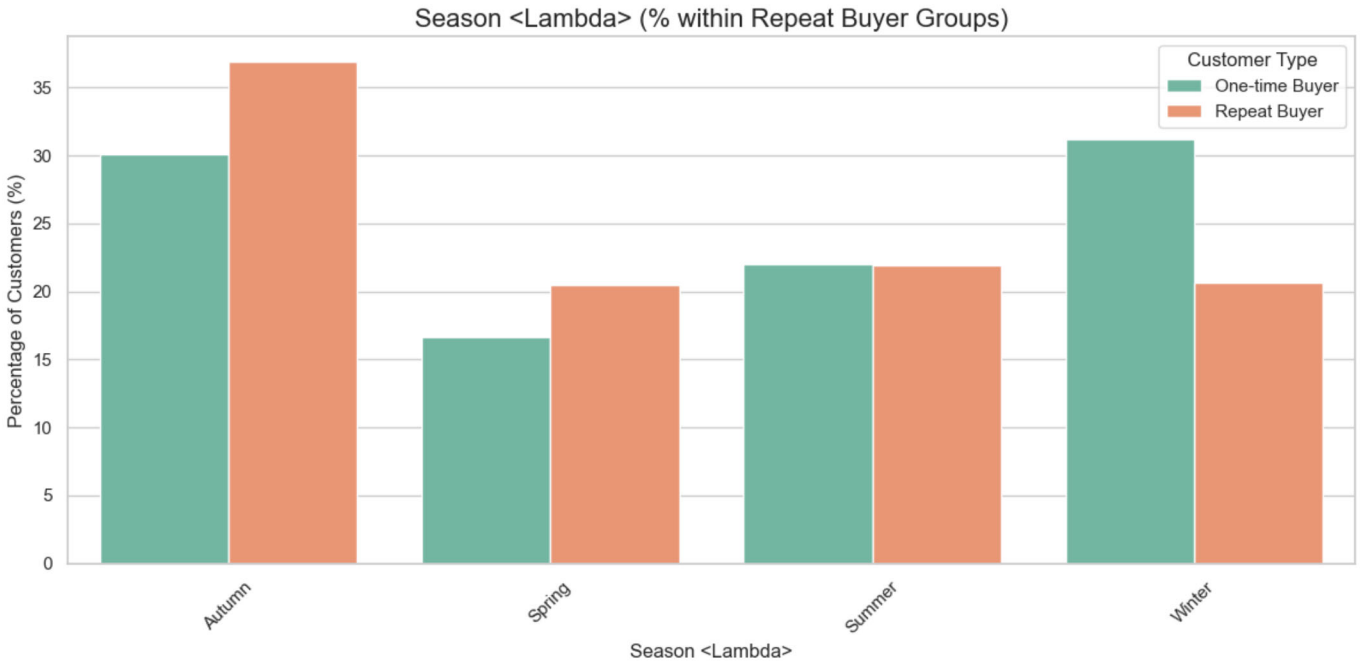
```
    repeat_coords,
    radius=8,
    blur=12,
    max_zoom=10,
    name='Repeat Buyers'
).add_to(m)

# adding toggling
folium.LayerControl().add_to(m)

m
```

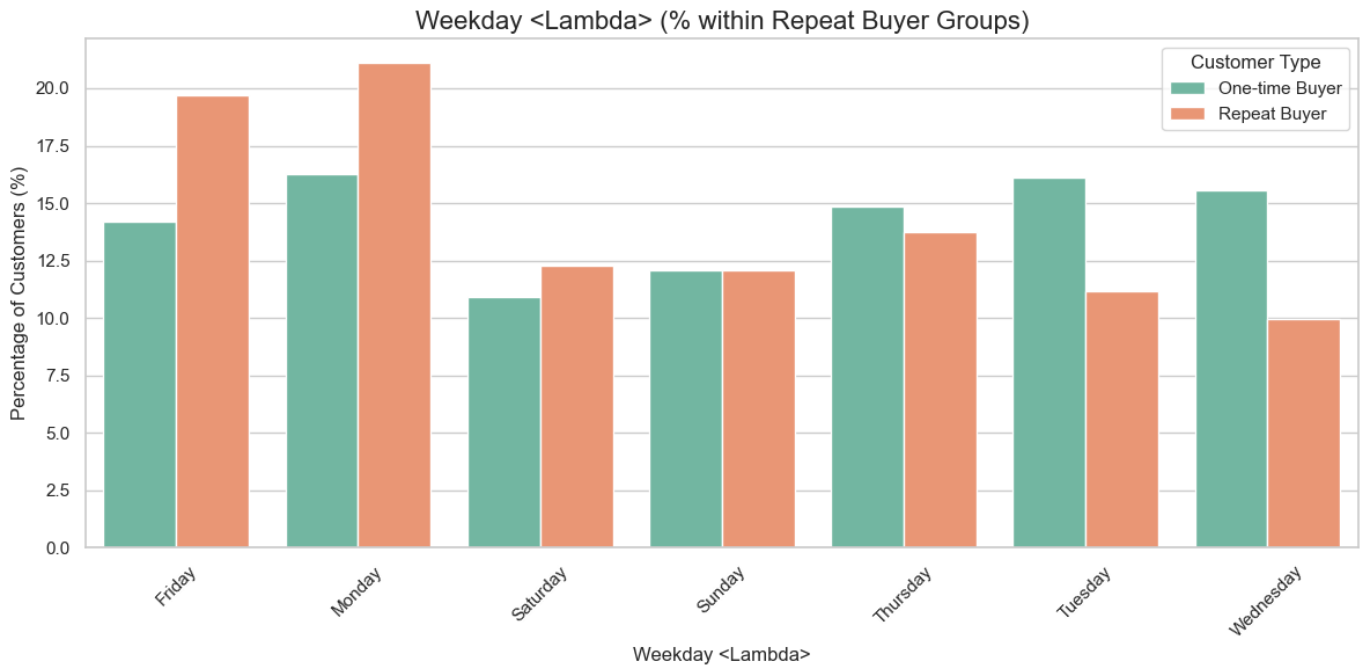
Out[45]:

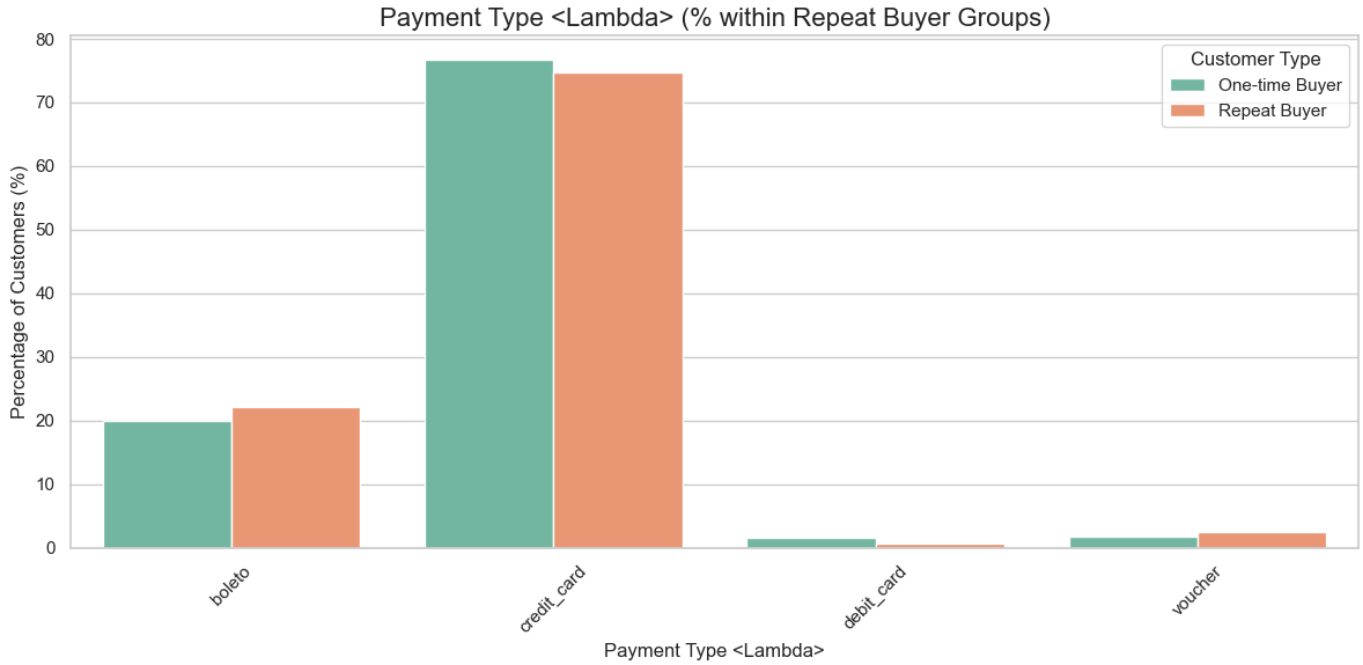


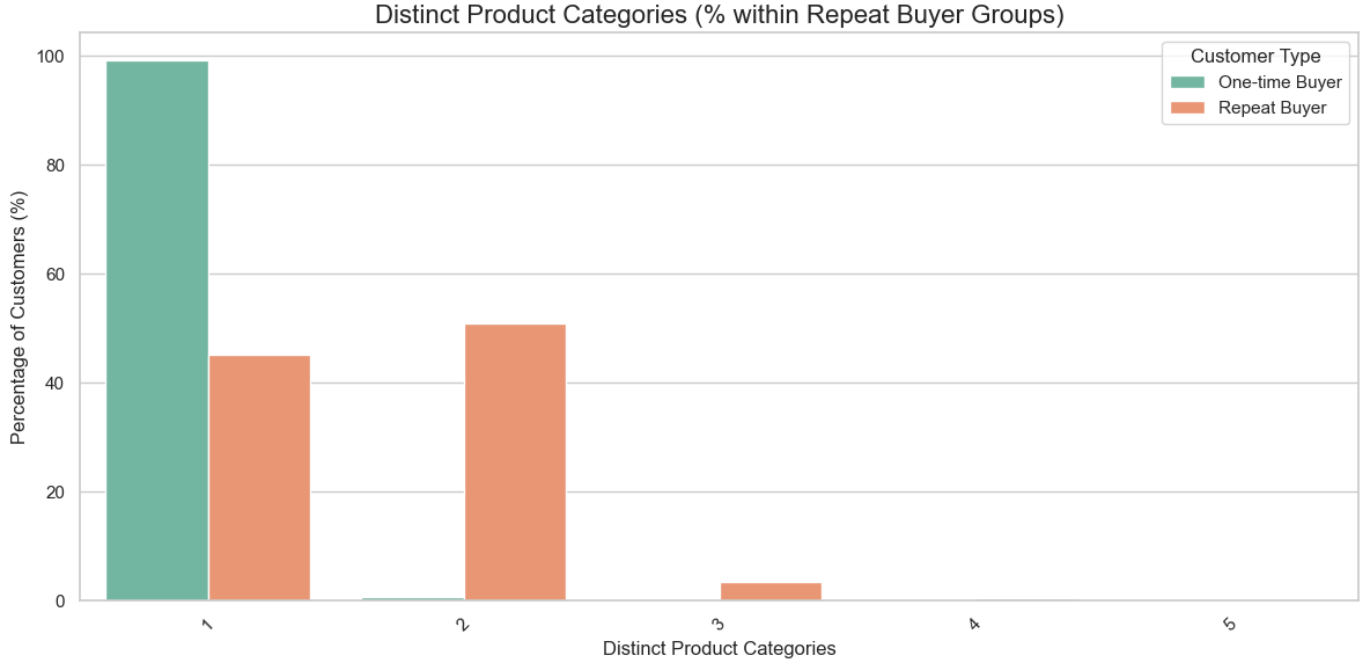
In [46]:

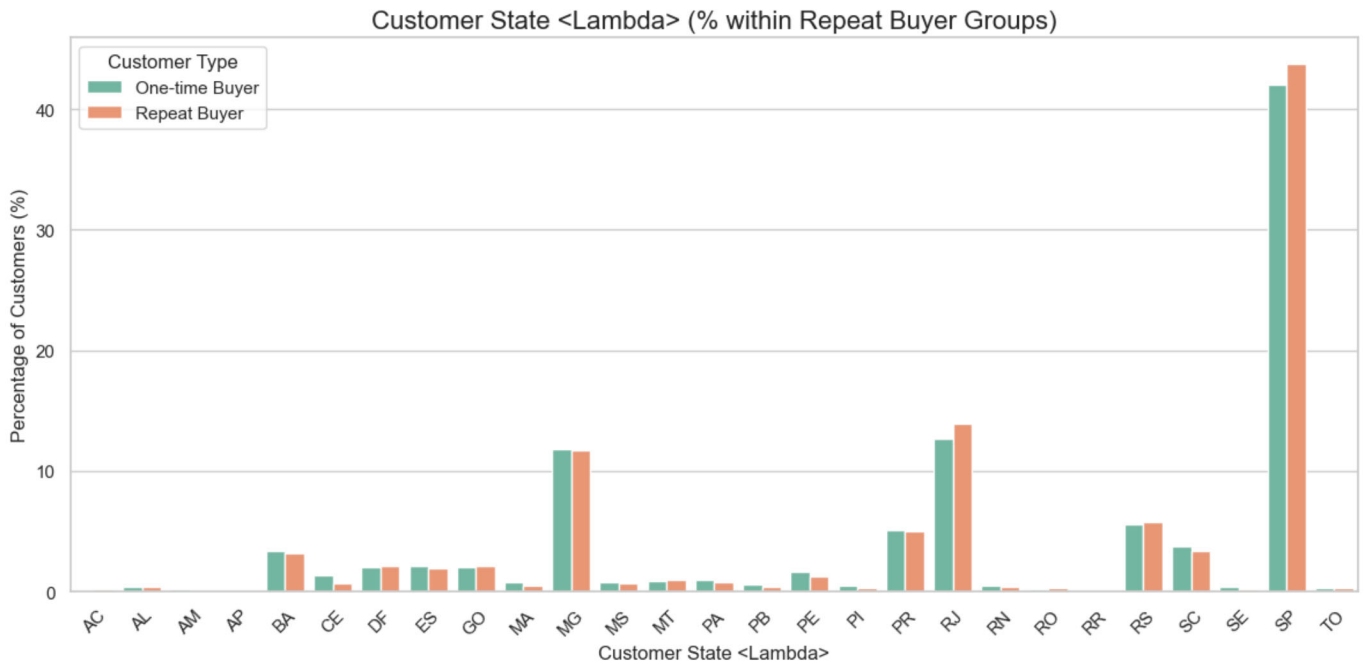
```
plot_feature_target_distribution(final_df, 'season_<lambda>', 'repeat_buyer')
plot_feature_target_distribution(final_df, 'weekday_<lambda>', 'repeat_buyer')
plot_feature_target_distribution(final_df, 'payment_type_<lambda>', 'repeat_buyer')
plot_feature_target_distribution(final_df, 'distinct_product_categories', 'repeat_buyer')
plot_feature_target_distribution(final_df, 'customer_state_<lambda>', 'repeat_buyer')
```




Weekday <Lambda> (% within Repeat Buyer Groups)
Customer Type
One-time Buyer
Repeat Buyer
Percentage of Customers (%)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
Weekday <Lambda>

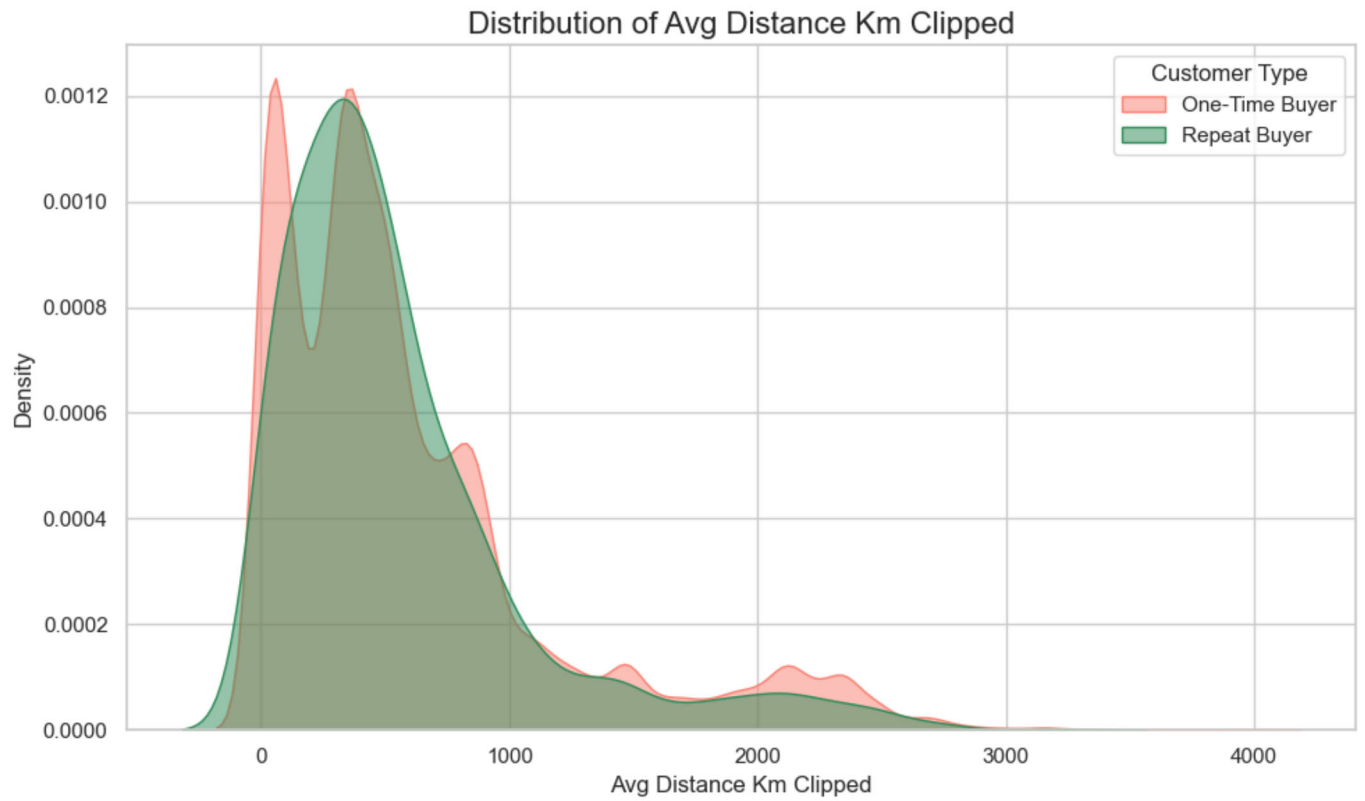

Payment Type <Lambda> (% within Repeat Buyer Groups)
Customer Type
One-time Buyer
Repeat Buyer
Percentage of Customers (%)
0
10
20
30
40
50
60
70
80
boleto
credit_card
debit_card
voucher
Payment Type <Lambda>

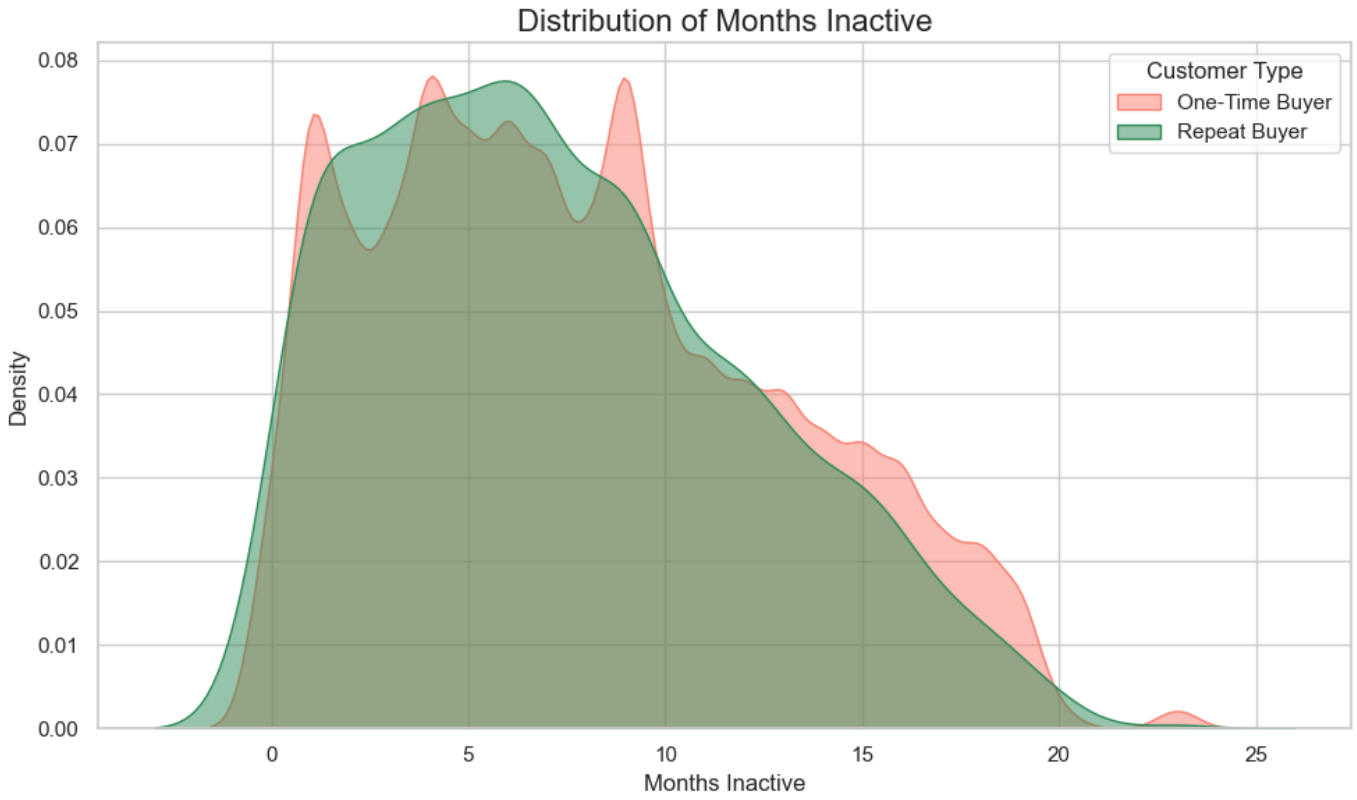

Distinct Product Categories (% within Repeat Buyer Groups)
Customer Type
One-time Buyer
Repeat Buyer
Percentage of Customers (%)
0
20
40
60
80
100
1
2
3
4
5
Distinct Product Categories

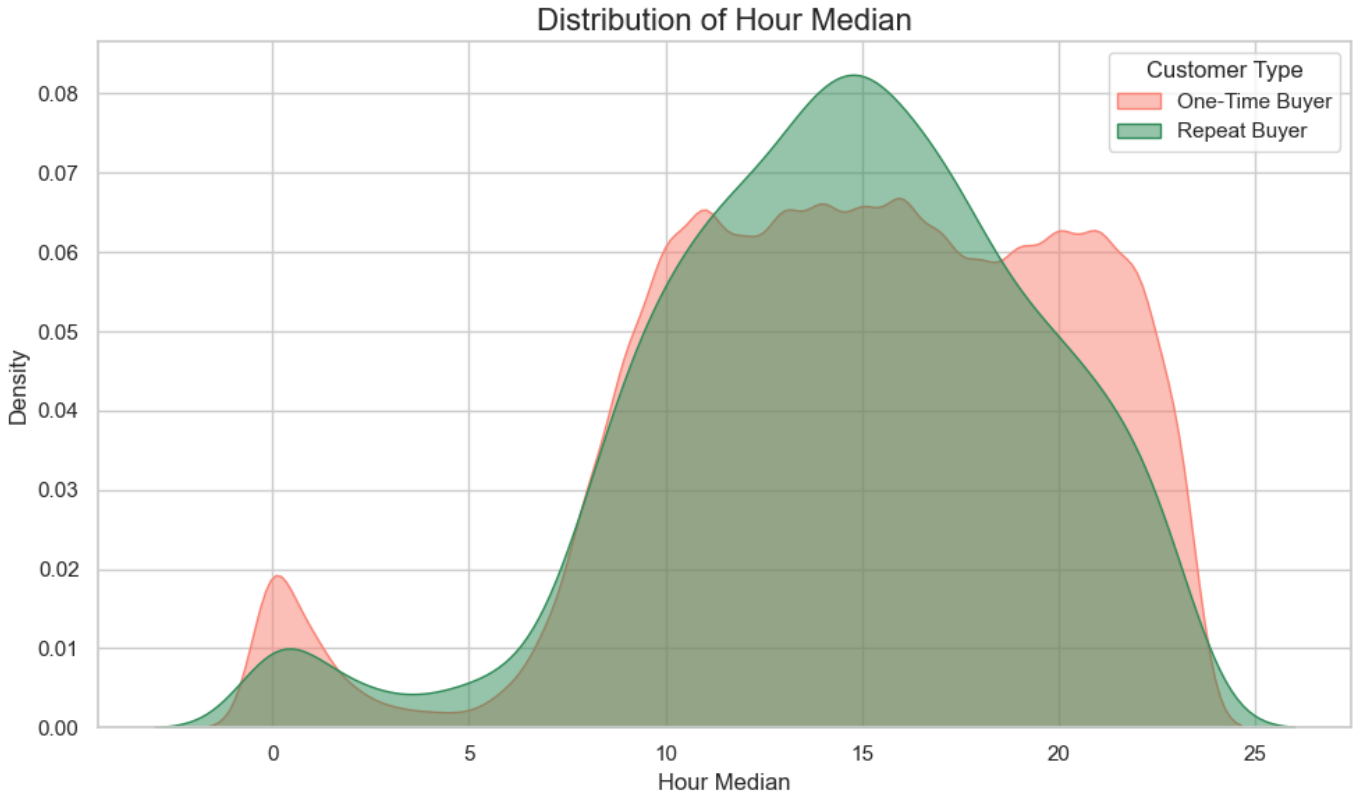


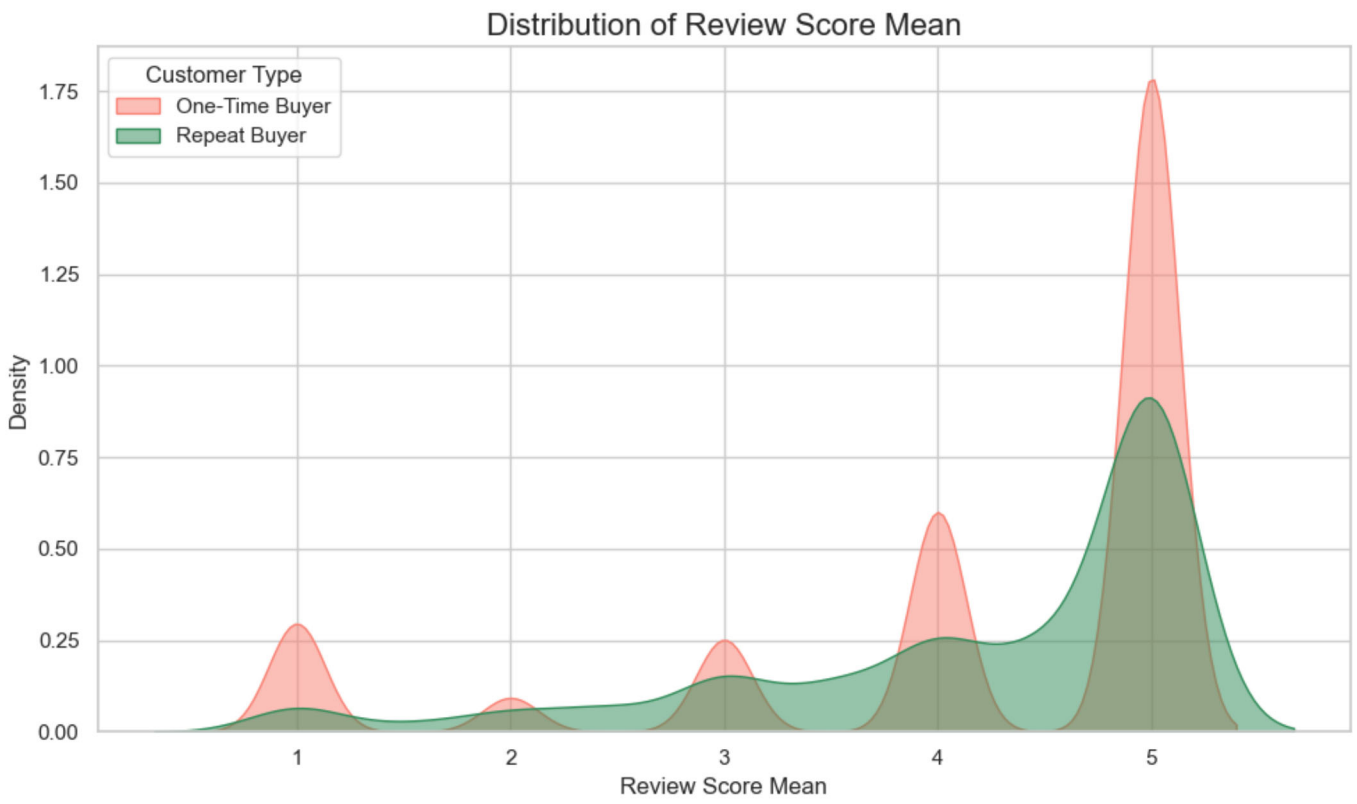
In [47]:
```
plot_kde_by_class_target(final_df, 'avg_distance_km', 'repeat_buyer', ["One-Time Buyer", "Repe
plot_kde_by_class_target(final_df, 'months_inactive', 'repeat_buyer', ["One-Time Buyer", "Repe
plot_kde_by_class_target(final_df, 'hour_median', 'repeat_buyer', ["One-Time Buyer", "Repeat B
plot_kde_by_class_target(final_df, 'review_score_mean', 'repeat_buyer', ["One-Time Buyer", "Re
```

```
Clipped 4 values > 4000 in 'avg_distance_km'.
```




Distribution of Months Inactive
Customer Type
One-Time Buyer
Repeat Buyer
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0
5
10
15
20
25
Months Inactive

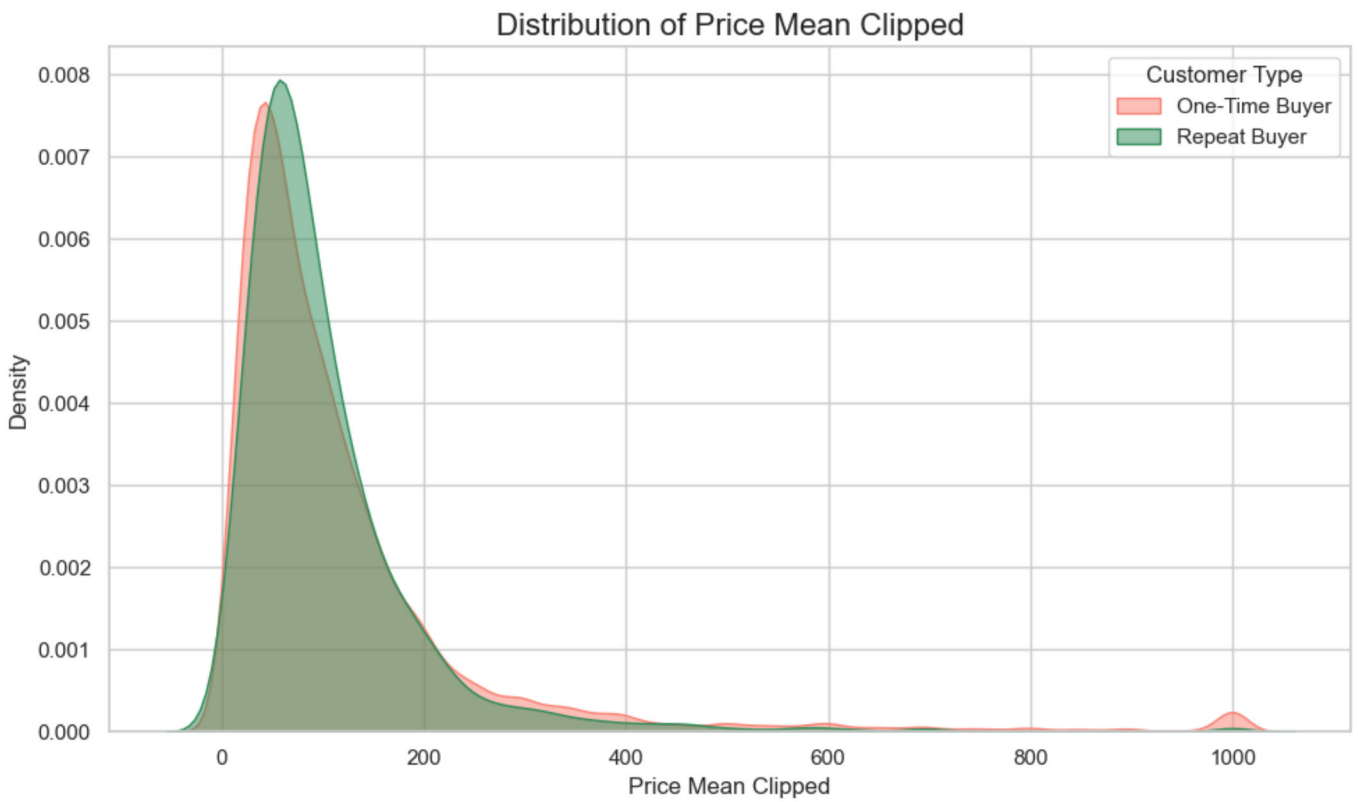

Distribution of Hour Median
Customer Type
One-Time Buyer
Repeat Buyer
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0
5
10
15
20
25
Hour Median



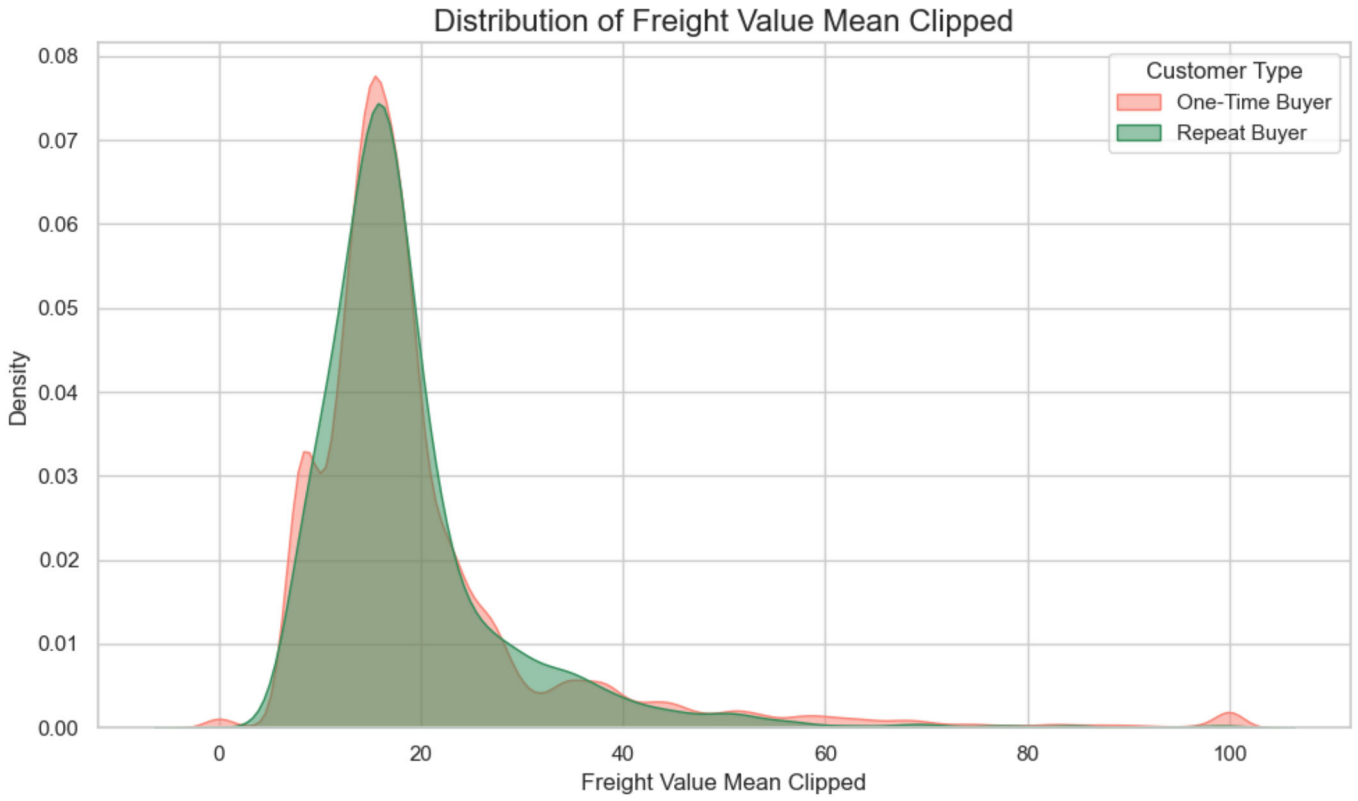
In [48]:
```
# Graphs of all the money-related features
plot_kde_by_class_target(final_df, 'price_sum', 'repeat_buyer', ["One-Time Buyer", "Repeat Buy
plot_kde_by_class_target(final_df, 'price_mean', 'repeat_buyer', ["One-Time Buyer", "Repeat Bu
plot_kde_by_class_target(final_df, 'freight_value_mean', 'repeat_buyer', ["One-Time Buyer", "F
```

```
Clipped 193 values > 2000 in 'price_sum'.
```


```
Clipped 739 values > 1000 in 'price_mean'.
```

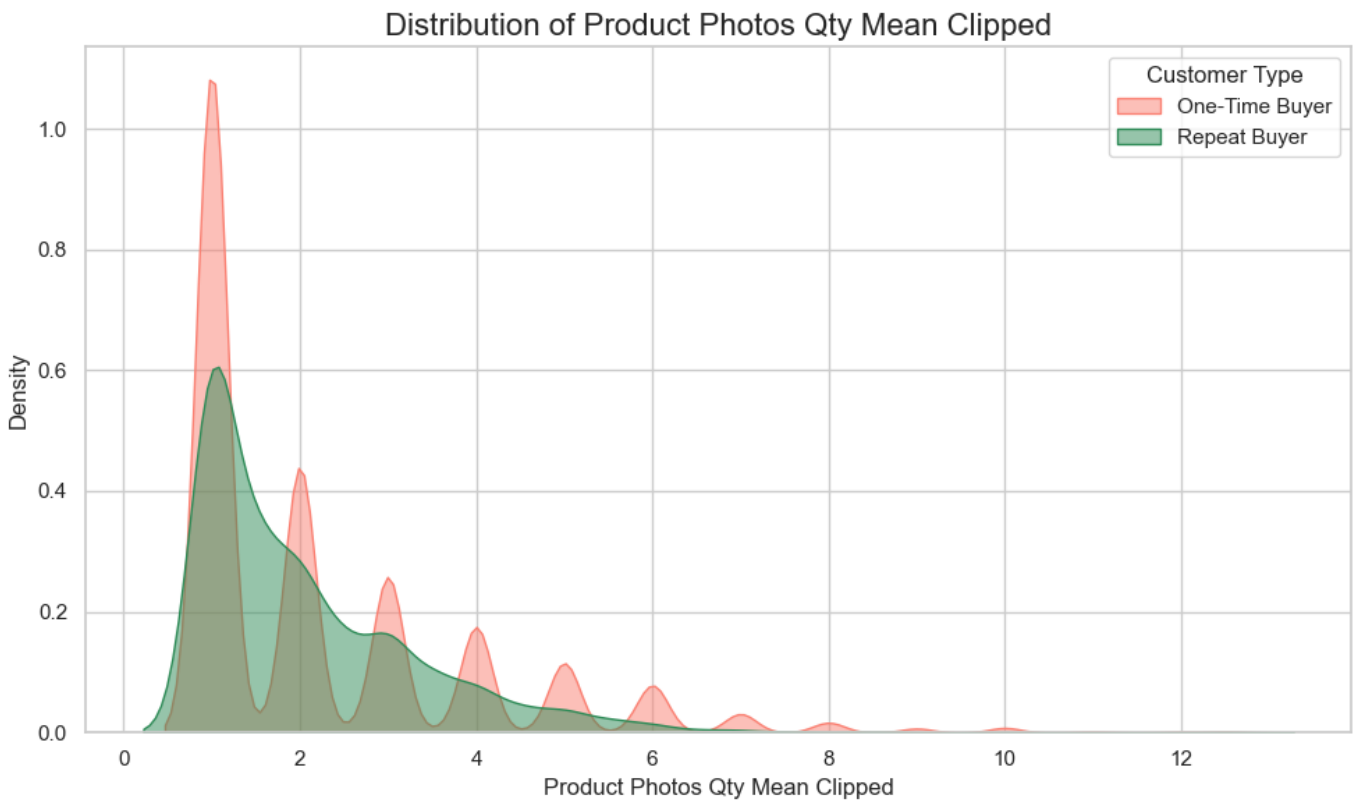



```
Clipped 551 values > 100 in 'freight_value_mean'.
```

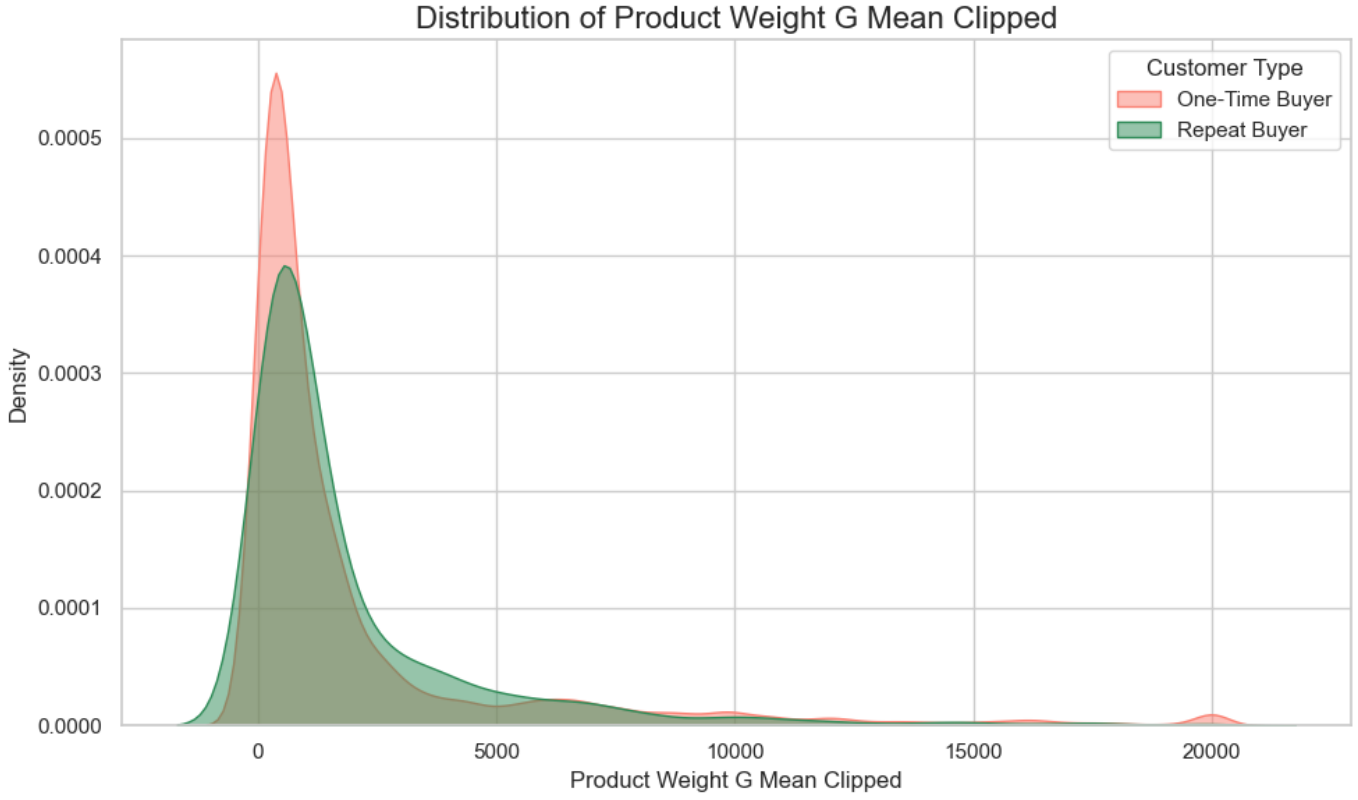


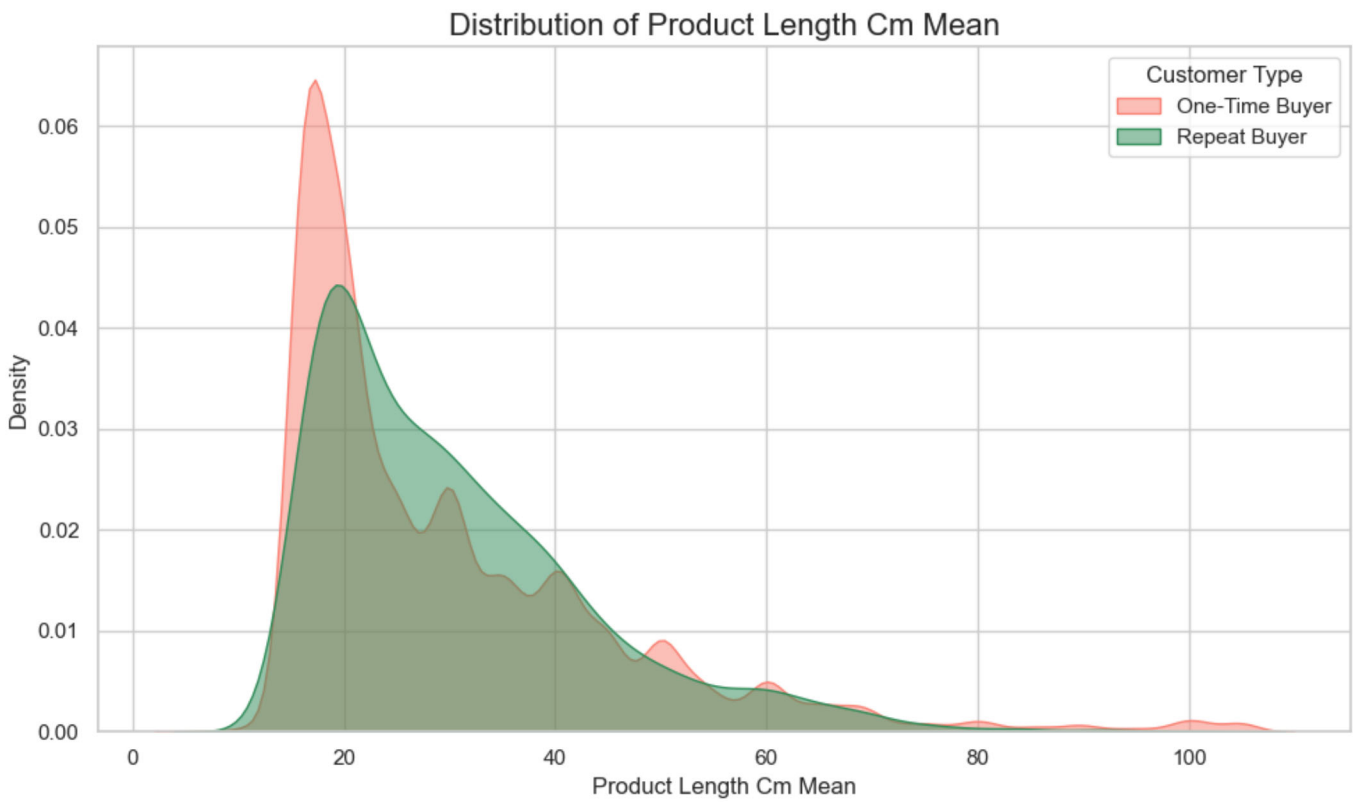
In [49]:
```
# Graphs of all the product-related features
plot_kde_by_class_target(final_df, 'product_photos_qty_mean', 'repeat_buyer', ["One-Time Buyer
plot_kde_by_class_target(final_df, 'product_weight_g_mean', 'repeat_buyer', ["One-Time Buyer",
plot_kde_by_class_target(final_df, 'product_length_cm_mean', 'repeat_buyer', ["One-Time Buyer'
plot_kde_by_class_target(final_df, 'product_height_cm_mean', 'repeat_buyer', ["One-Time Buyer'
```

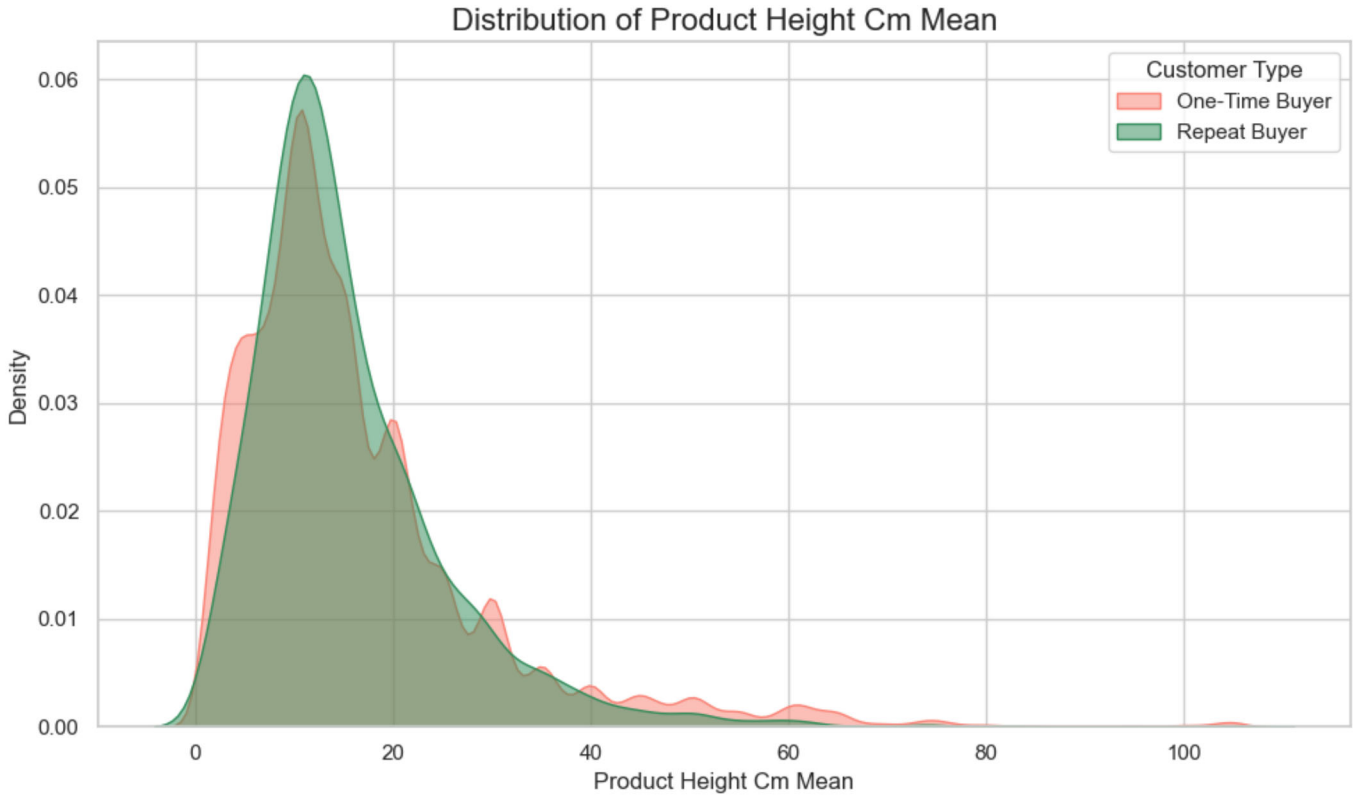
```
Clipped 54 values > 12.5 in 'product_photos_qty_mean'.
```




```
Clipped 719 values > 20000 in 'product_weight_g_mean'.
```

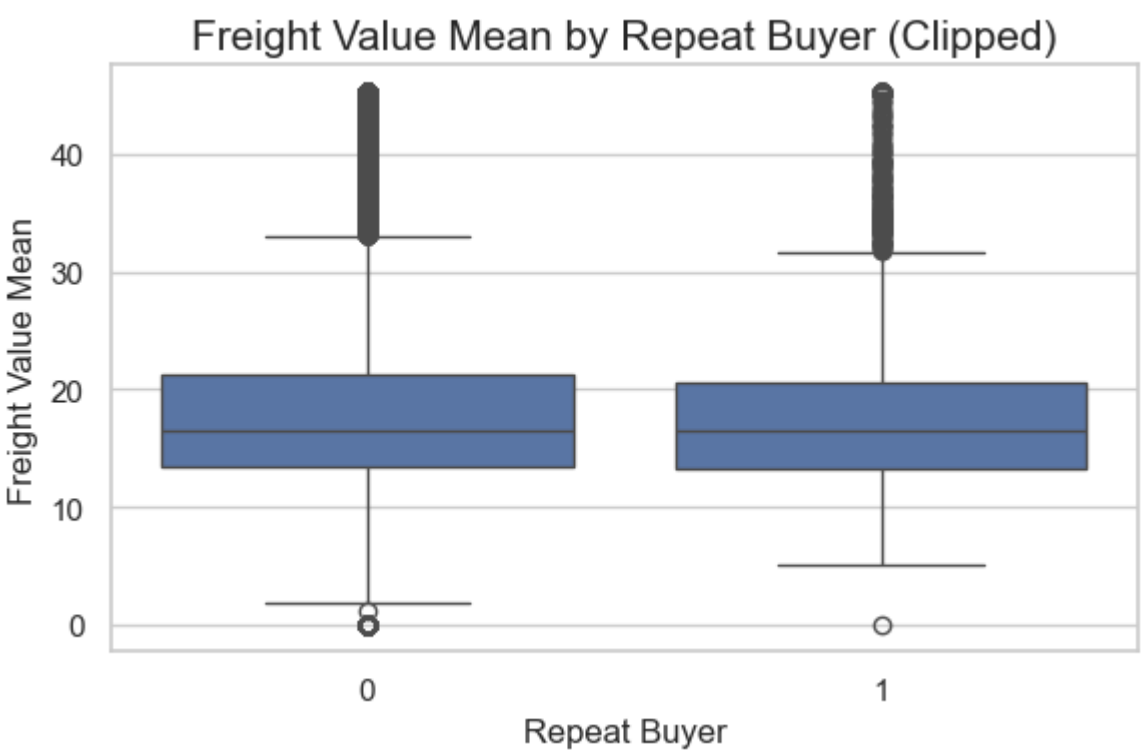




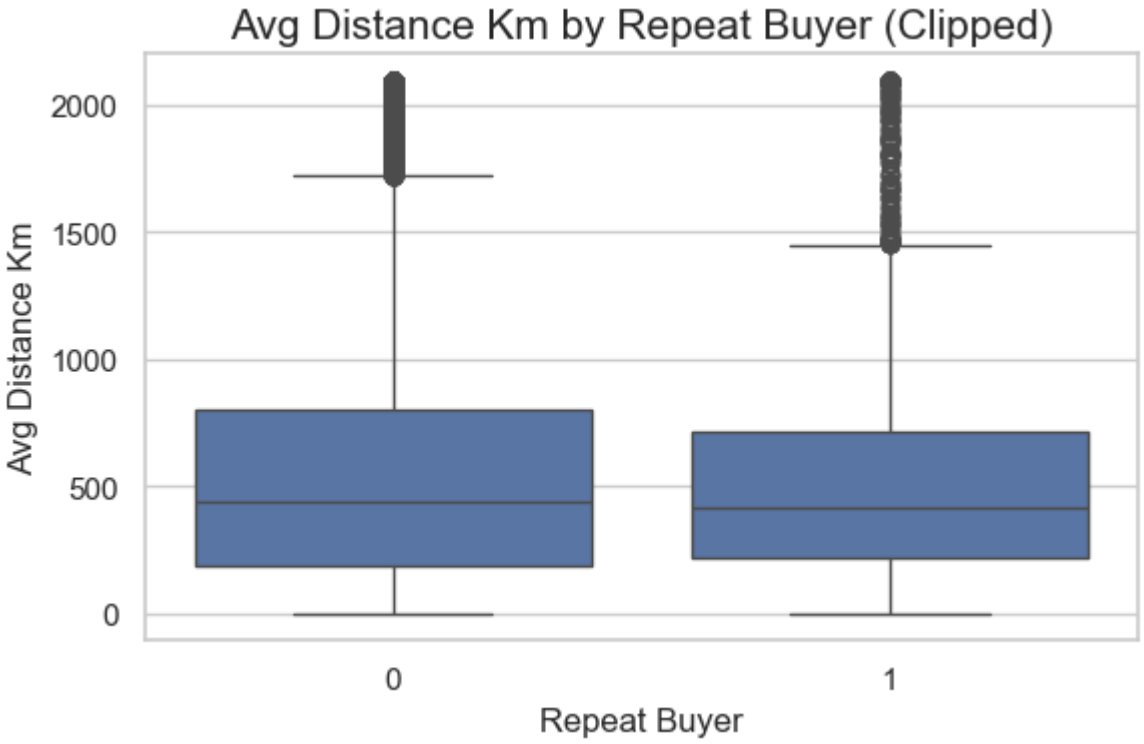


In [50]:
```
plot_boxplot_by_target(final_df, "freight_value_mean", "repeat_buyer")
plot_boxplot_by_target(final_df, "avg_distance_km", "repeat_buyer")
plot_boxplot_by_target(final_df, "months_inactive", "repeat_buyer")
plot_boxplot_by_target(final_df, "hour_median", "repeat_buyer")
plot_boxplot_by_target(final_df, "review_score_mean", "repeat_buyer")
```

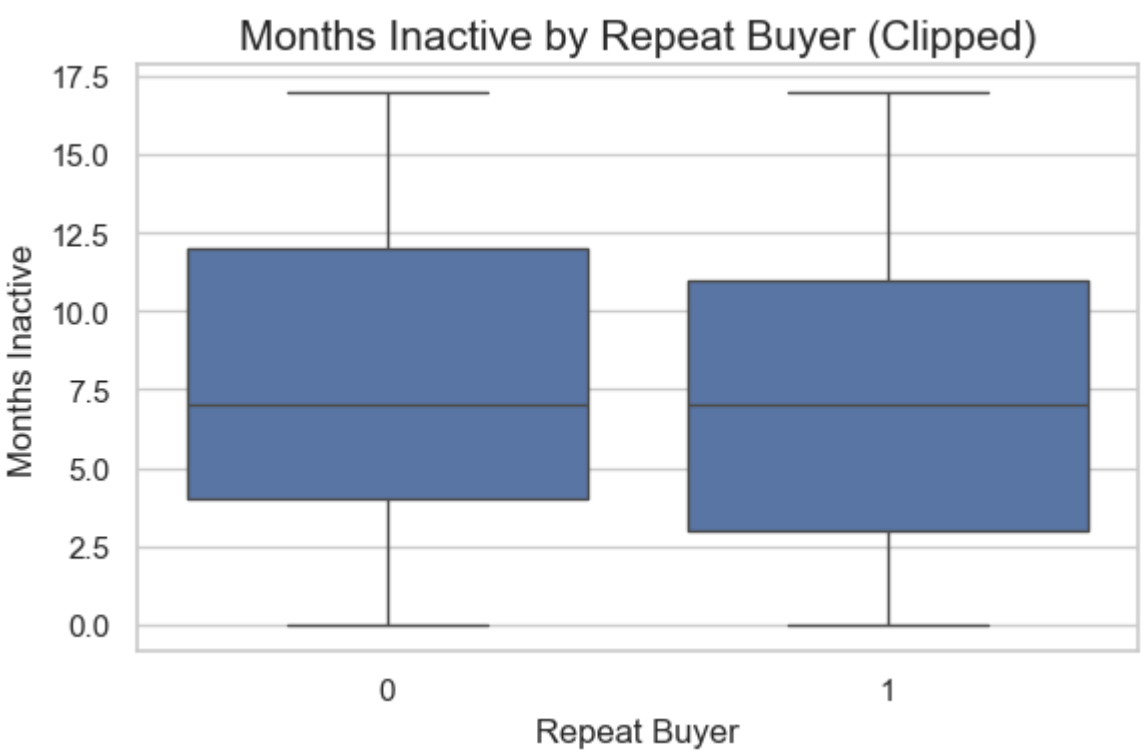
```
Note: 4549 values clipped at 95th percentile
```



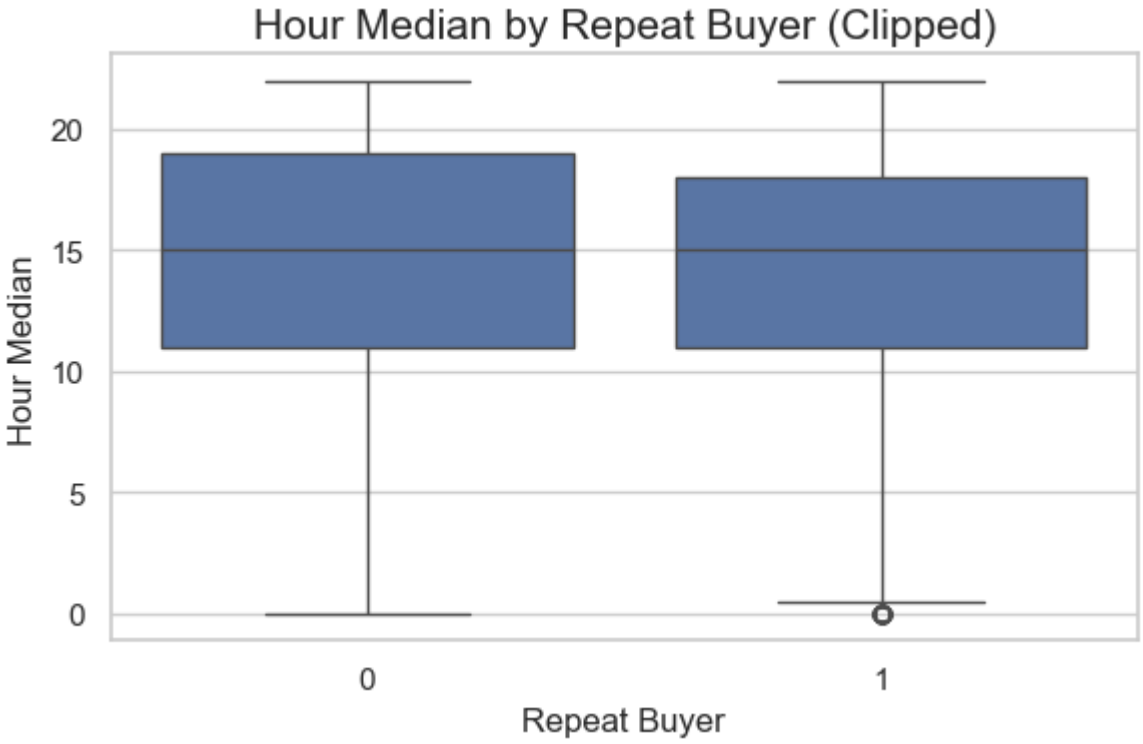
Note: 4551 values clipped at 95th percentile

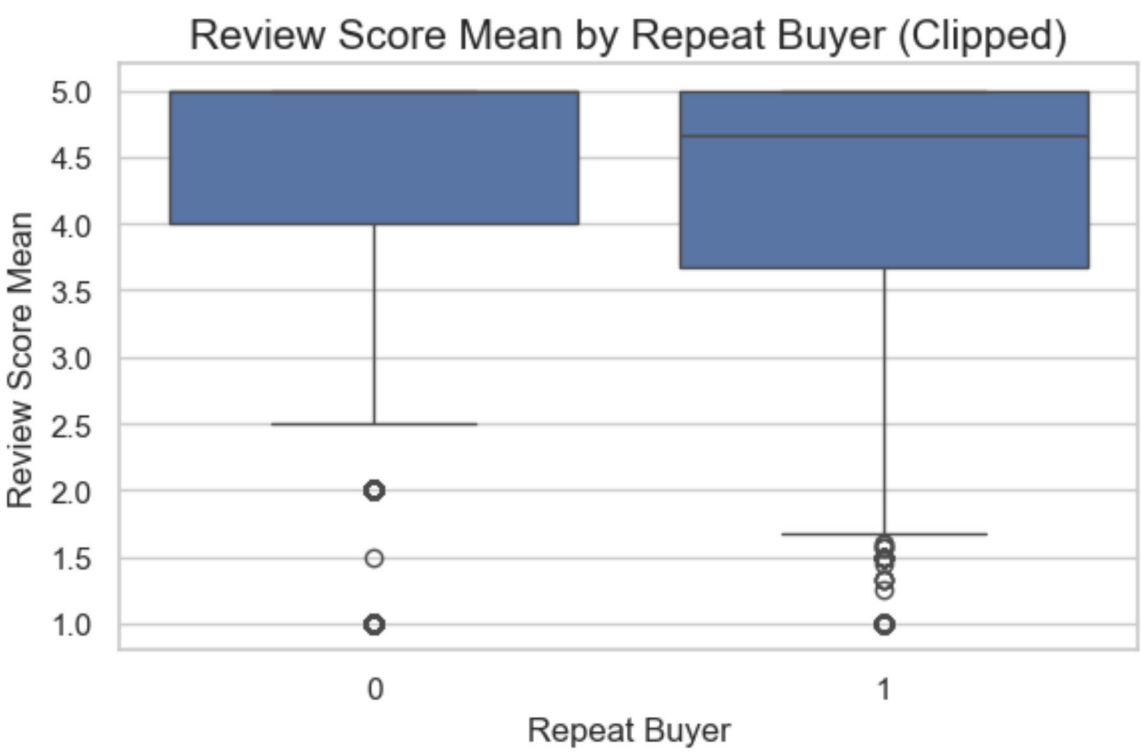


Note: 4061 values clipped at 95th percentile



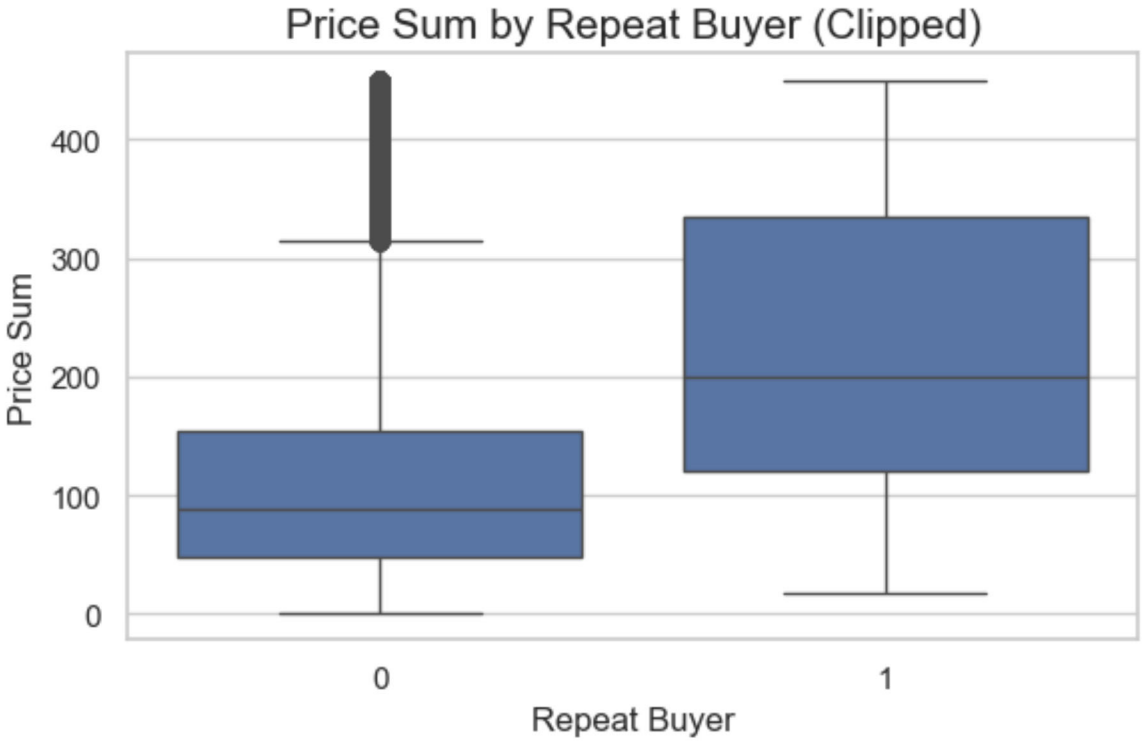
Note: 3756 values clipped at 95th percentile



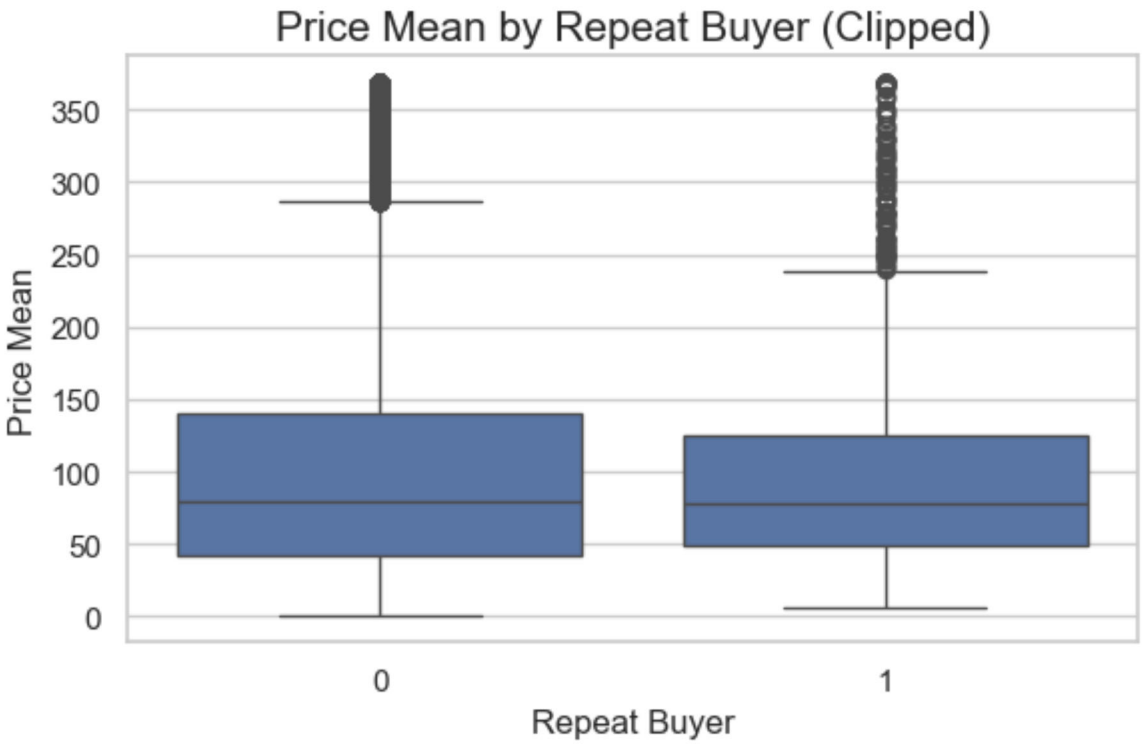


In [51]:
```
# Graphs of all the money-related features
plot_boxplot_by_target(final_df, "price_sum", "repeat_buyer")
plot_boxplot_by_target(final_df, "price_mean", "repeat_buyer")
plot_boxplot_by_target(final_df, "freight_value_mean", "repeat_buyer")
```

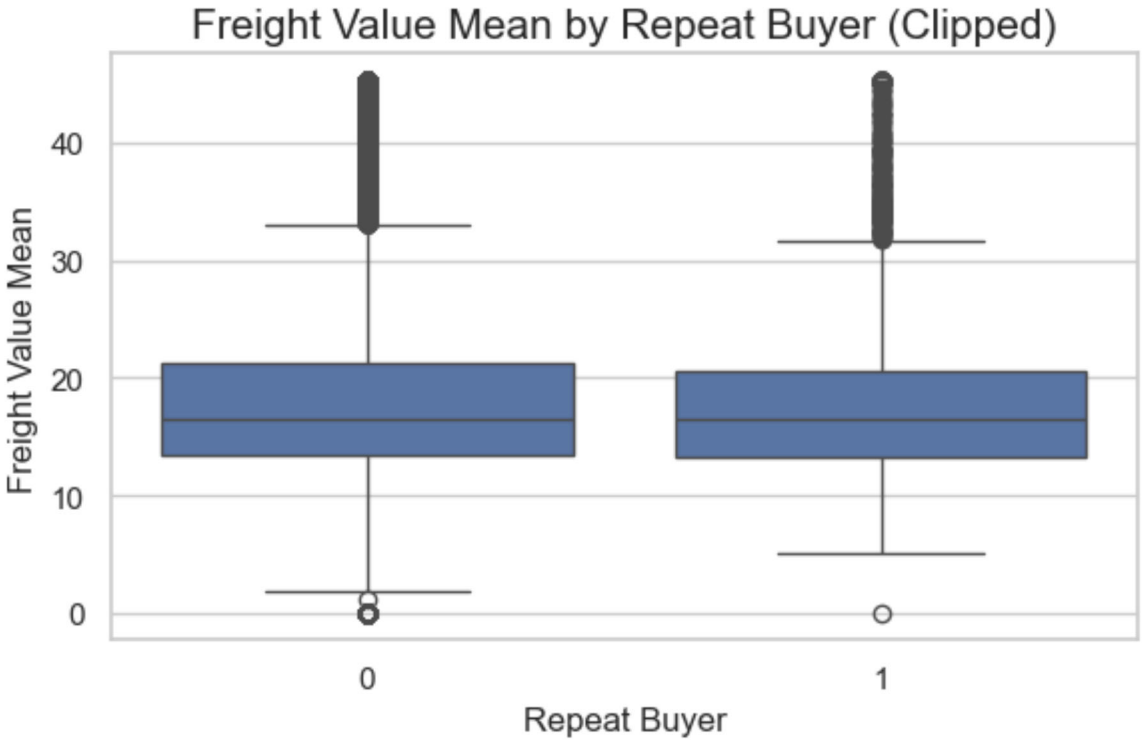
```
Note: 4542 values clipped at 95th percentile
```




```
Note: 4550 values clipped at 95th percentile
```

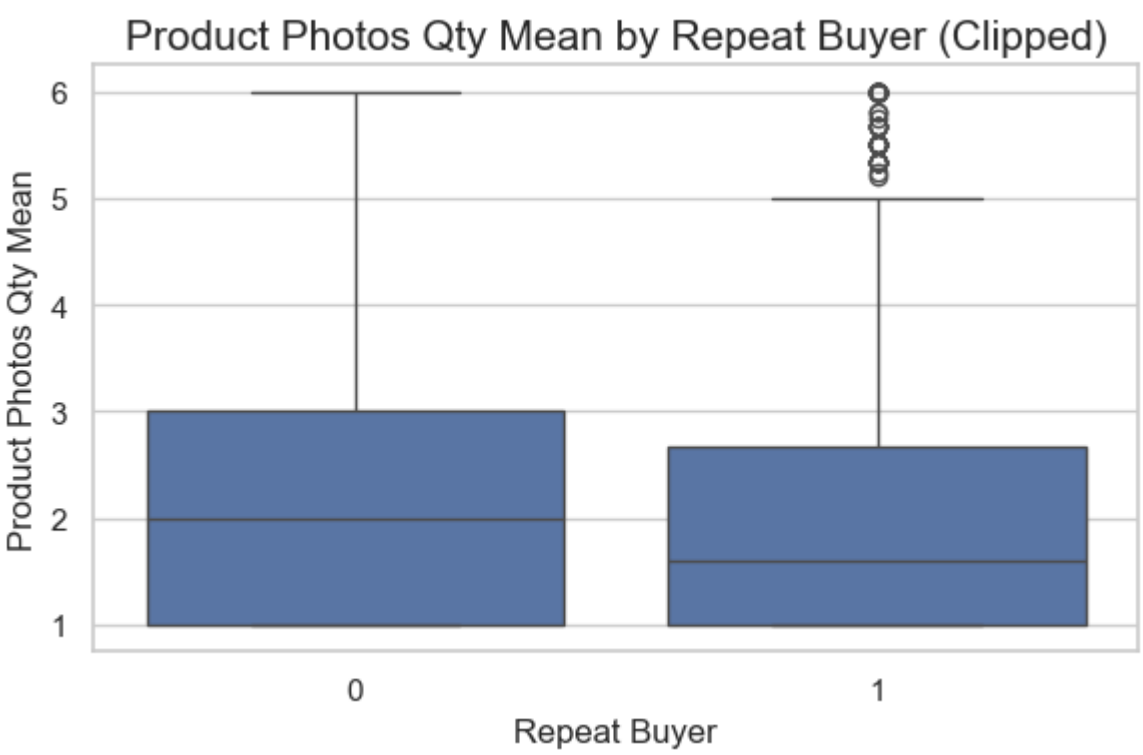


Note: 4549 values clipped at 95th percentile

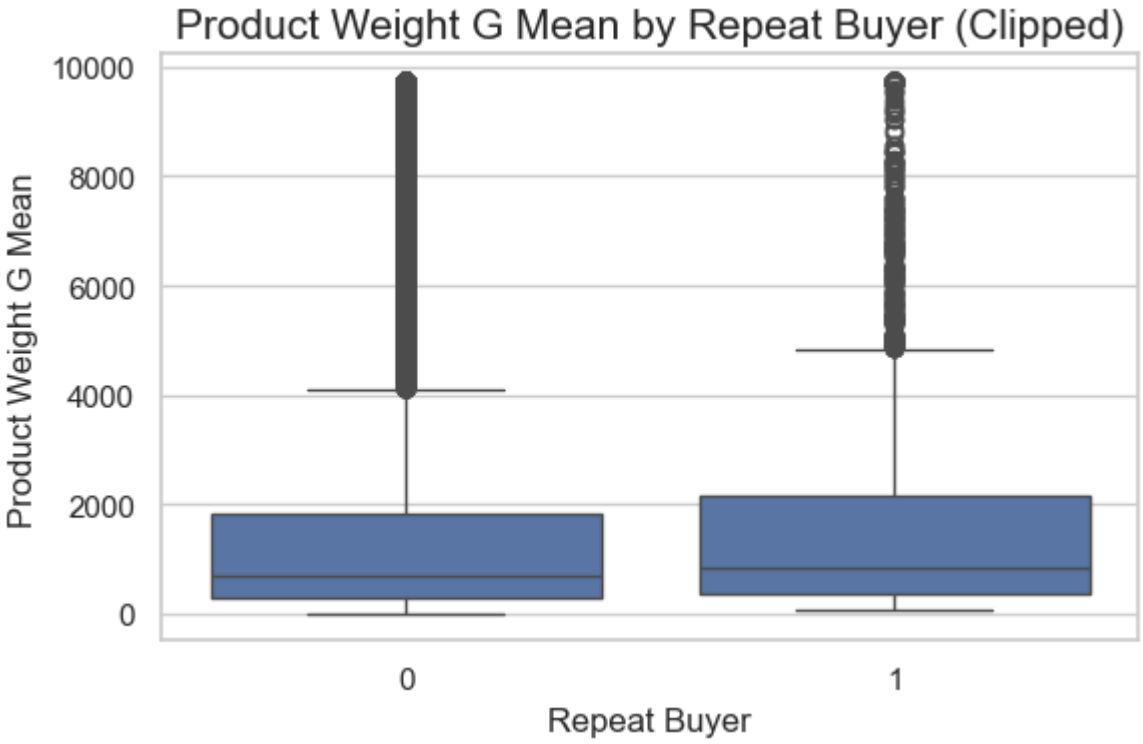


In [52]:
```
# Graphs of all the product-related features
plot_boxplot_by_target(final_df, 'product_photos_qty_mean', "repeat_buyer")
plot_boxplot_by_target(final_df, 'product_weight_g_mean', "repeat_buyer")
plot_boxplot_by_target(final_df, 'product_length_cm_mean', "repeat_buyer")
plot_boxplot_by_target(final_df, 'product_height_cm_mean', "repeat_buyer")
```

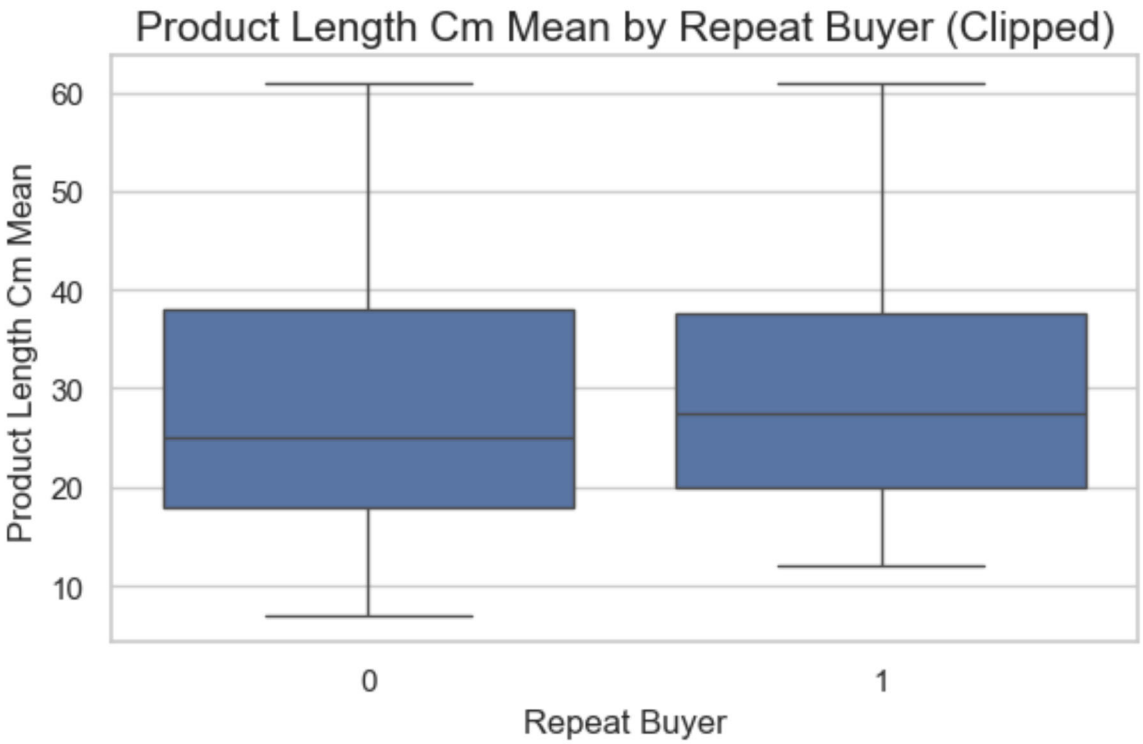
Note: 2585 values clipped at 95th percentile



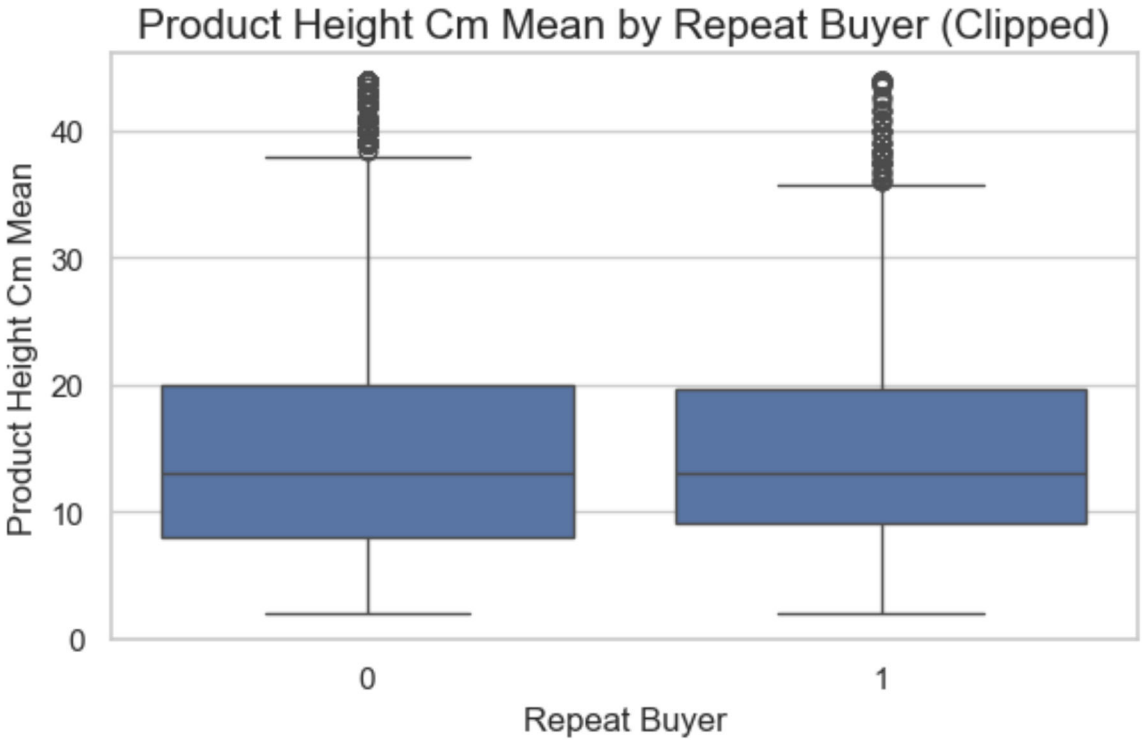
Note: 4407 values clipped at 95th percentile



Note: 4448 values clipped at 95th percentile




```
Note: 4421 values clipped at 95th percentile
```



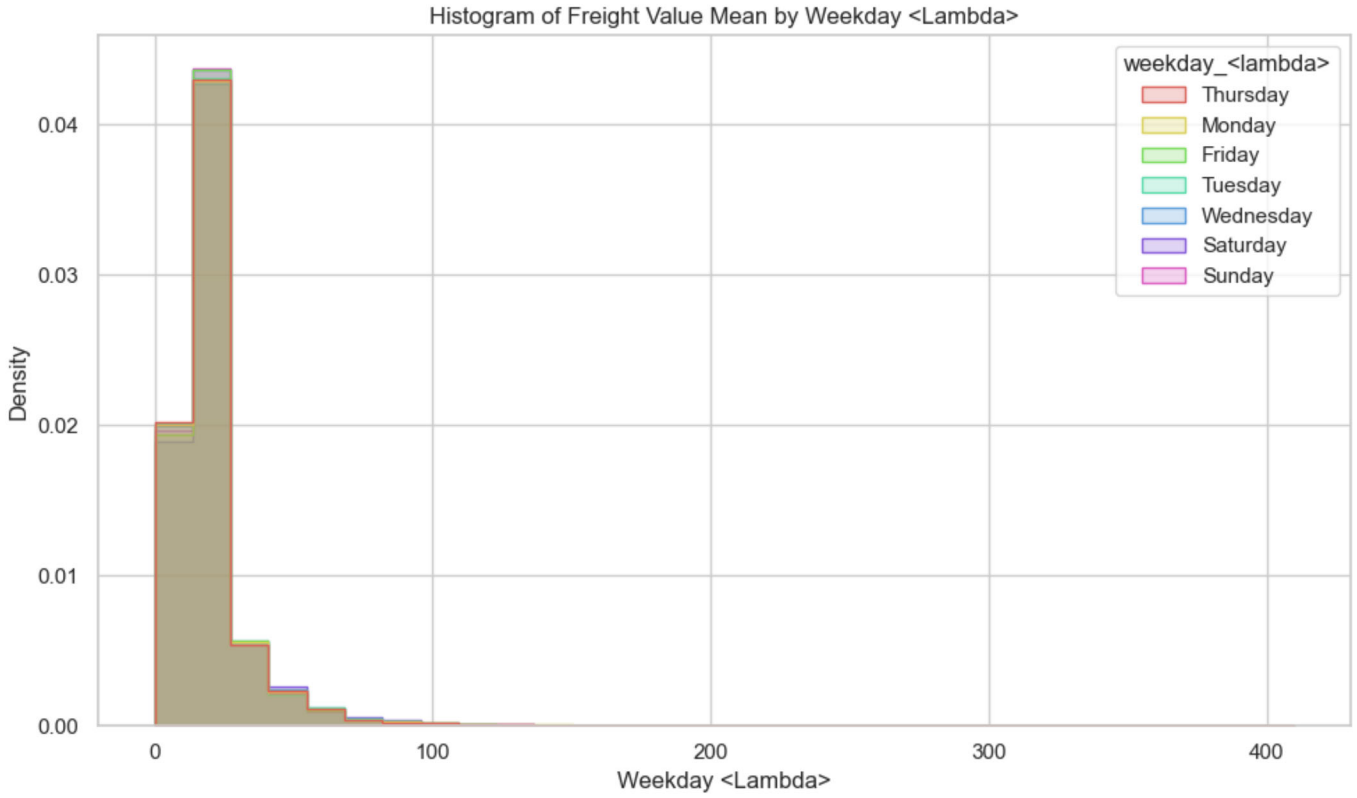
EDA: Freight Value Mean

Visualizing various data on Freight Value Mean for model building and to gather insights.

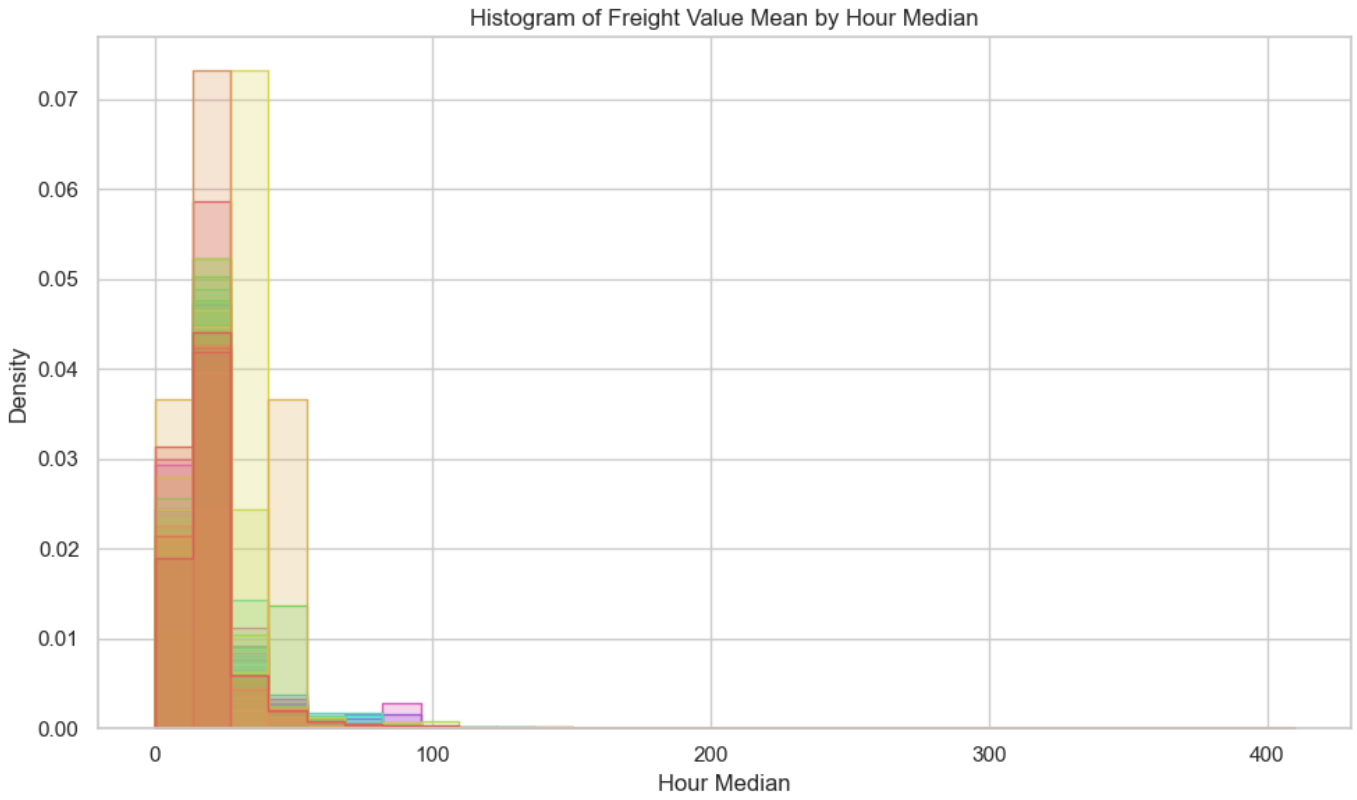
In [53]:
```
print(final_df.columns)
```

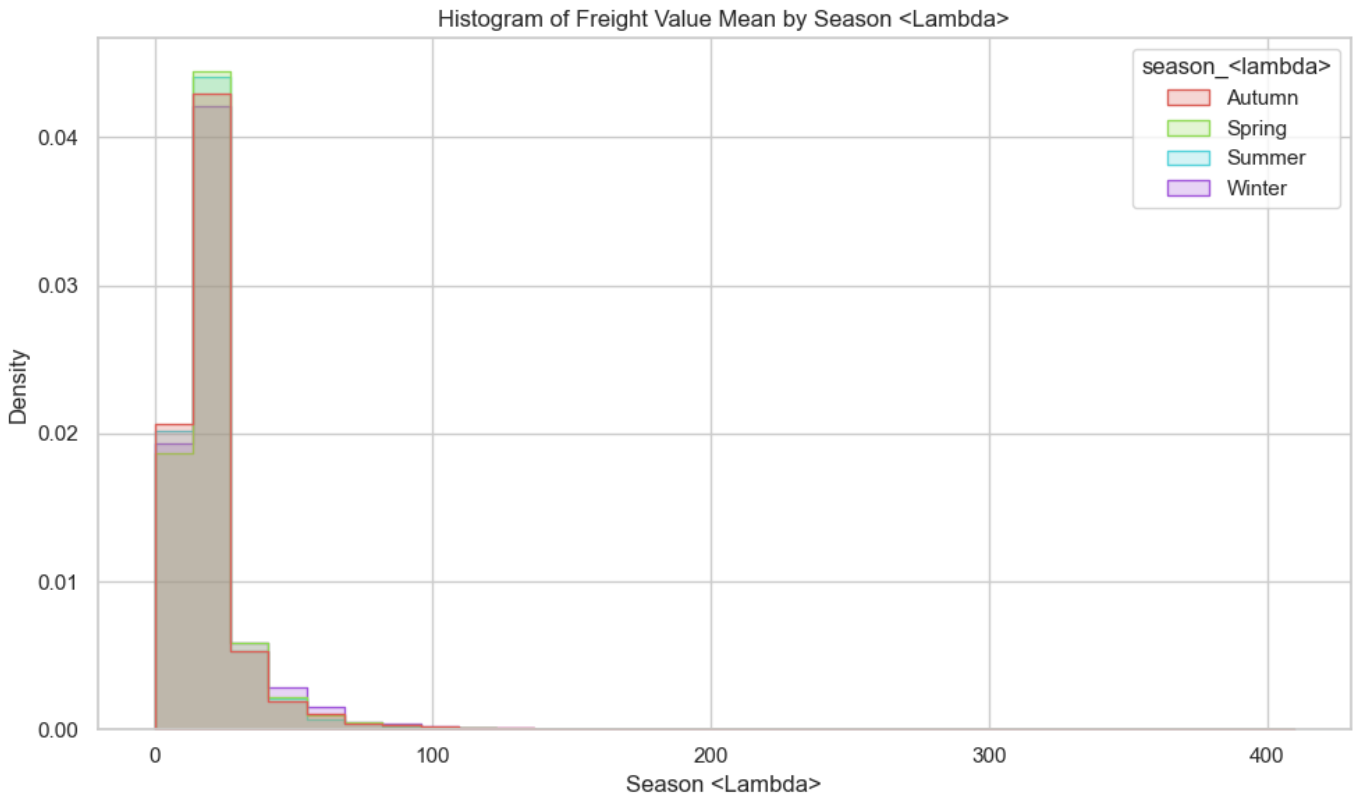
```
Index(['customer_unique_id', 'weekday_<lambda>', 'hour_median', 'price_sum',
       'price_mean', 'freight_value_mean', 'payment_type_<lambda>',
       'payment_installments_mean', 'payment_value_sum', 'payment_value_mean',
       'review_score_mean', 'product_photos_qty_mean', 'product_weight_g_mean',
       'product_length_cm_mean', 'product_height_cm_mean',
       'product_width_cm_mean', 'customer_city_<lambda>',
       'customer_state_<lambda>', 'geolocation_lat_mean',
       'geolocation_lng_mean', 'season_<lambda>', 'delivery_time_mean',
       'product_category_name_english', 'order_count', 'repeat_buyer',
       'distinct_product_categories', 'months_inactive', 'avg_distance_km',
       'volume_mean', 'cost_volume', 'density_mean', 'cost_weight', 'lh_ratio',
       'lw_ratio', 'hw_ratio'],
      dtype='object')
```

In [54]:
```
plot_histogram_by_feature(final_df, "weekday_<lambda>", "freight_value_mean")
plot_histogram_by_feature(final_df, "hour_median", "freight_value_mean")
plot_histogram_by_feature(final_df, "season_<lambda>", "freight_value_mean")
plot_histogram_by_feature(final_df, "customer_state_<lambda>", "freight_value_mean")
plot_histogram_by_feature(final_df, "distinct_product_categories", "freight_value_mean")
```

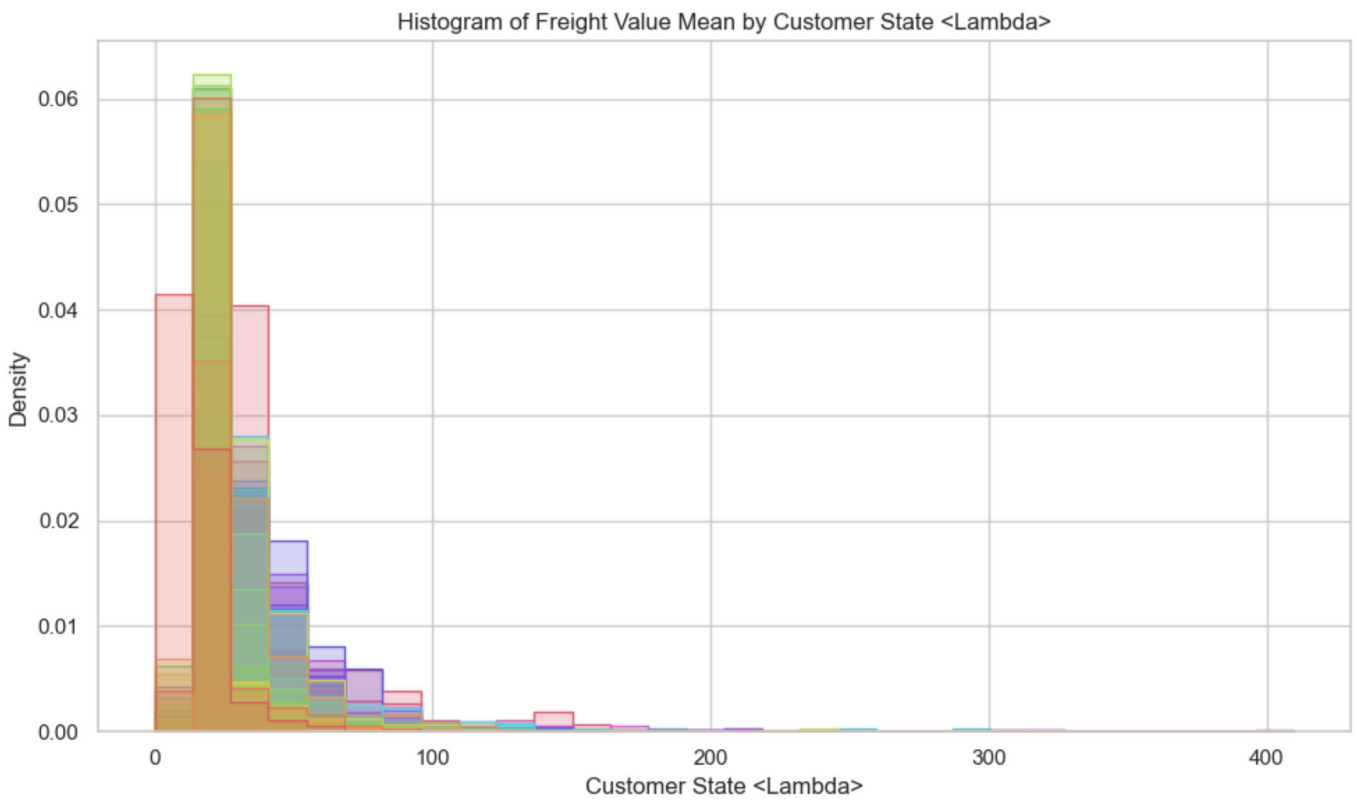


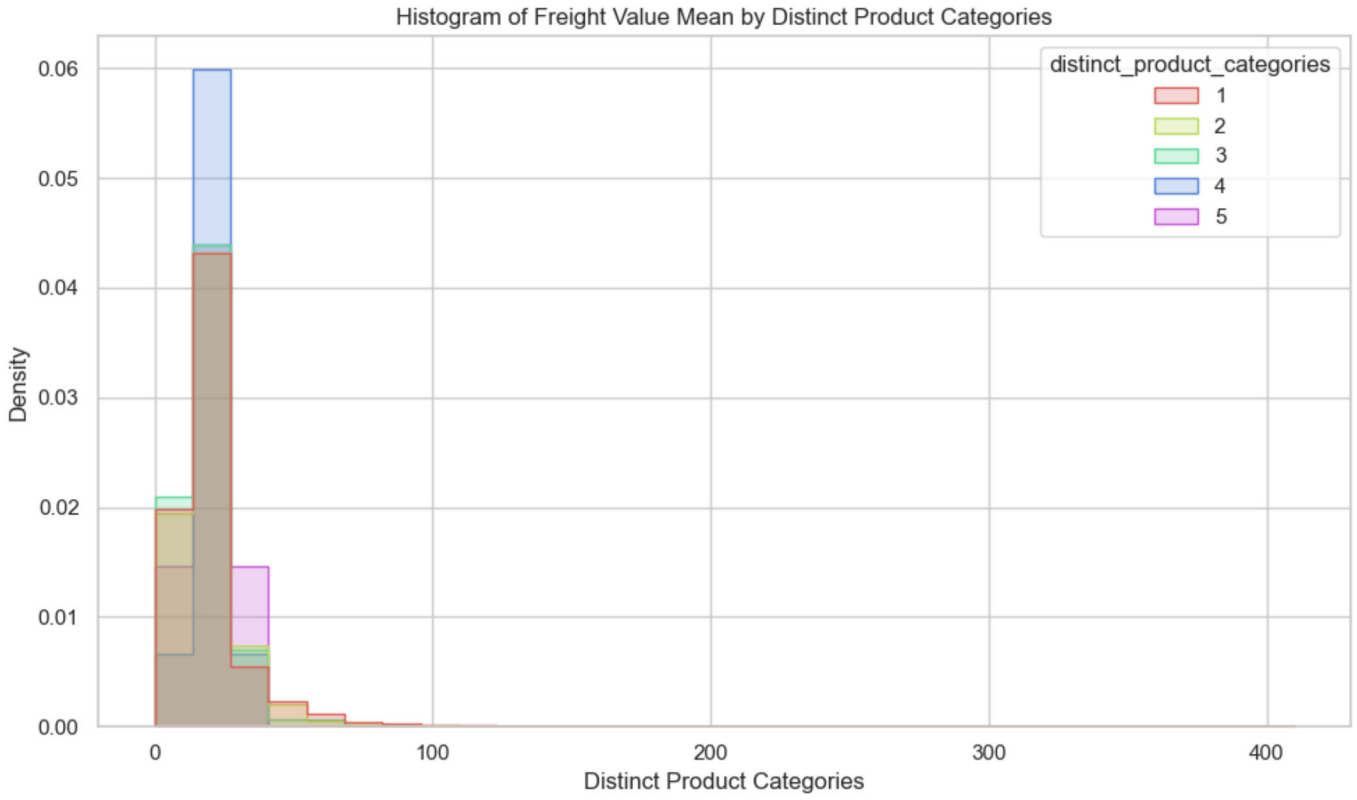

```
Too many labels:  47
```






```
Too many labels:  27
```

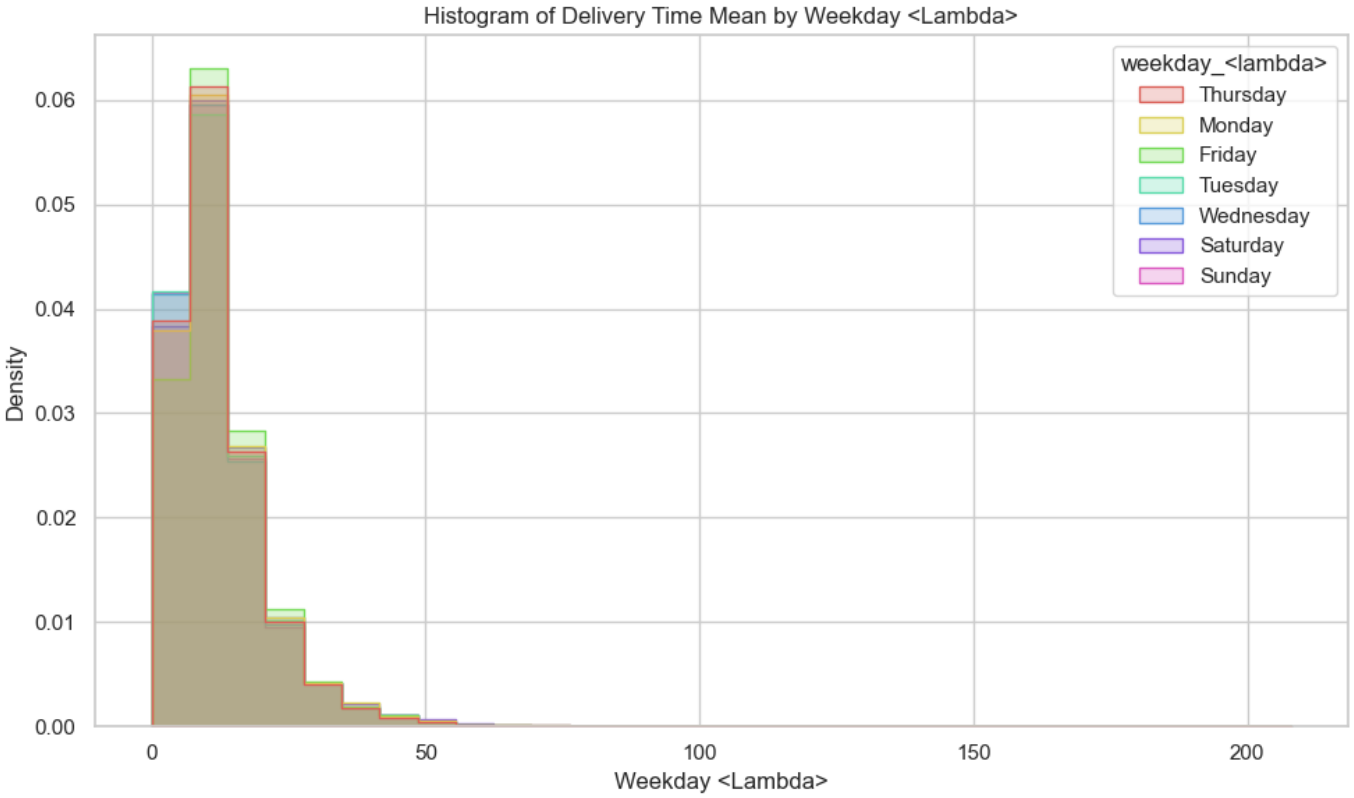




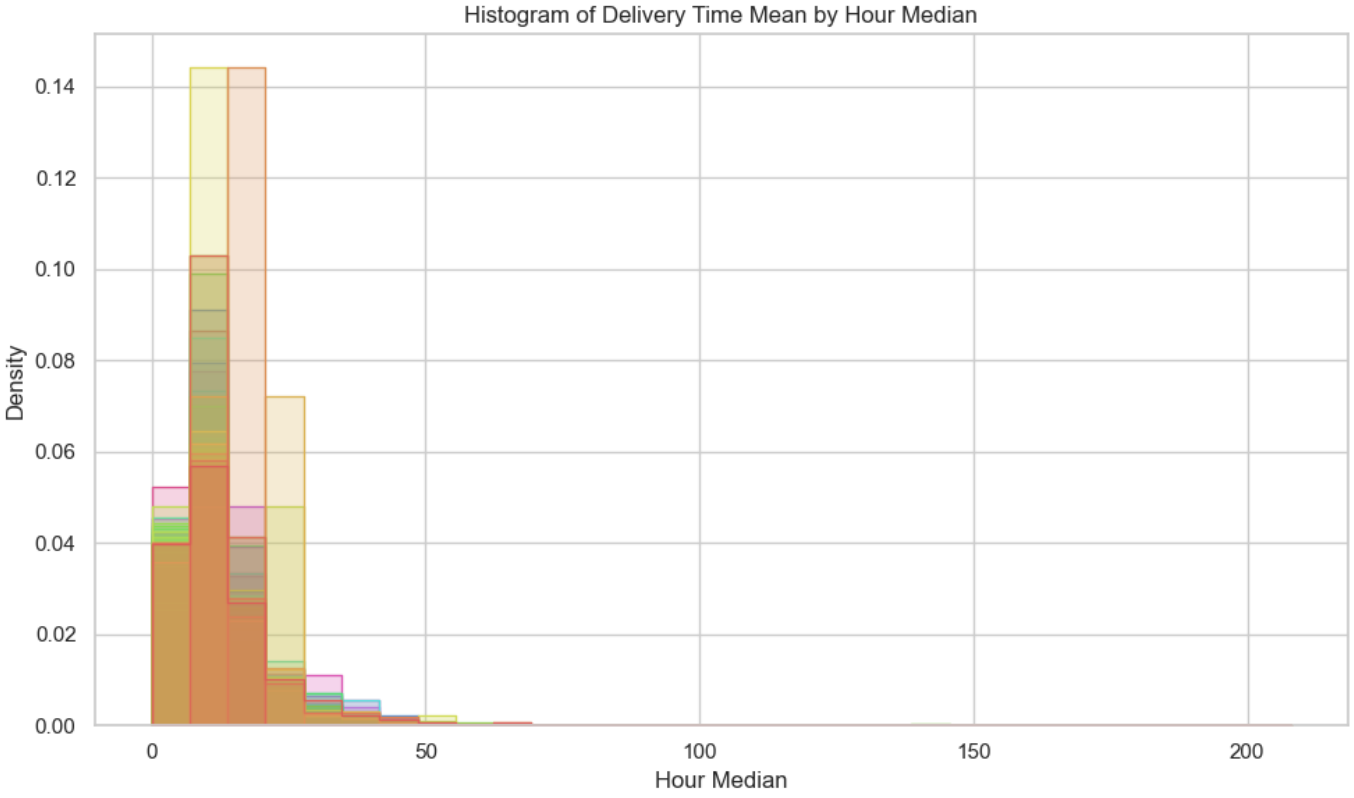
EDA: Delivery Time Mean

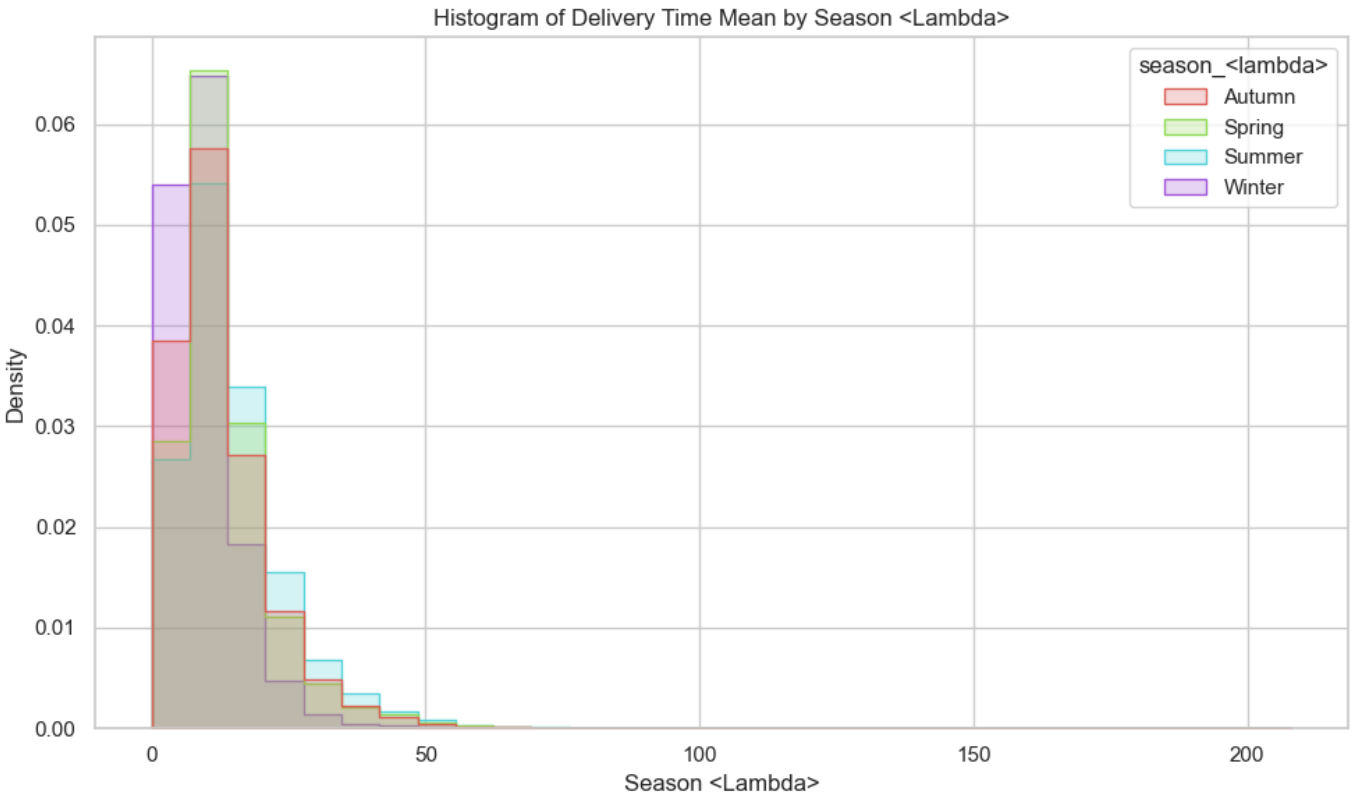
Visualizing various data on Delivery Time Mean for model building and to gather insights.

In [55]:
```
plot_histogram_by_feature(final_df, "weekday_<lambda>", "delivery_time_mean")
plot_histogram_by_feature(final_df, "hour_median", "delivery_time_mean")
plot_histogram_by_feature(final_df, "season_<lambda>", "delivery_time_mean")
plot_histogram_by_feature(final_df, "customer_state_<lambda>", "delivery_time_mean")
plot_histogram_by_feature(final_df, "distinct_product_categories", "delivery_time_mean")
```

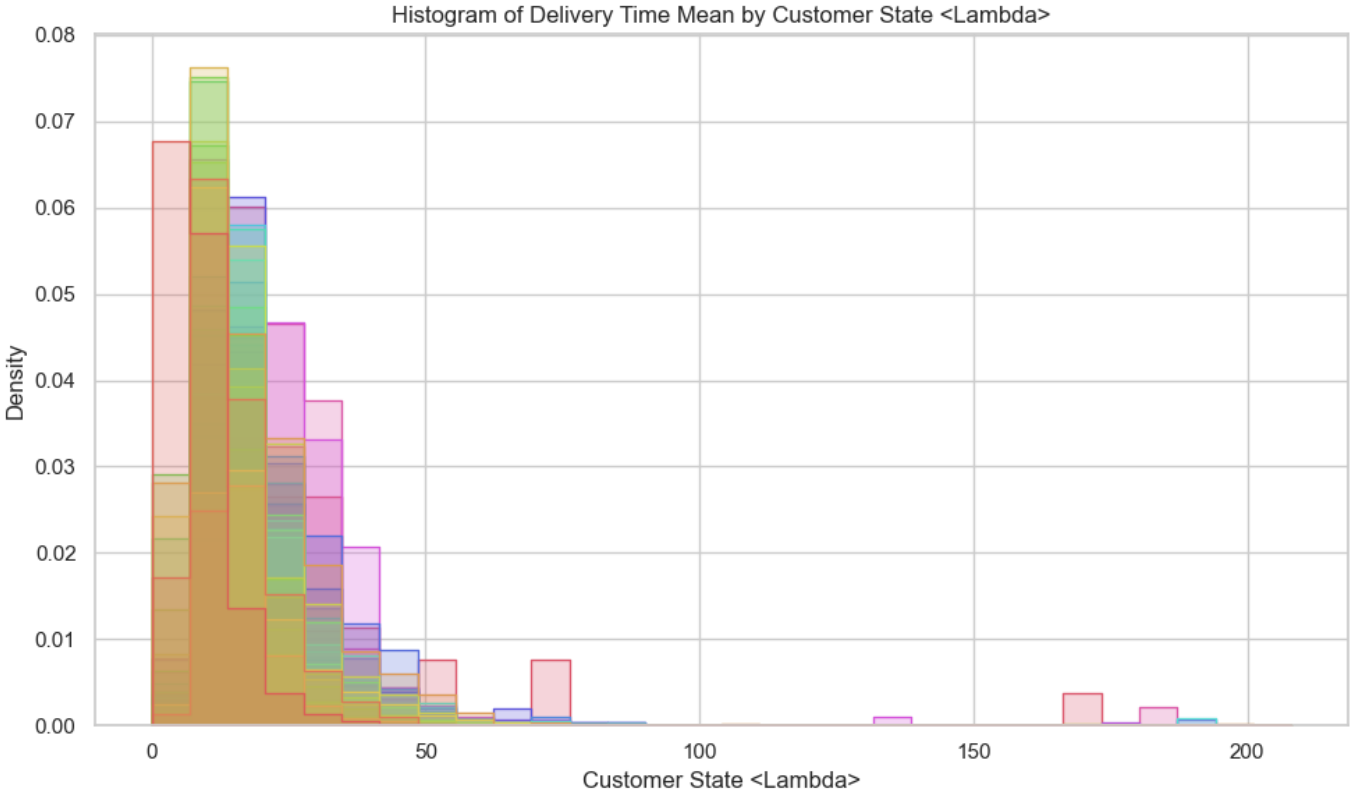


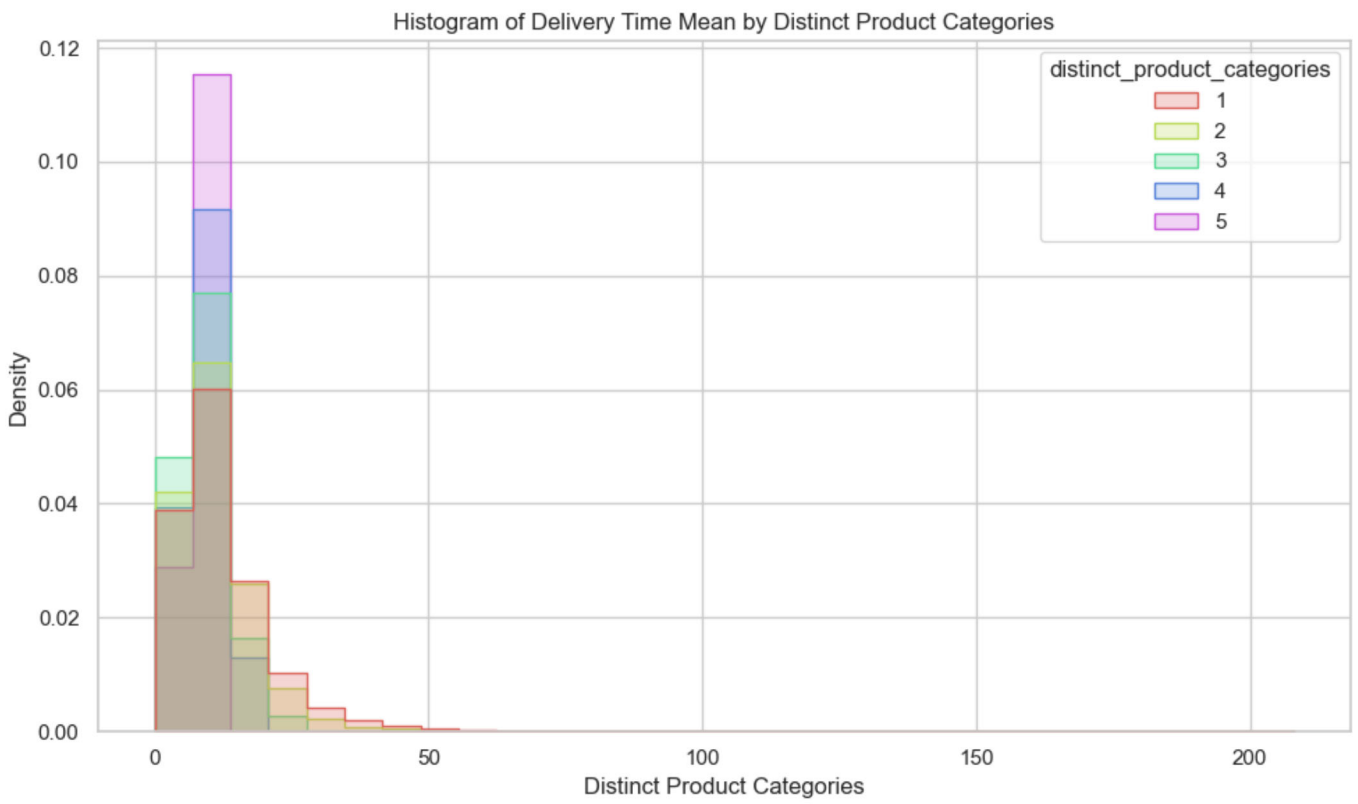

```
Too many labels:  47
```






```
Too many labels:  27
```



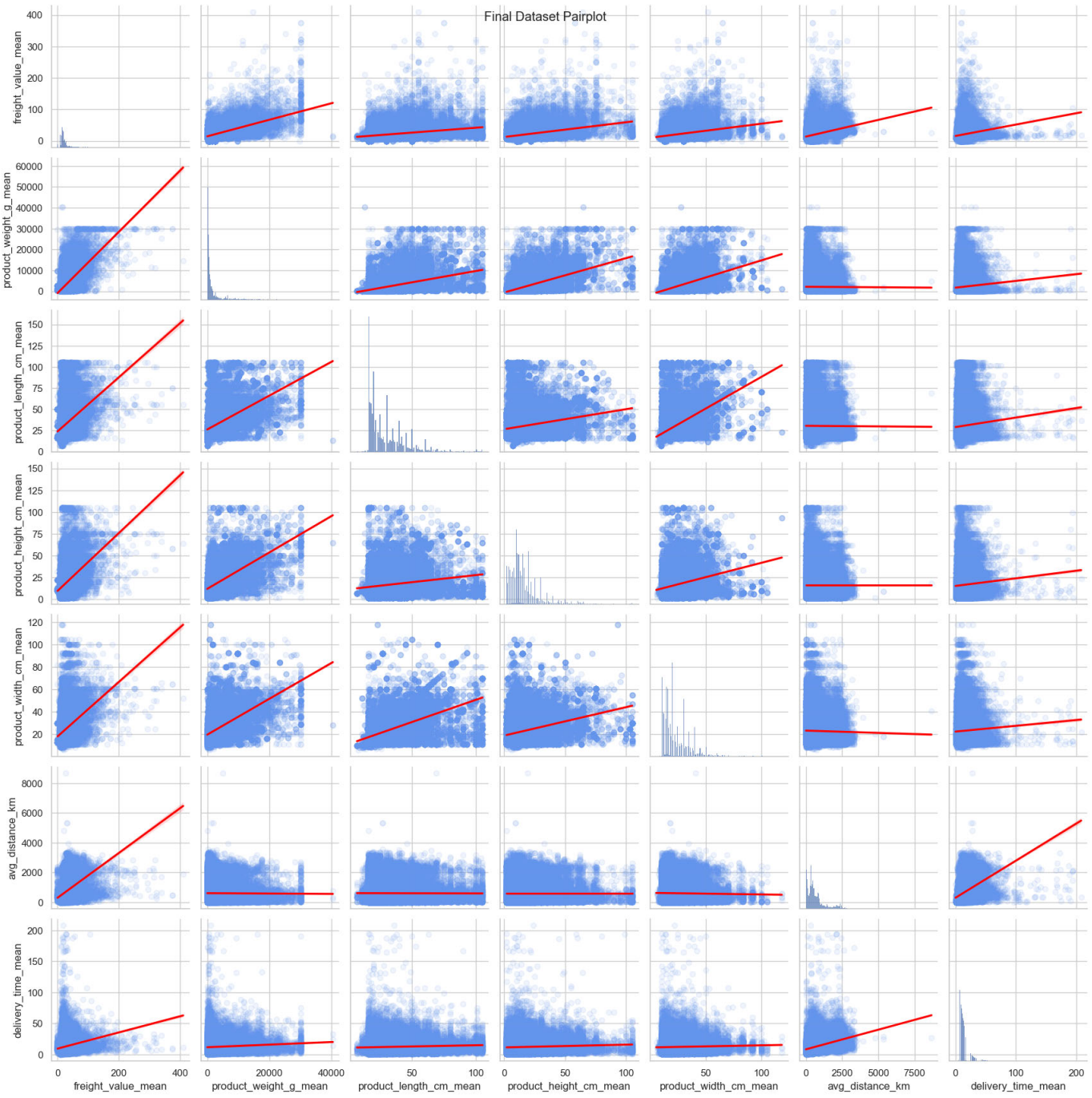


Freight Value Mean and Delivery Time Mean pairplots

Visualizing potential relationships between the target features and numerical features.

In [56]:
```
pair_plotter(final_df[['freight_value_mean', 'product_weight_g_mean', 'product_length_cm_mean
                       'product_height_cm_mean', 'product_width_cm_mean', 'avg_distance_km',
                       "Final")
```

```
<Figure size 640x480 with 0 Axes>
```



EDA and Visualization Conclusion

The EDA provided useful insights into repeat buyer classification and two regression targets: freight value and delivery time.

The repeat buyer target classes shows inbalance, some features showing noticeable differences between buyer types. The regression targets, both freight value and delivery time had skewed distributions, with a small number of extreme values. Several features suggest meaningful relationships with each target allowing for potential prediction.