olist

Machine Language

Lab I- Visualization and Data Preprocessing

Submitted By

Justin Ehly

John Olanipekun

Helene Barrera

Feby Thomas Cheruvathoor

1. Business Understanding

Olist is the largest department store in Brazilian marketplaces, although it is relatively new to the eCommerce space having been founded in 2016. Much like Amazon and Etsy, Olist provides an integrated platform which connects the small,medium sellers to reach out to international marketplaces. Merchants are able to sell their products through the Olist Store and ship them directly to customers using Olist logistics partners. Olist provides a unique sales experience while improving their logistics performance, specifically when it comes to fulfillment options and Last Mile Delivery. It is important for Olist to ensure their logistics partners are performing effectively to improve customer review scores and gaining customer confidence in Olist eCommerce platform.

Olist contributed its past years sales order data to kaggle(link), dataset consists of roughly 100,000 orders from 2016 to 2018 and is multidimensional covering order information, consumer information, seller information, geolocation information, product attributes and customer reviews. The dataset will allow us to meet our stated business objectives. We will process the data using a combination of Python for data cleaning, mining, wrangling, exploration, feature selection and data modeling and will possibly employ cloud services for tasks such as running sentiment analysis. We are intended to bring out data insights which will help the Olist platform to the next level.

Logistic Advancement: There are many factors that contribute to a customer's review score, but the majority of our data set is focused around the logistics side of eCommerce. We have detailed time stamps through each stage of purchase and delivery, as well as geographical information which will allow us to dig into how to make improvements to shipping time and estimated shipping time. Our success can be measured by improvements to the estimated delivery time and a decrease in bad reviews that mention shipping.

Improving Customer Satisfaction: Based on the attributes available in the data set, we decided to approach the dataset from the standpoint of working for Olist to help improve customer satisfaction, which in turn likely has a strong relationship with various logistics attributes. To that end, we will focus on understanding how factors like price, freight cost and estimated delivery time influence each other.

Accurate Price Predictions: Along the same lines, we will choose a specific product category and look at trends for products in that category for the past 3 years to understand seasonal influences on price and total sales. We can also identify patterns in customer purchases based on the day of the week and major holidays and festivals observed in Brazil. This will help ensure that Olist is prepared for fluctuations in the use of their site, allowing them to adjust their logistics partners and stock accordingly. A major success factor will be to obtain accuracy of at least 85%, precision of at least 80%, sensitivity of at least 85%. These values are subject to review, contingent upon exploratory data analysis.

2. Data Meaning Type

Olist supplemented 120MB of data and the high level data schema is depicted below. (Figure 2A)

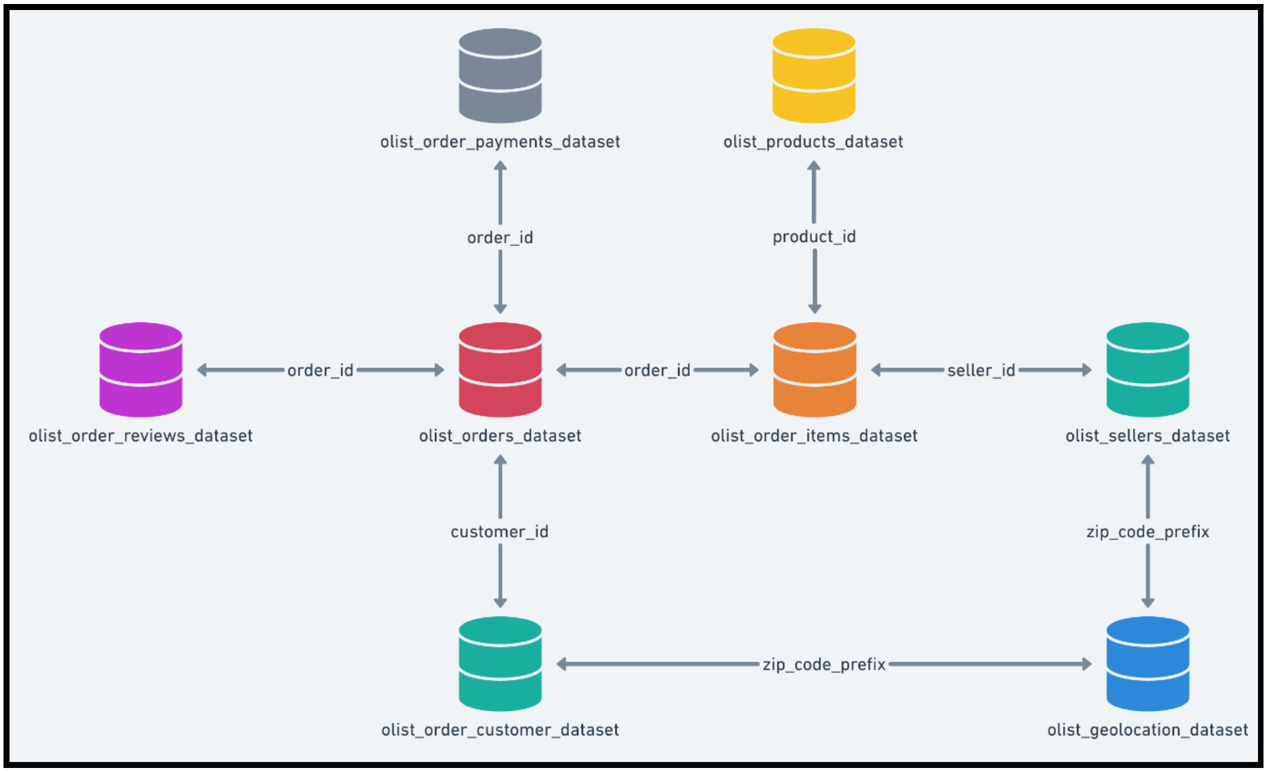


Figure 2A

Dataframe Info and the attribute description is listed below (Figure 2B)

```
olist.info() # now our data looks better!!

<class 'pandas.core.frame.DataFrame'>
Int64Index: 119148 entries, 0 to 119147
Data columns (total 39 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_id                       119148 non-null  category
 1   customer_id                    119148 non-null  category
 2   order_status                   119148 non-null  category
 3   order_purchase_timestamp       119148 non-null  object
 4   order_approved_at              118971 non-null  object
 5   order_delivered_carrier_date   117062 non-null  object
 6   order_delivered_customer_date  115727 non-null  object
 7   order_estimated_delivery_date  119148 non-null  object
 8   customer_unique_id             119148 non-null  category
 9   customer_zip_code_prefix       119148 non-null  category
 10  customer_city                  119148 non-null  category
 11  customer_state                 119148 non-null  category
 12  review_id                      119148 non-null  category
 13  review_score                   119148 non-null  float64
 14  review_comment_title           14189 non-null   category
 15  review_comment_message         51247 non-null   category
 16  review_creation_date           119148 non-null  object
 17  review_answer_timestamp        119148 non-null  object
 18  payment_sequential             119148 non-null  float64
 19  payment_type                   119148 non-null  category
 20  payment_installments           119148 non-null  float64
 21  payment_value                  119148 non-null  float64
 22  order_item_id                  118315 non-null  category
 23  product_id                     118315 non-null  category
 24  seller_id                      118315 non-null  category
 25  shipping_limit_date            118315 non-null  object
 26  price                          118315 non-null  float64
 27  freight_value                  118315 non-null  float64
 28  product_name_length            116606 non-null  float64
 29  product_description_length     116606 non-null  float64
 30  product_photos_qty             116606 non-null  float64
 31  product_weight_g               118295 non-null  float64
 32  product_length_cm              118295 non-null  float64
 33  product_height_cm              118295 non-null  float64
 34  product_width_cm               118295 non-null  float64
 35  product_category_english       116606 non-null  category
 36  seller_zip_code_prefix         118315 non-null  category
 37  seller_city                    118315 non-null  category
 38  seller_state                   118315 non-null  category
dtypes: category(18), float64(13), object(8)
memory usage: 42.2+ MB
```

Figure 2B. Dataframe Info and the attribute description

More detailed description of the dataset.

Attribute	Value_type	Description
order_id	category	order unique identifier (99,441 unique)
customer_id	category	key to the orders dataset - each order has a unique customer_id (99,431 unique)

order_status	category	order status, 7-levels (shipped, canceled, invoiced, processing, approved, unavailable, delivered)
order_purchase_timestamp	datetime64[ns]	purchase initiation timestamp (9/4/16 – 10/17/18)
order_approved_at	datetime64[ns]	payment approval timestamp (9/15/16-9/3/18)
order_delivered_carrier_date	datetime64[ns]	order posting timestamp when it was handed to the logistic partner (10/8/16-9/11/18)
order_delivered_customer_date	datetime64[ns]	actual order delivery date to the customer (10/11/16 – 10/17/18)
order_estimated_delivery_date	datetime64[ns]	estimated delivery date provided to the customer at the time of purchase initiation (9/29/16 – 11/11/18)
customer_unique_id	category	unique identifier of a customer (96,096)
customer_zip_code_prefix	category	first five digits of customer zip code (14,994 unique)
customer_city	category	customer city name (4,119 unique)
customer_state	category	customer state name (27 unique)
order_item_id	category	sequential number identifying number of items included in the same order (1-21)
product_id	category	product unique identifier (32,951 unique)
seller_id	category	seller unique identifier (3,095 unique)
shipping_limit_date	datetime64[ns]	seller shipping limit date for handing the order off to the logistic partner (9/18/16-4/9/20)
price	float64	item price (0.85-6,735)
freight_value	float64	item freight value (if an order has more than one item, the freight value is split between the items, scale: 0-409.68)
payment_sequential	float64	number of payment methods used by the customer (1-26)
payment_type	category	method of payment by customer [4 levels: credit_card, boleto, voucher, debit_card]
payment_installments	float64	number of payment installments by customer (0-24)
payment_value	float64	transaction value (0-13664.08, note vouchers don't count towards payment value)
seller_zip_code_prefix	category	first five digits of seller zip code (2246 unique)
seller_city	category	seller city name (611 unique)
seller_state	category	seller state name (23 unique)
product_category_name	category	root category of product in Portuguese (73 levels)

product_name_lenght	float64	number of characters extracted from the product name (5-76)
product_description_lenght	float64	number of characters extracted from the product description (4-3992)
product_photos_qty	float64	number of product photos published (1-20)
product_weight_g	float64	product weight measured in grams (0-40425)
product_length_cm	float64	product length measured in centimeters (7-105)
product_height_cm	float64	product height measured in cemitmeters (2-105)
product_width_cm	float64	product width measured in centimeters (6-118)
product_category_name_english	category	product category name in English (71 levels – 2 need to imputed)
review_id	category	review unique identifier (99,173 unique0
review_score	float64	1 to 5 rating given by the customer on a satisfaction survey (1-5)
review_comment_title	category	comment titles from the review left by the customer (4600 unique)
review_comment_message	category	comment message from the review left by the customer [note: 58% missing] (36,921 unique)
review_creation_date	datetime64[ns]	date satisfaction survey sent to customer (10/10/16-8/30/18)
review_answer_timestamp	datetime64[ns]	satisfaction survey answer timestamp (10/7/16 – 10/29/18)

3. Data Quality

a. Missing Values

- Rename the column names product_name_lenght and product_description_lenght to correct the spelling on length.

- Initially our dataset had quite a few thousand missing values, but we made the business decision to only focus on orders that were completed based on 2 factors,
    1. Order status was ‘delivered’
    2. Timestamp showing that the order was delivered to the customer.
- Once we reduced the initial dataset to only completed orders we found there were still some missing values:

order_delivered_carrier_date	1
order_approved_at	15
product_weight_g	20
product_length_cm	20
product_height_cm	20
product_width_cm	20
product_name_lenght	1638
product_description_lenght	1638
product_photos_qty	1638
product_category_english	1638
review_comment_message	66764
review_comment_title	101973

1. **Order_delivered_carrier_date:** The 1 missing value for the delivered to carrier attribute appears to be python not recognizing the datetime value in the cell, after much trial and error, it is just easier to delete the one record.
2. **Order_approved_at:** For the 15 order_approved_at missing values, we imputed those values by adding the difference between the average of the order_purchase_timestamp and the order_approved_at timestamp from the rest of the dataframe, that was about 10.5 hours.
3. **Product dimension information:** For the 20 values missing for weight and product dimensions, 19 of them had the same item number and the other 1 did not exist anywhere else in the dataset, so we deleted those since there was no way to impute them
4. **Product identity information:** For the 1,638 products missing product information such as name length (misspelled in the dataset from Kaggle), description length (also misspelled in the dataset from Kaggle), photo quantity and category, we performed a search of those same products across the entire dataset and did not find any other product_id’s that match those same product_id’s, in addition when we compiled the dataset, these records did not have categories assigned to them. With this information, it made more sense to remove them from the dataset

since we have so much data to work with already.[Note: This can have an adverse effect if/when we predict pricing and we may need to add those records back in at a later date.]

5. **Review Information:** Due to the nature of shoppers leaving comments, based on the *Online Review Statistics for 2021(Editor's Choice)*, about 5-10% of consumers write reviews of e-commerce purchases, so having the bulk of records missing reviews is not a surprise, but we want to keep those records in the dataframe.

b. Duplicate Data

- **order_ids** and **customer_ids:**
  - o Each order has an associated order_id and customer_id
  - o An order may contain more than one item, so these order_id and customer_id will be duplicated to show the association of those items with the corresponding order
  - o All values associated with an order_id will be duplicated as well, this includes timestamps, unique_customer_id's, etc.
- **unique_customer_id**: each customer has a unique_customer_id and that will be duplicated to show associated order_id and customer_id's for customers with more than one order
- **timestamps**: timestamps will be duplicated every time an order_id/ customer_id is duplicated due to an order having multiple items
- **freight_value**: for orders with more than one item, the freight_value is evenly (as possible) distributed across each item in the order
- **sellers and seller information:** sellers will be duplicated when they have more than one transaction or transactions with more than one item within the dataset

Duplicates

```
In [166]: # show the number of duplicated records in the dataframe
          dups = olist[olist.duplicated()].count()
          dups

Out[166]: order_id                          0
          customer_id                       0
          order_purchase_timestamp          0
          order_approved_at                 0
          order_delivered_carrier_date      0
          order_delivered_customer_date     0
          order_estimated_delivery_date     0
          customer_unique_id                0
          customer_zip_code_prefix          0
          customer_city                     0
          customer_state                    0
          review_id                         0
          review_score                      0
          review_comment_title              0
          review_comment_message            0
          review_creation_date              0
          review_answer_timestamp           0
          payment_sequential                0
          payment_type                      0
          payment_installments              0
          payment_value                     0
          order_item_id                     0
          product_id                        0
          seller_id                         0
          shipping_limit_date               0
          price                             0
          freight_value                     0
          product_name_length               0
          product_description_length        0
          product_photos_qty                0
          product_weight_g                  0
          product_length_cm                 0
          product_height_cm                 0
          product_width_cm                  0
          product_category_english          0
          seller_zip_code_prefix            0
          seller_city                       0
          seller_state                      0
          purchase_wk_day                   0
          purchase_month                    0
          tot_order_amt                     0
          dtype: int64
```

Figure 3A. Chart showing no duplicates in the dataset.

c. Outliers Continuous Variables

- We will start exploring outliers by visually inspecting boxplots that can quickly show us what may appear to be outliers or anomalies with the continuous variables. Then we will address each variable in order below starting with review_score and finishing with tot_order_amt. (Figure 3B)

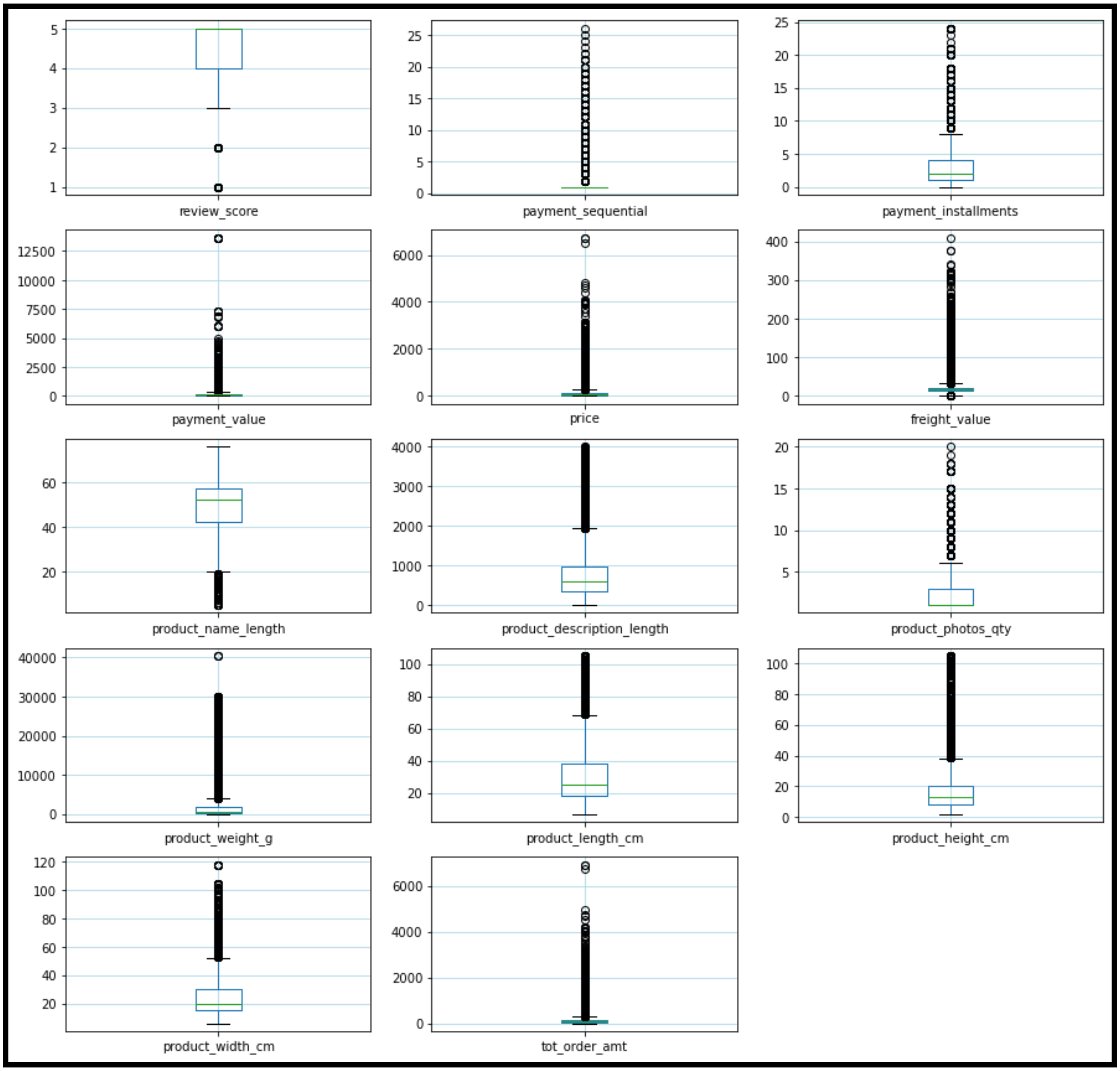


Figure 3B. Boxplots to visually show where they may be outliers in the continuous attributes.

- **Review Score:** Knowing that this range is from 1-5, we don't consider these low scores to be outliers
- **Payment Sequential:** this means how many forms of payment a customer used, the range is 1:26, sounds crazy, but there are 11 customers that used more than 20 payment types - looking at an individual sample, one customer used vouchers or small payments less than $2 each that added up to the total amount (price + shipping) of $40.85

- **Payment Installments:** 425 customers made more than 10 payment installments based on an average price of $251.39

- **Payment Value:** Payment Value = price+shipping-(any vouchers). (Figure 3C)

```
olist.payment_value.describe()

count    114080.000000
mean        172.142134
std         266.116465
min           0.000000
25%          60.950000
50%         108.060000
75%         189.370000
max       13664.080000
Name: payment_value, dtype: float64
```

Figure 3C. Payment Value statistics

- **Price:** Price has a range of R$0.85 : R$6,735 so it appears there are some expensive things for sale in the marketplace (Figure 3D)

```
olist.price.describe()

count    114080.000000
mean        120.016956
std         182.399977
min           0.850000
25%          39.900000
50%          74.900000
75%         133.000000
max        6735.000000
Name: price, dtype: float64
```

Figure 3D. Price Statistics

- **Freight Value:**
  - It makes sense that freight value has a relatively low 25% (R$13.08) : 75% (R$21.19) range when you compare it to the distribution of the price variable.
  - There are some items that cost a lot more to ship, but aren't shipped with the same frequency as the lower priced items and they fall into categories such as health and beauty, construction tools, housewares, baby and industry commerce and business
  - The last friday of April is free shipping day in Brazil, so that is why there is a minimum of R$0.

```
In [129]: olist.freight_value.describe()

Out[129]: count    114067.000000
          mean         20.009924
          std          15.726747
          min           0.000000
          25%          13.080000
          50%          16.320000
          75%          21.190000
          max         409.680000
          Name: freight_value, dtype: float64
```

Figure 3F. Freight Value statistics

- **Freight Value** and **Product Weight** are going be correlated just based on general knowledge of shipping: (Figure 3G)

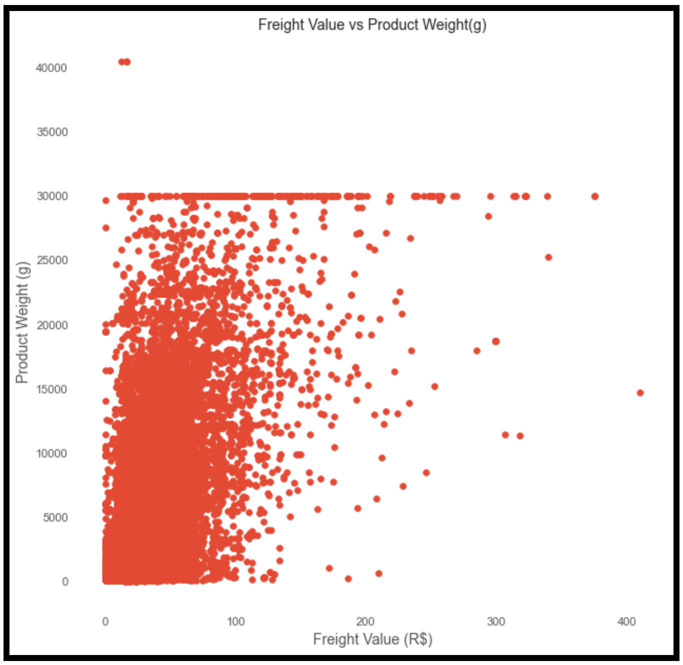


Figure 3G. Correlation between Product Weight and Freight Value

- **Product Height:** there are over 6,500 records with product height above 40cm, and 134 records with product height above or equal to 105cm, so we are confident these are correct
- **Product Width:** is very similar to product length in terms of stats (Figure 3H)

```
In [454]: olist.product_width_cm.describe()

Out[454]: count    114067.000000
          mean         23.103553
          std          11.738479
          min           6.000000
          25%          15.000000
          50%          20.000000
          75%          30.000000
          max         118.000000
          Name: product_width_cm, dtype: float64
```

Figure 3H. Product Wight statistics

        - Based on the boxplot where outliers seem to start above 50cm, there are nearly 2,900 records with widths over 50cm so they aren't considered outliers.
        - At the max of 105cm width we show 13 records, so again we are confident these measurements are accurate

- **Product Name Length** and **Product Description Length** are descriptors of the products and populated by the sellers, they can be as detailed as they want or not
- **Product Photos Qt:y** is the number of photos associated with a product, again this is entirely up the seller depending on how descriptive they want to be
- **Product Length:** shows some possible outliers above 60cm, but it turns out there more than 6,000 records with length over 50cm. Also a the longest length of 105cm there are 311 rows, so we are confident these measurements are accurate (Figure 3I)

```
In [451]: olist.product_length_cm.describe()

Out[451]: count    114067.000000
          mean         30.290522
          std          16.157939
          min           7.000000
          25%          18.000000
          50%          25.000000
          75%          38.000000
          max         105.000000
          Name: product_length_cm, dtype: float64
```

Figure 3I. Product Length statistics

- **Product Weight:** as charted above in the boxplot matrix does something interesting around 30k grams, it just seems to cutoff
    - The smallest weights are: 0, 2, 25, 50 and 53 grams
    - There are 13 records with a weight of 0 or 2 grams and when we look at the means of the items we get
        - Price: R$135.62
        - Freight Value: R$22.92
        - Height: 19.62cm
        - Width: 38.46cm
        - Length: 22.69cm
        - Categories: best_bath_table, funiture_decor, stationary

- Note: while stationary may weigh 2g, we are suspicious that all of these are mistakes so we are going to remove them from the dataset (Figure 3J)

In [140]:
```
no_weight[['price', 'freight_value', 'product_height_cm','product_width_cm',
           'product_length_cm']].describe()
```

Out[140]:

	price	freight_value	product_height_cm	product_width_cm	product_length_cm
count	13.000000	13.000000	13.000000	13.000000	13.000000
mean	135.619231	22.920769	19.615385	38.461538	22.692308
std	63.858025	6.548597	7.089176	11.140133	9.621024
min	90.000000	14.490000	11.000000	30.000000	11.000000
25%	100.000000	18.510000	11.000000	30.000000	11.000000
50%	129.900000	23.710000	25.000000	30.000000	30.000000
75%	129.900000	23.850000	25.000000	52.000000	30.000000
max	334.800000	39.600000	25.000000	52.000000	30.000000

In [142]:
```
no_weight.product_category_english.unique()
```

Out[142]:
```
[bed_bath_table, furniture_decor, stationery]
Categories (3, object): [bed_bath_table, furniture_decor, stationery]
```

Figure 3J. Statistics for items that weigh 0 or 2grams.

- The other numbers that jump out are the gap between about 30kg and 40kg,
  - There are 312 records with product weights above 29kg
  - There are 3 records with product weights above 40kg, they are all in the bed_bath_table category and that sounds reasonable

- **Total Order Amount:** total order amount = price + freight value (shipping), so it is not surprising that this column follows the price column very closely and based on the findings that there aren't any outliers in price we are confident this attribute is ok as is.

4. Simple Statistics

	count	max	mean	median	min	var
price	118315.0	6735.00	120.65	74.90	0.85	33896.35
payment_value	119148.0	13664.08	172.74	108.16	0.00	71700.79
tot_order_amt	118315.0	6929.31	140.68	92.02	6.08	36572.62
freight_value	118315.0	409.68	20.03	16.28	0.00	250.80
payment_installments	119148.0	24.00	2.94	2.00	0.00	7.72
review_score	119148.0	5.00	4.00	5.00	1.00	2.00
product_weight_g	118295.0	40425.00	2112.33	700.00	0.00	14339232.16
product_length_cm	118295.0	105.00	30.27	25.00	7.00	262.08
product_height_cm	118295.0	105.00	16.62	13.00	2.00	181.01
product_width_cm	118295.0	118.00	23.08	20.00	6.00	138.05
product_name_lenght	116606.0	76.00	48.77	52.00	5.00	100.67
product_description_lenght	116606.0	3992.00	785.94	600.00	4.00	425858.88
product_photos_qty	116606.0	20.00	2.21	1.00	1.00	2.95
product_category_english	116606.0	NaN	NaN	NaN	NaN	NaN
customer_city	119148.0	NaN	NaN	NaN	NaN	NaN
customer_state	119148.0	NaN	NaN	NaN	NaN	NaN
seller_city	118315.0	NaN	NaN	NaN	NaN	NaN
seller_state	118315.0	NaN	NaN	NaN	NaN	NaN

Figure 4A. Simple statistics for the dataset.

5. Visualize Attributes

a. Broad, high level visualization of the dataset

Below is another image of the boxplots of the data distribution for continuous attributes in the data set as we find that most are heavily skewed towards the top. (Figure 5A.)

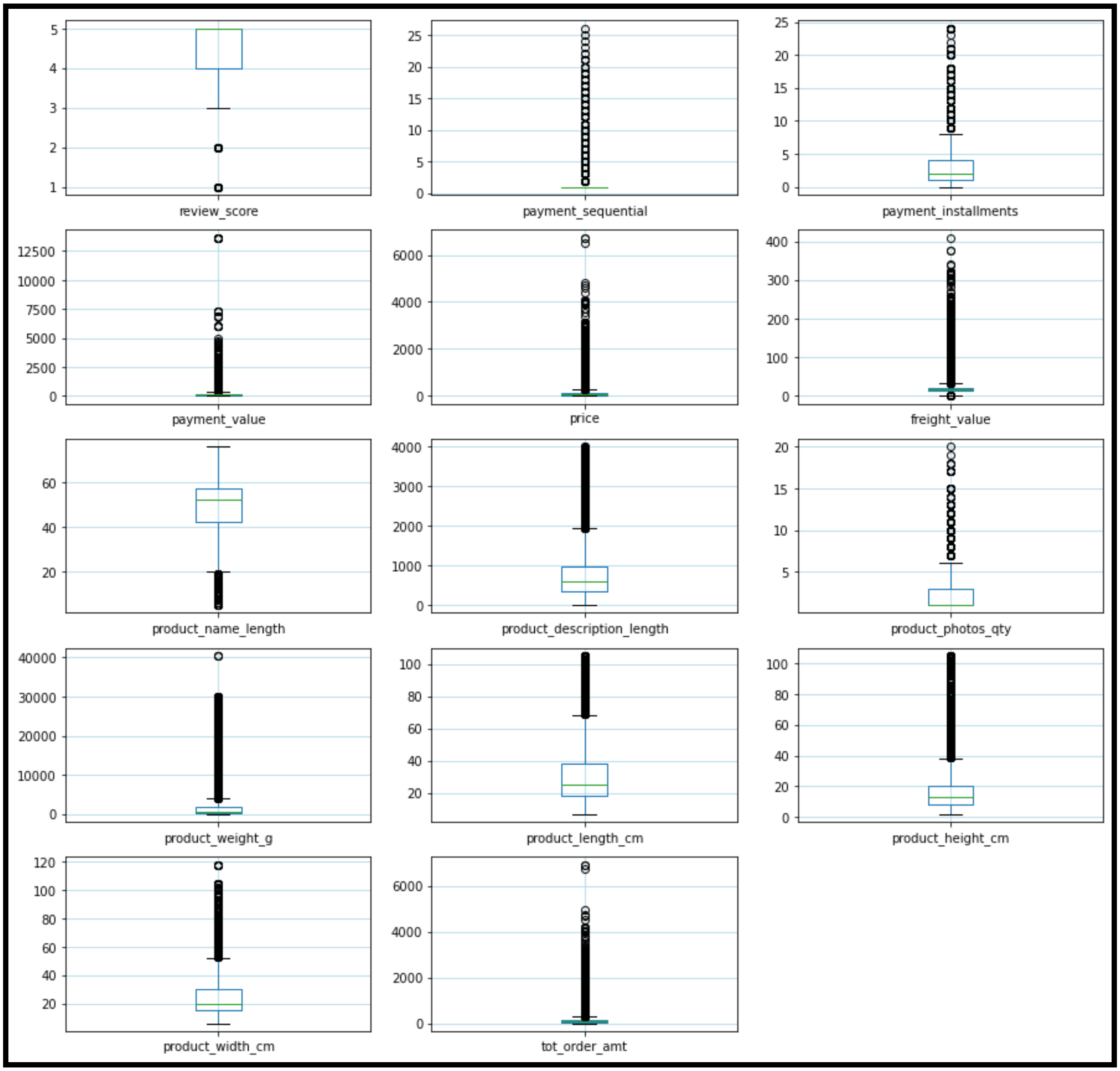


Figure 5A. Boxplots of the continuous attributes of the dataset.

Of particular interest are the skewed distribution shown for price related attributes (i.e. payment_value, price, feight_value). Figure 5B further displays the distribution as a histogram, showing strong skewness in the price data. The modeling tasks will strive to minimize the effect of this non-normality by scaling and/or transformation.

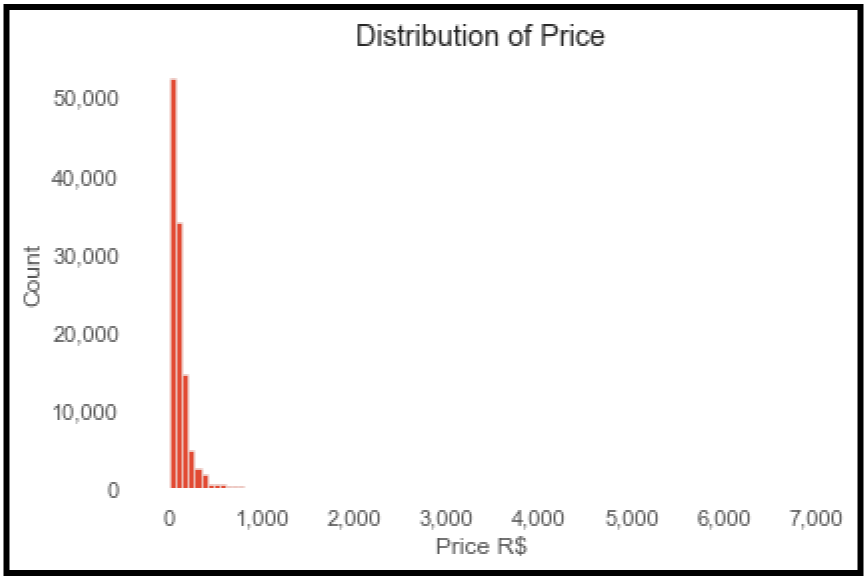


Figure 5B. Left skew of the price distribution.

A bar chart plot of review scores (Figure 5C) indicates that most of the sampled customers provided very good reviews (5). This is a key finding for understanding customer satisfaction and how to improve on it because we can drill into what constitutes a high score vs a low score.

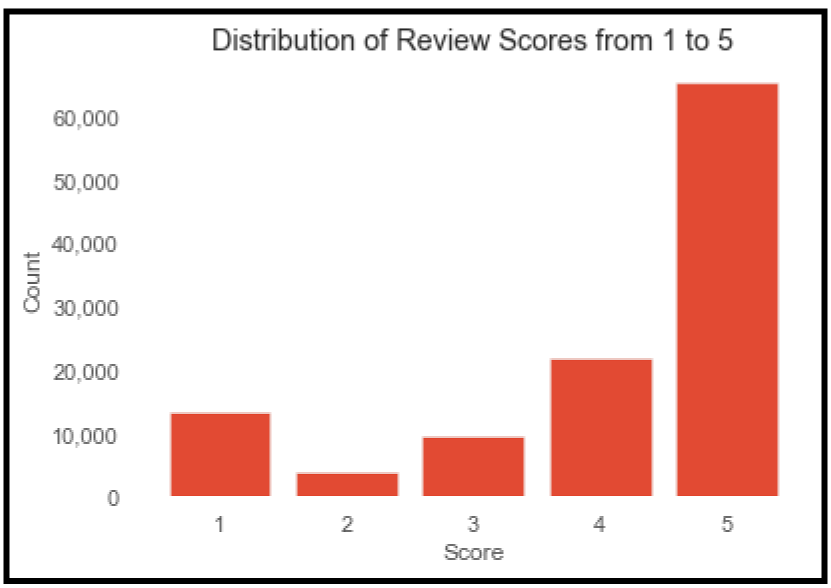


Figure 5C. Distribution of Review Scores

We see in Figure 5D. That the distribution of Freight Value is very similar to the distribution of price with a heavy left skew, suggesting that we should consider transforming it.

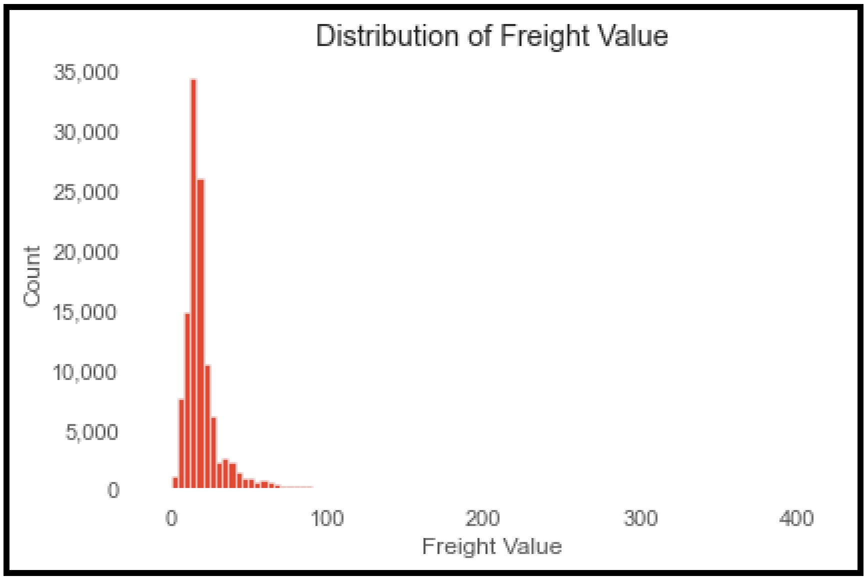


Figure 5D. Distribution of Freight Value.

We see that the majority of payments made from 2016-2018 on olist were by credit card, followed by boleto (a direct payment method like a bank wire), then by voucher and debit card.

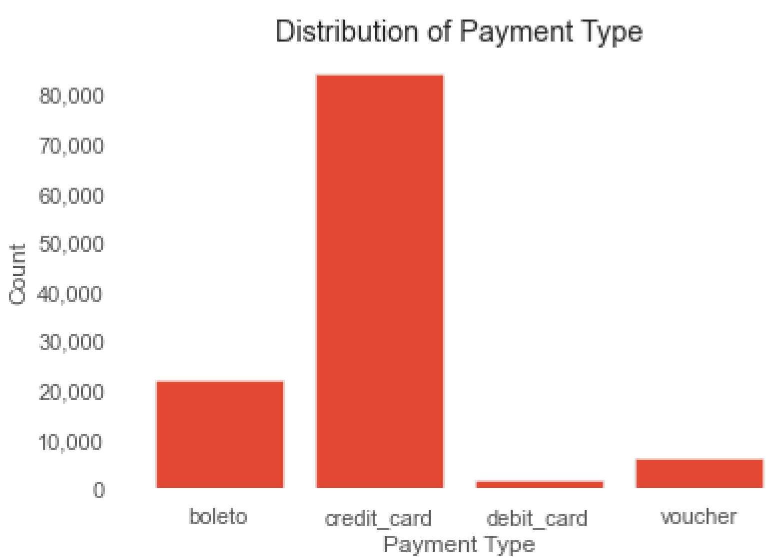


Figure 5F. Count of Payment Types

We also want to explore how many products are in each category to give us an idea of what the biggest or possibly more diverse categories are. This will help us later on determine the best category to work with for our accurate price predictions. The largest category is bed_bath_table with 11,808 products, followed by health_beauty with 9,814 products and third is sports_leisure with 8,791 products. The smallest 3 categories in order from smallest to largest are security_and_services (2), fashion_childrens_clothes (7) and pc_gamer (9).

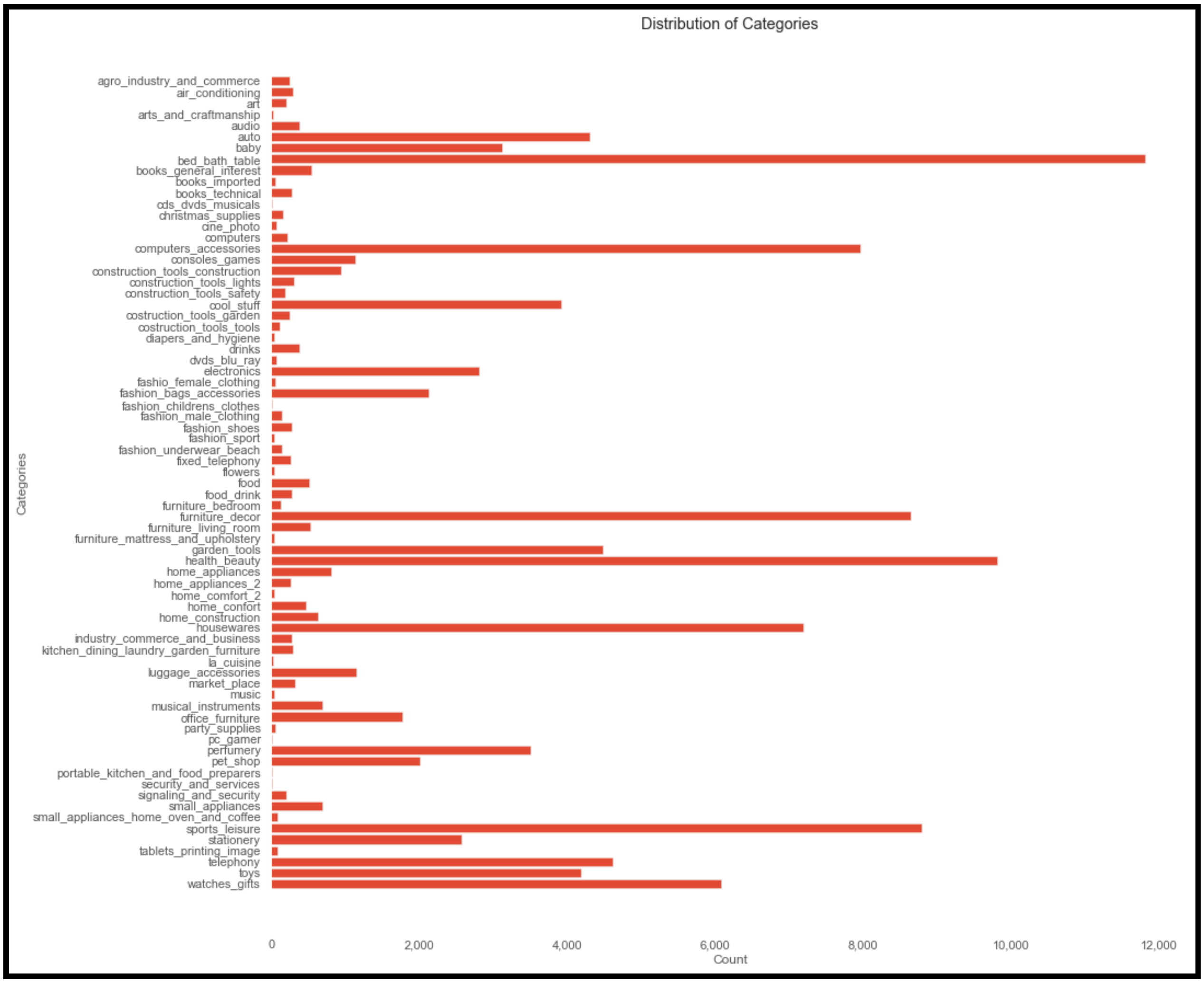


Figure 5G. The number of products by category in the dataset.

6. Explore Joint Attributes

Figure 6A below shows a correlation matrix for the raw numerical attributes in the Olist dataset. The best relationships that can be useful (i.e. correlation coefficient > 0.15) for the project objectives are:

Figure 6A. Correlation heatmap for the numerical attributes in the Olist dataset.

- Review Score score does not appear to be highly correlated with any of the other variables
- Payment Value and Price are highly correlated because Payment Value = Price + Freight - Voucher

- Payment Value and Total Order Amount are also highly correlated, Payment Value is a portion of Total Order Amount or the entire amount if no vouchers are used in the payment process.
- Price and Freight Value 0.42: not too highly correlated so cost doesn't necessarily dictate shipping cost
- Product Weight and Freight Value are highly correlated at 0.61, that makes sense because shipping costs are based on weight, package size, distance and type of shipping if that's any option.
- Interesting that the Product Length, Width and Height are not highly correlated with Freight Value, but they are correlated with Product Weight
- Product Length and Width also appear to be correlated with each other, we may consider merging the H+W+L into one attribute called Volume to reduce the overall number of attributes
- Total order amount and product description(s): There does not appear to be a strong association between price and the product descriptions (title length, description length, quantity of photos), but we suspect that there might be a correlation between the number of items sold by sellers and the length of the product title, description and the number of photos associated with a listing
- Price and payment instalments do not appear to be highly correlated either, that could be due to most people paying for entire orders up front in one payment vs electing to make multiple payments.
- Price and Payment Sequential similarly are not highly correlated and that makes it seem like more people are paying with a single payment method.

A bar chart plot of monthly sales (Figure 6B) shows that sales volume are highest in May, August and July and lowest in September and October.

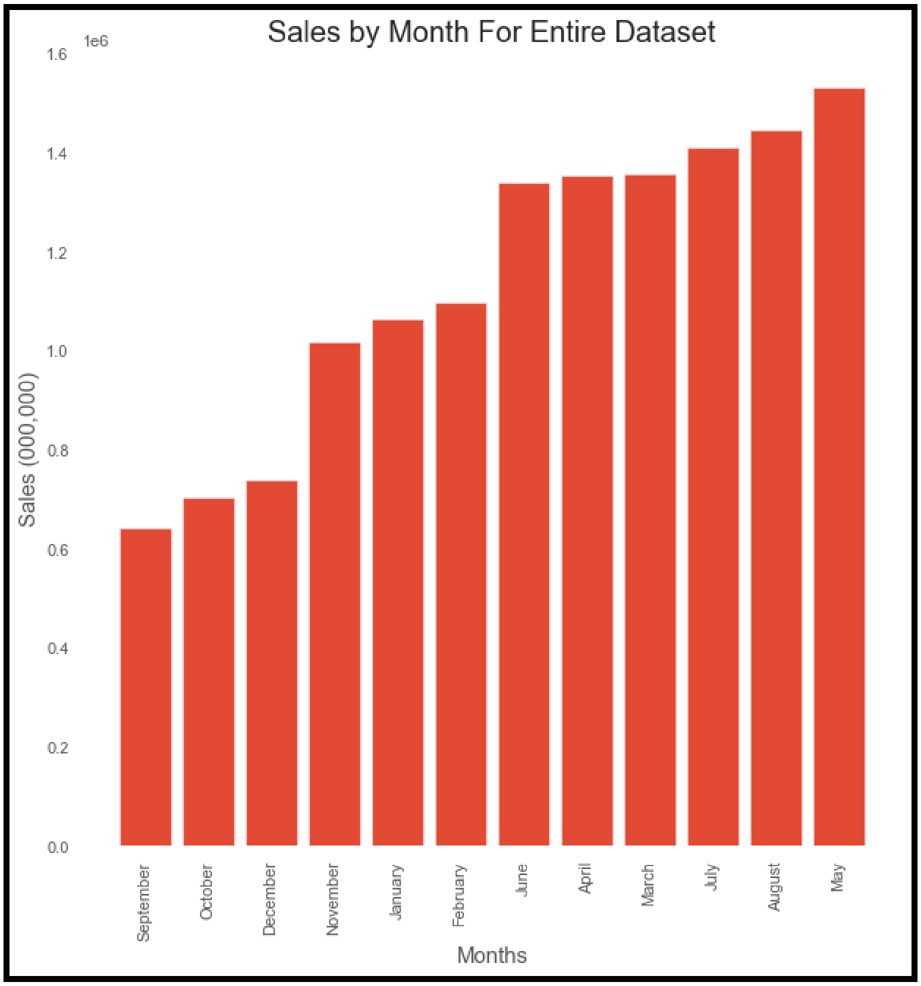


Figure 6B. Histogram of Sales volume versus Month.

Based on an article we found from *Latin America Business Stories* written on July 31, 2018, towards the end of our dataset, these trends can be explained as follows:Important dates: Consumers Day March 15th

- April: Free Shipping Day - last Friday in April, but in 2017 (that our dataset covers) some sellers gave customers an extra day of free shipping…this is important because according to research (not documented in the article), up to 90% of Brazilian consumers did not make an online purchase because of shipping costs
- May: Mother's Day - the 2nd Sunday in May (also contributes to free shipping day sales) - ranked as the 2nd busiest ecommerce day in the country

- June: June 12th ‘Dia do Namorados’, Brazil’s Valentine’s Day (in 2018 the most popular categories were Beauty and Health, Fashion and Technology
- August: Father’s Day - 2nd Sunday in August, most popular categories are Fashion and Accessories (31%), Electronics (22%), Books (14%), Fragrances (9%) and the rest are Travel, Shoes, CDs, DVDs and others.
- October: Children’s Day - October 12th - a lot of companies release new items around Children’s Day, this gives both kids and sellers a preview of what the Christmas season will look like
- November: Black Friday/ Cyber Monday or Black Week events are increasingly popular in November with big discounts on things like home appliances (-15.28%), electronics (-11.26%), cosmetics (-10.38%) and fashion (-10.35%)
- November: Chinese Double Eleven - November 11th. (basically celebrates Single’s Day), its a shopping day with deep discounts
- December: Christmas - the 13th paycheck is sort of a bonus that all registered workers are entitled to

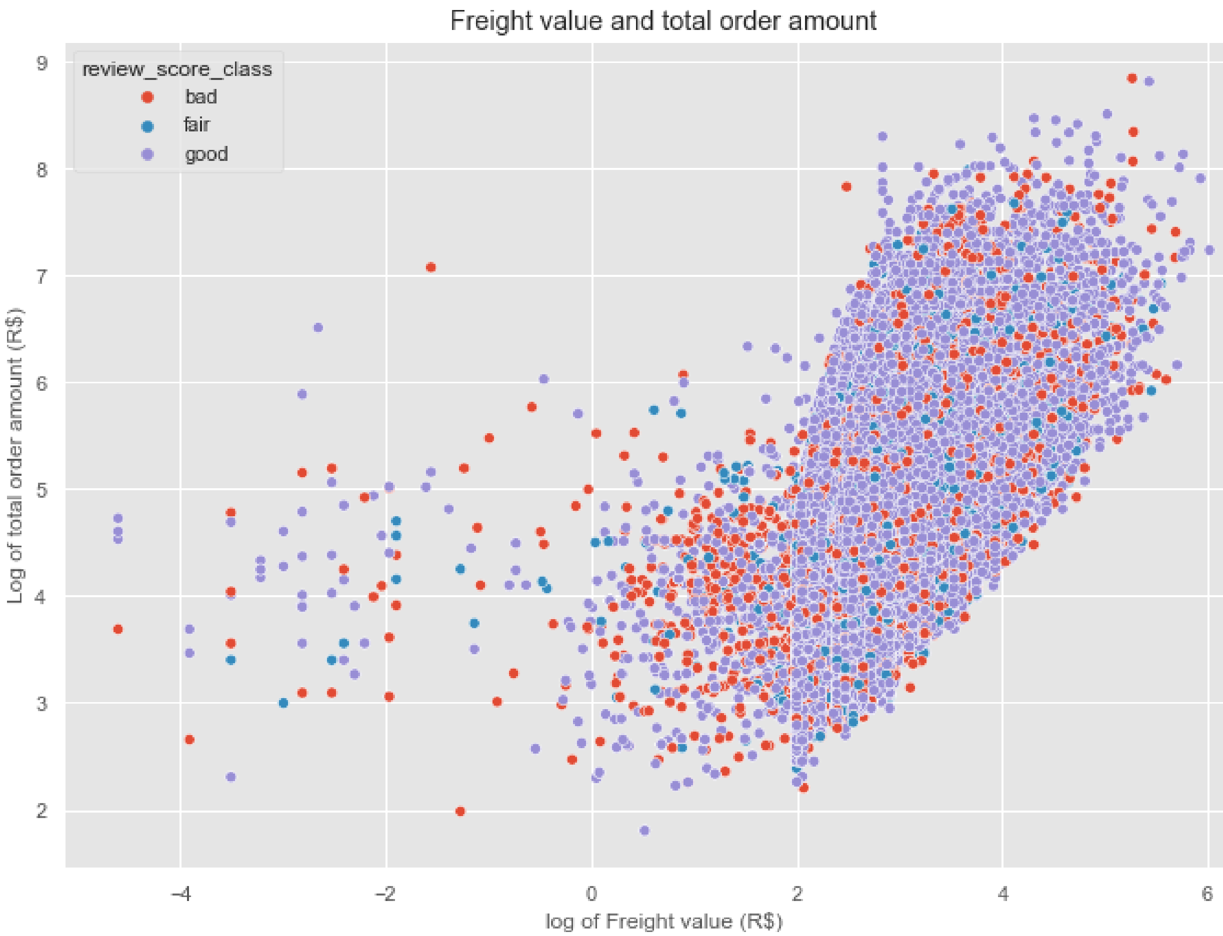


Figure 6C. Plot of log transformed total order amount vs log of freight value

Figure 6A (correlation heatmap) shows that total order amount and freight values have a relatively good association (correlation coefficient = 0.48), however the shape of the scatterplot of their raw values did not show this relationship. Therefore, these attributes were log transformed (Figure 6C). The log transformed plot depicts a positive linear association from log of freight value 0 and 2. Log of freight value is relatively invariant for lower values. The plot does not show an apparent trend for reviews score in total amount and freight value.

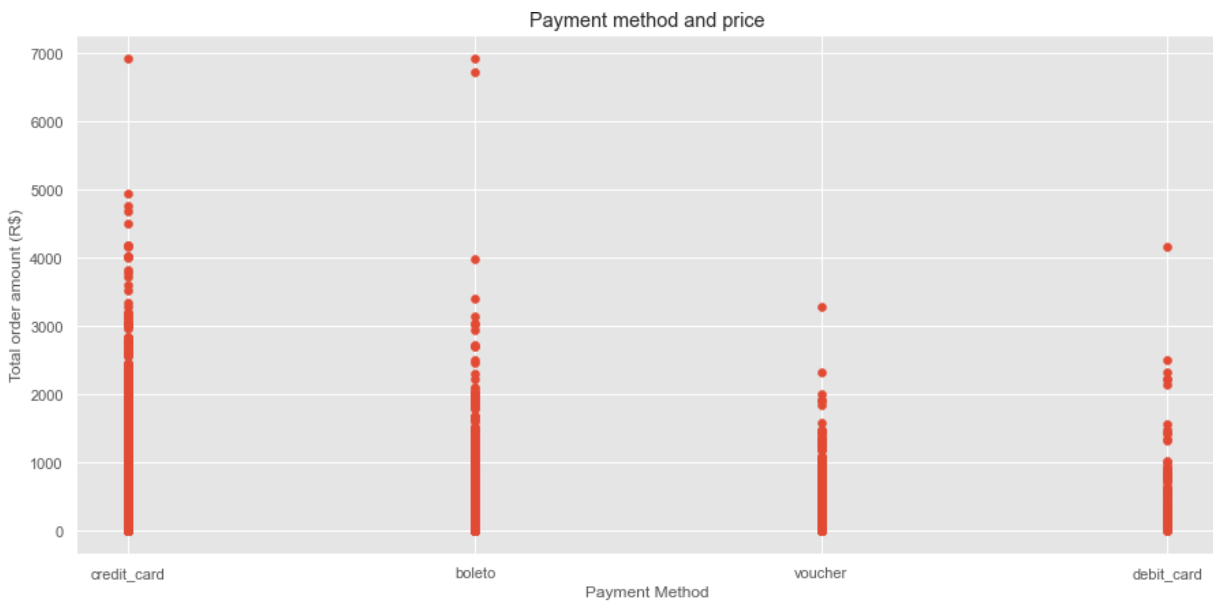


Figure 6D. Plot of Payment Method vs total order amount

A plot of payment method versus total amount paid for the order indicates that credit card payment is the most popular method for amounts higher than R$ 3000 and as noted earlier, credit cards are the most popular payment method.

7. Explore Attributes and Class

Delivery Performance (classification task)

One of the main objectives of this project is to understand factors that are related to customer satisfaction as measured by review attributes (e.g., review_score, review_comments).

As shown in Figure 7A, a relatively good correlation between delivery_estimate_discrepancy and review_score implies an association. The positive correlation suggests that customers that received their order earlier than the estimated delivery date tend to provide a high/good review_score. This relationship will be further explored during the modeling phase.

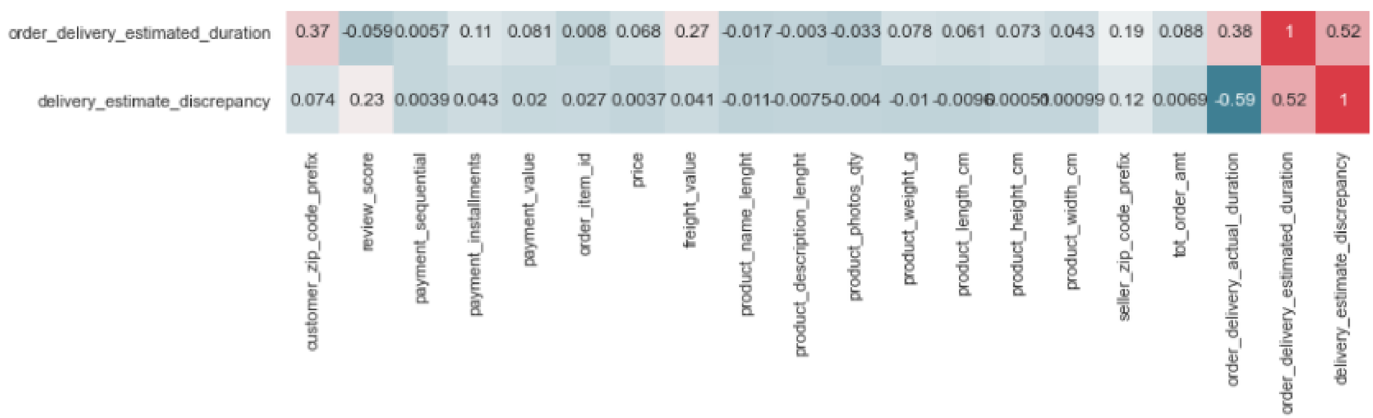


Figure 7A. Correlation heatmap that shows linear relationship between *delivery_estimate_discrepancy* and other numerical attributes in the Olist dataset.

The data distribution for delivery estimate discrepancy is not normally distributed as it shows a long-tailed distribution shape(Figure 5). This shape is due to the occurrence of outliers on both sides of the data distribution.

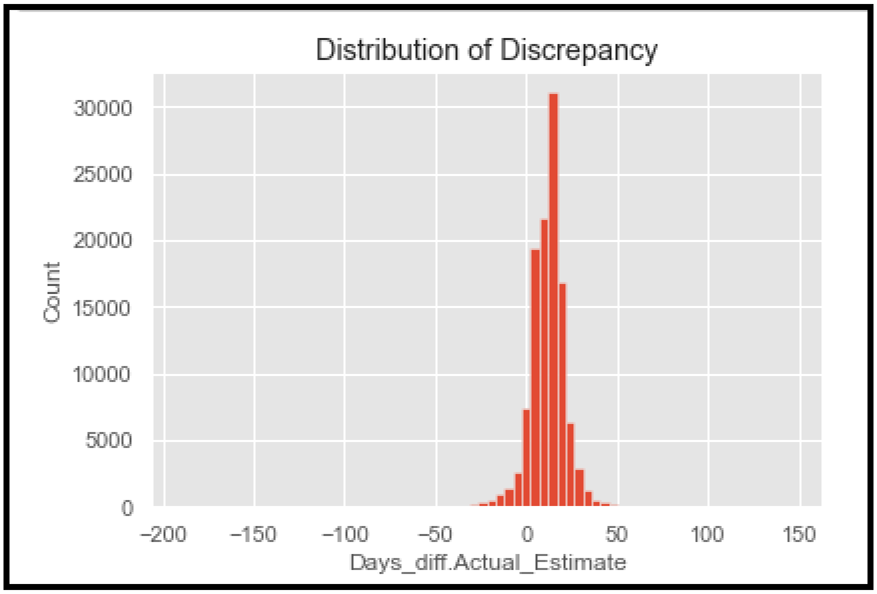


Figure 7B: Distribution of Delivery estimate discrepancy.

In this case, setting a threshold value of 95% quantile of the delivery estimate discrepancy values was able to exclude these outliers and resulted in a more representative data distribution that is less skewed (Figure 7C and 7D).

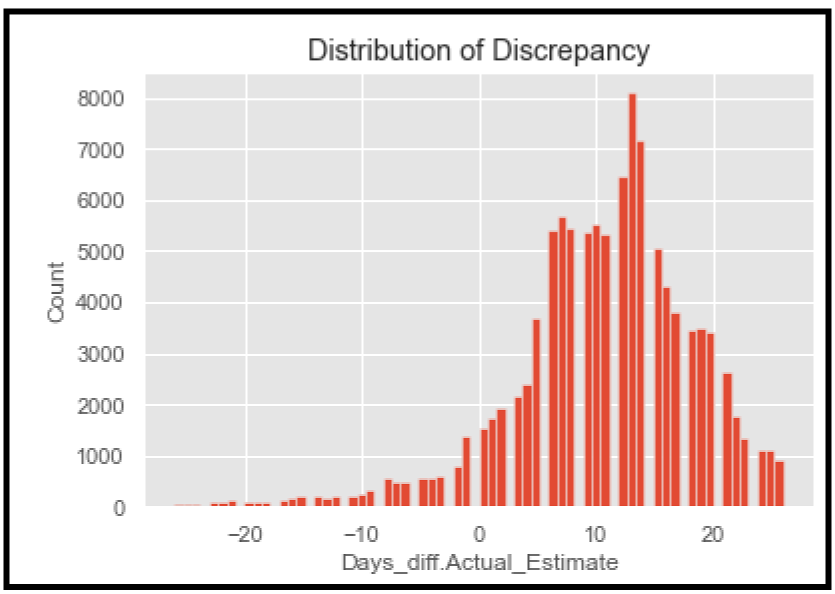


Figure 7C. Histogram of 95% percentile of delivery estimate discrepancy data.

In Figure 7D, each side of the gray line is the kernel density estimation which shows the distribution shape of the delivery estimate discrepancy. Looking at the wider sections of the violin plot, it can be said that there is a higher probability for delivery discrepancy values to fall within 5 and 25, unlike the skinnier sections which represent a lower probability.

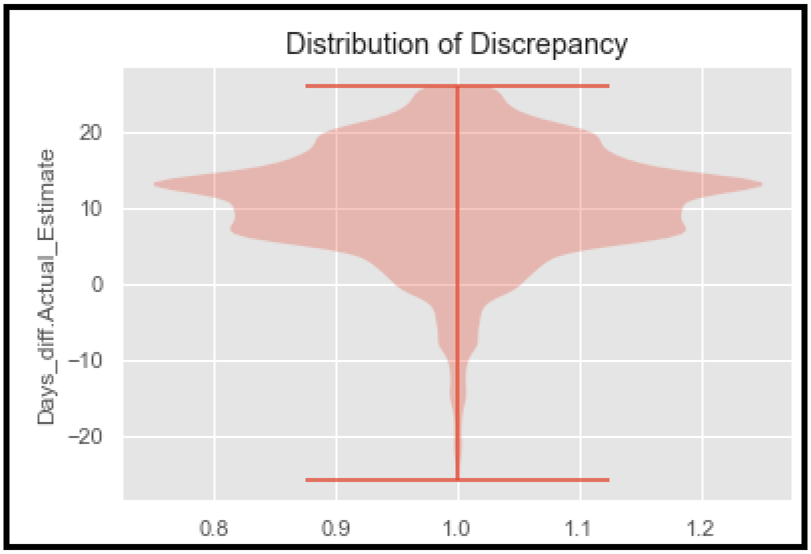


Figure 7D: Violin plot for 95% percentile of distribution of delivery estimate discrepancy.

Key Findings

One notable observation from figure 6 below was that customers that received their orders earlier than the estimated dates provided good review scores and vice versa for customers that received their orders late. This association deserves exploration to understand the nature of the relationship while noting any confounding variable.

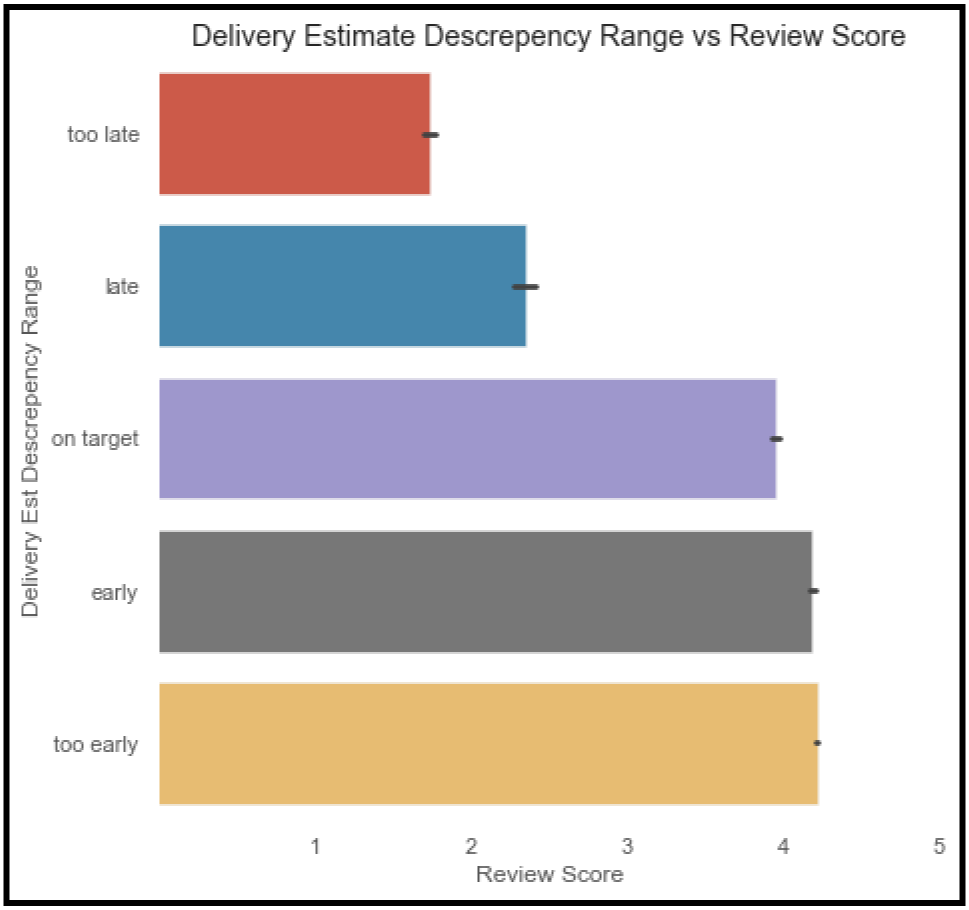


Figure 7E. Bar plot for levels of delivery estimate discrepancy range and their corresponding mean review scores to give us an idea of the relationship between the actual review score and the discrepancy between an estimated time of delivery and an actual time of delivery. More information for us to be able to help determine how to improve customer satisfaction in regards to logistic advancement.

Table 1 shows the distribution of delivery estimate discrepancy for respective months and corresponding review scores. It is normalized to show row proportions of the review scores.
From this table most of the processed orders are being delivered too early across all months of the year and even for some customers that provided low review scores. Despite the identified positive impact of early delivery, a major limitation to the practice is loss of sales from potential customers who view an estimated delivery date to be too long. These customers are not aware that, if placed, such orders might arrive 10 to 20 days earlier. For this reason, it is more informative to provide an estimated delivery date that is within 4 days of actual delivery date.

Table 1. Generated using pandas crosstab(). Delivery estimate discrepancy for respective months and corresponding review scores.

Month	Review_score	too late	Late	Target	Early	Too Early
April	1	0.15	0.04	0.09	0.06	0.65
	2	0.06	0.03	0.08	0.13	0.70
	3	0.03	0.02	0.10	0.13	0.72
	4	0.01	0.00	0.09	0.13	0.77
	5	0.00	0.00	0.05	0.11	0.83
August	1	0.10	0.07	0.19	0.12	0.52
	2	0.07	0.06	0.16	0.17	0.54
	3	0.02	0.03	0.24	0.15	0.56
	4	0.00	0.01	0.23	0.19	0.57
	5	0.00	0.00	0.24	0.17	0.59
December	1	0.23	0.06	0.10	0.09	0.54
	2	0.10	0.03	0.11	0.13	0.62
	3	0.03	0.01	0.12	0.12	0.72
	4	0.01	0.00	0.09	0.09	0.80
	5	0.01	0.01	0.05	0.07	0.86

February	1	0.34	0.10	0.12	0.09	0.35
	2	0.14	0.03	0.17	0.16	0.50
	3	0.07	0.02	0.17	0.13	0.61
	4	0.02	0.01	0.12	0.13	0.72
	5	0.01	0.01	0.09	0.12	0.78
January	1	0.19	0.05	0.09	0.04	0.63
	2	0.08	0.03	0.09	0.10	0.71
	3	0.02	0.01	0.14	0.10	0.73
	4	0.00	0.00	0.07	0.10	0.82
	5	0.00	0.00	0.05	0.07	0.88
July	1	0.11	0.03	0.12	0.11	0.63
	2	0.04	0.03	0.08	0.13	0.72
	3	0.02	0.01	0.12	0.16	0.70
	4	0.01	0.01	0.10	0.17	0.72
	5	0.00	0.00	0.08	0.16	0.75
June	1	0.07	0.01	0.05	0.07	0.80
	2	0.02	0.01	0.05	0.09	0.84
	3	0.01	0.00	0.05	0.09	0.83
	4	0.01	0.00	0.04	0.09	0.86
	5	0.00	0.00	0.03	0.08	0.89
March	1	0.36	0.10	0.12	0.08	0.34
	2	0.22	0.05	0.22	0.11	0.40
	3	0.08	0.03	0.22	0.15	0.52
	4	0.02	0.02	0.17	0.18	0.61

	5	0.01	0.01	0.13	0.19	0.66
May	1	0.14	0.06	0.09	0.10	0.61
	2	0.09	0.03	0.14	0.16	0.58
	3	0.04	0.02	0.19	0.11	0.64
	4	0.01	0.01	0.13	0.14	0.70
	5	0.01	0.01	0.10	0.13	0.76
November	1	0.31	0.08	0.13	0.10	0.37
	2	0.16	0.06	0.15	0.15	0.48
	3	0.06	0.02	0.21	0.16	0.55
	4	0.02	0.01	0.17	0.20	0.61
	5	0.01	0.01	0.13	0.18	0.67
October	1	0.11	0.03	0.09	0.10	0.67
	2	0.04	0.02	0.15	0.11	0.69
	3	0.02	0.03	0.15	0.13	0.67
	4	0.00	0.01	0.09	0.15	0.75
	5	0.00	0.00	0.08	0.13	0.78
September	1	0.16	0.03	0.13	0.14	0.54
	2	0.11	0.03	0.10	0.16	0.59
	3	0.03	0.01	0.12	0.15	0.68
	4	0.01	0.01	0.11	0.18	0.69
	5	0.00	0.00	0.09	0.15	0.75

Average Delay in Delivery - State Level

Another key finding revealed higher delivery delay (days) for the states such as AC (20), RO (18.9), AM(~ 19 days) while lowest delivery delays were measured for AL (8 days) and MA (9 days) respectively (Figure 7). Interestingly, states with highest delivery delay have the lowest population density which can negatively impact delivery network/supply chain efficiency in such states. (Figure 7G.) We should also point out that AM is the Amazonas state in Brazil, a state that is covered almost entirely by the Amazon rainforest.

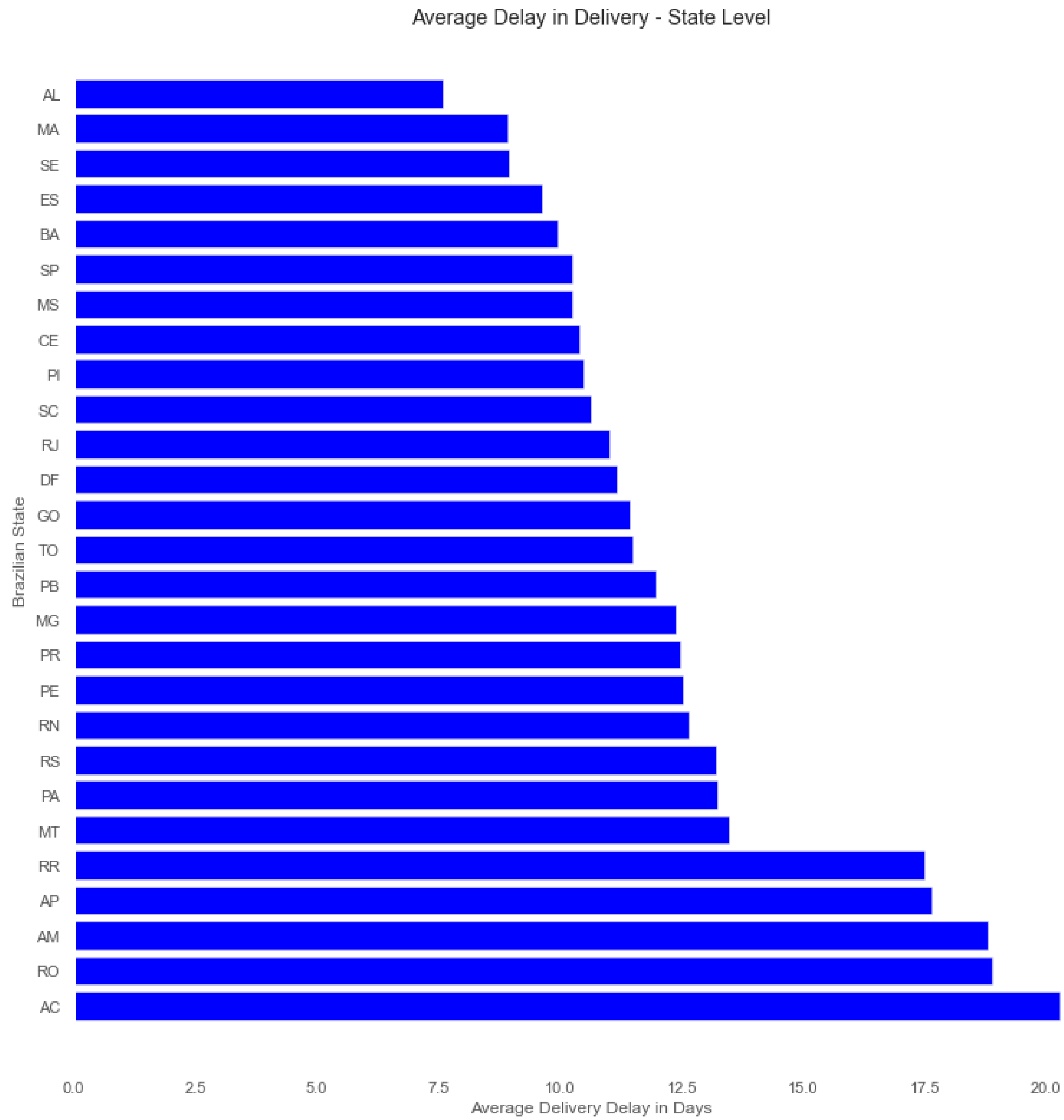


Figure 7G. Bar plot showing average delay in delivering products to remote states is plotted.

Approach to modeling delivery estimate

Modeling the delivery estimate will involve predicting actual delivery dates for the test set. Then 4 days will be added to that prediction to obtain the estimated delivery date. The resulting estimated delivery dates will be a refinement to the currently reported estimated delivery dates.

Predict Review Score (Classification task)

From the correlation coefficient heatmap presented in figure 6, review score is associated with item_id. But this relationship needs to be vetted. As previously stated, review_score is associated with delivery attributes.

Approach to modeling review score

The review score attribute is grouped into 3 classes viz: Bad (1 to 2), Fair (3) and Good (4 to 5) (Figure 8). A review score class will be predicted for the test set instances and the enumerated number of 'good' reviews will be used as a key performance index for customer satisfaction.

```
# verify that the cut function worked as expected.
range_review = pd.crosstab([olist['review_score_class']],
                           olist['review_score'])
print (range_review)
```

```
review_score           1     2     3      4      5
review_score_class
bad                13348  3947     0      0      0
fair                   0     0  9647      0      0
good                   0     0     0  21883  65255
```

Figure 8. Output showing the counts of review score classes generated from the review score attribute.

8. New Features

New features were created from their related counterparts. These are highlighted below.

- Product_dimension was formed from a product of product length, product width and product height.
- Total order amount was derived from price and freight_value attributes and it represents the uniform actual order amount because it excludes payment portions made by using vouchers and boleto payment methods.
- .purchase_month and purchase_wk_day were derived by extracting the month and day (respectively) portion of the order_purchase_timestamp converted datetime datatype.
- Actual duration of order delivery was derived from time duration between order approved date and actual date the customer received the order.
- Estimated duration of order delivery was obtained by calculating the time duration between order approved date and estimated date projected for delivery.
- Delivery estimate discrepancy is the difference between estimated duration of order delivery and actual duration of order delivery.**<u>This feature is key to optimizing delivery performance</u>**
- Delivery estimate range is binned levels of delivery estimate discrepancy set as too late, late, on-target, early, too early.
    - On-target delivery estimates are those delivered within +- 4 days (arbitrarily set to account for non-working days e.g. weekends) of the actual delivery date.
- Review_score_class is the grouped levels of review score attribute.

The next phase will add a new column for the total length of time it took between each stage of delivery could prove helpful in identifying steps in the logistics process where improvements could be made. Specifically, a factor called Processing, which is the time between the order_approved_at time and the order_delivered_carrier_date. This would show any slowdowns between the customer's payment being approved and it getting to the shipping/logistics partner. Another factor called Delivery could be created between order_delivered_carrier_date to order_delivered_customer_date, to show any delays with shipping times.

References

1. Olist dataset in Kaggle :
   https://www.kaggle.com/olistbr/brazilian-ecommerce?select=olist_order_items_dataset.csv
2. https://olist.com/
3. https://pt.wikipedia.org/wiki/Olist
4. Violin Plots 101: Visualizing Distribution and Probability Density:
   https://mode.com/blog/violin-plot-examples/. Accessed on 05/15/2021.
5. *Online Review Statistics for 2021(Editor's Choice)*, March 20, 2021
6. *Latin America Business Stories*, July 31, 2018
7. https://en.wikipedia.org/wiki/Amazonas_(Brazilian_state)

Appendix

Olist, a Brazilian E-commerce site provided a robust dataset of orders made at the Olist Store. The data set consists of roughly 100,000 orders from 2016 to 2018 and is multidimensional covering order information, consumer information, seller information, geolocation information, product attributes and customer reviews. The dataset will allow us to meet our stated business objectives. We will process the data using a combination of Python for data cleaning, mining, wrangling, exploration, feature selection and data modeling. and will possibly employ cloud services for tasks such as running sentiment analysis.

Data Meaning Type

There are 2,515 orders missing a customer delivery date. Of those missing delivery dates it appears there were 564 cancelled orders, 7 orders that were unavailable, 8 orders that show as delivered but are missing delivery dates and 364 orders that were invoiced but there is no record of the orders being delivered. Other reasons we don't have delivery data is because at the time the data was pulled, an order could've been shipped, processing or approved or otherwise still in the process of making its way to the customer. These are just good things to note, we may consider just removing all 2,515 lines since we have so much data already. There were 7 orders that were cancelled, but still delivered and more than likely that's because the order was cancelled after it shipped from the seller.

Modeling

Main objective is to determine the optimal delivery duration with the aim that this prediction will be better than the current estimated delivery. We will then recommend our best model to management to use for estimating delivery date for future orders.

Approach:

- Regression problem (can be changed to classification problem by transforming target to *'Late delivery', 'Early Delivery'* or *'Precise delivery'*)

- Duration_Actual =
    - Actual delivery date (order_delivered_customer_date) - order purchase date (order_purchase_timestamp)
- Duration_Estimated =
    - Estimated delivery date (order_estimated_delivery_date) - order purchase date(order_purchase_timestamp)

- Split to train-test
- Predict ‘actual duration’ for the test.
    - Set performance threshold (e.g., accuracy of 85%)
- Further compare the predicted result to ‘Estimated delivery’
- Modeling techniques:
    - Start with explainable models so we can determine the important features
        - E.g. linear or logistic regression, single trees model
    - For prediction, apply complex models. E.g. xgboost(), SVM(), ensemble learning techniques.
- Assess model:
    - Keep improving the model till the threshold is surpassed

Evaluation

- Evaluate results
    - Compare with the original project requirement.
- Review the process:
    - Review the steps for any mistake or misstep.
- Determine next steps: Based on the previous three tasks, determine whether to proceed to deployment, iterate further, or initiate new projects.

Success Criteria

Outcome Measurement:

Measured by improvements to accuracy of estimated delivery times and less bad reviews that mention shipping.

Estimated delivery time is within 1 day of the customer receiving their ordered package.

Data Importance:

Strengthen their competitiveness in the market by

Prediction Effectiveness:

A major success factor will be to obtain accuracy of at least 85%, precision of at least 80%, sensitivity of at least 85%. These values are subject to review, contingent upon exploratory data analyses.

Python Code:

Olist E-Commerce Dataset Project

Machine Learning 1 ¶

Team Members

• Helene Barrera • Justin Ehly • Babatunde “John” Olanipekun • Feby Thomas Cheruvathoor

In [1]:
```
# set up environment
import numpy as np
import pandas as pd
import os
from datetime import datetime as dt
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
```

In [2]:
```
# change working directory
#os.chdir(r"C:\Users\olani\OneDrive\Documents\Data Science\SMU-Data Science\Ma
chine Learning 1\Olist_Dataset\Olist_Datasets")
#"C:\Users\olani\OneDrive\Documents\Data Science\SMU-Data Science\Machine Lear
ning 1\Olist_Dataset"
os.chdir('C:/Users/justi/Documents/GitHub/olist/data')

# get current working directory
os.getcwd()
```

Out[2]:
```
'C:\\Users\\justi\\Documents\\GitHub\\olist\\data'
```

In [3]:
```
# set up some colors and text attributes to markdown
class color:
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'
```

In [4]:
```
customers = pd.read_csv('olist_customers_dataset.csv')
items = pd.read_csv('olist_order_items_dataset.csv')
payments = pd.read_csv('olist_order_payments_dataset.csv')
reviews = pd.read_csv('olist_order_reviews_dataset.csv')
orders = pd.read_csv('olist_orders_dataset.csv')
products = pd.read_csv('olist_products_dataset.csv')
sellers = pd.read_csv('olist_sellers_dataset.csv')
translation = pd.read_csv('product_category_name_translation.csv')
geolocation = pd.read_csv('olist_geolocation_dataset.csv')
```

In [5]:

```
the_df = {'customers': customers,
              'items': items,
              'payments': payments,
              'orders': orders,
              'products': products,
              'sellers': sellers,
              'reviews': reviews,
              'categories': translation,
              'geolocation': geolocation}
print("______________________________________________________________________
________________________")
print("Description of the {} dataframes".format(len(the_df)))
print("______________________________________________________________________
________________________")
for i, j in the_df.items():
    print('{} dataframe:      {} rows and {} columns'.format(str(i),j.shape[0
],j.shape[1]))
    print(list(j.columns))
    print("")
print("______________________________________________________________________
_________________________")
```

```
Description of the 9 dataframes

customers dataframe:      99441 rows and 5 columns
['customer_id', 'customer_unique_id', 'customer_zip_code_prefix', 'customer_c
ity', 'customer_state']

items dataframe:      112650 rows and 7 columns
['order_id', 'order_item_id', 'product_id', 'seller_id', 'shipping_limit_dat
e', 'price', 'freight_value']

payments dataframe:      103886 rows and 5 columns
['order_id', 'payment_sequential', 'payment_type', 'payment_installments', 'p
ayment_value']

orders dataframe:      99441 rows and 8 columns
['order_id', 'customer_id', 'order_status', 'order_purchase_timestamp', 'orde
r_approved_at', 'order_delivered_carrier_date', 'order_delivered_customer_dat
e', 'order_estimated_delivery_date']

products dataframe:      32951 rows and 9 columns
['product_id', 'product_category_name', 'product_name_lenght', 'product_descr
iption_lenght', 'product_photos_qty', 'product_weight_g', 'product_length_c
m', 'product_height_cm', 'product_width_cm']

sellers dataframe:      3095 rows and 4 columns
['seller_id', 'seller_zip_code_prefix', 'seller_city', 'seller_state']

reviews dataframe:      100000 rows and 7 columns
['review_id', 'order_id', 'review_score', 'review_comment_title', 'review_com
ment_message', 'review_creation_date', 'review_answer_timestamp']

categories dataframe:      71 rows and 2 columns
['product_category_name', 'product_category_name_english']

geolocation dataframe:      1000163 rows and 5 columns
['geolocation_zip_code_prefix', 'geolocation_lat', 'geolocation_lng', 'geoloc
ation_city', 'geolocation_state']
```

Merge the CSV Files

Begin the process of merging the csv files together

Keeping in mind there are going to duplicate keys for order_id and customer_id because one order may contain

- 1 item
- 1 payment
- 1 review (each item may have a separate review)

List of Pre-Merge Items

1. Need to merge products + translations as pdf
    - imput 2 missing translations as products_eng
    - remove original category
    - rename english category to shorten name to *product_category_english*

Merge order

1. inner merge customers + orders as df
2. right merge df + reviews as df2
3. right merge df2 + payments as df3
4. right merge df3 + items as df4
5. left merge df4 + pdf as df5
6. left merge df5 + sellers as df6
7. set olist = df6

DF1

Merge customers + orders df's

In [6]:
```
# get data from github repo on hard disk
df = pd.merge(orders, customers, on="customer_id")
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 99441 entries, 0 to 99440
Data columns (total 12 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
 8   customer_unique_id             99441 non-null  object
 9   customer_zip_code_prefix       99441 non-null  int64 
 10  customer_city                  99441 non-null  object
 11  customer_state                 99441 non-null  object
dtypes: int64(1), object(11)
memory usage: 9.9+ MB
```

In [8]:
```
#df.head(5)
```

DF2

Now let's add the reviews df (left)

1. how many duplicate keys order_id are there in the df
2. merge df with olist df
3. verify merge and check that we have the correct number of duplicated keys in the resulting merge

In [9]:
```
# how may duplicate order_id keys are in this dataframe?
dup_review_keys = reviews[reviews.order_id.duplicated()]
dup_review_keys.count()
#559 - this works out because we have 99441 unique order_ids that equals custo
mer_id's
```

Out[9]:
```
review_id                  559
order_id                   559
review_score               559
review_comment_title        18
review_comment_message     198
review_creation_date       559
review_answer_timestamp    559
dtype: int64
```

In [10]:
```
df2 = df.merge(reviews, on='order_id', how="right")
```

In [11]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100000 entries, 0 to 99999
Data columns (total 18 columns):
 #   Column                         Non-Null Count   Dtype 
---  ------                         --------------   ----- 
 0   order_id                       100000 non-null  object
 1   customer_id                    100000 non-null  object
 2   order_status                   100000 non-null  object
 3   order_purchase_timestamp       100000 non-null  object
 4   order_approved_at              99839 non-null   object
 5   order_delivered_carrier_date   98207 non-null   object
 6   order_delivered_customer_date  97013 non-null   object
 7   order_estimated_delivery_date  100000 non-null  object
 8   customer_unique_id             100000 non-null  object
 9   customer_zip_code_prefix       100000 non-null  int64 
 10  customer_city                  100000 non-null  object
 11  customer_state                 100000 non-null  object
 12  review_id                      100000 non-null  object
 13  review_score                   100000 non-null  int64 
 14  review_comment_title           11715 non-null   object
 15  review_comment_message         41753 non-null   object
 16  review_creation_date           100000 non-null  object
 17  review_answer_timestamp        100000 non-null  object
dtypes: int64(2), object(16)
memory usage: 14.5+ MB
```

In [12]:
```
df2.isnull().sum()
```

Out[12]:
```
order_id                             0
customer_id                          0
order_status                         0
order_purchase_timestamp             0
order_approved_at                  161
order_delivered_carrier_date      1793
order_delivered_customer_date     2987
order_estimated_delivery_date        0
customer_unique_id                   0
customer_zip_code_prefix             0
customer_city                        0
customer_state                       0
review_id                            0
review_score                         0
review_comment_title             88285
review_comment_message           58247
review_creation_date                 0
review_answer_timestamp              0
dtype: int64
```

In [13]:
```
# checking to make sure we are getting the correct results after merging the o
rder reviews with olist dataframe
duplicate_rows = df2[df2.order_id.duplicated(keep=False)]
duplicate_rows.count()
# 8545 records have duplicated order_ids, so we know that many reviews are par
t of multi-item orders
```

Out[13]:
```
order_id                         1114
customer_id                      1114
order_status                     1114
order_purchase_timestamp         1114
order_approved_at                1112
order_delivered_carrier_date     1094
order_delivered_customer_date    1070
order_estimated_delivery_date    1114
customer_unique_id               1114
customer_zip_code_prefix         1114
customer_city                    1114
customer_state                   1114
review_id                        1114
review_score                     1114
review_comment_title               36
review_comment_message            399
review_creation_date             1114
review_answer_timestamp          1114
dtype: int64
```

DF3

right merge the payments df to the olist df using the order_id key

1. check how many duplicate keys we have: order_id, there are 4445 more rows in this df than the original orders df
2. merge
3. confirm merge

In [14]:
```
# how many order_id's are duplicated in the payment dataframe, based on our in
itial
# kaggle data the df is 4445 recrods larger than the orders df
pymt_dups = payments[payments.order_id.duplicated(keep=False)] #false keeps th
e first record in the duplicates
pymt_dups.count()
# 7407 records were part of multi-payment orders
```

Out[14]:
```
order_id                7407
payment_sequential      7407
payment_type            7407
payment_installments    7407
payment_value           7407
dtype: int64
```

In [15]:
```python
# merge df's

df3 = df2.merge(payments, on="order_id", how="right")
df3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 104485 entries, 0 to 104484
Data columns (total 22 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   order_id                       104485 non-null  object 
 1   customer_id                    104485 non-null  object 
 2   order_status                   104485 non-null  object 
 3   order_purchase_timestamp       104485 non-null  object 
 4   order_approved_at              104309 non-null  object 
 5   order_delivered_carrier_date   102587 non-null  object 
 6   order_delivered_customer_date  101331 non-null  object 
 7   order_estimated_delivery_date  104485 non-null  object 
 8   customer_unique_id             104485 non-null  object 
 9   customer_zip_code_prefix       104485 non-null  int64  
 10  customer_city                  104485 non-null  object 
 11  customer_state                 104485 non-null  object 
 12  review_id                      104485 non-null  object 
 13  review_score                   104485 non-null  int64  
 14  review_comment_title           12151 non-null   object 
 15  review_comment_message         43623 non-null   object 
 16  review_creation_date           104485 non-null  object 
 17  review_answer_timestamp        104485 non-null  object 
 18  payment_sequential             104485 non-null  int64  
 19  payment_type                   104485 non-null  object 
 20  payment_installments           104485 non-null  int64  
 21  payment_value                  104485 non-null  float64
dtypes: float64(1), int64(4), object(17)
memory usage: 18.3+ MB
```

In [16]:
```
df3.isnull().sum()
```

Out[16]:
```
order_id                             0
customer_id                          0
order_status                         0
order_purchase_timestamp             0
order_approved_at                  176
order_delivered_carrier_date      1898
order_delivered_customer_date     3154
order_estimated_delivery_date        0
customer_unique_id                   0
customer_zip_code_prefix             0
customer_city                        0
customer_state                       0
review_id                            0
review_score                         0
review_comment_title             92334
review_comment_message           60862
review_creation_date                 0
review_answer_timestamp              0
payment_sequential                   0
payment_type                         0
payment_installments                 0
payment_value                        0
dtype: int64
```

DF4

left merge the items df to the olist df using the order_id key

1. how many duplicate keys are there: order_id, there are 13,209 more records in this df than in the original orders df
2. merge
3. confirm merge

In [17]:
```
# confirm duplicate keys
dup_item_keys = items[items.order_id.duplicated(keep=False)] #false counts the
first row of the duplicated rows
dup_item_keys.count()
# so we have about 23787 records that are part of multiple item order_ids - ju
st good to know for summary stats
```

Out[17]:
```
order_id               23787
order_item_id          23787
product_id             23787
seller_id              23787
shipping_limit_date    23787
price                  23787
freight_value          23787
dtype: int64
```

In [18]:
```python
# merge items df into olist df
# merge right because there are more entries in the items df than the orders d
f & items won't exist if they weren't ordered
df4 = df3.merge(items, on='order_id', how='left')
df4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 119148 entries, 0 to 119147
Data columns (total 28 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   order_id                       119148 non-null  object 
 1   customer_id                    119148 non-null  object 
 2   order_status                   119148 non-null  object 
 3   order_purchase_timestamp       119148 non-null  object 
 4   order_approved_at              118971 non-null  object 
 5   order_delivered_carrier_date   117062 non-null  object 
 6   order_delivered_customer_date  115727 non-null  object 
 7   order_estimated_delivery_date  119148 non-null  object 
 8   customer_unique_id             119148 non-null  object 
 9   customer_zip_code_prefix       119148 non-null  int64  
 10  customer_city                  119148 non-null  object 
 11  customer_state                 119148 non-null  object 
 12  review_id                      119148 non-null  object 
 13  review_score                   119148 non-null  int64  
 14  review_comment_title           14189 non-null   object 
 15  review_comment_message         51247 non-null   object 
 16  review_creation_date           119148 non-null  object 
 17  review_answer_timestamp        119148 non-null  object 
 18  payment_sequential             119148 non-null  int64  
 19  payment_type                   119148 non-null  object 
 20  payment_installments           119148 non-null  int64  
 21  payment_value                  119148 non-null  float64
 22  order_item_id                  118315 non-null  float64
 23  product_id                     118315 non-null  object 
 24  seller_id                      118315 non-null  object 
 25  shipping_limit_date            118315 non-null  object 
 26  price                          118315 non-null  float64
 27  freight_value                  118315 non-null  float64
dtypes: float64(4), int64(4), object(20)
memory usage: 26.4+ MB
```

In [19]:
```
df4.isna().sum()
```

Out[19]:
```
order_id                              0
customer_id                           0
order_status                          0
order_purchase_timestamp              0
order_approved_at                   177
order_delivered_carrier_date       2086
order_delivered_customer_date      3421
order_estimated_delivery_date         0
customer_unique_id                    0
customer_zip_code_prefix              0
customer_city                         0
customer_state                        0
review_id                             0
review_score                          0
review_comment_title             104959
review_comment_message            67901
review_creation_date                  0
review_answer_timestamp               0
payment_sequential                    0
payment_type                          0
payment_installments                  0
payment_value                         0
order_item_id                       833
product_id                          833
seller_id                           833
shipping_limit_date                 833
price                               833
freight_value                       833
dtype: int64
```

Products DF

Merger English translations of categories with the products dataframe

1. there are also 71 english category translations made available to us, we should merge those 2 df's
2. verify that merge
3. merge that revised df to the olist dataframe

In [20]:
```
# quick visual of the translation df
translation.info()
```

```
<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
```

In [21]:
```
unique_categories_portuguese = products.product_category_name.unique() #create
s an array objest of unique categories
len(unique_categories_portuguese) # counts the number of entries in the array
# we have 74 unique categories in portuguese and only 71 translated -
# we will have 3x missing category translations
# we will soon find out if kaggle did the homework for us and only translated
 the ones used in the overall dataset
```

Out[21]: 74

In [22]:
```
# merge products with translation
pdf = products.merge(translation, on='product_category_name', how='left')
pdf.head()
```

Out[22]:

	product_id	product_category_name	product_name_lenght	product_de
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	
3	cef67bcfe19066a932b7673e239eb23d	bebes	27.0	
4	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	37.0	

In [23]:
```
#let's make sure we didn't lose any data
pdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32951 entries, 0 to 32950
Data columns (total 10 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
 9   product_category_name_english  32328 non-null  object 
dtypes: float64(7), object(3)
memory usage: 2.8+ MB
```

In [24]:
```
# let's see what didn't translate
uniq = pdf.drop_duplicates(subset = ['product_category_name','product_category
_name_english']) # capture unique values only
uniq = uniq[['product_category_name','product_category_name_english']] # just
 want to keep these 2 columns
uniq
```

Out[24]:

	product_category_name	product_category_name_english
0	perfumaria	perfumery
1	artes	art
2	esporte_lazer	sports_leisure
3	bebes	baby
4	utilidades_domesticas	housewares
...	...	...
4713	casa_conforto_2	home_comfort_2
5821	portateis_cozinha_e_preparadores_de_alimentos	NaN
6060	seguros_e_servicos	security_and_services
7046	moveis_colchao_e_estofado	furniture_mattress_and_upholstery
18189	cds_dvds_musicais	cds_dvds_musicals

74 rows × 2 columns

In [25]:
```
# looks like we are missing some english translations for categories - let's s
ee how many
# create a boolean variable to filter the the products by the NaNs from the en
glish translatioins
no_eng_trans = uniq.product_category_name_english.isnull()

#filter the dataframe by the returned NA lines for english translations
no_trans = uniq[no_eng_trans]

#show how many orders were cancelled
no_trans.head()
```

Out[25]:

	product_category_name	product_category_name_english
105	NaN	NaN
1628	pc_gamer	NaN
5821	portateis_cozinha_e_preparadores_de_alimentos	NaN

In [26]:
```
#let's go ahead and impute that pc_gamer and
# also using google translate, the other one is: portable kitchen and food pre
parers

# disable chained assignments
pd.options.mode.chained_assignment = None

for i in range(len(pdf)):
    if pdf.iloc[i,1] == "pc_gamer":
        pdf.iloc[i,9] = "pc_gamer"
    else:
        if pdf.iloc[i,1] == "portateis_cozinha_e_preparadores_de_alimentos":
            pdf.iloc[i,9] = "portable_kitchen_and_food_preparers"
```

In [27]:
```
pdf.isnull().sum()
# looks like everything is matching up nicely here
```

Out[27]:
```
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
product_category_name_english    610
dtype: int64
```

In [28]:
```
unq_cat = pdf.product_category_name_english.unique()
for x in range(len(unq_cat)):
    print(unq_cat[x])
# ok looks like we successfully added those 2 categories to the english transl
ation list
```

perfumery
art
sports_leisure
baby
housewares
musical_instruments
cool_stuff
furniture_decor
home_appliances
toys
bed_bath_table
construction_tools_safety
computers_accessories
health_beauty
luggage_accessories
garden_tools
office_furniture
auto
electronics
fashion_shoes
telephony
stationery
fashion_bags_accessories
computers
home_construction
watches_gifts
construction_tools_construction
pet_shop
small_appliances
agro_industry_and_commerce
nan
furniture_living_room
signaling_and_security
air_conditioning
consoles_games
books_general_interest
costruction_tools_tools
fashion_underwear_beach
fashion_male_clothing
kitchen_dining_laundry_garden_furniture
industry_commerce_and_business
fixed_telephony
construction_tools_lights
books_technical
home_appliances_2
party_supplies
drinks
market_place
la_cuisine
costruction_tools_garden
fashio_female_clothing
home_confort
audio
food_drink
music
food
tablets_printing_image

```
books_imported
small_appliances_home_oven_and_coffee
fashion_sport
christmas_supplies
fashion_childrens_clothes
dvds_blu_ray
arts_and_craftmanship
pc_gamer
furniture_bedroom
cine_photo
diapers_and_hygiene
flowers
home_comfort_2
portable_kitchen_and_food_preparers
security_and_services
furniture_mattress_and_upholstery
cds_dvds_musicals
```

In [29]:

```python
# drop the original product category name in portuguese
pdf = pdf.drop(['product_category_name'], axis=1) # axis=1 means column
pdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32951 entries, 0 to 32950
Data columns (total 9 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   product_id                     32951 non-null  object 
 1   product_name_lenght            32341 non-null  float64
 2   product_description_lenght     32341 non-null  float64
 3   product_photos_qty             32341 non-null  float64
 4   product_weight_g               32949 non-null  float64
 5   product_length_cm              32949 non-null  float64
 6   product_height_cm              32949 non-null  float64
 7   product_width_cm               32949 non-null  float64
 8   product_category_name_english  32341 non-null  object 
dtypes: float64(7), object(2)
memory usage: 3.8+ MB
```

In [30]:
```
# rename category column
pdf = pdf.rename(columns={'product_name_lenght':'product_name_length',
                          'product_description_lenght': 'product_description_l
ength',
                          'product_category_name_english': "product_category_e
nglish"})
pdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 32951 entries, 0 to 32950
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   product_id                  32951 non-null  object 
 1   product_name_length         32341 non-null  float64
 2   product_description_length  32341 non-null  float64
 3   product_photos_qty          32341 non-null  float64
 4   product_weight_g            32949 non-null  float64
 5   product_length_cm           32949 non-null  float64
 6   product_height_cm           32949 non-null  float64
 7   product_width_cm            32949 non-null  float64
 8   product_category_english    32341 non-null  object 
dtypes: float64(7), object(2)
memory usage: 2.5+ MB
```

DF5

left merge the NEW REVISED products dataframe with the olist dateframe

In [31]:
```python
df5 = df4.merge(pdf, on="product_id", how='left')
df5.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 119148 entries, 0 to 119147
Data columns (total 36 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   order_id                       119148 non-null  object 
 1   customer_id                    119148 non-null  object 
 2   order_status                   119148 non-null  object 
 3   order_purchase_timestamp       119148 non-null  object 
 4   order_approved_at              118971 non-null  object 
 5   order_delivered_carrier_date   117062 non-null  object 
 6   order_delivered_customer_date  115727 non-null  object 
 7   order_estimated_delivery_date  119148 non-null  object 
 8   customer_unique_id             119148 non-null  object 
 9   customer_zip_code_prefix       119148 non-null  int64  
 10  customer_city                  119148 non-null  object 
 11  customer_state                 119148 non-null  object 
 12  review_id                      119148 non-null  object 
 13  review_score                   119148 non-null  int64  
 14  review_comment_title           14189 non-null   object 
 15  review_comment_message         51247 non-null   object 
 16  review_creation_date           119148 non-null  object 
 17  review_answer_timestamp        119148 non-null  object 
 18  payment_sequential             119148 non-null  int64  
 19  payment_type                   119148 non-null  object 
 20  payment_installments           119148 non-null  int64  
 21  payment_value                  119148 non-null  float64
 22  order_item_id                  118315 non-null  float64
 23  product_id                     118315 non-null  object 
 24  seller_id                      118315 non-null  object 
 25  shipping_limit_date            118315 non-null  object 
 26  price                          118315 non-null  float64
 27  freight_value                  118315 non-null  float64
 28  product_name_length            116606 non-null  float64
 29  product_description_length     116606 non-null  float64
 30  product_photos_qty             116606 non-null  float64
 31  product_weight_g               118295 non-null  float64
 32  product_length_cm              118295 non-null  float64
 33  product_height_cm              118295 non-null  float64
 34  product_width_cm               118295 non-null  float64
 35  product_category_english       116606 non-null  object 
dtypes: float64(11), int64(4), object(21)
memory usage: 33.6+ MB
```

In [32]:
```
df5.isna().sum()
```

Out[32]:
```
order_id                              0
customer_id                           0
order_status                          0
order_purchase_timestamp              0
order_approved_at                   177
order_delivered_carrier_date       2086
order_delivered_customer_date      3421
order_estimated_delivery_date         0
customer_unique_id                    0
customer_zip_code_prefix              0
customer_city                         0
customer_state                        0
review_id                             0
review_score                          0
review_comment_title             104959
review_comment_message            67901
review_creation_date                  0
review_answer_timestamp               0
payment_sequential                    0
payment_type                          0
payment_installments                  0
payment_value                         0
order_item_id                       833
product_id                          833
seller_id                           833
shipping_limit_date                 833
price                               833
freight_value                       833
product_name_length                2542
product_description_length         2542
product_photos_qty                 2542
product_weight_g                    853
product_length_cm                   853
product_height_cm                   853
product_width_cm                    853
product_category_english           2542
dtype: int64
```

DF6

left merge Sellers df with the olist df

There are 3,095 records in the seller df

In [33]:
```python
sellers.isna().sum()
#no missing values
```

Out[33]:
```
seller_id                 0
seller_zip_code_prefix    0
seller_city               0
seller_state              0
dtype: int64
```

In [34]:
```python
#merge df's
df6 = df5.merge(sellers, on="seller_id", how='left')
df6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 119148 entries, 0 to 119147
Data columns (total 39 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   order_id                       119148 non-null  object 
 1   customer_id                    119148 non-null  object 
 2   order_status                   119148 non-null  object 
 3   order_purchase_timestamp       119148 non-null  object 
 4   order_approved_at              118971 non-null  object 
 5   order_delivered_carrier_date   117062 non-null  object 
 6   order_delivered_customer_date  115727 non-null  object 
 7   order_estimated_delivery_date  119148 non-null  object 
 8   customer_unique_id             119148 non-null  object 
 9   customer_zip_code_prefix       119148 non-null  int64  
 10  customer_city                  119148 non-null  object 
 11  customer_state                 119148 non-null  object 
 12  review_id                      119148 non-null  object 
 13  review_score                   119148 non-null  int64  
 14  review_comment_title           14189 non-null   object 
 15  review_comment_message         51247 non-null   object 
 16  review_creation_date           119148 non-null  object 
 17  review_answer_timestamp        119148 non-null  object 
 18  payment_sequential             119148 non-null  int64  
 19  payment_type                   119148 non-null  object 
 20  payment_installments           119148 non-null  int64  
 21  payment_value                  119148 non-null  float64
 22  order_item_id                  118315 non-null  float64
 23  product_id                     118315 non-null  object 
 24  seller_id                      118315 non-null  object 
 25  shipping_limit_date            118315 non-null  object 
 26  price                          118315 non-null  float64
 27  freight_value                  118315 non-null  float64
 28  product_name_length            116606 non-null  float64
 29  product_description_length     116606 non-null  float64
 30  product_photos_qty             116606 non-null  float64
 31  product_weight_g               118295 non-null  float64
 32  product_length_cm              118295 non-null  float64
 33  product_height_cm              118295 non-null  float64
 34  product_width_cm               118295 non-null  float64
 35  product_category_english       116606 non-null  object 
 36  seller_zip_code_prefix         118315 non-null  float64
 37  seller_city                    118315 non-null  object 
 38  seller_state                   118315 non-null  object 
dtypes: float64(12), int64(4), object(23)
memory usage: 36.4+ MB
```

In [35]:
```
df6.isna().sum()
```

Out[35]:
```
order_id                              0
customer_id                           0
order_status                          0
order_purchase_timestamp              0
order_approved_at                   177
order_delivered_carrier_date       2086
order_delivered_customer_date      3421
order_estimated_delivery_date         0
customer_unique_id                    0
customer_zip_code_prefix              0
customer_city                         0
customer_state                        0
review_id                             0
review_score                          0
review_comment_title             104959
review_comment_message            67901
review_creation_date                  0
review_answer_timestamp               0
payment_sequential                    0
payment_type                          0
payment_installments                  0
payment_value                         0
order_item_id                       833
product_id                          833
seller_id                           833
shipping_limit_date                 833
price                               833
freight_value                       833
product_name_length                2542
product_description_length         2542
product_photos_qty                 2542
product_weight_g                    853
product_length_cm                   853
product_height_cm                   853
product_width_cm                    853
product_category_english           2542
seller_zip_code_prefix              833
seller_city                         833
seller_state                        833
dtype: int64
```

Create olist dataframe

Merge Order Referce

1. **df**: inner merge customers + orders as df
2. **df2**: right merge df + reviews as df2
3. **df3**: right merge df2 + payments as df3
4. **df4**: right merge df3 + items as df4
5. **df5**: left merge df4 + pdf as df5
6. **df6**: left merge df5 + sellers as df6
7. **olist**: set olist = df6

In [36]:
```
# now we have starting points we can refer back to in case we get lost somewhe
re in the shuffle
olist = df6
```

In [37]:
```
#changing attributes data types
continuous_features = ['price','freight_value', 'payment_sequential','payment_
installments','payment_value',
                      'product_name_length','product_description_length', 'pro
duct_photos_qty','product_weight_g',
                      'product_length_cm','product_height_cm','product_width_c
m', 'review_score']

cat_features = ['order_status', 'customer_city', 'customer_state', 'customer_z
ip_code_prefix', 'seller_zip_code_prefix',
                'seller_city', 'seller_state', 'product_category_english','rev
iew_id',
               'review_comment_title','review_comment_message','payment_type',
               'order_item_id', 'product_id','seller_id','order_id', 'customer
_id','customer_unique_id']

date_features = ['order_purchase_timestamp', 'order_approved_at', 'order_deliv
ered_carrier_date', 'order_delivered_customer_date',
                 'order_estimated_delivery_date', 'shipping_limit_date', 'revi
ew_creation_date', 'review_answer_timestamp']
```

In [38]:
```python
# use the "astype" function to change the variable type
olist[continuous_features] = olist.copy()[continuous_features].astype(np.float
64)
olist[cat_features] = olist.copy()[cat_features].astype("category")


olist.info() # now our data looks better!!
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 119148 entries, 0 to 119147
Data columns (total 39 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_id                       119148 non-null  category
 1   customer_id                    119148 non-null  category
 2   order_status                   119148 non-null  category
 3   order_purchase_timestamp       119148 non-null  object
 4   order_approved_at              118971 non-null  object
 5   order_delivered_carrier_date   117062 non-null  object
 6   order_delivered_customer_date  115727 non-null  object
 7   order_estimated_delivery_date  119148 non-null  object
 8   customer_unique_id             119148 non-null  category
 9   customer_zip_code_prefix       119148 non-null  category
 10  customer_city                  119148 non-null  category
 11  customer_state                 119148 non-null  category
 12  review_id                      119148 non-null  category
 13  review_score                   119148 non-null  float64
 14  review_comment_title           14189 non-null   category
 15  review_comment_message         51247 non-null   category
 16  review_creation_date           119148 non-null  object
 17  review_answer_timestamp        119148 non-null  object
 18  payment_sequential             119148 non-null  float64
 19  payment_type                   119148 non-null  category
 20  payment_installments           119148 non-null  float64
 21  payment_value                  119148 non-null  float64
 22  order_item_id                  118315 non-null  category
 23  product_id                     118315 non-null  category
 24  seller_id                      118315 non-null  category
 25  shipping_limit_date            118315 non-null  object
 26  price                          118315 non-null  float64
 27  freight_value                  118315 non-null  float64
 28  product_name_length            116606 non-null  float64
 29  product_description_length     116606 non-null  float64
 30  product_photos_qty             116606 non-null  float64
 31  product_weight_g               118295 non-null  float64
 32  product_length_cm              118295 non-null  float64
 33  product_height_cm              118295 non-null  float64
 34  product_width_cm               118295 non-null  float64
 35  product_category_english       116606 non-null  category
 36  seller_zip_code_prefix         118315 non-null  category
 37  seller_city                    118315 non-null  category
 38  seller_state                   118315 non-null  category
dtypes: category(18), float64(13), object(8)
memory usage: 42.2+ MB
```

Fix Dates as datetime64[ns]

In [39]:
```
# fix dates

for i in date_features:
    olist.loc[:,i] = pd.to_datetime(olist.copy().loc[:,i], errors="coerce")
```

In [40]:
```
olist.info() # now our data looks better!!
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 119148 entries, 0 to 119147
Data columns (total 39 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_id                       119148 non-null  category
 1   customer_id                    119148 non-null  category
 2   order_status                   119148 non-null  category
 3   order_purchase_timestamp       119148 non-null  datetime64[ns]
 4   order_approved_at              118971 non-null  datetime64[ns]
 5   order_delivered_carrier_date   117062 non-null  datetime64[ns]
 6   order_delivered_customer_date  115727 non-null  datetime64[ns]
 7   order_estimated_delivery_date  119148 non-null  datetime64[ns]
 8   customer_unique_id             119148 non-null  category
 9   customer_zip_code_prefix       119148 non-null  category
 10  customer_city                  119148 non-null  category
 11  customer_state                 119148 non-null  category
 12  review_id                      119148 non-null  category
 13  review_score                   119148 non-null  float64
 14  review_comment_title           14189 non-null   category
 15  review_comment_message         51247 non-null   category
 16  review_creation_date           119148 non-null  datetime64[ns]
 17  review_answer_timestamp        119148 non-null  datetime64[ns]
 18  payment_sequential             119148 non-null  float64
 19  payment_type                   119148 non-null  category
 20  payment_installments           119148 non-null  float64
 21  payment_value                  119148 non-null  float64
 22  order_item_id                  118315 non-null  category
 23  product_id                     118315 non-null  category
 24  seller_id                      118315 non-null  category
 25  shipping_limit_date            118315 non-null  datetime64[ns]
 26  price                          118315 non-null  float64
 27  freight_value                  118315 non-null  float64
 28  product_name_length            116606 non-null  float64
 29  product_description_length     116606 non-null  float64
 30  product_photos_qty             116606 non-null  float64
 31  product_weight_g               118295 non-null  float64
 32  product_length_cm              118295 non-null  float64
 33  product_height_cm              118295 non-null  float64
 34  product_width_cm               118295 non-null  float64
 35  product_category_english       116606 non-null  category
 36  seller_zip_code_prefix         118315 non-null  category
 37  seller_city                    118315 non-null  category
 38  seller_state                   118315 non-null  category
dtypes: category(18), datetime64[ns](8), float64(13)
memory usage: 42.2 MB
```

```
In [41]: # count NAs in the dataframe by column
         olist.isnull().sum()
```

```
Out[41]: order_id                              0
         customer_id                           0
         order_status                          0
         order_purchase_timestamp              0
         order_approved_at                   177
         order_delivered_carrier_date       2086
         order_delivered_customer_date      3421
         order_estimated_delivery_date         0
         customer_unique_id                    0
         customer_zip_code_prefix              0
         customer_city                         0
         customer_state                        0
         review_id                             0
         review_score                          0
         review_comment_title             104959
         review_comment_message            67901
         review_creation_date                  0
         review_answer_timestamp               0
         payment_sequential                    0
         payment_type                          0
         payment_installments                  0
         payment_value                         0
         order_item_id                       833
         product_id                          833
         seller_id                           833
         shipping_limit_date                 833
         price                               833
         freight_value                       833
         product_name_length                2542
         product_description_length         2542
         product_photos_qty                 2542
         product_weight_g                    853
         product_length_cm                   853
         product_height_cm                   853
         product_width_cm                    853
         product_category_english           2542
         seller_zip_code_prefix              833
         seller_city                         833
         seller_state                        833
         dtype: int64
```

Add Total*Order* Amt to olist dataframe

In [42]:
```
olist['tot_order_amt'] = olist.price + olist.freight_value
olist[['payment_type','price','freight_value','payment_value','tot_order_amt'
]].head(10)
```

Out[42]:

	payment_type	price	freight_value	payment_value	tot_order_amt
0	credit_card	29.99	8.72	18.12	38.71
1	voucher	29.99	8.72	2.00	38.71
2	voucher	29.99	8.72	18.59	38.71
3	boleto	118.70	22.76	141.46	141.46
4	credit_card	159.90	19.22	179.12	179.12
5	credit_card	45.00	27.20	72.20	72.20
6	credit_card	19.90	8.72	28.62	28.62
7	credit_card	147.90	27.36	175.26	175.26
8	credit_card	49.90	16.05	65.95	65.95
9	credit_card	59.99	15.17	75.16	75.16

In [43]:
```
pmt = olist[olist.order_id=='e481f51cbdc54678b7cc49136f2d6af7']
pmt=pmt[['order_id','product_id','payment_sequential','payment_type','price',
'freight_value','payment_value','tot_order_amt']]
pmt
```

Out[43]:

	order_id	product_id	payment_sequential	pa
0	e481f51cbdc54678b7cc49136f2d6af7	87285b34884572647811a353c7ac498a	1.0	
1	e481f51cbdc54678b7cc49136f2d6af7	87285b34884572647811a353c7ac498a	3.0	
2	e481f51cbdc54678b7cc49136f2d6af7	87285b34884572647811a353c7ac498a	2.0	

Data Quality

Verify data quality:

- Explain any missing values, duplicate data, and outliers.
- Are those mistakes?
- How do you deal with these problems?
- Be specific.

Before we begin to look at missing values, we decided based on our business objectives to look at how Olist can control logistics (shipping) costs, manager logistics partners and increase sales OR help sellers make more money through analysis of marketplace sales, we want to focus on completed sales only and we define that as *order_status* = *delivered* as well as a timestamp in the *order_delivered_customer_date*

In [44]:
```
# create a backup dateframe
olist_backup = olist
```

In [45]:
```
# first let's just grab all records with order_status of delivered since that
 attribute has no missing values
olist = olist[olist.order_status == "delivered"]
print(olist.info())
print('------------------------')
print('order_status', olist.order_status.describe())
# should remove order_status now that it is only one level, but I want to wait
and do that at the end.
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 115728 entries, 0 to 119147
Data columns (total 40 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_id                       115728 non-null  category
 1   customer_id                    115728 non-null  category
 2   order_status                   115728 non-null  category
 3   order_purchase_timestamp       115728 non-null  datetime64[ns]
 4   order_approved_at              115713 non-null  datetime64[ns]
 5   order_delivered_carrier_date   115726 non-null  datetime64[ns]
 6   order_delivered_customer_date  115720 non-null  datetime64[ns]
 7   order_estimated_delivery_date  115728 non-null  datetime64[ns]
 8   customer_unique_id             115728 non-null  category
 9   customer_zip_code_prefix       115728 non-null  category
 10  customer_city                  115728 non-null  category
 11  customer_state                 115728 non-null  category
 12  review_id                      115728 non-null  category
 13  review_score                   115728 non-null  float64
 14  review_comment_title           13751 non-null   category
 15  review_comment_message         48961 non-null   category
 16  review_creation_date           115728 non-null  datetime64[ns]
 17  review_answer_timestamp        115728 non-null  datetime64[ns]
 18  payment_sequential             115728 non-null  float64
 19  payment_type                   115728 non-null  category
 20  payment_installments           115728 non-null  float64
 21  payment_value                  115728 non-null  float64
 22  order_item_id                  115728 non-null  category
 23  product_id                     115728 non-null  category
 24  seller_id                      115728 non-null  category
 25  shipping_limit_date            115728 non-null  datetime64[ns]
 26  price                          115728 non-null  float64
 27  freight_value                  115728 non-null  float64
 28  product_name_length            114090 non-null  float64
 29  product_description_length     114090 non-null  float64
 30  product_photos_qty             114090 non-null  float64
 31  product_weight_g               115708 non-null  float64
 32  product_length_cm              115708 non-null  float64
 33  product_height_cm              115708 non-null  float64
 34  product_width_cm               115708 non-null  float64
 35  product_category_english       114090 non-null  category
 36  seller_zip_code_prefix         115728 non-null  category
 37  seller_city                    115728 non-null  category
 38  seller_state                   115728 non-null  category
 39  tot_order_amt                  115728 non-null  float64
dtypes: category(18), datetime64[ns](8), float64(14)
memory usage: 42.3 MB
None
------------------------
order_status count        115728
unique            1
top       delivered
freq         115728
Name: order_status, dtype: object
```

In [46]:
```
olist.isna().sum()
```

Out[46]:
```
order_id                              0
customer_id                           0
order_status                          0
order_purchase_timestamp              0
order_approved_at                    15
order_delivered_carrier_date          2
order_delivered_customer_date         8
order_estimated_delivery_date         0
customer_unique_id                    0
customer_zip_code_prefix              0
customer_city                         0
customer_state                        0
review_id                             0
review_score                          0
review_comment_title             101977
review_comment_message            66767
review_creation_date                  0
review_answer_timestamp               0
payment_sequential                    0
payment_type                          0
payment_installments                  0
payment_value                         0
order_item_id                         0
product_id                            0
seller_id                             0
shipping_limit_date                   0
price                                 0
freight_value                         0
product_name_length                1638
product_description_length         1638
product_photos_qty                 1638
product_weight_g                     20
product_length_cm                    20
product_height_cm                    20
product_width_cm                     20
product_category_english           1638
seller_zip_code_prefix                0
seller_city                           0
seller_state                          0
tot_order_amt                         0
dtype: int64
```

In [47]:
```python
# let's also drop the 8 records missing a customer delivered timestamp
olist = olist[olist.order_delivered_customer_date.notnull()]
olist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 115720 entries, 0 to 119147
Data columns (total 40 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_id                       115720 non-null  category
 1   customer_id                    115720 non-null  category
 2   order_status                   115720 non-null  category
 3   order_purchase_timestamp       115720 non-null  datetime64[ns]
 4   order_approved_at              115705 non-null  datetime64[ns]
 5   order_delivered_carrier_date   115719 non-null  datetime64[ns]
 6   order_delivered_customer_date  115720 non-null  datetime64[ns]
 7   order_estimated_delivery_date  115720 non-null  datetime64[ns]
 8   customer_unique_id             115720 non-null  category
 9   customer_zip_code_prefix       115720 non-null  category
 10  customer_city                  115720 non-null  category
 11  customer_state                 115720 non-null  category
 12  review_id                      115720 non-null  category
 13  review_score                   115720 non-null  float64
 14  review_comment_title           13747 non-null   category
 15  review_comment_message         48956 non-null   category
 16  review_creation_date           115720 non-null  datetime64[ns]
 17  review_answer_timestamp        115720 non-null  datetime64[ns]
 18  payment_sequential             115720 non-null  float64
 19  payment_type                   115720 non-null  category
 20  payment_installments           115720 non-null  float64
 21  payment_value                  115720 non-null  float64
 22  order_item_id                  115720 non-null  category
 23  product_id                     115720 non-null  category
 24  seller_id                      115720 non-null  category
 25  shipping_limit_date            115720 non-null  datetime64[ns]
 26  price                          115720 non-null  float64
 27  freight_value                  115720 non-null  float64
 28  product_name_length            114082 non-null  float64
 29  product_description_length     114082 non-null  float64
 30  product_photos_qty             114082 non-null  float64
 31  product_weight_g               115700 non-null  float64
 32  product_length_cm              115700 non-null  float64
 33  product_height_cm              115700 non-null  float64
 34  product_width_cm               115700 non-null  float64
 35  product_category_english       114082 non-null  category
 36  seller_zip_code_prefix         115720 non-null  category
 37  seller_city                    115720 non-null  category
 38  seller_state                   115720 non-null  category
 39  tot_order_amt                  115720 non-null  float64
dtypes: category(18), datetime64[ns](8), float64(14)
memory usage: 42.3 MB
```

In [48]:
```
olist.isna().sum()
```

Out[48]:
```
order_id                              0
customer_id                           0
order_status                          0
order_purchase_timestamp              0
order_approved_at                    15
order_delivered_carrier_date          1
order_delivered_customer_date         0
order_estimated_delivery_date         0
customer_unique_id                    0
customer_zip_code_prefix              0
customer_city                         0
customer_state                        0
review_id                             0
review_score                          0
review_comment_title             101973
review_comment_message            66764
review_creation_date                  0
review_answer_timestamp               0
payment_sequential                    0
payment_type                          0
payment_installments                  0
payment_value                         0
order_item_id                         0
product_id                            0
seller_id                             0
shipping_limit_date                   0
price                                 0
freight_value                         0
product_name_length                1638
product_description_length         1638
product_photos_qty                 1638
product_weight_g                     20
product_length_cm                    20
product_height_cm                    20
product_width_cm                     20
product_category_english           1638
seller_zip_code_prefix                0
seller_city                           0
seller_state                          0
tot_order_amt                         0
dtype: int64
```

NOTE:

That cleaned up a lot of missing values and didn't reduce the dataset very much.

Fix the 1 missing order_delivered_carrier value

In [49]:
```python
olist[olist.order_delivered_carrier_date.isnull()]
```

Out[49]:

	order_id	customer_id	order_status	orde
87582	2aa91108853cecb43c84a5dc5b277475	afeb16c7f46396c0ed54acb45ccaaa40	delivered	

1 rows × 40 columns

In [50]:
```python
olist.iloc[87582,4]
# strange, there is a time there - looks like it is a converting issue
```

Out[50]:
```
Timestamp('2018-04-19 22:11:37')
```

In [51]:
```python
olist.iloc[87582,4] = pd.to_datetime(olist.iloc[87582,4], unit='ns')
```

In [52]:
```python
olist[olist.order_delivered_carrier_date.isnull()]
# still not converting, let's just delete it
```

Out[52]:

	order_id	customer_id	order_status	orde
87582	2aa91108853cecb43c84a5dc5b277475	afeb16c7f46396c0ed54acb45ccaaa40	delivered	

1 rows × 40 columns

In [53]:
```
olist = olist[olist.order_delivered_carrier_date.notnull()]
olist.isna().sum()
```

Out[53]:
```
order_id                              0
customer_id                           0
order_status                          0
order_purchase_timestamp              0
order_approved_at                    15
order_delivered_carrier_date          0
order_delivered_customer_date         0
order_estimated_delivery_date         0
customer_unique_id                    0
customer_zip_code_prefix              0
customer_city                         0
customer_state                        0
review_id                             0
review_score                          0
review_comment_title             101972
review_comment_message            66763
review_creation_date                  0
review_answer_timestamp               0
payment_sequential                    0
payment_type                          0
payment_installments                  0
payment_value                         0
order_item_id                         0
product_id                            0
seller_id                             0
shipping_limit_date                   0
price                                 0
freight_value                         0
product_name_length                1638
product_description_length         1638
product_photos_qty                 1638
product_weight_g                     20
product_length_cm                    20
product_height_cm                    20
product_width_cm                     20
product_category_english           1638
seller_zip_code_prefix                0
seller_city                           0
seller_state                          0
tot_order_amt                         0
dtype: int64
```

Explore the 15 missing values for order_approved_at

In [54]:
```python
# what are those 15 missing values for order_approved_at?
missing_approved = olist[olist['order_approved_at'].isnull()]
missing_approved
```

Out[54]:

	order_id	customer_id	order_status	or
6292	e04abd8149ef81b95221e88f6ed9ab6a	2127dc6603ac33544953ef05ec155771	delivered	
19765	8a9adc69528e1001fc68dd0aaebbb54a	4c1ccc74e00993733742a3c786dc3c1f	delivered	
22728	7013bcfc1c97fe719a7b5e05e61c12db	2941af76d38100e0f8740a374f1a5dc3	delivered	
27092	5cf925b116421afa85ee25e99b4c34fb	29c35fc91fc13fb5073c8f30505d860d	delivered	
27691	12a95a3c06dbaec84bcfb0e2da5d228a	1e101e0daffaddce8159d25a8e53f2b2	delivered	
32110	c1d4211b3dae76144deccd6c74144a88	684cb238dc5b5d6366244e0e0776b450	delivered	
45903	d69e5d356402adc8cf17e08b5033acfb	68d081753ad4fe22fc4d410a9eb1ca01	delivered	
47166	d77031d6a3c8a52f019764e68f211c69	0bf35cac6cc7327065da879e2d90fae8	delivered	
57994	7002a78c79c519ac54022d4f8a65e6e8	d5de688c321096d15508faae67a27051	delivered	
73781	2eecb0d85f281280f79fa00f9cec1a95	a3d3c38e58b9d2dfb9207cab690b6310	delivered	
75350	51eb2eebd5d76a24625b31c33dd41449	07a2a7e0f63fd8cb757ed77d4245623c	delivered	
80968	88083e8f64d95b932164187484d90212	f67cd1a215aae2a1074638bbd35a223a	delivered	
80969	88083e8f64d95b932164187484d90212	f67cd1a215aae2a1074638bbd35a223a	delivered	
86635	3c0b8706b065f9919d0505d3b3343881	d85919cb3c0529589c6fa617f5f43281	delivered	
101696	2babbb4b15e6d2dfe95e2de765c97bce	74bebaf46603f9340e3b50c6b086f992	delivered	

15 rows × 40 columns

In [55]:
```python
# looks like we can impute those values...let's see what the average time is b
etween and order purchase and approval
purchase_to_approve = olist[['order_purchase_timestamp','order_approved_at','o
rder_status']]
purchase_to_approve.dtypes
```

Out[55]:
```
order_purchase_timestamp    datetime64[ns]
order_approved_at           datetime64[ns]
order_status                      category
dtype: object
```

In [56]:
```
# let's remove those 14 missing values
purchase_to_approve[purchase_to_approve.order_approved_at.isnull()]
```

Out[56]:

	order_purchase_timestamp	order_approved_at	order_status
6292	2017-02-18 14:40:00	NaT	delivered
19765	2017-02-18 12:45:31	NaT	delivered
22728	2017-02-18 13:29:47	NaT	delivered
27092	2017-02-18 16:48:35	NaT	delivered
27691	2017-02-17 13:05:55	NaT	delivered
32110	2017-01-19 12:48:08	NaT	delivered
45903	2017-02-19 01:28:47	NaT	delivered
47166	2017-02-18 11:04:19	NaT	delivered
57994	2017-01-19 22:26:59	NaT	delivered
73781	2017-02-17 17:21:55	NaT	delivered
75350	2017-02-18 15:52:27	NaT	delivered
80968	2017-02-18 22:49:19	NaT	delivered
80969	2017-02-18 22:49:19	NaT	delivered
86635	2017-02-17 15:53:27	NaT	delivered
101696	2017-02-18 17:15:03	NaT	delivered

In [57]:
```
# removing na's
purchase_to_approve = purchase_to_approve.dropna(axis=0)
```

In [58]:
```
purchase_to_approve[purchase_to_approve.order_approved_at.isnull()].sum()
```

Out[58]:
```
order_purchase_timestamp    0.0
order_approved_at           0.0
order_status                0.0
dtype: float64
```

In [59]:
```
# let's get some averages
purchase_to_approve['pta_time'] = purchase_to_approve['order_approved_at'] - p
urchase_to_approve['order_purchase_timestamp']
# convert timedelta to numeric value for averaging
purchase_to_approve['new'] = purchase_to_approve['pta_time'].values.astype(np.
int64)

purchase_to_approve.head()
```

Out[59]:

	order_purchase_timestamp	order_approved_at	order_status	pta_time	new
0	2017-10-02 10:56:33	2017-10-02 11:07:15	delivered	0 days 00:10:42	642000000000
1	2017-10-02 10:56:33	2017-10-02 11:07:15	delivered	0 days 00:10:42	642000000000
2	2017-10-02 10:56:33	2017-10-02 11:07:15	delivered	0 days 00:10:42	642000000000
3	2018-07-24 20:41:37	2018-07-26 03:24:27	delivered	1 days 06:42:50	110570000000000
4	2018-08-08 08:38:49	2018-08-08 08:55:23	delivered	0 days 00:16:34	994000000000

In [60]:
```
# average time to approve purchase from order
avg_time_to_app = purchase_to_approve['new'].sum()/purchase_to_approve['new'].
count()
avg_time_to_app
pd.to_timedelta(avg_time_to_app)
# looks like about 10.5 hours is the average so we can just copy the purchase
 time stamp to the approved for those 14
```

Out[60]: Timedelta('0 days 10:25:58.381836')

In [61]:
```
 #fill in missing values if indices match
olist.order_approved_at = olist.order_approved_at.fillna(olist.order_purchase_
timestamp)
```

```
In [62]: olist.isnull().sum()
```

```
Out[62]: order_id                              0
customer_id                           0
order_status                          0
order_purchase_timestamp              0
order_approved_at                     0
order_delivered_carrier_date          0
order_delivered_customer_date         0
order_estimated_delivery_date         0
customer_unique_id                    0
customer_zip_code_prefix              0
customer_city                         0
customer_state                        0
review_id                             0
review_score                          0
review_comment_title             101972
review_comment_message            66763
review_creation_date                  0
review_answer_timestamp               0
payment_sequential                    0
payment_type                          0
payment_installments                  0
payment_value                         0
order_item_id                         0
product_id                            0
seller_id                             0
shipping_limit_date                   0
price                                 0
freight_value                         0
product_name_length                1638
product_description_length         1638
product_photos_qty                 1638
product_weight_g                     20
product_length_cm                    20
product_height_cm                    20
product_width_cm                     20
product_category_english           1638
seller_zip_code_prefix                0
seller_city                           0
seller_state                          0
tot_order_amt                         0
dtype: int64
```

Freight Discussion

Because our research showed that shipping costs are a big factor (or they were during the period this dataset covers), any values missing for package weight and size should be removed since we have no way to compute those

Explore those 20 records missing package attributes

In [63]:

```python
# which rows have NAs - specifically those with just 20, I am thinking they ar
e the same rows
missing_pkg_desc = olist[olist['product_weight_g'].isnull()]
missing_pkg_desc[['product_id']]
```

Out[63]:

	product_id
8761	5eb564652db742ff8f28759cd8d2652a
9260	5eb564652db742ff8f28759cd8d2652a
9261	5eb564652db742ff8f28759cd8d2652a
22569	5eb564652db742ff8f28759cd8d2652a
22570	5eb564652db742ff8f28759cd8d2652a
22571	5eb564652db742ff8f28759cd8d2652a
33766	5eb564652db742ff8f28759cd8d2652a
38101	5eb564652db742ff8f28759cd8d2652a
42901	5eb564652db742ff8f28759cd8d2652a
42902	5eb564652db742ff8f28759cd8d2652a
45964	09ff539a621711667c43eba6a3bd8466
49381	5eb564652db742ff8f28759cd8d2652a
54768	5eb564652db742ff8f28759cd8d2652a
65846	5eb564652db742ff8f28759cd8d2652a
81377	5eb564652db742ff8f28759cd8d2652a
83792	5eb564652db742ff8f28759cd8d2652a
86973	5eb564652db742ff8f28759cd8d2652a
87530	5eb564652db742ff8f28759cd8d2652a
89079	5eb564652db742ff8f28759cd8d2652a
106819	5eb564652db742ff8f28759cd8d2652a

In [64]:
```
# 19 of the 20 are all the same product_id, let's see if we can find informati
on on that in the rest of the dataset
olist[olist.product_id=='5eb564652db742ff8f28759cd8d2652a']
```

Out[64]:

	order_id	customer_id	order_status	or
8761	6e150190fbe04c642a9cf0b80d83ee16	135a42a465867ff932f1222f71a3efb2	delivered	
9260	d38dcb503cd4ddc6ce7702552918bd8f	b0a3a02fe893d9a9385a98db1348244b	delivered	
9261	d38dcb503cd4ddc6ce7702552918bd8f	b0a3a02fe893d9a9385a98db1348244b	delivered	
22569	ddf16d77e858a32f36e10c289a28ef61	84cc013dd1790fdafb0fa598695cf3c3	delivered	
22570	ddf16d77e858a32f36e10c289a28ef61	84cc013dd1790fdafb0fa598695cf3c3	delivered	
22571	ddf16d77e858a32f36e10c289a28ef61	84cc013dd1790fdafb0fa598695cf3c3	delivered	
33766	1521c6bb7b1028154c8c67cf80fa809f	ca29b2bf57243228e98eab2dab805ae9	delivered	
38101	e3daea0200104991cb979c2fcc509ae7	4730251e8934a542a009d77dfd027375	delivered	
42901	415cfaaaa8cea49f934470548797fed1	a8dff6357fea30071032ff2091d16430	delivered	
42902	415cfaaaa8cea49f934470548797fed1	a8dff6357fea30071032ff2091d16430	delivered	
49381	101157d4fae1c9fb74a00a5dee265c25	f72b2f8d9295ef93fd40a4c49f67a42b	delivered	
54768	bf49f84a0580ef6751e13357776b7ed9	e7f41abe62db82cffe5c8f6138f18fb2	delivered	
65846	bbfc7badbed2f1828e22b6d629201bd4	f25f442c0ff3a9401eed8ed3a686f362	delivered	
81377	eb855beb3ac99461f7a076b4c3652472	c91289ce43149a8ea5560d446f1d1dd2	delivered	
83792	a7a43f469c0d7bdb0a23a82db125aefa	d7c95dc1ece116c14188092ead3d0951	delivered	
86973	595316a07cd3dea9db7adfcc7e247ae7	696e8f940eeee6b009d1539b59e47366	delivered	
87530	c1424efcde3c9e9febd9e1761667789e	8a80133b8ace6b21415367a131a75a26	delivered	
89079	6f497c40431d5fb0cfbd6c943dd29215	5beb36d1757aa17a044222a7d79b9820	delivered	
106819	a2456e7f02197951664897a94c87242d	7317f41f2cf650174af819cdb68284f0	delivered	

19 rows × 40 columns

In [65]:
```
# doesn't look like there is any data other than these 19 entries, so we can g
o ahead and remove them
no_prod_id1 = olist[olist.product_id=='5eb564652db742ff8f28759cd8d2652a'].inde
x.values
olist = olist.drop(no_prod_id1, axis=0)
```

In [66]:
```
# let's check those were removed
missing_pkg_desc1 = olist[olist['product_weight_g'].isnull()]
missing_pkg_desc1[['product_id']]
```

Out[66]:

	product_id
45964	09ff539a621711667c43eba6a3bd8466

In [67]:
```
# now let's see if we have any information on that package_id
olist[olist.product_id=='09ff539a621711667c43eba6a3bd8466']
```

Out[67]:

	order_id	customer_id	order_status	orde
45964	85f8ad45e067abd694b627859fa57453	1d088dea8732788ec35dd4ee6dd76112	delivered	

1 rows × 40 columns

In [68]:
```
# no information, so we will drop it too
olist=olist.drop([45964],axis=0)
```

In [69]:
```
olist.isna().sum()
```

Out[69]:
```
order_id                            0
customer_id                         0
order_status                        0
order_purchase_timestamp            0
order_approved_at                   0
order_delivered_carrier_date        0
order_delivered_customer_date       0
order_estimated_delivery_date       0
customer_unique_id                  0
customer_zip_code_prefix            0
customer_city                       0
customer_state                      0
review_id                           0
review_score                        0
review_comment_title           101952
review_comment_message          66757
review_creation_date                0
review_answer_timestamp             0
payment_sequential                  0
payment_type                        0
payment_installments                0
payment_value                       0
order_item_id                       0
product_id                          0
seller_id                           0
shipping_limit_date                 0
price                               0
freight_value                       0
product_name_length              1619
product_description_length       1619
product_photos_qty               1619
product_weight_g                    0
product_length_cm                   0
product_height_cm                   0
product_width_cm                    0
product_category_english         1619
seller_zip_code_prefix              0
seller_city                         0
seller_state                        0
tot_order_amt                       0
dtype: int64
```

Note: in practice we probably wouldn't have wasted this much time on 20 values, but it's good exercise with python

Product Information Missing

Let's look at those 1,619 missing values, I do recall there was an NaN in the product category that we had no way to translate to Enlgish - more than likely these are those, but let's take a look

In [70]:
```
missing_prod_info = olist[olist.product_category_english.isna()]
missing_prod_info.product_id.unique()
```

Out[70]:
```
[71225f49be70df4297892f6a5fa62171, 9820e04e332fc450d6fd975befc1bc28, 3bc5164b
c7f4be77002d6651da65c98c, 5a848e4ab52fd5445cdc07aab1c40e48, c600d7f13104e8db2
ca2b9fa78581409, ..., 0fa699aaa0a38aa5d1de8e7fe04d2204, cebad0ed16ecd450b97d2
be843d3da86, f22ccf28078d6ba8961033bc6a20bc12, 45d0bf74166b507caa830564130b5b
a0, e9cbc0910ab050cbd92fbeb051c270ea]
Length: 583
Categories (583, object): [71225f49be70df4297892f6a5fa62171, 9820e04e332fc450
d6fd975befc1bc28, 3bc5164bc7f4be77002d6651da65c98c, 5a848e4ab52fd5445cdc07aab
1c40e48, ..., cebad0ed16ecd450b97d2be843d3da86, f22ccf28078d6ba8961033bc6a20b
c12, 45d0bf74166b507caa830564130b5ba0, e9cbc0910ab050cbd92fbeb051c270ea]
```

In [71]:
```
# capture the unique product id's to see if there are any in the dataset that
 have a match with product information
miss_prod_info_prod_ids = missing_prod_info.product_id.unique()
miss_prod_info_prod_ids
```

Out[71]:
```
[71225f49be70df4297892f6a5fa62171, 9820e04e332fc450d6fd975befc1bc28, 3bc5164b
c7f4be77002d6651da65c98c, 5a848e4ab52fd5445cdc07aab1c40e48, c600d7f13104e8db2
ca2b9fa78581409, ..., 0fa699aaa0a38aa5d1de8e7fe04d2204, cebad0ed16ecd450b97d2
be843d3da86, f22ccf28078d6ba8961033bc6a20bc12, 45d0bf74166b507caa830564130b5b
a0, e9cbc0910ab050cbd92fbeb051c270ea]
Length: 583
Categories (583, object): [71225f49be70df4297892f6a5fa62171, 9820e04e332fc450
d6fd975befc1bc28, 3bc5164bc7f4be77002d6651da65c98c, 5a848e4ab52fd5445cdc07aab
1c40e48, ..., cebad0ed16ecd450b97d2be843d3da86, f22ccf28078d6ba8961033bc6a20b
c12, 45d0bf74166b507caa830564130b5ba0, e9cbc0910ab050cbd92fbeb051c270ea]
```

In [72]:
```
# Let's see how many matches we find in the dataset, if it is more than 1619,
 then we may be able to impute some of the NaNs
count=0
for i in range(len(olist)):
    if olist.iloc[i,23] in miss_prod_info_prod_ids:
        count = count+1

count
```

Out[72]:
```
1619
```

In [73]:
```
# confirmed, there are no additional matches of those product_ids outside of t
hese that are NaN for a category
# so we will go ahead and delete them
no_prod_cat = olist[olist.product_category_english.isna()].index
olist = olist.drop(no_prod_cat, axis=0)
```

In [74]:
```
olist.isna().sum()
```

Out[74]:
```
order_id                              0
customer_id                           0
order_status                          0
order_purchase_timestamp              0
order_approved_at                     0
order_delivered_carrier_date          0
order_delivered_customer_date         0
order_estimated_delivery_date         0
customer_unique_id                    0
customer_zip_code_prefix              0
customer_city                         0
customer_state                        0
review_id                             0
review_score                          0
review_comment_title             100413
review_comment_message            65853
review_creation_date                  0
review_answer_timestamp               0
payment_sequential                    0
payment_type                          0
payment_installments                  0
payment_value                         0
order_item_id                         0
product_id                            0
seller_id                             0
shipping_limit_date                   0
price                                 0
freight_value                         0
product_name_length                   0
product_description_length            0
product_photos_qty                    0
product_weight_g                      0
product_length_cm                     0
product_height_cm                     0
product_width_cm                      0
product_category_english              0
seller_zip_code_prefix                0
seller_city                           0
seller_state                          0
tot_order_amt                         0
dtype: int64
```

------End Missing Data------

In [75]:
```
# remove order_status column
olist=olist.drop(['order_status'], axis=1)
# will verify in next step that is simple stats
```

------Explore Outliers------

Note:

- 0 payment_installments - that means the whole payment was made at the time of purchase
- $0 for payment_value - explore this further
- 0g product_weight - exlore this further - is this an error?

In [76]:
```
# subset the continuous variables
olist_cont = olist.select_dtypes(np.number)
```

In [77]:
```
olist_cont.describe()
```

Out[77]:

	review_score	payment_sequential	payment_installments	payment_value	price
count	114080.000000	114080.000000	114080.000000	114080.000000	114080.000000
mean	4.067234	1.090515	2.946239	172.142134	120.016956
std	1.357761	0.684466	2.781688	266.116465	182.399977
min	1.000000	1.000000	0.000000	0.000000	0.850000
25%	4.000000	1.000000	1.000000	60.950000	39.900000
50%	5.000000	1.000000	2.000000	108.060000	74.900000
75%	5.000000	1.000000	4.000000	189.370000	133.000000
max	5.000000	26.000000	24.000000	13664.080000	6735.000000

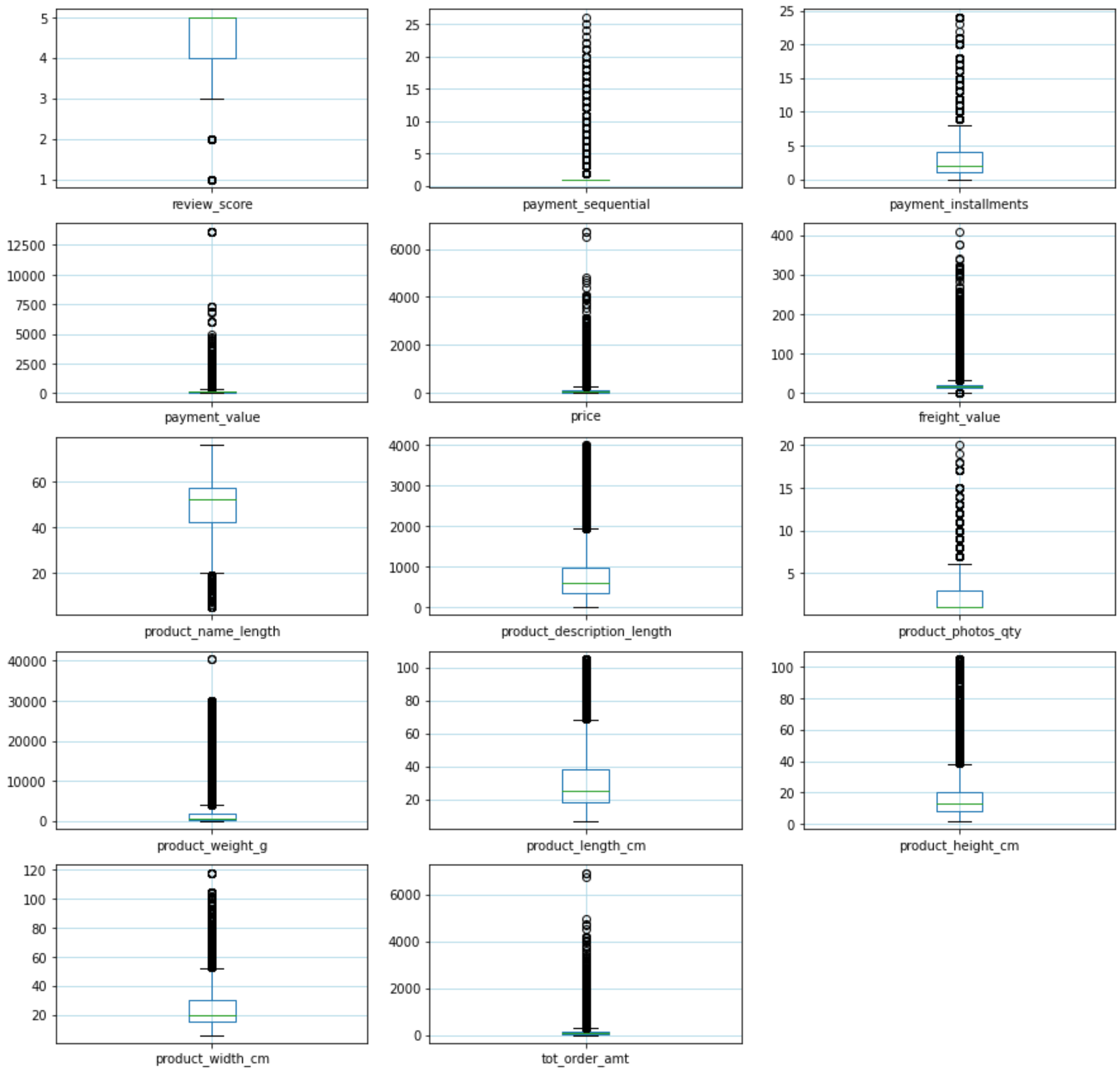
In [78]:
```
olist_cont.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 114080 entries, 0 to 119147
Data columns (total 14 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   review_score                114080 non-null  float64
 1   payment_sequential          114080 non-null  float64
 2   payment_installments        114080 non-null  float64
 3   payment_value               114080 non-null  float64
 4   price                       114080 non-null  float64
 5   freight_value               114080 non-null  float64
 6   product_name_length         114080 non-null  float64
 7   product_description_length  114080 non-null  float64
 8   product_photos_qty          114080 non-null  float64
 9   product_weight_g            114080 non-null  float64
 10  product_length_cm           114080 non-null  float64
 11  product_height_cm           114080 non-null  float64
 12  product_width_cm            114080 non-null  float64
 13  tot_order_amt               114080 non-null  float64
dtypes: float64(14)
memory usage: 13.1 MB
```

In [79]:
```
#create variables
dataseries = pd.Series(range(len(olist_cont.columns)))
dataseries
```

Out[79]:
```
0      0
1      1
2      2
3      3
4      4
5      5
6      6
7      7
8      8
9      9
10    10
11    11
12    12
13    13
dtype: int64
```

In [80]:
```
# boxplot of continuous variables
plt.figure(figsize=(15,15)) #creat plot area

c="red"
for index, plot_vars in enumerate(olist_cont): #plot_vars is a built-in functi
on
    plt.subplot(5,3, index+1) # nrows, ncols, index
    ax=olist_cont.boxplot(column=plot_vars) # fill with color
    ax.set_facecolor('white') #sets background to white
    ax.grid(color="lightblue")


plt.show()
```



Payment_Sequential

In [81]:
```python
olist.groupby(by='payment_sequential').count()
```

Out[81]:

	order_id	customer_id	order_purchase_timestamp	order_approved_at	order_
payment_sequential					
1.0	109205	109205	109205	109205	
2.0	3268	3268	3268	3268	
3.0	632	632	632	632	
4.0	304	304	304	304	
5.0	181	181	181	181	
6.0	124	124	124	124	
7.0	85	85	85	85	
8.0	55	55	55	55	
9.0	44	44	44	44	
10.0	38	38	38	38	
11.0	34	34	34	34	
12.0	26	26	26	26	
13.0	14	14	14	14	
14.0	12	12	12	12	
15.0	10	10	10	10	
16.0	8	8	8	8	
17.0	8	8	8	8	
18.0	8	8	8	8	
19.0	8	8	8	8	
20.0	5	5	5	5	
21.0	5	5	5	5	
22.0	2	2	2	2	
23.0	1	1	1	1	
24.0	1	1	1	1	
25.0	1	1	1	1	
26.0	1	1	1	1	

26 rows × 38 columns

In [82]:
```
olist[olist.payment_sequential>20]
```

Out[82]:

	order_id	customer_id	order_purchase_ti
9152	285c2e15bebd4ac83635ccc563dc71f4	b246eeed30b362c09d867b9e598bee51	2017-12-0
9167	285c2e15bebd4ac83635ccc563dc71f4	b246eeed30b362c09d867b9e598bee51	2017-12-0
85529	895ab968e7bb0d5659d16cd74cd1650c	270c23a11d024a44c896d1894b261a83	2017-08-0
85530	895ab968e7bb0d5659d16cd74cd1650c	270c23a11d024a44c896d1894b261a83	2017-08-0
85531	895ab968e7bb0d5659d16cd74cd1650c	270c23a11d024a44c896d1894b261a83	2017-08-0
92902	ccf804e764ed5650cd8759557269dc13	92cd3ec6e2d643d4ebd0e3d6238f69e2	2017-06-0
92904	ccf804e764ed5650cd8759557269dc13	92cd3ec6e2d643d4ebd0e3d6238f69e2	2017-06-0
92910	ccf804e764ed5650cd8759557269dc13	92cd3ec6e2d643d4ebd0e3d6238f69e2	2017-06-0
92911	ccf804e764ed5650cd8759557269dc13	92cd3ec6e2d643d4ebd0e3d6238f69e2	2017-06-0
92912	ccf804e764ed5650cd8759557269dc13	92cd3ec6e2d643d4ebd0e3d6238f69e2	2017-06-0
92919	ccf804e764ed5650cd8759557269dc13	92cd3ec6e2d643d4ebd0e3d6238f69e2	2017-06-0

11 rows × 39 columns

In [83]:
```
x = olist[olist.order_id=='285c2e15bebd4ac83635ccc563dc71f4']
x[['payment_type','price','freight_value','payment_value','tot_order_amt']]
# yup, looks like a bunch of smaller payments/ vouchers were used
```

Out[83]:

	payment_type	price	freight_value	payment_value	tot_order_amt
9150	voucher	29.0	11.85	1.75	40.85
9151	voucher	29.0	11.85	1.23	40.85
9152	voucher	29.0	11.85	1.05	40.85
9153	voucher	29.0	11.85	1.03	40.85
9154	voucher	29.0	11.85	1.96	40.85
9155	voucher	29.0	11.85	1.14	40.85
9156	voucher	29.0	11.85	1.05	40.85
9157	voucher	29.0	11.85	4.07	40.85
9158	voucher	29.0	11.85	1.70	40.85
9159	voucher	29.0	11.85	1.24	40.85
9160	voucher	29.0	11.85	2.85	40.85
9161	credit_card	29.0	11.85	1.62	40.85
9162	voucher	29.0	11.85	2.24	40.85
9163	voucher	29.0	11.85	1.40	40.85
9164	voucher	29.0	11.85	1.75	40.85
9165	voucher	29.0	11.85	1.00	40.85
9166	voucher	29.0	11.85	2.89	40.85
9167	voucher	29.0	11.85	3.08	40.85
9168	voucher	29.0	11.85	1.23	40.85
9169	voucher	29.0	11.85	1.09	40.85
9170	voucher	29.0	11.85	2.78	40.85
9171	voucher	29.0	11.85	2.70	40.85

Payment Installments

In [84]:
```
olist[olist.payment_installments>10].price.mean()
```

Out[84]: 251.38943529411725

payment_value

In [85]:

```
# let's look at the 0 payment-value
olist[olist.payment_value==0]
#  4 total, paid with voucher - curious - were those just refund/ exchange vou
chers - maybe other vouchers had payments in addition
```

Out[85]:

	order_id	customer_id	order_purchase_ti
504	45ed6e85398a87c253db47c2d9f48216	8eab8f9b3c744b76b65f7a2c0c8f2d6c	2017-06-08
38755	6ccb433e00daae1283ccc956189c82ae	843b211abe7b0264dd4a69eafc5bdf43	2017-10-26
102648	b23878b3e8eb4d25a158f57d96331b18	648121b599d98c420ef93f6135f8c80c	2017-05-27
116144	8bcbe01d44d147f901cd3192671144db	f2def7f64f36952f2f5a9791f0285f34	2018-01-24

4 rows × 39 columns

In [86]:
```
# let's look at orders where there was a voucher and more than one payment typ
e
olist[olist.payment_sequential>1]
```

Out[86]:

	order_id	customer_id	order_purchase_t
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-0
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-0
11	e69bfb5eb88e0ed6a785585b27e16dbf	31ad1d1b63eb9962463f764d4e6e0c9d	2017-07-2
23	83018ec114eee8641c97e08f7b4e926f	7f8c8b9c2ae27bf3300f670c3d478be8	2017-10-2
24	83018ec114eee8641c97e08f7b4e926f	7f8c8b9c2ae27bf3300f670c3d478be8	2017-10-2
...	...	...	
119035	4bafa54db6b060da198f23f810835969	48094f58f03bec9519bd0e004ce460df	2018-04-0
119036	4bafa54db6b060da198f23f810835969	48094f58f03bec9519bd0e004ce460df	2018-04-0
119137	9115830be804184b91f5c00f6f49f92d	da2124f134f5dfbce9d06f29bdb6c308	2017-10-0
119138	9115830be804184b91f5c00f6f49f92d	da2124f134f5dfbce9d06f29bdb6c308	2017-10-0
119139	aa04ef5214580b06b10e2a378300db44	f01a6bfcc730456317e4081fe0c9940e	2017-01-2

4875 rows × 39 columns

In [87]:
```
# this order has 3 payment types listed
olist[olist.order_id=='e481f51cbdc54678b7cc49136f2d6af7'][['order_id','payment
_sequential','payment_type','price','freight_value','payment_value','tot_order
_amt']]
#confirmed - vouchers are not included in the payment_value, so you payment_va
lues of R$0
```

Out[87]:

	order_id	payment_sequential	payment_type	price	freight_value	pa
0	e481f51cbdc54678b7cc49136f2d6af7	1.0	credit_card	29.99	8.72	
1	e481f51cbdc54678b7cc49136f2d6af7	3.0	voucher	29.99	8.72	
2	e481f51cbdc54678b7cc49136f2d6af7	2.0	voucher	29.99	8.72	

In [88]:
```
olist.payment_value.describe()
```

Out[88]:
```
count    114080.000000
mean        172.142134
std         266.116465
min           0.000000
25%          60.950000
50%         108.060000
75%         189.370000
max       13664.080000
Name: payment_value, dtype: float64
```

In [89]:
```
x = olist[olist.payment_value>10000]
x[x.order_id=='03caa2c082116e1d31e67e9ae3700499']
```

Out[89]:

	order_id	customer_id	order_purchase_ti
15916	03caa2c082116e1d31e67e9ae3700499	1617b1357756262bfa56ab541c47bc16	2017-09-29
15917	03caa2c082116e1d31e67e9ae3700499	1617b1357756262bfa56ab541c47bc16	2017-09-29
15918	03caa2c082116e1d31e67e9ae3700499	1617b1357756262bfa56ab541c47bc16	2017-09-29
15919	03caa2c082116e1d31e67e9ae3700499	1617b1357756262bfa56ab541c47bc16	2017-09-29
15920	03caa2c082116e1d31e67e9ae3700499	1617b1357756262bfa56ab541c47bc16	2017-09-29
15921	03caa2c082116e1d31e67e9ae3700499	1617b1357756262bfa56ab541c47bc16	2017-09-29
15922	03caa2c082116e1d31e67e9ae3700499	1617b1357756262bfa56ab541c47bc16	2017-09-29
15923	03caa2c082116e1d31e67e9ae3700499	1617b1357756262bfa56ab541c47bc16	2017-09-29

8 rows × 39 columns

Price

In [90]:
```
olist.price.describe()
```

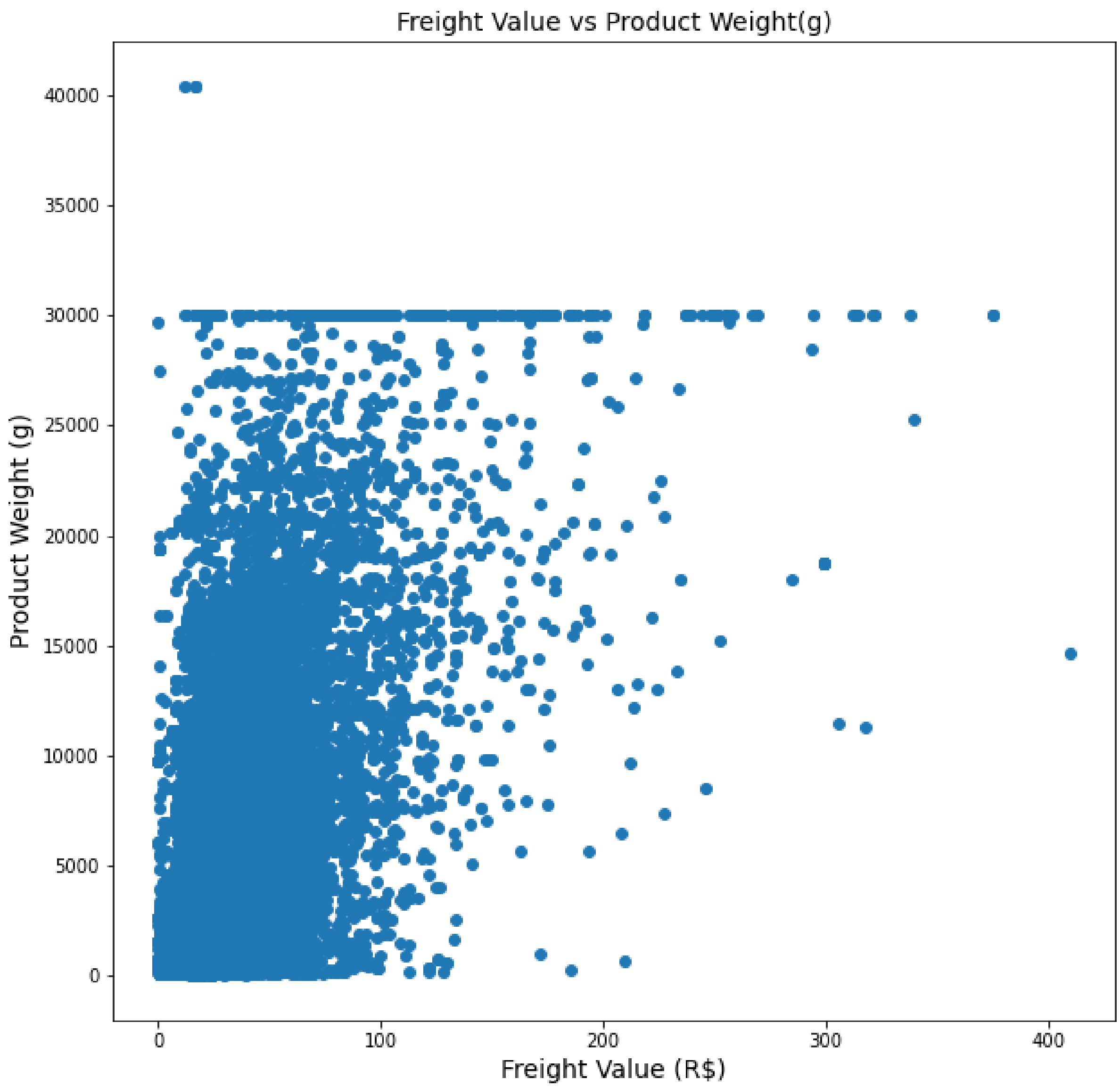
Out[90]:
```
count    114080.000000
mean        120.016956
std         182.399977
min           0.850000
25%          39.900000
50%          74.900000
75%         133.000000
max        6735.000000
Name: price, dtype: float64
```

In [91]:
```
p=olist[olist.price>2000]
p[p.order_id.duplicated(keep=False)].price.mean()
```

Out[91]: 2938.4833333333336

Freight Value

In [92]:
```
# plot frieght value + product weight
fig, ax = plt.subplots(figsize=(10,10))
plt.scatter(olist.freight_value,olist.product_weight_g)
ax.set_facecolor('white') #sets background to white
ax.set_xlabel('Freight Value (R$)', fontsize='14') # x label
ax.set_ylabel('Product Weight (g)', fontsize='14') # y label
ax.set_title('Freight Value vs Product Weight(g)', fontsize='14') # graph name
plt.show()
```



In [93]:
```python
#interesting there appears to be a R$3,000 order cutoff, let's investigate
list(olist[olist.freight_value>300]['product_category_english'].unique())
```

Out[93]:
```
['health_beauty',
 'construction_tools_construction',
 'housewares',
 'baby',
 'industry_commerce_and_business']
```

product weight

no to low weight

In [94]:
```python
# what are the smallest values in the weight column
olist.nsmallest(1000, 'product_weight_g')['product_weight_g'].unique()
```

Out[94]:
```
array([ 0.,  2., 25., 50., 53., 54., 55., 58., 60., 61., 63., 65., 67.,
       70., 75.])
```

In [95]:
```python
# find those records with 0g or 2g of weight (the lowet 5 weights are 0, 2 , 2
5, 50, 53)
weights = range(0,2)
no_weight = olist[olist.product_weight_g.isin([0,2])]
no_weight[['order_id', 'product_id','product_weight_g', 'order_item_id']]

# only 2 where part of multi-item orders, I have to think anything with less t
han 25g in weight is a mistake, let's remove those
```

Out[95]:

	order_id	product_id	product_weight_
6091	06afc1144eb9f51ef2aa90ec9223c7f4	e673e90efa65a5409ff4196c038bb5af	0.
6092	06afc1144eb9f51ef2aa90ec9223c7f4	e673e90efa65a5409ff4196c038bb5af	0.
14227	5ed3cdc2df4c7cb624fc3ac0f66d4a02	8aae4df46baf1278422b69edbb50bd35	2.
15273	4abc7b5330425bcf9c2f7f48151a88c0	8038040ee2a71048d4bdbbdc985b69ab	0.
42971	d1129445a7d52481f5cd8c9198d4957e	5837bba0ce6e35e6f2dc5c3e223e3276	2.
45756	3f50858a49c788f7633204f2fd7bf63f	f9fafac43d3416d92ecc303fdeb1743d	2.
47066	200b121c28e10ef638131a7c76753327	81781c0fed9fe1ad6e8c81fca1e1cb08	0.
59107	a12196b6febf252a10f39f6d76ac0a8d	7ddb76f2c7237acc852358b95e7946a8	2.
78056	476b812a7e4fc972646eb390517bddcb	e673e90efa65a5409ff4196c038bb5af	0.
105326	62e27d060f7d7b3d2fd424b0e1d7adaa	ad7d07f5775feab3f20504d1ad3fff11	2.
109574	b489f7ae130ba3fd26b0a20f8cc81c61	e673e90efa65a5409ff4196c038bb5af	0.
117476	06d9e69034388abf6da64378e10737b8	36ba42dd187055e1fbe943b2d11430ca	0.
117477	06d9e69034388abf6da64378e10737b8	36ba42dd187055e1fbe943b2d11430ca	0.

In [96]:
```
no_weight
```

Out[96]:

	order_id	customer_id	order_purchase_t
6091	06afc1144eb9f51ef2aa90ec9223c7f4	e8be078dee76002545a9c5f10b7d7c4e	2018-08-1
6092	06afc1144eb9f51ef2aa90ec9223c7f4	e8be078dee76002545a9c5f10b7d7c4e	2018-08-1
14227	5ed3cdc2df4c7cb624fc3ac0f66d4a02	4a8ce9a8cb458c67086e351552936cd2	2018-06-0
15273	4abc7b5330425bcf9c2f7f48151a88c0	d1568f1104d2015dc70bdf7d9ab88dd2	2018-07-3
42971	d1129445a7d52481f5cd8c9198d4957e	f215bd697983e112b9b42330cf503c50	2018-06-2
45756	3f50858a49c788f7633204f2fd7bf63f	b5d131af7951f37c72c8439e2eb9bf0b	2018-04-0
47066	200b121c28e10ef638131a7c76753327	26bcca10e5c9679c306d8333bf527929	2018-08-0
59107	a12196b6febf252a10f39f6d76ac0a8d	a2a2b3024be00dfdd15cae8f8c526076	2018-03-1
78056	476b812a7e4fc972646eb390517bddcb	18a1176652a9344ba489fa4ccaa3c20f	2018-08-1
105326	62e27d060f7d7b3d2fd424b0e1d7adaa	ebc37754efcd5ef1954d9980073a6c00	2018-04-0
109574	b489f7ae130ba3fd26b0a20f8cc81c61	99411e9599f8b7a90f2a362b874b66ca	2018-08-1
117476	06d9e69034388abf6da64378e10737b8	afef0047e43944e8c6630ec0d0f7de2e	2018-07-3
117477	06d9e69034388abf6da64378e10737b8	afef0047e43944e8c6630ec0d0f7de2e	2018-07-3

13 rows × 39 columns

In [97]:
```
no_weight.price.mean()
```

Out[97]: 135.6192307692308

In [98]:
```
no_weight.freight_value.mean()
```

Out[98]: 22.92076923076923

In [99]:
```
# get the order_id's so we can just dump them
index = no_weight.index
index
```

Out[99]:
```
Int64Index([  6091,   6092,  14227,  15273,  42971,  45756,  47066,  59107,
             78056, 105326, 109574, 117476, 117477],
           dtype='int64')
```

In [100]:
```
#drop those indexes from the data set
olist=olist.drop(index=index, axis=0)
```

Large Weights

In [101]:
```
# explore those weights around 30kg-40kg
weights = olist[olist.product_weight_g>29000]
```

In [102]:
```
weights[weights.product_weight_g>30000]
```

Out[102]:

	order_id	customer_id	order_purchase_ti
34975	4a45f9f66971302cf881ecfa142f42ba	ccd6a4af78390b7ae560c1cc1cb1a2ff	2017-12-2:
96682	6ecf1a4051b4c5ed613624b460970a26	958279c23050d6207d196c3057648f6f	2017-11-1
108107	9223919b300f6989e1715333fca0d6ce	51934b734e94e61d8efa4523e175c6c3	2018-03-0

3 rows × 39 columns

Product Height

In [103]:
```
# description
olist.product_height_cm.describe()
```

Out[103]:
```
count    114067.000000
mean         16.606214
std          13.439872
min           2.000000
25%           8.000000
50%          13.000000
75%          20.000000
max         105.000000
Name: product_height_cm, dtype: float64
```

In [104]:
```python
# the boxplot shows outliers above 40cm
olist[olist.product_height_cm>40]
```

Out[104]:

	order_id	customer_id	order_purchase_ti
11	e69bfb5eb88e0ed6a785585b27e16dbf	31ad1d1b63eb9962463f764d4e6e0c9d	2017-07-2!
12	e69bfb5eb88e0ed6a785585b27e16dbf	31ad1d1b63eb9962463f764d4e6e0c9d	2017-07-2!
37	f70a0aff17df5a6cdd9a7196128bd354	456dc10730fbdba34615447ea195d643	2017-08-1(
50	d22e9fa5731b9e30e8b27afcdc2f8563	756fb9391752dad934e0fe3733378e57	2018-08-0⁄
105	e4de6d53ecff736bc68804b0b6e9f635	9f6618c17568ac301465fe7ad056c674	2017-10-1(
...	...	...	
119071	53ca14c357e60c77cd57aa96c8f0b4a5	f1cf46100438d0ef4e1916f2aef26718	2018-04-1:
119088	83db27f85506380229913b0dfdf5cd18	472acc24324ad4cee482fe4ef5910dc1	2018-04-1{
119089	83db27f85506380229913b0dfdf5cd18	472acc24324ad4cee482fe4ef5910dc1	2018-04-1{
119109	f6f9344efc918f1e00ab84c014aa21d7	166478efeed4f9a861164b4ff5acfe8b	2017-05-2(
119144	83c1379a015df1e13d02aae0204711ab	1aa71eb042121263aafbe80c1b562c9c	2017-08-2:

6561 rows × 39 columns

In [105]:
```
olist[olist.product_height_cm>=105]
```

Out[105]:

	order_id	customer_id	order_purchase_t
256	412fccb2b44a99b36714bca3fef8ad7b	c6865c523687cb3f235aa599afef1710	2018-07-2
376	c619ac0f9cdcfcebdfb9490451166da4	e5f6020aa61432f0dcb0fcda1c5113d9	2017-05-2
1080	4edca0186e467f58fe99ab648f604ce8	1d06482d82c32c8752f4decbbffe7623	2018-03-1
2060	fa44b98d202360f1246681a3e0405db4	ef95061d7fabce59a19b991b10e9bfc5	2018-02-0
3000	4438691d291f7a436db5665e8d010ac9	294ee4cd4c0812c45ee3a69208051364	2017-07-1
...	...	...	
113673	a9a93c428c6103f2151bb63a1d32a520	99ed295fa7b1f749594ce5f709ef1073	2017-01-1
115019	d7a2c0c1ff66b314f3bf166fb4157fd4	c26acf0451e0f8ec1f5218731b9a51cf	2017-11-2
116207	4e50ee5ebc37a770f257082adfbeff18	de4c50d66aeaedf525016938c1892f51	2017-09-0
117947	7eaf906a67eb432a7736e3affdd3611a	c42fc702f762344fa2c3f3ca4bf7064b	2018-01-2
117948	7eaf906a67eb432a7736e3affdd3611a	c42fc702f762344fa2c3f3ca4bf7064b	2018-01-2

134 rows × 39 columns

product length

In [106]:
```
olist.product_length_cm.describe()
```

Out[106]:
```
count    114067.000000
mean         30.290522
std          16.157939
min           7.000000
25%          18.000000
50%          25.000000
75%          38.000000
max         105.000000
Name: product_length_cm, dtype: float64
```

In [107]:
```
# boxplot shows something going on above 60cm in length
olist[olist.product_length_cm>60]
```

Out[107]:

	order_id	customer_id	order_purchase_
7	a4591c265e18cb1dcee52889e2d8acc3	503740e9ca751ccdda7ba28e9ab8f608	2017-07-(
20	403b97836b0c04a622354cf531062e5f	738b086814c6fcc74b8cc583f8516ee3	2018-01-(
29	95266dbfb7e20354baba07964dac78d5	a166da34890074091a942054b36e4265	2018-01-(
39	989225ba6d0ebd5873335f7e01de2ae7	816f8653d5361cbf94e58c33f2502a5c	2017-12-1
42	8563039e855156e48fccee4d611a3196	5f16605299d698660e0606f7eae2d2f9	2018-02-1
...	...	...	
119001	3c042ee4b8b597c3d265a93a21bbf99f	d71a0d0cf6bbacec526203263382501b	2018-06-2
119052	a542babfe6f6339952a1f699ddc1868b	2ba83772da66cfb788b2130a97f46292	2017-12-(
119085	07fcf4ec8cadbea34c5b508e35e716c0	6ab5adc744d5c894470ce74466a78a27	2018-04-1
119087	c0524fb1b4c905d054adbddaffa2380c	92e8f9754238b9697d9dcbe02c20fc70	2017-11-2
119107	7fd85cb0143de098a4c5ab5a57bfbd91	d32034dfc685b1ae15dd4c78eace868e	2017-05-(

6060 rows × 39 columns

In [108]:
```
olist[olist.product_length_cm==105]
```

Out[108]:

	order_id	customer_id	order_purchase_
85	fa516182d28f96f5f5c651026b0749ee	55e6b290205c84ddd23ddf5eb134efd4	2018-04-
206	c8627adf430caff83161acbeb344d8f5	078d86e7bf40b188f0e5ed6d866c21c4	2018-08-
416	0b0f3c7a9bcb6ad1fccab28f9240da6f	7bb3b0d45b4e1a13cd914e4a135e6bd9	2018-07-
655	91bcade2288597459e1dbce52d6d0ef9	449057269c318304d8a753f64c181b07	2018-01-
941	c788aeedf593229d4ee84e790c2e5483	838aa327b0491d3cb3e9c1c6d9bb382c	2017-03-
...	...	...	
117117	ef27173cc11ca724c7073811a4348fe0	8473cae46f9b0388eda716c12eef4da1	2018-04-
117450	cbbadd3919b803b6cdeaa2e7811bc47b	3ce94b5a344798830d5641dda2b05195	2018-03-
118021	60b37209020e1a5a5857b1ceec4037de	90c5ced703f28718eef7ac75cb35e854	2018-08-
118785	d84dce79fd962faab40bc1d4b084d9d7	a0c58da1742fe623c609a92e21af9d5f	2018-07-
119087	c0524fb1b4c905d054adbddaffa2380c	92e8f9754238b9697d9dcbe02c20fc70	2017-11-

311 rows × 39 columns

Product Width

In [109]:
```
olist.product_width_cm.describe()
```

Out[109]:
```
count    114067.000000
mean         23.103553
std          11.738479
min           6.000000
25%          15.000000
50%          20.000000
75%          30.000000
max         118.000000
Name: product_width_cm, dtype: float64
```

In [110]:
```
olist[olist.product_width_cm>50]
```

Out[110]:

	order_id	customer_id	order_purchase_
7	a4591c265e18cb1dcee52889e2d8acc3	503740e9ca751ccdda7ba28e9ab8f608	2017-07-(
29	95266dbfb7e20354baba07964dac78d5	a166da34890074091a942054b36e4265	2018-01-(
68	2edfd6d1f0b4cd0db4bf37b1b224d855	241e78de29b3090cfa1b5d73a8130c72	2017-06-1
74	688052146432ef8253587b930b01a06d	81e08b08e5ed4472008030d70327c71f	2018-04-2
149	e37797aedc7cd4ef82278fbc169eecaf	4c9c7c2b6de6ee2568681b5599bb7495	2018-05-(
...	...	...	
119049	d2d558357a6c69ca47cc4eca7571f593	0864f6601de6547b99cb81aa5742a119	2017-05-2
119052	a542babfe6f6339952a1f699ddc1868b	2ba83772da66cfb788b2130a97f46292	2017-12-(
119085	07fcf4ec8cadbea34c5b508e35e716c0	6ab5adc744d5c894470ce74466a78a27	2018-04-1
119109	f6f9344efc918f1e00ab84c014aa21d7	166478efeed4f9a861164b4ff5acfe8b	2017-05-2
119127	d692ef54145c9cb3322ec2e5508aa3f4	82ddfcf9438b0cd1117b55ac33184df8	2018-03-2

2893 rows × 39 columns

In [111]:
```
olist[olist.product_width_cm==105]
```

Out[111]:

	order_id	customer_id	order_purchase_
15997	dbb763c04b5beadac2bbd9d8560f8728	e8eca07ef914ed5dc59f011540237b2e	2017-05-:
19044	0680b3722414028009984ae5c04c5df0	a95bf627a6d55bc2194028e5648c0fdd	2017-06-(
35036	4abe45d3258a19bc521cdd64b940acf1	9fad0cbea18c67d95c21db7c76670226	2017-06-:
42329	d40e7a80365572af2e27d31a9cf79169	d01c7d818c6b39a0657b11a9e85f99d2	2017-06-:
47950	1a4ed278e4797230fbc18916eb8e8dae	8dad82945d0eeea58e2e91b451171710	2018-01-
55668	fc1d26c4d0639d4e8f684b618b355146	a6314abf76ef0a93e6821a5e9c925de1	2017-06-(
67824	b6e5ddeaf2e5b56015e295beb333f305	18496a6fa6f46c453fe2e56cec440d0d	2017-06-
69989	d5d5a70a76401ecc139ee16c5299df4d	11529689a3fdf79b53cf853daf570b16	2018-05-:
75278	a97bd5e7cb82fc947d946bc4d3505805	e6a332325992b11bc1f237a65335cb40	2017-06-
83725	2a3a8593aad8c5ddedc827304dfedbab	9175f906b8ad6c804df1837458e0da1c	2017-07-
96387	cdd55639e282b90007cac9a8ea6a7798	6065fad852344e668bc2352e47d3db07	2017-10-:
100095	278b6dc641ac6cf9abcfb8b2aa95a0d0	28e769bf73caf3dc734588072640486c	2017-02-(
116517	826ca7e23c4d14c9beb4791e24239327	b613e8d4d389f24b5c1d80cc176ca3e3	2017-06-

13 rows × 39 columns

Duplicates

In [112]:
```python
# show the number of duplicated records in the dataframe
dups = olist[olist.duplicated()].count()
dups
```

Out[112]:
```
order_id                         0
customer_id                      0
order_purchase_timestamp         0
order_approved_at                0
order_delivered_carrier_date     0
order_delivered_customer_date    0
order_estimated_delivery_date    0
customer_unique_id               0
customer_zip_code_prefix         0
customer_city                    0
customer_state                   0
review_id                        0
review_score                     0
review_comment_title             0
review_comment_message           0
review_creation_date             0
review_answer_timestamp          0
payment_sequential               0
payment_type                     0
payment_installments             0
payment_value                    0
order_item_id                    0
product_id                       0
seller_id                        0
shipping_limit_date              0
price                            0
freight_value                    0
product_name_length              0
product_description_length       0
product_photos_qty               0
product_weight_g                 0
product_length_cm                0
product_height_cm                0
product_width_cm                 0
product_category_english         0
seller_zip_code_prefix           0
seller_city                      0
seller_state                     0
tot_order_amt                    0
dtype: int64
```

------Simple Stats------

In [113]:
```
olist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 114067 entries, 0 to 119147
Data columns (total 39 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_id                       114067 non-null  category
 1   customer_id                    114067 non-null  category
 2   order_purchase_timestamp       114067 non-null  datetime64[ns]
 3   order_approved_at              114067 non-null  datetime64[ns]
 4   order_delivered_carrier_date   114067 non-null  datetime64[ns]
 5   order_delivered_customer_date  114067 non-null  datetime64[ns]
 6   order_estimated_delivery_date  114067 non-null  datetime64[ns]
 7   customer_unique_id             114067 non-null  category
 8   customer_zip_code_prefix       114067 non-null  category
 9   customer_city                  114067 non-null  category
 10  customer_state                 114067 non-null  category
 11  review_id                      114067 non-null  category
 12  review_score                   114067 non-null  float64
 13  review_comment_title           13661 non-null   category
 14  review_comment_message         48220 non-null   category
 15  review_creation_date           114067 non-null  datetime64[ns]
 16  review_answer_timestamp        114067 non-null  datetime64[ns]
 17  payment_sequential             114067 non-null  float64
 18  payment_type                   114067 non-null  category
 19  payment_installments           114067 non-null  float64
 20  payment_value                  114067 non-null  float64
 21  order_item_id                  114067 non-null  category
 22  product_id                     114067 non-null  category
 23  seller_id                      114067 non-null  category
 24  shipping_limit_date            114067 non-null  datetime64[ns]
 25  price                          114067 non-null  float64
 26  freight_value                  114067 non-null  float64
 27  product_name_length            114067 non-null  float64
 28  product_description_length     114067 non-null  float64
 29  product_photos_qty             114067 non-null  float64
 30  product_weight_g               114067 non-null  float64
 31  product_length_cm              114067 non-null  float64
 32  product_height_cm              114067 non-null  float64
 33  product_width_cm               114067 non-null  float64
 34  product_category_english       114067 non-null  category
 35  seller_zip_code_prefix         114067 non-null  category
 36  seller_city                    114067 non-null  category
 37  seller_state                   114067 non-null  category
 38  tot_order_amt                  114067 non-null  float64
dtypes: category(17), datetime64[ns](8), float64(14)
memory usage: 41.9 MB
```

In [114]:

```
# describe dataframe
pd.options.display.max_columns = olist.shape[1] # this tells python to output
 all columns and not just the first 13
olist.describe(include="all").to_csv('Olist_Variable_Descriptions.csv')
print(olist.describe(include="all")) #tells python to include categorical vari
ables + continuous variables
```

```
                                order_id                       customer_id  \
count                             114067                            114067
unique                             95124                             95124
top     895ab968e7bb0d5659d16cd74cd1650c  270c23a11d024a44c896d1894b261a83
freq                                  63                                63
first                                NaN                               NaN
last                                 NaN                               NaN
mean                                 NaN                               NaN
std                                  NaN                               NaN
min                                  NaN                               NaN
25%                                  NaN                               NaN
50%                                  NaN                               NaN
75%                                  NaN                               NaN
max                                  NaN                               NaN

       order_purchase_timestamp    order_approved_at  \
count                    114067               114067
unique                    94620                87125
top         2017-08-08 20:26:31  2017-08-08 20:43:31
freq                         63                   63
first       2016-10-03 09:44:50  2016-10-04 09:43:32
last        2018-08-29 15:00:37  2018-08-29 15:10:26
mean                        NaN                  NaN
std                         NaN                  NaN
min                         NaN                  NaN
25%                         NaN                  NaN
50%                         NaN                  NaN
75%                         NaN                  NaN
max                         NaN                  NaN

       order_delivered_carrier_date order_delivered_customer_date  \
count                        114067                        114067
unique                        78919                         94336
top             2017-08-10 11:58:14           2017-08-14 12:46:18
freq                             63                            63
first           2016-10-08 10:34:01           2016-10-11 13:46:32
last            2018-09-11 19:48:28           2018-10-17 13:22:46
mean                            NaN                           NaN
std                             NaN                           NaN
min                             NaN                           NaN
25%                             NaN                           NaN
50%                             NaN                           NaN
75%                             NaN                           NaN
max                             NaN                           NaN

       order_estimated_delivery_date                customer_unique_id  \
count                         114067                            114067
unique                           444                             92076
top              2017-12-20 00:00:00  9a736b248f67d166d2fbb006bcb877c3
freq                             644                                75
first            2016-10-27 00:00:00                               NaN
last             2018-10-25 00:00:00                               NaN
mean                             NaN                               NaN
std                              NaN                               NaN
min                              NaN                               NaN
25%                              NaN                               NaN
50%                              NaN                               NaN
```

```
75%                                NaN                                NaN
max                                NaN                                NaN

        customer_zip_code_prefix customer_city customer_state  \
count                   114067.0        114067         114067
unique                   14844.0          4073             27
top                      24220.0     sao paulo             SP
freq                       152.0         18001          48121
first                        NaN           NaN            NaN
last                         NaN           NaN            NaN
mean                         NaN           NaN            NaN
std                          NaN           NaN            NaN
min                          NaN           NaN            NaN
25%                          NaN           NaN            NaN
50%                          NaN           NaN            NaN
75%                          NaN           NaN            NaN
max                          NaN           NaN            NaN

                               review_id   review_score review_comment_title
\
count                             114067  114067.000000                13661
unique                             94943            NaN                 4408
top     eef5dbca8d37dfce6db7d7b16dd0525e            NaN            Recomendo
freq                                  63            NaN                  492
first                                NaN            NaN                  NaN
last                                 NaN            NaN                  NaN
mean                                 NaN       4.067232                  NaN
std                                  NaN       1.357803                  NaN
min                                  NaN       1.000000                  NaN
25%                                  NaN       4.000000                  NaN
50%                                  NaN       5.000000                  NaN
75%                                  NaN       5.000000                  NaN
max                                  NaN       5.000000                  NaN

       review_comment_message review_creation_date review_answer_timestamp  \
count                   48220               114067                  114067
unique                  34585                  627                   94796
top                 Muito bom  2017-12-19 00:00:00     2017-08-17 22:17:55
freq                      253                  526                      63
first                     NaN  2016-10-15 00:00:00     2016-10-16 03:20:17
last                      NaN  2018-08-31 00:00:00     2018-10-29 12:27:35
mean                      NaN                  NaN                     NaN
std                       NaN                  NaN                     NaN
min                       NaN                  NaN                     NaN
25%                       NaN                  NaN                     NaN
50%                       NaN                  NaN                     NaN
75%                       NaN                  NaN                     NaN
max                       NaN                  NaN                     NaN

        payment_sequential payment_type  payment_installments  payment_value
\
count        114067.000000       114067         114067.000000  114067.000000
unique                 NaN            4                   NaN            NaN
top                    NaN  credit_card                   NaN            NaN
freq                   NaN        84163                   NaN            NaN
first                  NaN          NaN                   NaN            NaN
last                   NaN          NaN                   NaN            NaN
```

```
mean           1.090526         NaN              2.946032     172.138819
std            0.684505         NaN              2.781627     266.130118
min            1.000000         NaN              0.000000       0.000000
25%            1.000000         NaN              1.000000      60.950000
50%            1.000000         NaN              2.000000     108.040000
75%            1.000000         NaN              4.000000     189.370000
max           26.000000         NaN             24.000000   13664.080000

       order_item_id                        product_id  \
count       114067.0                            114067
unique          21.0                             31619
top              1.0  aca2eb7d00ea1a7b8ebd4e68314663af
freq         99862.0                               529
first            NaN                               NaN
last             NaN                               NaN
mean             NaN                               NaN
std              NaN                               NaN
min              NaN                               NaN
25%              NaN                               NaN
50%              NaN                               NaN
75%              NaN                               NaN
max              NaN                               NaN

                               seller_id  shipping_limit_date          price
\
count                             114067               114067  114067.000000
unique                              2914                90126            NaN
top     4a3ca9315b744ce9f8e9374361493884  2017-08-14 20:43:31            NaN
freq                                2116                   63            NaN
first                                NaN  2016-10-08 10:34:01            NaN
last                                 NaN  2020-04-09 22:35:08            NaN
mean                                 NaN                  NaN     120.015177
std                                  NaN                  NaN     182.409119
min                                  NaN                  NaN       0.850000
25%                                  NaN                  NaN      39.900000
50%                                  NaN                  NaN      74.900000
75%                                  NaN                  NaN     133.000000
max                                  NaN                  NaN    6735.000000

       freight_value  product_name_length  product_description_length  \
count  114067.000000        114067.000000               114067.000000
unique           NaN                  NaN                         NaN
top              NaN                  NaN                         NaN
freq             NaN                  NaN                         NaN
first            NaN                  NaN                         NaN
last             NaN                  NaN                         NaN
mean       20.009924            48.803011                  784.819904
std        15.726747            10.016580                  650.583210
min         0.000000             5.000000                    4.000000
25%        13.080000            42.000000                  345.000000
50%        16.320000            52.000000                  600.000000
75%        21.190000            57.000000                  983.000000
max       409.680000            76.000000                 3992.000000

       product_photos_qty  product_weight_g  product_length_cm  \
count       114067.000000     114067.000000      114067.000000
unique                NaN               NaN                NaN
```

```
top                NaN           NaN             NaN
freq               NaN           NaN             NaN
first              NaN           NaN             NaN
last               NaN           NaN             NaN
mean          2.206484   2108.955412       30.290522
std           1.718008   3768.828827       16.157939
min           1.000000     25.000000        7.000000
25%           1.000000    300.000000       18.000000
50%           1.000000    700.000000       25.000000
75%           3.000000   1800.000000       38.000000
max          20.000000  40425.000000      105.000000

        product_height_cm  product_width_cm product_category_english  \
count       114067.000000     114067.000000                   114067
unique                NaN               NaN                       73
top                   NaN               NaN           bed_bath_table
freq                  NaN               NaN                    11808
first                 NaN               NaN                      NaN
last                  NaN               NaN                      NaN
mean            16.606214         23.103553                      NaN
std             13.439872         11.738479                      NaN
min              2.000000          6.000000                      NaN
25%              8.000000         15.000000                      NaN
50%             13.000000         20.000000                      NaN
75%             20.000000         30.000000                      NaN
max            105.000000        118.000000                      NaN

        seller_zip_code_prefix seller_city seller_state  tot_order_amt
count                 114067.0      114067       114067  114067.000000
unique                  2136.0         588           22            NaN
top                    14940.0   sao paulo           SP            NaN
freq                    8188.0       28431        81370            NaN
first                      NaN         NaN          NaN            NaN
last                       NaN         NaN          NaN            NaN
mean                       NaN         NaN          NaN     140.025101
std                        NaN         NaN          NaN     189.478685
min                        NaN         NaN          NaN       6.080000
25%                        NaN         NaN          NaN      55.270000
50%                        NaN         NaN          NaN      91.790000
75%                        NaN         NaN          NaN     157.300000
max                        NaN         NaN          NaN    6929.310000
```

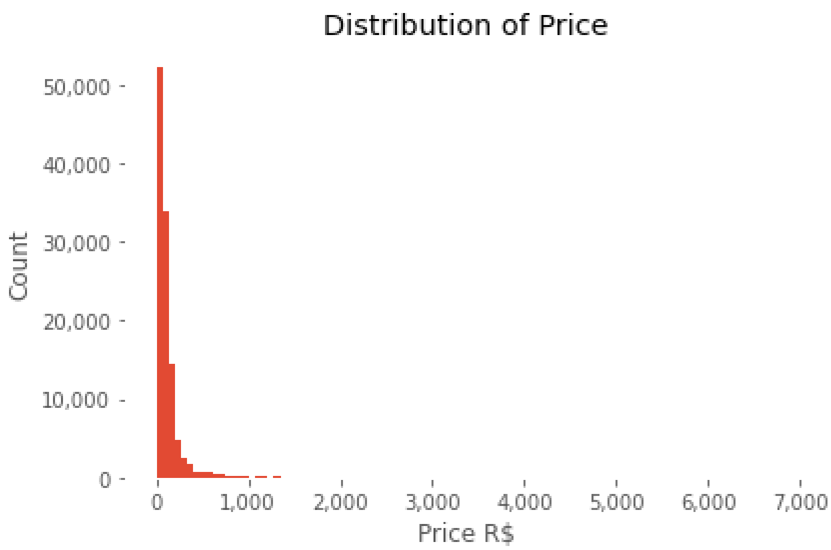
In [115]:
```
olist.payment_type.unique()
```

Out[115]:
```
[credit_card, voucher, boleto, debit_card]
Categories (4, object): [credit_card, voucher, boleto, debit_card]
```

Vizualize Attributes

In [116]:
```
# start withs some basic plotting
# this python magics will allow plot to be embedded into the notebook
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter('ignore', DeprecationWarning)
%matplotlib inline
```

In [117]:
```
#Simple histogram plot of price distribution
plt.style.use('ggplot')
fig, ax = plt.subplots(1, 1, sharey=True, tight_layout=True)
x= olist.price

ax.hist(x, bins=100)
ax.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x:,.0f}'))
ax.xaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x:,.0f}'))
ax.set_facecolor('white')

plt.title('Distribution of Price') #labels
plt.xlabel('Price R$')
plt.ylabel('Count')

plt.show()
```

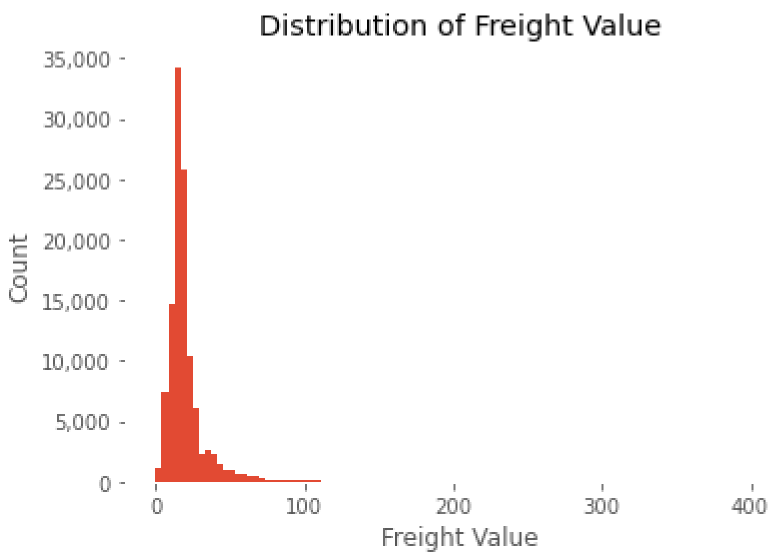


In [118]:
```
reviews = pd.DataFrame(olist.groupby(by='review_score')['review_score'].count
())
reviews.review_score
```

Out[118]:
```
review_score
1.0    13348
2.0     3947
3.0     9642
4.0    21881
5.0    65249
Name: review_score, dtype: int64
```

In [119]:
```
#Simple plot of review distribution
fig, ax = plt.subplots()
x= reviews.index
height= reviews.review_score

plt.rcParams['axes.facecolor'] = 'white'
ax.bar(x = x, height = height,align='center') #Specifying bin edges to make 5
 categories.

ax.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x:,.0f}'))
plt.title('Distribution of Review Scores from 1 to 5') #labels
plt.xlabel('Score')
plt.ylabel('Count')


plt.show()
```

In [120]:
```
#Simple histogram plot of freight value distribution


fig, ax = plt.subplots()
x= olist.freight_value

ax.hist(x, bins=100)
ax.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x:,.0f}'))
ax.xaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x:,.0f}'))
ax.set_facecolor('white')
plt.title('Distribution of Freight Value') #labels
plt.xlabel('Freight Value')
plt.ylabel('Count')

plt.show()
```



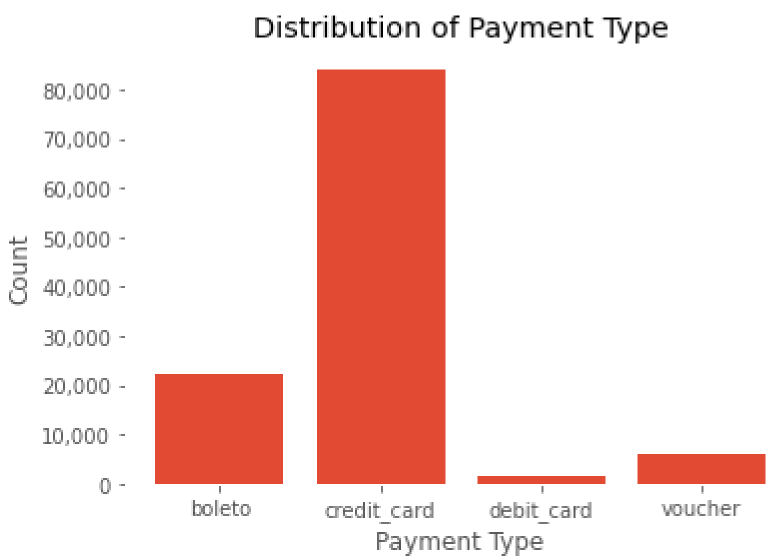
In [121]:
```
payment = pd.DataFrame(olist.groupby(by='payment_type')['payment_type'].count
())
payment = payment.rename({'payment_type' : 'count'})
payment['payment'] = payment.index
payment = payment.drop(payment.payment[3])
payment
```

Out[121]:

	payment_type	payment
payment_type		
boleto	22203	boleto
credit_card	84163	credit_card
debit_card	1633	debit_card
voucher	6068	voucher

In [122]:

```
# distribution of Payment Type

fig, ax = plt.subplots()
x= payment.payment
height = payment.payment_type

ax.bar(x, height=height, align='center')
ax.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x:,.0f}'))
ax.set_facecolor('white')
ax.set_xticklabels(x)

plt.title('Distribution of Payment Type')
plt.xlabel('Payment Type')
plt.ylabel('Count')
plt.show()
```



In [123]:

```
# product categories
prodcat = pd.DataFrame(olist.groupby('product_category_english')['product_cate
gory_english'].count())
prodcat['category'] = prodcat.index
prodcat['count']=prodcat.iloc[:,0]
prodcat = prodcat.drop(['product_category_english'], axis=1)
prodcat
```

Out[123]:

	category	count
product_category_english		
agro_industry_and_commerce	agro_industry_and_commerce	246
air_conditioning	air_conditioning	294
art	art	207
arts_and_craftmanship	arts_and_craftmanship	24
audio	audio	379
...	...	...
stationery	stationery	2571
tablets_printing_image	tablets_printing_image	87
telephony	telephony	4607
toys	toys	4192
watches_gifts	watches_gifts	6075

73 rows × 2 columns

In [124]:
```
pd.set_option("display.max_rows", None)
prodcat.sort_values(by='count')
```

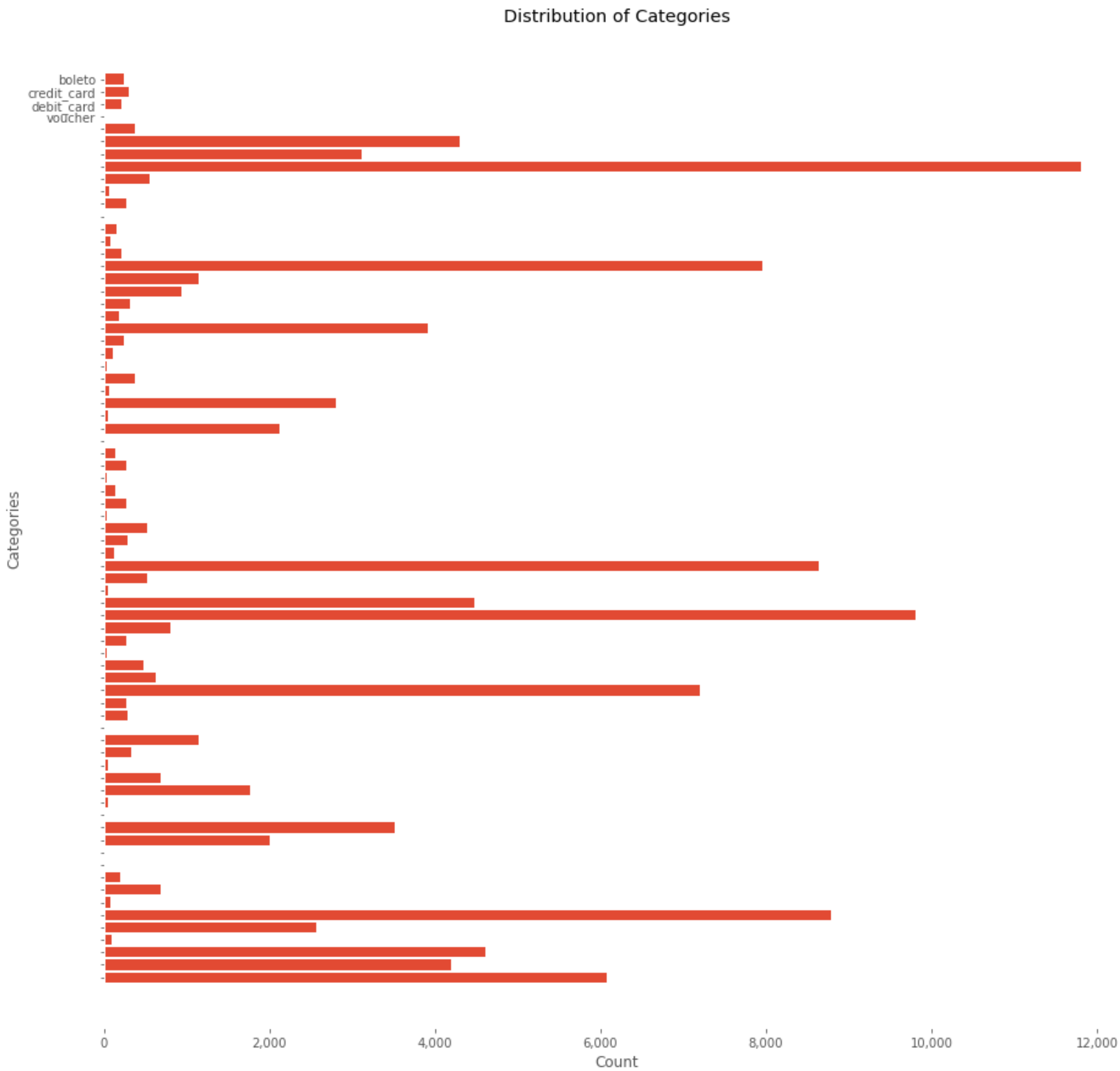
Out[124]:

	category	count
product_category_english		
security_and_services	security_and_services	2
fashion_childrens_clothes	fashion_childrens_clothes	7
pc_gamer	pc_gamer	9
cds_dvds_musicals	cds_dvds_musicals	14
portable_kitchen_and_food_preparers	portable_kitchen_and_food_preparers	14
la_cuisine	la_cuisine	16
arts_and_craftmanship	arts_and_craftmanship	24
fashion_sport	fashion_sport	30
home_comfort_2	home_comfort_2	31
flowers	flowers	33
diapers_and_hygiene	diapers_and_hygiene	37
furniture_mattress_and_upholstery	furniture_mattress_and_upholstery	40
music	music	40
party_supplies	party_supplies	45
fashio_female_clothing	fashio_female_clothing	46
books_imported	books_imported	59
dvds_blu_ray	dvds_blu_ray	67
cine_photo	cine_photo	72
small_appliances_home_oven_and_coffee	small_appliances_home_oven_and_coffee	75
tablets_printing_image	tablets_printing_image	87
costruction_tools_tools	costruction_tools_tools	105
furniture_bedroom	furniture_bedroom	119
fashion_male_clothing	fashion_male_clothing	138
fashion_underwear_beach	fashion_underwear_beach	140
christmas_supplies	christmas_supplies	152
construction_tools_safety	construction_tools_safety	187
signaling_and_security	signaling_and_security	199
art	art	207
computers	computers	216
costruction_tools_garden	costruction_tools_garden	241
agro_industry_and_commerce	agro_industry_and_commerce	246
fixed_telephony	fixed_telephony	262
home_appliances_2	home_appliances_2	264
books_technical	books_technical	268

	category	count
product_category_english		
industry_commerce_and_business	industry_commerce_and_business	268
fashion_shoes	fashion_shoes	273
food_drink	food_drink	281
kitchen_dining_laundry_garden_furniture	kitchen_dining_laundry_garden_furniture	291
air_conditioning	air_conditioning	294
construction_tools_lights	construction_tools_lights	311
market_place	market_place	325
drinks	drinks	371
audio	audio	379
home_confort	home_confort	473
food	food	515
furniture_living_room	furniture_living_room	524
books_general_interest	books_general_interest	548
home_construction	home_construction	627
small_appliances	small_appliances	684
musical_instruments	musical_instruments	689
home_appliances	home_appliances	809
construction_tools_construction	construction_tools_construction	942
consoles_games	consoles_games	1140
luggage_accessories	luggage_accessories	1148
office_furniture	office_furniture	1763
pet_shop	pet_shop	2007
fashion_bags_accessories	fashion_bags_accessories	2124
stationery	stationery	2571
electronics	electronics	2809
baby	baby	3119
perfumery	perfumery	3506
cool_stuff	cool_stuff	3918
toys	toys	4192
auto	auto	4301
garden_tools	garden_tools	4480
telephony	telephony	4607
watches_gifts	watches_gifts	6075
housewares	housewares	7196
computers_accessories	computers_accessories	7963

	category	count
product_category_english		
furniture_decor	furniture_decor	8639
sports_leisure	sports_leisure	8791
health_beauty	health_beauty	9814
bed_bath_table	bed_bath_table	11808

In [125]:
```
# Chart distribution of categories
fig, ax = plt.subplots(figsize=(15,15))
y = prodcat.category
y_pos = np.flip(np.arange(len(y)))
width = prodcat['count']

ax.barh(y_pos, width=width, align='center')
ax.xaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x:,.0f}'))
ax.set_facecolor('white')
ax.set_yticks(y_pos)
ax.set_yticklabels(x)

plt.title('Distribution of Categories')
plt.ylabel('Categories')
plt.xlabel('Count')
plt.show()
```



Explore Joint Attributes

In [126]:
```python
#date features: day of week and month of order_purchase using purchase timesta
mp. Further converted to categorical data type.
olist['purchase_wk_day'] = olist.copy()['order_purchase_timestamp'].dt.day_nam
e().astype('category')

olist['purchase_wk_day'].unique()
```

Out[126]:
```
[Monday, Tuesday, Wednesday, Saturday, Sunday, Thursday, Friday]
Categories (7, object): [Monday, Tuesday, Wednesday, Saturday, Sunday, Thursd
ay, Friday]
```

In [127]:
```python
# create a column for purchase month in regular english
olist['purchase_month'] = olist.copy()['order_purchase_timestamp'].dt.month_na
me().astype('category')

olist['purchase_month'].unique()
```

Out[127]:
```
[October, July, August, November, February, ..., June, March, December, Septe
mber, April]
Length: 12
Categories (12, object): [October, July, August, November, ..., March, Decemb
er, September, April]
```

In [128]:
```python
# number of records per month
olist['purchase_month'].value_counts()
```

Out[128]:
```
August       12348
May          12270
July         11868
March        11280
June         10965
April        10684
February      9660
January       9183
November      8761
December      6328
October       5761
September     4959
Name: purchase_month, dtype: int64
```

In [129]:
```
# subset months and price (month spent on items, not including shipping)

olist_sub = olist[['price','purchase_month']]

month_price = olist_sub.groupby(by='purchase_month').sum()
month_price = month_price.sort_values(['price'])
month_price
```

Out[129]:

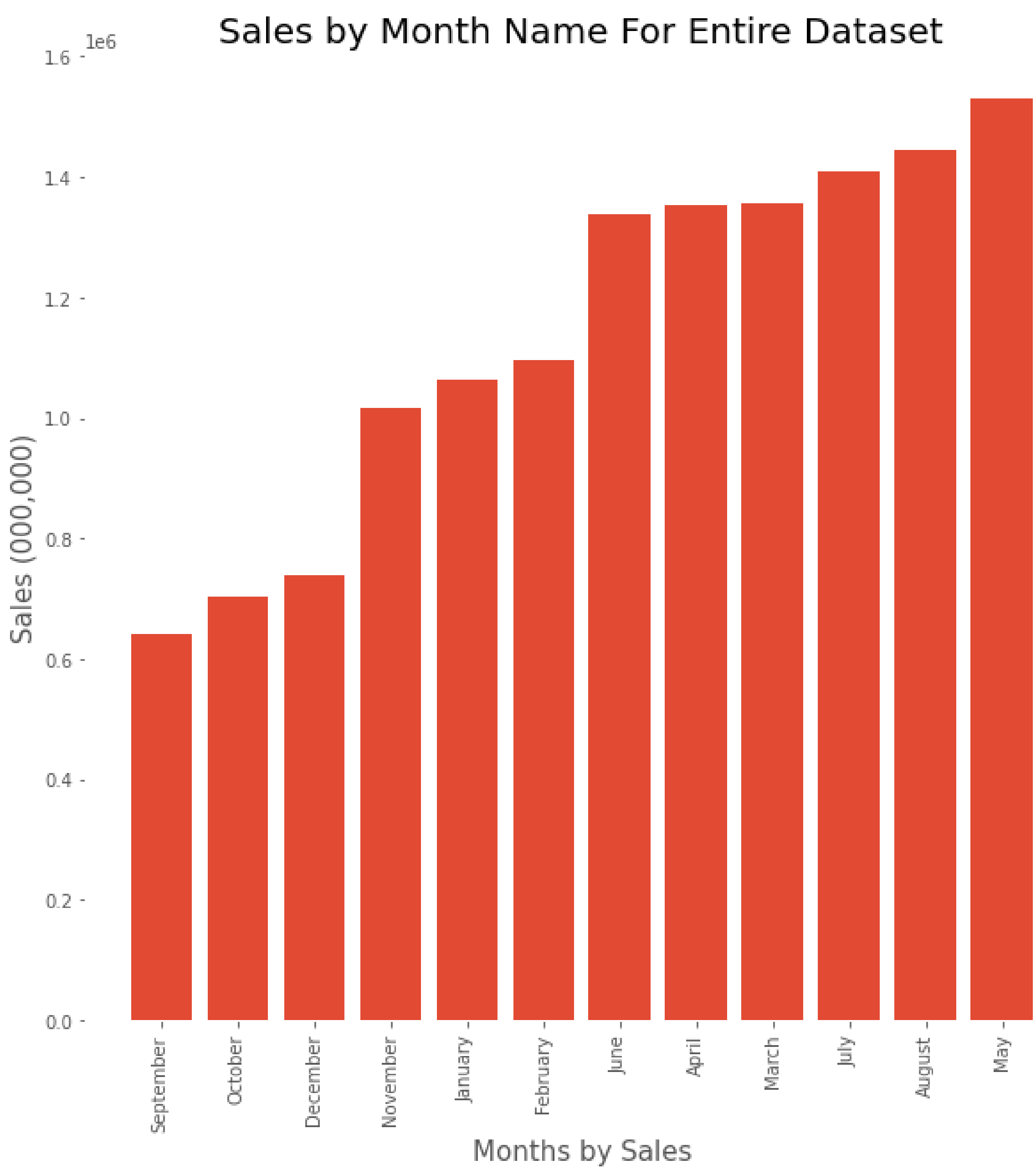
	price
purchase_month	
September	640081.48
October	704399.57
December	739558.09
November	1017007.79
January	1062612.50
February	1095760.81
June	1338495.70
April	1352199.99
March	1357340.47
July	1408669.08
August	1444355.23
May	1529290.53

In [130]:
```
month_price.index.values
```

Out[130]:
```
[September, October, December, November, January, ..., April, March, July, Au
gust, May]
Length: 12
Categories (12, object): [April, August, December, February, ..., May, Novemb
er, October, September]
```

In [131]:

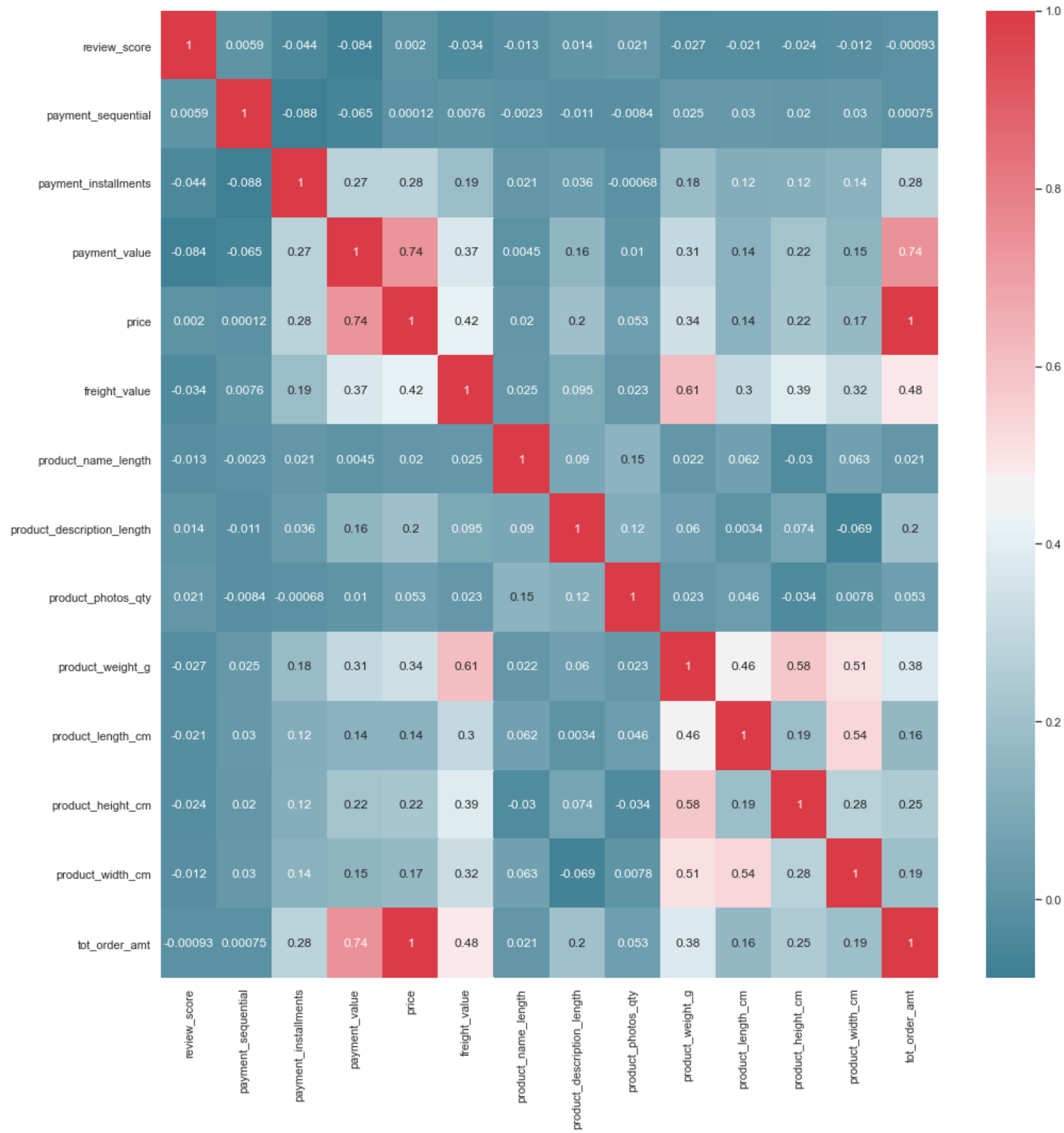
```
# plot some of these continuous varialbes
plt.style.use('ggplot')

fig, ax = plt.subplots(figsize=(10,10), facecolor='w') # create figure on an a
xes, determine size
ax.bar(month_price.index.values, month_price.price, label="linear") # plot the
data month_price created above in barchart
plt.xticks(month_price.index.values, month_price.index.values, rotation='verti
cal') # makes xticks vericle vs horizontal
ax.set_facecolor('white') #sets background to white
ax.set_xlabel('Months by Sales', fontsize='15') # x label
ax.set_ylabel('Sales (000,000)', fontsize='15') # y label
ax.set_title('Sales by Month Name For Entire Dataset', fontsize='20') # graph
 name
plt.show()
```



Explore Joint Attributes

Graphs

In [132]:
```
# let's try a heatmap
# plot the correlation matrix using seaborn
# sns.corrplot() was depricated with v0.6!!!
sns.set(style="darkgrid") # one of the many styles to plot using

cmap = sns.diverging_palette(220, 10, as_cmap=True) # one of the many color ma
ppings

f, ax = plt.subplots(figsize=(15,15))

sns.heatmap(olist.corr(), cmap=cmap, annot=True)

f.tight_layout()
```



Looks like price correlates with payment_value Product_weight_g with freight_value product_height with weight product_width with weight product_length with width

Business Objective EDA

In [133]:
```
olist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 114067 entries, 0 to 119147
Data columns (total 41 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   order_id                        114067 non-null  category
 1   customer_id                     114067 non-null  category
 2   order_purchase_timestamp        114067 non-null  datetime64[ns]
 3   order_approved_at               114067 non-null  datetime64[ns]
 4   order_delivered_carrier_date    114067 non-null  datetime64[ns]
 5   order_delivered_customer_date   114067 non-null  datetime64[ns]
 6   order_estimated_delivery_date   114067 non-null  datetime64[ns]
 7   customer_unique_id              114067 non-null  category
 8   customer_zip_code_prefix        114067 non-null  category
 9   customer_city                   114067 non-null  category
 10  customer_state                  114067 non-null  category
 11  review_id                       114067 non-null  category
 12  review_score                    114067 non-null  float64
 13  review_comment_title            13661 non-null   category
 14  review_comment_message          48220 non-null   category
 15  review_creation_date            114067 non-null  datetime64[ns]
 16  review_answer_timestamp         114067 non-null  datetime64[ns]
 17  payment_sequential              114067 non-null  float64
 18  payment_type                    114067 non-null  category
 19  payment_installments            114067 non-null  float64
 20  payment_value                   114067 non-null  float64
 21  order_item_id                   114067 non-null  category
 22  product_id                      114067 non-null  category
 23  seller_id                       114067 non-null  category
 24  shipping_limit_date             114067 non-null  datetime64[ns]
 25  price                           114067 non-null  float64
 26  freight_value                   114067 non-null  float64
 27  product_name_length             114067 non-null  float64
 28  product_description_length      114067 non-null  float64
 29  product_photos_qty              114067 non-null  float64
 30  product_weight_g                114067 non-null  float64
 31  product_length_cm               114067 non-null  float64
 32  product_height_cm               114067 non-null  float64
 33  product_width_cm                114067 non-null  float64
 34  product_category_english        114067 non-null  category
 35  seller_zip_code_prefix          114067 non-null  category
 36  seller_city                     114067 non-null  category
 37  seller_state                    114067 non-null  category
 38  tot_order_amt                   114067 non-null  float64
 39  purchase_wk_day                 114067 non-null  category
 40  purchase_month                  114067 non-null  category
dtypes: category(19), datetime64[ns](8), float64(14)
memory usage: 42.1 MB
```

Compare estimated delivery with actual delivery duration

select important attributes

- order_purchase_timestamp = purchase initiation timestamp
- order_approved_at = payment approval timestamp
- order_delivered_customer_date = actual order delivery date to the customer
- order_estimated_delivery_date = estimated delivery date provided to the customer at the time of purchase initiation

Objective

- A more accurate estimated delivery date helps the customer to make informed decision.
- Proposal: Estimated delivery date that is +/- 3 days of actual delivery date should be considered a great delivery estimate.

In [134]:
```
#Obtain delivery duration for both actual and estimated
olist["order_delivery_actual_duration"] = olist["order_delivered_customer_dat
e"]-olist["order_approved_at"]
```

In [135]:
```
olist["order_delivery_estimated_duration"] = olist["order_estimated_delivery_d
ate"]-olist["order_approved_at"]
```

In [136]:
```
#Round off the output to days.
olist["order_delivery_actual_duration"] = olist.copy()["order_delivery_actual_
duration"].dt.days
```

In [137]:
```
#Round off the output to days.
olist["order_delivery_estimated_duration"] = olist.copy()["order_delivery_esti
mated_duration"].dt.days
```

In [138]:
```
#How far apart at the estimated and actual delivery duration
olist["delivery_estimate_discrepancy"] = olist["order_delivery_estimated_durat
ion"] - olist["order_delivery_actual_duration"]
```

In [139]:
```python
olist.head()
```

Out[139]:

	order_id	customer_id	order_purchase_timest
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:5
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:5
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:5
3	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	2018-07-24 20:4
4	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	2018-08-08 08:3

5 rows × 44 columns

In [140]:
```python
olist['product_dimensions'] = olist['product_length_cm'] * olist['product_height_cm'] * olist['product_width_cm']
olist.head()
```

Out[140]:

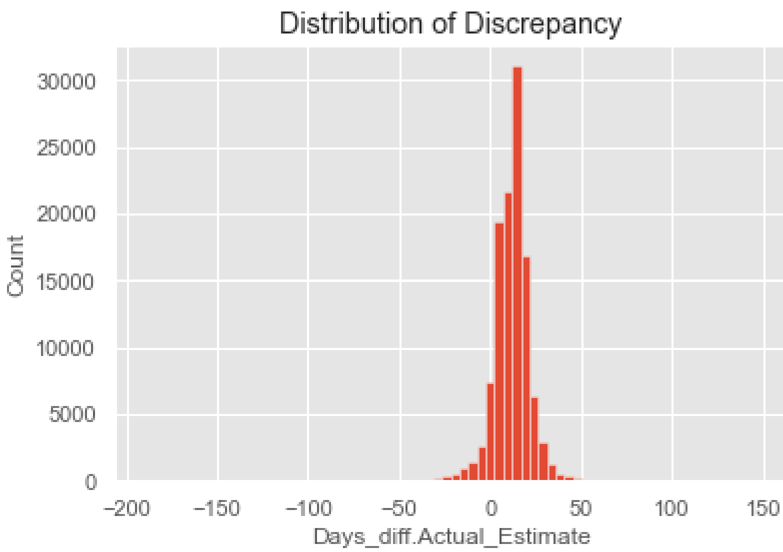
	order_id	customer_id	order_purchase_timest
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:5
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:5
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	2017-10-02 10:5
3	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	2018-07-24 20:4
4	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	2018-08-08 08:3

5 rows × 45 columns

In [141]:
```python
olist['delivery_estimate_discrepancy'].describe()
```

Out[141]:
```
count    114067.000000
mean         11.399485
std          10.168780
min        -189.000000
25%           7.000000
50%          12.000000
75%          16.000000
max         146.000000
Name: delivery_estimate_discrepancy, dtype: float64
```

In [142]:
```
#Distribution of the discrepancy

plt.style.use('ggplot')

plt.hist(x=olist.delivery_estimate_discrepancy, bins = 70) #sets number of bin
s. Max price is 6.7K, so the graph is heavily skewed.
plt.title('Distribution of Discrepancy') #labels
plt.xlabel('Days_diff.Actual_Estimate')
plt.ylabel('Count')

plt.show()

#This shows a long tailed distribution likely because of the extreme min and m
ax values.
```

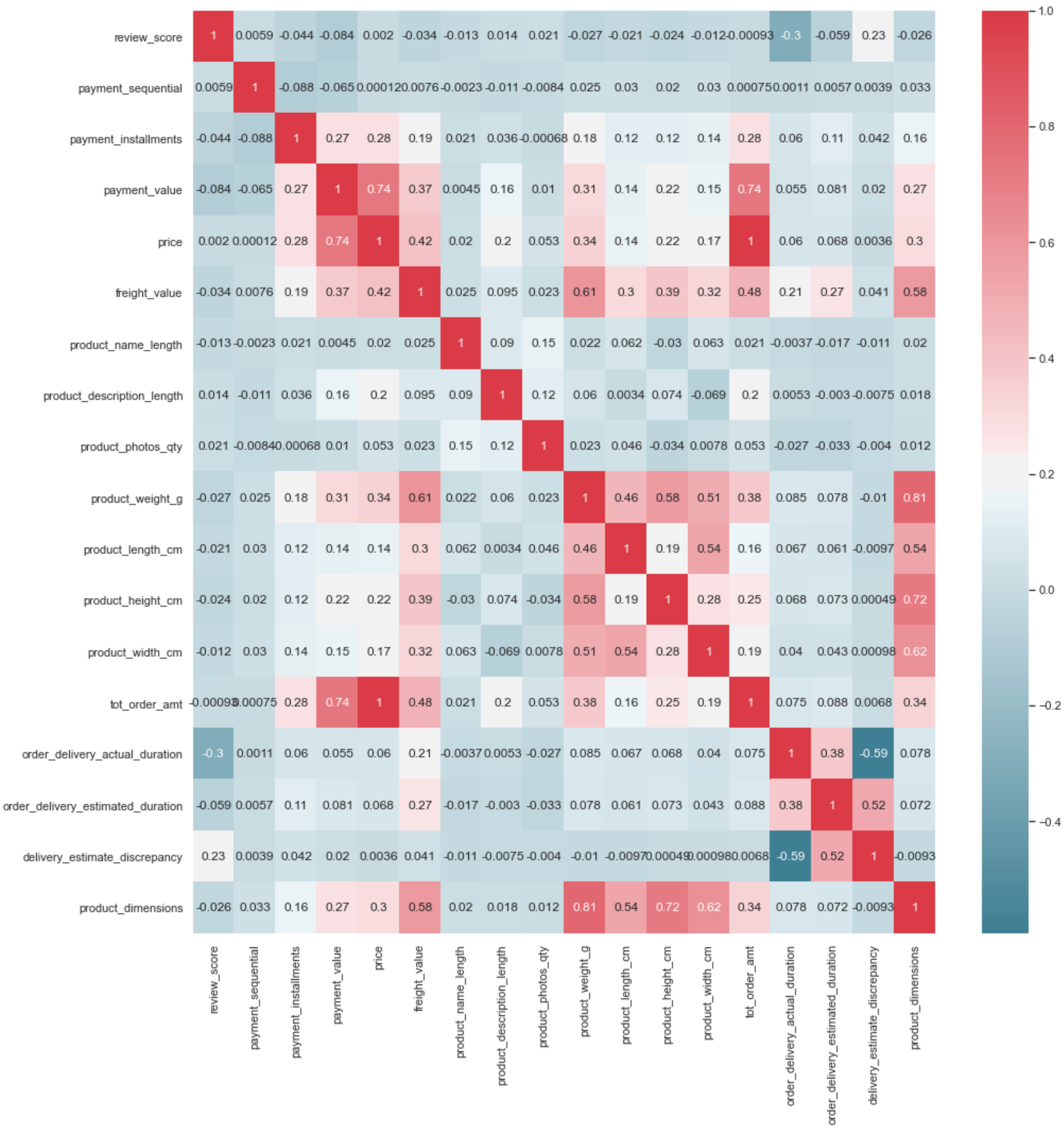


In [143]:
```
#What values form majority of our dataset.
#What is our 50%, 80% and 90% quantile
np.quantile(olist['delivery_estimate_discrepancy'], 0.5), np.quantile(olist['d
elivery_estimate_discrepancy'], 0.8), np.quantile(olist['delivery_estimate_dis
crepancy'], 0.95)

#So any value different from 22 is outside 90% of the data series.
```

Out[143]: (12.0, 18.0, 26.0)

In [144]:

```
#Correlations
# let's try a heatmap
# plot the correlation matrix using seaborn
sns.set(style="darkgrid") # one of the many styles to plot using

cmap = sns.diverging_palette(220, 10, as_cmap=True) # one of the many color ma
ppings

f, ax = plt.subplots(figsize=(15,15))

sns.heatmap(olist.corr(), cmap=cmap, annot=True)

f.tight_layout()
```



Correlation plot

- raw delivery_estimate_discrepancy and reveiw_score has 0.23 correlation coefficient
  - this is a good correlation relative to other correlation coefficient in the dataset.
- This means that customer satisfaction is improved by early delivery.

In [145]:
```
estimate_discrepancy = olist.copy()['delivery_estimate_discrepancy'].quantile(
0.95)
```

In [146]:

```
estimate_discrepancy = []
estimate_discrepancy_out = []

thresh = np.quantile(olist.copy()['delivery_estimate_discrepancy'], 0.95)
[estimate_discrepancy.append(i)
 if abs(i) <= thresh else estimate_discrepancy_out.append(i)
 for i in olist.copy()['delivery_estimate_discrepancy']]
#y = np.array(y)
```

```
Out[146]: [None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
    None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
```

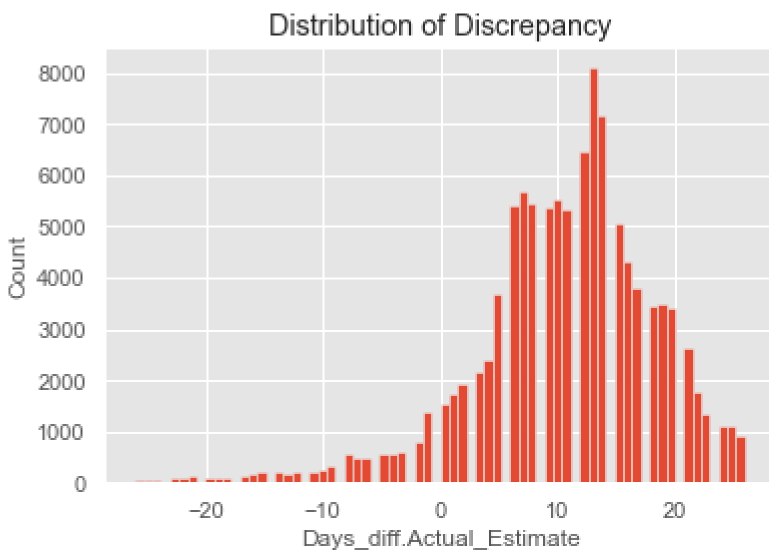
```
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 ...]
```

In [147]:
```
len(estimate_discrepancy), len(estimate_discrepancy_out)
```

Out[147]:
```
(108184, 5883)
```

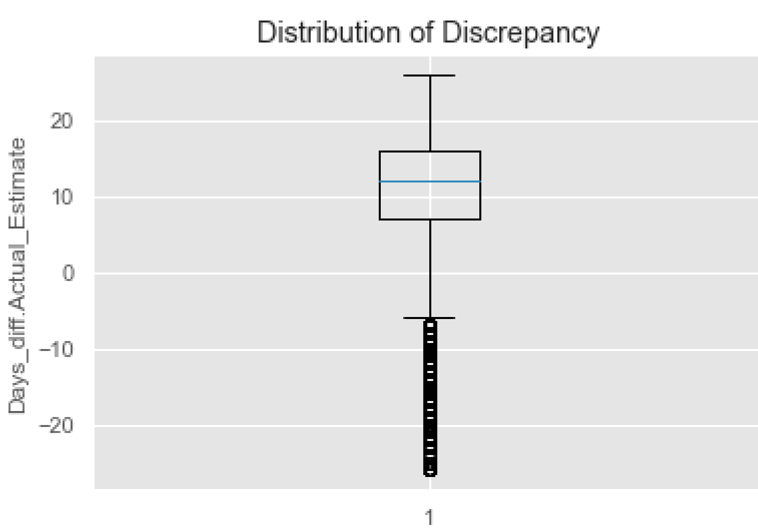
In [148]:
```
estimate_discrepancy,  estimate_discrepancy_out= np.array(estimate_discrepancy
), np.array(estimate_discrepancy_out)
```

In [149]:
```
#Distribution of the discrepancy

plt.style.use('ggplot')

plt.hist(x=estimate_discrepancy, bins = 70) #sets number of bins. Max price is 6.7K, so the graph is heavily skewed.
plt.title('Distribution of Discrepancy') #labels
plt.xlabel('Days_diff.Actual_Estimate')
plt.ylabel('Count')

plt.show()

#This shows a long tailed distribution likely because of the extreme min and max values.
```



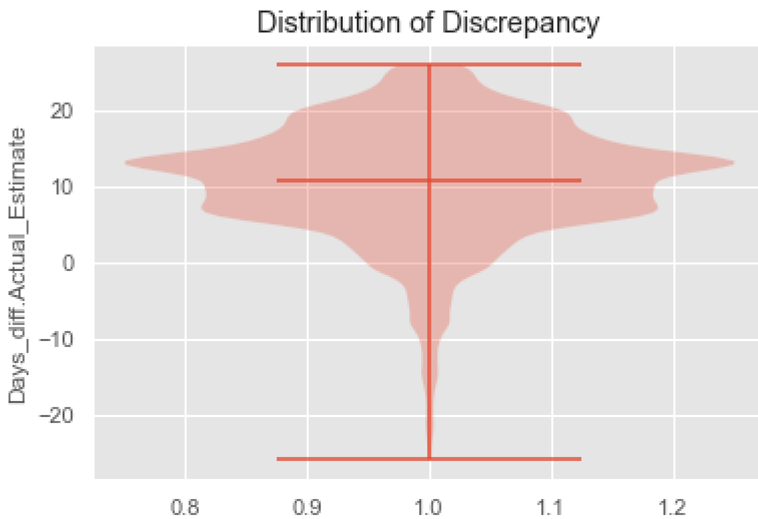
In [150]:
```
plt.boxplot(x=estimate_discrepancy)
plt.ylabel('Days_diff.Actual_Estimate')
plt.title('Distribution of Discrepancy') #labels
```

Out[150]: Text(0.5, 1.0, 'Distribution of Discrepancy')



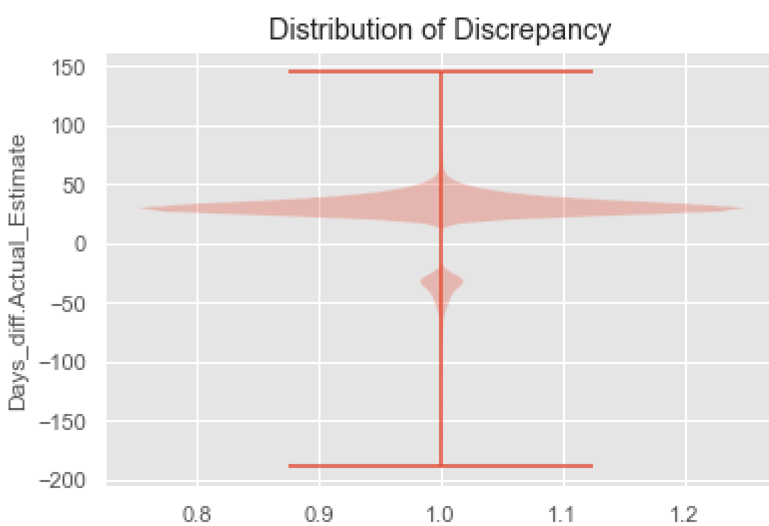
In [151]:
```
plt.violinplot(estimate_discrepancy, showmeans=True)
plt.ylabel('Days_diff.Actual_Estimate')
plt.title('Distribution of Discrepancy') #labels
```

Out[151]: Text(0.5, 1.0, 'Distribution of Discrepancy')



In [152]:
```
plt.violinplot(estimate_discrepancy_out)
plt.ylabel('Days_diff.Actual_Estimate')
plt.title('Distribution of Discrepancy') #labels
```

Out[152]: Text(0.5, 1.0, 'Distribution of Discrepancy')



In [153]:
```
olist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 114067 entries, 0 to 119147
Data columns (total 45 columns):
 #   Column                          Non-Null Count   Dtype
---  ------                          --------------   -----
 0   order_id                        114067 non-null  category
 1   customer_id                     114067 non-null  category
 2   order_purchase_timestamp        114067 non-null  datetime64[ns]
 3   order_approved_at               114067 non-null  datetime64[ns]
 4   order_delivered_carrier_date    114067 non-null  datetime64[ns]
 5   order_delivered_customer_date   114067 non-null  datetime64[ns]
 6   order_estimated_delivery_date   114067 non-null  datetime64[ns]
 7   customer_unique_id              114067 non-null  category
 8   customer_zip_code_prefix        114067 non-null  category
 9   customer_city                   114067 non-null  category
 10  customer_state                  114067 non-null  category
 11  review_id                       114067 non-null  category
 12  review_score                    114067 non-null  float64
 13  review_comment_title            13661 non-null   category
 14  review_comment_message          48220 non-null   category
 15  review_creation_date            114067 non-null  datetime64[ns]
 16  review_answer_timestamp         114067 non-null  datetime64[ns]
 17  payment_sequential              114067 non-null  float64
 18  payment_type                    114067 non-null  category
 19  payment_installments            114067 non-null  float64
 20  payment_value                   114067 non-null  float64
 21  order_item_id                   114067 non-null  category
 22  product_id                      114067 non-null  category
 23  seller_id                       114067 non-null  category
 24  shipping_limit_date             114067 non-null  datetime64[ns]
 25  price                           114067 non-null  float64
 26  freight_value                   114067 non-null  float64
 27  product_name_length             114067 non-null  float64
 28  product_description_length      114067 non-null  float64
 29  product_photos_qty              114067 non-null  float64
 30  product_weight_g                114067 non-null  float64
 31  product_length_cm               114067 non-null  float64
 32  product_height_cm               114067 non-null  float64
 33  product_width_cm                114067 non-null  float64
 34  product_category_english        114067 non-null  category
 35  seller_zip_code_prefix          114067 non-null  category
 36  seller_city                     114067 non-null  category
 37  seller_state                    114067 non-null  category
 38  tot_order_amt                   114067 non-null  float64
 39  purchase_wk_day                 114067 non-null  category
 40  purchase_month                  114067 non-null  category
 41  order_delivery_actual_duration  114067 non-null  int64
 42  order_delivery_estimated_duration 114067 non-null  int64
 43  delivery_estimate_discrepancy   114067 non-null  int64
 44  product_dimensions              114067 non-null  float64
dtypes: category(19), datetime64[ns](8), float64(15), int64(3)
memory usage: 45.6 MB
```

In [154]:
```
# let's break up the Discrepancy variable
#But needs outliers to be removed
olist['delivery_est_discrepancy_range'] = pd.cut(olist['delivery_estimate_disc
repancy'],
                                                 [-500, -7, -4, 4, 7,500],
                                                 5,
                                                 labels=['too late','late','on
target', 'early', 'too early']) # this creates a new variable
olist['delivery_est_discrepancy_range'].describe()
```

Out[154]:
```
count         114067
unique             5
top        too early
freq           81039
Name: delivery_est_discrepancy_range, dtype: object
```

In [155]:
```
#Convert the new binned column into categorical levels.
olist['delivery_est_discrepancy_range'] = olist.copy()['delivery_est_discrepan
cy_range'].astype("category")
olist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 114067 entries, 0 to 119147
Data columns (total 46 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   order_id                            114067 non-null  category
 1   customer_id                         114067 non-null  category
 2   order_purchase_timestamp            114067 non-null  datetime64[ns]
 3   order_approved_at                   114067 non-null  datetime64[ns]
 4   order_delivered_carrier_date        114067 non-null  datetime64[ns]
 5   order_delivered_customer_date       114067 non-null  datetime64[ns]
 6   order_estimated_delivery_date       114067 non-null  datetime64[ns]
 7   customer_unique_id                  114067 non-null  category
 8   customer_zip_code_prefix            114067 non-null  category
 9   customer_city                       114067 non-null  category
 10  customer_state                      114067 non-null  category
 11  review_id                           114067 non-null  category
 12  review_score                        114067 non-null  float64
 13  review_comment_title                13661 non-null   category
 14  review_comment_message              48220 non-null   category
 15  review_creation_date                114067 non-null  datetime64[ns]
 16  review_answer_timestamp             114067 non-null  datetime64[ns]
 17  payment_sequential                  114067 non-null  float64
 18  payment_type                        114067 non-null  category
 19  payment_installments                114067 non-null  float64
 20  payment_value                       114067 non-null  float64
 21  order_item_id                       114067 non-null  category
 22  product_id                          114067 non-null  category
 23  seller_id                           114067 non-null  category
 24  shipping_limit_date                 114067 non-null  datetime64[ns]
 25  price                               114067 non-null  float64
 26  freight_value                       114067 non-null  float64
 27  product_name_length                 114067 non-null  float64
 28  product_description_length          114067 non-null  float64
 29  product_photos_qty                  114067 non-null  float64
 30  product_weight_g                    114067 non-null  float64
 31  product_length_cm                   114067 non-null  float64
 32  product_height_cm                   114067 non-null  float64
 33  product_width_cm                    114067 non-null  float64
 34  product_category_english            114067 non-null  category
 35  seller_zip_code_prefix              114067 non-null  category
 36  seller_city                         114067 non-null  category
 37  seller_state                        114067 non-null  category
 38  tot_order_amt                       114067 non-null  float64
 39  purchase_wk_day                     114067 non-null  category
 40  purchase_month                      114067 non-null  category
 41  order_delivery_actual_duration      114067 non-null  int64
 42  order_delivery_estimated_duration   114067 non-null  int64
 43  delivery_estimate_discrepancy       114067 non-null  int64
 44  product_dimensions                  114067 non-null  float64
 45  delivery_est_discrepancy_range      114067 non-null  category
dtypes: category(20), datetime64[ns](8), float64(15), int64(3)
memory usage: 45.7 MB
```

In [156]:
```
#plt.boxplot(olist.review_score, olist.delivery_est_discrepancy_range)
#plt.ylabel('Days_diff.Actual_Estimate')
#plt.title('Distribution of Discrepancy') #labels
```

In [157]:
```
#plt.bar(olist.delivery_est_discrepancy_range, olist.review_score, label="line
ar")
```

In [ ]:
```

```

In [158]:

```
# the cross tab operator provides an easy way to get these numbers
discr_review = pd.crosstab([olist['purchase_month'],olist['review_score']],
                           olist['delivery_est_discrepancy_range'], normalize=
'index')
print (discr_review.sort_values(by='purchase_month'))
```

```
delivery_est_discrepancy_range  too late      late  on target     early  \
purchase_month review_score
April          1.0              0.152482  0.044326   0.085993  0.064716
               2.0              0.061425  0.027027   0.078624  0.130221
               3.0              0.030647  0.017026   0.103292  0.127128
               4.0              0.007944  0.003738   0.091121  0.127103
               5.0              0.003753  0.003590   0.052546  0.105744
August         1.0              0.095904  0.071928   0.191808  0.122877
               2.0              0.070496  0.057441   0.164491  0.167102
               3.0              0.018433  0.026498   0.239631  0.154378
               4.0              0.004985  0.005816   0.226007  0.188617
               5.0              0.002471  0.004682   0.236702  0.169723
December       5.0              0.005480  0.006057   0.054226  0.072974
               3.0              0.033272  0.009242   0.118299  0.120148
               4.0              0.006531  0.004082   0.093061  0.093878
               1.0              0.227328  0.055623   0.095526  0.085852
               2.0              0.104478  0.029851   0.111940  0.130597
February       1.0              0.337547  0.097345   0.123262  0.089128
               2.0              0.142500  0.030000   0.172500  0.157500
               3.0              0.066098  0.017058   0.170576  0.132196
               4.0              0.015748  0.011249   0.118110  0.134421
               5.0              0.006651  0.006046   0.093108  0.118702
January        1.0              0.188793  0.050862   0.085345  0.042241
               2.0              0.075419  0.025140   0.094972  0.097765
               3.0              0.021403  0.008323   0.137931  0.103448
               4.0              0.004678  0.004094   0.069591  0.097076
               5.0              0.004106  0.001955   0.046539  0.072155
July           4.0              0.006812  0.006812   0.095822  0.166667
               3.0              0.020720  0.007634   0.118866  0.157034
               5.0              0.002857  0.001905   0.083390  0.160114
               1.0              0.109405  0.033589   0.119002  0.111324
               2.0              0.036517  0.030899   0.081461  0.132022
June           1.0              0.071358  0.014866   0.045590  0.069376
               2.0              0.016949  0.005650   0.048023  0.090395
               3.0              0.014269  0.003567   0.053508  0.093936
               4.0              0.007282  0.003398   0.037864  0.091262
               5.0              0.001492  0.001194   0.026414  0.078943
March          1.0              0.359598  0.095717   0.123744  0.082496
               2.0              0.224586  0.047281   0.217494  0.106383
               3.0              0.075630  0.032680   0.224090  0.147526
               4.0              0.022130  0.017520   0.168741  0.183495
               5.0              0.010129  0.007335   0.130981  0.190884
May            5.0              0.006307  0.005484   0.099945  0.126542
               3.0              0.043186  0.022073   0.185221  0.106526
               4.0              0.010757  0.011998   0.133223  0.139843
               1.0              0.135181  0.057935   0.093199  0.099916
               2.0              0.092025  0.033742   0.138037  0.156442
November       1.0              0.314013  0.077770   0.134996  0.102715
               2.0              0.161473  0.062323   0.147309  0.152975
               3.0              0.057485  0.022754   0.208383  0.158084
               4.0              0.019406  0.009703   0.166161  0.197696
               5.0              0.007893  0.008551   0.129796  0.180224
October        5.0              0.003054  0.004582   0.075443  0.131949
               4.0              0.001742  0.005226   0.092334  0.146341
               2.0              0.041420  0.017751   0.147929  0.106509
               1.0              0.108470  0.032689   0.089153  0.101040
               3.0              0.016097  0.026157   0.154930  0.134809
```

```
September      4.0              0.007172  0.009221   0.110656  0.181352
               1.0              0.160083  0.027027   0.133056  0.137214
               2.0              0.113333  0.033333   0.100000  0.160000
               3.0              0.029730  0.013514   0.118919  0.154054
               5.0              0.004024  0.002347   0.089537  0.152582

delivery_est_discrepancy_range  too early
purchase_month review_score
April          1.0               0.652482
               2.0               0.702703
               3.0               0.721907
               4.0               0.770093
               5.0               0.834367
August         1.0               0.517483
               2.0               0.540470
               3.0               0.561060
               4.0               0.574574
               5.0               0.586422
December       5.0               0.861263
               3.0               0.719039
               4.0               0.802449
               1.0               0.535671
               2.0               0.623134
February       1.0               0.352718
               2.0               0.497500
               3.0               0.614072
               4.0               0.720472
               5.0               0.775494
January        1.0               0.632759
               2.0               0.706704
               3.0               0.728894
               4.0               0.824561
               5.0               0.875244
July           4.0               0.723887
               3.0               0.695747
               5.0               0.751734
               1.0               0.626679
               2.0               0.719101
June           1.0               0.798811
               2.0               0.838983
               3.0               0.834721
               4.0               0.860194
               5.0               0.891956
March          1.0               0.338445
               2.0               0.404255
               3.0               0.520075
               4.0               0.608114
               5.0               0.660671
May            5.0               0.761722
               3.0               0.642994
               4.0               0.704179
               1.0               0.613770
               2.0               0.579755
November       1.0               0.370506
               2.0               0.475921
               3.0               0.553293
               4.0               0.607035
```

```
                5.0               0.673537
October         5.0               0.784973
                4.0               0.754355
                2.0               0.686391
                1.0               0.668648
                3.0               0.668008
September       4.0               0.691598
                1.0               0.542620
                2.0               0.593333
                3.0               0.683784
                5.0               0.751509
```

In [159]:
```
# the cross tab operator provides an easy way to get these numbers
discr_review = pd.crosstab([olist['review_score']],
                           olist['delivery_est_discrepancy_range'])#, normaliz
e='index')
print (discr_review)
```

```
delivery_est_discrepancy_range  too late  late  on target  early  too early
review_score
1.0                                 2814   822       1485   1192       7035
2.0                                  389   136        503    521       2398
3.0                                  365   171       1521   1270       6315
4.0                                  218   174       2647   3208      15634
5.0                                  308   284       6405   8595      49657
```

In [160]:
```
discr_review.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Float64Index: 5 entries, 1.0 to 5.0
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   too late   5 non-null      int64
 1   late       5 non-null      int64
 2   on target  5 non-null      int64
 3   early      5 non-null      int64
 4   too early  5 non-null      int64
dtypes: int64(5)
memory usage: 240.0 bytes
```

```
In [162]: olist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 114067 entries, 0 to 119147
Data columns (total 46 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   order_id                           114067 non-null  category
 1   customer_id                        114067 non-null  category
 2   order_purchase_timestamp           114067 non-null  datetime64[ns]
 3   order_approved_at                  114067 non-null  datetime64[ns]
 4   order_delivered_carrier_date       114067 non-null  datetime64[ns]
 5   order_delivered_customer_date      114067 non-null  datetime64[ns]
 6   order_estimated_delivery_date      114067 non-null  datetime64[ns]
 7   customer_unique_id                 114067 non-null  category
 8   customer_zip_code_prefix           114067 non-null  category
 9   customer_city                      114067 non-null  category
 10  customer_state                     114067 non-null  category
 11  review_id                          114067 non-null  category
 12  review_score                       114067 non-null  float64
 13  review_comment_title               13661 non-null   category
 14  review_comment_message             48220 non-null   category
 15  review_creation_date               114067 non-null  datetime64[ns]
 16  review_answer_timestamp            114067 non-null  datetime64[ns]
 17  payment_sequential                 114067 non-null  float64
 18  payment_type                       114067 non-null  category
 19  payment_installments               114067 non-null  float64
 20  payment_value                      114067 non-null  float64
 21  order_item_id                      114067 non-null  category
 22  product_id                         114067 non-null  category
 23  seller_id                          114067 non-null  category
 24  shipping_limit_date                114067 non-null  datetime64[ns]
 25  price                              114067 non-null  float64
 26  freight_value                      114067 non-null  float64
 27  product_name_length                114067 non-null  float64
 28  product_description_length         114067 non-null  float64
 29  product_photos_qty                 114067 non-null  float64
 30  product_weight_g                   114067 non-null  float64
 31  product_length_cm                  114067 non-null  float64
 32  product_height_cm                  114067 non-null  float64
 33  product_width_cm                   114067 non-null  float64
 34  product_category_english           114067 non-null  category
 35  seller_zip_code_prefix             114067 non-null  category
 36  seller_city                        114067 non-null  category
 37  seller_state                       114067 non-null  category
 38  tot_order_amt                      114067 non-null  float64
 39  purchase_wk_day                    114067 non-null  category
 40  purchase_month                     114067 non-null  category
 41  order_delivery_actual_duration     114067 non-null  int64
 42  order_delivery_estimated_duration  114067 non-null  int64
 43  delivery_estimate_discrepancy      114067 non-null  int64
 44  product_dimensions                 114067 non-null  float64
 45  delivery_est_discrepancy_range     114067 non-null  category
dtypes: category(20), datetime64[ns](8), float64(15), int64(3)
memory usage: 45.7 MB
```

In [163]:
```
olist_subset = olist[['review_score', 'delivery_est_discrepancy_range']]
olist_subset.head()
```

Out[163]:

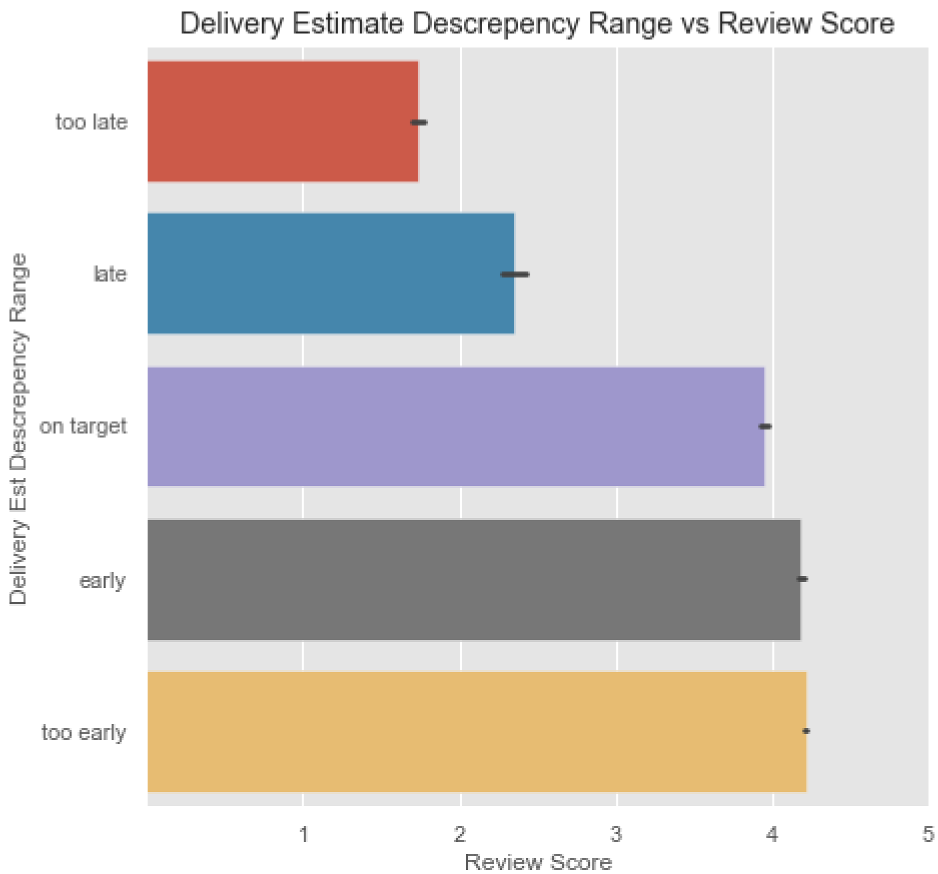
	review_score	delivery_est_discrepancy_range
0	4.0	early
1	4.0	early
2	4.0	early
3	4.0	early
4	5.0	too early

In [164]:
```
olist_subset.groupby('delivery_est_discrepancy_range').mean()
```

Out[164]:

	review_score
delivery_est_discrepancy_range	
too late	1.734001
late	2.345936
on target	3.954064
early	4.183079
too early	4.215217

In [165]:
```
# plot Delivery Estimate Descrepency Range
fig, ax = plt.subplots(figsize=(7,7))
ticks = [1,2,3,4,5]
ax = sns.barplot(x='review_score', y = 'delivery_est_discrepancy_range', data
= olist_subset)
ax.set(xlabel='Review Score',
       ylabel='Delivery Est Descrepency Range',
      title='Delivery Estimate Descrepency Range vs Review Score')
plt.xticks(ticks)
plt.show()
```



In [166]:
```
#import matplotlib.pyplot as plt
#fig = plt.figure()
#ax = fig.add_axes([0,0,1,1])
#ax.bar(olist['review_score'], olist['delivery_est_discrepancy_range'])
#plt.show()
```

In [167]:
```
discr_review.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Float64Index: 5 entries, 1.0 to 5.0
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   too late   5 non-null      int64
 1   late       5 non-null      int64
 2   on target  5 non-null      int64
 3   early      5 non-null      int64
 4   too early  5 non-null      int64
dtypes: int64(5)
memory usage: 240.0 bytes
```

In [168]:
```
# let's break up the Discrepancy variable
#But needs outliers to be removed
olist['review_score_class'] = pd.cut(olist['review_score'],
                                     [0, 2, 3, 5],
                                     3,
                                     labels=['bad','fair','good'])
# this creates a new variable
olist['review_score_class'] = olist.copy()['review_score_class'].astype("categ
ory")
olist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 114067 entries, 0 to 119147
Data columns (total 47 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   order_id                            114067 non-null  category
 1   customer_id                         114067 non-null  category
 2   order_purchase_timestamp            114067 non-null  datetime64[ns]
 3   order_approved_at                   114067 non-null  datetime64[ns]
 4   order_delivered_carrier_date        114067 non-null  datetime64[ns]
 5   order_delivered_customer_date       114067 non-null  datetime64[ns]
 6   order_estimated_delivery_date       114067 non-null  datetime64[ns]
 7   customer_unique_id                  114067 non-null  category
 8   customer_zip_code_prefix            114067 non-null  category
 9   customer_city                       114067 non-null  category
 10  customer_state                      114067 non-null  category
 11  review_id                           114067 non-null  category
 12  review_score                        114067 non-null  float64
 13  review_comment_title                13661 non-null   category
 14  review_comment_message              48220 non-null   category
 15  review_creation_date                114067 non-null  datetime64[ns]
 16  review_answer_timestamp             114067 non-null  datetime64[ns]
 17  payment_sequential                  114067 non-null  float64
 18  payment_type                        114067 non-null  category
 19  payment_installments                114067 non-null  float64
 20  payment_value                       114067 non-null  float64
 21  order_item_id                       114067 non-null  category
 22  product_id                          114067 non-null  category
 23  seller_id                           114067 non-null  category
 24  shipping_limit_date                 114067 non-null  datetime64[ns]
 25  price                               114067 non-null  float64
 26  freight_value                       114067 non-null  float64
 27  product_name_length                 114067 non-null  float64
 28  product_description_length          114067 non-null  float64
 29  product_photos_qty                  114067 non-null  float64
 30  product_weight_g                    114067 non-null  float64
 31  product_length_cm                   114067 non-null  float64
 32  product_height_cm                   114067 non-null  float64
 33  product_width_cm                    114067 non-null  float64
 34  product_category_english            114067 non-null  category
 35  seller_zip_code_prefix              114067 non-null  category
 36  seller_city                         114067 non-null  category
 37  seller_state                        114067 non-null  category
 38  tot_order_amt                       114067 non-null  float64
 39  purchase_wk_day                     114067 non-null  category
 40  purchase_month                      114067 non-null  category
 41  order_delivery_actual_duration      114067 non-null  int64
 42  order_delivery_estimated_duration   114067 non-null  int64
 43  delivery_estimate_discrepancy       114067 non-null  int64
 44  product_dimensions                  114067 non-null  float64
 45  delivery_est_discrepancy_range      114067 non-null  category
 46  review_score_class                  114067 non-null  category
dtypes: category(21), datetime64[ns](8), float64(15), int64(3)
memory usage: 50.8 MB
```

In [169]:
```
# verify that the cut function worked as expected.
range_review = pd.crosstab([olist['review_score_class']],
                           olist['review_score'])
print (range_review)
```

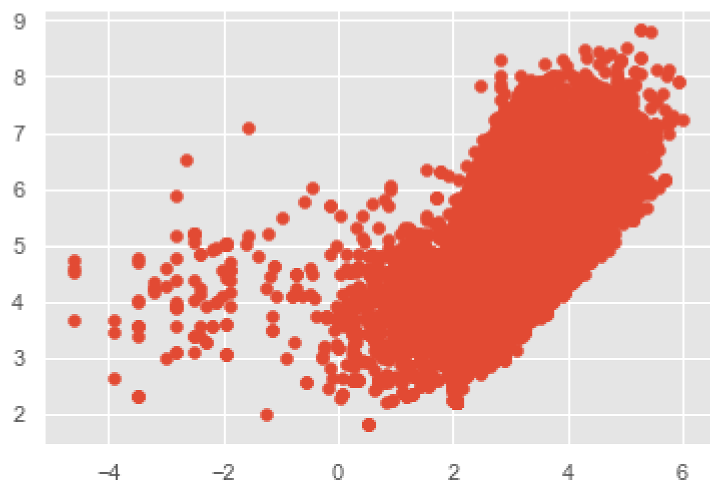
```
review_score        1.0   2.0   3.0    4.0    5.0
review_score_class
bad               13348  3947     0      0      0
fair                  0     0  9642      0      0
good                  0     0     0  21881  65249
```

In [170]:
```
olist['review_score'].unique()
```

Out[170]:
```
array([4., 5., 1., 2., 3.])
```

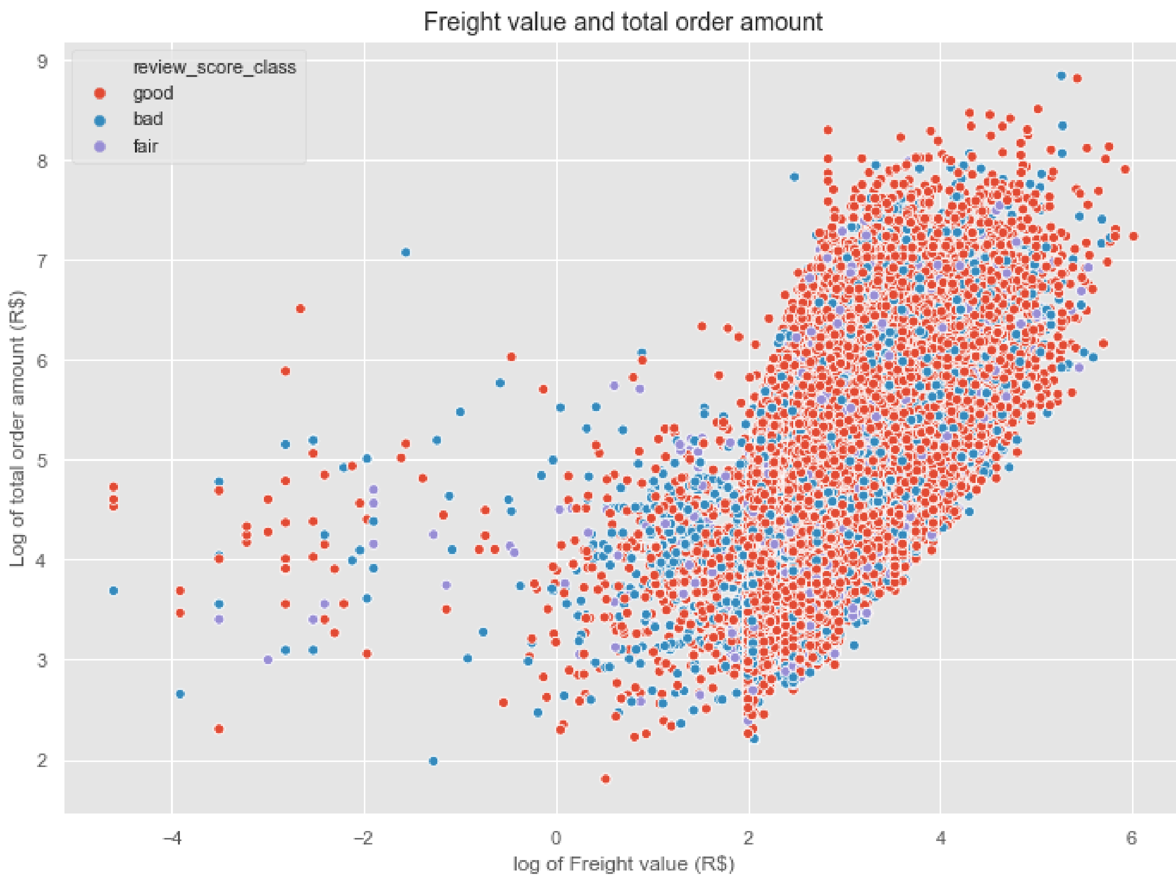
In [171]:
```
plt.scatter(x=np.log(olist.freight_value), y= np.log(olist.tot_order_amt))
plt.show()
```

```
C:\Users\justi\anaconda3\lib\site-packages\pandas\core\series.py:679: Runtime
Warning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



In [172]:
```python
a4_dims = (11.7, 8.27)
fig, ax = plt.subplots(figsize=a4_dims)
ax = sns.scatterplot(x=np.log(olist.freight_value), y= np.log(olist.tot_order_amt), hue = olist.review_score_class)
ax.set(xlabel = 'log of Freight value (R$)', ylabel = 'Log of total order amount (R$)',
       title = 'Freight value and total order amount')
```

Out[172]:
```
[Text(0, 0.5, 'Log of total order amount (R$)'),
 Text(0.5, 0, 'log of Freight value (R$)'),
 Text(0.5, 1.0, 'Freight value and total order amount')]
```



In [173]:
```python
# create a new df for average delay iin delivery by state
df4_new_Delay = olist.copy()
df4_new_Delay['Delay_Delivery'] = df4_new_Delay['order_estimated_delivery_date'] - df4_new_Delay['order_delivered_customer_date']
df4_new_Delay['Delay_Delivery'] = df4_new_Delay['Delay_Delivery'].astype('timedelta64[D]')
df4_new_Delay_stat = df4_new_Delay.groupby('customer_state',as_index=False,sort=True).agg({'Delay_Delivery':'mean'})
df4_new_Delay_stat = df4_new_Delay_stat.sort_values(by = ['Delay_Delivery', 'customer_state'], ascending = [False, True])
```

In [174]:
```
# chart average delay iin delivery by state
fig = plt.figure(figsize = (10,10))
ax = fig.add_axes([0,0,1,1])
y=df4_new_Delay_stat['customer_state']
width = df4_new_Delay_stat['Delay_Delivery']
ax.barh(y,width, color='blue')
ax.set(xlabel='Average Delivery Delay in Days',
       ylabel='Brazilian State',
      title='Average Delay in Delivery - State Level')
plt.show()
```