DSC 630

Course Project

Author: Kimberly Cable
Term: Fall, 2022

Preliminary Analysis

In [1]:
```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.cm as cm
import seaborn as sns
from scipy import stats
from datetime import datetime

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
```

In [2]:
```
# Supress warnings
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
# Read datasets
customers_df = pd.read_csv("data/olist_customers_dataset.csv")
geoloc_df = pd.read_csv("data/olist_geolocation_dataset.csv")
orderitems_df = pd.read_csv("data/olist_order_items_dataset.csv")
orderpay_df = pd.read_csv("data/olist_order_payments_dataset.csv")
orderreviews_df = pd.read_csv("data/olist_order_reviews_dataset.csv")
orders_df = pd.read_csv("data/olist_orders_dataset.csv")
products_df = pd.read_csv("data/olist_products_dataset.csv")
sellers_df = pd.read_csv("data/olist_sellers_dataset.csv")
catname_df = pd.read_csv("data/product_category_name_translation.csv")
```

Customers

In [4]:
```
customers_df.head()
```

Out[4]:

	customer_id	customer_unique_id	customer_zip_code_prefix
0	06b8999e2fba1a1fbc88172c00ba8bc7	861eff4711a542e4b93843c6dd7febb0	14409
1	18955e83d337fd6b2def6b18a428ac77	290c77bc529b7ac935b93aa66c333dc3	9790
2	4e7b3e00288586ebd08712fdd0374a03	060e732b5b29e8181a18229c7b0b2b5e	1151
3	b2b6027bc5c5109e529d4dc6358b12c3	259dac757896d24d7702b9acbbff3f3c	8775
4	4f2d8ab171c80ec8364f7c12e35b23ad	345ecd01c38d18a9036ed96c73b8d066	13056

In [5]:

```
customers_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 5 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   customer_id               99441 non-null  object
 1   customer_unique_id        99441 non-null  object
 2   customer_zip_code_prefix  99441 non-null  int64 
 3   customer_city             99441 non-null  object
 4   customer_state            99441 non-null  object
dtypes: int64(1), object(4)
memory usage: 3.8+ MB
```

In [6]:

```
# Change Zip code to string
customers_df['customer_zip_code_prefix'] = customers_df['customer_zip_code_prefix'].ap
```

In [7]:

```
customers_df.isnull().sum()
```

Out[7]:

```
customer_id                 0
customer_unique_id          0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64
```

In [8]:

```
# Drop customer_unique_id, zip code prefix, and city as they are not needed
customers_df.drop(['customer_unique_id', 'customer_zip_code_prefix', 'customer_city'],
```

In [9]:

```
customers_df.head()
```

Out[9]:

	customer_id	customer_state
0	06b8999e2fba1a1fbc88172c00ba8bc7	SP
1	18955e83d337fd6b2def6b18a428ac77	SP
2	4e7b3e00288586ebd08712fdd0374a03	SP
3	b2b6027bc5c5109e529d4dc6358b12c3	SP
4	4f2d8ab171c80ec8364f7c12e35b23ad	SP

In [10]:

```
customers_df.shape
```

Out[10]: (99441, 2)

Orders

In [11]:
```
orders_df.head()
```

Out[11]:

	order_id	customer_id	order_status	order_purcha
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-
2	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	2018-
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-

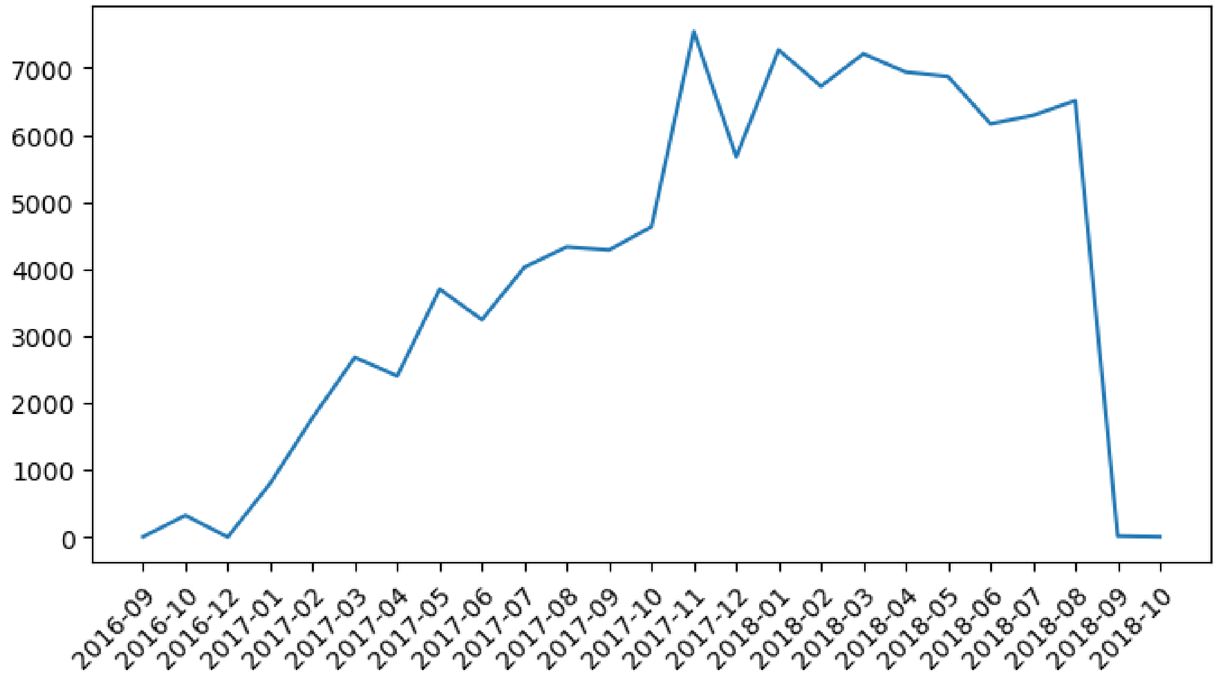
In [12]:
```
orders_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99441 entries, 0 to 99440
Data columns (total 8 columns):
 #   Column                         Non-Null Count  Dtype 
---  ------                         --------------  ----- 
 0   order_id                       99441 non-null  object
 1   customer_id                    99441 non-null  object
 2   order_status                   99441 non-null  object
 3   order_purchase_timestamp       99441 non-null  object
 4   order_approved_at              99281 non-null  object
 5   order_delivered_carrier_date   97658 non-null  object
 6   order_delivered_customer_date  96476 non-null  object
 7   order_estimated_delivery_date  99441 non-null  object
dtypes: object(8)
memory usage: 6.1+ MB
```

In [13]:
```
# Convert date columns to datetime
orders_df['order_purchase_timestamp'] = pd.to_datetime(orders_df['order_purchase_times
orders_df['order_approved_at'] = pd.to_datetime(orders_df['order_approved_at'])
orders_df['order_delivered_carrier_date'] = pd.to_datetime(orders_df['order_delivered_
orders_df['order_delivered_customer_date'] = pd.to_datetime(orders_df['order_delivered
orders_df['order_estimated_delivery_date'] = pd.to_datetime(orders_df['order_estimated
```

In [14]:
```
# Check purchase timestamp over time
orders_df_copy = orders_df.copy()
orders_df_copy['year_month'] = orders_df_copy['order_purchase_timestamp'].map(lambda 

group_year_month = orders_df_copy.groupby('year_month')['order_id'].size().to_frame("

fig, ax = plt.subplots(figsize = (8, 4))
```

```
plt.plot(group_year_month['year_month'], group_year_month['count'])
plt.xticks(rotation = 45, ha = 'right', rotation_mode = 'anchor')

plt.show()
```



In [15]:
```
orders_df.shape
```

Out[15]:
```
(99441, 8)
```

In [16]:
```
# Delete order before Jan, 2017 and After Aug, 2018
orders_df = orders_df.loc[(orders_df['order_purchase_timestamp'] > '2016-12-31') & (or
```

In [17]:
```
orders_df.shape
```

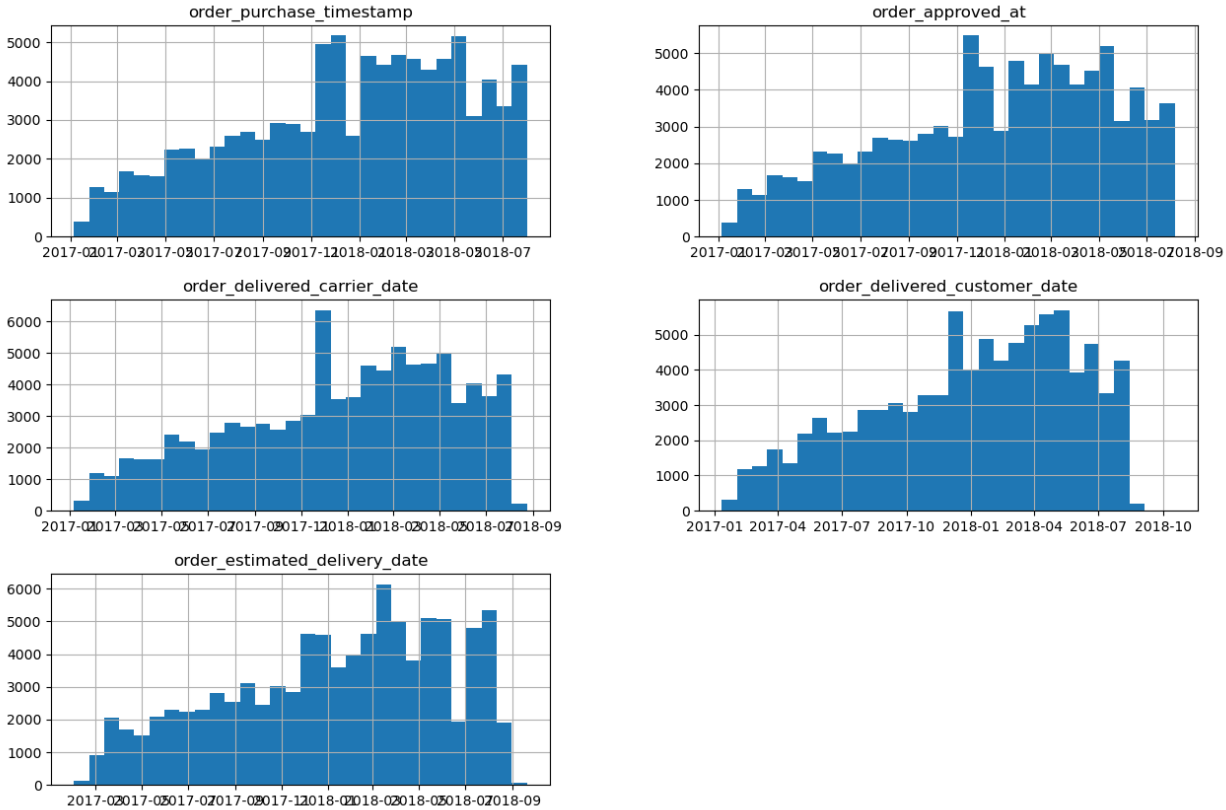
Out[17]:
```
(92580, 8)
```

In [18]:
```
# Check for empty values
orders_df.isnull().sum()
```

Out[18]:
```
order_id                            0
customer_id                         0
order_status                        0
order_purchase_timestamp            0
order_approved_at                  82
order_delivered_carrier_date     1602
order_delivered_customer_date    2727
order_estimated_delivery_date       0
dtype: int64
```

In [19]:
```
orders_df.hist(bins = 30, figsize = (15, 10))
```

Out[19]:
```
array([[<AxesSubplot:title={'center':'order_purchase_timestamp'}>,
        <AxesSubplot:title={'center':'order_approved_at'}>],
       [<AxesSubplot:title={'center':'order_delivered_carrier_date'}>,
        <AxesSubplot:title={'center':'order_delivered_customer_date'}>],
       [<AxesSubplot:title={'center':'order_estimated_delivery_date'}>,
        <AxesSubplot:>]], dtype=object)
```



Order Status

In [20]:
```
# Check orders by order status
orders_df['order_status'].value_counts()
```

Out[20]:
```
delivered      89860
shipped         1050
unavailable      595
canceled         496
processing       299
invoiced         273
created            5
approved           2
Name: order_status, dtype: int64
```

Order Status - order_approved_at

In [21]:
```
# Check orders with no order approved at and their order status
orders_df[orders_df['order_approved_at'].isna()]['order_status'].value_counts()
```

Out[21]:
```
canceled     63
delivered    14
created       5
Name: order_status, dtype: int64
```

A delivered order status should have an order approval date

In [22]:
```
approval_check = ((orders_df['order_approved_at'].isna()) & (orders_df['order_status']
orders_df[approval_check]
```

Out[22]:

	order_id	customer_id	order_status	order_p
5323	e04abd8149ef81b95221e88f6ed9ab6a	2127dc6603ac33544953ef05ec155771	delivered	
16567	8a9adc69528e1001fc68dd0aaebbb54a	4c1ccc74e00993733742a3c786dc3c1f	delivered	
19031	7013bcfc1c97fe719a7b5e05e61c12db	2941af76d38100e0f8740a374f1a5dc3	delivered	
22663	5cf925b116421afa85ee25e99b4c34fb	29c35fc91fc13fb5073c8f30505d860d	delivered	
23156	12a95a3c06dbaec84bcfb0e2da5d228a	1e101e0daffaddce8159d25a8e53f2b2	delivered	
26800	c1d4211b3dae76144deccd6c74144a88	684cb238dc5b5d6366244e0e0776b450	delivered	
38290	d69e5d356402adc8cf17e08b5033acfb	68d081753ad4fe22fc4d410a9eb1ca01	delivered	
39334	d77031d6a3c8a52f019764e68f211c69	0bf35cac6cc7327065da879e2d90fae8	delivered	
48401	7002a78c79c519ac54022d4f8a65e6e8	d5de688c321096d15508faae67a27051	delivered	
61743	2eecb0d85f281280f79fa00f9cec1a95	a3d3c38e58b9d2dfb9207cab690b6310	delivered	
63052	51eb2eebd5d76a24625b31c33dd41449	07a2a7e0f63fd8cb757ed77d4245623c	delivered	
67697	88083e8f64d95b932164187484d90212	f67cd1a215aae2a1074638bbd35a223a	delivered	
72407	3c0b8706b065f9919d0505d3b3343881	d85919cb3c0529589c6fa617f5f43281	delivered	
84999	2babbb4b15e6d2dfe95e2de765c97bce	74bebaf46603f9340e3b50c6b086f992	delivered	

In [23]:

```
# Use the order purchase timestamp as the order approved at
orders_df.loc[approval_check, 'order_approved_at'] = orders_df.loc[approval_check, 'or
```

In [24]:

```
# Check orders with no order approved at and their order status
orders_df[orders_df['order_approved_at'].isna()]['order_status'].value_counts()
```

Out[24]:

```
canceled    63
created      5
Name: order_status, dtype: int64
```

Order Status - order_delivered_carrier_date

In [25]:

```
# Check orders with no order delivered carrier date at and their order status
orders_df[orders_df['order_delivered_carrier_date'].isna()]['order_status'].value_cour
```

Out[25]:

```
unavailable    595
canceled       426
processing     299
invoiced       273
created          5
approved         2
delivered        2
Name: order_status, dtype: int64
```

The delivered status should have an order delivered carrier date.

In [26]:

```
carrier_check = ((orders_df['order_delivered_carrier_date'].isna()) & (orders_df['orde
orders_df[carrier_check]
```

Out[26]:

	order_id	customer_id	order_status	order_pur
73222	2aa91108853cecb43c84a5dc5b277475	afeb16c7f46396c0ed54acb45ccaaa40	delivered	20
92643	2d858f451373b04fb5c984a1cc2defaf	e08caf668d499a6d643dafd7c5cc498a	delivered	20

We will use the order approved at date for the the order delivered date

In [27]:
```
orders_df.loc[carrier_check, 'order_delivered_carrier_date'] = orders_df.loc[carrier_c
```

In [28]:
```
orders_df[orders_df['order_delivered_carrier_date'].isna()]['order_status'].value_cour
```

Out[28]:
```
unavailable    595
canceled       426
processing     299
invoiced       273
created          5
approved         2
Name: order_status, dtype: int64
```

Order Status - order_delivered_customer_date

In [29]:
```
# Check orders with no order delivered customer date at and their order status
orders_df[orders_df['order_delivered_customer_date'].isna()]['order_status'].value_cou
```

Out[29]:
```
shipped        1050
unavailable     595
canceled        495
processing      299
invoiced        273
delivered         8
created           5
approved          2
Name: order_status, dtype: int64
```

The delivered status should have a date in it

In [30]:
```
customerdate_check = ((orders_df['order_delivered_customer_date'].isna()) & (orders_df
orders_df[customerdate_check]
```

Out[30]:

	order_id	customer_id	order_status	order_pu
3002	2d1e2d5bf4dc7227b3bfebb81328c15f	ec05a6d8558c6455f0cbbd8a420ad34f	delivered	2
20618	f5dd62b788049ad9fc0526e3ad11a097	5e89028e024b381dc84a13a3570decb4	delivered	2
43834	2ebdfc4f15f23b91474edf87475f108e	29f0540231702fda0cfdee0a310f11aa	delivered	2
79263	e69f75a717d64fc5ecdfae42b2e8e086	cfda40ca8dd0a5d486a9635b611b398a	delivered	2
82868	0d3268bad9b086af767785e3f0fc0133	4f1d63d35fb7c8999853b2699f5c7649	delivered	2
92643	2d858f451373b04fb5c984a1cc2defaf	e08caf668d499a6d643dafd7c5cc498a	delivered	2
97647	ab7c89dc1bf4a1ead9d6ec1ec8968a84	dd1b84a7286eb4524d52af4256c0ba24	delivered	2
98038	20edc82cf5400ce95e1afacc25798b31	28c37425f1127d887d7337f284080a0f	delivered	2

To get the delivered customer date we will take the median the order delivered customer date - order delivered carrier date

In [31]:

```
orders_df['carrier_delivered_time'] = orders_df['order_delivered_customer_date'] - orc
orders_df.head()
```

Out[31]:

	order_id	customer_id	order_status	order_purchas
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-1
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-(
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-1
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-(
5	a4591c265e18cb1dcee52889e2d8acc3	503740e9ca751ccdda7ba28e9ab8f608	delivered	2017-(

In [32]:

```
mid_carrier_delivered_time = orders_df['carrier_delivered_time'].median()
mid_carrier_delivered_time
```

Out[32]:

```
Timedelta('7 days 05:21:22')
```

The median days between the two is 7 day so we will add these days to the order delivered customer date

In [33]:
```
orders_df.loc[customerdate_check, 'order_delivered_customer_date'] = orders_df.loc[cus

orders_df.loc[customerdate_check, 'carrier_delivered_time'] = mid_carrier_delivered_ti

orders_df[customerdate_check]
```

Out[33]:

	order_id	customer_id	order_status	order_pu
3002	2d1e2d5bf4dc7227b3bfebb81328c15f	ec05a6d8558c6455f0cbbd8a420ad34f	delivered	2
20618	f5dd62b788049ad9fc0526e3ad11a097	5e89028e024b381dc84a13a3570decb4	delivered	2
43834	2ebdfc4f15f23b91474edf87475f108e	29f0540231702fda0cfdee0a310f11aa	delivered	2
79263	e69f75a717d64fc5ecdfae42b2e8e086	cfda40ca8dd0a5d486a9635b611b398a	delivered	2
82868	0d3268bad9b086af767785e3f0fc0133	4f1d63d35fb7c8999853b2699f5c7649	delivered	2
92643	2d858f451373b04fb5c984a1cc2defaf	e08caf668d499a6d643dafd7c5cc498a	delivered	2
97647	ab7c89dc1bf4a1ead9d6ec1ec8968a84	dd1b84a7286eb4524d52af4256c0ba24	delivered	2
98038	20edc82cf5400ce95e1afacc25798b31	28c37425f1127d887d7337f284080a0f	delivered	2

In [34]:
```
# Check orders with no order delivered customer date at and their order status
orders_df[orders_df['order_delivered_customer_date'].isna()]['order_status'].value_cou
```

Out[34]:
```
shipped        1050
unavailable     595
canceled        495
processing      299
invoiced        273
created           5
approved          2
Name: order_status, dtype: int64
```

In [35]:
```
# Check empty values
orders_df.isna().sum()
```

Out[35]:
```
order_id                            0
customer_id                         0
order_status                        0
order_purchase_timestamp            0
order_approved_at                  68
order_delivered_carrier_date     1600
order_delivered_customer_date    2719
order_estimated_delivery_date       0
carrier_delivered_time           2719
dtype: int64
```

In [36]:
```
# Drop carrier delivered time as it is no longer needed
orders_df.drop(['carrier_delivered_time'], axis = 1, inplace = True)
```

In [37]:
```
orders_df.head()
```

Out[37]:

	order_id	customer_id	order_status	order_purcha
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-1
1	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-(
3	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	2017-1
4	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	2018-(
5	a4591c265e18cb1dcee52889e2d8acc3	503740e9ca751ccdda7ba28e9ab8f608	delivered	2017-(

In [38]:
```
orders_df.shape
```

Out[38]:
```
(92580, 8)
```

Order Items

In [39]:
```
orderitems_df.head()
```

Out[39]:

	order_id	order_item_id	product_id	
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade1
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032edd
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5(
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a

In [40]:
```
orderitems_df.isnull().sum()
```

Out[40]:
```
order_id               0
order_item_id          0
product_id             0
seller_id              0
shipping_limit_date    0
price                  0
freight_value          0
dtype: int64
```

In [41]:
```
orderitems_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112650 entries, 0 to 112649
Data columns (total 7 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   order_id             112650 non-null  object 
 1   order_item_id        112650 non-null  int64  
 2   product_id           112650 non-null  object 
 3   seller_id            112650 non-null  object 
 4   shipping_limit_date  112650 non-null  object 
 5   price                112650 non-null  float64
 6   freight_value        112650 non-null  float64
dtypes: float64(2), int64(1), object(4)
memory usage: 6.0+ MB
```

In [42]:
```
# Drop seller id, shipping limit date
orderitems_df.drop(['seller_id', 'shipping_limit_date'], axis = 1, inplace = True)
```

In [43]:
```
orderitems_df.head()
```

Out[43]:

	order_id	order_item_id	product_id	price	frei
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	58.90	
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	239.90	
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	199.00	
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	12.99	
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	199.90	

In [44]:
```
orderitems_df.shape
```

Out[44]:
```
(112650, 5)
```

Order Payments

In [45]:
```
orderpay_df.head()
```

Out[45]:

	order_id	payment_sequential	payment_type	payment_installments	pay
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	8	
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	2	

In [46]:
```
orderpay_df.shape
```

Out[46]:
```
(103886, 5)
```

The payment value is the price and freight together so I dropped this dataset as it was not needed

Order Reviews

In [47]:
```
orderreviews_df.head()
```

Out[47]:

	review_id	order_id	review_score	review_comr
0	7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	4	
1	80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	5	
2	228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	5	
3	e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	5	
4	f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	5	

As this analysis is on customer segmentation, I will keep only the order_id and review_score. The rest may be used for sentiment analysis at a later time.

In [48]:
```
# Drop all columns except order_id and review_score
orderreviews_df.drop(['review_id', 'review_comment_title', 'review_comment_message', '
                     axis = 1, inplace = True)
```

In [49]:
```
orderreviews_df.isnull().sum()
```

Out[49]:
```
order_id        0
review_score    0
dtype: int64
```

In [50]:
```
orderreviews_df.describe()
```

Out[50]:

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

In [51]:
```
orderreviews_df.shape
```

Out[51]: (99224, 2)

Products

In [52]:
```
products_df.head()
```

Out[52]:

	product_id	product_category_name	product_name_lenght	product_descrip
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 [53]:
```
# Merge the category names in english with the products and remove the portuguese name
products_df = pd.merge(products_df, catname_df, on='product_category_name', how='left'
products_df.head(3)
```

Out[53]:

	product_id	product_category_name	product_name_lenght	product_descrip
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40.0	
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44.0	
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	46.0	

In [54]:
```
# Remove unneccessary columns
products_df.drop(columns=["product_name_lenght", "product_description_lenght",
                          "product_photos_qty", "product_weight_g", "product_length_cm"
                          "product_height_cm", "product_width_cm"], axis = 1, inplace =
products_df.head()
```

Out[54]:

	product_id	product_category_name	product_category_name_english
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	perfumery
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	art
2	96bd76ec8810374ed1b65e291975717f	esporte_lazer	sports_leisure
3	cef67bcfe19066a932b7673e239eb23d	bebes	baby
4	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	housewares

In [55]:
```
products_df.isnull().sum()
```

Out[55]:
```
product_id                         0
product_category_name            610
product_category_name_english    623
dtype: int64
```

In [56]:
```
products_df[products_df["product_category_name_english"].isnull() == True]["product_ca
```

Out[56]:
```
portateis_cozinha_e_preparadores_de_alimentos    10
pc_gamer                                          3
Name: product_category_name, dtype: int64
```

In [57]:
```
null_1 = products_df[products_df["product_category_name"] == "portateis_cozinha_e_prep
null_2 = products_df[products_df["product_category_name"] == "pc_gamer"]["product_cate
```

In [58]:
```
products_df.loc[null_1.index,"product_category_name_english"] = "kitchen_laptops_and_f
products_df.loc[null_2.index,"product_category_name_english"] = "pc_gamer"
```

In [59]:
```
products_df.isnull().sum()
```

Out[59]:
```
product_id                         0
product_category_name            610
product_category_name_english    610
dtype: int64
```

In [60]:
```
products_df.drop(['product_category_name'], axis = 1, inplace = True)
```

In [61]:
```
# Change product category name column
products_df.columns = products_df.columns.str.replace('product_category_name_english',
```

In [62]:
```
# Fill all empty category names to Category None
products_df['product_category_name'] = products_df['product_category_name'].fillna('ca
```

In [63]:
```
products_df.isnull().sum()
```

Out[63]:
```
product_id               0
product_category_name    0
dtype: int64
```

In [64]:
```
products_df.head()
```

Out[64]:

	product_id	product_category_name
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumery
1	3aa071139cb16b67ca9e5dea641aaa2f	art
2	96bd76ec8810374ed1b65e291975717f	sports_leisure
3	cef67bcfe19066a932b7673e239eb23d	baby
4	9dc1a7de274444849c219cff195d0b71	housewares

In [65]:
```
products_df.shape
```

Out[65]:
```
(32951, 2)
```

Sellers

In [66]:
```
sellers_df.head()
```

Out[66]:

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

In [67]:

```
sellers_df.shape
```

Out[67]:

```
(3095, 4)
```

This dataset has no features that will be of use to this study, this dataset will not get merged into the final one.

Merge datasets for further analysis

In [68]:

```
olist_df = orders_df.merge(orderreviews_df, on = 'order_id')
olist_df = olist_df.merge(orderitems_df, on = 'order_id')
olist_df = olist_df.merge(customers_df, on = 'customer_id')
olist_df = olist_df.merge(products_df, on = 'product_id')
```

In [69]:

```
olist_df.head()
```

Out[69]:

	order_id	customer_id	order_status	order_purcha
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-
1	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	delivered	2017-
2	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	delivered	2017-
3	bfc39df4f36c3693ff3b63fcbea9e90a	53904ddbea91e1e92b2b3f1d09a7af86	delivered	2017-
4	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-

In [70]:

```
olist_df.shape
```

Out[70]:

```
(104782, 15)
```

In [71]:

```
olist_df.describe()
```

Out[71]:

	review_score	order_item_id	price	freight_value
count	104782.000000	104782.000000	104782.000000	104782.000000
mean	4.021502	1.197868	120.516142	19.941800
std	1.394186	0.698074	181.862447	15.688334
min	1.000000	1.000000	0.850000	0.000000
25%	4.000000	1.000000	39.900000	13.080000
50%	5.000000	1.000000	74.990000	16.220000
75%	5.000000	1.000000	134.900000	21.120000
max	5.000000	21.000000	6735.000000	409.680000

In [72]:

```
olist_df.describe(include = 'O')
```

Out[72]:

	order_id	customer_id	order_status	
count	104782	104782	104782	
unique	91182	91182	6	
top	5a3b1c29a49756e75f1ef513383c0c12	be1c4e52bb71e0c54b11a26b8e8d59f2	delivered	aca2eb7d
freq	22	22	102576	

In [73]:

```
olist_df.isnull().sum()
```

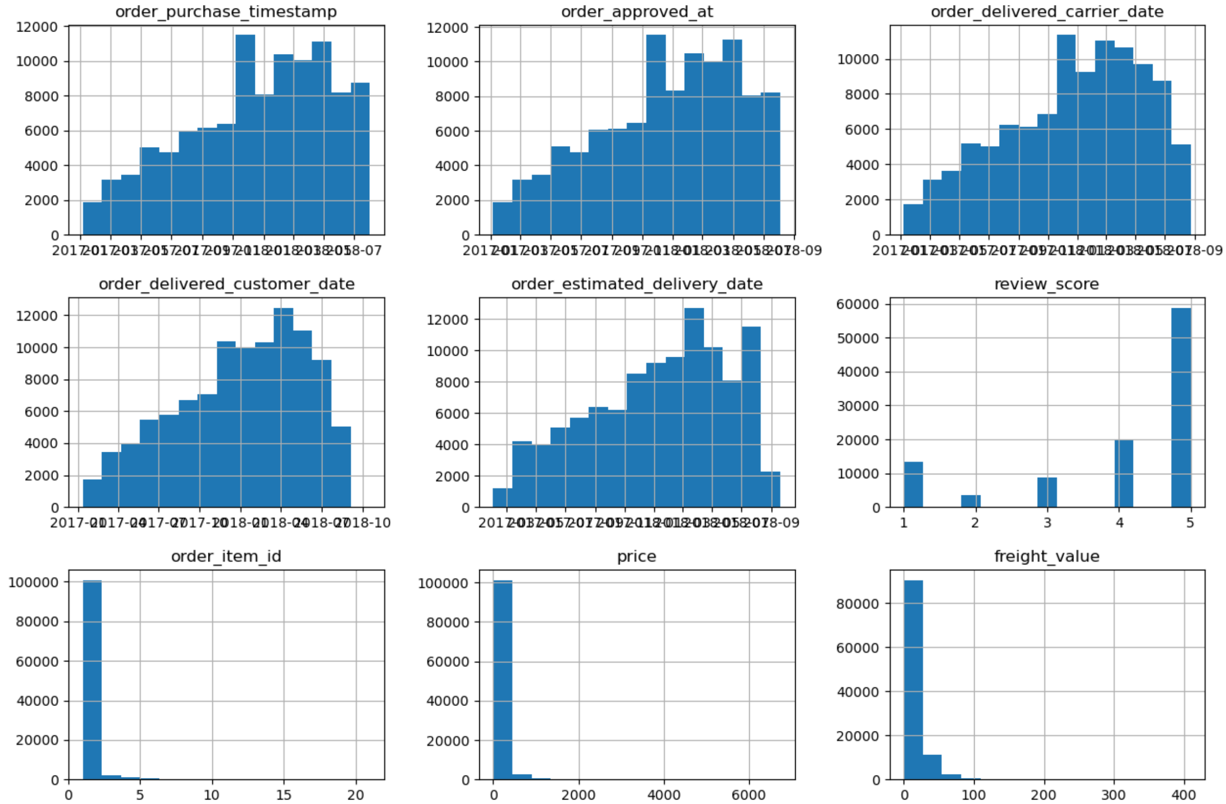
Out[73]:

```
order_id                          0
customer_id                       0
order_status                      0
order_purchase_timestamp          0
order_approved_at                 0
order_delivered_carrier_date   1089
order_delivered_customer_date  2205
order_estimated_delivery_date     0
review_score                      0
order_item_id                     0
product_id                        0
price                             0
freight_value                     0
customer_state                    0
product_category_name             0
dtype: int64
```

In [74]:

```
# Histograms
olist_df.hist(bins = 15, figsize = (15, 10))
```

Out[74]:
```
array([[<AxesSubplot:title={'center':'order_purchase_timestamp'}>,
        <AxesSubplot:title={'center':'order_approved_at'}>,
        <AxesSubplot:title={'center':'order_delivered_carrier_date'}>],
       [<AxesSubplot:title={'center':'order_delivered_customer_date'}>,
        <AxesSubplot:title={'center':'order_estimated_delivery_date'}>,
        <AxesSubplot:title={'center':'review_score'}>],
       [<AxesSubplot:title={'center':'order_item_id'}>,
        <AxesSubplot:title={'center':'price'}>,
        <AxesSubplot:title={'center':'freight_value'}>]], dtype=object)
```



Most of the data is skewed with the dates being left skewed and everything else right skewed

Visualizations

Create visualizations dataset

In [75]:
```
visualizations_df = olist_df.copy()
```

In [76]:
```
# Split out order purchase timestamp into separate parts
visualizations_df['year'] = visualizations_df['order_purchase_timestamp'].dt.year.appl
visualizations_df['month'] = visualizations_df['order_purchase_timestamp'].dt.month.ap
visualizations_df['dow'] = visualizations_df['order_purchase_timestamp'].dt.day_name()
visualizations_df['hour'] = visualizations_df['order_purchase_timestamp'].dt.hour.appl
visualizations_df['year_month'] = visualizations_df['order_purchase_timestamp'].map(la

visualizations_df.head()
```

Out[76]:

	order_id	customer_id	order_status	order_purcha
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-
1	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	delivered	2017-
2	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	delivered	2017-
3	bfc39df4f36c3693ff3b63fcbea9e90a	53904ddbea91e1e92b2b3f1d09a7af86	delivered	2017-
4	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-

Number of Orders per Year

In [77]:

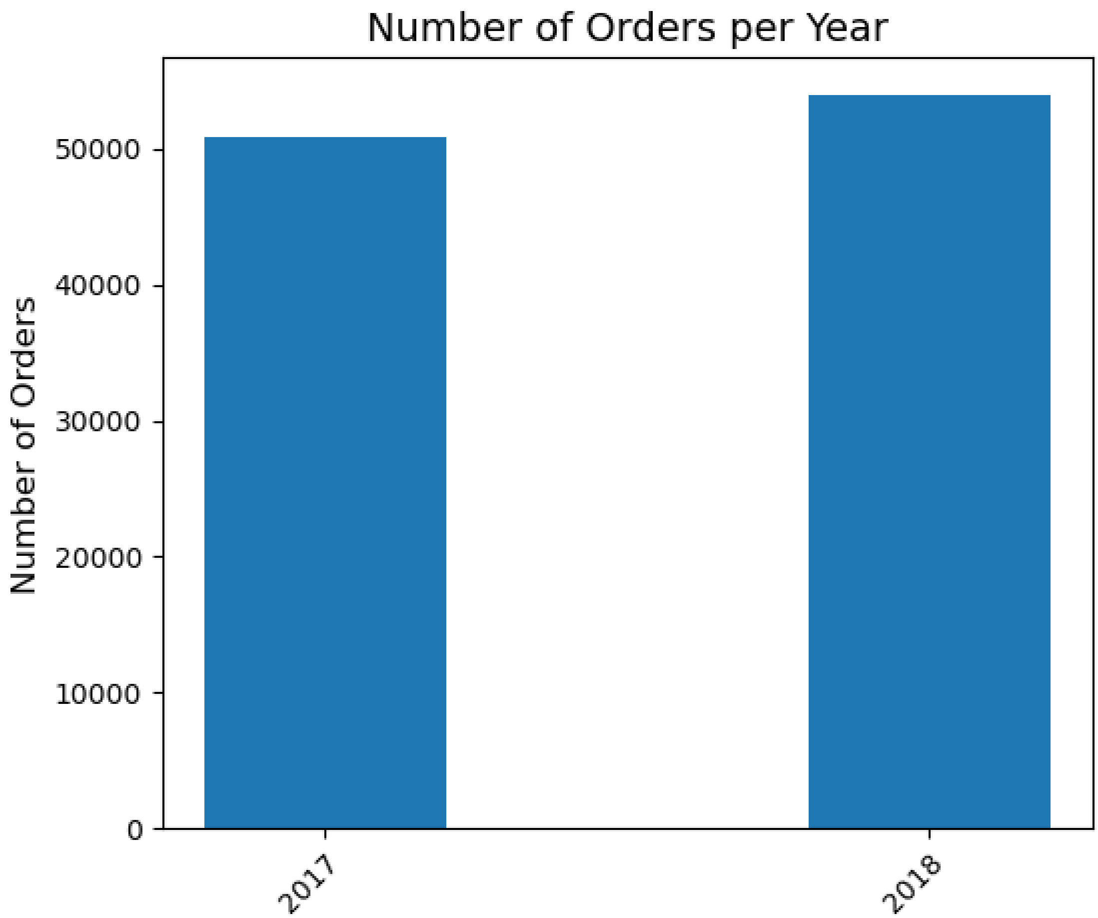
```
per_year = visualizations_df.groupby('year')['order_id'].count().reset_index(name = 'c
per_year
```

Out[77]:

	year	count
0	2017	50791
1	2018	53991

In [78]:

```
fig = plt.figure(figsize = (6, 5))

plt.bar(per_year['year'], per_year['count'], width = 0.4)

plt.xlabel("")
plt.xticks(per_year['year'], per_year['year'], rotation = 45, ha = 'right', rotation_m
plt.ylabel("Number of Orders", fontsize = 12)
plt.title("Number of Orders per Year", fontsize = 14)

plt.show()
```



Looks like 2018 was a better year than 2017 for the number of orders recevied.

Orders per Year per Month

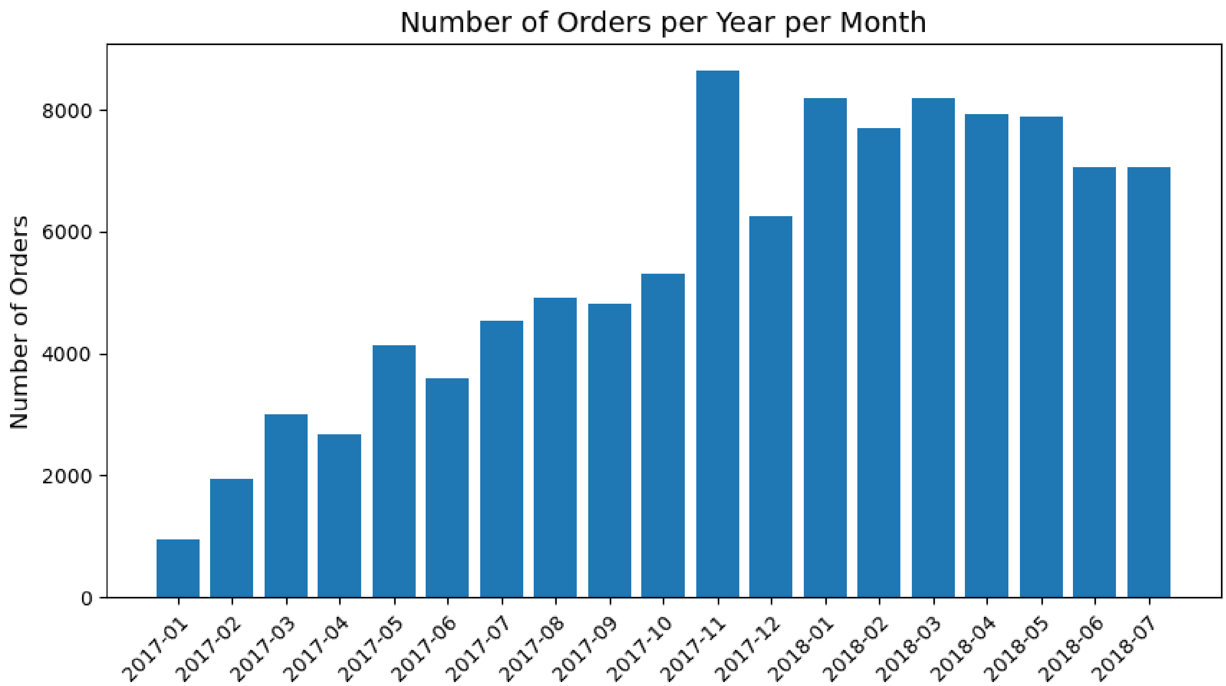
In [79]:
```
per_year_month = visualizations_df.groupby('year_month').size().to_frame("count").rese
per_year_month
```

Out[79]:

	year_month	count
0	2017-01	955
1	2017-02	1951
2	2017-03	2994
3	2017-04	2670
4	2017-05	4142
5	2017-06	3593
6	2017-07	4526
7	2017-08	4905
8	2017-09	4828
9	2017-10	5316
10	2017-11	8647
11	2017-12	6264
12	2018-01	8185
13	2018-02	7699
14	2018-03	8193
15	2018-04	7919
16	2018-05	7887
17	2018-06	7046
18	2018-07	7062

In [80]:

```
fig = plt.figure(figsize = (10, 5))

plt.bar(per_year_month['year_month'], per_year_month['count'])

plt.xlabel("")
plt.xticks(per_year_month['year_month'], per_year_month['year_month'], rotation = 45,
plt.ylabel("Number of Orders", fontsize = 12)
plt.title("Number of Orders per Year per Month", fontsize = 14)

plt.show()
```



November had the nighest number of orders due to holiday purchases and then January and March being good selling months.

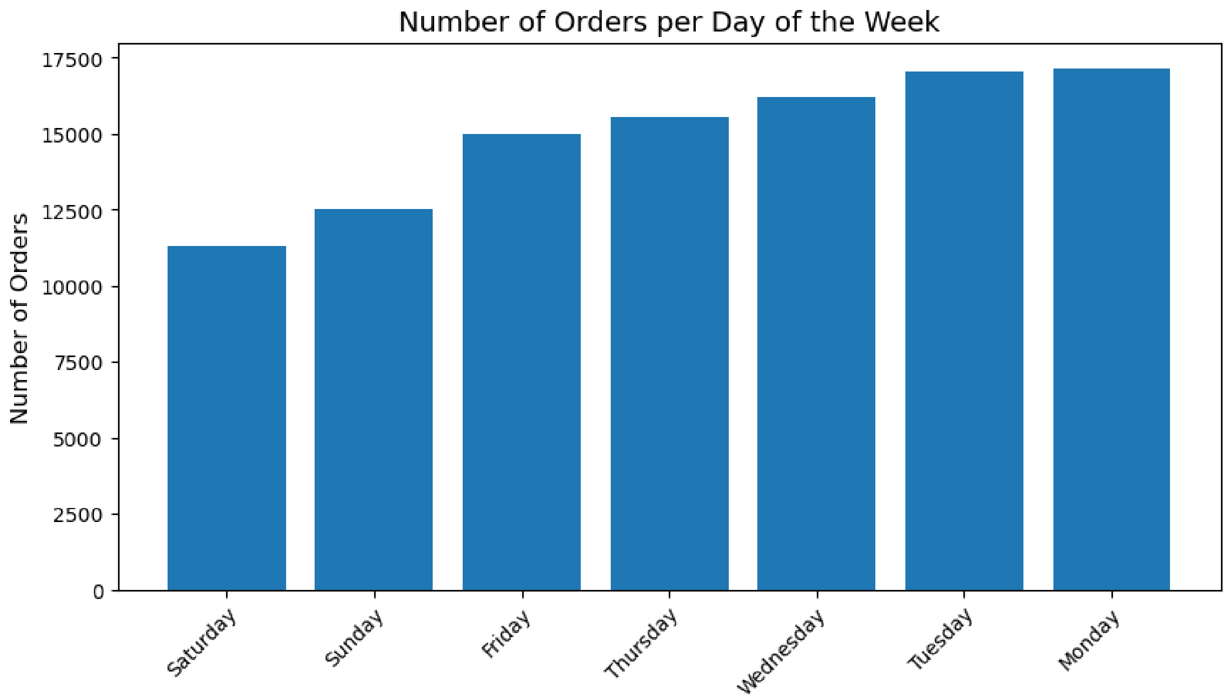
Orders per Day of the Week

In [81]:
```
per_dow = visualizations_df.groupby('dow')['order_id'].count().reset_index(name = 'cou
per_dow
```

Out[81]:

	dow	count
2	Saturday	11295
3	Sunday	12532
0	Friday	14971
4	Thursday	15551
6	Wednesday	16220
5	Tuesday	17063
1	Monday	17150

In [82]:
```
fig = plt.figure(figsize = (10, 5))

plt.bar(per_dow['dow'], per_dow['count'])

plt.xlabel("")
plt.xticks(per_dow['dow'], per_dow['dow'], rotation = 45, ha = 'right', rotation_mode
plt.ylabel("Number of Orders", fontsize = 12)
plt.title("Number of Orders per Day of the Week", fontsize = 14)

plt.show()
```



Purchases are higher during the weekdays vs the weekends and Monday and Tuesday are the highest.

Orders per Hour of the Day

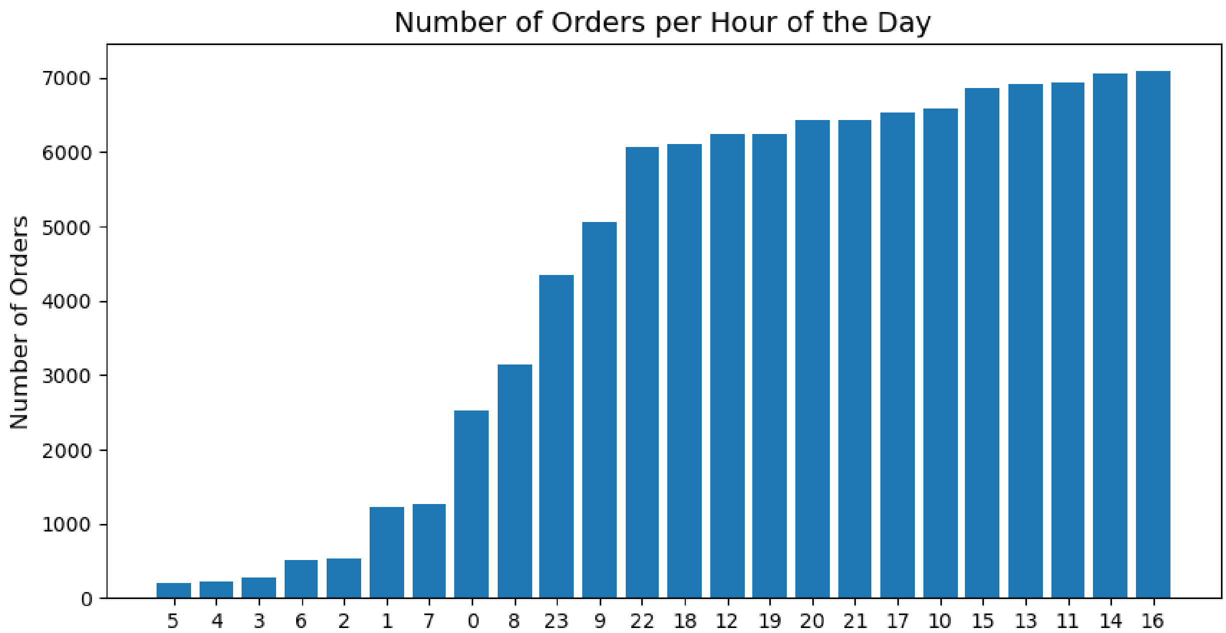
In [83]:
```
per_hour = visualizations_df.groupby('hour')['order_id'].count().reset_index(name = 'c
per_hour
```

Out[83]:

	hour	count
19	5	203
18	4	223
17	3	284
20	6	507
12	2	530
1	1	1217
21	7	1269
0	0	2517
22	8	3147
16	23	4340
23	9	5054
15	22	6065
10	18	6111
4	12	6243
11	19	6246
13	20	6425
14	21	6432
9	17	6534
2	10	6579
7	15	6861
5	13	6908
3	11	6936
6	14	7052
8	16	7099

In [84]:

```
fig = plt.figure(figsize = (10, 5))

plt.bar(per_hour['hour'], per_hour['count'])

plt.xlabel("")
plt.xticks(per_hour['hour'], per_hour['hour'])
plt.ylabel("Number of Orders", fontsize = 12)
plt.title("Number of Orders per Hour of the Day", fontsize = 14)

plt.show()
```



Purchases are mostly made in the evening after 8pm.

Top 10 categories purchased

In [85]:
```
visualizations_df.head()
```

Out[85]:

	order_id	customer_id	order_status	order_purcha
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-
1	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	delivered	2017-
2	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	delivered	2017-
3	bfc39df4f36c3693ff3b63fcbea9e90a	53904ddbea91e1e92b2b3f1d09a7af86	delivered	2017-
4	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-

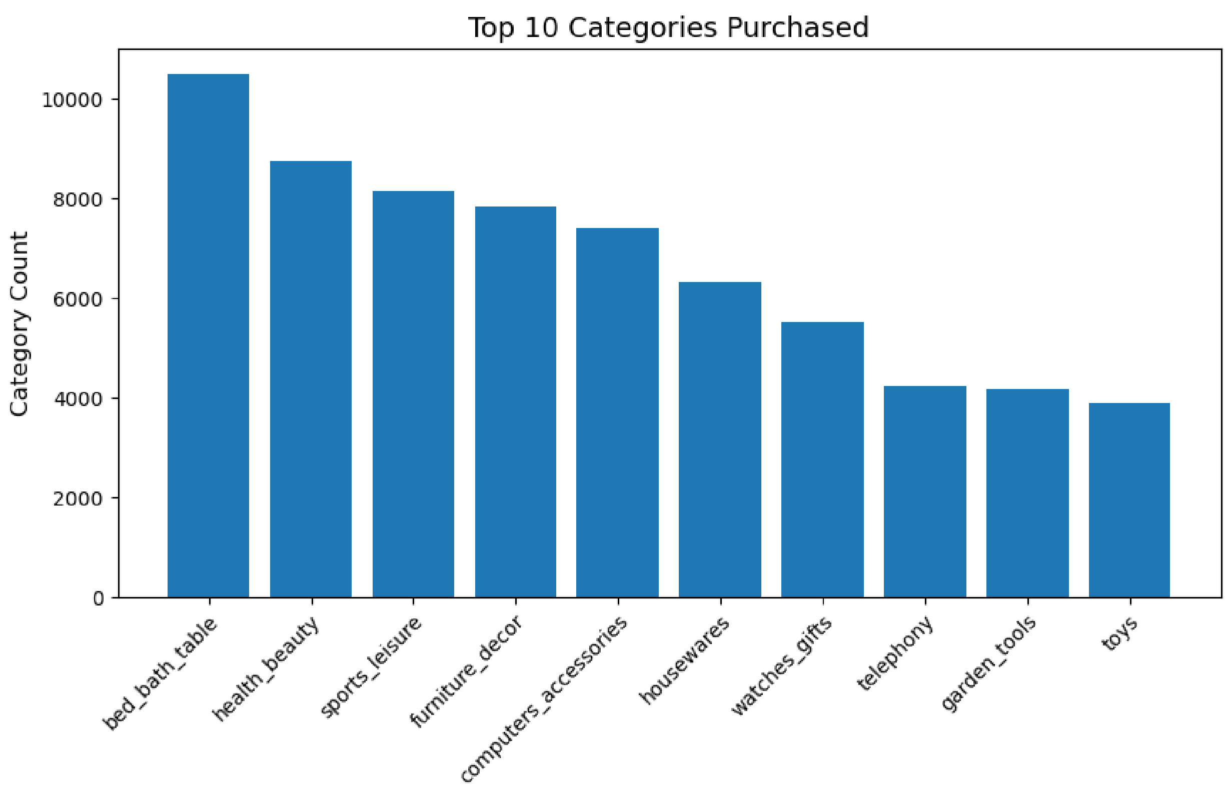
In [86]:
```
top_categories = visualizations_df.groupby('product_category_name')['order_id'].count(
                        .reset_index(name = 'count').nlargest(10, 'count')
top_categories
```

Out[86]:

	product_category_name	count
7	bed_bath_table	10475
44	health_beauty	8751
68	sports_leisure	8146
40	furniture_decor	7827
16	computers_accessories	7410
50	housewares	6307
73	watches_gifts	5529
71	telephony	4231
43	garden_tools	4186
72	toys	3899

In [87]:

```
fig = plt.figure(figsize = (10, 5))

plt.bar(top_categories['product_category_name'], top_categories['count'])

plt.xlabel("")
plt.xticks(top_categories['product_category_name'], top_categories['product_category_r
plt.ylabel("Category Count", fontsize = 12)
plt.title("Top 10 Categories Purchased", fontsize = 14)

plt.show()
```



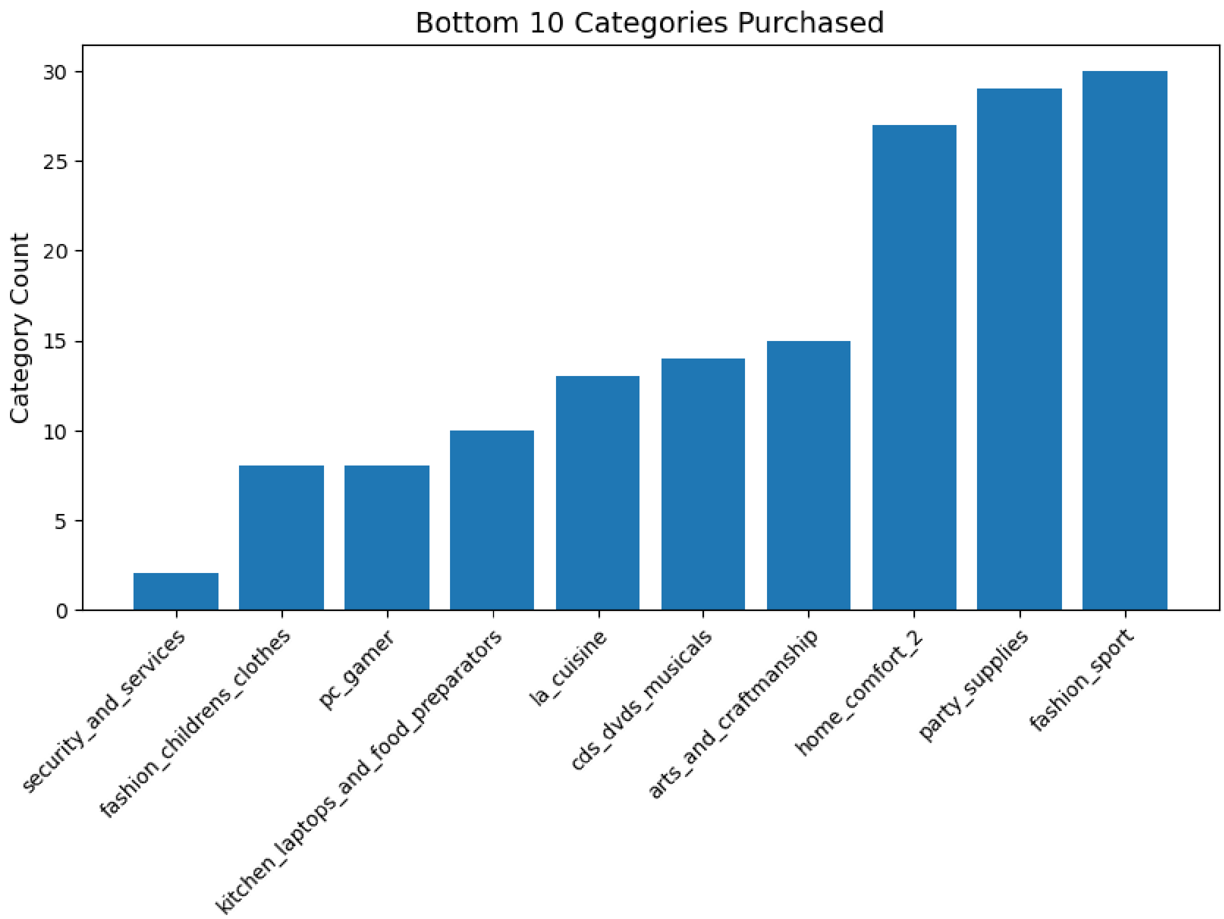
Bottom 10 categories purchased

In [88]:
```
bottom_categories = visualizations_df.groupby('product_category_name')['order_id'].cou
                    .reset_index(name = 'count').nsmallest(10, 'count')
bottom_categories
```

Out[88]:

	product_category_name	count
64	security_and_services	2
30	fashion_childrens_clothes	8
61	pc_gamer	8
53	kitchen_laptops_and_food_preparators	10
54	la_cuisine	13
12	cds_dvds_musicals	14
3	arts_and_craftmanship	15
47	home_comfort_2	27
60	party_supplies	29
33	fashion_sport	30

In [89]:
```
fig = plt.figure(figsize = (10, 5))

plt.bar(bottom_categories['product_category_name'], bottom_categories['count'])

plt.xlabel("")
plt.xticks(bottom_categories['product_category_name'], bottom_categories['product_cate
plt.ylabel("Category Count", fontsize = 12)
plt.title("Bottom 10 Categories Purchased", fontsize = 14)

plt.show()
```



Top 10 categories purchased in amount purchased

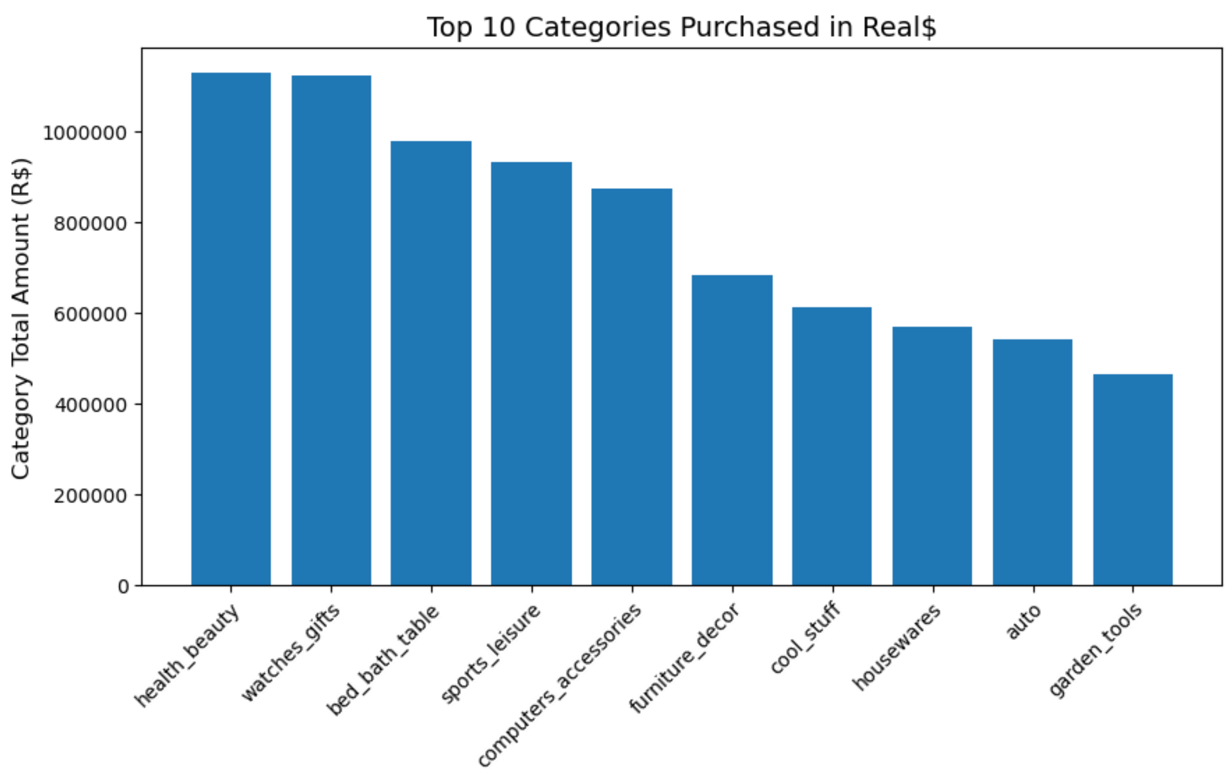
In [90]:
```
top_categories_amt = visualizations_df.groupby(['product_category_name'])["price"].sum
top_categories_amt
```

Out[90]:

	product_category_name	total
44	health_beauty	1128005.33
73	watches_gifts	1121928.18
7	bed_bath_table	978821.48
68	sports_leisure	932107.19
16	computers_accessories	872991.14
40	furniture_decor	682621.60
21	cool_stuff	612096.10
50	housewares	568963.32
5	auto	541243.76
43	garden_tools	465154.67

In [91]:
```
fig, ax = plt.subplots(figsize = (10, 5))

ax.bar(top_categories_amt['product_category_name'], top_categories_amt['total'])
```

```
plt.xlabel("")
plt.xticks(top_categories_amt['product_category_name'], top_categories_amt['product_ca
ax.ticklabel_format(axis="y", useOffset=False, style='plain')
plt.ylabel("Category Total Amount (R$)", fontsize = 12)
plt.title("Top 10 Categories Purchased in Real$", fontsize = 14)

plt.show()
```



The top categories purchased at bed/bath/table and health/beauty.

Bottom 10 categories in amount purchased

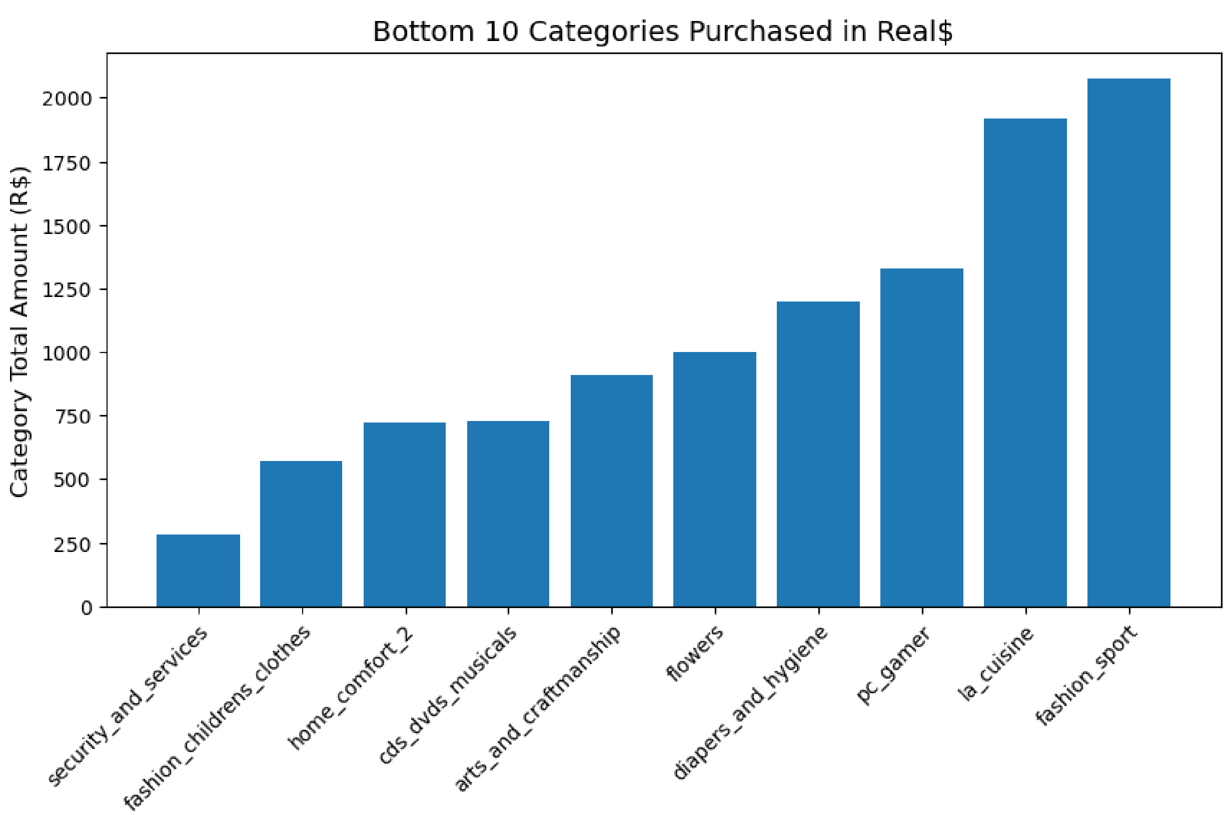
In [92]:
```
bottom_categories_amt = visualizations_df.groupby(['product_category_name'])["price"].
bottom_categories_amt
```

Out[92]:

	product_category_name	total
64	security_and_services	283.29
30	fashion_childrens_clothes	569.85
47	home_comfort_2	721.57
12	cds_dvds_musicals	730.00
3	arts_and_craftmanship	912.25
36	flowers	1000.24
24	diapers_and_hygiene	1200.80
61	pc_gamer	1326.95
54	la_cuisine	1917.99
33	fashion_sport	2074.60

In [93]:

```
fig, ax = plt.subplots(figsize = (10, 5))

ax.bar(bottom_categories_amt['product_category_name'], bottom_categories_amt['total'])

plt.xlabel("")
plt.xticks(bottom_categories_amt['product_category_name'], bottom_categories_amt['proc
ax.ticklabel_format(axis="y", useOffset=False, style='plain')
plt.ylabel("Category Total Amount (R$)", fontsize = 12)
plt.title("Bottom 10 Categories Purchased in Real$", fontsize = 14)

plt.show()
```



Number of orders per State

.

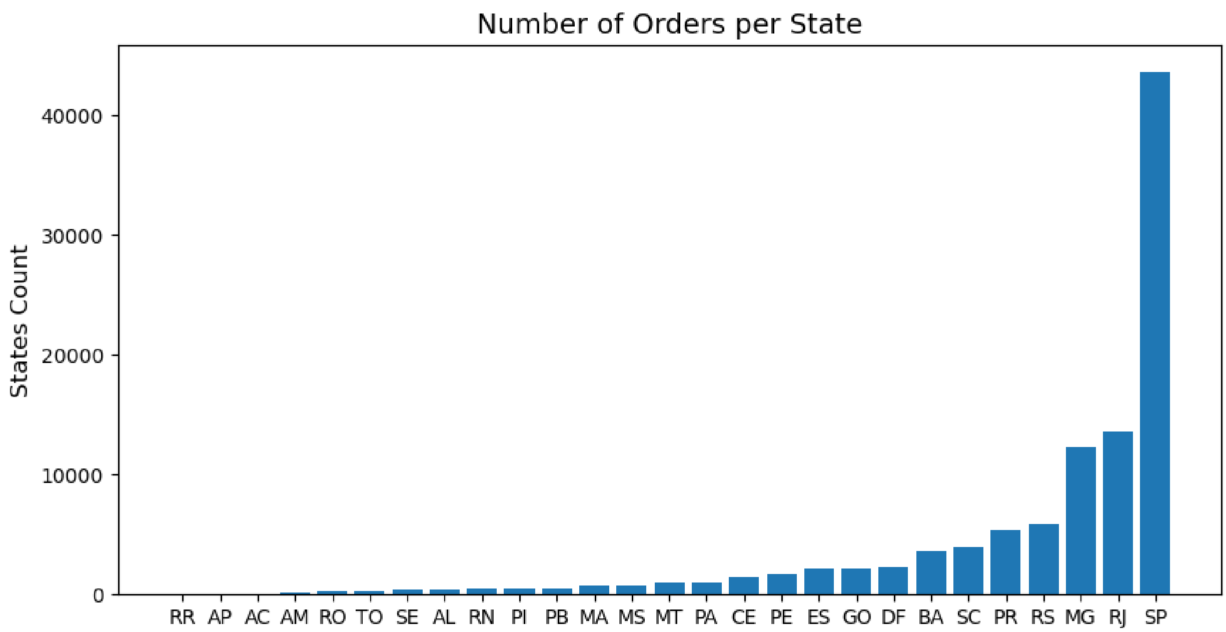
In [94]:
```
states = visualizations_df.groupby('customer_state')['order_id'].count().reset_index(r
states
```

Out[94]:

	customer_state	count
21	RR	49
3	AP	79
0	AC	89
2	AM	160
20	RO	268
26	TO	299
24	SE	357
1	AL	426
19	RN	502
16	PI	516
14	PB	565
9	MA	785
11	MS	789
12	MT	994
13	PA	1011
5	CE	1405
15	PE	1697
7	ES	2119
8	GO	2177
6	DF	2249
4	BA	3590
23	SC	3903
17	PR	5340
22	RS	5876
10	MG	12298
18	RJ	13604
25	SP	43635

In [95]:
```
fig, ax = plt.subplots(figsize = (10, 5))

plt.bar(states['customer_state'], states['count'])

plt.xlabel("")
plt.xticks(states['customer_state'], states['customer_state'])
```

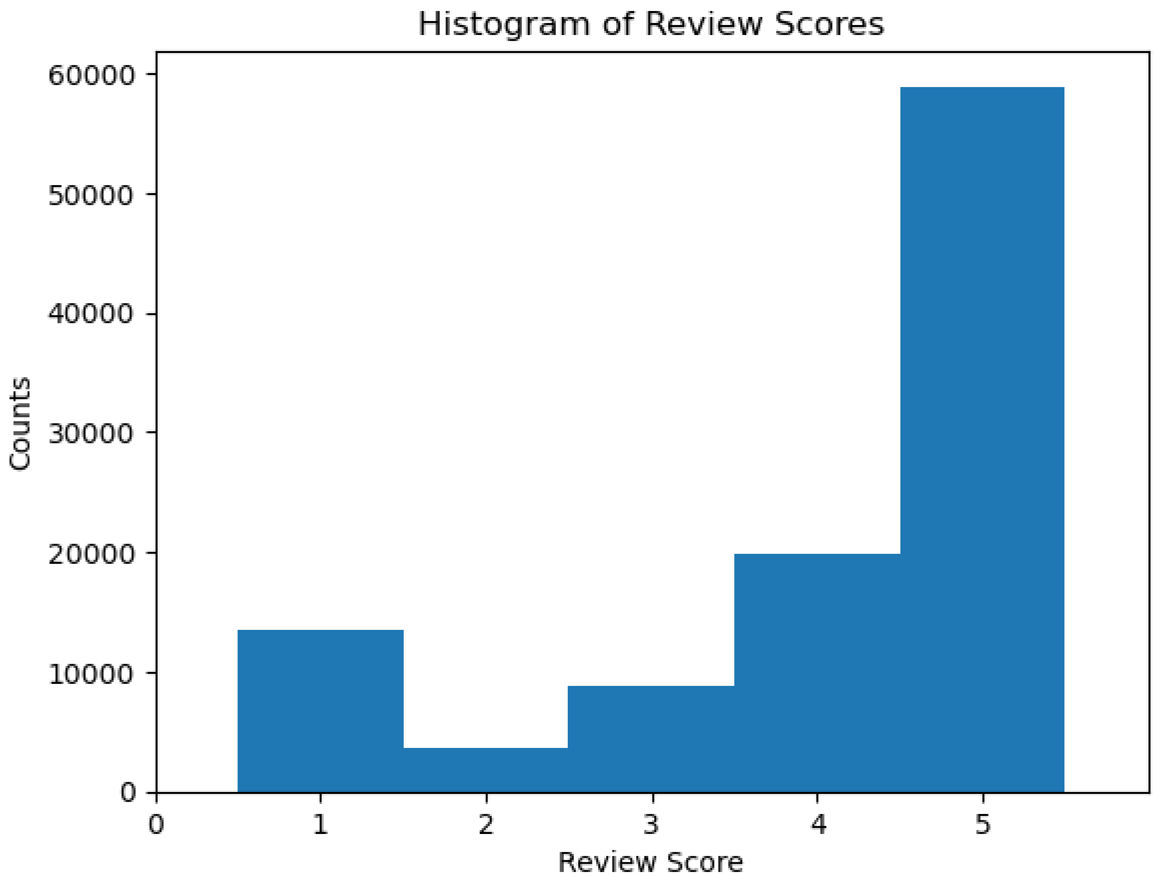
```
plt.ylabel("States Count", fontsize = 12)
plt.title("Number of Orders per State", fontsize = 14)

plt.show()

# Save figure
ax.get_figure().savefig('images/orders_per_state.png',
           bbox_inches = 'tight',
           transparent = True)
```



Sao Paulo has the far most orders of the country

Review Score

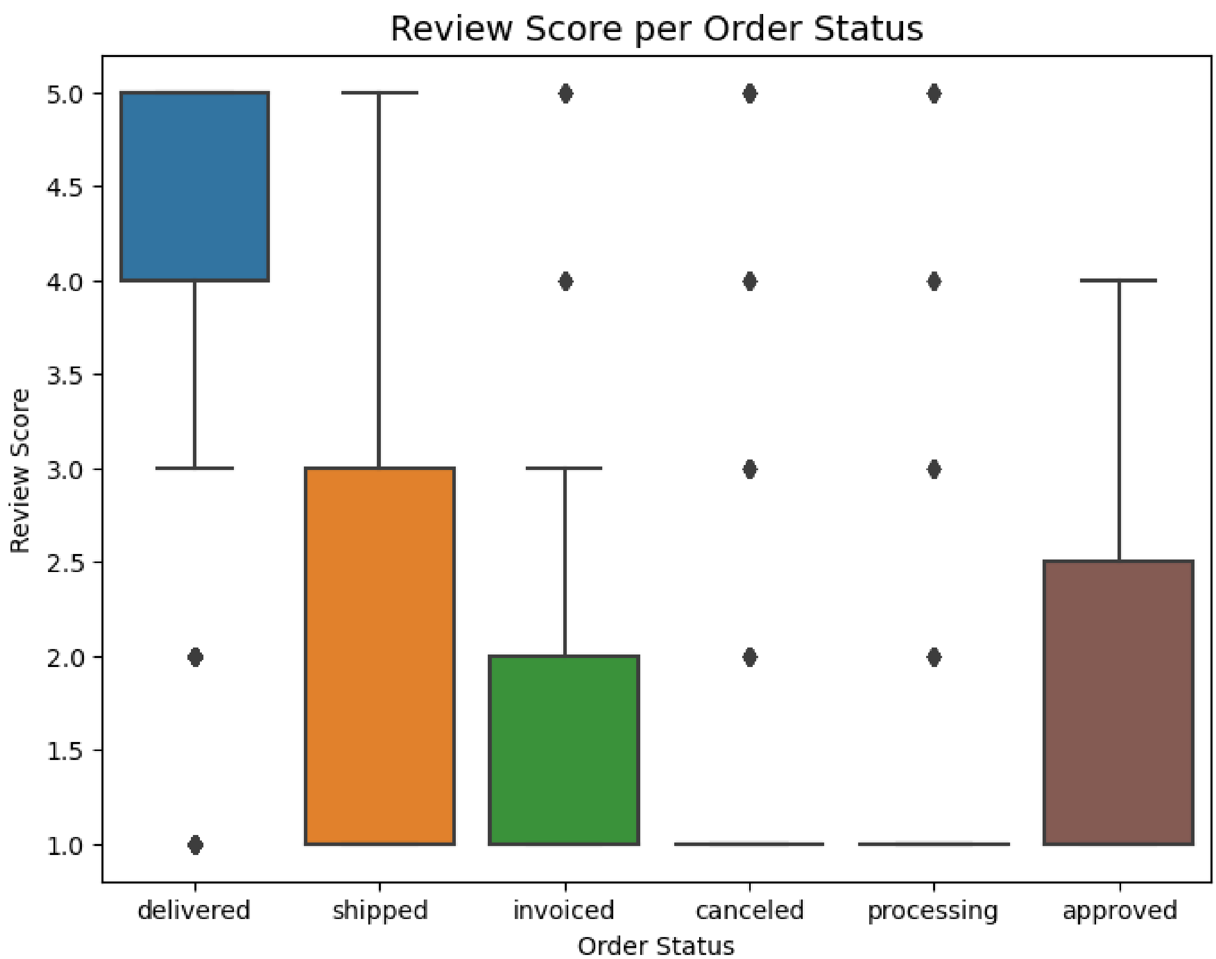
In [96]:
```
# Histogram of review scores
fig, ax = plt.subplots()

bins = np.arange(7) - 0.5
ax.hist(visualizations_df['review_score'], bins = bins)
plt.xticks(range(6))
plt.xlim([0, 6])
plt.title('Histogram of Review Scores')
plt.xlabel('Review Score')
plt.ylabel('Counts')

plt.show()
```



There are more 5 scores than the others.

Review Scores per Order Status

In [97]:
```
fig = plt.subplots(figsize = (8, 6))

sns.boxplot(x = 'order_status', y = 'review_score', data = visualizations_df )
plt.ylabel("Review Score")
plt.xlabel("Order Status")
plt.title("Review Score per Order Status", fontsize = 14)

plt.show()
```



The highest scores are with the delivered purchases

Review Score to Difference between purchase date and delivered date

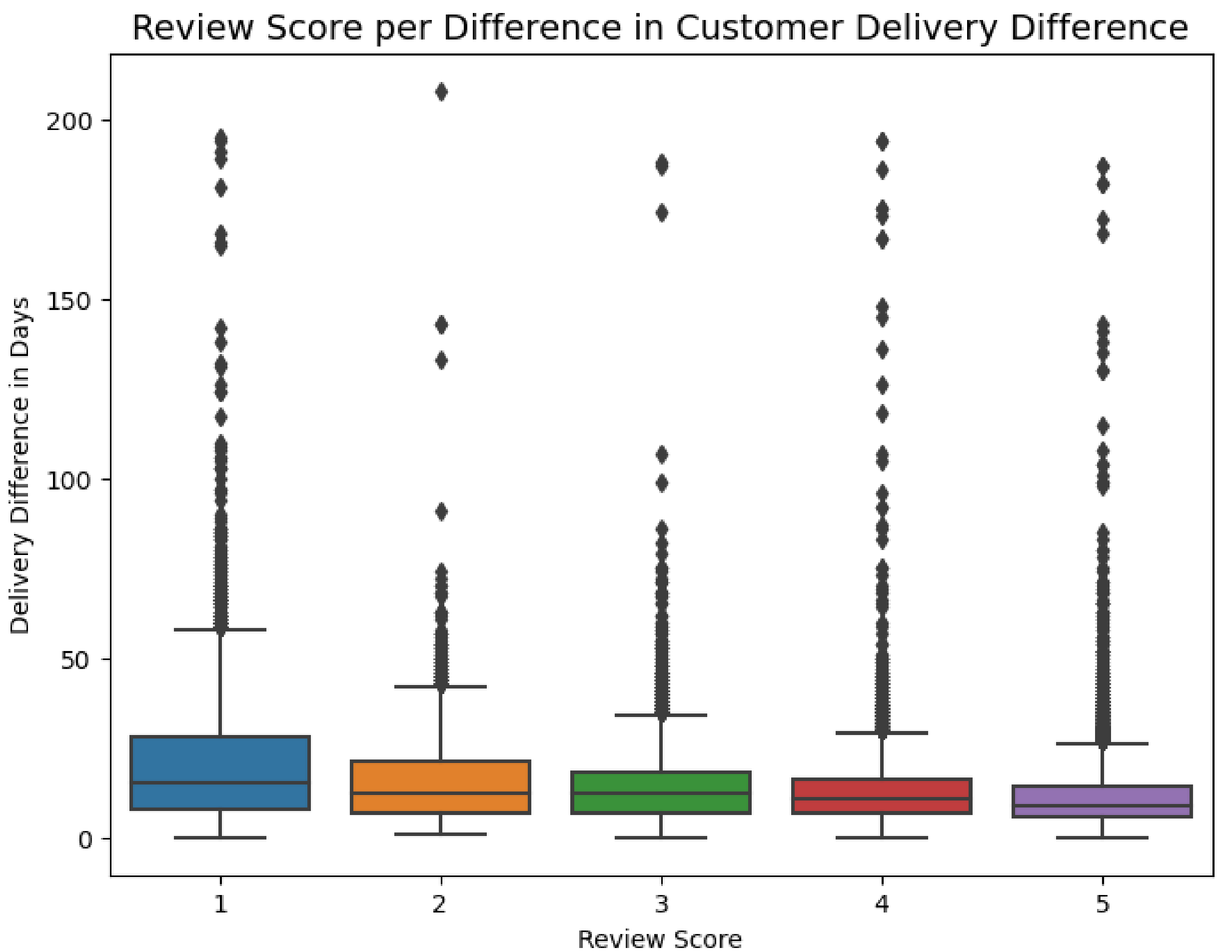
In [98]:
```
# Create new feature for Order Purchase Date - Delivered Customer Date to get differen
visualizations_df['cust_delivery_diff'] = (visualizations_df['order_delivered_customer
visualizations_df.head()
```

Out[98]:

	order_id	customer_id	order_status	order_purcha
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-
1	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	delivered	2017-
2	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	delivered	2017-
3	bfc39df4f36c3693ff3b63fcbea9e90a	53904ddbea91e1e92b2b3f1d09a7af86	delivered	2017-
4	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-

5 rows × 21 columns

In [99]:
```
fig = plt.subplots(figsize = (8, 6))

sns.boxplot(x = 'review_score', y = 'cust_delivery_diff', data = visualizations_df )
plt.xlabel("Review Score")
plt.ylabel("Delivery Difference in Days")
plt.title("Review Score per Difference in Customer Delivery Difference", fontsize = 14

plt.show()
```



The less days between purchase date and delivery date on average has higher scores.

Review Score to Difference between estimated delivery date and delivered date

In [100…
```
# Create new feature for Order Estimated Delivery Date - Delivered Customer Date to ge
visualizations_df['est_delivery_diff'] = (visualizations_df['order_estimated_delivery_
visualizations_df.head()
```

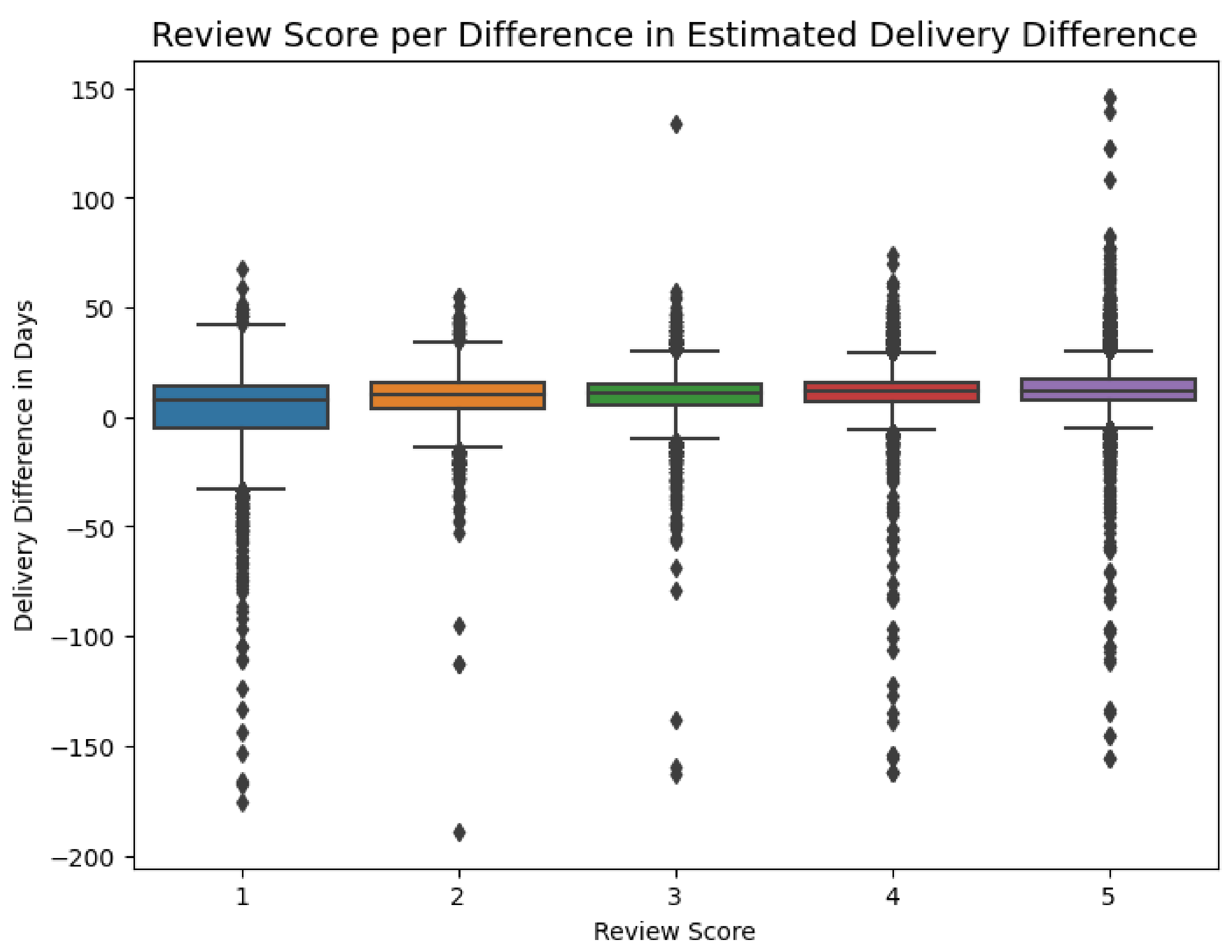
Out[100]:

	order_id	customer_id	order_status	order_purcha
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-
1	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	delivered	2017-
2	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	delivered	2017-
3	bfc39df4f36c3693ff3b63fcbea9e90a	53904ddbea91e1e92b2b3f1d09a7af86	delivered	2017-
4	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-

5 rows × 22 columns

In [101…

```
fig = plt.subplots(figsize = (8, 6))

sns.boxplot(x = 'review_score', y = 'est_delivery_diff', data = visualizations_df )
plt.xlabel("Review Score")
plt.ylabel("Delivery Difference in Days")
plt.title("Review Score per Difference in Estimated Delivery Difference", fontsize = 1

plt.show()
```

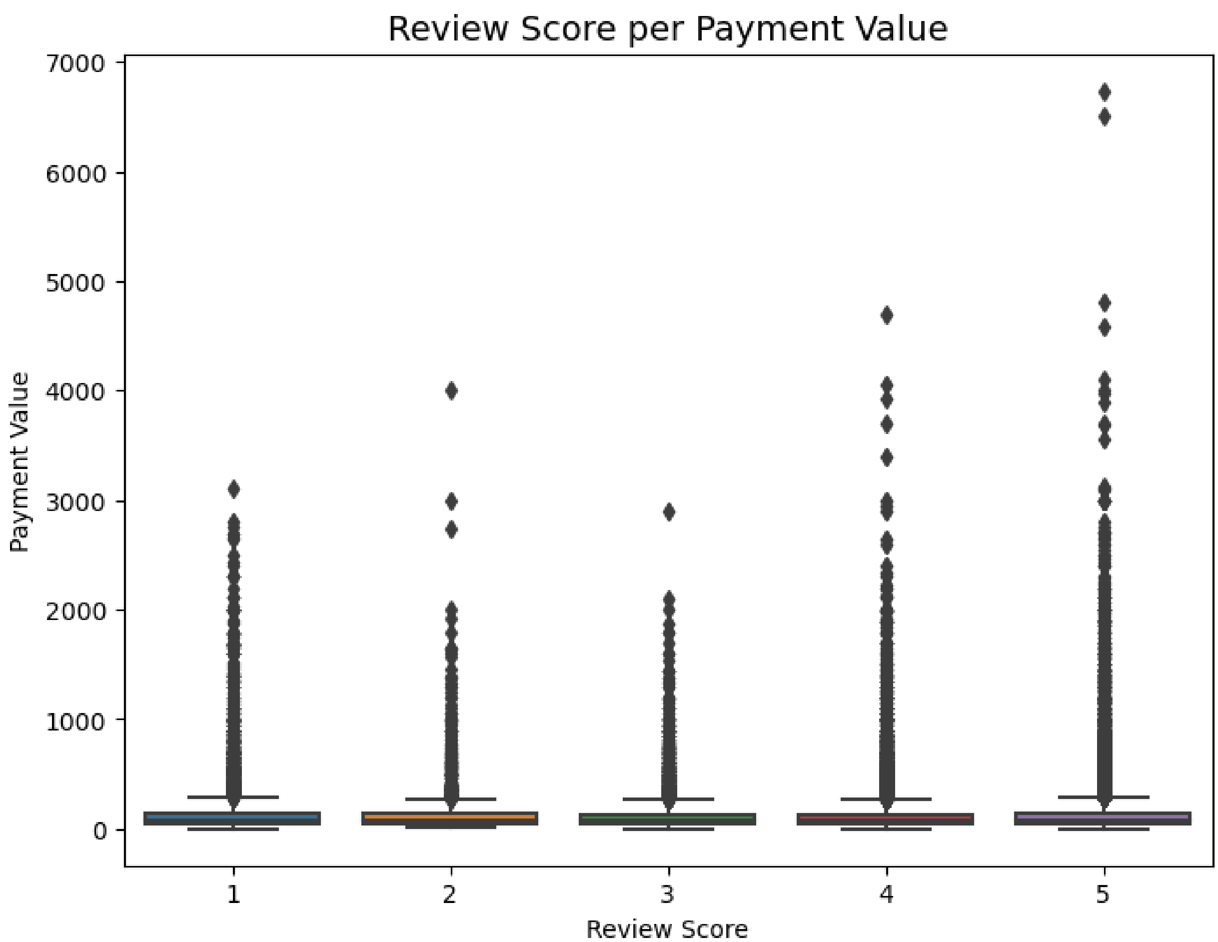


The review scores are about the same if an order was delivered sooner than actual vs later than the actual. There are a lot of outliers on the deliveries that came later.

Review Score to Payment Price

In [102…

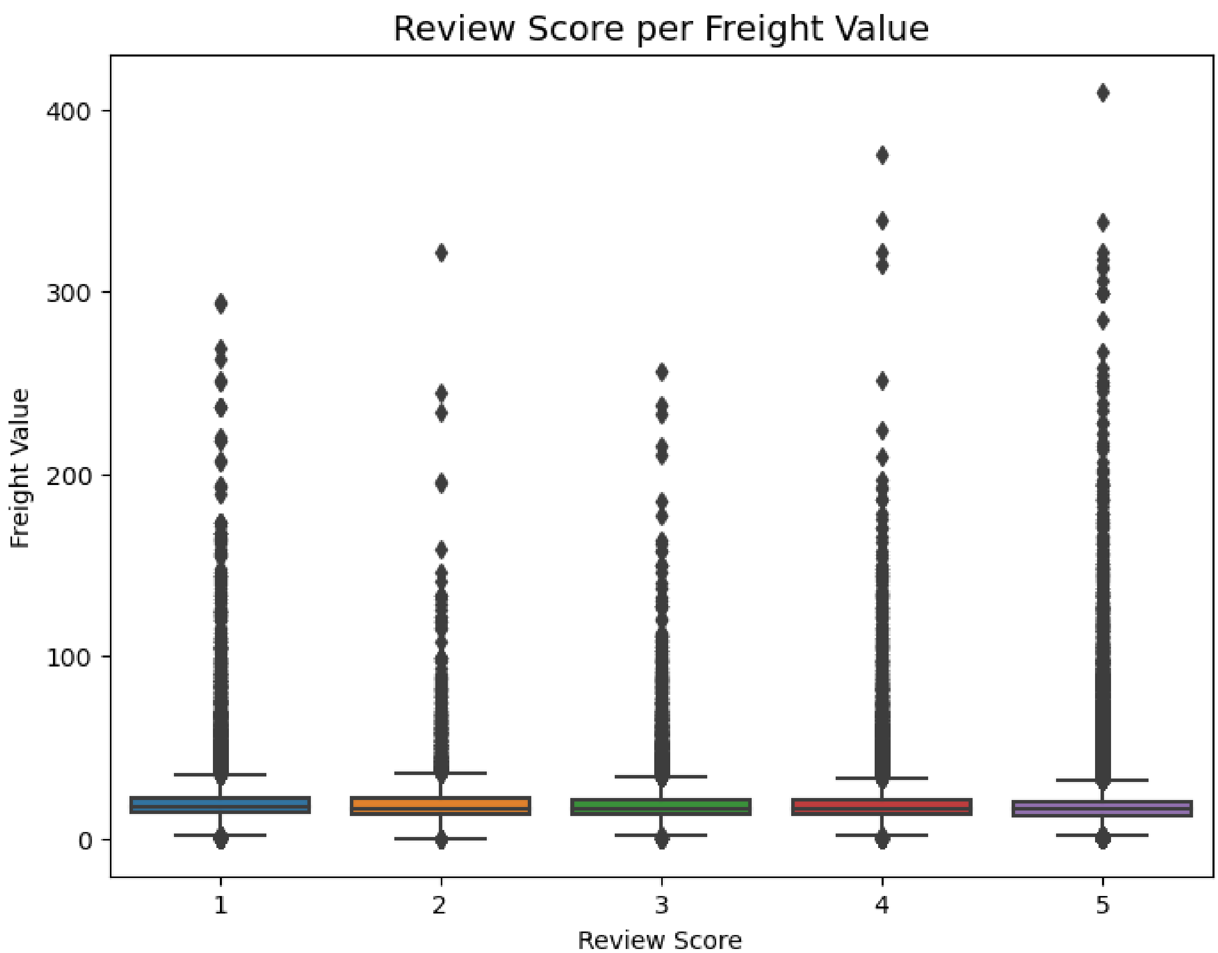
```
fig = plt.subplots(figsize = (8, 6))

sns.boxplot(x = 'review_score', y = 'price', data = visualizations_df )
plt.xlabel("Review Score")
plt.ylabel("Payment Value")
plt.title("Review Score per Payment Value", fontsize = 14)

plt.show()
```



Review Score to Freight Cost

In [103…

```
fig = plt.subplots(figsize = (8, 6))

sns.boxplot(x = 'review_score', y = 'freight_value', data = visualizations_df )
plt.xlabel("Review Score")
plt.ylabel("Freight Value")
plt.title("Review Score per Freight Value", fontsize = 14)

plt.show()
```



These scores all don't seem to matter about freight

Top 10 customers in spending

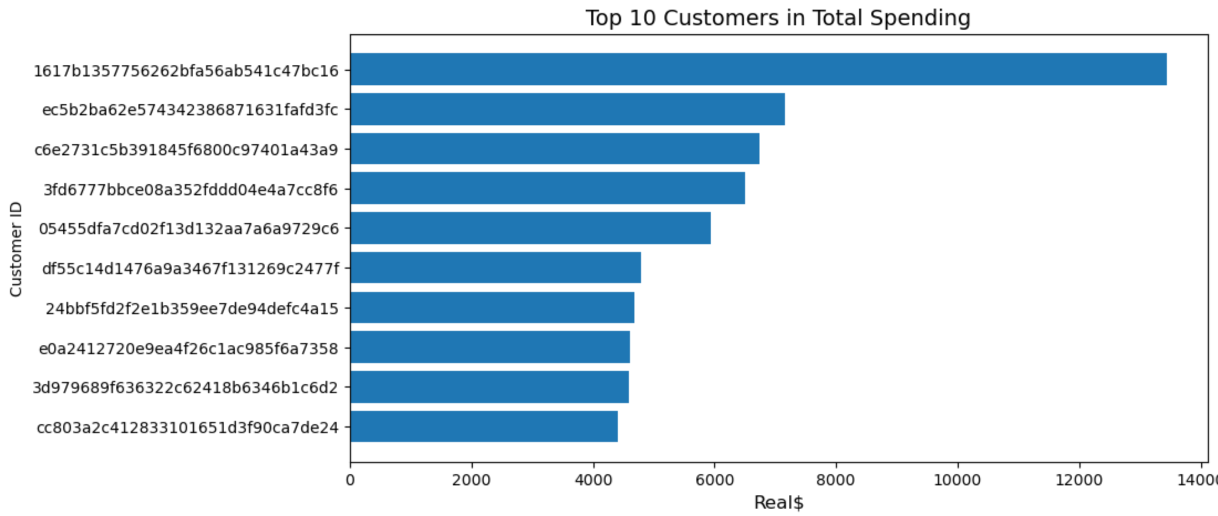
In [104…
```
total_spending = visualizations_df.groupby('customer_id')['price'].sum().reset_index(r
total_spending.head()
```

Out[104]:

	customer_id	total
0	00012a2ce6f8dcda20d059ce98491703	89.80
1	000161a058600d5901f007fab4c27140	54.90
2	0001fd6190edaaf884bcaf3d49edf079	179.99
3	0002414f95344307404f0ace7a26f1d5	149.90
4	000379cdec625522490c315e70c7a9fb	93.00

In [105…
```
top_spending = total_spending.nlargest(10, 'total')

fig, ax = plt.subplots(figsize = (10, 5))

plt.barh(top_spending['customer_id'], top_spending['total'])

plt.ylabel("Customer ID")
plt.yticks(top_spending['customer_id'], top_spending['customer_id'])
plt.xlabel("Real$", fontsize = 12)
```

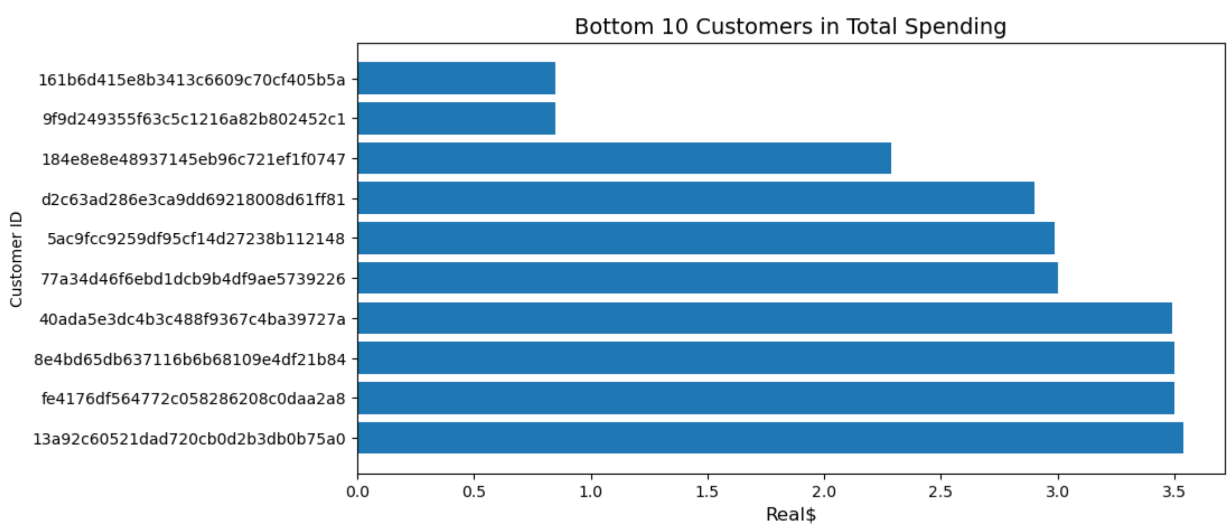
```
plt.title("Top 10 Customers in Total Spending", fontsize = 14)

ax.invert_yaxis()

plt.show()
```



Bottom 20 customers in spending

In [106…

```
bottom_spending = total_spending.nsmallest(10, 'total')

fig, ax = plt.subplots(figsize = (10, 5))

plt.barh(bottom_spending['customer_id'], bottom_spending['total'])

plt.ylabel("Customer ID")
plt.yticks(bottom_spending['customer_id'], bottom_spending['customer_id'])
plt.xlabel("Real$", fontsize = 12)
plt.title("Bottom 10 Customers in Total Spending", fontsize = 14)

ax.invert_yaxis()

plt.show()
```



Top 20 customers in number of orders

In [107…

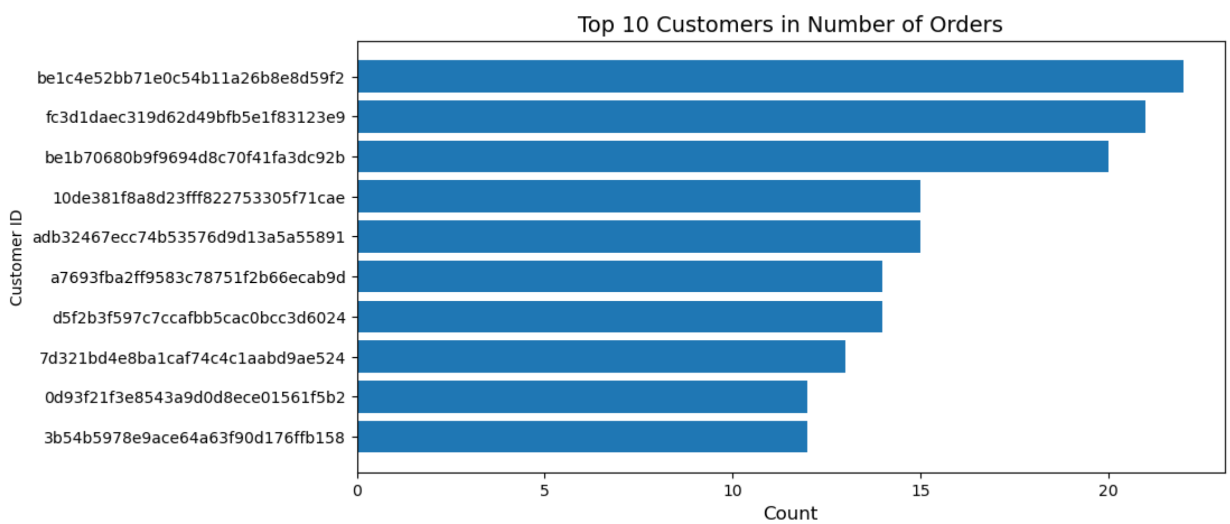
```
number_customers = visualizations_df.groupby('customer_id')['order_id'].count().reset_
number_customers.head()
```

Out[107]:

	customer_id	count
0	00012a2ce6f8dcda20d059ce98491703	1
1	000161a058600d5901f007fab4c27140	1
2	0001fd6190edaaf884bcaf3d49edf079	1
3	0002414f95344307404f0ace7a26f1d5	1
4	000379cdec625522490c315e70c7a9fb	1

In [108…

```
top_customers = number_customers.nlargest(10, 'count')

fig, ax = plt.subplots(figsize = (10, 5))

plt.barh(top_customers['customer_id'], top_customers['count'])

plt.ylabel("Customer ID")
plt.yticks(top_customers['customer_id'], top_customers['customer_id'])
plt.xlabel("Count", fontsize = 12)
plt.title("Top 10 Customers in Number of Orders", fontsize = 14)

ax.invert_yaxis()

plt.show()
```

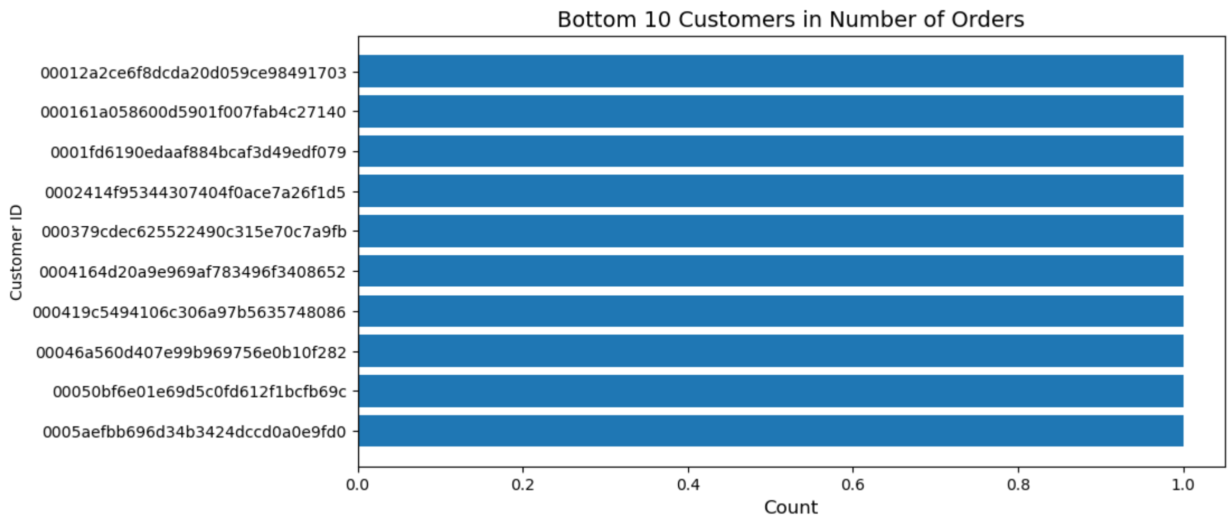


Bottom 20 customers in number of orders

In [109…

```
top_customers = number_customers.nsmallest(10, 'count')

fig, ax = plt.subplots(figsize = (10, 5))

plt.barh(top_customers['customer_id'], top_customers['count'])

plt.ylabel("Customer ID")
plt.yticks(top_customers['customer_id'], top_customers['customer_id'])
plt.xlabel("Count", fontsize = 12)
plt.title("Bottom 10 Customers in Number of Orders", fontsize = 14)
```

```
ax.invert_yaxis()

plt.show()
```



Top 10 customers in number of review scores of 5

In [110…
```
review_score_5 = visualizations_df.loc[visualizations_df.review_score == 5]
review_score_5.head()
```

Out[110]:

	order_id	customer_id	order_status	order_purcha
2	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	delivered	2017-
5	40c5e18f7d112b59b3e5113a59a905b3	67407057a7d5ee17d1cd09523f484d13	delivered	2018-
6	f913d229653fdd809c249ed98ab6b754	e1365d7b227b247b6bc0931771885eaf	delivered	2018-
7	9b85bbefeeacfebc3ff603d20511734f	7f4f07b97783e894fccff9d72e0988b3	delivered	2017-
8	df972aca1fba0a417674857678e2c4bb	322eae54daccdcbee96799ebd3a67830	delivered	2018-

5 rows × 22 columns

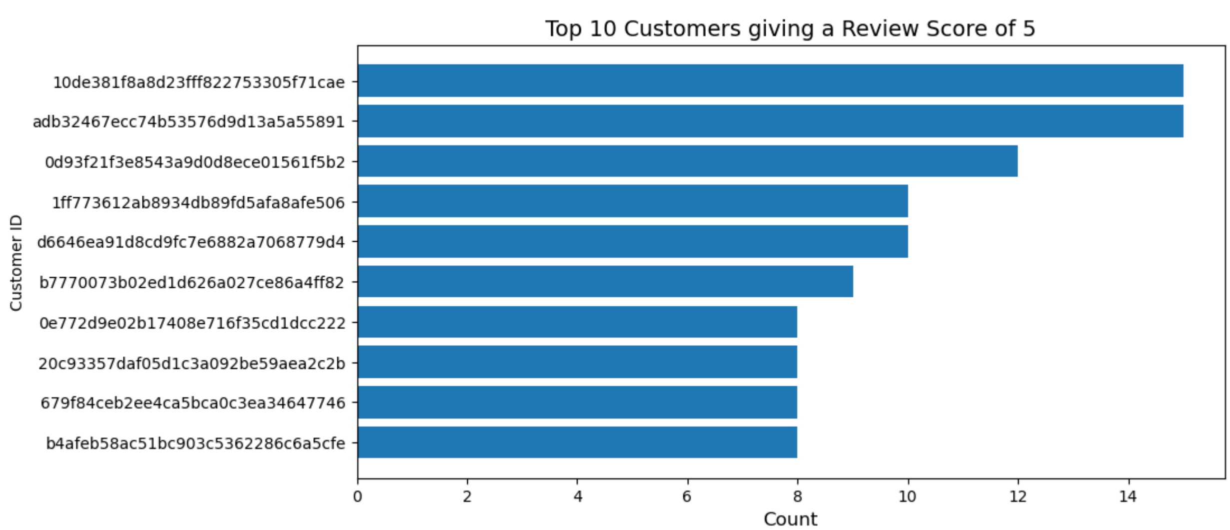
In [111…
```
top_customers_5 = review_score_5.groupby('customer_id')['review_score'].count().reset_
top_customers_5
```

Out[111]:

	customer_id	count
3469	10de381f8a8d23fff822753305f71cae	15
35785	adb32467ecc74b53576d9d13a5a55891	15
2778	0d93f21f3e8543a9d0d8ece01561f5b2	12
6571	1ff773612ab8934db89fd5afa8afe506	10
44310	d6646ea91d8cd9fc7e6882a7068779d4	10
37805	b7770073b02ed1d626a027ce86a4ff82	9
2960	0e772d9e02b17408e716f35cd1dcc222	8
6734	20c93357daf05d1c3a092be59aea2c2b	8
21239	679f84ceb2ee4ca5bca0c3ea34647746	8
37222	b4afeb58ac51bc903c5362286c6a5cfe	8

In [112…

```
fig, ax = plt.subplots(figsize = (10, 5))

plt.barh(top_customers_5['customer_id'], top_customers_5['count'])

plt.ylabel("Customer ID")
plt.yticks(top_customers_5['customer_id'], top_customers_5['customer_id'])
plt.xlabel("Count", fontsize = 12)
plt.title("Top 10 Customers giving a Review Score of 5", fontsize = 14)

ax.invert_yaxis()

plt.show()
```



In [113…

```
top_10 = visualizations_df['customer_id'].isin(top_customers_5['customer_id'])
top_10_customers_df = visualizations_df[top_10]
top_10_customers_df.drop_duplicates(subset = ['order_id', 'customer_id'], keep = 'firs
top_10_customers_df
```

Out[113]:

	order_id	customer_id	order_status	order_p
4201	30bdf3d824d824610a49887486debcaf	d6646ea91d8cd9fc7e6882a7068779d4	delivered	
4350	3cb5915708fd5b47246994508f858ffd	679f84ceb2ee4ca5bca0c3ea34647746	delivered	
13293	acbe07f22f29ad7e5a78f30008cc6ec7	b4afeb58ac51bc903c5362286c6a5cfe	delivered	
49414	428a2f660dc84138d969ccd69a0ab6d5	10de381f8a8d23fff822753305f71cae	delivered	
54499	df56136b8031ecd28e200bb18e6ddb2e	b7770073b02ed1d626a027ce86a4ff82	delivered	
67152	2c2a19b5703863c908512d135aa6accc	0d93f21f3e8543a9d0d8ece01561f5b2	delivered	
67939	e8fa22c3673b1dd17ea315021b1f0f61	1ff773612ab8934db89fd5afa8afe506	delivered	
77657	9a2b443dc8e6673e4fc330b3ea033569	20c93357daf05d1c3a092be59aea2c2b	delivered	
90355	c27cd942c2a926d25153090afa106ceb	0e772d9e02b17408e716f35cd1dcc222	delivered	
96279	9ef13efd6949e4573a18964dd1bbe7f5	adb32467ecc74b53576d9d13a5a55891	delivered	

10 rows × 22 columns

In [114…

```
# get differences in days for these customers to see if there is a trend
top_10_customers_df['cust_delivery_diff'] = (top_10_customers_df['order_delivered_cust
                                             top_10_customers_df['order_purchase_times
top_10_customers_df['est_delivery_diff'] = (top_10_customers_df['order_estimated_deliv
                                            top_10_customers_df['order_delivered_custo
```

In [115…

```
top_10_customers_df[['customer_id', 'review_score', 'order_status', 'price', 'freight_
```

Out[115]:

	customer_id	review_score	order_status	price	freight_value	product_
4201	d6646ea91d8cd9fc7e6882a7068779d4	5	delivered	81.99	14.51	comp
4350	679f84ceb2ee4ca5bca0c3ea34647746	5	delivered	59.90	17.67	
13293	b4afeb58ac51bc903c5362286c6a5cfe	5	delivered	19.30	11.73	
49414	10de381f8a8d23fff822753305f71cae	5	delivered	65.49	16.22	
54499	b7770073b02ed1d626a027ce86a4ff82	5	delivered	66.90	31.65	
67152	0d93f21f3e8543a9d0d8ece01561f5b2	5	delivered	20.70	16.11	
67939	1ff773612ab8934db89fd5afa8afe506	5	delivered	284.99	16.87	
77657	20c93357daf05d1c3a092be59aea2c2b	5	delivered	20.50	16.91	
90355	0e772d9e02b17408e716f35cd1dcc222	5	delivered	36.99	11.85	
96279	adb32467ecc74b53576d9d13a5a55891	5	delivered	51.00	1.20	

Top 10 customers in number of review scores of 1

In [116…

```
review_score_1 = visualizations_df.loc[visualizations_df.review_score == 1]
review_score_1.head()
```

Out[116]:

	order_id	customer_id	order_status	order_purch
10	6552ae78f1de31bcde1fc2cfcab0d25d	ccb212cf6faf1356d9b5509259de0940	delivered	2018
26	fc74153e0ac39bb68c8f8f9e4758f001	787c8dad81798b72c5ae7d0ed526192e	delivered	2018
46	29f95ab000e30a2a4dbeedb73c7357f2	a13f758577dd5c5b1b1897f294c2da52	delivered	2017
79	3a53d5a9a0c58d291ff3ae407b6df5fd	5612aa60cdbbd8e9d89ae0c409080375	shipped	2018
87	177777137ce0af9e9cd2cff572728cca	fedcdc6c89d60699c967422066834f65	delivered	2018

5 rows × 22 columns

In [117…

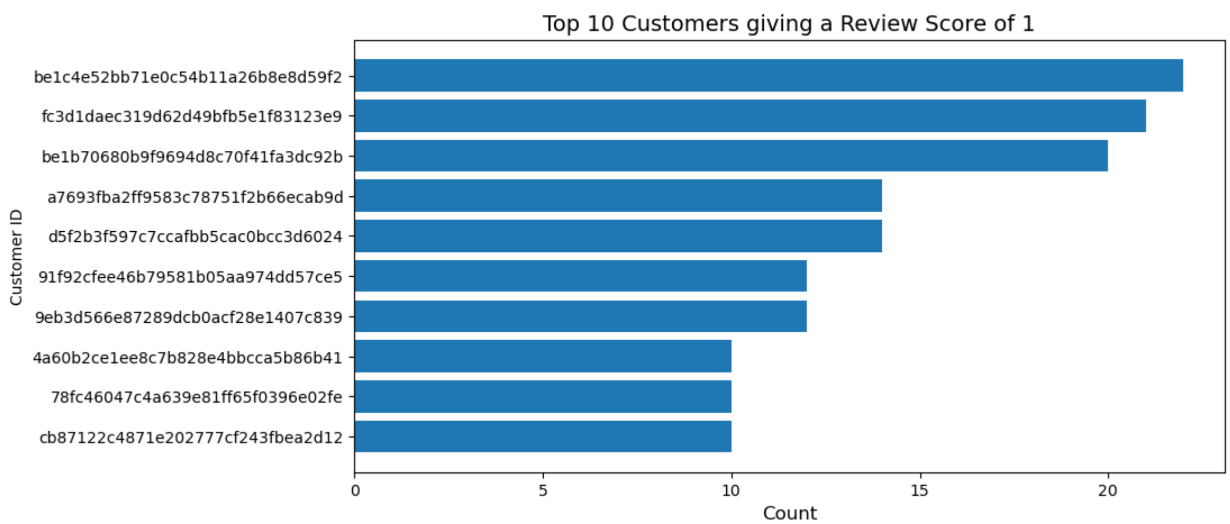
```
top_customers_1 = review_score_1.groupby('customer_id')['review_score'].count().reset_
top_customers_1.head()
```

Out[117]:

	customer_id	count
7659	be1c4e52bb71e0c54b11a26b8e8d59f2	22
10115	fc3d1daec319d62d49bfb5e1f83123e9	21
7658	be1b70680b9f9694d8c70f41fa3dc92b	20
6713	a7693fba2ff9583c78751f2b66ecab9d	14
8578	d5f2b3f597c7ccafbb5cac0bcc3d6024	14

In [118…

```
fig, ax = plt.subplots(figsize = (10, 5))

plt.barh(top_customers_1['customer_id'], top_customers_1['count'])

plt.ylabel("Customer ID")
plt.yticks(top_customers_1['customer_id'], top_customers_1['customer_id'])
plt.xlabel("Count", fontsize = 12)
plt.title("Top 10 Customers giving a Review Score of 1", fontsize = 14)

ax.invert_yaxis()

plt.show()
```



In [119…

```
bottom_10 = visualizations_df['customer_id'].isin(top_customers_1['customer_id'])
bottom_10_customers_df = visualizations_df[bottom_10]
bottom_10_customers_df.drop_duplicates(subset = ['order_id', 'customer_id'], keep = 'f
bottom_10_customers_df
```

Out[119]:

	order_id	customer_id	order_status	order_p
6639	3a213fcdfe7d98be74ea0dc05a8b31ae	91f92cfee46b79581b05aa974dd57ce5	delivered	
19725	73c8ab38f07dc94389065f7eba4f297a	d5f2b3f597c7ccafbb5cac0bcc3d6024	delivered	
24699	f80549a97eb203e1566e026ab66f045b	4a60b2ce1ee8c7b828e4bbcca5b86b41	delivered	
25887	5a3b1c29a49756e75f1ef513383c0c12	be1c4e52bb71e0c54b11a26b8e8d59f2	delivered	
37089	f60ce04ff8060152c83c7c97e246d6a8	78fc46047c4a639e81ff65f0396e02fe	delivered	
46421	1b15974a0141d54e36626dca3fdc731a	be1b70680b9f9694d8c70f41fa3dc92b	delivered	
68674	9f5054bd9a3c71702aa0917a7da29193	cb87122c4871e202777cf243fbea2d12	delivered	
77621	9bdc4d4c71aa1de4606060929dee888c	a7693fba2ff9583c78751f2b66ecab9d	delivered	
102470	8272b63d03f5f79c56e9e4120aec44ef	fc3d1daec319d62d49bfb5e1f83123e9	delivered	
102724	af822dacd6f5cff7376413c03a388bb7	9eb3d566e87289dcb0acf28e1407c839	delivered	

10 rows × 22 columns

In [120…

```
# get differences in days for these customers to see if there is a trend
bottom_10_customers_df['cust_delivery_diff'] = (bottom_10_customers_df['order_delivere
bottom_10_customers_df['est_delivery_diff'] = (bottom_10_customers_df['order_estimatec
```

In [121…

```
bottom_10_customers_df[['customer_id', 'review_score', 'order_status', 'price', 'freig
```

Out[121]:

	customer_id	review_score	order_status	price	freight_value	produc
6639	91f92cfee46b79581b05aa974dd57ce5	1	delivered	108.00	15.52	
19725	d5f2b3f597c7ccafbb5cac0bcc3d6024	1	delivered	59.00	13.43	
24699	4a60b2ce1ee8c7b828e4bbcca5b86b41	1	delivered	137.90	38.81	com
25887	be1c4e52bb71e0c54b11a26b8e8d59f2	1	delivered	49.99	7.10	
37089	78fc46047c4a639e81ff65f0396e02fe	1	delivered	109.97	34.04	furi
46421	be1b70680b9f9694d8c70f41fa3dc92b	1	delivered	100.00	10.12	com
68674	cb87122c4871e202777cf243fbea2d12	1	delivered	149.91	0.14	com
77621	a7693fba2ff9583c78751f2b66ecab9d	1	delivered	29.99	7.78	
102470	fc3d1daec319d62d49bfb5e1f83123e9	1	delivered	1.20	7.89	
102724	9eb3d566e87289dcb0acf28e1407c839	1	delivered	5.31	15.23	

Final Dataset for Model

In [122…

```
olist_df.head()
```

Out[122]:

	order_id	customer_id	order_status	order_purcha
0	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	2017-
1	128e10d95713541c87cd1a2e48201934	a20e8105f23924cd00833fd87daa0831	delivered	2017-
2	0e7e841ddf8f8f2de2bad69267ecfbcf	26c7ac168e1433912a51b924fbd34d34	delivered	2017-
3	bfc39df4f36c3693ff3b63fcbea9e90a	53904ddbea91e1e92b2b3f1d09a7af86	delivered	2017-
4	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	2018-

In [123…

```
olist_df_model = olist_df[['customer_id', 'order_purchase_timestamp', 'order_id', 'pri
olist_df_model
```

Out[123]:

	customer_id	order_purchase_timestamp	or
0	9ef432eb6251297304e76186b10a928d	2017-10-02 10:56:33	e481f51cbdc54678b7cc49136f2
1	a20e8105f23924cd00833fd87daa0831	2017-08-15 18:29:31	128e10d95713541c87cd1a2e482
2	26c7ac168e1433912a51b924fbd34d34	2017-08-02 18:24:47	0e7e841ddf8f8f2de2bad69267
3	53904ddbea91e1e92b2b3f1d09a7af86	2017-10-23 23:26:46	bfc39df4f36c3693ff3b63fcbea
4	b0830fb4747a6c6d20dea0b8c802d7ef	2018-07-24 20:41:37	53cdb2fc8bc7dce0b6741e21502
...	...	...	
104777	609b9fb8cad4fe0c7b376f77c8ab76ad	2017-08-10 21:21:07	e8fd20068b9f7e6ec07068bb753
104778	609b9fb8cad4fe0c7b376f77c8ab76ad	2017-08-10 21:21:07	e8fd20068b9f7e6ec07068bb753
104779	a2f7428f0cafbc8e59f20e1444b67315	2017-12-20 09:52:41	cfa78b997e329a5295b4ee6972c
104780	39bd1228ee8140590ac3aca26f2dfe00	2017-03-09 09:54:05	9c5dedf39a927c1b2549525ed64
104781	edb027a75a1449115f6b43211ae02a24	2018-03-08 20:57:30	66dea50a8b16d9b4dee7af250b4

104782 rows × 4 columns

In [124…

```
olist_df_model.describe(datetime_is_numeric = True)
```

Out[124]:

	order_purchase_timestamp	price
count	104782	104782.000000
mean	2017-12-17 22:55:15.929863680	120.516142
min	2017-01-05 11:56:06	0.850000
25%	2017-09-04 09:26:48.750000128	39.900000
50%	2018-01-08 11:16:11	74.990000
75%	2018-04-15 19:55:43.750000128	134.900000
max	2018-07-31 23:54:20	6735.000000
std	NaN	181.862447

In [125…

```
olist_df_model.isnull().sum()
```

Out[125]:

```
customer_id                 0
order_purchase_timestamp    0
order_id                    0
price                       0
dtype: int64
```

In [126…

```
olist_df_model.shape
```

Out[126]:

```
(104782, 4)
```

RFM Analysis

Recency

Get the last date of purchase. Find the most recent date and calculate number of days from the other purchases compared to this date.

In [127…

```
# Group dataset by customer id and get the max purchase date
recency_df = olist_df_model.groupby(by = 'customer_id', as_index = False)['order_purch
recency_df.rename(columns = {"order_purchase_timestamp": "last_purchase_date"}, inplac
recency_df["last_purchase_date"] = pd.to_datetime(recency_df["last_purchase_date"])
recency_df.head()
```

Out[127]:

	customer_id	last_purchase_date
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26
1	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32
2	0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43
3	0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20
4	000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:17

In [128…

```
# Get the most recent purchase date and use it to calculate number of days from this d
recent_date = olist_df_model['order_purchase_timestamp'].max()
```

```
recency_df['Recency'] = recency_df['last_purchase_date'].apply(lambda x: (recent_date
recency_df.head()
```

Out[128]:

	customer_id	last_purchase_date	Recency
0	00012a2ce6f8dcda20d059ce98491703	2017-11-14 16:08:26	259
1	000161a058600d5901f007fab4c27140	2017-07-16 09:40:32	380
2	0001fd6190edaaf884bcaf3d49edf079	2017-02-28 11:06:43	518
3	0002414f95344307404f0ace7a26f1d5	2017-08-16 13:09:20	349
4	000379cdec625522490c315e70c7a9fb	2018-04-02 13:42:17	120

Frequency

Get the number of orders a customer purchased and use this as the frequency

In [129…

```
frequency_df = olist_df_model.groupby(by = 'customer_id', as_index = False)['order_id'
frequency_df.columns = ['customer_id', 'Frequency']
frequency_df.head()
```

Out[129]:

	customer_id	Frequency
0	00012a2ce6f8dcda20d059ce98491703	1
1	000161a058600d5901f007fab4c27140	1
2	0001fd6190edaaf884bcaf3d49edf079	1
3	0002414f95344307404f0ace7a26f1d5	1
4	000379cdec625522490c315e70c7a9fb	1

In [130…

```
frequency_df.describe()
```

Out[130]:

	Frequency
count	91182.000000
mean	1.149152
std	0.554204
min	1.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	22.000000

Monetary

Get the total amount a customer purchased

In [131…
```
monetary_df = olist_df_model.groupby(by = 'customer_id', as_index = False)['price'].su
monetary_df.columns = ['customer_id', 'Monetary']
monetary_df.head()
```

Out[131]:

	customer_id	Monetary
0	00012a2ce6f8dcda20d059ce98491703	89.80
1	000161a058600d5901f007fab4c27140	54.90
2	0001fd6190edaaf884bcaf3d49edf079	179.99
3	0002414f95344307404f0ace7a26f1d5	149.90
4	000379cdec625522490c315e70c7a9fb	93.00

Merge the 3 datasets

In [132…
```
rf_df = recency_df.merge(frequency_df, on = 'customer_id')
rfm_df = rf_df.merge(monetary_df, on = 'customer_id').drop(columns = 'last_purchase_da
```

In [133…
```
rfm_df.head()
```

Out[133]:

	customer_id	Recency	Frequency	Monetary
0	00012a2ce6f8dcda20d059ce98491703	259	1	89.80
1	000161a058600d5901f007fab4c27140	380	1	54.90
2	0001fd6190edaaf884bcaf3d49edf079	518	1	179.99
3	0002414f95344307404f0ace7a26f1d5	349	1	149.90
4	000379cdec625522490c315e70c7a9fb	120	1	93.00

In [134…
```
rfm_df.tail()
```

Out[134]:

	customer_id	Recency	Frequency	Monetary
91177	fffcb937e9dd47a13f05ecb8290f4d3e	136	1	78.0
91178	fffecc9f79fd8c764f843e9951b11341	124	1	54.9
91179	fffeda5b6d849fbd39689bb92087f431	70	1	47.9
91180	ffff42319e9b2d713724ae527742af25	48	1	199.9
91181	ffffa3172527f765de70084a7e53aae8	332	2	21.8

Convert customer_id to index

In [135…
```
rfm_df = rfm_df.set_index('customer_id')
rfm_df.head()
```

Out[135]:

customer_id	Recency	Frequency	Monetary
00012a2ce6f8dcda20d059ce98491703	259	1	89.80
000161a058600d5901f007fab4c27140	380	1	54.90
0001fd6190edaaf884bcaf3d49edf079	518	1	179.99
0002414f95344307404f0ace7a26f1d5	349	1	149.90
000379cdec625522490c315e70c7a9fb	120	1	93.00

In [136…

```
rfm_df.describe()
```

Out[136]:

	Recency	Frequency	Monetary
count	91182.000000	91182.000000	91182.000000
mean	225.794137	1.149152	138.491396
std	144.446134	0.554204	210.737977
min	0.000000	1.000000	0.850000
25%	107.000000	1.000000	45.950000
50%	204.000000	1.000000	87.990000
75%	331.000000	1.000000	149.990000
max	572.000000	22.000000	13440.000000

Examine statistical distribution

In [137…

```
rfm_df_melted = pd.melt(rfm_df)
rfm_df_melted
```

Out[137]:

	variable	value
0	Recency	259.0
1	Recency	380.0
2	Recency	518.0
3	Recency	349.0
4	Recency	120.0
...	...	...
273541	Monetary	78.0
273542	Monetary	54.9
273543	Monetary	47.9
273544	Monetary	199.9
273545	Monetary	21.8

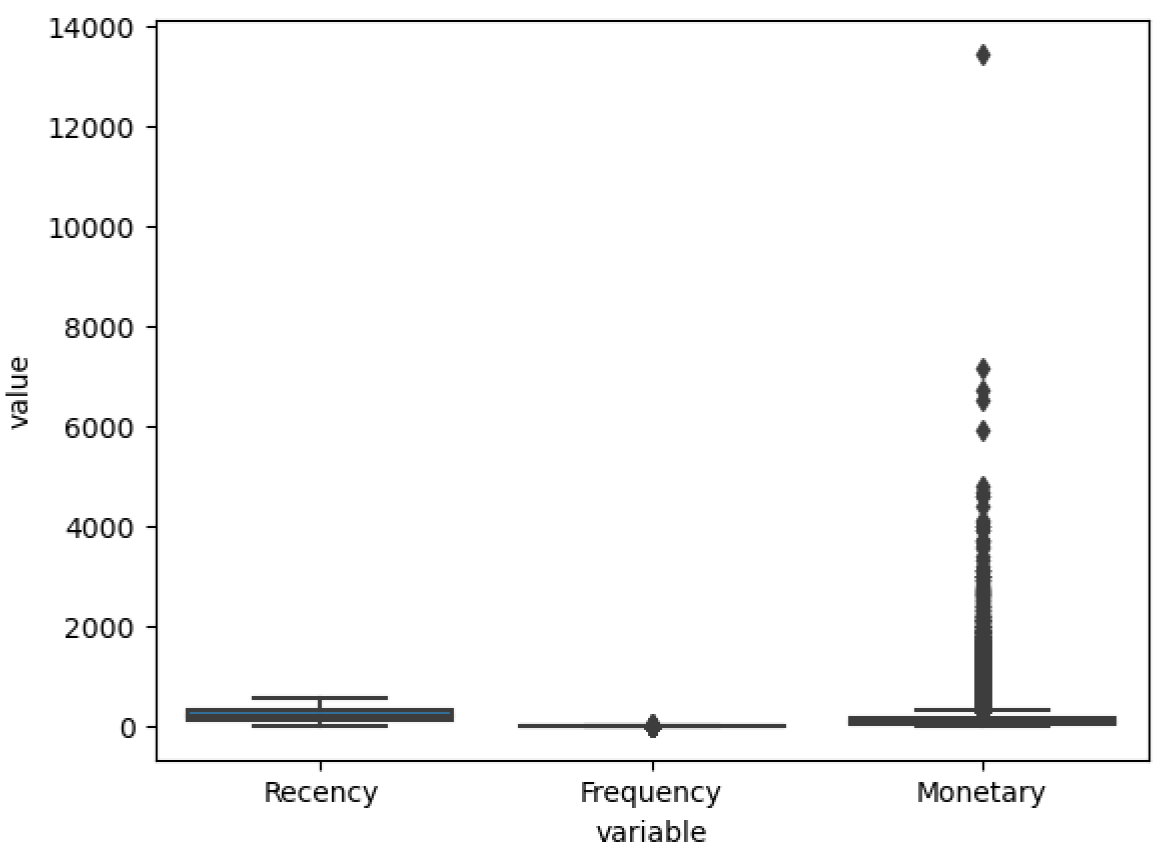
273546 rows × 2 columns

In [138…

```
sns.boxplot(data = rfm_df_melted, x = 'variable', y = 'value')
```

Out[138]:

```
<AxesSubplot:xlabel='variable', ylabel='value'>
```



The frequency values only had 1 value and the monetary values had a lot of outliers. I decided to drop the frequency values and used standard scaler to normalize the recency and monetary values.

Drop Frequency as all customers have only purchased 1 order with minimal items

In [139…
```
rm_df = rfm_df.drop('Frequency', axis = 1)
rm_df.head()
```

Out[139]:

	Recency	Monetary
customer_id		
00012a2ce6f8dcda20d059ce98491703	259	89.80
000161a058600d5901f007fab4c27140	380	54.90
0001fd6190edaaf884bcaf3d49edf079	518	179.99
0002414f95344307404f0ace7a26f1d5	349	149.90
000379cdec625522490c315e70c7a9fb	120	93.00

K-Means Clustering

Normalize the dataset

In [140…
```
standardizer = StandardScaler()
rm_scaled = standardizer.fit_transform(rm_df)
```

In [141…
```
rm_scaled
```

Out[141]:
```
array([[ 0.22988532, -0.2310531 ],
       [ 1.06757242, -0.39666251],
       [ 2.0229511 ,  0.19692147],
       ...,
       [-1.0785681 , -0.42987929],
       [-1.23087484,  0.2913995 ],
       [ 0.73526679, -0.55373045]])
```

Silhouette Score

In [142…
```
# Clusters for 2 - 6
range_n_clusters = [2, 3, 4, 5, 6]
silhouette_scores = []

for n_clusters in range_n_clusters:

    # Initialize the clusterer with n_clusters value and a random generator
    # seed of 10 for reproducibility.
    clusterer = KMeans(n_clusters = n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(rm_scaled)

    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the formed
    # clusters
    silhouette_avg = silhouette_score(rm_scaled, cluster_labels)
    silhouette_scores.append(silhouette_avg)
```

```
    print(f"For n_clusters = {n_clusters}. The average silhouette score is {silhouette
```

```
For n_clusters = 2. The average silhouette score is 0.46731442416207347
For n_clusters = 3. The average silhouette score is 0.5010325943820231
For n_clusters = 4. The average silhouette score is 0.5111099455792638
For n_clusters = 5. The average silhouette score is 0.4311208168994719
For n_clusters = 6. The average silhouette score is 0.43634171713433667
```

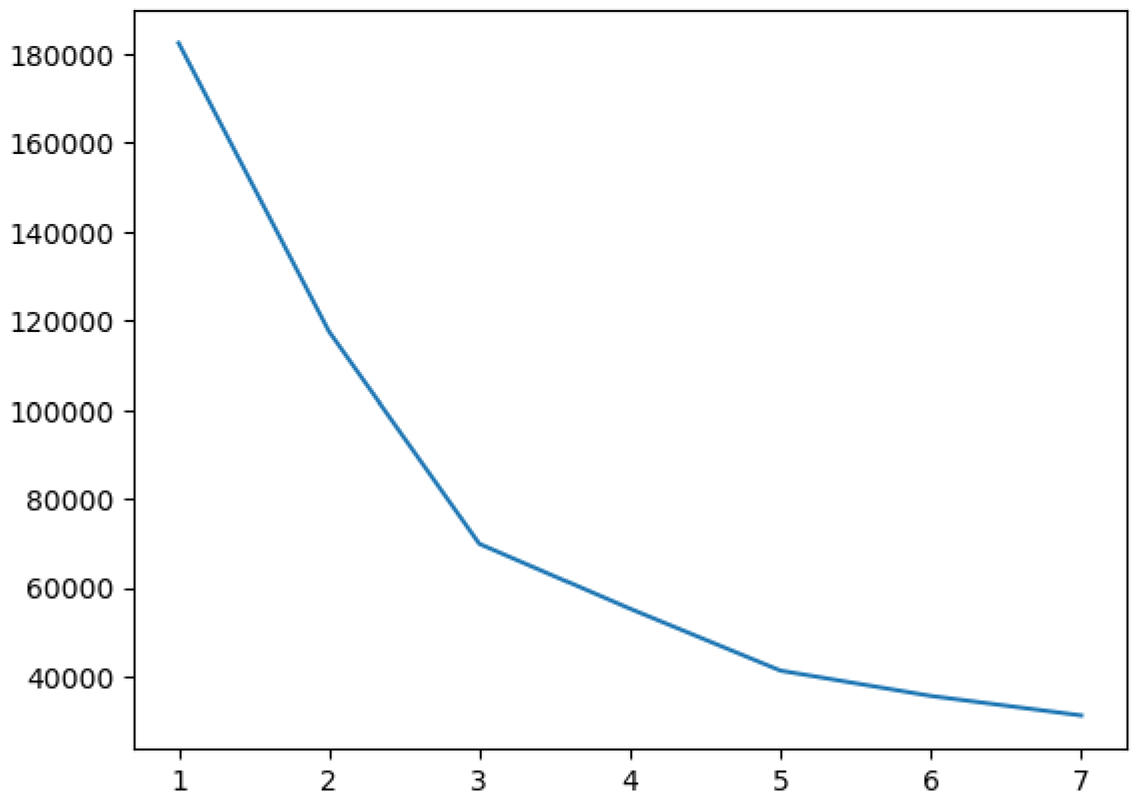
Elbow Method

In [143…

```
wcss = []

for i in range(1, 8):
    clustering = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    clustering.fit(rm_scaled)
    wcss.append(clustering.inertia_)

ks = [1, 2, 3, 4, 5, 6, 7]
sns.lineplot(x = ks, y = wcss)
```

Out[143]:

```
<AxesSubplot:>
```



K-Means

With the silhouette analysis and elbow method, it looks like the optimal number of clusters is 3.

In [144…

```
# perform k-means and fit the data
kmeans = KMeans(n_clusters = 3, max_iter = 50)
kmeans.fit(rm_scaled)
```

Out[144]:

```
KMeans(max_iter=50, n_clusters=3)
```

In [145…
```
# Determine which clusters each data point belongs to
clusters = kmeans.predict(rm_scaled)
```

In [146…
```
# Find the centers of each of the clusters
centers = kmeans.cluster_centers_
```

Convert to a dataset

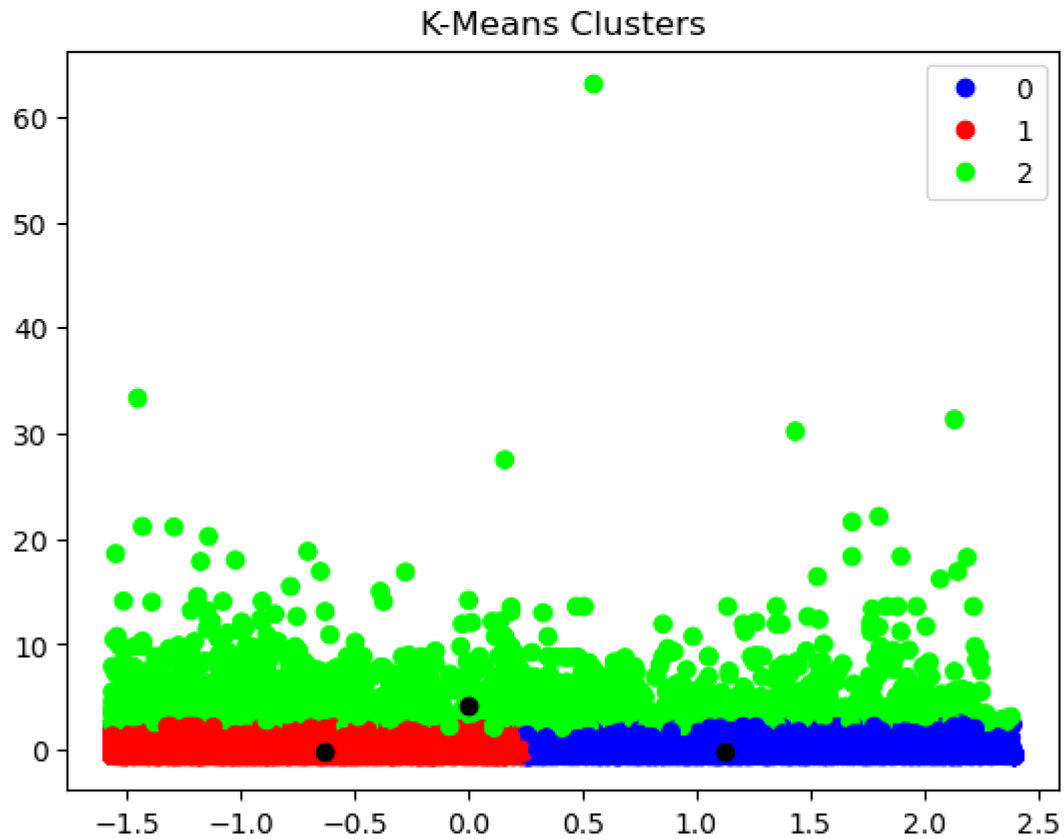
In [147…
```
# Add cluster number to the original data
rm_scaled_clustered = pd.DataFrame(rm_scaled, columns = rm_df.columns, index = rm_df.i
rm_scaled_clustered['cluster'] = clusters

rm_scaled_clustered.head()
```

Out[147]:

customer_id	Recency	Monetary	cluster
00012a2ce6f8dcda20d059ce98491703	0.229885	-0.231053	1
000161a058600d5901f007fab4c27140	1.067572	-0.396663	0
0001fd6190edaaf884bcaf3d49edf079	2.022951	0.196921	0
0002414f95344307404f0ace7a26f1d5	0.852958	0.054137	0
000379cdec625522490c315e70c7a9fb	-0.732416	-0.215868	1

Visualize the Clusters

In [148…
```
fig = plt.figure()

ax = fig.add_subplot()

cluster_plot = ax.scatter(rm_scaled_clustered['Recency'], rm_scaled_clustered['Monetar
ax.scatter(centers[:, 0], centers[:, 1], c = 'black')

plt.legend(*cluster_plot.legend_elements())
plt.title('K-Means Clusters')
plt.show()
```

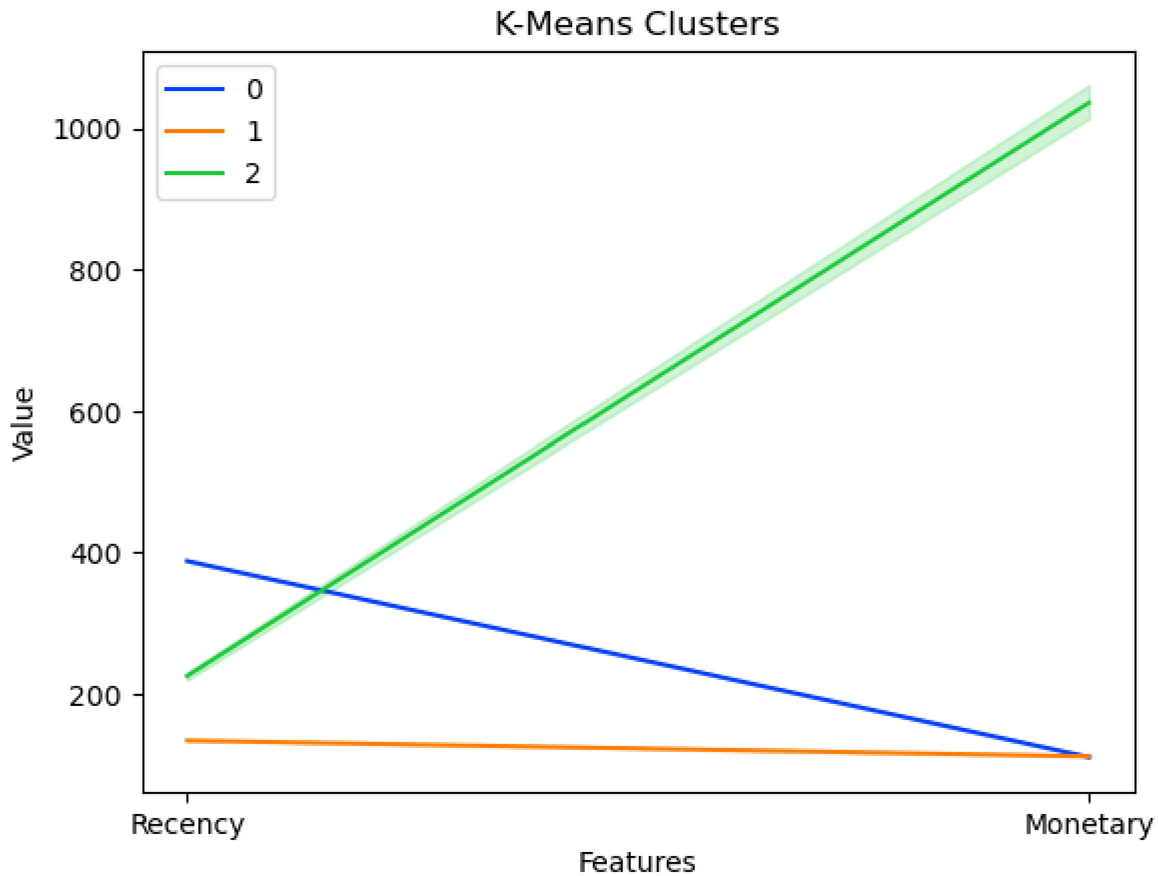


In [149…

```
rm_ = pd.DataFrame(standardizer.inverse_transform(rm_scaled))
rm_.columns = rm_df.columns
rm_['customer_id'] = rm_df.index
rm_['cluster'] = kmeans.labels_

rm_melted_normalized = pd.melt(rm_.reset_index(),
                                id_vars = ['customer_id', 'cluster'],
                                value_vars = ['Recency', 'Monetary'],
                                var_name = 'Features',
                                value_name = 'Value')
palette = sns.color_palette("bright", 4)
sns.lineplot(x = 'Features', y = 'Value', hue = 'cluster', palette = palette, data = r
plt.title('K-Means Clusters')
plt.legend()
```

Out[149]:

```
<matplotlib.legend.Legend at 0x1f88f428100>
```



Analysis

- Cluster 0: These are customers that haven't purchased items in a long time with a low monetary values (Lost customers)
- Cluster 1: These are customers that have purchased recently but with low monetary values
- Cluster 2: These are customers that have purchased items somewhat recently but have the highest monetary value

```
In [ ]:
```