Olist Data Analysis Project

Ahmed Nasser

2024-03-20

Introduction:

As a senior data analyst working at a fast-growing e-commerce platform, Olist, based in Brazil. -The company operates a marketplace connecting sellers and buyers across various product categories-. My task is to analyze the available dataset to derive actionable insights to improve business operations and customer experience, by first answering the business objectives.
In this analysis case study I'll use the six steps of the data analysis process, which is (Ask, Prepare, Process, Analyze, Share, and Act).

Business Objectives:

1. Exploring the data.
2. Find the customer segmentation.
3. Create Churn Analysis.
4. Design a Predictive Model.
5. Visualize the findings and reporting it.

Ask:

In this step we will ask the important questions -*and answering them in the upcoming steps*- to meet the business objectives.

- **Exploring the data:**
  1. What is the distribution of orders over time, and trends in order volume and order statuses?
  2. What the most popular product categories and sellers on the platform?
  3. What is the customer demographics, such as location and purchasing behavior?
- **Find the customer segmentation:**
  1. Create segment customers based on their purchasing behavior, geographic location, or other relevant factors.
  2. Explore different segmentation techniques tailored to the characteristics of the Brazilian e-commerce market.
- **Create Churn Analysis:**
  1. Analyze customer churn rate over time, identifying factors influencing churn and proposing strategies to retain customers.
  2. Examine the impact of product quality, delivery speed, and customer service on customer satisfaction and retention.
- **Design a Predictive Model:**

1. Build predictive models to forecast sales, customer demand, or product popularity.
2. Evaluate the performance of different forecasting algorithms and assess their suitability for the Brazilian e-commerce market.

- **Visualize the findings and reporting it:**

1. Create visualizations and dashboards to present key insights and trends.
2. Prepare a comprehensive report summarizing my analysis and recommendations.
3. Present my findings to stakeholders, highlighting actionable recommendations to drive business growth and improve customer satisfaction.

Prepare:

- Now we will download the Brazilian E-commerce Dataset from Olist from https://drive.google.com/file/d/1pHkWkE4lveePWOU9Ornn8uNeI7RjQ1BF/view and start exploring.

Process:

- In this case study I'll be using **R** to analyze the data.

1. Unzip the csv files in the same folder.
2. Open RStudio and start new session.
3. Create .RMD file and start our script.
4. Read csv files into multiple dataframes.

```
library(tidyverse)
orders <- read_csv('olist_orders_dataset.csv')
customers <- read_csv('olist_customers_dataset.csv')
products <- read_csv('olist_products_dataset.csv')
sellers <- read_csv('olist_sellers_dataset.csv')
payments <- read_csv('olist_order_payments_dataset.csv')
reviews <- read_csv('olist_order_reviews_dataset.csv')
geolocation <- read_csv('olist_geolocation_dataset.csv')
items <- read_csv('olist_order_items_dataset.csv')
category <- read_csv('product_category_name_translation.csv')
```

5. Use the **skimr** library to summarize the data and search for any inconsistency or null values in the important datasets (orders, customers, products, sellers, payments, and reviews).

```
# Load necessary libraries
library(dplyr)
library(lubridate)
library(cluster)
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(survival)
library(skimr)

#check the datasets
skim(orders)
```

Table 1: Data summary

Name	orders
Number of rows	99441
Number of columns	8
Column type frequency:	
character	3
POSIXct	5
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
order_id	0	1	32	32	0	99441	0
customer_id	0	1	32	32	0	99441	0
order_status	0	1	7	11	0	8	0

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
order_purchase_timestamp	0	1.00	2016-09-04 21:15:19	2018-10-17 17:30:18	2018-01-18 23:04:36	98875
order_approved_at	160	1.00	2016-09-15 12:16:38	2018-09-03 17:40:06	2018-01-19 11:36:13	90733
order_delivered_carrier_date	1783	0.98	2016-10-08 10:34:01	2018-09-11 19:48:28	2018-01-24 16:10:58	81018
order_delivered_customer_date	2965	0.97	2016-10-11 13:46:32	2018-10-17 13:22:46	2018-02-02 19:28:10	95664
order_estimated_delivery_date	0	1.00	2016-09-30 00:00:00	2018-11-12 00:00:00	2018-02-15 00:00:00	459

```r
skim(customers)
```

Table 4: Data summary

Name	customers
Number of rows	99441
Number of columns	5
Column type frequency:	

character	5
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
customer_id	0	1	32	32	0	99441	0
customer_unique_id	0	1	32	32	0	96096	0
customer_zip_code_prefix	0	1	5	5	0	14994	0
customer_city	0	1	3	32	0	4119	0
customer_state	0	1	2	2	0	27	0

```r
skim(products)
```

Table 6: Data summary

Name	products
Number of rows	32951
Number of columns	9
Column type frequency:	
character	2
numeric	7
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
product_id	0	1.00	32	32	0	32951	0
product_category_name	610	0.98	3	46	0	73	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
product_name_lenght	610	0.98	48.48	10.25	5	42	51	57	76	
product_description_lenght	610	0.98	771.50	635.12	4	339	595	972	3992	
product_photos_qty	610	0.98	2.19	1.74	1	1	1	3	20	
product_weight_g	2	1.00	2276.47	4282.04	0	300	700	1900	40425	
product_length_cm	2	1.00	30.82	16.91	7	18	25	38	105	
product_height_cm	2	1.00	16.94	13.64	2	8	13	21	105	
product_width_cm	2	1.00	23.20	12.08	6	15	20	30	118	

```r
skim(sellers)
```

Table 9: Data summary

Name	sellers
Number of rows	3095
Number of columns	4
Column type frequency:	
character	4
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
seller_id	0	1	32	32	0	3095	0
seller_zip_code_prefix	0	1	5	5	0	2246	0
seller_city	0	1	2	40	0	611	0
seller_state	0	1	2	2	0	23	0

```r
skim(payments)
```

Table 11: Data summary

Name	payments
Number of rows	103886
Number of columns	5
Column type frequency:	
character	2
numeric	3
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
order_id	0	1	32	32	0	99440	0
payment_type	0	1	6	11	0	5	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
payment_sequential	0	1	1.09	0.71	1	1.00	1	1.00	29.00	
payment_installments	0	1	2.85	2.69	0	1.00	1	4.00	24.00	
payment_value	0	1	154.10	217.49	0	56.79	100	171.84	13664.08	

```r
skim(reviews)
```

Table 14: Data summary

Name	reviews
Number of rows	99224
Number of columns	7
Column type frequency:	
character	4
numeric	1
POSIXct	2
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
review_id	0	1.00	32	32	0	98410	0
order_id	0	1.00	32	32	0	98673	0
review_comment_title	87658	0.12	1	26	0	4178	0
review_comment_message	58256	0.41	1	208	0	35743	18

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
review_score	0	1	4.09	1.35	1	4	5	5	5	

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
review_creation_date	0	1	2016-10-02 00:00:00	2018-08-31 00:00:00	2018-02-02 00:00:00	636
review_answer_timestamp	0	1	2016-10-07 18:32:28	2018-10-29 12:27:35	2018-02-04 22:41:47	98248

6. Now we will create a dataframe list with the datasets that has the same *Primary_key* or *Foreign_key* (orders, payments, reviews, items) to join the them together using *reduce* function in *dfx* dataframe.

```r
df_list <- list(orders,payments,reviews,items)
dfx <- df_list %>% reduce(full_join, by='order_id')
```

7. Join the rest of the datasets (products, customers, sellers) to the *dfx* and save it in *df_all* dataframe.

```r
dfx <- full_join(dfx, products)
```

```
## Joining with `by = join_by(product_id)`
```

```r
dfx <- full_join(dfx, customers)
```

```
## Joining with `by = join_by(customer_id)`
```

```r
df_all <- full_join(dfx, sellers)
```

```
## Joining with `by = join_by(seller_id)`
```

8. Now we need to review the collective dataframe's column names using *colnames()* function and the first few rows using *head()* function.

```r
colnames(df_all)
```

```
##  [1] "order_id"                      "customer_id"
##  [3] "order_status"                  "order_purchase_timestamp"
##  [5] "order_approved_at"             "order_delivered_carrier_date"
##  [7] "order_delivered_customer_date" "order_estimated_delivery_date"
##  [9] "payment_sequential"            "payment_type"
## [11] "payment_installments"          "payment_value"
## [13] "review_id"                     "review_score"
## [15] "review_comment_title"          "review_comment_message"
## [17] "review_creation_date"          "review_answer_timestamp"
## [19] "order_item_id"                 "product_id"
## [21] "seller_id"                     "shipping_limit_date"
## [23] "price"                         "freight_value"
## [25] "product_category_name"         "product_name_lenght"
## [27] "product_description_lenght"    "product_photos_qty"
## [29] "product_weight_g"              "product_length_cm"
## [31] "product_height_cm"             "product_width_cm"
## [33] "customer_unique_id"            "customer_zip_code_prefix"
## [35] "customer_city"                 "customer_state"
## [37] "seller_zip_code_prefix"        "seller_city"
## [39] "seller_state"
```

```r
head(df_all)
```

```
## # A tibble: 6 x 39
##   order_id   customer_id order_status order_purchase_times~1 order_approved_at
##   <chr>      <chr>       <chr>        <dttm>                 <dttm>
## 1 e481f51cb~ 9ef432eb62~ delivered    2017-10-02 10:56:33    2017-10-02 11:07:15
## 2 e481f51cb~ 9ef432eb62~ delivered    2017-10-02 10:56:33    2017-10-02 11:07:15
## 3 e481f51cb~ 9ef432eb62~ delivered    2017-10-02 10:56:33    2017-10-02 11:07:15
## 4 53cdb2fc8~ b0830fb474~ delivered    2018-07-24 20:41:37    2018-07-26 03:24:27
## 5 47770eb91~ 41ce2a54c0~ delivered    2018-08-08 08:38:49    2018-08-08 08:55:23
## 6 949d5b44d~ f88197465e~ delivered    2017-11-18 19:28:06    2017-11-18 19:45:59
## # i abbreviated name: 1: order_purchase_timestamp
## # i 34 more variables: order_delivered_carrier_date <dttm>,
## #   order_delivered_customer_date <dttm>, order_estimated_delivery_date <dttm>,
## #   payment_sequential <dbl>, payment_type <chr>, payment_installments <dbl>,
## #   payment_value <dbl>, review_id <chr>, review_score <dbl>,
## #   review_comment_title <chr>, review_comment_message <chr>,
## #   review_creation_date <dttm>, review_answer_timestamp <dttm>, ...
```

9. I noticed that some rows are duplicated due to the payment_type, so we'll remove the columns responsible for the duplication, and remove NA values.

```
df_all <- df_all %>% group_by(customer_id) %>%
  mutate(sum(payment_value))
```

```
df_all <- df_all %>% ungroup()
```

```
df_all_unique <- df_all %>% subset(select = -c(payment_sequential, payment_type, payment_installments, p
  rename(payment_value = `sum(payment_value)`) %>%
  unique()
```

```
colnames(df_all_unique)
```

```
##  [1] "order_id"                      "customer_id"
##  [3] "order_status"                  "order_purchase_timestamp"
##  [5] "order_approved_at"             "order_delivered_carrier_date"
##  [7] "order_delivered_customer_date" "order_estimated_delivery_date"
##  [9] "review_id"                     "review_score"
## [11] "review_comment_title"          "review_comment_message"
## [13] "review_creation_date"          "review_answer_timestamp"
## [15] "order_item_id"                 "product_id"
## [17] "seller_id"                     "shipping_limit_date"
## [19] "price"                         "freight_value"
## [21] "product_category_name"         "product_name_lenght"
## [23] "product_description_lenght"    "product_photos_qty"
## [25] "product_weight_g"              "product_length_cm"
## [27] "product_height_cm"             "product_width_cm"
## [29] "customer_unique_id"            "customer_zip_code_prefix"
## [31] "customer_city"                 "customer_state"
## [33] "seller_zip_code_prefix"        "seller_city"
## [35] "seller_state"                  "payment_value"
```

```
head(df_all_unique)
```

```
## # A tibble: 6 x 36
##   order_id   customer_id order_status order_purchase_times~1 order_approved_at
##   <chr>      <chr>       <chr>        <dttm>                 <dttm>
## 1 e481f51cb~ 9ef432eb62~ delivered    2017-10-02 10:56:33    2017-10-02 11:07:15
## 2 53cdb2fc8~ b0830fb474~ delivered    2018-07-24 20:41:37    2018-07-26 03:24:27
## 3 47770eb91~ 41ce2a54c0~ delivered    2018-08-08 08:38:49    2018-08-08 08:55:23
## 4 949d5b44d~ f88197465e~ delivered    2017-11-18 19:28:06    2017-11-18 19:45:59
## 5 ad21c59c0~ 8ab97904e6~ delivered    2018-02-13 21:18:39    2018-02-13 22:20:29
## 6 a4591c265~ 503740e9ca~ delivered    2017-07-09 21:57:05    2017-07-09 22:10:13
## # i abbreviated name: 1: order_purchase_timestamp
## # i 31 more variables: order_delivered_carrier_date <dttm>,
## #   order_delivered_customer_date <dttm>, order_estimated_delivery_date <dttm>,
## #   review_id <chr>, review_score <dbl>, review_comment_title <chr>,
## #   review_comment_message <chr>, review_creation_date <dttm>,
## #   review_answer_timestamp <dttm>, order_item_id <dbl>, product_id <chr>,
## #   seller_id <chr>, shipping_limit_date <dttm>, price <dbl>, ...
```

- Check the last few months to see if there are any missing data:

```
df_all_unique %>% count(date=format(order_purchase_timestamp,'%Y-%m')) %>%
  rename(count_of_orders=n) %>%
  arrange(desc(date))
```

```
## # A tibble: 25 x 2
##    date    count_of_orders
##    <chr>             <int>
##  1 2018-10               4
##  2 2018-09              16
##  3 2018-08            7310
##  4 2018-07            7134
##  5 2018-06            7091
##  6 2018-05            7955
##  7 2018-04            7991
##  8 2018-03            8288
##  9 2018-02            7807
## 10 2018-01            8312
## # i 15 more rows
```

- Apparently there's an issue in the last two months, so we will drop them:

```
df_all_unique <- df_all_unique %>% filter(as.Date(order_purchase_timestamp)<'2018-9-1')
```

After reviewing and cleaning, the data is consistent and descriptive.

Analyze & Visualize:

- Now it's time to analyze the data and answer the business task.
    1. What is the distribution of orders over time, and trends in order volume and order statuses?
    - Highest year (ranked)

```
df_all_unique %>% count(date=format(order_purchase_timestamp,'%Y')) %>%
  rename(count_of_orders=n) %>%
  mutate(rank=round(rank(-count_of_orders),0)) %>%
  arrange(desc(count_of_orders))
```

```
## # A tibble: 3 x 3
##   date  count_of_orders  rank
##   <chr>           <int> <dbl>
## 1 2018            61888     1
## 2 2017            51793     2
## 3 2016              391     3
```

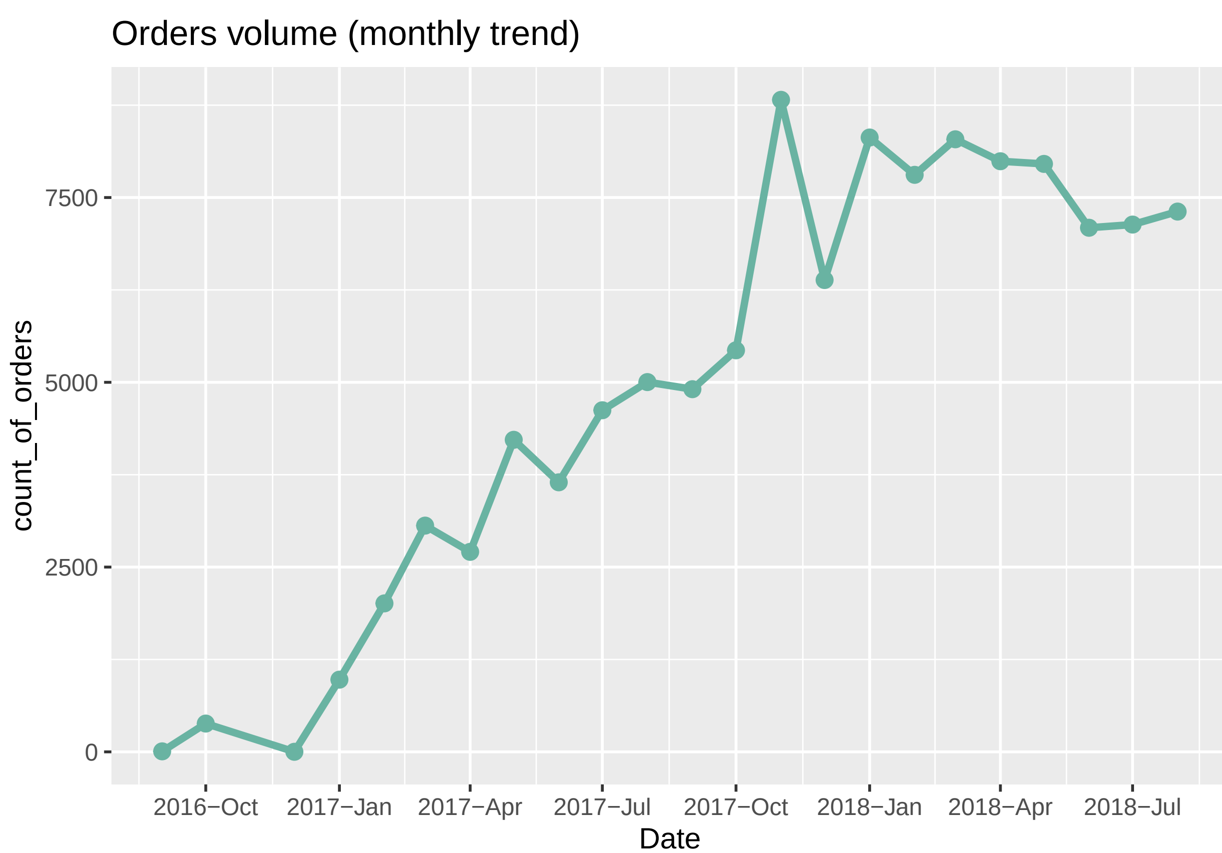
- Highest month of year (ranked)

```
df_all_unique %>% count(date=format(order_purchase_timestamp,'%Y-%m')) %>%
  rename(count_of_orders=n) %>%
  mutate(percent=round(count_of_orders/sum(count_of_orders)*100,2),rank=round(rank(-percent),0)) %>%
  arrange(desc(percent))
```

```
## # A tibble: 23 x 4
##    date    count_of_orders percent  rank
##    <chr>             <int>   <dbl> <dbl>
##  1 2017-11            8822    7.73     1
##  2 2018-01            8312    7.29     2
##  3 2018-03            8288    7.27     3
##  4 2018-04            7991    7.01     4
##  5 2018-05            7955    6.97     5
##  6 2018-02            7807    6.84     6
##  7 2018-08            7310    6.41     7
##  8 2018-07            7134    6.25     8
##  9 2018-06            7091    6.22     9
## 10 2017-12            6384    5.6     10
## # i 13 more rows
```

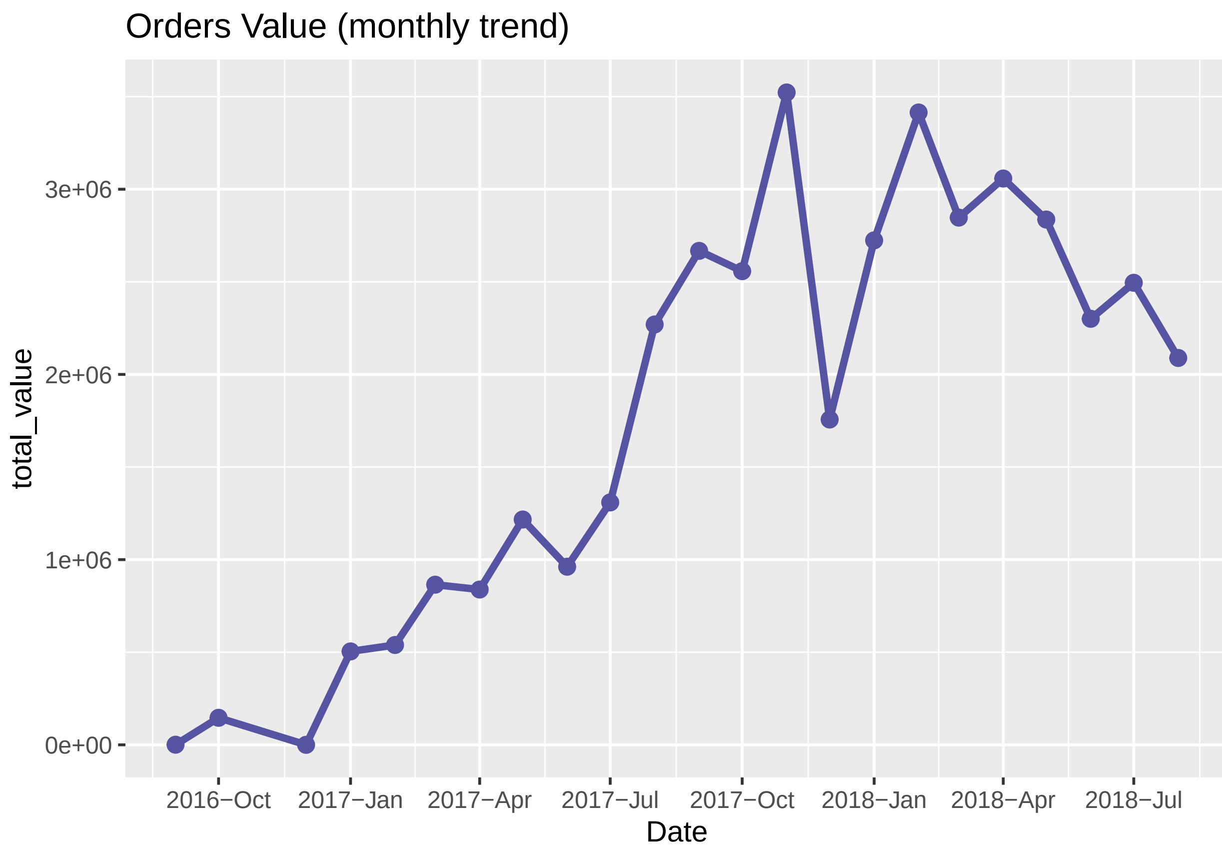
- Orders volume (monthly trend):

```
df_all_unique %>% count(date=format(order_purchase_timestamp,'%Y-%m')) %>%
  rename(count_of_orders=n) %>%
  ggplot(aes(ym(date),y=count_of_orders)) +
  geom_line(linewidth = 1.3,colour="#69b3a2") +
  geom_point(size=2.5,colour="#69b3a2") +
  scale_x_date(date_breaks = '3 months',date_labels='%Y-%b') +
  labs(title = 'Orders volume (monthly trend)',x='Date')
```



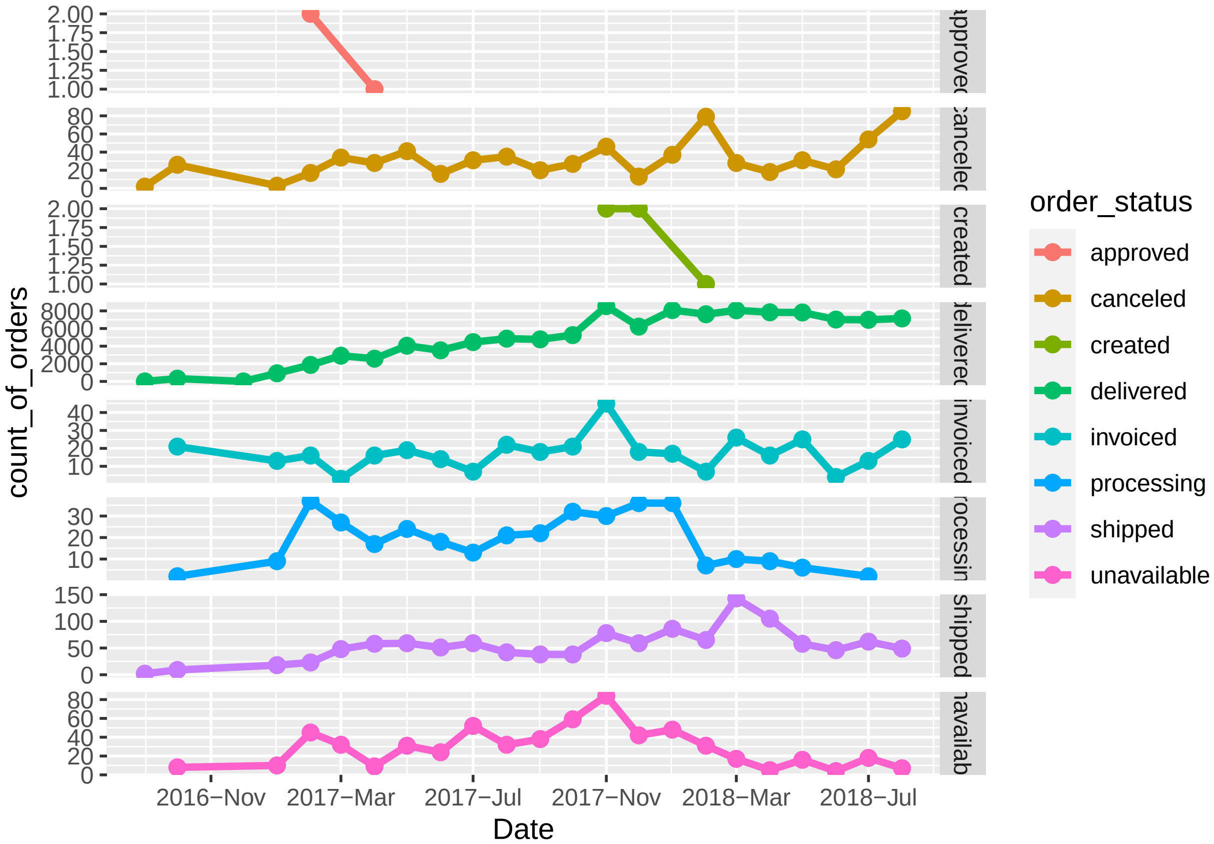
- Orders Value (monthly trend):

```r
df_all_unique %>% drop_na(payment_value) %>%
  group_by(date=format(order_purchase_timestamp,'%Y-%m')) %>%
  summarise(total_value=sum(payment_value)) %>%
  ggplot(aes(x=ym(date),y=total_value)) +
  geom_line(linewidth = 1.3,colour="#5653a2") +
  geom_point(size=2.5,colour="#5653a2") +
  scale_x_date(date_breaks = '3 months',date_labels='%Y-%b') +
  labs(title = 'Orders Value (monthly trend)',x='Date')
```



- number of order status per month:

```r
df_all_unique %>% count(order_status,date=format(order_purchase_timestamp,'%Y-%m')) %>%
  rename(count_of_orders=n) %>%
  ggplot(aes(ym(date),count_of_orders, colour=order_status, group=order_status)) +
  geom_line(linewidth = 1.3) +
  geom_point(size=2.5) +
  scale_x_date(date_breaks = '4 months',date_labels='%Y-%b') +
  labs(x='Date') +
  facet_grid(order_status~., scales = "free")
```



2. What the most popular product categories and sellers on the platform?

- Most ordered product categories:

```
df_all_unique %>% full_join(category) %>%
  count(product_category_name_english) %>%
  rename(count_of_orders=n) %>%
  arrange(desc(count_of_orders))
```

```
## # A tibble: 72 x 2
##    product_category_name_english count_of_orders
##    <chr>                                   <int>
##  1 bed_bath_table                          11270
##  2 health_beauty                            9727
##  3 sports_leisure                           8700
##  4 furniture_decor                          8415
##  5 computers_accessories                    7894
##  6 housewares                               6989
##  7 watches_gifts                            6001
##  8 telephony                                4550
##  9 garden_tools                             4361
## 10 auto                                     4256
## # i 62 more rows
```

- Highest sellers: (Note: the sellers are shown by the id, as we don't have their names for data privacy & security reasons)

```
df_all_unique %>% count(seller_id) %>%
  rename(count_of_orders=n) %>%
  arrange(desc(count_of_orders))
```

```
## # A tibble: 3,096 x 2
##    seller_id                        count_of_orders
##    <chr>                                      <int>
##  1 6560211a19b47992c3666cc44a7e94c0            2039
##  2 4a3ca9315b744ce9f8e9374361493884            2009
##  3 1f50f920176fa81dab994f9023523100            1940
##  4 cc419e0650a3c5ba77189a1882b7556a            1819
##  5 da8622b14eb17ae2831f4ac5b9dab84a            1574
##  6 955fee9216a65b617aa5c0531780ce60            1501
##  7 1025f0e2d44d7041d6cf58b6550e0bfa            1443
##  8 7c67e1448b00f6e969d365cea6b010ab            1375
##  9 ea8482cd71df3c1969d7b9473ff13abc            1204
## 10 7a67c85e85bb2ce8582c35f2203ad736            1175
## # i 3,086 more rows
```

3. What is the customer demographics, such as location and purchasing behavior?

- Highest cities in purchasing behavior:

```
df_all_unique %>% count(customer_city) %>%
  rename(count_of_orders=n) %>%
  arrange(desc(count_of_orders))
```

```
## # A tibble: 4,119 x 2
##    customer_city         count_of_orders
##    <chr>                           <int>
##  1 sao paulo                       18068
##  2 rio de janeiro                   7934
##  3 belo horizonte                   3191
##  4 brasilia                         2434
##  5 curitiba                         1769
##  6 campinas                         1676
##  7 porto alegre                     1632
##  8 salvador                         1429
##  9 guarulhos                        1342
## 10 sao bernardo do campo            1079
## # i 4,109 more rows
```

- **Find the customer segmentation:**
  1. Segment customers based on their purchasing behavior, geographic location, or other relevant factors.

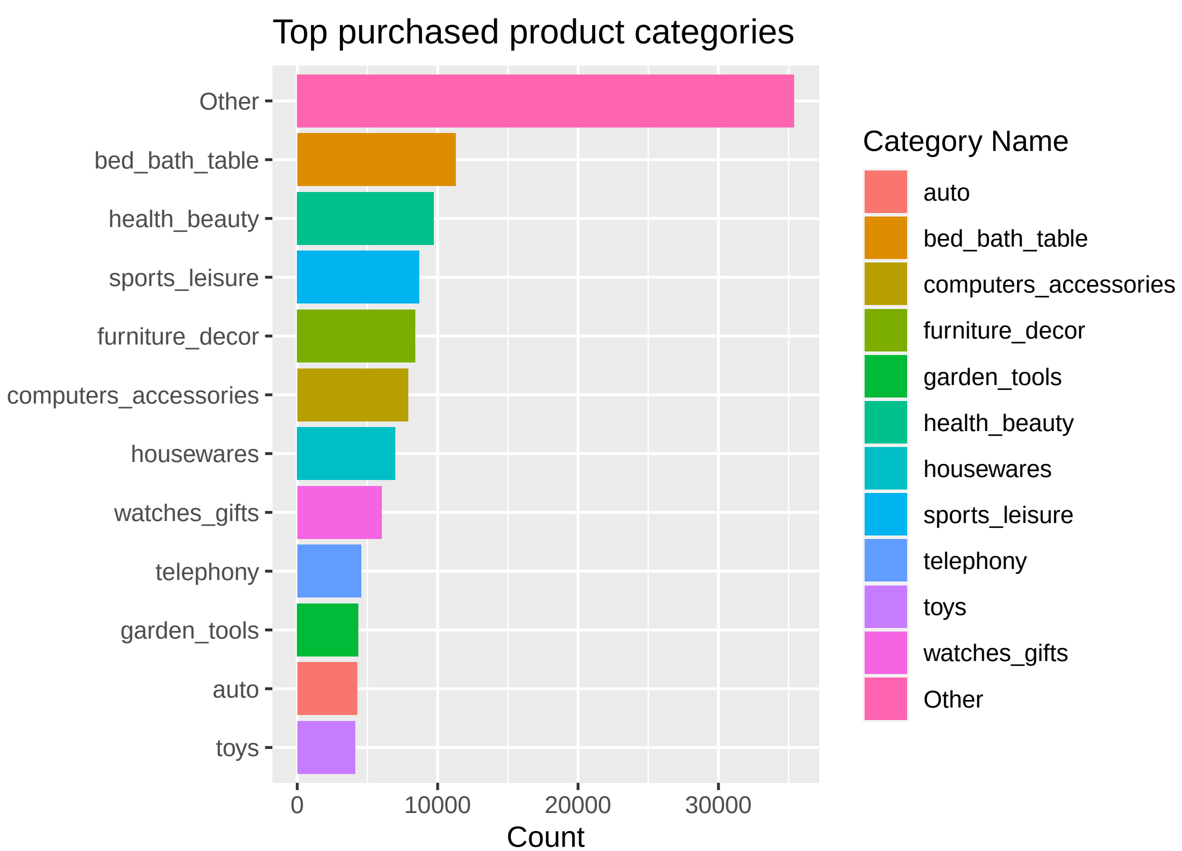
  - Highest customers in purchasing:

```
df_all_unique %>% group_by(customer_unique_id) %>%
  summarise(num_of_prod = n()) %>%
  arrange(desc(num_of_prod))
```

```
## # A tibble: 96,090 x 2
##    customer_unique_id               num_of_prod
##    <chr>                                  <int>
##  1 c8460e4251689ba205045f3ea17884a1          24
##  2 d97b3cfb22b0d6b25ac9ed4e9c2d481b          24
##  3 4546caea018ad8c692964e3382debd19          21
##  4 698e1cf81d01a3d389d96145f7fa6df8          20
##  5 c402f431464c72e27330a67f7b94d4fb          20
##  6 0f5ac8d5c31de21d2f25e24be15bbffb          18
##  7 8d50f5eadf50201ccdcedfb9e2ac8455          17
##  8 11f97da02237a49c8e783dfda6f50e8e          15
##  9 33176de67c05eeed870fd49f234387a0          15
## 10 eae0a83d752b1dd32697e0e7b4221656          15
## # i 96,080 more rows
```

- Top purchased product categories:

```
df_all_unique %>% full_join(category) %>%
  mutate(product_category_name_english=fct_lump_n(product_category_name_english, 11)) %>%
  drop_na(product_category_name_english) %>%
  group_by(product_category_name_english) %>%
  summarize(Count=n()) %>%
  ggplot() +
    geom_bar(aes(x=Count,y=reorder(product_category_name_english,Count),fill=product_category_name_engl
    labs(title = 'Top purchased product categories',y='',fill='Category Name' )
```



- Create a purchase_behavior df:

```r
purchase_behavior <- df_all_unique %>% drop_na(product_category_name) %>%
  mutate(
    # Extract hour of purchase to find peak shopping hours
    Hour_of_Purchase = hour(order_purchase_timestamp),

    # Extract day of purchase to find peak shopping days/seasons
    Day_of_Purchase = wday(order_purchase_timestamp, label = TRUE, abbr = FALSE),

    # Extract month of purchase to find peak shopping months/seasons
    Month_of_Purchase = month(order_purchase_timestamp, label = TRUE, abbr = FALSE)) %>%
  group_by(customer_unique_id) %>%
  summarise(
    # Consumer Preferences: Most frequently purchased product category
    Most_Frequent_Category = names(sort(table(product_category_name), decreasing = TRUE)[1]),

    # Consumer Preferences: Most frequently geographic location (city)
    Most_Frequent_City = names(sort(table(customer_city), decreasing = TRUE)[1]),

    # Purchase Frequency: Total number of purchases
    Total_Purchases = n(),

    # Average Hour of Purchase
    Avg_Hour_of_Purchase = mean(Hour_of_Purchase),

    # Most Common Day of Purchase
    Most_Common_Day = names(sort(table(Day_of_Purchase), decreasing = TRUE)[1]),

    # Most Common Month of Purchase
    Most_Common_Month = names(sort(table(Month_of_Purchase), decreasing = TRUE)[1]))

head(purchase_behavior)
```

```
## # A tibble: 6 x 7
##   customer_unique_id   Most_Frequent_Category Most_Frequent_City Total_Purchases
##   <chr>                <chr>                  <chr>                        <int>
## 1 0000366f3b9a7992bf8~ cama_mesa_banho        cajamar                          1
## 2 0000b849f77a49e4a4c~ beleza_saude           osasco                           1
## 3 0000f46a3911fa3c080~ papelaria              sao jose                         1
## 4 0000f6ccb0745a6a4b8~ telefonia              belem                            1
## 5 0004aac84e0df4da2b1~ telefonia              sorocaba                         1
## 6 0004bd2a26a76fe21f7~ ferramentas_jardim     sao paulo                        1
## # i 3 more variables: Avg_Hour_of_Purchase <dbl>, Most_Common_Day <chr>,
## #   Most_Common_Month <chr>
```

2. Explore different segmentation techniques tailored to the characteristics of the Brazilian e-commerce market.

- The percent of customers whose bought multiple products:

```
result <- df_all_unique %>% count(customer_unique_id) %>%
  summarise(total_customers_count = n())

df_all_unique %>% group_by(customer_unique_id) %>%
  summarise(num_of_prod = n(),count=n_distinct(customer_unique_id)) %>%
  filter(num_of_prod > 1) %>%
  reframe(num_retend_customers = sum(count),total_customers = result$total_customers_count,
          retend_cust_perc = round(num_retend_customers/total_customers*100,2))
```

```
## # A tibble: 1 x 3
##   num_retend_customers total_customers retend_cust_perc
##                  <int>           <int>            <dbl>
## 1                12012           96090             12.5
```

- Highest states in purchasing:

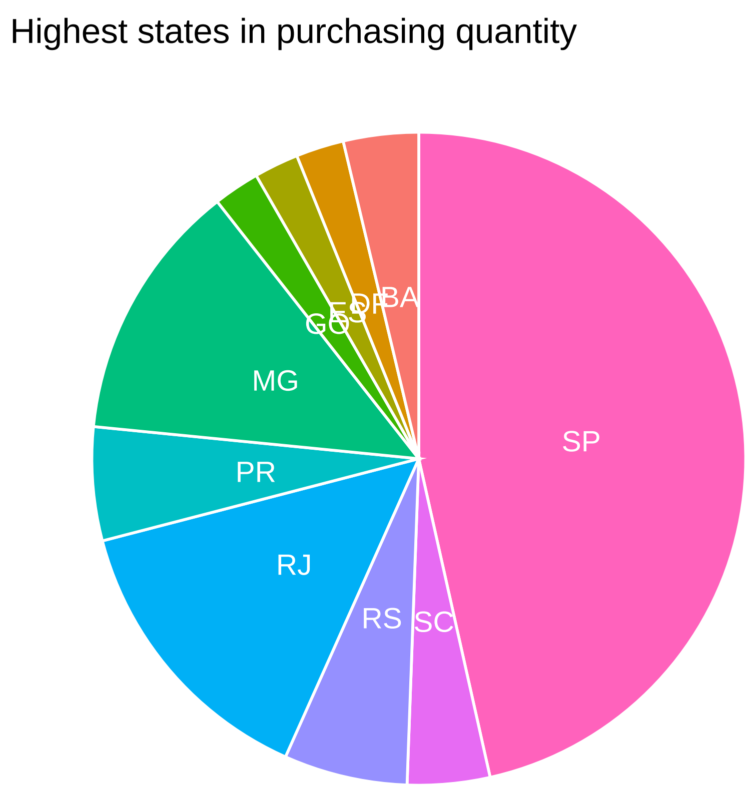
```
df_all_unique %>%
  mutate(customer_state=fct_lump_n(customer_state, 10)) %>%
  drop_na(customer_state) %>%
  group_by(customer_state) %>%
  summarize(Count=n()) %>%
  filter(customer_state != 'Other') %>%
  arrange(desc(customer_state))
```

```
## # A tibble: 10 x 2
##    customer_state Count
##    <fct>          <int>
##  1 SP             48083
##  2 SC              4215
##  3 RS              6321
##  4 RJ             14754
##  5 PR              5811
##  6 MG             13297
##  7 GO              2366
##  8 ES              2277
##  9 DF              2448
## 10 BA              3841
```

- Highest states in purchasing pie chart:

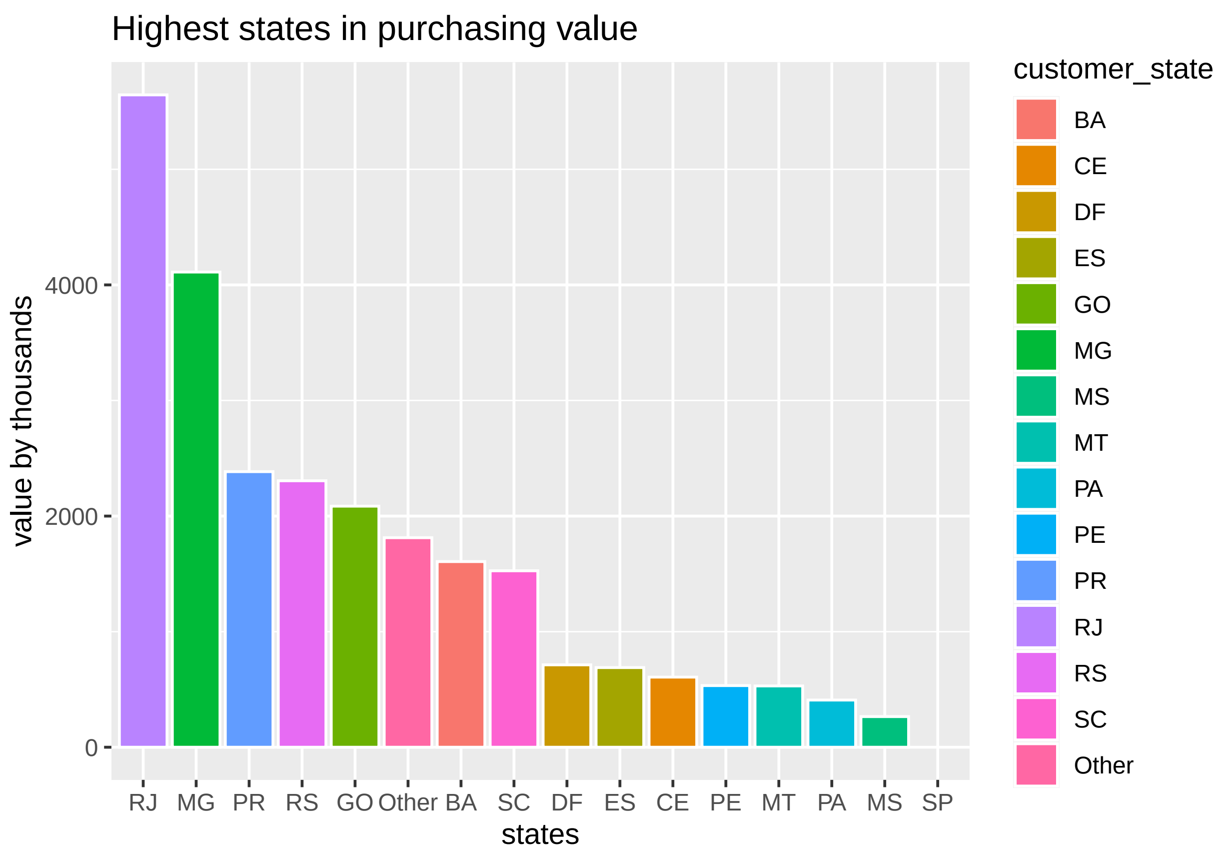
```
df_all_unique %>%
  mutate(customer_state=fct_lump_n(customer_state, 10)) %>%
  drop_na(customer_state) %>%
  group_by(customer_state) %>%
  summarize(Count=n()) %>%
  filter(customer_state != 'Other') %>%
  arrange(desc(customer_state)) %>%
  mutate(prop = Count / sum(Count) *100) %>%
  mutate(ypos = cumsum(prop)- 0.5*prop ) %>%
  ggplot(aes(x="", y=prop, fill=customer_state)) +
  geom_bar(stat="identity", width=1, color="white") +
```

```r
  coord_polar("y", start=0) +
  theme_void() +
  theme(legend.position="none") +
  geom_text(aes(y = ypos, label = customer_state), color = "white", size=4) +
  labs(title = 'Highest states in purchasing quantity')
```



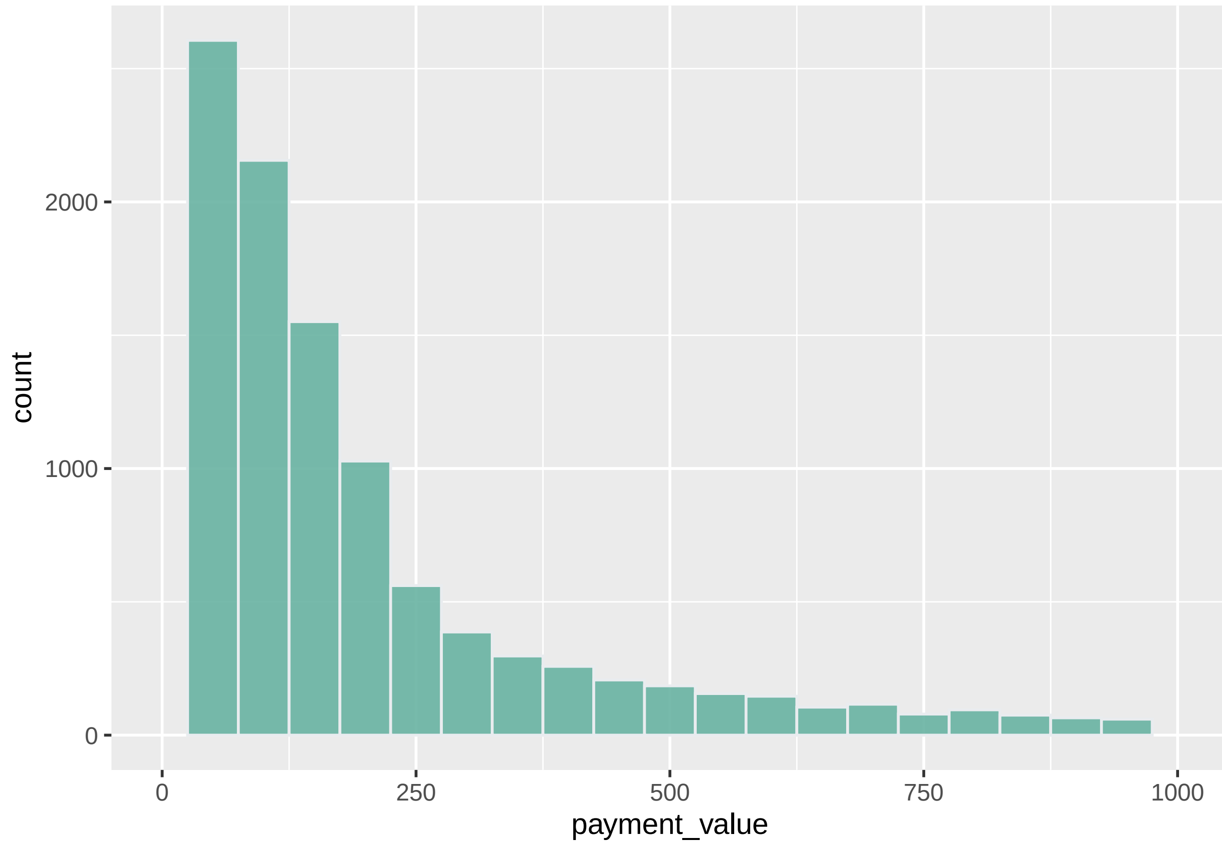
- Highest states in purchasing value:

```r
df_all_unique %>%
  mutate(customer_state=fct_lump_n(customer_state, 15)) %>%
  drop_na(customer_state) %>%
  group_by(customer_state) %>%
  summarize(total_value_k=sum(payment_value)/1000) %>%
  ggplot(aes(reorder(customer_state,desc(total_value_k)), total_value_k, fill=customer_state)) +
  geom_bar(stat="identity", color="white") +
  labs(title = 'Highest states in purchasing value', x='states', y='value by thousands')
```



- Histogram with the purchasing values:

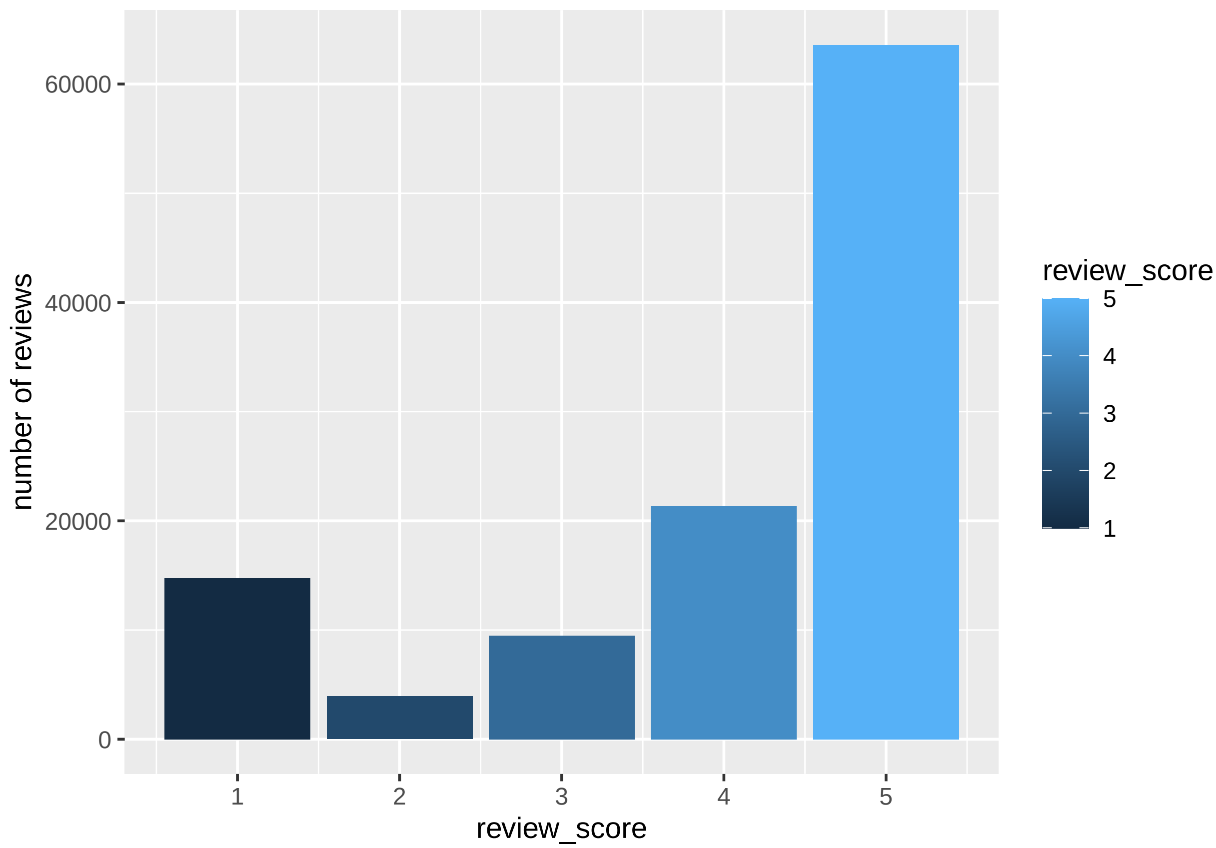
```
df_all_unique %>% drop_na() %>%
  ggplot() +
    geom_histogram(aes(payment_value),binwidth=50, fill="#69b3a2",color="#e9ecef", alpha=0.9)+
    xlim(0, 1000)
```



- **Create Churn Analysis:**
    1. Analyze customer churn rate over time, identifying factors influencing churn and proposing strategies to retain customers.

    – To calculate the churn rate we must analyze the reviews first.
    – Count of review scores:

```
df_all_unique %>% count(review_score) %>%
  drop_na() %>%
  arrange(desc(n)) %>%
  ggplot() +
  geom_col(aes(review_score,n,fill=review_score)) +
  labs(y='number of reviews')
```



- The most repeated review comment in the scored 1 reviews:

```
df_all_unique %>% group_by(review_comment_message,review_score) %>%
  filter(review_score==1, review_comment_message!='NA',order_status=='delivered') %>%
  select(review_comment_title,review_score,review_comment_message) %>%
  count(review_comment_message) %>%
  rename(count_of_review=n) %>%
  arrange(desc(count_of_review))
```

```
## # A tibble: 7,033 x 3
## # Groups:   review_comment_message, review_score [7,033]
##    review_comment_message                         review_score count_of_review
##    <chr>                                                 <dbl>           <int>
##  1 Não recebi o produto                                      1              37
##  2 Eu estou tentando cancelar faz tempo devido o v~          1              21
##  3 Ainda não recebi o produto                                1              17
##  4 Não recebi                                                1              17
##  5 Comprei 14 unidades e recebi somente 9. Ainda n~          1              14
##  6 nao recebi o produto                                      1              13
##  7 Eu recebi 1 taça de cada, eu comprei 6 de cada ~          1              12
##  8 bom dia ainda não recebi toa a minha encomenda,~          1              12
##  9 Ainda não recebi                                          1              11
## 10 Eu pedi 6 trios de pendentes e vcs só M entrega~          1              11
## # i 7,023 more rows
```

The most frequent review comment is 'I didn't receive the product', although the order status is delivered.

- The AVG lowest sellers in review score:

```
df_all_unique %>% group_by(seller_id) %>%
  summarise(AVG_score=round(mean(review_score)),count_of_reviews=n()) %>%
  arrange(AVG_score,desc(count_of_reviews))
```

```
## # A tibble: 3,096 x 3
##    seller_id                        AVG_score count_of_reviews
##    <chr>                                <dbl>            <int>
##  1 b37c4c02bda3161a7546a4e6d222d5b2         1               15
##  2 8d92f3ea807b89465643c219455e7369         1                8
##  3 ec2e006556300a79a5a91e4876ab3a56         1                8
##  4 a0e19590a0923cdd0614ea9427713ced         1                7
##  5 010da0602d7774602cd1b3f5fb7b709e         1                5
##  6 3bfad056cf05c00dabe2f895925d83b1         1                5
##  7 90d4125885ab6c86e8820a722be71974         1                5
##  8 40536e7ca18e1bce252828e5876466cc         1                4
##  9 4e42581f08e8cfc7c090f930bac4552a         1                4
## 10 bcf5566870987da7bc811fbc8c5b9fd9         1                4
## # i 3,086 more rows
```

- The AVG lowest cities in review score:

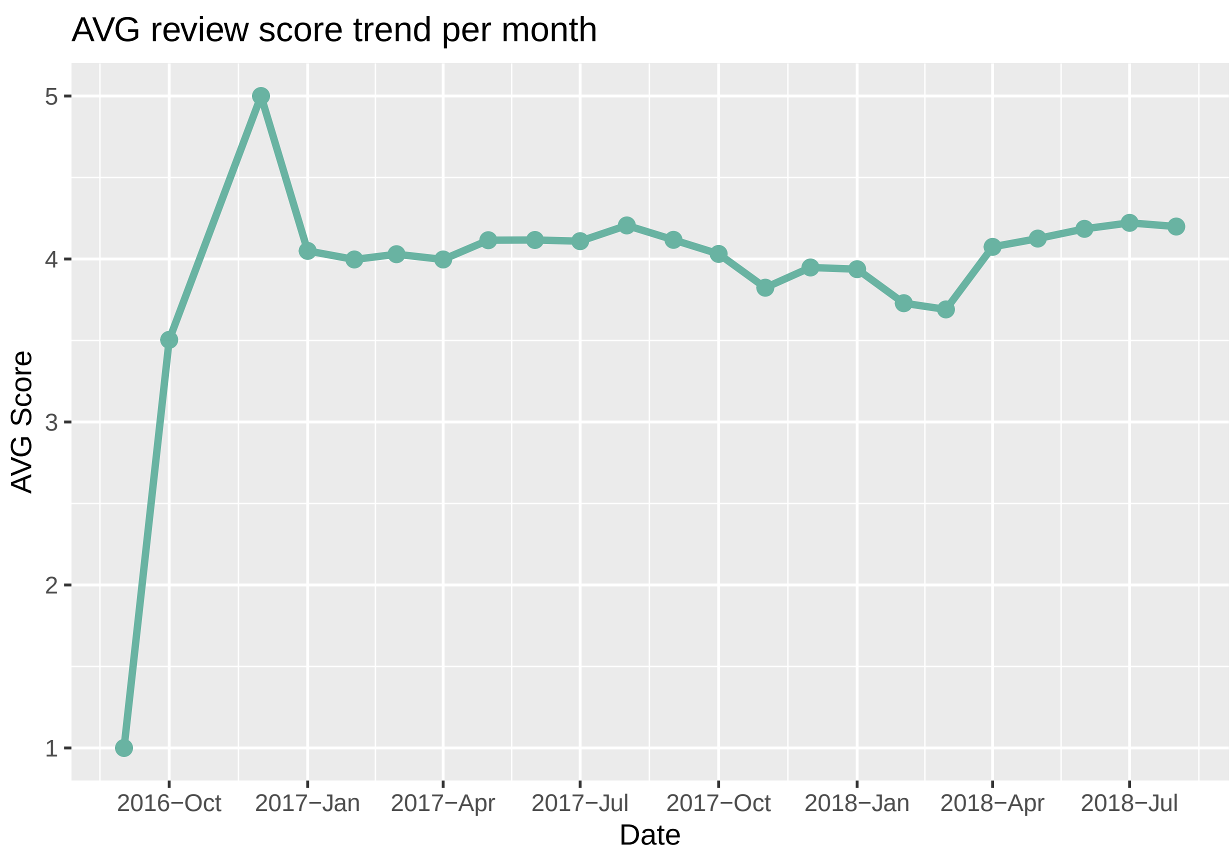
```
df_all_unique %>% group_by(customer_city) %>%
  drop_na(review_score) %>%
  summarise(AVG_score=round(mean(review_score)),count_of_reviews=n()) %>%
  arrange(AVG_score,desc(count_of_reviews))
```

```
## # A tibble: 4,117 x 3
##    customer_city       AVG_score count_of_reviews
##    <chr>                   <dbl>            <int>
##  1 candido godoi               1                6
##  2 belmonte                    1                4
##  3 cafeara                     1                4
##  4 itororo                     1                4
##  5 arembepe                    1                3
##  6 buriti                      1                3
##  7 chapadao do lageado         1                3
##  8 jose boiteux                1                3
##  9 maioba                      1                3
## 10 mercedes                    1                3
## # i 4,107 more rows
```

- AVG review score trend per month:

```
df_all_unique %>% drop_na(review_score) %>% group_by(date=format(order_purchase_timestamp,'%Y-%m')) %>%
  summarise(AVG_score=mean(review_score)) %>%
  ggplot(aes(ym(date),y=AVG_score)) +
  geom_line(linewidth = 1.3,colour="#69b3a2") +
```

```
  geom_point(size=2.5,colour="#69b3a2") +
  scale_x_date(date_breaks = '3 months',date_labels='%Y-%b') +
  labs(title = 'AVG review score trend per month', x='Date', y='AVG Score')
```



2. Examine the impact of product quality, delivery speed, and customer service on customer satisfaction and retention.

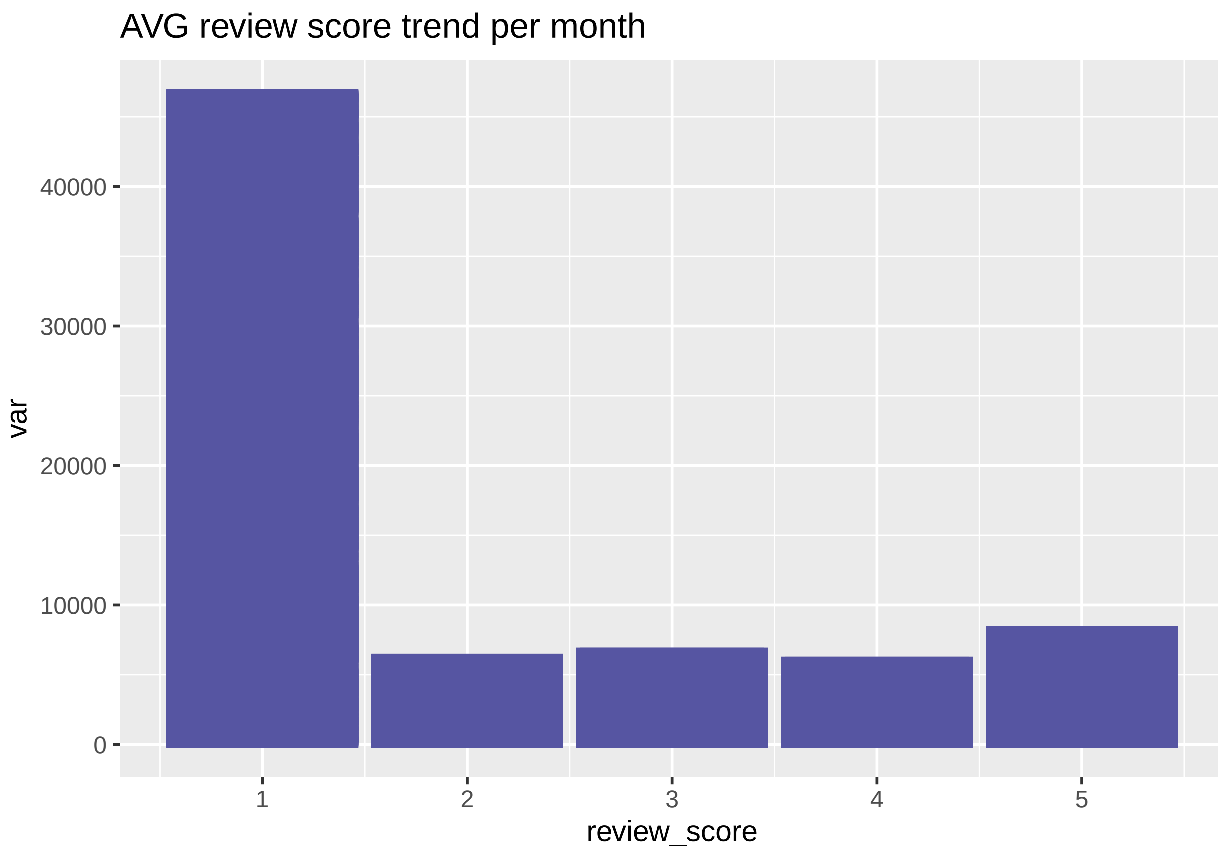
- The impact of the delayed orders on the review score.

```
count_of_1_review <- df_all_unique %>% drop_na(review_score) %>%
  filter(order_status == 'delivered',review_score==1) %>%
  summarise(count_of_1=n())

df_all_unique %>% drop_na(review_score) %>%
  mutate(var=date(order_delivered_customer_date) - date(order_estimated_delivery_date)) %>%
  filter(order_status == 'delivered',var>0, review_score==1) %>%
  summarise(percent_of_delay=n()/count_of_1_review$count_of_1*100)
```

```
## # A tibble: 1 x 1
##   percent_of_delay
##              <dbl>
## 1             30.8
```

```r
df_all_unique %>% drop_na(review_score) %>%
  mutate(var=date(order_delivered_customer_date) - date(order_estimated_delivery_date)) %>%
  filter(order_status == 'delivered',var>0) %>%
  ggplot() +
  geom_col(aes(var,x=review_score), linewidth = 1.3,colour="#5655a2") +
  labs(title = 'AVG review score trend per month')
```



- Customer Segmentation using K-Means Clustering based on purchase_behavior

```r
set.seed(123)
kmeans_result <- kmeans(purchase_behavior[, c("Total_Purchases", "Avg_Hour_of_Purchase")], centers = 5)
purchase_behavior$Segment <- kmeans_result$cluster
```

- create a Churn Analysis:

```r
# Define churn based on a condition, e.g., no purchases in the last 6 months
current_date <- max(df_all_unique$order_purchase_timestamp)
df_all_unique2 <- df_all_unique %>%
  group_by(customer_unique_id) %>%
  summarise(Last_Purchase_Date = max(order_purchase_timestamp)) %>%
  ungroup() %>%
  mutate(Churn = ifelse(difftime(current_date, Last_Purchase_Date, units = "days") > 180, 1, 0))

df_all_unique2 <- full_join(df_all_unique2,df_all_unique)
```

```
## Joining with `by = join_by(customer_unique_id)`
```

```
# Merge churn data back to purchase_behavior
purchase_behavior <- merge(purchase_behavior, df_all_unique2[, c("customer_unique_id", "Churn")], by =
head(purchase_behavior)
```

```
##                  customer_unique_id Most_Frequent_Category Most_Frequent_City
## 1 0000366f3b9a7992bf8c76cfdf3221e2          cama_mesa_banho            cajamar
## 2 0000b849f77a49e4a4ce2b2a4ca5be3f             beleza_saude            osasco
## 3 0000f46a3911fa3c0805444483337064                papelaria           sao jose
## 4 0000f6ccb0745a6a4b88665a16c9f078                telefonia              belem
## 5 0004aac84e0df4da2b147fca70cf8255                telefonia           sorocaba
## 6 0004bd2a26a76fe21f786e4fbd80607f      ferramentas_jardim          sao paulo
##   Total_Purchases Avg_Hour_of_Purchase Most_Common_Day Most_Common_Month
## 1               1                   10        Thursday               May
## 2               1                   11          Monday               May
## 3               1                   21          Friday             March
## 4               1                   20        Thursday           October
## 5               1                   19         Tuesday          November
## 6               1                   19        Thursday             April
##   Segment Churn
## 1       5     0
## 2       5     0
## 3       1     1
## 4       1     1
## 5       1     1
## 6       1     0
```

- **Design a Predictive Model:**
  1. Build predictive models to forecast sales, customer demand, or product popularity.
  - Building a Predictive Modeling for Sales Forecasting:

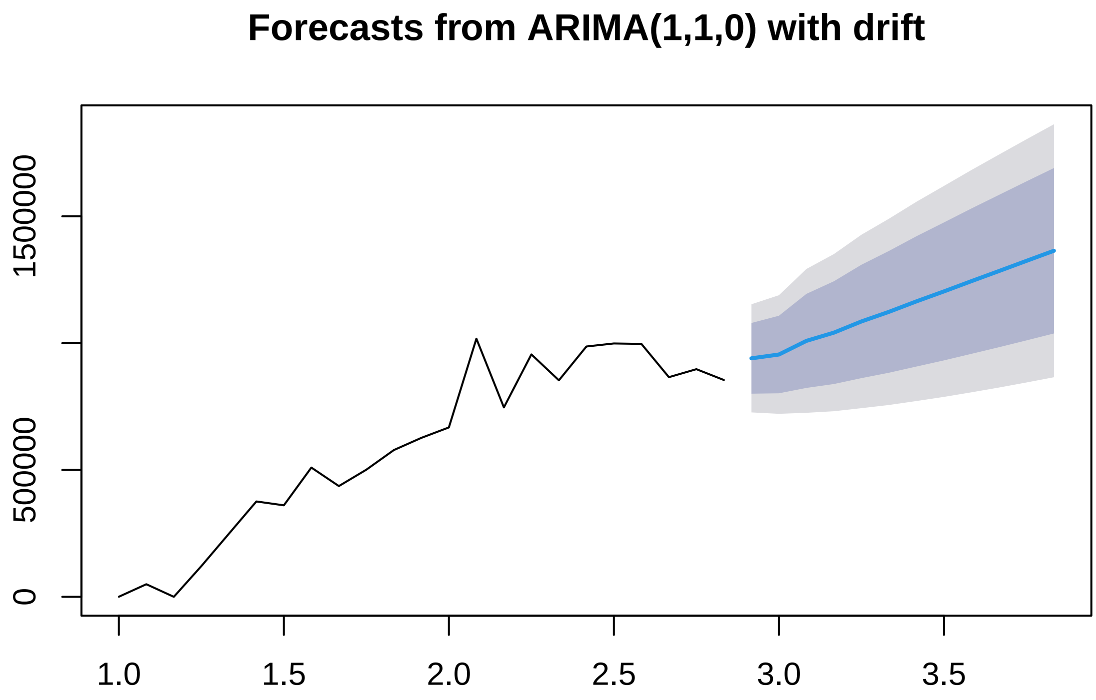
```
# Aggregate sales data by month
monthly_sales <- df_all_unique2 %>% drop_na(price) %>%
  group_by(Month = floor_date(order_purchase_timestamp, "month")) %>%
  summarise(Total_Sales = sum(price))
```

- Time Series Forecasting using ARIMA

```
sales_ts <- ts(monthly_sales$Total_Sales, frequency = 12)
arima_model <- auto.arima(sales_ts)
forecasted_sales <- forecast(arima_model, h = 12) # Forecasting next 12 months
```

- Plotting the forecast

```
plot(forecasted_sales)
```



- **Visualize the findings and reporting it:**
  - **The findings:**
    * The company is on a growing scale since the beginning.
    * The price & quantity sales seasonality peak is in the fourth quarter of the year.
    * Most selling categories are related to home furniture, beauty products, and sports.
    * The highest states in purchasing value are RJ, MG, and PR.
    * The payments value distribution are between 50 − 200.
    * The company's rating are normally, but the order delivery delay have a high impact on the low ratings.
  - **The Recommendations:**
    * Develop a main dashboard with the most selling categories in the company user's app.
    * Prioritize the low-price items in the search feature in the user's app, to enhance the customer experience.
    * Minimize the order delivery duration, to enhance the customer's reviews score or increase the estimated delivery time to develop honesty with the customers.
    * Cooperation with the SEO team to increase the advertising in the highest states and cities in purchasing, to increase the customer base.
  - Link for the repository on GitHub: https://github.com/a7mdNasrr/Olist-Brazilian-E-commerce-Analysis-Project