In this Part of the Project,we are Merging the 8 datasets and performing an EDA anlaysis on the data which was taken from kaggle source provided by Olist Ecommerce Store.

https://www.kaggle.com/olistbr/brazilian-ecommerce (https://www.kaggle.com/olistbr/brazilian-ecommerce)

```
import numpy as np
import pandas as pd
# Seaborn and matplotlib
import seaborn as sns
import matplotlib.pyplot as plt
# Import all of the libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
import numpy as np

# Others
import warnings
```

## Importing and Joining all datasets

```
# File location and type
file_location = "/FileStore/tables/olist_customers_dataset.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

# display(df)


df.createTempView('table_1')
```

Datasets which were uploaded to dfbs file system

```
# File uploaded to /FileStore/tables/olist_customers_dataset.csv
#  File uploaded to /FileStore/tables/olist_order_items_dataset.csv
#  File uploaded to /FileStore/tables/olist_order_payments_dataset.csv
#  File uploaded to /FileStore/tables/olist_order_reviews_dataset.csv
#  File uploaded to /FileStore/tables/olist_orders_dataset.csv
#  File uploaded to /FileStore/tables/olist_products_dataset.csv
#  File uploaded to /FileStore/tables/olist_sellers_dataset.csv
#  File uploaded to /FileStore/tables/product_category_name_trnslation.csv
# /FileStore/tables/olist_geolocation_dataset.csv
```

```
# File location and type
file_location = "/FileStore/tables/olist_order_items_dataset.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df2 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

# display(df2)
```

```
# File location and type
file_location = "/FileStore/tables/olist_orders_dataset.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df3 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

# display(df3)
```

```
df4 = df3.join(df2, ["order_id"])

# display(df4)
```

```
df5 = df4.join(df, ["customer_id"])

# display(df5)
```

```
# File uploaded to /FileStore/tables/olist_order_payments_dataset.csv
# File location and type
file_location = "/FileStore/tables/olist_order_payments_dataset.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df6 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

# display(df6)


df7 = df6.join(df5, ["order_id"])

# display(df7)



# File uploaded to /FileStore/tables/olist_order_reviews_dataset.csv
# File location and type
file_location = "/FileStore/tables/olist_order_reviews_dataset.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df8 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

# display(df8)



df9 = df8.join(df7, ["order_id"])

# display(df9)
```

```
# File uploaded to /FileStore/tables/olist_products_dataset.csv

# File location and type
file_location = "/FileStore/tables/olist_products_dataset.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df12 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

# display(df12)



df13 = df12.join(df9, ["product_id"])

# display(df13)



# File uploaded to /FileStore/tables/olist_sellers_dataset.csv

# File location and type
file_location = "/FileStore/tables/olist_sellers_dataset.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df14 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

# display(df14)


df15 = df14.join(df13, ["seller_id"])

# display(df15)
```

```
# /FileStore/tables/product_category_name_trnslation.csv


# File location and type
file_location = "/FileStore/tables/product_category_name_translation.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df16 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

# display(df16)



df17 = df16.join(df15, ["product_category_name"])

display(df17)
```

| product_category_name | product_category_name_english | seller_id | seller_zip_code_prefix | seller_city | seller_state | product_id | product_name_lenght | product_description_lenght | |
|---|---|---|---|---|---|---|---|---|---|
| perfumaria | perfumery | 325f3178fb58e2a9778334621eecdbf9 | 6790 | taboao da serra | SP | 6782d593f63105318f46bbf7633279bf | 30 | 487 | 1 |
| esporte_lazer | sports_leisure | a17f621c590ea0fab3d5d883e1630ec6 | 18055 | sorocaba | SP | e95ee6822b66ac6058e2e4aff656071a | 52 | 1153 | 1 |
| utilidades_domesticas | housewares | 1b4c3a6f53068f0b6944d2d005c9fc89 | 88730 | sao ludgero | SC | e9a69340883a438c3f91739d14d3a56d | 60 | 1912 | 5 |
| telefonia | telephony | ea8482cd71df3c1969d7b9473ff13abc | 4160 | sao paulo | SP | 036734b5a58d5d4f46b0616ddc047ced | 58 | 751 | 5 |
| cama_mesa_banho | bed_bath_table | d1c281d3ae149232351cd8c8cc885f0d | 14940 | ibitinga | SP | b1434a8f79cb3528540d9b21e686e823 | 57 | 184 | 1 |

Showing the first 1000 rows.

```
df17.printSchema()
# df17-- can take as base dataset WHERE ALL DATASETS COMBINED
```

```
root
 |-- product_category_name: string (nullable = true)
 |-- product_category_name_english: string (nullable = true)
 |-- seller_id: string (nullable = true)
 |-- seller_zip_code_prefix: integer (nullable = true)
 |-- seller_city: string (nullable = true)
 |-- seller_state: string (nullable = true)
 |-- product_id: string (nullable = true)
 |-- product_name_lenght: integer (nullable = true)
 |-- product_description_lenght: integer (nullable = true)
 |-- product_photos_qty: integer (nullable = true)
 |-- product_weight_g: integer (nullable = true)
 |-- product_length_cm: integer (nullable = true)
 |-- product_height_cm: integer (nullable = true)
 |-- product_width_cm: integer (nullable = true)
```

```
|-- order_id: string (nullable = true)
|-- review_id: string (nullable = true)
|-- review_score: string (nullable = true)
|-- review_comment_title: string (nullable = true)
|-- review_comment_message: string (nullable = true)
|-- review_creation_date: string (nullable = true)
```

```
df17.createTempView("table_3")
```

```
from pyspark.sql.functions import unix_timestamp
from pyspark.sql.functions import from_unixtime, to_date, date_format,datediff
from pyspark.sql.functions import col, expr, when
```

## Converting the Timestamp column to Correct Date Format for Respective Analysis

```
#CONVERTING DATE COLUMNS TO MONTHS(YYYY-MM) FORMAT
dfs = df17.withColumn("order_purchase_timestamp",date_format('order_purchase_timestamp','yyyy-MM').alias('month'))
dfs = dfs.withColumn("order_delivered_customer_date",date_format('order_delivered_customer_date','yyyy-MM'))
dfs = dfs.withColumn("order_delivered_carrier_date",date_format('order_delivered_carrier_date','yyyy-MM'))
dfs = dfs.withColumn("order_estimated_delivery_date",date_format('order_estimated_delivery_date','yyyy-MM'))
dfs = dfs.withColumn("review_creation_date",date_format('review_creation_date','yyyy-MM'))
dfs = dfs.withColumn("review_answer_timestamp",date_format('review_answer_timestamp','yyyy-MM'))
```

```
#CONVERTING DATE COLUMNS TO YEAR(YYYY-MM-DD) FORMAT
dfs2 = df17.withColumn("order_purchase_timestamp",date_format('order_purchase_timestamp','yyyy-MM-dd').alias('month'))
dfs2 = dfs2.withColumn("order_delivered_customer_date",date_format('order_delivered_customer_date','yyyy-MM-dd'))
dfs2 = dfs2.withColumn("order_delivered_carrier_date",date_format('order_delivered_carrier_date','yyyy-MM-dd'))
dfs2 = dfs2.withColumn("order_estimated_delivery_date",date_format('order_estimated_delivery_date','yyyy-MM-dd'))
dfs2 = dfs2.withColumn("review_creation_date",date_format('review_creation_date','yyyy-MM-dd'))
dfs2 = dfs2.withColumn("review_answer_timestamp",date_format('review_answer_timestamp','yyyy-MM-dd'))
```

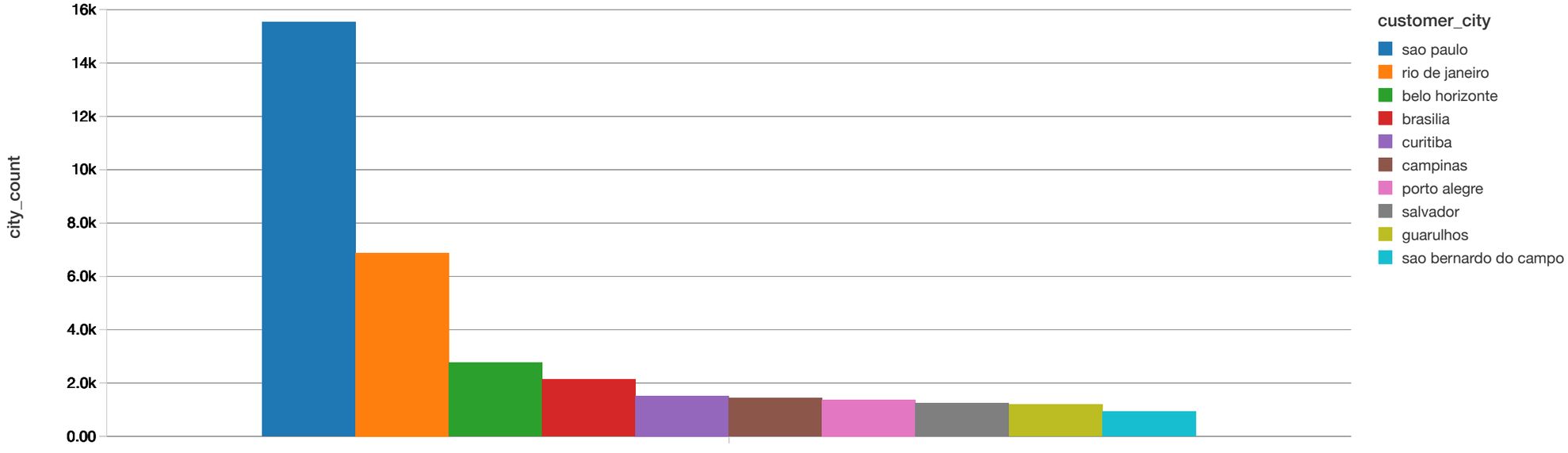
```
dfs.select(col('order_estimated_delivery_date')).show()
#Converted Date Columnn

+---------------------------+
|order_estimated_delivery_date|
+---------------------------+
|                    2017-09|
|                    2017-05|
|                    2018-02|
|                    2018-08|
|                    2017-03|
|                    2017-06|
|                    2018-01|
|                    2018-07|
|                    2018-03|
|                    2018-07|
|                    2018-04|
|                    2018-08|
|                    2018-08|
|                    2018-03|
|                    2018-03|
|                    2018-08|
|                    2018-05|
```

```
|                       2018-05|
|                       2017-09|
|                       2018-03|
+------------------------------+
only showing top 20 rows


dfs.createTempView("table_month")
#Table with Day Format


dfs2.createTempView("table_4")
#Table with Month Format
```

## EXPLORATORY DESCRIPTIVE ANALYSIS

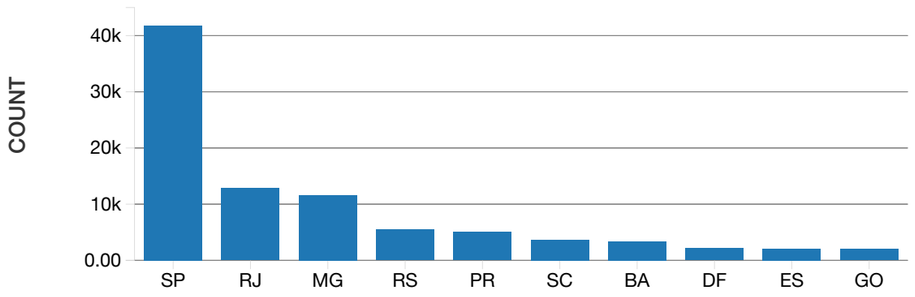
# #CITIES WITH HIGHEST ORDERS

```
%sql

select customer_city, count(customer_city) as city_count from table_1 group by customer_city order by city_count desc limit 10
--From the Plot,Customers from "Sao Paulo" has the Highest Orders Throughout 2017-2018 Sales followed by "rio de janeiro"
```




# ##STATES WITH HIGHEST ORDERS

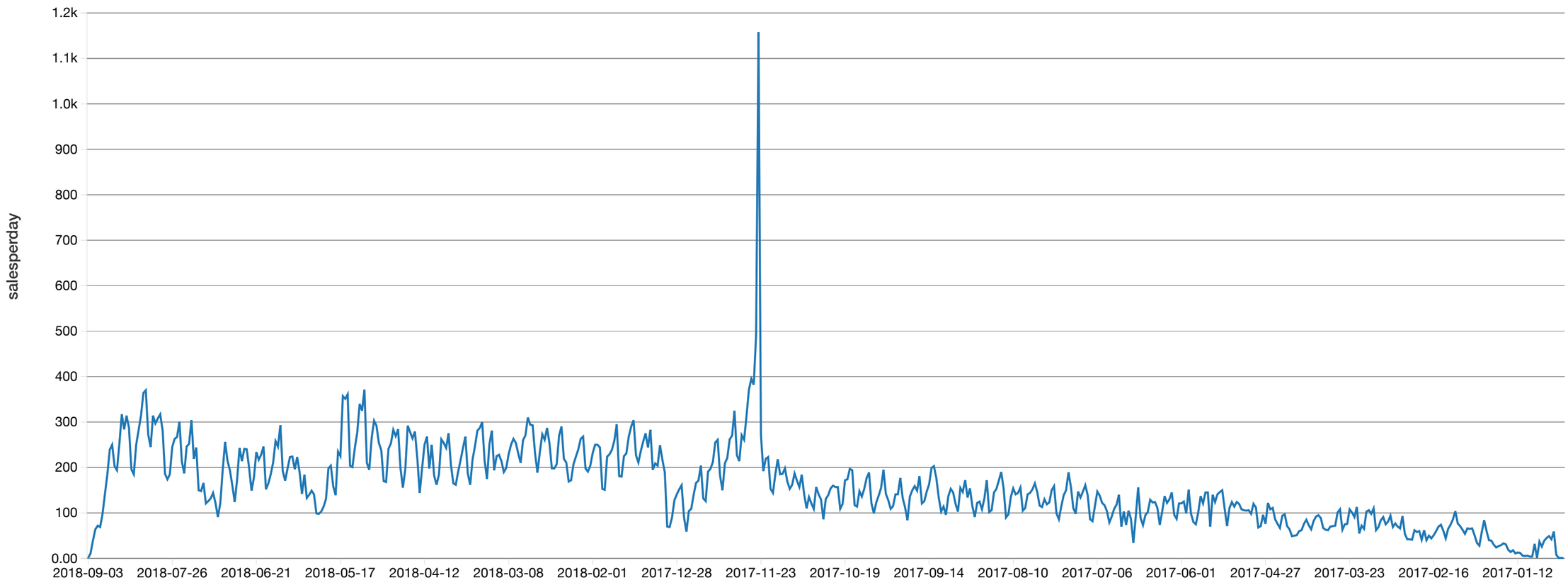
```
%sql
select customer_state as STATE, count(customer_state ) as COUNT from table_1 group by customer_state order by COUNT desc limit 10
```




```
##TREND OF SALES PER DAY FROM 2017-2018 PERIOD
```

# ##TREND OF SALES PER DAY IN 2017-2018 PERIOD

```
%sql
select order_purchase_timestamp, count(distinct(order_id)) as salesperday from table_4 group by order_purchase_timestamp order by order_purchase_timestamp desc
```



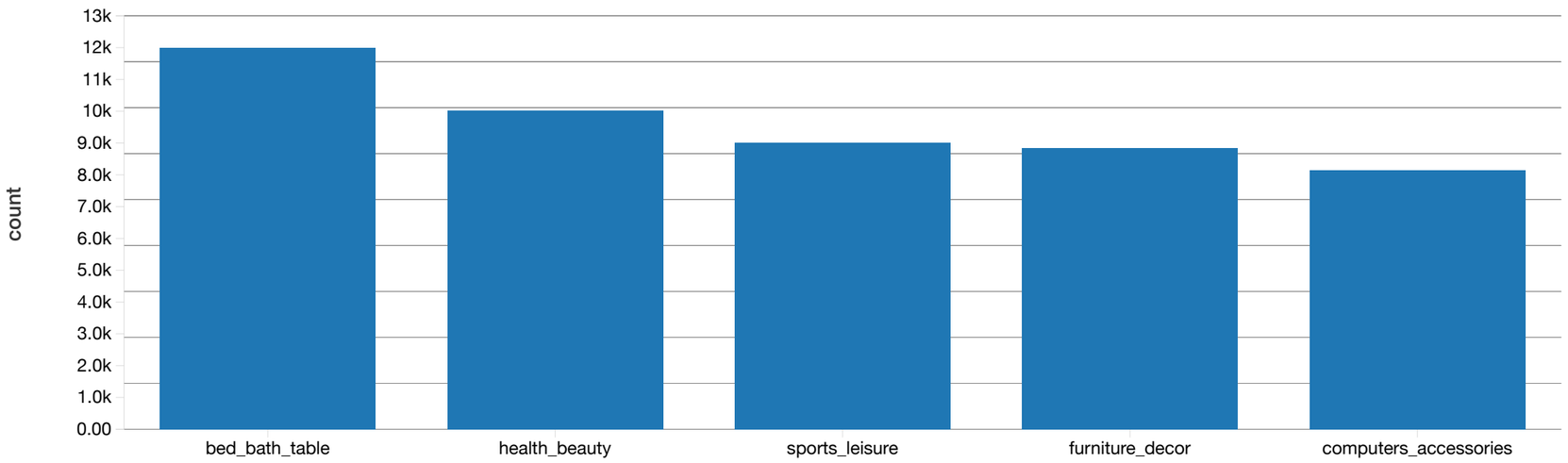

```
# There is a huge spike in Nov 24 due to Black Friday
# -- Sales are weak after Dec 20 (end-year holidays)
# -- There is a spike in products value in Jul 18.
```

```
Command skipped
```

```
##TOP 5 PRODUCT CATEGORIES THAT SOLD MOST
```

# ##TOP 5 PRODUCT CATEGORIES THAT SOLD MOST-

```
%sql
select product_category_name_english, count(product_category_name_english) as count from table_3 group by product_category_name_english order by count desc limit 5

-- # - Bed_Bath_Table type of Product category Leads the most sales follwed by Health_Beauty
```

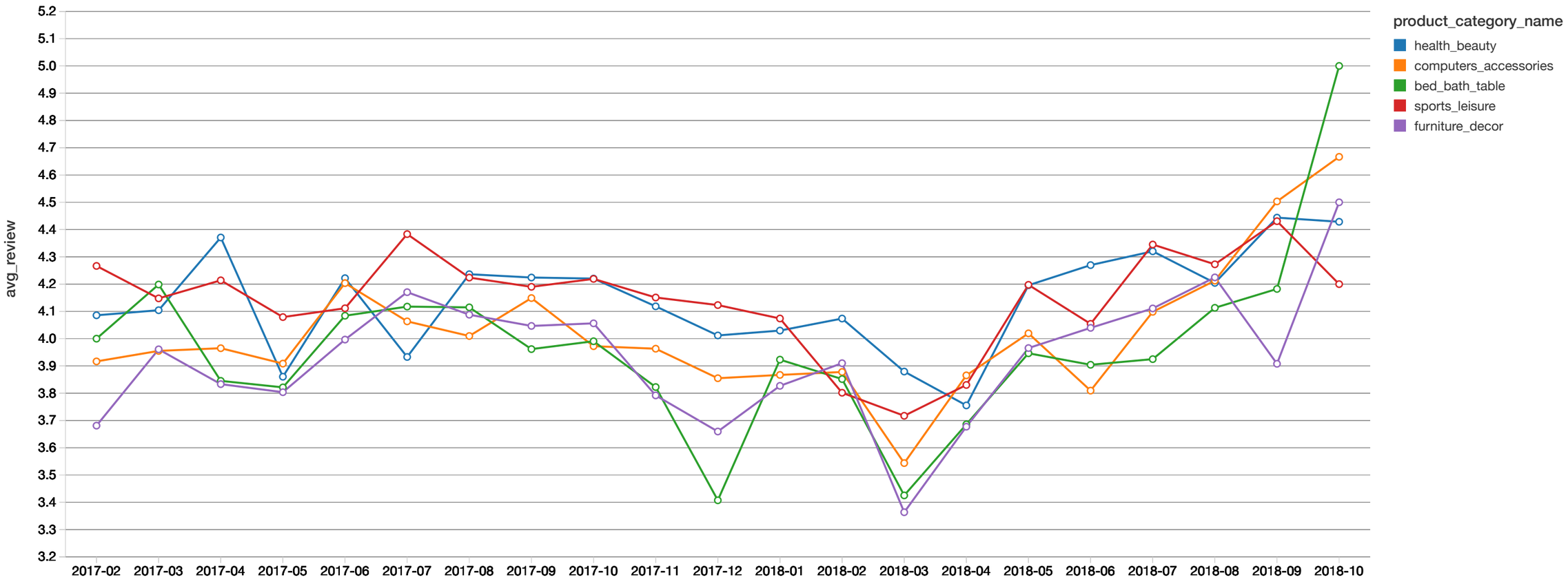



# Selecting Top five Product Categories

```
list_top_products = ['bed_bath_table',
'health_beauty',
'sports_leisure',
'furniture_decor',
'computers_accessories']
dfs_prod = dfs.filter(dfs.product_category_name_english.isin(list_top_products))


dfs_prod.createTempView('dfs_products1')


##PLOT FOR AVERAGE REVIEW FOR TOP FIVE PRODUCT CATEGORIES FOR EACH MONTH THROUGHOUT STHE SALES PERIOD
```

# TREND For AverageReview of  Top Five Categories for Each Month Throughout the Sales

```
%sql
select order_estimated_delivery_date, avg(review_score) as avg_review, product_category_name_english from dfs_products1 where order_estimated_delivery_date >= "2017-02" group by order_estimated_delivery_date,
product_category_name_english order by order_estimated_delivery_date


--From this Plot the Varaition for "Average Reviews"
--The Trend for Popular saled product Bed_bath_table started at 4.3 score and ended at an average review of 5.
--Four out of Five Product Categories hit the least Average Review score at March 2018.--These might be due to various factors.
```

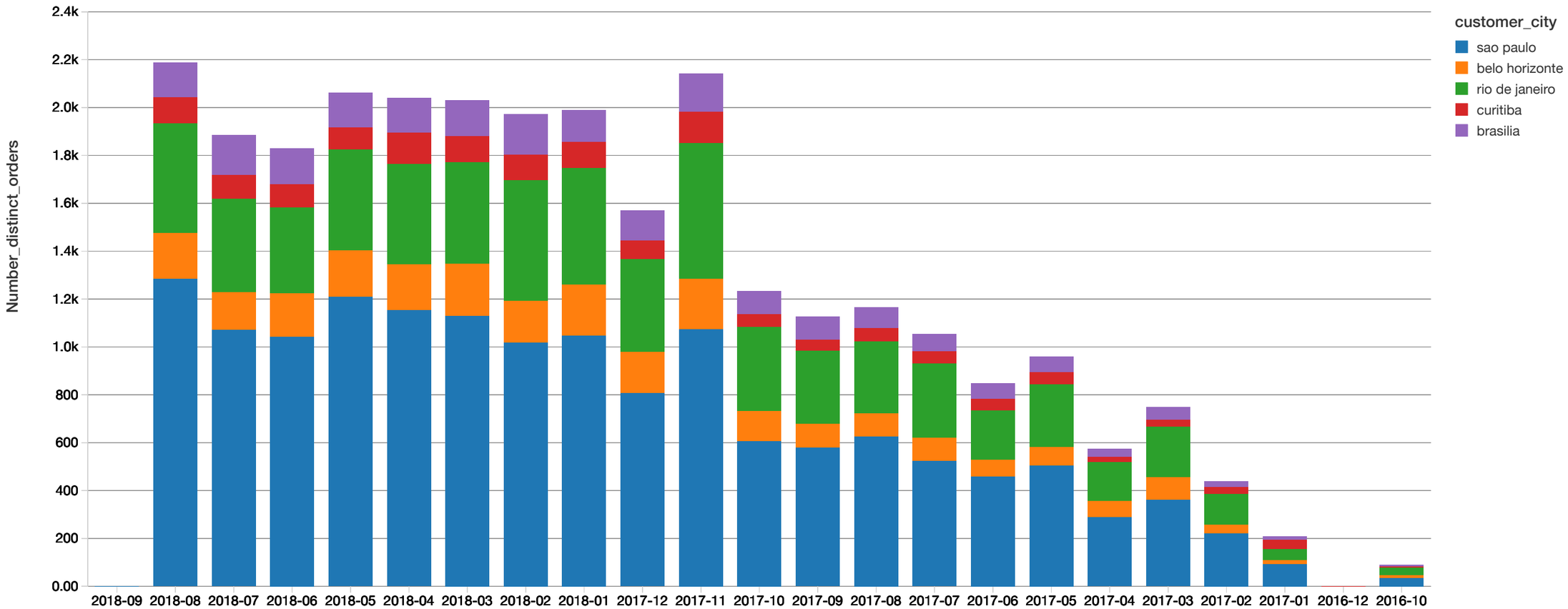



# Selecting Top Cities with Highest Sales

```
l = ['sao paulo',
'rio de janeiro',
'belo horizonte',
'brasilia',
'curitiba']
dfss = dfs.filter(dfs.customer_city.isin(l))


dfss.createTempView('dfssT')


##Plot for top five cites with most orders in each month throughout the sales period
```

# ##Plot for top five cites with most orders in each month throughout the sales period

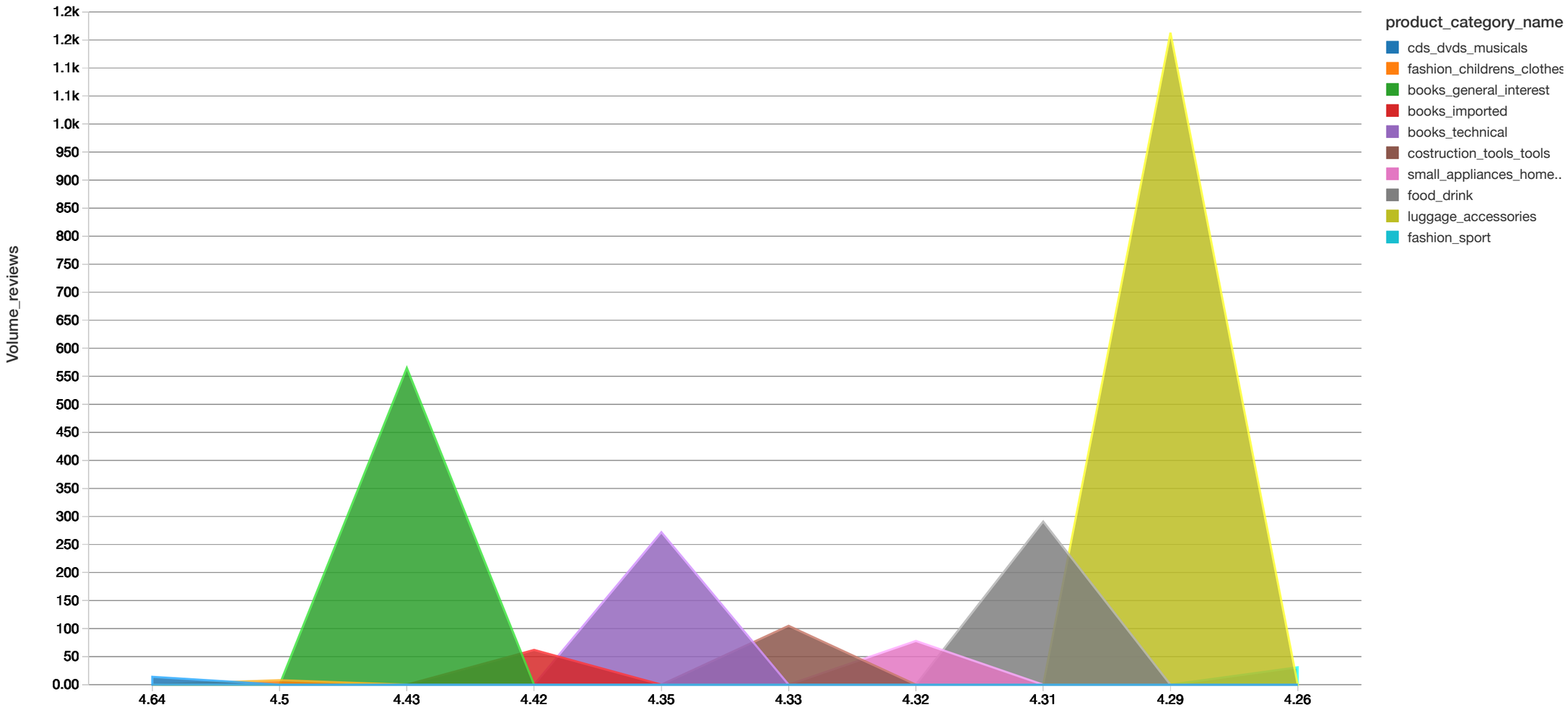
```
%sql
select order_purchase_timestamp as PurchaseTime, count(distinct(order_id)) as Number_distinct_orders, customer_city  from dfssT group by order_purchase_timestamp, customer_city order by order_purchase_timestamp desc

--From the plots,the share of orders for five cities where "Sao Paulo" leads with highest share of orders in every month the next being "Rio De Janeiro"
```




```
##Average reviews for top Product Categories by Volume of reviews
```

# ##Average reviews for top Product Categories by Volume of reviews

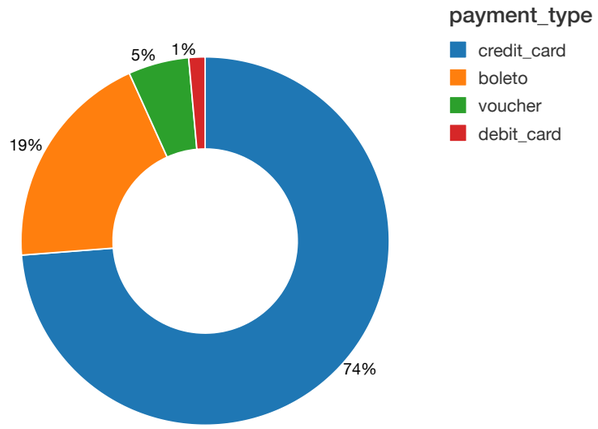
```
%sql
select product_category_name_english, round(avg(review_score), 2) as a , count(review_score) as Volume_reviews from table_4 group by product_category_name_english order by a desc limit 10
-- From the Plot,
-- "Luggage Accessories" has more volume of reviews with an average of 4.29 review score.
-- "Fashion Sport" has the highest review of 4.64 Review Score with least no. of reviews.
```




```
#Percentage of different types of Payments used by Customers to Order Products
```

# #Percentage of different types of Payments used by Customers to Order Products

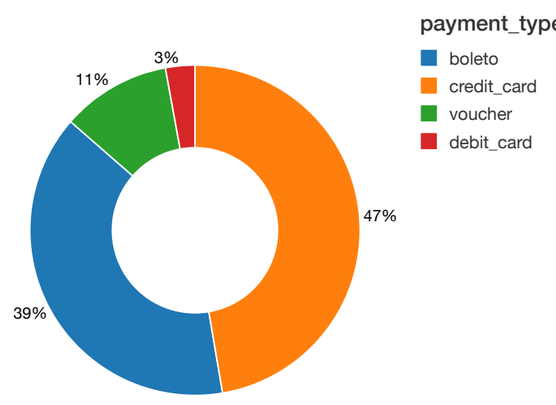
```
%sql
select payment_type, count(payment_type) as c from table_4 group by payment_type order by c desc limit 10
```




# #Percentage of different types of Payments used by Customers to Order Products when paid IN ONE INSTALLMENT

```
#Percentage of different types of Payments used by Customers to Order Products when payed through ONE INSTALLMENT-
# "Credit" card Being the type of Payment most used by the customers and also When Payed in One installment
```

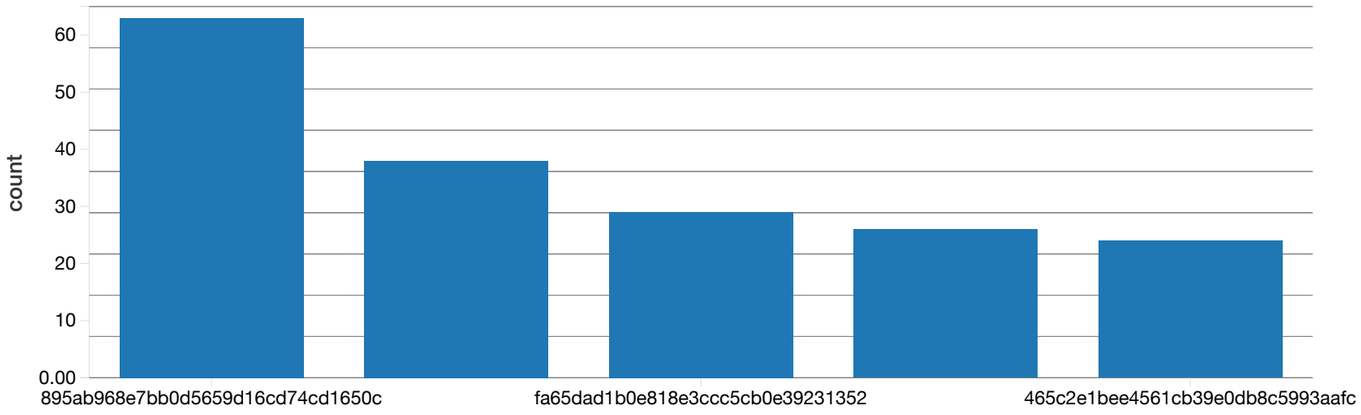
```
%sql
select payment_type, count(payment_type) as c from table_4 where payment_installments=1 group by payment_type   limit 10
```




Plot for max number of Products Ordered in a Single Order

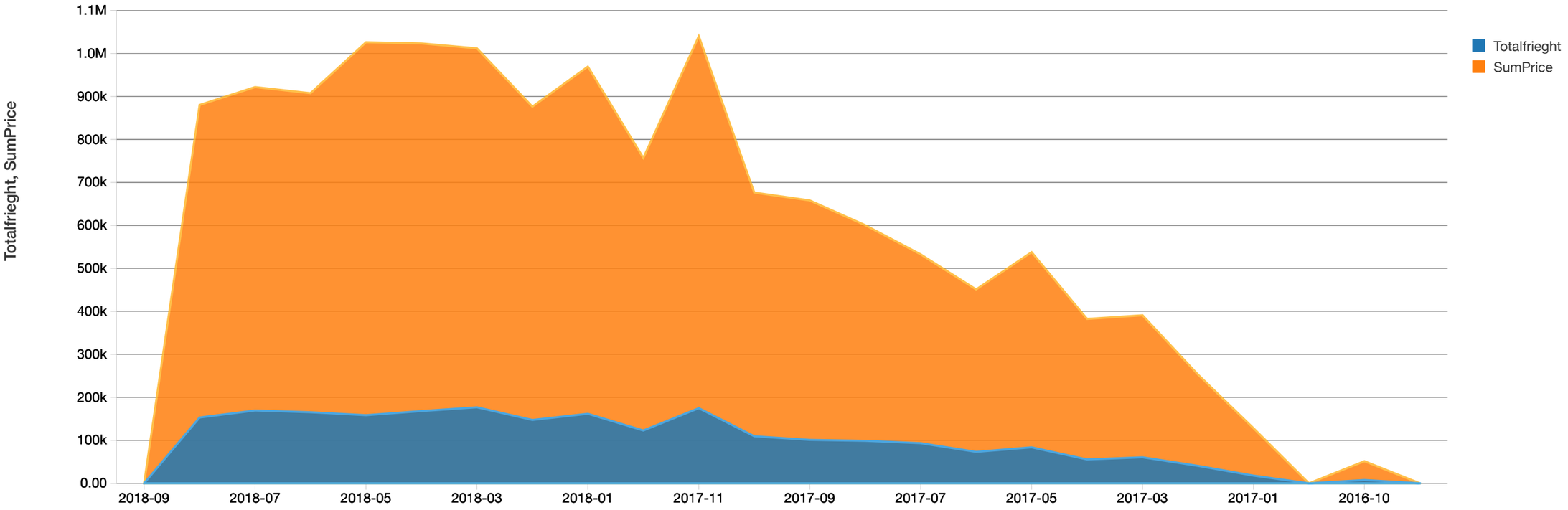
# Untitled

```
%sql
select order_id, count(order_id) as count from table_4  group by order_id  order by count desc LIMIT 5

--  Highest Number of Items in one single order is '63'.
```




## TREND OF FREIGHT VALUE AND PRICE THROUGHOUT THE PERIOD

```
%sql
select order_purchase_timestamp, count(order_purchase_timestamp) as Count_Orders , sum(freight_value) as Totalfrieght, sum(price) as SumPrice from table_month group by order_purchase_timestamp order by
order_purchase_timestamp desc
```

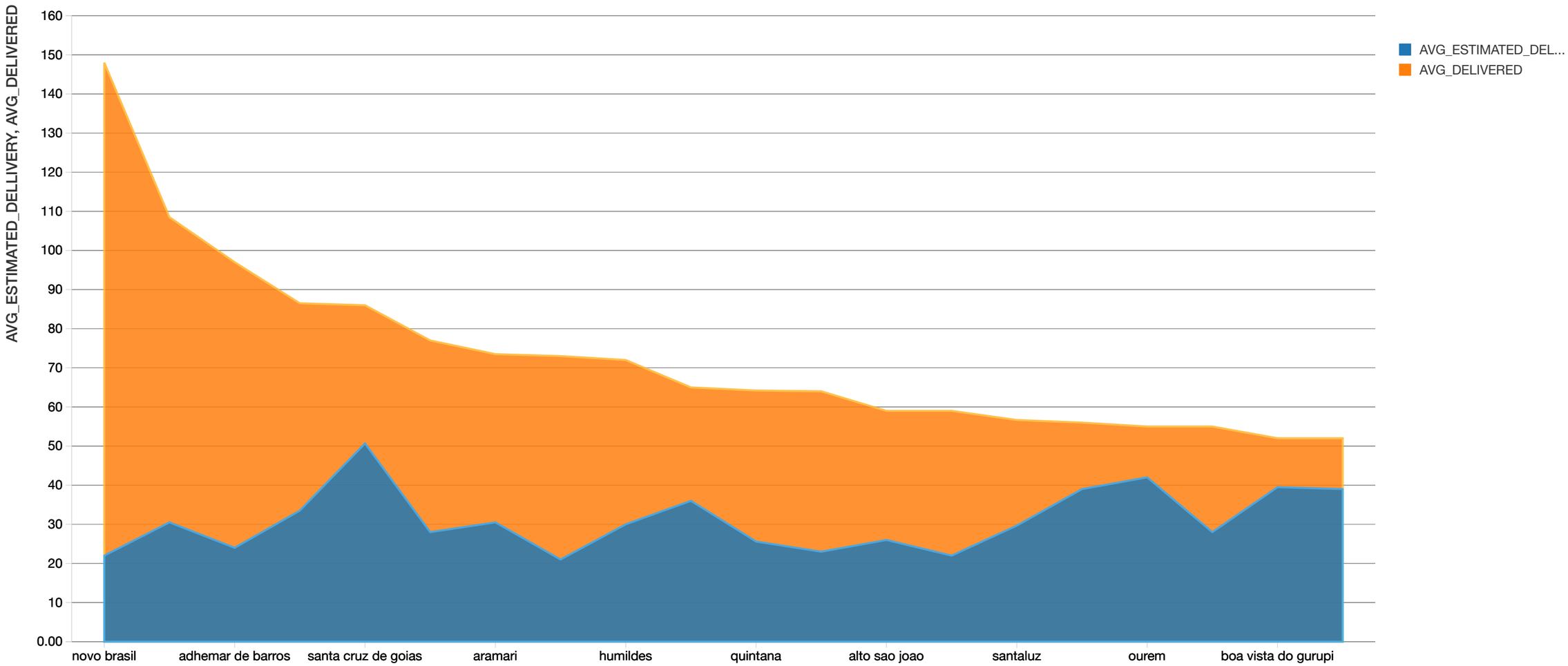


```
dfs3 = dfs2.withColumn(
  "age_in_days_delivery",
  datediff(col("order_delivered_customer_date"), col("order_approved_at"))
)
dfs3 = dfs3.withColumn(
  "estimated_in_days_delivery",
  datediff(col("order_estimated_delivery_date"), col("order_approved_at"))
)


dfs3.createTempView("daystodeliver5")
```

PLOT- TOP 20 CITIES WITH LONGEST AVERAGE DAYS FOR OLIST TO DELIVER AND COMPARING WITH ESTIMATED DELIVERY

# Untitled

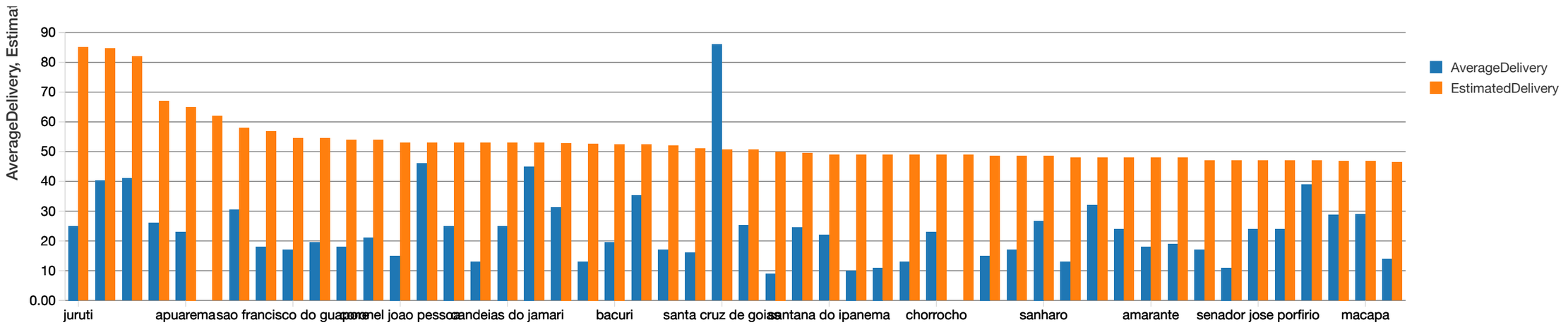
```
%sql
select avg(age_in_days_delivery) as AVG_DELIVERED, avg(estimated_in_days_delivery) as AVG_ESTIMATED_DELLIVERY, customer_city from daystodeliver5 group by customer_city order by AVG_DELIVERED desc limit 20
```

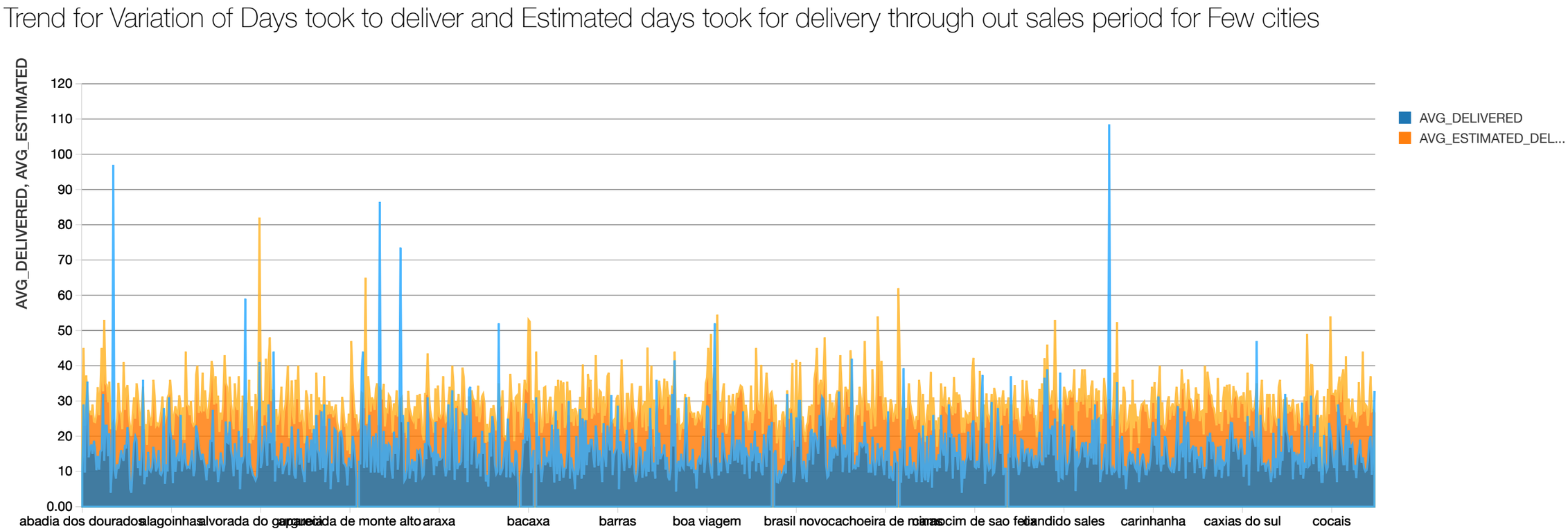


```
##PLOT- TOP 20 CITIES WITH LONGEST ESTIMATED DELIVERY DAYS FOR OLIST TO DELIVER AND COMPARING WITH AVERAGE DAYS
```

## PLOT- TOP 20 CITIES WITH LONGEST ESTIMATED DELIVERY DAYS FOR OLIST TO DELIVER AND COMPARING WITH AVERAGE DAYS

```
%sql
select avg(age_in_days_delivery) as AverageDelivery , avg(estimated_in_days_delivery) as EstimatedDelivery, customer_city from daystodeliver5 group by customer_city order by EstimatedDelivery desc limit 50

-- "juruti" City having Highest Estimated Delivery Days.
```




Showing the first 1000 rows.