PGPDSE FT Capstone Project

Final Report (Group 3)

TEAM MEMBERS

Kishalaya Dutta

Ayush Gurjar

Mungara Sai Kalyan Reddy

Ankit Sharma

Priyanka Srivastava

MENTOR - Mrs. Vidya K

Table of contents

Topic: E - Commerce Prediction and Segmentation

Industry Review:

The primary goal of an e-commerce site is to sell goods and services online. This project deals with developing an e-commerce website for Online product Sale.

There service consists of management of the sales process between shopkeepers and clients, and also includes a customer satisfaction report. The advantages for the shopkeepers is a better market presence and transparent reputation metrics. The driver for the business is to attract more clients and raise the quality of the process. The motivation in this project is to support this effort.

The Olist store is an e-commerce business headquartered in Sao Paulo, Brazil. This firm acts as a single point of contact between various small businesses and the customers who wish to buy their products. Olist is a Brazilian company founded in 2014 that operates a generalist e-marketplace that connects businesses with individual consumers. Unlike Amazon, Olist simply functions as an interface and does not hold any inventory nor sells product of its own.

The dataset we analyze was found on Kaggle. It contains record of 100k orders made on the platform. Each observation on the main csv represents one order and, using adjunct csv provided, we can match those order to additional information such as the type of product, the location of the seller, the buyer, the review given, and much more.

Using this dataset, we aim to optimize the delivery times, and according to this we can predict the future sales of the products using machine learning algorithms.

Project Statement

The analysis behind this business case investigation is to focused on supporting Olist's business objectives. Attracting more shopkeepers by enhancing the service and attracting more end-customers through a broader product spectrum and higher satisfaction. Part of this effort is also to investigate expanding business services to include logistics- and warehousing.

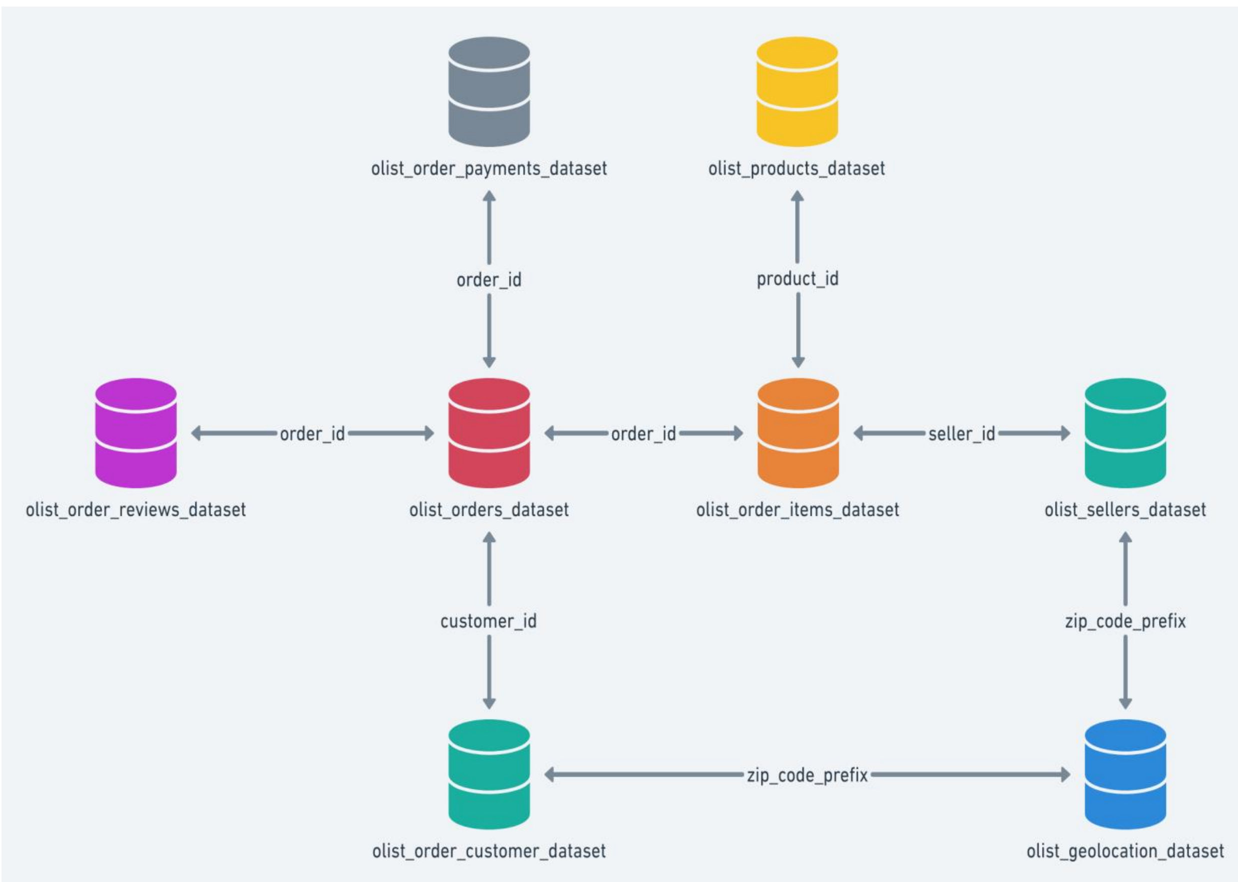
The main aim of this project is to build a model which will help in predicting future sales of different products of different categories. With this information, the company can make better decisions to mitigate losses in the future by releasing the new products based on target audience.

Hence in this project we will be using linear regression models to predict future sales and clustering techniques - unsupervised machine learning for customer segmentation, also we will be identifying the bestselling category in the dataset, so that if the company wants to expand in a particular segment, they can target that and improve their business.

We will also perform RFM Analysis, a marketing technique used to quantitatively rank and group customers based on the recency, frequency and monetary total of their recent transactions to identify the best customers and perform targeted marketing campaigns.

Dataset and Data Dictionary

The dataset contains 99441 rows and 52 columns, and the size of the data set is 126 MB. The dataset is divided into 9 csv files, each containing different information of the business and therefore, by merging relevant tables with the help of given schema. The missing value was imputed with the use of statistical method



Schema of Brazilian Olist E-Commerce Dataset

Dataset details after merging all the required data frames and changing the data types according to requirement. We have also changed data types where ever required, like we converted all date objects to datatime64 format.

Columns Name	Datatype
order_id	object
customer_id	object
order_status	object
order_purchase_timestamp	datetime64
order_approved_at	object
order_delivered_carrier_date	datetime64
order_delivered_customer_date	datetime64
order_estimated_delivery_date	datetime64
product_id	object
order_item_id	int64
product_width_cm	float64
seller_id	object
shipping_limit_date	object
price	float64
freight_value	float64
product_category_name	object
product_name_lenght	float64
product_description_lenght	float64
product_photos_qty	float64
product_weight_g	float64
product_length_cm	float64
product_height_cm	float64

Columns Name	Datatype
recency	int64
frequency	int64
Total Price	int64

Distribution of missing values for each columns:

Columns	Count of missing values	%
price	102425	0
freight_value	102425	0
product_category_name	100965	1.42
product_name_lenght	100965	1.42
product_description_lenght	100965	1.42
product_photos_qty	100965	1.42
product_weight_g	102409	0.01
product_length_cm	102409	0.01
product_height_cm	102409	0.01
product_width_cm	102409	0.01

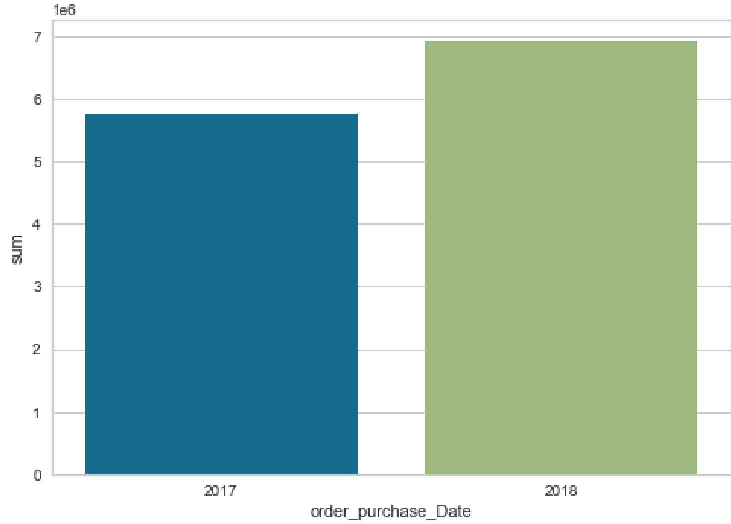
Columns	Count of missing values	%
order_id	102425	0
customer_id	102425	0
order_status	102425	0
order_purchase_timestamp	102425	0
order_approved_at	102411	0.01
order_delivered_carrier_date	101397	1.05
order_delivered_customer_date	100195	2.1
order_estimated_delivery_date	102425	0
order_item_id	102425	0
product_id	102425	0

There are negligible null values

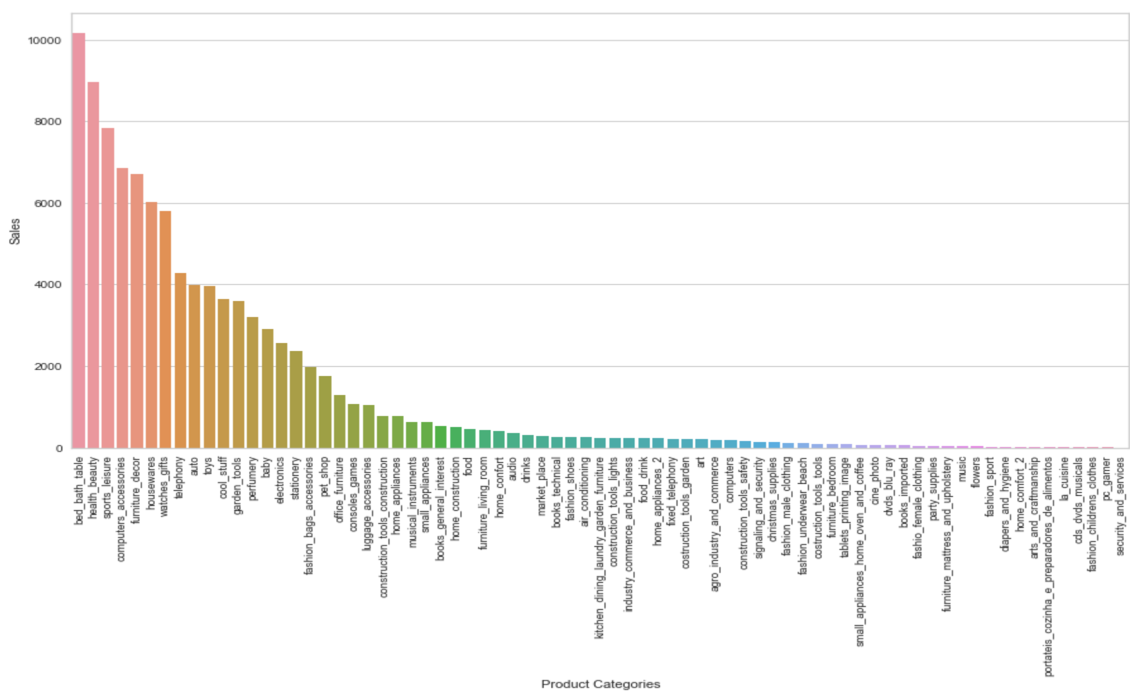
Count of categorical Data type: 8 Count of Datetime Data type: 4 Count of numeric Data type: 18

Exploratory Data Analysis

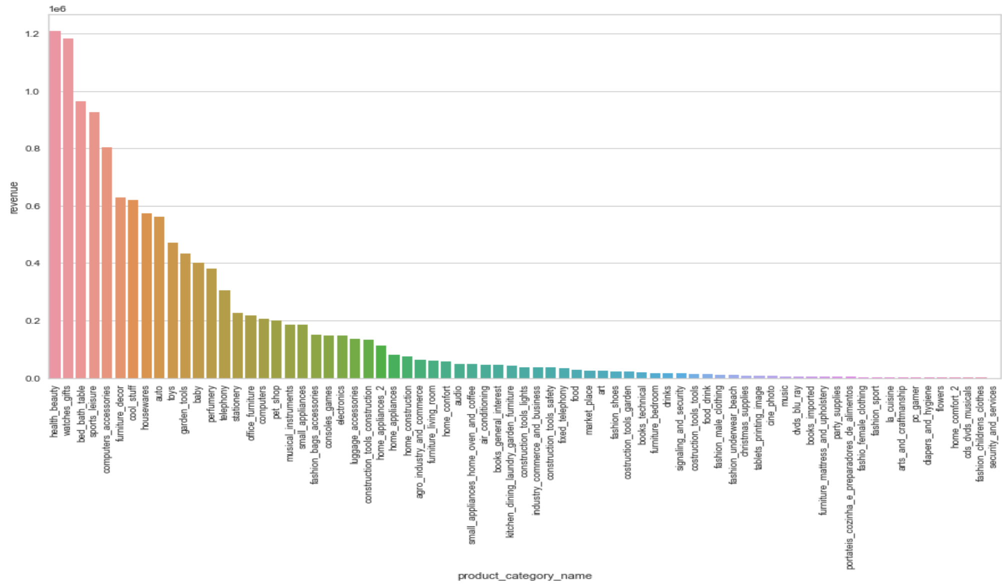
Bi-variate and Multivariate Analysis



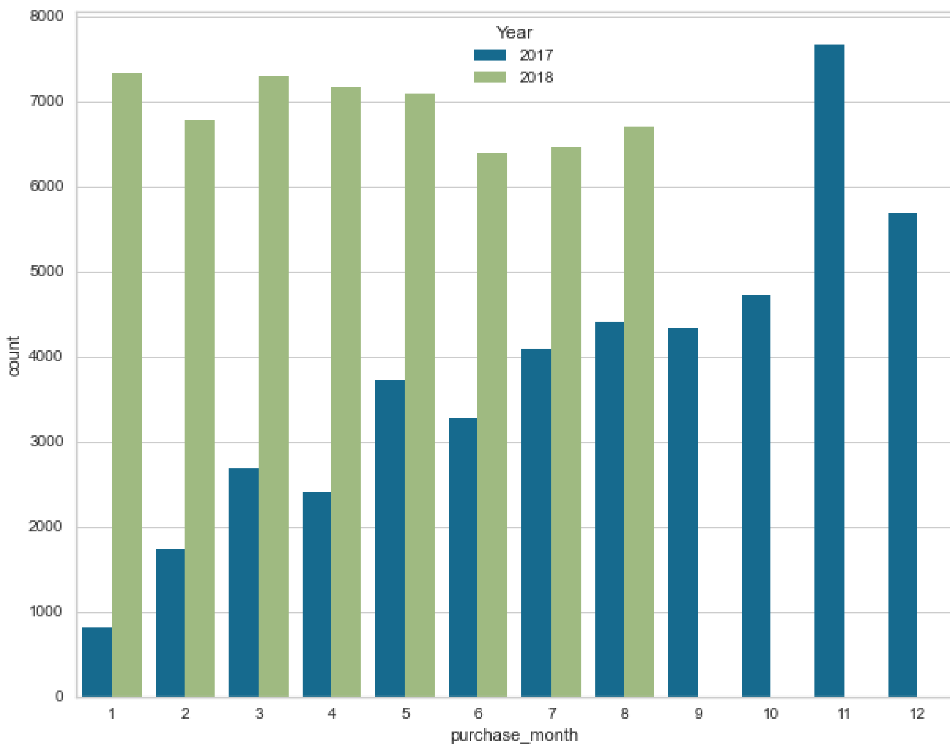
The revenue generated in 2017 is 5769410.72 Brazilian reals whereas in 2018 for 8 months is 6928048.98 Brazilian reals.



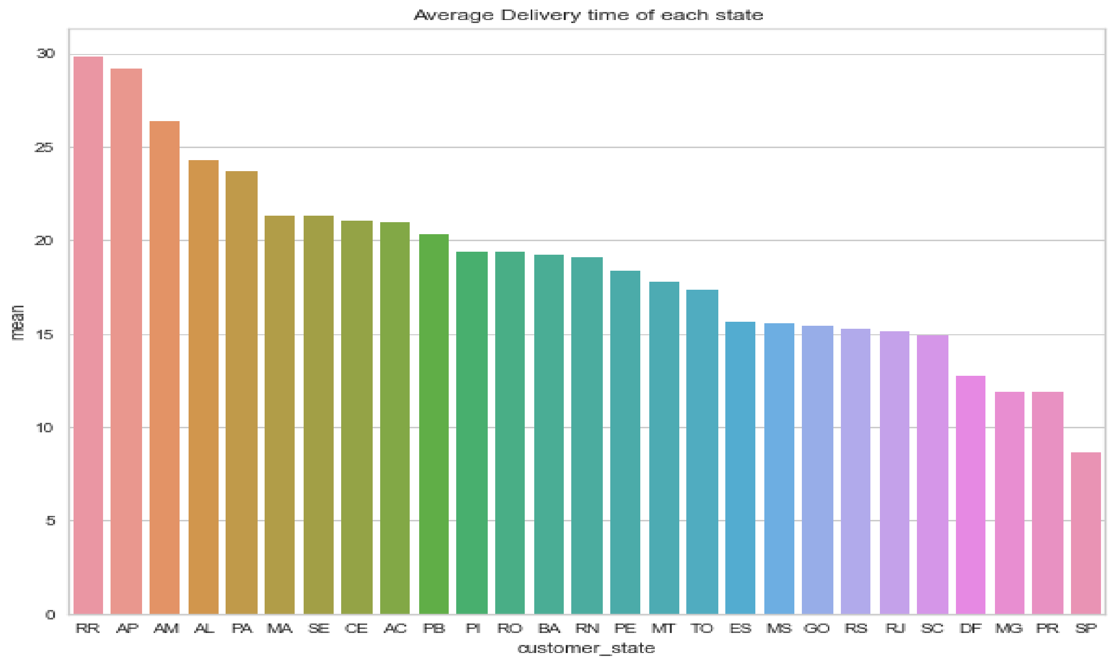
From the above barplot, bed_bath_table product category has got most sales and security_and_services has the least sales.



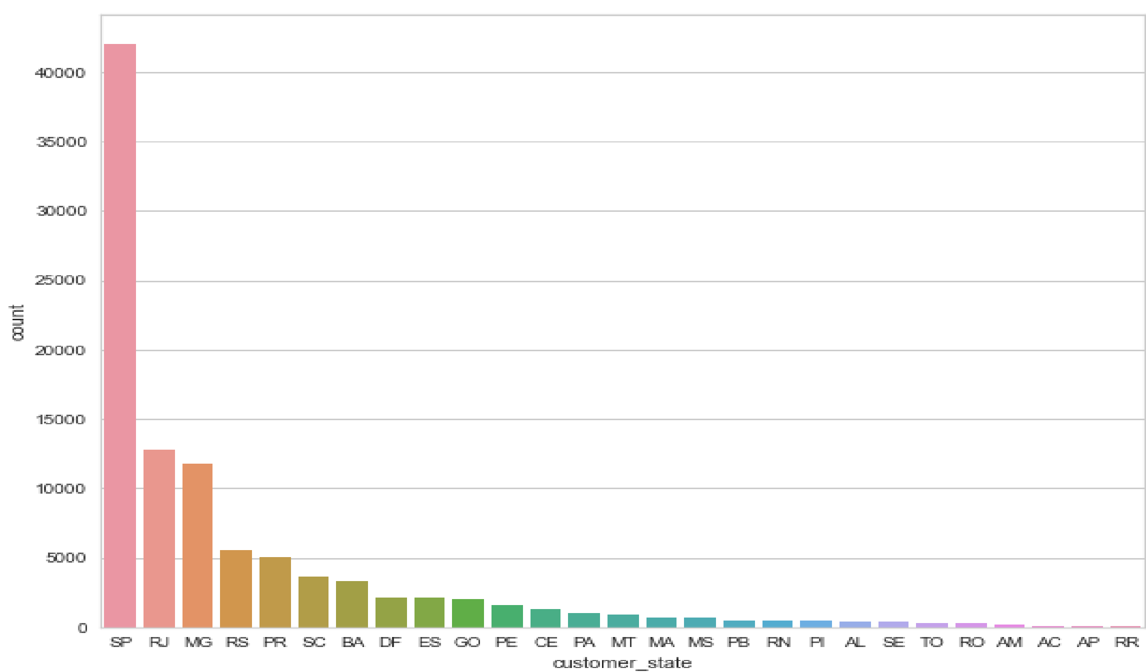
From the barplot, health_beauty category has produced the highest revenue and security_and_services has produced least revenue.



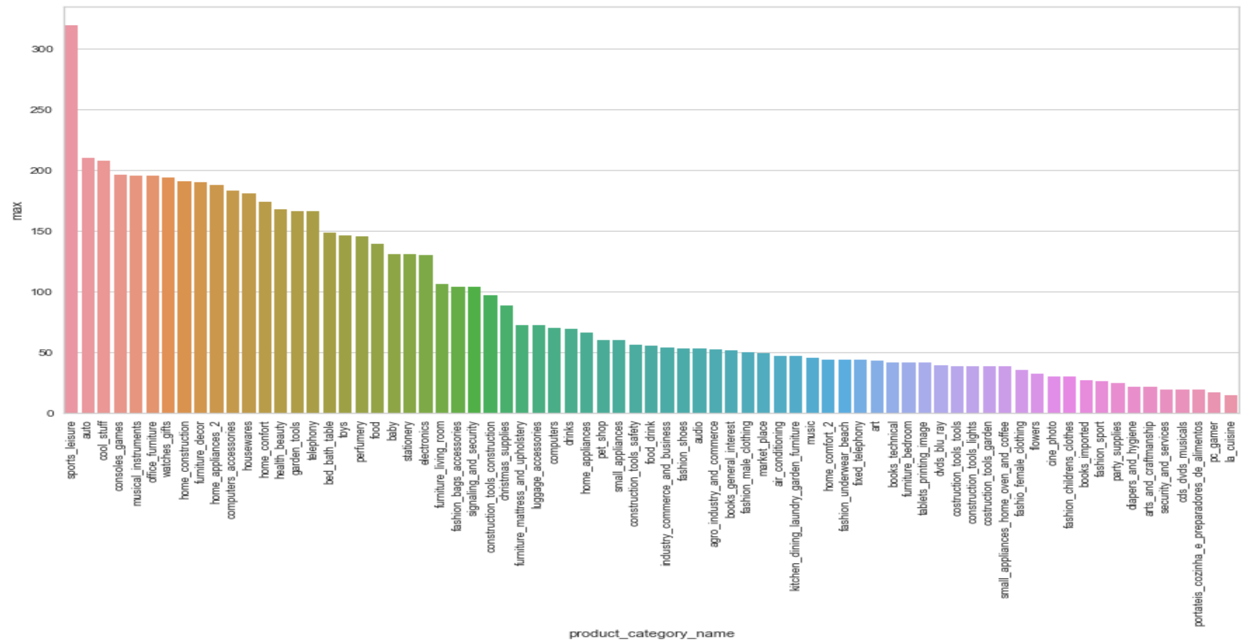
From the Monthly Sales chart, Most Sales happened in 2017 November and least sales happened in 2017 January.



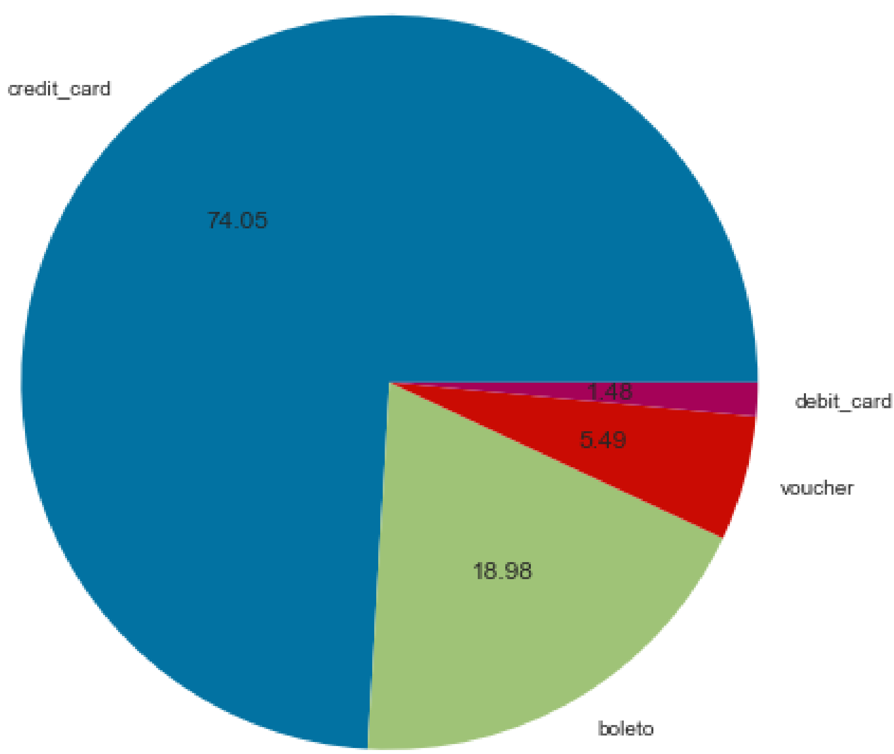
From the Plot, We observe that for RR state average delivery time of any product category is more and for SP state average delivery time of any product category is less.



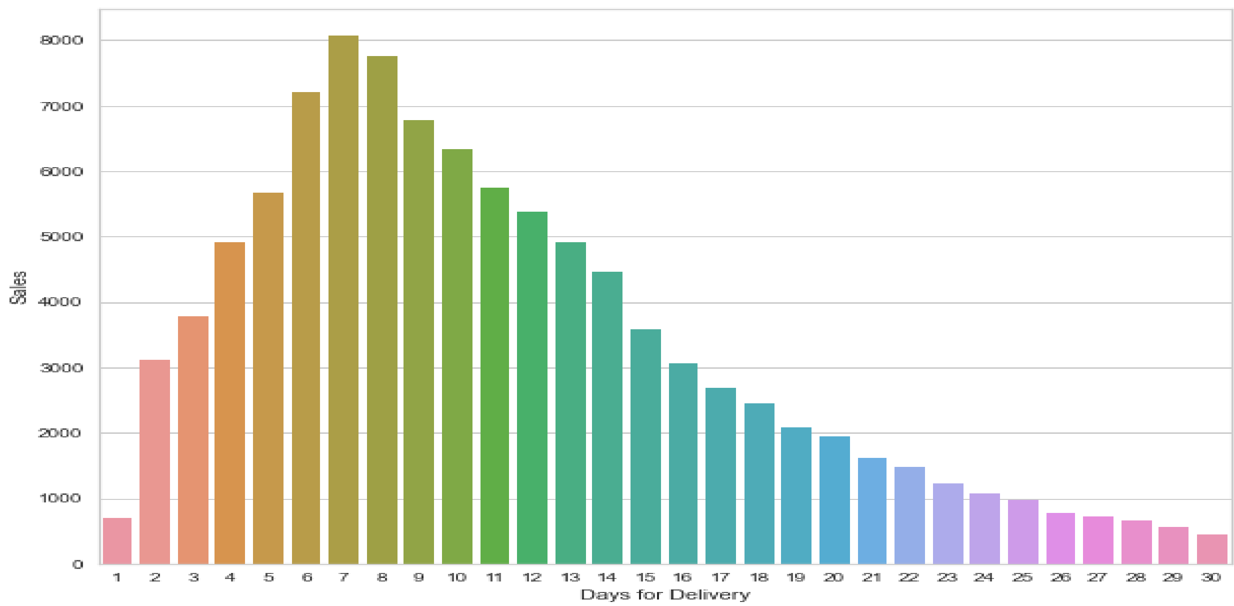
From the State wise Sales and Average Delivery Days, the state which has the least delivery time produces the highest sales whereas the state which has highest delivery time produces the least sales count.



From the Barplot, We observe that sports_leisure category has the highest max delivery time and la_cuisine has the least max delivery time.

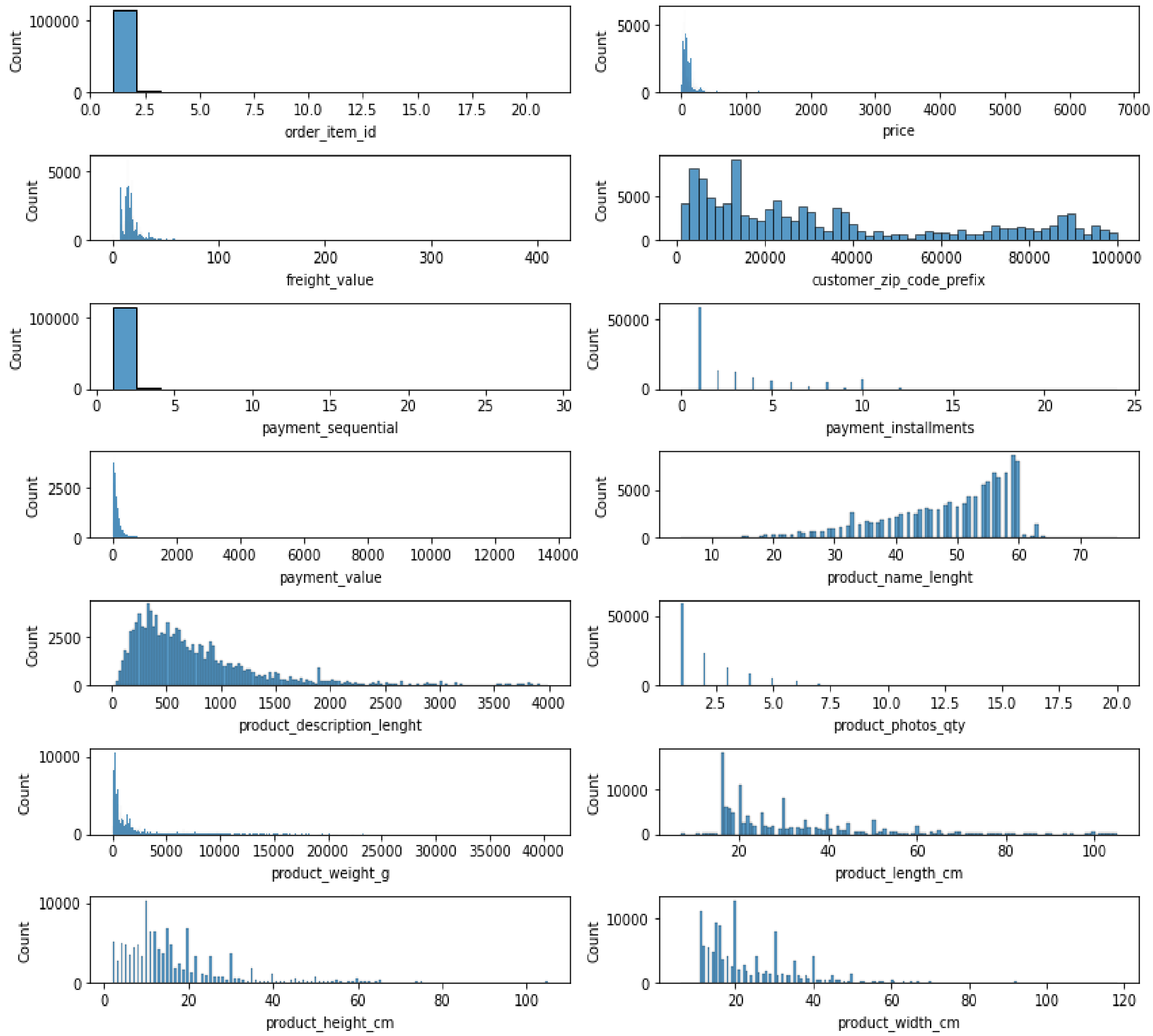


From the pie chart, it is observed that 73 percent of the transactions are done through credit card.



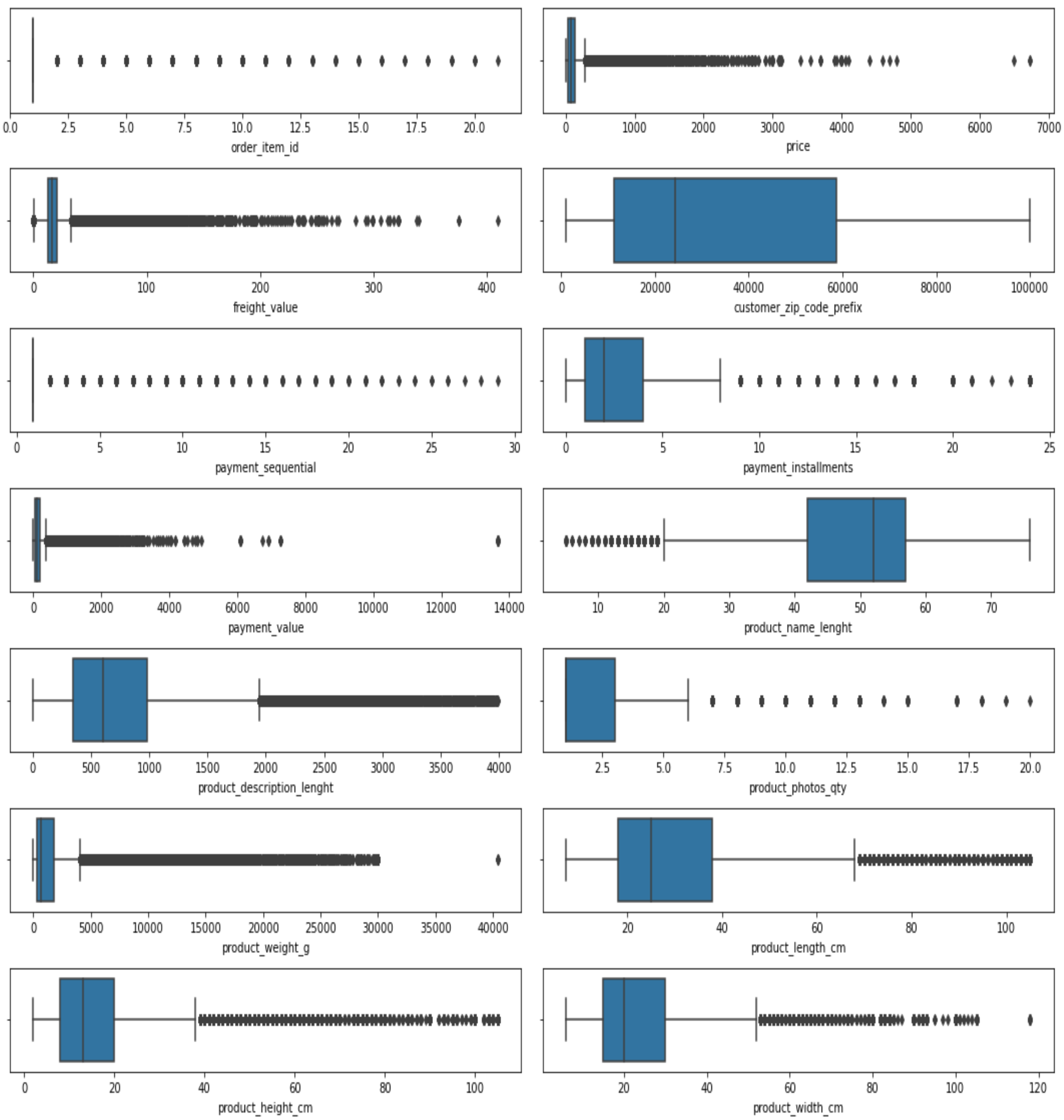
From the above plot, Most no of sales are happening when the average delivery days are around 7-8 days.

Distribution of variables



Most of the variables are not normally distributed.

Presence of outliers



For now, we are considering outliers for our initial model.

Handling Null Values

In [33]:
```
order_orderitems_products_merged.isna().sum()/len(order_orderitems_products_merged)*100
```

Out[33]:
```
order_id                         0.000000
customer_id                      0.000000
order_status                     0.000000
order_purchase_timestamp         0.000000
order_approved_at                0.013669
order_delivered_carrier_date     1.003661
order_delivered_customer_date    2.177203
order_estimated_delivery_date    0.000000
product_id                       0.000000
order_item_id                    0.000000
seller_id                        0.000000
shipping_limit_date              0.000000
price                            0.000000
freight_value                    0.000000
product_category_name            1.425433
dtype: float64
```

In [35]:
```
order_orderitems_products_merged['order_approved_at'].fillna(method='ffill',inplace=True)
```

In [36]:
```
order_orderitems_products_merged['order_delivered_carrier_date'].value_counts()
```

Out[36]:
```
2018-05-09 15:48:00    47
2018-05-10 18:29:00    32
2018-05-07 12:31:00    21
2018-05-17 15:06:00    18
2018-08-15 12:53:00    18
                       ..
2018-01-03 17:15:11     1
2017-12-13 13:26:52     1
2018-02-06 14:54:43     1
2018-04-07 01:18:36     1
2018-03-09 22:11:59     1
Name: order_delivered_carrier_date, Length: 81017, dtype: int64
```

In [37]:
```
order_orderitems_products_merged['order_delivered_carrier_date'].fillna(method='ffill',inplace=True)
```

In [38]:
```
order_orderitems_products_merged['order_delivered_customer_date'].fillna(method='ffill',inplace=True)
```

We have used forward fill (ffill) to replace null values.

Statistical Analysis

We will be performing a statistical test to check whether recency, frequency, total payment has any relationship.

In [121]:
```
ss=StandardScaler()
model_build_var.loc[:,:]=ss.fit_transform(model_build_var)
model_build_var.head()
```

Out[121]:

	recency	frequency	total payment
0	-0.829732	-0.159666	-0.054089
1	-0.809857	-0.159666	-0.568546
2	1.992503	-0.159666	-0.336345
3	0.561511	-0.159666	-0.535685
4	0.342887	-0.159666	0.178111

In [122]:
```
# We will be performing a statistical test to check whether recency frequency total payment has any relationship.
```

In [123]:
```
#Null: Correlation coefficient is  significantly equal to zero.
#Alternate: Correlation coefficient is not significantly equal to zero.
```

In [124]:
```
stats.pearsonr(model_build_var_withoutscale['recency'],model_build_var_withoutscale['frequency'])
```

Out[124]: (-0.021796639202701938, 2.904065729544129e-11)

In [125]:
```
# Since pvalue is less than 0.05. We reject NULL.So there is slight relation between recency and frequency.
```

In [126]:
```
stats.pearsonr(model_build_var_withoutscale['frequency'],model_build_var_withoutscale['total payment'])
```

Out[126]: (0.1057993082724789, 6.748330528195386e-230)

In [127]:
```
# Since pvalue is less than 0.05. We reject NULL.So there is slight relation between frequency and total payment.
```

In [128]:
```
stats.pearsonr(model_build_var_withoutscale['recency'],model_build_var_withoutscale['total payment'])
```

Out[128]: (-0.0008942426241804009, 0.7849682789030443)

In [129]:
```
# Since pvalue is greater than 0.05. We reject NULL.So there is no relation between recency and total payment.
```

We have calculated the Pearson correlation coefficient on recency and frequency, frequency and total payment, recency and total payment and since the p-value is less than 0.05, we reject the null hypothesis for the first two tests and conclude there is a slight correlation between the variables whereas for the last test, the p-value is greater than 0.05, we refuse to reject the null hypothesis and conclude that there is no correlation between the variables.

Model building

Sales prediction using linear regression

In [169]:
```
monthlysales_count
```

Out[169]:

	Year	purchase_month	count
0	2017	1	813
1	2017	2	1741
2	2017	3	2674
3	2017	4	2402
4	2017	5	3716
5	2017	6	3275
6	2017	7	4078
7	2017	8	4396
8	2017	9	4330
9	2017	10	4708
10	2017	11	7667
11	2017	12	5686
12	2018	1	7331
13	2018	2	6763
14	2018	3	7289
15	2018	4	7157
16	2018	5	7079
17	2018	6	6382
18	2018	7	6449
19	2018	8	6698
20	2018	9	1

In [175]:
```
linmodel=monthlysales_count.drop(index=20)
```

In [176]:
```
linmodel[['Year','purchase_month']]=linmodel[['Year','purchase_month']].astype(str)
```

In [177]:
```
linmodel_cat=linmodel.select_dtypes(include=object)
```

In [178]:
```
independent_var=pd.get_dummies(linmodel_cat,drop_first=True)
dependent_var=linmodel['count']
```

In [179]:
```
independent_var_const=sm.add_constant(independent_var)
```

We perform the train test split.

In [180]:
```
xtrain,xtest,ytrain,ytest=train_test_split(independent_var_const,dependent_var,random_state=10,test_size=0.3)
```

In [181]:

```
line_reg=sm.OLS(ytrain,xtrain).fit()
line_reg.summary()
```

Out[181]:

OLS Regression Results

Dep. Variable:	count	**R-squared:**	0.960
Model:	OLS	**Adj. R-squared:**	0.826
Method:	Least Squares	**F-statistic:**	7.192
Date:	Fri, 13 May 2022	**Prob (F-statistic):**	0.0656
Time:	14:14:07	**Log-Likelihood:**	-104.62
No. Observations:	14	**AIC:**	231.2
Df Residuals:	3	**BIC:**	238.3
Df Model:	10		
Covariance Type:	nonrobust		

The R2 value is 0.96 while the adjusted R2 value is 0.826, so we can conclude that our model explains roughly 85-90% variation and the model is fit well.

But p value of f-statistic is greater than 0.05, so it means our model not so significant.

	coef	std err	t	P>|t|	[0.025	0.975]
const	1632.2500	727.060	2.245	0.110	-681.581	3946.081
Year_2018	4879.5000	650.303	7.503	0.005	2809.947	6949.053
purchase_month_10	3075.7500	1172.350	2.624	0.079	-655.190	6806.690
purchase_month_11	2.202e-13	4.32e-13	0.510	0.645	-1.15e-12	1.59e-12
purchase_month_12	4053.7500	1172.350	3.458	0.041	322.810	7784.690
purchase_month_2	180.0000	919.667	0.196	0.857	-2746.790	3106.790
purchase_month_3	909.5000	919.667	0.989	0.396	-2017.290	3836.290
purchase_month_4	645.2500	1172.350	0.550	0.620	-3085.690	4376.190
purchase_month_5	1325.5000	919.667	1.441	0.245	-1601.290	4252.290
purchase_month_6	-129.7500	1172.350	-0.111	0.919	-3860.690	3601.190
purchase_month_7	-5.985e-15	9.38e-15	-0.638	0.569	-3.58e-14	2.39e-14
purchase_month_8	186.2500	1172.350	0.159	0.884	-3544.690	3917.190
purchase_month_9	2697.7500	1172.350	2.301	0.105	-1033.190	6428.690

Omnibus:	1.249	**Durbin-Watson:**	1.958
Prob(Omnibus):	0.535	**Jarque-Bera (JB):**	0.092
Skew:	-0.000	**Prob(JB):**	0.955
Kurtosis:	3.398	**Cond. No.**	1.91e+17

The Durbin-Watson is 1.958 which signifies that there is no auto-correlation.

The p value of Jarque Bera is greater than 0.05, so it signifies that the residuals are not normal.

The Condition number > 1000, signifying high multicollinearity.

K-Means Clustering for Customer Segmentation

In [102]:

```
order_orderitems_products_customers_merged.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 101198 entries, 0 to 102092
Data columns (total 18 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   order_id                       101198 non-null  object
 1   customer_id                    101198 non-null  object
 2   order_status                   101198 non-null  object
 3   order_delivered_carrier_date   101198 non-null  datetime64[ns]
 4   order_delivered_customer_date  101198 non-null  datetime64[ns]
 5   order_estimated_delivery_date  101198 non-null  datetime64[ns]
 6   product_id                     101198 non-null  object
 7   order_item_id                  101198 non-null  int64
 8   seller_id                      101198 non-null  object
 9   price                          101198 non-null  float64
 10  freight_value                  101198 non-null  float64
 11  product_category_name          99760 non-null   object
 12  order_purchase_Date            101198 non-null  datetime64[ns]
 13  customer_unique_id             101198 non-null  object
 14  customer_zip_code_prefix       101198 non-null  int64
 15  customer_state                 101198 non-null  object
 16  estimated_no_of_days_delivered 101198 non-null  int64
 17  actual_no_of_days_delivered    101198 non-null  int64
dtypes: datetime64[ns](4), float64(2), int64(4), object(8)
memory usage: 14.7+ MB
```

In [103]:

```
df_copy=order_orderitems_products_customers_merged[['customer_unique_id',
    'order_id',
    'order_status',
    'order_purchase_Date',
    'product_id',
    'order_item_id',
    'price',
    'product_category_name']]
```

In [104]:

```
df_copy.rename(columns={'order_item_id':'quantity'},inplace=True)
```

In [105]:

```
df_copy['order_status'].value_counts()
```

Out[105]:

```
delivered     99911
shipped         673
canceled        229
processing      201
invoiced        182
approved          2
Name: order_status, dtype: int64
```

In [106]:

```
df_copy=df_copy[df_copy['order_status']=='delivered']
```

In [107]:

```
df_copy.head()
```

Out[107]:

	customer_unique_id	order_id	order_status	order_purchase_Date	product_i
0	7c396fd4830fd04220f754e42b4e5bff	e481f51cbdc54678b7cc49136f2d6af7	delivered	2017-10-02	87285b34884572647811a353c7ac498
1	3a51803cc0d012c3b5dc8b7528cb05f7	128e10d95713541c87cd1a2e48201934	delivered	2017-08-15	87285b34884572647811a353c7ac498
2	ef0996a1a279c26e7ecbd737be23d235	0e7e841ddf8f8f2de2bad69267ecfbcf	delivered	2017-08-02	87285b34884572647811a353c7ac498
3	e781fdcc107d13d865fc7698711cc572	bfc39df4f36c3693ff3b63fcbea9e90a	delivered	2017-10-23	87285b34884572647811a353c7ac498
4	af07308b275d755c9edb36a90c618231	53cdb2fc8bc7dce0b6741e2150273451	delivered	2018-07-24	595fac2a385ac33a80bd5114aec74eb

In [108]:

```
# Setting the reference day
df_copy['today']=df_copy['order_purchase_Date'].max()

# Calculating the recency
df_copy['recency']=df_copy['today']-df_copy['order_purchase_Date']
```

```
In [112]: customer_segment=df_copy.groupby(by=['customer_unique_id','order_id'],as_index=False).agg({'order_purchase_Date':'first',
                                                                                              'quantity':'sum',
                                                                                              'total_price':'sum',
                                                                                              'today':'first',
                                                                                              'recency':'first'})
```

```
In [113]: # Changing recency datatype as integer
          customer_segment['recency']=customer_segment['recency'].dt.days
```

```
In [114]: df_customer_segment=customer_segment.groupby(by='customer_unique_id',as_index=False).agg({'recency':['min','max','count'],
                                                                                           'total_price':['sum','mean']})
```

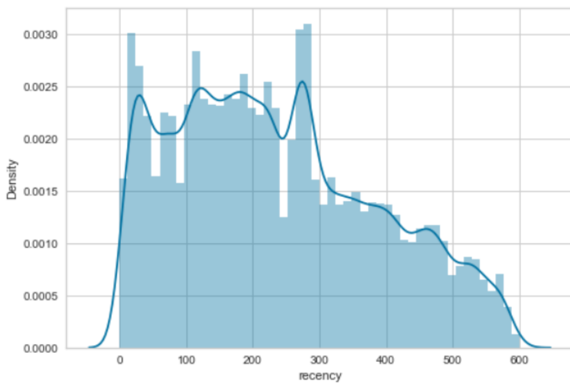
```
In [115]: df_customer_segment.columns=[' '.join(i).strip()   for i in df_customer_segment.columns.values]
```

```
In [116]: df_customer_segment.rename(columns={'recency max': 'days_since_first_order',
              'recency min': 'recency',
              'recency count': 'frequency',
              'total_price sum': 'total payment',
              'total_price mean': 'avg payment'},inplace=True)
```

```
In [117]: model_build_var=df_customer_segment[['recency','frequency','total payment']]
          model_build_var_withoutscale=df_customer_segment[['recency','frequency','total payment']]
```

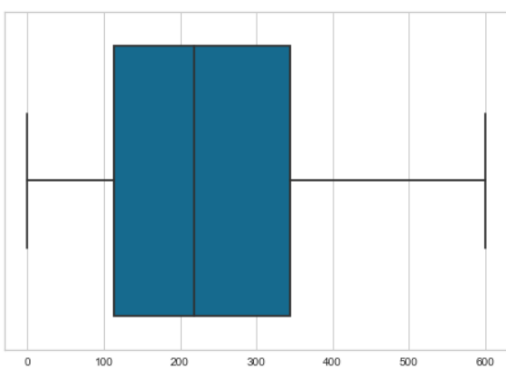
```
In [118]: sns.distplot(df_customer_segment['recency'])
```

```
Out[118]: <AxesSubplot:xlabel='recency', ylabel='Density'>
```




```
In [120]: sns.boxplot(model_build_var_withoutscale['recency'])
```

```
Out[120]: <AxesSubplot:xlabel='recency'>
```



In [121]:

```
ss=StandardScaler()
model_build_var.loc[:,:]=ss.fit_transform(model_build_var)
model_build_var.head()
```

Out[121]:

	recency	frequency	total payment
0	-0.829732	-0.159666	-0.054089
1	-0.809857	-0.159666	-0.568546
2	1.992503	-0.159666	-0.336345
3	0.561511	-0.159666	-0.535685
4	0.342887	-0.159666	0.178111

In [122]:

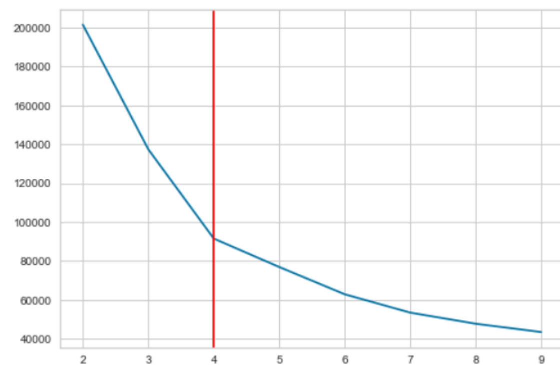
```
wcss=[]

for i in range(2,10):
    kmeans=KMeans(n_clusters=i,random_state=10)
    kmeans.fit(model_build_var)
    wcss.append(kmeans.inertia_)
wcss
```

Out[122]:

```
[201458.30465649045,
 137368.9528919505,
 91477.2440042515,
 76865.08971466008,
 62845.243455301345,
 53423.58972052539,
 47706.70459113506,
 43436.99807902035]
```

In [123]:

```
plt.plot(range(2,10),wcss)
plt.axvline(x=4,color='red')
plt.show()
```



From the Elbow Plot, we Observe that the inertia is minimum at k=4 , therefore the min no of clusters is 4

According to the elbow plot, we can see that the optimal K=4.

In [124]:
```
kmeans=KMeans(n_clusters=4,random_state=10)
kmeans.fit(model_build_var)
```
Out[124]: `KMeans(n_clusters=4, random_state=10)`

In [125]:
```
model_build_var_withoutscale['cluster']=kmeans.labels_
```

In [126]:
```
model_build_var_withoutscale.head()
```
Out[126]:

	recency	frequency	total payment	cluster
0	111	1	129.90	0
1	114	1	18.90	0
2	537	1	69.00	1
3	321	1	25.99	1
4	288	1	180.00	1

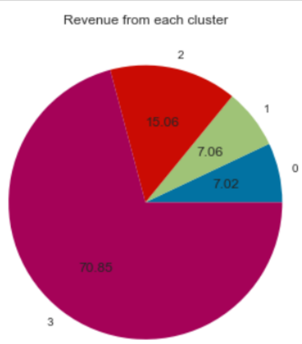
In [127]:
```
model_describe=model_build_var_withoutscale.groupby(by='cluster',as_index=False).agg({'recency': 'mean',
    'frequency': 'mean',
    'total payment': ['mean', 'count']}).round(2)
```

In [128]:
```
model_describe
```
Out[128]:

	cluster	recency	frequency	total payment	
		mean	mean	mean	count
0	0	127.10	1.00	113.37	50546
1	1	384.19	1.00	113.98	37614
2	2	219.23	2.11	243.11	2762
3	3	235.77	1.01	1143.68	2178

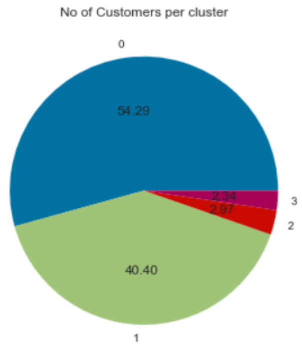
We have plotted the customer segmentation using pie chart.

In [129]:
```
plt.pie(model_describe['total payment','mean'],autopct='%.2f',labels=model_describe['cluster'],radius=1)
plt.title('Revenue from each cluster')
plt.show()
```



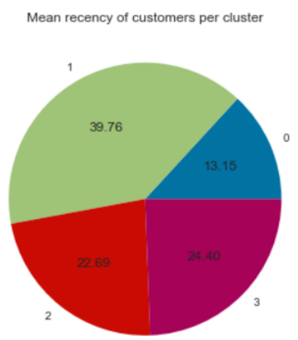
Majority revenue comes from cluster 3

```
In [130]: plt.pie(model_describe['total payment','count'],labels=model_describe['cluster'],autopct='%.2f')
          plt.title('No of Customers per cluster')
          plt.show()
```



Cluster 0 and 1 have the highest customers per cluster.

```
In [131]: plt.pie(x=model_describe['recency','mean'],labels=model_describe['cluster'],autopct='%.2f')
          plt.title('Mean recency of customers per cluster')
          plt.show()
```



Apart from cluster 1, all the clusters have same mean recency of customers.

RFM Analysis

In [134]:
```python
# finding the time range of the data given.
time_range=str((ord_delivered['order_purchase_timestamp'].max()-ord_delivered['order_purchase_timestamp'].min()))
```

In [135]:
```python
# Dividing the time range into 4 periods, since we have 4 clusters
period_days=int(re.sub(r'\s+days.*', '',time_range))/4
```

In [136]:
```python
df_customer_segment
```

Out[136]:

	customer_unique_id	recency	days_since_first_order	frequency	total payment	avg payment
0	0000366f3b9a7992bf8c76cfdf3221e2	111	111	1	129.90	129.90
1	0000b849f77a49e4a4ce2b2a4ca5be3f	114	114	1	18.90	18.90
2	0000f46a3911fa3c0805444483337064	537	537	1	69.00	69.00
3	0000f6ccb0745a6a4b88665a16c9f078	321	321	1	25.99	25.99
4	0004aac84e0df4da2b147fca70cf8255	288	288	1	180.00	180.00
...	...	...	...	...	...	...
93095	fffcf5a5ff07b0908bd4e2dbc735a684	447	447	1	1570.00	1570.00
93096	fffea47cd6d3cc0a88bd621562a9d061	262	262	1	64.89	64.89
93097	ffff371b4d645b6ecea244b27531430a	568	568	1	89.90	89.90
93098	ffff5962728ec6157033ef9805bacc48	119	119	1	115.00	115.00
93099	ffffd2657e2aad2907e67c3e9daecbeb	484	484	1	56.99	56.99

93100 rows × 6 columns

In [138]:
```python
#segmenting customers based on recency period
def seg_cust(rec):
    if rec<=period_days:
        return 'active'
    elif ((rec>period_days) & (rec<=(period_days*2))):
        return 'hot'
    elif ((rec>(period_days*2)) & (rec<=(period_days*3))):
        return 'active'
    elif (rec>(period_days*3)):
        return 'inactive'
```

In [139]:
```python
df_customer_segment['customer_type']=df_customer_segment['recency'].map(seg_cust)
```

In [140]:
```python
df_customer_segment['customer_type'].value_counts()
```

Out[140]:
```
active      56266
hot         33691
inactive     3143
Name: customer_type, dtype: int64
```

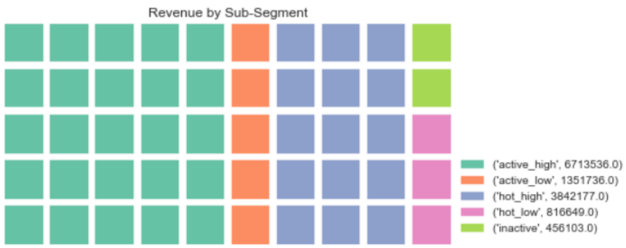
In [141]:
```python
# Getting the average payment median.
median_payment=df_customer_segment['avg payment'].median()
```

In [142]:
```python
# Dividing into sub classes based on the avg payment.
class_inactive_low = df_customer_segment['customer_type'] == 'inactive'
class_cold_low = (df_customer_segment['customer_type'] == 'cold')\
    & (df_customer_segment['avg payment'] < median_payment)
class_cold_high = (df_customer_segment['customer_type'] == 'cold')\
    & (df_customer_segment['avg payment'] >= median_payment)
class_hot_low = (df_customer_segment['customer_type'] == 'hot')\
    & (df_customer_segment['avg payment'] < median_payment)
class_hot_high = (df_customer_segment['customer_type'] == 'hot')\
    & (df_customer_segment['avg payment'] >= median_payment)
class_active_low = (df_customer_segment['customer_type'] == 'active')\
    & (df_customer_segment['avg payment'] < median_payment)
class_active_high = (df_customer_segment['customer_type'] == 'active')\
    & (df_customer_segment['avg payment'] >= median_payment)
```

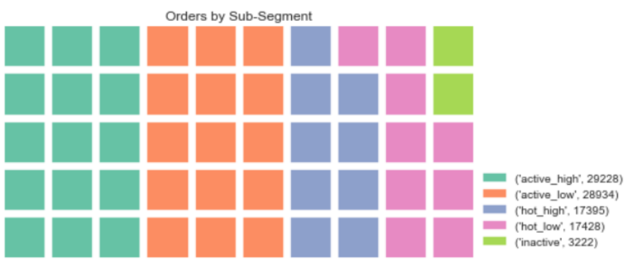
In [143]:
```python
# Adding sub labels to the dataframe
df_customer_segment.loc[class_inactive_low, "sub_segment"] = "inactive"
df_customer_segment.loc[class_cold_low, "sub_segment"] = "cold_low"
df_customer_segment.loc[class_cold_high, "sub_segment"] = "cold_high"
df_customer_segment.loc[class_hot_low, "sub_segment"] = "hot_low"
df_customer_segment.loc[class_hot_high, "sub_segment"] = "hot_high"
df_customer_segment.loc[class_active_low, "sub_segment"] = "active_low"
df_customer_segment.loc[class_active_high, "sub_segment"] = "active_high"
```

In [144]:
```python
def plot_waffle_chart(dat, metric, agg, title_txt, group='sub_segment'):

    '''Funtion to create a waffle chart. The visualization shows how the customer sub-segments are distributed
       according defined metrics.
       Input:
       - dat - dataframe
       - metric - feature/ kpi metric to visualize
       - agg - method to aggregate
       - title_txt - text to display as chart title
       Output:
       - waffle chart'''
    data_revenue = dict(round(dat.groupby(group).agg({metric: agg}))[metric])
    plt.figure(FigureClass=Waffle,rows=5,columns=10,values=data_revenue,labels=[f"{k, v}" for k, v in data_revenue.items()],
           legend={'loc': 'lower left', 'bbox_to_anchor': (1, 0)},
           figsize=(8, 5)
           )

    plt.title(title_txt)
```

In [145]:
```python
# plotting the waffle chart based on revenue by each sub segment.
plot_waffle_chart(df_customer_segment,'total payment','sum','Revenue by Sub-Segment')
```



In [146]:
```python
# plot for no of orders in each sub segment.
plot_waffle_chart(df_customer_segment, 'frequency', 'sum', 'Orders by Sub-Segment')
```



In [147]:
```python
# From the above waffle chart, it is observed that active high customers are spending more, followed by hot_high customers.
# Active Customers are the highest group , followed by hot customers
# Active Customers are also the group with highest orders.
# Inactive customer proportion is also very less.
```

In [148]:
```python
# function for frequency score assign
def assign_frequency(x):
    if x >= 7:
        return 4
    elif x >= 4:
        return 3
    elif x >= 2:
        return 2
    else:
        return 1
```

In [149]:
```python
# calculating R,F,M scores
df_customer_segment['R']=pd.qcut(df_customer_segment['recency'],q=4,labels=range(4,0,-1))
df_customer_segment['F']=df_customer_segment['frequency'].apply(assign_frequency)
df_customer_segment['M']=pd.qcut(df_customer_segment['total payment'],q=4,labels=range(1,5))
```

In [149]:

```python
# calculating R,F,M scores
df_customer_segment['R']=pd.qcut(df_customer_segment['recency'],q=4,labels=range(4,0,-1))
df_customer_segment['F']=df_customer_segment['frequency'].apply(assign_frequency)
df_customer_segment['M']=pd.qcut(df_customer_segment['total payment'],q=4,labels=range(1,5))
```

In [151]:

```python
df_customer_segment.head()
```

Out[151]:

	customer_unique_id	recency	days_since_first_order	frequency	total payment	avg payment	customer_type	sub_segment	R	F	M
0	0000366f3b9a7992bf8c76cfdf3221e2	111	111	1	129.90	129.90	active	active_high	4	1	3
1	0000b849f77a49e4a4ce2b2a4ca5be3f	114	114	1	18.90	18.90	active	active_low	4	1	1
2	0000f46a3911fa3c0805444483337064	537	537	1	69.00	69.00	inactive	inactive	1	1	2
3	0000f6ccb0745a6a4b88665a16c9f078	321	321	1	25.99	25.99	hot	hot_low	2	1	1
4	0004aac84e0df4da2b147fca70cf8255	288	288	1	180.00	180.00	hot	hot_high	2	1	4

In [155]:

```python
# Getting the customer segment and also rfm score
df_customer_segment['segment_RFM']=df_customer_segment['R'].astype(str)+df_customer_segment['F'].astype(str)\
+df_customer_segment['M'].astype(str)
df_customer_segment['score_rfm'] =df_customer_segment[['R','F','M']].sum(axis=1)
```

In [155]:

```python
# Getting the customer segment and also rfm score
df_customer_segment['segment_RFM']=df_customer_segment['R'].astype(str)+df_customer_segment['F'].astype(str)\
+df_customer_segment['M'].astype(str)
df_customer_segment['score_rfm'] =df_customer_segment[['R','F','M']].sum(axis=1)
```


In [156]:

```python
df_customer_segment.head()
```

Out[156]:

	customer_unique_id	recency	days_since_first_order	frequency	total payment	avg payment	customer_type	sub_segment	R	F	M	segment_RFM	sc
0	0000366f3b9a7992bf8c76cfdf3221e2	111	111	1	129.90	129.90	active	active_high	4	1	3	413	
1	0000b849f77a49e4a4ce2b2a4ca5be3f	114	114	1	18.90	18.90	active	active_low	4	1	1	411	
2	0000f46a3911fa3c0805444483337064	537	537	1	69.00	69.00	inactive	inactive	1	1	2	112	
3	0000f6ccb0745a6a4b88665a16c9f078	321	321	1	25.99	25.99	hot	hot_low	2	1	1	211	
4	0004aac84e0df4da2b147fca70cf8255	288	288	1	180.00	180.00	hot	hot_high	2	1	4	214	

In [158]:

```python
# Grouping customers based on rfm score.
def rfm_type_assign(df):
    if (int(df['segment_RFM']) >= 434) or (df['score_rfm'] >= 9):
        return 'Best customer'
    elif (df['score_rfm'] >= 8) and (df['M'] == 4):
        return 'Big Spender'
    elif (df['score_rfm'] >= 6) and (df['F'] >= 2):
        return 'Loyalist'
    elif (int(df['segment_RFM']) >= 231) or (df['score_rfm'] >= 6):
        return 'Potential Loyalists'
    elif ((int(df['segment_RFM']) >= 121) and (df['R'] == 1)) or df['score_rfm'] == 5:
        return 'Almost Lost'
    elif (df['score_rfm'] >= 4) and (df['R'] == 1):
        return 'Hibernating'
    else:
        return 'Lost Customer'
```

In [159]:

```python
df_customer_segment['customer_rfm_segment']=df_customer_segment.apply(rfm_type_assign,axis=1)
```

In [160]:

```python
dict_strategy={'Best customer': 'Personalized communication, offer loyalty program, no promotional offers needed',
               'Big Spender': 'Make them feel valued and offer quality products, encourage to stick with brands',
               'Loyalist': 'Offer loyalty program',
               'Potential Loyalists': 'Recommend products and offer discounts',
               'Almost Lost': 'Try to win them with limited sales promotions',
               'Hibernating': 'Make great offers with big discounts',
               'Lost Customer': 'Do not spent much effort and money to win them'}
```

In [161]:
```python
df_customer_segment['business strategy']=df_customer_segment['customer_rfm_segment'].apply(lambda x:dict_strategy[x])
```

In [162]:
```python
df_customer_segment.head()
```

Out[162]:

customer_unique_id	recency	days_since_first_order	frequency	total payment	avg payment	customer_type	sub_segment	R	F	M	segment_RFM	score_rfm
00366f3b9a7992bf8c76cfdf3221e2	111	111	1	129.90	129.90	active	active_high	4	1	3	413	8
0b849f77a49e4a4ce2b2a4ca5be3f	114	114	1	18.90	18.90	active	active_low	4	1	1	411	6
0f46a3911fa3c0805444483337064	537	537	1	69.00	69.00	inactive	inactive	1	1	2	112	4
0f6ccb0745a6a4b88665a16c9f078	321	321	1	25.99	25.99	hot	hot_low	2	1	1	211	4
04aac84e0df4da2b147fca70cf8255	288	288	1	180.00	180.00	hot	hot_high	2	1	4	214	7

In [163]:
```python

plot_waffle_chart(df_customer_segment,'customer_unique_id','count','RFM Segments','customer_rfm_segment')
```

In [166]:
```python
df_customer_segment.groupby('customer_rfm_segment').agg(
    Count = ('customer_unique_id', 'count'),
    Recency = ('recency', 'mean'),
    Frequency = ('frequency', 'mean'),
    Monetary = ('total payment', 'mean'),
    Strategy = ('business strategy', 'unique'),
).round(1)
```

Out[166]:

customer_rfm_segment	Count	Recency	Frequency	Monetary	Strategy
Almost Lost	11512	359.1	1.0	88.7	[Try to win them with limited sales promotions]
Best customer	6546	66.5	1.2	356.5	[Personalized communication, offer loyalty pro...
Big Spender	5710	172.7	1.1	346.1	[Make them feel valued and offer quality produ...
Hibernating	5766	443.0	1.0	64.8	[Make great offers with big discounts]
Lost Customer	11520	366.1	1.0	29.0	[Do not spent much effort and money to win them]
Loyalist	1132	289.6	2.1	174.5	[Offer loyalty program]
Potential Loyalists	50914	183.4	1.0	136.4	[Recommend products and offer discounts]

In [167]:
```python
# From the above table, it is observed that we have around 50914 customers who are potential based on the rfm analysis, so f
# those people the company should recommend products and offer them discounts so that they can increase their sales.
# Now for 11520 who are lost because their recency is more than an year, don't spend much money on them since they are alrea
# lost.
```

In [168]:
```python
# Also there are customers who are hibernating and almost lost customers where we can provide offers and discounts for them.
```

Limitations

1. We have used linear regression model to predict future sales. The Main limitation of Linear Regression is the assumption of linearity between the dependent variable and the independent variables. In the real world, the data is rarely linearly separable. It assumes that there is a straight-line relationship between the dependent and independent variables which is incorrect many times.

2. Since the given data is slightly imbalanced, the validity of the linear regression model suffers.

3. We have used K-Means clustering for customer segmentation, and the main limitation is that the user has to specify the number of clusters in the beginning. That's why we have used scree plot to determine the optimal value of k.

4. K-means can handle only numerical data, therefore we had to make dummy of categorical variables and omit the categorical variables which are redundant.

5. Due to its limitation, we are assuming the clusters are spherical and the model also assumes each cluster has equal number of observations.

Closing reflections

What we have learnt

1. How to proceed and what things to do first like understand the dataset properly so that we can understand the problem and what are the features and their nature.

2. Correct way to perform analysis on variables and which things to be keep in mind when describing the variables and inferences.

3. Outlier treatment not necessary all the time because it may have pattern in it and may affect our prediction. Although after treating the outliers it did not affect the models so we went with outliers.

4. Scaling affects the model's performance for which the scaling is required but for other models it did not affect the model performance.

5. Feature engineering helped us to understand the patterns and predicting the target variable in better way in machine learning models.

6. By performing RFM analysis after customer segmentation, we were able to quantitatively rank and group customers based on the recency, frequency and monetary total of their recent transactions to identify the best customers and perform targeted marketing campaigns.