Customer Segmentation using RFM Analysis on E-Commerce Dataset

Data Analysis Portfolio by Fara Rizkhi Karunia

Objective Statement

The purpose of processing this dataset is to get various insights from the data analysis process and can find out the customer segmentation contained in this e-commerce so that later it can be information to optimize marketing strategies and increase sales.

- Get information about E-Commerce Customer Demographics
- Knowing the level of accuracy of shipping estimates on E-Commerce
- Get business insight about the Seller with the Biggest and Smallest Revenue
- To get business insights about Sales Performance and Revenue in the last few months
- To find out the correlation of packaging and delivery time affecting ratings
- To determine customer segmentation based on RFM analysis

Datasets

Brazilian E-Commerce Public Dataset by Olist (Source : Kaggle)

This dataset has information from 100K orders from 2016 to 2018. Its features make it possible to view orders from multiple dimensions: from order status, price, payment, and delivery performance to customer location, product attributes, and finally reviews written by customers. There is also geolocation data that links Brazilian zip codes with latitude/longitude coordinates.

Data Dictionary

- **customer_id**
  unique customer code
- **customer_city**
  customer's city name
- **customer_state**
  customer's state name
- **order_id**
  unique order codes
- **order_item_id**
  quantity of order
- **product_id**
  unique product code
- **seller_id**
  unique seller code
- **review_id**
  unique review identifier

- **order_purchase_timestamp**
  customers payment time
- **order_approved_at**
  time of order receipt by the seller
- **order_delivered_carrier_date**
  time of order handled by the logistic partner
- **order_delivered_customer_date**
  time of order received by the customer
- **order_estimated_delivery_date**
  estimated delivery date
- **price**
  product prices per item (in Real Brasil)
- **freight_value**
  shipping prices per item

- **order_status**
  order status : delivered, invoiced, shipped, canceled, processing, unavailable, created, approved
- **review_score**
  ranging from 1 to 5 given by the customer on a satisfaction survey

Business Questions

- What are the Demographics of E-Commerce Customers?
- How is the Accuracy of Estimated Delivery on this E-Commerce?
- Which Seller has the Biggest and Smallest Revenue?
- What are the Best Seller Products?
- How is the Sales and Revenue Performance in the Last Few Months?
- How does packaging and shipping time affect the rating given?
- How is the customer segmentation formed?

Methodology

- Descriptive Analysis
- Correlation Analysis
- RFM Analysis

Data Wrangling

Asessing Data

From the asessing data process, several problems were found that must be overcome before the data analysis process is carried out. Some of these errors are as follows:

In the order_items_df table

- Shipping_limit_date column data type error

In the orders_df table

- Data type error in columns order_purchase_timestamp to order_estimated_delivery_date
- Missing value in columns order_approved_at, order_delivered_carrier_date, and order_delivered_customer_date
- There are inaccurate values in the order_purchase_timestamp, order_approved_at, order_delivered_carrier_date, and order_delivered_customer_date columns. Because these columns are date columns that will be processed sequentially, the column with a large index cannot be smaller than the previous column to avoid producing negative values.

In the order_reviews table

- Data type error in review_creation_date and review_answer_timestamp columns
- Missing values in review_comment_title and review_comment_message

Data Wrangling Data Cleaning

Handling Data Type Error

```
datetime_columns = ["shipping_limit_date"]

for column in datetime_columns:
    order_items_df[column] = pd.to_datetime(order_items_df[column])
```

```
datetime_columns = ["order_purchase_timestamp", "order_approved_at", "order_delivered_carrier_date",
                    "order_delivered_customer_date", "order_estimated_delivery_date"]

for column in datetime_columns:
    orders_df[column] = pd.to_datetime(orders_df[column])
```

Deleting unused columns

```
hapus_kolom = ["review_comment_title", "review_comment_message",
               "review_creation_date", "review_answer_timestamp"]
order_reviews_df = order_reviews_df.drop(hapus_kolom, axis=1)
order_reviews_df.head()
```

```
hapus_kolom = ["customer_unique_id"]
customers_df = customers_df.drop(hapus_kolom, axis=1)
customers_df.head()
```

```
# Menghapus tabel approved_time, packing_time, dan shipping_time karna tidak diperlukan lagi
hapus = ["approved_time", "packing_time", "shipping_time"]
orders_df = orders_df.drop(hapus, axis=1)
```

Data Wrangling Data Cleaning

Handling Missing Values

Filtering data

```
# display data rows that contain missing values
orders_df[orders_df.order_approved_at.isna()]
```

After filtering, I found the possibility that the column is empty is because the order_status is canceled. So I will check other order_status that may have missing values

```
# Check the number of order status
orders_df.order_status.value_counts()
```

```
order_status
delivered      96478
shipped         1107
canceled         625
unavailable      609
invoiced         314
processing       301
created            5
approved           2
Name: count, dtype: int64
```

The columns are empty because the order status hasn't been completed or the order has been canceled. I can't use the dropping method here because it will delete some data with status other than "delivered". So I will fill the empty columns with the imputation method and fill them with the purchased date.

Data Wrangling Data Cleaning

Handling Missing Values

```
# Resolve Missing Value and fill it with value from order_purchase_timestamp column
orders_df['order_approved_at'] = orders_df['order_approved_at'].fillna(orders_df['order_purchase_timestamp'])
```

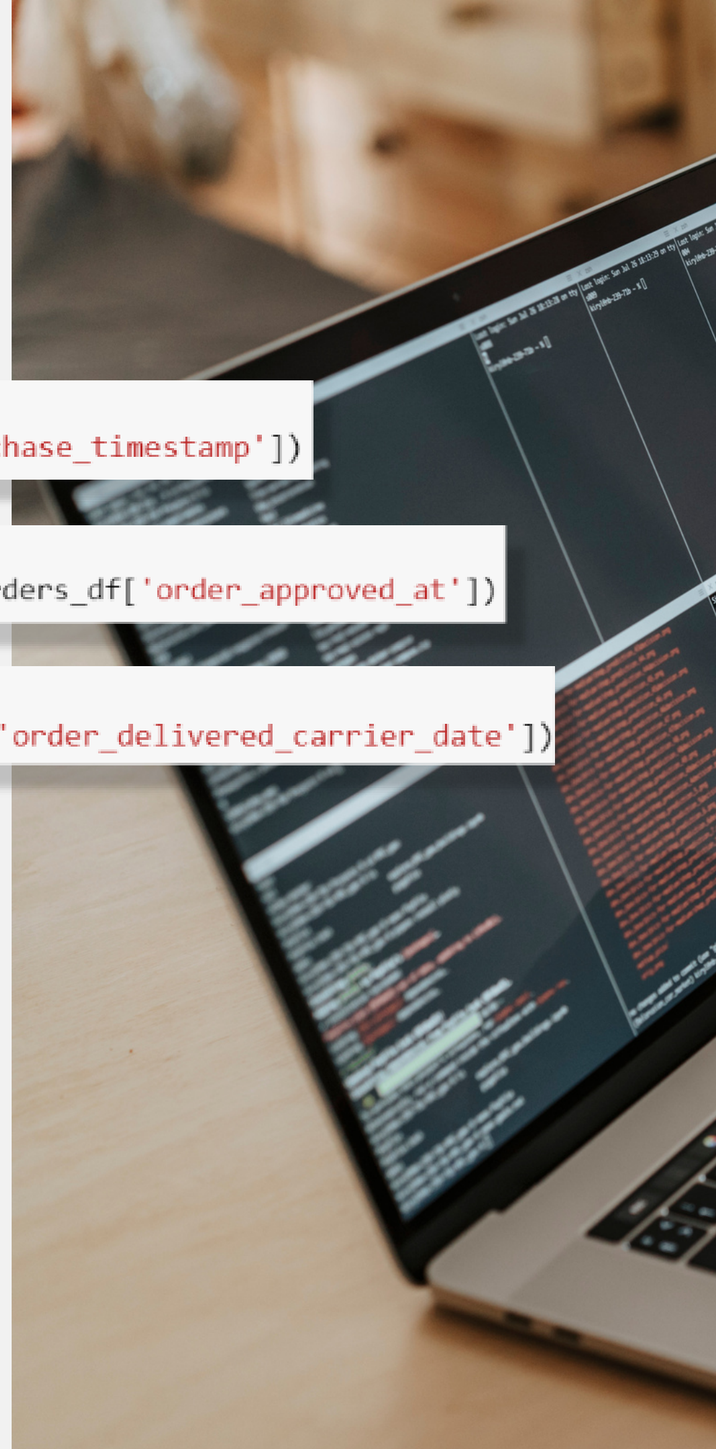
```
#  Resolve Missing Value and fill it with value from order_approved_at
orders_df['order_delivered_carrier_date'] = orders_df['order_delivered_carrier_date'].fillna(orders_df['order_approved_at'])
```

```
 # Resolve Missing Value and fill it with value from order_delivered_carrier_date
rder_delivered_customer_date'] = orders_df['order_delivered_customer_date'].fillna(orders_df['order_delivered_carrier_date'])
```

Checking Missing Values

```
# Checking Missing Values
orders_df.isna().sum()
```

```
order_id                         0
customer_id                      0
order_status                     0
order_purchase_timestamp         0
order_approved_at                0
order_delivered_carrier_date     0
order_delivered_customer_date    0
order_estimated_delivery_date    0
dtype: int64
```



Data Wrangling Data Cleaning

Handling Inaccurate Values

Checking for negative values in order_delivered_carrier_date - order_approved_at

```
# Inaccurate Value : Checking for negative values in order_delivered_carrier_date - order_approved_at
orders_df["packing_time"]=orders_df["order_delivered_carrier_date"]-orders_df["order_approved_at"]
orders_df["packing_days"]=orders_df["packing_time"].dt.days


orders_df["packing_days"].describe()
```

```
count    99441.000000
mean         2.260738
std          3.541257
min       -172.000000
25%          0.000000
50%          1.000000
75%          3.000000
max        125.000000
Name: packing_days, dtype: float64
```

Resolving negative value in the "packing_days" column

```
# Resolving negative value in the "packing_days" column by replacing it with the average packing days
orders_df.loc[orders_df["packing_days"] < 0, "packing_days"] = orders_df["packing_days"].mean()

# Replacing order_approved_at date by adding average "packing_days" to "order_approved_at"
orders_df["order_delivered_carrier_date"] = orders_df["order_approved_at"] + pd.to_timedelta(orders_df["packing_days"], unit

# Rechecking
orders_df["packing_days"].describe()
```

```
count    99441.000000
mean         9.054322
std          8.758264
min          0.000000
25%          4.000000
50%          7.000000
75%         12.000000
max        205.000000
Name: shipping_days, dtype: float64
```

Data Wrangling Data Cleaning

Handling Inaccurate Values

Checking for negative values in order_delivered_customer_date-order_delivered_carrier_date

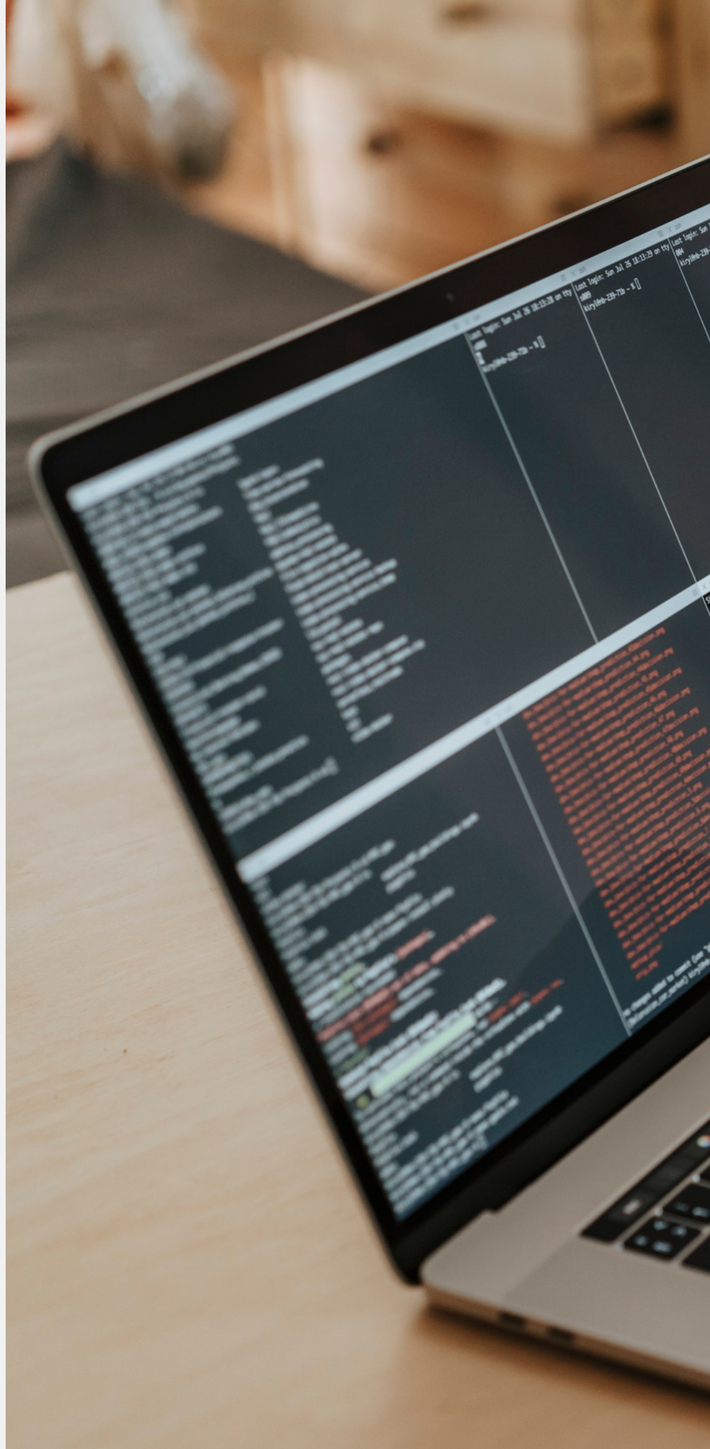
```
# Inaccurate Value : Checking for negative values in order_delivered_customer_date-order_delivered_carrier_date
orders_df["shipping_time"]=orders_df["order_delivered_customer_date"]-orders_df["order_delivered_carrier_date"]
orders_df["shipping_days"]=orders_df["shipping_time"].dt.days


orders_df["shipping_days"].describe()
```

```
count    99441.000000
mean         9.009976
std          8.787853
min        -16.000000
25%          4.000000
50%          7.000000
75%         12.000000
max        205.000000
Name: shipping_days, dtype: float64
```

Resolving negative value in the "shipping_days" column

```
# Resolving negative value in the "shipping_days" column by replacing it with the average shipping days
orders_df.loc[orders_df["shipping_days"] < 0, "shipping_days"] = orders_df["shipping_days"].mean()

# Replacing order_delivered_customer_date by adding average "shipping_days" to "order_delivered_carrier_date"
orders_df["order_delivered_customer_date"] = orders_df["order_delivered_carrier_date"] + pd.to_timedelta(orders_df["shipping

# Rechecking
orders_df["shipping_days"].describe()
```

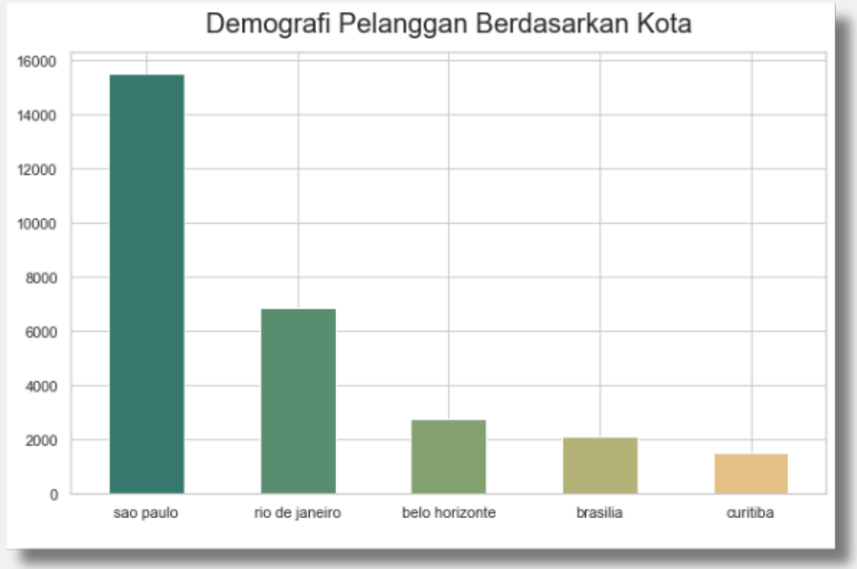
```
count    99441.000000
mean         9.054322
std          8.758264
min          0.000000
25%          4.000000
50%          7.000000
75%         12.000000
max        205.000000
Name: shipping_days, dtype: float64
```

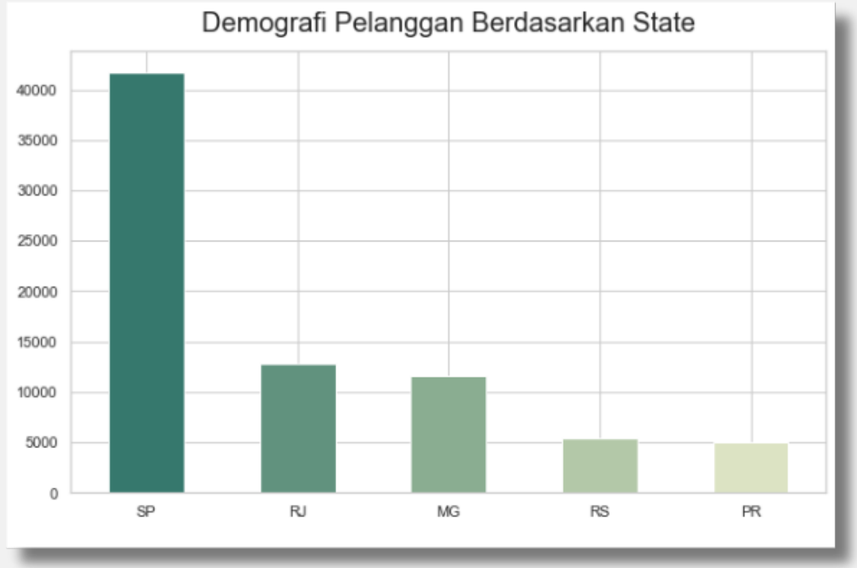


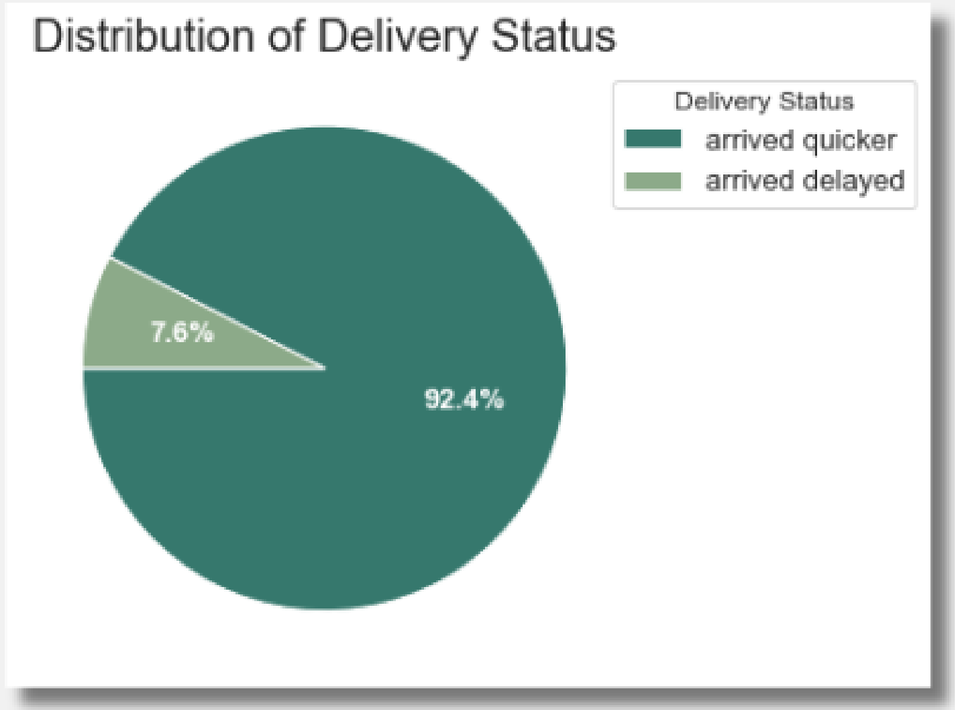
Exploratory Data Analysis

The Demographics of E-Commerce Customers

- Cities with the highest number of customers are sao paulo, rio de janeiro, belo horizonte, trasilia, and curitiba.
- State with the highest number of customers in the state of SP, RJ, MG, RS, and PR
- These customer demographics can be used as a consideration in determining marketing strategies or market expansion to these areas.



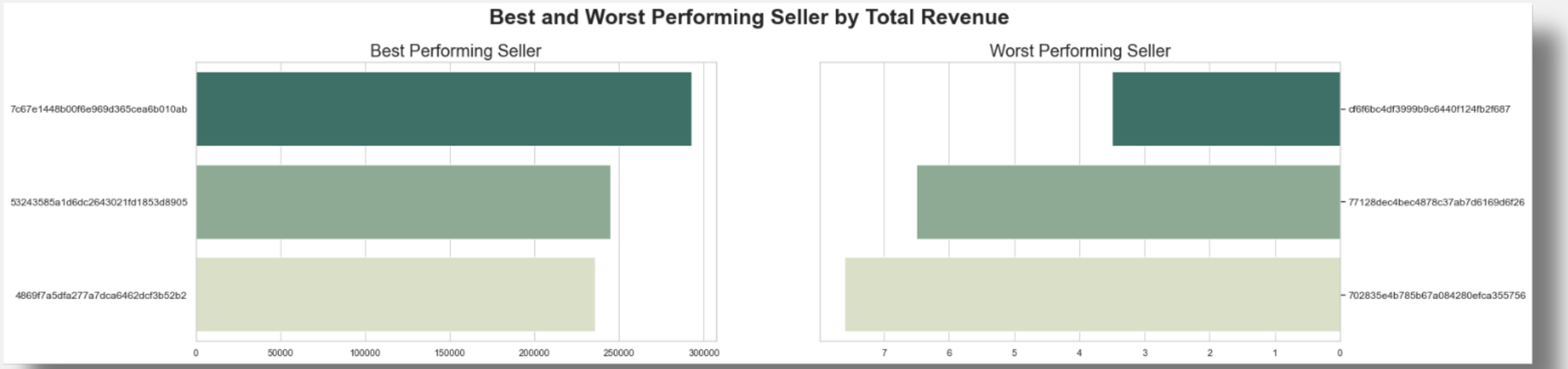




The Distribution of Delivery Status

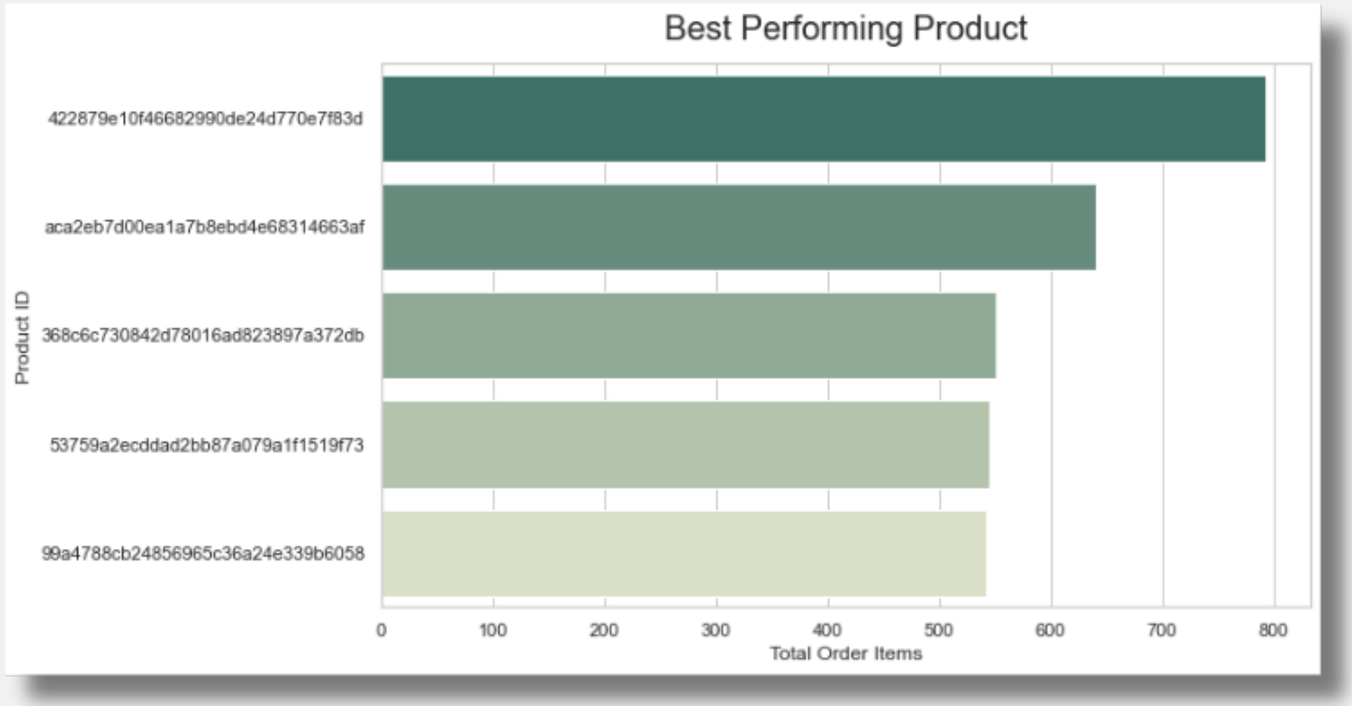
Based on the pie chart that has been visualized, it can be concluded that 92.4% of shipments are completed before the estimated date of arrival (arrived quickly), but there are still 7.6% of shipments that are completed not on time (arrived delayed). This shows that the estimated date of arrival has not shown sufficiently accurate results.

Best and Worst Performing Seller
by Total Revenue



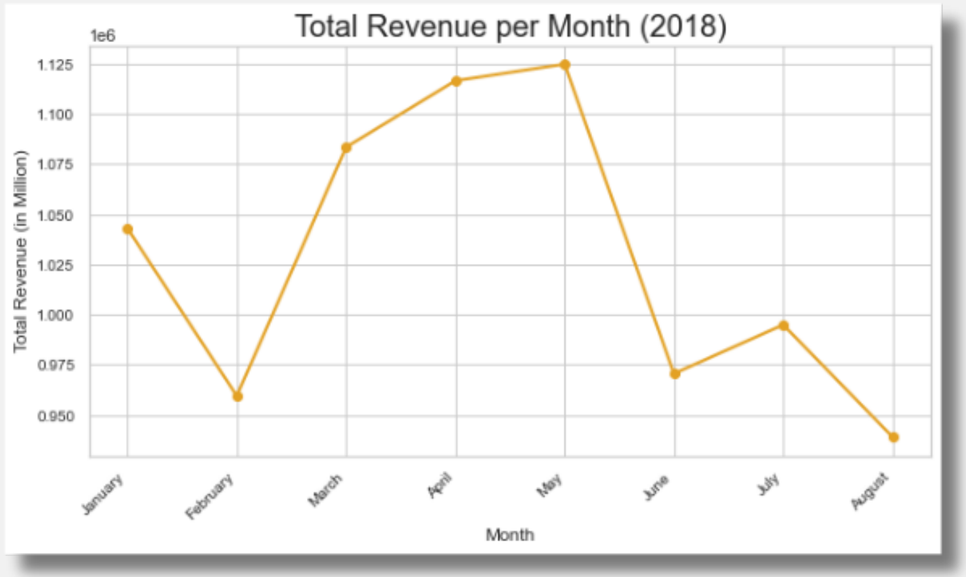
Based on the visualized bar chart, it can be concluded that the best performing seller is seller_id 7c67e1448b00f6e969d365cea6b010ab with a total revenue of 292489. Meanwhile, the worst performing seller is seller_id cf6f6bc4df3999b9c6440f124fb2f687 with a total revenue of 3.5.

Best Performing Product



Based on the visualized bar chart, the 3 best-selling products are product_id 422879e10f46682990de24d770e7f83d, aca2eb7d00ea1a7b8ebd4e68314663af, and 368c6c730842d78016ad823897a372db with a total of 793, 640, and 551 units sold.

Revenue Performance



- The highest number of orders was in March 2018 with 7005 order_id
- The lowest number of orders was in June 2018 with 6113 order_id
- The most significant increase in the number of orders was in February-March by 196 orders
- The most significant decrease in the number of orders was in May-June by 670 orders

Sales Performance

- The highest amount of revenue was in May 2018 with a total revenue of 1124649.55
- The lowest amount of revenue was in August 2018 with a total revenue of 939171.01
- The most significant increase in revenue was in February-March by 123578.36
- The most significant decrease in revenue was in May-June by 154010.53

Correlation Analysis

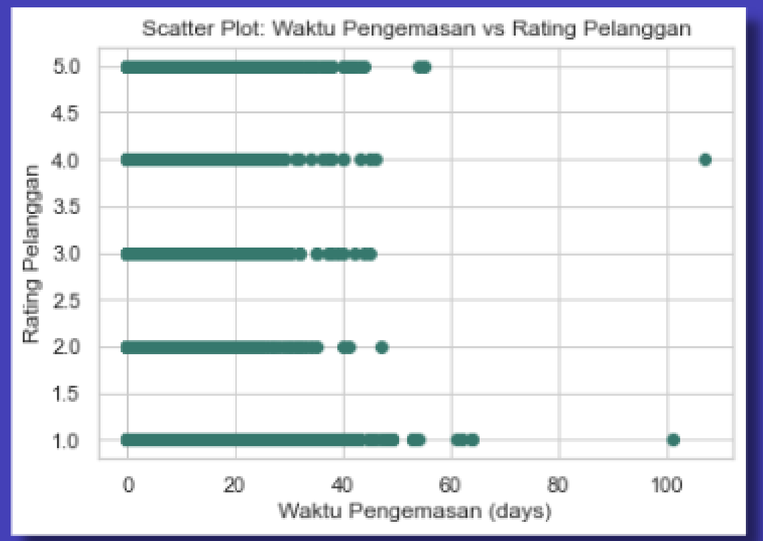
Packaging Time VS Rating

```python
# Mencari korelasi antara kolom packing_days dengan review_score
correlation = all_df["packing_days"].corr(all_df["review_score"])
print(f"Korelasi antara waktu pengemasan dan rating pelanggan: {correlation}")
```

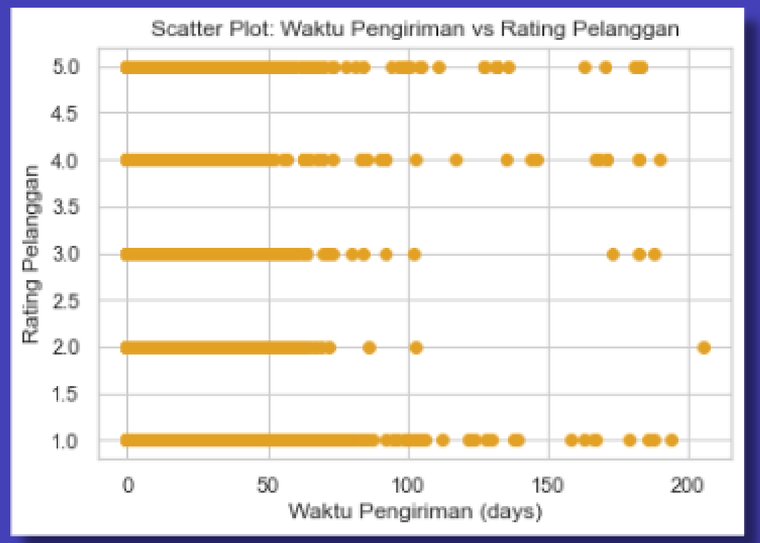
```
Korelasi antara waktu pengemasan dan rating pelanggan: -0.1339399795765589
```

Based on the analysis, the correlation value between packaging time and rating is -0.134. So it can be concluded that the packaging time and rating are negatively correlated, which means that the longer the packaging time, the lower the rating given by the customer. However, because the correlation value is quite small this correlation is not too strong so further analysis of other influencing factors is needed.

```python
colors = ["#36786d"]
plt.scatter(all_df["packing_days"], all_df["review_score"], color=colors)
plt.xlabel("Packing Days")
plt.ylabel("Customer's Rate")
plt.title("Scatter Plot: Packing Days vs Customer's Rate")
plt.show()
```



```
plt.scatter(all_df["shipping_days"], all_df["review_score"], color="#e3a124")
plt.xlabel("Shipping Days")
plt.ylabel("Customer's Rate")
plt.title("Scatter Plot: Shipping Days vs Customers Rate")
plt.show()
```



Correlation Analysis

Shipping Time VS Rating

```
# Mencari korelasi antara kolom shipping_days dengan review_score
correlation = all_df["shipping_days"].corr(all_df["review_score"])
print(f"Korelasi antara waktu pengiriman dan rating pelanggan: {correlation}")
```

```
Korelasi antara waktu pengiriman dan rating pelanggan: -0.22045529184106738
```

Based on the analysis, the correlation value between delivery time and rating is -0.220. So it can be concluded that the delivery time and rating are negatively correlated, which means that the longer the delivery time, the lower the rating given by the customer. Although the correlation value of delivery time with rating is greater than the previous analysis, this correlation value is also small so that this correlation is not too strong and further analysis needs to be done on other influencing factors.

RFM Analysis

RFM analysis is a marketing technique used to analyze and categorize customers based on their buying behavior. RFM stands for :

- Recency (R) : Refers to how recently a customer has made a purchase. It measures the time elapsed since the last customer transaction. Customers who have made a purchase more recently are often considered more engaged.
- Frequency (F) : Represents the number of transactions or purchases made by a customer within a specific time period. It provides insights into how often a customer interacts with a business. Higher frequency often indicates greater engagement and loyalty.
- Monetary Value (M) : Refers to the total amount of money a customer has spent on purchases. It measures the overall value or contribution of a customer to the business. Customers with higher monetary value are often more valuable to the business.

RFM Analysis

Calculating the RFM Value of Each Customer

```python
# Specify the analysis date using the last date in the dataset
analysis_date = all_df["order_purchase_timestamp"].max()

# Calculating Recency, Frequency, dan Monetary
rfm_df = all_df.groupby("customer_id").agg({
    "order_purchase_timestamp": lambda x: (analysis_date - x.max()).days,
    "order_id": "count",
    "revenue": "sum"
}).reset_index()

# Name the columns according to RFM
rfm_df.columns = ["customer_id", "recency", "frequency", "monetary"]

# Showing RFM Results
rfm_df.head()
```

	customer_id	recency	frequency	monetary
0	00012a2ce6f8dcda20d059ce98491703	292	1	89.80
1	000161a058600d5901f007fab4c27140	413	1	54.90
2	0001fd6190edaaf884bcaf3d49edf079	551	1	179.99
3	0002414f95344307404f0ace7a26f1d5	382	1	149.90
4	000379cdec625522490c315e70c7a9fb	153	1	93.00

```python
# Viewing rfm_df data description
rfm_df
rfm_df.describe()
```

	recency	frequency	monetary
count	96517.000000	96517.000000	96517.000000
mean	243.939845	1.147466	156.565976
std	153.422744	0.550264	357.193080
min	0.000000	1.000000	0.850000
25%	119.000000	1.000000	47.900000
50%	224.000000	1.000000	89.900000
75%	354.000000	1.000000	159.990000
max	728.000000	22.000000	60480.000000

RFM Analysis

Specifying the Binning Range

Determine the binning range by observing the quartile data in the data description.

```python
# Specifying the Binning Range Value
recency_bins = [0, 119, 224, 354, 730]
frequency_bins = [0, 1, 22]
monetary_bins = [0, 48, 90, 160, 60480]

# Apply binning range using pd.cut()
rfm_df["R"] = pd.cut(rfm_df["recency"], bins=recency_bins, labels=["R4", "R3", "R2", "R1"])
rfm_df["F"] = pd.cut(rfm_df["frequency"], bins=frequency_bins, labels=["F1", "F2"])
rfm_df["M"] = pd.cut(rfm_df["monetary"], bins=monetary_bins, labels=["M1", "M2", "M3", "M4"])

# Combining the RFM segments
rfm_df["RFM_Segment"] = rfm_df["R"].astype(str) + rfm_df["F"].astype(str) + rfm_df["M"].astype(str)
rfm_df[["customer_id", "recency", "frequency", "monetary", "RFM_Segment"]].head()
```

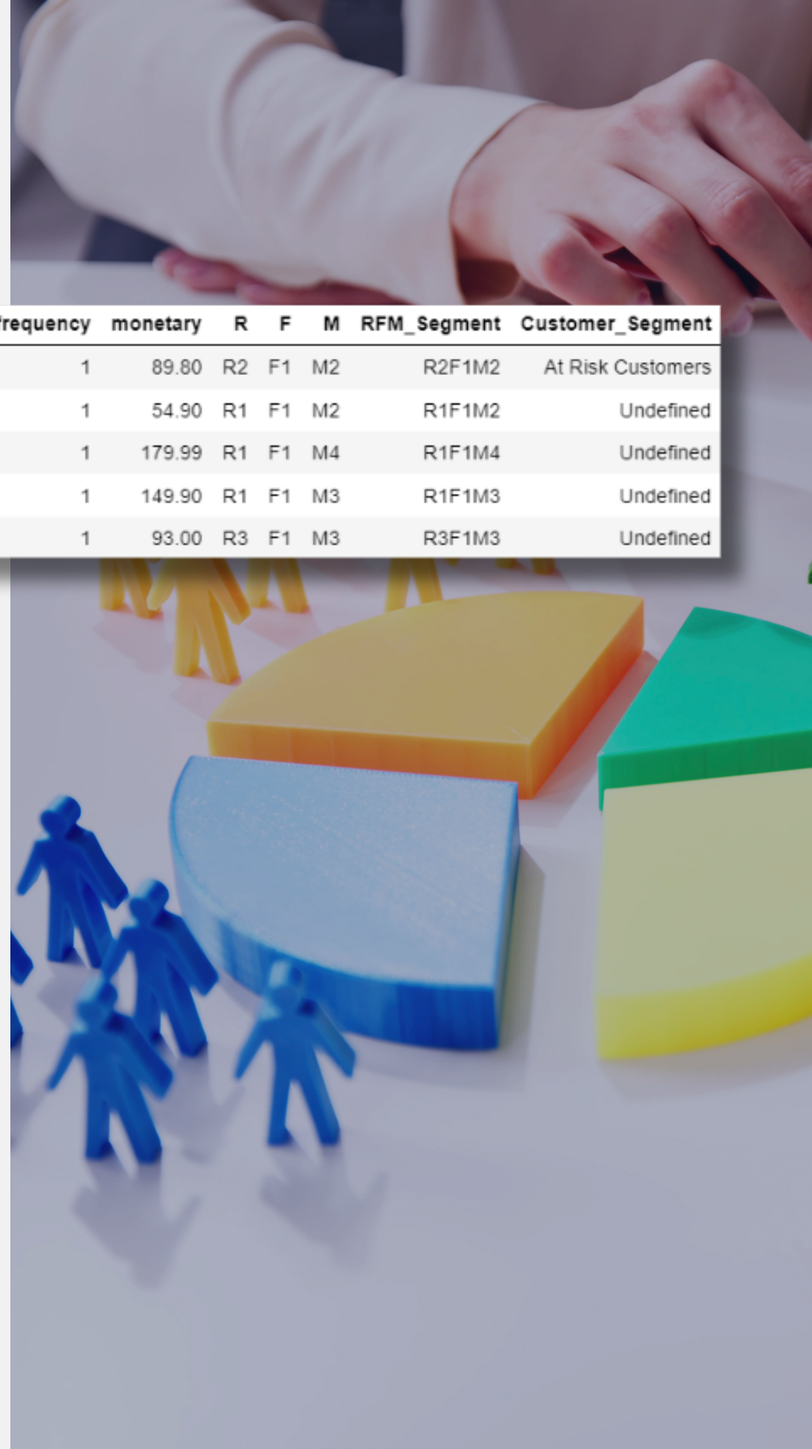
	customer_id	recency	frequency	monetary	RFM_Segment
0	00012a2ce6f8dcda20d059ce98491703	292	1	89.80	R2F1M2
1	000161a058600d5901f007fab4c27140	413	1	54.90	R1F1M2
2	0001fd6190edaaf884bcaf3d49edf079	551	1	179.99	R1F1M4
3	0002414f95344307404f0ace7a26f1d5	382	1	149.90	R1F1M3
4	000379cdec625522490c315e70c7a9fb	153	1	93.00	R3F1M3

Customer Segmentation

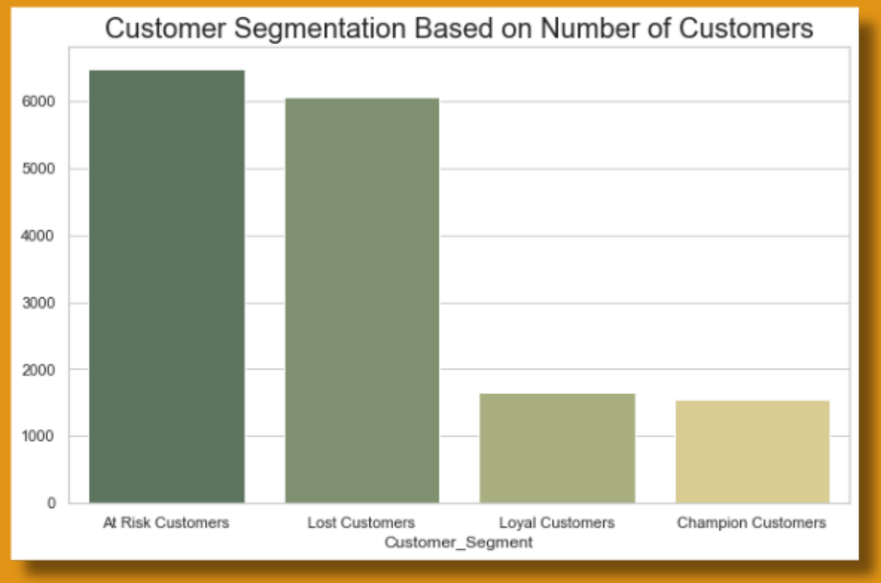
```python
# Create a customer_segment Column
rfm_df["Customer_Segment"] = "Undefined"

# Determine the conditions according to the desired customer segmentation
champion_condition = (rfm_df["RFM_Segment"] == "R4F2M4")
loyal_condition = (rfm_df["RFM_Segment"] == "R3F2M4")
at_risk_condition = (rfm_df["RFM_Segment"] == "R2F1M2")
lost_condition = (rfm_df["RFM_Segment"] == "R1F1M1")

# Establish segmentation based on defined conditions
rfm_df.loc[champion_condition, "Customer_Segment"] = "Champion Customers"
rfm_df.loc[loyal_condition, "Customer_Segment"] = "Loyal Customers"
rfm_df.loc[at_risk_condition, "Customer_Segment"] = "At Risk Customers"
rfm_df.loc[lost_condition, "Customer_Segment"] = "Lost Customers"

rfm_df.head()
```

	customer_id	recency	frequency	monetary	R	F	M	RFM_Segment	Customer_Segment
0	00012a2ce6f8dcda20d059ce98491703	292	1	89.80	R2	F1	M2	R2F1M2	At Risk Customers
1	000161a058600d5901f007fab4c27140	413	1	54.90	R1	F1	M2	R1F1M2	Undefined
2	0001fd6190edaaf884bcaf3d49edf079	551	1	179.99	R1	F1	M4	R1F1M4	Undefined
3	0002414f95344307404f0ace7a26f1d5	382	1	149.90	R1	F1	M3	R1F1M3	Undefined
4	000379cdec625522490c315e70c7a9fb	153	1	93.00	R3	F1	M3	R3F1M3	Undefined

```python
# Selected customer segmentation with 4 predefined categories
selected_segments = ["Champion Customers", "Loyal Customers", "At Risk Customers", "Lost Customers"]
selected_df = rfm_df[rfm_df["Customer_Segment"].isin(selected_segments)]

# Number of customers in each segment
selected_df_counts = selected_df["Customer_Segment"].value_counts()

colors= ['#59785c', '#7f976a', '#adb678', '#e3d388']

# Data visualization of the number of customers in each segment
plt.figure(figsize=(10, 6))
sns.barplot(x=selected_df_counts.index, y=selected_df_counts.values, palette=colors)
plt.title("Customer Segmentation Based on Number of Customers", size=20)
plt.show()
```



The Demographics of E-Commerce Customers



The segmentation obtained is in the form of customer loyalty segmentation which is divided into:

- Champion Customers: Customers who have recently made a purchase (low Recency), shop frequently (high Frequency), and spend a significant amount of money (high Monetary). Champion Customers must be well cared for in order to remain consistent in shopping, perhaps giving special treatment such as giving reward vouchers etc.
- Loyal Customers: Although they may not have recently made a purchase, these customers shop frequently (high Frequency) and have been loyal customers for some time. Monetary may vary, but they make a steady contribution. This segmentation has the potential to increase customer value with retention strategies.
- At Risk Customers: Customers in this segmentation have the potential to switch to competitors or be inactive. So it is necessary to carry out a strategy to attract at risk customers back.
- Lost Customers: This segmentation has low recency, frequency, and monetary values. Most likely already not interested or switched to a competitor. So what needs to be considered is to focus on marketing strategies to reactivate these lost customers.

Thank You!

Repositories on GitHub : Customer Segmentation using RFM Analysis

 : fararizkhi@gmail.com

 : linkedin.com/in/faraniaa

 : github.com/faraniaa