Python初級數據分析員證書

（六）數據分析 及可視化專案

# 13. 數據分析專案 - Demo 12 – Ecommerce Shipping Data Part 1

# Review

- Statistics
- Hypothesis testing
- Algebra
- Linear regression
- Propositional logic
- Python
- R
- SQL

- Pandas, NumPy, SciPy
- Data Visualization, Matplotlib, Seaborn, Plotly
- Dashboard Visualization, Business Intelligence
- Storytelling

## Chapter Summary

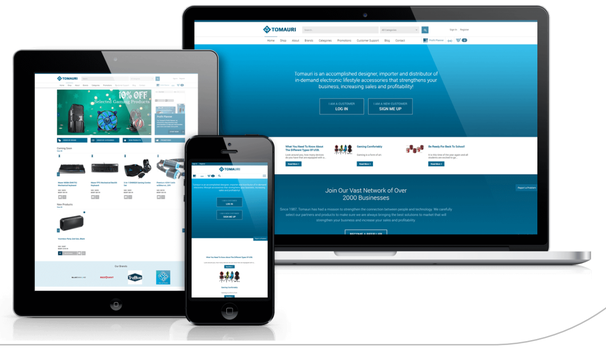
- Scenario
- Data Import
- df.select_dtypes()
- Missingno
- Relation of Shipment on time vs Categoric variables
- Relation of Shipment on time vs Numeric variables

# Scenario

An e-commerce company sells electronic products wants to discover key insights from their customer shipment database. The dataset used for model building contained 10,999 observations of 12 variables.

The goal is to:

- investigate the shipment on time data and
- improve the services and increase sales in future.



# Data

## The data contains the following information:

- **ID**: ID Number of Customers.
- **Warehouse block**: The Company have big Warehouse which is divided in to block such as A,B,C,D,E.
- **Mode of shipment**: The Company Ships the products in multiple way such as Ship, Flight and Road.
- **Customer care calls**: The number of calls made from enquiry for enquiry of the shipment.
- **Customer rating**: The company has rated from every customer. 1 is the lowest (Worst), 5 is the highest (Best).
- **Cost of the product**: Cost of the Product in US Dollars.

# Data

- **Prior purchases**: The Number of Prior Purchase.
- **Product importance**: The company has categorized the product in the various parameter such as low, medium, high.
- **Gender**: Male and Female.
- **Discount offered**: Discount offered on that specific product.
- **Weight in gms**: It is the weight in grams.
- **Reached on time**: It is the **target variable**, where **1** Indicates that the product has NOT reached on time and **0** indicates it has reached on time.

# Data import

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import plotly.express as px
6 import missingno as msno
```

```
1 df = pd.read_csv('EcomShipping.csv')
2 df.head(5).T
```

Import CSV as usual and **transpose**

the head 5 data.

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **ID** | 1 | 2 | 3 | 4 | 5 |
| **Warehouse_block** | D | F | A | B | C |
| **Mode_of_Shipment** | Flight | Flight | Flight | Flight | Flight |
| **Customer_care_calls** | 4 | 4 | 2 | 3 | 2 |
| **Customer_rating** | 2 | 5 | 2 | 3 | 2 |
| **Cost_of_the_Product** | 177 | 216 | 183 | 176 | 184 |
| **Prior_purchases** | 3 | 2 | 4 | 4 | 3 |
| **Product_importance** | low | low | low | medium | medium |
| **Gender** | F | M | M | M | F |
| **Discount_offered** | 44 | 59 | 48 | 10 | 46 |
| **Weight_in_gms** | 1233 | 3088 | 3374 | 1177 | 2484 |
| **Reached.on.Time_Y.N** | 1 | 1 | 1 | 1 | 1 |

# Glimpse at categoric raw data

```
# Categoric data
df.describe(include='O')
```

| | Warehouse_block | Mode_of_Shipment | Product_importance | Gender |
|---|---|---|---|---|
| **count** | 10999 | 10999 | 10999 | 10999 |
| **unique** | 5 | 3 | 3 | 2 |
| **top** | F | Ship | low | F |
| **freq** | 3666 | 7462 | 5297 | 5545 |

## Print out the categoric columns and its category

```
for c in df.select_dtypes(exclude='number').columns.tolist():
    print(c, sorted(df[c].unique()))
```

```
Warehouse_block ['A', 'B', 'C', 'D', 'F']
Mode_of_Shipment ['Flight', 'Road', 'Ship']
Product_importance ['high', 'low', 'medium']
Gender ['F', 'M']
```

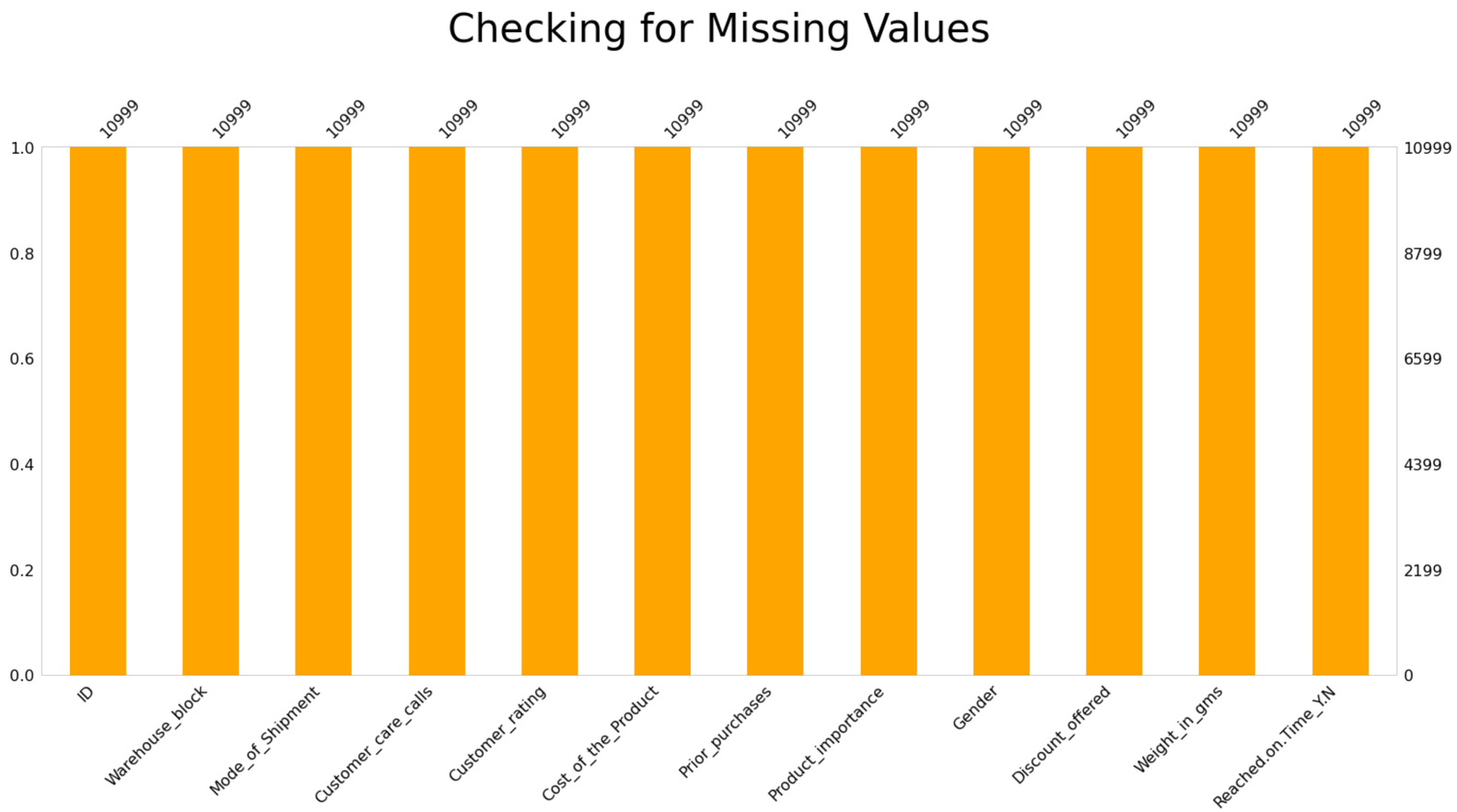
# Glimpse at numeric raw data

```
1 # Numeric data
2 df.describe()
```

| | ID | Customer_care_calls | Customer_rating | Cost_of_the_Product | Prior_purchases | Discount_offered | Weight_in_gms | Reached.on.Time_Y.N |
|---|---|---|---|---|---|---|---|---|
| **count** | 10999.00000 | 10999.000000 | 10999.000000 | 10999.000000 | 10999.000000 | 10999.000000 | 10999.000000 | 10999.000000 |
| **mean** | 5500.00000 | 4.054459 | 2.990545 | 210.196836 | 3.567597 | 13.373216 | 3634.016729 | 0.596691 |
| **std** | 3175.28214 | 1.141490 | 1.413603 | 48.063272 | 1.522860 | 16.205527 | 1635.377251 | 0.490584 |
| **min** | 1.00000 | 2.000000 | 1.000000 | 96.000000 | 2.000000 | 1.000000 | 1001.000000 | 0.000000 |
| **25%** | 2750.50000 | 3.000000 | 2.000000 | 169.000000 | 3.000000 | 4.000000 | 1839.500000 | 0.000000 |
| **50%** | 5500.00000 | 4.000000 | 3.000000 | 214.000000 | 3.000000 | 7.000000 | 4149.000000 | 1.000000 |
| **75%** | 8249.50000 | 5.000000 | 4.000000 | 251.000000 | 4.000000 | 10.000000 | 5050.000000 | 1.000000 |
| **max** | 10999.00000 | 7.000000 | 5.000000 | 310.000000 | 10.000000 | 65.000000 | 7846.000000 | 1.000000 |

# Data cleaning (check)

```
1 msno.bar(df, color = 'orange')
2 plt.title('Checking for Missing Values\n', fontsize = 40)
3 plt.show()
```



# Correlation

```
1  plt.figure(figsize = (18, 7))
2  sns.heatmap(df.select_dtypes(include='number').corr(), annot=True, fmt='0.2f',
3              annot_kws={'size': 15}, linewidth=2, linecolor='orange')
4  plt.show()
```

# Conclusions from Correlation matrix

- Discount Offered have high positive correlation with Reached on Time or Not of 40%.
- Weights in gram have negative correlation with Reached on Time or Not -27%.
- Discount Offered and weights in grams have negative correlation -38%.
- Customer care calls and weights in grams have negative correlation -28%.
- Customer care calls and cost of the product have positive correlation of 32%.

# Check value count on columns

Plot these 8 columns and their categories counts

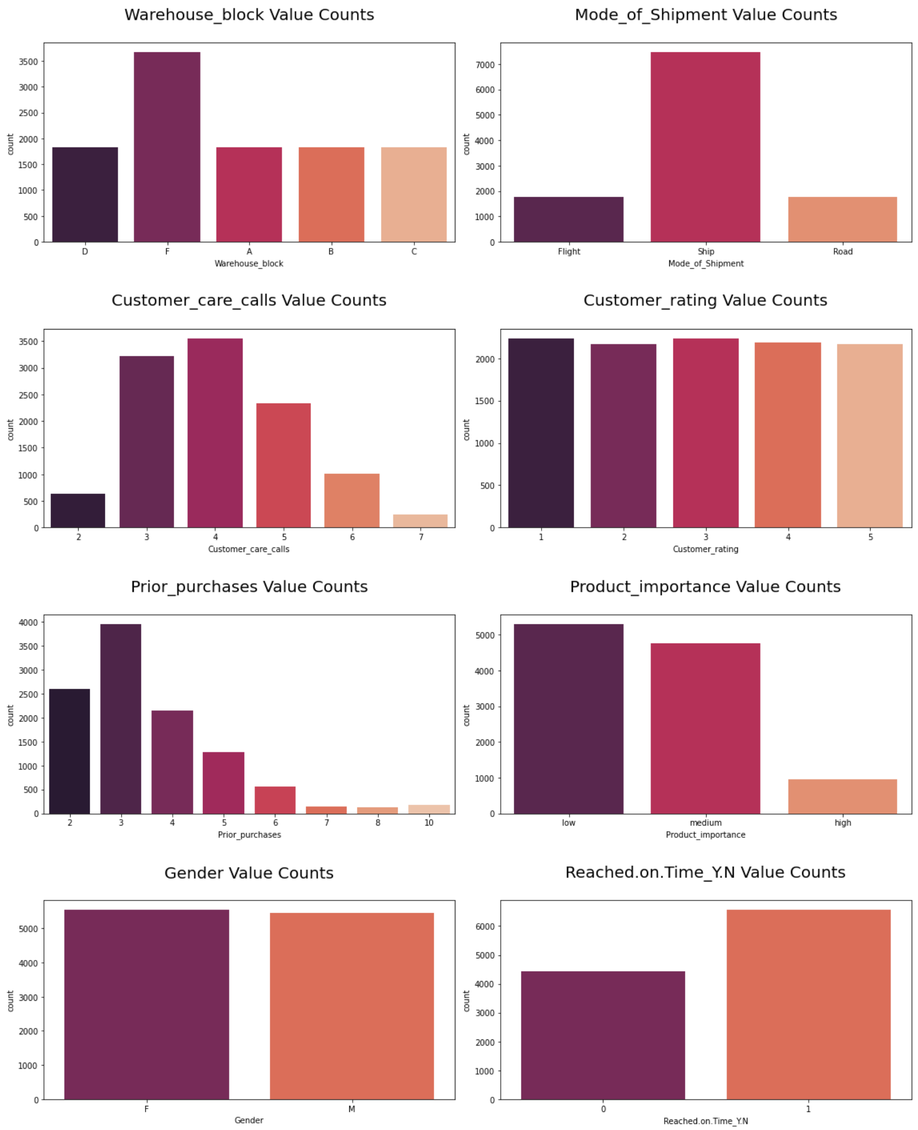
```
# create columns list to check
cols = ['Warehouse_block', 'Mode_of_Shipment', 'Customer_care_calls', 'Customer_rating',
        'Prior_purchases', 'Product_importance', 'Gender', 'Reached.on.Time_Y.N']

plt.figure(figsize = (16, 20))

# plotting the countplot of each categorical column.
for i, col in enumerate(cols):
    if i <= 8:
        ax = plt.subplot(4, 2, i+1)
        sns.countplot(x = col, data = df, ax = ax, palette='rocket')
        plt.title(f"\n{col} Value Counts\n", fontsize = 20)

plt.tight_layout()
plt.show()
```

# Check value count on categoric columns



# Check value count on columns

**From the above plots, we can conclude following:-**

- Warehouse block F have has more values than all other Warehouse blocks.
- In mode of shipment columns we can clearly see that ship delivers the most of products to the customers.
- Most of the customers calls 3 or 4 times to the customer care centres.
- Customer Ratings does not have much differences.
- Most of the customers have 3 prior purchases.
- We can say that most of the products are of low Importance.
- Gender Column doesn't have much differences.
- More products doesn't reach on time than products reached on time.

# Exploring relation of categorical columns

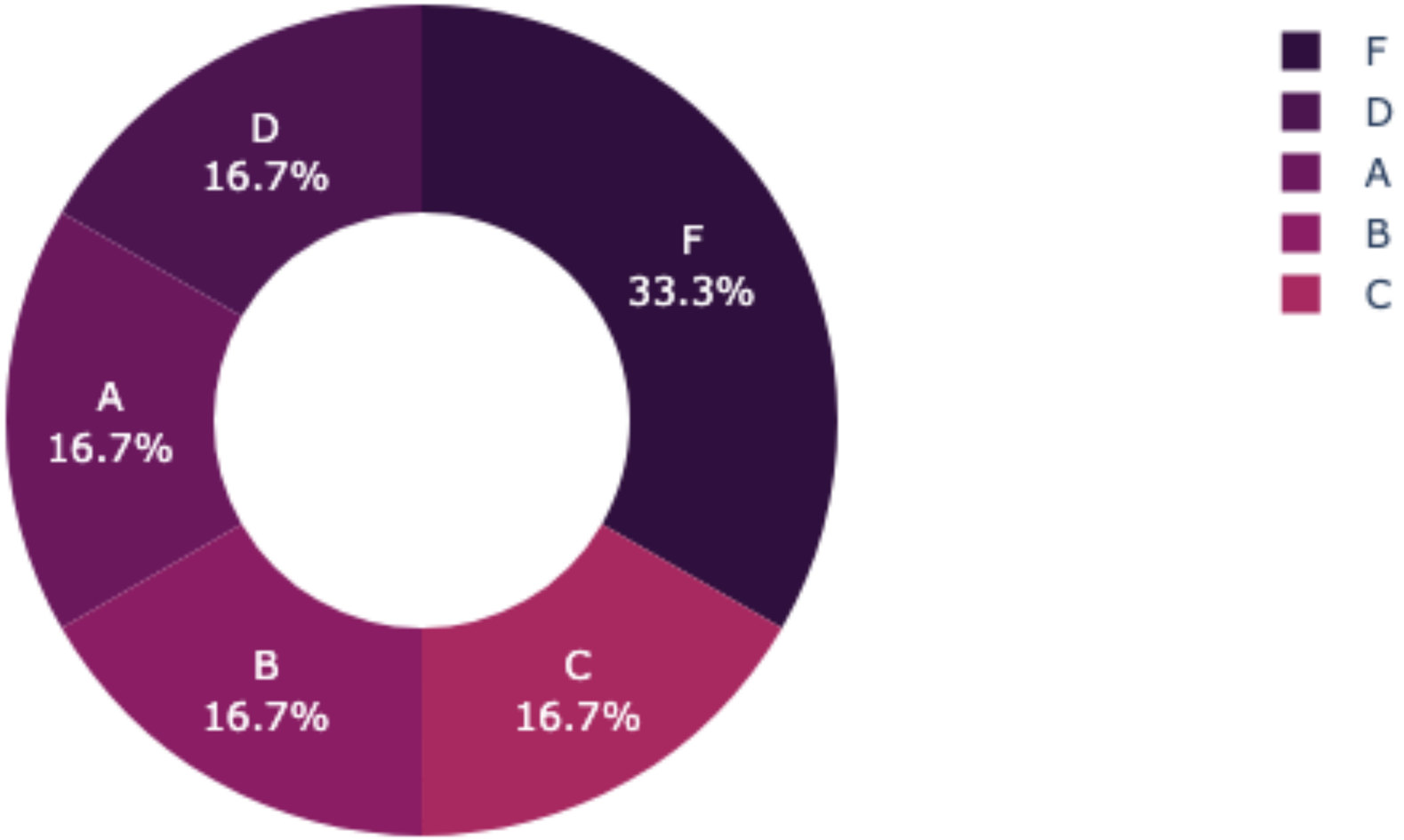
```
object_columns = df.select_dtypes(include = ['object'])
object_columns.sample(5)
```

| | Warehouse_block | Mode_of_Shipment | Product_importance | Gender |
|---|---|---|---|---|
| **3188** | A | Ship | medium | M |
| **2094** | D | Ship | medium | M |
| **156** | D | Ship | low | M |
| **5900** | A | Flight | low | M |
| **1447** | F | Ship | low | M |

## Warehouse column and what are the categories proportion

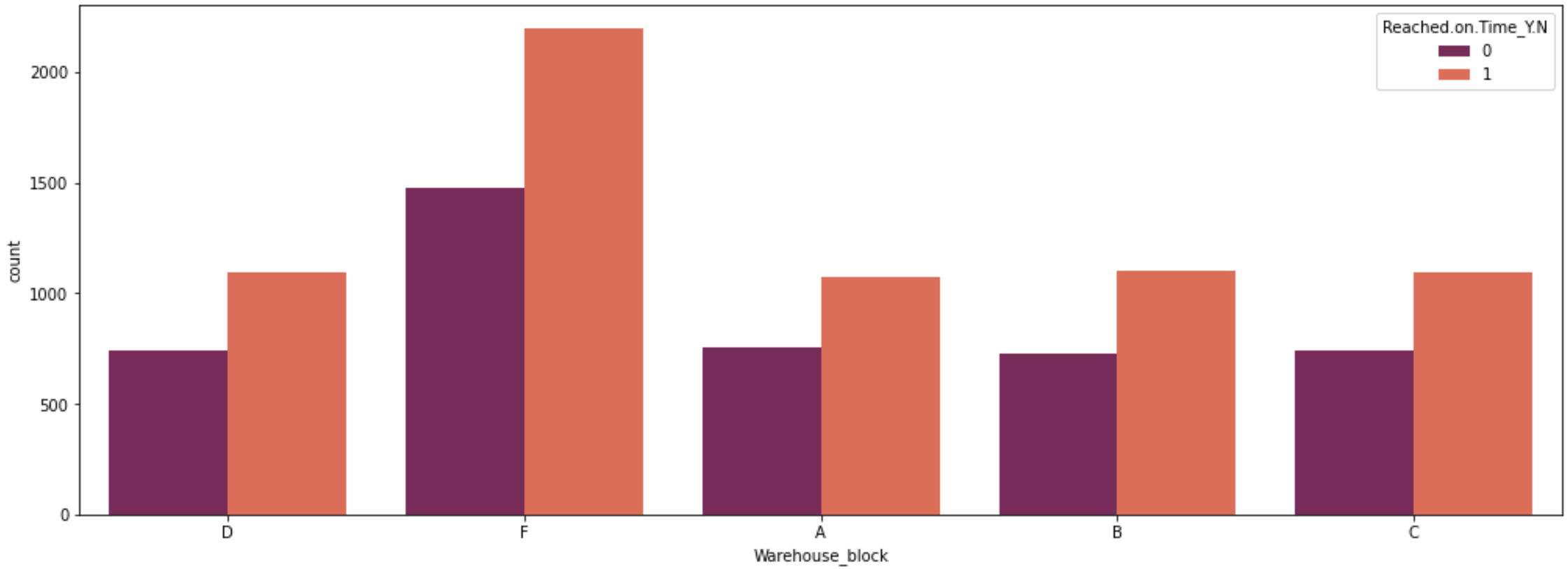
```
warehouse = object_columns['Warehouse_block'].value_counts().reset_index()
warehouse.columns = ['warehouse', 'value_counts']
fig = px.pie(warehouse, names = 'warehouse', values = 'value_counts',
             color_discrete_sequence = px.colors.sequential.matter_r, width = 650, height = 400,
             hole = 0.5)
fig.update_traces(textinfo = 'percent+label')
```

# Warehouse column and categories proportion



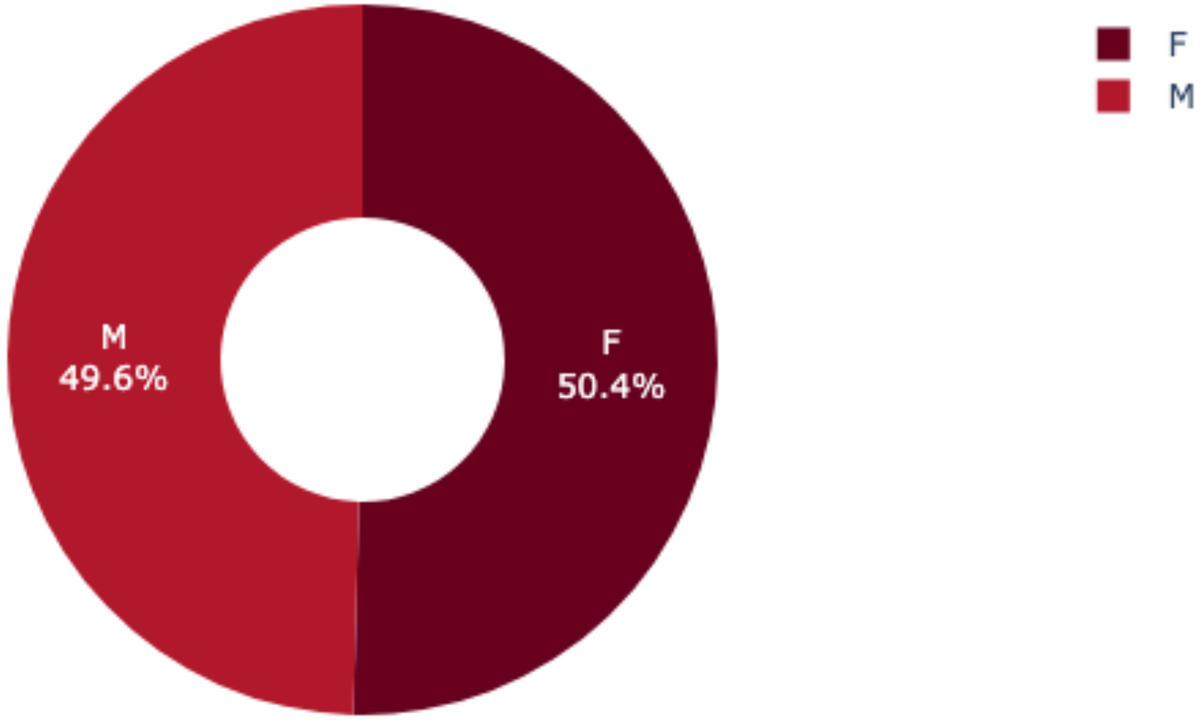
# Reach on time count in each warehouse

```
1  # 1 : NOT on time and 0: on time
2  plt.figure(figsize = (17, 6))
3  sns.countplot(data = df, x='Warehouse_block', hue = 'Reached.on.Time_Y.N',  palette='rocket')
4  plt.show()
```



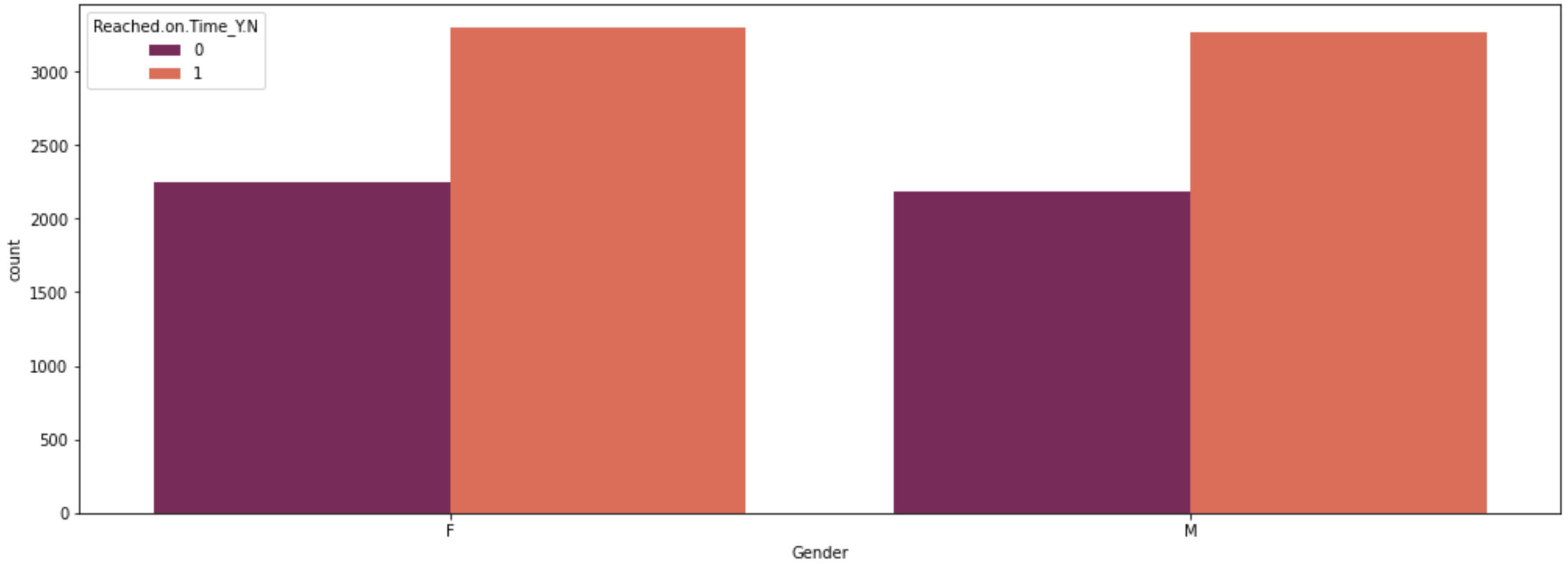
# Gender

```
gender = object_columns['Gender'].value_counts().reset_index()
gender.columns = ['Gender', 'value_counts']
fig = px.pie(gender, names = 'Gender', values = 'value_counts',
             width = 650, height = 400, hole = 0.4,
             color_discrete_sequence=px.colors.sequential.RdBu)
fig.update_traces(textinfo = 'percent+label')
```



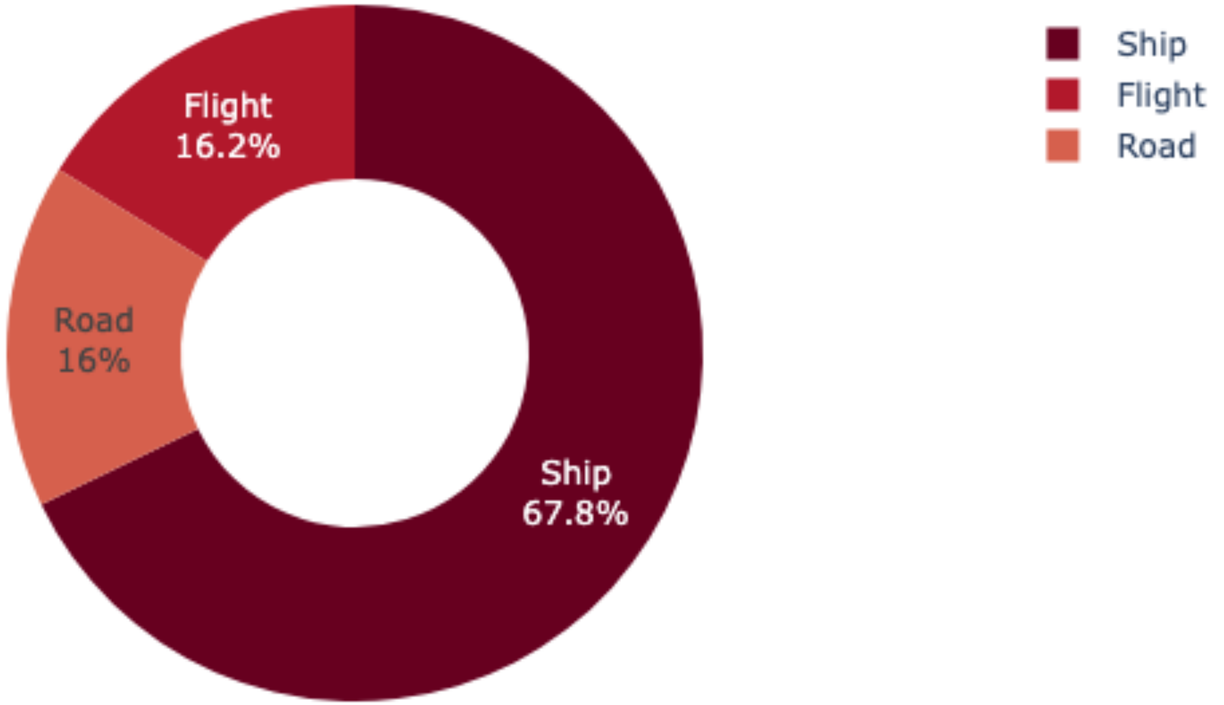
# Gender

```
1  # 1 : NOT on time and 0: on time
2  plt.figure(figsize = (17, 6))
3  sns.countplot(x='Gender', hue = 'Reached.on.Time_Y.N', data = df, palette='rocket')
4  plt.show()
```



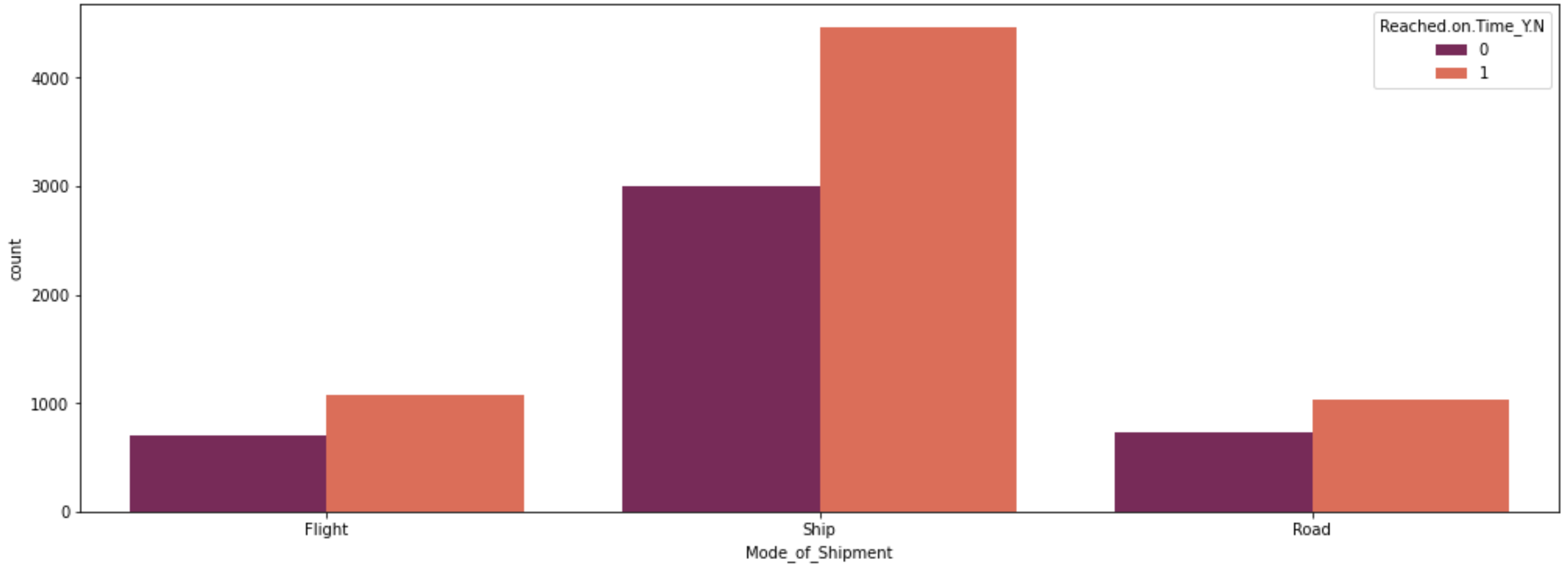
# Shipping method

```
mode = object_columns['Mode_of_Shipment'].value_counts().reset_index()
mode.columns = ['Mode_of_Shipment', 'value_counts']
fig = px.pie(mode, names = 'Mode_of_Shipment', values = 'value_counts',
             color_discrete_sequence = px.colors.sequential.RdBu,
             width = 650, height = 400, hole = 0.5)
fig.update_traces(textinfo = 'percent+label')
```



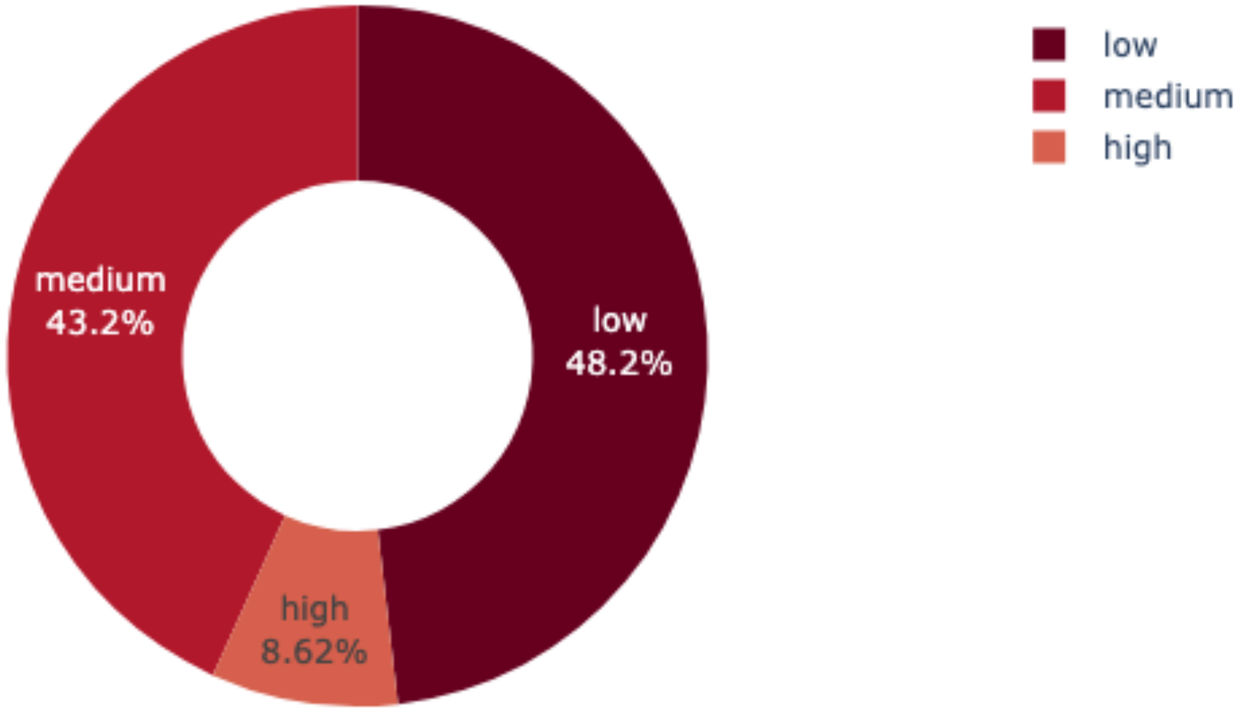
# Reach on time vs Shipping method

```
plt.figure(figsize = (17, 6))
sns.countplot(x='Mode_of_Shipment', hue = 'Reached.on.Time_Y.N',
              data = df, palette='rocket')
plt.show()
```



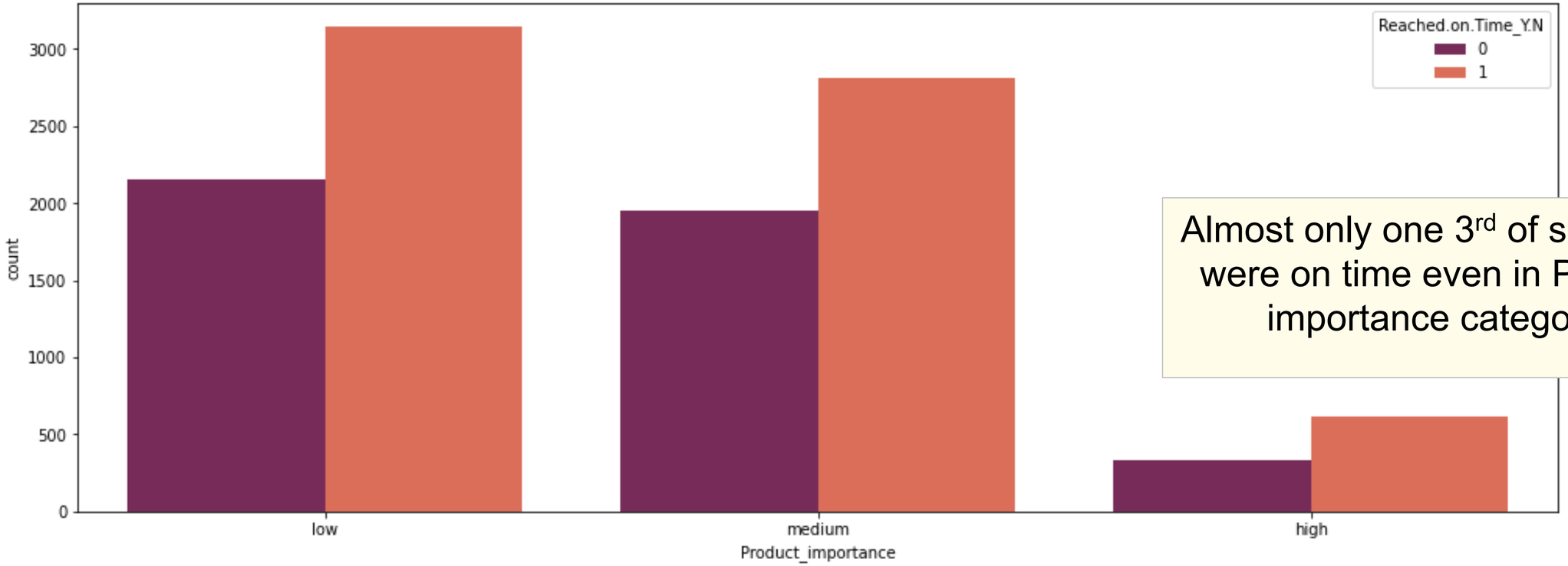
# Product importance

```
product_imp = object_columns['Product_importance'].value_counts().reset_index()
product_imp.columns = ['Product_importance', 'value_counts']
fig = px.pie(product_imp, names = 'Product_importance', values = 'value_counts',
             color_discrete_sequence = px.colors.sequential.RdBu,
             width = 650, height = 400, hole = 0.5)
fig.update_traces(textinfo = 'percent+label')
```



# Product importance vs Shipment on time

```
# 1 : NOT on time and 0: on time
plt.figure(figsize = (17, 6))
sns.countplot(x='Product_importance', hue = 'Reached.on.Time_Y.N',
              data = df, palette='rocket')
plt.show()
```




Almost only one 3rd of shipment were on time even in Product importance category!

# Relation of continuous columns /w on time or not

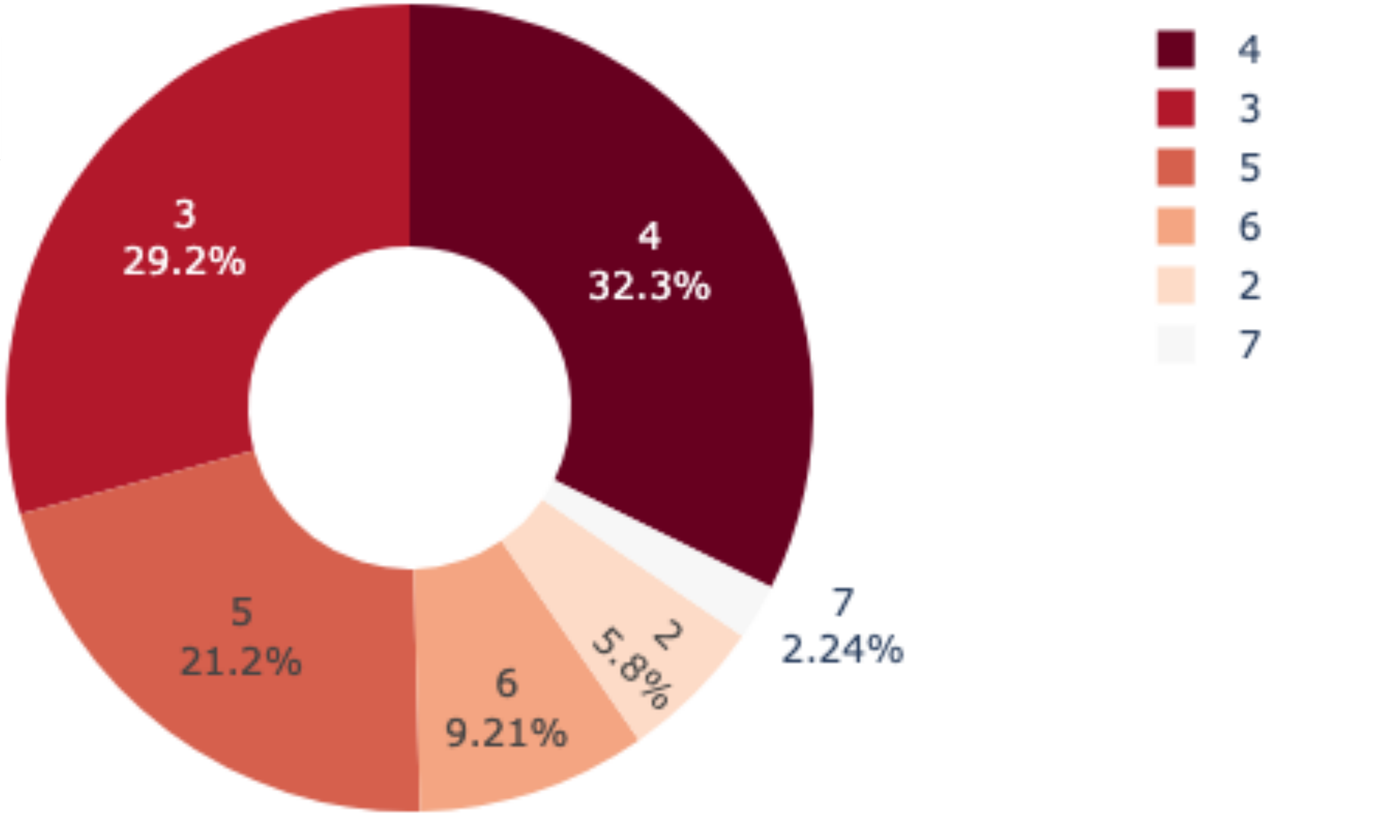
```
integer_columns = df.select_dtypes(include = ['int64'])
integer_columns.head()
```

| | ID | Customer_care_calls | Customer_rating | Cost_of_the_Product | Prior_purchases | Discount_offered | Weight_in_gms | Reached.on.Time_Y.N |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 4 | 2 | 177 | 3 | 44 | 1233 | 1 |
| **1** | 2 | 4 | 5 | 216 | 2 | 59 | 3088 | 1 |
| **2** | 3 | 2 | 2 | 183 | 4 | 48 | 3374 | 1 |
| **3** | 4 | 3 | 3 | 176 | 4 | 10 | 1177 | 1 |
| **4** | 5 | 2 | 2 | 184 | 3 | 46 | 2484 | 1 |

# Customer_care Calls

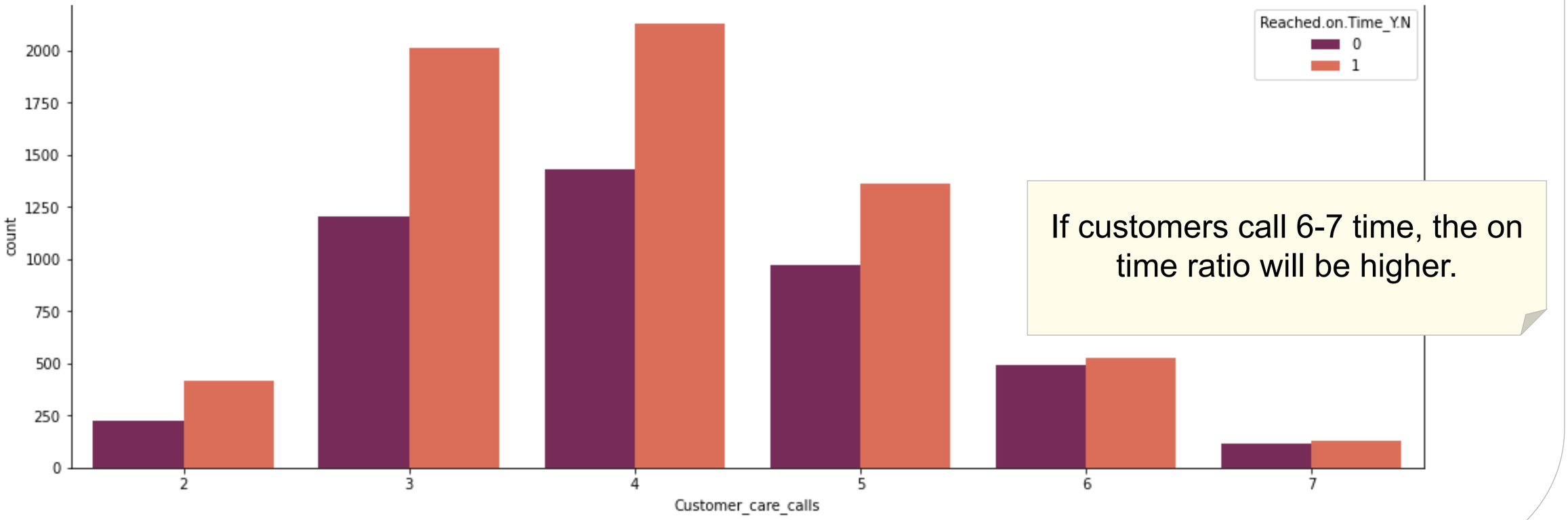
```
customer_care = integer_columns['Customer_care_calls'].value_counts().reset_index()
customer_care.columns = ['Customer_care_calls', 'value_counts']
fig = px.pie(customer_care, names = 'Customer_care_calls',
             values = 'value_counts', width = 650, height = 400,
             color_discrete_sequence = px.colors.sequential.RdBu, hole = 0.4)
fig.update_traces(textinfo = 'percent+label')
```

Customers call quite often



# Customer call vs Shipment on time

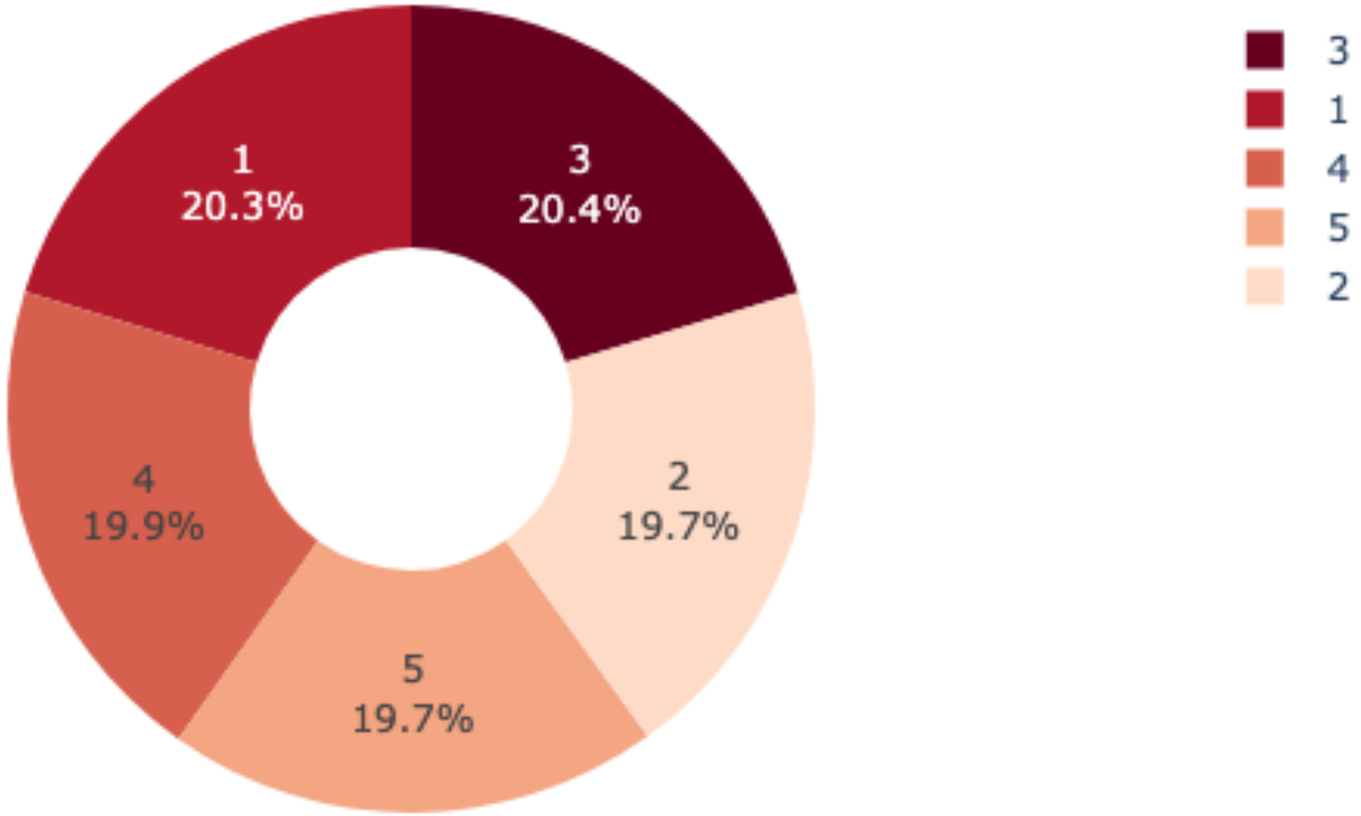
```
1 plt.figure(figsize = (17, 6))
2 sns.countplot(data = df, x='Customer_care_calls',
3                 palette='rocket', hue = 'Reached.on.Time_Y.N')
4 plt.show()
```




If customers call 6-7 time, the on time ratio will be higher.

# Customers rating

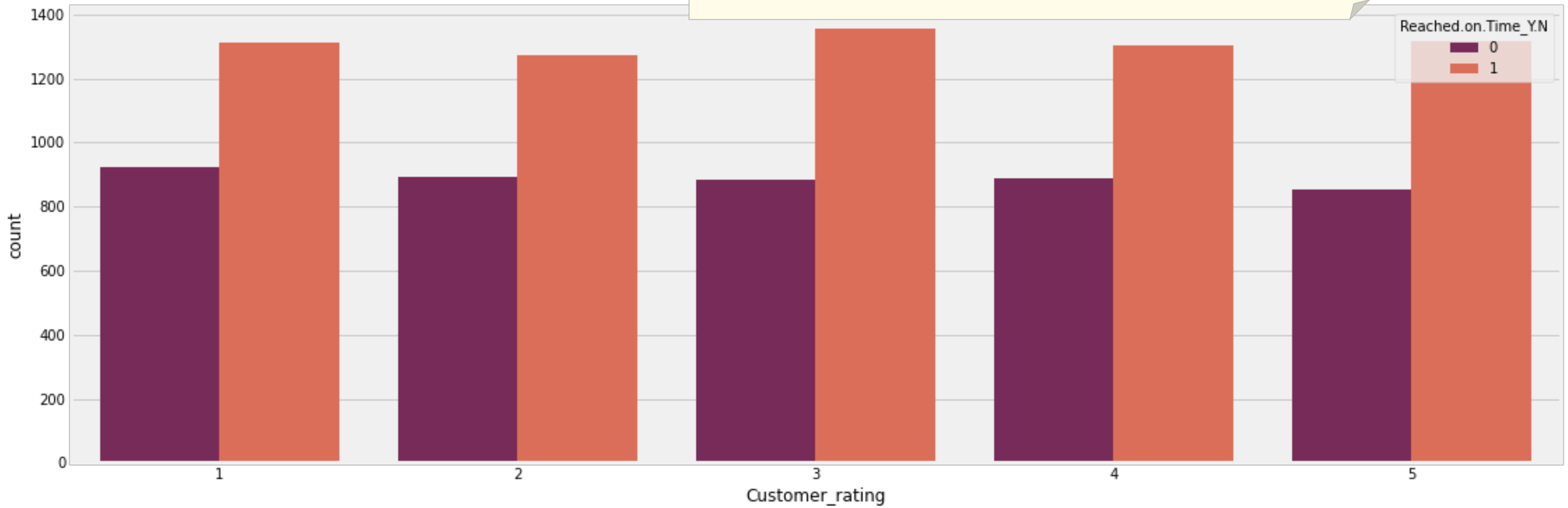
```
customer_ratings = integer_columns['Customer_rating'].value_counts().reset_index()
customer_ratings.columns = ['Customer_rating', 'value_counts']
fig = px.pie(customer_ratings, names = 'Customer_rating', values = 'value_counts',
             color_discrete_sequence = px.colors.sequential.RdBu,
             width = 650, height = 400, hole = 0.4)
fig.update_traces(textinfo = 'percent+label')
```



# Customers rating vs shipment on time

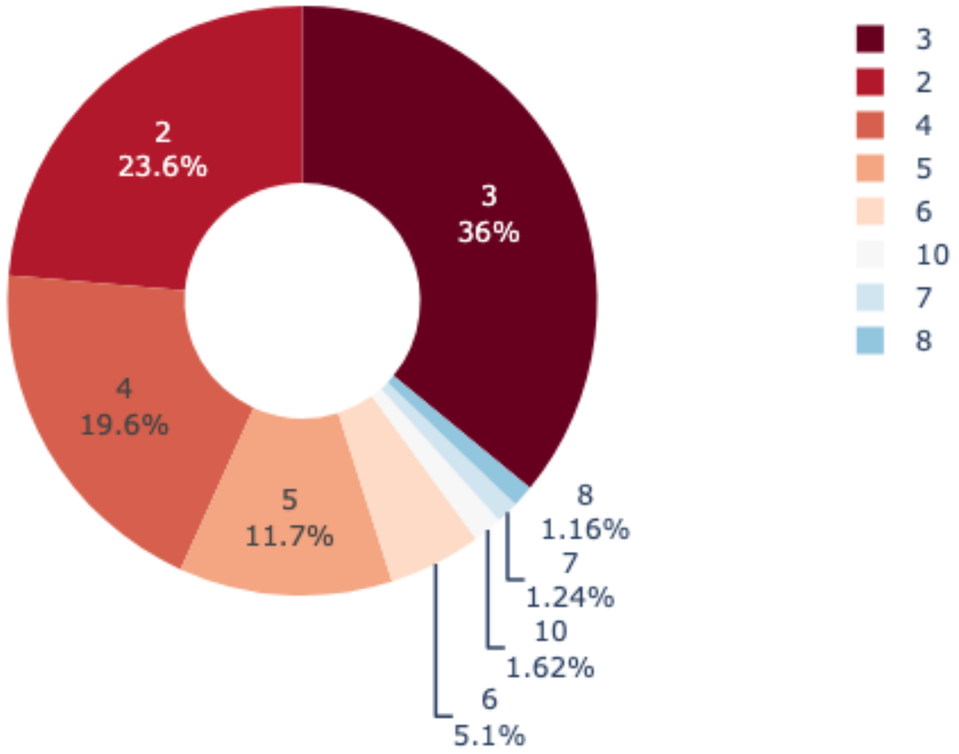
```
1 plt.figure(figsize = (17, 6))
2 sns.countplot(x='Customer_rating', hue = 'Reached.on.Time_Y.N',
3                 data = df, palette='rocket')
4 plt.show()
```

Looks like the rating doesn't reflect the shipment on time.



# Prior purchase and its categories

```
prior_purchases = integer_columns['Prior_purchases'].value_counts().reset_index()
prior_purchases.columns = ['Prior_purchases', 'value_counts']
fig = px.pie(prior_purchases, names = 'Prior_purchases', values = 'value_counts',
             color_discrete_sequence = px.colors.sequential.RdBu,
             width = 650, height = 400, hole = 0.4)
fig.update_traces(textinfo = 'percent+label')
```

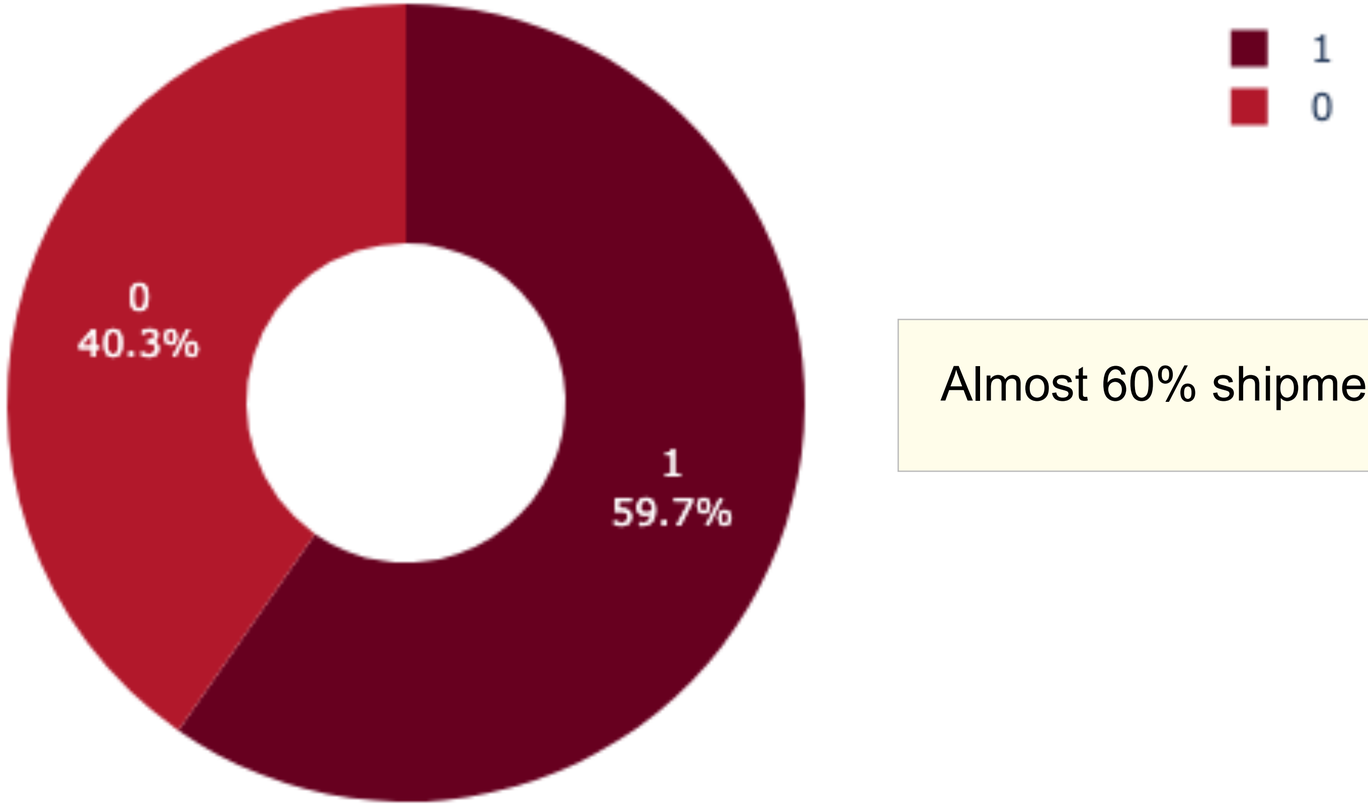


# Prior purchase vs Shipment on time

```
# 1 : NOT on time and 0: on time
plt.figure(figsize = (17, 6))
sns.countplot(x='Prior_purchases', hue = 'Reached.on.Time_Y.N', data = df, palette='rocket')
plt.show()
```

# Reach on time vs Not

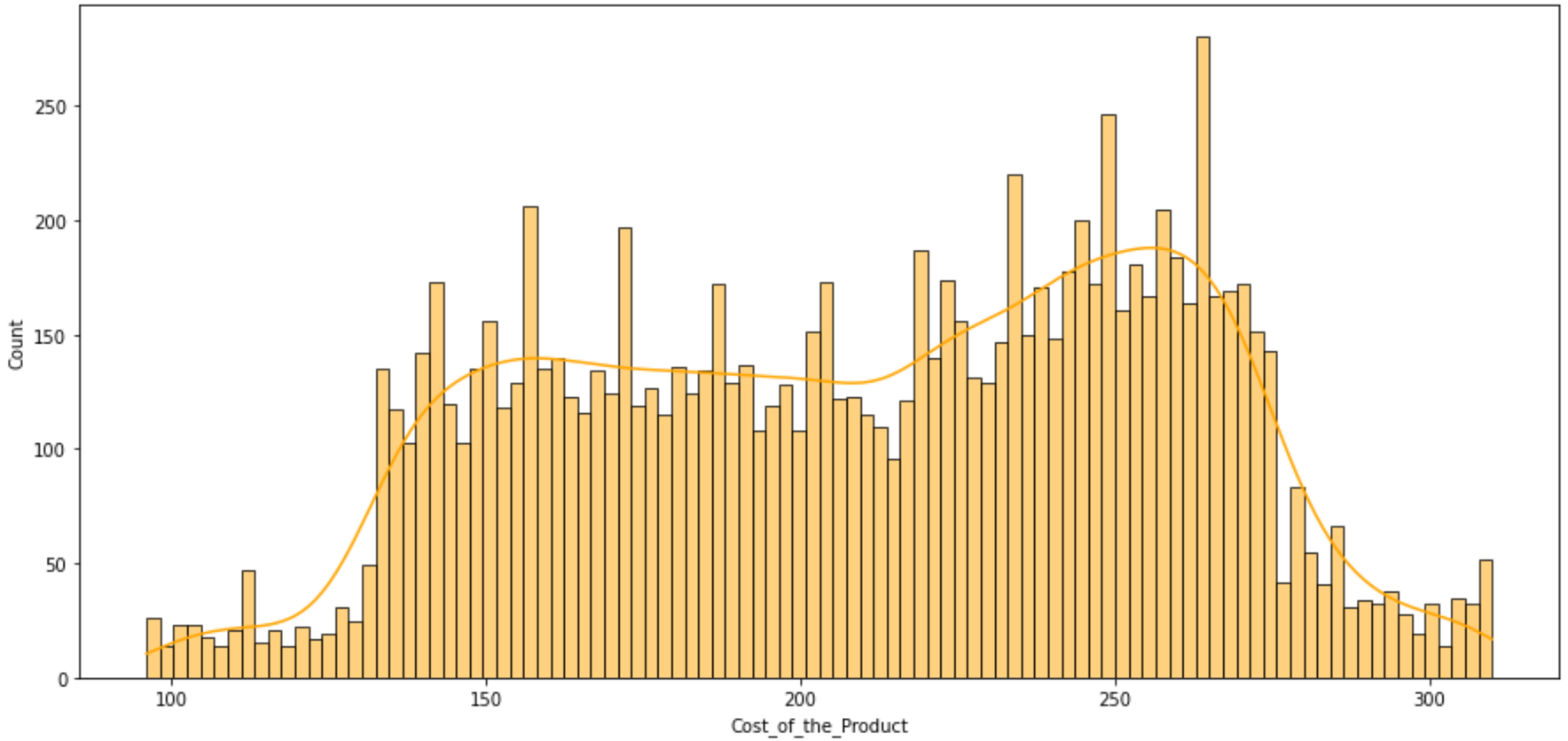
```
# 1 : NOT on time and 0: on time
reached_on_time_y_n = integer_columns['Reached.on.Time_Y.N'].value_counts().reset_index()
reached_on_time_y_n.columns = ['Reached.on.Time_Y.N', 'value_counts']
fig = px.pie(reached_on_time_y_n, names = 'Reached.on.Time_Y.N', values = 'value_counts',
             color_discrete_sequence = px.colors.sequential.RdBu,
             width = 650, height = 400, hole = 0.4)
fig.update_traces(textinfo = 'percent+label')
```




Almost 60% shipment is Not on time

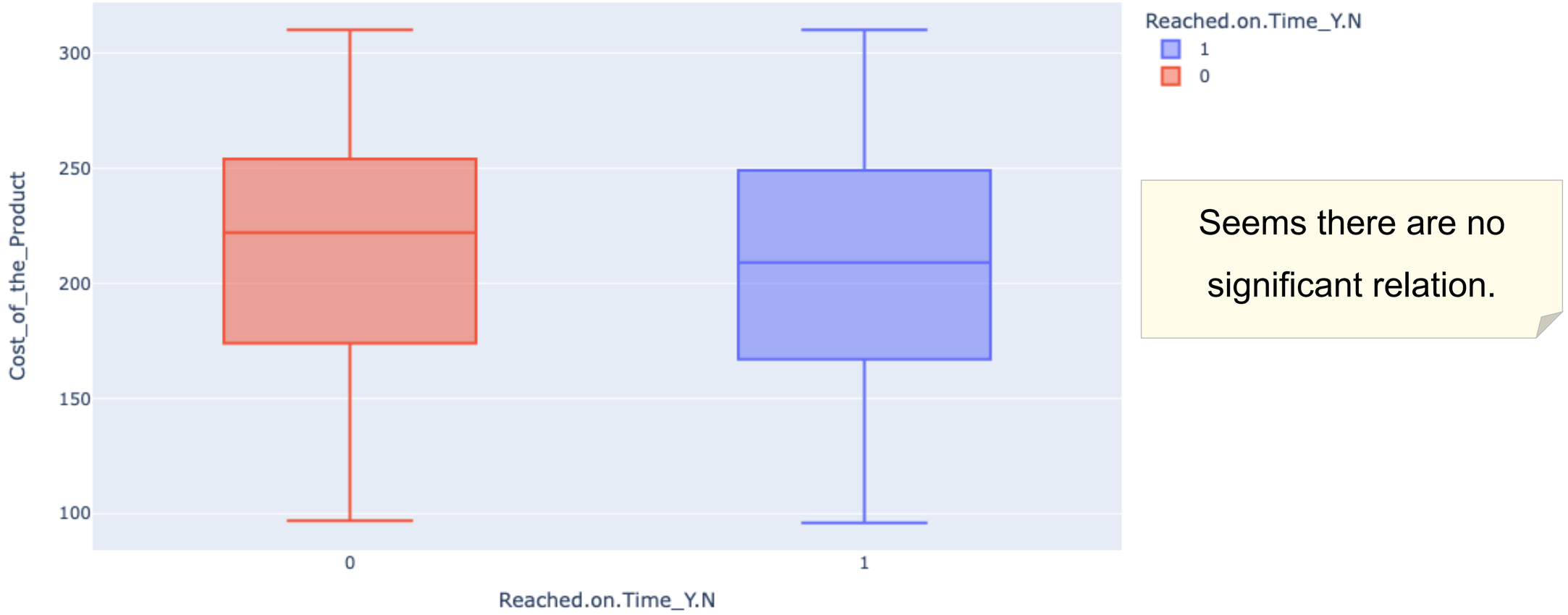
# Cost of Product

```
1 plt.figure(figsize = (15, 7))
2 ax = sns.histplot(df['Cost_of_the_Product'], bins = 100, color = 'orange', kde=True)
3
4 plt.show()
```



# Relation of Cost of product vs Shipment on time

```
1  # 1 : NOT on time and 0: on time
2  px.box(data_frame = df, x = 'Reached.on.Time_Y.N', y = 'Cost_of_the_Product',
3         color = 'Reached.on.Time_Y.N' )
```




Seems there are no significant relation.

# Reference & Resources

Official Website:

https://plotly.com/python/

Plotly Graph Objects:

https://plotly.com/python/graph-objects/

Seaborn:

https://seaborn.pydata.org/examples/index.html

LogisticRegression:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html