Data Collection and Preprocessing Phase

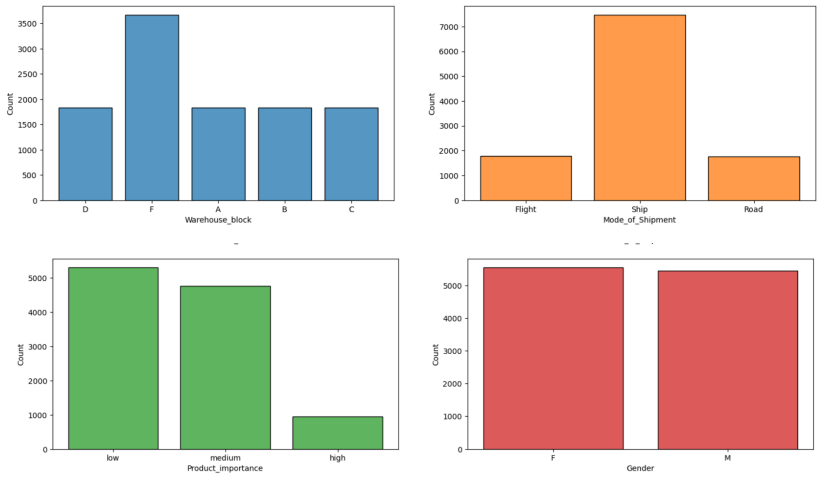
Date	16 July 2024
Team ID	SWTID1720190389
Project Title	E-Commerce Shipping Prediction
Maximum Marks	6 Marks

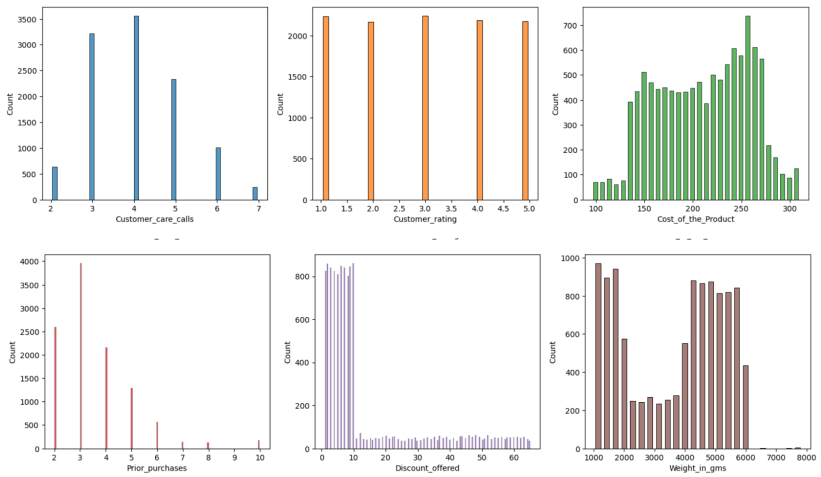
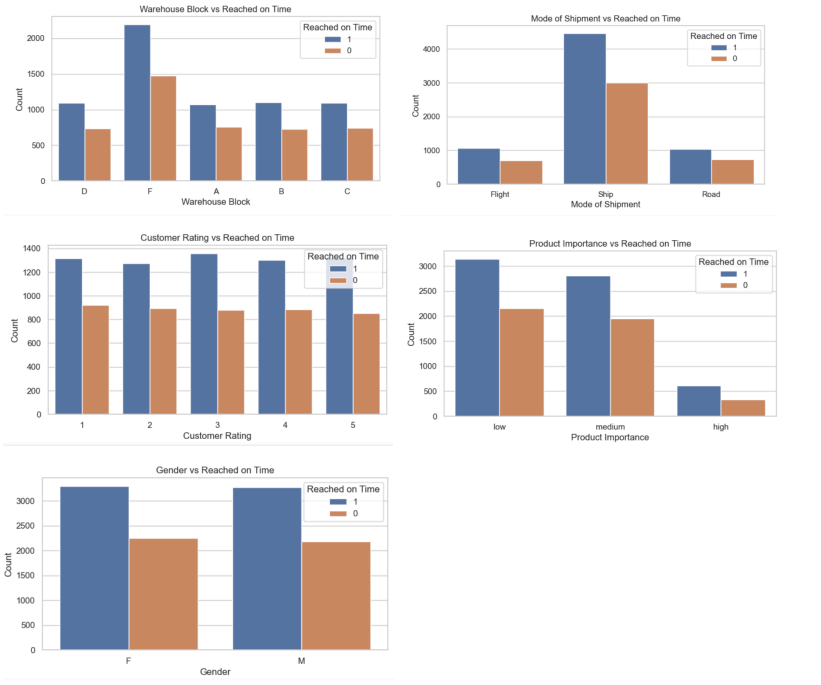
Data Exploration and Preprocessing Template

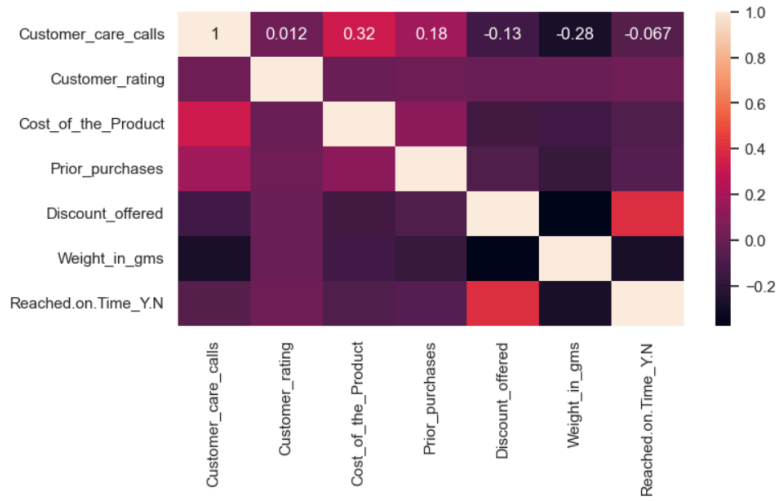
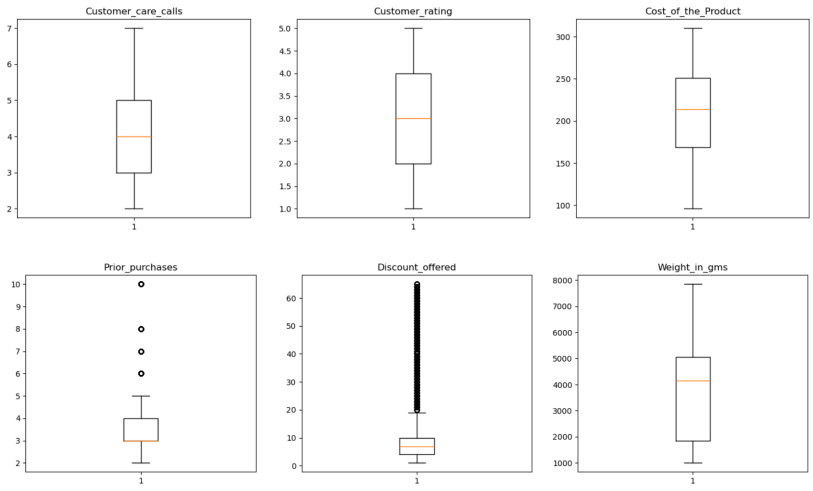
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	Description

Data Overview	`#Descriptive statistics`
	`data.describe(include='all')`
	
	Warehouse_block / Mode_of_Shipment / Customer_care_calls / Customer_rating / Cost_of_the_Product / Prior_purchases / Product_importance / Gender / Discount_offered / Weight_in_gms / Reached.on.Time_Y.N
	count: 10999 / 10999 / 10999.000000 / 10999.000000 / 10999.000000 / 10999.000000 / 10999 / 10999 / 10999.000000 / 10999.000000 / 10999.000000
	unique: 5 / 3 / NaN / NaN / NaN / NaN / 3 / 2 / NaN / NaN / NaN
	top: F / Ship / NaN / NaN / NaN / NaN / low / F / NaN / NaN / NaN
	freq: 3666 / 7462 / NaN / NaN / NaN / NaN / 5297 / 5545 / NaN / NaN / NaN
	mean: NaN / NaN / 4.054459 / 2.990545 / 210.196836 / 3.567597 / NaN / NaN / 13.373216 / 3634.016729 / 0.596691
	std: NaN / NaN / 1.141490 / 1.413603 / 48.063272 / 1.522860 / NaN / NaN / 16.205527 / 1635.377251 / 0.490584
	min: NaN / NaN / 2.000000 / 1.000000 / 96.000000 / 2.000000 / NaN / NaN / 1.000000 / 1001.000000 / 0.000000
	25%: NaN / NaN / 3.000000 / 2.000000 / 169.000000 / 3.000000 / NaN / NaN / 4.000000 / 1839.500000 / 0.000000
	50%: NaN / NaN / 4.000000 / 3.000000 / 214.000000 / 3.000000 / NaN / NaN / 7.000000 / 4149.000000 / 1.000000
	75%: NaN / NaN / 5.000000 / 4.000000 / 251.000000 / 4.000000 / NaN / NaN / 10.000000 / 5050.000000 / 1.000000
	max: NaN / NaN / 7.000000 / 5.000000 / 310.000000 / 10.000000 / NaN / NaN / 65.000000 / 7846.000000 / 1.000000

Univariate Analysis	

<table>
<tr>
<td></td>
<td> 
</td>
</tr>
<tr>
<td>Bivariate Analysis</td>
<td> 
</td>
</tr>
</table>

Multivariate Analysis	
Outliers and Anomalies	

```
from sklearn.neighbors import LocalOutlierFactor
DataForA = data.copy()
# to protect main
clf = LocalOutlierFactor()
clf.fit_predict(DataForA)
score = clf.negative_outlier_factor_
scoreSorted = np.sort(score)
print(scoreSorted[0:50])
# checking outlier, look where the biggest jump took place
# we can identify 6.index as point
point = scoreSorted[6]
print(point)
print("---"*10)
print(DataForA[score == point])
```

```
[-49.23420001 -45.83305039 -44.90793785 -43.80881891 -39.87437401
 -23.44503825  -4.7196879   -3.42435176  -3.37610349  -3.20090154
  -3.06223813  -2.86920018  -2.78754054  -2.74031948  -2.5840874
  -2.53241128  -2.51965873  -2.49954423  -2.45206587  -2.22651546
  -2.19794237  -2.14941097  -2.08814005  -2.07886447  -2.05597358
  -2.04178282  -1.85543019  -1.81595265  -1.79030482  -1.78492262
  -1.78367328  -1.78196285  -1.74562079  -1.72837548  -1.71070166
  -1.70775564  -1.69162048  -1.68935852  -1.68902088  -1.68238855
  -1.67774099  -1.66262386  -1.63914729  -1.63399915  -1.63292157
  -1.63241874  -1.62371866  -1.62080133  -1.61929576  -1.61737996]
-4.7196878981741435
------------------------------
     Warehouse_block  Mode_of_Shipment  Customer_care_calls  Customer_rating  \
251                4                 1                    2                2

     Cost_of_the_Product  Prior_purchases  Product_importance  Gender  \
251                  145                3                   2       0

     Discount_offered  Weight_in_gms  Reached.on.Time_Y.N
251                 5           6102                    1
```

```
outliers = score < point
print(data[outliers])
print("---"*20)
print(data[outliers].index)
```

```
     Warehouse_block  Mode_of_Shipment  Customer_care_calls  Customer_rating  \
198                3                 2                    2                3
199                4                 2                    2                2
205                4                 2                    2                3
213                1                 2                    2                5
245                4                 2                    2                4
257                4                 1                    2                2

     Cost_of_the_Product  Prior_purchases  Product_importance  Gender  \
198                  142                3                   2       0
199                  154                3                   2       1
205                  145                3                   2       0
213                  160                3                   2       0
245                  154                3                   2       0
257                  129                3                   2       0

     Discount_offered  Weight_in_gms  Reached.on.Time_Y.N
198                38           7640                    1
199                38           7846                    1
205                24           7588                    1
213                31           7401                    1
245                48           7684                    1
257                22           6614                    1
------------------------------------------------------------
Index([198, 199, 205, 213, 245, 257], dtype='int64')
```

```
#Deleting
outliersIndexList = [data[outliers].index]
print(type(outliersIndexList))
```

```
<class 'list'>
```

```
for d in outliersIndexList:
    data.drop(index=d,inplace=True)
```

```
data.shape
```

```
(10993, 11)
```

Data Preprocessing Code Screenshots

Loading Data

```
data = pd.read_csv('Train.csv')
data.head()
```

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

Handling Missing Data

```
[22]: data.shape
```

```
[22]: (10999, 11)
```

```
[24]: data.isnull().sum()
```

```
[24]: Warehouse_block        0
      Mode_of_Shipment       0
      Customer_care_calls    0
      Customer_rating        0
      Cost_of_the_Product    0
      Prior_purchases        0
      Product_importance     0
      Gender                 0
      Discount_offered       0
      Weight_in_gms          0
      Reached.on.Time_Y.N    0
      dtype: int64
```

Data Transformation

```
#As "ID"column is not necessary we can drop the "ID" column
data.drop("ID",inplace=True,axis=1)
```

Standardization

```
import statsmodels.stats.api as sms
```

```
#for Customer_rating
print(data["Customer_rating"].mode())
print(data["Customer_rating"].max())
print(data["Customer_rating"].min())
print(data["Customer_rating"].mean())
print(sms.DescrStatsW(data["Customer_rating"]).tconfint_mean())
```

```
0    3
Name: Customer_rating, dtype: int64
5
1
2.990448467206404
(2.9640174157017847, 3.016879518710233)
```

```
def func(x):
    if x < 2.99:
        return "BAD"
    else:
        return "GOOD"
```

```
data["Customer_rating"] = data["Customer_rating"].apply(lambda x: func(x))
```

```
print(data["Customer_rating"].value_counts())
```

```
Customer_rating
GOOD    6595
BAD     4398
Name: count, dtype: int64
```

```
data["Customer_rating"] = encode.fit_transform(data["Customer_rating"])
```

```
print(data["Customer_rating"].value_counts())
```

```
Customer_rating
1    6595
0    4398
Name: count, dtype: int64
```

```
#for Discount_offered
print(data["Discount_offered"].mode())
print(data["Discount_offered"].max())
print(data["Discount_offered"].min())
print(data["Discount_offered"].mean())
print(sms.DescrStatsW(data["Discount_offered"]).tconfint_mean())
```

```
0    10
Name: Discount_offered, dtype: int64
65
1
13.362230510324752
(13.059329677044607, 13.665131343604896)
```

```
def funcforD(x):
    if x < 13.36:
        return "LESS"
    elif 13.36 < x > 30:
        return "NORMAL"
    else:
        return "TOO MUCH"
```

```
data["Discount_offered"] = data["Discount_offered"].apply(lambda x: funcforD(x))
```

```
print(data["Discount_offered"].value_counts())
```

```
Discount_offered
LESS        8514
NORMAL      1706
TOO MUCH     773
Name: count, dtype: int64
```

```
data["Discount_offered"] = encode.fit_transform(data["Discount_offered"])
```

```
print(data["Discount_offered"].value_counts())
```

```
Discount_offered
0    8514
1    1706
2     773
Name: count, dtype: int64
```

Feature Engineering

Handling catogorical values(Encoding)

```
from sklearn.preprocessing import LabelEncoder

encode = LabelEncoder()
objectcolumns = data.select_dtypes(include=["object"])
print(objectcolumns.columns)
```

```
Index(['Warehouse_block', 'Mode_of_Shipment', 'Product_importance', 'Gender'], dtype='object')
```

```
for a in objectcolumns:
    data[a] = encode.fit_transform(data[a])
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Warehouse_block      10999 non-null  int32
 1   Mode_of_Shipment     10999 non-null  int32
 2   Customer_care_calls  10999 non-null  int64
 3   Customer_rating      10999 non-null  int64
 4   Cost_of_the_Product  10999 non-null  int64
 5   Prior_purchases      10999 non-null  int64
 6   Product_importance   10999 non-null  int32
 7   Gender               10999 non-null  int32
 8   Discount_offered     10999 non-null  int64
 9   Weight_in_gms        10999 non-null  int64
 10  Reached.on.Time_Y.N  10999 non-null  int64
dtypes: int32(4), int64(7)
memory usage: 773.5 KB
```

Save Processed Data

-