In [1]:

```python
#import numpy ,pandas ,matplotlib,seaborn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```python
# read csv file
df = pd.read_csv('Train.csv')
df
```

Out[3]:

use_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Pric
D	Flight	4	2	177	
F	Flight	4	5	216	
A	Flight	2	2	183	
B	Flight	3	3	176	
C	Flight	2	2	184	
...	...	...	...	...	
A	Ship	4	1	252	
B	Ship	4	1	232	
C	Ship	5	4	242	
F	Ship	5	2	223	
D	Ship	2	5	155	

ıns

In [4]:
```
#drop na values
df.dropna()
```

Out[4]:

	ID	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	C
0	1	D	Flight	4	2	
1	2	F	Flight	4	5	
2	3	A	Flight	2	2	
3	4	B	Flight	3	3	
4	5	C	Flight	2	2	
...	...	...	...	...	...	
10994	10995	A	Ship	4	1	
10995	10996	B	Ship	4	1	
10996	10997	C	Ship	5	4	
10997	10998	F	Ship	5	2	
10998	10999	D	Ship	2	5	

10999 rows × 12 columns

In [5]:
```
# drop duplicates
df.drop_duplicates()
```

Out[5]:

lode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases
Flight	4	2	177	3
Flight	4	5	216	2
Flight	2	2	183	4
Flight	3	3	176	4
Flight	2	2	184	3
...	...	...	...	...
Ship	4	1	252	5
Ship	4	1	232	5
Ship	5	4	242	5
Ship	5	2	223	6
Ship	2	5	155	5

In [6]:

```python
# set index as id
df.set_index('ID')
```

Out[6]:

	Warehouse_block	Mode_of_Shipment	Customer_care_calls	Customer_rating	Cost_of_
ID					
1	D	Flight	4	2	
2	F	Flight	4	5	
3	A	Flight	2	2	
4	B	Flight	3	3	
5	C	Flight	2	2	
...	...	...	...	...	
10995	A	Ship	4	1	
10996	B	Ship	4	1	
10997	C	Ship	5	4	
10998	F	Ship	5	2	
10999	D	Ship	2	5	

10999 rows × 11 columns

In [7]:

```python
# info of the data set
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10999 entries, 0 to 10998
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   ID                   10999 non-null  int64 
 1   Warehouse_block      10999 non-null  object
 2   Mode_of_Shipment     10999 non-null  object
 3   Customer_care_calls  10999 non-null  int64 
 4   Customer_rating      10999 non-null  int64 
 5   Cost_of_the_Product  10999 non-null  int64 
 6   Prior_purchases      10999 non-null  int64 
 7   Product_importance   10999 non-null  object
 8   Gender               10999 non-null  object
 9   Discount_offered     10999 non-null  int64 
 10  Weight_in_gms        10999 non-null  int64 
 11  Reached.on.Time_Y.N  10999 non-null  int64 
dtypes: int64(8), object(4)
memory usage: 1.0+ MB
```

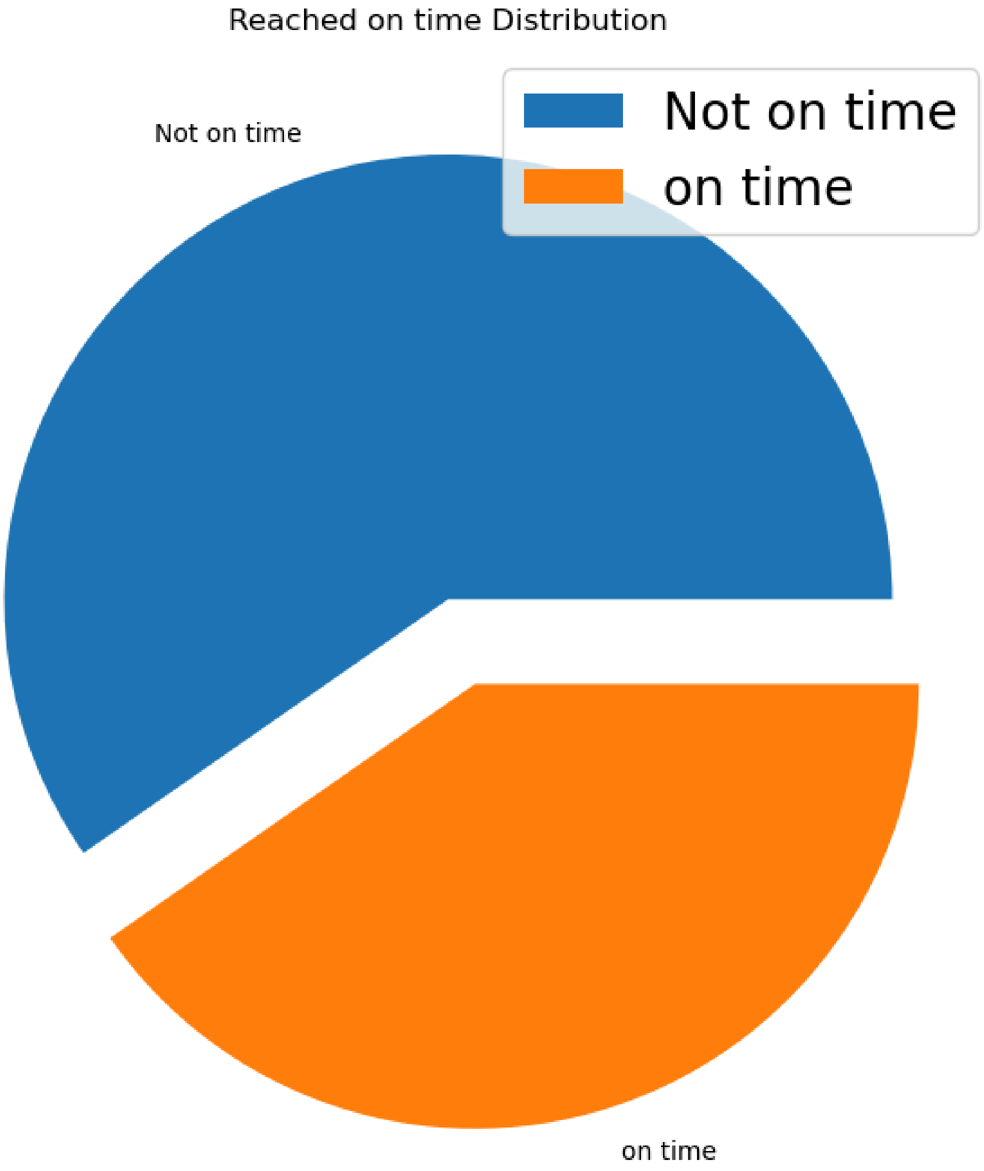
In [8]:

```
# describe data like mean, standard deviation ,max,min
df.describe()
```

Out[8]:

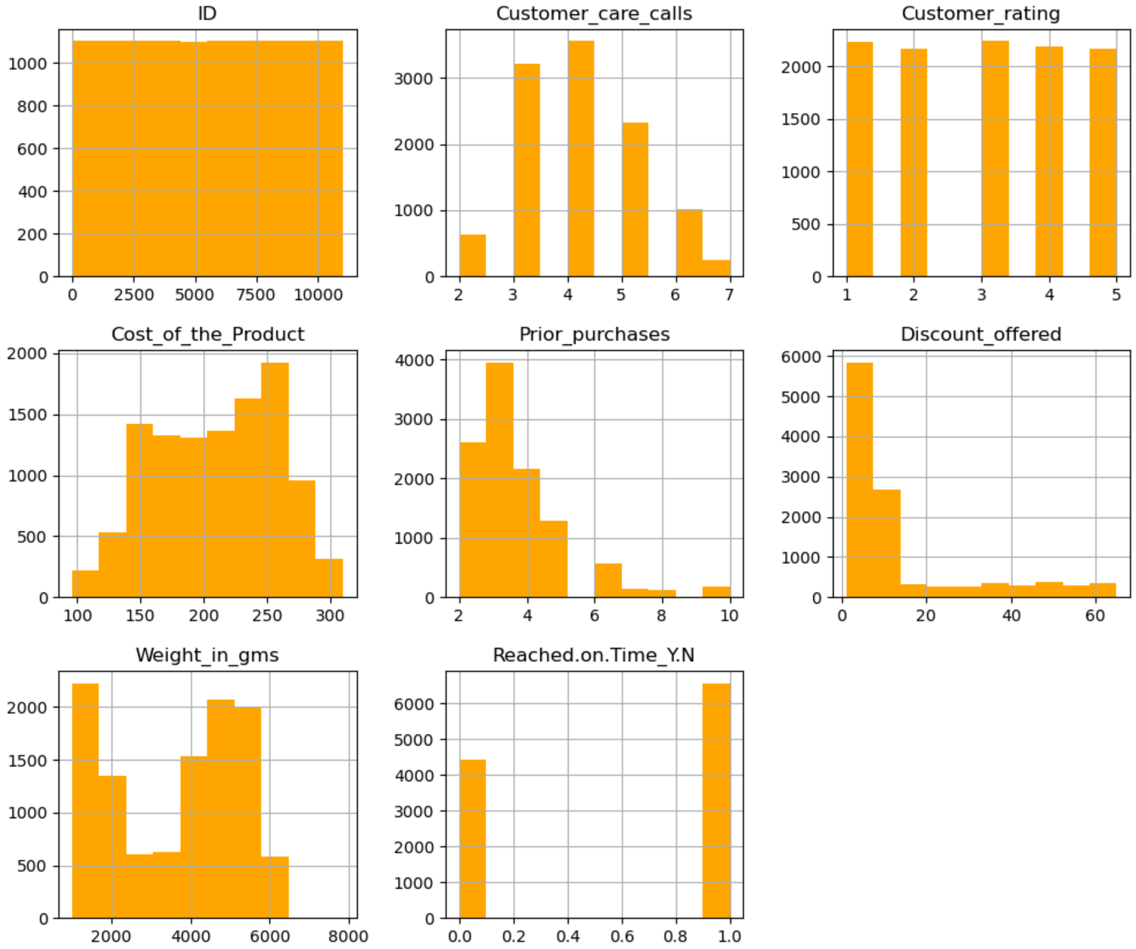
	ID	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purcha
count	10999.00000	10999.000000	10999.000000	10999.000000	10999.000
mean	5500.00000	4.054459	2.990545	210.196836	3.567
std	3175.28214	1.141490	1.413603	48.063272	1.522
min	1.00000	2.000000	1.000000	96.000000	2.000
25%	2750.50000	3.000000	2.000000	169.000000	3.000
50%	5500.00000	4.000000	3.000000	214.000000	3.000
75%	8249.50000	5.000000	4.000000	251.000000	4.000
max	10999.00000	7.000000	5.000000	310.000000	10.000

```
In [23]: # pie chart for on time delivery
         ontime_counts = df['Reached.on.Time_Y.N'].value_counts()
         plt.figure(figsize = (8,8))
         plt.pie(ontime_counts, labels=['Not on time','on time'] ,explode=(0,.2))
         plt.legend(loc = 'upper right', fontsize = '20')
         plt.title('Reached on time Distribution')
         plt.show()
```



In [16]:

```
# histogram for every dataset
df.hist(figsize=(12,10), color='orange')
plt.show()
```



In [46]:

```
# select dtypes only numerical values
noobj = df.select_dtypes(exclude='object')
```

In [49]:

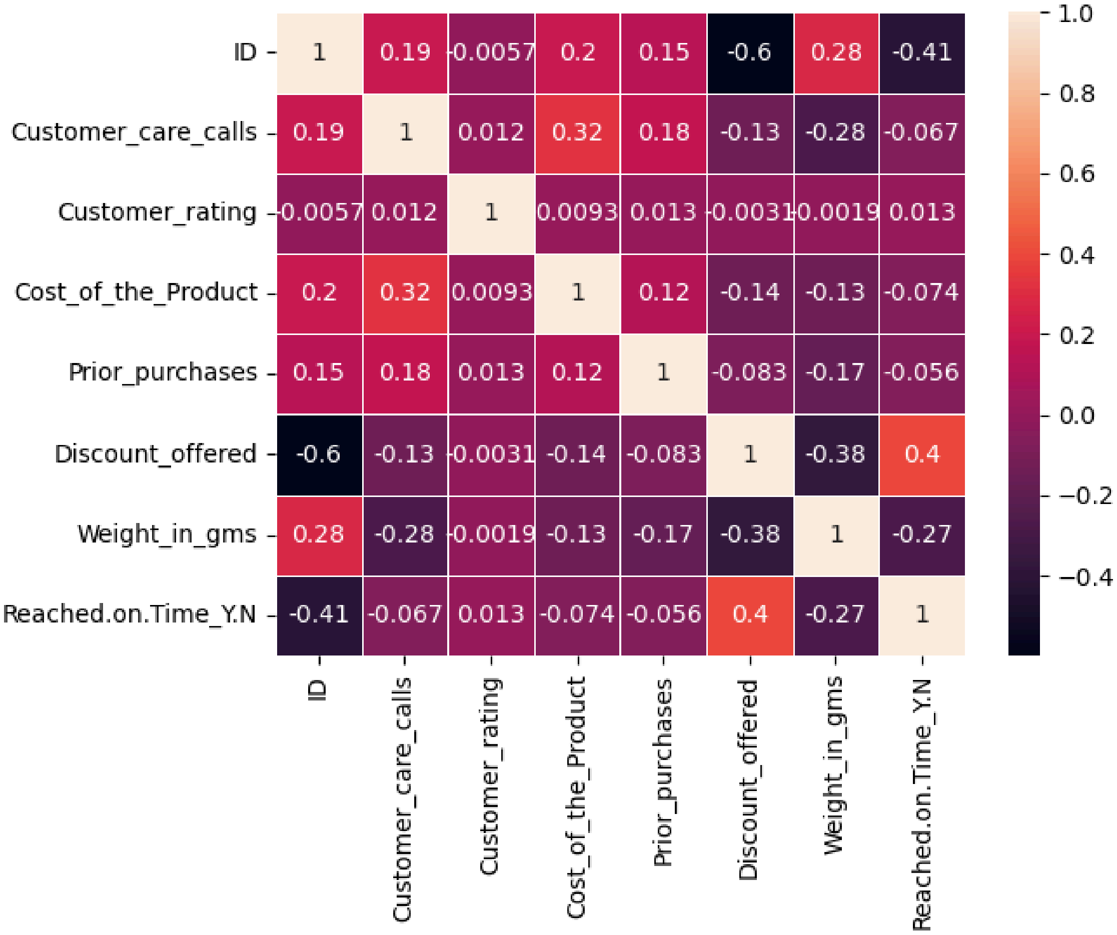
```
# correlation between individual data
cor = noobj.corr()
cor
```

Out[49]:

	ID	Customer_care_calls	Customer_rating	Cost_of_the_Product
ID	1.000000	0.188998	-0.005722	0.196791
Customer_care_calls	0.188998	1.000000	0.012209	0.323182
Customer_rating	-0.005722	0.012209	1.000000	0.009270
Cost_of_the_Product	0.196791	0.323182	0.009270	1.000000
Prior_purchases	0.145369	0.180771	0.013179	0.123676
Discount_offered	-0.598278	-0.130750	-0.003124	-0.138312
Weight_in_gms	0.278312	-0.276615	-0.001897	-0.132604
Reached.on.Time_Y.N	-0.411822	-0.067126	0.013119	-0.073587

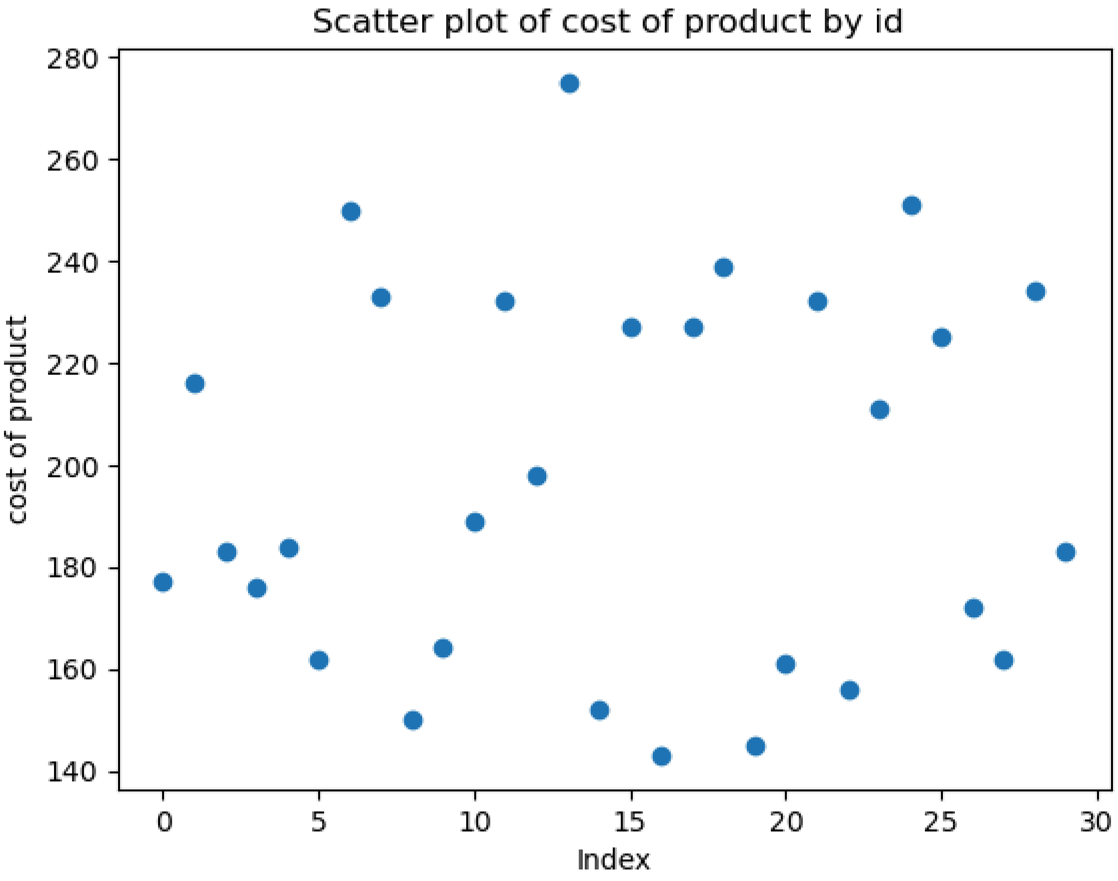
```
In [50]: # heat map for correlation data
         sns.heatmap(cor, annot=True,linewidth=.5)
```

```
Out[50]: <Axes: >
```



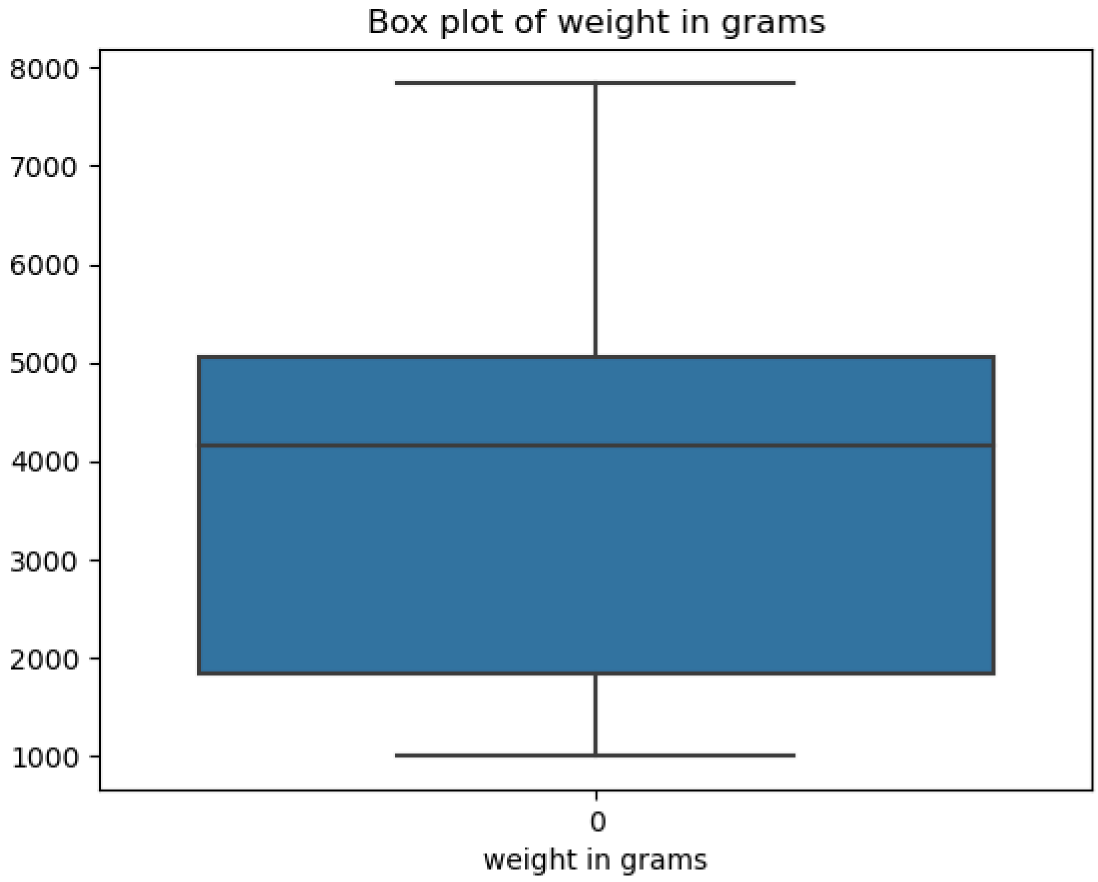
	ID	Customer_care_calls	Customer_rating	Cost_of_the_Product	Prior_purchases	Discount_offered	Weight_in_gms	Reached.on.Time_Y.N
ID	1	0.19	-0.0057	0.2	0.15	-0.6	0.28	-0.41
Customer_care_calls	0.19	1	0.012	0.32	0.18	-0.13	-0.28	-0.067
Customer_rating	-0.0057	0.012	1	0.0093	0.013	-0.0031	-0.0019	0.013
Cost_of_the_Product	0.2	0.32	0.0093	1	0.12	-0.14	-0.13	-0.074
Prior_purchases	0.15	0.18	0.013	0.12	1	-0.083	-0.17	-0.056
Discount_offered	-0.6	-0.13	-0.0031	-0.14	-0.083	1	-0.38	0.4
Weight_in_gms	0.28	-0.28	-0.0019	-0.13	-0.17	-0.38	1	-0.27
Reached.on.Time_Y.N	-0.41	-0.067	0.013	-0.074	-0.056	0.4	-0.27	1

```
In [42]:   # scatter plot for cost of each product
            data= df.head(n=30)
            plt.scatter(data.index, data['Cost_of_the_Product'])
            plt.title('Scatter plot of cost of product by id ')
            plt.xlabel('Index')
            plt.ylabel('cost of product')
            plt.show()
```



In [45]:

```
# box plot for weight
sns.boxplot(df['Weight_in_gms'])
plt.title('Box plot of weight in grams')
plt.xlabel('weight in grams')
plt.show()
```


Box plot of weight in grams
weight in grams