```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
import datetime as dt
from scipy import stats

df1 = pd.read_csv("Retail_Data_Response.csv")
df2 = pd.read_csv("Retail_Data_Transactions.csv")

df1.head()
```

	customer_id	response
0	CS1112	0
1	CS1113	0
2	CS1114	1
3	CS1115	1
4	CS1116	1

```
df2.head()
```

	customer_id	trans_date	tran_amount
0	CS5295	11-Feb-13	35
1	CS4768	15-Mar-15	39
2	CS2122	26-Feb-13	52
3	CS1217	16-Nov-11	99
4	CS1850	20-Nov-13	78

```
df1.info() # Check data types and missing values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6884 entries, 0 to 6883
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   customer_id  6884 non-null   object
 1   response     6884 non-null   int64
dtypes: int64(1), object(1)
memory usage: 107.7+ KB
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 125000 entries, 0 to 124999
Data columns (total 3 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   customer_id  125000 non-null  object
 1   trans_date   125000 non-null  object
 2   tran_amount  125000 non-null  int64
```

```
dtypes: int64(1), object(2)
memory usage: 2.9+ MB
```

```
df = df1.merge(df2, on="customer_id", how="inner")  #merge
df
```

```
       customer_id  response trans_date  tran_amount
0           CS1112         0  14-Jan-15           39
1           CS1112         0  16-Jul-14           90
2           CS1112         0  29-Apr-14           63
3           CS1112         0  04-Dec-14           59
4           CS1112         0  08-Apr-12           56
...            ...       ...         ...          ...
124964      CS9000         0  12-May-12           53
124965      CS9000         0  08-May-14           20
124966      CS9000         0  28-Feb-15           34
124967      CS9000         0  01-Jun-12           37
124968      CS9000         0  11-Dec-12           49

[124969 rows x 4 columns]
```

```
# Identify and remove duplicates
df = df.drop_duplicates().dropna()

df.describe() #Summary statistics
```

```
            response    tran_amount
count  124963.000000  124963.000000
mean        0.110761      64.995735
std         0.313837      22.860005
min         0.000000      10.000000
25%         0.000000      47.000000
50%         0.000000      65.000000
75%         0.000000      83.000000
max         1.000000     105.000000

df.dtypes

customer_id    object
response        int64
trans_date     object
tran_amount     int64
dtype: object

df.shape

(124963, 4)

df.tail()

       customer_id  response trans_date  tran_amount
124964      CS9000         0  12-May-12           53
```

```
124965      CS9000          0  08-May-14           20
124966      CS9000          0  28-Feb-15           34
124967      CS9000          0  01-Jun-12           37
124968      CS9000          0  11-Dec-12           49
```

```
df.isnull().sum()
```

```
customer_id    0
response       0
trans_date     0
tran_amount    0
dtype: int64
```

```
num = df.select_dtypes(include=['number']).columns
num
```

```
Index(['response', 'tran_amount'], dtype='object')
```

```
df["customer_id"].nunique()
```

```
6884
```

```
df["customer_id"].value_counts()
```

```
customer_id
CS4424    39
CS4320    38
CS3799    36
CS2620    35
CS3013    35
          ..
CS7224     4
CS7333     4
CS8559     4
CS8504     4
CS7716     4
Name: count, Length: 6884, dtype: int64
```

```
df["customer_id"].value_counts().count()
```

```
6884
```

```
#change dtypes
df["trans_date"]=pd.to_datetime(df["trans_date"])
df["response"]=df["response"].astype('int64')
```

```
C:\Users\ADITHYA\AppData\Local\Temp\ipykernel_12844\3408161854.py:2:
UserWarning: Could not infer format, so each element will be parsed
individually, falling back to `dateutil`. To ensure parsing is
consistent and as-expected, please specify a format.
  df["trans_date"]=pd.to_datetime(df["trans_date"])
```

```
set(df['response'])
```

```
{0, 1}
```

```
# Create a StandardScaler object
scaler = StandardScaler()

# Fit the scaler to the numerical data
scaler.fit(df[num])

# Standardize the numerical data
df[num] = scaler.transform(df[num])
df[num]
```

```
        response  tran_amount
0      -0.352926    -1.137176
1      -0.352926     1.093804
2      -0.352926    -0.087303
3      -0.352926    -0.262282
4      -0.352926    -0.393516
...          ...          ...
124964 -0.352926    -0.524750
124965 -0.352926    -1.968325
124966 -0.352926    -1.355899
124967 -0.352926    -1.224665
124968 -0.352926    -0.699729

[124963 rows x 2 columns]
```

```
sns.distplot(df["tran_amount"])
plt.title("Distribution of Transaction amount")
plt.show()
```

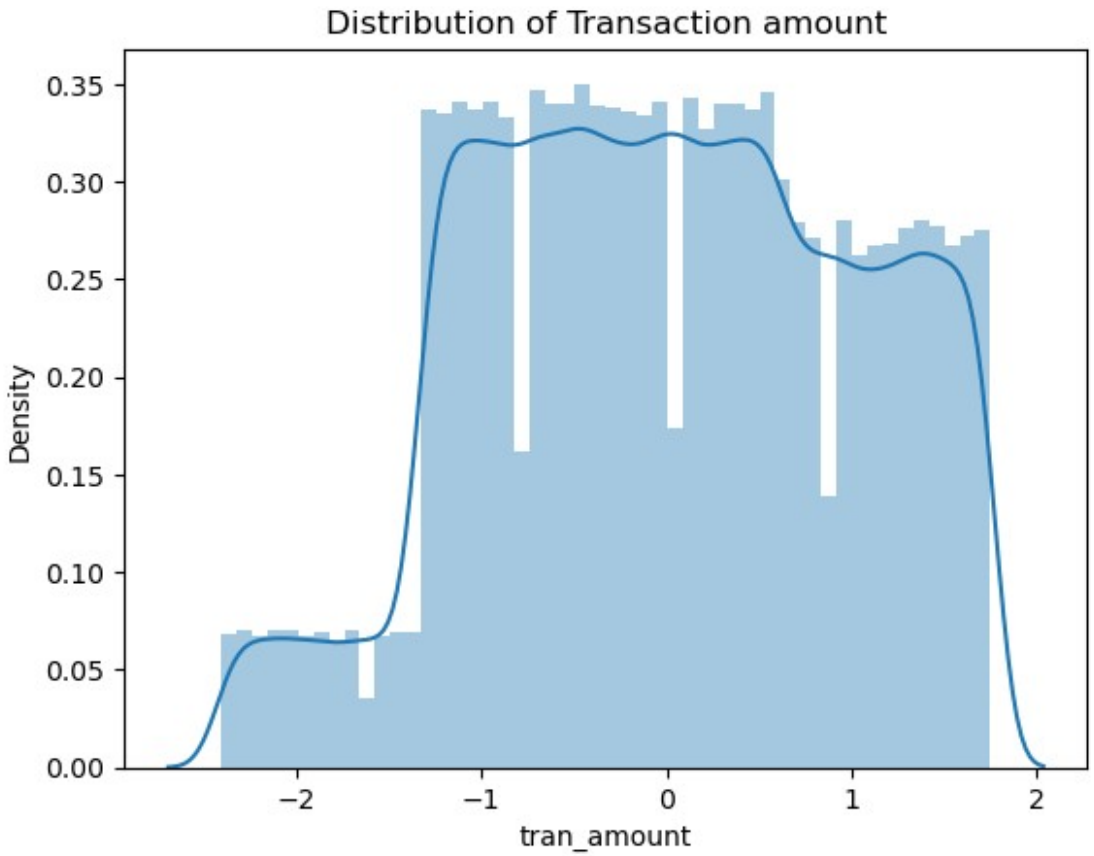
```
C:\Users\ADITHYA\AppData\Local\Temp\ipykernel_12844\1303522202.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["tran_amount"])
```




```
sns.distplot(df["response"])
plt.title("Distribution of Response")
plt.show()
```

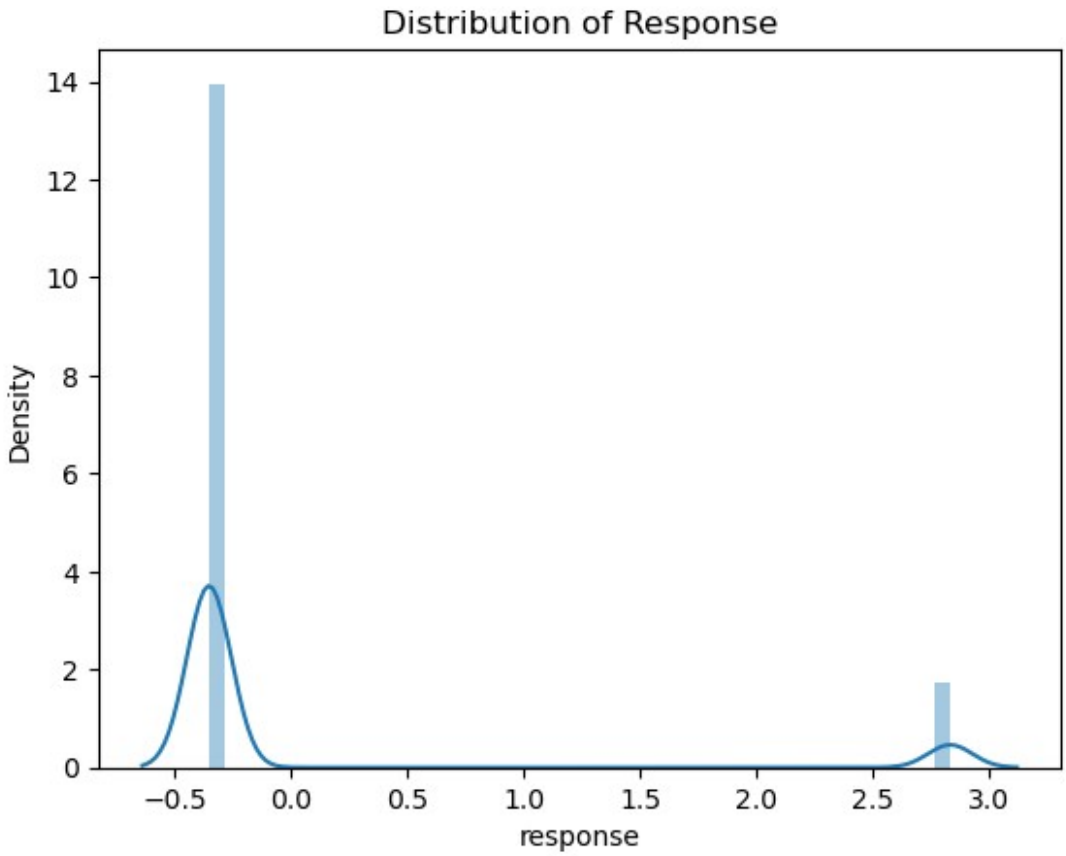
```
C:\Users\ADITHYA\AppData\Local\Temp\ipykernel_12844\1570856337.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level
function with
similar flexibility) or `histplot` (an axes-level function for
histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["response"])
```




```
# Calculate Pearson correlation coefficient between two columns
correlation_coefficient = df["tran_amount"].corr(df["response"])
print(f"Pearson correlation coefficient: {correlation_coefficient}")

# Calculate Spearman rank correlation coefficient
spearman_coefficient = df["tran_amount"].corr(df["response"],
method="spearman")
print(f"Spearman rank correlation coefficient:
{spearman_coefficient}")
```

```
Pearson correlation coefficient: 0.06234578967547238
Spearman rank correlation coefficient: 0.057656733187235944
```

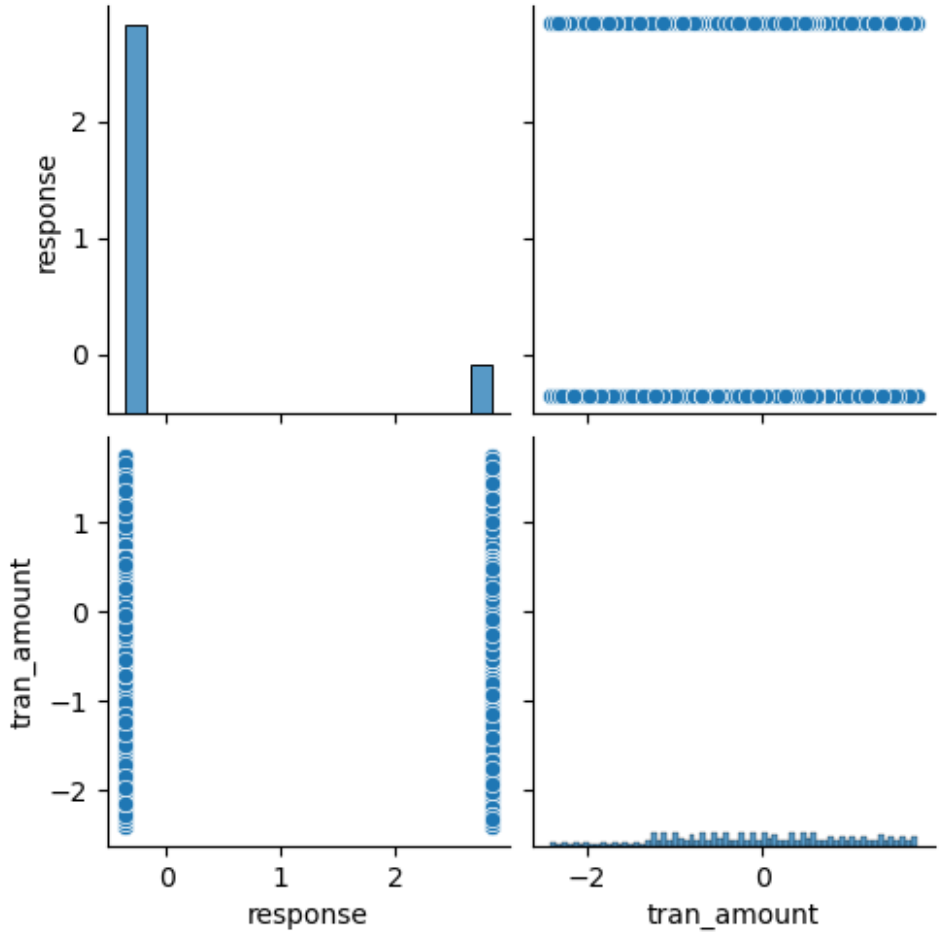
```
sns.boxplot(x=data["tran_amount"])
plt.show()
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
Cell In[27], line 1
----> 1 sns.boxplot(x=data["tran_amount"])
      2 plt.show()
```

```
NameError: name 'data' is not defined
```

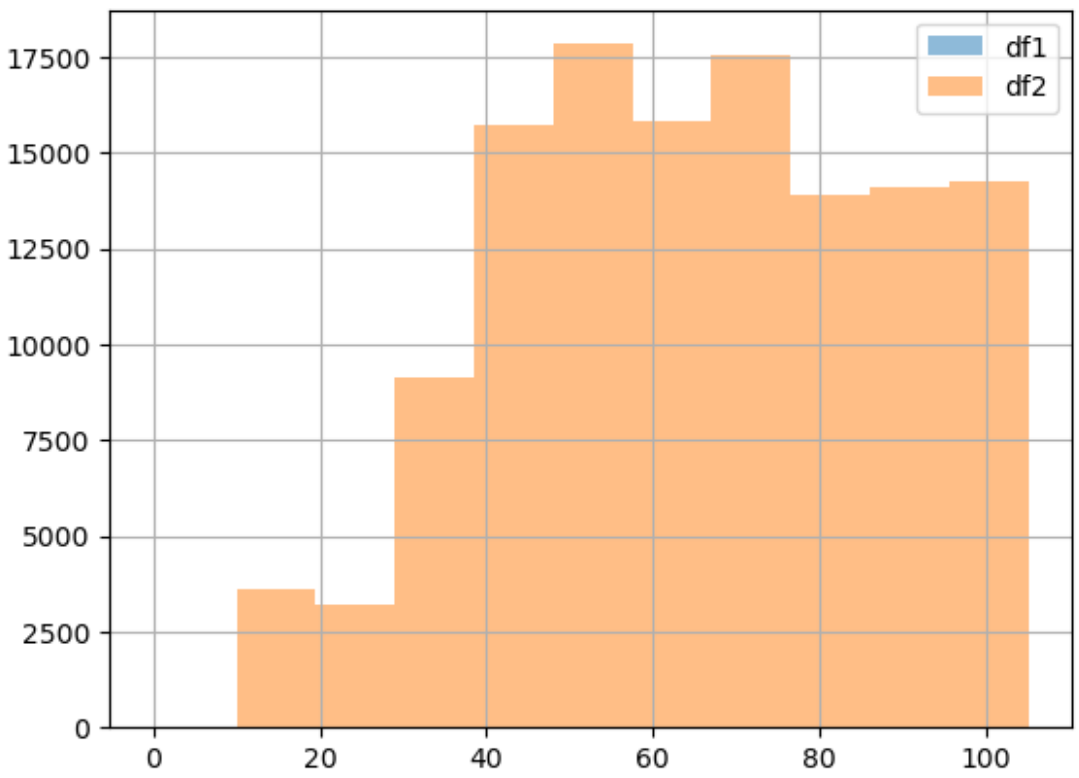
```
sns.pairplot(df)
plt.show()
```

```
E:\App Dev\Anaconda\Lib\site-packages\seaborn\axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




```
 import matplotlib.pyplot as plt

# Example: Plot distribution of "price" in both datasets
df1["response"].hist(alpha=0.5, label="df1")
df2["tran_amount"].hist(alpha=0.5, label="df2")
plt.legend()
plt.show()
```




```
df['month'] = df['trans_date'].dt.month

monthly_Sales = df.groupby('month')['tran_amount'].sum()
monthly_Sales =
monthly_Sales.sort_values(ascending=False).reset_index()
monthly_Sales
```

	month	tran_amount
0	11	200.547834
1	5	152.376863
2	3	117.742630
3	7	-1.659920
4	1	-9.775536
5	9	-12.267323
6	2	-35.416598
7	8	-37.281436
8	10	-54.783233
9	6	-90.994345
10	4	-107.830377
11	12	-120.658559

```
# Customers having highest num of orders
customer_counts= df['customer_id'].value_counts().reset_index()
customer_counts.columns=['customer_id','count']

#sort
```

```
top_cus= customer_counts.sort_values(by='count',
ascending=False).head(5)
top_cus
```

	customer_id	count
0	CS4424	39
1	CS4320	38
2	CS3799	36
3	CS2620	35
4	CS3013	35

```
sns.barplot(x='customer_id',y='count',data=top_cus)
plt.show()
```