ALLIANCE UNIVERSITY

Private University established in Karnataka State by Act No.34 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi

PROJRCT REPORT

BACHELOR OF COMPUTER APPLICATION

SEMESTER – II

STATISTICS AND DATASCIENCE

E-commerce User Behavior Analysis using Statistical and Predictive Techniques

BY

PRAGATHI.BR

2411021240037

DEPARTMENT OF COMPUTER APPLICATION

ALLIANCE UNIVERSITY

CHANDPURA ANEKAL MAIN, ANEKAL

BANGALORE-562106

APRIL – 2025

📊 E-commerce User Behavior Analysis using Statistical and Predictive Techniques

NAME: PRAGATHI.BR

Registration no. : 2411021240037

GITHUB LINK: https://github.com/Pragathi2006/-E-commerce-User-Behavior-Analysis-using-Statistical-and-Predictive-Techniques.

📚 Table of Contents

Welcome to project notebook! Below is the structured flow of analysis covered in this project

📘 Project Title and Info

📊 E-commerce User Behavior Analysis using Statistical and Predictive Techniques

In this project we analyze ecommerce user behavior using statisticaltechniques which provide valuable insights for business decision making. which helps increase sales,reduce waste and enhance customer satisfaction by understanding and predicting user behavior more effectively, without building complex machine learning models

📜 Project Overview

This project focuses on analyzing user behavior data from an ecommerce platform by applying statistical methods.we aim to understand user interactions,identify purchasing trends, and build models predict user actions and optimize engagement strategies. we apply Foundation statistics to:

- Understand customer behavior patterns
- Probability of user purchasing after viewing
- Make E-commerce analysis using hypothesis testing, Regreesion,categorical analysis***

🎯 Project Goal

The main obejective of this project are:

- understand user engagement and behavior patterns
- identify factors influencing product views, card additions, purchases
- Build regression and classification models for behavior production
- Support data-driven marketing and inventory strategies

🚧 Challenges Faced

- Handling missing end inconsistent data
- High cardinality in user IDs or session data
- Imbalanced data (e,g., fewer purchases than views)
- Categorical data needing encoding
- Lack of time-series structure for temporal trends

📦 Import Libraries & Load Dataset

We begin by importing essential libraries and loading the dataset for further analysis.

Import Libraries

```
# 📌 1. Import Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.metrics import  mean_squared_error, r2_score
from scipy.stats import norm, ttest_ind, chi2_contingency, shapiro
```

```
# 📌 2. Load the Dataset
data=pd.read_csv(r"C:\Users\Pragathi BR\Downloads\ecommerce_user_behavior_5000.csv")
data
```

	User_ID	Sessions	Clicks	Pages_Visited	Session_Duration_Minutes	Bounce_Rate	Conversion_Rate	Products_Viewed	Purchase	Device	Region	User_Type
0	1	4	79	45	21.986987	64.935604	19.761350	12	0	Desktop	South America	New
1	2	6	16	13	9.909687	73.829381	71.429564	9	0	Tablet	Europe	New
2	3	6	6	9	38.973622	55.880656	31.575336	19	1	Desktop	South America	New
3	4	8	11	12	3.974864	66.557102	1.446217	13	0	Desktop	Europe	Returning
4	5	5	32	38	51.519283	98.277109	56.625812	6	0	Mobile	North America	New
...	...	...	...	...	...	...	...	...	...	...	...	...
4995	4996	5	79	22	7.863328	68.549980	11.627298	9	1	Mobile	Europe	New
4996	4997	1	21	42	20.712306	52.710679	91.446606	7	0	Tablet	Asia	New
4997	4998	3	95	30	52.526198	49.415434	87.365218	9	1	Mobile	Asia	New
4998	4999	1	32	35	29.002925	80.669295	27.788015	11	1	Tablet	Africa	New
4999	5000	4	59	41	6.835687	78.954204	49.770447	3	1	Tablet	Africa	Returning

5000 rows × 12 columns

🖌️ Data Preprocessing

In this step, we:

- Check and end missing values
- Convert categorical columns(like event type) using label or one-hot encoding
- Drop irrelevant or duplicate rows
- Convert timestamps if available

```
# first columns of the dataset
data.head()
```

	User_ID	Sessions	Clicks	Pages_Visited	Session_Duration_Minutes	Bounce_Rate	Conversion_Rate	Products_Viewed	Purchase	Device	Region	User_Type
0	1	4	79	45	21.986987	64.935604	19.761350	12	0	Desktop	South America	New
1	2	6	16	13	9.909687	73.829381	71.429564	9	0	Tablet	Europe	New
2	3	6	6	9	38.973622	55.880656	31.575336	19	1	Desktop	South America	New
3	4	8	11	12	3.974864	66.557102	1.446217	13	0	Desktop	Europe	Returning
4	5	5	32	38	51.519283	98.277109	56.625812	6	0	Mobile	North America	New

```
# columns of the dataset
data.columns
```

```
Index(['User_ID', 'Sessions', 'Clicks', 'Pages_Visited',
       'Session_Duration_Minutes', 'Bounce_Rate', 'Conversion_Rate',
       'Products_Viewed', 'Purchase', 'Device', 'Region', 'User_Type'],
      dtype='object')
```

```python
# 📌 3. Basic Info & Data Overview
print("Shape of the dataset:", data.shape)
```

```
Shape of the dataset: (5000, 12)
```

```python
# Information of the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   User_ID                    5000 non-null   int64  
 1   Sessions                   5000 non-null   int64  
 2   Clicks                     5000 non-null   int64  
 3   Pages_Visited              5000 non-null   int64  
 4   Session_Duration_Minutes   5000 non-null   float64
 5   Bounce_Rate                5000 non-null   float64
 6   Conversion_Rate            5000 non-null   float64
 7   Products_Viewed            5000 non-null   int64  
 8   Purchase                   5000 non-null   int64  
 9   Device                     5000 non-null   object 
 10  Region                     5000 non-null   object 
 11  User_Type                  5000 non-null   object 
dtypes: float64(3), int64(6), object(3)
memory usage: 468.9+ KB
```

```python
# Statistical Summary of the data
data.describe(include="all")
```

	User_ID	Sessions	Clicks	Pages_Visited	Session_Duration_Minutes	Bounce_Rate	Conversion_Rate	Products_Viewed	Purchase	Device	Region
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000	5000	5000
unique	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3	5
top	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Mobile	Africa
freq	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1717	1029
mean	2500.500000	4.996600	49.414800	25.308600	30.739115	50.116711	50.002577	9.993000	0.50040	NaN	NaN
std	1443.520003	2.573463	28.671658	13.967161	17.178472	28.562569	28.893355	5.469418	0.50005	NaN	NaN
min	1.000000	1.000000	1.000000	1.000000	0.525038	0.030484	0.003059	1.000000	0.00000	NaN	NaN
25%	1250.750000	3.000000	24.000000	13.000000	15.905214	25.861284	25.377823	5.000000	0.00000	NaN	NaN
50%	2500.500000	5.000000	49.000000	25.000000	30.790858	50.257232	49.790168	10.000000	1.00000	NaN	NaN
75%	3750.250000	7.000000	74.000000	37.000000	46.071697	74.353398	74.800206	15.000000	1.00000	NaN	NaN
max	5000.000000	9.000000	99.000000	49.000000	59.999943	99.999420	99.986451	19.000000	1.00000	NaN	NaN

```
#check for dtypes
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   User_ID                    5000 non-null   int64  
 1   Sessions                   5000 non-null   int64  
 2   Clicks                     5000 non-null   int64  
 3   Pages_Visited              5000 non-null   int64  
 4   Session_Duration_Minutes   5000 non-null   float64
 5   Bounce_Rate                5000 non-null   float64
 6   Conversion_Rate            5000 non-null   float64
 7   Products_Viewed            5000 non-null   int64  
 8   Purchase                   5000 non-null   int64  
 9   Device                     5000 non-null   object 
 10  Region                     5000 non-null   object 
 11  User_Type                  5000 non-null   object 
dtypes: float64(3), int64(6), object(3)
memory usage: 468.9+ KB
```

```
#check the  null value
data.isna().sum()
```

```
User_ID                    0
Sessions                   0
Clicks                     0
Pages_Visited              0
Session_Duration_Minutes   0
Bounce_Rate                0
Conversion_Rate            0
Products_Viewed            0
Purchase                   0
Device                     0
Region                     0
User_Type                  0
dtype: int64
```

```
# 📌 4. Data Cleaning
# Checking for missing values
print(data.isnull().sum())
```

```
User_ID                      0
Sessions                     0
Clicks                       0
Pages_Visited                0
Session_Duration_Minutes     0
Bounce_Rate                  0
Conversion_Rate              0
Products_Viewed              0
Purchase                     0
Device                       0
Region                       0
User_Type                    0
dtype: int64
```

```
# Filling or removing missing values (if any)
data.dropna(inplace=True)

# Removing duplicates
data.drop_duplicates(inplace=True)

# Reset index
data.reset_index(drop=True, inplace=True)

# 📌 5. Data Preparation
# Convert date columns if any
if 'Date' in data.columns:
    data['Date'] = pd.to_datetime(data['Date'])

# Encode categorical columns
data_encoded = pd.get_dummies(data, drop_first=True)
```

📊 Exploratory Data Analysis (EDA)

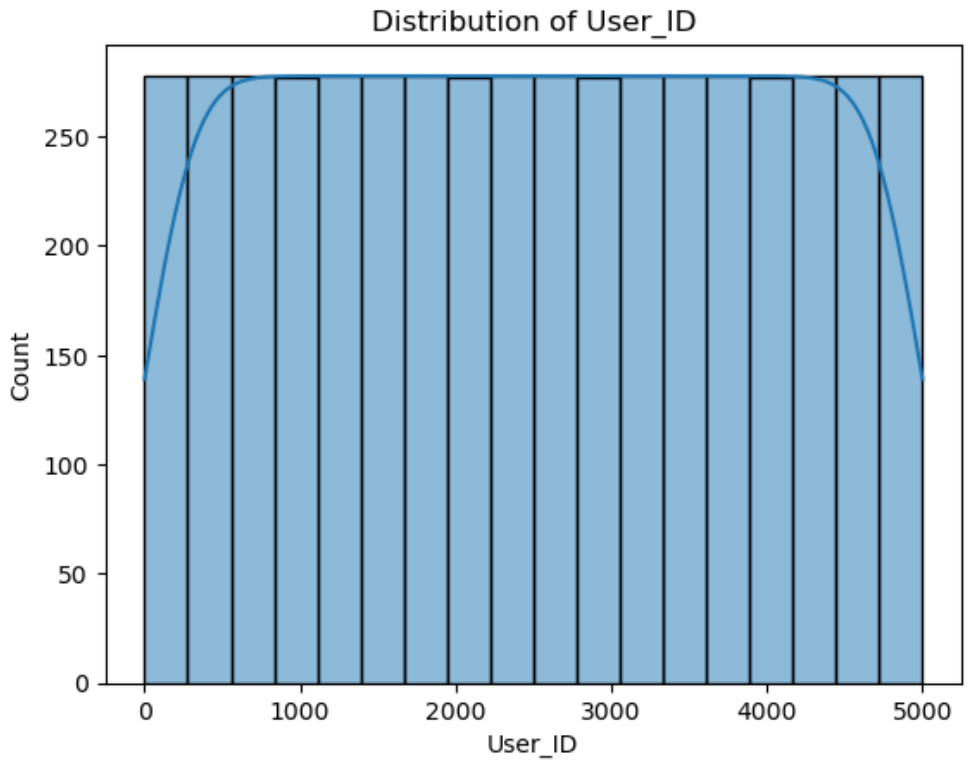
We explore trends and patterns using:

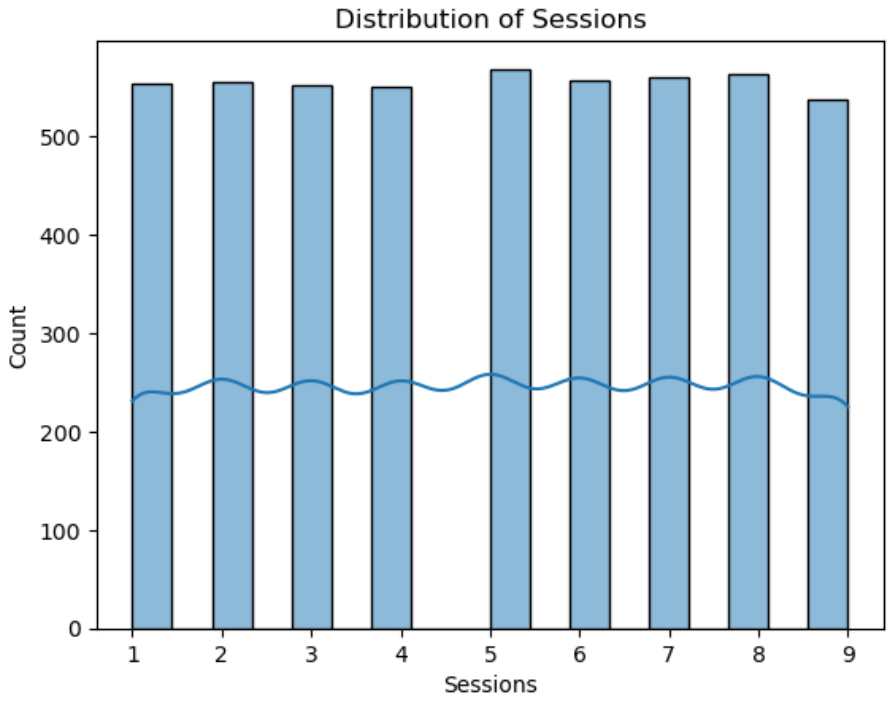
- Distribution of event types(e.g,.view,cart,purchase)
- Most viewed/purchase categories or products
- Correlation matrix of numerical features
- Time-based behavior trends(if timestamps available)

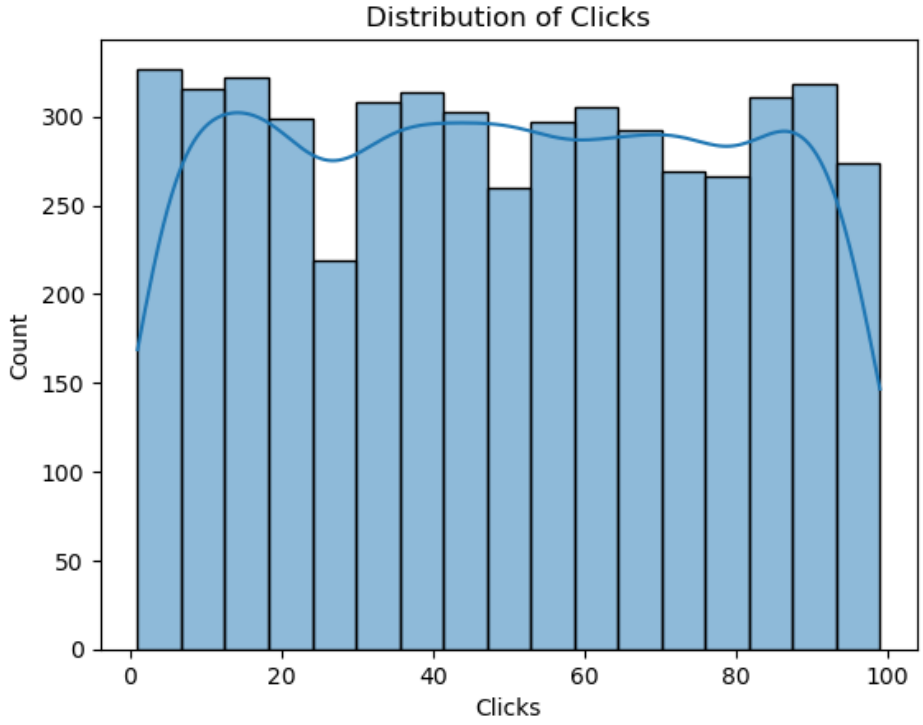
```
# 📌 6. Descriptive Statistics
data.describe()
```

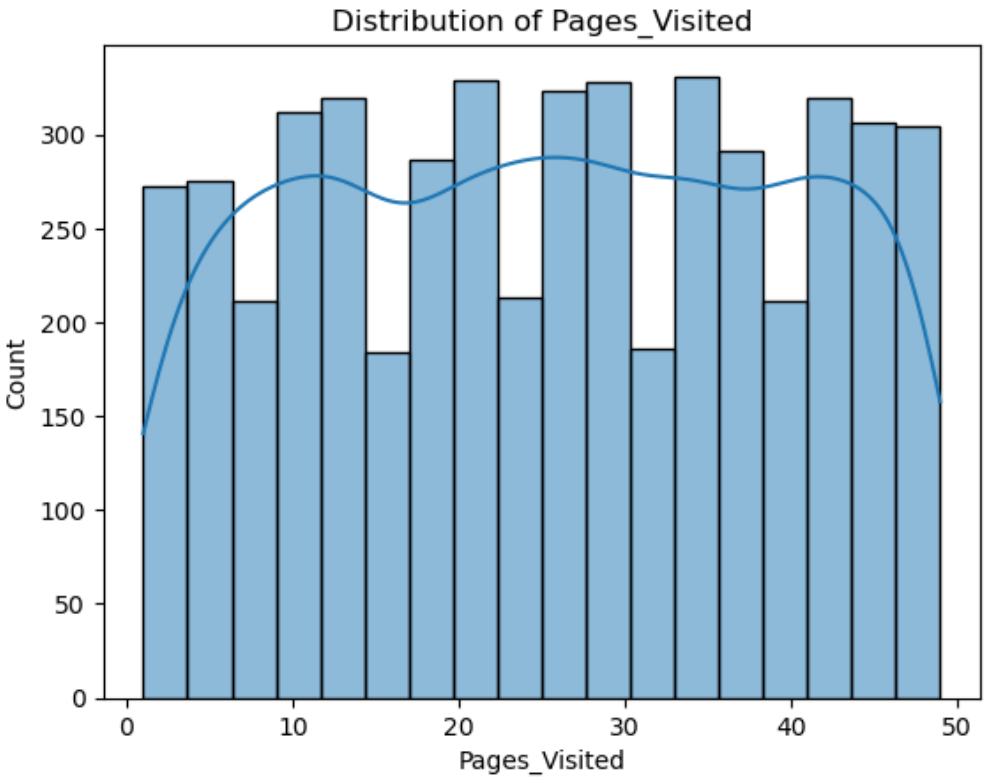
	User_ID	Sessions	Clicks	Pages_Visited	Session_Duration_Minutes	Bounce_Rate	Conversion_Rate	Products_Viewed	Purchase
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5000.00000
mean	2500.500000	4.996600	49.414800	25.308600	30.739115	50.116711	50.002577	9.993000	0.50040
std	1443.520003	2.573463	28.671658	13.967161	17.178472	28.562569	28.893355	5.469418	0.50005
min	1.000000	1.000000	1.000000	1.000000	0.525038	0.030484	0.003059	1.000000	0.00000
25%	1250.750000	3.000000	24.000000	13.000000	15.905214	25.861284	25.377823	5.000000	0.00000
50%	2500.500000	5.000000	49.000000	25.000000	30.790858	50.257232	49.790168	10.000000	1.00000
75%	3750.250000	7.000000	74.000000	37.000000	46.071697	74.353398	74.800206	15.000000	1.00000
max	5000.000000	9.000000	99.000000	49.000000	59.999943	99.999420	99.986451	19.000000	1.00000

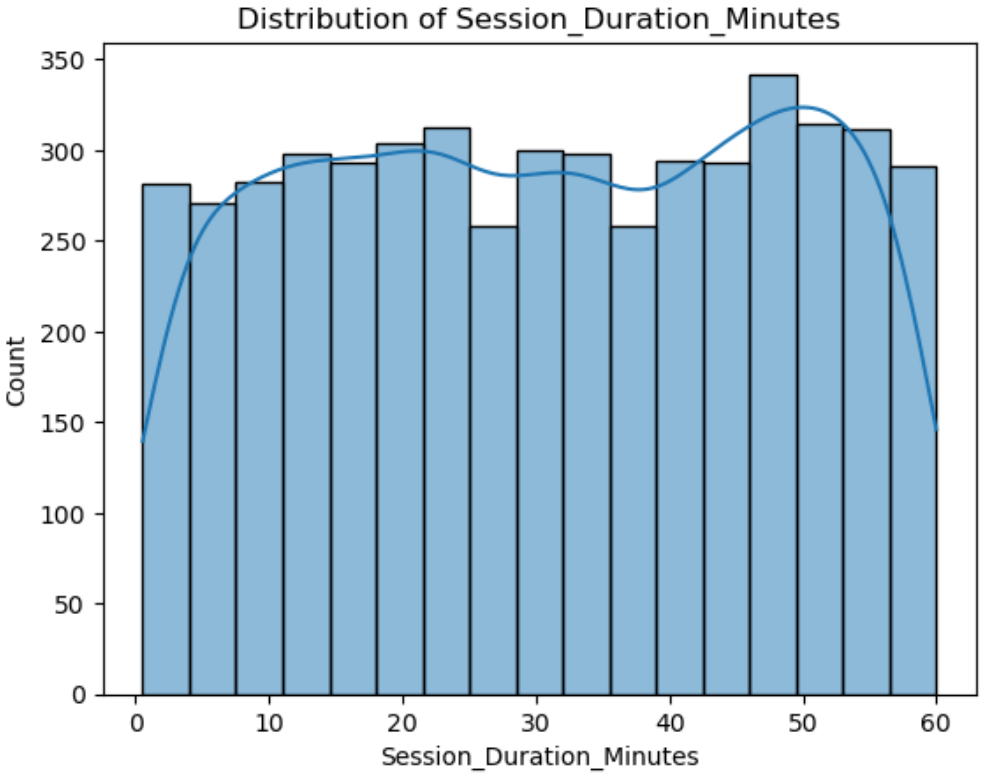
```
# 📌 7. Exploratory Data Analysis (EDA)

# Plotting distributions of numerical columns
numerical_cols = data.select_dtypes(include=np.number).columns
for col in numerical_cols:
    sns.histplot(data[col], kde=True)
    plt.title(f"Distribution of {col}")
    plt.show()
```

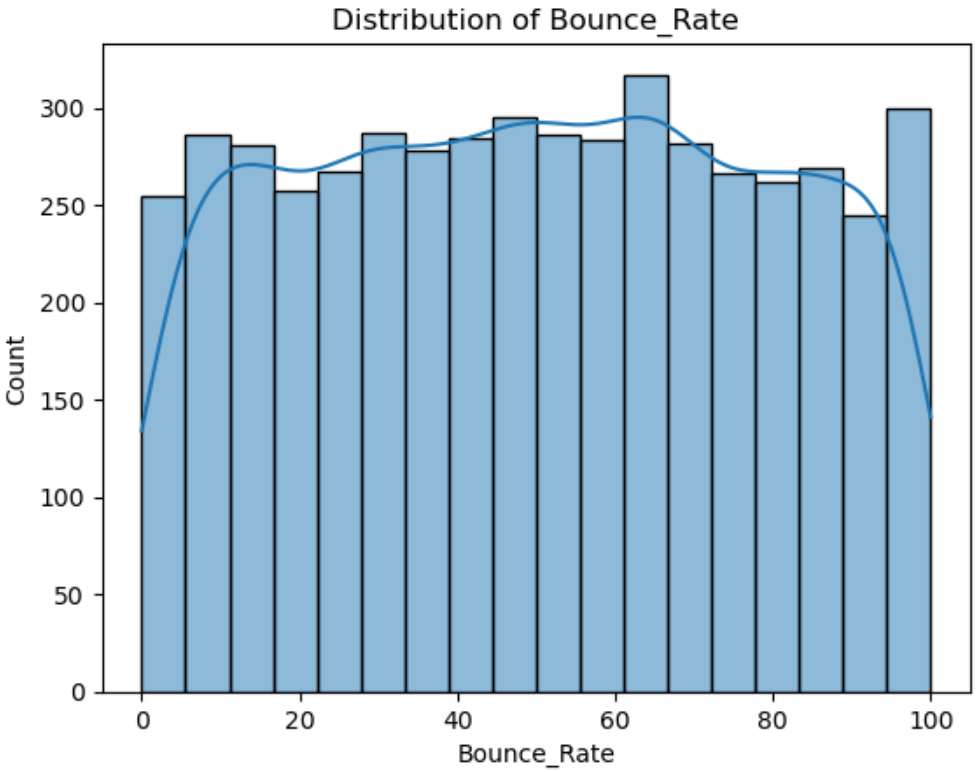


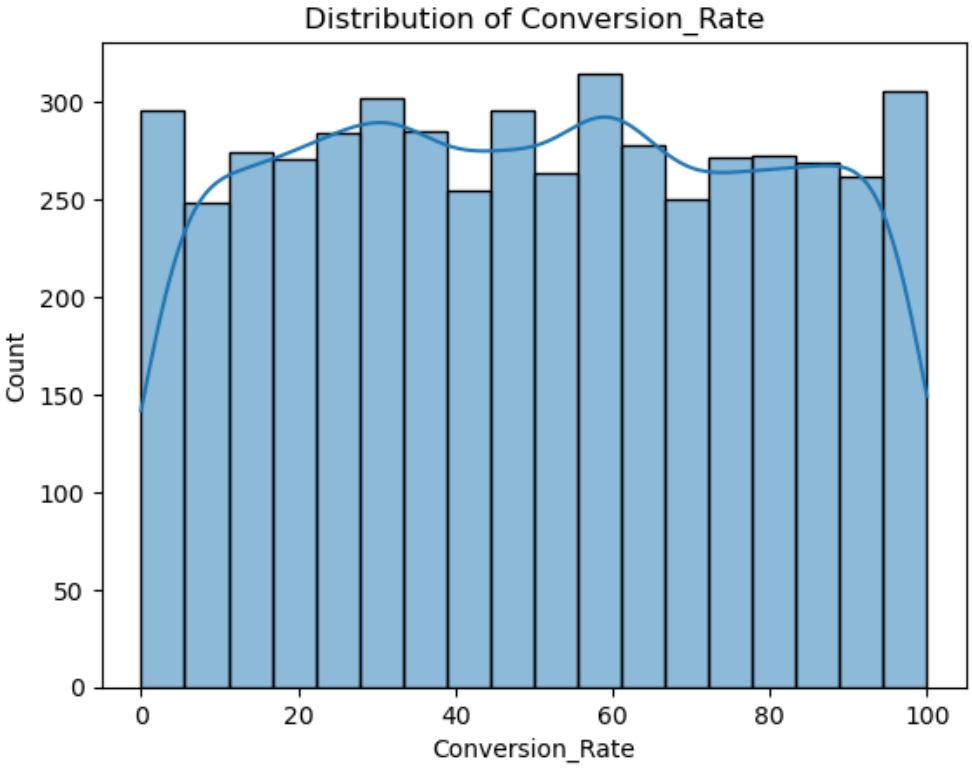

Distribution of Sessions
Count
Sessions
0
100
200
300
400
500
1
2
3
4
5
6
7
8
9

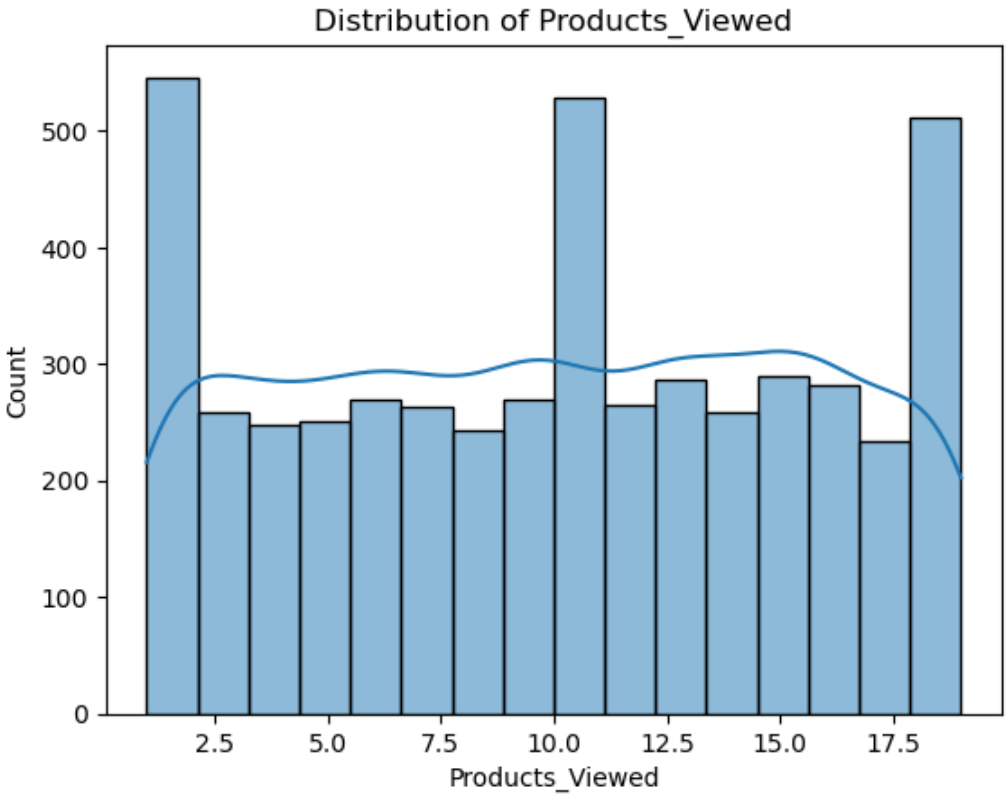

Distribution of Clicks
Count
Clicks
0
50
100
150
200
250
300
0
20
40
60
80
100

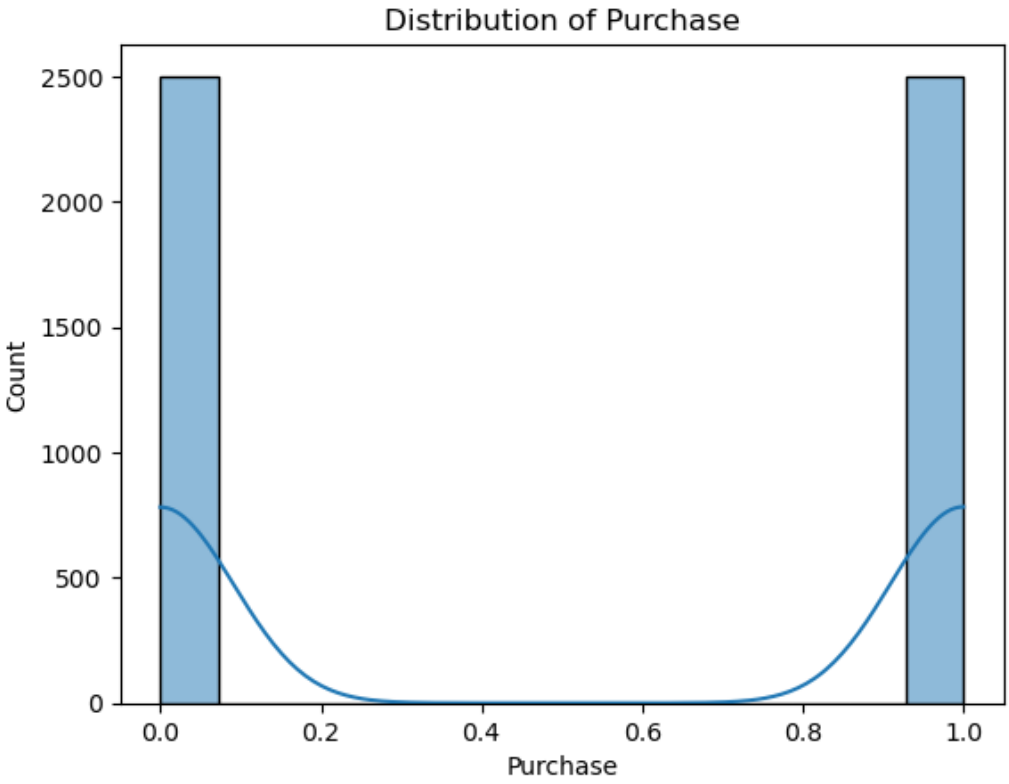

Distribution of Pages_Visited
Count
Pages_Visited
0
50
100
150
200
250
300
0
10
20
30
40
50


Distribution of Session_Duration_Minutes
Count
Session_Duration_Minutes
0
50
100
150
200
250
300
350
0
10
20
30
40
50
60


Distribution of Bounce_Rate
Count
Bounce_Rate
0
50
100
150
200
250
300
0
20
40
60
80
100

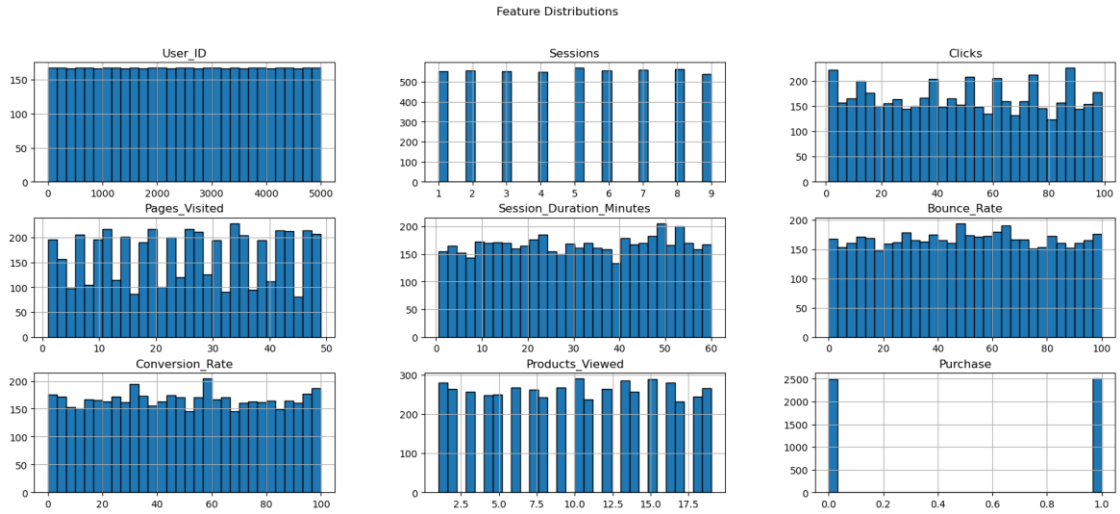

Distribution of Conversion_Rate
Count
Conversion_Rate
0
50
100
150
200
250
300
0
20
40
60
80
100


Distribution of Products_Viewed
Count
Products_Viewed
0
100
200
300
400
500
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Distribution of Purchase
Count
Purchase
0
500
1000
1500
2000
2500
0.0
0.2
0.4
0.6
0.8
1.0

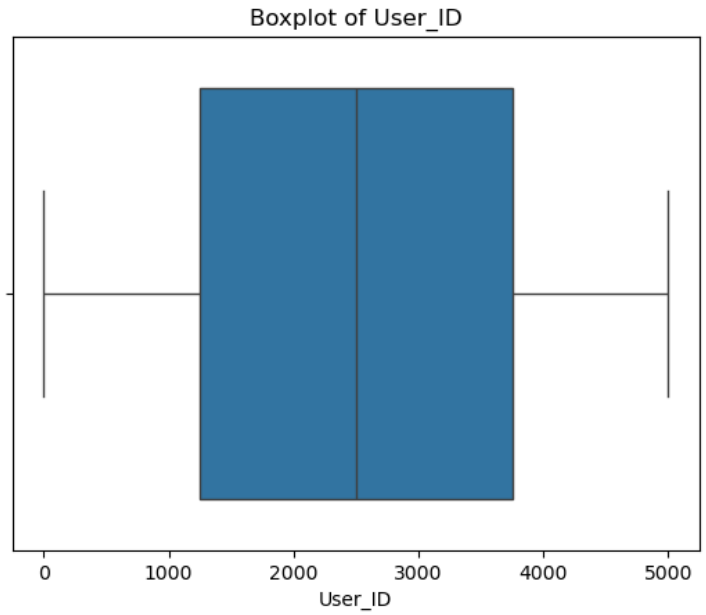
```
#EDA:Visualizing Data Distributions
plt.figure(figsize=(20,6))
data.hist(bins=30,figsize=(20,8),edgecolor="black")
plt.suptitle("Feature Distributions")
plt.show()
```

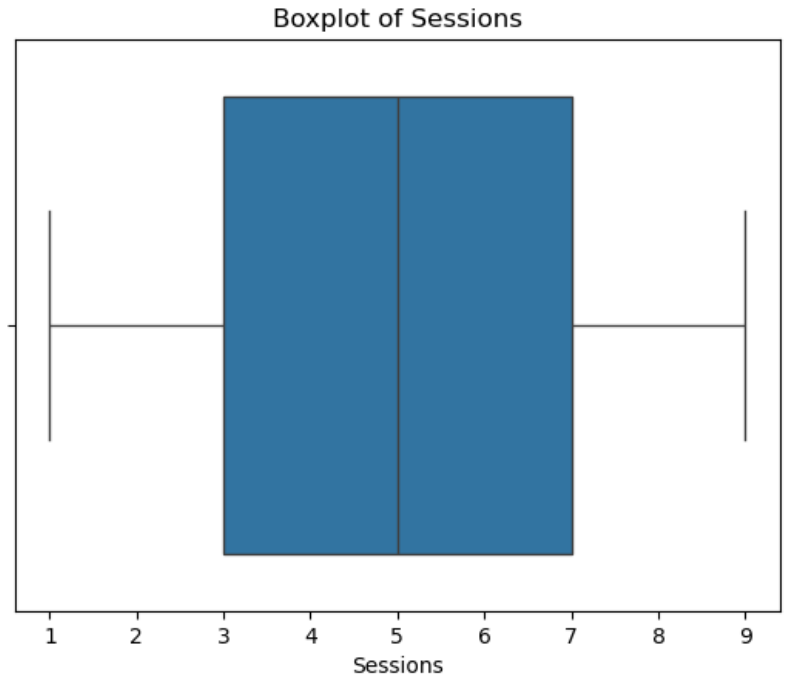
```
<Figure size 2000x600 with 0 Axes>
```

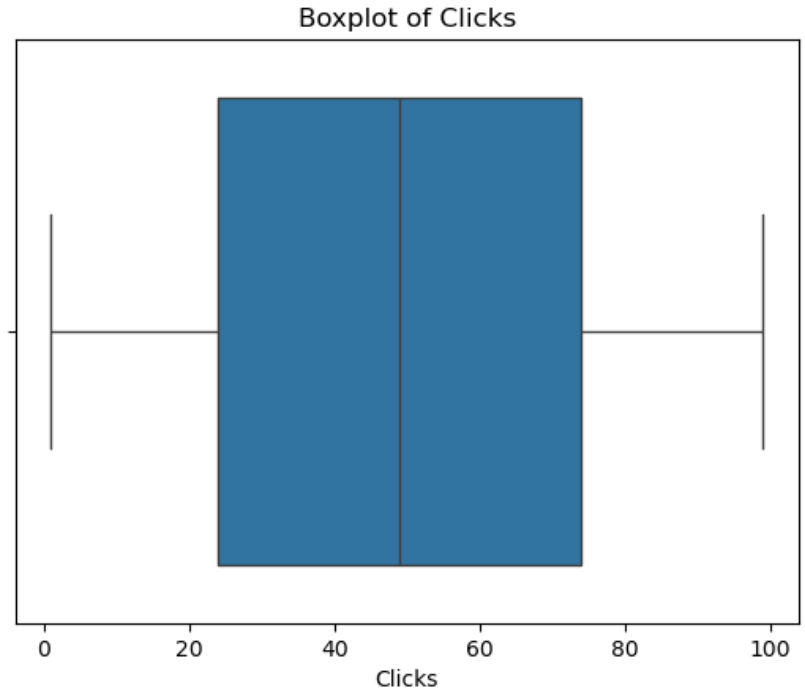


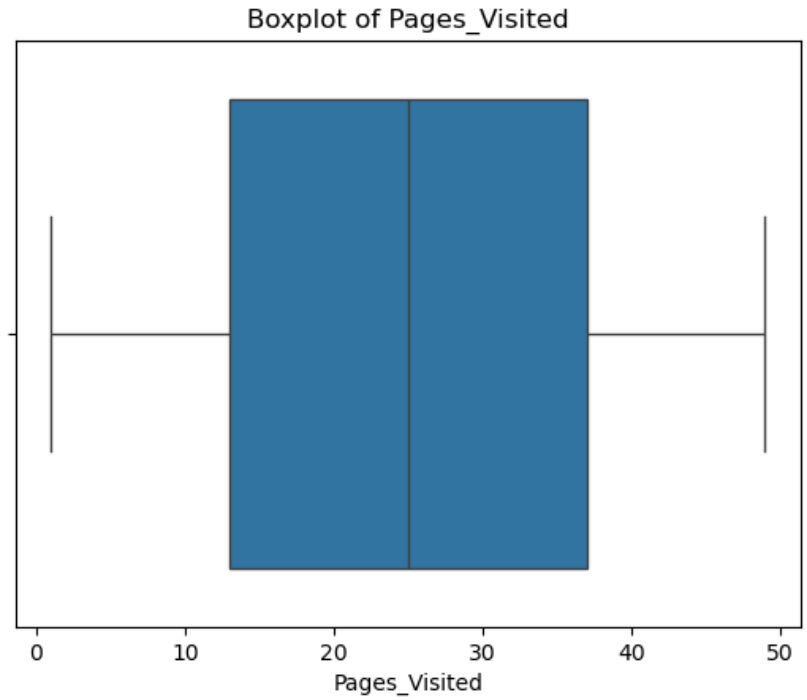
👆 This shows us each variable's that spreads a data across different value range

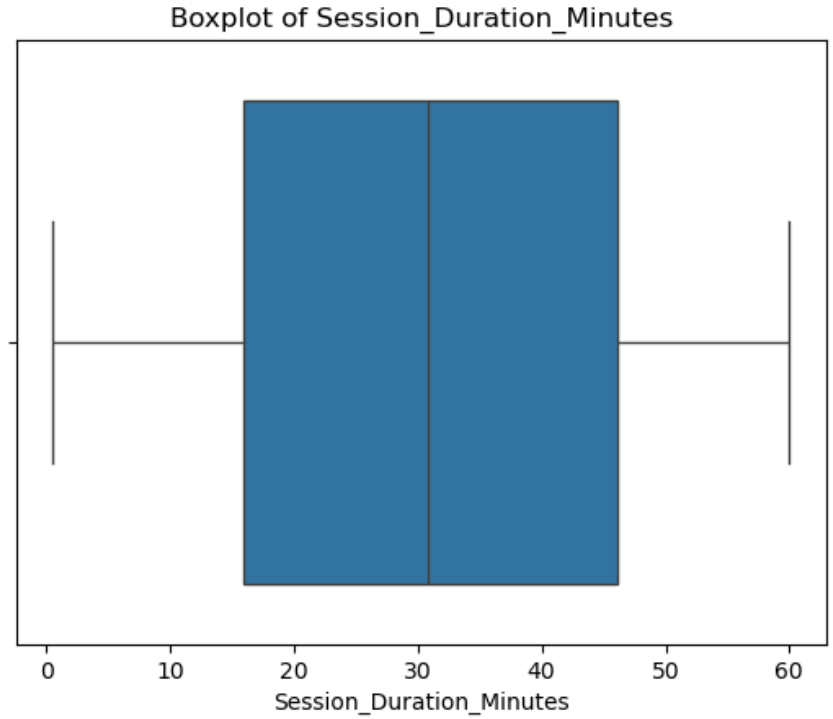
```
# Box plots for outliers
for col in numerical_cols:
    sns.boxplot(x=data[col])
    plt.title(f"Boxplot of {col}")
    plt.show()
```

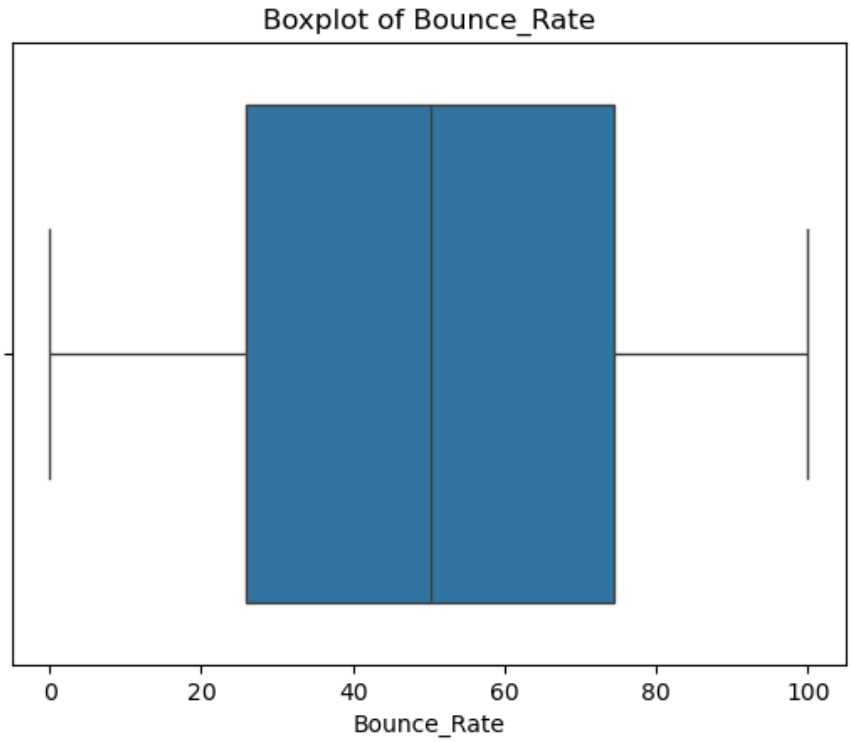


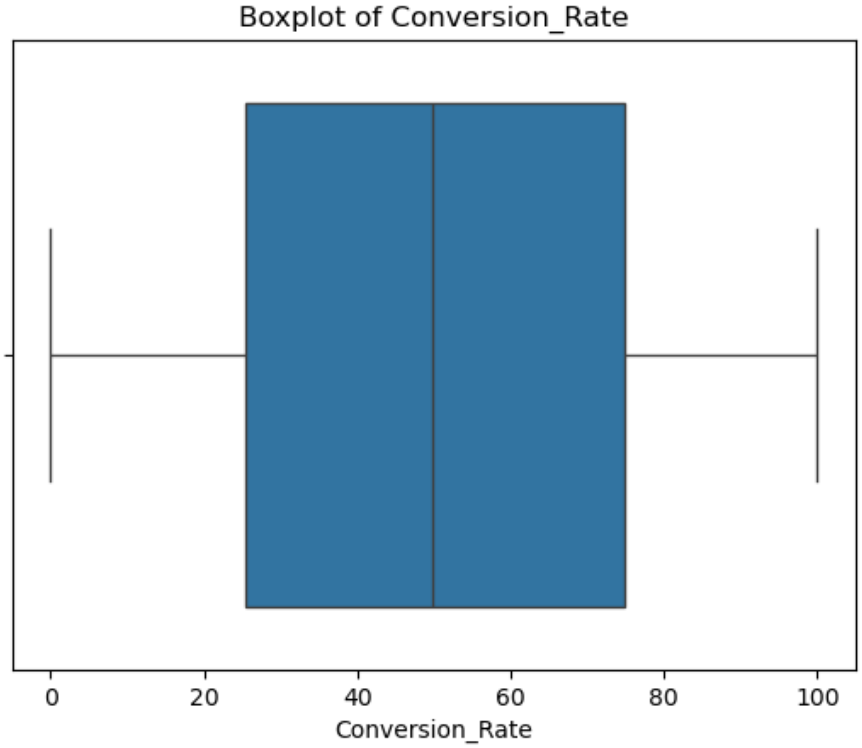

Boxplot of Sessions
1
2
3
4
5
6
7
8
9
Sessions

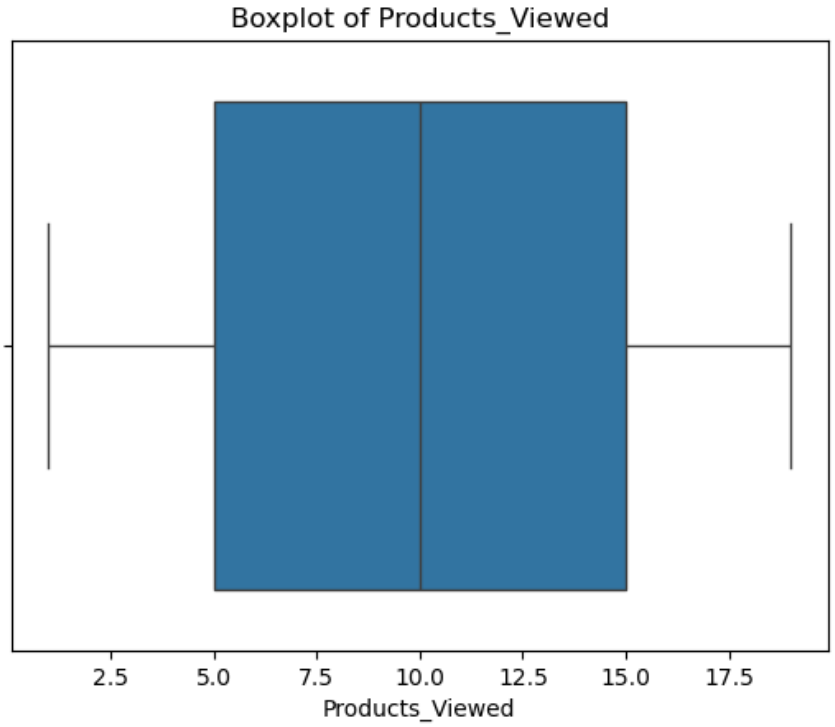

Boxplot of Clicks
0
20
40
60
80
100
Clicks

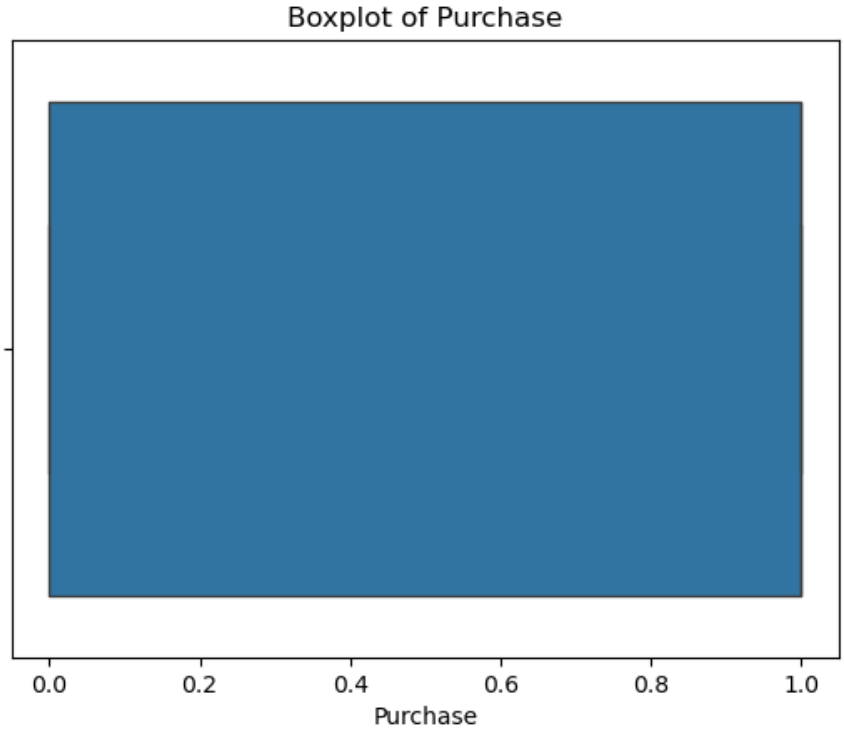

Boxplot of Pages_Visited
0
10
20
30
40
50
Pages_Visited


Boxplot of Session_Duration_Minutes
0
10
20
30
40
50
60
Session_Duration_Minutes


Boxplot of Bounce_Rate
0
20
40
60
80
100
Bounce_Rate

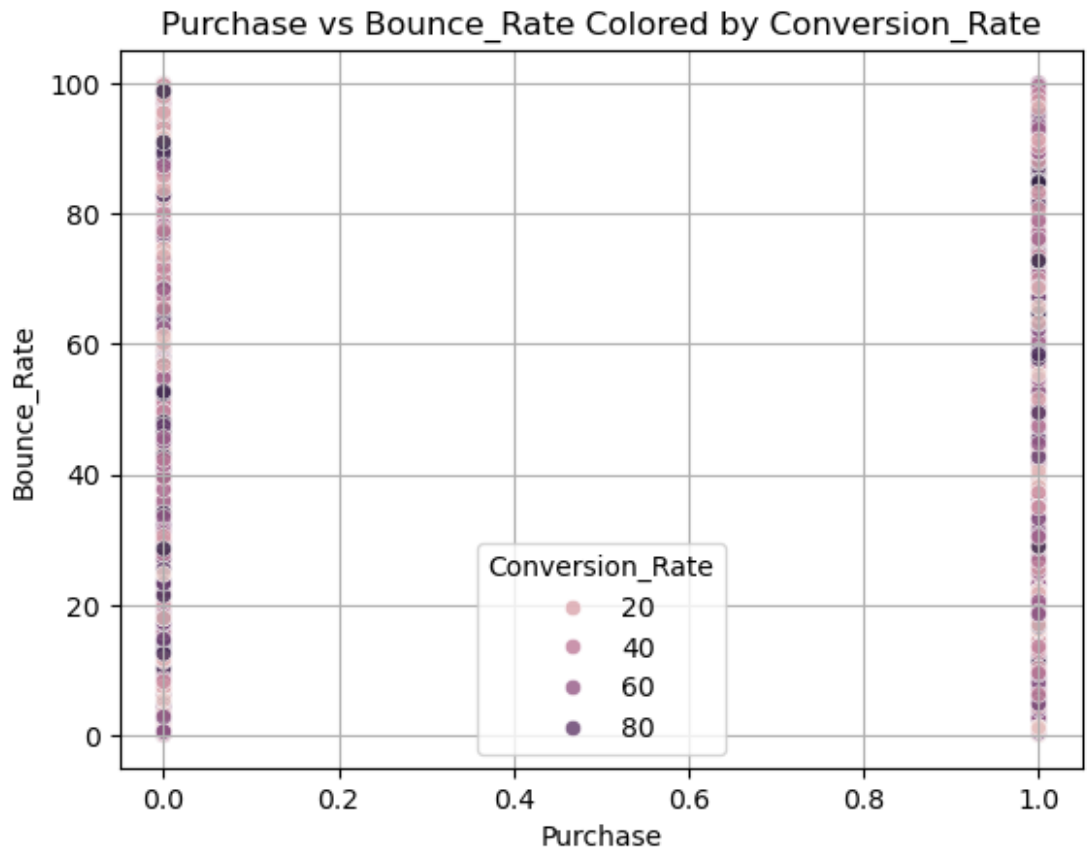

Boxplot of Conversion_Rate
0
20
40
60
80
100
Conversion_Rate






```
# scatterplot (Purchase vs bonus rate by conversion rate)
sns.scatterplot(x='Purchase', y='Bounce_Rate',hue='Conversion_Rate',
data=data, alpha=0.8)
plt.title('Purchase vs Bounce_Rate Colored by Conversion_Rate')
plt.xlabel('Purchase')
```

```
plt.ylabel('Bounce_Rate')
plt.grid(True)
plt.show()
```



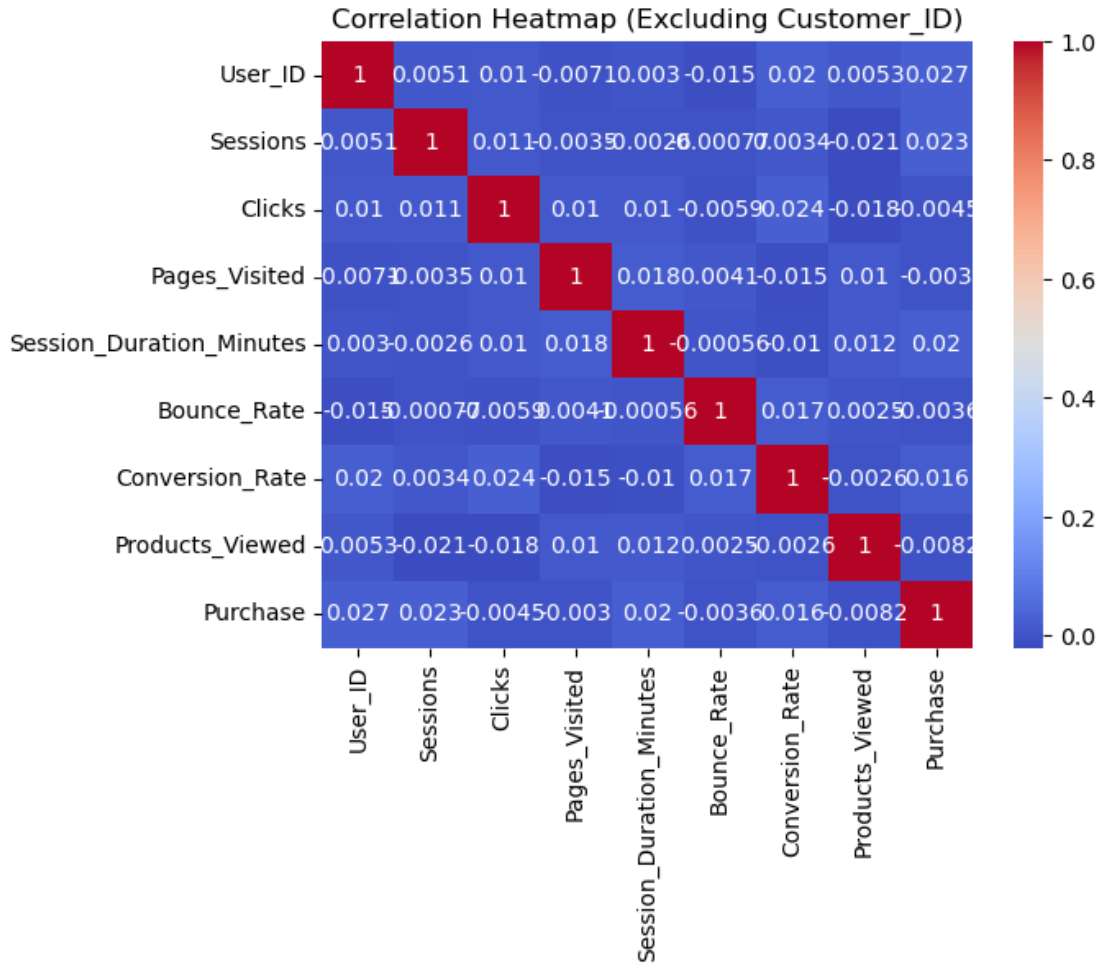
☝ Scatterplot shows us the relationship between variables and also to find outliers.

- Show a negative or positive trend between Purchase and Conversion_Rate.
- Help us to make business decision, while we notice a Bounce_Rate increases and Conversion_Rate increases.

🔥 Heatmap - Correlation Matrix

```
# 📌 8. Correlation Analysis (Updated: Dropping Customer_ID)
# Drop object-type columns like 'Customer_ID' before calculating correlation
data_corr = data.drop(columns=['Customer_ID'], errors='ignore')

# Select only numeric columns
data_corr = data_corr.select_dtypes(include=np.number)

# Compute and plot correlation
corr = data_corr.corr()
sns.heatmap(corr, annot=True, cmap="coolwarm")
```

```
plt.title("Correlation Heatmap (Excluding Customer_ID)")
plt.show()
```



☝ Correlation matrix helps us to find the most target variable and it shows the high relationship between variables in positive or negative ranges which is useful for predictation, and it detect multicollinearity features.

- +1 = Perfect positive correlation
- -1 = Perfect negative correlation
- 0 = No linear relationship

📊 Univariate and Multivariate Analysis

🔹 Univariate Analysis

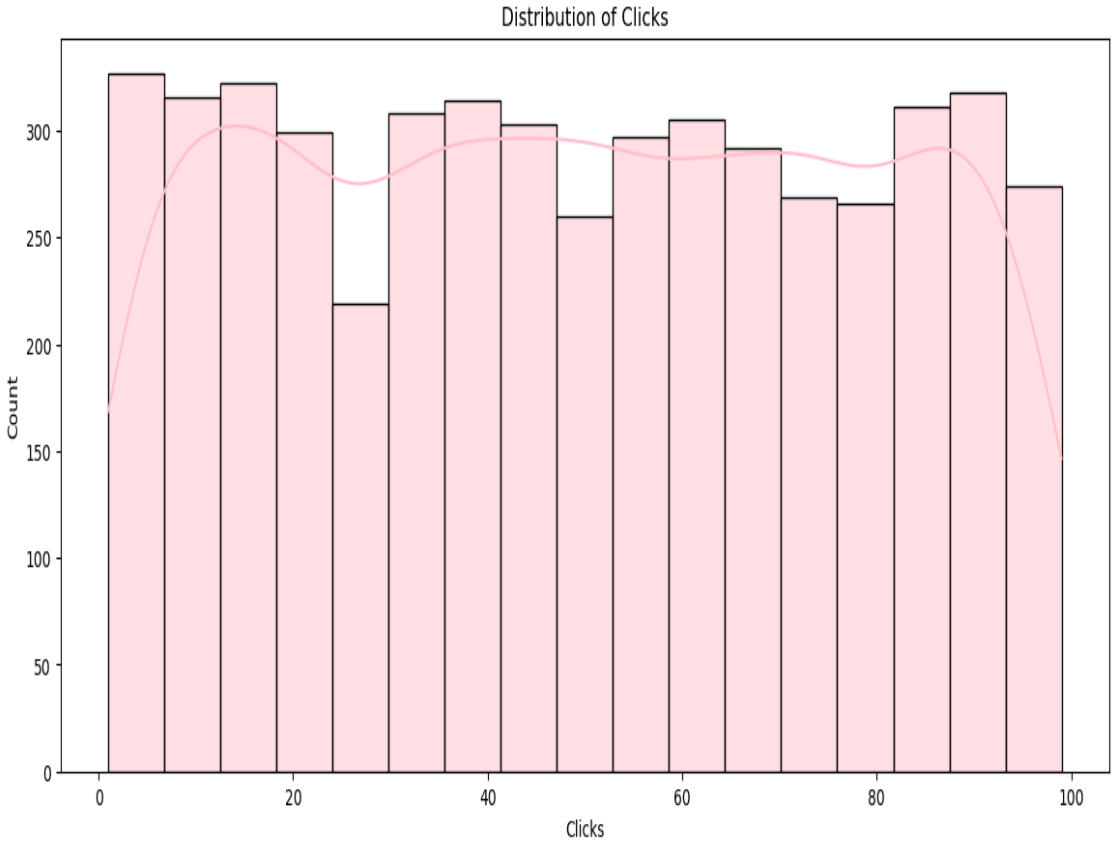
Univariate analysis is a statistical examination of a single variable at the time.it helps to describe the basic characteristics of the data, such as its distribution, central tendency(mean,median,mode),and description(range,variance,standard deviation).

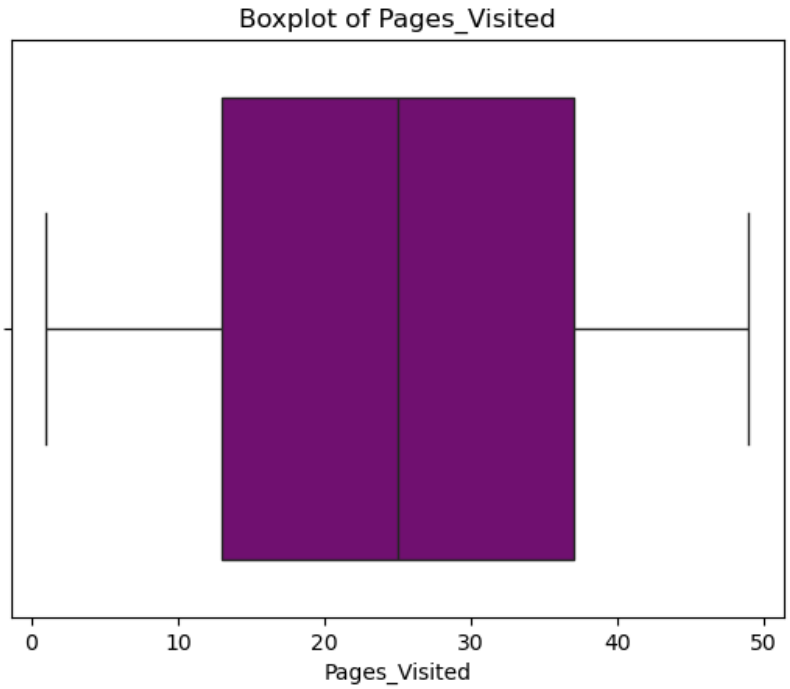
◆ Multivariate Analysis

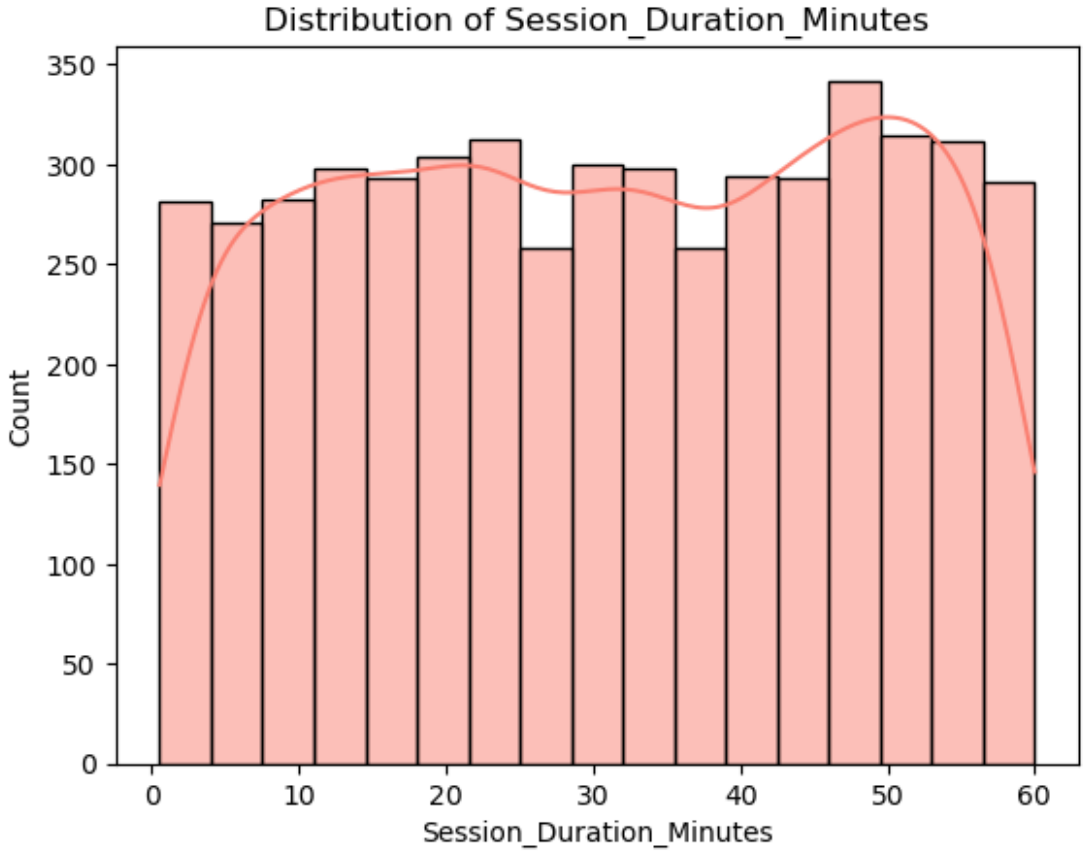
Multivariate analysis is a statistical technique used to examine the relationship between three or more variables at the same time. it helps to identify patterns, correlations and interactions among variables to understand how they influence each other.

We'll explore both types of analysis using visualizations and summary Summary statistics for key columns like `Clicks,Pages_Visited,Session_Duration_Minutes,Bouns_Rate,Conversion_Rate,Product_Viewed,andPurchase`.

```
# Univariate Analysis - Distribution plots for numeric variables
plt.figure(figsize=(14, 6))
sns.histplot(data['Clicks'], kde=True, color='pink')
plt.title('Distribution of Clicks')
plt.show()

sns.boxplot(x=data['Pages_Visited'], color='purple')
plt.title('Boxplot of Pages_Visited')
plt.show()

sns.histplot(data['Session_Duration_Minutes'], kde=True, color='salmon')
plt.title('Distribution of Session_Duration_Minutes')
plt.show()
```

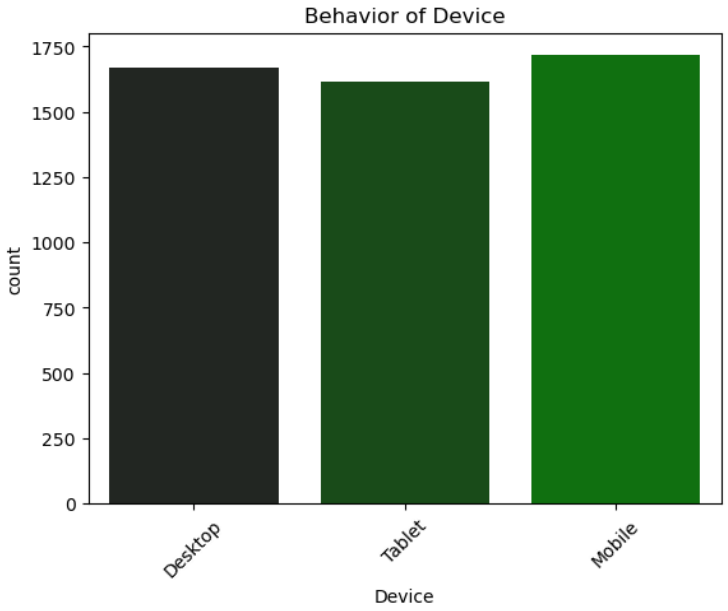


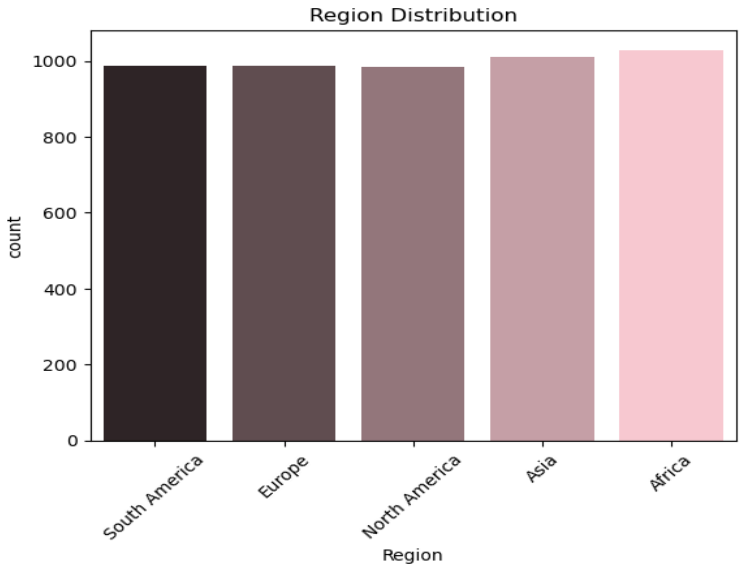

Boxplot of Pages_Visited
0
10
20
30
40
50
Pages_Visited


Distribution of Session_Duration_Minutes
Count
0
50
100
150
200
250
300
350
0
10
20
30
40
50
60
Session_Duration_Minutes

Univariate Analysis - 📐 Categorical variable count plot

```
# Univariate Analysis - Categorical variable count plot
sns.countplot(x='Device', hue="Device", data=data, color='green')
plt.title('Behavior of Device')
plt.xticks(rotation=45)
plt.show()

sns.countplot(x='Region', hue='Region',data=data, color='pink')
plt.title('Region Distribution')
plt.xticks(rotation=45)
plt.show()
```





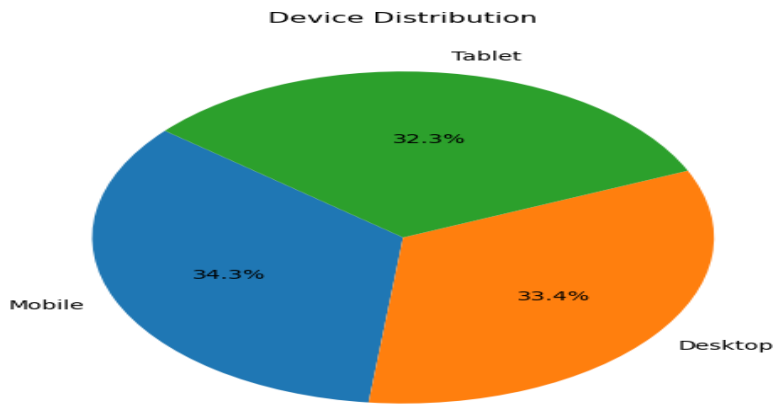
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   User_ID                   5000 non-null   int64
 1   Sessions                  5000 non-null   int64
 2   Clicks                    5000 non-null   int64
 3   Pages_Visited             5000 non-null   int64
 4   Session_Duration_Minutes  5000 non-null   float64
 5   Bounce_Rate               5000 non-null   int32
 6   Conversion_Rate           5000 non-null   int32
 7   Products_Viewed           5000 non-null   int32
 8   Purchase                  5000 non-null   datetime64[ns]
 9   Device                    5000 non-null   object
 10  Region                    5000 non-null   object
 11  User_Type                 5000 non-null   object
dtypes: datetime64[ns](1), float64(1), int32(3), int64(4), object(3)
memory usage: 410.3+ KB
```

👇🏻 Piechat shows has that how much each product category contributes to the whole dataset like:

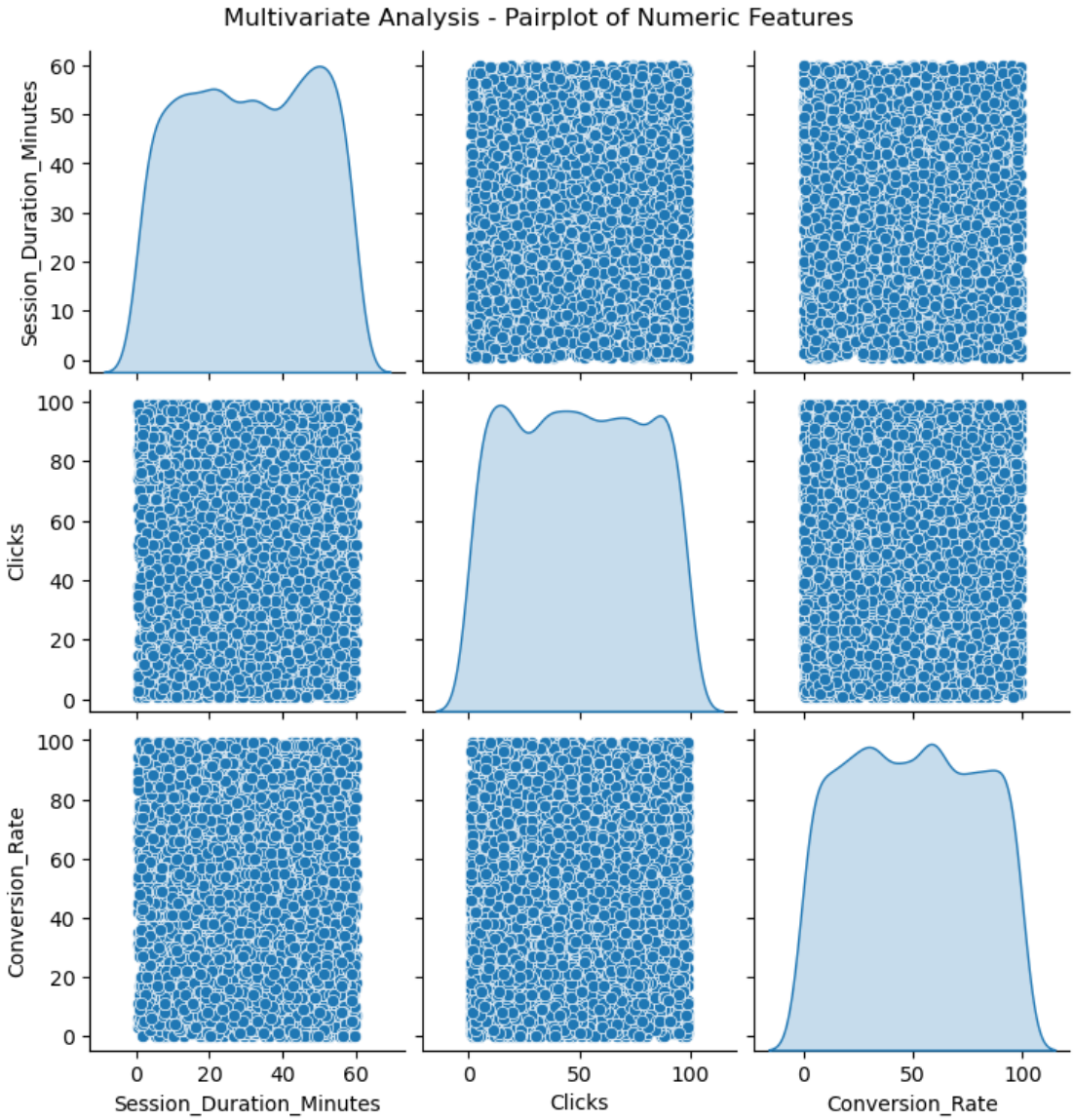
- Which category sells more.
- Which one to promote.
- Where to focus Devices.

```
# Univariate Pie Chart: Device distribution
Device_counts = data['Device'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(Device_counts, labels=Device_counts.index, autopct='%1.1f%%',
startangle=140)
plt.title('Device Distribution')
plt.show()
```



Multivariate Analysis - 🌀 Pairplot of key numeric features

```
# Multivariate Analysis - Pairplot of key numeric features
sns.pairplot(data[['Purchase', 'Session_Duration_Minutes', 'Clicks',
'Conversion_Rate']], diag_kind='kde')
plt.suptitle('Multivariate Analysis - Pairplot of Numeric Features', y=1.02)
plt.show()
```

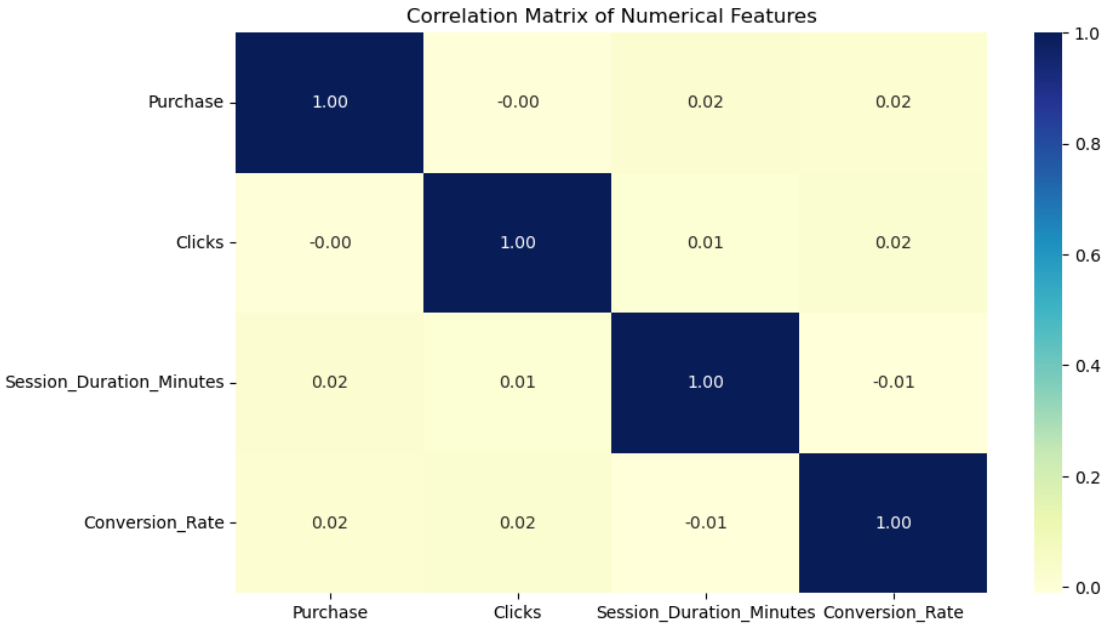


👆 Pair plot is used to Create scatter plots between every pair of numerical features.

- **pairplot** : [Session_Duration_Rate , Clicks , Conversion_Rate , Purchase]

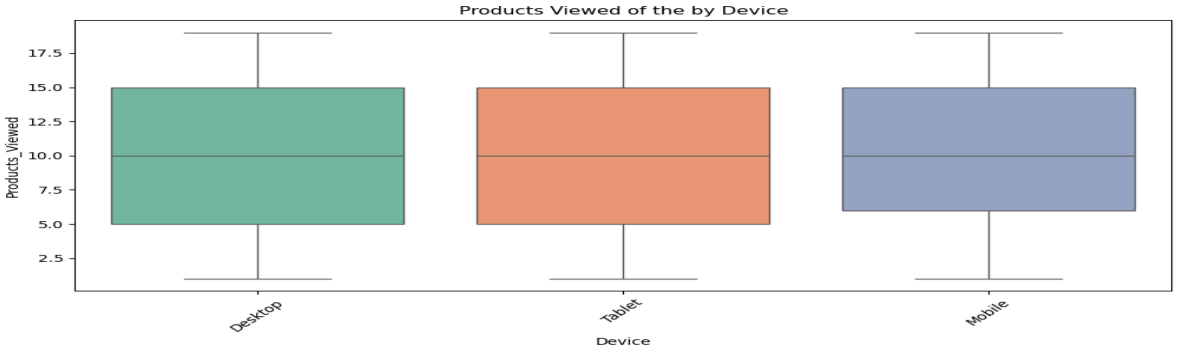
Multivariate Analysis - 🔥 Heatmap for correlation

```
# Multivariate Analysis - Heatmap for correlation
plt.figure(figsize=(10, 6))
corr = data[['Purchase', 'Clicks', 'Session_Duration_Minutes',
'Conversion_Rate']].corr()
sns.heatmap(corr, annot=True, cmap='YlGnBu', fmt='.2f')
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```



👆This heatmap is meaningful only if categorical variables are converted into numerical variables.

```
# Multivariate Analysis - Boxplot of Lifetime Value by Region
plt.figure(figsize=(12, 5))
sns.boxplot(x='Device', y='Products_Viewed', data=data, palette='Set2')
plt.title('Products Viewed of the by Device')
plt.xticks(rotation=45)
plt.show()
```



👆🏻 Boxplot is used to spread data and detect outliers in the dataset and it helps us to find extreme values.

- Helps to compare how Products_Viewed vary with different Device conditions.
- it is useful for identifying inconsistencies in stock levels based on device demand.

🎲 Probability & Hypothesis Testing

Here, We:

- Shapiro Test for normality
- T-test or ANOVA: Compare session duration by event type
- Chi-Square Test: Relationship between event type and product Category
- Validate assumptions like "Do products is different categories which is "mobile vs laptop" have significantly different average prices?"

📙 Probability Distribution Analysis

It is a process of understanding how values in a column are distributed meaning how frequently they occur and how they behave across the data set. To understand how certain numerical variables behave across the user behaviors,product prizes, categories and user sessions. so analyze the probability distribution helps you to understand customer behavior, product pricing patterns, business strategy, trends, and detect patterns.

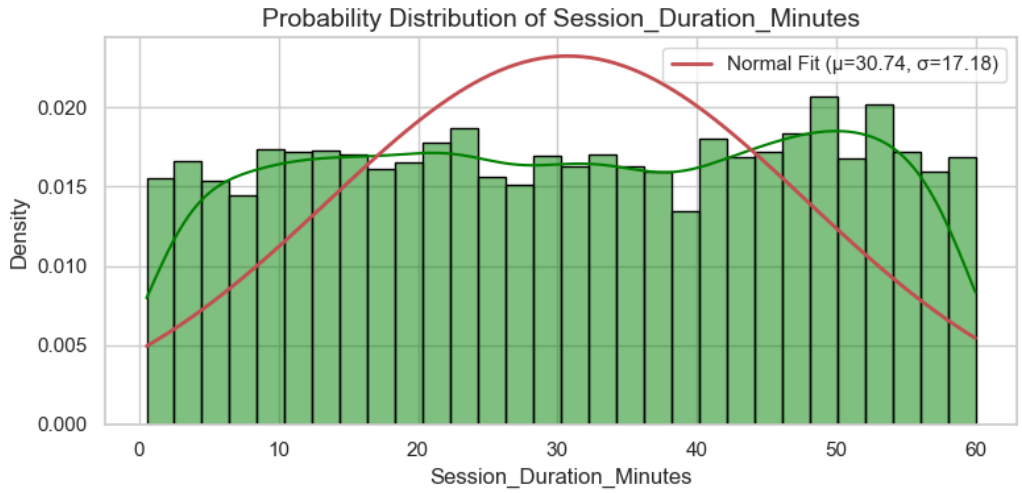
🧨 Columns Chosen:

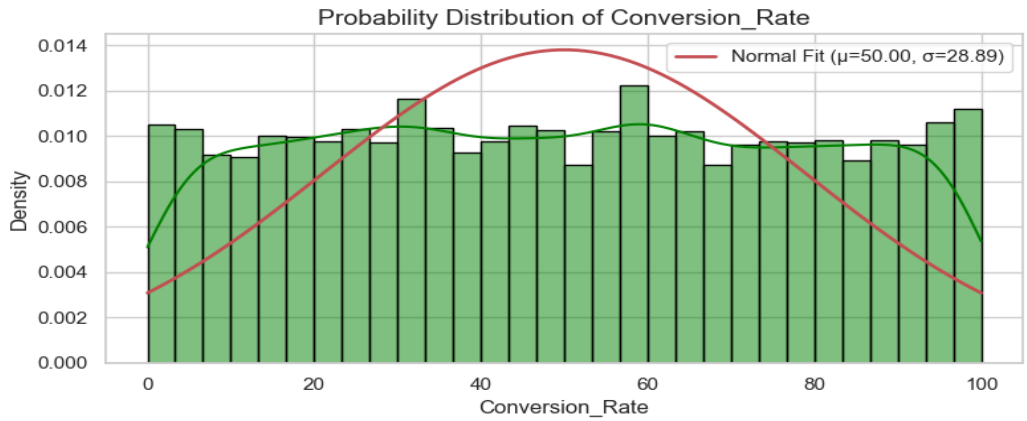
- **Session_Duration_Minutes**- How long users spend during a session.
- **Conversion_Rate**- How effective the platform is at turning views into buys.
- **Bounce_Rate**-percent Of users who leave after one interaction.
- **Clicks**- Which users are most engaged.
- **Purchases**- Buying behavior across users.

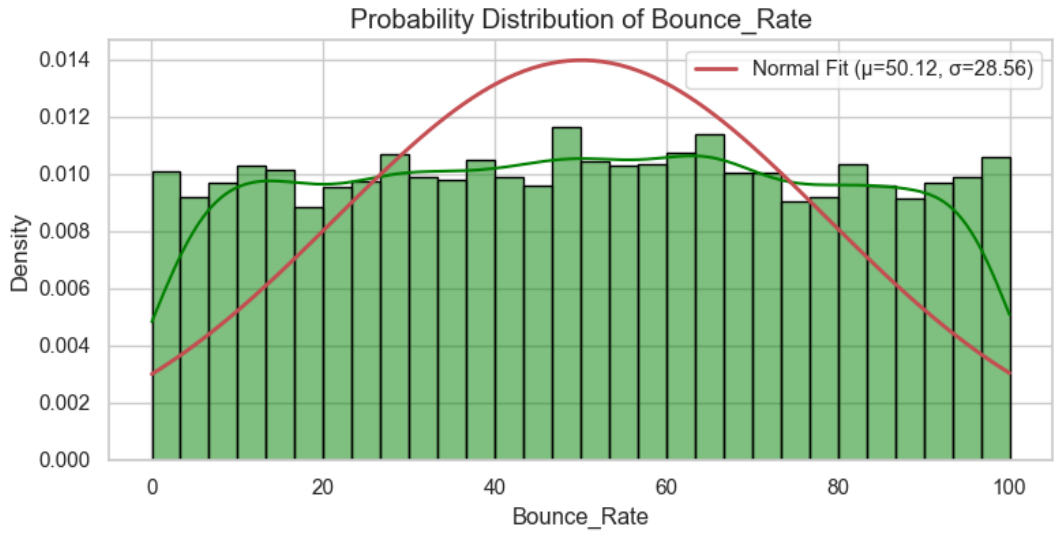
Each variable is visualized below with both histogram and normal distribution fit

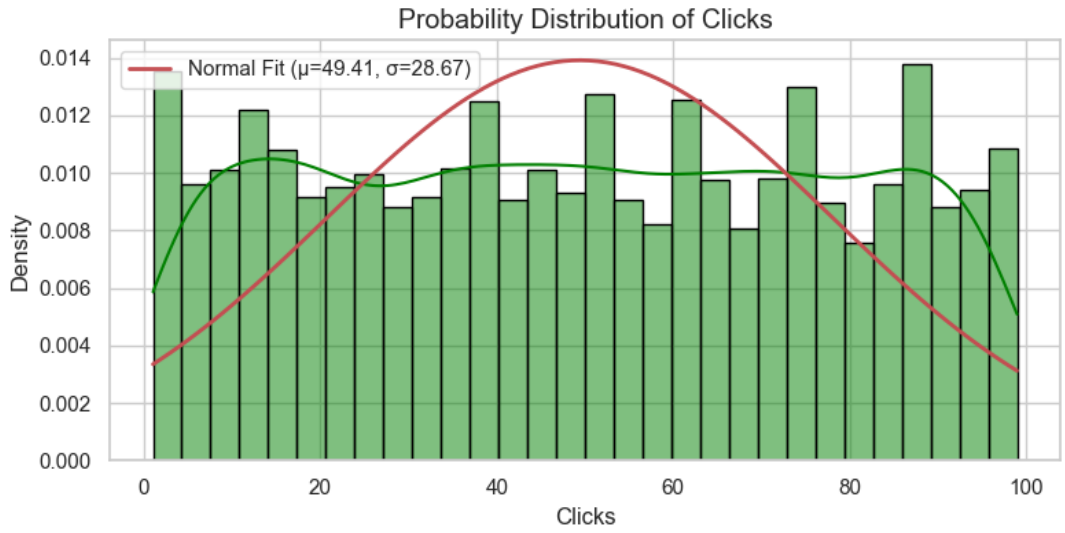
```
# Set the style for all plots
sns.set(style="whitegrid")

# Columns selected for probability distribution
columns_to_analyze =
['Session_Duration_Minutes','Conversion_Rate','Bounce_Rate','Clicks','Purchas
e']

# Plotting histogram with KDE and normal distribution fit
for col in columns_to_analyze:
    plt.figure(figsize=(8, 4))
```

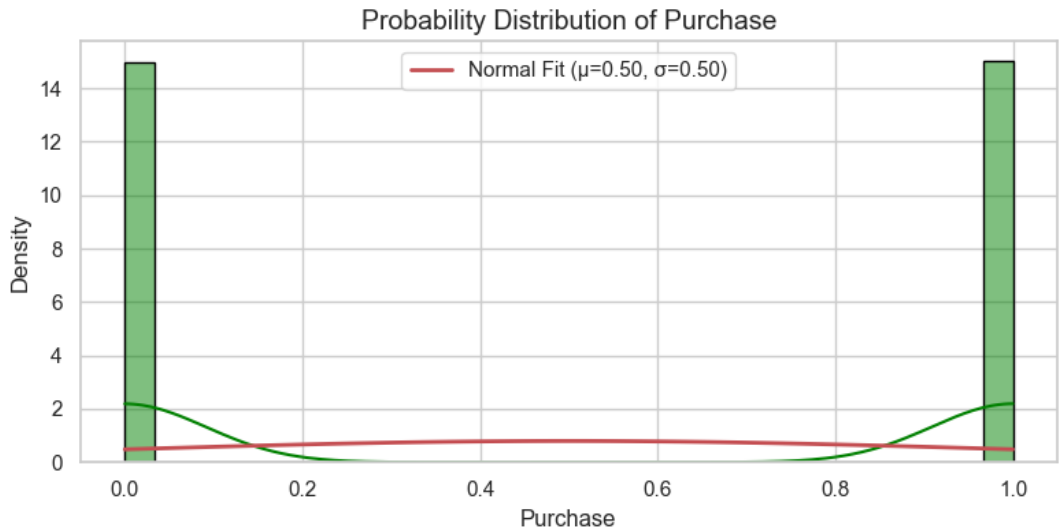
```
    # Plot histogram and KDE
    sns.histplot(data[col], kde=True, stat="density", bins=30, color="green",
edgecolor="black")

    # Fit normal distribution to the data
    mean, std = norm.fit(data[col])
    x = np.linspace(data[col].min(), data[col].max(), 100)
    p = norm.pdf(x, mean, std)

    # Plot fitted normal distribution
    plt.plot(x, p, 'r', linewidth=2, label=f'Normal Fit (μ={mean:.2f},
σ={std:.2f})')

    # Titles and labels
    plt.title(f'Probability Distribution of {col}', fontsize=14)
    plt.xlabel(col, fontsize=12)
    plt.ylabel('Density', fontsize=12)
    plt.legend()
    plt.tight_layout()
    plt.show()
```






Probability Distribution of Bounce_Rate
Normal Fit (μ=50.12, σ=28.56)
Density
Bounce_Rate
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0
20
40
60
80
100


Probability Distribution of Clicks
Normal Fit (μ=49.41, σ=28.67)
Density
Clicks
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0
20
40
60
80
100


Probability Distribution of Purchase
Normal Fit (μ=0.50, σ=0.50)
Density
Purchase
0
2
4
6
8
10
12
14
0.0
0.2
0.4
0.6
0.8
1.0

📝 Hypothesis Testing

by doing t-Test, chi-square test that helps us to make data-driven decisions by Statistically evaluating patterns assumptions and differences in data.

```
# 📌 11. Hypothesis Testing

# Choose two regions
region1 = data[data['Region'] == 'North']['Session_Duration_Minutes']
region2 = data[data['Region'] == 'South']['Session_Duration_Minutes']

# Perform independent two-sample t-test
t_stat, p_value = ttest_ind(region1, region2)

print("T-Statistic:", t_stat)
print("P-Value:", p_value)

# Interpretation
alpha = 0.05
if p_value < alpha:
    print("Reject Null Hypothesis: There is a significant difference in
Average Order Value between North and South regions.")
else:
    print("Fail to Reject Null Hypothesis: No significant difference found.")
```

```
T-Statistic: nan
P-Value: nan
Fail to Reject Null Hypothesis: No significant difference found.
```

📝 Simple Linear Regression

We predict `Conversion_Rate` using `Bounce_Rate` to Understand how prices is affects in Products.

```
# 📌 9. Simple Linear Regression (Example)
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
# Independent (X) and dependent (y) variables
X = data[['Conversion_Rate']]  # must be 2D for skLearn
y = data['Bounce_Rate']

# Create and fit the model
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

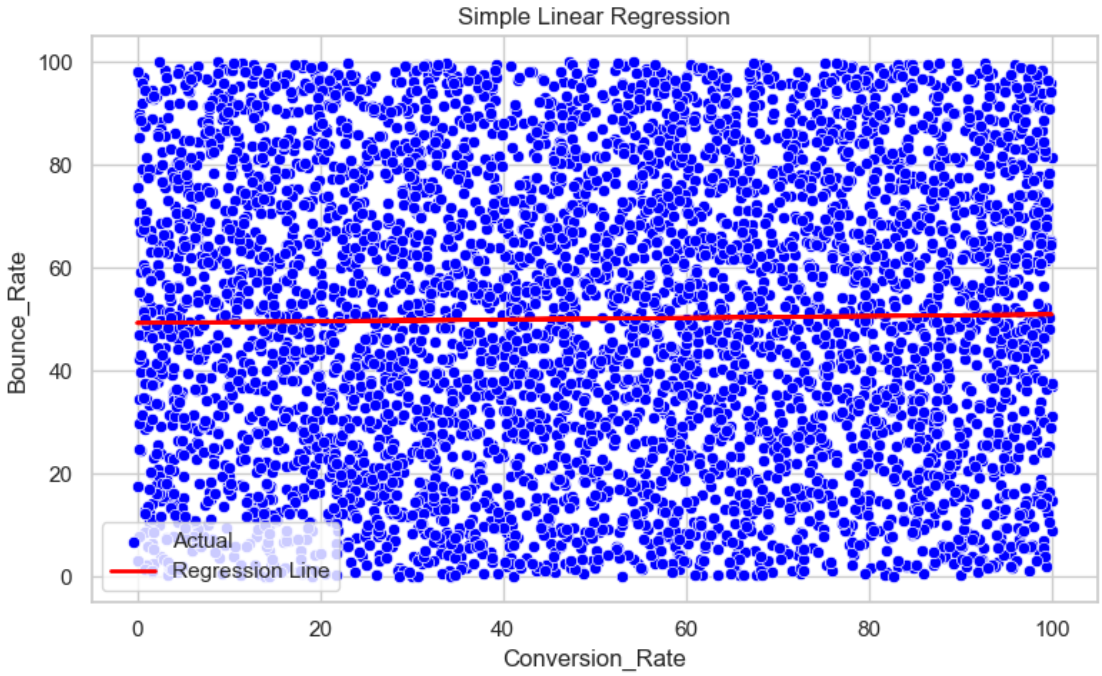
```
LinearRegression
LinearRegression()
```

```
# Predict values
y_pred = lin_reg.predict(X)
```

```
y_pred
```

```
array([49.6107294 , 50.47521701, 49.80839531, ..., 50.74184468,
       49.74502769, 50.11282712])
```

```
# Plotting regression line
plt.figure(figsize=(8, 5))
sns.scatterplot(x=X['Conversion_Rate'], y=y, color='blue', label='Actual')
plt.plot(X['Conversion_Rate'],y_pred, color='red', linewidth=2,
label='Regression Line')
plt.xlabel('Conversion_Rate')
plt.ylabel('Bounce_Rate')
plt.title('Simple Linear Regression')
plt.legend()
plt.tight_layout()
plt.show()
```



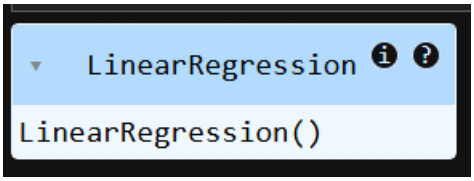

```
# Display coefficients
print("Intercept:", lin_reg.intercept_)
print("Slope:", lin_reg.coef_[0])
```

```
Intercept: 49.280092033529755
Slope: 0.016731517138996402
```

📄 Multiple Linear Regression

We now use tow variables to predict `Bounce_Rate` to improve our production accuracy and capture more user behavior.

```
# 📌 10. Multiple Linear Regression (Example)

# Define features and target
features = ['Conversion_Rate', 'Purchase']
X = data[features]
y = data['Bounce_Rate']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create and train model
mlr = LinearRegression()
mlr.fit(X_train, y_train)
```




```
# Predict
y_pred = mlr.predict(X_test)

y_pred
```

Output:

```
array([50.2582044 , 49.05001977, 50.30918921, 50.55997393, 50.52160887,
       49.52916551, 49.63203813, 49.64341357, 49.79886124, 50.19622811,
       49.93153571, 49.47939315, 50.36406185, 49.85201642, 50.55043025,
       49.08664665, 50.42648922, 49.34615044, 49.6176836 , 49.33332521,
       49.19279303, 50.51777394, 50.04902024, 49.80386424, 50.38879927,
       50.30047268, 50.16647465, 50.04323244, 50.2737358 , 50.31026656,
       50.27544293, 49.14581546, 49.40481221, 49.71324466, 50.5396729 ,
       50.08928668, 49.66546902, 49.5004304 , 49.58115553, 49.0352563 ,
       49.90774579, 49.33727183, 49.90098976, 50.04775917, 50.60158884,
       50.07725814, 49.56368213, 50.58108744, 50.49565985, 50.53213736,
       49.23851551, 49.62077252, 49.35874137, 49.22371245, 50.16330455,
       49.39923303, 49.48929356, 50.05191794, 49.79346315, 49.78982333,
```

49.61845607, 49.82646689, 49.88650376, 49.19323827, 50.60815701,
49.9746797 , 49.51652449, 49.57390263, 49.0893299 , 50.21810909,
49.70046871, 49.40066917, 49.54384041, 49.06740574, 50.5432641 ,
49.66642826, 50.46425899, 49.2931146 , 50.52496906, 50.43415791,
49.40924461, 49.45772965, 49.02307152, 49.08116619, 49.79137201,
50.16189515, 49.54994713, 50.3909694 , 50.51095972, 49.39493348,
49.84577013, 50.23698329, 50.34015423, 49.66103915, 49.92279464,
49.63309327, 49.24627398, 50.20702926, 49.81687645, 50.05587421,
50.58751671, 50.55416109, 49.73702023, 49.75184819, 49.90589571,
49.24763911, 49.08771626, 49.68753951, 50.40141152, 50.0022861 ,
49.6443234 , 50.11453018, 50.23886301, 50.50800453, 49.63406239,
49.74783236, 49.69224517, 49.60624901, 50.30784199, 49.92000492,
49.15582627, 49.55535168, 49.53509342, 49.6097368 , 50.055948 ,
50.42205602, 50.56926054, 50.01219845, 49.77967769, 49.66337903,
50.5888034 , 50.17064265, 50.3783249 , 49.48662725, 50.10036447,
50.55481188, 50.54675026, 50.21053566, 49.75971591, 50.08719433,
50.06403362, 49.12557709, 49.11072377, 50.60482695, 49.62490734,
49.73723907, 49.25724186, 50.36983036, 50.55314103, 49.11418535,
49.20052403, 50.03401108, 50.12687482, 50.41335132, 50.48590228,
49.79194364, 50.34832705, 49.38611549, 49.89744957, 49.05436626,
50.49388816, 50.11074273, 49.62371945, 49.31945392, 50.0868967 ,
49.217342 , 50.0902106 , 49.89742418, 49.05928667, 50.13027646,
50.50925773, 50.59025032, 49.04079178, 50.58818608, 49.72639466,
49.57707435, 50.61531082, 49.09437269, 50.09035709, 49.43871713,
49.3907257 , 49.94032838, 50.57220221, 49.1367904 , 49.84912217,
49.49772165, 49.50218144, 49.99257594, 49.58483059, 50.17754646,
49.90116027, 49.37174251, 49.89128017, 50.39108267, 50.50924442,
49.75054259, 49.73145039, 49.06498644, 50.04320527, 50.30574783,
49.85478247, 49.67914682, 49.30247197, 49.77681339, 50.18245136,
50.31512949, 49.30829808, 50.3074938 , 49.33946626, 49.63929418,
49.50658635, 49.41460366, 49.56205319, 49.16302818, 50.14705811,
50.33686983, 49.79527836, 49.22052807, 50.09410406, 49.89134541,
49.58510274, 49.93434201, 49.14021259, 49.97100182, 49.74812893,
50.280661 , 49.67633139, 50.53865144, 49.73817455, 49.36178333,
50.50514328, 49.97788983, 49.37737909, 49.87336556, 50.19554432,
49.80747895, 49.92343736, 49.63000249, 49.19624586, 49.72636222,
49.52345385, 49.81814299, 50.54629774, 49.12679162, 49.70505975,
49.72432934, 49.15679273, 49.74098278, 49.7889311 , 50.24619   ,
50.33329184, 50.58016227, 49.30673547, 49.39502417, 49.51413791,
49.46523593, 50.34209768, 49.79634252, 49.34350405, 49.1964926 ,
50.32224073, 49.33182108, 49.26463474, 49.98452993, 50.55646643,
49.50149478, 49.55354936, 50.02861846, 49.8379331 , 50.22274277,
49.91881149, 50.19547311, 50.46831104, 49.62874258, 49.62641642,
49.78469046, 49.29711923, 50.29909518, 49.38701379, 49.54064911,
49.61077427, 49.39404103, 50.33031921, 50.59189871, 50.40674575,
50.49416961, 50.02597636, 49.95044135, 49.06799307, 49.0401479 ,
49.41136797, 49.08058744, 49.26718573, 49.38178987, 49.88697598,
50.05527617, 49.88001893, 49.01660713, 49.14534143, 49.43889146,
49.11275387, 49.32809618, 50.04374489, 50.23808313, 49.05228467,
49.28326278, 49.14888389, 50.20712634, 50.59612108, 49.78822798,

50.37704772, 49.87853029, 49.09282287, 50.61179431, 50.08388313,
49.09726024, 49.43189993, 49.46297284, 49.88565277, 49.52321663,
49.85860779, 50.53243689, 49.50243298, 49.96860022, 50.30679989,
49.85046199, 49.71738089, 49.77046713, 50.46493194, 49.37882101,
50.33882656, 49.59624025, 49.76018867, 50.31601873, 49.48280986,
49.86652543, 49.12681442, 49.5347743 , 49.74487389, 49.56164742,
49.96431241, 50.53790273, 50.38461989, 49.2707478 , 49.38199224,
49.58654644, 50.23939413, 50.19399329, 49.0710276 , 50.10319333,
49.17081111, 49.42393652, 49.9847237 , 49.37703729, 49.17800855,
49.43530011, 49.1923165 , 49.03395268, 49.86238051, 50.37244238,
49.7744891 , 49.54169746, 50.06196277, 50.52509939, 50.00872445,
50.00446969, 49.25212185, 49.34692163, 50.48324697, 49.57979276,
50.34407949, 49.09877452, 50.03260602, 50.10532853, 50.04240744,
49.34017994, 49.30529206, 49.69320285, 49.76775249, 49.55943786,
50.10784589, 50.05325991, 49.33484486, 50.48344512, 49.06195488,
50.4055399 , 50.14181978, 50.1737231 , 49.11828273, 49.54614769,
49.61109098, 49.4548627 , 50.17147835, 49.39210293, 50.57056631,
49.28990402, 49.19579093, 50.30419157, 50.03763542, 50.24618861,
49.21983715, 50.07852891, 49.23820112, 50.02687012, 50.10617843,
50.37461161, 49.58677441, 49.75629379, 50.1124028 , 49.18191471,
50.4016461 , 49.41318411, 50.27564113, 49.89797203, 49.51326923,
50.41973249, 50.50277884, 49.22744457, 49.92090181, 50.47952838,
50.31576074, 49.13583173, 49.80028009, 50.60315611, 50.29803408,
49.01841663, 49.06806265, 50.2909132 , 50.46845007, 49.27510348,
50.63040868, 50.49685358, 50.05600732, 49.89429231, 49.16050449,
49.73652506, 50.36932601, 49.70927122, 50.09236206, 50.45516602,
49.76041608, 50.00373705, 49.99202148, 50.03624037, 50.30570434,
49.44680316, 49.88173573, 50.50268065, 49.78228209, 49.33394224,
49.86255189, 50.07204265, 50.36863591, 50.36174736, 50.59265563,
49.74679724, 49.82244465, 49.35962369, 49.58540255, 49.81255621,
49.50696341, 50.23720116, 49.56639626, 49.42252223, 49.56744686,
50.01663043, 49.50931674, 49.78889171, 49.75218987, 50.53401471,
49.30297691, 50.14830565, 50.14715442, 49.40840417, 50.33878731,
50.36787911, 49.43118924, 49.97555205, 49.28183902, 49.65769725,
49.32023147, 50.55890756, 50.05000229, 50.23689685, 50.12515127,
50.51893449, 50.38843106, 50.34240034, 49.96124674, 49.67139663,
49.50672505, 49.13583866, 50.42586124, 49.15038125, 49.94475964,
50.29511567, 49.26169979, 49.9029434 , 49.76493817, 49.67122619,
49.46706883, 49.99851306, 49.92463481, 49.54066852, 50.49623484,
50.35552317, 50.11081144, 50.48027173, 50.30509088, 50.28193712,
49.73893428, 50.25248641, 49.26379412, 49.27459826, 50.39932044,
49.62912667, 49.05404875, 49.10789206, 49.38769811, 49.13342005,
49.98177432, 50.42025973, 49.70853977, 49.19694341, 49.98416457,
50.27014139, 49.35906627, 50.35621218, 50.06139233, 49.47546413,
49.33922204, 49.98444392, 49.35575514, 49.68355537, 50.32399229,
49.46461805, 50.17757478, 50.30897325, 49.43595449, 49.39620634,
49.11297786, 50.23261367, 49.04341346, 50.23522686, 49.69143516,
49.98793454, 49.4997479 , 49.56854106, 50.38847934, 49.39664389,
50.16550171, 49.76101221, 49.13058688, 49.0210665 , 50.03307439,
49.88763892, 50.45943212, 49.45637227, 49.23183821, 49.7636306 ,

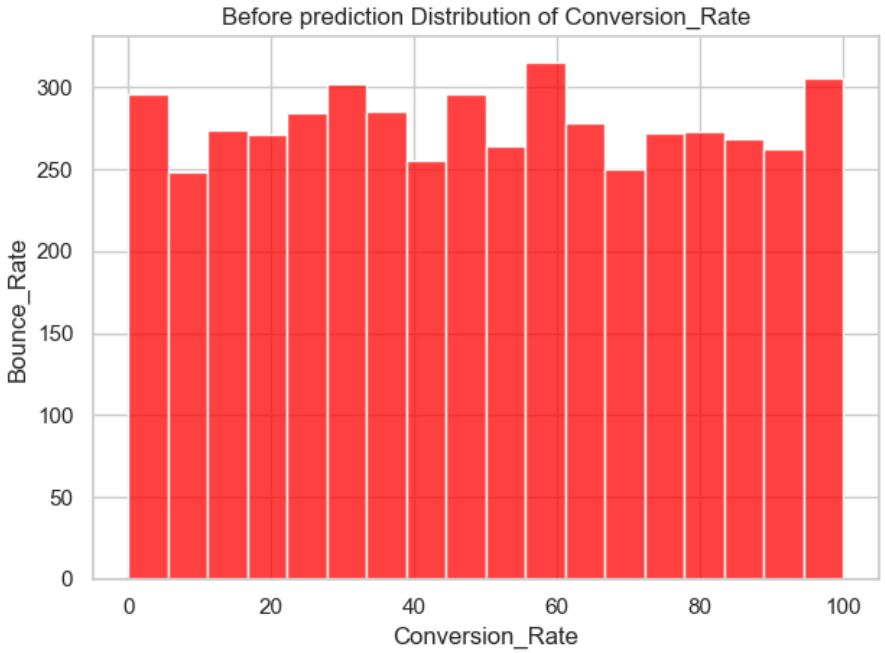
49.22261457, 49.97141098, 49.57263033, 49.74779373, 49.80592873,
49.20365961, 50.56031459, 49.86384313, 49.66638083, 49.0588278 ,
49.2774849 , 49.90222054, 49.10592259, 49.322187 , 49.45211668,
49.95978867, 49.12506836, 50.32651711, 50.19220647, 50.21931285,
50.23922292, 49.25305797, 49.94404884, 50.42907998, 49.25353791,
49.38168467, 50.10207774, 49.77809261, 50.04089468, 49.85562028,
50.32342201, 50.46719027, 49.40505489, 49.30767871, 49.70179519,
50.54897933, 49.34719882, 49.07693975, 50.05748837, 49.52057037,
50.2679773 , 50.30707464, 49.67538358, 49.02846721, 50.25677685,
49.94688496, 50.18507486, 50.30782911, 50.28089301, 50.45342466,
50.41019909, 49.60602685, 50.11172367, 50.02560876, 49.76876336,
50.35768578, 49.0307686 , 49.92062864, 50.218838 , 49.95303876,
50.24038306, 50.50467417, 50.0627301 , 50.14596816, 49.61027249,
49.36200685, 50.52393569, 49.54306328, 50.2809617 , 49.73350578,
49.55258181, 50.14953644, 49.40945036, 49.61647063, 49.9409825 ,
49.67275721, 50.01995114, 49.50190324, 49.9544921 , 50.58909665,
50.29529711, 49.61765114, 49.25493241, 49.61062055, 49.16639849,
49.56068249, 49.48258346, 49.64177143, 49.69180896, 50.38151209,
50.39238361, 49.5411229 , 49.5705035 , 49.96597753, 50.5847494 ,
49.41785885, 50.53785366, 50.04697188, 50.11460877, 50.510148 ,
50.59507775, 49.27161225, 50.49595146, 50.10862355, 49.75002326,
49.62592969, 49.87172249, 50.49891905, 49.39938837, 50.40023752,
49.59047875, 50.53026214, 49.30665004, 50.61177928, 50.52430526,
50.49311293, 49.63230515, 50.4750647 , 49.22648842, 50.24074078,
49.24340028, 49.42579019, 49.74667779, 50.21447023, 50.26252218,
49.79081427, 49.41235642, 50.49334021, 50.48004501, 49.30031922,
49.28453272, 49.27091119, 49.36934846, 49.39878794, 49.6155559 ,
50.48667124, 49.97596019, 50.51595833, 50.55464987, 49.08259561,
50.2085683 , 49.70025834, 49.37802583, 49.9701025 , 50.06156459,
50.30435245, 49.13307845, 49.05781121, 49.26133135, 49.62205528,
49.62889582, 49.6688136 , 49.80204766, 50.2138639 , 50.07134236,
50.46548707, 49.26063501, 49.85018303, 49.62118136, 50.3697714 ,
50.43545341, 50.15332553, 50.3725477 , 50.41495452, 50.5791849 ,
49.55864096, 49.90377996, 49.89545937, 49.45204624, 49.59981119,
49.80038563, 50.58463918, 50.35314052, 50.52259084, 49.80423001,
50.12606026, 49.9324584 , 49.33046642, 49.83106454, 49.52765399,
50.5341924 , 49.43251857, 49.62102602, 49.6799522 , 50.55395449,
49.61494268, 50.08358564, 50.0508807 , 49.15959932, 49.76252028,
49.75360414, 50.53591059, 49.48413952, 49.8147452 , 49.39619732,
50.25765867, 49.61610625, 50.12016182, 50.42651628, 50.13669102,
50.10638468, 49.54116667, 49.28449583, 49.42966593, 50.3406637 ,
49.39663103, 49.38193903, 49.35756287, 49.65264408, 49.40285712,
49.1572403 , 50.38694966, 49.73740632, 49.84153379, 49.80782159,
50.28783055, 49.30727348, 50.01108978, 49.55347265, 50.58307007,
49.71694698, 49.02440074, 49.28878897, 50.38713471, 49.98609571,
50.21014109, 49.2500241 , 49.92855352, 49.02869349, 49.22975807,
50.54945406, 50.18949925, 49.52211316, 49.56540158, 49.37733101,
50.01186456, 49.52478915, 50.49382816, 49.13351854, 50.59103457,
49.80575993, 50.09032023, 49.9152828 , 49.57392041, 49.09168589,
49.98005891, 50.5400283 , 50.58451103, 49.41441455, 50.33502143,

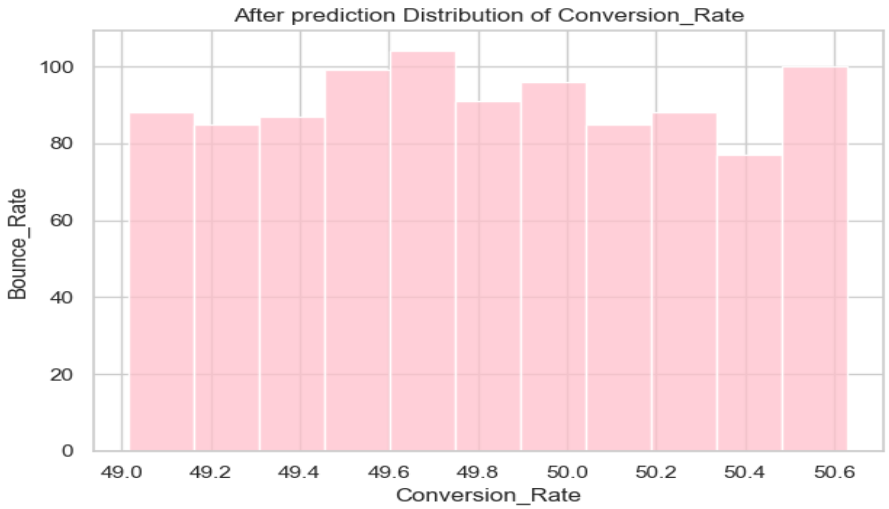
```
       49.03309186, 49.87980559, 50.52625326, 50.04156175, 49.47738568,
       50.47688469, 50.11873567, 49.92211737, 50.57722683, 49.30119371,
       49.04629342, 50.0548377 , 49.08875189, 50.10897724, 50.16793107,
       49.77709768, 49.78163814, 49.09690727, 49.59061698, 50.5705602 ,
       49.41070712, 49.82494228, 49.31210896, 50.43925891, 49.14609478,
       49.48432114, 49.64091001, 49.80938463, 49.40106338, 50.60119576,
       49.18013321, 50.35216409, 49.58221111, 49.18411654, 49.51898529,
       49.93510963, 49.27203731, 50.00898639, 49.93431281, 49.59426029,
       49.54928453, 50.60154814, 50.21275714, 49.91379671, 49.63697724,
       50.12242977, 49.20322985, 49.45873442, 49.64267642, 49.47512517,
       49.90081562, 49.16162283, 50.48013064, 49.8826371 , 49.86081447,
       49.52894452, 49.27281316, 49.89186092, 49.22732863, 50.51917164,
       49.83198915, 49.10733221, 49.20126322, 50.51722078, 49.70085641,
       49.70184966, 49.83145599, 49.65314005, 49.94864757, 49.71616973,
       49.08333895, 50.07231561, 50.3145283 , 49.73642757, 49.59777306,
       49.98519059, 49.27148865, 49.0559519 , 50.54625853, 49.48165193,
       49.71350819, 50.23108907, 49.91735854, 49.28190827, 50.10434473,
       49.05687964, 49.90541453, 49.89310095, 49.98798592, 50.44156085,
       50.27588999, 49.66834281, 49.1741396 , 50.54133917, 49.66463321,
       49.64256476, 49.12671017, 49.49835456, 49.30818142, 49.4589603 ,
       49.9646032 , 49.18456592, 49.95975144, 49.6174493 , 49.51414871,
       49.62277676, 49.79169658, 50.55014407, 49.43158056, 49.12183141,
       49.61101136, 50.46996521, 50.09665585, 49.26963372, 49.1248939 ,
       49.14364136, 50.46765854, 50.22543409, 50.07350987, 49.58213031,
       50.18263346, 50.32938543, 49.15218948, 49.67093656, 49.25065287,
       50.59011147, 49.87493155, 49.63032334, 50.2694398 , 49.48694744,
       49.56848552, 49.38898861, 49.53008578, 49.97587597, 50.468579  ,
       49.85699045, 50.43316453, 50.48474469, 49.2516629 , 49.44370606,
       49.1084737 , 49.05894576, 49.19456578, 50.33717904, 50.28699632,
       49.57461141, 49.98110687, 49.48991042, 50.10899588, 50.54690668,
       50.31841824, 49.11538868, 49.52393129, 50.08793351, 49.13502857,
       49.57940401, 49.13823372, 49.5461163 , 50.08608578, 50.26462213,
       49.40734592, 49.23631306, 49.29802228, 49.3502171 , 49.02836022,
       50.32101551, 50.57578916, 49.2471209 , 49.46249362, 49.17607957,
       49.74505043, 49.61694482, 49.54411413, 50.277952  , 50.48617628,
       50.26995959, 49.60573128, 49.88855372, 49.98369553, 49.23468312,
       49.94825745, 49.25583409, 50.12787071, 49.95328174, 49.94345443,
       50.01170201, 50.55902247, 49.75822133, 50.23226385, 49.73151786])
```

```python
# Evaluation metrics
print("Intercept:", mlr.intercept_)
print("Coefficients:", dict(zip(features, mlr.coef_)))
print("R-squared:", r2_score(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Intercept: 49.015922791748174
Coefficients: {'Conversion_Rate': 0.015883883981316252, 'Purchase': 0.02683512314669126}
R-squared: -0.0021528539391071355
RMSE: 29.257751131650785
```

```
sns.histplot(data["Conversion_Rate"],color="red")
plt.title("Before prediction Distribution of Conversion_Rate")
plt.xlabel("Conversion_Rate")
plt.ylabel("Bounce_Rate")
plt.tight_layout()
plt.show()

sns.histplot(y_pred,color="pink")
plt.title("After prediction Distribution of Conversion_Rate")
plt.xlabel("Conversion_Rate")
plt.ylabel("Bounce_Rate")
plt.tight_layout()
plt.show()
```





📑 Hypothesis Testing

We perform a T-test to see if there is a significant difference in the `Session_Duration_Minutes`of Users from the North vs South region.

```
# 📌 11. Hypothesis Testing

# Choose two regions
region1 = data[data['Region'] == 'North']['Session_Duration_Minutes']
region2 = data[data['Region'] == 'South']['Session_Duration_Minutes']

# Perform independent two-sample t-test
t_stat, p_value = ttest_ind(region1, region2)

print("T-Statistic:", t_stat)
print("P-Value:", p_value)

# Interpretation
alpha = 0.05
if p_value < alpha:
    print("Reject Null Hypothesis: There is a significant difference in
Average Order Value between North and South regions.")
else:
    print("Fail to Reject Null Hypothesis: No significant difference found.")
```

```
T-Statistic: nan
P-Value: nan
Fail to Reject Null Hypothesis: No significant difference found.
```

📏 Model Evaluation

We evaluated the models using the following techniques:

- **R_squared**: Measures how well the regression model explains the variation in the dependent variable.
- **P-values**: Determines the statistical significance of each predictor.

📗 Interpretation:

- The **Simple and Multiple Linear Regerssion Models** revealed how `Conversion_Rate` , `Purchase` and `Session_Duration_Minutes` impact `Bounce_Rate`.

📌 Final Conclusion

In this project, **"E-commerce User Behavior Analysis using Statistical and Predictive Techniques"** We explored a comprehensive dataset using a core statistical techniques to extract meaningful business insights. Here's a summary of our findings:

📍 Key Takeaways:

- **Descriptive statistics** Help us to understand the dataset using basic statistical measures which includes central tendencies and variation in purchase patterns and user behaviors and majority of actions fewer purchases. This data set provides valuable insights into user browsing patterns which can be used for further statistical modeling and business analysis.it includes a variety of product categories such as electronics, applianceS, computers with brands like Samsung and Apple appearing frequently.
- **Probability distributions** (Normal, binomial) Were explored on selected columns like `Session_Duration_Minutes` and `Conversion_Rate`, revealing Distribution tendencies useful for modeling. Race distribution are essential for selecting the correct statistical models and tests, understanding customer behavior and design effective marketing strategies.
- **Correlation analysis** Indicated strong relationship, expatriate between the `Conversion_Rate` and `Bonuce_Rate` can you please sales performance which is further valuated using regression.
- **Simple and Multiple Linear Regression** Modest demonstrated that `Conversion_Rate`, `Purcahse`significantly contribute to predicting users `Bonus_Rate`, With respectable R-squared values.
- **Hypothesis testing** helped validate assumptions, ensuring the statistical robustness of our Conclusions.

📈 Business Interpretation:

- Customer journey insights- The majority of users only view products without progressing to purchase. there's a significant drop off between views and purchases which eyelets potential conversion funnel leakage. Tracking user flow from due to purchase can help identify where users lose interest
- Product Performance- a small number of products received the highest views and purchase which confirming the 80/20 around 20% of products drive 80% of engagement. Price has moderate positive correlation with purchase right, indicating that users are willing to Pay more for certain brands or categories like Electronics and smartphones.
- Brand and Category Impact- Certain brands in categories dominate user interest for example smartphones, laptops kitchen appliances. This high engagement categories are ideal targets for future listenings, promotions are bundled details

🎯 Final Verdict

This analysis provides a strong statistical foundational **User_Behavioral**, **Product_Viewed**,**Purchases** and **Bonus_Rate**. This project bridges the cap between raw data and actionable business intelligence This statistical analysis uncovers key behavior-Price relationships:

- Simple behavior metrics have predictive value but aren't sufficient alone
- Combining multiple interaction signals improves business predictions
- the business should invest in behavior analytics, conversion tracking and advanced models

References:

- **Imran Ali Shah**. (2023). *Sales and Customer Insights Dataset*. Kaggle. Available at: https://www.kaggle.com/datasets/ashishsharma150102/behavior-data-for-engagement-and-conversion-metrics

- Pandas Documentation – Data preprocessing, handling missing values, and exploratory data analysis. https://pandas.pydata.org/docs/

- **Virtanen, P., Gommers, R., Oliphant, T. E., et al.** (2020). *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. Nature Methods, 17, 261–272. (Used for hypothesis testing, probability distributions)

- **McKinney, W.** (2010). *Data Structures for Statistical Computing in Python*. In Proceedings of the 9th Python in Science Conference, 51–56. (Used for structured data handling with Pandas)

- **Waskom, M. L.** (2021). *Seaborn: Statistical Data Visualization*. Journal of Open-Source Software, 6(60), 3021. (Used for histogram and KDE plots in probability analysis)

- **Pedregosa, F., Varoquaux, G., Gramfort, A., et al.** (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. (Used for linear regression and classification algorithms)