Import modules:

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

- ***pandas*** *- used to perform data manipulation and analysis.*
- ***NumPy*** *- used to perform a wide variety of mathematical operations on arrays.*
- ***matplotlib*** *- used for data visualization and graphical plotting.*
- ***seaborn*** *- built on top of matplotlib with similar functionalities.*

Dataset Information

This dataset comprises of sales transactions captured at a retail store. It's a classic dataset to explore and expand your feature engineering skills and day to day understanding from multiple shopping experiences. This is a regression problem. The dataset has 550,069 rows and 12 columns.

2.characteristics of this dataset.

Loading the dataset:

```
bf = pd.read_csv(r"D:\Python\blackfriday.csv")
bf
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Pr
0	1000001	P00069042	F	0-17	10	A	2	0	3	NaN	
1	1000001	P00248942	F	0-17	10	A	2	0	1	6.0	
2	1000001	P00087842	F	0-17	10	A	2	0	12	NaN	
3	1000001	P00085442	F	0-17	10	A	2	0	12	14.0	
4	1000002	P00285442	M	55+	16	C	4+	0	8	NaN	
...	...	...	...	...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	1	1	20	NaN	
550064	1006035	P00375436	F	26-35	1	C	3	0	20	NaN	
550065	1006036	P00375436	F	26-35	15	B	4+	1	20	NaN	
550066	1006038	P00375436	F	55+	1	C	2	0	20	NaN	
550067	1006039	P00371644	F	46-50	0	B	4+	1	20	NaN	

550068 rows × 12 columns

Let us see the statistical information of the attributes.

```
bf.describe()
```

	User_ID	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	376430.000000	166821.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9.842329	12.668243	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5.086590	4.125338	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	2.000000	3.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5.000000	9.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	9.000000	14.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	15.000000	16.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	18.000000	18.000000	23961.000000

Let us see the data type information of the attributes.

```
bf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   User_ID                     550068 non-null  int64
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64
 8   Product_Category_1          550068 non-null  int64
 9   Product_Category_2          376430 non-null  float64
 10  Product_Category_3          166821 non-null  float64
 11  Purchase                    550068 non-null  int64
dtypes: float64(2), int64(5), object(5)
memory usage: 50.4+ MB
```

- *We have categorical as well as numerical attributes which we will process separately.*
- *Product_Category_1 data type is different from Product_Category_2 and Product_Category_3, that won't affect the process or the result.*

*Prepare data for EDA (**Exploratory Data Analysis)***

1)Change some categorical column into numerical and binary it will help us to plot the data easily.

2)Why?

*Ans: **Improved accuracy:** With large amounts of data, machine learning algorithms can learn more complex relationships between inputs and outputs, leading to improved accuracy in predictions and classifications.*

Automation: Machine learning models can automate decision-making processes and can perform repetitive tasks more efficiently and accurately than humans.

Now convert gender in numerical data:

```
bf["Gender"]=bf["Gender"].map({"F":0,"M":1})
```

```
bf["Gender"].unique()
```

Now convert age column into normal interval and distribution:

```
bf["Age"].unique()
```

```
array(['0-17', '55+', '26-35', '46-50', '51-55', '36-45', '18-25'],
      dtype=object)
```

```
bf["Age"] = bf["Age"].map({"0-17":1,"18-25":2,"26-35":3,"36-45":4,"46-50":5,"51-55":6,"55+":7})
```

```
bf["Age"].unique()
```

Now city_category change into numerical data:

```
df.groupby('City_Category').size()
```

```
City_Category
A    147720
B    231173
C    171175
dtype: int64
```

```
df["City_Category_binary"]=df["City_Category"].map({"A":0,"B":1,"C":2})
```

No, we remove all the category inside the string from Stay_In_Current_City_Years Column:

```
df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.replace('+','').astype(int)
```

```
C:\Users\Subham Ranjan\AppData\Local\Temp\ipykernel_17300\3061240698.py:1: FutureWarning: The default value of regex will chang
e from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal st
rings when regex=True.
  df['Stay_In_Current_City_Years']=df['Stay_In_Current_City_Years'].str.replace('+','').astype(int)
```

```
df["Stay_In_Current_City_Years"].unique()
```

```
array([2, 4, 3, 1, 0])
```

```
df["Stay_In_Current_City_Years"].info()
```

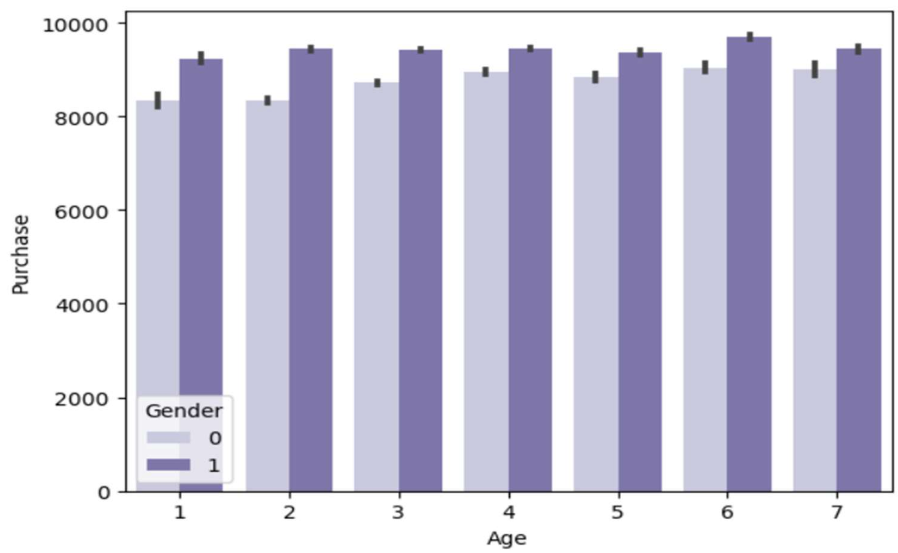
```
<class 'pandas.core.series.Series'>
RangeIndex: 550068 entries, 0 to 550067
Series name: Stay_In_Current_City_Years
Non-Null Count   Dtype
--------------   -----
550068 non-null  int32
dtypes: int32(1)
memory usage: 2.1 MB
```

Interpretation about your findings.

1)Now which gender having more purchasing power according to their age:

```
sns.barplot(x = "Age",y = "Purchase",hue = "Gender",data = df,palette = "Purples")
```

```
<Axes: xlabel='Age', ylabel='Purchase'>
```



- *This is the uniform distribution.*
- *According to this analysis Men have more purchasing power than women.*
- *May be the situation is married women don't pay their own money. Her's expenses fulfilled by their husbands.*

2)No of married and unmarried person according to gender?

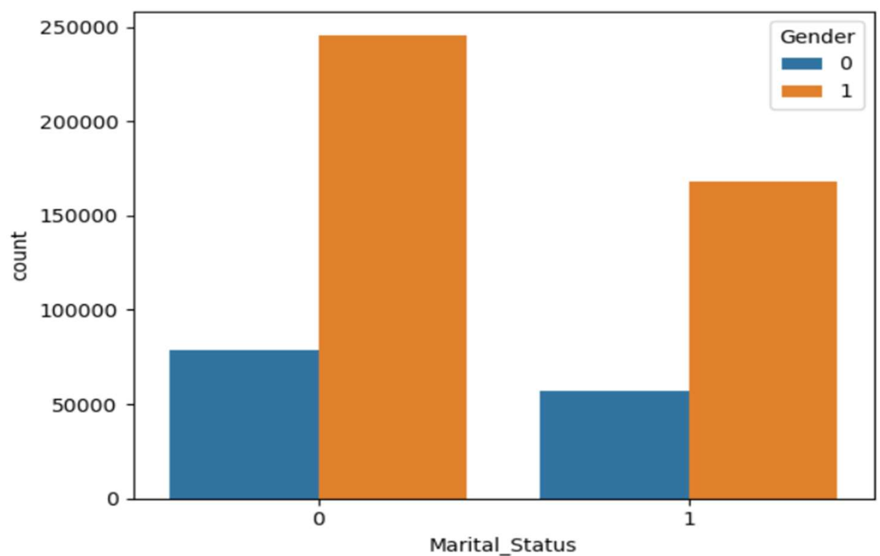
Married = 225337
Unmarried = 324731

```
df.groupby('Marital_Status').size()
```

```
Marital_Status
0    324731
1    225337
dtype: int64
```

```
sns.countplot(x = "Marital_Status",hue = "Gender",data = df)#No of Unmarried are people are 324731
```

```
<Axes: xlabel='Marital_Status', ylabel='count'>
```



- *With the help of this Analysis sellers can make marketing strategies according to gender.*
- *In this data no of married men is higher than women.*

3) Which age category has highest no of buyers?

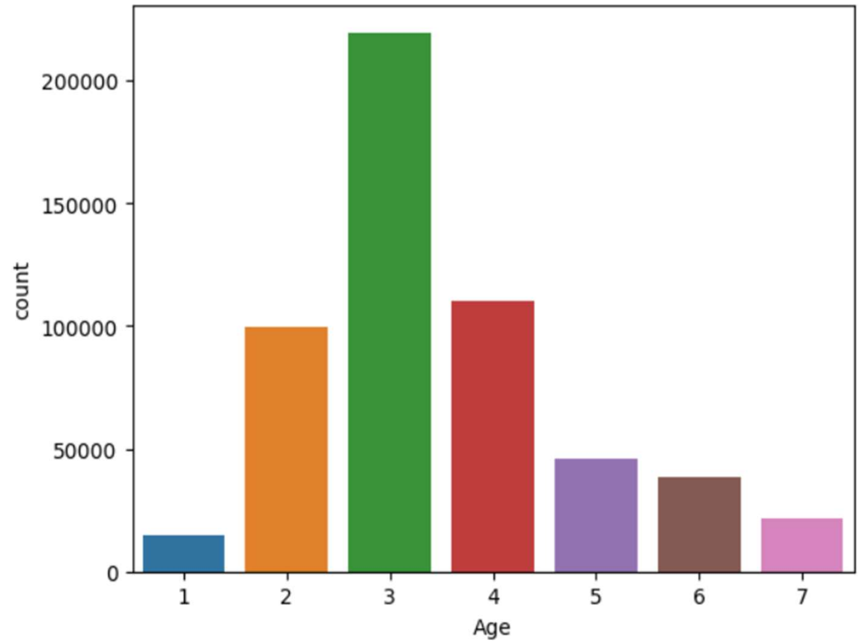
Age:

- ***# "0-17":1,***
- ***"18-25":2,***
- ***"26-35":3,***
- ***"36-45":4,***
- ***"46-50":5,***
- ***"51-55":6,***
- ***"55+":7***

 # According to this analysis no of young people who lie between 26-35 year age in large no.

```python
sns.countplot(x = "Age",data = bf)#26-35 age category people has highest no buyers
```

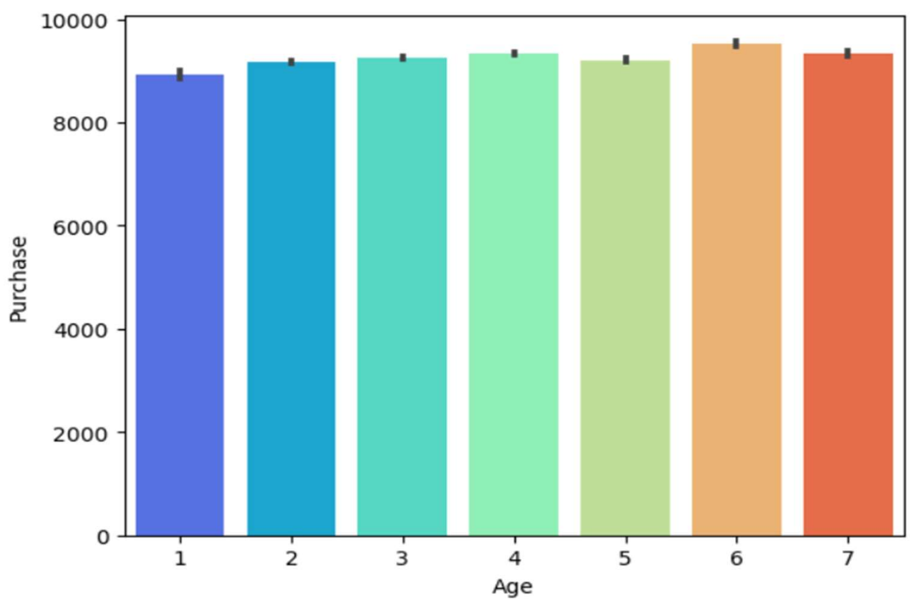
<Axes: xlabel='Age', ylabel='count'>



4 Which Age category has done highest no purchases?

```python
sns.barplot(x  = "Age",y ="Purchase" ,data = df,palette = "rainbow")
```

<Axes: xlabel='Age', ylabel='Purchase'>

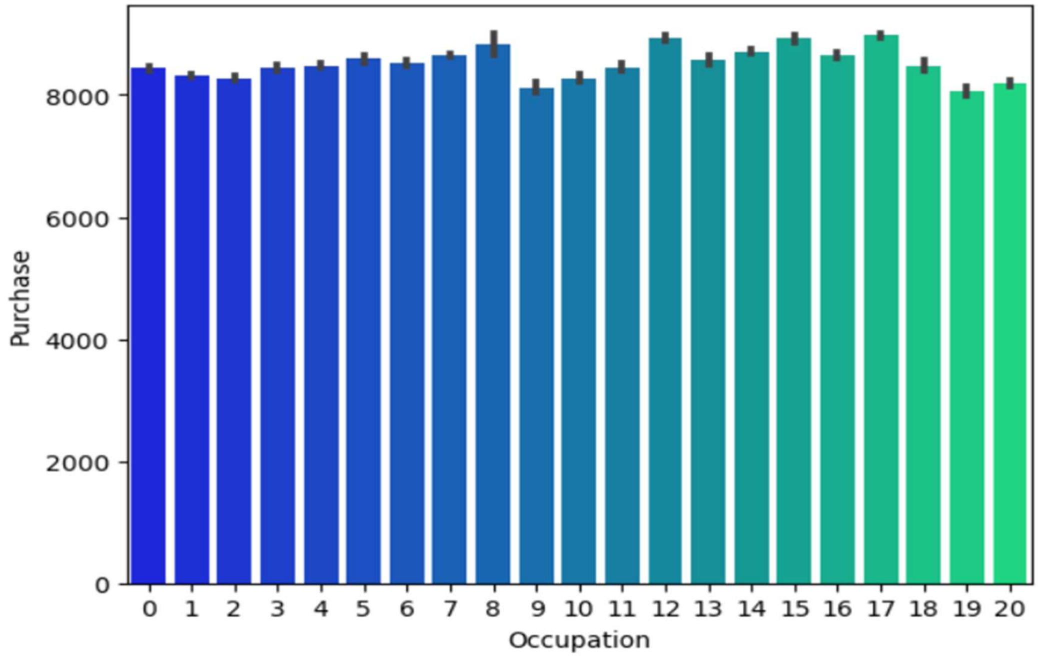


- *This the uniform distribution.*
- *According to this data 51 - 55 aged people has done highest no of purchases.*

5)Occupation and purchase analysis.

```
sns.barplot(x = "Occupation",y ="Purchase",data = df ,palette = "winter")
```

```
<Axes: xlabel='Occupation', ylabel='Purchase'>
```

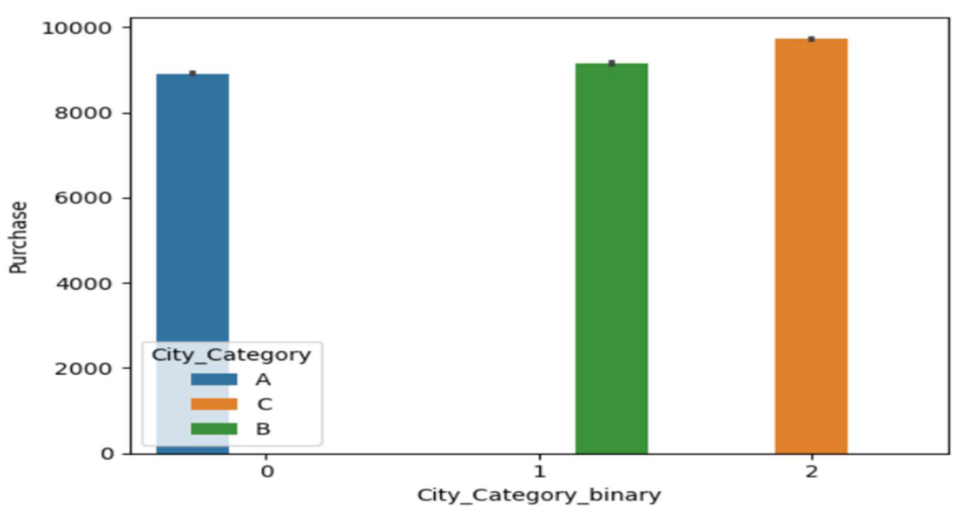


This is a uniform distribution.

6)City and Purchases analysis.

```
sns.barplot(x = 'City_Category_binary',y = 'Purchase',hue ="City_Category" ,data = df)
```

```
<Axes: xlabel='City_Category_binary', ylabel='Purchase'>
```

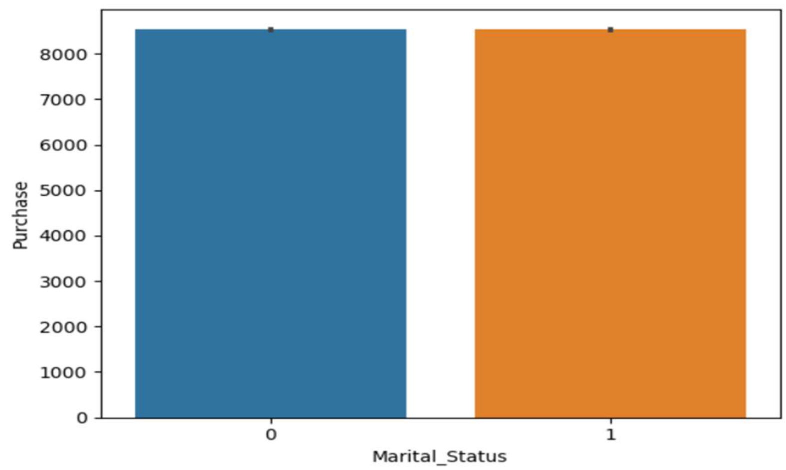


City C people purchase more than A and B

7) Marital_Status and Purchases Analysis.

```
sns.barplot(x = "Marital_Status",y = "Purchase",data = df)
```

<Axes: xlabel='Marital_Status', ylabel='Purchase'>

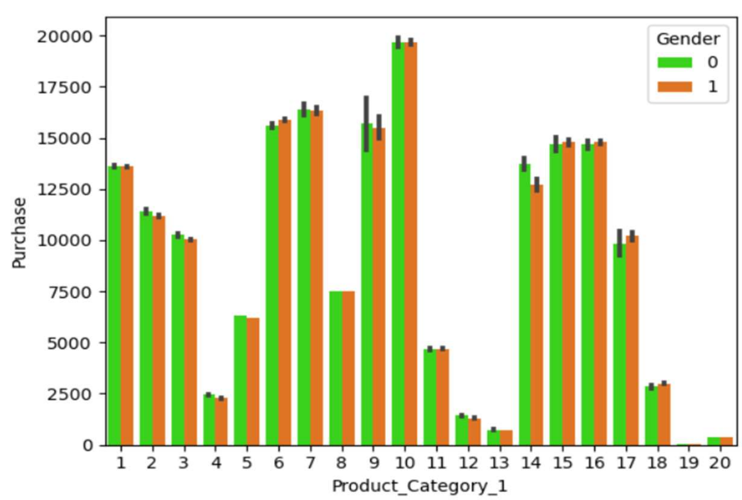


- *This is uniform distribution.*
- *According to overall analysis slightly married people has Purchased more than women.*

8)Product category_1 and Purchases Analysis

```
sns.barplot(x = "Product_Category_1",y  = "Purchase",hue = "Gender",data = df,palette = "gist_ncar")
```

<Axes: xlabel='Product_Category_1', ylabel='Purchase'>

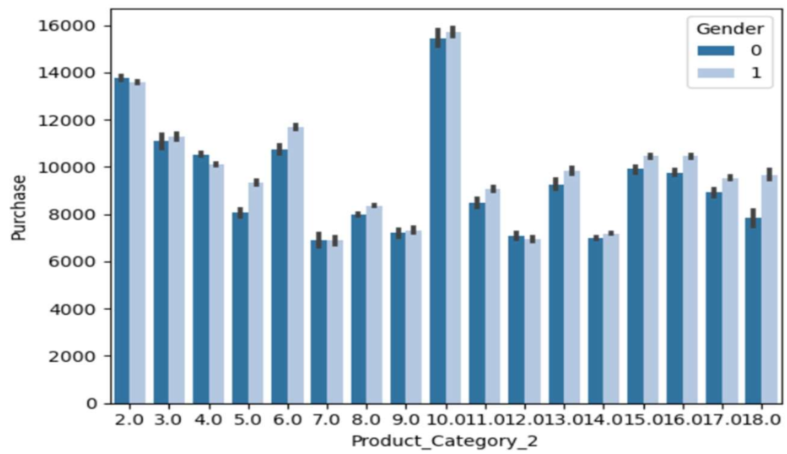


- *Category 19,20,13,12,4 product has negligible sale.*
- *People mostly purchases 9,10, 7,6 category.*

Product category_2 and Purchases Analysis

```
sns.barplot(x = "Product_Category_2",y  = "Purchase",hue = "Gender",data = df,palette = "tab20")
```

```
<Axes: xlabel='Product_Category_2', ylabel='Purchase'>
```

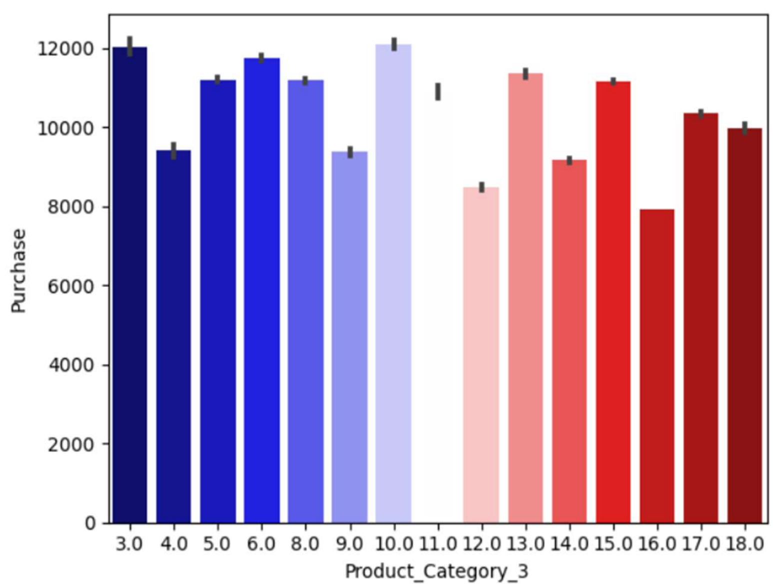


- *Category 10 has highest sale.*

Product category_3 and Purchases Analysis

```
sns.barplot(x = "Product_Category_3",y  = "Purchase",data = df,palette = "seismic")
```

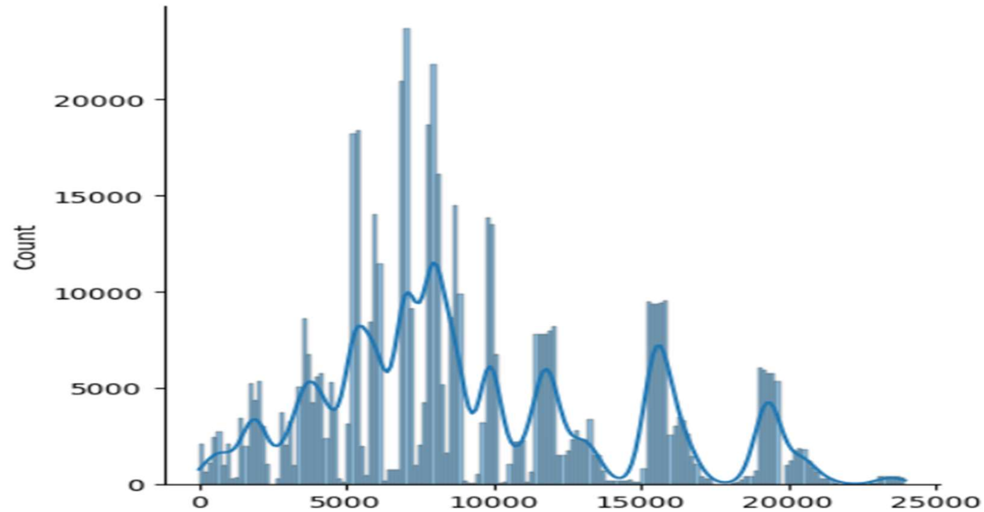
```
<Axes: xlabel='Product_Category_3', ylabel='Purchase'>
```



- *Category 10 has highest sale*

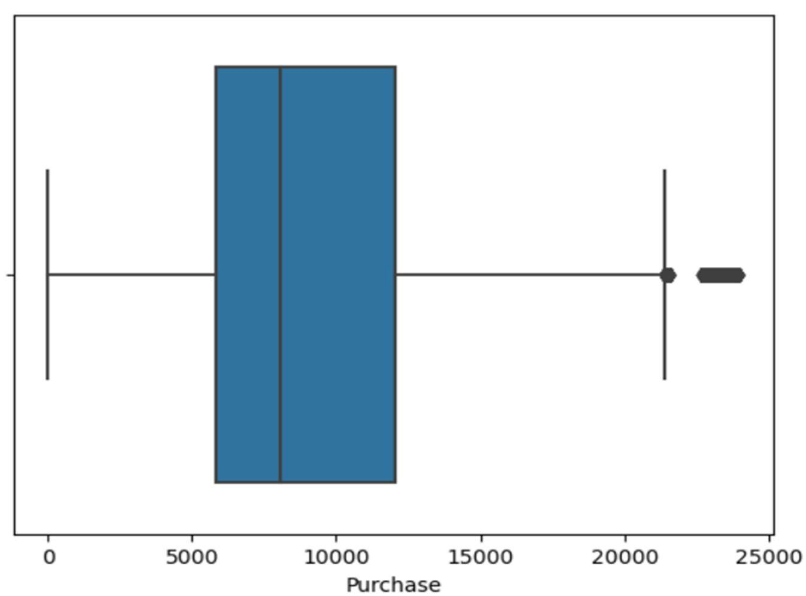
Outliers: In purchase column

```
sns.displot(x = "Purchase",data = df,kde = True)
<seaborn.axisgrid.FacetGrid at 0x2811d45cb50>
```



Purchase column has normal distribution.

```
sns.boxplot(x = "Purchase",data = df)
<Axes: xlabel='Purchase'>
```



```
IQR = df["Purchase"].quantile(0.75)-df["Purchase"].quantile(0.25)
IQR
```

```
6231.0
```

```
upperlimit = IQR + 1.5*std
upperlimit
```

```
13765.59809073094
```

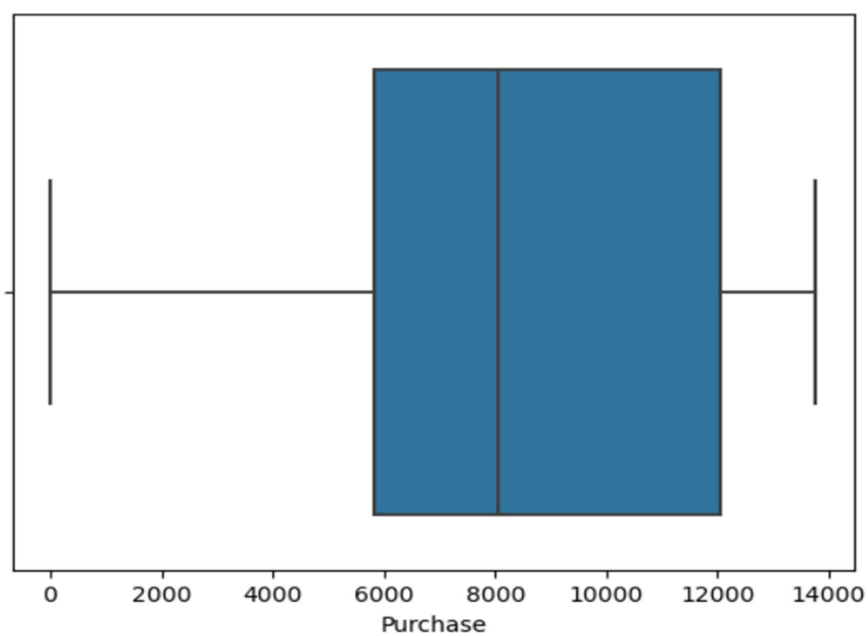
```
lowerlimit = IQR - 1.5*std
lowerlimit
```

```
-1303.5980907309404
```

```
df.loc[df["Purchase"]>13765.59809073094,"Purchase"] = 13765.59809073094
```

```
sns.boxplot(x = "Purchase",data = df)
```

```
<Axes: xlabel='Purchase'>
```



Handling null values

1)Mean: *we use mean to fill sales and numerical data*

Advantages:

- *It is a quick and computationally efficient way to handle missing values, especially in large datasets.*
- *Mean imputation is less sensitive to outliers compared to other imputation methods. Extreme values have less impact on the mean, making it a robust choice when dealing with data that may contain outliers.*
- *Mean imputation is particularly suitable for numeric data types. It is a natural choice when working with continuous variables, as it ensures that the imputed values are within the range of the observed data.*

2)median: *we use mean to fill sales and numerical data.*

- *If your data is not normally distributed, the median can be a more representative measure of central tendency than the mean. This is particularly relevant when dealing with skewed data.*
- *The median is a suitable choice for imputing missing values in time series data, especially when there is a need to preserve the temporal characteristics of the series.*
- *In the presence of extreme values, the median is less affected, providing a more stable estimate and reducing the impact of outliers on imputed values.*

3)Mode: *Replacing with high frequency value(categorical).*

- *The mode is particularly useful when dealing with categorical data, where using the mean or median may not be meaningful. For example, filling missing values in a column representing car colours with the mode colour makes more sense than using the mean or median.*

4)sampling: *we fill small no of null values with sampling and Random values from the dataset.*

- *Sampling helps in preserving relationships between variables. If the missing values are not completely random, using sampling can help maintain the correlations and dependencies between different features.*

5)Frontward filling and Backward filling: *fill null with preceding values and front value.*

Frontward Fill(ffill):

- *Ideal for time series data where missing values can be filled with the last known value.*
- *Suitable for datasets with a logical sequence, where the next value is expected to be like the previous one.*

Backward Fill(bfill):

- *Can be beneficial for scenarios where future values are influenced by past data, making backward fill suitable for forecasting or predictive modelling.*
- *Appropriate when missing values are assumed to be closer to the subsequent values rather than the previous ones.*

6)Capturing null values in a new feature (numerical and categorical): *we fill 0 and -1 when null values are need for particular column*

7)Replacing with value which is at the end/beg of the distribution (Numerical).