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

data=pd.read_csv('/content/Superstore.csv', encoding='latin-1')

data
```

```
{"type":"dataframe","variable_name":"data"}
```

```
Warning: Total number of columns (21) exceeds max_columns (20)
limiting to first (20) columns.
```

```
data.head(5)
```

```
{"type":"dataframe","variable_name":"data"}
```

```
Warning: Total number of columns (21) exceeds max_columns (20)
limiting to first (20) columns.
```

Tail()-Tail Methode Retreive Last Any Number Of Rows

```
data.tail(1)
```

```
{"type":"dataframe"}
```

Desribe () -Methode to Show Statstics Value This Methode Only Applicable For Numeric Columns

```
data.describe()
```

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 8,\n  \"fields\": [\
n    {\n      \"column\": \"Row ID\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 3601.5811575098865,\n
\"min\": 1.0,\n        \"max\": 9994.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          9994.0,\n
4997.5,\n          7495.75\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Postal Code\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 35860.31406157157,\n
\"min\": 1040.0,\n        \"max\": 99301.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
55190.3794276566,\n          56430.5,\n          9994.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sales\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 8197.010918685499,\n
\"min\": 0.444,\n        \"max\": 22638.48,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
```

```
229.85800083049833,\n          54.489999999999995,\n          9994.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Quantity\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3531.848471644344,\n        \"min\": 1.0,\n        \"max\": 9994.0,\n
\"num_unique_values\": 8,\n        \"samples\": [\n
3.789573744246548,\n          3.0,\n          9994.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Discount\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
3533.3336684667293,\n        \"min\": 0.0,\n        \"max\": 9994.0,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          9994.0,\n
0.15620272163297977,\n          0.8\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Profit\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
5288.326642672474,\n        \"min\": -6599.978,\n        \"max\":
9994.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
28.65689630778467,\n          8.6665,\n          9994.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


Data.Shape() - Return How Many Rows And Columns Present In Our Datasets

```
data.shape
```

```
(9994, 21)
```

Data.info() - Methode Return How Many Columns Are Presennt In Our Datasets With Data Types

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         9994 non-null   int64
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   object
 3   Ship Date      9994 non-null   object
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
```

```
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64
 12  Region         9994 non-null   object
 13  Product ID     9994 non-null   object
 14  Category       9994 non-null   object
 15  Sub-Category   9994 non-null   object
 16  Product Name   9994 non-null   object
 17  Sales          9994 non-null   float64
 18  Quantity       9994 non-null   int64
 19  Discount       9994 non-null   float64
 20  Profit         9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

Data.columns - Return All Columns Name

```
data.columns
```

```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City',
'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-
Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

Data.dtypes - Returnn Data Types With Specific Columns

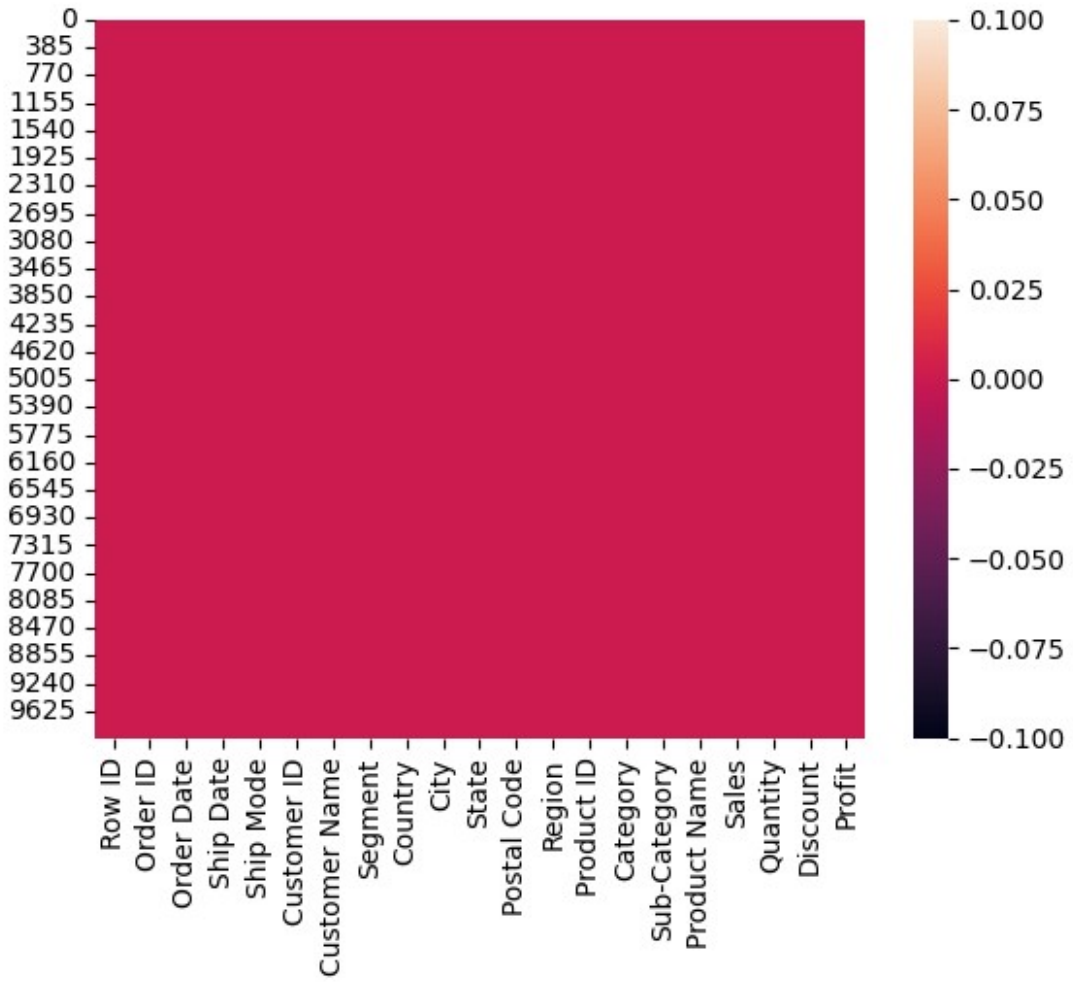
```
data.dtypes
```

```
Row ID             int64
Order ID          object
Order Date        object
Ship Date         object
Ship Mode         object
Customer ID       object
Customer Name     object
Segment           object
Country           object
City              object
State             object
Postal Code        int64
Region            object
Product ID        object
Category          object
Sub-Category      object
Product Name      object
Sales            float64
```

```
Quantity           int64
Discount         float64
Profit           float64
dtype: object
```

Find Out How Many Missing Content Present In Our Dataset And How To Handle It

```
data.isnull()

## False Denoted By Not Null Values
## True Denoteed By Null Values

{"type":"dataframe"}

Warning: Total number of columns (21) exceeds max_columns (20)
limiting to first (20) columns.

data.isnull().sum()

## Calculate Null Values With Specific Columns Wise
## In Our Datasets No Null Values Present

Row ID           0
Order ID         0
Order Date       0
Ship Date        0
Ship Mode        0
Customer ID      0
Customer Name    0
Segment          0
Country          0
City             0
State            0
Postal Code      0
Region           0
Product ID       0
Category         0
Sub-Category     0
Product Name     0
Sales            0
Quantity         0
Discount         0
Profit           0
dtype: int64

sns.heatmap(data.isnull())
plt.show()
```



Data Cleaning

```
# First Of All Check Data Types
data.dtypes
```

```
Row ID             int64
Order ID          object
Order Date        object
Ship Date         object
Ship Mode         object
Customer ID       object
Customer Name     object
Segment           object
Country           object
City              object
State             object
Postal Code        int64
Region            object
Product ID        object
Category          object
```

```
Sub-Category      object
Product Name      object
Sales            float64
Quantity           int64
Discount         float64
Profit           float64
dtype: object
```

```
data['Order ID'].head(2)
```

```
0    CA-2013-152156
1    CA-2013-152156
Name: Order ID, dtype: object
```

```
#First Of All Remove Hypen
data['Order ID']=data['Order ID'].str.replace('-','')
```

```
data['Order ID'].head(2)
```

```
0    CA2013152156
1    CA2013152156
Name: Order ID, dtype: object
```

```
# Change Data Types
# First extract only numeric part
data['Order ID'] = data['Order ID'].str.extract('(\d+)')
data['Order ID'] = pd.to_numeric(data['Order ID'],
errors='coerce').astype('Int64') # Safe conversion with NaN handling
```

```
data["Order Date"].head(3)
```

```
0    09-11-2013
1    09-11-2013
2    13-06-2013
Name: Order Date, dtype: object
```

```
## Change Data Types
data['Order Date'] = pd.to_datetime(data['Order Date'], format='%d-%m-
%Y', errors='coerce')
# Specify format and handle errors
```

```
## Change Data Types
data['Ship Date'] = pd.to_datetime(data['Ship Date'], format='%d-%m-
%Y', errors='coerce')
# Specify format and handle errors
```

```
data['Product ID'].head(3)
```

```
0    FUR-BO-10001798
1    FUR-CH-10000454
2    OFF-LA-10000240
Name: Product ID, dtype: object
```

```
#First Of All Remove Hypen
data['Product ID']=data['Product ID'].str.replace('-','')

# Change Data Types
data['Product ID'] = pd.to_numeric(data['Order ID'],
errors='coerce').astype('Int64') # Safe conversion with NaN handling

data.dtypes
```

```
Row ID                     int64
Order ID                   Int64
Order Date        datetime64[ns]
Ship Date         datetime64[ns]
Ship Mode                 object
Customer ID               object
Customer Name             object
Segment                   object
Country                   object
City                      object
State                     object
Postal Code                int64
Region                    object
Product ID                 Int64
Category                  object
Sub-Category              object
Product Name              object
Sales                    float64
Quantity                   int64
Discount                 float64
Profit                   float64
dtype: object
```

```
data.head(2)
```

{"type":"dataframe","variable_name":"data"}

```
# Remove Hypen From Customer ID
data['Customer ID']=data['Customer ID'].str.replace('-','')
```

Perform Data Anlysis And Find Business Problem Find Best Insight Who Helpfull To Increase The Profit Any Company

First Of Calculate How Many Ordered Placed From Superstore

Total 9994 Ordered Placed from superstore

```
total_orders = data['Order ID'].count()
print(f" Total Orders: {total_orders:,}")
```

```
 Total Orders: 9,994
```

2 . Total Ordered Category Wise

```
data.groupby('Category')['Order
ID'].count().sort_values(ascending=False)
orders_by_category = data.groupby('Category')['Order
ID'].count().sort_values(ascending=False)

print(" Orders by Category:\n")
for category, count in orders_by_category.items():
    print(f" {category}: {count:,} orders")
```

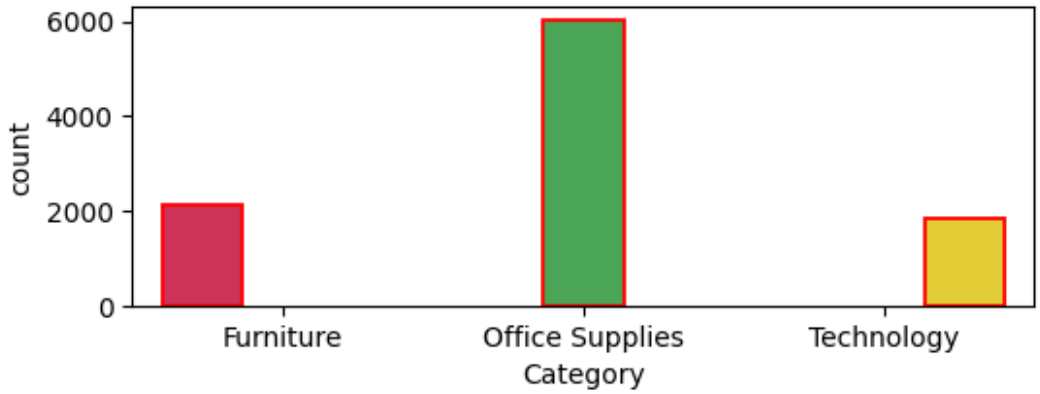
```
 Orders by Category:

 Office Supplies: 6,026 orders
 Furniture: 2,121 orders
 Technology: 1,847 orders
```

```
plt.figure(figsize=(6,2))
custom_colors = ['#e6194b', '#3cb44b', '#ffe119']
sns.countplot(data=data,
x='Category',color="blue",edgecolor='red',linewidth=1.3
,dodge=True,palette=custom_colors)
plt.show()
```

```
<ipython-input-82-7f2c54867435>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
```

`legend=False` for the same effect.

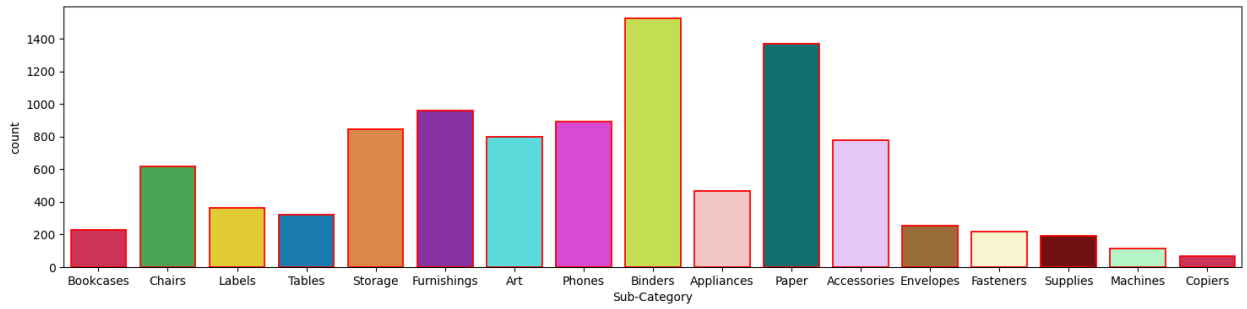
```
  sns.countplot(data=data,
x='Category',color="blue",edgecolor='red',linewidth=1.3
```




```
# Grouping and sorting
orders_by_sub_category = (
    data.groupby('Sub-Category')['Order ID']
    .count()
    .sort_values(ascending=False)
)

# Printing with emoji and formatting
print("□ Orders by Sub-Category:\n")
for sub_category, count in orders_by_sub_category.items():
    print(f"□ {sub_category}: {count:,} orders")
```

```
□ Orders by Sub-Category:

□ Binders: 1,523 orders
□ Paper: 1,370 orders
□ Furnishings: 957 orders
□ Phones: 889 orders
□ Storage: 846 orders
□ Art: 796 orders
□ Accessories: 775 orders
□ Chairs: 617 orders
□ Appliances: 466 orders
□ Labels: 364 orders
□ Tables: 319 orders
□ Envelopes: 254 orders
□ Bookcases: 228 orders
□ Fasteners: 217 orders
□ Supplies: 190 orders
```

```
 Machines: 115 orders
 Copiers: 68 orders

plt.figure(figsize=(18,4))
# Custom colors for each bar
custom_colors = [ '#e6194b', '#3cb44b', '#ffe119', '#0082c8',
    '#f58231', '#911eb4', '#46f0f0', '#f032e6',
    '#d2f53c', '#fabebe', '#008080', '#e6beff',
    '#aa6e28', '#fffac8', '#800000', '#aaffc3']

sns.countplot(data=data, x='Sub-
Category',color="blue",edgecolor='red',linewidth=1.3,palette=custom_co
lors)
plt.show()


<ipython-input-79-928dfc17b1a2>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(data=data, x='Sub-
Category',color="blue",edgecolor='red',linewidth=1.3,palette=custom_co
lors)
<ipython-input-79-928dfc17b1a2>:7: UserWarning:
The palette list has fewer values (16) than needed (17) and will
cycle, which may produce an uninterpretable plot.
  sns.countplot(data=data, x='Sub-
Category',color="blue",edgecolor='red',linewidth=1.3,palette=custom_co
lors)
```



Deal With Total Sales Category Wise And Sub Categorywise this is more importnnat for any e-commerce comapny paltform

Calculate Total Sales

```
total_sales = data['Sales'].sum()
print(f" Total Sales: ₹{total_sales:,.2f}")
```

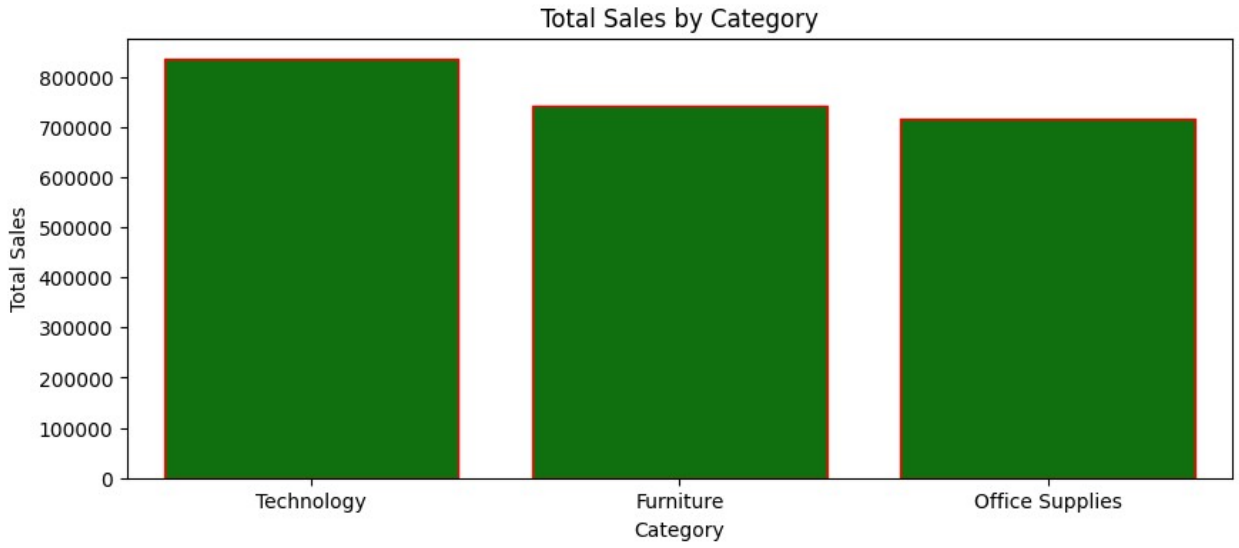
```
 Total Sales: ₹2,291,304.00
```

Calculate total sales category wise

```
sales_category = data.groupby('Category')
['Sales'].sum().sort_values(ascending=False)
print(" Total Sales by Category:\n")
for category, sales in sales_category.items():
    print(f" {category}: ₹{sales:,.2f}")
```

```
 Total Sales by Category:

 Technology: ₹834,815.00
 Furniture: ₹740,795.00
 Office Supplies: ₹715,694.00
```

```
plt.figure(figsize=(10,4))
sns.barplot(x=sales_category.index,
y=sales_category.values,estimator='mean',color='green',edgecolor='red'
)
plt.xlabel('Category')
plt.ylabel('Total Sales')
plt.title('Total Sales by Category')
plt.show()
```



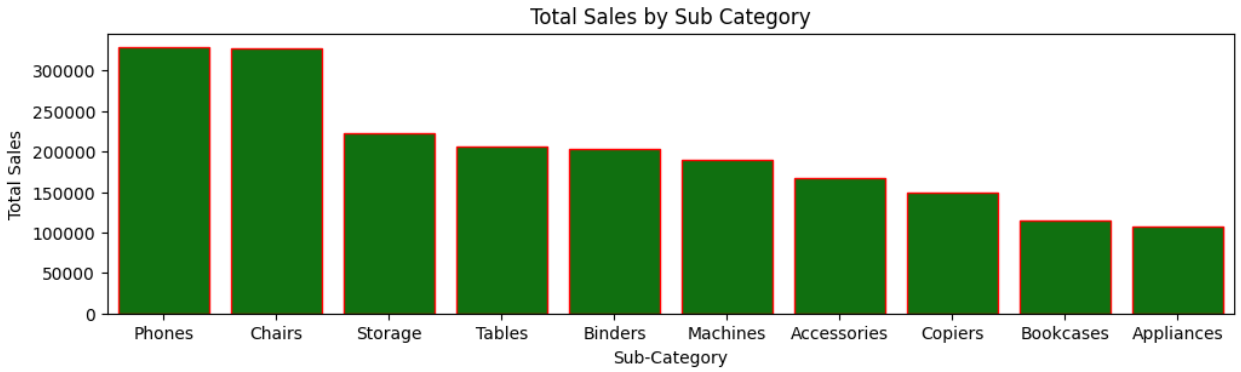
Total Sales Categorywise

```
sales_Sub_category = data.groupby('Sub-Category')
['Sales'].sum().sort_values(ascending=False)
print("□ Total Sales by Sub Category:\n")
for category, sales in sales_Sub_category.items():
    print(f"□ {category}: ₹{sales:,.2f}")
```

```
□ Total Sales by Sub Category:

□ Phones: ₹329,342.00
□ Chairs: ₹328,097.00
□ Storage: ₹223,368.00
□ Tables: ₹206,798.00
□ Binders: ₹202,593.00
□ Machines: ₹189,155.00
□ Accessories: ₹166,856.00
□ Copiers: ₹149,462.00
□ Bookcases: ₹114,728.00
□ Appliances: ₹107,266.00
□ Furnishings: ₹91,172.00
□ Paper: ₹77,675.00
□ Supplies: ₹46,573.00
□ Art: ₹26,685.00
□ Envelopes: ₹16,326.00
□ Labels: ₹12,294.00
□ Fasteners: ₹2,914.00
```

```
plt.figure(figsize=(12,3))
sns.barplot(x=sales_Sub_category.index[:10],
y=sales_Sub_category.values[:10],estimator='mean',color='green',edgeco
```

```
lor='red')
plt.xlabel('Sub-Category')
plt.ylabel('Total Sales')
plt.title('Total Sales by Sub Category')
plt.show()
```

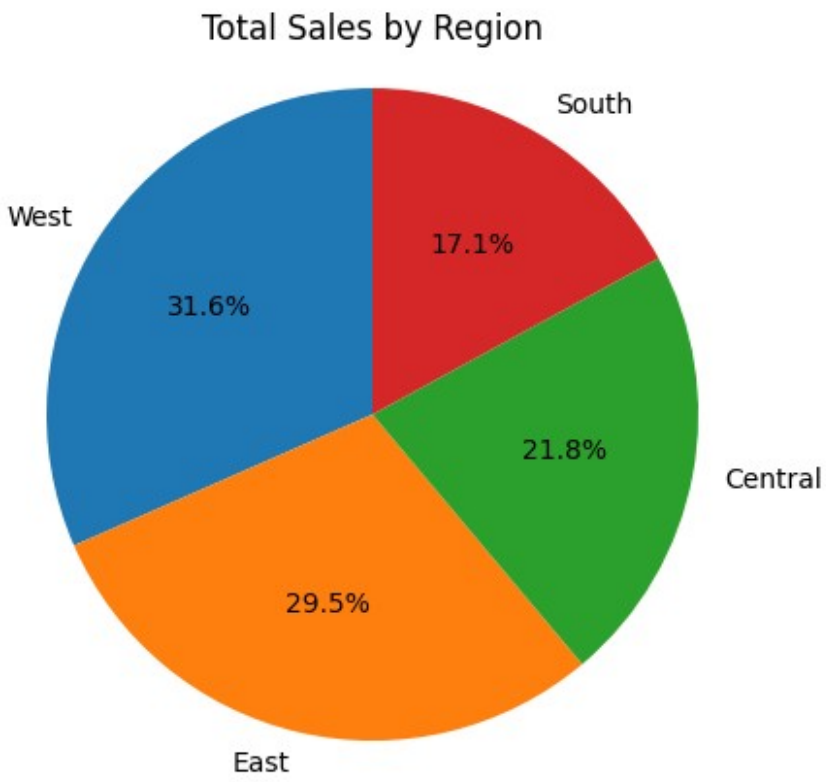


Total Sales Region Wise this is very most Insight from any company so company can be give sales offer who customer belongs from region

```
total_sales_region = data.groupby('Region')
['Sales'].sum().sort_values(ascending=False)
print(" Total Sales by Region:\n")
for region, sales in total_sales_region.items():
    print(f" {region}: ₹{sales:,.2f}")
```

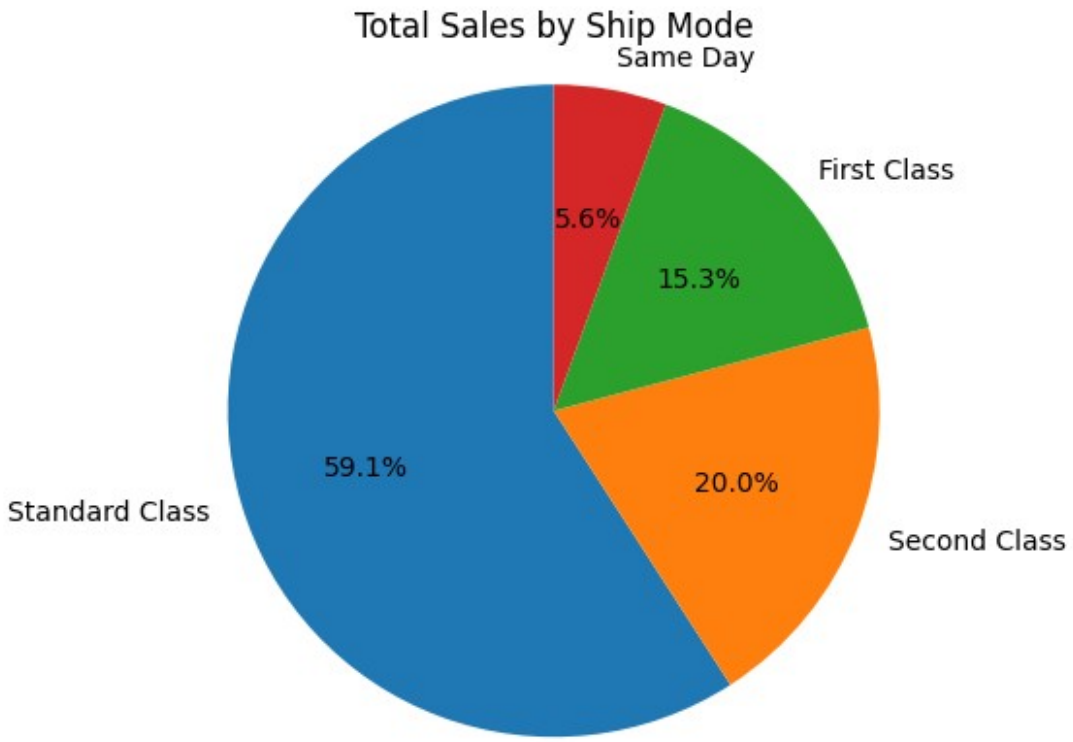
```
 Total Sales by Region:

 West: ₹723,573.00
 East: ₹677,079.00
 Central: ₹499,887.00
 South: ₹390,765.00
```

```
plt.pie(total_sales_region, labels=total_sales_region.index,
autopct='%1.1f%%', startangle=90)
plt.axis('equal')
plt.title('Total Sales by Region')
plt.show()
```



Total Sales Based On Shipmode

```
sales_ship_mode = data.groupby('Ship Mode')
['Sales'].sum().sort_values(ascending=False)
print(" Total Sales by Ship Mode:\n")
for ship_mode, sales in sales_ship_mode.items():
    print(f" {ship_mode}: ₹{sales:,.2f}")

 Total Sales by Ship Mode:

 Standard Class: ₹1,354,726.00
 Second Class: ₹458,024.00
 First Class: ₹350,505.00
 Same Day: ₹128,049.00

plt.pie(sales_ship_mode, labels=sales_ship_mode.index, autopct='%1.1f%
%', startangle=90)
plt.axis('equal')
plt.title('Total Sales by Ship Mode')
plt.show()
#
```



Total ordered Quantity

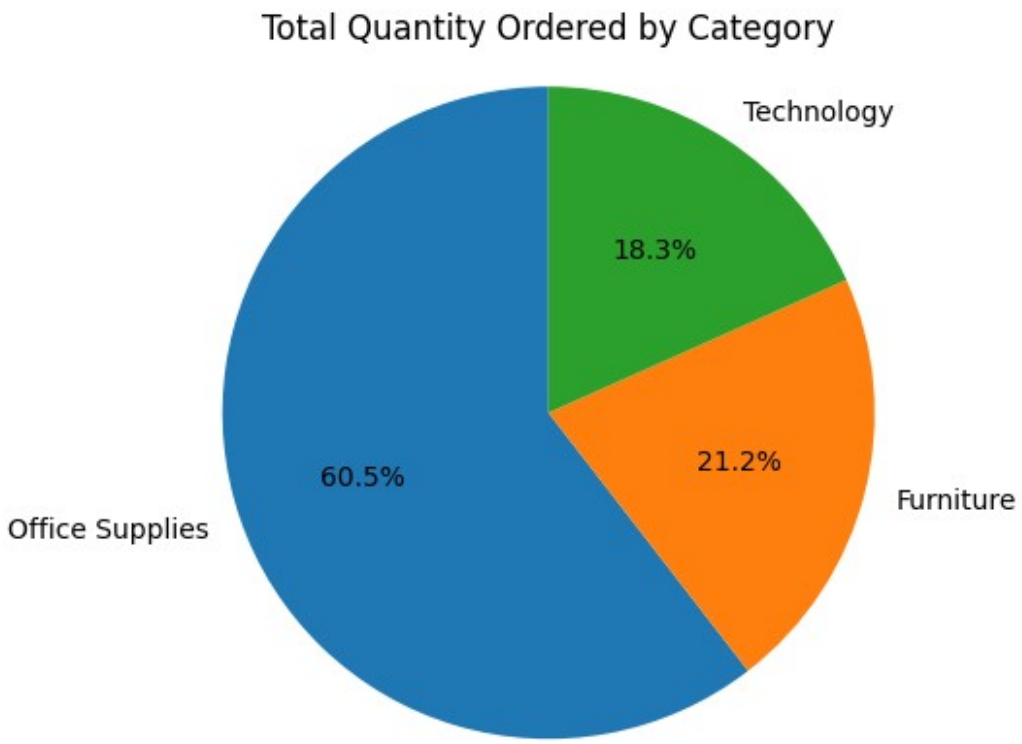
```
total_quantity = data['Quantity'].sum()
print(f" Total Quantity Ordered: {total_quantity:,}")
```

```
 Total Quantity Ordered: 37,873
```

```
# Total Quantity ordered Category-wise
total_quantity_category = data.groupby('Category')
['Quantity'].sum().sort_values(ascending=False)
print(" Total Quantity Ordered by Category:\n")
for category, quantity in total_quantity_category.items():
    print(f" {category}: {quantity:,} units")
```

```
 Total Quantity Ordered by Category:

 Office Supplies: 22,906 units
 Furniture: 8,028 units
 Technology: 6,939 units
```

```
plt.pie(total_quantity_category, labels=total_quantity_category.index,
autopct='%1.1f%%', startangle=90)
plt.axis('equal')
plt.title('Total Quantity Ordered by Category')
plt.show()
```




```
data.head(2)
```

```
{"type":"dataframe","variable_name":"data"}
```

```
# Extract Year From Order Date
data['Year'] = data['Order Date'].dt.year
data['Month'] = data['Order Date'].dt.month_name()
data['Days']=data['Order Date'].dt.day
data['ship_days']=data['Ship Date'].dt.day


# Calculate shipping time in days
data['Shipping Duration'] = (data['Ship Date'] - data['Order
Date']).dt.days
```

Average Shipping Duration

```
# Average Shipping Duration
average_shipping_duration = data['Shipping Duration'].mean().round(1)
print(f" Average Shipping Duration: {average_shipping_duration:.2f}
days")
```

```
 Average Shipping Duration: 4.00 days
```

Deal With Timestamp

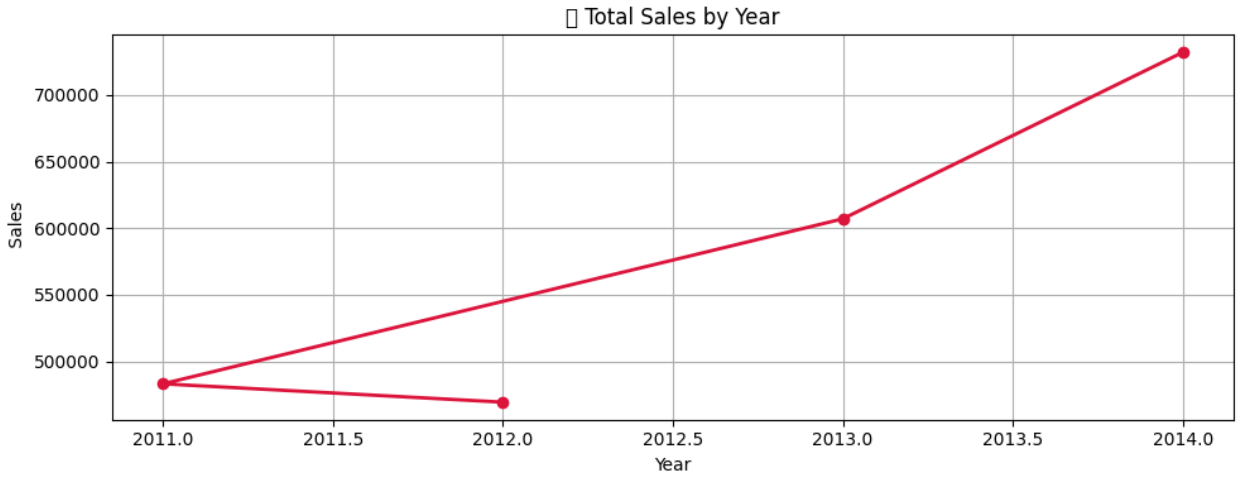
Calulate total sales yearwise

```
sales_year=data.groupby('Year')
['Sales'].sum().sort_values(ascending=False)
print("□ Total Sales by Year:\n")
for year, sales in sales_year.items():
    print(f"□ {year}: ₹{sales:,.2f}")
```

```
□ Total Sales by Year:

□ 2014: ₹731,986.00
□ 2013: ₹606,967.00
□ 2011: ₹483,063.00
□ 2012: ₹469,288.00
```

```
plt.figure(figsize=(10,4))
plt.plot(sales_year.index, sales_year.values,marker='o',
color='crimson', linewidth=2) # Plot with correct arguments
plt.xlabel('Year') # Add x-axis label
plt.title('□ Total Sales by Year')
plt.ylabel(' Sales') # Add title
plt.grid(True)
plt.tight_layout()
plt.show() # Display the plot
```

```
<ipython-input-177-ce150a662c96>:7: UserWarning: Glyph 128200 (\
N{CHART WITH UPWARDS TREND}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151
: UserWarning: Glyph 128200 (\N{CHART WITH UPWARDS TREND}) missing
from font(s) DejaVu Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```



Total Sales Monthwise

```
sales_month=data.groupby('Month')
["Sales"].sum().sort_values(ascending=False)
print("□ Total Sales by Month:\n")
for month, sales in sales_month.items():
    print(f"□ {month}: ₹{sales:,.2f}")
```

```
□ Total Sales by Month:

□ November: ₹348,247.00
□ December: ₹331,333.00
□ September: ₹308,947.00
□ March: ₹198,852.00
□ October: ₹196,632.00
□ August: ₹159,187.00
□ May: ₹155,690.00
□ July: ₹149,158.00
□ June: ₹146,676.00
□ April: ₹141,460.00
□ January: ₹95,139.00
□ February: ₹59,983.00
```

```
import matplotlib.ticker as ticker

# Assuming 'month_sales' is a DataFrame/Series with months and sales
data
plt.figure(figsize=(8, 5))
plt.plot(sales_month.index, sales_month.values, marker='o',
color='crimson', linewidth=2)

# Title and labels
plt.title('□ Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Sales')
plt.grid(True)
plt.xticks(rotation=45)
```

```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
 [Text(0, 0, 'November'),
  Text(1, 0, 'December'),
  Text(2, 0, 'September'),
  Text(3, 0, 'March'),
  Text(4, 0, 'October'),
  Text(5, 0, 'August'),
  Text(6, 0, 'May'),
  Text(7, 0, 'July'),
  Text(8, 0, 'June'),
  Text(9, 0, 'April'),
```

```
  Text(10, 0, 'January'),
  Text(11, 0, 'February')])
```


```
data.head(1)
```

{"type":"dataframe","variable_name":"data"}

Profit

Calculate Total Profit

```
data['Profit'].sum()
print(f" Total Profit: ₹{data['Profit'].sum():,.2f}")
```

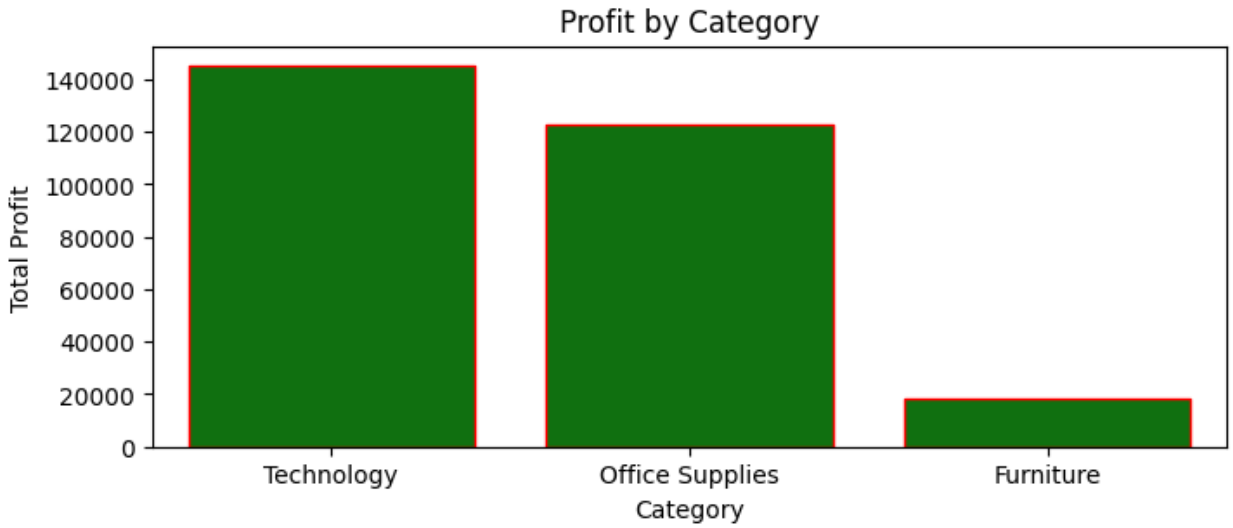
```
 Total Profit: ₹286,397.02
```

```
Profit_category = data.groupby('Category')
['Profit'].sum().sort_values(ascending=False)
print(" Total Profit by Category:\n")
```

```
for category, profit in Profit_category.items():
    print(f"□ {category}: ₹{profit:,.2f}")
```

```
□ Total Profit by Category:

□ Technology: ₹145,454.95
□ Office Supplies: ₹122,490.80
□ Furniture: ₹18,451.27
```

```
plt.figure(figsize=(8,3))
sns.barplot(x=Profit_category.index,
y=Profit_category.values,estimator='mean',color='green',edgecolor='red
')
plt.title('Profit by Category')
plt.xlabel('Category')
plt.ylabel('Total Profit')
plt.show()
```




```
year_profit=data.groupby('Year')
['Profit'].sum().sort_values(ascending=False)
print("□ Total Profit by Year:\n")
for year, profit in year_profit.items():
    print(f"□ {year}: ₹{profit:,.2f}")
```

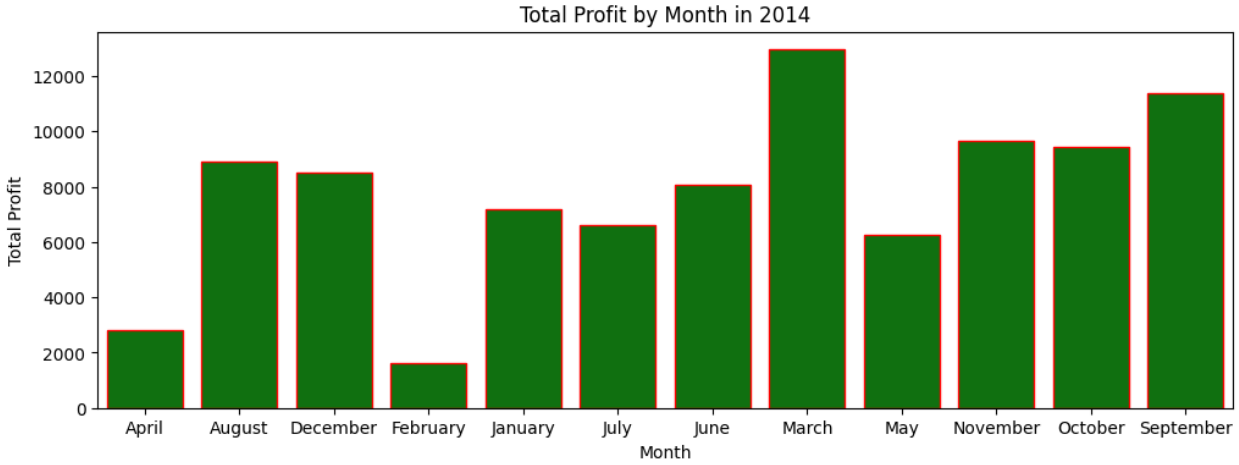
```
□ Total Profit by Year:

□ 2014: ₹93,507.51
□ 2013: ₹81,726.93
□ 2012: ₹61,618.60
□ 2011: ₹49,543.97
```

Profit Year 2014 And Monthwise

```
data_2014 = data[data['Year'] == 2014] # Create a subset for the year
2014
monthly_profit_2014 = data_2014.groupby('Month')
['Profit'].sum().sort_index()
print("□ Total Profit by Month in 2014:\n")
for month, profit in monthly_profit_2014.items():
    print(f"□ {month}: ₹{profit:,.2f}")
```

```
□ Total Profit by Month in 2014:

□ April: ₹2,803.63
□ August: ₹8,894.45
□ December: ₹8,532.87
□ February: ₹1,605.65
□ January: ₹7,208.68
□ July: ₹6,623.56
□ June: ₹8,087.67
□ March: ₹12,957.90
□ May: ₹6,274.46
□ November: ₹9,682.55
□ October: ₹9,440.66
□ September: ₹11,395.44
```

```
plt.figure(figsize=(12,4))
sns.barplot(x=monthly_profit_2014.index,
y=monthly_profit_2014.values,estimator='mean',color='green',edgecolor=
'red')
plt.xlabel('Month')
plt.ylabel('Total Profit')
plt.title('Total Profit by Month in 2014')
plt.show()
```



```
data_2013 = data[data['Year'] == 2013] # Create a subset for the year 2014
monthly_profit_2013 = data_2014.groupby('Month')
['Profit'].sum().sort_index()
print("□ Total Profit by Month in 2013:\n")
for month, profit in monthly_profit_2013.items():
    print(f"□ {month}: ₹{profit:,.2f}")
```

```
□ Total Profit by Month in 2013:

□ April: ₹2,803.63
□ August: ₹8,894.45
□ December: ₹8,532.87
□ February: ₹1,605.65
□ January: ₹7,208.68
□ July: ₹6,623.56
□ June: ₹8,087.67
□ March: ₹12,957.90
□ May: ₹6,274.46
□ November: ₹9,682.55
□ October: ₹9,440.66
□ September: ₹11,395.44
```

```
plt.figure(figsize=(12,4))
sns.barplot(x=monthly_profit_2013.index,
y=monthly_profit_2013.values,estimator='mean',color='red',edgecolor='green')
plt.xlabel('Month')
plt.ylabel('Total Profit')
plt.title('Total Profit by Month in 2013')
plt.show()
```


```
data_2012 = data[data['Year'] == 2012]
monthly_profit_2012 = data_2012.groupby('Month')
```

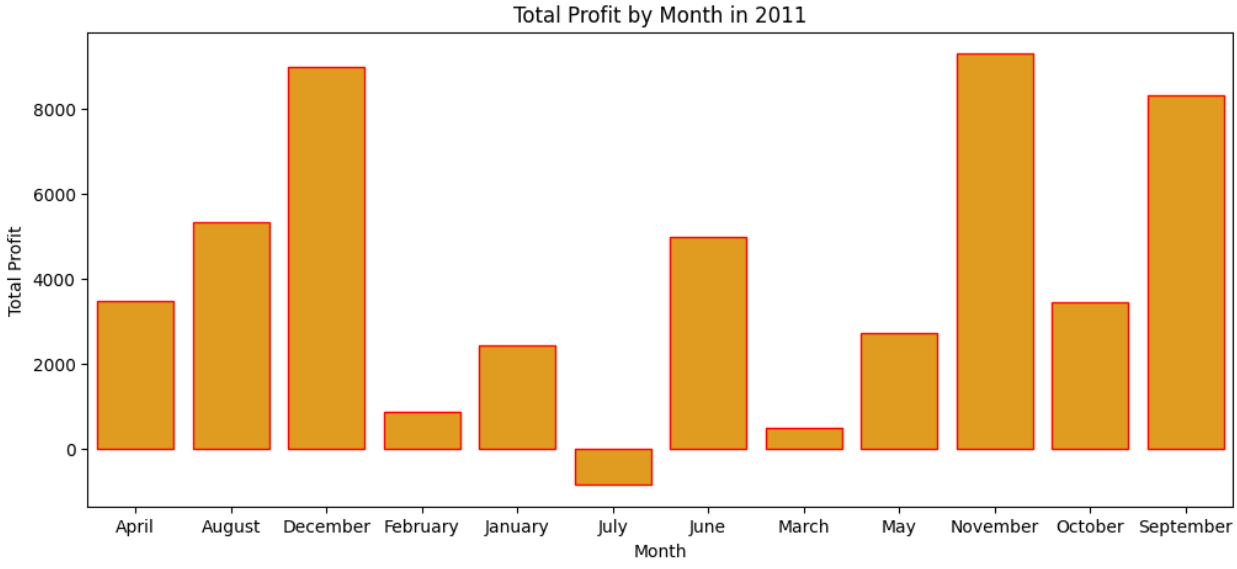
```
['Profit'].sum().sort_index()
print("□ Total Profit by Month in 2012:\n")
for month, profit in monthly_profit_2012.items():
    print(f"□ {month}: ₹{profit:,.2f}")
```

```
□ Total Profit by Month in 2012:

□ April: ₹4,187.50
□ August: ₹5,355.81
□ December: ₹8,016.97
□ February: ₹2,821.28
□ January: ₹-3,281.01
□ July: ₹3,288.65
□ June: ₹3,335.56
□ March: ₹9,724.67
□ May: ₹4,667.87
□ November: ₹12,474.79
□ October: ₹2,817.37
□ September: ₹8,209.16
```

```
plt.figure(figsize=(12,4))
sns.barplot(x=monthly_profit_2012.index,
y=monthly_profit_2012.values,estimator='mean',color='blue',edgecolor='
pink')
plt.xlabel('Month')
plt.ylabel('Total Profit')
plt.title('Total Profit by Month in 2012')
plt.show()
```


```
data_2011 = data[data['Year'] == 2011] # Create a subset for the year
2014
monthly_profit_2011 = data_2011.groupby('Month')
['Profit'].sum().sort_index()
print("□ Total Profit by Month in 2011:\n")
```

```
for month, profit in monthly_profit_2011.items():
    print(f"□ {month}: ₹{profit:,.2f}")

□ Total Profit by Month in 2011:

□ April: ₹3,488.84
□ August: ₹5,318.10
□ December: ₹8,983.57
□ February: ₹865.73
□ January: ₹2,446.77
□ July: ₹-841.48
□ June: ₹4,976.52
□ March: ₹498.73
□ May: ₹2,738.71
□ November: ₹9,292.13
□ October: ₹3,448.26
□ September: ₹8,328.10

plt.figure(figsize=(12,5))
sns.barplot(x=monthly_profit_2011.index,
y=monthly_profit_2011.values,estimator='mean',color='orange',edgecolor
='red')
plt.xlabel('Month')
plt.ylabel('Total Profit')
plt.title('Total Profit by Month in 2011')
plt.show()
```



Temporal Trends

Sales by Year: Sales show an overall increasing trend from 2011 to 2014.

Sales by Month: Sales tend to peak in November and December.

Profit by Year: Profit also generally increases over the years, but 2014 shows a slight dip compared to 2013.

Profit by Month in 2014: Profitability varies throughout the year, with losses in February and relatively lower profits in the early months.

3. Quantity Sold vs Profit

Why: High quantity doesn't always mean high profit.

Insight: Some products might be selling in bulk (e.g., binders or paper) but give little to no profit.

Action: Consider pricing strategies or bundling

```python
# Grouping the data by Sub-Category and summing Quantity and Profit
grouped_data = data.groupby('Sub-Category')[['Quantity',
'Profit']].sum()

print("□ Quantity Sold vs Profit:\n")
# Iterating over the grouped data
for sub_category, row in grouped_data.iterrows():
    print(f"{sub_category}: Quantity = {row['Quantity']}, Profit = ₹
{row['Profit']:.2f}")
```

```
□ Quantity Sold vs Profit:

Accessories: Quantity = 2976.0, Profit = ₹41936.64
Appliances: Quantity = 1729.0, Profit = ₹18138.01
Art: Quantity = 3000.0, Profit = ₹6527.79
Binders: Quantity = 5974.0, Profit = ₹30221.76
Bookcases: Quantity = 868.0, Profit = ₹-3472.56
Chairs: Quantity = 2356.0, Profit = ₹26590.17
Copiers: Quantity = 234.0, Profit = ₹55617.82
Envelopes: Quantity = 906.0, Profit = ₹6964.18
Fasteners: Quantity = 914.0, Profit = ₹949.52
Furnishings: Quantity = 3563.0, Profit = ₹13059.14
Labels: Quantity = 1400.0, Profit = ₹5546.25
Machines: Quantity = 440.0, Profit = ₹3384.76
Paper: Quantity = 5178.0, Profit = ₹34053.57
Phones: Quantity = 3289.0, Profit = ₹44515.73
Storage: Quantity = 3158.0, Profit = ₹21278.83
Supplies: Quantity = 647.0, Profit = ₹-1189.10
Tables: Quantity = 1241.0, Profit = ₹-17725.48
```

```python
data.head(1)
```

{"type":"dataframe","variable_name":"data"}

□ Conclusion: Sales Analysis

The sales analysis reveals significant patterns in customer purchasing behavior, product performance, and regional demand. Technology emerged as the top-performing category in terms of total sales, followed closely by Office Supplies, while Furniture generated comparatively lower sales.

□ Key Sales Insights:

Highest Sales were recorded in the Technology category, particularly for sub-categories like Phones and Copiers.

End-of-year months especially November and December show a sharp increase in sales, indicating a strong seasonal shopping trend.

#The West region leads in total sales, suggesting higher market potential or better customer engagement.

Consumer segment contributes the most to overall sales, highlighting it as the primary target audience.

Sales performance is positively influenced by moderate discounts but can be misleading when not accompanied by strong profits.

Sales Strategy Recommendations

Focus marketing and stock planning around high-performing months (Q4).

Invest more in Technology and Office Supplies, which show strong customer demand.

Expand efforts in high-sales regions, particularly the West, to leverage existing momentum.

Use data-driven discounting to increase sales without hurting margins.

Target the Consumer segment with tailored promotions to boost conversion.

🔍 Step-by-Step Data Analysis Process

1. 📥 Import the Dataset

Load the dataset using pandas, csv, or Excel readers.

2 . Understand The Datasets

View structure, types, and sample rows.

3. 🧹** Clean the Data**

Handle missing values, duplicates, and incorrect data types.

4. **Feature Engineering**

Create new columns for better analysis (e.g., Month, Year, Profit Margin).

5. 📊 Exploratory Data Analysis (EDA)

Use summary statistics and visualizations to uncover patterns:

Count plots

Bar/line charts

Pie charts

Heatmaps

6. □ Analyze Relationships

Correlations and patterns between features (e.g., Sales vs Profit, Discount vs Profit).

7. □ Group and Aggregate Data

Group by category, region, or time for summaries.

8. □ Draw Insights

Identify key findings (e.g., top products, low-profit areas, seasonal trends).

10. □ Report Your Analysis

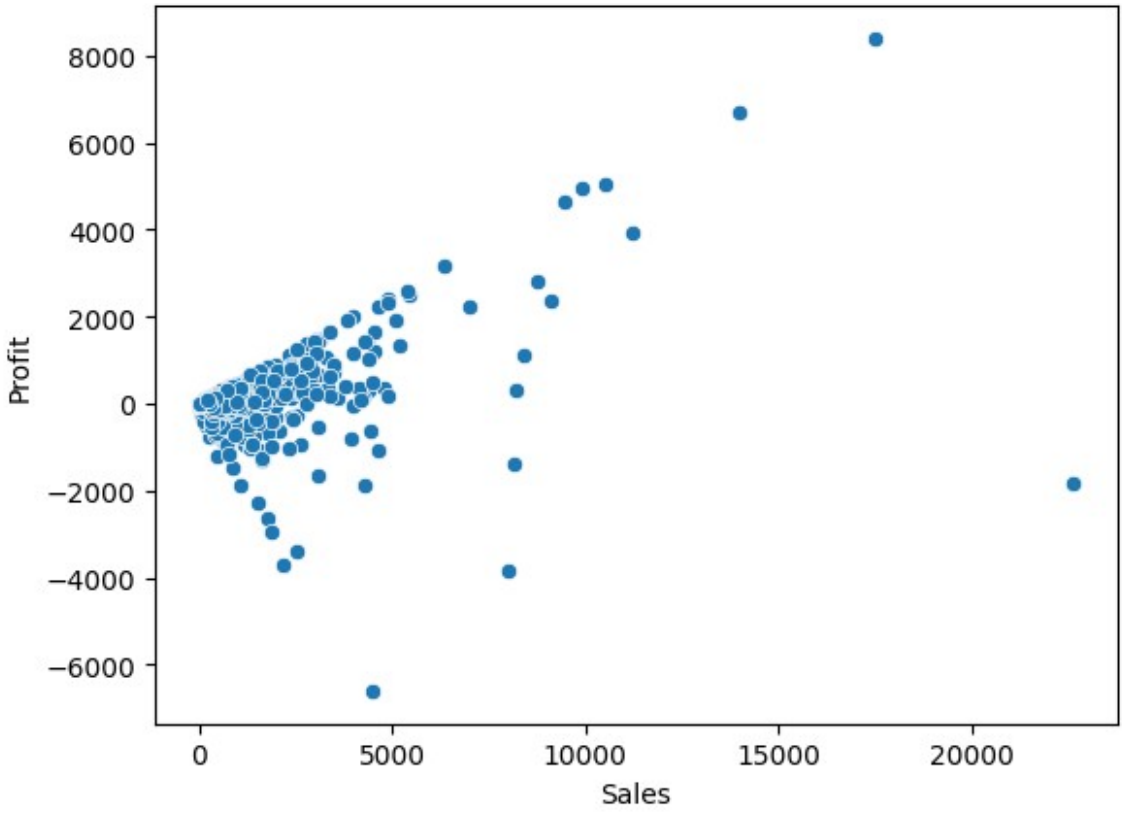
Present your findings using:

Power BI / Tableau dashboards

Jupyter Notebook reports

PDF/Slide decks with visuals

```
sns.scatterplot(x='Sales', y='Profit', data=data)
plt.show()
```



** Tools Required for Data Analysis**

1. Programming Languages

Python – Most popular for data analysis (pandas, numpy, matplotlib, seaborn).

R – Great for statistical analysis and visualization.

2. Data Handling & Analysis Libraries (Python)

pandas – Data manipulation and analysis.

numpy – Numerical computations.

matplotlib / seaborn / plotly – Data visualization.

seaborn – Basic Visulization library.

statsmodels – Statistical modeling.

3. Data Visualization Tools

Power BI – Interactive dashboards (Microsoft).

Tableau – Drag-and-drop data visualization.

Excel – Basic charts, pivot tables, and calculations.

Looker Studio (Google Data Studio) – Web-based BI tool.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```