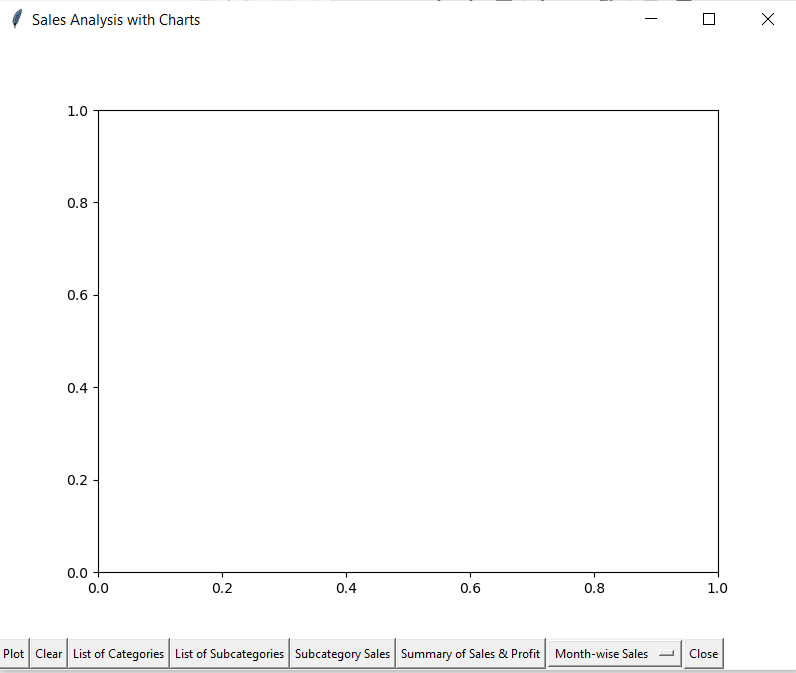
Python Project on Sales Analysis by Komal Khalpada

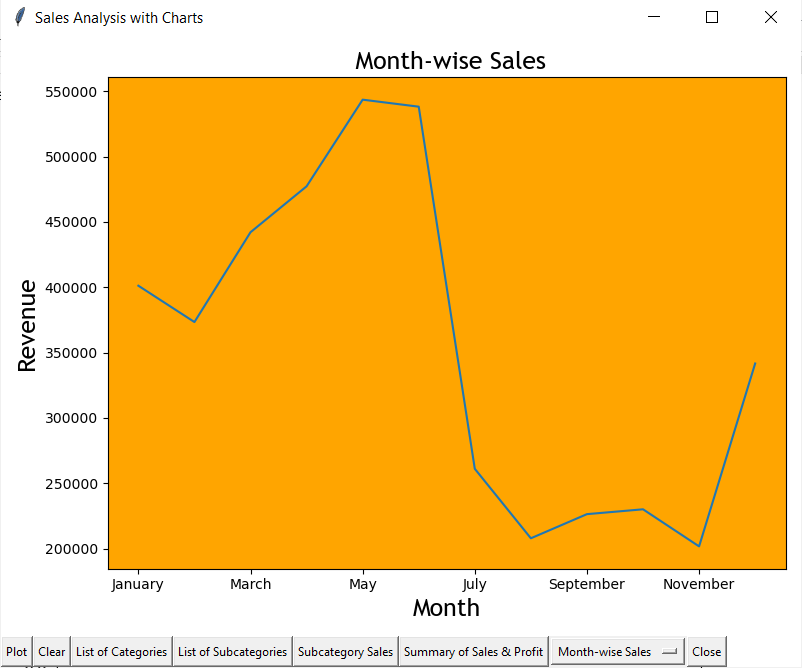


**import** tkinter **as** tk  
**import** pandas **as** pd  
**import** matplotlib.pyplot **as** plt  
**from** matplotlib.backends.backend\_tkagg **import** FigureCanvasTkAgg  
**import** numpy **as** np  
  
  
**class** PlotWindow:  
 **def** \_\_init\_\_(self, masterframe, size):  
 *"""  
 Initialize the PlotWindow object.  
  
 Args:  
 masterframe (tkinter.Tk or tkinter.Frame): The parent frame for the plot.  
 size (tuple): The size of the plot (width, height).  
 """* self.figure, self.axes = plt.subplots(figsize=size, dpi=100)  
 self.canvas = FigureCanvasTkAgg(self.figure, master=masterframe)  
 self.canvas.get\_tk\_widget().grid(row=0, column=0)  
 self.background\_color = **'orange'** self.font\_style = {**'family'**: **'Trebuchet MS'**, **'size'**: 18}  
  
 **def** plot\_data(self, chart\_type):  
 *"""  
 Plot the data based on the given chart type.  
  
 Args:  
 chart\_type (str): The type of chart to plot.  
 """* data = pd.read\_csv(**'SalesDL.csv'**)  
 self.axes.clear()  
  
 **if** chart\_type == **'Category-wise Total Sales'**:  
 print(PlotWindow.plot\_data\_prod\_cat.\_\_doc\_\_)  
 plot\_w.plot\_data\_prod\_cat(data)  
  
  
 **elif** chart\_type == **'Sub Category-wise(Bikes) Sales'**:  
 print(PlotWindow.plot\_sub\_cat.\_\_doc\_\_)  
 plot\_w.plot\_sub\_cat(data)  
  
 **elif** chart\_type == **'Month-wise Sales'**:  
 print(PlotWindow.plot\_month\_wise.\_\_doc\_\_)  
 plot\_w.plot\_month\_wise(data)  
  
 **elif** chart\_type == **'State-wise Sales'**:  
 print(PlotWindow.plot\_state\_wise.\_\_doc\_\_)  
 plot\_w.plot\_state\_wise(data)  
  
 **elif** chart\_type == **'Age-wise Sales'**:  
 print(PlotWindow.plot\_age\_wise.\_\_doc\_\_)  
 plot\_w.plot\_age\_wise(data)  
  
 self.axes.set\_facecolor(self.background\_color) *# Set background color* self.canvas.draw()  
  
 **def** plot\_data\_prod\_cat(self,data):  
 *"""  
 Category-wise Sales chart is plotted  
 """* group\_data = data.groupby(**'ProductCategory'**)[**'Revenue'**].sum()  
 labels = group\_data.index  
 values = group\_data.values  
 self.axes.pie(values, labels=labels, autopct=**'%1.1f%%'**, startangle=90)  
 self.axes.set\_title(**'Category-wise Total Sales'**, \*\*self.font\_style)  
  
 **def** plot\_sub\_cat(self,data):  
 *"""  
 Sub category-wise Sales Chart is plotted  
 """* bikes\_data = data[data[**'ProductCategory'**] == **'Bikes'**]  
 group\_data = bikes\_data.groupby(**'Sub Category'**)[**'Revenue'**].sum()  
 labels = group\_data.index  
 values = group\_data.values  
 self.axes.pie(values, labels=labels, autopct=**'%1.1f%%'**, startangle=90)  
 self.axes.set\_title(**'Sub Category-wise(Bikes) Sales '**, \*\*self.font\_style)  
  
 **def** plot\_month\_wise(self,data):  
 *"""  
 Month-wise Sales Chart is plotted  
 """* month\_order = [  
 **'January'**, **'February'**, **'March'**, **'April'**, **'May'**, **'June'**,  
 **'July'**, **'August'**, **'September'**, **'October'**, **'November'**, **'December'** ]  
 month\_data = data.groupby(**'Month'**)[**'Revenue'**].sum()  
 month\_data = month\_data.reindex(month\_order)  
 month\_data.plot(kind=**'line'**, ax=self.axes)  
 self.axes.set\_xlabel(**'Month'**, \*\*self.font\_style)  
 self.axes.set\_ylabel(**'Revenue'**, \*\*self.font\_style)  
 self.axes.set\_title(**' Month-wise Sales'**, \*\*self.font\_style)  
 self.figure.tight\_layout() *# Adjust layout for better visibility of labels* **def** plot\_state\_wise(self,data):  
 *"""  
 State-wise Sales Chart is plotted  
 """* state\_data = data.groupby(**'State'**)[**'Revenue'**].sum()  
 state\_data.plot(kind=**'bar'**, ax=self.axes)  
 self.axes.set\_xlabel(**'State'**, \*\*self.font\_style)  
 self.axes.set\_ylabel(**'Revenue'**, \*\*self.font\_style)  
 self.axes.set\_title(**' State-wise Sales'**, \*\*self.font\_style)  
 self.axes.tick\_params(axis=**'x'**, rotation=50) *# Rotate x-axis labels for better visibility  
  
 # Add value labels on top of bars* **for** index, value **in** enumerate(state\_data.values):  
 self.axes.text(index, value, str(round(value, 2)), ha=**'center'**, va=**'bottom'**, color=**'blue'**)  
 **def** plot\_age\_wise(self,data):  
 *"""  
 Age-wise Sales Chart is plotted  
 """* data[**'AgeGroup'**] = pd.cut(data[**'Customer Age'**], bins=[0, 30, 40, 50, 60, 100],  
 labels=[**'<=30'**, **'31-40'**, **'41-50'**, **'51-60'**, **'>=61'**])  
 age\_group\_data = data.groupby(**'AgeGroup'**)[**'Revenue'**].sum()  
 labels = age\_group\_data.index  
 values = age\_group\_data.values  
 self.axes.bar(labels, values)  
 self.axes.set\_xlabel(**'Age Group'**, \*\*self.font\_style)  
 self.axes.set\_ylabel(**'Sales Revenue'**, \*\*self.font\_style)  
 self.axes.set\_title(**'Histogram - Age-wise Sales'**, \*\*self.font\_style)  
 self.axes.tick\_params(axis=**'x'**, rotation=0) *# Rotate x-axis labels if needed* **def** clear\_plot(self):  
 self.axes.clear()  
 self.canvas.draw()  
  
**def** plot\_data():  
 selected\_chart = chart\_var.get()  
 plot\_w.plot\_data(selected\_chart)  
  
  
**def** clear():  
 plot\_w.clear\_plot()  
  
**def** show\_distinct\_categories():  
 data = pd.read\_csv(**'SalesDL.csv'**)  
 distinct\_categories = data[**'ProductCategory'**].unique()  
 distinct\_categories\_text = **"\n"**.join(distinct\_categories)  
 tk.messagebox.showinfo(**"Categories"**, distinct\_categories\_text)  
  
  
**def** show\_distinct\_subcategories():  
 data = pd.read\_csv(**'SalesDL.csv'**)  
 distinct\_categories = data[**'ProductCategory'**].unique()  
 subcategories\_by\_category = {}  
  
 **for** category **in** distinct\_categories:  
 subcategories = data[data[**'ProductCategory'**] == category][**'Sub Category'**].unique()  
 subcategories\_by\_category[category] = subcategories  
  
 message\_text = **""  
 for** category, subcategories **in** subcategories\_by\_category.items():  
 message\_text += **f"{**category**}: \nSubcategories: {', '**.join(subcategories)**}\n\n"** tk.messagebox.showinfo(**"Subcategories"**, message\_text)  
  
**def** show\_summary():  
 data = pd.read\_csv(**'SalesDL.csv'**)  
 total\_sales = data[**'Revenue'**].sum()  
 total\_profit = data[**'Profit'**].sum()  
 summary\_text = **f"Total Sales: {**total\_sales**}\nTotal Profit: {**total\_profit**}"** tk.messagebox.showinfo(**"Summary"**, summary\_text)  
  
  
**import** random  
  
**def** show\_subcategory\_sales():  
 data = pd.read\_csv(**'SalesDL.csv'**)  
  
 categories = data[**'ProductCategory'**].unique()  
  
 *# Generate random colors for each subcategory* colors = [**f'#{**random.randint(0, 0xFFFFFF)**:06x}' for** \_ **in** range(len(categories))]  
  
 fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 15), dpi=100)  
  
 **for** i, category **in** enumerate(categories):  
 category\_data = data[data[**'ProductCategory'**] == category]  
 subcategory\_data = category\_data.groupby(**'Sub Category'**)[**'Revenue'**].sum()  
  
 wedges, \_, autopct = axes[i].pie(subcategory\_data, labels=subcategory\_data.index, autopct=**'%1.1f%%'**, startangle=50, colors=colors)  
  
 *# Increase label font size and adjust label parameters for better visibility* plt.setp(autopct, fontsize=10, color=**'white'**)  
  
 *# Set equal aspect ratio to ensure circular shape* axes[i].axis(**'equal'**)  
  
 axes[i].set\_title(**f'Contribution of Subcategories to Sales ({**category**})'**)  
  
  
 plt.tight\_layout()  
 plt.show()  
  
  
root = tk.Tk() *# Creates the main tkinter window*root.configure(bg=**'white'**) *# Set background color of the main window*root.title(**"Sales Analysis with Charts"**) *# Main title of the window*mainframe = tk.Frame(root)*# Creates a frame inside the main window to hold the plot.*mainframe.grid(row=0, column=0, sticky=**"nsew"**)*# Places the mainframe in the first row and first column of the grid layout within the main window.*print(PlotWindow.plot\_data.\_\_doc\_\_)*#Prints the docstring of the plot\_data() method, which describes its functionality.*plot\_w = PlotWindow(mainframe, (8, 6))  
  
buttonframe = tk.Frame(root, bg=**"white"**)*#Creates a frame inside the root window to hold the buttons with a white background.*buttonframe.grid(row=1, column=0, sticky=**"nsew"**)*#Places the buttonframe in the second row and first column of the grid layout within the root window.*b1 = tk.Button(buttonframe, text=**"Plot"**, command=plot\_data)*#Creates a button with the label "Plot" and associates it with the plot\_data() function*b1.grid(row=1, column= 1, sticky=**"nsew"**)  
b2 = tk.Button(buttonframe, text=**"Clear"**, command=clear)  
b2.grid(row=1, column=2, sticky=**"nsew"**)  
b3 = tk.Button(buttonframe, text=**"List of Categories"**, command=show\_distinct\_categories)  
b3.grid(row=1, column=3, sticky=**"nsew"**)  
b4 = tk.Button(buttonframe, text=**"List of Subcategories"**, command=show\_distinct\_subcategories)  
b4.grid(row=1, column=4, sticky=**"nsew"**)  
b5 = tk.Button(buttonframe, text=**"Subcategory Sales"**, command=show\_subcategory\_sales)  
b5.grid(row=1, column=5, sticky=**"nsew"**)  
b6 = tk.Button(buttonframe, text=**"Summary of Sales & Profit"**, command=show\_summary)  
b6.grid(row=1, column=6, sticky=**"nsew"**)  
b7 = tk.Button(buttonframe, text=**"Close"**, command=root.quit)  
b7.grid(row=1, column=8, sticky=**"nsew"**)  
  
  
*# Dropdown menu to select chart type*chart\_types = [**'Month-wise Sales'**, **'State-wise Sales'**, **'Age-wise Sales'**,**'Category-wise Total Sales'**, **'Sub Category-wise(Bikes) Sales'**]  
chart\_var = tk.StringVar(root)  
chart\_var.set(chart\_types[0])  
chart\_dropdown = tk.OptionMenu(buttonframe, chart\_var, \*chart\_types)  
chart\_dropdown.grid(row=1, column=7, sticky=**"nsew"**)  
  
root.mainloop()

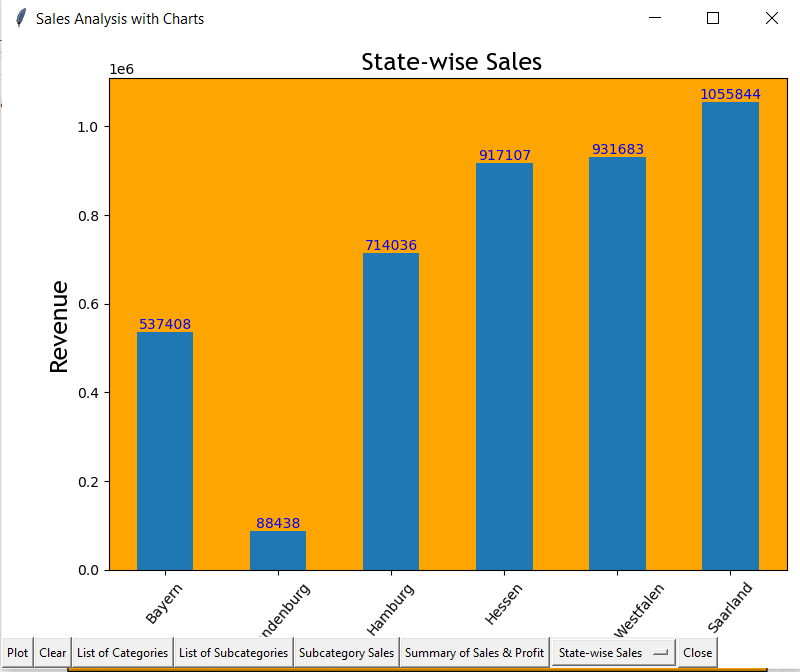
**Main window**



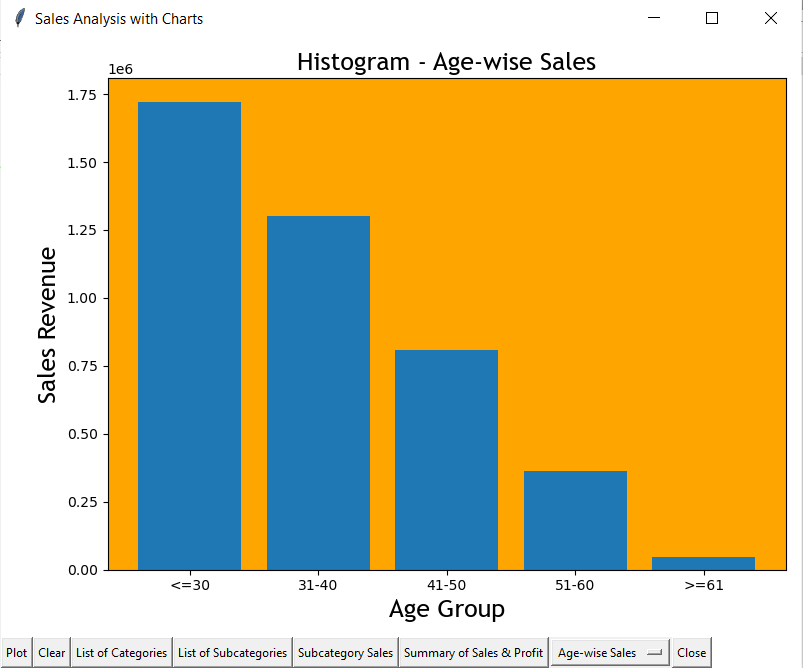
* **Plot : Month-wise Sales**



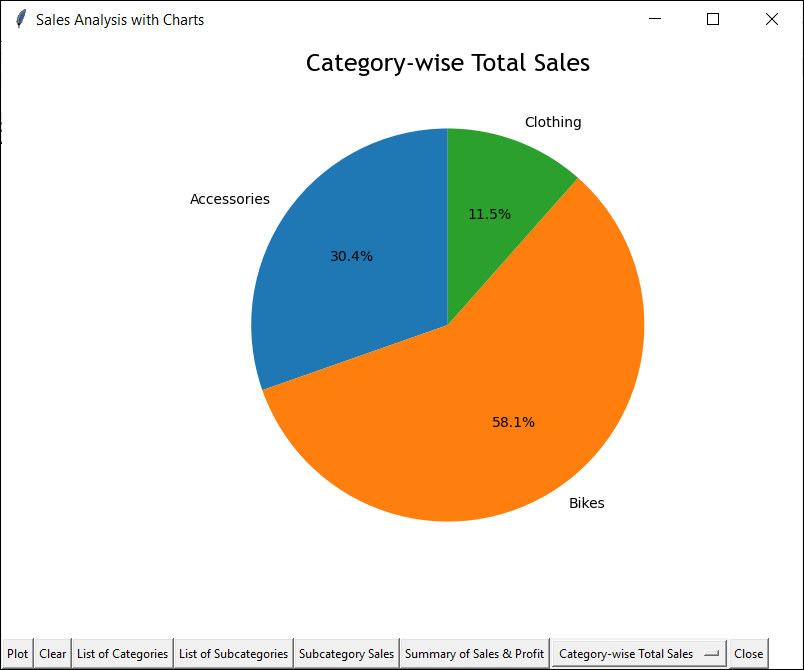
* **Plot : State-wise Sales**



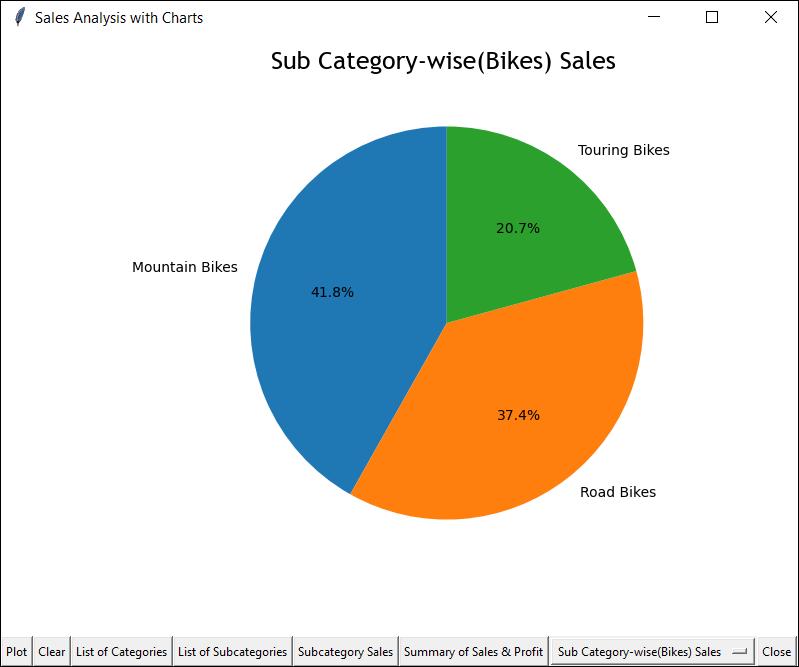
* **Plot : Age-wise Sales**



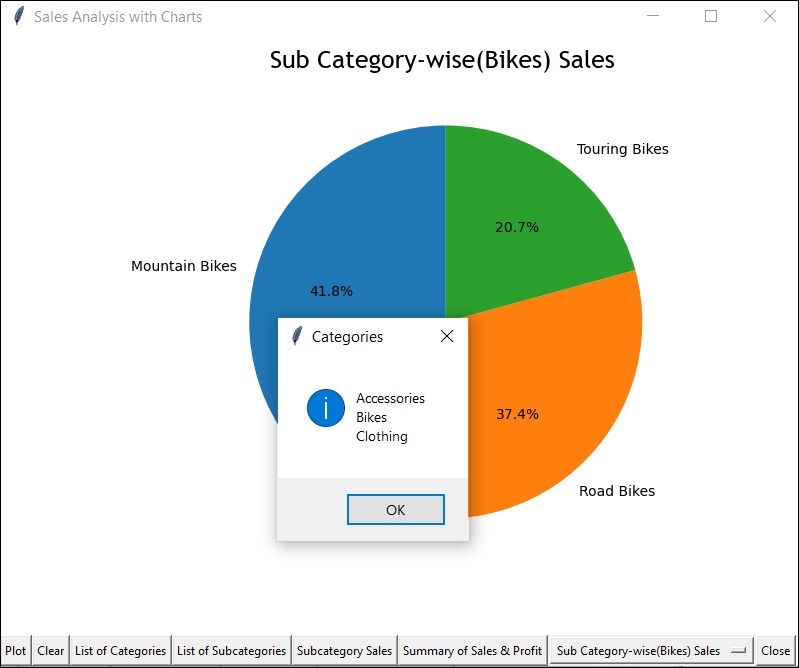
* **Plot : Category-wise Sales**



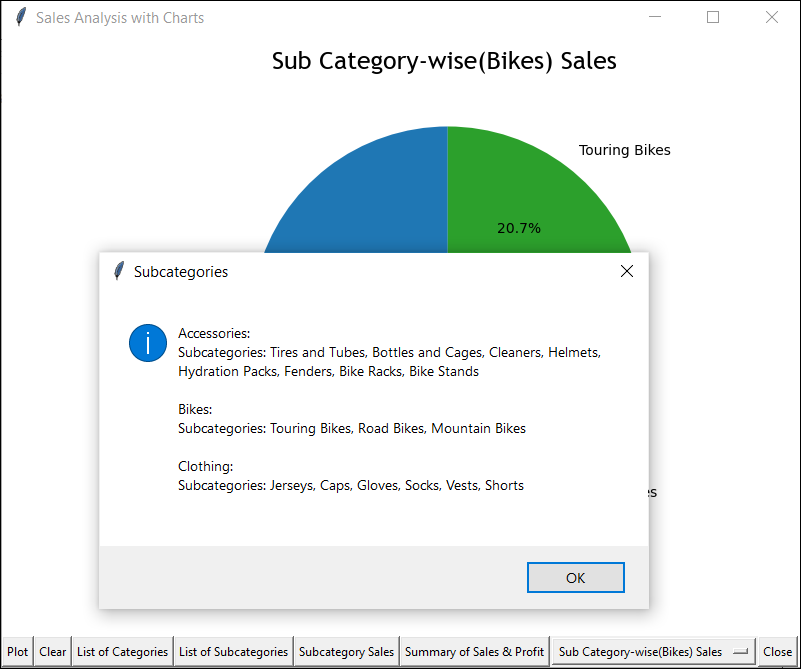
* **Plot : Sub-category-wise Sales**



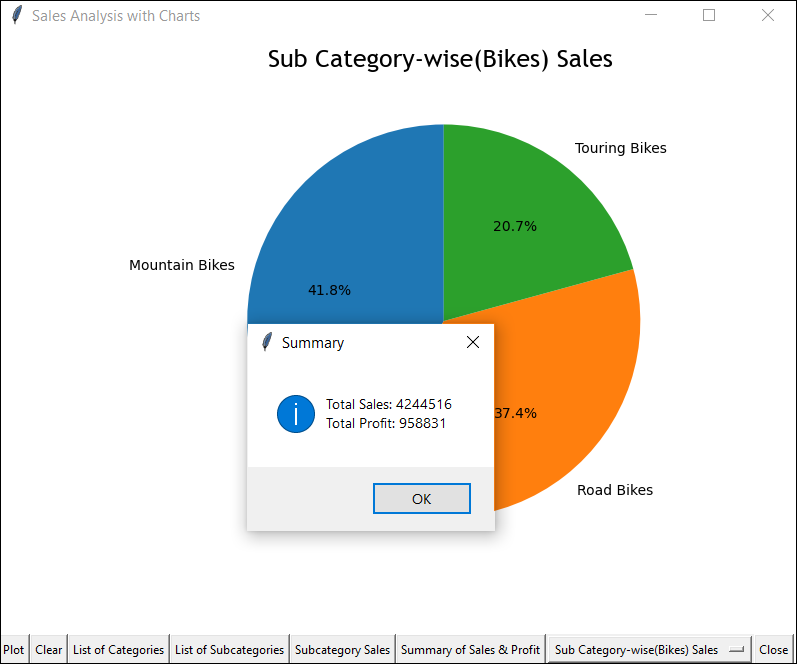
* **List of Categories**



* **List of sub-categories**



* **Summary of sales and profit**



* **Subcategory Sales**

