# Sales Data Analysis Tool Documentation

## Introduction

This document provides an overview of the Sales Data Analysis Tool. The tool is built using Python and includes functionalities for loading, analyzing, and visualizing sales data. The application features a graphical user interface (GUI) created with Tkinter, which integrates various types of visualizations including bar charts, line charts, and pie charts.

## Code

Below is the complete code for the Sales Data Analysis Tool:

import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
from tkinter import Tk, filedialog, Button, Label, Frame, Text, Scrollbar, LEFT, RIGHT, Y, END  
from tkinter import ttk  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
from matplotlib.widgets import Cursor  
import matplotlib.dates as mdates  
  
def load\_and\_clean\_data(file\_path):  
 df = pd.read\_excel(file\_path)  
 df['Order Date'] = pd.to\_datetime(df['Order Date'])  
 df['Total Revenue'] = pd.to\_numeric(df['Total Revenue'], errors='coerce')  
 df = df.dropna()  
 df['Total Revenue'] = df['Total Revenue'] / 1\_000\_000  
 return df  
  
def analyze\_data(df):  
 total\_sales = df['Total Revenue'].sum()  
 avg\_sales = df['Total Revenue'].mean()  
 sales\_by\_product = df.groupby('Item Type')['Total Revenue'].sum().reset\_index()  
 return total\_sales, avg\_sales, sales\_by\_product  
  
def generate\_visualizations(df, sort\_order='desc'):  
 sns.set\_style("darkgrid", {"axes.facecolor": "#2c3e50"})  
 bar\_data = df.groupby('Item Type')['Total Revenue'].sum().reset\_index()  
 bar\_data = bar\_data.sort\_values(by='Total Revenue', ascending=(sort\_order == 'asc'))  
 fig, ax = plt.subplots(figsize=(10, 6))  
 bars = sns.barplot(x='Item Type', y='Total Revenue', data=bar\_data, ax=ax, palette="viridis")  
 ax.set\_title('Total Sales by Product', color="#e0e0e0")  
 ax.set\_facecolor('#2c3e50')  
 ax.tick\_params(colors='#e0e0e0')  
 def on\_hover\_bar(event):  
 if event.inaxes == ax:  
 for bar in bars.patches:  
 if bar.contains(event)[0]:  
 height = bar.get\_height()  
 ax.annotate(f'{height:,.2f}M',  
 (bar.get\_x() + bar.get\_width() / 2, height),  
 xytext=(0, 5),  
 textcoords='offset points',   
 ha='center', va='bottom',  
 color='#e0e0e0')  
 fig.canvas.draw\_idle()  
 break  
 else:  
 ax.annotate("", xy=(0,0), xytext=(0,0))  
 fig.canvas.draw\_idle()  
 fig.canvas.mpl\_connect("motion\_notify\_event", on\_hover\_bar)  
 line\_fig, line\_ax = plt.subplots(figsize=(10, 6))  
 line\_data = df.set\_index('Order Date').resample('Y').sum()['Total Revenue']  
 line\_ax.plot(line\_data.index, line\_data.values, color='#ff6f61')  
 line\_ax.xaxis.set\_major\_formatter(mdates.DateFormatter('%Y'))  
 line\_ax.xaxis.set\_major\_locator(mdates.YearLocator())  
 line\_ax.set\_title('Sales Over Time', color="#e0e0e0")  
 line\_ax.set\_ylabel('Total Revenue (Millions)', color="#e0e0e0")  
 line\_ax.set\_xlabel('Year', color="#e0e0e0")  
 line\_ax.set\_facecolor('#2c3e50')  
 line\_ax.tick\_params(colors='#e0e0e0')  
 cursor = Cursor(line\_ax, useblit=True, color='red', linewidth=1)  
 def on\_hover\_line(event):  
 if event.inaxes == line\_ax:  
 xdata = event.xdata  
 ydata = event.ydata  
 line\_ax.annotate(f'{ydata:,.2f}M',  
 (xdata, ydata),  
 xytext=(0, 5),  
 textcoords='offset points',  
 ha='center', va='bottom',  
 color='#e0e0e0')  
 line\_fig.canvas.draw\_idle()  
 line\_fig.canvas.mpl\_connect("motion\_notify\_event", on\_hover\_line)  
 pie\_fig, pie\_ax = plt.subplots(figsize=(7, 7))  
 sales\_by\_product = df.groupby('Item Type')['Total Revenue'].sum()  
 wedges, texts, autotexts = pie\_ax.pie(sales\_by\_product, labels=sales\_by\_product.index, autopct='%1.1f%%',  
 startangle=140, colors=sns.color\_palette("magma"),  
 textprops={'color': '#e0e0e0'})  
 pie\_ax.set\_title('Sales Distribution by Product', color="#e0e0e0")  
 pie\_ax.set\_facecolor('#2c3e50')  
 def on\_hover\_pie(event):  
 if event.inaxes == pie\_ax:  
 for wedge in wedges:  
 if wedge.contains(event)[0]:  
 wedge.set\_edgecolor('yellow')  
 wedge.set\_linewidth(2)  
 pie\_fig.canvas.draw\_idle()  
 break  
 else:  
 for wedge in wedges:  
 wedge.set\_edgecolor('black')  
 wedge.set\_linewidth(1)  
 pie\_fig.canvas.draw\_idle()  
 pie\_fig.canvas.mpl\_connect("motion\_notify\_event", on\_hover\_pie)  
 plt.tight\_layout()  
 fig.patch.set\_facecolor('#2c3e50')  
 line\_fig.patch.set\_facecolor('#2c3e50')  
 pie\_fig.patch.set\_facecolor('#2c3e50')  
 return fig, line\_fig, pie\_fig  
  
def generate\_summary\_report(total\_sales, avg\_sales, sales\_by\_product):  
 report = f"Summary Report\n"  
 report += f"====================\n"  
 report += f"Total Sales: ${total\_sales:,.2f} Million\n"  
 report += f"Average Sales: ${avg\_sales:,.2f} Million\n\n"  
 report += "Sales by Product:\n"  
 report += sales\_by\_product.to\_markdown(index=False)  
 return report  
  
def show\_gui():  
 def on\_load():  
 file\_path = filedialog.askopenfilename(filetypes=[("Excel files", "\*.xlsx")])  
 if file\_path:  
 df = load\_and\_clean\_data(file\_path)  
 total\_sales, avg\_sales, sales\_by\_product = analyze\_data(df)  
 report = generate\_summary\_report(total\_sales, avg\_sales, sales\_by\_product)  
 text\_box.delete(1.0, END)  
 text\_box.insert(END, report)  
 fig\_dict = {  
 'bar\_desc': generate\_visualizations(df, 'desc')[0],  
 'bar\_asc': generate\_visualizations(df, 'asc')[0],  
 'line': generate\_visualizations(df, 'desc')[1],  
 'pie': generate\_visualizations(df, 'desc')[2]  
 }  
 def update\_canvas(fig):  
 for widget in canvas\_frame.winfo\_children():  
 widget.destroy()  
 canvas = FigureCanvasTkAgg(fig, master=canvas\_frame)  
 canvas.draw()  
 canvas.get\_tk\_widget().pack(side=LEFT, fill='both', expand=True)  
 def show\_bar\_chart\_asc():  
 update\_canvas(fig\_dict['bar\_asc'])  
 def show\_bar\_chart\_desc():  
 update\_canvas(fig\_dict['bar\_desc'])  
 def show\_line\_chart():  
 update\_canvas(fig\_dict['line'])  
 def show\_pie\_chart():  
 update\_canvas(fig\_dict['pie'])  
 btn\_bar\_asc.config(command=show\_bar\_chart\_asc)  
 btn\_bar\_desc.config(command=show\_bar\_chart\_desc)  
 btn\_line.config(command=show\_line\_chart)  
 btn\_pie.config(command=show\_pie\_chart)  
 root = Tk()  
 root.title("Sales Data Analysis")  
 root.geometry("1000x700")  
 root.configure(bg='lightskyblue1')  
 title\_label = Label(root, text="Sales Data Analysis", font=("Arial", 20), bg='lightskyblue1')  
 title\_label.pack(pady=10)  
 btn\_frame = Frame(root, bg='lightskyblue1')  
 btn\_frame.pack(pady=10)  
 btn\_load = Button(btn\_frame, text="Load Data", command=on\_load, font=("Arial", 14))  
 btn\_load.pack(side=LEFT, padx=10)  
 btn\_bar\_desc = Button(btn\_frame, text="Bar Chart (Desc)", font=("Arial", 14))  
 btn\_bar\_desc.pack(side=LEFT, padx=10)  
 btn\_bar\_asc = Button(btn\_frame, text="Bar Chart (Asc)", font=("Arial", 14))  
 btn\_bar\_asc.pack(side=LEFT, padx=10)  
 btn\_line = Button(btn\_frame, text="Line Chart", font=("Arial", 14))  
 btn\_line.pack(side=LEFT, padx=10)  
 btn\_pie = Button(btn\_frame, text="Pie Chart", font=("Arial", 14))  
 btn\_pie.pack(side=LEFT, padx=10)  
 text\_frame = Frame(root, bg='lightskyblue1')  
 text\_frame.pack(fill=BOTH, expand=True)  
 text\_box = Text(text\_frame, wrap=WORD, font=("Arial", 12))  
 text\_box.pack(side=LEFT, fill=BOTH, expand=True)  
 scrollbar = Scrollbar(text\_frame, command=text\_box.yview)  
 scrollbar.pack(side=RIGHT, fill=Y)  
 text\_box.config(yscrollcommand=scrollbar.set)  
 canvas\_frame = Frame(root, bg='lightskyblue1')  
 canvas\_frame.pack(fill=BOTH, expand=True)  
 root.mainloop()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 show\_gui()