**Python Final Project**

**Sales performance dashboard**

**{Python code}**

import openpyxl

from datetime import datetime

import os.path

import matplotlib.pyplot as plt

class ProductItem:

def \_\_init\_\_(self, itemName, itemCost, itemStock, itemType):

self.itemName = itemName

self.itemCost = itemCost

self.itemStock = itemStock

self.itemType = itemType

def calculateTotalProfit(item):

return item.itemCost \* item.itemStock

def formatItemDetails(items):

salesReport = "Store Sales Report:\n\n"

totalRevenue = 0

for item in items:

profit = calculateTotalProfit(item)

salesReport += f"Product: {item.itemName}\n"

salesReport += f"Cost: ${item.itemCost:.2f}\n"

salesReport += f"Stock: {item.itemStock}\n"

salesReport += f"Type: {item.itemType}\n"

salesReport += f"Profit: ${profit:.2f}\n\n"

totalRevenue += profit

salesReport += f"Overall profit from the entire store is: ${totalRevenue:.2f}\n"

return salesReport

def createSalesReport(items):

reportFileName = "storeSalesReport.xlsx"

if not os.path.isfile(reportFileName):

salesWorkbook = openpyxl.Workbook()

salesWorksheet = salesWorkbook.active

headers = ["Product Name", "Cost", "Stock", "Product Type", "Profit", "Date and Time"]

for columnNumber, columnHeader in enumerate(headers, 1):

salesWorksheet.cell(row=1, column=columnNumber, value=columnHeader)

else:

salesWorkbook = openpyxl.load\_workbook(reportFileName)

salesWorksheet = salesWorkbook.active

for rowNumber, productItem in enumerate(items, salesWorksheet.max\_row + 1):

salesWorksheet.cell(row=rowNumber, column=1, value=productItem.itemName)

salesWorksheet.cell(row=rowNumber, column=2, value=productItem.itemCost)

salesWorksheet.cell(row=rowNumber, column=3, value=productItem.itemStock)

salesWorksheet.cell(row=rowNumber, column=4, value=productItem.itemType)

salesWorksheet.cell(row=rowNumber, column=5, value=calculateTotalProfit(productItem))

salesWorksheet.cell(row=rowNumber, column=6, value=datetime.now().strftime("%Y-%m-%d %H:%M:%S"))

salesWorksheet.cell(row=salesWorksheet.max\_row + 1, column=1, value="Total Profit")

salesWorksheet.cell(row=salesWorksheet.max\_row, column=5, value=sum(calculateTotalProfit(productItem) for productItem in items))

try:

salesWorkbook.save(reportFileName)

print(f"Sales report updated successfully at {reportFileName}")

except PermissionError:

print(f"PermissionError: Unable to save the file. Check write permissions for the specified path.")

def getUserInput(prompt, dataType=float):

validInput = False

userInputData = None

while not validInput:

try:

userInputData = dataType(input(prompt))

validInput = True

except ValueError:

print("Invalid input. Please provide a valid value.")

return userInputData

def collectProductItems():

print("Welcome to the Store Sales Reporting System!")

numOfItems = int(getUserInput("Provide the number of items you want to create a report for: ", int))

productItems = []

for i in range(1, numOfItems + 1):

print(f"\nProvide details for item {i}:")

itemName = input("Provide the product name: ")

itemCost = getUserInput("Provide the cost of the item: ")

itemStock = getUserInput("Provide the current stock quantity: ", int)

itemType = input("Provide the type of the item: ")

productItem = ProductItem(itemName, itemCost, itemStock, itemType)

productItems.append(productItem)

print("\nThank you for providing the details!")

return productItems

productItems = collectProductItems()

salesReport = formatItemDetails(productItems)

print(salesReport)

createSalesReport(productItems)

totalRevenue = sum(calculateTotalProfit(productItem) for productItem in productItems)

itemProfits = [calculateTotalProfit(productItem) for productItem in productItems]

itemNames = [productItem.itemName for productItem in productItems]

percentages = [(profit / totalRevenue) \* 100 for profit in itemProfits]

plt.figure(figsize=(8, 8))

plt.pie(percentages, labels=itemNames, autopct='%1.1f%%', startangle=140)

plt.title("Profit Distribution Among Items")

plt.axis('equal')

plt.show()