```
import tkinter as tk
from tkinter import ttk, messagebox
import sqlite3
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas
import datetime

# Database setup
conn = sqlite3.connect('billing.db')
c = conn.cursor()

# Create tables if they don't exist
c.execute('''CREATE TABLE IF NOT EXISTS products
             (product_id INTEGER PRIMARY KEY, product_name TEXT, price
REAL, stock_quantity INTEGER)''')
c.execute('''CREATE TABLE IF NOT EXISTS customers
             (customer_id INTEGER PRIMARY KEY, customer_name TEXT,
contact_info TEXT)''')
c.execute('''CREATE TABLE IF NOT EXISTS transactions
             (transaction_id INTEGER PRIMARY KEY, customer_id INTEGER,
transaction_date TEXT, total_amount REAL)''')
c.execute('''CREATE TABLE IF NOT EXISTS transaction_details
             (detail_id INTEGER PRIMARY KEY, transaction_id INTEGER,
product_id INTEGER, quantity INTEGER, price REAL)''')

conn.commit()

# Main application class
class BillingApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Billing Software")
        self.root.geometry("800x600")

        # Notebook (tabbed interface)
        self.notebook = ttk.Notebook(root)
        self.notebook.pack(expand=1, fill='both')

        # Tabs
        self.product_tab = ttk.Frame(self.notebook)
        self.customer_tab = ttk.Frame(self.notebook)
        self.billing_tab = ttk.Frame(self.notebook)
        self.notebook.add(self.product_tab, text="Products")
        self.notebook.add(self.customer_tab, text="Customers")
        self.notebook.add(self.billing_tab, text="Billing")

        # Product Tab
        self.create_product_tab()

        # Customer Tab
```

```
        self.create_customer_tab()

        # Billing Tab
        self.create_billing_tab()

    def create_product_tab(self):
        # Product management UI
        tk.Label(self.product_tab, text="Product Name").grid(row=0,
column=0)
        tk.Label(self.product_tab, text="Price").grid(row=1, column=0)
        tk.Label(self.product_tab, text="Stock Quantity").grid(row=2,
column=0)

        self.product_name_entry = tk.Entry(self.product_tab)
        self.price_entry = tk.Entry(self.product_tab)
        self.stock_entry = tk.Entry(self.product_tab)

        self.product_name_entry.grid(row=0, column=1)
        self.price_entry.grid(row=1, column=1)
        self.stock_entry.grid(row=2, column=1)

        tk.Button(self.product_tab, text="Add Product",
command=self.add_product).grid(row=3, column=0, columnspan=2)
        tk.Button(self.product_tab, text="View Products",
command=self.view_products).grid(row=4, column=0, columnspan=2)

        self.product_list = tk.Listbox(self.product_tab)
        self.product_list.grid(row=5, column=0, columnspan=2)

    def create_customer_tab(self):
        # Customer management UI
        tk.Label(self.customer_tab, text="Customer Name").grid(row=0,
column=0)
        tk.Label(self.customer_tab, text="Contact Info").grid(row=1,
column=0)

        self.customer_name_entry = tk.Entry(self.customer_tab)
        self.contact_entry = tk.Entry(self.customer_tab)

        self.customer_name_entry.grid(row=0, column=1)
        self.contact_entry.grid(row=1, column=1)

        tk.Button(self.customer_tab, text="Add Customer",
command=self.add_customer).grid(row=2, column=0, columnspan=2)
        tk.Button(self.customer_tab, text="View Customers",
command=self.view_customers).grid(row=3, column=0, columnspan=2)

        self.customer_list = tk.Listbox(self.customer_tab)
        self.customer_list.grid(row=4, column=0, columnspan=2)
```

```
    def create_billing_tab(self):
        # Billing and invoice generation UI
        tk.Label(self.billing_tab, text="Select Customer").grid(row=0,
column=0)
        tk.Label(self.billing_tab, text="Select Product").grid(row=1,
column=0)
        tk.Label(self.billing_tab, text="Quantity").grid(row=2,
column=0)

        self.customer_combobox = ttk.Combobox(self.billing_tab)
        self.product_combobox = ttk.Combobox(self.billing_tab)
        self.quantity_entry = tk.Entry(self.billing_tab)

        self.customer_combobox.grid(row=0, column=1)
        self.product_combobox.grid(row=1, column=1)
        self.quantity_entry.grid(row=2, column=1)

        tk.Button(self.billing_tab, text="Add to Bill",
command=self.add_to_bill).grid(row=3, column=0, columnspan=2)
        tk.Button(self.billing_tab, text="Generate Invoice",
command=self.generate_invoice).grid(row=4, column=0, columnspan=2)

        self.bill_list = tk.Listbox(self.billing_tab)
        self.bill_list.grid(row=5, column=0, columnspan=2)

        self.load_customers_and_products()

    def add_product(self):
        product_name = self.product_name_entry.get()
        price = float(self.price_entry.get())
        stock = int(self.stock_entry.get())
        c.execute("INSERT INTO products (product_name, price,
stock_quantity) VALUES (?, ?, ?)", (product_name, price, stock))
        conn.commit()
        messagebox.showinfo("Success", "Product added successfully")
        self.product_name_entry.delete(0, tk.END)
        self.price_entry.delete(0, tk.END)
        self.stock_entry.delete(0, tk.END)

    def view_products(self):
        self.product_list.delete(0, tk.END)
        c.execute("SELECT * FROM products")
        products = c.fetchall()
        for product in products:
            self.product_list.insert(tk.END, f"{product[1]} - $
{product[2]} (Stock: {product[3]})")

    def add_customer(self):
        customer_name = self.customer_name_entry.get()
        contact_info = self.contact_entry.get()
```

```
        c.execute("INSERT INTO customers (customer_name, contact_info)
VALUES (?, ?)", (customer_name, contact_info))
        conn.commit()
        messagebox.showinfo("Success", "Customer added successfully")
        self.customer_name_entry.delete(0, tk.END)
        self.contact_entry.delete(0, tk.END)

    def view_customers(self):
        self.customer_list.delete(0, tk.END)
        c.execute("SELECT * FROM customers")
        customers = c.fetchall()
        for customer in customers:
            self.customer_list.insert(tk.END, f"{customer[1]} -
{customer[2]}")

    def load_customers_and_products(self):
        c.execute("SELECT customer_name FROM customers")
        customers = [customer[0] for customer in c.fetchall()]
        self.customer_combobox['values'] = customers

        c.execute("SELECT product_name FROM products")
        products = [product[0] for product in c.fetchall()]
        self.product_combobox['values'] = products

    def add_to_bill(self):
        product_name = self.product_combobox.get()
        quantity = int(self.quantity_entry.get())

        c.execute("SELECT price, stock_quantity FROM products WHERE
product_name=?", (product_name,))
        product = c.fetchone()

        if product and product[1] >= quantity:
            price = product[0]
            total_price = price * quantity
            self.bill_list.insert(tk.END, f"{product_name} -
{quantity} x ${price} = ${total_price}")
            c.execute("UPDATE products SET stock_quantity =
stock_quantity - ? WHERE product_name=?", (quantity, product_name))
            conn.commit()
        else:
            messagebox.showerror("Error", "Insufficient stock or
invalid product selected")

    def generate_invoice(self):
        customer_name = self.customer_combobox.get()
        date = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        total_amount = 0

        invoice_items = []
```

```
        for item in self.bill_list.get(0, tk.END):
            product_name, rest = item.split(' - ')
            quantity, total_price = rest.split(' x ')[1].split(' = $')
            total_price = float(total_price)
            total_amount += total_price
            invoice_items.append((product_name, int(quantity),
total_price))

        c.execute("SELECT customer_id FROM customers WHERE
customer_name=?", (customer_name,))
        customer_id = c.fetchone()[0]
        c.execute("INSERT INTO transactions (customer_id,
transaction_date, total_amount) VALUES (?, ?, ?)", (customer_id, date,
total_amount))
        transaction_id = c.lastrowid

        for product_name, quantity, price in invoice_items:
            c.execute("SELECT product_id FROM products WHERE
product_name=?", (product_name,))
            product_id = c.fetchone()[0]
            c.execute("INSERT INTO transaction_details
(transaction_id, product_id, quantity, price) VALUES (?, ?, ?, ?)",
                      (transaction_id, product_id, quantity, price))

        conn.commit()

        self.create_pdf_invoice(customer_name, date, invoice_items,
total_amount)
        messagebox.showinfo("Success", "Invoice generated
successfully")
        self.bill_list.delete(0, tk.END)

    def create_pdf_invoice(self, customer_name, date, items,
total_amount):
        filename = f"Invoice_{datetime.datetime.now().strftime('%Y%m%d
%H%M%S')}.pdf"
        c = canvas.Canvas(filename, pagesize=letter)
        c.setFont("Helvetica", 12)

        # Title and date
        c.drawString(100, 750, "INVOICE")
        c.drawString(100, 735, f"Date: {date}")

        # Customer info
        c.drawString(100, 715, f"Customer: {customer_name}")

        # Table header
        c.drawString(100, 685, "Product")
        c.drawString(300, 685, "Quantity")
        c.drawString(400, 685, "Price")
```

```
        # Table content
        y = 665
        for item in items:
            product_name, quantity, price = item
            c.drawString(100, y, product_name)
            c.drawString(300, y, str(quantity))
            c.drawString(400, y, f"${price}")
            y -= 20

        # Total amount
        c.drawString(100, y - 20, f"Total Amount: ${total_amount}")

        c.save()

# Main function
if __name__ == "__main__":
    root = tk.Tk()
    app = BillingApp(root)
    root.mainloop()
```