admin.py:

import tkinter as tk  
from tkinter import ttk, messagebox  
import pymysql  
from pymysql import OperationalError  
import datetime  
class Database:  
 def \_\_init\_\_(self):  
 self.config = {  
 "host": "localhost",  
 "user": "root",  
 "password": "zcy123456",  
 "database": "restaurant\_db",  
 "cursorclass": pymysql.cursors.DictCursor  
 }  
 self.connect()  
  
 def connect(self):  
 try:  
 self.conn = pymysql.connect(\*\*self.config)  
 except OperationalError as e:  
 messagebox.showerror("数据库连接失败", f"请检查配置: {str(e)}")  
 raise  
  
 def query(self, sql, params=None):  
 with self.conn.cursor() as cursor:  
 cursor.execute(sql, params or ())  
 return cursor.fetchall()  
  
 def execute(self, sql, params=None):  
 with self.conn.cursor() as cursor:  
 cursor.execute(sql, params or ())  
 self.conn.commit()  
class AdminApp:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("膳食阁-管理员界面")  
 self.root.geometry("1200x600")  
 self.db = Database()  
  
 self.create\_status\_bar()  
 self.notebook = ttk.Notebook(root)  
 self.notebook.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)  
  
 self.init\_menu\_tab()  
 self.init\_sales\_tab()  
 self.init\_profit\_tab()  
 self.init\_orders\_tab()  
  
 self.update\_clock()  
  
 def create\_status\_bar(self):  
 status\_frame = ttk.Frame(self.root)  
 status\_frame.pack(side=tk.BOTTOM, fill=tk.X)  
 ttk.Label(status\_frame, text="当前用户: 管理员", padding=(5, 2)).pack(side=tk.LEFT)  
 self.time\_label = ttk.Label(status\_frame, text="", padding=(5, 2))  
 self.time\_label.pack(side=tk.RIGHT)  
  
 def update\_clock(self):  
 current\_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
 self.time\_label.config(text=current\_time)  
 self.root.after(1000, self.update\_clock)  
  
 def init\_menu\_tab(self):  
 tab = ttk.Frame(self.notebook)  
 self.notebook.add(tab, text="菜单管理")  
  
 ttk.Label(tab, text="菜品管理", font=("微软雅黑", 14), background="#4a7abc", foreground="white").pack(pady=10,fill=tk.X,padx=10)  
  
 btn\_frame = ttk.Frame(tab)  
 btn\_frame.pack(pady=5, fill=tk.X)  
 ttk.Button(btn\_frame, text="添加菜品", command=self.add\_dish).pack(side=tk.LEFT, padx=5)  
 ttk.Button(btn\_frame, text="刷新", command=self.load\_menu).pack(side=tk.LEFT, padx=5)  
  
 search\_frame = ttk.Frame(tab)  
 search\_frame.pack(fill=tk.X, pady=5)  
 ttk.Label(search\_frame, text="搜索菜品:").pack(side=tk.LEFT, padx=5)  
 self.search\_entry = ttk.Entry(search\_frame, width=30)  
 self.search\_entry.pack(side=tk.LEFT, padx=5)  
 ttk.Button(search\_frame, text="搜索", command=self.search\_dishes).pack(side=tk.LEFT, padx=5)  
  
 self.menu\_tree = ttk.Treeview(tab, columns=("id", "name", "price", "profit", "desc", "edit", "delete"),show="headings")  
 self.menu\_tree.heading("id", text="ID")  
 self.menu\_tree.heading("name", text="名称")  
 self.menu\_tree.heading("price", text="价格")  
 self.menu\_tree.heading("profit", text="单份利润")  
 self.menu\_tree.heading("desc", text="描述")  
 self.menu\_tree.heading("edit", text="编辑")  
 self.menu\_tree.heading("delete", text="删除")  
  
 self.menu\_tree.column("id", width=50, anchor=tk.CENTER)  
 self.menu\_tree.column("name", width=150, anchor=tk.CENTER)  
 self.menu\_tree.column("price", width=80, anchor=tk.CENTER)  
 self.menu\_tree.column("profit", width=80, anchor=tk.CENTER)  
 self.menu\_tree.column("desc", width=300, anchor=tk.W)  
 self.menu\_tree.column("edit", width=60, anchor=tk.CENTER)  
 self.menu\_tree.column("delete", width=60, anchor=tk.CENTER)  
  
 self.menu\_tree.pack(fill=tk.BOTH, expand=True)  
 self.menu\_tree.bind("<ButtonRelease-1>", self.on\_menu\_click)  
 self.load\_menu()  
  
 def load\_menu(self, keyword=None):  
 self.menu\_tree.delete(\*self.menu\_tree.get\_children())  
 query = "SELECT dish\_id, dish\_name, price, profit\_per\_dish, description FROM menu"  
 params = ()  
 if keyword:  
 query += " WHERE dish\_name LIKE %s"  
 params = (f"%{keyword}%",)  
 data = self.db.query(query, params)  
 for row in data:  
 self.menu\_tree.insert("", tk.END, values=  
 (  
 row["dish\_id"], row["dish\_name"],  
 f"{row['price']:.2f}元", f"{row['profit\_per\_dish']:.2f}元",  
 row["description"], "编辑", "删除"  
 ))  
  
 def search\_dishes(self):  
 keyword = self.search\_entry.get().strip()  
 self.load\_menu(keyword if keyword else None)  
  
 def on\_menu\_click(self, event):  
 region = self.menu\_tree.identify\_region(event.x, event.y)  
 if region != "cell":  
 return  
 item = self.menu\_tree.selection()[0]  
 column = int(self.menu\_tree.identify\_column(event.x).replace('#', ''))  
 dish\_id = self.menu\_tree.item(item, "values")[0]  
 if column == 6:  
 self.edit\_dish(dish\_id)  
 elif column == 7:  
 self.delete\_dish(dish\_id)  
  
 def add\_dish(self):  
 win = tk.Toplevel(self.root)  
 win.title("添加菜品")  
 win.geometry("400x300")  
 win.resizable(False, False)  
  
 ttk.Label(win, text="菜品名称:").grid(row=0, column=0, padx=10, pady=10, sticky=tk.W)  
 name\_entry = ttk.Entry(win, width=30)  
 name\_entry.grid(row=0, column=1, padx=10, pady=10)  
  
 ttk.Label(win, text="价格(元):").grid(row=1, column=0, padx=10, pady=10, sticky=tk.W)  
 price\_entry = ttk.Entry(win, width=30)  
 price\_entry.grid(row=1, column=1, padx=10, pady=10)  
  
 ttk.Label(win, text="单份利润(元):").grid(row=2, column=0, padx=10, pady=10, sticky=tk.W)  
 profit\_entry = ttk.Entry(win, width=30)  
 profit\_entry.grid(row=2, column=1, padx=10, pady=10)  
  
 ttk.Label(win, text="描述:").grid(row=3, column=0, padx=10, pady=10, sticky=tk.NW)  
 desc\_text = tk.Text(win, width=30, height=5)  
 desc\_text.grid(row=3, column=1, padx=10, pady=10)  
  
 def save\_dish():  
 name = name\_entry.get()  
 price = price\_entry.get()  
 profit = profit\_entry.get()  
 desc = desc\_text.get("1.0", tk.END).strip()  
  
 if not name or not price or not profit:  
 messagebox.showerror("错误", "请填写必填字段")  
 return  
  
 try:  
 price = float(price)  
 profit = float(profit)  
 except ValueError:  
 messagebox.showerror("错误", "价格和利润必须为数字")  
 return  
  
 try:  
 self.db.execute(  
 "INSERT INTO menu (dish\_name, price, profit\_per\_dish, description) VALUES (%s, %s, %s, %s)",  
 (name, price, profit, desc)  
 )  
 messagebox.showinfo("成功", "菜品添加成功")  
 self.load\_menu()  
 win.destroy()  
 except Exception as e:  
 messagebox.showerror("错误", f"添加失败: {str(e)}")  
  
 ttk.Button(win, text="保存", command=save\_dish).grid(row=4, column=0, padx=10, pady=10)  
 ttk.Button(win, text="取消", command=win.destroy).grid(row=4, column=1, padx=10, pady=10)  
  
 def edit\_dish(self, dish\_id):  
 data = self.db.query("SELECT \* FROM menu WHERE dish\_id = %s", (dish\_id,))[0]  
  
 win = tk.Toplevel(self.root)  
 win.title("编辑菜品")  
 win.geometry("400x300")  
 win.resizable(False, False)  
  
 ttk.Label(win, text="菜品名称:").grid(row=0, column=0, padx=10, pady=10, sticky=tk.W)  
 name\_entry = ttk.Entry(win, width=30)  
 name\_entry.insert(0, data["dish\_name"])  
 name\_entry.grid(row=0, column=1, padx=10, pady=10)  
  
 ttk.Label(win, text="价格(元):").grid(row=1, column=0, padx=10, pady=10, sticky=tk.W)  
 price\_entry = ttk.Entry(win, width=30)  
 price\_entry.insert(0, str(data["price"]))  
 price\_entry.grid(row=1, column=1, padx=10, pady=10)  
  
 ttk.Label(win, text="单份利润(元):").grid(row=2, column=0, padx=10, pady=10, sticky=tk.W)  
 profit\_entry = ttk.Entry(win, width=30)  
 profit\_entry.insert(0, str(data["profit\_per\_dish"]))  
 profit\_entry.grid(row=2, column=1, padx=10, pady=10)  
  
 ttk.Label(win, text="描述:").grid(row=3, column=0, padx=10, pady=10, sticky=tk.NW)  
 desc\_text = tk.Text(win, width=30, height=5)  
 desc\_text.insert("1.0", data["description"])  
 desc\_text.grid(row=3, column=1, padx=10, pady=10)  
  
 def update\_dish():  
 name = name\_entry.get()  
 price = price\_entry.get()  
 profit = profit\_entry.get()  
 desc = desc\_text.get("1.0", tk.END).strip()  
  
 if not name or not price or not profit:  
 messagebox.showerror("错误", "请填写必填字段")  
 return  
  
 try:  
 price = float(price)  
 profit = float(profit)  
 except ValueError:  
 messagebox.showerror("错误", "价格和利润必须为数字")  
 return  
  
 try:  
 self.db.execute(  
 "CALL update\_menu\_price(%s, %s)",  
 (dish\_id, price)  
 )  
  
 self.db.execute(  
 """UPDATE menu   
 SET dish\_name = %s, profit\_per\_dish = %s, description = %s   
 WHERE dish\_id = %s""",  
 (name, profit, desc, dish\_id)  
 )  
  
 messagebox.showinfo("成功", "菜品更新成功")  
 self.load\_menu()  
 win.destroy()  
  
 except pymysql.Error as e:  
 if e.args[0] == 1644:  
 messagebox.showerror("价格错误", "价格必须大于0")  
 else:  
 messagebox.showerror("错误", f"更新失败: {str(e)}")  
 self.db.conn.rollback()  
  
 ttk.Button(win, text="保存", command=update\_dish).grid(row=4, column=0, padx=10, pady=10)  
 ttk.Button(win, text="取消", command=win.destroy).grid(row=4, column=1, padx=10, pady=10)  
  
 def delete\_dish(self, dish\_id):  
 if messagebox.askyesno("确认删除", "确定要删除此菜品吗？"):  
 try:  
 self.db.execute("DELETE FROM menu WHERE dish\_id = %s", (dish\_id,))  
 messagebox.showinfo("成功", "菜品已删除")  
 self.load\_menu()  
 except Exception as e:  
 messagebox.showerror("错误", f"删除失败: {str(e)}")  
  
 def init\_sales\_tab(self):  
 tab = ttk.Frame(self.notebook)  
 self.notebook.add(tab, text="销售统计")  
 ttk.Label(tab, text="菜品销售统计", font=("微软雅黑", 14), background="#4a7abc", foreground="white").pack(  
 pady=10, fill=tk.X, padx=10)  
  
 filter\_frame = ttk.Frame(tab)  
 filter\_frame.pack(fill=tk.X, pady=5)  
 ttk.Button(filter\_frame, text="按销量排序", command=lambda: self.load\_sales("total\_sales DESC")).pack(  
 side=tk.LEFT, padx=5)  
 ttk.Button(filter\_frame, text="按利润排序", command=lambda: self.load\_sales("total\_profit DESC")).pack(  
 side=tk.LEFT, padx=5)  
  
 self.sales\_tree = ttk.Treeview(tab, columns=("id", "name", "sales", "revenue", "profit"), show="headings")  
 self.sales\_tree.heading("id", text="ID")  
 self.sales\_tree.heading("name", text="名称")  
 self.sales\_tree.heading("sales", text="总销量")  
 self.sales\_tree.heading("revenue", text="总收入")  
 self.sales\_tree.heading("profit", text="总利润")  
  
 self.sales\_tree.column("id", width=50, anchor=tk.CENTER)  
 self.sales\_tree.column("name", width=150, anchor=tk.CENTER)  
 self.sales\_tree.column("sales", width=100, anchor=tk.CENTER)  
 self.sales\_tree.column("revenue", width=120, anchor=tk.CENTER)  
 self.sales\_tree.column("profit", width=120, anchor=tk.CENTER)  
  
 self.sales\_tree.pack(fill=tk.BOTH, expand=True)  
 self.load\_sales()  
  
 def load\_sales(self, order\_by="total\_sales DESC"):  
 self.sales\_tree.delete(\*self.sales\_tree.get\_children())  
 sql = """  
 SELECT   
 m.dish\_id,   
 m.dish\_name,   
 s.total\_sales,  
 s.total\_revenue,  
 s.total\_profit  
 FROM menu m   
 JOIN sales\_statistics s ON m.dish\_id = s.dish\_id  
 ORDER BY {order\_by}  
 """  
 data = self.db.query(sql.format(order\_by=order\_by))  
 for row in data:  
 self.sales\_tree.insert("", tk.END, values=(  
 row["dish\_id"],  
 row["dish\_name"],  
 row["total\_sales"],  
 f"{row['total\_revenue']:.2f}元",  
 f"{row['total\_profit']:.2f}元"  
 ))  
  
 def init\_profit\_tab(self):  
 tab = ttk.Frame(self.notebook)  
 self.notebook.add(tab, text="总利润统计")  
 ttk.Label(tab, text="餐厅总利润", font=("微软雅黑", 14), background="#4a7abc", foreground="white").pack(pady=10,fill=tk.X,padx=10)  
  
 profit\_frame = ttk.Frame(tab)  
 profit\_frame.pack(pady=20)  
 ttk.Label(profit\_frame, text="当前总利润:", font=("微软雅黑", 12)).pack(side=tk.LEFT, padx=10)  
 self.profit\_lbl = ttk.Label(profit\_frame, text="0.00元", font=("微软雅黑", 18, "bold"))  
 self.profit\_lbl.pack(side=tk.LEFT, padx=10)  
  
 ttk.Button(tab, text="刷新", command=self.refresh\_profit).pack(pady=10)  
 self.refresh\_profit()  
  
 def refresh\_profit(self):  
 data = self.db.query("SELECT total\_profit FROM total\_restaurant\_profit")  
 if data and data[0]['total\_profit'] is not None:  
 self.profit\_lbl.config(text=f"{data[0]['total\_profit']:.2f}元")  
 else:  
 self.profit\_lbl.config(text="0.00元")  
  
 def init\_orders\_tab(self):  
 tab = ttk.Frame(self.notebook)  
 self.notebook.add(tab, text="订单管理")  
  
 ttk.Label(tab, text="订单列表", font=("微软雅黑", 14), background="#4a7abc", foreground="white").pack(pady=10, fill=tk.X, padx=10)  
 filter\_frame = ttk.Frame(tab)  
 filter\_frame.pack(fill=tk.X, pady=5)  
 ttk.Button(filter\_frame, text="刷新", command=self.load\_orders).pack(side=tk.LEFT, padx=5)  
 self.orders\_tree = ttk.Treeview(tab, columns=("id", "time", "table", "amount"), show="headings")  
 self.orders\_tree.heading("id", text="订单ID")  
 self.orders\_tree.heading("time", text="下单时间")  
 self.orders\_tree.heading("table", text="桌号")  
 self.orders\_tree.heading("amount", text="总金额")  
 self.orders\_tree.column("id", width=80, anchor=tk.CENTER)  
 self.orders\_tree.column("time", width=180, anchor=tk.CENTER)  
 self.orders\_tree.column("table", width=80, anchor=tk.CENTER)  
 self.orders\_tree.column("amount", width=100, anchor=tk.E)  
 self.orders\_tree.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)  
 self.load\_orders()  
  
 def load\_orders(self):  
 self.orders\_tree.delete(\*self.orders\_tree.get\_children())  
 query = """  
 SELECT   
 o.order\_id,   
 MIN(o.order\_time) AS order\_time,  
 o.table\_id,   
 SUM(o.subtotal) AS total\_amount  
 FROM orders o  
 GROUP BY o.order\_id, o.table\_id  
 ORDER BY order\_time DESC  
 """  
 data = self.db.query(query)  
 for row in data:  
 self.orders\_tree.insert("", tk.END, values=(  
 row["order\_id"],  
 row["order\_time"].strftime("%Y-%m-%d %H:%M:%S"),  
 row["table\_id"],  
 f"{row['total\_amount']:.2f}元"  
 ))  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = AdminApp(root)  
 root.mainloop()