Testing code

Code

```
while True:
    # Use EasyGui's buttonbox to create a main menu for the user.
    # This function returns the text of the button that was clicked.
    choice = eg.buttonbox(
        "What would you like to do?",
        "Main Menu",
        choices=["Add Customer", "Add Order", "Show Customers", "Show Orders", "Exit"]
    )
```

What it does

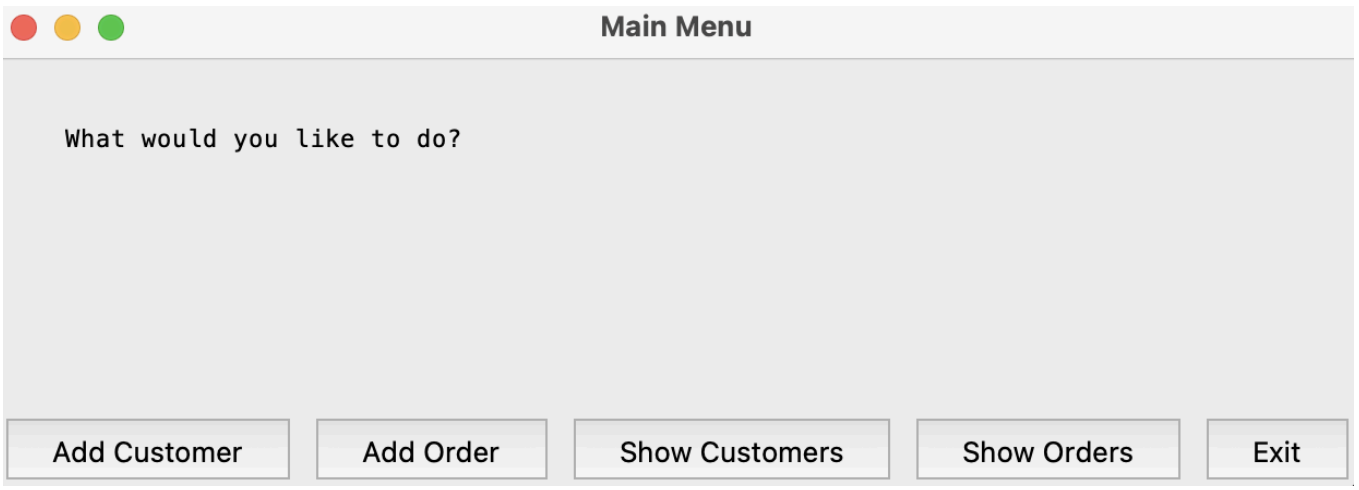
This block of code creates a loop that sets the user into a constant loop which can't be escaped unless you press the exit button or the X at the top corner, and it uses easygui to create a clean, tidy look.

Intended outcome

The intended outcome is for boxes to appear with fixed options, each one should take you to a different box, and if pressed, exit, it should exit the menu.

Actual outcome/evidence

In the screenshot below, we can clearly see that a box has appeared with three options to choose from.



Code

```
    """
    Prompts the user for a new customer's details and inserts them into the Customers table.
    """
    msg = "Enter new customer information"
    title = "Add Customer"
    fieldNames = ["customer_id", "first_name", "last_name"]

    fieldValues = eg.multenterbox(msg, title, fieldNames)
```

What it does

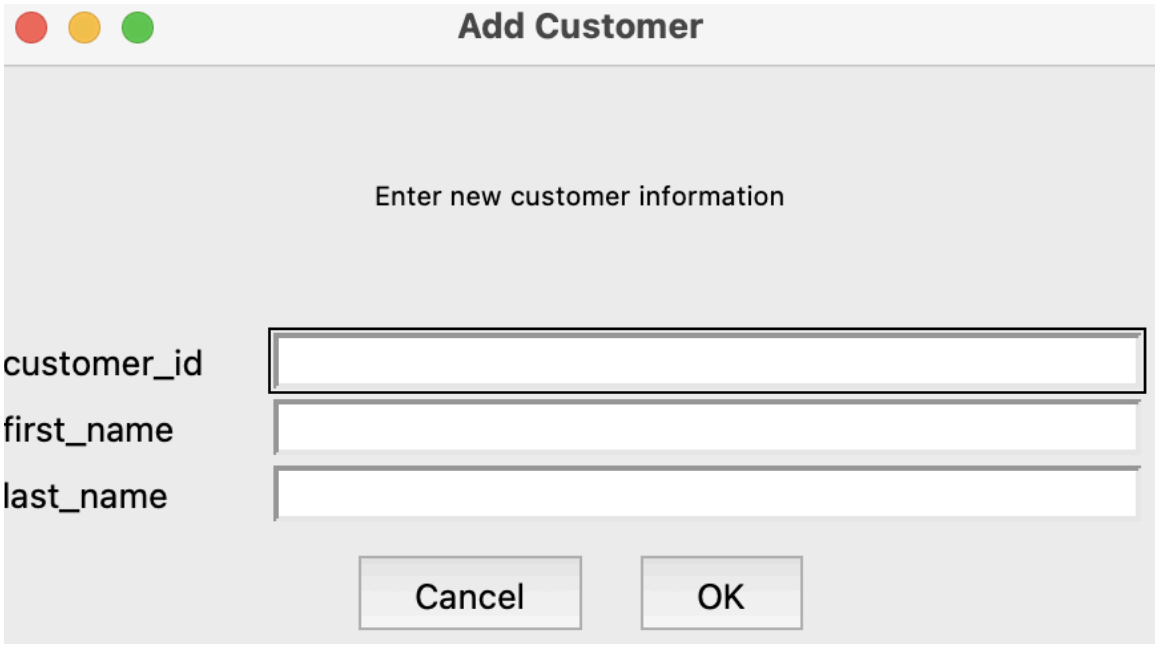
This block of code asks the user to enter the details of the customer, such as customer ID, first name, and last name. It also collects user input.

Intended outcome

After the user has pressed the Add order button, another box should appear that asks the user for user_input for the details of the customer they want to add.

Actual outcome/evidence

After pressing the add customer button, I was met with another box that asked for the details of the customer I wanted to add.



Code

```
# Gets all rows from the Customers table and stores them in 'rows'
cursor.execute("SELECT * FROM Customers")
rows = cursor.fetchall()

# If no customers are found, show a message and exit the function
if not rows:
    eg.msgbox("No customers found.")
    return
```

What it does

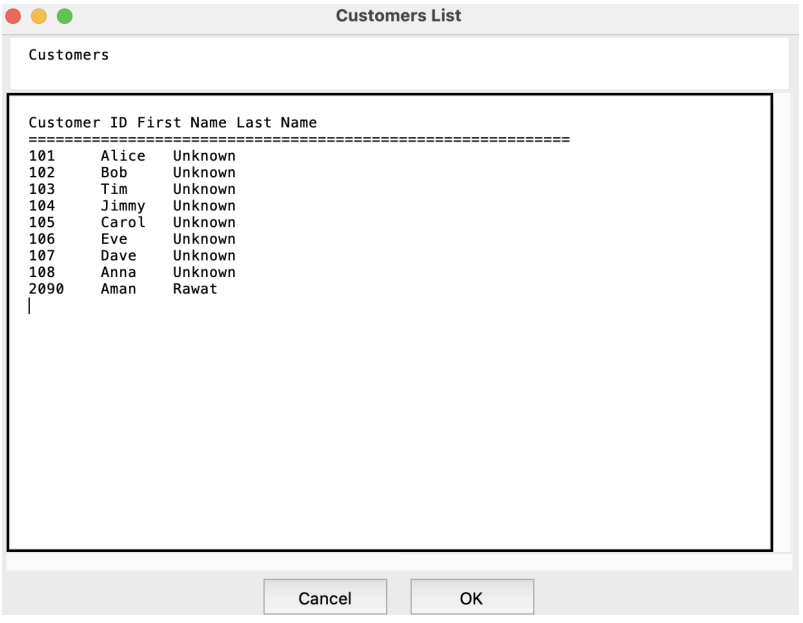
This code selects all the records from the Customers table and stores them in a list called rows. If the list is empty (meaning there are no customers in the database), it shows a message saying "No customers found," and then stops running the rest of the function.

Intended outcome

The aim is for the program to successfully load all customers from the database. If the table has no customers, it should warn the user by showing a message instead of continuing.

Actual outcome/evidence

In the table below we see that the program has successfully loaded all customers from the database.



Code

```
    """
    Prompts the user for a new order's details and inserts them into the order table.
    """
    msg = "Enter new order details"
    title = "Add Order"
    fieldNames = ["order_id", "customer_id", "order_date", "total_amount"]

    fieldValues = eg.multenterbox(msg, title, fieldNames)
```

What it does

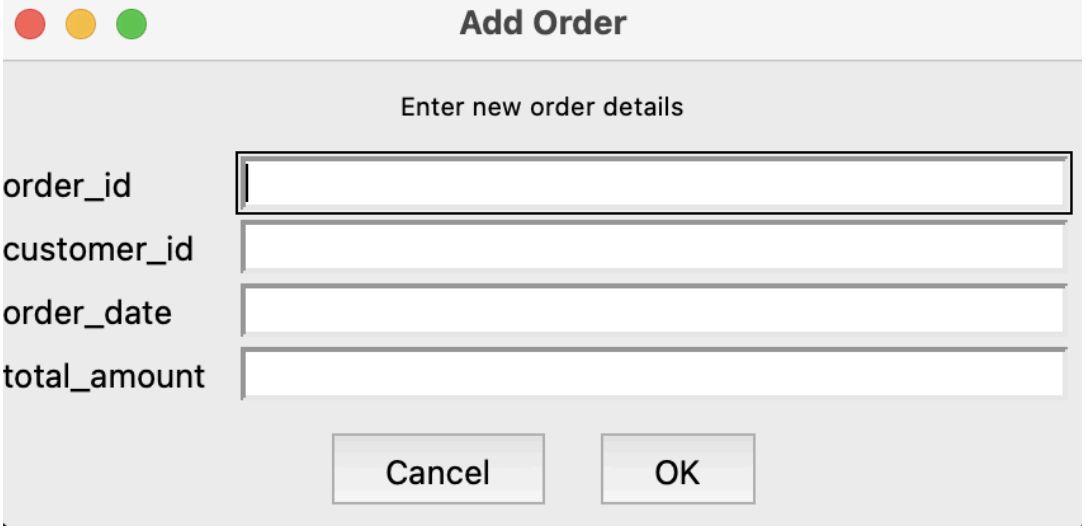
This block of code asks the user to enter the details of the order that is being placed, such as order ID, customer ID, order date and total amount. It also collects user input.

Intended outcome

After the user has pressed the Add customer button, another box should appear that asks the user for user_input for the details of the order they want to add.

Actual outcome/evidence

After pressing the add order button, I was met with another box that asked for the details of the order I wanted to add.



Code

```
# Gets all rows from the Orders table and stores them in 'rows'
cursor.execute("SELECT * FROM Orders")
rows = cursor.fetchall()

# If no orders are found, show a message and exit the function
if not rows:
    eg.msgbox("No orders found.")
    return
```

What it does

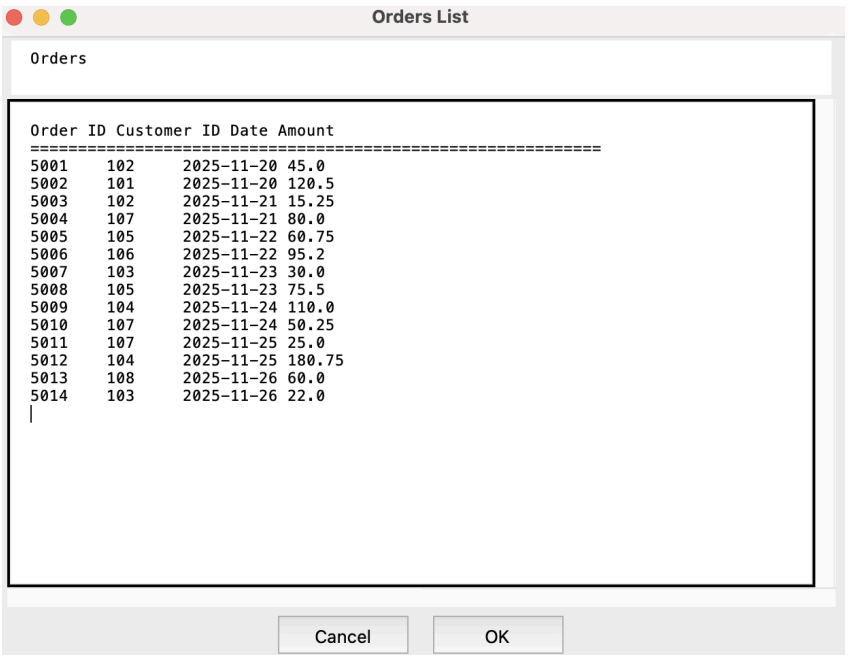
This code selects all the records from the order table and stores them in a list called rows. If the list is empty (meaning there are no order in the database), it shows a message saying "No orders found," and then stops running the rest of the function.

Intended outcome

The aim is for the program to successfully load all orders from the database. If the table has no customers, it should warn the user by showing a message instead of continuing.

Actual outcome/evidence

In the table below, we see that the program has successfully loaded all orders from the database.



Code

```
        # The loop breaks if the user clicks 'Exit' or closes the window
('None').
        else:
            break

    # Closes the database connection when the program's main loop ends.
    conn.close()
    eg.msgbox("Goodbye!", "Exiting Program")
```

What it does

When the user presses exit, it stops the loop, closes the data connection, and then shows a pop up message saying goodbye.

Intended outcome

The intended outcome for this code is for another box to appear when the user clicks, exist. Another message should appear saying goodbye, closing the program.

Actual outcome/evidence

When I pressed exit, another message appeared for the last time, saying goodbye. Then I pressed ok, and the program closed.