

Lab 12: SQL Database Connectivity

CS355/CE373 Database Systems
Fall 2024



Dhanani School of Science and Engineering

Habib University

Copyright © 2024 Habib University

Contents

1	Instructions	2
1.1	Marking scheme	2
1.2	Late submission policy	2
2	Objective	2
3	Exercise	2
4	Pyodbc example for CRUD operations	3
5	Skeleton File for Lab Task	6

1 Instructions

- This lab will contribute 1% towards the final grade.
- The deadline for this project is the end of your lab.
- The lab must be submitted online via CANVAS. You are required to submit a zip file that contains both *.py* and *.ui* files.
- The zip file should be named as *Lab_12_aa1234.zip* where *aa1234* will be replaced with your student id.
- **Files that don't follow the appropriate naming convention will not be graded.**

1.1 Marking scheme

This lab will be marked out of 100.

- 50 Marks are for the completion of the lab.
- 50 Marks are for progress and attendance during the lab.

1.2 Late submission policy

You can submit late till 11:59 PM on the same day as your lab with a 20% penalty. No submissions will be accepted afterward.

2 Objective

The objective of this lab is to provide hands-on practice on connecting your desktop application in developed in PyQt6 to the Northwind Database using the **pyodbc** library.

3 Exercise

In this lab, your task is to create a desktop applications that allows you to insert new orders and view existing orders based on the Northwind Database. You have been skeleton code files and your task to complete the implementations of the functions specified below.

	OrderID	CustomerID	EmployeeID	OrderDate	RequiredDate	ShippedDate	ShipVia	Freight	ShipName	ShipAddress	ShipCity	Ship
1	10248	VINET	5	1996-07-04 ...	1996-08-01 ...	1996-07-16 ...	3	32.3800	Vins et alcools ...	59 rue de ...	Reims	None
2	10249	TOMSP	6	1996-07-05 ...	1996-08-16 ...	1996-07-10 ...	1	11.6100	Toms ...	Luisenstr. 48	Münster	None
3	10250	HANAR	4	1996-07-08 ...	1996-08-05 ...	1996-07-12 ...	2	65.8300	Hanari Carnes	Rua do Paço, 67	Rio de Janeiro	RJ
4	10251	VICTE	3	1996-07-08 ...	1996-08-05 ...	1996-07-15 ...	1	41.3400	Victuailles en ...	2, rue du ...	Lyon	None
5	10252	SUPRD	4	1996-07-09 ...	1996-08-06 ...	1996-07-11 ...	2	51.3000	Suprêmes délices	Boulevard Tirou...	Charleroi	None
6	10253	HANAR	3	1996-07-10 ...	1996-07-24 ...	1996-07-16 ...	2	58.1700	Hanari Carnes	Rua do Paço, 67	Rio de Janeiro	RJ
7	10254	CHOPS	5	1996-07-11 ...	1996-08-08 ...	1996-07-23 ...	2	22.9800	Chop-suey ...	Hauptstr. 31	Bern	None
8	10255	RICSU	9	1996-07-12 ...	1996-08-09 ...	1996-07-15 ...	3	148.3300	Richter ...	Starenweg 5	Genève	None
9	10256	WELLI	3	1996-07-15 ...	1996-08-12 ...	1996-07-17 ...	2	13.9700	Wellington ...	Rua do Mercad...	Resende	SP
10	10257	HILAA	4	1996-07-16 ...	1996-08-13 ...	1996-07-22 ...	3	81.9100	HILARION-...	Carrera 22 con ...	San Cristóbal	Táchira
11	10258	ERNSH	1	1996-07-17 ...	1996-08-14 ...	1996-07-23 ...	1	140.5100	Ernst Handel	Kirchgasse 6	Graz	None
12	10259	CENTC	4	1996-07-18 ...	1996-08-15 ...	1996-07-25 ...	3	3.2500	Centro comerci...	Sierras de ...	México D.F.	None

Insert New Order

Figure 1: Homepage

The functions that you are required to implement are as follows:

1. `populate_table` which populates the `orderTable` with data from the Northwind database.
2. `populate_employee_table` which populates the `employeesTable` with employee data from the Northwind database.
3. `populate_customer_table` which populates the `customersTable` with customer data from the Northwind database.
4. `populate_ship_table` which populates the `shippersTable` with shippers data from the Northwind database.
5. `populate_product_table` which populates the `productsTable` with products data from the Northwind database.
6. `insert_order` which inserts a new order into the Northwind database based on user input.
7. `insert_order_details` which inserts a order details of a new order into the Northwind database based on user input.

In order to implement these functions, you can refer to the code provided in the two sections below.

4 Pyodbc example for CRUD operations

```

1 import pyodbc
2
3 # Replace these with your own database connection details
4 server = 'your_server_name'
5 database = 'Northwind' # Name of your Northwind database
6 use_windows_authentication = True # Set to True to use Windows Authentication
7 username = 'your_username' # Specify a username if not using Windows
8 password = 'your_password' # Specify a password if not using Windows
9
10 # Create the connection string based on the authentication method chosen

```

Employees

EmployeeID	FirstName	LastName
1	Nancy	Davolio
2	Andrew	Fuller
3	Janet	Leverling
4	Margaret	Peacock
5	Steven	Buchanan
6	Michael	Suyama

EmployeeID:
EmployeeName:

Customers

CustomerID	CustomerName
1	Alfreds Futterkiste
2	Ana Trujillo Emparedados y helados
3	Antonio Moreno Taquería
4	Around the Horn
5	Berglunds snabbköp
6	Blaauw Goud

CustomerID:
CustomerName:

Products

ProductID	ProductName	ProductPrice
1	Chai	18.0000
2	Chang	19.0000
3	Aniseed Syrup	10.0000

ProductID:
Quantity:
UnitPrice:
Discount:

Shippers

ShipperID	CompanyName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

ShipCompanyID:
ShipAddress:
ShipCompany:
ShipCity:
ShipName:
ShipRegion:
Freight:
ShipPostalCode:
ShippedDate:
ShipCountry:

Order Details

OrderID	ProductID	UnitPrice	Quantity	Discount
1	1	10	10	0.1

OrderDate:
RequiredDate:

Figure 2: Insert New Order Form

Order Inserted

Order ID: 11086 has been inserted successfully.

Figure 3: Prompt upon successful creation of a new order.

```

12 if use_windows_authentication:
13     connection_string = f'DRIVER={{ODBC Driver 17 for SQL Server}};SERVER={
        server};DATABASE={database};Trusted_Connection=yes;'
14 else:
15     connection_string = f'DRIVER={{ODBC Driver 17 for SQL Server}};SERVER={
        server};DATABASE={database};UID={username};PWD={password}'
16
17 # Establish a connection to the database
18 connection = pyodbc.connect(connection_string)
19
20 # Create a cursor to interact with the database
21 cursor = connection.cursor()
22
23 # CREATE - Insert a new employee record
24 new_employee = (
25     'Doe',
26     'John',
27     'Sales Manager',
28     'Mr.',
29     '1975-01-15',
30     '2023-10-06',
31     '123 Main St',

```

```

32     'New York',
33     'NY',
34     '10001',
35     'USA',
36     '555-123-4567',
37     '1234',
38     None, # You can insert the binary image data for the "Photo" field here if
           # needed
39     'Notes about John Doe',
40     2, # Replace with the actual ReportsTo value if applicable
41     'images/johndoe.jpg' # Update with the actual file path for the "PhotoPath
    " field
42 )
43 insert_query = """
44     INSERT INTO Employees
45     ([LastName], [FirstName], [Title], [TitleOfCourtesy], [BirthDate], [
    HireDate],
46     [Address], [City], [Region], [PostalCode], [Country], [HomePhone], [
    Extension],
47     [Photo], [Notes], [ReportsTo], [PhotoPath])
48     VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
49 """
50 cursor.execute(insert_query, new_employee)
51 connection.commit() # Commit the transaction
52
53 # READ - Fetch and print all employees
54 select_query = "SELECT FirstName, LastName FROM Employees"
55 cursor.execute(select_query)
56 print("All Employees:")
57 for row in cursor.fetchall():
58     print(row)
59
60 # UPDATE - Update an employee record
61 update_query = "UPDATE Employees SET Region = ? WHERE FirstName = ? AND
    LastName = ?"
62 updated_region = 'WA'
63 cursor.execute(update_query, (updated_region, 'John', 'Doe'))
64 connection.commit() # Commit the transaction
65
66 # READ - Fetch and print the updated employee
67 cursor.execute("select FirstName, LastName, Region from Employees WHERE
    FirstName = ? AND LastName = ?", ('John', 'Doe'))
68 print("\nUpdated Employee:")
69 for row in cursor.fetchall():
70     print(row)
71
72 # DELETE - Delete an employee record
73 delete_query = "DELETE FROM Employees WHERE FirstName = ? AND LastName = ?"
74 cursor.execute(delete_query, ('John', 'Doe'))
75 connection.commit() # Commit the transaction
76
77 # READ - Fetch and print all employees after deletion
78 cursor.execute(select_query)
79 print("\nEmployees After Deletion:")
80 for row in cursor.fetchall():
81     print(row)
82
83 # Close the cursor and connection
84 cursor.close()
85 connection.close()

```

Listing 1: Skeleton Python file (example.py)

5 Skeleton File for Lab Task

```
1 # Importing essential modules
2 from PyQt6 import QtWidgets, uic
3 from PyQt6.QtCore import QDate
4 from PyQt6.QtWidgets import QApplication, QMainWindow, QTableWidgetItem,
    QTableWidgetItem, QVBoxLayout, QWidget, QHeaderView
5 import sys
6 import pyodbc
7
8
9 # Main Window Class
10 class UI(QtWidgets.QMainWindow):
11     def __init__(self):
12         """
13         Initialize the main UI window.
14
15         This constructor is called when an instance of the UI class is created.
16         It performs the following tasks:
17         1. Calls the constructor of the inherited class.
18         2. Loads the user interface (UI) from the 'MainWindow.ui' file.
19         3. Populates the 'orderTable' with data.
20         4. Connects the "Insert Order" button to the event handler for opening
the
21         master transaction form.
22
23         Note:
24         - The 'MainWindow.ui' file should exist and contain the required UI
elements.
25
26         Returns:
27         None
28         """
29         # Call the inherited classes __init__ method
30         super(UI, self).__init__()
31
32         # Load the .ui file
33         uic.loadUi('MainWindow.ui', self)
34
35         # Load Orders data
36         self.populate_table()
37
38         # Connect Submit Button to Event Handling Code
39         self.InsertOrder.clicked.connect(self.open_master_form)
40
41     def populate_table(self):
42         """
43         Populates the 'orderTable' with data from the Northwind database.
44
45         This function connects to the Northwind database, retrieves orders data
,
46         and populates the 'orderTable' widget with the fetched data. It also
adjusts
47         the column widths for better content display.
48
49         Note:
50         - Ensure that the 'orderTable' widget is set up and available in the UI
.
51         - The database connection parameters (server, database, authentication)
should
52         be correctly configured.
53
54         Returns:
```

```

55     None
56     """
57     # TODO: Provide the connection string to connect to the Northwind
database
58     connection = pyodbc.connect(
59         ""
60     )
61
62     cursor = connection.cursor()
63
64     # TODO: Write SQL query to fetch orders data
65     cursor.execute("")
66
67     # Fetch all rows and populate the table
68     for row_index, row_data in enumerate(cursor.fetchall()):
69         self.orderTable.insertRow(row_index)
70         for col_index, cell_data in enumerate(row_data):
71             item = QTableWidgetItem(str(cell_data))
72             self.orderTable.setItem(row_index, col_index, item)
73
74     # Close the database connection
75     connection.close()
76
77     # Adjust content display
78     header = self.orderTable.horizontalHeader()
79     header.setSectionResizeMode(0, QHeaderView.ResizeMode.Stretch)
80     header.setSectionResizeMode(1, QHeaderView.ResizeMode.ResizeToContents)
81     header.setSectionResizeMode(2, QHeaderView.ResizeMode.ResizeToContents)
82
83     def open_master_form(self):
84         """
85         Opens the master transaction form.
86
87         This function is called when the "Insert Order" button is clicked in
the main
88         UI. It creates and displays the master transaction form (Master class).
89
90         Note:
91         - Ensure that the Master class (master_form) is defined and available
in the script.
92
93         Returns:
94         None
95         """
96         self.master_form = Master()
97         self.master_form.show()
98
99
100 class Master(QtWidgets.QMainWindow):
101     def __init__(self):
102         """
103         Initialize the master transaction form.
104
105         This constructor is called when an instance of the Master class is
created.
106         It performs the following tasks:
107         1. Calls the constructor of the inherited class.
108         2. Loads the user interface (UI) from the 'MasterTransactionForm.ui'
file.
109         3. Populates employee, customer, shipper, and product tables with data.
110         4. Sets up event handlers for table selection.
111         5. Makes certain input fields read-only or disabled.
112         6. Connects buttons like "Add," "Insert," and "Clear" to their

```



```

113     respective
114         event handling functions.
115
116     Note:
117     - The 'MasterTransactionForm.ui' file should exist and contain the
118       required UI elements.
119
120     Returns:
121     None
122     """
123
124     # Call the inherited classes __init__ method
125     super(Master, self).__init__()
126
127     # Load the .ui file
128     uic.loadUi('MasterTransactionForm.ui', self)
129
130     # Load employee table
131     self.populate_employee_table()
132
133     # Load customer table
134     self.populate_customer_table()
135
136     # Load shippers table
137     self.populate_ship_table()
138
139     # Load products table
140     self.populate_product_table()
141
142     # Employee Table Row selected
143     self.employeesTable.itemSelectionChanged.connect(
144         self.get_selected_employee_data)
145
146     # Customer Table Selected
147     self.customersTable.itemSelectionChanged.connect(
148         self.get_selected_customer_data)
149
150     # Product Table Selected
151     self.productsTable.itemSelectionChanged.connect(
152         self.get_selected_product_data)
153
154     # Shipper Table Selected
155     self.shippersTable.itemSelectionChanged.connect(
156         self.get_selected_shipper_data)
157
158     # Make EmployeeID and EmployeeName readonly
159     self.EmployeeID.setDisabled(True)
160     self.EmployeeName.setDisabled(True)
161
162     # Make CustomerID and CustomerName readonly
163     self.CustomerID.setDisabled(True)
164     self.CustomerName.setDisabled(True)
165
166     # Make ProductID and ProductName readonly
167     self.ProductID.setDisabled(True)
168     self.ProductName.setDisabled(True)
169
170     # Make ShipCompanyID and ShipCompany readonly
171     self.ShipCompanyID.setDisabled(True)
172     self.ShipCompany.setDisabled(True)
173
174     # Add Row to Product Table
175     self.Add.clicked.connect(self.add_product)

```

```

174         # Insert Product
175         self.Insert.clicked.connect(self.insert_order)
176
177         # Clear
178         self.Clear.clicked.connect(self.clear)
179
180     def populate_employee_table(self):
181         """
182         Populates the 'employeesTable' with employee data from the Northwind
183         database.
184
185         This function connects to the Northwind database, retrieves employee
186         data,
187         and populates the 'employeesTable' widget with the fetched data. It
188         also adjusts
189         the column widths for better content display.
190
191         Note:
192         - Ensure that the 'employeesTable' widget is set up and available in
193         the UI.
194         - The database connection parameters (server, database, authentication)
195         should
196         be correctly configured.
197
198         Returns:
199         None
200         """
201         # TODO: Provide the connection string to connect to the Northwind
202         database
203         connection = pyodbc.connect(
204             ""
205         )
206
207         cursor = connection.cursor()
208
209         # TODO: Write SQL query to fetch employee data
210         cursor.execute("")
211
212         # Fetch all rows and populate the table
213         for row_index, row_data in enumerate(cursor.fetchall()):
214             self.employeesTable.insertRow(row_index)
215             for col_index, cell_data in enumerate(row_data):
216                 item = QTableWidgetItem(str(cell_data))
217                 self.employeesTable.setItem(row_index, col_index, item)
218
219         # Close the database connection
220         connection.close()
221
222         # Adjust content display
223         header = self.employeesTable.horizontalHeader()
224         header.setSectionResizeMode(0, QHeaderView.ResizeMode.Stretch)
225         header.setSectionResizeMode(1, QHeaderView.ResizeMode.ResizeToContents)
226         header.setSectionResizeMode(2, QHeaderView.ResizeMode.ResizeToContents)
227
228     def populate_customer_table(self):
229         """
230         Populates the 'customersTable' with customer data from the Northwind
231         database.
232
233         This function connects to the Northwind database, retrieves customer
234         data,
235         and populates the 'customersTable' widget with the fetched data. It
236         also adjusts

```

```

228         the column widths for better content display.
229
230     Note:
231     - Ensure that the 'customersTable' widget is set up and available in
the UI.
232     - The database connection parameters (server, database, authentication)
should
233     be correctly configured.
234
235     Returns:
236     None
237     """
238     # TODO: Provide the connection string to connect to the Northwind
database
239     connection = pyodbc.connect(
240         ""
241     )
242
243     cursor = connection.cursor()
244
245     # TODO: Write SQL query to fetch customers data
246     cursor.execute("")
247
248     # Fetch all rows and populate the table
249     for row_index, row_data in enumerate(cursor.fetchall()):
250         self.customersTable.insertRow(row_index)
251         for col_index, cell_data in enumerate(row_data):
252             item = QTableWidgetItem(str(cell_data))
253             self.customersTable.setItem(row_index, col_index, item)
254
255     # Close the database connection
256     connection.close()
257
258     # Adjust content display
259     header = self.customersTable.horizontalHeader()
260     header.setSectionResizeMode(0, QHeaderView.ResizeMode.Stretch)
261     header.setSectionResizeMode(1, QHeaderView.ResizeMode.ResizeToContents)
262     header.setSectionResizeMode(2, QHeaderView.ResizeMode.ResizeToContents)
263
264     def populate_ship_table(self):
265         """
266         Populate the shippers table with data from the Northwind database.
267
268         This function connects to the Northwind database, executes an SQL query
to fetch
269         data from the Shippers table, and populates the shippersTable widget
with the
270         retrieved data. It also adjusts the column widths for better display.
271
272         Returns:
273         None
274         """
275         # TODO: Provide the connection string to connect to the Northwind
database
276         connection = pyodbc.connect(
277             ""
278         )
279
280         cursor = connection.cursor()
281
282         # TODO: Write SQL query to fetch customers data
283         cursor.execute("")
284

```

```

285     # Fetch all rows and populate the table
286     for row_index, row_data in enumerate(cursor.fetchall()):
287         self.shippersTable.insertRow(row_index)
288         for col_index, cell_data in enumerate(row_data):
289             item = QTableWidgetItem(str(cell_data))
290             self.shippersTable.setItem(row_index, col_index, item)
291
292     # Close the database connection
293     connection.close()
294
295     # Adjust the column widths for better display
296     header = self.shippersTable.horizontalHeader()
297     header.setSectionResizeMode(0, QHeaderView.ResizeMode.Stretch)
298     header.setSectionResizeMode(1, QHeaderView.ResizeMode.ResizeToContents)
299     header.setSectionResizeMode(2, QHeaderView.ResizeMode.ResizeToContents)
300
301 def populate_product_table(self):
302     """
303     Populate the products table with data from the Northwind database.
304
305     This function connects to the Northwind database, executes an SQL query
306     to fetch
307     specific data (ProductID, ProductName, UnitPrice) from the Products
308     table, and
309     populates the productsTable widget with the retrieved data. It also
310     adjusts
311     the column widths for better display.
312
313     Returns:
314     None
315     """
316     # TODO: Provide the connection string to connect to the Northwind
317     database
318     connection = pyodbc.connect(
319
320     )
321
322     cursor = connection.cursor()
323
324     # TODO: Write SQL query to fetch customers data
325     cursor.execute("")
326
327     # Fetch all rows and populate the table
328     for row_index, row_data in enumerate(cursor.fetchall()):
329         self.productsTable.insertRow(row_index)
330         for col_index, cell_data in enumerate(row_data):
331             item = QTableWidgetItem(str(cell_data))
332             self.productsTable.setItem(row_index, col_index, item)
333
334     # Close the database connection
335     connection.close()
336
337     # Adjust the column widths for better display
338     header = self.productsTable.horizontalHeader()
339     header.setSectionResizeMode(0, QHeaderView.ResizeMode.Stretch)
340     header.setSectionResizeMode(1, QHeaderView.ResizeMode.ResizeToContents)
341     header.setSectionResizeMode(2, QHeaderView.ResizeMode.ResizeToContents)
342
343 def get_selected_employee_data(self):
344     """
345     Retrieve and display information of the selected employee.
346
347     This function gets the index of the selected row in the employees table

```

```

344     ,
345     retrieves the Employee ID, First Name, and Last Name of the selected
employee,
346     and displays this information in the corresponding input fields.
347
348     Returns:
349     None
350     """
351     # Get the index of the selected row
352     selected_row = self.employeesTable.currentRow()
353     # Employee ID
354     EmployeeID = self.employeesTable.item(selected_row, 0).text()
355     # First Name
356     FirstName = self.employeesTable.item(selected_row, 1).text()
357     # Last Name
358     LastName = self.employeesTable.item(selected_row, 2).text()
359     # Set Employee ID
360     self.EmployeeID.setText(EmployeeID)
361     # Set Employee Name
362     self.EmployeeName.setText(FirstName + " " + LastName)
363
364     def get_selected_customer_data(self):
365         """
366         Retrieve and display information of the selected customer.
367
368         This function gets the index of the selected row in the customers table
369         ,
370         retrieves the Customer ID and Company Name of the selected customer,
371         and
372         displays this information in the corresponding input fields.
373
374         Returns:
375         None
376         """
377         # Get the index of the selected row
378         selected_row = self.customersTable.currentRow()
379         # Customer ID
380         customersID = self.customersTable.item(selected_row, 0).text()
381         # Company Name
382         customersName = self.customersTable.item(selected_row, 1).text()
383         # Set Customer ID
384         self.CustomerID.setText(customersID)
385         # Set Customer Name
386         self.CustomerName.setText(customersName)
387
388     def get_selected_product_data(self):
389         """
390         Retrieve and display information of the selected product.
391
392         This function gets the index of the selected row in the products table,
393         retrieves the Product ID and Product Name of the selected product, and
394         displays this information in the corresponding input fields.
395
396         Returns:
397         None
398         """
399         # Get the index of the selected row
400         selected_row = self.productsTable.currentRow()
401         # Product ID
402         ProductID = self.productsTable.item(selected_row, 0).text()
403         # Product Name
404         ProductName = self.productsTable.item(selected_row, 1).text()
405         # Set Product ID

```

```

403         self.ProductID.setText(ProductID)
404         # Set Product Name
405         self.ProductName.setText(ProductName)
406
407     def get_selected_shipper_data(self):
408         """
409         Retrieve and display information of the selected shipper.
410
411         This function gets the index of the selected row in the shippers table,
412         retrieves the Ship Company ID and Ship Company Name of the selected
413         shipper,
414         and displays this information in the corresponding input fields.
415
416         Returns:
417         None
418         """
419         # Get the index of the selected row
420         selected_row = self.shippersTable.currentRow()
421         # Ship Company ID
422         ShipCompanyID = self.shippersTable.item(selected_row, 0).text()
423         # Ship Company Name
424         ShipCompanyName = self.shippersTable.item(selected_row, 1).text()
425         # Set ShipCompany ID
426         self.ShipCompanyID.setText(ShipCompanyID)
427         # Set ShipCompany Name
428         self.ShipCompany.setText(ShipCompanyName)
429
430     def add_product(self):
431         """
432         Add a product to the order details table.
433
434         This function adds a new row to the order details table in the user
435         interface
436         and populates it with product information entered by the user, such as
437         Product ID, Unit Price, Quantity, and Discount. It also adjusts the
438         column
439         widths for proper display and clears the input fields for the next
440         entry.
441
442         Note:
443         - Ensure that the relevant input fields are correctly configured in the
444         UI.
445
446         Returns:
447         None
448         """
449         row_position = self.orderDetailsTable.rowCount()
450         self.orderDetailsTable.insertRow(row_position)
451
452         self.orderDetailsTable.setItem(
453             row_position, 0, QTableWidgetItem(self.ProductID.text()))
454         self.orderDetailsTable.setItem(
455             row_position, 1, QTableWidgetItem(self.UnitPrice.text()))
456         self.orderDetailsTable.setItem(
457             row_position, 2, QTableWidgetItem(self.Quantity.text()))
458         self.orderDetailsTable.setItem(
459             row_position, 3, QTableWidgetItem(self.Discount.text()))
460
461         header = self.orderDetailsTable.horizontalHeader()
462         header.setSectionResizeMode(0, QHeaderView.ResizeMode.Stretch)
463         header.setSectionResizeMode(1, QHeaderView.ResizeMode.ResizeToContents)
464         header.setSectionResizeMode(2, QHeaderView.ResizeMode.ResizeToContents)

```

```

461         self.ProductID.setText("")
462         self.ProductName.setText("")
463         self.UnitPrice.setText("")
464         self.Quantity.setText("")
465         self.Discount.setText("")
466
467     def insert_order(self):
468         """
469         Insert a new order into the Northwind database based on user input.
470
471         This function retrieves order information from various input fields,
472         such as
473         Customer ID, Employee ID, shipping details, and others. It constructs
474         an SQL
475         query with parameters to insert a new order into the 'Orders' table of
476         the
477         Northwind database. After successfully inserting the order, it
478         retrieves the
479         newly assigned Order ID and displays it in a message box.
480
481         Note:
482         - Ensure that the relevant input fields are correctly configured in the
483         UI.
484         - The function assumes that the database connection is already
485         established.
486
487         Returns:
488         None
489         """
490         # Get order information from input fields
491         CustomerID = self.CustomerID.text()
492         EmployeeID = self.EmployeeID.text()
493         OrderDate = self.OrderDate.date().toString("yyyy-MM-dd")
494         RequiredDate = self.RequiredDate.date().toString("yyyy-MM-dd")
495         ShippedDate = self.ShippedDate.date().toString("yyyy-MM-dd")
496         ShipVia = self.ShipCompanyID.text()
497         Freight = self.Freight.text()
498         ShipName = self.ShipName.text()
499         ShipAddress = self.ShipAddress.text()
500         ShipCity = self.ShipCity.text()
501         ShipRegion = self.ShipRegion.text() # Corrected from ShipName.text()
502         ShipPostalCode = self.ShipPostalCode.text()
503         ShipCountry = self.ShipCountry.text()
504
505         # TODO: Provide the connection string to connect to the Northwind
506         database
507         connection = pyodbc.connect(
508             ""
509         )
510
511         cursor = connection.cursor()
512
513         # TODO: Write SQL query with parameters to insert order
514         sql_query = ""
515
516         """
517
518         # Execute the SQL query with parameter values
519         cursor.execute(sql_query, (CustomerID, int(EmployeeID), OrderDate,
520 RequiredDate, ShippedDate, int(
521 ShipVia), float(Freight), ShipName, ShipAddress, ShipCity,
522 ShipRegion, ShipPostalCode, ShipCountry))
523         connection.commit()

```

```

515         # Retrieve the newly inserted order ID
516         cursor.execute("SELECT max(orderid) AS OrderID from orders")
517         result = cursor.fetchone()
518         order_id = result[0]
519
520
521         # Show a message box with the order ID
522         QtWidgets.QMessageBox.information(
523             self, "Order Inserted", f"Order ID: {order_id} has been inserted
524             successfully.")
525
526         # Close the database connection
527         connection.close()
528         self.insert_order_details(order_id)
529
530     def insert_order_details(self, order_id):
531         """
532         Inserts order details into the 'Order Details' table for a given order
533         ID.
534
535         This function is responsible for inserting the order details, including
536         Product ID,
537         Unit Price, Quantity, and Discount, into the 'Order Details' table in
538         the Northwind
539         database. The order details are associated with the specified order ID.
540
541         Parameters:
542         - order_id (int): The unique identifier of the order for which details
543         are being inserted.
544
545         Note:
546         - Ensure that the 'orderDetailsTable' is correctly populated with the
547         order details.
548         - The database connection parameters (server, database, authentication)
549         should be
550         correctly configured.
551
552         Returns:
553         None
554         """
555         # TODO: Provide the connection string to connect to the Northwind
556         database
557         connection = pyodbc.connect(
558             ""
559         )
560
561         cursor = connection.cursor()
562         num_rows = self.orderDetailsTable.rowCount()
563
564         # Iterate through the rows of 'orderDetailsTable' and insert order
565         details
566         for row in range(num_rows):
567             ProductID = int(self.orderDetailsTable.item(row, 0).text())
568             UnitPrice = float(self.orderDetailsTable.item(row, 1).text())
569             Quantity = int(self.orderDetailsTable.item(row, 2).text())
570             Discount = float(self.orderDetailsTable.item(row, 3).text())
571
572             # TODO: Write SQL query with parameters to insert orders details
573             sql_query = ""
574             # Execute the SQL query with parameter values
575             cursor.execute(sql_query, (int(order_id), ProductID,
576                                     UnitPrice, Quantity, Discount))
577             connection.commit()

```



```

569
570     # Close the database connection
571     connection.close()
572
573     # Clear the form after successfully inserting order details
574     self.clear()
575
576 def clear(self):
577     """
578     Clears and resets all input fields and tables in the master transaction
579     form.
580
581     This function resets all input fields, including Employee ID, Customer
582     ID, Product ID,
583     and other related fields, to their initial state or empty values. It
584     also clears
585     the content of the 'orderDetailsTable' and resets date fields to a
586     default date.
587     This function is typically used to clear the form after an order has
588     been submitted
589     or when the user wants to start with a fresh order entry.
590
591     Note:
592     - Ensure that the relevant input fields and tables are available in the
593     UI.
594
595     Returns:
596     None
597     """
598     self.EmployeeID.setText("")
599     self.EmployeeName.setText("")
600     self.CustomerID.setText("")
601     self.CustomerName.setText("")
602     self.ProductID.setText("")
603     self.ProductName.setText("")
604     self.Quantity.setText("")
605     self.UnitPrice.setText("")
606     self.Discount.setText("")
607     self.orderDetailsTable.clearContents()
608     self.ShipCompanyID.setText("")
609     self.ShipCompany.setText("")
610     self.ShipName.setText("")
611     self.Freight.setText("")
612     self.ShippedDate.setDate(QDate(2000, 1, 1))
613     self.OrderDate.setDate(QDate(2000, 1, 1))
614     self.RequiredDate.setDate(QDate(2000, 1, 1))
615     self.ShipAddress.setText("")
616     self.ShipCity.setText("")
617     self.ShipRegion.setText("")
618     self.ShipPostalCode.setText("")
619     self.ShipCountry.setText("")
620
621 def main():
622     app = QApplication(sys.argv)
623     window = UI()
624     window.show()
625     sys.exit(app.exec())
626
627 if __name__ == "__main__":

```

Listing 2: Skeleton Python file (`main.py`)