

# ASSIGNMENT 4

## COURIER MANAGEMENT SYSTEM

AKILESH K

### Task 1

1.

```
def check_order_status():  
    connection = create_connection()  
    if connection:  
        try:  
            cursor = connection.cursor()  
  
            Courierid=int(input("enter the courierid:"))  
            query=("SELECT status FROM courier WHERE courierid = %s")  
            cursor.execute(query,(Courierid,))  
            result = cursor.fetchone()  
            print(status)  
  
            if result:  
                status = result[0]  
                if status == "Delivered":  
                    print("The order has been delivered.")  
                elif status == "In Transit":  
                    print("The order is still being processed.")  
                elif status == "Shipped":  
                    print("The order has been shipped.")  
                elif status == "Cancelled":  
                    print("The order has been cancelled.")  
            else:  
                print("Invalid order status. Please check the status again.")
```

else:

```
print("Order not found.")
```

except Error as e:

```
print(f"Error retrieving order status: {e}")
```

finally:

```
connection.close()
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python class\
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../../
sign\courier_management\methods_courier.py'
enter the courierid:101
The order has been shipped.
```

```
mysql> select * from courier;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| courierid | sendername | senderaddress | receivername | receiveraddress | weight | status | trackingnumber | deliverydate |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 101 | John Sender | 123 Main St, CityA | Alice Receiver | 456 Oak St, CityA | 2.30 | Shipped | TNX123 | 2023-03-01 |
| 102 | Bob Sender | 789 Elm St, CityB | Eva Receiver | 987 Pine St, CityB | 3.50 | In Transit | TNX456 | 2023-03-03 |
| 103 | Charlie Sender | 321 Maple St, CityC | Grace Receiver | 654 Cedar St, CityC | 1.80 | Delivered | TNX789 | 2023-03-05 |
| 104 | David Sender | 567 Birch St, CityD | Helen Receiver | 876 Spruce St, CityD | 4.20 | Shipped | TNX987 | 2023-03-07 |
| 105 | Emma Sender | 987 Willow St, CityE | Ivan Receiver | 234 Oak St, CityE | 2.70 | Delivered | TNX654 | 2023-03-09 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

2.

```
def categorize_parcel(weight):
```

```
    category = None
```

```
    switch_case = {
```

```
        0 < weight <= 5: "Light",
```

```
        5 < weight <= 20: "Medium",
```

```
        weight > 20: "Heavy"
```

```
    }
```

```
    category = switch_case.get(True, "Invalid Weight")
```

```
return category
```

```
try:
```

```
    weight = float(input("Enter the weight of the parcel: "))
```

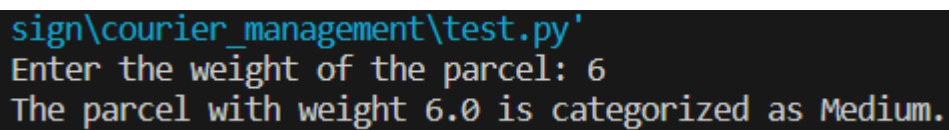
```
except ValueError:
```

```
    print("Invalid input. Please enter a valid numeric value for weight.")
```

```
    exit()
```

```
category = categorize_parcel(weight)
```

```
print(f"The parcel with weight {weight} is categorized as {category}.")
```



```
sign\courier_management\test.py'  
Enter the weight of the parcel: 6  
The parcel with weight 6.0 is categorized as Medium.
```

3.

```
import re
```

```
def validate_customer_information(data, detail):
```

```
    if detail.lower() == 'name':
```

```
        if re.match("^[A-Za-z]+(?: [A-Za-z]+)*$", data):
```

```
            return True
```

```
        else:
```

```
            return False
```

```
    elif detail.lower() == 'address':
```

```
        if re.match("^[A-Za-z0-9\s]+$", data):
```

```
            return True
```

```
        else:
```

```
            return False
```

```
    elif detail.lower() == 'phone number':
```

```
        if re.match("^\\d{3}-\\d{3}-\\d{4}$", data):
```

```
            return True
```

```

else:
    return False

else:
    print("Invalid detail provided.")
    return False

```

```

Name Validation: True
Address Validation: True
Phone Number Validation: True

```

4.

```

def assign_courier(shipments, couriers):
    for shipment in shipments:
        best_match = None
        min_distance = float('inf')

        for courier in couriers:
            distance = calculate_distance(shipment.location, courier.proximity)
            if distance < min_distance and shipment.weight <= courier.load_capacity:
                min_distance = distance
                best_match = courier

        if best_match:
            best_match.assigned_shipments.append(shipment)
            print(f"Shipment {shipment.id} assigned to Courier {best_match.id}")

```

```

PS C:\Users\Akilesh K\OneDrive\Documents\python class challenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Programs\Python\Python311\python.exe' -2023.22.1\pythonFiles\lib\python\debugpy\adapter/..sign\courier_management\test.py'
Shipment 101 assigned to Courier 1
Shipment 102 assigned to Courier 1
Shipment 103 assigned to Courier 2

```

## Task 2

5.

```
def display_orders(customer_id, orders):
```

```
    customer_orders = []
```

```
    for order in orders:
```

```
        if order['customer_id'] == customer_id:
```

```
            customer_orders.append(order)
```

```
    if customer_orders:
```

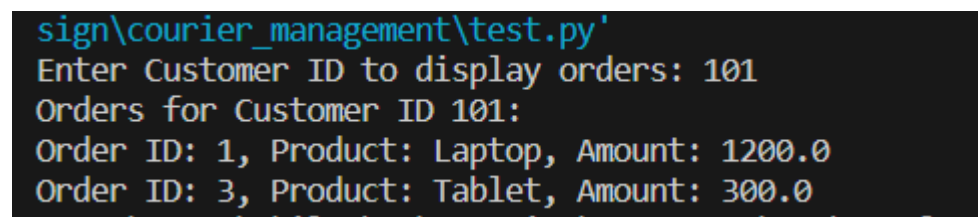
```
        print(f"Orders for Customer ID {customer_id}:")
```

```
        for order in customer_orders:
```

```
            print(f"Order ID: {order['order_id']}, Product: {order['product']}, Amount: {order['amount']}")
```

```
    else:
```

```
        print(f"No orders found for Customer ID {customer_id}")
```



```
sign\courier_management\test.py
Enter Customer ID to display orders: 101
Orders for Customer ID 101:
Order ID: 1, Product: Laptop, Amount: 1200.0
Order ID: 3, Product: Tablet, Amount: 300.0
```

6.

```
def track_courier(courier):
```

```
    print(f"Courier {courier.courier_id} is en route to {courier.destination}.")
```

```
    while not courier.reached_destination():
```

```
        courier.update_location()
```

```
print(f"Current Location: {courier.current_location}")
```

```
time.sleep(1)
```

```
print(f"Courier {courier.courier_id} has reached its destination at {courier.destination}.")
```

### TASK 3

7.

```
def display_tracking_history(parcel):
```

```
    print(f"Tracking history for Parcel {parcel.parcel_id}:")
```

```
    for idx, location in enumerate(parcel.tracking_history, start=1):
```

```
        print(f"{idx}. {location}")
```

8.

```
def find_nearest_courier(order_location, couriers):
```

```
    nearest_courier = None
```

```
    min_distance = float('inf')
```

```
    for courier in couriers:
```

```
        if courier.is_available:
```

```
            distance = calculate_distance(order_location, courier.current_location)
```

```
            if distance < min_distance:
```

```
                min_distance = distance
```

```
                nearest_courier = courier
```

```
    return nearest_courier
```

9.

```
class ParcelTracker:
```

```
    def __init__(self):
```

```
self.parcel_data = [  
    ["TN001", "Parcel in transit"],  
    ["TN002", "Parcel out for delivery"],  
    ["TN003", "Parcel delivered"],  
    ["TN004", "Parcel in transit"],  
]
```

```
def track_parcel(self, tracking_number):
```

```
    for parcel_info in self.parcel_data:  
        if parcel_info[0] == tracking_number:  
            status = parcel_info[1]  
            self.display_tracking_message(status)  
            return
```

```
    print("Tracking number not found.")
```

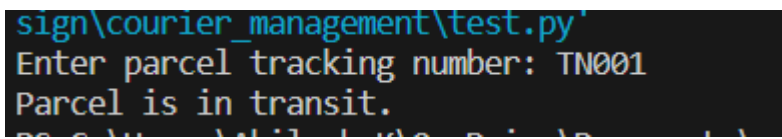
```
def display_tracking_message(self, status):
```

```
    if status == "Parcel in transit":  
        print("Parcel is in transit.")  
    elif status == "Parcel out for delivery":  
        print("Parcel is out for delivery.")  
    elif status == "Parcel delivered":  
        print("Parcel has been delivered.")
```

```
tracker = ParcelTracker()
```

```
tracking_number = input("Enter parcel tracking number: ")
```

```
tracker.track_parcel(tracking_number)
```



```
sign\courier_management\test.py
Enter parcel tracking number: TN001
Parcel is in transit.
press any key to continue
```

10.

```
import re
```

```
def validate_customer_info(data, detail):
```

```
    if detail == "name":
```

```
        if data.isalpha() and data.istitle():
```

```
            return True
```

```
        else:
```

```
            return False
```

```
    elif detail == "address":
```

```
        if all(char.isalnum() or char.isspace() for char in data):
```

```
            return True
```

```
        else:
```

```
            return False
```

```
    elif detail == "phone_number":
```

```
        phone_number_pattern = re.compile(r'^\d{3}-\d{3}-\d{4}$')
```

```
        if phone_number_pattern.match(data):
```

```
            return True
```

```
        else:
```

```
            return False
```

```
    else:
```

```
        print("Invalid detail. Use 'name', 'address', or 'phone_number'.")
```

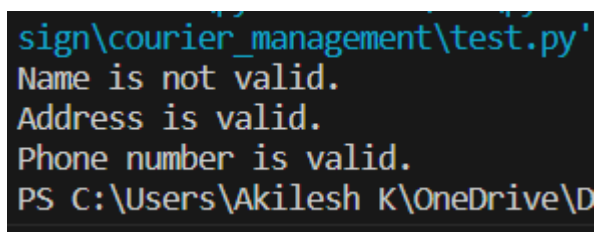


```
return False
```

```
name_result = validate_customer_info("John Doe", "name")  
print("Name is valid." if name_result else "Name is not valid.")
```

```
address_result = validate_customer_info("123 Main St", "address")  
print("Address is valid." if address_result else "Address is not valid.")
```

```
phone_result = validate_customer_info("123-456-7890", "phone_number")  
print("Phone number is valid." if phone_result else "Phone number is not valid.")
```



```
sign\courier_management\test.py'  
Name is not valid.  
Address is valid.  
Phone number is valid.  
PS C:\Users\Akilesh K\OneDrive\D
```

11.

```
def format_address(street, city, state, zip_code):
```

```
    formatted_street = ' '.join(word.capitalize() for word in street.split())
```

```
    formatted_city = city.capitalize()
```

```
    formatted_state = state.capitalize()
```

```
    formatted_zip_code = f"{zip_code[:5]}-{zip_code[5:]}" if len(zip_code) == 9 else zip_code
```

```
    formatted_address = f"{formatted_street}, {formatted_city}, {formatted_state}  
    {formatted_zip_code}"
```

```
    return formatted_address
```

```
hallenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Programs\Python\Python39-2023.22.1\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' 'sign\courier_management\test.py'  
Formatted Address: 123 Main Street, Example city, Sample state 12345-6789
```

12.

```
def generate_order_confirmation_email(self):  
    email_subject = "Order Confirmation"  
  
    email_body = f"Dear {self.customer_name},\n\n\  
        f"Thank you for placing an order with us. Your order number is {self.order_number}.\n\  
        f"Delivery Address: {self.delivery_address}\n\  
        f"Expected Delivery Date: {self.expected_delivery_date}\n\  
        "We appreciate your business. If you have any questions, please contact us.\n\  
        "Sincerely,\n\  
        "Your Company Name"  
  
    return email_subject, email_body
```

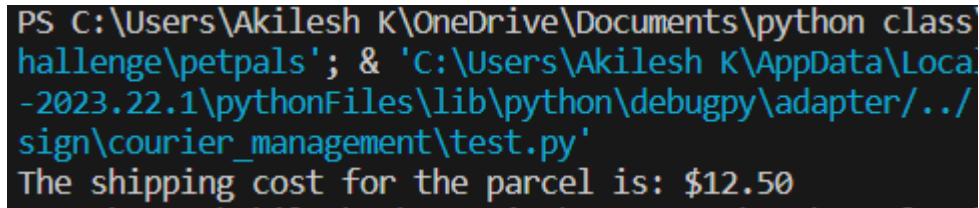
```
Subject: Order Confirmation  
  
Body:  
Dear John Doe,  
  
Thank you for placing an order with us. Your order number is 123456.  
Delivery Address: 123 Main St, Cityville, State  
Expected Delivery Date: January 15, 2023  
  
We appreciate your business. If you have any questions, please contact us.  
  
Sincerely,  
Your Company Name
```

13.

```
def calculate_shipping_cost(source_address, destination_address, parcel_weight):  
  
    base_shipping_rate = 2.5  
  
    distance = calculate_distance(source_address, destination_address)  
  
    shipping_cost = base_shipping_rate * distance * parcel_weight  
  
    return shipping_cost
```

```
def calculate_distance(source_address, destination_address):

    return abs(ord(source_address[0]) - ord(destination_address[0]))
```



```
PS C:\Users\Akilesh K\OneDrive\Documents\python class challenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local -2023.22.1\pythonFiles\lib\python\debugpy\adapter/../sign\courier_management\test.py'
The shipping cost for the parcel is: $12.50
```

14.

```
import string
```

```
import random
```

```
def generate_secure_password(length=12):
```

```
    uppercase_letters = string.ascii_uppercase
```

```
    lowercase_letters = string.ascii_lowercase
```

```
    numbers = string.digits
```

```
    special_characters = string.punctuation
```

```
    all_characters = uppercase_letters + lowercase_letters + numbers + special_characters
```

```
    password = random.choice(uppercase_letters) + random.choice(lowercase_letters) +
    random.choice(numbers) + random.choice(special_characters)
```

```
    for _ in range(length - 4):
```

```
        password += random.choice(all_characters)
```

```
    password_list = list(password)
```

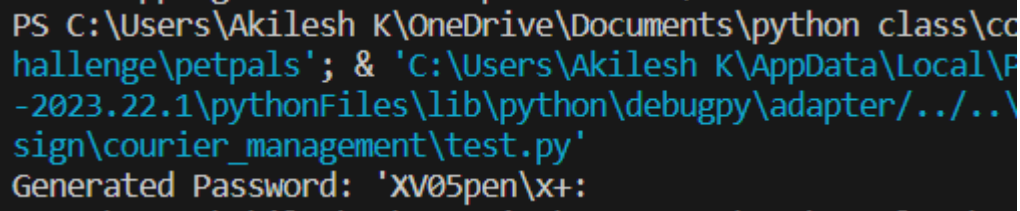
```
    random.shuffle(password_list)
```

```
    final_password = ''.join(password_list)
```

```
return final_password
```

```
secure_password = generate_secure_password()
```

```
print("Generated Password:", secure_password)
```



The screenshot shows a Windows command prompt with a black background and green text. The prompt is 'PS C:\Users\Akilesh K\OneDrive\Documents\python class\challenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\F...'. The command being executed is '-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../../sign\courier\_management\test.py'. The output of the command is 'Generated Password: 'XV05pen\x+:'. The text is partially cut off on the right side.

15.

```
def calculate_jaccard_similarity(str1, str2):
```

```
    set1 = set(str1.lower().split())
```

```
    set2 = set(str2.lower().split())
```

```
    intersection = len(set1.intersection(set2))
```

```
    union = len(set1.union(set2))
```

```
    return intersection / union if union != 0 else 0
```

```
def find_similar_addresses(target_address, addresses, threshold=0.5):
```

```
    similar_addresses = []
```

```
    for address in addresses:
```

```
        similarity = calculate_jaccard_similarity(target_address, address)
```

```
        if similarity >= threshold:
```

```
            similar_addresses.append(address)
```

```
    return similar_addresses
```

```

PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding challenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Programs\Python\Python27-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../../debugpy/sign\courier_management\test.py'
Similar addresses to '123 Main St, Cityville, State':
123 Main St, Cityville, State
124 Main St, Cityville, State

```

## Task 5

1.

class User:

```

def __init__(self, userID, userName, email, password, contactNumber, address):

    self.userID = userID

    self.userName = userName

    self.email = email

    self.password = password

    self.contactNumber = contactNumber

    self.address = address

```

2.

class Courier:

```

def __init__(self, courierID, senderName, senderAddress, receiverName, receiverAddress, weight, status, trackingNumber, deliveryDate, userId):

    self.courierID = courierID

    self.senderName = senderName

    self.senderAddress = senderAddress

    self.receiverName = receiverName

    self.receiverAddress = receiverAddress

    self.weight = weight

    self.status = status

    self.trackingNumber = trackingNumber

    self.deliveryDate = deliveryDate

    self.userId = userId

```

3.

class Employee:

```
def __init__(self, employeeID, employeeName, email, contactNumber, role, salary):
```

```
    self.employeeID = employeeID
```

```
    self.employeeName = employeeName
```

```
    self.email = email
```

```
    self.contactNumber = contactNumber
```

```
    self.role = role
```

```
    self.salary = salary
```

4.

class Location:

```
def __init__(self, locationID, locationName, address):
```

```
    self.locationID = locationID
```

```
    self.locationName = locationName
```

```
    self.address = address
```

5.

class CourierCompany:

```
def __init__(self, companyName):
```

```
    self.companyName = companyName
```

```
    self.courierDetails = []
```

```
    self.employeeDetails = []
```

```
    self.locationDetails = []
```

6.

class Payment:

```
def __init__(self, paymentID, courierID, amount, paymentDate):
```

```
    self.paymentID = paymentID
```

```
    self.courierID = courierID
```

```
    self.amount = amount
```

```
    self.paymentDate = paymentDate
```

## TASK 6

```
class ICourierUserService(ABC):

    @abstractmethod
    def place_order(self, courier_obj):
        pass

    @abstractmethod
    def get_order_status(self, tracking_number):
        pass

    @abstractmethod
    def cancel_order(self, tracking_number):
        pass

    @abstractmethod
    def get_assigned_orders(self, courier_staff_id):
        pass
```

```
class ICourierAdminService(ABC):

    @abstractmethod
    def add_courier_staff(self, name, contact_number):
        pass
```

```
class CourierUserService(ICourierUserService):

    def place_order(self, courier_obj):
        pass

    def get_order_status(self, tracking_number):
        pass
```

```
def cancel_order(self, tracking_number):  
    pass
```

```
def get_assigned_orders(self, courier_staff_id):  
    pass
```

```
class CourierAdminService(ICourierAdminService):  
    def add_courier_staff(self, name, contact_number):  
        pass
```

## **TASK 7**

```
class CourierService:  
    def get_order_status(self, tracking_number):  
        if not self.tracking_number_exists(tracking_number):  
            raise TrackingNumberNotFoundException  
  
    def get_assigned_orders(self, courier_staff_id):  
  
        if not self.is_valid_employee_id(courier_staff_id):  
            raise InvalidEmployeeIdException  
  
    def tracking_number_exists(self, tracking_number):  
        return True  
  
    def is_valid_employee_id(self, employee_id):  
  
        return True
```

## **TASK 9**



## Main

```
from datetime import datetime
```

```
import mysql.connector
```

```
from mysql.connector import Error
```

```
# Your existing code for class definitions, create_connection, and other functions goes here...
```

```
def main():
```

```
    while True:
```

```
        print("\nCourier Management System Menu:")
```

```
        print("1. Display Courier Summary")
```

```
        print("2. Check Order Status")
```

```
        print("3. Update Order Status")
```

```
        print("4. Add New Order")
```

```
        print("5. Delete Order")
```

```
        print("6. Exit")
```

```
        choice = input("Enter your choice (1-6): ")
```

```
        if choice == "1":
```

```
            display_courier_summary()
```

```
        elif choice == "2":
```

```
            check_order_status()
```

```
        elif choice == "3":
```

```
            update_value_in_table()
```

```
        elif choice == "4":
```

```
            add_new_order()
```

```
        elif choice == "5":
```

```
            delete_value_in_table()
```

```
        elif choice == "6":
```

```
            print("Exiting the Courier Management System. Goodbye!")
```

```
        break

    else:

        print("Invalid choice. Please enter a number between 1 and 6.")


if __name__ == "__main__":
    main()
```

## **Data connection**

```
from datetime import datetime
import mysql.connector
from mysql.connector import Error

# Database connection
def create_connection():
    try:
        connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="root",
            port='3306',
            database="courier_management"
        )
        return connection
    except Error as e:
        print(f"Error connecting to the database: {e}")
        return None
```

## **To display courier summary**

```

def display_courier_summary():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            courierid=int(input("enter courier ID:"))
            select_query=("SELECT * from courier where courierid = %s ")
            cursor.execute(select_query,(courierid,))
            cdata = cursor.fetchall()

            print("courier summary:")
            for courierd in cdata:
                print(courierd)

        except Error as e:
            print(f"Error retrieving data: {e}")
        finally:
            connection.close()

```

```

enter courier ID:101
courier summary:
(101, 'John Sender', '123 Main St, CityA', 'Alice Receiver', '456 Oak St, CityA', Decimal('2.30'), 'Shipped', 'TNX123', datetime.date(2023, 3, 1))
Name Validation: True
Address Validation: True
Phone Number Validation: True

```

### To show the status of the courier

```

def check_order_status():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()

```

```
Courierid=int(input("enter the courierid:"))
query=("SELECT status FROM courier WHERE courierid = %s")
cursor.execute(query,(Courierid,))
result = cursor.fetchone()

if result:
    status = result[0]
    if status == "Delivered":
        print("The order has been delivered.")
    elif status == "In Transit":
        print("The order is still being processed.")
    elif status == "Shipped":
        print("The order has been shipped.")
    elif status == "Cancelled":
        print("The order has been cancelled.")
    else:
        print("Invalid order status. Please check the status again.")
else:
    print("Order not found.")

except Error as e:
    print(f"Error retrieving order status: {e}")

finally:
    connection.close()
```

```
PS C:\Users\Akilesh K\OneDrive\Documents\python challenge\petpals'; & 'C:\Users\Akilesh K\AppData -2023.22.1\pythonFiles\lib\python\debugpy\adapte sign\courier_management\methods_courier.py'
enter the courierid:101
The order has been shipped.
```

### To update status of the courier

```
def update_value_in_table():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            courierid=int(input("enter courierid:"))
            new_status=input("enter the new status:")
            update_query = ("UPDATE courier SET status = %s WHERE courierid = %s")
            cursor.execute(update_query, (new_status, courierid))

            connection.commit()

            print(f"Updated value in the database.")

        except Error as e:
            print(f"Error updating value in the table: {e}")

    finally:
        connection.close()
```

```

PS C:\Users\Akilesh K\OneDrive\Documents\python challenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Microsoft\Windows\SelfHosted\Python\2023.22.1\pythonFiles\lib\python\debugpy\adapter\
sign\courier_management\methods_courier.py'
enter courierid:103
enter the new status:In Transit
Updated value in the database.

```

```

mysql> select * from courier;
+-----+-----+-----+-----+-----+-----+-----+
| courierid | sendername | senderaddress | receivername | receiveraddress | weight | status |
+-----+-----+-----+-----+-----+-----+-----+
| 101 | John Sender | 123 Main St, CityA | Alice Receiver | 456 Oak St, CityA | 2.30 | Shipped |
| 102 | Bob Sender | 789 Elm St, CityB | Eva Receiver | 987 Pine St, CityB | 3.50 | In Transit |
| 103 | Charlie Sender | 321 Maple St, CityC | Grace Receiver | 654 Cedar St, CityC | 1.80 | In Transit |
| 104 | David Sender | 567 Birch St, CityD | Helen Receiver | 876 Spruce St, CityD | 4.20 | Shipped |
| 105 | Emma Sender | 987 Willow St, CityE | Ivan Receiver | 234 Oak St, CityE | 2.70 | Delivered |
+-----+-----+-----+-----+-----+-----+-----+

```

## Add new order

```

def add_new_order():
    connection = create_connection()

    if connection:
        try:
            courierid=generate_courier_number()

            sender_name = input("Enter sender name: ")

            weight = float(input("Enter weight: "))

            sender_address = input("Enter sender address: ")

            receiver_name = input("Enter receiver name: ")

            receiver_address = input("Enter receiver address: ")

            status="In Transit"

            trackingno=generate_tracking_number()

            delivery_date = datetime.now().strftime("%Y-%m-%d")

            cursor = connection.cursor()

```

```

        cursor.execute("INSERT INTO
courier(courierid,sendername,senderaddress,receivername,receiveraddress,weight,status,trackingnu
mber,deliverydate) VALUES (%s, %s, %s, %s,%s, %s, %s, %s,%s)",

        (courierid,sender_name ,
sender_address,receiver_name,receiver_address,weight,status,trackingno,delivery_date ))

        connection.commit()

    print("courier details recorded successfully!")

except (Error, ValueError) as e:

    print(f"Error recording courier: {e}")

finally:

    connection.close()

```

```

PS C:\Users\Akilesh K\OneDrive\Documents\python class\coding
challenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\Progra
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../../debug
sign\courier_management\methods_courier.py'
Enter sender name: aki
Enter weight: 3.5
Enter sender address: 15th cross
Enter sender name: hari
Enter receiver address: steve avenue
courier details recorded successfully!

```

```

mysql> select * from courier;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| courierid | sendername | senderaddress | receivername | receiveraddress | weight | status | trackingnumber | deliverydate |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 101 | John Sender | 123 Main St, CityA | Alice Receiver | 456 Oak St, CityA | 2.30 | Shipped | TNX123 | 2023-03-01 |
| 102 | Bob Sender | 789 Elm St, CityB | Eva Receiver | 987 Pine St, CityB | 3.50 | In Transit | TNX456 | 2023-03-03 |
| 103 | Charlie Sender | 321 Maple St, CityC | Grace Receiver | 654 Cedar St, CityC | 1.80 | In Transit | TNX789 | 2023-03-05 |
| 104 | David Sender | 567 Birch St, CityD | Helen Receiver | 876 Spruce St, CityD | 4.20 | Shipped | TNX987 | 2023-03-07 |
| 105 | Emma Sender | 987 Willow St, CityE | Ivan Receiver | 234 Oak St, CityE | 2.70 | Delivered | TNX654 | 2023-03-09 |
| 1001 | aki | 15th cross | hari | steve avenue | 3.50 | In Transit | 2001 | 2023-12-25 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

```

## Delete a record

```

def delete_value_in_table():
    connection = create_connection()
    if connection:
        try:
            cursor = connection.cursor()
            courierid=int(input("enter courierid:"))

            delete_query = ("DELETE FROM courier WHERE courierid = %s")
            cursor.execute(delete_query, ( courierid,))

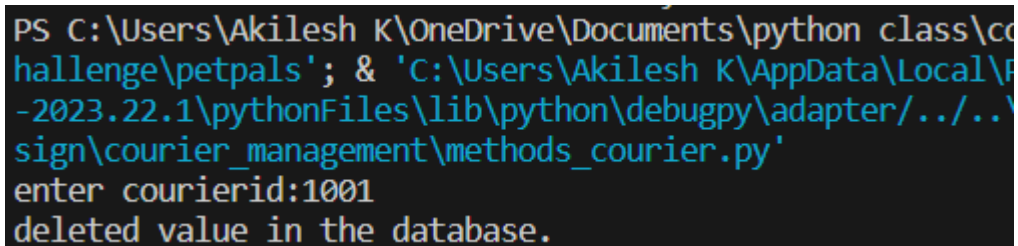
            connection.commit()

            print(f"deleted value in the database.")

        except Error as e:
            print(f"Error deleting value in the table: {e}")

        finally:
            connection.close()

```



```

PS C:\Users\Akilesh K\OneDrive\Documents\python class\challenge\petpals'; & 'C:\Users\Akilesh K\AppData\Local\F
-2023.22.1\pythonFiles\lib\python\debugpy\adapter/../../sign\courier_management\methods_courier.py'
enter courierid:1001
deleted value in the database.

```



```
mysql> select * from courier;
```

courierid	sendername	senderaddress	receivername	receiveraddress	weight	status	trackingnumber	deliverydate
101	John Sender	123 Main St, CityA	Alice Receiver	456 Oak St, CityA	2.30	Shipped	TNX123	2023-03-01
102	Bob Sender	789 Elm St, CityB	Eva Receiver	987 Pine St, CityB	3.50	In Transit	TNX456	2023-03-03
103	Charlie Sender	321 Maple St, CityC	Grace Receiver	654 Cedar St, CityC	1.80	In Transit	TNX789	2023-03-05
104	David Sender	567 Birch St, CityD	Helen Receiver	876 Spruce St, CityD	4.20	Shipped	TNX987	2023-03-07
105	Emma Sender	987 Willow St, CityE	Ivan Receiver	234 Oak St, CityE	2.70	Delivered	TNX654	2023-03-09

```
5 rows in set (0.00 sec)
```