Assignment-Courier Management System
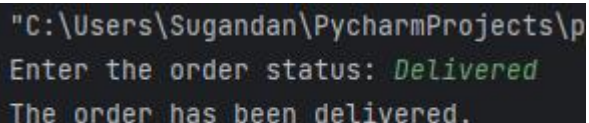
Student Name:Sugandan E

Coding

Task 1: Control Flow Statements

1. Write a program that checks whether a given order is delivered or not based on its status (e.g., "Processing," "Delivered," "Cancelled"). Use if-else statements for this.

```python
def check_delivery_status(order_status):
    if order_status == "Delivered":
        print("The order has been delivered.")
    elif order_status == "Processing":
        print("The order is still being processed.")
    elif order_status == "Cancelled":
        print("The order has been cancelled.")
    else:
        print("Invalid order status. Please check the status again.")

order_status_input = input("Enter the order status: ")
check_delivery_status(order_status_input)
```

```
"C:\Users\Sugandan\PycharmProjects\p
Enter the order status: Delivered
The order has been delivered.
```

2. Implement a switch-case statement to categorize parcels based on their weight into "Light," "Medium," or "Heavy."

```python
def categorize_parcel(weight):
    categories = {
        "Light": lambda x: x < 5,
        "Medium": lambda x: 5 <= x < 10,
        "Heavy": lambda x: x >= 10
    }

    for category, condition in categories.items():
        if condition(weight):
            print(f"The parcel is categorized as {category}.")
            break
    else:
```
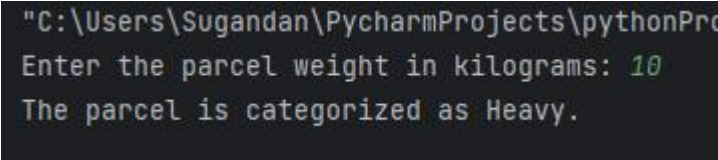
```
      print("Invalid weight. Please check the weight again.")

try:
    parcel_weight = float(input("Enter the parcel weight in kilograms: "))
    categorize_parcel(parcel_weight)
except ValueError:
    print("Invalid input. Please enter a valid numerical weight.")
```

```
"C:\Users\Sugandan\PycharmProjects\pythonPro
Enter the parcel weight in kilograms: 10
The parcel is categorized as Heavy.
```

3. Implement User Authentication 1. Create a login system for employees
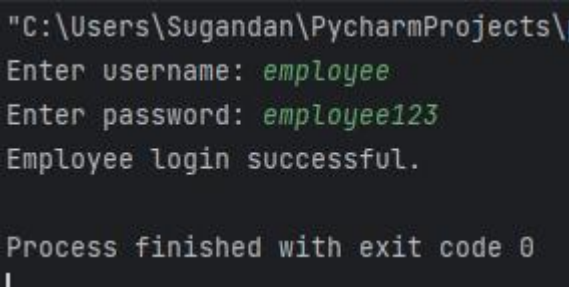and customers using control flow statements.

```
employee_username = "employee"
employee_password = "employee123"

customer_username = "customer"
customer_password = "customer123"

username_input = input("Enter username: ")
password_input = input("Enter password: ")

if username_input == employee_username and password_input == employee_password:
    print("Employee login successful.")
elif username_input == customer_username and password_input == customer_password:
    print("Customer login successful.")

else:
    print("Invalid username or password. Please try again.")
```

```
"C:\Users\Sugandan\PycharmProjects\
Enter username: employee
Enter password: employee123
Employee login successful.

Process finished with exit code 0
```

4. Implement Courier Assignment Logic 1. Develop a mechanism to assign
couriers to shipments based on predefined criteria (e.g., proximity, load
capacity) using loops.

```
class Courier:
    def __init__(self, name, proximity, load_capacity):
        self.name = name
```

```python
        self.proximity = proximity
        self.load_capacity = load_capacity
        self.shipments = []

    def assign_shipment(self, shipment):
        self.shipments.append(shipment)
        print(f"Assigned {shipment['weight']} kg shipment to {self.name}.")


courier1 = Courier("Courier A", 10, 20)
courier2 = Courier("Courier B", 5, 15)
courier3 = Courier("Courier C", 12, 25)

shipments = [
    {"weight": 18, "destination": "Location X"},
    {"weight": 8, "destination": "Location Y"},
    {"weight": 22, "destination": "Location Z"}
]

for shipment in shipments:
    suitable_couriers = [courier for courier in [courier1, courier2, courier3] if
                courier.proximity <= 10 and courier.load_capacity >= shipment["weight"]]

    if suitable_couriers:
        closest_courier = min(suitable_couriers, key=lambda x: x.proximity)
        closest_courier.assign_shipment(shipment)
    else:
        print(f"No suitable courier found for {shipment['weight']} kg shipment to {shipment['destination']}.")

for courier in [courier1, courier2, courier3]:
    print(f"{courier.name} has the following shipments: {courier.shipments}")
```

```
"C:\Users\Sugandan\PycharmProjects\pythonProject\Hexaware foundation\Scripts\python.exe"
Assigned 18 kg shipment to Courier A.
Assigned 8 kg shipment to Courier B.
No suitable courier found for 22 kg shipment to Location Z.
Courier A has the following shipments: [{'weight': 18, 'destination': 'Location X'}]
Courier B has the following shipments: [{'weight': 8, 'destination': 'Location Y'}]
Courier C has the following shipments: []
```

Task 2: Loops and Iteration
5. Write a Python program that uses a for loop to display all the orders for a specific customer.

```python
class Order:
    def __init__(self, order_id, customer_name, status):
```

```python
        self.order_id = order_id
        self.customer_name = customer_name
        self.status = status

orders = [
    Order(1, "CustomerA", "Processing"),
    Order(2, "CustomerB", "Delivered"),
    Order(3, "CustomerA", "Cancelled"),
    Order(4, "CustomerC", "Processing"),
    Order(5, "CustomerA", "Delivered"),
]

def display_orders_for_customer(customer_name):
    customer_orders = [order for order in orders if order.customer_name == customer_name]

    if customer_orders:
        print(f"Orders for {customer_name}:")
        for order in customer_orders:
            print(f"Order ID: {order.order_id}, Status: {order.status}")
    else:
        print(f"No orders found for {customer_name}.")

customer_name_input = input("Enter customer name: ")
display_orders_for_customer(customer_name_input)
```
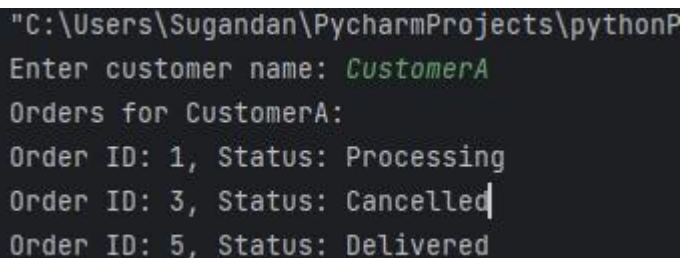
```
"C:\Users\Sugandan\PycharmProjects\pythonP
Enter customer name: CustomerA
Orders for CustomerA:
Order ID: 1, Status: Processing
Order ID: 3, Status: Cancelled
Order ID: 5, Status: Delivered
```

6. Implement a while loop to track the real-time location of a courier until it reaches its destination.

```python
import time

class Courier:
    def __init__(self, name, current_location, destination):
        self.name = name
        self.current_location = current_location
        self.destination = destination

    def update_location(self):

        if self.current_location < self.destination:
            self.current_location += 1
            print(f"{self.name}'s current location: {self.current_location}")
```
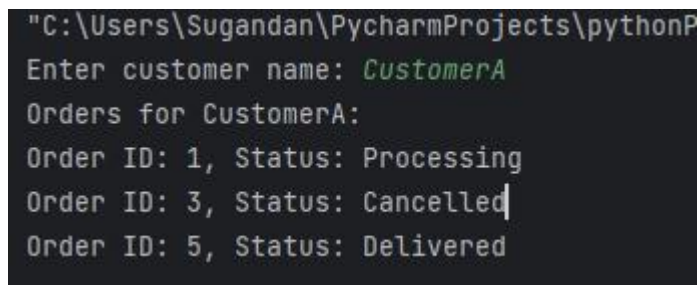
```
        else:
            print(f"{self.name} has reached the destination.")

courier_name = "Courier A"
starting_location = 0
destination_location = 10

courier = Courier(courier_name, starting_location, destination_location)

while courier.current_location < courier.destination:
    courier.update_location()
    time.sleep(1)
print("Tracking completed.")
```



```
"C:\Users\Sugandan\PycharmProjects\pythonP
Enter customer name: CustomerA
Orders for CustomerA:
Order ID: 1, Status: Processing
Order ID: 3, Status: Cancelled
Order ID: 5, Status: Delivered
```

Task 3: Arrays and Data Structures

7. Create an array to store the tracking history of a parcel, where each entry represents a location update.

```python
class Parcel:
    def __init__(self, tracking_id):
        self.tracking_id = tracking_id
        self.tracking_history = []

    def update_location(self, location):
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
        update_entry = {"timestamp": timestamp, "location": location}
        self.tracking_history.append(update_entry)

import time

parcel_tracking_id = "ABC123"
parcel = Parcel(parcel_tracking_id)

locations = ["Warehouse", "In Transit", "Local Distribution Center", "Delivered"]

for location in locations:
    parcel.update_location(location)
    time.sleep(1)

print(f"Tracking history for Parcel {parcel.tracking_id}:")
for entry in parcel.tracking_history:
    print(f"{entry['timestamp']} - Location: {entry['location']}")
```

```
"C:\Users\Sugandan\PycharmProjects\pythonProject\Hexaware for
Tracking history for Parcel ABC123:
2024-03-13 12:21:36 - Location: Warehouse
2024-03-13 12:21:37 - Location: In Transit
2024-03-13 12:21:38 - Location: Local Distribution Center
2024-03-13 12:21:39 - Location: Delivered

Process finished with exit code 0
```

8. Implement a method to find the nearest available courier for a new order using an array of couriers.

```python
import math

class Courier:
    def __init__(self, name, current_location, availability):
        self.name = name
        self.current_location = current_location
        self.availability = availability

def find_nearest_courier(new_order_location, couriers):
    available_couriers = [courier for courier in couriers if courier.availability]

    if not available_couriers:
        print("No available couriers.")
        return None

    nearest_courier = min(available_couriers, key=lambda x: abs(x.current_location -
new_order_location))

    return nearest_courier

couriers = [
    Courier("Courier A", 5, True),
    Courier("Courier B", 8, True),
    Courier("Courier C", 12, False),
    Courier("Courier D", 3, True),
]

new_order_location = 7
nearest_courier = find_nearest_courier(new_order_location, couriers)

if nearest_courier:
    print(f"The nearest available courier for the new order is {nearest_courier.name}.")
else:
    print("No available couriers.")
```

Task 4: Strings, 2d Arrays, user defined functions, Hashmap
9. Parcel Tracking: Create a program that allows users to input a parcel tracking number. Store the tracking number and Status in 2d String Array. Initialize the array with values. Then, simulate the tracking process by displaying messages like "Parcel in transit," "Parcel out for delivery," or "Parcel delivered" based on the tracking number's status.

```python
class ParcelTracker:
    def __init__(self):
        self.tracking_data = [
            ["ABC123", "In Transit"],
            ["XYZ456", "Out for Delivery"],
            ["123DEF", "Processing"],
            ["789GHI", "Delivered"],
        ]

    def get_tracking_status(self, tracking_number):
        for item in self.tracking_data:
            if item[0] == tracking_number:
                return item[1]
        return "Tracking number not found."

    def simulate_tracking_process(self, tracking_number):
        status = self.get_tracking_status(tracking_number)

        if status == "In Transit":
            print(f"Parcel {tracking_number} is currently in transit.")
        elif status == "Out for Delivery":
            print(f"Parcel {tracking_number} is out for delivery.")
        elif status == "Processing":
            print(f"Parcel {tracking_number} is still processing.")
        elif status == "Delivered":
            print(f"Parcel {tracking_number} has been delivered.")
        else:
            print(f"Invalid tracking number: {tracking_number}")

parcel_tracker = ParcelTracker()

user_tracking_number = input("Enter the parcel tracking number: ")
parcel_tracker.simulate_tracking_process(user_tracking_number)
```

```
"C:\Users\Sugandan\PycharmProjects\pythonPro
Enter the parcel tracking number: ABC123
Parcel ABC123 is currently in transit.

Process finished with exit code 0
```

10. Customer Data Validation: Write a function which takes 2 parameters, data-denotes the data and detail-denotes if it is name addtress or phone number.Validate customer information based on following critirea. Ensure that names contain only letters and are properly capitalized, addresses do not contain special characters, and phone numbers follow a specific format (e.g., ###-###-####).

```python
import re

def validate_customer_information(data, detail):
    if detail == "name":
        if data.isalpha() and data.istitle():
            return True
        else:
            return False

    elif detail == "address":
        if data.isalnum() or data.replace(" ", "").isalpha():
            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:

        return False

customer_name = "John Doe"
customer_address = "123 Main Street"
customer_phone_number = "555-123-4567"

if validate_customer_information(customer_name, "name"):
    print("Name is valid.")
else:
```
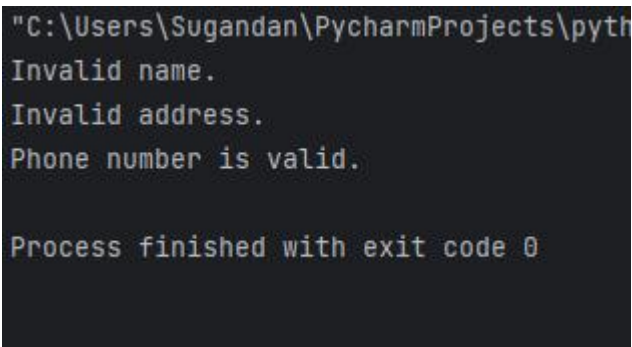
```python
    print("Invalid name.")

if validate_customer_information(customer_address, "address"):
    print("Address is valid.")
else:
    print("Invalid address.")

if validate_customer_information(customer_phone_number, "phone_number"):
    print("Phone number is valid.")
else:
    print("Invalid phone number.")
```

```
"C:\Users\Sugandan\PycharmProjects\pyth
Invalid name.
Invalid address.
Phone number is valid.


Process finished with exit code 0
```

11. Address Formatting: Develop a function that takes an address as input (street, city, state, zip code) and formats it correctly, including capitalizing the first letter of each word and properly formatting the zip code.

```python
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.upper()
    formatted_zip_code = 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

street_input = input("Enter street address: ")
city_input = input("Enter city: ")
state_input = input("Enter state: ")
zip_code_input = input("Enter zip code: ")

formatted_address = format_address(street_input, city_input, state_input, zip_code_input)
print("Formatted Address:", formatted_address)
```

```
"C:\Users\Sugandan\PycharmProjects\pythonProject\Hexar
Enter street address: Rambo St
Enter city: Austin
Enter state: Texas
Enter zip code: 000241
Formatted Address: Rambo St, Austin, TEXAS 000241


Process finished with exit code 0
```

12. Order Confirmation Email: Create a program that generates an order confirmation email. The email should include details such as the customer's name, order number, delivery address, and expected delivery date.

```python
import datetime

def generate_order_confirmation_email(customer_name, order_number, delivery_address):
    expected_delivery_date = datetime.date.today() + datetime.timedelta(days=2)

    email_content = f"""
    Subject: Order Confirmation - Order

    Dear {customer_name},

    Thank you for placing an order with us! Your order #{order_number} has been confirmed.

    Order Details:
    - Order Number: {order_number}
    - Delivery Address: {delivery_address}
    - Expected Delivery Date: {expected_delivery_date}

    If you have any questions or concerns, please feel free to contact our customer support.

    Thank you for choosing our service!

    Best regards,
    Your Company Name
    """
    return email_content
customer_name_input = input("Enter customer's name: ")
order_number_input = input("Enter order number: ")
delivery_address_input = input("Enter delivery address: ")

order_confirmation_email = generate_order_confirmation_email(customer_name_input,
order_number_input, delivery_address_input)
print("Order Confirmation Email:")
print(order_confirmation_email)
```

```
"C:\Users\Sugandan\PycharmProjects\pythonProject\Hexaware foundation\Scripts\python.exe" C:\Users\Sug
Enter customer's name: Sugandan
Enter order number: 2012
Enter delivery address: 302,2 nd cross st,pondichery 605110
Order Confirmation Email:

    Subject: Order Confirmation - Order #2012

    Dear Sugandan,

    Thank you for placing an order with us! Your order #2012 has been confirmed.

    Order Details:
    - Order Number: 2012
    - Delivery Address: 302,2 nd cross st,pondichery 605110
    - Expected Delivery Date: 2024-03-15

    If you have any questions or concerns, please feel free to contact our customer support.

    Thank you for choosing our service!

    Best regards,
    Your Company Name
```

13. Calculate Shipping Costs: Develop a function that calculates the shipping cost based on the distance between two locations and the weight of the parcel. You can use string inputs for the source and destination addresses.

```python
def calculate_shipping_cost(source_address, destination_address, parcel_weight):
    BASE_COST = 5
    DISTANCE_COST_FACTOR = 0.1
    WEIGHT_COST_FACTOR = 0.2

    distance_km = 50


    distance_cost = distance_km * DISTANCE_COST_FACTOR
    weight_cost = parcel_weight * WEIGHT_COST_FACTOR

    total_cost = BASE_COST + distance_cost + weight_cost
    return total_cost

source_address_input = input("Enter source address: ")
destination_address_input = input("Enter destination address: ")
parcel_weight_input = float(input("Enter parcel weight in kilograms: "))

shipping_cost = calculate_shipping_cost(source_address_input, destination_address_input,
parcel_weight_input)
print(f"The estimated shipping cost is ${shipping_cost:.2f}")
```

```
"C:\Users\Sugandan\PycharmProjects\pythonProject\Hexaware foundation
Enter source address: 123 Main Street, CityA, StateA, 12345
Enter destination address: 456 Broadway, CityB, StateB, 67890
Enter parcel weight in kilograms: 2.5
The estimated shipping cost is $10.50

Process finished with exit code 0
```

14. Password Generator: Create a function that generates secure passwords for courier system accounts. Ensure the passwords contain a mix of uppercase letters, lowercase letters, numbers, and special characters.

```python
import random
import string

def generate_secure_password(length=12):

    uppercase_letters = string.ascii_uppercase
    lowercase_letters = string.ascii_lowercase
    digits = string.digits
    special_characters = string.punctuation

        all_characters = uppercase_letters + lowercase_letters + digits + special_characters


    password = random.choice(uppercase_letters) + random.choice(lowercase_letters) +
random.choice(digits) + random.choice(special_characters)

    for _ in range(length - 4):
        password += random.choice(all_characters)

    password_list = list(password)
    random.shuffle(password_list)
    password = ''.join(password_list)

    return password

generated_password = generate_secure_password()
print("Generated Password:", generated_password)
```

```
"C:\Users\Sugandan\PycharmProjects\pytho
Generated Password: \V].1&DviRsp

Process finished with exit code 0
```

15. Find Similar Addresses: Implement a function that finds similar addresses in the system. This can be useful for identifying duplicate customer entries or optimizing delivery routes.Use string functions to implement this.

```python
def find_similar_addresses(address, addresses, similarity_threshold=0.8):
    similar_addresses = []

    for other_address in addresses:
        similarity_score = calculate_similarity(address, other_address)

        if similarity_score >= similarity_threshold:
            similar_addresses.append(other_address)

    return similar_addresses

def calculate_similarity(address1, address2):
    address1 = address1.lower()
    address2 = address2.lower()

    intersection = set(address1.split()) & set(address2.split())
    union = set(address1.split()) | set(address2.split())

    similarity_score = len(intersection) / len(union)
    return similarity_score

all_addresses = [
    "123 Main Street, CityA, StateA, 12345",
    "124 Main St, CityA, StateA, 12345",
    "456 Broadway, CityB, StateB, 67890",
    "789 Elm St, CityC, StateC, 98765",
]

input_address = input("Enter a address:")
similar_addresses = find_similar_addresses(input_address, all_addresses)

print(f"Similar addresses to '{input_address}':")
for similar_address in similar_addresses:
    print(similar_address)
```

```
"C:\Users\Sugandan\PycharmProjects\pythonProject\Hexaware foundati
Enter a address:123 Main Street, CityA, StateA, 12345
Similar addresses to '123 Main Street, CityA, StateA, 12345':
123 Main Street, CityA, StateA, 12345

Process finished with exit code 0
```

Task 5: Object Oriented Programming
Scope : Entity classes/Models/POJO, Abstraction/Encapsulation
Create the following model/entity classes within package entities with
variables declared private, constructors(default and
parametrized,getters,setters and toString())

1. User Class: Variables: userID , userName , email , password ,
contactNumber , address

2. Courier Class Variables: courierID , senderName , senderAddress ,
receiverName , receiverAddress , weight , status, trackingNumber ,
deliveryDate ,userId

3. Employee Class: Variables employeeID , employeeName , email ,
contactNumber , role String, salary

4. Location Class Variables LocationID , LocationName , Address

5. CourierCompany Class Variables companyName , courierDetails -collection
of Courier Objects, employeeDetailscollection of Employee Objects,
locationDetails - collection of Location Objects. 6. Payment Class:
Variables PaymentID long, CourierID long, Amount double, PaymentDate Date

```python
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

    def get_userID(self):
        return self.__userID

    def set_userID(self, userID):
        self.__userID = userID
```

```python
    def get_userName(self):
        return self.__userName

    def set_userName(self, userName):
        self.__userName = userName

    def get_email(self):
        return self.__email

    def set_email(self, email):
        self.__email = email

    def get_password(self):
        return self.__password

    def set_password(self, password):
        self.__password = password

    def get_contactNumber(self):
        return self.__contactNumber

    def set_contactNumber(self, contactNumber):
        self.__contactNumber = contactNumber

    def get_address(self):
        return self.__address

    def set_address(self, address):
        self.__address = address

    def __str__(self):
        return f"UserID: {self.__userID}, UserName: {self.__userName}, Email: {self.__email}, Password: {self.__password}, ContactNumber: {self.__contactNumber}, Address: {self.__address}"


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
```

```python
        self.__userId = userId

    def get_courierID(self):
        return self.__courierID

    def set_courierID(self, courierID):
        self.__courierID = courierID

    def get_senderName(self):
        return self.__senderName

    def set_senderName(self, senderName):
        self.__senderName = senderName

    def get_senderAddress(self):
        return self.__senderAddress

    def set_senderAddress(self, senderAddress):
        self.__senderAddress = senderAddress

    def get_receiverName(self):
        return self.__receiverName

    def set_receiverName(self, receiverName):
        self.__receiverName = receiverName

    def get_receiverAddress(self):
        return self.__receiverAddress

    def set_receiverAddress(self, receiverAddress):
        self.__receiverAddress = receiverAddress

    def get_weight(self):
        return self.__weight

    def set_weight(self, weight):
        self.__weight = weight

    def get_status(self):
        return self.__status

    def set_status(self, status):
        self.__status = status

    def get_trackingNumber(self):
        return self.__trackingNumber

    def set_trackingNumber(self, trackingNumber):
```

```python
        self.__trackingNumber = trackingNumber

    def get_deliveryDate(self):
        return self.__deliveryDate

    def set_deliveryDate(self, deliveryDate):
        self.__deliveryDate = deliveryDate

    def get_userId(self):
        return self.__userId

    def set_userId(self, userId):
        self.__userId = userId

    def __str__(self):
        return f"CourierID: {self.__courierID}, SenderName: {self.__senderName}, SenderAddress:
{self.__senderAddress}, ReceiverName: {self.__receiverName}, ReceiverAddress:
{self.__receiverAddress}, Weight: {self.__weight}, Status: {self.__status}, TrackingNumber:
{self.__trackingNumber}, DeliveryDate: {self.__deliveryDate}, UserID: {self.__userId}"


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

    def get_employeeID(self):
        return self.__employeeID

    def set_employeeID(self, employeeID):
        self.__employeeID = employeeID

    def get_employeeName(self):
        return self.__employeeName

    def set_employeeName(self, employeeName):
        self.__employeeName = employeeName

    def get_email(self):
        return self.__email

    def set_email(self, email):
        self.__email = email
```

```python
    def get_contactNumber(self):
        return self.__contactNumber

    def set_contactNumber(self, contactNumber):
        self.__contactNumber = contactNumber

    def get_role(self):
        return self.__role

    def set_role(self, role):
        self.__role = role

    def get_salary(self):
        return self.__salary

    def set_salary(self, salary):
        self.__salary = salary

    def __str__(self):
        return f"EmployeeID: {self.__employeeID}, EmployeeName: {self.__employeeName}, Email: {self.__email}, ContactNumber: {self.__contactNumber}, Role: {self.__role}, Salary: {self.__salary}"


class Location:
    def __init__(self, LocationID, LocationName, Address):
        self.__LocationID = LocationID
        self.__LocationName = LocationName
        self.__Address = Address

    def get_LocationID(self):
        return self.__LocationID

    def set_LocationID(self, LocationID):
        self.__LocationID = LocationID

    def get_LocationName(self):
        return self.__LocationName

    def set_LocationName(self, LocationName):
        self.__LocationName = LocationName

    def get_Address(self):
        return self.__Address

    def set_Address(self, Address):
        self.__Address = Address

    def __str__(self):
```

```python
        return f"LocationID: {self.__LocationID}, LocationName: {self.__LocationName}, Address: {self.__Address}"


class CourierCompany:
    def __init__(self, companyName):
        self.__companyName = companyName
        self.__courierDetails = []
        self.__employeeDetails = []
        self.__locationDetails = []

    def get_companyName(self):
        return self.__companyName

    def set_companyName(self, companyName):
        self.__companyName = companyName

    def add_courier(self, courier):
        self.__courierDetails.append(courier)

    def remove_courier(self, courier):
        self.__courierDetails.remove(courier)

    def add_employee(self, employee):
        self.__employeeDetails.append(employee)

    def remove_employee(self, employee):
        self.__employeeDetails.remove(employee)

    def add_location(self, location):
        self.__locationDetails.append(location)

    def remove_location(self, location):
        self.__locationDetails.remove(location)

    def __str__(self):
        return f"CompanyName: {self.__companyName}, CourierDetails: {self.__courierDetails}, EmployeeDetails: {self.__employeeDetails}, LocationDetails: {self.__locationDetails}"


class Payment:
    def __init__(self, PaymentID, CourierID, LocationID, Amount, PaymentDate, EmployeeID):
        self.__PaymentID = PaymentID
        self.__CourierID = CourierID
        self.__LocationID = LocationID
        self.__Amount = Amount
        self.__PaymentDate = PaymentDate
        self.__EmployeeID = EmployeeID
```

```python
    def get_PaymentID(self):
        return self.__PaymentID

    def set_PaymentID(self, PaymentID):
        self.__PaymentID = PaymentID

    def get_CourierID(self):
        return self.__CourierID

    def set_CourierID(self, CourierID):
        self.__CourierID = CourierID

    def get_LocationID(self):
        return self.__LocationID

    def set_LocationID(self, LocationID):
        self.__LocationID = LocationID

    def get_Amount(self):
        return self.__Amount

    def set_Amount(self, Amount):
        self.__Amount = Amount

    def get_PaymentDate(self):
        return self.__PaymentDate

    def set_PaymentDate(self, PaymentDate):
        self.__PaymentDate = PaymentDate

    def get_EmployeeID(self):
        return self.__EmployeeID

    def set_EmployeeID(self, EmployeeID):
        self.__EmployeeID = EmployeeID

    def __str__(self):
        return f"PaymentID: {self.__PaymentID}, CourierID: {self.__CourierID}, LocationID:
{self.__LocationID}, Amount: {self.__Amount}, PaymentDate: {self.__PaymentDate}, EmployeeID:
{self.__EmployeeID}"e}"
```

Task 6: Service Provider Interface /Abstract class
Create 2 Interface /Abstract class ICourierUserService and ICourierAdminService
interface ICourierUserService { // Customer-related functions

```
 placeOrder()
 /** Place a new courier order. * @param courierObj Courier object created using values
entered by users * @return The unique tracking number for the courier order . Use a
static variable to generate unique tracking number. Initialize the static variable in
Courier class with some random value. Increment the static variable each time in the
constructor to generate next values.
 getOrderStatus();
 /**Get the status of a courier order. *@param trackingNumber The tracking number of
the courier order. * @return The status of the courier order (e.g., yetToTransit, In
Transit, Delivered). */
 cancelOrder()
 /** Cancel a courier order. * @param trackingNumber The tracking number of the courier
order to be canceled. * @return True if the order was successfully canceled, false
otherwise.*/


getAssignedOrder(); /** Get a list of orders assigned to a specific courier staff
member * @param courierStaffId The ID of the courier staff member. * @return A list of
courier orders assigned to the staff member.*/
// Admin functions



ICourierAdminService
int addCourierStaff(Employee obj);
/** Add a new courier staff member to the system. * @param name The name of the courier
staff member. * @param contactNumber The contact number of the courier staff member. *
@return The ID of the newly added courier staff member. */
```

from abc import ABC, abstractmethod

```python
class ICourierUserService(ABC):
    @abstractmethod
    def placeOrder(self, courierObj):
        pass

    @abstractmethod
    def getOrderStatus(self, trackingNumber):
        pass

    @abstractmethod
    def cancelOrder(self, trackingNumber):
        pass

    @abstractmethod
    def getAssignedOrder(self, courierStaffId):
        pass

class ICourierAdminService(ABC):
    @abstractmethod
```

```
    def addCourierStaff(self, name, contactNumber):
        Pass
```

Task 7: Exception Handling
(Scope: User Defined Exception/Checked /Unchecked Exception/Exception handling using try..catch finally, thow & throws keyword usage)
Define the following custom exceptions and throw them in methods whenever needed . Handle all the excpetionsin main method,
1. TrackingNumberNotFoundException :throw this exception when user try to withdraw amount or transfer amount to another acco
2. InvalidEmployeeIdException throw this exception when id entered for the employee not existing in the system

```python
from abc import ABC, abstractmethod


class TrackingNumberNotFoundException(Exception):
    pass


class InvalidEmployeeIdException(Exception):
    pass


class ICourierUserService(ABC):
    @abstractmethod
    def placeOrder(self, courierObj):
        pass

    @abstractmethod
    def getOrderStatus(self, trackingNumber):
        pass

    @abstractmethod
    def cancelOrder(self, trackingNumber):
        pass

    @abstractmethod
    def getAssignedOrder(self, courierStaffId):
        pass


class ICourierAdminService(ABC):
    @abstractmethod
    def addCourierStaff(self, name, contactNumber):
```

```python
        pass

def getCouriersByEmployee(self, employee_id):
    try:
        cursor = self.connection.cursor()


        sql_query = """SELECT *
                    FROM Couriers
                    WHERE EmployeeID = ?"""


        cursor.execute(sql_query, (employee_id,))


        couriers = cursor.fetchall()

        print("Couriers retrieved successfully.")
        return couriers
    except exception.InvalidEmployeeIdException as ex:
        print(f"Error retrieving assigned orders: {ex}")
    except Exception as ex:
        print(f"Error retrieving assigned orders: {ex}")
    finally:
        cursor.close()


 def updateCourierStatus(self, trackingNumber, newStatus):
    try:
        cursor = self.connection.cursor()


        sql_query = """UPDATE Couriers
                    SET Status = ?
                    WHERE TrackingNumber = ?"""


        cursor.execute(sql_query, (newStatus, trackingNumber))


        self.connection.commit()

        print("Order cancelled successfully.")
    except exception.TrackingNumberNotFoundException as ex:
        print(f"Error cancelling order: {ex}")
    except Exception as ex:
        print(f"Error cancelling order: {ex}")
    finally:
```

```python
        cursor.close()

def addCourierStaff(self, empID, name, email, contact_number, role, salary):
    try:
        cursor = self.connection.cursor()


        sql_query = """INSERT INTO Employees (EmployeeID,Name, Email, ContactNumber, Role, Salary)
                    VALUES (?,?, ?, ?, ?, ?)"""


        cursor.execute(sql_query, (empID, name, email, contact_number, role, salary))


        self.connection.commit()

        print("Courier staff added successfully.")
    except Exception as ex:
        print(f"Error adding courier staff: {ex}")
    finally:
        cursor.close()

  def insertOrder(self, courierID, sender_name, sender_address, receiver_name, receiver_address, weight, status,
            tracking_number, delivery_date, location_id, employee_id, service_id):
    try:
        cursor = self.connection.cursor()


        sql_query = """INSERT INTO Couriers (CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, LocationID, EmployeeID, ServiceID)
                    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"""


        cursor.execute(sql_query,
                    (courierID, sender_name, sender_address, receiver_name, receiver_address, weight, status,
                    tracking_number, delivery_date, location_id, employee_id, service_id))


        self.connection.commit()

        print("Order inserted successfully.")
    except Exception as ex:
        print(f"Error inserting order: {ex}")
    finally:
```

```
        cursor.close()
```

Task 8: Service implementation

1. Create CourierUserServiceImpl class which implements ICourierUserService interface which holds a variable named companyObj of type CourierCompany. This variable can be used to access the Object Arrays to access data relevant in method implementations.
2. Create CourierAdminService Impl class which inherits from CourierUserServiceImpl and implements ICourierAdminService interface.
3. Create CourierAdminServiceCollectionImpl class which inherits from CourierUserServiceColectionImpl and implements ICourierAdminService interface.

```python
class ICourierUserService(ABC):
    @abstractmethod
    def placeOrder(self, courierObj):
        pass

    @abstractmethod
    def getOrderStatus(self, trackingNumber):
        pass

    @abstractmethod
    def cancelOrder(self, trackingNumber):
        pass

    @abstractmethod
    def getAssignedOrder(self, courierStaffId):
        pass

class ICourierAdminService(ICourierUserService):
    @abstractmethod
    def addCourierStaff(self, name, contactNumber):
        pass




class CourierUserServiceImpl(ICourierUserService):
    def __init__(self, company_name):
        self.companyObj = CourierCompany(company_name)

    def placeOrder(self, courierObj):
        self.companyObj.add_courier(courierObj)
        print("Order placed successfully.")

    def getOrderStatus(self, trackingNumber):
```

```python
            for courier in self.companyObj.courier_details:
                if courier.trackingNumber == trackingNumber:
                    return courier.status
            return "Order not found."

    def cancelOrder(self, trackingNumber):
        for courier in self.companyObj.courier_details:
            if courier.trackingNumber == trackingNumber:
                self.companyObj.remove_courier(courier)
                print("Order canceled successfully.")
                return
        print("Order not found.")

    def getAssignedOrder(self, courierStaffId):
        assigned_orders = []
        for courier in self.companyObj.courier_details:
            if courier.employeeId == courierStaffId:
                assigned_orders.append(courier)
        return assigned_orders


class CourierAdminServiceImpl(CourierUserServiceImpl, ICourierAdminService):
    def addCourierStaff(self, name, contactNumber):

        new_employee_id = len(self.companyObj.employee_details) + 1

        new_employee = Employee(new_employee_id, name, None, contactNumber, None, None)


        self.companyObj.add_employee(new_employee)

        print("Courier staff added successfully.")


class CourierAdminServiceCollectionImpl(CourierUserServiceImpl, ICourierAdminService):
    def __init__(self, company_name):
        super().__init__(company_name)

    def addCourierStaff(self, name, contactNumber):
        new_employee_id = len(self.companyObj.employee_details) + 1

        new_employee = Employee(new_employee_id, name, None, contactNumber, None, None)

        self.companyObj.add_employee(new_employee)

        print("Courier staff added successfully.")
```

Task 9: Database Interaction Connect your application to the SQL database for the Courier Management System

1. Write code to establish a connection to your SQL database. Create a class DBConnection in a package connectionutil with a static variable connection of Type Connection and a static method getConnection() which returns connection. Connection properties supplied in the connection string should be read from a property file.
2. Create a Service class CourierServiceDb in dao with a static variable named connection of type Connection which can be assigned in the constructor by invoking the method in DBConnection Class.
3. Include methods to insert, update, and retrieve data from the database (e.g., inserting a new order, updating courier status).
4. Implement a feature to retrieve and display the delivery history of a specific parcel by querying the database. 1. Generate and display reports using data retrieved from the database (e.g., shipment status report, revenue report).

```python
import CMS.dao
import CMS.exception

class CourierServiceDb:
    connection = CMS.dao.connect_to_sql_server()

    def __init__(self):
        self.connection = CMS.dao.connect_to_sql_server()

    def cancelOrder(self, tracking_number):
        try:
            cursor = self.connection.cursor()


            sql_query = """UPDATE Couriers
                    SET Status = 'Cancelled'
                    WHERE TrackingNumber = ?"""


            cursor.execute(sql_query, (tracking_number,))


            self.connection.commit()

            print("Order cancelled successfully.")
```

```python
        except Exception as ex:
            print(f"Error cancelling order: {ex}")
        finally:
            cursor.close()

    def getCouriersByEmployee(self, employee_id):
        try:
            cursor = self.connection.cursor()


            sql_query = """SELECT *
                    FROM Couriers
                    WHERE EmployeeID = ?"""


            cursor.execute(sql_query, (employee_id,))


            couriers = cursor.fetchall()

            print("Couriers retrieved successfully.")
            return couriers
        except exception.InvalidEmployeeIdException as ex:
            print(f"Error retrieving assigned orders: {ex}")
        except Exception as ex:
            print(f"Error retrieving assigned orders: {ex}")
        finally:
            cursor.close()

    def addCourierStaff(self, empID, name, email, contact_number, role, salary):
        try:
            cursor = self.connection.cursor()


            sql_query = """INSERT INTO Employees (EmployeeID,Name, Email, ContactNumber, Role,
Salary)
                    VALUES (?,?, ?, ?, ?, ?)"""


            cursor.execute(sql_query, (empID, name, email, contact_number, role, salary))


            self.connection.commit()

            print("Courier staff added successfully.")
        except Exception as ex:
            print(f"Error adding courier staff: {ex}")
        finally:
```

```python
            cursor.close()

    def insertOrder(self, courierID, sender_name, sender_address, receiver_name,
    receiver_address, weight, status,
                tracking_number, delivery_date, location_id, employee_id, service_id):
        try:
            cursor = self.connection.cursor()


            sql_query = """INSERT INTO Couriers (CourierID, SenderName, SenderAddress,
    ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, LocationID,
    EmployeeID, ServiceID)
                    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"""


            cursor.execute(sql_query,
                    (courierID, sender_name, sender_address, receiver_name, receiver_address,
    weight, status,
                    tracking_number, delivery_date, location_id, employee_id, service_id))


            self.connection.commit()

            print("Order inserted successfully.")
        except Exception as ex:
            print(f"Error inserting order: {ex}")
        finally:
            cursor.close()

    def updateCourierStatus(self, trackingNumber, newStatus):
        try:
            cursor = self.connection.cursor()


            sql_query = """UPDATE Couriers
                    SET Status = ?
                    WHERE TrackingNumber = ?"""


            cursor.execute(sql_query, (newStatus, trackingNumber))


            self.connection.commit()

            print("Order cancelled successfully.")
        except exception.TrackingNumberNotFoundException as ex:
            print(f"Error cancelling order: {ex}")
        except Exception as ex:
```

```python
            print(f"Error cancelling order: {ex}")
        finally:
            cursor.close()

    def retrieveDeliveryHistory(self, trackingNumber):
        try:
            cursor = self.connection.cursor()


            sql_query = """SELECT *
                    FROM Couriers
                    WHERE TrackingNumber = ?"""


            cursor.execute(sql_query, (trackingNumber,))


            delivery_history = cursor.fetchall()

            print("Delivery history retrieved successfully.")
            return delivery_history
        except Exception as ex:
            print(f"Error retrieving delivery history: {ex}")
        finally:
            cursor.close()

    def generateShipmentStatusReport(self):
        try:
            cursor = self.connection.cursor()


            sql_query = """SELECT TrackingNumber, Status
                    FROM Couriers"""


            cursor.execute(sql_query)

            shipment_status_report = cursor.fetchall()

            print("Shipment status report generated successfully.")
            return shipment_status_report
        except Exception as ex:
            print(f"Error generating shipment status report: {ex}")
        finally:
            cursor.close()

    def generateRevenueReport(self):
        try:
```

```python
        cursor = self.connection.cursor()

        sql_query = """SELECT SUM(Amount) as TotalRevenue
                FROM Payments"""

        cursor.execute(sql_query)

        total_revenue = cursor.fetchone()[0]

        print("Revenue report generated successfully.")
        return total_revenue
    except Exception as ex:
        print(f"Error generating revenue report: {ex}")
    finally:
        cursor.close()
```
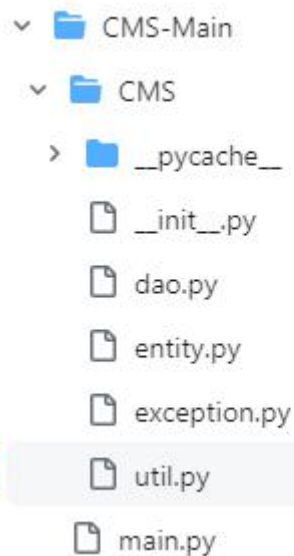
File Structure

```
∨ 📁 CMS-Main
    ∨ 📁 CMS
        > 📁 __pycache__
          📄 __init__.py
          📄 dao.py
          📄 entity.py
          📄 exception.py
          📄 util.py
        📄 main.py
```

MAIN.py

```python
import CMS




def main():
    connection = CMS.dao.connect_to_sql_server()

    courier_service = CMS.util.CourierServiceDb()

    while True:
        print("\nCourier Service Menu:")
        print("1. Place an order")
        print("2. Get order status")
        print("3. Cancel an order")
        print("4. Get assigned orders")
        print("5. Add courier staff (Admin)")
        print("6. Generate report")
        print("7. Exit")

        choice = input("Enter your choice: ")

        if choice == '1':

            courierID = int(input("Enter your courier ID: "))
            sender_name = input("Enter sender's name: ")
            sender_address = input("Enter sender's address: ")
            receiver_name = input("Enter receiver's name: ")
            receiver_address = input("Enter receiver's address: ")
            weight = float(input("Enter weight: "))
            tracking_number = input("Enter tracking number: ")
            delivery_date = input("Enter delivery date (YYYY-MM-DD): ")
            location_id = int(input("Enter location ID: "))
            employee_id = int(input("Enter employee ID: "))
            service_id = int(input("Enter service ID: "))

            courier_service.insertOrder(courierID, sender_name, sender_address, receiver_name, receiver_address, weight,
                            "Processing", tracking_number, delivery_date, location_id, employee_id,
                            service_id)


        elif choice == '2':
```

```python
        tracking_number = input("Enter tracking number: ")
        status = courier_service.retrieveDeliveryHistory(tracking_number)
        print(f"Order status for tracking number {tracking_number}: {status}")

    elif choice == '3':

        tracking_number = input("Enter tracking number: ")
        courier_service.cancelOrder(tracking_number)

    elif choice == '4':

        employee_id = input("Enter employee ID: ")

        couriers = courier_service.getCouriersByEmployee(employee_id)

        print("Couriers handled by Employee ID:", employee_id)
        for courier in couriers:
            print(courier)




    elif choice == '5':

        empID = int(input("Enter staff ID:"))
        name = input("Enter staff name: ")
        email = input("Enter staff email: ")
        contact_number = input("Enter staff contact number: ")
        role = input("Enter staff role: ")
        salary = float(input("Enter staff salary: "))
        courier_service.addCourierStaff(empID, name, email, contact_number, role, salary)

    elif choice == '6':
        shipment_status_report = courier_service.generateShipmentStatusReport()
        print("Shipment status report:")
        for row in shipment_status_report:
            print(row)

        total_revenue = courier_service.generateRevenueReport()
        print("Total Revenue:", total_revenue)
    elif choice == '7':

        print("Exiting program...")
        break

    else:
        print("Invalid choice. Please enter a number between 1 and 7.")
```

```python
        CMS.dao.close_connection(connection)


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

Util.py

```python
import CMS.dao
import CMS.exception

class CourierServiceDb:
    connection = CMS.dao.connect_to_sql_server()

    def __init__(self):
        self.connection = CMS.dao.connect_to_sql_server()

    def cancelOrder(self, tracking_number):
        try:
            cursor = self.connection.cursor()


            sql_query = """UPDATE Couriers
                    SET Status = 'Cancelled'
                    WHERE TrackingNumber = ?"""


            cursor.execute(sql_query, (tracking_number,))


            self.connection.commit()

            print("Order cancelled successfully.")
        except Exception as ex:
            print(f"Error cancelling order: {ex}")
        finally:
            cursor.close()

    def getCouriersByEmployee(self, employee_id):
        try:
            cursor = self.connection.cursor()


            sql_query = """SELECT *
                    FROM Couriers
                    WHERE EmployeeID = ?"""
```

```python
            cursor.execute(sql_query, (employee_id,))


            couriers = cursor.fetchall()

            print("Couriers retrieved successfully.")
            return couriers
        except exception.InvalidEmployeeIdException as ex:
            print(f"Error retrieving assigned orders: {ex}")
        except Exception as ex:
            print(f"Error retrieving assigned orders: {ex}")
        finally:
            cursor.close()

    def addCourierStaff(self, empID, name, email, contact_number, role, salary):
        try:
            cursor = self.connection.cursor()


            sql_query = """INSERT INTO Employees (EmployeeID,Name, Email, ContactNumber, Role, Salary)
                        VALUES (?,?, ?, ?, ?, ?)"""


            cursor.execute(sql_query, (empID, name, email, contact_number, role, salary))


            self.connection.commit()

            print("Courier staff added successfully.")
        except Exception as ex:
            print(f"Error adding courier staff: {ex}")
        finally:
            cursor.close()

    def insertOrder(self, courierID, sender_name, sender_address, receiver_name, receiver_address, weight, status,
                tracking_number, delivery_date, location_id, employee_id, service_id):
        try:
            cursor = self.connection.cursor()


            sql_query = """INSERT INTO Couriers (CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, LocationID, EmployeeID, ServiceID)
                        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"""
```

```python
            cursor.execute(sql_query,
                    (courierID, sender_name, sender_address, receiver_name, receiver_address,
weight, status,
                     tracking_number, delivery_date, location_id, employee_id, service_id))


            self.connection.commit()

            print("Order inserted successfully.")
        except Exception as ex:
            print(f"Error inserting order: {ex}")
        finally:
            cursor.close()

    def updateCourierStatus(self, trackingNumber, newStatus):
        try:
            cursor = self.connection.cursor()


            sql_query = """UPDATE Couriers
                    SET Status = ?
                    WHERE TrackingNumber = ?"""


            cursor.execute(sql_query, (newStatus, trackingNumber))


            self.connection.commit()

            print("Order cancelled successfully.")
        except exception.TrackingNumberNotFoundException as ex:
            print(f"Error cancelling order: {ex}")
        except Exception as ex:
            print(f"Error cancelling order: {ex}")
        finally:
            cursor.close()

    def retrieveDeliveryHistory(self, trackingNumber):
        try:
            cursor = self.connection.cursor()


            sql_query = """SELECT *
                    FROM Couriers
                    WHERE TrackingNumber = ?"""
```

```python
            cursor.execute(sql_query, (trackingNumber,))

            delivery_history = cursor.fetchall()

            print("Delivery history retrieved successfully.")
            return delivery_history
        except Exception as ex:
            print(f"Error retrieving delivery history: {ex}")
        finally:
            cursor.close()

    def generateShipmentStatusReport(self):
        try:
            cursor = self.connection.cursor()


            sql_query = """SELECT TrackingNumber, Status
                    FROM Couriers"""


            cursor.execute(sql_query)

            shipment_status_report = cursor.fetchall()

            print("Shipment status report generated successfully.")
            return shipment_status_report
        except Exception as ex:
            print(f"Error generating shipment status report: {ex}")
        finally:
            cursor.close()

    def generateRevenueReport(self):
        try:
            cursor = self.connection.cursor()


            sql_query = """SELECT SUM(Amount) as TotalRevenue
                    FROM Payments"""


            cursor.execute(sql_query)


            total_revenue = cursor.fetchone()[0]

            print("Revenue report generated successfully.")
            return total_revenue
        except Exception as ex:
```

```python
        print(f"Error generating revenue report: {ex}")
    finally:
        cursor.close()
```

Entity.py

```python
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

    def get_userID(self):
        return self.__userID

    def set_userID(self, userID):
        self.__userID = userID

    def get_userName(self):
        return self.__userName

    def set_userName(self, userName):
        self.__userName = userName

    def get_email(self):
        return self.__email

    def set_email(self, email):
        self.__email = email

    def get_password(self):
        return self.__password

    def set_password(self, password):
        self.__password = password

    def get_contactNumber(self):
        return self.__contactNumber

    def set_contactNumber(self, contactNumber):
        self.__contactNumber = contactNumber

    def get_address(self):
        return self.__address
```

```python
    def set_address(self, address):
        self.__address = address

    def __str__(self):
        return f"UserID: {self.__userID}, UserName: {self.__userName}, Email: {self.__email},
Password: {self.__password}, ContactNumber: {self.__contactNumber}, Address: {self.__address}"


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

    def get_courierID(self):
        return self.__courierID

    def set_courierID(self, courierID):
        self.__courierID = courierID

    def get_senderName(self):
        return self.__senderName

    def set_senderName(self, senderName):
        self.__senderName = senderName

    def get_senderAddress(self):
        return self.__senderAddress

    def set_senderAddress(self, senderAddress):
        self.__senderAddress = senderAddress

    def get_receiverName(self):
        return self.__receiverName

    def set_receiverName(self, receiverName):
        self.__receiverName = receiverName
```

```python
    def get_receiverAddress(self):
        return self.__receiverAddress

    def set_receiverAddress(self, receiverAddress):
        self.__receiverAddress = receiverAddress

    def get_weight(self):
        return self.__weight

    def set_weight(self, weight):
        self.__weight = weight

    def get_status(self):
        return self.__status

    def set_status(self, status):
        self.__status = status

    def get_trackingNumber(self):
        return self.__trackingNumber

    def set_trackingNumber(self, trackingNumber):
        self.__trackingNumber = trackingNumber

    def get_deliveryDate(self):
        return self.__deliveryDate

    def set_deliveryDate(self, deliveryDate):
        self.__deliveryDate = deliveryDate

    def get_userId(self):
        return self.__userId

    def set_userId(self, userId):
        self.__userId = userId

    def __str__(self):
        return f"CourierID: {self.__courierID}, SenderName: {self.__senderName}, SenderAddress:
{self.__senderAddress}, ReceiverName: {self.__receiverName}, ReceiverAddress:
{self.__receiverAddress}, Weight: {self.__weight}, Status: {self.__status}, TrackingNumber:
{self.__trackingNumber}, DeliveryDate: {self.__deliveryDate}, UserID: {self.__userId}"


class Employee:
    def __init__(self, employeeID, employeeName, email, contactNumber, role, salary):
        self.__employeeID = employeeID
        self.__employeeName = employeeName
        self.__email = email
```

```python
        self.__contactNumber = contactNumber
        self.__role = role
        self.__salary = salary

    def get_employeeID(self):
        return self.__employeeID

    def set_employeeID(self, employeeID):
        self.__employeeID = employeeID

    def get_employeeName(self):
        return self.__employeeName

    def set_employeeName(self, employeeName):
        self.__employeeName = employeeName

    def get_email(self):
        return self.__email

    def set_email(self, email):
        self.__email = email

    def get_contactNumber(self):
        return self.__contactNumber

    def set_contactNumber(self, contactNumber):
        self.__contactNumber = contactNumber

    def get_role(self):
        return self.__role

    def set_role(self, role):
        self.__role = role

    def get_salary(self):
        return self.__salary

    def set_salary(self, salary):
        self.__salary = salary

    def __str__(self):
        return f"EmployeeID: {self.__employeeID}, EmployeeName: {self.__employeeName}, Email:
{self.__email}, ContactNumber: {self.__contactNumber}, Role: {self.__role}, Salary: {self.__salary}"


class Location:
    def __init__(self, LocationID, LocationName, Address):
        self.__LocationID = LocationID
```

```python
        self.__LocationName = LocationName
        self.__Address = Address

    def get_LocationID(self):
        return self.__LocationID

    def set_LocationID(self, LocationID):
        self.__LocationID = LocationID

    def get_LocationName(self):
        return self.__LocationName

    def set_LocationName(self, LocationName):
        self.__LocationName = LocationName

    def get_Address(self):
        return self.__Address

    def set_Address(self, Address):
        self.__Address = Address

    def __str__(self):
        return f"LocationID: {self.__LocationID}, LocationName: {self.__LocationName}, Address: {self.__Address}"


class CourierCompany:
    def __init__(self, companyName):
        self.__companyName = companyName
        self.__courierDetails = []
        self.__employeeDetails = []
        self.__locationDetails = []

    def get_companyName(self):
        return self.__companyName

    def set_companyName(self, companyName):
        self.__companyName = companyName

    def add_courier(self, courier):
        self.__courierDetails.append(courier)

    def remove_courier(self, courier):
        self.__courierDetails.remove(courier)

    def add_employee(self, employee):
        self.__employeeDetails.append(employee)
```

```python
    def remove_employee(self, employee):
        self.__employeeDetails.remove(employee)

    def add_location(self, location):
        self.__locationDetails.append(location)

    def remove_location(self, location):
        self.__locationDetails.remove(location)

    def __str__(self):
        return f"CompanyName: {self.__companyName}, CourierDetails: {self.__courierDetails}, EmployeeDetails: {self.__employeeDetails}, LocationDetails: {self.__locationDetails}"


class Payment:
    def __init__(self, PaymentID, CourierID, LocationID, Amount, PaymentDate, EmployeeID):
        self.__PaymentID = PaymentID
        self.__CourierID = CourierID
        self.__LocationID = LocationID
        self.__Amount = Amount
        self.__PaymentDate = PaymentDate
        self.__EmployeeID = EmployeeID

    def get_PaymentID(self):
        return self.__PaymentID

    def set_PaymentID(self, PaymentID):
        self.__PaymentID = PaymentID

    def get_CourierID(self):
        return self.__CourierID

    def set_CourierID(self, CourierID):
        self.__CourierID = CourierID

    def get_LocationID(self):
        return self.__LocationID

    def set_LocationID(self, LocationID):
        self.__LocationID = LocationID

    def get_Amount(self):
        return self.__Amount

    def set_Amount(self, Amount):
        self.__Amount = Amount

    def get_PaymentDate(self):
```

```python
        return self.__PaymentDate

    def set_PaymentDate(self, PaymentDate):
        self.__PaymentDate = PaymentDate

    def get_EmployeeID(self):
        return self.__EmployeeID

    def set_EmployeeID(self, EmployeeID):
        self.__EmployeeID = EmployeeID

    def __str__(self):
        return f"PaymentID: {self.__PaymentID}, CourierID: {self.__CourierID}, LocationID:
{self.__LocationID}, Amount: {self.__Amount}, PaymentDate: {self.__PaymentDate}, EmployeeID:
{self.__EmployeeID}"
```

Dao.py

```python
import pyodbc

def connect_to_sql_server():
    try:
        conn = pyodbc.connect('Driver={SQL Server};'
                    'Server=DESKTOP-A08GADU\SQLEXPRESS01;'
                    'Database=Courier;'
                    'Trusted_Connection=yes;')
        print("Connected Successfully")
        return conn
    except pyodbc.Error as ex:
        print(f"Error: {ex}")


def close_connection(conn):
    conn.close()
    print("Connection closed.")
```

Exception.py

```python
from abc import ABC, abstractmethod


class TrackingNumberNotFoundException(Exception):
    pass


class InvalidEmployeeIdException(Exception):
    pass
```

```python
class ICourierUserService(ABC):
    @abstractmethod
    def placeOrder(self, courierObj):
        pass

    @abstractmethod
    def getOrderStatus(self, trackingNumber):
        pass

    @abstractmethod
    def cancelOrder(self, trackingNumber):
        pass

    @abstractmethod
    def getAssignedOrder(self, courierStaffId):
        pass


class ICourierAdminService(ABC):
    @abstractmethod
    def addCourierStaff(self, name, contactNumber):
        pass
```

```
Connected Successfully

Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. Add courier staff (Admin)
6. Generate report
7. Exit
Enter your choice: 1
Enter your courier ID: 225
Enter sender's name: francis
Enter sender's address: fgftyftf,gfgyfty
Enter receiver's name: john
Enter receiver's address: ftyftrffuyfufkikkkk
Enter weight: 35
Enter tracking number: 321
Enter delivery date (YYYY-MM-DD): 2024-06-06
Enter location ID: 2
Enter employee ID: 3
Enter service ID: 2
Order inserted successfully.
```

```
Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. Add courier staff (Admin)
6. Generate report
7. Exit
Enter your choice: 2
Enter tracking number: 321
Delivery history retrieved successfully.
Order status for tracking number 321: [(225, 'francis', 'fgftyftf,gfgyfty', 'john', 'ftyftrffuyfufkikkkk', Decimal('35.00'), 'Processing', '321',
  '2024-06-06', 2, 3, 2)]
```

```
Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. Add courier staff (Admin)
6. Generate report
7. Exit
Enter your choice: 3
Enter tracking number: 321
Order cancelled successfully.
```

```
Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. Add courier staff (Admin)
6. Generate report
7. Exit
Enter your choice: 4
Enter employee ID: 2
Couriers retrieved successfully.
Couriers handled by Employee ID: 2
(1, 'Rajesh Kumar', '12 Gandhi Nagar, Chennai', 'Ananya Singh', '78 Vindhya Nagar, Coimbatore', Decimal('2.50'), 'In Transit', 'TN123456',
 '2024-03-01', 1, 2, 1)
(2, 'Priya Sharma', '34 Kaveri Street, Bangalore', 'Amit Patel', '56 Krishna Lane, Hyderabad', Decimal('1.80'), 'Delivered', 'TN789012', '2024-03-02',
 1, 2, 2)
(5, 'Sara Khan', '90 Yamuna Road, Delhi', 'David Lee', '67 Forest Lane, Pune', Decimal('2.00'), 'In Transit', 'TN234567', '2024-03-04', 1, 2, 2)
(7, 'Emma Wilson', '45 Lake Avenue, Kolkata', 'Michael Johnson', '23 Park Street, Mumbai', Decimal('3.50'), 'In Transit', 'TN456789', '2024-03-06', 3,
 2, 2)
(8, 'David Lee', '67 Forest Lane, Pune', 'Sara Khan', '90 Yamuna Road, Delhi', Decimal('4.50'), 'In Transit', 'TN567890', '2024-03-07', 3, 2, 1)
```

```
Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. Add courier staff (Admin)
6. Generate report
7. Exit
Enter your choice: 5
Enter staff ID:12
Enter staff name: Rahul
Enter staff email: rahul@123.com
Enter staff contact number: 9699456446
Enter staff role: Manager
Enter staff salary: 35000
Courier staff added successfully.
```

```
Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. Add courier staff (Admin)
6. Generate report
7. Exit
Enter your choice: 6
Shipment status report generated successfully.
Shipment status report:
('TN123456', 'In Transit')
('TN789012', 'Delivered')
('TN345678', 'In Transit')
('TN345679', 'In Transit')
('TN234567', 'In Transit')
('TN345688', 'Delivered')
('TN456789', 'In Transit')
('TN567890', 'In Transit')
('TN678901', 'Delivered')
('123654', 'Cancelled')
('321', 'Cancelled')
Revenue report generated successfully.
Total Revenue: 44601.50
```

```
Courier Service Menu:
1. Place an order
2. Get order status
3. Cancel an order
4. Get assigned orders
5. Add courier staff (Admin)
6. Generate report
7. Exit
Enter your choice: 7
Exiting program...
Connection closed.

Process finished with exit code 0
```

SQL Assignment Courier Management System

Student Name: Sugandan Elangovan

Task1 :

Database Design Design a SQL schema for a Courier Management System with tables for Customers, Couriers, Orders, and Parcels. Define the relationships between these tables using appropriate foreign keys.

```sql
CREATE TABLE Users (
    UserID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
    Password VARCHAR(255),
    ContactNumber VARCHAR(20),
    Address TEXT
);


CREATE TABLE CourierServices (
    ServiceID INT PRIMARY KEY,
    ServiceName VARCHAR(100),
    Cost DECIMAL(8, 2)
);


CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
    ContactNumber VARCHAR(20),
    Role VARCHAR(50),
    Salary DECIMAL(10, 2)
);


CREATE TABLE Locations (
    LocationID INT PRIMARY KEY,
    LocationName VARCHAR(100),
    Address TEXT
);

CREATE TABLE Couriers (
    CourierID INT PRIMARY KEY,
    SenderName VARCHAR(255),
    SenderAddress TEXT,
    ReceiverName VARCHAR(255),
    ReceiverAddress TEXT,
    Weight DECIMAL(5, 2),
    Status VARCHAR(50),
    TrackingNumber VARCHAR(20) UNIQUE,
    DeliveryDate DATE
    LocationID INT,
    EmployeeID INT,
    ServiceID INT,
    CONSTRAINT FK_Couriers_Location FOREIGN KEY (LocationID) REFERENCES Locations(LocationID),
    CONSTRAINT FK_Couriers_Employee FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),
    CONSTRAINT FK_Couriers_Service FOREIGN KEY (ServiceID) REFERENCES CourierServices(ServiceID)
);
```

```sql
CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY,
    CourierID INT,
    LocationID INT,
    Amount DECIMAL(10, 2),
    PaymentDate DATE,
        EmployeeID INT,
    FOREIGN KEY (CourierID) REFERENCES Couriers(CourierID),
    FOREIGN KEY (LocationID) REFERENCES Locations(LocationID),
        FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID),

);


CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE,
    CONSTRAINT FK_Orders_Customers FOREIGN KEY (CustomerID) REFERENCES Users(UserID)
);


CREATE TABLE Parcels (
    ParcelID INT PRIMARY KEY,
    OrderID INT,
    CourierID INT,
    ServiceID INT,
    Weight DECIMAL(5, 2),
    Status VARCHAR(50),
    TrackingNumber VARCHAR(20) UNIQUE,
    DeliveryDate DATE,
        EmployeeID INT,
    CONSTRAINT FK_Parcels_Orders FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    CONSTRAINT FK_Parcels_Couriers FOREIGN KEY (CourierID) REFERENCES Couriers(CourierID),
    CONSTRAINT FK_Parcels_Services FOREIGN KEY (ServiceID) REFERENCES CourierServices(ServiceID),
        CONSTRAINT FK_Parcels_Employee FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
);
```

Messages
  Commands completed successfully.

  Completion time: 2024-03-05T10:03:29.9539297+05:30

```sql
INSERT INTO Users (UserID, Name, Email, Password, ContactNumber, Address)
VALUES(1, 'Rajesh Kumar', 'rajesh.kumar@email.com', 'password123', '9876543210', '12 Gandhi Nagar,
Chennai'),
(2, 'Priya Sharma', 'priya.sharma@email.com', 'password456', '8765432109', '34 Kaveri Street,
Bangalore'),
(3, 'Amit Patel', 'amit.patel@email.com', 'password789', '7654321098', '56 Krishna Lane, Hyderabad'),
(4, 'Ananya Singh', 'ananya.singh@email.com', 'passwordabc', '6543210987', '78 Vindhya Nagar,
Coimbatore');

INSERT INTO CourierServices (ServiceID, ServiceName, Cost)
VALUES(1, 'Standard', 10.00),
(2, 'Express', 15.00);


INSERT INTO Employees (EmployeeID, Name, Email, ContactNumber, Role, Salary)
```

```sql
VALUES(1, 'Manager1', 'manager1@email.com', '1112223333', 'Manager', 50000.00),
(2, 'DeliveryPerson1', 'delivery1@email.com', '4445556666', 'Delivery Person', 30000.00),
(3, 'DeliveryPerson2', 'delivery2@email.com', '5556667777', 'Delivery Person', 30000.00);

INSERT INTO Employees (EmployeeID, Name, Email, ContactNumber, Role, Salary)
VALUES(4, 'John cena', 'john.cena@email.com', '1234567890', 'Manager', 60000.00),
(5, 'Authur Johnson', 'authur.johnson@email.com', '9876543210', 'Clerk', 45000.00);


INSERT INTO Locations (LocationID, LocationName, Address)
VALUES(1, 'Warehouse1', '789 Storage St, Chennai'),
(2, 'Warehouse2', '456 Logistics Ave, Bangalore'),
(3, 'Warehouse3', '123 Distribution Rd, Hyderabad');


INSERT INTO Couriers (CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight,
Status, TrackingNumber, DeliveryDate, LocationID,EmployeeID,ServiceID)
VALUES(1, 'Rajesh Kumar', '12 Gandhi Nagar, Chennai', 'Ananya Singh', '78 Vindhya Nagar, Coimbatore',
2.5, 'In Transit', 'TN123456', '2024-03-01',1,2,1),
(2, 'Priya Sharma', '34 Kaveri Street, Bangalore', 'Amit Patel', '56 Krishna Lane, Hyderabad', 1.8,
'Delivered', 'TN789012', '2024-03-02',1,2,2),
(3, 'Amit Patel', '56 Krishna Lane, Hyderabad', 'Priya Sharma', '34 Kaveri Street, Bangalore', 3.0,
'In Transit', 'TN345678', '2024-03-03',2,3,2),
(4, 'Ananya Singh', '78 Vindhya Nagar, Coimbatore', 'Rajesh Kumar', '12 Gandhi Nagar, Chennai', 4.0,
'In Transit', 'TN345679', '2024-03-05',2,3,2);

INSERT INTO Payments (PaymentID, CourierID, LocationID, Amount, PaymentDate,EmployeeID)
VALUES(1, 1, 1, 10000.00, '2024-03-03',2),
(2, 2, 2, 1500.00, '2024-03-04',2),
(3, 3, 3, 1200.50, '2024-03-05',3);

---

INSERT INTO Orders (OrderID, CustomerID, OrderDate)
VALUES(1, 1, '2024-02-01'),
(2, 2, '2024-02-02'),
(3, 3, '2024-02-03');


INSERT INTO Parcels (ParcelID, OrderID, CourierID, ServiceID, Weight, Status, TrackingNumber,
DeliveryDate,EmployeeID)
VALUES(1, 1, 1, 1, 2.5, 'In Transit', 'TN123456', '2024-03-01',2),
(2, 2, 2, 2, 1.8, 'Delivered', 'TN789012', '2024-03-02',2),
(3, 3, 3, 1, 3.0, 'In Transit', 'TN345678', '2024-03-03',3);


INSERT INTO Users (UserID, Name, Email, Password, ContactNumber, Address)
VALUES
(5, 'Sara Khan', 'sara.khan@email.com', 'passwordxyz', '5432109876', '90 Yamuna Road, Delhi'),
(6, 'Michael Johnson', 'michael.johnson@email.com', 'password123', '6547893210', '23 Park Street,
Mumbai'),
(7, 'Emma Wilson', 'emma.wilson@email.com', 'password456', '7896543210', '45 Lake Avenue, Kolkata'),
(8, 'David Lee', 'david.lee@email.com', 'password789', '9876543210', '67 Forest Lane, Pune'),
(9, 'Sophia Garcia', 'sophia.garcia@email.com', 'passwordabc', '8765432109', '89 Ocean View, Goa');



INSERT INTO Orders (OrderID, CustomerID, OrderDate)
VALUES
(4, 4, '2024-02-04'),
(5, 5, '2024-02-05'),
(6, 6, '2024-02-06'),
(7, 7, '2024-02-07'),
(8, 8, '2024-02-08'),
(9, 1, '2024-02-08');

INSERT INTO Couriers (CourierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight,
Status, TrackingNumber, DeliveryDate, LocationID, EmployeeID, ServiceID)
VALUES
```

```sql
(5, 'Sara Khan', '90 Yamuna Road, Delhi', 'David Lee', '67 Forest Lane, Pune', 2.0, 'In Transit',
'TN234567', '2024-03-04', 1, 2, 2),
(6, 'Michael Johnson', '23 Park Street, Mumbai', 'Emma Wilson', '45 Lake Avenue, Kolkata', 1.5,
'Delivered', 'TN345688', '2024-03-05', 2, 3, 1),
(7, 'Emma Wilson', '45 Lake Avenue, Kolkata', 'Michael Johnson', '23 Park Street, Mumbai', 3.5, 'In
Transit', 'TN456789', '2024-03-06', 3, 2, 2),
(8, 'David Lee', '67 Forest Lane, Pune', 'Sara Khan', '90 Yamuna Road, Delhi', 4.5, 'In Transit',
'TN567890', '2024-03-07', 3, 2, 1),
(9, 'Rajesh Kumar', '12 Gandhi Nagar, Chennai', 'Sara Khan', '90 Yamuna Road, Delhi', 2.8,
'Delivered', 'TN678901', '2024-03-08', 2, 3, 2);


INSERT INTO Payments (PaymentID, CourierID, LocationID, Amount, PaymentDate, EmployeeID)
VALUES
(4, 4, 1, 1500.00, '2024-03-06', 2),
(5, 5, 2, 1000.00, '2024-03-07', 3),
(6, 6, 3, 25000.00, '2024-03-08', 2),
(7, 7, 1, 2000.00, '2024-03-09', 3),
(8, 8, 2, 1200.50, '2024-03-10', 2),
(9, 9, 2, 1200.50, '2024-03-10', 2);


INSERT INTO Parcels (ParcelID, OrderID, CourierID, ServiceID, Weight, Status, TrackingNumber,
DeliveryDate, EmployeeID)
VALUES
(4, 4, 4, 1, 4.0, 'In Transit', 'TN456789', '2024-03-04', 2),
(5, 5, 5, 2, 2.5, 'Delivered', 'TN567890', '2024-03-05', 3),
(6, 6, 6, 1, 3.2, 'In Transit', 'TN678901', '2024-03-06', 2),
(7, 7, 7, 2, 2.0, 'Delivered', 'TN789013', '2024-03-07', 2),
(8, 8, 8, 1, 3.5, 'In Transit', 'TN89012', '2024-03-08', 2),
(9, 9, 9, 1, 3.5, 'In Transit', 'TN89014', '2024-03-08', 2);
```

Task 2: Select,Where

1. List all customers

```sql
SELECT * FROM Users;
```

Results | Messages

	UserID	Name	Email	Password	ContactNumber	Address
1	1	Rajesh Kumar	rajesh.kumar@email.com	password123	9876543210	12 Gandhi Nagar, Chennai
2	2	Priya Sharma	priya.sharma@email.com	password456	8765432109	34 Kaveri Street, Bangalore
3	3	Amit Patel	amit.patel@email.com	password789	7654321098	56 Krishna Lane, Hyderabad
4	4	Ananya Singh	ananya.singh@email.com	passwordabc	6543210987	78 Vindhya Nagar, Coimbatore
5	5	Sara Khan	sara.khan@email.com	passwordxyz	5432109876	90 Yamuna Road, Delhi
6	6	Michael Johnson	michael.johnson@email.com	password123	6547893210	23 Park Street, Mumbai
7	7	Emma Wilson	emma.wilson@email.com	password456	7896543210	45 Lake Avenue, Kolkata
8	8	David Lee	david.lee@email.com	password789	9876543210	67 Forest Lane, Pune
9	9	Sophia Garcia	sophia.garcia@email.com	passwordabc	8765432109	89 Ocean View, Goa

2. List all orders for a specific customer:

```sql
SELECT * FROM Orders WHERE CustomerID = 1;
```

Results | Messages

	OrderID	CustomerID	OrderDate
1	1	1	2024-03-01
2	9	1	2024-03-08

3. List all couriers:

```sql
SELECT * FROM Couriers;
```

4. List all packages for a specific order:

```sql
SELECT * FROM Orders WHERE OrderID = 2;
```


	OrderID	CustomerID	OrderDate
1	2	2	2024-02-04

5. List all deliveries for a specific courier:

```sql
SELECT * FROM Parcels WHERE CourierID = 1;
```


	ParcelID	OrderID	CourierID	ServiceID	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID
1	1	1	1	1	2.50	In Transit	TN123456	2024-03-01	2

6. List all undelivered packages:

```sql
SELECT * FROM Parcels WHERE Status = 'In Transit'
```


	ParcelID	OrderID	CourierID	ServiceID	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID
1	1	1	1	1	2.50	In Transit	TN123456	2024-03-01	2
2	3	3	3	1	3.00	In Transit	TN345678	2024-03-03	3
3	4	4	4	1	4.00	In Transit	TN456789	2024-03-04	2
4	6	6	6	1	3.20	In Transit	TN678901	2024-03-06	2
5	8	8	8	1	3.50	In Transit	TN89012	2024-03-08	2
6	9	9	9	1	3.50	In Transit	TN89014	2024-03-08	2

7. List all packages that are scheduled for delivery today:

```sql
SELECT * FROM Couriers WHERE DeliveryDate = CAST(GETDATE() AS DATE);
```


	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
1	4	Sender4	SenderAddress4	Receiver4	ReceiverAddress4	4.00	In Transit	TN345679	2024-03-05	2	3	2
2	6	Sender6	SenderAddress6	Receiver6	ReceiverAddress6	1.50	Delivered	TN345688	2024-03-05	2	3	1

8. List all packages with a specific status:

```sql
SELECT * FROM Parcels WHERE Status = 'Delivered';
```


	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
1	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2024-03-02	1	2	2
2	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50	Delivered	TN345688	2024-03-05	2	3	1
3	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08	2	3	2

9. Calculate the total number of packages for each courier.

```sql
SELECT CourierID, COUNT(*) AS TotalPackages FROM Parcels GROUP BY CourierID;
```


	CourierID	TotalPackages
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1

10. Find the average delivery time for each courier

```sql
SELECT CourierID,AVG(DATEDIFF(DAY, o.OrderDate, c.DeliveryDate)) AS AvgDeliveryTime
FROM Couriers c INNER JOIN Orders o ON c.CourierID=o.OrderID AND c.Status = 'Delivered' GROUP BY
c.CourierID;
```

	CourierID	AvgDeliveryTime
1	2	29
2	6	28
3	9	29

11. List all packages with a specific weight range:

```sql
SELECT * FROM Parcels WHERE Weight BETWEEN 1.0 AND 2.0;
```

	ParcelID	OrderID	CourierID	ServiceID	Weight	Status	TrackingNumber	DeliveryDate	EmployeeID
1	2	2	2	2	1.80	Delivered	TN789012	2024-03-02	2
2	7	7	7	2	2.00	Delivered	TN789013	2024-03-07	2

12. Retrieve employees whose names contain 'John'

```sql
SELECT * FROM Employees WHERE Name LIKE '%John%';
```

	EmployeeID	Name	Email	ContactNumber	Role	Salary
1	4	John cena	john.cena@email.com	1234567890	Manager	60000.00
2	5	Authur Johnson	authur.johnson@email.com	9876543210	Clerk	45000.00

13. Retrieve all courier records with payments greater than $50

```sql
SELECT * FROM Payments WHERE Amount > 50.00;
```

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID
2	2	2	2	1500.00	2024-03-04	2
3	3	3	3	1200.50	2024-03-05	3
4	4	4	1	1500.00	2024-03-06	2
5	5	5	2	1000.00	2024-03-07	3
6	6	6	3	25000.00	2024-03-08	2
7	7	7	1	2000.00	2024-03-09	3
8	8	8	2	1200.50	2024-03-10	2
9	9	9	2	1200.50	2024-03-10	2

Task 3: GroupBy, Aggregate Functions, Having, Order By, where

14. Find the total number of couriers handled by each employee.

```sql
SELECT e.EmployeeID, e.Name, COUNT(p.CourierID) AS TotalCouriersHandled FROM Employees e
LEFT JOIN Couriers p ON e.EmployeeID = p.EmployeeID GROUP BY e.EmployeeID, e.Name;
```

	EmployeeID	Name	TotalCouriersHandled
1	1	Manager1	0
2	2	DeliveryPerson1	5
3	3	DeliveryPerson2	4
4	4	John cena	0
5	5	Authur Johnson	0

15. Calculate the total revenue generated by each location

```sql
SELECT l.LocationID, l.LocationName, SUM(py.Amount) AS TotalRevenue FROM Locations l
LEFT JOIN Payments py ON l.LocationID = py.LocationID GROUP BY l.LocationID, l.LocationName;
```

Results | Messages

	LocationID	LocationName	TotalRevenue
1	1	Warehouse1	13500.00
2	2	Warehouse2	4901.00
3	3	Warehouse3	26200.50

16. Find the total number of couriers delivered to each location.

```sql
SELECT l.LocationID, l.LocationName, COUNT(p.CourierID) AS TotalCouriersDelivered
FROM Locations l LEFT JOIN Couriers p ON l.LocationID = p.LocationID
WHERE p.Status = 'Delivered' GROUP BY l.LocationID, l.LocationName;
```

Results | Messages

	LocationID	LocationName	TotalCouriersDelivered
1	1	Warehouse1	1
2	2	Warehouse2	2

17. Find the courier with the highest average delivery time:

```sql
SELECT TOP 1 CourierID, AVG(DATEDIFF(DAY, p.DeliveryDate, o.OrderDate)) AS AvgDeliveryTime FROM
Parcels p, Orders o
WHERE Status = 'Delivered'  AND p.OrderID = o.OrderID GROUP BY CourierID ORDER BY AvgDeliveryTime
DESC;
```

Results | Messages

	CourierID	AvgDeliveryTime
1	5	-29

18. Find Locations with Total Payments Less Than a Certain Amount

```sql
SELECT LocationID, LocationName FROM Locations WHERE LocationID
IN (SELECT LocationID FROM Payments GROUP BY LocationID HAVING SUM(Amount) < 5000);
```

Results | Messages

	LocationID	LocationName
1	2	Warehouse2

19. Calculate Total Payments per Location

```sql
SELECT LocationID, SUM(Amount) AS TotalPayments FROM Payments GROUP BY LocationID;
```

Results | Messages

	LocationID	LocationName	TotalRevenue
1	1	Warehouse1	13500.00
2	2	Warehouse2	4901.00
3	3	Warehouse3	26200.50

20. Retrieve couriers who have received payments totaling more than $1000 in a specific location (LocationID = X):

```sql
SELECT p.CourierID, c.SenderName, c.ReceiverName, p.LocationID, SUM(p.Amount) AS TotalPayments
FROM Payments p JOIN Couriers c ON p.CourierID = c.CourierID WHERE p.LocationID = 1
```

```sql
GROUP BY p.CourierID, c.SenderName, c.ReceiverName, p.LocationID HAVING SUM(p.Amount) > 1000;
```

	CourierID	SenderName	ReceiverName	LocationID	TotalPayments
1	1	Rajesh Kumar	Ananya Singh	1	10000.00
2	4	Ananya Singh	Rajesh Kumar	1	1500.00
3	7	Emma Wilson	Michael Johnson	1	2000.00

21. Retrieve couriers who have received payments totaling more than $1000 after a certain date (PaymentDate > 'YYYY-MM-DD'):

```sql
SELECT p.CourierID, c.SenderName, c.ReceiverName, SUM(p.Amount) AS TotalPayments
FROM Payments p JOIN Couriers c ON p.CourierID = c.CourierID WHERE p.PaymentDate > '2024-03-01'
GROUP BY p.CourierID, c.SenderName, c.ReceiverName HAVING SUM(p.Amount) > 1000;
```

	CourierID	SenderName	ReceiverName	TotalPayments
1	1	Rajesh Kumar	Ananya Singh	10000.00
2	2	Priya Sharma	Amit Patel	1500.00
3	3	Amit Patel	Priya Sharma	1200.50
4	4	Ananya Singh	Rajesh Kumar	1500.00
5	6	Michael Johnson	Emma Wilson	25000.00
6	7	Emma Wilson	Michael Johnson	2000.00
7	8	David Lee	Sara Khan	1200.50
8	9	Rajesh Kumar	Sara Khan	1200.50

22. Retrieve locations where the total amount received is more than $5000 before a certain date (PaymentDate > 'YYYY-MM-DD')

```sql
SELECT l.LocationID, l.LocationName, SUM(p.Amount) AS TotalAmountReceived FROM Locations l
JOIN Payments p ON l.LocationID = p.LocationID WHERE p.PaymentDate > '2024-03-01'
GROUP BY l.LocationID, l.LocationName HAVING SUM(p.Amount) > 5000;
```

	LocationID	LocationName	TotalAmountReceived
1	1	Warehouse1	13500.00
2	3	Warehouse3	26200.50

Task 4: Inner Join, Full Outer Join, Cross Join, Left Outer Join, Right Outer Join

23. Retrieve Payments with Courier Information

```sql
SELECT p.*, c.* FROM Payments p INNER JOIN Couriers c ON p.CourierID = c.CourierID;
```

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight
1	1	1	1	10000.00	2024-03-03	2	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50
2	2	2	2	1500.00	2024-03-04	2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80
3	3	3	3	1200.50	2024-03-05	3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00
4	4	4	1	1500.00	2024-03-06	2	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00
5	5	5	2	1000.00	2024-03-07	3	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00
6	6	6	3	25000.00	2024-03-08	2	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50
7	7	7	1	2000.00	2024-03-09	3	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50
8	8	8	2	1200.50	2024-03-10	2	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50
9	9	9	2	1200.50	2024-03-10	2	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
1	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2024-03-01	1	2	1
2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2024-03-02	1	2	2
3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00	In Transit	TN345678	2024-03-03	2	3	2
4	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679	2024-03-05	2	3	2
5	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00	In Transit	TN234567	2024-03-04	1	2	2
6	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50	Delivered	TN345688	2024-03-05	2	3	1
7	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789	2024-03-06	3	2	2
8	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2024-03-07	3	2	1
9	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08	2	3	2

24. Retrieve Payments with Location Information

SELECT p.*, l.* FROM Payments p INNER JOIN Locations l ON p.LocationID = l.LocationID;

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID	LocationID	LocationName	Address
1	1	1	1	10000.00	2024-03-03	2	1	Warehouse1	789 Storage St, Chennai
2	2	2	2	1500.00	2024-03-04	2	2	Warehouse2	456 Logistics Ave, Bangalore
3	3	3	3	1200.50	2024-03-05	3	3	Warehouse3	123 Distribution Rd, Hyderabad
4	4	4	1	1500.00	2024-03-06	2	1	Warehouse1	789 Storage St, Chennai
5	5	5	2	1000.00	2024-03-07	3	2	Warehouse2	456 Logistics Ave, Bangalore
6	6	6	3	25000.00	2024-03-08	2	3	Warehouse3	123 Distribution Rd, Hyderabad
7	7	7	1	2000.00	2024-03-09	3	1	Warehouse1	789 Storage St, Chennai
8	8	8	2	1200.50	2024-03-10	2	2	Warehouse2	456 Logistics Ave, Bangalore
9	9	9	2	1200.50	2024-03-10	2	2	Warehouse2	456 Logistics Ave, Bangalore

25. Retrieve Payments with Courier and Location Information

SELECT p.*, c.*, l.* FROM Payments p INNER JOIN Couriers c ON p.CourierID = c.CourierID
INNER JOIN Locations l ON p.LocationID = l.LocationID;

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight
1	1	1	1	10000.00	2024-03-03	2	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50
2	2	2	2	1500.00	2024-03-04	2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80
3	3	3	3	1200.50	2024-03-05	3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00
4	4	4	1	1500.00	2024-03-06	2	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00
5	5	5	2	1000.00	2024-03-07	3	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00
6	6	6	3	25000.00	2024-03-08	2	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50
7	7	7	1	2000.00	2024-03-09	3	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50
8	8	8	2	1200.50	2024-03-10	2	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50
9	9	9	2	1200.50	2024-03-10	2	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80

	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID	LocationID	LocationName	Address
1	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2024-03-01	1	2	1	1	Warehouse1	789 Storage St, Chennai
2	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2024-03-02	1	2	2	2	Warehouse2	456 Logistics Ave, Bangalore
3	Priya Sharma	34 Kaveri Street, Bangalore	3.00	In Transit	TN345678	2024-03-03	2	3	2	3	Warehouse3	123 Distribution Rd, Hyderabad
4	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679	2024-03-05	2	3	2	1	Warehouse1	789 Storage St, Chennai
5	David Lee	67 Forest Lane, Pune	2.00	In Transit	TN234567	2024-03-04	1	2	2	2	Warehouse2	456 Logistics Ave, Bangalore
6	Emma Wilson	45 Lake Avenue, Kolkata	1.50	Delivered	TN345688	2024-03-05	2	3	1	3	Warehouse3	123 Distribution Rd, Hyderabad
7	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789	2024-03-06	3	2	2	1	Warehouse1	789 Storage St, Chennai
8	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2024-03-07	3	2	1	2	Warehouse2	456 Logistics Ave, Bangalore
9	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08	2	3	2	2	Warehouse2	456 Logistics Ave, Bangalore

26. List all payments with courier details

SELECT p.*, c.* FROM Payments p INNER JOIN Couriers c ON p.CourierID = c.CourierID;

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight
1	1	1	1	10000.00	2024-03-03	2	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50
2	2	2	2	1500.00	2024-03-04	2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80
3	3	3	3	1200.50	2024-03-05	3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00
4	4	4	1	1500.00	2024-03-06	2	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00
5	5	5	2	1000.00	2024-03-07	3	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00
6	6	6	3	25000.00	2024-03-08	2	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50
7	7	7	1	2000.00	2024-03-09	3	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50
8	8	8	2	1200.50	2024-03-10	2	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50
9	9	9	2	1200.50	2024-03-10	2	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80

Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
2.50	In Transit	TN123456	2024-03-01	1	2	1
1.80	Delivered	TN789012	2024-03-02	1	2	2
3.00	In Transit	TN345678	2024-03-03	2	3	2
4.00	In Transit	TN345679	2024-03-05	2	3	2
2.00	In Transit	TN234567	2024-03-04	1	2	2
1.50	Delivered	TN345688	2024-03-05	2	3	1
3.50	In Transit	TN456789	2024-03-06	3	2	2
4.50	In Transit	TN567890	2024-03-07	3	2	1
2.80	Delivered	TN678901	2024-03-08	2	3	2

27. Total payments received for each courier

```
SELECT c.CourierID, c.SenderName, c.ReceiverName, SUM(p.Amount) AS TotalPayments
FROM Couriers c LEFT JOIN Payments p ON c.CourierID = p.CourierID GROUP BY c.CourierID, c.SenderName,
c.ReceiverName;
```

Results | Messages

	CourierID	SenderName	ReceiverName	TotalPayments
1	1	Rajesh Kumar	Ananya Singh	10000.00
2	2	Priya Sharma	Amit Patel	1500.00
3	3	Amit Patel	Priya Sharma	1200.50
4	4	Ananya Singh	Rajesh Kumar	1500.00
5	5	Sara Khan	David Lee	1000.00
6	6	Michael Johnson	Emma Wilson	25000.00
7	7	Emma Wilson	Michael Johnson	2000.00
8	8	David Lee	Sara Khan	1200.50
9	9	Rajesh Kumar	Sara Khan	1200.50

28. List payments made on a specific date

```
SELECT * FROM Payments WHERE PaymentDate = '2024-03-08';
```

Results | Messages

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID
1	6	6	3	25000.00	2024-03-08	2

29. Get Courier Information for Each Payment

```
SELECT p.*, c.* FROM Payments p LEFT JOIN Couriers c ON p.CourierID = c.CourierID;
```

Results | Messages

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight
1	1	1	1	10000.00	2024-03-03	2	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50
2	2	2	2	1500.00	2024-03-04	2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80
3	3	3	3	1200.50	2024-03-05	3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00
4	4	4	1	1500.00	2024-03-06	2	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00
5	5	5	2	1000.00	2024-03-07	3	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00
6	6	6	3	25000.00	2024-03-08	2	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50
7	7	7	1	2000.00	2024-03-09	3	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50
8	8	8	2	1200.50	2024-03-10	2	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50
9	9	9	2	1200.50	2024-03-10	2	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80

Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
In Transit	TN123456	2024-03-01	1	2	1
Delivered	TN789012	2024-03-02	1	2	2
In Transit	TN345678	2024-03-03	2	3	2
In Transit	TN345679	2024-03-05	2	3	2
In Transit	TN234567	2024-03-04	1	2	2
Delivered	TN345688	2024-03-05	2	3	1
In Transit	TN456789	2024-03-06	3	2	2
In Transit	TN567890	2024-03-07	3	2	1
Delivered	TN678901	2024-03-08	2	3	2

30. Get Payment Details with Location

```sql
SELECT p.*, l.* FROM Payments p LEFT JOIN Locations l ON p.LocationID = l.LocationID;
```

Results   Messages

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID	LocationID	LocationName	Address
1	1	1	1	10000.00	2024-03-03	2	1	Warehouse1	789 Storage St, Chennai
2	2	2	2	1500.00	2024-03-04	2	2	Warehouse2	456 Logistics Ave, Bangalore
3	3	3	3	1200.50	2024-03-05	3	3	Warehouse3	123 Distribution Rd, Hyderabad
4	4	4	1	1500.00	2024-03-06	2	1	Warehouse1	789 Storage St, Chennai
5	5	5	2	1000.00	2024-03-07	3	2	Warehouse2	456 Logistics Ave, Bangalore
6	6	6	3	25000.00	2024-03-08	2	3	Warehouse3	123 Distribution Rd, Hyderabad
7	7	7	1	2000.00	2024-03-09	3	1	Warehouse1	789 Storage St, Chennai
8	8	8	2	1200.50	2024-03-10	2	2	Warehouse2	456 Logistics Ave, Bangalore
9	9	9	2	1200.50	2024-03-10	2	2	Warehouse2	456 Logistics Ave, Bangalore

31. Calculating Total Payments for Each Courier

```sql
SELECT c.CourierID, c.SenderName, c.ReceiverName, SUM(p.Amount) AS TotalPayments FROM Couriers c
LEFT JOIN Payments p ON c.CourierID = p.CourierID GROUP BY c.CourierID, c.SenderName, c.ReceiverName;
```

Results   Messages

	CourierID	SenderName	ReceiverName	TotalPayments
1	1	Rajesh Kumar	Ananya Singh	10000.00
2	2	Priya Sharma	Amit Patel	1500.00
3	3	Amit Patel	Priya Sharma	1200.50
4	4	Ananya Singh	Rajesh Kumar	1500.00
5	5	Sara Khan	David Lee	1000.00
6	6	Michael Johnson	Emma Wilson	25000.00
7	7	Emma Wilson	Michael Johnson	2000.00
8	8	David Lee	Sara Khan	1200.50
9	9	Rajesh Kumar	Sara Khan	1200.50

32. List Payments Within a Date Range

```sql
SELECT * FROM Payments WHERE PaymentDate BETWEEN '2024-02-01' AND '2024-03-08';
```

Results   Messages

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID
1	1	1	1	10000.00	2024-03-03	2
2	2	2	2	1500.00	2024-03-04	2
3	3	3	3	1200.50	2024-03-05	3
4	4	4	1	1500.00	2024-03-06	2
5	5	5	2	1000.00	2024-03-07	3
6	6	6	3	25000.00	2024-03-08	2

33. Retrieve a list of all users and their corresponding courier records, including cases where there are no matches on either side

```sql
SELECT u.*, c.* FROM Users u FULL OUTER JOIN Couriers c ON u.name = c.SenderName;
```

	UserID	Name	Email	Password	ContactNumber	Address	CourierID	SenderName	SenderAddress	ReceiverName
1	1	Rajesh Kumar	rajesh.kumar@email.com	password123	9876543210	12 Gandhi Nagar, Chennai	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh
2	1	Rajesh Kumar	rajesh.kumar@email.com	password123	9876543210	12 Gandhi Nagar, Chennai	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan
3	2	Priya Sharma	priya.sharma@email.com	password456	8765432109	34 Kaveri Street, Bangalore	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel
4	3	Amit Patel	amit.patel@email.com	password789	7654321098	56 Krishna Lane, Hyderabad	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma
5	4	Ananya Singh	ananya.singh@email.com	passwordabc	6543210987	78 Vindhya Nagar, Coimbatore	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar
6	5	Sara Khan	sara.khan@email.com	passwordxyz	5432109876	90 Yamuna Road, Delhi	5	Sara Khan	90 Yamuna Road, Delhi	David Lee
7	6	Michael Johnson	michael.johnson@email.com	password123	6547893210	23 Park Street, Mumbai	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson
8	7	Emma Wilson	emma.wilson@email.com	password456	7896543210	45 Lake Avenue, Kolkata	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson
9	8	David Lee	david.lee@email.com	password789	9876543210	67 Forest Lane, Pune	8	David Lee	67 Forest Lane, Pune	Sara Khan
10	9	Sophia Garcia	sophia.garcia@email.com	passwordabc	8765432109	89 Ocean View, Goa	NULL	NULL	NULL	NULL

ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2024-03-01	1	2	1
90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08	2	3	2
56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2024-03-02	1	2	2
34 Kaveri Street, Bangalore	3.00	In Transit	TN345678	2024-03-03	2	3	2
12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679	2024-03-05	2	3	2
67 Forest Lane, Pune	2.00	In Transit	TN234567	2024-03-04	1	2	2
45 Lake Avenue, Kolkata	1.50	Delivered	TN345688	2024-03-05	2	3	1
23 Park Street, Mumbai	3.50	In Transit	TN456789	2024-03-06	3	2	2
90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2024-03-07	3	2	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

34. Retrieve a list of all couriers and their corresponding services, including cases where there are no matches on either side

```sql
SELECT c.*, cs.* FROM Couriers c FULL OUTER JOIN CourierServices cs ON c.ServiceID = cs.ServiceID;
```

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
1	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2024-03-01
2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2024-03-02
3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00	In Transit	TN345678	2024-03-03
4	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679	2024-03-05
5	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00	In Transit	TN234567	2024-03-04
6	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50	Delivered	TN345688	2024-03-05
7	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789	2024-03-06
8	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2024-03-07
9	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08

LocationID	EmployeeID	ServiceID	ServiceID	ServiceName	Cost
1	2	1	1	Standard	10.00
1	2	2	2	Express	15.00
2	3	2	2	Express	15.00
2	3	2	2	Express	15.00
1	2	2	2	Express	15.00
2	3	1	1	Standard	10.00
3	2	2	2	Express	15.00
3	2	1	1	Standard	10.00
2	3	2	2	Express	15.00

35. Retrieve a list of all employees and their corresponding payments, including cases where there are no matches on either side

```sql
SELECT e.*, p.*FROM Employees e FULL OUTER JOIN Payments p ON e.EmployeeID = p.EmployeeID;
```

	EmployeeID	Name	Email	ContactNumber	Role	Salary	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID
1	1	Manager1	manager1@email.com	1112223333	Manager	50000.00	NULL	NULL	NULL	NULL	NULL	NULL
2	2	DeliveryPerson1	delivery1@email.com	4445556666	Delivery Person	30000.00	1	1	1	10000.00	2024-03-03	2
3	2	DeliveryPerson1	delivery1@email.com	4445556666	Delivery Person	30000.00	2	2	2	1500.00	2024-03-04	2
4	2	DeliveryPerson1	delivery1@email.com	4445556666	Delivery Person	30000.00	4	4	1	1500.00	2024-03-06	2
5	2	DeliveryPerson1	delivery1@email.com	4445556666	Delivery Person	30000.00	6	6	3	25000.00	2024-03-08	2
6	2	DeliveryPerson1	delivery1@email.com	4445556666	Delivery Person	30000.00	8	8	2	1200.50	2024-03-10	2
7	2	DeliveryPerson1	delivery1@email.com	4445556666	Delivery Person	30000.00	9	9	2	1200.50	2024-03-10	2
8	3	DeliveryPerson2	delivery2@email.com	5556667777	Delivery Person	30000.00	3	3	3	1200.50	2024-03-05	3
9	3	DeliveryPerson2	delivery2@email.com	5556667777	Delivery Person	30000.00	5	5	2	1000.00	2024-03-07	3
10	3	DeliveryPerson2	delivery2@email.com	5556667777	Delivery Person	30000.00	7	7	1	2000.00	2024-03-09	3
11	4	John cena	john.cena@email.com	1234567890	Manager	60000.00	NULL	NULL	NULL	NULL	NULL	NULL
12	5	Authur Johnson	authur.johnson@email.com	9876543210	Clerk	45000.00	NULL	NULL	NULL	NULL	NULL	NULL

36. List all users and all courier services, showing all possible combinations.

```
SELECT u.*, cs.*FROM Users u CROSS JOIN CourierServices cs;
```

	UserID	Name	Email	Password	ContactNumber	Address	ServiceID	ServiceName	Cost
1	1	Rajesh Kumar	rajesh.kumar@email.com	password123	9876543210	12 Gandhi Nagar, Chennai	1	Standard	10.00
2	2	Priya Sharma	priya.sharma@email.com	password456	8765432109	34 Kaveri Street, Bangalore	1	Standard	10.00
3	3	Amit Patel	amit.patel@email.com	password789	7654321098	56 Krishna Lane, Hyderabad	1	Standard	10.00
4	4	Ananya Singh	ananya.singh@email.com	passwordabc	6543210987	78 Vindhya Nagar, Coimbatore	1	Standard	10.00
5	5	Sara Khan	sara.khan@email.com	passwordxyz	5432109876	90 Yamuna Road, Delhi	1	Standard	10.00
6	6	Michael Johnson	michael.johnson@email.com	password123	6547893210	23 Park Street, Mumbai	1	Standard	10.00
7	7	Emma Wilson	emma.wilson@email.com	password456	7896543210	45 Lake Avenue, Kolkata	1	Standard	10.00
8	8	David Lee	david.lee@email.com	password789	9876543210	67 Forest Lane, Pune	1	Standard	10.00
9	9	Sophia Garcia	sophia.garcia@email.com	passwordabc	8765432109	89 Ocean View, Goa	1	Standard	10.00
10	1	Rajesh Kumar	rajesh.kumar@email.com	password123	9876543210	12 Gandhi Nagar, Chennai	2	Express	15.00
11	2	Priya Sharma	priya.sharma@email.com	password456	8765432109	34 Kaveri Street, Bangalore	2	Express	15.00
12	3	Amit Patel	amit.patel@email.com	password789	7654321098	56 Krishna Lane, Hyderabad	2	Express	15.00
13	4	Ananya Singh	ananya.singh@email.com	passwordabc	6543210987	78 Vindhya Nagar, Coimbatore	2	Express	15.00
14	5	Sara Khan	sara.khan@email.com	passwordxyz	5432109876	90 Yamuna Road, Delhi	2	Express	15.00
15	6	Michael Johnson	michael.johnson@email.com	password123	6547893210	23 Park Street, Mumbai	2	Express	15.00
16	7	Emma Wilson	emma.wilson@email.com	password456	7896543210	45 Lake Avenue, Kolkata	2	Express	15.00
17	8	David Lee	david.lee@email.com	password789	9876543210	67 Forest Lane, Pune	2	Express	15.00
18	9	Sophia Garcia	sophia.garcia@email.com	passwordabc	8765432109	89 Ocean View, Goa	2	Express	15.00

37. List all employees and all locations, showing all possible combinations:

```
SELECT e.*, l.* FROM Employees e CROSS JOIN Locations l;
```

	EmployeeID	Name	Email	ContactNumber	Role	Salary	LocationID	LocationName	Address
1	1	Manager1	manager1@email.com	1112223333	Manager	50000.00	1	Warehouse1	789 Storage St, Chennai
2	2	DeliveryPerson1	delivery1@email.com	4445556666	Delivery Person	30000.00	1	Warehouse1	789 Storage St, Chennai
3	3	DeliveryPerson2	delivery2@email.com	5556667777	Delivery Person	30000.00	1	Warehouse1	789 Storage St, Chennai
4	4	John cena	john.cena@email.com	1234567890	Manager	60000.00	1	Warehouse1	789 Storage St, Chennai
5	5	Authur Johnson	authur.johnson@email.com	9876543210	Clerk	45000.00	1	Warehouse1	789 Storage St, Chennai
6	1	Manager1	manager1@email.com	1112223333	Manager	50000.00	2	Warehouse2	456 Logistics Ave, Bangalore
7	2	DeliveryPerson1	delivery1@email.com	4445556666	Delivery Person	30000.00	2	Warehouse2	456 Logistics Ave, Bangalore
8	3	DeliveryPerson2	delivery2@email.com	5556667777	Delivery Person	30000.00	2	Warehouse2	456 Logistics Ave, Bangalore
9	4	John cena	john.cena@email.com	1234567890	Manager	60000.00	2	Warehouse2	456 Logistics Ave, Bangalore
10	5	Authur Johnson	authur.johnson@email.com	9876543210	Clerk	45000.00	2	Warehouse2	456 Logistics Ave, Bangalore
11	1	Manager1	manager1@email.com	1112223333	Manager	50000.00	3	Warehouse3	123 Distribution Rd, Hyderabad
12	2	DeliveryPerson1	delivery1@email.com	4445556666	Delivery Person	30000.00	3	Warehouse3	123 Distribution Rd, Hyderabad
13	3	DeliveryPerson2	delivery2@email.com	5556667777	Delivery Person	30000.00	3	Warehouse3	123 Distribution Rd, Hyderabad
14	4	John cena	john.cena@email.com	1234567890	Manager	60000.00	3	Warehouse3	123 Distribution Rd, Hyderabad
15	5	Authur Johnson	authur.johnson@email.com	9876543210	Clerk	45000.00	3	Warehouse3	123 Distribution Rd, Hyderabad

38. Retrieve a list of couriers and their corresponding sender information (if available)

```
SELECT c.*, u.*FROM Couriers c LEFT JOIN Users u ON c.SenderName = u.Name;
```

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID
1	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2024-03-01	1
2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2024-03-02	1
3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00	In Transit	TN345678	2024-03-03	2
4	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679	2024-03-05	2
5	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00	In Transit	TN234567	2024-03-04	1
6	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50	Delivered	TN345688	2024-03-05	2
7	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789	2024-03-06	3
8	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2024-03-07	3
9	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08	2

EmployeeID	ServiceID	UserID	Name	Email	Password	ContactNumber	Address
2	1	1	Rajesh Kumar	rajesh.kumar@email.com	password123	9876543210	12 Gandhi Nagar, Chennai
2	2	2	Priya Sharma	priya.sharma@email.com	password456	8765432109	34 Kaveri Street, Bangalore
3	2	3	Amit Patel	amit.patel@email.com	password789	7654321098	56 Krishna Lane, Hyderabad
2	2	4	Ananya Singh	ananya.singh@email.com	passwordabc	6543210987	78 Vindhya Nagar, Coimbatore
2	2	5	Sara Khan	sara.khan@email.com	passwordxyz	5432109876	90 Yamuna Road, Delhi
3	1	6	Michael Johnson	michael.johnson@email.com	password123	6547893210	23 Park Street, Mumbai
2	2	7	Emma Wilson	emma.wilson@email.com	password456	7896543210	45 Lake Avenue, Kolkata
2	1	8	David Lee	david.lee@email.com	password789	9876543210	67 Forest Lane, Pune
3	2	1	Rajesh Kumar	rajesh.kumar@email.com	password123	9876543210	12 Gandhi Nagar, Chennai

39. Retrieve a list of couriers and their corresponding receiver information (if available):

```sql
SELECT c.*, u.*FROM Couriers c LEFT JOIN Users u ON c.ReceiverName = u.Name;
```

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID
1	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2024-03-01	1
2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2024-03-02	1
3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00	In Transit	TN345678	2024-03-03	2
4	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679	2024-03-05	2
5	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00	In Transit	TN234567	2024-03-04	1
6	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50	Delivered	TN345688	2024-03-05	2
7	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789	2024-03-06	3
8	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2024-03-07	3
9	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08	2

EmployeeID	ServiceID	UserID	Name	Email	Password	ContactNumber	Address
2	1	4	Ananya Singh	ananya.singh@email.com	passwordabc	6543210987	78 Vindhya Nagar, Coimbatore
2	2	3	Amit Patel	amit.patel@email.com	password789	7654321098	56 Krishna Lane, Hyderabad
3	2	2	Priya Sharma	priya.sharma@email.com	password456	8765432109	34 Kaveri Street, Bangalore
3	2	1	Rajesh Kumar	rajesh.kumar@email.com	password123	9876543210	12 Gandhi Nagar, Chennai
2	2	8	David Lee	david.lee@email.com	password789	9876543210	67 Forest Lane, Pune
3	1	7	Emma Wilson	emma.wilson@email.com	password456	7896543210	45 Lake Avenue, Kolkata
2	2	6	Michael Johnson	michael.johnson@email.com	password123	6547893210	23 Park Street, Mumbai
2	1	5	Sara Khan	sara.khan@email.com	passwordxyz	5432109876	90 Yamuna Road, Delhi
3	2	5	Sara Khan	sara.khan@email.com	passwordxyz	5432109876	90 Yamuna Road, Delhi

40. Retrieve a list of couriers along with the courier service details (if available):

```sql
SELECT c.*, cs.*FROM Couriers c LEFT JOIN CourierServices cs ON c.ServiceID = cs.ServiceID;
```

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber
1	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456
2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012
3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00	In Transit	TN345678
4	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679
5	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00	In Transit	TN234567
6	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50	Delivered	TN345688
7	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789
8	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890
9	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901

DeliveryDate	LocationID	EmployeeID	ServiceID	ServiceID	ServiceName	Cost
2024-03-01	1	2	1	1	Standard	10.00
2024-03-02	1	2	2	2	Express	15.00
2024-03-03	2	3	2	2	Express	15.00
2024-03-05	2	3	2	2	Express	15.00
2024-03-04	1	2	2	2	Express	15.00
2024-03-05	2	3	1	1	Standard	10.00
2024-03-06	3	2	2	2	Express	15.00
2024-03-07	3	2	1	1	Standard	10.00
2024-03-08	2	3	2	2	Express	15.00

41. Retrieve a list of employees and the number of couriers assigned to each employee:

SELECT e.EmployeeID, e.Name, COUNT(c.CourierID) AS TotalCouriersAssigned FROM Employees e
LEFT JOIN Couriers c ON e.EmployeeID = c.EmployeeID GROUP BY e.EmployeeID, e.Name;

	EmployeeID	Name	TotalCouriersAssigned
1	1	Manager1	0
2	2	DeliveryPerson1	5
3	3	DeliveryPerson2	4
4	4	John cena	0
5	5	Authur Johnson	0

42. Retrieve a list of locations and the total payment amount received at each location:

SELECT l.LocationID, l.LocationName, SUM(p.Amount) AS TotalPaymentsReceived
FROM Locations l LEFT JOIN Payments p ON l.LocationID = p.LocationID GROUP BY l.LocationID,
l.LocationName;

	LocationID	LocationName	TotalPaymentsReceived
1	1	Warehouse1	13500.00
2	2	Warehouse2	4901.00
3	3	Warehouse3	26200.50

43. Retrieve all couriers sent by the same sender (based on SenderName).

SELECT c1.*FROM Couriers c1 JOIN Couriers c2 ON c1.SenderName = c2.SenderName WHERE c1.CourierID <>
c2.CourierID;;

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
1	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2024-03-01	1	2	1
2	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08	2	3	2

44. List all employees who share the same role.

SELECT e1.*FROM Employees e1 JOIN Employees e2 ON e1.Role = e2.Role WHERE e1.EmployeeID <>
e2.EmployeeID;

	EmployeeID	Name	Email	ContactNumber	Role	Salary
1	1	Manager1	manager1@email.com	1112223333	Manager	50000.00
2	2	DeliveryPerson1	delivery1@email.com	4445556666	Delivery Person	30000.00
3	3	DeliveryPerson2	delivery2@email.com	5556667777	Delivery Person	30000.00
4	4	John cena	john.cena@email.com	1234567890	Manager	60000.00

45. Retrieve all payments made for couriers sent from the same location.

```sql
SELECT Payments.*FROM Payments JOIN Couriers ON Payments.CourierID = Couriers.CourierID WHERE
SenderAddress LIKE '12 Gandhi Nagar, Chennai';
```

	PaymentID	CourierID	LocationID	Amount	PaymentDate	EmployeeID
1	1	1	1	10000.00	2024-03-03	2
2	9	9	2	1200.50	2024-03-10	2

46. Retrieve all couriers sent from the same location (based on SenderAddress).

```sql
SELECT *FROM Couriers WHERE  SenderAddress LIKE '12 Gandhi Nagar, Chennai';
```

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
1	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2024-03-01	1	2	1
2	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08	2	3	2

47. List employees and the number of couriers they have delivered:

```sql
SELECT e.EmployeeID, e.Name, COUNT(c.CourierID) AS TotalCouriersDelivered FROM Employees e
LEFT JOIN Couriers c ON e.EmployeeID = c.EmployeeID WHERE c.Status = 'Delivered' GROUP BY
e.EmployeeID, e.Name;
```

	EmployeeID	Name	TotalCouriersDelivered
1	2	DeliveryPerson1	1
2	3	DeliveryPerson2	2

48. Find couriers that were paid an amount greater than the cost of their respective courier services

```sql
SELECT c.*FROM Couriers c JOIN Payments p ON c.CourierID = p.CourierID
JOIN CourierServices cs ON c.ServiceID = cs.ServiceID WHERE p.Amount > cs.Cost;
```

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
1	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2024-03-01	1	2	1
2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2024-03-02	1	2	2
3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00	In Transit	TN345678	2024-03-03	2	3	2
4	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679	2024-03-05	2	3	2
5	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00	In Transit	TN234567	2024-03-04	1	2	2
6	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50	Delivered	TN345688	2024-03-05	2	3	1
7	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789	2024-03-06	3	2	2
8	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2024-03-07	3	2	1
9	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08	2	3	2

Scope: Inner Queries, Non Equi Joins, Equi joins,Exist,Any,All

49. Find couriers that have a weight greater than the average weight of all couriers

```sql
SELECT * FROM Couriers WHERE Weight > (SELECT AVG(Weight) FROM Couriers);
```

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
1	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00	In Transit	TN345678	2024-03-03	2	3	2
2	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679	2024-03-05	2	3	2
3	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789	2024-03-06	3	2	2
4	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2024-03-07	3	2	1

50. Find the names of all employees who have a salary greater than the average salary:

`SELECT Name FROM Employees WHERE Salary > (SELECT AVG(Salary) FROM Employees);`

	Name
1	Manager1
2	John cena
3	Authur Johnson

51. Find the total cost of all courier services where the cost is less than the maximum cost

`SELECT SUM(Amount) AS TotalCost FROM Payments WHERE Amount < (SELECT MAX(Amount) FROM Payments);`

	TotalCost
1	19601.50

52. Find all couriers that have been paid for

`SELECT c.*FROM Couriers c WHERE EXISTS (SELECT 1 FROM Payments p WHERE p.CourierID = c.CourierID);`

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
1	1	Rajesh Kumar	12 Gandhi Nagar, Chennai	Ananya Singh	78 Vindhya Nagar, Coimbatore	2.50	In Transit	TN123456	2024-03-01	1	2	1
2	2	Priya Sharma	34 Kaveri Street, Bangalore	Amit Patel	56 Krishna Lane, Hyderabad	1.80	Delivered	TN789012	2024-03-02	1	2	2
3	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00	In Transit	TN345678	2024-03-03	2	3	2
4	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679	2024-03-05	2	3	2
5	5	Sara Khan	90 Yamuna Road, Delhi	David Lee	67 Forest Lane, Pune	2.00	In Transit	TN234567	2024-03-04	1	2	2
6	6	Michael Johnson	23 Park Street, Mumbai	Emma Wilson	45 Lake Avenue, Kolkata	1.50	Delivered	TN345688	2024-03-05	2	3	1
7	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789	2024-03-06	3	2	2
8	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2024-03-07	3	2	1
9	9	Rajesh Kumar	12 Gandhi Nagar, Chennai	Sara Khan	90 Yamuna Road, Delhi	2.80	Delivered	TN678901	2024-03-08	2	3	2

53. Find the locations where the maximum payment amount was made

`SELECT l.*FROM Locations l WHERE LocationID = (SELECT TOP 1 LocationID FROM Payments ORDER BY Amount DESC);`

	LocationID	LocationName	Address
1	3	Warehouse3	123 Distribution Rd, Hyderabad

54. Find all couriers whose weight is greater than the weight of all couriers sent by a specific sender (e.g., 'SenderName'):

`SELECT c.*FROM Couriers c WHERE Weight > ALL (SELECT Weight FROM Couriers WHERE SenderName = 'Rajesh Kumar');`

	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	EmployeeID	ServiceID
1	3	Amit Patel	56 Krishna Lane, Hyderabad	Priya Sharma	34 Kaveri Street, Bangalore	3.00	In Transit	TN345678	2024-03-03	2	3	2
2	4	Ananya Singh	78 Vindhya Nagar, Coimbatore	Rajesh Kumar	12 Gandhi Nagar, Chennai	4.00	In Transit	TN345679	2024-03-05	2	3	2
3	7	Emma Wilson	45 Lake Avenue, Kolkata	Michael Johnson	23 Park Street, Mumbai	3.50	In Transit	TN456789	2024-03-06	3	2	2
4	8	David Lee	67 Forest Lane, Pune	Sara Khan	90 Yamuna Road, Delhi	4.50	In Transit	TN567890	2024-03-07	3	2	1