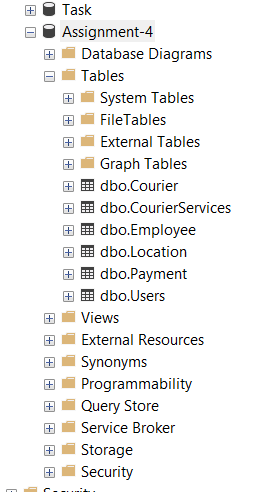
Task1 Database Design

• Define the Database Schema • Create SQL tables for entities such as User, Courier, Employee, Location, Payment.



• Populate Sample Data • Insert sample data into the tables to simulate real-world scenarioA screenshot of a computer

Description automatically generateds.

Task 2: Select, Where

Solve the following queries in the Schema that you have created above

1. List all customers:

A screenshot of a computer

Description automatically generated

2. List all orders for a specific customer:

A screenshot of a computer

Description automatically generated

3. List all couriers:

A screenshot of a computer

Description automatically generated

4. List all packages for a specific order:

A screenshot of a computer

Description automatically generated

5. List all deliveries for a specific courier:

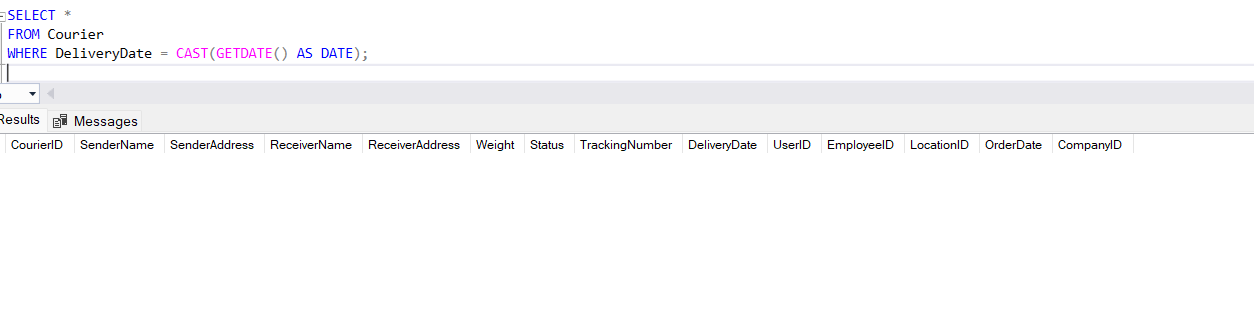
A screenshot of a computer

Description automatically generated

6. List all undelivered packages:

A screenshot of a computer

Description automatically generated

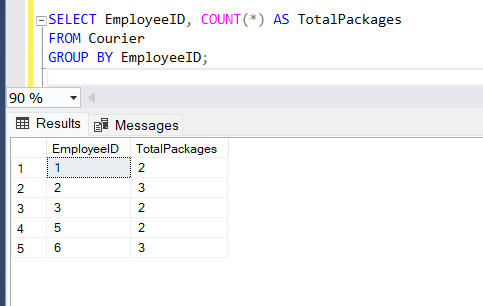
7. List all packages that are scheduled for delivery today:

8. List all packages with a specific status:

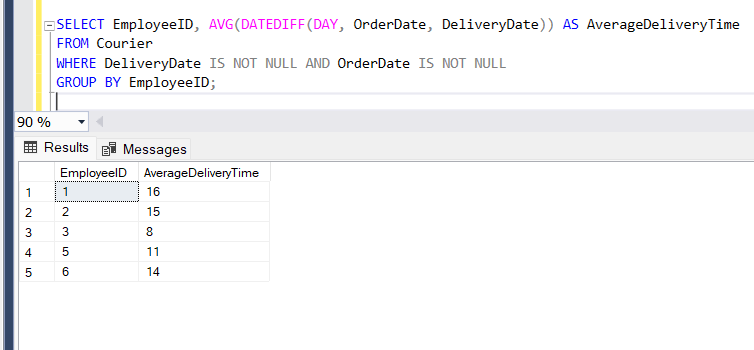
A screenshot of a computer

Description automatically generated

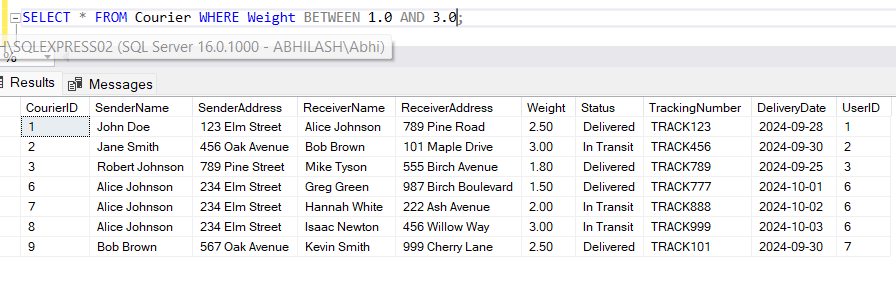
9. Calculate the total number of packages for each courier.



10. Find the average delivery time for each courier



11. List all packages with a specific weight range:



12. Retrieve employees whose names contain 'John'

A screenshot of a computer

Description automatically generated

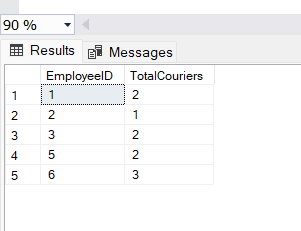
13. Retrieve all courier records with payments greater than $50.

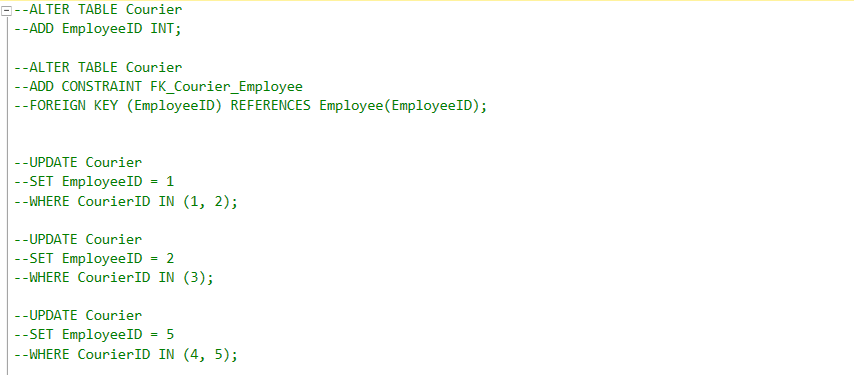
A screenshot of a computer screen

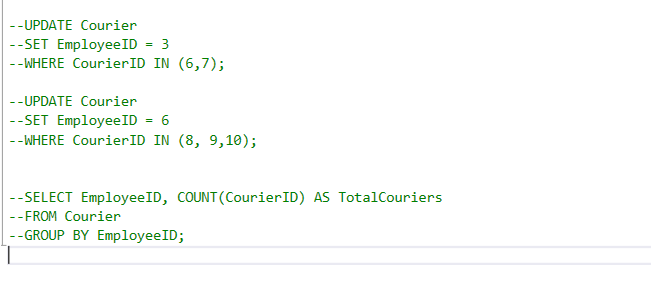
Description automatically generated

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

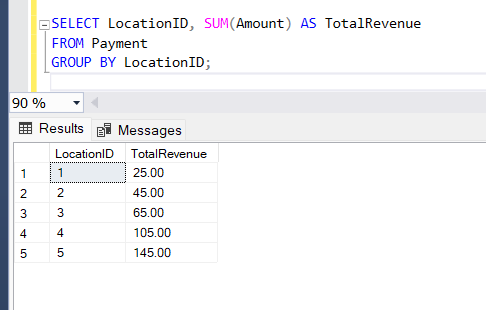
14. Find the total number of couriers handled by each employee.



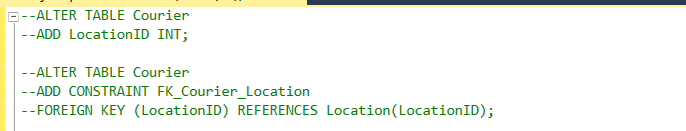


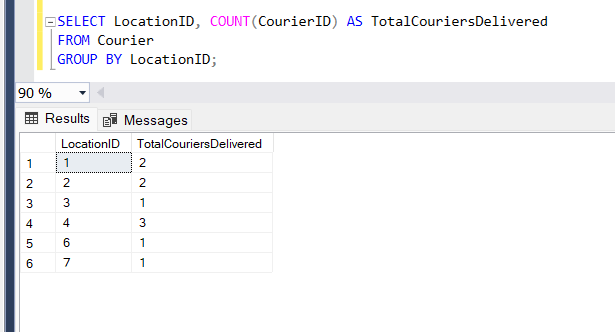
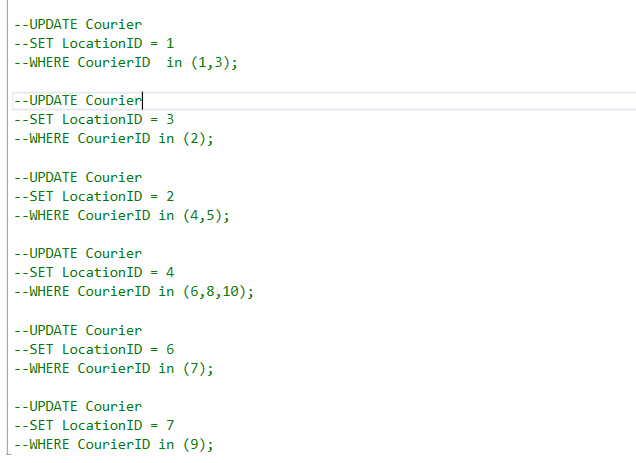


15. Calculate the total revenue generated by each location

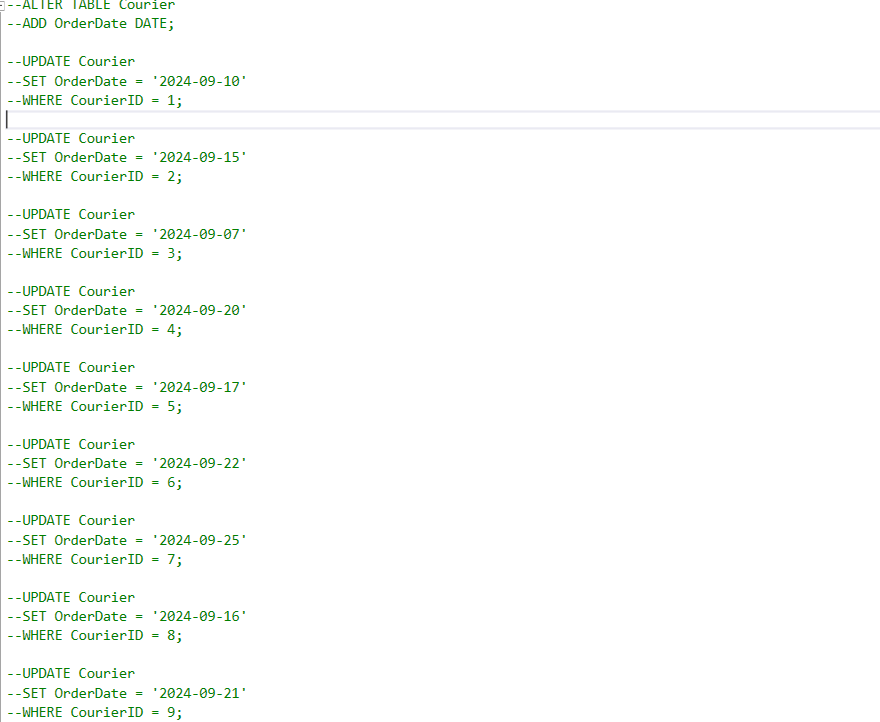


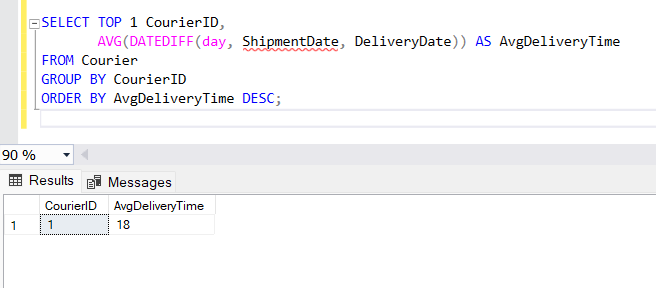
16. Find the total number of couriers delivered to each location.



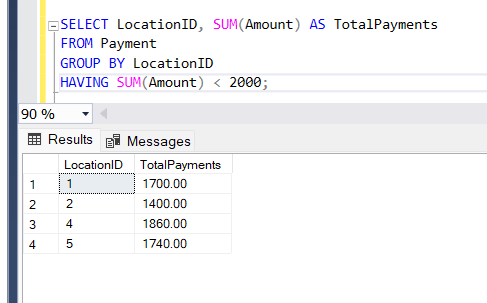


17. Find the courier with the highest average delivery time:

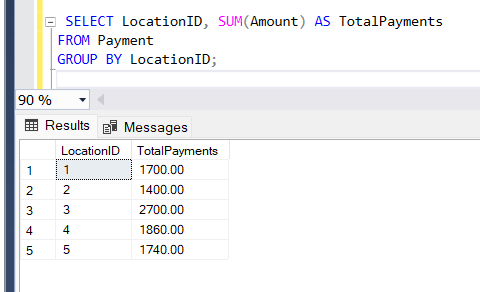




18. Find Locations with Total Payments Less Than a Certain Amount

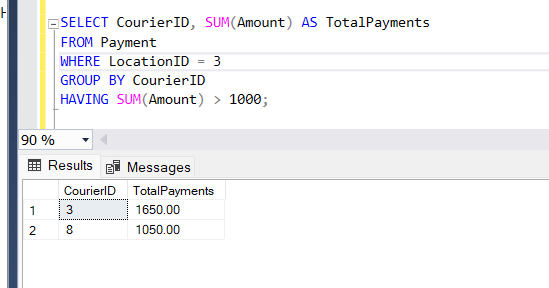


19. Calculate Total Payments per Location



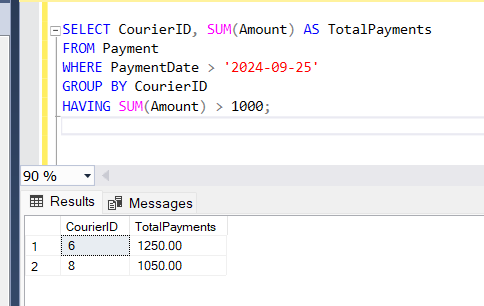
20. Retrieve couriers who have received payments totaling more than $1000 in a specific location

(LocationID = X):



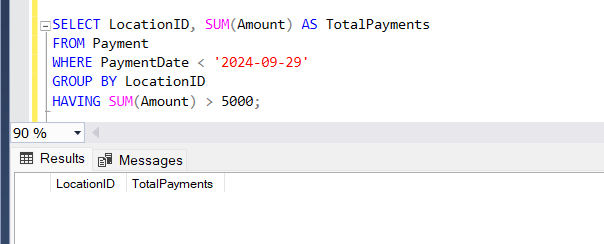
21. Retrieve couriers who have received payments totaling more than $1000 after a certain date

(PaymentDate > 'YYYY-MM-DD'):



22. Retrieve locations where the total amount received is more than $5000 before a certain date

(PaymentDate > 'YYYY-MM-DD')

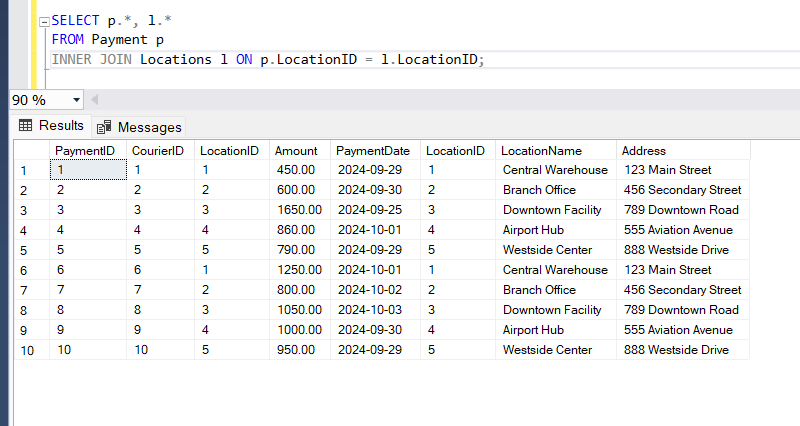


Task 4: Inner Join,Full Outer Join, Cross Join, Left Outer Join,Right Outer Join

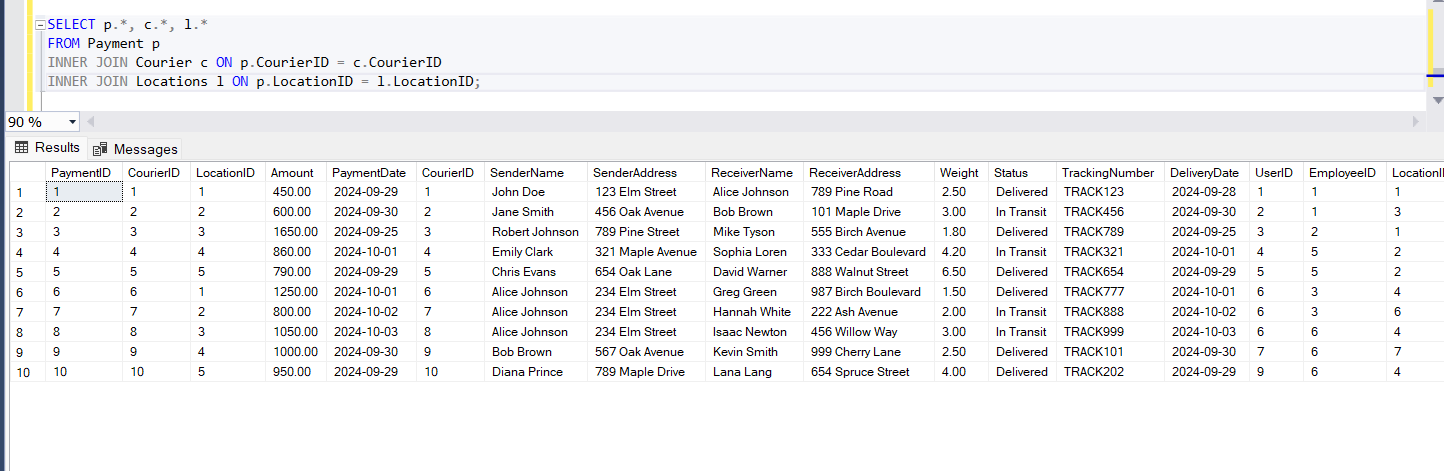
23. Retrieve Payments with Courier Information



24. Retrieve Payments with Location Information



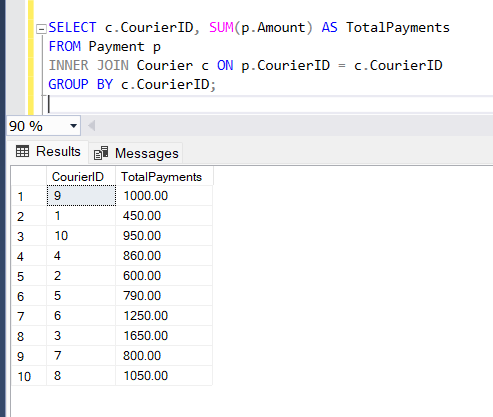
25. Retrieve Payments with Courier and Location Information



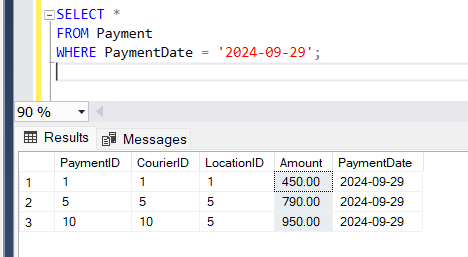
26. List all payments with courier details



27. Total payments received for each courier



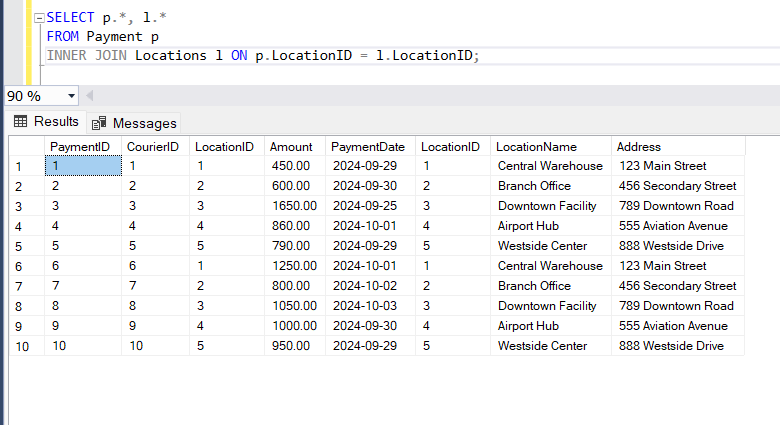
28. List payments made on a specific date



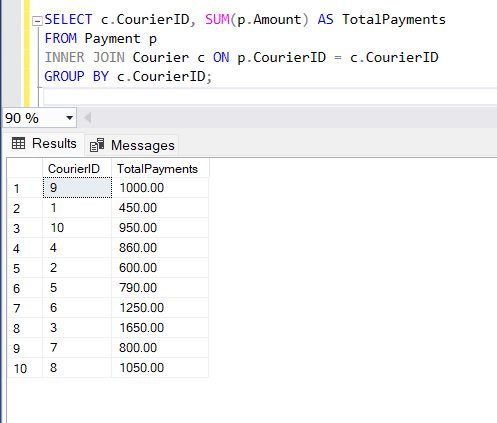
29. Get Courier Information for Each Payment



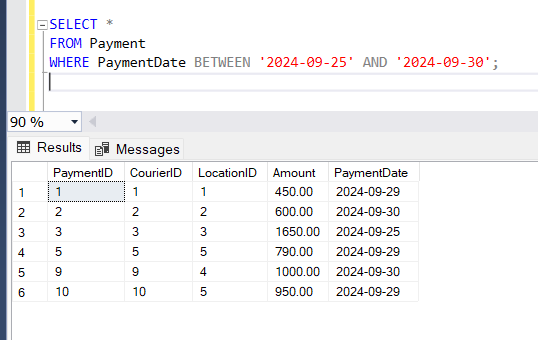
30. Get Payment Details with Location



31. Calculating Total Payments for Each Courier

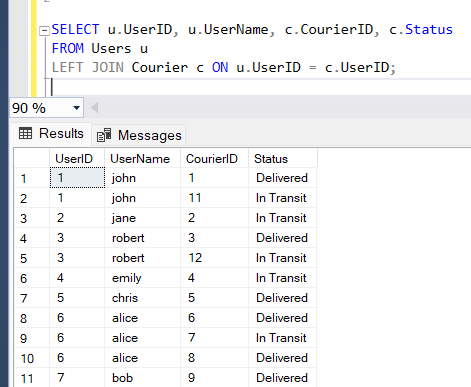


32. List Payments Within a Date Range



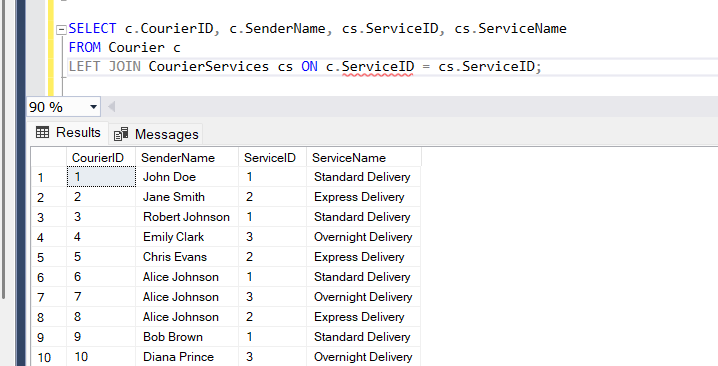
33. Retrieve a list of all users and their corresponding courier records, including cases where there are

no matches on either side



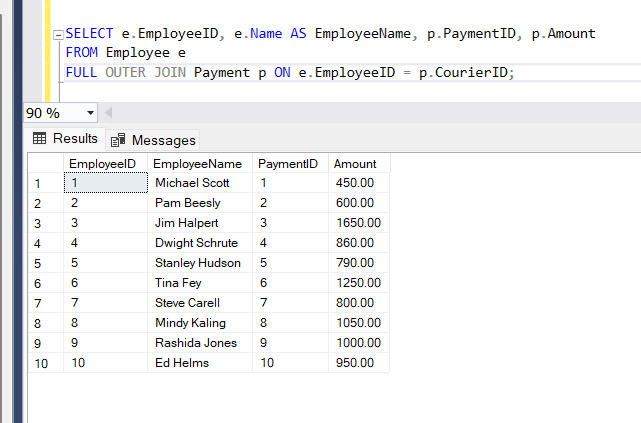
34. Retrieve a list of all couriers and their corresponding services, including cases where there are no

matches on either side

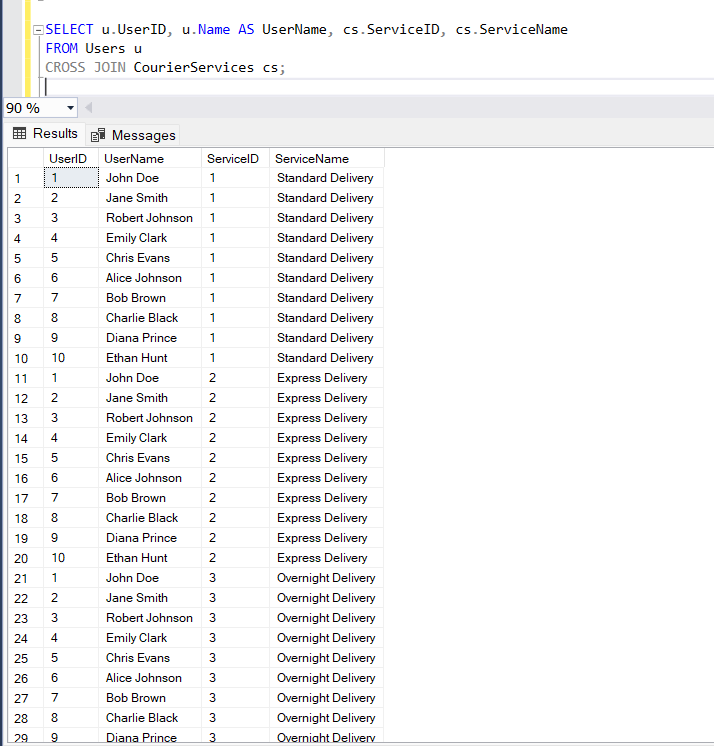


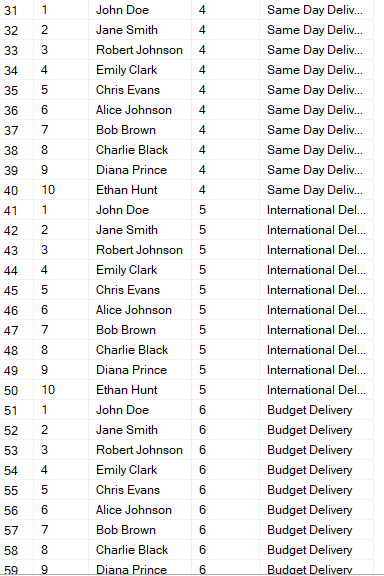
35. Retrieve a list of all employees and their corresponding payments, including cases where there are

no matches on either side

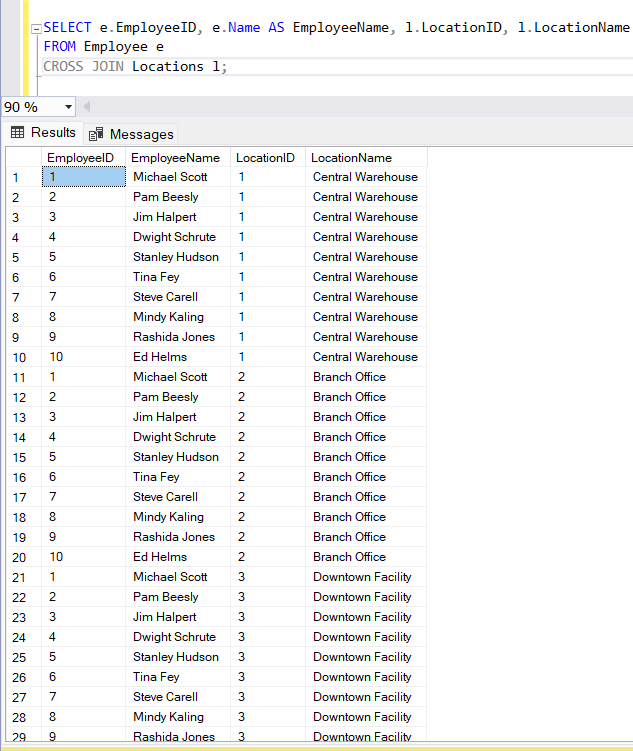


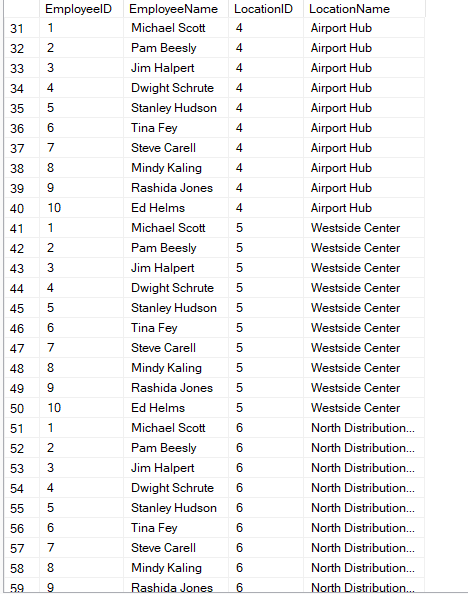
36. List all users and all courier services, showing all possible combinations.



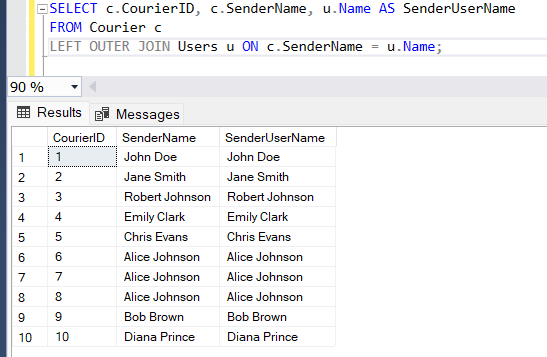


37. List all employees and all locations, showing all possible combinations:

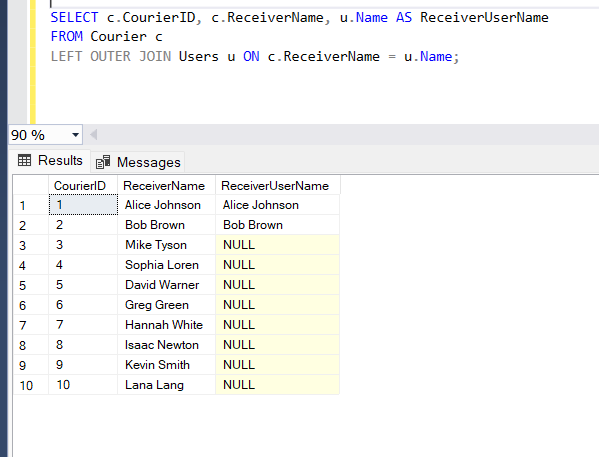




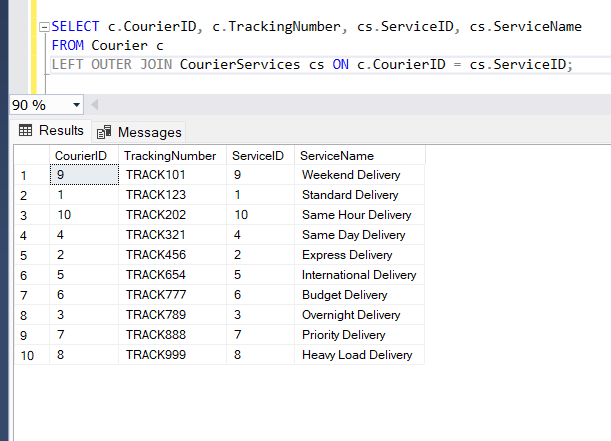
38. Retrieve a list of couriers and their corresponding sender information (if available)



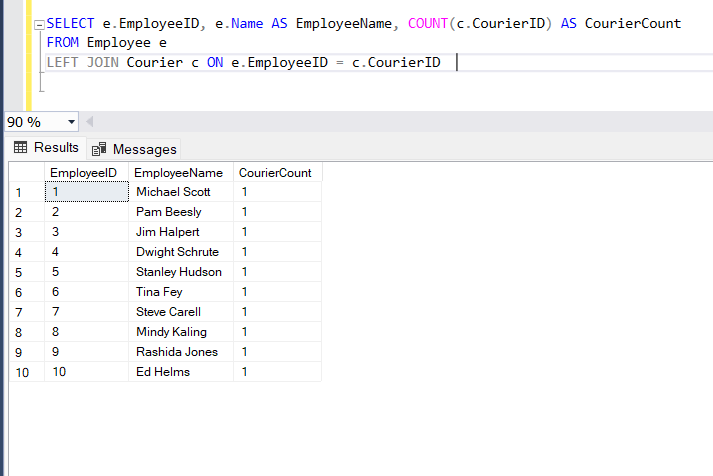
39. Retrieve a list of couriers and their corresponding receiver information (if available):



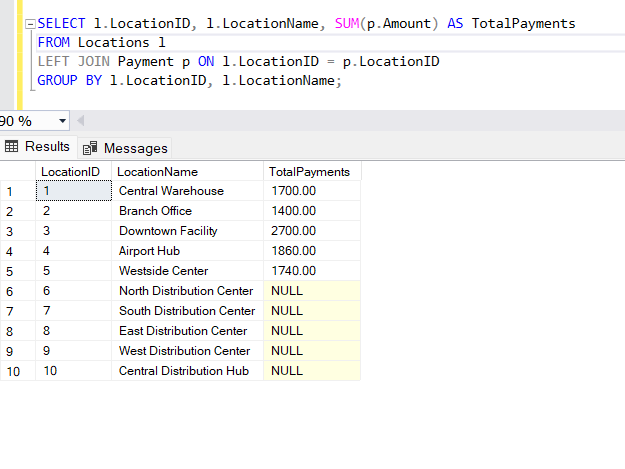
40. Retrieve a list of couriers along with the courier service details (if available):



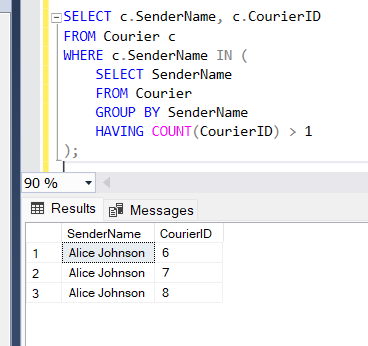
41. Retrieve a list of employees and the number of couriers assigned to each employee:



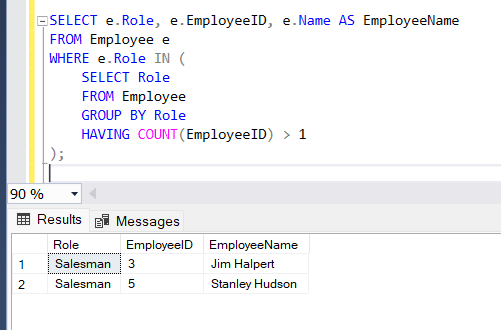
42. Retrieve a list of locations and the total payment amount received at each location:



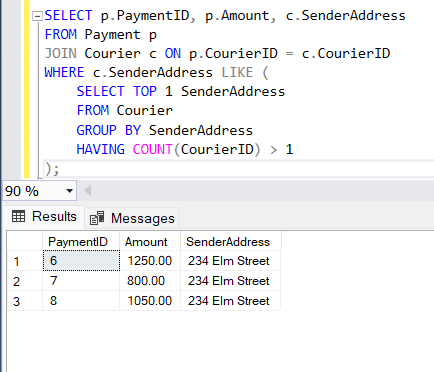
43. Retrieve all couriers sent by the same sender (based on SenderName).



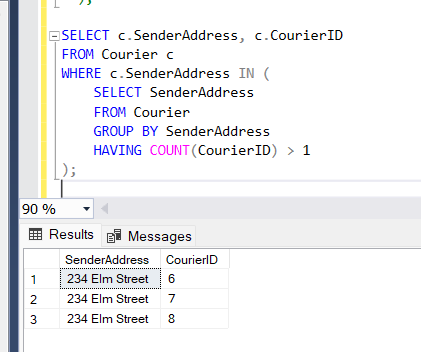
44. List all employees who share the same role.



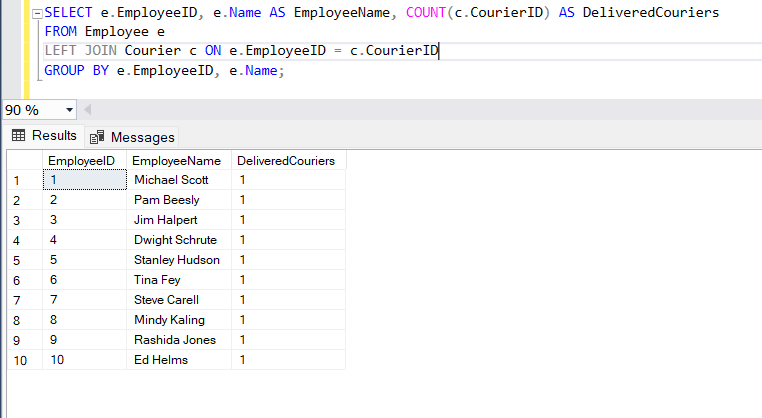
45. Retrieve all payments made for couriers sent from the same location.



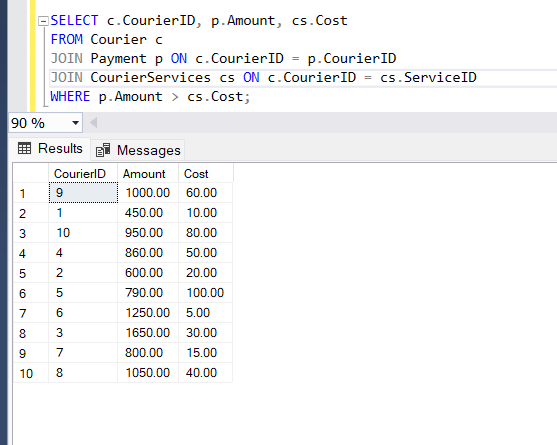
46. Retrieve all couriers sent from the same location (based on SenderAddress).



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

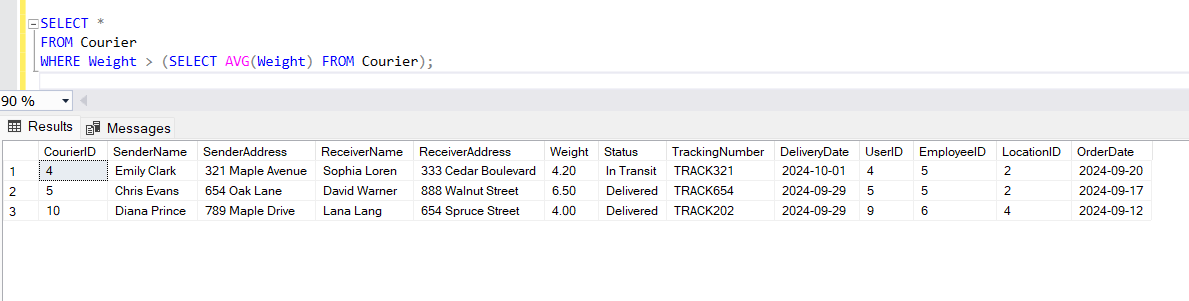


48. Find couriers that were paid an amount greater than the cost of their respective courier services

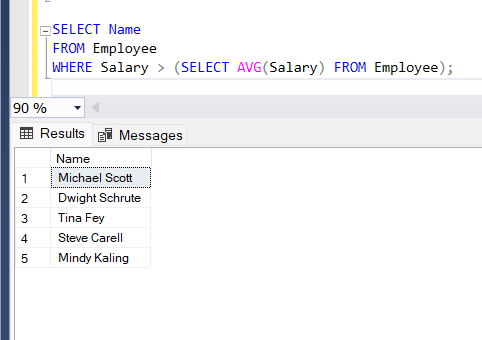


Scope: Inner Queries, Non Equi Joins, Equi joins,Exist,Any,All

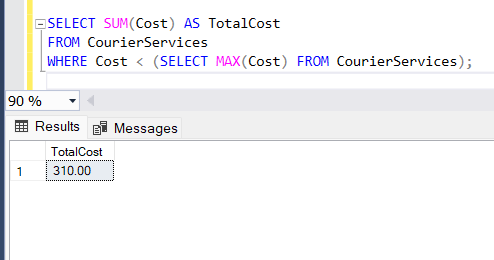
49. Find couriers that have a weight greater than the average weight of all couriers



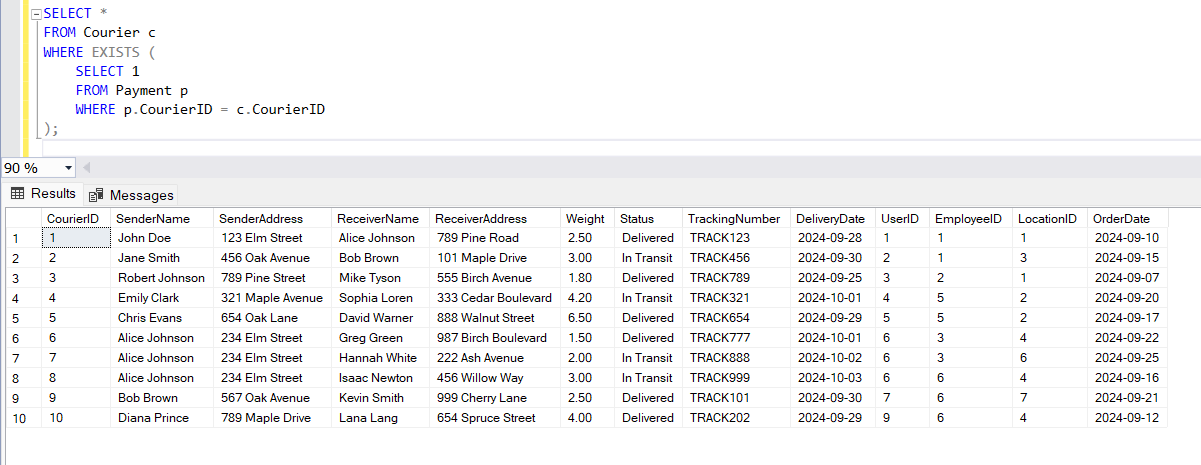
50. Find the names of all employees who have a salary greater than the average salary:



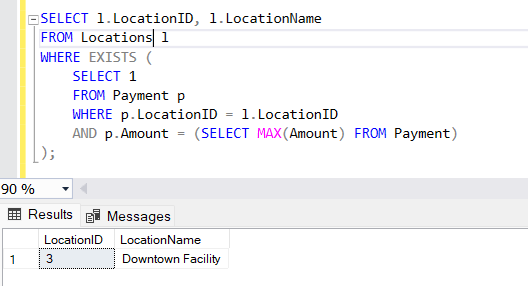
51. Find the total cost of all courier services where the cost is less than the maximum cost



52. Find all couriers that have been paid for

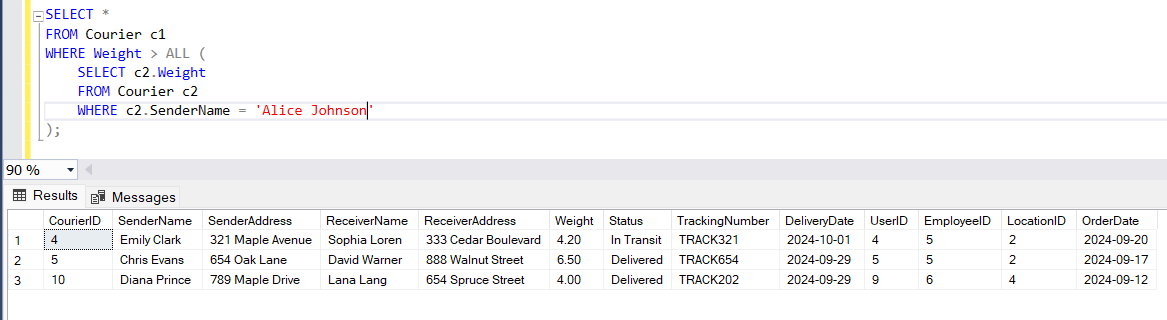


53. Find the locations where the maximum payment amount was made



54. Find all couriers whose weight is greater than the weight of all couriers sent by a specific sender

(e.g., 'SenderName'):

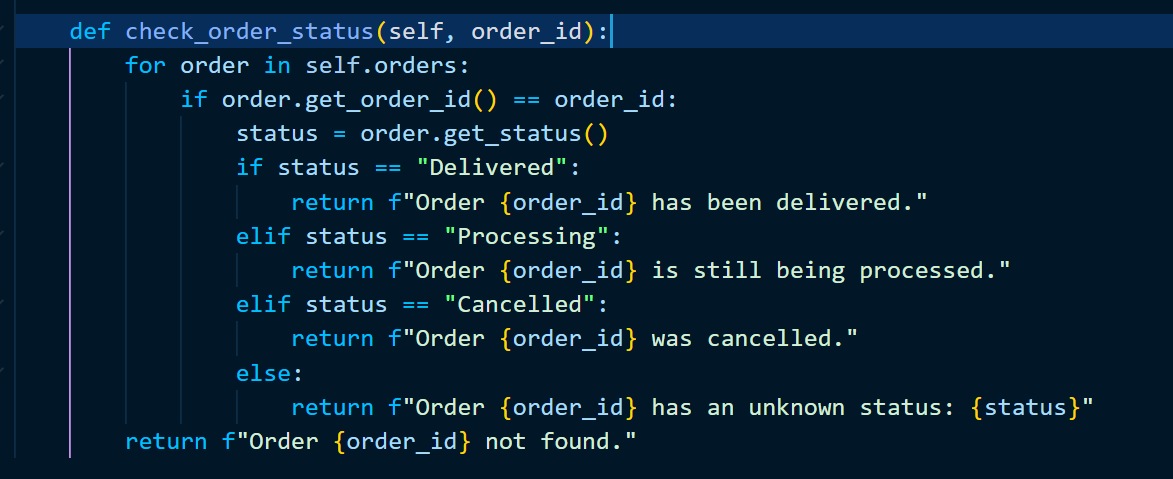


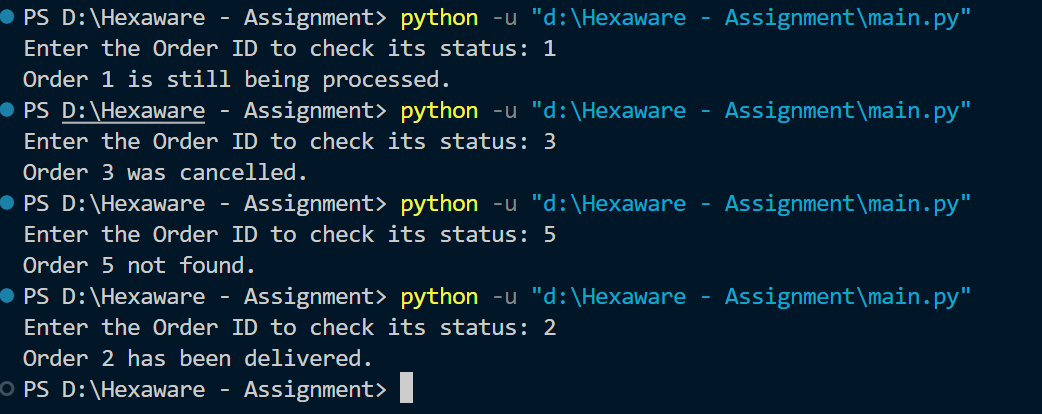
**Coding**

**Task 1: Control Flow Statements**

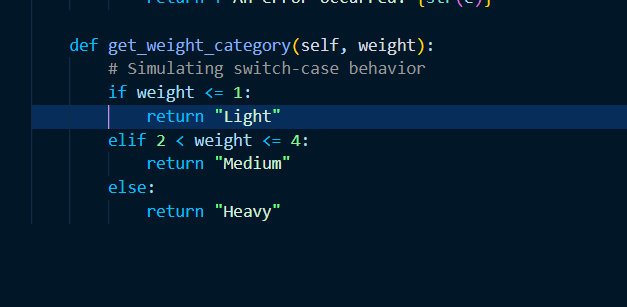
1. Write a program that checks whether a given order is delivered or not based on its status (e.g.,

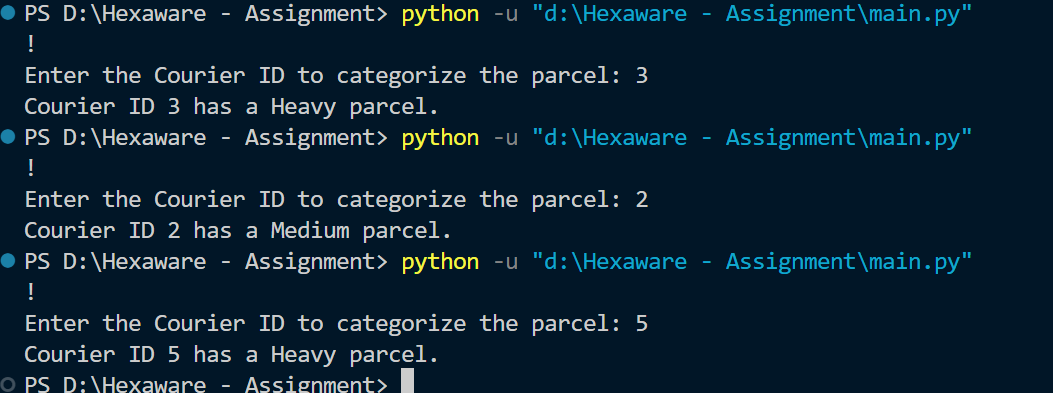
"Processing," "Delivered," "Cancelled"). Use if-else statements for this.

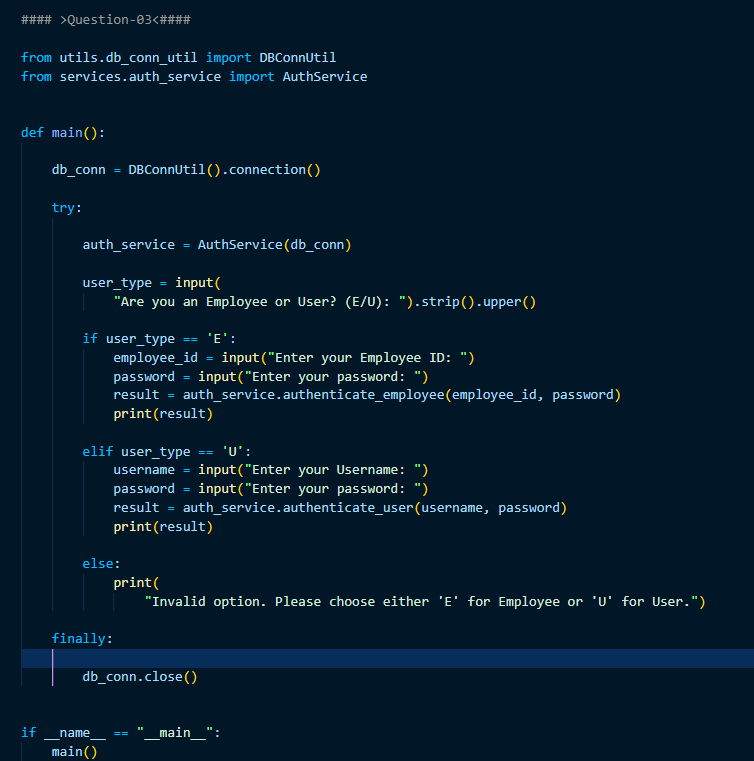


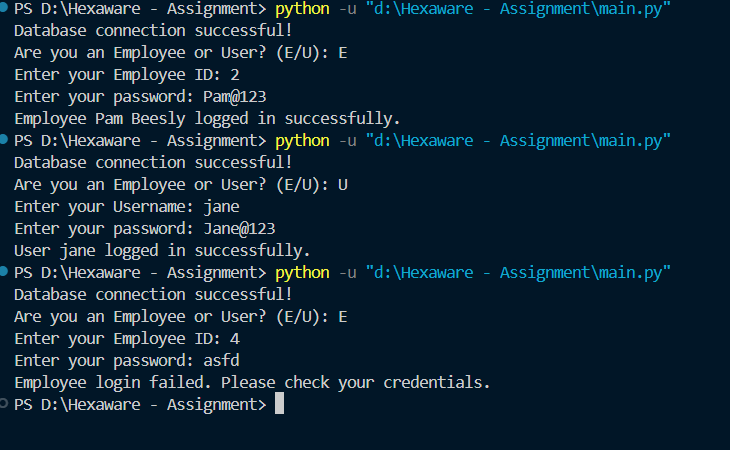


2. Implement a switch-case statement to categorize parcels based on their weight into "Light," "Medium," or "Heavy."



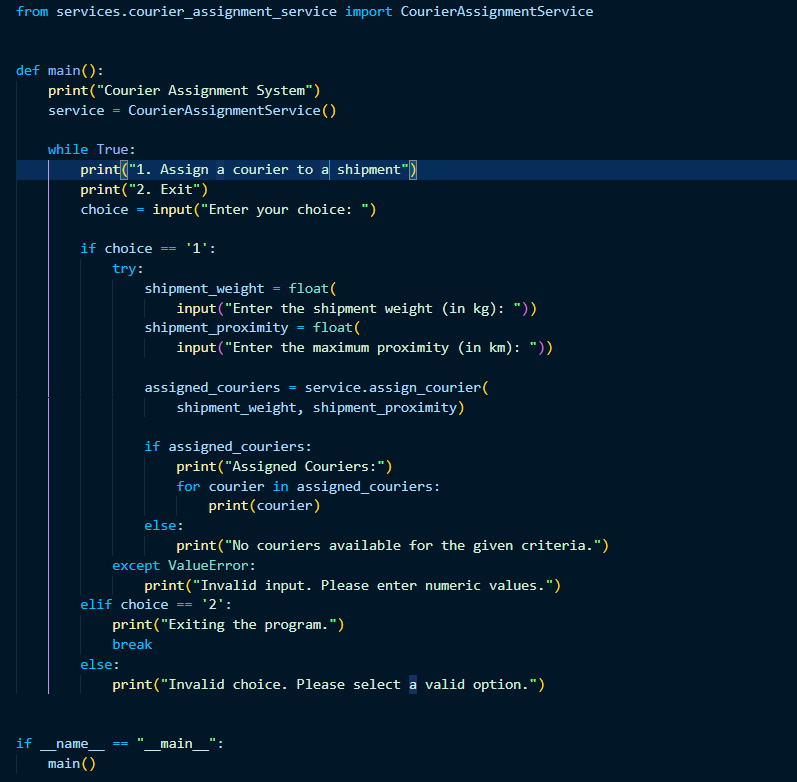


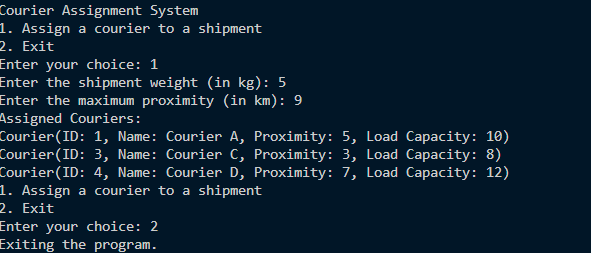
3. Implement User Authentication 1. Create a login system for employees and customers using Java control flow statements. 



4. Implement Courier Assignment Logic 1. Develop a mechanism to assign couriers to shipments based

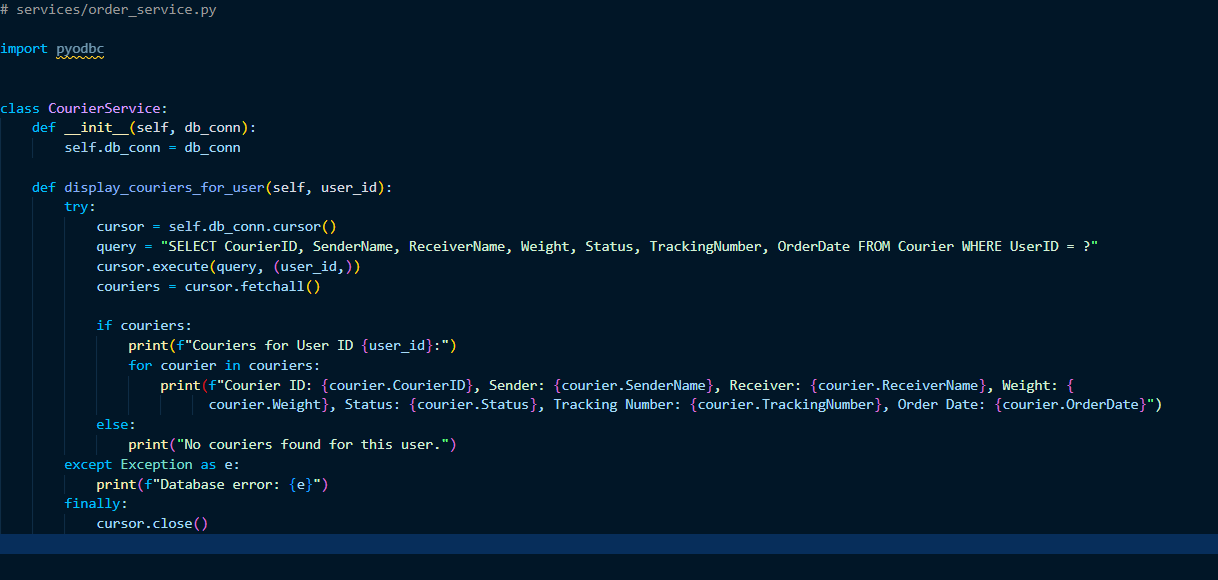
on predefined criteria (e.g., proximity, load capacity) using loops.

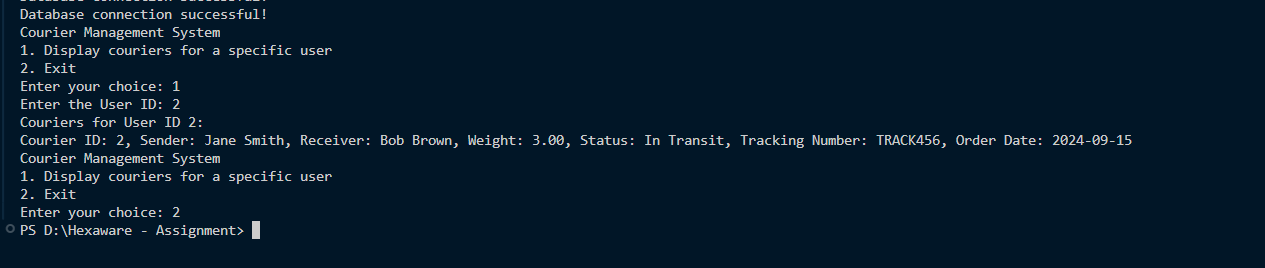




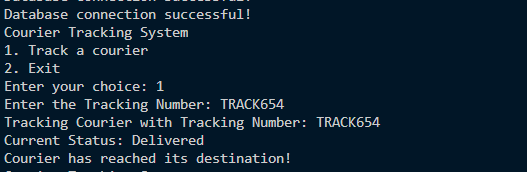
**Task 2: Loops and Iteration**

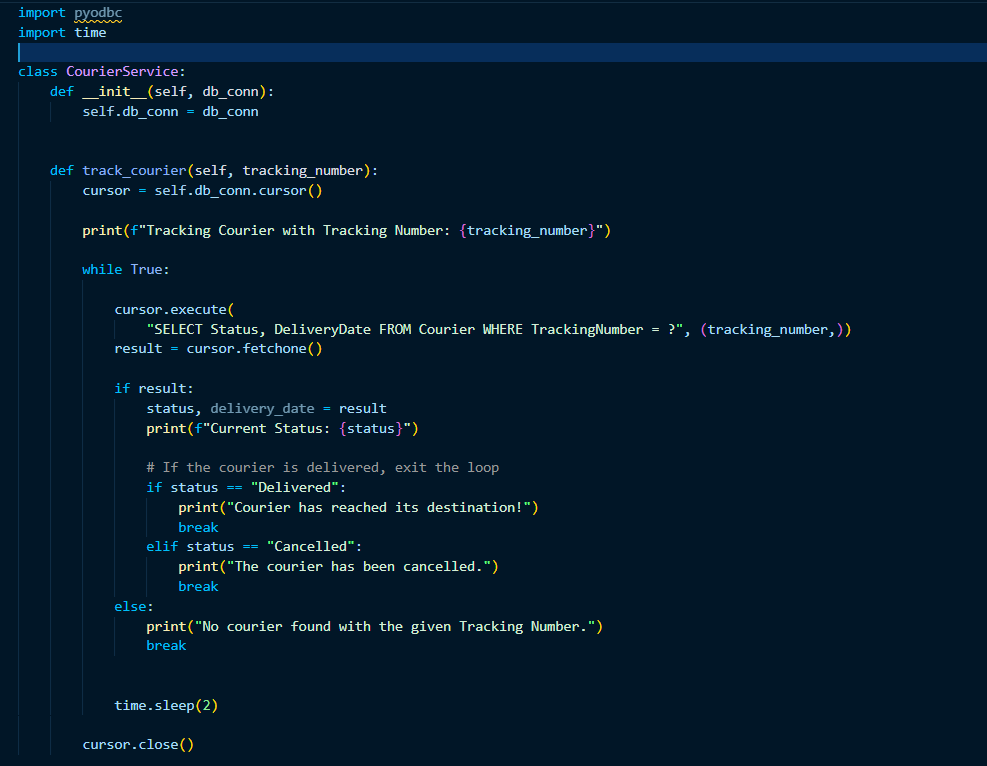
5. Write a Java program that uses a for loop to display all the orders for a specific customer.





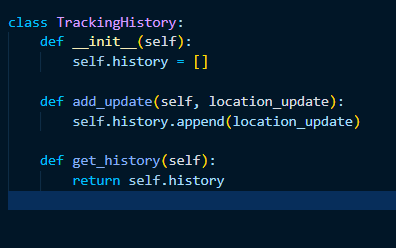
6. Implement a while loop to track the real-time location of a courier until it reaches its destination.

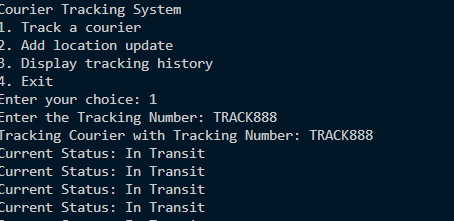


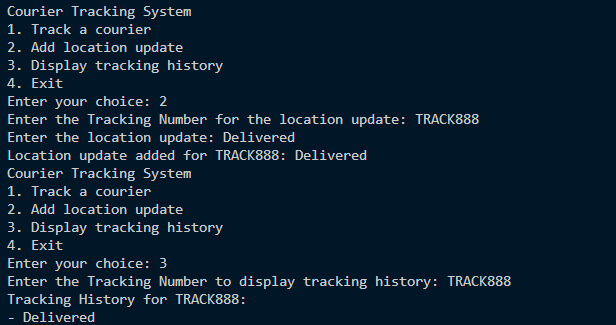


**Task 3: Arrays and Data Structures**

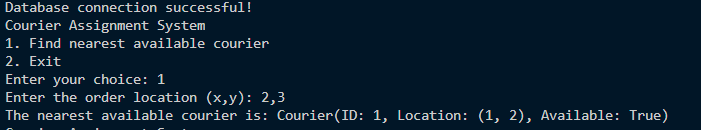
7. Create an array to store the tracking history of a parcel, where each entry represents a location update.

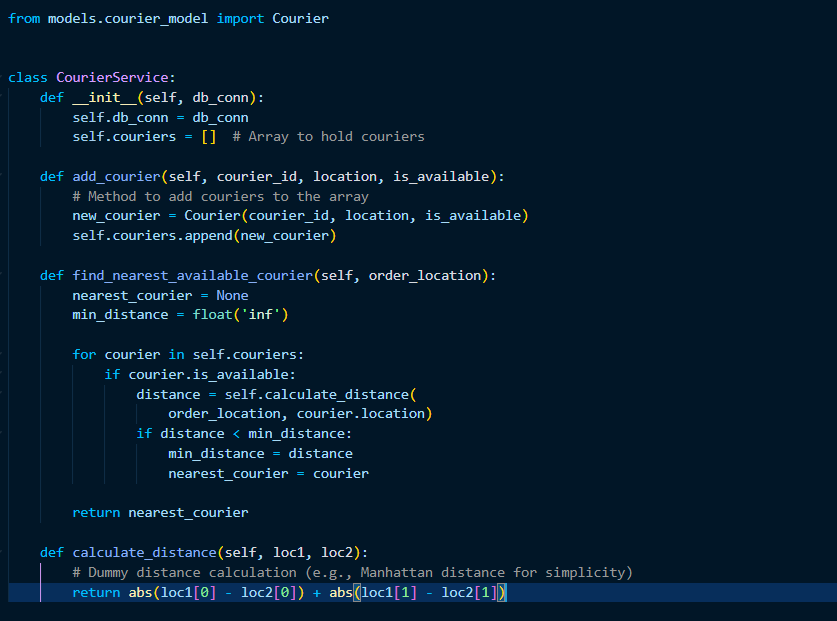






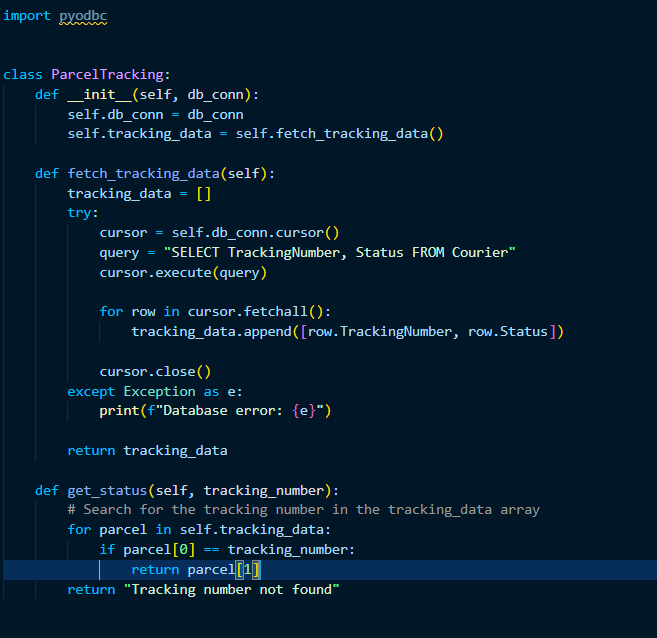
8. Implement a method to find the nearest available courier for a new order using an array of couriers.

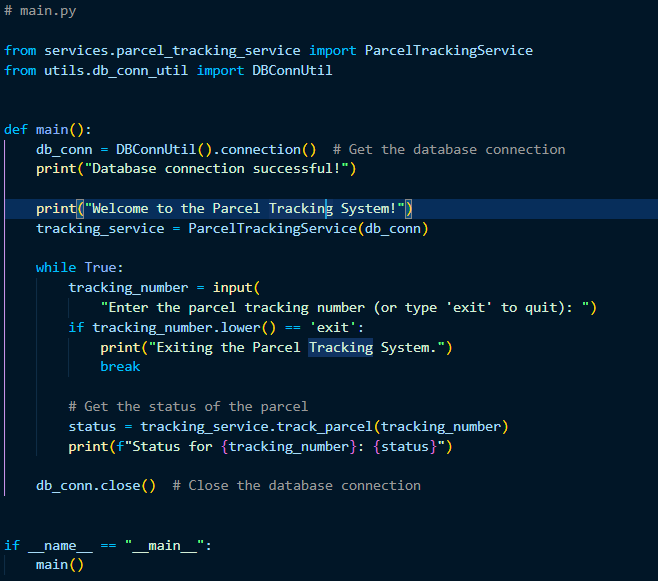


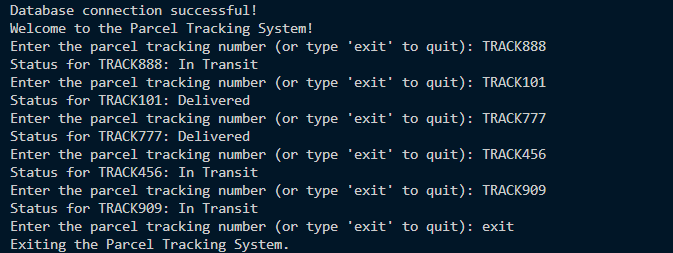


**Task 4: Strings,2d Arrays, user defined functions,Hashmap**

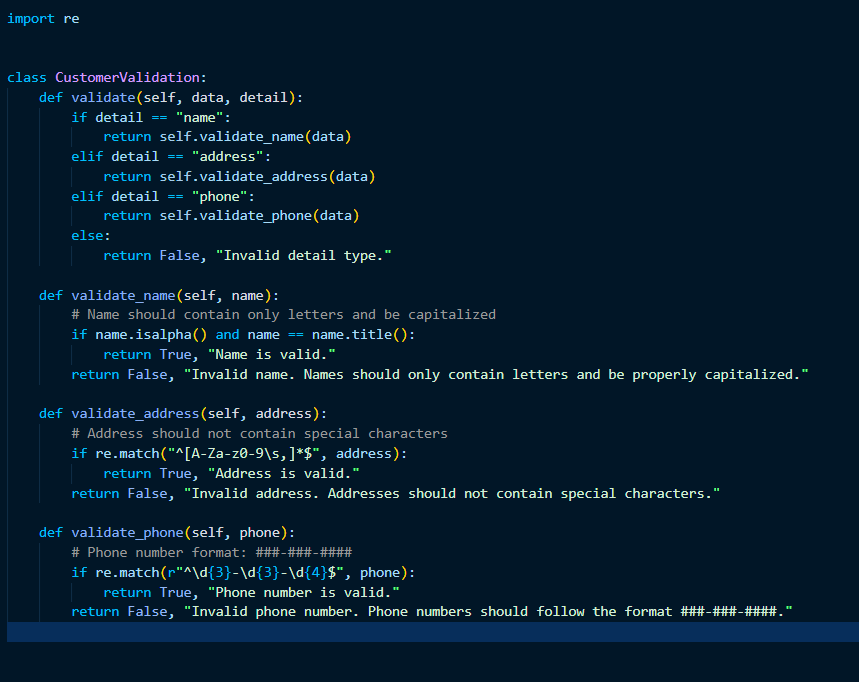
9. Parcel Tracking: Create a program that allows users to input a parcel tracking number.Store the tracking number and Status in 2d String Array. Initialize the array with values. Then, simulate the tracking process by displaying messages like "Parcel in transit," "Parcel out for delivery," or "Parcel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

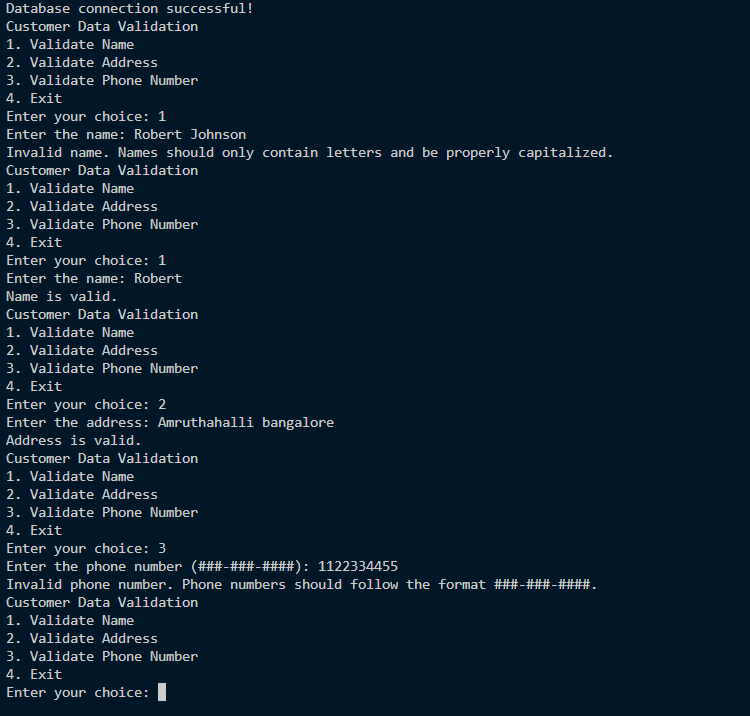






10. Customer Data Validation: Write a function which takes 2 parameters, data-denotes the data and detail-denotes if it is name addtress or phone number.Validate customer information based on following critirea. Ensure that names contain only letters and are properly capitalized, addresses do not contain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

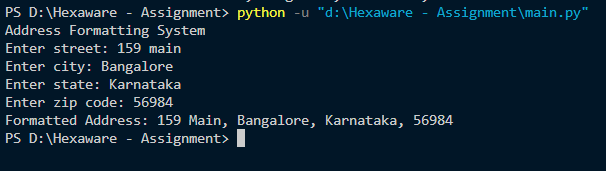


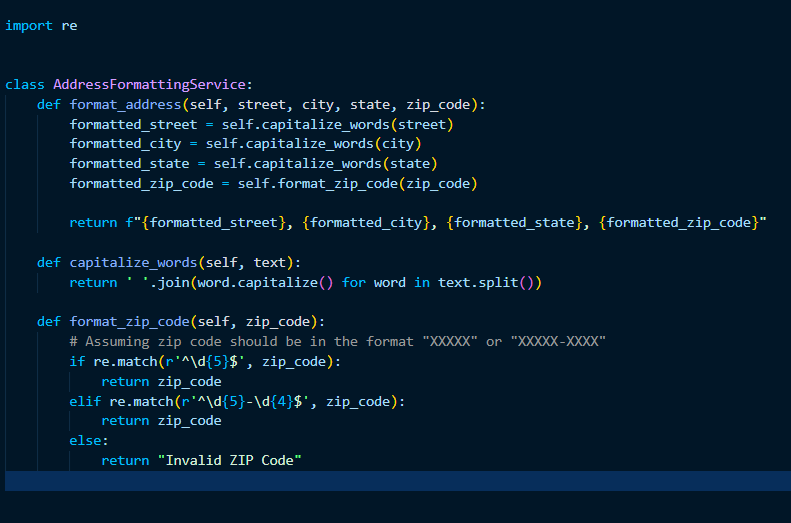


11. Address Formatting: Develop a function that takes an address as input (street, city, state, zip code)

and formats it correctly, including capitalizing the first letter of each word and properly formatting the

zip code.

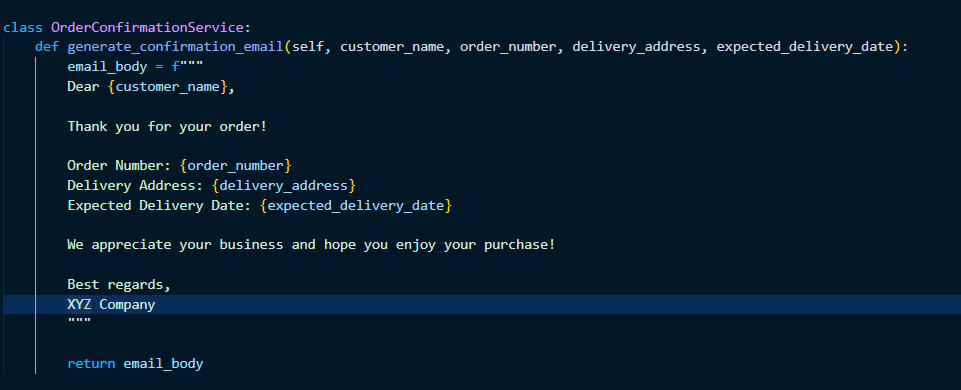


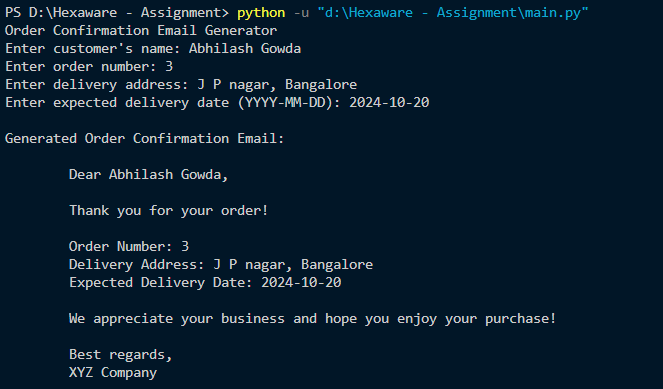


12. Order Confirmation Email: Create a program that generates an order confirmation email. The email

should include details such as the customer's name, order number, delivery address, and expected

delivery date.

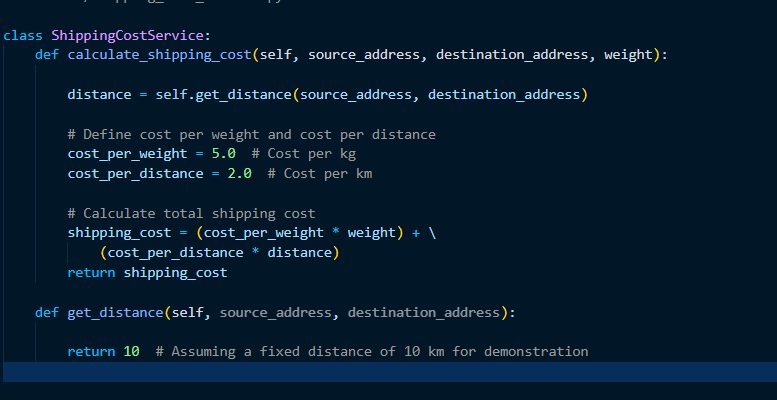


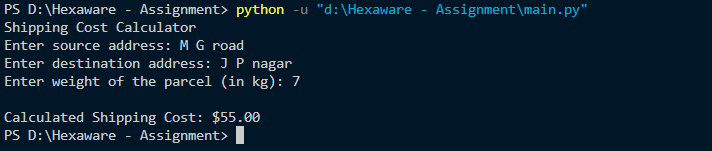


13. Calculate Shipping Costs: Develop a function that calculates the shipping cost based on the distance

between two locations and the weight of the parcel. You can use string inputs for the source and

destination addresses.

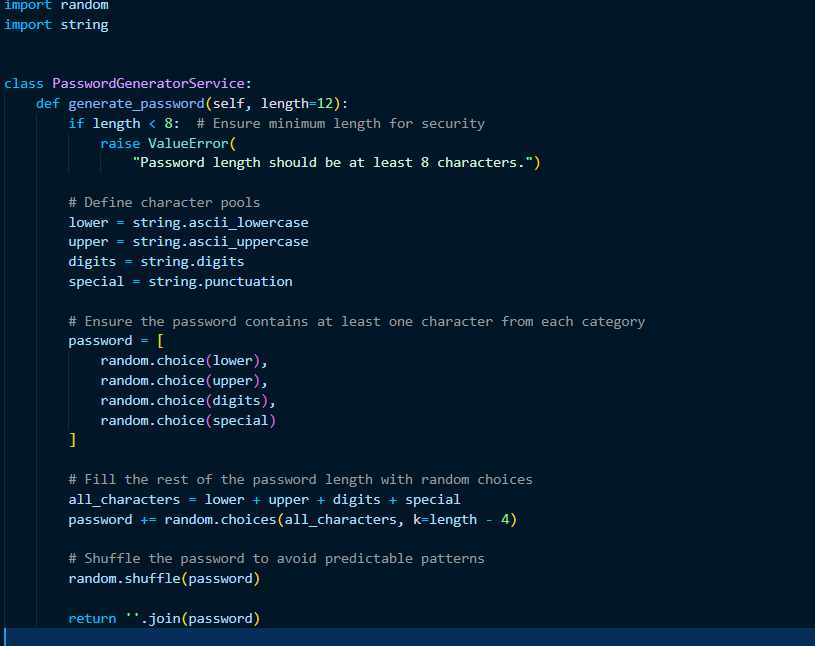


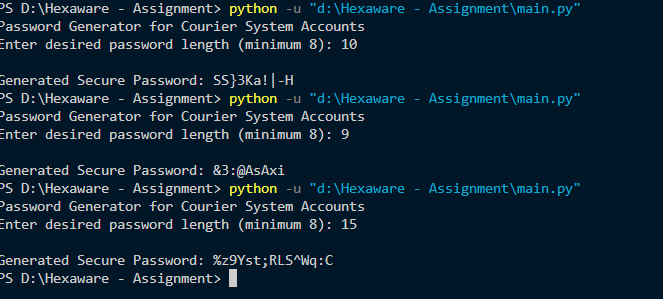


14. Password Generator: Create a function that generates secure passwords for courier system

accounts. Ensure the passwords contain a mix of uppercase letters, lowercase letters, numbers, and

special characters.





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.

