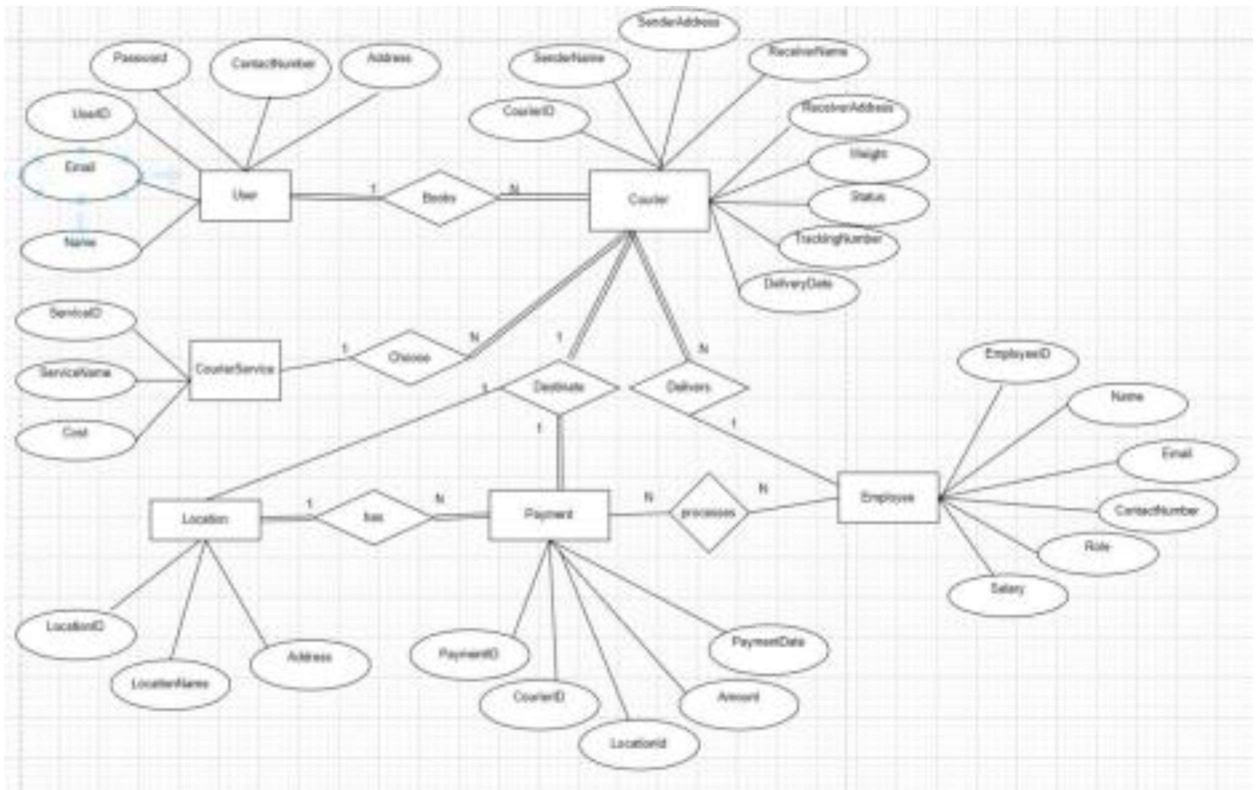
ASSIGNMENT 4

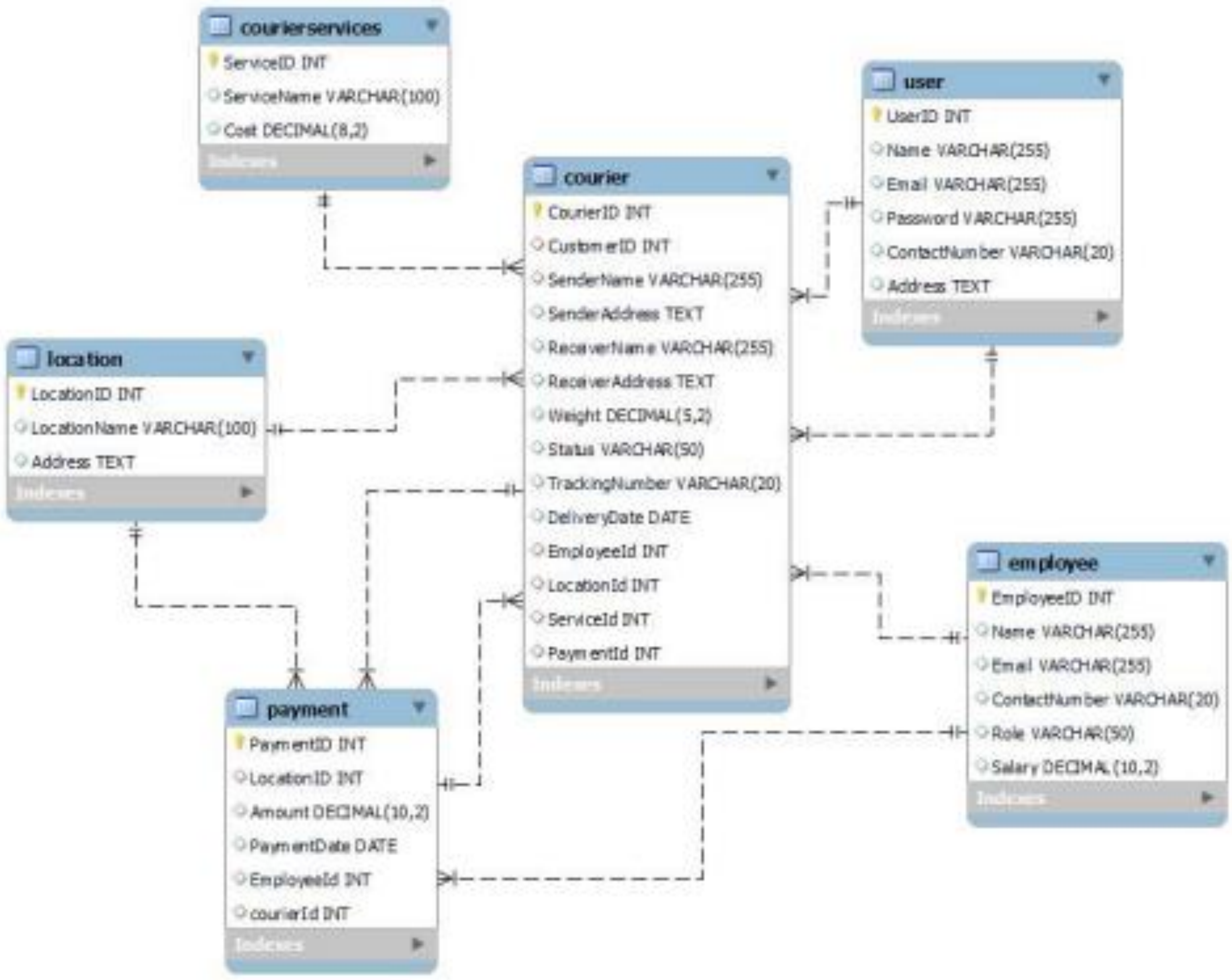
Courier Management System

E-R Diagram



ER diagram for Courier Management system

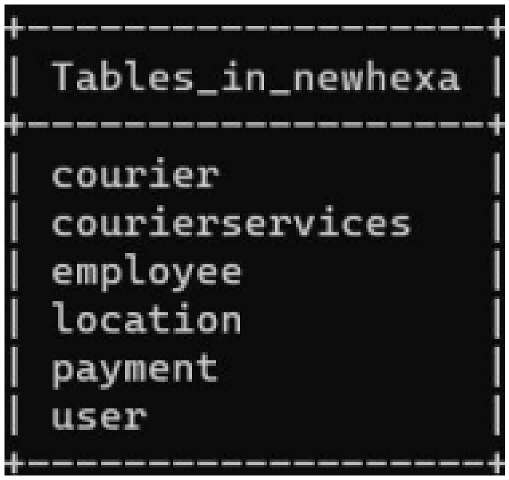
SCHEMA DIAGRAM



Task1 Database Design

create database newhexa;

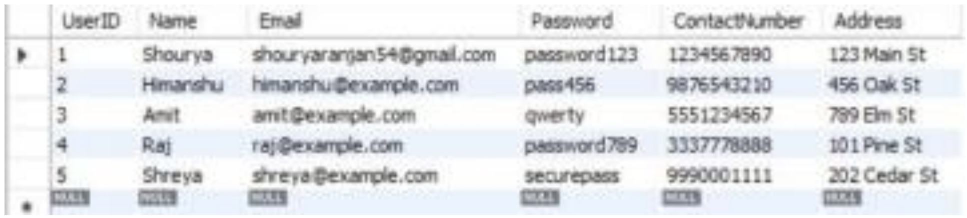
use newhexa;



```
+-------------------+
| Tables_in_newhexa |
+-------------------+
| courier           |
| courierservices   |
| employee          |
| location          |
| payment           |
| user              |
+-------------------+
```

-- User Table

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



UserID	Name	Email	Password	ContactNumber	Address
1	Shourya	shouryaranjan54@gmail.com	password123	1234567890	123 Main St
2	Himanshu	himanshu@example.com	pass456	9876543210	456 Oak St
3	Amit	amit@example.com	qwerty	5551234567	789 Elm St
4	Raj	raj@example.com	password789	3337778888	101 Pine St
5	Shreya	shreya@example.com	securepass	9990001111	202 Cedar St
NULL	NULL	NULL	NULL	NULL	NULL

-- Courier Table

```
CREATE TABLE Courier (
 CourierID INT PRIMARY KEY,
 CustomerID INT, -- Foreign key referencing User table
 SenderName VARCHAR(255),
 SenderAddress TEXT,
 ReceiverName VARCHAR(255),
 ReceiverAddress TEXT,
```

Weight DECIMAL(5, 2),

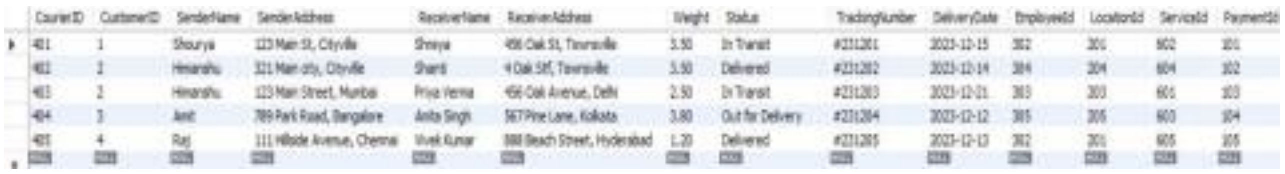
Status VARCHAR(50),

TrackingNumber VARCHAR(20) UNIQUE,

DeliveryDate DATE,

FOREIGN KEY (CustomerID) REFERENCES User(UserID)

);



	CourierID	CustomerID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	EmployeeId	LocationId	ServiceId	PaymentId
▶	401	1	Shourya	123 Main St, Cityville	Shreya	456 Oak St, Townsville	5.50	In Transit	#231201	2023-12-15	302	201	602	101
	402	1	Himanshu	321 Main city, Cityville	Shanti	4 Oak Stf, Townsville	5.50	Delivered	#231202	2023-12-14	304	204	604	102
	403	2	Himanshu	123 Main Street, Mumbai	Priya Verma	456 Oak Avenue, Delhi	2.50	In Transit	#231203	2023-12-21	303	203	601	103
	404	3	Amit	789 Park Road, Bangalore	Anita Singh	567 Pine Lane, Kolkata	3.80	Out for Delivery	#231204	2023-12-12	305	205	603	104
	405	4	Raj	111 Hillside Avenue, Chennai	Vivek Kumar	888 Beach Street, Hyderabad	1.20	Delivered	#231205	2023-12-13	302	201	605	105
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

-- CourierServices Table

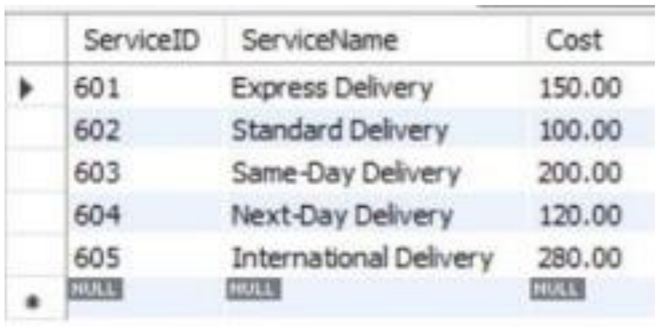
CREATE TABLE CourierServices (

ServiceID INT PRIMARY KEY,

ServiceName VARCHAR(100),

Cost DECIMAL(8, 2)

);



	ServiceID	ServiceName	Cost
▶	601	Express Delivery	150.00
	602	Standard Delivery	100.00
	603	Same-Day Delivery	200.00
	604	Next-Day Delivery	120.00
	605	International Delivery	280.00
*	NULL	NULL	NULL

-- Employee Table

CREATE TABLE Employee (

EmployeeID INT PRIMARY KEY,

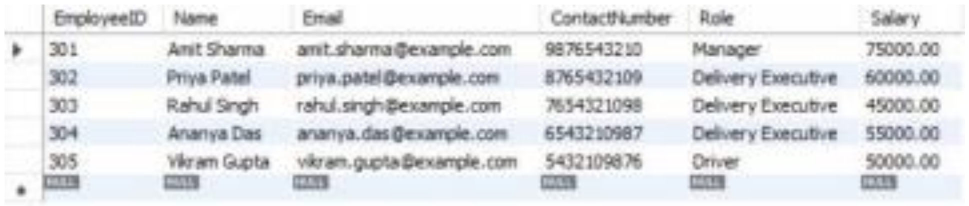
Name VARCHAR(255),

Email VARCHAR(255) UNIQUE,

ContactNumber VARCHAR(20),

Role VARCHAR(50),

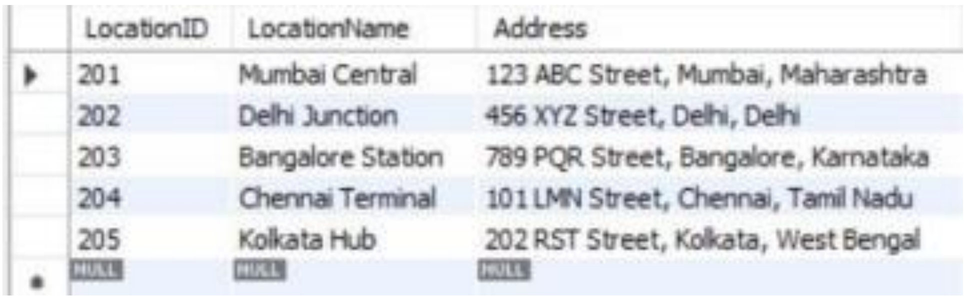
```
 Salary DECIMAL(10, 2)

);
```



EmployeeID	Name	Email	ContactNumber	Role	Salary
301	Amit Sharma	amit.sharma@example.com	9876543210	Manager	75000.00
302	Priya Patel	priya.patel@example.com	8765432109	Delivery Executive	60000.00
303	Rahul Singh	rahul.singh@example.com	7654321098	Delivery Executive	45000.00
304	Ananya Das	ananya.das@example.com	6543210987	Delivery Executive	55000.00
305	Vikram Gupta	vikram.gupta@example.com	5432109876	Driver	50000.00
NULL	NULL	NULL	NULL	NULL	NULL

-- Location Table

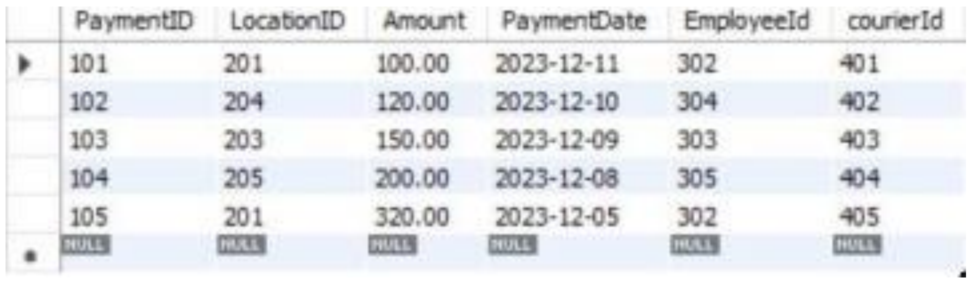
```
CREATE TABLE Location (

 LocationID INT PRIMARY KEY,

 LocationName VARCHAR(100),

 Address TEXT

);
```



LocationID	LocationName	Address
201	Mumbai Central	123 ABC Street, Mumbai, Maharashtra
202	Delhi Junction	456 XYZ Street, Delhi, Delhi
203	Bangalore Station	789 PQR Street, Bangalore, Karnataka
204	Chennai Terminal	101 LMN Street, Chennai, Tamil Nadu
205	Kolkata Hub	202 RST Street, Kolkata, West Bengal
NULL	NULL	NULL

-- Payment Table

```
CREATE TABLE Payment (

 PaymentID INT PRIMARY KEY,

 CourierID INT,

 LocationID INT,

 Amount DECIMAL(10, 2),
 PaymentDate DATE,

 FOREIGN KEY (CourierID) REFERENCES Courier(CourierID),

 FOREIGN KEY (LocationID) REFERENCES Location(LocationID)

);
```



PaymentID	LocationID	Amount	PaymentDate	EmployeeId	courierId
101	201	100.00	2023-12-11	302	401
102	204	120.00	2023-12-10	304	402
103	203	150.00	2023-12-09	303	403
104	205	200.00	2023-12-08	305	404
105	201	320.00	2023-12-05	302	405
NULL	NULL	NULL	NULL	NULL	NULL

-- Insert data into User table

INSERT INTO User (UserID, Name, Email, Password, ContactNumber,

Address) VALUES

(1, 'Shourya', 'shouryaranjan54@gmail.com', 'password123', '1234567890', '123 Main St'),

(2, 'Himanshu', 'himanshu@example.com', 'pass456', '9876543210', '456 Oak St');

-- Insert data into Courier table

INSERT INTO Courier (CourierID, CustomerID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate)

VALUES

(401, 1, 'Shourya', '123 main St, cityville', 'shreya', '456 oak St townsville', 3.50, 'In Transit', 'TN231201', '2023-12-15'),

(402, 2, 'himanshu', '321 main city, cityville', 'shanti', '4 oak St townsville ', 3.50, 'Delivered', 'TN231202', '2023-12-14');

-- Insert data into CourierServices table

INSERT INTO CourierServices (ServiceID, ServiceName, Cost)

VALUES

(601, 'Standard Delivery', 100.00),
(2, 'Express Delivery', 150.00);

-- Insert data into Employee table

INSERT INTO Employee (EmployeeID, Name, Email, ContactNumber, Role,

Salary) VALUES

(1, 'Amit sharma', ‘amit.sharma@example.com', '9876543210', 'Manager', 75000.00), (2, 'Courier Driver', 'driver@example.com', '8765432109', 'delivery executive', 60000.00);

-- Insert data into Location table

```
INSERT INTO Location (LocationID, LocationName, Address)
VALUES
 (201, 'mumbai central', '123 ABC Street, Mumbai, Maharashtra),
 (202, 'Delhi Junction', '456 XYZ Street, Delhi, Delhi');
```

-- Insert data into Payment table

```
INSERT INTO Payment (PaymentID, LocationID, Amount,
PaymentDate) VALUES
 (101, 201, 100.00, '2023-12-11'),
 (2, 204, 120.00, '2023-12-10');
```

<u>-- Task 2</u>

-- 1) List all customers

```
SELECT * FROM User;
```

-- 2) List all orders for a specific customer:

```
SELECT *
FROM Courier
WHERE senderName="Himanshu";
```

-- 3. List all couriers:

```
SELECT * FROM Courier;
```

-- 4. List all packages for a specific order:

```
SELECT *

FROM Courier

WHERE CourierID = 401;
```

-- 5. List all deliveries for a specific courier:

```
SELECT *

FROM Courier

WHERE CourierID = 1;
```

-- 6. List all undelivered packages:

```
SELECT *

FROM Courier

WHERE Status !="Delivered";
```

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

```
SELECT *

FROM Courier

WHERE DeliveryDate = CURDATE();
```

-- 8. List all packages with a specific status:

```
SELECT *

FROM Courier

WHERE Status = 'In Transit';
```

-- 9. Calculate the total number of packages for each courier. (Conflicting

Query) SELECT CourierID, COUNT(*) AS TotalPackages

```
FROM Courier

GROUP BY CourierID;


select * from courier;
```

-- 10. Find the average delivery time for each courier. (tHIS is an estimated delivery date rather than average delivery date)

```
SELECT

 CourierID,

 AVG(DATEDIFF(DeliveryDate, CURDATE())) AS EstimatedDeliveryDays

FROM

 Courier

GROUP BY

 CourierID;
```

-- 11. List all packages with a specific weight range: (Here, 2 to 4)

```
SELECT *

FROM Courier

WHERE Weight BETWEEN 2 AND 4;
```

-- 12. Retrieve employees whose names contain 'ya'

```
SELECT *
FROM Employee

WHERE Name LIKE '%ya%';
```

-- 13. Retrieve all courier records with payments greater than Rs. 50.

```
SELECT Courier.*, Payment.Amount

FROM Courier
```

JOIN Payment ON Courier.CourierID = Payment.CourierID

WHERE Payment.Amount > 150;

<u>-- TASK 3</u>

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

SELECT

 e.EmployeeID,

 e.Name AS EmployeeName,

 COUNT(c.CourierID) AS TotalCouriersHandled

FROM

 Employee e

JOIN

 Courier c ON e.EmployeeID = c.EmployeeID

GROUP BY

 e.EmployeeID, e.Name;

-- For listing all the employeeId who are assigned more than 1 couriers;--

-- select count(courierId)as TotalCouriers, employeeId from courier group by employeeId having totalCouriers >1;--

-- 15. Calculate the total revenue generated by each

location SELECT

 l.LocationID,

 l.LocationName,

 SUM(p.Amount) AS TotalRevenue

FROM

 Location l

JOIN

```
 Payment p ON l.LocationID = p.LocationID
GROUP BY
 l.LocationID;
```

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

```
SELECT
 l.LocationID,
 l.LocationName,
 COUNT(c.CourierID) AS TotalCouriersDelivered
FROM
 Location l
JOIN
 Courier c ON l.LocationID = c.LocationID
WHERE
 c.Status = 'Delivered'
GROUP BY
 l.LocationID;
```

-- 17. Find the courier with the highest average delivery time: (Here, Estimated delivery day instead of average delivery time)

```
SELECT
 CourierID,
 AVG(DATEDIFF(DeliveryDate, CURDATE())) AS AverageDeliveryTime
FROM
 Courier
GROUP BY
 CourierID
```

```
ORDER BY
 AverageDeliveryTime
 DESC
LIMIT 1;
```

-- 18. Find Locations with Total Payments Less Than a Certain Amount (Here, Sum of payments made at a certain location)

```
SELECT
 LocationID,
 LocationName,
 (
 SELECT SUM(p.Amount)
 FROM Payment p
 WHERE p.LocationID = l.LocationID
 ) AS TotalPayments
FROM
 Location l
HAVING
 TotalPayments < 500;
```

-- 19. Calculate Total Payments per Location

```
 SELECT
 LocationID,
 LocationName,
 SUM(Amount) AS TotalPayments
FROM
 Payment
```

```
GROUP BY
 LocationID;
```

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

```
 SELECT
 c.CourierID,
 c.SenderName,
 c.ReceiverName,
 c.TrackingNumber,
 sum(p.Amount) AS TotalPayments
FROM
 Courier c
JOIN
 Payment p ON p.courierId=c.courierId
WHERE
 c.LocationID = 201
GROUP BY
 c.CourierID, c.SenderName, c.ReceiverName, c.TrackingNumber
HAVING
 TotalPayments > 80;
```

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

```
SELECT
 c.CourierID,
 c.SenderName,

 c.ReceiverName,

 c.TrackingNumber,

 SUM(p.Amount) AS TotalPayments

FROM

 Courier c

JOIN

 Payment p ON c.CourierID = p.CourierID

WHERE

 p.PaymentDate > '2023-12-01'

GROUP BY

 c.CourierID, c.SenderName, c.ReceiverName, c.TrackingNumber

HAVING

 TotalPayments > 100;
```

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

```
 SELECT

 l.LocationID,

 l.LocationName,
 SUM(p.Amount) AS TotalAmountReceived

FROM

 Location l

JOIN

 Courier c ON l.LocationID = c.LocationID

JOIN

 Payment p ON c.CourierID = p.CourierID
```

WHERE

 p.PaymentDate > '2022-12-06'

GROUP BY

 l.LocationID, l.LocationName

HAVING

 TotalAmountReceived > 200;

-- TASK 4 --

-- 23. Retrieve Payments with Courier

Information SELECT

 p.PaymentID,

 p.CourierID,

 p.LocationID,

 p.Amount,

 p.PaymentDate,

 c.SenderName,

 c.ReceiverName,

 c.TrackingNumber

FROM
 Payment p

JOIN

 Courier c ON p.CourierID = c.CourierID;

-- 24. Retrieve Payments with Location Information (JOINT ON THE BASIS OF

LOCATION) SELECT

```
 p.PaymentID,
 p.CourierID,
 p.LocationID,
 p.Amount,
 p.PaymentDate,
 l.LocationName,
 l.Address AS LocationAddress
FROM
 Payment p
JOIN
 Location l ON p.LocationID = l.LocationID;
```

-- 25. Retrieve Payments with Courier and Location Information

```
SELECT
 p.PaymentID,
 p.CourierID,
 p.LocationID,
 p.Amount,
 p.PaymentDate,
 c.SenderName,
 c.ReceiverName,
 c.TrackingNumber,
 l.LocationName,
 l.Address AS LocationAddress
FROM
 Payment p
```

```
JOIN
 Courier c ON p.CourierID = c.CourierID
JOIN
 Location l ON p.LocationID = l.LocationID;
```

-- 26. List all payments with courier details SELECT

```
 p.PaymentID,
 p.CourierID,
 p.LocationID,
 p.Amount,
 p.PaymentDate,
 c.SenderName,
 c.SenderAddress,
 c.ReceiverName,
 c.ReceiverAddress,
 c.Weight,
 c.Status,
 c.TrackingNumber,
 c.DeliveryDate
FROM
 Payment p
JOIN
 Courier c ON p.CourierID = c.CourierID;
```

-- 27. Total payments received for each courier

```
SELECT
```

c.CourierID,

c.SenderName,

c.ReceiverName,

c.TrackingNumber,

SUM(p.Amount) AS TotalPaymentsReceived

FROM

Courier c

JOIN

Payment p ON c.CourierID = p.CourierID

GROUP BY

c.CourierID, c.SenderName, c.ReceiverName, c.TrackingNumber;

-- 28. List payments made on a specific date

SELECT

PaymentID,

CourierID,

LocationID,

Amount,

PaymentDate
FROM

Payment

WHERE

PaymentDate = '2023-12-09';

-- 29. Get Courier Information for Each

Payment SELECT

p.PaymentID,

```
 p.CourierID,
 p.LocationID,
 p.Amount,
 p.PaymentDate,
 c.SenderName,
 c.SenderAddress,
 c.ReceiverName,
 c.ReceiverAddress,
 c.Weight,
 c.Status,
 c.TrackingNumber,
 c.DeliveryDate
FROM
 Payment p
JOIN
 Courier c ON p.CourierID = c.CourierID;
-- SELECT
-- *
-- FROM
-- Payment p
-- JOIN
-- Courier c ON p.CourierID = c.CourierID; (--giving repeating data)
```

-- 30. Get Payment Details with Location

```
SELECT
 p.PaymentID,
 p.CourierID,
 p.LocationID,
 p.Amount,
 p.PaymentDate,
 l.LocationName,
 l.Address AS LocationAddress
FROM
 Payment p
JOIN
 Location l ON p.LocationID = l.LocationID;
```

-- 31. Calculating Total Payments for Each Courier

(Conflicting) SELECT

```
 c.CourierID,
 c.SenderName,
 c.ReceiverName,
 c.TrackingNumber,
 SUM(p.Amount) AS TotalPayments
FROM
 Courier c
JOIN
 Payment p ON c.CourierID = p.CourierID
GROUP BY
 c.CourierID, c.SenderName, c.ReceiverName, c.TrackingNumber;
```

-- 32. List Payments Within a Date Range

```
SELECT
 PaymentID,
 CourierID,

 LocationID,

 Amount,

 PaymentDate

FROM

 Payment

WHERE

 PaymentDate BETWEEN '2023-12-05' AND '2023-12-10';
```

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

```
SELECT

 u.UserID, -- CROSS JOIN does not take null values if present.  u.Name AS

UserName,

 u.Email,

 u.ContactNumber,

 u.Address AS UserAddress,

 c.CourierID,
 c.SenderName,

 c.ReceiverName,

 c.TrackingNumber,

 c.Status

FROM

 User u

left join

 Courier c ON u.UserID = c.CustomerId;
```

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

```
SELECT

c.CourierID,

c.customerId,

c.SenderName,

c.ReceiverName,

c.TrackingNumber,

c.Status,

cs.ServiceID,

 cs.ServiceName,

 cs.Cost,

 senderName

from

 Courier c

LEFT JOIN

 CourierServices cs ON c.ServiceID = cs.ServiceID
 order by courierId;
```

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

```
SELECT

 e.EmployeeID,

 e.Name AS EmployeeName,

 e.Email,

e.ContactNumber,
```

e.Role,

-- e.Salary,

 p.PaymentID,

 p.CourierID,

-- p.LocationID,

 p.Amount,

 p.PaymentDate

FROM

 Employee e

LEFT JOIN

 Payment p ON e.EmployeeID = p.EmployeeID;

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

combinations. SELECT

 u.UserID,

 u.Name AS UserName,

 u.Email AS UserEmail,

 u.ContactNumber,
 u.Address AS UserAddress,

 cs.ServiceID,

 cs.ServiceName,

 cs.Cost

FROM

 User u

CROSS JOIN

 CourierServices cs;

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

```
 u.employeeId,
 u.Name,
 u.Email ,
 u.ContactNumber,
 u.Role,
 cs.LocationId,
 cs.LocationName,
 cs.Address

FROM
 employee u
CROSS JOIN
 location cs;
```

-- 38. Retrieve couriers and their corresponding sender information

```
SELECT
 c.CourierID,
 u.UserID AS SenderUserID,
 c.SenderName,
 u.Email AS SenderUserEmail,
 c.SenderAddress,
 c.Weight,
 c.Status,
 c.TrackingNumber,
 c.DeliveryDate
FROM
 Courier c
LEFT JOIN
 User u ON c.SenderName = u.Name;
```

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

```
select courierId, receiverName, receiverAddress, weight, trackingNumber, status from courier;
```

-- 40. Retrieve couriers and their corresponding courier service details

```
SELECT
 c.CourierID,
 c.SenderName,
 c.SenderAddress,
 c.ReceiverName,
 c.ReceiverAddress,
 c.Weight,
 c.Status,
 c.TrackingNumber,
 c.DeliveryDate,
 cs.ServiceID,
 cs.ServiceName,
 cs.Cost
FROM
 Courier c
left JOIN
 CourierServices cs on cs.serviceId=c.serviceId;
```

```
-- 41. Retrieve employees and the number of couriers assigned to each
employee SELECT
 e.EmployeeID,
 e.Name AS EmployeeName,
 e.Email AS EmployeeEmail,
 e.ContactNumber AS EmployeeContact,
 e.Role,
 e.Salary,
 COUNT(c.CourierID) AS NumberOfCouriers
FROM
 Employee e
LEFT JOIN
 Courier c ON e.EmployeeID = c.EmployeeID
GROUP BY
 e.EmployeeID;

-- 42. Retrieve locations and the total payment amount received at each
location SELECT
 l.LocationID,
 l.LocationName,
 l.Address AS LocationAddress,
 SUM(p.Amount) AS TotalPaymentAmount
FROM
 Location l
Left JOIN
 Payment p ON l.LocationID = p.LocationID
```

```
GROUP BY
 l.LocationID;


 SELECT
 l.LocationID,
 l.LocationName,
 l.Address AS LocationAddress,
 (
 SELECT SUM(p.Amount)
 FROM Payment p
 WHERE p.LocationID = l.LocationID  )
AS TotalPaymentAmount
FROM
 Location ld;
```

-- 43. Retrieve all couriers sent by the same sender select * from courier

where senderName='Himanshu';

-- 44. List all employees who share the same role.

```
SELECT * FROM Employee
-- WHERE employeeId IN (
-- SELECT employeeId
-- FROM Employee
 WHERE Role IN (
 SELECT Role
 FROM Employee

 GROUP BY Role

 HAVING COUNT(Role) > 1

);
```

-- 45. Retrieve all payments made for couriers sent from the same location (without using subqueries)

```
select

p.locationId,

l.LocationName,

p.paymentId,

 p.amount,

 p.courierId,

 p.paymentDate

from payment p inner join courier c join location l

on p.locationId=c.locationId

and p.locationId=L.locationId

order by p.locationId;
```

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

```
select

* from courier

 where senderAddress='123 Main Street, Mumbai';
```

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

```
select e.Name, count(status) as TotalOrders from employee e join courier

c on e.employeeId=c.employeeId

where c.status='Delivered'
```

group by e.employeeId;

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

select cs.serviceId, c.courierId, c.SenderName, c.ReceiverName,c.Weight,p.amount, cs.cost, cs.serviceName from courier c join courierservices cs join payment p

on c.serviceId=cs.serviceId and c.courierId=p.courierID

where cs.cost<p.amount;

<u>**-- TASK 4--**</u>

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

select

courierID,

SenderName,

Status,

TrackingNumber,Weight,

DeliveryDate,
ReceiverAddress

from courier where

weight >(select avg(weight) from courier)

order by weight;

select avg(weight) from courier;

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

```
select
name, salary
 from employee
 where salary >(select avg(salary) from employee) ;

select avg(salary) from employee;
```

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

```
SELECT
serviceName,
 Cost AS TotalCost
FROM
 CourierServices
WHERE
 Cost < (SELECT MAX(Cost) FROM CourierServices);
```

-- 52. Find all couriers that have been paid for

```
SELECT
 CourierID,
 SenderName,
 ReceiverName,
 Status,
 PaymentID
FROM
 Courier
WHERE
```

```
 PaymentId is not null ;
```

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

```
select l.locationId, l.locationName, max(p.amount) as MaximumAmount from location l join
payment P where p.locationId=l.locationId and amount=(select max(amount) from payment)
group by l.locationId;
```

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

```
select

CourierID,

 SenderName,

 Status, TrackingNumber, weight

 from courier where weight>=(select max(weight) from courier where
senderName="Himanshu");
```

Coding

Task 1:

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.

```java
import java.util.*;
public class OrderDeliveryStatus
{
    void statusChecker(String orderStatus)
    {

        if (orderStatus.equals("Delivered")) {
            System.out.println("The order has been delivered.");
        } else if (orderStatus.equals("Processing")) {
            System.out.println("The order is still processing.");
        } else if (orderStatus.equals("Cancelled")) {
            System.out.println("The order has been cancelled.");
        } else {
            System.out.println("Invalid order status.");
        }
    }
    public static void main(String[] args) {
        Scanner sc = new  Scanner(System.in);
        System.out.println("Enter your status {Delivered, Processing, 
Cancelled}");
        String inputStatus=sc.next();
        OrderDeliveryStatus obj=new OrderDeliveryStatus();
        obj.statusChecker(inputStatus);
    }
}
```

Output:

```
Enter your status {Delivered, Processing, Cancelled}

Delivered

The order has been delivered.
```

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

Code:

```
import java.util.Scanner;
public class Categorize {
 public static void main(String[] args) {
 Scanner scanner = new Scanner(System.in);
 System.out.print("Enter the weight of the parcel: ");
 int weight = scanner.nextInt();
 String a;
 switch (weight)
 {
 case 0:
 case 1:
 case 2:
 case 3:
 case 4:
 a = "Light";
 break;
 case 5:
 case 6:
 case 7:
 case 8:
 a = "Medium";
 break;
 default:
 a = "Heavy";
 break;
 }
 System.out.println("The weight of the parcel is: " + a);
 }
}
```

Output:

```
Enter the weight of the parcel: 3
The weight of the parcel is: Light
```

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

Code:

```
import java.util.*;
class UserAuth
{
    public static void main(String args[])
    {
        String empid="emp123";
        String emppassword="emp123";
        String custid="cust456";
        String custpassword="cust456";
        Scanner sc= new Scanner(System.in);
        System.out.println("Are you Employee or Customer");
        String input=sc.next();
        if (input.equals("Employee"))
        {
            System.out.println("Enter your employee id");
            String eid=sc.next();
            System.out.println("Enter your employee password");
            String epass=sc.next();
            if (eid.equals(empid) && epass.equals(emppassword))
            {
                System.out.println("Login Sucessfull !! welcome dear employee");
            }
            else{
                System.out.println("Either empid or password is wrong");
            }

        }
        else if(input.equals("Customer"))
        {
            System.out.println("Enter your customer id");
            String cid=sc.next();
            System.out.println("Enter your customer password");
            String cpass=sc.next();
            if (cid.equals(custid) && cpass.equals(custpassword))
            {
                System.out.println("Login Sucessfull !! welcome dear customer");
            }
            else{
                System.out.println("Either  id or password is wrong");
            }
        }
        else{
            System.out.println("invalid input please try again");
        }

    }
}
```

Output:

Are you Employee or Customer

Employee

Enter your employee id

emp123

Enter your employee password

emp123

Login Sucessfull !! welcome dear employee

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.

Code:

```java
import java.util.ArrayList;
import java.util.List;
public class CourierAssignmentSystem {
    public static void main(String[] args) {
        List<Courier> couriers = new ArrayList<>();
        couriers.add(new Courier("Courier1", 10.0, 50.0));
        couriers.add(new Courier("Courier2", 15.0, 40.0));
        couriers.add(new Courier("Courier3", 8.0, 60.0));

        List<Shipment> shipments = new ArrayList<>();
        shipments.add(new Shipment(12.0, 30.0));
        shipments.add(new Shipment(18.0, 45.0));
        shipments.add(new Shipment(9.0, 55.0));
        for (Shipment shipment : shipments) {
            Courier assignedCourier = null;
            double minProximityDifference = Double.MAX_VALUE;

            for (Courier courier : couriers) {
                double proximityDifference = Math.abs(courier.proximity -
shipment.destinationProximity);

                if (proximityDifference < minProximityDifference &&
courier.loadCapacity >= shipment.shipmentLoad) {
                    minProximityDifference = proximityDifference;
                    assignedCourier = courier;
                }
            }

            if (assignedCourier != null) {
                System.out.println("Shipment assigned to " +
assignedCourier.name);
```

```
            } else {
                System.out.println("No available courier for the shipment");
            }
        }
    }
}

class Courier {
    String name;
    double proximity;
    double loadCapacity;

    public Courier(String name, double proximity, double loadCapacity) {
        this.name = name;
        this.proximity = proximity;
        this.loadCapacity = loadCapacity;
    }

}

class Shipment {
    double destinationProximity;
    double shipmentLoad;

    public Shipment(double destinationProximity, double shipmentLoad) {
        this.destinationProximity = destinationProximity;
        this.shipmentLoad = shipmentLoad;
    }
}
```

Output:

```
Shipment assigned to Courier1
Shipment assigned to Courier1
Shipment assigned to Courier3
```

Task 2: Loops and Iteration

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

code:

```
import java.util.ArrayList;
import java.util.List;
public class CustomerOrderDisplay {
```

```java
    public static void main(String[] args) {

        Customer customer1 = new Customer("Cust123");
        customer1.addOrder(new Order(1, "ProductA"));
        customer1.addOrder(new Order(2, "ProductB"));
        customer1.addOrder(new Order(3, "ProductC"));


        displayOrdersForCustomer(customer1);
    }

    private static void displayOrdersForCustomer(Customer customer) {
        System.out.println("Orders for Customer " + customer.getCustomerId() +
":");

        for (Order order : customer.getOrders()) {
            System.out.println("Order ID: " + order.getOrderId() + ", Product: " +
order.getProduct());
        }
    }
}

class Order {
    private int orderId;
    private String product;

    public Order(int orderId, String product) {
        this.orderId = orderId;
        this.product = product;
    }

    public int getOrderId() {
        return orderId;
    }

    public String getProduct() {
        return product;
    }
}

class Customer {
    private String customerId;
    private List<Order> orders;

    public Customer(String customerId) {
        this.customerId = customerId;
        this.orders = new ArrayList<>();
    }

    public String getCustomerId() {
        return customerId;
    }

    public List<Order> getOrders() {
        return orders;
    }

    public void addOrder(Order order) {
        orders.add(order);
    }
}
```

Output:

Orders for Customer Cust123:

Order ID: 1, Product: ProductA

Order ID: 2, Product: ProductB

Order ID: 3, Product: ProductC

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

Code:

```
import java.time.LocalDate;

public class CourierTracking {

    public static void main(String[] args) {

        LocalDate destinationDate = LocalDate.now().plusDays(1);
        while (LocalDate.now().isBefore(destinationDate)) {
            String currentLocation = getCourierLocation();
            System.out.println("Courier's current location: " + currentLocation);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
```

```java
        }
        System.out.println("Courier has reached the destination!");
    }
    private static String getCourierLocation() {

        return "Latitude: 12.9714, Longitude: 77.5946";
    }
}
```

Output:

```
Courier's current location: Latitude: 12.9714, Longitude: 77.5946
Courier's current location: Latitude: 12.9714, Longitude: 77.5946
Courier's current location: Latitude: 12.9714, Longitude: 77.5946
Courier's current location: Latitude: 12.9714, Longitude: 77.5946
Courier's current location: Latitude: 12.9714, Longitude: 77.5946
Courier's current location: Latitude: 12.9714, Longitude: 77.5946
Courier's current location: Latitude: 12.9714, Longitude: 77.5946
```

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.

Code:

```java
import java.util.ArrayList;

public class ParcelTracking {

    public static void main(String[] args) {

        ArrayList<String> trackingHistory = new ArrayList<>();


        updateTrackingHistory(trackingHistory, "Location 1");
        updateTrackingHistory(trackingHistory, "Location 2");
        updateTrackingHistory(trackingHistory, "Location 3");
        displayTrackingHistory(trackingHistory);
    }
    private static void updateTrackingHistory(ArrayList<String> trackingHistory,
String location) {

        trackingHistory.add(location);
    }


    private static void displayTrackingHistory(ArrayList<String> trackingHistory)
{
        System.out.println("Parcel Tracking History:");
        for (int i = 0; i < trackingHistory.size(); i++) {
            System.out.println("Update " + (i + 1) + ": " +
trackingHistory.get(i));
        }
    }
}
```

Output:

Parcel Tracking History:

Update 1: Location 1

Update 2: Location 2

Update 3: Location 3

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

Code:

```
import java.util.ArrayList;
public class CourierService {
    public static void main(String[] args) {
        Courier[] couriers = {
                new Courier("Location1", true),
                new Courier("Location2", true),
                new Courier("Location3", false),
                new Courier("Location4", true),
        };
        String orderLocation = "OrderLocation";
        Courier nearestCourier = findNearestAvailableCourier(couriers,
orderLocation);

        if (nearestCourier != null) {
            System.out.println("Nearest available courier for the order is at " +
nearestCourier.getLocation());
        } else {
            System.out.println("No available couriers nearby.");
        }
    }
    private static Courier findNearestAvailableCourier(Courier[] couriers, String
orderLocation) {
        Courier nearestCourier = null;
        double minDistance = Double.MAX_VALUE;

        for (Courier courier : couriers) {

            if (courier.isAvailable()) {

                double distance = calculateDistance(orderLocation,
courier.getLocation());


                if (distance < minDistance) {
                    minDistance = distance;
                    nearestCourier = courier;
                }
            }
        }

        return nearestCourier;
    }

    private static double calculateDistance(String location1, String location2) {

        return Math.abs(location1.hashCode() - location2.hashCode());
    }
}
class Courier {
    private String location;
    private boolean isAvailable;
    public Courier(String location, boolean isAvailable) {
        this.location = location;
        this.isAvailable = isAvailable;
    }

    public String getLocation() {
        return location;
    }
```

```java
    public boolean isAvailable() {
        return isAvailable;
    }
}
```

Output:

Nearest available courier for the order is at Location1

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.

Code:

```java
import java.util.Scanner;

public class ParcelTrackingSystem {

    public static void main(String[] args) {

        String[][] trackingArray = {
                {"123456", "Parcel in transit"},
                {"789012", "Parcel out for delivery"},
                {"345678", "Parcel delivered"}
        };
        displayTrackingInformation(trackingArray);
        simulateTrackingProcess(trackingArray);
    }
    private static void displayTrackingInformation(String[][] trackingArray) {
        System.out.println("Initial Tracking Information:");
        for (String[] tracking : trackingArray) {
            System.out.println("Tracking Number: " + tracking[0] + ", Status: " +
tracking[1]);
        }
    }
    private static void simulateTrackingProcess(String[][] trackingArray) {
        Scanner scanner = new Scanner(System.in);


        System.out.print("\nEnter a Parcel Tracking Number: ");
        String inputTrackingNumber = scanner.nextLine();
        String status = findTrackingStatus(trackingArray, inputTrackingNumber);


        if (status != null) {
            switch (status.toLowerCase()) {
                case "parcel in transit":
                    System.out.println("Parcel in transit. Expected delivery
soon.");
                    break;
                case "parcel out for delivery":
                    System.out.println("Parcel is out for delivery. Please be
available to receive it.");
                    break;
                case "parcel delivered":
                    System.out.println("Parcel has been delivered. Thank you for
choosing our service.");
                    break;
                default:
                    System.out.println("Invalid tracking status.");
            }
        } else {
            System.out.println("Tracking number not found.");
        }

        scanner.close();
    }


    private static String findTrackingStatus(String[][] trackingArray, String
trackingNumber) {
        for (String[] tracking : trackingArray) {
            if (tracking[0].equals(trackingNumber)) {
                return tracking[1];
            }
        }
        return null;
    }
}
```

Output:

Tracking Number: 123456, Status: Parcel in transit

Tracking Number: 789012, Status: Parcel out for delivery

Tracking Number: 345678, Status: Parcel delivered

Enter a Parcel Tracking Number: 123456

Parcel in transit. Expected delivery soon.

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., ###-###-####).

Code:

```java
import java.util.regex.*;

public class CustomerDataValidation {

    public static void main(String[] args) {

        String name = "John Doe";
        String address = "123 Main Street";
        String phoneNumber = "555-123-4567";

        if (validateCustomerInformation(name, "name")) {
            System.out.println("Name is valid: " + name);
        } else {
            System.out.println("Invalid name: " + name);
        }

        if (validateCustomerInformation(address, "address")) {
            System.out.println("Address is valid: " + address);
        } else {
            System.out.println("Invalid address: " + address);
        }

        if (validateCustomerInformation(phoneNumber, "phone number")) {
            System.out.println("Phone number is valid: " + phoneNumber);
        } else {
            System.out.println("Invalid phone number: " + phoneNumber);
        }
    }
    private static boolean validateCustomerInformation(String data, String detail)
{
        if (detail.equalsIgnoreCase("name")) {
            return data.matches("^[A-Z][a-z]+\\s[A-Z][a-z]+$");
        } else if (detail.equalsIgnoreCase("address")) {

            return data.matches("^[\\w\\s]+$");
        } else if (detail.equalsIgnoreCase("phone number")) {

            return data.matches("^\\d{3}-\\d{3}-\\d{4}$");
        } else {

            System.out.println("Invalid detail for validation.");
            return false;
        }
    }
}
```

Output:

Name is valid: John Doe

Address is valid: 123 Main Street

Phone number is valid: 555-123-4567

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.

Code:

```
import java.util.*;

public class AddressFormatter {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter your street");
        String street=sc.nextLine();
        System.out.println("Enter your city");
        String city=sc.nextLine();
        System.out.println("Enter your state");
        String state=sc.nextLine();
        System.out.println("Enter your zipCode");
        String zipCode=sc.nextLine();
        String formattedAddress = formatAddress(street, city, state, zipCode);
        System.out.println("Formatted Address: " + formattedAddress);
    }
    private static String formatAddress(String street, String city, String state,
String zipCode) {

        street = capitalizeEachWord(street);
        city = capitalizeEachWord(city);
        state = capitalizeEachWord(state);


        String formattedZipCode = formatZipCode(zipCode);


        return street + ", " + city + ", " + state + "-" + formattedZipCode;
    }
    private static String capitalizeEachWord(String input) {
        String[] words = input.split("\\s+");
        StringBuilder result = new StringBuilder();

        for (String word : words) {
            result.append(word.substring(0, 1).toUpperCase())
                    .append(word.substring(1).toLowerCase())
                    .append(" ");
        }

        return result.toString().trim();
    }


    private static String formatZipCode(String zipCode) {

        if (zipCode.matches("^\\d{5}$")) {
            return zipCode;
        } else {

            return zipCode;
        }
    }
}
```

Output:

Enter your street

madhamvaripalli

Enter your city

Chottoor

Enter your state

Andhra Pradesh

Enter your zipCode

12345

Formatted Address: Madhamvaripalli, Chottoor, Andhra Pradesh-12345

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.

Code:

```
import java.time.LocalDate;

public class OrderConfirmationEmailGenerator {

    public static void main(String[] args) {

        String customerName = "John Doe";
        int orderNumber = 123456;
        String deliveryAddress = "123 Main Street, Cityville, State, 12345";
        LocalDate expectedDeliveryDate = LocalDate.now().plusDays(3);


        String emailContent = generateOrderConfirmationEmail(customerName,
orderNumber, deliveryAddress, expectedDeliveryDate);


        System.out.println("Order Confirmation Email:\n" + emailContent);
    }
    private static String generateOrderConfirmationEmail(String customerName, int
orderNumber, String deliveryAddress, LocalDate expectedDeliveryDate) {
        StringBuilder emailContent = new StringBuilder();
        emailContent.append("Dear ").append(customerName).append(",\n\n")
                .append("Thank you for your order! Here are the details:\n\n");
        emailContent.append("Order Number: ").append(orderNumber).append("\n")
                .append("Delivery Address: ").append(deliveryAddress).append("\n")
                .append("Expected Delivery Date:
").append(expectedDeliveryDate).append("\n\n");
```

```
        emailContent.append("If you have any questions or concerns, please contact 
our customer support.\n\n")
                .append("Thank you for choosing our service!\n")
                .append("Best regards,\n")
                .append("Hexaware");

        return emailContent.toString();
    }
}
```

Output:

Order Confirmation Email:

Dear John Doe,

Thank you for your order! Here are the details:

Order Number: 123456

Delivery Address: 123 Main Street, Cityville, State, 12345

Expected Delivery Date: 2024-01-01

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

Thank you for choosing our service!

Best regards,

Hexaware.

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.

Code:

```
import java.util.*;
public class ShippingCostCalculator {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter source address");
        String sourceAddress=sc.nextLine();
        System.out.println("Enter destination address");
        String destinationAddress=sc.nextLine();
        System.out.println("Enter parcelweight");
        double parcelWeight=sc.nextDouble();
        double shippingCost = calculateShippingCost(sourceAddress,
destinationAddress, parcelWeight);
        System.out.println("Shipping Cost: Rs " + shippingCost);
    }
    private static double calculateShippingCost(String sourceAddress, String
destinationAddress, double parcelWeight) {

        double distance = calculateDistance(sourceAddress, destinationAddress);

        double costPerKilometer = 0.10;
        double costPerKilogram = 1.00;

        return distance * costPerKilometer + parcelWeight * costPerKilogram;
    }
    private static double calculateDistance(String sourceAddress, String
destinationAddress) {
        return Math.abs(sourceAddress.hashCode() - destinationAddress.hashCode());
    }
}
```

Output:

```
Enter source address
Chennai
Enter destination address
Hyderabad
Enter parcelweight
5
Shipping Cost: Rs 1.712479574E8
```

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.

Code:

```
import java.security.SecureRandom;

public class PasswordGenerator {

    public static void main(String[] args) {

        String generatedPassword = generatePassword();
        System.out.println("Generated Password: " + generatedPassword);
```

```java
    }
    private static String generatePassword() {

        String uppercaseLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        String lowercaseLetters = "abcdefghijklmnopqrstuvwxyz";
        String numbers = "0123456789";
        String specialCharacters = "!@#$%^&*()-_=+[]{}|;:'\",.<>?/";
        String allCharacters = uppercaseLetters + lowercaseLetters + numbers +
specialCharacters;
        int passwordLength = 12;
        SecureRandom random = new SecureRandom();
        StringBuilder passwordBuilder = new StringBuilder(passwordLength);
        passwordBuilder.append(uppercaseLetters.charAt(random.nextInt(uppercaseLet
ters.length())));
        passwordBuilder.append(lowercaseLetters.charAt(random.nextInt(lowercaseLet
ters.length())));
        passwordBuilder.append(numbers.charAt(random.nextInt(numbers.length())));
        passwordBuilder.append(specialCharacters.charAt(random.nextInt(specialChar
acters.length())));
        for (int i = 4; i < passwordLength; i++) {
            passwordBuilder.append(allCharacters.charAt(random.nextInt(allCharacte
rs.length())));
        }
        char[] passwordChars = passwordBuilder.toString().toCharArray();
        for (int i = passwordChars.length - 1; i > 0; i--) {
            int index = random.nextInt(i + 1);
            char temp = passwordChars[index];
            passwordChars[index] = passwordChars[i];
            passwordChars[i] = temp;
        }
        return new String(passwordChars);
    }
}
```

Output:

Generated Password: Oj::k[pZKC!4

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.

Code:

```java
import java.util.HashSet;
import java.util.Set;
```

```java
public class AddressSimilarityFinder {

    public static void main(String[] args) {

        String address1 = "123 Main St, Cityville, State1, 12345";
        String address2 = "123 Main Street, Cityville, State2, 54321";
        String address3 = "456 Broadway, Townsville, State3, 67890";
        if (areAddressesSimilar(address1, address2)) {
            System.out.println("Address 1 and Address 2 are similar.");
        } else {
            System.out.println("Address 1 and Address 2 are not similar.");
        }

        if (areAddressesSimilar(address1, address3)) {
            System.out.println("Address 1 and Address 3 are similar.");
        } else {
            System.out.println("Address 1 and Address 3 are not similar.");
        }
    }
    private static boolean areAddressesSimilar(String address1, String address2) {

        Set<String> set1 = tokenizeAddress(address1);
        Set<String> set2 = tokenizeAddress(address2);
        double intersectionSize = intersectionSize(set1, set2);
        double unionSize = unionSize(set1, set2);
        double similarityThreshold = 0.6;
        return (intersectionSize / unionSize) >= similarityThreshold;
    }
    private static Set<String> tokenizeAddress(String address) {
        String[] words = address.split("[\\s,]+");
        Set<String> tokenSet = new HashSet<>();
        for (String word : words) {
            tokenSet.add(word.toLowerCase());
        }
        return tokenSet;
    }
    private static double intersectionSize(Set<String> set1, Set<String> set2) {
        Set<String> intersection = new HashSet<>(set1);
        intersection.retainAll(set2);
        return intersection.size();
    }


    private static double unionSize(Set<String> set1, Set<String> set2) {
        Set<String> union = new HashSet<>(set1);
        union.addAll(set2);
        return union.size();
    }
}
```

Output:

Address 1 and Address 2 are not similar.

Address 1 and Address 3 are not similar.

Task5: Object Oriented Programming

<u>Package Entity</u>

User.java

```java
package entities;


public class User {
    private int userID;
    private String userName;
    private String email;
    private String password;
    private String contactNumber;
    private String address;

    // Default Constructor
    public User() {
    }

    // Parameterized Constructor
    public User(int userID, String userName, String email, String password,
String contactNumber, String address) {
        this.userID = userID;
        this.userName = userName;
        this.email = email;
        this.password = password;
        this.contactNumber = contactNumber;
        this.address = address;
    }

    // Getters and Setters
    public int getUserID() {
        return userID;
    }

    public void setUserID(int userID) {
        this.userID = userID;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }

    public String getContactNumber() {
        return contactNumber;
    }

    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    // toString method
    @Override
    public String toString() {
        return "User{" +
                "userID=" + userID +
                ", userName='" + userName + '\'' +
                ", email='" + email + '\'' +
                ", password='" + password + '\'' +
                ", contactNumber='" + contactNumber + '\'' +
                ", address='" + address + '\'' +
                '}';
    }


}
```

Courier.java

```java
package entities;

public class Courier {

      public static  long SysTrackingNumber=(long) (Math.random() * 900000) +
100000;
    private int courierID;
    private String senderName;
    private String senderAddress;
    private String receiverName;
    private String receiverAddress;
    private double weight;
    private String status;
    private long trackingNumber;
    private String deliveryDate;
    private int userId;
    private int empid;
    // Default Constructor
    public Courier() {
      SysTrackingNumber++;
    }
    public int getCourierID() {
        return courierID;
    }

    public Courier(int courierID, String senderName, String senderAddress,
String receiverName, String receiverAddress,
                  double weight, String status, long trackingNumber, String
deliveryDate, int userId, int empid) {
            super();
            this.courierID = courierID;
            this.senderName = senderName;
            this.senderAddress = senderAddress;
            this.receiverName = receiverName;
            this.receiverAddress = receiverAddress;
            this.weight = weight;
            this.status = status;
            this.trackingNumber = trackingNumber;
            this.deliveryDate = deliveryDate;
            this.userId = userId;
            this.empid = empid;
      }

      public void setCourierID(int courierID) {
        this.courierID = courierID;
    }

    public String getSenderName() {
        return senderName;
    }

    public void setSenderName(String senderName) {
        this.senderName = senderName;
    }

    public String getSenderAddress() {
        return senderAddress;
    }

    public void setSenderAddress(String senderAddress) {
        this.senderAddress = senderAddress;
    }

    public String getReceiverName() {
        return receiverName;
    }

    public void setReceiverName(String receiverName) {
        this.receiverName = receiverName;
    }

    public String getReceiverAddress() {
```

```java
        return receiverAddress;
    }

    public void setReceiverAddress(String receiverAddress) {
        this.receiverAddress = receiverAddress;
    }

    public double getWeight() {
        return weight;
    }

    public void setWeight(double weight) {
        this.weight = weight;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }

    public long getTrackingNumber() {
        return trackingNumber;
    }

    public void setTrackingNumber(long trackingNumber) {
        this.trackingNumber = trackingNumber;
    }

    public String getDeliveryDate() {
        return deliveryDate;
    }

    public void setDeliveryDate(String deliveryDate) {
        this.deliveryDate = deliveryDate;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }
    public int getEmpId() {
        return userId;
    }

    public void setEmpId(int userId) {
        this.userId = userId;
    }

    // toString method
    @Override
    public String toString() {
        return "Courier{" +
                "courierID=" + courierID +
                ", senderName='" + senderName + '\'' +
                ", senderAddress='" + senderAddress + '\'' +
                ", receiverName='" + receiverName + '\'' +
                ", receiverAddress='" + receiverAddress + '\'' +
                ", weight=" + weight +
                ", status='" + status + '\'' +
                ", trackingNumber='" + trackingNumber + '\'' +
                ", deliveryDate=" + deliveryDate +
                ", userId=" + userId +
                ", empid="+empid+
                '}';

    }

}
```

Employee.java

```java
package entities;

public class Employee {
    private int employeeID;
    private String employeeName;
    private String email;
    private String contactNumber;
    private String role;
    private double salary;

    // Default Constructor
    public Employee() {
    }

    // Parameterized Constructor
    public Employee(int employeeID, String employeeName, String email, String
contactNumber, String role, double salary) {
        this.employeeID = employeeID;
        this.employeeName = employeeName;
        this.email = email;
        this.contactNumber = contactNumber;
        this.role = role;
        this.salary = salary;
    }

    // Getters and Setters
    public int getEmployeeID() {
        return employeeID;
    }

    public void setEmployeeID(int employeeID) {
        this.employeeID = employeeID;
    }

    public String getEmployeeName() {
        return employeeName;
    }

    public void setEmployeeName(String employeeName) {
        this.employeeName = employeeName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getContactNumber() {
        return contactNumber;
    }

    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    // toString method
    @Override
    public String toString() {
        return "Employee{" +
                "employeeID=" + employeeID +
                ", employeeName='" + employeeName + '\'' +
                ", email='" + email + '\'' +
                ", contactNumber='" + contactNumber + '\'' +
                ", role='" + role + '\'' +
                ", salary=" + salary +
                '}';
    }
}
```

Location.java

```
package entities;
```

```java
public class Location {
    private int locationID;
    private String locationName;
    private String address;

    // Default Constructor
    public Location() {
    }

    // Parameterized Constructor
    public Location(int locationID, String locationName, String address) {
        this.locationID = locationID;
        this.locationName = locationName;
        this.address = address;
    }

    // Getters and Setters
    public int getLocationID() {
        return locationID;
    }

    public void setLocationID(int locationID) {
        this.locationID = locationID;
    }

    public String getLocationName() {
        return locationName;
    }

    public void setLocationName(String locationName) {
        this.locationName = locationName;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    // toString method
    @Override
    public String toString() {
        return "Location{" +
                "locationID=" + locationID +
                ", locationName='" + locationName + '\'' +
                ", address='" + address + '\'' +
                '}';
    }
}
```

CourierCompany.java

```java
package entities;

import java.util.List;

public class CourierCompany {
    private String companyName;
    private List<Courier> courierDetails;
    private List<Employee> employeeDetails;
    private List<Location> locationDetails;

    // Default Constructor
    public CourierCompany() {
    }

    // Parameterized Constructor
    public CourierCompany(String companyName, List<Courier> courierDetails,
List<Employee> employeeDetails,
                          List<Location> locationDetails) {
        this.companyName = companyName;
        this.courierDetails = courierDetails;
        this.employeeDetails = employeeDetails;
        this.locationDetails = locationDetails;
    }

    // Getters and Setters
    public String getCompanyName() {
        return companyName;
    }

    public void setCompanyName(String companyName) {
        this.companyName = companyName;
    }

    public List<Courier> getCourierDetails() {
        return courierDetails;
    }

    public void setCourierDetails(List<Courier> courierDetails) {
        this.courierDetails = courierDetails;
    }

    public List<Employee> getEmployeeDetails() {
        return employeeDetails;
    }

    public void setEmployeeDetails(List<Employee> employeeDetails) {
        this.employeeDetails = employeeDetails;
    }

    public List<Location> getLocationDetails() {
        return locationDetails;
    }

    public void setLocationDetails(List<Location> locationDetails) {
        this.locationDetails = locationDetails;
    }
    // toString method
    @Override
    public String toString() {
        return "CourierCompany{" +
                "companyName='" + companyName + '\'' +
                ", courierDetails=" + courierDetails +
                ", employeeDetails=" + employeeDetails +
                ", locationDetails=" + locationDetails +
                '}';
    }
}
```

Payment.java

```java
package entities;
import java.util.*;


public class Payment {
    private long paymentID;
    private long courierID;
    private double amount;
    private Date paymentDate;

    // Default Constructor
    public Payment() {
    }

    // Parameterized Constructor
    public Payment(long paymentID, long courierID, double amount, Date
paymentDate) {
        this.paymentID = paymentID;
        this.courierID = courierID;
        this.amount = amount;
        this.paymentDate = paymentDate;
    }

    // Getters and Setters
    public long getPaymentID() {
        return paymentID;
    }

    public void setPaymentID(long paymentID) {
        this.paymentID = paymentID;
    }

    public long getCourierID() {
        return courierID;
    }

    public void setCourierID(long courierID) {
        this.courierID = courierID;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public Date getPaymentDate() {
        return paymentDate;
    }

    public void setPaymentDate(Date paymentDate) {
        this.paymentDate = paymentDate;
    }

    // toString method
    @Override
    public String toString() {
        return "Payment{" +
                "paymentID=" + paymentID +
                ", courierID=" + courierID +
                ", amount=" + amount +
                ", paymentDate=" + paymentDate +
                '}';
    }
}
```

<u>Package Dao</u>

IcourierAdminService.java

```
package dao;
import entities.Employee;
//ICourierAdminService Interface
public interface ICourierAdminService {
boolean addCourierStaff(Employee Obj);
}
```

IcouierUserservice.java

```
package dao;
import java.util.*;

import entities.Courier;
//ICourierUserService Interface
public interface ICourierUserService {
 // Customer-related functions
 long placeOrder(Courier courierObj);

 String getOrderStatus(long trackingNumber);

 boolean cancelOrder(long trackingNumber);

 List<Courier> getAssignedOrders(int courierStaffId);
}
```

Courieradminserviceimpl.java

```
package dao;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import entities.Employee;
import exception.*;

public class CourierAdminServiceImpl extends CourierUserServiceImpl implements
ICourierAdminService{
	static Connection conn = CourierServiceDb.connection;
	@Override
	public boolean addCourierStaff(Employee Obj) {
			boolean status=false;
			try {
				Statement stmt=conn.createStatement();
				String query = "INSERT INTO employee VALUES (" +
					Obj.getEmployeeID() + ", " +
					"'" + Obj.getEmployeeName() + "', " +

					"'" + Obj.getEmail() + "', " +
					"'" + Obj.getContactNumber() + "', "+
				"'" + Obj.getRole() + "', " +
				Obj.getSalary() +"); " ;
//				System.out.println(query);
				return stmt.execute(query);
			}
			catch(Exception e) {
				System.out.println(e.getMessage());
			}
			return status;
		}
	public void courierStatusUpdate(long trackingNumber, String Status) {
		try {
			Statement stmt=conn.createStatement();
			String query = "update courier set status='"+Status+"' where
trackingnumber='"+trackingNumber+"';";
			int a=stmt.executeUpdate(query);
			//System.out.println(a);
			if (a>0)
			{
				System.out.println("Updated status to "+Status+" for
the tracking number "+trackingNumber);
			}
			else
			{
				throw new
TrackingNumberNotFoundException(trackingNumber);
			}
		}
		catch(Exception e) {
			System.out.println(e.getMessage());
		}
	}
	public void employeeList() {
		try {
			Statement stmt=conn.createStatement();
			String query = "select  * from employee;";
			ResultSet res= stmt.executeQuery(query);
			int c=0;
			 System.out.println("-----------------------------------------------
----------------------------------------------------------");
			System.out.printf("| %-6s | %-14s | %-20s | %-30s | %-10s
|\n", "S.no", "EmployeeID", "Name", "Email", "Salary");
			System.out.println("----------------------------------------------
-------------------------------------------------------");
			while (res.next()) {
				System.out.printf("| %-6d | %-14d | %-20s | %-30s | %-
10.2f |\n",
						++c, res.getInt("EmployeeId"),
res.getString("Name"),
						res.getString("Email"), res.getDouble("salary"));
			}
			System.out.println("---------------------------------------------
---------------------------------------------------------");
		}
		catch(Exception e) {
			System.out.println(e.getMessage());
		}
	}

	public void deleteEmployee(int empid) {
		try {
			Statement stmt=conn.createStatement();
			String query = "delete from employee where
employeeid="+empid+";";

			int a=stmt.executeUpdate(query);
			if (a>0)
			{
				System.out.println("Employee Deleted Successfully!!");
			}
			else {
				throw new InvalidEmployeeIdException(empid);
			}
		}
		catch(Exception e) {
			System.out.println(e.getMessage());
		}
	}

}
```

CourierUserServiceImpl.java

```
package dao;

import java.util.ArrayList;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import entities.Courier;
import exception.*;

public class CourierUserServiceImpl implements ICourierUserService {
	static Connection conn = CourierServiceDb.connection;

	@Override
	public long placeOrder(Courier courierObj) {









			boolean status=false;
			try {

				// Logic to generate random employee Id
				String qq="SELECT COUNT(*) from employee;";
				PreparedStatement pstmt = conn.prepareStatement(qq);
				ResultSet EmpCount=pstmt.executeQuery();
				EmpCount.next();
				int empID=(int) (Math.random() *
EmpCount.getInt("COUNT(*)")) + 1;

				// Logic to insert values in courier
				String query = "INSERT INTO courier (SenderName,
userId, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status,
TrackingNumber, DeliveryDate, empId) VALUES (?, ?, ?, ?, ?, ?, ?, ?, CURDATE(),
?)";
				PreparedStatement stmt = conn.prepareStatement(query,
PreparedStatement.RETURN_GENERATED_KEYS);

				stmt.setString(1, courierObj.getSenderName());
				stmt.setInt(2, courierObj.getUserId());
				stmt.setString(3, courierObj.getSenderAddress());
				stmt.setString(4, courierObj.getReceiverName());
				stmt.setString(5, courierObj.getReceiverAddress());
				stmt.setDouble(6, courierObj.getWeight());
				stmt.setString(7, "Pending");
				stmt.setDouble(8, Courier.SysTrackingNumber);
				stmt.setInt(9, empID);


				// Execute the prepared statement
				status=stmt.executeUpdate()>0;

				 if(status) {
					System.out.println("Order Placed
Successfully!!");
					return Courier.SysTrackingNumber;
				 }
				}
				catch(Exception e) {
					System.out.println(e.getMessage());
				}

				return -1;

		}

		@Override
		public String getOrderStatus(long trackingNumber) {

			try {
				Statement stmt = conn.createStatement();
				String query = "select * from courier where trackingNumber='"
+ trackingNumber + "';";
//				System.out.println(query);

				ResultSet res = stmt.executeQuery(query);

				String result;
				if (res.next()) {
					result = res.getString("status");
					return result;
				} else {
					throw new
TrackingNumberNotFoundException(trackingNumber);

				}
			} catch (Exception e) {
				System.out.println(e.getMessage());
			}
			return " ";
		}

		@Override
		public boolean cancelOrder(long trackingNumber) {

			boolean status = false;
			try {
				Statement stmt = conn.createStatement();
				String query = "delete from courier where trackingNumber='" +
trackingNumber + "';";
//				System.out.println(query);
				stmt.execute("use courier");
				// System.out.println(stmt.execute(query));
				status= stmt.executeUpdate(query)>0;
				// System.out.println(a);
				if(status) {
					return status;
				}
				else {
					throw new
TrackingNumberNotFoundException(trackingNumber);
				}
```

```
            } catch (Exception e) {
                  System.out.println(e.getMessage());
            }
            return status;

      }

      @Override
      public ArrayList<Courier> getAssignedOrders(int empid) {


            try {
                  Statement stmt = conn.createStatement();
                  String query = "select  * from courier where empid=" + empid
+ ";";
//                System.out.println(query);
                  ResultSet res = stmt.executeQuery(query);

                  ArrayList<Courier> arr = new ArrayList<Courier>();
                  while (res.next()) {
                        arr.add(new
Courier(res.getInt("courierID"),res.getString("senderName"),res.getString("send
erAddress"),res.getString("receiverName"),res.getString("receiverAddress"),res.
getDouble("weight"),res.getString("status"),

      res.getLong("trackingNumber"),res.getString("deliveryDate"),res.getInt("u
serId"),res.getInt("empid")));
                  }
                  if(arr.isEmpty()) {
                        throw new InvalidEmployeeIdException(empid);
                  }

                  return arr;
            } catch (Exception e) {
                  System.out.println(e.getMessage());
            }
            return null;

      }

      public void getOrderHistory(long trackingNumber) {
            try {
                  Statement stmt = conn.createStatement();
                  String query = "select * from courier where trackingNumber='"
+ trackingNumber + "';";
                  ResultSet res = stmt.executeQuery(query);
                  if(res==null) {
                        throw new
TrackingNumberNotFoundException(trackingNumber);
                  }

                  if (res.next()) {
                        String status = res.getString("status");
                        String senderName = res.getString("senderName");
                        String deliveryDate = res.getString("deliveryDate");
                        if (status.equals("Pending"))
                        {
                              System.out.println("--------------Courier Order 
History----------------");
                              System.out.println("-------------------------------
--------------------");
                          System.out.printf("| %-20s | %-23s |\n", "Field", 
"Value");
                          System.out.println("-----------------------------------
----------------");
                          System.out.printf("| %-20s | %-23s |\n", "Courier 
Sent by", senderName);
                          System.out.printf("| %-20s | %-23s |\n", "Expected 
Delivery by", deliveryDate);
                          System.out.printf("| %-20s | %-23s |\n", "Status", 
"Pending");
                          System.out.println("-----------------------------------
----------------");
                        }
                        else if (status.equals("In Transit")) {
                        System.out.println("--------------Courier Status Report-
----------------");
                        System.out.println("-----------------------------------
---------------");
                    System.out.printf("| %-20s | %-23s |\n", "Field", "Value");
                    System.out.println("-----------------------------------------
----------");
                    System.out.printf("| %-20s | %-23s |\n", "Courier Sent by", 
senderName);
                    System.out.printf("| %-20s | %-23s |\n", "Expected Delivery 
by", deliveryDate);
                    System.out.printf("| %-20s | %-23s |\n", "Status", "In 
Transit");
                    System.out.println("-----------------------------------------
----------");
                        }
                        else {
                              System.out.println(
                                          "Courier Sent by : " + senderName + 
"\nDelivered on : " + deliveryDate);
                  }}} catch (Exception e) {
                  System.out.println(e.getMessage());
            }

      }
}
```

CourierStaff.java

```
package dao;
import java.util.Scanner;
import entities.Employee;

public class CourierStaff {

    public  void couriermenu() {
        int choice;
```

```java
        Scanner scanner = new Scanner(System.in);
        CourierAdminServiceImpl obj=new CourierAdminServiceImpl();
        do {
            //displayMenu();
            System.out.println("1. Add an Employee\n"
                        + "2. Delete Employee\n"
                        + "3. Update Courier Status\n"
                        + "4. Get Employee List \n"
                        + "5. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                  System.out.println("Enter employeeid: ");
                  int EmployeeID =scanner.nextInt();
                  scanner.nextLine();
                  System.out.println("Enter employee name: ");
                  String Name=scanner.nextLine();

                  System.out.println("Enter email: ");
                  String email=scanner.nextLine();

                  System.out.println("Enter phone number: ");
                  String contactNumber=scanner.nextLine();
                  System.out.println("Enter employee role: ");
                  String role=scanner.nextLine();
                  System.out.println("Enter salary: ");
                  double salary=scanner.nextDouble();
                  Employee empobj=new
Employee(EmployeeID,Name,email,contactNumber,role,salary);
                    System.out.println("Placing a new employee details...");

                    obj.addCourierStaff(empobj);


                    break;

                case 2:
                  System.out.println("Enter employeeid: ");
                  int EmpID =scanner.nextInt();
                  obj.deleteEmployee(EmpID);
                    break;

                case 3:
                    // Cancel Order
                    System.out.println("Enter tracking number: ");
                    long trackingNumber=scanner.nextLong();
                    System.out.println("Enter status");
                    String status=scanner.next();
                    obj.courierStatusUpdate(trackingNumber,status);
                    break;

                case 4:
                  obj.employeeList();
                    break;
                case 5:
                    System.out.println("!!Thanks For Using Courier Management
System!!");
                    break;
                default:
                    System.out.println("Invalid choice. Please enter a valid
option.");
            }
        } while (choice != 5);
        scanner.close();
    }
}
```

CourierMenu.java

```
package dao;

import entities.Courier;

import java.util.ArrayList;
import java.util.Scanner;

public class CourierMenu {

    public static void couriermenu() {

        Scanner scanner = new Scanner(System.in);
        CourierUserServiceImpl obj=new CourierUserServiceImpl();
        int choice;

        do {
            System.out.println("1. Place an order\n"
                        + "2. Get Order Status\n"
                        + "3. Cancel Order\n"
                        + "4. Get Assigned Orders\n"
                        + "5. Get Order History\n"
                        + "6. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    // Place Order
                  Courier courierObject = new Courier();
                  System.out.println("Enter  senderName: ");
                    String senderName = scanner.nextLine();
                    courierObject.setSenderName(senderName);
                    System.out.println("Enter  senderAddress: ");
                    String senderAddress = scanner.nextLine();
                    courierObject.setSenderAddress(senderAddress);
                    System.out.println("Enter receiverName : ");
                    String receiverName = scanner.nextLine();
                    courierObject.setReceiverName(receiverName);
                    System.out.println("Enter  receiverAddress: ");
                    String receiverAddress = scanner.nextLine();
                    courierObject.setReceiverAddress(receiverAddress);
                    System.out.println("Enter parcel weight : ");
                    double weight = scanner.nextDouble();
                    courierObject.setWeight(weight);
                    scanner.nextLine();
                    System.out.println("Enter userId : ");
                    int userId = scanner.nextInt();
                    courierObject.setUserId(userId);

                    System.out.println("Placing a new courier order...");
                    long trackingId=obj.placeOrder(courierObject);

                    System.out.println("Tracking Id Of your order is :
"+trackingId);
                    break;

                case 2:
                    // Get Order Status
                    System.out.println("Enter tracking number to get order
status:");
                    long trackingNumber=scanner.nextLong();
                    String Status=obj.getOrderStatus(trackingNumber);
                    System.out.println("--------------Courier Order Status-----
----------");
                    System.out.println("----------------------------------------
----------");
                    System.out.printf("| %-20s | %-23s|\n", "Field", "Value");
                    System.out.println("----------------------------------------
----------");
                    System.out.printf("| %-20s | %-23s|\n", "Tracking Number",
trackingNumber);
                    System.out.printf("| %-20s | %-23s|\n", "Status",Status) ;
                    System.out.println("----------------------------------------
----------");
                    break;

                case 3:
                    // Cancel Order
                    System.out.println("Enter tracking number to cancel the
order:");
                    trackingNumber=scanner.nextLong();

                    if (obj.cancelOrder(trackingNumber))
                        System.out.println("Order Cancelled!!");
                    break;

                case 4:

                  System.out.println("Enter Employee id : ");
                    int empid=scanner.nextInt();
                    ArrayList<Courier> al=obj.getAssignedOrders(empid);
                    if(al!=null) {
                        System.out.println("------------------------------------
----------------------------------Courier Details--------------------------------
-------------------------------------------------------------------");
```

```
                        System.out.println("-----------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------");
                        System.out.printf("| %-10s | %-15s | %-15s | %-15s | %-
20s | %-10s | %-15s | %-15s | %-12s | %-10s | %-10s |\n",
                                "courierID", "senderName", "senderAddress",
"receiverName", "receiverAddress", "weight", "status",
                                "trackingNumber", "deliveryDate", "userId",
"empid");
                        System.out.println("-----------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------");

                        for (Courier c : al) {
                            System.out.printf("| %-10d | %-15s | %-15s | %-15s
| %-20s | %-10.2f | %-15s | %-15s | %-12s | %-10d | %-10d |\n",
                                    c.getCourierID(), c.getSenderName(),
c.getSenderAddress(), c.getReceiverName(),
                                    c.getReceiverAddress(), c.getWeight(),
c.getStatus(),c.getTrackingNumber(),
                                    c.getDeliveryDate(), c.getUserId(),
c.getEmpId());
                        }

                        System.out.println("-----------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------");

                    }


                    break;

                case 5:

                    System.out.println("Enter you tracking number to get
courier history");
                    long tn=scanner.nextLong();
                    obj.getOrderHistory(tn);
                    break;

                case 6:
                  System.out.println("!!Thanks For Using Courier Management
System!!");
                  break;

                default:
                    System.out.println("Invalid choice. Please enter a valid
option.");
            }
        } while (choice != 6);

        scanner.close();
    }

}
```

CourierServiceDB.java

```
package dao;
import java.sql.Connection;
import util.DBConnection;

public class CourierServiceDb {

    // Static variable to hold the database connection
    public static Connection connection;

    // Constructor to initialize the database connection
    public CourierServiceDb() {
        // Invoke the method in DBConnection class to get the connection
        connection = DBConnection.getConnection();
    }


    // method to close the database connection
    public  void closeConnection() {
        try {
            if (connection != null && !connection.isClosed()) {
//                  System.out.println("Closing db connection");
                connection.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

<u>Package Exception</u>

InvalidEmployeeIdException.java

```
package exception;

@SuppressWarnings("serial")
public  class InvalidEmployeeIdException extends Exception {
    public  InvalidEmployeeIdException(int id) {
        super("No Employee Found with Id : "+id);
    }
}
```

TrackingNumberNotFoundException.java

```
package exception;

@SuppressWarnings("serial")
public class TrackingNumberNotFoundException extends Exception {
      public TrackingNumberNotFoundException(long No) {
        super("Invalid Tracking Number "+No);
    }
}
```

Package Util

DBconnection.java

```
package util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnection {
    private static Connection connection;

    private DBConnection() {
        // Private constructor to prevent instantiation
    }

    public static Connection getConnection() {
        if (connection == null) {
            try {
                // Load the JDBC driver
                Class.forName("com.mysql.cj.jdbc.Driver");
                //System.out.println("Driver Loaded");
                // Get the connection string from the property file
                String connectionString = PropertyUtil.getPropertyString();
                // Establish the connection
                connection = DriverManager.getConnection(connectionString);
                //System.out.println("Db connected Successfully!!");
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace(); // Handle the exception appropriately
            }
        }
        return connection;
    }
}
```

PropertyUtil.java

```
package util;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;

public class PropertyUtil {
    private PropertyUtil() {
        // Private constructor to prevent instantiation
    }

    public static String getPropertyString() {
        Properties properties = new Properties();
        String propertyFilePath = "C:\\Users\\atiwa\\eclipse-
workspace\\cc\\src\\util\\db.properties";
        try (FileInputStream input = new FileInputStream(propertyFilePath)) {
            properties.load(input);

            // Construct the connection string using properties
```

```java
            String host = properties.getProperty("hostname");
            String dbName = properties.getProperty("dbname");
            String username = properties.getProperty("username");
            String password = properties.getProperty("password");
            String port = properties.getProperty("port");
            String connStr="jdbc:mysql://" + host + ":" + port + "/" + dbName +
"?user=" + username + "&password=" + password;
            //System.out.println(connStr);
            return connStr;

        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + propertyFilePath);
        } catch (IOException e) {
            System.out.println("Error reading the property file");
            e.printStackTrace(); // Handle the exception appropriately
        }

        return null; // Return null if unable to read properties or construct
connection string
    }
}
```

db.properties

```
hostname=localhost
dbname=courier
username=root
password=1234
port=3306
```

Package Main

MainModule.java

```java
package main;
import java.util.*;
import dao.CourierMenu;
import dao.CourierServiceDb;
import dao.CourierStaff;
public class MainModule {
      public static void main(String args[])
      {
            Scanner sc=new Scanner(System.in);
            CourierServiceDb dbCon= new CourierServiceDb();
            System.out.println("................Are you user or
Admin................");
            String us=sc.next();
            if (us.equalsIgnoreCase("Admin"))
            {
                  System.out.println("................Welcome
Admin................");
                  CourierStaff staffobj=new CourierStaff();
                  staffobj.couriermenu();
            }
            else
            {
                  System.out.println("................Welcome
User................");
```

```
            CourierMenu.couriermenu();
        }
        dbCon.closeConnection();
        sc.close();

    }
}
```