ASSIGNMENT – 1

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

TASK 1:

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

-- Courier Table
CREATE TABLE Couriers (
    CourierID INT PRIMARY KEY AUTO_INCREMENT,
    SenderName VARCHAR(255),
    SenderAddress TEXT,
    ReceiverName VARCHAR(255),
    ReceiverAddress TEXT,
    Weight DECIMAL(5, 2),
    Status VARCHAR(50),
    TrackingNumber VARCHAR(20) UNIQUE,
    DeliveryDate DATE
);

-- Employee Table
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(255),
    Email VARCHAR(255) UNIQUE,
    ContactNumber VARCHAR(20),
    Role VARCHAR(50),
```

```
    Salary DECIMAL(10, 2)
);

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

-- Payment Table
CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY AUTO_INCREMENT,
    CourierID INT,
    LocationID INT,
    Amount DECIMAL(10, 2),
    PaymentDate DATE,
    FOREIGN KEY (CourierID) REFERENCES Couriers(CourierID),
    FOREIGN KEY (LocationID) REFERENCES Locations(LocationID)
);
```

Task 2: Select,Where

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

1. List all customers:

```
mysql> select * from users;
+--------+------------------+---------------------+-------------+---------------+-----------------+
| UserID | Name             | Email               | Password    | ContactNumber | Address         |
+--------+------------------+---------------------+-------------+---------------+-----------------+
|      1 | John Doe         | john@example.com    | password123 | 123-456-7890  | 123 Main St     |
|      2 | Jane Smith       | jane@example.com    | password456 | 987-654-3210  | 456 Elm St      |
|      3 | Emily Brown      | emily@example.com   | password789 | 111-222-3333  | 789 Oak St      |
|      4 | Michael Johnson  | michael@example.com | passwordABC | 444-555-6666  | 456 Maple St    |
|      5 | Sophia Wilson    | sophia@example.com  | passwordDEF | 777-888-9999  | 101 Pine St     |
|      6 | Oliver Taylor    | oliver@example.com  | passwordGHI | 123-456-7890  | 321 Cedar St    |
|      7 | Emma Davis       | emma@example.com    | passwordJKL | 333-444-5555  | 222 Walnut St   |
|      8 | William Martinez | william@example.com | passwordMNO | 666-777-8888  | 555 Chestnut St |
|      9 | Isabella Anderson| isabella@example.com| passwordPQR | 999-000-1111  | 888 Oak St      |
|     10 | Sophie Thompson  | sophie@example.com  | passwordSTU | 222-333-4444  | 777 Maple St    |
+--------+------------------+---------------------+-------------+---------------+-----------------+
```

2. List all orders for a specific customer:

```
mysql> select * from  couriers where sendername= 'sender A';
+-----------+------------+---------------+-------------+-----------------+--------+------------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status     | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+-------------+-----------------+--------+------------+----------------+--------------+
|         1 | Sender A   | 123 Sender St | Receiver X   | 456 Receiver St |   3.50 | In Transit | TRACK123       | 2024-04-12   |
+-----------+------------+---------------+-------------+-----------------+--------+------------+----------------+--------------+
1 row in set (0.00 sec)
```

3. List all couriers:

```
mysql> select * from couriers;
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status    | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
|         1 | Sender A   | 123 Sender St | Receiver X   | 456 Receiver St |   3.50 | In Transit | TRACK123       | 2024-04-12   |
|         2 | Sender B   | 789 Sender St | Receiver Y   | 101 Receiver St |   2.20 | Delivered  | TRACK456       | 2024-04-10   |
|         3 | Sender C   | 321 Sender St | Receiver Z   | 789 Receiver St |   4.00 | Pending    | TRACK789       | 2024-04-15   |
|         4 | Sender D   | 555 Sender St | Receiver W   | 888 Receiver St |   1.80 | In Transit | TRACKXYZ1      | 2024-04-11   |
|         5 | Sender E   | 999 Sender St | Receiver V   | 222 Receiver St |   3.70 | Delivered  | TRACKLMN       | 2024-04-09   |
|         6 | Sender F   | 111 Sender St | Receiver U   | 333 Receiver St |   2.50 | Pending    | TRACKOPQ       | 2024-04-14   |
|         7 | Sender G   | 777 Sender St | Receiver T   | 444 Receiver St |   5.20 | In Transit | TRACKRST       | 2024-04-13   |
|         8 | Sender H   | 888 Sender St | Receiver S   | 555 Receiver St |   1.90 | Delivered  | TRACKUVW       | 2024-04-08   |
|         9 | Sender I   | 222 Sender St | Receiver R   | 666 Receiver St |   3.30 | Pending    | TRACKXYZ2      | 2024-04-16   |
|        10 | Sender J   | 444 Sender St | Receiver Q   | 777 Receiver St |   2.10 | In Transit | TRACK1234      | 2024-04-17   |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
10 rows in set (0.00 sec)
```

4. List all packages for a specific order

```
mysql> select * from couriers where sendername = 'sender d';
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status    | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
|         4 | Sender D   | 555 Sender St | Receiver W   | 888 Receiver St |   1.80 | In Transit | TRACKXYZ1      | 2024-04-11   |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
1 row in set (0.00 sec)
```

5. List all deliveries for a specific courier:

```
mysql> select * from couriers where courierid=5;
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status    | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
|         5 | Sender E   | 999 Sender St | Receiver V   | 222 Receiver St |   3.70 | Delivered | TRACKLMN       | 2024-04-09   |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
1 row in set (0.00 sec)
```

6. List all undelivered packages:

```
mysql> select * from couriers where status <> 'delivered';
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status    | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
|         1 | Sender A   | 123 Sender St | Receiver X   | 456 Receiver St |   3.50 | In Transit | TRACK123       | 2024-04-12   |
|         3 | Sender C   | 321 Sender St | Receiver Z   | 789 Receiver St |   4.00 | Pending    | TRACK789       | 2024-04-15   |
|         4 | Sender D   | 555 Sender St | Receiver W   | 888 Receiver St |   1.80 | In Transit | TRACKXYZ1      | 2024-04-11   |
|         6 | Sender F   | 111 Sender St | Receiver U   | 333 Receiver St |   2.50 | Pending    | TRACKOPQ       | 2024-04-14   |
|         7 | Sender G   | 777 Sender St | Receiver T   | 444 Receiver St |   5.20 | In Transit | TRACKRST       | 2024-04-13   |
|         9 | Sender I   | 222 Sender St | Receiver R   | 666 Receiver St |   3.30 | Pending    | TRACKXYZ2      | 2024-04-16   |
|        10 | Sender J   | 444 Sender St | Receiver Q   | 777 Receiver St |   2.10 | In Transit | TRACK1234      | 2024-04-17   |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
7 rows in set (0.00 sec)
```

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

```
mysql> SELECT * FROM couriers WHERE DeliveryDate = CURDATE();
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status    | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
|         7 | Sender G   | 777 Sender St | Receiver T   | 444 Receiver St |   5.20 | In Transit | TRACKRST       | 2024-04-13   |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
1 row in set (0.00 sec)
```

8. List all packages with a specific status:

```
mysql> select * from couriers where status= 'in transit';
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status    | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
|         1 | Sender A   | 123 Sender St | Receiver X   | 456 Receiver St |   3.50 | In Transit | TRACK123       | 2024-04-12   |
|         4 | Sender D   | 555 Sender St | Receiver W   | 888 Receiver St |   1.80 | In Transit | TRACKXYZ1      | 2024-04-11   |
|         7 | Sender G   | 777 Sender St | Receiver T   | 444 Receiver St |   5.20 | In Transit | TRACKRST       | 2024-04-13   |
|        10 | Sender J   | 444 Sender St | Receiver Q   | 777 Receiver St |   2.10 | In Transit | TRACK1234      | 2024-04-17   |
+-----------+------------+---------------+-------------+-----------------+--------+-----------+----------------+--------------+
```

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

```
mysql> SELECT
    ->     CourierID,
    ->     COUNT(*) AS TotalPackages
    -> FROM
    ->     Couriers
    -> GROUP BY
    ->     CourierID;
+-----------+-----------------+
| CourierID | TotalPackages |
+-----------+-----------------+
|         1 |               1 |
|         2 |               1 |
|         3 |               1 |
|         4 |               1 |
|         5 |               1 |
|         6 |               1 |
|         7 |               1 |
|         8 |               1 |
|         9 |               1 |
|        10 |               1 |
+-----------+-----------------+
```

10. Find the average delivery time for each courier

```
mysql> Select p.courierId, avg(datediff(c.deliveryDate,p.paymentDate)) as avgTime from payments p inner join couriers c on c.courierId =p.courierId grou
p by courierId;
+-----------+---------+
| courierId | avgTime |
+-----------+---------+
|         1 |  0.0000 |
|         2 |  0.0000 |
|         3 |  0.0000 |
|         4 |  0.0000 |
|         5 |  0.0000 |
|         6 |  6.0000 |
|         7 |  6.0000 |
|         8 |  2.0000 |
|         9 | 11.0000 |
|        10 | 13.0000 |
+-----------+---------+
```

11. List all packages with a specific weight range:

```
mysql> select * from couriers where weight between 2.0 AND 3.0;
+-----------+------------+---------------+--------------+-----------------+--------+------------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status     | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+--------------+-----------------+--------+------------+----------------+--------------+
|         2 | Sender B   | 789 Sender St | Receiver Y   | 101 Receiver St |   2.20 | Delivered  | TRACK456       | 2024-04-10   |
|         6 | Sender F   | 111 Sender St | Receiver U   | 333 Receiver St |   2.50 | Pending    | TRACKOPQ       | 2024-04-14   |
|        10 | Sender J   | 444 Sender St | Receiver Q   | 777 Receiver St |   2.10 | In Transit | TRACK1234      | 2024-04-17   |
+-----------+------------+---------------+--------------+-----------------+--------+------------+----------------+--------------+
3 rows in set (0.00 sec)
```

12. Retrieve employees whose names contain 'John'

```
mysql> select * from employees where name like '%John';
Empty set (0.00 sec)
```

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

```
mysql> SELECT couriers.*
    -> FROM couriers
    -> JOIN payments ON couriers.courierid = payments.courierid
    -> WHERE payments.amount > 50.00;
Empty set (0.00 sec)
```

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

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

```
mysql> SELECT e.employeeid, COUNT(c.Courierid) AS Totalcouriershandled
    -> FROM employees e
    -> LEFT JOIN couriers c ON e.employeeid = c.courierid
    -> GROUP BY e.employeeid;
+------------+----------------------+
| employeeid | Totalcouriershandled |
+------------+----------------------+
|          1 |                    1 |
|          2 |                    1 |
|          3 |                    1 |
|          4 |                    1 |
|          5 |                    1 |
|          6 |                    1 |
|          7 |                    1 |
|          8 |                    1 |
|          9 |                    1 |
|         10 |                    1 |
+------------+----------------------+
```

15. Calculate the total revenue generated by each location

```
mysql> select locationid, sum(amount) as totalrevenue from payments group by
 locationid;
+------------+--------------+
| locationid | totalrevenue |
+------------+--------------+
|          1 |        10.00 |
|          2 |        20.00 |
|          3 |        30.00 |
|          4 |        15.00 |
|          5 |        10.00 |
|          6 |         5.00 |
|          7 |        25.00 |
|          8 |        12.00 |
|          9 |        18.00 |
|         10 |         8.00 |
+------------+--------------+
```

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

```
mysql> select LocationId,count(CourierId) from payments group by LocationId;
+------------+--------------------+
| LocationId | count(CourierId)   |
+------------+--------------------+
|          1 |                  1 |
|          2 |                  1 |
|          3 |                  1 |
|          4 |                  1 |
|          5 |                  1 |
|          6 |                  1 |
|          7 |                  1 |
|          8 |                  1 |
|          9 |                  1 |
|         10 |                  1 |
+------------+--------------------+
```

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

```
mysql> select p.courierid, avg(datediff(c.deliverydate, p.paymentdate)) as a
vgdeliverytime from payments p
    -> inner join couriers c on c.courierid = p.courierid group by courierid
 order by
    -> avgdeliverytime desc limit 1;
+-----------+-----------------+
| courierid | avgdeliverytime |
+-----------+-----------------+
|        10 |         13.0000 |
+-----------+-----------------+
1 row in set (0.01 sec)
```

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

```
mysql> select locationid, sum(amount) as Totalrevenue from payments group by
 locationid having totalrevenue < 25;\
+------------+--------------+
| locationid | Totalrevenue |
+------------+--------------+
|          1 |        10.00 |
|          2 |        20.00 |
|          4 |        15.00 |
|          5 |        10.00 |
|          6 |         5.00 |
|          8 |        12.00 |
|          9 |        18.00 |
|         10 |         8.00 |
+------------+--------------+
```

19. Calculate Total Payments per Location

```
mysql> select locationid, sum(amount) as totalpayments from payments group b
y locationid;
+------------+---------------+
| locationid | totalpayments |
+------------+---------------+
|          1 |         10.00 |
|          2 |         20.00 |
|          3 |         30.00 |
|          4 |         15.00 |
|          5 |         10.00 |
|          6 |          5.00 |
|          7 |         25.00 |
|          8 |         12.00 |
|          9 |         18.00 |
|         10 |          8.00 |
+------------+---------------+
```

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

(LocationID = X):

```
mysql> select courierid from payments where locationid=2 group by courierid
having sum(amount) > 10;
+-----------+
| courierid |
+-----------+
|         2 |
+-----------+
```

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

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

```
mysql> select courierid from payment where paymentdate> '2024-04-08' group b
y courierid having sum(amount) >25;
ERROR 1146 (42S02): Table 'cms.payment' doesn't exist
mysql> select courierid from payments where paymentdate> '2024-04-08' group
by courierid having sum(amount) >25;
+-----------+
| courierid |
+-----------+
|         3 |
+-----------+
```

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

(PaymentDate > 'YYYY-MM-DD')

```
mysql> select locationid from payments where paymentdate> '2024-04-08' group
 by locationid having sum(amount) >5000;
Empty set (0.00 sec)
```

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

23. Retrieve Payments with Courier Information

```
mysql> SELECT payments.*, couriers.*
    -> FROM payments
    -> JOIN couriers ON payments.courierid = couriers.courierid;
```

PaymentID	CourierID	LocationID	Amount	PaymentDate	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
1	1	1	10.00	2024-04-12	1	Sender A	123 Sender St	Receiver X	456 Receiver St	3.50	In Transit	TRACK123	2024-04-12
2	2	2	20.00	2024-04-10	2	Sender B	789 Sender St	Receiver Y	101 Receiver St	2.20	Delivered	TRACK456	2024-04-10
3	3	3	30.00	2024-04-15	3	Sender C	321 Sender St	Receiver Z	789 Receiver St	4.00	Pending	TRACK789	2024-04-15
4	4	4	15.00	2024-04-11	4	Sender D	555 Sender St	Receiver W	888 Receiver St	1.80	In Transit	TRACKXYZ1	2024-04-11
5	5	5	10.00	2024-04-09	5	Sender E	999 Sender St	Receiver V	222 Receiver St	3.70	Delivered	TRACKLMN	2024-04-09
6	6	6	5.00	2024-04-08	6	Sender F	111 Sender St	Receiver U	333 Receiver St	2.50	Pending	TRACKOPQ	2024-04-14
7	7	7	25.00	2024-04-07	7	Sender G	777 Sender St	Receiver T	444 Receiver St	5.20	In Transit	TRACKRST	2024-04-13
8	8	8	12.00	2024-04-06	8	Sender H	888 Sender St	Receiver S	555 Receiver St	1.90	Delivered	TRACKUVW	2024-04-08
9	9	9	18.00	2024-04-05	9	Sender I	222 Sender St	Receiver R	666 Receiver St	3.30	Pending	TRACKXYZ2	2024-04-16
10	10	10	8.00	2024-04-04	10	Sender J	444 Sender St	Receiver Q	777 Receiver St	2.10	In Transit	TRACK1234	2024-04-17

24. Retrieve Payments with Location Information

```
mysql> select payments.*, locations.* from payments join locations on paymen
ts.locationid = locations.locationid;
```

PaymentID	CourierID	LocationID	Amount	PaymentDate	LocationID	LocationName	Address
1	1	1	10.00	2024-04-12	1	Warehouse A	789 Warehouse Ave
2	2	2	20.00	2024-04-10	2	Warehouse B	101 Warehouse St
3	3	3	30.00	2024-04-15	3	Warehouse C	555 Warehouse St
4	4	4	15.00	2024-04-11	4	Warehouse D	777 Warehouse Ave
5	5	5	10.00	2024-04-09	5	Office Headquarters	123 Office St
6	6	6	5.00	2024-04-08	6	Branch Office 1	456 Branch St
7	7	7	25.00	2024-04-07	7	Branch Office 2	789 Branch St
8	8	8	12.00	2024-04-06	8	Branch Office 3	101 Branch St
9	9	9	18.00	2024-04-05	9	Retail Store	555 Retail St
10	10	10	8.00	2024-04-04	10	Distribution Center	888 Distribution Ave

25. Retrieve Payments with Courier and Location Information

```
mysql> select payments.*, couriers.*, locations.* from payments join couriers on payments.courierid=couriers.courierid join locations on payments.locati
onid = locations.locationid;
```

PaymentID	CourierID	LocationID	Amount	PaymentDate	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	LocationID	LocationName	Address
1	1	1	10.00	2024-04-12	1	Sender A	123 Sender St	Receiver X	456 Receiver St	3.50	In Transit	TRACK123	2024-04-12	1	Warehouse A	789 Warehouse Ave
2	2	2	20.00	2024-04-10	2	Sender B	789 Sender St	Receiver Y	101 Receiver St	2.20	Delivered	TRACK456	2024-04-10	2	Warehouse B	101 Warehouse St
3	3	3	30.00	2024-04-15	3	Sender C	321 Sender St	Receiver Z	789 Receiver St	4.00	Pending	TRACK789	2024-04-15	3	Warehouse C	555 Warehouse St
4	4	4	15.00	2024-04-11	4	Sender D	555 Sender St	Receiver W	888 Receiver St	1.80	In Transit	TRACKXYZ1	2024-04-11	4	Warehouse D	777 Warehouse Ave
5	5	5	10.00	2024-04-09	5	Sender E	999 Sender St	Receiver V	222 Receiver St	3.70	Delivered	TRACKLMN	2024-04-09	5	Office Headquarters	123 Office St
6	6	6	5.00	2024-04-08	6	Sender F	111 Sender St	Receiver U	333 Receiver St	2.50	Pending	TRACKOPQ	2024-04-14	6	Branch Office 1	456 Branch St
7	7	7	25.00	2024-04-07	7	Sender G	777 Sender St	Receiver T	444 Receiver St	5.20	In Transit	TRACKRST	2024-04-13	7	Branch Office 2	789 Branch St
8	8	8	12.00	2024-04-06	8	Sender H	888 Sender St	Receiver S	555 Receiver St	1.90	Delivered	TRACKUVW	2024-04-08	8	Branch Office 3	101 Branch St
9	9	9	18.00	2024-04-05	9	Sender I	222 Sender St	Receiver R	666 Receiver St	3.30	Pending	TRACKXYZ2	2024-04-16	9	Retail Store	555 Retail St
10	10	10	8.00	2024-04-04	10	Sender J	444 Sender St	Receiver Q	777 Receiver St	2.10	In Transit	TRACK1234	2024-04-17	10	Distribution Center	888 Distribution Ave

10 rows in set (0.01 sec)

26. List all payments with courier details

```
mysql> select * from payments p left join couriers c on p.courierid=c.courie
rid;
```

PaymentID	CourierID	LocationID	Amount	PaymentDate	CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
1	1	1	10.00	2024-04-12	1	Sender A	123 Sender St	Receiver X	456 Receiver St	3.50	In Transit	TRACK123	2024-04-12
2	2	2	20.00	2024-04-10	2	Sender B	789 Sender St	Receiver Y	101 Receiver St	2.20	Delivered	TRACK456	2024-04-10
3	3	3	30.00	2024-04-15	3	Sender C	321 Sender St	Receiver Z	789 Receiver St	4.00	Pending	TRACK789	2024-04-15
4	4	4	15.00	2024-04-11	4	Sender D	555 Sender St	Receiver W	888 Receiver St	1.80	In Transit	TRACKXYZ1	2024-04-11
5	5	5	10.00	2024-04-09	5	Sender E	999 Sender St	Receiver V	222 Receiver St	3.70	Delivered	TRACKLMN	2024-04-09
6	6	6	5.00	2024-04-08	6	Sender F	111 Sender St	Receiver U	333 Receiver St	2.50	Pending	TRACKOPQ	2024-04-14
7	7	7	25.00	2024-04-07	7	Sender G	777 Sender St	Receiver T	444 Receiver St	5.20	In Transit	TRACKRST	2024-04-13
8	8	8	12.00	2024-04-06	8	Sender H	888 Sender St	Receiver S	555 Receiver St	1.90	Delivered	TRACKUVW	2024-04-08
9	9	9	18.00	2024-04-05	9	Sender I	222 Sender St	Receiver R	666 Receiver St	3.30	Pending	TRACKXYZ2	2024-04-16
10	10	10	8.00	2024-04-04	10	Sender J	444 Sender St	Receiver Q	777 Receiver St	2.10	In Transit	TRACK1234	2024-04-17

27. Total payments received for each courier

```
mysql> select courierid, sum(amount) as totalpayments from payments group by
 courierid;
+-----------+---------------+
| courierid | totalpayments |
+-----------+---------------+
|         1 |         10.00 |
|         2 |         20.00 |
|         3 |         30.00 |
|         4 |         15.00 |
|         5 |         10.00 |
|         6 |          5.00 |
|         7 |         25.00 |
|         8 |         12.00 |
|         9 |         18.00 |
|        10 |          8.00 |
+-----------+---------------+
```

28. List payments made on a specific date

```
mysql> select * from payments where paymentdate='2024-04-06';
+-----------+-----------+------------+--------+-------------+
| PaymentID | CourierID | LocationID | Amount | PaymentDate |
+-----------+-----------+------------+--------+-------------+
|         8 |         8 |          8 |  12.00 | 2024-04-06  |
+-----------+-----------+------------+--------+-------------+
```

29. Get Courier Information for Each Payment

```
mysql> select * from couriers c where c.courierid in (select courierid from
payments);
+-----------+------------+---------------+--------------+-----------------+--------+-----------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status    | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+--------------+-----------------+--------+-----------+----------------+--------------+
|         1 | Sender A   | 123 Sender St | Receiver X   | 456 Receiver St |   3.50 | In Transit | TRACK123       | 2024-04-12   |
|         2 | Sender B   | 789 Sender St | Receiver Y   | 101 Receiver St |   2.20 | Delivered  | TRACK456       | 2024-04-10   |
|         3 | Sender C   | 321 Sender St | Receiver Z   | 789 Receiver St |   4.00 | Pending    | TRACK789       | 2024-04-15   |
|         4 | Sender D   | 555 Sender St | Receiver W   | 888 Receiver St |   1.80 | In Transit | TRACKXYZ1      | 2024-04-11   |
|         5 | Sender E   | 999 Sender St | Receiver V   | 222 Receiver St |   3.70 | Delivered  | TRACKLMN       | 2024-04-09   |
|         6 | Sender F   | 111 Sender St | Receiver U   | 333 Receiver St |   2.50 | Pending    | TRACKOPQ       | 2024-04-14   |
|         7 | Sender G   | 777 Sender St | Receiver T   | 444 Receiver St |   5.20 | In Transit | TRACKRST       | 2024-04-13   |
|         8 | Sender H   | 888 Sender St | Receiver S   | 555 Receiver St |   1.90 | Delivered  | TRACKUVW       | 2024-04-08   |
|         9 | Sender I   | 222 Sender St | Receiver R   | 666 Receiver St |   3.30 | Pending    | TRACKXYZ2      | 2024-04-16   |
|        10 | Sender J   | 444 Sender St | Receiver Q   | 777 Receiver St |   2.10 | In Transit | TRACK1234      | 2024-04-17   |
+-----------+------------+---------------+--------------+-----------------+--------+-----------+----------------+--------------+
```

30. Get Payment Details with Location

```
mysql> select payments.*, locations.* from payments left join locations on p
ayments.locationid = locations.locationid;
+-----------+-----------+------------+--------+-------------+------------+--------------------+----------------------+
| PaymentID | CourierID | LocationID | Amount | PaymentDate | LocationID | LocationName       | Address              |
+-----------+-----------+------------+--------+-------------+------------+--------------------+----------------------+
|         1 |         1 |          1 |  10.00 | 2024-04-12  |          1 | Warehouse A        | 789 Warehouse Ave    |
|         2 |         2 |          2 |  20.00 | 2024-04-10  |          2 | Warehouse B        | 101 Warehouse St     |
|         3 |         3 |          3 |  30.00 | 2024-04-15  |          3 | Warehouse C        | 555 Warehouse St     |
|         4 |         4 |          4 |  15.00 | 2024-04-11  |          4 | Warehouse D        | 777 Warehouse Ave    |
|         5 |         5 |          5 |  10.00 | 2024-04-09  |          5 | Office Headquarters | 123 Office St        |
|         6 |         6 |          6 |   5.00 | 2024-04-08  |          6 | Branch Office 1    | 456 Branch St        |
|         7 |         7 |          7 |  25.00 | 2024-04-07  |          7 | Branch Office 2    | 789 Branch St        |
|         8 |         8 |          8 |  12.00 | 2024-04-06  |          8 | Branch Office 3    | 101 Branch St        |
|         9 |         9 |          9 |  18.00 | 2024-04-05  |          9 | Retail Store       | 555 Retail St        |
|        10 |        10 |         10 |   8.00 | 2024-04-04  |         10 | Distribution Center | 888 Distribution Ave |
+-----------+-----------+------------+--------+-------------+------------+--------------------+----------------------+
```

31. Calculating Total Payments for Each Courier

```
mysql> select courierid, sum(amount) as totalpayments from payments group by
 courierid;
+-----------+---------------+
| courierid | totalpayments |
+-----------+---------------+
|         1 |         10.00 |
|         2 |         20.00 |
|         3 |         30.00 |
|         4 |         15.00 |
|         5 |         10.00 |
|         6 |          5.00 |
|         7 |         25.00 |
|         8 |         12.00 |
|         9 |         18.00 |
|        10 |          8.00 |
+-----------+---------------+
```

32. List Payments Within a Date Range

```
mysql> select * from payments where paymentdate between '2024-03-31' and '20
24-04-6';
+-----------+------------+-------------+--------+-------------+
| PaymentID | CourierID | LocationID | Amount | PaymentDate |
+-----------+------------+-------------+--------+-------------+
|         8 |          8 |           8 |  12.00 | 2024-04-06  |
|         9 |          9 |           9 |  18.00 | 2024-04-05  |
|        10 |         10 |          10 |   8.00 | 2024-04-04  |
+-----------+------------+-------------+--------+-------------+
```

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

no matches on either side

```
    -> u.name = c.senderName;^C
mysql> select u.*,c.courierId,c.status,c.TrackingNumber,c.DeliveryDate from users u left join couriers c on
    -> u.name = c.senderName;
+--------+------------------+---------------------+-----------+---------------+----------------+-----------+--------+----------------+--------------+
| UserID | Name             | Email               | Password  | ContactNumber | Address        | courierId | status | TrackingNumber | DeliveryDate |
+--------+------------------+---------------------+-----------+---------------+----------------+-----------+--------+----------------+--------------+
|      1 | John Doe         | john@example.com    | password123 | 123-456-7890  | 123 Main St    |      NULL | NULL   | NULL           | NULL         |
|      2 | Jane Smith       | jane@example.com    | password456 | 987-654-3210  | 456 Elm St     |      NULL | NULL   | NULL           | NULL         |
|      3 | Emily Brown      | emily@example.com   | password789 | 111-222-3333  | 789 Oak St     |      NULL | NULL   | NULL           | NULL         |
|      4 | Michael Johnson  | michael@example.com | passwordABC | 444-555-6666  | 456 Maple St   |      NULL | NULL   | NULL           | NULL         |
|      5 | Sophia Wilson    | sophia@example.com  | passwordDEF | 777-888-9999  | 101 Pine St    |      NULL | NULL   | NULL           | NULL         |
|      6 | Oliver Taylor    | oliver@example.com  | passwordGHI | 123-456-7890  | 321 Cedar St   |      NULL | NULL   | NULL           | NULL         |
|      7 | Emma Davis       | emma@example.com    | passwordJKL | 333-444-5555  | 222 Walnut St  |      NULL | NULL   | NULL           | NULL         |
|      8 | William Martinez | william@example.com | passwordMNO | 666-777-8888  | 555 Chestnut St |      NULL | NULL   | NULL           | NULL         |
|      9 | Isabella Anderson | isabella@example.com | passwordPQR | 999-000-1111  | 888 Oak St     |      NULL | NULL   | NULL           | NULL         |
|     10 | Sophie Thompson  | sophie@example.com  | passwordSTU | 222-333-4444  | 777 Maple St   |      NULL | NULL   | NULL           | NULL         |
+--------+------------------+---------------------+-----------+---------------+----------------+-----------+--------+----------------+--------------+
```

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

no matches on either side

```
mysql> select * from users cross join courierservices;
+--------+------------------+---------------------+-----------+---------------+----------------+-----------+-----------------+-------+
| UserID | Name             | Email               | Password  | ContactNumber | Address        | ServiceID | ServiceName     | Cost  |
+--------+------------------+---------------------+-----------+---------------+----------------+-----------+-----------------+-------+
|     10 | Sophie Thompson  | sophie@example.com  | passwordSTU | 222-333-4444  | 777 Maple St   |         1 | Standard Delivery |  10.00 |
|      9 | Isabella Anderson | isabella@example.com | passwordPQR | 999-000-1111  | 888 Oak St     |         1 | Standard Delivery |  10.00 |
|      8 | William Martinez | william@example.com | passwordMNO | 666-777-8888  | 555 Chestnut St |         1 | Standard Delivery |  10.00 |
|      7 | Emma Davis       | emma@example.com    | passwordJKL | 333-444-5555  | 222 Walnut St  |         1 | Standard Delivery |  10.00 |
|      6 | Oliver Taylor    | oliver@example.com  | passwordGHI | 123-456-7890  | 321 Cedar St   |         1 | Standard Delivery |  10.00 |
|      5 | Sophia Wilson    | sophia@example.com  | passwordDEF | 777-888-9999  | 101 Pine St    |         1 | Standard Delivery |  10.00 |
|      4 | Michael Johnson  | michael@example.com | passwordABC | 444-555-6666  | 456 Maple St   |         1 | Standard Delivery |  10.00 |
|      3 | Emily Brown      | emily@example.com   | password789 | 111-222-3333  | 789 Oak St     |         1 | Standard Delivery |  10.00 |
|      2 | Jane Smith       | jane@example.com    | password456 | 987-654-3210  | 456 Elm St     |         1 | Standard Delivery |  10.00 |
|      1 | John Doe         | john@example.com    | password123 | 123-456-7890  | 123 Main St    |         1 | Standard Delivery |  10.00 |
|     10 | Sophie Thompson  | sophie@example.com  | passwordSTU | 222-333-4444  | 777 Maple St   |         2 | Express Delivery |  20.00 |
|      9 | Isabella Anderson | isabella@example.com | passwordPQR | 999-000-1111  | 888 Oak St     |         2 | Express Delivery |  20.00 |
|      8 | William Martinez | william@example.com | passwordMNO | 666-777-8888  | 555 Chestnut St |         2 | Express Delivery |  20.00 |
|      7 | Emma Davis       | emma@example.com    | passwordJKL | 333-444-5555  | 222 Walnut St  |         2 | Express Delivery |  20.00 |
|      6 | Oliver Taylor    | oliver@example.com  | passwordGHI | 123-456-7890  | 321 Cedar St   |         2 | Express Delivery |  20.00 |
|      5 | Sophia Wilson    | sophia@example.com  | passwordDEF | 777-888-9999  | 101 Pine St    |         2 | Express Delivery |  20.00 |
|      4 | Michael Johnson  | michael@example.com | passwordABC | 444-555-6666  | 456 Maple St   |         2 | Express Delivery |  20.00 |
|      3 | Emily Brown      | emily@example.com   | password789 | 111-222-3333  | 789 Oak St     |         2 | Express Delivery |  20.00 |
|      2 | Jane Smith       | jane@example.com    | password456 | 987-654-3210  | 456 Elm St     |         2 | Express Delivery |  20.00 |
|      1 | John Doe         | john@example.com    | password123 | 123-456-7890  | 123 Main St    |         2 | Express Delivery |  20.00 |
|     10 | Sophie Thompson  | sophie@example.com  | passwordSTU | 222-333-4444  | 777 Maple St   |         3 | Same Day Delivery |  30.00 |
|      9 | Isabella Anderson | isabella@example.com | passwordPQR | 999-000-1111  | 888 Oak St     |         3 | Same Day Delivery |  30.00 |
|      8 | William Martinez | william@example.com | passwordMNO | 666-777-8888  | 555 Chestnut St |         3 | Same Day Delivery |  30.00 |
|      7 | Emma Davis       | emma@example.com    | passwordJKL | 333-444-5555  | 222 Walnut St  |         3 | Same Day Delivery |  30.00 |
|      6 | Oliver Taylor    | oliver@example.com  | passwordGHI | 123-456-7890  | 321 Cedar St   |         3 | Same Day Delivery |  30.00 |
|      5 | Sophia Wilson    | sophia@example.com  | passwordDEF | 777-888-9999  | 101 Pine St    |         3 | Same Day Delivery |  30.00 |
```

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

```
mysql> select * from employees cross join locations;
```

EmployeeID	Name	Email	ContactNumber	Role	Salary	LocationID	LocationName	Address
10	Jack Evans	jack@example.com	777-999-1111	Security Officer	30000.00	1	Warehouse A	789 Warehouse Ave
9	Ivy Adams	ivy@example.com	111-333-5555	Logistics Coordinator	42000.00	1	Warehouse A	789 Warehouse Ave
8	Henry Thomas	henry@example.com	555-777-9999	Supervisor	48000.00	1	Warehouse A	789 Warehouse Ave
7	Grace Martinez	grace@example.com	987-654-3210	Driver	32000.00	1	Warehouse A	789 Warehouse Ave
6	Frank Wilson	frank@example.com	321-654-9876	Warehouse Manager	55000.00	1	Warehouse A	789 Warehouse Ave
5	Ella Taylor	ella@example.com	123-456-7890	Customer Service	35000.00	1	Warehouse A	789 Warehouse Ave
4	David Lee	david@example.com	999-000-1111	Accountant	45000.00	1	Warehouse A	789 Warehouse Ave
3	Charlie Brown	charlie@example.com	777-888-9999	Dispatcher	40000.00	1	Warehouse A	789 Warehouse Ave
2	Bob Smith	bob@example.com	444-555-6666	Courier	30000.00	1	Warehouse A	789 Warehouse Ave
1	Alice Johnson	alice@example.com	111-222-3333	Manager	50000.00	1	Warehouse A	789 Warehouse Ave
10	Jack Evans	jack@example.com	777-999-1111	Security Officer	30000.00	2	Warehouse B	101 Warehouse St
9	Ivy Adams	ivy@example.com	111-333-5555	Logistics Coordinator	42000.00	2	Warehouse B	101 Warehouse St
8	Henry Thomas	henry@example.com	555-777-9999	Supervisor	48000.00	2	Warehouse B	101 Warehouse St
7	Grace Martinez	grace@example.com	987-654-3210	Driver	32000.00	2	Warehouse B	101 Warehouse St
6	Frank Wilson	frank@example.com	321-654-9876	Warehouse Manager	55000.00	2	Warehouse B	101 Warehouse St
5	Ella Taylor	ella@example.com	123-456-7890	Customer Service	35000.00	2	Warehouse B	101 Warehouse St
4	David Lee	david@example.com	999-000-1111	Accountant	45000.00	2	Warehouse B	101 Warehouse St
3	Charlie Brown	charlie@example.com	777-888-9999	Dispatcher	40000.00	2	Warehouse B	101 Warehouse St
2	Bob Smith	bob@example.com	444-555-6666	Courier	30000.00	2	Warehouse B	101 Warehouse St
1	Alice Johnson	alice@example.com	111-222-3333	Manager	50000.00	2	Warehouse B	101 Warehouse St
10	Jack Evans	jack@example.com	777-999-1111	Security Officer	30000.00	3	Warehouse C	555 Warehouse St
9	Ivy Adams	ivy@example.com	111-333-5555	Logistics Coordinator	42000.00	3	Warehouse C	555 Warehouse St
8	Henry Thomas	henry@example.com	555-777-9999	Supervisor	48000.00	3	Warehouse C	555 Warehouse St
7	Grace Martinez	grace@example.com	987-654-3210	Driver	32000.00	3	Warehouse C	555 Warehouse St
6	Frank Wilson	frank@example.com	321-654-9876	Warehouse Manager	55000.00	3	Warehouse C	555 Warehouse St
5	Ella Taylor	ella@example.com	123-456-7890	Customer Service	35000.00	3	Warehouse C	555 Warehouse St
4	David Lee	david@example.com	999-000-1111	Accountant	45000.00	3	Warehouse C	555 Warehouse St
3	Charlie Brown	charlie@example.com	777-888-9999	Dispatcher	40000.00	3	Warehouse C	555 Warehouse St
2	Bob Smith	bob@example.com	444-555-6666	Courier	30000.00	3	Warehouse C	555 Warehouse St
1	Alice Johnson	alice@example.com	111-222-3333	Manager	50000.00	3	Warehouse C	555 Warehouse St
10	Jack Evans	jack@example.com	777-999-1111	Security Officer	30000.00	4	Warehouse D	777 Warehouse Ave
9	Ivy Adams	ivy@example.com	111-333-5555	Logistics Coordinator	42000.00	4	Warehouse D	777 Warehouse Ave
8	Henry Thomas	henry@example.com	555-777-9999	Supervisor	48000.00	4	Warehouse D	777 Warehouse Ave

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

```
mysql> Select * from employees cross join locations;
```

EmployeeID	Name	Email	ContactNumber	Role	Salary	LocationID	LocationName	Address
10	Jack Evans	jack@example.com	777-999-1111	Security Officer	30000.00	1	Warehouse A	789 Warehouse Ave
9	Ivy Adams	ivy@example.com	111-333-5555	Logistics Coordinator	42000.00	1	Warehouse A	789 Warehouse Ave
8	Henry Thomas	henry@example.com	555-777-9999	Supervisor	48000.00	1	Warehouse A	789 Warehouse Ave
7	Grace Martinez	grace@example.com	987-654-3210	Driver	32000.00	1	Warehouse A	789 Warehouse Ave
6	Frank Wilson	frank@example.com	321-654-9876	Warehouse Manager	55000.00	1	Warehouse A	789 Warehouse Ave
5	Ella Taylor	ella@example.com	123-456-7890	Customer Service	35000.00	1	Warehouse A	789 Warehouse Ave
4	David Lee	david@example.com	999-000-1111	Accountant	45000.00	1	Warehouse A	789 Warehouse Ave
3	Charlie Brown	charlie@example.com	777-888-9999	Dispatcher	40000.00	1	Warehouse A	789 Warehouse Ave
2	Bob Smith	bob@example.com	444-555-6666	Courier	30000.00	1	Warehouse A	789 Warehouse Ave
1	Alice Johnson	alice@example.com	111-222-3333	Manager	50000.00	1	Warehouse A	789 Warehouse Ave
10	Jack Evans	jack@example.com	777-999-1111	Security Officer	30000.00	2	Warehouse B	101 Warehouse St
9	Ivy Adams	ivy@example.com	111-333-5555	Logistics Coordinator	42000.00	2	Warehouse B	101 Warehouse St
8	Henry Thomas	henry@example.com	555-777-9999	Supervisor	48000.00	2	Warehouse B	101 Warehouse St
7	Grace Martinez	grace@example.com	987-654-3210	Driver	32000.00	2	Warehouse B	101 Warehouse St
6	Frank Wilson	frank@example.com	321-654-9876	Warehouse Manager	55000.00	2	Warehouse B	101 Warehouse St
5	Ella Taylor	ella@example.com	123-456-7890	Customer Service	35000.00	2	Warehouse B	101 Warehouse St
4	David Lee	david@example.com	999-000-1111	Accountant	45000.00	2	Warehouse B	101 Warehouse St
3	Charlie Brown	charlie@example.com	777-888-9999	Dispatcher	40000.00	2	Warehouse B	101 Warehouse St
2	Bob Smith	bob@example.com	444-555-6666	Courier	30000.00	2	Warehouse B	101 Warehouse St
1	Alice Johnson	alice@example.com	111-222-3333	Manager	50000.00	2	Warehouse B	101 Warehouse St
10	Jack Evans	jack@example.com	777-999-1111	Security Officer	30000.00	3	Warehouse C	555 Warehouse St
9	Ivy Adams	ivy@example.com	111-333-5555	Logistics Coordinator	42000.00	3	Warehouse C	555 Warehouse St
8	Henry Thomas	henry@example.com	555-777-9999	Supervisor	48000.00	3	Warehouse C	555 Warehouse St
7	Grace Martinez	grace@example.com	987-654-3210	Driver	32000.00	3	Warehouse C	555 Warehouse St
6	Frank Wilson	frank@example.com	321-654-9876	Warehouse Manager	55000.00	3	Warehouse C	555 Warehouse St
5	Ella Taylor	ella@example.com	123-456-7890	Customer Service	35000.00	3	Warehouse C	555 Warehouse St
4	David Lee	david@example.com	999-000-1111	Accountant	45000.00	3	Warehouse C	555 Warehouse St
3	Charlie Brown	charlie@example.com	777-888-9999	Dispatcher	40000.00	3	Warehouse C	555 Warehouse St
2	Bob Smith	bob@example.com	444-555-6666	Courier	30000.00	3	Warehouse C	555 Warehouse St
1	Alice Johnson	alice@example.com	111-222-3333	Manager	50000.00	3	Warehouse C	555 Warehouse St
10	Jack Evans	jack@example.com	777-999-1111	Security Officer	30000.00	4	Warehouse D	777 Warehouse Ave
9	Ivy Adams	ivy@example.com	111-333-5555	Logistics Coordinator	42000.00	4	Warehouse D	777 Warehouse Ave
8	Henry Thomas	henry@example.com	555-777-9999	Supervisor	48000.00	4	Warehouse D	777 Warehouse Ave
7	Grace Martinez	grace@example.com	987-654-3210	Driver	32000.00	4	Warehouse D	777 Warehouse Ave
6	Frank Wilson	frank@example.com	321-654-9876	Warehouse Manager	55000.00	4	Warehouse D	777 Warehouse Ave
5	Ella Taylor	ella@example.com	123-456-7890	Customer Service	35000.00	4	Warehouse D	777 Warehouse Ave
4	David Lee	david@example.com	999-000-1111	Accountant	45000.00	4	Warehouse D	777 Warehouse Ave
3	Charlie Brown	charlie@example.com	777-888-9999	Dispatcher	40000.00	4	Warehouse D	777 Warehouse Ave
2	Bob Smith	bob@example.com	444-555-6666	Courier	30000.00	4	Warehouse D	777 Warehouse Ave
1	Alice Johnson	alice@example.com	111-222-3333	Manager	50000.00	4	Warehouse D	777 Warehouse Ave
10	Jack Evans	jack@example.com	777-999-1111	Security Officer	30000.00	5	Office Headquarters	123 Office St
9	Ivy Adams	ivy@example.com	111-333-5555	Logistics Coordinator	42000.00	5	Office Headquarters	123 Office St
8	Henry Thomas	henry@example.com	555-777-9999	Supervisor	48000.00	5	Office Headquarters	123 Office St

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

```
mysql> select * from couriers left join users on couriers.sendername=users.name;
```

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate	UserID	Name	Email	Password	ContactNumber	Address
1	Sender A	123 Sender St	Receiver X	456 Receiver St	3.50	In Transit	TRACK123	2024-04-12	NULL	NULL	NULL	NULL	NULL	NULL
2	Sender B	789 Sender St	Receiver Y	101 Receiver St	2.20	Delivered	TRACK456	2024-04-10	NULL	NULL	NULL	NULL	NULL	NULL
3	Sender C	321 Sender St	Receiver Z	789 Receiver St	4.00	Pending	TRACK789	2024-04-15	NULL	NULL	NULL	NULL	NULL	NULL
4	Sender D	555 Sender St	Receiver W	888 Receiver St	1.80	In Transit	TRACKXYZ1	2024-04-11	NULL	NULL	NULL	NULL	NULL	NULL
5	Sender E	999 Sender St	Receiver V	222 Receiver St	3.70	Delivered	TRACKLMN	2024-04-09	NULL	NULL	NULL	NULL	NULL	NULL
6	Sender F	111 Sender St	Receiver U	333 Receiver St	2.50	Pending	TRACKOPQ	2024-04-14	NULL	NULL	NULL	NULL	NULL	NULL
7	Sender G	777 Sender St	Receiver T	444 Receiver St	5.20	In Transit	TRACKRST	2024-04-13	NULL	NULL	NULL	NULL	NULL	NULL
8	Sender H	888 Sender St	Receiver S	555 Receiver St	1.90	Delivered	TRACKUVW	2024-04-08	NULL	NULL	NULL	NULL	NULL	NULL
9	Sender I	222 Sender St	Receiver R	666 Receiver St	3.30	Pending	TRACKXYZ2	2024-04-16	NULL	NULL	NULL	NULL	NULL	NULL
10	Sender J	444 Sender St	Receiver Q	777 Receiver St	2.10	In Transit	TRACK1234	2024-04-17	NULL	NULL	NULL	NULL	NULL	NULL

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

```
mysql> select * from couriers left join users on couriers.receivername=users
.name;
+-----------+------------+---------------+--------------+-----------------+--------+------------+----------------+--------------+--------+------+-------+----------+---------------+---------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status     | TrackingNumber | DeliveryDate | UserID | Name | Email | Password | ContactNumber | Address |
+-----------+------------+---------------+--------------+-----------------+--------+------------+----------------+--------------+--------+------+-------+----------+---------------+---------+
|         1 | Sender A   | 123 Sender St | Receiver X   | 456 Receiver St |   3.50 | In Transit | TRACK123       | 2024-04-12   |   NULL | NULL | NULL  | NULL     | NULL          | NULL    |
|         2 | Sender B   | 789 Sender St | Receiver Y   | 101 Receiver St |   2.20 | Delivered  | TRACK456       | 2024-04-10   |   NULL | NULL | NULL  | NULL     | NULL          | NULL    |
|         3 | Sender C   | 321 Sender St | Receiver Z   | 789 Receiver St |   4.00 | Pending    | TRACK789       | 2024-04-15   |   NULL | NULL | NULL  | NULL     | NULL          | NULL    |
|         4 | Sender D   | 555 Sender St | Receiver W   | 888 Receiver St |   1.80 | In Transit | TRACKXYZ1      | 2024-04-11   |   NULL | NULL | NULL  | NULL     | NULL          | NULL    |
|         5 | Sender E   | 999 Sender St | Receiver V   | 222 Receiver St |   3.70 | Delivered  | TRACKLMN       | 2024-04-09   |   NULL | NULL | NULL  | NULL     | NULL          | NULL    |
|         6 | Sender F   | 111 Sender St | Receiver U   | 333 Receiver St |   2.50 | Pending    | TRACKOPQ       | 2024-04-14   |   NULL | NULL | NULL  | NULL     | NULL          | NULL    |
|         7 | Sender G   | 777 Sender St | Receiver T   | 444 Receiver St |   5.20 | In Transit | TRACKRST       | 2024-04-13   |   NULL | NULL | NULL  | NULL     | NULL          | NULL    |
|         8 | Sender H   | 888 Sender St | Receiver S   | 555 Receiver St |   1.90 | Delivered  | TRACKUVW       | 2024-04-08   |   NULL | NULL | NULL  | NULL     | NULL          | NULL    |
|         9 | Sender I   | 222 Sender St | Receiver R   | 666 Receiver St |   3.30 | Pending    | TRACKXYZ2      | 2024-04-16   |   NULL | NULL | NULL  | NULL     | NULL          | NULL    |
|        10 | Sender J   | 444 Sender St | Receiver Q   | 777 Receiver St |   2.10 | In Transit | TRACK1234      | 2024-04-17   |   NULL | NULL | NULL  | NULL     | NULL          | NULL    |
+-----------+------------+---------------+--------------+-----------------+--------+------------+----------------+--------------+--------+------+-------+----------+---------------+---------+
```

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

```
mysql> SELECT l.LocationID, l.LocationName, COALESCE(SUM(p.Amount), 0) AS To
talPayments
    -> FROM Locations l
    -> LEFT JOIN Payments p ON l.LocationID = p.LocationID
    -> GROUP BY l.LocationID, l.LocationName;
+------------+----------------------+---------------+
| LocationID | LocationName         | TotalPayments |
+------------+----------------------+---------------+
|          1 | Warehouse A          |         10.00 |
|          2 | Warehouse B          |         20.00 |
|          3 | Warehouse C          |         30.00 |
|          4 | Warehouse D          |         15.00 |
|          5 | Office Headquarters  |         10.00 |
|          6 | Branch Office 1      |          5.00 |
|          7 | Branch Office 2      |         25.00 |
|          8 | Branch Office 3      |         12.00 |
|          9 | Retail Store         |         18.00 |
|         10 | Distribution Center  |          8.00 |
+------------+----------------------+---------------+
```

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

```
mysql> select * from couriers where sendername in (select sendername from co
uriers group by sendername having count(*) > 1);
Empty set (0.01 sec)
```

44. List all employees who share the same role.

```
mysql> select e1.*, e2.* from employees e1, employees e2 where e1.role=e2.ro
le and e1.employeeid <> e2.employeeid;
Empty set (0.01 sec)
```

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

```
mysql> select * from payments p1 inner join payments p2 on p1.locationid=p2.
locationid and p1.paymentid <> p2.paymentid;
Empty set (0.00 sec)
```

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

```
mysql> select * from couriers c1 inner join couriers c2 on c1.senderaddress=
c2.senderaddress and c1.courierid <> c2.courierid;
Empty set (0.00 sec)
```

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

```
mysql> select * from couriers where weight > (select avg(weight) from courie
rs);
+-----------+------------+---------------+--------------+-----------------+--------+-----------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status    | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+--------------+-----------------+--------+-----------+----------------+--------------+
|         1 | Sender A   | 123 Sender St | Receiver X   | 456 Receiver St |   3.50 | In Transit | TRACK123       | 2024-04-12   |
|         3 | Sender C   | 321 Sender St | Receiver Z   | 789 Receiver St |   4.00 | Pending    | TRACK789       | 2024-04-15   |
|         5 | Sender E   | 999 Sender St | Receiver V   | 222 Receiver St |   3.70 | Delivered  | TRACKLMN       | 2024-04-09   |
|         7 | Sender G   | 777 Sender St | Receiver T   | 444 Receiver St |   5.20 | In Transit | TRACKRST       | 2024-04-13   |
|         9 | Sender I   | 222 Sender St | Receiver R   | 666 Receiver St |   3.30 | Pending    | TRACKXYZ2      | 2024-04-16   |
+-----------+------------+---------------+--------------+-----------------+--------+-----------+----------------+--------------+
```

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

```
mysql> select name from employees where salary > (select avg(salary) from em
ployees);
+---------------+
| name          |
+---------------+
| Alice Johnson |
| David Lee     |
| Frank Wilson  |
| Henry Thomas  |
| Ivy Adams     |
+---------------+
```

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

```
mysql> select sum(cost) as totalcost from courierservices where cost < (sele
ct max(cost) from courierservices);
+-----------+
| totalcost |
+-----------+
|    180.00 |
+-----------+
```

52. Find all couriers that have been paid for

```
mysql> select * from couriers c1 where weight > all(select weight from couri
ers c2 where c2.sendername='john');
+-----------+------------+---------------+--------------+-----------------+--------+-----------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status    | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+--------------+-----------------+--------+-----------+----------------+--------------+
|         1 | Sender A   | 123 Sender St | Receiver X   | 456 Receiver St |   3.50 | In Transit | TRACK123       | 2024-04-12   |
|         2 | Sender B   | 789 Sender St | Receiver Y   | 101 Receiver St |   2.20 | Delivered  | TRACK456       | 2024-04-10   |
|         3 | Sender C   | 321 Sender St | Receiver Z   | 789 Receiver St |   4.00 | Pending    | TRACK789       | 2024-04-15   |
|         4 | Sender D   | 555 Sender St | Receiver W   | 888 Receiver St |   1.80 | In Transit | TRACKXYZ1      | 2024-04-11   |
|         5 | Sender E   | 999 Sender St | Receiver V   | 222 Receiver St |   3.70 | Delivered  | TRACKLMN       | 2024-04-09   |
|         6 | Sender F   | 111 Sender St | Receiver U   | 333 Receiver St |   2.50 | Pending    | TRACKOPQ       | 2024-04-14   |
|         7 | Sender G   | 777 Sender St | Receiver T   | 444 Receiver St |   5.20 | In Transit | TRACKRST       | 2024-04-13   |
|         8 | Sender H   | 888 Sender St | Receiver S   | 555 Receiver St |   1.90 | Delivered  | TRACKUVW       | 2024-04-08   |
|         9 | Sender I   | 222 Sender St | Receiver R   | 666 Receiver St |   3.30 | Pending    | TRACKXYZ2      | 2024-04-16   |
|        10 | Sender J   | 444 Sender St | Receiver Q   | 777 Receiver St |   2.10 | In Transit | TRACK1234      | 2024-04-17   |
+-----------+------------+---------------+--------------+-----------------+--------+-----------+----------------+--------------+
```

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

```
mysql> SELECT LocationID
    -> FROM Payments
    -> WHERE Amount = (SELECT MAX(Amount) FROM Payments);
+------------+
| LocationID |
+------------+
|          3 |
+------------+
```

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

(e.g., 'SenderName'):

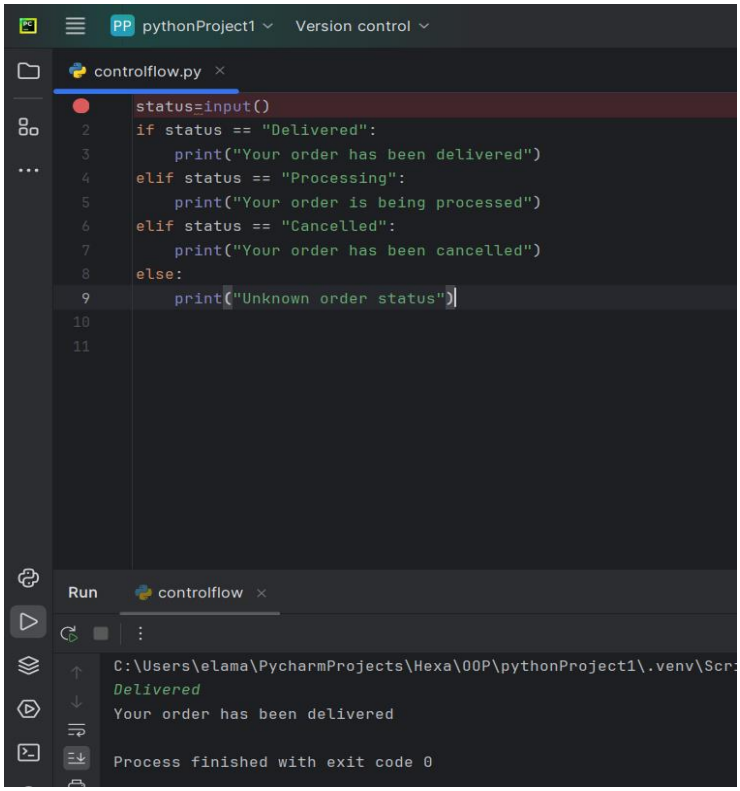
```
mysql> select * from couriers c1 where weight > all (select weight from cour
iers c2 where c2.sendername = 'john');
+-----------+------------+---------------+--------------+-----------------+--------+------------+----------------+--------------+
| CourierID | SenderName | SenderAddress | ReceiverName | ReceiverAddress | Weight | Status     | TrackingNumber | DeliveryDate |
+-----------+------------+---------------+--------------+-----------------+--------+------------+----------------+--------------+
|         1 | Sender A   | 123 Sender St | Receiver X   | 456 Receiver St |   3.50 | In Transit | TRACK123       | 2024-04-12   |
|         2 | Sender B   | 789 Sender St | Receiver Y   | 101 Receiver St |   2.20 | Delivered  | TRACK456       | 2024-04-10   |
|         3 | Sender C   | 321 Sender St | Receiver Z   | 789 Receiver St |   4.00 | Pending    | TRACK789       | 2024-04-15   |
|         4 | Sender D   | 555 Sender St | Receiver W   | 888 Receiver St |   1.80 | In Transit | TRACKXYZ1      | 2024-04-11   |
|         5 | Sender E   | 999 Sender St | Receiver V   | 222 Receiver St |   3.70 | Delivered  | TRACKLMN       | 2024-04-09   |
|         6 | Sender F   | 111 Sender St | Receiver U   | 333 Receiver St |   2.50 | Pending    | TRACKOPQ       | 2024-04-14   |
|         7 | Sender G   | 777 Sender St | Receiver T   | 444 Receiver St |   5.20 | In Transit | TRACKRST       | 2024-04-13   |
|         8 | Sender H   | 888 Sender St | Receiver S   | 555 Receiver St |   1.90 | Delivered  | TRACKUVW       | 2024-04-08   |
|         9 | Sender I   | 222 Sender St | Receiver R   | 666 Receiver St |   3.30 | Pending    | TRACKXYZ2      | 2024-04-16   |
|        10 | Sender J   | 444 Sender St | Receiver Q   | 777 Receiver St |   2.10 | In Transit | TRACK1234      | 2024-04-17   |
+-----------+------------+---------------+--------------+-----------------+--------+------------+----------------+--------------+
```

Coding

Task 1: Control Flow Statements

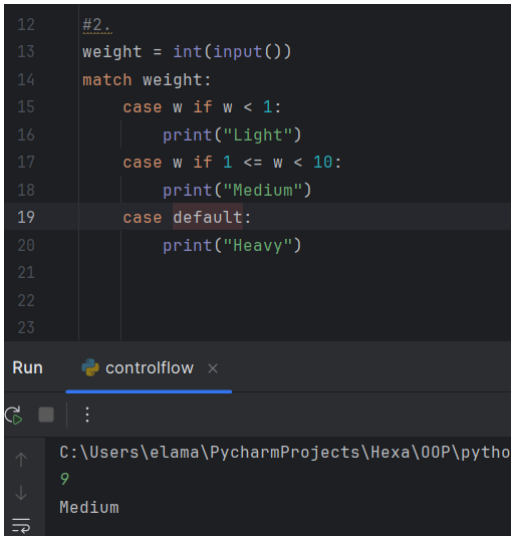
1. Write a program that checks whether a given order is delivered or not based on its status (e.g.,

"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.

```python
def login(username, password):
    if username in users:
        if users[username]["password"] == password:
            role = users[username]["role"]
            return f"Login successful! Role: {role}"
        else:
            return "Incorrect password."
    else:
        return "Username not found."
username = input("Enter your username: ")
password = input("Enter your password: ")

authentication_result = login(username, password)
print(authentication_result)
```

```
controlflow
:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\.venv\
nter your username: employee1
nter your password: pass123
ogin successful! Role: Employee
```

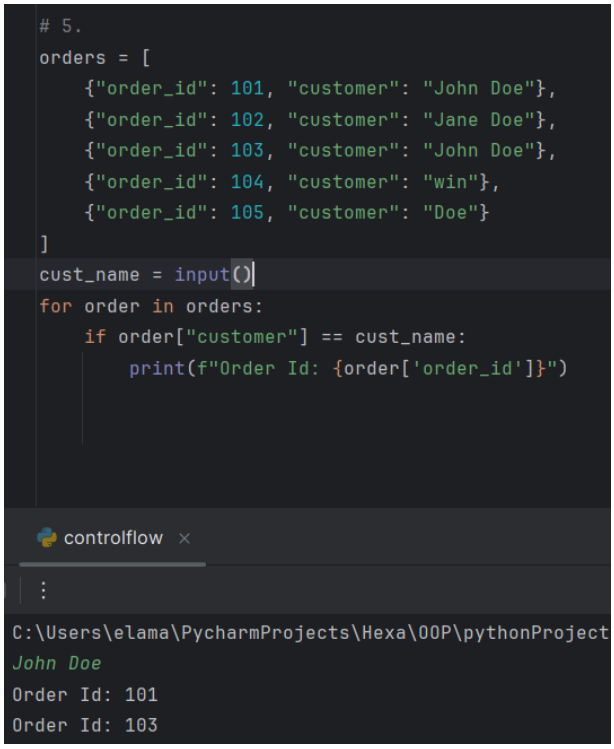
4. Implement Courier Assignment Logic 1. Develop a mechanism to assign couriers to shipments based on predefined criteria (e.g., proximity, load capacity) using loops.

```python
# 4.
couriers = [
    {"name": "Courier 1", "capacity": 50},
    {"name": "Courier 2", "capacity": 100},
    {"name": "Courier 3", "capacity": 200}
]
1 usage
def ass_courier(ship_weight):
    for courier in couriers:
        if courier["capacity"] >= ship_weight:
            return courier["name"]
    return "No suitable courier found"

ship_weight = float(input())
res = ass_courier(ship_weight)
print(res)
```

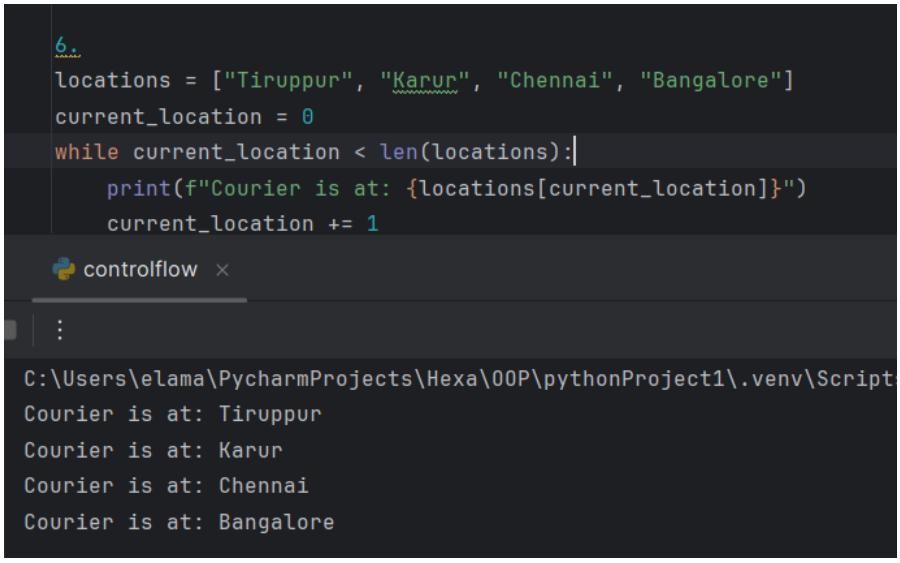
```
controlflow
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1
100
Courier 2
```

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.

```python
#7.
tracking_history = ["Tiruppur", "Karur", "Chennai", "Bangalore"]
for location in tracking_history:
    print(location)
```

```
controlflow ×
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\.venv\Scripts
Tiruppur
Karur
Chennai
Bangalore
```

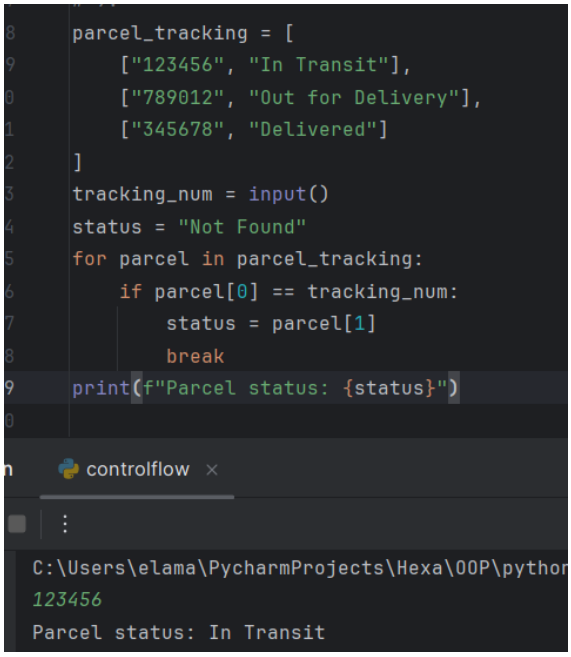
8. Implement a method to find the nearest available courier for a new order using an array of couriers.

```python
#8.
couriers = [
    {"name": "Courier A", "distance": 12},
    {"name": "Courier B", "distance": 8},
    {"name": "Courier C", "distance": 15}
]
nearest_courier=couriers[0]
for courier in couriers:
    if courier["distance"] < nearest_courier["distance"]:
        nearest_courier = courier
print(f"{nearest_courier['name']}")
```

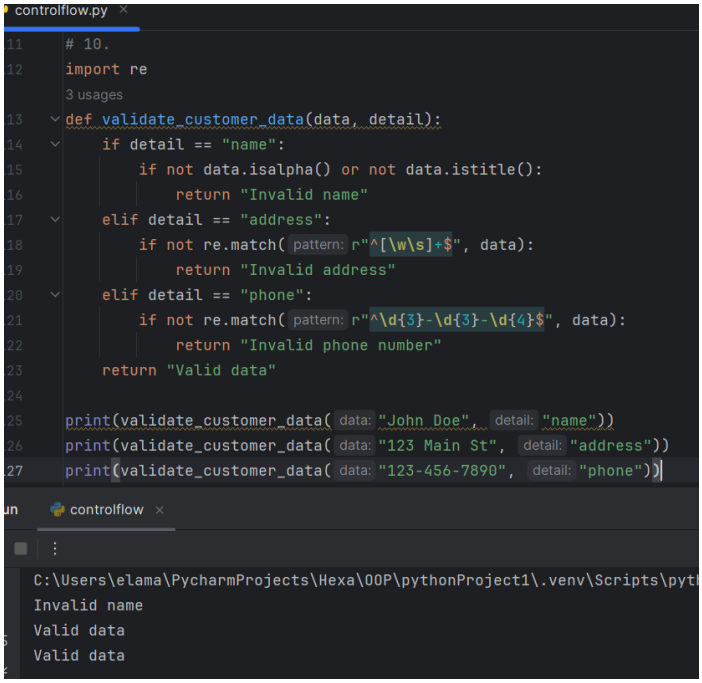
```
un  controlflow ×
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\.venv\S
Courier B
```

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

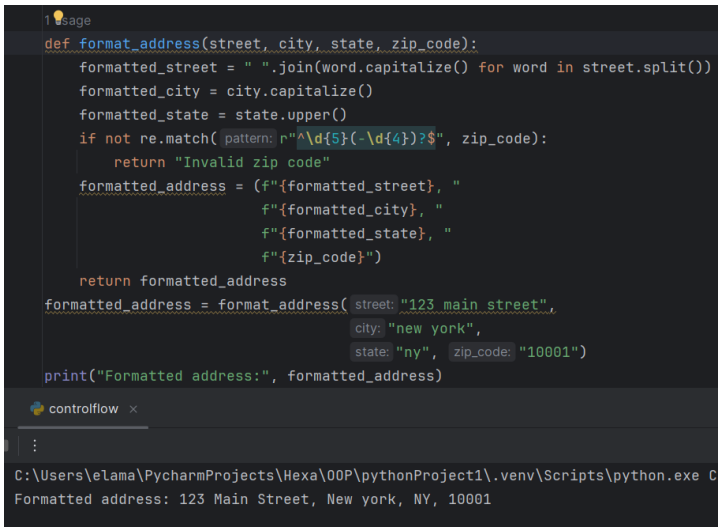
9. Parcel Tracking: Create a program that allows users to input a parcel tracking number.Store the tracking number and Status in 2d String Array. Initialize the array with values. Then, simulate the tracking process by displaying messages like "Parcel in transit," "Parcel out for delivery," or "Parcel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



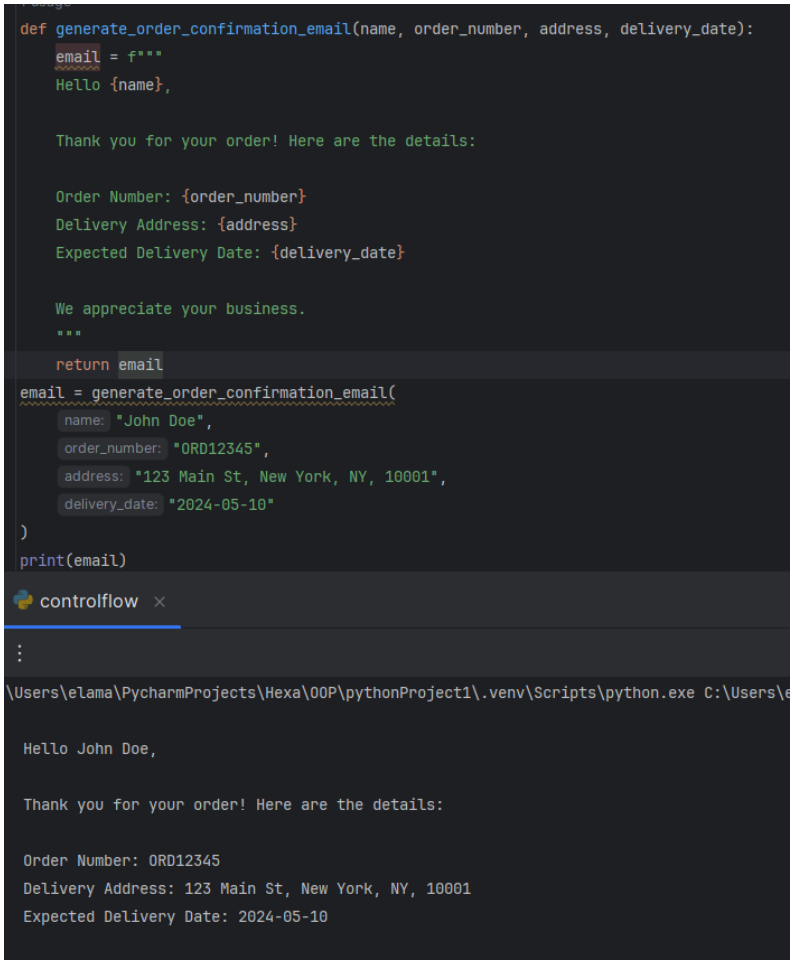
10. Customer Data Validation: Write a function which takes 2 parameters, data-denotes the data and detail-denotes if it is name addtress or phone number.Validate customer information based on following critirea. Ensure that names contain only letters and are properly capitalized, addresses do not contain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



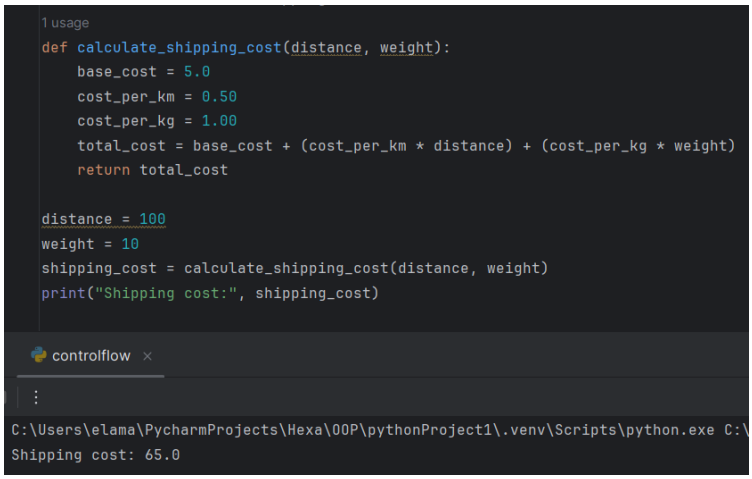
11. Address Formatting: Develop a function that takes an address as input (street, city, state, zip code) and formats it correctly, including capitalizing the first letter of each word and properly formatting the zip code.



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.



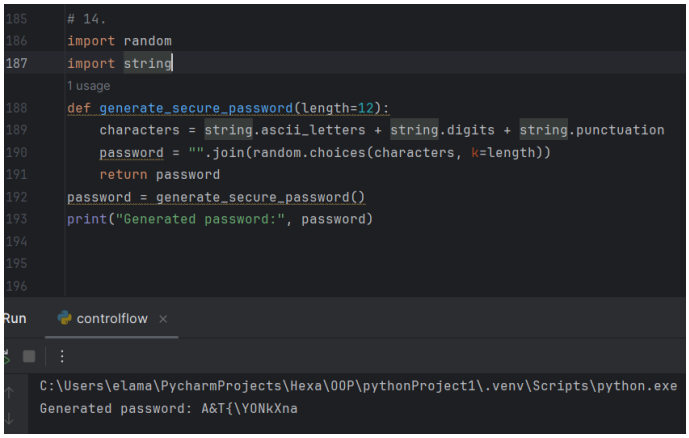
13. Calculate Shipping Costs: Develop a function that calculates the shipping cost based on the distance between two locations and the weight of the parcel. You can use string inputs for the source and destination addresses.



```
1 usage
def calculate_shipping_cost(distance, weight):
    base_cost = 5.0
    cost_per_km = 0.50
    cost_per_kg = 1.00
    total_cost = base_cost + (cost_per_km * distance) + (cost_per_kg * weight)
    return total_cost

distance = 100
weight = 10
shipping_cost = calculate_shipping_cost(distance, weight)
print("Shipping cost:", shipping_cost)
```

```
controlflow ×
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\.venv\Scripts\python.exe C:\
Shipping cost: 65.0
```

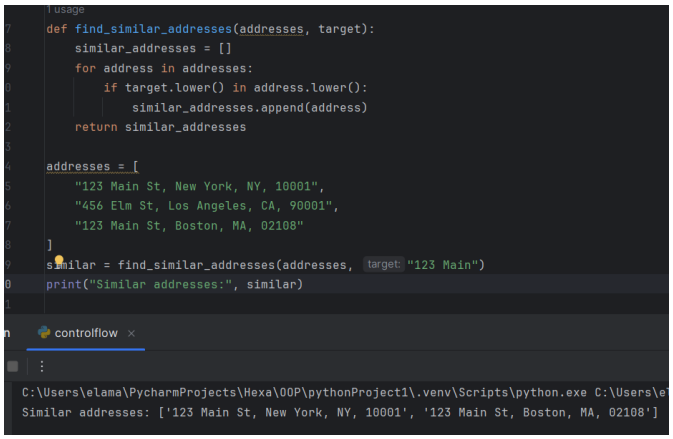
14. Password Generator: Create a function that generates secure passwords for courier system accounts. Ensure the passwords contain a mix of uppercase letters, lowercase letters, numbers, and special characters.



```
# 14.
import random
import string
1 usage
def generate_secure_password(length=12):
    characters = string.ascii_letters + string.digits + string.punctuation
    password = "".join(random.choices(characters, k=length))
    return password
password = generate_secure_password()
print("Generated password:", password)
```

```
Run  controlflow ×
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\.venv\Scripts\python.exe
Generated password: A&T{\YONkXna
```

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.



```
1 usage
def find_similar_addresses(addresses, target):
    similar_addresses = []
    for address in addresses:
        if target.lower() in address.lower():
            similar_addresses.append(address)
    return similar_addresses

addresses = [
    "123 Main St, New York, NY, 10001",
    "456 Elm St, Los Angeles, CA, 90001",
    "123 Main St, Boston, MA, 02108"
]
similar = find_similar_addresses(addresses, target: "123 Main")
print("Similar addresses:", similar)
```

```
controlflow ×
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\.venv\Scripts\python.exe C:\Users\e
Similar addresses: ['123 Main St, New York, NY, 10001', '123 Main St, Boston, MA, 02108']
```

Task 5: Object Oriented Programming

Scope : Entity classes/Models/POJO, Abstraction/Encapsulation

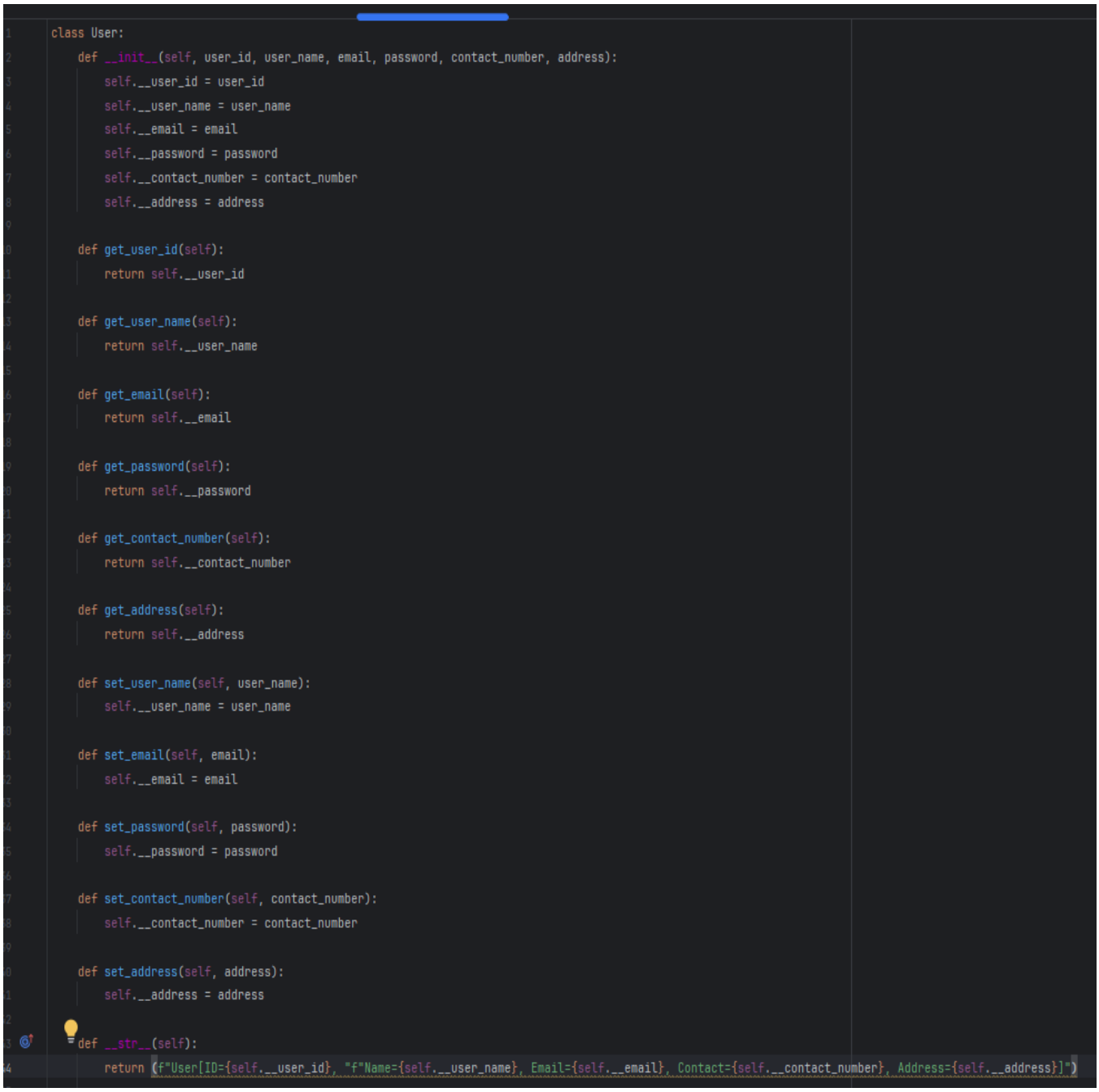
Create the following model/entity classes within package entities with variables declared

private, constructors(default and parametrized,getters,setters and toString())

1. User Class:

Variables:

userID , userName , email , password , contactNumber , address



```python
class User:
    def __init__(self, user_id, user_name, email, password, contact_number, address):
        self.__user_id = user_id
        self.__user_name = user_name
        self.__email = email
        self.__password = password
        self.__contact_number = contact_number
        self.__address = address

    def get_user_id(self):
        return self.__user_id

    def get_user_name(self):
        return self.__user_name

    def get_email(self):
        return self.__email

    def get_password(self):
        return self.__password

    def get_contact_number(self):
        return self.__contact_number

    def get_address(self):
        return self.__address

    def set_user_name(self, user_name):
        self.__user_name = user_name

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

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

    def set_contact_number(self, contact_number):
        self.__contact_number = contact_number

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

    def __str__(self):
        return (f"User[ID={self.__user_id}, "f"Name={self.__user_name}, Email={self.__email}, Contact={self.__contact_number}, Address={self.__address}]")
```

2. Courier Class

Variables: courierID , senderName , senderAddress , receiverName , receiverAddress , weight ,

status, trackingNumber , deliveryDate ,userId

```python
class Courier:
    def __init__(self, courier_id, sender_name, sender_address, receiver_name, receiver_address, weight, status,
                 tracking_number, delivery_date, user_id):
        self.__courier_id = courier_id
        self.__sender_name = sender_name
        self.__sender_address = sender_address
        self.__receiver_name = receiver_name
        self.__receiver_address = receiver_address
        self.__weight = weight
        self.__status = status
        self.__tracking_number = tracking_number
        self.__delivery_date = delivery_date
        self.__user_id = user_id

    def get_courier_id(self):
        return self.__courier_id

    def get_sender_name(self):
        return self.__sender_name

    def get_sender_address(self):
        return self.__sender_address

    def get_receiver_name(self):
        return self.__receiver_name

    def get_receiver_address(self):
        return self.__receiver_address

    def get_weight(self):
        return self.__weight

    def get_status(self):
        return self.__status

    def get_tracking_number(self):
        return self.__tracking_number
```

```python
def get_user_id(self):
    return self.__user_id

def set_sender_name(self, sender_name):
    self.__sender_name = sender_name

def set_sender_address(self, sender_address):
    self.__sender_address = sender_address

def set_receiver_name(self, receiver_name):
    self.__receiver_name = receiver_name

def set_receiver_address(self, receiver_address):
    self.__receiver_address = receiver_address

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

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

def set_tracking_number(self, tracking_number):
    self.__tracking_number = tracking_number

def set_delivery_date(self, delivery_date):
    self.__delivery_date = delivery_date

def set_user_id(self, user_id):
    self.__user_id = user_id

def __str__(self):
    return (f"Courier[ID={self.__courier_id}, Sender={self.__sender_name}, Receiver={self.__receiver_name}, "
            f"Weight={self.__weight}kg, Status={self.__status}, Tracking={self.__tracking_number}, "
            f"Delivery Date={self.__delivery_date}]")
```

3. Employee Class:

Variables employeeID , employeeName , email , contactNumber , role String, salary

```python
class Employee:
    def __init__(self, employee_id, employee_name, email, contact_number, role, salary):
        self.__employee_id = employee_id
        self.__employee_name = employee_name
        self.__email = email
        self.__contact_number = contact_number
        self.__role = role
        self.__salary = salary

    # Getters
    def get_employee_id(self):
        return self.__employee_id

    def get_employee_name(self):
        return self.__employee_name

    def get_email(self):
        return self.__email

    def get_contact_number(self):
        return self.__contact_number

    def get_role(self):
        return self.__role

    def get_salary(self):
        return self.__salary

    # Setters
    def set_employee_name(self, employee_name):
        self.__employee_name = employee_name

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

    def set_contact_number(self, contact_number):
        self.__contact_number = contact_number
```

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

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

def __str__(self):
    return (f"Employee[ID={self.__employee_id}, Name={self.__employee_name}, Email={self.__email}, "
            f"Contact={self.__contact_number}, Role={self.__role}, Salary=${self.__salary}]")
```

4. Location Class

Variables LocationID , LocationName , Address

```python
class Location:
    def __init__(self, location_id, location_name, address):
        self.__location_id = location_id
        self.__location_name = location_name
        self.__address = address

    # Getters
    def get_location_id(self):
        return self.__location_id

    def get_location_name(self):
        return self.__location_name

    def get_address(self):
        return self.__address

    # Setters
    def set_location_name(self, location_name):
        self.__location_name = location_name

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

    def __str__(self):
        return f"Location[ID={self.__location_id}, Name={self.__location_name}, Address={self.__address}]"
```

5. CourierCompany Class

Variables companyName , courierDetails -collection of Courier Objects, employeeDetailscollection of Employee Objects, locationDetails - collection of Location Objects.

```python
class CourierCompany:
    def __init__(self, company_name):
        self.__company_name = company_name
        self.__courier_details = []  # Collection of Courier objects
        self.__employee_details = []  # Collection of Employee objects
        self.__location_details = []  # Collection of Location objects

    # Getters
    def get_company_name(self):
        return self.__company_name

    def get_courier_details(self):
        return self.__courier_details

    def get_employee_details(self):
        return self.__employee_details

    def get_location_details(self):
        return self.__location_details

    # Setters
    def set_company_name(self, company_name):
        self.__company_name = company_name

    def __str__(self):
        return (f"CourierCompany[Name={self.__company_name}, "
                f"Couriers={len(self.__courier_details)}, "
                f"Employees={len(self.__employee_details)}, "
                f"Locations={len(self.__location_details)}]")
```

6. Payment Class:

Variables PaymentID long, CourierID long, Amount double, PaymentDate Date

```python
from datetime import date
class Payment:
    def __init__(self, payment_id, courier_id, amount, payment_date):
        self.__payment_id = payment_id
        self.__courier_id = courier_id
        self.__amount = amount
        self.__payment_date = payment_date

    def get_payment_id(self):
        return self.__payment_id

    def get_courier_id(self):
        return self.__courier_id

    def get_amount(self):
        return self.__amount

    def get_payment_date(self):
        return self.__payment_date

    def set_courier_id(self, courier_id):
        self.__courier_id = courier_id

    def set_amount(self, amount):
        self.__amount = amount

    def set_payment_date(self, payment_date):
        self.__payment_date = payment_date

    def __str__(self):
        return (f"Payment[ID={self.__payment_id}, CourierID={self.__courier_id}, "
                f"Amount=${self.__amount}, Date={self.__payment_date}]")
```

Task 6: Service Provider Interface /Abstract class

Create 2 Interface /Abstract class ICourierUserService and ICourierAdminService interface

ICourierUserService {

// Customer-related functions

placeOrder()

/** Place a new courier order.

* @param courierObj Courier object created using values entered by users

* @return The unique tracking number for the courier order .

Use a static variable to generate unique tracking number. Initialize the static variable in Courier class with some random value. Increment the static variable each time in the constructor to generate next values.

getOrderStatus();

/**Get the status of a courier order.

*@param trackingNumber The tracking number of the courier order.

* @return The status of the courier order (e.g., yetToTransit, In Transit, Delivered).

*/

cancelOrder()

/** Cancel a courier order.

* @param trackingNumber The tracking number of the courier order to be canceled.

* @return True if the order was successfully canceled, false otherwise.*/

getAssignedOrder();

/** Get a list of orders assigned to a specific courier staff member

* @param courierStaffId The ID of the courier staff member.

* @return A list of courier orders assigned to the staff member.*/

// Admin functions

ICourierAdminService

int addCourierStaff(Employee obj);

/** Add a new courier staff member to the system.

* @param name The name of the courier staff member.

* @param contactNumber The contact number of the courier staff member.

* @return The ID of the newly added courier staff member.

*/

```python
# dao/admin_service_impl.py
from dao.i_courier_admin_service import ICourierAdminService
from entity.employee import Employee


2 usages
class AdminServiceImpl(ICourierAdminService):
    def __init__(self):
        self.employees = []  # List of courier staff

    1 usage
    def add_courier_staff(self, employee_obj):
        self.employees.append(employee_obj)
        return employee_obj.get_employee_id()
```

```python
# dao/courier_service_impl.py
import random
from entity.courier import Courier
from dao.i_courier_user_service import ICourierUserService


2 usages
class CourierServiceImpl(ICourierUserService):
    # Static variable for unique tracking number generation
    _tracking_number_counter = random.randint( a: 1000,  b: 9999)

    def __init__(self):
        self.couriers = []  # List of courier orders

    1 usage
    def place_order(self, courier_obj):
        self._tracking_number_counter += 1  # Increment to generate a unique tracking numbe
        courier_obj.set_tracking_number(f"TRACK-{self._tracking_number_counter}")
        self.couriers.append(courier_obj)
        return courier_obj.get_tracking_number()

    1 usage
    def get_order_status(self, tracking_number):
        for courier in self.couriers:
            if courier.get_tracking_number() == tracking_number:
                return courier.get_status()
        return "Tracking number not found"
```

```python
# dao/i_courier_admin_service.py
from abc import ABC, abstractmethod

2 usages
class ICourierAdminService(ABC):
    @abstractmethod
    def add_courier_staff(self, employee_obj):
        """Add a new courier staff member to the system."""
        pass
```

```python
# dao/i_courier_user_service.py
from abc import ABC, abstractmethod

2 usages
class ICourierUserService(ABC):
    @abstractmethod
    def place_order(self, courier_obj):
        """Place a new courier order. Should return a unique tracking number."""
        pass

    @abstractmethod
    def get_order_status(self, tracking_number):
        """Get the status of a courier order based on the tracking number."""
        pass

    @abstractmethod
    def cancel_order(self, tracking_number):
        """Cancel a courier order based on the tracking number."""
        pass

    @abstractmethod
    def get_assigned_order(self, courier_staff_id):
        """Get a list of orders assigned to a specific courier staff member."""
        pass
```

```python
# main/main_module.py
from entity.courier import Courier
from entity.employee import Employee
from dao.courier_service_impl import CourierServiceImpl
from dao.admin_service_impl import AdminServiceImpl
from datetime import date
1 usage
def main():
    # Create instances of the service implementations
    courier_service = CourierServiceImpl()
    admin_service = AdminServiceImpl()

    # Place a new order
    courier = Courier( courier_id: 1,  sender_name: "Alice",  sender_address: "123 Main St",  receiver_name: "Bob",
                       receiver_address: "456 Elm St",  weight: 5.5,  status: "In Transit",  tracking_number: "N/A",
                       delivery_date: "2024-05-10",  user_id: 1)
    tracking_number = courier_service.place_order(courier)

    print(f"Order placed with tracking number: {tracking_number}")

    # Get order status
    status = courier_service.get_order_status(tracking_number)
    print(f"Order status for {tracking_number}: {status}")

    # Cancel an order
    success = courier_service.cancel_order(tracking_number)
    print(f"Order cancellation for {tracking_number}: {'Success' if success else 'Failed'}")

    # Add a new courier staff
    employee = Employee( employee_id: 2,  employee_name: "Charlie Brown",  email: "charlie@example.com",  contact_number: "789-456-1230",  role: "Courier",  salary: 40000)
    staff_id = admin_service.add_courier_staff(employee)

    print(f"New courier staff added with ID: {staff_id}")

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

```
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\.venv\Scripts\py
Order placed with tracking number: TRACK-5162
Order status for TRACK-5162: In Transit
Order cancellation for TRACK-5162: Success
New courier staff added with ID: 2

Process finished with exit code 0
```

Task 7: Exception Handling

(Scope: User Defined Exception/Checked /Unchecked Exception/Exception handling using try..catch finally,thow & throws keyword usage)

Define the following custom exceptions and throw them in methods whenever needed . Handle all the excpetionsin main method,

1. TrackingNumberNotFoundException :throw this exception when user try to withdraw amount or transfer amount to another acco

2. InvalidEmployeeIdException throw this exception when id entered for the employee not existing in the system

```python
# main/main_module.py
from entity.courier import Courier
from entity.employee import Employee
from dao.courier_service_impl import CourierServiceImpl
from dao.admin_service_impl import AdminServiceImpl
from exceptions.custom_exceptions import TrackingNumberNotFoundException, InvalidEmployeeIdException
from datetime import date

1 usage
def main():
    # Create instances of the service implementations
    courier_service = CourierServiceImpl()
    admin_service = AdminServiceImpl()

    try:
        # Place a new order
        courier = Courier( courier_id: 1,  sender_name: "Alice",  sender_address: "123 Main St",  receiver_name: "Bob",
                           receiver_address: "456 Elm St",  weight: 5.5,  status: "In Transit",  tracking_number: "N/A",  delivery_date: "2024-05-10",  user_id: 1)
        tracking_number = courier_service.place_order(courier)
        print(f"Order placed with tracking number: {tracking_number}")

        # Get order status
        status = courier_service.get_order_status(tracking_number)
        print(f"Order status for {tracking_number}: {status}")

        # Attempt to get order status with an invalid tracking number (triggers exception)
        invalid_tracking_number = "TRACK-9999"
        try:
            status = courier_service.get_order_status(invalid_tracking_number)
        except TrackingNumberNotFoundException as e:
            print(e)

        # Cancel the order and catch exceptions if any
        try:
            success = courier_service.cancel_order(invalid_tracking_number)
            print(f"Order cancellation for {invalid_tracking_number}: {'Success' if success else 'Failed'}")
        except TrackingNumberNotFoundException as e:
            print(f"Failed to cancel order: {e}")
```

```python
        # Cancel the order and catch exceptions if any
        try:
            success = courier_service.cancel_order(invalid_tracking_number)
            print(f"Order cancellation for {invalid_tracking_number}: {'Success' if success else 'Failed'}")
        except TrackingNumberNotFoundException as e:
            print(f"Failed to cancel order: {e}")

        # Add a new courier staff and then attempt to retrieve with a valid and invalid ID
        employee = Employee( employee_id: 2,  employee_name: "Charlie Brown",  email: "charlie@example.com",  contact_number: "789-456-1230",  role: "Courier",  salary: 40000)
        staff_id = admin_service.add_courier_staff(employee)
        print(f"New courier staff added with ID: {staff_id}")

        # Attempt to get employee by a valid ID
        valid_employee = admin_service.get_employee_by_id(staff_id)
        print(f"Retrieved employee: {valid_employee}")

        # Attempt to get employee with an invalid ID
        invalid_employee_id = 9999
        try:
            invalid_employee = admin_service.get_employee_by_id(invalid_employee_id)
        except InvalidEmployeeIdException as e:
            print(f"Error: {e}")

    except Exception as e:
        print("An unexpected error occurred:", e)

# Run the main function
if __name__ == "__main__":
    main()
```

```
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\.venv\Scripts\python.exe C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\main\main_module.py
Order placed with tracking number: TRACK-6462
Order status for TRACK-6462: In Transit
Tracking number 'TRACK-9999' not found.
Failed to cancel order: Tracking number 'TRACK-9999' not found.
New courier staff added with ID: 2
Retrieved employee: Employee[ID=2, Name=Charlie Brown, Email=charlie@example.com, Contact=789-456-1230, Role=Courier, Salary=$40000]
Error: Employee ID '9999' is invalid or does not exist.

Process finished with exit code 0
```

Task 8: Collections

Scope: ArrayList/Hashmap

Task: Improve the Courier Management System by using Java collections:

1. Create a new model named CourierCompanyCollection in entity package replacing the Array of Objects with List to accommodate dynamic updates in the CourierCompany class

```python
# entities/courier_company_collection.py
class CourierCompanyCollection:
    def __init__(self, company_name):
        self.company_name = company_name
        self.courier_list = []  # List to store Courier objects
        self.employee_list = []  # List to store Employee objects
        self.location_list = []  # List to store Location objects

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

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

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

    def get_courier_list(self):
        return self.courier_list

    def get_employee_list(self):
        return self.employee_list

    def get_location_list(self):
        return self.location_list

    def __str__(self):
        return (f"CourierCompanyCollection[Name={self.company_name}, "
                f"Couriers={len(self.courier_list)}, Employees={len(self.employee_list)}, "
                f"Locations={len(self.location_list)}]")
```

2. Create a new implementation class CourierUserServiceCollectionImpl class in package dao which implements ICourierUserService interface which holds a variable named companyObj of type CourierCompanyCollection

```python
from dao.i_courier_user_service import ICourierUserService
from entity.courier_company_collection import CourierCompanyCollection
from entity.courier import Courier
from exceptions.custom_exceptions import TrackingNumberNotFoundException
import random
class CourierUserServiceCollectionImpl(ICourierUserService):
    _tracking_number_counter = random.randint( a: 1000, b: 9999)
    def __init__(self):
        self.company = CourierCompanyCollection("My Courier Company")  # Instance of CourierCo
    def place_order(self, courier_obj):
        self._tracking_number_counter += 1  # Increment the tracking number
        tracking_number = f"TRACK-{self._tracking_number_counter}"
        courier_obj.set_tracking_number(tracking_number)
        self.company.add_courier(courier_obj)
        return tracking_number
    def get_order_status(self, tracking_number):
        for courier in self.company.get_courier_list():
            if courier.get_tracking_number() == tracking_number:
                return courier.get_status()
        raise TrackingNumberNotFoundException(tracking_number)
    def cancel_order(self, tracking_number):
        couriers = self.company.get_courier_list()
        for courier in couriers:
            if courier.get_tracking_number() == tracking_number:
                couriers.remove(courier)
                return True
        raise TrackingNumberNotFoundException(tracking_number)
    def get_assigned_order(self, courier_staff_id):
        couriers = self.company.get_courier_list()
        return [courier for courier in couriers if courier.get_user_id() == courier_staff_id]
```

```
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\.venv\Scripts\python.exe C:\Users\elama\P
Order placed with tracking number: TRACK-6634
Order status for TRACK-6634: In Transit
Order cancellation for TRACK-6634: Success
Orders assigned to staff member with ID 1: []
```

Task 8: Service implementation

1.Create CourierUserServiceImpl class which implements ICourierUserService interface which holds a variable named companyObj of type CourierCompany.

This variable can be used to access the Object Arrays to access data relevant in method implementations.

```python
# dao/courier_user_service_impl.py
from dao.i_courier_user_service import ICourierUserService
from entity.courier_company import CourierCompany
from entity.courier import Courier
from exceptions.custom_exceptions import TrackingNumberNotFoundException
import random
class CourierUserServiceImpl(ICourierUserService):
    _tracking_number_counter = random.randint( a: 1000,  b: 9999)

    def __init__(self):
        self.company_obj = CourierCompany("My Courier Company")  # Holds a CourierCompan

    def place_order(self, courier_obj):
        self._tracking_number_counter += 1  # Increment tracking number
        tracking_number = f"TRACK-{self._tracking_number_counter}"
        courier_obj.set_tracking_number(tracking_number)
        self.company_obj.get_courier_details().append(courier_obj)  # Access couriers
        return tracking_number

    def get_order_status(self, tracking_number):
        couriers = self.company_obj.get_courier_details()  # Access couriers
        for courier in couriers:
            if courier.get_tracking_number() == tracking_number:
                return courier.get_status()
        raise TrackingNumberNotFoundException(tracking_number)

    def cancel_order(self, tracking_number):
        couriers = self.company_obj.get_courier_details()
        for courier in couriers:
            if courier.get_tracking_number() == tracking_number:
                couriers.remove(courier)
                return True
        raise TrackingNumberNotFoundException(tracking_number)

    def get_assigned_order(self, courier_staff_id):
        couriers = self.company_obj.get_courier_details()
        assigned_orders = [c for c in couriers if c.get_user_id() == courier_staff_id]
        return assigned_orders
```

2. Create CourierAdminService Impl class which inherits from CourierUserServiceImpl and implements ICourierAdminService interface.

```python
from dao.courier_user_service_impl import CourierUserServiceImpl
from dao.i_courier_admin_service import ICourierAdminService
from exceptions.custom_exceptions import InvalidEmployeeIdException


class CourierAdminServiceImpl(CourierUserServiceImpl, ICourierAdminService):
    def __init__(self):
        super().__init__()  # Initialize the base class

    def add_courier_staff(self, employee):
        self.company_obj.get_employee_details().append(employee)
        return employee.get_employee_id()

    def get_employee_by_id(self, employee_id):
        employees = self.company_obj.get_employee_details()
        for employee in employees:
            if employee.get_employee_id() == employee_id:
                return employee
        raise InvalidEmployeeIdException(employee_id)
```

3. Create CourierAdminServiceCollectionImpl class which inherits from CourierUserServiceColectionImpl and implements ICourierAdminService interface.

```python
# dao/courier_admin_service_collection_impl.py
from dao.courier_user_service_collection_impl import CourierUserServiceCollectionImpl
from dao.i_courier_admin_service import ICourierAdminService
from exceptions.custom_exceptions import InvalidEmployeeIdException


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

    def add_courier_staff(self, employee):
        self.company.get_employee_list().append(employee)
        return employee.get_employee_id()

    def get_employee_by_id(self, employee_id):
        employees = self.company.get_employee_list()
        for employee in employees:
            if employee.get_employee_id() == employee_id:
                return employee
        raise InvalidEmployeeIdException(employee_id)
```

```
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProject1\.venv\Scripts\
Order placed with tracking number: TRACK-8877
New courier staff added with ID: 2
Collection-based order tracking number: TRACK-6164
Collection-based new staff ID: 2

Process finished with exit code 0
```

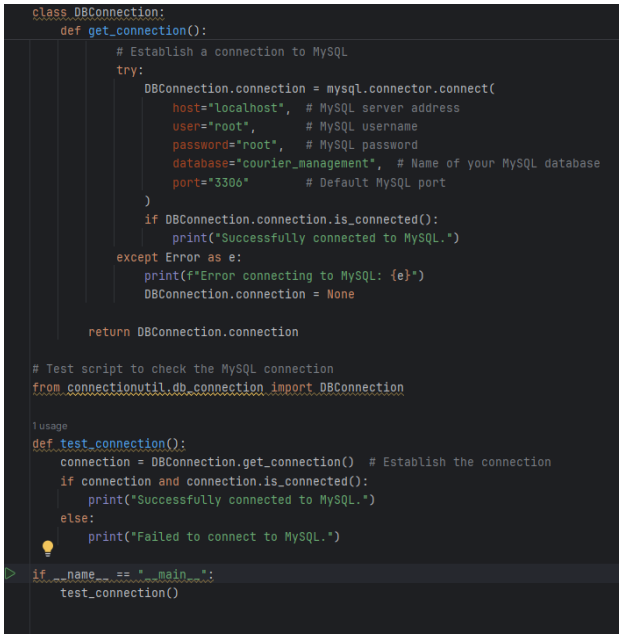
Task 9: Database Interaction

Connect your application to the SQL database for the Courier Management System

1. Write code to establish a connection to your SQL database.

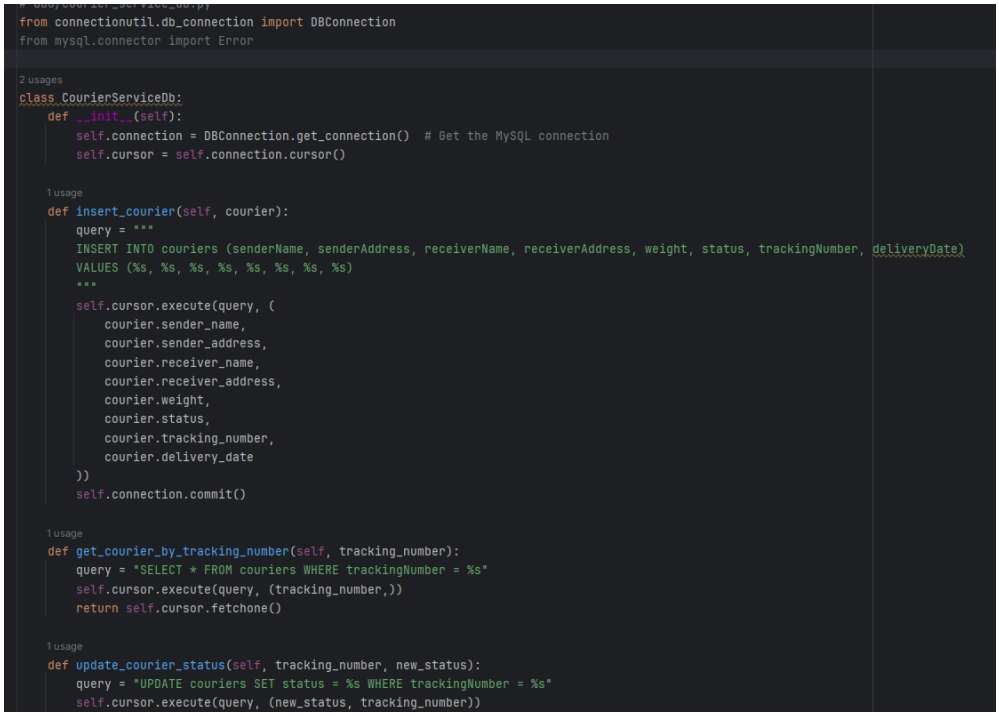
Create a class DBConnection in a package connectionutil with a static variable connection of Type Connection and a static method getConnection() which returns connection.

Connection properties supplied in the connection string should be read from a property file.



```
class DBConnection:
    def get_connection():
            # Establish a connection to MySQL
            try:
                DBConnection.connection = mysql.connector.connect(
                    host="localhost",  # MySQL server address
                    user="root",       # MySQL username
                    password="root",   # MySQL password
                    database="courier_management",  # Name of your MySQL database
                    port="3306"        # Default MySQL port
                )
                if DBConnection.connection.is_connected():
                    print("Successfully connected to MySQL.")
            except Error as e:
                print(f"Error connecting to MySQL: {e}")
                DBConnection.connection = None

        return DBConnection.connection

# Test script to check the MySQL connection
from connectionutil.db_connection import DBConnection

1 usage
def test_connection():
    connection = DBConnection.get_connection()  # Establish the connection
    if connection and connection.is_connected():
        print("Successfully connected to MySQL.")
    else:
        print("Failed to connect to MySQL.")

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

2. Create a Service class CourierServiceDb in dao with a static variable named connection of type Connection which can be assigned in the constructor by invoking the method in DBConnection Class.



```
from connectionutil.db_connection import DBConnection
from mysql.connector import Error

2 usages
class CourierServiceDb:
    def __init__(self):
        self.connection = DBConnection.get_connection()  # Get the MySQL connection
        self.cursor = self.connection.cursor()

    1 usage
    def insert_courier(self, courier):
        query = """
        INSERT INTO couriers (senderName, senderAddress, receiverName, receiverAddress, weight, status, trackingNumber, deliveryDate)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
        """
        self.cursor.execute(query, (
            courier.sender_name,
            courier.sender_address,
            courier.receiver_name,
            courier.receiver_address,
            courier.weight,
            courier.status,
            courier.tracking_number,
            courier.delivery_date
        ))
        self.connection.commit()

    1 usage
    def get_courier_by_tracking_number(self, tracking_number):
        query = "SELECT * FROM couriers WHERE trackingNumber = %s"
        self.cursor.execute(query, (tracking_number,))
        return self.cursor.fetchone()

    1 usage
    def update_courier_status(self, tracking_number, new_status):
        query = "UPDATE couriers SET status = %s WHERE trackingNumber = %s"
        self.cursor.execute(query, (new_status, tracking_number))
```

```
1 usage
def update_courier_status(self, tracking_number, new_status):
    query = "UPDATE couriers SET status = %s WHERE trackingNumber = %s"
    self.cursor.execute(query, (new_status, tracking_number))
    self.connection.commit()

1 usage
def delete_courier(self, tracking_number):
    query = "DELETE FROM couriers WHERE trackingNumber = %s"
    self.cursor.execute(query, (tracking_number,))
    self.connection.commit()

1 usage
def get_courier_by_tracking_number(self, tracking_number):
    # SQL query to get the courier with the given tracking number
    query = "SELECT * FROM couriers WHERE trackingNumber = %s"  # Use %s to
    self.cursor.execute(query, (tracking_number,))  # Execute the query with
    courier = self.cursor.fetchone()  # Fetch one record

    if courier:
        return courier  # If found, return the courier
    else:
        return None  # If not found, return None
```

3. Include methods to insert, update, and retrieve data from the database (e.g., inserting a new order, updating courier status).

```
# main/main_module.py
import uuid  # For generating unique tracking numbers
from dao.courier_service_db import CourierServiceDb
from entity.courier import Courier
from datetime import date  # For handling dates

1 usage
def main():
    courier_service_db = CourierServiceDb()  # Initialize the service class

    while True:
        # Menu-driven interface
        print("Courier Management System")
        print("1. Add a new courier")
        print("2. Get a courier by tracking number")
        print("3. Update a courier's status")
        print("4. Delete a courier")
        print("5. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            # Add a new courier
            unique_tracking_number = f"TRACK-{uuid.uuid4().hex[:8]}"  # Generate a uniq

            courier = Courier(
                courier_id=None,  # If auto-generated, set to None
                sender_name="Alice",
                sender_address="123 Main St",
                receiver_name="Bob",
                receiver_address="456 Elm St",
                weight=5.5,
                status="In Transit",
                tracking_number=unique_tracking_number,
                delivery_date=str(date.today()),
                user_id=1  # Assuming 1 as an example user ID
            )
```

4. Implement a feature to retrieve and display the delivery history of a specific parcel by querying the database. 1. Generate and display reports using data retrieved from the database (e.g., shipment status report, revenue report).

```
C:\Users\elama\PycharmProjects\Hexa\OOP\pythonProjec
Successfully connected to MySQL.
Courier Management System
1. Add a new courier
2. Get a courier by tracking number
3. Update a courier's status
4. Delete a courier
5. Exit
Enter your choice: 1
Courier order inserted successfully.
Courier Management System
1. Add a new courier
2. Get a courier by tracking number
3. Update a courier's status
4. Delete a courier
5. Exit
Enter your choice: 2
Enter tracking number: 123456
Courier not found.
Courier Management System
1. Add a new courier
2. Get a courier by tracking number
3. Update a courier's status
4. Delete a courier
5. Exit
Enter your choice: 3
Enter tracking number: 1234
Enter new status: sent
Courier status updated.
Courier Management System
1. Add a new courier
2. Get a courier by tracking number
3. Update a courier's status
4. Delete a courier
5. Exit
```

```
Enter your choice: 4
Enter tracking number: 123456
Courier deleted.
Courier Management System
1. Add a new courier
2. Get a courier by tracking number
3. Update a courier's status
4. Delete a courier
5. Exit
Enter your choice: 5
Exiting...
```