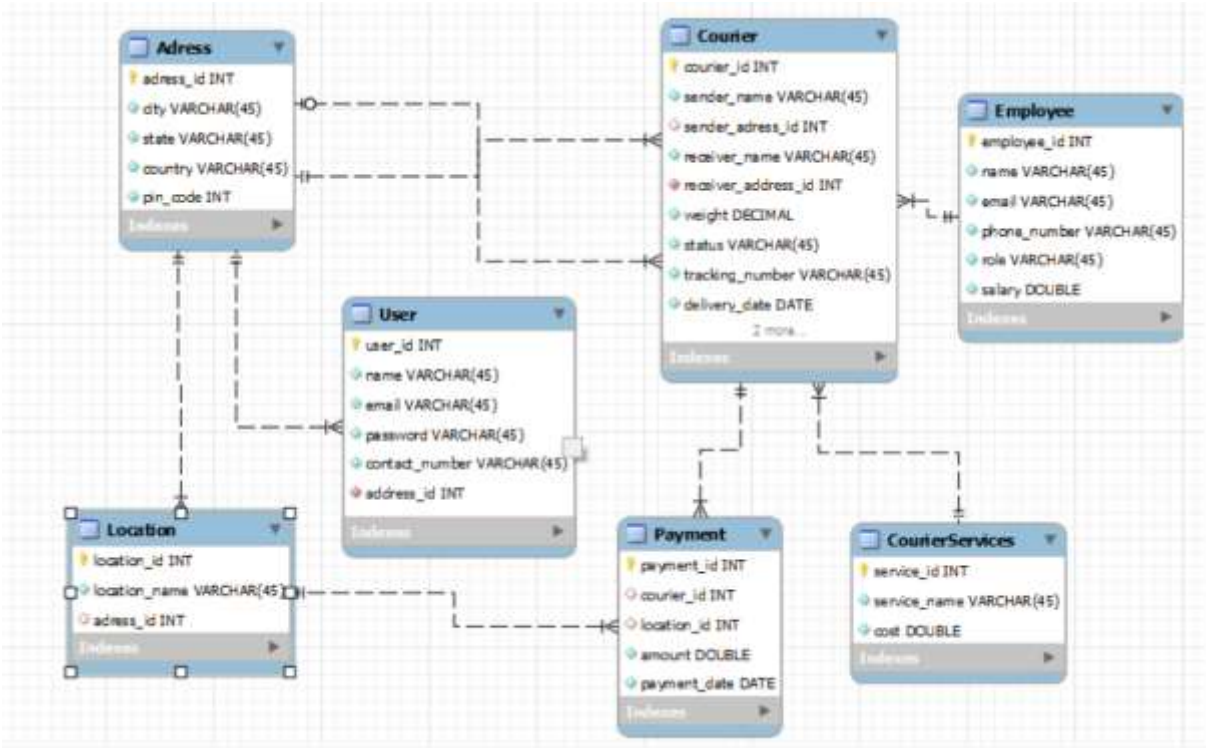

-- MySQL Script generated by MySQL Workbench

-- Wed Mar 6 15:21:20 2024

-- Model: New Model   Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ER
ROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- ---

-- Schema CourierManagementSystem

-- ---

-- ---

```
-- Schema CourierManagementSystem
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `CourierManagementSystem` DEFAULT CHARACTER SET utf8 ;
USE `CourierManagementSystem` ;

-- -----------------------------------------------------
-- Table `CourierManagementSystem`.`Adress`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CourierManagementSystem`.`Adress` (
  `adress_id` INT NOT NULL AUTO_INCREMENT,
  `city` VARCHAR(45) NOT NULL,
  `state` VARCHAR(45) NOT NULL,
  `country` VARCHAR(45) NOT NULL,
  `pin_code` INT NOT NULL,
  PRIMARY KEY (`adress_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CourierManagementSystem`.`User`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CourierManagementSystem`.`User` (
  `user_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `contact_number` VARCHAR(45) NOT NULL,
  `address_id` INT NOT NULL,
  PRIMARY KEY (`user_id`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) ,
  INDEX `user_adress_id_idx` (`address_id` ASC),
```

```
  CONSTRAINT `user_adress_id`
    FOREIGN KEY (`address_id`)
    REFERENCES `CourierManagementSystem`.`Adress` (`adress_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CourierManagementSystem`.`Employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CourierManagementSystem`.`Employee` (
  `employee_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `phone_number` VARCHAR(45) NOT NULL,
  `role` VARCHAR(45) NOT NULL,
  `salary` DOUBLE NOT NULL,
  PRIMARY KEY (`employee_id`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `CourierManagementSystem`.`CourierServices`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CourierManagementSystem`.`CourierServices` (
  `service_id` INT NOT NULL AUTO_INCREMENT,
  `service_name` VARCHAR(45) NOT NULL,
  `cost` DOUBLE NOT NULL,
  PRIMARY KEY (`service_id`))
```

```
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `CourierManagementSystem`.`Courier`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CourierManagementSystem`.`Courier` (
  `courier_id` INT NOT NULL AUTO_INCREMENT,
  `sender_name` VARCHAR(45) NOT NULL,
  `sender_adress_id` INT NULL,
  `receiver_name` VARCHAR(45) NOT NULL,
  `receiver_address_id` INT NOT NULL,
  `weight` DECIMAL NOT NULL,
  `status` VARCHAR(45) NOT NULL,
  `tracking_number` VARCHAR(45) NOT NULL,
  `delivery_date` DATE NOT NULL,
  `employee_id` INT NULL,
  `service_id` INT NULL,
  PRIMARY KEY (`courier_id`),
  UNIQUE INDEX `TrackingNumber_UNIQUE` (`tracking_number` ASC),
  INDEX `sender_adress_id_idx` (`sender_adress_id` ASC),
  INDEX `reciever_adress_id_idx` (`receiver_address_id` ASC),
  INDEX `employee_id_idx` (`employee_id` ASC) ,
  INDEX `service_id_idx` (`service_id` ASC) ,
  CONSTRAINT `sender_adress_id`
    FOREIGN KEY (`sender_adress_id`)
    REFERENCES `CourierManagementSystem`.`Adress` (`adress_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `reciever_adress_id`
    FOREIGN KEY (`receiver_address_id`)
```

```
    REFERENCES `CourierManagementSystem`.`Adress` (`adress_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `employee_id`
    FOREIGN KEY (`employee_id`)
    REFERENCES `CourierManagementSystem`.`Employee` (`employee_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `service_id`
    FOREIGN KEY (`service_id`)
    REFERENCES `CourierManagementSystem`.`CourierServices` (`service_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `CourierManagementSystem`.`Location`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CourierManagementSystem`.`Location` (
  `location_id` INT NOT NULL AUTO_INCREMENT,
  `location_name` VARCHAR(45) NOT NULL,
  `adress_id` INT NULL,
  PRIMARY KEY (`location_id`),
  INDEX `location_adress_id_idx` (`adress_id` ASC),
  CONSTRAINT `location_adress_id`
    FOREIGN KEY (`adress_id`)
    REFERENCES `CourierManagementSystem`.`Adress` (`adress_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `CourierManagementSystem`.`Payment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `CourierManagementSystem`.`Payment` (
  `payment_id` INT NOT NULL AUTO_INCREMENT,
  `courier_id` INT NULL,
  `location_id` INT NULL,
  `amount` DOUBLE NOT NULL,
  `payment_date` DATE NOT NULL,
  PRIMARY KEY (`payment_id`),
  INDEX `Courier_id_idx` (`courier_id` ASC) ,
  INDEX `location_id_idx` (`location_id` ASC),
  CONSTRAINT `Courier_id`
    FOREIGN KEY (`courier_id`)
    REFERENCES `CourierManagementSystem`.`Courier` (`courier_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `location_id`
    FOREIGN KEY (`location_id`)
    REFERENCES `CourierManagementSystem`.`Location` (`location_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

```
insert into Adress(city,state,country,pin_code)
values('vzm','ap','india',535280),
('vizag','ap','india',535280),
('chennai','TN','india',535280),
('mumbai','MR','india',535280),
('hyderabad','TS','india',535280);

INSERT INTO User (name,email,password, contact_number, address_id)
VALUES
('John Doe', 'john.doe@example.com', 'password123', '123-456-7890', 1),
( 'Alice Smith', 'alice.smith@example.com', 'securepass', '987-654-3210', 2),
( 'Bob Johnson', 'bob.johnson@example.com', 'letmein', '555-123-4567', 3),
( 'Emma Davis', 'emma.davis@example.com', 'password321', '777-888-9999', 4),
( 'Michael Wilson', 'michael.wilson@example.com', '123456', '222-333-4444',5),
( 'Sophia Brown', 'sophia.brown@example.com', 'brownie', '111-222-3333', 1),
( 'Olivia Taylor', 'olivia.taylor@example.com', 'qwerty', '444-555-6666', 3),
( 'James Martinez', 'james.martinez@example.com', 'password', '999-888-7777',5),
( 'Ethan Anderson', 'ethan.anderson@example.com', 'anderson', '333-222-1111', 4),
( 'Ava Hernandez', 'ava.hernandez@example.com', 'welcome', '666-777-8888', 2);

INSERT INTO employee ( name, email, phone_number, role, salary)
VALUES
( 'Michael Johnson', 'michael.johnson@example.com', '111-222-3333', 'Manager', 50000.00),
( 'Emily Brown', 'emily.brown@example.com', '444-555-6666', 'Courier', 30000.00),
( 'William Davis', 'william.davis@example.com', '777-888-9999', 'Customer Service', 32000.00),
( 'Sophia Miller', 'sophia.miller@example.com', '123-456-7890', 'Administrator', 45000.00),
( 'Daniel Wilson', 'daniel.wilson@example.com', '987-654-3210', 'Courier', 31000.00),
( 'Emma Garcia', 'emma.garcia@example.com', '111-222-3333', 'Customer Service', 32000.00),
( 'James Martinez', 'james.martinez@example.com', '444-555-6666', 'Courier', 30000.00),
( 'Olivia Taylor', 'olivia.taylor@example.com', '777-888-9999', 'Customer Service', 33000.00),
( 'Ethan Anderson', 'ethan.anderson@example.com', '123-456-7890', 'Courier', 31000.00),
```

('Ava Hernandez', 'ava.hernandez@example.com', '987-654-3210', 'Courier', 30000.00);

INSERT INTO CourierServices (service_name, Cost)

VALUES

('Standard Delivery', 10.99),

('Express Delivery', 20.99),

('International Shipping', 30.99),

('Same-Day Delivery', 25.99),

('Next-Day Delivery', 15.99),

('Overnight Shipping', 18.99),

('Economy Shipping', 8.99),

('Priority Shipping', 22.99),

('Two-Day Shipping', 12.99),

('Three-Day Shipping', 14.99);

INSERT INTO Location (Location_name, adress_id)

VALUES

('Main Office', 2),

('Branch Office', 1),

('Warehouse A', 3),

('Warehouse B', 1),

('Retail Store 1', 4),

('Retail Store 2', 1),

('Headquarters', 1),

('Customer Service Center', 1),

('Regional Office North', 2),

('Regional Office South', 3);

INSERT INTO Payment (courier_iD, location_id, amount, payment_date)

VALUES

(11 , 1, 15.99, '2024-02-28'),

(12, 2, 25.99, '2024-02-25'),

(13, 3, 30.99, '2024-02-26'),

(14, 4, 20.99, '2024-02-27'),

(15, 5, 12.99, '2024-02-24'),

(16, 6, 18.99, '2024-02-23'),

(17, 7, 22.99, '2024-02-22'),

(18, 8, 28.99, '2024-02-21'),

(19, 9, 16.99, '2024-02-20');

select * from location;

drop table courier;

INSERT INTO Courier (employee_id, service_id, sender_name, sender_adress_id, receiver_name, receiver_address_id, Weight, Status, tracking_number, delivery_date)

VALUES

(5, 3, 'John Doe', 1, 'Alice Smith', 3, 1, 'In Transit', 'TRACK123', '2024-03-01'),

(2, 2, 'Bob Johnson', 2, 'Emma Davis', 4, 2, 'Delivered', 'TRACK456', '2024-02-25'),

(9, 4, 'Michael Wilson', 3, 'Sophia Brown', 1, 3, 'Pending', 'TRACK789', '2024-03-01'),

(7, 1, 'Olivia Taylor', 4, 'James Martinez', 2, 1, 'In Transit', 'TRACKABC', '2024-02-27'),

(10, 8, 'Ethan Anderson', 4, 'Ava Hernandez', 3, 2, 'Delivered', 'TRACKDEF', '2024-02-20'),

(10, 2, 'David Clark', 2, 'Sophia Brown', 4, 1, 'In Transit', 'TRACKGHI', '2024-03-03'),

(7, 7, 'Sarah Adams', 3, 'John Doe',1, 2, 'Pending', 'TRACKJKL', '2024-03-01'),

(9, 9, 'Ryan Garcia', 1, 'Olivia Taylor', 2, 1, 'In Transit', 'TRACKMNO', '2024-02-29'),

(2, 10,'Sophia Brown',2, 'David Clark', 3, 2, 'Delivered', 'TRACKPQR', '2024-02-22'),

(5, 6,'Emily White', 3,'Olivia Taylor', 1, 2, 'Delivered', 'TRACKSTU', '2024-02-18');

describe courier;

```
-- Task-2

-- 1.List all customers
select name from user;

-- 2.List all orders for a specific customer
select  * from courier where sendername='Ethan Anderson';

-- 3. List all couriers
select * from courier;

-- 4. List all packages for a specific order
 select * from courier where trackingnumber='TRACK789';

-- 5. List all deliveries for a specific courier
select * from courier where courierid=4;

-- 6. List all undelivered packages
select * from courier where not status= 'delivered';

-- 7. List all packages that are scheduled for delivery today
select * from courier where DeliveryDate=current_date();

-- 8. List all packages with a specific status
select * from courier where status= 'pending';

-- 9. Calculate the total number of packages for each courier
select courierid, count(*) as total_packages from courier group by courierid;

-- 10. Find the average delivery time for each courier
```

```
select abs(avg(datediff(courier.DeliveryDate, payment.PaymentDate))) as Avg_deliverytime from courier inner join payment on courier.courierid=payment.courierid group by payment.courierid;


-- 11. List all packages with a specific weight range

select * from courier where weight between 1.5 and 2.5;


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


select * from employee where name like "%john%";


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

select * from courier inner join  payment on courier.courierid=payment.courierid where amount>20;



-- Task-3

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

select e.employeeid, e.name, count(c.courierid) as total_courier  from employee e left join courier c on e.name in (c.sendername, c.receivername) group by e.employeeid, e.name;


-- 15. Calculate the total revenue generated by each location

select l.locationname, sum(p.amount) from location l join payment p on l.locationid=p.locationid group by l.locationid;

-- update payment set locationid=4 where locationid=3;


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

select count(p.courierid) as num_of_courier, l.locationid, l.locationname from payment p join location l on l.locationid=p.locationid group by p.locationid;


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

select max(abs(datediff(courier.DeliveryDate, payment.PaymentDate))) as high_Avg_deliverytime from courier inner join payment on courier.courierid=payment.courierid;


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

```
select l.locationname, sum(p.amount) as totalpayment from payment p left outer join location l on p.locationid=l.locationid  group by p.locationid having sum(p.amount)>20;


-- 19. Calculate Total Payments per Location

select l.locationname, count(p.amount) as total_payments from payment p join location l on p.locationid=l.locationid group by p.locationid;


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

SELECT c.* FROM courier c JOIN payment p ON c.courierId = p.courierId WHERE p.locationId = 4 GROUP BY c.courierId HAVING SUM(p.amount) > 10;


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

SELECT c.* FROM courier c JOIN payment p ON c.courierId = p.courierId WHERE p.PaymentDate > date('2024-02-24') GROUP BY c.courierId HAVING SUM(p.amount) > 20;


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

select LocationId,LocationName,total_amount,PaymentDate from (select l.LocationId,l.LocationName,sum(p.amount) as total_amount , max(p.PaymentDate) as PaymentDate from location l  join Payment p on p.LocationId=l.LocationId  group by l.LocationId,l.LocationName ) as subquery where total_amount>20 and PaymentDate>'2024-02-25';



-- Task-4

-- 23. Retrieve Payments with Courier Information

select p.*,  c.* from courier c join payment p on c.courierid=p.courierid;


-- 24. Retrieve Payments with Location Information

select l.*,  p.* from location l join payment p on l.locationid=p.locationid;


-- 25. Retrieve Payments with Courier and Location Information

select p.paymentid,  c.*, l.* from courier c join payment p on c.courierid=p.courierid join location l on l.locationid=p.locationid;
```

-- 26. List all payments with courier details

```
select p.* , c.*from payment p join courier c on c.CourierId = p.CourierId;
```

-- 27. Total payments received for each courier

```
SELECT C.CourierID, SUM(P.Amount) AS TotalPaymentsReceived FROM Courier C LEFT JOIN Payment P ON C.CourierID = P.CourierID GROUP BY C.CourierID;
```

-- 28. List payments made on a specific date

```
select * from payment where paymentdate='2024-02-23';
```

-- 29. Get Courier Information for Each Payment

```
select p.paymentid, c.* from courier c  join payment p on c.courierid=p.courierid;
```

-- 30. Get Payment Details with Location

```
select l.locationid,  p.* from location l join payment p on l.locationid=p.locationid;
```

-- 31. Calculating Total Payments for Each Courier

```
select c.courierid, count(p.courierid) as TotalPayments from courier c left join payment p on c.courierid=p.courierid group by c.courierid;
```

-- 32. List Payments Within a Date Range

```
select paymentid from payment where paymentdate between '2024-02-23' and '2024-02-27';
```

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

```
select u.userid, c.courierid, u.name , c.trackingnumber from user u left outer join courier c on u.name=c.sendername

union

select u.userid, c.courierid, u.name , c.trackingnumber from user u right outer join courier c on u.name=c.sendername;
```

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

```
select c.courierid, cs.serviceid, c.sendername, cs.servicename from courier c left outer join courierservices cs on c.serviceid=cs.serviceid

union

select c.courierid, cs.serviceid, c.sendername, cs.servicename from courier c right outer join courierservices cs on c.serviceid=cs.serviceid;
```

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

```
select e.employeeid, e.Name, c.courierid, p.paymentid from employee e left join courier c on c.employeeid=e.employeeID left join Payment P on p.courierid=c.courierID

union

select e.employeeid, e.Name, c.courierid, p.paymentid from employee e right join courier c on c.employeeid=e.employeeID right join Payment P on p.courierid=c.courierID ;
```

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

```
select u.*,cs.* from user u cross join CourierServices cs;
```

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

```
select e.*,l.* from employee e cross join location l;
```

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

```
select c.CourierId,u.* from courier c left outer join user u on c.senderName = u.name;
```

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

```
select c.CourierId,u.* from courier c left outer join user u on c.receiverName = u.name;
```

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

```
select c.courierid, cs.* from courier c left outer join courierservices cs on c.serviceid=cs.serviceid;
```

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

select e.employeeid, count(c.courierid) as NoOfCouriers from employee e left outer join courier c on e.employeeid= c.employeeid group by e.employeeid;

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

select l.*, sum(p.amount) as TotalPayment from location l join payment p on l.locationid=p.locationid group by(locationid);

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

select * from courier where sendername='Ethan Anderson';

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

select Role , count(Role) , group_concat(Name) as EmployeeNames from employee group by (Role);

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

select l.locationid, count(l.locationid) as NumOfOrders, group_concat(p.paymentid) as PaymentID from payment p left outer join location l on p.locationid=l.locationid group by l.locationid;

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

select senderAddress,count(SenderAddress) as NumOfCouriers, group_concat(courierID) as CourierID from courier group by(senderAddress);

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

select e.employeeid, e.name, count(c.courierid) as NumOfCouriers from employee e left outer join courier c on c.employeeid= e.employeeid where c.status='delivered' group by e.employeeid;

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

select c.*from courier c join payment p on c.courierId=p.CourierId join courierServices cs on cs.ServiceId = c.Serviceid Where p.amount > cs.cost;

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

select courierid, weight from courier having weight> (select avg(weight) from courier);

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

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

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

```
select sum(cost) from courierservices where cost < (select max(cost) from courierservices);
```

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

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

```
select l.locationid, l.locationname, p.amount from location l join payment p on p.locationid=
l.locationid  where amount= (select max(amount) from payment);
```

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

```
select * from courier where weight > all (select weight from courier where sendername='Ethan
Anderson');
```