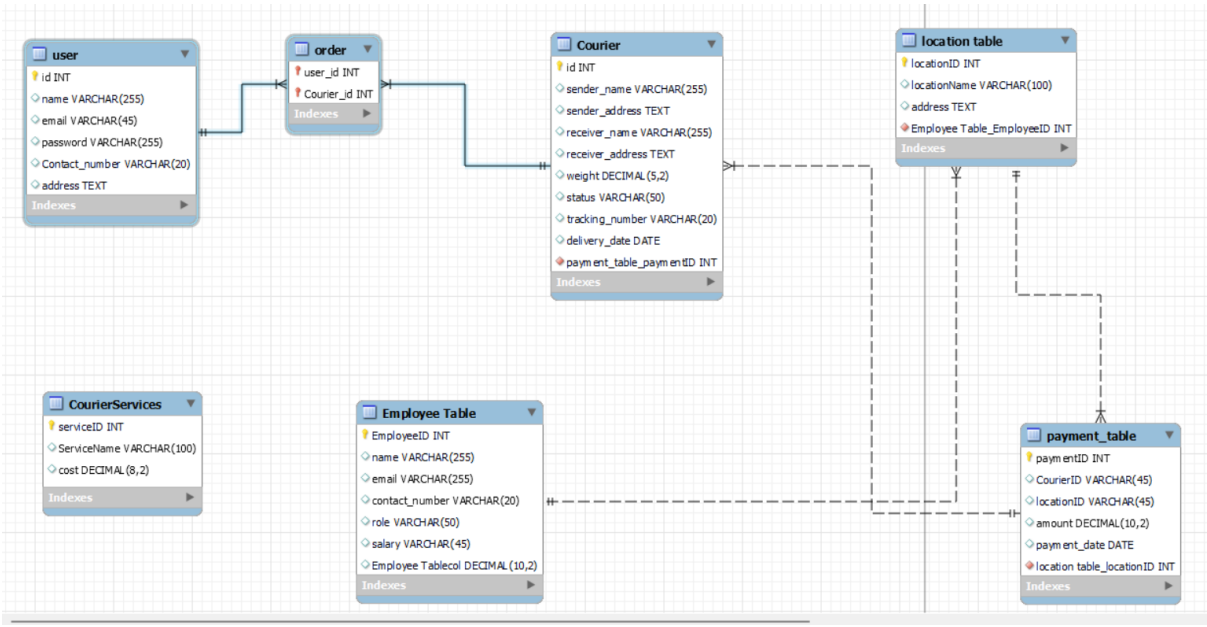
# ASSIGNMENT 2 – COURIER MANAGEMENT




```
-- MySQL Workbench Forward Engineering

-- -----------------------------------------------------

-- Schema courier_management

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema courier_management

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `courier_management` DEFAULT CHARACTER SET utf8 ;

USE `courier_management` ;

-- -----------------------------------------------------

-- Table `courier_management`.`user`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `courier_management`.`user` (

  `id` INT NOT NULL AUTO_INCREMENT,

  `name` VARCHAR(255) NULL,

  `email` VARCHAR(45) NULL,
```

```
  `password` VARCHAR(255) NULL,
  `Contact_number` VARCHAR(20) NULL,
  `address` TEXT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) )
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `courier_management`.`Employee Table`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management`.`Employee Table` (
  `EmployeeID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NULL,
  `email` VARCHAR(255) NULL,
  `contact_number` VARCHAR(20) NULL,
  `role` VARCHAR(50) NULL,
  `salary` VARCHAR(45) NULL,
  `Employee Tablecol` DECIMAL(10,2) NULL,
  PRIMARY KEY (`EmployeeID`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) )
ENGINE = InnoDB;
-- -----------------------------------------------------
-- Table `courier_management`.`location table`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management`.`location table` (
  `locationID` INT NOT NULL,
  `locationName` VARCHAR(100) NULL,
  `address` TEXT NULL,
  `Employee Table_EmployeeID` INT NOT NULL,
  PRIMARY KEY (`locationID`),
  INDEX `fk_location table_Employee Table_idx` (`Employee Table_EmployeeID` ASC) ,
  CONSTRAINT `fk_location table_Employee Table`
```

```
  FOREIGN KEY (`Employee Table_EmployeeID`)
  REFERENCES `courier_management`.`Employee Table` (`EmployeeID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `courier_management`.`payment_table`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management`.`payment_table` (
 `paymentID` INT NOT NULL AUTO_INCREMENT,
 `CourierID` VARCHAR(45) NULL,
 `locationID` VARCHAR(45) NULL,
 `amount` DECIMAL(10,2) NULL,
 `payment_date` DATE NULL,
 `location table_locationID` INT NOT NULL,
 PRIMARY KEY (`paymentID`),
 INDEX `fk_payment_table_location table1_idx` (`location table_locationID` ASC) ,
 CONSTRAINT `fk_payment_table_location table1`
  FOREIGN KEY (`location table_locationID`)
  REFERENCES `courier_management`.`location table` (`locationID`)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `courier_management`.`Courier`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management`.`Courier` (
 `id` INT NOT NULL AUTO_INCREMENT,
 `sender_name` VARCHAR(255) NULL,
 `sender_address` TEXT NULL,
 `receiver_name` VARCHAR(255) NULL,
```

```
  `receiver_address` TEXT NULL,
  `weight` DECIMAL(5,2) NULL,
  `status` VARCHAR(50) NULL,
  `tracking_number` VARCHAR(20) NULL,
  `delivery_date` DATE NULL,
  `payment_table_paymentID` INT NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `tracking_number_UNIQUE` (`tracking_number` ASC) ,
  INDEX `fk_Courier_payment_table1_idx` (`payment_table_paymentID` ASC) ,
  CONSTRAINT `fk_Courier_payment_table1`
    FOREIGN KEY (`payment_table_paymentID`)
    REFERENCES `courier_management`.`payment_table` (`paymentID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `courier_management`.`CourierServices`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management`.`CourierServices` (
  `serviceID` INT NOT NULL AUTO_INCREMENT,
  `ServiceName` VARCHAR(100) NULL,
  `cost` DECIMAL(8,2) NULL,
  PRIMARY KEY (`serviceID`))
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Table `courier_management`.`order`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management`.`order` (
  `user_id` INT NOT NULL,
  `Courier_id` INT NOT NULL,
  PRIMARY KEY (`user_id`, `Courier_id`),
```

```
  INDEX `fk_user_has_Courier_Courier1_idx` (`Courier_id` ASC) ,
  INDEX `fk_user_has_Courier_user1_idx` (`user_id` ASC) ,
  CONSTRAINT `fk_user_has_Courier_user1`
    FOREIGN KEY (`user_id`)
    REFERENCES `courier_management`.`user` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_user_has_Courier_Courier1`
    FOREIGN KEY (`Courier_id`)
    REFERENCES `courier_management`.`Courier` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

---

```
-- Inserting sample data into the user_table
INSERT INTO user_table (name, email, password, contact_num, address) VALUES
('Ajay', 'ajay@gmail.com', 'abc', '1234567890', '123 Main St'),
('Krishna', 'krishna@gmail.com', 'efg', '2345678901', '456 Elm St'),
('Vivek', 'vivek@gmail.com', 'hij', '3456789012', '789 Oak St'),
('Niranjan', 'niranjan@gmail.com', 'xyz', '4567890123', '101 Pine St'),
('Kowshik', 'kowshik@gmail.com', 'mno', '5678901234', '202 Maple St');
```

```
-- Inserting sample data into the courier table
INSERT INTO courier (SenderName, SenderAddress, ReceiverName, ReceiverAddress, weight, status, trackingnumber, deliverydate) VALUES
('Ajay', '123 Main St', 'Krishna', '456 Elm St', '5', 'Pending', '10', '2024-03-07'),
('Vivek', '456 Elm St', 'Ajay', '789 Oak St', '10', 'Delivered', '20', '2024-03-06'),
('Kowshik', '789 Oak St', 'Niranjan', '101 Pine St', '7', 'In Transit', '30', '2024-03-05'),
('Krishna', '101 Pine St', 'Kowshik', '202 Maple St', '3', 'Pending', '40', '2024-03-07'),
('Vivek', '202 Maple St', 'Krishna', '123 Main St', '8', 'Delivered', '50', '2024-03-06');
```

```
-- Inserting sample data into the payment table
INSERT INTO payment (amount, payment_date, location_id, courier_id) VALUES
(50.00, '2024-03-07', 1, 1),
(60.00, '2024-03-06', 2, 2),
(70.00, '2024-03-05', 3, 3),
(80.00, '2024-03-07', 4, 4),
(90.00, '2024-03-06', 5, 5);

-- Inserting sample data into the payment table
INSERT INTO employee (name, email, contact_num, role, salary, location_location_id) VALUES
('Ajay', 'ajay@gmail.com', '1234567890', 'Manager', 50000, 1),
('Krishna', 'krishna@gmail.com', '9876543210', 'Driver', 40000, 2),
('Vivek', 'vivek@gmail.com', '3456789012', 'Warehouse Staff', 35000, 3),
('Niranjan', 'niranjan@gmail.com', '5678901234', 'Courier', 45000, 1),
('Kowshik', 'kowshik@gmail.com', '6789012345', 'Dispatcher', 38000, 2);
INSERT INTO location (location_name, address) VALUES

-- Inserting sample data into the Location table
('Main Office', '123 Main St, City, Country'),
('Warehouse', '456 Elm St, City, Country'),
('Branch Office', '789 Oak St, City, Country'),
('Distribution Center', '101 Pine St, City, Country'),
('Regional Office', '202 Maple St, City, Country');

-- Inserting sample data into the courier_service  table
INSERT INTO courier_service (service_name, cost) VALUES
('Standard Delivery', 10.00),
('Express Delivery', 20.00),
('Same-Day Delivery', 30.00),
('Overnight Delivery', 15.00),
('International Delivery', 50.00);
```

Task 2: Select,Where Solve the following queries in the Schema that you have created above

-- Query to list all customers

```
select * from user;
```

-- Query to list all orders for a specific customer

```
SELECT * FROM courier WHERE sender_name = 'John Doe';
```

-- Query to list all couriers

```
select * from courier;
```

-- Query to list all packages for a specific order

```
select * from courier where tracking_num = '1001';
```

-- Query to list all deliveries for a specific courier

```
select * from courier where courier_id = 1;
```

-- Query to list all undelivered packages

```
select * from courier where status != 'delivered';
```

-- Query to list all packages that are scheduled for delivery today

```
select * from courier where delivery_date = curdate();
```

-- Query to list all packages with a specific status

```
select * from courier where status = 'pending';
```

-- Query to calculate the total number of packages for each courier

```
select courier_id, count(*) as total_packages from courier group by courier_id;
```

-- Query to find the average delivery time for each courier

```
select courier_id, avg(datediff(delivery_date, curdate())) as avg_delivery_time from courier group by courier_id;
```

```
-- Query to list all packages with a specific weight range
select * from courier where weight between 5 and 10;

-- Query to retrieve employees whose names contain 'John'
select * from employee where name like '%john%';

-- Query to retrieve all courier records with payments greater than $50
select c.* from courier c
join payment p on c.courier_id = p.courier_courier_id
where p.amount > 50;
```

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

```
---- Find the total number of couriers handled by each employee.
select e.name, count(c.courier_id) as total_couriers_handled
from employee e
left join courier c on e.employee_id = c.employee_id
group by e.name;

 ----- Calculate the total revenue generated by each location
select l.location_name, sum(p.amount) as total_revenue
from location l
left join payment p on l.location_id = p.location_id
group by l.location_name;

----Find the total number of couriers delivered to each location
select l.location_name, count(c.courier_id) as total_couriers_delivered
from location l
left join courier c on l.location_id = c.delivery_location_id
group by l.location_name;
```

------ Find the courier with the highest average delivery time:

```
select c.courier_id, avg(c.delivery_time) as avg_delivery_time
from courier c
group by c.courier_id
order by avg_delivery_time desc
limit 1;
```

---- Find Locations with Total Payments Less Than a Certain Amount

```
select location_id, sum(amount) as total_amount
from payment
group by location_id
having total_amount < 5000;
```

------- Calculate Total Payments per Location

```
select location_id, sum(amount) as total_amount
from payment
group by location_id;
```

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

```
select courier_id, sum(amount) as total_amount
from payment
where location_id = x
group by courier_id
having total_amount > 1000;
```

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

```
select courier_id, sum(amount) as total_amount
from payment
where payment_date > 'yyyy-mm-dd'
group by courier_id
having total_amount > 1000;
```

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

```
select location_id, sum(amount) as total_amount
from payment
where payment_date > 'yyyy-mm-dd'
group by location_id
having total_amount > 5000;
```

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

---Retrieve Payments with Courier Information

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

---Retrieve Payments with Location Information

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

---Retrieve Payments with Courier and Location Information

```
select p.*, c.*, l.*
from payment p
inner join courier c on p.courier_id = c.courier_id
inner join location l on p.location_id = l.location_id;
```

---List all payments with courier details

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

---- Total payments received for each courier

```
select c.courier_id, sum(p.amount) as total_payments
```

```
from courier c
left join payment p on c.courier_id = p.courier_id
group by c.courier_id;

 --- List payments made on a specific date
select *
from payment
where payment_date = '2024-03-07';

--- Get Courier Information for Each Payment
select p.*, c.*
from payment p
left join courier c on p.courier_id = c.courier_id;

 ---- Get Payment Details with Location
select p.*, l.*
from payment p
left join location l on p.location_id = l.location_id;

--- Calculating Total Payments for Each Courier
select c.courier_id, sum(p.amount) as total_payments
from courier c
left join payment p on c.courier_id = p.courier_id
group by c.courier_id;

 --- List Payments Within a Date Range
select *
from payment
where payment_date between 'start_date' and 'end_date';
```

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

```
select *
from user u
left join courier c on u.user_id = c.user_id;
```

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

```
select *
from courier c
left join courier_service cs on c.courier_service_id = cs.courier_service_id;
```

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

```
select *
from employee e
left join payment p on e.employee_id = p.employee_id;
```

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

```
select *
from user
cross join courier_service;
```

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

```
select * from employee
cross join location;
```

--- Retrieve a list of couriers and their corresponding sender information

```
select c.*, s.*
from courier c
left join (select distinct sendername, senderaddress from courier) s on c.sendername =
s.sendername and c.senderaddress = s.senderaddress;
```

--- Retrieve a list of couriers and their corresponding receiver information

```
select c.*, r.*
from courier c
left join (select receivername, receiveraddress from courier) r on c.receivername = r.receivername
and c.receiveraddress = r.receiveraddress;
```

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

```
select c.*, cs.*
from courier c
left join courier_service cs on c.courier_service_id = cs.courier_service_id;
```

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

```
select e.employee_id, e.name, count(c.courier_id) as num_couriers_assigned
from employee e
left join courier c on e.employee_id = c.employee_id
group by e.employee_id;
```

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

```
select l.location_id, l.location_name, sum(p.amount) as total_payment_amount
from location l
left join payment p on l.location_id = p.location_id
group by l.location_id;
```

-----Retrieve all couriers sent by the same sender

```
select c1.*
from courier c1
join courier c2 on c1.sendername = c2.sendername
where c1.courier_id != c2.courier_id;
```

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

```
select e1.*
from employee e1
```

```
join employee e2 on e1.role = e2.role
where e1.employee_id != e2.employee_id;

---Retrieve all payments made for couriers sent from the same location.
select p1.*
from payment p1
join payment p2 on p1.location_id = p2.location_id
where p1.payment_id != p2.payment_id;
-----Retrieve all couriers sent from the same location
select c1.*
from courier c1
join courier c2 on c1.senderaddress = c2.senderaddress
where c1.courier_id != c2.courier_id;

-----List employees and the number of couriers they have delivered:
select e.employee_id, e.name, count(c.courier_id) as num_couriers_delivered
from employee e
left join courier c on e.employee_id = c.delivered_by_employee_id
group by e.employee_id;

------- 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
```