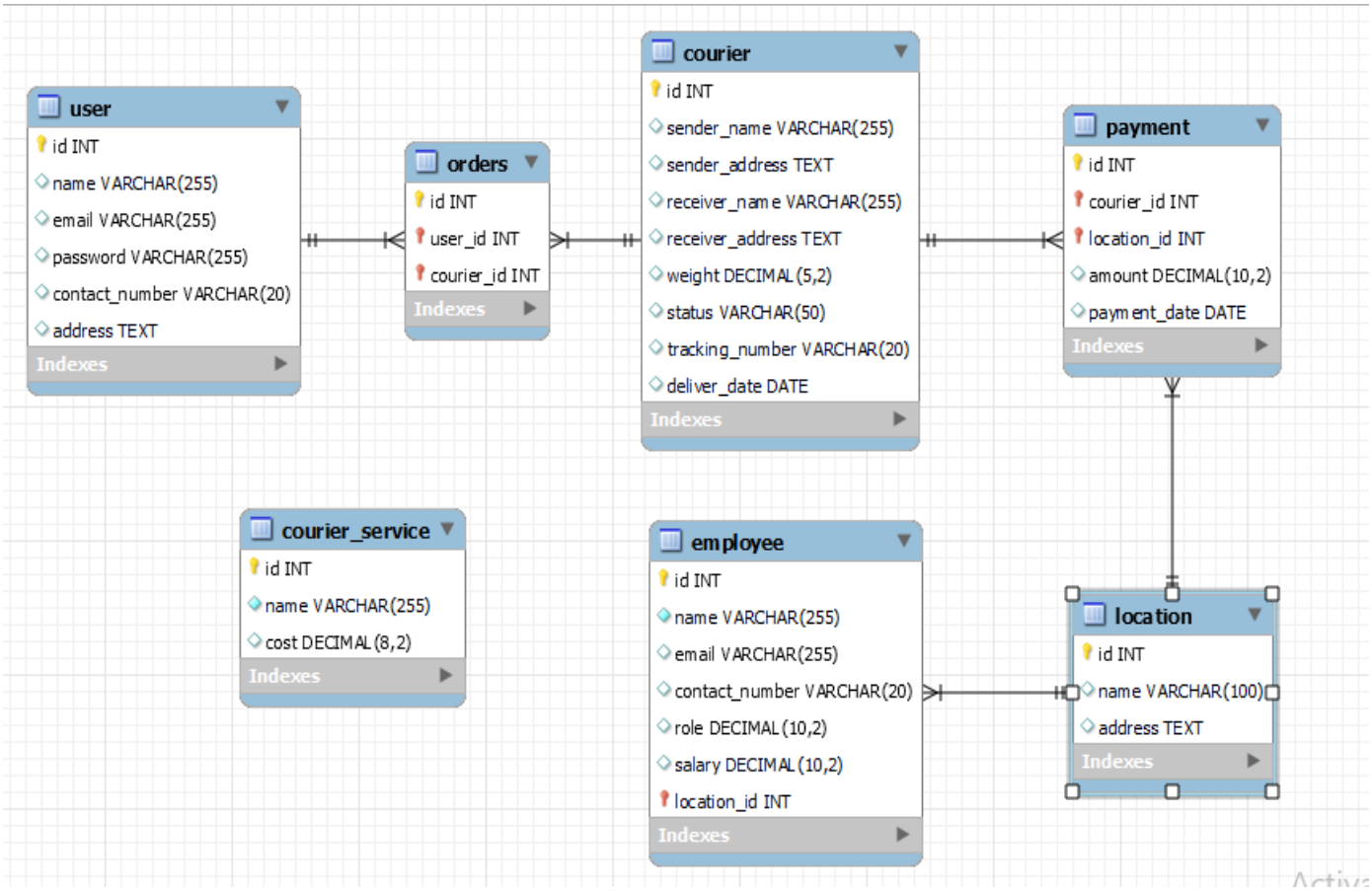
ASSIGNMENT NO : 4

Courier Management System

ER DIAGRAM:



Task:1. Database Design:

-- MySQL Workbench Forward Engineering

-- ---

-- Schema courier_db

-- ---

-- ---

-- Schema courier_db

-- ---

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

```
USE `courier_db`;


-- -----------------------------------------------------
-- Table `courier_db`.`user`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_db`.`user` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NULL,
  `email` VARCHAR(255) 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_db`.`courier`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_db`.`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,
  `deliver_date` DATE NULL,
  PRIMARY KEY (`id`),
```

```
  UNIQUE INDEX `tracking_number_UNIQUE` (`tracking_number` ASC) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `courier_db`.`courier_service`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_db`.`courier_service` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NOT NULL,
  `cost` DECIMAL(8,2) NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `courier_db`.`location`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_db`.`location` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NULL,
  `address` TEXT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `courier_db`.`employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_db`.`employee` (
  `id` INT NOT NULL AUTO_INCREMENT,
```

```
  `name` VARCHAR(255) NOT NULL,
  `email` VARCHAR(255) NULL,
  `contact_number` VARCHAR(20) NULL,
  `role` DECIMAL(10,2) NULL,
  `salary` DECIMAL(10,2) NULL,
  `location_id` INT NOT NULL,
  PRIMARY KEY (`id`, `location_id`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) ,
  INDEX `fk_employee_location1_idx` (`location_id` ASC) ,
  CONSTRAINT `fk_employee_location1`
    FOREIGN KEY (`location_id`)
    REFERENCES `courier_db`.`location` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------
-- Table `courier_db`.`payment`
-- -----------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_db`.`payment` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `courier_id` INT NOT NULL,
  `location_id` INT NOT NULL,
  `amount` DECIMAL(10,2) NULL,
  `payment_date` DATE NULL,
  PRIMARY KEY (`id`, `courier_id`, `location_id`),
  INDEX `fk_courier_has_location_location1_idx` (`location_id` ASC) ,
  INDEX `fk_courier_has_location_courier_idx` (`courier_id` ASC) ,
  CONSTRAINT `fk_courier_has_location_courier`
    FOREIGN KEY (`courier_id`)
```

```
    REFERENCES `courier_db`.`courier` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_courier_has_location_location1`
    FOREIGN KEY (`location_id`)
    REFERENCES `courier_db`.`location` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `courier_db`.`orders`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_db`.`orders` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `user_id` INT NOT NULL,
  `courier_id` INT NOT NULL,
  PRIMARY KEY (`id`, `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_db`.`user` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_user_has_courier_courier1`
    FOREIGN KEY (`courier_id`)
    REFERENCES `courier_db`.`courier` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;
```

INSERTION:

```
-- user insertion
insert into user(name,email,password,contact_number,address) values
('dhana','dhana@gmail.com','dhana@123','9360805403','ranipet'),
('dhivya','dhivya@gmail.com','dhivya@123','9786205333','mudhaliarpet'),
('bharathy','bharathy@gmail.com','bharathy@123','9787333355','muthialpet'),
('bhavana','bhavana@gmail.com','bhavana@123','8220681348','john paul'),
('pradheesha','pradheesha@gmail.com','pradheesha@123','8072607205','anna nagar'),
('kaviya','kaviya@gmail.com','kaviya@123','9443500160','T nagar'),
('prabha','prabha@gmail.com','prabha@123','8610248302','rainbow nagar'),
('dheepika','dheepika@gmail.com','dheepika@123','9787333394','anna nagar'),
('roshini','roshini@gmail.com','roshini@123','9787333394','ranipet'),
('hema','hema@gmail.com','hema@123','9787333364','muthialpet');

-- courier insertion
insert into courier(sender_name,sender_address,receiver_name,receiver_address,weight,status,
tracking_number,deliver_date) values
('dhana','ranipet','dhivya','mudhaliarpet','0.5','shipping started','1A','2024-03-09'),
('dhivya','mudhaliarpet','bharathy','muthialpet','0.6','delivered','2A','2024-03-06'),
('bharathy','muthialpet','bhavana','john paul','0.7','ordered','3A','2024-03-14'),
('bhavana','john paul','pradheesha','anna nagar','1.0','shipping started','1B','2024-03-30'),
('pradheesha','anna nagar','kaviya','T nagar','2.5','ordered','4A','2024-03-31'),
('kaviya','T nagar','prabha','rainbow nagar','3.5','delivered','2B','2024-02-29'),
('prabha','rainbow nagar','dheepika','anna nagar','0.9','shipping started','3B','2024-03-15'),
('dheepika','anna nagar','bharathy','muthialpet','0.8','shipping started','4B','2024-03-16'),
('roshini','ranipet','dheepika','anna nagar','1.5','ordered','1C','2024-03-29'),
('bharathy','muthialpet','dhana','ranipet','2.0','delivered','2C','2024-01-31');
```

```
-- order insertion
insert into orders(user_id,courier_id) values
(1,1),
(2,2),
(3,3),
(4,4),
(5,5),
(6,6),
(7,7),
(8,8),
(9,9),
(3,10);

insert into orders(user_id,courier_id) values(3,10),(7,7);

-- location insertion
insert into location (name,address) values
('chennai','ranipet'),
('puducherry','mudhaliarpet'),
('bangalore','muthialpet'),
('cochin','john paul'),
('mumbai','anna nagar'),
('chennai','T nagar'),
('mahe','rainbow nagar'),
('yanam','lawspet'),
('mumbai','daba'),
('puducherry','white town');

-- employee insertion
insert into employee(name,email,contact_number,role,salary,location_id)values
('leakha','leakha@gmail.com','9360805401','manager',50000,1),
```

```
('ackshaiya','ackshaiya@gmail.com','9786205332','dispatcher',20000,3),
('vishnu','vishnu@gmail.com','9787363353','packing',10000,4),
('ashwini','ashwini@gmail.com','8220681344','delivery',15000,5),
('swetha','swetha@gmail.com','8072607206','packing',10000,6),
('agalya','agalya@gmail.com','9443500165','dispatcher',20000,7),
('devi','devi@gmail.com','8610248304','delivery',15000,8),
('vedha','vedha@gmail.com','9787333397','manager',50000,9),
('ratchana','ratchana@gmail.com','9787353398','dispatcher',20000,10),
('harini','harini@gmail.com','9787323369','packing',10000,10);

alter table employee
modify role varchar(255);

update employee
set role=case
        when id=1 then 'manager'
   when id=2 then 'dispatcher'
   when id=3 then 'packing'
   when id=4 then 'delivery'
   when id=5 then 'packing'
   when id=6 then 'dispatcher'
   when id=7 then 'delivery'
   when id=8 then 'manager'
   when id=9 then 'dispatcher'
   when id=10 then 'packing' end;

-- payment insertion
insert into payment      (courier_id,location_id,amount,payment_date) values
(1,2,500,'2024-03-06'),
(2,3,600,'2024-03-01'),
(3,4,700,'2024-03-07'),
```

```
(4,5,500,'2024-03-06'),
(5,6,100,'2024-03-07'),
(6,7,200,'2024-02-24'),
(7,5,250,'2024-03-06'),
(8,3,275,'2024-03-05'),
(9,5,300,'2024-03-07'),
(10,1,400,'2024-01-25');

-- courier service insertion
insert into courier_service (name,cost) values
('rathimeena',1000),
('shanthimeena',1500),
('harish',2000),
('Ameen',1750),
('sakthi',1250);

insert into courier_service (name,cost) values
('meena',1000),
('wonderchef',1500),
('potter',2000),
('granger',1750),
('weasley',1250);
```

Task 2: Select,Where

```
-- 1. List all customers:

   select * from user;

-- 2. List all orders for a specific customer:
```

```
        select * from courier where sender_name='bharathy';

-- 3. List all couriers:

        select * from courier;

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

   select o.id,c.* from orders o,courier c
   where o.courier_id=c.id
   and o.id=3;

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

   select id from orders
        where courier_id=7;

-- 6. List all undelivered packages:

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

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

   select * from courier
   where deliver_date=curdate();

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

    select * from courier
   where status='shipping started';
```

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

```
select c.*,count(o.courier_id) as total_packages
from courier c,orders o
where c.id=o.courier_id
group by c.id;
```

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

```
select sender_name, avg(datediff(deliver_date,curdate())) as avg_del_date
from courier
group by id;
```

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

```
select * from courier
where weight between 1 and 2;
```

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

```
select name from employee
where name like '%ha%';
```

-- 13. Retrieve all courier records with payments greater than 500.

```
select c.*,p.amount
from payment p,courier c
where p.courier_id=c.id
and p.amount>500;
```

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

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

```
alter table employee
add column courier_id int,
add constraint fk_courier
foreign key (courier_id) references courier(id);

alter table employee
drop foreign key fk_courier;

alter table employee
drop column courier_id;

alter table courier
add column employee_id int,
add constraint fk_emp
foreign key (employee_id) references employee(id);

update courier
set employee_id = case
    when id=1 then 1
    when id=2 then 2
    when id=3 then 3
    when id=4 then 4
    when id=5 then 1
    when id=6 then 1
    when id=7 then 1
    when id=8 then 5
    when id=9 then 6
```

```
    when id=10 then 6 end;

  select e.name, count(c.employee_id) as total_couriers_handled
  from employee e , courier c
  where e.id=c.employee_id
   group by e.name;

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

 select l.*,sum(p.amount)
  from location l ,payment p
  where p.location_id=l.id
  group by l.id;

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

  select receiver_address,count(*) as delivered_count
  from courier
  where status='delivered'
  group by receiver_address;

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

  select sender_name, avg(datediff(deliver_date,curdate())) as avg_del_date
  from courier
  order by avg_del_date limit 0,1;

 select l.*, p.amount
  from location l , payment p
  where l.id=p.location_id
  and p.amount<250;
```

-- 19. Calculate Total Payments per Location

```
select l.*,count(p.amount) as total_payment
from location l ,payment p
where p.location_id=l.id
group by l.id;
```

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

```
select c.*
from courier c,payment p,location l
where l.id=p.location_id and p.courier_id=c.id
and p.amount<=500 and c.receiver_address='anna nagar';
```

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

```
select c.*
from courier c,payment p
where p.courier_id=c.id
and p.amount=500 and payment_date<'2024-03-07';
```

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

```
select l.*,sum(p.amount) as total_amount
from location l ,payment p
where l.id=p.location_id and p.payment_date>'2024-03-03'
```

```
group by l.id
having total_amount>700;
```

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

-- 23. Retrieve Payments with Courier Information

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

-- 24. Retrieve Payments with Location Information

```
select p.*,l.name as city_name,l.address
from payment p join location l
 on p.location_id=l.id;
```

-- 25. Retrieve Payments with Courier and Location Information

```
select c.*,p.amount,p.payment_date,l.name,l.address
from courier c join payment p on c.id=p.courier_id
join location l on l.id=p.location_id;
```

-- 26. List all payments with courier details

```
select c.*,p.amount,p.payment_date
from courier c join payment p on c.id=p.courier_id;
```

-- 27. Total payments received for each courier

```
select c.*,p.amount
```

```
from courier c join payment p on c.id=p.courier_id;
```

-- 28. List payments made on a specific date

```
select c.*,p.amount,p.payment_date
from courier c join payment p on c.id=p.courier_id
where p.payment_date='2024-03-06';
```

-- 29. Get Courier Information for Each Payment

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

-- 30. Get Payment Details with Location

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

-- 31. Calculating Total Payments for Each Courier

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

-- 32. List Payments Within a Date Range

```
select * from payment
where payment_date between '2024-03-05' and '2024-03-07';
```

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

```
select
u.name,u.email,c.id,c.sender_name,c.sender_address,c.receiver_name,c.receiver_address
from user u left join orders o on o.user_id=u.id
left join courier c on c.id=o.courier_id;
```

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

```
select c.id as
courier_id,c.sender_name,c.sender_address,c.receiver_name,c.receiver_address,
cs.id as service_id,cs.name as service_name, cs.cost
from courier c left join courier_service cs on c.id=cs.id;
```

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

```
select e.name,sum(p.amount)
from employee e join payment p on p.location_id=e.location_id
group by e.location_id;
```

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

```
select * from user,courier_service;
```

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

```
select * from employee,location;
```

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

```
select id,sender_name,sender_address from courier;
```

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

```
select id,receiver_name,receiver_address from courier;
```

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

```
select c.id as
courier_id,c.sender_name,c.sender_address,c.receiver_name,c.receiver_address,
cs.id as service_id,cs.name as service_name, cs.cost
from courier c left join courier_service cs on c.id=cs.id;
```

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

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

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

```
select l.name,sum(p.amount) as total_paymnet
from location l join payment p
on p.location_id=l.id
group by l.id;
```

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

```
select * from courier where sender_name='bharathy';
```

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

```
select role,group_concat(name) from employee group by role;
```

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

```
select l.name as city,l.address, p.amount
from location l join payment p on p.location_id=l.id;

select l.name as city,l.address, sum(p.amount) as toatl_payment
from location l join payment p on p.location_id=l.id
group by l.id;
```

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

```
select * from courier
where sender_address in (select sender_address from courier
                         group by sender_address
                         having count(*)>1);
```

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

```
select e.name,count(c.id) as courier_delivered
from employee e join courier c on c.employee_id=e.id
where c.status='delivered'
group by e.id;
```

-- 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.id=p.courier_id
join courier_service cs on c.id=cs.id
where cs.cost<p.amount;
```

-- 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

```
select * from courier
where weight> (select avg(weight) from courier);
```

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

```
select * 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) as total_cost
from courier_service
where cost<(select max(cost) from courier_service);
```

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

```
select c.* from courier c join payment p on p.courier_id=c.id
 where p.payment_date< curdate();
```

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

```
select l.name as city,l.address
from location l join payment p on p.location_id=l.id
where p.amount=(select max(amount) from payment);
```

```
select l.name as city,l.address , sum(p.amount) as total_payment
from location l join payment p on p.location_id=l.id
group by l.id
order by total_payment desc;
```

/* 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>(select sum(weight) from courier where sender_name='prabha');
```