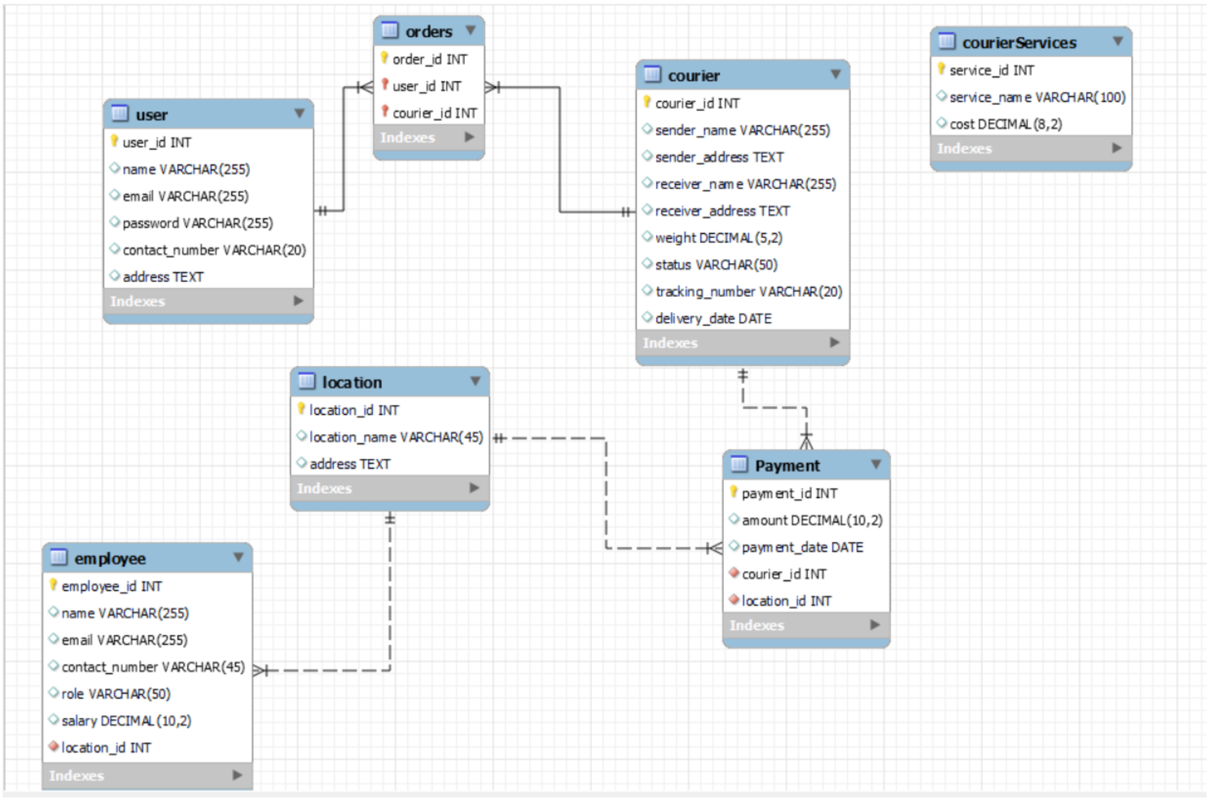
Courier Management Assignment

ER Diagram



TASK-1:

-- MySQL Workbench Forward Engineering

-- ---

-- Schema courier_management_db

-- ---

-- ---

-- Schema courier_management_db

-- ---

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

USE `courier_management_db` ;

-- ---

-- Table `courier_management_db`.`user`

-- ---

```
CREATE TABLE IF NOT EXISTS `courier_management_db`.`user` (
  `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 (`user_id`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `courier_management_db`.`courier`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management_db`.`courier` (
  `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,
  PRIMARY KEY (`courier_id`),
  UNIQUE INDEX `tracking_number_UNIQUE` (`tracking_number` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `courier_management_db`.`courierServices`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management_db`.`courierServices` (
  `service_id` INT NOT NULL AUTO_INCREMENT,
  `service_name` VARCHAR(100) NULL,
  `cost` DECIMAL(8,2) NULL,
  PRIMARY KEY (`service_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `courier_management_db`.`location`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management_db`.`location` (
  `location_id` INT NOT NULL AUTO_INCREMENT,
  `location_name` VARCHAR(45) NULL,
  `address` TEXT NULL,
  PRIMARY KEY (`location_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `courier_management_db`.`employee`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management_db`.`employee` (
  `employee_id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255) NULL,
  `email` VARCHAR(255) NULL,
  `contact_number` VARCHAR(45) NULL,
  `role` VARCHAR(50) NULL,
  `salary` DECIMAL(10,2) NULL,
  `location_id` INT NOT NULL,
  PRIMARY KEY (`employee_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_management_db`.`location` (`location_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `courier_management_db`.`Payment`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `courier_management_db`.`Payment` (
  `payment_id` INT NOT NULL AUTO_INCREMENT,
  `amount` DECIMAL(10,2) NULL,
  `payment_date` DATE NULL,
  `courier_id` INT NOT NULL,
  `location_id` INT NOT NULL,
  PRIMARY KEY (`payment_id`),
  INDEX `fk_Payment_courier_idx` (`courier_id` ASC),
  INDEX `fk_Payment_location1_idx` (`location_id` ASC),
  CONSTRAINT `fk_Payment_courier`
    FOREIGN KEY (`courier_id`)
    REFERENCES `courier_management_db`.`courier` (`courier_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Payment_location1`
    FOREIGN KEY (`location_id`)
    REFERENCES `courier_management_db`.`location` (`location_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `courier_management_db`.`orders`
-- -----------------------------------------------------
```

```
CREATE TABLE IF NOT EXISTS `courier_management_db`.`orders` (
  `order_id` INT NOT NULL AUTO_INCREMENT,
  `user_id` INT NOT NULL,
  `courier_id` INT NOT NULL,
  PRIMARY KEY (`order_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_management_db`.`user` (`user_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_user_has_courier_courier1`
    FOREIGN KEY (`courier_id`)
    REFERENCES `courier_management_db`.`courier` (`courier_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

TASK 2: SELECT, WHERE

1. List all customers:

```
select * from user;
```

2. List all orders for a specific customer:

```
Select * from orders o join courier c on o. courier_id=c.courier_id
where o. user_id=1;
```

3. List all couriers:

```
select * from courier;
```

4. List all packages for a specific order:

```
select * from orders where order_id=3;
```

5. List all deliveries for a specific courier:

```
select * from courier where courier_id=2;
```

6. List all undelivered packages:

```
select * from courier where status not like '%delivered%';
```

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

```
select *
from courier
where DATE(delivery_date) = CURDATE();
```

8. List all packages with a specific status:

```
select * from courier where status='out for delivery';
```

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

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

11. List all packages with a specific weight range:

```
select * from courier where weight between 100 and 200;
```

12. Retrieve employees whose names contain 'John'

```
select * from employee where name='John';
```

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

```
select courier_id, amount from payment
where amount>50;
```

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

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

```
select employee_id,count(location_id) as Total_number_of_couriers_handled
from employee
group by employee_id;
```

15. Calculate the total revenue generated by each location

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

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

```
select l.location_id,l.location_name,count(c.courier_id) as couriers_delivered
from courier c,payment p,location l where c.courier_id=p.courier_id and p.location_id=l.location_id
group by l.location_id;
```

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

```
select location_id from payment where amount <2000;
```

19. Calculate Total Payments per Location

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

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

```
select c.courier_id, sum(p.amount) as total_payment
from courier c,payment p
where c.courier_id=p.courier_id and p.location_id=2
group by p.courier_id
having total_payment>1000;
```

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

```
select c.courier_id, sum(p.amount) as total_payment
from courier c,payment p
where c.courier_id=p.courier_id and p.payment_date> '2024-03-31'
group by p.courier_id
having total_payment>1000;
```

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

```
select l.location_id, sum(p.amount) as total_payment,p.payment_date
from payment p,location l
where l.location_id=p.location_id and p.payment_id < '2024-04-31'
group by p.location_id
having total_payment>5000;
```

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

23. Retrieve Payments with Courier Information

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

24. Retrieve Payments with Location Information

```
select p.payment_id,p.amount,p.payment_date,l.* from
location l join payment p on l.location_id=p.location_id;
```

25. Retrieve Payments with Courier and Location Information

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

26. List all payments with courier details

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

27. Total payments received for each courier

```
select c.courier_id, (case
                    when sum(amount) is null then '0'
      else sum(amount)
      end) as Total_payment
from courier c left join payment p on c.courier_id=p.courier_id
group by c.courier_id
order by total_payment desc;
```

28. List payments made on a specific date

```
select * from payment
where payment_date='2024-04-03';
```

29. Get Courier Information for Each Payment

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

30. Get Payment Details with Location

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

31. Calculating Total Payments for Each Courier

```
select c.courier_id, (case
                  when sum(amount) is null then '0'
      else sum(amount)
      end) as Total_payment
from courier c left join payment p on c.courier_id=p.courier_id
group by c.courier_id
order by total_payment desc;
```

32. List Payments Within a Date Range

```
select * from payment
where payment_date between  '2024-04-01' and '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

```
(select u.*,c.* from user u
left join orders o on u.user_id=o.user_id
left join courier c on o.courier_id=c.courier_id)
union
(select u.*,c.* from user u
right join orders o on u.user_id=o.user_id
right join courier c on o.courier_id=c.courier_id);
```

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

select * from

employee,location;

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 service_id,sum(cost) as total_cost from courierServices

where cost<(select max(cost) from courierServices);

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

select * from location

where location_id in (select location_id from payment 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> (select weight from courier where sender_name='Arjun');