The National University of Computer and Emerging Sciences

Assignment 04:
Database Systems

Sr no.	Name:	Roll no:	Section:
1.	Hasan Yahya	22L-7971	BSE-4B
2.	Wasee Ur Rehman	22L-7992	BSE-4B
3.	Usama Walayat	22L-7874	BSE-4B
4.	Momin Shehzad	22L-7959	BSE-4B

Question 01:

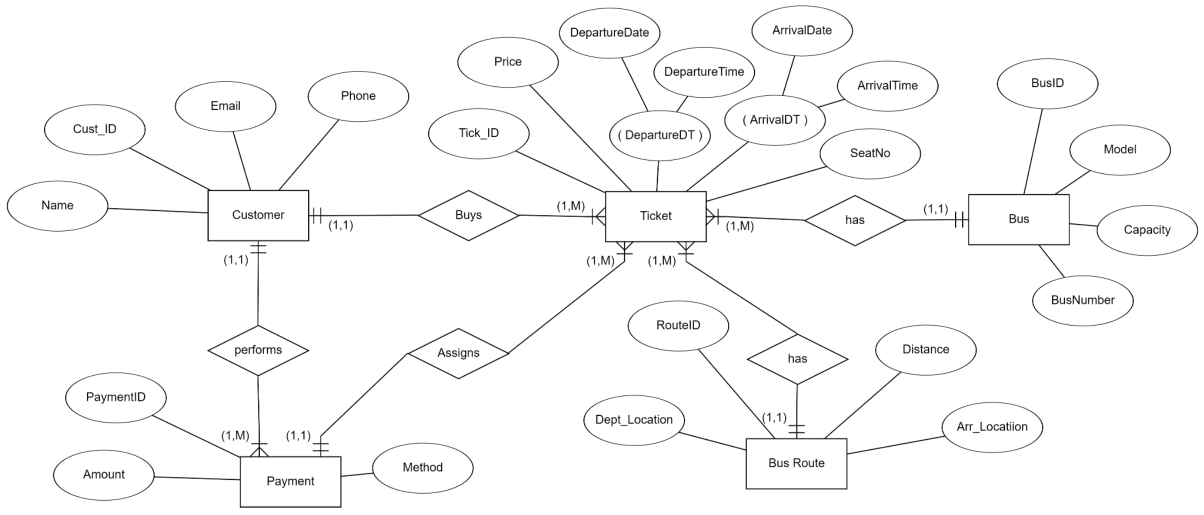
Consider a scenario where you're tasked with designing an ERD for a bus ticketing system. This system facilitates the sale of tickets for bus transportation services.

Entities:

- **Customer:** Represents individuals who purchase tickets. Attributes may include CustomerID, Name, Email, Phone, etc.
- **Ticket:** Represents the ticket purchased by a customer for a bus journey. Attributes may include TicketID, Price, DepartureDateTime, ArrivalDateTime, SeatNumber, etc.
- **Bus Route:** Represents the specific route for bus journeys. Attributes may include RouteID, DepartureLocation, ArrivalLocation, Distance, etc.
- **Bus:** Represents the individual buses available for booking. Attributes may include BusID, BusNumber, Model, Capacity, etc.
- **Payment:** Represents the payment details for each ticket transaction. Attributes may include PaymentID, Amount, PaymentMethod, etc.

Relationships:

- **Customer - Ticket (1:M):** A customer can purchase multiple tickets, but each ticket is associated with only one customer.
- **Ticket - Bus Route (M:1):** Each ticket corresponds to a specific bus route, but a bus route can have multiple tickets associated with it.
- **Ticket - Bus (M:1):** Each ticket is for a specific bus, but each bus can have multiple tickets associated with it.
- **Customer - Payment (1:M):** A customer can make multiple payments, but each payment is associated with only one customer.
- **Payment - Ticket (1:1 or 1:M):** Each payment is associated with one or more tickets, depending on whether a customer purchases multiple tickets in a single transaction or not. I'm going to assume it is **(1:M)**.



Step 1: Identify Candidate Keys

For each of our entities, the candidate keys are:

- Customer: Candidate Key is **CustomerID**
- Ticket: Candidate Key is **TicketID**
- Bus Route: Candidate Key is **RouteID**
- Bus: Candidate Key is **BusID**
- Payment: Candidate Key is **PaymentID**

Step 2: Functional Dependencies

Customer Entity:

- **CustomerID** → Name, Email, Phone
  - The **CustomerID** uniquely identifies each customer and determines the other attributes such as Name, Email, and Phone.

Ticket Entity:

- **TicketID** → Price, DepartureDateTime, ArrivalDateTime, SeatNumber, CustomerID, RouteID, BusID
  - The **TicketID** uniquely identifies each ticket and determines the price, departure and arrival times, seat number, the customer who purchased it, the route it pertains to, and the bus it is associated with.

Bus Route Entity:

- **RouteID** → DepartureLocation, ArrivalLocation, Distance
  - The **RouteID** uniquely identifies each bus route and determines the departure and arrival locations as well as the distance of the route.

Bus Entity:

- **BusID** → BusNumber, Model, Capacity
  - The **BusID** uniquely identifies each bus and determines the bus number, model, and capacity.

Payment Entity:

- **PaymentID** → Amount, PaymentMethod, CustomerID.
  - The **PaymentID** uniquely identifies each payment and determines the amount, method of payment, and the customer who made the payment.

Relationships:

Customer - Ticket Relationship:

- **CustomerID** → TicketID (Partial dependency due to 1:M relationship).
    - A **CustomerID** can be associated with multiple **TicketIDs**, but each **TicketID** is associated with only one **CustomerID**.

Ticket - Bus Route Relationship:

- **TicketID** → RouteID (Partial dependency due to M:1 relationship)
    - Each **TicketID** is associated with one **RouteID**, but a **RouteID** can have multiple **TicketIDs** associated with it.#

Ticket - Bus Relationship:

- **TicketID** → BusID (Partial dependency due to M:1 relationship).
    - Each **TicketID** is associated with a specific **BusID**, but a **BusID** can have multiple **TicketIDs** associated with it.

Customer - Payment Relationship:

- **CustomerID** → PaymentID (Partial dependency due to 1:M relationship).
    - A **CustomerID** can be associated with multiple **PaymentIDs**, but each **PaymentID** is associated with only one **CustomerID**.

Payment - Ticket Relationship:

- **PaymentID** → TicketID (Partial dependency due to 1:M relationship).
    - A **PaymentID** may be associated with multiple **TicketIDs** if a customer purchases multiple tickets in a single transaction.

Step 3: Check for BCNF Violations:

Using the functional dependencies, we check for any violations. For example, in the Ticket entity, TicketID should be the only determinant for any attribute, since it's the candidate key.

Step 4: Decompose Non-BCNF Relations:

Now, we will decompose the relations that are not in BCNF:

Customer:

- CustomerID → Name, Email, Phone
    - This relation is already in BCNF because the determinant **CustomerID** is a candidate key.

Ticket:

- TicketID → Price, DepartureDateTime, ArrivalDateTime, SeatNumber, CustomerID, RouteID, BusID
    - This relation is not in BCNF because **TicketID** does not functionally determine **CustomerID**, **RouteID**, or **BusID** on its own; these are part of other entities.
    - **Decomposition**:
        - Ticket (TicketID, Price, DepartureDateTime, ArrivalDateTime, SeatNumber)
        - CustomerTicket (CustomerID, TicketID) // Linking table for Customer to Ticket
        - RouteTicket (RouteID, TicketID) // Linking table for Route to Ticket
        - BusTicket (BusID, TicketID) // Linking table for Bus to Ticket.

Bus Route:

- RouteID → DepartureLocation, ArrivalLocation, Distance.
    - This relation is already in BCNF because the determinant **RouteID** is a candidate key.

Bus:

- BusID → BusNumber, Model, Capacity
    - This relation is already in BCNF because the determinant **BusID** is a candidate key.

Payment:

- PaymentID → Amount, PaymentMethod, CustomerID.
    - This relation is not in BCNF because **PaymentID** does not functionally determine **CustomerID** on its own; it's part of the Customer entity.
    - **Decomposition**:
        - Payment (PaymentID, Amount, PaymentMethod)
        - CustomerPayment (CustomerID, PaymentID) // Linking table for Customer to Payment

Final BCNF Relations:

1. **Customer**: (CustomerID, Name, Email, Phone)
2. **Ticket**: (TicketID, Price, DepartureDateTime, ArrivalDateTime, SeatNumber)
3. **Bus Route**: (RouteID, DepartureLocation, ArrivalLocation, Distance)
4. **Bus**: (BusID, BusNumber, Model, Capacity)
5. **Payment**: (PaymentID, Amount, PaymentMethod)
6. **CustomerTicket**: (CustomerID, TicketID)
7. **RouteTicket**: (RouteID, TicketID)
8. **BusTicket**: (BusID, TicketID)

9. **CustomerPayment**: (CustomerID, PaymentID).
10. **PaymentTicket**: (PaymentID, TicketID).

Question 02:

EMPLOYEE (Entity) has the following attributes:

- **Employee ID (identifier)**
- **Name**
- **Address**
- **Birthdate**

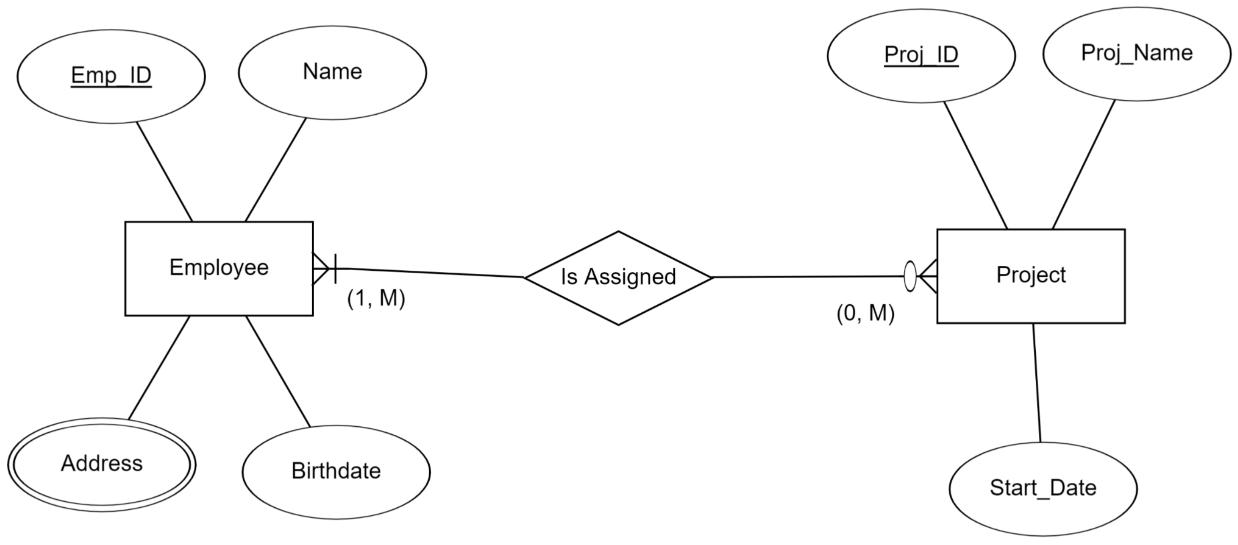
PROJECT (Entity) has the following attributes:

- **Project ID (identifier)**
- **Project Name**
- **Start Date**

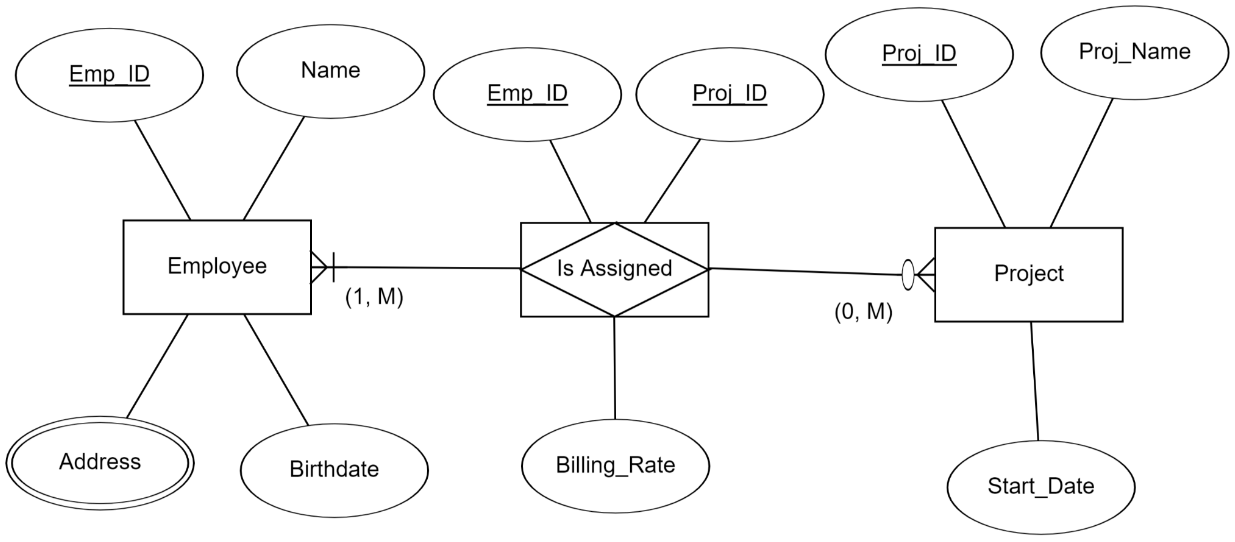
Each project is assigned to one or more Employees and each Employee is assigned to one or more projects. This means the relationship is named as **Assignment**.

- **Each employee may be assigned to one or more projects or may not be assigned to a project:** This means that the cardinality between EMPLOYEE and PROJECT is zero or many because each employee can be assigned to none or many projects.
- **A project must have at least one employee assigned and may have any number of employees assigned:** This means the project and employee have a one or many relationship because a project has one or many employees assigned to it.

Assumption: Address is a multivalued attribute because it has alphanumeric values (i think).



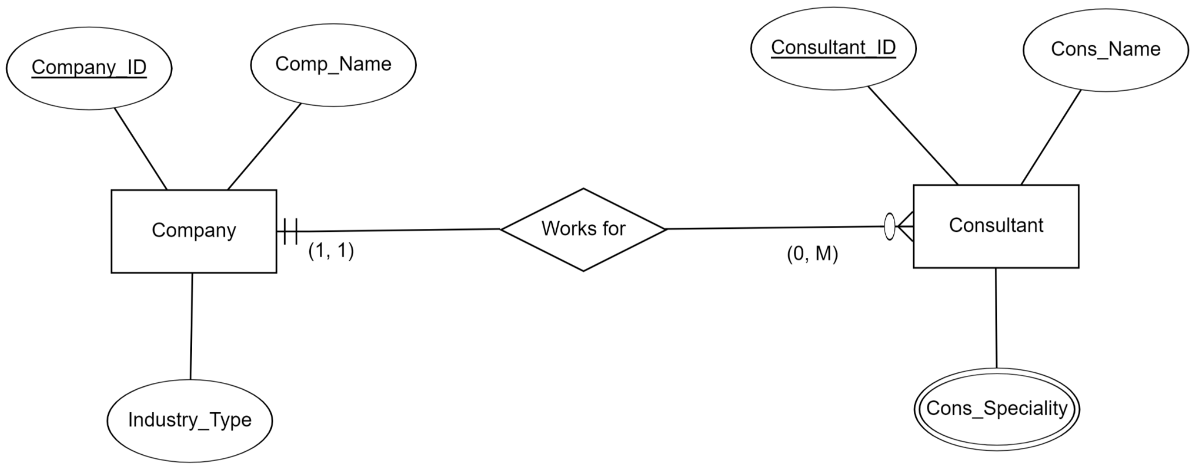
For the Assignment relationship, it is represented as an entity because the company wishes to record the applicable billing rate (Billing_Rate) for each employee when assigned to a particular project. So Billing_Rate becomes an attribute of Assignment.



Question 03:

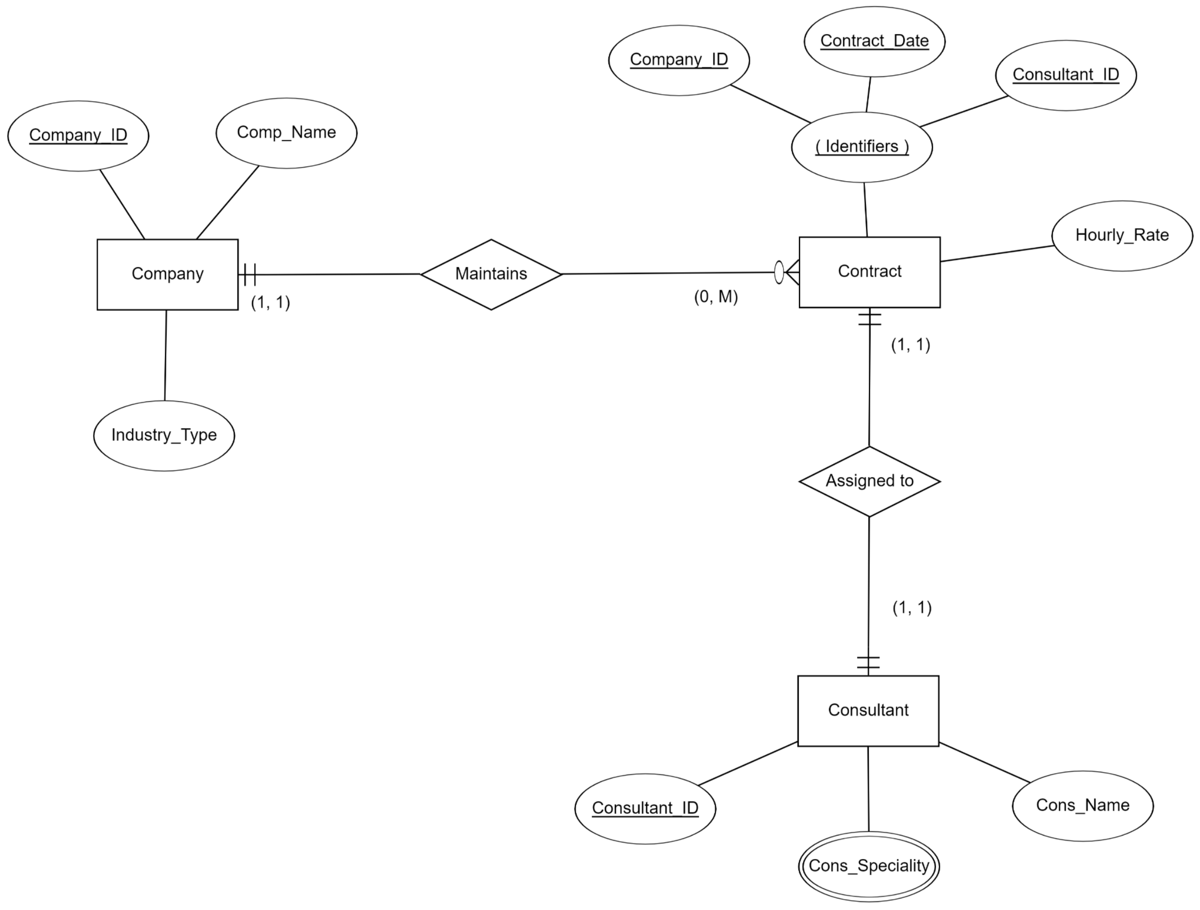
Companies, identified by a Company_ID and described by Company_Name and Industry_Type, hire consultants, identified by Consultant_ID and described by Consultant_Name, Consultant_Speciality, which is multivalued. Assume that a consultant can work for only one company at a time, and we need to track only current consulting engagements. A company can also have multiple consultants working for them, or at times, they have no need of consultants.

(a): Draw an ERD for this Situation:



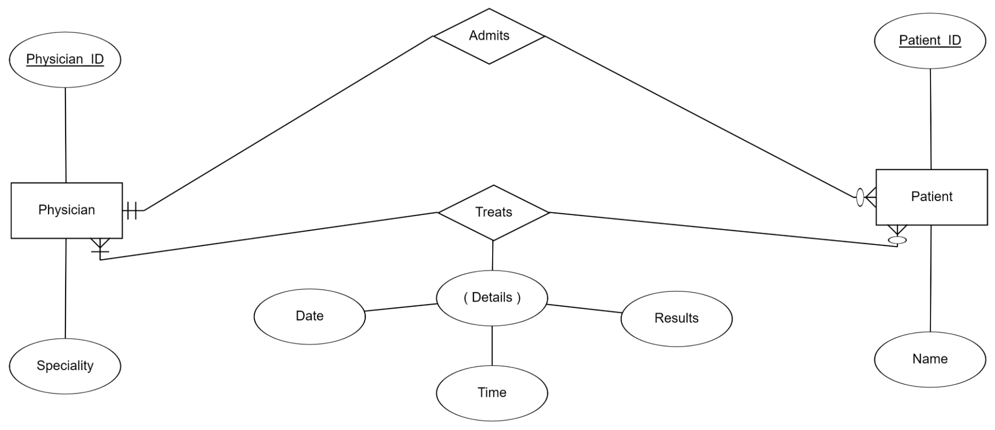
Now, consider that each time a consultant works for a company, a contract is written describing the terms for this consultant engagement. Contract is identified by a composite identifier of Company ID, Consultant ID, and Contract Date. The contract should also include the hourly rate at which the consultant is paid. Assuming the consultant can still work for only one company at a time.

(b): Redraw the ERD for this situation.

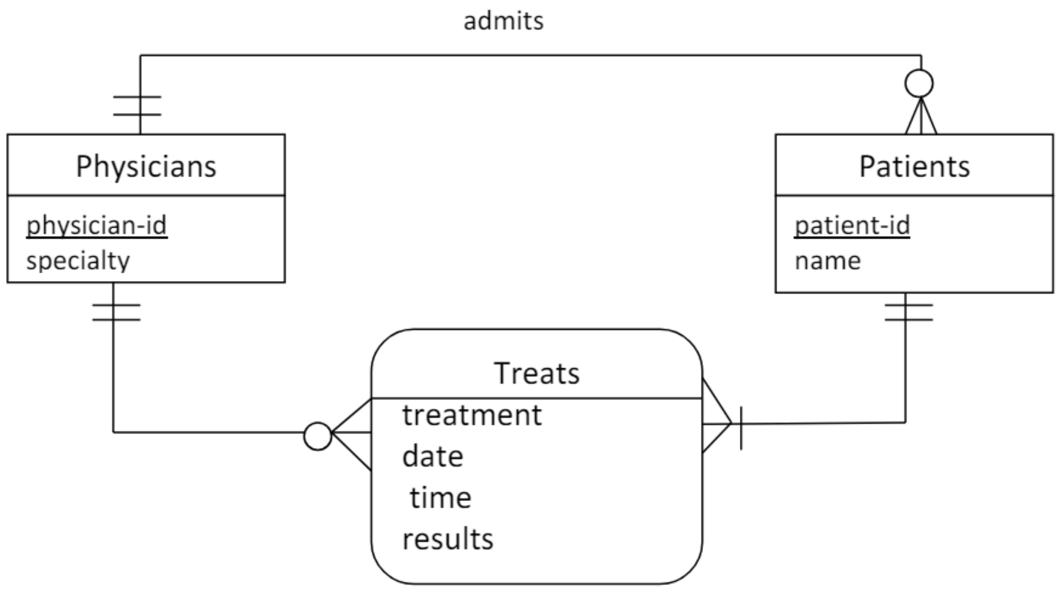


Question 04:

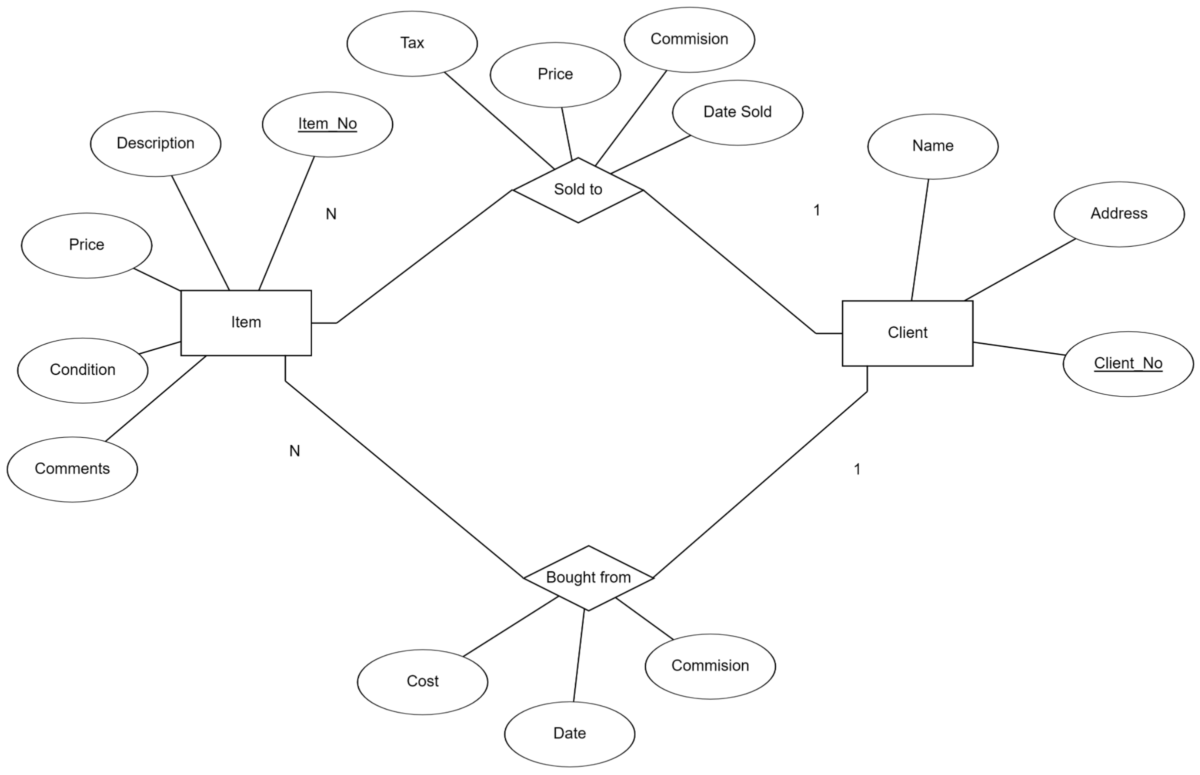
The ERD for the hospital situation consists of three main entity types, PHYSICIAN, PATIENT, and TREATMENT. The relationships between these entities allow for the tracking of multiple admissions and treatments over time. The hospital is not included as an entity as it has no attributes or relationships to be recorded in this model. Moreover, I am assuming that both Physician and Patient have IDs that are Primary Keys (identifiers). Also assuming that, details are Composite



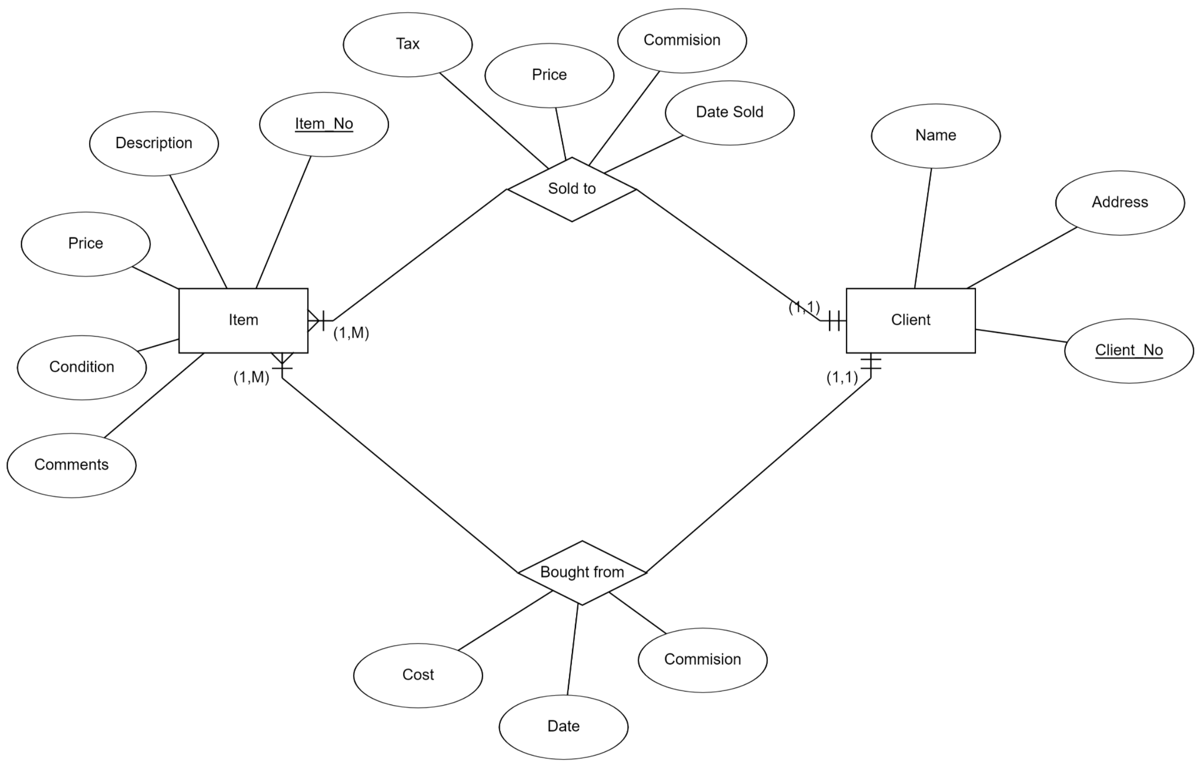
In other Notation we can also draw ERD as:



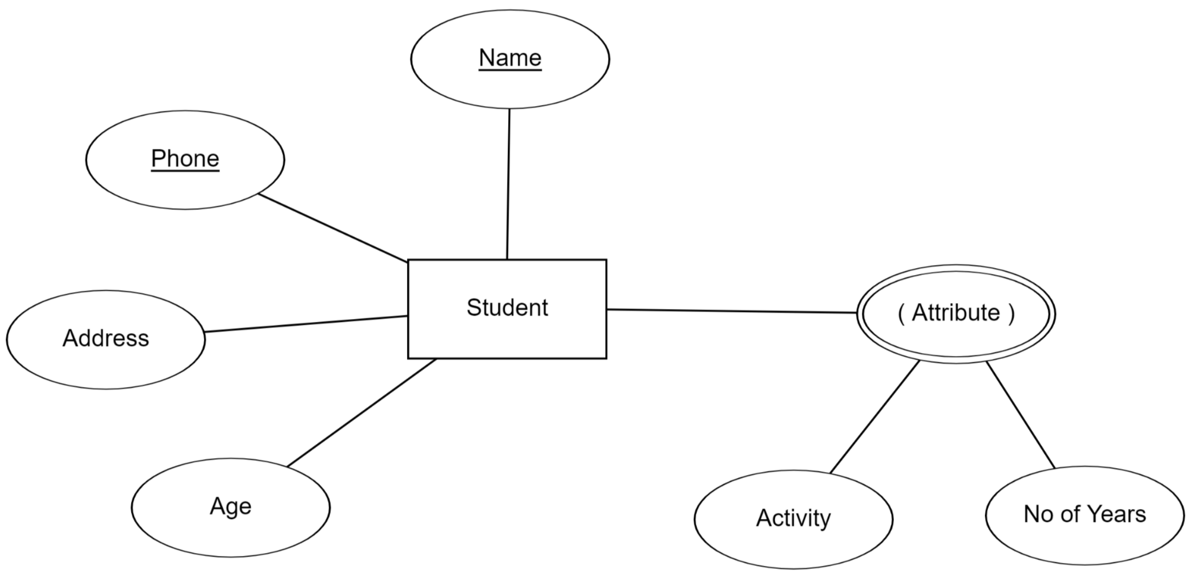
Question 05:



However, if i use the (min, max) notation with the Assumptions that a Client must have a minimum of 1 Item bought or sold and that for Item the minimum is 1 and maximum is M. Then, we can also use the following ERD:



Question 06:



Here I have taken both Name and Phone as a Composite Primary Key (identifier). You can take Name only as an identifier, but then it is possible that 2 people might have the same name. But 2 people with the same Name and Phone (with the same country code) isn't really possible. Moreover, Attribute is both a Multivalued key and Composite satisfying the condition of [Student may engage in more than one activity].

Question 07:

An associative entity is an entity type used for associating one or more entity instances that have a many-to-many relationship. It contains attributes important for all entity instances that are associated with the associative entity.

A good example is an associative entity that associates entity instances of Book and Customer entity types. A new entry is created each time when the Customer borrows a book in the bookstore.

A weak entity is an entity whose existence is dependent on the other (strong) entities. It doesn't have its identifier, ie: it is identified by the primary attribute of the strong entity that is implemented as a foreign key in the weak entity.

So, Associative entities are a type of weak entity because they cannot exist on their own and are dependent on other entities in the many-to-many relationship. The speciality about their weakness is that they are dependent on both entities that are forming the relationship that needed the associative entity to resolve their relationship.

Question 08:

Given Functional Dependencies:

- **StaffID → StaffName**
- **CustomerID → Cust_Name, Cust_Address**
- **Cust_Address → Cust_Phone**
- **OrderID → OrderDate, CustomerID, StaffID**
- **FoodDishID → FoodDishName, UnitPrice**
- **OrderID, FoodDishID → Quantity, QuantityPrice**

The definition of partial dependency is that if a relation has a candidate key that consists of more than one attribute, and if an attribute is dependent on only a part of that candidate key, then such a dependency is referred to as partial dependency. First finding all Candidate Keys:

From above FDs, we can infer that:

- `StaffID` identifies `StaffName`.
- `CustomerID` identifies `Cust_Name` and `Cust_Address`.
- `Cust_Address` identifies `Cust_Phone`.
- `OrderID` identifies `OrderDate`, `CustomerID` and `StaffID`.
- `FoodDishID` identifies `FoodDishName` and `UnitPrice`.
- Combination of `OrderID` and `FoodDishID` identifies `Quantity`, `QuantityPrice`.

According to the rules of Partial Dependency:

1. **Left side of the arrow in a functional dependency:** Must be a proper subset of any candidate key.
2. **Right side of the arrow in a functional dependency:** Must be a non-prime attribute.

This way to Convert the given 1NF in the first Step is to find the Candidate Keys, which in this case are `OrderID` and `FoodDishID`. First checking 1NF, as LHS of 6th Dependency LHS is a proper Subset of both Candidate Keys so it is in a 1NF Form. So to convert to 2NF, we divide the table into 3 tables.

Relation 1: OrderID, OrderDate, CustomerID, StaffID, StaffName, Cust_Name, Cust_Address, Cust_Phone.

- CustomerID → Cust_Address, Cust_Name
- Cust_Address → Cust_Phone
- OrderID → StaffID, CustomerID, OrderDate
- StaffID → StaffName

Relation 2: FoodDishId, FoodDishName, UnitPrice.

- FoodDishId → FoodDishName, UnitPrice

Relation 3: OrderID, FoodDishId, Quantity, QuantityPrice

- FoodDishId, OrderID → Quantity, QuantityPrice

These 3 Relations describe the 2NF Form. Now for 3NF, we remove the Transitive Dependencies. Find the minimal cover of FDs, which contains

- StaffID → StaffName
- CustomerID → Cust_Name
- CustomerID → Cust_Address
- Cust_Address → Cust_Phone
- OrderID → OrderDate
- OrderID → CustomerID
- OrderID → StaffID
- FoodDishId → FoodDishName
- FoodDishId → UnitPrice
- OrderID, FoodDishId → Quantity
- OrderID, FoodDishId → QuantityPrice

Merge FDs with same LHS and whose RHS are non-key attributes, we get the given set which contains:

- StaffID → StaffName
- CustomerID → Cust_Address, Cust_Name
- Cust_Address → Cust_Phone
- OrderID → StaffID, CustomerID, OrderDate
- FoodDishId → UnitPrice, FoodDishName
- OrderID,FoodDishId → QuantityPrice, Quantity

Check each FD in the set F1 for violation of 3NF, and split the table accordingly.

Checking FD [StaffID → StaffName]. The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes). The following 3NF table is obtained:

- StaffID, StaffName with FDs StaffID → StaffName.

Checking FD [CustomerID → Cust_Address, Cust_Name]. The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes). The following 3NF table is obtained:

- CustomerID, Cust_Address, Cust_Name with FDs CustomerID → Cust_Address, Cust_Name.

Checking FD [Cust_Address → Cust_Phone]. The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes). The following 3NF table is obtained:

- Cust_Address, Cust_Phone with FDs Cust_Address → Cust_Phone.

Checking FD [OrderID → StaffID, CustomerID, OrderDate]. The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes). The following 3NF table is obtained:

- OrderID, StaffID, CustomerID, OrderDate with FDs OrderID → CustomerID, StaffID, OrderDate.

Checking FD [FoodDishId → UnitPrice, FoodDishName]. The FD violates 3NF as its LHS is not a superkey (and RHS is a set of non-key attributes). The following 3NF table is obtained:

- FoodDishId, UnitPrice, FoodDishName with FDs FoodDishId → UnitPrice, FoodDishName.

Checking FD [OrderID, FoodDishId → QuantityPrice, Quantity]. FD does not violate 3NF. No change done. Thus the table is now in 3NF. For BCNF, we can check if LHS of the Arrow is a Super Key or not. If it is not, we split the table into 2 or more (only sometimes). The final Relations for BCNF are:

Relation 1: StaffID, StaffName

- StaffID → StaffName

Relation 2: Cust_Address, Cust_Phone

- Cust_Address → Cust_Phone

Relation 3: CustomerID, Cust_Address, Cust_Name

- CustomerID → Cust_Address, Cust_Name

Relation 4: OrderID, CustomerID, StaffID, OrderDate

- OrderID → CustomerID, StaffID, OrderDate

Relation 5: FoodDishId, FoodDishName, UnitPrice

- FoodDishId → FoodDishName, UnitPrice

Relation 6: OrderID, FoodDishId, Quantity, QuantityPrice

- OrderID, FoodDishId → Quantity, QuantityPrice

This final table is in BCNF Form. All requirements are satisfied. You can also check the Ruf Work for this Question at the End of the `Document`.

Question 09:

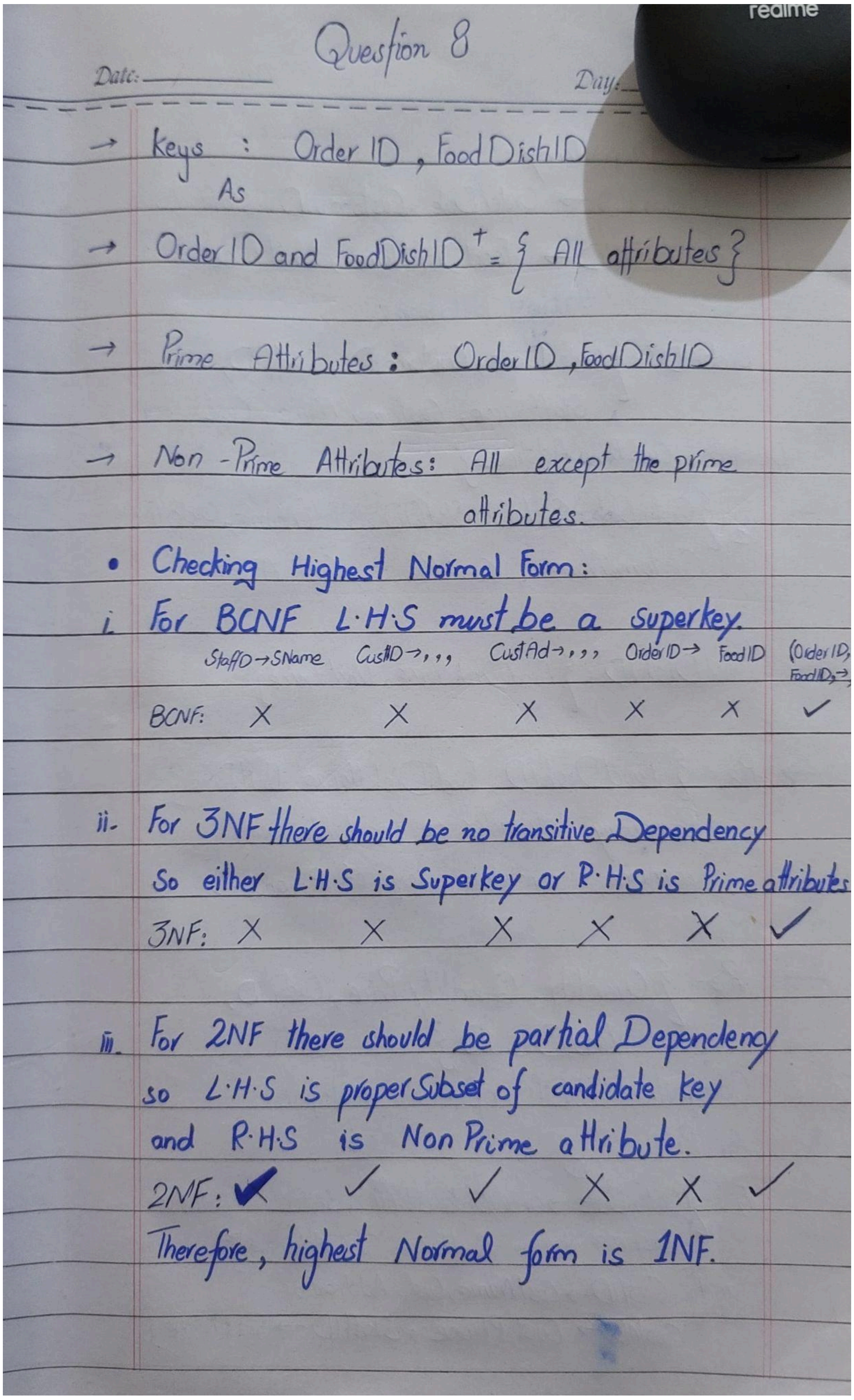
We are given the relation R(ABCDEFGH), and FDs {AB → C, AC → B, AD → E, B → D, BC → A, E → G}. Turning the relation into its lowest form by removing its dependencies, we get ABFH.

Taking the closure of ABFH, we get {A, B, C, D, E, F, G, H}, so it can be called a super key. But, closure of BFH gives {B, D, F, H} which means it is not a candidate key. Similarly, the closure of AFH gives {A, F, H}, thus it is also not a candidate key. Thus, ABFH is one candidate key. Closure of ACFH gives all values in the relation, ie: {A, B, C, D, E, F, G, H}. While, taking closure of AFH and CFH does not give all values in the Relation. So, ACFH is a Candidate key.

Moreover, the closure of BCFH gives all values in the relation mainly {B, C, A, D, E, G, F, H}. So, BCFH is a Candidate key. There are 3 candidate keys: **ABFH, ACFH, BCFH.** Moreover, the Paper written version of this Question is also given at the End of this `Document`. Moreover, the Diagrams in this Assignment have been created using ERDPlus. The Source Files for each Document have been attached to Google Classroom.

NEXT PAGE

Question 08 (Hand Written):



Question 8

Date: ____________ Day: ____

→ Keys : Order ID , Food DishID
As

→ Order ID and FoodDishID$^+$ = { All attributes }

→ Prime Attributes : Order ID , Food DishID

→ Non-Prime Attributes: All except the prime attributes.

• Checking Highest Normal Form:

i. For BCNF L·H·S must be a superkey.

	StaffID→SName	CustID→,,,	CustAd→,,,	OrderID→	FoodID	(OrderID, FoodID)→
BCNF:	X	X	X	X	X	✓

ii. For 3NF there should be no transitive Dependency So either L·H·S is Superkey or R·H·S is Prime attributes

3NF:	X	X	X	X	X	✓

iii. For 2NF there should be partial Dependency so L·H·S is proper Subset of candidate key and R·H·S is Non Prime attribute.

2NF:	✓	✓	✓	X	X	✓

Therefore, highest Normal form is 1NF.

Date.________ Day:________

i Partial Dependency:

a. OrderID → OrderDate, CustomerID, Staff ID

b. FoodDishID → FoodDishName, Unit Price

Making Relations

OrderID$^+$ = {OrderDate, CustomeID, StaffID, StaffName, Customer Name, Cust_OrderID, Cust-Phone, Address}

→ R_1 = {OrderDate, CustID, StaffID, Staff Name, OrderID CustomeName, Cust Phone, Cust_Addres}

FoodDishID$^+$ = {FoodDishName, Unit Price}

→ R_2 = {FoodDishID, Food Dish Name, Unit Price}

- Remaining attributes in one table with common candidate keys of R_1 and R_2.

→ R_3 = {Quantity, Quantity Price, OrderID, FoodDishID}

Computing FD's of each Relation:

For R_1

F_1 : {(Order~~Date~~ ID → OrderDate, StaffID, StaffName, CustID Customer Name, CustPhone, Cust Address)}

and CustID → CustName, Cust_Address,

CustADD → Cust Phone, StaffID → StafName}

Date: ____________ Day: ____________

For R_2 :

F_2 : { FoodDishID → FoodDishName, UnitPrice }

For R_3 :

F_3 : { OrderID, FoodDishID → Quantity, QuantityPrice }

So for R_1

CK = OrderID

Checking normal form,

BCNF: ✓ × × ×
3NF: ✓ × × ×
2NF: ✓ ✓ ✓ ✓

Successfully Converted into 2NF.

So for R_2

CK: FoodDishID

BCNF: ✓
3NF: ✓
2NF: ✓

Successfully converted into BCNF

R_3

CK: OrderID, FoodDishID

BCNF: ✓
3NF: ✓
2NF: ✓

Sucessfully converted in BCNF

Problem is transitive Dependency in R_1 therefore, FDs causing this Problem are,

{ CustID → CustName, Cust-Address
CustAdd → CustPhone
StaffID → StaffName

Making Relations

CustID

CustID$^+$ = {CustName, CustAdd}

R1 { CustID, CustName, Custadd }

Computing FD,

F_1 = { CustID → CustName, Custadd
~~Custadd → Cust~~P }

CK: ID Checking NF

BCNF ✓

CustAdd$^+$ = {add, Phone}

R2 {Custadd, Phone}

F_2 = { Custadd → Cust Phone }

CK: add

BCNF ✓

StaffID → Staff Name

R3 = { StaffID, Name }

F_3 = { StID → Name

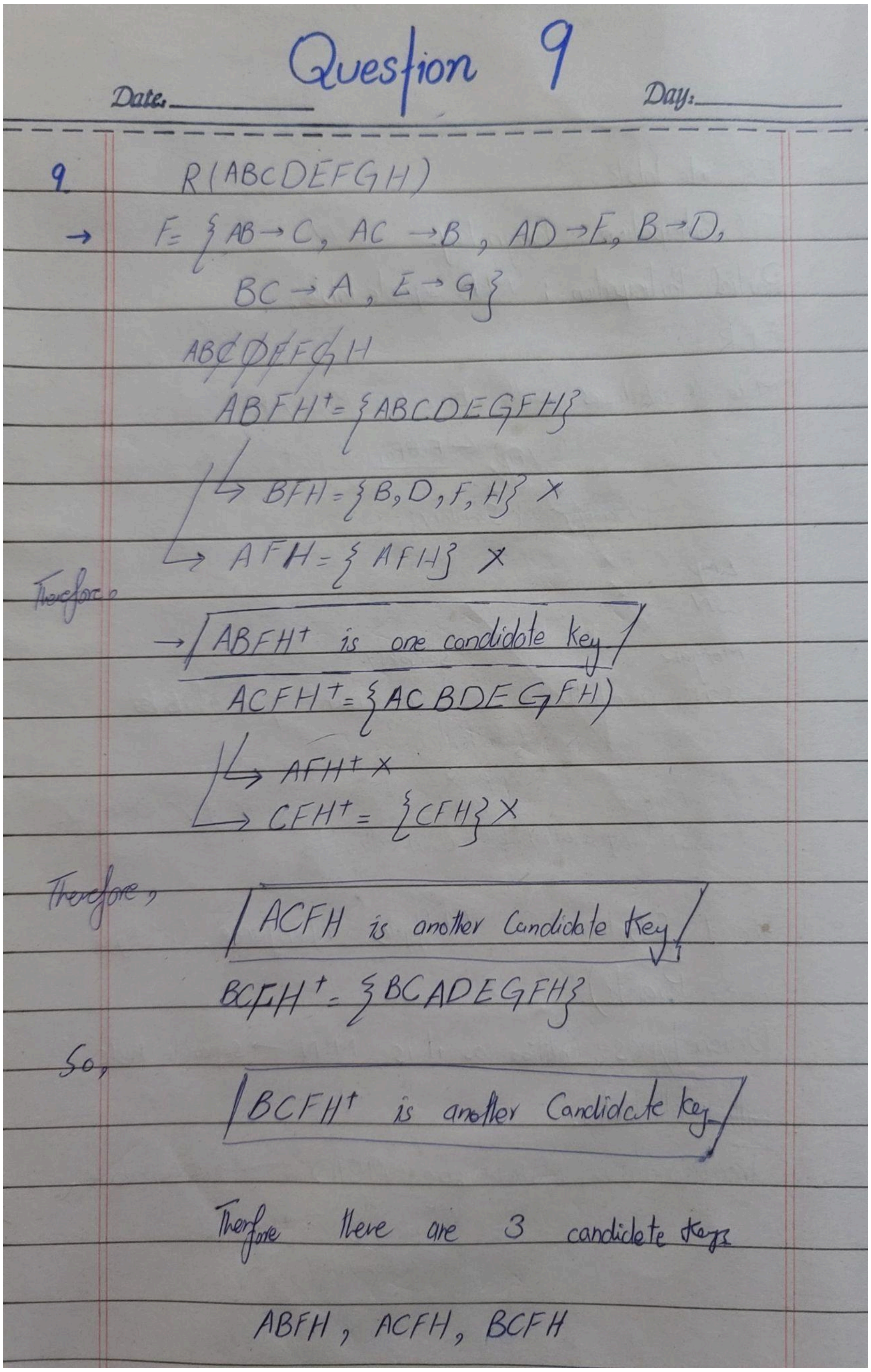
CK: ID **BCN**✓

R4 = { OrderID, Date, Name, CustName, CustPhone, }

F_4 = { OrderID → Date, Name, Phone, CustID }

BCNF ✓

Question 09 (Handwritten):



Question 9

Date: ______ Day: ______

9. R(ABCDEFGH)

→ $F = \{AB \to C,\ AC \to B,\ AD \to E,\ B \to D,\ BC \to A,\ E \to G\}$

AB~~C~~~~D~~~~E~~F~~G~~H

$ABFH^+ = \{ABCDEGFH\}$

↳ $BFH = \{B, D, F, H\}$ ×

↳ $AFH = \{AFH\}$ ×

Therfore,

→ $\boxed{ABFH^+ \text{ is one candidate key}}$

$ACFH^+ = \{ACBDEGFH)$

↳ ~~AFH^+ ×~~

↳ $CFH^+ = \{CFH\}$ ×

Therefore,

$\boxed{ACFH \text{ is another Candidate Key}}$

$BCFH^+ = \{BCADEGFH\}$

So,

$\boxed{BCFH^+ \text{ is another Candidate key}}$

Therfore there are 3 candidate keys

ABFH, ACFH, BCFH