UCS2404 - DATABASE MANAGEMENT SYSTEMS

SAISANDEEP SANGEETHAM - 3122 22 5001 119
SHARANRAJ K - 3122 22 5001 126
SRIVATHSAN S - 3122 22 5001 141

Department of Computer Science and Engineering
Sri Sivasubramaniya Nadar College of Engineering
(An Autonomous Institution, Affiliated to Anna University)
Kalavakkam – 603110

2023-24

Regulations – R2021

Project Report

INDEX:

Problem statement: Pharmacy supply chain Management system

Background

Pharmacy chains often face challenges in managing their inventories and orders effectively. To address these challenges, a comprehensive database management system is essential. This system will enable pharmacies to maintain optimal inventory levels, place orders for restocking, and ensure that customer demands are met without delays. The database will also facilitate the management of relationships between pharmacies, suppliers, and products, ensuring that orders are fulfilled based on real-time inventory data.

Objective

Develop a robust database system to manage the inventory and order processes of a pharmacy chain. The system should track product availability, manage orders placed with suppliers, and ensure that pharmacies can meet customer demands. Additionally, the system must validate stock levels before allowing orders to be placed, ensuring that orders do not exceed available inventory.

Assumptions

- Each pharmacy has its own inventory of products, which may differ from other pharmacies.
- Products are restocked by placing orders with suppliers.
- Orders contain multiple products, and each product's quantity must be specified.
- The total amount of an order is calculated based on the sum of the prices of all ordered products.
- Before placing an order, the system checks if the requested quantity of each product is available in the supplier’s inventory
- Triggers are used to automate stock validation and updates, ensuring that inventory data is always accurate

Project Report

Pharmacy supply chain Management system

Tables and their attributes :

1. **Supplie**r

- Sup_id
- Prod_id
- Sup_name
- Sup_location
- Sup_phone
- Quantity
- Date_mfg
- Date_exp

2. **Pharmacy**

- Pharm_id
- Pharm_name
- phone_no
- Email
- Location
- Pincode
- Prod_id
- In_stock

3. **Pharmacy_Order**

- Pharm_order_id
- Pharm_id
- Sup_id
- Prod_id
- Qty
- Order_date
- total_amount

4. Products

- Prod_id
- Prod_name
- Prod_type
- Unit_price

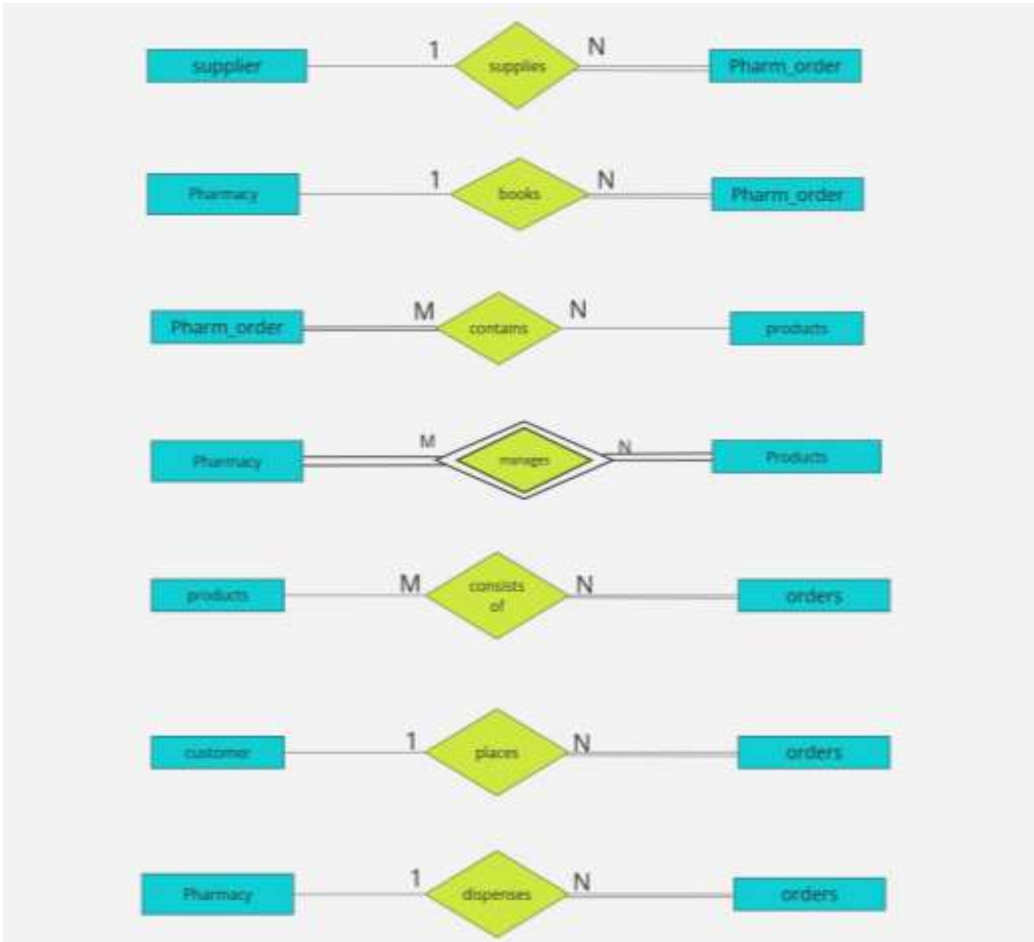
5. Orders

- Order_no
- Cust_id
- Prod_id
- Pharm_id
- Qty
- Order_date
- Total_amount

6. Customers

- Cust_id
- Cust_name
- Location,
- Phone_no
- Age

Relationships With Cardinality Ratio:



Functional Dependencies :

Pharmacy :

- Pharm_id -A
- Pharm_name -B
- Phone_no -C
- Email -D
- Location -E
- Pincode -F

- Opening Hours -G
- Prod_id - H
- Stock_qty - I

Identified functional dependencies:

- A-> B
- A->C
- A->D
- BC->E
- BC->F
- BC->G
- AH->I

Minimal set of Functional Dependencies

1. A->B
   With Without
   A^+ = {A,B,C,D,E,F,G} A^+ = {A,C,D}
   Here both the closure set of attributes of A^+ are not the same. So, we can consider this Functional dependency.
2. A->C
   With Without
   A^+ = {A,B,C,D,E,F,G} A^+ = {A,B,D}
   Here both the closure set of attributes of A^+ are not the same. So, we can consider this Functional dependency.

3. A->D
   With Without
   A^+ = {A,B,C,D,E,F,G} A^+ = {A,B,C,E,F,G}
   Here both the closure set of attributes of $A^+$are not the same. So, We can consider this Functional dependency.

4. BC->E
   With Without
   BC^+ = {B,C,E,F,G} BD^+ = {B,C,F,G}
   Here both the closure set of attributes of $BC^+$are not the same. So, We can consider this Functional dependency.
5. BC->F
   With Without
   BC^+ = {B,C,E,F,G } BC^+ = {B,C,E,G}
   Here both the closure set of attributes of BC^+ are not the same. So, We can consider this Functional dependency.
6. BC->G
   With Without

BC^+ = {B,C,E,F,G } BC^+ = {B,C,E,F}

Here both the closure set of attributes of BC^+ are not the same. So, We can consider this Functional dependency.

7. AH->I

   With   Without

   AH^+ = {A,B,C,D,E,F,G,H,I } AH^+ = {A,B,C,D,E,F,G,H }

   Here both the closure set of attributes of AH^+ are not the same. So, We can consider this Functional dependency.

The Minimal set are:

A-> B
A->C
A->D
BC->E
BC->F
BC->G
AH->I

We can write the above functional dependencies as

A->BCD
BC->EFG
AH->I

Finding the Candidate Keys

$ABCDEFGHI^+$ = {A,B,C,D, E, F,G,H,I}

Decomposing with the FD: AH->I

$ABCDEFGH^+$ ={A,B,C,D, E, F,G,H,I}

Decomposing with the FD: BC->EFG

$ABCDH^+$ ={A,B,C,D, E, F,G,H,I}

Decomposing with the FD: A->BCD

AH^+ ={A,B,C,D, E, F,G,H,I}

Therefore AH^+ includes all the Attributes hence it is a super key, and it is a Candidate Key

Customers

- Cust_id **- A**
- Cust_name **- B**
- Location **- C**
- Phone_no **- D**
- Age **- E**

Identified Functional Dependencies:

- AB -> CDE
- A -> B
- A -> C
- A -> D
- A -> E

Minimal set of Functional Dependencies:

1.AB -> CDE

With:	Without:
$AB^+ = \{A,B,C,D,E\}$	$AB^+ = \{A,B,C,D,E\}$

Here both the closure sets of attributes of AB^+ are the same. This functional dependency is redundant. So, we are not considering this Functional dependency

2. A -> B

With:	Without:
$A^+ = \{A,B,C,D,E\}$	$A^+ = \{A,C,D,E\}$

Here both the closure sets of attributes of A^+ are not the same. So, We can consider this Functional dependency

3. A -> C

With:	Without:
$A^+ = \{A,B,C,D,E\}$	$A^+ = \{A,B,D,E\}$

Here both the closure sets of attributes of A^+ are not the same. So, we can consider this Functional dependency.

4. A -> D

With:	Without:
$A^+ = \{A, B,C,D,E\}$	$A^+ = \{A,B,C,E\}$

Here both the closure sets of attributes of A^+ are not the same. So, we can consider this Functional dependency.

5. A -> E

With: A^+ = {A, B,C,D,E}

Without: A^+ = {A,B,C,D}

Here both the closure sets of attributes of A^+ are not the same. So, we can consider this Functional dependency.

Identified functional dependencies:

A->B
A->C
A->D
A->E

Finding Candidate Keys

$ABCDE^+$ ={A,B,C,D,E}

Decomposing with the A -> B
$ACDE^+$ ={A,B,C,D,E}

Decomposing with the A -> C
ADE^+ ={A,B,C,D,E}

Decomposing with the A -> D
AE^+ ={A,B,C,D,E}

Decomposing with the A -> E
A^+ ={A,B,C,D,E}

Therefore A^+ includes all the Attributes hence it is a super key.
Hence A^+ is the Candidate Key in the table. Which uniquely identifies all other attributes in the relation.

Product:

- Prod_id **-A**
- Prod_name **-B**
- Type **-C**
- Unit_price **-D**

Identified Functional Dependencies:

- AB -> CD
- A -> B
- A -> C

- A->D

Minimal set of Functional Dependencies :

1.AB -> CD

With: $AB^+ = \{A, B, C, D\}$

Without: $AB^+ = \{A,B,C,D\}$

Here both the closure sets of attributes of A^+ are the same. This is the redundant functional dependency. So, We are not considering this Functional dependency.

2.A -> B

With: $A^+ = \{A,B,C,D\}$

Without: $A^+ = \{A,C,D\}$

Here both the closure sets of attributes of A^+ are not the same. So, We are considering this Functional dependency.

3.A -> C

With: $A^+ = \{A,B,C,D\}$

Without: $A^+ = \{A,B,D\}$

Here both the closure sets of attributes of A^+ are not the same. So, We are considering this Functional dependency.

4.A -> D

With: $A^+ = \{A,B,C,D\}$

Without: $A^+ = \{A,B,C\}$

Here both the closure sets of attributes of A^+ are not the same. So, We are considering this Functional dependency.

So the minimal set of functional dependencies:

- A->B
- A->C
- A->D

We can write the above Functional dependency as A->BCD

Finding Candidate Keys

$ABCD^+ =\{A,B,C,D\}$

Decomposing with the A -> B

$ACD^+ =\{A,B,C,D\}$

Decomposing with the A -> C

AD^+ ={A,B,C,D}

Decomposing with the A -> D

A^+ ={A,B,C,D}

Therefore A^+ includes all the Attributes hence it is a super key. Let's check the subset of A includes a super key or not.

Closure of A

A^+ = {A,C,B,D}------------------ super key

Hence A^+ is the Candidate Key in the table. Which uniquely identifies all other attributes in the relation.

Pharmacy_Order(A,B,C,D,E,F,G) :

a. Pharm_order_id **- A**
b. Pharm_id **- B**
c. Sup_id **- C**
d. Prod_id **- D**
e. Qty **- E**
f. Order_date **- F**
g. Total_amount **- G**

Identified Functional Dependencies :

- A-> B
- A->F
- A->G
- A->C
- AD->E

Minimal set of Functional Dependencies :

1. A->B

With:	Without:
A^+ = {A,B,F,G,C}	A^+ = {A,F,G,C}

   Here the both closure set of attributes of A^+ are not the same. So, we can consider this Functional dependency.

2. A->F

With:	Without:
A^+ = {A,B,F,G,C}	A^+ = {A,B,G,C}

   Here both the closure set of attributes of A^+ are not the same. So, We can consider this Functional dependency.

3. A->G

With: Without:

A^+ = {A,B,F,G,C} A^+ = {A,B,F,C}

Here both the closure set of attributes of A^+ are not the same. So, we can consider this Functional dependency.

4. A->C

   With: Without:

   A^+ = {A,B,F,G,C} A^+ = {A,B,F,G}

   Here both the closure set of attributes of A^+ are not the same. So, We can consider this Functional dependency.

5. AD->E

   With: Without:

   AD^+ = {A,D,E,B,F,G,C} AD^+ = {A,D,B,F,G,C}

   Here both the closure set of attributes of AD^+ are not the same. So, We can consider this Functional dependency.

So the minimal set of functional dependencies :

- A->B
- A->F
- A->G
- A->C
- AD->E

We can write the above Functional dependency as A->BFGC. And AD->E

Finding Candidate Keys :

$ABCDEFG^+$ ={A,B,C,D,E,F,G}

Decomposing with the FD A->B:

$ACDEFG^+$ ={A,B,C,D,E,F,G}

Decomposing with the FD A->F :

$ACDEG^+$ ={A,B,C,D,E,F,G}

Decomposing with the FD A->G :

$ACDE^{+}$ ={A,B,C,D,E,F,G}

Decomposing with the FD A->C :

ADE^{+} ={A,B,C,D,E,F,G}

Decomposing with the FD AD->E:

AD^{+} ={A,B,C,D,E,F,G}

Therefore AD^{+} includes all the Attributes hence it is a super key. Let's check the subset of AC includes a super key or not.

The subsets of AD^{+} = {A,D}

Closure of A

A^{+} = {A,B,F,G,C} —--------not a super key

Closure of D

D^{+} = {D} —--------------------- not a super key

Hence AD^{+} is the Candidate Key in the table. Which uniquely identifies all other attributes in the relation.

Orders

- Order_no **- A**
- Cust_id **- B**
- Prod_id **- C**
- Pharm_id **- D**
- Qty **- E**
- Order_date **- F**
- Total_amount **- G**

Identified Functional Dependencies:

- A-> B
- A->D
- A->F
- A->G
- AC->E

Minimal set of Functional Dependencies:

1. A->B

With: Without:

$A^+ = \{A,B,D,F,G\}$ $A^+ = \{A,D,F,G\}$

Here both closure set of attributes of A^+ are not the same. So, we can consider this Functional dependency.

2. A->D

   With: Without:

   $A^+ = \{A,D,B,F,G\}$ $A^+ = \{A,B,F,G\}$

   Here both the closure set of attributes of A^+ are not the same. So, We can consider this Functional dependency.

3. A->F

   With: Without:

   $A^+ = \{A,F,B,D,G\}$ $A^+ = \{A,B,D,G\}$

   Here both the closure set of attributes of A^+ are not same. So, We can consider this Functional dependency.

4. A->G

   With: Without:

   $A^+ = \{A,G,B,D,F\}$ $A^+ = \{A,B,D,F\}$

   Here both the closure set of attributes of A^+ are not same. So, We can consider this Functional dependency.

5. AC->E

   With: Without:

   $AC^+ = \{A,C,E,B,D,F,G\}$ $AC^+ = \{A,C,D,B,F,G\}$

   Here both the closure set of attributes of AC^+ are not same. So, We can consider this Functional dependency.

So the minimal set of functional dependencies :

- A->B
- A->D
- A->F
- A->G
- AC->E

We can write the above Functional dependency as A->BDFG. And AC->E

Finding Candidate Keys

$ABCDEFG^+ = \{A,B,C,D,E,F,G\}$

Decomposing with the FD A->B

$ACDEFG^+ = \{A,B,C,D,E,F,G\}$

Decomposing with the FD A->D

$ACEFG^+ = \{A,B,C,D,E,F,G\}$

Decomposing with the FD A->F

$ACG^+ = \{A,B,C,D,E,F,G\}$

Decomposing with the FD A->G

$ACE^+ = \{A,B,C,D,E,F,G\}$

Decomposing with the FD AC->E

$AC^+ = \{A,B,C,D,E,F,G\}$

Therefore AC^+ includes all the Attributes hence it is a super key. Let's check if the subset of AC includes a super key or not.

The subsets of $AC^+ = \{A,C\}$

Closure of A

$A^+ = \{A,B,D,F,G\}$ ----------not a super key

Closure of C

$C^+ = \{C\}$ ---------------------- not a super key

Hence AC^+ is the Candidate Key in the table. Which uniquely identifies all other attributes in the relation.

Supplier(A,B,C,D,E,F,G,H) :

- Sup_id **- A**
- Prod_id **- B**
- Sup_name **- C**
- Sup_location **- D**
- Sup_phone **- E**
- Quantity **- F**
- Date_mfg **- G**
- Date_exp **- H**

Identified Functional Dependencies:

- A->C
- A->D
- A->E
- AB->F
- AB->G

- AB->H

Minimal set of Functional Dependencies:

6. A->C
   With: $A^+ = \{A,C,D,E\}$
   Without: $A^+ = \{A,D,E\}$

   Here both the closure set of attributes of A^+ are not the same. So, we can consider this Functional dependency.

7. A->D
   With: $A^+ = \{A,D,C,E\}$
   Without: $A^+ = \{A,C,E\}$

   Here both the closure set of attributes of $A^+$are not the same. So, We can consider this Functional dependency.

8. A->E
   With: $A^+ = \{A,E,D,C\}$
   Without: $A^+ = \{A,D,C\}$

   Here both the closure set of attributes of $A^+$are not the same. So, We can consider this Functional dependency.

9. AB->F
   With: $AB^+ = \{A,B,F,G,H,C,D,E\}$
   Without: $AB^+ = \{A,B,G,H,C,D,E\}$

   Here both the closure set of attributes of AB^+ are not the same. So, We can consider this Functional dependency.

10. AB->G
    With: $AB^+ = \{A,B,F,G,H,C,D,E\}$
    Without: $AB^+ = \{A,B,F,H,C,D,E\}$

    Here both the closure set of attributes of AB^+ are not the same. So, We can consider this Functional dependency.

11. AB->H
    With: $AB^+ = \{A,B,H,F,G,C,D,E\}$
    Without: $AB^+ = \{A,B,F,G,C,D,E\}$

Here both the closure set of attributes of AB^+ are not the same. So, We can consider this Functional dependency.

So the minimal set of functional dependencies:

- A->C
- A->D
- A->E
- AB->F
- AB->G
- AB->H

We can write the above Functional dependencies as

1. A->CDE
2. AB->FGH.

Finding Candidate Keys:

$ABCDEFGH^+$ ={A,B,C,D,E,F,G,H}

Decomposing with the FD A->CDE

$ABFGH^+$ ={A,B,C,D,E,F,G,H}

Decomposing with the FD AB->FGH

AB^+ ={A,B,C,D,E,F,G,H}

Therefore AB^+ includes all the Attributes hence it is a super key. Let's check the subset of AB includes a super key or not.

The subsets of AB^+ = {A,B}

Closure of A

A^+ = {A,C,D,E} —--------not a super key

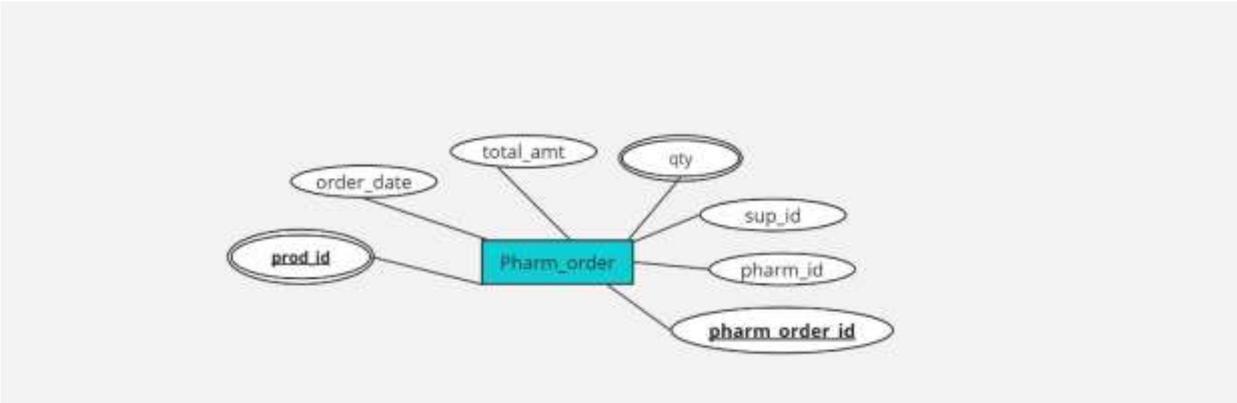
Closure of C

B^+ = {B} —--------------------- not a super key

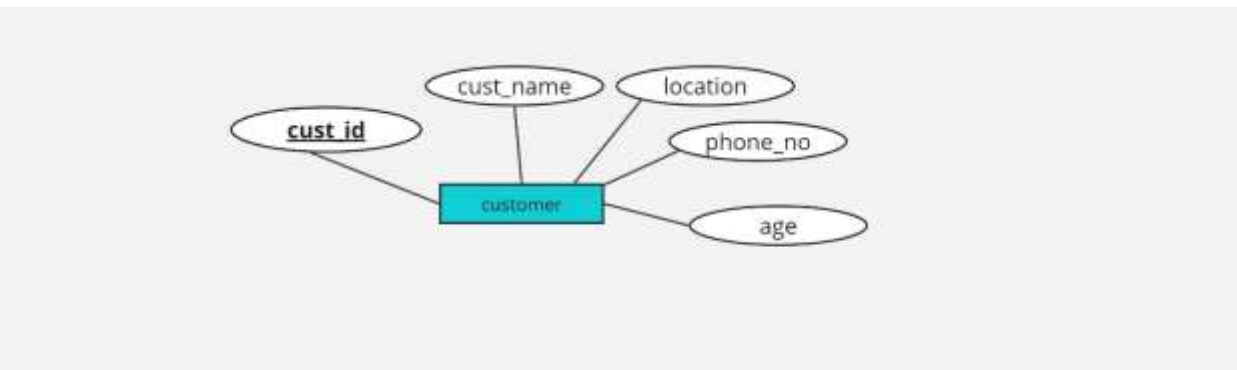
Hence AB^+ is the Candidate Key in the table. Which uniquely identifies all other attributes in the relation.

ER DIAGRAM:

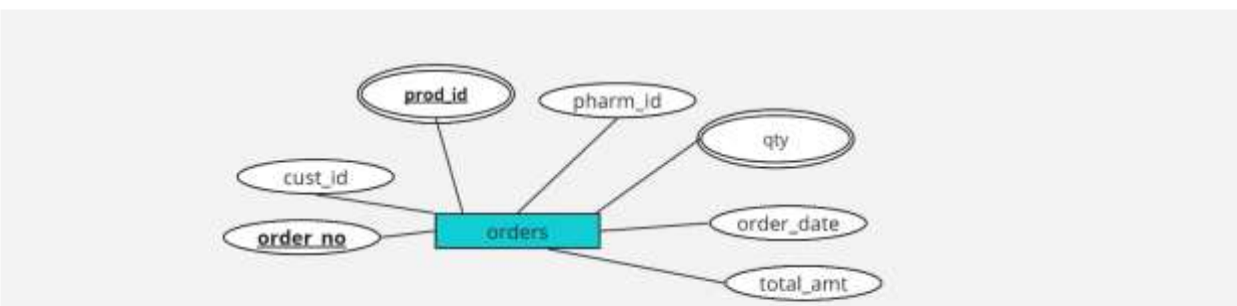
Pharmacy_Order:



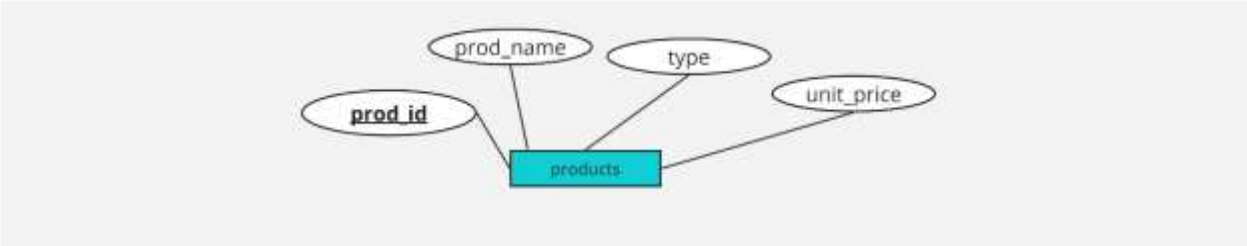
Customer:



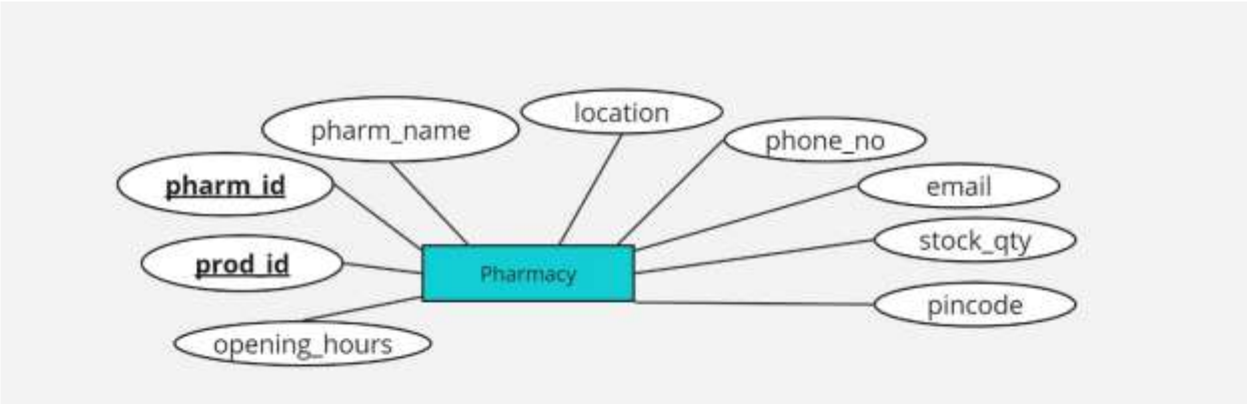
Orders:



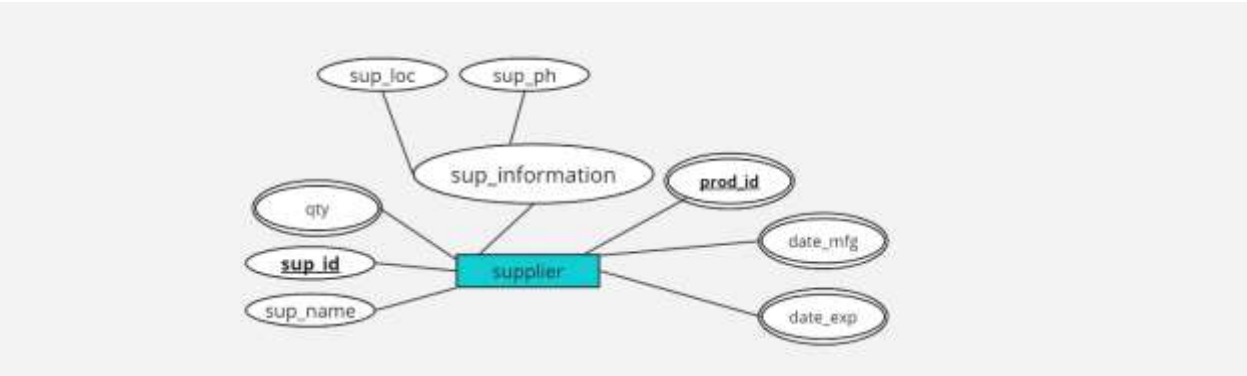
Products:



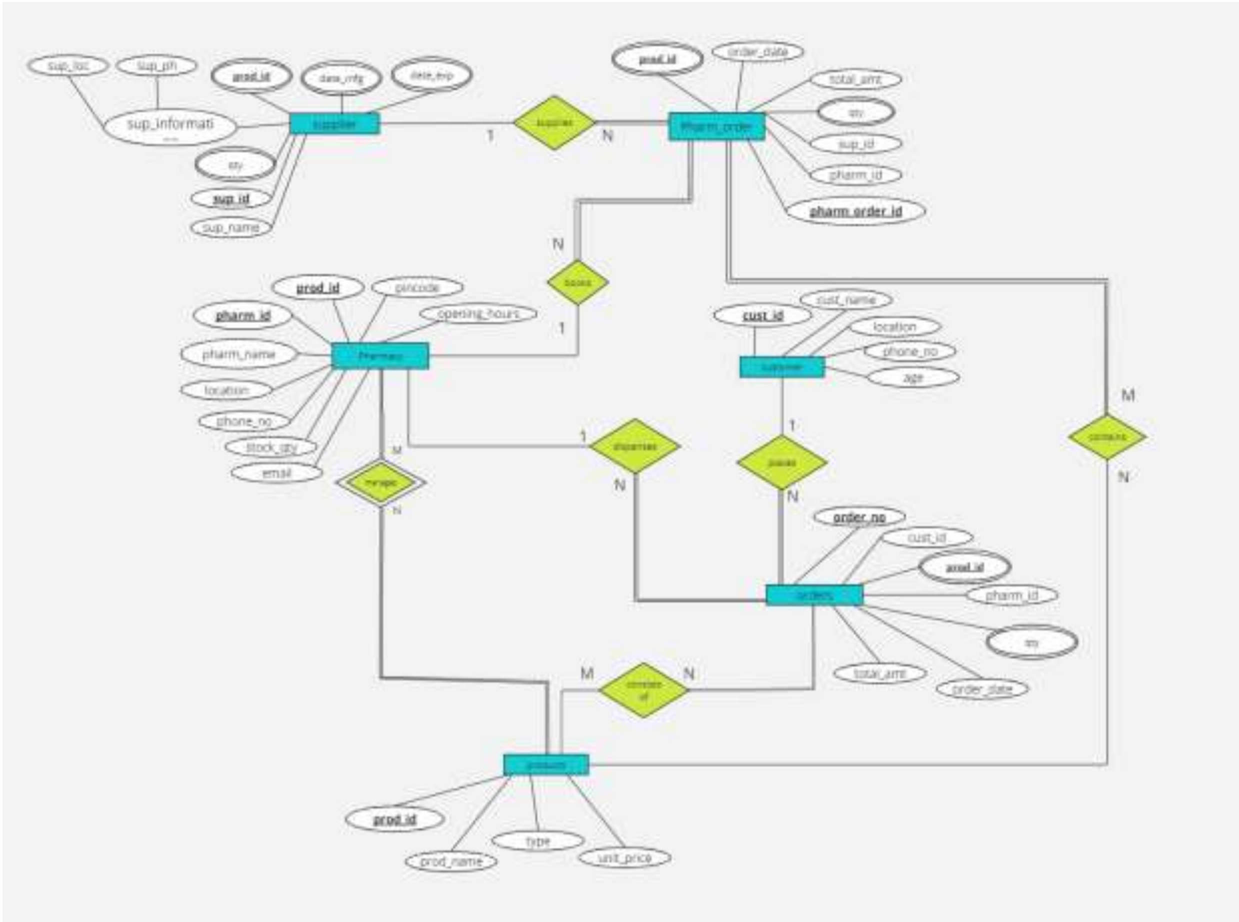
Pharmacy:



Supplier:



ER-DIAGRAM:



sup_loc
sup_ph
sup_informati
prod_id
date_mfg
sup_id
sup_name
1
N
prod_id
order_date
total_amt
sup_id
pharm_id
pharm_order_id
N
1
prod_id
pincode
pharm_id
opening_hours
pharm_name
location
phone_no
stock_qty
email
M
N
1
N
cust_id
cust_name
location
phone_no
age
1
N
M
N
order_no
cust_id
prod_id
pharm_id
total_amt
order_date
M
N
prod_id
type
prod_name
unit_price

Pharmacy supply chain
Management system

Tables and their attributes :

1. **Supplier** :

- Sup_id
- Prod_id
- Sup_name
- Sup_location
- Sup_phone
- Quantity
- Date_mfg
- Date_exp

2. **Pharmacy :**

- Pharm_id
- Pharm_name
- Location
- phone_no
- Prod_id
- In_stock

3. **Pharmacy_Order :**

- Pharm_order_id
- Pharm_id
- Sup_id
- Prod_id
- Qty
- Order_date
- total_amount

4. **Products :**

- Prod_id
- Prod_name
- Prod_type
- Unit_price

5. **Orders :**

- Order_no
- Cust_id
- Prod_id

- Pharm_id
- Qty
- Order_date
- Total_amount

6. **Customers :**

- Cust_id
- Cust_name
- Location,
- Phone_no
- Age

Functional Dependencies :

<u>Customers</u>

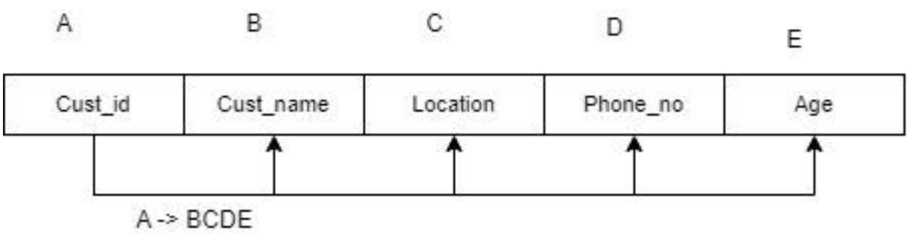
- Cust_id **- A**
- Cust_name **- B**
- Location **- C**
- Phone_no **- D**
- Age **- E**

Identified Functional Dependencies:

- AB -> CDE
- BD -> C
- A -> B
- A -> C
- A -> D
- A -> E

From the previous pdf we can see the minimal set of functional dependencies are:

A->B
A->C
A->D
A->E



From evaluating with the minimal set of FD's we arrive that the cust_id(A) is the only one candidate key, and It is the primary key.

Checking for the 1NF:

Conditions need to be satisfied are:

- Each column is a single valued attribute.
- Each attribute has a value related to the domain.
- Each attribute has a unique name.
- No rows get duplicated.

So, it is in 1NF.

Checking for the 2NF:

- In the 2NF, relation must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key. In simple words there shouldn't present any partial functional dependencies.

Prime Attributes: { }

Non-Prime Attributes: {B, C, D, E}

As we see that there is no prime attributes for the table hence the relation is already in 2NF.

Checking for the 3NF:

- A relation will be in 3NF if it is in 2 NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in 3NF.

Since there exists no transitive partial dependency in the above functional dependencies.

The relation is already in 3NF.

Checking for the BCNF:

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X->Y, X is the super key of the table.
- For BCN, the table should be in 3NF, and for every FD, LHS is super key.

Functional Dependencies:

- A->B
- A->C
- A->D
- A->E

We can write it as A->BCDE

Candidate Key : A^+

- Here the LHS of the candidate key is a super key.
- So, it is in BCNF.

Hence by the functional dependencies we that the cust_id(A) is the super key, candidate key as well as primary key. It also satisfies the BCNF also.

Pharmacy_Order(A,B,C,D,E,F,G) :

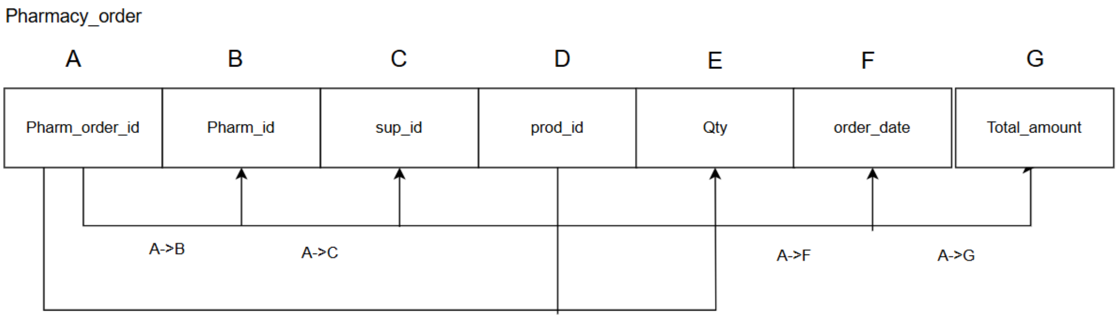
a. Pharm_order_id **- A**
b. Pharm_id **- B**
c. Sup_id **- C**
d. Prod_id **- D**
e. Qty **- E**
f. Order_date **- F**
g. Total_amount **- G**

Identified Functional Dependencies:

- A-> B
- A->F
- A->G
- A->C
- AD->E

From the Previous PDF we can see the minimal set of functional dependencies for the table is:

- A->B - FD 1
- A->F – FD 2
- A->G – FD 3
- A->C – FD 4
- AD->E – FD 5



Checking for 1NF:

Conditions need to be satisfied are:

- Each column is a single valued attribute.
- Each attribute has a value related to the domain.
- Each attribute has a unique name.
- No rows get duplicated.

It violates each attribute has a single value,

So, it is not in 1NF.

Converting it into 1NF :

Splitting the table into :

R1

Pharm_order_id(A)	Pharm_id(B)	sup_id(C)	**prod_id(D)**	Qty(E)	order_date(F)	Total_amount(G)

R11

Pharm_order_id(A)	Pharm_id(B)	sup_id(C)	order_date(F)	Total_amount(G)

R12

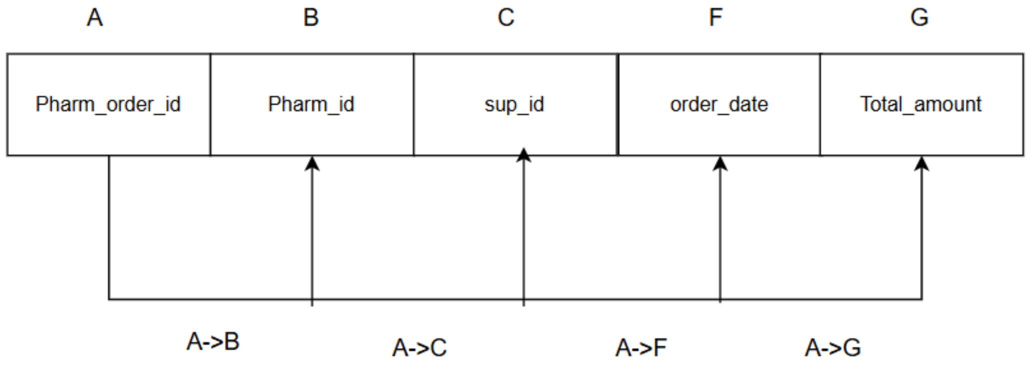
pharm_order_id(A)	prod_id(D)	quantity(E)

So, Both the tables R11, R12 are in 1NF

Checking for 1NF in R11:

- Each attribute has a value related to the domain.
- Each attribute has a unique name.
- No rows get duplicated.
- No rows get duplicated.

pharm_order



Finding the candidate Key:

$ABCFG^+ = \{A, B, C, F, G\}$

Decomposing with the A -> B

$ACFG^+ = \{A, B, C, F, G\}$

Decomposing with the A -> C

$AFG^+ = \{A, B, C, F, G\}$

Decomposing with the A -> F

$AG^+ = \{A, B, C, F, G\}$

Decomposing with the A -> G

$A^+ = \{A, B, C, F, G\}$

Therefore A^+ includes all the Attributes hence it is a super key.
Hence A^+ is the Candidate Key in the table. Which uniquely identifies all other attributes in the relation.

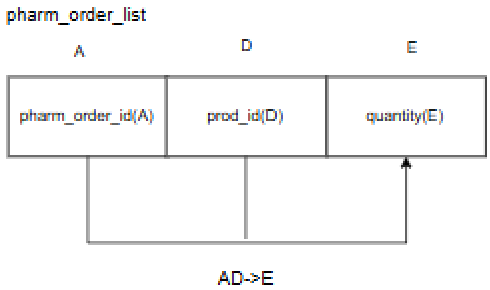
Prime Attribute: {A}
Non-Prime Attribute: {B, C, F, G}

Hence, we see that there are no partial dependencies in the FD's of the R11

In R12(A, D, E)

Set of functional dependencies:

AD->E



Finding the candidate Key:

ADE^+ = {A, D, E}

Decomposing with the AD -> E

AD^+ ={A, D, E}

Therefore AD^+ includes all the Attributes hence it is a super key. Let's check if the subset of the AD includes a super key or not.

Closure of A^+

A^+ = {A} ----------not a super key

Closure of D^+

D^+ = {D} ---------not a super key

Hence (pharm_order_id, prod_id)AD^+ is the Candidate Key in the table. Which uniquely identifies all other attributes in the relation.

Prime Attribute: {A, D}
Non-Prime Attribute: {E}

Hence, we see that there are no partial dependencies in the FD's of the R12

Therefore, the relation is in 2NF

Checking for the 2NF:

For R11

the fd’s are

- ∉ A-> B
- ∉ A->F
- ∉ A->G
- ∉ A->C

- In the 2NF, relation must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key. In simple words there shouldn’t present any partial functional dependencies.

Prime Attributes: {A}

Non-Prime Attributes: {B, F ,G ,C}

There is no proper subset for the prime attributes.

Hence the relation is already in 2NF.

For R12

FD’s are

AD -> E

Candidate Key : AD

Prime Attributes : {A, D}

Non prime Attributes : {E}

- ∉ There is no proper subset.
- ∉ So, It is in 2NF.

Checking for 3NF:

- A relation will be in 3NF if it is in 2 NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in 3NF.

Since there exists no transitive partial dependency in the above functional dependencies.

The relation is already in 3NF

Checking for the BCNF:

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X->Y, X is the super key of the table.

- For BCN, the table should be in 3NF, and for every FD, LHS is super key.

It also satisfies the BCNF.

Checking BCNF for R11(A, B, C, F, G) :

Functional Dependencies:

A->B
A->C
A->F
A->G

We can write it as A->BCFG

Candidate Key : A^+

- Here the LHS of the candidate key is a super key.
- So, it is in BCNF.

Checking BCNF for R12(A, D, E) :

Functional Dependencies:

- AD->E

Candidate Key : AD^+

- Here the LHS of the candidate key is a super key.
- So, it is in BCNF.

Product(A,B,C,D) :

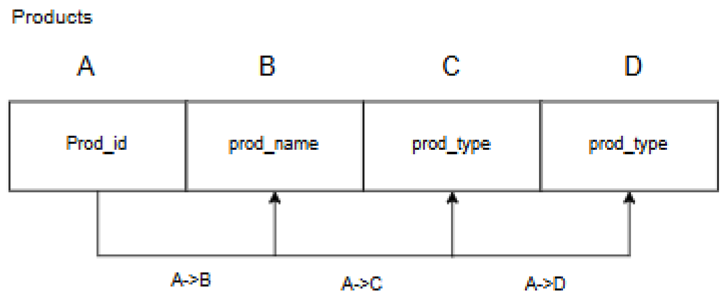
- Prod_id **-A**
- Prod_name **-B**
- Type **-C**
- Unit_price **-D**

Identified Functional Dependencies:

- AB -> CD
- A -> B
- A -> C
- A->D

From the previous pdf we can see the minimal set of functional dependencies are:

- A -> B
- A -> C
- A -> D



From evaluating with the minimal set of FD's we arrive that the prod_id(A) is the only one candidate key, and It is the primary key.

Checking for the 1NF:

Conditions need to be satisfied are:

- Each column is a single valued attribute.
- Each attribute has a value related to the domain.
- Each attribute has a unique name.
- No rows get duplicated.

So, it is in 1NF.

Checking for the 2NF:

- In the 2NF, relation must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key. In simple words there shouldn't present any partial functional dependencies.

Prime Attributes: {A}
Non-Prime Attributes: {B,C,D}

There is no proper subset for the prime attributes.

Hence the relation is already in 2NF.

Checking for the 3NF:

- A relation will be in 3NF if it is in 2 NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in 3NF.

Violating Functional Dependencies : { }
Prime Attribute: { }
Non-Prime Attribute: { }

- So,it is in 3NF.

Checking for the BCNF:

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X->Y, X is the super key of the table.
- For BCN, the table should be in 3NF, and for every FD, LHS is super key.

Checking for BCNF :

Candidate Key : A^+
FD's :

- A -> B
- A -> C
- A -> D

- Here the LHS of the candidate key is a super key.
- So, it is in BCNF.

Pharmacy

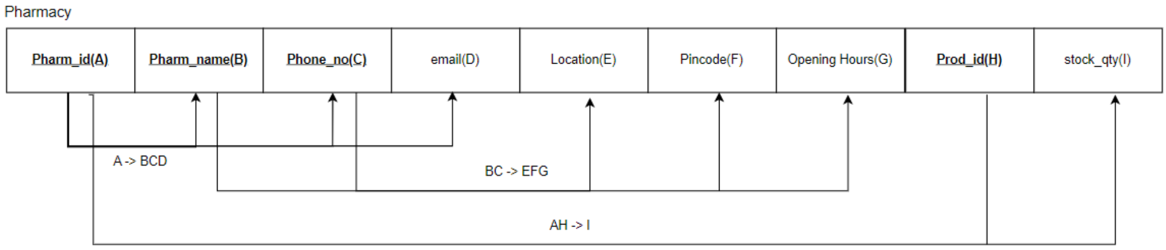
- Pharm_id -A
- Pharm_name -B
- Phone_no -C
- Email -D
- Location -E
- Pincode -F
- Opening Hours -G
- Prod_id - H**(multi valued Attribute)**
- Stock_qty - I **(multi valued Attribute)**

Identified Functional Dependencies:

- A-> B
- A->C
- A->D
- BC->E
- BC->F
- BC->G
- AH->I

From the previous pdf we can see the minimal set of functional dependencies are:

- **A->BCD**
- **BC->EFG**
- **AH->I**



From evaluating with the minimal set of FD's we arrive that the pharm_id,Prod_id(A,H) is the only one candidate key, and It is the primary key.

Checking for the 1NF:

Conditions need to be satisfied are:

- Each column is a single valued attribute.
- Each attribute has a value related to the domain.
- Each attribute has a unique name.
- No rows get duplicated.

The above table violates, each column is a single valued attribute.

Converting it into 1NF :

Splitting the table into :

1 NF for R11 :

- Each attribute has a value related to the domain.
- Each attribute has a unique name.
- No rows get duplicated.
- No rows get duplicated.

1 NF for R12 :

- Each attribute has a value related to the domain.
- Each attribute has a unique name.

- No rows get duplicated.
- No rows get duplicated.

Finding candidate key for R11(A,B,C,D,E,F,G) :

FD's :

- A->B
- A->C
- A->D
- BC->E
- BC->F
- BC->G

Finding candidate key:

A->BCD
BC->EFG

Finding the Candidate Keys

$ABCDEFG^+$ = {A,B,C,D,E,F,G}

Decomposing with the FD: BC->EFG

$ABCD^+$ ={A,B,C,D,E,F,G}

Decomposing with the FD: A->B

ACD^+ ={A,B,C,D,E,F,G}

Decomposing with the FD: A->C

AD^+ ={A,B,C,D,E,F,G}

Decomposing with the FD: A->D

A^+ ={A,B,C,D,E,F,G}

Therefore A^+ includes all the Attributes hence it is a super key, and it is a Candidate Key

Candidate Key : A^+

Finding candidate key for R12(A, H, I) :

FD's :

- **AH->I**

AHI^+ = {A, H, I}
AH^+= {A, H, I}

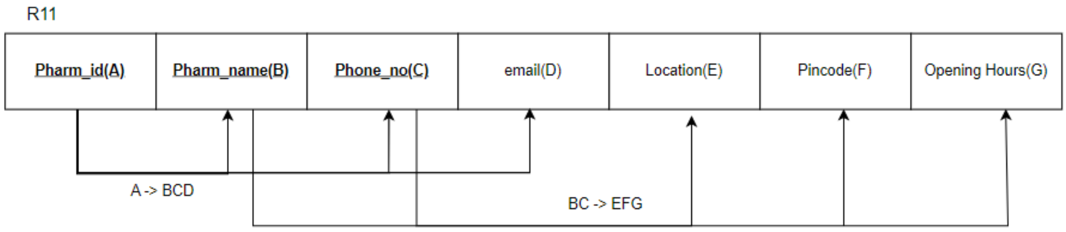
Candidate key : AH

Checking for the 2NF:

For R11:

FD's :

- A->B
- A->C
- A->D
- BC->EFG



- In the 2NF, relation must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key. In simple words there shouldn't present any partial functional dependencies.

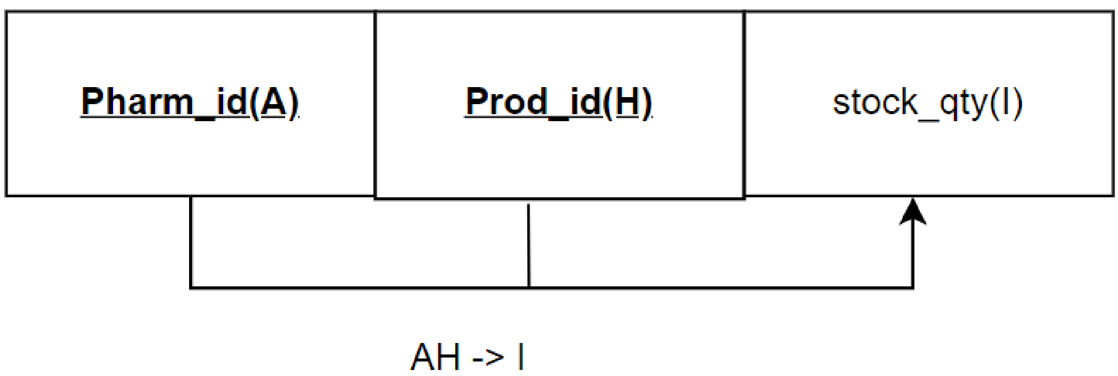
Prime Attributes: {A}
Non-Prime Attributes: {B, C, D,E,F,G}

There is no proper subset for the prime attributes.
Hence the relation is already in 2NF.

For R12:

FD's :

- **AH->I**



Candidate Key : AH

Prime Attributes : {A,H}
Non-prime Attributes : {I}
Closure of A;

A^+ = {A} -------------not a super key.

Closure of H:

H^+ = {H} ------------not a super key.

- There is no partial dependency in the above relation.
- So, It is in 2NF.

Checking for the 3NF:

- A relation will be in 3NF if it is in 2 NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in 3NF

For R11(A, B, C, D,E,F,G).

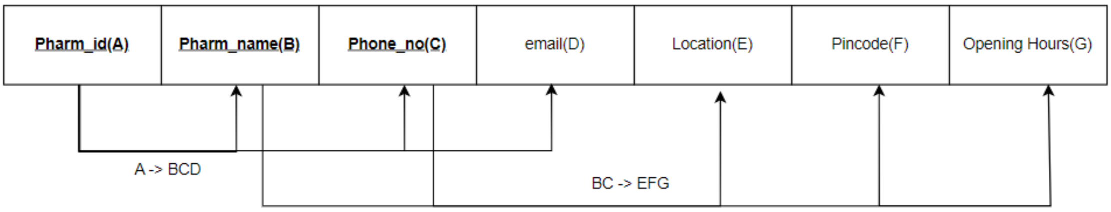
Violating Functional Dependencies:

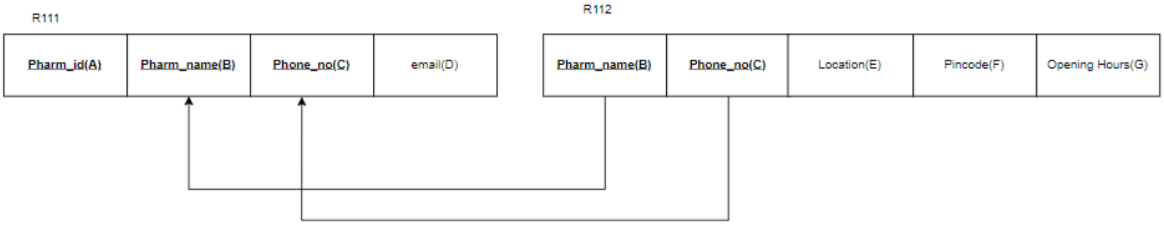
- BC->EFG

Decomposing:

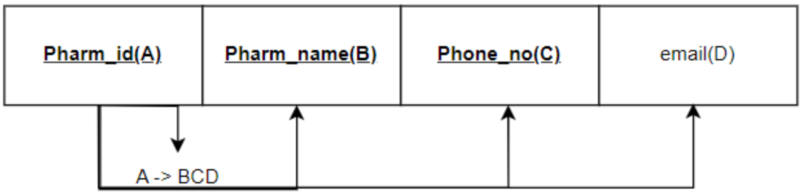
BC^+ = {B,C,E,F,G}

R11

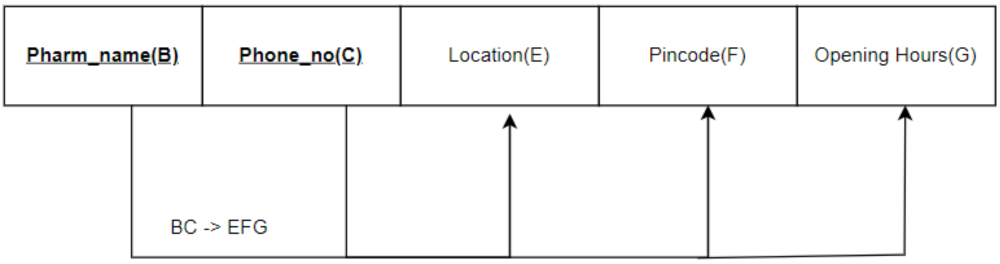




R111



R112



Checking for 3NF in R111(A,B,C,D) :

Functional Dependencies :

- A->B
- A->C
- A->D

Finding candidate key :

$ABCD^+$ ={A,B,C,D}

Decomposing with the A -> B
ACD^+ ={A,B,C,D }

Decomposing with the A -> C
AD^+ ={A,B,C,D}

Decomposing with the A -> D
A^+ ={A,B,C,D}

Therefore A^+ includes all the Attributes hence it is a super key.
Hence A^+ is the Candidate Key in the table. Which uniquely identifies all other attributes in the relation.

Prime Attribute : {A}
Non Prime Attribute : {B,C,D}

- There is no Violating FD's.
- So, it is in 3NF.

Checking for 3NF in R112(B,C,E,F,G) :

Functional Dependencies :

- BC->E
- BC->F
- BC->G

Finding candidate key :

$BCEFG^+$ = {B,C,E,F,G}

Decomposing with the BC->EFG
BC^+ ={B,C,E,F,G}

Here the candidate key is "BC"

Decomposing candidate key :
B^+ = {B}
C^+ = {C}

So the candidate key : BC

Prime attribute : {B,C}

Non prime Attribute : {E,F,G}

- There is no Violating FD's.
- So, it is in 3NF.

Checking for 3NF R12(A,H,I)

Functional Dependencies:

AH->I

- A relation will be in 3NF if it is in 2 NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in 3NF

The relation satisfies the 3NF.

Checking for the BCNF:

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X->Y, X is the super key of the table.
- For BCN, the table should be in 3NF, and for every FD, LHS is super key.

Checking BCNF for R111(A,B,C,D) :

Functional Dependencies :

- A->B
- A->C
- A->D

We can write it as **A->BCD**

Candidate Key : A^+

- Here the LHS of the candidate key is a super key.
- So,it is in BCNF.

Checking BCNF for R112(B,C,E,F,G) :

Functional Dependencies :

- BC->EFG

Candidate Key : BC^+

- Here the LHS of the candidate key is a super key.
- so, it is in BCNF.

Checking BCNF for R12(A,E,I)

Functional Dependencies:

AH->I

Candidate Key: AE

- Here the LHS of the candidate key is a super key.
- So,it is in BCNF.

Orders:

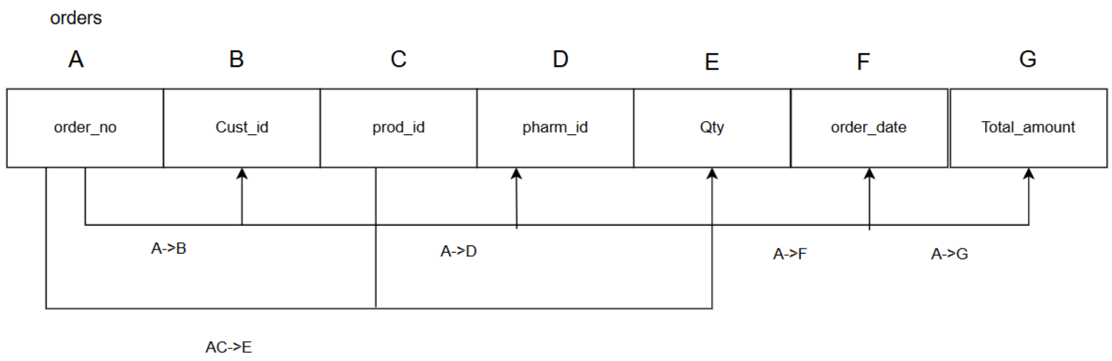
- Order_no - **A**
- Cust_id - **B**
- Prod_id - **C (multi valued attribute)**
- Pharm_id - **D**
- Qty - **E (multi valued attribute)**
- Order_date - **F**
- Total_amount - **G**

Identified Functional Dependencies:

- A-> B
- A->D
- A->F
- A->G
- AC->E

From the previous pdf we can see the minimal set of functional dependencies are :

- A->B
- A->D
- A->F
- A->G
- AC->E



From evaluating with the minimal set of FD's we arrive that the AC^+ is the only one candidate key, and It is the primary key.

Checking for the 1NF:

Conditions need to be satisfied are:

- Each attribute has a value related to the domain.
- Each attribute has a unique name.
- No rows get duplicated.

It violates each attribute has a single value,

So, it is not in 1NF.

Converting it into 1NF :

Splitting the table into :

R2

Order_no(A)	Cust_id(B)	Prod_id(C)	Pharm_id(D)	Qty(E)	order_date(F)	Total_amount(G)

Orders

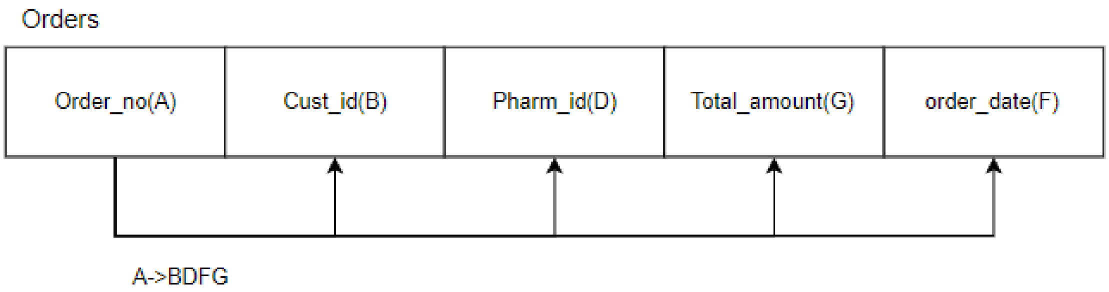
Order_no(A)	Cust_id(B)	Pharm_id(D)	Total_amount(G)	order_date(F)

Order_list

Order_no(A)	Prod_id(C)	Qty(E)

1 NF for Orders :

- Each attribute has a value related to the domain.
- Each attribute has a unique name.
- No rows get duplicated.
- No rows get duplicated.



Finding candidate key for Orders(A,B,D,G) :

FD's :

- A->B
- A->D

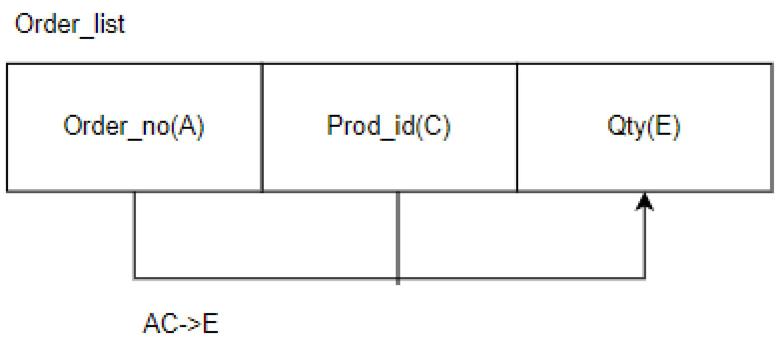
- A->G
- A->F

ABDGF$^+$ = {A,B,D,G}
ADGF+ = {A,B,D,G}
AGF$^+$ = {A,B,D,G}
AF$^+$ = {A,B,D,G}
A$^+$ = {A,B,D,G}

Candidate Key : A$^+$

1 NF for Order_list :

- Each attribute has a value related to the domain.
- Each attribute has a unique name.
- No rows get duplicated.
- No rows get duplicated.



Finding candidate key for Order_list(A,C,E) :

FD's :

- **AC->E**

ACE$^+$ = {A, C, E}
AC$^+$= {A, C, E}

Candidate key : AC

Checking for the 2NF:

<u>For Orders table:</u>

FD's :

- **A->B**
- **A->D**
- **A->G**

- In the 2NF, relation must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key. In simple words there shouldn't present any partial functional dependencies.

  Prime Attributes: {A}
  Non-Prime Attributes: {B, D,G}

  There is no proper subset for the prime attributes.
  Hence the relation is already in 2NF.

<u>For Order_List :</u>

FD's :

- **AC->E**

Candidate Key : AC

Prime Attributes : {A,C}
Non prime Attributes : {E}

- There is no proper subset.
- So,It is in 2NF.

Checking for the Orders:

A relation will be in 3NF if it is in 2 NF and not contain any transitive partial dependency.

- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in 3NF.

  For Orders Table :

  Violating Functional Dependencies : NULL
  So, the orders table is in 3NF.

  For order_list :

  Violating Functional Dependencies : NULL

So, the order_list table is in 3NF.

Checking for the BCNF:

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X->Y, X is the super key of the table.
- For BCN, the table should be in 3NF, and for every FD, LHS is super key.

Checking BCNF for Orders(A,B,D,G) :

Functional Dependencies :

- **A->B**
- **A->D**
- **A->G**

We can write it as **A->BDG**

Candidate Key : A^+

- Here the LHS of the candidate key is a super key.
- So,it is in BCNF.

Checking BCNF for Order_list(A,C,E) :

Functional Dependencies :

- AC->E

Candidate Key : AC^+

- Here the LHS of the candidate key is a super key.
- So,it is in BCNF.

<u>Supplier(A,B,C,D,E,F,G,H) :</u>

- Sup_id **- A**
- Prod_id **- B(multi valued attribute)**
- Sup_name **- C**
- Sup_location **- D**
- Sup_phone **- E**
- Quantity **- F(multi valued attribute)**
- Date_mfg **- G(multi valued attribute)**
- Date_exp **- H(multi valued attribute)**

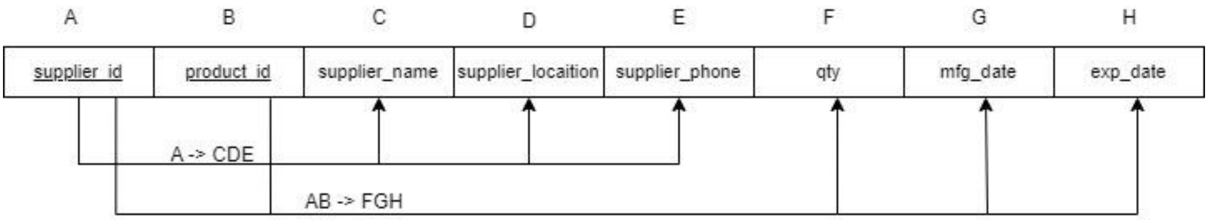
Identified Functional Dependencies:

- A->C
- A->D
- A->E
- AB->F
- AB->G
- AB->H

From the previous PDF the minimal set of functional dependencies are:

- A->C – FD 1
- A->D – FD 2
- A->E – FD 3
- AB->F – FD 4
- AB->G – FD 5
- AB->H – FD 6

Hence, the candidate key for the above relation is (sup_id, prod_id) AB^+



Checking for the 1NF:

Conditions need to be satisfied are:

- Each attribute has a value related to the domain.
- Each attribute has a unique name.
- No rows get duplicated.

It violates each attribute has a single value,

So, it is not in 1NF.

Converting it into 1NF :

Splitting the table into :

R11

sup_id(A)	sup_name(C)	sup_location(D)	sup_phone(E)

R12

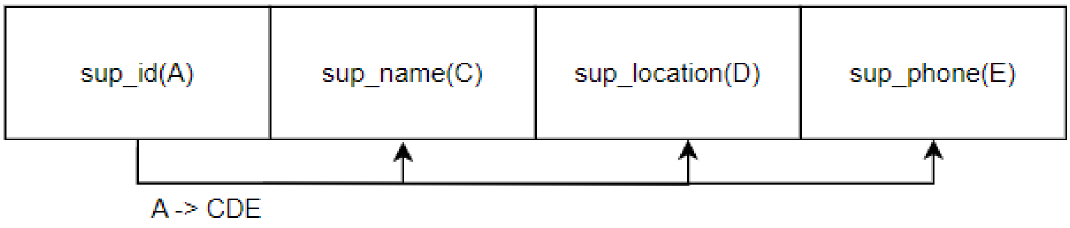
sup_id(A)	prod_id(B)	quantity(F)	date_mfg(G)	date_exp(H)

1NF for R11:

Set of functional dependencies:

A->C
A->D
A->E

R11



Finding the candidate Key:

$ACDE^+ = \{A, C, D, E\}$

Decomposing with the A -> C

$ADE^+ = \{A, C, D, E\}$

Decomposing with the A -> D

$AE^+ = \{A, C, D, E\}$

Decomposing with the A -> E

$A^+ = \{A, C, D, E\}$

Therefore A^+ includes all the Attributes hence it is a super key. Hence A^+ is the Candidate Key in the table. Which uniquely identifies all other attributes in the relation.

Prime Attribute: {A}

Non-Prime Attribute: {B, C, F, G}

Hence, we see that there are no partial dependencies in the FD's of the R11

In R12(A, B, F, G, H)

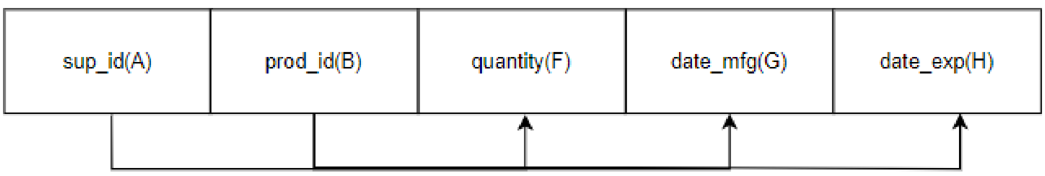
Set of functional dependencies:

AB->F

AB->G

AB->H

R12



Finding the candidate Key:

$ABFGH^+ = \{A, B, F, G, H\}$

Decomposing with the AB -> F

$ABGH^+ = \{A, B, F, G, H\}$

Decomposing with the AB -> G

$ABH^+ = \{A, B, F, G, H\}$

Decomposing with the AB -> H

$AB^+ = \{A, B, F, G, H\}$

Therefore AB^+ includes all the Attributes hence it is a super key. Let's check if the subset of the AD includes a super key or not.

Closure of A^+

$A^+ = \{A\}$ ----------not a super key

Closure of B^+

$B^+ = \{B\}$ ---------not a super key

Hence (sup_id, prod_id)AB^+ is the Candidate Key in the table. Which uniquely identifies all other attributes in the relation.

Prime Attribute: {A, B}

Non-Prime Attribute: {F, G, H}

Hence, we see that there are no partial dependencies in the FD's of the R12

Checking for the 2NF:

For R11:

FD's :

- **A->C**
- **A->D**
- **A->E**

- In the 2NF, relation must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key. In simple words there shouldn't present any partial functional dependencies.

Prime Attributes: {A}

Non-Prime Attributes: {C, D, E}

There is no proper subset for the prime attributes.

Hence the relation is already in 2NF.

For R12:

FD's are

AB->F

AB->G

AB->H

Candidate Key : AB

Prime Attributes : {A,B}
Non prime Attributes : {F, G, H}

- There is no proper subset.
- So,It is in 2NF.

Checking for 3NF:

- A relation will be in 3NF if it is in 2 NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in 3NF.

  Since there exists no transitive partial dependency in the above functional dependencies.

  The relation is already in 3NF

Checking for the BCNF:

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X->Y, X is the super key of the table.
- For BCN, the table should be in 3NF, and for every FD, LHS is super key.

It also satisfies the BCNF.

Checking BCNF for R11(A, C, D, E):

Functional Dependencies:

A->C
A->D
A->E

We can write it as A->CDE

Candidate Key : A^+

- Here the LHS of the candidate key is a super key.
- So, it is in BCNF.

Checking BCNF for R12(A, B, F, G, H):

Functional Dependencies:

- AB->F
- AB->G
- AB->H

Candidate Key : AB^+

- Here the LHS of the candidate key is a super key.
- So, it is in BCNF.

Schema Diagram

<u>CONVERTING ER DIAGRAM TO SCHEMA DIAGRAM:</u>

Representing entities:

- Since all the entities present in our model are strong entity types, a relation, or table, is directly created for each.

Representing relationships:

a) 1:1 relationship/cardinality (one-to-one):

- When 2 entities are related by a one-to-one cardinality, the primary key of one of the entities is linked as a foreign key to the other, or they share the same primary key.

b) 1:N relationship/cardinality (one-to-many):

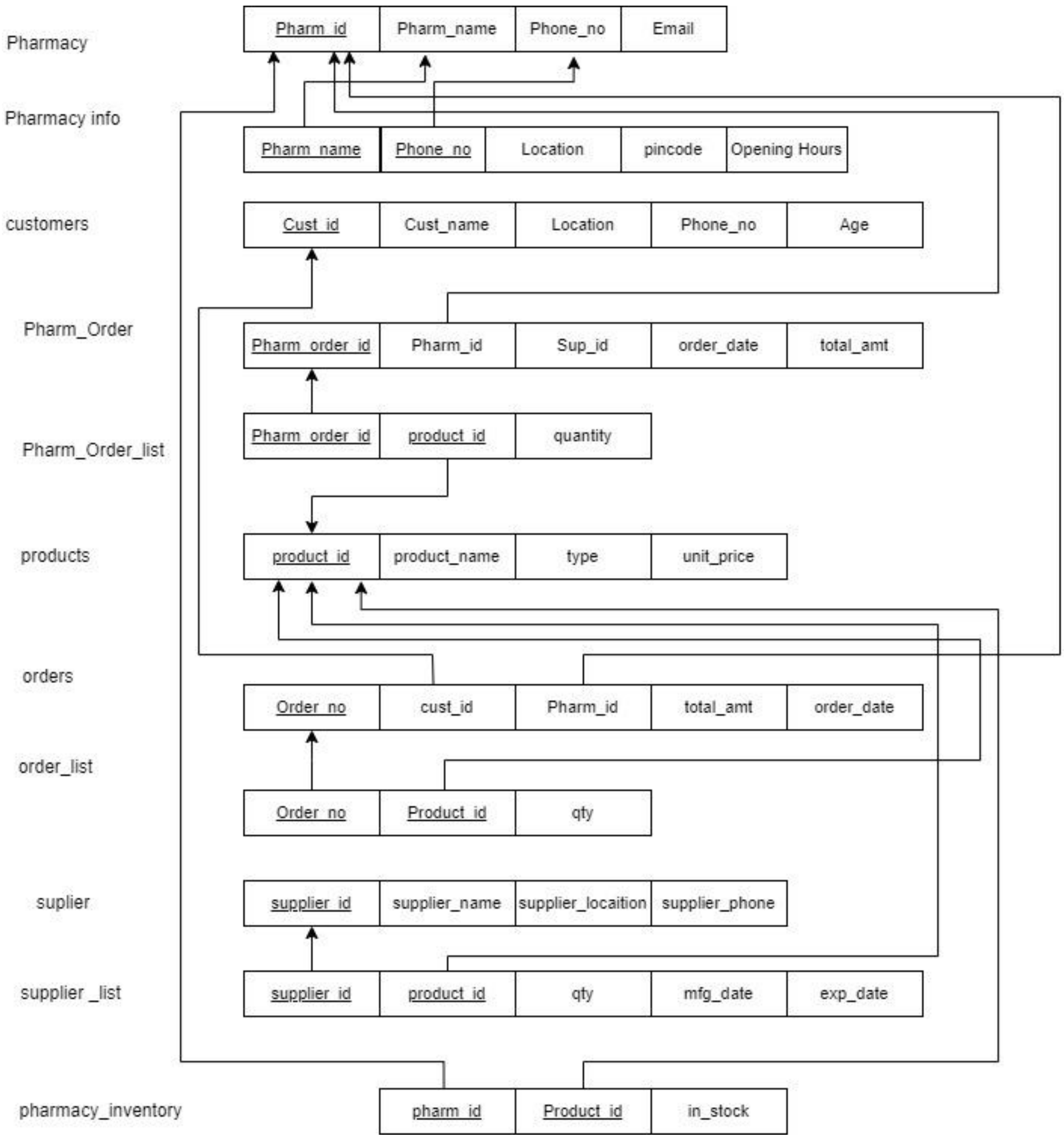
- When 2 entities are related by a one-to-many cardinality, the primary key of the “one” entity is linked as a foreign key to the “many” entity.
- This is illustrated in the Customer-places-Order or Supplier-receives-Order relationship in our ER

c) N:1 relationship/cardinality (many-to-one):

- When 2 entities are related by a one-to-many cardinality, the primary key of the “one” entity is linked as a foreign key to the “many” entity
- This is illustrated in the Order-completes-Transaction relationship in our ER.

d) M:N relationship/cardinality (many-to-many):

- When 2 entities are related by a many-to-many cardinality, the primary key of one entity is linked as a foreign key to the other entity.
- Alternatively, a junction table could be used.
- This is illustrated in the Supplier-supplies to-Pharmacy, Supplier-supplies to-Pharmacy or Pharmacy-contains-Product relationship in our ER


Pharmacy
Pharm_id
Pharm_name
Phone_no
Email
Pharmacy info
Pharm_name
Phone_no
Location
pincode
Opening Hours
customers
Cust_id
Cust_name
Location
Phone_no
Age
Pharm_Order
Pharm_order_id
Pharm_id
Sup_id
order_date
total_amt
Pharm_Order_list
Pharm_order_id
product_id
quantity
products
product_id
product_name
type
unit_price
orders
Order_no
cust_id
Pharm_id
total_amt
order_date
order_list
Order_no
Product_id
qty
suplier
supplier_id
supplier_name
supplier_locaition
supplier_phone
supplier _list
supplier_id
product_id
qty
mfg_date
exp_date
pharmacy_inventory
pharm_id
Product_id
in_stock

ASSUMPTIONS AND WORK FLOW:

CREATION OF TABLES:

```sql
      -- Table: Pharmacy

CREATE TABLE Pharmacy (
    Pharm_id VARCHAR2(4) CONSTRAINT Pharmacy_pk PRIMARY KEY,
    Pharm_name VARCHAR2(50),
    Phone_no VARCHAR2(10),
    Email VARCHAR2(100)
);

-- Table: Pharmacy_info
CREATE TABLE Pharmacy_info (
    Pharm_name VARCHAR2(50) CONSTRAINT Pharmacy_info_Pharm_name_fk REFERENCES
Pharmacy(Pharm_name),
    Phone_no VARCHAR2(10)  CONSTRAINT Pharmacy_info_Phone_no_fk REFERENCES
Pharmacy(Phone_no),
    location VARCHAR2(100),
    pincode NUMBER(6),
    Opening_Hours VARCHAR2(50),
    CONSTRAINT Pharmacy_info_pk PRIMARY KEY (Pharm_name, Phone_no)
);

-- Table: Customers
CREATE TABLE Customers (
    Cust_id VARCHAR2(4) CONSTRAINT Customers_pk PRIMARY KEY,
    Cust_name VARCHAR2(50),
    Location VARCHAR2(50),
    Phone_no VARCHAR2(10),
    Age NUMBER(3)
);

-- Table: Products
CREATE TABLE Products (
    product_id VARCHAR2(5) CONSTRAINT Products_pk PRIMARY KEY,
    product_name VARCHAR2(100),
    type VARCHAR2(50),
    unit_price DECIMAL(10, 2)
);

-- Table: Supplier
CREATE TABLE Supplier (
    supplier_id VARCHAR2(4) CONSTRAINT Supplier_pk PRIMARY KEY,
    supplier_name VARCHAR2(100),
    supplier_location VARCHAR2(100),
```

```sql
    supplier_phone VARCHAR2(10)
);

-- Table: Supplier_List
CREATE TABLE Supplier_List (
    supplier_id VARCHAR2(4) CONSTRAINT
Supplier_List_supplier_id_fk  REFERENCES Supplier(supplier_id),
    product_id VARCHAR2(5) CONSTRAINT Supplier_List_product_id_fk  REFERENCES
Products(product_id),
    qty NUMBER,
    mfg_date DATE,
    exp_date DATE,
    CONSTRAINT Supplier_List_pk PRIMARY KEY (supplier_id, product_id),
    CONSTRAINT mfg_exp_dates_check CHECK (mfg_date < exp_date)
);

-- Table: Pharm_Order
CREATE TABLE Pharm_Order (
    Pharm_order_id VARCHAR2(5) CONSTRAINT Pharm_Order_pk PRIMARY KEY,
    Pharm_id VARCHAR2(4) CONSTRAINT Pharm_Order_Pharm_id_fk REFERENCES
Pharmacy(Pharm_id),
    Sup_id VARCHAR2(4) CONSTRAINT Pharm_Order_Sup_id_fk REFERENCES
Supplier(supplier_id) ,
    order_date DATE,
    total_amt DECIMAL(10, 2),
);

-- Table: Pharm_Order_list
CREATE TABLE Pharm_Order_list (
    Pharm_order_id VARCHAR2(5) CONSTRAINT
Pharm_Order_list_Pharm_order_id_fk  REFERENCES Pharm_Order(Pharm_order_id) ,
    product_id VARCHAR2(5)CONSTRAINT Pharm_Order_list_product_id_fk REFERENCES
Products(product_id),
    quantity NUMBER,
    CONSTRAINT Pharm_Order_list_pk PRIMARY KEY (Pharm_order_id, product_id),
);

-- Table: Orders
CREATE TABLE Orders (
    Order_no VARCHAR2(4) CONSTRAINT Orders_pk PRIMARY KEY,
    cust_id VARCHAR2(4) CONSTRAINT Orders_cust_id_fk REFERENCES
Customers(Cust_id),
    Pharm_id VARCHAR2(4) CONSTRAINT Orders_Pharm_id_fk REFERENCES
Pharmacy(Pharm_id),
    total_amt DECIMAL(10, 2),
    order_date DATE,
);
```

```sql
-- Table: Order_List
CREATE TABLE Order_List (
    Order_no VARCHAR2(4) CONSTRAINT Order_List_Order_no_fk  REFERENCES
Orders(Order_no),
    product_id VARCHAR2(5) CONSTRAINT Order_List_Product_id_fk  REFERENCES
Products(product_id),
    qty NUMBER,
    CONSTRAINT Order_List_pk PRIMARY KEY (Order_no, Product_id),
);

-- Table: Pharmacy_Inventory
CREATE TABLE Pharmacy_Inventory (
    Pharm_id VARCHAR2(4) CONSTRAINT Pharmacy_Inventory_Pharm_id_fk REFERENCES
Pharmacy(Pharm_id),
    product_id VARCHAR2(5) CONSTRAINT Pharmacy_Inventory_Product_id_fk
REFERENCES Products(product_id),
    in_stock numeric(4),
    CONSTRAINT Pharmacy_Inventory_pk PRIMARY KEY (Pharm_id, Product_id),
);
```

TRIGGERS/PROCEDURES USED

1. Pharmacy Add Total Amount:

This trigger (pharm_add_total_amt) updates the total_amt field in the pharm_order table when a new row is inserted into the pharm_order_list table. It performs an update operation, calculates the total amount, and ensures accuracy by adding the product quantity and unit price of the new row. This trigger maintains data integrity by eliminating the need for manual updates and ensuring database consistency.

```sql
-- pharm order order list trigger total amount
create or replace trigger pharm_add_total_amt
after insert on pharm_order_list
for each row
begin
    update pharm_order
        set total_amt = total_amt + (:NEW.qty * (select unit_price from
products where product_id = :NEW.product_id))
    where pharm_order_id=:NEW.pharm_order_id;
end;
/
```

2. Customer Add Total Amount:

This trigger updates the total_amt field in the orders table when a new row is inserted into the order_list table. It performs an update operation, calculates the total amount, and ensures accuracy by adding the product of the newly inserted row and the unit price of the corresponding product. This trigger maintains data integrity and avoids manual updates, ensuring consistency in the database.

```
-- customer order order list total amount
create or replace trigger customer_add_total_amt
after insert on order_list
for each row
begin
    update orders
        set total_amt = total_amt + (:NEW.qty * (select unit_price from
products where product_id = :NEW.product_id))
    where order_no=:NEW.order_no;
end;
/
```

3. Supplier Quantity Check:

This trigger ensures the availability of stock before inserting a new row into the Pharm_order_list table. It retrieves the supplier ID and available quantity from the supplier_list, checks if the quantity is less than or equal to the requested quantity, and raises an application error if it's insufficient. The trigger maintains stock integrity by preventing orders exceeding available stock, and automatically adjusts the stock based on the ordered quantity. This is crucial for accurate inventory management and pharmacy order fulfillment.

```
CREATE OR REPLACE TRIGGER pharm_quan_check
BEFORE INSERT ON Pharm_order_list
FOR EACH ROW
DECLARE
    sel_sup_id Pharm_order.sup_id%TYPE;
    sel_quan supplier_list.qty%TYPE;
BEGIN
    -- Retrieve the Supplier ID for the given Pharmacy order ID
    SELECT sup_id INTO sel_sup_id
    FROM Pharm_Order
    WHERE pharm_order_id = :NEW.Pharm_order_id;

    -- Retrieve the available quantity in stock for the product from
supplier_list
    SELECT qty INTO sel_quan
    FROM supplier_list
    WHERE supplier_id = sel_sup_id
```

```
      AND product_id = :NEW.product_id;

    -- Check if the available quantity is less than the requested quantity
    IF sel_quan <= :NEW.quantity THEN
        RAISE_APPLICATION_ERROR(-20001, 'The selected quantity is not
available');
    ELSE
        -- Update the supplier_list to subtract the ordered quantity
        UPDATE supplier_list
        SET qty = qty - :NEW.quantity
        WHERE supplier_id = sel_sup_id
          AND product_id = :NEW.product_id;
    END IF;
END;
/
```

4. Pharmacy Quantity Check:

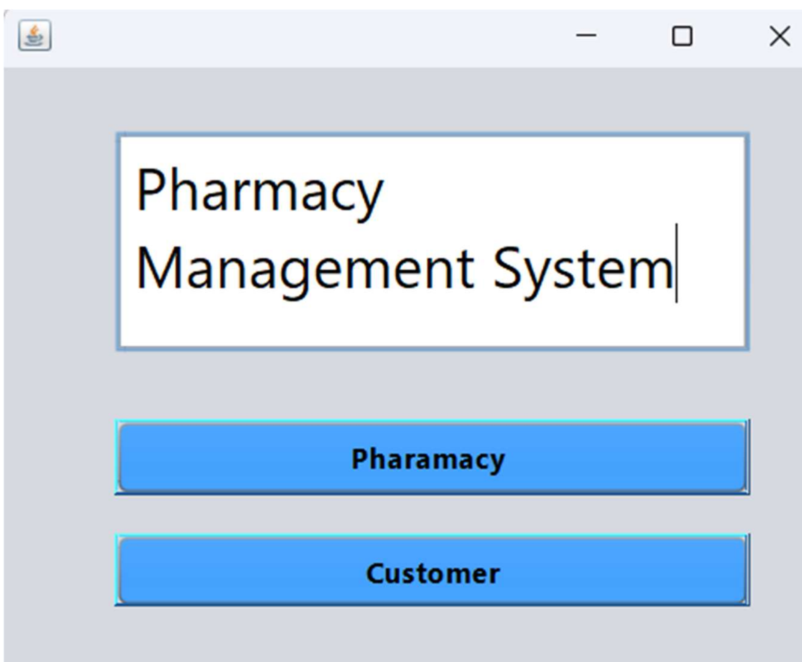
This trigger ensures the availability of stock in Pharmacy_Inventory for the specified pharmacy and product before inserting a new row into the order_list table. It checks if the available quantity is less than the requested quantity, raises an application error if true, or updates Pharmacy_Inventory to subtract the ordered quantity from the stock. This trigger maintains stock integrity, prevents overselling or stockouts, and automatically adjusts the available stock based on the ordered quantity.

```
-- check if you can buy the quantity from customer side

CREATE OR REPLACE TRIGGER cus_quan_check
BEFORE INSERT ON order_list
FOR EACH ROW
DECLARE
    sel_Pharm_id Pharmacy_Inventory.Product_id%TYPE;
    sel_quan Pharmacy_Inventory.in_stock%TYPE;
BEGIN
    -- Retrieve the Pharmacy ID for the given order number
    SELECT Pharm_id INTO sel_Pharm_id
    FROM Orders
    WHERE order_no = :NEW.order_no;

    -- Retrieve the available quantity in stock for the product from
Pharmacy_Inventory
    SELECT in_stock INTO sel_quan
    FROM Pharmacy_Inventory
    WHERE pharm_id = sel_Pharm_id
      AND product_id = :NEW.product_id;
```

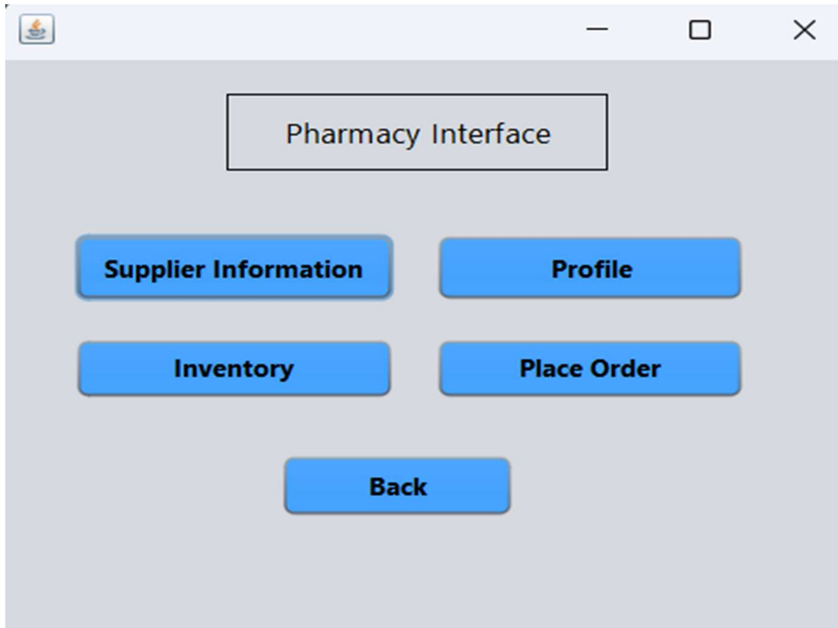
```
    -- Check if the available quantity is less than the requested quantity
    IF sel_quan <= :NEW.qty THEN
        RAISE_APPLICATION_ERROR(-20001, 'The selected quantity is not
available');
    ELSE
        -- Update the Pharmacy_Inventory to subtract the ordered quantity
        UPDATE Pharmacy_Inventory
        SET in_stock = in_stock - :NEW.qty
        WHERE pharm_id = sel_Pharm_id
          AND product_id = :NEW.product_id;
    END IF;
END;
/
```

SCREENSHOTS OF WORKFLOW:

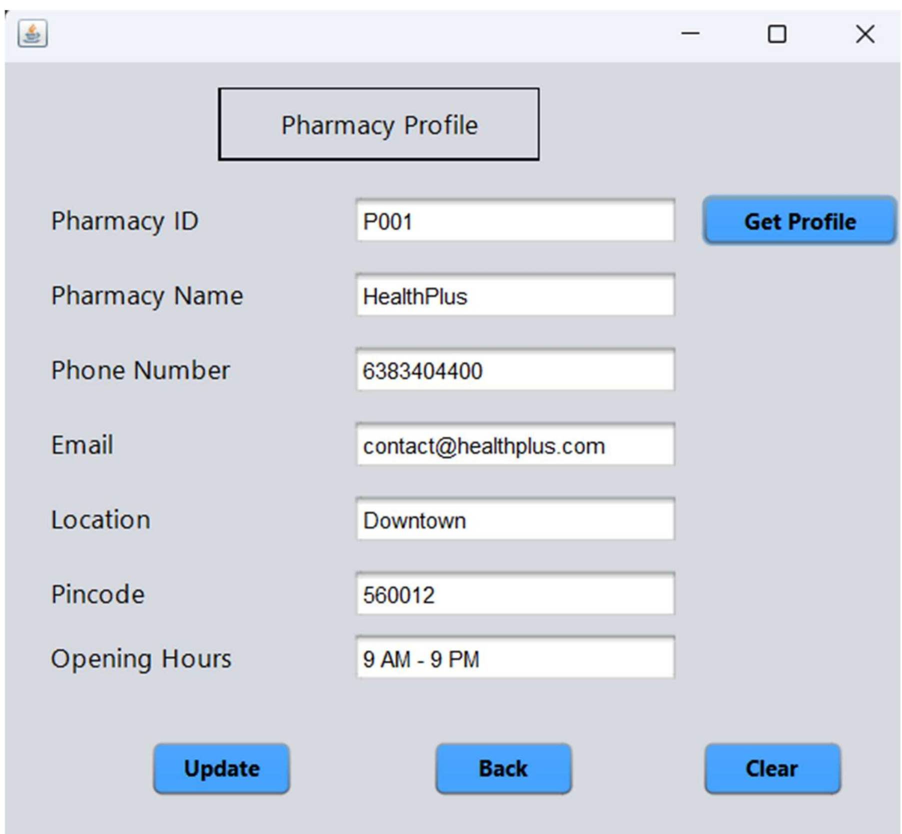
MAIN INTERFACE:



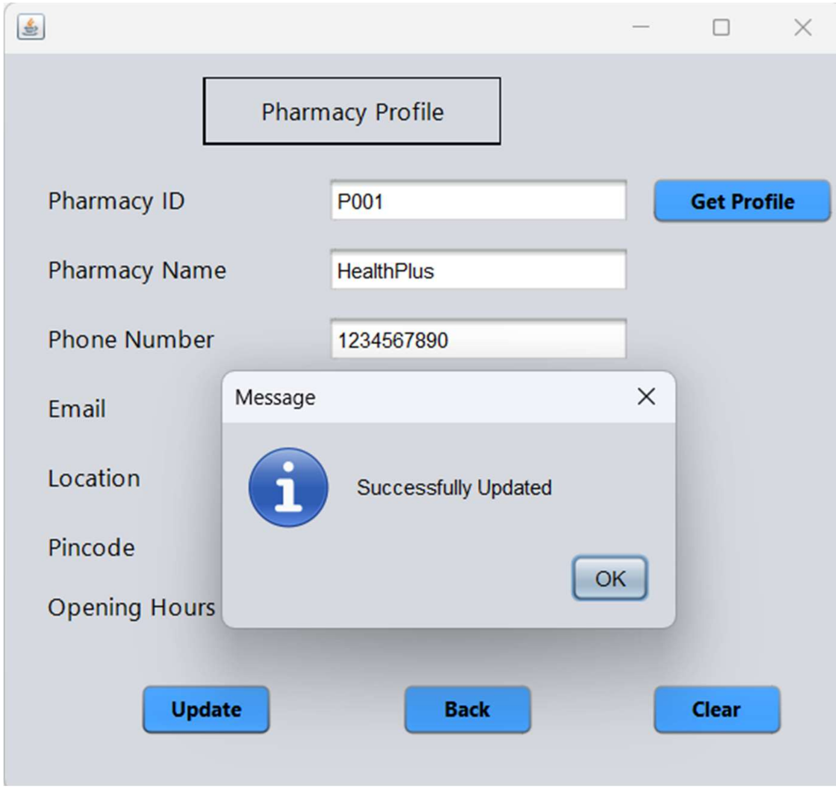
PHARMACY INTERFACE:



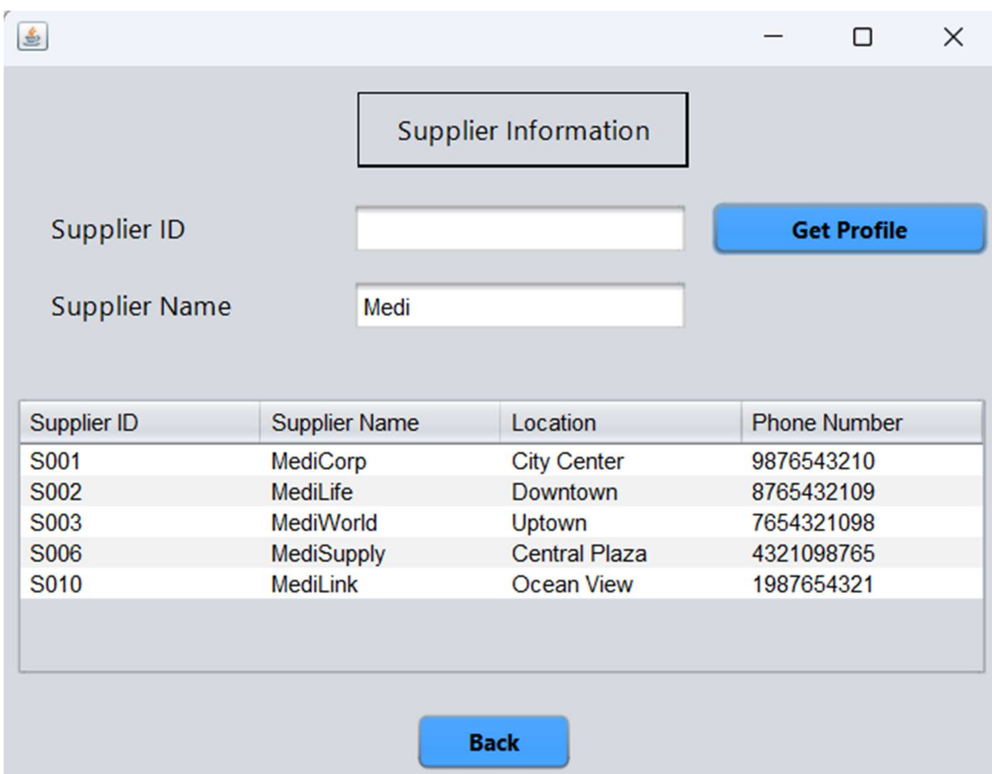
PROFILE



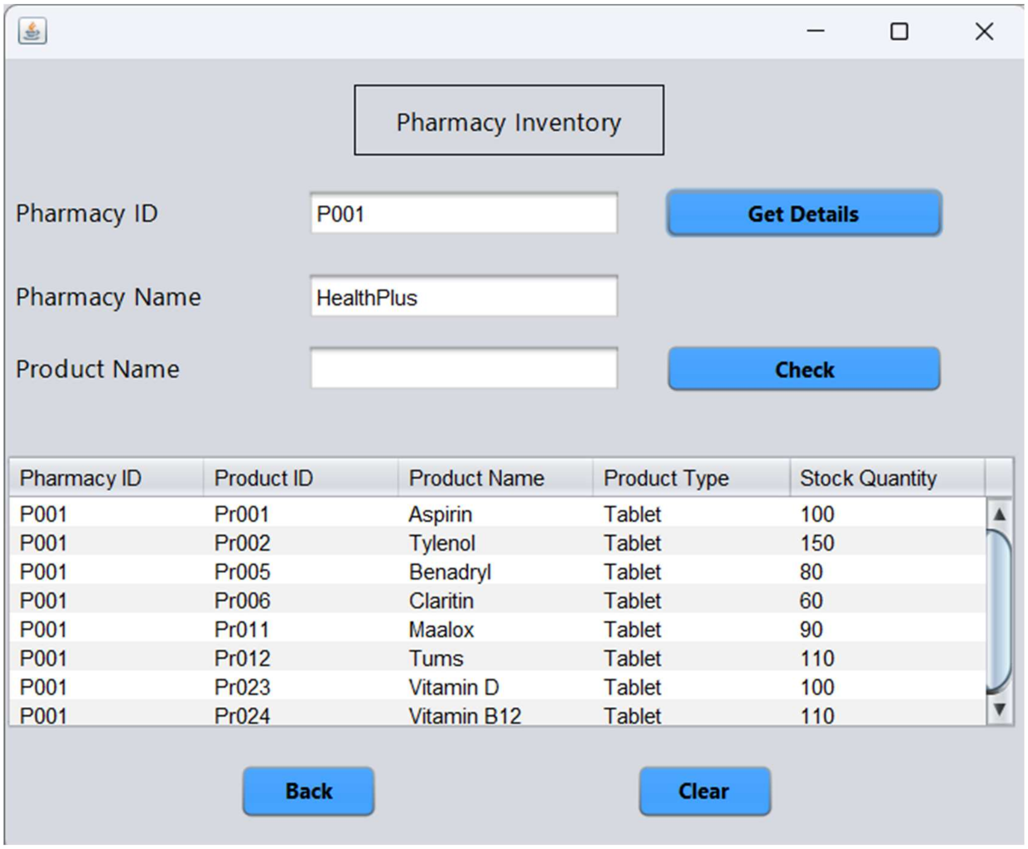
UPDATE FUNCTIONALITY (Phone number updated):



SUPPLIER INFORMATION (To know about the supplier):

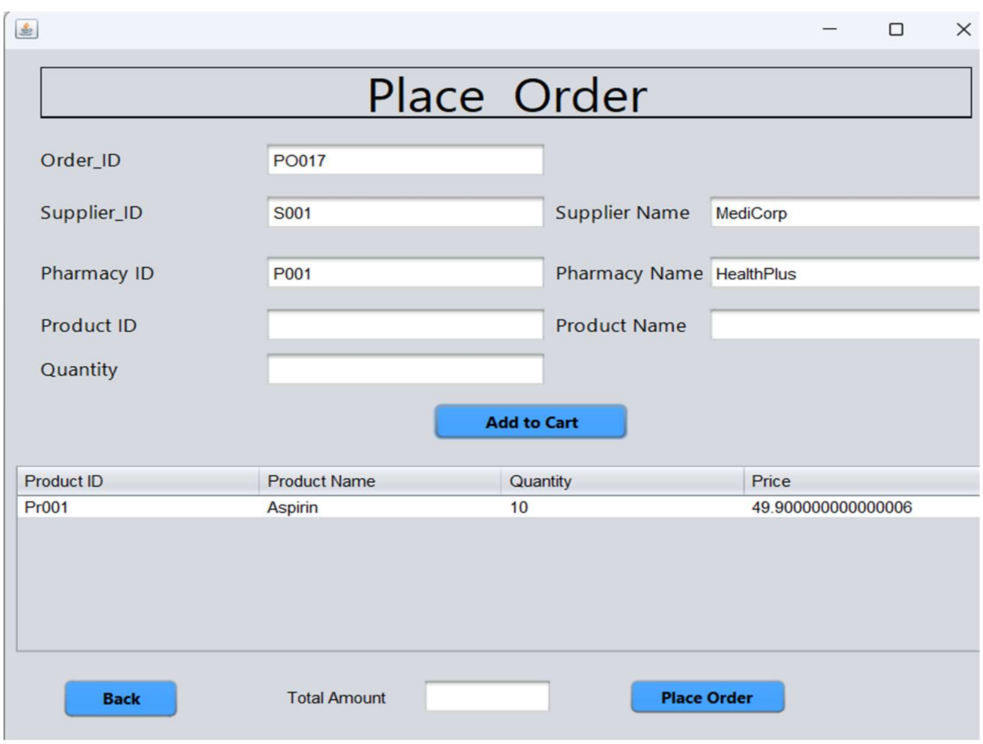


PHARMACY INVENTORY-(To know about the stock quantity);

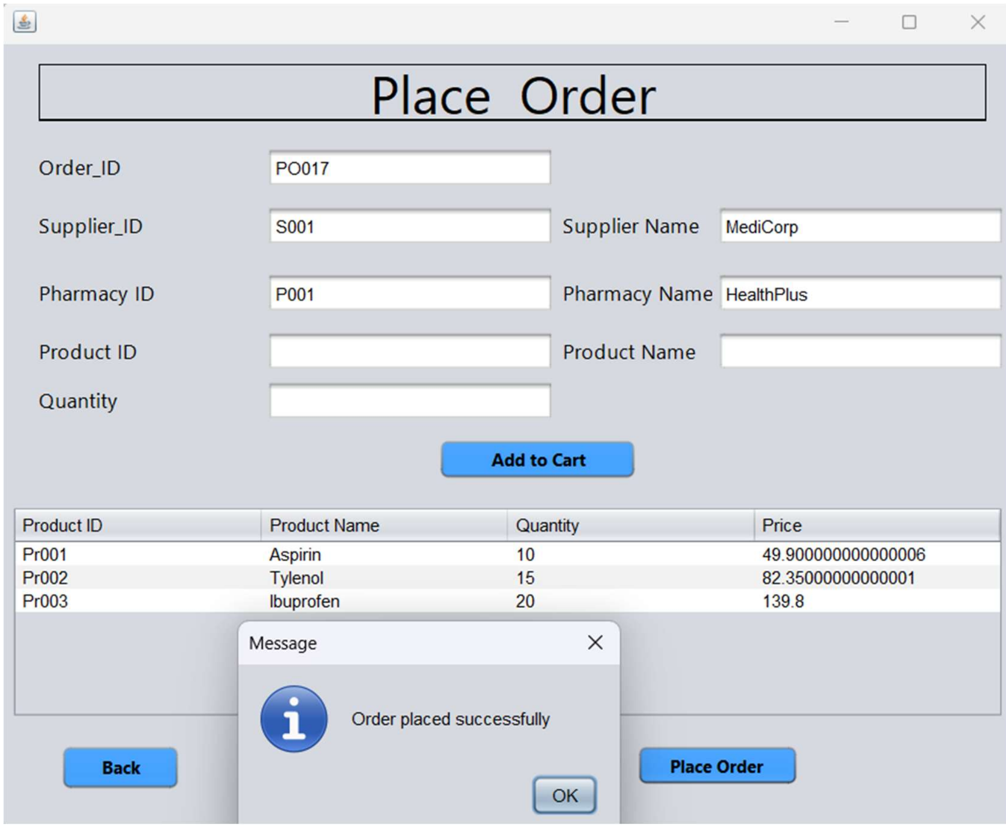


PHARMACY INVENTORY (Check functionality):

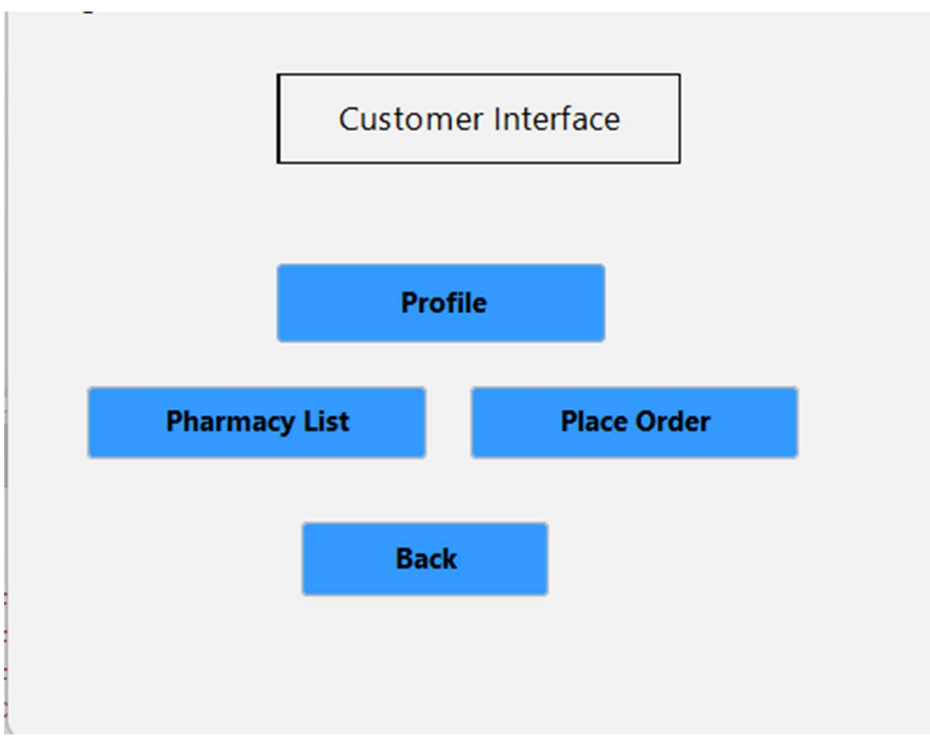
PHARMACY PLACE ORDER TO SUPPLIERS (Add to Cart)



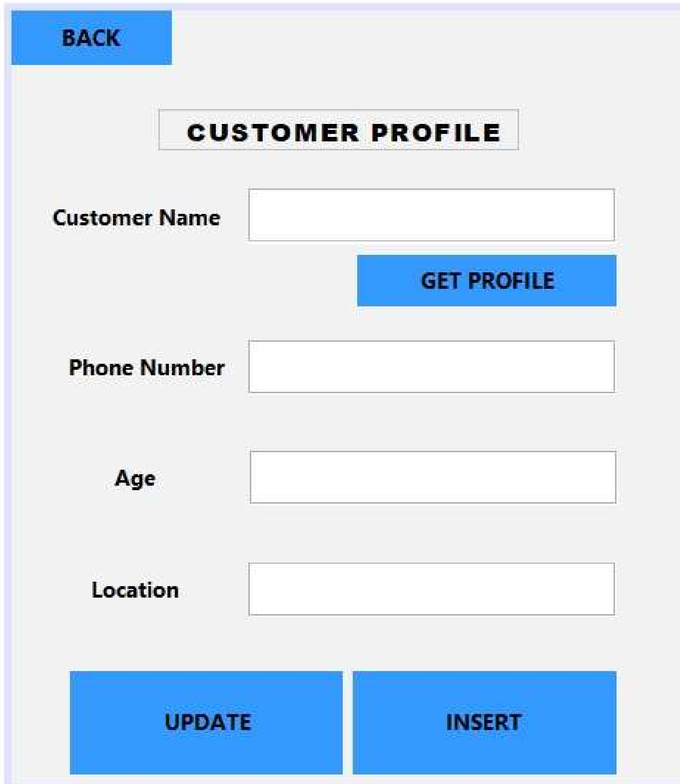
PLACING THE ORDER:



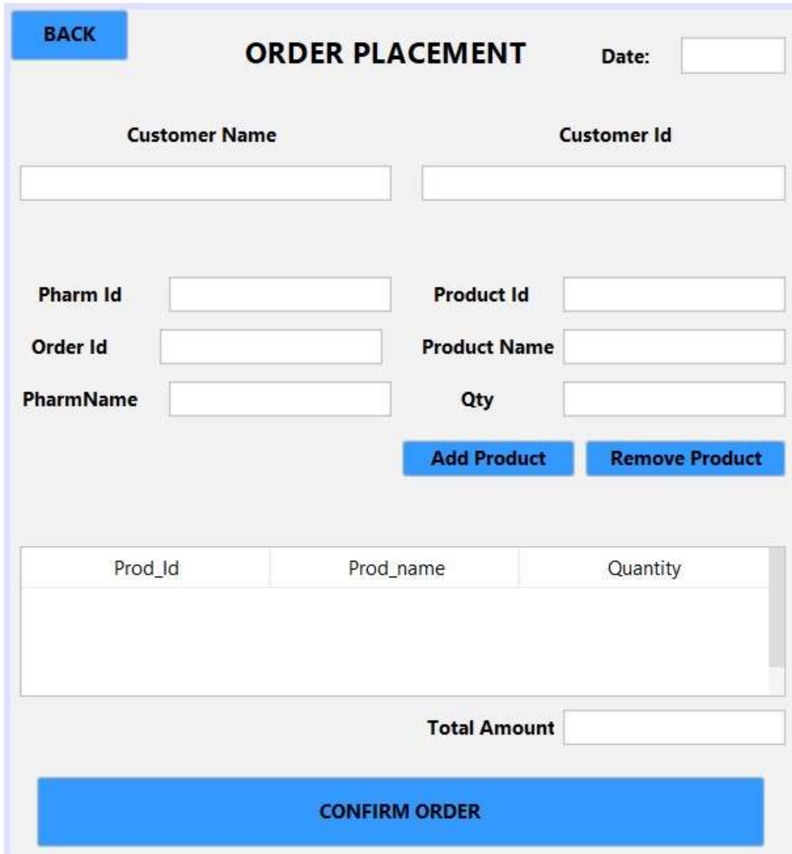
CUSTOMER INTERFACE:



CUSTOMER PROFILE:



ORDER PLACEMENT:


BACK
ORDER PLACEMENT
Date:
Customer Name
Customer Id
Pharm Id
Product Id
Order Id
Product Name
PharmName
Qty
Add Product
Remove Product
Prod_Id
Prod_name
Quantity
Total Amount
CONFIRM ORDER

NOVELTY:

- The pharmacy management system project aims to streamline operations and improve user experience through innovative database management and a user-friendly interface.
- A robust database schema is designed to manage inventory, order processing, and supplier relations. An intuitive user interface using NetBeans IDE simplifies navigation and enhances usability for pharmacists and administrators.
- Real-time inventory tracking is implemented, preventing stockouts and ensuring timely medication replenishment. Order processing is seamless, with functionalities for product quantities and unit prices.
- Reporting capabilities generate insights into sales trends, profitability, and supplier performance, enabling decision-making for optimizing inventory levels and supplier relationships.
- This project showcases the application of database principles and NetBeans development to meet pharmacy operations' specific needs.