DBMS Project Report

PES University

Database Management Systems

UE18CS252

A Project On

RETAIL SHOP MANAGEMENT

Submitted By

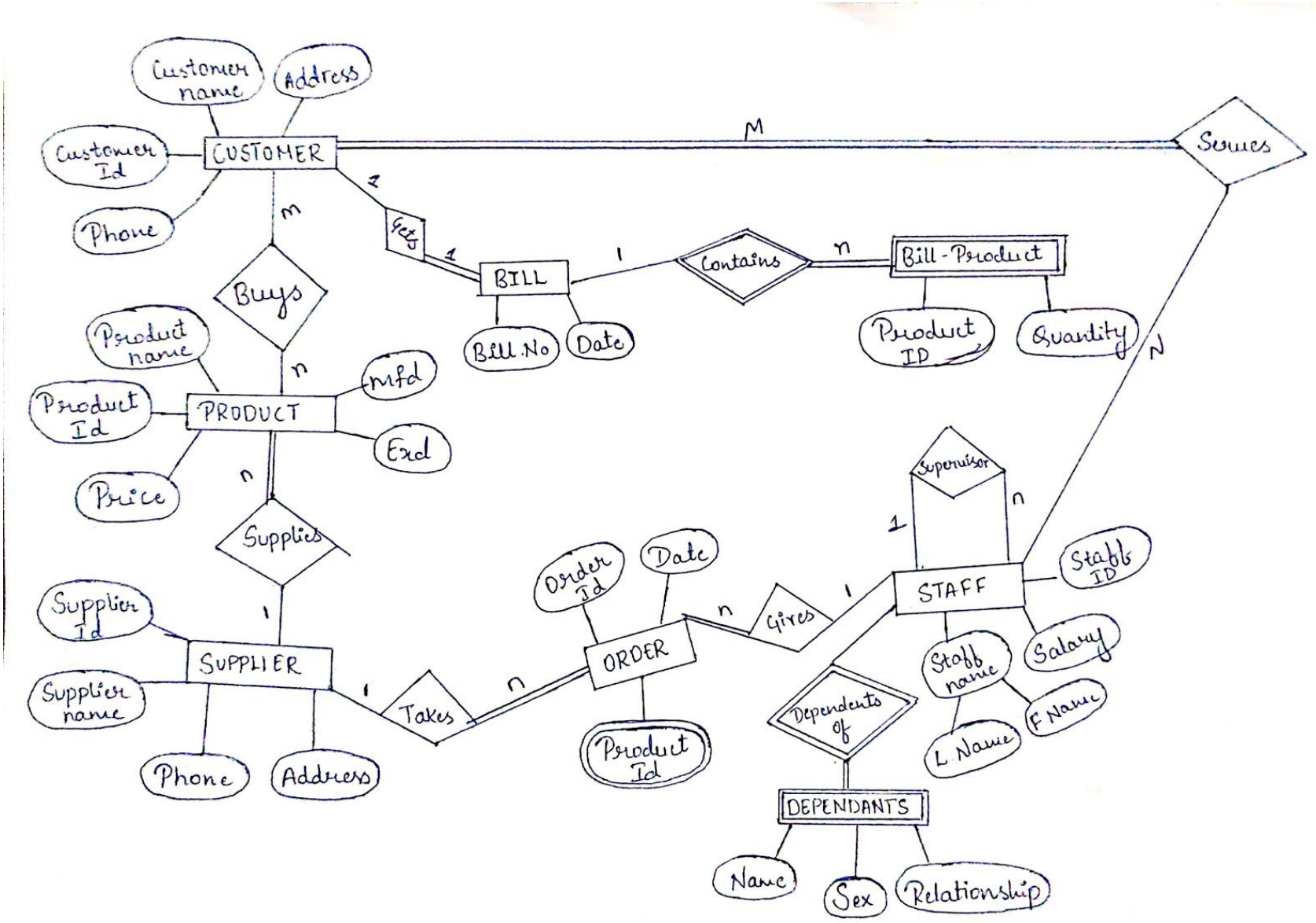
PES2201800058 Prathik B Jain

TABLE OF CONTENT

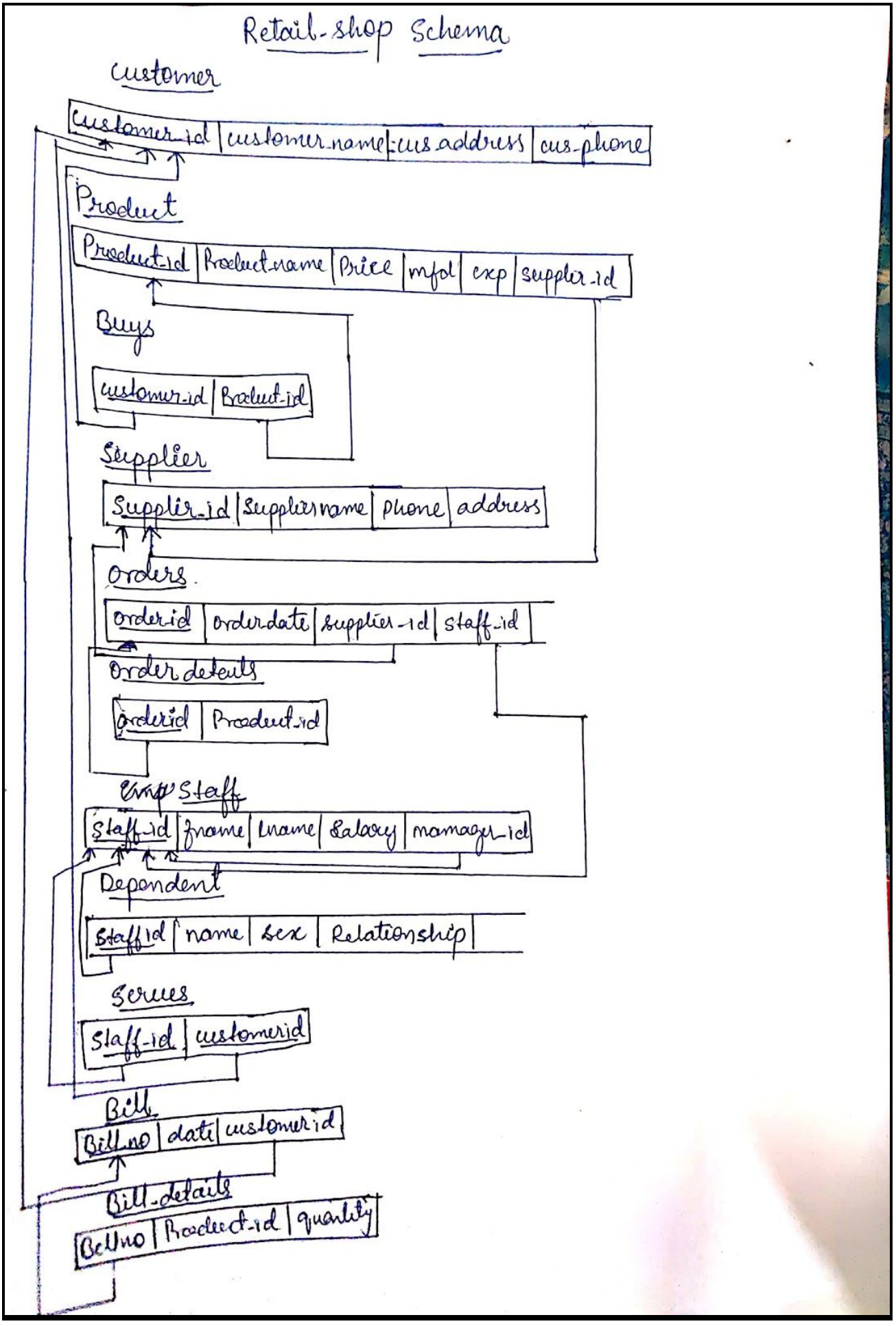
Introduction:

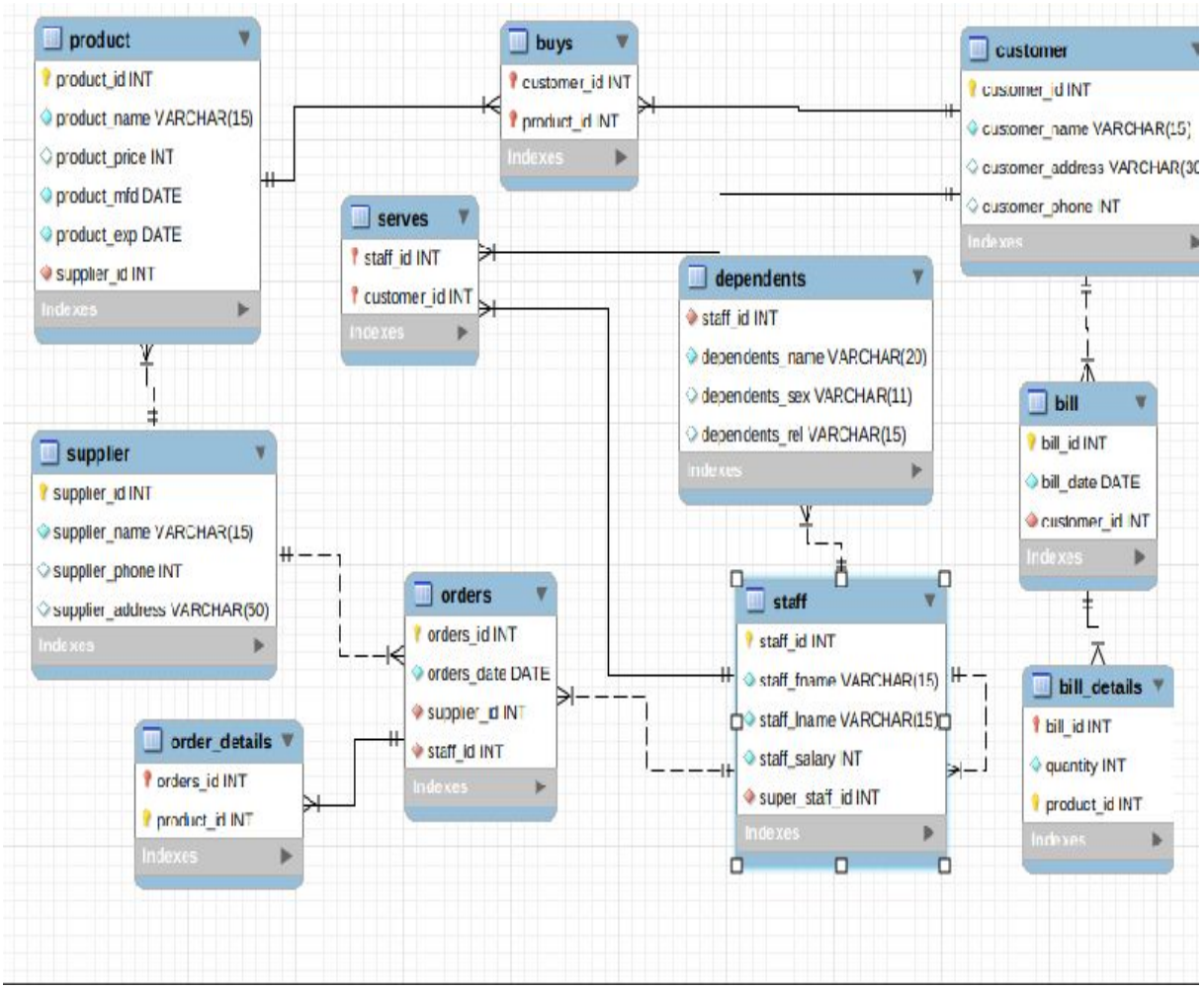
The Retail_shop management system helps the small retail shops to manage their transaction , customer informations ,different orders, bills, And the details about their staff. This system is created to help the retail shop owner to maintain and store the details with ease . Here we store the maximum useful data which can be further analysed by the owner to take certain steps in business and improve the income and also helps in satisfying the customers.

Customers are registered with their bare minimum details such as Name,address and contact number.we give each customer a unique identity code.customers buys products . product informations are stored With their name ,price, manufacturing date and expiry date,which helps the owner by alerting about the expiry date.we give each product a unique id.We also store the details about the supplier who supplies product.supplier supplies products. Details about the order are stored with order id and date .staff gives the order.Information about staff such as their name,salary and their dependents.staff serves the customers. The information about the bills are stored and all the customers gets their bill.


Customer name
Address
Customer Id
CUSTOMER
Phone
M
Serves
m
1
Gets
1
BILL
1
Contains
n
Bill - Product
Buys
Bill No
Date
Product ID
Quantity
N
Product name
Product Id
PRODUCT
Mfd
Exd
Price
n
n
Supplies
Supervisor
1
n
1
Order Id
Date
STAFF
Staff ID
Supplier Id
SUPPLIER
n
Gives
1
Salary
Staff name
Supplier name
1
Takes
n
ORDER
F Name
L Name
Dependents of
Phone
Address
Product Id
DEPENDANTS
Name
Sex
Relationship

Retail-shop Schema





Let us discuss about the tables :

1.Customer:it stores the information about the customers

1.Customer_id : Integer [Primary key]

2.customer_name: varchar

3.phone : integer

4.address: varchar

In this table we have taken customer_id as primary key , because the other attributes cannot assure uniqueness.

2.Product:This table stores the information about the products

1.product_id : Integer [Primary Key]

2.product_name : varchar

3.price: integer

4.mfd: Date

5.exp : Date
6.supplier_id : Integer [Foreign Key]

3. Buys:This tables stores the information about the products bought by the customers

1.customer_id : Integer [Foreign Key]
2. Product_id : Integer [Foreign Key]
Primary Key → {customer_id,product_id}

4.Supplier: This table stores the details about the supplier who supplies the products

1.supplier_id : Integer [Primary Key]
2.suppplier_name: Varchar
3. Phone : Integer
4.address : Varchar

5.orders : This table hold the details about the orders placed by the store staff

1.order_id : Integer [Primary Key]
2.Order_date : Date
3.supplier_id : Integer [Foreign Key]
4.staff_id : Integer [Foreign Key]

6.order-details: As the product is multivalued ,it leads to the repetitions of other attributes.

1.order_id : Integer [Foreign Key, Primary Key]
2. Product_id : Integer

7.Staff: This table stores the information about the staff working in the retail shop.

1. staff_id : Integer [Primary Key]
2. fname : Varchar
3. Lname : varchar
4. Salary: Integer

5.manager_id : Integer [Foreign Key]

8.Dependent:This table stores the information about the people dependent on the staff working in the retail shop.

1.staff_id : Integer [Foreign Key]
2.name : Varchar
3.sex : Varchar
4.Relationship : Varchar

9.Serves: This table keeps track about the information of number of customers attended by each staff .

1.staff_id : Integer [Foreign Key]
2.customer_id : Integer [Foreign Key]
Primary key : { staff_id,customer_id }

10.bill : This table stores the information about the customers bill

1.Bill_number:Integer [Primary Key]
2.Bill_date : Date
3.customer_id : Integer [Foreign Key]

11.Bill_details: This table store the information about quantity of products bought, as this is multivalued we have stored in different table to reduce redundancy

1.bill_number : Integer [Foreign Key]
2.product_id : Integer
3. Quantity : Integer
Primary Key : {bill_number,product_id}

FD and Normalization

As we have followed the approach of converting the ER diagram to schema we have got all the tables Normalized.

Let us discuss about the cases which would lead in violation of Normal forms
Before describing the cases lets us explain the rules of Normal forms

First Normal Form:

1.each column should have atomic values
2.Column should contain values of same data type
3.each column must have unique name
4.order doesn't matter in the table

Second Normal Form:

1.It should be in first normal form
2.There should be no partial dependency.

What is partial dependency?
Consider in a table we have 2 attributes together forming the primary key.In the same table if we can uniquely identify any other column with the help of any one of those attributes ,then it is said to have partial dependency.

Let us consider the table 'buys' from the RETAIL_SHOP database.
Here ''customer_id' and 'product_id' together forms the primary key.
Imagine if we add product name to the table then it leads to partial dependency because we can identify the column product name uniquely

with the help of product_id alone hence we shouldn't consider it in the table.

Similarly if we add a customer name in the same table we can identify the column customer name alone with the help of customer_id and not the composite pair of primary key,hence if added leads to partial dependency.

Lets us consider another table bill_details from the RETAIL_SHOP database
Here we have 3 columns bill_number, product_id, quantity.here the primary key is the pair
{bill_number,product_id }.Here if we add product name then the attribute product_id can uniquely find out the column values hence it leads to partial dependency and violates second normal form.

To solve this we have many ways. Let us follow a simple method for the above table
We can remove the column product_name and make a separate table with product_id
And product_name.

Third Normal Form:

1.It should be in second normal form
2.There should be no transitive dependency

What is transitive dependency?
Consider a table in which we have a column which is not dependent on the primary key but depends on another column of the table.

Let us consider the table orders from the RETAIL_SHOP database.

The table orders have the following attributes
-order_id,order_date,supplier_id,staff_id.
The primary key is ‘order-id’.To this table if we add a new column supplier_name then it leads to transitive dependency,because the supplier name depends on the supplier_id and not on the primary key.

Let us consider another example for the table bill.
In this table we have the primary key as bill_no.imagine we add a new column customer name,then it leads to transitive dependency violating the third normal form because the customer name depends on customer_id and not on the primary key.

Functional Dependency:
It is used to specify formal measures of the ‘goodness’ of relational design.
A set of attributes X functionally determines a set of attributes Y ,if the value ofX determines the unique value of Y.

Functional Dependency of this database are listed below:
1.customer_id ---> { customer_name,customer_phone,customer_address }
2.Product_id ----> { Product_name,Product_price,Product_mfd,Product_exp }
3.supplier_id ---> { supplier_name,supplier_phone,supplier_address }
4.order_id ---> { order_date,supplier_id,staff_id }
5.staff_id ---> { staff_name,staff_salary,staff_phone,staff_address,staff_manager }
6.bill_no ---> { bill_date,customer_id }
7.{ bill_no,product_id } ---> { quantity }

Testing for lossless join property

When we decompose a Relation R to {R1,R2} we need to make sure that there is no loss of data when we split the relations/tables

In this system as prouct_id in orders is a multivalued we have split them into 2 tables to reduce the dependency.we have reduced the dependency now we need to check that we didn't lose any data.

The tables
Order : {order_id,date,supplier_id,emp_id}
Order_details : {order_id,product_id}

Let X : R1 intersection R2 = order_id
Let Y : R1 - R2 = {date,supplier_id,emmp_id}

We can clearly see with the help of functional dependency that X → Y
Hence we can say that there is no loss of data due to binary decomposition.

DDL:Data Definition Language

```
CREATE DATABASE `RETAIL_SHOP`;
USE RETAIL_SHOP;

CREATE TABLE `RETAIL_SHOP`.`customer` (
        `customer_id` INT(11) NOT NULL,
        `customer_name` VARCHAR(15) NOT NULL,
        `customer_address` VARCHAR(30) NULL DEFAULT NULL,
        `customer_phone` INT(11) NULL DEFAULT NULL,
        PRIMARY KEY (customer_id)
);

CREATE TABLE `RETAIL_SHOP`.`product` (
        `product_id` INT(11) NOT NULL,
        `product_name` VARCHAR(15) NOT NULL,
        `product_price` INT(11) NULL,
        `product_mfd` DATE NOT NULL,
```

```
	`product_exp` DATE NOT NULL,
	`supplier_id` INT(11) NOT NULL,
	PRIMARY KEY (product_id)
);

CREATE TABLE `RETAIL_SHOP`.`buys` (
	`customer_id` INT(11) NOT NULL,
	`product_id` INT(11) NOT NULL,
	FOREIGN KEY (customer_id)
	REFERENCES customer (customer_id),
	FOREIGN KEY (product_id)
	REFERENCES product (product_id)
);


CREATE TABLE `RETAIL_SHOP`.`supplier` (
	`supplier_id` INT(11) NOT NULL,
	`supplier_name` VARCHAR(15) NOT NULL,
	`supplier_phone` INT(11) NULL DEFAULT NULL,
	`supplier_address` VARCHAR(50) NULL DEFAULT NULL,
	PRIMARY KEY (supplier_id)
);

ALTER TABLE product
ADD CONSTRAINT supplier_id
FOREIGN KEY (supplier_id) REFERENCES
supplier(supplier_id);

ALTER TABLE buys
ADD CONSTRAINT PK PRIMARY KEY
(customer_id,product_id);

CREATE TABLE `RETAIL_SHOP`.`orders` (
	`orders_id` INT(11) NOT NULL,
	`orders_date` DATE NOT NULL,
```

```
	`supplier_id` INT(11) NOT NULL,
	`staff_id` INT(11) NOT NULL,
	PRIMARY KEY (orders_id),
	FOREIGN KEY (supplier_id)
	REFERENCES supplier (supplier_id)
);



CREATE TABLE `RETAIL_SHOP`.`order_details` (
	`orders_id` INT(11) NOT NULL,
	`product_id` INT(11) NOT NULL,
	PRIMARY KEY (orders_id , product_id),
	FOREIGN KEY (orders_id)
	REFERENCES orders (orders_id)
);

CREATE TABLE `RETAIL_SHOP`.`staff` (
	`staff_id` INT(11) NOT NULL,
	`staff_fname` VARCHAR(15) NOT NULL,
	`staff_lname` VARCHAR(15) NOT NULL,
	`staff_salary` INT(8) NOT NULL,
	`super_staff_id` INT(11) NOT NULL,
	PRIMARY KEY (staff_id),
	FOREIGN KEY (super_staff_id)
	REFERENCES staff (staff_id)
);

ALTER TABLE orders
ADD CONSTRAINT staff_id
FOREIGN KEY (staff_id) REFERENCES
staff(staff_id);



CREATE TABLE `RETAIL_SHOP`.`dependents` (
```

```
	`staff_id` INT(11) NOT NULL,
	`dependents_name` VARCHAR(20) NOT NULL,
	`dependents_sex` VARCHAR(11) NULL DEFAULT NULL,
	`dependents_rel` VARCHAR(15) NULL DEFAULT NULL,
	FOREIGN KEY (staff_id)
	REFERENCES staff (staff_id)
);


CREATE TABLE `RETAIL_SHOP`.`serves` (
	`staff_id` INT(11) NOT NULL,
	`customer_id` INT(11) NOT NULL,
	PRIMARY KEY (staff_id , customer_id),
	FOREIGN KEY (staff_id)
	REFERENCES staff (staff_id),
	FOREIGN KEY (customer_id)
	REFERENCES customer (customer_id)
);

CREATE TABLE `RETAIL_SHOP`.`bill` (
	`bill_id` INT(11) NOT NULL,
	`bill_date` DATE NOT NULL,
	`customer_id` INT(11) NOT NULL,
	FOREIGN KEY (customer_id)
	REFERENCES customer (customer_id),
	PRIMARY KEY (bill_id)
);

CREATE TABLE `RETAIL_SHOP`.`bill_details` (
	`bill_id` INT(11) NOT NULL,
	`quantity` INT(11) NOT NULL CHECK (quantity > 0),
	`product_id` INT(11) NOT NULL,
	PRIMARY KEY (bill_id , product_id),
	FOREIGN KEY (bill_id)
	REFERENCES bill (bill_id)
```

```
);


ALTER TABLE product
ADD CHECK (product_mfd<product_exp);

SHOW  TABLES;


insert into customer values
(10001,'Dhanraj','church road hyr',968642524),
(10002,'shreyas','BJRextension hyr',810590798),
(10003,'preetham','teachers colony blore',988071991),
(10004,'koushik','channel road hosur',960627894),
(10005,'Prathik','nehru circle blore',914848962);


insert into supplier values
(30001,'sandeep',984530070,'ratandeep shimoga'),
(30002,'vishal',962000003,'di-fabri india'),
(30003,'praveen',94812782,'chamrajpet blore'),
(30004,'neha',962015163,'kavdi dugra'),
(30005,'santhosh',966327838,'MM mohan hyr');

insert into product values
(20001,'parle-g',10,'2020-5-10','2021-5-10',30001),
(20002,'drak_fantacy',30,'2020-5-10','2021-7-10',30002),
(20003,'milkbikis',40,'2020-5-10','2021-7-10',30003),
(20004,'hide_n_seek',25,'2020-5-10','2021-5-10',30004),
(20005,'bourbon',50,'2020-5-10','2021-7-10',30005);


insert into buys values
(10001,20001),
(10003,20002),
```

```
(10002,20002);


insert into staff values
(50001,'sharan','keshav',10000,'50001'),
(50002,'varun','keshav',10000,'50001'),
(50003,'guru','kiran',8000,'50003'),
(50004,'manu','kumar',7000,'50003'),
(50005,'sujay','jain',5000,'50003');

insert into orders values
(60001,'2020-4-15',30001,50001),
(60002,'2020-4-17',30001,50001),
(60003,'2020-4-20',30002,50003);

insert into order_details values
(60001,20001),
(60002,20001),
(60003,20003);

show databases;
show tables;
SELECT
        *
FROM
        product;
desc dependents;

insert  into dependents value
(50001,'samarth','male','son'),
(50002,'samrudh','male','brother'),
(50005,'suhas','male','brother');


insert into serves values
```

```
(50001,10001),
(50002,10002),
(50003,10003);

insert into bill values
(1,'2020-5-23',10001),
(2,'2020-5-23',10002),
(3,'2020-5-23',10003);

insert into bill_details values
(1,5,20001),
(1,5,20003),
(2,5,20005),
(2,10,20002),
(3,8,20003);
```

Integrity Constraint:

1. Domain constraint
2. entity integrity Constraint
3. Referential integrity constraint
4. key constraint

In the above code snippet you can see that all the 4 types of Integrity constraint are followed.

```
# Definition of valid set of values for an attribute
# Values of the attributes are in the corresponding domain
# It is made sure that primary keys are not null
# All the foreign keys are linked
#Primary keys has unique value and are not null
```

Triggers

A trigger is a stored program invoked automatically in response to an event such as insert, update, or delete that occurs in the associated table. For example, you can define a trigger that is invoked automatically before a new row is inserted into a table.

Let us see a few examples about triggers

1.

```
DELIMITER $$
CREATE TRIGGER check_expiry_date
AFTER INSERT
ON product FOR EACH ROW
BEGIN
        IF NEW.product_exp < current_date() THEN

    signal sqlstate VALUE '45000' set message_text = 'Product is expired
';



        END IF;
END$$
DELIMITER ;
```

In the above trigger , we are checking the expiry date of the product before inserting into the store.If the product is expired then it raises an error and alerts the shop owner.

2.

```
DELIMITER $$
CREATE TRIGGER before_staff_delete
BEFORE DELETE
ON `staff` FOR EACH ROW
BEGIN
        INSERT INTO `staff_backup`(`staff_id`,
`staff_fname`,`staff_lname`,`staff_salary`,`super_staff_id` ) VALUES
```

```
(OLD.staff_id,OLD.staff_fname,OLD.staff_lname,OLD.staff_salary,OLD.
super_staff_id);
END$$
DELIMITER ;
```

In this trigger we are creating a backup table for staff data where after deleting details from the table they are stored in another backup table.

3.

```
DELIMITER $$
CREATE TRIGGER display_improper_salary1
BEFORE insert
ON staff FOR EACH ROW

BEGIN
        if (new.staff_salary < 1) then
        signal sqlstate VALUE '45000' set message_text = 'salary cannot
be Zero ';
        end if;
END$$
DELIMITER ;
```

In this trigger we are taking care that whenever we insert or update the salary of the staff we are taking care that it is not set to improper value.

4.

```
DELIMITER $$
CREATE TRIGGER increment_salary
AFTER INSERT
ON staff FOR EACH ROW
BEGIN
        IF datediff(NOW(),old.staff_join_date) = 365 or
datediff('2019-06-02', NOW()) = 730 THEN
        set staff.staff_salary = staff.staff_salary + 1000;
```

```
        END IF;
END$$
DELIMITER ;
```

In this trigger we are incrementing the salary of the staff for 2 times once in a year and again after a year.

SQL Queries

1.

```
# customers served by employees
select c.customer_id, c.customer_name
from serves as s ,customer as c
where s.customer_id = c.customer_id;
```

In this simple query we are printing the list of customers and the employee who served them.

2.

```
#printing bill details
select p.product_name,bd.quantity,
p.product_price,p.product_price*bd.quantity
from  bill_details as bd, product as p
where bd.product_id = p.product_id and bd.bill_id = 1
order by p.product_name , p.product_price;
union
select sum(p.product_price*bd.quantity) as total_price
from  bill_details as bd, product as p
where bd.product_id = p.product_id and bd.bill_id = 1;
```

In this above query we are printing the bill of the customer with complete details of price and product name and product quantity

3.
to find total number of times the products is sold [liked by customers]

```
 CREATE VIEW
customers_favourite(number_of_customer_who_bought,productname)
    as select count(*) as number_of_pieces_got_saled,p.product_name
     from buys as B,product as p
     where B.product_id=p.product_id
     group by(p.product_name) ;


 show tables;
select *
from customers_favourite;
```

In the above query we are creating a view about the information about the number of times a product is bought by different customers and we can make a study about it.It also helps us in placing proper order for the shop.

```
4.#name of the staff who placed the order
SELECT
        e.staff_fname, e.staff_lname
FROM
        staff AS e
WHERE
        EXISTS( SELECT
        e.staff_fname, e.staff_lname
        FROM
```

```
        supplier AS s,
        orders AS o
        WHERE
        s.supplier_name = 'vishal'
                AND s.supplier_id = o.supplier_id
                AND o.staff_id = e.staff_id);
```

In the above query we can find out the staff who has placed the order.

```
5.# staff who served a particular customer
SELECT
        e.staff_fname, e.staff_lname
FROM
        staff AS e
WHERE
        EXISTS( SELECT
        e.staff_fname, e.staff_lname
        FROM
        serves AS s,
        customer AS c
        WHERE
        c.customer_name = 'preetham'
                AND c.customer_id = s.customer_id
                AND s.staff_id = e.staff_id);
```

The above query helps us to find out the staff who served that particular customer .This table helps us to maintain the records about how many customers are attended by a particular staff and his importance

Conclusion

The retails shop database management system is very important system .As we know there are many small scale retail shops ,this system helps them just not only manage their database but also helps in

improving their business .We can find out the the most liked product in the shop and make sure the product is always available in the shop.We can also check the worth of the staff by checking the number of customers attended by him/her in a particular duration of time and many more.

We can still improvise the system and make it a better application. We can also keep the track about the quantity of product in the shop and create triggers to automatically and intelligently alert the owner when the quantity of a product is less inn the shop .We can also check if the product is about to get expired and if we have a huge quantity we can alert the owner so that he somehow gets rid of those product and get prevented from the loss.We can still improvise and make it better