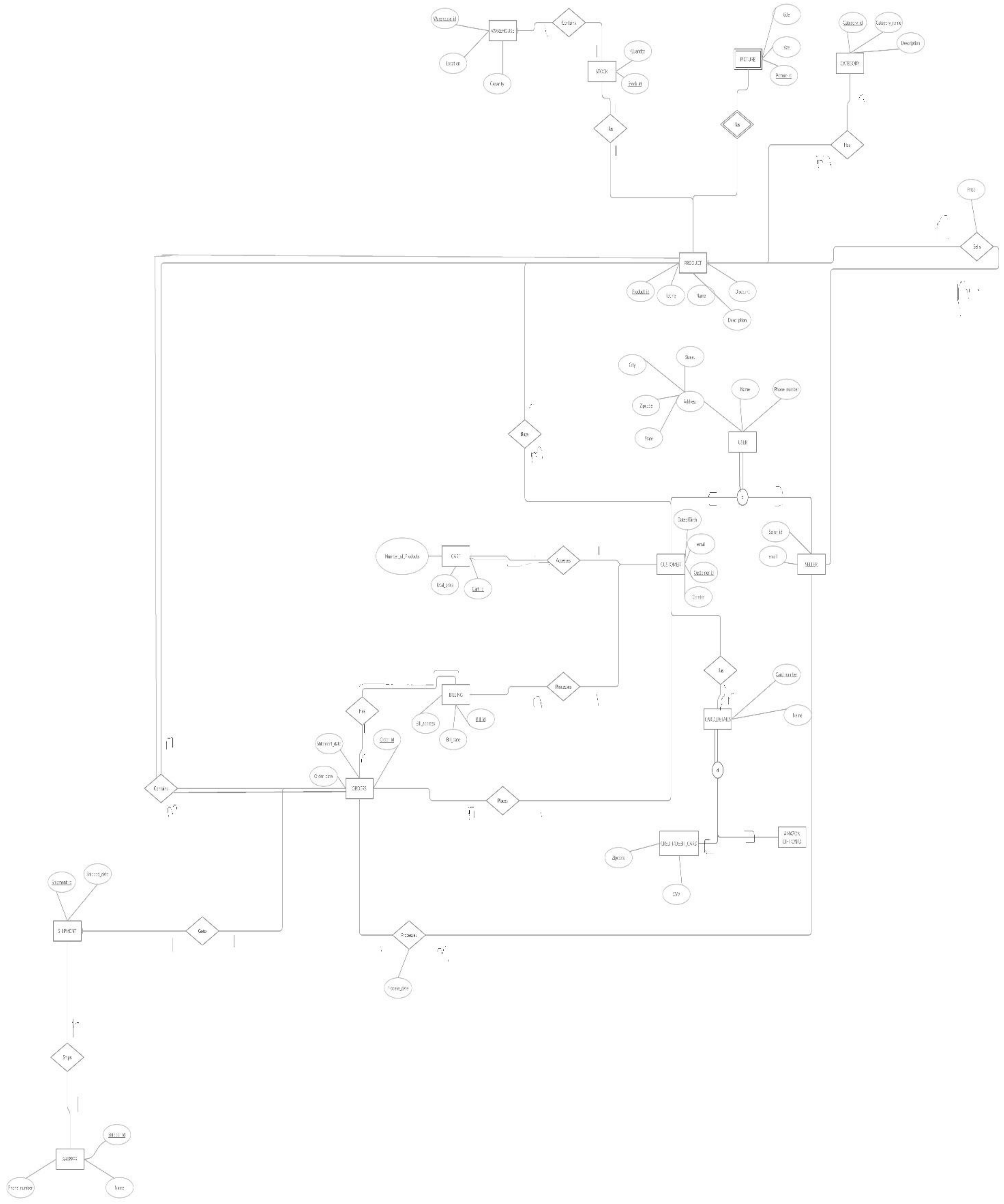
AMAZON-4

Course/Section	:	CS 6360.002
Team number	:	4
Team members	:	Satya Sarvani Baru (SXB180153)
		Sai Teja Peddi (SXP180107)

- **Requirements:**

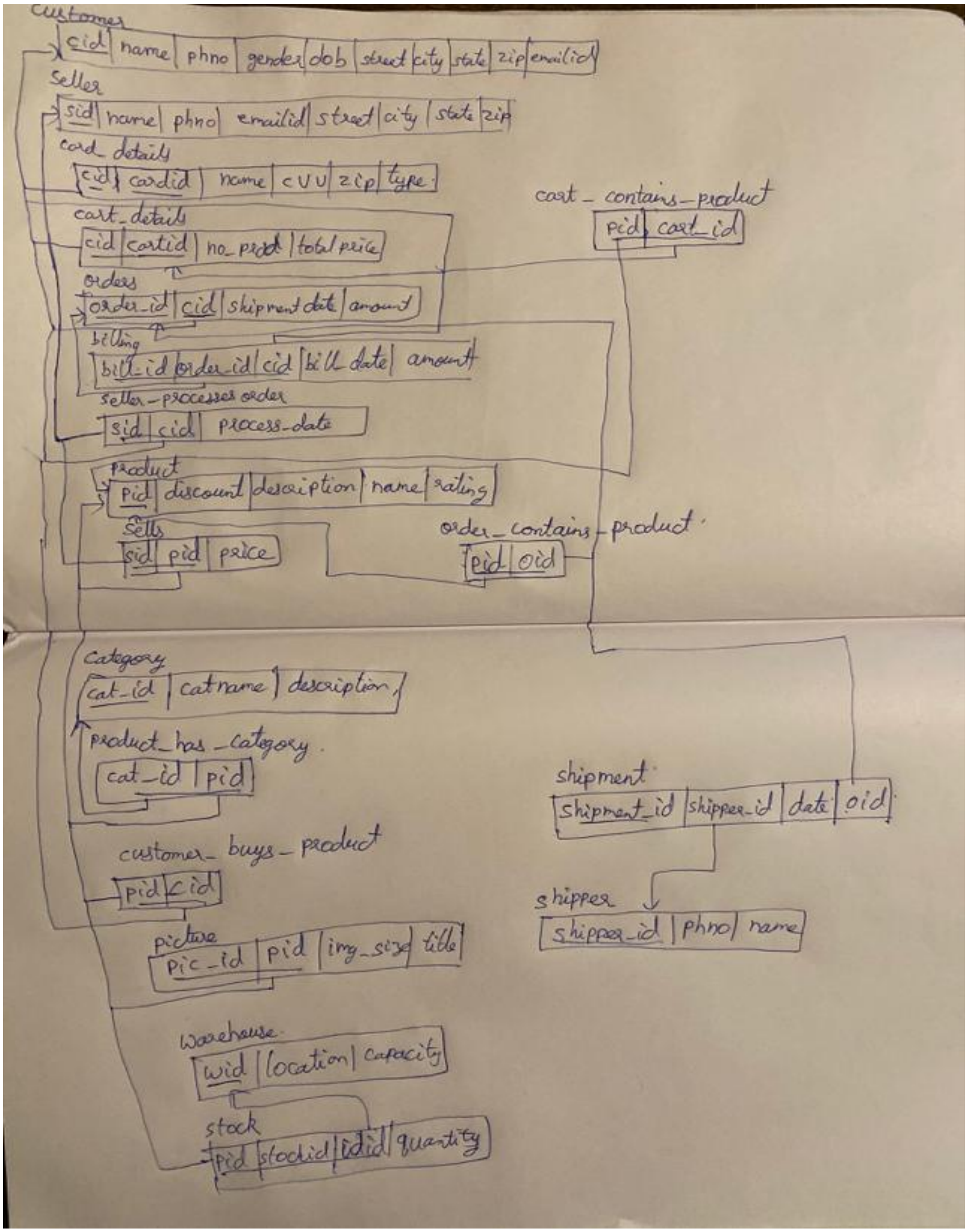
  - Amazon is an online e-commerce website that deals with several kind of products which can be supplied to customers as per their requirement.
  - Few assumptions were made while designing this project.
  - There is a user who can either be a seller or a customer.
  - Each user can have name, contact number, address. Address is a composite attribute consisting of street, city, state, zip.
  - Each customer has 1 cart and the cart can have cart id and product details.
  - Each customer can have ‘n’ card details which can be a credit/debit card or amazon gift card. customers can buy more than 1 products.
  - Similarly, sellers can sell many products. A customer can add a product to cart. Also, a customer can process ‘n’ billings.
  - Product contains product id, name, description, rating.
  - Each product can be divided into 1or more categories and each category can have more than 1 product.
  - A customer can place many orders which can be processed by many sellers with different process dates.
  - Order contains 1 or more products and each product can be sold in more than 1 order.
  - Each order goes to each shipment on a specific shipment date and they are shipped by a shipper as per requirement.
  - Each shipment is shipped by a shipper with unique shipment id.
  - Shipment consists of shipment id, shipper id, shipment date.
  - Each product has a picture (ISA relationship) that is displayed with its title and size.
  - Various products have various categories with their corresponding names and description.
  - Each product has a stock that indicates quantity or availability of that product.
  - Each warehouse is present in a location with a certain capacity.

- **EER Diagram:**



Note: EER Diagram need to be zoomed to view it clearly. (Also attached separately)

- **RELATIONAL SCHEMA:**


customer
cid | name | phno | gender | dob | street | city | state | zip | emailid
seller
sid | name | phno | emailid | street | city | state | zip
card_details
cid | cardid | name | cvv | zip | type
cart_details
cid | cartid | no_prod | total price
cart_contains_product
pid | cart_id
orders
order_id | cid | shipment date | amount
billing
bill_id | order_id | cid | bill date | amount
seller_processes order
sid | cid | process_date
product
pid | discount | description | name | rating
sells
sid | pid | price
order_contains_product
pid | oid
category
cat_id | catname | description
product_has_category
cat_id | pid
shipment
shipment_id | shipper_id | date | oid
shipper
shipper_id | phno | name
customer_buys_product
pid | cid
picture
pic_id | pid | img_size | title
warehouse
wid | location | capacity
stock
pid | stockid | wid | quantity

- **SQL:**

```
create table customer ( cid varchar (25) primary key, name varchar (30) not null, phno varchar
(25), gender varchar (10), dob date,street varchar (30), city varchar (25),  state varchar (25), zip
int,  emailid varchar (50) not null    );
    desc customer;
    create table seller (  sid varchar (25) primary key,name varchar (30) not null,phno varchar (25),
    emailid varchar (50) not null,street varchar (30), city varchar (25), state varchar (25), zip int );
    desc seller;

create table card_details
    (cid varchar (25), cardid varchar (25), name varchar (25) not null,
    cvv varchar (25) not null, zip int not null, type varchar (20), primary key (cid, cardid))
    alter table card_details add constraint crd foreign key(cid) references customer(cid) ON
DELETE SET NULL;
desc card_details;
    create table cart_details
    (cid varchar (25), cartid varchar (25), no_prod int,
    totalprice int, primary key (cid, cartid))
    alter table cart_details add constraint crt foreign key(cid) references customer(cid) ON
DELETE SET NULL;
    create table orders (order_id varchar (25) primary key, cid varchar (25) not null, shipment_date
varchar (25), amount int);
    alter table orders add constraint ord foreign key(cid) references customer(cid) ON DELETE
SET NULL;
    create table billing (bill_id varchar (25), order_id varchar (25, cid varchar (25), bill_date date,
amount int, primary key (bill_id, order_id, cid));
    alter table billing add constraint bc foreign key(cid) references customer(cid) ON DELETE
SET NULL;
    alter table billing add constraint bc1 foreign key(order_id) references orders(order_id) ON
DELETE SET NULL;
 create table seller_process_order (sid varchar (25), cid varchar (25), process_date date, primary
key (sid, cid));
     alter table seller_process_order add constraint spo foreign key(sid) references seller(sid) ON
DELETE SET NULL;
     alter table seller_process_order add constraint spo1 foreign key(cid) references customer(cid)
ON DELETE SET NULL;
create table product (pid varchar (25), discount int, description varchar (50), name varchar (25),
rating varchar (25), primary key(pid));
create table sells (sid varchar (25), pid varchar (25), price int, primary key (sid, pid));
     alter table sells add constraint sellc foreign key(sid) references seller(sid) ON DELETE SET
NULL;
     alter table sells add constraint sellc1 foreign key(pid) references product(pid) ON DELETE
SET NULL;
create table order_contains_product (pid varchar (25), oid varchar (25), primary key (pid, oid));
alter table order_contains_product add constraint ocp foreign key(pid) references product(pid)
ON DELETE SET NULL;
```

```
alter table order_contains_product add constraint ocp1 foreign key(oid) references
orders(order_id) ON DELETE SET NULL;
create table cart_contains_product (pid varchar (25), cart_id varchar (25), primary key (cart_id,
pid));
alter table cart_contains_product add constraint cp foreign key(pid) references product(pid) ON
DELETE SET NULL;

create table category (cat_id varchar (25) primary key, cat_name varchar (25), description varchar
(25));
create table product_has_category (cat_id varchar (25), pid varchar (25), primary key (cat_id,
pid));
alter table product_has_category add constraint pc foreign key(pid) references product(pid) ON
DELETE SET NULL;
alter table product_has_category add constraint pc1 foreign key(cat_id) references
category(cat_id) ON DELETE SET NULL;
create table customer_buys_product (pid varchar (25), cid varchar (25), primary key (cid, pid));
alter table customer_buys_product add constraint cbp foreign key(pid) references product(pid)
ON DELETE SET NULL;
alter table customer_buys_product add constraint cbp1 foreign key(cid) references customer(cid)
ON DELETE SET NULL;

create table picture (pic_id varchar (25), img_size int, title varchar (25), pid varchar (25), primary
key (pic_id, pid));
alter table picture add constraint pic foreign key(pid) references product(pid) ON DELETE SET
NULL;
create table shipper (shipper_id varchar (25) primary key, phno varchar (25), name varchar (25));
create table shipment (shipper_id varchar (25) primary key, shipment_id varchar (25),
shipped_date date, oid varchar (25));
alter table shipment add constraint sh foreign key(shipper_id) references shipper(shipper_id) ON
DELETE SET NULL;
alter table shipment add constraint sh1 foreign key(oid) references orders(order_id) ON DELETE
SET NULL;
create table warehouse (wid varchar (25) primary key, location varchar (25), capacity int);
create table stock (pid varchar (25), stockid varchar (25), wid varchar (25), quantity int, primary
key (pid, stockid, wid));
alter table stock add constraint s1 foreign key(pid) references product(pid) ON DELETE SET
NULL;
alter table stock add constraint s2 foreign key(wid) references warehouse(wid) ON DELETE SET
NULL;
```

- **PL-SQL:**

<u>TRIGGERS:</u>

<u>/*Trigger for storing the change in discounts of a product. If the discounts of the products are changed then its stored in discount_log table which logs the old and new discount.*/</u>

```
CREATE TABLE discount_log (
  pid     VARCHAR (9),
  New_discount NUMBER,
  Old_discount NUMBER,
  Log_date   DATE
);


create or replace TRIGGER Log_discount_changes
  AFTER UPDATE OF discount ON product
  FOR EACH ROW
BEGIN
  INSERT INTO discount_log (pid, New_discount, Old_discount, Log_Date)
  VALUES (:new.pid, :new.discount, :old.discount, SYSDATE);
END;
```

<u>/* stock change log. This log stores the change in stock of the items along with the date.*/</u>

```
CREATE TABLE stock_log (
  stockid     VARCHAR(9),
  New_stock NUMBER,
  Old_stock NUMBER,
  Log_date   DATE
);


create or replace TRIGGER Log_stock_changes
  AFTER UPDATE OF quantity ON stock
  FOR EACH ROW
BEGIN
  INSERT INTO discount_log
  VALUES (:new.stockid, :new.quantity, :old.quantity, SYSDATE);
END;
```

<u>STORED PROCEDURES:</u>

<u>/*set discount as 50 when discount>50. Assume that we want to limit the maximum discount to 50, then decrease the discount of all the items to 50*/</u>

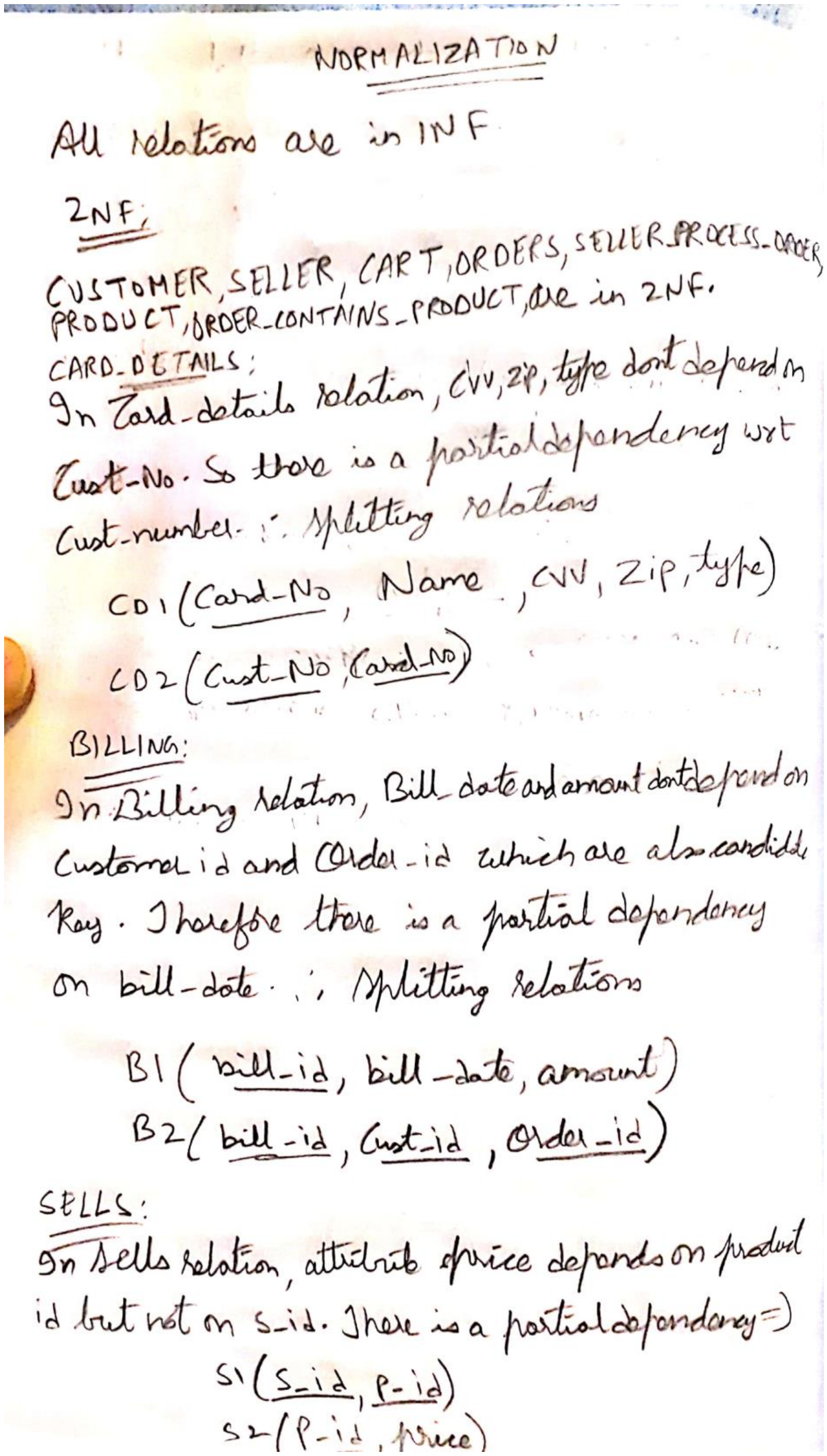
```
create or replace PROCEDURE Decrease_discount AS
```

```
thisProduct Product%ROWTYPE;

CURSOR discount_50 IS
SELECT * FROM product where discount>50 FOR UPDATE;

BEGIN
OPEN discount_50;
LOOP
  FETCH discount_50 INTO thisProduct;
  EXIT WHEN (discount_50%NOTFOUND);
  dbms_output.put_line(thisProduct.pid);
  dbms_output.put_line (thisProduct.discount);
  UPDATE product SET discount = 50
  WHERE CURRENT OF discount_50;

END LOOP;
CLOSE discount_50;
END;
```

/* Procedure that will change the price of a particular product sold by a particular seller. This procedure updates the price of the product belonging to a particular seller.*/

```
create or replace PROCEDURE Change_Price (pid IN sells.pid%TYPE, sid IN sells. %type,
newprice int) AS

thisSells sells%ROWTYPE;

CURSOR sells_c IS
SELECT * FROM sells;

BEGIN
OPEN sells_c;
LOOP
FETCH sells_c INTO thisSells;
EXIT WHEN (sells_c%NOTFOUND);
update sells set price=newprice where sid=sid and pid=pid;
dbms_output.put_line (' Price has been updated');

END LOOP;CLOSE sells_c;
END;
```

NORMALIZATION:



NORMALIZATION

All relations are in 1NF

2NF;

CUSTOMER, SELLER, CART, ORDERS, SELLER_PROCESS_ORDER, PRODUCT, ORDER_CONTAINS_PRODUCT, are in 2NF.

CARD_DETAILS;

In Card-details relation, CVV, zip, type dont depend on Cust-No. So there is a partial dependency wrt Cust-number ∴ Splitting relations

CD1(<u>Card-No</u>, Name, CVV, Zip, type)

CD2(<u>Cust-No</u>, <u>Card-No</u>)

BILLING:

In Billing relation, Bill_date and amount dont depend on Customer id and Order-id which are also candidate key. Therefore there is a partial dependency on bill-date. ∴ Splitting relations

B1(<u>bill-id</u>, bill-date, amount)

B2(<u>bill-id</u>, <u>Cust-id</u>, <u>Order-id</u>)

SELLS:

In Sells relation, attribute price depends on product id but not on s-id. There is a partial dependency ⇒

S1(<u>s-id</u>, <u>P-id</u>)

S2(P-id, price)

CART_CONTAINS_PRODUCT is already in 2NF.
CATEGORY relation is already in 2NF.
PRODUCT_HAS_CATEGORY relation is already in 2NF.
CUSTOMER_BUYS_PRODUCT relation is already in 2NF.

PICTURE:

In Picture relation, img_size attribute depends on pic_id and title attribute depends on product id. So there is partial dependency =)

P1(<u>product_id</u>, title)
P2(<u>pic_id</u>, size, <u>product_id</u>)

SHIPPER relation is already in 2NF. Warehouse relation is already in 2NF. Stock relation is already in 2NF. SHIPMENT relation is already in 2NF.

3NF:

CARD_DETAILS, BILLING, SELLS, PICTURE relations are already in 3NF, ORDERS is also in 3NF.

In Customer, relation, Street, city, State and Zip code donot depend on Cust_id. There is a transitive dependency that gets created between Cust_id, street, city, state, zipcode =)

C1(<u>C_id</u>, Name, Ph_No, Gender, dob, email id)

C2(street, city, state, <u>Zipcode</u>)

In seller relation also, there is a transitive dependency =)

S1(<u>Sid</u>, Name, Phno, emailid)

S2(street, city, state, <u>zip</u>)

<u>CART_DETAILS</u>:

In this relation, total price depends on no-prod which is a non-prime attribute. =)

Cart-d1(<u>cid</u>, <u>cartid</u>)

Cart-d2(<u>no-prod</u>, totalprice)

SELLER_PROCESS_ORDER, PRODUCT, ORDER_CONTAINS_PRODU
CART-CONTAINS_PRODUCT, CATEGORY, PRODUCT_HAS_CATEG
CUSTOMER_BUYS_PRODUCT, SHIPPER, SHIPMENT, WAREHOUSE,
STOCK relations are already in 3NF.

∴ Final <u>relational schema</u>:

<u>Customer</u>

C1(<u>C_id</u>, Name, PhNo, gender, dob, emailid)

C2(street, city, state, <u>zipcode</u>)

<u>SELLER</u>:

S1(<u>Sid</u>, Name, PhNo, emailid)

S2(street, city, state, <u>zip</u>)

<u>CARD-DETAILS</u>:

CD1(<u>Card_No</u>, Name, cvv, zip, type)

CD2(Cust-No, Card-No)

CART_DETAILS:

Cart d1(cid, Cartid)

Cart_d2(no.prod , total price)

ORDERS:

ORDERS(Order_id , Cust id, amount, Shipment date)

BILLING:

B1(bill_id , bill date, amount)

B2(bill_id , Cust_id, Order_id)

SELLER_PROCESS ORDER:

SPO(sid, cid ,process date)

PRODUCT:

P(Pid ,discount, description, name, rating)

SELLS:

S1 (S_id , P_id)

S2 (P_id, price)

ORDER_CONTAINS_PRODUCT:

OCP(pid , oid)

CART-CONTAINS_PRODUCT:

CCP(pid , cid)

CATEGORY:

C (cat_id, cat_name, description)

PRODUCT_HAS_CATEGORY:

PHC(cat_id , pid)

CUSTOMER_BUYS_PRODUCT:

CBP($\underline{\text{pid}}$, $\underline{\text{cid}}$)

PICTURE:

P1($\underline{\text{pid}}$, title)

P2($\underline{\text{pid}}$, $\underline{\text{pic-id}}$, size)

SHIPPER:

SH($\underline{\text{shipper-id}}$, phno, name)

SHIPMENT:

SHIPM($\underline{\text{shipper-id}}$, shipment_id, oid, shipped-date)

WAREHOUSE:

W($\underline{\text{wid}}$, location, capacity)

STOCK:

st($\underline{\text{pid}}$, $\underline{\text{stockid}}$, $\underline{\text{wid}}$, quantity)