Triggers

1)

```
create or replace function supply_to_stock()
returns trigger as $$
DECLARE
        var supply_record.qty%type;
        s items.stock%type;
BEGIN
        IF(TG_OP='INSERT') THEN
                var:=new.qty;
                select stock into s from items where itemcode=new.itemcode;
                update items set stock=s+var where itemcode=new.itemcode;
        ELSIF(TG_OP='UPDATE') THEN
                var:=old.qty;
                select stock into s from items where itemcode=old.itemcode;
                update items set stock=s-var where itemcode=new.itemcode;
                var:=new.qty;
                select stock into s from items where itemcode=new.itemcode;
                update items set stock=s+var where itemcode=new.itemcode;
        END IF;
RETURN NULL;
END;
$$ LANGUAGE 'plpgsql';


create trigger supply_to_stock
after insert or update on supply_record
for each row execute procedure supply_to_stock();
```

2)

```
create or replace function discounts()
returns trigger as $$
```

```
DECLARE
        dis numeric(9,2);
        dcode bill_details.discount_applied%type :=NULL;
        r record;
        pp bill_details.purchaseprice%type;
        tem numeric(9,2);
        d bill.bill_date%type;
BEGIN
                        SELECT bill_date INTO d from bill where invno=new.invno;
                        dis:=0.00;
                        select mrp into tem from items where itemcode=new.itemcode;
                        for r in select * from (discount_products as dp join discount as dis on
dp.discount_code=dis.code) where itemcode=new.itemcode and d<=dis.valid_till and
d>=dis.valid_from and dp.qty<=new.qty
                        LOOP

                                if(tem*(new.qty)-
tem*(mod(new.qty,r.qty)+(new.qty/r.qty)*r.percentage*0.01)>dis) then
                                        dcode=r.discount_code;
                                        dis=tem*(new.qty)-
tem*(mod(new.qty,r.qty)+(new.qty/r.qty)*r.percentage*0.01);
                                end if;
                        end LOOP;

                        --raise notice 'Value: %', tem*new.qty-dis;
                                pp=(tem*new.qty-dis)*1.00/new.qty;

                                NEW.purchaseprice = pp;
                                NEW.discount_applied = dcode;
RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';
```

```
create trigger discounts
before insert or update on bill_details
for each row execute procedure discounts();
```

3)

```
create or replace function bills_update()
returns trigger as $$
DECLARE
          pp bill.bill_amount%type;
          total bill.bill_amount%type;

BEGIN
                     IF(TG_OP='INSERT') THEN
                              select bill_amount into total from bill where invno=new.invno;
                              pp=new.purchaseprice*new.qty;
                                        if(total is null) then
                                                  total:=0;
                                        end if;
                              total:=total+pp;
                              update bill set bill_amount=total where invno=new.invno;
                              RETURN NEW;
                    ELSIF(TG_OP='DELETE') THEN
                              select bill_amount into total from bill where invno=old.invno;
                              pp=old.purchaseprice*old.qty;
                              total:=total-pp;
                              update bill set bill_amount=total where invno=old.invno;
                              RETURN OLD;
                    ELSE
                              select bill_amount into total from bill where invno=old.invno;
                              pp=old.purchaseprice*old.qty;
                              total:=total-pp;
```

```
                update bill set bill_amount=total where invno=old.invno;
                select bill_amount into total from bill where invno=new.invno;
                pp=new.purchaseprice*new.qty;
                total:=total+pp;
                update bill set bill_amount=total where invno=new.invno;
                RETURN NEW;
        END IF;

RETURN NULL;
END;
$$ LANGUAGE 'plpgsql';

create trigger bills_update
after insert or update or delete on bill_details
for each row execute procedure bills_update();
```

4)

```
create or replace function stock_to_bill()
returns trigger as $$
DECLARE
    var bill_details.qty%type;
    s items.stock%type;
BEGIN
    IF(TG_OP='INSERT') THEN
                var:=new.qty;
                select stock into s from items where itemcode=new.itemcode;
                update items set stock=s-var where itemcode=new.itemcode;
    ELSIF(TG_OP='UPDATE') THEN
        var:=old.qty;
        select stock into s from items where itemcode=old.itemcode;
        update items set stock=s+var where itemcode=new.itemcode;
```

```
                    var:=new.qty;
                    select stock into s from items where itemcode=new.itemcode;
                    update items set stock=s-var where itemcode=new.itemcode;
          END IF;
RETURN NULL;
END;
$$ LANGUAGE 'plpgsql';

create trigger stock_to_bill
after insert or update on bill_details
for each row execute procedure stock_to_bill();
```