```sql
create table t_produit
(
    idProduit integer not null primary key,   /* identifiant du produit */
    libelle varchar(20),                      /* libelle du produit */
    stock integer,                            /* quantité en stock du produit */
    stockMax integer                          /* valeur maximum que peu avoir le stock pour ce produit */
);

commit;

create table t_achat
(
    idProduit integer not null,             /* identifiant du produit */
    dateAchat date default current_date,    /* date d'achat au fournisseur */
    quantite integer,                       /* quantité achetée du produit à une certaine date */
    prixAchat decimal(7,2),                 /* prix d'achat */
    primary key(idProduit, dateAchat),
    constraint fk_produit
        foreign key(idProduit) references t_produit(idProduit)
);

commit;

create table t_vente
(
    idProduit integer not null,             /* identifiant du produit */
    dateVente date default current_date,    /* date de vente aux clients */
    quantite integer,                       /* quantité vendue du produit à une certaine date */
    prixVente decimal(7,2),                 /* prix de vente du produit */
    primary key(idProduit, dateVente),
    constraint fk_produit_vente
        foreign key(idProduit) references t_produit(idProduit)
);

commit;
```

Question 1 /6

Faire en sorte que l'identifiant du produit (*idProduit*) soit auto-incrémenté ensuite insérer un produit dans la table

```
create generator idProduitGenerator;
commit;
set generator idProduitGenerator to 0;
commit;

set term ^;
create trigger generateIdProduit for t_produit before insert
as begin
    new.idProduit = gen_id(idProduitGenerator, 1);

end^
set term ;^
```

Pour insérer un produit

```
insert into t_produit(libelle, stock, stockMax)
values('Agenda', 0, 30);
```

Question 2 /8

a) Créer une exception suivant :
stockErrone qui affiche le message « Le stock ne peut pas être négatif et ne doit pas dépasse le stock maximum »

```
create exception stockErrone 'Le stock ne peut pas être négatif et ne doit pas dépasse le stock maximum';
commit;
```

b) Faire en sorte que le stock (*stock*) ne soit jamais négatif et ne dépasse pas le stock maximum (*stockMax*)

```
set term ^;
create trigger VerifyStock for t_produit before insert or update
as begin
    if (new.stock < 0 or new.stock > new.stockMax)
    then
        exception stockErrone;
end^
set term ;^
```

Question 3 /7

Lors d'une suppression d'une vente, faire en sorte que le stock du produit correspondant soit mise à jour

```
set term ^;
create trigger deleteVente for t_Vente after delete
as begin
    update t_produit set stock = stock + old.quantite
        where idProduit=old.idProduit;
end^
set term ;^
```

Question 4 /9

a) Créer une procédure qui supprime une vente d'un produit donné (*idproduit*) à une date donnée

```
set term ^;
create procedure supprimeVente
    (idProduit integer, dateVente date)
as begin
    delete from T_vente where idProduit=:idProduit and dateVente=:dateVente;
end^
set term ;^
```

b) exécuter cette procédure

```
execute procedure supprimeVente(5,'3/3/23');
```

Question 5 /10

a) Créer une requête paramétrée qui affiche les produits (*idProduit et libellé*) dont le stock est inférieur à une valeur donnée

```
set term ^;
create procedure afficheProduitStockInf(valeur integer)
returns (idProduit integer, libelle varchar(20))
as begin
    for select idProduit, libelle
        from t_produit
        where stock < :valeur
        into :idProduit, :libelle
    do
        suspend;
end^
set term ;^
```

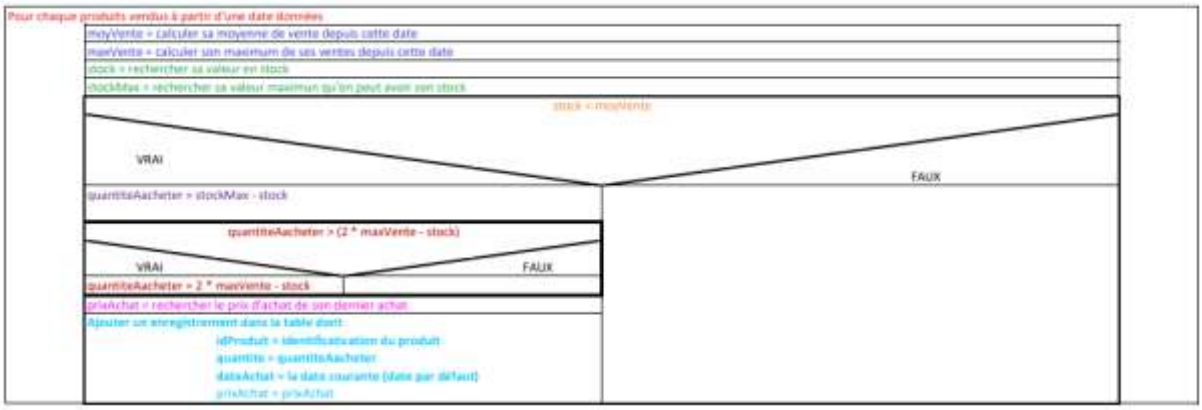
b) Exécuter cette procédure

```
select * from AFFICHEPRODUITSTOCKINF(25);
```

Question 6 /15

Créer une procédure qui ajoute des achats pour chaque produit dont le stock est inférieure à la moyenne de ses ventes qui ont été réalisés après une date donnée. La quantité acheté correspond au minimum de la différence de la quantité de stock maximum et de la quantité en stock (*stockMax - stock*) et de la différence du double de *maxPeriode* et de la quantité en stock (*2 * MaxPeriode - stock*) où *MaxPeriode* est le maximum de vente à partir de la date donnée pour ce produit.

On suppose que le prix d'achat n'a pas changé depuis le dernier achat.




```
SET TERM ^ ;
create PROCEDURE INSEREACHAT (
    DATEDEB Date )
AS
declare variable stock integer;
    declare variable stockMax integer;
    declare variable moyVente integer;
    declare variable maxVente integer;
    declare variable idProduit integer;
    declare variable quantiteAachete integer;
    declare variable prixAchat decimal(10,2);
begin
    for select t_produit.idProduit, stock, stockMax, max(quantite), avg(quantite)
        from t_produit inner join t_vente
                on t_produit.idProduit = t_vente.idProduit
        where (dateVente > :datedeb)
        group by t_produit.idProduit,stock, stockMax
        into :idProduit, :stock, :stockMax, :maxVente, :moyVente
    do
        begin
            if (stock < moyVente)
            then
                begin
                    quantiteAachete = stockMax - stock;
                    if (quantiteAachete > (2 * maxVente - stock))
                    then
                        quantiteAachete = 2 * maxVente - stock;

                    select first 1 prixAchat
                        from t_achat
                        where idProduit = :idProduit
                        order by dateAchat desc
                        into :prixAchat;

                    insert into t_achat (idProduit, quantite, prixAchat)
                        values(:idProduit, :quantiteAachete, :prixAchat);
                end
        end
end^
SET TERM ; ^
```

Ou

```
SET TERM ^ ;
create PROCEDURE INSEREACHAT_TER (
    DATEDEB Date )
AS
declare variable stock integer;
    declare variable stockMax integer;
    declare variable maxVente integer;
    declare variable idProduit integer;
    declare variable quantiteAachete integer;
    declare variable prixAchat decimal(10,2);
begin
    for select t_produit.idProduit, stock, stockMax, max(quantite)
        from t_produit inner join t_vente
                on t_produit.idProduit = t_vente.idProduit
        where (dateVente > :datedeb) and
              (stock < (select avg(quantite) from t_vente where dateVente > :dateDeb))
        group by t_produit.idProduit,stock, stockMax
        into :idProduit, :stock, :stockMax, :maxVente
    do
        begin
            quantiteAachete = stockMax - stock;
            if (quantiteAachete > (2 * maxVente - stock))
            then
                quantiteAachete = 2 * maxVente - stock;

            select first 1 prixAchat
                from t_achat
                where idProduit = :idProduit
                order by dateAchat desc
                into :prixAchat;

            insert into t_achat (idProduit, quantite, prixAchat)
                values(:idProduit, :quantiteAachete, :prixAchat);

        end
end^
SET TERM ; ^
```

ou

```
SET TERM ^ ;
create PROCEDURE insereAchatBis(
    DATEdeb Date )

AS
declare variable stock integer;
    declare variable stockmax integer;
    declare variable moyVente integer;
    declare maxVente integer;
    declare idproduit integer;
    declare prixAchat decimal(7,2);
    declare quantiteAacheter integer;

begin
    for select avg(quantite), max(quantite), IDPRODUIT
        from t_vente
        where dateVente > :datedeb
        group by IDPRODUIT
        into :moyVente, :maxVente, :IDPRODUIT
    do
        begin
            select stock, stockmax
            from t_produit
            where idProduit = :IDPRODUIT
            into :stock, :stockmax;
            if (stock < moyVente)
            then
            begin
                quantiteAacheter = stockmax - stock;
                quantiteAacheter = stockmax - stock;
                if (quantiteAacheter < 2 * maxVente - stock)
                then
                    quantiteAacheter = 2 * maxVente - stock;
                select first 1 prixAchat
                from t_achat
                where idproduit = :IDPRODUIT
                order by idProduit, dateAchat desc
                into :prixAchat;
                insert into t_achat(idproduit, quantite, prixAchat)
                values(:idproduit, :quantiteAacheter, :prixAchat);
            end
        end

    end^
SET TERM ; ^
```