**2) Trigger pour vérifier la quantité en stoch**

Delimiter $$

Create trigger control\_stock

Before insert on orderDetails

For each row

Begin

Declare stock INT ;

Select quantityInStock into stock

from products, orderdetails

where products.productCode = orderdetails.productCode ;

If new.quantityOrdered > stock then

Signal sqlstate ‘45000’

Set message\_text = ‘Le stock est inférieur à la quantité commandée’

End if ;

End $$

**3) Trigger pour enregistrer des payements supprimés**

Delimiter $$

Create trigger DeletedPayments

Before delete on payments

For each row

Begin

If exists DeletedPayments then

Insert into DeletedPayments(customerNumber, checkNumber, paymentDate, amount, dateDeleted)

Values(old. customerNumber, old. checkNumber, old. paymentDate, old. Amount, now());

Else

Create table DeletedPayments(

customerNumber int,

checkNumber varchar(50),

paymentDate date,

amount varchar(100),

primary key(checkNumber),

froreign key(customerNumber) references customers);

Insert into DeletedPayments(customerNumber, checkNumber, paymentDate, amount, dateDeleted)

Values(old. customerNumber, old. checkNumber, old. paymentDate, old. Amount, now());

End if;

End $$

4) Calculer le total d’une commande

Create trigger mise\_a\_jour\_orderTotal

After insert on orderDetails

For each row

Begin

Declare col\_exist int;

SELECT COUNT(\*) into col\_exist FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = 'nom\_table' AND COLUMN\_NAME = 'nom\_colonne' AND TABLE\_SCHEMA = 'nom\_base\_de\_données';

If col\_exist = 0 then

Alter table orderDetails add column orderTotal interger;

Update orders

Set orderTotal = sum(quantityOrdered \* eachPrice)

End if;

End $$

5) Trigger pour verifier si un client est actif avant d’accepter un paiement : Bloquer le paiement si le client est inactif.

Delimiter $$

Create trigger client\_actif

Before insert on payments

For each row

Begin

Declare lastPayment date;

Select paymentDate into lastPayment

From payments

Where checkNumber = new.checkNumber;

If customers.customerNumber < > payments.customerNumber and lastPayment > date\_sub( now(), interval 1 month ) then

Signal sqlstate ‘44000’

Set message\_text = ‘Vous avez été inactif pendant un mois.’ ;

End if ;

End $$

6)

Delimiter $$

Create trigger control\_suppression\_produit

Before delete on products

For each row

Begin

Declare order\_stock int;

Select count(\*) into order\_check

From orderDetails

Where productCode = old.productCode;

If order\_check > 0 then

Signal sqlqstate ‘39000’

Set message\_text = ‘Vous ne pouvez pas supprimer une commande en cours’ ;

End if ;

End $$

7) Trigger pour archiver les clients supprimés

Delimiter $$

Create trigger archive\_clients

Before delete on customers

For each row

Begin

If exists ArchivedCustomers then

Insert into ArchivedCustomers(customerNumber, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country, salesRepEmployeeNumber, creditLimit, deleteDate)

Values(old.customerNumber, old.customerName, old.contactLastName, old.contactFirstName, old.phone, old.addressLine1, old.addressLine2, old.city, old.state, old.postalCode, old.country, old.salesRepEmployeeNumber, old.creditLimit, now());

Else

Create table ArchivedCustomers(

customerNumber integer,

customerName integer,

contactLastName varchar(50),

contactFirstName varchar(50),

phone varchar(25),

addressLine1 varchar(100),

addressLine2 varchar(100),

city varchar(25),

state varchar(25),

postalCode varchar(15),

country varchar(25),

salesRepEmployeeNumber integer,

creditLimit integer,

deleteDate date);

Insert into ArchivedCustomers(customerNumber, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country, salesRepEmployeeNumber, creditLimit, deleteDate)

Values(old.customerNumber, old.customerName, old.contactLastName, old.contactFirstName, old.phone, old.addressLine1, old.addressLine2, old.city, old.state, old.postalCode, old.country, old.salesRepEmployeeNumber, old.creditLimit, now());

End if;

End $$

8) Trigger pour restreindre les augmentations de salaire supérieures à 10%.

Delimiter $$

Create trigger restriction\_salaire

Before insert on employees

For each row

Begin

Declare augmentation\_salaire decimal(10, 2)

SELECT COUNT(\*) into augmentation\_salaire

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = ‘employees’

AND COLUMN\_NAME = ‘augmentation\_salaire’

AND TABLE\_SCHEMA = ‘Boutique’;

If augmentation\_salaire = 0 then

Alter table employees add column augmentation\_salaire decimal(10, 2);

End if;

If new.augmentation\_salaire > 0.01 then

Signal sqlstate ‘41000’

Set message\_text = ‘L’augmentation du salaire ne peut dépasser 10%’ ;

End $$

9) Trigger pour enregistrer les changements de statut de commande

Delimiter $$

Create trigger changement\_statut

After insert on orders

For each row

Begin

If exists OrderStatusHistory then

Insert into OrderStatusHistory(orderNumber, customerNumber, orderDate, status)

Values(old.orderNumber, old.customerNumber, old.orderDate, old.status);

Else

Create table OrderStatusHistory(

orderNumber integer,

customerNumber integer,

orderDate date,

status varchar(100),

primary key(orderNumber),

foreign key(customerNumber) references customers);

Insert into OrderStatusHistory(orderNumber, customerNumber, orderDate, status)

Values(old.orderNumber, old.customerNumber, old.orderDate, old.status)

End if;

End $$