

UNIVERSITE ABDELMALEK ESSAADI

Master IA et science de données

Sujet : Projet de fin de module

Projet :

Rapport de TP de Base de Données (SQL Server)

Réalisé PAR :

KHATTABI IDRIS
BOUFARHI AYMAN

ENCADRE PAR :

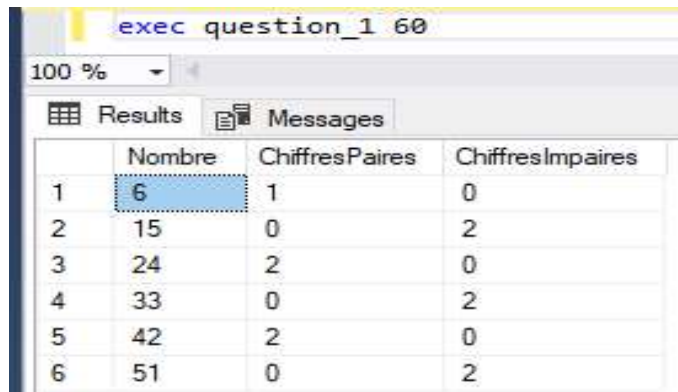
Prof. EZZIYYANI MOSTAFA

Année Universitaire : 2023/2024

1)

```
create PROCEDURE question_1
@nombre_max INT
AS
BEGIN
    -- Création de la table temporaire pour stocker les résultats
    CREATE TABLE #TMP (
        Nombre INT,
        ChiffresPaires INT,
        ChiffresImpaires INT
    );
    DECLARE @i INT = 1;
    DECLARE @somme INT;
    DECLARE @chiffre INT;
    DECLARE @nbPaires INT;
    DECLARE @nbImpaires INT;
    -- Boucle pour parcourir les nombres inférieurs à nombre_max
    WHILE @i < @nombre_max
    BEGIN
        SET @somme = 0;
        SET @chiffre = @i;
        SET @nbPaires = 0;
        SET @nbImpaires = 0;
        -- Calcul de la somme des chiffres du nombre @i
        WHILE @chiffre > 0
        BEGIN
            SET @somme = @somme + (@chiffre % 10);
            IF (@chiffre % 10) % 2 = 0
                SET @nbPaires = @nbPaires + 1;
            ELSE
                SET @nbImpaires = @nbImpaires + 1;
            SET @chiffre = FLOOR(@chiffre / 10);
        END;
        -- Vérification si la somme est égale à 6
        IF @somme = 6
        BEGIN
            -- Insertion du nombre et des informations dans la table temporaire
            INSERT INTO #TMP (Nombre, ChiffresPaires, ChiffresImpaires) VALUES (@i,
@nbPaires, @nbImpaires);
        END;
        SET @i = @i + 1;
    END;
    -- Sélection des résultats de la table temporaire
    SELECT * FROM #TMP;
    -- Suppression de la table temporaire
    DROP TABLE #TMP;
END;
```

⇒ Resultat :



	Nombre	ChiffresPaires	ChiffresImpaires
1	6	1	0
2	15	0	2
3	24	2	0
4	33	0	2
5	42	2	0
6	51	0	2

2)

```

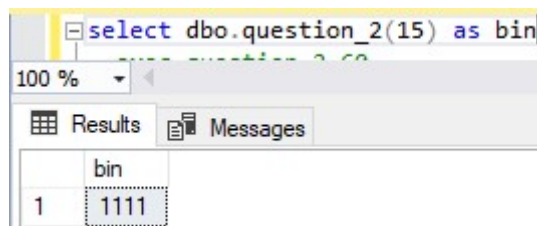
create FUNCTION question_2
(@entier INT)
RETURNS VARCHAR(MAX)
AS
BEGIN
    DECLARE @code_binaire VARCHAR(MAX) = '';
    DECLARE @quotient INT = @entier;
    DECLARE @reste INT;

    -- Cas particulier si l'entier est zéro
    IF @entier = 0
    BEGIN
        SET @code_binaire = '0';
    END
    ELSE
    BEGIN
        -- Conversion en code binaire
        WHILE @quotient > 0
        BEGIN
            SET @reste = @quotient % 2;
            SET @quotient = @quotient / 2;
            SET @code_binaire = CONVERT(CHAR(1), @reste) + @code_binaire;
        END;
    END;

    RETURN @code_binaire;
END;

```

⇒ Resultat :



	bin
1	1111

3)

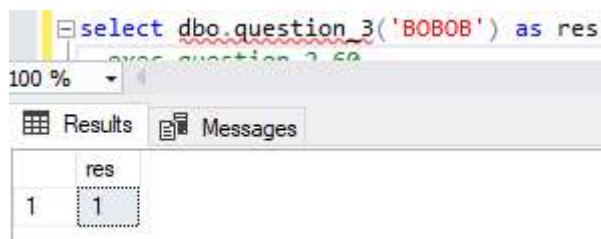
```

create FUNCTION question_3 (@str1 VARCHAR(MAX))
RETURNS INT
AS
BEGIN
    DECLARE @len INT = LEN(@str1);
    DECLARE @moitie INT = @len / 2;
    DECLARE @i INT = 1;
    while @i <= @moitie
    begin
        IF SUBSTRING(@str1, @i, 1) != SUBSTRING(@str1, @len - @i + 1, 1)
            RETURN 0; -- Non palindrome

        set @i = @i + 1
    end
    return 1 -- palindrome
END;

```

⇒ Resultat :



	res
1	1

4)

```

create FUNCTION question_4 (@str VARCHAR(MAX))
RETURNS INT

```

```

AS
BEGIN
    DECLARE @nombre_mots INT = 0;
    DECLARE @i INT = 1;
    DECLARE @longueur INT = LEN(@str);
    DECLARE @caractere_precedent CHAR(1) = ' ';
    DECLARE @caractere_actuel CHAR(1);

    WHILE @i <= @longueur
    BEGIN
        SET @caractere_actuel = SUBSTRING(@str, @i, 1);

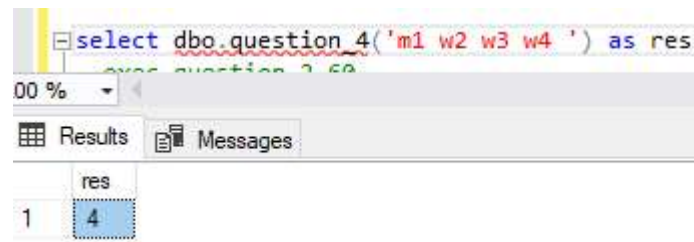
        IF @caractere_actuel LIKE '[A-Za-z]' AND @caractere_precedent = ' '
        BEGIN
            SET @nombre_mots = @nombre_mots + 1;
        END;

        SET @caractere_precedent = @caractere_actuel;
        SET @i = @i + 1;
    END;

    RETURN @nombre_mots;
END;

```

⇒ Resultat :



	res
1	4

5)

```

create FUNCTION question_5 (@str1 VARCHAR(MAX), @sub_str VARCHAR(MAX))
RETURNS INT
AS
BEGIN
    DECLARE @len_str1 int = LEN(@str1)
    declare @i int = 0
    declare @j int = 0
    declare @nbr_of_occurences int = 0

    while @i < @len_str1
    begin
        -- CHARINDEX(substring, string, start) => return the position
        set @j = CHARINDEX(@sub_str, @str1, @i + 1)

        if @j != 0
        begin
            set @i = @j
            set @nbr_of_occurences = @nbr_of_occurences + 1
        end
        else
        begin
            break
        end
    end
    return @nbr_of_occurences
END;

```

⇒ Resultat :

	res
1	3

6)

```
CREATE FUNCTION question_6 ( @chaine VARCHAR(MAX))
RETURNS VARCHAR(MAX)
AS
BEGIN
    DECLARE @longueur_max INT = 0;
    DECLARE @mot_max VARCHAR(MAX) = '';
    DECLARE @mot_temp VARCHAR(MAX) = '';
    DECLARE @caractere CHAR(1);
    DECLARE @i INT = 1;

    WHILE @i <= LEN(@chaine)
    BEGIN
        SET @caractere = SUBSTRING(@chaine, @i, 1);

        -- Si le caractère est une lettre ou un chiffre, on ajoute au mot temporaire
        IF @caractere LIKE '[a-zA-Z0-9]'
        BEGIN
            SET @mot_temp = @mot_temp + @caractere;
        END
        ELSE
        BEGIN
            -- Si le mot temporaire est plus long que le mot max actuel, on le met à
            jour
            IF LEN(@mot_temp) > @longueur_max
            BEGIN
                SET @longueur_max = LEN(@mot_temp);
                SET @mot_max = @mot_temp;
            END

            -- Réinitialisation du mot temporaire pour commencer un nouveau mot
            SET @mot_temp = '';
        END

        SET @i = @i + 1;
    END

    -- Gestion du cas où le mot le plus long est à la fin de la chaîne
    IF LEN(@mot_temp) > @longueur_max
    BEGIN
        SET @mot_max = @mot_temp;
    END

    RETURN @mot_max;
END;
```

⇒ Resultat :

	res
1	w4rtybf

7)

```
create FUNCTION question_7 ( @minute int)
RETURNS VARCHAR(MAX)
```

```

AS
BEGIN
    DECLARE @years INT = 0;
    DECLARE @months INT = 0;
    DECLARE @days INT = 0;
    DECLARE @hours INT = 0;
    DECLARE @minutes INT = 0;

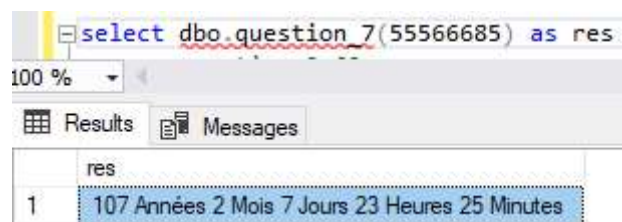
    set @minutes = @minute % 60
        set @hours = (@minute / 60) % 24
    set @days = (((@minute / 60) / 24) % 30)
    set @months = (((@minute / 60) / 24) / 30) % 12
    set @years = (((@minute / 60) / 24) / 30) / 12

    DECLARE @result VARCHAR(MAX) = CONCAT(@years, ' Années ', @months, ' Mois ',
@days, ' Jours ', @hours, ' Heures ', @minutes, ' Minutes');

    RETURN @result
END;

```

⇒ **Resultat :**



	res
1	107 Années 2 Mois 7 Jours 23 Heures 25 Minutes

8)

```

create procedure question_8
AS
BEGIN
    CREATE TABLE Vols (
        Num_Vol INT PRIMARY KEY,
        Date_Depart DATE,
        Heure_Depart TIME,
        Ville_Depart VARCHAR(100),
        Ville_Arrivee VARCHAR(100),
        Code_Avion INT,
        Code_Pilote INT,
        Prix_Vol DECIMAL(10, 2),
        FOREIGN KEY (Code_Avion) REFERENCES dbo.Avions(Num_Avion),
        FOREIGN KEY (Code_Pilote) REFERENCES dbo.Pilotes(Num_Pilote)
    );
END;

```

⇒ **Resultat :**

	Column Name	Data Type	Allow Nulls
▶	Num_Vol	int	<input type="checkbox"/>
	Date_Depart	date	<input checked="" type="checkbox"/>
	Heure_Depart	time(7)	<input checked="" type="checkbox"/>
	Ville_Depart	nchar(20)	<input checked="" type="checkbox"/>
	Ville_Arrivee	nchar(20)	<input checked="" type="checkbox"/>
	Code_Avion	int	<input checked="" type="checkbox"/>
	Code_Pilote	int	<input checked="" type="checkbox"/>
	Prix_Vol	float	<input checked="" type="checkbox"/>

9)

```

CREATE PROCEDURE question_9
    @date_donnee DATE
AS
BEGIN

```

```

SELECT *
FROM dbo.Reservations
WHERE Date_Validation IS NULL
AND CONVERT(DATE, Date_Reservation) = @date_donnee;
END;

```

⇒ Resultat :

exec question_9 '2024-02-10'

100 %

Results Messages

	Num_Reservation	Date_Reservation	Date_Validation	Etat_Reservation	Code_Agence	Code_Passager	Prix_Total
1	4	2024-02-10	NULL	No Valide	6465	1	555

10)

```

CREATE PROCEDURE question_10
    @num_vol int
AS
BEGIN
    SELECT *
    FROM dbo.Vols
    WHERE Num_Vol = @num_vol;
END;

```

⇒ Resultat :

exec question_10 1

100 %

Results Messages

	Num_Vol	Date_Depart	Heure_Depart	Ville_Depart	Ville_Arree	Code_Avion	Code_Pilote	Prix_Vol
1	1	2024-02-09	09:00:00.0000000	larache	tanger	1	2	55

11)

```

create PROCEDURE question_11
    @num_vol int
AS
BEGIN
    SELECT v1.Num_Vol, v1.Ville_Depart, v1.Date_Depart, v1.Heure_Depart, v1.Ville_Arree,
    v1.Prix_Vol, pl.Nom_Pilote+' '+pl.Prenom_Pilote as "pilote", av.Nom_Companie as
    "Avion_Companie"
    FROM dbo.Vols as v1
    INNER JOIN dbo.Avions AS av ON v1.Code_Avion = av.Num_Avion
    INNER JOIN dbo.Pilotes AS pl ON v1.Code_Pilote = pl.Num_Pilote
    WHERE v1.Num_Vol = @num_vol;
END;

```

⇒ Resultat :

exec question_11 1

100 %

Results Messages

	Num_Vol	Ville_Depart	Date_Depart	Heure_Depart	Ville_Arree	Prix_Vol	pilote	Avion_Companie
1	1	larache	2024-02-09	09:00:00.0000000	tanger	55	idr2 kh2	azerty

12)

```

create PROCEDURE question_12
    @num_reservation INT
AS
BEGIN
    SELECT

```

```

        r.Num_Reservation, r.Date_Reservation, r.Date_Validation, r.Etat_Reservation,
        r.Code_Agence, p.Code_Passager, p.Nom_Passager, p.Pre_Passager, p.Num_Passport,
        p.Categorie, p.Num_Tel, b.Num_Billet
    FROM Reservations r
    INNER JOIN Passagers p ON r.Code_Passager = p.Code_Passager
    INNER JOIN Billets b ON b.Num_Reservation = r.Num_Reservation
    WHERE r.Num_Reservation = @num_reservation AND r.Etat_Reservation = 'Valide';
END;

```

⇒ **Resultat :**

exec question_12 2

100 %

Results Messages

	Num_Reservation	Date_Reservation	Date_Validation	Etat_Reservation	Code_Agence	Code_Passager	Nom_Passager	Pre_Passager	Num_Passport	Categorie	Num_Tel	Num_Billet
1	2	2024-05-03	2024-08-02	Valide	684	2	idr2	kh2	6656	B	2565	3

13)

```

create procedure question_13
as
begin
    select Code_Avion, COUNT(Code_Avion) as counts
    from Vols
    group by Code_Avion
    order by counts DESC
end

```

⇒ **Resultat :**

exec question_13

100 %

Results Messages

	Code_Avion	counts
1	2	3
2	1	2
3	3	1

14)

```

create procedure question_14
    @Passager_Id int
as
begin
    select COUNT(*) as nbrs_of_vols
    from Voyages
    where Code_Passager = @Passager_Id
end

```

⇒ **Resultat :**

exec question_14 1

100 %

Results Messages

	nbrs_of_vols
1	3

15)

```

create function question_15 (@Vol_Id int)
returns float
as
begin

```

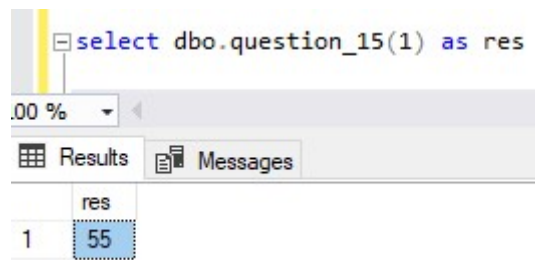


```

declare @Prix_Vol float
select @Prix_Vol = Prix_Vol
from Vols
where Num_Vol = @Vol_Id
return @Prix_Vol
end

```

⇒ **Resultat :**



res
55

16)

```

CREATE PROCEDURE question_16
AS
BEGIN
    DELETE FROM Reservations
    WHERE Etat_Reservation <> 'Valide';
END;

```

⇒ **Resultat :**

17)

```

CREATE PROCEDURE question_17
    @Code_Passager INT,
    @Num_Billet INT,
    @Num_Vol INT,
    @Num_Place INT
AS
BEGIN
    IF NOT EXISTS (SELECT * FROM Voyages WHERE Code_Passager = @Code_Passager AND
Num_Billet = @Num_Billet AND Num_Vol = @Num_Vol)
    BEGIN
        IF EXISTS (SELECT * FROM Billets WHERE Num_Billet = @Num_Billet) --- il faut
fixé cet condition
        BEGIN
            IF NOT EXISTS (SELECT * FROM Voyages WHERE Num_Place = @Num_Place)
            BEGIN
                INSERT INTO Voyages (Code_Passager, Num_Billet, Num_Vol, Num_Place)
                VALUES (@Code_Passager, @Num_Billet, @Num_Vol, @Num_Place);
                PRINT 'L'enregistrement a été inséré avec succès dans la table
Voyages.';
            END
        ELSE
        BEGIN
            PRINT 'Erreur : Le numéro de place est déjà utilisé pour un autre
passager.';
        END
    END
    ELSE
    BEGIN
        PRINT 'Erreur : Le numéro du billet ne correspond pas au passager et au
vol.';
    END
    ELSE
    BEGIN
        PRINT 'Erreur : Un enregistrement avec les mêmes valeurs existe déjà dans la
table Voyages.';
    END
END

```

END;

⇒ Resultat :

The screenshot shows a SQL Server interface. At the top, a query window displays the command `exec question_17 2, 2, 3, 66`. Below the command bar, a status bar indicates '100 %'. A 'Messages' pane shows the message: 'L'enregistrement a été inséré avec succès dans la table Voyages.' Below this, a table window titled 'id-kh\IDRISS.bd_devoir - dbo.Voyages' displays the following data:

	Code_Passager	Num_Billet	Num_Vol	Num_place
▶	1	1	1	22
	1	2	2	33
	2	3	1	15
	3	4	3	9
	2	5	2	17
	1	6	4	103
	2	2	3	66

An orange arrow points to the last row of the table, where the values are 2, 2, 3, and 66.

18)

```
CREATE PROCEDURE question_18
    @Num_Ligne INT OUTPUT,
    @Num_Order INT,
    @Num_Vol INT,
    @Num_Reservation INT
AS
BEGIN
    DECLARE @Previous_Ville_Arrivee VARCHAR(100);

    -- Vérifier si la ville de départ du vol de la nouvelle réservation coïncide avec
    la ville d'arrivée du vol de la réservation précédente
    IF @Num_Order > 1
    BEGIN
        SELECT @Previous_Ville_Arrivee = Ville_Arrivee
        FROM Vols
        WHERE Num_Vol = (SELECT Num_Vol FROM Ligne_Reservation WHERE Num_Order =
@Num_Order - 1 AND Num_Reservation = @Num_Reservation);

        IF NOT EXISTS (SELECT * FROM Vols WHERE Num_Vol = @Num_Vol AND Ville_Depart =
@Previous_Ville_Arrivee)
        BEGIN
            PRINT 'Erreur : La ville de départ du vol ne coïncide pas avec la ville
d'arrivée du vol précédent.';
            RETURN;
        END
    END

    -- Vérifier s'il y a encore une place dans l'avion
    DECLARE @Nbr_Places_Occupees INT;
    DECLARE @Nbr_Places INT;

    SELECT @Nbr_Places_Occupees = COUNT(*) FROM Ligne_Reservation WHERE Num_Vol =
@Num_Vol;
    SELECT @Nbr_Places = Nbr_Place FROM Avions INNER JOIN Vols ON Avions.Num_Avion =
Vols.Code_Avion WHERE Vols.Num_Vol = @Num_Vol;

    IF @Nbr_Places_Occupees >= @Nbr_Places
    BEGIN
        PRINT 'Erreur : Plus de place disponible dans l'avion pour ce vol.';
        RETURN;
    END
END
```

```

-- Insérer l'enregistrement dans la table Ligne_Reservation
INSERT INTO Ligne_Reservation (Num_Order, Num_Vol, Num_Reservation)
VALUES (@Num_Order, @Num_Vol, @Num_Reservation);

SET @Num_Ligne = SCOPE_IDENTITY(); -- Obtenir la clé primaire de l'enregistrement
nouvellement inséré

PRINT 'L'enregistrement a été inséré avec succès dans la table
Ligne_Reservation.';
END;

```

⇒ **Resultat :**

19)

```

CREATE PROCEDURE question_19
AS
BEGIN
    DECLARE @sql NVARCHAR(MAX);

    -- Vérifier d'abord si les colonnes existent déjà dans la table
    IF NOT EXISTS (SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'Vols'
AND COLUMN_NAME = 'Nbr_Res')
    BEGIN
        SET @sql = 'ALTER TABLE Vols ADD Nbr_Res INT DEFAULT 0;';
        EXECUTE sp_executesql @sql;
        PRINT 'La colonne Nbr_Res a été ajoutée avec succès à la table Vols.';
    END

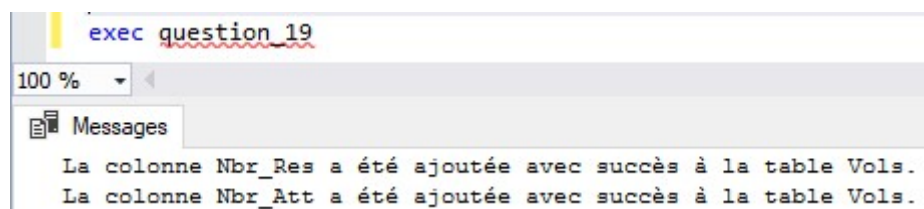
    IF NOT EXISTS (SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'Vols'
AND COLUMN_NAME = 'Nbr_Att')
    BEGIN
        SET @sql = 'ALTER TABLE Vols ADD Nbr_Att INT DEFAULT 0;';
        EXECUTE sp_executesql @sql;
        PRINT 'La colonne Nbr_Att a été ajoutée avec succès à la table Vols.';
    END
END;

```

⇒ **Resultat :**

Avant :

Num_Vol	Date_Départ	Heure_Départ	Ville_Départ	Ville_Arrivée	Code_Avion	Code_Pilote	Prix_Vol
1	2024-02-09	09:00:00	larache	tanger	1	2	55
2	2024-02-10	22:00:00	ouazzane	rabat	2	1	60.66
3	2024-02-09	23:00:00	azt1	aer2	1	NULL	99
4	2024-02-14	12:00:00	aaa1	aaa2	3	1	55
5	2024-02-13	13:00:00	bbb1	bbb2	2	3	66
6	2024-02-09	NULL	rrr1	rrr2	2	2	66
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



Après :

Num_Vol	Date_Départ	Heure_Départ	Ville_Départ	Ville_Arrivée	Code_Avion	Code_Pilote	Prix_Vol	Nbr_Res	Nbr_Att
1	2024-02-09	09:00:00	larache	tanger	1	2	55	NULL	NULL
2	2024-02-10	22:00:00	ouazzane	rabat	2	1	60.66	NULL	NULL
3	2024-02-09	23:00:00	azt1	aer2	1	NULL	99	NULL	NULL
4	2024-02-14	12:00:00	aaa1	aaa2	3	1	55	NULL	NULL
5	2024-02-13	13:00:00	bbb1	bbb2	2	3	66	NULL	NULL
6	2024-02-09	NULL	rrr1	rrr2	2	2	66	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

20)

CREATE PROCEDURE question_20

@Num_Vol INT

AS

BEGIN

DECLARE @Nbr_Places_Res INT;

DECLARE @Nbr_Places_Att INT;

-- Calculer le nombre de places réservées pour le vol donné

SELECT @Nbr_Places_Res = COUNT(Num_Ligne)

FROM Ligne_Reservation

WHERE Num_Vol = @Num_Vol;

-- Calculer le nombre de places attribuées pour le vol donné

SELECT @Nbr_Places_Att = COUNT(Num_Place)

FROM Voyages

WHERE Num_Vol = @Num_Vol;

-- Mettre à jour les colonnes Nbr_Res et Nbr_Att dans la table Vols

UPDATE Vols

SET Nbr_Res = @Nbr_Places_Res,

Nbr_Att = @Nbr_Places_Att

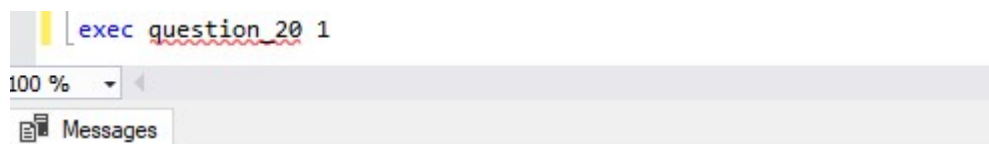
WHERE Num_Vol = @Num_Vol;

PRINT 'Les colonnes Nbr_Res et Nbr_Att ont été mises à jour pour le vol ' +

CAST(@Num_Vol AS VARCHAR(10)) + '.';

END;

⇒ Resultat :



Num_Vol	Date_Départ	Heure_Départ	Ville_Départ	Ville_Arrivée	Code_Avion	Code_Pilote	Prix_Vol	Nbr_Res	Nbr_Att
1	2024-02-09	09:00:00	larache	tanger	1	2	55	0	2
2	2024-02-10	22:00:00	ouazzane	rabat	2	1	60.66	NULL	NULL

21)

CREATE PROCEDURE question_21

@Code_Passager INT

AS

BEGIN

DECLARE @Nbr_Voyages INT;

DECLARE @Montant_Paiement DECIMAL(10, 2);

DECLARE @Categorie VARCHAR(50);

-- Calculer le nombre de voyages effectués par le passager durant l'année en cours

SELECT @Nbr_Voyages = COUNT(*)

FROM Voyages

INNER JOIN Vols ON Voyages.Num_Vol = Vols.Num_Vol

WHERE Voyages.Code_Passager = @Code_Passager

```

AND YEAR(Vols.Date_Depart) = YEAR(GETDATE());

-- Calculer le montant total de paiement du passager durant l'année en cours
SELECT @Montant_Paiement = SUM(Prix_Vol)
FROM Voyages
INNER JOIN Vols ON Voyages.Num_Vol = Vols.Num_Vol
WHERE Voyages.Code_Passager = @Code_Passager
AND YEAR(Vols.Date_Depart) = YEAR(GETDATE());

-- Déterminer la catégorie du passager
IF @Nbr_Voyages > 20 AND @Montant_Paiement > 200000
BEGIN
    SET @Categorie = 'Très Actif';
END
ELSE IF @Nbr_Voyages > 20
BEGIN
    SET @Categorie = 'Actif';
END
ELSE
BEGIN
    SET @Categorie = 'Moyen';
END

-- Mettre à jour le champ Categorie du passager dans la table Passagers
UPDATE Passagers
SET Categorie = @Categorie
WHERE Code_Passager = @Code_Passager;

PRINT 'La catégorie du passager ' + CAST(@Code_Passager AS VARCHAR(10)) + ' a été
mise à jour avec succès.';
END;

```

⇒ **Resultat :**

exec question_21 2

100 %

Messages

(1 row affected)
La catégorie du passager 2 a été mise à jour avec succès.

	Code_Passager	Nom_Passager	Pre_Passager	Num_Passport	Categorie	Num_Tel
► 1	idriss	kh	68465	Moyen	123	
2	idr2	kh2	6656	Moyen	2565	

22)

```

CREATE PROCEDURE question_22
AS
BEGIN
    SELECT p.Code_Passager, p.Nom_Passager, p.Pre_Passager, COUNT(v.Num_Vol) AS
Nombre_Voyages
FROM Passagers p
LEFT JOIN Voyages v ON p.Code_Passager = v.Code_Passager
GROUP BY p.Code_Passager, p.Nom_Passager, p.Pre_Passager;
END;

```

⇒ **Resultat :**

exec question_22

100 %

Results Messages

	Code_Passager	Nom_Passager	Pre_Passager	Nombre_Voyages
1	1	idriiss	kh	3
2	2	idr2	kh2	4
3	3	moha1	lll3	1
4	4	mr1	mr2	0

23)

CREATE PROCEDURE question_23

AS

BEGIN

-- Afficher le coût de revient pour chaque vol

SELECT Num_Vol,
Date_Depart,
Heure_Depart,
Ville_Depart,
Ville_Arrivee,
ISNULL(SUM(Prix_Vol), 0) AS Cout_Revient

FROM Vols V

GROUP BY Num_Vol, Date_Depart, Heure_Depart, Ville_Depart, Ville_Arrivee;

END;

⇒ Resultat :

exec question_23

100 %

Results Messages

	Num_Vol	Date_Depart	Heure_Depart	Ville_Depart	Ville_Arrivee	Cout_Revient
1	1	2024-02-09	09:00:00.0000000	larache	tanger	55
2	2	2024-02-10	22:00:00.0000000	ouazzane	rabat	60.66
3	3	2024-02-09	23:00:00.0000000	azt1	aer2	99
4	4	2024-02-14	12:00:00.0000000	aaa1	aaa2	55
5	5	2024-02-13	13:00:00.0000000	bbb1	bbb2	66
6	6	2024-02-09	NULL	mr1	mr2	66

24)

25)

CREATE PROCEDURE question_25

@pourcentage DECIMAL(5, 2)

AS

BEGIN

SELECT p.Num_Pilote, p.Nom_Pilote, p.Prenom_Pilote, COUNT(DISTINCT v.Code_Avion)

AS Nombre_Avions

FROM Pilotes p

INNER JOIN Vols v ON p.Num_Pilote = v.Code_Pilote

GROUP BY p.Num_Pilote, p.Nom_Pilote, p.Prenom_Pilote

HAVING COUNT(DISTINCT v.Code_Avion) > (SELECT COUNT(*) * @pourcentage / 100 FROM

Avions);

END;

⇒ Resultat :

exec question_25 20

100 %

Results Messages

	Num_Pilote	Nom_Pilote	Prenom_Pilote	Nombre_Avions
1	1	idr1	kh1	2
2	2	idr2	kh2	2

26)

CREATE PROCEDURE question_26

AS

BEGIN

-- Vérifier si les colonnes existent déjà dans la table Pilotes

IF NOT EXISTS (

SELECT *

FROM INFORMATION_SCHEMA.COLUMNS

WHERE TABLE_NAME = 'Pilotes'

AND COLUMN_NAME IN ('NbrAvions', 'NbrVoyages', 'Statut')

)

BEGIN

-- Ajouter les colonnes à la table Pilotes

ALTER TABLE Pilotes

ADD NbrAvions INT,

NbrVoyages INT,

Statut VARCHAR(50);

-- Initialiser les colonnes NbrAvions et NbrVoyages à zéro

UPDATE Pilotes

SET NbrAvions = 0,

NbrVoyages = 0;

end;

-- Calculer le nombre total d'avions et de voyages dans la compagnie

DECLARE @TotalAvions INT, @TotalVoyages INT;

SELECT @TotalAvions = COUNT(*)

FROM Avions;

SELECT @TotalVoyages = COUNT(*)

FROM Voyages;

-- Mettre à jour les colonnes NbrAvions et NbrVoyages pour chaque pilote

UPDATE Pilotes

SET NbrAvions = (

SELECT COUNT(DISTINCT v.Code_Avion) FROM Vols v

WHERE v.Code_Pilote = Pilotes.Num_Pilote);

UPDATE Pilotes

SET NbrVoyages = (

SELECT COUNT(*) FROM Voyages v

INNER JOIN Vols vo ON v.Num_Vol = vo.Num_Vol

WHERE vo.Code_Pilote = Pilotes.Num_Pilote);

-- Initialiser la colonne Statut selon les critères spécifiés

UPDATE Pilotes

SET Statut = CASE

WHEN NbrAvions > 0 AND (NbrAvions * 100.0 / @TotalAvions) > 50 THEN 'Expert'

WHEN NbrAvions > 0 AND (NbrAvions * 100.0 / @TotalAvions) >= 5 AND

(NbrAvions * 100.0 / @TotalAvions) <= 50 THEN 'Qualifie'

ELSE 'Débiteur'

END;

END;

⇒ Resultat :

	Num_Pilote	Nom_Pilote	Prenom_Pilote	NbrAvions	NbrVoyages	Statut
▶	1	idr1	kh1	2	3	Qualifie
	2	idr2	kh2	2	2	Qualifie
	3	idr3	kh3	1	0	Qualifie
*	NULL	NULL	NULL	NULL	NULL	NULL

27)

```
CREATE PROCEDURE question_27(
    @ville_depart VARCHAR(100),
    @ville_arrivee VARCHAR(100),
    @nombre_escales INT)
AS
BEGIN
    SELECT DISTINCT b.Num_Billet, b.Num_Reservation, v.Num_Vol, v.Date_Depart,
        v.Heure_Depart, v.Ville_Depart, v.Ville_Arrivee, v.Prix_Vol
    FROM Billets b
    INNER JOIN Ligne_Reservation lr ON b.Num_Reservation = lr.Num_Reservation
    INNER JOIN Vols v ON lr.Num_Vol = v.Num_Vol
    WHERE v.Ville_Depart = @ville_depart
    AND v.Ville_Arrivee = @ville_arrivee
    AND (SELECT COUNT(*) FROM Ligne_Reservation WHERE Num_Reservation =
lr.Num_Reservation) = @nombre_escales
    ORDER BY v.Prix_Vol DESC;
END;
```

⇒ Resultat :

28)

Le code de la fct « Complet » :

```
CREATE FUNCTION dbo.Complet(@Num_Vol INT)
RETURNS BIT
AS
BEGIN
    DECLARE @TotalSeats INT;
    DECLARE @BookedSeats INT;

    -- Calculate the total number of seats on the flight
    SELECT @TotalSeats = a.Nbr_Place
    FROM Vols v
    INNER JOIN Avions a ON v.Code_Avion = a.Num_Avion
    WHERE v.Num_Vol = @Num_Vol;

    -- Calculate the number of seats already booked
    SELECT @BookedSeats = COUNT(*)
    FROM Voyages
    WHERE Num_Vol = @Num_Vol;

    -- Check if the flight is full
    IF @BookedSeats >= @TotalSeats
        RETURN 1; -- Flight is full

    RETURN 0; -- Flight is not full
END;
```

Le code de la fct « Complet » :

```
CREATE FUNCTION dbo.Occuper(@Num_Vol INT, @Num_Place VARCHAR(10))
RETURNS BIT
AS
BEGIN
    DECLARE @Occupied BIT;

    -- Check if the seat is occupied on the specified flight
```



```

SELECT @Occupied = CASE WHEN EXISTS (
    SELECT 1
    FROM Voyages
    WHERE Num_Vol = @Num_Vol
    AND Num_Place = @Num_Place
)
    THEN 1
    ELSE 0
END;

RETURN @Occupied;
END;

```

Le code de la TRIGGER **question_28** :

```

CREATE TRIGGER question_28
ON Voyages
AFTER INSERT
AS
BEGIN
    DECLARE @Num_Vol INT, @Code_Passager INT, @Num_Place INT;

    SELECT @Num_Vol = Num_Vol, @Code_Passager = Code_Passager
    FROM inserted;

    IF dbo.Complet(@Num_Vol) = 0 AND dbo.Occuper(@Num_Vol, @Code_Passager) = 1
    BEGIN
        SELECT TOP 1 @Num_Place = Num_Place
        FROM Places
        WHERE Num_Vol = @Num_Vol
        AND Etat_Place = 'Disponible'
        ORDER BY Num_Place;

        IF @Num_Place IS NOT NULL
        BEGIN
            -- Mettre à jour l'état de la place
            UPDATE Places
            SET Etat_Place = 'Occupée'
            WHERE Num_Vol = @Num_Vol
            AND Num_Place = @Num_Place;

            -- Afficher le numéro de la place disponible
            PRINT 'Une place a été attribuée automatiquement au passager ' +
            CAST(@Code_Passager AS VARCHAR(10)) + '. Numéro de place : ' + CAST(@Num_Place AS
            VARCHAR(10));
        END
        ELSE
        BEGIN
            PRINT 'Aucune place disponible pour le passager ' + CAST(@Code_Passager AS
            VARCHAR(10)) + '.';
        END
    END
END;

```

⇒ **Resultat :**

29)

```

CREATE TRIGGER question_29
ON Passagers
INSTEAD OF INSERT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @Code_Passager INT, @Nom_Passager VARCHAR(100), @Pre_Passager
    VARCHAR(100);

    DECLARE InsertCursor CURSOR FOR

```

```

SELECT Code_Passager, UPPER(Nom_Passager), UPPER(Pre_Passager)
FROM inserted;

OPEN InsertCursor;

FETCH NEXT FROM InsertCursor INTO @Code_Passager, @Nom_Passager, @Pre_Passager;

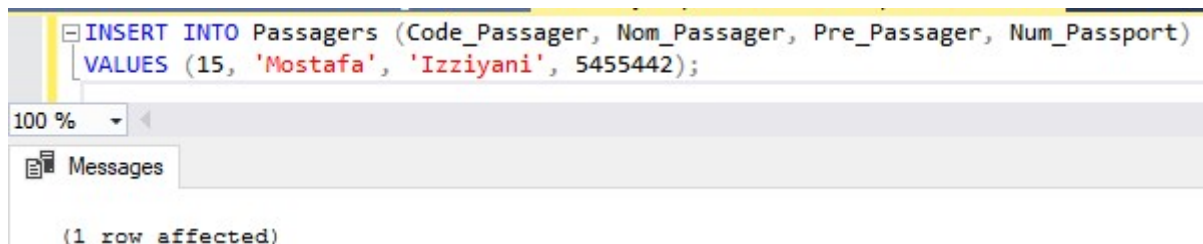
WHILE @@FETCH_STATUS = 0
BEGIN
    -- Vérifier l'unicité de la clé
    IF NOT EXISTS (SELECT 1 FROM Passagers WHERE Code_Passager = @Code_Passager)
    BEGIN
        INSERT INTO Passagers (Code_Passager, Nom_Passager, Pre_Passager)
        VALUES (@Code_Passager, @Nom_Passager, @Pre_Passager);
    END
    ELSE
    BEGIN
        PRINT 'La clé Code_Passager ' + CAST(@Code_Passager AS VARCHAR(10)) + '
existe déjà.';
    END

    FETCH NEXT FROM InsertCursor INTO @Code_Passager, @Nom_Passager,
@Pre_Passager;
END

CLOSE InsertCursor;
DEALLOCATE InsertCursor;
END;

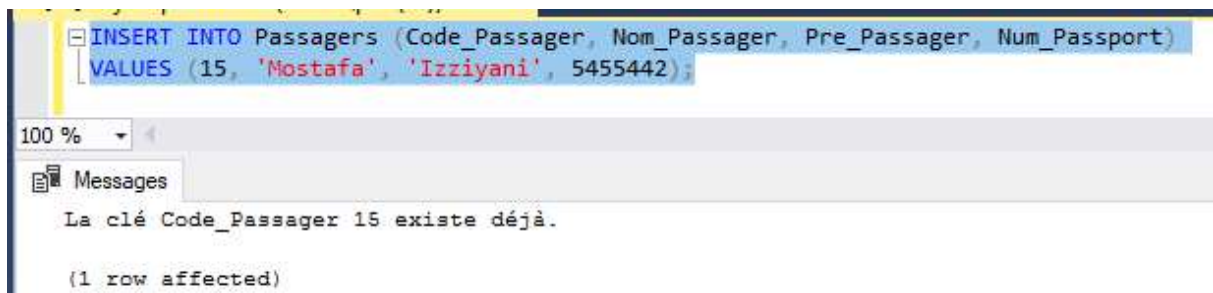
```

⇒ Resultat :



	Code_Passager	Nom_Passager	Pre_Passager	Num_Passport	Categorie	Num_Tel
	1	idriss	kh	68465	Moyen	123
	2	idr2	kh2	6656	Moyen	2565
	3	moha1	III3	6545	A	9756
	4	rrr1	rrrr2	96846	C	96456
	15	MOSTAFA	IZZİYANI	5455442	NULL	NULL

Re-exécuté la requête :



30)

```

CREATE TRIGGER question_30
ON Voyages
FOR INSERT
AS

```

```

BEGIN
    DECLARE @Num_Billet INT, @Num_Vol INT;

    -- Récupérer le numéro de billet et le numéro de vol de l'insertion
    SELECT @Num_Billet = i.Num_Billet, @Num_Vol = i.Num_Vol
    FROM inserted i;

    -- Vérifier si le billet est réservé pour le passager
    IF NOT EXISTS (SELECT * FROM Billets WHERE Num_Billet = @Num_Billet)
    BEGIN
        RAISERROR('Le billet n''est pas réservé pour le passager.', 16, 1);
        ROLLBACK TRANSACTION; -- Annuler l'insertion
        RETURN;
    END;
END;

```

⇒ **Resultat :**

31)

```

CREATE TRIGGER question_31
ON Voyages
AFTER INSERT
AS
BEGIN
    -- Execute the stored procedure to update Pilotes table
    EXEC question_26;
END;

```

⇒ **Resultat :**

	Num_Pilote	Nom_Pilote	Prenom_Pilote	NbrAvions	NbrVoyages	Statut
	1	idr1	kh1	2	3	Qualifie
	2	idr2	kh2	2	2	Qualifie
	3	idr3	kh3	1	0	Qualifie
	4	idr4	kh4	NULL	NULL	NULL

☐ INSERT INTO Voyages (Code_Passager, Num_Billet, Num_Vol, Num_place)
VALUES (4, 5, 4, 30);

100 %

Messages

(4 rows affected)

	Num_Pilote	Nom_Pilote	Prenom_Pilote	NbrAvions	NbrVoyages	Statut
▶	1	idr1	kh1	2	4	Qualifie
	2	idr2	kh2	2	3	Qualifie
	3	idr3	kh3	1	0	Qualifie
	4	idr4	kh4	0	0	Débitteur

32)

```

CREATE TRIGGER question_32
ON Voyages
INSTEAD OF INSERT
AS
BEGIN
    -- Check if the number of assigned seats exceeds the capacity of the plane
    IF EXISTS (
        SELECT v.Num_Vol
        FROM inserted i

```

```

        JOIN Vols v ON i.Num_Vol = v.Num_Vol
        JOIN Avions a ON v.Code_Avion = a.Num_Avion
        GROUP BY v.Num_Vol, a.Nbr_Place
        HAVING COUNT(*) > a.Nbr_Place
    )
BEGIN
    -- Raise an error if the capacity is exceeded
    RAISERROR ('The number of assigned seats exceeds the capacity of the plane.',
16, 1);
END
ELSE
BEGIN
    -- Insert the rows into the Voyages table if capacity is not exceeded
    INSERT INTO Voyages
    SELECT *
    FROM inserted;
END;
END;

```

⇒ **Resultat :**

Insérer 3 passagers dans un vols d'un avion qui contient sur 2 places :

```

INSERT INTO Voyages (Num_Vol, Code_Passager, Num_Billet)
VALUES
    (7, 3, 7),
    (7, 4, 8),
    (7, 15, 9);

```

Msg 50000, Level 16, State 1, Procedure question_32, Line 17 [Batch Start Line 0]
The number of assigned seats exceeds the capacity of the plane.

33)

-- Étape 1 : Créer la table User_Reservations pour stocké les users

```

CREATE TABLE User_Reservations (
    User_ID INT IDENTITY(1,1) PRIMARY KEY,
    Action VARCHAR(10), -- Insert, Delete, Update
    User_Name VARCHAR(50),
    Timestamp DATETIME,
    Reservation_ID INT, -- La clé primaire de la réservation modifiée
    Old_Values NVARCHAR(MAX), -- Anciennes valeurs des attributs (pour les mises à
jour)
    New_Values NVARCHAR(MAX) -- Nouvelles valeurs des attributs (pour les mises à
jour)
);

```

-- Étape 2 : Créer le déclencheur

```

CREATE TRIGGER ReservationAuditTrigger
ON Reservations
AFTER INSERT, DELETE, UPDATE
AS
BEGIN

```

```

    DECLARE @Action VARCHAR(10);
    DECLARE @User_Name VARCHAR(50);
    DECLARE @Timestamp DATETIME;
    DECLARE @Reservation_ID INT;

    -- Déterminer l'action effectuée (insertion, suppression, mise à jour)
    IF EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM deleted)
        SET @Action = 'Update';
    ELSE IF EXISTS (SELECT * FROM inserted)
        SET @Action = 'Insert';
    ELSE IF EXISTS (SELECT * FROM deleted)
        SET @Action = 'Delete';

```

```

-- Obtenir le nom d'utilisateur et l'horodatage actuels
SET @User_Name = SYSTEM_USER;
SET @Timestamp = GETDATE();

-- Obtenir l'ID de la réservation
IF @Action = 'Insert'
    SET @Reservation_ID = (SELECT Num_Reservation FROM inserted);
ELSE IF @Action = 'Delete'
    SET @Reservation_ID = (SELECT Num_Reservation FROM deleted);
ELSE
    SET @Reservation_ID = (SELECT Num_Reservation FROM inserted);

-- Obtenir les anciennes et nouvelles valeurs (uniquement pour les mises à jour)
DECLARE @Old_Values NVARCHAR(MAX), @New_Values NVARCHAR(MAX);
IF @Action = 'Update'
BEGIN
    SELECT @Old_Values = (SELECT * FROM deleted FOR JSON AUTO);
    SELECT @New_Values = (SELECT * FROM inserted FOR JSON AUTO);
END

-- Insérer les informations dans la table User_Reservation
INSERT INTO Audit_Reservations (Action, User_Name, Timestamp, Reservation_ID,
Old_Values, New_Values)
VALUES (@Action, @User_Name, @Timestamp, @Reservation_ID, @Old_Values,
@New_Values);
END;

```

⇒ **Resultat :**

On va tester cet trigger par cet requete :

```

-- Insertion
INSERT INTO Reservations (Num_Reservation, Date_Reservation, Date_Validation,
Etat_Reservation, Code_Agence, Code_Passager, Prix_Total)
VALUES (8, '2024-02-08', '2024-02-09', 'Validated', 454, 15, 100);

-- Mise à jour
UPDATE Reservations
SET Prix_Total = 150
WHERE Num_Reservation = 1;

-- Afficher les entrées dans la table User_Reservations
SELECT * FROM User_Reservations;

```

	User_ID	Action	User_Name	Timestamp	Reservation_ID	Old_Values	New_Values
	1	Insert	ID-KH\pro	2024-03-10 19:5...	8	NULL	NULL
	2	Update	ID-KH\pro	2024-03-10 19:5...	1	["Num_Reserv...	["Num_Reserv...

34)

```

CREATE TRIGGER question_34
ON Passagers
AFTER DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- Suppression en cascade dans la table Voyages
    DELETE FROM Voyages WHERE Code_Passager IN (SELECT Code_Passager FROM deleted);

    -- Suppression en cascade dans la table Reservations
    DELETE FROM Reservations WHERE Code_Passager IN (SELECT Code_Passager FROM
deleted);

END;

```

35) Meme question de 34

36)

```
CREATE TRIGGER question_36
ON Passagers
INSTEAD OF INSERT, UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    -- Parcourir les lignes affectées par l'insertion ou la mise à jour
    DECLARE cursorPassagers CURSOR FOR
    SELECT Code_Passager, REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(Num_Tel, '-', '.'), ' ', '.'),
'0', '0'), '0', '0'), '.', '')
    FROM inserted;

    OPEN cursorPassagers;

    -- Déclarer les variables @Code_Passager et @CorrectedPhone dans la portée appropriée
    DECLARE @Code_Passager INT, @CorrectedPhone VARCHAR(20);

    FETCH NEXT FROM cursorPassagers INTO @Code_Passager, @CorrectedPhone;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        -- Mettre à jour le numéro de téléphone dans la table Passagers
        UPDATE Passagers
        SET Num_Tel = @CorrectedPhone
        WHERE Code_Passager = @Code_Passager;

        FETCH NEXT FROM cursorPassagers INTO @Code_Passager, @CorrectedPhone;
    END;

    CLOSE cursorPassagers;
    DEALLOCATE cursorPassagers;
END;
```

37)

```
CREATE TRIGGER question_37
ON Vols
AFTER INSERT, UPDATE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @InvalidDateDepart VARCHAR(10);
    DECLARE @InvalidDateArrivee VARCHAR(10);

    -- Vérifier et corriger la date de départ
    SELECT TOP 1 @InvalidDateDepart = i.Date_Depart
    FROM inserted i
    WHERE
        i.Date_Depart NOT LIKE '%[^0-9-%]' AND LEN(i.Date_Depart) <= 10
        AND
        CHARINDEX('0', i.Date_Depart) = 0 AND CHARINDEX('Q', i.Date_Depart) = 0;

    IF @InvalidDateDepart IS NOT NULL
    BEGIN
        SET @InvalidDateDepart = REPLACE(REPLACE(@InvalidDateDepart, '0', '0'), 'Q', '0');
    END;

    -- Vérifier et corriger la date d'arrivée
    SELECT TOP 1 @InvalidDateArrivee = i.Date_Arrivee
    FROM inserted i
    WHERE
        i.Date_Arrivee NOT LIKE '%[^0-9-%]' AND LEN(i.Date_Arrivee) <= 10
        AND
        CHARINDEX('0', i.Date_Arrivee) = 0 AND CHARINDEX('Q', i.Date_Arrivee) = 0;

    IF @InvalidDateArrivee IS NOT NULL
    BEGIN
```

```

SET @InvalidDateArrivee = REPLACE(REPLACE(@InvalidDateArrivee, 'O', '0'), 'Q', '0');
END;

-- Mettre à jour les dates dans la table principale
UPDATE v
SET
    Date_Depart = CASE WHEN @InvalidDateDepart IS NOT NULL THEN @InvalidDateDepart ELSE
v.Date_Depart END,
    Date_Arrivee = CASE WHEN @InvalidDateArrivee IS NOT NULL THEN @InvalidDateArrivee ELSE
v.Date_Arrivee END
FROM Vols v
INNER JOIN inserted i ON v.Num_Vol = i.Num_Vol;

-- Afficher un message d'erreur pour les dates invalides
IF @InvalidDateDepart IS NOT NULL OR @InvalidDateArrivee IS NOT NULL
BEGIN
    PRINT 'Les dates de départ et/ou d'arrivée contiennent des caractères non valides. Les
caractères 'O' et 'Q' ont été remplacés par '0'.';
END;
END;

```

⇒ **Resultat :**

Pour Tester :

```

-- Insertion de valeurs incorrectes pour tester le déclencheur
INSERT INTO Vols (Num_Vol, Date_Depart, Date_Arrivee, Ville_Depart, Ville_Arrivee,
Code_Avion, Code_Pilote, Prix_Vol)
VALUES (11, '2Q24-02-15', '2024-02-20', 'Casablanca', 'Rabat', 4, 3, 25);

-- Mise à jour de valeurs incorrectes pour tester le déclencheur
UPDATE Vols
SET Date_Depart = '2Q24-02-15', Date_Arrivee = '2024-02-20'
WHERE Num_Vol = 8;

```

38)

Pour créer un déclencheur qui archive toutes les opérations de suppression sur la table Voyages, on va :

- 1- Créez une nouvelle table pour l'archivage des voyages supprimés. Cette table doit avoir une structure similaire à celle de la table Voyages, mais elle peut également inclure des colonnes supplémentaires pour enregistrer des informations telles que la date de suppression ou l'utilisateur qui a effectué la suppression.

```

CREATE TABLE Voyages_Archive (
    Code_Passager INT PRIMARY KEY,
    Num_Billet INT,
    Num_Vol INT,
    Num_Place INT,
    Date_Suppression DATETIME DEFAULT GETDATE()
);

```

- 2- Écrivez un déclencheur AFTER DELETE sur la table Voyages qui insère les lignes supprimées dans la table d'archivage.

```

CREATE TRIGGER question_38
ON Voyages
AFTER DELETE
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO Voyages_Archive (Code_Passager, Num_Billet, Num_Vol, Num_Place)
    SELECT Code_Passager, Num_Billet, Num_Vol, Num_Place
    FROM deleted;
END;

```

⇒ **Resultat:**

```
DELETE FROM Voyages WHERE Code_Passager=4 AND Num_Billet=6 AND Num_Vol=6 And Num_place=6;
```

100 %

Messages

(1 row affected)

```
SELECT * FROM Voyages_Archive;
```

00 %

Results Messages

	Code_Passager	Num_Billet	Num_Vol	Num_Place	Date_Suppression
1	4	6	6	6	2024-03-16 23:55:10.613

39)

Pour créer un déclencheur pour archiver les suppressions de réservations en fonction de leur nature de traitement :

- 1- On va Créer un table pour l'archive des réservations :

```
CREATE TABLE Reservations_Archive (
    Num_Reservation INT PRIMARY KEY,
    Date_Suppression DATETIME DEFAULT GETDATE(),
    Nature_Suppression VARCHAR(50)
);
```

- 2- On va Créer le déclencheur pour archiver les suppressions de réservations :

```
CREATE TRIGGER question_39
ON Reservations
AFTER DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @Num_Reservation INT, @Date_Suppression DATETIME, @Nature_Suppression VARCHAR(50);

    -- Sélectionner les réservations supprimées et leurs caractéristiques
    DECLARE DeletedReservations CURSOR FOR
    SELECT Num_Reservation, GETDATE(), CASE WHEN Date_Reservation < DATEADD(DAY, -10, GETDATE()) THEN
'Validée' ELSE 'Annulée' END
    FROM deleted;

    OPEN DeletedReservations;

    FETCH NEXT FROM DeletedReservations INTO @Num_Reservation, @Date_Suppression,
@Nature_Suppression;

    WHILE @@FETCH_STATUS = 0
    BEGIN
        -- Insérer les réservations supprimées dans la table d'archive
        INSERT INTO Reservations_Archive (Num_Reservation, Date_Suppression, Nature_Suppression)
        VALUES (@Num_Reservation, @Date_Suppression, @Nature_Suppression);

        FETCH NEXT FROM DeletedReservations INTO @Num_Reservation, @Date_Suppression,
@Nature_Suppression;
    END;

    CLOSE DeletedReservations;
    DEALLOCATE DeletedReservations;
END;
```

⇒ **Resultat :**

40)

```
CREATE VIEW question_40 AS
```

```
SELECT *
```

```
FROM Reservations
```

```
WHERE Etat_Reservation = 'Valide' AND Code_Agence = 001;
```

⇒ Resultat:



	Num_Reservation	Date_Reservation	Date_Validation	Etat_Reservation	Code_Agence	Code_Passager	Prix_Total
1	5	2024-03-18	2024-03-20	Valide	1	4	572
2	6	2024-03-18	NULL	Valide	1	3	532

41)

42)