Université Abdelmalek Essaadi

Faculté des Sciences et Techniques

Tanger

Rapport Réalisation TP T-SQL

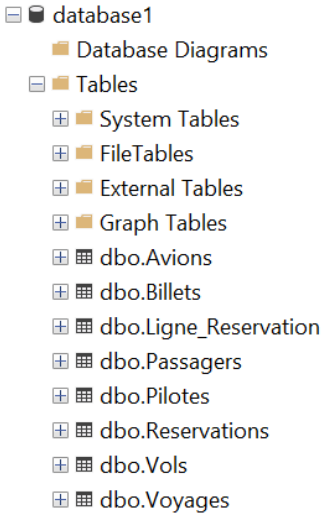
Réalisé par : ARICHI Fatima zahra et ELMEZIANE Chaima.

Encadré par : EZZIYYANI Mostafa

Année Universitaire : 2023/2024

On commence tout d'abord par créer notre base de données de la manière suivante :

```
CREATE DATABASE database1;
```



Et après on va insérer les données pour chaque table nécessite un remplissage.

Exercice 1 :

```
CREATE PROCEDURE EX1
@max_number INT
AS
BEGIN
-- Create a temporary table to store the results
CREATE TABLE #TMP (
NBR INT,
Nbr_paire INT,
Nbr_Impaire INT
);
DECLARE @i INT;
DECLARE @digit_sum INT;
DECLARE @num_str VARCHAR(255);
DECLARE @num_len INT;
DECLARE @even_count INT;
DECLARE @odd_count INT;
-- Initialize variables
SET @i = 0;
-- Loop to iterate through all integers less than max_number
WHILE @i < @max_number
BEGIN
SET @i = @i + 1;
SET @num_str = CAST(@i AS VARCHAR(255));
SET @digit_sum = 0;
SET @num_len = LEN(@num_str);
SET @even_count = 0;
SET @odd_count = 0;
-- Calculate the sum of digits and count even and odd digits
WHILE @num_len > 0
BEGIN
SET @digit_sum = @digit_sum + CAST(SUBSTRING(@num_str, @num_len, 1) AS INT);
IF CAST(SUBSTRING(@num_str, @num_len, 1) AS INT) % 2 = 0
SET @even_count = @even_count + 1;
ELSE SET @odd_count = @odd_count + 1;
SET @num_len = @num_len - 1;
END
-- Check if the sum of digits is equal to 6
IF @digit_sum = 6
BEGIN
-- Insert the results into the temporary table
INSERT INTO #TMP (NBR, Nbr_paire, Nbr_Impaire) VALUES (@i, @even_count,
@odd_count);
END
END
-- Select the results from the temporary table
SELECT * FROM #TMP;
-- Drop the temporary table
DROP TABLE #TMP;
```

Results | Messages

	NBR	Nbr_paire	Nbr_Impaire
1	6	1	0
2	15	0	2
3	24	2	0
4	33	0	2
5	42	2	0
6	51	0	2
7	60	2	0

```
        END
        EXECUTE EX1 @max_number = 80;
```

Exercice 2 :

```
        CREATE FUNCTION EX2
        (
            @nombre INT
        )
        RETURNS VARCHAR(MAX)
        AS
        BEGIN
            DECLARE @resultat VARCHAR(MAX) = '';
            DECLARE @quotient INT;
            DECLARE @reste INT;

            -- Calcul du code binaire
            WHILE @nombre > 0
            BEGIN
                SET @quotient = @nombre / 2;
                SET @reste = @nombre % 2;
                SET @resultat = CAST(@reste AS VARCHAR(1)) + @resultat;
                SET @nombre = @quotient;
            END

            RETURN @resultat;
        END

        -- Utilisation de la fonction pour calculer le code binaire de l'entier 10
               SELECT dbo.EX2(10) AS CodeBinaireDe10;
```

Results Messages

	CodeBinaireDe10
1	1010

Exercice 3 :

```
        create function EX3(@mot varchar(100)) returns varchar(100)
        as
        begin
        declare @motReverser varchar(100);
        set @motReverser = REVERSE(@mot);
        declare @result varchar(100)
        if @motReverser = @mot
        set @result = @mot + ' est Palindrome'
        else
        set @result = @mot + ' nest pas Palindrome'
        return @result
        END
        -- Appel de la fonction EX3 avec un mot
        DECLARE @motTest varchar(100);
        SET @motTest = 'radar'; -- Vous pouvez remplacer 'radar' par n'importe quel mot que vous
        souhaitez tester

        -- Appel de la fonction et stockage du résultat dans une variable
        DECLARE @resultat varchar(100);
        SET @resultat = dbo.EX3(@motTest);

        -- Affichage du résultat
               SELECT @resultat AS Resultat;
```

Results Messages

	Resultat
1	radar est Palindrome

Exercice 4 :

```
        create function EX4(@inputString varchar(MAX)) returns int
        as
        begin
         declare @wordCount INT = 0;
         declare @startIndex INT = 1;
         -- Ignorer les espaces au début de la chaîne
         WHILE @startIndex <= LEN(@inputString) AND SUBSTRING(@inputString, @startIndex, 1)
        = ' '
         SET @startIndex = @startIndex + 1;
         WHILE @startIndex <= LEN(@inputString)
         BEGIN
         -- Trouver la position du prochain espace
         DECLARE @spaceIndex INT = CHARINDEX(' ', @inputString, @startIndex);
         -- Si aucun espace n'est trouvé, c'est le dernier mot
         IF @spaceIndex = 0
         SET @spaceIndex = LEN(@inputString) + 1;
```

```
        -- Incrémenter le compteur de mots
        SET @wordCount = @wordCount + 1;
        -- Ignorer les espaces suivant le mot
        WHILE @spaceIndex <= LEN(@inputString) AND SUBSTRING(@inputString, @spaceIndex,
       1) = ' '
        SET @spaceIndex = @spaceIndex + 1;
        -- Mettre à jour l'index de départ pour la prochaine itération
        SET @startIndex = @spaceIndex;
        END
        RETURN @wordCount;
       END
       -- Appel de la fonction EX4 avec une chaîne de caractères
       DECLARE @phrase varchar(MAX);
       SET @phrase = 'master intelligence artificielle'; -- Vous pouvez
       remplacer cette phrase par n'importe quelle chaîne que vous souhaitez
       tester

       -- Appel de la fonction et stockage du résultat dans une variable
       DECLARE @nombreMots int;
       SET @nombreMots = dbo.EX4(@phrase);

       -- Affichage du résultat
              SELECT @nombreMots AS NombreDeMots;
```

Results Messages

	NombreDeMots
1	3

```
Exercice 5 :
       create function dbo.EX5
       (
        @mainString NVARCHAR(MAX),
        @substring NVARCHAR(MAX)
       )
       returns int
       AS
       BEGIN
        DECLARE @occurrenceCount INT = 0;
        DECLARE @startIndex INT = 1;
        WHILE @startIndex <= LEN(@mainString)
        BEGIN
        SET @startIndex = CHARINDEX(@substring, @mainString, @startIndex);
        IF @startIndex = 0
        BREAK; -- Sortir de la boucle si aucune occurrence n'est trouvée
        SET @occurrenceCount += 1;
        SET @startIndex += LEN(@substring); -- Passer à la position suivante après l'occurrence
       trouvée
        END
        RETURN @occurrenceCount;
       END

       -- Appel de la fonction EX5 avec une chaîne principale et une sous-chaîne
       DECLARE @chainePrincipale NVARCHAR(MAX);
       DECLARE @sousChaine NVARCHAR(MAX);

       SET @chainePrincipale = 'Master intelligence artificielle et science de
       données'; -- Remplacez cette chaîne par la chaîne principale que vous
       souhaitez analyser
       SET @sousChaine = 'ce'; -- Remplacez cette chaîne par la sous-chaîne que vous souhaitez
       compter

       -- Appel de la fonction et stockage du résultat dans une variable
       DECLARE @nombreOccurences INT;
       SET @nombreOccurences = dbo.EX5(@chainePrincipale, @sousChaine);

       -- Affichage du résultat
              SELECT @nombreOccurences AS NombreOccurences;

Exercice 6 :
       CREATE FUNCTION EX6
       (
        @inputString NVARCHAR(MAX)
       )
       RETURNS NVARCHAR(MAX)
       AS
       BEGIN
        DECLARE @longestWord NVARCHAR(MAX) = '';
```

Results Messages

	NombreOccurences
1	2

```
 DECLARE @startIndex INT = 1;
 DECLARE @currentWord NVARCHAR(MAX);
 WHILE @startIndex <= LEN(@inputString)
 BEGIN
 -- Ignorer les espaces au début de la chaîne
 WHILE @startIndex <= LEN(@inputString) AND SUBSTRING(@inputString, @startIndex,
1) = ' '
 SET @startIndex = @startIndex + 1;
 -- Trouver la position du prochain espace
 DECLARE @spaceIndex INT = CHARINDEX(' ', @inputString, @startIndex);
 -- Si aucun espace n'est trouvé, c'est le dernier mot
 IF @spaceIndex = 0
 SET @spaceIndex = LEN(@inputString) + 1;
 -- Extraire le mot actuel
 SET @currentWord = SUBSTRING(@inputString, @startIndex, @spaceIndex -
@startIndex);
 -- Vérifier si le mot actuel est plus long que le plus long mot trouvé jusqu'à présent
 IF len(@currentWord) > len(@longestWord)
 SET @longestWord = @currentWord;
 -- Mettre à jour l'index de départ pour la prochaine itération
 SET @startIndex = @spaceIndex;
 END
 RETURN @longestWord;
END
-- Appel de la fonction EX6 avec une chaîne de caractères
DECLARE @phrase NVARCHAR(MAX);
SET @phrase = 'master intelligence artificielle et sciences de
données'; -- Remplacez cette phrase par la chaîne que vous souhaitez
analyser

-- Appel de la fonction et stockage du résultat dans une variable
DECLARE @motLePlusLong NVARCHAR(MAX);
SET @motLePlusLong = dbo.EX6(@phrase);

-- Affichage du résultat
	SELECT @motLePlusLong AS MotLePlusLong;
```

Results Messages

	MotLePlusLong
1	intelligence

```
Exercice 7 :
	CREATE PROCEDURE EX7
	 @minutes INT
	AS
	BEGIN
	 DECLARE @years INT, @months INT, @days INT, @hours INT, @remainingMinutes INT
	 -- Calculate years
	 SET @years = @minutes / (60 * 24 * 365)
	 SET @remainingMinutes = @minutes % (60 * 24 * 365)
	 -- Calculate months
	 SET @months = @remainingMinutes / (60 * 24 * 30)
	 SET @remainingMinutes = @remainingMinutes % (60 * 24 * 30)
	 -- Calculate days
	 SET @days = @remainingMinutes / (60 * 24)
	 SET @remainingMinutes = @remainingMinutes % (60 * 24)
	 -- Calculate hours
	 SET @hours = @remainingMinutes / 60
	 SET @remainingMinutes = @remainingMinutes % 60
	 -- Print the result
	 PRINT CAST(@years AS VARCHAR) + ' Années ' +
	 CAST(@months AS VARCHAR) + ' Mois ' +
	 CAST(@days AS VARCHAR) + ' Jours ' +
	 CAST(@hours AS VARCHAR) + ' Heures ' +
	 CAST(@remainingMinutes AS VARCHAR) + ' Minutes'
	END
	-- Appel de la procédure EX7 avec un nombre de minutes
	DECLARE @minutes INT;
	SET @minutes = 3600; -- Remplacez ce nombre par le nombre
	de minutes que vous souhaitez convertir

	-- Exécution de la procédure
		EXEC dbo.EX7 @minutes;
```

Messages

```
0 Années 0 Mois 2 Jours 12 Heures 0 Minutes

Completion time: 2024-03-12T23:59:06.5828763+01:00
```

```
Exercice 8 :
Exercice 9 :
	CREATE PROCEDURE AfficherReservNV
```

```
        AS
        BEGIN
            SELECT * FROM Reservations WHERE CONVERT(VARCHAR(MAX), Etat_Reservation) = 'Etat77';
        END
        GO

         exec AfficherReservNV ;
```

	Num_Reservation	Date_Reservation	Date_Validation	Etat_Reservation	Code_Agence	Code_Passager	Prix_Total
1	85	2024-03-12	2024-03-12	Etat77	88	4663	5524.00

```
Exercice 10 :
        Create procedure AfficherVol
         @NumVol INT
        as
        Begin
         select * from Vols where Num_vol = @Numvol
        END

                exec AfficherVol @NumVol = 516
```

	Num_Vol	Date_Depart	Heure_Depart	Ville_Depart	Ville_Arrivee	Code_Avion	Code_Pilote	Prix_Vol
1	516	2023-03-17	22:00:00.0000000	VilleDepart84	VilleArrivee98	574	609	768.00

```
Exercice 11 :
        Create procedure AfficherVolpil
         @NumVol INT
        as
        begin
         select * from Vols,Pilotes where Num_vol = @NumVol and Code_Pilote = Num_Pilote
        END

                exec AfficherVolpil
        @NumVol = 516
```

	Num_Vol	Date_Depart	Heure_Depart	Ville_Depart	Ville_Arrivee	Code_Avion	Code_Pilote	Prix_Vol	Num_Pilote	Nom_Pilote	Prenom_Pilote
1	516	2023-03-17	22:00:00.0000000	VilleDepart84	VilleArrivee98	574	609	768.00	609	NomPilote4959	PrenomPilote9399

```
Exercice 12 :
        CREATE PROCEDURE AfficherReservBillet
        AS
        BEGIN
            SELECT *
            FROM Reservations AS r
            INNER JOIN Billets AS b ON b.Num_Reservation = r.Num_Reservation
            WHERE CAST(r.Etat_Reservation AS VARCHAR(MAX)) = 'Etat33';
        END
        GO

                exec AfficherReservBillet
```

	Num_Reservation	Date_Reservation	Date_Validation	Etat_Reservation	Code_Agence	Code_Passager	Prix_Total	Num_Billet	Num_Reservation
1	1149	2024-03-12	2024-03-12	Etat33	41	2420	3114.00	7352	1149

```
Exercice 13 :
        Create procedure AfficherVoyAvDesc
        as
        begin
         select Code_Avion, count(*) as nbr_voyages from Vols Group by Code_Avion Order
        by nbr_voyages Desc
        END

                exec AfficherVoyAvDesc
```

	Code_Avion	nbr_voyages
1	239	15
2	574	15

```
Exercice 14 :
        Create procedure EX14
         @CodePas INT
        as
        begin
         select count(*) as nbr_voyages from Voyages where Code_Passager = @CodePas
        END

                exec EX14 @CodePas = 615
```

	nbr_voyages
1	225

```
Exercice 15 :
        Create function EX15(@Num_Vol INT)
        returns DECIMAL(10, 2)
        as
         begin
         declare @CostPrice DECIMAL(10, 2);
         -- Calcul du prix de revient en fonction des coûts associés (à adapter selon votre modèle)
         select @CostPrice = (Prix_Vol * Nbr_Place)
         from Vols, Avions
         where Code_Avion = Num_Avion and Num_Vol = @Num_Vol;
```

```
         RETURN @CostPrice;
                END
        -- Appel de la fonction EX15 avec un numéro de vol
        DECLARE @NumVol INT;
        SET @NumVol = 516; -- Remplacez 123 par le numéro de vol que vous
        souhaitez utiliser

        -- Appel de la fonction et stockage du résultat dans une variable
        DECLARE @PrixRevient DECIMAL(10, 2);
        SET @PrixRevient = dbo.EX15(@NumVol);

        -- Affichage du résultat
               SELECT @PrixRevient AS PrixRevient;
```

Results | Messages

	PrixRevient
1	254976.00

```
Exercice 16 :
        Create procedure DeleteEtat
        AS
        begin
        delete Reservations where Etat_Reservation like 'Etat77'
        END

                exec DeleteEtat
Exercice 17 :
        CREATE PROCEDURE EX17
            @CodePass INT,
            @NumBillet INT,
            @NumVol INT,
            @NumPlace INT
        AS
        BEGIN
            IF EXISTS (SELECT * FROM Voyages WHERE Num_Billet = @NumBillet AND Code_passager =
        @CodePass AND Num_Vol = @NumVol)
            BEGIN
                RAISERROR ('Cet enregistrement existe déjà !', 12, 1);
                RETURN;
            END
            ELSE IF NOT EXISTS (SELECT * FROM Billets, Passagers, Vols WHERE Num_Billet =
        @NumBillet AND Code_Passager = @CodePass AND Num_Vol = @NumVol)
            BEGIN
                RAISERROR ('Le billet ne correspond pas au passager/vol !', 12, 1);
                RETURN;
            END
            ELSE IF EXISTS (SELECT * FROM Voyages WHERE Num_Place = @NumPlace AND Num_Vol =
        @NumVol)
            BEGIN
                RAISERROR ('La place est déjà prise !', 12, 1);
                RETURN;
            END;

            -- Insert the record into the Voyages table if all
        constraints are satisfied
            INSERT INTO Voyages (Code_Passager, Num_Billet,
        Num_Vol, Num_Place)
            VALUES (@CodePass, @NumBillet, @NumVol, @NumPlace);

            SELECT 'Enregistrement inséré avec succès.' AS Message;
        END

       EXEC EX17 @CodePass = 615, @NumBillet = 1166, @NumVol = 516, @NumPlace = 570;
Exercice 18 :
        Create procedure InsererLigneReserv (@NumLigne INT,@NumOrdre INT,@NumVol
        INT,@NumReservation INT)
        as
        begin
         DECLARE @VilleDepartNvReserv NVARCHAR(255);
         DECLARE @VilleArriveeReservPrec NVARCHAR(255);
        DECLARE @NbrPlacesOccupees INT;
        Select @VilleArriveeReservPrec = v.Ville_Arrivee
        from Vols as v INNER JOIN Ligne_Reservation as rv on v.Num_Vol = rv.Num_Vol
        WHERE rv.Num_Reservation = @NumReservation and rv.Num_Order = @NumOrdre -
```

Results | Messages

	Message
1	Enregistrement inséré avec succès.

```
1 ;
Select @VilleDepartNvReserv = Ville_Depart from Vols where Num_Vol =
@NumVol ;
Select @NbrPlacesOccupees = count(*) from Ligne_Reservation where Num_Vol
= @NumVol;
IF EXISTS (select * from Ligne_Reservation where Num_Ligne = @NumLigne)
 BEGIN
 RAISERROR ('Cet Ligne de réservation existe déja !', 12, 1)
Return
 End
else IF @NumOrdre != (Select max(Num_Order) + 1 from Ligne_Reservation where
Num_Reservation = @NumReservation )
 BEGIN
 RAISERROR ('le numéro d''ordre de cette réservation n''est pas sérial !',
12, 1)
Return
 End
else IF @VilleArriveeReservPrec is not null and @VilleArriveeReservPrec !=
@VilleDepartNvReserv
 BEGIN
 RAISERROR ('La ville de départ du vol ne correspond pas à la ville d''arrivée du vol
précédent.', 12, 1)
Return
 End
else If @NbrPlacesOccupees >= (Select Nbr_Place from Avions a INNER JOIN Vols v
on v.Code_Avion = a.Num_Avion where v.Num_Vol = @NumVol)
 BEGIN
 RAISERROR ('Il n''y a plus de places disponibles dans l''avion pour ce vol!', 12, 1)
Return
 End
 -- Si toutes les contraintes sont vérifiées, insérer l'enregistrement dans la table
Voyages
 Insert into Ligne_Reservation (Num_Ligne,Num_Order, Num_Vol, Num_Reservation)
VALUES (@NumLigne,@NumOrdre, @NumVol, @NumReservation);
 SELECT 'Enregistrement inséré avec succès.' AS 'Message';
END

        EXEC InsererLigneReserv @NumLigne = 563, @NumOrdre = 805, @NumVol = 1250,
@NumReservation = 7319;
```

Cas 1 :

```
EXEC InsererLigneReserv @NumLigne = 563, @NumOrdre = 2, @NumVol = 1207, @NumReservation = 7319;
```

110 %

Messages

```
Msg 50000, Level 12, State 1, Procedure InsererLigneReserv, Line 18 [Batch Start Line 472]
Cet Ligne de réservation existe déja !

Completion time: 2024-03-15T02:31:48.5181830+01:00
```

Cas 2 :

```
EXEC InsererLigneReserv @NumLigne = 503, @NumOrdre = 3, @NumVol = 1207, @NumReservation = 7319;
```

110 %

Messages

```
Msg 50000, Level 12, State 1, Procedure InsererLigneReserv, Line 24 [Batch Start Line 472]
le numéro d'ordre de cette réservation n'est pas sérial !

Completion time: 2024-03-15T02:32:28.3070392+01:00
```

Cas 3 :

```
EXEC InsererLigneReserv @NumLigne = 3, @NumOrdre = 100, @NumVol = 127, @NumReservation = 739;
```

110 %

Results Messages

	Message
1	Enregistrement inséré avec succès.

Exercice 19 :

```
CREATE PROCEDURE EX19
AS
BEGIN
 DECLARE @sql NVARCHAR(MAX);
 -- Ajouter la colonne Nbr_Res
 SET @sql = 'ALTER TABLE Vols ADD Nbr_Res INT DEFAULT 0;';
 EXECUTE sp_executesql @sql;
 -- Ajouter la colonne Nbr_Att
 SET @sql = 'ALTER TABLE Vols ADD Nbr_Att INT DEFAULT 0;';
```

```
        EXECUTE sp_executesql @sql;
       -- Mettre à jour les colonnes nouvellement ajoutées à 0
        SET @sql = 'UPDATE Vols SET Nbr_Res = 0, Nbr_Att = 0';
        EXECUTE sp_executesql @sql;
       END
             exec EX19
```

	Num_Vol	Date_Depart	Heure_Depart	Ville_Depart	Ville_Arrivee	Code_Avion	Code_Pilote	Prix_Vol	Nbr_Res	Nbr_Att
1	516	2023-03-17	22:00:00.0000000	VilleDepart84	VilleArrivee98	574	609	768.00	0	0
2	614	2023-04-29	21:07:00.0000000	VilleDepart45	VilleArrivee68	574	609	584.00	0	0
3	1207	2023-08-24	10:42:00.0000000	VilleDepart79	VilleArrivee74	239	830	454.00	0	0
4	1250	2023-12-07	21:18:00.0000000	VilleDepart25	VilleArrivee57	239	830	171.00	0	0
5	1665	2023-01-26	04:09:00.0000000	VilleDepart96	VilleArrivee67	574	609	807.00	0	0
6	1929	2023-12-18	21:53:00.0000000	VilleDepart7	VilleArrivee11	574	609	690.00	0	0
7	1970	2023-10-18	09:28:00.0000000	VilleDepart8	VilleArrivee43	574	609	393.00	0	0

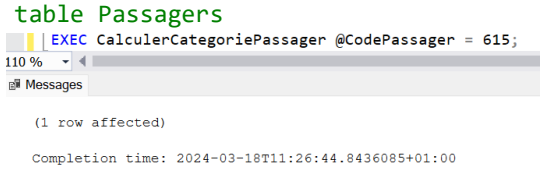
```
Exercice 20 :
       CREATE PROCEDURE EX20
       @NumVol INT
       AS
       BEGIN
        DECLARE @NbrRes INT;
        DECLARE @NbrAtt INT;
        -- Calculer le nombre de places réservées pour le vol donné
        SELECT @NbrRes = COUNT(*) FROM Ligne_Reservation WHERE Num_Vol = @NumVol;
        -- Calculer le nombre de places attribuées pour le vol donné
        SELECT @NbrAtt = COUNT(*) FROM Ligne_Reservation as lr, Avions as av , Vols as vo
       WHERE vo.Num_Vol = @NumVol AND av.Num_Avion = vo.Code_Avion and av.Nbr_Place IS NOT
       NULL;
        -- Mettre à jour les colonnes Nbr_Res et Nbr_Att dans la table Vols
        UPDATE Vols
        SET Nbr_Res = @NbrRes,
        Nbr_Att = @NbrAtt
        WHERE Num_Vol = @NumVol;
       END;

       exec EX20 @NumVol = 1
       use database1
       select * from Vols
```

110 %

Results Messages

	Num_Vol	Date_Depart	Heure_Depart	Ville_Depart	Ville_Arrivee	Code_Avion	Code_Pilote	Prix_Vol	Nbr_Res	Nbr_Att
1	516	2023-03-17	22:00:00.0000000	VilleDepart84	VilleArrivee98	574	609	768.00	0	0
2	614	2023-04-29	21:07:00.0000000	VilleDepart45	VilleArrivee68	574	609	584.00	0	0

```
Exercice 21 :
       CREATE PROCEDURE CalculerCategoriePassager
        @CodePassager INT
       AS
       BEGIN
        DECLARE @NombreVoyages INT;
        DECLARE @MontantTotal DECIMAL(10, 2);
        DECLARE @Categorie NVARCHAR(50);
        -- Calculer le nombre de voyages effectués par le passager donné
        SELECT @NombreVoyages = COUNT(*)
        FROM Voyages,vols
        WHERE Code_Passager = @CodePassager
        AND YEAR(Date_Depart) = YEAR(GETDATE()); -- Filtrer les voyages de l'année en cours
        -- Calculer le montant total dépensé par le passager donné
        SELECT @MontantTotal = SUM(Prix_Vol)
        FROM Voyages V
        INNER JOIN Vols VL ON V.Num_Vol = VL.Num_Vol
        WHERE V.Code_Passager = @CodePassager
        AND YEAR(VL.Date_Depart) = YEAR(GETDATE()); -- Filtrer les voyages de l'année en
       cours
        -- Déterminer la catégorie du passager en fonction des critères
        IF @NombreVoyages > 20 AND @MontantTotal > 200000
        SET @Categorie = 'Très Actif';
        ELSE IF @NombreVoyages > 20
        SET @Categorie = 'Actif';
        ELSE
        SET @Categorie = 'Moyen';
        -- Mettre à jour la colonne "Categorie" dans la table Passagers
        UPDATE Passagers
        SET Categorie = @Categorie
        WHERE Code_Passager = @CodePassager;
       END;
```



```
EXEC CalculerCategoriePassager @CodePassager = 615;
```

110 %

Messages

```
(1 row affected)

Completion time: 2024-03-18T11:26:44.8436085+01:00
```

```
Exercice 22 :
       CREATE PROCEDURE EX22
       AS
       BEGIN
        -- Créer une table temporaire pour stocker les résultats
        CREATE TABLE #NumberOfVoyagesPerPassager (
```

```
        Code_Passager INT,
        Nom_Passager VARCHAR(255),
        Prenom_Passager VARCHAR(255),
        Nombre_Voyages INT
        );
        -- Remplir la table temporaire avec le nombre de voyages par passager
        INSERT INTO #NumberOfVoyagesPerPassager (Code_Passager, Nom_Passager,
      Prenom_Passager, Nombre_Voyages)
        SELECT
        P.Code_Passager,
        P.Nom_Passager,
        P.Pre_Passager,
        COUNT(V.Code_Passager) 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;
        -- Afficher le résultat
        SELECT
        Code_Passager,
        Nom_Passager,
        Prenom_Passager,
        Nombre_Voyages
        FROM
        #NumberOfVoyagesPerPassager;
        -- Supprimer la table temporaire
        DROP TABLE #NumberOfVoyagesPerPassager;
      END;
```

Exercice 23 :

```
      CREATE PROCEDURE CalculateCostOfFlights
      AS
      BEGIN
        -- Créer une table temporaire pour stocker les résultats
        CREATE TABLE #CostOfFlights (
        Num_Vol INT,
        Date_Depart DATE,
        Heure_Depart TIME,
        Ville_Depart VARCHAR(255),
        Ville_Arrivee VARCHAR(255),
        Code_Avion INT,
        Code_Pilote INT,
        Prix_Vol DECIMAL(10, 2),
        CostOfFlight DECIMAL(10, 2)
        );
        -- Remplir la table temporaire avec le coût de revient de chaque vol
        INSERT INTO #CostOfFlights (Num_Vol, Date_Depart, Heure_Depart, Ville_Depart,
      Ville_Arrivee, Code_Avion, Code_Pilote, Prix_Vol, CostOfFlight)
        SELECT
        V.Num_Vol,
        V.Date_Depart,
        V.Heure_Depart,
        V.Ville_Depart,
        V.Ville_Arrivee,
        V.Code_Avion,
        V.Code_Pilote,
        V.Prix_Vol,
        (V.Prix_Vol + A.Poids_Max * 0.01) AS CostOfFlight
        FROM
        Vols V
        INNER JOIN
        Avions A ON V.Code_Avion = A.Num_Avion;
        -- Afficher le résultat
        SELECT
        Num_Vol,
        Date_Depart,
        Heure_Depart,
```

```
        Ville_Depart,
        Ville_Arrivee,
        Code_Avion,
        Code_Pilote,
        Prix_Vol,
        CostOfFlight
        FROM
        #CostOfFlights;
        -- Supprimer la table temporaire
        DROP TABLE #CostOfFlights;
              END;
```

```
exec CalculateCostOfFlights
```

110 %

Results | Messages

	Num_Vol	Date_Depart	Heure_Depart	Ville_Depart	Ville_Arrivee	Code_Avion	Code_Pilote	Prix_Vol	CostOfFlight
1	516	2023-03-17	22:00:00.0000000	VilleDepart84	VilleArrivee98	574	609	768.00	1134.64
2	614	2023-04-29	21:07:00.0000000	VilleDepart45	VilleArrivee68	574	609	584.00	950.64

Exercice 24 :

```
CREATE FUNCTION CalculerNombreTotalVoyages()
RETURNS INT
AS
BEGIN
 DECLARE @NombreTotalVoyages INT;
 -- Calculer le nombre total de
voyages
 SELECT @NombreTotalVoyages =
COUNT(*)
 FROM Voyages;
 -- Retourner le nombre total de voyages
 RETURN @NombreTotalVoyages;
       END;
```

```
DECLARE @Resultat INT;
SET @Resultat = dbo.CalculerNombreTotalVoyages(); -- Assurez-vous de spécifier le schéma approprié si nécessaire

-- Affichage du résultat
SELECT @Resultat AS NombreTotalVoyages;
```

110 %

Results | Messages

	NombreTotalVoyages
1	3376

Exercice 25 :

```
CREATE PROCEDURE AfficherPilotesPilotagePourc
 @Pourcentage DECIMAL(5, 2)
AS
BEGIN
 DECLARE @NombreTotalAvions INT;
 -- Calculer le nombre total d'avions dans la compagnie
 SELECT @NombreTotalAvions = COUNT(*) FROM Avions;
 -- Afficher les pilotes qui ont piloté plus d'un pourcentage donné des avions
 SELECT P.Num_Pilote, P.Nom_Pilote, P.Prenom_Pilote,
 COUNT(*) AS NombreAvionsPilotes,
 ROUND(CAST(COUNT(*) AS DECIMAL(10, 2)) / @NombreTotalAvions * 100, 2) AS
PourcentagePilotage
 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 ROUND(CAST(COUNT(*) AS
DECIMAL(10, 2)) / @NombreTotalAvions *
100, 2) >
@Pourcentage;
END ;
```

```
EXEC AfficherPilotesPilotagePourc @Pourcentage = 2.00;
```

110 %

Results | Messages

	Num_Pilote	Nom_Pilote	Prenom_Pilote	NombreAvionsPilotes	PourcentagePilotage
1	609	NomPilote4959	PrenomPilote9399	15	50.0000000000000
2	830	NomPilote2577	PrenomPilote3114	15	50.0000000000000

Exercice 26 :

```
CREATE PROCEDURE AjouterColonnesPilotes
AS
BEGIN
 DECLARE @NombreTotalAvions INT;
 DECLARE @sql NVARCHAR(MAX);

 -- Ajouter les colonnes NbrAvions, NbrVoyages et Statut à la table Pilotes
 SET @sql = '
 ALTER TABLE Pilotes
 ADD NbrAvions INT;
 ALTER TABLE Pilotes
 ADD NbrVoyages INT;
 ALTER TABLE Pilotes
 ADD Statut NVARCHAR(50);
 ';
 EXEC sp_executesql @sql;
 -- Calculer le nombre total d'avions dans la compagnie
 SELECT @NombreTotalAvions = COUNT(*) FROM Avions;
 -- Mettre à jour les colonnes ajoutées pour chaque pilote
 SET @sql = '
 UPDATE Pilotes
```

```
        SET NbrAvions = (SELECT COUNT(*) FROM Vols WHERE Code_Pilote =
       Pilotes.Num_Pilote),
        NbrVoyages = (SELECT COUNT(*) FROM Vols WHERE Code_Pilote =
       Pilotes.Num_Pilote),
        Statut =
        CASE
        WHEN (SELECT COUNT(*) FROM Vols WHERE Code_Pilote =
       Pilotes.Num_Pilote) > 0 THEN
        CASE
        WHEN CAST((SELECT COUNT(*) FROM Vols WHERE Code_Pilote =
       Pilotes.Num_Pilote) AS DECIMAL) / ' +
        ISNULL(CAST(@NombreTotalAvions AS NVARCHAR), '1') + ' > 0.5
       THEN ''Expert''
        WHEN CAST((SELECT COUNT(*) FROM Vols WHERE Code_Pilote =
       Pilotes.Num_Pilote) AS DECIMAL) / ' +
        ISNULL(CAST(@NombreTotalAvions AS NVARCHAR), '1') + ' >=
       0.05 THEN ''Qualifie''
        ELSE ''Débiteur''
        END
        ELSE ''Débiteur'' -- Si le pilote n''a effectué aucun vol
        END;
        ';
        EXEC sp_executesql @sql;
              END;
```

```
exec AjouterColonnesPilotes
```

110 %

Messages

```
(30 rows affected)

Completion time: 2024-03-19T14:44:50.5410793+01:00
```

Exercice 27 :

```
       CREATE PROCEDURE ProposerBillets
        @VilleDepart NVARCHAR(100),
        @VilleArrivee NVARCHAR(100),
        @NombreEscales INT = NULL
       AS
       BEGIN
        -- Créer une table temporaire pour stocker les billets proposés
        CREATE TABLE #BilletsProposes (
        Num_Billet INT,
        Num_Reservation INT,
        Prix_Total DECIMAL(10, 2)
        );
        -- Insérer les billets correspondant aux critères dans la table temporaire
        INSERT INTO #BilletsProposes (Num_Billet, Num_Reservation, Prix_Total)
        SELECT B.Num_Billet, B.Num_Reservation, R.Prix_Total
        FROM Billets B
        INNER JOIN Reservations R ON B.Num_Reservation = R.Num_Reservation
        INNER JOIN Ligne_Reservation LR ON B.Num_Reservation = LR.Num_Reservation
        INNER JOIN Vols V1 ON LR.Num_Vol = V1.Num_Vol
        WHERE V1.Ville_Depart = @VilleDepart
        AND V1.Ville_Arrivee = @VilleArrivee
        AND (SELECT COUNT(*) FROM Ligne_Reservation LR2 WHERE LR2.Num_Reservation =
       LR.Num_Reservation) - 1 = @NombreEscales;
        -- Afficher les billets proposés classés par ordre décroissant des prix
        SELECT Num_Billet, Num_Reservation, Prix_Total
        FROM #BilletsProposes
        ORDER BY Prix_Total DESC;
        -- Supprimer la table temporaire
        DROP TABLE #BilletsProposes;
       END;
```

Exercice 28 :

```
       CREATE FUNCTION Complet (@NumVol INT)
       RETURNS BIT
       AS
       BEGIN
        DECLARE @TotalPlaces INT, @PlacesReservees INT, @Retour INT;
        SELECT @TotalPlaces = A.Nbr_Place, @PlacesReservees = COUNT(*)
        FROM Vols V
        INNER JOIN Voyages Vg ON V.Num_Vol = Vg.Num_Vol
       INNER JOIN Avions A ON V.Code_Avion = A.Num_Avion
        WHERE V.Num_Vol = @NumVol
        GROUP BY V.Num_Vol, Nbr_Place;
        IF @PlacesReservees >= @TotalPlaces
```

```
 SET @Retour = 1; -- Voyage complet
 ELSE
 SET @Retour = 0; -- Voyage non complet
Return @Retour;
END;
go
CREATE FUNCTION Occuper (@NumVol INT, @NumPlace INT)
RETURNS BIT -- Spécifiez le type de retour comme BIT pour représenter une valeur
booléenne (1 ou 0)
AS
BEGIN
 DECLARE @PlaceOccupee BIT; -- Déclarez une variable pour stocker le résultat
 IF EXISTS (SELECT 1 FROM Voyages WHERE Num_Vol = @NumVol AND Num_Place = @NumPlace)
 SET @PlaceOccupee = 1; -- Place occupée
 ELSE
 SET @PlaceOccupee = 0; -- Place disponible
 RETURN @PlaceOccupee; -- Retournez la valeur stockée dans la variable
END;
go
CREATE TRIGGER ControleDisponibilitePlace
ON Voyages
INSTEAD OF INSERT
AS
BEGIN
 DECLARE @NumVol INT, @CodePassager INT, @NumPlace INT;
 -- Récupérer les valeurs insérées dans la table Voyages
 SELECT @NumVol = Num_Vol, @CodePassager = Code_Passager, @NumPlace = Num_Place FROM
inserted;
 -- Vérifier si le voyage est complet
 IF dbo.Complet(@NumVol) = 0
 BEGIN
 -- Vérifier si la place est occupée
 IF dbo.Occuper(@NumVol, @NumPlace) = 1
 BEGIN
 -- Trouver un numéro de place disponible automatiquement
 DECLARE @NouvellePlace INT;
 SET @NouvellePlace = 1;
 WHILE dbo.Occuper(@NumVol, @NouvellePlace) = 1
 BEGIN
 SET @NouvellePlace = @NouvellePlace + 1;
 END
 -- Insérer le nouveau voyage avec le numéro de place disponible
 INSERT INTO Voyages (Code_Passager, Num_Vol, Num_Place) VALUES
(@CodePassager, @NumVol, @NouvellePlace);
 PRINT 'Place occupée. Un nouveau voyage a été inséré avec le numéro de place
disponible: ' + CAST(@NouvellePlace AS VARCHAR(10));
 END
 ELSE
 BEGIN
 PRINT 'La place est déjà occupée. Aucune nouvelle insertion nest effectuée.';
 END
 END
END;
```

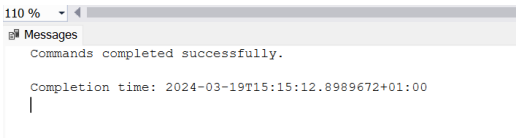
Exercice 29 :

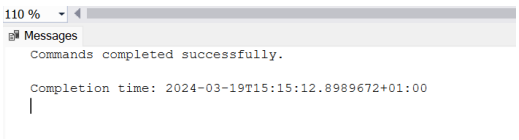
```
CREATE TRIGGER MajusculesEtUnicitePassager
ON Passagers
INSTEAD OF INSERT
AS
BEGIN
 SET NOCOUNT ON;
 -- Insérer les données en majuscules avec contrôle d'unicité
 INSERT INTO Passagers (Code_Passager, Nom_Passager, Pre_Passager, Num_Passport,
Categorie, Num_Tel)
 SELECT
 I.Code_Passager,
 UPPER(I.Nom_Passager),
 UPPER(I.Pre_Passager),
 I.Num_Passport,
```

```
        I.Categorie,
        I.Num_Tel
        FROM inserted AS I
        LEFT JOIN Passagers AS P ON I.Code_Passager = P.Code_Passager
        WHERE P.Code_Passager IS NULL;
        -- Afficher un message si des lignes ont été insérées avec succès
        IF @@ROWCOUNT > 0
        BEGIN
        PRINT 'Insertion des passagers réussie.';
        END
        ELSE
        BEGIN
        PRINT 'Aucune insertion effectuée (clé en double).';
        END
              END;
Exercice 30 :
       CREATE TRIGGER ControleInsertionVoyage
       ON Voyages
       INSTEAD OF INSERT
       AS
       BEGIN
        SET NOCOUNT ON;
        DECLARE @NumPassager INT, @NumBillet INT, @NumVol INT;
        -- Récupérer les valeurs insérées dans la table Voyages
        SELECT @NumPassager = Code_Passager, @NumBillet = Num_Billet, @NumVol = Num_Vol
        FROM inserted;
        -- Vérifier si le passager a réservé le billet pour le vol correspondant
        IF EXISTS (
        SELECT 1
        FROM Billets AS B
        INNER JOIN Ligne_Reservation AS LR ON B.Num_Reservation = LR.Num_Reservation
        WHERE B.Num_Billet = @NumBillet AND LR.Num_Vol = @NumVol
        )
        BEGIN
        -- Insertion autorisée
        INSERT INTO Voyages (Code_Passager, Num_Billet, Num_Vol, Num_Place)
        SELECT Code_Passager, Num_Billet, Num_Vol, Num_Place
        FROM inserted;

        PRINT 'Insertion du voyage autorisée.';
        END
        ELSE
        BEGIN
        -- Insertion non autorisée
        PRINT 'Impossible dinsérer le voyage. Le passager na pas réservé le billet pour
       le vol correspondant.';
        END
       END;
Exercice 31 :
       CREATE TRIGGER MiseAJourPilote
       ON Voyages
       AFTER INSERT
       AS
       BEGIN
        SET NOCOUNT ON;
        DECLARE @NumPilote INT, @NombreTotalAvions INT;
        -- Récupérer le numéro de pilote lié au vol inséré
        SELECT @NumPilote = V.Code_Pilote
        FROM inserted AS I
        INNER JOIN Vols AS V ON I.Num_Vol = V.Num_Vol;
        -- Mettre à jour le nombre d'avions pilotés par le pilote
        UPDATE Pilotes
        SET NbrAvions = (SELECT COUNT(DISTINCT Code_Avion) FROM Vols WHERE Code_Pilote =
       @NumPilote)
        WHERE Num_Pilote = @NumPilote;
        -- Mettre à jour le nombre de voyages du pilote
        UPDATE Pilotes
        SET NbrVoyages = (SELECT COUNT(*) FROM Voyages V INNER JOIN Vols VO on
       V.Num_Vol=VO.Num_Vol WHERE Code_Pilote = @NumPilote)
```

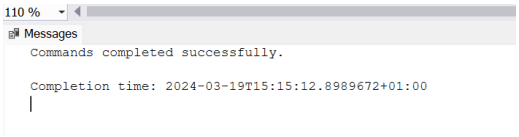
```
        WHERE Num_Pilote = @NumPilote;
        -- Mettre à jour le statut du pilote
       SELECT @NombreTotalAvions = COUNT(*) FROM Avions;
        UPDATE Pilotes
        SET Statut =
        CASE
        WHEN (SELECT COUNT(*) FROM Vols WHERE Code_Pilote = @NumPilote) > 0 THEN
        CASE
        WHEN CAST((SELECT COUNT(*) FROM Vols WHERE Code_Pilote = @NumPilote)
       AS DECIMAL) / ISNULL(CAST(@NombreTotalAvions AS NVARCHAR), '1') > 0.5 THEN 'Expert'
        WHEN CAST((SELECT COUNT(*) FROM Vols WHERE Code_Pilote = @NumPilote)
       AS DECIMAL) / ISNULL(CAST(@NombreTotalAvions AS NVARCHAR), '1') >= 0.05 THEN
       'Qualifie'
        ELSE 'Débiteur'
        END
        ELSE 'Débiteur' -- Si le pilote n'a effectué aucun vol
        END
        WHERE Num_Pilote = @NumPilote;
              END;

Exercice 32 :
       CREATE TRIGGER VerifierCapaciteAvion
       ON Voyages
       INSTEAD OF INSERT
       AS
       BEGIN
        SET NOCOUNT ON;
        DECLARE @NumVol INT, @NumPlace INT, @CapaciteAvion INT;
        SELECT @NumVol = Num_Vol, @NumPlace = Num_Place
        FROM inserted;
        -- Récupérer la capacité de l'avion pour le vol spécifié
        SELECT @CapaciteAvion = A.Nbr_Place
        FROM Vols AS V
        INNER JOIN Avions AS A ON V.Code_Avion = A.Num_Avion
        WHERE V.Num_Vol = @NumVol;
        -- Vérifier si le nombre de places accordées dépasse la capacité de l'avion
        IF (SELECT COUNT(*) FROM Voyages WHERE Num_Vol = @NumVol) >= @CapaciteAvion
        BEGIN
        -- Si le nombre de places accordées dépasse la capacité de l'avion, générer une
       erreur
        RAISERROR ('Le nombre de places accordées dépasse la capacité de l''avion pour
       ce vol.', 16, 1);
        END
        ELSE
        BEGIN
        -- Insertion autorisée
        INSERT INTO Voyages (Code_Passager, Num_Billet, Num_Vol, Num_Place)
        SELECT Code_Passager, Num_Billet, Num_Vol, Num_Place
        FROM inserted;

        PRINT 'Insertion du voyage autorisée.';
        END
              END;
 Exercice 33 :
       CREATE TABLE JournalModifications (
        ID INT IDENTITY(1,1) PRIMARY KEY,
        Action VARCHAR(10),
        Utilisateur NVARCHAR(100),
        Heure DATETIME,
        AnciennesValeurs NVARCHAR(MAX),
        NouvellesValeurs NVARCHAR(MAX)
       );
       go
       CREATE TRIGGER MemoriserModifications
       ON Reservations
       AFTER INSERT, UPDATE, DELETE
       AS
       BEGIN
        SET NOCOUNT ON;
```

```
        DECLARE @Action VARCHAR(10);
        DECLARE @User NVARCHAR(100);
        DECLARE @Time DATETIME;
        SET @Time = GETDATE(); -- Récupérer l'heure actuelle
        SET @User = SYSTEM_USER; -- Récupérer l'utilisateur actuel
        -- Déterminer l'action effectuée (insertion, mise à jour ou suppression)
        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';
        -- Insérer les informations dans la table de journalisation des modifications
        INSERT INTO JournalModifications (Action, Utilisateur, Heure, AnciennesValeurs,
      NouvellesValeurs)
        SELECT @Action, @User, @Time, (SELECT * FROM deleted FOR JSON AUTO), (SELECT * FROM
      inserted FOR JSON AUTO);
            END ;

Exercice 34 /35 :
      CREATE TRIGGER SuppressionEnCascadePassager
      ON Passagers
      INSTEAD OF DELETE
      AS
      BEGIN
        SET NOCOUNT ON;
        -- Supprimer les enregistrements correspondants dans les tables dépendantes
      DELETE FROM Voyages WHERE Num_Billet IN (SELECT Num_Billet FROM deleted);
      DELETE FROM Ligne_Reservation WHERE Num_Reservation IN (SELECT Num_Reservation
      FROM deleted);
      DELETE FROM Billets WHERE Num_Reservation IN (SELECT Num_Reservation FROM
      deleted);
        DELETE FROM Reservations WHERE Code_Passager IN (SELECT Code_Passager FROM deleted);
        DELETE FROM Voyages WHERE Code_Passager IN (SELECT Code_Passager FROM deleted);
        -- Supprimer le passager de la table Passagers
        DELETE FROM Passagers WHERE Code_Passager IN
      (SELECT Code_Passager FROM deleted);
            END;
Exercice 36 :
      CREATE TRIGGER CorrectPhoneNumber
      ON Passagers
      AFTER INSERT, UPDATE
      AS
      BEGIN
        SET NOCOUNT ON;
        -- Mettre à jour les numéros de téléphone avec les corrections nécessaires
        UPDATE Passagers
        SET Passagers.Num_Tel = REPLACE(REPLACE(REPLACE(REPLACE(inserted.Num_Tel, '-', '.'),
      ' ', '.'), 'O', '0'), 'o', '0')
        FROM Passagers
        INNER JOIN inserted ON Passagers.Code_Passager
      = inserted.Code_Passager;
      END;
      go
Exercice 37 :
      CREATE TRIGGER CheckDateValidity
      ON Vols
      AFTER INSERT, UPDATE
      AS
      BEGIN
        SET NOCOUNT ON;
        IF EXISTS (
        SELECT 1
        FROM inserted
        WHERE LEN(ISNULL(Date_Depart, '')) > 10
        OR LEN(ISNULL(Date_Arrivee, '')) > 10
        OR PATINDEX('%[^0-9/]%', ISNULL(Date_Depart, '')) > 0
        )
        BEGIN
```



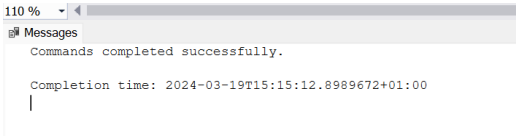


```
        RAISERROR('Invalid date format. Please use only digits, "/", and ensure the
       length is not more than 10 characters.', 16, 1);
        ROLLBACK;
        RETURN;
        END;
        -- Remplacer les caractères 'O' ou 'Q' par '0' dans les dates
        UPDATE Vols
        SET Date_Depart = REPLACE(REPLACE(ISNULL(inserted.Date_Depart, ''), 'O', '0'), 'Q',
       '0')
        FROM Vols
        INNER JOIN inserted ON Vols.Num_Vol =
       inserted.Num_Vol;
       END;
```



Exercice 38 :

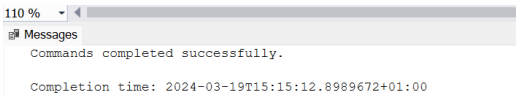
```
       CREATE TABLE VoyageArchive (
        Num_Passager INT,
        Num_Billet INT,
        Num_Vol INT,
        Num_Place INT,
        Date_Archive DATETIME,
        Type_Operation VARCHAR(50) -- Par exemple, "Suppression"
        -- Ajoutez d'autres colonnes d'archive si nécessaire
       );
       go
       CREATE TRIGGER ArchiveDeletedVoyages
       ON Voyages
       AFTER DELETE
       AS
       BEGIN
        SET NOCOUNT ON;
        INSERT INTO VoyageArchive (Num_Passager, Num_Billet, Num_Vol, Num_Place,
       Date_Archive, Type_Operation)
        SELECT
        Code_Passager,
        Num_Billet,
        Num_Vol,
        Num_Place,
        GETDATE(), -- Date actuelle d'archivage
        'Suppression'
        FROM
        deleted;
       END;
```



Exercice 39 :

```
       CREATE TABLE ReservationArchive (
        Num_Reservation INT PRIMARY KEY,
        Date_Archive DATETIME,
        Nature_Traitement VARCHAR(50) -- Annulée ou Validée
        -- Ajoutez d'autres colonnes d'archive si nécessaire
       );
       -- Insérer des lignes aléatoires dans la table ReservationArchive
       INSERT INTO ReservationArchive (Num_Reservation, Date_Archive, Nature_Traitement)
       VALUES
        (1, DATEADD(DAY, -5, GETDATE()), 'Annulée'),
        (2, DATEADD(DAY, -15, GETDATE()), 'Validée'),
        (3, DATEADD(DAY, -2, GETDATE()), 'Annulée'),
        (4, DATEADD(DAY, -8, GETDATE()), 'Validée'),
        (5, DATEADD(DAY, -12, GETDATE()), 'Validée');
       go
       CREATE TRIGGER ArchiveDeletedReservations
       ON Reservations
       AFTER DELETE
       AS
       BEGIN
        SET NOCOUNT ON;
        INSERT INTO ReservationArchive (Num_Reservation, Date_Archive, Nature_Traitement)
        SELECT
        Num_Reservation,
```

```
        GETDATE(), -- Date actuelle d'archivage
        CASE
        WHEN Etat_Reservation = 'Annulée' THEN 'Annulée'
        WHEN Etat_Reservation = 'Validée' AND DATEADD(DAY, 10, Date_Reservation) <
       GETDATE() THEN 'Validée'
        ELSE NULL

        END
        FROM
        deleted;
       END;
```

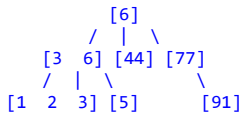


Exercice 40 :

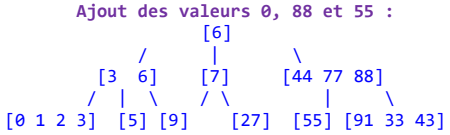
```
       CREATE VIEW ReservationValidees
       AS
       SELECT
        Num_Reservation,
        Date_Reservation,
        Code_Passager,
        Prix_Total
       FROM
        Reservations
       WHERE
        Etat_Reservation = 'Etat77'
              AND Code_Agence = '1';
```

Exercice 41 :

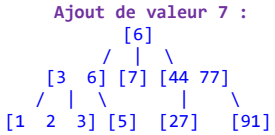
Schéma de l'arbre initial :



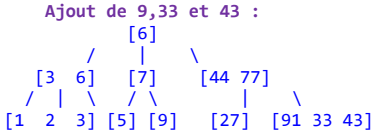
Ajout des valeurs 0, 88 et 55 :



Ajout de valeur 7 :

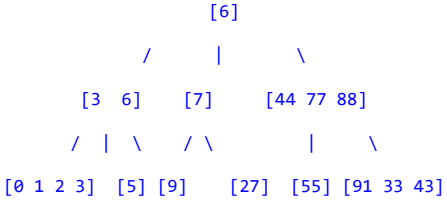


Ajout de 9,33 et 43 :

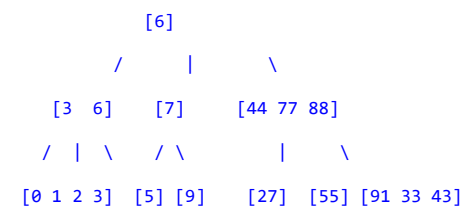


Exercice 42 :

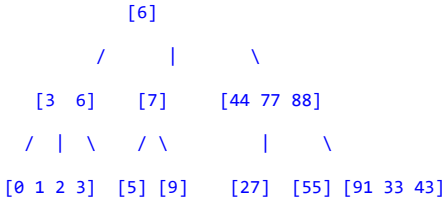
Schéma de l'arbre initial :



Suppression de la valeur 24 :



Suppression de la valeur 25 :



Suppression de la valeur 98 :