Speed Recap 1

En Vrac 1

Auto incrementation : IDENTITY

```
CREATE TABLE Employes
(
        CodeEmp int PRIMARY KEY IDENTITY,   -- auto incrementation
        ...
```

DEFAULT + GETDATE()

```
CREATE TABLE Employes
(
        DateEmbauche datetime NOT NULL DEFAULT GETDATE(),
        ...
```

DROP TABLE, IF EXISTS, FK, PK Composite, NUMERIC type

```
DROP TABLE IF EXISTS Conges;
CREATE TABLE Conges
(
        CodeEmp int REFERENCES Employes(CodeEmp),
        Annee numeric(4,0),
        NbJoursAcquis numeric(2,0),
        CONSTRAINT PK_Conges PRIMARY KEY(CodeEmp, Annee)  -- 1 pk avec 2
elements
);
```

- `col type REFERENCES other_table(other_PK)` : definition de FK
- `numeric(total_chiffre,précision)` : ex. numeric(4,2) est 4 chiffres significatifs et 2 décimales
- `CONSTRAINT PK_table PRIMARY KEY(col1, col2)` : PK composite à 2 éléments, en contrainte de table.

FK en contrainte de table

```
CREATE TABLE Employes (
    ...
    CONSTRAINT FK_CongesMens_Conges
        FOREIGN KEY(CodeEmp, Annee)
        REFERENCES Conges(CodeEmp,Annee)
);
```

Les FONCTIONS de DATE

```
SELECT GETDATE();
SELECT DATEPART(month, GETDATE());   -- isole une partie de la date, ici le mois
SELECT MONTH(GETDATE());
SELECT YEAR(GETDATE());
SELECT GETDATE()+3;
SELECT DATEADD(week, 2, GETDATE());
```

```
SELECT DATEDIFF(DAY, GETDATE(),'24/12/2019'); -- datediff(unit, start, end)
SELECT CONVERT(varchar, GETDATE(), 103);
```

ORDRE D'EXECUTION : from > where > group by > having > select > order by

> **FROM > WHERE > GROUP BY > HAVING > SELECT > ORDER BY**

Alias, titre col, ISNULL(), CONVERT(), UPPER()

+ operation sur colonne (rare)

```
SELECT
    UPPER(nom) NOM_DE_FAMILLE,   -- alias
    prenom 'prenom d''usage',    -- titre colonne v1
  /* ISNULL(dateNaissance, 'NR') : fonctionne pas, doit etre du meme type */
    ISNULL(CONVERT(varchar,dateNaissance,103), 'NR'),
    Salaire,
    Augmentation = Salaire * 1.1   -- titre colonne v2
FROM Employes;
```

Note: Pas besoin de `AS` pour la definition d'alias.

Restriction, col IN ('re', 'no'), LIKE

```
SELECT * FROM Employes
WHERE
    (
        CodeService = 'INFOR'
        OR CodeService = 'DIRGE'
        OR CodeService in ('INFOR', 'DIRGE')
    ) AND nom LIKE 'H%'
```

***Note**: L'indentation est faussée pour les besoins de l'impression.*

LES FONCTIONS D'AGGREGATION DISPONIBLES

```
SELECT COUNT(*) FROM Employes;     -- compte tout ce qui n'est PAS NULL

SELECT
        COUNT(dateNaissance),
        COUNT(*),
        SUM(Salaire),
        AVG(Salaire),
        MAX(Salaire),
        MIN(Salaire)
FROM
        Employes;
```

REGROUPEMENT

```
SELECT CodeService, COUNT(*)
FROM   Employes
GROUP BY
    CodeService;
```

```
=> Output:
  code   | count
  ------+------
  INFOR | 3
  DIRGE | 5
```

Note REGROUPEMENT: si je regroupe, je veux des infos de groupe.

DISTINCT : Supprimer des DOUBLONS d'un résultat (SQL SERVER)

- Je veux afficher que les code service
- Mauvaise pratique : ne jamais utiliser de GROUP BY, mais uniquement du **DISTINCT**
- Car le GROUP BY est lourd a processed pour le moteur

```
SELECT   DISTINCT CodeService    -- syntaxe originale
FROM     Employes;
```

HAVING : RESTRICTION sur REGROUPEMENT

```
SELECT
        CodeService,
        COUNT(*)
FROM
        Employes
GROUP BY
        CodeService
HAVING
        COUNT(*) > 1;
```

Note: HAVING ne connait pas les COUNT du SELECT parce que celui-ci ne sera connu qu'à la fin de la transaction (voir ordre).

> Ordre d'exécution: **FROM > WHERE > GROUP BY > HAVING > SELECT > ORDER BY**

JOINTURE

```
SELECT
        e.Nom,
        s.Libelle titre
FROM
        Employes e
        JOIN Services s ON e.CodeService = s.CodeService
ORDER BY
        e.Nom ASC, titre DESC;
```

=> Affiche nom (qui figure dans une table), et titre correspondant (qui figure dans une autre) le tout grace leur point commun, CodeService

(WHERE ... GROUP BY) vs. (GROUP BY ... HAVING)

> Ordre d'exécution: **FROM > WHERE > GROUP BY > HAVING > SELECT > ORDER BY**

Différence entre `WHERE ... GROUP BY` et `GROUP BY ... HAVING` : - `where` fait des restrictions sur le résultat général - `having`fait des restrictions sur des infos de regroupement, des aggregations.

Equivalent SHOW DATABASE (MySQL) en SQL SERVER

```
SELECT name, database_id, create_date FROM sys.databases;
```

=> noms des bdd présentes et leur date de création.

Equivalent de DESC (alias DESCRIBE en MySQL) en SQL SERVER

```
exec sp_columns maTable
```

C'est bien une requête SQL... Pour voir le schéma d'une table.