|  |  |
| --- | --- |
| **Les Variables** | |
| declare @V1 int  declare @V2 int=10  --------------------------------------------------------  declare @V3 int, @V4 varchar(20) | Déclaration des variables locales |
| set @V1=7  --------------------------------------------------------  select @V2=10, @V3='TEST' | Affectation |
| declare @nbr int  select @nbr=count(\*) from personne **√**  select @nbr=(select count(\*) from personne) **√**  set @nbr=count(\*) from personne **×** |
| select @V1  select @V2,@V3  --------------------------------------------------------  print @V1  print cast(@V1 as varchar) + cast(@V2 as varchar) | Affichage des valeurs  Select & print |
| declare @Personne table  (  id int primary key,  nom varchar(20),  prenom varchar(20)  ) | Variable de type table  \*le même principe des tables (select, insert, delete …) |
| Exec sp\_addtype @TypeName = 'NomType', @PhysType = 'varchar(5)' | Type de données utilisateur |
| declare @n int  begin  set @n=3  print @n  end | Bloc d'instructions  \*on doit sélectionner tous le bloc |
| **Les Structure(IF/ELSE & CASE)** | |
| if( /\*Condition\*/ )  begin  --instruction  end  else  begin  --instruction  end | IF / ELSE |
| select \*,case when prix=1000 then 'Pas Cher'  when prix=2000 then 'moyen'  when prix=3000 then 'cher'  else 'inconnu'  end  as Comment from produit | CASE |
| declare @anyNumber int, @anyString varchar(20)  set @anyNumber=24  set @anyString = CASE @anyNumber  when 10 then 'Dix'  when 24 then 'Vingt Quatre'  when 42 then 'Quarante Deux'  else 'inconnu'  end  print @anyString  --------------------------------------------------------  set @anyString = CASE  when @anyNumber<10 then 'Inferieur a Dix'  when @anyNumber<24 then 'Inferieur a Vingt Quatre'  when @anyNumber<42 then 'Inferieur a Quarante Deux'  else 'Très Grand'  end  print @anyString |
| **Les Boucles** | |
| declare @i int=0  while (@i<=10)  begin  print @i  set @i=@i+1  end | While |
| declare @i int=0  NomEtiquette:  print @i  set @i=@i+1  if (@i<=10) goto NomEtiquette | Etiquette |
| **Les Curseurs** | |
| declare CUR1 X  Forward\_only -->  Scroll <-->  Fast\_forward <--  Read\_only  -------------------  fetch X from  Next  Prior  First  Last | Intorduction video |
| --Declaration du Cerseur  declare CUR1 scroll cursor  --Select  for select empno, empname from employee  --Ouverture du Cerseur  open CUR1  --Declaration des variables  declare @empno int, @empname varchar(20)  fetch first from CUR1 into @empno, @empname  --Récupération  while (@@FETCH\_STATUS=0)  begin  print 'Numero : '+convert(varchar(20),@empno)+' Le Nom : '+ @empname  fetch next from CUR1 into @empno, @empname  end  --Fermer le curseur  close CUR1  --Supprimer le curseur  deallocate CUR1 | Syntaxe générale  (si le curseur est déjà déclaré il suffit d’exécuter (open 🡪 close) |
| --Cursor1  declare Dep scroll cursor  for select deptno,deptname from department  open Dep  declare @deptnoDep int, @deptname varchar(20)  fetch first from Dep into @deptnoDep, @deptname  --Cursor2  declare Emp scroll cursor  for select empname, deptno from employee  open Emp  declare @empname varchar(20),@deptnoEmp int  fetch first from Emp into @empname, @deptnoEmp  while(@@FETCH\_STATUS=0)  begin  print @deptname  while(@@FETCH\_STATUS=0)  begin  if(@deptnoDep=@deptnoEmp)  print @empname  fetch next from Emp into @empname, @deptnoEmp  end  fetch next from Dep into @deptnoDep, @deptname  end | Curseur sous curseur |
| **Les Pocédures** | |
| create proc PC\_Select1  as  select \* from employee  go  exec PC\_Select1 | Procédure stockée sans paramètres |
| create proc PC\_Select2 (@numero int)  as  select \* from employee where employee.empno=@numero  go  exec PC\_Select2 2 | Procédure stockée avec paramètres |
| create proc Select\_P3(@empno int, @empname varchar(20) output)  as  set @empname=(select empname from employee where employee.empno=@empno)  go  declare @nom varchar(20)  exec Select\_P3 1,@nom output  print @nom | Procédure stockée renvoi une valeur  \*output |
| **alter** proc PC\_Select1  as  select \* from departement  go  exec PC\_Select1 | Modifier Procédure stockée |
| **drop** proc PC\_Select1 | Supprimer Procédure stockée |
| **Les Fonctions** | |
| --Création  create function nom\_fonction(@var1 int, @var2 varchar(20))  returns /\* type\_variable\_de\_retour\*/  as  begin  /\*code de la fonction\*/  declare @variable\_a\_retourner  return @variable\_a\_retourner  end  go  --Appelle  declare @n1 int=2, @n2 int=3  select dbo.nom\_fonction(@n1,@n2) | Fonction return une seul valeur |
| --Création  create function nom\_fonction(@var1 int, @var2 varchar(20))  returns table  as  return select \* from stagiaire where nom like @var2 and numero=@var1  go  --Appelle  declare @num int=2  select \* from dbo.nom\_fonction (@num,'Nom1') | Fonction return une table |
| create function Funct\_Q1()  returns table  as  return select top 1 nom\_Enfant from Enfant order by dateNaissance ASC  --Appelle  select \* from dbo.Funct\_Q1() | Fonction sans paramètres |
| **alter** function nom\_fonction(@var1 int, @var2 varchar(20))  returns table  as  return select \* from formateur where nom like @var2 and numero=@var1  go | Modifier fonction |
| **drop** function nom\_fonction | Supprimer fonction |
| **Les Triggers** | |
| alter Trigger Trig\_Q4  on Circonscription for insert  as  begin tran  if((select inserted.NombreSiege from inserted)<0 or (select inserted.NombreSiege from inserted)>6)  begin  print 'ERROR!'  rollback transaction  return  end  else  COMMIT tran | For |
| create Trigger Trigger\_Q2  on Membre after insert  as  begin  declare @v1 varchar(50)  set @v1 = (select login\_membre from inserted)  insert into Role\_Utilisateur values(@v1,'Utilisateur')  end | After |
| create trigger Trigger\_Q5  on Inscrire instead of delete  as  begin  print 'ERROR!'  end | Insead of |
| **Les Exceptions (try-Catch)** | |
|  |  |

cast(id as varchar(10)) convert(varchar(10),id) space(5)

cast(@var as stype) conversion

--------------------

begin try

select 2/0

end try

begin catch

select ERROR\_LINE()

select ERROR\_MESSAGE()

end catch

2éme type de transaction

begin try

instriction …

.

.

.

commit

end try

begin catch

rollback

end catch