بسمه تعالی

جزوه آزمایشگاه پایگاه داده ها

استاد کدخدایی

تهیه و تنظیم :

مجید بصیرتی

شروع کار با Sql Server 2005

اگر در هنگام نصب نرم افزار sql server چند instance را نصب کرده باشیم در هنگام بالا آمدن نرم افزار درقسمت server name نام کامپیوتر به همراه بک اسلش وسپس نام instance را وارد میکنیم.

تحقیق:

انواع data type ها به همراه شرح آنها و موارد استفاده آنها:

انواع داده های صحیح:

Data type	Range	Storage
Bigint	-2^{63} to $2^{63}-1$	8 Byte
Int	-2^{31} to $2^{31}-1$	4 Byte
Smallint	-2^{15} to $2^{15}-1$	2 Byte
Tinyint	0 to 255	1 Byte

Char(n) :

نوع داده کاراکتری با طول ثابت و بدون یونیکد با طول n که n می تواند از ۱ تا ۸۰۰۰ باشد.

Varchar(n) :

نوع داده کاراکتری با طول ثابت

Decimal(p[,s]),Numeric(p[,s]) :

شامل مقادیر اعشاری می باشد. که مقادیر بین $10^{38}+1$- تا $10^{38}-1$ را شامل می شود.

P(precision) : طول عدد هم قسمت صحیح و هم قسمت اعشار که می تواند از 1 تا 38 باشد. مقدار پیش فرض آن 18 می باشد.

S(scale) : طول عدد در قسمت اعشار ، که می تواند از 0 تا p باشد. مقدار پیش فرض آن 0 است.

فضای اشغالی توسط این دو نوع براساس p متغیر است.

Precision	Storage
1-9	5 Byte
10-19	9 Byte
20-28	13 Byte
29-38	17 Byte

Bit :

نوع داده عددی که می تواند مقادیر True ، False و *NULL* را بگیرد.

Money , Smallmoney :

برای مقادیر پولی و مالی استفاده می شود.

Data type	Range	storage
Money	-922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 Byte
Smallmoney	-214,748.3648 to 214,748.3647	4 Byte

نوع داده با تقریب (Approximate numeric) :

این نوع فیلد ها برای نگهداری اعداد غیر صحیح با تعداد ارقام اعشار متغیر و یا تخمینی استفاده می شود. این نوع داده ها رتبه سوم سرعت در بین انواع داده های عددی را دارند و استفاده از آنها به دلیل کندی، توصیه نمی گردد.

دو نوع داده که در این دسته قرار می گیرند Float[(n)] و Real می باشد.

Data type	Range	Storage
Float[(n)]	- 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308	Depends on the value of *n*
Real	- 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38	4 Bytes

میزان حافظه ای که نوع داده ای Float[(n)] اشغال می کند به n بستگی دارد. n می تواند بین ۱ تا ۵۳ باشد.

n value	Precision	Storage size
1-24	7 digit	4 Byte
25-53	15 digit	8 Byte

SQL Server 2005 treats *n* as one of two possible values. If **1**<=n<=**24**, *n* is treated as **24**. If **25**<=n<=**53**, *n* is treated as **53**.

نوع داده ساعت و تاریخ:

Date type	Range	accuracy
Datetime	1753/1/1 to 9999/12/31	3.33 milliseconds
Smalldatetime	1900/1/1 to 2079/6/6	1 min

انواع داده های کاراکتری:

Char(n) : داده کاراکتری با طول ثابت و بدون یونیکد که در آن n می تواند از ۱ تا ۸۰۰۰ باشد. فضای اشغال شده توسط این نوع داده n بایت می باشد. دراین نوع داده اگر n را به طور مثال ۵ درنظر بگیریم و در هنگام ورود داده ۳ کاراکتر وارد کنیم باز هم به اندازه ۵ کاراکتر فضا اشغال می شود.

Varchar(n|max) : فقط در دو مورد با قبلی تفاوت دارد. اول اینکه طول آن متغیر است یعنی اگر ما n را ۸ درنظر بگیریم ولی ۵ کاراکتر وارد کنیم فقط به اندازه همان ۵ کاراکتر فضا اشغال می کند. دومین تفاوت آن این است که علاوه بر n می توان از max استفاده کرد که نشاندهنده حداکثر فضای ذخیره سازی تا 2^{31}-1 می باشد.

فضای اشغال شده توسط این نوع داده برابراست با طول داده وارد شده به علاوه ۲ بایت.

Text : نوعی داده که در آن اطلاعات در خود رکورد ذخیره نمی شود بلکه توسط یک پوینتر به جای دیگری اشاره می کند. طول داده در این نوع داده می تواند تا 2^{31}-1 باشد.

انواع داده های کاراکتری یونیکد:

Nchar(n) : همانند نوع داده char(n) می باشد با این تفاوت که کاراکترها به صورت یونیکد ذخیره می شود. تعداد کاراکترها از ۱ تا ۴۰۰۰ کاراکتر می تواند باشد.

Nvarchar(n|max) : مثل نوع داده varchar بوده با این تفاوت که کاراکترهای یونیکد را نیز ذخیره می کند. حداکثر تا ۴۰۰۰ کاراکتر را می تواند ذخیره کند.

Ntext : همانند نوع داده ای text می باشد که با این تفاوت که کاراکترهای یونیکد را نیز پشتیبانی می کند. این نوع داده می تواند تا 2^{30}-1 کاراکتر را در خود جای دهد. برای متن نامه های فارسی بسیار مناسب می باشد.

نوع داده دودویی رشته ای:

Binary : نوع داده ای با طول ثابت و حداکثر ۸۰۰۰ بایت که برای ذخیره سازی فایل های باینری نظیر عکس و فیلم و ... استفاده می شود.

Varbinary : همانند Binary می باشد با این تفاوت که طول آن متغیر است. این نوع برای ذخیره فایل های ورد ، پاور پوینت و ... مناسب می باشد.

Image : متغیر باینری با طول متغیر

TSQL (Transactional SQL) : کد نویسی در SQL

برای کدنویسی در SQL SERVER از همان محیط کوئری نویسی استفاده می کنیم. برای اجرا کردن یک کد از کلید F5 صفحه کلید استفاده می شود.

وقتی که یک کد را در این محیط می نویسیم می توانیم فقط یک خط از آن را به تنهایی اجرا کنیم. برای این کار باید آن خط را انتخاب کرده و سپس کلید F5 را فشار دهیم.

برای تعریف یک متغیر از دستور Declare استفاده می کنیم که نام متغیرها با علامت @ شروع می شوند. مثال :

```
Declare          @a1  Int
```

برای مقدار دهی به متغیرها از دستور Set استفاده می شود :

```
Set    @a1=14;
```

برای چاپ مقدار یک متغیر از دستور Print استفاده می شود:

```
Print  @a1
```

مثال: قطعه کد زیر سه متغیر از نوع Int تعریف می کند و سپس در هرکدام یک مقداری را قرار می دهد و در نهایت مقدار هرکدام را چاپ می کند.

```
declare @a1 int,@a2 int,@a3 int;
set @a1=20;
set @a2=@a1+10;
set @a3=@a1+@a2;
print @a1;
print @a2;
print @a3;
print 'majid';
```

خروجی :

```
20
30
50
majid
```

یک نمونه خطای سرریز:

```
declare @a1 tinyint;
set @a1=1000;
```

خروجی:

```
Msg 220, Level 16, State 2, Line 2
Arithmetic overflow error for data type tinyint, value = 1000.
```

دستورات شرطی :

۱) دستور IF

```
If      Conditional
        Block  A
[Else
        Block  B]
```

مثال :

```
Declare @x Int , @y Int
Set @x= 200
Set @y= 300 - @x
IF @x > @y
Set @y = @x + 2
ELSE Begin
Set @y = @x + 3
Set @x = @x + 1
END
```

نکته : اگر به جای دستور Print از دستور Select استفاده کنیم خروجی به صورت جدول خواهیم داشت :

```
declare @a1 tinyint;
set @a1=100;
select @a1;
```

۲) دستور Case :

```
Case   value
When value1 then    output1
When value2 then    output2
.
.
else    output n

end
```

نکته : دستور Case همیشه یک خروجی دارد ، یعنی این دستور به تنهایی نمی تواند استفاده شود وباید در دستورات انتساب یا چاپ قرار بگیرد.

صورت بعدی دستور Case :

```
Case
        When condition1    then    output1
        When condition1    then    output2
        .
        .
        else    output n

end
```

مثال : خروجی دستور زیر چیست؟

```
declare @a1 int,@a2 int,@a3 int;
set @a1=4;
print case
when @a1 between 3 and 10 then 5
when @a1 between 2 and 6 then 6
end
```

خروجی :

```
5
```

نکته ای که ازاین مثال درمی یابیم این است که اگر شرط اول برقرار باشد دیگر شروط بعدی اجرا نخواهد شد.

نکته: برای کامنت کردن یک خط از کد از دو خط تیره در اول خط دستور استفاده می کنیم.

حلقه تکرار While :

```
While condition
begin
        ------
        ------
        continue        ادامه دستورات را رهاکرده و به اول حلقه بازمی گردد
        break           ادامه دستورات را رها کرده و از حلقه خارج می شود

end
```

تمرین سرکلاس: برنامه ای که بیست جمله اول سری فیبوناچی را چاپ کند.

```
declare @a int,@b int,@c int,@n int
set @n=18
set @a=0
set @b=1;
print @a;
print @b;
while @n>0
begin
```

```
set @c=@a+@b;
print @c;
set @a=@b;
set @b=@c;
set @n=@n-1
end
```

تمرین سرکلاس : برنامه ای که عدد بیست فاکتوریل را محاسبه کند.

```
declare @a bigint,@b bigint
set @a=1
set @b=1
while @a<=20
begin
set @b=@b*@a
set @a=@a+1
end
print @b
```

تمرین سرکلاس : برنامه ای که بزرگترین مقسوم علیه مشترک دوعدد را حساب کند.

```
declare @a int,@b int, @r int
set @a=120
set @b=18
while @b>0
begin
set @r=@a%@b
set @a=@b
set @b=@r
end
print @a
```

دستور Insert کردن در یک جدول :

```
Insert [into] table1(field_name) values (مقادیر)
```

تمرین سرکلاس : فرض کنیم جدولی با سه فیلد a و b و c داریم. می خواهیم اعداد از ۱ تا ۵۰۰۰ را در فیلد a وارد کنیم.

دستور آن را بنویسید.

```
declare @a int
set @a=1
while @a<=5000
begin
insert into t1(a) values (@a)
set @a=@a+1
end
select * from t1
```

تابع Rand() :

این تابع همان طور که ازاسمش پیداس برای تولید یک عدد تصادفی استفاده می شود. این تابع یک عدد تصادفی بین ۰ تا ۱ ایجاد می کند.

تمرین منزل : جدولی داریم با دو فیلد a و b که فیلد a کلید می باشد. می خواهیم با استفاده از تابع rand ، فیلد b را با هزار عدد تصادفی پر نماییم(۱۰۰۰ رکورد) و سپس تعداد رکوردهایی که عدد تصادفی تولید شده در آنها کوچکتر و بزرگتر از 0.5 می باشد را به دست آوریم. دستورات لازم برای این کار را بنویسید.

```
declare @a real,@n int,@c int
set @n=1000
delete from t1
set @c=0
while @n>0
begin
set @a=rand()
insert into t1(b) values(@a)
set @n=@n-1
end
select count(b) as 'تعداد اعداد کوچکتر از ۰.۵' from t1 where b<0.5
select count(b) as 'تعداد اعداد بزرگتر از ۰.۵' from t1 where b>=0.5
select * from t1
```

انواع توابع در Sql Server 2005

۱. تابع Left :

```
Left ( رشته مورد نظر , تعداد کاراکترها )
```

این تابع کاراکترهای رشته را از طرف چپ به تعداد عدد داخل پرانتز برمی گرداند.

مثال:

```
Print  Left  ('university',3)
```

خروجی :

```
uni
```

۲. تابع Right :

```
Right ( رشته مورد نظر , تعداد کاراکترها )
```

این تابع کاراکترهای رشته را از طرف راست به تعداد عدد داخل پرانتز برمی گرداند.

مثال :

```
Print Right ('university',3)
```

خروجی :

```
ity
```

۳. تابع Substring :

```
Substring ( رشته مورد نظر , اندیس شروع , تعداد کاراکترها )
```

این تابع برای خروج یک زیررشته از یک رشته به کار می رود.

نکته : اندیس رشته ها در Sql Server 2005 از ۱ شروع می شود.

مثال :

```
Print substring ('university', 4 , 3)
```

خروجی :

```
ver
```

مثال :

```
Print substring ('university', 0 , 3 )
```

خروجی :

```
un
```

مثال : کد دستور زیر چه چیزی را درخروجی چاپ می کند.

```
print left('salam',2) + substring ('hello',4,1) + right('payam',2)
```

خروجی :

```
salam
```

۴. تابع LTrim :

```
LTrim ( رشته مورد نظر )
```

تمام فضاهای خالی که در سمت چپ رشته است را حذف می کند.

مثال :

```
Print ltrim ( '          hello')
```

خروجی :

```
hello
```

۵. تابع RTrim :

```
RTrim ( رشته مورد نظر )
```

تمام فضاهای خالی که درسمت راست رشته است را حذف می کند.

مثال :

```
Print rtrim ( 'hello      ')
```

خروجی :

```
hello
```

مثال : کد زیر فضاهای خالی که دراول و آخر رشته باشد را حذف می کند.

Rtrim (ltrim (string)) or ltrim (rtrim (string))

۶. تابع Upper :

```
Upper ( رشته مورد نظر )
```

یک رشته را دریافت می کند و تمام کاراکترهای آن را بزرگ می کند.

مثال :

```
Print upper ( 'hello' )
```

خروجی :

```
HELLO
```

۷. تابع Lower : یک رشته را دریافت می کند و تمام کاراکترهای آن را کوچک می کند.

۸. تابع Len : طول یک رشته را برمی گرداند. این تابع رشته را از سمت راست trim می کند.

```
Len ( رشته مورد نظر )
```

۹. تابع Char : یک عدد را دریافت می کند و معادل کاراکتری آن را برمی گرداند.

```
Char ( کد کاراکتر )
```

مثال :

```
print char (65)
```

خروجی :

```
A
```

تمرین سر کلاس : برنامه ای بنویسید که هر ۲۵۵ کاراکتر را به همراه کد اسکی آن چاپ کند.

```
declare @n int
set @n=0
while @n<=255
     begin
          print cast(@n as varchar)+'='+ char (@n)
          set @n=@n+1
     end
```

نکته : تابع **Cast** برای تبدیل یک نوع به نوع دیگر به کار می رود. در این مثال متغیر n را که ازنوع int است به نوع varchar تبدیل کرده است.

۱۰. تابع Ascii : این تابع کد اسکی یک کاراکتر را برمی گرداند.

```
Ascii  ( کاراکتر مورد نظر )
```

تمرین سرکلاس : برنامه ای که یک رشته دریافت کند و تعداد کاراکترهای کوچک و بزرگ و اعداد آن را بشمارد.

```
declare @n int,@c char,@st varchar(50),@l int,@u int,@num int
set @n=1
set @l=0
set @u=0
set @num=0
set @st='This Is A teSt 123'
set @n=len(@st)
while (@n>0)
begin
     set @c=substring(@st,@n,1)
     if ascii(@c) between 65 and 96
          set @u=@u+1
     if ascii(@c) between 97 and 122
          set @l=@l+1
     if @c>='0' and @c<='9'
          set @num=@num+1
     set @n=@n-1
end
print 'تعداد کاراکترهای بزرگ '
print @u
print 'تعداد کاراکترهای کوچک'
print @l
print 'تعداد اعداد'
print @num
```

تمرین سرکلاس : برنامه ای که یک رشته را دریافت کند و آن را معکوس کند.

```
declare @st1 varchar(50),@st2 varchar(50),@n int,@char char
set @st2=''
set @st1='my name is majid'
set @n=len(@st1)
while(@n>0)
begin
     set @char=substring(@st1,@n,1)
     set @st2=@st2+@char
     set @n=@n-1
end
print @st1
print @st2
```

سوال : دراین مثال اگر متغیر st2 را ازنوع char(50) در نظر بگیریم جواب نمی گیریم . چرا؟

۱۱. تابع Reverse : این تابع یک رشته را معکوس می کند.

۱۲. تابع Replace : یک رشته را با رشته دیگر جایگزین می کند.

Replace (رشته اصلی , زیررشته ای که باید عوض شود , رشته ای که باید جایگزین زیررشته شود)

مثال :

```
print replace ( 'hello','lo','ci')
```

خروجی :

```
helci
```

۱۳. تابع Space : به تعداد عدد داخل پرانتز space چاپ می کند.

Space (تعداد فضاهای خالی)

تمرین منزل : برنامه ای که یک رشته بگیرد و از ده حرف اول انگلیسی سه تایی را که بیشتر ازهمه تکرار شده را چاپ کند.

```
declare @st varchar(50),@n int,@l int,@char char,@a int,@b int,@c
int,@d int,@e int,@f int,@g int,@h int,@i int,@j int,@c1 char,@c2
char
declare @max1 int,@max2 int,@max3 int
set @n=1
set @a=0  set @b=0  set @c=0  set @d=0  set @e=0  set @f=0  set
@g=0 set @h=0  set @i=0  set @j=0
set @st='he is a student'
while (@n<=len(@st))
     begin
          set @char=substring(@st,@n,1)
          if(@char='a')
               set @a=@a+1
          if(@char='b')
               set @b=@b+1
          if(@char='c')
               set @c=@c+1
          if(@char='d')
               set @d=@d+1
          if(@char='e')
               set @e=@e+1
          if(@char='f')
               set @f=@f+1
          if(@char='g')
               set @g=@g+1
          if(@char='h')
```

```
                set @h=@h+1
           if(@char='i')
                set @i=@i+1
           if(@char='j')
                set @j=@j+1

           set @n=@n+1
      end
set @max1=0
set @max2=0
set @max3=0
      if(@max1<@a)
           begin
                set @max1=@a
                set @c1='a'
           end
      if(@max1<@b)
           begin
                set @max1=@b
                set @c1='b'
           end
      if(@max1<@c)
           begin
                set @max1=@c
                set @c1='c'
           end
      if(@max1<@d)
           begin
                set @max1=@d
                set @c1='d'
           end
      if(@max1<@e)
           begin
                set @max1=@e
                set @c1='e'
           end
      if(@max1<@f)
           begin
                set @max1=@f
                set @c1='f'
           end
      if(@max1<@g)
           begin
                set @max1=@g
                set @c1='g'
           end
      if(@max1<@h)
           begin
                set @max1=@h
                set @c1='h'
           end
      if(@max1<@i)
           begin
```

```
                set @max1=@i
                set @c1='i'
          end
     if(@max1<@j)
          begin
                set @max1=@j
                set @c1='j'
          end
-----------------------------------------------
     if(@max2<@a and @c1<>'a')
          begin
                set @max2=@a
                set @c2='a'
          end
     if(@max2<@b and @c1<>'b')
          begin
                set @max2=@b
                set @c2='b'
          end
     if(@max2<@c and @c1<>'c')
          begin
                set @max2=@c
                set @c2='c'
          end
     if(@max2<@d and @c1<>'d')
          begin
                set @max2=@d
                set @c2='d'
          end
     if(@max2<@e and @c1<>'e')
          begin
                set @max2=@e
                set @c2='e'
          end
     if(@max2<@f and @c1<>'f')
          begin
                set @max2=@f
                set @c2='f'
          end
     if(@max2<@g and @c1<>'g')
          begin
                set @max2=@g
                set @c2='g'
          end
     if(@max2<@h and @c1<>'h')
          begin
                set @max2=@h
                set @c2='h'
          end
     if(@max2<@i and @c1<>'i')
          begin
                set @max2=@i
                set @c2='i'
```

```
          end
     if(@max2<@j and @c1<>'j')
          begin
               set @max2=@j
               set @c2='j'
          end
-----------------------------------------------------
     if(@max3<@a and @c1<>'a' and @c2<>'a')
          set @max3=@a
     if(@max3<@b and @c1<>'b' and @c2<>'b')
          set @max3=@b
     if(@max3<@c and @c1<>'c' and @c2<>'c')
          set @max3=@c
     if(@max3<@d and @c1<>'d' and @c2<>'d')
          set @max3=@d
     if(@max3<@e and @c1<>'e' and @c2<>'e')
          set @max3=@e
     if(@max3<@f and @c1<>'f' and @c2<>'f')
          set @max3=@f
     if(@max3<@g and @c1<>'g' and @c2<>'g')
          set @max3=@g
     if(@max3<@h and @c1<>'h' and @c2<>'h')
          set @max3=@h
     if(@max3<@i and @c1<>'i' and @c2<>'i')
          set @max3=@i
     if(@max3<@j and @c1<>'j' and @c2<>'j')
          set @max3=@j
-----------------------------------------------
print @max1
print @max2
print @max3
```

ادامه بحث توابع در Sql Server 2005

۱. توابع User_name( ) و Current_user :

```
User_name( )

Current_user
```

نام کاربری که به Sql Server متصل است را برمیگرداند. به جای این تابع می توان از current_user استفاده کرد.

مثال :

```
Print     user_name( )
Print     Current_user
```

خروجی :

```
dbo
dbo
```

Dbo مخفف DataBase Owner می باشد.

چون یوزری که ما از آن استفاده می کنیم Admin است این خروجی را داریم ، اگر با یک یوزر جدید کار کنیم اسم همان یوزر چاپ می شود.

۲. تابع Abs :

```
Abs   (      عدد     )
```

قدر مطلق یک عدد را برمیگرداند. ورودی آن می تواند عدد اعشاری باشد.

۳. تابع Floor :

```
Floor  (      عدد     )
```

کوچکترین عدد صحیح نزدیک به عدد را برمیگرداند.

۴. تابع Power :

```
Power      (      عدد     ,      توان     )
```

برای به توان رساندن استفاده می شود. یعنی خروجی به این صورت است : $عدد^{توان}$

نکته : نمی توان یک عدد منفی را به توان یک عدد اعشاری رساند.

۵. تابع Sqrt :

```
Sqrt   (        عدد        )
```

ریشه دوم عدد را برمیگرداند. عدد منفی را قبول نمی کند.

۶. تابع Isnull :

```
Isnull  (        عبارت   ,        مقدار پیش فرض  )
```

اگر عبارت ، مخالف null باشد خود عبارت را برمیگرداند و اگر null باشد مقدار پیش فرض را برمیگرداند.

مثال :

```
declare @x int
print isnull(@x,8)
```

خروجی :

```
8
```

یکی از کاربردهای این تابع در خروجی گرفتن از جداول با استفاده از کوئری هاست. اگر یکی از فیلدهای جدول null باشد می توان با این تابع یک مقدار دلخواه را وارد آن فیلد کرد، مثال:

```
select St#, StName, StFamily, isnull(StFatherName,' مقدار وارد نشده است')
from stt
```

در این مثال رکوردهایی از جدول که درآن فیلد StFatherName ، خالی یا Null باشد عبارت "مقدار وارد نشده است" قرار می گیرد.

کاربرد تابع Cast :

می خواهیم با استفاده از تابع Isnull رکوردهایی از جدول که در آن فیلد Grade خالی است را بایک عبارت پر کنیم. همان طور که می دانیم این فیلد از نوع Float است پس اگر بخواهیم عبارت رشته ای درون آن قرار دهیم با خطا مواجه می شویم. پس باید یک تبدیل نوع انجام گیرد و بهتراست که این تبدیل نوع از نوع Float به Char انجام شود. پس :

```
select St#, C#, Term#, isnull(cast(Grade as varchar(50)),' نمره وارد نشده است')
```

```
from reg
```

دراین حالت در خروجی ودر عنوان ستون Grade عبارت No column name نوشته شده است که برای اینکه عنوان ستون را به دلخواه تعیین کنیم از کد زیر استفاده می کنیم :

```
select St#, C#, Term#, isnull(cast(Grade as varchar(50)),' نمره وارد نشده است') as 'grade'
from reg
```

۷. تابع Getdate( ) :

این تابع تاریخ و ساعت Server را برمی گرداند. خروجی این تابع از نوع DateTime می باشد.

مثال :

```
Print getdate()
```

خروجی :

```
Oct 29 2010  7:34PM
```

۸. تابع Convert :

Convert (نوع مورد نظر , عبارت [, style])

این تابع همانند تابع Cast برای تبدیل نوع به کار می رود.

مثال :

```
print convert(varchar,getdate())
```

خروجی

```
Oct 29 2010  7:37PM
```

مثال :

```
print convert(varchar,getdate(),110)
```

خروجی

```
10-29-2010
```

۹. تابع Datepart :

برای جدا کردن قسمتهای مختلف تاریخ و ساعت استفاده می شود.

مثال :

```
print datepart(day,getdate())
```

خروجی :

```
29
```

۱۰. تابع CharIndex :

```
CharIndex ( رشته مورد نظر , رشته اصلی [ , شروع جست وجو ] )
```

این تابع برای جست وجوی یک رشته (رشته مورد نظر) در یک رشته دیگر (رشته اصلی) به کار می رود. هم چنین می توان اندیس شروع جست و جو را تعیین کرد. درصورتی که رشته مورد نظر در رشته اصلی یافت نشود تابع عدد صفر را بر می گرداند.

مثال :

```
print charindex('lo','hello')
```

خروجی :

```
4
```

ایجاد تابع :

```
Create Function نام تابع ( نام پارامتر [as] نوع پارامتر [ = مقدار پیش فرض ] , ... )
        Returns نوع خروجی
[as]
Begin

End
        Return مقدار برگشتی
end
```

نکته : برای فراخوانی یک تابع باید از کلمه dbo قبل از نام تابع استفاده کنیم.

مثال :

```
create function addnum(@n1 int,@n2 int)
     returns int
as
begin
     return @n1+@n2
end

print dbo.addnum(3,5)
```

خروجی :

```
8
```

نکته : برای مشاهده لیست توابعی که ما با دستور create table ایجاد کرده ایم باید از پنجره object explorer مسیر زیر را دنبال کنیم :

Databases → نام دیتابیس → Programmability → Functions → Scalar-valued functions

تمرین سرکلاس : تابعی که دوعدد را در هم ضرب کند.

```
create function multiply(@n1 int,@n2 int)
     returns int
as
begin
     return @n1*@n2
end


print dbo.multiply(7,8)
```

خروجی :

```
56
```

این برنامه را اگر چندبار اجرا کنیم Error می دهد. زیرا وقتی یک بار یک تابع ایجاد شد دوباره اجازه ایجاد آن را نداریم. اگر مجبور شدیم در بدنه تابع تغییری ایجاد کنیم و بعد بخواهیم این تغییرات را به تابع ذخیره شده اعمال کنیم باید به جای کلمه کلیدی Create که در ابتدای تابع استفاده کرده ایم از کلمه Alter استفاده کنیم.

نکته بعد اینکه درهنگام تعریف تابع حتما باید خط Print را حذف کنیم و نیز برای اجرای دستور Print باید فقط همین دستور را اجرا کنیم که برای این کار هم باید فقط خط کد Print را انتخاب کرده و بعد کلید F5 را بزنیم.

تمرین منزل : تابعی بنویسید که کوچکترین مضرب مشترک دو عدد را بیابد.

روش اول :

```
alter function kmm(@a int,@b int)
     returns int
as
```

```
begin
     declare @temp int
     if (@a<@b)
     begin
          set @temp=@a
          set @a=@b
          set @b=@temp
     end
     declare @c int
     set @c=1
     while (@c<=@b)
     begin
          if ((@a*@c)%@b=0)
               break;
          set @c=@c+1
     end
     return @a*@c
end

print dbo.kmm(16,6)
```

روش دوم :

```
alter function kmm(@a int,@b int)
     returns int
as
begin
     declare @r int,@a1 int,@b1 int
     set @a1=@a
     set @b1=@b
     while @b>0
     begin
          set @r=@a%@b
          set @a=@b
          set @b=@r
     end
     return @a1*@b1/@a

end


print dbo.kmm(16,6)
```

تمرین : برنامه فاکتوریل به صورت بازگشتی

```
alter function fact(@n int)
     returns int
as
begin
     declare @a int
     if (@n=0)
```

```
            return 1
      else
            set @a=@n*dbo.fact(@n-1)
            return @a
end


print dbo.fact(7)
```

تمرین سرکلاس : تابعی بنویسید که دویا سه عدد صحیح گرفته و آنها را باهم جمع کرده و نتیجه را برگرداند. اگر درهنگام فراخوانی آرگومان سوم ارسال نشد مقدار آن را ۲۰ فرض کند.(اسم تابع ، sum1)

```
create function sum1(@a int,@b int,@c int =20)
     returns int
as
begin
     return @a+@b+@c

end
print dbo.sum1(5,2,default)
```

خروجی :

```
27
```

برنامه قبلی با ورودی متفاوت :

```
create function sum1(@a int,@b int,@c int =20)
     returns int
as
begin
     return @a+@b+@c

end
print dbo.sum1(5,2,10)
```

خروجی :

```
17
```

تمرین سر کلاس : تابعی بنویسید که شماره دانشجویی را بگیرد و نام دانشجو را برگرداند.

```
alter function findname(@a int)
     returns varchar(50)
as
begin
     return (select stname from stt where st#=@a)

end
print dbo.findname(100)
```

خروجی :

```
ali
```

راه دوم:

```
alter function findname(@a int)
```

```
     returns varchar(50)
as
begin
     declare @stname varchar(50)
     select @stname=stname from stt where st#=@a
     return @stname

end
print dbo.findname(100)
```

ایجاد پروسیجر :

```
Create  proc [edure]  نام پروسیجر  (     نام پارامتر     ,     نوع پارامتر  [  =  مقدار اولیه ] , ... )

As

[begin]

     .
     .
     .

[end]
```

نکته ۱ : درهنگام فراخوانی پروسیجر نیاز به کلمه dbo نداریم.

نکته ۲ : درهنگام فراخوانی پروسیجر نیاز به پرانتز برای آرگومانها نداریم.

تمرین سرکلاس : پروسیجری که دو عدد را جمع می کند.

```
create proc add1(@x int,@y int)
as
--begin
     print @x+@y
--end

add1 3,7
```

خروجی :

```
10
```

برنامه بالا رو طوری تغییر می دهیم که اگر یکی از آرگومانها وارد نشد در اجرای برنامه خللی وارد نشود و خود برنامه برای آن مقدار پیش فرض در نظر بگیرد.

```
alter proc add1(@x int,@y int=20)
as
```

```
--begin
     print @x+@y
--end

add1 3
```

خروجی :

```
23
```

تراکنش (Transaction):

```
Begin  tran [ saction ]     [ نام تراکنش ]



[ commit | rollback ]     { tran[saction] | work }     [ نام تراکنش ]
```

اگر از commit استفاده کنیم دستورات همه اجرا می شود و اگر rollback استفاده کنیم هیچکدام اجرا نمی شود.

مثال :

استفاده از commit و rollback های تو در تو

```
select * from stt
begin tran s2
begin tran s1
update stt set stname='qqqqqqq' where st#=100
update stt set stname='eeeeeee' where st#=101
commit tran s1
rollback tran s2
select * from stt
```

برنامه بالا کلا تغییرات رو rollback میکند.

```
select * from stt
begin tran s2
begin tran s1
update stt set stname='qqqqqqq' where st#=100
update stt set stname='eeeeeee' where st#=101
rollback tran s1
commit tran s2
select * from stt
```

برنامه فوق ارور می دهد ولی تغییرات هم اعمال می شود. ارور آن به شکل زیر است :

```
(3 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)
Msg 6401, Level 16, State 1, Line 6
Cannot roll back s1. No transaction or savepoint of that name was
found.

(3 row(s) affected)
```

تمرین منزل : ۴ تا تراکنش تو درتو بنویسید و توضیح دهید چه رابطه ای بین کامیت ها و رل بک ها وجود دارد؟

```
rollback tran S1
commit tran S2
commit tran S3
```

این کد خطای همراه با خروجی تغییریافته داشت

```
rollback tran S1
commit tran S2
```

خطالی همراه با خروجی تغییریافته

```
rollback tran S1
rollback tran s3
commit tran S2
```

خطای همراه با خروجی تغییر یافته

برنامه زیر هم جواب می دهد در حالی که نام تراکنش ها جا به جا نوشته شده ، چرا؟

```
select * from stt
begin tran s1
begin tran s2
update stt set stname='333333' where st#=100
update stt set stname='bbbbbbb' where st#=101
commit tran s1
rollback tran s2
select * from stt
```

دستور Goto :

Goto	محل پرش

مثال :

```
declare @a int
set @a=1
If   (@a>0)
     Goto end1
Print 'ok'
End1:
Print 'no'
```

خروجی :

```
no
```

دستور Select :

Select	نام فیلد یا فیلدها	from	نام جدول
Where	شرط		

دستوری که کلیه سطرها و ستونهای جدول Stt را به ما نشان می دهد:

```
Select * from stt
```

Results | Messages

	St#	StName	StFamily	StFatherName	roomate
1	100	mmmmmmm	alian	ahmad	104
2	101	44444444	madihi	ali	103
3	102	majid	klsajd	NULL	100
4	103	ali	ahmadi	NULL	102
5	104	majid	basirati	ali	104
6	105	hasan	ahmadzade	reza	107
7	106	sadegh	rahmani	mashal	106
8	107	ahmad	rezaee	hosein	102
9	108	ali	majid	akjsd	103

نکته : علامت ستاره یعنی همه فیلدهای جدول

دستور زیر ستون های st# و stfamily از جدول Stt را به ما نشان می دهد:

```
Select st#,stfamily from stt
```

Results | Messages

	st#	stfamily
1	100	alian
2	101	madihi
3	102	klsajd
4	103	ahmadi
5	104	basirati
6	105	ahmadzade
7	106	rahmani
8	107	rezaee
9	108	majid

دستور زیر یک جدول به تعداد سطرهای جدول stt به ما می دهد که دارای دوستون است که هدر هردوستون no column name نوشته شده است و در تمام سطرهای ستون اول کلمه st# و درتمام سطرهای ستون دوم کلمه stfamily نوشته شده است.

```
Select 'st#','stfamily' from stt
```

Results | Messages

	(No column name)	(No column name)
1	st#	stfamily
2	st#	stfamily
3	st#	stfamily
4	st#	stfamily
5	st#	stfamily
6	st#	stfamily
7	st#	stfamily
8	st#	stfamily
9	st#	stfamily

عملکرد دستور زیر نیز همانند دستور قبل است با این تفاوت که محتوای سطرها فرق کرده است ، سوال پیش می آید که پس این قطعه کد چه استفاده ای از جدول stt می کند؟ درجواب باید گفت که تنها استفاده آن این است که تعداد سطرهای جدول خروجی با جدول stt یکسان خواهد بود.

```
Select 1,2 from stt      or      select '1','2' from stt
```

Results | Messages

	(No column name)	(No column name)
1	1	2
2	1	2
3	1	2
4	1	2
5	1	2
6	1	2
7	1	2
8	1	2
9	1	2

کد زیر یک جدول با ۱۰ ستون و با هدرهای بی نام (no column name) ایجاد می کند که محتوای هر سطر برابر مقادیری است که در قسمت select وارد شده است.

```
select 1,2,3,4,5,6,7,8,9,10 from stt
```

Results | Messages

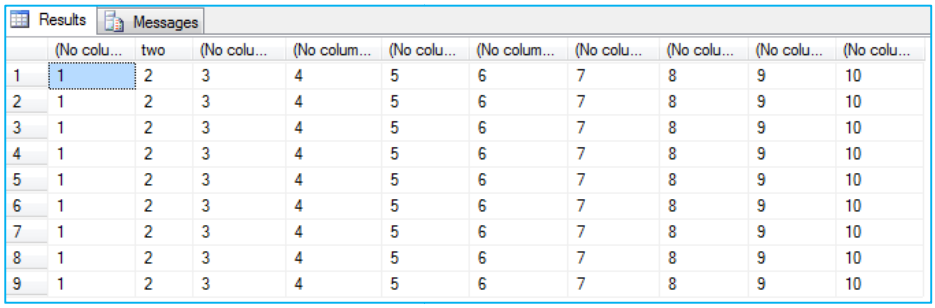
	(No colu...	(No colu...	(No colum...	(No colum...	(No colu...	(No colum...	(No colum...	(No colu...	(No colu...	(No colu...
1	1	2	3	4	5	6	7	8	9	10
2	1	2	3	4	5	6	7	8	9	10
3	1	2	3	4	5	6	7	8	9	10
4	1	2	3	4	5	6	7	8	9	10
5	1	2	3	4	5	6	7	8	9	10
6	1	2	3	4	5	6	7	8	9	10
7	1	2	3	4	5	6	7	8	9	10
8	1	2	3	4	5	6	7	8	9	10
9	1	2	3	4	5	6	7	8	9	10

کد زیر دارای خطا می باشد. چرا؟

```
select 1,2  two,3,4,5,6,7,8,9,10 from (select * from stt )
```

خطای کد بالا به صورت زیر برطرف می شود:

```
select 1,2  two,3,4,5,6,7,8,9,10 from (select * from stt ) d
```

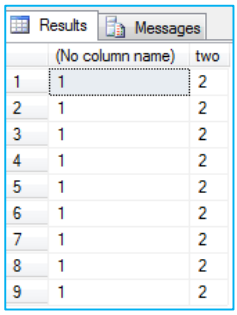


	(No colu...	two	(No colu...	(No colum...	(No colu...	(No colum...	(No colu...	(No colu...	(No colu...	(No colu...
1	1	2	3	4	5	6	7	8	9	10
2	1	2	3	4	5	6	7	8	9	10
3	1	2	3	4	5	6	7	8	9	10
4	1	2	3	4	5	6	7	8	9	10
5	1	2	3	4	5	6	7	8	9	10
6	1	2	3	4	5	6	7	8	9	10
7	1	2	3	4	5	6	7	8	9	10
8	1	2	3	4	5	6	7	8	9	10
9	1	2	3	4	5	6	7	8	9	10

select دوم خروجی جدول دارد که باید یک اسم به آن اختصاص داد که حرف d برای این کار است.

سه قطعه کد زیر باهم معادلند و تنها نکته ای که دارند این است که جدولی که درخروجی داریم در ستون دوم دارای هدر two می باشد:

```
Select 1,2 as 'two' from stt
Select 1,2 'two' from stt
Select 1,2 two from stt
```



	(No column name)	two
1	1	2
2	1	2
3	1	2
4	1	2
5	1	2
6	1	2
7	1	2
8	1	2
9	1	2

گذاشتن عدد در جلو select به جای ستاره یا نام فیلدها سرعت بالاتری دارد ، برای مثال برای اینکه بفهمیم چند رکورد در جدول داریم باگذاشتن عدد می توان به این جواب رسید و بهتراست که از ستاره استفاده نکنیم.

نکته : تاجایی که می توان باید از علامت ستاره پرهیزکرد.

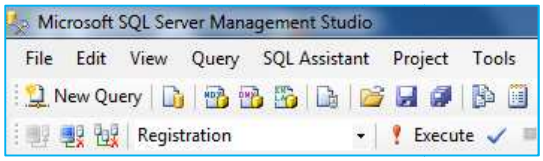
دستور use :

Use نام دیتابیس

این دستور سبب می شود که در لیست Available Database ، نام دیتابیسی که ما نوشته ایم انتخاب شود.

مثال :

```
USE Registration
```



دستور select top :

```
Select  top  ( عدد )  نام فیلدها  where  …
```

این دستور به تعداد عدد داخل پرانتز رکورد را از بالای جدول برمی گرداند.

برای مثال قطعه کد زیر ۲ رکورد بالای جدول stt را برمی گرداند.گذاشتن پرانتز اختیاری است.

```
select top 2 * from stt
```

Results | Messages

	St#	StName	StFamily	StFatherName	roomate
1	100	mmmmmm	alian	ahmad	104
2	101	44444444	madihi	ali	103

نکته : اگر بخواهیم تعداد رکوردهای پایین جدول را برگردانیم از این دستور استفاده می کنیم:

```
select top 2 * from stt
order by st# desc
```

Results | Messages

	St#	StName	StFamily	StFatherName	roomate
1	108	ali	majid	akjsd	103
2	107	ahmad	rezaee	hosein	102

این دستور ابتدا جدول رابه صورت نزولی مرتب می کند وسپس ۲ رکورد بالای جدول که درواقع همان دورکورد آخری می باشد را برمی گرداند.

تحقیق : نحوه استفاده از دستور row_number را تحقیق کنیم .(مثلا برای برگرداندن رکوردهای ۵۰ تا ۶۰ از یک جدول).

این تابع برای رتبه بندی بر اساس یک فیلد خاص به کار می رود.

شکل نحوی آن به صورت زیر است :

```
ROW_NUMBER ( )  OVER ( [ <partition_by_clause> ] <order_by_clause> )
```

Partition_by_clause : به وسیله این آرگومان می توان برای هر فیلد شماره گذاری را براساس یک چیز خاص ریست کرد.

order_by_clause : از این آرگومان برای مرتب سازی بر اساس یک فیلد خاص استفاده کرد و سپس بر اساس این فیلد رتبه بندی را انجام داد.

مثال :

```
WITH stt1 AS
(SELECT *,
ROW_NUMBER() OVER ( order by st#)as RowNumber
FROM stt)
SELECT *
FROM  stt1
WHERE RowNumber between 2 and 5
```

Results | Messages

	St#	StName	StFamily	StFatherName	roomate	RowNumber
1	101	44444444	madihi	ali	103	2
2	102	majid	klsajd	NULL	100	3
3	103	ali	ahmadi	NULL	102	4
4	104	majid	basirati	ali	104	5

```
WITH stt1 AS
(SELECT *,
ROW_NUMBER() OVER (partition by substring(stname,1,1) order by
st#)as RowNumber
FROM stt)
SELECT *
FROM  stt1
WHERE RowNumber between 2 and 5
```

Results | Messages

	St#	StName	StFamily	StFatherName	roomate	RowNumber
1	101	44444444	madihi	ali	103	1
2	103	ali	ahmadi	NULL	102	1
3	107	ahmad	rezaee	hosein	102	2
4	108	ali	majid	akjsd	103	3
5	105	hasan	ahmadzade	reza	107	1
6	100	mmmmmmm	alian	ahmad	104	1
7	102	majid	klsajd	NULL	100	2
8	104	majid	basirati	ali	104	3
9	106	sadegh	rahmani	mashal	106	1

استفاده از تابع row_number بدون استفاده از with :

```
select * from (select *,row_number() over (order by st#) as
rownumber
from stt) stt1
where rownumber between 1 and 5
```

نکته : برای دسترسی به یک دیتا بیس بدون نیاز به use از کد زیر استفاده می کنیم:

```
Select  *  from  نام دیتابیس . dbo . نام جدول
```

مثال :

```
select * from db1.dbo.test
select * from db1..test
```

دستور where :

```
select * from reg where st#=101
```

Results | Messages

	St#	C#	Term#	Grade
1	101	202	2	NULL
2	101	202	3	15
3	101	203	4	16

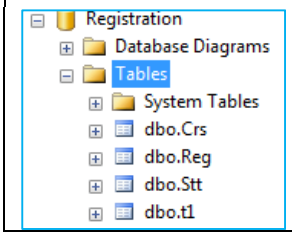
فیلدهایی از جدول reg که نمره آنها وارد شده است را بر میگرداند.

```
select * from reg where grade  is not null
```

ساختن جدول با استفاده از دستور select :

```
select * into t1 from stt
```

این دستور یک کپی از جدول Stt را در جدول t1 قرار می دهد. این جدول در لیست جداول دیتا بیس اضافه می شود. برای مشاهده آن باید روی table در دیتابیس registration در پنجره object explorer راست کلیک کنیم و refresh را بزنیم.وقتی دستور بالا را یک بار اجرا کنیم برای باردوم خطا می دهد چون جدول t1 ساخته شده و وجود دارد و ما اجازه ساخت دو جدول هم نام را در دیتابیس نداریم.



این دستور هم یک جدول به نام t2 ایجاد می کند که حاوی اطلاعات دانشجوی ۱۰۱ از جدول stt می باشد.

```
select  * into t2 from stt
where st#=101
```

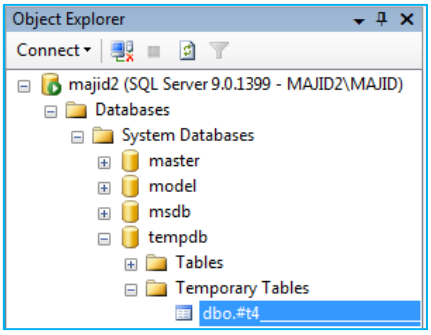
Table - dbo.t2 | majid2.Registration - SQLQuery1.sql* | Summary

St#	StName	StFamily	StFatherName	roomate
101	44444444	madihi	ali	103
NULL	NULL	NULL	NULL	NULL

ساخت جداول موقت :

ساخت جداول موقت در sql server کاربردهای بسیاری دارد. ساخت جدول موقت همانند ساخت جدول با استفاده از دستور select است که در قسمت قبل دیدیم با این تفاوت که قبل از نام جدول باید از علامت # استفاده کرد.

```
select  * into #t4 from stt
where st#=101
```



این جدول t4 به صورت موقت ساخته و در دیتابیس های سیستمی و در tempdb ذخیره می شود. اگر یک بار نرم افزار sql server را بسته و دوباره باز کنیم این جدول دیگر وجود ندارد.

هنگامی که یک جدول موقت را در یک کوئری خاص ایجاد می کنیم اگر با استفاده از دکمه new query یک کوئری جدید ایجاد کنیم نمی توانیم از این جدول موقت در کوئری جدید استفاده کنیم.درواقع می توان گفت که این جدول یک جدول محلی (همانند متغیرهای محلی) می باشد.

یکی از کاربردهای جداول موقت در مواقعی است که بخواهیم بدون استفاده از تابع row_number چند رکورد از وسط جدول را استخراج کنیم. برای این کار هنگام ساخت جدول موقت یک فیلد با نوع داده auto number به آن اضافه میکنیم و سپس با استفاده از این فیلد رکوردها را استخراج می کنیم :

```
select identity(int,1,1) as radif,* into #t5 from stt
select * from #t5
where radif between 20 and 30
```

تمرین سر کلاس : قطعه دستوری بنویسید که رکوردهای یک جدول را یک درمیان بدهد.

```
select identity(int,1,1) as radif,* into #t6 from stt
select * from #t6 where radif % 2=1
```

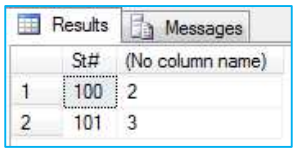
نکته : در صورتی که بخواهیم از جدول موقتی که در یک کوئری ایجاد کرده ایم در کوئری های دیگر نیز استفاده کنیم هنگام ایجاد جدول موقت به جای یک علامت # دو علامت # می گذاریم:

```
select  * into ##t4 from stt
```

درواقع با گذاشتن دو علامت # ما یک جدول سراسری ایجاد کرده ایم. البته باید به این نکته توجه کنیم که دراین مورد بازهم جدول ما موقت است و با بستن و باز کردن مجدد نرم افزار ، جدول حذف می شود.

تمرین سر کلاس : دستوری که شماره دانشجویی دانشجویانی را نشان دهد که بیش از یک درس اخذ کرده اند.با این فرض که دانشجویی در هیچ درسی نمی افتد.

```
SELECT St#,COUNT (St#) FROM reg
GROUP BY st#
HAVING COUNT(st#)>1
```



Results | Messages

	St#	(No column name)
1	100	2
2	101	3

تمرین سر کلاس : دستوری بنویسید که نام دانشجویانی که درس شماره ۲۰۲ را اخذ کرده اند نشان دهد.

راه اول :

```
SELECT stname FROM stt,reg
WHERE stt.st#=reg.St# AND reg.C#=202
```

راه دوم :

```
SELECT * FROM stt
WHERE st# IN(SELECT st# FROM reg WHERE c#=202)
```

راه سوم : استفاده از Inner Join

```
SELECT stname FROM stt INNER JOIN reg ON stt.St#=reg.St#
WHERE reg.C#=202
```

دستور Inner Join :

شکل کلی آن به صورت زیر است :

Table 1	inner join	table 2	on	شرط

در قسمت شرط ، باید شرط join را بنویسیم ، یعنی تعیین کنیم که join براساس کدام فیلد دو جدول صورت گیرد.

خروجی این دو دستور کاملا مشابه است :

```
1. SELECT * FROM stt,Reg WHERE stt.st#=reg.st#
2. SELECT * FROM stt INNER JOIN Reg ON stt.St#=reg.St#
```

تمرین سر کلاس : نام دانشجویانی را که بیش از یک درس اخذ کرده اند.

```
SELECT Stname FROM stt INNER JOIN reg ON stt.St#=reg.St#
GROUP BY stname
HAVING COUNT(stname)>1
```

تمرین سر کلاس : دستوری که نام درس هایی که دانشجوی علی اخذ کرده را چاپ کند.

```
SELECT cname  FROM crs
INNER JOIN reg ON crs.C#=reg.C#
INNER JOIN stt ON stt.st#=reg.St#
WHERE stt.StName='ali'
```

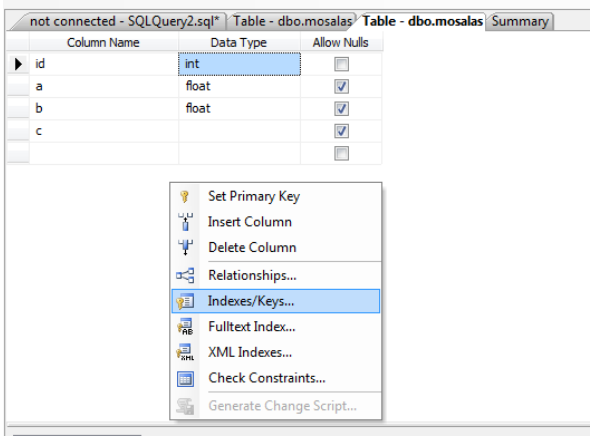
تمرین سر کلاس : دستوری بنویسید که نام هم اتاقی دانشجوی ۱۰۱ را برگرداند.

```
SELECT s2.stname FROM stt s1 INNER JOIN stt s2 ON s1.roomate=s2.St#
WHERE s1.st#=101
```

تمرین : دستوری بنویسید که نام دانشجویانی که همه دروس موجود در جدول Crs را اخذ کرده اند را برگرداند.

```
SELECT stname FROM Reg
INNER JOIN Stt ON stt.St#=reg.St#
GROUP BY stname
HAVING COUNT(DISTINCT(c#)) = (SELECT COUNT(*) FROM Crs)
```

تعریف ایندکس (Index):



برای تعریف ایندکس ، در نمای modify یک جدول ، ودرمحیط سفید رنگ کلیک راست کرده و گزینه Indexes/key... را انتخاب می کنیم.

فیلدهایی که در Join ها و select ها زیاد شرکت می کنند برای ایندکس کردن مناسب هستندو سرعت کار رابالا می برند.

اس کیو ال سرور به طور اتوماتیک فیلد کلید را ایندکس گذاری می کند.

نکته : می توان ایندکس ترکیبی هم استفاده کرد.

رابطه (Relationship) :

کاربرد : برای قوانین جامعیت (سازگاری)

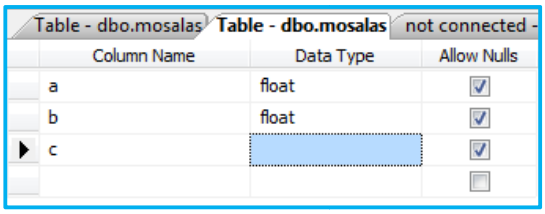
تمرین ۲ : توضیح برخی قسمت های Column Properties درهنگام طراحی جداول :

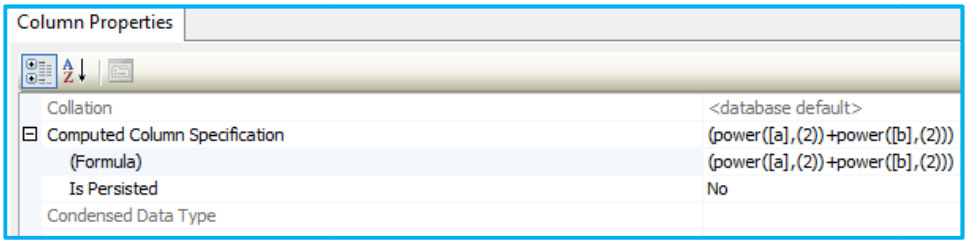
۱) **Default value or binding :**

اگر برای یک فیلد این مقدار وارد شود در هنگام وارد کردن اطلاعات در جدول اگر این فیلد را خالی رها کنیم مقداری که در این قسمت در هنگام طراحی جدول وارد کردیم به عنوان مقدار پیش فرض در این فیلد قرار خواهد گرفت.

۲) **Computed column specification :**

اگر مقدار یک فیلد از جدول به محاسباتی نیاز داشته باشد در این قسمت محاسبات مورد نیاز را وارد می کنیم. برای مثال فرض کنیم می خواهیم جدولی با سه فیلد داشته باشیم که اطلاعات اضلاع یک مثلث قائم الزاویه را نگه می دارد. همان طور که می دانیم ضلع وتر یک مثلث قائم الزاویه با داشتن دوضلع دیگر قابل محاسبه است. پس ما می توانیم مقدار فیلد وتر را با محاسبه مجموع مجذور دو فیلد دیگر به دست آوریم. برای این منظور ما فیلد وتر را تعریف کرده و در قسمت computed column specification→ Formula فرمول power(a,2)+power(b,2) را وارد می کنیم (بافرض اینکه دو فیلد دیگر a و b باشند).

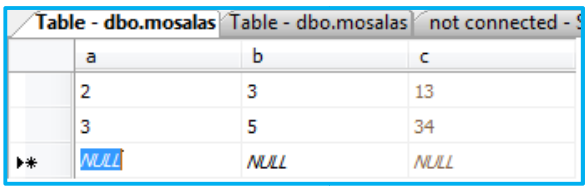




دونکته دراینجا قابل توجه است :

نکته اول اینکه همان طور که در شکل اول می بینید اگر یک فیلد ، محاسباتی باشد دیگر نمی توان برای آن نوع تعریف کرد ، پس مطابق شکل می بینید که در فیلد c قسمت data type خالی است.

نکته دوم اینکه در هنگام وارد کردن اطلاعات دیگر کاربر اجازه وارد کردن اطلاعات در فیلدc را ندارد بلکه این فیلد به صورت خودکار پر می شود.



❖ **Is persisted :**

اگر این گزینه برابر No باشد، اصلاً فضایی برای فیلد C در جدول در نظر گرفته نخواهد شد و همواره این مقدار محاسبه می گردد. اما اگر Yes باشد، مقدار C در جدول ذخیره می‌شود.

۳) **Is identity :**

اگر این گزینه را برابر yes قرار دهیم می توانیم فیلدی از نوع Auto Number ایجاد کنیم که مقدارش با ایجاد هر رکورد افزایش می یابد.

❖ **Indentity increment :**

هنگامی که یک فیلد Auto number ایجاد کردیم به وسیله این گزینه مشخص می کنیم که این فیلد چه مقدار به ازای هر رکوردی که به جدول اضافه می شود افزایش یابد.

❖ **Indentity seed :**

به وسیله این گزینه مشخص می کنیم که فیلد Auto number از چه عددی شروع شود.

تمرین ۳ : آیا می توان یک فیلد که Identity آن yes است (یعنی auto number است) را به صورت دستی مقدار دهی کرد؟

بله ، با استفاده از کد زیر می توان این کار را انجام داد. این دستور به ما اجازه می دهد که به صورت دستی یک فیلد Identity را مقدار دهی کنیم.

```
SET IDENTITY_INSERT [ database_name . [ schema_name ] . ] table { ON | OFF }
```

اگر از ON استفاده کنیم یعنی اجازه دستی وارد کردن داریم و اگر از OFF استفاده کنیم این اجازه را نداریم.

مثال : فیلدی به نام identit در جدول Stt داریم که از نوع int و identity تعریف کرده ایم و حالا می خواهیم با یک دستور insert همه فیلدهارا مقدار دهی کنیم. برای این کار باید ابتدا قابلیت دستی وارد کردن این فیلد را فعال کنیم و سپس این کار را انجام دهیم.

```
SET IDENTITY_INSERT  dbo.stt ON
INSERT INTO stt
(identit,St#,StName,StFamily,StFatherName,roomate) VALUES
(6,175,'majid','basirati','ali',200)
```

تمرین ۴ : هنگامی یک فیلد را ازنوع auto number تعریف می کنیم مقدار این فیلد با اضافه کردن هر رکورد افزایش می یابد. حتی اگر تمام فیلدهای جدول را حذف کنیم و مجددا شروع به درج رکورد کنیم مقدار این فیلد از ادامه درج می شود و مقدار آن ریست نمی شود. چگونه می توان کاری کرد که با حذف تمام رکورهای جدول مقدار این فیلد نیز ریست شود؟(بدون استفاده از truncate)

جواب :۱) اگر با استفاده از دستور **Truncate** جدول را پاک کنیم مقدار این فیلد نیز reset می شود. البته دراین حالت مقدار آن مجددا از یک شروع می شود ونیز جدول کامل پاک می شود.

Truncate	**Table**	نام جدول

۲) با استفاده از دستور DBCC CHECKINDENT :

DBCC CHECKIDENT **(** '*table_name*' [**,** { NORESEED | { RESEED [**,** *new_reseed_value*] } }] **)** [WITH NO_INFOMSGS]

به وسیله این دستور می توان مقدار Identity را از ریست کرد (دوباره تنظیم کرد) با این تفاوت که دیگر همه فیلدهای جدول دراین حالت پاک نمی شود.

مثال: جدولی داریم به نام ident که شامل دو فیلد است که یکی id از نوع int و Identity است و دیگری از نوع varchar. می خواهیم به وسیله دستوری مقدار فیلد id را با توجه به آخرین رکورد موجود در جدول تنظیم کنیم. این کاربرای این است که اگر چند سطر از انتهای جدول پاک شد مقدار این فیلد دوباره با توجه به مقدار آخرین رکورد مقداردهی شود.

```
DECLARE @a INT
SELECT TOP 1 @a=id FROM ident ORDER BY id DESC
DBCC CHECKIDENT ('ident',RESEED,@a)
```

Trigger :

یک نوع خاص از روال ذخیره شده است که با وقوع یک رویداد در sql server شروع به کار کرده و هیچ پارامتر ورودی یا خروجی ندارد.کاربرد آن وقتی است که بخواهیم درکنار دستورات DML ویا DDLکارهای دیگری نیز انجام دهیم.

```
Create      trigger     نام تریگر      on      نام جدول یا ویو

{  after | instead of } { [ insert ] | , [ update ] | , [ delete ] }

As

دستورات
```

علامت { } یعنی این قسمت در گرامر اجباری است.

مثال : میخواهیم بعد ازاجرای دستور insert یک پیغام OK چاپ شود.

```
CREATE TRIGGER trigger1 ON stt
AFTER   INSERT
AS
PRINT 'OK'
```

مثال ۲ :

```
CREATE TRIGGER TRIGGER2 ON stt
AFTER DELETE
AS
PRINT 'delete completed'
```

مثال ۳ :

```
CREATE TRIGGER trigger3 ON stt
INSTEAD of   INSERT
AS
PRINT 'not inserted completed'
```

دراین مثال به جای AFTER از INSTEAD OF استفاده شده است که تفاوت آن در این است که وقتی AFTER است پس از عمل INSERT ، دستورات تریگر اجرا می شود ولی هنگامی که از INSTEAD OF استفاده کنیم به جای آن عمل INSERT ، دستورات تریگر اجرا می شود.پس درمثال فوق هرعمل درجی که بخواهد روی جدول Stt انجام شود ، عمل درج صورت نگرفته و پیغام چاپ می شود.

نکته : امکان گذاشتن after , instead of روی یک جدول وجود دارد ولی instead of ارجحیت دارد.

از تریگرها می توان برای محافظت و یا ثبت اطلاعات افرادی که در دیتابیس ما دستکاری می کنند استفاده کرد. برای مثال در زیر یک تریگر تعریف شده که هنگام حذف رکوردی از جدول Stt شماره دانشجویی دانشجوی حذف شده و تاریخ و ساعت انجام این حذف را در جدولی ذخیره می کند. این کار در دیتابیس های تحت وب که امکان هک شدن وجود دارد بسیار مناسب است.

```
CREATE TRIGGER td1 ON stt
AFTER DELETE
AS
DECLARE @s INT
SELECT @s=st# FROM deleted
INSERT INTO del_st (st#,date) VALUES (@s,GETDATE())
```

هنگامی که ما یک تریگر برای یک جدول ایجاد می کنیم یک جدول مجازی ایجاد می کند مثلا وقتی تریگر insert ایجاد می کنیم یک جدول به نام inserted و وقتی یک تریگر delete ایجادمی کینم یک جدول به نام deleted ایجاد می شود که سطرهای حذفی یا درجی ما را در این جداول درج می کند. شرح اعمال به صورت زیر است:

Insert : هنگامی که ما یک عمل درج را در جدولی انجام می دهیم تمام اطلاعات آن رکورد(ها) در جدول inserted ذخیره می شود.

Delete : هنگام عمل حذف یک یا چند رکورد ، اطلاعات حذفی در جدول deleted ذخیره می شود.

Update : هنگامی که ما یک رکورد را ویرایش می کنیم رکورد قبل از ویرایش در جدول deleted و رکورد بعد از ویرایش در جدول inserted ذخیره می شود.

مثال : تریگر زیر هنگام حذف یک یا چند رکورد ، محتوای جدول deleted را به ما نشان می دهد.

```
CREATE TRIGGER td2 ON stt
AFTER DELETE
AS
SELECT * FROM deleted
```

تمرین : یک تریگر از نوع after insert برای یک جدول ایجاد کرده ایم که بدنه این تریگر قطعه کدی است که کار درج در یک جدول دیگر را انجام می دهد.اگر هنگام اجرای تریگر که مسلما بعداز عمل درج در جدول اتفاق می افتد خطایی رخ دهد(مثلا برق قطع شود) عمل درج در جدول اولیه انجام می شود یا نه؟

جواب : خیر ، عمل درج انجام نمی شود.

تمرین : اگر چندین تریگر after insert روی یک جدول ایجاد کنیم و بعد عمل درج را انجام دهیم ترتیب اجرای تریگرها چگونه است؟

جواب : تریگر های اول و آخر را می توان به وسیله دستور sp_settriggerorder به دلخواه تنظیم کرد ولی بقیه کاملا random اجرا می شوند. مطلب زیر در همین مورد دقیقا از help نرم افزار sql server 2005 گرفته شده است:

The **first** and **last** AFTER triggers to be executed on a table **can be specified** by using **sp_settriggerorder**. Only one first and one last AFTER trigger for each INSERT, UPDATE, and DELETE operation can be specified on a table. If there are other AFTER triggers on the same table, they are randomly executed.

تمرین : طرز کار با sp_settriggerorder را شرح دهید و مثال بزنید.

sp_settriggerorder [**@triggername** =] '[*triggerschema***.**] *triggername*'

, [**@order** =] '*value*'

, [**@stmttype** =] '*statement_type*'

[, [@namespace =] { 'DATABASE' | 'SERVER' | NULL **}**]

@triggername : نام تریگر

@order : تعیین می کند که تریگر به عنوان اولین ، آخرین یا بدون تنظیم برای اجرا تنظیم شود. مقادیری که این آرگومان می تواند بپذیرد : First , Last , None

@stmttype : تعیین نوع تریگر. مقادیری که می تواند بگیرد : ... , Update , Insert , Delete

@namespace : تعیین دیتابیس برای مواقعی که تریگرها ازنوع DDL می باشند.

مثال : برای جدول Stt ، ده تریگر after insert با نام های m1 تا m10 ایجاد کرده ایم که عمل چاپ عدد ۱ تا ۱۰ را انجام می دهند. به طور معمول وقتی عمل درج انجام می شود عدد ۱ ابتدا و عدد ۱۰ در انتها چاپ می شود. به وسیله دستور sp_settriggerorder تریگری که در ابتدا و انتها اجرا می شوند را تغییر می دهیم.

```
sp_settriggerorder @triggername='m8' , @order='first' ,
@stmttype='insert'
sp_settriggerorder @triggername='m5' , @order='last' ,
@stmttype='insert'
```

پس از اجرای این دو خط کد ، هرعمل درجی انجام دهیم عدد ۸ ابتدا و عدد ۵ در انتها چاپ می شود.

تمرین سرکلاس: یک تریگر برای یک جدول بنویسید که وقتی کاربر دستور delete را انجام میدهد حرف D را چاپ کند ، با اجرای دستور insert حرف I و با اجرای دستور update حرف U را چاپ کند.

```
Create Trigger tr1 On stt
After Insert , UpDate , Delete AS
Declare @I BigInt , @D BigInt , @T Char (1)
```

```
Set @I = (Select Count (*) From Inserted )
Set @D = (Select Count (*) From Deleted )
Set @T = Case
When @I < > 0 and @D = 0 Then 'I'
When @I < > 0 and @D < > 0 Then 'U'
When @I = 0 and @D < > 0 Then 'D'
END
IF @t='I' PRINT 'I'
IF @t='D' PRINT 'D'
IF @t='U' PRINT 'U'
```

تمرین: تریگری بنویسید که شماره دانشجویی زیر ۵۰۰ را اجازه ورود به stt ندهد.

```
ALTER TRIGGER trr1 ON stt
INSTEAD OF INSERT
AS
IF (SELECT COUNT(*) FROM INSERTED WHERE st#<500)=0
     INSERT INTO stt  SELECT * FROM INSERTED
ELSE
     PRINT 'not inserted'
```

دستوراتی بنویسید که اعداد تصادفی بین ۱۰۰۰ تا ۲۰۰۰ رو در فیلد b جدول x1 وارد کند.(x1 ، یک جدول با دوفیلد است ، a و b که فیلد a در آن autonumber و b هم int است.

```
DECLARE @a INT,@count INT
SET @count=500
WHILE (@count >0)
BEGIN
     SET @a= FLOOR(RAND()*1000)+1000
     INSERT INTO x1 (b) VALUES (@a)
     SET @count=@count-1
END
```

دستوری بنویسید که یک عدد تصافی از یک تا ۵۰۰ ایجاد کرده و رکورد مورد نظر با اون عدد را حذف کند.

```
DECLARE @a1 INT
SET @a1=FLOOR(RAND()*500)
DELETE FROM x1 WHERE a=@a1
```

تمرین منزل :دستوری بنویسید که با استفاده از select اون رکورد پاک شده را تشخیص دهد و سپس خاصیت idnetity فیلد a رو off کنید و دوباره آن عدد را درج کنید.

با دستورکرسر

```
DECLARE @s1 INT,@s2 INT
DECLARE a CURSOR
FOR
     SELECT a FROM x1

OPEN a
FETCH NEXT FROM  a INTO @s1
WHILE (@@FETCH_STATUS=0)
BEGIN
          FETCH NEXT FROM a INTO @s2
     IF @s1+1<>@s2
          PRINT @s1+1
     SET @s1=@s2
END
CLOSE a
DEALLOCATE a
```

بدون کرسر

```
SELECT s1.a-1 a FROM x1 s1
INNER JOIN x1 s2 ON s1.a=s2.a+2
EXCEPT
SELECT a FROM x1
```

Cursor :

برای تعریف کرسر

```
Declare نام کرسر cursor

For

      Select دستور
```

مثال :

```
DECLARE a CURSOR
FOR
     SELECT * FROM Stt


OPEN a
FETCH NEXT FROM  a
CLOSE a
DEALLOCATE a
```

```
OPEN a
FETCH NEXT FROM  a
WHILE (@@FETCH_STATUS=0)
     FETCH NEXT FROM  a
CLOSE a
DEALLOCATE a
```

@@fetch_status : این تابع تا زمانی که کارش با موفقیت انجام شود عدد صفر برمیگرداند ولی در غیر این صورت عدد غیر صفر برمیگرداند.

```
DECLARE @s1 INT,@s2 INT , @s3 VARCHAR(50)
DECLARE a CURSOR
FOR
     SELECT st#,StName FROM Stt

OPEN a
```

```
FETCH NEXT FROM  a
INTO @s1,@s3
WHILE (@@FETCH_STATUS=0)
     FETCH NEXT FROM  a
CLOSE a
DEALLOCATE a

PRINT @s1
PRINT @s3
```

برای اینکه در کرسر دوطرفه حرکت کنیم باید هنگام تعریف کلمه کلیدی scroll استفاده کنیم.

```
DECLARE @s1 INT,@s2 INT
DECLARE a CURSOR SCROLL
FOR
     SELECT a FROM x1

OPEN a
FETCH PRIOR FROM  a

CLOSE a
DEALLOCATE a
```

برای رفتن به یک رکورد خاص(مثلا ۳۰)

```
FETCH ABSOLUTE 30 FROM  a
```

برای رفتن به مثلا بیستمین رکورد نسبت به رکورد فعلی

```
FETCH RELATIVE 20 FROM  a
```

دستور زیر به رکورد ۱۰۰ از پایین میرود

```
Fetch absoulute -100 from a
```

تمرین سرکلاس :

فیلد b را با دنباله فیبوناچی پرکنید.(این تمرین را با آپدیت هم می توان انجام داد)

```
DECLARE @s1 bigINT,@s2 bigINT,@s3 bigINT,@a INT,@bikhod INT

DECLARE c1 CURSOR SCROLL
FOR
     SELECT a,b FROM x1

OPEN c1
FETCH NEXT FROM  c1 INTO @a,@s1
```

```
FETCH NEXT FROM c1 INTO @a,@s2

WHILE (@@FETCH_STATUS=0)
BEGIN
          SET @s3=@s1+@s2
          FETCH NEXT FROM c1 INTO @a,@bikhod
          UPDATE x1 SET b=@s3 WHERE a=@a
          SET @s1=@s2
          SET @s2=@s3

END

CLOSE c1
DEALLOCATE c1
```