ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ &
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ **ΙΙ** ΕΡΓΑΣΤΗΡΙΟ

Σημειώσεις για την Transact-SQL

Η Transact-SQL (T-SQL) είναι ή υλοποίηση της γλώσσας ερωταπαντήσεων SQL (Structured Query Language) που έχει δημιουργηθεί από τη Microsoft και επεκτείνει τις δυνατότητές της SQL. Η Transact-SQL είναι ενσωματωμένη στο περιβάλλον του SQL Server και προσθέτει στην SQL επιπλέον δομές που επιτρέπουν την πιο αποτελεσματική και ευέλικτη διαχείριση μιας σχεσιακής βάσης δεδομένων. Μερικές από αυτές είναι η χρήση μεταβλητών, δομές ελέγχου και επανάληψης, δημιουργία διαδικασιών και συναρτήσεων.

Το παρόν κεφάλαιο των σημειώσεων ασχολείται με κάποιες από τις επεκτάσεις που έχουν ενσωματωθεί στην Transact-SQL. Αυτές οι επεκτάσεις μπορούν να χρησιμοποιηθούν έτσι ώστε να δημιουργήσουν προγράμματα τα οποία δίνουν την ευελιξία στο χρήστη να διαχειρίζεται πιο αποδοτικά μια βάση δεδομένων. Ο SQL Server μας παρέχει έναν query editor στον οποίο γράφουμε και εκτελούμε (**!**Execute) τα προγράμματα της Transact-SQL. Σε περίπτωση που υπάρχει σφάλμα ο SQL-Server εμφανίζει ένα μήνυμα με το είδος του σφάλματος και σε ποια γραμμή αυτό βρίσκεται.

1. Δήλωση μεταβλητών

Η Transact-SQL μας δίνει τη δυνατότητα να δημιουργήσουμε μπλοκ κώδικα με μια ή περισσότερες εντολές SQL. Επίσης μας επιτρέπει να χρησιμοποιήσουμε δομές ελέγχουν ροής των δεδομένων για χρήση των προτάσεων αυτών καθώς και δήλωση μεταβλητών στις οποίες θα αποθηκεύσουμε πληροφορίες που μπορούμε ύστερα να τις διαχειριστούμε μέσα από αυτά. Οι μεταβλητές που ορίζονται από το χρήστη ονομάζονται τοπικές μεταβλητές και αναφέρονται μόνο στο μπλοκ κώδικα στο οποίο δηλώνονται. Ο SQL Server υποστηρίζει και καθολικές μεταβλητές, που παρέχονται από το σύστημα και είναι προκαθορισμένες.
Μπορούμε να δηλώσουμε τοπικές μεταβλητές χρησιμοποιώντας την εντολή declare ακολουθούμενο από το όνομα της μεταβλητής και τον τύπο δεδομένων της. Τύποι δεδομένων που υποστηρίζονται από την Transact-SQL παρουσιάζονται στο **Παράρτημα Α**. Οι μεταβλητές θα πρέπει να ξεκινάνε με το χαρακτήρα @ ως πρώτο χαρακτήρα του ονόματός τους.

```
Declare  @όνομα_μεταβλητής  τύπος δεδομένων, …
```

Παράδειγμα 1.1:

```
DECLARE @MyInt int, @MyVariable nvarchar(20), @f float
```

Μετά τη δήλωση τους παίρνουν την τιμή null. Για την εμφάνιση των τιμών των μεταβλητών χρησιμοποιούμε την εντολή print. Μπορούμε να εκχωρήσουμε μια τιμή σε μια τοπική μεταβλητή χρησιμοποιώντας:

1. Την εντολή **SELECT**
2. Την εντολή **SET**

Παράδειγμα 1.2:

```
declare @var1 int, @var2 int
set @var1=5
select @var2=5
print 'to apotelesma einai'
```

```
print @var1+@var2
```

Η εκτέλεση του παραδείγματος δίνει την έξοδο:

```
to apotelesma einai 10
```

Παράδειγμα 1.2:

```
declare @salary money
declare @max_salary money

set @salary = (select salary from EMPLOYEE where emp_id ='A276654')
set @max_salary = (select MAX(salary) from employee)
print @salary
print @max_salary
```

Η εκτέλεση του παραδείγματος εμφανίζει το μισθό του εργαζομένου με emp_id ='A276654' και το μέγιστο μισθό όλων των εργαζομένων:

```
1500.00
2000.00
```

2. Συναθροιστικές συναρτήσεις και Ομαδοποίηση

Οι συναθροιστικές συναρτήσεις και η ομαδοποίηση τιμών είναι χρήσιμες σε πολλές περιπτώσεις στη διαχείριση μιας βάσης δεδομένων. Γι αυτό το λόγο η T-SQL υποστηρίζει τις συναρτήσεις αυτές. Οι συναθροιστικές συναρτήσεις μπορούν να εφαρμοστούν σε μια ομάδα τιμών από τη βάση δεδομένων (πολλαπλές τιμές από μια στήλη) και επιστρέφουν πάντα μια τιμή. Συνηθισμένες συναρτήσεις που εφαρμόζονται σε ομάδες αριθμητικών τιμών είναι το άθροισμα (SUM), ο μέσος όρος (AVG), η μέγιστη τιμή (MAX), η ελάχιστη τιμή (MIN). Επίσης υπάρχει η συνάρτηση COUNT (ΠΛΗΘΟΣ) που υπολογίζει το πλήθος των πλειάδων (γραμμών) σε ένα πίνακα. Οι συναρτήσει αυτές μπορούν να χρησιμοποιηθούν με την εντολή select και το αποτέλεσμά τους να αποθηκευτεί σε μια μεταβλητή.

Παράδειγμα 2.1

```
DECLARE @rows int;
SET @rows = (SELECT COUNT(*) FROM employee)
print @rows
```

Το παραπάνω παράδειγμα εκχωρεί το πλήθος όλων των γραμμών του πίνακα employee στη μεταβλητή @rows και ύστερα εμφανίζει το περιεχόμενό της στην οθόνη.

Παράδειγμα 2.2

```
Declare @sum_salary money, @avg_salary money
Select @sum_salary = sum(salary) from employee where dnumber=1
Select @avg_salary = avg(salary) from employee where dnumber=2
Print @sum_salary
Print @avg_salary
```

Στο παραπάνω παράδειγμα δηλώνονται δύο μεταβλητές. Στην πρώτη αποθηκεύεται το άθροισμα από όλες τις τιμές στην στήλη salary του πίνακα employee όπου dnumber=1. Στη

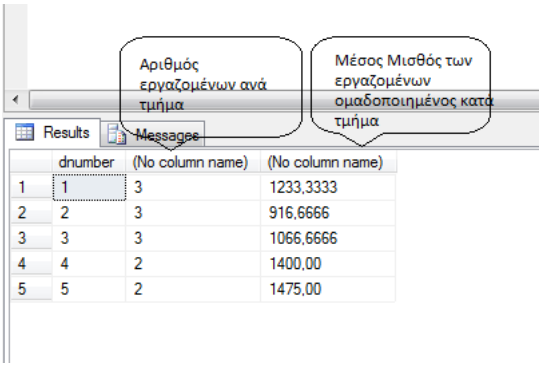
δεύτερη αποθηκεύεται ο μέσος όρος των μισθών των εργαζομένων όπου dnumber=2 και οι τιμές τους εμφανίζονται με την εντολή print.
Αν θέλουμε να εφαρμόσουμε συναθροιστικές συναρτήσεις σε υποομάδες πλειάδων ενός πίνακα με βάση τις τιμές κάποιων γνωρισμάτων χρησιμοποιούμε την εντολή group by.

Παράδειγμα 2.3

Για κάθε τμήμα να ανακτηθεί ο κωδικός του τμήματος, ο αριθμός των εργαζομένων που εργάζονται σε αυτό και οι μέσοι μισθοί τους.

```
Select dnumber, count(*), avg(salary) from  employee
group by dnumber
```

Το αποτέλεσμα τις εκτέλεσης του παραπάνω παραδείγματος είναι:



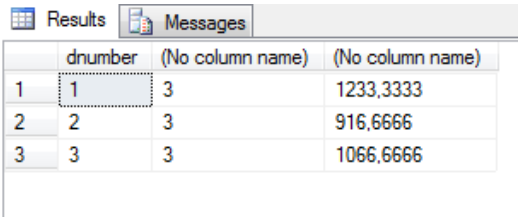
	dnumber	(No column name)	(No column name)
1	1	3	1233,3333
2	2	3	916,6666
3	3	3	1066,6666
4	4	2	1400,00
5	5	2	1475,00

Μπορούμε να εμφανίσουμε μόνο τις ομαδοποιημένες πλειάδες που μας ενδιαφέρουν σύμφωνα με κάποια συνθήκη χρησιμοποιώντας την πρόταση HANING. Η πρόταση αυτή εισάγει μια συνθήκη πάνω στις ομάδες πλειάδων, έτσι ώστε να εμφανίζονται μόνο οι ομάδες που ικανοποιούν τη συνθήκη.

Παράδειγμα 2.4

```
declare @number int
set @number=2
Select dnumber, count(*), avg(salary) from  employee
group by dnumber
having count(*)>@number
```

Το αποτέλεσμα τις εκτέλεσης του παραπάνω παραδείγματος είναι:



	dnumber	(No column name)	(No column name)
1	1	3	1233,3333
2	2	3	916,6666
3	3	3	1066,6666

Σε αντίθεση με το παράδειγμα 2.3 εδώ επιστέφονται μόνο ο αριθμός τμήματος, ο αριθμός εργαζομένων και ο μέσος μισθός των εργαζομένων των τμημάτων που έχουν πάνω από 2 εργαζομένους (`having count(*)>@number`).

3. Πρόταση if … else

Η πρόταση if λειτουργεί με τον ίδιο τρόπο όπως σε όλες τις γλώσσες προγραμματισμού. Η εντολή αυτή μπορεί να εκτελεί μια ή περισσότερες εντολές SQL ακολουθούνται από μια συνθήκη αν αυτή είναι true. Αν περιέχεται και η εντολή else σε περίπτωση που η συνθήκη είναι false εκτελείται η δεύτερη ομάδα εντολών.
Η εντολή if συντάσσεται ως εξής:

```
If    <συνθήκη>

      Begin
        <εντολές>
      End

Else
      Begin
        <εντολές>
      End
```

Παράδειγμα 3.1

```
Declare @avg_salary money , @extra_money money, @salary money,
@emp_id char(8)

Set @extra_money= 200
Set @emp_id = 'H576543'
Set @avg_salary = (select avg(salary) from employee)
Set @salary = (select salary from employee where emp_id =@emp_id)

If @avg_salary >@salary
Begin
Update employee
set salary = salary +@extra_money where emp_id =@emp_id
end
else
print 'Ικανοποιητικός μισθός'
```

Στο παραπάνω παράδειγμα υπολογίζεται ο μέσος μισθός όλων των εργαζομένων και το αποτέλεσμα εκχωρείται στη μεταβλητή `@avg_salary` και ο μισθός του εργαζομένου με αριθμό ταυτότητας ίσο με το περιεχόμενο της μεταβλητής `@emp_id`. Το αποτέλεσμα εκχωρείται στη μεταβλητή `@salary`. Στη συνέχεια με τη βοήθεια των μεταβλητών ελέγχεται αν αυτός έχει μικρότερο μισθό από τον μέσο. Αν ισχύει η συνθήκη τότε προστίθενται στον συγκεκριμένο μισθό το περιεχόμενο της μεταβλητής @extra_money. Αλλιώς εμφανίζεται το μήνυμα `'Ικανοποιητικός μισθός'`

4. Πρόταση While

Η πρόταση while εκτελεί επαναληπτικά μια ή περισσότερες εντολές SQL που περιέχονται σε ένα μπλοκ, για όσο η συνθήκη είναι αληθής. Μετά από κάθε εκτέλεση των εντολών του μπλοκ ελέγχεται αν η συνθήκη είναι αληθής έτσι ώστε να καθοριστεί αν πρέπει αυτό να εκτελεστεί ξανά. Η εκτέλεση των εντολών στο μπλοκ μπορεί να ελεγχθεί μέσα στο μπλοκ με τις εντολές BREAK και CONTINUE. Η BREAK σταματάει την επαναληπτική εκτέλεση των

εντολών του μπλοκ και ξεκινά την εκτέλεση των εντολών μετά από αυτό το μπλοκ, δηλαδή όποια εντολή υπάρχει μετά το end. Η εντολή CONTINUE ξεκινά πάλι από την αρχή την επαναληπτική εκτέλεση του μπλοκ αγνοώντας κάθε εντολή μετά από αυτή. Η σύνταξη της while είναι η εξής:

```
while <εκφραση>
     begin
          <εντολές>
          break
          <εντολές>
          continue
          <εντολές>
     end
```

Παράδειγμα 4.1

```
Declare @counter int
Set @counter=1
While @counter<=5
  BEGIN
     Select pname, plocation, pnumber from project
     Where pnumber=@counter
     Set @counter=@counter + 2
  END
```

Στο παραπάνω παράδειγμα επιλέγεται ο αριθμός, το όνομα, η τοποθεσία των έργων (project) με μονό αριθμό (pnumber) μέχρι ο μετρητής (@counter) να ξεπεράσει το 5. Δηλαδή όσο ο μετρητής είναι μικρότερος ή ίσος του 5, επαναλαμβάνεται η εντολή Select και η εντολή set που αυξάνει το μετρητή κατά 2. Το αποτέλεσμα τις εκτέλεσης του παραπάνω παραδείγματος είναι:

Results | Messages

	pname	plocation	pnumber
1	Apografi proiontwn	Giannena	1

	pname	plocation	pnumber
1	Proothisi proiontwn	Arta	3

	pname	plocation	pnumber
1	Nees Agores	Giannena	5

Παράδειγμα 4.2

```
Declare @extra_money money, @max_salary float
set @extra_money=100
WHILE (SELECT AVG(Salary) FROM employee) < 900
BEGIN
   UPDATE employee
      SET Salary = Salary + @extra_money
```

```
      set @max_salary = (SELECT MAX(salary) FROM employee)
   IF (@max_salary) > 3000
      BREAK
   ELSE
      CONTINUE
END
PRINT 'Δεν θα γίνουν άλλες αυξήσεις μισθών'
```

Στο παραπάνω παράδειγμα γίνεται μια συνολική αύξηση μισθών. Υπολογίζεται ο μέσος μισθός όλων των εργαζομένων και αν είναι κάτω από 900 αυξάνει όλους τους μισθούς κατά την τιμή της μεταβλητής @extra_money. Κάθε φορά ελέγχεται αν ο μέγιστος μισθός είναι πάνω από 3000. Αν όντως ο μέγιστος μισθός είναι παραπάνω από 3000 η επανάληψη θα σταματήσει με το BREAK έτσι θα εκτελεστεί η εντολή μετά το **end**, δηλαδή η PRINT. Αλλιώς η επανάληψη θα ξεκινήσει από την αρχή.

5. Αποθηκευμένες διαδικασίες –Stored Procedures της t-sql

Οι αποθηκευμένες διαδικασίες, τυπικά, εκτελούν ένα σύνολο προτάσεων της Transact-SQL και χρησιμοποιούνται συνήθως για να αυξήσουν την απόδοση και την ταχύτητα επαναλαμβανόμενων εργασιών στη βάση δεδομένων. Οι αποθηκευμένες διαδικασίες μεταγλωττίζονται μια φορά, κατά τη δημιουργία τους και αποθηκεύεται ως αντικείμενο στη βάση. Ύστερα μπορούν να απλά εκτελεστούν όσες φορές επιθυμούμε. Μπορούν να δέχονται παραμέτρους και να επιστρέφουν πολλαπλές τιμές ως έξοδο, να περιέχουν εντολές που διαχειρίζονται τη βάση δεδομένων ή να καλούν άλλες διαδικασίες. Οι αποθηκευμένες διαδικασίες δημιουργούνται με την εντολή create procedure. Η γενική δομή μιας αποθηκευμένης διαδικασίας της Transact-SQL είναι η εξής:

```
Create procedure όνομα_διαδικασίας
<Παράμετροι-προαιρετικές>
        As
Σώμα διαδικασίας - sql εντολές, δηλώσεις μεταβλητών, cursor
```

Εκτελούμε τη διαδικασία με την πρόταση **execute** ακολουθούμενη από το όνομα της διαδικασίας.

```
execute όνομα_διαδικασίας
```

Παράδειγμα 5.1

```
create procedure select_employee
as
select emp_id, lastname, salary from employee


execute select_employee
```

Η διαδικασία `select_employee` επιλέγει τον αριθμό ταυτότητας, το επώνυμο και το μισθό όλων των εργαζομένων. Μετά την εκτέλεση της εμφανίζει τα εξής αποτελέσματα που είναι όλοι οι εργαζόμενοι καταχωρημένοι στον πίνακα employee:

Results | Messages

	emp_id	lastname	salary
1	H576543	Karras	900,00
2	A276654	Kwsta	1500,00
3	A475637	Floros	800,00
4	AE342134	Andreou	2000,00
5	B276548	Dimou	2000,00
6	E375898	Mitsios	950,00
7	G276548	Georgiou	2000,00
8	H576543	Toli	800,00
9	J674532	Foti	950,00
10	K475637	Petrou	800,00
11	K876543	Florou	800,00
12	S676548	Pappas	1000,00
13	X674532	Dara	950,00

Χρήση παραμέτρων

Οι παράμετροι είναι τα ορίσματα της διαδικασίας. Ένα όρισμα αντιπροσωπεύει την τιμή που παρέχεται σε μια παράμετρο όταν καλείται η διαδικασία. Μπορούμε προαιρετικά να δηλώσουμε μια ή περισσότερες παραμέτρους μετά την πρόταση Create procedure. Το όνομα των παραμέτρων ξεκινά με @ και είναι τοπικές στη διαδικασία που τις δημιουργεί. Δηλώνουμε τις παραμέτρους ως εξής:

```
Create procedure όνομα_διαδικασίας
--παράμετροι
@παράμετρος_1 τύπος_δεδομένων,
@παράμετρος_2 τύπος_δεδομένων, …


As < sql εντολές, δηλώσεις μεταβλητών, cursor>
```

Τυπικά οι παράμετροι χρησιμοποιούνται για να περνάνε τιμές από το χρήστη μέσα στη διαδικασία (πχ στα sql statements). Οι τιμές αυτές περνιούνται κατά την εκτέλεση της διαδικασίας και τοποθετούνται σε τη σειρά ως εξής:

`Execute` <όνομα_διαδικασίας> τιμή_παραμέτρου_1, τιμή_παραμέτρου_2 , ….

ή

`Execute` <όνομα_διαδικασίας> @παράμετρος_1= τιμή_παραμέτρου_1,
@παράμετρος_2= τιμή_παραμέτρου_2, ….

Παράδειγμα 5.2

```
create procedure select_employee
@emp_id char(8)
as
select emp_id, lastname, salary from employee
where emp_id=@emp_id
```

```
execute select_employee 'A276654'
```

Στην παραπάνω διαδικασία επιλέγεται μόνο ο αριθμός ταυτότητας, το επώνυμο και ο μισθός του εργαζομένου που ο αριθμός ταυτότητας δίνεται στην παράμετρο `@emp_id` από το χρήστη. Για παράδειγμα το αποτέλεσμα για την τιμή παραμέτρου A276654 είναι:

Results | Messages

	emp_id	lastname	salary
1	A276654	Kwsta	1500,00

Παράδειγμα 5.3

```
create procedure ADD_EMPLOYEE
@EMP_ID char(8),
@LASTNAME varchar (15),
@FIRSTNAME varchar (15),
@SALARY money,
@E_ADDRESS varchar (30),
@SEX char,
@BDATE datetime,
@SUPERVISOR_ID char(8),
@DNUMBER int

As
begin
insert into EMPLOYEE
(EMP_ID, LASTNAME, FIRSTNAME, SALARY, E_ADDRESS, SEX, BDATE,
SUPERVISOR_ID, DNUMBER)

values (@EMP_ID, @LASTNAME, @FIRSTNAME, @SALARY, @E_ADDRESS, @SEX,
@BDATE, @SUPERVISOR_ID, @DNUMBER)
end
```

Η παραπάνω διαδικασία εισάγει γραμμές στον πίνακα **employee**. Η διαδικασία αυτή έχει εννέα παραμέτρους που αντιπροσωπεύουν κάθε ένα πεδίο του πίνακα. Όταν εκτελεστεί η διαδικασία, οι τιμές θα περαστούν στις παραμέτρους από το χρήστη ως εξής:

```
execute ADD_EMPLOYEE 'B276548','Dimou','Giorgos',2000,'Araxthou 4',
'M', '10/08/1956', null, 1
```

Στη συνέχεια θα εκτελεστεί η εντολή insert της διαδικασίας, η οποία θα εισάγει τις τιμές που δόθηκαν στις παραμέτρους στον πίνακα **employee.** Τα πεδία που μπορούν να πάρουν τιμές null μπορούμε να τα παραλείψουμε.

6. Cursors

Οι cursor είναι ένας ειδικός τύπος μεταβλητής που χρησιμοποιείται για να αποθηκεύει πλειάδες (γραμμές) ενός πίνακα. Τις πλειάδες που αποθηκεύονται στον cursor μπορούν ύστερα να τις επεξεργαστούν εντολές της transact-SQL μέσα σε κάποιο πρόγραμμα. Αρχικά πρέπει να δηλώσουμε τον cursor και να καθορίσουμε το σύνολο των πεδίων (στηλών του πίνακα) που θα αποτελούν τις πλειάδες με τη χρήση της εντολής select. Με απλά λόγια με τη εντολή select επιλέγουμε για τον cursor τα επιθυμητά δεδομένα. Θα πρέπει επίσης να δηλώσουμε τις μεταβλητές στις οποίες θα καταχωρηθούν οι τιμές του cursor, μια για κάθε στήλη που αποθηκεύει. Οι μεταβλητές αυτές θα έχουν τύπο δεδομένων ίδιο με τον τύπο της κάθε στήλης. Η γενική δομή δήλωσης και επεξεργασίας ενός cursor είναι η εξής:

```
DECLARE όνομα_cursor CURSOR --δήλωση cursor
     FOR εντολή select     --επιλογή των επιθυμητών πλειάδων

OPEN όνομα_cursor       --εκτέλεση και γέμισμα του cursor με δεδομένα

FETCH (NEXT FROM) όνομα_cursor INTO <μεταβλητές>
/* φέρνει τις γραμμές (μια τη φορά) και τοποθετεί τις τιμές από κάθε
στήλη στην αντίστοιχη μεταβλητή*/

WHILE @@FETCH_STATUS = 0 --έλεγχος για το τέλος του cursor
Begin
<εντολές της transact-SQL>
FETCH (NEXT FROM) όνομα_cursor INTO <μεταβλητές>
End
CLOSE όνομα_cursor     --κλείσιμο του cursor
DEALLOCATE όνομα_cursor  /*απελευθέρωση της μνήμης που είχε διατεθεί
στον cursor με τη δήλωσή του*/
```

Παράδειγμα 6.1

Να δημιουργηθεί ένα μπλοκ κώδικα που να εμφανίζει τον αριθμό (**pnumber**), το όνομα (**pname**) και την περιοχή (**plocation**) των project του τμήματος με αριθμό (**dnumber**) = 4, με τη χρήση cursor και της εντολής print.

```
DECLARE  @pnumber varchar(30), @pname varchar(30), @plocation
varchar(20)
/*δήλωση των μεταβλητών στις οποίες θα καταχωρείται για κάθε γραμμή η
τιμή του αντίστοιχου πεδίου*/

DECLARE print_projects CURSOR
     FOR select pnumber, pname, plocation from project
     where dnumber = 4
--δήλωση cursor και καθορισμός των στηλών που θα περιέχει
OPEN print_projects --εκτέλεση cursor

FETCH print_projects INTO  @pnumber,  @pname, @plocation
/*καταχώρηση των τιμών της κάθε στήλης, για κάθε γραμμή του cursor,
στην αντίστοιχη μεταβλητή*/

WHILE @@FETCH_STATUS=0 --έλεγχος για το τέλος του cursor
Begin
print @pnumber
```

```
print @pname
print @plocation --εμφάνιση των τιμών των μεταβλητών
FETCH print_projects INTO  @pnumber,  @pname, @plocation
End

CLOSE print_projects    --κλείσιμο cursor
DEALLOCATE print_projects  --απελευθέρωση των πόρων του cursor
```

Στο παραπάνω παράδειγμα, μετά το **open,** στον cursor αποθηκεύεται το αποτέλεσμα της εντολής select. Δηλαδή τα περιεχόμενα του cursor θα είναι τα εξής:

3	Proothisi proiontwn	Arta
4	Proothisi proiontwn	Giannena
7	Diafimisi proiontwn	Giannena

Με το **Fetch** επιλέγεται με τη σειρά κάθε γραμμή του cursor. Το περιεχόμενο της κάθε στήλης αποθηκεύεται στις μεταβλητές, και μέσω των μεταβλητών, με τη χρήση της print εμφανίζεται στην οθόνη. Αυτό επαναλαμβάνεται μέχρι να τελειώσουν οι γραμμές του cursor, δηλαδή όσο η τιμή της **@@FETCH_STATUS=**0. Η μεταβλητή **@@FETCH_STATUS** είναι καθολική. Παρέχεται από τον sql server και αναπαριστά την κατάσταση του τελευταίου **Fetch.** Όσο η τιμή της είναι 0 σημαίνει ότι το τελευταίο **Fetch** εκτελέστηκε επιτυχώς.

Έτσι μετά το πρώτο fetch το περιεχόμενο των μεταβλητών είναι: `@pnumber=3, @pname=Proothisi proiontwn, @plocation=Arta.`

Το αποτέλεσμα του παραπάνω μπλοκ κώδικα είναι το εξής:

```
 3
Proothisi proiontwn
Arta

4
Proothisi proiontwn
Giannena

7
Diafimisi proiontwn
Giannena
```

7. Συναρτήσεις –Functions της t-sql που ορίζονται από το χρήστη

Η transact-SQL δίνει τη δυνατότητα στο χρήστη να δημιουργήσει functions με παρόμοιο τρόπο όπως τις procedures. Τυπικά οι functions εκτελούν κάποιες εντολές και επιστρέφουν μια τιμή ως αποτέλεσμα των εντολών αυτών. Η γενική δομή μιας function είναι η εξής:

```
CREATE FUNCTION όνομα_function
         <δηλώσεις παραμέτρων-προαιρετικά >

RETURNS return_data_type   --τύπος δεδομένων που θα επιστρέψει
    AS
    BEGIN
        <Σώμα συνάρτησης-εντολές>
        RETURN return_expression
    END
```

Η πρόταση **RETURNS** ορίζει τον τύπο δεδομένων της τιμής που θα επιστρέφεται από την function.

Παράδειγμα 7.1

Να δημιουργήσετε function με όνομα decrease_ salary η οποία να υπολογίζει το νέο μισθό που προκύπτει μετά από μια μείωση μισθού. Η συνάρτηση θα έχει ως παράμετρο τον αριθμό ταυτότητας (@emp_id).

```
CREATE FUNCTION decrease_salary
(@emp_id as char(8))
RETURNS decimal(10,5)
AS
BEGIN
DECLARE @new_salary decimal, @old_salary decimal
set @old_salary = (select salary from employee where emp_id=@emp_id)
set @new_salary = @old_salary - @old_salary*0.1
RETURN (@new_salary)
END

execute  decrease_salary @emp_id='B276548'
```

Παράρτημα Α: Τύποι δεδομένων και οι περιγραφές τους.

Data types	Description
bit	Integer data with either a 1 or 0 value.
int	Integer data from -2^31 through 2^31–1.
smallint	Integer data from 2^15 through 2^15–1.
bigint	Integer data from -2^63 through 2^63–1.
tinyint	Integer data from 0 through 255.
decimal	Fixed precision and scale numeric data from -10^38-1 through 10^38-1.
numeric	Same as decimal (synonym).
money	Monetary data values from -2^63 through 2^63-1, with accuracy to a ten-thousandth of a monetary unit.
smallmoney	Monetary data values from -214,748.3648 through +214,748.3647, with accuracy to a ten-thousandth of a monetary unit.
float	Floating precision number data from -1.79E+308 through 1.79E+308.
real	Floating precision number data from -3.40E+38 through 3.40E+38.
datetime	Date and time data from January 1, 1753, to December 31, 9999, with an accuracy of 3.33 milliseconds.
smalldatetime	Date and time data from January 1, 1900, through June 6, 2079, with an accuracy of one minute.
cursor	A reference to a cursor.
timestamp	A database-wide unique number.
uniqueidentifier	A globally unique identifier (GUID).
char	Non-Unicode character data with a maximum length of 8,000 characters.
varchar	Non-Unicode data with a maximum of 8,000 characters.
text	Non-Unicode data with a maximum length of 2^31-1 characters.
nchar	Unicode data with a maximum length of 4,000 characters.
nvarchar	Unicode data with a maximum length of 4,000 characters.
ntext	Unicode data with a maximum length of 2^31-1 characters.
binary	Fixed length binary data with a maximum length of 8,000 bytes.
varbinary	Binary data with a maximum length of 8,000 bytes.
image	Binary data with a maximum length of 2^31–1 bytes.
xml	Variable for storing XML data (column instances or xml type).
sql_variant	Various SQL Server data types excluding text, ntext, image, time_stamp, sql_variant.