

ΕΡΓΑΣΙΑ1 ΜΕΡΟΣ-B

Το SQL script που διοχετεύει τα δεδομένα στους πίνακες φαίνεται παρακάτω.

```

insert into Staff (staffNo, staffName, staffSurname)
select distinct staffNo, staffName, staffSurname
from MainTable

insert into Customer (custCode, custName, custSurname, custPhone)
select distinct custCode, custName, custSurname, custPhone
from MainTable

insert into Camping (campCode, campName, numOfEmp)
select distinct campCode, campName, numOfEmp
from MainTable

insert into Category (catCode, areaM2, unitCost)
select distinct catCode, areaM2, unitCost
from MainTable

insert into Emplacement (campCode, empNo, catCode)
select distinct campCode, empNo, catCode
from MainTable

insert into Payment (payCode, payMethod)
select distinct payCode, payMethod
from MainTable

insert into Booking (bookCode, bookDt, payCode, custCode, staffNo)
select distinct bookCode, bookDt, payCode, custCode, staffNo
from MainTable

insert into Rental (bookCode, campCode, empNo, startDt, endDt, noPers)
select distinct bookCode, campCode, empNo, startDt, endDt, noPers
from MainTable

```

Τα SQL queries που απαντούν στα ερωτήματα του ζητήματος Νο4 φαίνονται παρακάτω.

```

--(a)
select payMethod, count(bookCode) as totalReservations
from Payment, Booking
where Payment.payCode = Booking.payCode
group by payMethod

--(b)
select staffSurname, staffName, count(bookCode) as totalReservations
from Staff, Booking
where Staff.staffNo = Booking.staffNo
group by staffSurname, staffName
having count(bookCode) >= All (select count(bookCode)
                                from Staff, Booking
                                where Staff.staffNo = Booking.staffNo
                                group by staffSurname, staffName)

--(c)
select count(distinct R.bookCode) as totalReservations
from Rental as R, Emplacement, Category
where R.campCode = Emplacement.campCode and R.empNo = Emplacement.empNo and
      Emplacement.catCode = Category.catCode and Category.catCode = 'A' and
      Category.catCode = All(select Category.catCode
                              from Rental, Emplacement, Category
                              where R.bookCode = bookCode and Rental.campCode = Emplacement.campCode and
                                    Rental.empNo = Emplacement.empNo and Emplacement.catCode = Category.catCode)

```

```
-- (d)
select custSurname, custName, count(bookCode) as totalReservations
  from Customer, Booking
 where Customer.custCode = Booking.custCode and bookDt between '2000-01-01' and '2000-12-31'
 group by custSurname, custName
 order by custSurname

-- (e)
select campName, sum(unitCost*(DATEDIFF(day, startDt, endDt)+1)*noPers) as totalRevenue
  from Rental, Emplacement, Category, Camping
 where Rental.campCode = Emplacement.campCode and
        Rental.empNo = Emplacement.empNo and
        Emplacement.catCode = Category.catCode and
        Emplacement.campCode = Camping.campCode
 group by campName
```

Οι εντολές δημιουργίας των δεικτών για την επιτάχυνση της εκτέλεσης των επερωτημάτων (d) και (e), καθώς επίσης και τα στοιχεία που αποδεικνύουν ότι οι δείκτες αυτοί καθιστούν αποδοτικότερη την εκτέλεση των συγκεκριμένων επερωτημάτων φαίνονται παρακάτω.

δείκτης για την επιτάχυνση του επερωτήματος (d)

```
create index indx_booking on Booking(bookDt) include(custCode)
```

δείκτης για την επιτάχυνση του επερωτήματος (e)

```
create index indx_rental on Rental(campCode,empNo) include(endDt,noPers)
```

Χρόνος εκτέλεσης του επερωτήματος (d), πριν τη δημιουργία του δείκτη

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure, with the 'dbo.Booking' table and its 'PK\_Booking' index highlighted. The main window shows the execution of a query. The query is as follows:

```
1 SET STATISTICS TIME ON
2 select custSurname, custName, count(bookCode) as totalReservations
3   from Customer, Booking
4  where Customer.custCode = Booking.custCode and bookDt between '2000-01-01' and '2000-12-31'
5         group by custSurname, custName
6         order by custSurname
7 SET STATISTICS TIME OFF
```

The results pane shows that 4839 rows were affected. The SQL Server Execution Times are displayed in a red box:

```
SQL Server Execution Times:
  CPU time = 528 ms,  elapsed time = 904 ms.
```

The status bar at the bottom indicates that the query was executed successfully, taking 00:00:01 to complete, and affecting 4839 rows.

Χρόνος εκτέλεσης του επερωτήματος (d), μετά τη δημιουργία του δείκτη

The screenshot shows the same SQL Server Enterprise Manager interface, but now the 'indx\_booking' index is visible in the Object Explorer under the 'dbo.Booking' table. The same query is executed, and the results are shown. The SQL Server Execution Times are displayed in a red box:

```
SQL Server Execution Times:
  CPU time = 63 ms,  elapsed time = 196 ms.
```

The status bar at the bottom indicates that the query was executed successfully, taking 00:00:00 to complete, and affecting 4839 rows.

Χρόνος εκτέλεσης του ερωτήματος (ε), πριν τη δημιουργία του δείκτη

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the Object Explorer displays the database structure, with **dbo.Rental** and its primary key **PK\_Rental** highlighted. The main pane shows a SQL query in the SQL Query Editor:

```
1 SET STATISTICS TIME ON
2 select campName, sum(unitCost*(DATEDIFF(day, startDt, endDt)+1)*noPers) as totalRevenue
3   from Rental, Emplacement, Category, Camping
4   where Rental.campCode = Emplacement.campCode and
5         Rental.empNo = Emplacement.empNo and
6         Emplacement.catCode = Category.catCode and
7         Emplacement.campCode = Camping.campCode
8   group by campName
9 SET STATISTICS TIME OFF
```

The Results pane shows the execution time:

```
SQL Server Execution Times:
  CPU time = 5216 ms,  elapsed time = 2057 ms.
```

The status bar at the bottom indicates "Query executed successfully." and "5 rows" were returned.

Χρόνος εκτέλεσης του ερωτήματος (ε), μετά τη δημιουργία του δείκτη

The screenshot shows the same SQL Server Enterprise Manager interface, but now an index **indx\_rental** has been created on the **dbo.Rental** table. The Object Explorer shows **indx\_rental** under the **Indexes** folder. The SQL query in the SQL Query Editor is identical to the previous one:

```
1 SET STATISTICS TIME ON
2 select campName, sum(unitCost*(DATEDIFF(day, startDt, endDt)+1)*noPers) as totalRevenue
3   from Rental, Emplacement, Category, Camping
4   where Rental.campCode = Emplacement.campCode and
5         Rental.empNo = Emplacement.empNo and
6         Emplacement.catCode = Category.catCode and
7         Emplacement.campCode = Camping.campCode
8   group by campName
9 SET STATISTICS TIME OFF
```

The Results pane shows the execution time after the index is created:

```
SQL Server Execution Times:
  CPU time = 1922 ms,  elapsed time = 1560 ms.
```

The status bar at the bottom indicates "Query executed successfully." and "5 rows" were returned.