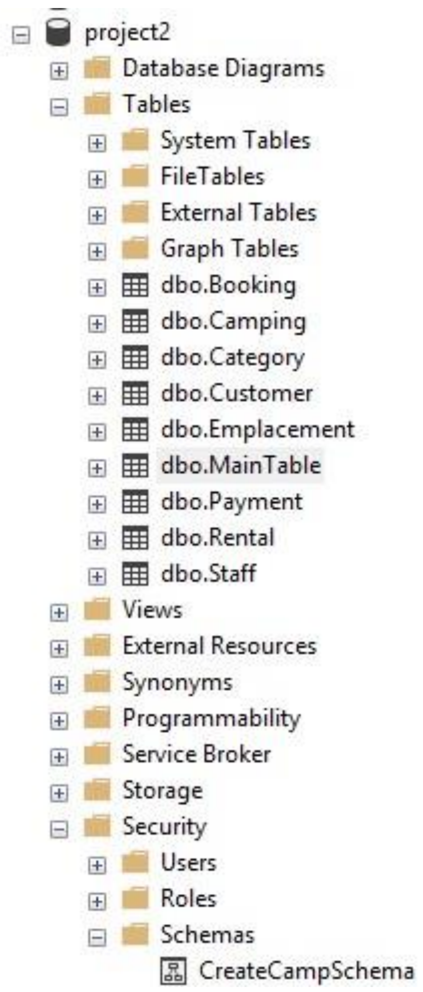
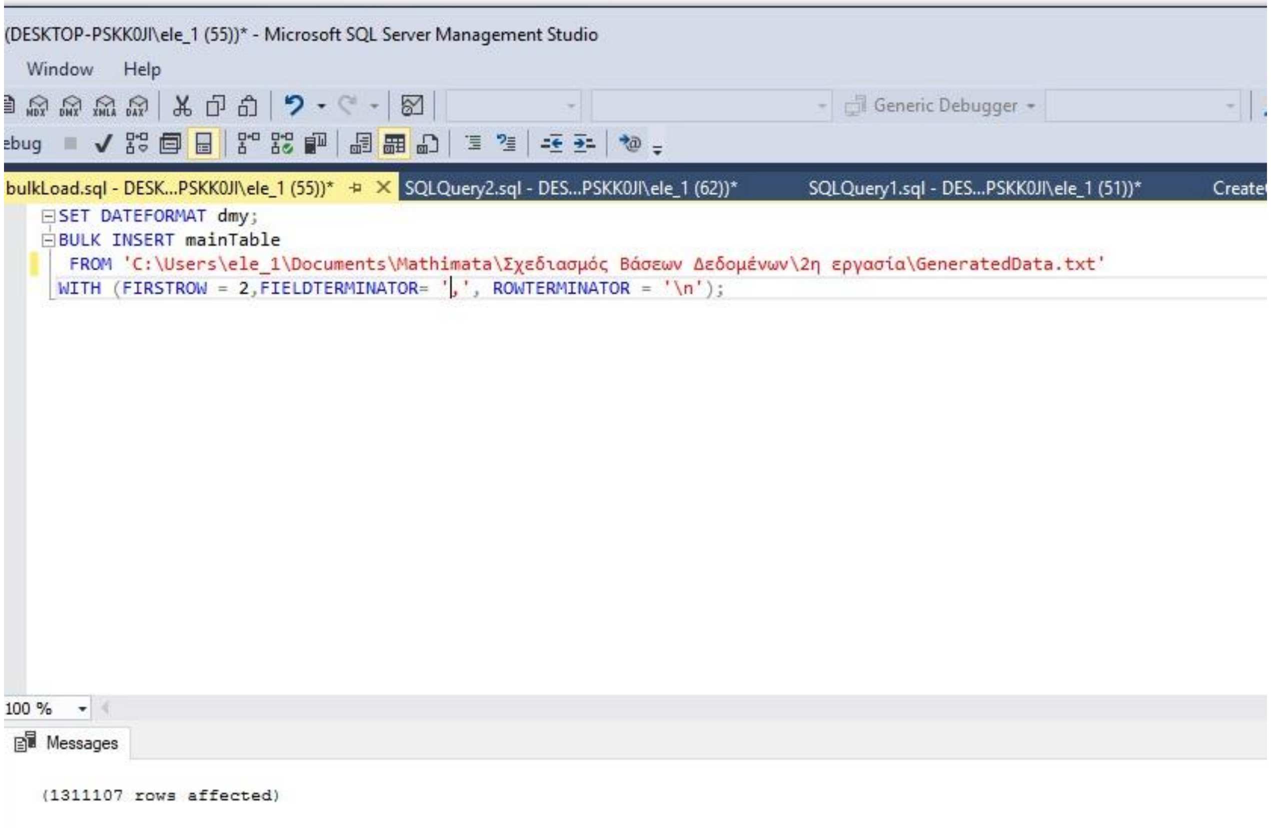
Question 1 - Question 2

For these questions I have included screenshots, showing the performance of the required steps.



```
project2
  Database Diagrams
  Tables
    System Tables
    FileTables
    External Tables
    Graph Tables
    dbo.Booking
    dbo.Camping
    dbo.Category
    dbo.Customer
    dbo.Emplacement
    dbo.MainTable
    dbo.Payment
    dbo.Rental
    dbo.Staff
  Views
  External Resources
  Synonyms
  Programmability
  Service Broker
  Storage
  Security
    Users
    Roles
    Schemas
      CreateCampSchema
```

Picture 1. Above we can see the creation of the given schema.


(DESKTOP-PSKK0JI\ele_1 (55))* - Microsoft SQL Server Management Studio
Window
Help
Generic Debugger
bulkLoad.sql - DESK...PSKK0JI\ele_1 (55))*
SQLQuery2.sql - DES...PSKK0JI\ele_1 (62))*
SQLQuery1.sql - DES...PSKK0JI\ele_1 (51))*
100 %
Messages

```
SET DATEFORMAT dmy;
BULK INSERT mainTable
 FROM 'C:\Users\ele_1\Documents\Mathimata\Σχεδιασμός Βάσεων Δεδομένων\2η εργασία\GeneratedData.txt'
WITH (FIRSTROW = 2,FIELDTERMINATOR= ',', ROWTERMINATOR = '\n');
```

```
(1311107 rows affected)
```

Figure 2. Above we can see the successful loading of data with the use of BulkLoad.sql script

question 3

Having loaded data in mainTable table, all the corresponding information of all the other tables must be updated automatically without violating the integrity. (I've generated a Script respectively, but it was impossible to include it here). So we have the following commands:

```
- Inserting data into Staff table
INSERT INTO project2.dbo.Staff(staffNo, staffName, staffSurname)SELECT DISTINCT staffNo,
staffName, staffSurname FROM project2.dbo.MainTable? - RICHTIG
--Select * from project2.dbo.Staff

--Inserting data into Payment table
INSERT INTO project2.dbo.Payment(payCode, payMethod)SELECT DISTINCT payCode, payMethod
FROM project2.dbo.MainTable?
--Select * from project2.dbo.Payment

--Inserting data into Customer table
INSERT INTO project2.dbo.Customer(custCode, custName, custSurname, custPhone)SELECT
DISTINCT custCode, custName, custSurname, custPhone FROM project2.dbo.MainTable?
--Select * from project2.dbo.Customer
--Select distinct custCode, custName, custSurname, custPhone from project2.dbo.MainTable
```

```
--Inserting data into Booking table
INSERT INTO project2.dbo.Booking(bookCode, bookDt, payCode, custCode, staffNo)SELECT
DISTINCT bookCode, bookDt, payCode, custCode, staffNo FROM project2.dbo.MainTable?
--Select * from project2.dbo.Booking
--Select * from project2.dbo.MainTable

--Inserting data into Camping table
INSERT INTO project2.dbo.Camping(campCode, campName, numOfEmp)SELECT DISTINCT campCode,
campName, numOfEmp FROM project2.dbo.MainTable?
--Select * from project2.dbo.Camping

--Inserting data into Category table
INSERT INTO project2.dbo.Category(catCode, areaM2, unitCost)SELECT DISTINCT catCode,
areaM2, unitCost FROM project2.dbo.MainTable?
--Select * from project2.dbo.Category

--Inserting data into Emplacement table
INSERT INTO project2.dbo.Emplacement(campCode, empNo, catCode)SELECT DISTINCT campCode,
empNo, catCode FROM project2.dbo.MainTable?
--Select * from project2.dbo.Emplacement

--Inserting data into Rental table
INSERT INTO project2.dbo.Rental(bookCode, campCode, empNo, startDt, endDt, noPers)SELECT
DISTINCT bookCode, campCode, empNo, startDt, endDt, noPers FROM project2.dbo.MainTable?
--Select * from project2.dbo.Rental
```

The commands in the comments sections help us confirm that the results are correct.

Question 4

For each question listed there is a corresponding command. There are clearly alternative ways to perform these and probably more efficient, however that was not requested.

a. Display the total number of reservations per payment.

```
--a
 SELECT COUNT(bookCode) payMethod
FROM project2.dbo.Booking AS B, project2.dbo.Payment AS P
WHERE B.payCode= P.payCode
GROUP BY payMethod?
/ * Or like that --a
SELECT COUNT (DISTINCT bookCode) AS totalBookingsPerCategory
 FROM project2.dbo.Booking
 GROUP BY payCode
* /
```

b. Display the name of the official who handled most reservations. Next to the name to be displayed and the number of bookings handled (to take into account the case where more than one staff handled most reservations).

```
--b
```

```
SELECT COUNT(*) as HighestTotalRentals, project2.dbo.Staff.staffName,
project2.dbo.Staff.staffSurname
FROM project2.dbo.Booking
INNER JOIN project2.dbo.Staff
ON project2.dbo.Booking.staffNo = project2.dbo.Staff.staffNo
GROUP BY project2.dbo.Staff.staffName, project2.dbo.Staff.staffSurname
HAVING COUNT(*) = (SELECT MAX (y.c) AS m FROM (SELECT COUNT(*) AS c, B.staffNo
FROM project2.dbo.Booking AS B
GROUP BY B.staffNo)y)?
```

c. Display the total number of bookings containing only class positions "A" one or more camps.

```
--c
SELECT COUNT (*) AS totalBookingswithCatA
FROM project2.dbo.Booking
INNER JOIN project2.dbo.Rental
ON project2.dbo.Booking.bookCode = project2.dbo.Rental.bookCode
INNER JOIN project2.dbo.Emplacement
ON project2.dbo.Emplacement.campCode = project2.dbo.Rental.campCode
WHERE catCode ='A'
- Of course there can be a more efficient way to do the above as well.
```

d. Display a list with the name of each customer and the total number of reservations made in 2000. The list will be sorted by customer name.

```
--d
SELECT custSurname, custName, COUNT (*) AS totalBookingsIn2000
from project2.dbo.Customer
INNER JOIN project2.dbo.Booking
ON project2.dbo.Booking.custCode = project2.dbo.Customer.custCode
Where bookDt BETWEEN '2000-01-01' AND '12.31.2000'
GROUP BY custSurname, custName
ORDER BY custSurname
```

e. Display the total value of bookings (total revenue) per camp.

```
--e
SELECT campName, SUM((DATEDIFF(day, startDt, endDt) +1) * unitCost * noPers )AS totalCost

from project2.dbo.Booking, project2.dbo.Camping, project2.dbo.Rental,
project2.dbo.Emplacement, project2.dbo.Category
Where Booking.bookCode= Rental.bookCodeAND Camping.campCode= Rental.campCodeAND
Emplacement.campCode=Camping.campCodeAND Category.catCode=Emplacement.catCode
GROUP BY campName
```

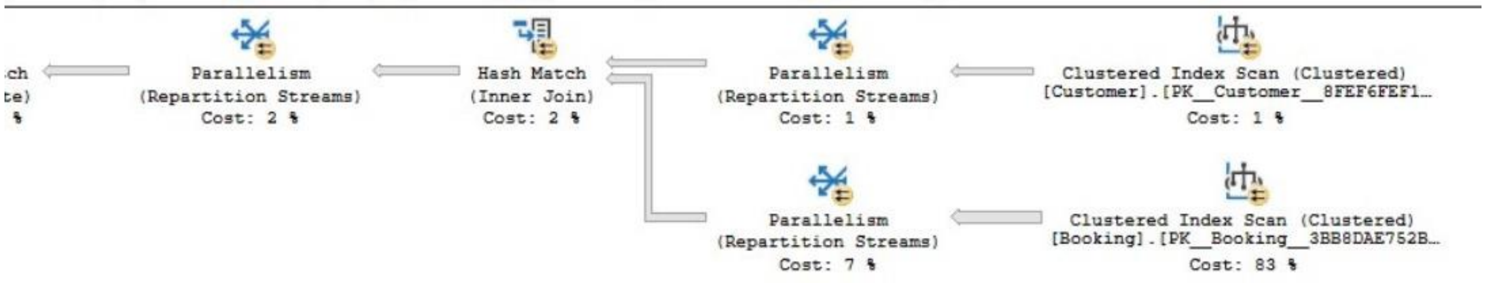
Question 5

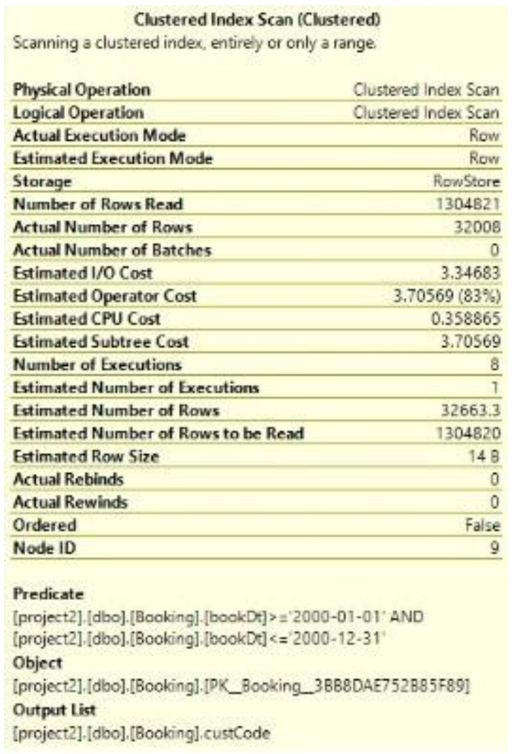
```
/ * The index helps us to do some functions faster. Before we proceed to create an index
must first follow the needs of our database and statistics providing our environment. For
example, in some cases-queries are efficient to have no index to other preferred
clustered index, while in other unclustered index etc.
       In this case, if these questions in our basically interested in making retrieval
will create indexes on columns used in the queries. Let say that the requested query four
commands are often used by the system. Thus, by making the following index and making
comparisons with the statistics we had without the existence of this I IN GENERAL lines
to see if it helps or not. However several times resulting non expected results, which of
course due
on factors such as how many times we ran the command etc. For example, sometimes I
happened to come off much better results with the index x, and next time do not be so
profitable existence of the index for the same query ...
       Should generally time to fall and fall and logical reads. Unfortunately, in
several different indexes tests run in a fall in the other climbed without any change in
the code.
* /
```

Generally various tests were made to find an appropriate index. Screenshots are included to show the procedure. The code of d. & e. were put into the following commands. “Statistics” were activated and several tools of MSSMS.

```
SET STATISTICS TIME ON

SET STATISTICS TIME OFF
```

d. I noticed, as shown below that the system consumes a lot of time in order to do the Clustered Index Scan of “Booking”, which led me to think that this is a point where we can improve on.





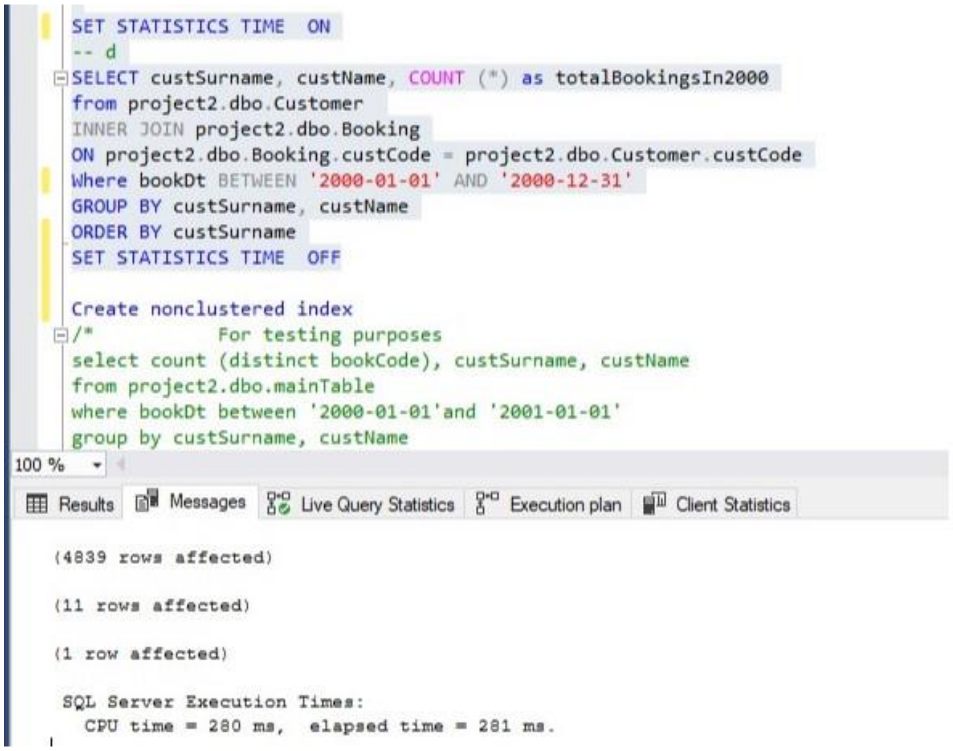
Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Number of Rows Read	1304821
Actual Number of Rows	32008
Actual Number of Batches	0
Estimated I/O Cost	3.34683
Estimated Operator Cost	3.70569 (83%)
Estimated CPU Cost	0.358865
Estimated Subtree Cost	3.70569
Number of Executions	8
Estimated Number of Executions	1
Estimated Number of Rows	32663.3
Estimated Number of Rows to be Read	1304820
Estimated Row Size	14 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	9

Predicate
[project2].[dbo].[Booking].[bookDt]>='2000-01-01' AND [project2].[dbo].[Booking].[bookDt]<='2000-12-31'
Object
[project2].[dbo].[Booking].[PK__Booking__3BB8DAE752B85F89]
Output List
[project2].[dbo].[Booking].custCode

Picture - Here various elements are shown on which we will rely on to reduce the time it takes for a query to be executed. We pay attention to the part of logical reads.



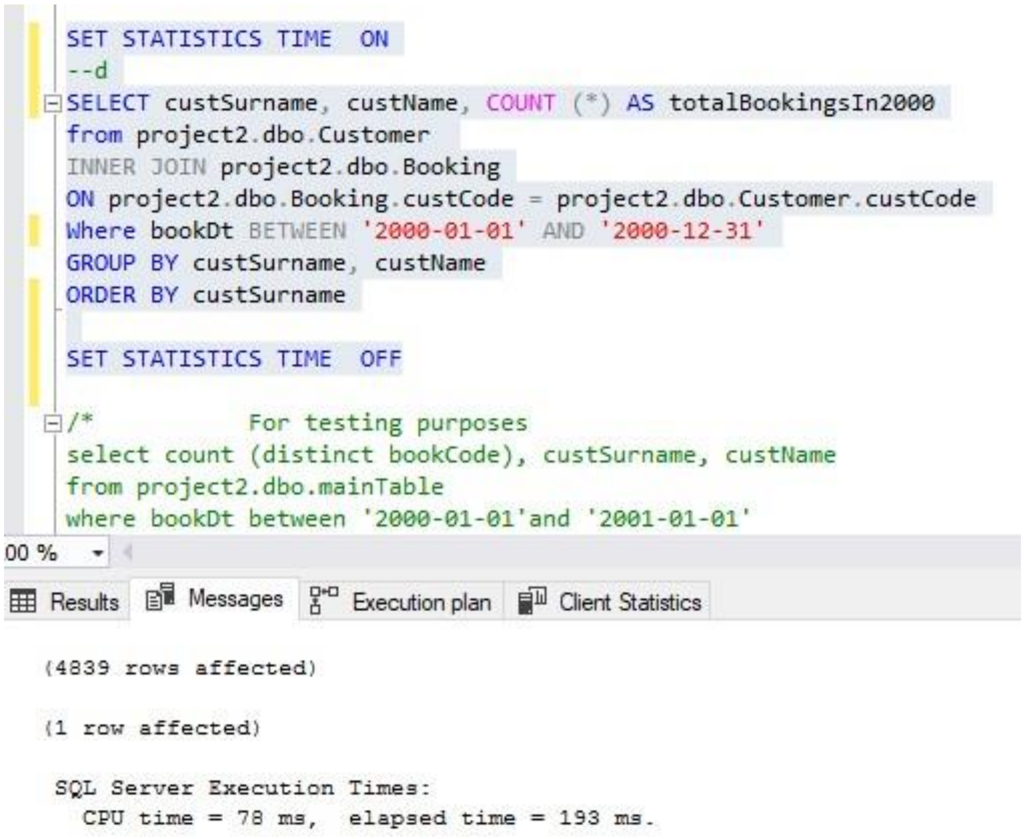
```
SET STATISTICS TIME  ON
-- d
SELECT custSurname, custName, COUNT (*) as totalBookingsIn2000
from project2.dbo.Customer
INNER JOIN project2.dbo.Booking
ON project2.dbo.Booking.custCode = project2.dbo.Customer.custCode
Where bookDt BETWEEN '2000-01-01' AND '2000-12-31'
GROUP BY custSurname, custName
ORDER BY custSurname
SET STATISTICS TIME  OFF

Create nonclustered index
/*            For testing purposes
select count (distinct bookCode), custSurname, custName
from project2.dbo.mainTable
where bookDt between '2000-01-01'and '2001-01-01'
group by custSurname, custName
```

```
(4839 rows affected)

(11 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 280 ms,  elapsed time = 281 ms.
```

Figure – The time consumed shown here.

For d. I used, in the end, the following index.

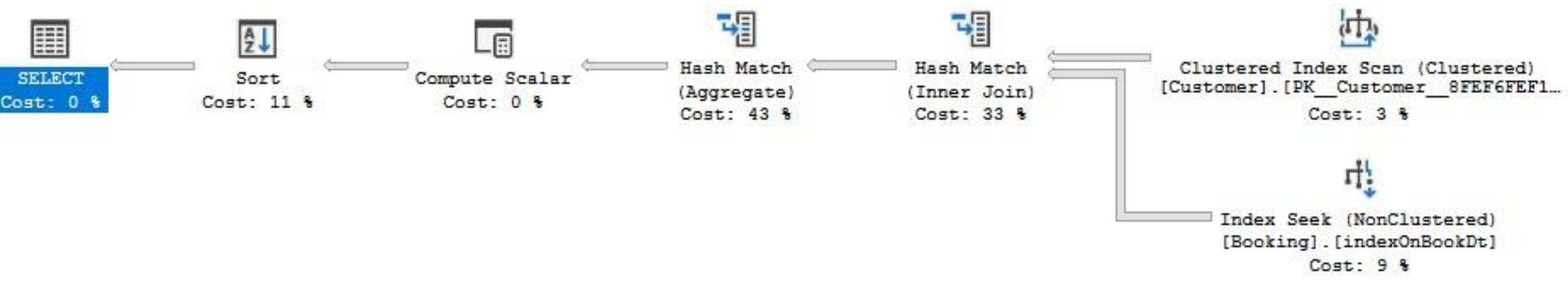
```
- Creating index for d
CREATE NONCLUSTERED INDEX indexOnBookDt ON project2.dbo.Booking (bookDt) INCLUDE
(custCode)?
```



```
SET STATISTICS TIME  ON
--d
SELECT custSurname, custName, COUNT (*) AS totalBookingsIn2000
from project2.dbo.Customer
INNER JOIN project2.dbo.Booking
ON project2.dbo.Booking.custCode = project2.dbo.Customer.custCode
Where bookDt BETWEEN '2000-01-01' AND '2000-12-31'
GROUP BY custSurname, custName
ORDER BY custSurname

SET STATISTICS TIME  OFF

/*          For testing purposes
select count (distinct bookCode), custSurname, custName
from project2.dbo.mainTable
where bookDt between '2000-01-01'and '2001-01-01'
```

```
(4839 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 78 ms,  elapsed time = 193 ms.
```

This shows the new statistics after adding the indicator (above) in our database.



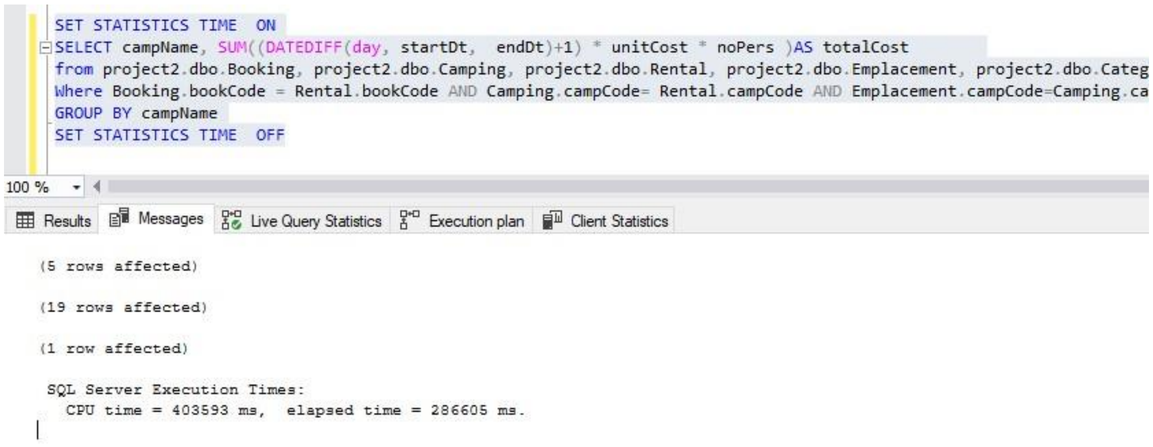
Above we can see the new plan, where we can see that our new index was used. ("Index Seek" instead of "Index Scan" by using the "indexOnBookDt".

e.

Similarly.

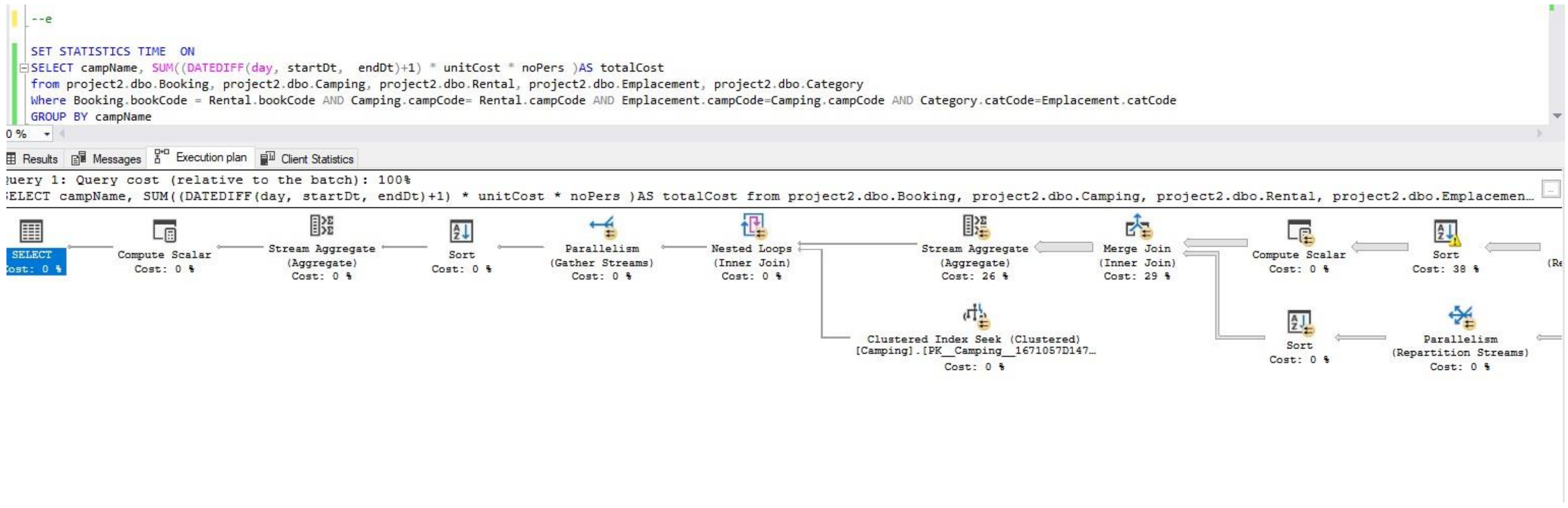
Here, unfortunately, the results had variations in terms of performance. Observing the statistics, we conclude that this part needs indexes to run faster.

```
Where Booking.bookCode= Rental.bookCodeAND Camping.campCode= Rental.campCodeAND
Emplacement.campCode=Camping.campCodeAND Category.catCode=Emplacement.catCode
```



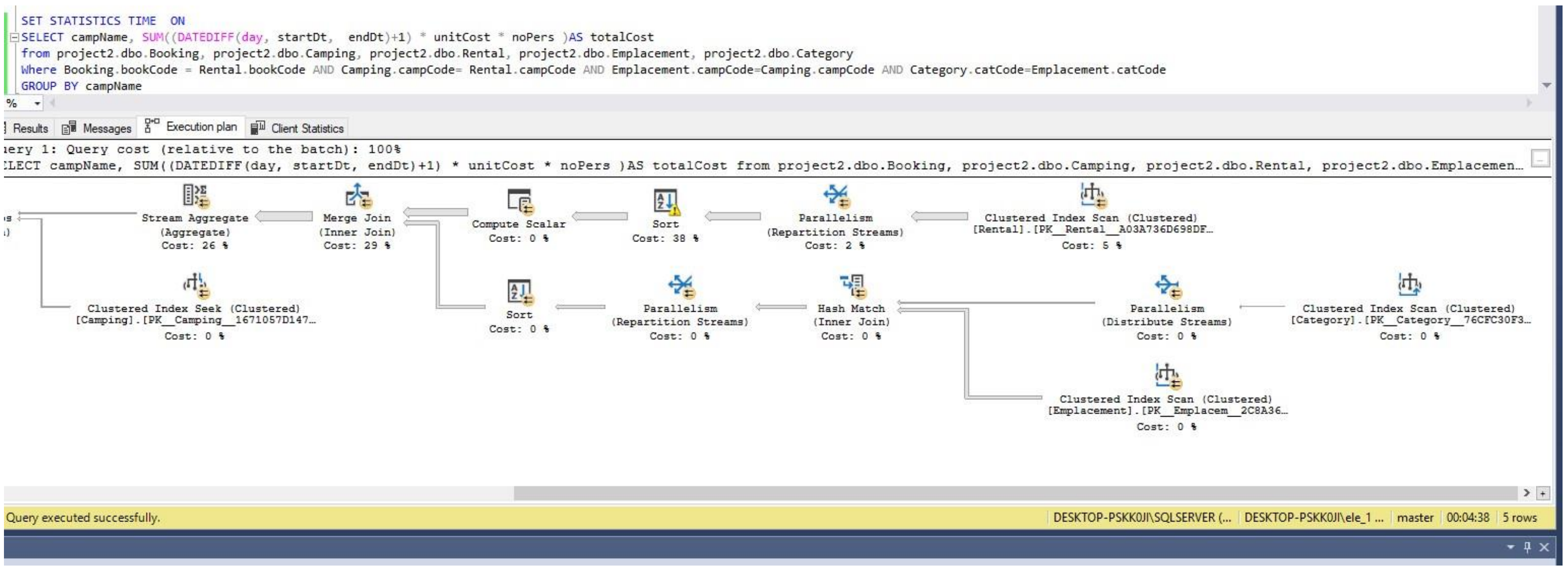
```
SET STATISTICS TIME  ON
SELECT campName, SUM((DATEDIFF(day, startDt,  endDt)+1) * unitCost * noPers )AS totalCost
from project2.dbo.Booking, project2.dbo.Camping, project2.dbo.Rental, project2.dbo.Emplacement, project2.dbo.Categ
Where Booking.bookCode = Rental.bookCode AND Camping.campCode= Rental.campCode AND Emplacement.campCode=Camping.ca
GROUP BY campName
SET STATISTICS TIME  OFF
```

```
(5 rows affected)

(19 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 403593 ms,  elapsed time = 286605 ms.
```



```
--e

SET STATISTICS TIME  ON
SELECT campName, SUM((DATEDIFF(day, startDt,  endDt)+1) * unitCost * noPers )AS totalCost
from project2.dbo.Booking, project2.dbo.Camping, project2.dbo.Rental, project2.dbo.Emplacement, project2.dbo.Category
Where Booking.bookCode = Rental.bookCode AND Camping.campCode= Rental.campCode AND Emplacement.campCode=Camping.campCode AND Category.catCode=Emplacement.catCode
GROUP BY campName
```

```
uery 1: Query cost (relative to the batch): 100%
ELECT campName, SUM((DATEDIFF(day, startDt, endDt)+1) * unitCost * noPers )AS totalCost from project2.dbo.Booking, project2.dbo.Camping, project2.dbo.Rental, project2.dbo.Emplacemen…
```



```
SET STATISTICS TIME  ON
SELECT campName, SUM((DATEDIFF(day, startDt,  endDt)+1) * unitCost * noPers )AS totalCost
from project2.dbo.Booking, project2.dbo.Camping, project2.dbo.Rental, project2.dbo.Emplacement, project2.dbo.Category
Where Booking.bookCode = Rental.bookCode AND Camping.campCode= Rental.campCode AND Emplacement.campCode=Camping.campCode AND Category.catCode=Emplacement.catCode
GROUP BY campName
```

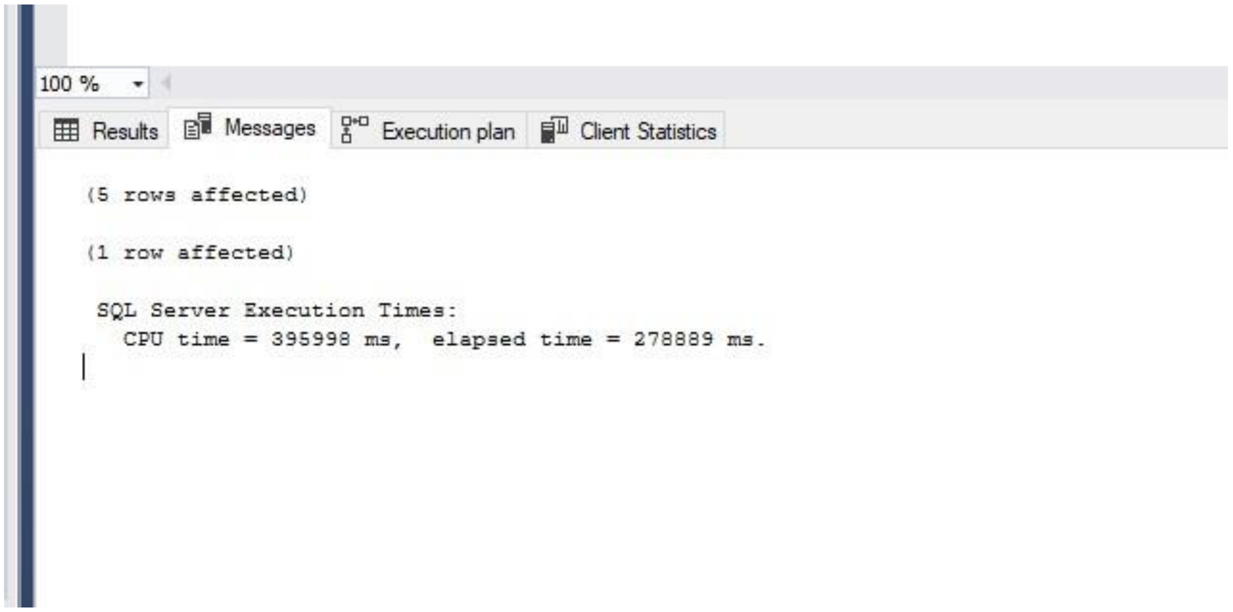
```
uery 1: Query cost (relative to the batch): 100%
LECT campName, SUM((DATEDIFF(day, startDt, endDt)+1) * unitCost * noPers )AS totalCost from project2.dbo.Booking, project2.dbo.Camping, project2.dbo.Rental, project2.dbo.Emplacemen…
```

Results
Messages
Execution plan
Client Statistics
Stream Aggregate (Aggregate) Cost: 26 %
Merge Join (Inner Join) Cost: 29 %
Compute Scalar Cost: 0 %
Sort Cost: 38 %
Parallelism (Repartition Streams) Cost: 2 %
Clustered Index Scan (Clustered) [Rental].[PK__Rental__A03A736D698DF… Cost: 5 %
Clustered Index Seek (Clustered) [Camping].[PK__Camping__1671057D147… Cost: 0 %
Sort Cost: 0 %
Parallelism (Repartition Streams) Cost: 0 %
Hash Match (Inner Join) Cost: 0 %
Parallelism (Distribute Streams) Cost: 0 %
Clustered Index Scan (Clustered) [Category].[PK__Category__76CFC30F3… Cost: 0 %
Clustered Index Scan (Clustered) [Emplacement].[PK__Emplacem__2C8A36… Cost: 0 %

```
Query executed successfully.    DESKTOP-PSKK0JI\SQLSERVER (...  DESKTOP-PSKK0JI\ele_1 ...  master  00:04:38  5 rows
```

I proceeded with the creation of the following indexes, which I used them alone, but also in pairs with the index of question d.

After a lot of effort, I managed to reach the following result, which is obviously way better than before.



```
(5 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 395998 ms,  elapsed time = 278889 ms.
```

```
CREATE NONCLUSTERED INDEX indexOnRental
ON project2.dbo.Rental (startDt, endDt)
INCLUDE (bookCode)?
```

```
CREATE NONCLUSTERED INDEX indexOnRentalCampCode
ON project2.dbo.Rental (campCode)
INCLUDE (bookCode)?

CREATE NONCLUSTERED INDEX indexOnEmpl
ON project2.dbo.Emplacement (campCode,empNo)
INCLUDE (catCode)?
```