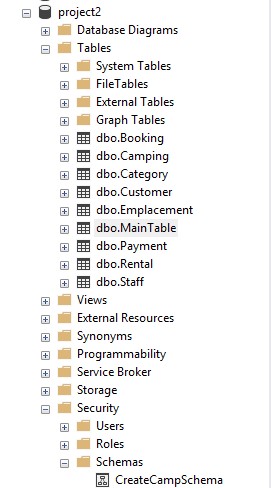
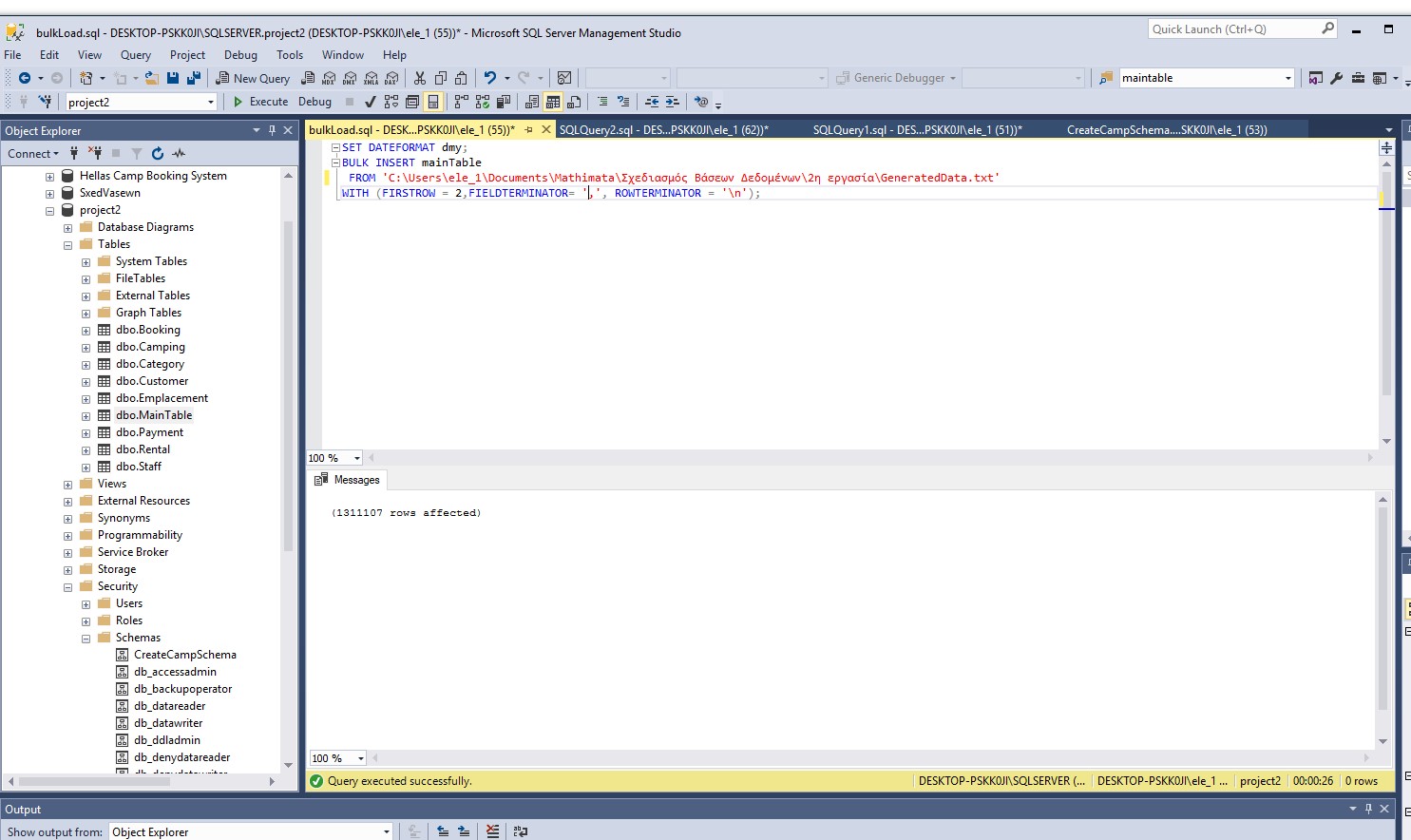
**Question 1 - Question 2**

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



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



**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.

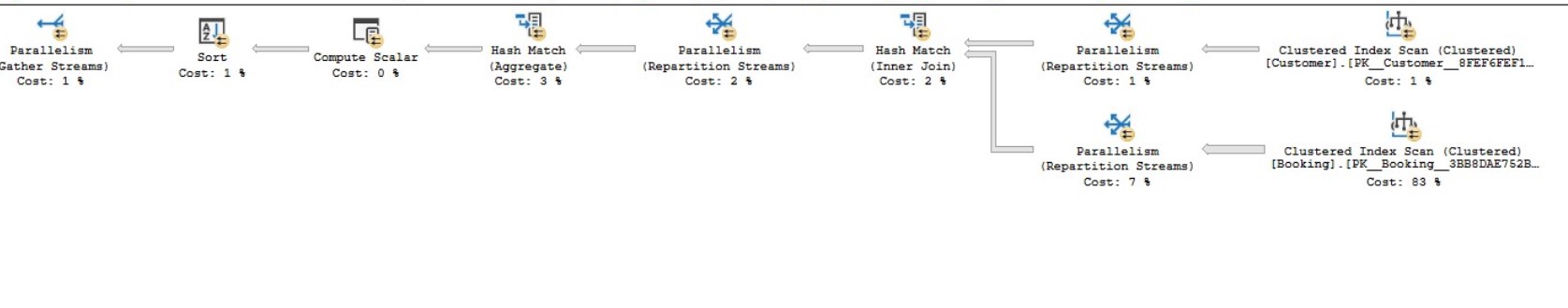
\* /

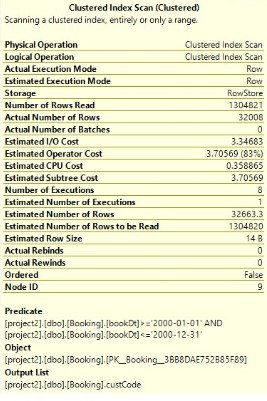
Generally various tests were made to find an appropriate index. Screenshots are included to show the procedure. The code of d. & e. were put into the following commands. “Statistics” were activated and several tools of MSSMS.

SET STATISTICS TIME ON

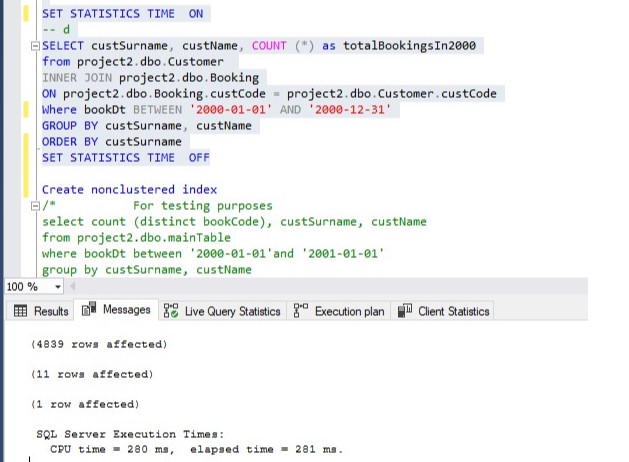
SET STATISTICS TIME OFF

**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.





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.

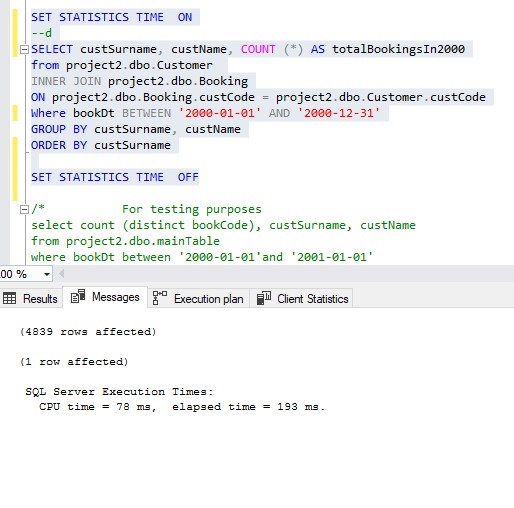


*Figure – The time consumed shown here.*

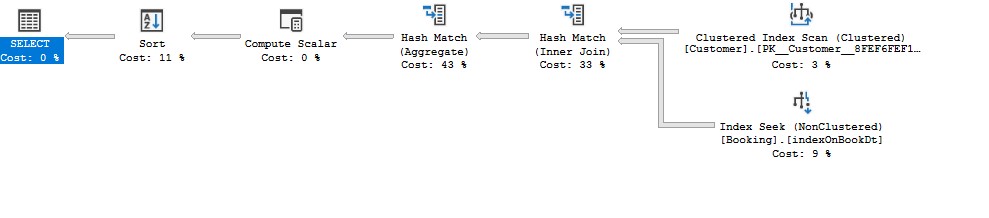
For d. I used, in the end, the following index.

- Creating index for d

CREATE NONCLUSTERED INDEX indexOnBookDt ON project2.dbo.Booking (bookDt) INCLUDE (custCode)?



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



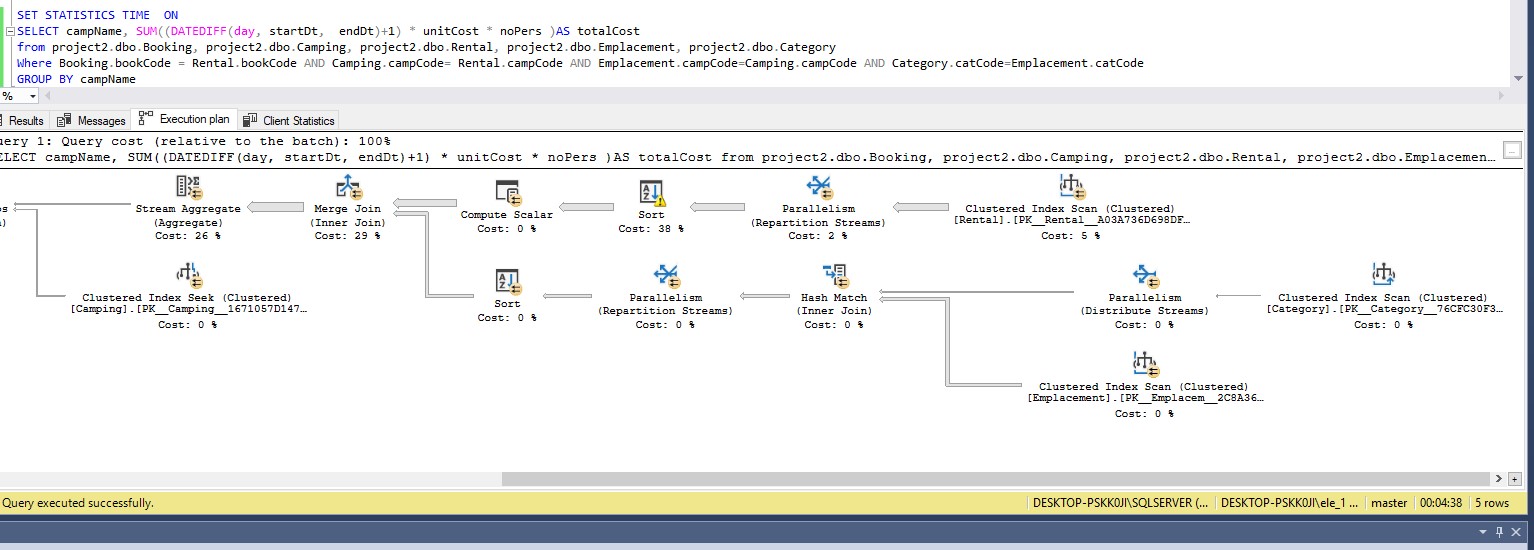
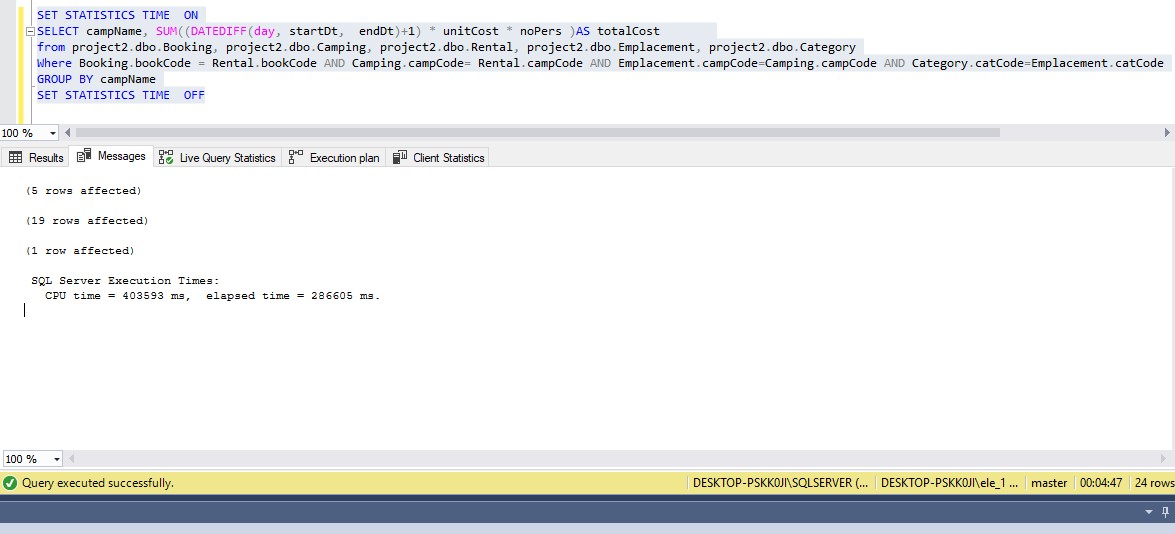
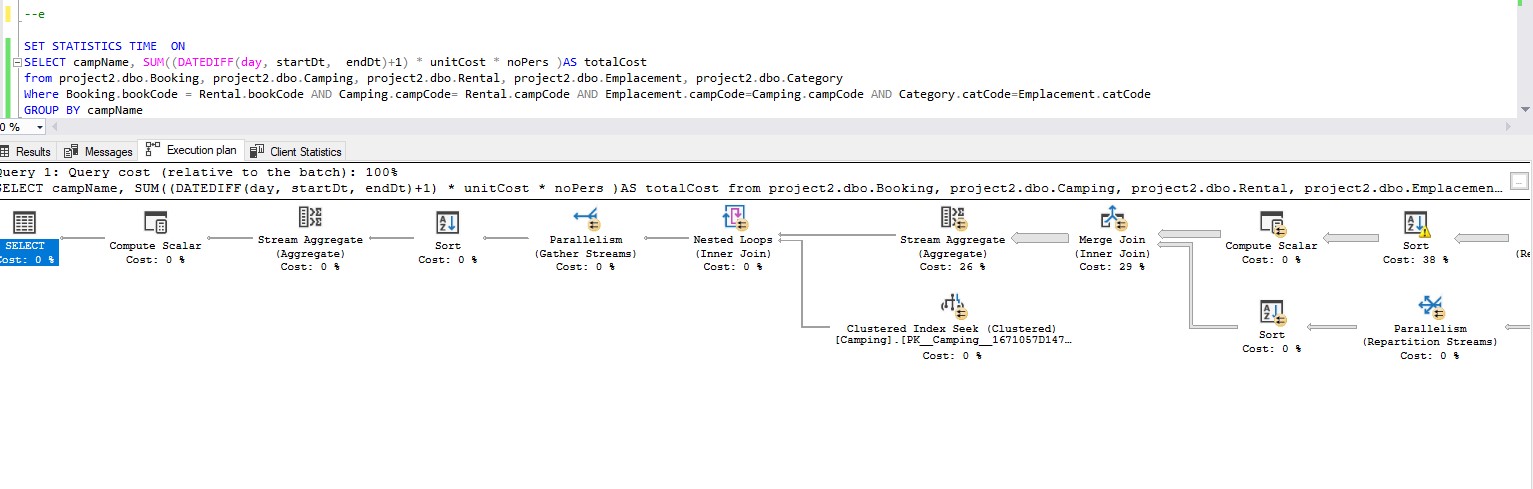
Above we can see the new plan, where we can see that our new index was used. (“Index Seek” instead of “Index Scan” by using the “indexOnBookDt”.

**e.**

Similarly.

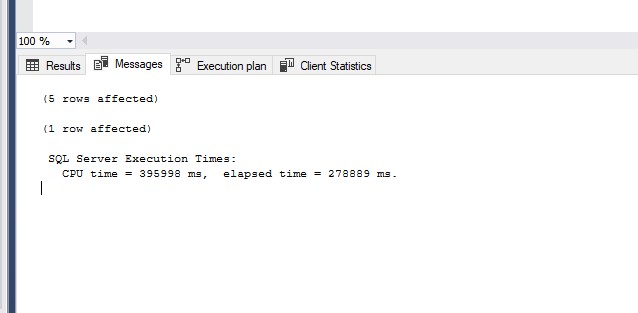
**Here, unfortunately, the results had variations in terms of performance. Observing the statistics, we conclude that this part needs indexes to run faster.**

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



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.



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)?