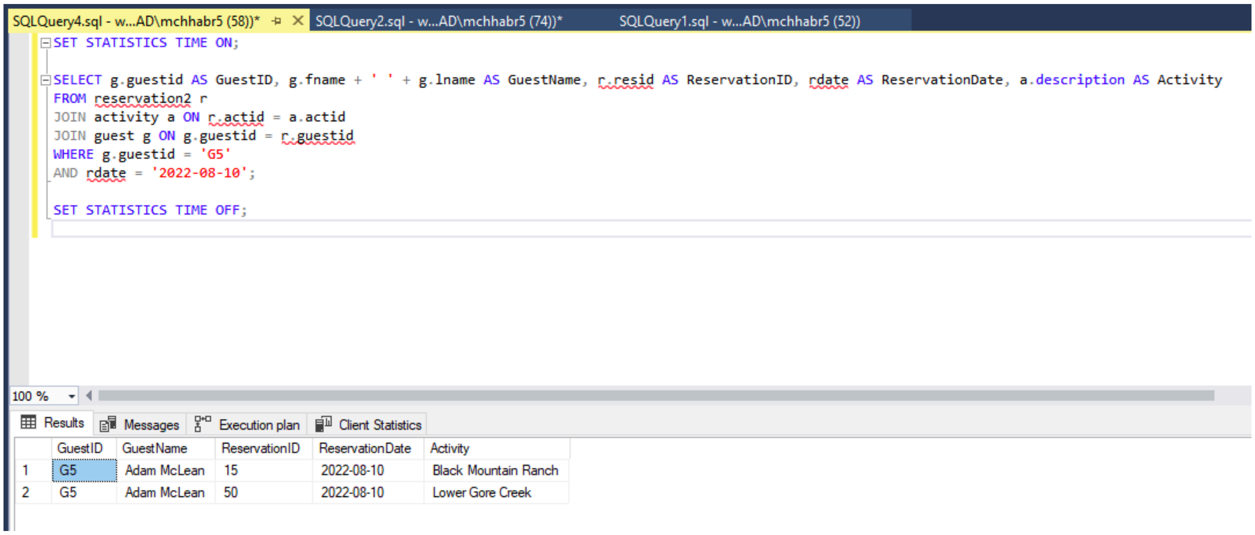
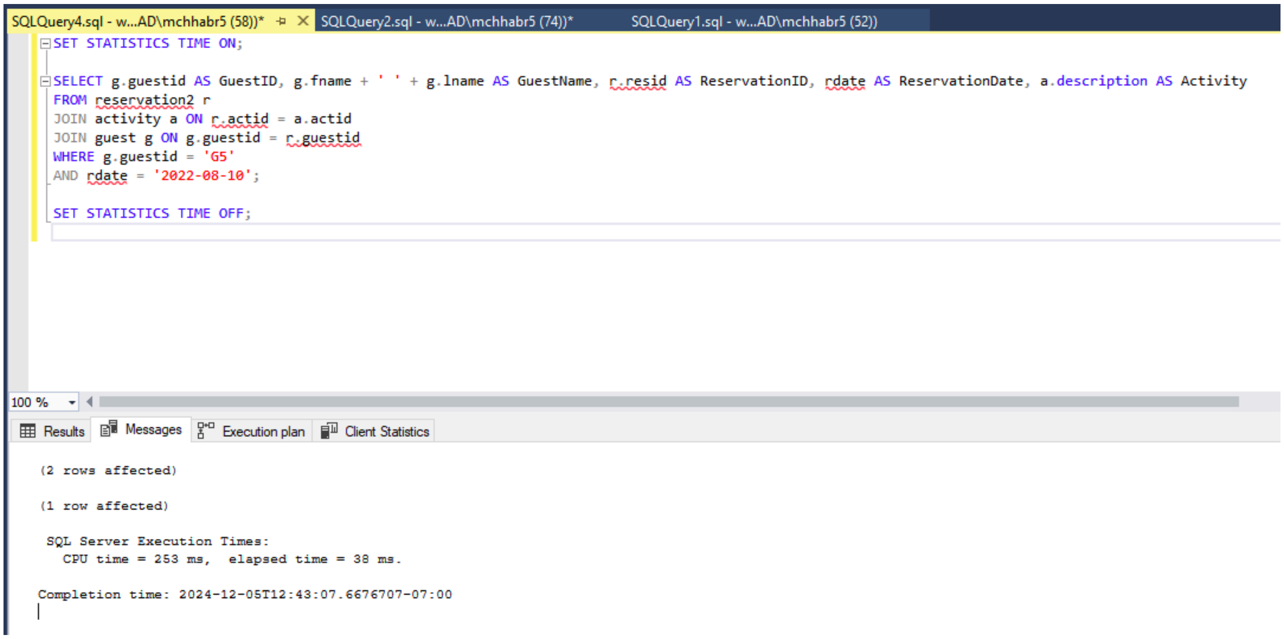
Question 1:


SQLQuery4.sql - w...AD\mchhabr5 (58))*
SQLQuery2.sql - w...AD\mchhabr5 (74))*
SQLQuery1.sql - w...AD\mchhabr5 (52))
100 %
Results
Messages
Execution plan
Client Statistics

```sql
SET STATISTICS TIME ON;

SELECT g.guestid AS GuestID, g.fname + ' ' + g.lname AS GuestName, r.resid AS ReservationID, rdate AS ReservationDate, a.description AS Activity
FROM reservation2 r
JOIN activity a ON r.actid = a.actid
JOIN guest g ON g.guestid = r.guestid
WHERE g.guestid = 'G5'
AND rdate = '2022-08-10';

SET STATISTICS TIME OFF;
```

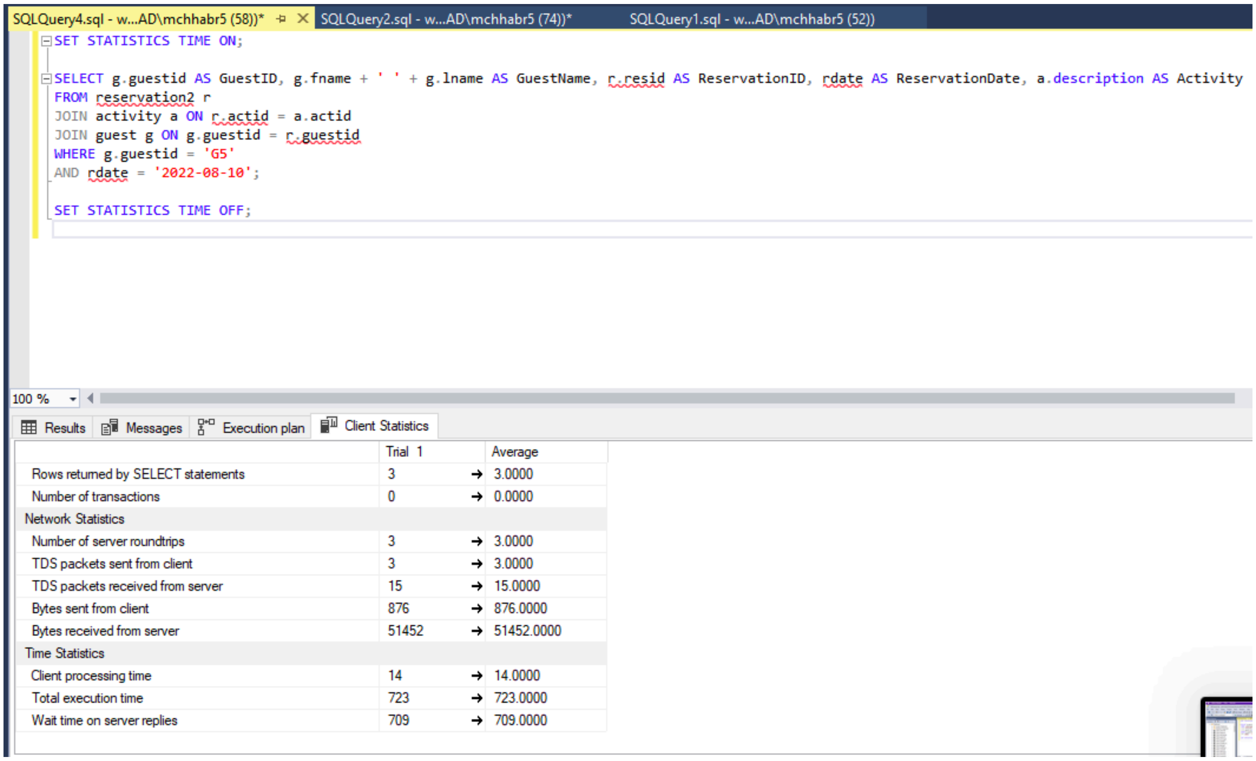
	GuestID	GuestName	ReservationID	ReservationDate	Activity
1	G5	Adam McLean	15	2022-08-10	Black Mountain Ranch
2	G5	Adam McLean	50	2022-08-10	Lower Gore Creek


SQLQuery4.sql - w...AD\mchhabr5 (58))*
SQLQuery2.sql - w...AD\mchhabr5 (74))*
SQLQuery1.sql - w...AD\mchhabr5 (52))
100 %
Results
Messages
Execution plan
Client Statistics

```sql
SET STATISTICS TIME ON;

SELECT g.guestid AS GuestID, g.fname + ' ' + g.lname AS GuestName, r.resid AS ReservationID, rdate AS ReservationDate, a.description AS Activity
FROM reservation2 r
JOIN activity a ON r.actid = a.actid
JOIN guest g ON g.guestid = r.guestid
WHERE g.guestid = 'G5'
AND rdate = '2022-08-10';

SET STATISTICS TIME OFF;
```

```
(2 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 253 ms,  elapsed time = 38 ms.

Completion time: 2024-12-05T12:43:07.6676707-07:00
```

Question 2:

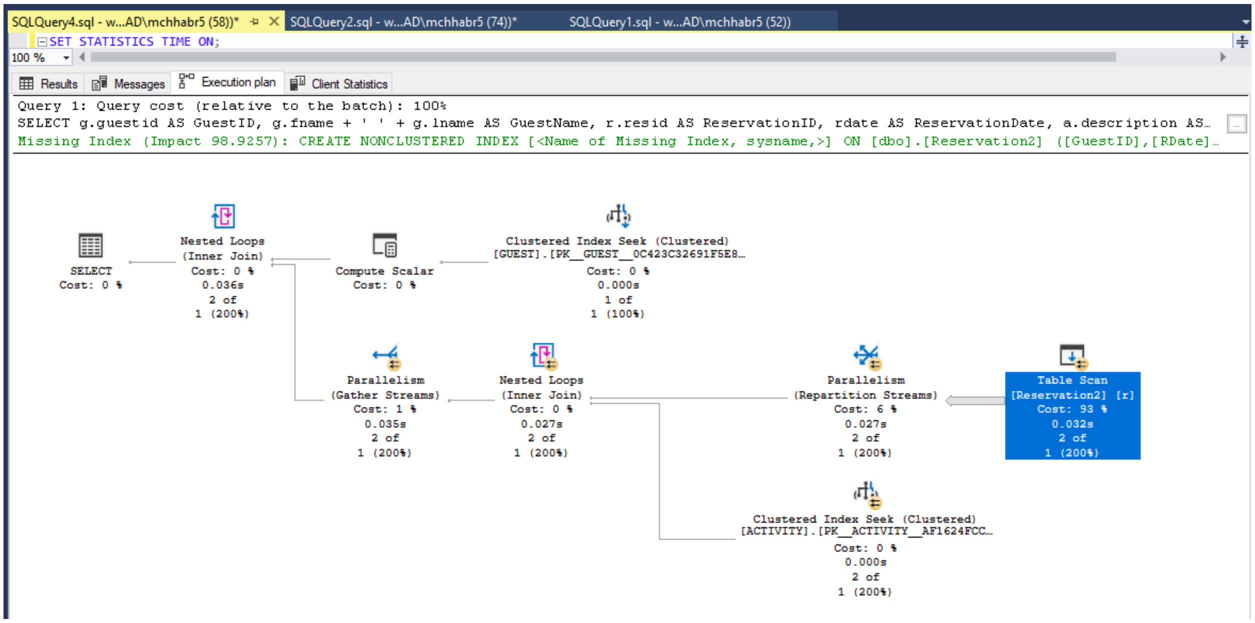


SQLQuery4.sql - w...AD\mchhabr5 (58))*
SQLQuery2.sql - w...AD\mchhabr5 (74))*
SQLQuery1.sql - w...AD\mchhabr5 (52))
100 %
Results
Messages
Execution plan
Client Statistics

```sql
SET STATISTICS TIME ON;

SELECT g.guestid AS GuestID, g.fname + ' ' + g.lname AS GuestName, r.resid AS ReservationID, rdate AS ReservationDate, a.description AS Activity
FROM reservation2 r
JOIN activity a ON r.actid = a.actid
JOIN guest g ON g.guestid = r.guestid
WHERE g.guestid = 'G5'
AND rdate = '2022-08-10';

SET STATISTICS TIME OFF;
```

	Trial 1		Average
Rows returned by SELECT statements	3	→	3.0000
Number of transactions	0	→	0.0000
Network Statistics			
Number of server roundtrips	3	→	3.0000
TDS packets sent from client	3	→	3.0000
TDS packets received from server	15	→	15.0000
Bytes sent from client	876	→	876.0000
Bytes received from server	51452	→	51452.0000
Time Statistics			
Client processing time	14	→	14.0000
Total execution time	723	→	723.0000
Wait time on server replies	709	→	709.0000

Question 3:

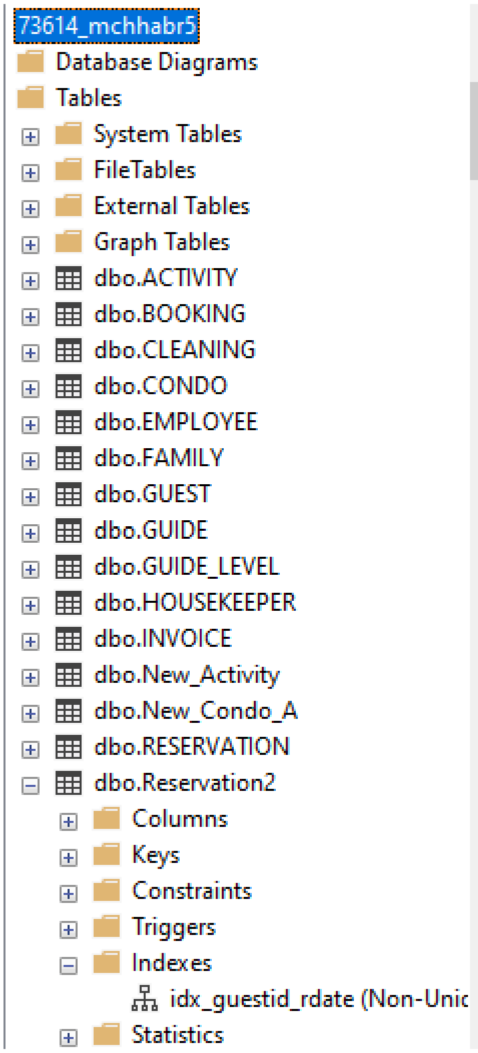


SQLQuery4.sql - w...AD\mchhabr5 (58))*
SQLQuery2.sql - w...AD\mchhabr5 (74))*
SQLQuery1.sql - w...AD\mchhabr5 (52))
SET STATISTICS TIME ON;
100 %
Results
Messages
Execution plan
Client Statistics

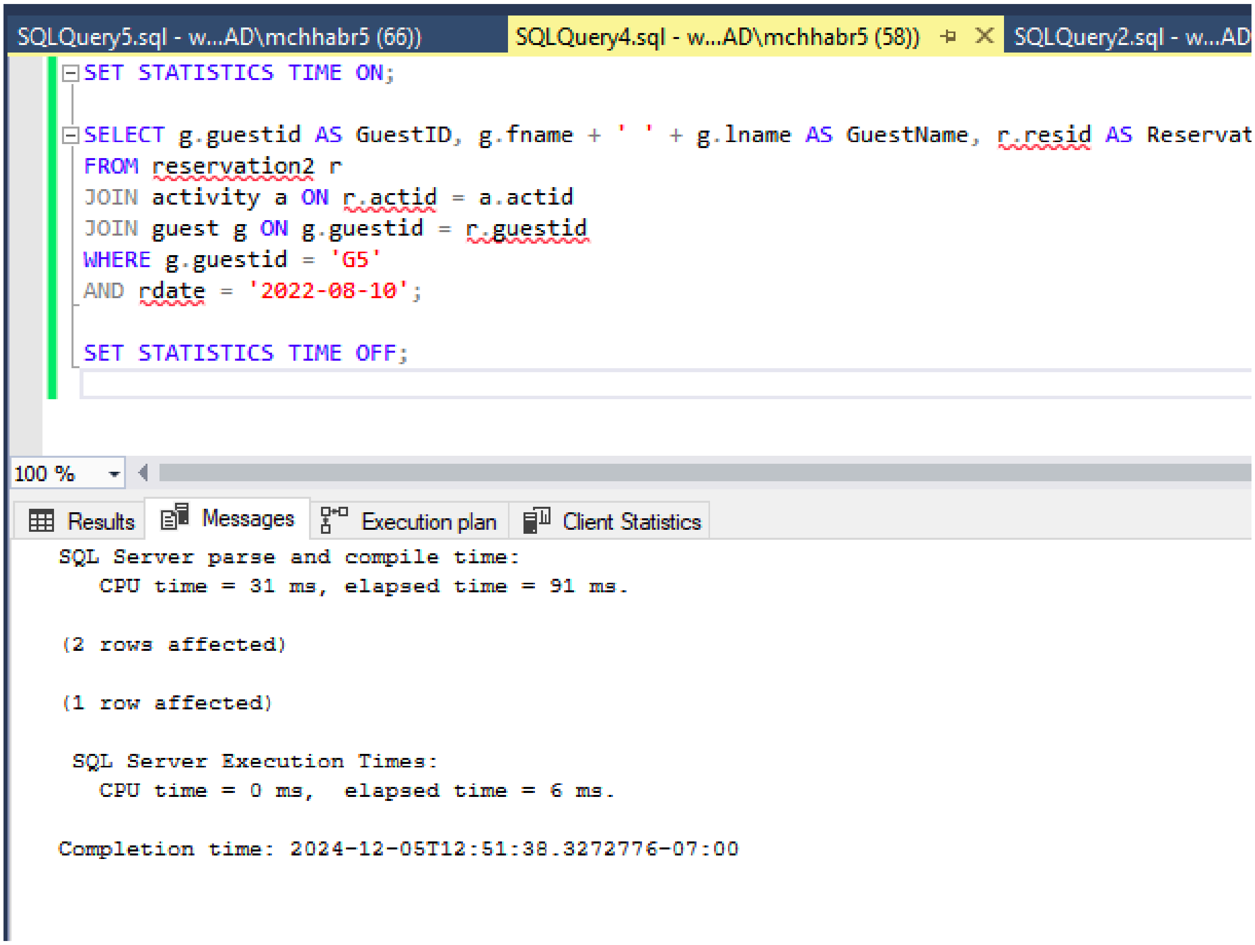
```
Query 1: Query cost (relative to the batch): 100%
SELECT g.guestid AS GuestID, g.fname + ' ' + g.lname AS GuestName, r.resid AS ReservationID, rdate AS ReservationDate, a.description AS…
Missing Index (Impact 98.9257): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [dbo].[Reservation2] ([GuestID],[RDate]…
```

Question 4:
Syntax:

```
CREATE NONCLUSTERED INDEX idx_guestid_rdate ON Reservation2 (guestid, rdate);
```



Question 5:

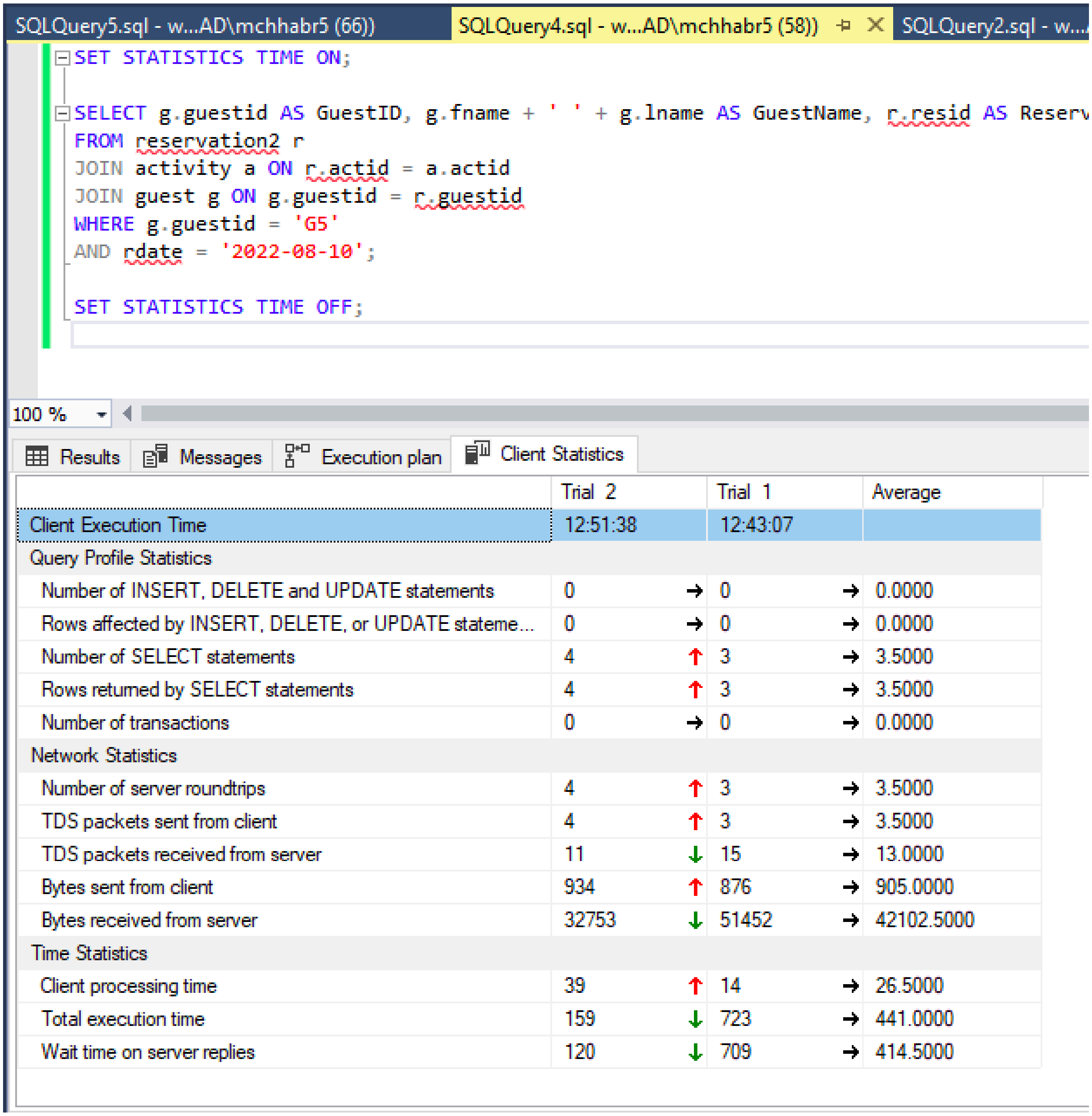


SQLQuery5.sql - w...AD\mchhabr5 (66))
SQLQuery4.sql - w...AD\mchhabr5 (58))
SQLQuery2.sql - w...AD
100 %
Results
Messages
Execution plan
Client Statistics

```sql
SET STATISTICS TIME ON;

SELECT g.guestid AS GuestID, g.fname + ' ' + g.lname AS GuestName, r.resid AS Reservat
FROM reservation2 r
JOIN activity a ON r.actid = a.actid
JOIN guest g ON g.guestid = r.guestid
WHERE g.guestid = 'G5'
AND rdate = '2022-08-10';

SET STATISTICS TIME OFF;
```

```
SQL Server parse and compile time:
   CPU time = 31 ms, elapsed time = 91 ms.

(2 rows affected)

(1 row affected)

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 6 ms.

Completion time: 2024-12-05T12:51:38.3272776-07:00
```

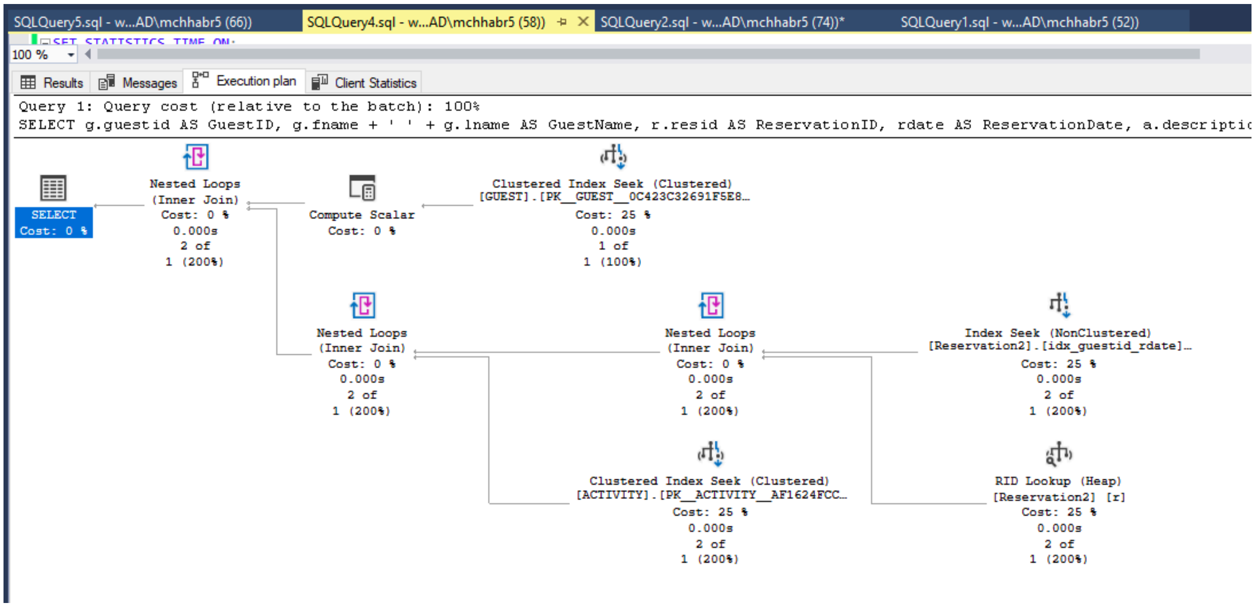
Question 6:


SQLQuery5.sql - w...AD\mchhabr5 (66))
SQLQuery4.sql - w...AD\mchhabr5 (58))
SQLQuery2.sql - w...
100 %
Results
Messages
Execution plan
Client Statistics

```sql
SET STATISTICS TIME ON;

SELECT g.guestid AS GuestID, g.fname + ' ' + g.lname AS GuestName, r.resid AS Reserv
FROM reservation2 r
JOIN activity a ON r.actid = a.actid
JOIN guest g ON g.guestid = r.guestid
WHERE g.guestid = 'G5'
AND rdate = '2022-08-10';

SET STATISTICS TIME OFF;
```

	Trial 2	Trial 1	Average
Client Execution Time	12:51:38	12:43:07	
Query Profile Statistics			
Number of INSERT, DELETE and UPDATE statements	0 →	0 →	0.0000
Rows affected by INSERT, DELETE, or UPDATE stateme...	0 →	0 →	0.0000
Number of SELECT statements	4 ↑	3 →	3.5000
Rows returned by SELECT statements	4 ↑	3 →	3.5000
Number of transactions	0 →	0 →	0.0000
Network Statistics			
Number of server roundtrips	4 ↑	3 →	3.5000
TDS packets sent from client	4 ↑	3 →	3.5000
TDS packets received from server	11 ↓	15 →	13.0000
Bytes sent from client	934 ↑	876 →	905.0000
Bytes received from server	32753 ↓	51452 →	42102.5000
Time Statistics			
Client processing time	39 ↑	14 →	26.5000
Total execution time	159 ↓	723 →	441.0000
Wait time on server replies	120 ↓	709 →	414.5000

Question 7:



```
SET STATISTICS TIME ON;
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT g.guestid AS GuestID, g.fname + ' ' + g.lname AS GuestName, r.resid AS ReservationID, rdate AS ReservationDate, a.descriptic
```

Question 8:

1. Execution Time Reduction:

Observed Change: The cpu time was reduced from 253 ms to 0 ms and elapsed time from 38 ms to 6 ms.

Explanation: Finally, the introduction of the secondary index greatly decreased the time taken to execute queries. Indexes assist the database engine to search for data within the table without looking for the data from all the table. Here, using the index on guestid and rdate, SQL Server was able to retrieve only those rows which conform to the query conditions and thus the CPU and elapsed times were drastically cut down.

2. Improvement in Client Statistics:

Observed Change: These are the kinds of improvements that are easier to understand because anyone can readily see that the total execution time decreased from 723 ms to 159 ms and the bytes received from the server from 51452 to 32753 bytes.

Explanation: With the help of the second index, the database engine trimmed the data acquisition, which in turn means fewer bytes to transfer and less time for client-side computations. It also served to minimize the data fetch scope index which in turn inevitably reduced the number of data packets returned to the client thus improving the efficiency of the query.

3. Query Plan Optimization:

Observed Change: There was a change in the query execution plan which changed from a Clustered Index Scan to an Index Seek on the Reservation2 table.

Explanation: Firstly, before the index existed, the database engine required searching for the particular rows in a particular table, which was time-consuming as it had to daily scan through the whole table in order to retrieve the necessary data. After the addition of an index, row seeking changed into Index Seek, which directly seeks the tree for rows without scanning unneeded rows. This method is significantly faster and less expensive in terms of consumable resources, which is also evident in terms of, for example, cost optimization and improved data access patterns in the execution plan.

Altogether, each of these changes proves that indexing is an essential element in the enhancement of database search operations. Specifically, indexes facilitate direct data access that lessens the burden on the native database engine thus enhancing the application's efficiency and extensibility. This case brings out the need to always employ proper indexing techniques that would be useful in the day to day retrieval of data.