

Database implementation

Requirements check



12 December 2017

Table of contents

[1 Booking a ticket 1](#_Toc9968696)

[2 Cancelling a ticket booking 2](#_Toc9968697)

[3 Changing number of tickets in booking 2](#_Toc9968698)

[4 Changing status of booking 4](#_Toc9968699)

[5 Removing unpurchased bookings from the database after 3 days from booking 4](#_Toc9968700)

[6 Removing the unpurchased bookings of the past events from the database 5](#_Toc9968701)

[7 Cancelling an event 7](#_Toc9968702)

[8 Refunding a client in case of a cancelled event 8](#_Toc9968703)

[9 Performances coming up next month 8](#_Toc9968704)

[10 Performance coming up this month 8](#_Toc9968705)

[11 What is Saara Aalto's phone number? 9](#_Toc9968706)

[12 When will ZZ TOP perform in Kuru and what their special requests for catering 9](#_Toc9968707)

[13 How many tickets have been sold to Michael Flatley's dance perfomance "River dance" that takes place on 20.7.2018 9](#_Toc9968708)

[14 How many tickets are there left for Adele's concert on 1.3.2018 10](#_Toc9968709)

[15 How much money has the Kuru Culture got from sold tickets this year? 11](#_Toc9968710)

[16 Physical design of diagram 11](#_Toc9968711)

# Booking a ticket

First, the amount of tickets booked so far for different events and their max capacity is observed.

*SELECT Event.eventId, eventName, Venue.capacity,*

*SUM(Booking.ticketQuantity) AS 'Tickets booked so far',*

*(Venue.capacity-SUM(Booking.ticketQuantity)) AS 'Number of free places'*

*FROM Event*

*JOIN Venue ON (Venue.venueId = Event.venueId)*

*JOIN Booking ON (Booking.eventId = Event.eventId)*

*GROUP BY Event.eventId, eventName, Venue.capacity*

Above query produced the following result:

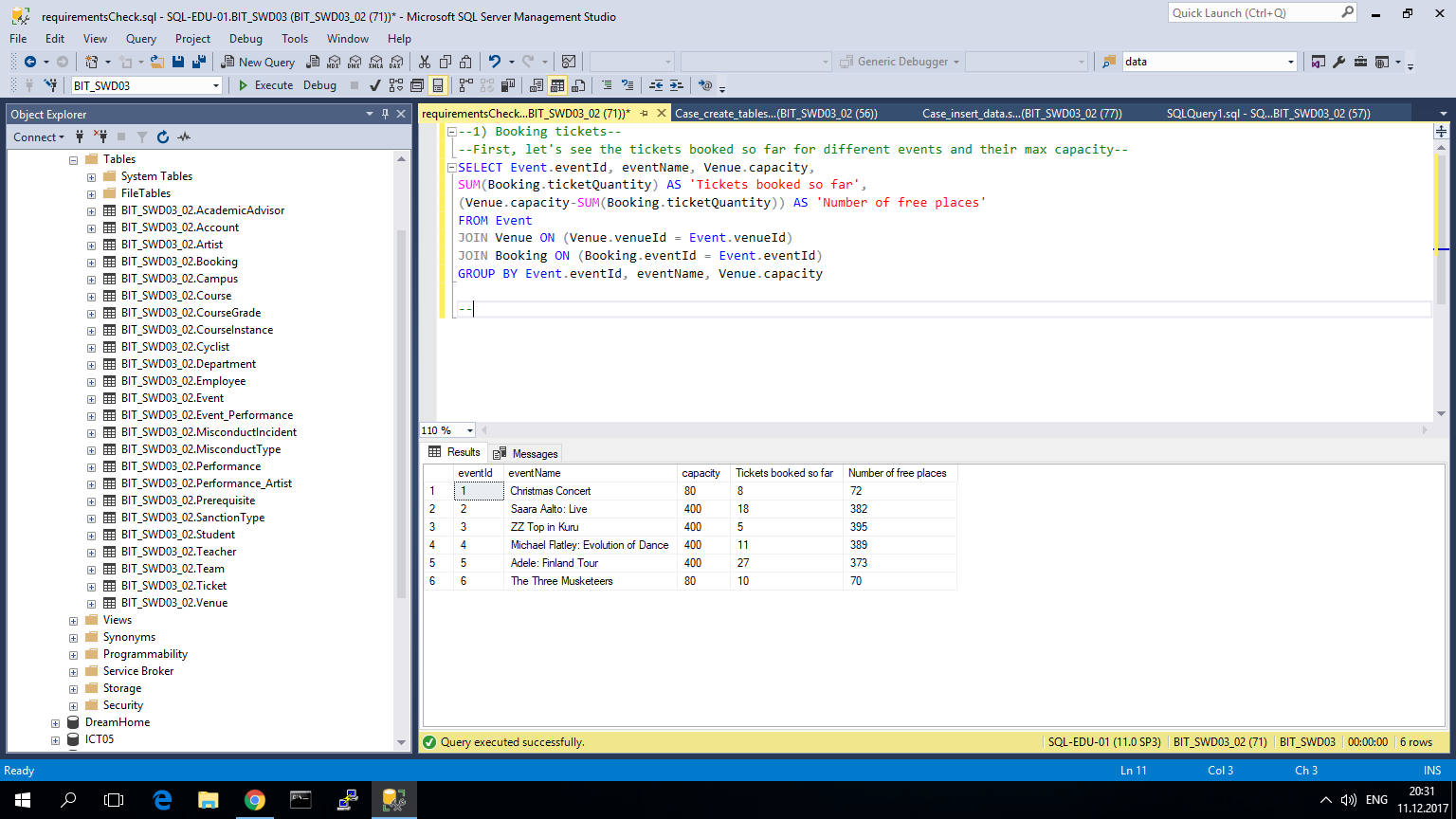


Figure 1 Event ticket capacity, number of tickets booked so far, and number of free space

Second, a new booking query with 70 tickets for event 1 – Christmas Concert, was written; the result was checked in table.

*INSERT INTO Booking (eventId, bookingDate, ticketQuantity, bookingStatus, memo) VALUES*

*(1, '2017-12-11', 70, 'Booked', '0413682916')*

Result:

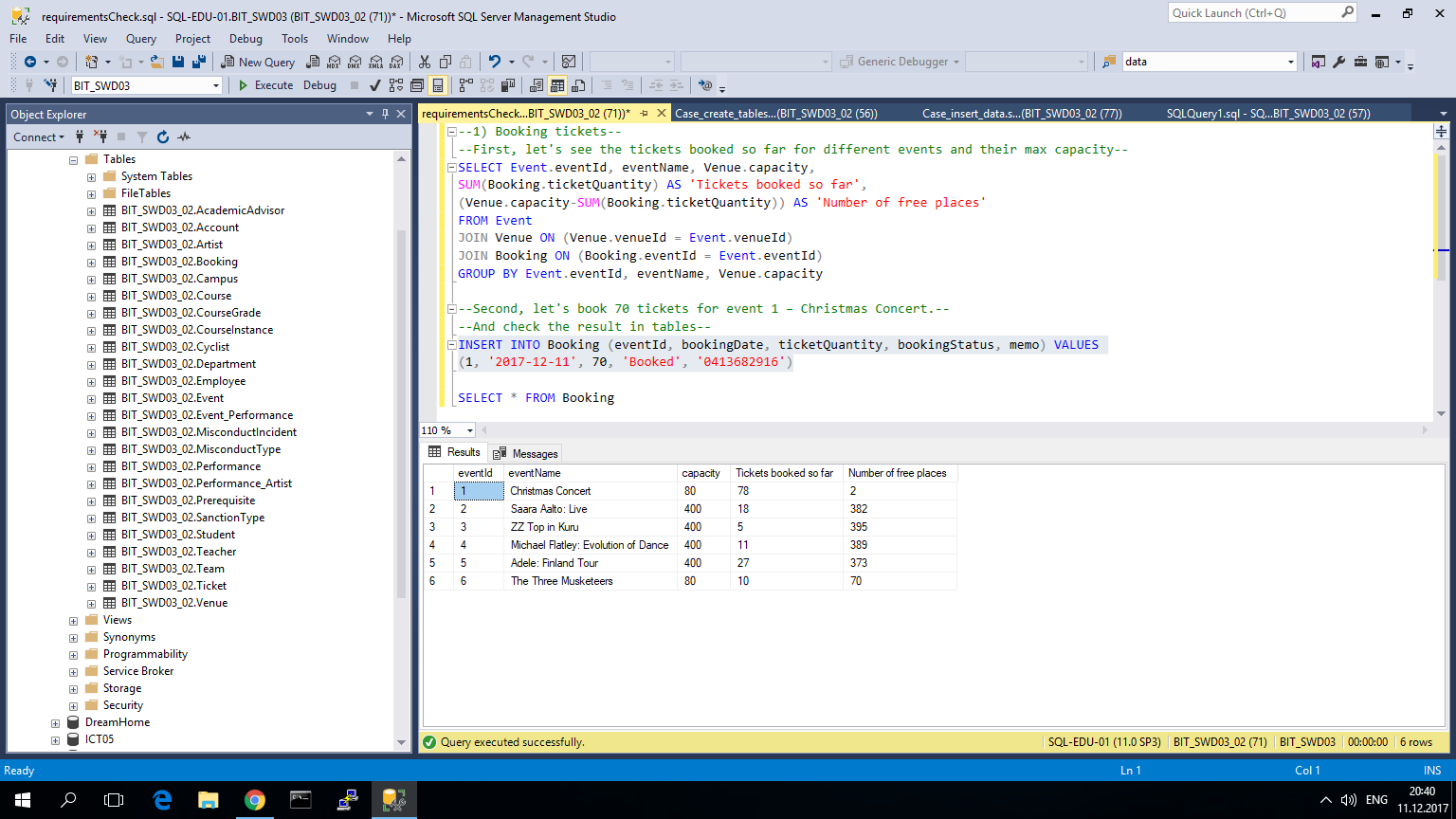


Figure 2 Changes in the table after querying a new booking

It is clear, that query worked properly, and only 2 tickets were left for the Christmas Concert.

**Booking process works properly!**

# Cancelling a ticket booking

The booking with tickets of 70 has bookingId = '10'. To cancel the booking a DELETE query must be performed and the result must be checked.

*DELETE FROM Booking WHERE bookingId = '10'*

Result:

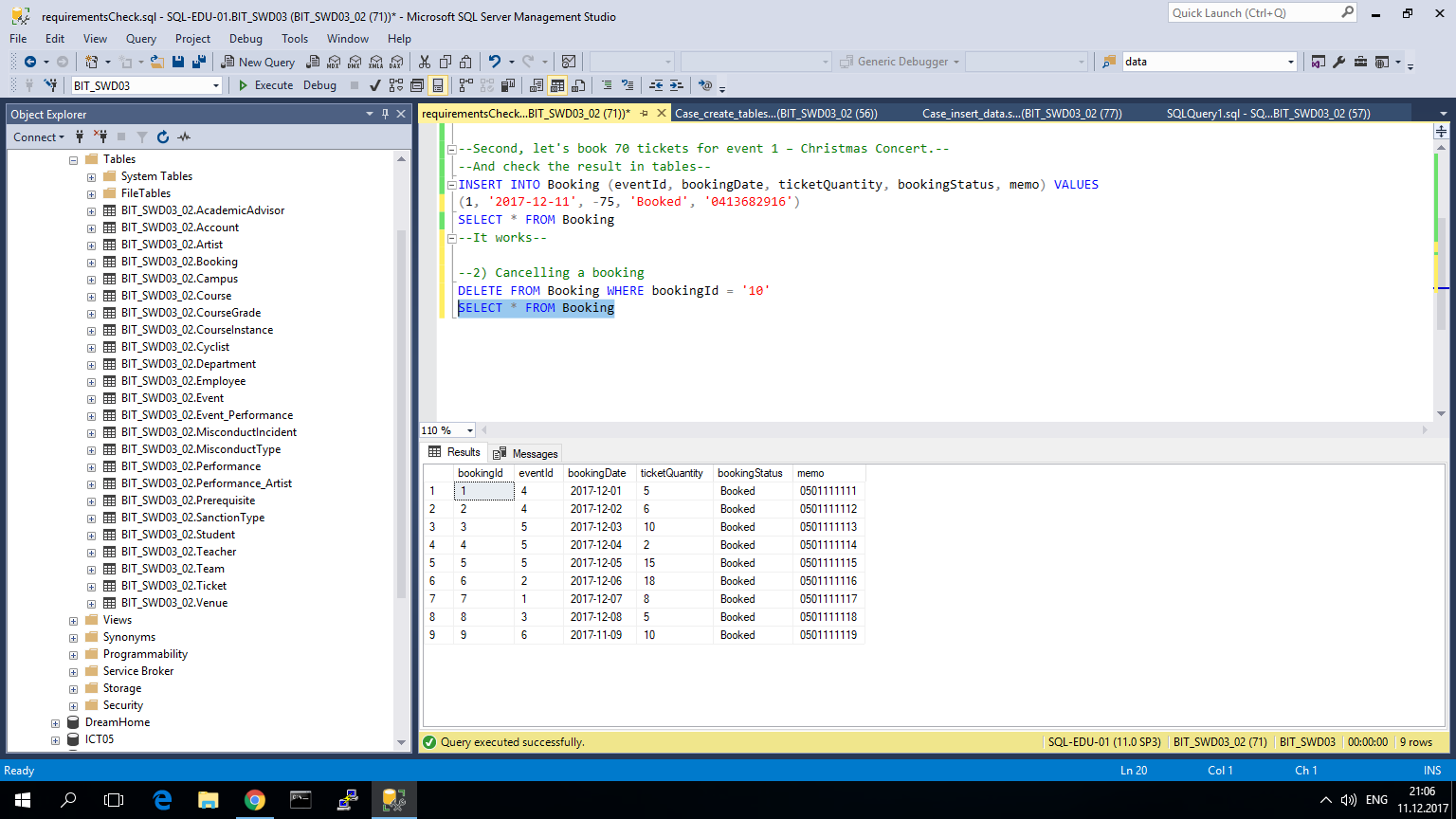


Figure 3 Result in Booking table after DELETE query of a booking

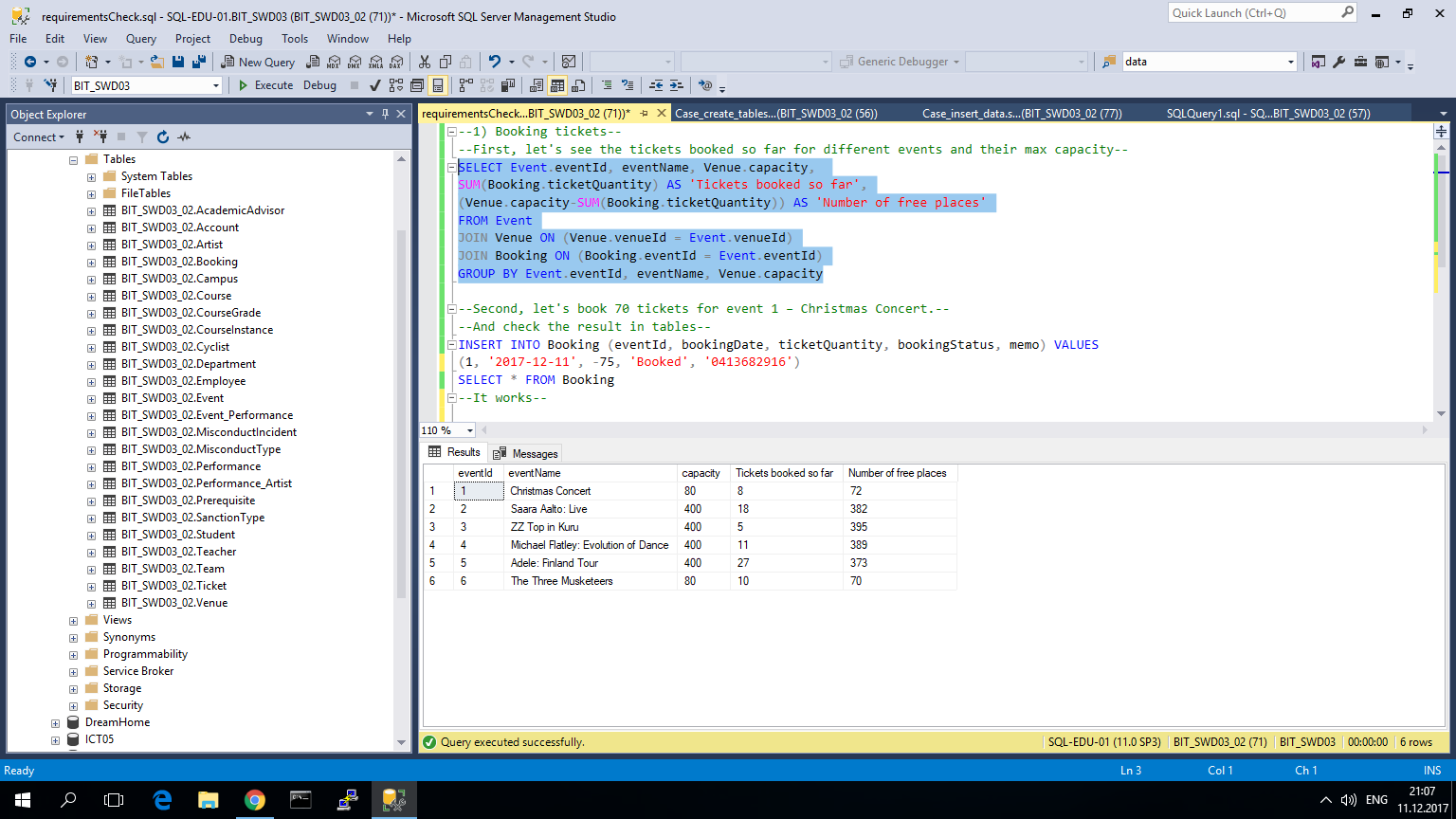


Figure 4 DELETE of booking also affected booked tickets for the Christmas Concert

As it is seen, the booking entry was removed from Booking table, and Event table has again only 8 tickets booked for Christmas Concert.

**Booking cancelling works just right!**

# Changing number of tickets in booking

First, a booking with 10 tickets for Christmas Concert is created.

*INSERT INTO Booking (eventId, bookingDate, ticketQuantity, bookingStatus, memo) VALUES*

*(1, '2017-12-11', 10, 'Booked', '0413682916')*

Result:

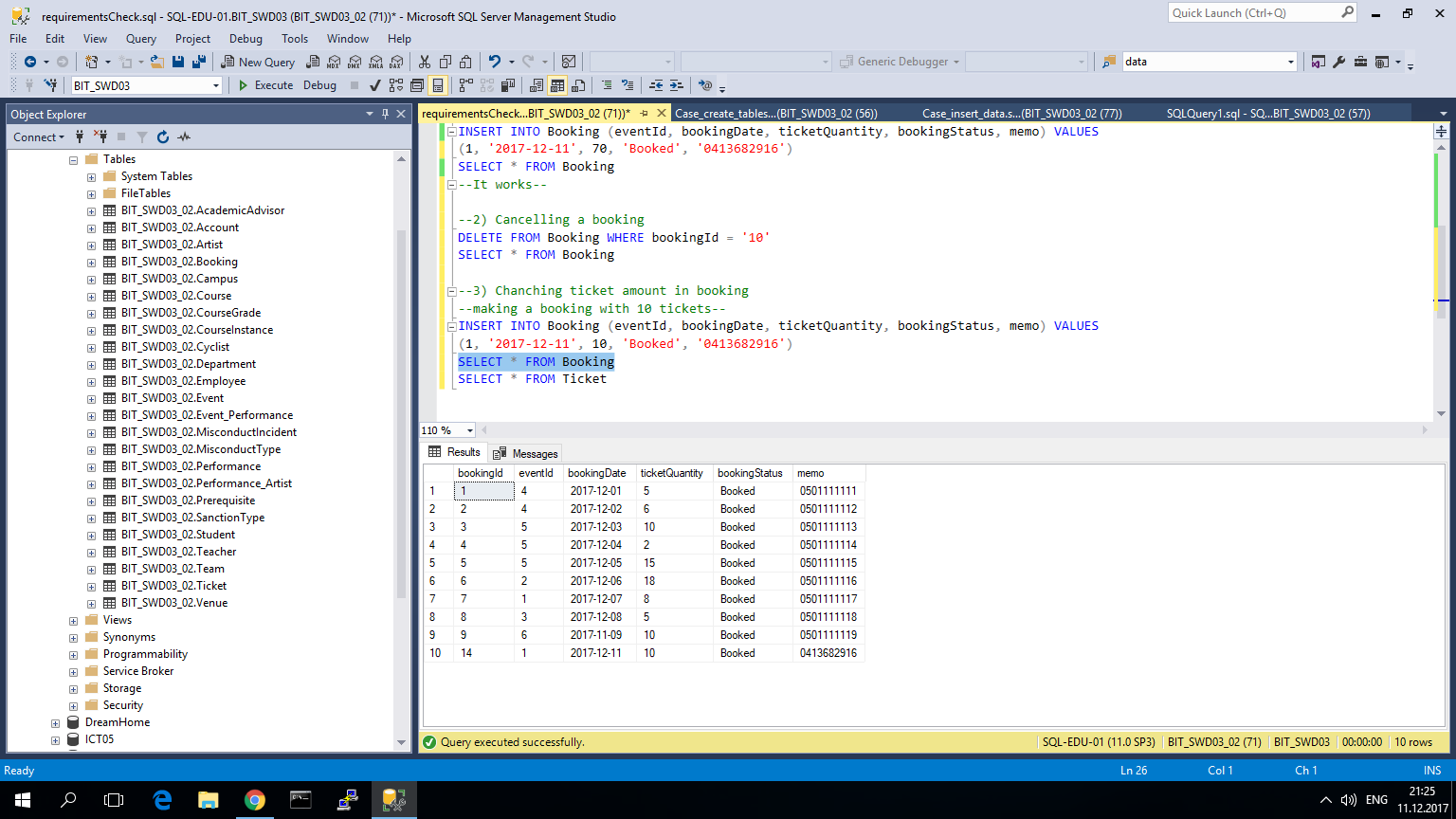


Figure 5 A new entry in Booking table with ID 14 s created

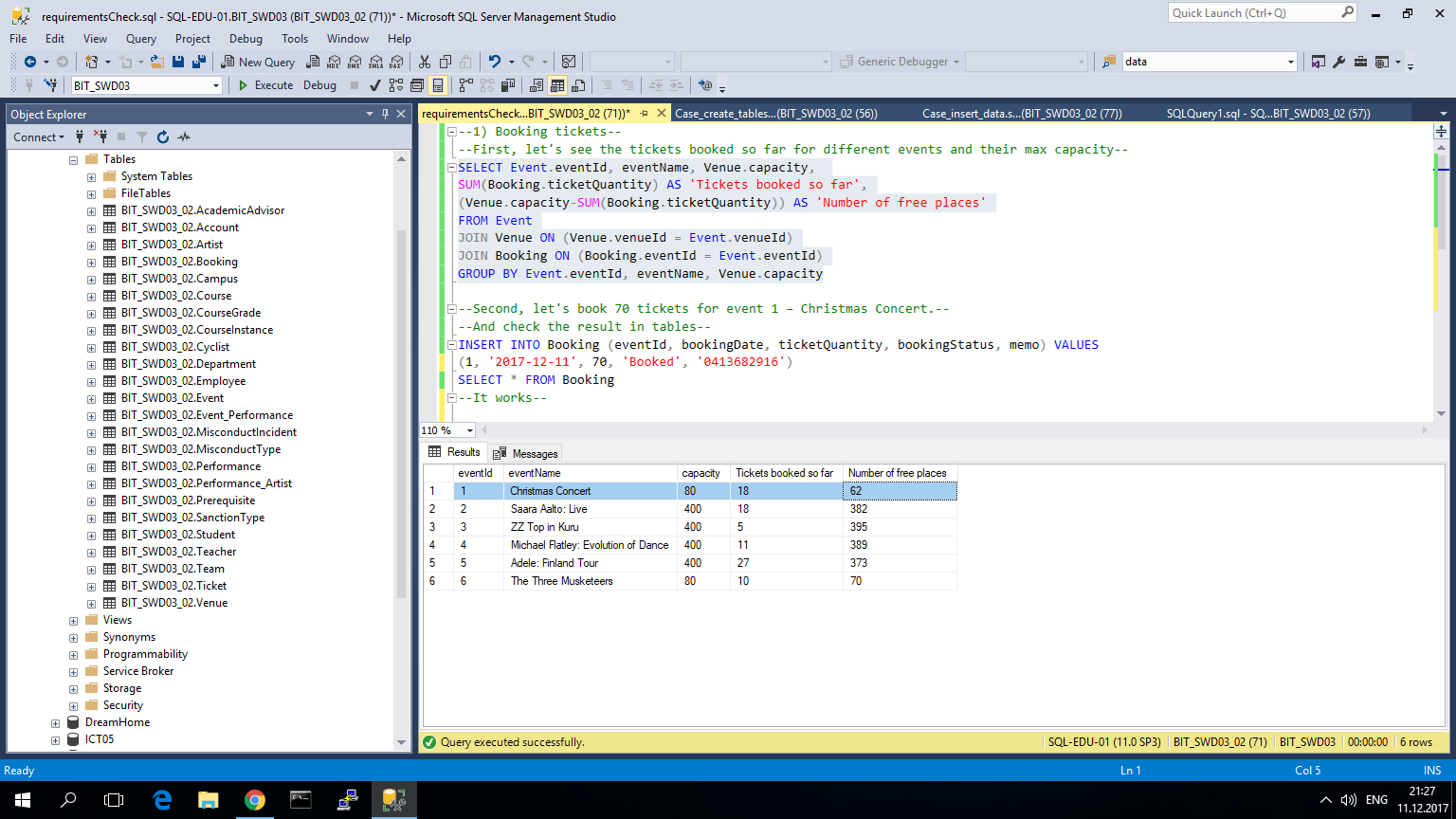


Figure 6 new booking affected the ticket amount in Christmas Concert event

Booking was successful, since the number of booked tickets for event with ID 1 increased from 8 to 18. Booking table got also a new row with ID 14 and ticket quantity of 10.

**Second**, the booking (ID 14) is modified and instead of 10 tickets, booking is set to 5 tickets.

*UPDATE Booking SET ticketQuantity = '5' WHERE bookingId = '14'*

Result:

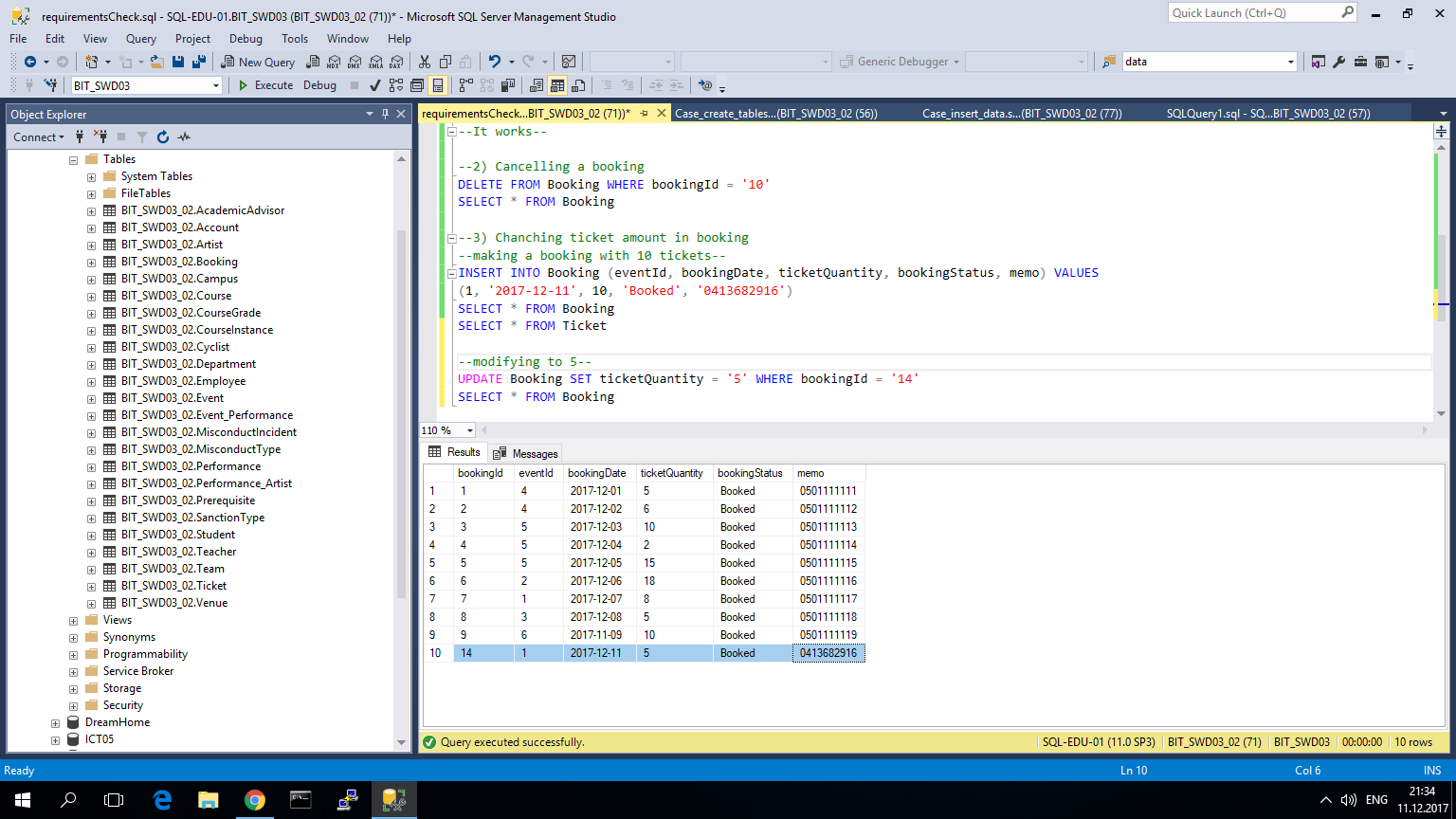


Figure 7 Changes in the booking entry after modification

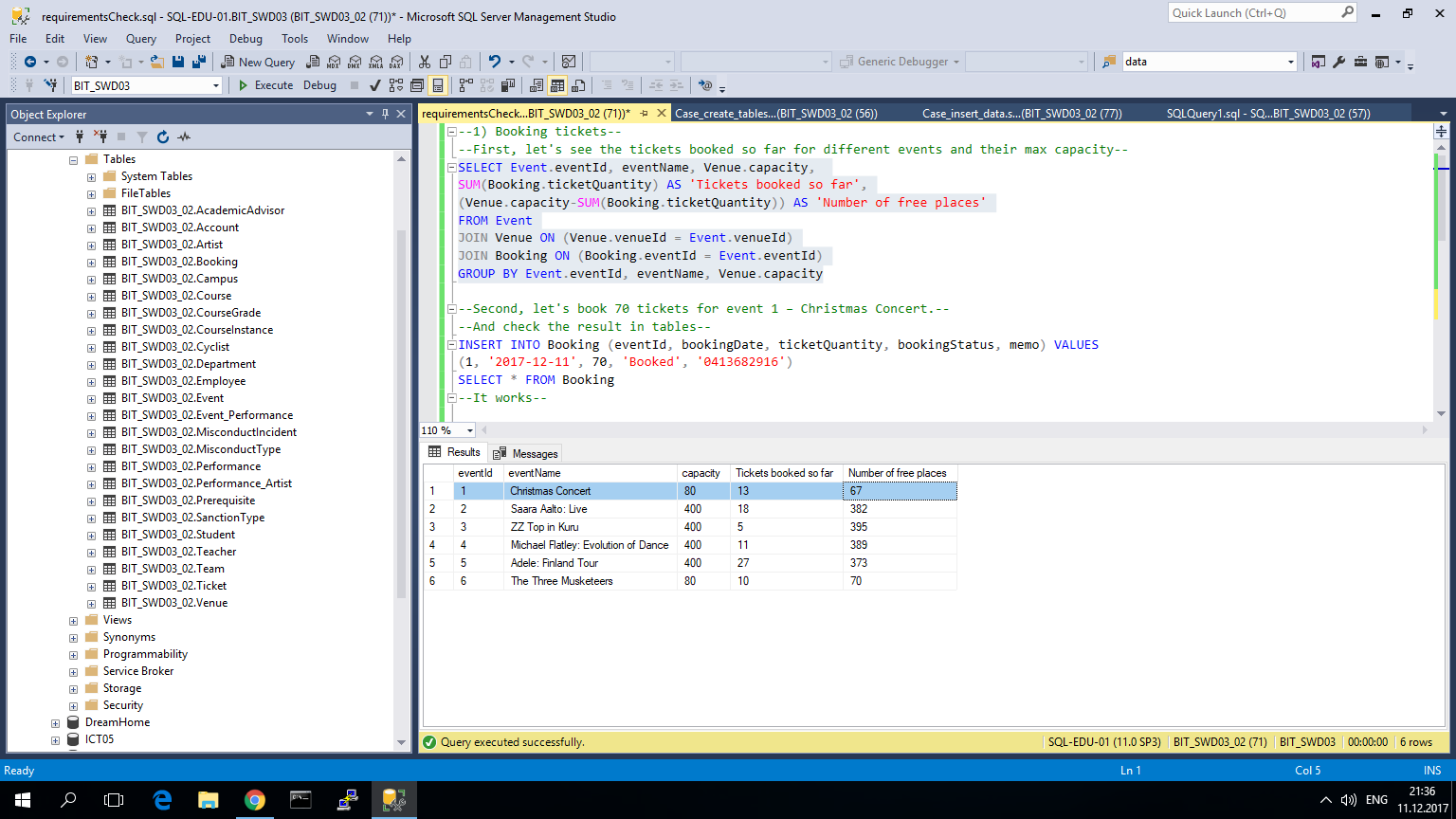


Figure 8 Modification of the booking entry also affected the quantity of tickets for the according event

The query works properly. Ticket quantity has changed and number of tickets for Christmas Concert was also affected.

**It works!**

# Changing status of booking

*UPDATE Booking SET bookingStatus = 'Paid' WHERE bookingId = '14'*

Result:

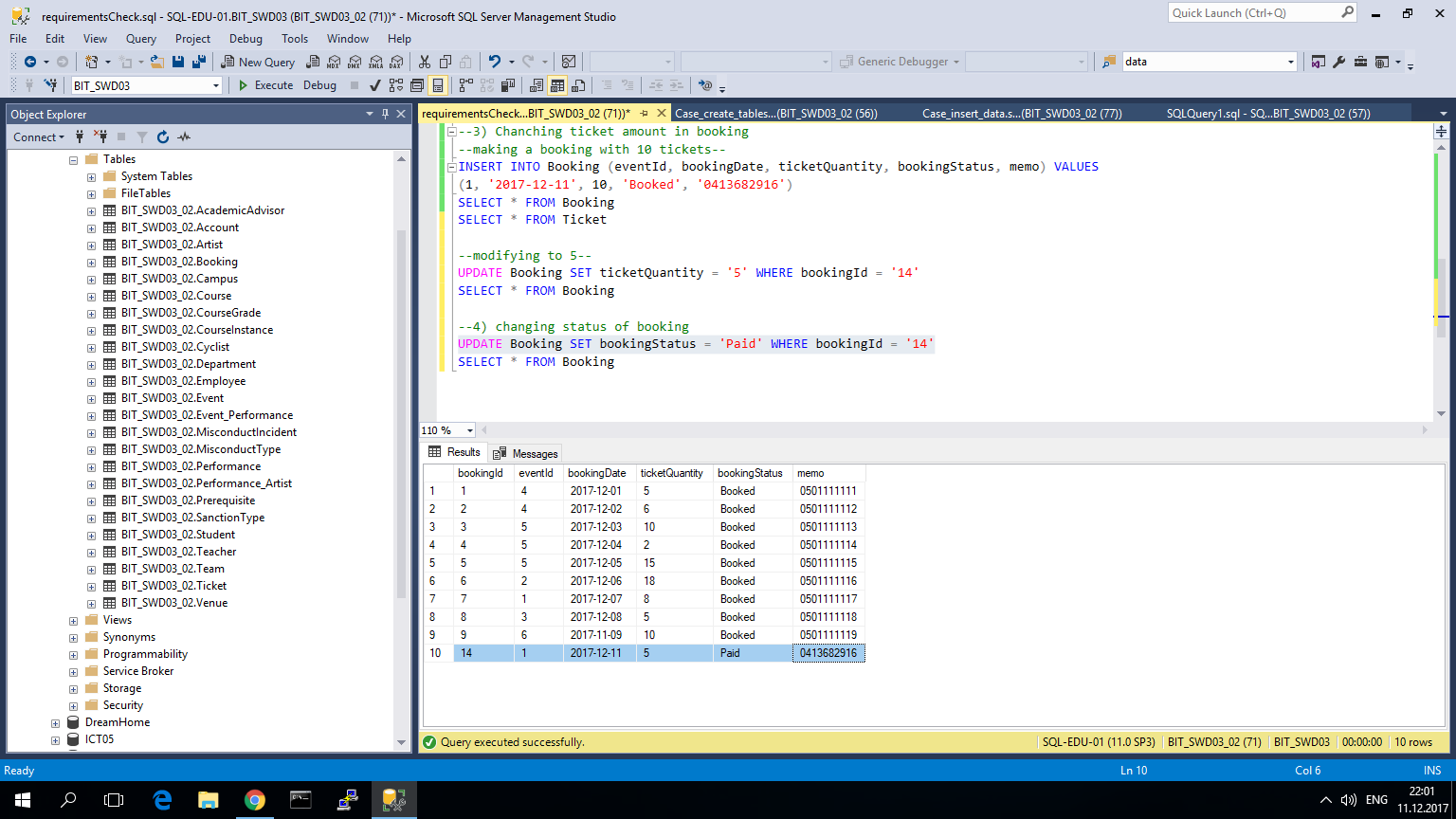


Figure 9 Booking status change

**It works!**

# Removing unpurchased bookings from the database after 3 days from booking

First, the situation is observed.

*SELECT \*, (DAY(SYSDATETIME())-DAY(bookingDate)) AS 'days past since booking' FROM Booking*

Result (the observation date was Dec 11, 2017):

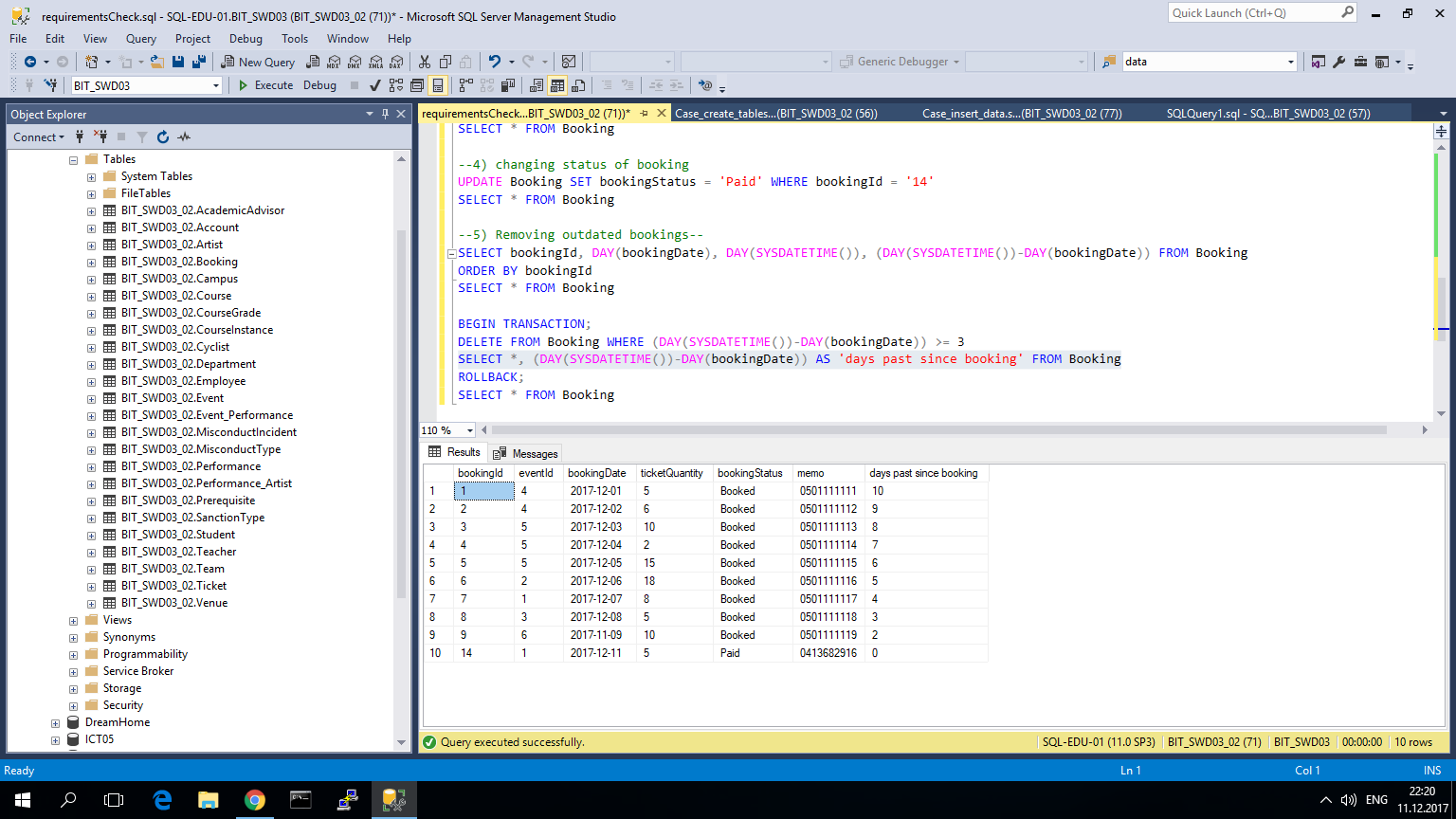


Figure 10 observation of bookings' freshness

The next action is aimed at removing bookings that were not paid after 3 days since the time of their record.

*DELETE FROM Booking WHERE (DAY(SYSDATETIME())-DAY(bookingDate)) > 3*

Result:

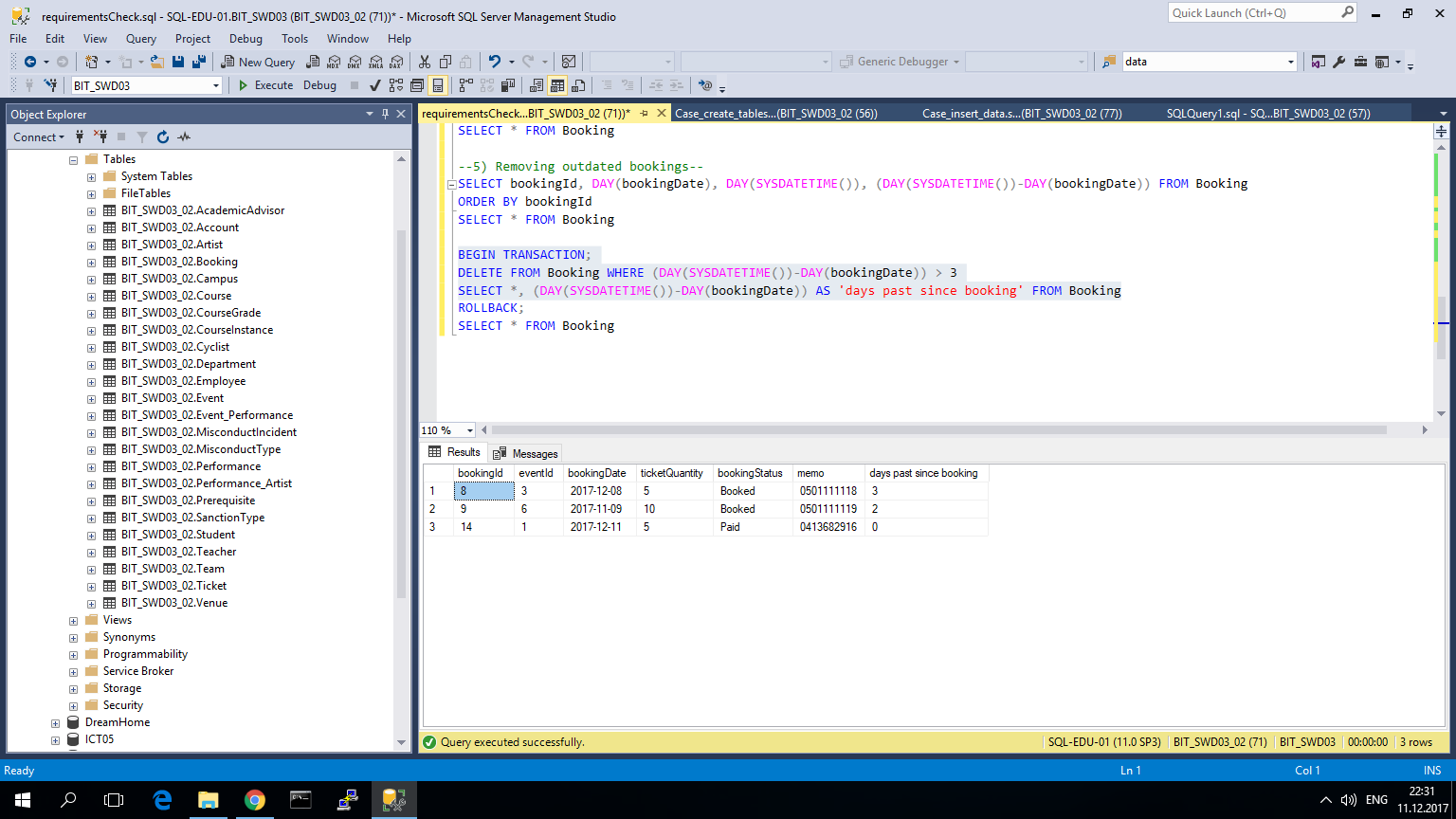


Figure 11 Only paid and fresh bookings are left

The query left only those bookings, that were made no more than 3 days ago.

**The requirement is satisfied!**

# Removing the unpurchased bookings of the past events from the database

First, 2 new events, that have happened in the past, were added.

*INSERT INTO Event (eventName, eventDate, price, venueId) VALUES*

*('HH Graduation', '2017-05-30', 5, 1),*

*('Rolling Stones COncert', '2016-06-24', 115, 2)*

*SELECT \*, CONVERT(date,SYSDATETIME()) AS 'current date' FROM Event*

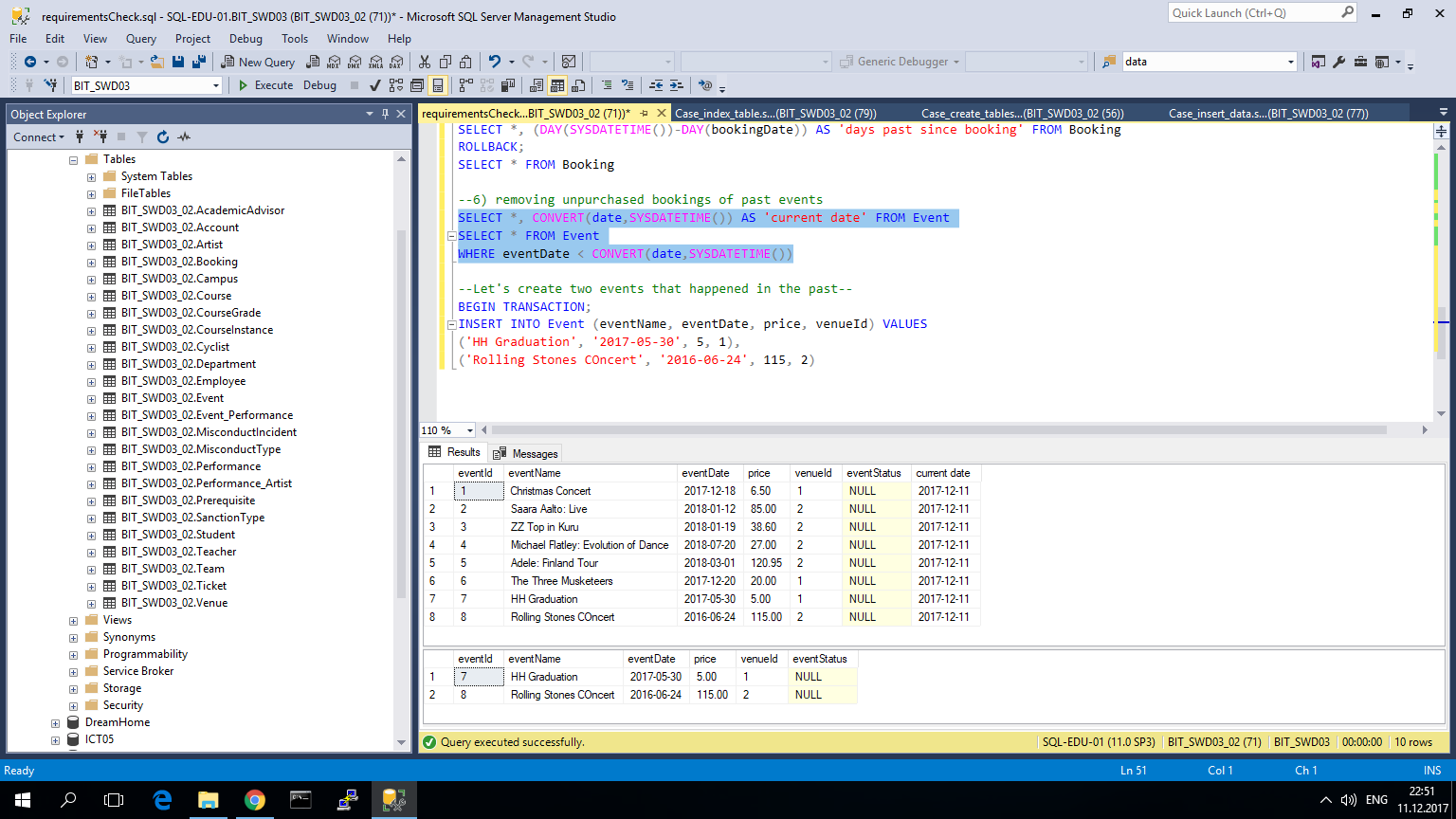


Figure 12 New events appeared in the table

Second, bookings to those 2 events were added.

*INSERT INTO Booking (eventId, bookingDate, ticketQuantity, bookingStatus, memo) VALUES*

*(7, '2017-05-15', 5, 'Booked', '0413682917'),*

*(7, '2017-05-12', 6, 'Paid', '0413682918'),*

*(8, '2016-06-03', 10, 'Booked', '0413682919'),*

*(8, '2016-06-04', 2, 'Paid', '0413682915')*

Result:

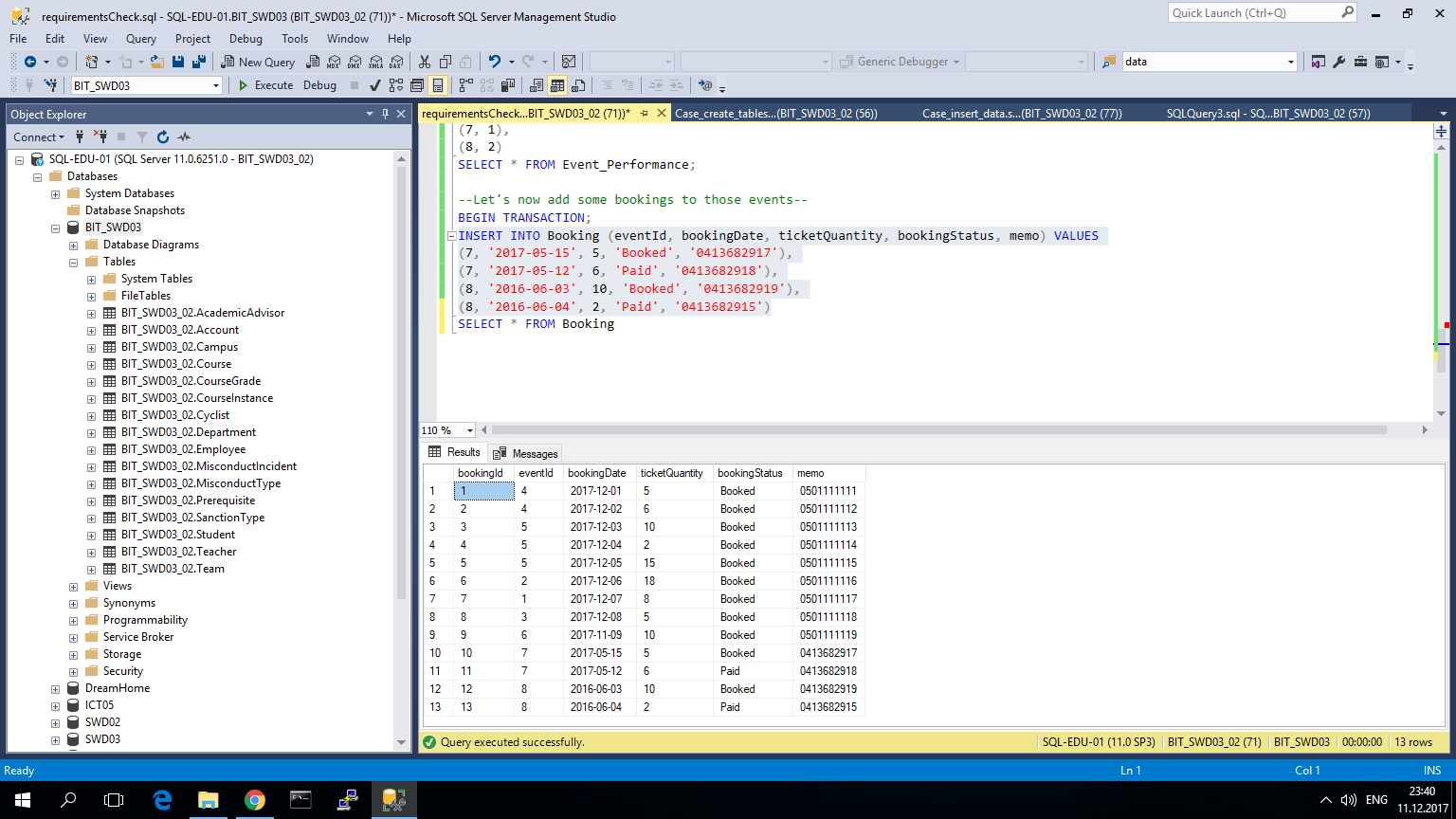


Figure 13 New bookings are recorded

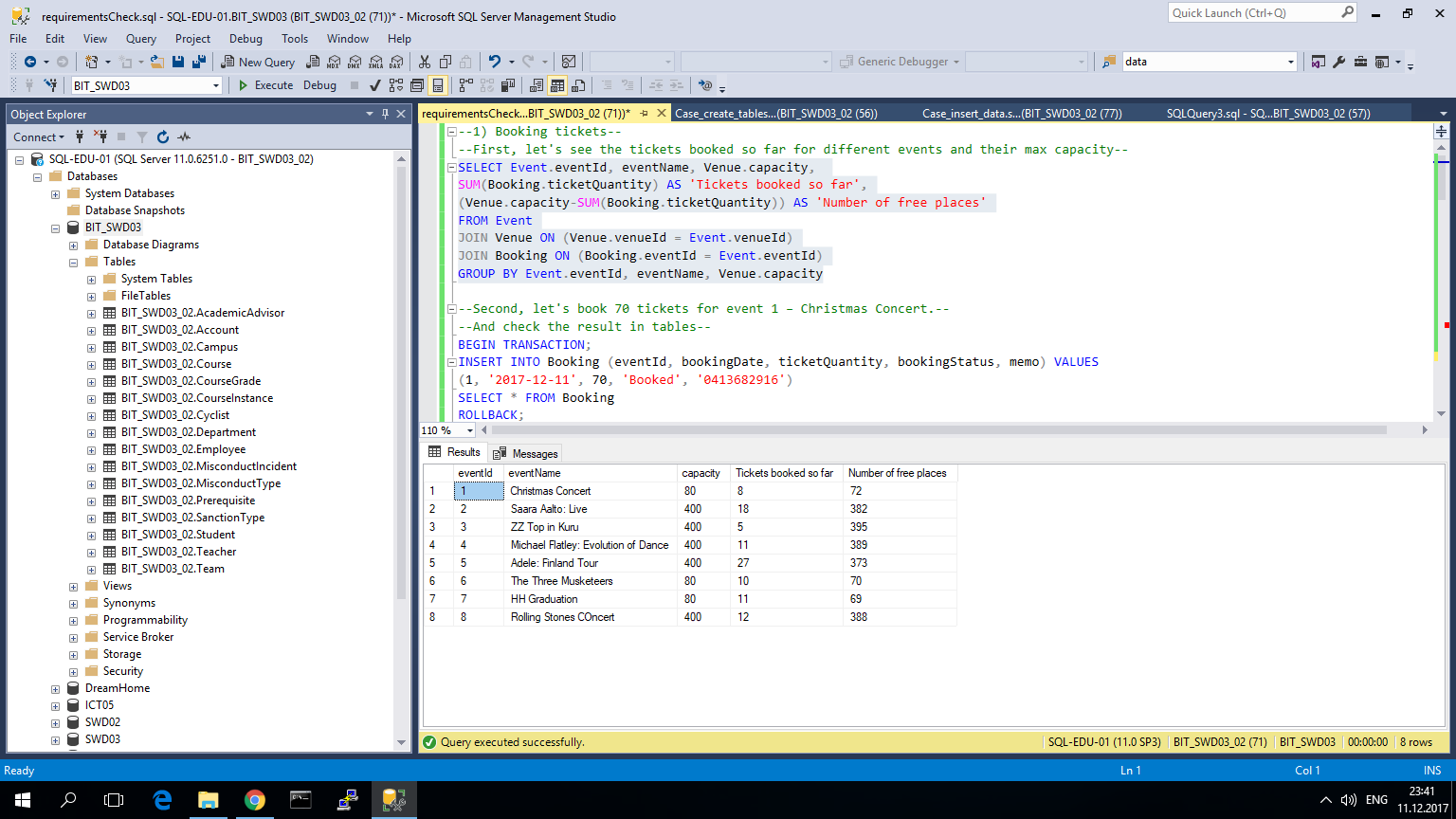


Figure 14 New bookings affected the according events

So, new events and bookings for them are inserted in the database. Events have pasted, but for now both booked and paid bookings are visible. Finally, only paid number of tickets (unpaid bookings of past events must be deleted) must be shown.

*DELETE FROM Booking*

*WHERE bookingStatus = 'Booked' AND eventId IN (SELECT eventId FROM Event*

*WHERE eventDate < CONVERT(date,SYSDATETIME()))*

Result:

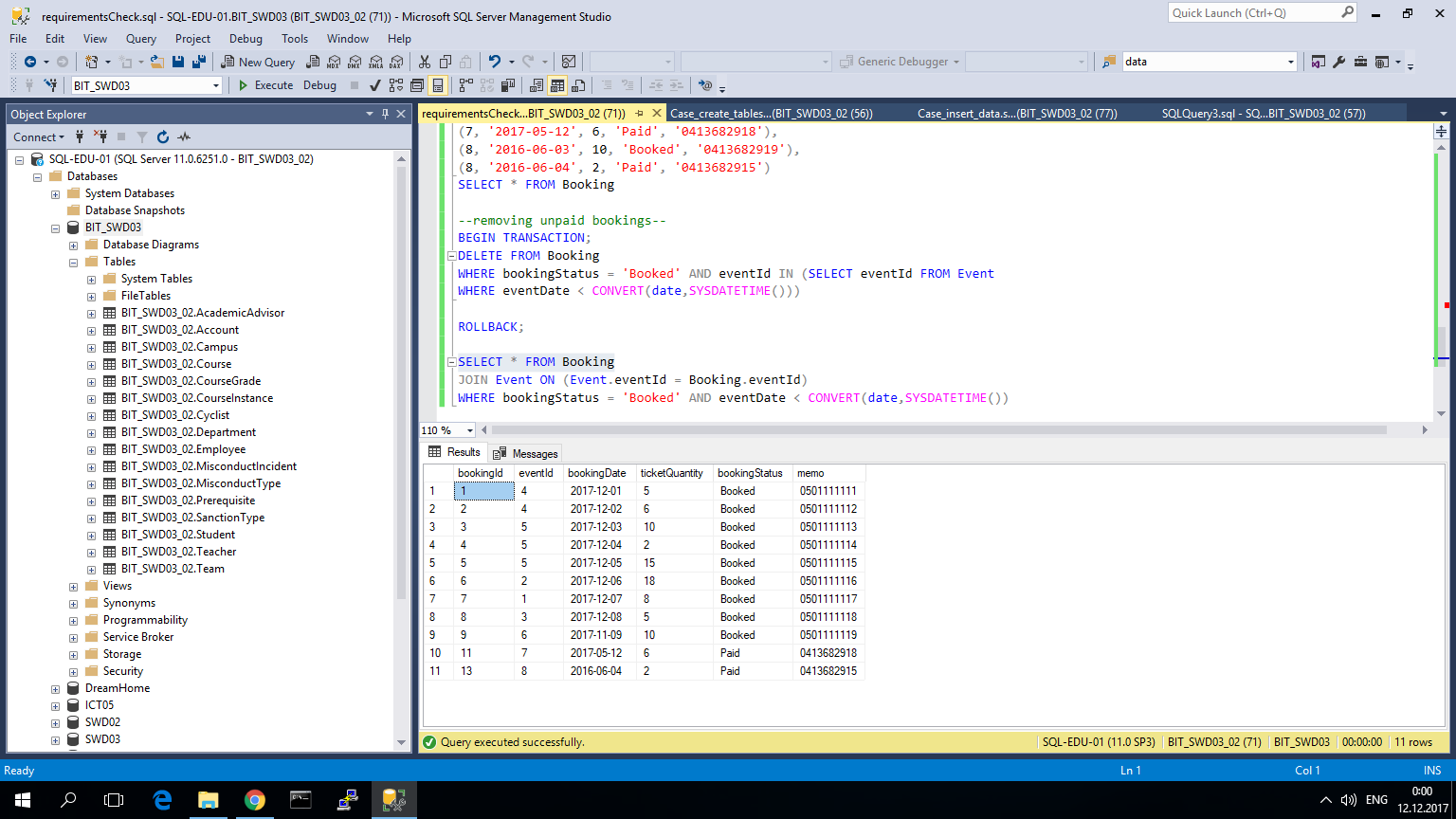


Figure 15 Query result: only paid bookings of past events are saved

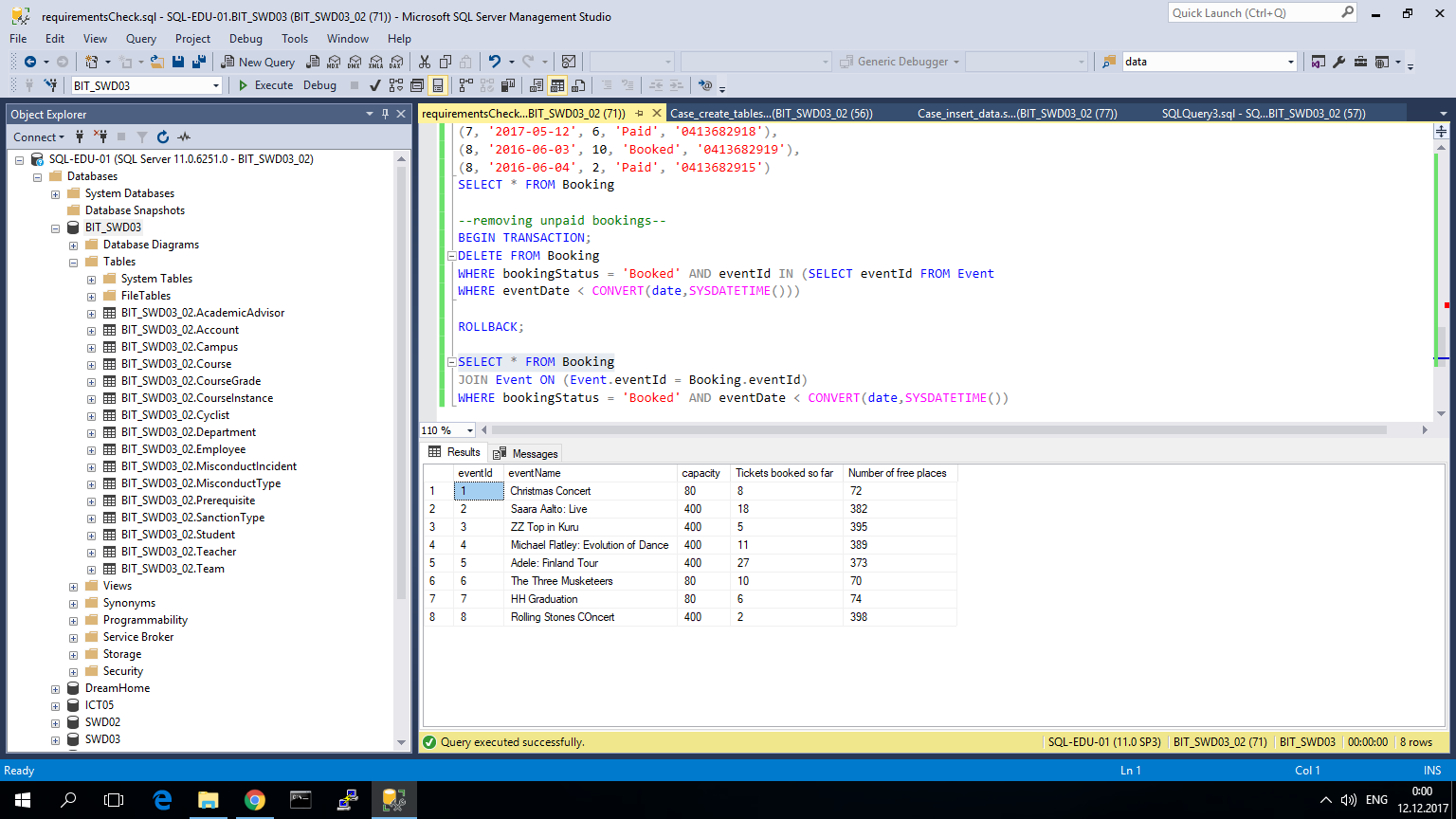


Figure 16 Logic is applied to events also

Now unpaid bookings of events of the past are removed, also in the ticket quantity for the events of the past only number of tickets that were actually paid are saved.

**Requirement is satisfied!**

# Cancelling an event

Observation of the current situation.

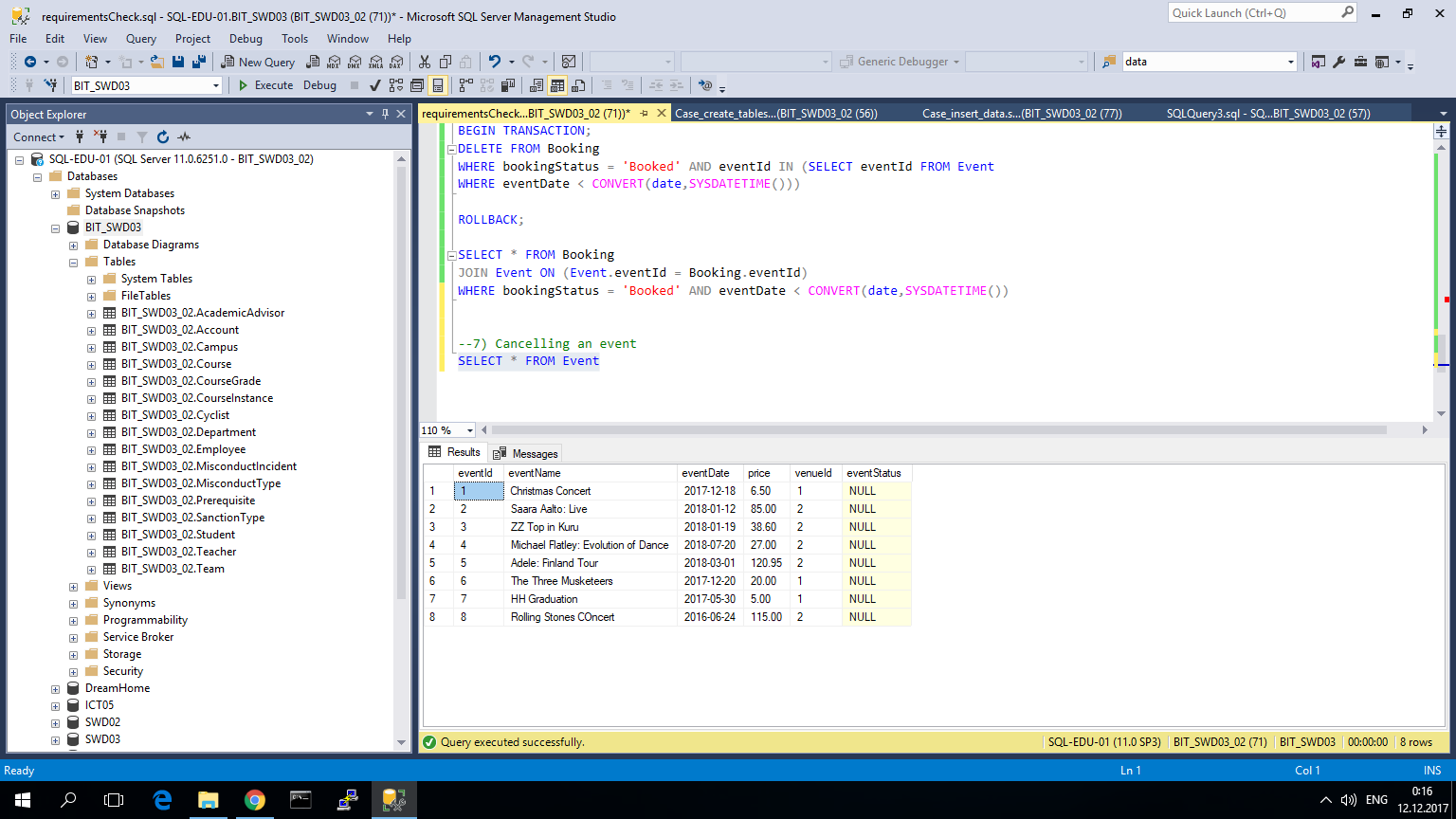


Figure 17 Events in the Event table

According to the next step in the scenario Adele's tour in Finland got cancelled, because she fell ill.

*UPDATE Event SET eventStatus = 'Cancelled' WHERE eventId = '5'*

Result:

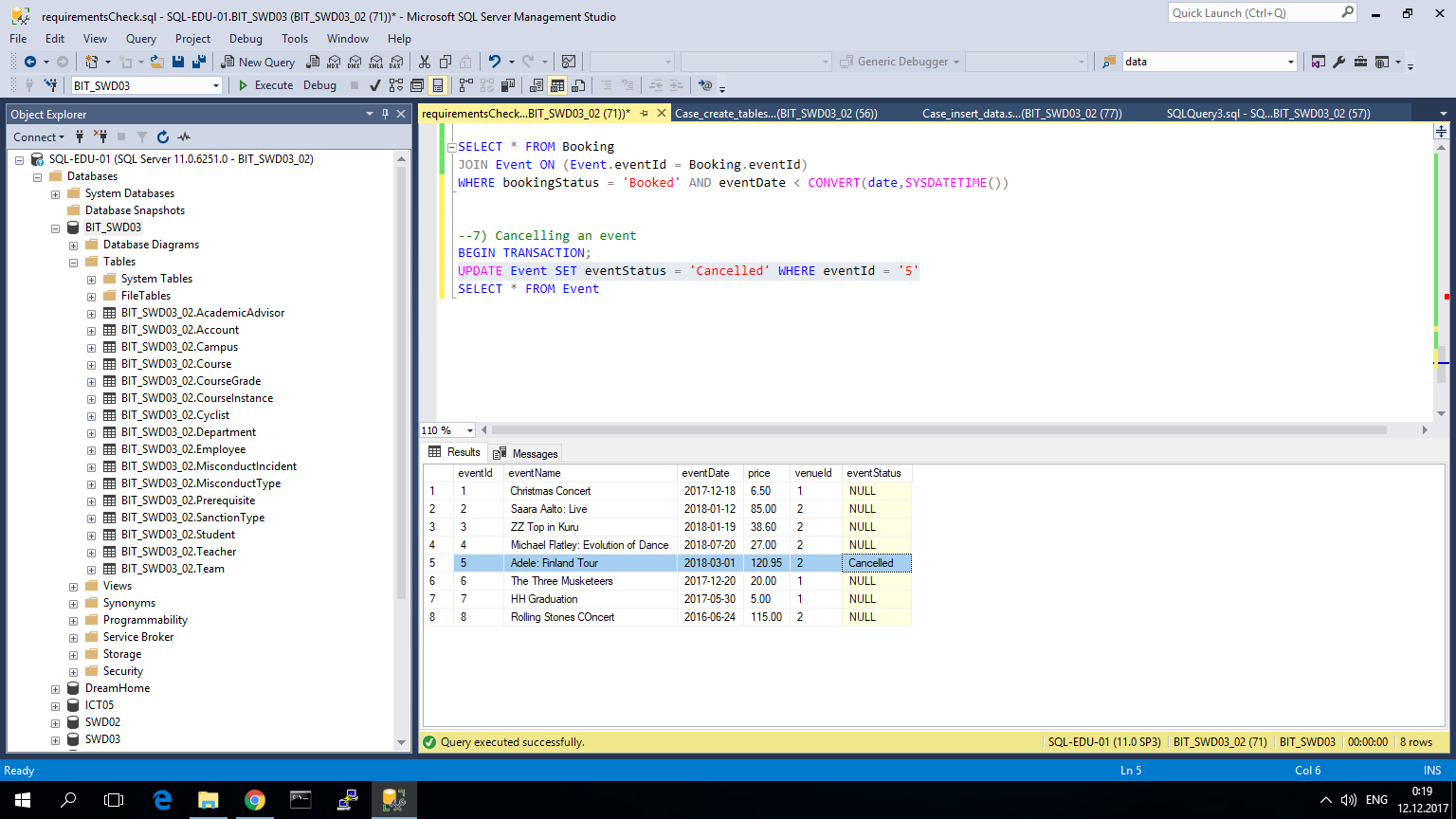


Figure 18 Result of the query execution

**Event has been successfully cancelled!**

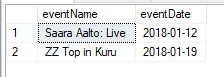
# Refunding a client in case of a cancelled event

Because Ticket is linked to Booking that in turn is linked to Event, then in case of event cancellation and refunding for ticket, ticket's status (isRefunded) will be changed to 'Yes'.

# Performances coming up next month

*Select eventName,eventDate from Event*

*where eventDate between '2018-01-11' and '2018-02-11'*



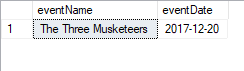
# Performance coming up this month

*select eventName,eventDate from Event*

*Join Event\_Performance on (Event\_Performance.eventId=Event.eventId)*

*join Performance on (Performance.performanceId=Event\_Performance.performanceId)*

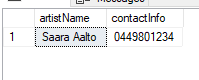
*Where MONTH(eventDate) = MONTH(GETDATE()) and type ='Theater';*



# What is Saara Aalto's phone number?

*Select artistName,contactInfo from Artist*

*where artistName = 'Saara Aalto';*



# When will ZZ TOP perform in Kuru and what their special requests for catering

*Select eventName,eventDate,specialRequest,artistName*

*from Event*

*Join Event\_Performance on (Event\_Performance.eventId=Event.eventId)*

*Join Performance\_Artist on (Performance\_Artist.performanceId=Event\_Performance.performanceId)*

*Join Artist on (Artist.artistId = Performance\_Artist.artistId)*

*where Artist.artistName = 'ZZ TOP';*



# How many tickets have been sold to Michael Flatley's dance perfomance "River dance" that takes place on 20.7.2018

*Select count(\*) as "Number of tickets sold",artistName,performanceName,eventDate*

*from Ticket join Booking on (Booking.bookingId=Ticket.bookingID)*

*join Event on (Event.eventId=Booking.eventId)*

*Join Event\_Performance on (Event\_Performance.eventId=Event.eventId)*

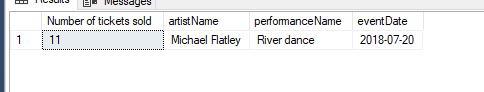
*join Performance on (Performance.performanceId =Event\_Performance.performanceId)*

*Join Performance\_Artist on (Performance\_Artist.performanceId=Performance.performanceId)*

*Join Artist on (Artist.artistId = Performance\_Artist.artistId)*

*Group by artistName,performanceName,eventDate*

*Having eventDate = '2018-07-20' and performanceName='River dance' and artistName='Michael Flatley'*



# How many tickets are there left for Adele's concert on 1.3.2018

*select(400-COUNT(ticketId)) as "Number of tickets left", Event.eventDate,Performance.type,Artist.artistName from Ticket*

*join Booking on (Booking.bookingId=Ticket.bookingID)*

*join Event on (Event.eventId=Booking.eventId)*

*Join Event\_Performance on (Event\_Performance.eventId=Event.eventId)*

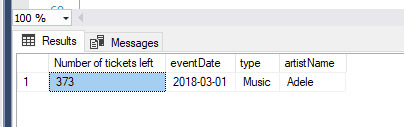
*join Performance on (Performance.performanceId =Event\_Performance.performanceId)*

*Join Performance\_Artist on (Performance\_Artist.performanceId=Performance.performanceId)*

*Join Artist on (Artist.artistId = Performance\_Artist.artistId)*

*Group by artistName,eventDate,type*

*having artistName='Adele' and eventDate ='2018-03-01' and type ='Music'*



# How much money has the Kuru Culture got from sold tickets this year?

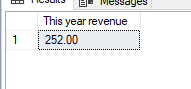
*select Sum(price\*Booking.ticketQuantity) as 'This year revenue'*

*from event*

*join Booking on(booking.eventId= event.eventId)*

*where YEAR(eventDate) = 2017 and bookingStatus='Paid'*

*group by YEAR(eventDate)*



# Physical design of diagram

Below we present our database diagram

****