```
/* Authors: Jessica McEwan C3393168, Lena Dahlin C3391146
   Task: Assignment 1 Make Reservation stored procedure
   Date Created: 18/03/2023 Last updated: 26/03/2023
*/

-- Drop procedures + types
DROP PROCEDURE IF EXISTS usp_makeReservation
GO
DROP TYPE IF EXISTS bookedPackages
DROP TYPE IF EXISTS guestList
GO
--creating a type bookedPackages
CREATE TYPE bookedPackages AS TABLE(
   packageID CHAR(10),
   qtyBooked INT,
   startDate DATETIME,
   endDate DATETIME
)
GO
--creating guest list type
CREATE TYPE guestList AS TABLE(
   name VARCHAR(30),
   phone VARCHAR(10),
   email VARCHAR(30),
   streetNo VARCHAR(10),
   streetName VARCHAR(30),
   city VARCHAR(30),
   postcode VARCHAR(10),
   country VARCHAR(30)
)
GO

CREATE PROCEDURE usp_makeReservation
@bookedPackages bookedPackages READONLY,
@guests guestList READONLY,
@custName VARCHAR(30),
@custPhone VARCHAR(10),
@custEmail VARCHAR(30),
--address
@streetNo VARCHAR(10),
@streetName VARCHAR(40),
@city VARCHAR(30),
@postcode VARCHAR(10),
@country VARCHAR(30),
@reservationID CHAR(10) OUTPUT
AS
BEGIN
```

```
    --check to see if the dates booked for the package fall within the packages available dates
    DECLARE AdvertisedDates CURSOR FOR
        SELECT bp.packageID, bp.startDate, bp.endDate
        FROM @bookedPackages bp
    DECLARE
        @packageID CHAR(10),
        @startDate DATETIME,
        @endDate DATETIME;
    OPEN AdvertisedDates
    FETCH NEXT FROM AdvertisedDates INTO @packageID, @startDate, @endDate
    BEGIN TRY
    WHILE @@FETCH_STATUS = 0
    BEGIN
        --check to see if the dates booked for the package fall within the packages available
dates
        BEGIN TRANSACTION
            IF NOT EXISTS (SELECT bp.packageID, bp.startDate, bp.endDate
            FROM @bookedPackages bp
            JOIN Package p ON bp.packageID = p.packageID
            WHERE bp.startDate >= p.startDate AND bp.endDate <= p.endDate)
            BEGIN
                DECLARE @errorDate NVARCHAR(100) = 'Package cannot be booked outside
package available dates'
                RAISERROR (@errorDate, 16, 1) WITH NOWAIT;
            END
        COMMIT TRANSACTION
         -- fetch next row from cursor
        FETCH NEXT FROM AdvertisedDates INTO @packageID, @startDate, @endDate
    END
    END TRY
    --error handling
    BEGIN CATCH
        SELECT ERROR_MESSAGE() AS ErrorMessage,
        ERROR_SEVERITY() AS ErrorSeverity,
        ERROR_STATE() AS ErrorState
        ROLLBACK TRANSACTION
        CLOSE AdvertisedDates
        DEALLOCATE AdvertisedDates
        RETURN 0
    END CATCH
    CLOSE AdvertisedDates
    DEALLOCATE AdvertisedDates
    -- Check that the quantity of a package is greater than 0
    BEGIN TRY
    BEGIN TRANSACTION
    IF EXISTS( SELECT *
            FROM @bookedPackages
            WHERE qtyBooked < 1)
   BEGIN
            DECLARE @negativeQty NVARCHAR(100) = 'Package quantity must be greater than
0'
            RAISERROR (@negativeQty, 11, 1) WITH NOWAIT;
   END
   COMMIT TRANSACTION
   END TRY
   --error handling
   BEGIN CATCH
      SELECT ERROR_MESSAGE() AS ErrorMessage,
      ERROR_SEVERITY() AS ErrorSeverity,
      ERROR_STATE() AS ErrorState
      ROLLBACK TRANSACTION
      RETURN 0
   END CATCH
   -- Check the qtyBooked is less than the capacity available
   DECLARE CheckCapacity CURSOR FOR
   SELECT bp.packageID, bp.qtyBooked, bp.startDate, bp.endDate
   FROM @bookedPackages bp

   DECLARE @currentpackageID CHAR(10);
   DECLARE @currentDate DATE;
   DECLARE @capacity INT;
   DECLARE @serviceID CHAR(7);
   DECLARE @qtyBooked INT;
   DECLARE @currBooking INT;

   OPEN CheckCapacity;
   FETCH NEXT FROM CheckCapacity INTO @packageID, @qtyBooked, @startDate,
@endDate;

   WHILE @@FETCH_STATUS = 0
   BEGIN
      DECLARE serviceItems CURSOR FOR
         SELECT psi.serviceID
         FROM PackageServiceItem psi
         WHERE psi.packageID = @packageID

      OPEN serviceItems;
      FETCH NEXT FROM serviceItems INTO @serviceID;

      WHILE @@FETCH_STATUS = 0
      BEGIN
         --Loop through each date from start to end date
         SET @currentDate = @startDate
         WHILE @currentDate <= @endDate
         BEGIN
            --Get the capacity
            SELECT @capacity = capacity
            FROM ServiceItem s
            WHERE s.serviceID = @serviceID
            --subtract any existing bookings
            SELECT @qtyBooked = ISNULL(SUM(b.qtyBooked), 0)
            FROM Booking b
            WHERE b.packageID = @packageID
            AND b.startDate <= @currentDate
            AND b.endDate >= @currentDate;

            SET @capacity = @capacity - @qtyBooked;

            --subtract the booking we want to allocate throw error if capacity < 0
            SELECT @currBooking = ISNULL(SUM(bp.qtyBooked), 0)
            FROM @bookedPackages bp
            WHERE bp.packageID = @packageID
            AND bp.startDate <= @currentDate
            AND bp.endDate >= @currentDate;

            SET @capacity = @capacity - @currBooking;

            --RAISEERROR if @Capacity is < 0
            IF @capacity < 0
            BEGIN
               DECLARE @errorCapacity NVARCHAR(500);
               SET @errorCapacity = 'Capacity for service item ' + CONVERT(NVARCHAR(10),
@serviceID)
                  + ' on date ' + CONVERT(NVARCHAR(10), @currentDate) + ' is fully booked';
               RAISERROR(@errorCapacity, 11, 1);
               RETURN;
            END;
            --Step forward one day in the date range
            SET @currentDate = DATEADD(day, 1, @currentDate);
         END;
         --fetch next service item
         FETCH NEXT FROM serviceItems INTO @serviceID;
      END;

      CLOSE serviceItems;
      DEALLOCATE serviceItems;
      --fetch next package
      FETCH NEXT FROM CheckCapacity INTO @packageID, @qtyBooked, @startDate,
@endDate;
   END;
```

```
    CLOSE CheckCapacity
    DEALLOCATE CheckCapacity



    -- Set reservation ID
    SET @reservationID = CONCAT('R', ABS(CHECKSUM(NEWID())))
    WHILE EXISTS(SELECT * FROM Reservation WHERE reservationID = @reservationID)
       BEGIN
          SET @reservationID = CONCAT('R', ABS(CHECKSUM(NEWID())))
       END
    -- Set customer ID
       DECLARE @customerID CHAR(10) = CONCAT('C', ABS(CHECKSUM(NEWID())))
       WHILE EXISTS(SELECT * FROM Reservation WHERE reservationID = @reservationID)
       BEGIN
          SET @customerID = CONCAT('C', ABS(CHECKSUM(NEWID())))
       END
    -- Insert into Customer
    INSERT INTO Customer VALUES (@customerID, @custName, @custPhone, @custEmail);
    -- Insert into Customer Address
    INSERT INTO CustomerAddress VALUES (@customerID, @streetNo, @streetName, @city,
@postcode, @country);
    -- Insert into Reservation
    INSERT INTO Reservation VALUES (@reservationID, @customerID, NULL, DEFAULT)
    -- Insert into Booking table
    INSERT INTO Booking(reservationID, packageID, qtyBooked, startDate, endDate)
    SELECT @reservationID, packageID, qtyBooked, startDate, endDate
    FROM @bookedPackages
    --Insert into Guest
    INSERT INTO ReservationGuest(reservationID, name, phone, email, streetNo, streetName,
city, postcode, country)
    SELECT @reservationID, name, phone, email, streetNo, streetName, city, postcode,
country
    FROM @guests

    -- Update the pricing in the reservation table
    UPDATE Reservation
    SET totalPrice = (
    SELECT SUM(p.advPrice * b.qtyBooked)
    FROM Booking b
    JOIN Package p ON b.packageID = p.packageID
    WHERE b.reservationID = @reservationID) WHERE reservationID = @reservationID

    -- Create a payment for the deposit
    DECLARE @totalPrice DECIMAL(18, 2)
    SELECT @totalPrice = totalPrice * 0.25
    FROM Reservation
    WHERE reservationID = @reservationID
    INSERT INTO Payment VALUES (@reservationID, @totalPrice, GETDATE())
END
```