

COMP3350 group 12 assignment 1 section 3-5

Group Members:

Victor Chua Jia Zhi Student ID: c3418248

Sander Fabian Student ID: c3418982

Section 3: Each table contain the screenshot of how the table was being created and altering table with foreign constraints by following our relational schema, population of table, and the result of the population. As mentioned in section 1-2, since we have a lot of foreign keys, many of the constraints would be set to default, which is on update no action, on delete no action.

Creating Capacity Table.

```
-- creating table for capacity
create table Capacity(
  capacityID INT primary key,
  name varchar(50) not null,
  size INT not null);
```

Populating data into capacity Table.

```
Insert into Capacity(capacityID, name, size) values
(1, 'Small', 2),
(2, 'Medium', 4),
(3, 'Small-Capacity', 100),
(4, 'Medium-Capacity', 500),
(5, 'Large-Capacity', 1000),
(6, 'Hotel-Capacity', 1500);
```

Results

	capacityID	name	size
1	1	Small	2
2	2	Medium	4
3	3	Small-Capacity	100
4	4	Medium-Capacity	500
5	5	Large-Capacity	1000
6	6	Hotel-Capacity	1500

Creating Location Table

```
-- creating table for location
create table Location (
  locationID INT IDENTITY(1,1) PRIMARY KEY,
  city varchar(50) not null,
  country varchar(50) not null,
);
```

Populating data into Location Table.

```

Insert into Location(country,city) values
('Singapore', 'Singapore') ,
('Australia', 'Sydney'),
('India', 'Delhi'),
('Australia', 'Graman'),
('Malaysia', 'Johor');

```

Result

	locationID	city	country
1	1	Singapore	Singapore
2	2	Sydney	Australia
3	3	Delhi	India
4	4	Graman	Australia
5	5	Johor	Malaysia

Creating Hotel Address Table

```

-- creating table for Hotel address
create table HotelAddress(
hotelAddressID INT primary key,
unitNo varchar(50) not null,
street varchar(50) not null,
postalCode varchar(50) not null,
locationID int
);
Alter table HotelAddress
add constraint FK_HotelAddress_Location foreign key (locationID) references Location(locationID)

```

Populating data into Hotel Address Table

```

Insert into HotelAddress(hotelAddressID, unitNo, street, postalCode, locationID) values
(1, 01-231, '67 Passage Avenue', 21313, 4),
(2, 02-123, 'Marina Square', 58127, 1),
(3, 01-213, '85 Noalimba Avenue', 123141,2),
(4, 17-123, 'New Delhi St 12', 012931, 3),
(5, 15-123, 'Johor ave 9', 341121, 5)

```

Result

	hotelAddressID	unitNo	street	postalCode	locationID
1	1	01-231	67 Passage Avenue	21313	4
2	2	02-123	Marina Square	58127	1
3	3	01-213	85 Noalimba Avenue	123141	2
4	4	17-123	New Delhi St 12	12931	3
5	5	15-123	Johor ave 9	341121	5

Creating Hotel Table

```

-- creating table for hotel and adding FK constraints
create TABLE Hotel(
  hotelID INT Primary key not null
  , name VARCHAR(50) not null,
  description varchar(500) null,
  tellNo varchar(50) not null,
  hotelAddressID INT not null,
  capacityID int not null
);

Alter table Hotel
add constraint FK_Hotel_HotelAddress foreign key (hotelAddressID) references HotelAddress(hotelAddressID)

Alter table Hotel
add constraint FK_Hotel_Capacity foreign key (capacityID) references Capacity(capacityID)
on update cascade

```

Populating data into Hotel Table

```

Insert into Hotel ( hotelID, name, description, tellNo, capacityID, hotelAddressID) values
(1, 'Queen Hotel', '5 star hotel', 95313456, 6, 1),
(2, 'King Hotel', '4 star hotel', 90283184, 6, 2),
(3, 'Joker Hotel', '3 star hotel', 8727138, 6, 3),
(4, 'Knight Hotel', '5 star hotel', 9812371, 6, 4),
(5, 'Citizen Hotel', '4 star hotel', 9128371, 6, 5);

```

Result

	hotelID	name	description	tellNo	hotelAddressID	capacityID
1	1	Queen Hotel	5 star hotel	95313456	1	6
2	2	King Hotel	4 star hotel	90283184	2	6
3	3	Joker Hotel	3 star hotel	8727138	3	6
4	4	Knight Hotel	5 star hotel	9812371	4	6
5	5	Citizen Hotel	4 star hotel	9128371	5	6

Creating Department Table

```

-- creating table for department and adding FK constraints
create table Department(
  departmentID INT primary key,
  name varchar(50) not null,
  area varchar(150) not null);

```

Populating into Department Table

```

Insert into Department(departmentID, name, area) values
(1, 'MarketingTeam', 'Marketing'),
(2, 'FacilitiesTeam', 'Facilities'),
(3, 'FoodManagement', 'F&B'),
(4, 'HouseKeepingTeam', 'Housekeeping'),
(5, 'DiscountTeam', 'DiscountManagement')

```

Result

	departmentID	name	area
1	1	MarketingTeam	Marketing
2	2	FacilitiesTeam	Facilities
3	3	FoodManagement	F&B
4	4	HouseKeepingTeam	Housekeeping
5	5	DiscountTeam	DiscountManagement

Creating Employee Address Table

```
-- creating table for employee address and adding FK constraints
create table EmployeeAddress(
  employeeAddressID INT primary key,
  unitNo varchar(50) not null,
  street varchar(50) not null,
  postalCode varchar(50) not null,
  locationID int
);
Alter table EmployeeAddress
add constraint FK_EmployeeAddress_Location foreign key (locationID) references Location(locationID)
```

Populating into Employee Address Table

```
Insert into EmployeeAddress(employeeAddressID,unitNo, street, postalCode, locationID) values
(1, 05-10, 'Choa Chua Kang Crescent', 68210, 1),
(2, 09-11, 'Sydney street 10', 12931, 2),
(3, 10-406, 'New Delhi ave 11', 20984, 3),
(4, 11-120, 'Woodlands Ave 12', 61821, 2),
(5, 17-140, 'Kranji St 08', 612818, 2);
```

Result

	employeeAddressID	unitNo	street	postalCode	locationID
1	1	05-10	Choa Chua Kang Crescent	68210	1
2	2	09-11	Sydney street 10	12931	2
3	3	10-406	New Delhi ave 11	20984	3
4	4	11-120	Woodlands Ave 12	61821	2
5	5	17-140	Kranji St 08	612818	2

Creating Employee Table

```
-- creating table for employee and adding FK constraints
create table Employee(
  employeeID INT IDENTITY(1,1) PRIMARY KEY,
  name varchar(50) not null,
  dob date,
  tellNo varchar(50) not null,
  employeeAddressID INT not null,
  hotelID INT not null,
  departmentID INT not null);

Alter table Employee
add constraint FK_Employee_Address foreign key (employeeAddressID) references EmployeeAddress(employeeAddressID)

Alter table Employee
add constraint FK_Employee_Department foreign key (departmentID) references Department(departmentID)
on update cascade

Alter table Employee
add constraint FK_Employee_Hotel foreign key (hotelID) references Hotel(hotelID)
on update cascade
```

Populating into Employee Table

```
Insert into Employee(name, tellNo, dob, employeeAddressID, hotelID, departmentID) values
('Chan Wai Leng', 9271731, '1999-12-10', 1, 2, 1),
('Victor Chua', 98127823, '1997-11-10', 2, 1, 3),
('Sander Fabian', 9182731, '1996-11-18', 3, 3, 2),
('Gina Lee', 96982171, '1995-11-10', 4, 4, 4),
('Girvin Chua', 9182472, '1994-06-10', 5, 5, 5)
```

Result

	employeeID	name	dob	tellNo	employeeAddressID	hotelID	departmentID
1	1	Chan Wai Leng	1999-12-10	9271731	1	2	1
2	2	Victor Chua	1997-11-10	98127823	2	1	3
3	3	Sander Fabian	1996-11-18	9182731	3	3	2
4	4	Gina Lee	1995-11-10	96982171	4	4	4
5	5	Girvin Chua	1994-06-10	9182472	5	5	5

Creating Manager Table

```
-- creating table for manager and adding FK constraints
create table Manager(
  managerID INT IDENTITY(1,1) PRIMARY KEY,
  employeeID INT unique);

Alter table Manager
add constraint FK_Manager_Employee foreign key (employeeID) references Employee(employeeID)
on update cascade
on delete cascade
```

Populating into Manager Table

```
Insert into Manager(employeeID) values
(3), (2), (1);
```

Result

	managerID	employeeID
1	3	1
2	2	2
3	1	3

Creating HeadOffice Table

```
-- creating table for headOffice and adding FK constraints
create table HeadOffice
(
  headOfficeID int primary key,
  tellNo varchar(50) ,
  departmentID INT not null);

Alter table HeadOffice
add constraint FK_HeadOffice_Department foreign key (departmentID) references Department(departmentID)
on update cascade
```

Populating into HeadOffice Table

```
Insert into HeadOffice(headOfficeID, tellNo, departmentID) values
(1, 9812131, 1),
(2, 9845511, 2),
(3, 8172318, 3),
(4, 8247192, 4),
(5, 8912831, 5);
```

Result

	headOfficeID	tellNo	departmentID
1	1	9812131	1
2	2	9845511	2
3	3	8172318	3
4	4	8247192	4
5	5	8912831	5

Creating Facility Table

```
-- creating table for facility and adding FK constraints
create table Facility(
  facilityID INT primary key,
  name varchar(50) not null,
  description varchar(500) null,
  statusCode varchar(20) null,
  hotelID INT not null);

Alter table Facility
add constraint FK_Facility_Hotel foreign key (hotelID) references Hotel(hotelID)
on update cascade
```

Populating into Facility Table

```
Insert into Facility(facilityID, name, description, statusCode, hotelID) values
(1, 'Swimming Pool', 'Swimming pools with different sizes', 'Active', 1),
(2, 'Gym', 'Gym equipments for gymers', 'InActive', 2),
(3, 'Rooms', 'Rooms for guests', 'Active', 3),
(4, 'BallRoom', 'Venue for many purposes', 'Active', 4),
(5, 'HallRoom', 'Venue for weddings', 'Active', 5);
```

Result

	facilityID	name	description	statusCode	hotelID
1	1	Swimming Pool	Swimming pools with different sizes	Active	1
2	2	Gym	Gym equipments for gymers	InActive	2
3	3	Rooms	Rooms for guests	Active	3
4	4	BallRoom	Venue for many purposes	Active	4
5	5	HallRoom	Venue for weddings	Active	5

Creating Facility Type Table

```
-- creating table for facilityType and adding FK constraints
create table FacilityType(
facilityTypeID int primary key,
name varchar(50) not null,
description varchar(500) null,
quantity INT not null,
capacityID INT not null,
facilityID INT not null);

Alter table FacilityType
add constraint FK_FacilityType_Capacity foreign key (capacityID) references Capacity(capacityID)

Alter table FacilityType
add constraint FK_FacilityType_Facility foreign key (facilityID) references Facility(facilityID)
on update cascade
on delete cascade
```

Populating into Facility Type Table

```
Insert into FacilityType(facilityTypeID, name, description, capacityID, facilityID, quantity) values
(1, 'Swimming Pool for kids', 'Pools suitable for kids!', 3, 1, 1),
(2, 'Gym Spa room', 'Spa-ing after a good session in the gym', 3, 2, 5),
(3, 'Couple suite room', 'Nice room for couples', 1, 3, 200),
(4, 'Ballroom for Restaurants', 'a good room for serving', 4, 4, 1),
(5, 'Wedding Hall Room', 'a room catered for weddings', 5, 5, 1);
```

Result

	facilityTypeID	name	description	quantity	capacityID	facilityID
1	1	Swimming Pool for kids	Pools suitable for kids!	1	3	1
2	2	Gym Spa room	Spa-ing after a good...	5	3	2
3	3	Couple suite room	Nice room for couples	200	1	3
4	4	Ballroom for Restaur...	a good room for ser...	1	4	4
5	5	Wedding Hall Room	a room catered for w...	1	5	5

Creating Base Currency Table

```
-- creating table for base currency and adding FK constraints
create table BaseCurrency(
  currencySymbol varchar(10) primary Key,
  name varchar(50));
```

Populating into Base Currency Table

```
Insert into BaseCurrency( currencySymbol, name) values
('SGD', 'Singapore Dollar'),
('AUD', 'Australian Dollar'),
('RM', 'Malaysian Dollar'),
('USD', 'United State Dollar'),
('YEN', 'Japan Dollar'),
('INR', 'Indian Rupee')
;
```

Result

	currencySymbol	name
1	AUD	Australian Dollar
2	INR	Indian Rupee
3	RM	Malaysian Dollar
4	SGD	Singapore Dollar
5	USD	United State Dollar
6	YEN	Japan Dollar

Creating Service Category Table

```
-- creating table for serviceCategory and adding FK constraints
create table ServiceCategory(
  serviceCatID INT primary key,
  name varchar(50) not null,
  description varchar (500) null,
  serviceType varchar (50) not null,
  hotelID int not null
);
Alter table ServiceCategory
add constraint FK_ServiceCategory_Hotel foreign key (hotelID) references Hotel(hotelID)
on update cascade
on delete cascade
```

Populating into Service Category Table

```
Insert into ServiceCategory( serviceCatID, name , description, serviceType, hotelID) values
(1, 'Accomodation', 'For guest enjoyment purposes', 'Accomodation', 1),
(2, 'Accomodation', 'For guest enjoyment purposes', 'Accomodation', 2),
(3, 'Accomodation', 'For guest enjoyment purposes', 'Accomodation', 3),
(4, 'Accomodation', 'For guest enjoyment purposes', 'Accomodation', 4),
(5, 'Accomodation', 'For guest enjoyment purposes', 'Accomodation', 5),
(6, 'Event', 'For special events', 'Events', 1),
(7, 'Event', 'For special events', 'Events', 2),
(8, 'Event', 'For special events', 'Events', 3),
(9, 'Event', 'For special events', 'Events', 4),
(10, 'Event', 'For special events', 'Events', 5);
```


Result

	serviceCatID	name	description	serviceType	hotelID
1	1	Accomodation	For guest enjoyment purposes	Accomodation	1
2	2	Accomodation	For guest enjoyment purposes	Accomodation	2
3	3	Accomodation	For guest enjoyment purposes	Accomodation	3
4	4	Accomodation	For guest enjoyment purposes	Accomodation	4
5	5	Accomodation	For guest enjoyment purposes	Accomodation	5
6	6	Event	For special events	Events	1
7	7	Event	For special events	Events	2
8	8	Event	For special events	Events	3
9	9	Event	For special events	Events	4
10	10	Event	For special events	Events	5

Creating Service Item Table

```
-- creating table for service item and adding FK constraints
create table ServiceItem(
  serviceItemID INT IDENTITY(1,1) PRIMARY KEY,
  name varchar (50) not null,
  description varchar (500) null,
  restriction varchar (200) null,
  note varchar (200) null,
  comment varchar (250) null,
  availableTime varchar(200),
  quantity int not null,
  baseCost decimal not null,
  statusCode VARCHAR(20),
  baseCurrency varchar(10) not null,
  capacityID INT not null,
  facilityTypeID INT null,
  serviceCatID INT
);
Alter table ServiceItem
add constraint FK_ServiceItem_FacilityType foreign key (facilityTypeID) references FacilityType(facilityTypeID)
on update cascade
on delete cascade

Alter table ServiceItem
add constraint FK_ServiceItem_BaseCurrency foreign key (baseCurrency) references BaseCurrency(currencySymbol)

Alter table ServiceItem
add constraint FK_ServiceItem_ServiceCategory foreign key (serviceCatID) references ServiceCategory(serviceCatID)

Alter table ServiceItem
add constraint FK_ServiceItem_Capacity foreign key (capacityID) references Capacity(capacityID)
```

Populating into Service Item Table

```
Insert into ServiceItem( name, description, restriction, note, comment, availableTime, quantity, baseCost, statusCode, baseCurrency, capacityID, facilityTypeID, serviceCatID) values
('Food services', 'Serving Food', 'Restriction 1', 'Note1', 'comment1', '9:00 AM - 5:00 PM', 250, 2000, 'Active', 'AUD', 3, 1, 6),
('Music services', 'Concert', 'Restriction1', 'Note2', 'comment2', '10:00 - 5:30 PM', 100, 750, 'Active', 'AUD', 3, 2, 6),
('Entertainment services', 'Movie Showtime', 'Restriction3', 'Note3', 'comment3', '10:00 AM - 2:00 PM', 150, 300, 'Inactive', 'AUD', 3, 3, 6),
('Couple room services', 'Rental for couples', 'Restriction4', 'Note4', 'comment4', '10:00 AM - 10:00 AM', 75, 30000, 'Active', 'AUD', 1, 3, 1),
('Family room services', 'Rental for family', 'Restriction5', 'Note5', 'comment5', '08:00 AM - 12:00PM', 75, 45000, 'Active', 'AUD', 2, 3, 1),
('Wedding services', 'Catering to wedding', 'Restriction6', 'Note6', 'comment6', '1:00PM - 10:00 PM', 2, 15000, 'Inactive', 'AUD', 5, 5, 6)
```

Result

serviceItemID	name	description	restriction	note	comment	availableTime	quantity	baseCost
1	Food services	Serving Food	Restriction 1	Note	comment	9:00 AM - 5:00 PM	250	2000
2	Music services	Concert	Restriction1	Note2	comment	10:00 - 5:30 PM	100	750
3	Entertainment services	Movie Showtime	Restriction3	Note3	comment	10:00 AM - 2:00 PM	150	300
4	Couple room services	Rental for couples	Restriction4	Note4	comment4	10:00 AM - 10:00 AM	75	30000
5	Family room services	Rental for family	Restriction5	Note5	comment5	08:00 AM - 12:00PM	75	45000
6	Wedding services	Catering to wedding	Restriction6	Note6	comment6	1:00PM - 10:00 PM	2	15000

statusCode	baseCurrency	capacityID	facilityTypeID	serviceCatID
Active	AUD	3	1	6
Active	AUD	3	2	6
InActive	AUD	3	3	6
Active	AUD	1	3	1
Active	AUD	2	3	1
InActive	AUD	5	5	6

Creating Advertised Package Table

```
-- creating table for AdvertisedPackage and adding FK constraints
create table AdvertisedPackage(
adPackID INT IDENTITY(1,1) PRIMARY KEY,
name varchar (50) not null,
description varchar(50) null,
price decimal (10,2) not null,
inclusion varchar(250) null,
exclusion varchar(250) null,
startDate date not null,
endDate date not null,
statusCode varchar(20) not null,
employeeID int not null,
currencySymbol varchar(10) not null);

Alter table AdvertisedPackage
add constraint FK_AdvertisedPackage_Employee foreign key (employeeID) references Employee(employeeID)
ON UPDATE NO ACTION
ON DELETE NO ACTION
Alter table AdvertisedPackage
add constraint FK_AdvertisedPackage_BaseCurrency foreign key (currencySymbol) references BaseCurrency(currencySymbol)
```

Populating into Advertised Package Table

```
Insert into AdvertisedPackage (name, description, price, currencySymbol, exclusion, inclusion, startDate, endDate, statusCode, employeeID) values
('Package1', 'Family bundle', 500, 'AUD', 'No buffet', 'Free usage of pool', '2023-06-15', '2023-06-20', 'Active', 1),
('Package2', 'Couple bundle', 350, 'AUD', 'No pool', 'Free buffet', '2023-06-18', '2023-06-25', 'Active', 2),
('Package3', 'Friend bundle', 300, 'AUD', 'No gym', 'Free taxis', '2023-07-19', '2023-07-25', 'InActive', 1),
('Wedding Package', 'Wedding bundle', 20000, 'AUD', 'No gym', 'Free stay for a week', '2023-07-20', '2023-07-25', 'InActive', 3);
```

Result

	adPackID	name	description	price	inclusion	exclusion	startDate	endDate	statusCode	employee
1	1	Package1	Family bundle	500.00	Free usage of pool	No buffet	2023-06-15	2023-06-20	Active	1
2	2	Package2	Couple bundle	350.00	Free buffet	No pool	2023-06-18	2023-06-25	Active	2
3	3	Package3	Friend bundle	300.00	Free taxis	No gym	2023-07-19	2023-07-25	InActive	1
4	4	Wedding Package	Wedding bundle	20000.00	Free stay for a week	No gym	2023-07-20	2023-07-25	InActive	3

currencySymbol
AUD
AUD
AUD
AUD

Creating Advertised Service Table

```
-- creating a composite FK primary key table as we want to be able to identify what services in advertisedpackage
create table AdvertisedService(
  adPackID INT,
  serviceItemID INT,
  quantity int not null,
  primary key ( adPackID, serviceItemID));
```

Populating into Advertised Service Table

```
Insert into AdvertisedService(adPackID,serviceItemID, quantity) values
(1,1,2),
(1,2, 1),
(1,5, 1);
```

Result

	adPackID	serviceItemID	quantity
1	1	1	2
2	1	2	1
3	1	5	1

Creating Grace Period Table

```
-- creating a table for grace period and adding constraint as we are able to references to which advertisedpackage
create table GracePeriod (
  gracePeriodID int primary key,
  durationDays varchar(50) not null,
  adPackID int not null
);
```

Populating into Grace Period Table

```
Insert into GracePeriod(gracePeriodID, durationDays, adPackID) values
(1, '1 week', 1),
(2, '2 week', 2),
(3, '3 week', 3),
(4, '4 week', 4);
```

Result

	gracePeriodID	durationDays	adPackID
1	1	1 week	1
2	2	2 week	2
3	3	3 week	3
4	4	4 week	4

Creating Customer Guest Address Table

```
-- creating table for customer guest address and adding FK constraints
create table CustomerGuestAddress(
customerAddressID INT IDENTITY(1,1) primary key,
unitNo varchar(50) not null,
street varchar(50) not null,
postalCode varchar(50) not null,
locationID int
);

Alter table CustomerGuestAddress
add constraint FK_CustomerGuestAddress_Location foreign key (locationID) references Location (locationID)
```

Populating into Customer Guest Address Table

```
Insert into CustomerGuestAddress( unitNo, street ,postalCode, locationID) values
(09-10, 'Bukit Batok Ave 1', '674812', 1),
(05-19, 'Commonwealth Ave 9', '93182', 1),
(03-12, 'Clementi St 90', '19283', 1),
(02-20, 'Chinese Garden St 99', '19281', 1);
```

Result

	customerAddressID	unitNo	street	postalCode	locationID
1	1	09-10	Bukit Batok Ave 1	674812	1
2	2	05-19	Commonwealth Ave 9	93182	1
3	3	03-12	Clementi St 90	19283	1
4	4	02-20	Chinese Garden St 99	19281	1

Creating Customer Table

```
-- creating table for customer and adding FK constraints
create table Customer(
customerID INT IDENTITY(1,1) PRIMARY KEY,
name varchar (50) not null,
DOB date not null,
tellNo varchar(50) not null,
email varchar (50) not null,
customerAddressID int);

Alter table Customer
add constraint FK_Customer_CustomerGuestAddress foreign key (customerAddressID) references CustomerGuestAddress(customerAddressID)
ON UPDATE CASCADE
ON DELETE NO ACTION
```

Populating into Customer Table

```
Insert into Customer(name,DOB, tellNo, email, customerAddressID) values
('Carol', '1999-12-10', 8317317, 'Carol@gmail.com', 1),
('Lebron', '1979-12-10', 8418912, 'Lebron@gmail.com', 2),
('Larry', '1989-11-05', 9813881, 'larry@gmail.com', 3),
('Crystal', '1998-07-05', 918281, 'crystal@gmail.com', 4);
```

Result

	customerID	name	DOB	tellNo	email	customerAddressID
1	1	Carol	1999-12-10	8317317	Carol@gmail.com	1
2	2	Lebron	1979-12-10	8418912	Lebron@gmail.com	2
3	3	Larry	1989-11-05	9813881	larry@gmail.com	3
4	4	Crystal	1998-07-05	918281	crystal@gmail.com	4

Creating Reservation Table

```
-- creating table for reservation and adding FK constraints
create table Reservation(
reservationID INT IDENTITY(1,1) PRIMARY KEY,
payment varchar(50) not null,
deposit varchar(10) not null,
customerID int not null,
hotelID int not null,
employeeID int null
);
Alter table Reservation
add constraint FK_Reservation_Customer foreign key (customerID) references Customer(customerID)
on update cascade
Alter table Reservation
add constraint FK_Reservation_Employee foreign key (employeeID) references Employee(employeeID)
on update cascade
Alter table Reservation
add constraint FK_Reservation_Hotel foreign key (hotelID) references Hotel(hotelID)
```

Populating into Reservation Table

```
Insert into Reservation( payment, deposit, customerID, hotelID, employeeID) values
('Visa', '250', 1, 1, 1),
('Master', '80', 2, 1, 1),
('GrabPay', '100', 3, 1, 2),
('UOB', '600', 4, 1, 3);
```

Result

	reservationID	payment	deposit	customerID	hotelID	employeeID
1	1	Visa	250	1	1	1
2	2	Master	80	2	1	1
3	3	GrabPay	100	3	1	2
4	4	UOB	600	4	1	3

Creating Guest Table

```
-- creating table for Guest and adding FK constraints which customer it is associated with
create table Guest(
  guestID INT IDENTITY(1,1) PRIMARY KEY,
  name varchar (50) not null,
  DOB date not null,
  customerID int not null,
  tellNo varchar(50) null,
  email varchar(50) null,
  customerAddressID INT
);

Alter table Guest
add constraint FK_Guest_CustomerGuestAddress foreign key (customerAddressID) references CustomerGuestAddress (customerAddressID)

Alter table Guest
add constraint FK_Guest_Customer foreign key (customerID) references Customer(customerID)
on update cascade
on delete cascade
```

Populating into Guest Table

```
Insert into Guest(name,DOB,tellNo,customerID,email,customerAddressID) values
('Gabriel','1999-12-10',8271722, 2, 'gabriel@gmail.com', 3),
('Sabrina','1999-11-10', 8399138,3, 'sabrina@gmail.com', 2),
('Rita', '1997-11-07', 89139104, 4, 'rita@gmail.com',4);
```

Result

	guestID	name	DOB	customerID	tellNo	email	customerAddressID
1	1	Gabriel	1999-12-10	2	8271722	gabriel@gmail.com	3
2	2	Sabrina	1999-11-10	3	8399138	sabrina@gmail.com	2
3	3	Rita	1997-11-07	4	89139104	rita@gmail.com	4

Creating Booking Table

```
-- creating table for Booking and adding FK constraints
create table Booking(
  bookingID int IDENTITY(1,1) PRIMARY KEY,
  quantity int not null,
  startDate date,
  endDate date,
  reservationID INT not null,
  adPackID int not null,
  customerID int not null,
  facilityTypeID int null
);

Alter table Booking
add constraint FK_Booking_Reservation foreign key (reservationID) references Reservation(reservationID)

Alter table Booking
add constraint FK_Booking_AdvertisedPackage foreign key (adPackID) references AdvertisedPackage(adPackID)

Alter table Booking
add constraint FK_Booking_Customer foreign key (customerID) references Customer(customerID)
on update cascade

Alter table Booking
add constraint FK_Booking_FacilityType foreign key (facilityTypeID) references FacilityType(facilityTypeID)
```

Populating into Booking Table

```
Insert into Booking(quantity,startDate, endDate,adPackID,customerID,reservationID,facilityTypeID) values
(2, '2023-06-18', '2023-06-20',3, 2,2, 1),
(2, '2023-06-18', '2023-06-20',2, 3,3, 4),
(1, '2023-06-18', '2023-06-20',1,1,1, 3);
```

Result

	bookingID	quantity	startDate	endDate	reservationID	adPackID	customerID	facilityTypeID
1	1	2	2023-06-18	2023-06-20	2	3	2	1
2	2	2	2023-06-18	2023-06-20	3	2	3	4
3	3	1	2023-06-18	2023-06-20	1	1	1	3

Creating Transaction Billing Table

```
create table TransactionBilling(
transactionID int primary key,
bookingID int,
adPackID int);

Alter table TransactionBilling
add constraint FK_TransactionBilling_Booking foreign key (bookingID) references Booking(bookingID)

Alter table TransactionBilling
add constraint FK_TransactionBilling_AdvertisedPackage foreign key (adPackID) references AdvertisedPackage(adPackID)
```

Populating into Transaction Billing Table

```
Insert into TransactionBilling(transactionID,bookingID,adPackID) values
(1,1,1),
(2,1,2),
(3,1,3);
select * from TransactionBilling
```

Result

	transactionID	bookingID	adPackID
1	1	1	1
2	2	1	2
3	3	1	3

Creating Discount Table

```
-- creating a table for discount
create table Discount(
discountID int primary key,
discountPercentage varchar(50) null,
managerID int null,
headOfficeID int null);

Alter table Discount
add constraint FK_Discount_Manager foreign key (managerID) references Manager(managerID)
on update cascade
on delete no action

Alter table Discount
add constraint FK_Discount_HeadOffice foreign key (headOfficeID) references HeadOffice(headOfficeID)
```

Populating into Discount Table

```
Insert into Discount(discountID, discountPercentage, managerID, headOfficeID) values
(1, '10%', 3, null),
(2, '20%', 3, null),
(3, '25%', null, 5),
(4, '30%', null, 5),
(5, '35%', null, 5);
```

Result

	discountID	discountPercentage	managerID	headOfficeID
1	1	10%	3	NULL
2	2	20%	3	NULL
3	3	25%	NULL	5
4	4	30%	NULL	5
5	5	35%	NULL	5

Creating Billing Table

```
-- creating a table for billing and referencing to what bookingID it has and the extra charges
create table Billing(
billingID int primary key,
paymentDate date not null,
paymentmethod varchar(100) not null,
customerID int not null,
transactionID int not null,
reservationID int not null,
discountID int null,
);

Alter table Billing
add constraint FK_Billing_Reservation foreign key (reservationID) references Reservation(reservationID)

Alter table Billing
add constraint FK_Billing_Discount foreign key (discountID) references Discount(discountID)
on update cascade

Alter table Billing
add constraint FK_Billing_Customer foreign key (customerID) references Customer(customerID)

Alter table Billing
add constraint FK_Billing_TransactionBilling foreign key (transactionID) references TransactionBilling(transactionID)
```

Populating into Billing Table

```
Insert into Billing(billingID, paymentDate, paymentmethod, customerID, reservationID ,transactionID, discountID)
values
(1, '2023-06-18', 'Visa', 1, 1, 1, null),
(2, '2023-06-18', 'Master', 2, 1, 1, 1),
(3, '2023-05-18', 'UOB', 3, 3, 1, 2);
```

Result

	billingID	paymentDate	paymentmethod	customerID	transactionID	reservationID	discountID
1	1	2023-06-18	Visa	1	1	1	NULL
2	2	2023-06-18	Master	2	1	1	1
3	3	2023-05-18	UOB	3	1	3	2

SECTION 4 (Create Package) Stored Procedure

```
-- Creating TVP for serviceitems to be able to pass them into a list in Stored Procedure
CREATE TYPE dbo.ServiceItemList AS TABLE (
    serviceItemID int primary key,
    quantity INT
);

GO

-- Creating stored procedure
CREATE PROCEDURE usp_createPackage
    @packageName VARCHAR(50),
    @serviceItemList dbo.ServiceItemList READONLY,
    @description VARCHAR(50),
    @validPeriodStartDate DATE,
    @validPeriodEndDate DATE,
    @advertisedPrice DECIMAL(10, 2),
    @advertisedCurrency VARCHAR(10),
    @employeeID INT,
    @statusCode varchar(20),
    @advertisedPackageID INT OUTPUT
AS
BEGIN
    -- Executing try and catch for error handling
    BEGIN TRY
        -- Stopping the message of how many rows are being affected
        SET NOCOUNT ON;

        -- declaring variables inside SP
        DECLARE @packageID INT;
        INSERT INTO AdvertisedPackage (name, description, startDate, endDate, price, currencySymbol, employeeID, statusCode)
        VALUES (@packageName, @description, @validPeriodStartDate, @validPeriodEndDate, @advertisedPrice, @advertisedCurrency, @employeeID, @statusCode);
        -- Our table are created with identity(1,1) primary key, so in order to get the newly created ID we used scope identity
        SET @advertisedPackageID = SCOPE_IDENTITY();
        -- Inserting data into Advertised service, a composite key to be able to identify the advertisedpackage and their relations with the serviceitems
        INSERT INTO AdvertisedService (adPackID, serviceItemID, quantity)
        SELECT @advertisedPackageID, serviceItemID, quantity
        FROM @serviceItemList;
    END TRY
    BEGIN CATCH
        -- a simple error handling, to show the user what kind of error and the severity, state
        DECLARE @errorMessage nvarchar(200) = error_message();
        DECLARE @errorSeverity INT = error_severity();
        DECLARE @errorState INT = error_state();

        PRINT 'There is an error when executing' + @errorMessage;
        PRINT 'The error severity is : ' + @errorSeverity;
        PRINT 'The error state is : ' + @errorState;
    END CATCH
END
```

By following our assumption where advertised packages can be associated with more than one service item, we decided to create a new table called advertised services where it will show the association between advertised package and the service items and how many quantity it has. We have implemented try and catch for basic error handling. To pass in a list of items in the parameters, we used table valued parameter for read only purposes and to store the data of the list into our respective tables.

Test Package

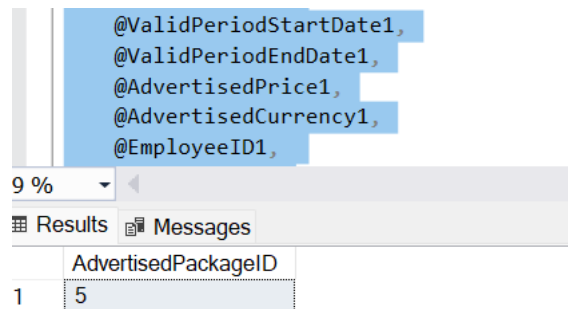
```
-- Declaring a TVP, to send a list inside a Stored procedure
DECLARE @serviceItems dbo.ServiceItemList;
INSERT INTO @serviceItems (serviceItemID,quantity)
VALUES (1,3),(2,2),(3,4);
DECLARE @PackageName1 VARCHAR(50) = 'Package Name';
DECLARE @Description1 VARCHAR(50) = 'Package Description';
DECLARE @ValidPeriodStartDate1 DATE = '2023-06-01';
DECLARE @ValidPeriodEndDate1 DATE = '2023-06-30';
DECLARE @AdvertisedPrice1 DECIMAL(10, 2) = 100.00;
DECLARE @AdvertisedCurrency1 VARCHAR(10) = 'USD';
DECLARE @EmployeeID1 INT = 1;
DECLARE @AdvertisedPackageID1 INT;
DECLARE @statusCode1 varchar(20) = 'Active';

-- Executing Stored Procedure named usp_createPackage
EXECUTE usp_createPackage
    @PackageName1,
    @serviceItems,
    @Description1,
    @ValidPeriodStartDate1,
    @ValidPeriodEndDate1,
    @AdvertisedPrice1,
    @AdvertisedCurrency1,
    @EmployeeID1,
    @statusCode1,
    @AdvertisedPackageID1 OUTPUT;

-- To show the output of the ID
SELECT @AdvertisedPackageID1 AS AdvertisedPackageID;
-- for testing purposes ---
SELECT * FROM AdvertisedPackage;
SELECT * FROM ServiceItem;
```

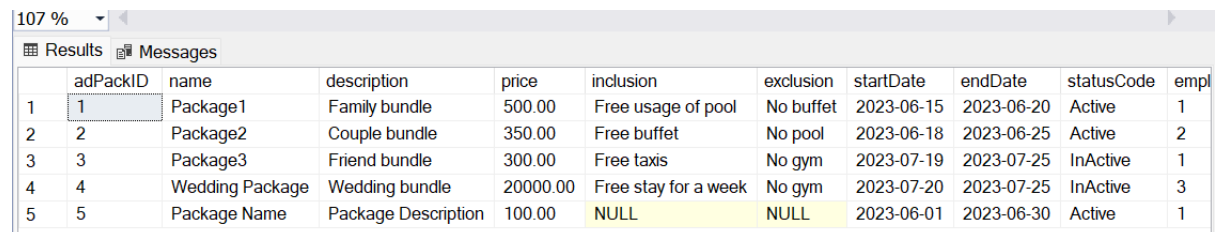
Result

Output variable for advertised package ID



AdvertisedPackageID
5

Newly inserted Advertised Package



	adPackID	name	description	price	inclusion	exclusion	startDate	endDate	statusCode	empl
1	1	Package1	Family bundle	500.00	Free usage of pool	No buffet	2023-06-15	2023-06-20	Active	1
2	2	Package2	Couple bundle	350.00	Free buffet	No pool	2023-06-18	2023-06-25	Active	2
3	3	Package3	Friend bundle	300.00	Free taxis	No gym	2023-07-19	2023-07-25	InActive	1
4	4	Wedding Package	Wedding bundle	20000.00	Free stay for a week	No gym	2023-07-20	2023-07-25	InActive	3
5	5	Package Name	Package Description	100.00	NULL	NULL	2023-06-01	2023-06-30	Active	1

Newly inserted Advertised Package ID associated with newly inserted service item list in advertised service table.

	adPackID	serviceItemID	quantity
1	1	1	2
2	1	2	1
3	1	5	1
4	5	1	3
5	5	2	2
6	5	3	4

Section 4 (Create Reservation) Stored Procedure

Creating table valued parameters to allow user to stored lists of data and pass it in the stored procedure as a script.

```
-- Creating a TVP type for advertisedPackage list input purposes
CREATE type bookingReservedItems as TABLE (
    advertisedPackageID INT,
    quantity INT,
    startDate date,
    endDate date
);

-- Creating a TVP type for Guest list input purposes
Create type customerGuestList as Table (
    guestName varchar(50),
    guestDOB date,
    guestStreet varchar(50),
    guestUnitNo varchar(50),
    guestPostalCode varchar(50),
    guestLocationID int,
    tellNo varchar(50),
    guestEmail varchar(50));

GO
```

```

CREATE PROCEDURE usp_makeReservation
@hotelID int,
@customerName VARCHAR(50),
@customerDob date,
@street VARCHAR(50),
@unitNo VARCHAR(50),
@postalCode Varchar(50),
@locationID varchar(50),
@tellNo VARCHAR(50),
@email VARCHAR(50),
@payment1 VARCHAR(50),
@bookingStartDate date,
@bookingEndDate date,
@bookedServices dbo.bookingReservedItems READONLY,
@customerGuestList dbo.customerGuestList READONLY,
@reservationID INT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @totalAmount DECIMAL(10, 2);
    DECLARE @depositDue DECIMAL(10, 2);
    DECLARE @capacityCheck TABLE (
        serviceItemId INT,
        remainingCapacity INT
    );

    DECLARE @customerIDz INT;
    DECLARE @bookingIDz INT;
    DECLARE @guestIDz INT;
    DECLARE @totalCapacity INT;
    DECLARE @totalReservations INT;
    DECLARE @customerAddressID INT;
    DECLARE @reservationIDList table (reservationID INT);
    DECLARE @facilityTypeID INT;

    -- Check the total capacity of the hotel by joining the Capacity table
    SELECT @totalCapacity = c.size
    FROM Hotel h
    INNER JOIN Capacity c ON h.capacityID = c.capacityID
    WHERE h.hotelID = @hotelID;

    -- Check total reservations and the dates in TVP booked services
    SELECT @totalReservations = COUNT(*)
    FROM Reservation
    WHERE hotelID = @hotelID
    AND EXISTS (
        SELECT 1
        FROM @bookedServices bs
        WHERE startDate <= bs.endDate
        AND endDate >= bs.startDate
    );

    -- For debugging purposes, to check outcome on the message panel
    PRINT 'Total Reservations: ' + CONVERT(VARCHAR(10), @totalReservations);

    -- Using if statement to check if totalReservation exceed over total capacity
    IF (@totalReservations + 1) > @totalCapacity
    BEGIN
        -- Raise an error if the reservation exceeds the hotel capacity (severity: 16)
        RAISERROR('The reservation exceeds the hotel capacity.', 16, 1);
        RETURN;
    END;

    -- calculating the total amount of the advertised packages and storing it in total amount
    SET @totalAmount = (
        SELECT SUM(br.quantity * ap.price)
        FROM @bookedServices br
        INNER JOIN AdvertisedPackage ap ON br.advertisedPackageID = ap.adPackID
    );

    -- for debug purposes to store in the message to see the output
    PRINT 'Total Amount: ' + CONVERT(VARCHAR(50), @totalAmount);

    -- Calculate 25% deposit of the total amount for deposit due and storing it in depositDue variable
    SET @depositDue = ISNULL(@totalAmount * 0.25, 0);
    PRINT 'Deposit Due: ' + CONVERT(VARCHAR(50), @depositDue);

```

```

-- Insert values into the CustomerGuestAddress table
INSERT INTO CustomerGuestAddress (unitNo, street, postalCode, locationID)
VALUES (@unitNo, @street, @postalCode, @locationID);

-- Since we are using an identity primary key, we are using scope identity to extract the latest generated ID when
SET @customerAddressID = SCOPE_IDENTITY();

-- Insert values into the Customer table with input variables
INSERT INTO Customer (name, customerAddressID, tellNo, email, DOB)
VALUES (@customerName, @customerAddressID, @tellNo, @email, @customerDob);

-- Since we are using an identity primary key, we are using scope identity to extract the latest generated ID when
SET @customerIDz = SCOPE_IDENTITY();
-- Debug purposes
PRINT 'Customer ID: ' + CONVERT(VARCHAR(50), @customerIDz);

-- Insert into CustomerGuestAddress table for both customer and guest as they might be staying in the same apartment
INSERT INTO CustomerGuestAddress (unitNo, street, postalCode, locationID)
SELECT cg.guestUnitNo, cg.guestStreet, cg.guestPostalCode, cg.guestLocationID
FROM @customerGuestList cg;

-- Since we are using an identity primary key, we are using scope identity to extract the latest generated ID when
SET @customerAddressID = SCOPE_IDENTITY();

-- Insert into the Guest table for storing guest details
INSERT INTO Guest (name, customerAddressID, tellNo, email, customerID, DOB)
SELECT cg.guestName, @customerAddressID, cg.tellNo, cg.guestEmail, @customerIDz, cg.guestDOB
FROM @customerGuestList cg;

-- Since we are using an identity primary key, we are using scope identity to extract the latest generated ID when
SET @guestIDz = SCOPE_IDENTITY();

-- Get the facility type ID
SELECT @facilityTypeID = si.facilityTypeID
FROM ServiceItem si
JOIN AdvertisedService asv ON si.serviceItemID = asv.serviceItemID
WHERE asv.adPackID = (SELECT TOP 1 advertisedPackageID FROM @bookedServices br);

-- Insert data into the Reservation table and store the generated reservation ID in a list
INSERT INTO Reservation (deposit, customerID, payment, hotelID)
OUTPUT inserted.reservationID INTO @reservationIDList (reservationID)
SELECT @depositDue, @customerIDz, @payment1, @hotelID
FROM @bookedServices;

-- Since we are using an identity primary key, we are using scope identity to extract the latest generated ID when a value is placed
SET @reservationID = SCOPE_IDENTITY();
-- Select the reservation IDs from the @reservationIDList table variable to display the newly generated reservation ids
SELECT reservationID FROM @reservationIDList;

-- Inserting data with the reservation list and inserting the rest of the generated data. The reason why we used row_number is to un
INSERT INTO Booking (reservationID, adPackID, quantity, customerID, startDate, endDate, facilityTypeID)
SELECT r1.reservationID, bs.advertisedPackageID, bs.quantity, @customerIDz, @bookingStartDate, @bookingEndDate, si.facilityTypeID
FROM (
    SELECT ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS rowNumber, reservationID
    FROM @reservationIDList
) r1
JOIN (
    SELECT ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS rowNumber, advertisedPackageID, quantity
    FROM @bookedServices
) bs ON r1.rowNumber = bs.rowNumber
JOIN ServiceItem si ON bs.advertisedPackageID = si.serviceItemID;
END

```

Same as before, we have used table valued parameters to pass a list as a parameter which can only be read only. We have declared table in the stored procedure as following our assumption, we believe that a customer can have multiple reservations made so first we would go through basic error handling to check whether the number of reservations exceed the hotel capacity and if it doesn't exceed the stored procedure would run. The stored procedure would add the data into the respective tables, and we have used alias for a more efficient implementation.

Test Reservation Stored Procedure

```
-- Testing Stored Procedure for make reservation
DECLARE @customerName1 VARCHAR(50) = 'Steve Bob';
DECLARE @customerDob1 date = '1999-12-11';
DECLARE @tellNo1 VARCHAR(50) = '831931883';
DECLARE @email1 VARCHAR(50) = 'stevebob@gmail.com';
DECLARE @paymentType varchar(50) = 'Master';
DECLARE @reservationID1 INT;
DECLARE @hotelID123 int = 1
DECLARE @locationID2 int = 1
DECLARE @customerStreet2 varchar(50) = 'Marina Square Campus'
DECLARE @customerPostal varchar(50) = 938113
DECLARE @customerUnitNo varchar(50) = '10-501'
DECLARE @bookingStartDate1 date = '2023-06-18'
DECLARE @bookingEndDate1 date = '2023-06-25'

-- Table value parameter for AdvertisedPackages, in order to send it in as a readonly list
DECLARE @bookedServices1 dbo.bookingReservedItems;
INSERT INTO @bookedServices1 (advertisedPackageID, quantity, startDate, endDate)
VALUES (1, 1, '2023-06-15', '2023-06-17'),
       (2, 1, '2023-06-16', '2023-06-18');

-- Table value parameter for Customer Guest, using TVP to send it in as a readonly list
DECLARE @customerGuestList1 dbo.customerGuestList;
INSERT INTO @customerGuestList1 (guestName, guestDOB, guestUnitNo, guestStreet, guestPostalCode, tellNo, guestLocationID, guestEmail)
VALUES ('Guest 1', '1998-12-10', '10-111', 'Joo Koon Ave 1', 61821, 9812718, 1, 'guest1@example.com'),
       ('Guest 2', '1997-02-12', '09-123', 'Pasir ris Ave 3', 61341, 8491381, 1, 'guest2@example.com');

-- Executing makeReservation
EXEC usp_makeReservation
    @customerName = @customerName1,
    @customerDob = @customerDob1,
    @unitNo = @customerUnitNo,
    @street = @customerStreet2,
    @postalCode = @customerPostal,
    @locationID = @locationID2,
    @tellNo = @tellNo1,
    @email = @email1,
    @bookingStartDate = @bookingStartDate1,
    @bookingEndDate = @bookingEndDate1,
    @payment1 = @paymentType,
    @bookedServices = @bookedServices1,
    @customerGuestList = @customerGuestList1,
    @hotelID = @HotelID123,
    @reservationID = @reservationID1 OUTPUT
;

-- For testing purposes, For out parameter.
SELECT @reservationID1 AS LatestReservationID;
```

Result:

Since we assumed that customer could have more than one reservation, in the stored procedure make reservation we have displayed the list in the stored procedure to show case how many reservations was added and for testing purposes, we have included the output variable which collect the latest generated reservation ID as it is impossible to output a whole list.

Results Messages	
reservationID	
1	5
2	6

LatestReservationID	
1	6

Reservation newly inserted data with calculation of 25% deposit

	reservationID	payment	deposit	customerID	hotelID	employeeID
1	1	Visa	250	1	1	1
2	2	Master	80	2	1	1
3	3	GrabPay	100	3	1	2
4	4	UOB	600	4	1	3
5	5	Master	375.00	5	1	NULL
6	6	Master	375.00	5	1	NULL

Booking newly inserted data which show the newest reservation ID.

	bookingID	quantity	startDate	endDate	reservationID	adPackID	customerID	facilityTypeID
1	1	2	2023-06-18	2023-06-20	2	3	2	1
2	2	2	2023-06-18	2023-06-20	3	2	3	4
3	3	1	2023-06-18	2023-06-20	1	1	1	3
4	4	1	2023-06-18	2023-06-25	5	1	5	1
5	5	2	2023-06-18	2023-06-25	6	1	5	1

Updated guest list with the associated customer.

	guestID	name	DOB	customerID	tellNo	email	customerAddressID
1	1	Gabriel	1999-12-10	2	8271722	gabriel@gmail.com	3
2	2	Sabrina	1999-11-10	3	8399138	sabrina@gmail.com	2
3	3	Rita	1997-11-07	4	89139104	rita@gmail.com	4
4	4	Guest 1	1998-12-10	5	9812718	guest1@example.com	7
5	5	Guest 2	1997-02-12	5	8491381	guest2@example.com	7

Newly inserted customer data.

	customerID	name	DOB	tellNo	email	customerAddressID
1	1	Carol	1999-12-10	8317317	Carol@gmail.com	1
2	2	Lebron	1979-12-10	8418912	Lebron@gmail.com	2
3	3	Larry	1989-11-05	9813881	larry@gmail.com	3
4	4	Crystal	1998-07-05	918281	crystal@gmail.com	4
5	5	Steve Bob	1999-12-11	831931883	stevebob@gmail.com	5

If reservation is more than the capacity, an error message will pop up which will terminate the rest of the process.

```
(2 rows affected)
```

```
(2 rows affected)
```

```
Total Reservations: 8
```

```
Msg 50000, Level 16, State 1, Procedure usp_makeReservation, Line 62 [Batch Start Line 2]
```

```
The reservation exceeds the hotel capacity.
```

```
(1 row affected)
```