PART 2

Implementation of Happy Vally Campsite Database

JOE SHEPHERD

Contents

Table Creation and Inserting Data

For the implementation of the Robinsons family campsite database, the first step was to create the tables on SQL Server. Having looked at the feedback that was given to me it was clear that there were some errors in my original design. So before creating the tables, I made some changes to the entity relation diagram from part 1. All the tables were created using SQL statements which I have given screenshots for each table below.

Guest

Figure 1

```sql
CREATE TABLE Guest (
GuestID         INT PRIMARY KEY IDENTITY (1,1),
Title           VARCHAR(4)   NOT NULL
                    CONSTRAINT chkTitle
                    CHECK  (Title IN ('Mr', 'Mrs', 'Miss', 'Ms', 'Dr')),
FirstName       VARCHAR(20)  NOT NULL,
LastName        VARCHAR(20)  NOT NULL,
DOB             DATE         NOT NULL
                    CONSTRAINT chkBrithDate
                    CHECK (DOB >= '1916-01-01'),
StreetAddress   VARCHAR(100) NOT NULL,
PostCode        VARCHAR(10)  NOT NULL,
City            VARCHAR(20)  NOT NULL,
Country         VARCHAR(15)  NOT NULL,
PhoneNo         VARCHAR(20)  NOT NULL,
Email           VARCHAR(80)  NOT NULL UNIQUE
                    CONSTRAINT chkEmail
                    CHECK (Email LIKE '%_@__%.__%'),
LeadGuest       VARCHAR(3)   NOT NULL
                    CONSTRAINT leadGuest
                    CHECK (LeadGuest IN('YES','NO'))
);
```

Messages

Command(s) completed successfully.

Column Name	Data Type
GuestID	int
Title	varchar(4)
FirstName	varchar(20)
LastName	varchar(20)
DOB	date
StreetAddress	varchar(100)
PostCode	varchar(10)
City	varchar(20)
Country	varchar(15)
PhoneNo	varchar(20)
Email	varchar(80)
LeadGuest	varchar(3)

Figure 1 shows the Guest table. I added a leadguest column from my design in part 1 to show who is a lead guest. It has a check constraint to allow only values of 'yes' or 'no'. This table has three other check constraints to improve data integrity. chkTitle only allows values of a person's title to be inserted in the title column. chkBirthDate checks that people do not enter unrealistic values of a person's birth date as I figured no one would be over 100 years old. Finally, chkEmail ensures that the format of the data inserted into this column is the same format as an email, so it only can contain email addresses. At first I set this as unique, however later removed this constraint as it caused problems later in my design.

For the primary keys that contain IDs, an identity constraint is added so that the values are automatically inserted as a unique number starting from 1.

Figure 1.1

```sql
ALTER TABLE Guest
ADD CONSTRAINT LeadGuestDefault
DEFAULT 'NO' FOR LeadGuest;
```

```sql
ALTER TABLE Guest
ALTER COLUMN LeadGuest VARCHAR(3) NULL;
```

I also later changed the leadguest column (figure 1.1) to allow NULL values, and setting the default value as NO, so that data in this column only needs to be entered as 'YES' for lead guests which would make it simpler to insert data.

Figure 1.2

```sql
INSERT INTO Guest(Title,FirstName,LastName,DOB,StreetAddress,PostCode,City,Country,PhoneNo,Email, LeadGuest) VALUES
('Mr','Dahlia','Oliver','19960306','788-6280 Non, Rd.','BH37JH','Belfast','Ireland','(025) 2893 2924','cursus.Nunc@egetnisi.com','YES'),
('Mr','Cynthia','Dillon','19301220','3605 Proin Av.','07911','Wuppertal','Germany','(025) 7615 6479','non.enim@nequepellentesque.ca','NO'),
('Mr','Nash','Rowe','19520408','Ap #429-4129 Nisl Road','54265','Bourges','France','0500 920389','enim.Suspendisse.aliquet@dignissimpharetra.ca', 'YES'),
('Mrs','Cadman','Nash','19900410','331 Et Rd.','07987','Galway','Ireland','055 9005 1783','dolor@elit.edu', 'YES'),
('Mrs','Quemby','Fitzpatrick','19690308','P.O. Box 570, 4982 Ac Rd.','97601','Bègles','France','0500 639044','mi.felis@disparturientmontes.org', 'YES'),
('Mrs','Rajah','Mccarthy','19650630','P.O. Box 462, 309 Nec Ave','BH4TH4','Galway','Ireland','0500 248759','Nullam@risus.edu','NO'),
('Dr','Darrel','Farrell','19700512','P.O. Box 125, 6765 Adipiscing Rd.','78937','Galway','Ireland','0500 019181','Proin.eget.odio@tortor.ca','NO'),
('Ms','Quintessa','Lewis','19500929','9112 Penatibus Rd.','25149','Reutlingen','Germany','(0117) 237 3449','ornare.egestas.ligula@lectus.org','NO'),
('Miss','September','Washington','20100715','434 Mauris St.','40658','Pontarlier','France','(0111) 804 0650','dui.Suspendisse@sagittisfelisDonec.net','YES'),
('Ms','Gail','Henson','19700509','334-2361 Pede. Road','51795','Schifferstadt','Germany','055 8327 4338','quis.pede.Praesent@condimentumDonecat.com','NO'),
('Mrs','Jesse','Weaver','19821225','726-1189 In Road','66886','Cholet','France','0800 1111','nunc.interdum.feugiat@lacuspede.org','YES'),
('Mr','Chiquita','Donovan','20000225','P.O. Box 224, 2855 Ridiculus Road','HT56FH','Galway','Ireland','056 5373 9144','turpis.nec.mauris@nascetur.com','NO'),
('Mrs','Malcolm','Jordan','19950619','P.O. Box 646, 9838 Mus. Ave','89846','Belfast','Ireland','(0116) 236 3624','neque@pretiumaliquetmetus.net','YES'),
('Ms','Joelle','Barron','19401113','16 Franklin Street','TH56GH','London','United Kingdom','0800 522 7340','Cum.sociis.natoque@a.co.uk','YES'),
('Mr','Cyrus','Tillman','20010824','16 Franklin Street','TH56GH','London','United Kingdom','0800 522 7340','in.cursus@dictumeu.net','NO'),
('Mr','Harper','Valentine','20031005','16 Franklin Street','TH56GH','London','United Kingdom','0800 522 7340','Ut.tincidunt.vehicula@consequat.net','NO'),
('Mr','Willa','Walker','19800416','Ap #220-2078 Felis Rd.','37229','Orvault','France','0818 884 0095','nulla.Donec@cursus.net','YES'),
('Mr','Octavia','Navarro','19500829','716-9307 Eget, Street','F15 2KC','Stroud','United Kingdom','0920 810 5551','euismod.et.commodo@vehiculaaliquetlibero.edu','YES'),
('Miss','Imani','Martin','20061012','716-9307 Eget, Street','F15 2KC','Stroud','United Kingdom','0920 810 5551','in.magna@utquam.net','NO'),
('Mr','Hyatt','Gray','20080615','716-9307 Eget, Street','F15 2KC','Stroud','United Kingdom','0920 810 5551','Lorem@ipsumsodales.com','NO');

SELECT * FROM Guest;
```

	GuestID	Title	FirstName	LastName	DOB	StreetAddress	PostCode	City	Country	PhoneNo	Email	LeadGuest
1	110	Mr	Dahlia	Oliver	1996-03-06	788-6280 Non, Rd.	BH37JH	Belfast	Ireland	(025) 2893 2924	cursus.Nunc@egetnisi.com	YES
2	111	Mr	Cynthia	Dillon	1930-12-20	3605 Proin Av.	07911	Wuppertal	Germany	(025) 7615 6479	non.enim@nequepellentesque.ca	NO
3	112	Mr	Nash	Rowe	1952-04-08	Ap #429-4129 Nisl Road	54265	Bourges	France	0500 920389	enim.Suspendisse.aliquet@dignissimpharetra.ca	YES
4	113	Mrs	Cadman	Nash	1990-04-10	331 Et Rd.	07987	Galway	Ireland	055 9005 1783	dolor@elit.edu	YES
5	114	Mrs	Quemby	Fitzpatrick	1969-03-08	P.O. Box 570, 4982 Ac Rd.	97601	Bègles	France	0500 639044	mi.felis@disparturientmontes.org	YES
6	115	Mrs	Rajah	Mccarthy	1965-06-30	P.O. Box 462, 309 Nec Ave	BH4TH4	Galway	Ireland	0500 248759	Nullam@risus.edu	NO
7	116	Dr	Darrel	Farrell	1970-05-12	P.O. Box 125, 6765 Adipis...	78937	Galway	Ireland	0500 019181	Proin.eget.odio@tortor.ca	NO
8	117	Ms	Quintessa	Lewis	1950-09-29	9112 Penatibus Rd.	25149	Reutling...	Germany	(0117) 237 3449	ornare.egestas.ligula@lectus.org	NO
9	118	Miss	Septem...	Washin...	2010-07-15	434 Mauris St.	40658	Pontarlier	France	(0111) 804 0650	dui.Suspendisse@sagittisfelisDonec.net	YES
10	119	Ms	Gail	Henson	1970-05-09	334-2361 Pede. Road	51795	Schiffers...	Germany	055 8327 4338	quis.pede.Praesent@condimentumDonecat.c...	NO
11	120	Mrs	Jesse	Weaver	1982-12-25	726-1189 In Road	66886	Cholet	France	0800 1111	nunc.interdum.feugiat@lacuspede.org	YES
12	121	Mr	Chiquita	Donovan	2000-02-25	P.O. Box 224, 2855 Ridic...	HT56FH	Galway	Ireland	056 5373 9144	turpis.nec.mauris@nascetur.com	NO
13	122	Mrs	Malcolm	Jordan	1995-06-19	P.O. Box 646, 9838 Mus. ...	89846	Belfast	Ireland	(0116) 236 3624	neque@pretiumaliquetmetus.net	YES
14	123	Ms	Joelle	Barron	1940-11-13	16 Franklin Street	TH56GH	London	United ...	0800 522 7340	Cum.sociis.natoque@a.co.uk	YES
15	124	Mr	Cyrus	Tillman	2001-08-24	16 Franklin Street	TH56GH	London	United ...	0800 522 7340	in.cursus@dictumeu.net	NO
16	125	Mr	Harper	Valentine	2003-10-05	16 Franklin Street	TH56GH	London	United ...	0800 522 7340	Ut.tincidunt.vehicula@consequat.net	NO
17	126	Mr	Willa	Walker	1980-04-16	Ap #220-2078 Felis Rd.	37229	Orvault	France	0818 884 0095	nulla.Donec@cursus.net	YES
18	127	Mr	Octavia	Navarro	1950-08-29	716-9307 Eget, Street	F15 2KC	Stroud	United ...	0920 810 5551	euismod.et.commodo@vehiculaaliquetlibero.e...	YES
19	128	Miss	Imani	Martin	2006-10-12	716-9307 Eget, Street	F15 2KC	Stroud	United ...	0920 810 5551	in.magna@utquam.net	NO
20	129	Mr	Hyatt	Gray	2008-06-15	716-9307 Eget, Street	F15 2KC	Stroud	United ...	0920 810 5551	Lorem@ipsumsodales.com	NO

Figure 1.3

```sql
INSERT INTO Guest(Title,FirstName,LastName,DOB,StreetAddress,PostCode,City,Country,PhoneNo,Email, LeadGuest)
VALUES ('mr','Joe','David','19100306','788-6280 Non, Rd.','BH37JH','Belfast','Ireland','(025) 2893 2924','cursus.Nuncegetnisi.com','Y');
```

Messages

```
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chkBrithDate". The conflict occurred in database "Happy Vally Campsite", table "db
The statement has been terminated.
```

```sql
INSERT INTO Guest(Title,FirstName,LastName,DOB,StreetAddress,PostCode,City,Country,PhoneNo,Email, LeadGuest)
VALUES ('mrr','Joe','David','19100306','788-6280 Non, Rd.','BH37JH','Belfast','Ireland','(025) 2893 2924','cursus.Nuncegetnisi.com','Y');
```

Messages

```
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chkTitle". The conflict occurred in database "Happy Vally Campsite", table "dbo.Gu
The statement has been terminated.
```

```sql
INSERT INTO Guest(Title,FirstName,LastName,DOB,StreetAddress,PostCode,City,Country,PhoneNo,Email, LeadGuest)
VALUES ('mr','Joe','David','19970306','788-6280 Non, Rd.','BH37JH','Belfast','Ireland','(025) 2893 2924','cursus.Nuncegetnisi.com','Y');
```

Messages

```
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chkEmail". The conflict occurred in database "Happy Vally Campsite", table "dbo.Gu
The statement has been terminated.
```

Figure 1.2 is the code to insert the values into the Guest table, with the results of the SQL statement shown underneath. To test that all the check constraints are working correctly, I entered incorrect data into the columns to see the results (see figure 1.3).

Party

Figure 2

```
PartyID         INT PRIMARY KEY IDENTITY (1,1),
PartySize       TINYINT         NOT NULL,
Adults          TINYINT         NOT NULL,
Children        TINYINT         NOT NULL,
Pets            TINYINT         NOT NULL,
LeadGuestID     INT REFERENCES  Guest(GuestID)
                    ON DELETE NO ACTION
);
```

Messages

```
Command(s) completed successfully.
```

Column Name	Data Type
PartyID	int
PartySize	tinyint
Adults	tinyint
Children	tinyint
Pets	tinyint
LeadGuestID	int

Figure 2 shows the party table. LeadGuest is a foreign key relating to the Guest table, with an on delete rule set as no action so that if the partyID is references in another table, it cannot be deleted as it could ruin data integrity in the database.

Figure 2.1

	GuestID	StreetAddress	DOB	LeadGuest
5	114	P.O. Box 570, 4982 Ac Rd.	1969-03-08	YES
6	115	P.O. Box 462, 309 Nec Ave	1965-06-30	NO
7	116	P.O. Box 125, 6765 Adipiscing Rd.	1970-05-12	NO
8	117	9112 Penatibus Rd.	1950-09-29	NO
9	118	434 Mauris St.	2010-07-15	YES
10	119	334-2361 Pede. Road	1970-05-09	NO
11	120	726-1189 In Road	1982-12-25	YES
12	121	P.O. Box 224, 2855 Ridiculus Road	2000-02-25	NO
13	122	P.O. Box 646, 9838 Mus. Ave	1995-06-19	YES
14	123	16 Franklin Street	1940-11-13	YES
15	124	16 Franklin Street	2001-08-24	NO
16	125	16 Franklin Street	2003-10-05	NO
17	126	Ap #220-2078 Felis Rd.	1980-04-16	YES
18	127	716-9307 Eget, Street	1950-08-29	YES
19	128	716-9307 Eget, Street	2006-10-12	NO
20	129	716-9307 Eget, Street	2008-06-15	NO

```
INSERT INTO Party(PartySize,Adults,Children,Pets,LeadGuestID) VALUES
(2, 2, 0, 0, 110),  (1, 1, 0, 0, 112),  (1, 1, 0, 2, 113),  (4, 4, 0, 0, 114),
(2, 1, 1, 0, 118),  (2, 1, 1, 1, 120),  (1, 0, 0, 0, 122),  (3, 1, 2, 0, 123),
(1, 1, 0, 2, 126),  (3, 1, 2, 0, 127);

SELECT * FROM Party;
```

100 %

Results | Messages

	PartyID	PartySize	Adults	Children	Pets	LeadGuestID
1	1	2	2	0	0	110
2	2	1	1	0	0	112
3	3	1	1	0	2	113
4	4	4	4	0	0	114
5	5	2	1	1	0	118
6	6	2	1	1	1	120
7	7	1	0	0	0	122
8	8	3	1	2	0	123
9	9	1	1	0	2	126
10	10	3	1	2	0	127

To insert data in the party column, for simplicity I used a select statement to return all information to the guest table, and entered the leadguest ID by looking at the table (see figure 2.1)

PartyMembers

Figure 3

```sql
CREATE TABLE PartyMembers (
GuestID         INT  NOT NULL,
PartyID         INT  NOT NULL

CONSTRAINT PK_PartyMem        PRIMARY KEY (GuestID, PartyID),
CONSTRAINT FK_PartyMemGuest   FOREIGN KEY (GuestID)
                              REFERENCES  Guest(GuestID),
CONSTRAINT FK_PartyMemParty   FOREIGN KEY (PartyID)
                              REFERENCES  Party(PartyID)
);
```

Messages

```
Command(s) completed successfully.
```

Column Name	Data Type
GuestID	int
PartyID	int

This is the PartyMembers table (Figure 3). Originally I named this party details in part 1, however the way I designed the table it would only allow for one guest for each party. To overcome this I made changes to the guest table as seen previously, and completely re-designed this table. GuestID and PartyID and both foreign keys and primary keys. It shows which guest is associated with which party.

I did not set any constraints for the on update or on delete, as sql server sets them as NO ACTION automatically which is what I would have set them as.

Figure 3.1

```sql
INSERT INTO PartyMembers VALUES
(110, 1),(111,1), (112,2), (113,3), (114,4), (115,4),(116,4),(117,4),(118,5),
(119,5),(120,6),(121,6),(122,7),(123,8),(124,8),(125,8),(126,9),(127,10),(128,10),(129,10);

SELECT * FROM PartyMembers;
```

	GuestID	PartyID
1	110	1
2	111	1
3	112	2
4	113	3
5	114	4
6	115	4
7	116	4
8	117	4
9	118	5
10	119	5
11	120	6
12	121	6
13	122	7
14	123	8
15	124	8
16	125	8
17	126	9
18	127	10
19	128	10
20	129	10

To insert data into the party members table I again just looked at the guest table for simplicity (Figure 3.1).

Vehicle

Figure 4

```sql
CREATE TABLE Vehicle (
NumberPlate      VARCHAR(20) PRIMARY KEY,
VehicleType      VARCHAR(20)  NOT NULL,
Colour           VARCHAR(20)  NOT NULL,
PartyID          INT REFERENCES Party(PartyID)
);
```

Messages

Command(s) completed successfully.

NumberPlate	varchar(20)
VehicleType	varchar(20)
Colour	varchar(20)
PartyID	int

Figure 4.1

```sql
INSERT INTO Vehicle( NumberPlate,VehicleType,Colour,PartyID) VALUES
('OOH 848Y', 'Car', 'Black', 1),
('BP99 NPC', 'Motorbike', 'White', 2),
('LB56 VWD', 'Motorbike', 'Grey', 3),
('NA57 ZGG', 'Car', 'Blue', 4),
('KH55 HLJ', 'Car', 'Silver', 5),
('ZT66 YHT', 'Car', 'White', 6),
('FG53 YTT', 'Car', 'Orange', 7),
('JK43 HDG', 'Car', 'Green', 8),
('HH00 5HR', 'Car', 'Yellow', 9),
('FS33 GGH', 'Car', 'Black', 10);

SELECT * FROM Vehicle;
```

	NumberPlate	VehicleType	Colour	PartyID
1	BP99 NPC	Motorbike	White	2
2	FG53 YTT	Car	Orange	7
3	FS33 GGH	Car	Black	10
4	HH00 5HR	Car	Yellow	9
5	JK43 HDG	Car	Green	8
6	KH55 HLJ	Car	Silver	5
7	LB56 VWD	Motorbike	Grey	3
8	NA57 ZGG	Car	Blue	4
9	OOH 848Y	Car	Black	1
10	ZT66 YHT	Car	White	6

The vehicle table (figure 4) has a foreign key that references the party table. I could have added a check constraint that would check to see if the format of the data entered in the number plate column is that of a number plate, however I was unsure of how to do this. Figure 4.1 shows the insert statements and the results from the inputted values. To insert the data, I just picked each partyID from the party table.

Accommodation

Figure 5

```sql
CREATE TABLE Accomodation (
AccomodationID      INT PRIMARY KEY IDENTITY (1,1),
AccType             VARCHAR(30) NOT NULL,
                    CONSTRAINT  chkType
                    CHECK (AccType IN ('Tent', 'Motorhome','Caravan','Glamping yurt')),
Capacity            TINYINT     NOT NULL,
PricePerNight       SMALLMONEY  NOT NULL,
);
```

Messages

Command(s) completed successfully.

	Column Name	Data Type
	AccomodationID	int
	AccType	varchar(30)
	Capacity	tinyint
	PricePerNight	smallmoney

The accommodation table (figure 5) only has one check constraint, named chkType. What this constraint does is checks to see that the data inserted is either 'Tent', 'Motorhome', 'Caravan' or 'Glamping yurt'. This is to improve data integrity so there are not any random values inserted into accommodation type, as the campsite only provides these four types of accommodation.

Figure 5.1

```sql
INSERT INTO Accomodation (AccType,Capacity,PricePerNight) VALUES
('Caravan', 8, 50.00), ('Tent', 4, 15.00), ('Motorhome', 4, 30.00),
('Glamping Yurt', 5 , 80.00), ('Tent', 6, 22.00), ('Tent', 2, 10.00),
('Caravan', 6, 35.00), ('Motorhome', 7, 35.00), ('Glamping Yurt', 5, 80.00),
('Glamping Yurt', 6, 85.00), ('Tent', 5, 18.00), ('Caravan', 7, 43.00);

SELECT * FROM Accomodation;
);
```

```sql
INSERT INTO Accomodation (AccType,Capacity,PricePerNight) VALUES
('House', 8, 50.00);
```

100 %

Messages

```
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chkType". The conflict occurred in database "Happy Vally Campsite", table "dbo.Acc
The statement has been terminated.
```

To insert into the accommodation table (figure 5.1) the ID column is not required as it is set as Identity, therefore inserts a unique number automatically. Underneath the insert statement is the code to try and insert an incorrect value for the accType column. The screenshot show that the constraint on this column works correctly.

	AccomodationID	AccType	Capacity	PricePerNight
1	25	Caravan	8	50.00
2	26	Tent	4	15.00
3	27	Motorhome	4	30.00
4	28	Glamping Yurt	5	80.00
5	29	Tent	6	22.00
6	30	Tent	2	10.00
7	31	Caravan	6	35.00
8	32	Motorhome	7	35.00
9	33	Glamping Yurt	5	80.00
10	34	Glamping Yurt	6	85.00
11	35	Tent	5	18.00
12	36	Caravan	7	43.00

Booking

Figure 6

```sql
CREATE TABLE Booking (
BookingID           INT PRIMARY KEY IDENTITY(1,1),
PartyID             INT REFERENCES Party(PartyID),
AccommodationID     INT REFERENCES Accomodation(AccomodationID),
CheckIn             DATE     NOT NULL
                    CONSTRAINT  chkDate
                    CHECK       (CheckIn >= GETDATE()),
CheckOut            DATE     NOT NULL,
);
```

Messages

Command(s) completed successfully.

Column Name	Data Type
BookingID	int
PartyID	int
AccommodationID	int
CheckIn	date
CheckOut	date

```sql
ALTER TABLE Booking ADD
  CONSTRAINT chkDate CHECK ( CheckIn !< (GETDATE() - 1));
  CONSTRAINT chkCheckOutDate CHECK (CheckOut > CheckIn);
```

```sql
ALTER TABLE Booking
  ADD NumOfNights AS (DATEDIFF(day,CheckIn,CheckOut));
```

The Booking table (figure 6) has one check constraint at the table level. When creating the table I realised the constraint chkDate was not allowing me to insert a check in date that was the current date, therefore I had to use the ALTER TABLE statement to change this constraint. It checks that the check in date is not less than the current date – 1, because if I had just used GETDATE(), only values greater than the current date could be entered. It also checks that the checkout date is in the future from the check in date, as if you could set a date in the past this would not be realistic data.

I later added another column named NumOfNights to the table to make some of the sql statements in the invoice table easier to execute. This column is a computed column that works out how many nights a guest is staying. I used the DATEDIFF function which takes two parameters of DATE and works out the difference.

Figure 6.1 shows the insert statement and the table after the data had been entered.

Figure 6.1

```sql
INSERT INTO Booking(PartyID,AccommodationID,CheckIn,CheckOut) VALUES
(1, 30, '20170427','20170511'), (2, 26, '20170427','20170511'),
(3, 30, '20170512','20170520'), (4, 27, '20170427','20170505'),
(5, 28, '20170427','20170511'), (6, 34, '20170427','20170511'),
(7, 32, '20170414','20170502'), (8, 32, '20170507','20170520'),
(9, 31, '20170418','20170428'), (10, 27, '20170505','20170513');

SELECT * FROM Booking;
```

100 %

Results | Messages

	BookingID	PartyID	AccommodationID	CheckIn	CheckOut	NumOfNights
1	1	1	30	2017-04-27	2017-05-11	14
2	2	2	26	2017-04-27	2017-05-11	14
3	3	3	30	2017-05-12	2017-05-20	8
4	4	4	27	2017-04-27	2017-05-05	8
5	5	5	28	2017-04-27	2017-05-11	14
6	6	6	34	2017-04-27	2017-05-11	14
7	7	7	32	2017-04-14	2017-05-02	18
8	8	8	32	2017-05-07	2017-05-20	13
9	9	9	31	2017-04-18	2017-04-28	10
10	10	10	27	2017-05-05	2017-05-13	8

To ensure that accommodation cannot get double booked, I added a trigger onto the booking table (figure 6.2). This trigger is activated after data is inserted or updated and checks to see if there is two of the same accommodation IDs that have an overlapping date in either the check in or checkout date. If a row exists it means that accommodation has been double booked, then the database is rolled back to before the data was entered, so the booking does not go through and an error appears.

Figure 6.2

```
CK CONTRSINT...oe Shepherd (54))*    SQLQuery1.sql - not connected*
1  CREATE TRIGGER Booked ON dbo.Booking
2  after INSERT, UPDATE
3  AS
4
5
6  IF EXISTS
7  (
8
9      SELECT 1 FROM inserted as x
10  INNER JOIN Booking ON x.AccommodationID = Booking.AccommodationID AND x.BookingID != Booking.BookingID
11  WHERE x.CheckOut >= Booking.CheckIn AND Booking.CheckOut >= x.CheckIn
12
13  )
14
15  BEGIN
16  ROLLBACK TRANSACTION;
17  RETURN
18  END
```

Figure 6.3

```
SQLQuery8.sql - JO...\Joe Shepherd (56))*    accommodation.sql...Joe Shepherd (55))*    dataApr-8-2017 (1)....\Joe Shepherd (54))*
INSERT INTO Booking(PartyID,AccommodationID,CheckIn,CheckOut) VALUES
(1, 30, '20170327','20170511');

100 %
Messages
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chkDate". The conflict occurred in database "Happy Vally Campsite", table "dbo.Boo
The statement has been terminated.

2
3  INSERT INTO Booking VALUES
4  (6, 28 , '20170510', '20170410');

39 %
Messages
Msg 547, Level 16, State 0, Line 3
The INSERT statement conflicted with the CHECK constraint "chkCheckOutDate". The conflict occurred in database "Happy Vally Campsite", table "dbo.Booking".
The statement has been terminated.
```

Figure 6.4

	BookingID	PartyID	AccommodationID	CheckIn	CheckOut	NumOfNights
1	4	4	27	2017-04-27	2017-05-05	8

```
2
3  INSERT INTO Booking VALUES
4  (6, 27 , '20170426', '20170510');

%
Messages
Msg 3609, Level 16, State 1, Line 3
The transaction ended in the trigger. The batch has been aborted.
```

To ensure the check constraints on the booking table did the correct thing, I inserted some test data (figure 6.3). The first insert statement tests that a date in the past cannot be set for the checkin date. The second insert statement checks that a checkout date set before the checkin date cannot be inserted in the table. Both came back with errors which shows both constraints are working correctly.

To test that the trigger does not allow for double bookings, I selected a row from the booking table (figure 6.4), and attempted to enter a new booking with the same accommodation ID and a date that overlaps with the other booking date. As you can see in the screenshot the error appears as 'transaction ended in trigger' therefore accommodation cannot be overbooked.

Invoice

Figure 7

```
CREATE TABLE Invoice (
AccInvoiceID        INT PRIMARY KEY IDENTITY(1,1),
BookingID           INT REFERENCES Booking(BookingID),
TotalPrice          MONEY  NOT NULL,
Deposit             MONEY  NOT NULL,
TotalPaid           MONEY,
DatePaid            DATETIME

);
```

Messages

Command(s) completed successfully.

Column Name	Data Type
AccInvoiceID	int
BookingID	int
TotalPrice	money
Deposit	money
TotalPaid	money
DatePaid	datetime

Figure 7.1

```
DECLARE @i INT
SET @i = 1;
WHILE @i <=25
BEGIN
INSERT INTO Invoice (BookingID,TotalPrice,Deposit) VALUES
((SELECT BookingID FROM Booking WHERE BookingID IN( @i)) ,
(SELECT SUM(NumOfNights * PricePerNight) FROM Booking, Accomodation WHERE Booking.BookingID IN(@i) AND Accomodation.AccomodationID = Booking.AccommodationID)
, 0)
SET @i = @i +1
END

UPDATE Invoice
SET Deposit = TotalPrice * 0.25
WHERE BookingID IN (SELECT Booking.BookingID FROM Booking WHERE Booking.CheckIn > GETDATE());

 SELECT * FROM Invoice;
```

	AccInvoiceID	BookingID	TotalPrice	Deposit	TotalPaid
1	61	1	140.00	35.00	35.00
2	62	2	210.00	52.50	52.50
3	63	3	80.00	20.00	20.00
4	64	4	240.00	60.00	60.00
5	65	5	1120.00	280.00	280.00
6	66	6	1190.00	297.50	297.50
7	67	7	630.00	157.50	630.00
8	68	8	455.00	113.75	113.75
9	69	9	350.00	87.50	350.00
10	85	25	150.00	0.00	150.00

Figure 7 shows the sql statement to create the invoice table. I did not set any constraints for this table. To insert the data into this table (figure 7.1) I used the WHILE statement to automatically enter all the bookings into the invoice table. By declaring a variable INT that iterates through 1 to 25 as 25 was the

max bookingID at the time, then the bookingID value inserted was that of the variable, and I used a subquery with the SUM function to work out how much the total cost of each booking is from the NumofNights column in the booking table and PricePernight column from the accommodation table. After that I used the update statement to work out the deposit by multiplying the total cost by 0.25 (25%) only if the check in date was not the current date because guests who are making a booking the same day as the check in they must pay all in advance. The total paid was set as the deposit amount as the deposit must be payed straight away, unless the deposit was 0 which the total paid was the full amount. I used this method as it was a lot easier than inserting all the data manually and so that the data entered was correct.

Booking Status

Figure 8

```
CREATE TABLE BookingStatus  (
BookingID       INT REFERENCES Booking(BookingID),
BookStatus      VARCHAR(50)     NOT NULL

CONSTRAINT PK_BOOKINGSTATUS PRIMARY KEY (BookingID)
);
```

Messages

```
Command(s) completed successfully.
```

Column Name	Data Type
BookingID	int
BookStatus	varchar(50)

Figure 8.1

```
INSERT INTO BookingStatus VALUES
(1, 'Paid in full'),
(2, 'Current Guests'),
(3, 'Reserved'),
(4, 'Reserved'),
(5, 'Deposit Due'),
(6, 'Deposit Paid'),
(7, 'Payment Due'),
(8, 'Paid in Full'),
(9, 'Current Guests'),
(25, 'Reserverd');

SELECT * FROM BookingStatus
```

Results | Messages

	BookingID	BookStatus
1	1	Paid in full
2	2	Current Guests
3	3	Reserved
4	4	Reserved
5	5	Deposit Due
6	6	Deposit Paid
7	7	Payment Due
8	8	Paid in Full
9	9	Current Guests
10	25	Reserverd

Figure 8 shows the create table statement for the booking status table. 8.1 shows the insert statement and the table after data has been inserted. This table has no check constraints.

Activity

Figure 9

```sql
CREATE TABLE Activity (
ActivityID          INT PRIMARY KEY IDENTITY(1,1),
Description         VARCHAR(50),
SuitableAgeMin      INT
                    CONSTRAINT chkMin
                    CHECK (SuitableAgeMin !< 0),
SuitableAgeMax      INT ,

LocationOfActivity  VARCHAR(50)     NOT NULL,
Date                DATE            NOT NULL,
                    CONSTRAINT  chkActivityDate
                    CHECK (Date !< GETDATE()),

StartTime           TIME            NOT NULL,
EndTime             TIME            NOT NULL,
                    CONSTRAINT chkTime
                    CHECK (EndTime > StartTime),

TotalPeople         SMALLINT        NOT NULL,
PricePerPerson      SMALLMONEY      NOT NULL,
OrganiserID         INT NOT NULL REFERENCES ActivityOrganiser(OrganiserID)
);
```

Messages

Command(s) completed successfully.

JOESHEPS.Happy Vall...site - dbo.Activity

Column Name	Data Type
ActivityID	int
Description	varchar(50)
SuitableAgeMin	int
SuitableAgeMax	int
LocationOfActivity	varchar(50)
Date	date
StartTime	time(7)
EndTime	time(7)
TotalPeople	smallint
PricePerPerson	smallmoney
OrganiserID	int

This is the activity table (figure 9). I added three check constraints. chkMin ensures that the suitable minimum age is not less than 0. chkActivityDate ensures that the date of the activity is not set in the past, and chkTime ensures that the end time is not before the starting time. When inserting data into the table I left some max age values as NULL. I thought there was not much need of setting a max age if an activity was to allow all ages above a certain age. However I couldn't get some queries to work so later updates the values (figure 9.1).

Figure 9.1

```sql
INSERT INTO Activity VALUES

('Horse Riding', 8, 16,  '18 Fredrick Manor', '20170419', '10:30:00', '14:00:00', 5, 20.00, 1),
('Horse Riding', 17, NULL, '18 Fredrick Manor', '20170419', '10:30:00', '14:00:00', 5, 25.00, 9),
('Arts and Crafts', 2, 8 , '21 Broadway Road', '20170423', '12:00:00', '13:30:00',  10, 10.00, 3),
('Go-Karting', 4, 15, '67 Manor House', '20170429','13:00:00', '15:00:00', 15, 20.00, 2),
('Rock Climbing', 10, NULL, 'On Site', '20170502','12:30:00', '13:30:00', 10, 15.00, 4),
('Go-Karting', 16, NULL, '71 Ramsgate Rd', '20170503', '11:00:00', '13:00:00', 10, 30.00, 2),
('Arts and Crafts', 2, 8, '21 Broadway Road', '20170507', '14:00:00', '15:00:00', 10, 10.00, 3),
('Swimming', 1, NULL, '52 Park Place', '20170507', '12:00:00', '15:00:00', 20, 05.00, 6),
('WaterSkiing', 16, NULL, '39 Scarcroft Road', '20170512', '12:00:00', '13:30:00', 6, 40.00, 8),
('Assualt Course', 12, NULL, '43 West Lane', '20170515', '12:00:00', '14:00:00', 15, 20.00, 10);

SELECT * FROM Activity;
```

	ActivityID	Description	SuitableAgeMin	SuitableAgeMax	LocationOfActivity	Date	StartTime	EndTime	TotalPeople	PricePerPerson	OrganiserID
1	1	Horse Riding	8	16	18 Fredrick Manor	2017-04-19	10:30:00.0000000	14:00:00.0000000	5	20.00	1
2	2	Horse Riding	17	100	18 Fredrick Manor	2017-04-19	10:30:00.0000000	14:00:00.0000000	5	25.00	9
3	3	Arts and Crafts	2	8	21 Broadway Road	2017-04-23	12:00:00.0000000	13:30:00.0000000	10	10.00	3
4	4	Go-Karting	4	15	67 Manor House	2017-04-29	13:00:00.0000000	15:00:00.0000000	15	20.00	2
5	5	Rock Climbing	10	16	On Site	2017-05-02	12:30:00.0000000	13:30:00.0000000	10	15.00	4
6	6	Go-Karting	16	100	71 Ramsgate Rd	2017-05-03	11:00:00.0000000	13:00:00.0000000	10	30.00	2
7	7	Arts and Crafts	2	8	21 BroadWay Road	2017-05-07	14:00:00.0000000	15:00:00.0000000	10	10.00	3
8	8	Swimming	1	100	52 Park Place	2017-05-07	12:00:00.0000000	15:00:00.0000000	20	5.00	6
9	9	WaterSkiing	16	60	39 Scarcroft Road	2017-05-12	12:00:00.0000000	13:30:00.0000000	6	40.00	8
10	10	Assualt Course	12	50	43 West Lane	2017-05-15	12:00:00.0000000	14:00:00.0000000	15	20.00	10

I inserted a range of test data to see if the check constraints do not allow data that is not supposed to be allowed (figure 9.2) As the screenshots show all three constraints are working correctly.

Figure 9.2

```
SQLQuery3.sql - JO...\Joe Shepherd (56))*   SQLQuery2.sql - JO...\Joe Shepherd (55))   SQLQuery1.sql - JO...\Joe Shepherd (54))
1 INSERT INTO Activity VALUES
2 ('Horse Riding', -1, 60, 'anywhere', '20170505', '10:30:00', '12:00:00', 5, 20.00, 1);
89 %
Messages
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chkMin". The conflict occurred in database "Happy Vally Campsite", table "dbo.Activity", column 'SuitableAgeMin'.
The statement has been terminated.
```

```
SQLQuery3.sql - JO...\Joe Shepherd (56))*   SQLQuery2.sql - JO...\Joe Shepherd (55))   SQLQuery1.sql - JO...\Joe Shepherd (54))
1 INSERT INTO Activity VALUES
2 ('Horse Riding', 10, 60, 'anywhere', '20170305', '10:30:00', '12:00:00', 5, 20.00, 1);
89 %
Messages
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chkActivityDate". The conflict occurred in database "Happy Vally Campsite", table "dbo.Activity", column 'Date'.
The statement has been terminated.
```

```
SQLQuery3.sql - JO...\Joe Shepherd (56))*   SQLQuery2.sql - JO...\Joe Shepherd (55))   SQLQuery1.sql - JO...\Joe Shepherd (54))
1 INSERT INTO Activity VALUES
2 ('Horse Riding', 10, 60, 'anywhere', '20170505', '10:30:00', '10:00:00', 5, 20.00, 1);
89 %
Messages
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chkTime". The conflict occurred in database "Happy Vally Campsite", table "dbo.Activity".
The statement has been terminated.
```

Activity Booking

Figure 10

```
activityInvoic.sql -...S\Joe Shepherd (57))   ACTIVITYBOOKING.s...oe Shepherd (54))*
CREATE TABLE ActivityBooking (
PartyID         INT NOT NULL  REFERENCES Party(PartyID),
ActivityID      INT NOT NULL  UNIQUE REFERENCES Activity(ActivityID),
NumOfPeople     SMALLINT NOT NULL

CONSTRAINT PK_ActivityBooking PRIMARY KEY (PartyID, ActivityID)
);
100 %
Messages
Command(s) completed successfully.
```

Column Name	Data Type
PartyID	int
ActivityID	int
NumOfPeople	smallint

This is the ActivityBooking table (figure 10), which has partyID and ActivityID as both primary and foreign keys that reference the party and activity table respectively. I add a NumOfPeople column to make it easier to work out how many people are participating in each activity. The activity ID is set to UNQIUE so that an activity cannot be booked more than once.

To insert into this table (figure 10.1) I simply put the partyID, ActivityID and a select statement that selects the partysize column from the party table as the number of people participating.

The only test data I inserted was to see if an activity can be booked more than once (figure 10.2). The screenshot shows that if someone is trying to book the same activity it cannot be done.

Figure 10.1

```sql
INSERT INTO ActivityBooking (PartyID, ActivityID, NumOfPeople) VALUES
(1, 4, (SELECT PartySize FROM Party WHERE PartyID = 1)),
(2, 5, (SELECT PartySize FROM Party WHERE PartyID = 2)),
(3, 9, (SELECT PartySize FROM Party WHERE PartyID = 3)),
(4, 6, (SELECT PartySize FROM Party WHERE PartyID = 4)),
(5, 7, (SELECT PartySize FROM Party WHERE PartyID = 5)),
(6, 8, (SELECT PartySize FROM Party WHERE PartyID = 6)),
(7, 1, (SELECT PartySize FROM Party WHERE PartyID = 7)),
(8, 10, (SELECT PartySize FROM Party WHERE PartyID = 8)),
(9, 3, (SELECT PartySize FROM Party WHERE PartyID = 9)),
(7, 2, (SELECT PartySize FROM Party WHERE PartyID = 1));

SELECT * FROM ActivityBooking;
```

110 %

Results | Messages

	PartyID	ActivityID	NumOfPeople
1	1	4	2
2	2	5	1
3	3	9	1
4	4	6	4
5	5	7	2
6	6	8	2
7	7	1	1
8	7	2	2
9	8	10	3
10	9	3	1

Figure 10.2

```sql
INSERT INTO ActivityBooking (PartyID, ActivityID, NumOfPeople) VALUES
(1, 4, (SELECT PartySize FROM Party WHERE PartyID = 1)),
(2, 4, (SELECT PartySize FROM Party WHERE PartyID = 2));

SELECT * FROM ActivityBooking;
```

.0 %

Results | Messages

```
Msg 2627, Level 14, State 1, Line 11
Violation of PRIMARY KEY constraint 'PK_ActivityBooking'. Cannot insert duplicate key in object 'dbo.ActivityBooking'. The duplicate key value is
The statement has been terminated.

(10 row(s) affected)
```

Activity Organiser

Figure 11

```sql
CREATE TABLE ActivityOrganiser (
OrganiserID         INT PRIMARY KEY IDENTITY(1,1),
CompanyName         VARCHAR(100)    NOT NULL,
CompanyAddress      VARCHAR(100)    NOT NULL,
Website             VARCHAR(150)    NOT NULL,
Email               VARCHAR(80)     NOT NULL,
                    CONSTRAINT checkEmail
                    CHECK (Email LIKE '%_@__%.__%'),
Phone               VARCHAR(20)     NOT NULL
);
```

0 %

Messages

```
Command(s) completed successfully.
```

Column Name	Data Type
OrganiserID	int
CompanyName	varchar(100)
CompanyAddress	varchar(100)
Website	varchar(150)
Email	varchar(80)
Phone	varchar(20)

```sql
ALTER TABLE ActivityOrganiser
ADD CONSTRAINT chkPhoneNumber CHECK (Phone NOT LIKE '%[a-z]%'
AND DATALENGTH(Phone) < 13 AND DATALENGTH(Phone) > 10 );
```

This is the Activity Organiser table (figure 11). I added a check constraint named checkEmail which does the same thing as the chkEmail constraint on the guest table. I later added another constraint named chkPhoneNumber, which checks that no alphabetic characters are entered into this field, and that is the length of a phone number.

Figure 11.1

```sql
INSERT INTO ActivityOrganiser VALUES
('Lacys HorseRiding School' , '401 Aliquam Road','LacyHorses.com','Lacy_Horse@Hotmail.co.uk','01273 767524'),
('JetKarts', '89 Lorem Street', 'JetKarts.co.uk', 'JetKarts@gmail.com', '06305 288443'),
('Arts & Crafts', '484 Neque St', 'Crafting.com', 'ArtsCrafts@gmail.com', '01273 056787'),
('BrocksClimbing', '89 Shepherds Close', 'BrocksClimbing.co.uk','Brock@hotmail.com', '05067 002321'),
('Warrens Lesuire Centre', '99 Dictum Street','Warrenslesuire.com','Warrenssupport@gmail.com','02094 747844'),
('Get Waved', '34 Enim Road','Getwaved.co.uk','Waveycontact@gmail.com','07162 944585'),
('Minisrty Assualt', '53 Massa. Rd. ', 'MinsitryAssualt.co.uk', 'ContactMinistry@hotmail.co.uk', '01273 456788'),
('MarinaSports', '101 Marina Drive','MarinaSpots.com','Marinacustomerservice@gmail.com','02094 777866'),
('J&G Stables', '55 Park Lane','jmstables.co.uk','jmstables@gmail.com','07152 777585'),
('Adventurists', '78 Stablefield Road ', 'Adventuristsuk.co.uk', 'customerderviceadventurists@hotmail.co.uk', '01273 468678');

SELECT * FROM ActivityOrganiser;
```

Results | Messages

	OrganiserID	CompanyName	CompanyAddress	Website	Email	Phone
1	1	Lacys HorseRiding School	401 Aliquam Road	LacyHorses.com	Lacy_Horse@Hotmail.co.uk	01273 767524
2	2	JetKarts	89 Lorem Street	JetKarts.co.uk	JetKarts@gmail.com	06305 288443
3	3	Arts & Crafts	484 Neque St	Crafting.com	ArtsCrafts@gmail.com	01273 056787
4	4	BrocksClimbing	89 Shepherds Close	BrocksClimbing.co.uk	Brock@hotmail.com	05067 002321
5	5	Warrens Lesuire Centre	99 Dictum Street	Warrenslesuire.com	Warrenssupport@gmail.com	02094 747844
6	6	Get Waved	34 Enim Road	Getwaved.co.uk	Waveycontact@gmail.com	07162 944585
7	7	Minisrty Assualt	53 Massa. Rd.	MinsitryAssualt.co.uk	ContactMinistry@hotmail.co.uk	01273 456788
8	8	MarinaSports	101 Marina Drive	MarinaSpots.com	Marinacustomerservice@gmail.com	02094 777866
9	9	J&G Stables	55 Park Lane	jmstables.co.uk	jmstables@gmail.com	07152 777585
10	10	Adventurists	78 Stablefield Road	Adventuristsuk.co.uk	customerderviceadventurists@hotmail.co.uk	01273 468678

Figure 11.2

```sql
INSERT INTO ActivityOrganiser VALUES
('Lacys HorseRiding School' , '401 Aliquam Road','LacyHorses.com','Lacy_HorseHotmail.co.uk','01273 767524');

SELECT * FROM ActivityOrganiser;
```

Results | Messages

```
Msg 547, Level 16, State 0, Line 2
The INSERT statement conflicted with the CHECK constraint "checkEmail". The conflict occurred in database "Happy Vally Campsite", table "dbo.ActivityOrga
The statement has been terminated.
```

```sql
INSERT INTO ActivityOrganiser VALUES
('Lacys HorseRiding School' , '401 Aliquam Road','LacyHorses.com','Lacy_Horse@Hotmail.co.uk','0127');

SELECT * FROM ActivityOrganiser;
```

Results | Messages

```
Msg 547, Level 16, State 0, Line 2
The INSERT statement conflicted with the CHECK constraint "chkPhoneNumber". The conflict occurred in database "Happy Vally Campsite", table "dbo.Activity
The statement has been terminated.
```

```sql
INSERT INTO ActivityOrganiser VALUES
('Lacys HorseRiding School' , '401 Aliquam Road','LacyHorses.com','Lacy_Horse@Hotmail.co.uk','01273 35445644424');

SELECT * FROM ActivityOrganiser;
```

Results | Messages

```
Msg 547, Level 16, State 0, Line 2
The INSERT statement conflicted with the CHECK constraint "chkPhoneNumber". The conflict occurred in database "Happy Vally Campsite", table "dbo.Activity
The statement has been terminated.
```

Figure 11.1 shows the insert statement to fill in the activity organiser table and the resulting table. Figure 11.2 shows the test data inserted to check the constraints on the table are working correctly. All the tests show an error with the constraint that is being enforced.

Activity Invoice

Figure 12

```sql
CREATE TABLE ActivityInvoice (
 PartyID            INT NOT NULL,
 ActivityID         INT NOT NULL,
 OrganiserID        INT NOT NULL,
 TotalCost          SMALLMONEY NOT NULL,
 RobinsonShare      SMALLMONEY NOT NULL,
 OrgainserShare     SMALLMONEY NOT NULL

 CONSTRAINT FK_ActivityOrganiser FOREIGN KEY (OrganiserID)
                                 REFERENCES ActivityOrganiser(OrganiserID),

 CONSTRAINT FK_ActivityBook FOREIGN KEY (PartyID, ActivityID)
                            REFERENCES ActivityBooking(PartyID,ActivityID),

 CONSTRAINT PK_ActivityInvoice PRIMARY KEY (OrganiserID, PartyID, ActivityID),

 CONSTRAINT checkNegative    CHECK
                             (TotalCost !< 0 AND RobinsonShare !<0 AND OrgainserShare !<0)
);
```

```
Command(s) completed successfully.
```

Column Name	Data Type
PartyID	int
ActivityID	int
OrganiserID	int
TotalCost	smallmoney
RobinsonShare	smallmoney
OrgainserShare	smallmoney

This is the Activity Invoice table (figure 12). There is once check constraint named chkNegative that checks to see if the data inserted into the totalCost, robinsonShare and OrganiserShare columns is not negative, as they cannot earn a minus amount of money from the activities.

Figure 12.1

```sql
INSERT INTO ActivityInvoice (PartyID,ActivityID,OrganiserID,TotalCost,RobinsonShare,OrgainserShare)

SELECT PartyID , ActivityBooking.ActivityID, Activity.OrganiserID ,  0, 0, 0
 FROM   ActivityBooking, Activity, ActivityOrganiser
 WHERE  Activity.ActivityID = ActivityBooking.ActivityID  AND Activity.OrganiserID = ActivityOrganiser.OrganiserID;


UPDATE ActivityInvoice
SET TotalCost =
 PricePerPerson * NumOfPeople
FROM Activity, ActivityBooking
WHERE ActivityInvoice.ActivityID = Activity.ActivityID And Activity.ActivityID =  ActivityBooking.ActivityID;

UPDATE ActivityInvoice
SET RobinsonShare =
TotalCost*0.1;

UPDATE ActivityInvoice
SET OrgainserShare =
TotalCost*0.9;

SELECT * FROM ActivityInvoice ORDER BY PartyID;
```

To insert data into this table, I used a select statement to get the IDs from the activity booking, activity and activity organiser table that match the values for each row in the activity booking table. I then used the update statement that works out the total cost for each activity by multiplying the PricePerPerson and NumOfPeople column from the activity and activity booking table, and two more update statements to work out the robinson share and organiser share. This made it much easier to

work out the values in an automatic way rather than doing each row manually. The results of the query are shown below.

Results | Messages

	PartyID	ActivityID	OrganiserID	TotalCost	RobinsonShare	OrgainserShare
1	7	1	1	20.00	2.00	18.00
2	1	4	2	40.00	4.00	36.00
3	4	6	2	120.00	12.00	108.00
4	5	7	3	20.00	2.00	18.00
5	9	3	3	10.00	1.00	9.00
6	2	5	4	15.00	1.50	13.50
7	6	8	6	10.00	1.00	9.00
8	3	9	8	40.00	4.00	36.00
9	7	2	9	50.00	5.00	45.00
10	8	10	10	60.00	6.00	54.00

Figure 12.2

```
UPDATE ActivityInvoice
SET TotalCost = -10;
```

3 %

Messages

```
Msg 547, Level 16, State 0, Line 7
The UPDATE statement conflicted with the CHECK constraint "checkNegative". The conflict occurred in database "Happy Vally Campsite", table "dbo.ActivityInvoice".
The statement has been terminated.
```

Figure 12.2 shows the test data I inserted to see if I could have entered negative values in one of the columns, which returned an error.

Banned Guest

Figure 13

```
CREATE TABLE BannedGuest (
GuestID             INT NOT NULL REFERENCES Guest(GuestID),
DateBanned          DATE NOT NULL,
Reasons             VARCHAR(100) NOT NULL,

CONSTRAINT chkDateBanned CHECK (DateBanned !< GETDATE() -1),
CONSTRAINT PK_Bannedguest PRIMARY KEY (GuestID, DateBanned)
);
```

Column Name	Data Type
GuestID	int
DateBanned	datetime
Reasons	varchar(100)

This is the final table banned guest. I added one check constraint to check if the person being banned cannot be banned from a date set in the past. Figure 13.1 shows the insert statement. I used a subquery to select the GuestID from the guest table with a where condition of firstname and lastname, as I figured if you are banning someone you would not know their guestID, only their name. The last screenshot shows the test data entered to show that the check constraint does not allow a guest to be banned from a date set in the past. An error was returned.

Figure 13.1

```sql
INSERT INTO BannedGuest
VALUES
((SELECT GuestID FROM Guest WHERE FirstName = 'Dahlia' AND LastName = 'Oliver'), '20170511', 'Fighting with other guests'),
((SELECT GuestID FROM Guest WHERE FirstName = 'Cynthia' AND LastName = 'Dillon'), '20170511', 'Vandalsim'),
((SELECT GuestID FROM Guest WHERE FirstName = 'Quemby' AND LastName = 'Fitzpatrick'), '20170505', 'Unpaid fee'),
((SELECT GuestID FROM Guest WHERE FirstName = 'Gail' AND LastName='Henson'), '20170511', 'Littering'),
((SELECT GuestID FROM Guest WHERE FirstName = 'Jesse'AND LastName='Weaver'), '20170511', 'Abuse towards staff'),
((SELECT GuestID FROM Guest WHERE FirstName = 'Joelle'AND LastName='Barron'), '20170520', 'Vandalism'),
((SELECT GuestID FROM Guest WHERE FirstName = 'Octavia'AND LastName='Navarro'), '20170515', 'Unpaid fee'),
((SELECT GuestID FROM Guest WHERE FirstName = 'Willa'AND LastName='Walker'), '20170428', 'Littering'),
((SELECT GuestID FROM Guest WHERE FirstName = 'Hyatt'AND LastName='Gray'), '20170415', 'Fighting with other guests'),
((SELECT GuestID FROM Guest WHERE FirstName = 'Imani' AND LastName= 'Martin'), '20170415', 'Abuse towards staff');

SELECT * FROM BannedGuest;
```

	GuestID	DateBanned	Reasons
1	110	2017-05-11	Fighting with other guests
2	111	2017-05-11	Vandalsim
3	114	2017-05-05	Unpaid fee
4	119	2017-05-11	Littering
5	120	2017-05-11	Abuse towards staff
6	123	2017-05-20	Vandalism
7	126	2017-04-28	Littering
8	127	2017-05-15	Unpaid fee
9	128	2017-04-15	Abuse towards staff
10	129	2017-04-15	Fighting with other guests

```sql
INSERT INTO BannedGuest
VALUES
((SELECT GuestID FROM Guest WHERE FirstName = 'Dahlia' AND LastName = 'Oliver'), '20170211', 'Fighting with other guests');
```

Messages

```
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the CHECK constraint "chkDateBanned". The conflict occurred in database "Happy Vally Campsite", table "dbo.BannedGuest", column 'DateBanne
The statement has been terminated.
```

Scenarios and Queries

1.

BOOKINGSTATUS.sql...oe Shepherd (53))*

```sql
SELECT * FROM Guest
WHERE Country = 'United Kingdom'
ORDER BY LastName;
```

110 %

Results | Messages

	GuestID	Title	FirstName	LastName	DOB	StreetAddress	PostCode	City	Country	PhoneNo	Email	LeadGuest
1	123	Ms	Joelle	Barron	1940-11-13	16 Franklin Street	TH56GH	London	United Kingdom	0800 522 7340	Cum.sociis.natoque@a.co.uk	YES
2	129	Mr	Hyatt	Gray	2008-06-15	716-9307 Eget, Street	F15 2KC	Stroud	United Kingdom	0920 810 5551	Lorem@ipsumsodales.com	NO
3	128	Miss	Imani	Martin	2006-10-12	716-9307 Eget, Street	F15 2KC	Stroud	United Kingdom	0920 810 5551	in.magna@utquam.net	NO
4	127	Mr	Octavia	Navarro	1950-08-29	716-9307 Eget, Street	F15 2KC	Stroud	United Kingdom	0920 810 5551	euismod.et.commodo@vehiculaaliquetlibero.edu	YES
5	124	Mr	Cyrus	Tillman	2001-08-24	16 Franklin Street	TH56GH	London	United Kingdom	0800 522 7340	in.cursus@dictumeu.net	NO
6	125	Mr	Harper	Valentine	2003-10-05	16 Franklin Street	TH56GH	London	United Kingdom	0800 522 7340	Ut.tincidunt.vehicula@consequat.net	NO

This is the first query from appendix B. It is a simple select statement that shows all the guests that live in the UK. To achieve this I used a WHERE clause.

2.

```sql
BEGIN TRANSACTION CreateBooking
BEGIN TRY

DECLARE @guestid INT;
DECLARE @partyid INT;

INSERT INTO Guest VALUES
('Mr','Luke','Ribero','19940816', '18 FreshWater Avenue', 'RM6 4DT','London','United Kingdom', '07447400448' , 'lukerib94@gmail.com', 'YES');
SELECT @guestId = SCOPE_IDENTITY();

INSERT INTO Party VALUES
(4, 2, 2, 0, @guestId);
SELECT @partyid = SCOPE_IDENTITY();

INSERT INTO PartyMembers VALUES (@guestId, @partyid);

INSERT INTO Guest (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Mrs','Bella','Ribero','19960505', '18 FreshWater Avenue', 'RM6 4DT','London', 'United Kingdom', '07676345123' , 'bellarib@gmail.com');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);

INSERT INTO Guest  (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Mr','Paul','Ribero','20070528', '18 FreshWater Avenue', 'RM6 4DT','London', 'United Kingdom', '07676345123' , 'paulisthebest@hotmail.com');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);

INSERT INTO Guest (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Miss','Tara','Ribero','2011220', '18 FreshWater Avenue', 'RM6 4DT','London', 'United Kingdom', '07676345123' , 'Tara123@gmail.com');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);

INSERT INTO Booking VALUES
(@partyid,
(SELECT TOP 1 AccomodationID FROM Accomodation WHERE AccType = 'Motorhome' AND Capacity = 4),
'20170418', '20170423');

COMMIT TRANSACTION CreateBooking;
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION CreateBooking;
END CATCH;
```

For this query, a guest had to be added to the system and create a booking, recording the details of the party members. To achieve this I decided to use a transaction because if any of the entries had for some reason failed, none of the tables would be updated, which protects the data integrity otherwise there could have been incorrect data for the party members or booking table. By declaring two variables @GuestID and @PartyID, it made it very easy to insert the correct guestID

and PartyID into the party members table. After each guest had been added to the guest table, I used the SCOPE_IDENTITY function to select the last entered guestID and set @GuestID as this value. The party was created after the lead guest was added, and again using SCOPE_IDENTITY to set @PartyID as the last entered PartyID in the table. Then to add a guest to the party members table I only had to use the variables, instead of searching for their ID from the table and adding it afterwards. To create the booking, I again used the @PartyID variable and a select subquery to select the accommodation that the guests required with the capacity of their party size. The results of this query are show below highlighted in blue to show the guests and booking that was created.

GuestID	Title	FirstName	LastName	DOB	StreetAddress	PostCode	City	Country	PhoneNo	Email	LeadGuest
137	Mr	Luke	Ribero	1994-08-16	18 FreshWater Avenue	RM6 4DT	London	United ...	07447400448	lukerib94@gmail.com	YES
138	Mrs	Bella	Ribero	1996-05-05	18 FreshWater Avenue	RM6 4DT	London	United ...	07676345123	bellarib@gmail.com	NO
139	Mr	Paul	Ribero	2007-05-28	18 FreshWater Avenue	RM6 4DT	London	United ...	07676345123	paulisthebest@hotmail.com	NO
140	Miss	Tara	Ribero	2010-12-20	18 FreshWater Avenue	RM6 4DT	London	United ...	07676345123	Tara123@gmail.com	NO

PartyID	PartySize	Adults	Children	Pets	LeadGuestID
10	3	1	2	0	127
12	4	2	2	0	137

	GuestID	PartyID
21	137	12
22	138	12
23	139	12
24	140	12

BookingID	PartyID	AccommodationID	CheckIn	CheckOut	NumOfNights
25	10	27	2017-04-10	2017-04-15	5
28	12	27	2017-04-18	2017-04-23	5

3.

Query6.sql - JO...\Joe Shepherd (56))* | 3.sql - JOESHEPS.H...\Joe Shepherd (54))*

```sql
BEGIN TRANSACTION CreateBooking2
BEGIN TRY

DECLARE @guestid INT;
DECLARE @partyid INT;

INSERT INTO Guest VALUES
('Mr','David','Jones','19820306', '39 Brookside', 'EN4 8TT','London','United Kingdom', '0782678392' , 'DJones@trentschool.org', 'YES');
SELECT @guestId = SCOPE_IDENTITY();

INSERT INTO Party VALUES
(10, 3, 7, 0, @guestId);
SELECT @partyid = SCOPE_IDENTITY();

INSERT INTO PartyMembers VALUES (@guestId, @partyid);

INSERT INTO Guest (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Miss','Amelia','Scott','19700509', '1 littlegrove', 'EN4 89T','London', 'United Kingdom', '0776354789' , 'Ascott@trentschool.org');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);

INSERT INTO Guest  (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Mr','Fred','Henderson','19730816', '18 Churchhill Road', 'EN5 7NR','London', 'United Kingdom', '07662333456' , 'Fhenderson@trentschool.org');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);
```

Query6.sql - JO...\Joe Shepherd (56))* | 3.sql - JOESHEPS.H...\Joe Shepherd (54))*

```sql
INSERT INTO Guest (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Miss','Kim','Kardasian','20011012', '52 Bournechurch Road', 'EN4 5YT','London', 'United Kingdom', '02084470036' , 'Kimmygirl@gmail.com');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);

INSERT INTO Guest (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Miss','Michelle','Simpson','20010318', '57 Bournechurch Road', 'EN4 5YT','London', 'United Kingdom', '02088876354' , 'Michsimpson@gmail.com');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);

INSERT INTO Guest (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Mr','Jake','Simpson','20010318', '57 Bournechurch Road', 'EN4 5YT','London', 'United Kingdom', '02088876354' , 'Jakesimpson@gmail.com');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);

INSERT INTO Guest (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Miss','Frankie','Holiday','20010504', '25 Milner Road', 'EN6 7HY','London', 'United Kingdom', '02089987655' , 'Frankie001@gmail.com');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);

INSERT INTO Guest (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Mr','Liam','Moony','20010917', '11 Avendale Av', 'EN3 3XR','London', 'United Kingdom', '02084432256' , 'FootballKing@gmail.com');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);

INSERT INTO Guest (Title, FirstName, LastName, DOB, StreetAddress, PostCode, City, Country, PhoneNo, Email) VALUES
('Mr','Ryan','Crossley','20011129', '82 Cresent Road', 'EN5 9JJ','London', 'United Kingdom', '02089967836' , 'Fatboyryan@gmail.com');
SELECT @guestid = SCOPE_IDENTITY();
INSERT INTO PartyMembers VALUES (@guestid,@partyid);
```

```sql


INSERT INTO Booking VALUES
(@partyid,
 (SELECT TOP 1 AccomodationID FROM Accomodation WHERE AccType = 'Tent' AND Capacity = 6),
'20170601', '20170608');

INSERT INTO Booking VALUES
(@partyid,
 (SELECT TOP 1 AccomodationID FROM Accomodation WHERE AccType = 'Tent' AND Capacity = 4),
'20170601', '20170608');


COMMIT TRANSACTION CreateBooking2;
END TRY
BEGIN CATCH
ROLLBACK TRANSACTION CreateBooking2;
END CATCH;
```

Messages

```
(1 row(s) affected)
(1 row(s) affected)
(1 row(s) affected)
(1 row(s) affected)
(1 row(s) affected)
```

Results | Messages

	GuestID	Title	FirstName	LastName	DOB	StreetAddress	PostCode	City	Country	PhoneNo	Email	LeadGuest
23	139	Mr	Paul	Ribero	2007-05-28	18 FreshWater Avenue	RM6 4DT	London	United ...	07676345123	paulisthebest@hotmail.com	NO
24	140	Miss	Tara	Ribero	2010-12-20	18 FreshWater Avenue	RM6 4DT	London	United ...	07676345123	Tara123@gmail.com	NO
25	173	Mr	David	Jones	1982-03-06	39 Brookside	EN4 8TT	London	United ...	0782678392	DJones@trentschool.org	YES
26	174	Miss	Amelia	Scott	1970-05-09	1 littlegrove	EN4 89T	London	United ...	0776354789	Ascott@trentschool.org	NO
27	175	Mr	Fred	Henderson	1973-08-16	18 Churchhill Road	EN5 7NR	London	United ...	07662333456	Fhenderson@trentschool.org	NO
28	176	Miss	Kim	Kardasian	2001-10-12	52 Bournechurch Road	EN4 5YT	London	United ...	02084470036	Kimmygirl@gmail.com	NO
29	177	Miss	Michelle	Simpson	2001-03-18	57 Bournechurch Road	EN4 5YT	London	United ...	02088876354	Michsimpson@gmail.com	NO
30	178	Mr	Jake	Simpson	2001-03-18	57 Bournechurch Road	EN4 5YT	London	United ...	02088876354	Jakesimpson@gmail.com	NO
31	179	Miss	Frankie	Holiday	2001-05-04	25 Milner Road	EN6 7HY	London	United ...	02089987655	Frankie001@gmail.com	NO
32	180	Mr	Liam	Moony	2001-09-17	11 Avendale Av	EN3 3XR	London	United ...	02084432256	FootballKing@gmail.com	NO
33	181	Mr	Ryan	Crossley	2001-11-29	82 Cresent Road	EN5 9JJ	London	United ...	02089967836	Fatboyryan@gmail.com	NO
34	182	Miss	Ester	Oljua	2001-02-07	88 Park Street	EN7 4ZZ	London	United ...	02089995425	Prettygirl123@gmail.com	NO

PartyID	PartySize	Adults	Children	Pets	LeadGuestID
9	1	1	0	2	126
10	3	1	2	0	127
12	4	2	2	0	137
16	10	3	7	0	173

GuestID	PartyID
140	12
173	16
174	16
175	16
176	16
177	16
178	16
179	16
180	16
181	16
182	16

BookingID	PartyID	AccommodationID	CheckIn	CheckOut	NumOfNights
28	12	27	2017-04-18	2017-04-23	5
104	16	29	2017-06-01	2017-06-08	7
105	16	26	2017-06-01	2017-06-08	7

This query is essentially the same as query 2, however more guests are added as it is for a school party. I could not manage to allow multiple pitches to be assigned to one booking, so I had to create two bookings with the same partyID. I realised to do this query I could have maybe added another table named BookedAccommodation, which references the BookingID from the booking table. Then removed the accommodationID from the booking table, moving it to the BookedAccomodation table to show that a booking can have more than one accomodationID in the booking. So, the new table would have been used as an intersection table. However, because I had created the whole database and entered all the data into it, this would have been too time consuming.

4.

```
select Guest.GuestID, Guest.FirstName, Guest.LastName, Booking.CheckIn,Booking.CheckOut, Accomodation.AccType, (SELECT GETDATE()) AS TodaysDate
FROM Guest
INNER JOIN PartyMembers ON PartyMembers.GuestID = Guest.GuestID
INNER JOIN Booking ON PartyMembers.PartyID = Booking.PartyID
INNER JOIN Accomodation ON Booking.AccommodationID = Accomodation.AccomodationID
WHERE GETDATE() BETWEEN Booking.CheckIn AND Booking.CheckOut
ORDER BY AccType;
```

Results | Messages

GuestID	FirstName	LastName	CheckIn	CheckOut	AccType	TodaysDate
126	Willa	Walker	2017-04-18	2017-04-28	Caravan	2017-04-22 ...
137	Luke	Ribero	2017-04-18	2017-04-23	Motorhome	2017-04-22 ...
138	Bella	Ribero	2017-04-18	2017-04-23	Motorhome	2017-04-22 ...
139	Paul	Ribero	2017-04-18	2017-04-23	Motorhome	2017-04-22 ...
140	Tara	Ribero	2017-04-18	2017-04-23	Motorhome	2017-04-22 ...
122	Malcolm	Jordan	2017-04-14	2017-05-02	Motorhome	2017-04-22 ...

Query 4 was to list the guests staying overnight tonight, grouped by accommodation type. To do this I selected the guest's details from the guest table, the checkin and checkout dates from the booking table, and the acctype from the accommodation table. I then used three inner joins to the guest table. The first being the party members table to match the guest ids together, so we know what guest is associated with what party. The second table being the booking table to match the party ids together, so know we know what party is associated with what booking. Thirdly, the accommodation table to match the accommodation ids together, so know we know what accommodation type has been booked out. Then by using a WHERE clause to only return the guests who are booked in over the current date returns which guests are staying overnight tonight.

5

ytest.sql - J...PS\Joe Shepherd (52))* | SQLQuery1.sql - JO...\Joe Shepherd (53))*

```sql
SELECT Activity.*, (SELECT GETDATE()) AS CurrentDate FROM Activity
WHERE (10 BETWEEN SuitableAgeMin AND SuitableAgeMax)
AND (16 BETWEEN SuitableAgeMin AND SuitableAgeMax)
AND Date = dateadd(dd, datediff(dd, 0, getdate()) + 1, 0);
```

Results | Messages

	ActivityID	Description	SuitableAgeMin	SuitableAgeMax	LocationOfActivity	Date	StartTime	EndTime	TotalPeople	PricePerPerson	OrganiserID	CurrentDate
1	16	Horse Riding	10	20	18 Fredrick Manor	2017-04-25	10:30:00.0000000	14:00:00.0000000	5	20.00	1	2017-04-24 15:50:00.460
2	17	Arts and Crafts	8	16	21 Broadway Road	2017-04-25	12:00:00.0000000	13:30:00.0000000	10	10.00	3	2017-04-24 15:50:00.460

To return activities suitable for a 10-16 year old, a simple select statement with a WHERE clause to check if 10 or 16 is between the minAge and maxAge, and the dateadd function to search for activities that are on tomorrow is able return the data required. The dateadd function truncates the time from the getdate function because just using date = GETDATE does not work. The getdate + 1 gives tomorrows date. I added a current date column at the end to prove only activities set for tomorrow are returned.

6 .

Results | Messages

	BookingID	BookStatus
1	28	Payment Due
2	104	Deposit Due
3	105	Deposit Due

5.sql - JOESHEPS.H...\Joe Shepherd (56))*

```sql
UPDATE BookingStatus
SET BookStatus = 'Current Guest'
WHERE BookingID = 28;

UPDATE BookingStatus
SET BookStatus = 'Reserved'
WHERE BookingID = 28;

UPDATE BookingStatus
SET BookStatus = 'Reserved'
WHERE BookingID = 28;
```

82 %

Results | Messages

	BookingID	BookStatus
1	28	Reserved
2	104	Deposit Due
3	105	Deposit Due

For this query, the booking status had to be updated. To do this an UPDATE statement is all that is needed, setting the bookStatus to the value specified, using a WHERE clause to select which guests booking status needs to be updated.

The picture at the top shows the booking status before it was updated.

7.

6.sql - JOESHEPS.H...\Joe Shepherd (54))*

```sql
SELECT
(SELECT COUNT(GuestID) FROM PartyMembers
INNER JOIN Booking ON Booking.PartyID = PartyMembers.PartyID
WHERE
    (Booking.CheckIn BETWEEN '20170401' AND '20170430')
 OR (Booking.CheckOut  BETWEEN '20170401' AND '20170430')) AS TotalGuests,

SUM(Invoice.TotalPaid) AS MoneyPaid,
SUM ( Invoice.TotalPrice - Invoice.TotalPaid) AS MoneyDue
FROM Invoice
INNER JOIN Booking ON Booking.BookingID = Invoice.BookingID
WHERE (Booking.CheckIn BETWEEN '20170401' AND '20170430')
OR (Booking.CheckOut  BETWEEN '20170401' AND '20170430');
```

81 %

Results | Messages

	TotalGuests	MoneyPaid	MoneyDue
1	20	2005.00	2175.00

This query was to find out the total guests staying in the last month, and to find out the total money paid and total money due. To achieve this, I used a subquery for the TotalGuests column. This subquery uses the count function to count the number of people from the party members table, who have stayed at the campsite in the last month. To find out what guests stayed in the last month I had to inner join the booking table, matching the partyIDs and used a WHERE clause that returns the people who have a checkin or checkout date that is between the month of April.

To get the total money paid, I used the SUM function to add the TotalPaid column from invoice, and a sum of the TotalPrice minus the TotalPaid column in the invoice table for the money due column. Again, using an inner join on the booking table like before to add up only the guests who have stayed in the last month.

8.

6.sql - JOESHEPS.H...\Joe Shepherd (54))*

```sql
INSERT INTO Activity VALUES
('Horse Riding', 8, 16, '18 Fredrick Manor', '20170423', '14:00:00', '16:00:00', 10, 20.00, 1);

SELECT *, (SELECT DATENAME(dw ,Date) ) AS DAY FROM Activity WHERE Description ='Horse Riding';
```

81 %

Results | Messages

	ActivityID	Description	SuitableAgeMin	SuitableAgeMax	LocationOfActivity	Date	StartTime	EndTime	TotalPeople	PricePerPerson	OrganiserID	DAY
1	1	Horse Riding	8	16	18 Fredrick Manor	2017-04-19	10:30:00...	14:00:00....	5	20.00	1	Wednesday
2	2	Horse Riding	17	100	18 Fredrick Manor	2017-04-19	10:30:00...	14:00:00....	5	25.00	9	Wednesday
3	11	Horse Riding	8	16	18 Fredrick Manor	2017-04-23	14:00:00...	16:00:00....	10	20.00	1	Sunday

This query had to add a new horse riding activity to the rota between 2-4pm on a sunday. This was just a simple insert statement. I added DAY column at the end of the table to prove the activity was added on the following Sunday.

9.

income from activit...\Joe Shepherd (53))*

```sql
SELECT SUM(RobinsonShare) AS TotalActivityIncome FROM ActivityInvoice
INNER JOIN ActivityBooking ON ActivityInvoice.ActivityID = ActivityBooking.ActivityID
INNER JOIN Activity ON Activity.ActivityID = ActivityBooking.ActivityID
WHERE Activity.Date BETWEEN '20170401' AND '20170430';
```

81 %

Results | Messages

	TotalActivityIncome
1	12.00

This query had to return the total income from activities in the month of January. I did not have any activities that happened in the month of January, so I used April instead. I used the SUM function to add up the RobinsonShare column in the activity invoice table, and used an inner join to join the activity booking table by activityID, so that I could use a WHERE clause on the activity booking table to return only the activities that were booked in April.

10.

```sql
SELECT
    Guest.FirstName + ' ' + Guest.LastName AS Name,  Booking.NumOfNights ,
    Party.PartySize - 1 AS AdditonalGuests
FROM
    Booking
INNER JOIN Party ON Party.PartyID = Booking.PartyID
INNER  JOIN Guest ON Guest.GuestID = Party.LeadGuestID
WHERE Guest.FirstName = 'Quemby' AND Guest.LastName='Fitzpatrick' ;
```

Results | Messages

	Name	NumOfNights	AdditonalGuests
1	Quemby Fitzpatrick	8	3

This query had to find the total number of nights, and additional party members, for a given guest. This was achieved by joining two tables to the booking table. The first join being the party table by matching PartyID to the Booking table, then joining the guest table by GuestID on LeadGuestID. This finds out what party the lead guest is associated with, and uses the partysize column -1 to find additional guests, as the guest in question is not an additional guest so 1 had to be taken away from the party size.

11.

```sql
SELECT BannedGuest.*, PartyMembers.PartyID FROM BannedGuest
INNER JOIN PartyMembers ON PartyMembers.GuestID = BannedGuest.GuestID;

INSERT INTO Booking VALUES
(1,25, '20170701', '20170705');
```

Results | Messages

GuestID	DateBanned	Reasons	PartyID
110	2017-05-11	Fighting with other guests	1
111	2017-05-11	Vandalsim	1
114	2017-05-05	Unpaid fee	4
119	2017-05-11	Littering	5
120	2017-05-11	Abuse towards staff	6
123	2017-05-20	Vandalism	8
126	2017-04-28	Littering	9
127	2017-05-15	Unpaid fee	10
128	2017-04-15	Abuse towards staff	10
129	2017-04-15	Fighting with other guests	10

This query was to find out what guests have cause trouble in the past, and ensure they cannot make a booking again. To return the guests that cause trouble this was just a simple select statement, joining the Party table to show what party they are associated with. However, to make sure that they could never book again, I created a trigger on the booking table as seen below.

```sql
CREATE TRIGGER Banned ON dbo.Booking
after INSERT, UPDATE
AS

IF EXISTS
(

    SELECT 1 FROM inserted as x
 INNER JOIN PartyMembers ON x.PartyID = PartyMembers.PartyID
 INNER JOIN BannedGuest ON PartyMembers.GuestID = BannedGuest.GuestID

 )


 BEGIN
 RAISERROR('Banned Guest is trying to make a booking', 16, 1)
 ROLLBACK TRANSACTION;
 RETURN
 END
```

Messages

```
Command(s) completed successfully.
```

This trigger is for after data is inserted or updated in the booking table. It checks to see if a guest who is banned is part of a party that is trying to make a booking, it will come up as an error and not

allow the booking to go through. The result of executing the insert statement in the first screen shot produces this error below.

```
81 %
Messages
Msg 50000, Level 16, State 1, Procedure Banned, Line 16 [Batch Start Line 3]
Banned Guest is trying to make a booking
Msg 3609, Level 16, State 1, Line 4
The transaction ended in the trigger. The batch has been aborted.
```

Triggers/ Stored Procedures and Functions

A.

The first stored procedure is required that will create a new booking using parameters giving the detail of the date and time, the main guest and party members, and the type of accommodation required. To do this I first created another stored procedure to automatically work out the values that needed to be inserted into the invoice table.

This procedure is the same as the insert statement I used to insert the data into the Invoice table. It works out the total price of the booking, the deposit and sets the total paid as either the deposit price for advance bookings or the full price for on the day bookings.

```sql

CREATE PROCEDURE WorkoutPayment
@BookingID INT
AS
BEGIN

INSERT INTO Invoice VALUES
    (@BookingID,
    (SELECT SUM(NumOfNights * PricePerNight) FROM Booking, Accomodation WHERE Booking.BookingID = @BookingID AND
     Accomodation.AccomodationID = Booking.AccommodationID),
     0, 0);


IF (SELECT Booking.CheckIn FROM Booking WHERE BookingID = @BookingID) > GETDATE()
UPDATE Invoice
    SET Deposit = TotalPrice * 0.25,
    TotalPaid = TotalPrice * 0.25
    WHERE BookingID = @BookingID

END;
```

Query4.sql - JO...\Joe Shepherd (53))*

```sql
CREATE PROCEDURE CreateBooking
@partyid INT,
@checkin DATE,
@checkout DATE,
@acctype VARCHAR(30)
AS
BEGIN

INSERT INTO Booking VALUES
    (@partyid,
    (SELECT TOP 1 Accomodation.AccomodationID FROM Accomodation
     INNER JOIN Party ON Party.PartySize <= Accomodation.Capacity
     WHERE Party.PartyID = @partyid AND Accomodation.AccType = @acctype
     ORDER BY Accomodation.Capacity),
     @checkin,
     @checkout);

DECLARE @BookingID INT = SCOPE_IDENTITY();
EXEC WorkoutPayment @BookingID;

INSERT INTO BookingStatus VALUES
    (@BookingID, 'Reserved');

END;
```

Above is the Stored Procedure that creates a booking. It takes in parameters of the PartyID (which is essential the main guest and party members information), the check in and check out date, and the accommodation type. To find the accommodationID for the booking, I used a sub query that selects the top 1 accommodation, where the accommodation type is the same as the input parameter @acctype, and the capacity is greater than or equal to the party size. To achieve this an inner join was necessary to match the accommodation capacity to the party size.

Then I called the workout payment stored procedure to insert the data into the invoice table automatically from the booking id that was just created. Executing the query produces the results shown below.

```sql
EXECUTE CreateBooking 12, '20170609', '20170615', 'Tent';
SELECT * FROM Booking;
SELECT * FROM Invoice;
SELECT * FROM BookingStatus;
```

	BookingID	PartyID	AccommodationID	CheckIn	CheckOut	NumOfNights
13	105	16	26	2017-06-01	2017-06-08	7
14	115	12	26	2017-06-09	2017-06-15	6

	AccInvoiceID	BookingID	TotalPrice	Deposit	TotalPaid
13	163	105	105.00	26.25	26.25
14	166	115	90.00	22.50	22.50

	BookingID	BookStatus
13	105	Deposit Due
14	115	Reserved

B.

SQLQuery4.sql - JO...\Joe Shepherd (53))*

```sql
CREATE PROCEDURE ActivityList
AS
BEGIN


    SELECT
    Activity.ActivityID,
    Activity.Description,
    TotalPeople - NumOfPeople AS Spaces
    FROM Activity
    INNER JOIN ActivityBooking ON Activity.ActivityID = ActivityBooking.ActivityID
    WHERE Activity.Date >= GETDATE()
    ORDER BY Activity.ActivityID;


    SELECT Activity.ActivityID , Guest.FirstName +' '+ Guest.LastName AS Participant
    FROM Activity
    INNER JOIN Activitybooking oN ActivityBooking.ActivityID = Activity.ActivityID
    INNER JOIN PartyMembers ON ActivityBooking.PartyID = PartyMembers.PartyID
    inner join guest ON Guest.GuestID = PartyMembers.GuestID
    WHERE Activity.Date >= GETDATE()
    order by ActivityID;

END;
```

SQLQuery4.sql - JO...\Joe Shepherd (53))*

```sql
EXECUTE ActivityList;
```

89 %

Results | Messages

	ActivityID	Description	Spaces
1	3	Arts and Crafts	9
2	4	Go-Karting	13
3	5	Rock Climbing	9
4	6	Go-Karting	6
5	7	Arts and Crafts	8
6	8	Swimming	18
7	9	WaterSkiing	5
8	10	Assualt Course	12

	ActivityID	Participant
1	3	Willa Walker
2	4	Dahlia Oliver
3	4	Cynthia Dill...
4	5	Nash Rowe
5	6	Cadman N...
6	7	Malcolm J...
7	8	Jesse We...
8	8	Chiquita D...
9	9	Cadman N...
10	10	Joelle Barron
11	10	Cyrus Tillm...
12	10	Harper Val...

This procedure was to produce an activity list with the participants involved, indicating any spaces on the activities. To do this I had to create two separate tables, one showing the number of people, and the spaces in each activity, and another table showing who is involved in each activity.

To return how many spaces are left in each activity, I used the totalpeople – numofpeople from the activity and activity booking table respectively using a join on the activityID to match the tables together.

To get each participate in each activity I had to use three joins on the activity table. The first was the activity booking table, which matches the activity to the booking, then another join on the party members, to match which party member is in each party, and finally the guest table, to match the guest's details to each party. From this It then returns the names of the guests in each activity. The results are shown to the left.

C.

```
ityinice.sql -...PS\Joe Shepherd (58))        JOESHEPS.Happy Val...site - dbo.Booking        SQLQuery6
CREATE TRIGGER CanceledBooking ON Booking
FOR DELETE
AS
BEGIN


    DELETE FROM ActivityInvoice
    WHERE ActivityInvoice.PartyID = (SELECT deleted.PartyID FROM deleted)

    DELETE FROM ActivityBooking
    WHERE ActivityBooking.PartyID = (SELECT deleted.PartyID FROM deleted)

END ;
```

This is a trigger which is actioned when a booking is cancelled. This will check if the party have made any additional activity bookings and remove these from the system. To achieve this the Trigger is set to FOR DELETE, and then deletes any data from the activity invoice and activity booking table, that have a matching party ID as the party ID in the booking that is being cancelled. The activity invoice table had to be delete first as this references the activity booking table. Results of before and after are shown below. You can see that the party ID being deleted from the booking is no longer in the activity booking or activity invoice table.

Before

Results | Messages

	PartyID	ActivityID	NumOfPeople
1	3	1	1
2	3	6	4
3	3	9	1
4	6	8	2

After

```
select * from ActivityBooking
DELETE FROM Booking WHERE PartyID = 3

Select * from ActivityBooking
```

Results | Messages

	PartyID	ActivityID	NumOfPeople
1	6	8	2
2	7	7	2
3	8	10	3
4	9	3	1

After

```
select * from ActivityInvoice
```

Results | Messages

PartyID	ActivityID	OrganiserID	TotalCost	RobinsonShare	OrgainserShare
7	7	3	20.00	2.00	18.00
9	3	3	10.00	1.00	9.00
6	8	6	10.00	1.00	9.00
8	10	10	60.00	6.00	54.00