CSI_5_BDD: Coursework CASE STUDY C

Deadline: 17:00 Wed 12 May 2021

Student number: 3807942

Table of Contents

Design 4
Entity-Relationship Diagrams 4
Considerations for Questions 3 & 4 5
Functional Dependencies 5
Customer 5
Reservation 6
Employees 6
Seaport 6
CruiseLinerEmployees 6
Ship 6
Voyage 7
Meal 7
Activities 7
Cabins 7
Ticket 7
Ticket_Activities 8
Implementation 9
Create statements 9
Insert Statements 12
SQL queries 26
Query 1 26
Query 2 27
Query 3 28
Query 4 30
Query 5 32
Dashboard (Bhatti, 2021) 36
Video Demo (Bhatti, 2021) 37
References 38

Figure 1 4
Figure 2 11
Figure 3 12
Figure 4 13
Figure 5 13
Figure 6 14
Figure 7 14
Figure 8 15
Figure 9 15

Figure 10 16
Figure 11 16
Figure 12 17
Figure 13 17
Figure 14 18
Figure 15 18
Figure 16 19
Figure 17 19
Figure 18 20
Figure 19 20
Figure 20 21
Figure 21 22
Figure 22 23
Figure 23 24
Figure 24 24
Figure 25 25
Figure 26 25
Figure 27 26
Figure 28 27
Figure 29 27
Figure 30 27
Figure 31 27
Figure 32 28
Figure 33 28
Figure 34 29
Figure 35 29
Figure 36 30
Figure 37 31
Figure 38 31
Figure 39 31
Figure 40 34
Figure 41 34
Figure 42 35
Figure 43 36

Design

Entity-Relationship Diagrams

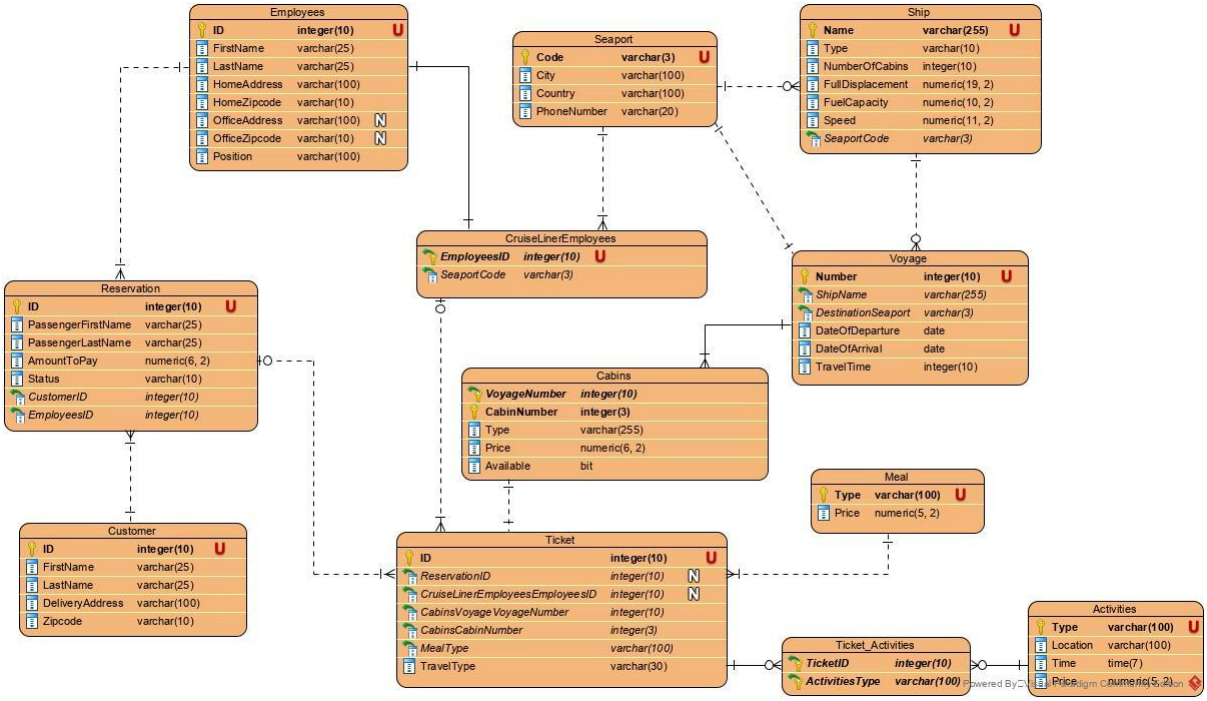

Figure 1

(Kroenke & Auer, 2016) was used to address the cardinalities of each of the tables for instance a ship would always belong to a seaport however a seaport may not always have a ship associated with it. Furthermore, a voyage will always be associated with a ship but until a ship departs on a voyage, an entry for the ship in the Voyage table would not exist. Also, we are developing the database on the assumption that all crew members will belong to a seaport. Although a passenger may choose to not have any meal during their voyage, the Meal and Ticket tables show a minimum cardinality of one as we have assumed that no meal could be a meal type in the Meal table. Lastly, tickets could be linked to activities however, the Ticket_Activities table is used to avoid many to many relationships between the Ticket and Activities tables where the minimum cardinality is zero as a ticket may not be related to an activity and an activity may not be related to any tickets.

The Employees table will contain all of the employees for LSBU Cruise Liners however, employees in the CruiseLinerEmployees table will be crew members which would board a ship during a voyage thus they may have a SeaportCode for where they would board the voyage from. The Employees table have an OfficeAddress and OfficeZipcode attribute which can be null for employees that do not work in an office, for example, ship crew members.

A seaport belongs to a city in a country and would have a phone number and would be uniquely identified by the code assigned to the seaport which has been included in the Seaport table. Ships a classified by type, the number of cabins, fuel displacement, fuel capacity and speed and would be identified by their name which is included as attributes in the Ship table. Additionally, the SeaportCode has been included in the Ship table to accommodate for the Ship starting and ending a voyage in different countries which would need to be updated in the database. A voyage will be uniquely identified by an identifier and would contain the date of departure and arrival and the total time travelled whilst linking to the Seaport table to retrieve the destination for the voyage and would link to the Ship table for the ship that would depart on the voyage.

The Customer and Reservation tables have been split as customers can make multiple reservations under the assumption that a customer could make a reservation for their wife or children thus, the Customer table contains the first and last name of the customer while the Reservation table would contain the first and last name of the passenger which would depart on the voyage. A reservation would contain the amount to pay, the status of the reservation for if it has been paid and would contain the ID of the employee who issued the reservation.

A ticket would link to a ReservationID or crew members EmployeeID wherein the Ticket table, either attribute could be null but one of the attributes must contain an ID as shown by the zero or one cardinality on the Reservation and CruiseLinerEmployees tables. A ticket would be linked to a cabin via a CabinNumber and VoyageNumber in the Cabins table where, even if a reservation hasn't been paid for a ticket, the associated cabin on that voyage would depict an unavailable status as that cabin on that voyage would be reserved to avoid double bookings. The Cabins table would also depict the type of cabin it is such as a luxury suite or an economy single cabin with an associated price as detailed in the case study. Furthermore, a ticket would link to a meal type for whether meals would be included and for certain dietary restrictions with associated prices. Lastly, a ticket contains a TravelType attribute to account for a passenger having two tickets, one for departing and another for returning voyages.

Considerations for Questions 3 & 4

The case study details that payments must be made with a specific payment type; the payment type wouldn't affect the functionality of the database thus payment information and transactional data have been omitted from Figure 1. Although the case study details on-board entertainment and fitness activities separately, they have been grouped under the Activities table where they would be uniquely identified by the name of the activity while occurring at the same location on each ship however, the time and price of the activity will vary depending on the type of activity. Furthermore, the case study states that a passenger table may be required however, by combining the passenger fields into a reservation and allowing a customer to make multiple reservations, the need for a passenger table has been circumvented.

To retrieve the number of reservations an employee has issued, the ID of the employee would be recorded when each reservation is made for a customer. Additionally, the Reservations table contains a status field to accommodate for complete, incomplete and cancelled reservations. To identify available cabins, the Cabins table has an availability attribute assigned to each cabin in a specific voyage which uses a bit to indicate the availability of the cabin with available being represented by a one and unavailable being a zero. Lastly, prices have been assigned to each meal type, activity and cabin to determine the price to be paid before and after VAT.

Functional Dependencies

(Kroenke & Auer, 2016) was used to construct the functional dependencies in each of the database tables however, although the source identifies multivalued dependencies, they aren't necessary for the construction of the coursework other than providing a better understanding of how the system will function thus they shall be omitted from the report.

Customer

CustomerID → (FirstName, LastName, DeliveryAddress, Zipcode)

The CustomerID attribute is the only attribute that is unique within the table as a customer could share the same name as another customer and multiple customers could share the same delivery address if they live in the same building or if they choose to have the tickets delivered to the office that issued their reservations.

Reservation

ReservationID → (PassengerFirstName, PassengerLastName, AmountToPay, Status, CustomerID, EmployeesID)

(ReservationID, EmployeesID) → (PassengerFirstName, PassengerLastName, AmountToPay, Status, CustomerID)

(ReservationID, CustomerID) → (PassengerFirstName, PassengerLastName, AmountToPay, Status, EmployeesID)

(ReservationID, CustomerID, EmployeesID) → (PassengerFirstName, PassengerLastName, AmountToPay, Status)

Although the Reservation table has multiple functional dependencies where composite determinants can be constructed with some of the attributes in the table, all of the determinants in each of the functional dependencies include the primary key ReservationID and use a combination of foreign keys to identify an individual record in the table. Using a composite key for CustomerID and EmployeesID could return multiple records if a customer makes a reservation on behalf of multiple individuals thus, using CustomerID and EmployeesID in a composite key would create a multivalued dependency.

Employees

EmployeesID → (FirstName, LastName, HomeAddress, HomeZipcode, OfficeAddress, OfficeZipcode, Position)

(EmployeesID, Position) → (FirstName, LastName, HomeAddress, HomeZipcode, OfficeAddress, OfficeZipcode)

The Employees table can only return a single record if an EmployeesID is referenced when querying the table thus, although the Position attribute is included in the composite key of one of the functional dependencies, to return only one record the Position attribute must rely on the EmployeesID primary key to identify a specific employee.

Seaport

SeaportCode → (City, Country, PhoneNumber)

PhoneNumber → (SeaportCode, City, Country)

The primary key SeaportCode would be unique in the Seaport table thus it is included in the Seaport table. PhoneNumber has been used as a determinant for a functional dependency under the assumption that a phone number is unique for each seaport. Using a Country and City in the determinant would return multiple SeaportCodes and PhoneNumbers assuming that each city in a country could have multiple seaports thus creating a multivalued dependency.

CruiseLinerEmployees

CruiseLinerEmployeesID → SeaportCode

The only functional dependency in the CruiseLinerEmployees table would be from an EmployeesID identifying the Seaport an employee would be working at as multiple employees could work at the same seaport while the EmployeesID would be unique to each employee.

Ship

ShipName → (Type, NumberOfCabins, FullDisplacement, FuelCapacity, Speed, SeaportCode)

Although the Ship table could have multiple functional dependencies when each attribute in the Ship table is paired with the ShipName to act as a composite key, any functional dependencies would stem

from the unique primary key ShipName being used to identify a specific record in the Ship table thus any combination of composite keys for the Ship table have been omitted. Furthermore, the attributes in the Ship table other than ShipName can be shared between multiple ships thus creating a multivalued dependency if used individually or in a composite key assuming the ShipName is not included in the composite key.

Voyage

Number → (ShipName, DestinationSeaport, DateOfDeparture, DateOfArrival, TravelTime)

The voyage table could have a multifunctional dependency if the determinant was any of the other attributes besides the voyage number. The only functional dependency would be from a voyage number identifying the ship partaking on the voyage, the destination of the voyage and the time it would take to complete the voyage as each voyage number is unique.

Meal

MealType → Price

MealType is used as the determinant as it would be the unique field in the Meal table as meals could share the same price whilst assuming that a MealType would dictate the dietary requirements of a passenger.

Activities

ActivitiesType → (Location, Time, Price)

(Location, Time) → ActivitiesType

The type of activity is used as a determinant for the Location, Time and Price of the activity however, Location and Time can be used as a determinant to dictate the type of activity that will take place assuming that only one activity can occur in a specific location at a specified time. We have also assumed that activities will be scheduled at the same time and at the same location on each voyage to reduce complexity when handling the implementation of activities. Using Price and Location or Price and Time as a composite key would form a multivalued dependency as activities may share these attributes.

Cabins

(VoyageNumber, CabinNumber) → (Type, Price, Available)

Type → Price

The composite key of VoyageNumber and CabinNumber creates a functional dependency as they are both primary keys used to identify a cabin on a specific voyage furthermore, the Type of a cabin would return only one Price assuming that the price of each cabin Type is the same on each Ship. Using Price or the Available attribute in the determinant would create a multivalued dependency as multiple cabins would share the same availability status or Price.

Ticket

TicketID → (ReservationID, CruiseLinerEmployeesID, VoyageNumber, CabinNumber, MealType, TravelType)

(VoyageNumber, CabinNumber) → (TicketID, ReservationID, CruiseLinerEmployeesID, MealType, TravelType)

Similar to the Reservation table, any combination of attributes with the unique primary key TicketID to form a composite key would create a functional dependency thus returning a single record from the Ticket table thus, to reduce complexity they have been omitted. Additionally, VoyageNumber and

CabinNumber can be used to form a composite key to create a functional dependency. Any other composite keys which isn't a superset from TicketID or the composite key of VoyageNumber and CabinNumber would create a multivalued dependency.

Ticket_Activities

Using TicketID as the determinant would return each activity a passenger would partake in on a voyage while using ActivitiesType as the determinant would return the TicketID of every passenger who has taken part in that activity thus creating multivalued dependencies if either the TicketID or ActivitiesType is used as a determinant.

Implementation

All source code can be found at (Bhatti, 2021).

Create statements

```
1  CREATE DATABASE Cruise_Liner;
2  GO
3
4  USE Cruise_Liner;
5  GO
6
7  CREATE TABLE Seaport(
8   Code    Varchar(3) NOT NULL,
9   City    Varchar(100)    NOT NULL,
10  Country Varchar(100)    NOT NULL,
11  PhoneNumber Varchar(20) NOT NULL,
12  CONSTRAINT  Code    PRIMARY KEY(Code)
13 );
14
15 CREATE TABLE Ship(
16  [Name]  Varchar(255)    NOT NULL,
17  [Type]  Varchar(10) NOT NULL,
18  NumberOfCabins  Int NOT NULL,
19  FullDisplacement    Numeric(19,2)   NOT NULL,
20  FuelCapacity    Numeric(10,2)   NOT NULL,
21  Speed   Numeric (11,2)  NOT NULL,
22  SeaportCode Varchar(3) NOT NULL,
23  CONSTRAINT  ShipName PRIMARY KEY([Name]),
24  CONSTRAINT  OriginSeaport   FOREIGN KEY(SeaportCode)    REFERENCES
Seaport(Code)
25 );
26
27 CREATE TABLE Voyage(
28  Number  Int NOT NULL    IDENTITY (1,1),
29  ShipName    Varchar(255)    NOT NULL,
30  DestinationSeaport  Varchar(3) NOT NULL,
31  DateOfDeparture Date    NOT NULL,
32  DateOfArrival   Date    NOT NULL,
33  TravelTime  Int NOT NULL,
34  CONSTRAINT Number   PRIMARY KEY(Number),
35  CONSTRAINT  ShipName2   FOREIGN KEY(ShipName)   REFERENCES
Ship([Name]),
36  CONSTRAINT  DestinationSeaport  FOREIGN KEY(DestinationSeaport)
REFERENCES  Seaport(Code)
37 );
38
39 CREATE TABLE Cabins(
40  VoyageNumber    Int NOT NULL,
41  CabinNumber Int NOT NULL,
42  [Type]  Varchar(255)    NOT NULL,
43  Price   Numeric(6,2)    NOT NULL,
44  Available   Bit NOT NULL,
45  CONSTRAINT  CabinsPK    PRIMARY KEY(VoyageNumber, CabinNumber),
46  CONSTRAINT  VoyageFK    FOREIGN KEY(VoyageNumber)   REFERENCES
Voyage(Number)
47 );
48
49 CREATE TABLE Meal(
50  [Type]  Varchar(100)    NOT NULL,
51  Price   Numeric(5,2)    NOT NULL,
52  CONSTRAINT MealType PRIMARY KEY([Type])
53 );
54
55 CREATE TABLE Activities(
56  [Type]  Varchar(100)    NOT NULL,
57  [Location] Varchar(100) NOT NULL,
```

```
58  [Time]  Time(7) NOT NULL,
59  Price   Numeric(5,2)    NOT NULL,
60  CONSTRAINT  ActivityType    PRIMARY KEY([Type])
61 );
62
63 CREATE TABLE Customer(
64  ID  Int NOT NULL    IDENTITY (1,1),
65  FirstName   Varchar(25) NOT NULL,
66  LastName    Varchar(25) NOT NULL,
67  DeliveryAddress Varchar(100)    NOT NULL,
68  Zipcode Varchar(10) NOT NULL,
69  CONSTRAINT  CustomerID  PRIMARY KEY(ID)
70 );
71
72 CREATE TABLE Employees(
73  ID  Int NOT NULL    IDENTITY(1,1),
74  FirstName   Varchar(25) NOT NULL,
75  LastName    Varchar(25) NOT NULL,
76  HomeAddress Varchar(100)    NOT NULL,
77  HomeZipcode Varchar(10) NOT NULL,
78  OfficeAddress   Varchar(100)    NULL,
79  OfficeZipcode   Varchar(10) NULL,
80  Position    Varchar(100)    NOT NULL,
81  CONSTRAINT  EmployeesID PRIMARY KEY(ID)
82 );
83
84 CREATE TABLE CruiseLinerEmployees(
85  EmployeesID Int NOT NULL,
86  SeaportCode Varchar(3) NOT NULL,
87  CONSTRAINT  CrewID  PRIMARY KEY(EmployeesID),
88  CONSTRAINT  EmployeesFK2    FOREIGN KEY(EmployeesID)    REFERENCES
Employees(ID),
89  CONSTRAINT  SeaportCode FOREIGN KEY(SeaportCode)    REFERENCES
Seaport(Code)
90 );
91
92 CREATE TABLE Reservation(
93  ID  Int NOT NULL    IDENTITY(1,1),
94  PassengerFirstName  Varchar(25) NOT NULL,
95  PassengerLastName   Varchar(25) NOT NULL,
96  AmountToPay Numeric(6,2)    NOT NULL,
97  [Status]    Varchar(10) NOT NULL,
98  CustomerID  Int NOT NULL,
99  EmployeesID Int NOT NULL,
100 CONSTRAINT  ReservationID   PRIMARY KEY(ID),
101 CONSTRAINT  CustomerFK  FOREIGN KEY(CustomerID) REFERENCES
Customer(ID),
102 CONSTRAINT  EmployeesFK FOREIGN KEY(EmployeesID)    REFERENCES
Employees(ID)
103);
104
105CREATE TABLE Ticket(
106 ID  Int NOT NULL    IDENTITY(1,1),
107 ReservationID   Int NULL,
108 CrewID  Int NULL,
109 VoyageNumber    Int NOT NULL,
110 CabinNumber Int NOT NULL,
111 MealType    Varchar(100)    NOT NULL,
112 TravelType  Varchar(30)    NOT NULL,
113 CONSTRAINT  TicketID    PRIMARY KEY(ID),
```

```
114 CONSTRAINT  ReservationFK   FOREIGN KEY(ReservationID)  REFERENCES
Reservation(ID),
115 CONSTRAINT  CrewFK  FOREIGN KEY(CrewID) REFERENCES
CruiseLinerEmployees(EmployeesID),
116 CONSTRAINT  CabinFK FOREIGN KEY(VoyageNumber, CabinNumber)  REFERENCES
Cabins(VoyageNumber, CabinNumber),
117 CONSTRAINT  MealTypeFK  FOREIGN KEY(MealType)   REFERENCES
Meal([Type])
118);
119
120CREATE TABLE Ticket_Activities(
121 TicketID    Int NOT NULL,
122 ActivityType    Varchar(100)    NOT NULL,
123 CONSTRAINT  TicketFK    FOREIGN KEY(TicketID)   REFERENCES  Ticket(ID),
124 CONSTRAINT  ActivitiesFK    FOREIGN KEY(ActivityType)   REFERENCES
Activities([Type])
125);
```

Figure 2

(Kroenke & Auer, 2016) was referenced when creating the constraints for each of the database tables and specifically when creating composite primary and composite foreign keys.

The specification details that seaport codes are a unique three-character code however when creating the table in the database we have chosen to use a primary key constraint on the seaport code attribute instead of a unique constraint as primary keys are inherently unique and this mitigates the need for a candidate key in the seaport table. Similarly, the ship name in the ship table conforms to the same criteria in that it is unique to each ship thus instead of using a candidate key for the ship table and making the ship name conform to a unique constraint, we have made the ship name the primary key in the ship table. We have also chosen to store the original seaport with each ship in the ship table thus when a ship arrives at a different seaport its original seaport code in the ship table will be updated before departing on another voyage.

In the voyage table, we have used the voyage number attribute as the primary key as each voyage will have a unique voyage number. We have also used a foreign key constraint to relate a voyage to each ship that will depart on the voyage and we have used a foreign key constraint to relate each voyage to a destination through the DestinationSeaport attribute.

Although a candidate key in the Cabins table would simplify queries that would be made in the future to access the table, the composite key contains only two attributes where a candidate key would be more suitable if the Cabins table used three or more attributes in the composite key as its primary key. We have also used a foreign key constraint on voyage number thus, although voyage number and cabin number may appear multiple times throughout the Cabins table, the combination of voyage number and cabin number will dictate the record to be returned when querying the database for available cabins. We have chosen to assign a price to each cabin type that will dictate how much each individual must pay each night for staying in a specific cabin. Even though we could have used the *char* or *varchar* datatype for availability, we have favoured the *bit* datatype for availability where a 0 would dictate that a cabin on a specific voyage isn't available while a 1 shows a cabin on a voyage is available as availability can only have 2 states.

For the Meal table, we have used a primary key constraint on the type of meal where the meal type dictates the dietary restrictions of each passenger during their voyage where we have assumed that a meal-type will have an associated price. As stated in the specification, activity will have a type, location and time where a primary key constraint has been used on the activity type. Similarly, we have assumed each activity will have an associated price like in the Meal table.

The Customer table implements a primary key constraint on the ID which acts as a unique identifier for each customer which is similar to the Employees and CruiseLinerEmployees table however, in the CruiseLinerEmployees table there are foreign key constraints that relate to the primary key of the table to the Employees table and the SeaportCode where an employee would relate to the Code attribute in the Seaport table. An issue with the design of the Employees table is that the OfficeAddress and OfficeZipcode attributes could be null thus, although a clerk must have an OfficeAddress and OfficeZipcode, there could be issues when querying the table if these fields were null as a clerk wouldn't know what office to work at, therefore, we must ensure that Clerks have an OfficeAddress and OfficeZipcode when inserting into the table.

The Reservation table uses a primary key constraint on the ID attribute to uniquely identify each record in the table while the CustomerID and EmployeesID attributes are related to their respective tables via foreign key constraints. As the Status attribute has more than 2 states, we have favoured the *varchar* datatype instead of the *bit* datatype for this attribute so that we can record cancelled, complete and incomplete reservations in the table.

The Ticket table has 4 foreign key constraints thus, we have implemented a candidate key to act as the identifier in the table to simplify queries to the table in the future. Although the travel type attribute has 2 states, either as a departure or return ticket, we have favoured the use of a *varchar* data type instead of the *bit* datatype as using the *bit* datatype would be confusing as a 0 or 1 doesn't necessarily indicate a departure or return for the travel type attribute. Similar to the Employees table, an issue in how the table is designed would be that the ReservationID or CrewID must be populated in the table, not both as a Ticket must belong to either a Reservation or a Crew member, not both. If both CrewID and ReservationID are null then a Ticket would be issued with no owner thus we must ensure that either attribute is populated when inserting into the Ticket table.

The Ticket_Activities table uses foreign key constraints to relate the TicketID to the Ticket table and the ActivitiyType to the Activities table where both the TicketID and ActivitiesType can be repeated within the table.

Insert Statements

```
INSERT INTO Customer(FirstName, LastName, DeliveryAddress, Zipcode)
VALUES
('Spencer', 'Vazquez', 'Stinsford Hill House, Stinsford', 'DT2 8PS'),
('Porfirio', 'Jordan', 'Torrington, 10 Fairmile Lane, Cobham', 'KT11 2DJ'),
('Autumn', 'Flores', '2 Alaska Villas, Hull', 'HU8 7TF'),
('Michelle', 'Torres', 'Dashmonden Mews, High Halden Road, Biddenden', 'TN27 8BD'),
('Hong', 'Berg', 'Myotis Barn, South Duffield Road, Osgodby', 'YO8 5HW'),
('Alec', 'Velasquez', 'Top Floor Flat, 37 North Road, Brighton', 'BN1 1YB'),
('Norma', 'Cochran', '64 Beech Road, Tiverton', 'EX16 6HS'),
('Jaclyn', 'Blevins', '144 Main Street, Frizington', 'CA26 3SB'),
('Philip', 'Prince', 'Close Bye, Hynetown Road, Strete', 'TQ6 0RS'),
('Fausto', 'Case', '6 Mayfield Drive, Port Isaac', 'PL29 3SL'),
('Orval', 'Garrison', '2 Southon Close, Portslade', 'BN41 2RX'),
('Maura', 'Anthony', '36 Tyninghame Avenue, Wolverhampton', 'WV6 9PW'),
('Arthur', 'Hester', 'Burfield House, Morley Lane, Wymondham', 'NR18 9BT'),
('Allison', 'Tapia', '3 Melville Close, Loughborough', 'LE11 4FN'),
('Victor', 'Oconnell', '32B High Street, Leamington Spa', 'CV31 1LW'),
('Jerrold', 'Hayden', 'The Old Vicarage, Star', 'SA35 0AR'),
('Rico', 'Ibarra', 'Oakleaze, Bristol Road, Compton Martin', 'BS40 6NH'),
('Dale', 'Herman', '15 Kenning Road, Hoddesdon', 'EN11 9HE'),
('Mari', 'Bailey', '91 Beresford Road, Portsmouth', 'PO2 0NG'),
('Britt', 'Pittman', '2 Pear Tree Acre, Thorp Arch', 'LS23 7AS');
```

Figure 3

	ID	FirstName	LastName	DeliveryAddress	Zipcode
1	1	Spencer	Vazquez	Stinsford Hill House, Stinsford	DT2 8PS
2	2	Porfirio	Jordan	Torrington, 10 Fairmile Lane, Cobham	KT11 2DJ
3	3	Autumn	Flores	2 Alaska Villas, Hull	HU8 7TF
4	4	Michelle	Torres	Dashmonden Mews, High Halden Road, Biddenden	TN27 8BD
5	5	Hong	Berg	Myotis Barn, South Duffield Road, Osgodby	YO8 5HW
6	6	Alec	Velasquez	Top Floor Flat, 37 North Road, Brighton	BN1 1YB
7	7	Norma	Cochran	64 Beech Road, Tiverton	EX16 6HS
8	8	Jaclyn	Blevins	144 Main Street, Frizington	CA26 3SB
9	9	Philip	Prince	Close Bye, Hynetown Road, Strete	TQ6 0RS
10	10	Fausto	Case	6 Mayfield Drive, Port Isaac	PL29 3SL
11	11	Orval	Garrison	2 Southon Close, Portslade	BN41 2RX
12	12	Maura	Anthony	36 Tyninghame Avenue, Wolverhampton	WV6 9PW
13	13	Arthur	Hester	Burfield House, Morley Lane, Wymondham	NR18 9BT
14	14	Allison	Tapia	3 Melville Close, Loughborough	LE11 4FN
15	15	Victor	Oconnell	32B High Street, Leamington Spa	CV31 1LW
16	16	Jerrold	Hayden	The Old Vicarage, Star	SA35 0AR
17	17	Rico	Ibarra	Oakleaze, Bristol Road, Compton Martin	BS40 6NH
18	18	Dale	Herman	15 Kenning Road, Hoddesdon	EN11 9HE
19	19	Mari	Bailey	91 Beresford Road, Portsmouth	PO2 0NG
20	20	Britt	Pittman	2 Pear Tree Acre, Thorp Arch	LS23 7AS

Figure 4

```
INSERT INTO Employees(FirstName, LastName, HomeAddress, HomeZipcode, OfficeAddress, OfficeZipcode, Position)
VALUES
('Sung', 'Frazier', 'Flats 1-17, 37 - 39 Marine Parade, Worthing', 'BN11 3PH', 'Paddock View, Netherfield Lane, Stanstead Abbotts', 'SG12 8HD', 'Clerk'),
('Maximo', 'Ibarra', '12 Cumberland Avenue, Bedlington', 'NE22 6DH', 'Paddock View, Netherfield Lane, Stanstead Abbotts', 'SG12 8HD', 'Clerk'),
('Newton', 'Combs', '440 Stockport Road West, Bredbury', 'SK6 2EE', '1 Mill Cottages, Mill Road, Long Stratton', 'NR15 2RT', 'Clerk'),
('Brant', 'Irwin', 'Somerton House, Somerton Rise, Boothby Graffoe', 'LN5 0LS', '1 Mill Cottages, Mill Road, Long Stratton', 'NR15 2RT', 'Clerk'),
('Cameron', 'Pope', '9 Lord Rosebery Mews, Norwich', 'NR7 0GX', 'Doxford Newhouses Farm, Doxford', 'NE67 5EA', 'Clerk'),
('Olga', 'Gates', '28 Whalley Road, Clitheroe', 'BB7 1AW', 'Doxford Newhouses Farm, Doxford', 'NE67 5EA', 'Clerk'),
('Hilario', 'Cordova', '8 College Way, Nottingham', 'NG8 4JH', '67A Devonport Road, Plymouth', 'PL3 4DL', 'Clerk'),
('Eric', 'Lutz', 'Brook Hollow, Heath Lane, Gunthorpe', 'NR24 2NT', '67A Devonport Road, Plymouth', 'PL3 4DL', 'Clerk'),
('Raquel', 'Kelley', 'Flat 2, Keynes Court, 593 Ringwood Road, Ferndown', 'BH22 9AJ', '170 Byron Road, Southampton', 'SO19 6QW', 'Clerk'),
('Latasha', 'Mayer', '92 Park Road, Chesterfield', 'S40 2JX', '170 Byron Road, Southampton', 'SO19 6QW', 'Clerk'),
('Keneth', 'Patel', '11A Quebec Gardens, Blackwater', 'GU17 9DE', null, null, 'Crew'),
('Rosetta', 'Barrera', '9 Moorhouses, Hightown', 'L38 9ER', null, null, 'Crew'),
('Jared', 'Hudson', '16 Pen Y Dre, Llanrwst', 'LL26 0BG', null, null, 'Crew'),
('Gordon', 'Lloyd', 'Greenlands, 93 Skipton Road, Silsden', 'BD20 9DA', null, null, 'Crew'),
('Carla', 'Barajas', 'Hillcrest, Birmingham Road, Kingshurst', 'B37 6RB', null, null, 'Crew'),
('Kate', 'Daugherty', 'White Gates, Mayflower Close, Bawtry', 'DN10 6NG', null, null, 'Crew'),
('Kathleen', 'Dixon', '8 Whitsand Bay Holiday Park, Millbrook', 'PL10 1JZ', null, null, 'Crew'),
('Lula', 'Wolf', '5 Dennis Road, Padstow', 'PL28 8DD', null, null, 'Crew'),
('Dave', 'Nichols', '2 Summerhill Gardens, Market Drayton', 'TF9 1BG', null, null, 'Crew'),
('Demarcus', 'Chavez', '1 Swaledale, Sunderland', 'SR6 8AH', null, null, 'Crew'),
('Quentin', 'Gill', '3 Berrymoor Road, Banbury', 'OX16 9QD', null, null, 'Crew'),
('Jimmy', 'Kelly', '14 Northerwood Avenue, Lyndhurst', 'SO43 7DU', null, null, 'Crew'),
('Elias', 'Mendez', 'Palmer House, Berrier', 'CA11 0XD', null, null, 'Crew'),
('Cheryl', 'Wong', '2 Chapel Street, Caistor', 'LN7 6UF', null, null, 'Crew'),
('Kayla', 'Osborne', '3 Linforth Way, Coleshill', 'B46 3LH', null, null, 'Crew'),
('Sonya', 'Black', '88 The Oval, Leeds', 'LS14 6BB', null, null, 'Crew'),
('Elliott', 'Henderson', '14 Ardleigh Green Road, Hornchurch', 'RM11 2LW', null, null, 'Crew'),
('Rosetta', 'Carter', '66 Felcourt Drive, Bradford', 'BD4 0LJ', null, null, 'Crew'),
('Rene', 'Boyer', '39 Brinklow Road, Binley', 'CV3 2HZ', null, null, 'Crew'),
('Pete', 'Harris', '6 Mariners Terrace, Bosham', 'PO18 8JA', null, null, 'Crew');
```

Figure 5

	ID	FirstName	LastName	HomeAddress	HomeZipcode	OfficeAddress	OfficeZipcode	Position
1	1	Sung	Frazier	Flats 1-17, 37 - 39 Marine Parade, Worthing	BN11 3PH	Paddock View, Netherfield Lane, Stanstead Abbotts	SG12 8HD	Clerk
2	2	Maximo	Ibarra	12 Cumberland Avenue, Bedlington	NE22 6DH	Paddock View, Netherfield Lane, Stanstead Abbotts	SG12 8HD	Clerk
3	3	Newton	Combs	440 Stockport Road West, Bredbury	SK6 2EE	1 Mill Cottages, Mill Road, Long Stratton	NR15 2RT	Clerk
4	4	Brant	Irwin	Somerton House, Somerton Rise, Boothby Graffoe	LN5 0LS	1 Mill Cottages, Mill Road, Long Stratton	NR15 2RT	Clerk
5	5	Cameron	Pope	9 Lord Rosebery Mews, Norwich	NR7 0GX	Doxford Newhouses Farm, Doxford	NE67 5EA	Clerk
6	6	Olga	Gates	28 Whalley Road, Clitheroe	BB7 1AW	Doxford Newhouses Farm, Doxford	NE67 5EA	Clerk
7	7	Hilario	Cordova	8 College Way, Nottingham	NG8 4JH	67A Devonport Road, Plymouth	PL3 4DL	Clerk
8	8	Eric	Lutz	Brook Hollow, Heath Lane, Gunthorpe	NR24 2NT	67A Devonport Road, Plymouth	PL3 4DL	Clerk
9	9	Raquel	Kelley	Flat 2, Keynes Court, 593 Ringwood Road, Ferndown	BH22 9AJ	170 Byron Road, Southampton	SO19 6QW	Clerk
10	10	Latasha	Mayer	92 Park Road, Chesterfield	S40 2JX	170 Byron Road, Southampton	SO19 6QW	Clerk
11	11	Keneth	Patel	11A Quebec Gardens, Blackwater	GU17 9DE	NULL	NULL	Crew
12	12	Rosetta	Barrera	9 Moorhouses, Hightown	L38 9ER	NULL	NULL	Crew
13	13	Jared	Hudson	16 Pen Y Dre, Llanrwst	LL26 0BG	NULL	NULL	Crew
14	14	Gordon	Lloyd	Greenlands, 93 Skipton Road, Silsden	BD20 9DA	NULL	NULL	Crew
15	15	Carla	Barajas	Hillcrest, Birmingham Road, Kingshurst	B37 6RB	NULL	NULL	Crew
16	16	Kate	Daugherty	White Gates, Mayflower Close, Bawtry	DN10 6NG	NULL	NULL	Crew
17	17	Kathleen	Dixon	8 Whitsand Bay Holiday Park, Millbrook	PL10 1JZ	NULL	NULL	Crew
18	18	Lula	Wolf	5 Dennis Road, Padstow	PL28 8DD	NULL	NULL	Crew
19	19	Dave	Nichols	2 Summerhill Gardens, Market Drayton	TF9 1BG	NULL	NULL	Crew
20	20	Demarcus	Chavez	1 Swaledale, Sunderland	SR6 8AH	NULL	NULL	Crew
21	21	Quentin	Gill	3 Berrymoor Road, Banbury	OX16 9QD	NULL	NULL	Crew
22	22	Jimmy	Kelly	14 Northerwood Avenue, Lyndhurst	SO43 7DU	NULL	NULL	Crew
23	23	Elias	Mendez	Palmer House, Berrier	CA11 0XD	NULL	NULL	Crew
24	24	Cheryl	Wong	2 Chapel Street, Caistor	LN7 6UF	NULL	NULL	Crew
25	25	Kayla	Osborne	3 Linforth Way, Coleshill	B46 3LH	NULL	NULL	Crew
26	26	Sonya	Black	88 The Oval, Leeds	LS14 6BB	NULL	NULL	Crew
27	27	Elliott	Henders...	14 Ardleigh Green Road, Hornchurch	RM11 2LW	NULL	NULL	Crew
28	28	Rosetta	Carter	66 Felcourt Drive, Bradford	BD4 0LJ	NULL	NULL	Crew
29	29	Rene	Boyer	39 Brinklow Road, Binley	CV3 2HZ	NULL	NULL	Crew
30	30	Pete	Harris	6 Mariners Terrace, Bosham	PO18 8JA	NULL	NULL	Crew

Figure 6

```
INSERT INTO Seaport(Code, City, Country, PhoneNumber)
VALUES
('LDN', 'London', 'United Kingdom', '+44 7457 850959'),
('SMO', 'Saint-Malo', 'France', '+33 6 40 67 00 06'),
('PAN', 'Pont-Aven', 'France', '+33 7 00 18 38 85'),
('BDX', 'Bordeaux', 'France', '+33 7 00 05 09 74'),
('BBO', 'Bilbao', 'Spain', '+34 721 21 86 18'),
('LCA', 'La Coruna', 'Spain', '+34 590 60 07 92'),
('ORO', 'Oporto', 'Portugal', '+351 639 367 973'),
('LBN', 'Lisbon', 'Portugal', '+351 639 306 175'),
('SVE', 'Seville', 'Spain', '+34 695 64 17 37'),
('CDZ', 'Cadiz', 'Spain', '+34 973 90 72 03'),
('BCA', 'Barcelona', 'Spain', '+34 973 90 40 20'),
('TVT', 'Tivat', 'Montenegro', '+382 60 775 241'),
('SDP', 'Santa Cruz De La Palma', 'Spain', '+34 673 92 09 86'),
('SNS', 'Sines', 'Portugal', '+351 659 230 848'),
('CTA', 'Constanta', 'Romania', '+40 702 081 640'),
('CUA', 'Ceuta', 'Spain', '+34 973 90 57 61'),
('STI', 'Stratoni', 'Greece', '+30 689 487 8551'),
('CVA', 'Civitavecchia', 'Italy', '+39 349 325 4928'),
('ATN', 'Ardalstangen', 'Norway', '+47 481 44 187'),
('RNK', 'Runavik', 'Faroe Islands', '+298 517228');
```

Figure 7

	Code	City	Country	PhoneNumber
1	ATN	Ardalstangen	Norway	+47 481 44 187
2	BBO	Bilbao	Spain	+34 721 21 86 18
3	BCA	Barcelona	Spain	+34 973 90 40 20
4	BDX	Bordeaux	France	+33 7 00 05 09 74
5	CDZ	Cadiz	Spain	+34 973 90 72 03
6	CTA	Constanta	Romania	+40 702 081 640
7	CUA	Ceuta	Spain	+34 973 90 57 61
8	CVA	Civitavecchia	Italy	+39 349 325 4928
9	LBN	Lisbon	Portugal	+351 639 306 175
10	LCA	La Coruna	Spain	+34 590 60 07 92
11	LDN	London	United Kingdom	+44 7457 850959
12	ORO	Oporto	Portugal	+351 639 367 973
13	PAN	Pont-Aven	France	+33 7 00 18 38 85
14	RNK	Runavik	Faroe Islands	+298 517228
15	SDP	Santa Cruz De La Palma	Spain	+34 673 92 09 86
16	SMO	Saint-Malo	France	+33 6 40 67 00 06
17	SNS	Sines	Portugal	+351 659 230 848
18	STI	Stratoni	Greece	+30 689 487 8551
19	SVE	Seville	Spain	+34 695 64 17 37
20	TVT	Tivat	Montenegro	+382 60 775 241

Figure 8

```
INSERT INTO Meal([Type], Price)
VALUES
('No food', 0),
('Vegitarian', 20),
('Vegan', 25),
('Halal', 20),
('Kosher', 35),
('Gluten-free', 100),
('Pescetarian', 50),
('Diabetic', 45),
('Lactose-free', 55),
('Remba', 75),
('Intermittent fasting', 25),
('Atkins', 15),
('Dukan', 35),
('Monotrophic', 32),
('Ketogenic', 41),
('Low-FODMAP', 27),
('Pegan', 29),
('Pollotarian', 55),
('Climatarian', 43),
('Macrobiotic', 37);
```

Figure 9

	Type	Price
1	Atkins	15.00
2	Climatarian	43.00
3	Diabetic	45.00
4	Dukan	35.00
5	Gluten-free	100.00
6	Halal	20.00
7	Intermittent fasting	25.00
8	Ketogenic	41.00
9	Kosher	35.00
10	Lactose-free	55.00
11	Low-FODMAP	27.00
12	Macrobiotic	37.00
13	Monotrophic	32.00
14	No food	0.00
15	Pegan	29.00
16	Pescetarian	50.00
17	Pollotarian	55.00
18	Remba	75.00
19	Vegan	25.00
20	Vegitarian	20.00

Figure 10

```
INSERT INTO Activities([Type], [Location], [Time], Price)
VALUES
('Cats', ' Tchaikovsky Hall', '8:00:00', 50),
('The Wizard of Oz', ' Tchaikovsky Hall', '10:00:00', 50),
('Oliver!', ' Tchaikovsky Hall', '12:00:00', 50),
('West Side Story', ' Tchaikovsky Hall', '14:00:00', 50),
('The Sound of Music', ' Tchaikovsky Hall', '16:00:00', 50),
('The Phantom of the Opera', ' Tchaikovsky Hall', '18:00:00', 50),
('Les Miserables', ' Tchaikovsky Hall', '20:00:00', 50),
('Cabaret', ' Tchaikovsky Hall', '22:00:00', 50),
('Hip Hop Music', ' Music Hall', '12:00:00', 0),
('Contemporary Music', ' Music Hall', '14:00:00', 0),
('Classical Music', ' Music Hall', '6:00:00', 0),
('Pop Music', ' Music Hall', '18:00:00', 0),
('Rock Music', ' Music Hall', '20:00:00', 0),
('Disco Music', ' Music Hall', '22:00:00', 0),
('Jogging', ' Main Deck', '6:00:00', 0),
('Swimming', ' Swimming Pool', '8:00:00', 10),
('Yoga', ' Main Deck', '10:00:00', 20),
('Pilates', ' Main Deck', '12:00:00', 20),
('Tai Chi', ' Main Deck', '14:00:00', 20),
('Walking', ' Main Deck', '16:00:00', 0);
```

Figure 11

	Type	Location	Time	Price
1	Cabaret	Tchaikovsky Hall	22:00:00.0000000	50.00
2	Cats	Tchaikovsky Hall	08:00:00.0000000	50.00
3	Classical Music	Music Hall	06:00:00.0000000	0.00
4	Contemporary Music	Music Hall	14:00:00.0000000	0.00
5	Disco Music	Music Hall	22:00:00.0000000	0.00
6	Hip Hop Music	Music Hall	12:00:00.0000000	0.00
7	Jogging	Main Deck	06:00:00.0000000	0.00
8	Les Miserables	Tchaikovsky Hall	20:00:00.0000000	50.00
9	Oliver!	Tchaikovsky Hall	12:00:00.0000000	50.00
10	Pilates	Main Deck	12:00:00.0000000	20.00
11	Pop Music	Music Hall	18:00:00.0000000	0.00
12	Rock Music	Music Hall	20:00:00.0000000	0.00
13	Swimming	Swimming Pool	08:00:00.0000000	10.00
14	Tai Chi	Main Deck	14:00:00.0000000	20.00
15	The Phantom of the Opera	Tchaikovsky Hall	18:00:00.0000000	50.00
16	The Sound of Music	Tchaikovsky Hall	16:00:00.0000000	50.00
17	The Wizard of Oz	Tchaikovsky Hall	10:00:00.0000000	50.00
18	Walking	Main Deck	16:00:00.0000000	0.00
19	West Side Story	Tchaikovsky Hall	14:00:00.0000000	50.00
20	Yoga	Main Deck	10:00:00.0000000	20.00

Figure 12

```
INSERT INTO CruiseLinerEmployees(EmployeesID, SeaportCode)
VALUES
(11, 'ATN'),
(12, 'BBO'),
(13, 'BCA'),
(14, 'BDX'),
(15, 'CDZ'),
(16, 'CTA'),
(17, 'CUA'),
(18, 'CVA'),
(19, 'LBN'),
(20, 'LCA'),
(21, 'LDN'),
(22, 'ORO'),
(23, 'PAN'),
(24, 'RNK'),
(25, 'SDP'),
(26, 'SMO'),
(27, 'SNS'),
(28, 'STI'),
(29, 'SVE'),
(30, 'TVT');
```

Figure 13

	EmployeesID	SeaportCode
1	11	ATN
2	12	BBO
3	13	BCA
4	14	BDX
5	15	CDZ
6	16	CTA
7	17	CUA
8	18	CVA
9	19	LBN
10	20	LCA
11	21	LDN
12	22	ORO
13	23	PAN
14	24	RNK
15	25	SDP
16	26	SMO
17	27	SNS
18	28	STI
19	29	SVE
20	30	TVT

Figure 14

```
INSERT INTO Reservation(PassengerFirstName, PassengerLastName, AmountToPay, Status, CustomerID, EmployeesID)
VALUES
('Spencer', 'Vazquez', 1000, 'complete', 1, 1),
('Porfirio', 'Jordan', 2000, 'incomplete', 2, 2),
('Autumn', 'Flores', 2500, 'cancelled', 3, 3),
('Michelle', 'Torres', 3000, 'complete', 4, 4),
('Hong', 'Berg', 1500, 'incomplete', 5, 5),
('Alec', 'Velasquez', 1200, 'cancelled', 6, 1),
('Norma', 'Cochran', 1600, 'complete', 7, 2),
('Jaclyn', 'Blevins', 1400, 'incomplete', 8, 3),
('Philip', 'Prince', 1700, 'cancelled', 9, 4),
('Fausto', 'Case', 2100, 'complete', 10, 5),
('Orval', 'Garrison', 2500, 'incomplete', 11, 2),
('Maura', 'Anthony', 2600, 'cancelled', 12, 3),
('Arthur', 'Hester', 1000, 'complete', 13, 4),
('Allison', 'Tapia', 1200, 'incomplete', 14, 5),
('Victor', 'Oconnell', 2400, 'cancelled', 15, 3),
('Jerrold', 'Hayden', 2300, 'complete', 16, 4),
('Rico', 'Ibarra', 3200, 'incomplete', 17, 5),
('Dale', 'Herman', 3100, 'cancelled', 18, 4),
('Mari', 'Bailey', 2400, 'complete', 19, 5),
('Britt', 'Pittman', 2600, 'incomplete', 20, 5);
```

Figure 15

	ID	PassengerFirstName	PassengerLastName	AmountToPay	Status	CustomerID	EmployeesID
1	1	Spencer	Vazquez	1000.00	complete	1	1
2	2	Porfirio	Jordan	2000.00	incomplete	2	2
3	3	Autumn	Flores	2500.00	cancelled	3	3
4	4	Michelle	Torres	3000.00	complete	4	4
5	5	Hong	Berg	1500.00	incomplete	5	5
6	6	Alec	Velasquez	1200.00	cancelled	6	1
7	7	Norma	Cochran	1600.00	complete	7	2
8	8	Jaclyn	Blevins	1400.00	incomplete	8	3
9	9	Philip	Prince	1700.00	cancelled	9	4
10	10	Fausto	Case	2100.00	complete	10	5
11	11	Orval	Garrison	2500.00	incomplete	11	2
12	12	Maura	Anthony	2600.00	cancelled	12	3
13	13	Arthur	Hester	1000.00	complete	13	4
14	14	Allison	Tapia	1200.00	incomplete	14	5
15	15	Victor	Oconnell	2400.00	cancelled	15	3
16	16	Jerrold	Hayden	2300.00	complete	16	4
17	17	Rico	Ibarra	3200.00	incomplete	17	5
18	18	Dale	Herman	3100.00	cancelled	18	4
19	19	Mari	Bailey	2400.00	complete	19	5
20	20	Britt	Pittman	2600.00	incomplete	20	5

Figure 16

```
INSERT INTO Ship([Name], [Type], NumberOfCabins, FullDisplacement, FuelCapacity, Speed, SeaportCode)
VALUES
('Apollo', 'F70', 87, 169000, 7450000, 18000, 'ATN'),
('Abyss', 'F80', 97, 225000, 8450000, 17000, 'BBO'),
('Atlantis', 'F90', 107, 228000, 9450000, 16000, 'BCA'),
('Calypso', 'F50', 67, 165000, 5450000, 20000, 'BDX'),
('Encore', 'F60', 77, 168000, 6450000, 19000, 'CDZ'),
('Eclipse', 'F70', 87, 169000, 7450000, 18000, 'CTA'),
('Fantasia', 'F80', 97, 225000, 8450000, 17000, 'CUA'),
('Gemini', 'F90', 107, 228000, 9450000, 16000, 'CVA'),
('Jubilee', 'F50', 67, 165000, 5450000, 20000, 'LBN'),
('Liberty', 'F60', 77, 168000, 6450000, 19000, 'LCA'),
('Luna', 'F70', 87, 169000, 7450000, 18000, 'LDN'),
('Grand Bleu', 'F80', 97, 225000, 8450000, 17000, 'ORO'),
('Maverick', 'F90', 107, 228000, 9450000, 16000, 'PAN'),
('North Star', 'F50', 67, 165000, 5450000, 20000, 'RNK'),
('Neptune', 'F60', 77, 168000, 6450000, 19000, 'SDP'),
('Orion', 'F70', 87, 169000, 7450000, 18000, 'SMO'),
('Odysse', 'F80', 97, 225000, 8450000, 17000, 'SNS'),
('Pegasus', 'F90', 107, 228000, 9450000, 16000, 'STI'),
('Silver Lining', 'F50', 67, 165000, 5450000, 20000, 'SVE'),
('Ultra Violet', 'F60', 77, 168000, 6450000, 19000, 'TVT');
```

Figure 17

	Name	Type	NumberOfCabins	FullDisplacement	FuelCapacity	Speed	SeaportCode
1	Abyss	F80	97	225000.00	8450000.00	17000.00	BBO
2	Apollo	F70	87	169000.00	7450000.00	18000.00	ATN
3	Atlantis	F90	107	228000.00	9450000.00	16000.00	BCA
4	Calypso	F50	67	165000.00	5450000.00	20000.00	BDX
5	Eclipse	F70	87	169000.00	7450000.00	18000.00	CTA
6	Encore	F60	77	168000.00	6450000.00	19000.00	CDZ
7	Fantasia	F80	97	225000.00	8450000.00	17000.00	CUA
8	Gemini	F90	107	228000.00	9450000.00	16000.00	CVA
9	Grand Bleu	F80	97	225000.00	8450000.00	17000.00	ORO
10	Jubilee	F50	67	165000.00	5450000.00	20000.00	LBN
11	Liberty	F60	77	168000.00	6450000.00	19000.00	LCA
12	Luna	F70	87	169000.00	7450000.00	18000.00	LDN
13	Maverick	F90	107	228000.00	9450000.00	16000.00	PAN
14	Neptune	F60	77	168000.00	6450000.00	19000.00	SDP
15	North Star	F50	67	165000.00	5450000.00	20000.00	RNK
16	Odysse	F80	97	225000.00	8450000.00	17000.00	SNS
17	Orion	F70	87	169000.00	7450000.00	18000.00	SMO
18	Pegasus	F90	107	228000.00	9450000.00	16000.00	STI
19	Silver Lining	F50	67	165000.00	5450000.00	20000.00	SVE
20	Ultra Violet	F60	77	168000.00	6450000.00	19000.00	TVT

Figure 18

```
INSERT INTO Voyage(ShipName, DestinationSeaport, DateOfDeparture, DateOfArrival, TravelTime)
VALUES
('Apollo', 'TVT', '2021-05-11', '2021-05-16', 5),
('Abyss', 'ATN', '2021-05-12', '2021-05-18', 6),
('Atlantis', 'BBO', '2021-05-13', '2021-05-20', 7),
('Calypso', 'BCA', '2021-05-14', '2021-05-22', 8),
('Encore', 'BDX', '2021-05-15', '2021-05-20', 5),
('Eclipse', 'CDZ', '2021-05-16', '2021-05-22', 6),
('Fantasia', 'CTA', '2021-05-17', '2021-05-24', 7),
('Gemini', 'CUA', '2021-05-18', '2021-05-26', 8),
('Jubilee', 'CVA', '2021-05-19', '2021-05-24', 5),
('Liberty', 'LBN', '2021-05-20', '2021-05-26', 6),
('Luna', 'LCA', '2021-05-21', '2021-05-28', 7),
('Grand Bleu', 'LDN', '2021-05-22', '2021-05-30', 8),
('Maverick', 'ORO', '2021-05-23', '2021-05-28', 5),
('North Star', 'PAN', '2021-05-24', '2021-05-30', 6),
('Neptune', 'RNK', '2021-05-25', '2021-06-02', 7),
('Orion', 'SDP', '2021-05-26', '2021-06-04', 8),
('Odysse', 'SMO', '2021-05-27', '2021-06-02', 5),
('Pegasus', 'SNS', '2021-05-28', '2021-06-04', 6),
('Silver Lining', 'STI', '2021-05-29', '2021-06-06', 7),
('Ultra Violet', 'SVE', '2021-05-30', '2021-06-08', 8);
```

Figure 19

	Number	ShipName	DestinationSeaport	DateOfDeparture	DateOfArrival	TravelTime
1	1	Apollo	TVT	2021-05-11	2021-05-16	5
2	2	Abyss	ATN	2021-05-12	2021-05-18	6
3	3	Atlantis	BBO	2021-05-13	2021-05-20	7
4	4	Calypso	BCA	2021-05-14	2021-05-22	8
5	5	Encore	BDX	2021-05-15	2021-05-20	5
6	6	Eclipse	CDZ	2021-05-16	2021-05-22	6
7	7	Fantasia	CTA	2021-05-17	2021-05-24	7
8	8	Gemini	CUA	2021-05-18	2021-05-26	8
9	9	Jubilee	CVA	2021-05-19	2021-05-24	5
10	10	Liberty	LBN	2021-05-20	2021-05-26	6
11	11	Luna	LCA	2021-05-21	2021-05-28	7
12	12	Grand Bleu	LDN	2021-05-22	2021-05-30	8
13	13	Maverick	ORO	2021-05-23	2021-05-28	5
14	14	North Star	PAN	2021-05-24	2021-05-30	6
15	15	Neptune	RNK	2021-05-25	2021-06-02	7
16	16	Orion	SDP	2021-05-26	2021-06-04	8
17	17	Odysse	SMO	2021-05-27	2021-06-02	5
18	18	Pegasus	SNS	2021-05-28	2021-06-04	6
19	19	Silver Lini...	STI	2021-05-29	2021-06-06	7
20	20	Ultra Violet	SVE	2021-05-30	2021-06-08	8

Figure 20

```
INSERT INTO Cabins(VoyageNumber, CabinNumber, [Type], Price, Available)
VALUES
(1, 106, 'Suite', 300, 0),
(1, 112, 'Junior suite', 200, 1),
(2, 112, 'Junior suite', 200, 0),
(2, 106, 'Suite', 300, 1),
(3, 202, 'Double', 100, 0),
(3, 136, 'Single', 50, 1),
(4, 136, 'Single', 50, 0),
(4, 202, 'Double', 100, 1),
(5, 321, 'Triple', 150, 0),
(5, 326, 'Triple', 150, 1),
(6, 106, 'Suite', 300, 0),
(6, 112, 'Junior suite', 200, 1),
(7, 112, 'Junior suite', 200, 0),
(7, 106, 'Suite', 300, 1),
(8, 202, 'Double', 100, 0),
(8, 136, 'Single', 50, 1),
(9, 136, 'Single', 50, 0),
(9, 202, 'Double', 100, 1),
(10, 321, 'Triple', 150, 0),
(10, 326, 'Triple', 150, 1),
(1, 107, 'Suite', 300, 0),
(2, 113, 'Junior suite', 200, 0),
(3, 310, 'Double', 100, 0),
(4, 240, 'Single', 50, 0),
(5, 334, 'Triple', 150, 0),
(6, 107, 'Suite', 300, 0),
(7, 113, 'Junior suite', 200, 0),
(8, 310, 'Double', 100, 0),
(9, 240, 'Single', 50, 0),
(10, 334, 'Triple', 150, 0);
```

Figure 21

	VoyageNumber	CabinNumber	Type	Price	Available
1	1	106	Suite	300.00	0
2	1	107	Suite	300.00	0
3	1	112	Junior suite	200.00	1
4	2	106	Suite	300.00	1
5	2	112	Junior suite	200.00	0
6	2	113	Junior suite	200.00	0
7	3	136	Single	50.00	1
8	3	202	Double	100.00	0
9	3	310	Double	100.00	0
10	4	136	Single	50.00	0
11	4	202	Double	100.00	1
12	4	240	Single	50.00	0
13	5	321	Triple	150.00	0
14	5	326	Triple	150.00	1
15	5	334	Triple	150.00	0
16	6	106	Suite	300.00	0
17	6	107	Suite	300.00	0
18	6	112	Junior suite	200.00	1
19	7	106	Suite	300.00	1
20	7	112	Junior suite	200.00	0
21	7	113	Junior suite	200.00	0
22	8	136	Single	50.00	1
23	8	202	Double	100.00	0
24	8	310	Double	100.00	0
25	9	136	Single	50.00	0
26	9	202	Double	100.00	1
27	9	240	Single	50.00	0
28	10	321	Triple	150.00	0
29	10	326	Triple	150.00	1
30	10	334	Triple	150.00	0

Figure 22

```
INSERT INTO Ticket(ReservationID, CrewID, VoyageNumber, CabinNumber, MealType, TravelType)
VALUES
(1, null, 1, 106, 'No food', 'Departure'),
(2, null, 1, 107, 'No food', 'Departure'),
(3, null, 2, 112, 'Vegitarian', 'Return'),
(4, null, 2, 113, 'Vegitarian', 'Return'),
(5, null, 3, 202, 'Vegan', 'Departure'),
(6, null, 3, 310, 'Vegan', 'Departure'),
(7, null, 4, 136, 'Halal', 'Return'),
(8, null, 4, 240, 'Halal', 'Return'),
(9, null, 5, 321, 'Kosher', 'Departure'),
(10, null, 5, 334, 'Kosher', 'Departure'),
(null, 11, 6, 106, 'Gluten-free', 'Return'),
(null, 12, 6, 107, 'Gluten-free', 'Return'),
(null, 13, 7, 112, 'Pescetarian', 'Departure'),
(null, 14, 7, 113, 'Pescetarian', 'Departure'),
(null, 15, 8, 202, 'Diabetic', 'Return'),
(null, 16, 8, 310, 'Diabetic', 'Return'),
(null, 17, 9, 136, 'Lactose-free', 'Departure'),
(null, 18, 9, 240, 'Lactose-free', 'Departure'),
(null, 19, 10, 321, 'Remba', 'Return'),
(null, 20, 10, 334, 'Remba', 'Return');
```

Figure 23

	ID	ReservationID	CrewID	VoyageNumber	CabinNumber	MealType	TravelType
1	1	1	NULL	1	106	No food	Departure
2	2	2	NULL	1	107	No food	Departure
3	3	3	NULL	2	112	Vegitarian	Return
4	4	4	NULL	2	113	Vegitarian	Return
5	5	5	NULL	3	202	Vegan	Departure
6	6	6	NULL	3	310	Vegan	Departure
7	7	7	NULL	4	136	Halal	Return
8	8	8	NULL	4	240	Halal	Return
9	9	9	NULL	5	321	Kosher	Departure
10	10	10	NULL	5	334	Kosher	Departure
11	11	NULL	11	6	106	Gluten-f...	Return
12	12	NULL	12	6	107	Gluten-f...	Return
13	13	NULL	13	7	112	Pesceta...	Departure
14	14	NULL	14	7	113	Pesceta...	Departure
15	15	NULL	15	8	202	Diabetic	Return
16	16	NULL	16	8	310	Diabetic	Return
17	17	NULL	17	9	136	Lactose...	Departure
18	18	NULL	18	9	240	Lactose...	Departure
19	19	NULL	19	10	321	Remba	Return
20	20	NULL	20	10	334	Remba	Return

Figure 24

```
INSERT INTO Ticket_Activities(TicketID, ActivityType)
VALUES
(1, 'Cats'),
(2, 'The Wizard of Oz'),
(3, 'Oliver!'),
(4, 'West Side Story'),
(5, 'The Sound of Music'),
(6, 'The Phantom of the Opera'),
(7, 'Les Miserables'),
(8, 'Cabaret'),
(9, 'Hip Hop Music'),
(10, 'Contemporary Music'),
(1, 'Classical Music'),
(2, 'Pop Music'),
(3, 'Rock Music'),
(4, 'Disco Music'),
(5, 'Jogging'),
(6, 'Swimming'),
(7, 'Yoga'),
(8, 'Pilates'),
(9, 'Tai Chi'),
(10, 'Walking');
```

Figure 25

	TicketID	ActivityType
1	1	Cats
2	2	The Wizard of Oz
3	3	Oliver!
4	4	West Side Story
5	5	The Sound of Music
6	6	The Phantom of the Opera
7	7	Les Miserables
8	8	Cabaret
9	9	Hip Hop Music
10	10	Contemporary Music
11	1	Classical Music
12	2	Pop Music
13	3	Rock Music
14	4	Disco Music
15	5	Jogging
16	6	Swimming
17	7	Yoga
18	8	Pilates
19	9	Tai Chi
20	10	Walking

Figure 26

(Random Word Generator, 2021) and (Bell, 2000) were used when inserting test data into the Customer and Employees tables for addresses and names. (Map Developers, 2021), (Ports.com, 2010) and (Randommer, 2021) were used to insert test data for countries, seaports and phone numbers related to each seaport into the Seaport table. (Wikipedia, 2021) was used to determine the dietary constraints inserted into the Meal table. (ChrisWalczyk55, 2017) was used to determine the type of

activities to insert into the Activities table while (Khan, 2016) was used as reference when inserting time data alongside activities where locations for each activity were gathered from the example layout of a cruise ship in the specification. (Chilton, 2019) was used to determine the names of each boat while (CruiseMapper, 2015) was used to estimate the cruising speed of each ship. (S., 2021) was used to estimate a ships' full displacement while (Tell Me How Much, 2021) was used to estimate a ships fuel capacity where the ship type was determined from the example reservation in the specification. (Tutorialspoint, 2021) was consulted when inserting information related to dates for the Voyage table.

Although all of the tables should have 20 records, some of the tables have 30 records as after creating the tables, it felt necessary to insert more records into the Cabins table to depict that multiple records can be inserted into the table as long as the VoyageNumber was different. Additionally, more records were inserted into the Employees table to accommodate for the CruiseLinerEmployees table which would contain employees with a Position attribute returning that they are a crew member. After discovering that some data types were incorrect during the creation of the database, (Gavin, 2021) was used to delete all the tables in the database so that we could remedy any errors found due to the design of the database such as foreign key relationships not being implemented correctly.

SQL queries

Query 1

```
1 USE Cruise_Liner;
2 GO
3
4 -- for seeing only the highest and lowest reservations by employee
5 DROP VIEW IF EXISTS query1;
6 GO
7
8 CREATE VIEW query1 as(
9 SELECT FirstName, LastName, OfficeAddress, OfficeZipcode,
COUNT(EmployeesID) as NumberOfReservations
10FROM Employees, Reservation
11WHERE Employees.ID = Reservation.EmployeesID
12GROUP BY FirstName, LastName, OfficeAddress, OfficeZipcode);
13GO
14
15SELECT *
16FROM query1
17WHERE query1.NumberOfReservations=(SELECT MIN(NumberOfReservations) FROM
query1) OR query1.NumberOfReservations=(SELECT MAX(NumberOfReservations)
FROM query1);
18
19-- to see all employees ordered by number of reservations
20SELECT FirstName, LastName, OfficeAddress, OfficeZipcode,
COUNT(EmployeesID) as NumberOfReservations
21FROM Employees, Reservation
22WHERE Employees.ID = Reservation.EmployeesID
23GROUP BY FirstName, LastName, OfficeAddress, OfficeZipcode
24ORDER BY NumberOfReservations DESC;
```

Figure 27

Although a traditional subquery as depicted in (Diwan, 2019) could have been used to retrieve the employees with the highest and lowest number of reservations, an issue was encountered where the number of reservations for each employee wasn't being recognised from within the subquery thus, (Gheorghiu, 2014) was used to store the results of querying the Employees and Reservations table to then retrieve the employees with the highest and lowest number of reservations as depicted in lines 8 to 17. (w3resource.com, 2020) was used to count the number of reservations made by each employee.

Executing the query returns Figure 28 assuming that we want to return the employees with the highest and lowest number of reservations.

	FirstName	LastName	OfficeAddress	OfficeZipcode	NumberOfReservations
1	Cameron	Pope	Doxford Newhouses Farm, Doxford	NE67 5EA	6
2	Sung	Frazier	Paddock View, Netherfield Lane, Stanstead Abbotts	SG12 8HD	2

Figure 28

If we want to retrieve all of the employees which have made reservations, lines 20 to 24 would be executed where (Refsnes Data, 1999) was used to order the queried data by the number of reservations made by each employee in descending order as seen in Figure 29.

	FirstName	LastName	OfficeAddress	OfficeZipcode	NumberOfReservations
1	Cameron	Pope	Doxford Newhouses Farm, Doxford	NE67 5EA	6
2	Brant	Irwin	1 Mill Cottages, Mill Road, Long Stratton	NR15 2RT	5
3	Newton	Combs	1 Mill Cottages, Mill Road, Long Stratton	NR15 2RT	4
4	Maximo	Ibarra	Paddock View, Netherfield Lane, Stanstead Abbotts	SG12 8HD	3
5	Sung	Frazier	Paddock View, Netherfield Lane, Stanstead Abbotts	SG12 8HD	2

Figure 29

Query 2

```
1  USE Cruise_Liner;
2  GO
3
4  -- if for each status type (as in cancelled, incomplete or complete)
5  SELECT [Status], COUNT([Status]) as TotalPerStatus
6  FROM Employees, Reservation
7  WHERE Employees.ID = Reservation.EmployeesID
8  GROUP BY [Status]
9
10-- if for each employee
11SELECT FirstName, LastName, OfficeAddress, OfficeZipcode, [Status],
COUNT(EmployeesID) as NumberOfReservations
12FROM Employees, Reservation
13WHERE Employees.ID = Reservation.EmployeesID
14GROUP BY FirstName, LastName, OfficeAddress, OfficeZipcode, [Status]
```

Figure 30

A similar mechanism has been used for counting the number of complete, incomplete and cancelled reservations as seen in lines 5 to 8 where, upon execution, we are returned the total amount of complete, incomplete and cancelled reservations as seen in Figure 31.

	Status	TotalPerStatus
1	cancelled	6
2	complete	7
3	incomplete	7

Figure 31

If we want to retrieve the total number of cancelled, incomplete and complete reservations for each employee, we execute lines 11 to 14 as seen in Figure 32.

	FirstName	LastName	OfficeAddress	OfficeZipcode	Status	NumberOfReservations
1	Brant	Irwin	1 Mill Cottages, Mill Road, Long Stratton	NR15 2RT	cancelled	2
2	Brant	Irwin	1 Mill Cottages, Mill Road, Long Stratton	NR15 2RT	complete	3
3	Cameron	Pope	Doxford Newhouses Farm, Doxford	NE67 5EA	complete	2
4	Cameron	Pope	Doxford Newhouses Farm, Doxford	NE67 5EA	incomplete	4
5	Maximo	Ibarra	Paddock View, Netherfield Lane, Stanstead Abbotts	SG12 8HD	complete	1
6	Maximo	Ibarra	Paddock View, Netherfield Lane, Stanstead Abbotts	SG12 8HD	incomplete	2
7	Newton	Combs	1 Mill Cottages, Mill Road, Long Stratton	NR15 2RT	cancelled	3
8	Newton	Combs	1 Mill Cottages, Mill Road, Long Stratton	NR15 2RT	incomplete	1
9	Sung	Frazier	Paddock View, Netherfield Lane, Stanstead Abbotts	SG12 8HD	cancelled	1
10	Sung	Frazier	Paddock View, Netherfield Lane, Stanstead Abbotts	SG12 8HD	complete	1

Figure 32

Query 3

```
1 USE Cruise_Liner;
2 GO
3
4 -- See all cabins and their availability
5 SELECT Cabins.VoyageNumber, Cabins.CabinNumber, Cabins.Type,
Cabins.Price, CASE WHEN Cabins.Available = 1 THEN 'Available'
6 ELSE 'Unavailable'
7 END as Availability
8 FROM Cabins, Voyage
9 WHERE Cabins.VoyageNumber = Voyage.Number;
10
11-- See only available cabins
12SELECT Cabins.VoyageNumber, Cabins.CabinNumber, Cabins.Type,
Cabins.Price, CASE WHEN Cabins.Available = 1 THEN 'Available'
13ELSE 'Unavailable'
14END as Availability
15FROM Cabins, Voyage
16WHERE Cabins.VoyageNumber = Voyage.Number AND Cabins.Available = 1;
```

Figure 33

(Refsnes Data, 1999) was used to convert the bit values stored in the Available attribute to contextual terms to dictate whether a cabin on a voyage is available where, if lines 5 to 9 are executed all statuses of all cabins will be retrieved as seen in Figure 34.

If we only want to retrieve cabins on voyages that are available, lines 12 to 16 would return the values depicted in Figure 35.

	VoyageNumber	CabinNumber	Type	Price	Availability
1	1	106	Suite	300.00	Unavailable
2	1	107	Suite	300.00	Unavailable
3	1	112	Junior suite	200.00	Available
4	2	106	Suite	300.00	Available
5	2	112	Junior suite	200.00	Unavailable
6	2	113	Junior suite	200.00	Unavailable
7	3	136	Single	50.00	Available
8	3	202	Double	100.00	Unavailable
9	3	310	Double	100.00	Unavailable
10	4	136	Single	50.00	Unavailable
11	4	202	Double	100.00	Available
12	4	240	Single	50.00	Unavailable
13	5	321	Triple	150.00	Unavailable
14	5	326	Triple	150.00	Available
15	5	334	Triple	150.00	Unavailable
16	6	106	Suite	300.00	Unavailable
17	6	107	Suite	300.00	Unavailable
18	6	112	Junior suite	200.00	Available
19	7	106	Suite	300.00	Available
20	7	112	Junior suite	200.00	Unavailable
21	7	113	Junior suite	200.00	Unavailable
22	8	136	Single	50.00	Available
23	8	202	Double	100.00	Unavailable
24	8	310	Double	100.00	Unavailable
25	9	136	Single	50.00	Unavailable
26	9	202	Double	100.00	Available
27	9	240	Single	50.00	Unavailable
28	10	321	Triple	150.00	Unavailable
29	10	326	Triple	150.00	Available
30	10	334	Triple	150.00	Unavailable

Figure 34

	VoyageNumber	CabinNumber	Type	Price	Availability
1	1	112	Junior suite	200.00	Available
2	2	106	Suite	300.00	Available
3	3	136	Single	50.00	Available
4	4	202	Double	100.00	Available
5	5	326	Triple	150.00	Available
6	6	112	Junior suite	200.00	Available
7	7	106	Suite	300.00	Available
8	8	136	Single	50.00	Available
9	9	202	Double	100.00	Available
10	10	326	Triple	150.00	Available

Figure 35

Query 4

```
1 USE Cruise_Liner;
2 GO
3
4 -- Should check availability of Cabin via Cabin Number and Voyage Number
to check if the cabin on that voyage is available
5 DROP PROCEDURE IF EXISTS Check_Availability;
6 GO
7
8 CREATE PROCEDURE Check_Availability
9   @VoyageNumber   Int,
10  @CabinNumber    Int
11AS
12
13BEGIN
14  DECLARE @rowCount AS Int;
15  DECLARE @availability AS Bit;
16
17  SELECT @rowCount = COUNT(*)
18  FROM Cabins
19  WHERE Cabins.CabinNumber = @CabinNumber AND Cabins.VoyageNumber =
@VoyageNumber;
20
21  IF @rowCount = 0
22      BEGIN
23          PRINT 'Cabin number: ' + CONVERT(Char(12), @CabinNumber) + ' on
Voyage number: ' + CONVERT(Char(12),@VoyageNumber) + ' doesnt exist in the
cabins table'
24          RETURN;
25      END;
26
27  BEGIN
28      SELECT @availability = Cabins.Available
29      FROM Cabins, Voyage
30      WHERE Cabins.VoyageNumber = Voyage.Number AND Cabins.CabinNumber =
@CabinNumber AND Cabins.VoyageNumber = @VoyageNumber;
31
32      IF @availability = 1
33          BEGIN
34              PRINT 'Cabin number: ' + CONVERT(Char(12), @CabinNumber) +
' on Voyage number: ' + CONVERT(Char(12),@VoyageNumber) + ' is available'
35              RETURN;
36          END;
37      ELSE
38          BEGIN
39              PRINT 'Cabin number: ' + CONVERT(Char(12), @CabinNumber) +
' on Voyage number: ' + CONVERT(Char(12),@VoyageNumber) + ' is unavailable'
40              RETURN;
41          END;
42  END;
43END;
44
45EXEC Check_Availability @VoyageNumber=1, @CabinNumber=200;
```

Figure 36

A stored procedure has been implemented to check for the availability of a cabin on a specified voyage assuming that both the voyage and cabin number exist in the Cabins table. Lines 21 to 25 check to see if the cabin and voyage are present in the cabins table to which, if they aren't a message is returned to depict that both the cabin and voyage aren't in the table. If we provide a cabin number

of 1000 and a voyage number of 1, although the voyage number exists as the cabin number doesn't exist, we are returned that the cabin on that voyage doesn't exist as seen in Figure 37.

```
Cabin number: 1000          on Voyage number: 1            doesnt exist in the cabins table

Completion time: 2021-04-28T16:37:22.1492886+01:00
```

Figure 37

If we assume that the user enters a cabin and voyage number which is present in the Cabins table, the query will return that either the cabin is or isn't available. If we provide a cabin number of 112 and voyage number of 1, we are returned that the cabin is available as seen in Figure 38. However, if we provide a cabin number of 136 and voyage number of 4 the procedure returns that the cabin is unavailable as seen in Figure 39.

```
Cabin number: 112           on Voyage number: 1            is available

Completion time: 2021-04-28T16:39:59.5423989+01:00
```

Figure 38

```
Cabin number: 136           on Voyage number: 4            is unavailable

Completion time: 2021-04-28T16:42:43.3239352+01:00
```

Figure 39

Although we could have automated the execution of the stored procedure through the use of triggers, as depicted in (Pedamkar, 2020), before a record is inserted into the Reservation table however we are unsure of how to capture the cabin and voyage number entered into the insert statement to then be passed to the stored procedure. The functionality of the stored procedure is correct in assessing whether a cabin is available or unavailable however, it doesn't execute automatically when a reservation is assigned to a customer thus an employee must manually run the stored procedure before assigning a cabin number to a customer for a specific voyage.

Query 5

```
1  USE Cruise_Liner;
2  GO
3
4  DROP PROCEDURE IF EXISTS Invoice_Slip;
5  GO
6
7  CREATE PROCEDURE Invoice_Slip
8   @customerID Int
9  AS
10
11 BEGIN
12  DECLARE @rowCount AS Int;
13  DECLARE @firstName AS Varchar(25);
14  DECLARE @lastName AS Varchar(25);
15  DECLARE @deliveryAddress AS Varchar(100);
16  DECLARE @zipcode AS Varchar(10);
17  DECLARE @reservationID AS Int;
18  DECLARE @reservationPrice AS Numeric(6,2);
19  DECLARE @ticketID AS Int;
20  DECLARE @voyage AS Int;
21  DECLARE @cabin AS Int;
22  DECLARE @cabinType AS Varchar(255);
23  DECLARE @cabinPrice AS Numeric(6,2);
24  DECLARE @meal AS Varchar(100);
25  DECLARE @mealPrice AS Numeric(5,2);
26  DECLARE @activityType AS Varchar(100);
27  DECLARE @activityPrice AS Numeric(5,2);
28  DECLARE @totalActivityPrice AS Numeric(6,2);
29  DECLARE @beforeVAT AS Numeric(12,2);
30
31  SELECT @rowCount = COUNT(*)
32  FROM Reservation
33  WHERE Reservation.CustomerID = @customerID;
34
35  IF @rowCount = 0
36      BEGIN
37          PRINT
'##############################################################################
#########'
38          PRINT'          That customer either doesnt exist or has no
reservations listed          '
39          PRINT
'##############################################################################
#########'
40          RETURN;
41      END;
42  BEGIN
43      SELECT @firstname = FirstName, @lastName = LastName, @deliv-
eryAddress = DeliveryAddress, @zipcode = Zipcode
44      FROM Customer
45      WHERE ID = @customerID;
46
47      SELECT @reservationID = ID, @reservationPrice = AmountToPay
48      FROM Reservation
49      WHERE CustomerID = @customerID;
50
51      SELECT @ticketID = ID, @voyage = VoyageNumber, @cabin = Cabin-
Number, @meal = MealType
52      FROM Ticket
53      WHERE ReservationID = @reservationID;
54
```

```
55      SELECT @cabinType = [Type], @cabinPrice = Price
56      FROM Cabins
57      WHERE VoyageNumber = @voyage AND CabinNumber = @cabin;
58
59      SELECT @mealPrice = Price
60      FROM Meal
61      WHERE [Type] = @meal;
62
63      DECLARE activityCursor CURSOR FOR
64          SELECT ActivityType, Price
65          FROM Ticket_Activities, Activities
66          WHERE TicketID = @ticketID AND ActivityType = Activi-
ties.[Type];
67
68      SELECT @totalActivityPrice = SUM(Price)
69      FROM Ticket_Activities, Activities
70      WHERE TicketID = @ticketID AND ActivityType = Activities.[Type];
71
72      SELECT @beforeVAT = @cabinPrice + @mealPrice + @reservationPrice +
@totalActivityPrice;
73      PRINT
'################################################################################
#########'
74      PRINT '-----------------------------LSBU Cruise Liner Invoice------
----------------------'
75      PRINT
'################################################################################
#########'
76      PRINT 'Firstname: '+CONVERT(Char(25),@firstName)+' Lastname: '+CON-
VERT(Char(25),@lastName)
77      PRINT 'Address: '+CONVERT(Char(100),@deliveryAddress)
78      PRINT 'Zipcode: '+CONVERT(Char(10),@zipcode)
79      PRINT '------------------------------------------------------------
----------------------'
80      PRINT 'Price of Reservation: '+CONVERT(Char(10),FORMAT(@reserva-
tionPrice, 'C2', 'gb-GB'))
81      PRINT '------------------------------------------------------------
----------------------'
82      PRINT 'Cabin type: '+CONVERT(Char(255),@cabinType)
83      PRINT 'Cabin Costs '+CONVERT(Char(6),FORMAT(@cabinPrice, 'C2', 'gb-
GB'))
84      PRINT '------------------------------------------------------------
----------------------'
85      PRINT 'Meal Chosen: '+CONVERT(Char(100),@meal)
86      PRINT 'Meal Costs '+CONVERT(Char(5),FORMAT(@mealPrice, 'C2', 'gb-
GB'))
87      PRINT '------------------------------------------------------------
----------------------'
88      PRINT 'Activities: '
89      OPEN activityCursor;
90      FETCH NEXT FROM activityCursor into @activityType, @activityPrice
91      WHILE @@FETCH_STATUS = 0
92          BEGIN
93              PRINT @activityType+' costs '+CONVERT(Char(5),FORMAT(@ac-
tivityPrice, 'C2', 'gb-GB'))
94              FETCH NEXT FROM activityCursor into @activityType, @activi-
tyPrice
95          END;
96      CLOSE activityCursor;
97      DEALLOCATE activityCursor;
```

```
98      PRINT 'Total cost of Activities:
'+CONVERT(Char(6),FORMAT(@totalActivityPrice, 'C2', 'gb-GB'))
99      PRINT '-----------------------------------------------------------
----------------------'
100     PRINT'Total Price Before VAT: '+CONVERT(Char(12),FORMAT(@beforeVAT,
'C2', 'gb-GB'))
101     PRINT '-----------------------------------------------------------
----------------------'
102     PRINT'Total Price After VAT(+20%):
'+CONVERT(Char(12),FORMAT((@beforeVAT*1.2), 'C2', 'gb-GB'))
103     PRINT '-----------------------------------------------------------
----------------------'
104     RETURN;
105 END;
106
107END;
108
109EXEC Invoice Slip @customerID = 1;
```

Figure 40

(Kroenke & Auer, 2016) was used to construct the stored procedure depicted in Figure 40. Firstly, a check is performed to determine if a customer has made a reservation where, if a reservation hasn't been made a message would be returned depicting that the customer doesn't have any reservations as seen in Figure 41.

```
##########################################################################################
          That customer either doesnt exist or has no reservations listed
##########################################################################################

Completion time: 2021-05-11T10:21:29.9168115+01:00
```

Figure 41

Otherwise, the customers details, their reservation details, the cabin they stayed in, the meal they chose and the activities they partook in are displayed with their corresponding prices and a summation given at the end with and without including VAT. A cursor has been used to retrieve the activities that have been chosen by each customer with their associated prices as seen from lines 89 to 97. (Ian, 2019) was used to format each of the price variables which would be outputted from the invoice thus, a future developer could change the currency or number of signification figures for each of the prices by changing the third and second argument given to the FORMAT function respectively. If the Customer ID provided is 1, the stored procedure provides the output depicted in Figure 42. We could query the Voyage table to display information relating to the voyage the customer was on however, it isn't necessary to meet the requirements stated in the specification.

```
################################################################################################
-----------------------------LSBU Cruise Liner Invoice-----------------------------
################################################################################################
Firstname: Spencer                    Lastname: Vazquez
Address: Stinsford Hill House, Stinsford
Zipcode: DT2 8PS
------------------------------------------------------------------------------------------
Price of Reservation: £1,000.00
------------------------------------------------------------------------------------------
Cabin type: Suite
Cabin Costs £300.0
------------------------------------------------------------------------------------------
Meal Chosen: No food
Meal Costs £0.00
------------------------------------------------------------------------------------------
Activities:
Cats costs £50.0
Classical Music costs £0.00
Total cost of Activities: £50.00
------------------------------------------------------------------------------------------
Total Price Before VAT: £1,350.00
------------------------------------------------------------------------------------------
Total Price After VAT(+20%): £1,620.00
------------------------------------------------------------------------------------------

Completion time: 2021-05-11T10:25:57.8963939+01:00
```

Figure 42

The current implementation assumes that the customer has 1 reservation and 1 ticket tied to a reservation where in actuality, a customer could make multiple reservations where each reservation would have a departure and return ticket. To remedy this, we could use a nested cursor to retrieve the reservations made by each customer with another nested cursor to retrieve the tickets associated with each reservation however, we are unsure of how to pass variables between each of the nested cursors to carry out the queries to create the invoice.

Dashboard (Bhatti, 2021)

Although we could have used Power BI to create the dashboard, we chose to use Tableau due to our familiarity with the software from use in tutorials. Additionally, Tableau will allow for better performance for large datasets, as described in (Pedamkar, 2020), thus accounting for the scalability of the Cruise Liner business in the future.

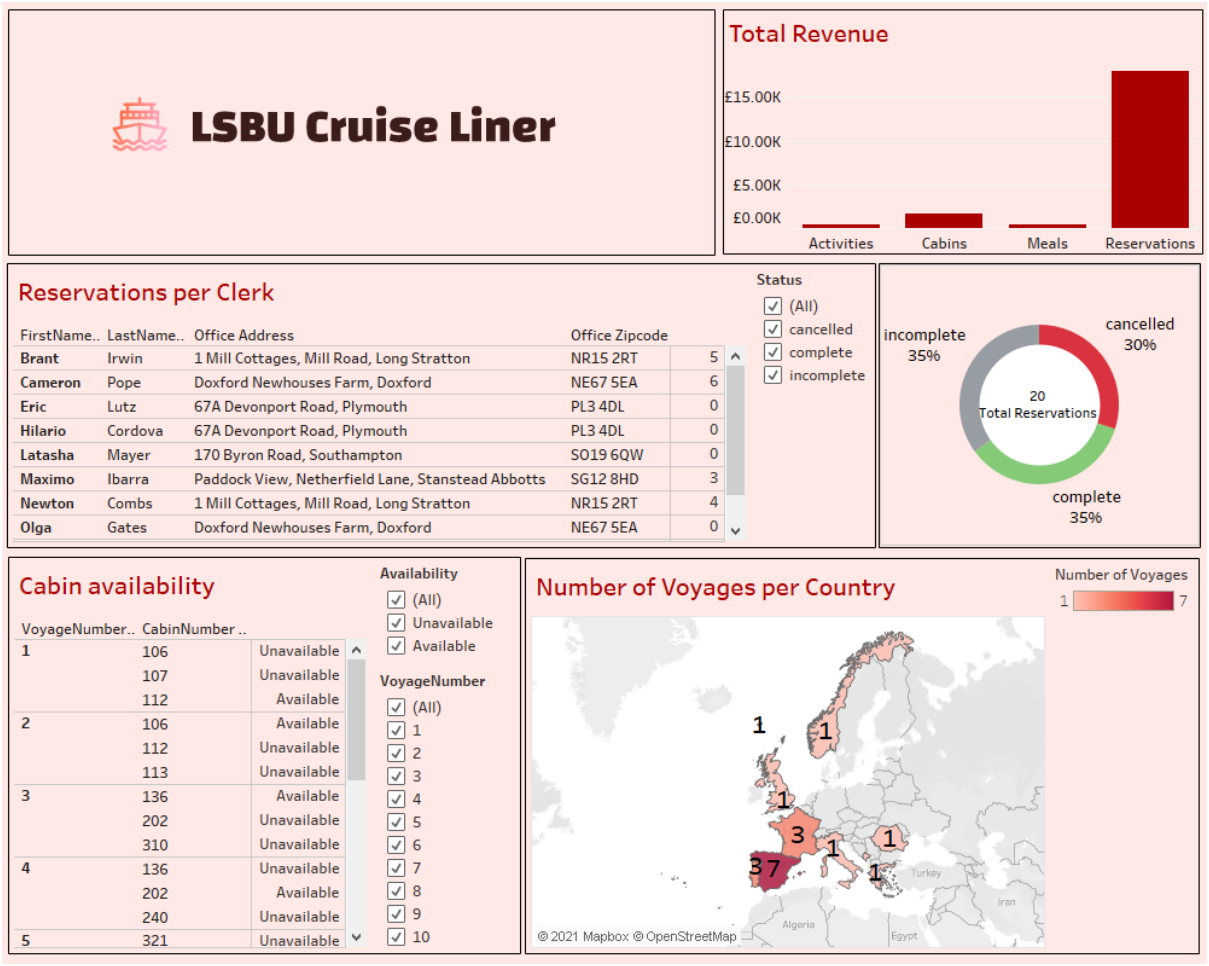

Figure 43

Figure 43 is the dashboard created in Tableau to depict how the LSBU Cruise Liner business operates. (Namecheap, 2021) was used to create the logo in the top left corner of the dashboard where if clicked, it would take the user to the business's website however, as the website doesn't exist, we are directing users that click the logo to www.lsbu.ac.uk. Although similar to the colour scheme used by (The Financial Times, 2021), (Kerin, 2012) was used to colour match the background colour of each of the visualisations to the colours used in the logo.

The visualisation in the top right corner depicts the total revenue made per revenue stream which depicts that the business's prime revenue stream is from reservations while the business is making the least revenue from activities and meals. For the business to improve profits, the business could either raise the prices of cabins, meals and activities or the business would need to refine the types of cabins, meals and activities they are offering. (Dar, 2021) was used when creating the doughnut chart for the total number of reservations made. This visualisation is useful in comparing the number of complete reservations to the total number of reservations. This allows us to investigate why the number of completed reservations is so high or low in comparison to incomplete and cancelled reservations so that we may increase the number of completed reservations in the future.

(tekweirdo, 2018) was used when formatting both table visualisations to remedy the shading caused by assigning a band size to alternating rows. The reservations per clerk visualisation are similar to the query in Task 4 to retrieve the total number of reservations made by each clerk where a filter is used to find the number of cancelled, incomplete and completed reservations made by each clerk. This visualisation can help assess how individual clerks are performing as if a clerk has a high number of reservations but the majority of reservations they have made have been incomplete or cancelled, this could indicate that the clerk is performing inadequately. Alternatively, if many clerks have made no reservations in a specific office location this could indicate that the office is located in a secluded or non-profitable area thus these clerks could be allocated to other office locations. The visualisation of cabin availability is similar to the query in Task 4 to retrieve all available cabins. This visualisation could be used to assess whether a specific cabin type or voyage is popular as if a cabin is available on a voyage after it has departed, the business could investigate why that specific cabin or cabin type was not booked for a customer.

The map visualisation is used to depict the number of voyages to a specific country where the legend dictates how many voyages there are to a country with a lighter colour indicating fewer voyages than a darker colour. This visualisation would be used to advertise more voyages to popular destinations which would improve the overall revenue the business makes.

The dashboard can be found at:
https://public.tableau.com/profile/kamran.bhatti#!/vizhome/CruiseLinerTableauWorkbook/Dashboard2improved

Video Demo (Bhatti, 2021)

https://youtu.be/4RkBQu0-cHA

References

Bell, C., 2000. *Random Addresses Generator.* [Online]
Available at: https://www.doogal.co.uk/RandomAddresses.php
[Accessed 21 April 2021].

Bhatti, K., 2021. *3807942 BDD CW DEMO.* [Online]
Available at: https://youtu.be/4RkBQu0-cHA
[Accessed 13 May 2021].

Bhatti, K., 2021. *CruiseLiner Tableau Workbook.* [Online]
Available at:
https://public.tableau.com/profile/kamran.bhatti#!/vizhome/CruiseLinerTableauWorkbook/Dashboard2improved
[Accessed 7 May 2021].

Bhatti, K., 2021. *LSBUCruiseLiner Case Study C.* [Online]
Available at: https://github.com/k5924/LSBUCruiseLiner
[Accessed 2 May 2021].

Chilton, C., 2019. *The Cleverest Names for Your Boat from A to Z.* [Online]
Available at: https://www.townandcountrymag.com/society/money-and-power/g717/best-boat-names-from-a-to-z/
[Accessed 21 April 2021].

ChrisWalczyk55, 2017. *Top 20 Greatest Musicals of All Time (The Ultimate List).* [Online]
Available at: https://www.imdb.com/list/ls051686144/
[Accessed 21 April 2021].

CruiseMapper, 2015. *Cruise Ship Cruising Speed.* [Online]
Available at: https://www.cruisemapper.com/wiki/762-cruise-ship-cruising-speed
[Accessed 21 April 2021].

Dar, P., 2021. *How to Create a Donut Chart in Tableau.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2021/02/how-to-create-donut-chart-tableau/
[Accessed 7 May 2021].

Diwan, A., 2019. *Get row data for the lowest and highest values in a MySQL column.* [Online]
Available at: https://www.tutorialspoint.com/get-row-data-for-the-lowest-and-highest-values-in-a-mysql-column
[Accessed 28 April 2021].

Gavin, J., 2021. *Drop All Tables in a SQL Server Database.* [Online]
Available at: https://www.mssqltips.com/sqlservertip/6798/drop-all-tables-sql-server/
[Accessed 21 April 2021].

Gheorghiu, R., 2014. *CREATE VIEW must be the only statement in the batch.* [Online]
Available at: https://stackoverflow.com/questions/27272194/create-view-must-be-the-only-statement-in-the-batch
[Accessed 28 April 2021].

Ian, 2019. *How to Format Numbers as Currency in SQL Server (T-SQL).* [Online]
Available at: https://database.guide/how-to-format-numbers-as-currency-in-sql-server-t-sql/
[Accessed 28 April 2021].

Kerin, A., 2012. *Dasbhoards - Changing background colors of the views and legends.* [Online]
Available at: https://community.tableau.com/s/question/0D54T00000C5cyASAR/dasbhoards-changing-background-colors-of-the-views-and-legends
[Accessed 7 May 2021].

Khan, M., 2016. *Insert value for Time Datatype in SQL Server example.* [Online]
Available at: https://www.aspsnippets.com/Articles/Insert-value-for-Time-Datatype-in-SQL-Server-example.aspx
[Accessed 21 April 2021].

Kroenke, D. M. & Auer, D. J., 2016. Data Modeling with the Entity-Relationship Model. In: W. W. University, ed. *Database Processing Fundamentals, Design, and Implementation.* 14th ed. Harlow: Pearson Education Limited, pp. 233-234.

Kroenke, D. M. & Auer, D. J., 2016. SQL for Database Construction and Application Processing. In: W. W. University, ed. *Database Processing Fundamentals, Design, and Implementation.* 14th ed. Harlow: Pearson Education Limited, pp. 338-348.

Kroenke, D. M. & Auer, D. J., 2016. SQL for Database Construction and Application Processing. In: W. W. University, ed. *Database Processing Fundamentals, Design, and Implementation.* 14th ed. Harlow: Pearson Education Limited, pp. 372-385.

Kroenke, D. M. & Auer, D. J., 2016. The Relational Model and Normalization. In: W. W. University, ed. *Database Processing Fundamentals, Design and Implementation.* 14th ed. Harlow: Pearson Education Limited, pp. 172-177.

Map Developers, 2021. *What Country Am I In - Locate your country based on your location.* [Online]
Available at: https://mapdevelopers.com/what-country-am-i-in.php
[Accessed 21 April 2021].

Namecheap, 2021. *Your Logos.* [Online]
Available at: https://www.namecheap.com/logo-maker/app/
[Accessed 7 May 2021].

Pedamkar, P., 2020. *Power BI vs Tableau.* [Online]
Available at: https://www.educba.com/power-bi-vs-tableau/
[Accessed 2 May 2021].

Pedamkar, P., 2020. *Triggers in SQL.* [Online]
Available at: https://www.educba.com/triggers-in-sql/
[Accessed 28 April 2021].

Ports.com, 2010. *ports in Europe (3024).* [Online]
Available at: http://ports.com/browse/europe/#/?cf=&df=A&tf=&view=list&page=false
[Accessed 21 April 2021].

Random Word Generator, 2021. *Random Name Generator.* [Online]
Available at: https://randomwordgenerator.com/name.php
[Accessed 21 April 2021].

Randommer, 2021. *Generate random phone numbers.* [Online]
Available at: https://randommer.io/Phone
[Accessed 21 April 2021].

Refsnes Data, 1999. *MySQL CASE Statement.* [Online]
Available at: https://www.w3schools.com/mysql/mysql_case.asp
[Accessed 28 April 2021].

Refsnes Data, 1999. *SQL DESC Keyword.* [Online]
Available at: https://www.w3schools.com/sql/sql_ref_desc.asp
[Accessed 28 April 2021].

S., D., 2021. *How Much Does a Cruise Ship Weigh?.* [Online]
Available at: https://boatinggeeks.com/how-much-does-a-cruise-ship-weigh/
[Accessed 21 April 2021].

tekweirdo, 2018. *TABLEAU APPLYING COLORS FOR ALTERNATE ROWS.* [Online]
Available at: https://tek-jedi.com/2018/07/18/tableau-applying-colors-for-alternate-rows/
[Accessed 7 May 2021].

Tell Me How Much, 2021. *How Much Fuel Can a Cruise Ship Hold?.* [Online]
Available at: http://www.tellmehowmuch.net/how-much-fuel-can-a-cruise-ship-hold.html
[Accessed 21 April 2021].

The Financial Times, 2021. *Financial Times.* [Online]
Available at: https://www.ft.com/
[Accessed 7 May 2021].

Tutorialspoint, 2021. *T-SQL - Date Functions.* [Online]
Available at: https://www.tutorialspoint.com/t_sql/t_sql_date_functions.htm
[Accessed 21 April 2021].

w3resource.com, 2020. *SQL COUNT() with GROUP by.* [Online]
Available at: https://www.w3resource.com/sql/aggregate-functions/count-with-group-by.php
[Accessed 28 April 2021].

Wikipedia, 2021. *List of diets.* [Online]
Available at: https://en.wikipedia.org/wiki/List_of_diets
[Accessed 21 April 2021].