Introduction

SQL Boat Rental LLC is a company that rents out top class boats for customers to enjoy their breaks as well as maintenance service. They require a database to keep track of the daily transactions of involving customers, rentals, staff members and services.

For each rental order, we need to keep track of the date of the order, rental period, rental status and the number of people and associate them with the payment. The payment should have the payment method, payment status, discount rate if it applies, deposit, and final price. It's also important to keep track of the boats being used to the rental order and the details of the boat (boat type, boat size, boat color, price rate, and boat capacity.)

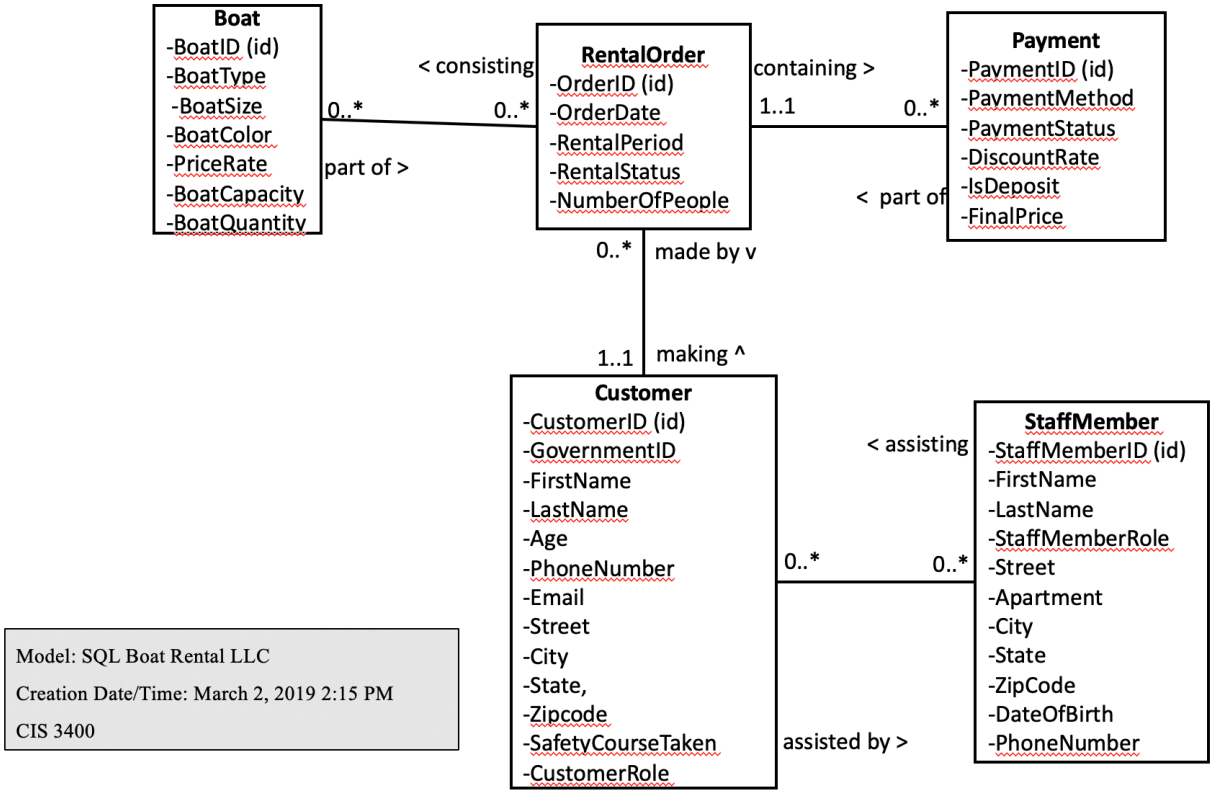
There are a vast amount of staff members available to assist the customers. For each staff member we need to keep track of their personal contract information as well as their staff roles and associate them with the customer.

For each order, we need to keep track of their personal information and their status of safety course taken along with their government ID.

An initial list of entities:

- Customer (CustomerID, GovernmentID, FirstName, LastName, Age, PhoneNumber, Email, Street, City, State, Zipcode, SafetyCourseTaken, Role(Primary/Secondary))
- Boat (BoatID, BoatType, BoatSize, BoatColor, PriceRate, BoatCapacity)
- RentalOrder (OrderID, OrderDate, RentalPeriod, RentalStatus, NumberOfPeople)
- PaymentInfo (TransactionID, PaymentMethod, PaymentStatus, DiscountRate, Deposit, FinalPrice)
- StaffMember (StaffMemberID, StaffRole, FirstName, LastName, PhoneNumber, Email, Street, City, State, Zipcode)

Entity - Relationship Model Diagram



UML Notations

1. One **boat** *may be* <u>part of</u> one or many **rental orders**.
2. One **rental order** *may be* <u>consisting of</u> one or many **boats**.
3. One **rental order** *may be* <u>containing</u> many **payments**.
4. One **payment** *must be* <u>part of</u> one **rental order**.
5. One **rental order** *must be* <u>made by</u> one **customer**.
6. One **customer** *may be* <u>making</u> one or many **rental orders**.
7. One **customer** *may be* <u>assisted by</u> one or many **staff members**.
8. One **staff member** *may be* <u>assisting</u> one or many **customers**.

Normalization

Set of Relations before normalizing:

1. Boat (BoatID, BoatType, BoatSize, BoatColor, PriceRate, BoatCapacity, BoatQuality)
2. RentalOrder (OrderID, CustomerID, OrderDate, RentalPeriod, RentalStatus, NumberOfPeople)
3. Payment (PaymentID, PaymentMethod, PaymentStatus, DiscountRate, isDeposit, FinalPrice)
4. StaffMember (StaffMemberID, FirstName, LastName, StaffMemberRole, Street, Apartment, City, State, ZipCode, DateOfBirth, PhoneNumber, CustomerID)
5. Customer (CustomerID, GovernmentID, FirstName, LastName, Age, PhoneNumber, Email, Street, City, State, ZipCode, SafetyCourseTaken, CustomerRole)

Normalization begins:

Boat (BoatID, BoatType, BoatSize, BoatColor, PriceRate, BoatCapacity, BoatQuality)

Key: BoatID

FD1: BoatID → BoatType, BoatSize, BoatColor, PriceRate, BoatCapacity, BoatQuality

1NF: Yes, meets the definition of a relation

2NF: Yes, no partial key dependencies

3NF: Yes, no transitive key dependencies

RentalOrder (OrderID, CustomerID, OrderDate, RentalPeriod, RentalStatus, NumberOfPeople)

Key: OrderID, CustomerID

FD1: OrderID, CustomerID → OrderDate, RentalPeriod, RentalStatus, NumberOfPeople

1NF: Yes, meets the definition of a relation

2NF: Yes, no partial key dependencies

3NF: Yes, no transitive key dependencies

Order_Boat(OrderID (fk)(key), BoatID (fk)(key))

Key: OrderID, BoatID

Payment (PaymentID, PaymentMethod, PaymentStatus, DiscountRate, isDeposit, FinalPrice)

Key: PaymentID

FD1:PaymentID → PaymentMethod, PaymentStatus, DiscountRate, isDeposit, FinalPrice

1NF: Yes, meets the definition of a relation

2NF: Yes, no partial key dependencies

3NF: Yes, no transitive key dependencies

StaffMember (StaffMemberID, FirstName, LastName, StaffMemberRole, Street, Apartment, City, State, ZipCode, DateOfBirth, PhoneNumber, CustomerID)

Key: StaffMemberID

FD1: StaffMemberID → FirstName, LastName, StaffMemberRole, Street, Apartment, City, State, ZipCode, DateOfBirth, PhoneNumber, CustomerID

FD2: ZipCode → City, State

1NF: Meets the definition of a relation

2NF: No partial key dependencies

3NF: Transitive dependency exists: StaffMemberID → ZipCode and ZipCode → City, State

⇒ Solution: Remove City, State and copy ZipCode

StaffMember (StaffMemberID, FirstName, LastName, StaffMemberRole, Street, Apartment, City, State, ZipCode, DateOfBirth, PhoneNumber)

Key: StaffMemberID

FD1: StaffMemberID → FirstName, LastName, StaffMemberRole, Street, Apartment, ZipCode(fk), DateOfBirth, PhoneNumber

1NF: Meets the definition of a relation

2NF: No partial Key dependencies

3NF: No Transitive dependencies

ZipCodes (ZipCode (key), City, State)

Key: ZipCode

FD1: ZipCode -> City, State

1NF: Meets the definition of a relation

2NF: No partial Key dependencies

3NF: No Transitive dependencies

Customer (CustomerID, GovernmentID, FirstName, LastName, Age, PhoneNumber, Email, Street, City, State, ZipCode, SafetyCourseTaken, CustomerRole)

Key: CustomerID

FD1: CustomerID → GovernmentID, FirstName, LastName, Age, PhoneNumber, Email, Street, City, State, ZipCode, SafetyCourseTaken, CustomerRole

FD2: ZipCode → City, State

FD3: GovernmentID → FirstName, LastName, Age, Street, City, State, ZipCode

1NF: Meets the definition of a relation

2NF: No partial key dependencies

3NF: Transitive dependency exists: CustomerID → ZipCode and ZipCode → City, State

⇒ Solution: Remove City, State and Copy ZipCode

Customer (CustomerID, GovernmentID, FirstName, LastName, Age, PhoneNumber, Email, Street, ZipCode(fk), SafetyCourseTaken, CustomerRole)

Key: CustomerID

FD1: CustomerID → GovernmentID, FirstName, LastName, Age, PhoneNumber, Email, Street, ZipCode, SafetyCourseTaken, CustomerRole

FD2: GovernmentID → FirstName, LastName, Age, Street, ZipCode

1NF: Meets the definition of a relation

2NF: No partial key dependencies

3NF: Transitive dependency exists: CustomerID-> GovernmentID and GovernmentID -> FirstName, LastName, Age, Street, ZipCode

⇒ Solution: Remove FirstName, LastName, Age, Street, ZipCode and Copy GovernmentID

Customer (CustomerID, GovernmentID(fk), PhoneNumber, Email, ZipCode(fk), SafetyCourseTaken, CustomerRole)

Key: CustomerID

FD1: CustomerID → GovernmentID, PhoneNumber, Email, SafetyCourseTaken, CustomerRole

Gov(GovernmentID, Name, LastName, Age, Street, ZipCode)

Key: GovernmentID

FD1: GovermentID → Name, LastName, Age, Street, ZipCode

Initial Set of Relations:

1. Boat(BoatID(key), BoatType, BoatSize, BoatColor, PriceRate, BoatCapacity, BoatQuality)

2. Rental Order (OrderID(key), CustomerID (fk), OrderDate, RentalPeriod, RentalStatus, NumberOfPeople)
3. Payment (PaymentID(key), PaymentMethod, PaymentStatus, DiscountRate, IsDeposit, FinalPrice, OrderID(fk))
4. StaffMember (StaffMemberID(key), FirstName, LastName, StaffMemberRole, Street, Apartment, City, State, ZipCode(fk), DateOfBirth, PhoneNumber, CustomerID(fk))
5. Customer (CustomerID(key), GovernmentID(fk), PhoneNumber, Email, ZipCode(fk), SafetyCourseTaken, CustomerRole)
6. Gov(GovernmentID(key), Name, LastName, Age, Street, ZipCode(fk))
7. ZipCodes(ZipCode (key), City, State)
8. Order_Boat(OrderID (fk)(key), BoatID (fk)(key)

Generally, the steps to follow for each relation are to write out the relation including all attribute names and indicate keys and foreign keys. Next, the key and all functional dependencies should be stated. Then go through the definitions of each normal form starting with 1NF to 3NF. If a relation meets the definition of a normal form, move up to the next higher normal form. Otherwise (a partial-key and/ or transitive dependency exist), split the relation into two new relations and then begin the normalization process from the beginning with each of these two new relations. Last but not least, wrap up with the list of the normalized relations and their data. In this project, an intersection relation containing a composite key Order_Boat with (OrderID (fk)(Key), BoatID (fk) (Key)) was added due do the fact that one order could include many boats, and a boat could show up and be rented on many different orders. Moreover, the ZipCode relation was denormalized to reduce the number of relations in the database schema as well as keep the project simpler; and it was also removed from the Customer relation as it existed in the Gov relation as the customers' zip codes.

Final Set of Relations:

1. Boat(BoatID(key), BoatType, BoatSize, BoatColor, PriceRate, BoatCapacity, BoatQuantity)

Boat

BoatID	BoatType	BoatSize	BoatColor	PriceRate	BoatCapacity	BoatQuantity
3401	BR10 Racer Extended	18' - 22'	Blue	65	5	2
3402	BR20 Fishing Walk Th	20' - 24'	White	70	5	3
3403	BR30 Sport SS Extend	18' - 22'	Red	65	8	3
3404	BR40 Ultra Fish & Cr	20' - 24'	Black/Gray	70	6	3
3405	BR50 Deck Boat	20' - 24'	Green	70	4	2
3406	BR11 Racer Extended	22' - 26'	White	100	10	3
3407	BR21 Fishing Walk Th	24' - 28'	Blue	110	10	3
3408	BR31 Sport SS Extend	22' - 26'	Red	105	12	4
3409	BR41 Ultra Fish & Cr	24' - 28'	Gray	105	12	4
3410	BR32 Sport SS Conver	24' - 28'	White	100	6	3

2. Rental Order (OrderID(key), CustomerID (fk), OrderDate, RentalPeriod, RentalStatus, NumberOfPeople)

RentalOrder

OrderID	CustomerID	OrderDate	RentalPeriod	RentalStatus	NumberOfPeople
9901	1	5/5/2019	5 hours	Available	3
9902	2	5/5/2019	2 hours	Available	4
9903	3	5/6/2019	4 hours	Available	6
9904	4	5/6/2019	8 hours	Available	8
9905	5	5/7/2019	6 hours	Available	5
9906	6	5/7/2019	7 hours	Available	6
9907	7	5/7/2019	5 hours	Available	4
9908	8	5/8/2019	4 hours	Available	8
9909	9	5/10/2019	5 hours	Available	11
9910	10	5/12/2019	6 hours	Available	9

3. Payment (PaymentID(key), PaymentMethod, PaymentStatus, DiscountRate, isDeposit, FinalPrice, OrderID(fk))

Payment

PaymentID	OrderID	PaymentMethod	PaymentStatus	DiscountRate	isDeposit	FinalPrice
8801	9901	Credit Card	Paid	0	50	325
8802	9902	Credit Card	Paid	0	50	140
8803	9903	Cash	Unpaid	0.1	50	234
8804	9904	Credit Card	Paid	0	50	560
8805	9905	Cash	Unpaid	0.1	50	378
8806	9906	Credit Card	Paid	0	50	700
8807	9907	Credit Card	Paid	0	50	550
8808	9908	Cash	Unpaid	0.1	50	378
8809	9909	Credit Card	Paid	0	50	525
8810	9910	Credit Card	Paid	0	50	600

4. StaffMember (StaffMemberID(key), FirstName, LastName, StaffMemberRole, Street, City, State, ZipCode, DateOfBirth, PhoneNumber, CustomerID(fk))

StaffMember

StaffMemberID	FirstName	LastName	StaffMemberRole	Street	City	State	ZipCode	DateOfBirth	PhoneNumber	CustomerID
1101	Suho	Kim	President	8349 El Dorado Lane	Hanover Park	IL	60133	5/22/1991	6937594756	1
1102	Xiumin	Kim	Manager	967 Glen Eagles Ave	Sumter	SC	29150	3/26/1990	3464373535	2
1103	Lay	Zhang	Manager	49 East Holly Court	Clarksburg	WV	26301	10/7/1991	7545635642	3
1104	Baekhyun	Byun	Manager	5 High Lane	Watertown	MA	02472	5/6/1992	3463467809	4
1105	Chen	Kim	Staff	884 Hamilton Court	Westerville	OH	43081	9/21/1992	5478965156	5
1106	Chanyeol	Staff	President	591 Primrose Drive	Wayne	NJ	07470	11/27/1992	8749651287	6
1107	Kyung Soo	Staff	President	350 Plumb Branch Drive	Port Huron	MI	48060	1/12/1993	9871543248	7
1108	Sehun	Oh	Staff	9971 W. Acacia Ave	Forney	TX	75126	4/12/1994	9871426588	8
1109	Kris	Wu	Staff	157 Maple Lane	Adrian	MI	49221	11/6/1990	3165489651	9
1110	Zitao	Huang	Staff	Hartford St.	Front Royal	VA	22630	5/2/1993	9182457121	10

5. Customer (CustomerID(key), GovernmentID(fk), PhoneNumber, Email, SafetyCourseTaken, CustomerRole)

Customer

CustomerID	GovernmentID	PhoneNumber	Email	SafetyCourseTaken	CustomerRole
1	3301	8151234567	gs@upllink.com	Yes	Customer
2	3302	8184578989	rg@email.com	Yes	Customer
3	3303	2081459898	ws@prof.edu	Yes	Customer
4	3304	6038987412	jh@snow.com	Yes	Customer
5	3305	3035899865	sp@uplink.com	Yes	Customer
6	3306	2078742356	sp@oal.com	Yes	Customer
7	3307	3079865879	ap@uplink.com	Yes	Customer
8	3308	6081545698	tk@oal.com	Yes	Customer
9	3309	8155545568	sk@uplink.com	Yes	Customer
10	3310	3035589865	ph@geo.com	Yes	Customer

6. Gov(GovernmentID(key), FirstName, LastName, DateOfBirth, Street, City, State, ZipCode)

Gov

GovernmentID	FirstName	LastName	DateOfBirth	Street	City	State	ZipCode
3301	Eunbi	Kwon	9/27/1995	404 Prairie Rd	Sycamore	IL	60178
3302	Sakura	Miyawaki	3/19/1998	109 Hagen Dr	Burbank	CA	91501
3303	Hyewon	Kang	7/5/1999	494 Crane Cir	Boise	ID	83703
3304	Yena	Choi	9/29/1999	9003 Lincoln Hwy	Portsmouth	NH	00215
3305	Chaeyeon	Lee	1/11/2000	124 Timber Ln	Denver	CO	80201
3306	Chaewon	Kim	8/1/2000	33 Kimberly Dr	Stonington	ME	04681
3307	Minjoo	Kim	2/5/2001	981 Ridge Rd	Cheyenne	WY	82001
3308	Nako	Yabuki	6/18/2001	4 Spring St	Madison	WI	53701
3309	Hitomi	Honda	2/6/2001	99 Charles Pl	DeKalb	IL	60115
3310	Yuri	Jo	4/22/2001	199 16th St	Denver	CO	80201

7. Order_Boat(OrderID (fk)(key), BoatID (fk)(key))

Order_Boat

OrderID	BoatID
9901	3401
9902	3401
9903	3403
9904	3406
9905	3410
9906	3406
9907	3402
9908	3408
9909	3409
9910	3408

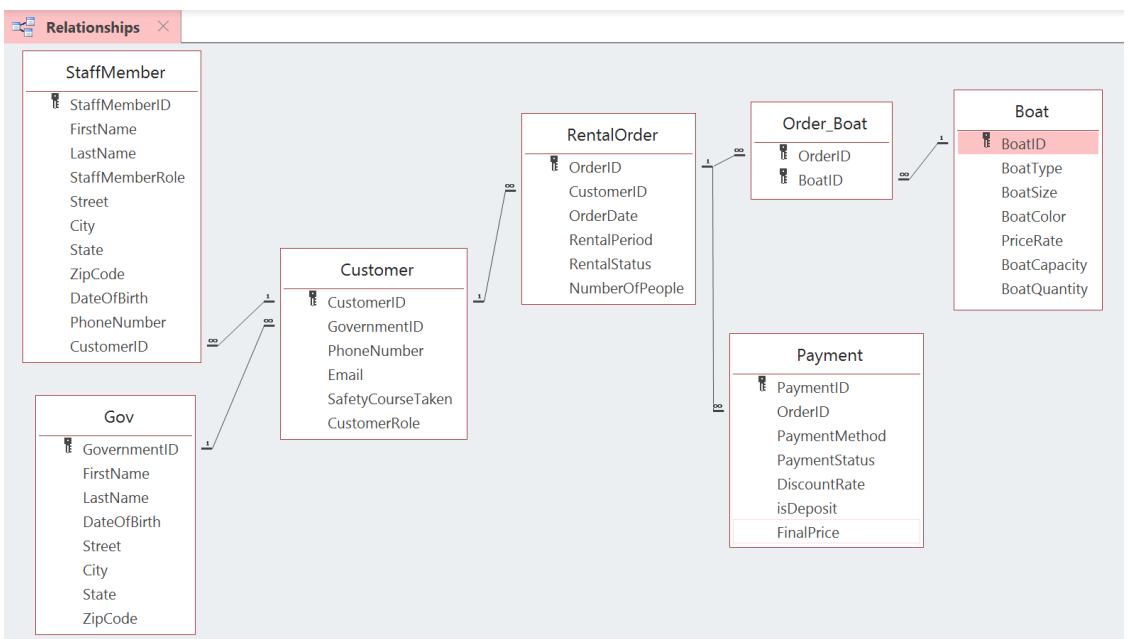
Data Definition Language in SQL

```
CREATE TABLE Boat (
```

```
   BoatID          NUMBER NOT NULL,
   BoatType        VARCHAR(20),
   BoatSize        VARCHAR(10),
   BoatColor       VARCHAR(20),
   PriceRate       NUMBER,
   BoatCapacity    NUMBER,
   BoatQuantity    NUMBER,
   CONSTRAINT      pk_boat
   PRIMARY KEY     (BoatID)
);

CREATE TABLE RentalOrder (
   OrderID          NUMBER NOT NULL,
   CustomerID       NUMBER,
   OrderDate        DATE,
   RentalPeriod     VARCHAR(10),
   RentalStatus     VARCHAR(20),
   NumberOfPeople   NUMBER,
   CONSTRAINT       pk_rentalorder
   PRIMARY KEY      (OrderID)
);

CREATE TABLE Payment (
   PaymentID           NUMBER NOT NULL,
   OrderID             NUMBER,
   PaymentMethod       VARCHAR(20),
   PaymentStatus       VARCHAR(20),
   DiscountRate        NUMBER,
   isDeposit           NUMBER,
   FinalPrice          NUMBER,
   CONSTRAINT          pk_payment
   PRIMARY KEY         (PaymentID)
);

CREATE TABLE Customer (
   CustomerID        NUMBER NOT NULL,
   GovernmentID      NUMBER,
   PhoneNumber       NUMBER,
   Email             VARCHAR(50),
   SafetyCourseTaken VARCHAR(3),
   CustomerRole      VARCHAR(20),
   CONSTRAINT        pk_customer
   PRIMARY KEY       (CustomerID)
);
CREATE TABLE StaffMember (
   StaffMemberID    NUMBER NOT NULL,
   FirstName        VARCHAR(30),
   LastName         VARCHAR(30),
   StaffMemberRole  VARCHAR(20),
```

```
   Street           VARCHAR(50),
   City             VARCHAR(50),
   State            VARCHAR(50),
   ZipCode          VARCHAR(10),
   DateOfBirth      DATE,
   PhoneNumber      NUMBER,
   CustomerID       NUMBER,
   CONSTRAINT       pk_staffmember
   PRIMARY KEY      (StaffMemberID)
);

CREATE TABLE Gov (
   GovernmentID  NUMBER NOT NULL,
   FirstName     VARCHAR(30),
   LastName      VARCHAR(30),
   DateOfBirth   DATE,
   Street        VARCHAR(50),
   City          VARCHAR(50),
   State         VARCHAR(50),
   ZipCode       VARCHAR(10),
   CONSTRAINT    pk_gov
   PRIMARY KEY   (GovernmentID)
);

CREATE TABLE Order_Boat (
   OrderID      NUMBER NOT NULL,
   BoatID       NUMBER NOT NULL,
   CONSTRAINT   pk_order_boat
   PRIMARY KEY  (OrderID, BoatID)
);

ALTER TABLE Customer
   ADD CONSTRAINT fk_Customer_Gov
     FOREIGN KEY (GovernmentID)
        REFERENCES Gov (GovernmentID)

ALTER TABLE StaffMember
 ADD CONSTRAINT fk_StaffMember_Customer
     FOREIGN KEY (CustomerID)
        REFERENCES Customer (CustomerID)

ALTER TABLE Order_Boat
   ADD CONSTRAINT fk_Order_Boat_RentalOrder
```

```
    FOREIGN KEY (OrderID)
       REFERENCES RentalOrder (OrderID)

ALTER TABLE Order_Boat
   ADD CONSTRAINT fk_Order_Boat_Boat
     FOREIGN KEY (BoatID)
        REFERENCES Boat (BoatID)

ALTER TABLE Payment
   ADD CONSTRAINT fk_Payment_RentalOrder
     FOREIGN KEY (OrderID)
        REFERENCES RentalOrder (OrderID)

ALTER TABLE RentalOrder
   ADD CONSTRAINT fk_RentalOrder_Customer
     FOREIGN KEY (CustomerID)
        REFERENCES Customer (CustomerID)
```




```
INSERT INTO Boat VALUES (3401, 'BR10 Racer Extended Walk Thru Boat',
"18' - 22'", 'Blue', 65, 5, 2);

INSERT INTO Boat VALUES (3402, 'BR20 Fishing Walk Thru Boat', "20' -
24'", 'White', 70, 5, 3);

INSERT INTO Boat VALUES (3403, 'BR30 Sport SS Extended Walk Thru
Boat', "18' - 22'", 'Red', 65, 8, 3);

INSERT INTO Boat VALUES (3404, 'BR40 Ultra Fish & Cruise Boat', "20' -
24'", "Black/Gray", 70, 6, 3);
```

```
INSERT INTO Boat VALUES (3405, 'BR50 Deck Boat', "20' - 24'", 'Green',
70, 4, 2);

INSERT INTO Boat VALUES (3406, 'BR11 Racer Extended Walk Thru Boat
XL', "22' - 26'", 'White', 100, 10, 3);

INSERT INTO Boat VALUES (3407, 'BR21 Fishing Walk Thru Boat XL', "24'
- 28'", 'Blue', 110, 10, 3);

INSERT INTO Boat VALUES (3408, 'BR31 Sport SS Extended Walk Thru Boat
XL', "22' - 26'", 'Red', 105, 12, 4);

INSERT INTO Boat VALUES (3409, 'BR41 Ultra Fish & Cruise Boat XL',
"24' - 28'", 'Gray', 105, 12, 4);

INSERT INTO Boat VALUES (3410, 'BR32 Sport SS Conversation Lounge
Boat', "24' - 28'", 'White', 100, 6, 3);


INSERT INTO Gov VALUES (3301, 'Eunbi', 'Kwon', '09/27/1995', '404
Prairie Rd', 'Sycamore', 'IL', '60178');

INSERT INTO Gov VALUES (3302, 'Sakura', 'Miyawaki', '03/19/1998', '109
Hagen Dr', 'Burbank', 'CA', '91501');

INSERT INTO Gov VALUES (3303, 'Hyewon', 'Kang', '07/05/1999', '494
Crane Cir', 'Boise', 'ID', '83703');

INSERT INTO Gov VALUES (3304, 'Yena', 'Choi', '09/29/1999', '9003
Lincoln Hwy', 'Portsmouth', 'NH', '00215');

INSERT INTO Gov VALUES (3305, 'Chaeyeon', 'Lee', '01/11/2000', '124
Timber Ln', 'Denver', 'CO', '80201');

INSERT INTO Gov VALUES (3306, 'Chaewon', 'Kim', '08/01/2000', '33
Kimberly Dr', 'Stonington', 'ME', '04681');

INSERT INTO Gov VALUES (3307, 'Minjoo', 'Kim', '02/05/2001', '981
Ridge Rd', 'Cheyenne', 'WY', '82001');

INSERT INTO Gov VALUES (3308, 'Nako', 'Yabuki', '06/18/2001', '4
Spring St', 'Madison', 'WI', '53701');

INSERT INTO Gov VALUES (3309, 'Hitomi', 'Honda', '02/06/2001', '99
Charles PI', 'DeKalb', 'IL', '60115');

INSERT INTO Gov VALUES (3310, 'Yuri', 'Jo', '04/22/2001', '199 16th
St', 'Denver', 'CO', '80201');


INSERT INTO Customer VALUES (001, 3301, 8151234567, "gs@upllink.com",
'Yes', 'Customer');

INSERT INTO Customer VALUES (002, 3302, 8184578989, "rg@email.com",
'Yes', 'Customer');
```

```
INSERT INTO Customer VALUES (003, 3303, 2081459898, "ws@prof.edu",
'Yes', 'Customer');

INSERT INTO Customer VALUES (004, 3304, 6038987412, "jh@snow.com",
'Yes', 'Customer');

INSERT INTO Customer VALUES (005, 3305, 3035899865, "sp@uplink.com",
'Yes', 'Customer');

INSERT INTO Customer VALUES (006, 3306, 2078742356, "sp@oal.com",
'Yes', 'Customer');

INSERT INTO Customer VALUES (007, 3307, 3079865879, "ap@uplink.com",
'Yes', 'Customer');

INSERT INTO Customer VALUES (008, 3308, 6081545698, "tk@oal.com",
'Yes', 'Customer');

INSERT INTO Customer VALUES (009, 3309, 8155545568, "sk@uplink.com",
'Yes', 'Customer');

INSERT INTO Customer VALUES (010, 3310, 3035589865, "ph@geo.com",
'Yes', 'Customer');


INSERT INTO StaffMember VALUES (1101, "Suho", "Kim", "President",
"8349 El Dorado Lane", "Hanover Park", "IL", "60133", "05/22/1991",
6937594756, NULL);

INSERT INTO StaffMember VALUES (1102, "Xiumin", "Kim", "Manager", "967
Glen Eagles Ave", "Sumter", "SC", "29150", "03/26/1990", 3464373535,
002);

INSERT INTO StaffMember VALUES (1103, "Lay", "Zhang", "Manager", "49
East Holly Court", "Clarksburg", "WV", "26301", "10/07/1991",
7545635642, 003);

INSERT INTO StaffMember VALUES (1104, "Baekhyun", "Byun", "Staff", "5
High Lane", "Watertown", "MA", "02472", "05/06/1992", 3463467809,
004);

INSERT INTO StaffMember VALUES (1105, "Chen", "Kim", "Staff", "884
Hamilton Court", "Westerville", "OH", "43081", "09/21/1992",
5478965156, 005);

INSERT INTO StaffMember VALUES (1106, "Chanyeol", "Park", "Staff",
"591 Primrose Drive", "Wayne", "NJ", "07470", "11/27/1992",
8749651287, 006);

INSERT INTO StaffMember VALUES (1107, "Kyung Soo", "Do", "Staff", "350
Plumb Branch Drive", "Port Huron", "MI", "48060", "01/12/1993",
9871543248, 007);

INSERT INTO StaffMember VALUES (1108, "Sehun", "Oh", "Staff", "9971 W.
Acacia Ave", "Forney", "TX", "75126", "04/12/1994", 9871426588, 008);
```

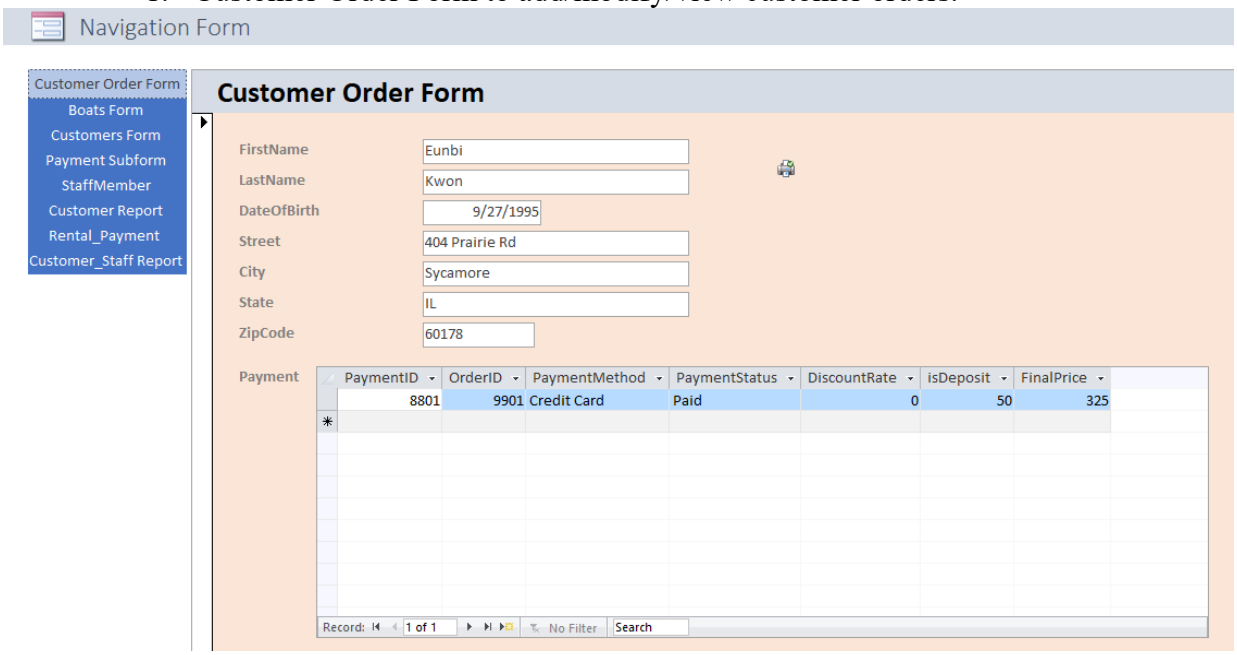
```
INSERT INTO StaffMember VALUES (1109, "Kris", "Wu", "Staff", "157
Maple Lane", "Adrian", "MI", "49221", "11/06/1990", 3165489651, 009);

INSERT INTO StaffMember VALUES (1110, "Zitao", "Huang", "Staff",
"Hartford St.", "Front Royal", "VA", "22630", "05/02/1993",
9182457121, 010);

INSERT INTO StaffMember VALUES (1111, "Han", "Lu", "Staff", "Benson
St.", "Vancity", "VA", "23631", "04/20/1990", 5473637424,001);


INSERT INTO RentalOrder VALUES (9901, 001, "05/05/2019", "5 hours",
"Available", 3);

INSERT INTO RentalOrder VALUES (9902, 002, "05/05/2019", "2 hours",
"Available", 4);

INSERT INTO RentalOrder VALUES (9903, 003, "05/06/2019", "4 hours",
"Available", 6);

INSERT INTO RentalOrder VALUES (9904, 004, "05/06/2019", "8 hours",
"Available", 8);

INSERT INTO RentalOrder VALUES (9905, 005, "05/07/2019", "6 hours",
"Available", 5);

INSERT INTO RentalOrder VALUES (9906, 006, "05/07/2019", "7 hours",
"Available", 6);

INSERT INTO RentalOrder VALUES (9907, 007, "05/07/2019", "5 hours",
"Available", 4);

INSERT INTO RentalOrder VALUES (9908, 008, "05/08/2019", "4 hours",
"Available", 8);

INSERT INTO RentalOrder VALUES (9909, 009, "05/10/2019", "5 hours",
"Available", 11);

INSERT INTO RentalOrder VALUES (9910, 010, "05/12/2019", "6 hours",
"Available", 9);


INSERT INTO Order_Boat VALUES (9901, 3401);

INSERT INTO Order_Boat VALUES (9902, 3401);

INSERT INTO Order_Boat VALUES (9903, 3403);

INSERT INTO Order_Boat VALUES (9904, 3406);

INSERT INTO Order_Boat VALUES (9905, 3410);

INSERT INTO Order_Boat VALUES (9906, 3406);

INSERT INTO Order_Boat VALUES (9907, 3402);

INSERT INTO Order_Boat VALUES (9908, 3408);
```

```
INSERT INTO Order_Boat VALUES (9909, 3409);

INSERT INTO Order_Boat VALUES (9910, 3408);


INSERT INTO Payment VALUES (8801, 9901, "Credit Card", "Paid", 0, 50,
325);

INSERT INTO Payment VALUES (8802, 9902, "Credit Card", "Paid", 0, 50,
140);

INSERT INTO Payment VALUES (8803, 9903, "Cash", "Unpaid", .1, 50,
234);

INSERT INTO Payment VALUES (8804, 9904, "Credit Card", "Paid", 0, 50,
560);

INSERT INTO Payment VALUES (8805, 9905, "Cash", "Unpaid", .1, 50,
378);

INSERT INTO Payment VALUES (8806, 9906, "Credit Card", "Paid", 0, 50,
700);

INSERT INTO Payment VALUES (8807, 9907, "Credit Card", "Paid", 0, 50,
550);

INSERT INTO Payment VALUES (8808, 9908, "Cash", "Unpaid", .1, 50,
378);

INSERT INTO Payment VALUES (8809, 9909, "Credit Card", "Paid", 0, 50,
525);

INSERT INTO Payment VALUES (8810, 9910, "Credit Card", "Paid", 0, 50,
600);
```

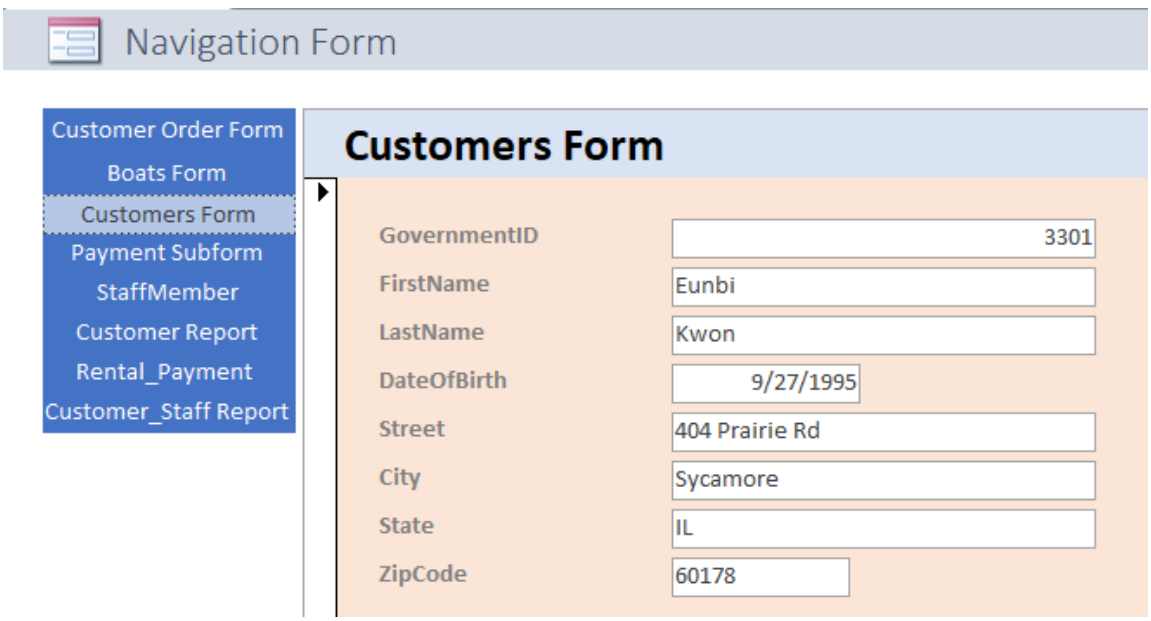
Applications with Forms, Reports and Queries

A. Forms and Reports

1. Customer Order Form to add/modify/view customer orders.



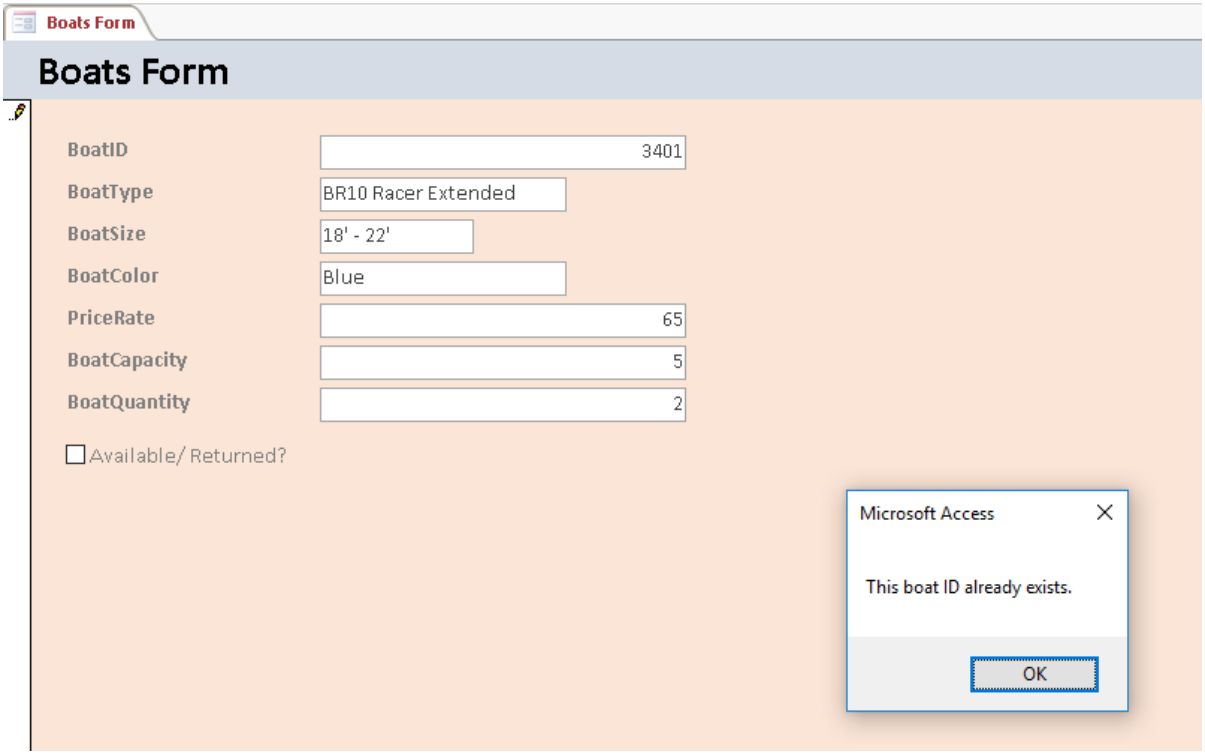
2. Customers Form to add add/modify/view customers



VBA Code:

```
Private Sub FirstName_AfterUpdate()
      FirstName = StrConv(FirstName, vbProperCase)
End Sub
Private Sub LastName_AfterUpdate()
      LastName = StrConv(LastName, vbProperCase)
End Sub
```

3. Boats Form to add/modify/view boats

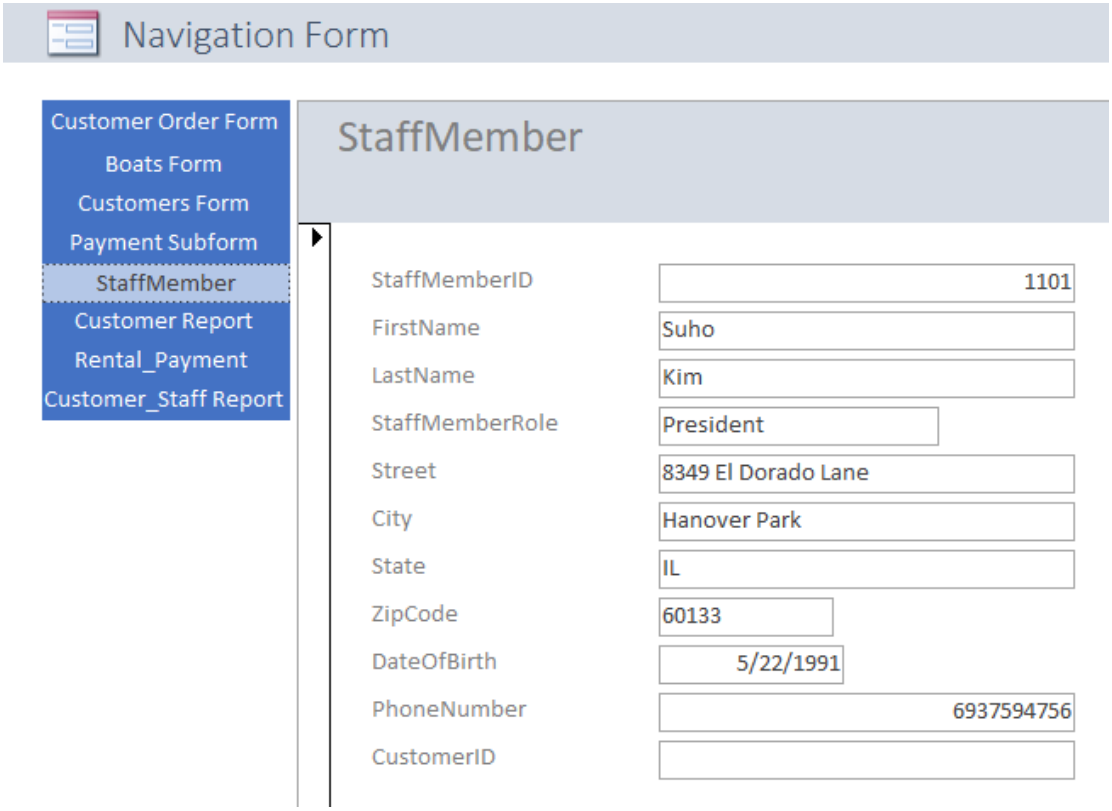


Note: New boolean (Yes/No) field has been added in the tables and the form to check if the boat are available or returned.

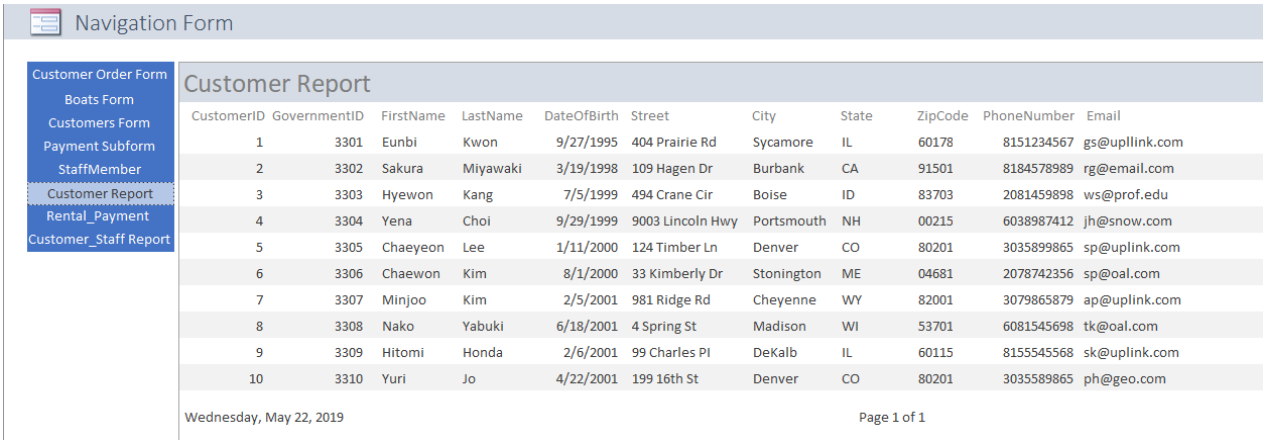
VBA Code: This event checks if the boat and/or an ID has already existed or not.

```
Private Sub BoatID_AfterUpdate()

Dim NewBoat As String
Dim stLinkCriteria As String

NewBoat = Me.BoatID.Value
stLinkCriteria = "[BoatID] = " & NewBoat

If Me.BoatID = DLookup("[BoatID]", "boat", _
stLinkCriteria) Then
     MsgBox ("This boat ID already exists.")
End If

End Sub
```

4. Staff Member Form to add/modify/view staff members



5. Customer Report to view list of all customers



CustomerID	GovernmentID	FirstName	LastName	DateOfBirth	Street	City	State	ZipCode	PhoneNumber	Email
1	3301	Eunbi	Kwon	9/27/1995	404 Prairie Rd	Sycamore	IL	60178	8151234567	gs@upllink.com
2	3302	Sakura	Miyawaki	3/19/1998	109 Hagen Dr	Burbank	CA	91501	8184578989	rg@email.com
3	3303	Hyewon	Kang	7/5/1999	494 Crane Cir	Boise	ID	83703	2081459898	ws@prof.edu
4	3304	Yena	Choi	9/29/1999	9003 Lincoln Hwy	Portsmouth	NH	00215	6038987412	jh@snow.com
5	3305	Chaeyeon	Lee	1/11/2000	124 Timber Ln	Denver	CO	80201	3035899865	sp@uplink.com
6	3306	Chaewon	Kim	8/1/2000	33 Kimberly Dr	Stonington	ME	04681	2078742356	sp@oal.com
7	3307	Minjoo	Kim	2/5/2001	981 Ridge Rd	Cheyenne	WY	82001	3079865879	ap@uplink.com
8	3308	Nako	Yabuki	6/18/2001	4 Spring St	Madison	WI	53701	6081545698	tk@oal.com
9	3309	Hitomi	Honda	2/6/2001	99 Charles Pl	DeKalb	IL	60115	8155545568	sk@uplink.com
10	3310	Yuri	Jo	4/22/2001	199 16th St	Denver	CO	80201	3035589865	ph@geo.com

6. Rental_Payment to view Rentals

Navigation Form

Customer Order Form
Boats Form
Customers Form
Payment Subform
StaffMember
Customer Report
Rental_Payment
Customer_Staff Report

Rental_Payment Report

FirstName	LastName	OrderDate	RentalPeriod	RentalStatus	lumberOfPeople	Pmt_Method	Pmt_Status	DiscountRate	isDeposit	FinalPrice
Eunbi	Kwon									
		5/5/2019	5 hours	Available	3					
						Credit Card	Paid	0	50	325
Sakura	Miyawaki									
		5/5/2019	2 hours	Available	4					
						Credit Card	Paid	0	50	140
Hyewon	Kang									
		5/6/2019	4 hours	Available	6					
						Cash	Unpai	0.1	50	234
Yena	Choi									
		5/6/2019	8 hours	Available	8					
						Credit Card	Paid	0	50	560
Chaeyeon	Lee									
		5/7/2019	6 hours	Available	5					
						Cash	Unpai	0.1	50	378
Chaewon	Kim									
		5/7/2019	7 hours	Available	6					
						Credit Card	Paid	0	50	700
Minjoo	Kim									
		5/7/2019	5 hours	Available	4					
						Credit Card	Paid	0	50	550
Nako	Yabuki									
		5/8/2019	4 hours	Available	8					
						Cash	Unpai	0.1	50	378
Hitomi	Honda									
		#########	5 hours	Available	11					
						Credit Card	Paid	0	50	525
Yuri	Jo									
		#########	6 hours	Available	9					
						Credit Card	Paid	0	50	600

Wednesday, May 22, 2019 Page 1 of 1

7. Customer_Staff Report to view who was helped by which staff member

Navigation Form

Customer Order Form
Boats Form
Customers Form
Payment Subform
StaffMember
Customer Report
Rental_Payment
Customer_Staff Report

Customer_Staff Report

CustomerID	Gov_FirstName	Gov_LastName	StaffMemberID	StaffMember_FirstNam	StaffMember_LastName	StaffMemberRole
1	Eunbi	Kwon				
			1111	Han	Lu	Staff
2	Sakura	Miyawaki				
			1102	Xiumin	Kim	Manager
3	Hyewon	Kang				
			1103	Lay	Zhang	Manager
4	Yena	Choi				
			1104	Baekhyun	Byun	Staff
5	Chaeyeon	Lee				
			1105	Chen	Kim	Staff
6	Chaewon	Kim				
			1106	Chanyeol	Park	Staff
7	Minjoo	Kim				
			1107	Kyung Soo	Do	Staff
8	Nako	Yabuki				
			1108	Sehun	Oh	Staff
9	Hitomi	Honda				
			1109	Kris	Wu	Staff
10	Yuri	Jo				
			1110	Zitao	Huang	Staff

Wednesday, May 22, 2019 Page 1 of 1

B. Queries

1. Customer Reservation form:

```
SELECT g.FirstName, g.LastName, c.PhoneNumber, c.Email,
       r.OrderDate, r.RentalPeriod,r.NumberOfPeople, b.BoatType,
       b.BoatColor
FROM (((Gov g INNER JOIN Customer c
                  ON g.GovernmentID = c.GovernmentID)
                 INNER JOIN RentalOrder r
                  ON r.CustomerID=c.CustomerID)
                INNER JOIN Order_Boat ob
                  ON ob.OrderID = r.OrderID)
                 INNER JOIN Boat b
                  ON b.BoatID = ob.BoatID
```

Customer Reservation Form

FirstName	LastName	PhoneNumber	Email	OrderDate	RentalPeriod	NumberOfPeople	BoatType	BoatColor
Eunbi	Kwon	8151234567	gs@upllink.com	5/5/2019	5 hours	3	BR10 Racer Extended	Blue
Sakura	Miyawaki	8184578989	rg@email.com	5/5/2019	2 hours	4	BR10 Racer Extended	Blue
Hyewon	Kang	2081459898	ws@prof.edu	5/6/2019	4 hours	6	BR30 Sport SS Extend	Red
Yena	Choi	6038987412	jh@snow.com	5/6/2019	8 hours	8	BR11 Racer Extended	White
Chaeyeon	Lee	3035899865	sp@uplink.com	5/7/2019	6 hours	5	BR32 Sport SS Conver	White
Chaewon	Kim	2078742356	sp@oal.com	5/7/2019	7 hours	6	BR11 Racer Extended	White
Minjoo	Kim	3079865879	ap@uplink.com	5/7/2019	5 hours	4	BR20 Fishing Walk Th	White
Nako	Yabuki	6081545698	tk@oal.com	5/8/2019	4 hours	8	BR31 Sport SS Extend	Red
Hitomi	Honda	8155545568	sk@uplink.com	5/10/2019	5 hours	11	BR41 Ultra Fish & Cr	Gray
Yuri	Jo	3035589865	ph@geo.com	5/12/2019	6 hours	9	BR31 Sport SS Extend	Red

Explanation: INNER JOIN function was applied to join following tables: 'Gov', 'Customer', 'RentaOrder', 'Order_Boat', and 'Boat'. The results display FirstName and LastName from Gov table, PhoneNumber, Email, OrderDate, RentalPeriod, NumberOfPeople from RentalOrder table, and BoatType and BoatColor from Boat table. The form shows us who placed the order, when the orders were created, and all the reservation details.

2. Email Customers who did not make their payments:

```
SELECT "Dear " &  g.FirstName & ": Your Order  #" & p.OrderID & "
is not completed because you have not made your payment at" &"
$ " &p.FinalPrice& ". Please complete your payment to reserve
your boat." AS Email_Message

FROM ((Gov g INNER JOIN Customer c
               ON g.GovernmentID = c.GovernmentID)
             INNER JOIN RentalOrder r
               ON r.CustomerID=c.CustomerID)
             INNER JOIN Payment p
               ON p.OrderID = r.OrderID
WHERE p.PaymentStatus ="Unpaid";
```

Email_Msg_For_Unpaid_Order

Email_Message
Dear Hyewon: Your Order #9903 is not completed because you have not made your payment at $ 234. Please complete your payment to reserve your boat.
Dear Chaeyeon: Your Order #9905 is not completed because you have not made your payment at $ 378. Please complete your payment to reserve your boat.
Dear Nako: Your Order #9908 is not completed because you have not made your payment at $ 378. Please complete your payment to reserve your boat.

Explanation: Some customers have not completed their payment. Therefore, their boat reservations cannot be confirmed. We want to send these customers an customized email message to remind them of completing their payments.

3. Customers who reside in Midwest:

```
SELECT FirstName, LastName, State
FROM Gov
WHERE State IN ('IL','WY','ID','WI','CO');
```

Midwest Customer

FirstName	LastName	State
Eunbi	Kwon	IL
Hyewon	Kang	ID
Chaeyeon	Lee	CO
Minjoo	Kim	WY
Nako	Yabuki	WI
Hitomi	Honda	IL
Yuri	Jo	CO

Explanation: We sorted out the customers who live in Midwest: IL, WY, ID, WI, CO.

4. Number of boats needed for each order and number of available boats left after reservation:

```
SELECT r.OrderID, r.NumberOfPeople, b.BoatCapacity,
```

```
        Format( (r.NumberOfPeople/b.BoatCapacity) , "0")
        AS Number_Of_Boat_Needed,
    Format(b.BoatQuantity-(r.NumberOfPeople/b.BoatCapacity), "0")
        AS Number_Of_Boat_Left
FROM (RentalOrder r INNER JOIN Order_Boat ob
                        ON ob.OrderID =r.OrderID)
                    INNER JOIN Boat b
                        ON b.BoatID = ob.BoatID
```

Number of Boat Reserve/ Left

OrderID	NumberOfPeople	BoatType	BoatCapacity	Number_Of_Boat_Needed	Number_Of_Boat_Left
9901	3	BR10 Racer Extended	5	1	1
9902	4	BR10 Racer Extended	5	1	1
9903	6	BR30 Sport SS Extend	8	1	2
9904	8	BR11 Racer Extended	10	1	2
9905	5	BR32 Sport SS Conver	6	1	2
9906	6	BR11 Racer Extended	10	1	2
9907	4	BR20 Fishing Walk Th	5	1	2
9908	8	BR31 Sport SS Extend	12	1	3
9909	11	BR41 Ultra Fish & Cr	12	1	3
9910	9	BR31 Sport SS Extend	12	1	3

Explanation: INNER JOIN function was applied to join following tables: 'RentalOrder', 'Order_Boat' and 'Boat'. We want to show the number of people going on the boat, the type of boats that they reserved, and the capacity of the boat. Knowing the number of people in each order, we want to know the number of boats needed to accommodate these people. Number of people divides by boat capacity equals to the number of boats needed for each order. Then we use FORMAT function to round the result to integer. BoatQuantity substracts Number of boat needed (BoatCapacity/NumberOfPeople) for each order equals to the number of available boats left. The FORMAT function is also used to round the final results into integer.

5. Order Assisted By Manager:

```
SELECT r.OrderID, g.FirstName, g.LastName, s.StaffMemberRole
AS      Assisted_By
FROM    ((Gov g INNER JOIN Customer c
```

```
                    ON g.GovernmentID = c.GovernmentID)
               INNER JOIN StaffMember s
                    ON s.CustomerID=c.CustomerID)
               INNER JOIN RentalOrder
                    ON c.CustomerID= r.CustomerID
WHERE s.StaffMemberRole ='Manager'
```

Order Assisted by Manager

OrderID	FirstName	LastName	Assisted_By
9902	Sakura	Miyawaki	Manager
9903	Hyewon	Kang	Manager

Explanation: Staffs will assist specific group of customers upon their arrivals. Staff roles are regular staffs, president, and managers. These two orders and corresponding customers will be assisted by staffs who are at manager level.

6. Revenue (Only includes paid payment):

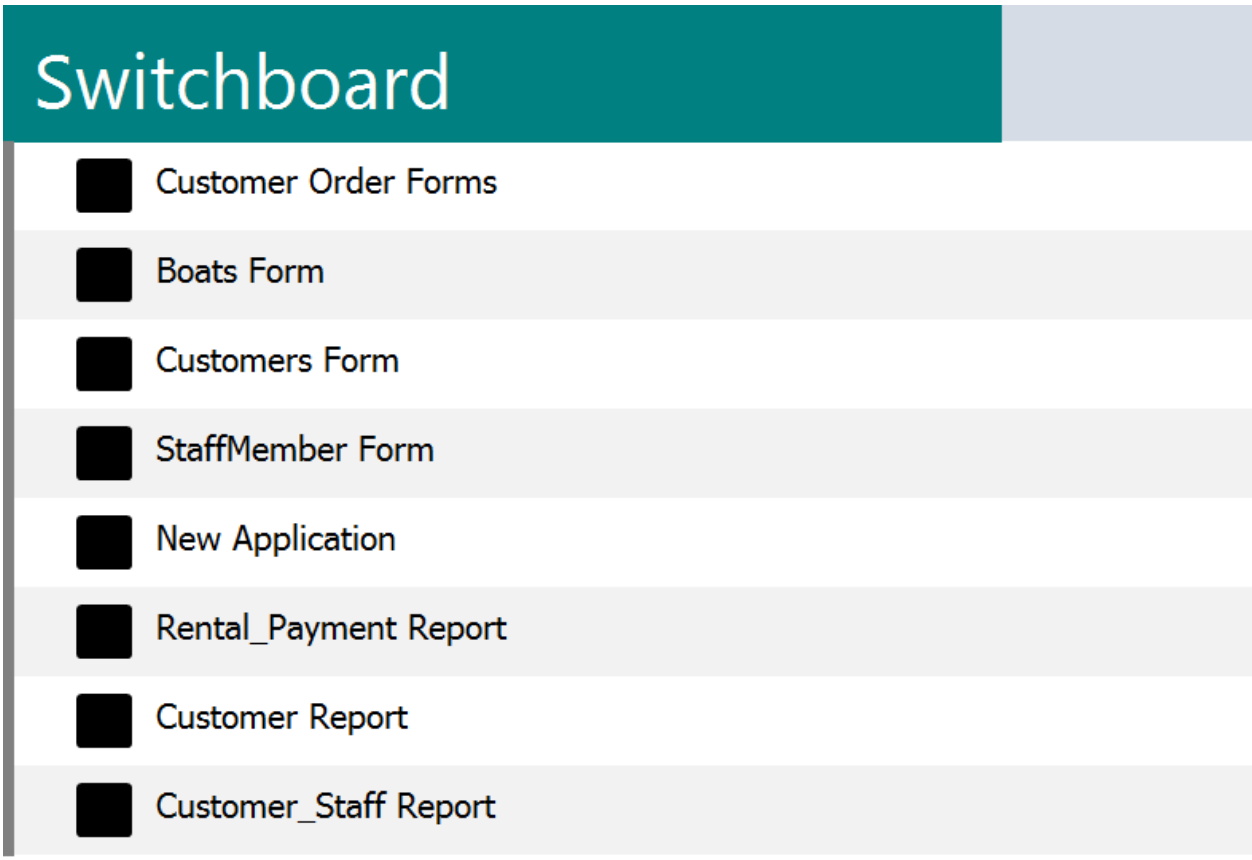
```
SELECT SUM(FinalPrice) AS Revenue
FROM Payment
WHERE PaymentStatus ="Paid"
```

Revenue

Revenue
3400

Explanation: SUM function was applied to sum the final prices of all orders as our revenue. We only sum the orders that are paid.

C. Switchboard



Note: Another shortcut to forms and reports. For Customer Order and Customers Form, this switchboard will navigate to a blank page to add a new record, while for the StaffMember and Boats Form will received an editing command. Reports will also be opened via this switchboard. User can also create a new application from here.

Conclusion

In this project, the difficult part was writing SQL and VBA codes. The codes were not working at the first time, but after several trials, the anticipated results were acquired. The easiest part was to build the conceptual model, having an ideal model that aligns the business goals. This part was less technical, so it is easier. Creating a navigation form is extremely useful because it would show all the reports and forms of the database like a content table; user can navigate the database easily with this form. The database will benefit the boat rental business because the owners are able to see the reservation details of the customers, payment status to remind some customers of completing the payments, staff's responsibility (the orders that are assisted by certain staff), and revenue of the month.