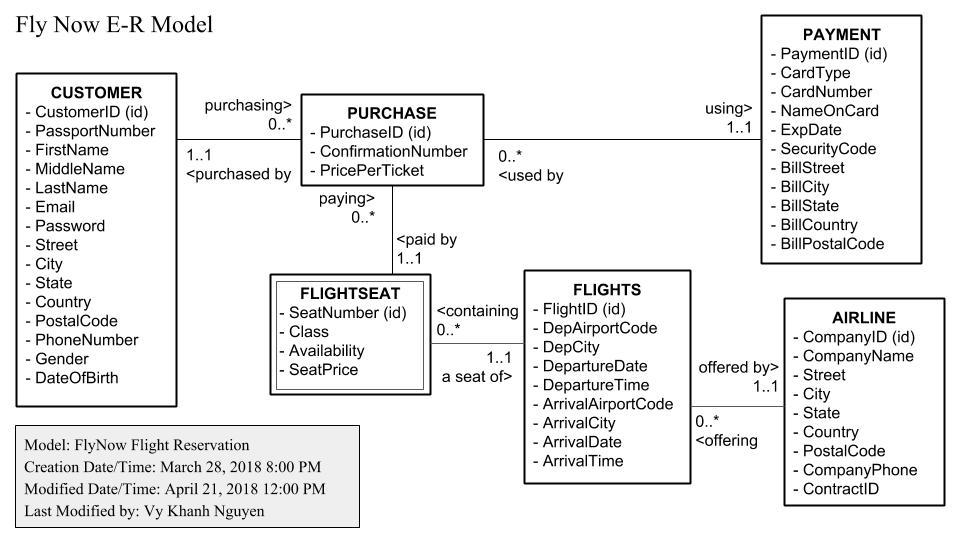
**INTRODUCTION**

Fly Now is a flight reservation online service based in New York, United States. Travelers in the world has been spending a lot of time finding the airplane tickets that best matched with their needs in terms of which airlines, prices, and time of flights they prefer. Our company offers them the shortcut by providing them the best options based on the information they searched for their trips on our site. We also allow customers to make purchase of their flights on our site once they have found a good deal available.

When a customer comes in our site, he will first make a search, which includes a location he will travel from, a destination he will travel to, date of departure, class of seat (first-class, business, economy), optional choice of price range, and an optional choice of airlines. Once a search is entered, a lists of all possible results will be displayed for our customers to choose and he will eventually select the one that fits his needs the most. Before our customer proceed to make a purchase with his chosen flight, we will promptly have him register a customer account that will ask for personal information (name, email, password, address, region of residence, phone number, gender, and date of birth). Once the customer account is created, a form asking for payment information will follow. Here, customer will be asked for his credit or debit card information that includes card number, name on card, expiration date, security code, and mailing address (street, city, state, country, and postal code).

After all the necessary information is entered into our site, our customer will be ready to make his purchase. The purchase will be recorded with a purchase ID, a confirmation number, and all the important flight information associated with it. Critical to our service, the data of airlines are constantly updated. For each airline, we need their two type of information. One is the airline’s information as an airline company with airline name, company address (street, city, state, postal code), company phone, and contacts with our site. The second type of information is their flight schedules in terms of place and time of the flights offered (departure, destination, departure date and time, arrival date and time, airline name, Flight ID, and seats capacity). For each flight on the schedule, its price varies according to the type of seat that customer choose, so we have seat information contain the type of seat, availability status, and its associated price for every flight.

**ENTITY RELATIONSHIP MODEL**



**LOGICAL MODELING**

Customer ( CustomerID (key), PassportNumber, FirstName, MiddleName, LastName, Email, Password, Street, City, State, Country, PostalCode, PhoneNumber, Gender, DateOfBirth )

Purchase ( PurchaseID (key), ConfirmationNumber, PricePerTicket, CustomerID (fk), PaymentID (fk) , FlightID (fk), SeatNumber (fk) )

Payment ( PaymentID (key), CardType , CardNumber, NameOnCard, ExpDate, SecurityCode, BillStreet, BillCity, BillState, BillCountry, BillPostalCode )

FlightSeat ( FlightID (key)(fk), SeatNumber (key) , Class, Availability, SeatPrice )

Flights ( FlightID (key), DepAirportCode, DepCity, DepartureDate, DepartureTime, ArrivalAirportCode, ArrivalCity, ArrivalDate, ArrivalTime , CompanyID (fk))

Airline ( CompanyID (key), CompanyName, Street, City, State, Country, PostalCode, CompanyPhone, ContractID )

**NORMALIZATION**

**Customer Relation**

Customer ( CustomerID (key), PassportNumber, FirstName, MiddleName, LastName, Email, Password, Street, City, State, Country, PostalCode, PhoneNumber, Gender, DateOfBirth )

Key: CustomerID

FD1: CustomerID → PassportNumber, FirstName, MiddleName, LastName, Email,

Password, Street, City, State, Country, PostalCode, PhoneNumber, Gender, DateOfBirth

FD2: Country, PostalCode → City, State

1NF: Meets the definition of a relation

2NF: No partial-key dependencies

3NF: Transitive dependency exists: CustomerID → Country, PostalCode → City, State

Solution: Split Customer relation into two new relations named Address and Customer

New relation:

Address ( Country, PostalCode, City, State )

Key: Country, PostalCode

FD1: Country, PostalCode → City, State

1NF: Meets the definition of a relation

2NF: No partial-key dependencies

3NF: No transitive dependencies

BCNF: All determinants are candidate keys

Customer ( CustomerID, PassportNumber, FirstName, MiddleName, LastName, Email, Password, Street, PhoneNumber, Gender, DateOfBirth, Address (fk) )

Key: CustomerID

FD1: CustomerID →  PassportNumber, FirstName, MiddleName, LastName, Email, Password, Street, PhoneNumber, Gender, DateOfBirth, Address (fk)

1NF: Meets the definition of a relation

2NF: No partial-key dependencies

3NF: No transitive dependencies

BCNF: All determinants are candidate keys

**Purchase Relation**

Purchase ( PurchaseID (key), ConfirmationNumber, PricePerTicket, CustomerID (fk) , PaymentID (fk) , FlightID (fk), SeatNumber (fk) )

Key: PurchaseID

FD1: PurchaseID → ConfirmationNumber, PricePerTicket, CustomerID, PaymentID, FlightID, SeatNumber

1NF: Meets the definition of a relation

2NF: No partial-key dependencies

3NF: No transitive dependencies

BCNF: All determinants are candidate keys

**Payment Relation**

Payment ( PaymentID (key), CardType , CardNumber, NameOnCard, ExpDate,

                  SecurityCode, BillStreet, BillCity, BillState, BillCountry, BillPostalCode )

Key: PaymentID

FD1: PaymentID → CardType, CardNumber, NameOnCard, ExpDate, SecurityCode, BillStreet, BillCity, BillState, BillCountry, BillPostalCode

1NF: Meets the definition of a relation

2NF: No partial-key dependencies

3NF: No transitive dependencies

BCNF: All determinants are candidate keys

We have de-normalized the Bill Address in this relation in order to make it much easier for us to implement in the database.

**FlightSeat Relation**

FlightSeat (  FlightID (key)(fk), SeatNumber (key) , Class, Availability, SeatPrice )

Key: FlightID, SeatNumber

FD1: FlightID, SeatNumber → Class, Availability, SeatPrice

1NF: Meets the definition of a relation

2NF: No partial-key dependencies

3NF: No transitive dependencies

BCNF: All determinants are candidate keys

**Flights Relation**

Flights ( FlightID (key), DepAirportCode, DepCity, DepartureDate, DepartureTime,

              ArrivalAirportCode, ArrivalCity, ArrivalDate, ArrivalTime , CompanyID (fk))

Key: FlightID

FD1: FlightID → DepAirportCode, DepCity, DepartureDate, DepartureTime, ArrivalAirportCode, ArrivalCity, ArrivalDate, ArrivalTime

1NF: Meets the definition of a relation

2NF: No partial-key dependencies

3NF: No transitive dependencies

BCNF: All determinants are candidate keys

**Airline Relation**

Airline ( CompanyID (key), CompanyName, Street, City, State, Country, PostalCode,

                  CompanyPhone, ContractID )

Key: CompanyID

FD1: ComapnyID → ComapnyName, Street, City, State, Country, PostalCode, CompanyPhone, ContractID

FD2: Country, PostalCode → City, State

1NF: Meets the definition of a relation

2NF: No partial-key dependencies

3NF: Transitive dependency exists: CompanyID → Country, PostalCode → City, State

Solution: Split Customer relation into two new relations named Address and Airline

New relations:

Address ( Country, PostalCode, City, State )

Key: Country, PostalCode

FD1: Country, PostalCode → City, State

1NF: Meets the definition of a relation

2NF: No partial-key dependencies

3NF: No transitive dependencies

BCNF: All determinants are candidate keys

Airline ( CompanyID, CompanyName, Street, CompanyPhone, ContractID,

              Address (fk)  )

Key: CompanyID

FD1: ComapnyID → CompanyName, Street, CompanyPhone, ContractID, Address (fk)

1NF: Meets the definition of a relation

2NF: No partial-key dependencies

3NF: No transitive dependencies

BCNF: All determinants are candidate keys

**Final Set of Relations**

Address ( Country (key), PostalCode (key), City, State )

Customer ( CustomerID (key), PassportNumber, FirstName, MiddleName, LastName, Email, Password, Street, PhoneNumber, Gender, DateOfBirth, Address (fk) )

Purchase ( PurchaseID (key), ConfirmationNumber, PricePerTicket, CustomerID (fk) , PaymentID (fk) , FlightID (fk), SeatNumber (fk) )

Payment ( PaymentID (key), CardType , CardNumber, NameOnCard, ExpDate, SecurityCode, BillStreet )

FlightSeat ( FlightID (key)(fk), SeatNumber (key) , Class, Availability, SeatPrice )

Flights ( FlightID (key), DepAirportCode, DepCity, DepartureDate, DepartureTime, ArrivalAirportCode, ArrivalCity, ArrivalDate, ArrivalTime , CompanyID (fk))

Airline ( CompanyID (key), CompanyName, Street, CompanyPhone, ContractID, Address (fk) )

We have de-normalized the Bill Address in this relation in order to make it much easier for us to implement in the database.

**SQL DDL**

CREATE TABLE ADDRESSES

(

Country VARCHAR(50) NOT NULL,

PostalCode VARCHAR(20) NOT NULL,

City VARCHAR(50),

State VARCHAR(50),

CONSTRAINT pk\_addresses PRIMARY KEY (Country, PostalCode)

)

CREATE TABLE CUSTOMER

(

CustomerID VARCHAR(10) NOT NULL,

PassportNumber VARCHAR(50),

FirstName VARCHAR(50),

MiddleName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

Password VARCHAR(15),

Street VARCHAR(100),

Country VARCHAR(50),

PostalCode VARCHAR(20),

PhoneNumber VARCHAR(15),

Gender VARCHAR(1),

DateOfBirth DATE,

CONSTRAINT pk\_customer

  PRIMARY KEY (CustomerID)

)

CREATE TABLE AIRLINE

(

CompanyID VARCHAR(10) NOT NULL,

CompanyName VARCHAR(100),

Street VARCHAR(100),

Country VARCHAR(50),

PostalCode VARCHAR(20),

CompanyPhone VARCHAR(15),

ContractID VARCHAR(25),

CONSTRAINT pk\_airline PRIMARY KEY (CompanyID)

)

CREATE TABLE FLIGHTS

(

FlightID VARCHAR(10) NOT NULL,

DepAirportCode VARCHAR(10),

DepCity VARCHAR(50),

DepartureDate DATE,

DepartureTime TIME,

ArrivalAirportCode VARCHAR(10),

ArrivalCity VARCHAR(50),

ArrivalDate DATE,

ArrivalTime TIME,

CompanyID VARCHAR(10),

CONSTRAINT pk\_flights PRIMARY KEY (FlightID)

)

CREATE TABLE FLIGHTSEAT

(

FlightID VARCHAR(10) NOT NULL,

SeatNumber VARCHAR(5) NOT NULL,

Class VARCHAR(15),

Availability BIT,

SeatPrice DOUBLE,

CONSTRAINT pk\_flightseat PRIMARY KEY (FlightID, SeatNumber)

)

CREATE TABLE PAYMENT

(

PaymentID VARCHAR(15) NOT NULL,

CardType VARCHAR(15),

CardNumber VARCHAR(20),

NameOnCard VARCHAR(50),

ExpDate DATE,

SecurityCode VARCHAR(4),

BillStreet VARCHAR(100),

BillCountry VARCHAR(50),

BillPostalCode VARCHAR(20),

CONSTRAINT pk\_payment

  PRIMARY KEY (PaymentID)

)

CREATE TABLE PURCHASE

(

PurchaseID VARCHAR(15) NOT NULL,

ConfirmationNumber VARCHAR(20),

PricePerTicket FLOAT,

CustomerID VARCHAR(10),

PaymentID   VARCHAR(15),

FlightID VARCHAR(10),

SeatNumber VARCHAR(5),

CONSTRAINT pk\_purchase PRIMARY KEY (PurchaseID)

)

**CONSTRAINTS**

ALTER TABLE CUSTOMER   ADD CONSTRAINT fk\_CustomerAddresses

FOREIGN KEY (Country, PostalCode)  REFERENCES ADDRESSES (Country, PostalCode);

ALTER TABLE AIRLINE    ADD CONSTRAINT fk\_AirlineAddresses

FOREIGN KEY (Country, PostalCode)   REFERENCES ADDRESSES (Country, PostalCode);

ALTER TABLE FLIGHTS ADD CONSTRAINT fk\_Airline FOREIGN KEY (CompanyID) REFERENCES AIRLINE (CompanyID);

ALTER TABLE FLIGHTSEAT ADD CONSTRAINT fk\_flights FOREIGN KEY (FlightID) REFERENCES FLIGHTS (FlightID);

ALTER TABLE PAYMENT    ADD CONSTRAINT fk\_BillingAddresses

 FOREIGN KEY (BillCountry, BillPostalCode)  REFERENCES ADDRESSES (Country, PostalCode);

ALTER TABLE PURCHASE ADD CONSTRAINT fk\_PurchaseCustomer FOREIGN KEY (CustomerID) REFERENCES CUSTOMER (CustomerID);

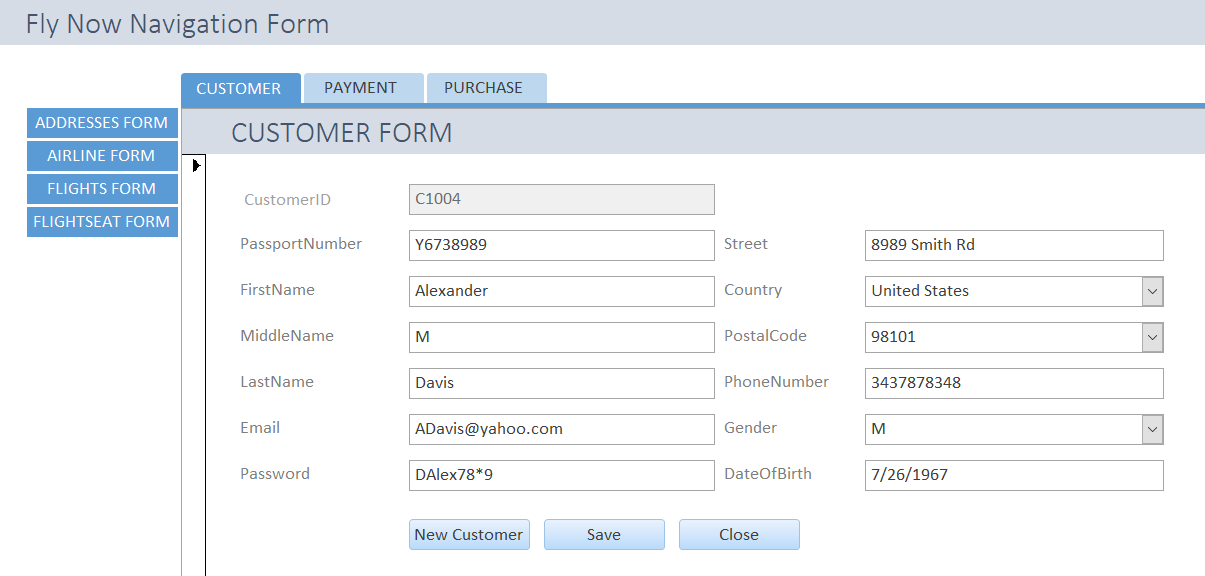
ALTER TABLE PURCHASE ADD CONSTRAINT fk\_PurchasePayment FOREIGN KEY (PaymentID)  REFERENCES PAYMENT (PaymentID);

ALTER TABLE PURCHASE ADD CONSTRAINT fk\_PurchaseFlightSeat FOREIGN KEY (FlightID, SeatNumber) REFERENCES FLIGHTSEAT (FlightID, SeatNumber);

**DATABASE IMPLEMENTATION**

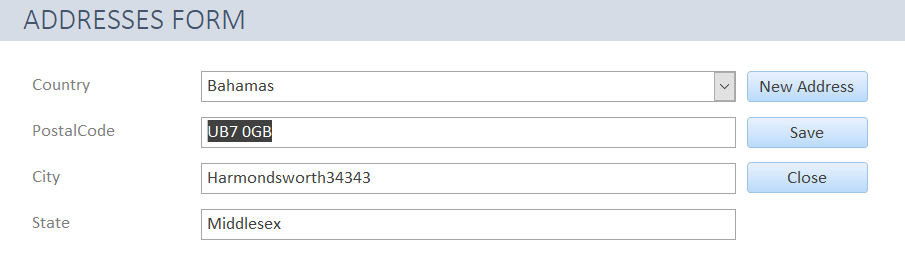
The Database Application consists of a set of Forms, Reports and Queries that are linked together on a Navigation Form. Fly Now Navigation Form is the first form that appears when the database is opened.

**Fly Now Navigation Form**



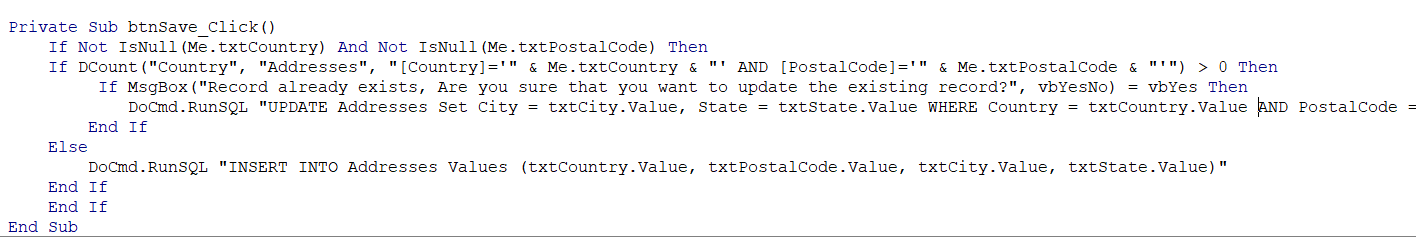
Different data entry forms and reports can be displayed by clicking on the selection on the tabs.

**Addresses Data Entry Form**

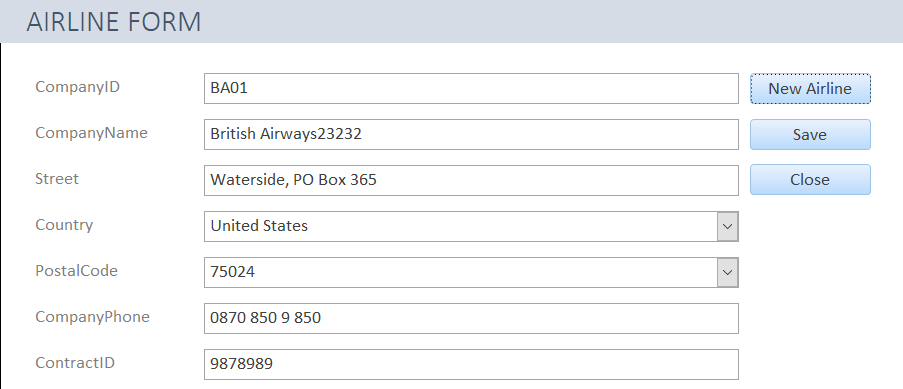


The Addresses Data Entry form is used to look up existing Addresses and to input new address information.

**VBA code**



**Airline Data Entry Form**

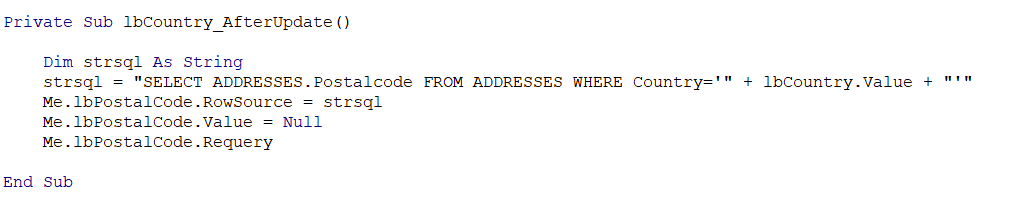


The Airline Data Entry Form is used to query, Update and add new Airline details.

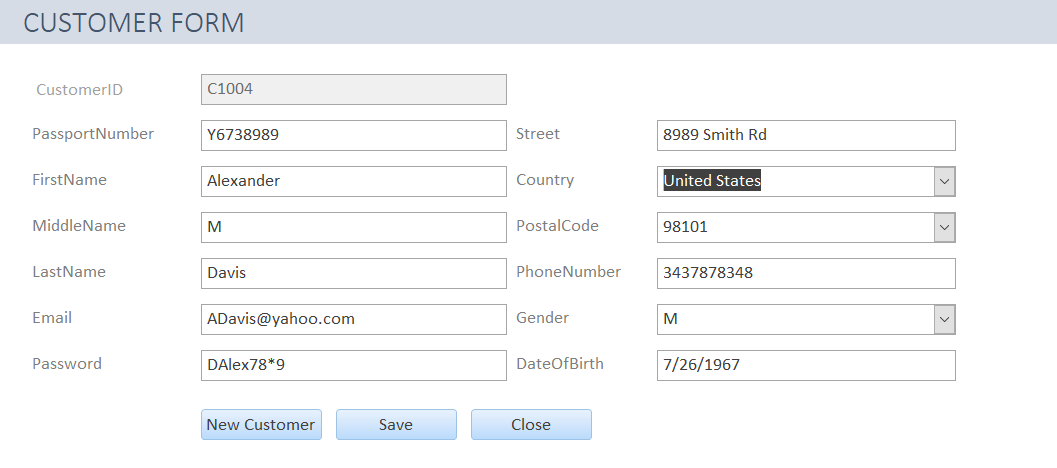
When a Country is selected from the Combo box, the PostalCode Combo box is automatically

filled in with the PostalCode’s mapped to the selected country from the addresses table.

**VBA Code**



**Customer Data Entry Form**

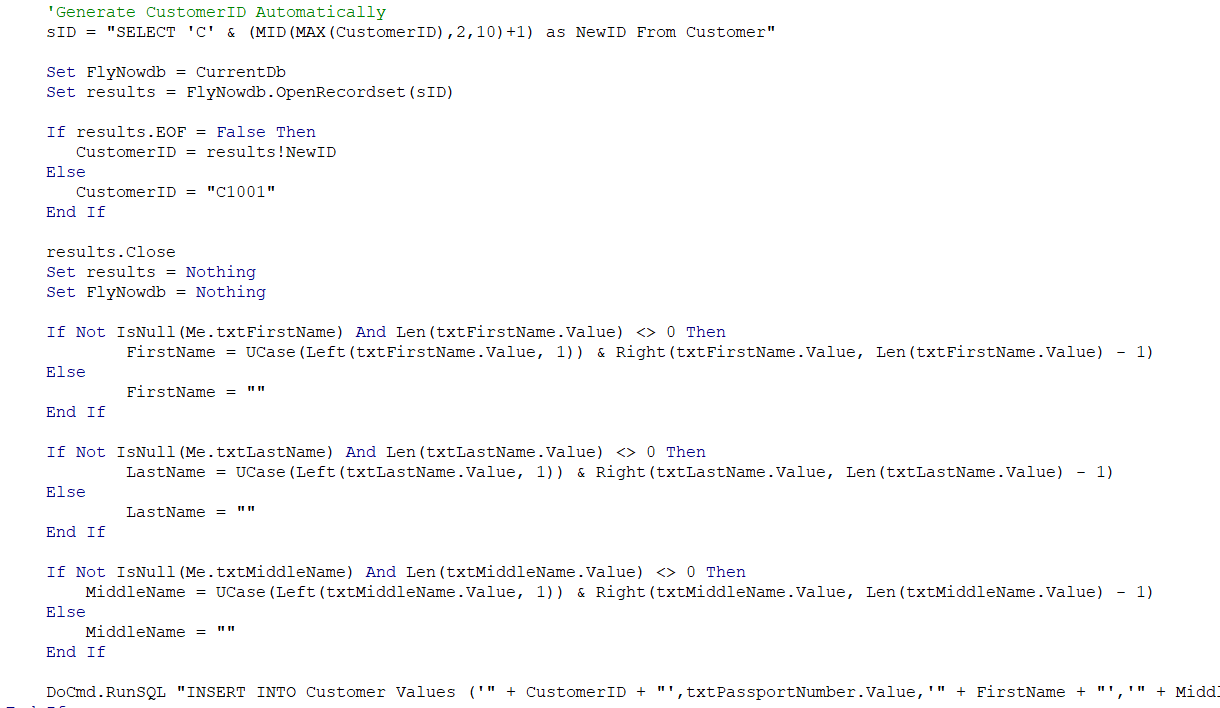


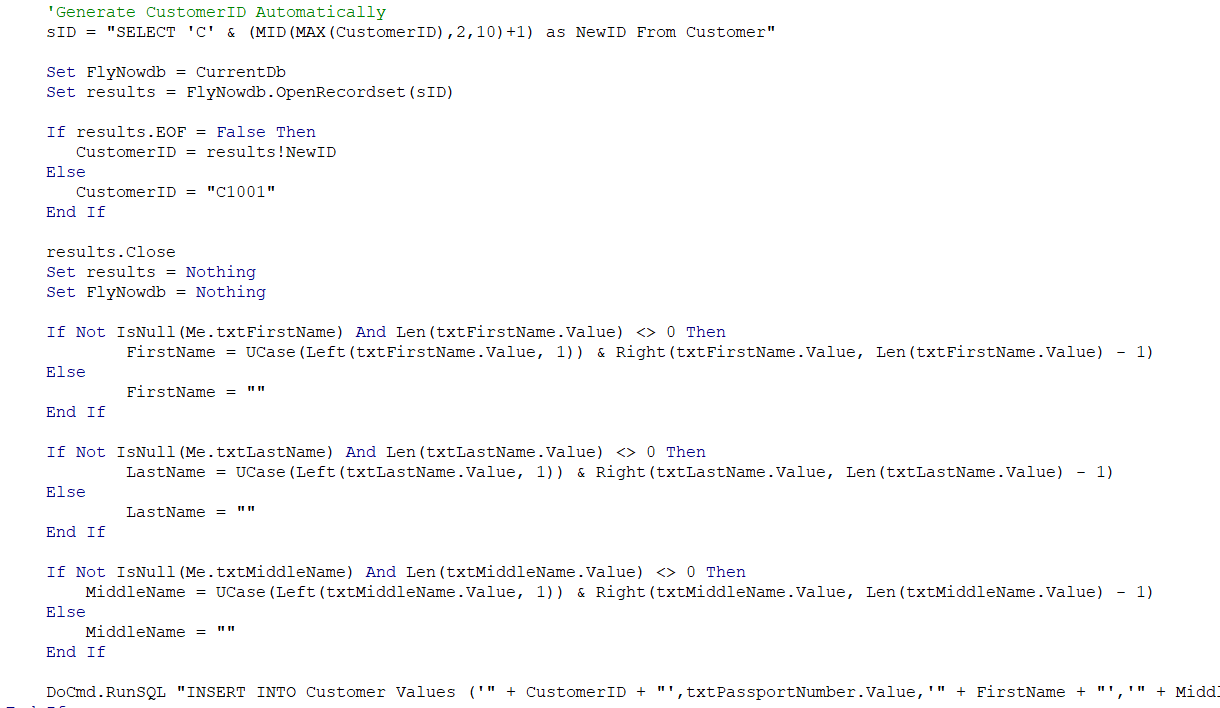
The Customer Data Entry Form is used to query, update and add new customer information.

When a Country is selected from the Combo box, the PostalCode Combo box is automatically filled in with the PostalCode’s mapped to the selected country from the addresses table.

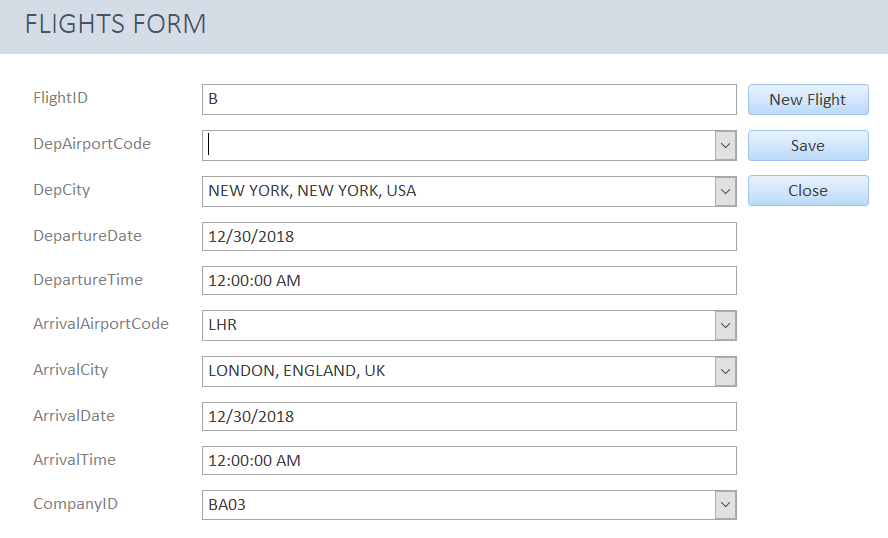
The form has custom code to convert the FirstName and LastName to proper case and to generate a new and unique customer ID.

**VBA Code**





**Flights Data Entry Form**

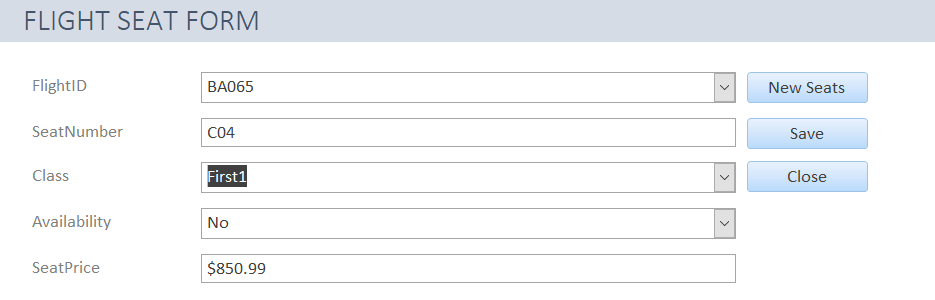


The Flights Data Entry Form is used to query, update and add new Flights information. Custom Queries are mapped to retrieve companyID information for Combo Box.

**SQL Used**

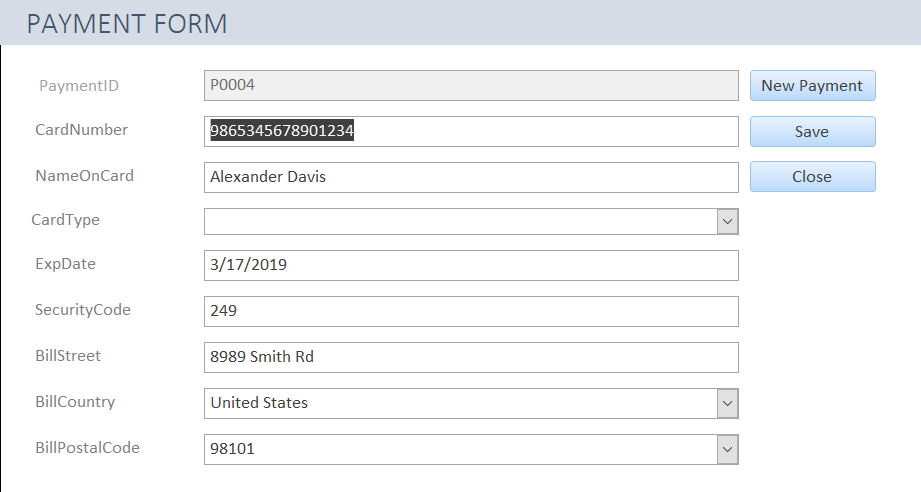
https://lh5.googleusercontent.com/q0Eq7MABoe2rA60Ig-HNYaREUcXiJjmGSChyiobol82LKJqxiRwXwyw_FvaJz4l_hg-ZdW8wAF2v4ADU6W5xbC30LjcKTYiU_7hs5dvTwdBhkjROQUyNban2319mPiLn7StL8yAYGOW3i6rjfw

**Flight Seat Data Entry Form**



The Flights Seat Data Entry Form is used to query, update and add Seat information in Flights.

**Payment Data Entry Form**

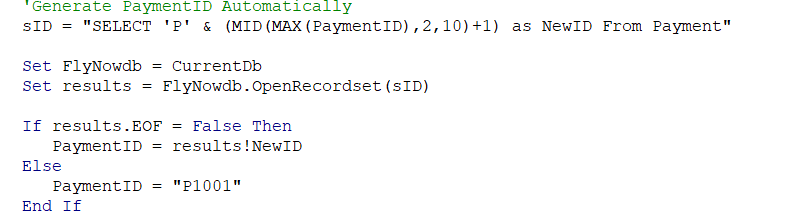


The Payment Data Entry Form is used to query, update and add new Payments information.

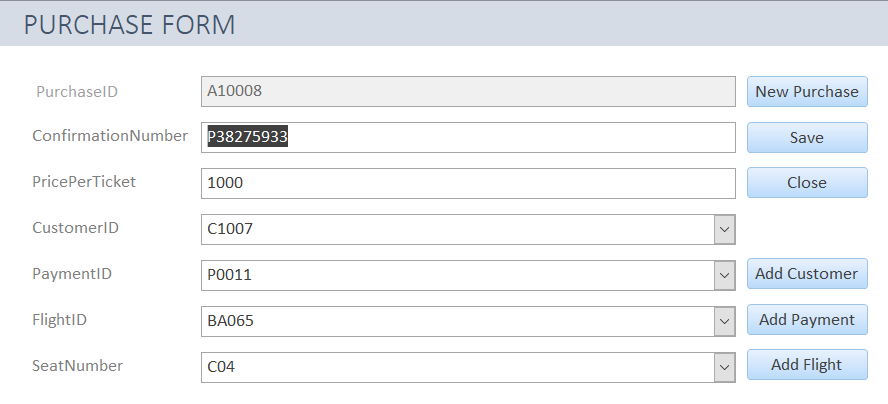
When a BillCountry is selected from the Combo box, the BillPostalCode Combo box is automatically filled in with the PostalCode’s mapped to the selected country from the addresses table.

The form has custom code to generate a new, unique PaymentID.

**VBA Code**

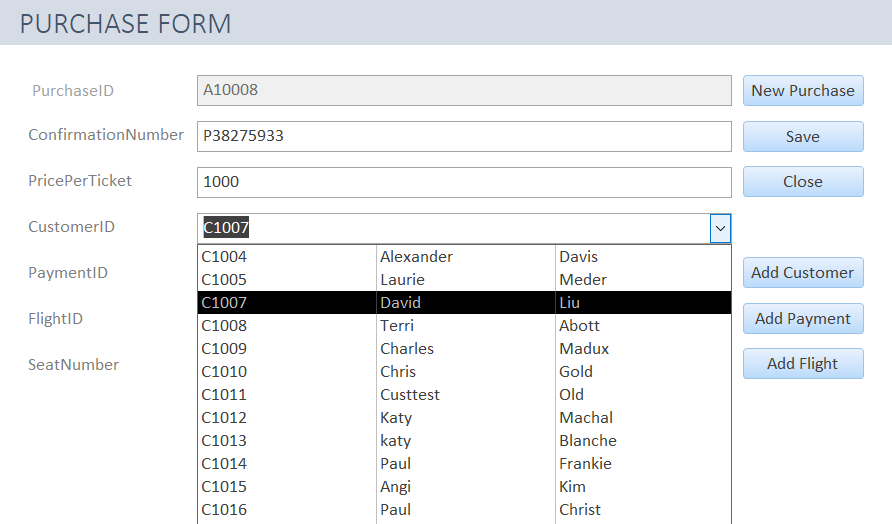


**Purchase Data Entry Form**



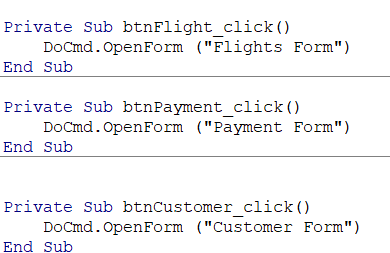
The Purchase Data Entry form is used to create a new purchase order for a customer.

The Customer can be selected from the Customer ID combo box as shown below:

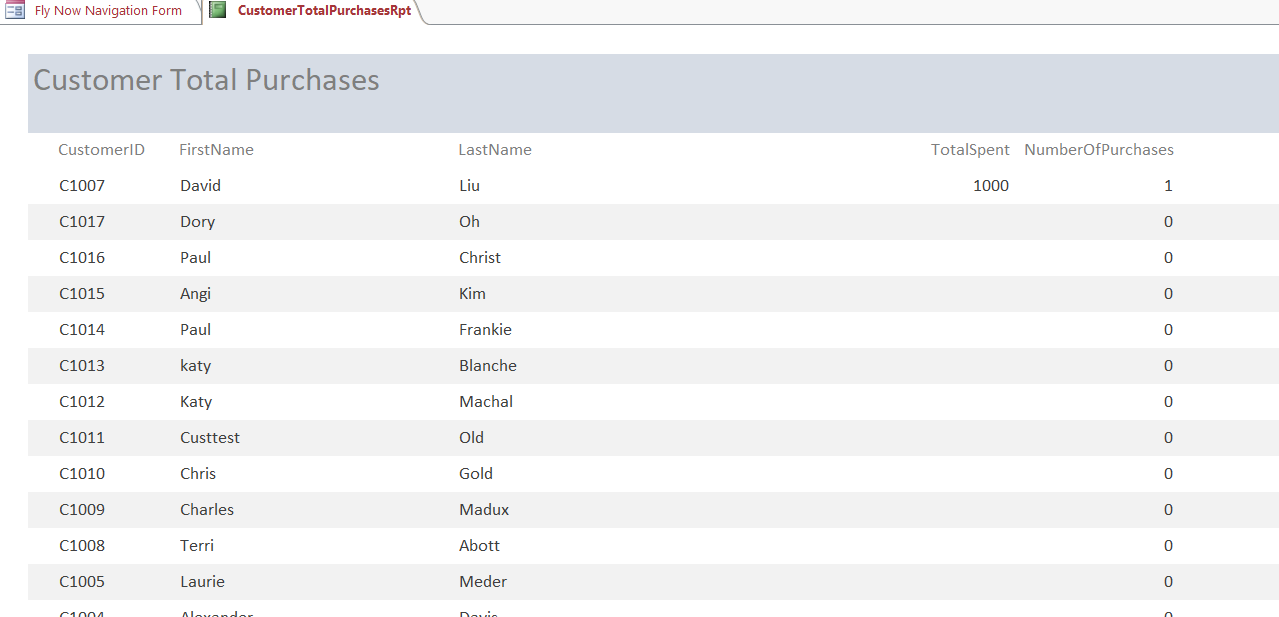


If this is a new customer purchasing a ticket, **Add Customer** button can be clicked to bring up the Customer Data Entry Form. After the new customer’s information is saved, user can return to the Purchase Data Entry Form and make the purchase. **Add Payment** and **Add Flight** button’s have similar feature to add new Payment and Flight information if not available.

**VBA Code**



**Customer Total Purchases Report**



This report provides a summary of purchases made and total spent by each customer.

**SQL Used:**

SELECT

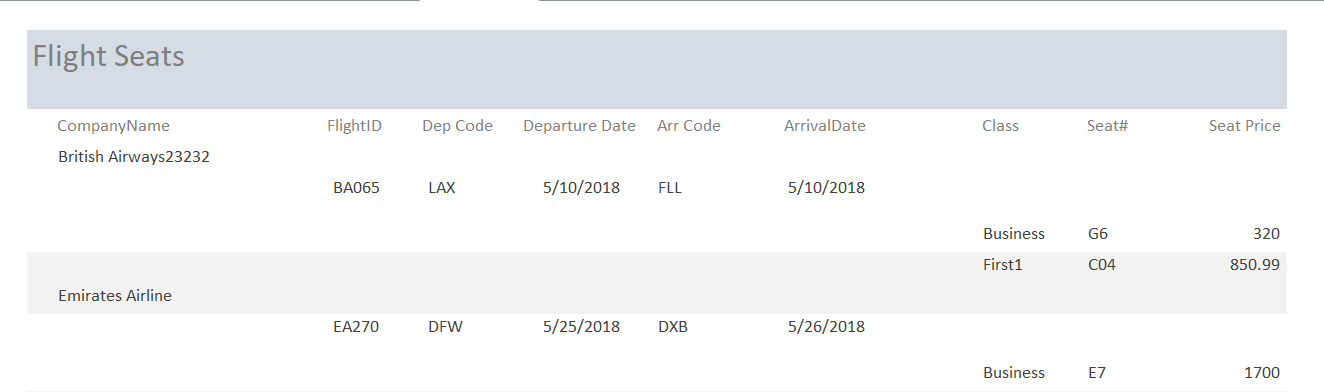
 C.CustomerID, C.FirstName, C.LastName, SUM(p.PricePerTicket) AS TotalSpent,

 COUNT(p.PurchaseId) AS NumberOfPurchases

FROM     Customer AS C LEFT JOIN Purchase AS P ON C.CustomerID = P.CustomerID

GROUP BY C.CustomerID, C.FirstName, C.LastName

ORDER BY SUM(p.PricePerTicket) DESC;

**Flight Seats Report**

This report provides a summary of Flight Seats info in the system for each and every airline and the associated flights.

**SQL Used**

SELECT

AIRLINE.CompanyName, FLIGHTS.FlightID, FLIGHTS.DepAirportCode,

FLIGHTS.DepartureDate, FLIGHTS.ArrivalAirportCode, FLIGHTS.ArrivalDate, FLIGHTSEAT.Class,

FLIGHTSEAT.SeatNumber, FLIGHTSEAT.SeatPrice

FROM   (AIRLINE INNER JOIN FLIGHTS ON AIRLINE.CompanyID = FLIGHTS.CompanyID)

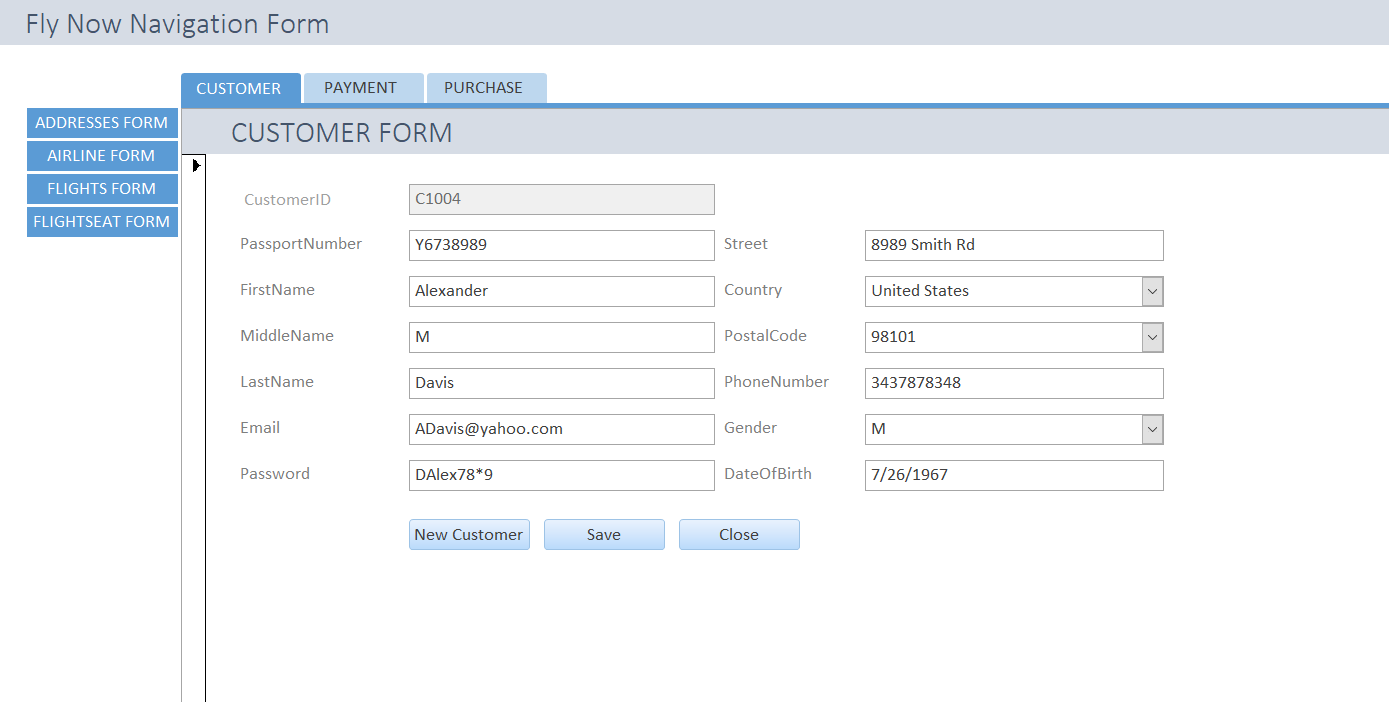
INNER JOIN FLIGHTSEAT ON FLIGHTS.FlightID = FLIGHTSEAT.FlightID

GROUP BY   AIRLINE.CompanyName, FLIGHTS.FlightID, FLIGHTS.DepAirportCode,

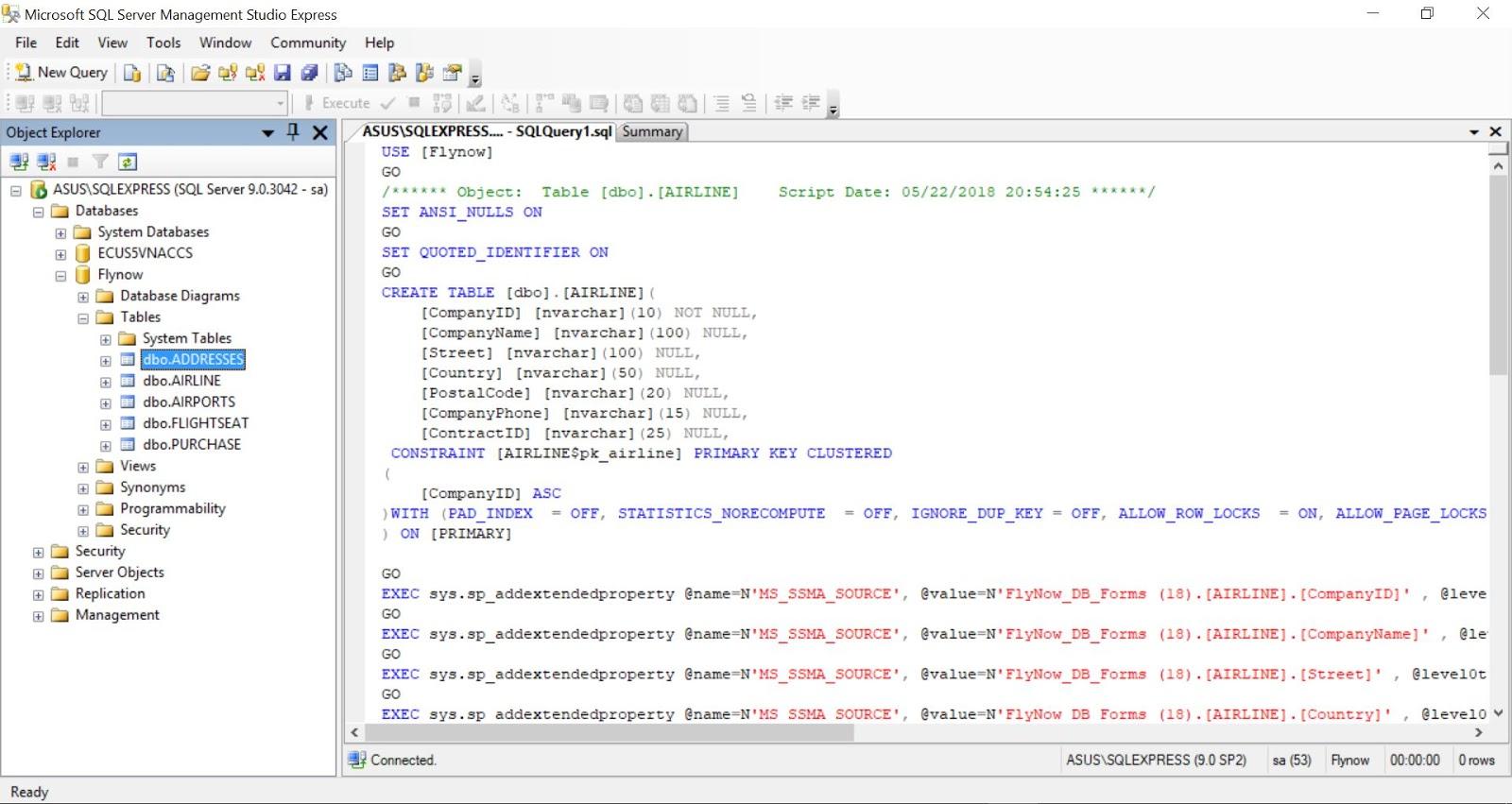
  FLIGHTS.DepartureDate, FLIGHTS.ArrivalAirportCode, FLIGHTS.ArrivalDate,

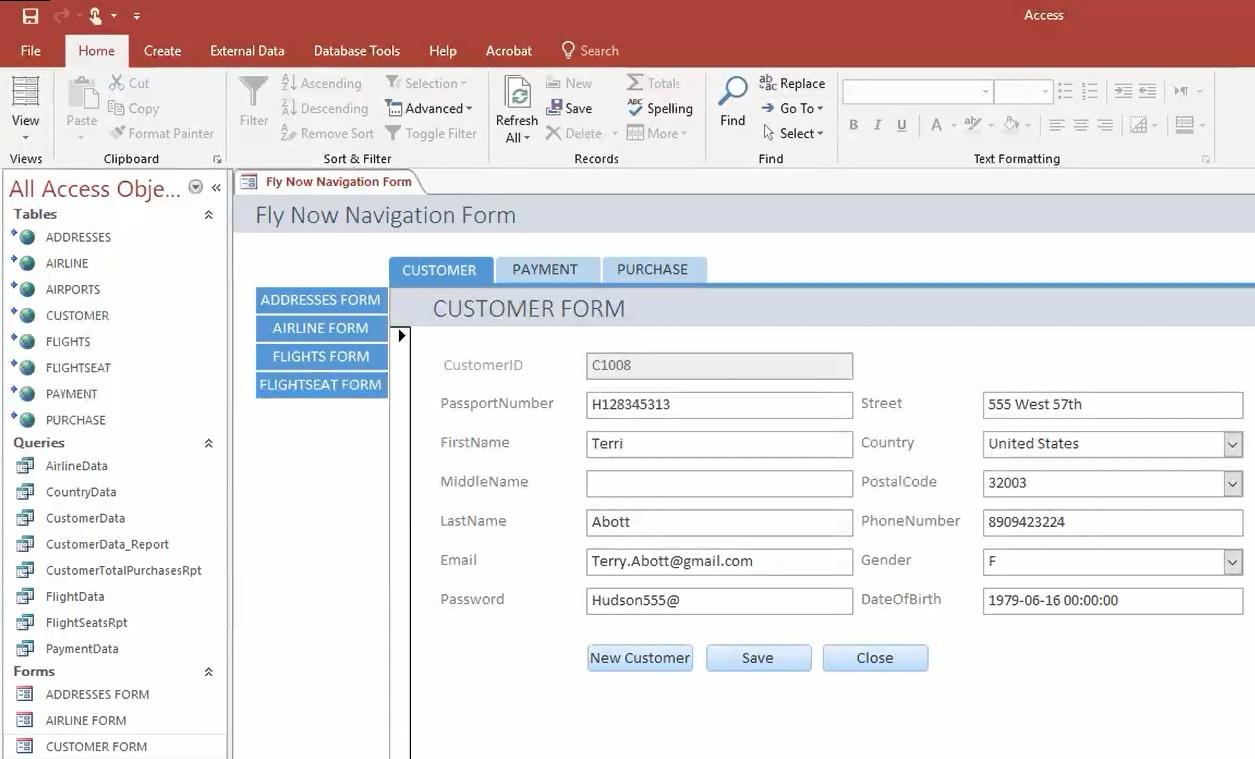
  FLIGHTSEAT.Class, FLIGHTSEAT.SeatNumber, FLIGHTSEAT.SeatPrice;

**NAVIGATION FORM**



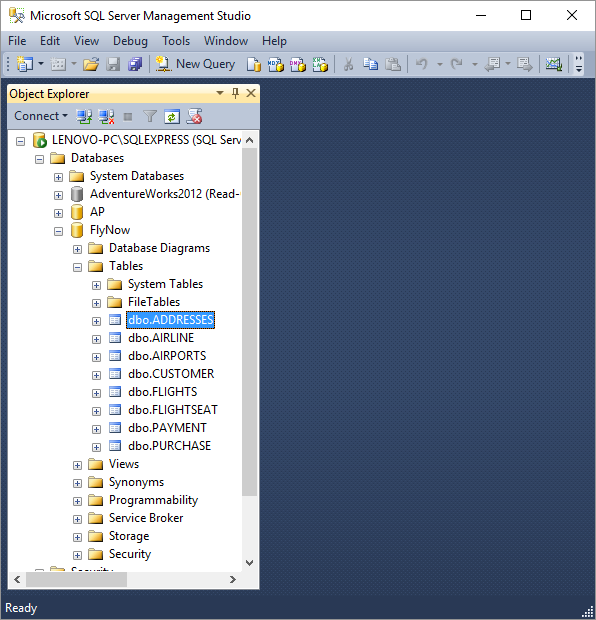
**EXTRA CREDIT**

****

****

**(Myungseon) Migration a Microsoft Access database to SQL server**

**Software Specification**   
Microsoft SQL Server Migration Assistant v7.8 for Access  
Microsoft Access 2013 SP1  
Microsoft Access Runtime 2012  
Microsoft SQL server 2012

**After Migration in Microsoft SQL Server** 

**After Migration in Microsoft Access**

